

APPLICATION OF VOICE RECOGNITION

By

KAMARUL AZMAN BIN ABU BAKAR

FINAL PROJECT 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

© Copyright 2005

by

Kamarul Azman Bin Abu Bakar 2005

CERTIFICATION OF APPROVAL

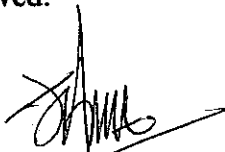
APPLICATION OF VOICE RECOGNITION

by

Kamarul Azman Bin Abu Bakar

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

Approved:



Mohd Azman 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.



Kamarul Azman Bin Abu Bakar

ABSTRACT

Voice recognition or speech recognition is the capability of a computer to recognize or respond to spoken words and commands. Speech & Voice recognition enables “hands-free” control of various electronic devices which is a particular help to many disabled persons and the automatic creation of “print-ready” dictation. Among the earliest applications for speech & voice recognition were automated telephone systems and medical dictation software (Transcription). These applications can be enhanced to wider application where persons can control a system by using voice command.

Therefore to fill up the gap, this project is basically to create a system that can control the part or the whole house appliances by applying voice command added with some other extra features. The extra features included operating windows application, words translation, and reading notes. The house appliances can be lighting, fan, air-conditioner, garage door and etc. This system operates like a switching system where the main objective is to control the application ON/OFF operation. The main different is this system used voice command as the controller rather than manual switching by hand. There will be graphical interface that helps the user to monitor and control the system. Throughout this report, the author will explain in detail about the project.

ACKNOWLEDGEMENTS

First of all I would like to thank Allah the Almighty, for blessing me the whole time in completing this Final Year Project (FYP). I would also like to thank my family for their support and encouragement.

My great thank you to my project supervisor; Mr Mohd. Azman Zakariya for the supervision, guidance, and support. Your advice and comments really help me all the way through understanding and finishing this project.

Last but not least, thank you to lecturer Mr. Zuki for lending me the PIC simulator IDE, lab technicians Miss Yanti and Miss Siti Hawa for guidance in the lab, previous student's report Wan Zahara and all my friends and everyone that has been involved in helping my Final Year Project.

THANK YOU.

TABLE OF CONTENTS

LIST OF TABLES.....	viii
LIST OF FIGURES	ix
LIST OF ABBREVIATIONS.....	xi
CHAPTER 1 INTRODUCTION.....	12
1.1 Background of Study.....	12
1.2 Problem Statement and Identification	13
1.3 Objectives of the project	14
1.4 Scope of study	14
1.4.1 Speech recognition software.....	15
1.4.2 Microsoft Visual Basic 6.0	16
1.4.3 PIC	19
CHAPTER 2 LITERATURE REVIEW AND THEORY	20
2.1 Detecting input voice	20
2.2 Speech recognition	21
CHAPTER 3 METHODOLOGY AND PROJECT WORK	23
3.1 Procedure Identification	23
3.2 Setting up Input	25
3.2.1 Setting Microphone	25
3.2.2 Sound card	29
3.2.3 Training speech recognition software	30
3.3 Microsoft Visual Basic 6.0.....	34
3.3.1 Link VB to Microsoft Access.....	35
3.3.2 Build GUI	39
3.3.3 Communication port.....	41
3.3.4 Serial Communication Interface.....	42
3.4 Output interface.....	44
3.4.1 PIC Simulator IDE	44
3.4.2 Warp13 software	45
3.4.3 HyperTerminal application.....	46

CHAPTER 4 RESULTS AND DISCUSSION	48
4.1 Results.....	48
4.2 GUI.....	49
4.2.1 Controlling other application.....	50
4.2.2 Translation and Read Notes	51
4.2.3 Home Automation	52
4.3 Choosing microphone	53
4.4 Programming PIC16F84	54
CHAPTER 5 CONCLUSION AND RECOMMENDATION	55
5.1 Challenges and problems	55
5.2 Recommendation.....	56
REFERENCES	57
APPENDICES	58
Appendix A PIC16F84 Connection	59
Appendix B MAX232 Connection.....	60
Appendix C Table of Ascii code.....	61
Appendix D Table of ascii code.....	62
Appendix E User Manual to Operate the system	63
Appendix F Visual Basic Source Code.....	64
Appendix G Coding for PIC.....	65
Appendix H PIC Simulator Manual	69

LIST OF TABLES

Table 1 Visual Basic Naming Conventions.....	18
Table 2 Types of microphone.....	27

LIST OF FIGURES

Figure 1 Basic idea of project application.	13
Figure 2 Voice Command Option for Microsoft Speech 4.0.....	15
Figure 3 Accuracy Center for Dragon NaturallySpeaking 8.0	16
Figure 4 Visual Basic startup.....	17
Figure 5 Example picture of PIC.	19
Figure 6 Converting voice input into binary code.	20
Figure 7 Determining speech recognition.....	21
Figure 8 Block Diagram of Project Work.....	23
Figure 9 Project layout.....	24
Figure 10 Diagram inside the microphone.	25
Figure 11 A picture of sound card installed in PC.....	29
Figure 12 An analog-to-digital converter measures sound waves at frequent intervals.	29
Figure 13 First setup to train DNS.....	30
Figure 14 Window for new user wizard.	31
Figure 15 Selecting text for training.....	31
Figure 16 Training window.	32
Figure 17 DNS application.	32
Figure 18 Block diagram of visual basic.	34
Figure 19 Window to add new component.....	35
Figure 20 Window of Data Link Properties.....	36
Figure 21 Connection destination.	37
Figure 22 Property page for RecorSource.	37
Figure 23 Microsoft Access.....	38
Figure 24 Table for database.	38
Figure 25 Create GUI.	39
Figure 26 Visual Basic editor in view code mode.....	40
Figure 27 Options to add new component.....	41
Figure 28 Visual Basic editor with added component.....	41
Figure 29 RS-232 Pin Out on a DB-9 Pin Used for Asynchronous Data.....	43
Figure 30 PIC Simulator IDE software.....	45

Figure 31 Warp13 PIC programmer.	46
Figure 32 The location of HyperTerminal application.	47
Figure 33 HyperTerminal window.	47
Figure 34 Results block diagram.	48
Figure 35 Main menu of the project.	49
Figure 36 Section 1 of the project.....	50
Figure 37 Section 2 of the project.....	51
Figure 38 Section 3 of the project.....	52
Figure 39 BASIC Compiler window.	54

LIST OF ABBREVIATIONS

PC	-	Personal Computer
PIC	-	Peripheral Interface Controller
DNS	-	Dragon NaturallySpeaking
ADC	-	Analog-to-Digital Converter
S/N	-	Signal-to-Noise ratio
Tx	-	Transmitter
Rx	-	Receiver
SDK	-	Software Development ToolKit
VB	-	Visual Basic
RAD	-	Rapid Application Development
GUI	-	Graphical User Interface
DAO	-	Data Access Object
RDO	-	Remote Data Object
ADO	-	ActiveX Data Object

CHAPTER 1

INTRODUCTION

1.1 Background of Study

Voice recognition or speech recognition is the capability of a computer to recognize or respond to spoken words and commands. Speech & Voice recognition enables “hands-free” control of various electronic devices which is a particular help to many disabled persons and the automatic creation of “print-ready” dictation.

Among the earliest applications for speech & voice recognition were automated telephone systems and medical dictation software (Transcription). Speaking to the computer can be used in way of typing or mouse clicking to input information to the computer.

There are two major types of speech recognition programs which are speech-to-text and command-and-control. Speech-to-text systems transfer spoken words into written words for use in any applications such as word processing programs. It can be Notepad, Microsoft Office Word and a lot more.

Meanwhile, command-and-control systems interpret spoken words as commands, for example, to open a new program or save a file and it can be any command that is programmed.

1.2 Problem Statement and Identification

Nowadays, there is lots of voice recognition software in the market. As explained before the software can do lots of task that resemble the keyboard and mouse. Not only for dictation and command, it also can reduce the time taken to type on keyboard and mouse clicking.

However, its capability can be applied only for dictation and command to do simple works. When this advantage combined with the visual basic programming and microcontroller PIC programming more advance technologies can be made. It can control hardware application such as home automation or car automation and etc.

Moreover, the project can be enhanced to manipulate the power of speech recognition software to become the translation program and to read notes. The basic concept is as figure below.

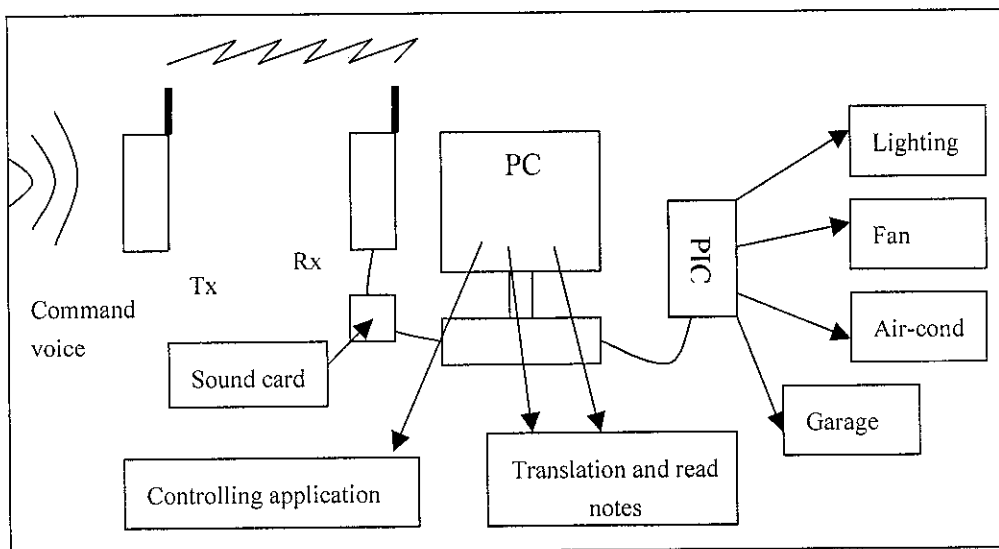


Figure 1 Basic idea of project application.

The figure above shows that the transmitter takes the voice as input and transmits to the receiver which connects to the sound card installed in PC. There will be some operation in the sound card to convert the input voice into digital signal. Then the signal triggers the command and display output based on the input.

The output can be in the PC itself which is controlling other application, read notes or translate the input to Malay text. Other output is to control the home appliances such as lighting, fan, air-condition, garage and etc.

1.3 Objectives of the project

- To have voice recognition system that can dictate spoken words to text
- To create a programmable circuit that has speech command as input
- To have user interface application between user and home automation system
- The program can translate word from English to Malay

1.4 Scope of study

There are several topics and issues that must be considered before proceeding any further in the design of the system. The scope of study depends mainly on these few areas:

- controlling the speech as input and produce the dictation as output
- interfacing the controller to the PC via serial communication
- interface at the PC for the exchange of inputs and outputs from the controller

The applications and devices that are considered must be reliable, cost-effective and practical for a feasible implementation. This will ensure that the design produced is economical and marketable.

1.4.1 Speech recognition software

Firstly, through research on available speech recognition software in market, there are two most popular software which are Microsoft Speech 4.0 and Dragon NaturallySpeaking 8.0. To choose the best software, both speech recognition need to test out and made the comparison.

For Microsoft Speech 4.0, it came with the Software Development ToolKit (SDK) which enables the programmer to add the speech recognition tools in their program. This way the program itself contains the tools for speech recognition. However after several experiment, it shows that the voice detection is less accurate and less user friendly than DNS.

The figure shown below is one of the options of the Microsoft Speech 4.0. Compared to the DNS it does not have the training tool to make the recognition more accurate as shown in the figure 3.

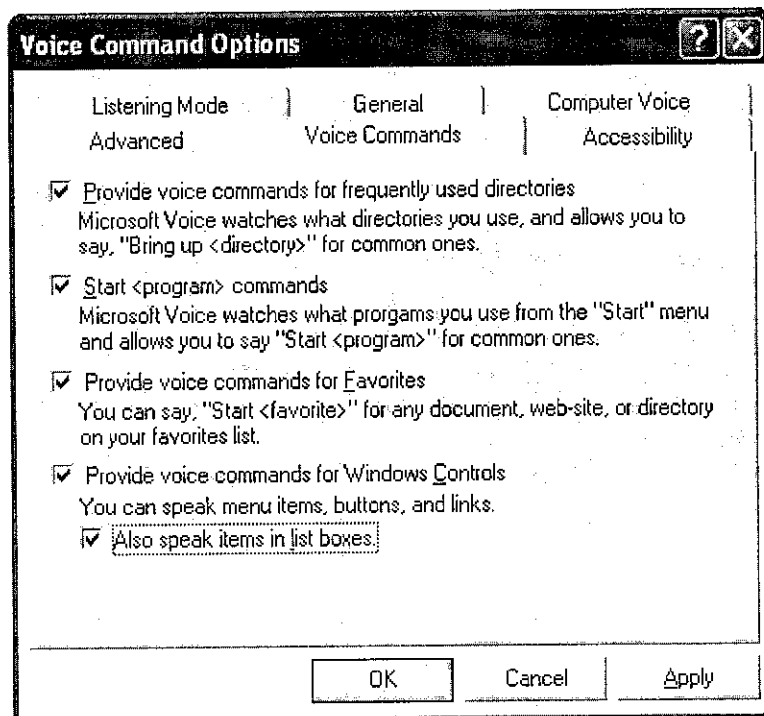


Figure 2 Voice Command Option for Microsoft Speech 4.0

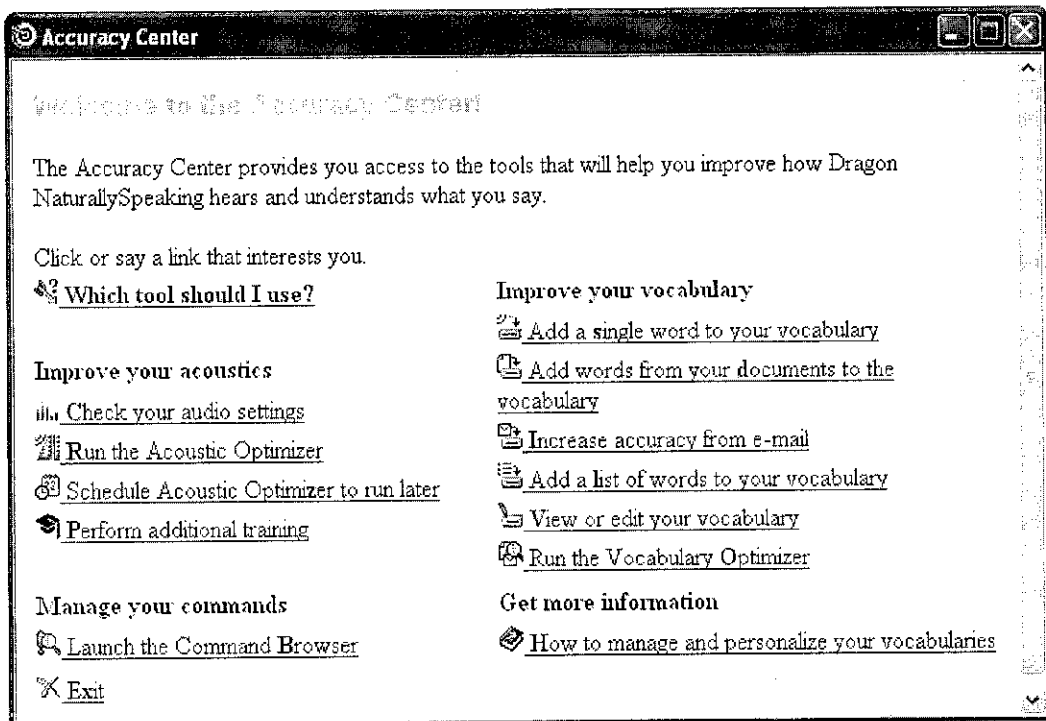


Figure 3 Accuracy Center for Dragon NaturallySpeaking 8.0

Despite having more accurate recognition and accuracy center to make the recognition even accurate, DNS software less one part which is not having the SDK application included for free. From the research and study the SDK only included within the professional version. Moreover, the price for Dragon NaturallySpeaking SDK is \$695.00 and due to the cost constraint this is hard to achieve.

1.4.2 Microsoft Visual Basic 6.0

Another main scope in this project is to study the software that can build the graphical user interface and easy to handle. Therefore, Microsoft Visual Basic 6.0 is chosen to fulfill the need. Visual Basic (VB) is an event driven programming language and associated development environment created by Microsoft.

It is derived heavily from BASIC and enables Rapid Application Development (RAD) of graphical user interface (GUI) applications, access to databases using Data Access Object (DAO), Remote Data Object (RDO), or ActiveX Data Object (ADO), and creation of ActiveX controls and objects. A programmer can put together an application using the components provided with Visual Basic itself.

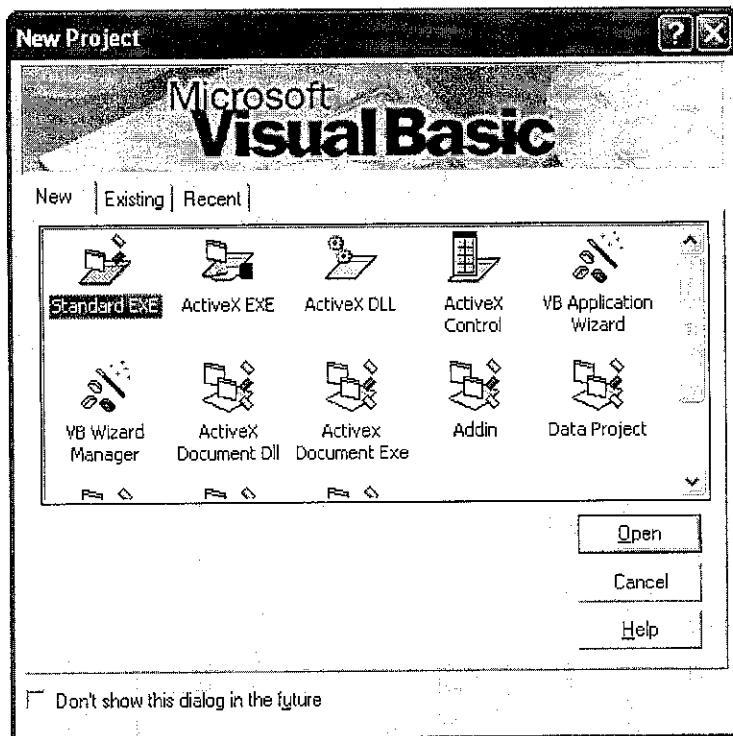


Figure 4 Visual Basic startup.

Visual Basic was designed to be usable by all programmers, whether novice or expert. The language is designed to make it easy to create simple GUI applications, but also has the flexibility to develop fairly complex applications as well. Forms are created using drag and drop techniques. A tools palette is used to place controls (e.g., text boxes, buttons, etc.) on the form (window). Controls have attributes and event handlers associated with them.

For example, code can be inserted into the form resize event handler to reposition a control so that it remains centered on the form, expands to fill up the form, etc. By inserting code into the event handler for a keypress in a text box, the program can automatically translate the case of the text being entered, or even prevent certain characters from being inserted.

Table 1 Visual Basic Naming Conventions

Object	Prefix
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
Shape	shp

1.4.3 PIC

The Peripheral Interface Controller (PIC) is a complete computer on a chip. Such chips can be built into a device to make the product more intelligent and easier to use. PIC chips are programmed to perform only one very specific task, e.g. - a microwave oven may use a single micro-controller to process information from the keypads, display user information on the seven segment display, and control the output devices (turntable motor, light, bell and magnetron).

By altering the software program, the same PIC chip can be used to complete different tasks. The same type of chip can therefore be used in a range of different products by simply programming them with a different software program. There are many types of PIC depending on their usage and programming.

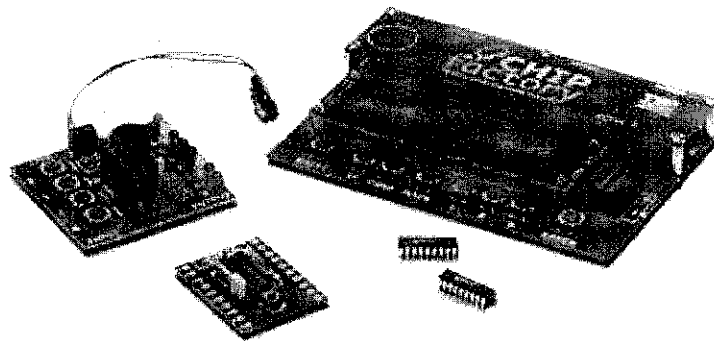


Figure 5 Example picture of PIC.

The PIC16F84 microcontroller is made the microcontroller of choice for the specified task. This choice is obvious as this microcontroller is competitively priced, not to mention its sound applicability in this system. It is also good to note that this microcontroller comes with many interesting features.

CHAPTER 2

LITERATURE REVIEW AND THEORY

Main parts for using the speech recognition by a personal computer requires a microphone for audio input, a sound card to process the input, and software including a speech engine. Before going any further the theory behind all this parts and how it works had to be studied.

2.1 Detecting input voice

This is how the way it works in speech recognition. Firstly, the person voice which is speech sounds begin as vibrations in the throat that cause disruptions in the nearby air pressure. A graph of the pressure changes over time is called a waveform.

Then audio input that is a microphone converts the pressure changes into electrical voltage changes (an analog signal). The voltage then received by a computer sound card which is an analog-to-digital converter that converts the voltage changes back into pressure changes and stores them as binary code (digital) on the hard drive.

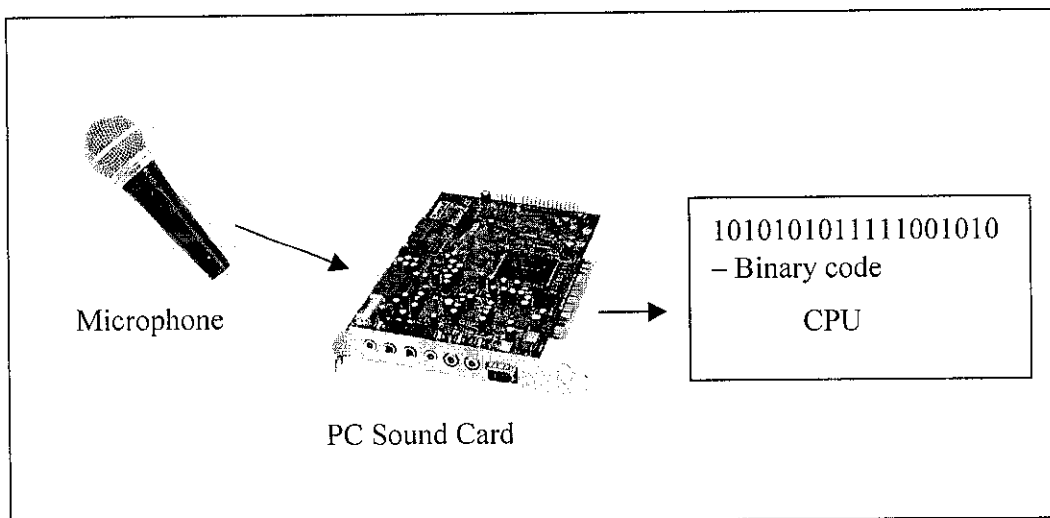


Figure 6 Converting voice input into binary code.

2.2 Speech recognition

From this part, the computer reads the spectrogram patterns and compares them to known patterns for particular sounds or phonemes. A spectrogram is a graph of frequency over time. A phoneme is a distinct speech sound.

This includes individual letters, like "b" or "p" and sounds like "ch" or "ng." Some vowels in particular can have different sounds depending on their usage, e.g., the words "book" and "boot" have two different "oo" phonemes [3]. Some speech recognition programs require discrete speech, i.e., a pause after each and every word.

Others are natural language systems that allow a speaker to speak continuously. They use contextual information to choose which words make sense and to distinguish homophones such as "to," "too," and "two." The programs store frequently used words in an active vocabulary list or dictionary in the computer's random-access memory (RAM) [3]. A larger back-up dictionary, which can contain hundreds of thousands of words, is stored on the hard drive.

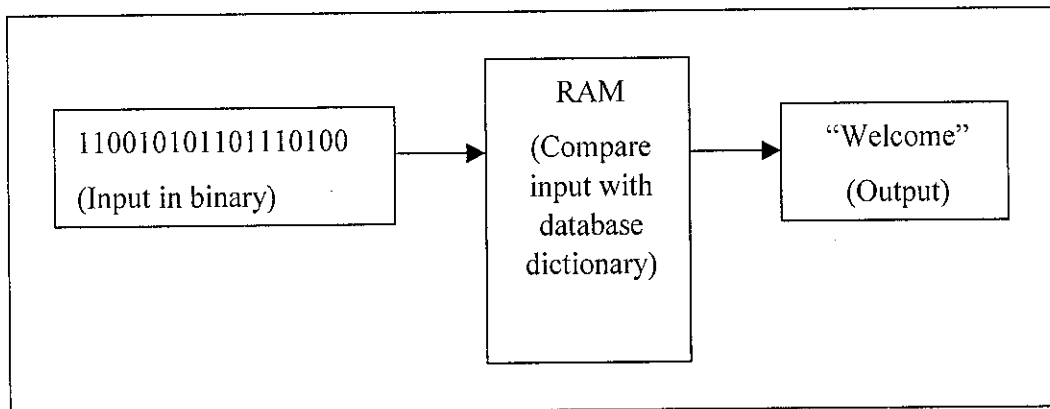


Figure 7 Determining speech recognition.

Some programs are speaker dependent. They are designed to become familiar with one particular speaker's voice and language patterns. The user initially trains the program by reading exercises and identifying mistaken identifications. Eventually the program develops a reliable voice file for that individual. Although the training can be time consuming, a large vocabulary is possible with this type of program. A speaker independent system requires no initial training. It comes equipped with a limited vocabulary and can conceivably recognize any person's speech.

The best speech recognition programs on the market achieve word recognition rates of approximately 90 percent [2]. Roughly this means one mistaken word per sentence. Voice tone, pitch, inflection, and rate of speaking are all determining factors. Speakers with voice disorders, colds, or heavy accents can easily be misunderstood. Best performance is obtained when the program vocabulary is limited, background noise is minimized, and the speaker has a clear consistent voice with good articulation and enunciation [1].

Speech recognition technology is improving quickly and has become an area of focus for many software designers. The latest innovations are Internet voice portals. These systems use Voice Extensible Markup Language (VXML) to direct web browsers to URL addresses that are spoken rather than typed [2]. Many experts predict that the keyboard and mouse will be obsolete within a decade. Growing consumer demand for voice responsive computers and technological advances in neural networks and artificial intelligence are expected to result in computers that not only recognize spoken words, but comprehend their meaning and can communicate with their human counterparts.

CHAPTER 3

METHODOLOGY AND PROJECT WORK

3.1 Procedure Identification

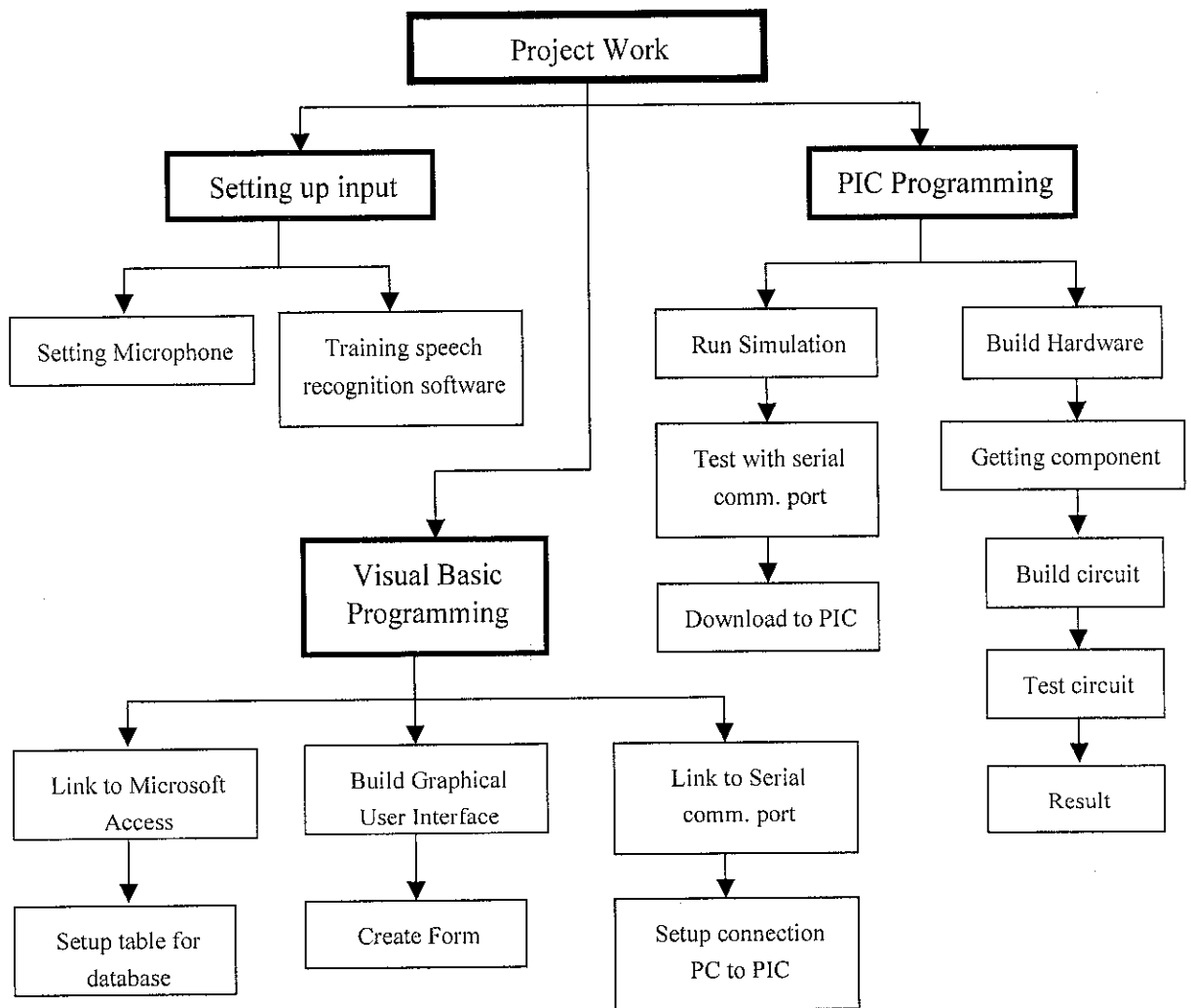


Figure 8 Block Diagram of Project Work.

Procedure identification is very important to keep the project always on the track which will help the project finish on time. Those procedures for the implementation had to pass several systematic steps. Each step had to finish before move on to another step to make sure the project success.

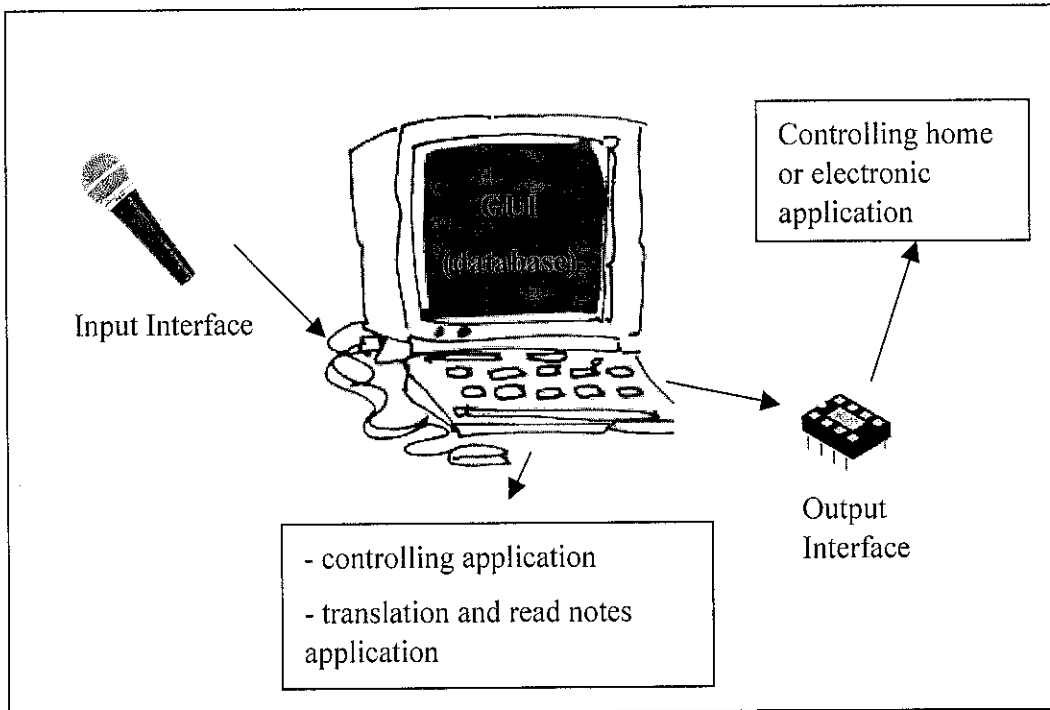


Figure 8 Project layout.

The figure above shows the main parts in the project. The basic structure for the project system is illustrated as in figure above. The system consists of:

1. Input interface, which consists of microphone, sound card and voice recognition software to detect speech and change to text.
2. Graphical User Interface (GUI) that allow user to store database and monitor the system in details.
3. Output interface consists of microcontroller circuit that connected with the controlled devices such as lamp, fan and motor.

3.2 Setting up Input

There are two parts to consider for input interface which are hardware and software. For hardware part it consists of microphone and sound card while speech recognition for the software part. Details in input interface project work are described below.

3.2.1 Setting Microphone

Probably the simplest and most used input to detect voice is the microphone. A microphone is an object that transforms acoustical energy (sound) into electrical energy (an audio signal). There are many ways to do this, and many types of microphones.

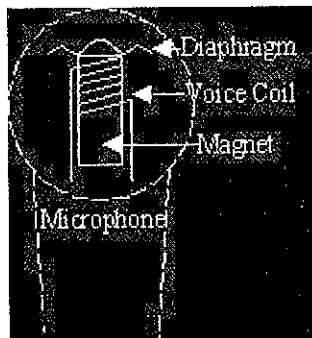


Figure 9 Diagram inside the microphone.

A microphone wants to take varying pressure waves in the air and convert them into varying electrical signals. There are five different technologies commonly used to accomplish this conversion:

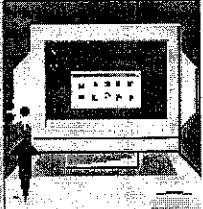
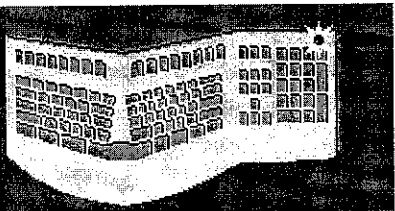
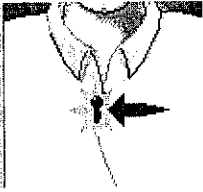

- Carbon microphones - The oldest and simplest microphone uses carbon dust. This is the technology used in the first telephones and is still used in some telephones today. The carbon dust has a thin metal or plastic diaphragm on one side. As sound waves hit the diaphragm, they compress the carbon dust, which changes its resistance. By running a current through the carbon, the changing resistance changes the amount of current that flows.

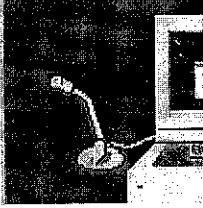

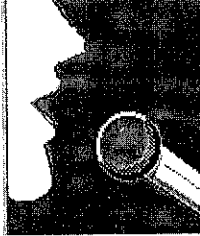
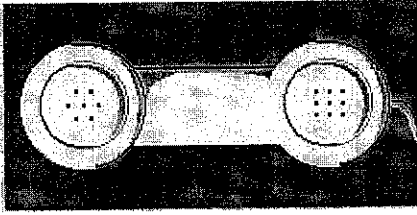
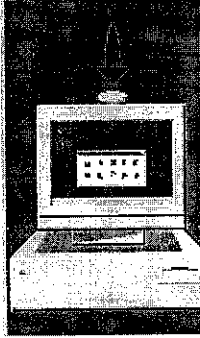
- Dynamic microphones - A dynamic microphone takes advantage of electromagnet effects. When a magnet moves past a wire (or coil of wire), the magnet induces current to flow in the wire. In a dynamic microphone, the diaphragm moves either a magnet or a coil when sound waves hit the diaphragm, and the movement creates a small current.
- Ribbon microphones - In a ribbon microphone, a thin ribbon is suspended in a magnetic field. Sound waves move the ribbon which changes the current flowing through it.
- Condenser microphones - A condenser microphone is essentially a capacitor, with one plate of the capacitor moving in response to sound waves. The movement changes the capacitance of the capacitor, and these changes are amplified to create a measurable signal. Condenser microphones usually need a small battery to provide a voltage across the capacitor.
- Crystal microphones - Certain crystals change their electrical properties as they change shape. By attaching a diaphragm to a crystal, the crystal will create a signal when sound waves hit the diaphragm.

There are several types of microphone that available for the usage in this project.

Table below shows all the types

Table 2 Types of microphone.

Types	Description
<p>➤ Built-into the computer/monitor</p> 	<p>Microphones built-into computers or computer monitors work for command & control speech recognition, but do not work well for dictation because they are too far away and pick up a lot of noise.</p>
<p>➤ Built-into the keyboard</p> 	<p>Microphones built-into keyboards do not work well for speech recognition because they are too far away and pick up a lot of noise.</p>
<p>➤ Clip-on</p> 	<p>Clip-on microphones clip onto shirt, just below the collar. They are not as good as close-talk microphones. Also, the cord sometimes gets in the way.</p>
<p>➤ Close-talk</p> 	<p>Close-talk microphone is wears on the head so that the microphone is as close to mouth as possible. These work the best for speech recognition and dictation because they hear only the person speaking and not any background noise. Unfortunately, the cord sometimes gets in the way.</p>

<p>➤ Desktop</p> 	<p>Desktop microphones usually rest in a microphone stand. If it keeps pointing towards user and about 6" (15cm) from mouth, they work well, but only in quiet rooms.</p>
<p>➤ Ear-piece</p> 	<p>Ear-piece microphones rest on user ear. These work well for speech recognition and dictation but not as well as close-talk microphones. Unfortunately, the cord sometime gets in the way.</p>
<p>➤ Hand-held</p> 	<p>Although hand-held microphones pick up very little noise, they are not convenient for most speech recognition purposes because they must be held 4" (10 cm) away from the user's mouth.</p>
<p>➤ Handset</p> 	<p>These microphones look like normal telephone handsets, but they plug into the microphone jack rather than the telephone jack. Handsets get good accuracy.</p>
<p>➤ Rest on the computer</p> 	<p>Microphones that rest on the computer do not work well for dictation because they are too far away and pick up a lot of noise, but they will work for command and control.</p>

3.2.2 Sound card.

A sound card which is installed in the PC is acting like an Analog-to-Digital Converter (ADC). The ADC translates the analog waves of the voice into digital data that the computer can understand. To do this, it samples, or digitizes, the sound by taking precise measurements of the wave at frequent intervals. This is as shown in figure below.

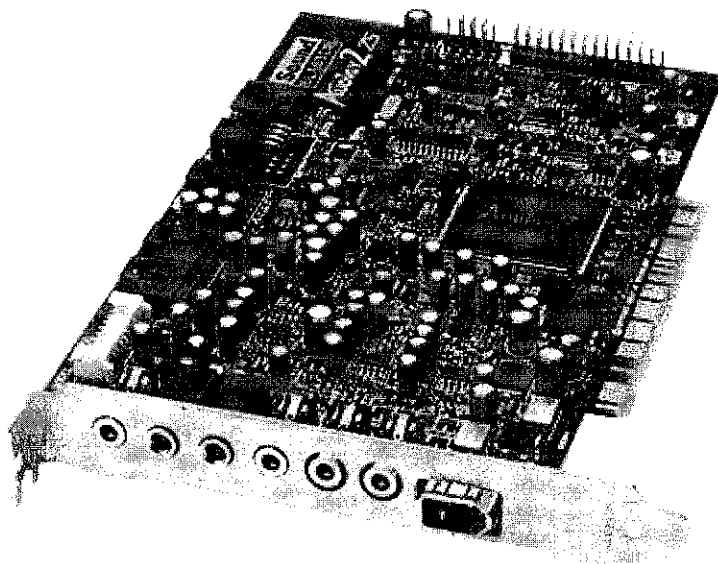


Figure 10 A picture of sound card installed in PC.

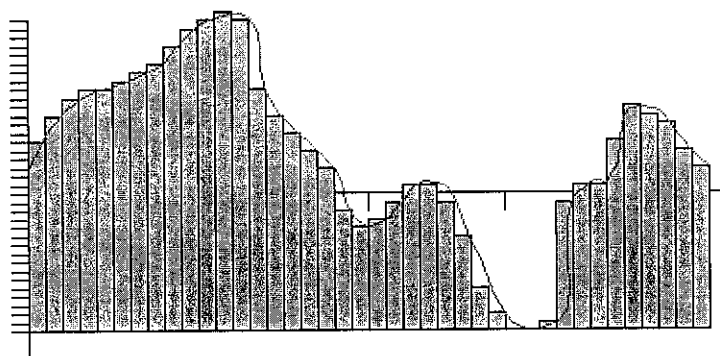


Figure 11 An analog-to-digital converter measures sound waves at frequent intervals.

The number of measurements per second, called the sampling rate, is measured in kHz. The faster a card's sampling rate, the more accurate its reconstructed wave is.

3.2.3 Training speech recognition software

Dragon NaturallySpeaking® 8 enables PC users to create and edit documents and e-mails, fill out forms, and streamline workflow tasks—all by speaking! Talk to the computer and the words instantly and accurately appear in the full Microsoft® Office® Suite, plus Corel® WordPerfect®, Lotus® Notes®, and virtually all other Windows®-based applications.

First of all, the software needs to be trained and this is done by following the New User wizard window shown in the figure below. As shown in the figure, it is recommended to use the close talk microphone and the way to positioning the microphone. No matter what types of microphone used, the correct positioning of the microphone is one of the most important factors in getting good dictation accuracy.

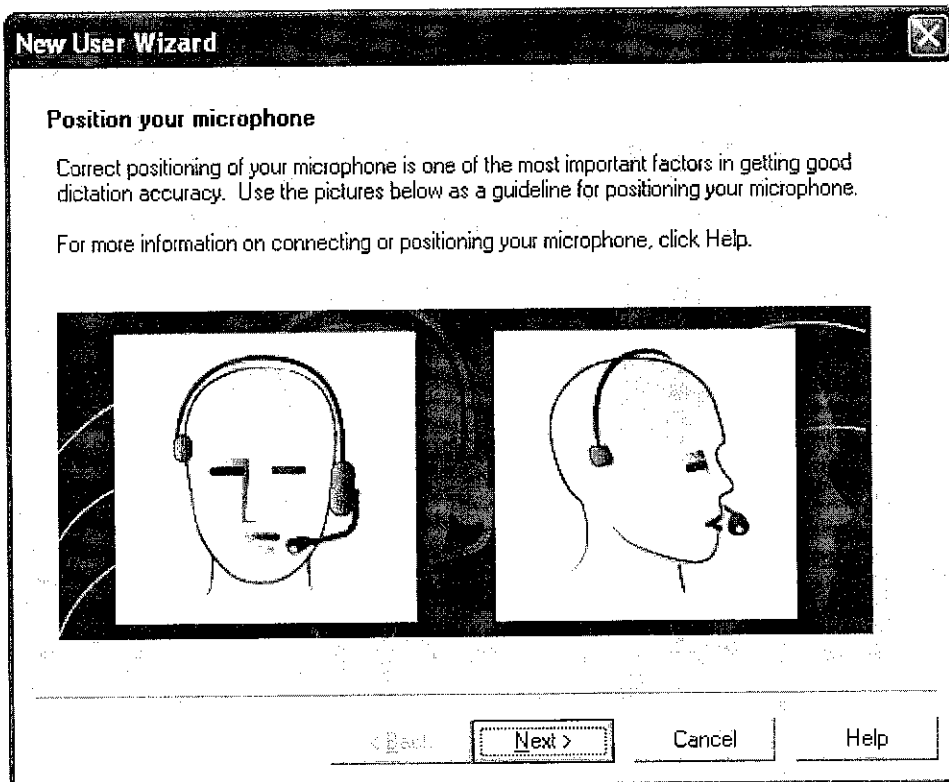


Figure 12 First setup to train DNS.

Then it needs to adjust the volume check of the microphone as shown in the figure below. In this step the computer checks the audio input from the sound system. Having high-quality audio input is very important for good speech recognition. Poor audio input will make it difficult or impossible for the program to recognize speech accurately. When the computer has finished checking the audio quality, it beeps to signal that the test is complete.

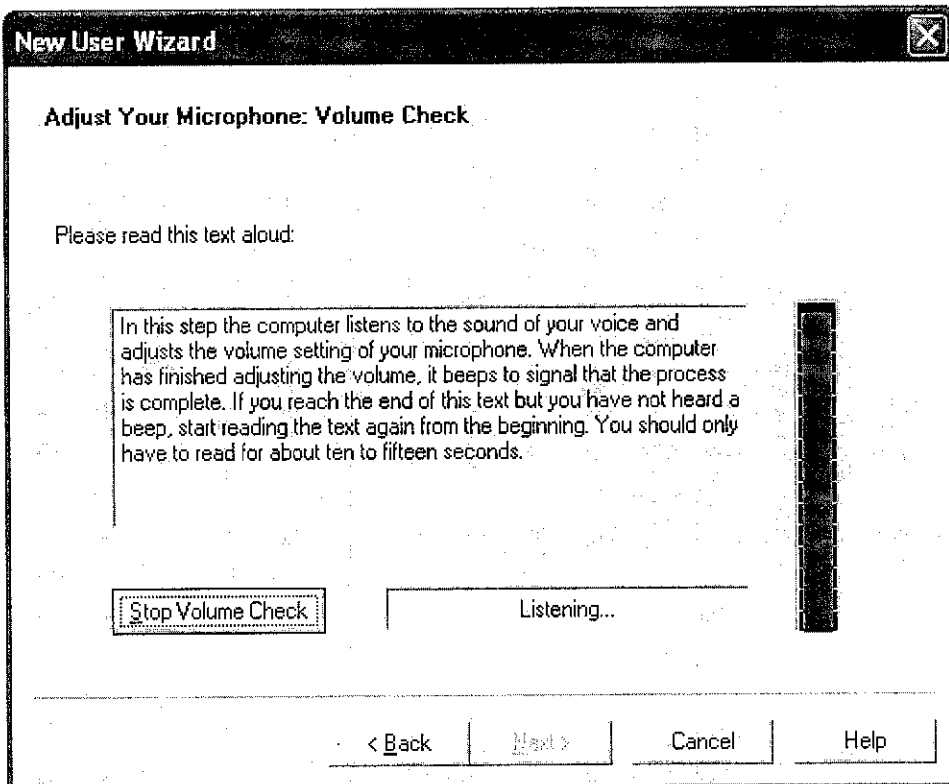


Figure 13 Window for new user wizard.

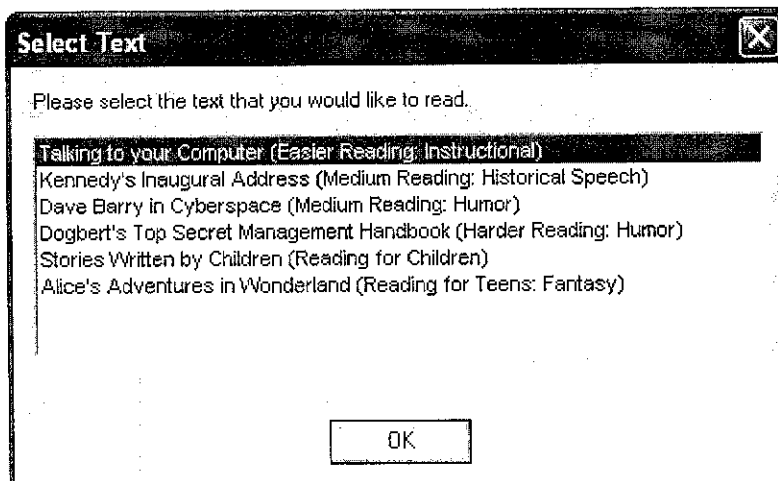


Figure 14 Selecting text for training.

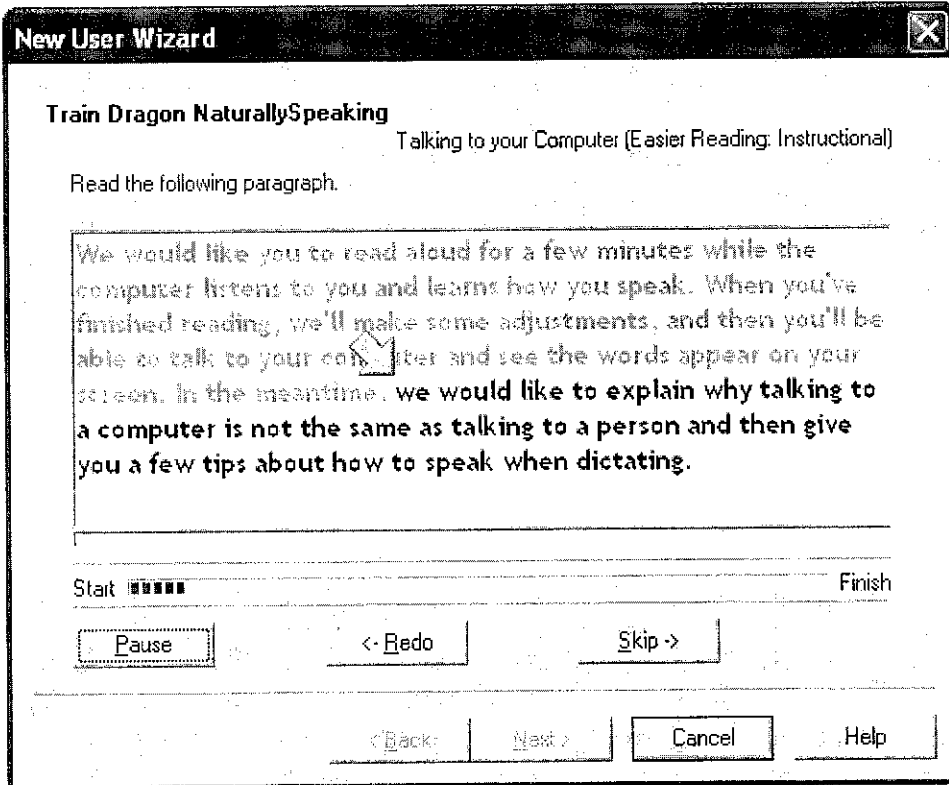


Figure 15 Training window.

After selecting the available text to read as shown in the figure, the training window will appear and here the training will begin. The text provide for training is divided to easier, medium and harder depending on the user. The choice also will affect the recognition accuracy as harder text having more difficult word to pronounce.

After the training session, the DNS is ready to use for dictation. Figure below show the toolbars for DNS application.



Figure 16 DNS application.

There are lots of benefits using DNS software which are:

✓ More Accurate Than Ever Before

With a 25% increase in accuracy, Dragon NaturallySpeaking® 8 is up to 99% accurate and frequently more accurate than typing.

✓ Faster Than Typing

Most people speak over 160 words per minute, but type less than 40 words a minute. That means user can create documents and e-mails about three times faster with Dragon NaturallySpeaking 8.

✓ Safer Than Typing

Dragon NaturallySpeaking 8 will help protect user from repetitive stress injuries like carpal tunnel syndrome - injuries that can cost tens of thousands of dollars.

✓ Easy To Use

Simply install the Dragon NaturallySpeaking CD, and in just a few minutes, user will be creating documents, e-mails and filling out forms by voice! It even included with a free noise-canceling microphone.

3.3 Microsoft Visual Basic 6.0

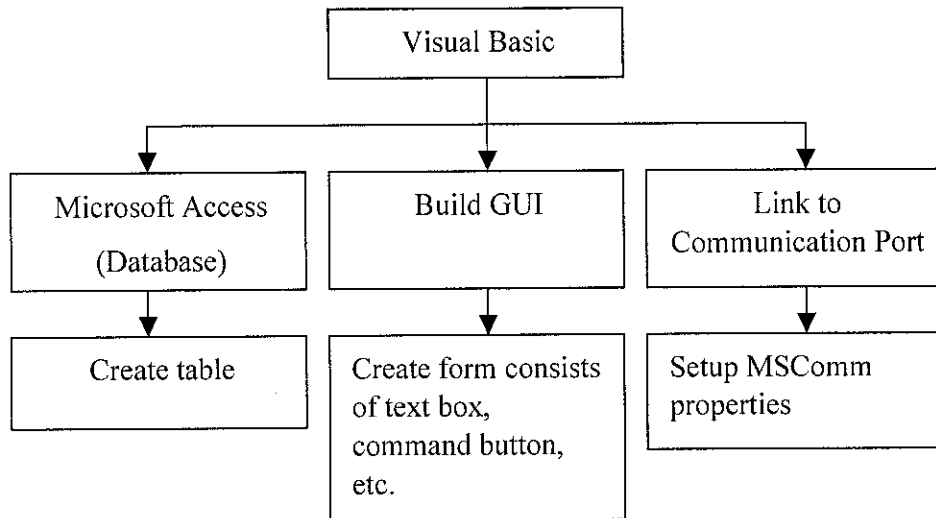


Figure 17 Block diagram of visual basic.

There are three criteria that have to include in order creating the GUI as shown in block diagram above. Firstly, visual basic requires a tool that can interconnect with Microsoft Access which operates as database. It also requires tool to create form or interface which include some coding for user to operate the program. To communicate with serial port, it involves extra component to be able to do so.

In order to write a proper Visual Basic project, there several important elements to learn and understand. The two vital steps are:

1. *Planning*

- Design the user interface
- Plan the properties
- Plan the Basic code - procedures are associated with the events, actions written in pseudocode

2. *Programming*

- Define the user interface - define objects
- Set the properties
- Write the Basic code

3.3.1 Link VB to Microsoft Access.

One of the functions of visual basic is able to link with other application such as Microsoft Access. Therefore, to setup the link visual basic needs to have component that can communicate with Microsoft Access. This is shown in the figure below how to add the component Microsoft ADO Data Control 6.0 (OLEDB).

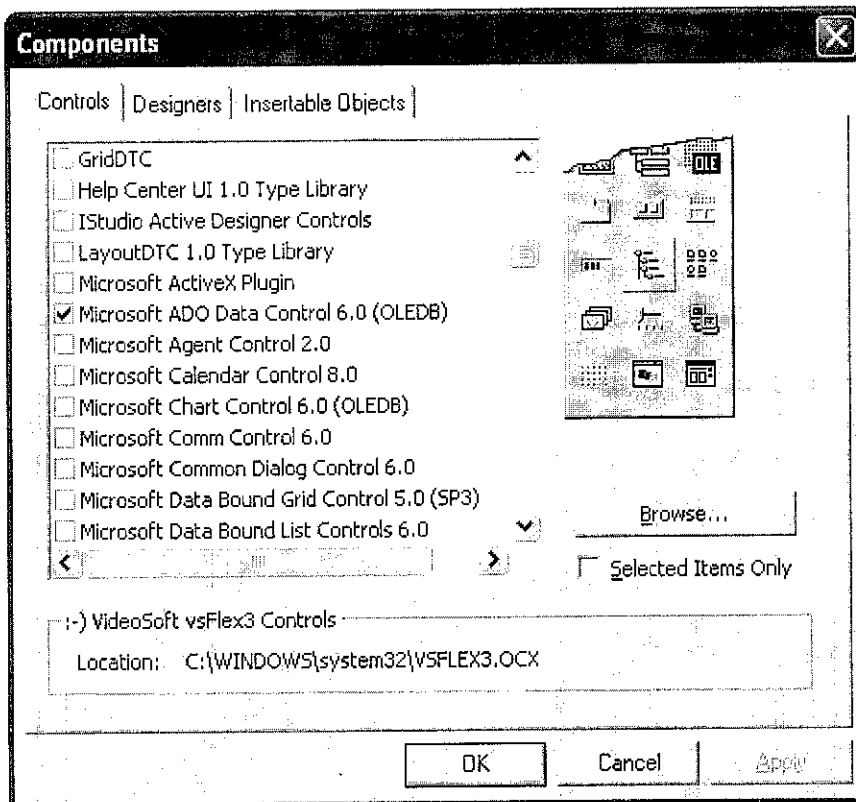


Figure 18 Window to add new component.

Then the component of the Adodc is dragging to the form to activate the link with Microsoft Access. A few properties need to setup ensure the connection with table from Microsoft Access. The properties are shown in the figure below and how to setup it.

The three important settings that must be initialized under the ADO Data Control properties are:

- Connection String; using the Microsoft Jet 4.0 OLEDB Provider, which is the normal choice for real-time application
- Connection Destination; declaring the path of the designated database
- RecordSource; setting the command type and table or stored procedure name to compliment the RecordSet

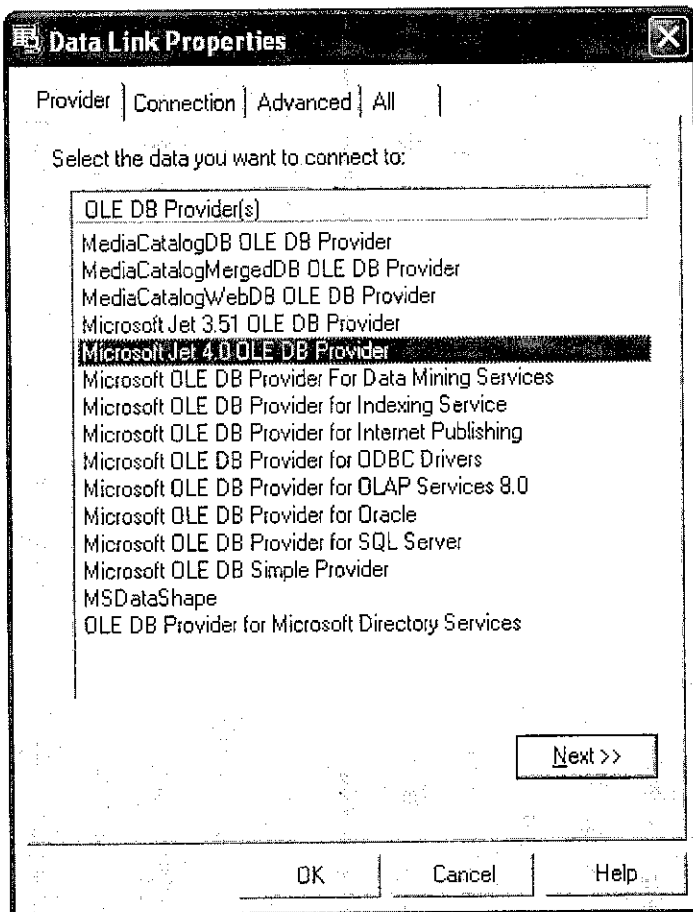


Figure 19 Window of Data Link Properties.

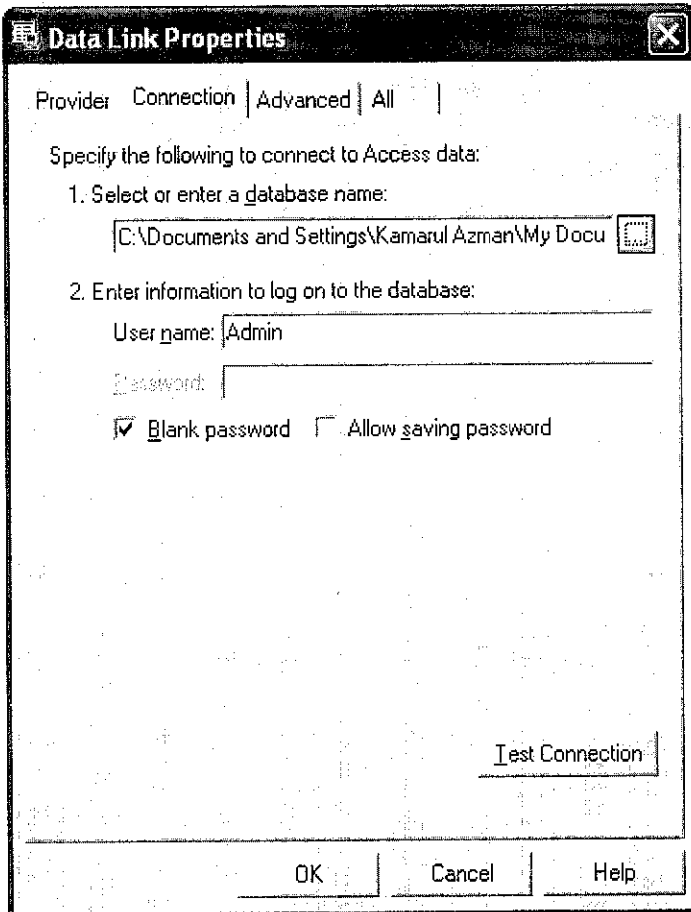


Figure 20 Connection destination.

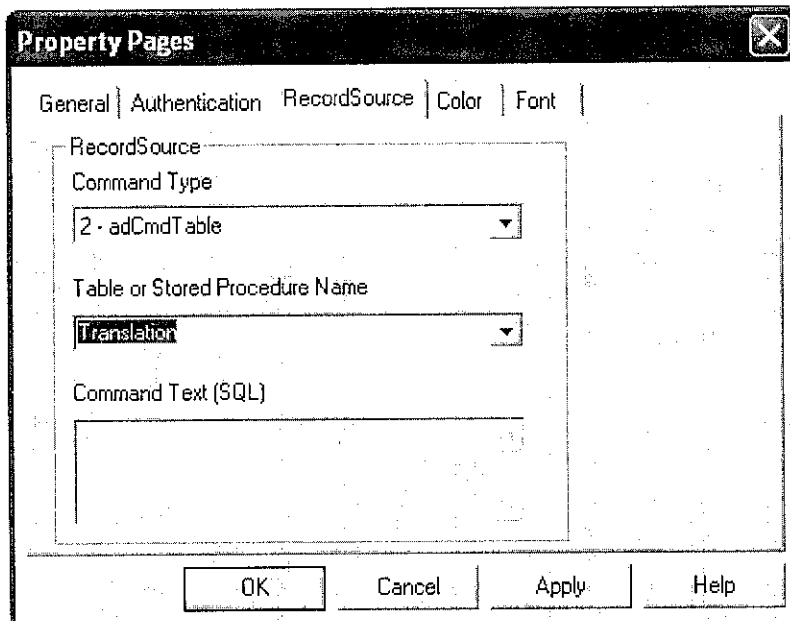


Figure 21 Property page for RecorSource.

The built-in Visual Basic functions proved to be a real boost as it allows Microsoft Access to communicate with Visual Basic to exchange data and provide proper database storage.

For this project the database consists of a translation table that store a few simple words that can be translate for this project. This is presented in figure below.

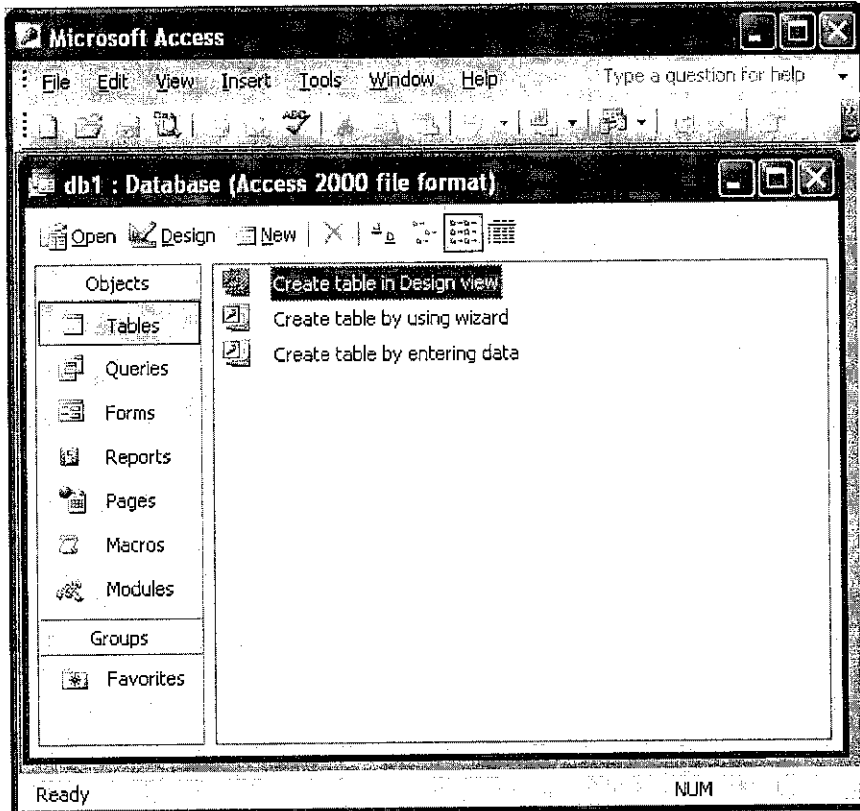


Figure 22 Microsoft Access.

Translation : Table		
No	Spoken	Translation
1	we	Kita
2	they	Mereka
3	food	Makanan
4	night	Malam
*	(AutoNumber)	

Figure 23 Table for database.

3.3.2 Build GUI.

Another feature that makes visual basic is a powerful GUI application is that it can create a form for user to operate the program. A GUI is a graphical, (rather than purely textual) user interface to a computer. The term came into existence because the first interactive user interfaces to computers were not graphical; they were text-and-keyboard oriented and usually consisted of commands that we had to remember and computer responses that were infamously brief.

The command interface of the DOS operating system is an example of the typical user-computer interface before GUIs arrived. An intermediate step in user interfaces between the command line interface and the GUI was the non-graphical menu-based interface, which allows interaction by using a mouse rather than by having to type in keyboard commands.

Figure presented below shows the main elements to have good graphical user interface for this project which are label, text box and command button. Basically label is to display information on how to operate the form. While text box is to allow the user to enter text such as user name or password. Last but not least is command button that basically to run the program or to enter to another form.

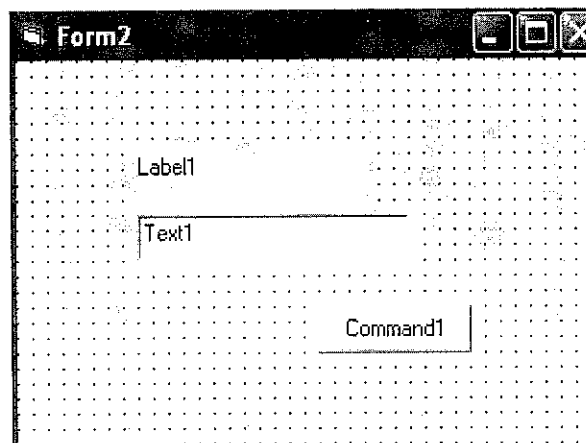


Figure 24 Create GUI.

The figure below shows the Visual Basic editor in view code mode and the red circle indicate run button to running the program written. The code mode is where to write the Basic code to program the event or action should be taken when command button is pushed or to set the properties for link to the communication port.

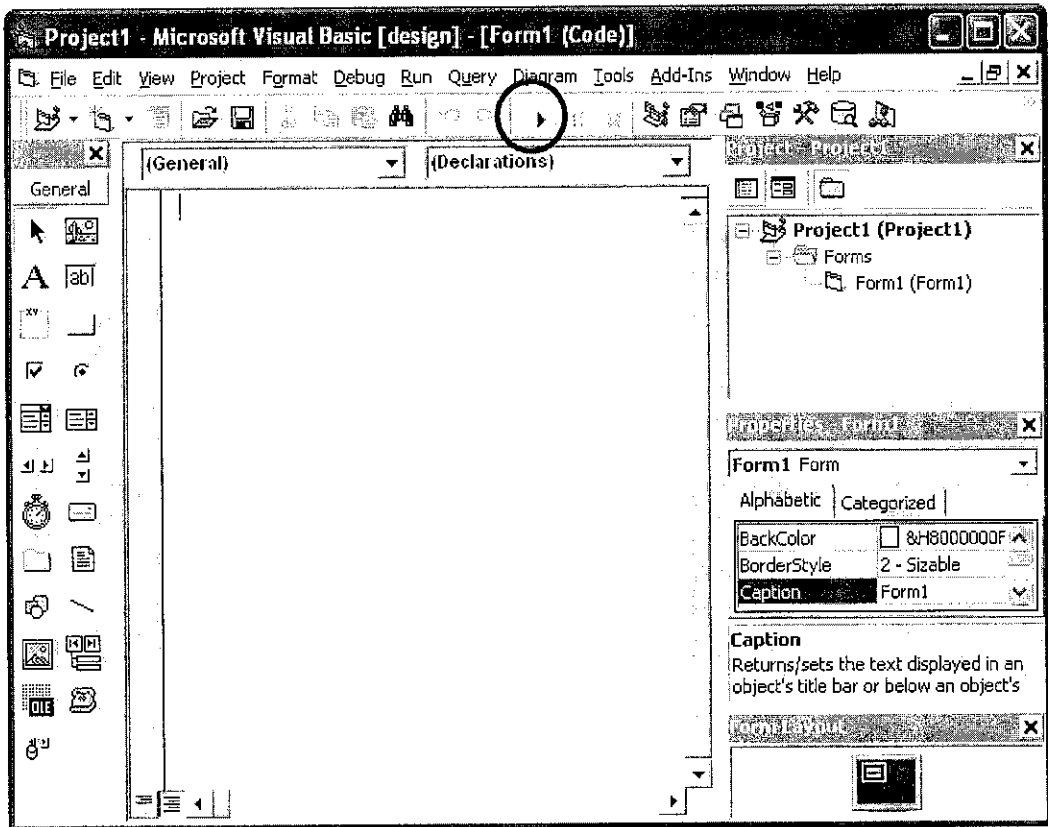


Figure 25 Visual Basic editor in view code mode.

3.3.3 Communication port.

A component known as the Microsoft Comm Control 6.0 is applied to allow the connection between Visual Basic and the serial port. This is shown in figure below to add the new component and where it will appear.

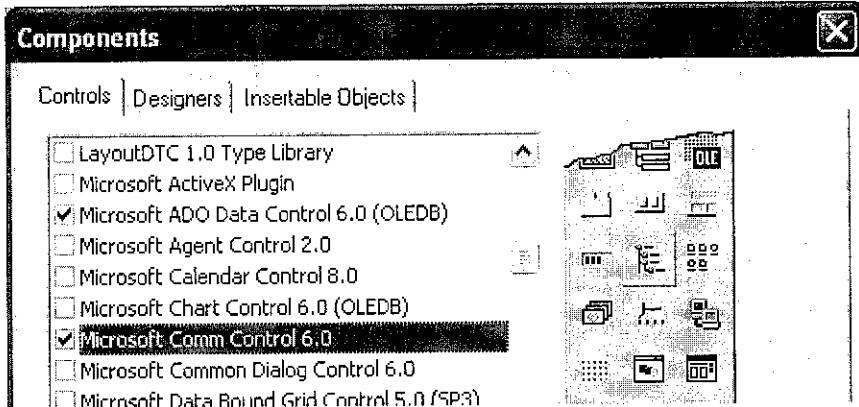


Figure 26 Options to add new component.

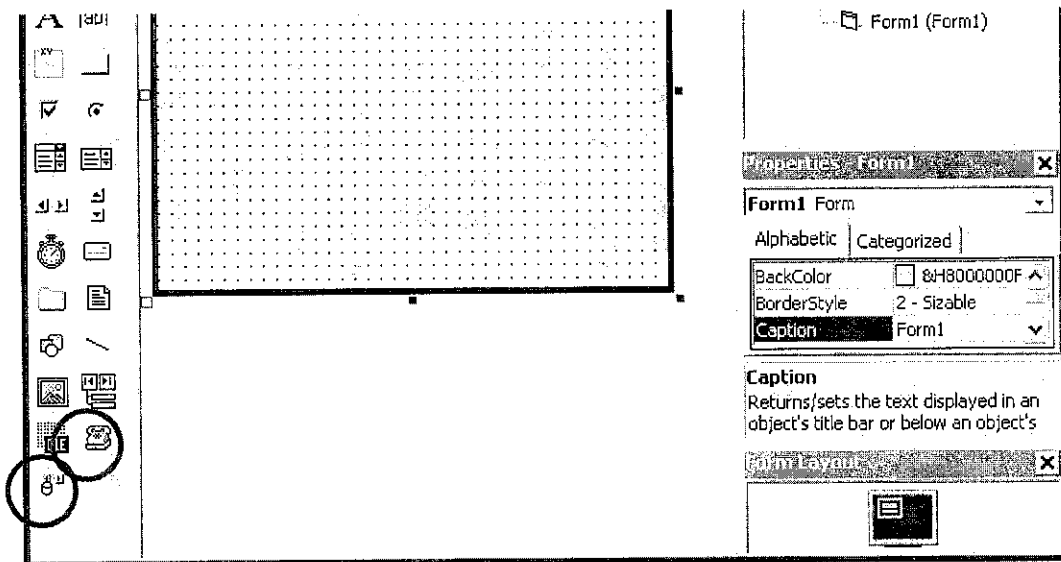


Figure 27 Visual Basic editor with added component.

To setup the connection from visual basic to serial port, the figure that looks like telephone is drag to the form to activate the connection. Another important thing is to set the properties of the communication port by typing the code as shown below in the code mode under form load.

```
Private Sub Form_Load()  
    'open port  
    MSCComm1.CommPort = 1  
    MSCComm1.Settings = "9600,n,8,1"  
    MSCComm1.PortOpen = True  
End Sub
```

3.3.4 Serial Communication Interface

When using the component Microsoft Comm Control 6.0, it needs the transmission medium which is serial communication interface. The serial port is chosen because it proves to be feasible.

As far as interfacing between the PIC16F84 and PC is concerned, initialization is the first consideration by modifying the settings on both ends to accommodate each other. These settings include:

- Baud rate: 9600
- Parity bit: None
- Data bits: 8
- Stop bits: 1

These initializations will coincide with that of the PIC16F84. The two pins from PIC16F84 that interacts with the serial port are:

- Transmit pin (B1)
- Receive pin (B2)

Old PC's used 25 pin connectors but only about 9 pins were actually used so today most connectors are only 9-pin. Each of the 9 pins usually connects to a wire. Besides the two wires used for transmitting and receiving data, another pin (wire) is signal ground. The voltage on any wire is measured with respect to this ground. Thus the minimum number of wires to use for 2-way transmission of data is 3. Despite this method, it has also been known to work with no signal ground wire but with degraded performance and sometimes with errors.

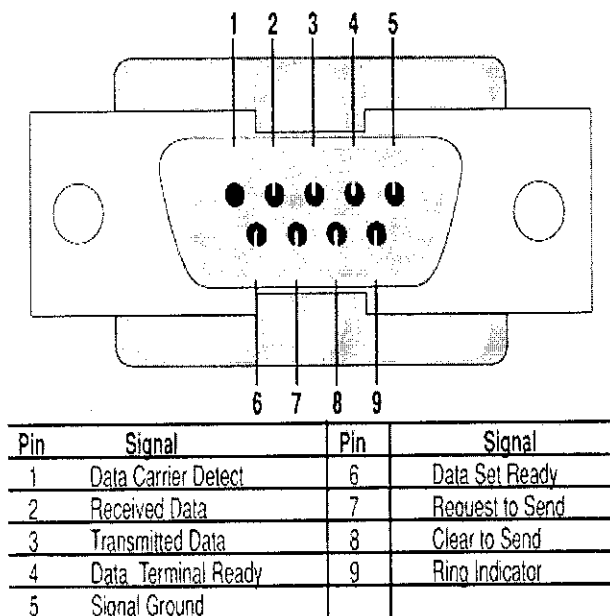


Figure 28 RS-232 Pin Out on a DB-9 Pin Used for Asynchronous Data

Almost all digital devices used nowadays require either TTL or CMOS logic levels. Therefore the first step to connecting a device to the RS-232 port is to transform the RS-232 levels back into 0 and 5 Volts. This is done by RS-232 level converters. Two common RS-232 level converters are the 1488 RS-232 driver and the 1489 RS-232 receiver. Each package contains 4 inverters of the one type, either drivers or receivers. The driver requires two supply rails, +7.5V to +15V and -7.5V to -15V. As expected, this may pose a problem in many instances where only a single supply of +5V is present. However the advantages of these IC's are they are cheap.

Besides the RS-232, another device is the MAX-232. It includes a charge pump, which generates +10V and -10V from a single 5V supply. This IC also includes two receivers and two transmitters in the same package. This is handy in many cases when only the Transmit and Receive data Lines are needed

3.4 Output interface

To make the project able to run with external program such as PIC microcontroller, the visual basic program needs to communicate with the circuit through serial port. Before that, a simple experiment needs to test out whether the communication port works properly with the PIC microcontroller.

The experiment consists of PIC Simulator IDE software, Warp13 software, and HyperTerminal application.

3.4.1 PIC Simulator IDE

PIC Simulator IDE is software that help user to program the PIC microcontroller. PIC Simulator IDE is powerful application that supplies PIC developers with user-friendly graphical development environment for Windows with integrated simulator (emulator), BASIC compiler, assembler, disassembler and debugger.

Default extension for basic source files is BAS. The compiler output is assembler source file (with ASM extension) that can be translated to binary code using integrated assembler. Smart editor marks all reserved keywords in different color that simplifies debugging process. BASIC compiler's assembler output has all necessary comment lines that make it very useful for educational purposes, also.

Figure below shows the windows of the PIC simulator IDE software.

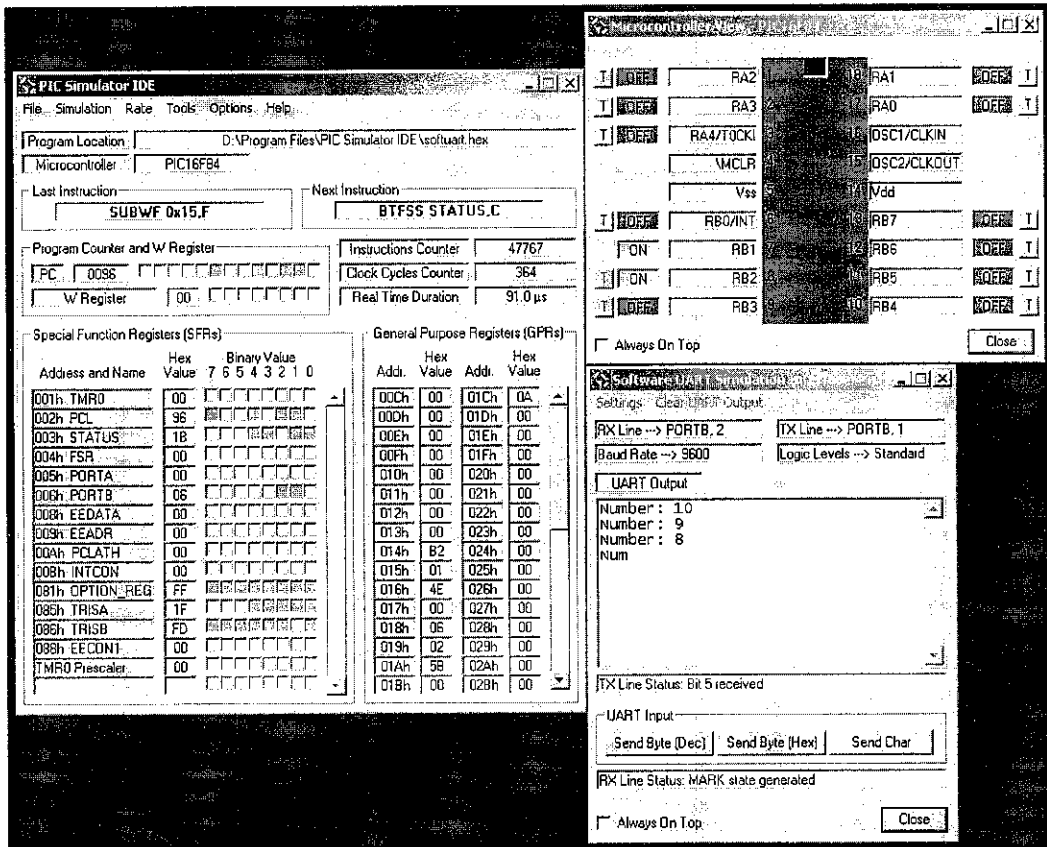


Figure 29 PIC Simulator IDE software.

3.4.2 Warp13 software

After make sure the program is running smoothly using the simulator, it needs to be transfer to the real PIC microcontroller. To do this, the program has to download in the PIC using WARP13 software which capable to do so. The requirement to download it is the file that is called '.hex' file. This file is automatically generated when the user compile the programming.

The WARP13 software window is shown in this figure below.

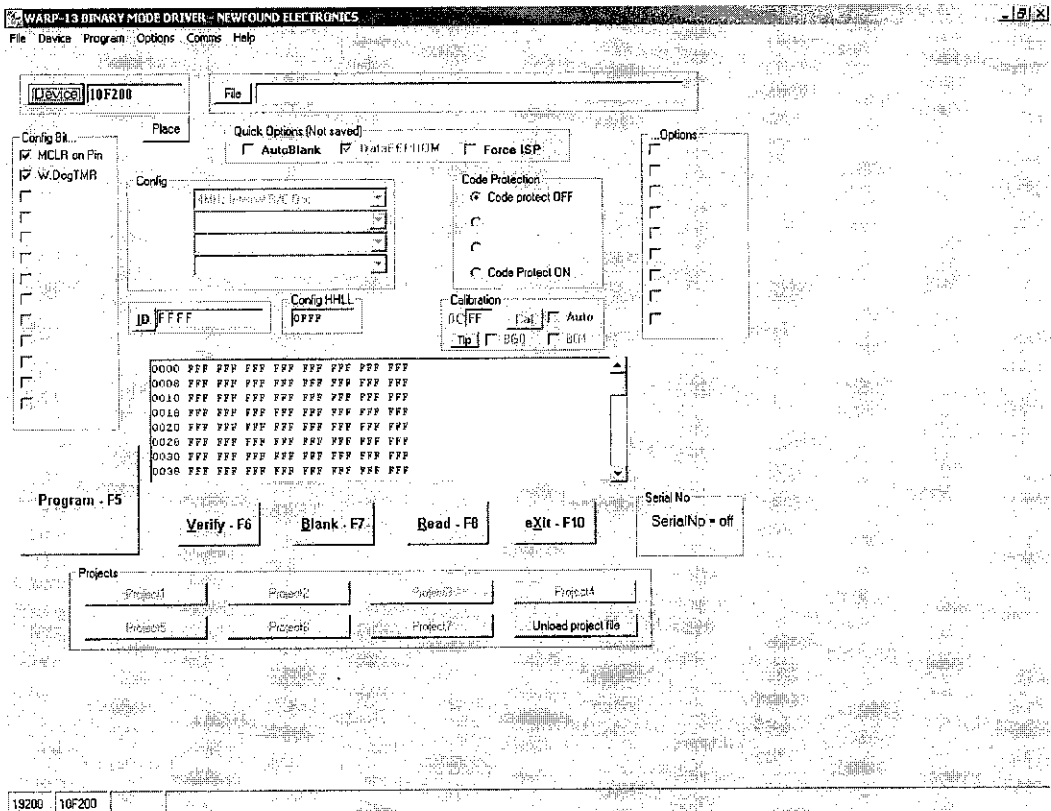


Figure 30 Warp13 PIC programmer.

3.4.3 HyperTerminal application

The last element in this output interface experiment is HyperTerminal application where usually automatically installed in the PC. Its location is shown in the figure below.

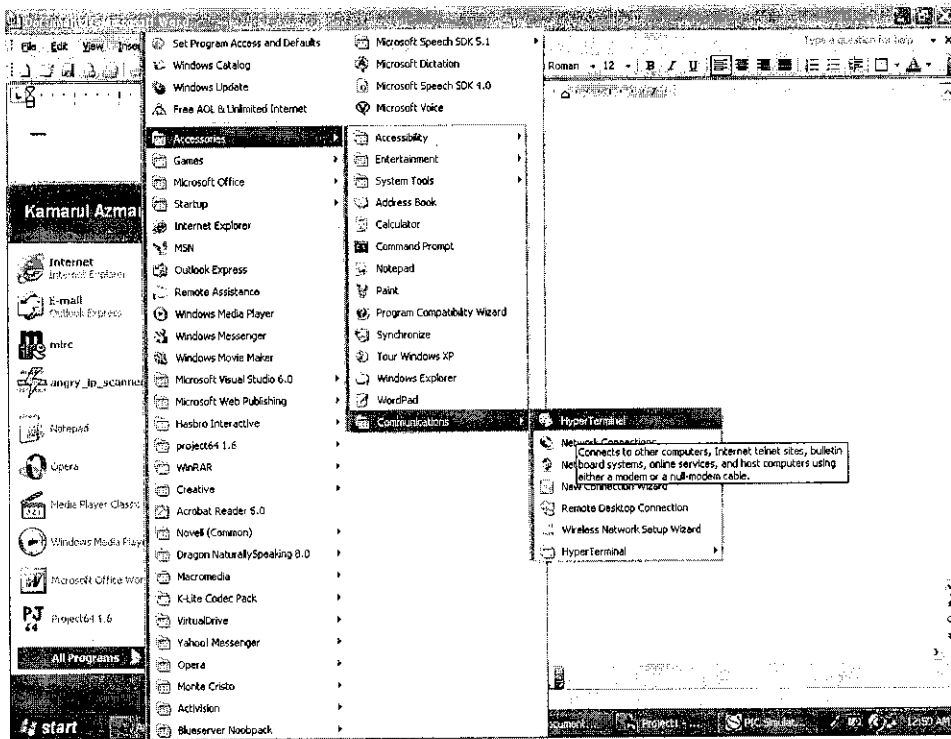


Figure 31 The location of HyperTerminal application.

HyperTerminal is a program that user can use to connect to other computers, Telnet sites, bulletin board systems (BBSs), online services, and host computers, using either modem or a null modem cable

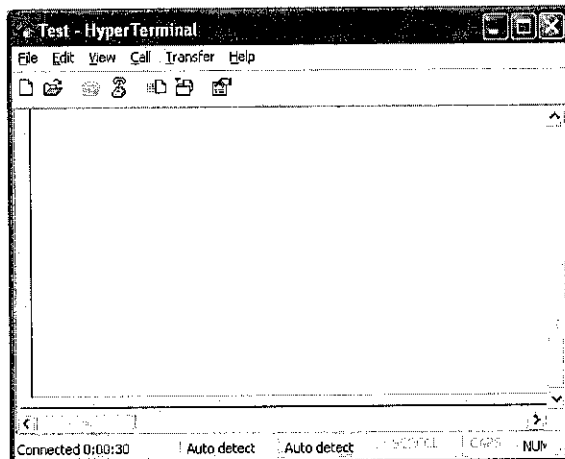


Figure 32 HyperTerminal window.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Results.

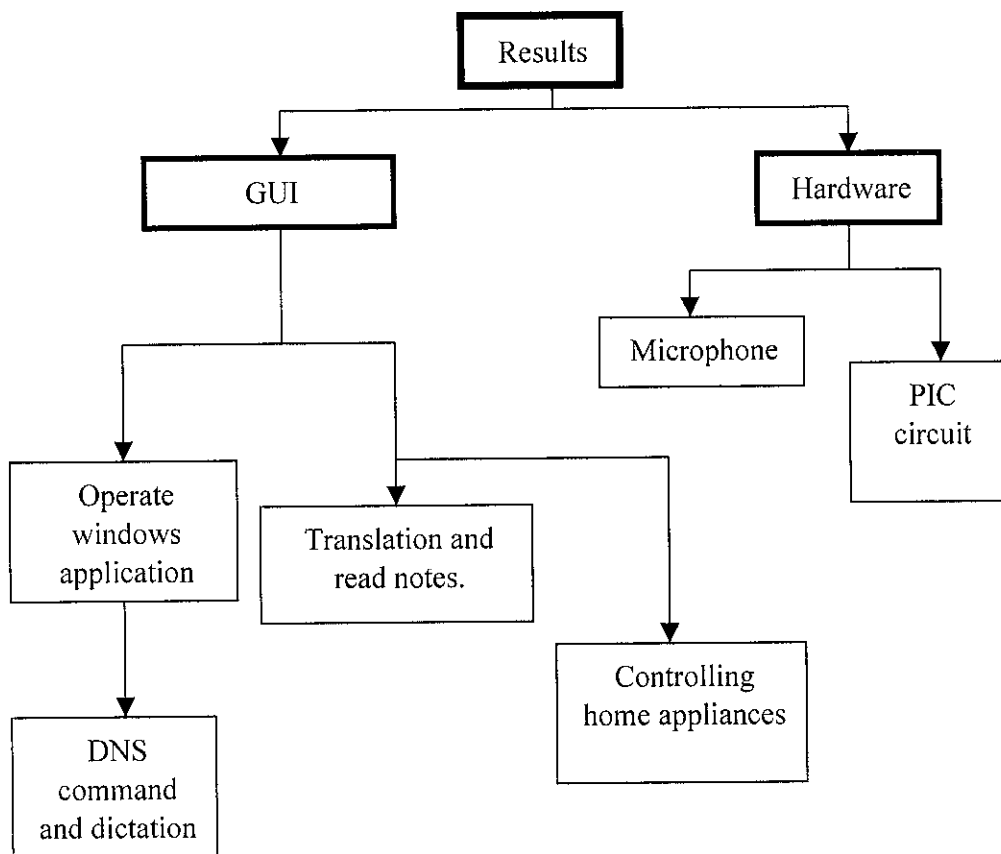


Figure 33 Results block diagram.

4.2 GUI

For Graphical User Interface part, the author had divided it into four windows which are Main Menu, Section 1, Section 2 and Section 3. Main Menu windows as shown in figure below consists title of the program and options to the other sections.

From the Main Menu window, it states all the three sections which are Controlling other application, Translation and Read notes and Home Automation. To enter the section of choice, user needs to click on the appropriate button or just simply say the button command e.g “Section 1”.

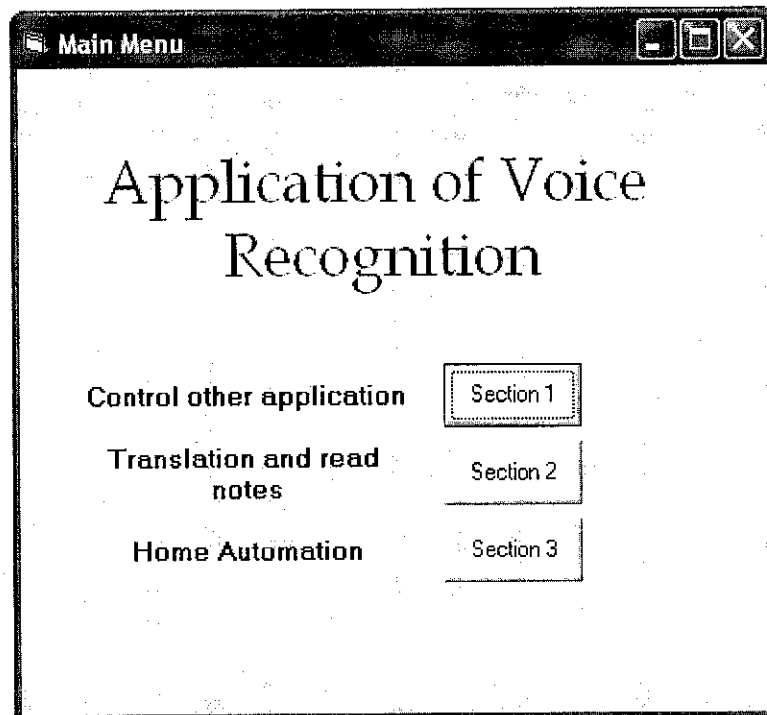


Figure 34 Main menu of the project.

4.2.1 Controlling other application

Once the Section 1 is chosen, the window as shown below is appearing. The window consists of simple instruction, a text box and main menu button. The text box here is acting as the trigger to open a new or different application. When user speaks to the microphone, the voice recognition will detect and transcript it to text and display in the text box provided.

As the example shown, to open the Notepad application user need to say "Start Notepad" and wait for the application to appear. Then the other command can be continue such as to write a document, open file and etc. To open another application, user can simply say "Start <application>" for the application of choice.

All types of application can be open unless the application is not to complex to spell out. At this part, the voice recognition software DNS itself play most of the command. As stated earlier, if only the SDK tools come free with the software then only the command can run from the visual basic.

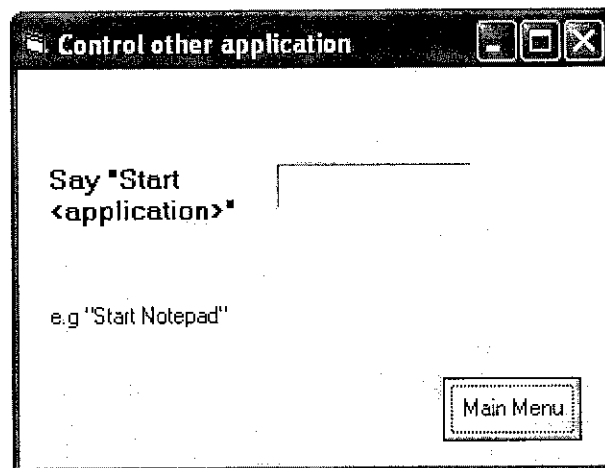


Figure 35 Section 1 of the project.

4.2.2 Translation and Read Notes

For Section 2 – Translation and read Notes, it requires three text box, and two command buttons. The first text box to recognize the spoken word by user while second text box is to view the translation word in Malay. As stated in the third text box, it lets user to paste the note to be read here. When the read button is click, the computer will read all the text written in the text box.

To achieve this, author had program several words only that can be translated by using Microsoft Access to create database. This is because to show the demonstration on how it can be done. This is perhaps to be a learning module to kids or even greater as tool of communication between two person with different language.

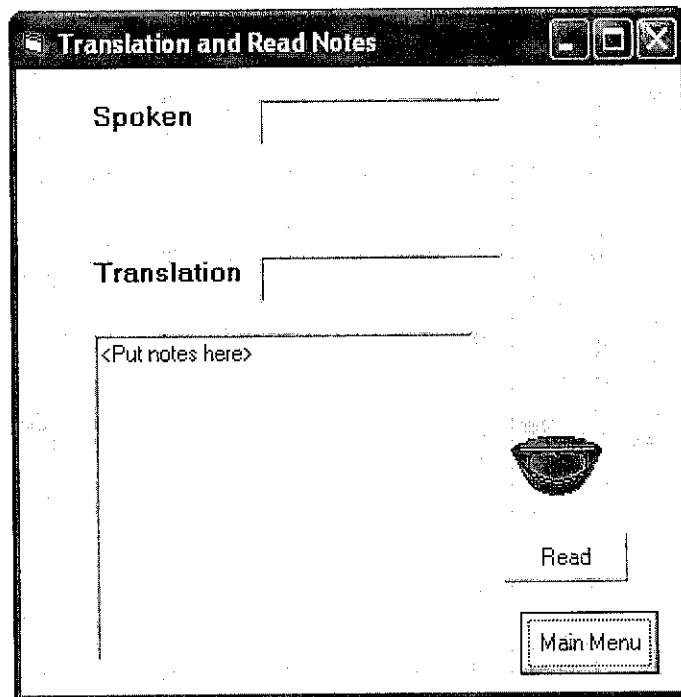


Figure 36 Section 2 of the project.

4.2.3 Home Automation

Last but not least is Home Automation section. This section is more complicated than the other two because it link with the serial communication port to control the PIC microcontroller. Home Automation window as shown below consists of recognition part, home status and button to return to main menu.

Basically this section provide user to control their home appliances by using speech command. It will acting like a remote switch to turn on and off the application. When user gives command, the program will compare the speech recognition with the command database. If the recognition is match it will trigger the output circuitry and execute the command given. If there is not match the program will ask user to try again the command.

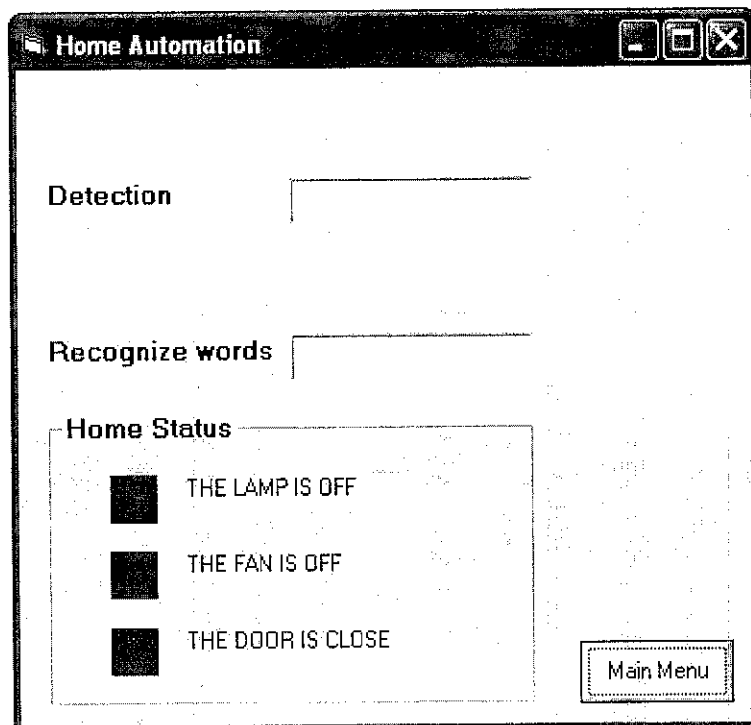


Figure 37 Section 3 of the project.

4.3 Choosing microphone

After recognize and try out a few types of microphone, the author had decided to choose the wireless hand held microphone. The microphone with better performance can influence the accuracy of speech recognition for this project beside the program itself. There are reasons why this type of microphone had been chosen.

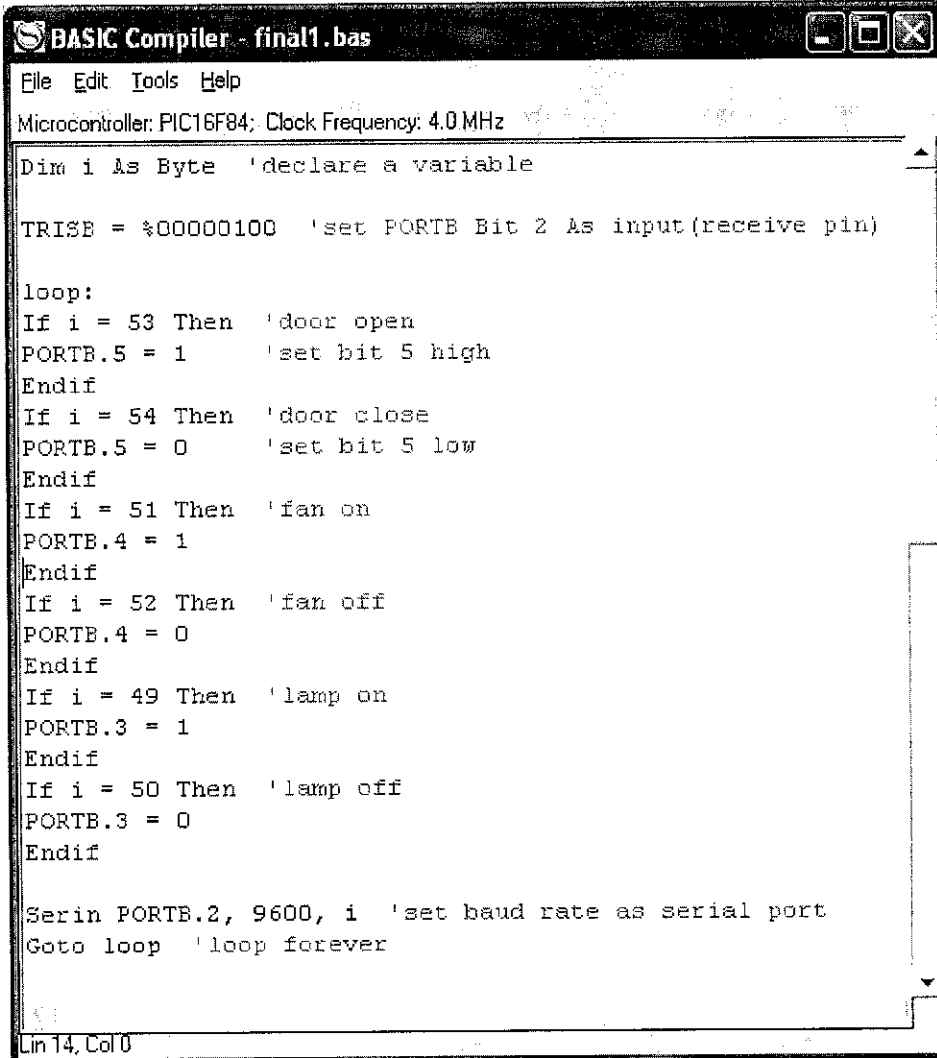
Firstly, the price of the microphone is within the budget provided by the Final Year Project committee. The budget is around RM 250 while the price for the microphone is RM180. Although it is quite expensive, but the Signal-to-Noise (S/N) ratio is high which is larger than 100dB. If compared to other less expensive microphone, this wireless microphone is more robust.

Due to the high S/N ratio, the voice recognition is more accurate compared to other microphone. This wireless microphone is included with receiver which able the service areas around 50 meters in the best condition. The best condition indicate that the quite and noise free environment.

Moreover, wireless microphone is free with problem tangling or caught up with long wire. Therefore this makes it easy to handle. The details of the overall specification can be found in appendix.

4.4 Programming PIC16F84

As stated earlier, to program the microcontroller PIC the author chooses to use PIC Simulator IDE. Figure below shows the BASIC compiler where the programming is been done. ASCII code is chosen to be the medium of data transmission.



```
BASIC Compiler - final1.bas
File Edit Tools Help
Microcontroller: PIC16F84; Clock Frequency: 4.0 MHz
Dim i As Byte 'declare a variable
TRISB = %00000100 'set PORTB Bit 2 As input(receive pin)
loop:
If i = 53 Then 'door open
PORTB.5 = 1 'set bit 5 high
Endif
If i = 54 Then 'door close
PORTB.5 = 0 'set bit 5 low
Endif
If i = 51 Then 'fan on
PORTB.4 = 1
Endif
If i = 52 Then 'fan off
PORTB.4 = 0
Endif
If i = 49 Then 'lamp on
PORTB.3 = 1
Endif
If i = 50 Then 'lamp off
PORTB.3 = 0
Endif
Serin PORTB.2, 9600, i 'set baud rate as serial port
Goto loop 'loop forever
Lin 14, Col 0
```

Figure 38 BASIC Compiler window.

The details on circuitry are attached to the appendix.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 Challenges and problems

As conclusion, the project is success although there are some problems that interfere. The first problem in this project is to find the best speech recognition software in the market. There are lots of the software where it needs to be familiarize before decided to apply in the project.

Another one is challenge to keep the accuracy of the speech recognition. There are few factors that can reduce accuracy to recognize the spoken words. First is the type of microphone chosen need to have high S/N ratio. This will help to eliminate the noise better. Second is the environment where to apply this system the place need to be quite and peace.

Last but not least, is user condition and emotion. The condition of the user needs to be normal as possible so that the words spoken are clear and easy to recognize. When user had a cough or fevers their tone and pitch of voice is changes and will affect the system to recognize the user.

5.2 Recommendation

To achieve better performance of the system, it is recommended to use high quality input hardware for microphone and sound card. But due to cost constraint, it cannot achieve for this project. For the system to be global, means everyone can use it, the system should allow to recognize different accent and voice pitch of person. To achieve this new guideline for the standardize and simple command need to be create.

REFERENCES

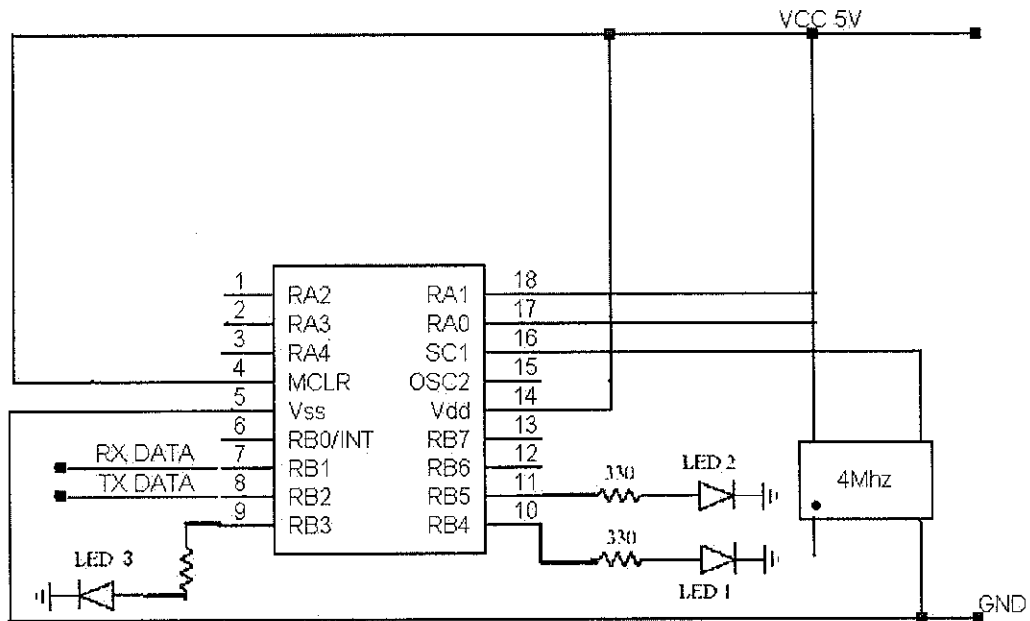
1. History of Speech & Voice Recognition and Transcription Software, http://www.dragon-medical-transcription.com/history_speech_recognition.html
2. A Timeline & History of Voice Recognition Software, http://www.dragon-medical-transcription.com/history_speech_recognition_timeline.html
3. Article of Speech Recognition, Gunish Rai Chawla, <http://www.developerdotstar.com/community/user/49/>
4. How Microphone Works, <http://electronics.howstuffworks.com/question309.htm>
5. How Sound Card Works, <http://computer.howstuffworks.com/sound-card2.htm>
6. Programming the PIC Controller, <http://www.microchip.com/>
7. PIC Tutorial, <http://www.mstracey.btinternet.co.uk/index.htm>
8. PIC Simulator IDE, <http://www.oshonsoft.com/index.html>
9. PIC16F84 Fundamental, <http://www.boondog.com//tutorials/pic16F84/pic16f84.html#Overview>
10. Microchip PIC Project, <http://www.users.bigpond.com/armagh/page2.html>
11. PIC Programmer, <http://www.bobblick.com/index.html>
12. Error in Opening Comm port in VB, http://www.control.com/index_html
13. Visual Basic Tutorial, <http://www.vbtutor.net/vbtutor.html>
14. General PIC tutorial, <http://www.sparkfun.com/tutorial/coding/1-intro.htm>
15. Home Automated Living, <http://www.automatedliving.com/>
16. Home Automation System, <http://www.chiark.greenend.org.uk/~theom/electronics/has/index.html>
17. Voice Recognition Project, Wan Zahara's Report.
18. Home Surveillance Project, Wan Nubli's Report
19. Microsoft Speech SDK Getting Help.

APPENDICES

<u>Appendix A PIC16F84 Connection</u>	59
<u>Appendix B MAX232 Connection</u>	60
<u>Appendix C Table of Ascii code</u>	61
<u>Appendix D Table of ascii code</u>	62
<u>Appendix E User Manual to Operate the system</u>	63
<u>Appendix F Visual Basic Source Code</u>	64
<u>Appendix G Coding for PIC</u>	65
<u>Appendix H PIC Simulator Manual</u>	69

APPENDIX A

PIC16F84 CONNECTION



Legend:

Vss = GND

Vdd = +5V

SC1 = External clock

RB1 = Transmit data to MAX232

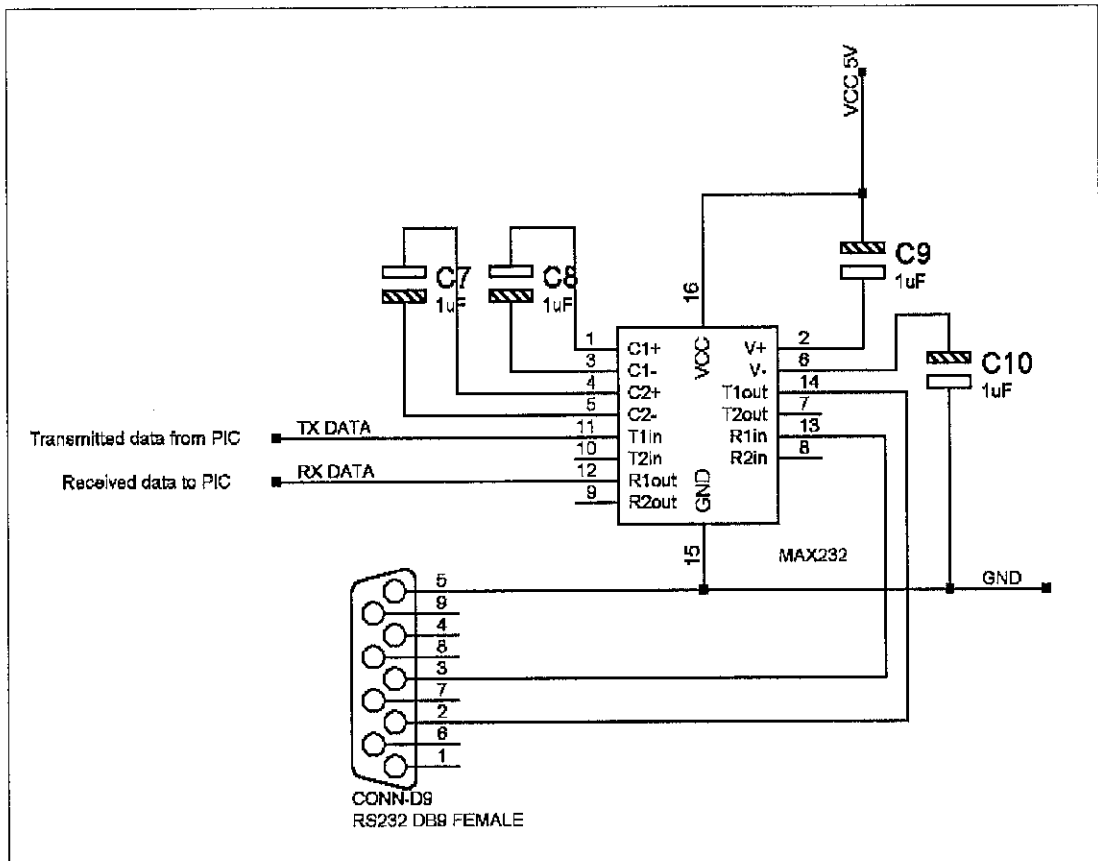
RB2 = Receive data from MAX232

RB3 = RB4 = RB5 = Output pin

Figure above shows the basic connection of microcontroller PIC16F84 which used in the project. The PIC is power up with +5V power supply and the timing is set by the crystal clock 4Mhz. The microcontroller will receive data from the PC through the MAX232 chip. Then the program inside the PIC will read the received data and determine which output pin should be high or low.

APPENDIX B

MAX232 CONNECTION



Legend:

Vcc = +5V

GND = 0V

T1in = Received transmitted data from PIC

R1out = Transmit data to PIC

T1out = Transmit data to communication serial port

R1in = Received data from communication serial port

Figure above shows the basic connection of MAX232 between PIC16F84 and communication serial port. The chip is power up with +5V power supply. MAX232 is behave to translate the output data (analog) from PC which used ASCII as data transmit format (digital) to microcontroller PIC.

APPENDIX C

TABLE OF ASCII CODE

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	 	Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	\$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	((72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051))	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[[123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174	|	
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135]]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177		DEL

Source: www.asciitable.com

APPENDIX D

TABLE OF ASCII CODE

128	Ç	144	É	160	á	176	☐	193	±	209	≡	225	ß	241	±
129	ü	145	æ	161	í	177	☐	194	⌈	210	≡	226	Γ	242	≥
130	é	146	Æ	162	ó	178	☐	195	⌋	211	≡	227	π	243	≤
131	â	147	ô	163	ú	179		196	–	212	≡	228	Σ	244	∫
132	ã	148	ö	164	ñ	180	†	197	+	213	≡	229	σ	245	∫
133	ä	149	ò	165	Ñ	181	‡	198	‡	214	≡	230	μ	246	+
134	å	150	û	166	ª	182	‡	199	‡	215	‡	231	τ	247	±
135	ç	151	ù	167	º	183	≡	200	≡	216	‡	232	Φ	248	°
136	ê	152	_	168	¿	184	≡	201	≡	217	∫	233	⊙	249	·
137	ë	153	Ö	169	_	185	≡	202	≡	218	∫	234	Ω	250	·
138	è	154	Û	170	¬	186	≡	203	≡	219	■	235	δ	251	√
139	í	156	ê	171	½	187	≡	204	‡	220	■	236	∞	252	_
140	î	157	ÿ	172	¾	188	≡	205	=	221	■	237	φ	253	²
141	ï	158	_	173	¡	189	≡	206	‡	222	■	238	ε	254	■
142	Ä	159	ƒ	174	«	190	∫	207	±	223	■	239	∩	255	
143	Å	192	ℒ	175	»	191	∫	208	≡	224	∞	240	≡		

Source: www.asciitable.com

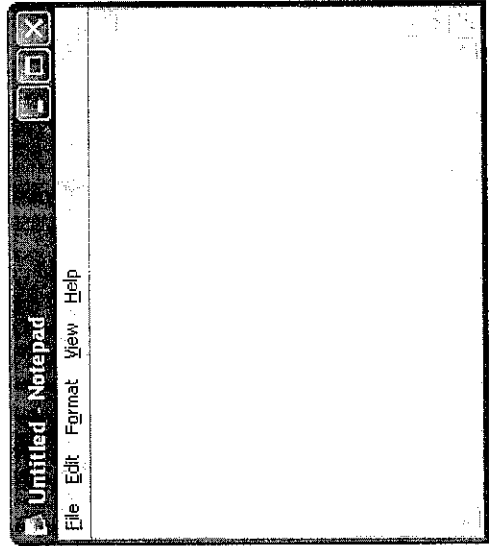
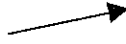
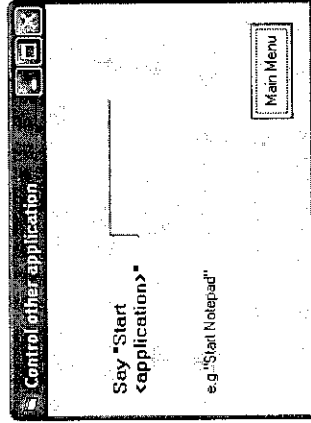
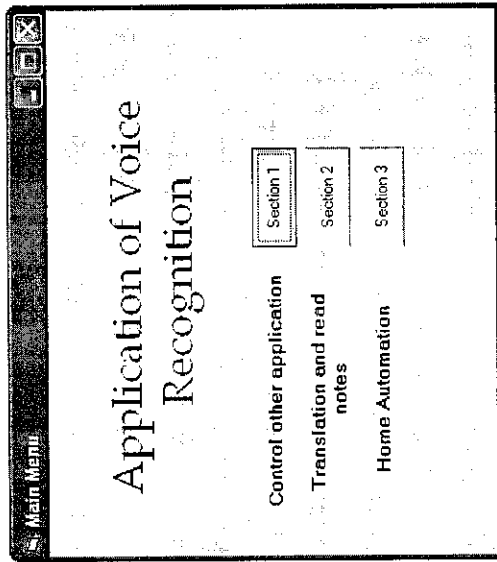
APPENDIX E
USER MANUAL TO OPERATE THE SYSTEM

SYSTEM OPERATION (USER MANUAL)

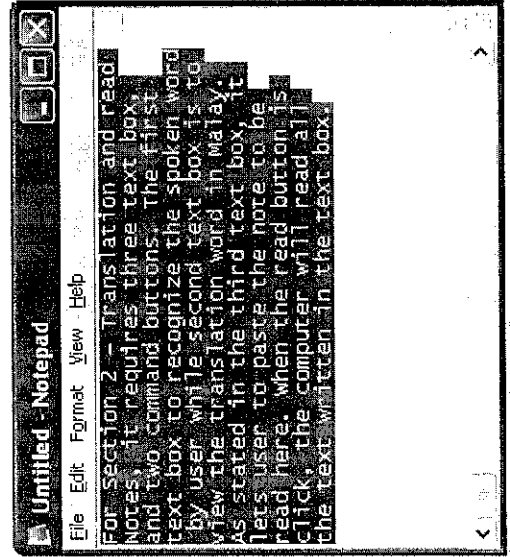
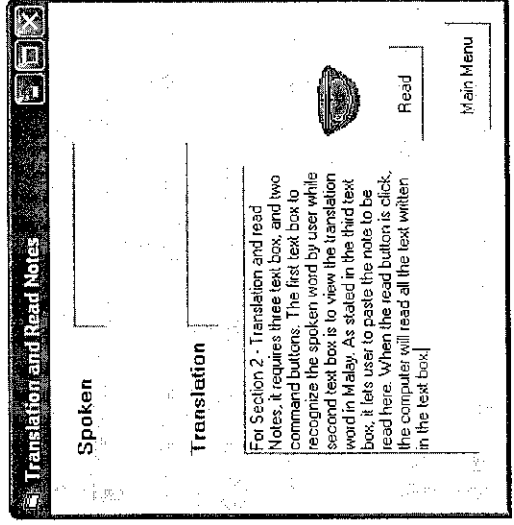
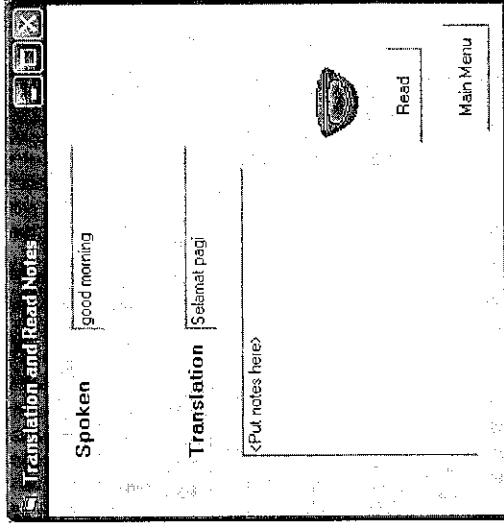
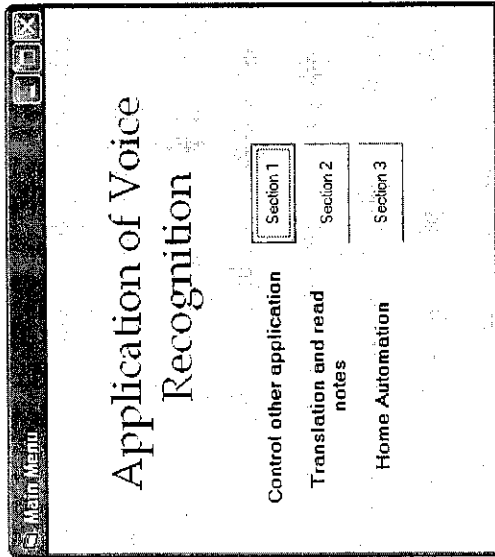
PROCESS/PROCEDURE	DRAGON NATURALLY SPEAKING	VISUAL BASIC	PIC CONTROLLER
Installation software	<ol style="list-style-type: none"> 1. Insert CD into Cd-ROM 2. Follow instruction of the installation wizard. 3. Train the Dragon NaturallySpeaking by following the tutorial. 	<ol style="list-style-type: none"> 1. Copy file "Voice Recognition.exe" to desktop. 	
Installation hardware	<ol style="list-style-type: none"> 1. Make sure that microphone is properly connected. 2. Switch on the microphone. 		<ol style="list-style-type: none"> 1. Connect the 9-pin female connector from circuit to the PC. 2. Turn on the circuit.
Turn on software application	<ol style="list-style-type: none"> 1. Start the Dragon NaturallySpeaking application. 2. Double click the icon DNS.exe on the desktop. 	<ol style="list-style-type: none"> 1. Start the voice recognition application. 2. Double click the icon Voice Recognition.exe on the desktop. 	
Controlling other window applications.		<ol style="list-style-type: none"> 1. Click on the Section 1 button. 2. Start desired application by saying "Start <application>". 	

		<ol style="list-style-type: none"> e.g: "Start Notepad". Refer to figure in App. E1. 	
Using translation and read notes.		<ol style="list-style-type: none"> Click on the Section 2 button. Start saying words that need translation. Copy text that needs to be read and paste in the box provided. Click the read button and let computer read the text. Refer to figure in App. E2. 	
Using home automation system.		<ol style="list-style-type: none"> Click on the Section 3 button. Say the appropriate command to operate the circuit. Check the status at home status box. Refer to figure in App. E3. 	<ol style="list-style-type: none"> Check the status of LED, buzzer and fan. Refer to figure in App. E3.
Close the system	<ol style="list-style-type: none"> Switch off the microphone and close the software. 	<ol style="list-style-type: none"> Click on the close button. 	<ol style="list-style-type: none"> Turn off circuit.

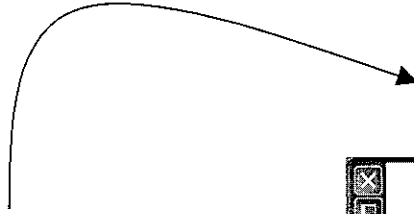
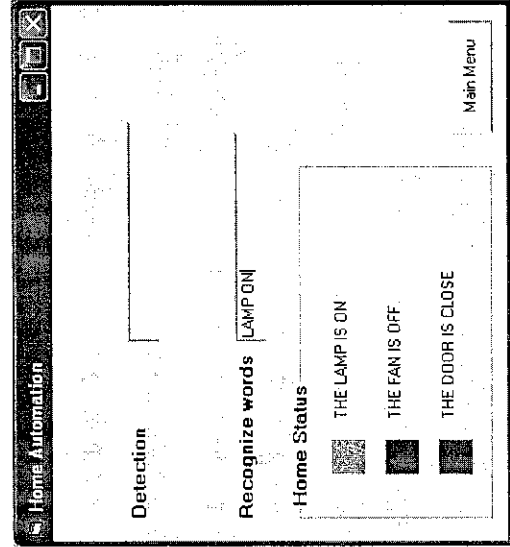
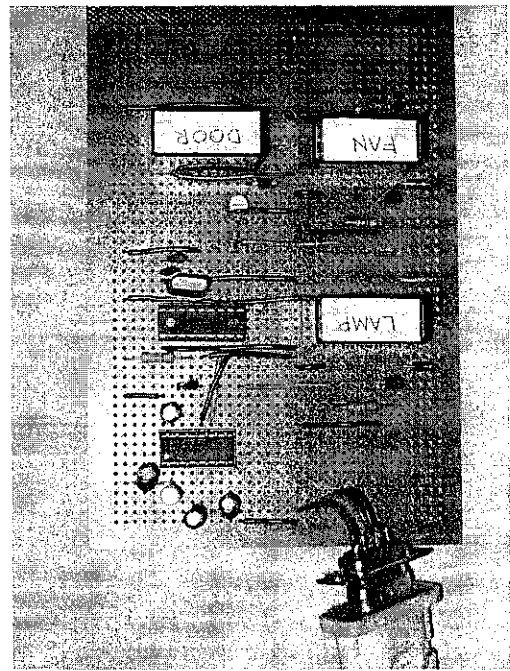
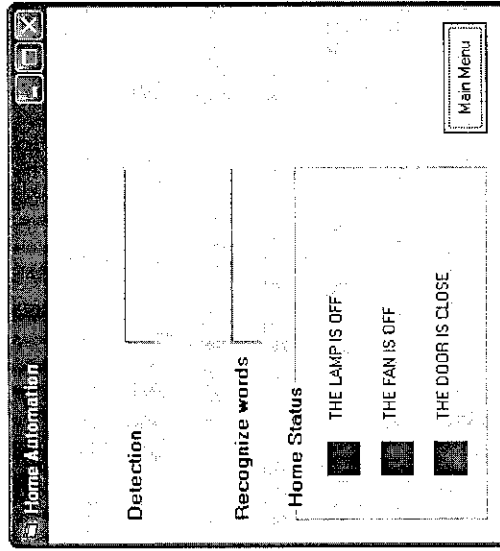
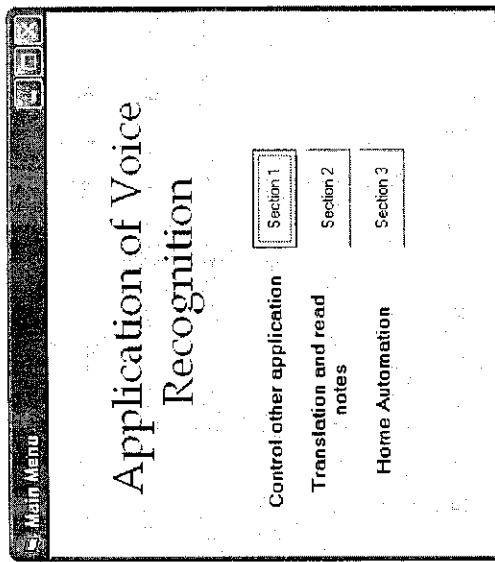
App. E1



APP. E2



App. E3



APPENDIX F
VISUAL BASIC SOURCE CODE

Main Menu Window Source Code

```
Private Sub Command1_Click()  
frmMain.Hide  
frmApp.Show  
End Sub  
  
Private Sub Command2_Click()  
frmMain.Hide  
frmTrans.Show  
End Sub  
  
Private Sub Command3_Click()  
frmMain.Hide  
frmHome.Show  
End Sub
```

Section 1 Window Source Code

```
Private Sub Command1_Click()  
frmApp.Hide  
frmMain.Show  
End Sub
```

Section 2 Window Source Code

```
Private Sub Command1_Click()  
frmTrans.Hide  
frmMain.Show  
  
End Sub  
  
Private Sub Command2_Click()  
If Command2.Caption = "Read" Then  
tts.Speak Text1.Text  
Command2.Caption = "Stop"  
Else  
If Command2.Caption = "Stop" Then  
tts.StopSpeaking  
Command2.Caption = "Read"  
End If  
End If  
  
End Sub  
  
Private Sub txtSpk_Change()  
Adodc1.Recordset.MoveFirst  
Do Until Adodc1.Recordset.EOF  
If Adodc1.Recordset.Fields("Spoken").Value = txtSpk.Text Then  
txtTrans.DataField = "Translation"  
  
Exit Sub  
Else  
Adodc1.Recordset.MoveNext
```

```

End If
Loop
txtSpk.Text = ""
txtTrans.Text = "<Unrecognized word>"
End Sub

```

Section 3 Window Source Code

```

Private Sub Command1_Click()
frmHome.Hide
frmMain.Show
End Sub
Private Sub Form_Load()
'open port
MSComm1.CommPort = 1
MSComm1.Settings = "9600,n,8,1"
MSComm1.PortOpen = True
'initials condition
lblLamp.BackColor = vbRed
lblLampStat.Caption = "THE LAMP IS OFF"
lblFan.BackColor = vbRed
lblFanStat.Caption = "THE FAN IS OFF"

```

```

lblDoor.BackColor = vbRed
lblDoorStat.Caption = "THE DOOR IS CLOSE"
End Sub
Private Sub lblDoor_Change()
If lblDoor.BackColor = vbRed Then
lblDoorStat.Caption = "THE DOOR IS CLOSE"
Else
lblDoorStat.Caption = "THE DOOR IS OPEN"
End If
End Sub
End Sub
Private Sub lblFan_Change()
If lblFan.BackColor = vbRed Then
lblFanStat.Caption = "THE FAN IS OFF"
Else
lblFanStat.Caption = "THE FAN IS ON"
End If
End Sub
End Sub
Private Sub lblLamp_Change()
If lblLamp.BackColor = vbRed Then
lblLampStat.Caption = "THE LAMP IS OFF"
Else
lblLampStat.Caption = "THE LAMP IS ON"
End If
End Sub
Private Sub txtDict_Change()
Dim txt1 As String

```

```

txt1 = txtDict.Text

adoVoice.Recordset.MoveFirst
Do Until adoVoice.Recordset.EOF
If adoVoice.Recordset.Fields("LampOn").Value = txt1 Then
Text2.Text = "LAMP ON"
lblLampStat.Caption = "LAMP ON"
txtRec.Text = "Lamp On"
MSComm1.Output = Chr(49)
lblLamp.BackColor = vbGreen
Exit Sub
Else
adoVoice.Recordset.MoveNext
End If
Loop

adoVoice.Recordset.MoveFirst
Do Until adoVoice.Recordset.EOF
If adoVoice.Recordset.Fields("LampOff").Value = txt1 Then
Text2.Text = "LAMP OFF"
lblLampStat.Caption = "LAMP OFF"
txtRec.Text = "Lamp Off"
MSComm1.Output = Chr(50)
lblLamp.BackColor = vbRed
Exit Sub
Else

```

```

adoVoice.Recordset.MoveNext
End If
Loop

adoVoice.Recordset.MoveFirst
Do Until adoVoice.Recordset.EOF
If adoVoice.Recordset.Fields("FanOn").Value = txt1 Then
Text2.Text = "FAN ON"
lblFanStat.Caption = "FAN ON"
txtRec.Text = "Fan On"
MSComm1.Output = Chr(51)
lblFan.BackColor = vbGreen
Exit Sub
Else
adoVoice.Recordset.MoveNext
End If
Loop

adoVoice.Recordset.MoveFirst
Do Until adoVoice.Recordset.EOF
If adoVoice.Recordset.Fields("FanOff").Value = txt1 Then
Text2.Text = "FAN OFF"
lblFanStat.Caption = "FAN OFF"
txtRec.Text = "Fan Off"
MSComm1.Output = Chr(52)
lblFan.BackColor = vbRed

```



```
Exit Sub

Else

adoVoice.Recordset.MoveNext

End If

Loop

adoVoice.Recordset.MoveFirst
Do Until adoVoice.Recordset.EOF
If adoVoice.Recordset.Fields("Open").Value = txt1 Then
Text2.Text = "DOOR OPEN"
lblDoorStat.Caption = "DOOR OPEN"
txtRec.Text = "Open Door"
MSComm1.Output = Chr(53)
lblDoor.BackColor = vbGreen
Exit Sub
Else

adoVoice.Recordset.MoveNext

End If

Loop

adoVoice.Recordset.MoveFirst
Do Until adoVoice.Recordset.EOF
If adoVoice.Recordset.Fields("Close").Value = txt1 Then
```

```
Text2.Text = "DOOR CLOSE"
lblDoorStat.Caption = "DOOR CLOSE"
txtRec.Text = "Close Door"
MSComm1.Output = Chr(54)
lblDoor.BackColor = vbRed
Exit Sub

Else

adoVoice.Recordset.MoveNext

End If

Loop

Text2.Text = ""
txtDict.Text = " "
txtRec.Text = "<Unrecognized word>"

End Sub
```

APPENDIX G

CODING FOR PIC

1. “.bas file”

```
PORTB.5 = 0          'set bit 5 low
Endif
If i = 51 Then 'fan on
PORTB.4 = 1
Endif
If i = 52 Then 'fan off
PORTB.4 = 0
Endif
If i = 49 Then 'lamp on
PORTB.3 = 1
Endif
If i = 50 Then 'lamp off
PORTB.3 = 0
Endif

Serin PORTB.2, 9600, i 'set baud rate as serial port
Goto loop 'loop forever
```

2. “.asm file”

```
; Begin
R0L EQU 0xC
R0H EQU 0xD
R1L EQU 0xE
R1H EQU 0xF
R2L EQU 0x10
R2H EQU 0x11
R3L EQU 0x12
R3H EQU 0x13
R4L EQU 0x14
R4H EQU 0x15
R5L EQU 0x16
R5H EQU 0x17
SO_PORT EQU 0x18
SO_BIT EQU 0x19
SO_INTL EQU 0x1A
SO_INTH EQU 0x1B
ORG 0x0000
BCF PCLATH,3
BCF PCLATH,4
GOTO L0002
ORG 0x0004
RETFIE

L0002:
; 1: Dim i As Byte 'declare a variable
;   The address of 'i' is 0x1C
;   i EQU 0x1C
; 2:
; 3: TRISB = %00000100 'set PORTB Bit 2 As input(receive pin)
BSF STATUS,RP0
MOVLW 0x04
MOVWF 0x06
BCF STATUS,RP0
```

```

; 4:
; 5: loop:
L0001:
; 6: If i = 53 Then 'door open
      MOVF 0x1C,W
      SUBLW 0x35
      BTFSS STATUS,Z
      GOTO L0003
; 7: PORTB.5 = 1 'set bit 5 high
      BSF 0x06,5
; 8: Endif
L0003: MOVLW 0x1F
      ANDWF STATUS,F
; 9: If i = 54 Then 'door close
      MOVF 0x1C,W
      SUBLW 0x36
      BTFSS STATUS,Z
      GOTO L0004
; 10: PORTB.5 = 0 'set bit 5 low
      BCF 0x06,5
; 11: Endif
L0004: MOVLW 0x1F
      ANDWF STATUS,F
; 12: If i = 51 Then 'fan on
      MOVF 0x1C,W
      SUBLW 0x33
      BTFSS STATUS,Z
      GOTO L0005
; 13: PORTB.4 = 1
      BSF 0x06,4
; 14: Endif
L0005: MOVLW 0x1F
      ANDWF STATUS,F
; 15: If i = 52 Then 'fan off
      MOVF 0x1C,W
      SUBLW 0x34
      BTFSS STATUS,Z
      GOTO L0006
; 16: PORTB.4 = 0
      BCF 0x06,4
; 17: Endif
L0006: MOVLW 0x1F
      ANDWF STATUS,F
; 18: If i = 49 Then 'lamp on
      MOVF 0x1C,W
      SUBLW 0x31
      BTFSS STATUS,Z
      GOTO L0007
; 19: PORTB.3 = 1
      BSF 0x06,3
; 20: Endif
L0007: MOVLW 0x1F
      ANDWF STATUS,F
; 21: If i = 50 Then 'lamp off
      MOVF 0x1C,W
      SUBLW 0x32
      BTFSS STATUS,Z
      GOTO L0008
; 22: PORTB.3 = 0
      BCF 0x06,3
; 23: Endif

```

```

L0008: MOVLW 0x1F
      ANDWF STATUS,F
; 24:
; 25: Serin PORTB.2, 9600, i 'set baud rate as serial port
      BSF STATUS,RP0
      BSF TRISB,2
      BCF STATUS,RP0
      MOVLW 0x06
      MOVWF SO_PORT
      MOVLW 0x04
      MOVWF SO_BIT
      MOVLW 0x5B
      MOVWF SO_INTL
      MOVLW 0x00
      MOVWF SO_INTH
      CALL SI01
      MOVWF 0x1C
; 26: Goto loop 'loop forever
      GOTO L0001
; 27:
; 28:
; 29:
; End of program
L0009: GOTO L0009
; Waitus Routine - Word Argument
Y001: MOVLW 0x10
      SUBWF R4L,F
      CLRW
      BTFSS STATUS,C
      ADDLW 0x01
      SUBWF R4H,F
      BTFSS STATUS,C
      RETURN
      GOTO Y002
Y002: MOVLW 0x0A
      SUBWF R4L,F
      CLRW
      BTFSS STATUS,C
      ADDLW 0x01
      SUBWF R4H,F
      BTFSS STATUS,C
      RETURN
      GOTO Y002
; Serin Routine
SI01: CALL SI03
      BTFSC STATUS,C
      GOTO SI01
      MOVF SO_INTL,W
      MOVWF R4L
      MOVF SO_INTH,W
      MOVWF R4H
      RRF R4H,F
      RRF R4L,F
      BCF STATUS,C
      RRF R4H,F
      RRF R4L,F
      CALL Y001
      MOVLW 0x08
      MOVWF R5H
SI02: MOVF SO_INTL,W
      MOVWF R4L

```

```
MOVF SO_INTH,W
MOVWF R4H
CALL Y001
CALL SI03
RRF R5L,F
DECFSZ R5H,F
GOTO SI02
MOVF SO_INTL,W
MOVWF R4L
MOVF SO_INTH,W
MOVWF R4H
CALL Y001
MOVF R5L,W
RETURN
SI03: MOVF SO_PORT,W
MOVWF FSR
MOVF SO_BIT,W
ANDWF INDF,W
ADDLW 0xFF
RETURN
; End of listing
END
```

APPENDIX H

PIC SIMULATOR MANUAL

Default extension for basic source files is BAS. The compiler output is assembler source file (with ASM extension) that can be translated to binary code using integrated assembler. Smart editor marks all reserved keywords in different color, that simplifies debugging process. BASIC compiler's assembler output has all necessary comment lines, that makes it very useful for educational purposes, also.

Three data types are supported:

- Bit (1-bit, 0 or 1)
- Byte (1-byte integers in the range 0 to 255)
- Word (2-byte integers in the range 0 to 65,535)

Declarations may be placed anywhere in the program. All variables are considered global. The total number of variables is limited by the available microcontroller RAM memory. Variables are declared using

DIM statement:

```
DIM A AS BIT
DIM B AS BYTE
DIM X AS WORD
```

It is also possible to use one-dimensional arrays. For example:

```
DIM A(10) AS BYTE
```

declares an array of 10 Byte variables with array index in the range [0-9].

RESERVE statement allows advanced usage by reserving some of the RAM locations to be used by in-code assembler routines or by MPLAB In-Circuit Debugger. For example:

```
RESERVE 0x70
```

It is possible to make conversions between Byte and Word data types using .LB and .HB extensions or directly:

```
DIM A AS BYTE
DIM B AS WORD
```

```
A = B.HB
```

```
A = B.LB 'This statement is equivalent to A = B
```

```
B.HB = A
```

```
B.LB = A
```

```
B = A 'This statement will also clear the high byte of B variable
```

All special function registers (SFRs) are available as Byte variables in basic programs. Individual bits of a Byte variable can be addressed by .0, .1, .2, .3, .4, .5, .6 and .7 extensions or using official names of the bits:

```
DIM A AS BIT
DIM B AS BYTE
```

```
A = B.7
```

```
B.6 = 1
```

```
TRISA.1 = 0
```

```
TRISB = 0
```

```
PORTA.1 = 1
```

```
PORTB = 255
```

```
STATUS.RP0 = 1
```

```
INTCON.INTF = 0
```

Standard short forms for accessing port registers and individual chip pins are also available (RA, RB, RC, RD, RE can be used as Byte variables; RA0, RA1, RA2, ..., RE6, RE7 are available as Bit variables):

```
RA = 0xFF
RB0 = 1
```

It is also possible to use symbolic names (symbols) in programs:

```
SYMBOL LED1 = PORTB.0
LED1 = 1
SYMBOL AD_ACTION = ADCON0.GO_DONE
```

Constants can be used in decimal number system with no special marks, in hexadecimal number system with leading 0x notation (or with H at the end) and in binary system with leading % mark (or with B at the end). Keywords True and False are also available for Bit type constants.

For example:

```
DIM A AS BIT
DIM B AS BYTE
A = TRUE
B = 0x55
B = %01010101
```

Constants can also be assigned to symbolic names using CONST directive:

```
DIM A AS WORD
CONST PI = 314
A = PI
```

There are three statements that are used for bit manipulation - HIGH, LOW and TOGGLE. If the argument of these statements is a bit in one of the PORT registers, then the same bit in the corresponding TRIS register is automatically cleared, setting the affected pin as an output pin. Some examples:

```
HIGH PORTB.0
LOW ADCON0.ADON
TOGGLE OPTION_REG.INTEDG
```

Five arithmetic operations (+, -, *, /, MOD) are available for Byte and Word data types. The compiler is able to compile all possible complex arithmetic expressions. For example:

```
DIM A AS WORD
DIM B AS WORD
DIM X AS WORD
A = 123
B = A * 234
X = (12345 - B * X) / (A + B)
```

Square root of a number can be calculated using SQR function:

```
DIM A AS WORD
A = 3600
A = SQR(A)
```

For Bit data type variables seven logical operations are available. It is possible to make only one logical operation in one single statement. Logical operations are also available for Byte and Word variables. For example:

```
DIM A AS BIT
DIM B AS BIT
DIM X AS BIT
X = NOT A
X = A AND B
X = A OR B
X = A XOR B
X = A NAND B
X = A NOR B
```

```

X = A NXOR B

DIM A AS WORD
DIM B AS WORD
A = A OR B
PORTB = PORTC AND %11110000

```

There are two parameters CONF_WORD and CONF_WORD_2 (not available for all devices) that can be set using DEFINE directive to override the default values:

```

DEFINE CONF_WORD = 0x3F72

```

The GOTO statement uses line label name as argument. Line labels must be followed by colon mark ":". Here is one example:

```

DIM A AS WORD
A = 0
loop: A = A + 1
GOTO loop

```

WAITMS and WAITUS statements can be used to force program to wait for the specified number of milliseconds or microseconds. It is also possible to use variable argument of Byte or Word data type. These routines use Clock Frequency parameter that can be changed from the Options menu. WAITUS routine has minimal delay and step that also depends on the Clock Frequency parameter.

```

DIM A AS WORD
A = 100
WAITMS A
WAITUS 50

```

PLEASE NOTE: When writing programs for real PIC devices you will most likely use delay intervals that are comaprable to 1 second or 1000 milliseconds. Many examples in this help file also use such 'real-time' intervals. But, if you want to simulate those programs you have to be very patient to see something to happen, even on very powerful PCs available today. For simulation of 'WaitMs 1000' statement on 4MHz you have to wait the simulator to simulate 1000000 instructions and it will take considerable amount of time even if 'extremely fast' simulation rate is selected. So, just for the purpose of simulation you should recompile your programs with adjusted delay intervals, that should not exceed 1-10ms. But, be sure to recompile your program with original delays before you download it to a real device.

Access to EEPROM data memory can be programmed using READ and WRITE statements. The first argument is the address of a byte in EEPROM memory and can be a constant or Byte variable. The second argument is data that is read or written (for READ statement it must be a Byte variable). It is suggested to keep interrupts disabled during the execution of WRITE statement.

```

DIM A AS BYTE
DIM B AS BYTE
A = 10
READ A, B
WRITE 11, B

```

Three standard BASIC statements are supported:
FOR-TO-STEP-NEXT, WHILE-WEND and IF-THEN-ELSE-ENDIF.

Here are several examples:

```

DIM A AS BYTE
TRISB = 0
A = 255
WHILE A > 0
  PORTB = A
  A = A - 1
  WAITMS 100
WEND
PORTB = A

TRISB = 0

```



```

loop:
IF PORTA.0 THEN
  PORTB.0 = 1
ELSE
  PORTB.0 = 0
ENDIF
GOTO loop

```

```

DIM A AS WORD
TRISB = 0
FOR A = 0 TO 10000 STEP 10
  PORTB = A.LB
NEXT A

```

```

DIM A AS BYTE
DIM B AS BYTE
DIM X AS BYTE
B = 255
X = 2
TRISB = 0
FOR A = B TO 0 STEP -X
  PORTB = A
NEXT A

```

After IF-THEN statement in the same line can be placed almost every other possible statement and then ENDIF is not used. There are no limits for the number of nested statements of any kind.

LOOKUP function can be used to select one from the list of Byte constants, based on the value in the index Byte variable, that is supplied as the last separated argument of the function. The first constant in the list has index value 0. The selected constant will be loaded into the result Byte data type variable. If the value in the index variable goes beyond the number of constants in the list, the result variable will not be affected by the function. Here is one small example for a 7-segment LED display:

```

DIM DIGIT AS BYTE
DIM MASK AS BYTE
loop:
TRISB = %00000000
FOR DIGIT = 0 TO 9
  MASK = LOOKUP(0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07,
0x7F, 0x6F), DIGIT
  PORTB = MASK
  WAITMS 1000
NEXT DIGIT
GOTO loop

```

SHIFLEFT and SHIFTRIGHT functions can be used to shift bit-level representation of a variable left and right. The first argument is input variable and the second argument is number of shifts to be performed.

Here are two examples:

```

TRISB = 0x00
PORTB = %00000011
goleft:
WAITMS 250
PORTB = SHIFLEFT(PORTB, 1)
IF PORTB = %11000000 THEN GOTO goright
GOTO goleft
goright:
WAITMS 250
PORTB = SHIFTRIGHT(PORTB, 1)
IF PORTB = %00000011 THEN GOTO goleft
GOTO goright

```

```

TRISB = 0x00
PORTB = %00000001
goleft:
WAITMS 250
PORTB = SHIFTLLEFT(PORTB, 1)
IF PORTB.7 THEN goright
GOTO goleft
goright:
WAITMS 250
PORTB = SHIFTRIGHT(PORTB, 1)
IF PORTB.0 THEN goleft
GOTO goright

```

ADCIN statement is available as a support for internal A/D converter. Its first argument is ADC channel number and the second argument is a variable that will be used to store the result of A/D conversion. ADCIN statement uses two parameters ADC_CLOCK and ADC_SAMPLEUS that have default values 3 and 20. These default values can be changed using DEFINE directive. ADC_CLOCK parameter determines the choice for ADC clock source (allowed range is 0-3 or 0-7 depending on the device used). ADC_SAMPLEUS parameter sets the desired ADC acquisition time in microseconds (0-255). ADCIN statement presupposes that the corresponding pin is configured as an analog input (TRIS, ADCON1 register and on some devices ANSEL register). Here is one example:

```

DIM V(5) AS BYTE
DIM VM AS WORD
DIM I AS BYTE
DEFINE ADC_CLOCK = 3
DEFINE ADC_SAMPLEUS = 50
TRISA = 0xFF
TRISB = 0
ADCON1 = 0
FOR I = 0 TO 4
  ADCIN 0, V(I)
NEXT I
VM = 0
FOR I = 0 TO 4
  VM = VM + V(I)
NEXT I
VM = VM / 5
PORTB = VM.LB

```

Structured programs can be written using subroutine calls with GOSUB statement that uses line label name as argument. Return from a subroutine is performed by RETURN statement. User need to take care that the program structure is consistent. When using subroutines, main routine need to be ended with END statement. END statement is compiled as an infinite loop. Here is an example:

```

SYMBOL ad_action = ADCON0.GO_DONE
SYMBOL display = PORTB
TRISB = %00000000
TRISA = %11111111
ADCON0 = 0xC0
ADCON1 = 0
HIGH ADCON0.ADON
main:
GOSUB getadresult
display = ADRESH
GOTO main
END
getadresult:
HIGH ad_action
WHILE ad_action
WEND
RETURN

```

Interrupt routine should be placed as all other subroutines after the END statement. It should begin with ON INTERRUPT and end with RESUME statement. If arithmetic operations, arrays or any other complex statements are used in interrupt routine, then SAVE SYSTEM statement should be placed right after ON INTERRUPT statement to save the content of registers used by system. ENABLE and DISABLE statements can be used in main program to control GIE bit in INTCON register. RESUME statement will set the GIE bit and enable new interrupts. For example:

```
DIM A AS BYTE
A = 255
TRISA = 0
PORTA = A
INTCON.INTE = 1
ENABLE
END
ON INTERRUPT
  A = A - 1
  PORTA = A
  INTCON.INTF = 0
RESUME
```

```
DIM T AS WORD
T = 0
TRISA = 0xFF
ADCON1 = 0
TRISB = 0
OPTION_REG.T0CS = 0
INTCON.T0IE = 1
ENABLE
loop:
  ADCIN 0, PORTB
GOTO loop
END
ON INTERRUPT
  SAVE SYSTEM
  T = T + 1
  INTCON.T0IF = 0
RESUME
```

Basic compiler also features the support for LCD modules. Prior to using LCD related statements, user should set up LCD interface using DEFINE directives. Here is the list of available parameters:

LCD_BITS - defines the number of data interface lines (allowed values are 4 and 8; default is 4)
LCD_DREG - defines the port where data lines are connected to (default is PORTB)
LCD_DBIT - defines the position of data lines for 4-bit interface (0 or 4; default is 4), ignored for 8-bit interface
LCD_RSREG - defines the port where RS line is connected to (default is PORTB)
LCD_RSBIT - defines the pin where RS line is connected to (default is 3)
LCD_EREG - defines the port where E line is connected to (default is PORTB)
LCD_EBIT - defines the pin where E line is connected to (default is 2)
LCD_RWREG - defines the port where R/W line is connected to (set to 0 if not used; 0 is default)
LCD_RWBIT - defines the pin where R/W line is connected to (set to 0 if not used; 0 is default)
LCD_COMMANDUS - defines the delay after LCDCMDOUT statement (default value is 5000)
LCD_DATAUS - defines the delay after LCDOUT statement (default value is 50)
LCD_INITMS - defines the delay for LCDINIT statement (default value

is 100)

The last three parameters should be set to low values when using integrated LCD module simulator.

LCDINIT statement should be placed in the program before any of LCDOUT (used for sending data) and LCDCMDOUT (used for sending commands) statements. Its argument is used to define the cursor type: 0 = no cursor (default), 1 = blink, 2 = underline, 3 = blink + underline.

LCDOUT and LCDCMDOUT statements may have multiple arguments separated by ','. Strings, constants and variables can be used as arguments of LCDOUT statement. If '#' sign is used before the name of a variable then its decimal representation is sent to the LCD module. Constants and variables can be used as arguments of LCDCMDOUT statement and the following keywords are also available: LcdClear, LcdHome, LcdLine2Home, LcdLeft, LcdRight, LcdShiftLeft, LcdShiftRight, LcdLine1Clear, LcdLine2Clear, LcdLine1Pos() and LcdLine2Pos(). Argument of LcdLine1Pos() and LcdLine2Pos() statements can be a number in the range (1-40) or Byte data type variable. The value contained in that variable should be in the same range. Here are some examples:

```
DEFINE LCD_BITS = 8
DEFINE LCD_DREG = PORTB
DEFINE LCD_DBIT = 0
DEFINE LCD_RSREG = PORTD
DEFINE LCD_RSBIT = 1
DEFINE LCD_EREG = PORTD
DEFINE LCD_EBIT = 3
DEFINE LCD_RWREG = PORTD
DEFINE LCD_RWBIT = 2
DEFINE LCD_COMMANDUS = 10000
DEFINE LCD_DATAUS = 100
DEFINE LCD_INITMS = 1000
LCDINIT
```

loop:

```
LCDOUT "Hello world!"
WAITMS 1000
LCDCMDOUT LcdClear
WAITMS 1000
```

GOTO loop

```
DEFINE LCD_BITS = 8
DEFINE LCD_DREG = PORTB
DEFINE LCD_DBIT = 0
DEFINE LCD_RSREG = PORTD
DEFINE LCD_RSBIT = 1
DEFINE LCD_EREG = PORTD
DEFINE LCD_EBIT = 3
DEFINE LCD_RWREG = PORTD
DEFINE LCD_RWBIT = 2
DEFINE LCD_COMMANDUS = 10000
DEFINE LCD_DATAUS = 100
DEFINE LCD_INITMS = 1000
DIM A AS WORD
```

A = 65535

LCDINIT 3

WAITMS 1000

loop:

```
LCDOUT "I am counting!"
LCDCMDOUT LcdLine2Home
LCDOUT #A
A = A - 1
WAITMS 250
LCDCMDOUT LcdClear
```

GOTO loop

LCD related statements will take control over TRIS registers connected with pins used for LCD interface, but if you use PORTA or PORTE pins on devices with A/D Converter Module then you should take control over ADCON1 register to set used pins as digital I/O.

You can setup up to eight user defined characters to be used on LCD. This can easily be done with LCDDEFCHAR statement. The first argument of this statement is char number and must be in the range 0-7. Next 8 arguments form 8-line char pattern (from the top to the bottom) and must be in the range 0-31 (5-bits wide). These 8 user characters are assigned to char codes 0-7 and 8-15 and can be displayed using LCDOUT statement. After LCDDEFCHAR statement the cursor will be in HOME position. For example:

```
LCDDEFCHAR 0, 10, 10, 10, 10, 10, 10, 10, 10
LCDDEFCHAR 1, %11111, %10101, %10101, %10101, %10101,
%10101, %10101, %11111
LCDOUT 0, 1, "Hello!", 1, 0
```

The support for both hardware and software serial communication is also available. HSEROPEN, HSEROUT, HSERIN and HSERGET statements can be used with PIC devices that have internal hardware UART. HSEROPEN statement sets up the hardware UART. Its only argument is baud rate and allowed values are: 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 31250, 38400 and 57600. If the argument is omitted UART will be set up for 9600 baud rate. If parameter ALLOW_MULTIPLE_HSEROPEN is set to 1 using DEFINE directive, it will be possible to use HSEROPEN statement more than once in the program, for example to change selected baud rate. If ALLOW_ALL_BAUDRATES parameter is set to 1 using DEFINE directive all baud rates in the range 100-57600 will be allowed. HSEROUT statement is used for serial transmission. HSEROUT statement may have multiple arguments separated by ','. You can use strings, LF keyword for Line Feed character or CRLF keyword for Carriage Return - Line Feed sequence, constants and variables. If '#' sign is used before the name of a variable then its decimal representation is sent to the serial port. HSERIN statement can be used to load a list of Byte and Word variables with the values received on serial port. This statement will wait until the required number of bytes is received on serial port. HSERGET statement have one argument that must be a Byte variable. If there is a character waiting in the receive buffer it will be loaded in the variable, otherwise 0 value will be loaded. Here are some examples:

```
DIM I AS BYTE
HSEROPEN 38400
WAITMS 1000
FOR I = 20 TO 0 STEP -1
  HSEROUT "Number: ", #I, CrLf
  WAITMS 500
NEXT I
```

```
DIM I AS BYTE
HSEROPEN 19200
loop:
  HSERIN I
  HSEROUT "Number: ", #I, CrLf
GOTO loop
```

```
DIM I AS BYTE
HSEROPEN 19200
loop:
  HSERGET I
  IF I > 0 THEN
    HSEROUT "Number: ", #I, CrLf
    WAITMS 50
  ENDIF
GOTO loop
```

On all supported PIC devices you can use software serial communication routines with SEROUT and SERIN statements. The first argument of both statements must be one of the microcontroller's pins, and the second argument is baud rate: 300, 600, 1200, 2400, 4800 or 9600. For SEROUT statement then follows the list of arguments to be sent to serial port. You can use strings, LF keyword for Line Feed character or CRLF keyword for Carriage Return - Line Feed sequence, constants and variables. If '#' sign is used before the name of a variable then its decimal representation is sent to the serial port. SEROUT statement uses SEROUT_DELAYUS parameter that can be set by DEFINE directive and has default value of 1000 microseconds. This defines the delay interval before a character is actually sent to the port and it is used to increase the reliability of software SEROUT routine. For SERIN statement then follows the list of Byte and Word variables to be loaded with the values received on serial port. This statement will wait until the required number of bytes is received on serial port. For serial interface with inverted logic levels there are SERININV and SEROUTINV statements available.

Some examples:

```
DEFINE SEROUT_DELAYUS = 5000
SEROUT PORTC.6, 1200, "Hello world!", CrLf
```

```
DIM I AS BYTE
```

```
loop:
```

```
  SERIN PORTC.7, 9600, I
  SEROUT PORTC.6, 9600, "Number: ", #I, CrLf
GOTO loop
```

I2C communication can be implemented in basic programs using I2CWRITE and I2CREAD statements. The first argument of both statements must be one of the microcontroller's pins that is connected to the SDA line of the external I2C device. The second argument of both statements must be one of the microcontroller's pins that is connected to the SCL line. The third argument of both statements must be a constant value or Byte variable called 'slave address'. Its format is described in the datasheet of the used device. For example, for EEPROMs from 24C family (with device address inputs connected to ground) the value 0xA0 should be used for slave address parameter. Both statements will take control over bit 0 of slave address during communication. The fourth argument of both statements must be a Byte or Word variable (this depends on the device used) that contains the address of the location that will be accessed. If a constant value is used for address parameter it must be in Byte value range. The last (fifth) argument of I2CWRITE statement is a Byte constant or variable that will be written to the specified address, and for I2CREAD statement it must be a Byte variable to store the value that will be read from the specified address. It is allowed to use more than one 'data' argument. For I2C devices that do not support data address argument there is short form of I2C statements (I2CWRITE1 and I2CREAD1) available where slave address argument is followed with one or more data arguments directly. For some I2C slave devices it is necessary to make a delay to make sure device is ready to respond to I2CREAD statement. For that purpose there is I2CREAD_DELAYUS parameter that can be set by DEFINE directive and has default value of 0 microseconds. Here is one combined example with LCD module and 24C64 EEPROM (SDA connected to RC2; SCL connected to RC3):

```
DEFINE LCD_BITS = 8
DEFINE LCD_DREG = PORTB
DEFINE LCD_DBIT = 0
DEFINE LCD_RSREG = PORTD
DEFINE LCD_RSBIT = 1
DEFINE LCD_EREG = PORTD
DEFINE LCD_EBIT = 3
DEFINE LCD_RWREG = PORTD
DEFINE LCD_RWBIT = 2
DEFINE LCD_COMMANDUS = 10000
DEFINE LCD_DATAUS = 100
DEFINE LCD_INITMS = 1000
DIM ADDR AS WORD
DIM DATA AS BYTE
SYMBOL SDA = PORTC.2
SYMBOL SCL = PORTC.3
LCDINIT 3
WAITMS 1000
FOR ADDR = 0 TO 31
```

```

LCDCMDOUT LcdClear
DATA = 255 - ADDR
I2CWRITE SDA, SCL, 0xA0, ADDR, DATA
LCDOUT "Write To EEPROM"
LCDCMDOUT LcdLine2Home
LCDOUT "(, #ADDR, ) = ", #DATA
WAITMS 1000
NEXT ADDR
FOR ADDR = 0 TO 31
  LCDCMDOUT LcdClear
  I2CREAD SDA, SCL, 0xA0, ADDR, DATA
  LCDOUT "Read From EEPROM"
  LCDCMDOUT LcdLine2Home
  LCDOUT "(, #ADDR, ) = ", #DATA
  WAITMS 1000
NEXT ADDR

```

There is a set of low-level I2C communication statements available, if the user needs to get more control over I2C communication process. I2CPREPARE statement has two arguments that must be one of the microcontroller's pins. The first argument defines SDA line and second argument defines SCL line. This statement will prepare these lines for I2C communication. I2CSTART statement will generate start condition, and I2CSTOP statement will generate stop condition. One byte can be sent to the I2C slave using I2CSEND statement. After the statement is executed C bit in STATUS register will hold the copy of the state on the SDA line during the acknowledge cycle. There are two statements that can be used to receive one byte from I2C slave. I2CRECA or I2CRECEIVEACK will generate acknowledge signal during acknowledge cycle after the byte is received. I2CRECN or I2CRECEIVENACK will generate not acknowledge signal during acknowledge cycle after the byte is received. One example:

```

DIM ADDR AS WORD
DIM DATA(31) AS BYTE
SYMBOL SDA = PORTC.4
SYMBOL SCL = PORTC.3
ADDR = 0
I2CPREPARE SDA, SCL
I2CSTART
I2CSEND 0xA0
I2CSEND ADDR.HB
I2CSEND ADDR.LB
I2CSTOP
I2CSTART
I2CSEND 0xA1
FOR ADDR = 0 TO 30
  I2CRECEIVEACK DATA(ADDR)
NEXT ADDR
I2CRECN DATA(31)
I2CSTOP

```

It is possible to use comments in basic source programs. The comments must begin with single quote symbol (') and may be placed anywhere in the program.

Lines of assembler source code may be placed anywhere in basic source program and must begin with ASM: prefix. For example:

```

ASM:      NOP
ASM:LABEL1:  MOVLW 0xFF

```

Symbolic names of declared variables can be used in assembler routines because proper variable address will be assigned to those names by EQU directive:

```

DIM VARNAME AS BYTE
ASM:      MOVLW 0xFF
ASM:      MOVWF VARNAME

```