REMOTE CONTROL FOR PC

By

ASRUL NIZAR BIN AHMAD

FINAL REPORT

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

Universiti Teknologi Petronas
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

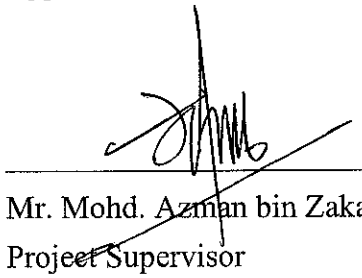
REMOTE CONTROL FOR PC

by

Asrul Nizar bin Ahmad

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

Approved:



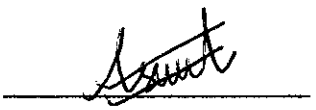
Mr. Mohd. Azman bin Zakariya
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK

December 2005

CERTIFICATION OF ORIGINALITY

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



Asrul Nizar bin Ahmad

ABSTRACT

The system of remote control for PC available in the market is usually found at a high cost. The project is carried out in delivering a low cost PC remote control system by only designing the receiver. As for the transmitter, the available TV remote control can be used in integration with the system. The primary objective is to come out with an infrared receiver which is able to detect the signals received and send the signal into the PC through serial communication port. The infrared receiver is designed based on PIC16F84A microcontroller. The infrared receiver is embedded to enable the receiver in sensing the presence of infrared signals. The cores of the project are the applications of electronic fundamentals in circuit design and software development using C in delivering the receiver functionality. The graphical user interface is designed using Microsoft Visual Basic as to provide the means of interfacing between the user and the PC through the utilization of infrared remote control system. The project methodology stresses on the research and analysis, circuit design, circuit modeling, microcontroller programming, circuit fabrication and finally, designing the interfacing software. The end system comprises of the hardware for receiver, universal remote control and interfacing software, which can be utilized in controlling several PC applications.

ACKNOWLEDGEMENTS

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

In the name of Allah, The Beneficent, The Merciful.

Alhamdulillah, all praises to Him that I have been able to complete the two-semester final year project of remote control for PC.

Highest appreciation and sincere gratitude regarded to the project supervisor, Mr. Mohd. Azman bin Zakariya, for the guidance and attention in helping me to come out with a device of PC remote control. All comments given had urged me to struggle hard in fulfilling the required deliverables of the project.

Utmost thanks to the laboratory technologist, Miss Siti Hawa Mohd Tahir, for the cooperation given in utilizing the related laboratory equipments while progressing up with the project. In addition, thanks to all my peers, for the continuous support and willingness to share their ideas regarding this project.

Thank you to all of you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

1. IR - Infrared
2. RF - Radio Frequency
3. IC - Integrated Circuit
4. PIC - Peripheral Interface Controller
5. CPU - Central Processing Unit
6. ROM - Read Only Memory
7. RAM - Random Access Memory
8. I/O - Input/ Output
9. DTE - Data Terminal Equipment
10. DCE - Data Communication Equipment
11. PCB - Printed Circuit Board
12. RISC - Reduced Instruction Set Computer
13. EPROM - Erasable Programmable Read Only Memory
14. A/D - Analog-to-Digital
15. MSSP - Master Synchronous Serial Port
16. USART - Universal Synchronous Asynchronous Receiver Transmitter
17. PSP - Parallel Slave Port
18. USB - Universal Serial Port
19. ISR - Interrupt Service Routine
20. DIP - Dual-in Package
21. SMD - Surface Mount Device
22. I^2C - Inter-Integrated Circuit
23. OSC - Oscillator
24. GUI - Graphical User Interface
25. Hz - Hertz

CHAPTER 1
INTRODUCTION

1.1 Background of Study

A remote control is an electronic device used for the remote operation of a machine. It is mostly used for televisions or other consumer electronics such as stereo systems and DVD players, with the common purpose of turning on and off the main plug. Remote control for these devices is usually a small handheld object with an array of buttons for adjusting various settings such as television channel, track number and volume. Most of these remotes communicate to their respective devices via infrared signals or few via radio signals. They are usually powered by small AA or AAA size batteries [1]. **Figure 1** shows varieties of remote control available in the market.

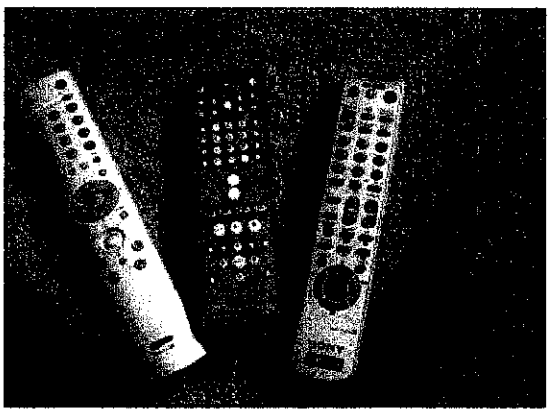


Figure 1 Varieties of remote control [2]

In order for PC to receive Infrared signals sent from a remote control, the PC should have an Infrared receiver attached to its system. Otherwise, the signals sent cannot be retrieved. The manner at which the PC reads the signals received is the similar to the way that the television received the signals sent from its remote control except that, serial or parallel communication is used to enable the communication between the Infrared receiver and the PC.

The communications links across which computers or parts of computers talk to one another may be either serial or parallel. Parallel link transmits several streams of data along multiple channels, where all the bits of each symbol are sent together.

Whereby, serial link transmits a single stream of data. The data are sent one bit at a time, sequentially, over a communications channel or computer bus. In many cases, serial is a better option because it is cheaper to be implemented. The costs of cable and synchronization difficulties make parallel communications become impractical. Many ICs have serial interfaces, as opposed to parallel ones, so that they have fewer pins. **Figure 2** shows the 25-pin and 9-pin serial connectors.

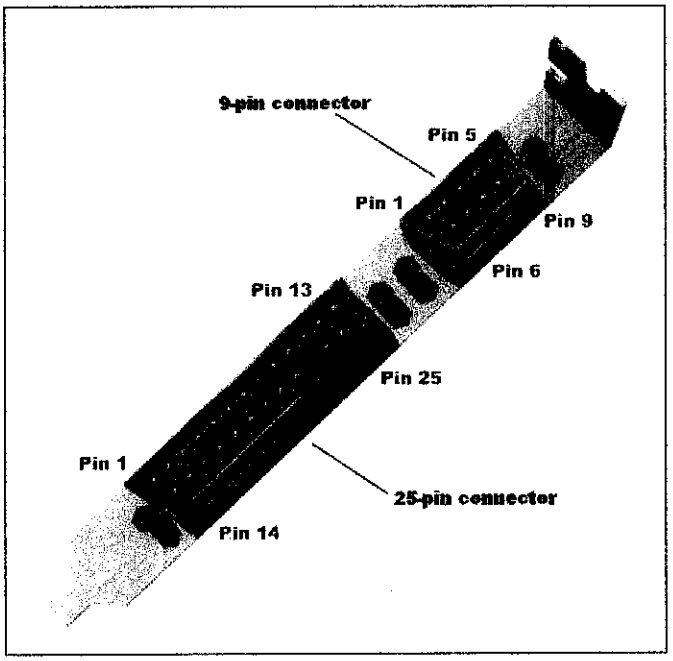


Figure 2 25-pin (bottom left) and 9-pin serial connectors(upper right) [3]

Nowadays, there are three common ways of communicating wirelessly; Infrared, radio frequency (RF) and Bluetooth. These means are applied according to their advantages and practicality in electronic devices. Infrared (IR) radiation is an electromagnetic radiation of a wavelength longer than visible light, but shorter than microwave radiation. The name means "below red", red being the color of visible light of longest wavelength. Infrared radiation spans three orders of magnitude and has wavelengths between 700 nm and 1 mm [4]. Infrared transmission requires that devices be aimed at each other (line of sight) in order for transmission to occur.

Radio frequency, or RF, refers to that portion of the electromagnetic spectrum in which electromagnetic waves can be generated by alternating current fed to an

antenna. Such frequencies account for the respective parts of the spectrum [5]. Bluetooth is a wireless radio standard primarily designed for low power consumption, with a short range (up to 10 meters and with a low-cost transceiver microchip in each device. Bluetooth uses omnidirectional radio waves that can transmit through walls and other non-metal barriers. If there is interference from other devices, the transmission does not stop, but its speed is downgraded [6].

A microcontroller has played a pervasive role in this modern era. The emphasis is upon the integration of variables features in a single chip, thus makes the characteristics of a microcontroller enable to sense the inputs and deliver the required outputs of a device. The microcontroller is usually found in photocopy machine and air-conditioner to control their applications.

1.2 Problem Statement

The project is emphasizing on delivering a complete system of a remote control for PC. For the hardware part, only an Infrared receiver is designed and fabricated. As the microcontroller is integrated into the circuit, the programming of PIC16F84A is required to enable the system to receive and retrieve the incoming infrared signals. At the end, the highlight is given to the making of interfacing software for the system. The deliverables of the project are represented in the diagram of **Figure 3**.

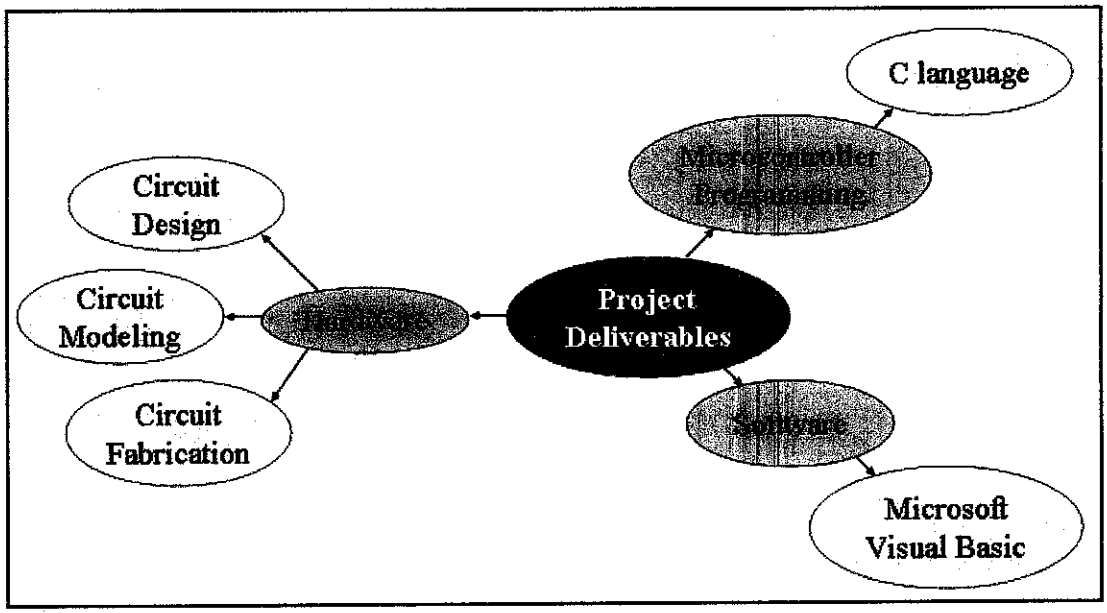


Figure 3 Deliverables of the project

1.2.1 Problem Identification

The PC remote control system available in the market comprises of both remote control and receiver. Due to this reason, the price of the system is quite high. Furthermore, user might think it is not really worth the money to purchase the system but the utilizations are minimal. In this project, a receiver, which is capable to receive signals sent from various remote controls, is designed.

The receiver circuit is first modeled on the project board as to ensure there will be no faulty made in the design. Besides, if any additional components required to be embedded on the circuit, they can be identified and easily added before the finalized circuit is soldered on the Vero board. In producing more than one unit of the receiver, PCB is fabricated once all regarding tasks have been accomplished.

Programming the microcontroller is the most critical part of the project. It is essential in enabling the system to read the infrared signal received before sending the signal through the serial communication of a PC. The thorough study and good understanding on C programming are necessary as to guarantee the smoothness while programming the microcontroller.

Designing the interfacing software is another important stage of the project. Familiarization with the Microsoft Visual Basic software is necessary as to ensure that the software designed meets the users' requirements and can be utilized as desired.

1.2.2 Significance of the Project

As described previously, only Infrared receiver is designed for the project. User can use the available remote control at their house or purchase a very cheap universal remote control in order to make use of the system. **Figure 4** of the following page shows the block diagram to represent the basic operation of the PC remote control system, from the remote control as the transmitter to the receiver connected to the PC.

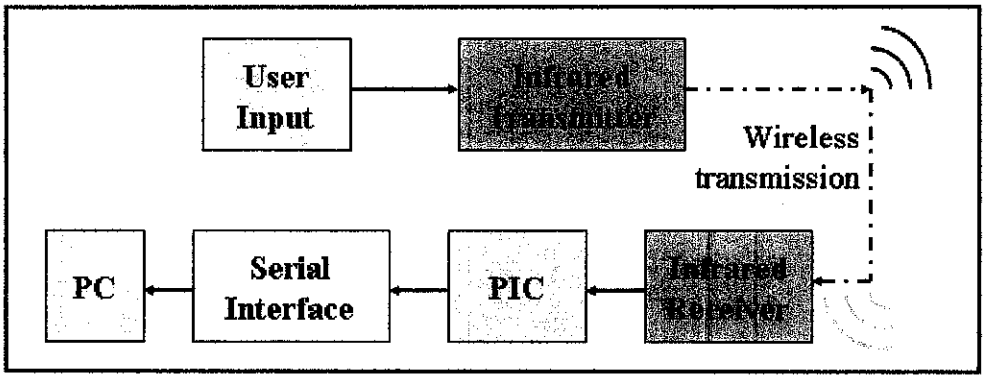


Figure 4 Basic block diagram for remote control system

There is no battery required to power up the receiver circuit since its +5V supply comes from the USB port of the computer. Thus, there are two cables connected from the receiver circuit into the PC, one is for serial port and the other one is for USB port. The project offers improvement to the existing PC remote control system in order to allow better control from the users. The controllability is not limited only to the Multimedia applications but also include the professional application, which is relevant to be controlled wirelessly.

1.3 Objective and Scope of Study

The foremost objective of the project is to design an infrared receiver that can be utilized interactively by users in controlling the PC applications. The system should be cost-effective as to allow utilizations from wide range of users.

Another objective is to enhance the design and performance of the existing design. The project is differentiated with the existing PC remote control system by using the USB port for +5V supply. Moreover, the users are freed to use several types of remote control available in the market.

The project coverage also includes the programming knowledge of the microcontroller. With this, the knowledge enhancement on C language to program the microcontroller is considered as another main objective of the project.

Additionally, the knowledge in developing an interfacing software using Microsoft Visual Basic can be acquired. A software subsystem is developed for the user to link and control the PC applications such as Power Point Presentation, Winamp and Windows Media Player. At the end, the system should be user-friendly and reliable.

1.3.1 The Relevancy of the Project

The relevancy of the project is viewed from three different perspectives, by which include hardware design, PC applications and remote control applications.

1.3.1.1 Hardware Design

The microprocessor is facilitated in the Infrared receiver circuit as an alternative to having a large number of electronics gates on the circuit. The reduction or elimination of gates is necessary to reduce the component count and the circuit size. Furthermore, this can ensure that the final product will be medium-sized and compact.

1.3.1.2 PC Application

With PC remote control system, user can control PC application wirelessly within some distance. The highlights are given on the multimedia applications such as Windows Media Player and Winamp. Sometimes, user might watch movie from the bed. If the user needs to pause or adjust the volume of the movie, the task can just be done from the bed without getting near to the PC.

Another example can be taken from the situation faced by the lecturer. While giving the lecture presentation, many lecturers tend to move front and back. But, when they need to change the slide, they have to walk to the PC just to change to the next slide. If the PC remote control system is being utilized, the slide can just be changed from the current location as long as it is within the specified range.

1.3.1.3 Remote Control Application

Several types of remote control available at home can be utilized for the system. Thus, the remote control can be used bi-functionally, for television and PC. The application of external remote control into the system is considered reasonable since the invention of another remote control circuit may increase the cost and size of overall system. Furthermore, the arrays of button on a remote control can be assigned for various commands in controlling the PC applications.

1.3.2 Feasibility of the Project within the Scope and Time Frame

In general, all scopes covered to complete the project are feasible for a final year student. The allocated time frame of approximately one year is sufficient to carry out the entire task required in the project. **Appendix A** summarized the allocated time frames for all tasks performed throughout the two semesters in a Gantt chart. There are three fundamental parts contributing to the major accomplishment of the project; hardware design, microcontroller programming and software design.

1.3.2.1 Hardware Design

The scopes of study covered in the hardware design include the existing remote control and specifically, the PC remote control. The fundamentals of infrared devices and how they play their role in data transmission of remote control system are also revised. The basics of electronics learned during early years of study are reapplied in producing the receiver circuit. The time frame for hardware design is during the first semester of the project.

1.3.2.2 Microcontroller Programming

The knowledge on C++ programming learned previously had given some advantages in quick grasping of C language used in the programming part. As this is the crucial part of the project, microcontroller programming is carried out during the first half of the second semester, once the receiver circuit is finalized.

1.3.2.3 Software Design

The new scope covered in producing interfacing software using Microsoft Visual Basic requires self-learned since it never been officially learned previously. In ensuring the feasibility of the scope, the related text books and existing programs are studied thoroughly. Half of the second semester is consumed to come out with the new software.

CHAPTER 2
LITERATURE REVIEW/ THEORY

2.1 PIC16F84A Microcontroller

2.1.1 Introduction

The PIC16F84A employs an advance Reduced Instruction Set Computer (RISC). This microcontroller had enhanced core features, eight-level deep stack, and multiple internal and external interrupt sources. The separate instruction and data buses of the Harvard architecture allow a 14-bit wide instruction word with a separate 8-bit wide data bus. The two stage instruction pipeline allows all instruction to execute in a single cycle except for program branches which is required two cycles. A total 35 instruction are available with an additions of large register set is used to achieve a very high performance level.

2.1.2 Applications

The PIC16F84 fits perfectly in application ranging from high speed automotive and appliance motor control to low-power remote sensors, electronic locks, security devices and smart cards. The Flash/EEPROM technology makes customization of application programs acted extremely fast and convenient.

This chip is also low-cost, low power, high performance, easy to use and I/O flexibility make it very versatile even in areas where no microcontroller use has been considered before. The serial in-system programming features (via two pins) offer flexibility of customizing the product after complete assembling and testing. This feature can be used to serialize a product, store calibration data or program the devices with the current firmware before shipping.

2.1.3 Clock/ Instruction Cycles

Clock is a microcontroller's main starter, and is obtained from an external component called an "oscillator". The clock input from OSC1 is internally divided by four to generate four non-overlapping quadrature clocks namely Q1, Q2, Q3, and Q4. Internally, the instruction is fetched from the program memory and latched into instruction register in Q4. The instruction is decoded and executed during the following Q1 through Q4. An "Instruction Cycle" consists of four Q cycles. The clocks and instruction execution flow is show in **Figure 5**.

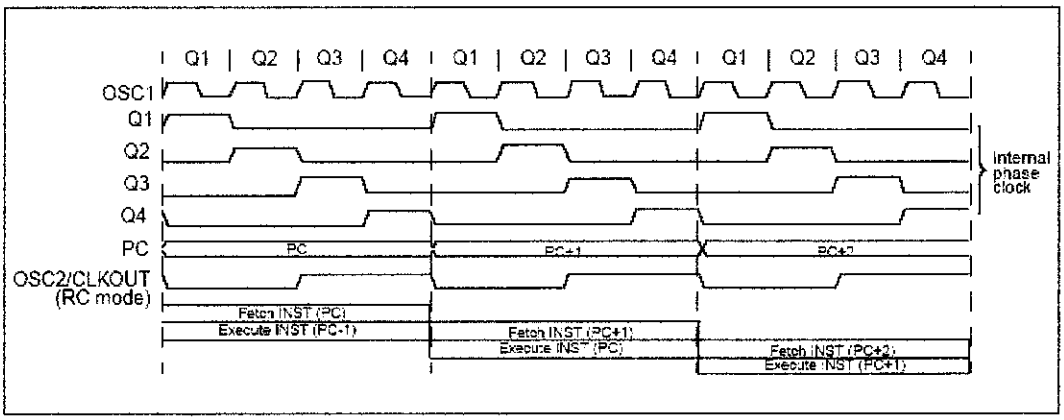


Figure 5 Clock and instruction execution flow [7]

2.1.4 Pin Configurations

PIC16F84A has a total of 18 pins. It is most frequently found in a DIP18 type of case but can also be found in SMD case which is smaller from a DIP. SMD suggests that holes for pins to go through when mounting aren't necessary in soldering this type of a component. **Figure 6** shows the pin configurations for PIC16F84A microcontroller.

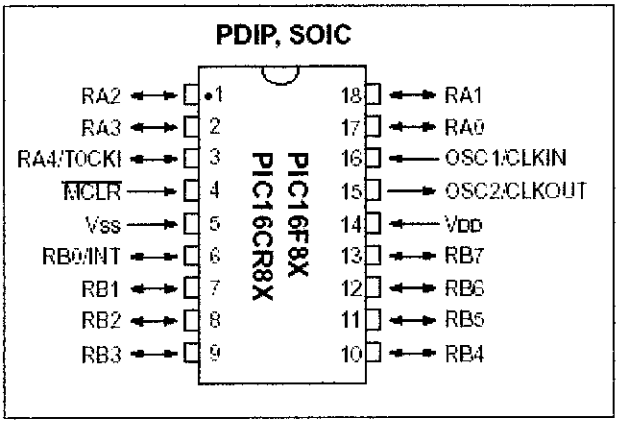


Figure 6 Pin configurations for PIC16F84A [7]

2.2 Serial Communication Interface

2.2.1 RS-232

The RS-232-C was originally set to standardize the interconnections of terminals and host computers through public telephone networks. In the mid- to late 1960's, nearly all serial links for remote access to computers were through a telephone line. Remote access to the large mainframes of the time was accomplished almost exclusively by using the telephone network.

RS-232 data is a bi-polar where +3 to +12 volts indicate an "ON or 0-state (SPACE) condition" while A -3 to -12 volts indicates an "OFF" 1-state (MARK) condition. Modern computer equipment ignores the negative level and accepts a zero voltage level as the "OFF" state. In fact, the "ON" state may be achieved with lesser positive potential. This means circuits powered by 5Vdc are capable of driving RS-232 circuits directly; however, the overall range that the RS-232 signal may be transmitted/ received may be dramatically reduced.

The output signal level usually swings between +12V and -12V. The "dead area" between +3V and -3V is designed to absorb the line noise. This dead area may vary in various RS-232. For instance, the definition for V.10 has a dead area from +0.3V to -0.3V. Many receivers designed for RS-232 to be sensitive to differentials of 1V or less.

The following descriptions outlined the advantages of using serial data transfer rather than parallel are:

- Serial cables can be longer than parallel cables. The serial port transmits a '1' as -3V to -25V and a '0' as +3V to +25V where as a parallel port transmits a '0' as 0V and a '1' as 5V. Therefore the serial port can have a maximum swing of 50V compared to the parallel port which has a maximum swing of 5V and cable loss is not going to be as much of a problem for serial cables as they are for parallel.
- Fewer wires required compared to parallel transmission. If your device needs to be mounted a far distance away from the computer, then 3 core cable (Null Modem Configuration) is going to be a lot cheaper that running 19 or 25 core cable. However the cost of the interfacing at each end must be taken into account.

Figure 7 shows the pin configurations for 9-pin female connector. Serial port found at the PC is of male type connector.

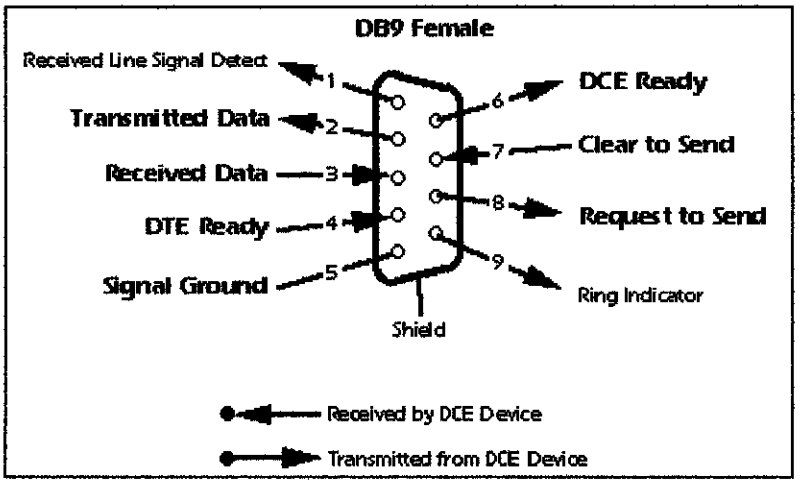


Figure 7 Pin configurations for DB9 female connector

Originally, the primary use of a serial port was to connect a modem to your computer. The pin assignments reflect that. **Table 1** gives brief explanation on the function of each pin at 9-pin connector.

Table 1 Function descriptions of 9-pin connector

Carrier Detect	Determines if the modem is connected to a working phone line
Receive Data	Computer receives information sent from the modem
Transmit Data	Computer sends information to the modem
Data Terminal Ready	Computer tells the modem that it is ready to talk
Signal Ground	Pin is grounded
Data Set Ready	Modem tells the computer that it is ready to talk
Request To Send	Computer asks the modem if it can send information
Clear To Send	Modem tells the computer that it can send information
Ring Indicator	Once a call has been placed, computer acknowledges signal (sent from modem) that a ring is detected

2.2.2 MAX232

MAX232 is a dual driver/receiver that has two internal charge-pumps that convert +5V to ±10V (unloaded) for RS-232 drivers. Each receiver converts EIA-232 inputs to 5-V TTL/CMOS levels. These receivers have a typical threshold of 1.3V and a typical hysteresis of 0.5V, and can accept ±30V inputs. Each driver converts TTL/CMOS input levels into EIA-232 levels. The first converter uses capacitor C1 to double the +5V input to 10V on C3 at the V+ output. The second converter uses capacitor C2 to invert +10V to -10V on C4 at the V- output. **Figure 8** shows the pin configurations of MAX232 and its internal circuitry.

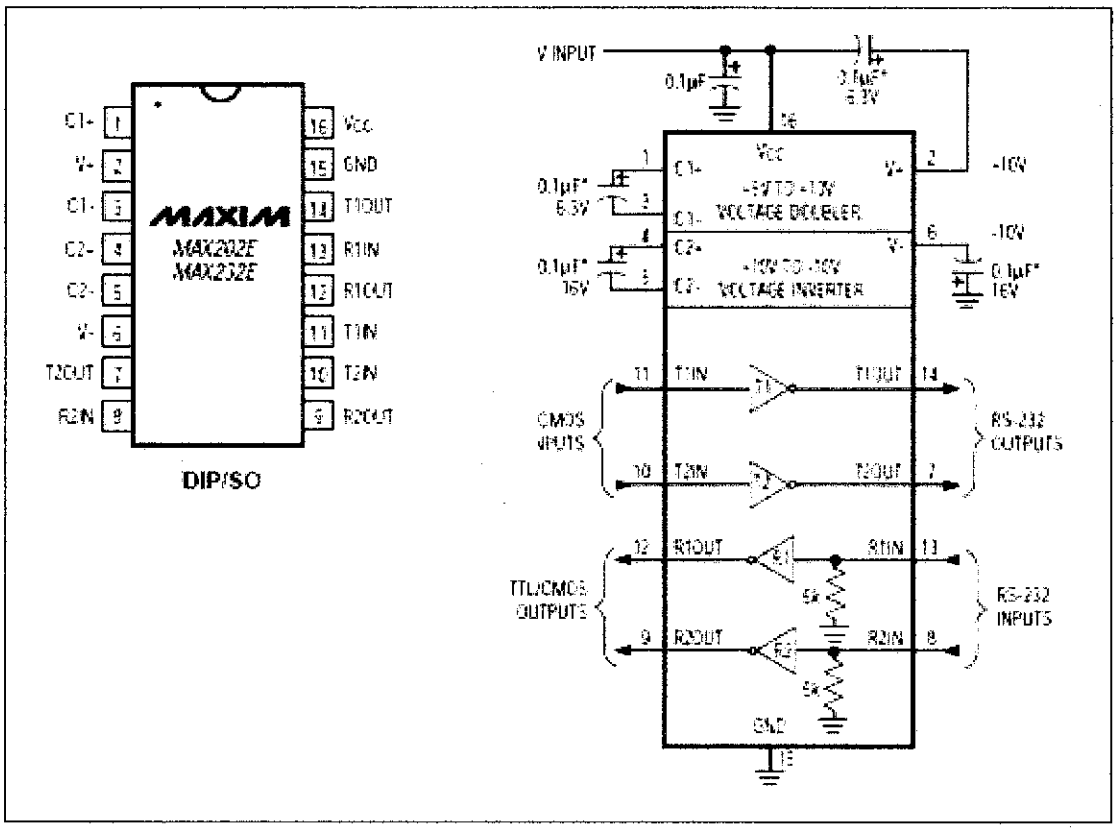


Figure 8 Pin configurations and internal circuitry of MAX232 [8]

2.3 Infrared Characteristics

Commonly, a wireless system consists of transmitter and receiver. The encoder generates serial data that contains address bits and data bits. Decoder compares the data received address bit with own bit settings before process the data bits. Address bits is used to provide identity of transmitter and receiver. In this process only identical address settings can process particular data bits.

Each transmitter and receiver has own exclusive address settings. If transmission is initiated by one of transmitter, both receivers received the IR signal. However, transmitter address bits only match to a decoder and allow the particular decoder to process the data bits and another decoder will reject the signal due to invalid address.

2.3.1 Infrared Remote [9]

All IR remote controls use some kind of IR signal. The remotes transmit pulses of IR light to send the signal to the receiver. These IR LEDs transmit light in the frequency range of 30 to 40 kHz. These high frequencies were chosen so that other light sources would not interfere with the receivers' ability to correctly receive the transmitted signals.

When a button on a remote is pushed it sends a string of signals. The first piece of information in the string is called the Header. The Header usually contains a burst of highs that alerts all the IR receivers in the area to the string of data being sent. Following the burst of highs is the address to the specific machine to receive the next piece of data, the command. As long as the button is held down (depressed), the command will continue to repeat. When the button is released, a string of code called the stop is transmitted. The stop tells the machine to stop its executing of the command.

2.3.2 Binary Coded Signals

The signals are transmitted by the IR LED in some type of binary code. This coding holds information such as the address to the machine that is using the remote and the command that the machine must follow. The address is important because without it the signal would be processed by another IR receiver in the area. It turns out that for most consumer electronics this coding is the same.

The binary signal varies in length for both time and bit length. There are really only three different ways that manufacturers choose to code these signals. This coding is usually based on varying the length of pulses, varying the length of spaces between pulses, or altering the order between spaces or pulses.

2.3.2.1 Pulse-Width Coded Signals

Pulse-width-coded signals (**Figure 9**) vary the length of pulses to code the information. In this case, if the pulse width is short (approximately 550 μs) it corresponds to a logical zero or low. If the pulse width is long (approximately 2200 μs), it corresponds to a logical one or high.

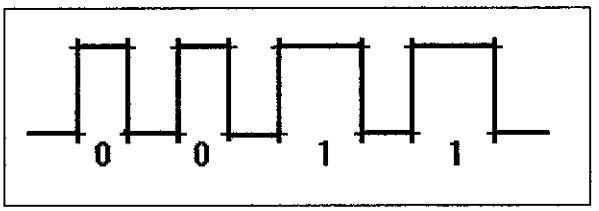


Figure 9 Pulse-coded signal [9]

2.3.2.2 Space-Coded Signals

Space-coded signals (**Figure 10**) vary the length of the spaces between pulses to code the information. In this case if the space width is short (approximately 550 μs) it corresponds to a logical zero or low. If the space width is long (approximately 1650 μs), it corresponds to a logical one or high.

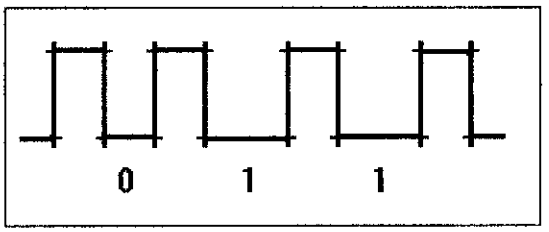


Figure 10 Space-coded signal [9]

2.3.2.3 Shift-Coded Signals

Shift-coded signals (**Figure 11**) vary the order of pulse space to code the information. In this case, if the space width is short (approximately 550 μs) and the pulse width is long (approximately 1100 μs), the signal corresponds to a logical one or high. If the space is long and the pulse is short, the signal corresponds to a logical zero or low.

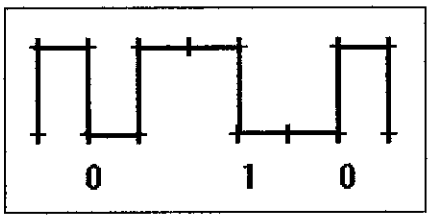


Figure 11 Shift-coded signal [9]

2.4 Microsoft Visual Basic Programming Tool

The Microsoft Visual Basic is the most popular choice to create the Window GUI (Graphical User Interface). In Visual Basic, new windows created are called form. Elements as text boxes and button that placed in the form are called control. The visual basic allows event-driven programming where the user actions cause events, and such event in turn triggers a procedure that is associated with it.

2.4.1 Visual Basic Editions

Visual Basic is available in three versions, each geared to meet a specific set of development requirements. The Visual Basic Learning edition allows programmers to easily create powerful applications for Microsoft Windows and Windows NT. It includes all intrinsic controls, plus grid, tab, and data-bound controls. Documentation provided with this edition includes the Learn VB Now CD plus the Microsoft Developer Network (MSDN) Library CDs containing full online documentation.

The Professional edition provides computer professionals with a full-featured set of tools for developing solutions for others. It includes all the features of the Learning edition, plus additional ActiveX controls, the Internet Information Server Application Designer, integrated Visual Database Tools and Data Environment, Active Data Objects, and the Dynamic HTML Page Designer. Documentation provided with the Professional edition includes the Visual Studio Professional Features book plus Microsoft Developer Network CDs containing full online documentation.

The Enterprise edition allows professionals to create robust distributed applications in a team setting. It includes all the features of the Professional edition, plus Back Office tools such as SQL Server, Microsoft Transaction Server, Internet Information Server, Visual SourceSafe, SNA Server, and more. Printed documentation provided with the Enterprise edition includes the Visual Studio Enterprise Features book plus Microsoft Developer Network CDs containing full online documentation.

2.4.2 The Structure of Microsoft Visual Basic

An application is really nothing more than a set of instructions directing the computer to perform a task or tasks. The structure of an application is the way in which the instructions are organized; that is, where the instructions are stored and the order in which instructions are executed. As applications become more complex, the need for organization or structure becomes obvious. In addition to controlling the execution of an application, the structure is important to the programmer.

As Visual Basic application is based on objects, the structure of its code closely models its physical representation on screen. By definition, objects contain data and code. The form that you see on screen is a representation of the properties that define its appearance and intrinsic behavior. For each form in an application, there is a related *form module* (with file name extension .frm) that contains its code.

Each form module contains *event procedures* — sections of code where you place the instructions that will execute in response to specific events. Forms can contain controls. For each control on a form, there is a corresponding set of event procedures in the form module. In addition to event procedures, form modules can contain general procedures that are executed in response to a call from any event procedure.

Code that is not related to a specific form or control can be placed in a different type of module, a *standard module* (.BAS). A procedure that might be used in response to events in several different objects should be placed in a standard module, rather than duplicating the code in the event procedures for each object. A *class module* (.CLS) is used to create objects that can be called from procedures within your application. Whereas a standard module contains only code, a class module contains both code and data.

The following describe the different types of files and objects that user can include in a project.

- **Form Modules**

  Form modules (.frm file name extension) can contain textual descriptions of the form and its controls, including their property settings. They can also contain form-level declarations of constants, variables, and external procedures; event procedures; and general procedures.

- **Class Modules**

  Class modules (.cls file name extension) are similar to form modules, except that they have no visible user interface. You can use class modules to create your own objects, including code for methods and properties.

- **Standard Modules**

  Standard modules (.bas file name extension) can contain public or module-level declarations of types, constants, variables, external procedures, and public procedures.

- **Resource Files**

  Resource files (.res file name extension) contain bitmaps, text strings, and other data that user can change without having to re-edit your code. For example, if user plans to localize your application in a foreign language, he can keep all of the user-interface text strings and bitmaps in a resource file, which he can then localize instead of the entire application. A project can contain no more than one resource file.

- **ActiveX Documents**

  ActiveX documents (.dob) are similar to forms, but are displayable in an Internet browser such as Internet Explorer. The Professional and Enterprise editions of Visual Basic are capable of creating ActiveX documents.

- **User Control and Property Page Modules**

  User Control (.ctl) and Property Page (.pag) modules are also similar to forms, but are used to create ActiveX controls and their associated property pages for displaying design-time properties. The Professional and Enterprise editions of Visual Basic are capable of creating ActiveX controls.

In addition to files and modules, several other types of components can be added to the project.

- **ActiveX Controls**

  ActiveX controls (.ocx file name extension) are optional controls which can be added to the toolbox and used on forms.

- **Insertable Objects**

  Insertable objects, such as a Microsoft Excel Worksheet object, are components you can use as building blocks to build integrated solutions. An *integrated*

solution can contain data in different formats, such as spreadsheets, bitmaps, and text, which were all created by different applications.

- **References**

  User can also add references to external ActiveX components that may be used by application. User assigns references by using the References dialog, accessed from the References menu item on the Project menu.

- **ActiveX Designers**

  ActiveX designers are tools for designing classes from which objects can be created. The design interface for forms is the default designer. Additional designers may be available from other sources.

- **Standard Controls**

  Standard controls are supplied by Visual Basic. Standard controls, such as the command button or frame control, are always included in the toolbox, unlike ActiveX controls and insertable objects, which can be removed from or added to the toolbox.

Table 2 Visual Basic naming conversion

[illegible]	[illegible]
Form	frm
Command Button	cmd
Text Box	txt
Label	lbl
Option Button	opt
CheckBox	chk
Frame	fra
Horizontal Scrollbar	hsb
Vertical Scrollbar	vsb
Image	img
Picture Box	pic
Combo Box	cbo
List Box	lst
Shape	shp

CHAPTER 3
METHODOLOGY/ PROJECT WORK

3.1 Procedure Identification

In ensuring the smoothness of project execution, all required procedures are identified. **Figure 12** of the following page outlines the main approaches in accomplishing the project. All procedures are carried out subsequently at all times. If necessary, the previous procedure is revised as to do any modification on the project.

3.1.1 Research and Analysis

Research is the prerequisite in the early stage of the project as to pursue on the next five procedures of the project. Through the Internet and several references, the preliminary researches are performed on the following matters; remote control applications, Infrared transmission and reception system through C programming and software design using Microsoft Visual Basic.

3.1.1.1 Remote Control Applications

The basics of Infrared receivers are investigated. In choosing the most suitable means to control the remote control system functionalities, the existing PIC microcontrollers are critically and objectively analyzed. In addition to the research on the Infrared receiver and microcontroller, the examples of PC remotes control projects available on the Internet are gathered [10,11,12]. The projects are analytically reviewed as to get apparent ideas on the system to be designed, specifically the ones integrated with microcontroller. The sample projects gathered are not limited to the PC remote control, but also include the projects those applies wireless transmission, such as remote control system for Sony Walkman [13] and remote control for camera [14].

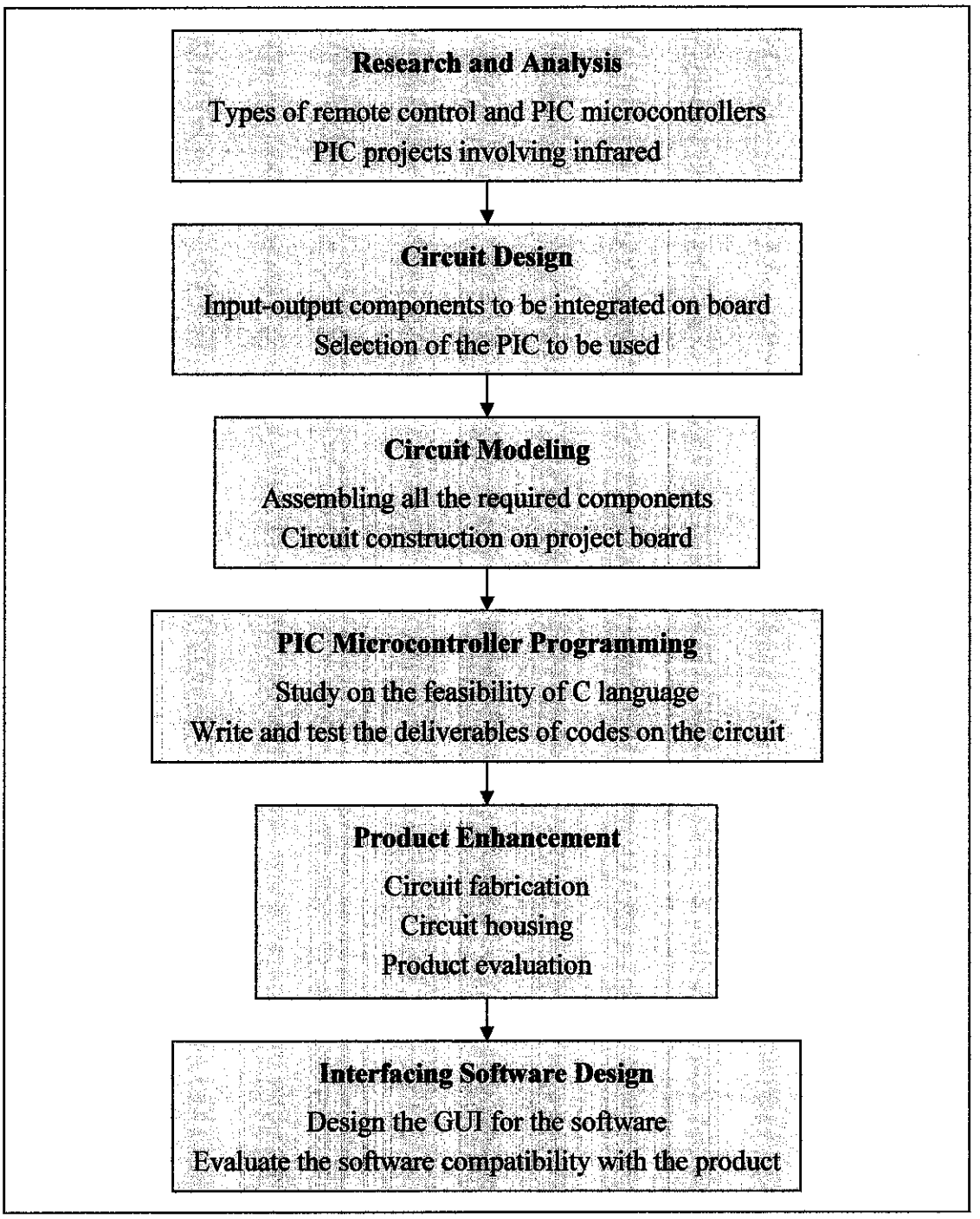


Figure 12 Stages of procedure in delivering the project

3.1.1.2 Infrared Control through C Programming

The behavior in which the receiver works to retrieve the data sent through Infrared signals depends on the program burned into the microcontroller. Thus, the samples of Infrared programs, either using assembly language or C language are analyzed as to ensure the correctness of the C program written for the receiver circuit. The errors made in the programming may affect the whole system functionality and the worst situation is that the receiver fails to receive the Infrared signals sent.

3.1.1.3 Software Design

The study on the software design is also carried out by collecting the tutorials of Microsoft Visual Basic. The software examples with coding disclosed are collected for reference in the later stage of interfacing software design.

3.1.2 Circuit Design

During this stage, the input and output components to be integrated on the remote control system circuit are determined. Since only the Infrared receiver is going to be designed for the project, the only important input component to be integrated is the infrared receiver. A single LED is used as well to confirm that the receiver circuit had been powered-up correctly.

In designing a circuit, the current flows from one component to another should be taken care as excessive current may cause damage to the component and insufficient current may cause the component to function incorrectly. Thus, the resistors are usually placed in between the input or output pin of the microcontroller to the respective input or output components.

Since the system requires communication between the circuit and the PC through the serial port, the MAX232 IC is used. MAX232 provides interface translation between the receiver and PC, by which the received infrared signal is sent to the PC and translated into the ASCII characters.

The examples of project gathered from the Internet are giving enormous guidance in the circuit design. The microcontroller datasheet [7] is reviewed continuously in ensuring that the input and output components are connected to the correct I/O pins of the microcontroller.

3.1.3 Circuit Modeling

The circuit designed in previous procedure is first assembled on the project board for evaluation purpose. Any alterations on the circuit design can be done easily while all the components are being assembled on the board. If in case the interfacing between the circuit and PC fails, the connections can always be modified until the expected results are obtained.

In order to perform the PIC programming, there should be an assembled circuit to evaluate the feasibility of the program written. Else, the burned program is not giving any use since there is no other viable means to observe the input-output poses of the circuit based on the program written.

3.1.4 PIC Microcontroller Programming

There are three most common ways can be used to program the PIC microcontroller, either using the assembly language, PIC Basic or C language. The assembly language and PIC Basic are considered to be quite messy and complicated to be used in programming. Furthermore, it may take more time to understand the deliverables of the built-in functions of assembly language compared to C.

C language is more straightforward and takes shorter time to be understood. C compiler is made by the third parties to provide viability to the programmer in coding the PICs. The basic knowledge on C syntax, its built-in functions and pre-processor are essential in pursuing the project to program the microcontroller.

3.1.5 Product Enhancement

Once all the procedures outlined previously are accomplished, the circuit is going to be fabricated on a Vero board. The product is evaluated after all the components are soldered on the board. This is to ensure that no fault had occurred during soldering process. If there is a fault, the troubleshooting will take place.

The final touch on the hardware design would be putting the Vero board into a presentable casing. The casing should be small and compact. The wiring from the receiver circuit to the serial port for interfacing and to the USB port for power supply must be adequate so that it can be moved easily around the PC.

3.1.6 Interfacing Software Design

The Graphical User Interface (GUI) of the interfacing software is designed by determining the ASCII characters received at the Hyper Terminal of the PC. The applications which are required to be activated through the remote control system are registered based on particular ASCII characters received.

Once the software is finalized, the compatibility between the software and the remote control system is evaluated. This covers the capability of the receiver to catch the signals sent from a remote control, transmit the signal to the PC through serial communication and at the end the interfacing software should be able to start an application if the received ASCII characters match any of the defined characters.

3.2 Tools Required

Several engineering tools have been utilized in working out with the project. The tools used include EAGLE Layout Editor 4.13, PIC C Compiler and WARP 13. During the interfacing verification phase, the Hyper Terminal and PC Remote Control have been utilized. Microsoft Visual Basic is employed during the interfacing software design stage.

3.2.1 EAGLE Layout Editor 4.13

EAGLE Layout Editor can be considered as a complete tool for circuit design where the schematic can be drawn and the PCB can be automatically routed from the designed schematic. All the components used in the circuit are available in its library including PIC16F84A. The only limitation with the tool is that the maximum size of PCB board allowed to be laid out is only 4" X 3.6". As more than one unit of the receiver circuit needs to be produced, the circuit had been transformed into PCB layout. The layout is further edited to ensure that the PCB produced is of the smallest size possible. **Figure 13** shows the program window for EAGLE.

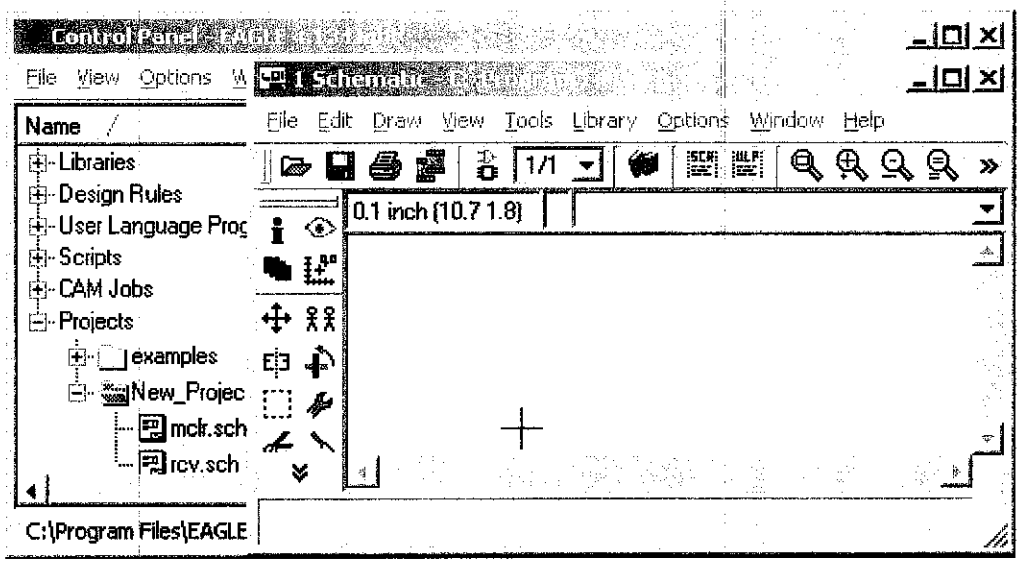


Figure 13 EAGLE Layout Editor program window

3.2.2 PIC C Compiler

PIC C Compiler (**Figure 14**) is the most crucial tools used in writing the PIC program using C language. In general, the program written should include the header file of the PIC used, its configuration bits, clock speed which is the value of oscillator used, defining the inputs and outputs used and the main() function. Once the program is completely written, it is compiled to generate the HEX file. If there is an error in the program, the C Compiler stops compiling and at the bottom side of the window, the error identified is highlighted in red.

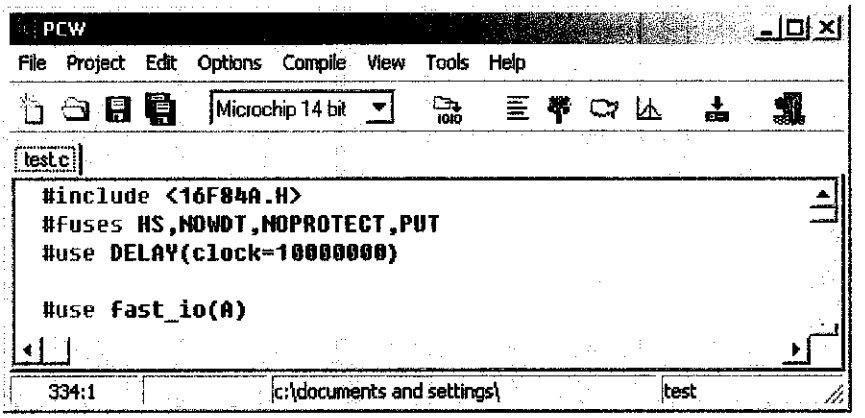


Figure 14 PIC C Compiler program window

3.2.3 WARP 13

In succession to the HEX file generation, the HEX code can be burned into the PIC using a chip burner using WARP 13 program. This is the vital part of the microcontroller programming as to ensure its capability to control the circuit system as desired. Before burning any particular program into the PIC, its EEPROM should be erased or blanked. Once the program button is pressed, the program will be written into the PIC memory. **Figure 15** shows the chip burner used for program burning.

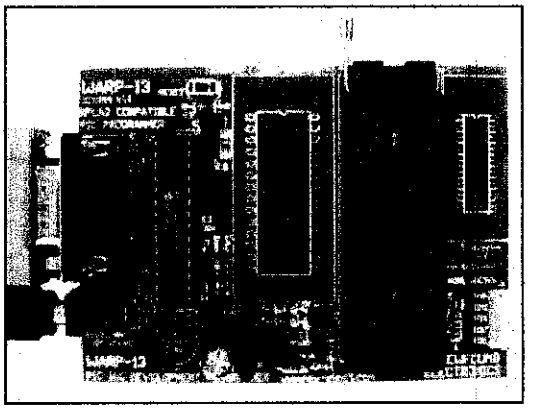


Figure 15 WARP13 PIC programmer [15]

3.2.4 Hyper Terminal

The Hyper Terminal is used to check the signal received through serial communication port. If there is connectivity between receiver circuit and PC, the signal received at the infrared receiver can be read in ASCII code at the Hyper Terminal window. The Hyper Terminal application shown in **Figure 16** can be accessed under Accessories/ Communications.

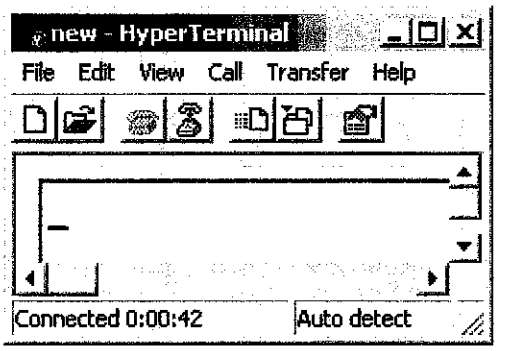


Figure 16 Hyper terminal window

3.2.5 PC Remote Control

The software is used to check the ASCII character received through the serial port using its Test Origin capability. The obtained ASCII characters obtained for the respective number pressed on the remote control are stored. These data are important in defining the conditions to execute particular application using Microsoft Visual Basic. The test origin window of PC Remote Control is shown in **Figure 17**.

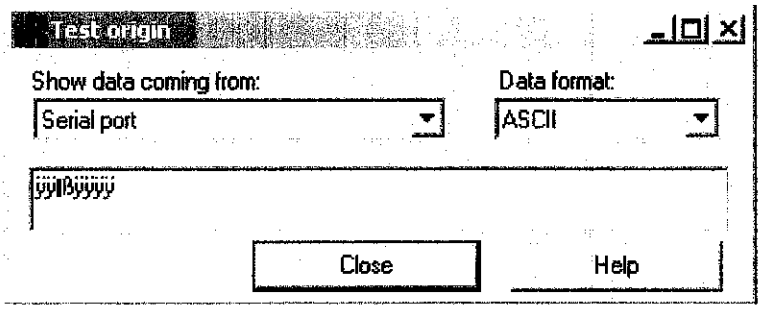


Figure 17 Test Origin window for PC Remote Control

PC Remote Control can also be used to register several tasks of an application. As an example, for Windows Media Player, any button on the remote control can be set for the Play button by learning the signal origin. **Figure 18** shows the window to learn the signal from any remote control. By pressing the same button repeatedly, the software will learn the sent signal.

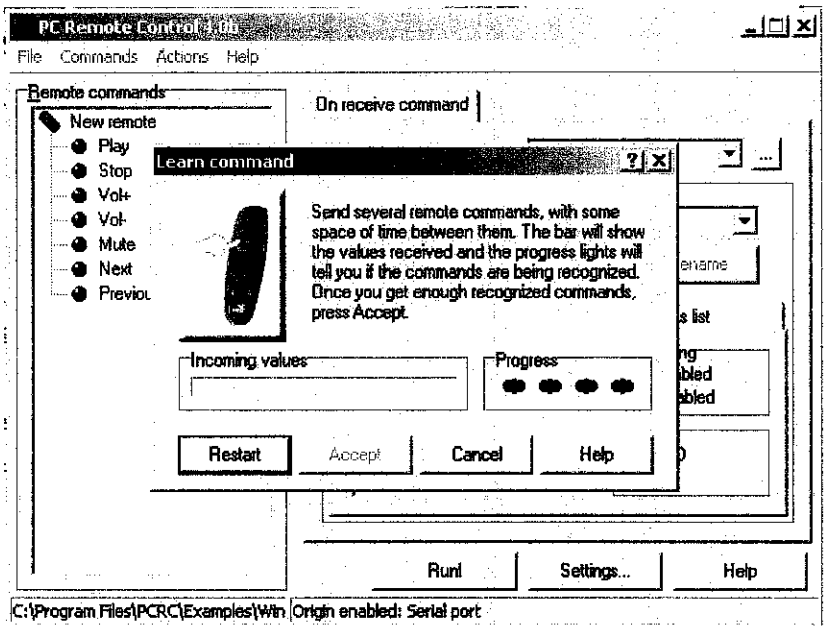


Figure 18 Learn command window for PC Remote Control

3.2.6 Microsoft Visual Basic 6.0

Microsoft Visual Basic is windows GUI programming software. The tool is utilized in designing the GUI for the system. Therefore, the software is suitable to be used in extracting the input from the serial port and representing it via graphical interface. The main application of interfacing software will work in the similar manner to the PC Remote Control software. **Figure 19** exhibits design form of Microsoft Visual Basic.

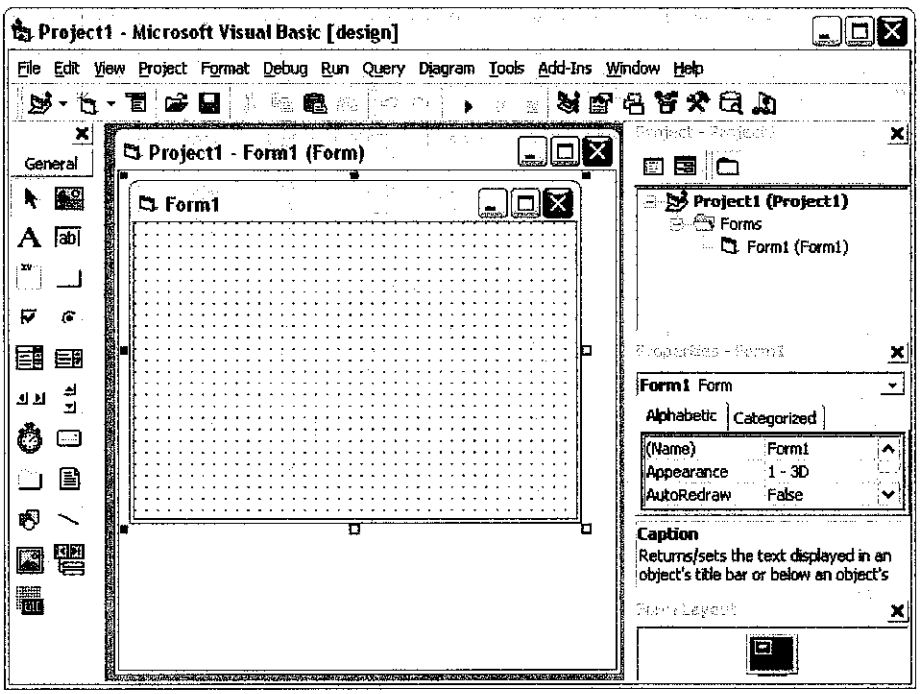


Figure 19 Design form of Microsoft Visual Basic [16]

3.3 Hardware Required

There are four classifications of the hardware integrated into the infrared receiver circuit. There is a microcontroller that controls the viability of the receiver system, an RS232 driver, the input component for receiving the infrared signal and the output component to indicate the circuit had been powered up.

3.3.1 PIC16F84A Microcontroller

The PIC16F84A is chosen as the microcontroller to control the infrared receiver operations due to the number of I/O pins available for the entire microcontrollers analyzed. In the project, two I/O pins are sufficient to carry the receiver viability for infrared receiver module and connection indicator. **Figure 20** shows the PIC16F84A utilized for the project

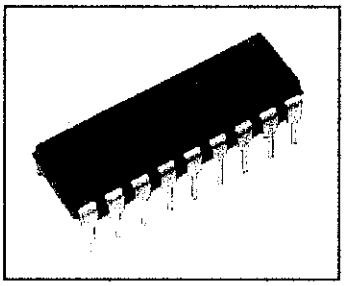


Figure 20 PIC16F84A microcontroller [17]

3.3.2 MAX232 Level Converter IC

The MAXIM MAX233 chip converts between "TTL levels" (0 to +5 volts) of the PIC and "RS232 levels" (+9 to -9 volts) of the serial line. The magic of the MAXIM chip is that it does this conversion with a single +5 V supply; oscillators and charge pumps internal to the chip generate the required + and - 9 volts. The MAX233 consumes about 5mA whether or not serial data is being sent or received. The IC of MAX232 is shown in **Figure 21**.

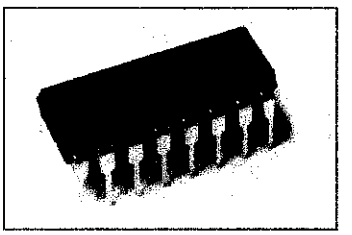


Figure 21 MAX232 IC [18]

3.3.3 Input and Output Components

The infrared receiver module is the only input component integrated on the circuit. The Sharp IS1U20 is a compliant Infrared receiver. When it detects Infrared, it outputs a low signal. The LED is used as an output component, flash when the circuit is first powered up. **Figure 22** shows the Infrared receiver module and LED integrated in the receiver circuit.

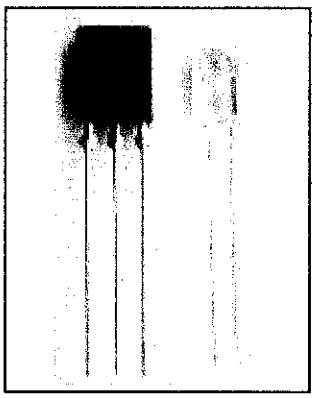


Figure 22 Infrared receiver module (left) and LED (right)

3.3.4 Additional Components

There are some additional components required for interfacing the receiver circuit with the PC including 9-pin female connector and serial cable, as shown in **Figure 23**. The serial cable is required when the serial wiring from the circuit is not long enough to be connected to the serial port of the PC.

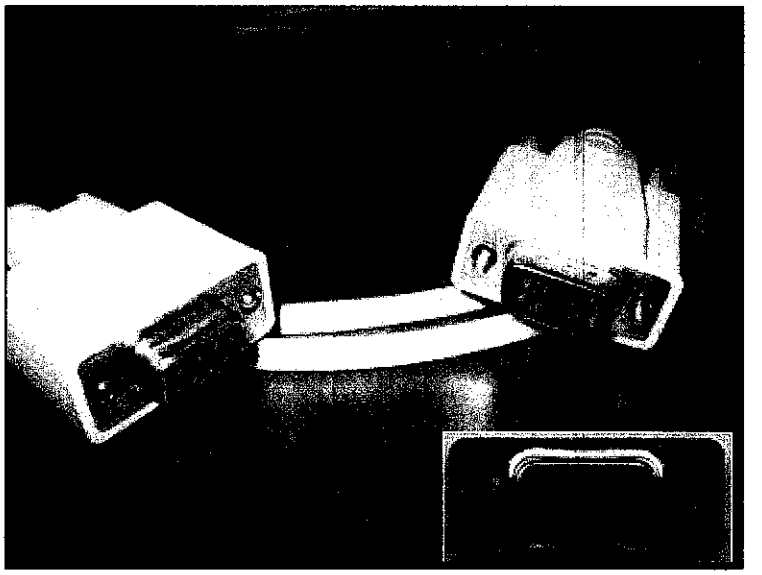


Figure 23 Serial cable and 9-pin female connector (small) [19]

CHAPTER 4
RESULTS AND DISCUSSION

4.1 Findings

In general, the Infrared receiver for the remote control system had been successfully produced. Several tests have been performed on the device, by which have given the expected results. The Gantt chart enclosed in **Appendix A** shows the project planning throughout the first and second semesters.

4.1.1 Circuit Design

The schematic shown in **Figure 25** of the following page represents the circuit design of the Infrared receiver. It can be seen that the PIC16F84 is used instead of PIC16F84A. Actually, there is no PIC16F84A symbol available in the design library. The reason is that the pin configurations for both microcontrollers are similar. So, the symbol for PIC16F84 can also be used to represent the PIC16F84A.

4.1.1.1 PIC Oscillation Circuit

In establishing high speed oscillations (HS), an oscillator of 10MHz, with 22pF capacitors are connected to the OSC1 and OSC2 pins of the PIC. The connections shown in **Figure 24** are the heartbeat of the microcontroller.

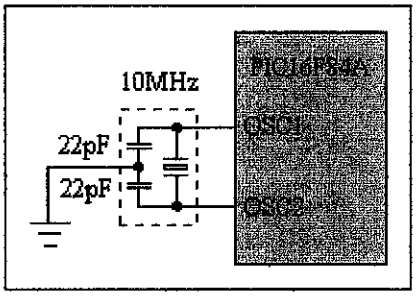


Figure 24 PIC oscillation circuit

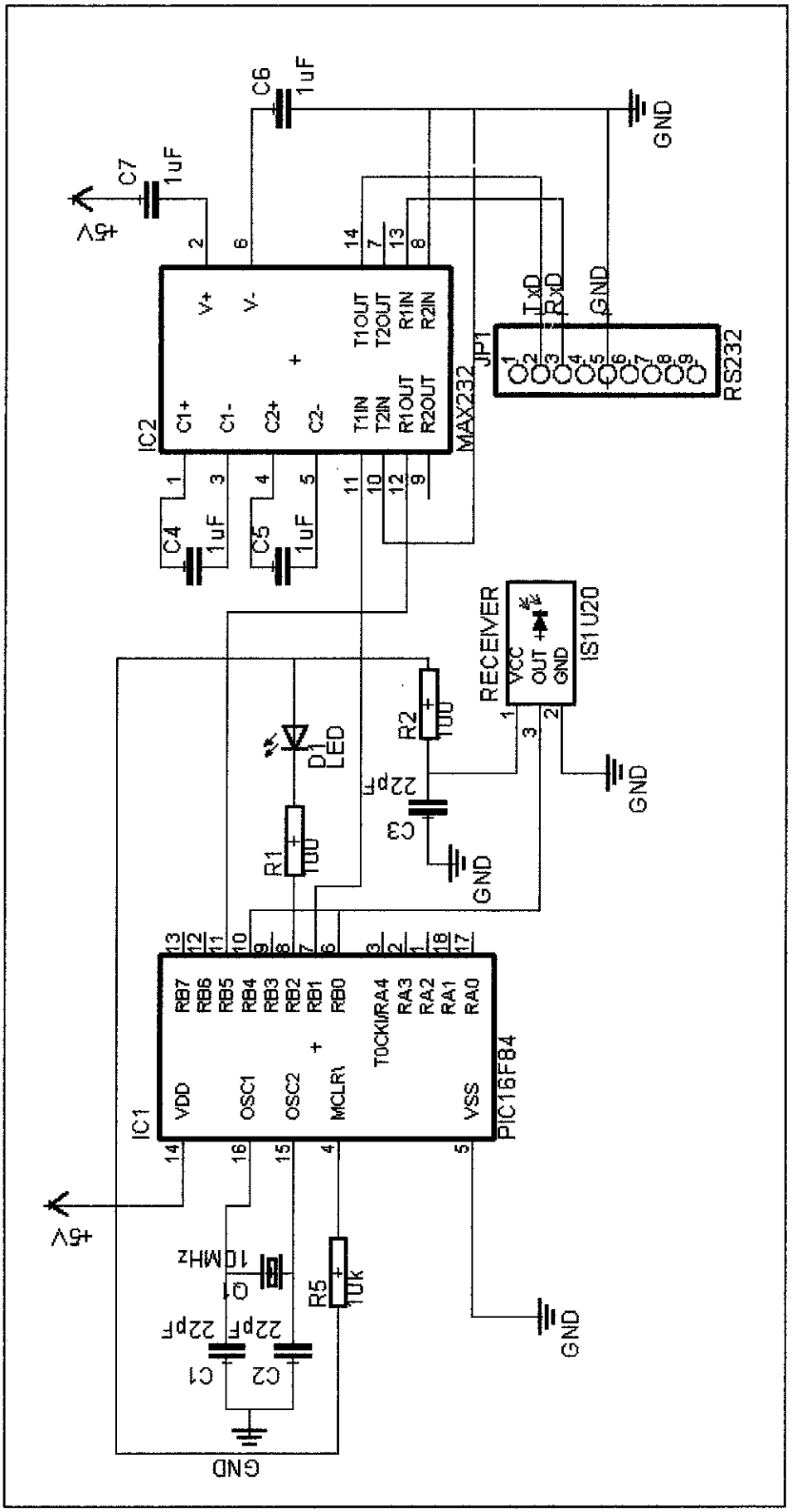


Figure 25 Schematic of the circuit diagram for Infrared receiver

4.1.1.2 Infrared Receiver Circuit

The infrared receiver module is connected to pin RB0 of the PIC as an interrupt-on-change input. The Infrared receiver can generate false IR strings when there is high frequency distortion on the +5V supply. In eliminating or reducing the occurrences of this problem, the connections of **Figure 26** have been made as the circuitry part of the receiver. The 22pF capacitor is connected in between the +5V and ground to filter the noise received.

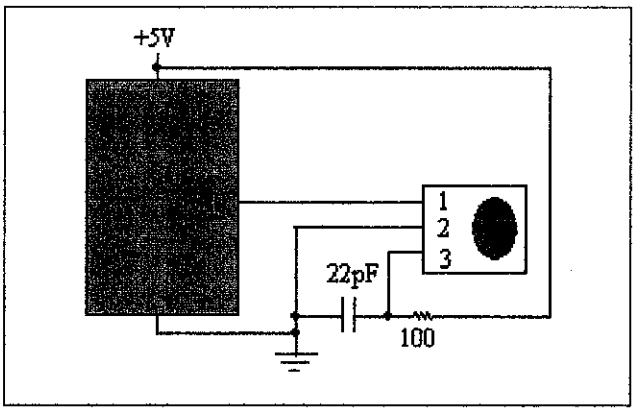


Figure 26 Infrared receiver circuit

4.1.1.3 Power-up Indicator Circuit

The resistor is connected to the power up indicator as to avoid the LED from being burned due to the excessive current sourced to it. The cathode of the LED is connected to pin RB2, and its anode is connected to +5V supply. Thus, the LED is made low for it to turn on. The circuitry power up indicator is shown in **Figure 27**.

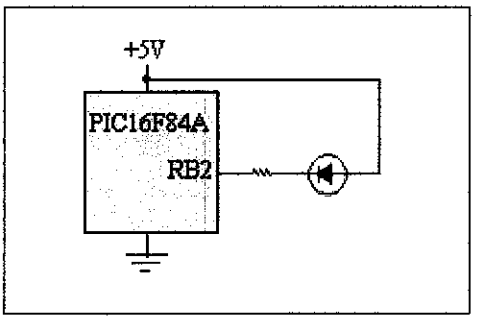


Figure 27 Power-up indicator circuit

4.1.1.4 Serial Communication Circuit

MAX232 is used as the translation IC for serial I/O communication between the circuit and the PC. Pin RB1 and RB5 of the PIC are connected to the CMOS input and CMOS output of MAX232. From MAX232, the RS232 output and RS232 input pins are connected to the receive-in and transmit-out pins of 9-pin female connector, respectively. **Figure 28** exhibits the connections for serial communication circuit.

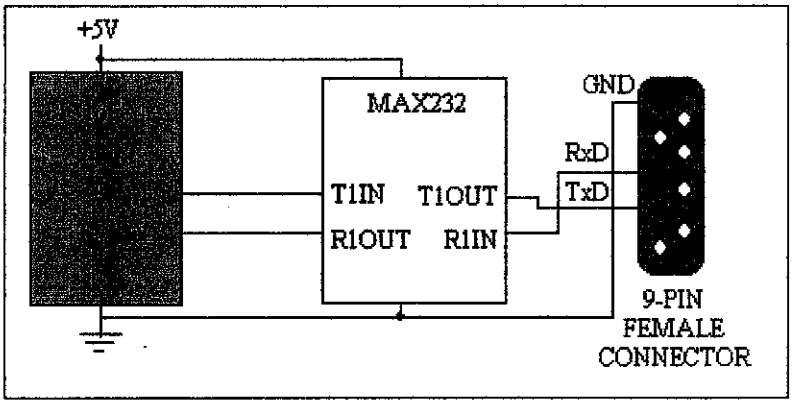


Figure 28 Serial communication circuit

4.1.1.5 Power Supply Circuit

The receiver circuit gets +5V power supply from the computer USB port. The circuit shown in **Figure 29** shows both PIC and MAX232 power pins connected to the supply from USB port.

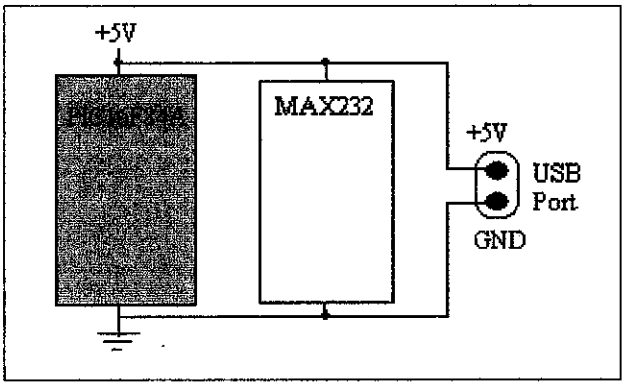


Figure 29 +5V supply circuit from USB port

4.1.2 Circuit Modeling

During circuit modeling, the voltage is obtained from the +5V supplied by the USB port of the computer. The serial connection is connected to the serial cable as the wiring from breadboard to the female connector is short.

Figure 30 exhibits the assembled components of the receiver circuit on a read board. Only one project board is used to assemble all required components for the Infrared receiver circuit. The circuit modeling is quite messy due to the usages of jumpers. The jumpers are used as there is limited space to allow all resistors to be connected in series with the respective microcontroller pins.

The Infrared receiver is located at the right side of the breadboard in the figure below. The usage of the breadboard is optimized with the usage of PIC16F84A and MAX232. Those ICs are located at the right and left side of the board, respectively.

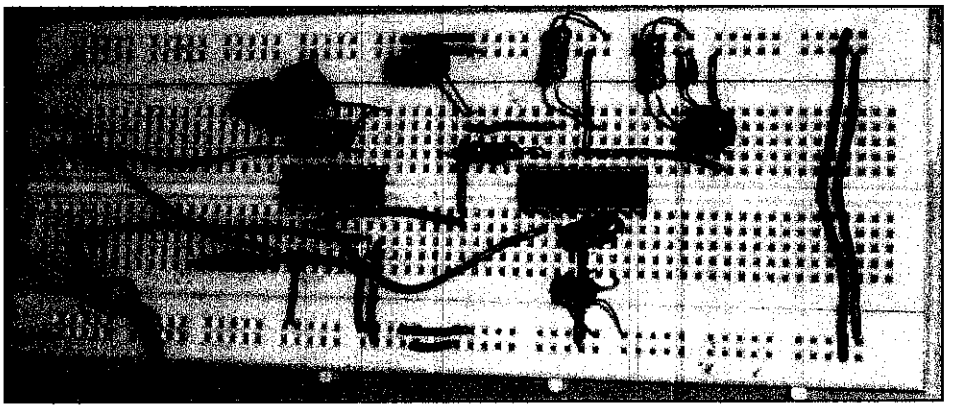


Figure 30 Circuit modeling on a breadboard

4.1.3 Product Enhancement

The final enhancement stage emphasizes on circuit fabrication and product evaluation. Later, a casing is designed to allocate the fabricated circuit. Once accomplished, the final product is evaluated.

4.1.3.1 Circuit Fabrication

While performing the soldering task, the heat sensitive components like MAX232, PIC16F84A and Infrared receiver module should not be overheated. Else, they might get damage. As an early precaution, these components are not going to be soldered directly onto the board. Instead, the IC sockets as shown in **Figure 31** are

going to be used. Later, the IC can be plugged in. This provides an advantage for the PIC since it can be easily removed for further modifications of the PIC program.

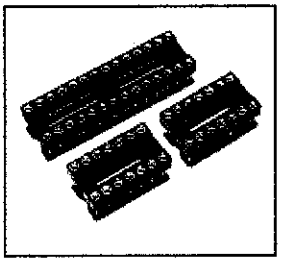


Figure 31 IC sockets

The fabricated Infrared receiver circuit on the Vero board is shown in **Figure 32**. The area used for locating the whole circuitry is approximately 7.5cm x 7.5cm. The reduced number of jumpers is used as the connections are made with line soldering at the bottom side of the Vero board. The green connector connects the circuit to the PC serial port. Whereby, the +5V supply is connected to the circuit through the white connector.

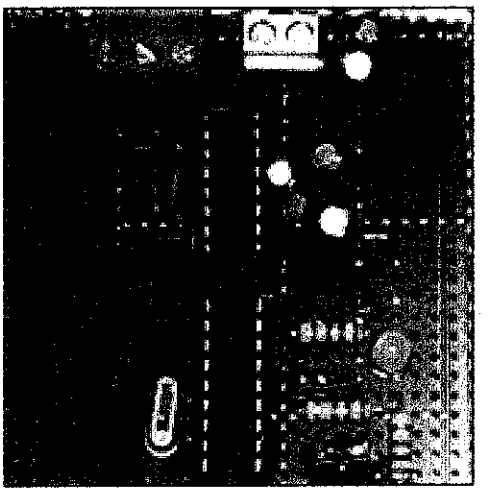


Figure 32 Fabricated Infrared receiver circuit; top side (left), bottom side (right)

4.1.3.2 Circuit Housing

As the circuit is verified to function as expected, the fabricated receiver is then transferred into a designated housing. The material used for housing is a transparent board (**Figure 33**) so that the internal circuitry can be seen clearly. Some more, the power up indicator need to be visible to the user.

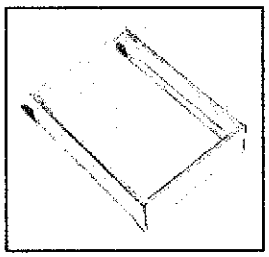


Figure 33 Housing for the circuit

The height of the housing designed is approximately 4cm, shown in **Figure 34**. One side of the housing is drilled with two holes to allow the connection from serial port and USB port to their respective connectors on the board. Sides surrounding the circuit are covered with black film in lowering the amount of undesirable signals detected.

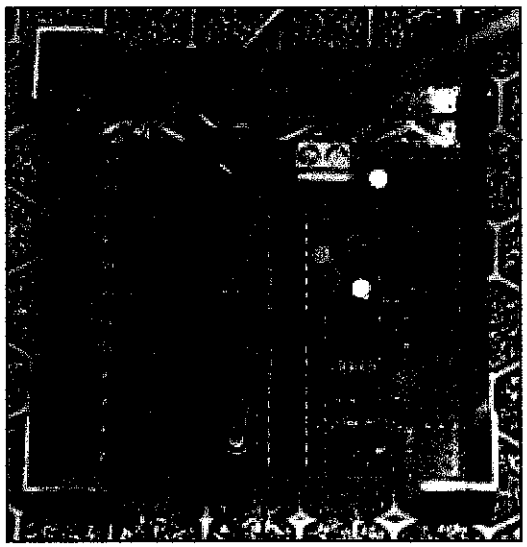


Figure 34 Fabricated circuit with housing

4.1.3.3 Product Evaluation

The product evaluation is performed by integrating the receiver with the PC. A universal remote control is used to send the Infrared signals to be received by the receiver. The testing is then done using the available PC remote control software, where several applications tried to be initiated through the system. The product works as expected and the final stage of designing new interfacing software is carried out. **Figure 35** shows the hardware of the remote control for PC, where the Infrared receiver and universal remote control are included.

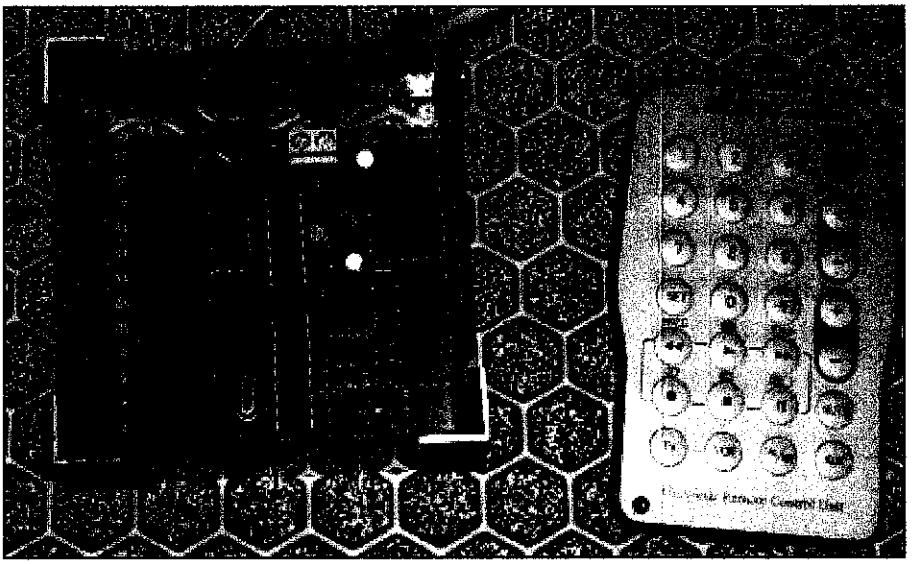


Figure 35 Hardware of remote control for PC

4.1.4 Interfacing Software Design

In designing the interfacing software, the Hyper Terminal and PC Remote Control software have been used. Initially, only Hyper Terminal is used to check the signal received. But, the long strings of characters received made the single signal unrecognizable. The received signals from Sony and Philips remote control through Hyper Terminal are shown in **Figure 36** and **Figure 37**, respectively.

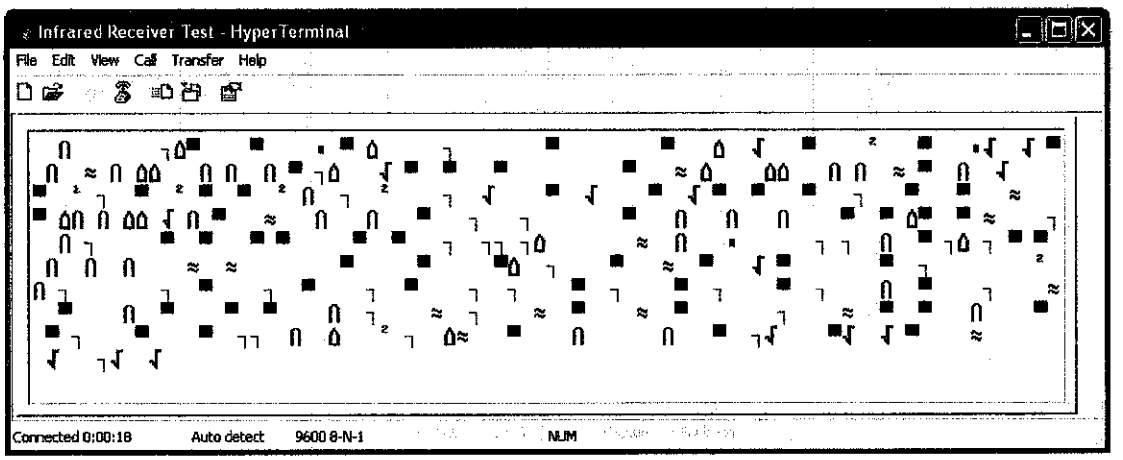


Figure 36 Received signals from Sony remote control

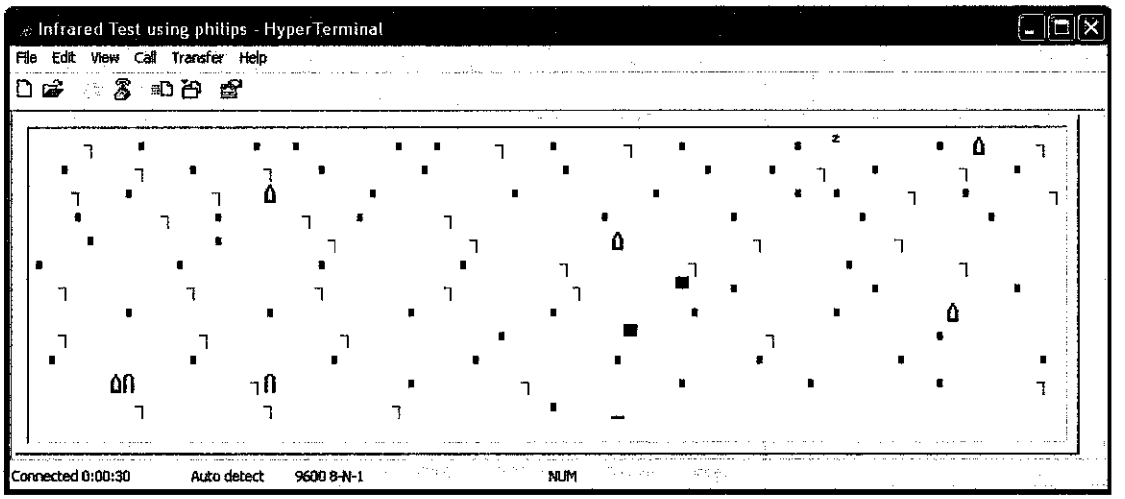


Figure 37 Received signals from Philips remote control

Since the repetition of the signals received cannot be controlled and the real string cannot be detected, other alternative had been tried by using the PC Remote Control software. By learning the origin of the signal, PC Remote Control can display a single string of every signal. The window in learning the origin is shown previously in **Figure17**.

The applications to be initiated wireless are registered using the codes of origin detected through PC Remote Control. There are registered to perform particular tasks according to the application used.

Figure 38 shows the main form of the interfacing software designed using Microsoft Visual Basic. There are several multimedia applications made enable to be accessed wirelessly. The selection of the application depends on the availability of shortcut keys used to control their main task. From the figure shown below, the most relevant applications to be controlled wirelessly include Winamp, Windows Media Player and Microsoft Power Point.

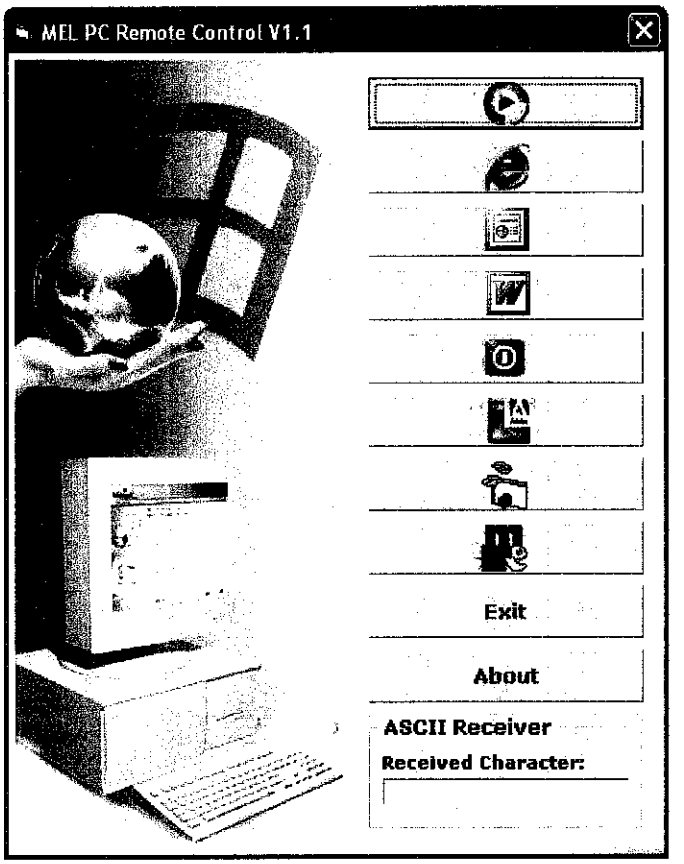


Figure 38 The main form of the interfacing software designed

By pressing any assigned button at the remote control, the applications can be accessed wirelessly. The operation manual of the remote control system is enclosed in **Appendix D**. If the Exit button is pressed, the program is linked to the form shown in **Figure 39**. The confirmation form is designed so that the software would not be unintentionally closed.

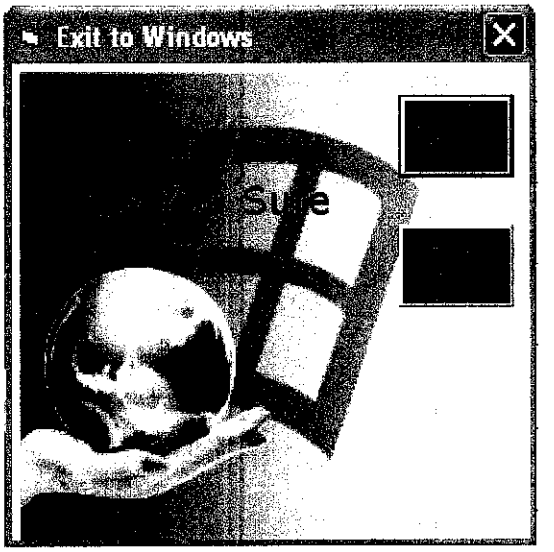


Figure 39 Exit form linked from the main form

4.2 Problems Encountered

Naturally, it is almost inevitable to avoid problems and complications when it comes to developing a system from scratch. It can be said that the time spent on designing is almost equal to the time spent for troubleshooting. There are several occasions where implementation has suffered some problems even though the design theory was successful. The main area affected by this consequence is the integration of all the various components to establish the serial communication.

4.2.1 PIC Programming

Previously, the PIC16F84A cannot be burned using the WARP-13 programmer. These give difficulties as the circuit functionalities are controlled by the program burned inside the microcontroller. But, if the PIC16F877 is used for burning using the same burner and the same software, the burning can be carried out smoothly. The problem encountered is then referred to the manufacturer website, which is Newfound Electronics. The solution to the problem is to get updated WARP-13 software with its firmware to allow for the burning of PIC16F84A to take place.

4.2.2 Circuit Design

The finalized circuit also needs to go through a series of troubleshooting. The main problem is that the infrared signals received are not constant for the same button pressed on the remote control. This gives difficulty in defining the condition to be executed for any button pressed. In addition to that, there are some buttons those output the same characters. The approach to this problem is to check for most of the available brand protocol using the universal remote control. The protocol that gives most stable signals is then chosen to be defined during the initialization of an applications at the software designed. Any of the buttons that give the same character when pressed is not being utilized in the software designed. Only the buttons those able to generate stable and different signals are being employed. For each protocol tested, there are around ten buttons can be utilized.

Actually, the plan is to design both receiver and transmitter for the remote control system. But, as for some merits and demerits considered, it is decided to keep with only designing the receiver. The main obstacle towards designing the transmitter is to transmit signal wirelessly. The specific command to generate this task cannot be found. Several ways have been tested, and those were all failures. Another demerit of designing a transmitter is to increase the overall cost for the system. The use of another piece of PIC, Vero board and switches may sum the cost to more than RM10.

4.2.3 Circuit Fabrication

During circuit fabrication on the Vero board, the problem had been faced while performing the soldering task. The first circuit fabricated on the Vero board cannot function as it had been functioned while being assembled on the breadboard. Unfortunately, before fabricating the circuit on Vero board, all the components assemble on the breadboard had been removed for soldering. Although the design circuit had been followed correctly, the error is still undetectable. Finally, the circuit modeling is performed again. But, this time the modifications are made at the connections to MAX232 where the serial communication 1 is used to be connected to the RS232 and PIC serial in and out. The modeling circuit works as expected after the modifications have been made.

The second fabrication is made by not disassembling the components on the breadboard. Instead, new components are gathered for soldering. In case of any fault, the working model can referred directly.

4.3 Discussion

During early stage of circuit design, several receiver circuits have been assembled for testing purpose. The idea is to get acquainted with the C language used to program the PIC and to test the circuit capability of receiving the Infrared signals. There are some receiver circuits, which do not deploy PIC, but can still receive the signals sent from the Infrared transmitter. Instead, it is using 555 Timer, to check for the received signals. As the deliverables of the project had outlined on PIC facilitation, the finalized circuit designed had used the microcontroller to control the detection of Infrared signals by the receiver.

The PIC with smallest size and number of I/O pin is used for the project. The assignments of I/O pins from PIC16F84A used to the respective components or connections are summarized in **Table 2.**

Table 3 Summary of I/O pins assignments from PIC16F84A

RB0	Infrared receiver input
RB1	Serial out (to CMOS input of MAX232)
RB2	LED out
RB5	Serial in (to CMOS output of MAX232)

For power supply, USB port at the computer is used to give the +5V supply. Previously, some tests have been performed in using the supply from the serial port. But, due to the internal resistance, the voltage supplied through some of PC serial port is not sufficiently 5V. This condition is obviously experienced if the circuit is connected to the serial port of a notebook. As an alternative, the supply from USB port is used for the circuit. There is no extra circuitry required. The +5V and ground points from the circuit can be connected directly to the +5V and ground of the USB port, respectively. The use of USB port for powering up the circuit is essential as the circuit is considered as one of the PC-based device. Furthermore, there is no need to purchase for 9V battery to power up the circuit.

CHAPTER 5
CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusion

The allocated project tasks for the whole two semesters had been performed completely. The outlined study scopes have been covered thoroughly. The provided time frame is proved to be adequate for the project accomplishment.

The produced Infrared receiver that totalized to the complete system of remote control for PC had fulfilled all the required deliverables outlined, PIC integration, serial communication and interfacing software design using Microsoft Visual Basic.

5.2 Relevancy to the Objectives

The knowledge of basic electronics learned during the early years of study is reapplied in designing the circuit. In addition, the new experience had been gained in using microcontroller as the mean of controlling the application designed.

The skills in C programming are augmented through the C language written to program the PIC. The understanding of the C syntax and its built-in functions had been enhanced through the analysis on program examples. In the beginning of the programming stage, the heuristic approach is the applied in getting the desired output behavior from the circuit.

The design of interfacing software using Microsoft Visual Basic had give very new knowledge to the Graphical User Interface design. The familiarization and quick grasping of the software had allowed the ability the design new interfacing software for the remote control system instead of only using the existing software in the market.

5.3 Suggested Future Work for Expansion and Continuation

The receiver circuit design can be added with its own transmitter, means that to design a new transmitter to send the raw signals. The transmitter should work in the same way as the available remote control, which is able to send strings of characters. It gives advantage in term of security although the cost of the whole system may increase. Herewith, there will be no case such as other person is obstructing the system control using the same type of remote control.

Another recommendation made is to modify the receiver circuit in such a way that it can be used as a transceiver. The transceiver functions like both transmitter and receiver. It can receives signal from universal remote control and at other time it can transmit signal to control the applications of television. The necessity of using the transmitter to control the applications of electronic devices is during the misplacement of the remote control. The transceiver is utilizing the half-duplex communication where only one system can send at a time.

As for the interfacing software, it can be designed to be more efficient in handling the signals from remote control. The coverage of utilization should be widened by defining several ASCII character from different protocol to execute particular errand in applications. With this, the control on the software is not limited to a single type of remote control but can be made using varieties of remote control available in the market.

The final recommendation is to fabricate the circuit on a PCB board instead of being soldered on the Vero board. This way can reduce the size of the circuit until the whole circuit with housing is just the size of the palm.

REFERENCES

[1] Remote Control <www.answers.com/remote_control>

[2] Remote Controls <http://www.eriding.net/media/photos/ict/040929_rfoster_mp_ict_remote_controls.jpg>

[3] How Serial Ports Work <http://computer.howstuffworks.com/serial-port2.htm>

[4] Infrared <www.answers.com/infrared-1.htm>

[5] Radio Frequency <www.answers.com/radio_frequency.htm>

[6] Bluetooth <www.answers.com/bluetooth.htm>

[7] Datasheet of PIC16F84A <www.microchip.com>

[8] Datasheet of MAX232 <

[9] The Infrared Remote <http://www.ee.washington.edu/conselec/A95/projects/pierreg/works/works.htm>

[10] iRX 2.2 PIC Development Board <http://web.media.mit.edu/~ayb/irx/>

[11] Infrared Receiver for Serial Port http://www.lirc.org/receivers.html

[12] Universal Infrared Receiver < http://www.geocities.com/SiliconValley/Sector/3863/uir/index.html>

[13] Universal Infrared remote Control System for Sony Walkman <http://minidisconline.cjb.net/>

[14] Nikon IR Remote Shutter Release <http://www.alanmacek.com/nikon/>

[15] WARP-13 Programmer Board <www.junun.org/MarkIII/images/warp-13.jpg>

[16] Visual Basic <http://www.didya.com/products.asp?id=5>

[17] PIC16F84A microcontroller <www.sparkfun.com/shop/index.php?shop=1&cat=80>

[18] MAX232 IC <http://www.radioshack.com/home/index.jsp>

[19] Serial Interface <http://www.campbellsci.com/10873-cable>

[20] IR Remote < http://www.armory.com/~spcecdt/remote/IRremote.html>

[21] Build Your Own Microcontroller Project http://chaokhun.kmitl.ac.th/~kswichit/

[22] Practical PIC Projects <http://www.petesworld.demon.co.uk/homebrew/PIC/picprojects.htm#IR%20Remote%20Control%20Repeater>

[23] Discover Circuits - Computer Interfaces <http://www.discovercircuits.com/C/comp-interf9.htm>

[24] C and IR <http://www.vermontficks.org/pic.htm>

[25] PICmicro C Compiler <http://www.ccsinfo.com/picc.shtml>

[26] PIC C Routines, Peter H. Anderson.
<http://www.phanderson.com/PIC/PICC/pic_c_routines.html>

[27] PIC tutorial <http://www.mstracey.btinternet.co.uk/pictutorial/picmain.htm>

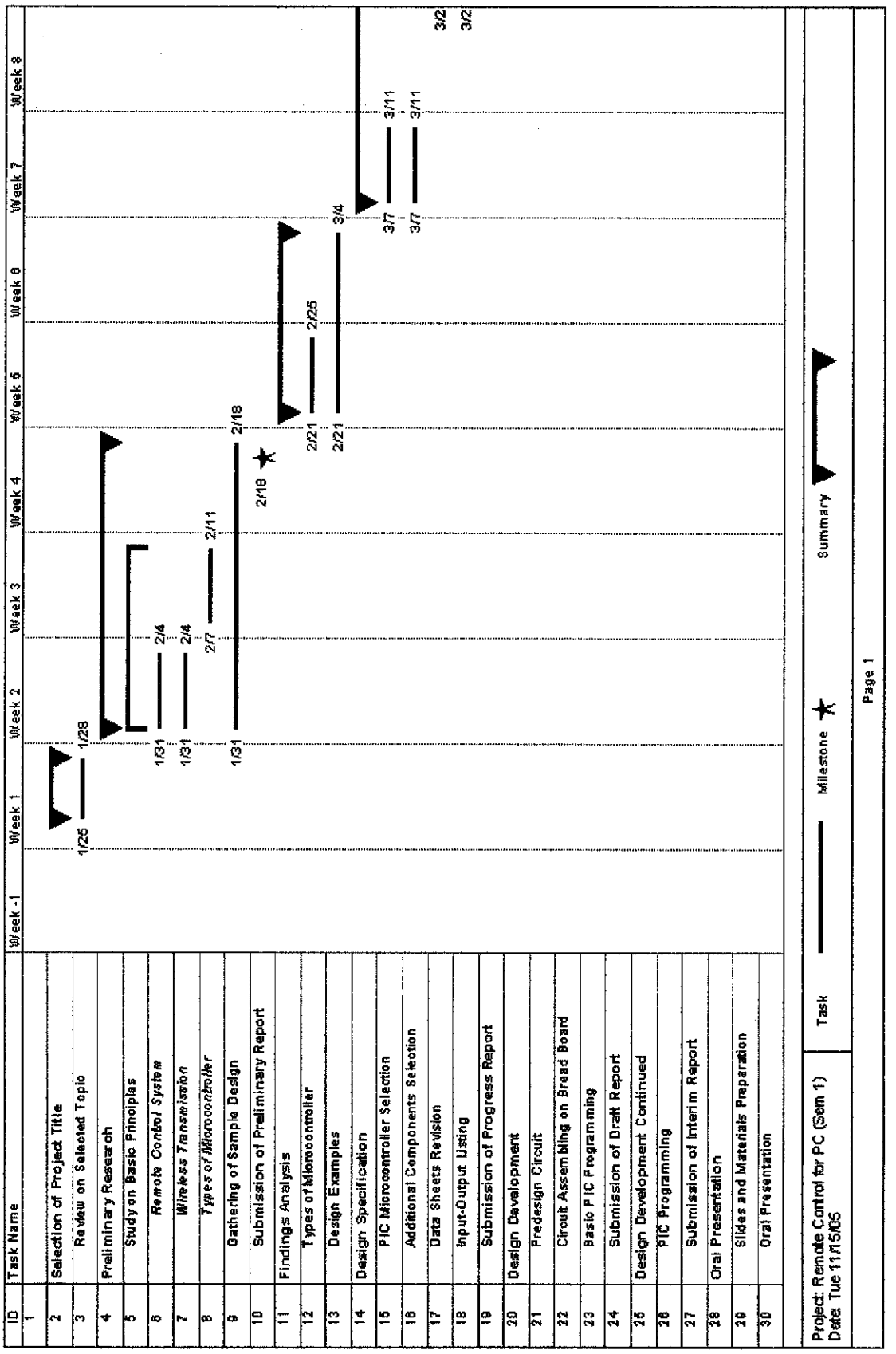
[28] Infrared Data Acquisition Web
<http://www.dataacquisitionweb.com/interfaces/infrared>

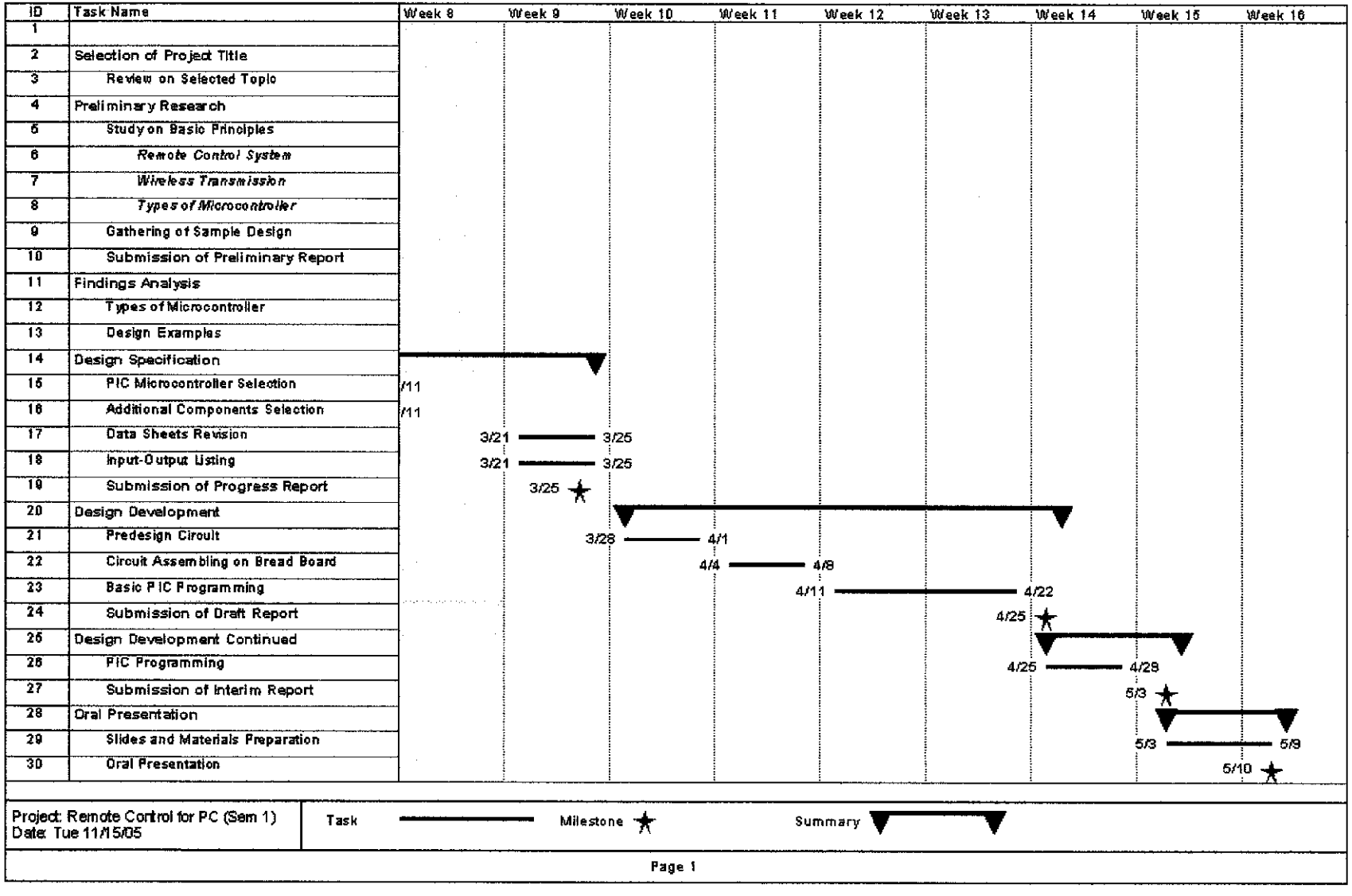
[29] Richard Barnett, Larry O'Cull & Sarah Cox. *Embedded C Programming and the Microchip PIC*, Thomson Delmar Learning, Canada, 2004.

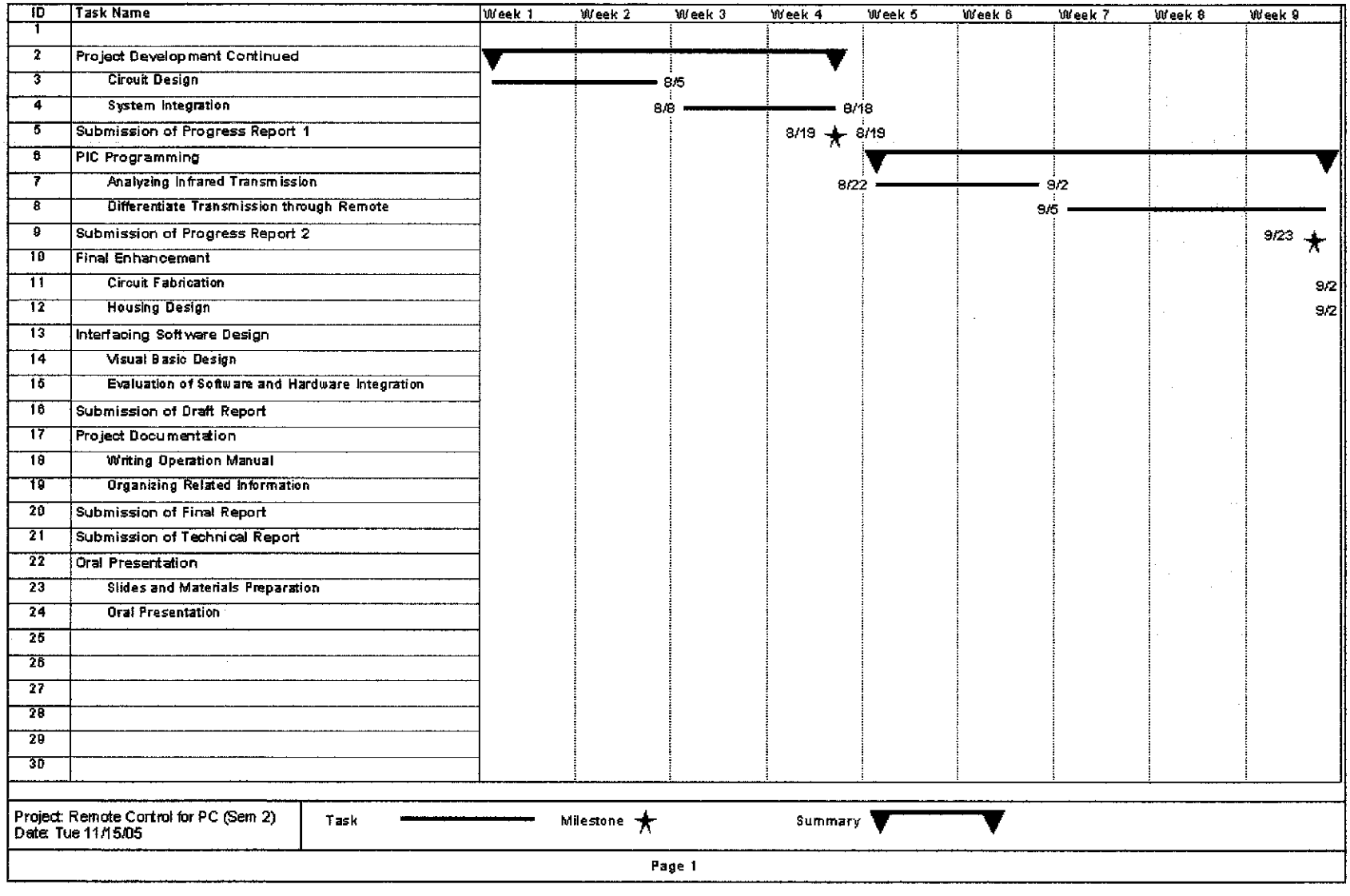
[30] Nazirah Bt. Harun. *A Microcontroller Based Digital Clock with Infrared Remote Control*, Project Final Report, Universiti Teknologi Petronas, 2003.

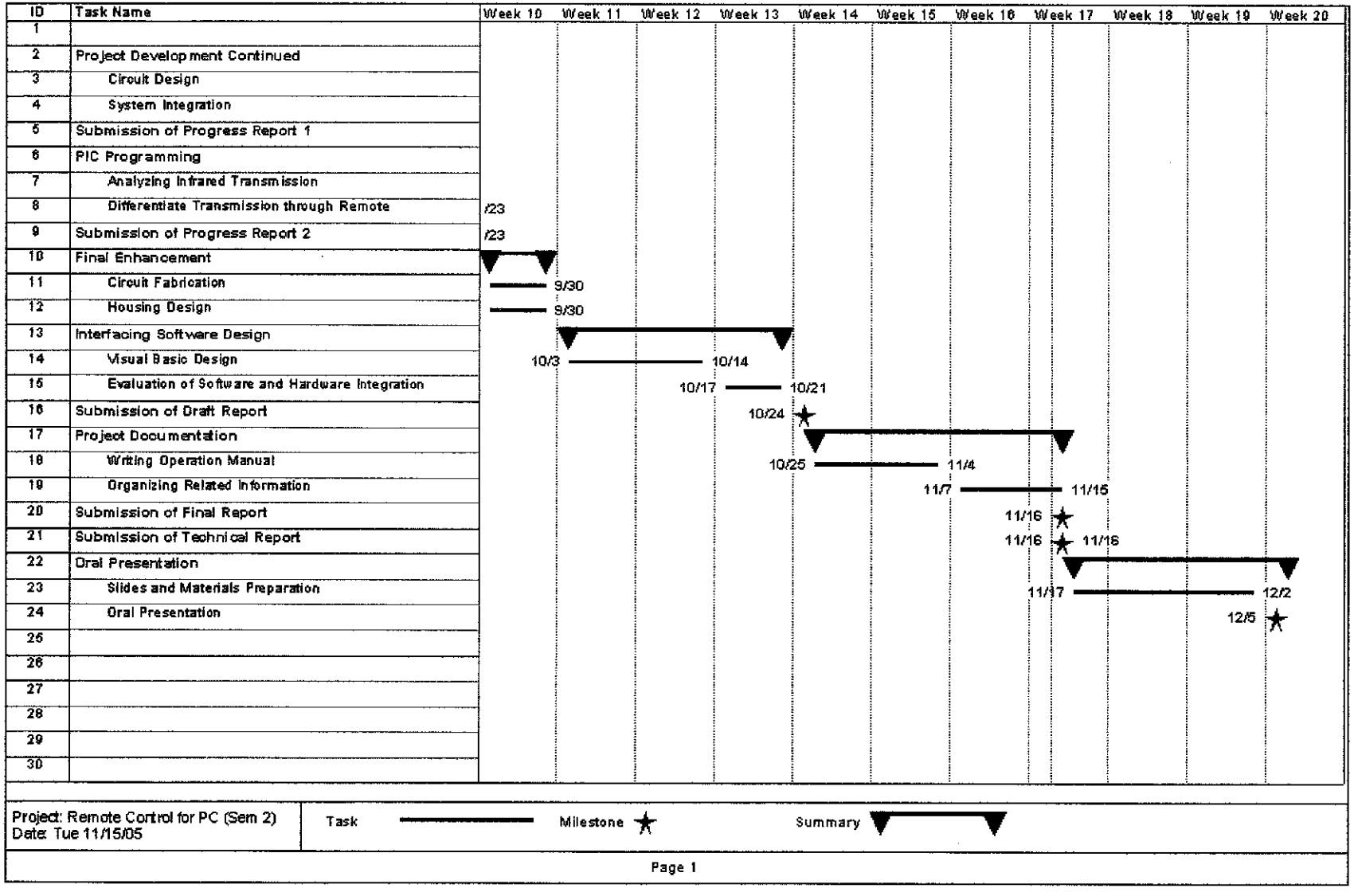
APPENDIX A

PROJECT GANTT CHART



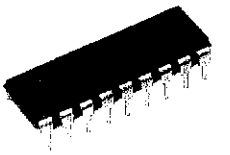
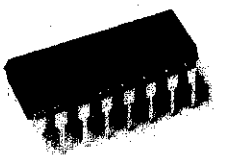
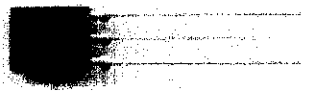

ID
Task Name
Week 8
Week 9
Week 10
Week 11
Week 12
Week 13
Week 14
Week 15
Week 16
1
2 Selection of Project Title
3 Review on Selected Topic
4 Preliminary Research
5 Study on Basic Principles
6 Remote Control System
7 Wireless Transmission
8 Types of Microcontroller
9 Gathering of Sample Design
10 Submission of Preliminary Report
11 Findings Analysis
12 Types of Microcontroller
13 Design Examples
14 Design Specification
15 PIC Microcontroller Selection
16 Additional Components Selection
17 Data Sheets Revision
18 Input-Output Listing
19 Submission of Progress Report
20 Design Development
21 Predesign Circuit
22 Circuit Assembling on Bread Board
23 Basic PIC Programming
24 Submission of Draft Report
26 Design Development Continued
26 PIC Programming
27 Submission of Interim Report
28 Oral Presentation
29 Slides and Materials Preparation
30 Oral Presentation
/11
/11
3/21
3/25
3/21
3/25
3/25
3/28
4/1
4/4
4/8
4/11
4/22
4/25
4/25
4/28
5/3
5/3
5/9
5/10
Project: Remote Control for PC (Sem 1)
Date: Tue 11/15/05
Task
Milestone
Summary
Page 1


ID
Task Name
Week 1
Week 2
Week 3
Week 4
Week 5
Week 6
Week 7
Week 8
Week 9
1
2 Project Development Continued
3 Circuit Design
4 System Integration
5 Submission of Progress Report 1
6 PIC Programming
7 Analyzing Infrared Transmission
8 Differentiate Transmission through Remote
9 Submission of Progress Report 2
10 Final Enhancement
11 Circuit Fabrication
12 Housing Design
13 Interfacing Software Design
14 Visual Basic Design
15 Evaluation of Software and Hardware Integration
16 Submission of Draft Report
17 Project Documentation
18 Writing Operation Manual
19 Organizing Related Information
20 Submission of Final Report
21 Submission of Technical Report
22 Oral Presentation
23 Slides and Materials Preparation
24 Oral Presentation
25
26
27
28
29
30
8/5
8/8
8/18
8/19
8/19
8/22
9/2
9/5
9/23
9/2
9/2
Project: Remote Control for PC (Sem 2)
Date: Tue 11/15/05
Task
Milestone
Summary
Page 1


ID
Task Name
Week 10
Week 11
Week 12
Week 13
Week 14
Week 15
Week 16
Week 17
Week 18
Week 19
Week 20
1
2 Project Development Continued
3 Circuit Design
4 System Integration
5 Submission of Progress Report 1
6 PIC Programming
7 Analyzing Infrared Transmission
8 Differentiate Transmission through Remote
9 Submission of Progress Report 2
10 Final Enhancement
11 Circuit Fabrication
12 Housing Design
13 Interfacing Software Design
14 Visual Basic Design
15 Evaluation of Software and Hardware Integration
16 Submission of Draft Report
17 Project Documentation
18 Writing Operation Manual
19 Organizing Related Information
20 Submission of Final Report
21 Submission of Technical Report
22 Oral Presentation
23 Slides and Materials Preparation
24 Oral Presentation
25
26
27
28
29
30
/23
/23
9/30
9/30
10/3
10/14
10/17
10/21
10/24
10/25
11/4
11/7
11/15
11/16
11/16
11/16
11/17
12/2
12/5
Project: Remote Control for PC (Sem 2)
Date: Tue 11/15/05
Task
Milestone
Summary
Page 1

APPENDIX B

LIST OF COMPONENTS USED

No	Component	Quantity	Figure
1	PIC16F84A microcontroller	1	
2	Maxim MAX232	1	
3	10MHz oscillator	1	
4	Infrared receiver module	1	
5	RS232 female connector	1	
6	Serial cable	1	
7	USB cable	1	
8	22pF capacitor	2	
9	1μF capacitor	4	
10	100Ω resistor	2	
11	10kΩ resistor	1	
12	Red light emitting diode (LED)	1	
13	16-pin IC socket	1	
14	18-pin IC socket	1	
15	Vero board (6cm x 6cm)	1	
16	2-pin connector	1	
17	3-pin connector	1	

APPENDIX C

C CODING FOR INFRARED RECEIVER OPERATIONS

```
#include <16F84A.H>
#fuses HS,NOWDT,NOPROTECT,PUT
#use DELAY(clock=10000000)
#use rs232(baud=9600, xmit=pin_b1, rcv=pin_b5)

#use fast_io(A)
#use fast_io(B)

#define rs_xmit     pin_b1  // (output) RS232 serial transmit
#define led         pin_b2  // (output) Red LED (low true)
#define receiver    pin_b4  // (input) IR sensor
#define rs_rcv      pin_b5  // (input) RS232 serial receive

// ===========================================================================
// General definitions

struct {
  short int RBIF;
  short int INTF;
  short int T0IF;
  short int RBIE;
  short int INTE;
  short int T0IE;
  short int PEIE;
  short int GIE;
} INTCON;
#byte  INTCON = 0x0B

// general I/O buffer
#define BUF_SIZE      32
char gBuf[BUF_SIZE];

// ===========================================================================
// IRDA constants

#define IRDA_TRIS     (IRX_B_TRIS | 0b00000001)

// Minimum IRDA pulse is 1.63 uSec.  With 400 nSec instruction
// cycle time (10MHz clock), this requires 4+ cycles, ergo 5.
#define IRDA_PULSE_CYCLES     5

// RTCC prescaler of 4:1, or 1600 nSec tics
#define IRDA_BIT_TICS         65      // 1/9600 baud
#define IRDA_STARTBIT_TICS    98      // 1.5 * 1/9600 baud
#define IRDA_TIMEOUT_TICS     255

// ===========================================================================
// IR Input

int findIRDAStart() {
  output_high(pin_a1);
  set_rtcc(256-IRDA_TIMEOUT_TICS);
  INTCON.T0IF = 0;

  while (!INTCON.T0IF) {
    if (INTCON.INTF) {
      output_high(pin_a2);

      set_rtcc(256-IRDA_STARTBIT_TICS);
      INTCON.T0IF = 0;
      INTCON.INTF = 0;
      output_low(pin_a1);
      output_low(pin_a2);
      return 1;
    }
```

```
  }
  output_low(pin_a1);
  return 0;
}

int getIRDABit() {
  int seen;
  output_high(pin_a3);
  do {
    output_high(pin_b7);
    seen = INTCON.INTF;
    output_low(pin_b7);
  } while (!INTCON.T0IF);
  set_rtcc(256-IRDA_BIT_TICS);
  INTCON.T0IF = 0;
  INTCON.INTF = 0;
  output_low(pin_a3);
  return !seen;
}

int getIRDAByte() {
  char ch;

  shift_right(&ch, 1, getIRDABit());
  shift_right(&ch, 1, getIRDABit());
  shift_right(&ch, 1, getIRDABit());
  shift_right(&ch, 1, getIRDABit());
  shift_right(&ch, 1, getIRDABit());
  shift_right(&ch, 1, getIRDABit());
  shift_right(&ch, 1, getIRDABit());
  shift_right(&ch, 1, getIRDABit());
  return ch;
}

int getIRDABuf() {
  int count = 0;
  output_high(pin_a0);
  do {
    gBuf[count++] = getIRDAByte();
    if (count == BUF_SIZE) break;
  } while (findIRDAStart());
  output_low(pin_a0);
  return count;
}

// ========================================================================
// RS232 constants

// RTCC prescaler of 4:1, or 1600 nSec tics
#define RS232_TIMEOUT_TICS     255

// ========================================================================
// RS232 Input

int findRS232Start() {
  set_rtcc(256-RS232_TIMEOUT_TICS);
  INTCON.T0IF = 0;

  while (!INTCON.T0IF) {
    if (kbhit()) return 1;
  }
  return 0;
}

int getRS232Buf() {
  int count = 0;
  do {
    gBuf[count++] = getc();
    if (count == BUF_SIZE) break;
  } while (findRS232Start());
  return count;
}
```

```
// ===========================================================================
// Main read/write loop

#define IRDA_SEEN      1
#define RS232_SEEN     0

// loop until an IrDA start bit or a serial start bit is detected.

int waitForStart() {
  output_high(pin_b6);
  while (1) {
    if (INTCON.INTF) {
      set_rtcc(256-IRDA_STARTBIT_TICS);
      INTCON.TOIF = 0;
      INTCON.INTF = 0;
      output_low(pin_b6);
      return IRDA_SEEN;
    } else if (kbhit()) {
      output_low(pin_b6);
      return RS232_SEEN;
    }
  }
}

void main() {
  set_tris_a(0x00);    // all outputs
  set_tris_b(0x30);    // 2 input (pin b4 and b5)

  output_low(pin_a0);
  output_low(pin_a1);
  output_low(pin_a2);
  output_low(pin_a3);
  output_low(pin_a4);

  setup_counters(RTCC_INTERNAL, RTCC_DIV_4);

  output_high(led);    // check for connectivity
  delay_ms(1000);
  output_low(led);

  ext_int_edge(H_TO_L);       // watch for leading edge of IR pulse

  printf("MEL PC REMOTE CONTROL V1.1\r\n");

  while (1) {
    if (waitForStart() == IRDA_SEEN) {
      putRS232Buf(getIRDABuf());
    } else {
      putIRDABuf(getRS232Buf());
    }
  }
}
```

APPENDIX D

VISUAL BASIC CODING FOR INTERFACING SOFTWARE

Coding for Main Form (refer Figure 38)

```
Public program_taskid
Public current_program
Private Declare Function ExitWindowsEx Lib "user32" (ByVal uFlags As Long, ByVal
dwReserved As Long) As Long
Const EWX_LOGOFF = 0 ' parameter for log off
Const EWX_REBOOT = 2 ' parameter for reboot
Const EWX_SHUTDOWN = 1 ' parameter for shutdown

Dim i As Integer ' this variable holds the value of countdown
Dim w As String ' this is the string ...

Private Sub cmd_open_Click(Index As Integer)
    'If error occurred, open find exe dialog box
    On Error GoTo open_dialog

    '-------------------------------------------
    'Execute program



    Exit Sub
    '-------------------------------------------

open_dialog:
    '-------------------------------------------
    'Open dialog box
    CommonDialog1.CancelError = False
    CommonDialog1.Filter = "Executable Files (*.exe)|*.exe"

    CommonDialog1.ShowOpen


    Exit Sub
    '-------------------------------------------

End Sub

Private Sub Command1_Click()
frmAbout.Show
 End Sub

Private Sub Command2_Click()
Load Form10
Form10.Show
End Sub

Private Sub Command4_Click()
Load Form2
Form2.Show
End Sub

Private Sub Command5_Click()
program_4 = "H:\Program Files\Adobe\Acrobat 7.0\Reader\AcroRd32.exe"
program_taskid = Shell(program_4, vbMaximizedFocus)
End Sub

Private Sub Command6_Click()
program_7 = "H:\Program Files\Microsoft Visual Studio\VB98\VB6.EXE"
program_taskid = Shell(program_7, vbMaximizedFocus)
End Sub
```

```
Private Sub Command7_Click()
program_8 = "C:\mirc\bmIRC\bmIRC.exe"
program_taskid = Shell(program_8, vbMaximizedFocus)
End Sub

Private Sub Form_Load()
        current_program = 0

        program_1 = "H:\Program Files\Windows Media Player\wmplayer.exe"
        program_2 = "H:\Program Files\Microsoft Office\Office11\POWERPNT.exe"
        program_3 = "H:\Program Files\PCRC\PCRemote.exe"
        program_4 = "H:\Program Files\Windows NT\hypertrm.exe"
        program_5 = "H:\Program Files\Microsoft Office\OFFICE11\winword.EXE"
        program_6 = "H:\Program Files\Internet Explorer\IEXPLORE.EXE"
        program_7 = "H:\Program Files\Microsoft Visual Studio\VB98\VB6.EXE"
        program_8 = "C:\mirc\bmIRC\bmIRC.exe"


    Call SERIAL_CONNECT

End Sub

Private Sub SERIAL_CONNECT()
    'set the active serial port
    MSComm1.CommPort = 1

    'set the baudrate,parity,databits,stopbits for the connection
    MSComm1.Settings = "9600,N,8,1"

    'enable the oncomm event for every received character
    'RThreshold=1,comEvReceive=enabled
    'RThreshold=0,comEvReceive=disabled
    MSComm1.RThreshold = 1

    'disable the oncomm event for send characters
    'SThreshold=1,comEvSend=enabled
    'SThreshold=0,comEvSend=disabled
    MSComm1.SThreshold = 0

    On Error GoTo errorhandler
    'open the serial port
    MSComm1.PortOpen = True

    'This exit sub is to prevent the normal flow (without error) goes into
errorhandler
    Exit Sub

errorhandler:
    a1 = MsgBox(Err.Description & vbCrLf & "[Error no. = " & Err.Number & "]",
vbExclamation, "Error")

End Sub

Private Sub MSComm1_OnComm()

    Dim rcvd_char As String

    Select Case MSComm1.CommEvent
        Case comEvReceive

            'display incoming event data to displaying textbox
            rcvd_char = MSComm1.Input

            Text1.Text = rcvd_char

            'L (button 1) is pressed
            If rcvd_char = "ÿÿÿÿÿ" Then
```

```
    If current_program = 0 Then
        current_program = 1
        cmd_open_Click (0)
    Else
        SendKeys "{LEFT}", 1
    End If

'R (button 2) is pressed
ElseIf rcvd_char = "ÿ□ÿ□ßÿ□" Then

    If current_program = 0 Then
        current_program = 2
        cmd_open_Click (1)
    Else
        SendKeys "{RIGHT}", 1
    End If

'G (button 3) is pressed
ElseIf rcvd_char = "ÿ□ÿÿ□ÿ□" Then

    If current_program = 0 Then
        current_program = 3
        cmd_open_Click (2)
    Else
        SendKeys "{f5}", 1
    End If

'B (button 4) is pressed
ElseIf rcvd_char = "ÿ□ÿÿßÿÿþ" Then

    If current_program = 0 Then
        current_program = 4
        cmd_open_Click (3)
    Else
        SendKeys "{DOWN}", 1
    End If

'G+L (button 6) is pressed
ElseIf rcvd_char = "ÿ□ÿ□ÿ÷ý" Then

    If current_program = 0 Then
        current_program = 5
        cmd_open_Click (4)
    Else
        SendKeys "%", 1
    End If


'G+R (button 5)is pressed
ElseIf rcvd_char = "ÿÿÿÿÿ" Then

    If current_program = 0 Then
        current_program = 6
        cmd_open_Click (5)
    Else
        SendKeys "{TAB}", 1
    End If

'R+B (button 8) is pressed
```

```
            ElseIf rcvd_char = "ÿ□ÿÿÿ÷ÿ□" Then

                If current_program = 0 Then
                    current_program = 7
                    cmd_open_Click (6)
                Else
                    SendKeys "{ENTER}", 1
                End If

            '0 is pressed
            ElseIf rcvd_char = "ÿ□ÿ□ÿÿÿþ" Then

                If current_program = 0 Then
                    current_program = 8
                    cmd_open_Click (7)
                Else
                    current_program = 0
                    SendKeys "%{F4}", 1
                End If

            End If

    End Select

End Sub

Private Sub cmd_refresh_Click()
    current_program = 0
End Sub

Private Sub office_Click()
program_5 = "H:\Program Files\Microsoft Office\OFFICE11\winword.EXE"
program_taskid = Shell(program_5, vbMaximizedFocus)
End Sub

Private Sub PowerPoint_Click()
program_2 = "H:\Program Files\Microsoft Office\Office11\POWERPNT.exe"
program_taskid = Shell(program_2, vbMaximizedFocus)
End Sub

Private Sub Timer1_Timer()
'depending on the variable w do the
'required tasks
If i = 5 Then '
    Timer1.Enabled = False
    If w = "Shut" Then
        Call ExitWindowsEx(EWX_SHUTDOWN, 0) ' last parameter is reserved and it is
always zero
    ElseIf w = "Reboot" Then
        Call ExitWindowsEx(EWX_REBOOT, 0)
    ElseIf w = "Log" Then
        Call ExitWindowsEx(EWX_LOGOFF, 0)
    End If
    Exit Sub ' exit the sub after completing the task
End If

DoEvents ' just to make windows relax for a while
i = i + 1
PBar.Value = i
```

```
End Sub

Private Sub WinAmp_Click()
program_6 = "H:\Program Files\Internet Explorer\IEXPLORE.EXE"
program_taskid = Shell(program_6, vbMaximizedFocus)
End Sub

Private Sub WMP_Click(Index As Integer)
program_1 = "H:\Program Files\Windows Media Player\wmplayer.exe"
program_taskid = Shell(program_1, vbMaximizedFocus)

End Sub
```

Coding for Exit Form (refer Figure 39)

```
Private Sub Command1_Click()
Unload Me
Unload Form1
End Sub

Private Sub Command2_Click()
Unload Me
End Sub
```

Coding for About Form

```
Option Explicit

' Reg Key Security Options...
Const READ_CONTROL = &H20000
Const KEY_QUERY_VALUE = &H1
Const KEY_SET_VALUE = &H2
Const KEY_CREATE_SUB_KEY = &H4
Const KEY_ENUMERATE_SUB_KEYS = &H8
Const KEY_NOTIFY = &H10
Const KEY_CREATE_LINK = &H20
Const KEY_ALL_ACCESS = KEY_QUERY_VALUE + KEY_SET_VALUE + _
                       KEY_CREATE_SUB_KEY + KEY_ENUMERATE_SUB_KEYS + _
                       KEY_NOTIFY + KEY_CREATE_LINK + READ_CONTROL

' Reg Key ROOT Types...
Const HKEY_LOCAL_MACHINE = &H80000002
Const ERROR_SUCCESS = 0
Const REG_SZ = 1                         ' Unicode nul terminated string
Const REG_DWORD = 4                      ' 32-bit number

Const gREGKEYSYSINFOLOC = "SOFTWARE\Microsoft\Shared Tools Location"
Const gREGVALSYSINFOLOC = "MSINFO"
Const gREGKEYSYSINFO = "SOFTWARE\Microsoft\Shared Tools\MSINFO"
Const gREGVALSYSINFO = "PATH"

Private Declare Function RegOpenKeyEx Lib "advapi32" Alias "RegOpenKeyExA" (ByVal
hKey As Long, ByVal lpSubKey As String, ByVal ulOptions As Long, ByVal samDesired As
Long, ByRef phkResult As Long) As Long
Private Declare Function RegQueryValueEx Lib "advapi32" Alias "RegQueryValueExA"
(ByVal hKey As Long, ByVal lpValueName As String, ByVal lpReserved As Long, ByRef
lpType As Long, ByVal lpData As String, ByRef lpcbData As Long) As Long
Private Declare Function RegCloseKey Lib "advapi32" (ByVal hKey As Long) As Long


Private Sub cmdSysInfo_Click()
  Call StartSysInfo
End Sub

Private Sub cmdOK_Click()
```

```
  Unload Me

End Sub

Private Sub Form_Load()
    Me.Caption = "About " & App.Title


End Sub

Public Sub StartSysInfo()
    On Error GoTo SysInfoErr

    Dim rc As Long
    Dim SysInfoPath As String

    ' Try To Get System Info Program Path\Name From Registry...
    If GetKeyValue(HKEY_LOCAL_MACHINE, gREGKEYSYSINFO, gREGVALSYSINFO, SysInfoPath)
Then
    ' Try To Get System Info Program Path Only From Registry...
    ElseIf GetKeyValue(HKEY_LOCAL_MACHINE, gREGKEYSYSINFOLOC, gREGVALSYSINFOLOC,
SysInfoPath) Then
        ' Validate Existance Of Known 32 Bit File Version
        If (Dir(SysInfoPath & "\MSINFO32.EXE") <> "") Then
            SysInfoPath = SysInfoPath & "\MSINFO32.EXE"

        ' Error - File Can Not Be Found...
        Else
            GoTo SysInfoErr
        End If
    ' Error - Registry Entry Can Not Be Found...
    Else
        GoTo SysInfoErr
    End If

    Call Shell(SysInfoPath, vbNormalFocus)

    Exit Sub
SysInfoErr:
    MsgBox "System Information Is Unavailable At This Time", vbOKOnly
End Sub

Public Function GetKeyValue(KeyRoot As Long, KeyName As String, SubKeyRef As String,
ByRef KeyVal As String) As Boolean
    Dim i As Long                                           ' Loop Counter
    Dim rc As Long                                          ' Return Code
    Dim hKey As Long                                        ' Handle To An Open
Registry Key
    Dim hDepth As Long                                      '
    Dim KeyValType As Long                                  ' Data Type Of A Registry
Key
    Dim tmpVal As String                                    ' Tempory Storage For A
Registry Key Value
    Dim KeyValSize As Long                                  ' Size Of Registry Key
Variable
    '------------------------------------------------------------
    ' Open RegKey Under KeyRoot {HKEY_LOCAL_MACHINE...}
    '------------------------------------------------------------
    rc = RegOpenKeyEx(KeyRoot, KeyName, 0, KEY_ALL_ACCESS, hKey) ' Open Registry Key

    If (rc <> ERROR_SUCCESS) Then GoTo GetKeyError          ' Handle Error...

    tmpVal = String$(1024, 0)                             ' Allocate Variable Space
    KeyValSize = 1024                                       ' Mark Variable Size

    '------------------------------------------------------------
    ' Retrieve Registry Key Value...
    '------------------------------------------------------------
    rc = RegQueryValueEx(hKey, SubKeyRef, 0, _
                         KeyValType, tmpVal, KeyValSize)    ' Get/Create Key Value

    If (rc <> ERROR_SUCCESS) Then GoTo GetKeyError          ' Handle Errors

    If (Asc(Mid(tmpVal, KeyValSize, 1)) = 0) Then               ' Win95 Adds Null
```

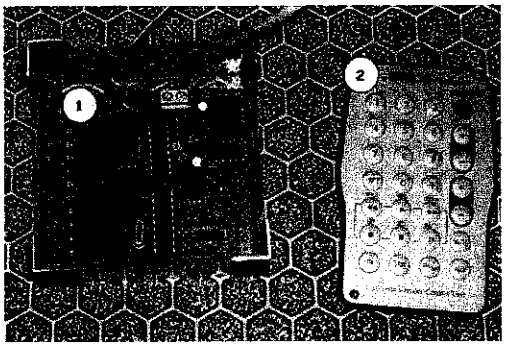
```
Terminated String...
        tmpVal = Left(tmpVal, KeyValSize - 1)                ' Null Found, Extract
From String
    Else                                                     ' WinNT Does NOT Null
Terminate String...
        tmpVal = Left(tmpVal, KeyValSize)                   ' Null Not Found, Extract
String Only
    End If
    '-------------------------------------------------------------
    ' Determine Key Value Type For Conversion...
    '-------------------------------------------------------------
    Select Case KeyValType                                  ' Search Data Types...
    Case REG_SZ                                                 ' String Registry Key
Data Type
        KeyVal = tmpVal                                     ' Copy String Value
    Case REG_DWORD                                            ' Double Word Registry
Key Data Type
        For i = Len(tmpVal) To 1 Step -1                    ' Convert Each Bit
            KeyVal = KeyVal + Hex(Asc(Mid(tmpVal, i, 1)))    ' Build Value Char. By
Char.
        Next
        KeyVal = Format$("&h" + KeyVal)                      ' Convert Double Word To
String
    End Select

    GetKeyValue = True                                      ' Return Success
    rc = RegCloseKey(hKey)                                  ' Close Registry Key
    Exit Function                                           ' Exit

GetKeyError:      ' Cleanup After An Error Has Occured...
    KeyVal = ""                                              ' Set Return Val To Empty
String
    GetKeyValue = False                                     ' Return Failure
    rc = RegCloseKey(hKey)                                  ' Close Registry Key
End Function

Private Sub picIcon_Click()

End Sub

Private Sub lblDescription_Click()

End Sub
```

APPENDIX E

OPERATION MANUAL FOR PC REMOTE CONTROL SYSTEM (HARDWARE AND SOFTWARE)

Chapter 1 Introduction

PC Remote Control System

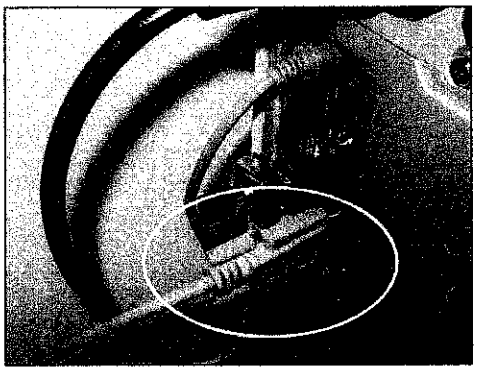


(1) **Infrared Receiver**
Detects the presence of Infrared

(2) **Universal Remote Control**
Transmits IR signals according to the protocol set

Chapter 2 Connecting System to PC

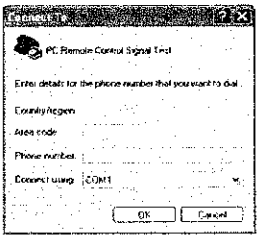
1. The 9-pin connector is connected to the PC serial port (COM1 or COM2) through serial cable.



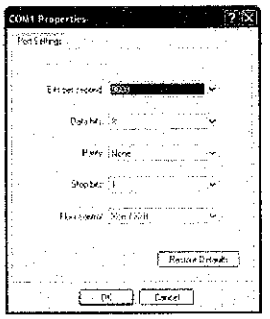
Chapter 3 Setting the Hyper Terminal

1. Open the Hyper Terminal application under **Start/ Accessories/ Hyper Terminal**
2. In the Hyper Terminal dialog box put any type of name desired and press **OK**.
   (E.g.: PC Remote Control Signal Test)

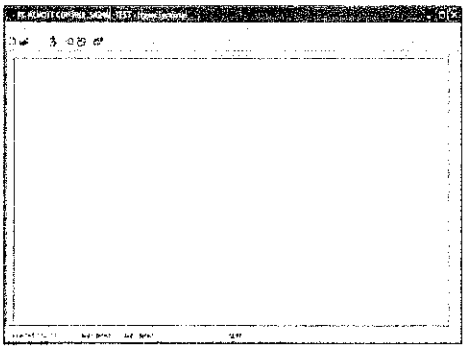
3. Then set at the **Connect Using** command box to **COM1** or **COM2** (Based on serial port connected) and press **OK**.



5. Then set at the respective **COM Properties** as shown below. Then press **APPLY** and **OK**.



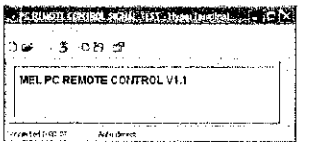
6. Then the Hyper Terminal is ready to be used to test the circuit.



7. The system is ready to be tested

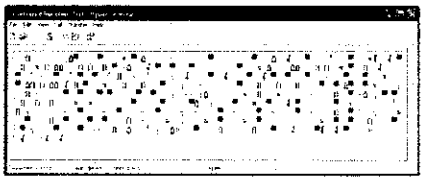
Chapter 4 Retrieving Signal from the System

1. Connect supply cable of the system to PC USB port.
2. If the system is correctly connected, the indicator LED blinks once and the following is observed at Hyper Terminal.



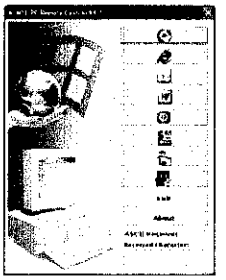
3. Using any type of remote control, press any button on it and observe for ASCII code shown at the Hyper Terminal window.

   (E.g.: Signals received from Philips remote control)



Chapter 5 Utilizing MEL PC Remote Control

1. Run the MEL PC Remote Control application. The main window of the program will be displayed.



The application is controlled using several keyboard buttons represented by the buttons on the remote control; arrows, ALT, TAB, ENTER, F4. For particular application executed, the control on that application will depends on the shortcuts already defined for that application.

Refer the following table in order to use Sony TV Remote Control to control the applications.

Button on Sony TV Remote Control	Function of the button to the program
Button 1	Move LEFT
Button 2	Move RIGHT
Button 3	Move UP
Button 4	Move DOWN
Button 6	ALT button
Button 9	ENTER button
Button 0	ALT + F4 (shortcut for CLOSE)
Button Volume UP(+)	TAB button

2. As an example, to click on About; move the button 1, 2, 3 or 4 to select the about and press button 9 to open the About form.
3. If the Exit button is pressed, the Exit form will appear confirming the termination of the MEL PC Remote Control application.100
4.