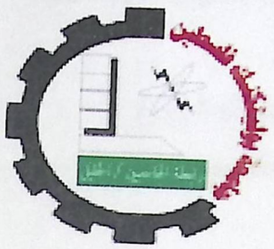
Palestine Polytechnic University



College of Engineering & Technology
Electrical &Computer Department

Graduation Project

Voice Recognition Robotic Car

Project Team
Abeer Abu Ghaith
Rami Ali Mohsen

Project Supervisor
Eng. Khalid AL-Daghameen

Hebron-Palestine

June , 2007

جامعة بوليتكنك فلسطين
الخليل- فلسطين
كلية الهندسة والتكنولوجيا

دائرة الهندسة الكهربائية والحاسوب

اسم المشروع :

Voice Recognition Robotic Car

أسماء الطلبة:

رامي علي محسن عبير أبو غيث

بناء على نظام كلية الهندسة والتكنولوجيا وإشراف ومتابعة المشرف المباشر وموافقة أعضاء اللجنة الممتحنة تم تقديم هذا المشروع إلى دائرة الهندسة الكهربائية والحاسوب وذلك للوفاء بمتطلبات درجة البكالوريوس في الهندسة تخصص هندسة أنظمة حاسوب

توقيع المشرف

.....

توقيع اللجنة الممتحنة

....

توقيع رئيس الدائرة

.....

Abstract

The project will be interested with designing a robotic car that will be controlled by human voice. The car will recognize five Arabic words (أمام، يمين، يسار، خلف، قف) according to the frequency content of the detected word, we will build the car using various electrical and mechanical domains such as digital signal processing, analog circuit design, and interfacing the car with the controller.

The system as known consists of three main parts: input, processing, and output. The project will start with microphone as input of the system that will be converted to digital signals (through ADC). The processing part includes filtering the digital signal, generating fingerprints, comparing generated fingerprints with fingerprint templates, and generating control signals according to comparison results. Finally applying these control signals to the car motors as output.

ملخص

يقوم المشروع على فكرة تصميم سيارة يتم التحكم بها من خلال الصوت البشري، تقوم السيارة بتمييز خمس كلمات عربية (أمام، يمين، يسار، خلف، قف) وذلك من خلال الترددات للكلمات المكتشفة، لذا سوف نبني السيارة مستخدمين المجالات الكهربائية والميكانيكية مثل معالجة الإشارات الرقمية، تصميم الدوائر الموجية، وعمل دائرة ربط بين السيارة والمتحكم (microcontroller).

النظام يتكون من ثلاثة أجزاء رئيسية: المدخلات، المعالجة، والمخرجات. وحدة الإدخال في المشروع تبدأ من المايكروفون، التي يتم فيها تحويل الإشارة من الموجية (Analog) إلى الرقمية (Digital). ثم يأتي قسم المعالجة الذي يقوم بترشيح الإشارات الرقمية ثم توليد الـ (fingerprints) ومقارنتها مع (fingerprint) templates ومن ثم توليد إشارة التحكم بناءً على نتيجة المقارنة, وأخيرا تطبق إشارة التحكم على محرك السيارة, فتتحرك السيارة بناء"على الأمر المعطى.

Table of contents

List of Tables

List of Figures

Glossary

ADC, A/D: Analog to Digital Converter.

DAC: Digital to Analog Converter.

PIC: Peripheral Interface Controller.

Fingerprint: the final state of the input spoken words after digitizing and filtering them.

WDT: Watchdog Timer

CPU: Central Processing Unit

RAM: Random Access Memory

EEPROM/EPROM/PROM/ROM: Electrical Erasable Programmable Read Only Memory

PC: Personal Computer

DC: Direct Current

PWM: Pulse Width Modulator

OpAmp: Operational Amplifier

MCLR: Master Clear. (a pin that exist on PIC18F4550)

1

Chapter One

Introduction

1.1 Overview.

1.2 Main Idea of the Project.

1.3 Project Objectives.

1.4 Literature Review.

1.5 Team Work.

1.6 Time Plan.

1.7 Project Schedule.

1.8 Estimated Cost.

1.9 Project Benefits.

1.10 Project Risk Management.

1.11 Report Contents.

1.1 Overview

As we start the 21 century, we are living in the age of modern technology. While all over the world life is mastered by computer and technology, Robots are expected to do every task.

This project is an outlook to the future and we hope it will be a simple participation in the field of voice recognition.

Voice recognition has reached a level of accuracy and ease of use that it is now used and accepted by the general public. It is a valuable and essential tool for people, especially people with physical disabilities who cannot type as well as for those with learning disabilities. Using speech, it is now possible to dictate commands to control the Windows desktop and applications search the web, send emails. We maintain a client base of individuals, workers in government and private companies who are able to work with the assistance of their voice recognition systems.

1.2 General Idea

The design of the project focuses on building a small-vocabulary (5 Arabic words) voice command recognition system to be used in directing the movements of a small "robotic car" in real time. This project task is to build a robust and accurate

system that is intelligent enough to recognize spoken arabic word commands and operate with reasonable speed. The commands we will use are as follows:

- أمام (Forward)
- خلف (Backward)
- قف (Stop)
- يسار (Left)
- يمين (Right)

The project consists of two main parts: the software component which will be built to carry out the actual speech recognition and the hardware component (Robotic car) which is designed to carry out the spoken commands.

The system will use a microphone attached to a personal computer to obtain voice samples for template creation and training. The system will process the training input using algorithms written in MATLAB 7.

The proposed system needs a microphone, toy car, filtering circuit, microcontroller that contains analog to digital circuit. The first step in the system is for the user to speak a word into a microphone. The electrical signal from the microphone is digitized by an "analog-to-digital (A/D) converter", and is stored in memory. To determine the "meaning" of this voice input, the computer attempts to match the input with a digitized voice sample, or template that is stored in the EPROM. The program deals with the input template, and attempts to match this

template with the actual template, then transfers the accepted command into toy car to be carried out.

The difficulty of this project is in using voice as an input to a computer simulation lies in the fundamental differences between human speech and more traditional forms of computer input; While computer programs are commonly designed to produce a precise and well-defined response upon receiving the proper input, the human voice and spoken words are anything but precise. Each human voice is different, and identical words can have different meanings if spoken with different inflections or in different contexts.

1.3 Project Objectives

This project accomplishes the following:

- Build a system that controls the movement of robotic car and accepts a voice signal as control command in four directions probably.

- The ability of the voice recognition system to operate without physical prompting (control should be achieved through voice alone, without requiring the user to press keys or to issue a record command by pressing some button. (i.e. .system, once started, must be completely hands-free).
- The ability to interpret and receive commands (discrete words) by voice recognition systems.

- Speaker dependent system.

- The ability to carry out those commands quickly and physically (this relies on the microcontroller's ability to receive and interpret the commands within the real time).

- The ability of the speech recognition system to gear towards the control of a car.

1.4 Literature Review

There were several people who were interested in studying voice recognition and building projects in this field .These projects are:

ROBOKART, by Adrian Abordo, Jon Liao: This project was focused on building a small-vocabulary (24-word) voice command recognition system to be used in directing the movements of a small "robokart" in real time. It was built using a Motorola 68HC11 microcontroller which was embedded on the CME119-EVBU evaluation board. [1]

Cell Phone-Based Controller for a Toy Car , by Ahmad Hasasneh,Bashar Jaradat, Sahar Quasmeh: This project was designed with a main control circuit to navigate a toy car as an application by human voice for controlling the car remotely, monitoring and sensing data using voice command, this was done through two cell

phones .One of them is any valid cell phone with a client, the other one was built inside the toy car and connected to the control circuit that contains the whole chips that the system needs. The heart of the system is the 8051 microcontroller. And the program code which is designed to control the toy car was written in assembly language. [2]

Voice Recognition Security System, Xiaowen Lu and Shihjia Lee: The function of this speech recognition security system is to have a system that will only unlock upon recognizing a voice password spoken by the administrator or password holder.

The key point of this project is how to design filters and how to implement them. There were two major difficulties needed to be solved: the first difficulty was to reduce the running time of each filter in order to get all the fingerprints before next new sample comes, so they had to use the fixed-point algorithm. The second one was to set the reasonable cutoff rate for each filter and to set the number of stages of the filters. [3]

1.5 Team Work

The supervisor of our project is Eng. Khalid Daghameen. The team work consists of Rami Mohsen and Abeer Abu Ghaith.

1.6 Time Planning

In this section the time schedule must be determined in specific way and includes the related tasks of study and system analysis.

The time planning consists of two time estimation schedules; the first one demonstrates what is done in the first semester and the second demonstrates the expected scheduling time of the second semester.

1.4.1 The First Time Planning

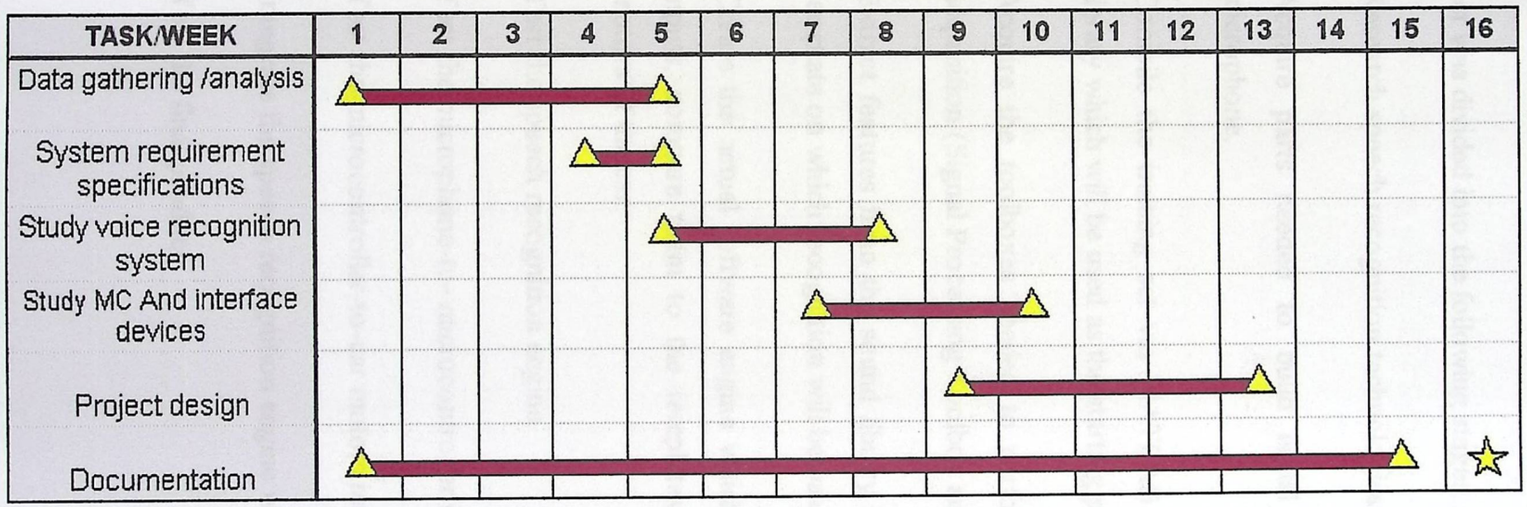


Figure1.1: The First Time Planning

1.6.2 The Second Time Planning

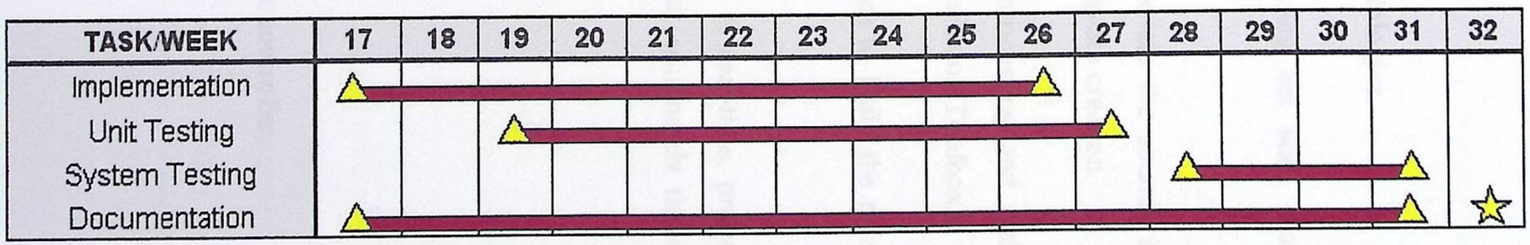


Figure1.2: The Second Time Planning

1.6 Project Schedule

The project was divided into the following stages:

- Research speech recognition technologies and methodologies.
- Acquire parts needed to build initial voice training set such that PC microphone.
- Compile the training set via MATLAB and incorporate the sounds into a library which will be used as the starting point for template creation.
- Acquire the toolboxes needed to perform signal processing and real-time acquisition (Signal Processing Toolbox and Data Acquisition Toolbox).
- Extract features from the sound library and use them to build the reference template on which recognition will be based.
- Create the actual software engine which will work in real-time, process new inputs, compare them to the templates, and output commands through the motors of the car.
- Test the speech recognition engine.
- Test the microphone-to- microcontroller interface.
- Test the microcontroller-to-car motors interface.
- Integrate the speech recognition engine with the microcontroller.
- Test the final product.

1.8 Estimated Development Cost

The following table shows the main and necessary components that are needed for the system. The prices are in US dollars.

Table 1.1 Estimated Development Cost

IC	UNITS	TOTAL
Microcontroller Chip	2	48$
Toy car	1	20$
Microphone	1	10$
Resisters	100	10$
LM324	2	5$
MATLAB student edition	1	10$
Wires and Cables	-	30$
Crystal	1	2$
Capacitors	10	3$
Transistors	12	5$
Invertors Chip	7	5$
Connectors DB25	1	2$
H-bridge(MAX4427)	4	10$
Printing	-	100$
Computer Rents	-	50$
Internet	-	70$
TOTAL		380$

1.9 Project Benefits

There are several benefits for the project as follow:

- This project will be very useful for people who can't use legs and hands.
- The speech recognition system is geared up towards the control of an instrument.
- Providing a fairly natural and intuitive way of controlling the simulation while allowing the user's hands to remain free.

1.10 Project Risk Management

1.10.1 Hardware Risk

The most important hardware part in our project is the microcontroller. The predicted risks are:

- Device failure: the microcontroller may crash because of high voltage supply or other problems.
- The device operates differently than intended.
- Car doesn't react correctly due to a specific response due to many causes.

- Improper readings of analog signal.
- Addition of noise from the outer environment.
- The car may become overloaded by the weight of system components.

1.10.2 Software Risk

There are two important software components to be used in our system. These are C language of the PIC and MATLAB software for template generation and training code. There may be some risks with the software such as:

- Difficulty of handling the noise cause of unknown reasons
- Problems that will appear during the analysis of the voice in MATLAB; to determine the main frequencies spectrums for all spoken words.
- Disability of the car to match the nearest word to the spoken word; because of the sickness of the person...
- Appearance of viruses in the program, which may lead to improper execution.

1.10.3 Team Risk

- Absence of one or more of group members due to some reasons such as occupations or illness.

1.10.4 Requirement Risks

- Several requirements may be affected by other requirements due to the high cost of some components.
- Team fails to understand impact of requirements changes.

1.10.5 Project Risks

- Some requirements need changes may arise lately.
- Schedule not accurate.
- Budget not sufficient.
- Supervisor change.
- Latency of devices arrival.

1.11 Report Contents

The documentation of this project is divided into seven chapters; each chapter focused on the parts of the system as the following

Chapter one describes the general idea about the project and its important , the general review of the system, project objective, project risk management (hardware and software), system cost, scheduling time and report scheduling.

The second chapter shows the theoretical subjects related to the main ideas of the project, (hypothesis, hardware, and software) related to the project, project integrity and theoretical background about project components.

Chapter three demonstrates detailed project objectives, system design options and justifies those that are chosen in the project, design realization approach, project interaction with the surrounding environment and general block diagram that shows how the system works.

Chapter four shows detailed description about the different project phases also is included subsystem detailed design and overall system design.

Chapter five handles the software related to our system, depicts flow charts about system operation and illustrates different algorithms and techniques that will be considered in writing the software.

Chapter six will manifest the implementation procedures to be acted so as to integrate the project. Then, a sequence of procedural testing will be listed. The testing comprises both software and hardware testing.

Chapter seven will list the problems that faced us in accomplishing the system and how they were solved. Notes and Conclusions that will help readers are also included. A future work is also proposed.

2

Chapter Two

Theoretical Background

2.1 Introduction.

2.2 Theoretical Background of the Project.

2.3 Hypothesis, Hardware, and Software related to the Project.

2.3.1 Voice Capture Unit.

2.3.2 Voice Recognizer Unit.

2.3.3 Navigation Unit.

2.3.4 Programmer Unit.

2.4 Project Integrity.

2.5 Project Components.

2.1 Introduction

This chapter discusses the main theoretical subjects which are needed to explain the main ideas of this system. And will illustrate the hypothesis, hardware and software related to the project. Also will argue the project integrity and demonstrate the theoretical background of the project components.

2.2 Theoretical Background of the Project

There are some theories and truths that the project depends on while performing it, some of these are:

- **A person's vocal cords**

A person's vocal cords cause air to vibrate and generate sound waves. These waves travel through the air to the ear where the brain interprets the sounds. Words consist of speech sounds, which are known as phonemes. They have characteristics which allow humans to identify them. [4]

Figure 2.1: Human Ear

- **Frequency Range**

The frequency range of sounds perceived by humans is nearly between 30 Hz (hertz, cycles per second) and 20,000 Hz. Above 20 KHz we hear nothing. Human hearing is not equally sensitive to all frequencies. At very high frequencies, a sound must be louder to be perceived. The same is true at very low frequencies. The optimal frequency is about 1000 Hz. A sound at 1000 Hz is more easily heard than another sound with the same amount of energy at any other frequency. [5]

- **Frequency Response:**

The best microphones have a uniform frequency response, meaning that they "hear" sounds equally well from a broad range of frequencies. For voice recognition, the microphone should have a good frequency response throughout the normal human hearing range.[5]

- **Sampling Process:**

Sampling is the process of converting a signal (for example, a function of continuous time) into a numeric sequence (a function of discrete time).

The theorem (Nyquist) states that:

> "Exact reconstruction of a continuous-time base band signal from its samples is possible if the signal is band limited and the sampling frequency is greater than twice the signal bandwidth".[6]

- **Voice Recognition**

Voice recognition is the technology by which sounds, words or phrases spoken by humans are converted into electrical signals, and these signals are transformed into coding patterns that can be identified by a computer. Based on this identification, the computer usually takes some action.[7]

- **Framing**

Framing involves separating the sample data into specific sizes. This also involves preparing the sample boundaries for analysis (removing edge clicks, etc.)

- **Feature Analysis**

The method of voice recognition where words and phrases are distinguished from each other based on differences in certain characteristics of the sounds. These characteristics include timing, frequency content. [8]

- **Fourier Transform:**

Since frequency is one of the important parts of information that is necessary to accurately recognize sound, it is necessary to have a transformation that allows one to break a signal into its frequency components. The Fourier transform of a signal is the representation of the frequency of that signal.

- **Template Matching** :

Template matching is a form of pattern recognition, where each word or phrase in an application is stored as a separate template. The input is then compared with the stored templates and the template that most closely match to the incoming speech pattern is identified as the recognized word or phrase. The selected template is called the best match for the input. The input samples are compared with each of the training sets and the one with the least Euclidean distance. [9]

- **The Euclidean Distance (deucl)**

The Euclidean distance is the straight line distance between two points and is derived from the Gaussian distribution. The Euclidean distance in a K dimensional space is defined as

$$d_{eucl}(x\|y) = \sqrt{\sum_{k=1}^{K} (x(k) - y(k))^2}$$

The squared of the Euclidean distance is often referred as the Mean squared Error criterion. [10]

- **Accuracy**

The ability of a recognizer can be examined by measuring its accuracy – or how well it recognizes utterances. This includes not only correctly identifying an utterance but also identifying if the spoken utterance is not in its vocabulary. Good speech recognition systems have an accuracy of 98% or more! The acceptable accuracy of a system really depends on the application.[11]

- **Factors Affecting Voice Recognition**

There are many factors that affect voice recognition process such as:

1. **Isolated Words**

The system that operates single words at a time requires a pause between saying each word.

2. **Vocabulary Size**

A vocabulary is the collection of words that the pattern-matching algorithm "knows" and compares the input against. Larger vocabularies are more likely to contain ambiguous words than small vocabularies.

3. **Environment**

The setting can have an effect on the recognition accuracy. Background noise and changes in the microphone characteristics and loudness can all influence the recognition process. The following points can influence the recognition process:

- Misspoken or misread prompted phrases.
- Extreme emotional states (e.g., stress or duress).
- Time varying (intra- or intersession) microphone placement.
- Poor or inconsistent room acoustics (e.g., multipath and noise).

- Channel mismatch (e.g., using different microphones for enrollment and verification).
- Sickness (e.g., head colds can alter the vocal tract).
- Aging (the vocal tract can drift away from models with age).

4. A Speaker Dependent

A speaker-dependent is a system that requires samples of speech from individual speakers. It requires users to participate in training sessions that "teach" the computer to recognize the user's voice. The computer then makes a voice profile that matches the require training.

The different systems are classified according to the methodologies used to attain these goals. There are speaker dependant systems with discrete or continuous speech recognition and speaker independent systems with discrete or continuous speech recognition as shown in figure 2.2. In this system we will use speaker dependent and discrete word system. [4]

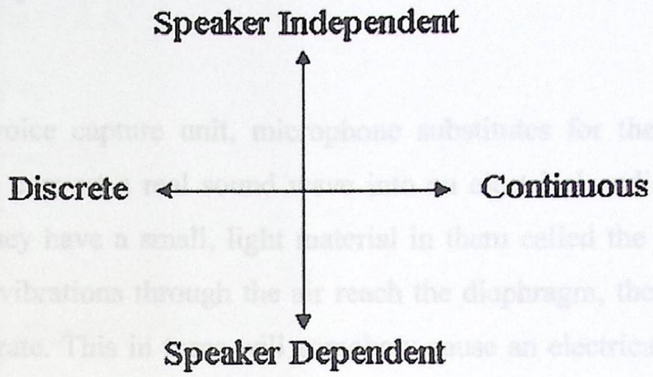


Figure 2.2 Methodologies of Speech System

2.3 Hypothesis, Hardware and Software related to the Project

The system aims to control a toy car by human voice recognition using microphone according to this idea the system has these main units:

- Voice Capture Unit.
- Voice Recognizer Unit.
- Navigation Unit.
- Programmer Unit.

2.3.1 Voice Capture Unit

In the voice capture unit, microphone substitutes for the ear drum. Microphones just convert a real sound wave into an electrical audio signal. In order to do so, they have a small, light material in them called the diaphragm. When the sound vibrations through the air reach the diaphragm, they cause the diaphragm to vibrate. This in turns will somehow cause an electrical current in the microphone to vary, whereupon it is sent out to a mixer, preamplifier or amplifier for use.

Figure 2.3 Microphone

Microphones are typically classified according to how the diaphragms produce sound.

The first step for the user is to speak a word into a microphone, the microphone include three stages as follow:

- Low Pass Filter.
- Amplifier.
- High Pass Filter.

2.3.2 Voice Recognizer Unit

Voice recognizer unit performs three primary tasks as shown in Figure. 2.4.

- **Preprocessing:** converts the spoken input into a form that the recognizer can process (digital form). The system will use the analog to digital converter that exits inside the microcontroller.

- **Recognition:** identifies what has been said by comparing the input with the built-in reference models. To accomplish these tasks the system needs to perform feature extraction process that extracts the main information from the signal by using Fourier transform. Fourier transform is used to determine the frequency components of a particular signal. This procedure shows what high, medium, and low sounds, and what volume of each, are combined to make a complex sound like the human voice, and PIC to store the template in.

The recognition process includes using of much software that is necessary to accomplish this process:

- MATLAB software: MATLAB is used in this project as voice analyzer program that will provide the project with the needed template to be compared with input voice template.

- C program: The C program is used as voice recognizer program that will do the matching algorithms and take the result of matching to perform appropriate action to the car.

- MPLAB software: MPLAB is software that will debug the program, insert appropriate files to the program, find errors and fixed them and generate .HEX file that will be entered to microcontroller.

- WinPic800 software: It is the PIC18F4550 Programmer that will take the .HEX file, read it and program the controller to be ready for independent use.

- **Communication:** The task of the communication unit is to send the recognized input from the PIC to the toy car to carry out the commands through H-bridge.

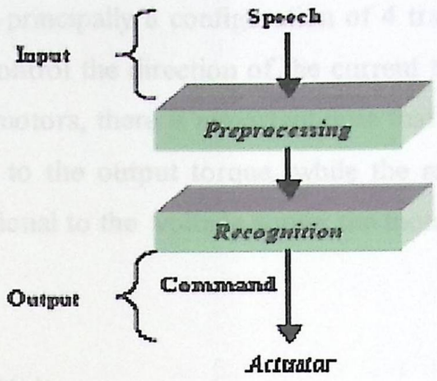


Figure 2.4: The Structure of Speech Recognition System

2.3.3 Navigation Unit

This unit includes the toy car that has the ability of moving in four directions and H-Bridge circuit that interface between the PIC and toy car.

- **The H-bridge & DC Motors**

An H-bridge is an electronic circuit which enables DC electric motors to be run forwards or backwards. These circuits are often used in robotics. H-bridges are available as integrated circuits, or can be built from separate components.

The H-Bridge is principally a configuration of 4 transistors that are arranged in a specific manner to control the direction of the current through the motor. While we are talking about DC motors, there is important note that : "Current flowing though a motor is proportional to the output torque, while the angular velocity (rpm) of the output shaft is proportional to the voltage across the motor windings".

2.3.4 Programmer Unit

The programmer unit is a step which includes building of the programmer circuit for PIC18F4550 that can be transfer the assembly code to the controller

2.4 Project Integrity

This project can be applied in many positive sides, for instance, it can help people with special needs to drive the cars and finish their works without any help from others. So the project will make there life easier and more comfortable.

This project can provide the users with a large system of security, such as preventing children from driving the car without their parents' knowledge.

2.5 Project Components

2.5.1 Microphone

Microphone is transducer used to convert sound in the environment to an analog signal. The microphone leads have polarity, meaning it mattered which side was connected to ground and which was connected to Vcc, as shown in figure 2.5. According to the microphone specification sheet, the standard operating voltage is 2 V. A voltage divider was used to take 2 V from the 5V source on the breadboard.

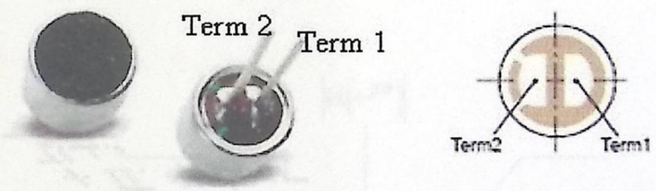


Figure 2.5 Polarities of Microphone Leads

The first step in the system is to speak to a microphone. The microphone consists of three main steps

- **Low Pass Filter**: A filter that passes low frequencies and attenuates high frequencies.

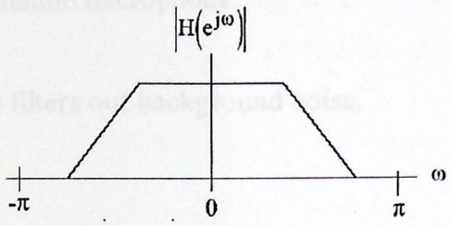


Figure 2.6 Low pass filter ($H(e^{j\omega})$ is the signal power)

- **Amplifier**: The signal that comes from the high pass filter will be amplified by using an operational amplifier (op-amp).

- **High pass filter**: A filter that passes high frequencies and attenuates low frequencies.

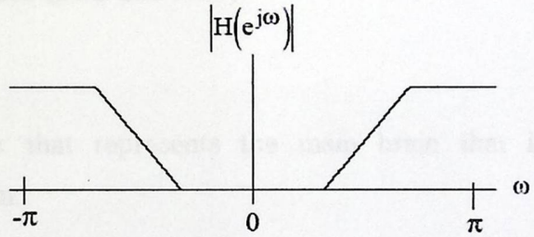


Figure 2.7: High Pass Filter ($H(e^{j\omega})$ is the signal power)

Microphone Features and Specifications

A quality of microphone is a key of point when utilizing speech recognition. So the most microphone features are:

- Noise cancellation microphone.
- Microphone filters out background noise.

Its specifications:

- Sensitivity: -67dB/μBar, -47dBV/pascal $^{+}$. 4B.
- Mic power source voltage: 1.5 V DC.
- Impedance: 2000 Ohms.
- Frequency response: 100-16,000 Hz.

2.5.2 Microcontroller (PIC 18F4550)

Microcontroller that represents the main brain that interface integrated circuits in the system.

The system needs a special microcontroller that should have the capability of storage, quick response and internal analog to digital converter with maximum number of channels and bits number. As a result of researches, the team has found the PIC18F4550 microcontroller chip that has all needed parts of a controller.

The PIC18F4550 microcontroller chip provides an easy-to-use and low-cost method to control several processes of the toy car. The built-in ports on the chip are an integrity system to execute tasks.

The microcontroller includes:

- CPU (central processing unit).
- RAM (Random Access Memory).
- EPROM/PROM/ROM (Erasable Programmable Read Only Memory).
- I/O (input/output).
- DAC \ ADC ports.
- Interrupt controller.

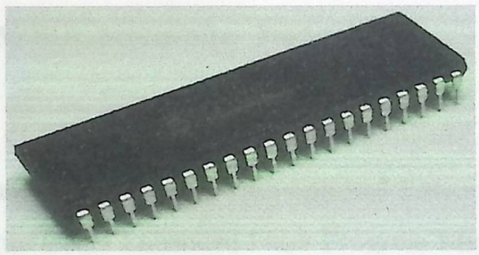


Figure 2.8: PIC 18F4550 Microcontroller

The microcontroller has an included package of features. These main features are:

- DC – 48 MHz Operating Frequency.
- 32768 (Bytes) Program Memory.
- 16384 (Instructions) Program Memory.
- 2048 (Bytes) Data Memory.
- 256 (Bytes) Data EEPROM Memory.
- 20 Interrupt Sources.
- A, B, C, D, E I/O Ports.
- 4 Timers.
- 1Capture/Compare/PWM Modules, 1 Enhanced Capture/Compare/PWM Modules.
- 13 Input Channels 10-Bit Analog-to-Digital Module.
- 75 Instructions; 83 with Extended Instruction Set enabled.

Special PIC18F4550 Microcontroller Features

- C Compiler Optimized Architecture with optional Extended Instruction Set.
- 100,000 Erase/Write Cycle Enhanced Flash Program Memory typical.
- 1,000,000 Erase/Write Cycle Data EEPROM Memory typical.
- Flash/Data EEPROM Retention: > 40 years.
- Self-Programmable under Software Control.
- Priority Levels for Interrupts.
- Programmable Code Protection.
- Wide Operating Voltage Range (2.0V to 5.5V).

2.5.3 Analog to Digital Converter

The microcontroller PIC 18F4550 has Analog-to-Digital (A/D) converter module that has 10 inputs for the 28-pin devices and 13 for the 40/44-pin devices. This module allows conversion of an analog input signal to a corresponding 10-bit digital number.

The analog reference voltage is software selectable to either the device's positive and negative supply voltage (VDD and VSS) or the voltage level on the RA3/AN3/VREF+ and RA2/AN2/VREF-/CVREF pins.

10-bit digital number can be read as integer number (N). N is converted to its corresponding voltage value using the following equation

$$N = \frac{2^{10}}{(Vref+ - Vref-)} Vin$$

precision :

$$\text{if } N=1 \qquad dVin = \frac{Vref+ - Vref-}{1024}$$

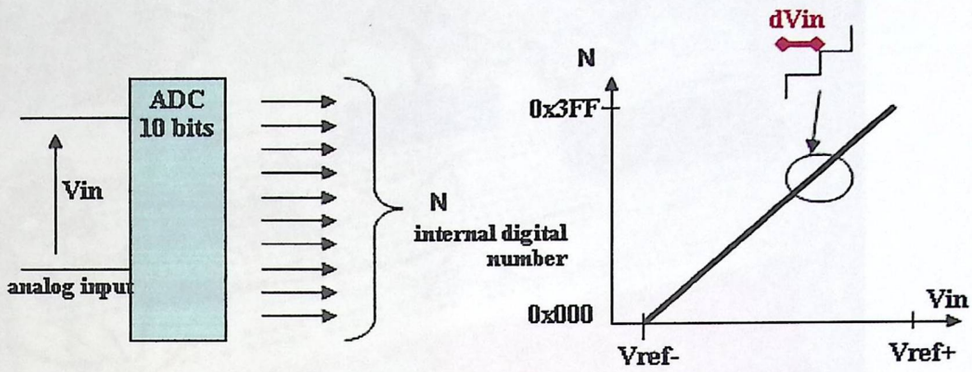


Figure 2.9: Analog-to-Digital (A/D) converter module

2.5.4 PIC18F4550 Programmer

Most of the programming devices (called programmers in PIC jargon) as well as the programming environments, only work in Windows, leaving Mac users out-played by Windows colleagues. Even though there exist several compiler options (gputils, CCS) and at least one IDE (PICC lite) for Mac OS X. The availability of programmers (device programmers) that are out-of-the-box compatible with Mac OS X is inexistent. Needless to say that overall solutions that include the IDE for a high level language (C, Pascal or Basic) with the programmer are also absent in Mac OS X.

The PIC programmer in this project will program PIC Microcontroller devices via PC parallel port in windows environment which supports different software such as EPICWin, WinPic800, and P18 etc. There are two indicator LEDs on figure 2.10. One for power supply and another for programming progress.

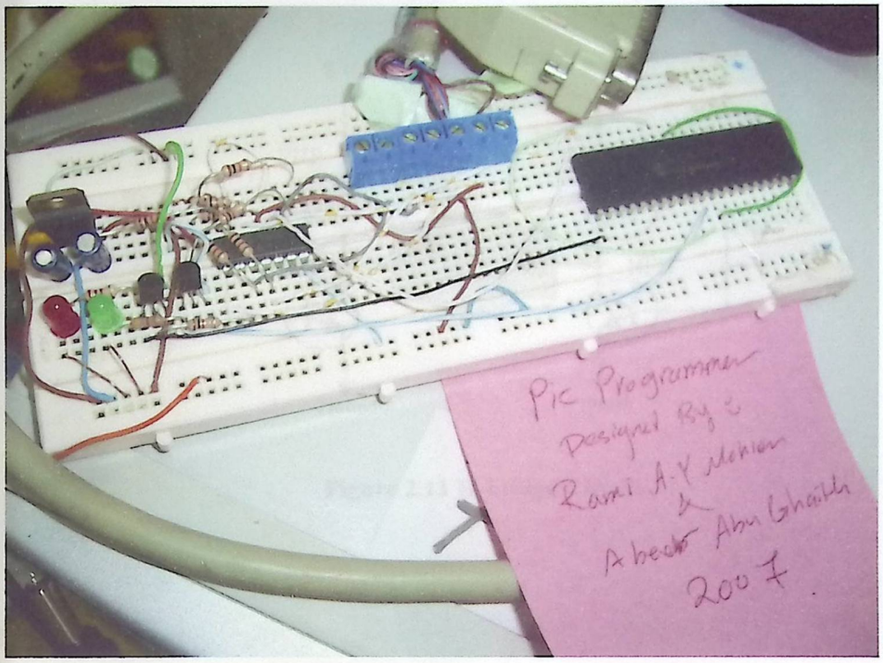


Figure 2.10: The PIC18F4550 Programmer.

2.5.5 H-bridge

A very popular circuit for driving DC motors is called an H-bridge. It's called that because it looks like the capital letter 'H' on classic schematics as shown in figure 2.11. The great ability of an H-bridge circuit is that the motor can be driven forward or backward at any speed, optionally using a completely independent power source.

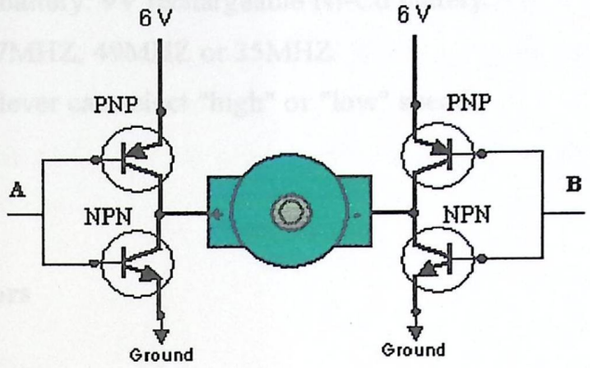


Figure 2.11 H-bridge Circuit

2.5.6 Car

There are several types of toy cars that can be used in many applications. This system needs only one type of toy cars that have the most popular features that the system needs.

Figure 2.12: Toy Car Form.

- **Car Features**

1) 1:10 scale super –speed RC car
2) Full function: left, right, forward, backward, stop

3) Charger and battery: 9V rechargeable Ni-Cd battery.

4) Frequency 27MHZ, 49MHZ or 35MHZ

5) 2 step speed lever can select "high" or "low" speeds.

- **Car Motors**

The toy car has the ability of moving in four directions. It consists of two main DC motors of maximum 9V.These motors are used to obtain the required voltage of each execution command. The first motor is used to control both Forward and Backward directions, while the second is used to navigate both Right and Left direction.

The right–angle and left-angle are fixed from the manufactured company and they equal 45. So every command to the right-left, motor will rotate the car 45 degree.

- **Toy Car Size and Weight**

The toy car has a medium size that consists of the motors, battery and the whole body. This indicates that the size is suitable to be an experimental surface for team application.

The car is nearly 1.2 Kg weight. It is tested to load up to 1.5 Kg, while keeping the speed as it is. These data is comfort to build our own circuits that may not weighted more than the maximum weight.

The dimensions of the toy car relative with its size. The tall of the car is 30 cm while the width is 20 cm.

- **Toy Car Battery**

The toy car needs 9V DC batteries that must be connected to operate the car motors and the microcontroller.

3

Chapter Three

Design Concepts

3.1 Introduction.

3.2 Project Objectives.

3.3 Design Options.

3.4 Design Realization Approach.

3.5 General Block Diagram.

3.6 How Does System Works?

3.7 Project Interaction with Environment.

3.1 Introduction

In this project there is a need to select a suitable car with acceptable size to control its navigation. The team has to setup the required microphone and other components that has the best specifications as mentioned in chapter 2, we wanted to develop a voice recognition system that was flexible, but not necessarily speaker-independent (for speaker-independence is a very hard thing to achieve). Since the voice recognition system is geared towards the control of an instrument, we placed particular importance on accuracy and robustness, envisioning that this system could be one day incorporated into the hands-free control of certain devices and instruments (microwave oven, car dash board, non-critical airplane flight controls).

On a personal level, we wanted a system that relied on our own ingenuity and originality. We therefore used current voice recognition approaches as a starting point for building a knowledge base, but we tried not to copy approaches that are already known to work. In short, we wanted to come up with a voice recognition system that was, for the most part.

3.2 Project Objectives

This project has many ideas and objectives that can be summarized as following:

- Build a car in wired technology based on microphone and PIC18F4550 microcontroller.

- The ability of the system to recognize 5 spoken Arabic commands (أمام Forward in English, خلف Backward in English, يمين Right in English, يسار Left in English), and take the appropriate action according to the spoken word.
- Recognition catch rate (RCR) of at least 50 percent (i.e. the spoken word should appear in the "Recognized Word Buffer" for at least half the duration that it was spoken).

- Ability of the recognition system to operate without physical prompting (control should be achieved through voice alone, without requiring the user to press keys, signal an intent to speak, or to issue a record command by pressing some button. System, once started, must be completely hands-free).

- The ability to design a system that minimizes the probability of verification errors. Thus, the underlying objective is to discriminate between the given speaker and all others.

- Ability to perform word-recognition within 1 second of when the person has finished speaking (near real-time performance).

- Ability to physically carry out those commands quickly (this relies on the microcontroller's ability to receive and interpret the serial commands within 1 second).

- Total system lag from voice-input to car response of no more than 3 seconds (it would be disastrous to say "قف Stop in English," but car responds too slowly and plows into a wall).

3.3 Design Options

From the strategies that were used in this project the strategy of replacing chips and circuits with others in case of non availability or in case of damage. And there are some of these alternate strategies that were used:

- We used microcontroller of type PIC18F4550 as a substitute of DSP processor because of:

    1. Lack of DSP processor in the local markets.
    2. Non availability of programmers for DSP processor in our university.
    3. Using the DSP processor needs too much time to learn , and the time of the project is limited.

- Internal ADC of PIC18F4550 which is used to convert the analog signal that comes from microphone to digital signal was used instead of building external ADC for the following reasons :

    1. To increase the response of the system so increasing the execution of the commands.
    2. To reduce the size of the final circuit of the system to make the car load light.

- Although H-bridge chip (MAX4427) increases the response and reduces the size of the final circuit for the system, but H-bridge chip (MAX4427) doesn't provide the car with enough current .So we built an equivalent H-bridge consists of 4 transistors for each car motor instead of H-bridge chip (MAX4427).

3.4 Design Realization Approach

Design realization approach can be of three type's .Actual implementation, modeling, and simulation. But our project will apply the modeling approach. A toy car will be used as a model for a real car. Other approaches such as actual implementation can be used in this project, but applying this project using this approach will need more experience specially in mechanical engineering, also it will cost more many, also applying it on the project will need more time than we have .So using the model approach will be the more suitable and efficient choice for the project. Besides of that, the project goal is to recognize spoken word rather than studying the mechanical sides of the project.

3.5 General Block Diagram

The system as whole consist of many subsystems that must explained (some blocks will have a number written inside them. This indicates that the block is composed of several sub-components which are not visible in that particular diagram but which will be covered in more detail later on "How Does System work?" section).

The general block diagram of this project shown as following:

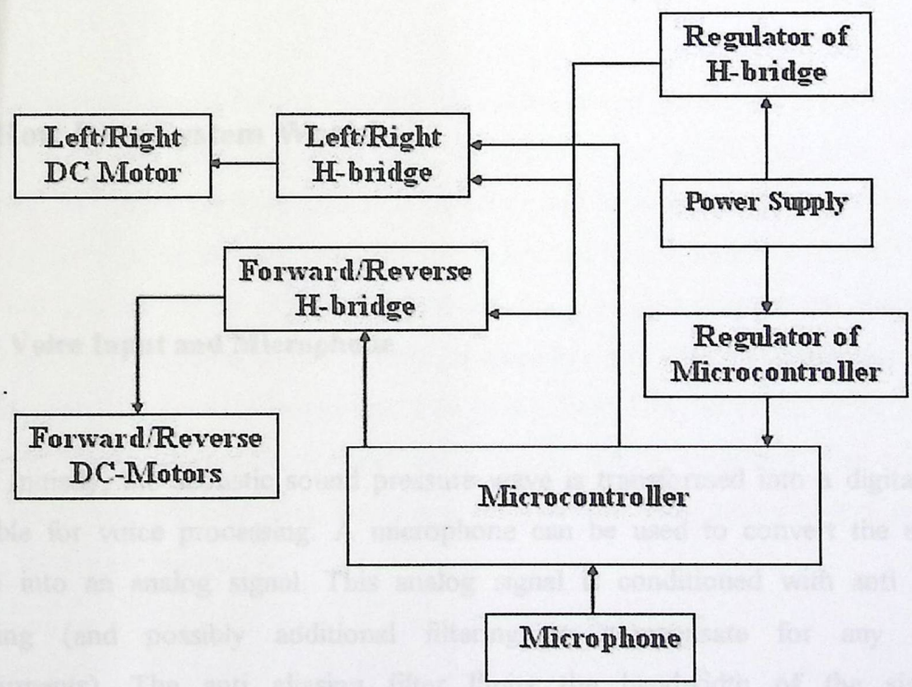


Figure 3.1The General Block Diagram of the System

3.5.1 Behaviors for a the System

Behaviors of the system will go through several of steps; these steps are shown as in figure 3.2

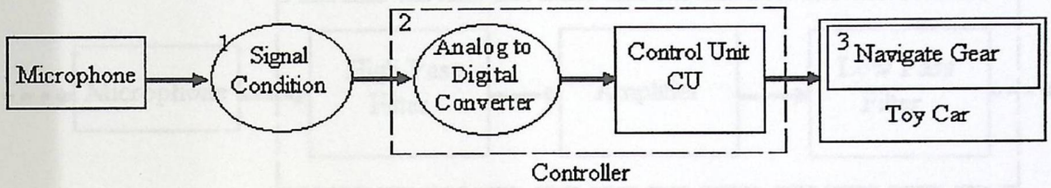


Figure 3.2: Behavior of the System

3.6 How Does System Work?

3.6.1 Voice Input and Microphone

Initially, the acoustic sound pressure wave is transformed into a digital signal suitable for voice processing. A microphone can be used to convert the acoustic wave into an analog signal. This analog signal is conditioned with anti aliasing filtering (and possibly additional filtering to compensate for any channel impairments). The anti aliasing filter limits the bandwidth of the signal to approximately the Nyquist rate (half the sampling rate) before sampling.

3.6.2 Signal conditions

Any signal conditions that are associated with microphones actually consist generally of three main parts that the following block diagram explains:

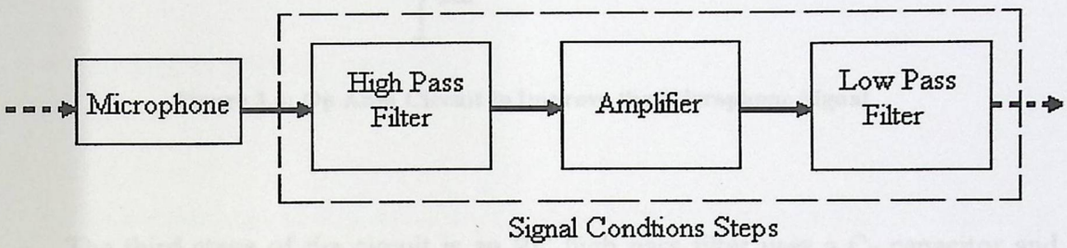


Figure 3.3: Signal Conditions Block Diagram

The three stages for the microphone analog circuit are shown in more details in figure 3.3. It will need high pass filter, an amplifier and a low pass filter. The first stage, an RC high pass filter uses a C_1 capacitor and R_1 resistor. The cutoff frequency is $1/(2\pi*R_1*C_1)$, which represent the lower limit of human spectrum (near 100 Hz). This will also cut off the 60Hz surrounding noise. The next stage of the circuit is the amplifier. The microphone has a built in amplifier but after checking through the oscilloscope, the signal strength could be pretty low. The design of signal amplifier signal to improve the peak to peak audio signal will be necessary. The op. Amps must be in negative feed back loops and are inverting and thereby have a gain of $-R_f/R_{in}$ as show in figure 3.4.

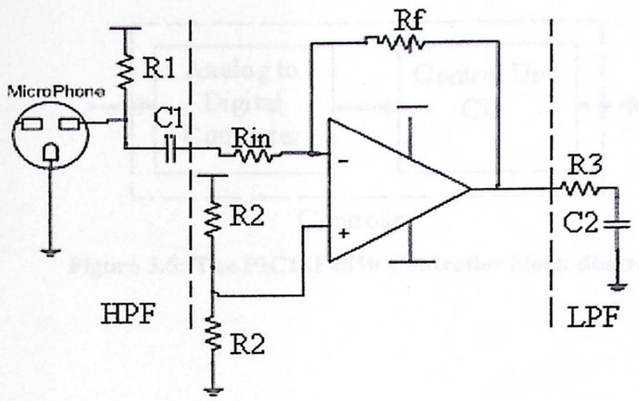


Figure 3.4: Op Amp Circuit to Improve the Microphone Signal.

The third stage of the circuit is an RC high pass filter uses a C_2 capacitor and R_3 resistor. The cutoff frequency is $1/(2\pi*R_3*C_2)$, which represents the upper limit of human spectrum (near 3000 Hz).

The system will choose the appropriate values in the three stages that help us to enter the final and suitable signal to the next steps, *see more details in chapter 4.*

3.6.3 The PIC18F4550 Controller

There are two main parts in this step of the block diagram: analog to digital converter and the other controlling unit. The reason for separate ADC from microcontroller during explanation, refers to its importance in our project, see Figure 3.5.

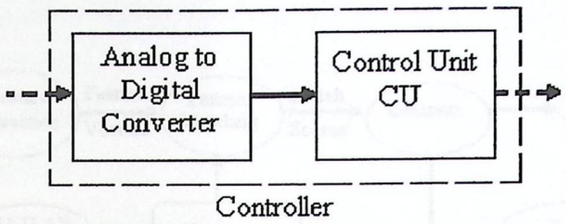


Figure 3.5: The PIC18F4550 Controller block diagram

- **Analog to Digital Converter**

The conditioned analog signal that comes from the previous step is sampled to form a digital signal by an analog-to digital (A/D) converter. Today's A/D converters for speech applications typically sample with 10 to 16 bits of resolution at 8000 to 20000 samples per second.

Over sampling is commonly used to allow a simpler analog anti aliasing filter and to control the fidelity of the sampled signal precisely. In local speaker-verification applications, the analog channel is simply the microphone, its cable, and analog signal conditioning. Thus, the resulting digital signal can be very high quality.

- **Control Unit**

The control unit step starting indicates the software design beginning. Many of steps and algorithms will be used in order to achieve our goal, the following block diagram (Figure 3.6) show the software stages.

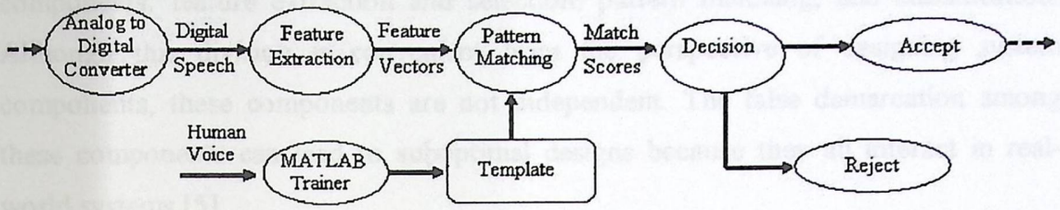


Figure 3.6: The Control Unit block diagram (software design block diagram).

This approach consists of four steps: feature extraction, pattern matching, making an accept/reject decision, and template. A block diagram of this procedure is shown in Figure 3.6. Feature extraction maps each interval of speech to a multidimensional feature space. (A speech interval typically spans 10 to 30 ms of the speech waveform and is referred to as a frame of speech.) This sequence of feature vectors $\mathbf{x}_i$ is then compared to template by pattern matching. This results in a match score z_i for each vector or sequence of vectors. The match score measures the similarity of the computed input feature vectors to models of the claimed speaker or feature vector patterns for the claimed speaker. finally, a decision is made to either accept or reject the claimant according to the match score or sequence of match scores, which is a hypothesis-testing problem.

1) Feature Extraction

The speech signal can be represented by a sequence of feature vectors. The selection of appropriate features and methods to estimate (extract or measure) them are known as feature extraction.

Traditionally, pattern-recognition paradigms are divided into three components: feature extraction and selection, pattern matching, and classification. Although this division is convenient from the perspective of designing system components, these components are not independent. The false demarcation among these components can lead to suboptimal designs because they all interact in real-world systems.[5]

Since one of the project goal is to design a system that minimizes the probability of verification errors. Thus, the underlying objective is to discriminate between the given speaker and all others.

2) Pattern Matching and Template

The pattern-matching task involves computing a match score, which is a measure of the similarity between the input feature vectors and some model. Template models are constructed from the features extracted from the speech signal. To enroll users into the system, a model of the voice, based on the extracted features, is generated and stored (on the EEPROM). Then, to authenticate a user, the matching algorithm compares/scores the incoming speech signal with the model of the claimed user.

3) Decision

Having computed a match score between the input speech-feature vector and a model of the claimed speaker's voice, a verification decision is made whether to

accept or reject the speaker or request another utterance. If a system rejects an impostor, it doesn't move the car. If the system accepts a valid user, it makes a car move according to his command.

3.6.4 Toy Car and Navigate Gear

The toy car turns start when the decision block for specific spoken word are toggled to an accept state, this makesthe controlling unit send the word that matched as electrical signal to the proper motors. See Figure 3.7

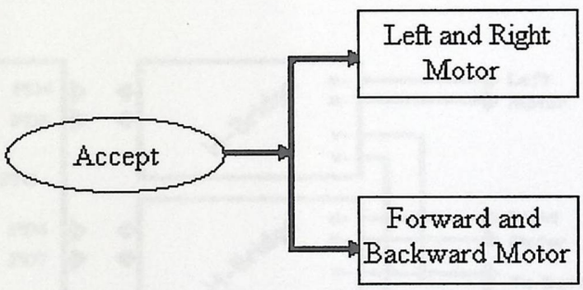


Figure 3.7: Toy Car and Navigate Gear block diagram

The main car motors probabilities are: Forward, backward, left, right, stop. The combinations between these commands are acceptable within certain conditions.

- **DC-Motors and H-bridges**

In order to establish proper motor control for the car, the use of an H-bridge circuit is required.

The H-bridge circuit consists of a set of four transistors that are arranged in an "H" orientation. This layout allows for current to flow bi-directionally through the circuit thus allow controlling of the car direction additionally; logic inputs signals can be used to determine which direction the motors are spinning. Depending on the paired combination of logic 1's and 0's the motor shaft can turn left, turn right, and brake. The H-bridges will act as interfaces between microcontroller and the motors.

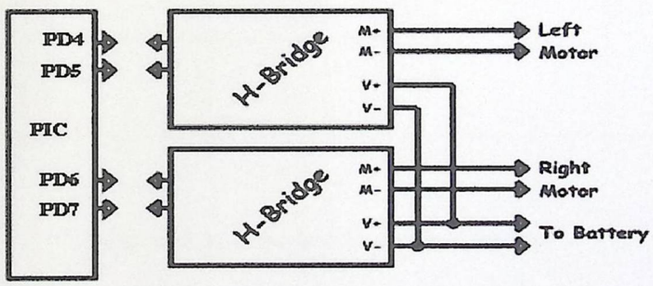


Figure 3.8 PIC with H-bridge and Motors

3.7 Project Interaction with Surrounding Environment

The user interacts with system through microphone that speak to word, then the analog signal convert into digital signal using the A\D converter that in the PIC.

Software in the PIC is performs the word recognition step, if word is matched send appropriate action to the car. Otherwise, the system returns to get another sample from capturing stage.

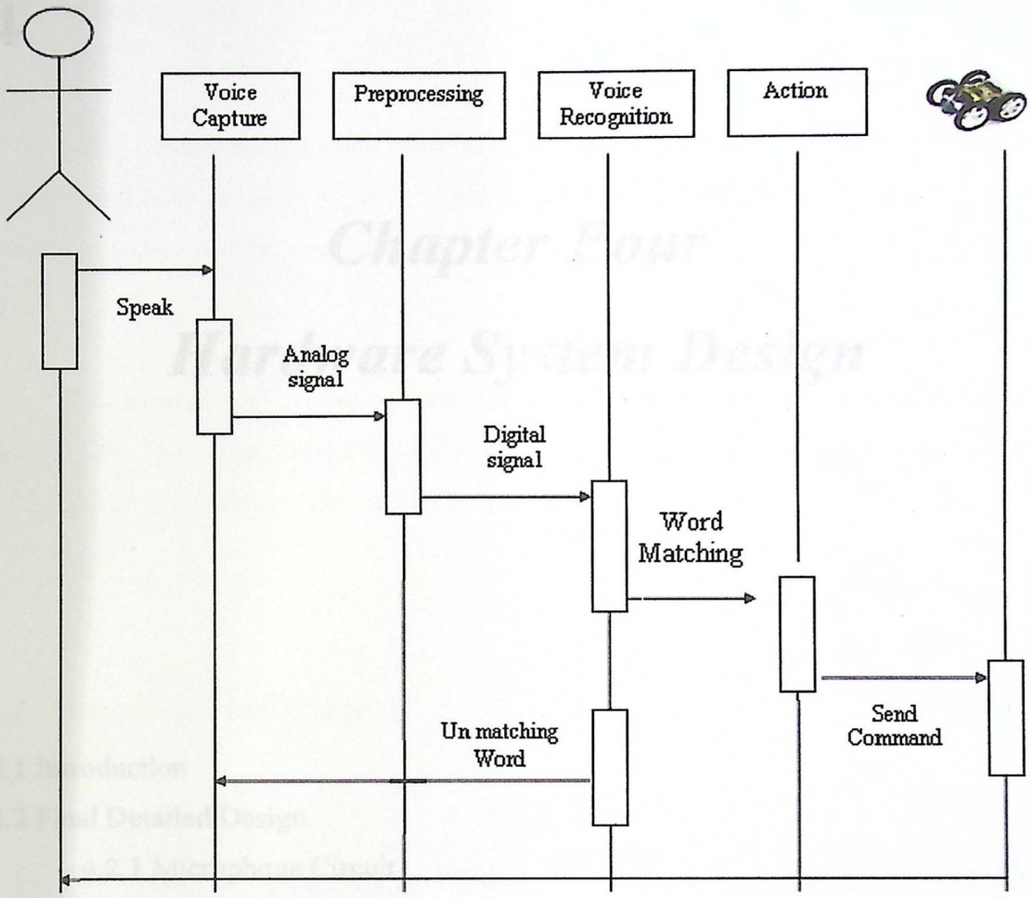


Figure 3.9: Sequence Diagram of System

4

Chapter Four

Hardware System Design

4.1 Introduction

After explaining the theoretical background, the general block diagram of the system, and how the system works, there is a need to view what is the design of this system in more specific, powerful and more formal terms.

This chapter will explain in details the final detailed design, and final schematic diagram for the system and subsystem schematic diagrams .Also besides to showing all the control signals and the interface circuits in the system.

4.2 Final Detailed Design

The system consists of the following 5 main modules that assembled on a single board as shown in figure 4.1:

- Microphone Circuit.
- PIC18F4550 Circuit.
- H-bridge Circuit.
- Car Motors and Power.
- PIC18F4550 Programmer Circuit

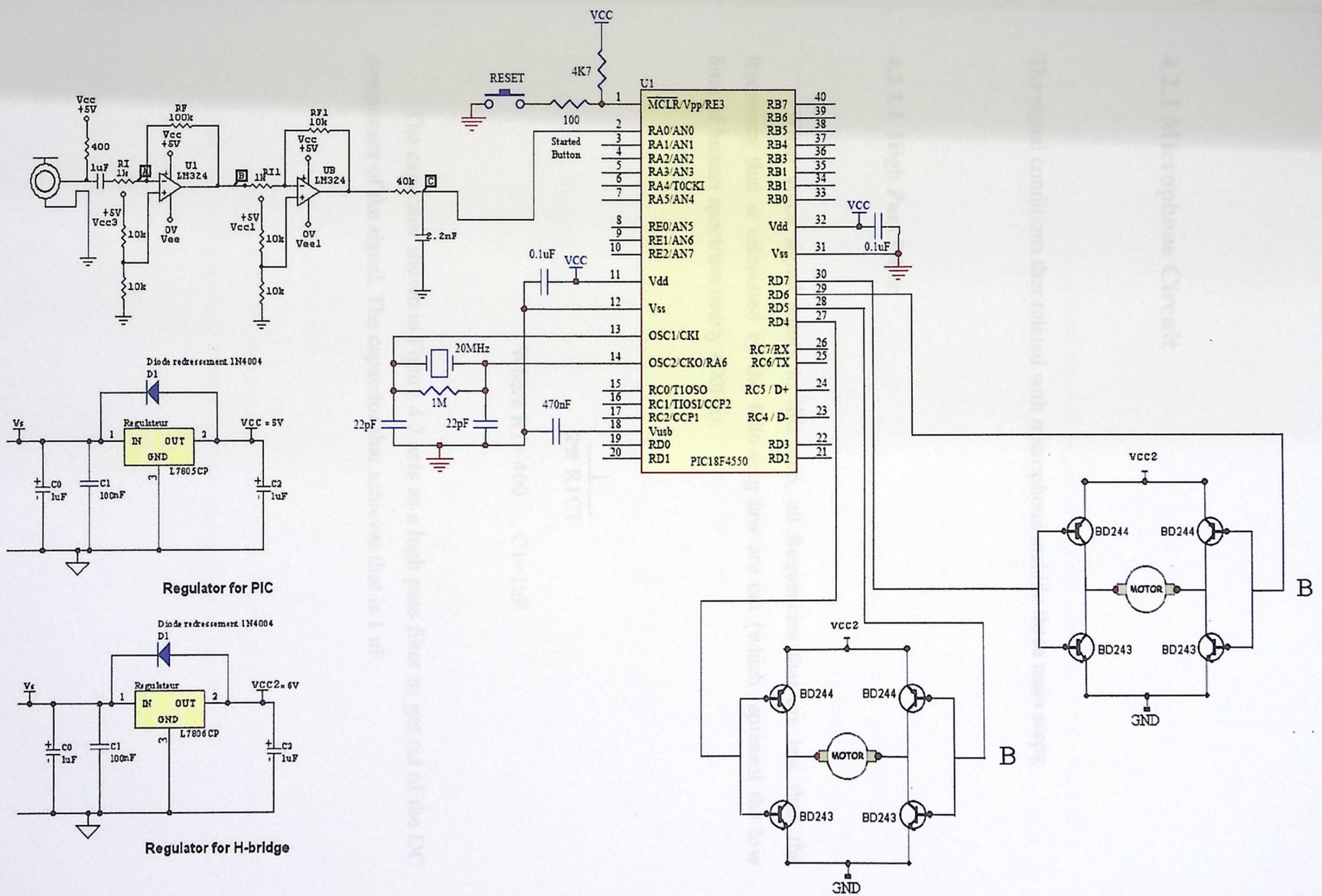


Figure 4.1 Schematic of the System

4.2.1 Microphone Circuit

The signal conditions that related with microphone include three main steps:

4.2.1.1 High Pass Filter

When voice signal enters to this step, all frequencies that are less than the frequency that is calculated by the following law are cut (which represent the low limit of human spectrum nearly 400Hz).

$$\frac{1}{2\pi R1C1}$$

Where R1= 400 , C1=1uF

The capacitor shown in Figure 4.2 acts as a high pass filter to get rid of the DC component of the signal. The capacitor that achieves that is 1 uF.

4.2.1.2 Amplifier

When the high pass filter finished, signals becomes weakness so we need to enlarge the signal using OP amplifier which increases the peak to peak of the audio signal.

Amplifier circuits were used to move the microphone output to an amplitude of approximately 5 volts. Inverting amplifier circuit included two resistors, R1 and R2, wired in series, with a LM324 op-amp wired in parallel across R2. This circuit provides a gain of R2/R1. We implemented amplifier with gains of 1000, where R1 = 1k, R2 = 100k, R3=1k and R4=10k.as shown in the figure 4.2.

4.2.1.3 Low Pass Filter

The final step of Microphone circuit is low passing filter .In this step; all frequencies more than 1800Hz are cut which represent the upper limit of human spectrum.

To prevent aliasing, a low-pass filter was implemented by grounding one of the amplification circuits through a capacitor and a resistor as in Figure 4.2.

Where R1=40K, C1=2.2nF

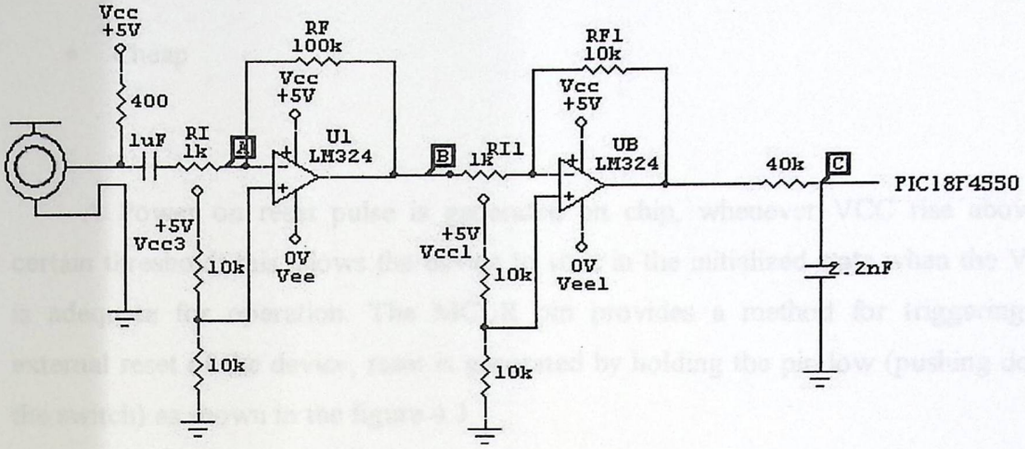


Figure 4.2 Microphone Circuit

4.2.2 The Microcontroller (PIC18F4550)

Choosing this type of microprocessor was for the following reasons:

- 32768(bytes) Program memory. Which represent enough memory to perform the project.
- 13 input channels 10 Bit Analog-to-Digital Module that convert analog signal to digital signal.
- Availability of PIC programmer in the university.

- Availability of MPLAB that used to programming PIC18F4550.
- Cheap

A Power on reset pulse is generated on chip, whenever VCC rise above a certain threshold; this allows the device to start in the initialized state when the VCC is adequate for operation. The MCLR pin provides a method for triggering an external reset of the device, reset is generated by holding the pin low (pushing down the switch) as shown in the figure 4.3.

A 20MHZ crystal is connected to the OSC1 &OSC2 pins to establish oscillation as shown in the figure 4.3. Capacitor values required to produce acceptable oscillator operation. Higher capacitance increases the stability of oscillator, but also increases the start-up time. The PIC18F4550 device includes an internal oscillator which generates two different clock signals; either can be used as the microcontroller's clock source. If the USB peripheral is not used, the internal oscillator may eliminate the need for external oscillator circuits on the OSC1 and/or OSC2 pins.

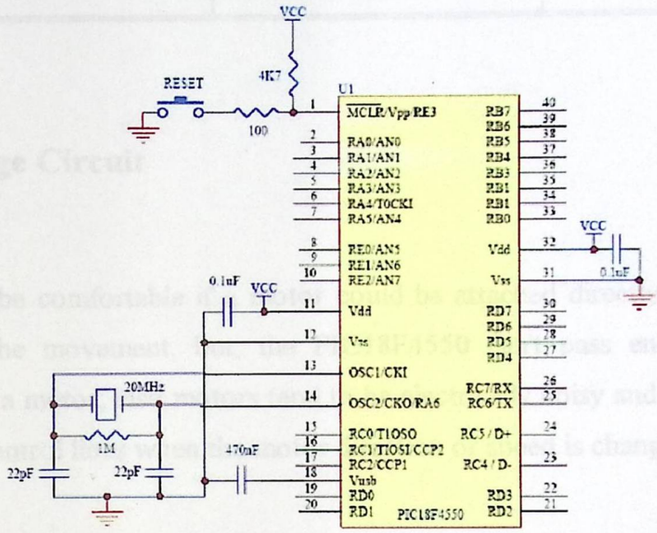


Figure4.3 PIC18F4550 with Oscillator and Reset

The mapping between pins on the PIC18F4550 controller and the desired modules is outlined in Table 4-1 below.

Table 4-1: Pins Configuration

SPECIFIC CONNECTION	PIC18F4520 PIN	PIN CONFIGURATION
Direction of Left Motor	RE0	Output
Direction of Right Motor	RE1	Output
Amplified Voice Signal	RA0	Input
External Oscillator	OSC1/OSC2	Input
Voltage Alimentation	VDD/VSS	Power

USB Connector	SD0/SDA/SCL	Input
Reset Circuit	MCLR	

4.2.3 H-bridge Circuit

It would be comfortable if a motor could be attached directly to PIC18F4550 that controls the movement, but, the PIC18F4550 can't pass enough current or voltage to spin a motor; also motors tend to be electrically noisy and can bring power back into the control lines when the motor direction or speed is changed.

A very popular circuit for driving DC motors is called H-bridge. The great ability of H-bridge circuit is that the motor can be driven forward or backward. Also it connects the PIC18F4550 with car motors.

H-bridge can be implemented with various kinds of components (common bipolar transistors, FET transistors, MOSFET transistors, power MOSFETs, or even chips).but in this system we used common power transistors.

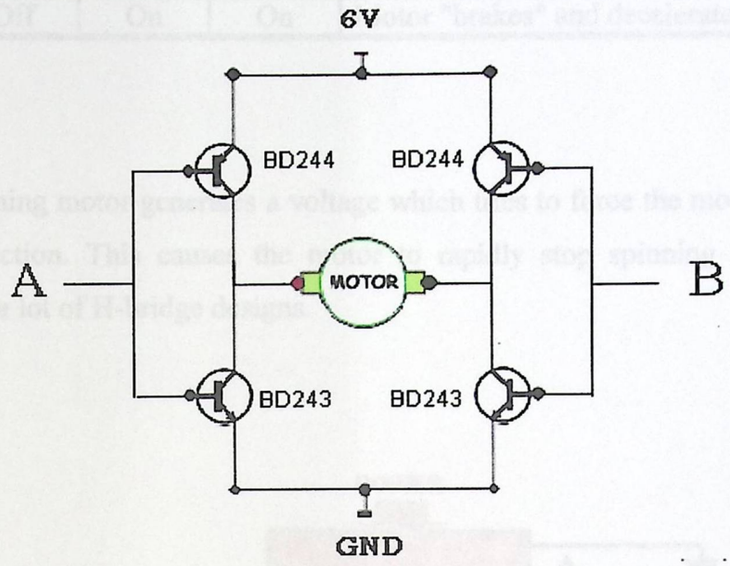


Figure 4.4 H-bridge Circuit

The control inputs A and B determine the flow of current in the circuit. If Control input A is HIGH and Control input B is LOW, the path of the current will be "Counterclockwise". But if Control input A is LOW and Control input B is HIGH then the path of the current will be "clockwise" as shown in figure 4.4.

There are in fact only four useful states (the four quadrants) where the transistors are turned on. As shown in below table 4.2

Table 4.2: States of Transistors

High Side Left	High Side Right	Lower Left	Lower Right	Quadrant Description
On	Off	Off	On	Motor goes Clockwise
Off	On	On	Off	Motor goes Counter-clockwise

On	On	Off	Off	Motor "brakes" and decelerates
Off	Off	On	On	Motor "brakes" and decelerates

The turning motor generates a voltage which tries to force the motor to turn the opposite direction. This causes the motor to rapidly stop spinning and is called "braking" on a lot of H-bridge designs.

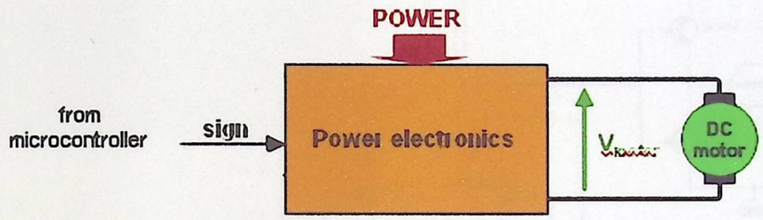


Figure 4.5 Block Diagram of H-bridge Circuit

The motor speed depends on the average voltage. The power electronic block contains an H-bridge designed for motion control applications. The signal sign controls the direction of the rotation as shown in figure 4.5.

Figure 4.6 shows the schematic diagram that demonstrates how the H-bridge relates to the PIC and car motors.

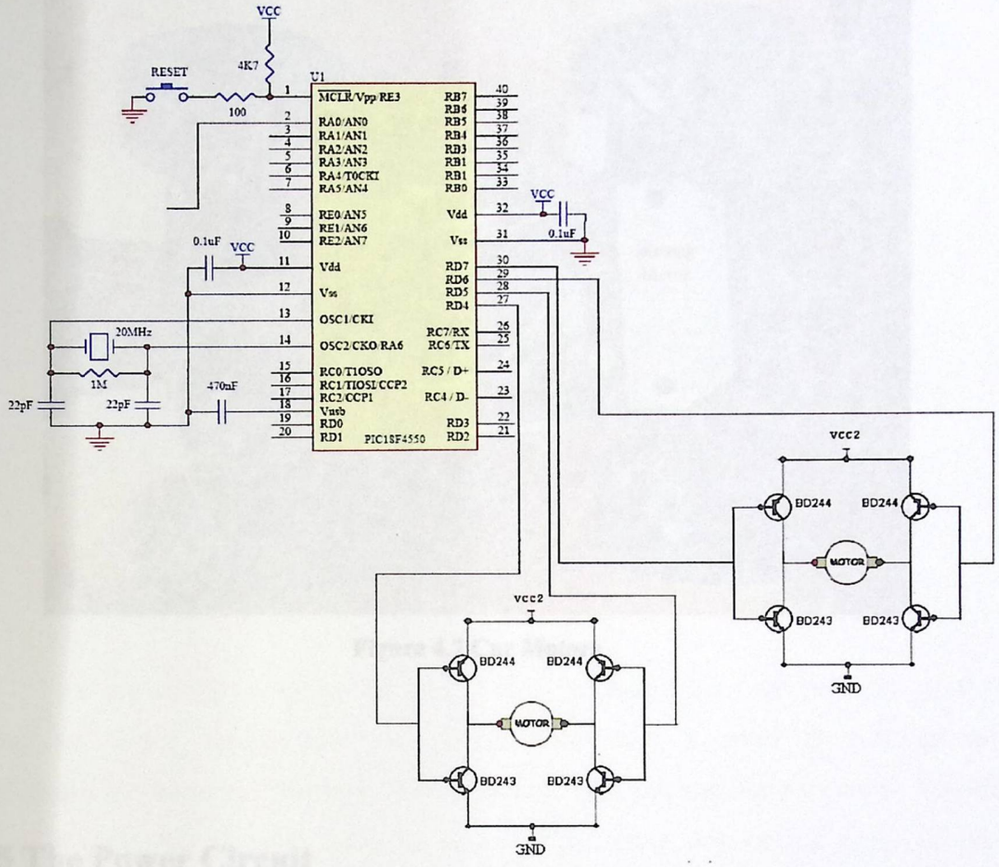


Figure 4.6 PIC18F4550 with the H-bridge

4.2.4 The Car

The system that is used in this car contains two motors, the first motor controls the movement of the car forward or backward and the second motor used to control the movement left or right as shown in figure 4.7.

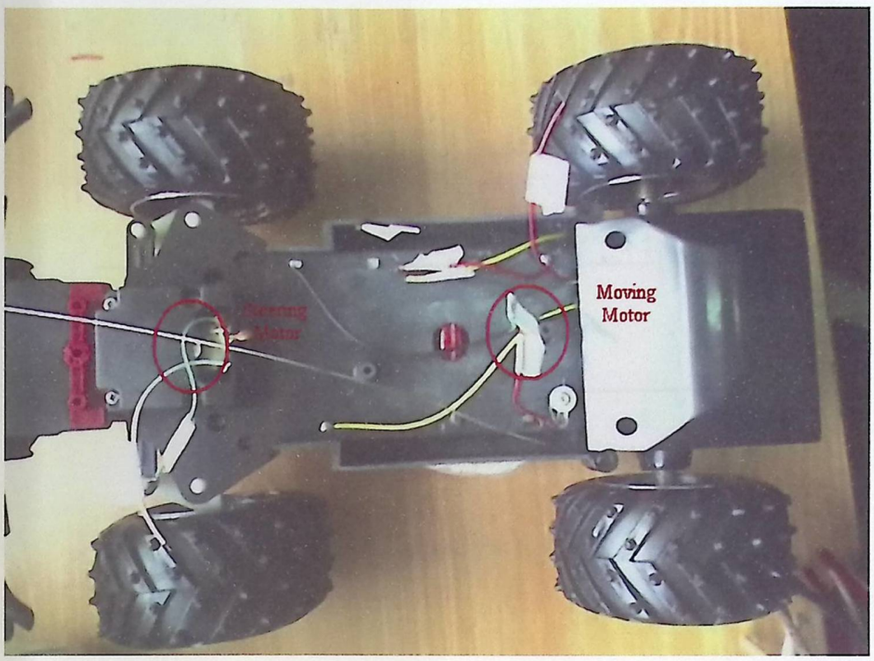


Figure 4.7 Car Motors

4.2.5 The Power Circuit

While analyzing the car composition we notice that the total voltage (9volt) is divided into two parts, 4.5volt for motor of movement (forward and backward), and 5.5 volt for steering motor as shown in figure 4.8.but this partitions of voltage is not efficient enough to this project ,so the system used 12V rechargeable battery.

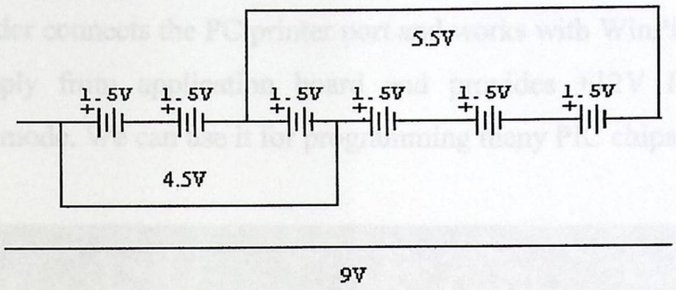


Figure 4.8 Batteries in the Car

The toy car is supplied by 12 V using rechargeable batteries, the system divide the power supply into two parts. First one is used to provide the microcontroller with 5volt through 7805 regulator, and the another part is provide the car motor with 6V through 7806 regulator. The connection of the batteries and regulator is shown in Figure (4-9) below.

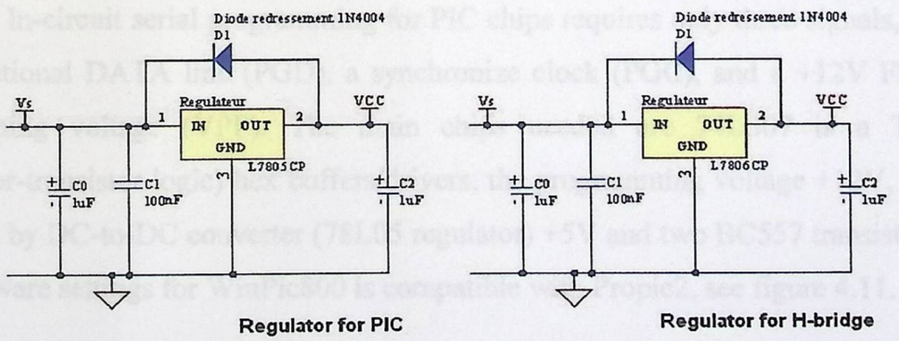


Figure 4.9: Regulated Voltage

4.2.6 PIC18F4550 Programmer Circuit

This loader connects the PC printer port and works with WinPic800. The loader gets DC supply from application board and provides +12V for high voltage programming mode. We can use it for programming many PIC chips easily.

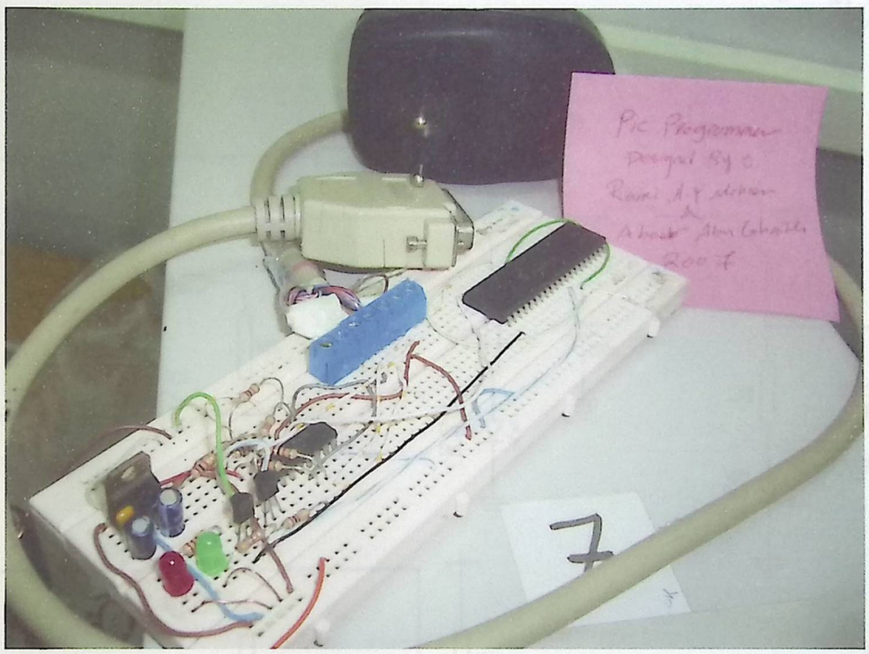


Figure 4.10 In-Circuit PIC Loader.

The In-circuit serial programming for PIC chips requires only three signals, i.e. a bi-directional DATA line (PGD), a synchronize clock (PGC), and a +12V Flash programming voltage (VPP). The main chips needed are 74LS07 is a TTL (Transistor-transistor logic) hex buffers/drivers, the programming voltage +12V, and generated by DC-to-DC converter (78L05 regulator) +5V and two BC557 transistors. The hardware settings for WinPic800 is compatible with Propic2, see figure 4.11.

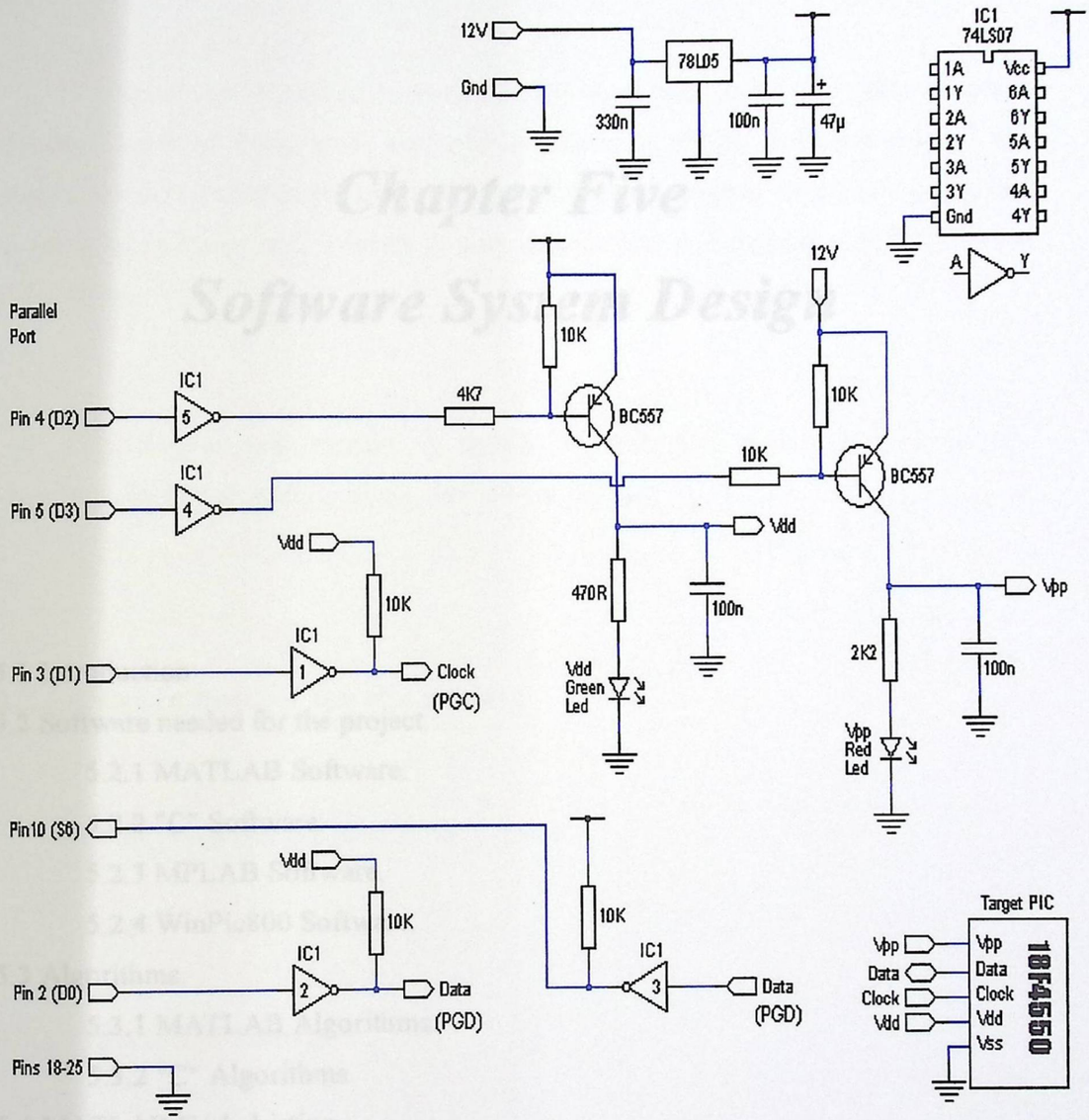


Figure 4.11 PIC Loader Schematic.

5

Chapter Five

Software System Design

5.1 Introduction

In chapter two of this documentation, the voice recognizer unit performs many of tasks. Most of these tasks are actually belonging to the software side of this project, the system needs intelligent software that accomplishes the project goals, and so many of software will mention in this chapter and will explain according to the order of use.

This chapter will explain in details all software needed in this project, algorithms that used, and packages that used in Matlab.

5.2 Software needed for the project

The software that are used in this project:

5.2.1 MATLAB Software

MATLAB is a very useful tool in order to understand the basic properties of discrete signals and digital filters. In MATLAB it is easy to make calculations, acquire to signals and plot them in both the time and frequency domain.

In this project MATLAB is used as most important software which the project begin, it has powerful abilities to analyze the voice including changes its properties, get its spectrum, and performs many types of filters on it.

5.2.2 "C" Software

The MPLAB software supports high level language like "C". This language will make the programming easier than using the assembly of PIC and getting more done in a shorter time with a higher level language. Besides, there is no need to worry about bank and page selection. The compiler holds your hand and relieves you of certain responsibilities.

In this project the "C" language is used as system software that help in write sophisticated algorithms related to this project such as voice recognizer algorithm, and template generation algorithm.

5.2.3 MPLAB Software

MPLAB IDE is a software program that runs on a PC to develop applications for Microchip microcontrollers. It is called an Integrated Development Environment, or IDE, because it provides a single integrated “environment” to develop code for embedded microcontrollers. MPLAB provides a comprehensive editor, project

manager and design desktop for application development of embedded designs using Microchip PICmicro and dsPIC microcontrollers.

The compiler of MPLAB should generate a **HEX** file and a **COD** file if the program builds successfully. The COD file contains all the information necessary for high-level debugging. The HEX file will be load in the programmer that builds in this project. Thus the mission of the MPLAB is finished.

5.2.4 WinPic800 Software

There is much software that support PIC programmer such as EpicWin, WinPic800, ProPIC18 etc. For each software can support different devices. For example EpicWin support PIC12F, PIC16F and some PIC18F and can be run on all windows. But, Winpic800 support PIC12F, PIC16F, PIC18F including dsPic (when setting up hardware as ProPIC2) and ProPIC18 support PIC18F only and run on all windows.

The WinPic800 supports PIC18F4550 controller that can load .HEX file in, program, verify, and erase. To use this project with winPic800 the hardware setting must be as the following picture.

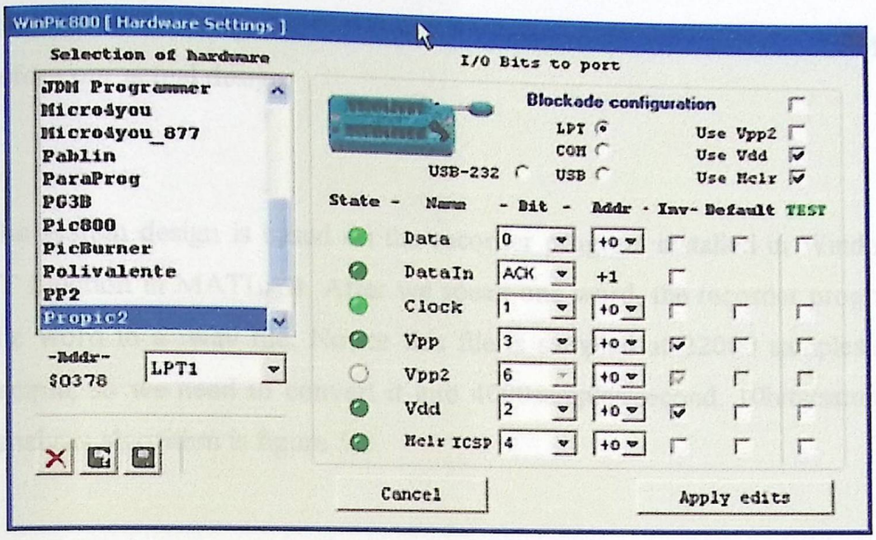


Figure 5.1 WinPic800 Hardware setting.

5.3 Algorithms

The project has two main stages of algorithms that help the system to recognize the spoken word perfectly:

5.3.1 MATLAB Algorithms

Generally the human speech spectrum is less than 2000Hz. according to Nyquist theory; the minimum sampling rate for speech should be 4000samples/second. Due to

the system is voice-recognition system; it is very helpful to analyze the speaker's voice before our actual design.

The system design is based on the recorder program installed in Windows XP and FFT function in MATLAB. After we speak one word, the recorder program will store the word in a .wav file. Notice this file is sampled at 22000 samples/second, 16bit/sample, so we need to convert it into 4000samples/second, 10bits/sample. The whole analysis algorithm is figure 5.2.

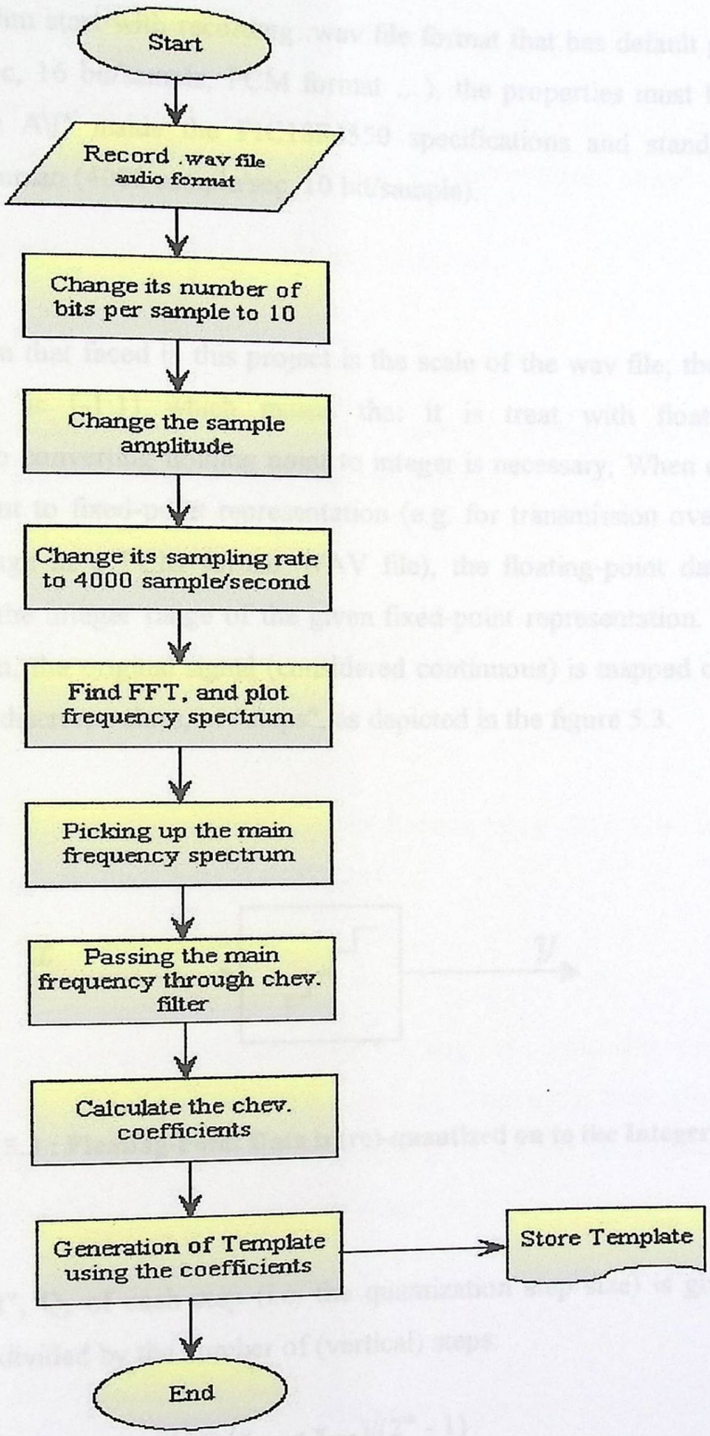


Figure 5.2 MATLAB Flowchart to generate template

This algorithm start with recording .wav file format that has default properties (22000 sample/sec, 16 bit/sample, PCM format ...), the properties must be change according to the A\D inside the PIC18F4550 specifications and standard voice spectrum of the human (4000 sample/sec, 10 bit/sample).

The problem that faced in this project is the scale of the wav file; the standard wav file format is [-1:1] which means that it is treat with floating point representation. So converting floating point to integer is necessary, When converting from floating-point to fixed-point representation (e.g. for transmission over a digital line, or for storage in a PCM format WAV file), the floating-point data is (re)-quantized on to the integer range of the given fixed-point representation. For linear PCM quantization, the original signal (considered continuous) is mapped on to a set of evenly-spaced discrete values, or "steps", as depicted in the figure 5.3.

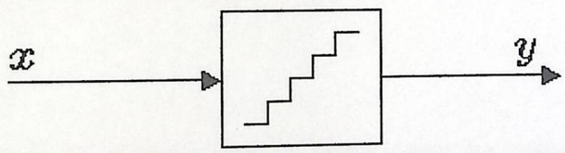


Figure 5.3 : Floating-Point Data is (re)-quantized on to the Integer

The "height", Q, of each step (i.e. the quantization step size) is given by the total input range divided by the number of (vertical) steps:

$$Q = (x_{max} - x_{min})/(2^{\omega} - 1)$$

With the symbols defined as follows:

ω: quantization bit depth (i.e. "valid bits" used in the WAV file).

Q: quantization step size, often called LSB since a change in input level of Q corresponds to a change in the LSB (Least Significant Bit) of binary coded output

$[x_{max} : x_{min}]$: amplitude range of data sent to the quantizer.

According to convention, digital audio samples are mapped over the floating-point range -1:1. Hence, the quantization step size becomes:

$$Q = 2/(2^{\omega}-1) = 1/(2^{\omega-1}-1/2)$$

In the project, 10-bit quantization (of floating-point data over the range -1:+1), will have a quantization step size of 1/511.5.

There are numerous approaches to performing the conversion from floating-point to integer (e.g. algorithms based on rounding, truncation, etc). However, the most natural method for creating a signed integer is to use the floor operation since this directly maps the data on to the required range:

$$Y_{int} = \text{floor}(x/Q)$$

With the symbols defined as follows:

x: (pre-quantized) floating-point sample (within nominal range:-1:+1)

Y_{int}: signed integer representation of the (quantized) sample

Whereby the nominal floating-point range maps on to the appropriate signed-integer range as follows:

$$[-1 : +1] \Rightarrow [-2^{\omega-1} : +2^{\omega-1}-1]$$

The range mapping for 10 bit quantization is given by:

$$[-1 : +1] \Rightarrow [-512 : +511]$$

Then ,the system will make shift of the scale by 512,so the range mapping for 10 bit become

$$[-1:+1] \Rightarrow [0 :1023]$$

Because of voltage reference of microphone is 2.5V that compensate 512.

The most important step is to find Fast Fourier Transform (FFT) which is an efficient algorithm to compute the discrete Fourier transform (DFT) and its inverse. The DFT is extremely important in the area of frequency (spectrum) analysis because it takes a discrete signal in the time domain and transforms that signal into its discrete frequency domain representation.

After finding the FFT of the spoken word become the following step which is drawing the frequency spectrum of the words in order to pick up the main frequency spectrum and ranges. Figure 5.4 and 5.5 shows the frequency spectrum for ("امام", Front) and ("قف", Stop) word respectively.

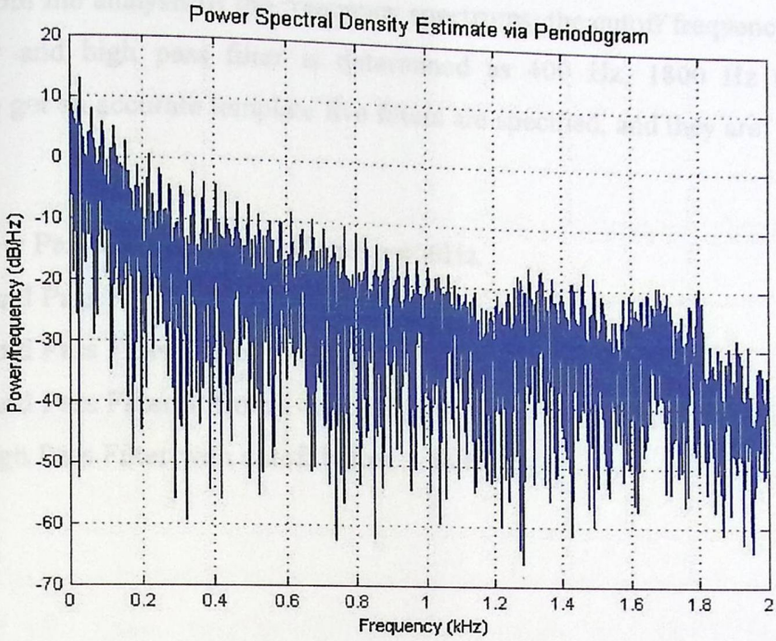


Figure 5.4 The frequency spectrum for "أمام"

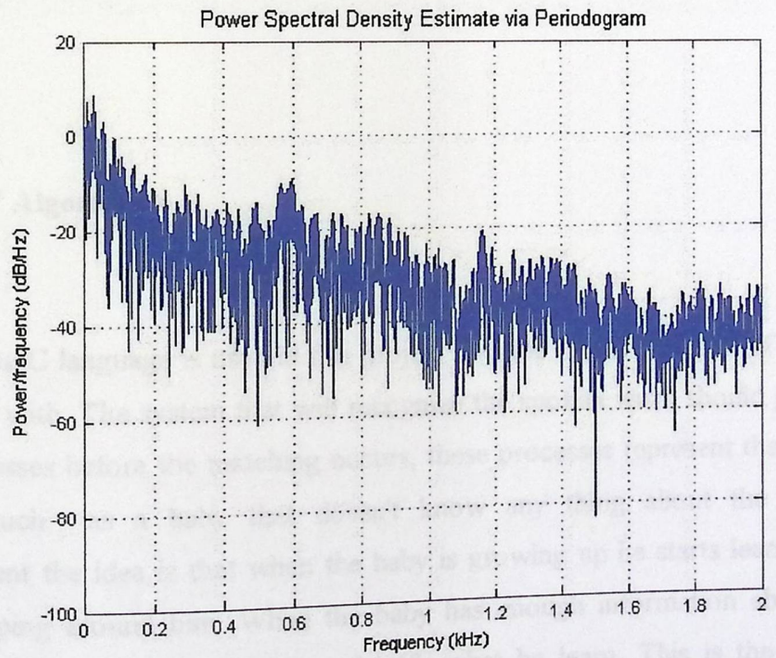


Figure 5.5 The frequency spectrum for "قف"

From the analysis of the frequency spectrums, the cutoff frequency of the low pass filter and high pass filter is determined as 400 Hz, 1800 Hz respectively. Besides to get an accurate template five filters are specified, and they are:

- Low Pass Filter with cutoff above 400Hz.
- Band Pass Filter 1 from 400Hz-900Hz.
- Band Pass Filter 2 from 900Hz-1400Hz.
- Band Pass Filter 3 from 1400Hz-1800Hz.
- High Pass Filter with cutoff below 1.8KHz.

Here's MATLAB complete its turn when it knows the frequency range for each filter and generate their coefficients, this project used ChebychevII filter to perform this job.

5.3.2 "C" Algorithms

The C language is used in this project because of the complier of the MPLAB is dealing with. The system that will recognize the spoken word should pass through two processes before the matching occurs, these processes represent the brain of the system; such as a baby that doesn't know any thing about the surrounding environment the idea is that when the baby is growing up he starts learning through what happing around him. When the baby has enough information about what he wants to know he will be able to classify what he learn. This is the idea of this

project; the toy car just like a baby that needs to learn the words that want to recognize, then develop its abilities to classify what it is learn.

The system according to this is separated into two processes that appear in the figure 5.6.

- **Training Process**

This process is starting by getting 4000 samples that generated and manipulated by MATLAB, these samples will passes through five filters that explained previously, the filters will reduce the number of digitized sample from 4000 to 40, after filter process done a number of times (in this project 20 times); we sum this trials to calculate the average of them before store them as final template.

- **Classification Process**

The classification process is similar some way to the training process, but there are some differences; one of the most differences is the correlation calculation process. From mathematics view, the correlation is to detect the linear relationship between two vectors. Suppose Y and X are two vectors, if $Y=aX+b$, where a and b are constants, we say Y and X are closely correlated.

After found the correlation number of the spoken word for each stored words in the template, then check the minimum correlation (which will be 5 correlation

numbers according to 5 words). The minimum doesn't necessary mean that the spoken word is matched correctly; so determine a specific threshold value for each stored word protect from these problems.

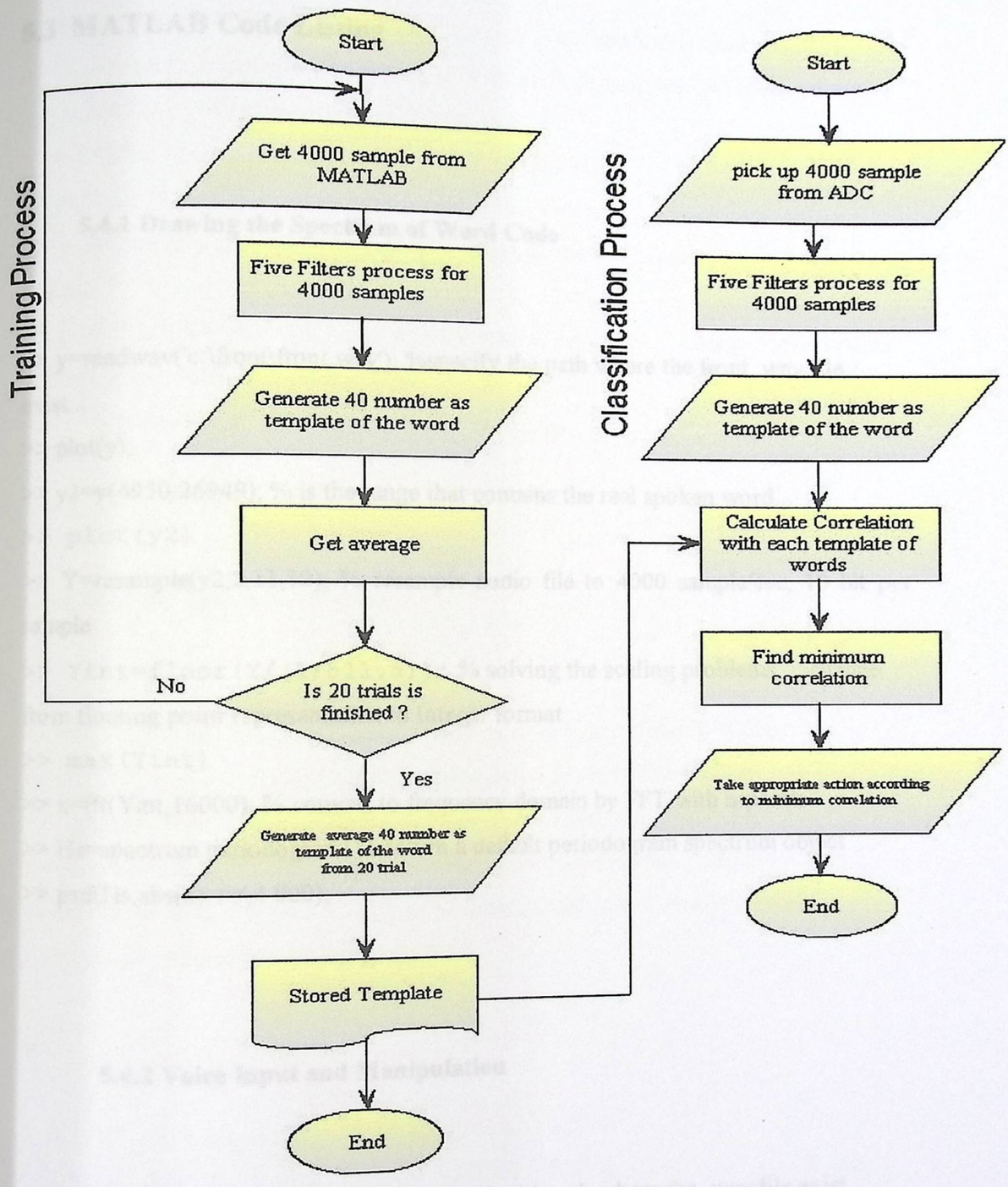


Figure 5.6 "C" Flowchart to recognize spoken word (Training process, Classification process)

5.3 MATLAB Code Listing

5.4.1 Drawing the Spectrum of Word Code

```
>> y=readwav('c:\front\front.wav'); %specify the path where the front .wav file exist...
>> plot(y);
>> y2=y(4950:26949); % is the range that contains the real spoken word...
>> plot(y2)
>> Y=resample(y2,2,11,10); % resample audio file to 4000 sample/sec, 10 bit per sample
>> Yint=floor(Y/(1/511.5)); % solving the scaling problems ... change from floating point representation to Integer format
>> max(Yint)
>> x=fft(Yint,16000); % convert to frequency domain by FFT with n-points
>> Hs=spectrum.periodogram; % return a default periodogram spectrum object
>> psd(Hs,abs(x),'Fs',4 000);
```

5.4.2 Voice Input and Manipulation

```
>> y=readwav('c:\front\front.wav'); %specify the path where the .wav file exist...
>> plot(y) % plot the signal to determine the spoken word range...
>> y2=y(4950:26949); % is the range that contains the real spoken word...
```

>> % y2 contains 22000 samples ... this number is not randomly come...
>> % we need y2 contains 22000 sample because front.wav has 22KHz sample rate.
>> % 8KHz --> will give us --> 8000sample/sec
>> %22KHz --> will give us --> ?? = 22000 sample/sec
>> plot(y2) % plot y2 to ensure from your signal crops...
>> Y=resample(y2,2,11,10); % resample audio file to 4000 sample/sec, 10 bit per sample
>> Yint=floor(Y/(1/511.5)); % solving the scaling problems ... change from floating point representation to Integer format
>> .... % store Yint to a file to pass to "C" code step...

5.4.3 Find the Filters Coefficients

>> [B1,A1]=cheby2(4,40,0.2,'Low'); %lowpass with cutoff above 400Hz
>> [B2,A2]=cheby2(2,20,[0.2 0.45]); %Bandpass from 400Hz-900Hz
>> [B3,A3]=cheby2(2,20,[0.45 0.7]); %Bandpass from 900Hz-1.4KHz
>> [B4,A4]=cheby2(2,20,[0.7 0.9]); %Bandpass from 1.4KHz-1.8KHz
>> [B5,A5]=cheby2(4,20,0.9,'High'); %Highpass with cutoff below 1.8KHz.
>> [sos1,g1]=tf2sos(B1,A1,'up','inf')

sos1 =

0.0511	-0.0166	0.0511	1.0000	-1.4212	0.5181
2.5183	-3.9278	2.5183	1.0000	-1.7096	0.8046

g1 =

```
    0.0970

>> [sos2,g2]=tf2sos(B2,A2,'up','inf')

sos2 =

    0.2228    0.0090    0.2228    1.0000   -0.7780    0.7647
    2.2293   -3.8444    2.2293    1.0000   -1.2046    0.8011


g2 =

    0.2108

>> [sos3,g3]=tf2sos(B3,A3,'up','inf')

sos3 =

    0.3251   -0.1984    0.3251    1.0000    0.2044    0.7757
    1.4460    1.9706    1.4460    1.0000    0.6814    0.7896


g3 =

    0.2227

>> [sos4,g4]=tf2sos(B4,A4,'up','inf')

sos4 =
```

```
    0.1272    0.1113    0.1272    1.0000    1.4059    0.7969
    6.2540   12.0986    6.2540    1.0000    1.6649    0.8525


g4 =

    0.1252

>> [sos5,g5]=tf2sos(B5,A4,'up','inf')

sos5 =

    0.3047    0.4312    0.3047    1.0000    1.4059    0.7969
    2.2075    4.1629    2.2075    1.0000    1.6649    0.8525


g5 =

    0.1252
```

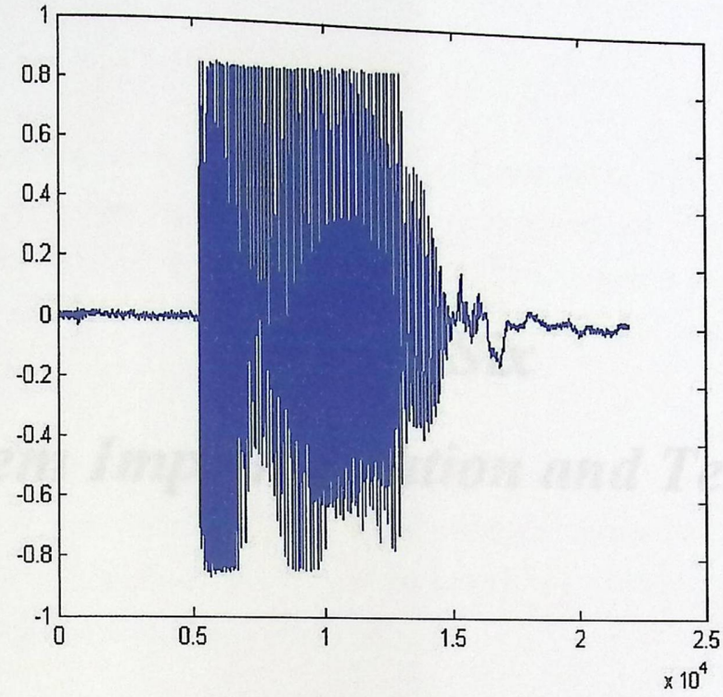


Figure5.7 "أمام" command before manipulation process

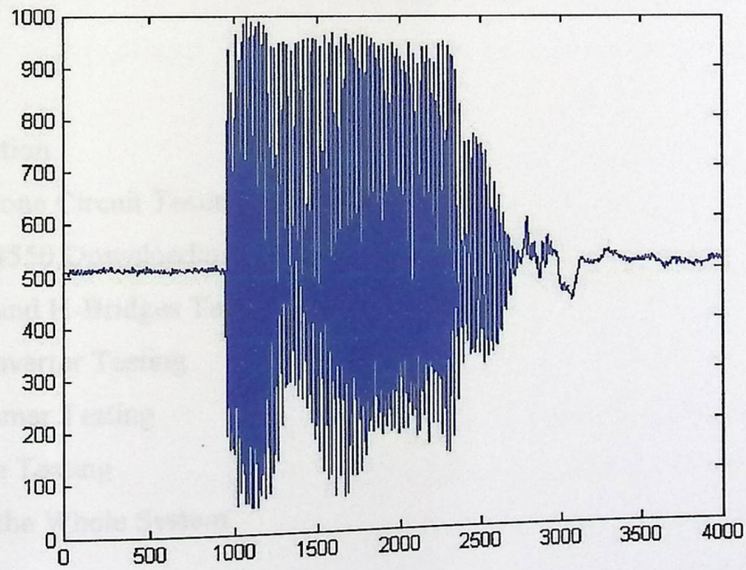


Figure 5.8"أمام" command after manipulation process

6

Chapter Six

System Implementation and Testing

6.1 Introduction.

6.2 Microphone Circuit Testing.

6.3 PIC18F4550 Downloading Programs Testing.

6.4 Motors and H-Bridges Testing.

6.5 A/D Converter Testing.

6.6 Programmer Testing

6.7 Software Testing

6.8 Testing the Whole System.

6.1 Introduction

In this chapter methods and ways which we follow will be explained in a check process the subsystem separately and check system completely after gathering to make sure that required aims can be achieved .Testing process is one of the main steps in carrying out this project.

There are different kinds of testing which were carried out in this project (e.g.) testing for HW part .There is another testing which was carried out at the level of SW, but there are specific circuits its check include the HW part and the SW together like the H-bridge circuit and the PIC circuit.

The implementation process is done synchronized with the testing operation, because each implementation phase will take many times to ensure that are no errors.

6.2 The Microphone Circuit Testing

Because there are two stages, the total gain is 1000 .this is because the input voice is around 0-50mV.Therefore a 1000 gain is necessary. In this circuit the resulted signal was checked (voltage of signal) if it was suitable and magnified enough to be inserted into the PIC, during the analysis of the signal through the oscilloscope to see the resulted signal, the peak to peak for this signal is amplified around 5V. As shown in figure 6.1

In this stage of testing, the LM741 has large offset that will affect on the signal so, the replacement of this chip with LM342 chip that has 4 op-amplifiers and auto offset is better choice in this project.

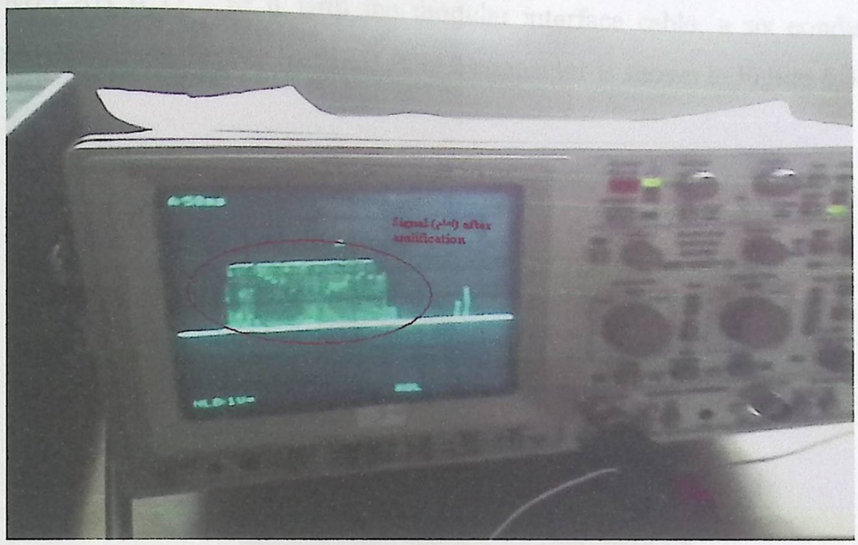


Figure 6.1 signal for forward command

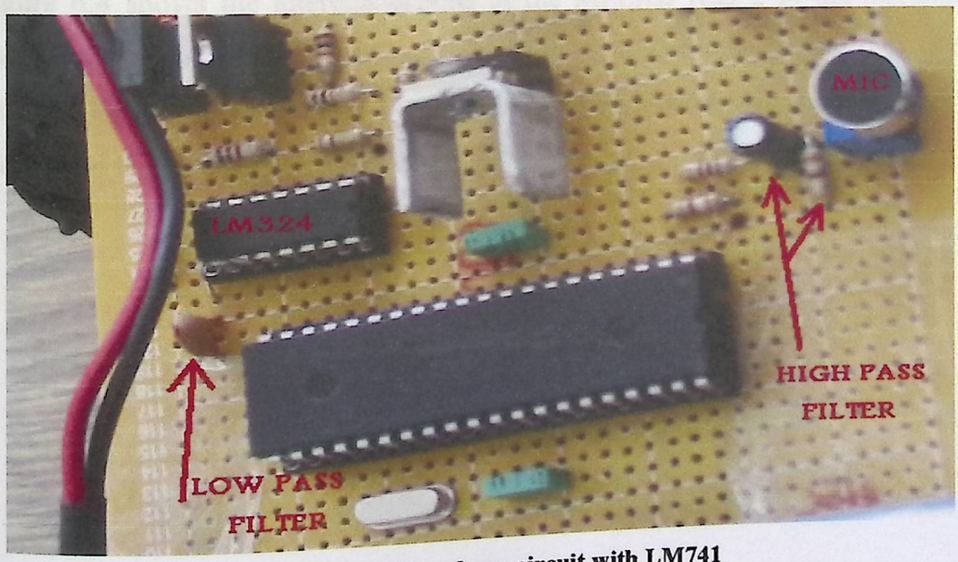


Figure 6.2 Microphone circuit with LM741

6.3 Testing PIC18F4550 Downloading Programs

Testing the ability to download a program on the PIC18F4550 is done by connecting MPLAB ICD 2 to it with the modular interface cable, a six conductor cable. The pin numbering for the MPLAB ICD 2 connector is shown in Figure 6.3.

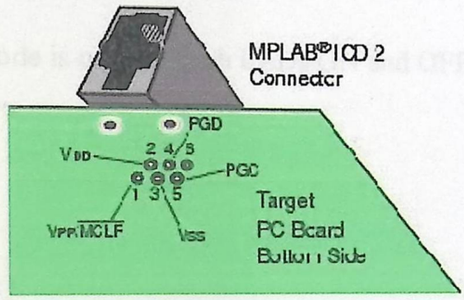


Figure 6.3: Pin Numbering for Modular Connector

Figure 6.4 shows the interconnections of the MPLAB ICD 2 to the modular connector on the target board. There are six pins on the ICD connector, but only five are used. The figure 6.4 also shows the wiring from the connector to the PIC micro MCU device on the target PC board.

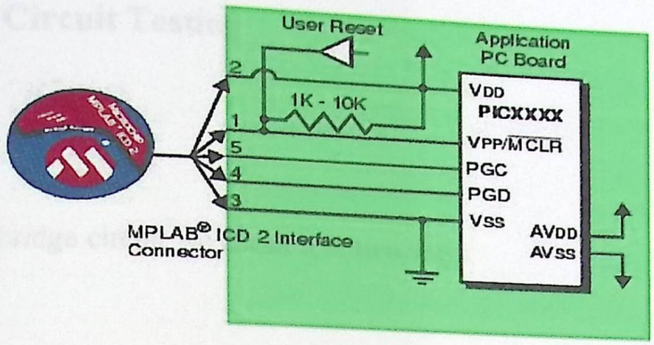


Figure 6.4: MPLAB ICD2 Connections to (figure 6.3) Target Board

The following testing code is used to flash LEDs ON and OFF on the output ports (portA):

```
#include<p18f4550.h>

Void main (void)
{
int a;
TRISA=0;

        While (1)
        {
                For (a=0;a<100;++a)
                  PORTA=1;

                For (a=0;a<100;++a)
            PORTA=0;
        }
}
```

6.4 H-Bridge Circuit Testing

To check the H-bridge circuit we check it in two ways:

6.4.1 (555) Circuit with H-bridge Chip

This test is based on establishing 555 circuit (Figure 6.5), which generate square wave which represent PWM signal that generated from the PIC18F4550 as shown in figure6.5, this signal will be the input of H-bridge chip (MAX4427) to control the car's speed. To control the car's direction, the pin4 for the H-bridge do this job .When its state VCC, it move back, however when it's grounded, it move forward .As at the figure6.7.

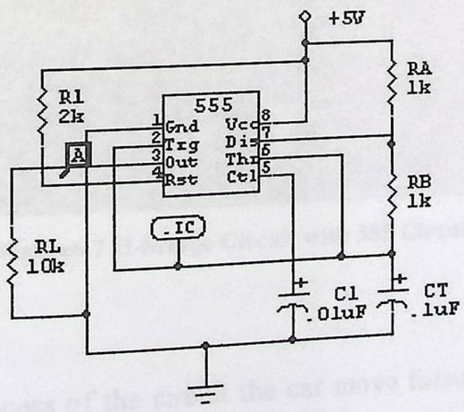


Figure 6.5: 555Ccircuit

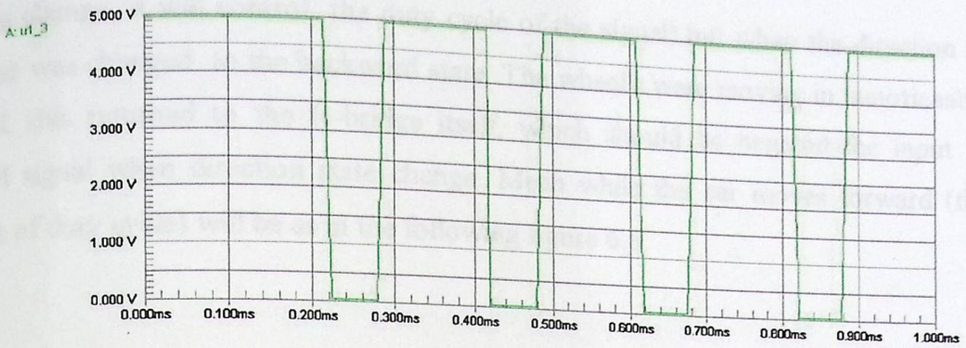


Figure 6.6:The Output of 555 Circuit

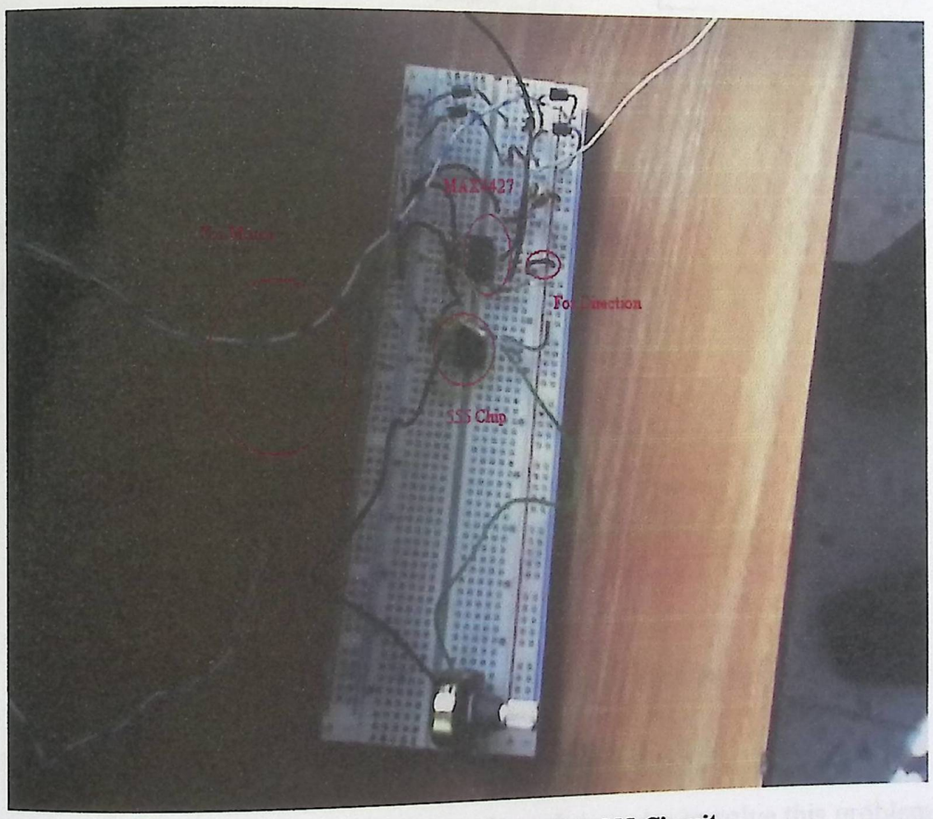


Figure6.7 H-bridge Circuit with 555 Circuit

In the checking process of the circuit the car move forward and the control of its speed is applied through the change potentiometer values in 555 circuit (when its

values change, it will control the duty cycle of the signal).but when the direction of the car was changed to the backward state. The wheel's were moving in unnoticeable speed, this returned to the H-bridge itself, which should be negated the input of PWM signal when direction state change. Mean while the car moves forward (the width of duty cycle) will be as in the following figure 6.8.

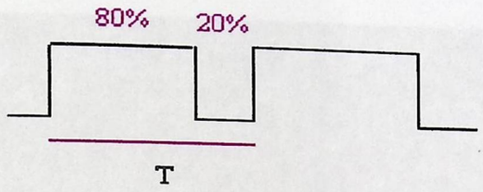


Figure 6.8 Forward State with un negated PWM Signal

And when the direction is reflected, the width of duty cycle becomes as in the following figure 6.9

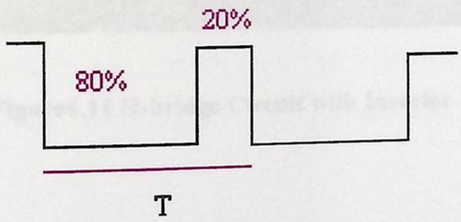


Figure 6.9 Backward State with un negated PWM Signal

This is not enough to move the car backward, in order to solve this problem we decided to make complement of duty cycle through putting inverter on the H-bridge entrance when it moves backward ,so the duty cycle becomes as in the following figure 6.10

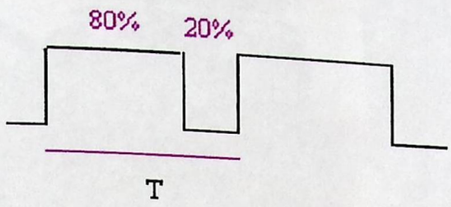


Figure 6.10 Backward States with negated PWM Signal

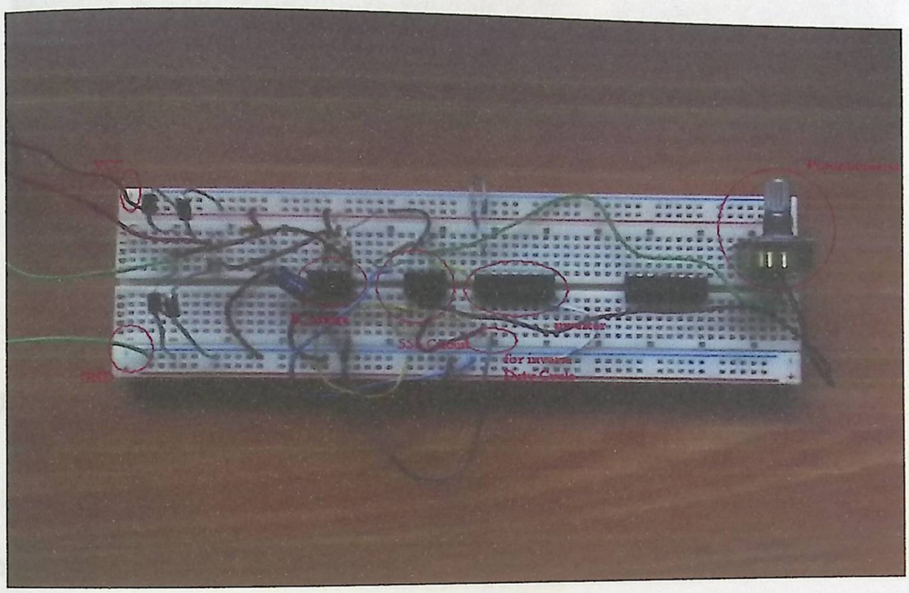


Figure6.11 H-bridge Circuit with Inverter

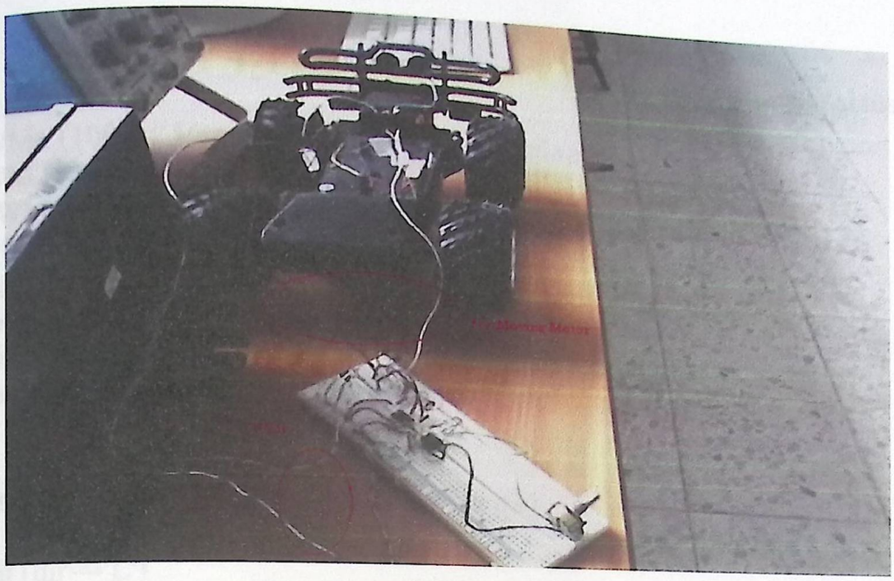


Figure6.12 H-bridge Circuit with Car

6.4.2 H-bridge Chip with PIC 18F4550

The second method in checking H-bridge is implemented by connecting them to the PIC as shown in figure (6.13) and applying the following code to motivate the motors backward, forward, left, right. We solve the complement problem by the SW.

```
#include <p18f4550.h>
#include <timers.h>
#include <pwm.h>
#include "gamelmotors.h"
#pragma config FOSC = INTOSC_HS
#pragma config WDT = OFF
#pragma config LVP = OFF
void motors_init(void)
{
        OpenTimer2 (                                    // Timer2 configuration
                TIMER_INT_OFF &                         // No interrupt
```

```
                T2_PS_1_1 &                     // Prescale 1:4
                T2_POST_1_1                     // Don't use postscaler actually
        );
OpenPWM1 (100); // Use of CCP1 as PWM module at ~20kHz
OpenPWM2 (100); // Use of CCP2 as PWM module at ~20kHz
SetDCPWM1 (0);                          // Set 0% duty cycle to PWM1
SetDCPWM2 (0);                          // Set 0% duty cycle to PWM2
TRISCbits.TRISC2 = 0;           // Make RC2 (left pwm) an output
TRISCbits.TRISC1 = 0;           // Make RC1 (right pwm) an output
TRISEbits.TRISE0 = 0;           // Make RE0 (right sign) an output
TRISEbits.TRISE1 = 0;           // Make RE1 (left sign) an output
}

void motors_steering(unsigned int speed, char dir)
{
        if (dir=='L')
        {
        PORTEbits.RE1 =0;
        SetDCPWM2 (speed);
        }
  If (dir=='R')
        {
        PORTEbits.RE1 =1;
        SetDCPWM2 (speed);
        }

}
void motors_moving(unsigned int speed,char dir)
{

        if (dir=='B' )
        {

        PORTEbits.RE0 =0;
        SetDCPWM1(speed);

        }
  if (dir=='F')
        {

        PORTEbits.RE0 =1;
        SetDCPWM1 (speed);

        }
```

```
}

Void main ()
{
int i;
motors_init();

TRISA=0x00;

while(1)
 {
        motors_moving(100,'B');
        motors_steering(90,'L');
  }
}
```

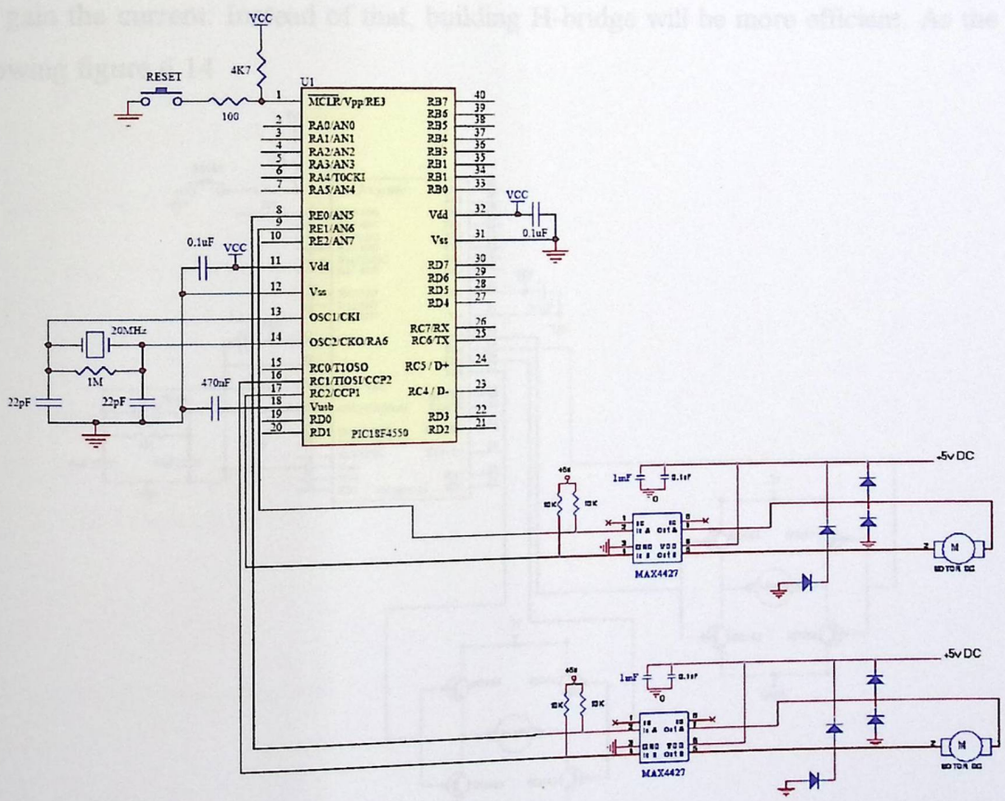


Figure 6.13: Motors and H-Bridges Testing

6.4.3 Building H-bridge Using Transistors

The car wheels were moving without any load during test1 and test2 process, when the car is putting on the ground, an important observation a raised that the car can't move. The reason of this problem was that the H-bridge chip didn't provide the car with suitable current so two alternative solutions exist one is based on making gain to the output current from H-bridge, another is build H-bridge through four transistors.

The first solution is not practical since we need to add another transistor that will gain the current. Instead of that, building H-bridge will be more efficient. As the following figure 6.14

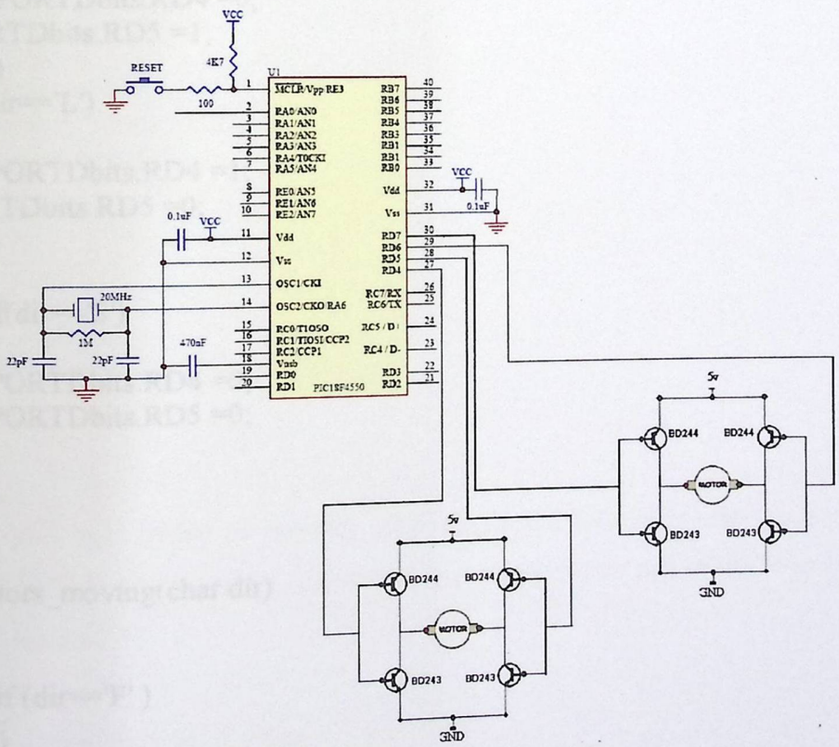


Figure 6.14 PIC18F4550 with H-bridge

The following code to motivate the motors backward, forward, left, right.

```
void motors_init(void)
{
        OpenTimer2(                                  // Timer2 configuration
                TIMER_INT_OFF &                      // No interrupt
                T2_PS_1_1 &                          // Prescale 1:1
                T2_POST_1_1                          // Don't use postscaler actually
        );
TRISDbits.TRISD4 = 0;              // Make RD4 an output
TRISDbits.TRISD5 = 0;              // Make RD5 an output
TRISDbits.TRISD6 = 0;              // Make RD6 an output
TRISDbits.TRISD7 = 0;              // Make RD7 an output
}

void motors_steering(char dir)
{
         if (dir=='R')
         {
         PORTDbits.RD4 =0;
     PORTDbits.RD5 =1;
         }
     if (dir=='L')
         {
         PORTDbits.RD4 =1;
     PORTDbits.RD5 =0;
         }

         if(dir=='S')
         {
         PORTDbits.RD4 =0;
         PORTDbits.RD5 =0;

         }

}
void motors_moving(char dir)
{

         if (dir=='F' )
         {
         PORTDbits.RD6 =0;
     PORTDbits.RD7 =1;
```

```
        }
        if (dir=='B')
        {
        PORTDbits.RD6 =1;
    PORTDbits.RD7 =0;
        }
        if(dir=='S')
        {
        PORTDbits.RD6 =0;
        PORTDbits.RD7 =0;

        }


}

void main()
{
int i;
motors_init();

while(1)
 {
        motors_moving('B');
        motors_steering('S');
    motors_moving('F');
}}
```

6.5 A/D Converter Testing

The ADC was tested by entering an analog input (potentiometer) entered at pin (RA0) of PIC18F4550, and by connecting the digital outputs at PORTB to LED's.As shown in figure 6.15.

Some of the test results were

1. When analog input was 5V, the digital output [11111111] was FFH.
2. When analog input was 2.5V, the digital output [10000000] was 80H.
3. When analog input was 0V, the digital output [00000000] was 00H.

PIC Reading Values were come from microphone circuit depending on the following rules:

- The Analog-to-Digital (A/D) converter module has 13 inputs that allow conversion of an analog input signal to a corresponding 10-bit digital number.
- 10-bit digital number can be read as integer number (N).
- N is converted to its corresponding voltage value using the following equation:

$$N = \frac{2^{10}}{(Vref+ - Vref-)} \quad Vin$$

Where:

$0x000 \leq N \leq 0x3ff \qquad 0 \leq N \leq 1023$

Vreff- = -2.5 Volt and Vref+ = 2.5 Volt.

precision :

$$\text{if } N=1 \quad dVin = \frac{Vref+ - Vref-}{1024}$$

The following code used to test if the A\D Converter in PIC18f4550 converts the analog signal to digital correctly.

```
#include <p18f4550.h>
#include <adc.h>

void main()
{
int i;
int N;
ADCON1=0x0A;
TRISB=0;
// configure A/D convertor

OpenADC(ADC_FOSC_64 &ADC_RIGHT_JUST &ADC_2_TAD,ADC_CH0 &
ADC_INT_OFF &
ADC_VREFPLUS_VDD & ADC_VREFMINUS_VSS,0x01);
//SetChanADC(ADC_CH2);
ConvertADC(); // Start conversion

PORTB=0;
while( BusyADC()==1 ); // Wait for completion
N = ReadADC(); // Read result

for(i=0;i<100;++i)
Nop();
if(N>10 && N<100)
PORTB=1;
else if(N>100)
PORTB=2;
CloseADC();// Disable A/D converter
}
```

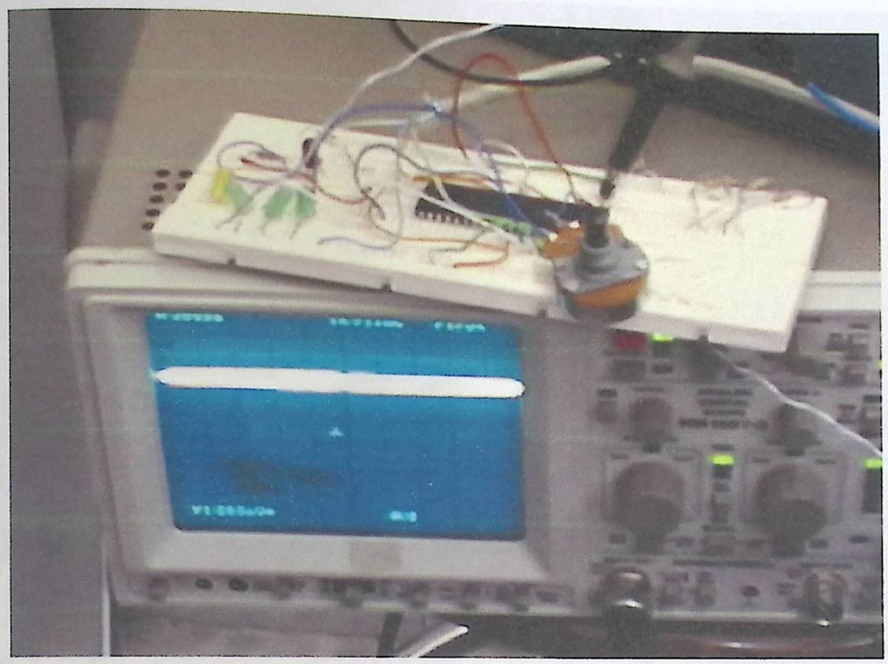


Figure 6.15: ADC Testing Circuit

6.6 Programmer Testing

The testing of the programmer that builds in this project includes testing of program load, read, program, verify, hardware and erase.

First of all and after finishing the programmer circuit, the team in this project make sure that an indicators must be exists for all type of programmer testing processes; these indicators include two LEDs (Red and Green) as shown in figure 6.16.

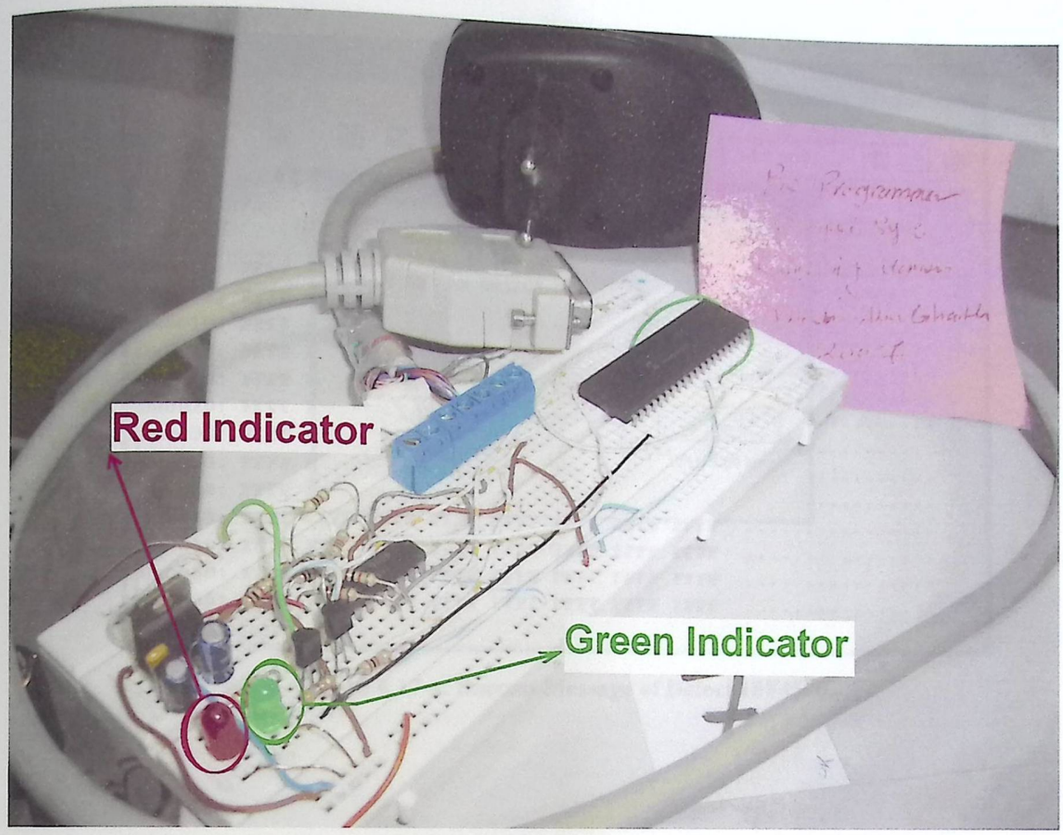


Figure 6.16: PIC18F4550 Programmer Indicators

The detecting device is very important step to check if the programmer is working probably, to ensure from that the detected message from WinPic800 must be appeared telling the user what device is detected and the ID number of the device (See the figure 5.17 a), if the detection of the device is not matched the UNKNOWN message appears (See the figure 5.17 b). Anther thing that refer to the success of the detection of the device is the LEDs indicators itself; the both of LEDs (Red and Green) must be lighted to indicate that the detection is finished successfully.

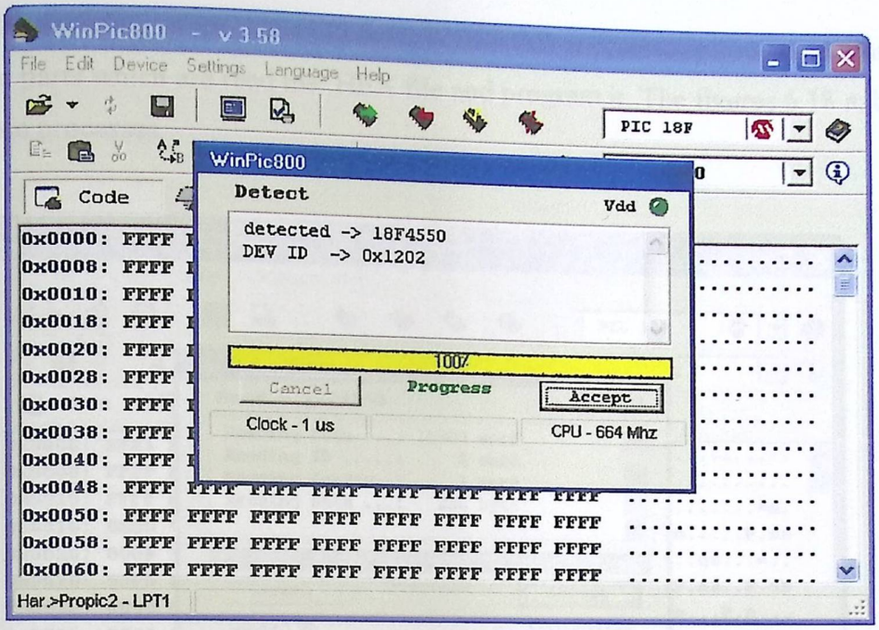


Figure 6.17 a: Success Message of Detect 18F4550

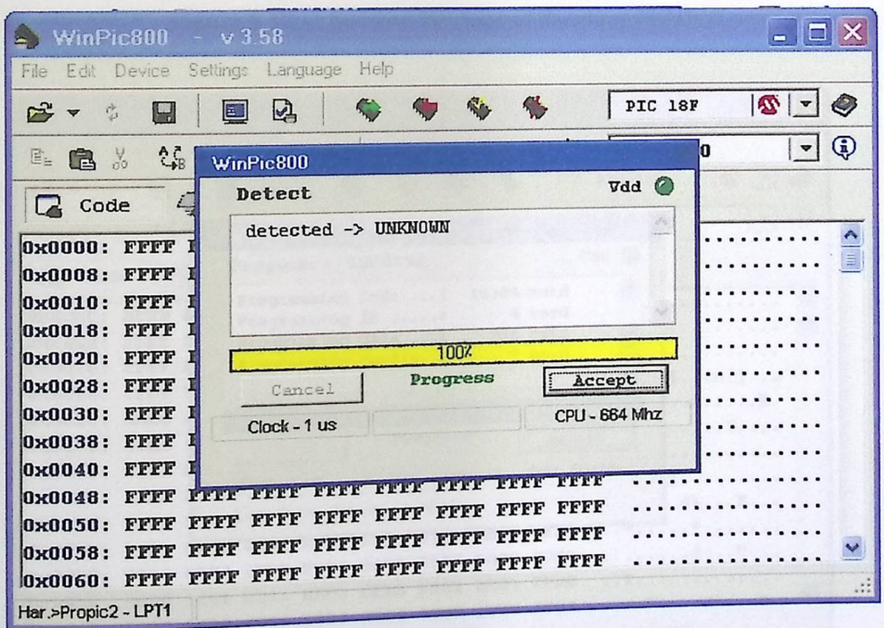


Figure 6.17 b: Fail Message of Detect 18F4550

After detecting of PIC18F4550 from WinPic800 successfully become the most important parts which are read the .HEX file and program it. The figures 6.18 a, b show these processes.

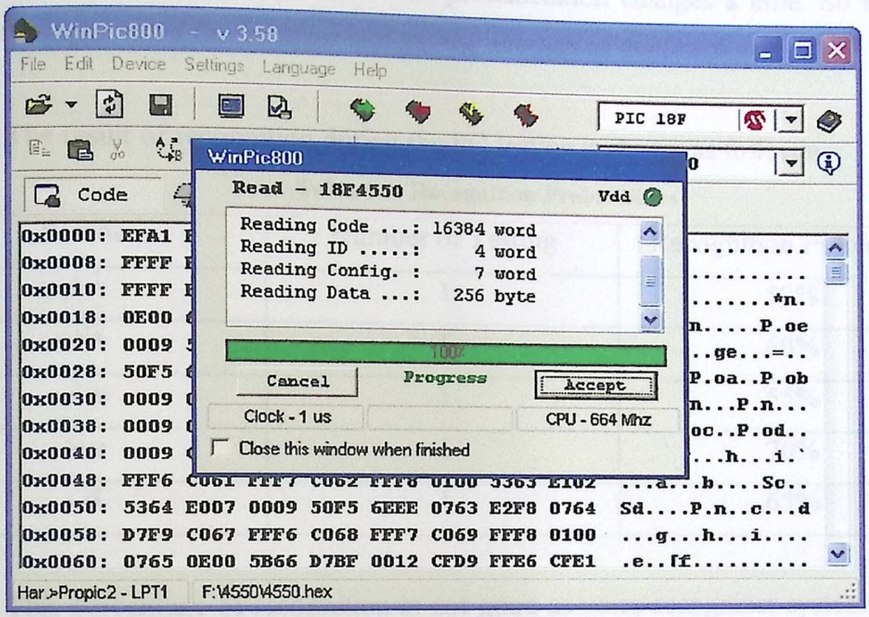


Figure 6.18 a: Success Message of Reading 18F4550

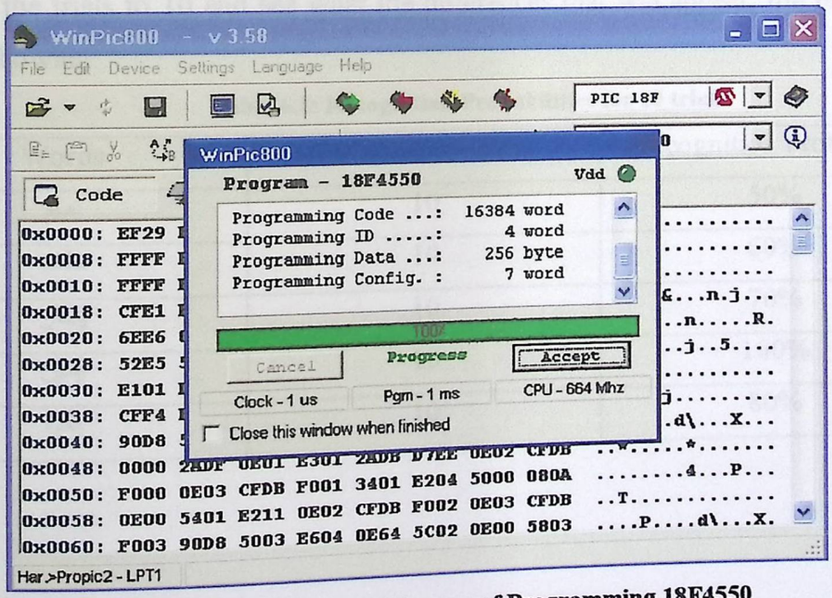


Figure 6.18 b: Success Message of Programming 18F4550

6.7 Software Testing

Actually, the project have big problem during testing. The team found that the template will change a lot even if his pronunciation changes a little. So tried to record the same word for 20 times and get the average of the templates.

The result of recognition during the PC testing is shown as following:

Table 6.1: Recognition Probabilities

Words	Number of Testing	Recognition Probability
أمام	1	50%
خلف	1	60%
يسار	1	55%
يمين	1	70%
قف	1	65%

This percentage of recognition in not good as voice recognizer system so, the needed for improve this probabilities is necessary, the project team decided to increase the trials to 10 and see what the differences that will appear, the result was as the following:

Table 6.2: Recognition Probabilities for 10 trials

Words	Number of Testing	Recognition Probability
أمام	10	50%
خلف	10	60%
يسار	10	70%
يمين	10	100%
قف	10	80%

6.7 Software Testing

Actually, the project have big problem during testing. The team found that the template will change a lot even if his pronunciation changes a little. So tried to record the same word for 20 times and get the average of the templates.

The result of recognition during the PC testing is shown as following:

Table 6.1: Recognition Probabilities

Words	Number of Testing	Recognition Probability
أمام	1	50%
خلف	1	60%
يسار	1	55%
يمين	1	70%
قف	1	65%

This percentage of recognition in not good as voice recognizer system so, the needed for improve this probabilities is necessary, the project team decided to increase the trials to 10 and see what the differences that will appear, the result was as the following:

Table 6.2: Recognition Probabilities for 10 trials

Words	Number of Testing	Recognition Probability
أمام	10	50%
خلف	10	60%
يسار	10	70%
يمين	10	100%
قف	10	80%

The result of after getting the average of 10 trials still not enough, the project team increase the number of trials until the system is reach to its maximum perfect values, from the practical issue the system was more accurate just only if 20 trials used as shown in table 6.3

Table 6.3: Recognition Probabilities for 20 trials

Words	Number of Testing	Recognition Probability
أمام	20	80%
خلف	20	80%
يسار	20	75%
يمين	20	100%
قف	20	90%

The system actually doesn't reach the 100% of recognition probability for all words when we increase the number of trials; the reason for that returns to the characteristics of the system that the team analyze, where the recognition probability start to decrease when the number of trials exceeds 20 trials.

6.8 Testing the Whole System

In this section the system will be tested as a whole, the assembling of the subsystems could raises some problems that will be faced any designer. Actually these problems will appear if the design has some weakness that reflects to the hardware. Figure 6.19 show the whole system of the project.

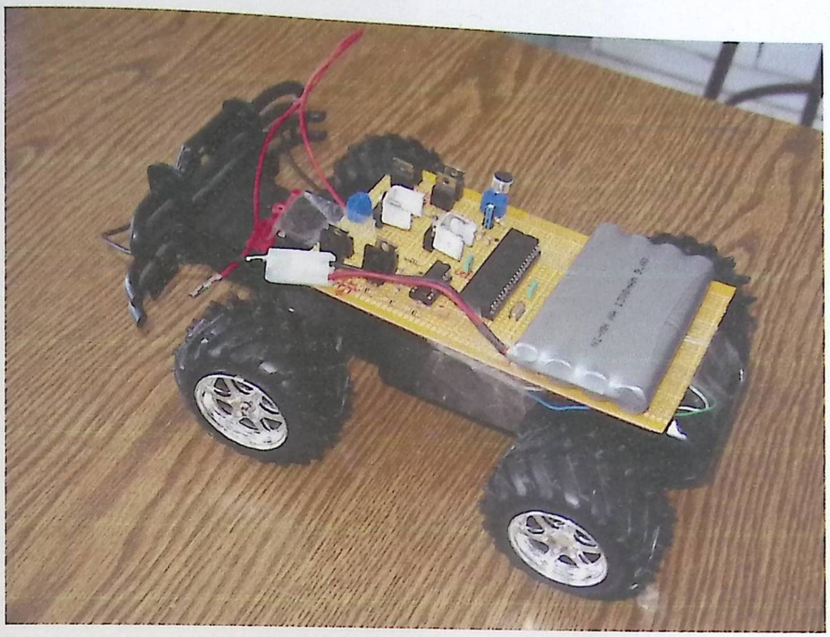


Figure 6.19 The whole project testing car

After finishing all hardware connections of the system, it will an important issue to apply some programs that ensure from these subsystems will work kindly when they connected with each other. The testing code for ADC and H-bridge are tested on the system as a whole and the results was within the project expects.

Both of H-bridge must be tested to a void any mistakes that could happened during wrapping, simple testing code needed to ensure from that. The code must set command to the car in four directions and stop to see that work well.

The microphone too is an important part that must tested. The testing part of the microphone includes testing of its amplification circuit for the voice signal that comes from it. The resulted signal is achieved the project needs and the peak to peak signal is gained around 4V.

7

Chapter Seven

Conclusions and Future Works

7.1 Introduction

In this chapter we will present some conclusions that we got from performing and testing our project, also we will explain in details the goals that we have achieved from the project .And finally we will talk about future work that will help us to develop our project.

7.2 Conclusions

Although many difficulties had faced us in our work, but we have passed them successfully, on the other hand these difficulties increased our experience as engineers and gave us the chance to apply what have learned in the previous years, and so we have reached to the following:

- We proved that our methods in the project were executable, and many of them were applicable.
- In the first semester, the team of the project put the aims of the project and studied the theoretical part of the project (theories and laws), general block diagram to the system and explained how the system works. (You can see this in chapter 1, 2 and 3).
- In this semester, we have divide the system into modules (microphone module, microcontroller module, H-bridge module and car & power module)

.we had built these subsystems, and we checked that they were working correctly and achieved the aims.

- To program the PIC18F4550, we have used the MPLAB ICE with MPLAB debugger and the programming device.
- The microcontroller can be programmed in many languages, but we have used C language, and so all the programs in the systems were written in C.
- The subsystems were joined to each others on a single board and we test it, it works correctly.
- The main goals of the project were to recognize five words (أمام, خلف, يمين, يسار, قف) from the system, and this goal was performed successfully.
- Templates were built for the five words using MATLAB and they stored in the program memory of the PIC.
- The voice recognition program was written using C.
- The project has not met our expectations fully, as we initially specified that the system would be able to recognize five words perfectly, but we are more than happy that it is able to recognize words by more than 80%-90%.

7.3 Problems

Our project as any other projects faced several problems while working in. These problems can be divided into two parts HW problems and SW problems.

7.3.1 Hardware Problems

- A problem was faced us in making the voltage of the signal in microphone circuit appropriate.
- The work was paused for three weeks at least, because the PIC programmer was destroyed completely .and there was no spare PIC programmer in the university, and it was so hard to the university to provide us with this programmer in the limited time.
- The H-bridge (LMD18200) chip that provides the enough current to the car motors was not available in local markets so we were forced to build it manually.

7.3.2 Software Problems

- We haven't any knowledge in using MATLAB, so we had learned it.
- Our scientific background about voice recognition was limited.
- It was hard to use the MATLAB in indicating the main frequency spectrum that needed in templates building.
- Euclidean distance can't recognize the spoken words completely so, we have used a correlation to help the Euclidean distance in recognizing the words.
- Recording the voice using MATLAB and changing the properties of the .wav file to be suitable to the internal ADC of the PIC18F4550 was so difficult.

7.4 Future Work

We can add some ideas that the students can work on in the future such that:

- Develop system to be recognized sentences of speech rather than separated spoken words.
- Make the system user independent instead of dependency on the user in this project.
- Replace the build H-bridge with LMD18200 H-bridge chip that provide many features that will develop robotic car.
- Replace the PIC18F4550 with DSP that provides many features that will develop robotic car.
- Use wireless technology instead of wired microphone to make the car have more mobility.
- This technology can be applied on a real car to be controlled under human voice.

Appendix

A

C Code for Generating Template (Training Code)

This code read 4000 sample from files that generated from MATLAB and generate equivalent template. The number of trials for each word is 20, the algorithm take the average of them and store the final template to be used in *PIC18F4550 Microcontroller Code*

```
#include <conio.h>
#include <math.h>
#include <stdio.h>
#define NSAMPLES_PER_INTERVAL 500
#define NSAMPLES 4000
#define NDIM     40
#define NWORDS   5

int Buffer[NSAMPLES];
float word[NDIM];
float dic[NWORDS][NDIM]={
28.731697,
23.806240,
20.743170,
20.729727,
20.746929,
28.744881,
25.084929,
21.103733,
22.558046,
21.043518,
28.933342,
26.885273,
21.959538,
23.883915,
21.716377,
28.796268,
27.978359,
22.364197,
25.513716,
22.400810,
28.842926,
27.068165,
21.975847,
23.866861,
21.658831,
28.775497,
23.888414,
20.776482,
20.809769,
```

20.779552,
28.731445,
23.801704,
20.738886,
20.704576,
20.743683,
28.733120,
23.803177,
20.740131,
20.705616,
20.744926,
//////////////////
28.730694,
23.805521,
20.742664,
20.732788,
20.746441,
28.738544,
23.989052,
20.831535,
22.224514,
21.071753,
28.791235,
27.701809,
21.999575,
24.949852,
22.311052,
28.796368,
27.500851,
21.774183,
24.109571,
21.862074,
28.782408,
24.250769,
20.852444,
21.571100,
20.860710,
28.756866,
23.841282,
20.768417,
20.861317,
20.777674,
28.739346,
23.809923,
20.745745,
20.713623,
20.750641,
28.728535,
23.798317,
20.735703,

20.701450,
20.740458,
//////////////////////
28.731129,
23.806149,
20.742559,
20.732464,
20.746326,
28.767147,
25.331806,
20.896290,
21.932724,
21.054085,
28.780079,
26.062557,
21.177767,
22.934605,
21.273487,
28.824829,
28.132717,
22.422709,
23.851662,
21.877268,
28.797853,
28.033220,
22.454182,
23.326935,
21.711346,
28.751104,
25.265984,
21.255527,
21.642654,
20.968977,
28.731586,
23.853800,
20.781469,
20.811054,
20.749832,
28.737186,
23.807840,
20.744469,
20.710148,
20.749212,
//////////////////////
28.728233,
23.802544,
20.739546,
20.727024,
20.743114,
28.745340,

24.168457,
20.768169,
20.857685,
20.774775,
28.840431,
26.344090,
21.297783,
21.953094,
21.159582,
28.890203,
25.034172,
21.061024,
21.273497,
20.928539,
28.853207,
25.346708,
20.981083,
21.439224,
20.947149,
28.812363,
24.179874,
20.794943,
20.864792,
20.806000,
28.731918,
23.804096,
20.739159,
20.707567,
20.744026,
28.733498,
23.803875,
20.740778,
20.708618,
20.745659,
//////////////////
28.731701,
23.814907,
20.745893,
20.831062,
20.755859,
28.737976,
24.641148,
21.041561,
21.934521,
21.085316,
28.808460,
27.517452,
22.095715,
24.724064,
22.240616,

```
28.838337,
25.047195,
21.145464,
22.485378,
21.219467,
28.797178,
24.115572,
20.820148,
21.117802,
20.855694,
28.729425,
23.810867,
20.736769,
20.714859,
20.741554,
28.738997,
23.809818,
20.746243,
20.711502,
20.751064,
28.731434,
23.801487,
20.738579,
20.704308,
20.743324 };

void initialize();
void get_sample();
void analyze( int *, float * );
int  lookup();
float correlation(int x[NDIM], float y[NDIM]);
int find_min(float a[NWORDS]);
void control(int command);
int float2fix (float a);
float fix2float (int a);


float filter1( float x );
float filter2( float x );
float filter3( float x );
float filter4( float x );
float filter5( float x );

FILE *spoken_in;
FILE *spoken_outfile;

void main()
{
clrscr();
        int i, j;
```

```
        initialize(); // open files

                int command = -1;

                get_sample();// read in a word sample (assume framing is done by low
level routine)
                //printf("%d\n",Buffer[0]);
                fclose(spoken_in);
                analyze( Buffer, word );

                for( j=0; j < NDIM; j++ ) {
                        fprintf(spoken_outfile,"%f\n",word[j] );
                }

                command = lookup();
                control( command );

getch();
}

////////////////////////////////////////////////////////////////////////////
//

void initialize()
{

        spoken_in = fopen( "d:\\right4k\\right12.txt", "r" );
                if (spoken_in  == NULL)
                printf("Cannot open input file.\n");
        spoken_outfile = fopen( "c:\\spokentemp.txt", "w" );

}

////////////////////////////////////////////////////////////////////////////


void get_sample()
{
        int i;
        for( i=0; i < 4000; i++ ){
                fscanf( spoken_in, "%d\n", Buffer+i );
        }
}

////////////////////////////////////////////////////////////////////////////


void analyze( int *sample, float *fingerprint )
{
        int i, k, x;
```

```
        k = 0;
        for( i=0; i < NSAMPLES; i+= NSAMPLES_PER_INTERVAL ) {
                int j;
                float y1=0.0f, y2=0.0f, y3=0.0f, y4=0.0f, y5=0.0f;
                float m;

                for( j=0; j < NSAMPLES_PER_INTERVAL; j++ ) {
                        float y = filter1( (float) sample[i+j] );
                        y1 += y*y;
                }
                m = frexp(y1+1, &x);
                fingerprint[k++] = x+m;

                for( j=0; j < NSAMPLES_PER_INTERVAL; j++ ) {
                        float y = filter2( (float) sample[i+j] );
                        y2 += y*y;
                }
                m = frexp(y2+1, &x);
                fingerprint[k++] = x+m;

                for( j=0; j < NSAMPLES_PER_INTERVAL; j++ ) {
                        float y = filter3( (float) sample[i+j] );
                        y3 += y*y;
                }
                m = frexp(y3+1, &x);
                fingerprint[k++] = x+m;

                for( j=0; j < NSAMPLES_PER_INTERVAL; j++ ) {
                        float y = filter4( (float) sample[i+j] );
                        y4 += y*y;
                }
                m = frexp(y4+1, &x);
                fingerprint[k++] = x+m;

                for( j=0; j < NSAMPLES_PER_INTERVAL; j++ ) {
                        float y = filter5( (float) sample[i+j] );
                        y5 += y*y;
                }
                m = frexp(y5+1, &x);
                fingerprint[k++] = x+m;

        }
}
int lookup()
{
        int result, i, j,temp;
        float differ[NWORDS];
        int tmp_array[NDIM];
```

```
        for(i=0; i< NWORDS ; i++){
                differ[i]=0.0;
        }

        for(i=0 ; i< NWORDS ; i++){
                for(j=0; j<NDIM ; j++){
                        tmp_array[j]=float2fix(dic[i][j]);
                }
                differ[i]=correlation(tmp_array, word);
                printf("i:%d \t, differ: %f\r\n",i,differ[i]);
        }
            result = find_min(differ);  //  case: only one code has large coefficients

          return result;
}

float correlation(int x[NDIM], float y[NDIM])
{
        int i;
        float meanx1=0,meanx2=0,meanxy=0,meany1=0,meany2=0;
        float re;

        for(i=0 ; i<NDIM ;i++){
        meanx1=meanx1+fix2float(x[i]);
        meanx2=meanx2+fix2float(x[i])*fix2float(x[i]);
        meany1=meany1+y[i];
        meany2=meany2+y[i]*y[i];
        meanxy=meanxy+fix2float(x[i])*y[i];

        }
        if( ((NDIM*meanx2-meanx1*meanx1)==0) && ((NDIM*meany2-
meany1*meany1)==0) )
                re=0;
        else{
                re=(NDIM*meanxy-meanx1*meany1)*(NDIM*meanxy-
meanx1*meany1)/(NDIM*meanx2-meanx1*meanx1)/(NDIM*meany2-
meany1*meany1);
                re=1-re;
        }

        if(re < 0)
                re= -re;

        return re;

}

int find_min(float a[NWORDS])
{
  float smallest;
```

```
 int i;
 int index;

   index =0;
   smallest = a[0];

   for(i=1; i<NWORDS ;i++)
   {
    if(a[i]<smallest)
    {
        smallest=a[i];
        index = i;
    }
   }

   return index;
}
////////////////////////////////////
int float2fix (float a){
 return ((int)((a)*256.0));
}

float fix2float (int a){
 return ((float)((a)/256.0));
}


////////////////////////////////////

void control( int command )
{
        switch(command){
                case 0: printf("You speek the FRONT word...");
                        break;
                case 1: printf("You speek the BACK word...");
                        break;
                case 2: printf("You speek the LEFT word...");
                        break;
                case 3: printf("You speek the RIGHT word...");
                        break;
                case 4: printf("You speek the STOP word...");
                        break;
        }

}

#define A11 -1.4212f
#define A21  0.5181f
#define B01  0.0511f
#define B11 -0.0166f
```

```
#define B21  0.0511f

#define A12 -1.7096f
#define A22  0.8046f
#define B02  2.5183f
#define B12 -3.9278f
#define B22  2.5183f

#define G 0.0970f

float filter1( float x )
{
        static float d01 = 0.0;
        static float d11 = 0.0;
        static float d02 = 0.0;
        static float d12 = 0.0;
        float y1, y2, t0, t1;

        /* first 2nd-order filter stage */
        t0 = x - A11*d01 - A21*d11;
        y1 = B01*t0 + B11*d01 + B21*d11;
        d11 = d01;
        d01 = t0;
        /* second 2nd-order filter stage */
        t1 = y1 - A12*d02 - A22*d12;
        y2 = B02*t1 + B12*d02 + B21*d12;
        d12 = d02;
        d02 = t1;

        return G*y2;
}

#undef A11
#undef A21
#undef B01
#undef B11
#undef B21

#undef A12
#undef A22
#undef B02
#undef B12
#undef B22

#undef G

/*************************************************************************
**********/

#define A11 -0.7780f
```

```
#define A21  0.7647f
#define B01  0.2228f
#define B11  0.0090f
#define B21  0.2228f

#define A12 -1.2046f
#define A22  0.8011f
#define B02  2.2293f
#define B12 -3.8444f
#define B22  2.2293f

#define G 0.2108f

float filter2( float x )
{
        static float d01 = 0.0;
        static float d11 = 0.0;
        static float d02 = 0.0;
        static float d12 = 0.0;
        float y1, y2, t0, t1;

        /* first 2nd-order filter stage */
        t0 = x - A11*d01 - A21*d11;
        y1 = B01*t0 + B11*d01 + B21*d11;
        d11 = d01;
        d01 = t0;
        /* second 2nd-order filter stage */
        t1 = y1 - A12*d02 - A22*d12;
        y2 = B02*t1 + B12*d02 + B21*d12;
        d12 = d02;
        d02 = t1;

        return G*y2;
}


#undef A11
#undef A21
#undef B01
#undef B11
#undef B21

#undef A12
#undef A22
#undef B02
#undef B12
#undef B22

#undef G
```

```
/**********************************************************************
**********/

#define A11  0.2044f
#define A21  0.7757f
#define B01  0.3251f
#define B11 -0.1984f
#define B21  0.3251f

#define A12  0.6814f
#define A22  0.7896f
#define B02  1.4460f
#define B12  1.9706f
#define B22  1.4460f

#define G    0.2227f

float filter3( float x )
{
        static float d01 = 0.0;
        static float d11 = 0.0;
        static float d02 = 0.0;
        static float d12 = 0.0;
        float y1, y2, t0, t1;

        /* first 2nd-order filter stage */
        t0 = x - A11*d01 - A21*d11;
        y1 = B01*t0 + B11*d01 + B21*d11;
        d11 = d01;
        d01 = t0;
        /* second 2nd-order filter stage */
        t1 = y1 - A12*d02 - A22*d12;
        y2 = B02*t1 + B12*d02 + B21*d12;
        d12 = d02;
        d02 = t1;

        return G*y2;
}

#undef A11
#undef A21
#undef B01
#undef B11
#undef B21

#undef A12
#undef A22
#undef B02
#undef B12
#undef B22
```

```
#undef G

/*****************************************************************************
**********/

#define A11   1.4059f
#define A21   0.7969f
#define B01   0.1272f
#define B11   0.1113f
#define B21   0.1272f

#define A12   1.6649f
#define A22   0.8525f
#define B02   6.2540f
#define B12  12.0986f
#define B22   6.2540f

#define G     0.1252f

float filter4( float x )
{
        static float d01 = 0.0;
        static float d11 = 0.0;
        static float d02 = 0.0;
        static float d12 = 0.0;
        float y1, y2, t0, t1;

        /* first 2nd-order filter stage */
        t0 = x - A11*d01 - A21*d11;
        y1 = B01*t0 + B11*d01 + B21*d11;
        d11 = d01;
        d01 = t0;
        /* second 2nd-order filter stage */
        t1 = y1 - A12*d02 - A22*d12;
        y2 = B02*t1 + B12*d02 + B21*d12;
        d12 = d02;
        d02 = t1;

        return G*y2;
}

#undef A11
#undef A21
#undef B01
#undef B11
#undef B21

#undef A12
#undef A22
```

```
#undef G

/*****************************************************************************
**********/

#define A11   1.4059f
#define A21   0.7969f
#define B01   0.1272f
#define B11   0.1113f
#define B21   0.1272f

#define A12   1.6649f
#define A22   0.8525f
#define B02   6.2540f
#define B12  12.0986f
#define B22   6.2540f

#define G     0.1252f

float filter4( float x )
{
        static float d01 = 0.0;
        static float d11 = 0.0;
        static float d02 = 0.0;
        static float d12 = 0.0;
        float y1, y2, t0, t1;

        /* first 2nd-order filter stage */
        t0 = x - A11*d01 - A21*d11;
        y1 = B01*t0 + B11*d01 + B21*d11;
        d11 = d01;
        d01 = t0;
        /* second 2nd-order filter stage */
        t1 = y1 - A12*d02 - A22*d12;
        y2 = B02*t1 + B12*d02 + B21*d12;
        d12 = d02;
        d02 = t1;

        return G*y2;
}

#undef A11
#undef A21
#undef B01
#undef B11
#undef B21

#undef A12
#undef A22
```

```
#undef B02
#undef B12
#undef B22

#undef G


/*****************************************************************************
**********/

#define A11   1.4059f
#define A21   0.7969f
#define B01   0.3047f
#define B11   0.4312f
#define B21   0.3047f

#define A12   1.6649f
#define A22   0.8525f
#define B02   2.2075f
#define B12   4.1629f
#define B22   2.2075f

#define G     0.1252f

float filter5( float x )
{
        static float d01 = 0.0;
        static float d11 = 0.0;
        static float d02 = 0.0;
        static float d12 = 0.0;
        float y1, y2, t0, t1;

        /* first 2nd-order filter stage */
        t0 = x - A11*d01 - A21*d11;
        y1 = B01*t0 + B11*d01 + B21*d11;
        d11 = d01;
        d01 = t0;
        /* second 2nd-order filter stage */
        t1 = y1 - A12*d02 - A22*d12;
        y2 = B02*t1 + B12*d02 + B21*d12;
        d12 = d02;
        d02 = t1;

        return G*y2;
}

#undef A11
#undef A21
#undef B01
#undef B11
```

```
#undef B21

#undef A12
#undef A22
#undef B02
#undef B12
#undef B22

#undef G
```

PIC18F4550 Microcontroller Code

```
#include <math.h>
#include <p18f4550.h>
#include <adc.h>
#include <timers.h>///becarefull that the file will not copy probably
#pragma config FOSC = HS //must be external
#pragma config WDT = OFF
#pragma config LVP = OFF
#define NSAMPLES_PER_INTERVAL 500
#define NSAMPLES 4000
#define NDIM     40
#define NWORDS   5
#define IDLE 0
#define OK 1
#define COMPARE 2
#define float2fix(a) ((int)((a)*256.0))
#define fix2float(a) ((float)((a)/256.0))
int N;// sample value that get from ADC
float word[NDIM];
float y1, y2, t0, t1;
float filt1=0.0,filt2=0.0,filt3=0.0,filt4=0.0,filt5=0.0;
int counter = 500,state=IDLE,ptr=0;
#pragma romdata const_table
const rom float dic[NWORDS][NDIM]={
28.731697,
23.806240,
20.743170,
20.729727,
20.746929,
28.744881,
25.084929,
21.103733,
22.558046,
21.043518,
```

28.933342,
26.885273,
21.959538,
23.883915,
21.716377,
28.796268,
27.978359,
22.364197,
25.513716,
22.400810,
28.842926,
27.068165,
21.975847,
23.866861,
21.658831,
28.775497,
23.888414,
20.776482,
20.809769,
20.779552,
28.731445,
23.801704,
20.738886,
20.704576,
20.743683,
28.733120,
23.803177,
20.740131,
20.705616,
20.744926,
////////////////// end of "امام"spoken word template
28.730694,
23.805521,
20.742664,
20.732788,
20.746441,
28.738544,
23.989052,
20.831535,
22.224514,
21.071753,
28.791235,
27.701809,
21.999575,
24.949852,
22.311052,
28.796368,
27.500851,
21.774183,
24.109571,

21.862074,
28.782408,
24.250769,
20.852444,
21.571100,
20.860710,
28.756866,
23.841282,
20.768417,
20.861317,
20.777674,
28.739346,
23.809923,
20.745745,
20.713623,
20.750641,
28.728535,
23.798317,
20.735703,
20.701450,
20.740458,
////////////////////// end of "خلف"spoken word template
28.731129,
23.806149,
20.742559,
20.732464,
20.746326,
28.767147,
25.331806,
20.896290,
21.932724,
21.054085,
28.780079,
26.062557,
21.177767,
22.934605,
21.273487,
28.824829,
28.132717,
22.422709,
23.851662,
21.877268,
28.797853,
28.033220,
22.454182,
23.326935,
21.711346,
28.751104,
25.265984,
21.255527,

21.642654,
20.968977,
28.731586,
23.853800,
20.781469,
20.811054,
20.749832,
28.737186,
23.807840,
20.744469,
20.710148,
20.749212,
////////////////////// end of "يسار"spoken word template
28.728233,
23.802544,
20.739546,
20.727024,
20.743114,
28.745340,
24.168457,
20.768169,
20.857685,
20.774775,
28.840431,
26.344090,
21.297783,
21.953094,
21.159582,
28.890203,
25.034172,
21.061024,
21.273497,
20.928539,
28.853207,
25.346708,
20.981083,
21.439224,
20.947149,
28.812363,
24.179874,
20.794943,
20.864792,
20.806000,
28.731918,
23.804096,
20.739159,
20.707567,
20.744026,
28.733498,
23.803875,

```
20.740778,
20.708618,
20.745659,
////////////////// end of "يمين" spoken word template
28.731701,
23.814907,
20.745893,
20.831062,
20.755859,
28.737976,
24.641148,
21.041561,
21.934521,
21.085316,
28.808460,
27.517452,
22.095715,
24.724064,
22.240616,
28.838337,
25.047195,
21.145464,
22.485378,
21.219467,
28.797178,
24.115572,
20.820148,
21.117802,
20.855694,
28.729425,
23.810867,
20.736769,
20.714859,
20.741554,
28.738997,
23.809818,
20.746243,
20.711502,
20.751064,
28.731434,
23.801487,
20.738579,
20.704308,
20.743324 };/// end of "قف"spoken word template
#pragma romdata

void high_isr(void);
void initialize(void);
void analyze( int );
int lookup(void);
```

```
float correlation(int x[NDIM], float y[NDIM]);
int find_min(float a[NWORDS]);
void control(int command);
void motors_steering(char dir);
void motors_moving(char dir);

float filter1( float x );
float filter2( float x );
float filter3( float x );
float filter4( float x );
float filter5( float x );


void main(void)
{

int x, k, j; float m;
        //unsigned char command='0';
        TRISAbits.TRISA1 = 0;
        TRISAbits.TRISA2 = 0;
        TRISAbits.TRISA3 = 0;
        TRISAbits.TRISA4 = 0;
//////////////////////////
        PORTAbits.RA1 = 1;
        PORTAbits.RA2 = 1;
        PORTAbits.RA3 = 1;
        PORTAbits.RA4 = 1;

   TRISEbits.TRISE0 = 1;
        TRISBbits.TRISB0 = 0; ///TESTING LED (Green)
        TRISBbits.TRISB1 = 0; ///TESTING LED2 (Red)
        PORTBbits.RB0=0;
        PORTBbits.RB1=0;

        control(4); //set default "قف"state...
        ADCON0=0x01;
        ADCON1=0x0E;

        initialize(); // open ADC ... to get samples with timing

        INTCON=0b11100100;


        while(1){

        if(state == COMPARE){
                control( lookup() ); /// take appropriate action according matching
algorithm                                                       /// return to IDLE state
                state=IDLE;                                     /// activate timer
                WriteTimer0(0);
```

```
            }
        else if(state == OK && counter > 0){
             counter--;
            analyze(N);//--------
            ConvertADC(); // Start conversion
            while( BusyADC()==1 ); // Wait for completion
                    N = ReadADC(); // Read result
            ///CloseADC();// Disable A/D converter
            }
        else if(counter == 0){
            counter=500;

            m = frexp(filt1+1, &x);
            word[ptr++]== x+m;
            m = frexp(filt2+1, &x);
            word[ptr++]=x+m;
            m = frexp(filt3+1, &x);
            word[ptr++]=x+m;
            m = frexp(filt4+1, &x);
            word[ptr++]=x+m;
            m = frexp(filt5+1, &x);
            word[ptr++]=x+m;
                    PORTBbits.RB0=!PORTBbits.RB0; // appear 8 flashing led
            if(ptr > NDIM){

            state = COMPARE;
            ptr=0;
            filt1=0;
            filt2=0;
            filt3=0;
            filt4=0;
            filt5=0;
            }
            }
        }

}

//////////////////////////////////////////////////////////////////////////////

/*********************************************************************
**********/
/*
* For PIC18 devices the high interrupt vector is found at
* 00000008h. The following code will branch to the
* high_interrupt_service_routine function to handle
* interrupts that occur at the high vector.
*/
```

```
#pragma code high_vector=0x08
void interrupt_at_high_vector(void)
{
_asm GOTO high_isr _endasm
}
#pragma code /* return to the default code section */

#pragma interrupt high_isr


void high_isr (void)
{
if(INTCONbits.TMR0IF) {          // Timer 0 ..

        ConvertADC(); // Start conversion
        while( BusyADC()==1 ); // Wait for completion
                        N = ReadADC(); // Read result

        INTCONbits.TMR0IF=0;          // clear bit IRQ
        WriteTimer0(0);          // reinitializ TMR0  63536


        if((PORTEbits.RE0==0 )&& (state == IDLE)&& N<150){///determine if the
threshold exceeded...+150
                        state = OK;
        PORTBbits.RB0=!PORTBbits.RB0; // flash led when threshold exceeded
        }

}
}
//////////////////////

void initialize(void)
{

int i,j;
// configure A/D convertor
// ADC initilaize
                OpenADC(ADC_FOSC_64   &
                ADC_RIGHT_JUST &
                ADC_2_TAD ,
                ADC_CH0    &
                ADC_INT_OFF & ADC_VREFPLUS_VDD &
ADC_VREFMINUS_VSS , 0x0E);

// configure Timer0
OpenTimer0( TIMER_INT_ON &T0_16BIT &T0_SOURCE_INT &T0_PS_1_1  );
INTCONbits.GIE = 1;
WriteTimer0(0);///
/*
```

```
*/
}

////////////////////////////////////////////////////////////////////////

void analyze( int sample)
{
        float y;

                        y = filter1( (float) sample );
                        filt1 += y*y;

                        y = filter2( (float) sample );
                        filt2 += y*y;

                        y = filter3( (float) sample );
                        filt3 += y*y;

                        y = filter4( (float) sample );
                        filt4 += y*y;

                        y = filter5( (float) sample );
                        filt5 += y*y;

}
int lookup(void)
{
        int result;
        int i, j,temp;
        float differ[NWORDS]; // array that stored correlation values that resulted by
matching process
        int tmp_array[NDIM];

        for(i=0; i< NWORDS ; i++){
                differ[i]=0.0;
        }

        for(i=0 ; i< NWORDS ; i++){
                for(j=0; j<NDIM ; j++){
                        tmp_array[j]=float2fix(dic[i][j]);
                }
                differ[i]=correlation(tmp_array, word);
        }
            result = find_min(differ);  // find the minimum correlation

          return result;
}
```

```
float correlation(int x[NDIM], float y[NDIM])
{
        int i;
        float meanx1=0,meanx2=0,meanxy=0,meany1=0,meany2=0;
        float re;

        for(i=0 ; i<NDIM ;i++){
        meanx1=meanx1+fix2float(x[i]);
        meanx2=meanx2+fix2float(x[i])*fix2float(x[i]);
        meany1=meany1+y[i];
        meany2=meany2+y[i]*y[i];
        meanxy=meanxy+fix2float(x[i])*y[i];

        }
        if( ((NDIM*meanx2-meanx1*meanx1)==0) && ((NDIM*meany2-
meany1*meany1)==0) )
                re=0;
        else{
                re=(NDIM*meanxy-meanx1*meany1)*(NDIM*meanxy-
meanx1*meany1)/(NDIM*meanx2-meanx1*meanx1)/(NDIM*meany2-
meany1*meany1);
                re=1-re;
        }

        if(re < 0)
                re= -re;

        return re;

}

int find_min(float a[NWORDS])
{
 float smallest;
 int i;
 int index;
int re;
    index =0;
    smallest = a[0];

    for(i=1; i<NWORDS ;i++)
    {
     if(a[i]<smallest)
     {
         smallest=a[i];
         index = i;
     }
    }

    return re;
```

```
}
////////////////////////////////////

void motors_steering(char dir)
{
        if (dir=='R')
        {
        PORTAbits.RA1 =0;
    PORTAbits.RA2 =1;
        }
    if (dir=='L')
        {
        PORTAbits.RA1 =1;
    PORTAbits.RA2 =0;
        }

        if(dir=='S')
{
PORTAbits.RA1 =0;
PORTAbits.RA2 =0;

}

}

void motors_moving(char dir)
{

        if (dir=='B' )
        {
        PORTAbits.RA3 =0;
    PORTAbits.RA4 =1;
        }
        if (dir=='F')
        {
        PORTAbits.RA3 =1;
    PORTAbits.RA4 =0;
        }
if(dir=='S')
{
PORTAbits.RA3 =0;
PORTAbits.RA4 =0;

}

}

////////////////////////////////////////////////////////////////////////////
```

```
void control( int command )
{
//PORTBbits.RB0=!PORTBbits.RB0;
        switch(command){
                case 0: motors_moving('S');
                                motors_steering('S');
                                motors_moving('F');// "أمام" word is detected ...
                        break;
                case 1: motors_moving('S');
                                motors_steering('S');
                                motors_moving('B');// "خلف" word is detected ...
                        break;
                case 2: motors_steering('L');// "يسار" word is detected ...
                        break;
                case 3: motors_steering('R');// "يمين" word is detected ...
                        break;
                case 4:
                                motors_moving('S');// "قف" word is detected ...
                                motors_steering('S');
                        break;
        }

}

#define A11 -1.4212f
#define A21  0.5181f
#define B01  0.0511f
#define B11 -0.0166f
#define B21  0.0511f

#define A12 -1.7096f
#define A22  0.8046f
#define B02  2.5183f
#define B12 -3.9278f
#define B22  2.5183f

#define G 0.0970f

float filter1( float x )
{

overlay float d01 = 0.0;
overlay float d11 = 0.0;
overlay float d02 = 0.0;
overlay float d12 = 0.0;
        /* first 2nd-order filter stage */
        t0 = x - A11*d01 - A21*d11;
        y1 = B01*t0 + B11*d01 + B21*d11;
        d11 = d01;
```

```
        d01 = t0;
        /* second 2nd-order filter stage */
        t1 = y1 - A12*d02 - A22*d12;
        y2 = B02*t1 + B12*d02 + B21*d12;
        d12 = d02;
        d02 = t1;

        return G*y2;
}

#undef A11
#undef A21
#undef B01
#undef B11
#undef B21

#undef A12
#undef A22
#undef B02
#undef B12
#undef B22

#undef G

/*************************************************************************
**********/

#define A11 -0.7780f
#define A21  0.7647f
#define B01  0.2228f
#define B11  0.0090f
#define B21  0.2228f

#define A12 -1.2046f
#define A22  0.8011f
#define B02  2.2293f
#define B12 -3.8444f
#define B22  2.2293f

#define G 0.2108f

float filter2( float x )
{

overlay float d01 = 0.0;
overlay float d11 = 0.0;
overlay float d02 = 0.0;
overlay float d12 = 0.0;
        /* first 2nd-order filter stage */
        t0 = x - A11*d01 - A21*d11;
```

```
        y1 = B01*t0 + B11*d01 + B21*d11;
        d11 = d01;
        d01 = t0;
        /* second 2nd-order filter stage */
        t1 = y1 - A12*d02 - A22*d12;
        y2 = B02*t1 + B12*d02 + B21*d12;
        d12 = d02;
        d02 = t1;

        return G*y2;
}


#undef A11
#undef A21
#undef B01
#undef B11
#undef B21

#undef A12
#undef A22
#undef B02
#undef B12
#undef B22

#undef G

/*************************************************************************
**********/

#define A11  0.2044f
#define A21  0.7757f
#define B01  0.3251f
#define B11 -0.1984f
#define B21  0.3251f

#define A12  0.6814f
#define A22  0.7896f
#define B02  1.4460f
#define B12  1.9706f
#define B22  1.4460f

#define G    0.2227f

float filter3( float x )
{

overlay float d01 = 0.0;
overlay float d11 = 0.0;
overlay float d02 = 0.0;
```

```
overlay float d12 = 0.0;
        /* first 2nd-order filter stage */
        t0 = x - A11*d01 - A21*d11;
        y1 = B01*t0 + B11*d01 + B21*d11;
        d11 = d01;
        d01 = t0;
        /* second 2nd-order filter stage */
        t1 = y1 - A12*d02 - A22*d12;
        y2 = B02*t1 + B12*d02 + B21*d12;
        d12 = d02;
        d02 = t1;

        return G*y2;
}

#undef A11
#undef A21
#undef B01
#undef B11
#undef B21

#undef A12
#undef A22
#undef B02
#undef B12
#undef B22

#undef G

/*****************************************************************************
**********/

#define A11   1.4059f
#define A21   0.7969f
#define B01   0.1272f
#define B11   0.1113f
#define B21   0.1272f

#define A12   1.6649f
#define A22   0.8525f
#define B02   6.2540f
#define B12  12.0986f
#define B22   6.2540f

#define G     0.1252f

float filter4( float x )
{

overlay float d01 = 0.0;
```

```
overlay float d11 = 0.0;
overlay float d02 = 0.0;
overlay float d12 = 0.0;
        /* first 2nd-order filter stage */
        t0 = x - A11*d01 - A21*d11;
        y1 = B01*t0 + B11*d01 + B21*d11;
        d11 = d01;
        d01 = t0;
        /* second 2nd-order filter stage */
        t1 = y1 - A12*d02 - A22*d12;
        y2 = B02*t1 + B12*d02 + B21*d12;
        d12 = d02;
        d02 = t1;

        return G*y2;
}

#undef A11
#undef A21
#undef B01
#undef B11
#undef B21

#undef A12
#undef A22
#undef B02
#undef B12
#undef B22

#undef G


/*************************************************************************
**********/

#define A11   1.4059f
#define A21   0.7969f
#define B01   0.3047f
#define B11   0.4312f
#define B21   0.3047f

#define A12   1.6649f
#define A22   0.8525f
#define B02   2.2075f
#define B12   4.1629f
#define B22   2.2075f

#define G     0.1252f

float filter5( float x )
```

```
{
overlay float d01 = 0.0;
overlay float d11 = 0.0;
overlay float d02 = 0.0;
overlay float d12 = 0.0;
        /* first 2nd-order filter stage */
        t0 = x - A11*d01 - A21*d11;
        y1 = B01*t0 + B11*d01 + B21*d11;
        d11 = d01;
        d01 = t0;
        /* second 2nd-order filter stage */
        t1 = y1 - A12*d02 - A22*d12;
        y2 = B02*t1 + B12*d02 + B21*d12;
        d12 = d02;
        d02 = t1;

        return G*y2;
}

#undef A11
#undef A21
#undef B01
#undef B11
#undef B21

#undef A12
#undef A22
#undef B02
#undef B12
#undef B22

#undef G
```

Appendix

B

References

[1] http://www.ee.ucr.edu/~dgiles/sendesign03/Speech%20Recognition.pdf

[2] H.M.Ahmad and J.I.Bashar and Q.K.Sahar, Palestine Polytechnic University, Cell Phone-Based Controller for a Toy Car, June 2005

[3]http://instruct1.cit.cornell.edu/courses/ee476/FinalProjects/s2006/XL76_SL362/XL76%20SL362/index.html

[4]Joseph P. Campbell, Jr.-SPEAKER RECOGNITION- Department of Defense, IEEE.

[5] http://ifsc.ualr.edu/sxdagtas/MM/SpeechRecognitionTutorial.pdf

[6] http://en.wikipedia.org/wiki/Nyquist%E2%80%93Shannon_sampling_theorem#The_sampling_process

[7] http://www.hitl.washington.edu/scivw/EVE/IV.Definitions.html

[8] http://www.hitl.washington.edu/scivw/EVE/IV.Definitions.html

[9] http://www.hitl.washington.edu/scivw/EVE/IV.Definitions.html

[10] ftp://ftp.idiap.ch/pub/reports/2007/gaudard-idiap-com-07-02.pdf

[11]http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/pdf/Speech-Recognition-HOWTO.pdf

PIC18F2455/2550/4455/4550
Data Sheet

28/40/44-Pin, High-Performance,
Enhanced Flash, USB Microcontrollers
with nanoWatt Technology

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Trademarks

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2002

Microchip received ISO/TS-16949:2002 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona and Mountain View, California in October 2003. The Company's quality system processes and procedures are for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

MICROCHIP PIC18F2455/2550/4455/4550

28/40/44-Pin, High-Performance, Enhanced Flash, USB Microcontrollers with nanoWatt Technology

Universal Serial Bus Features:

- USB V2.0 Compliant
- Low Speed (1.5 Mb/s) and Full Speed (12 Mb/s)
- Supports Control, Interrupt, Isochronous and Bulk Transfers
- Supports up to 32 Endpoints (16 bidirectional)
- 1-Kbyte Dual Access RAM for USB
- On-Chip USB Transceiver with On-Chip Voltage Regulator
- Interface for Off-Chip USB Transceiver
- Streaming Parallel Port (SPP) for USB streaming transfers (40/44-pin devices only)

Power-Managed Modes:

- Run: CPU on, peripherals on
- Idle: CPU off, peripherals on
- Sleep: CPU off, peripherals off
- Idle mode currents down to 5.8 µA typical
- Sleep mode currents down to 0.1 µA typical
- Timer1 Oscillator: 1.1 µA typical, 32 kHz, 2V
- Watchdog Timer: 2.1 µA typical
- Two-Speed Oscillator Start-up

Flexible Oscillator Structure:

- Four Crystal modes, including High Precision PLL for USB
- Two External Clock modes, up to 48 MHz
- Internal Oscillator Block:
  - 8 user-selectable frequencies, from 31 kHz to 8 MHz
  - User-tunable to compensate for frequency drift
- Secondary Oscillator using Timer1 @ 32 kHz
- Dual Oscillator options allow microcontroller and USB module to run at different clock speeds
- Fail-Safe Clock Monitor:
  - Allows for safe shutdown if any clock stops

Peripheral Highlights:

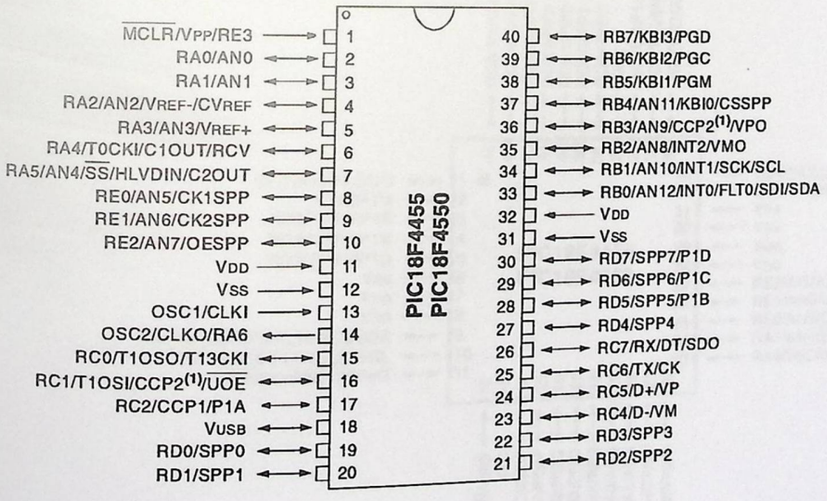
- High-Current Sink/Source: 25 mA/25 mA
- Three External Interrupts
- Four Timer modules (Timer0 to Timer3)
- Up to 2 Capture/Compare/PWM (CCP) modules:
  - Capture is 16-bit, max. resolution 5.2 ns (TCY/16)
  - Compare is 16-bit, max. resolution 83.3 ns (TCY)
  - PWM output: PWM resolution is 1 to 10-bit
- Enhanced Capture/Compare/PWM (ECCP) module:
  - Multiple output modes
  - Selectable polarity
  - Programmable dead time
  - Auto-shutdown and auto-restart
- Enhanced USART module:
  - LIN bus support
- Master Synchronous Serial Port (MSSP) module supporting 3-wire SPI (all 4 modes) and I²C™ Master and Slave modes
- 10-bit, up to 13-channel Analog-to-Digital Converter module (A/D) with Programmable Acquisition Time
- Dual Analog Comparators with Input Multiplexing

Special Microcontroller Features:

- C Compiler Optimized Architecture with optional Extended Instruction Set
- 100,000 Erase/Write Cycle Enhanced Flash Program Memory typical
- 1,000,000 Erase/Write Cycle Data EEPROM Memory typical
- Flash/Data EEPROM Retention: > 40 years
- Self-Programmable under Software Control
- Priority Levels for Interrupts
- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
  - Programmable period from 41 ms to 131s
- Programmable Code Protection
- Single-Supply 5V In-Circuit Serial Programming™ (ICSP™) via two pins
- In-Circuit Debug (ICD) via two pins
- Optional dedicated ICD/ICSP port (44-pin devices only)
- Wide Operating Voltage Range (2.0V to 5.5V)

Device	Program Memory		Data Memory		I/O	10-Bit A/D (ch)	CCP/ECCP (PWM)	SPP	MSSP		EAUSART	Comparators	Timers 8/16-Bit
	Flash (bytes)	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)					SPI	Master I²C™			
PIC18F2455	24K	12288	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F2550	32K	16384	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F4455	24K	12288	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3
PIC18F4550	32K	16384	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3

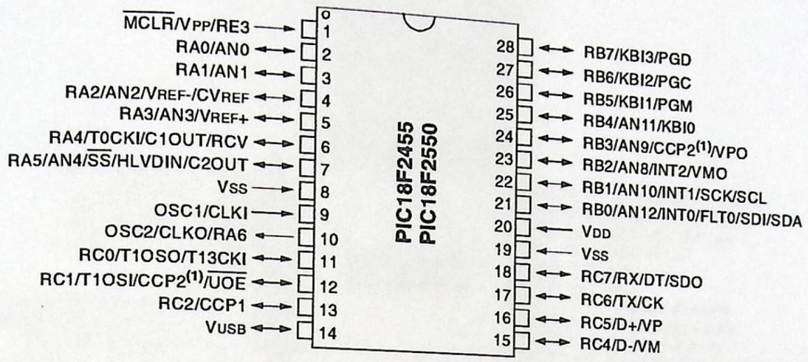
40-Pin PDIP



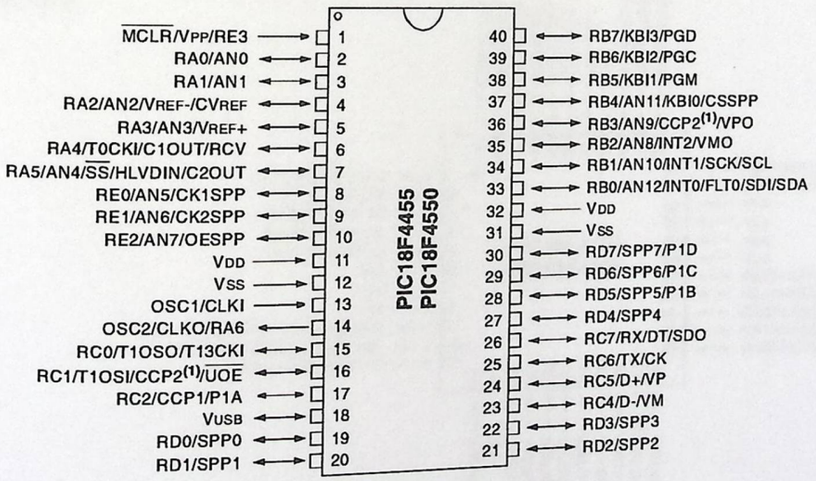
Note 1: RB3 is the alternate pin for CCP2 multiplexing.

Pin Diagrams

28-Pin PDIP, SOIC



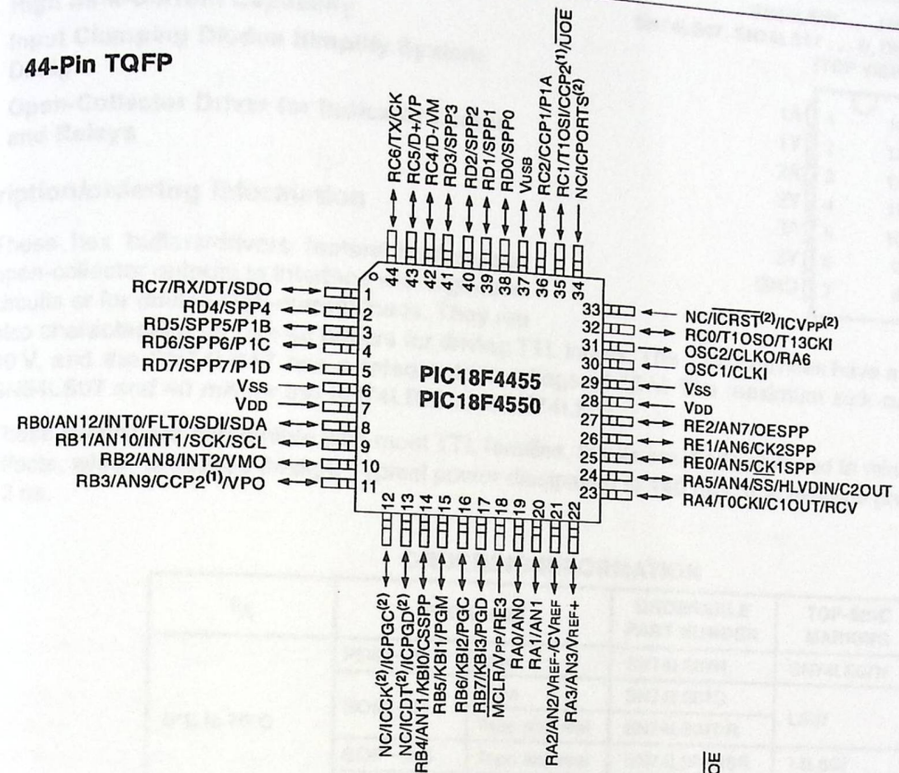
40-Pin PDIP



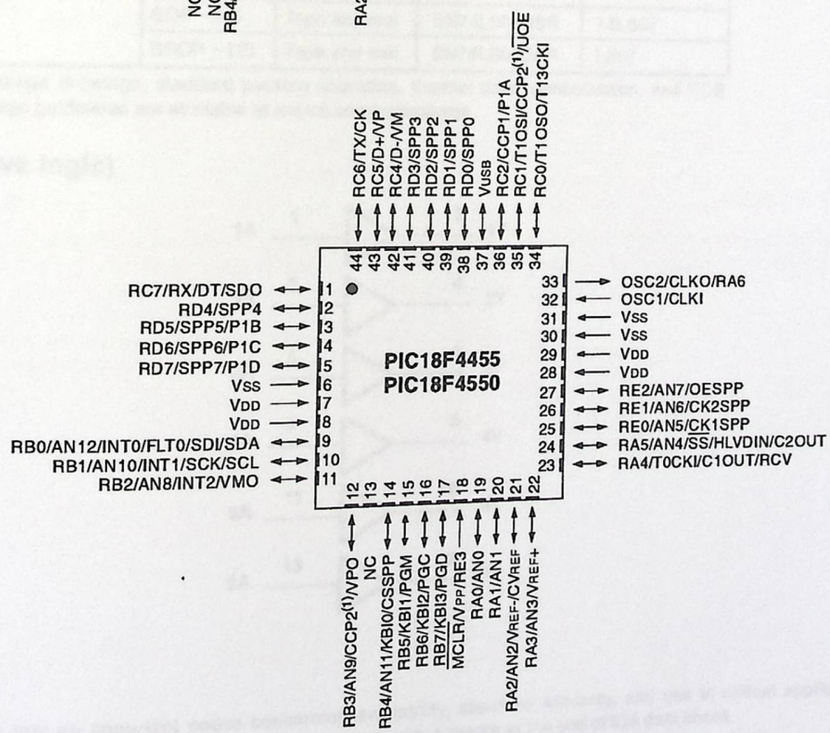
Note 1: RB3 is the alternate pin for CCP2 multiplexing.

Pin Diagrams (Continued)

44-Pin TQFP



44-Pin QFN



Note 1: RB3 is the alternate pin for CCP2 multiplexing.

2: Special ICPORTS features available in select circumstances. See **Section 25.9 "Special ICPORT Features (Designated Packages Only)"** for more information.

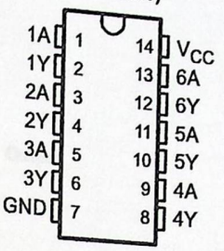
The SN54LS07 and SN74LS17 are obsolete and are no longer supplied.

SN54LS07, SN74LS07, SN74LS17
HEX BUFFERS/DRIVERS WITH OPEN-COLLECTOR HIGH-VOLTAGE OUTPUTS

SDLS021C – MAY 1990 – REVISED FEBRUARY 2004

- **Convert TTL Voltage Levels to MOS Levels**
- **High Sink-Current Capability**
- **Input Clamping Diodes Simplify System Design**
- **Open-Collector Driver for Indicator Lamps and Relays**

SN54LS07 . . . J PACKAGE
SN74LS07, SN74LS17 . . . D, DB, N, OR NS PACKAGE
(TOP VIEW)



description/ordering information

These hex buffers/drivers feature high-voltage open-collector outputs to interface with high-level circuits or for driving high-current loads. They are also characterized for use as buffers for driving TTL inputs. The 'LS07 devices have a rated output voltage of 30 V, and the SN74LS17 has a rated output voltage of 15 V. The maximum sink current is 30 mA for the SN54LS07 and 40 mA for the SN74LS07 and SN74LS17.

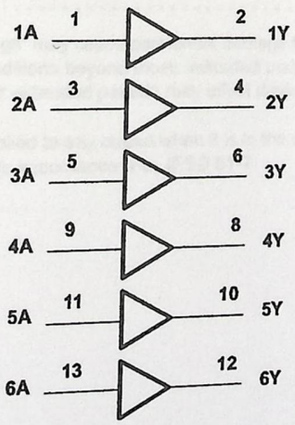
These circuits are compatible with most TTL families. Inputs are diode-clamped to minimize transmission-line effects, which simplifies design. Typical power dissipation is 140 mW, and average propagation delay time is 12 ns.

ORDERING INFORMATION

T_A	PACKAGE†		ORDERABLE PART NUMBER	TOP-SIDE MARKING
0°C to 70°C	PDIP – N	Tube	SN74LS07N	SN74LS07N
	SOIC – D	Tube	SN74LS07D	LS07
		Tape and reel	SN74LS07DR	
	SOP – NS	Tape and reel	SN74LS07NSR	74LS07
	SSOP – DB	Tape and reel	SN74LS07DBR	LS07

† Package drawings, standard packing quantities, thermal data, symbolization, and PCB design guidelines are available at www.ti.com/sc/package.

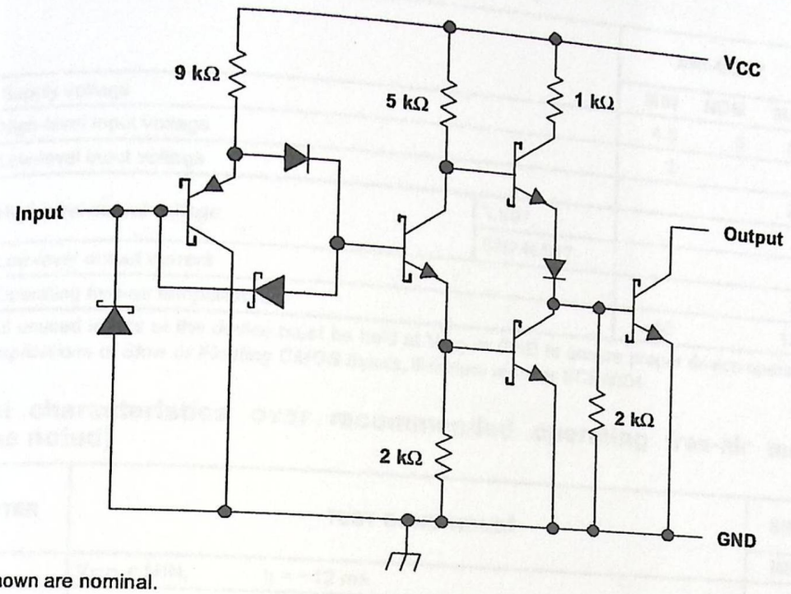
logic diagram (positive logic)



Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

The SN54LS07 and SN74LS17 are obsolete and are no longer supplied.

schematic (each gate)



Resistor values shown are nominal.

absolute maximum ratings over operating free-air temperature range (unless otherwise noted)†

Supply voltage, V_{CC} ..... 7 V
Input voltage, V_I (see Note 1) ..... 7 V
Output voltage, V_O (see Notes 1 and 2): SN54LS07, SN74LS07 ..... 30 V
SN74LS17 ..... 15 V
Package thermal impedance, θ_{JA} (see Note 3): D package ..... 86°C/W
DB package ..... 96°C/W
N package ..... 80°C/W
NS package ..... 76°C/W
Storage temperature range, T_{stg} ..... −65°C to 150°C

† Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTES: 1. All voltage values are with respect to GND.
2. This is the maximum voltage that should be applied to any output when it is in the off state.
3. The package thermal impedance is calculated in accordance with JESD 51-7.

TEXAS INSTRUMENTS
POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

recommended operating conditions (see Note 4)

			SN54LS07			SN74LS07 SN74LS17			UNIT
			MIN	NOM	MAX	MIN	NOM	MAX	
V_{CC}	Supply voltage		4.5	5	5.5	4.75	5	5.25	V
V_{IH}	High-level input voltage		2			2			V
V_{IL}	Low-level input voltage				0.8			0.8	V
V_{OH}	High-level output voltage	'LS07			30			30	V
		SN74LS17						15	
I_{OL}	Low-level output current				30			40	mA
T_A	Operating free-air temperature		−55	125		0		70	°C

NOTE 4: All unused inputs of the device must be held at V_{CC} or GND to ensure proper device operation. Refer to the TI application report, *Implications of Slow or Floating CMOS Inputs*, literature number SCBA004.

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER	TEST CONDITIONS‡			SN54LS07		SN74LS07 SN74LS17		UNIT
				MIN	MAX	MIN	MAX	
V_{IK}	V_{CC} = MIN,	I_I = −12 mA			−1.5		−1.5	V
I_{OH}	V_{CC} = MIN,	V_{IH} = 2 V	'LS07, V_{OH} = 30 V		0.25		0.25	mA
			SN74LS17, V_{OH} = 15 V				0.25	
V_{OL}	V_{CC} = MIN,	V_{IL} = 0.8 V	I_{OL} = 16 mA		0.4		0.4	V
			I_{OL} = MAX§		0.7		0.7	
I_I	V_{CC} = MAX,	V_I = 7 V			1		1	mA
I_{IH}	V_{CC} = MAX,	V_I = 2.4 V			20		20	µA
I_{IL}	V_{CC} = MAX,	V_I = 0.4 V			−0.2		−0.2	mA
I_{CCH}	V_{CC} = MAX				14		14	mA
I_{CCL}	V_{CC} = MAX				45		45	mA

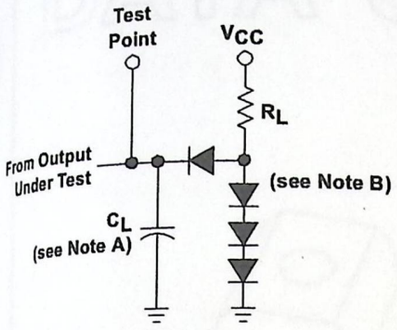
‡For conditions shown as MIN or MAX, use the appropriate value specified under *recommended operating conditions*.
§I_{OL} = 30 mA for SN54 series parts and 40 mA for SN74 series parts.

switching characteristics, V_{CC} = 5 V, T_A = 25°C (see Figure 1)

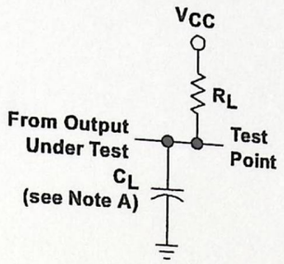
PARAMETER	FROM (INPUT)	TO (OUTPUT)	TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{PLH}	A	Y	R_L = 110 Ω, C_L = 15 pF		6	10	ns
t_{PHL}					19	30	

The SN54LS07 and SN74LS17 are obsolete and are no longer supplied.

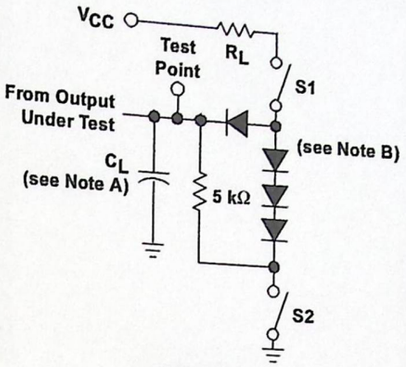
PARAMETER MEASUREMENT INFORMATION



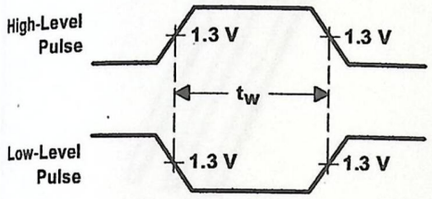
LOAD CIRCUIT
FOR 2-STATE TOTEM-POLE OUTPUTS



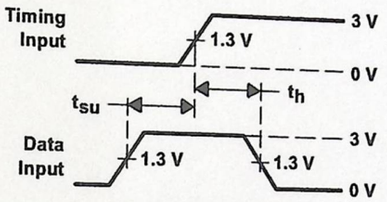
LOAD CIRCUIT
FOR OPEN-COLLECTOR OUTPUTS



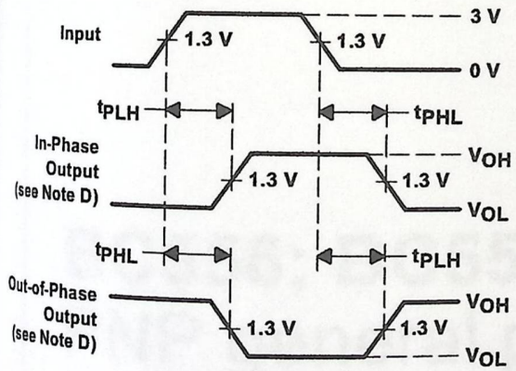
LOAD CIRCUIT
FOR 3-STATE OUTPUTS



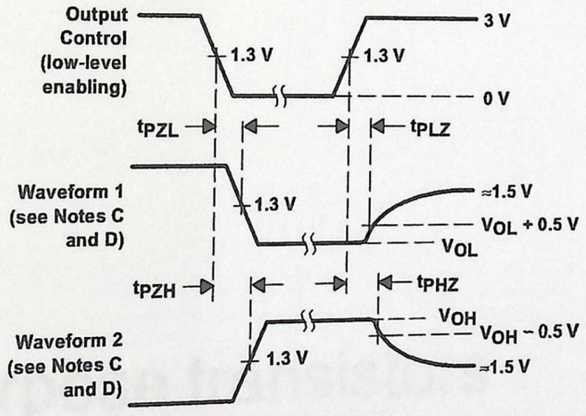
VOLTAGE WAVEFORMS
PULSE DURATIONS



VOLTAGE WAVEFORMS
SETUP AND HOLD TIMES



VOLTAGE WAVEFORMS
PROPAGATION DELAY TIMES



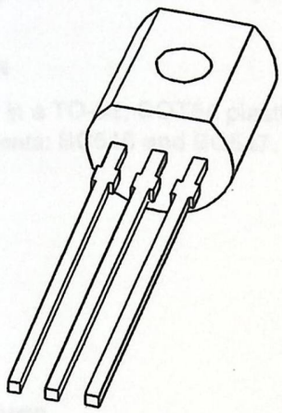
VOLTAGE WAVEFORMS
ENABLE AND DISABLE TIMES, 3-STATE OUTPUTS

NOTES: A. C_L includes probe and jig capacitance.
B. All diodes are 1N3064 or equivalent.
C. Waveform 1 is for an output with internal conditions such that the output is low, except when disabled by the output control. Waveform 2 is for an output with internal conditions such that the output is high, except when disabled by the output control.
D. S1 and S2 are closed for t_{PLH}, t_{PHL}, t_{PHZ}, and t_{PLZ}; S1 is open and S2 is closed for t_{PZH}; S1 is closed and S2 is open for t_{PZL}.
E. Phase relationships between inputs and outputs have been chosen arbitrarily for these examples.
F. All input pulses are supplied by generators having the following characteristics: PRR ≤ 1 MHz, $Z_O \approx 50\ \Omega$, $t_r \leq 1.5$ ns, $t_f \leq 2.6$ ns.
G. The outputs are measured one at a time, with one input transition per measurement.

Figure 1. Load Circuits and Voltage Waveforms

TEXAS INSTRUMENTS
POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

DATA SHEET



BC556; BC557
PNP general purpose transistors

1999 Apr 15

Product specification
Supersedes data of 1997 Mar 27

Philips
Semiconductors

PHILIPS

PNP general purpose transistors

BC556; BC557

FEATURES

- Low current (max. 100 mA)
- Low voltage (max. 65 V).

APPLICATIONS

- General purpose switching and amplification.

DESCRIPTION

PNP transistor in a TO-92; SOT54 plastic package.
NPN complements: BC546 and BC547.

PINNING

PIN	DESCRIPTION
1	emitter
2	base
3	collector

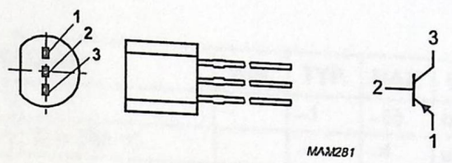


Fig.1 Simplified outline (TO-92; SOT54) and symbol.

LIMITING VALUES

In accordance with the Absolute Maximum Rating System (IEC 134).

SYMBOL	PARAMETER	CONDITIONS	MIN.	MAX.	UNIT
V_{CBO}	collector-base voltage	open emitter			
	BC556		–	–80	V
	BC557		–	–50	V
V_{CEO}	collector-emitter voltage	open base			
	BC556		–	–65	V
	BC557		–	–45	V
V_{EBO}	emitter-base voltage	open collector	–	–5	V
I_C	collector current (DC)		–	–100	mA
I_{CM}	peak collector current		–	–200	mA
I_{BM}	peak base current		–	–200	mA
P_{tot}	total power dissipation	$T_{amb} \leq 25$ °C	–	500	mW
T_{stg}	storage temperature		–65	+150	°C
T_j	junction temperature		–	150	°C
T_{amb}	operating ambient temperature		–65	+150	°C

THERMAL CHARACTERISTICS

SYMBOL	PARAMETER	CONDITIONS	VALUE	UNIT
$R_{th\ j\text{-}a}$	thermal resistance from junction to ambient	note 1	250	K/W

Note

1. Transistor mounted on an FR4 printed-circuit board.

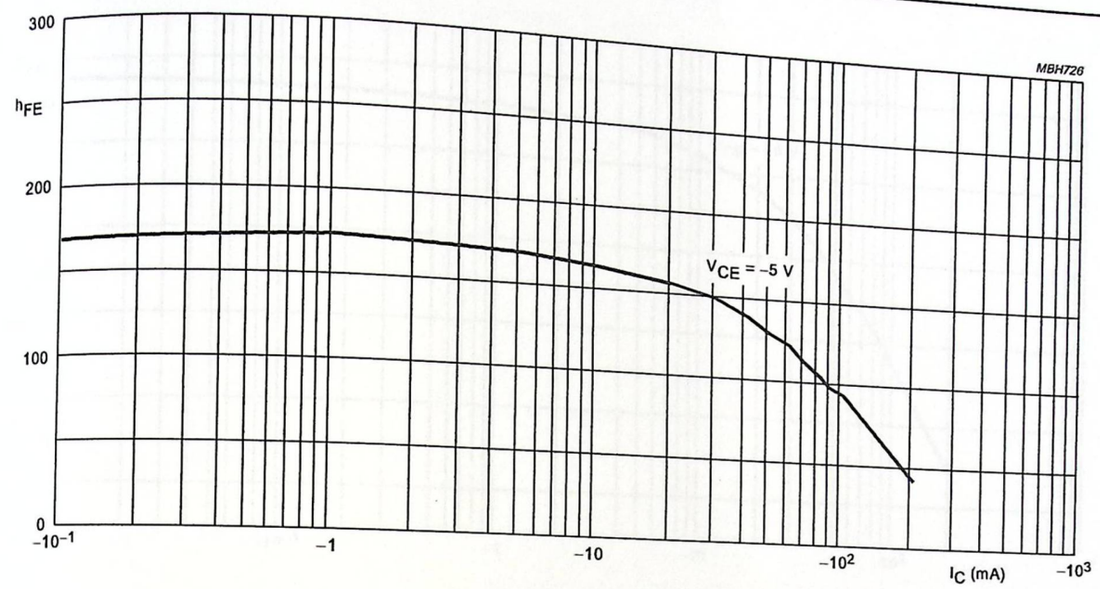
CHARACTERISTICS

T_j = 25 °C unless otherwise specified.

SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
I_{CBO}	collector cut-off current	I_E = 0; V_{CB} = –30 V	–	–1	–15	nA
		I_E = 0; V_{CB} = –30 V; T_j = 150 °C	–	–	–4	µA
I_{EBO}	emitter cut-off current	I_C = 0; V_{EB} = –5 V	–	–	–100	nA
h_{FE}	DC current gain	I_C = –2 mA; V_{CE} = –5 V; see Figs 2, 3 and 4				
	BC556		125	–	475	
	BC557		125	–	800	
	BC556A		125	–	250	
	BC556B; BC557B		220	–	475	
	BC557C		420	–	800	
V_{CEsat}	collector-emitter saturation voltage	I_C = –10 mA; I_B = –0.5 mA	–	–60	–300	mV
		I_C = –100 mA; I_B = –5 mA	–	–180	–650	mV
V_{BEsat}	base-emitter saturation voltage	I_C = –10 mA; I_B = –0.5 mA; note 1	–	–750	–	mV
		I_C = –100 mA; I_B = –5 mA; note 1	–	–930	–	mV
V_{BE}	base-emitter voltage	I_C = –2 mA; V_{CE} = –5 V; note 2	–600	–650	–750	mV
		I_C = –10 mA; V_{CE} = –5 V; note 2	–	–	–820	mV
C_c	collector capacitance	$I_E = i_e$ = 0; V_{CB} = –10 V; f = 1 MHz	–	3	–	pF
C_e	emitter capacitance	$I_C = i_c$ = 0; V_{EB} = –0.5 V; f = 1 MHz	–	10	–	pF
f_T	transition frequency	I_C = –10 mA; V_{CE} = –5 V; f = 100 MHz	100	–	–	MHz
F	noise figure	I_C = –200 µA; V_{CE} = –5 V; R_S = 2 kΩ; f = 1 kHz; B = 200 Hz	–	2	10	dB

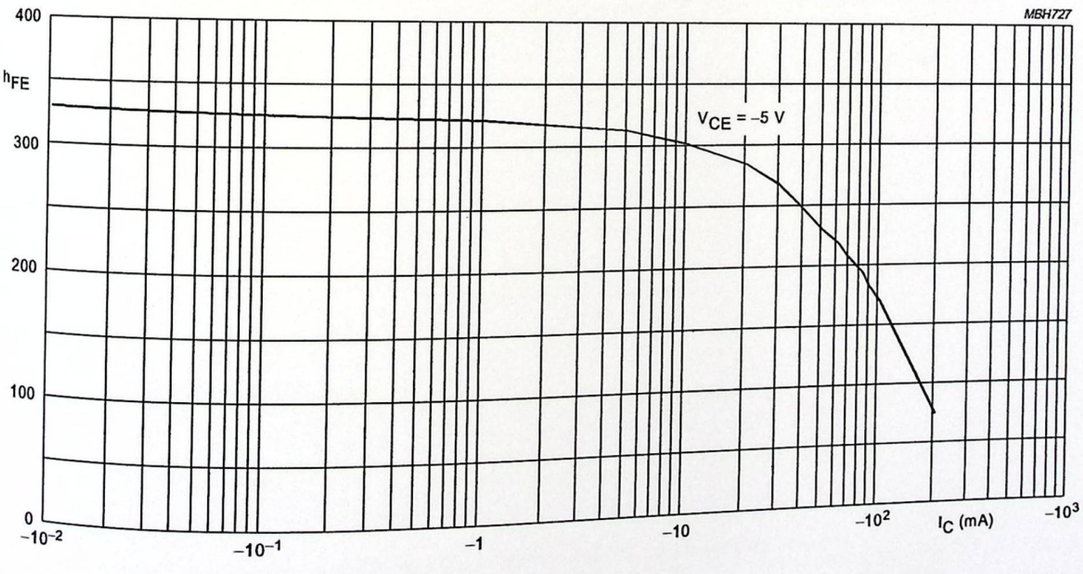
Notes

1. V_{BEsat} decreases by about –1.7 mV/K with increasing temperature.
2. V_{BE} decreases by about –2 mV/K with increasing temperature.



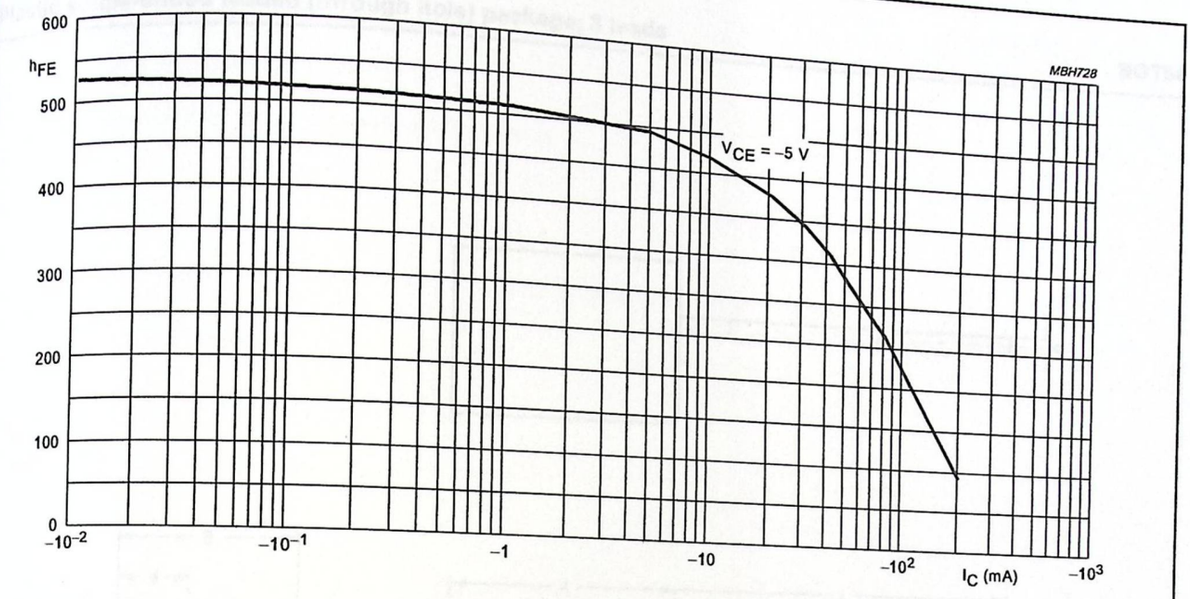
BC556A.

Fig.2 DC current gain; typical values.



BC556B; BC557B.

Fig.3 DC current gain; typical values.



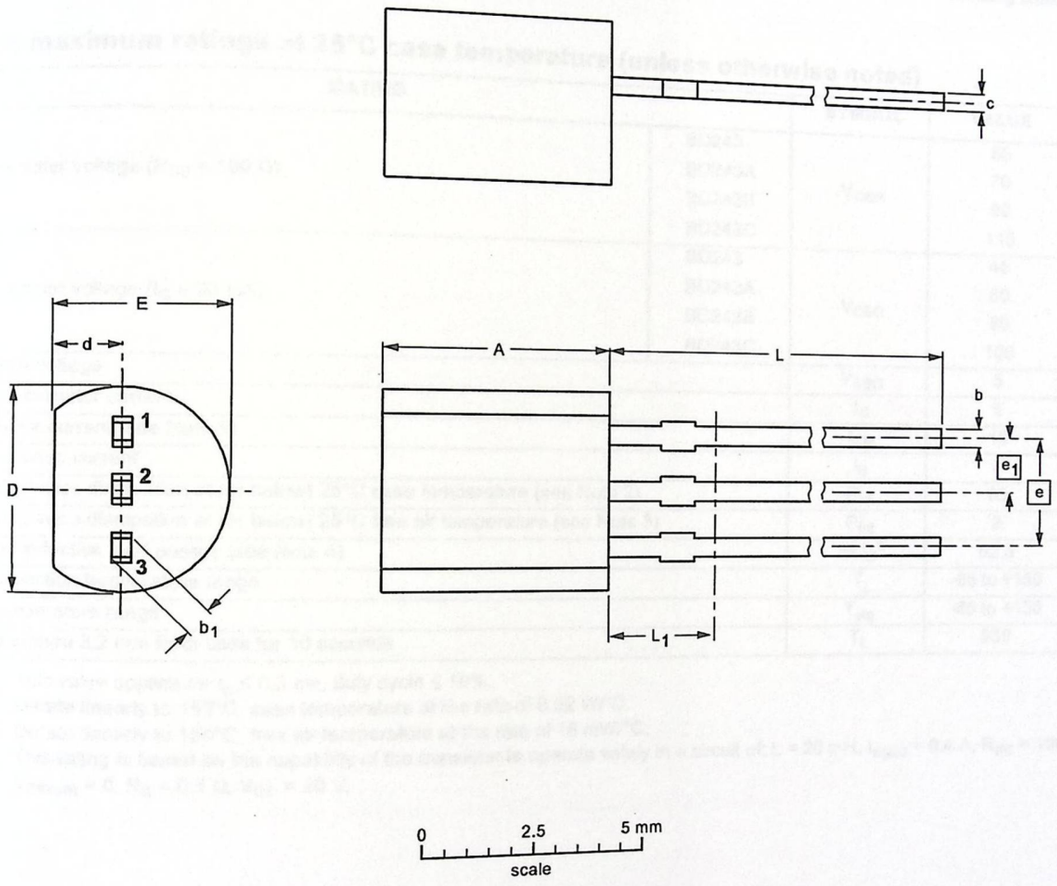
BC557C.

Fig.4 DC current gain; typical values.

PACKAGE OUTLINE

Plastic single-ended leaded (through hole) package; 3 leads

SOT54



DIMENSIONS (mm are the original dimensions)

UNIT	A	b	b_1	c	D	d	E	e	e_1	L	L_1 (1)
mm	5.2 5.0	0.48 0.40	0.66 0.56	0.45 0.40	4.8 4.4	1.7 1.4	4.2 3.6	2.54	1.27	14.5 12.7	2.5

Note

1. Terminal dimensions within this zone are uncontrolled to allow for flow of plastic and terminal irregularities.

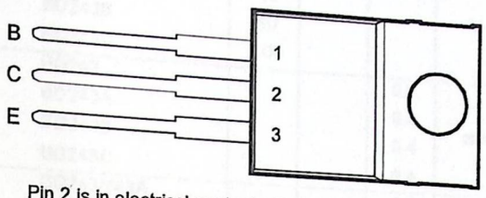
OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	EIAJ			
SOT54		TO-92	SC-43			97-02-28

BD243, BD243A, BD243B, BD243C
NPN SILICON POWER TRANSISTORS

JUNE 1973 - REVISED MARCH 1997

- **Designed for Complementary Use with the BD244 Series**
- **65 W at 25°C Case Temperature**
- **6 A Continuous Collector Current**
- **10 A Peak Collector Current**
- **Customer-Specified Selections Available**

TO-220 PACKAGE
(TOP VIEW)



Pin 2 is in electrical contact with the mounting base.

MDTRACA

absolute maximum ratings at 25°C case temperature (unless otherwise noted)

RATING		SYMBOL	VALUE	UNIT
Collector-emitter voltage (R_{BE} = 100 Ω)	BD243	V_{CER}	55	V
	BD243A		70	
	BD243B		90	
	BD243C		115	
Collector-emitter voltage (I_C = 30 mA)	BD243	V_{CEO}	45	V
	BD243A		60	
	BD243B		80	
	BD243C		100	
Emitter-base voltage		V_{EBO}	5	V
Continuous collector current		I_C	6	A
Peak collector current (see Note 1)		I_{CM}	10	A
Continuous base current		I_B	3	A
Continuous device dissipation at (or below) 25°C case temperature (see Note 2)		P_{tot}	65	W
Continuous device dissipation at (or below) 25°C free air temperature (see Note 3)		P_{tot}	2	W
Unclamped inductive load energy (see Note 4)		$½LI_C^2$	62.5	mJ
Operating junction temperature range		T_j	-65 to +150	°C
Storage temperature range		T_{stg}	-65 to +150	°C
Lead temperature 3.2 mm from case for 10 seconds		T_L	250	°C

NOTES: 1. This value applies for $t_p \leq 0.3$ ms, duty cycle ≤ 10%.
2. Derate linearly to 150°C case temperature at the rate of 0.52 W/°C.
3. Derate linearly to 150°C free air temperature at the rate of 16 mW/°C.
4. This rating is based on the capability of the transistor to operate safely in a circuit of: L = 20 mH, $I_{B(on)}$ = 0.4 A, R_{BE} = 100 Ω, $V_{BE(off)}$ = 0, R_S = 0.1 Ω, V_{CC} = 20 V.

JUNE 1973 - REVISED MARCH 1997

electrical characteristics at 25°C case temperature

PARAMETER		TEST CONDITIONS			MIN	TYP	MAX	UNIT
$V_{(BR)CEO}$	Collector-emitter breakdown voltage	I_C = 30 mA (see Note 5)	I_B = 0	BD243	45			V
				BD243A	60			
				BD243B	80			
				BD243C	100			
I_{CES}	Collector-emitter cut-off current	V_{CE} = 55 V	V_{BE} = 0	BD243			0.4	mA
		V_{CE} = 70 V	V_{BE} = 0	BD243A			0.4	
		V_{CE} = 90 V	V_{BE} = 0	BD243B			0.4	
		V_{CE} = 115 V	V_{BE} = 0	BD243C			0.4	
I_{CEO}	Collector cut-off current	V_{CE} = 30 V	I_B = 0	BD243/243A			0.7	mA
		V_{CE} = 60 V	I_B = 0	BD243B/243C			0.7	
I_{EBO}	Emitter cut-off current	V_{EB} = 5 V	I_C = 0				1	mA
h_{FE}	Forward current transfer ratio	V_{CE} = 4 V	I_C = 0.3 A					
		V_{CE} = 4 V	I_C = 3 A	(see Notes 5 and 6)	30			
					15			
$V_{CE(sat)}$	Collector-emitter saturation voltage	I_B = 1 A	I_C = 6 A	(see Notes 5 and 6)			1.5	V
V_{BE}	Base-emitter voltage	V_{CE} = 4 V	I_C = 6 A	(see Notes 5 and 6)			2	V
h_{fe}	Small signal forward current transfer ratio	V_{CE} = 10 V	I_C = 0.5 A	f = 1 kHz	20			
$\|h_{fe}\|$	Small signal forward current transfer ratio	V_{CE} = 10 V	I_C = 0.5 A	f = 1 MHz	3			

NOTES: 5. These parameters must be measured using pulse techniques, t_p = 300 µs, duty cycle ≤ 2%.
6. These parameters must be measured using voltage-sensing contacts, separate from the current carrying contacts.

thermal characteristics

PARAMETER		MIN	TYP	MAX	UNIT
$R_{\theta JC}$	Junction to case thermal resistance			1.92	°C/W
$R_{\theta JA}$	Junction to free air thermal resistance			62.5	°C/W

resistive-load-switching characteristics at 25°C case temperature

PARAMETER		TEST CONDITIONS †			MIN	TYP	MAX	UNIT
t_{on}	Turn-on time	I_C = 1 A	$I_{B(on)}$ = 0.1 A	$I_{B(off)}$ = -0.1 A		0.3		µs
t_{off}	Turn-off time	$V_{BE(off)}$ = -3.7 V	R_L = 20 Ω	t_p = 20 µs, dc ≤ 2%		1		µs

† Voltage and current values shown are nominal; exact values vary slightly with transistor parameters.

TYPICAL CHARACTERISTICS

TYPICAL DC CURRENT GAIN vs COLLECTOR CURRENT

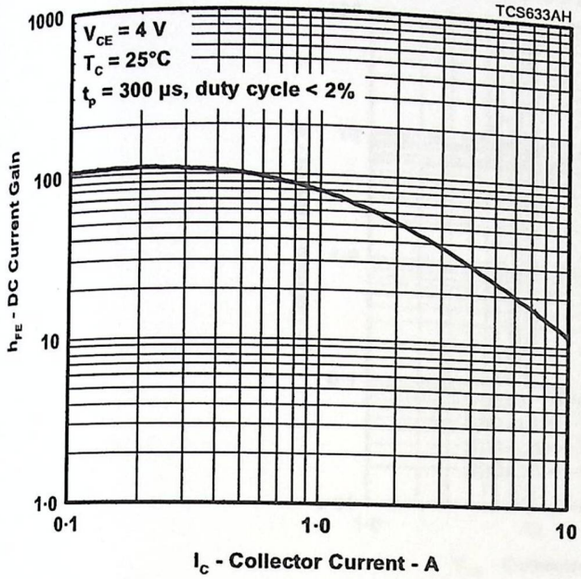


Figure 1.

COLLECTOR-EMITTER SATURATION VOLTAGE vs BASE CURRENT

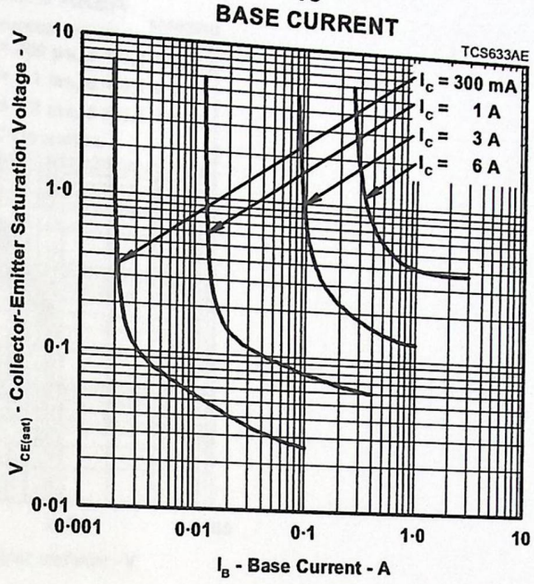


Figure 2.

BASE-EMITTER VOLTAGE vs COLLECTOR CURRENT

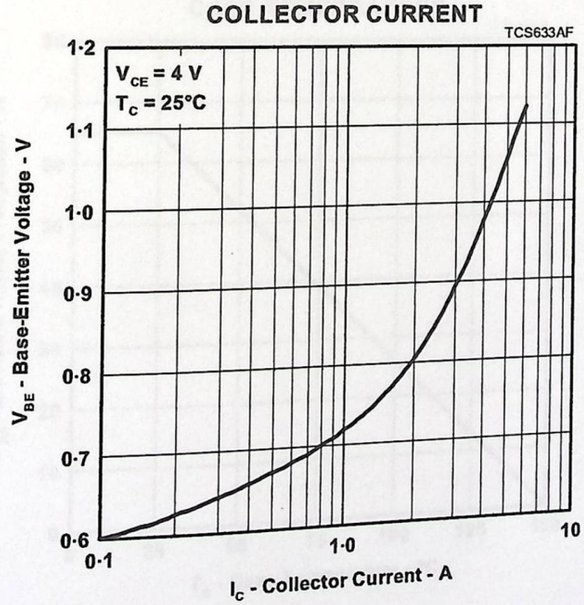


Figure 3.

MAXIMUM SAFE OPERATING REGIONS

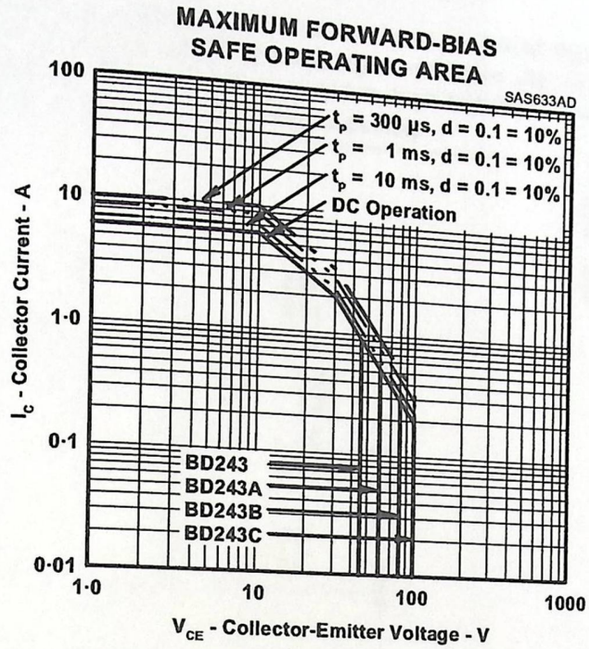


Figure 4.

THERMAL INFORMATION

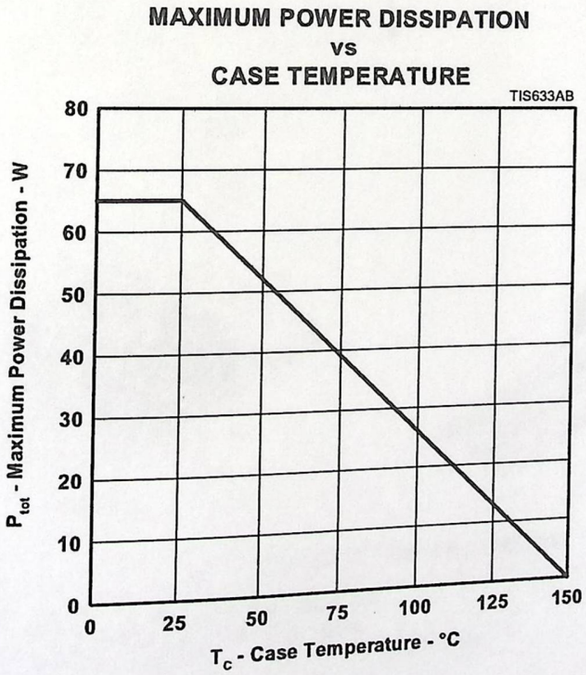


Figure 5.

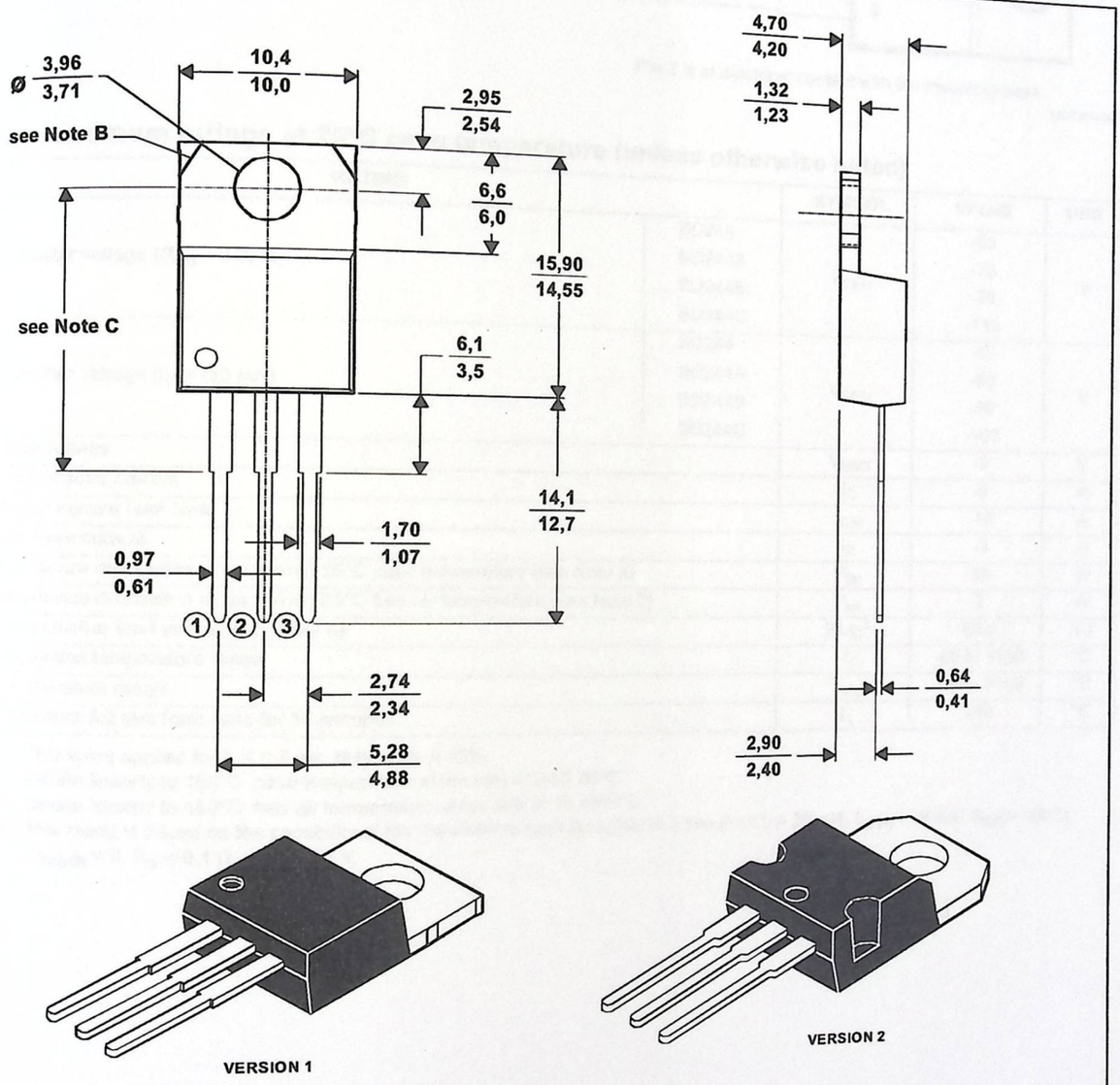
JUNE 1973 - REVISED MARCH 1997

MECHANICAL DATA

TO-220
3-pin plastic flange-mount package

This single-in-line package consists of a circuit mounted on a lead frame and encapsulated within a plastic compound. The compound will withstand soldering temperature with no deformation, and circuit performance characteristics will remain stable when operated in high humidity conditions. Leads require no additional cleaning or processing when used in soldered assembly.

TO220



ALL LINEAR DIMENSIONS IN MILLIMETERS

MDXXBE

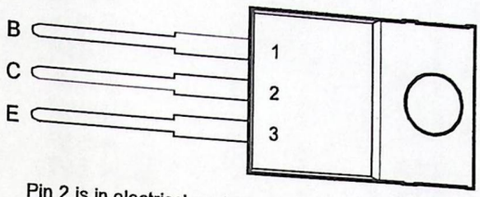
NOTES: A. The centre pin is in electrical contact with the mounting tab.
B. Mounting tab corner profile according to package version.
C. Typical fixing hole centre stand off height according to package version.
Version 1, 18.0 mm. Version 2, 17.6 mm.

BD244, BD244A, BD244B, BD244C
PNP SILICON POWER TRANSISTORS

JUNE 1973 - REVISED MARCH 1997

- **Designed for Complementary Use with the BD243 Series**
- **65 W at 25°C Case Temperature**
- **6 A Continuous Collector Current**
- **10 A Peak Collector Current**
- **Customer-Specified Selections Available**

TO-220 PACKAGE
(TOP VIEW)



Pin 2 is in electrical contact with the mounting base.

MDTRACA

absolute maximum ratings at 25°C case temperature (unless otherwise noted)

RATING		SYMBOL	VALUE	UNIT
Collector-emitter voltage (R_{BE} = 100 Ω)	BD244	V_{CER}	-55	V
	BD244A		-70	
	BD244B		-90	
	BD244C		-115	
Collector-emitter voltage (I_C = -30 mA)	BD244	V_{CEO}	-45	V
	BD244A		-60	
	BD244B		-80	
	BD244C		-100	
Emitter-base voltage		V_{EBO}	-5	V
Continuous collector current		I_C	-6	A
Peak collector current (see Note 1)		I_{CM}	-10	A
Continuous base current		I_B	-3	A
Continuous device dissipation at (or below) 25°C case temperature (see Note 2)		P_{tot}	65	W
Continuous device dissipation at (or below) 25°C free air temperature (see Note 3)		P_{tot}	2	W
Unclamped inductive load energy (see Note 4)		$½LI_C^2$	62.5	mJ
Operating junction temperature range		T_j	-65 to +150	°C
Storage temperature range		T_{stg}	-65 to +150	°C
Lead temperature 3.2 mm from case for 10 seconds		T_L	250	°C

NOTES: 1. This value applies for $t_p \leq 0.3$ ms, duty cycle ≤ 10%.
2. Derate linearly to 150°C case temperature at the rate of 0.52 W/°C.
3. Derate linearly to 150°C free air temperature at the rate of 16 mW/°C.
4. This rating is based on the capability of the transistor to operate safely in a circuit of: L = 20 mH, $I_{B(on)}$ = -0.4 A, R_{BE} = 100 Ω, $V_{BE(off)}$ = 0, R_S = 0.1 Ω, V_{CC} = -20 V.

JUNE 1973 - REVISED MARCH 1997

electrical characteristics at 25°C case temperature

PARAMETER		TEST CONDITIONS			MIN	TYP	MAX	UNIT
$V_{(BR)CEO}$	Collector-emitter breakdown voltage	I_C = -30 mA (see Note 5)	I_B = 0	BD244	-45			V
				BD244A	-60			
				BD244B	-80			
				BD244C	-100			
I_{CES}	Collector-emitter cut-off current	V_{CE} = -55 V	V_{BE} = 0	BD244			-0.4	mA
		V_{CE} = -70 V	V_{BE} = 0	BD244A			-0.4	
		V_{CE} = -90 V	V_{BE} = 0	BD244B			-0.4	
		V_{CE} = -115 V	V_{BE} = 0	BD244C			-0.4	
I_{CEO}	Collector cut-off current	V_{CE} = -30 V	I_B = 0	BD244/244A			-0.7	mA
		V_{CE} = -60 V	I_B = 0	BD244B/244C			-0.7	
I_{EBO}	Emitter cut-off current	V_{EB} = -5 V	I_C = 0				-1	mA
h_{FE}	Forward current transfer ratio	V_{CE} = -4 V	I_C = -0.3 A		30			
		V_{CE} = -4 V	I_C = -3 A	(see Notes 5 and 6)	15			
$V_{CE(sat)}$	Collector-emitter saturation voltage	I_B = -1 A	I_C = -6 A	(see Notes 5 and 6)			-1.5	V
V_{BE}	Base-emitter voltage	V_{CE} = -4 V	I_C = -6 A	(see Notes 5 and 6)			-2	V
h_{fe}	Small signal forward current transfer ratio	V_{CE} = -10 V	I_C = -0.5 A	f = 1 kHz	20			
$\|h_{fe}\|$	Small signal forward current transfer ratio	V_{CE} = -10 V	I_C = -0.5 A	f = 1 MHz	3			

NOTES: 5. These parameters must be measured using pulse techniques, t_p = 300 µs, duty cycle ≤ 2%.
6. These parameters must be measured using voltage-sensing contacts, separate from the current carrying contacts.

thermal characteristics

PARAMETER		MIN	TYP	MAX	UNIT
$R_{\theta JC}$	Junction to case thermal resistance			1.92	°C/W
$R_{\theta JA}$	Junction to free air thermal resistance			62.5	°C/W

resistive-load-switching characteristics at 25°C case temperature

PARAMETER		TEST CONDITIONS †			MIN	TYP	MAX	UNIT
t_{on}	Turn-on time	I_C = -1 A	$I_{B(on)}$ = -0.1 A	$I_{B(off)}$ = 0.1 A		0.3		µs
t_{off}	Turn-off time	$V_{BE(off)}$ = 3.7 V	R_L = 20 Ω	t_p = 20 µs, dc ≤ 2%		1		µs

† Voltage and current values shown are nominal; exact values vary slightly with transistor parameters.

TYPICAL CHARACTERISTICS

TYPICAL DC CURRENT GAIN vs COLLECTOR CURRENT

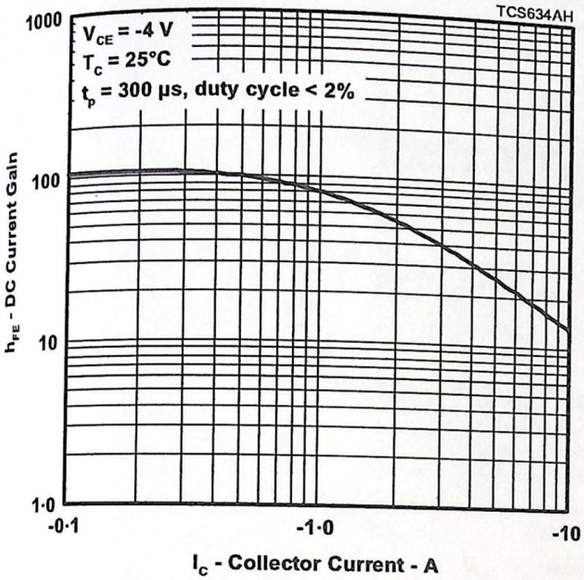


Figure 1.

COLLECTOR-EMITTER SATURATION VOLTAGE vs BASE CURRENT

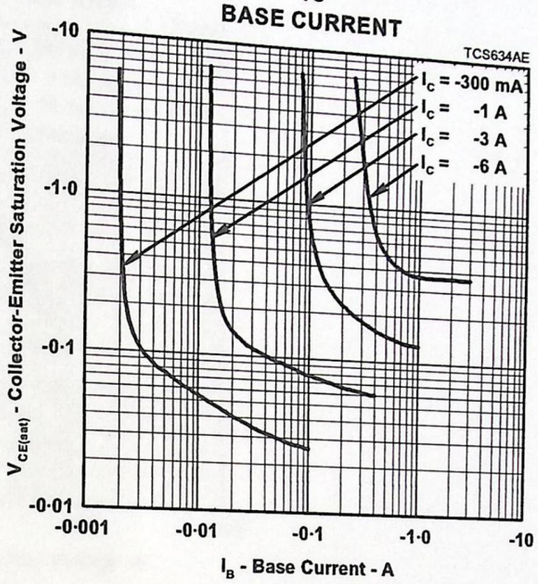


Figure 2.

BASE-EMITTER VOLTAGE vs COLLECTOR CURRENT

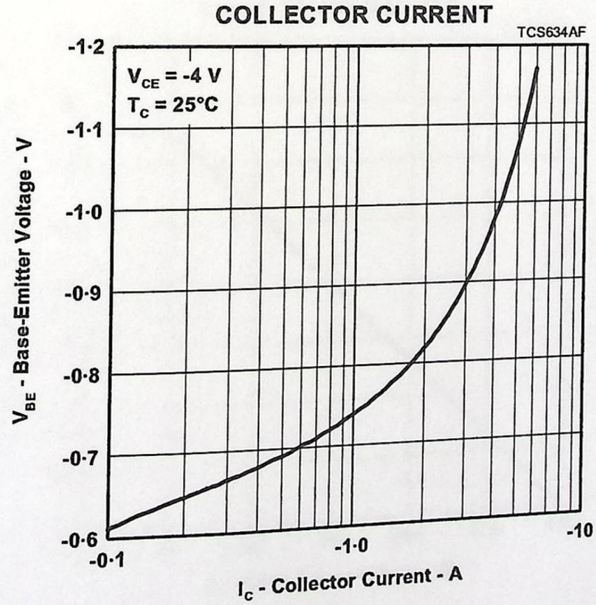


Figure 3.

MAXIMUM SAFE OPERATING REGIONS

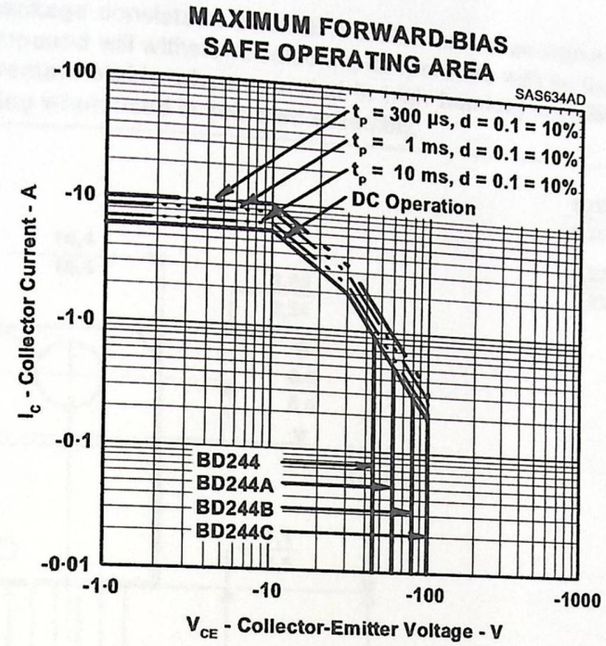


Figure 4.

THERMAL INFORMATION

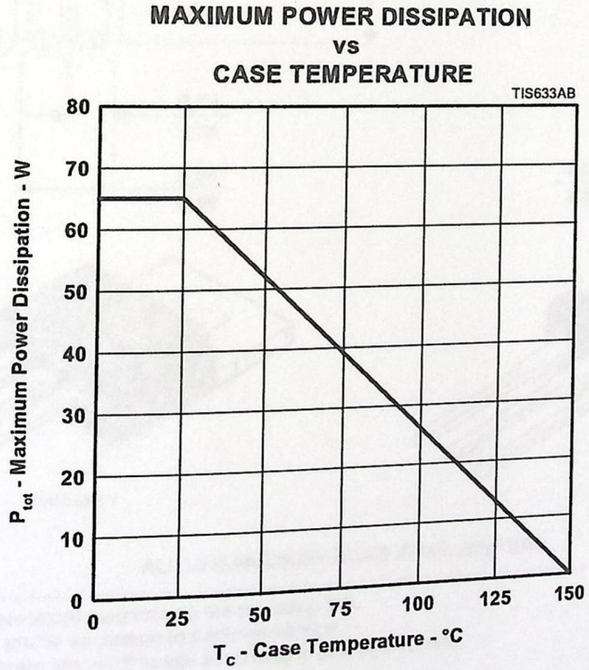


Figure 5.

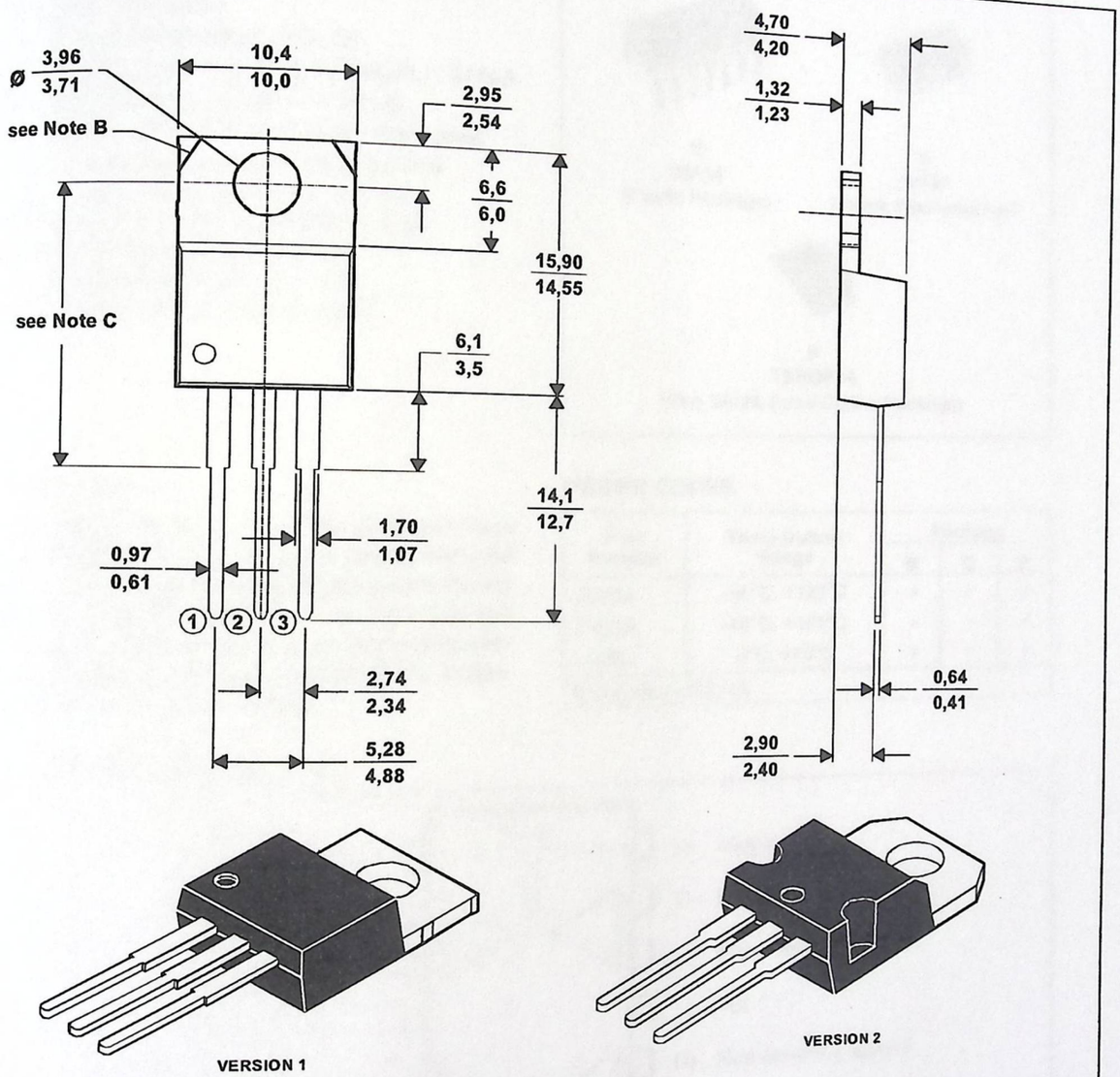
JUNE 1973 - REVISED MARCH 1997

MECHANICAL DATA

TO-220
3-pin plastic flange-mount package

This single-in-line package consists of a circuit mounted on a lead frame and encapsulated within a plastic compound. The compound will withstand soldering temperature with no deformation, and circuit performance characteristics will remain stable when operated in high humidity conditions. Leads require no additional cleaning or processing when used in soldered assembly.

TO220



ALL LINEAR DIMENSIONS IN MILLIMETERS

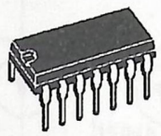
MDXXBE

NOTES: A. The centre pin is in electrical contact with the mounting tab.
B. Mounting tab corner profile according to package version.
C. Typical fixing hole centre stand off height according to package version.
Version 1, 18.0 mm. Version 2, 17.6 mm.

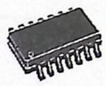
LM124
LM224 - LM324

LOW POWER QUAD OPERATIONAL AMPLIFIERS

- WIDE GAIN BANDWIDTH : 1.3MHz
- INPUT COMMON-MODE VOLTAGE RANGE INCLUDES GROUND
- LARGE VOLTAGE GAIN : 100dB
- VERY LOW SUPPLY CURRENT/AMPLI : 375µA
- LOW INPUT BIAS CURRENT : 20nA
- LOW INPUT OFFSET VOLTAGE : 5mV max.
  (for more accurate applications, use the equivalent parts LM124A-LM224A-LM324A which feature 3mV max)
- LOW INPUT OFFSET CURRENT : 2nA
- WIDE POWER SUPPLY RANGE :
  SINGLE SUPPLY : +3V TO +30V
  DUAL SUPPLIES : ±1.5V TO ±15V



N
DIP14
(Plastic Package)



D
SO14
(Plastic Micropackage)

P
TSSOP14
(Thin Shrink Small Outline Package)

DESCRIPTION

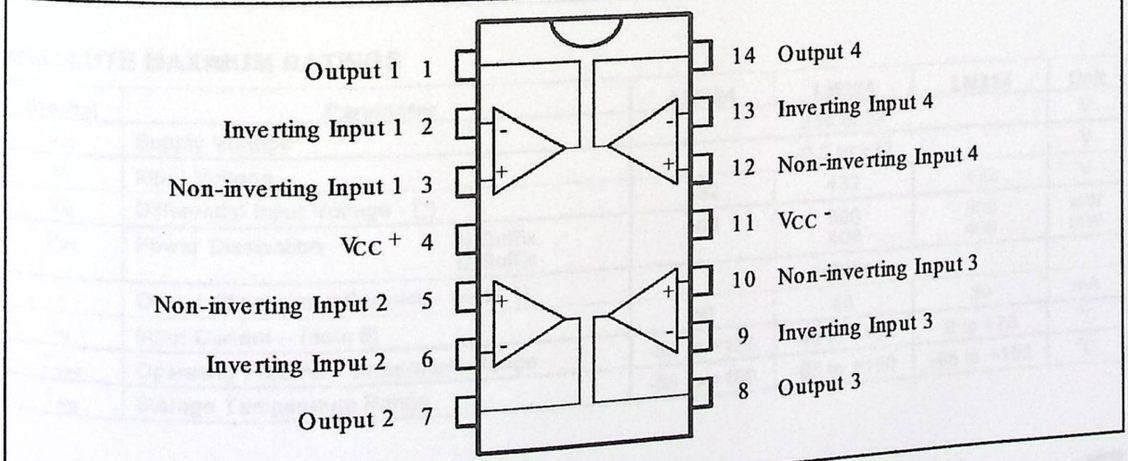
These circuits consist of four independent, high gain, internally frequency compensated operational amplifiers . They operate from a single power supply over a wide range of voltages. Operation from split power supplies is also possible and the low power supply current drain is independent of the magnitude of the power supply voltage.

ORDER CODES

Part Number	Temperature Range	Package		
		N	D	P
LM124	-55°C, +125°C	•	•	•
LM224	-40°C, +105°C	•	•	•
LM324	0°C, +70°C	•	•	•

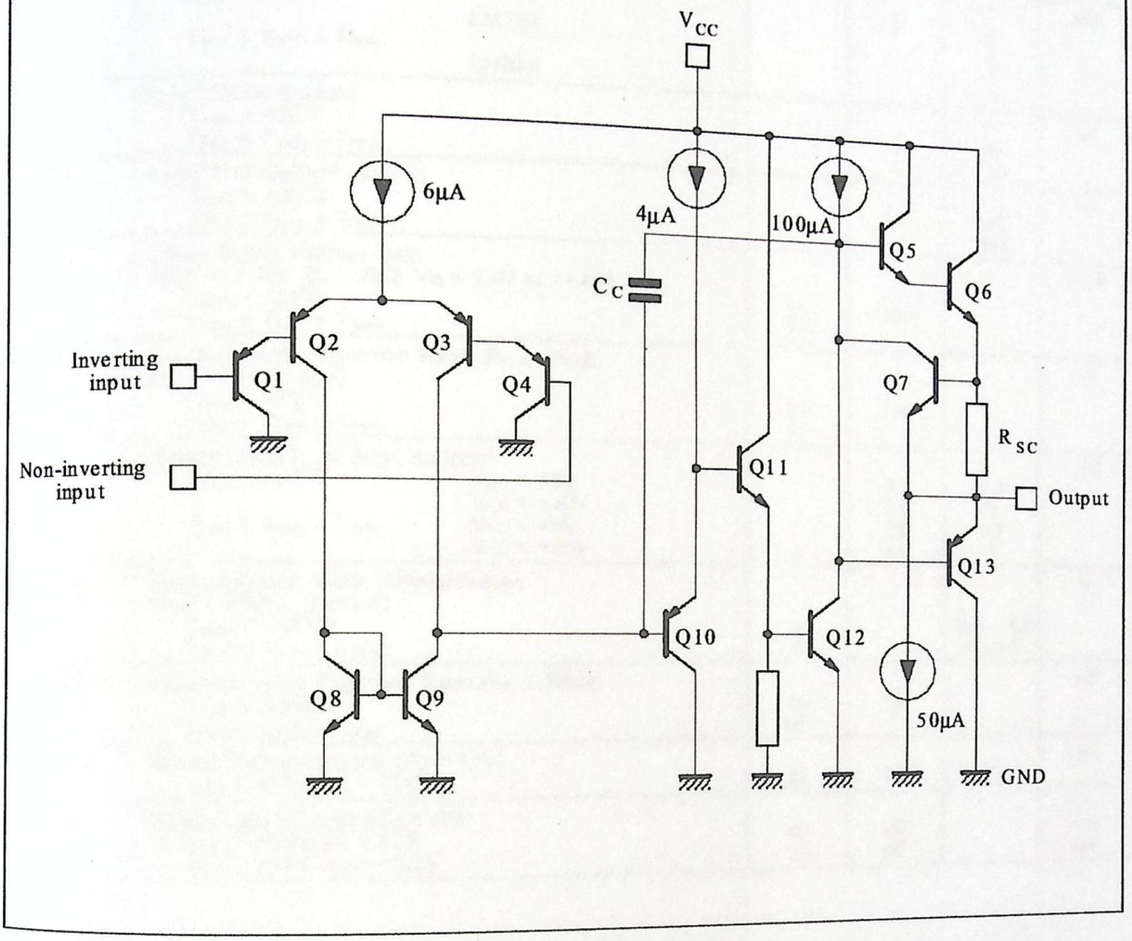
Example : LM224N

PIN CONNECTIONS (top view)



June 1999

SCHEMATIC DIAGRAM (1/4 LM124)



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	LM124	LM224	LM324	Unit
V_{cc}	Supply Voltage	±16 or 32			V
V_i	Input Voltage	-0.3 to +32			V
V_{id}	Differential Input Voltage - (*)	+32	+32	+32	V
P_{tot}	Power Dissipation N Suffix D Suffix	500 -	500 400	500 400	mW mW
-	Output Short-circuit Duration - (note 1)	Infinite			
I_{in}	Input Current – (note 6)	50	50	50	mA
T_{oper}	Operating Free Air Temperature Range	-55 to +125	-40 to +105	0 to +70	°C
T_{stg}	Storage Temperature Range	-65 to +150	-65 to +150	-65 to +150	°C

herwise specified)

124 - LM224 - LM324			Unit
	Typ.	Max.	
		5 7 7 9	mV
		30 100	nA
		150 300	nA
	100		V/mV
	110		dB
	0.7 1.5 0.8 1.5	1.2 3 1.2 3	mA
0 0		V_{CC} -1.5 V_{CC} -2	V
70 60	80		dB
20	40	70	mA
10 12	20 50		mA µA

ELECTRICAL CHARACTERISTICS

V_{CC}^+ = +5V, V_{CC}^- = Ground, V_O = 1.4V, T_{amb} = +25°C (unless otherwise specified)

Symbol	Parameter	LM124 - LM224 - LM324			Unit
		Min.	Typ.	Max.	
V_{io}	Input Offset Voltage (note 3) T_{amb} = +25°C $T_{min.} \le T_{amb} \le T_{max.}$ LM324 LM324		2	5 7 7 9	mV
I_{io}	Input Offset Current T_{amb} = +25°C $T_{min.} \le T_{amb} \le T_{max.}$		2	30 100	nA
I_{ib}	Input Bias Current (note 2) T_{amb} = +25°C $T_{min.} \le T_{amb} \le T_{max.}$		20	150 300	nA
A_{vd}	Large Signal Voltage Gain (V_{CC}^+ = +15V, R_L = 2kΩ, V_O = 1.4V to 11.4V) T_{amb} = +25°C $T_{min.} \le T_{amb} \le T_{max.}$	50 25	100		V/mV
SVR	Supply Voltage Rejection Ratio ($R_S \le 10k\Omega$) (V_{CC}^+ = 5V to 30V) T_{amb} = +25°C $T_{min.} \le T_{amb} \le T_{max.}$	65 65	110		dB
I_{CC}	Supply Current, all Amp, no load T_{amb} = +25°C V_{CC} = +5V V_{CC} = +30V $T_{min.} \le T_{amb} \le T_{max.}$ V_{CC} = +5V V_{CC} = +30V		0.7 1.5 0.8 1.5	1.2 3 1.2 3	mA
V_{icm}	Input Common Mode Voltage Range (V_{CC} = +30V) - (note 4) T_{amb} = +25°C $T_{min.} \le T_{amb} \le T_{max.}$	0 0		V_{CC} -1.5 V_{CC} -2	V
CMR	Common-mode Rejection Ratio ($R_S \le 10k\Omega$) T_{amb} = +25°C $T_{min.} \le T_{amb} \le T_{max}$	70 60	80		dB
I_{source}	Output Current Source (V_{id} = +1V) V_{CC} = +15V, V_o = +2V	20	40	70	mA
I_{sink}	Output Sink Current (V_{id} = -1V) V_{CC} = +15V, V_o = +2V V_{CC} = +15V, V_o = +0.2V	10 12	20 50		mA µA

ELECTRICAL CHARACTERISTICS (continued)

Symbol	Parameter	LM124 - LM224 - LM324			Unit
		Min.	Typ.	Max.	
V_{OH}	High Level Output Voltage (V_{CC} = +30V) T_{amb} = +25°C (R_L = 2kΩ) $T_{min.} \le T_{amb} \le T_{max.}$ (R_L = 2kΩ) T_{amb} = +25°C (R_L = 10kΩ) $T_{min.} \le T_{amb} \le T_{max.}$ (R_L = 10kΩ) (V_{CC} = +5V, R_L = 2kΩ) T_{amb} = +25°C $T_{min.} \le T_{amb} \le T_{max.}$	 26 26 27 27 3.5 3	 27 28		V
V_{OL}	Low Level Output Voltage (R_L = 10kΩ) T_{amb} = +25°C $T_{min.} \le T_{amb} \le T_{max.}$		5	20 20	mV
SR	Slew Rate V_{CC} = 15V, V_I = 0.5 to 3V, R_L = 2kΩ, C_L = 100pF, unity gain)		0.4		V/µs
GBP	Gain Bandwidth Product V_{CC} = 30V, f = 100kHz, V_{in} = 10mV R_L = 2kΩ, C_L = 100pF		1.3		MHz
THD	Total Harmonic Distortion f = 1kHz, A_V = 20dB, R_L = 2kΩ, V_O = $2V_{pp}$ C_L = 100pF, V_{CC} = 30V		0.015		%
e_n	Equivalent Input Noise Voltage f = 1kHz, R_s = 100Ω, V_{CC} = 30V		40		$\frac{nV}{\sqrt{Hz}}$
DV_{io}	Input Offset Voltage Drift		7	30	µV/°C
DI_{IO}	Input Offset Current Drift		10	200	pA/°C
V_{O1}/V_{O2}	Channel Separation (note 5) 1kHz ≤ f ≤ 20kHz		120		dB

Notes : 1. Short-circuits from the output to V_{CC} can cause excessive heating if V_{CC} > 15V. The maximum output current is approximately 40mA independent of the magnitude of V_{CC}. Destructive dissipation can result from simultaneous short-circuit on all amplifiers.

2. The direction of the input current is out of the IC. This current is essentially constant, independent of the state of the output so no loading change exists on the input lines.

3. V_o = 1.4V, R_s = 0Ω, 5V < V_{CC}^+ < 30V, 0 < V_{ic} < V_{CC}^+ - 1.5V

4. The input common-mode voltage of either input signal voltage should not be allowed to go negative by more than 0.3V. The upper end of the common-mode voltage range is V_{CC}^+ - 1.5V, but either or both inputs can go to +32V without damage.

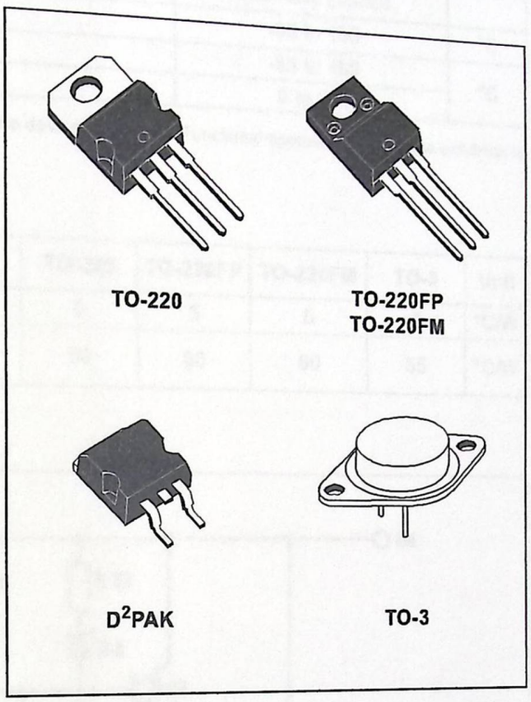
5. Due to the proximity of external components insure that coupling is not originating via stray capacitance between these external parts. This typically can be detected as this type of capacitance increases at higher frequences.

6. This input current only exists when the voltage at any of the input leads is driven negative. It is due to the collector-base junction of the input PNP transistor becoming forward biased and thereby acting as input diodes clamps. In addition to this diode action, there is also NPN parasitic action on the IC chip. this transistor action can cause the output voltages of the Op-amps to go to the V_{CC} voltage level (or to ground for a large overdrive) for the time duration than an input is driven negative.
This is not destructive and normal output will set up again for input voltage higher than -0.3V.

L7800 SERIES

POSITIVE VOLTAGE REGULATORS

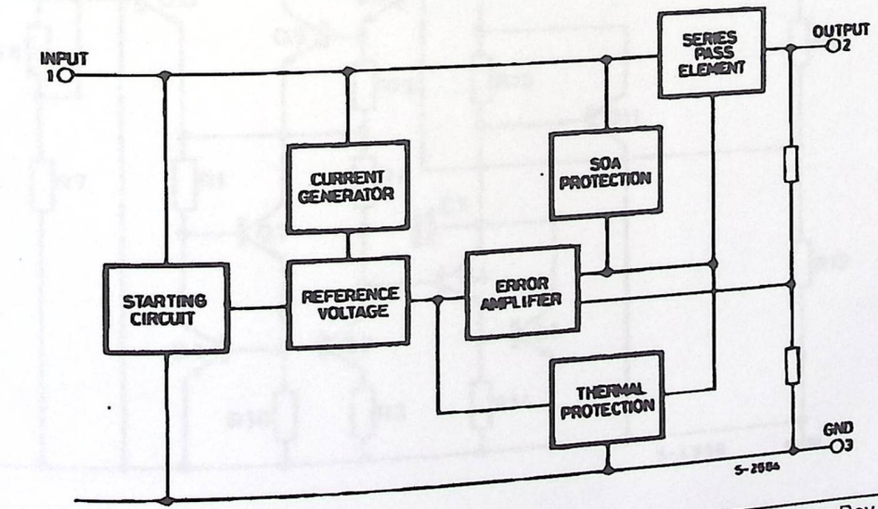
- OUTPUT CURRENT TO 1.5A
- OUTPUT VOLTAGES OF 5; 5.2; 6; 8; 8.5; 9; 10; 12; 15; 18; 24V
- THERMAL OVERLOAD PROTECTION
- SHORT CIRCUIT PROTECTION
- OUTPUT TRANSITION SOA PROTECTION



DESCRIPTION

The L7800 series of three-terminal positive regulators is available in TO-220, TO-220FP, TO-220FM, TO-3 and D^2PAK packages and several fixed output voltages, making it useful in a wide range of applications. These regulators can provide local on-card regulation, eliminating the distribution problems associated with single point regulation. Each type employs internal current limiting, thermal shut-down and safe area protection, making it essentially indestructible. If adequate heat sinking is provided, they can deliver over 1A output current. Although designed primarily as fixed voltage regulators, these devices can be used with external components to obtain adjustable voltage and currents.

Figure 1: Schematic Diagram



November 2004

Table 1: Absolute Maximum Ratings

Symbol	Parameter		Value	Unit
V_I	DC Input Voltage	for V_O= 5 to 18V	35	V
		for V_O= 20, 24V	40	
I_O	Output Current		Internally Limited	
P_{tot}	Power Dissipation		Internally Limited	
T_{stg}	Storage Temperature Range		-65 to 150	°C
T_{op}	Operating Junction Temperature Range	for L7800	-55 to 150	°C
		for L7800C	0 to 150	

Absolute Maximum Ratings are those values beyond which damage to the device may occur. Functional operation under these condition is not implied.

Table 2: Thermal Data

Symbol	Parameter	D^2PAK	TO-220	TO-220FP	TO-220FM	TO-3	Unit
$R_{thj\text{-}case}$	Thermal Resistance Junction-case Max	3	5	5	5	4	°C/W
$R_{thj\text{-}amb}$	Thermal Resistance Junction-ambient Max	62.5	50	60	60	35	°C/W

Figure 2: Schematic Diagram

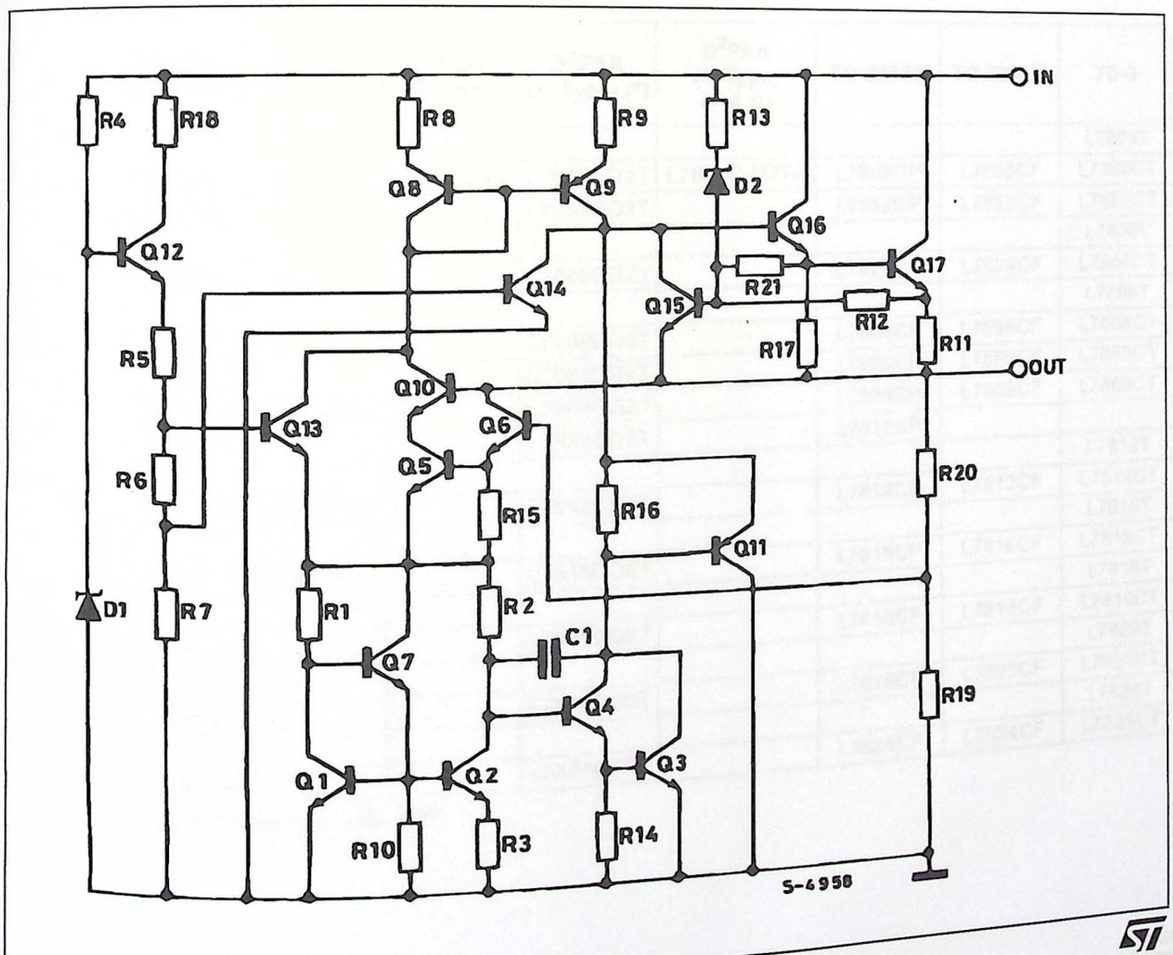


Figure 3: Connection Diagram (top view)

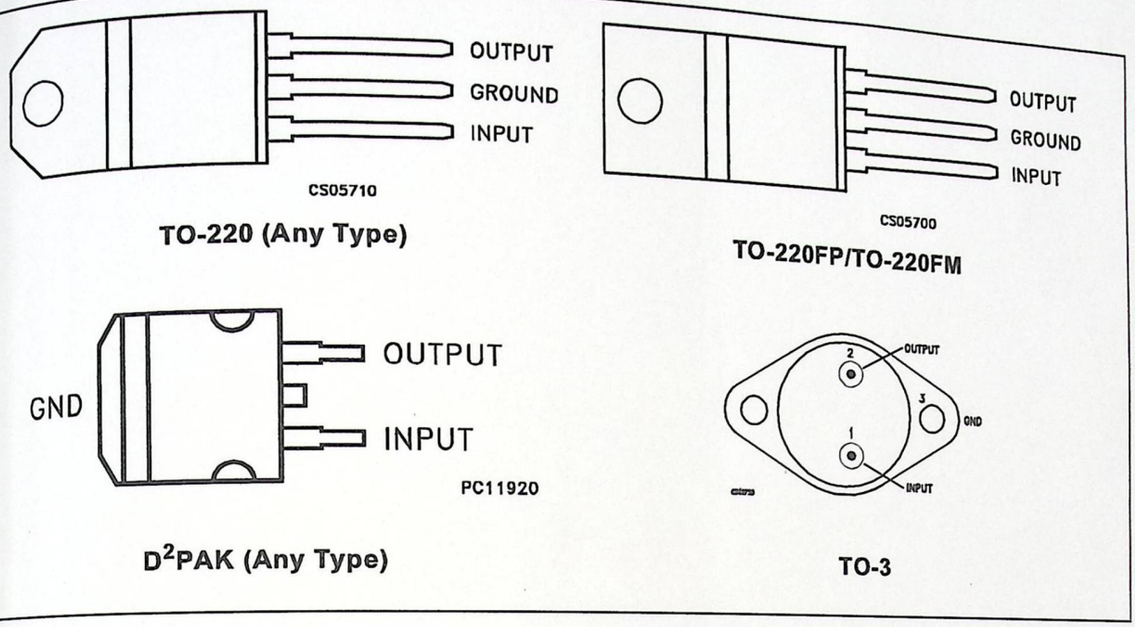


Table 3: Order Codes

TYPE	TO-220 (A Type)	TO-220 (C Type)	TO-220 (E Type)	D²PAK (A Type) (*)	D²PAK (C Type) (T & R)	TO-220FP	TO-220FM	TO-3
L7805								L7805T
L7805C	L7805CV	L7805C-V	L7805CV1	L7805CD2T	L7805C-D2TR	L7805CP	L7805CF	L7805CT
L7852C	L7852CV			L7852CD2T		L7852CP	L7852CF	L7852CT
L7806								L7806T
L7806C	L7806CV	L7806C-V		L7806CD2T		L7806CP	L7806CF	L7806CT
L7808								L7808T
L7808C	L7808CV	L7808C-V		L7808CD2T		L7808CP	L7808CF	L7808CT
L7885C	L7885CV			L7885CD2T		L7885CP	L7885CF	L7885CT
L7809C	L7809CV	L7809C-V		L7809CD2T		L7809CP	L7809CF	L7809CT
L7810C	L7810CV			L7810CD2T		L7810CP		
L7812								L7812T
L7812C	L7812CV	L7812C-V		L7812CD2T		L7812CP	L7812CF	L7812CT
L7815								L7815T
L7815C	L7815CV	L7815C-V		L7815CD2T		L7815CP	L7815CF	L7815CT
L7818								L7818T
L7818C	L7818CV			L7818CD2T		L7818CP	L7818CF	L7818CT
L7820								L7820T
L7820C	L7820CV			L7820CD2T		L7820CP	L7820CF	L7820CT
L7824								L7824T
L7824C	L7824CV			L7824CD2T		L7824CP	L7824CF	L7824CT

(*) Available in Tape & Reel with the suffix "-TR".