REMOTE CONTROL OF A MOVING VEHICLE

SUZALEY BIN SULAIMAN

This thesis is submitted as partial fulfillment of the requirements for the award of the Bachelor of Electrical Engineering (Hons.) (Electronics)

Faculty of Electrical & Electronics Engineering
Universiti Malaysia Pahang

NOVEMBER, 2008

DECLARATION

"All the trademark and copyrights use herein are property of their respective owner. References of information from other sources are quoted accordingly; otherwise the information presented in this report is solely work of the author."

Signature	: ______________________
Author	: SUZALEY BIN SULAIMAN
Date	: 17TH NOVEMBER 2008

DEDICATION

Specially dedicate to

My beloved family and those people who have guided and inspired me

through out my journey of education.

ACKNOWLEDGEMENT

In the name of Allah S.W.T, the most Gracious, the ever Merciful, Praise is to Allah, Lord of the universe and Peace and Prayers be upon His final Prophet and Messenger Muhammad S.A.W.

First, I would like to express my acknowledgment and gratitude to my supervisor, Mrs. Nurul Haslina binti Noordin for the encouragement, advice, information, motivation, guidance and co-operation that been given throughout the progress and to complete this project.

My sincere appreciation also extends to all my colleagues and others who have provided assistance at various occasions. Their views and tips are useful indeed. Unfortunately, it is not possible to list all of them in this limited space.

Finally, special thanks extended to my beloved family who had given me moral support and prayed for my success.

Thank you,

Suzaley Bin Sulaiman

ABSTRACT

The remote control of a moving vehicle is a project that uses a radio frequency to control the maneuver and movement of the model car. The Peripheral Integration Controller (PIC) 16F877A is used in this project. The LCD is placed at the remote control to help user to determine the maneuver angle of the model car. The potentiometer is used as a joystick for the transmitter. There are 2 potentiometer used in this project. The first potentiometer is used to control the maneuver angle and the other one is used to control the movement of the car. The stepper motor is used instead of DC motor to maneuver the car. This is because the stepper motor is more precise than DC motor. DC motor is used to for the movement of the car. The stepper motor and DC motor are interface with PIC on receiver board. The frequency of RF module that used is 343MHz. The car will maneuver to the left or right with the exact angle and the angle is already program in PIC. The angle that has been program in the PIC is 5°, 15° and 45°. To ensure the maneuver more precise, the movement of the motor will stop whenever the car is maneuvering to the left or right. The head light will automatically ON when the car detect the surrounding a little bit darker than usual. The bottom light of the car will automatically ON whenever the surrounding is completely dark. There are also 2 other head light that can be ON using the remote control.

ABSTRAK

Projek ini mengguna gelombang radio frekuensi untuk mengawal pergerakan kereta. PIC 16F877A digunakan di dalam projek ini. LCD dipasang pada bahagian alat kawalan jauh bagi memudahkan pengguna menentukan sudut belokan yang dikehendaki. Perintang boleh laras digunakan sebagai alat kawalan. Terdapat 2 perintang boleh laras yg digunakan, satu perintang digunakan untuk mengawal sudut belokan kereta dan satu lagi digunakan untuk mengawal pergerakan kereta. Kereta ini menggunakan "stepper motor" untuk mengawal sudut belokan. "Stepper motor" adalah lebih cekap berbanding DC motor. DC motor digunakan untuk mengawal pergerakan ke depan atau ke belakang kereta. DC motor dan "stepper motor" disambung ke PIC. Frekuensi gelombang radio yang digunakan adalah 343MHz. Kereta akan bergerak ke kiri atau ke kanan berdasarkan sudut belokan yang telah diprogramkan di dalam PIC. Sudut yang telah diprogramkan di dalam PIC adalah 5°, 15° dan 45°. Untuk memastikan sudut belokan adalah tepat, kereta akan berhenti ketika kereta membelok ke kanan atau ke kiri. Lampu hadapan akan menyala secara automatik apabila keadaan sekeliling berubah menjadi lebih gelap daripada biasa. Lampu bawah akan menyala apabila keadaan berubah menjadi terlalu gelap. Terdapat 2 lagi lampu hadapan yang boleh dikawal dengan menggunakan alat kawalan jauh.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Component	**The Description**
PIC	Programmable Interface Controller
LCD	Liquid Crystal Display
DC	Direct Current
RC	Radio-Controlled
vs.	Versus

LIST OF APPENDIXES

BORANG PENGESAHAN STATUS TESIS♦

JUDUL: **REMOTE CONTROL OF A MOVING VEHICLE**

SESI PENGAJIAN: **2008/2009**

Saya **SUZALEY BIN SULAIMAN (860717-33-5781)**

(HURUF BESAR)

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

1. Tesis adalah hakmilik Universiti Malaysia Pahang (UMP).
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan tesis ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. **Sila tandakan (√)

☐	**SULIT**	(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)
☐	**TERHAD**	(Mengandungi maklumat TERHAD yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan)
√	**TIDAK TERHAD**	

Disahkan oleh:

(TANDATANGAN PENULIS)

(TANDATANGAN PENYELIA)

Alamat Tetap:

PT 917 TAMAN MURNI, BERIS LANJUT,
16100 PENGKALAN CHEPA,
KELANTAN

NURUL HAZLINA BT NOORDIN
(Nama Penyelia)

Tarikh: **17 NOVEMBER 2008**

Tarikh: : **17 NOVEMBER 2008**

CATATAN:

* Potong yang tidak berkenaan.

** Jika tesis ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali tempoh tesis ini perlu dikelaskan sebagai atau TERHAD.

♦ Tesis dimaksudkan sebagai tesis bagi Ijazah doktor Falsafah dan Sarjana secara Penyelidikan, atau disertasi bagi pengajian secara kerja kursus dan penyelidikan, atau Laporan Projek Sarjana Muda (PSM).

"I hereby acknowledge that the scope and quality of this thesis is qualified for the award of the Bachelor Degree of Electrical Engineering (Electronics)"

Signature	: ______________________________
Name	: NURUL HAZLINA BT NOORDIN
Date	: 17 NOVEMBER 2008

CHAPTER 1

INTRODUCTION

1.1 Overview

Nowadays, we often heard about nature disaster and how it can kill thousands of people. For an example, sometimes when a building collapsed, there are still survivors under the ruin but we can't enter and search for the survivors under the ruin because it is too dangerous and sometimes there is just a small opening. Besides that, sometimes it is dangerous to enter dark area bluntly. We can get hurt or worst we can die because we enter without knowing what is beyond it.

Realizing this problem, this project is developed to help us scout the surrounding inside a dangerous area and unreachable to human. We can control the movement of the vehicle using a joystick. The vehicle can be control within a range of 7 meter. When entering a dark place, the vehicle will automatically switch on the vehicle light to make the vision clearer. A camera is attached to the vehicle to help us see the visual inside the area.

In the first phase, this project will only focus on making the vehicle movement according to the joystick and move within the range of 7 meter. After the first phase is successfully done, the vehicle will be added with a vehicle light that will automatically on when entering a dark area. After finish adding the vehicle light, a camera will be attached to the vehicle and connected to a laptop or a computer.

1.2 Problem Statement

There is some areas that can not be entered by human being such as a small hole and a place that dangerous to human. This project is developed to help human scout the surrounding inside a dangerous area and unreachable to human.

1.3 Objective

The main objective of this project is to build a device that can control a vehicle maneuver and movement within a range of 7 meter.

1.4 Scope

This project is focused to design and build the model of a remote control and a model car that would be used to scout the dangerous and unreachable areas to human. Therefore, this model will cover the scope as followed:

(i) The vehicle can be control in a range of 7 meter.
(ii) The vehicle can move forward, reverse, left and right.
(iii) The degree movement of the vehicle can be control precisely.

1.5 Methodology

The purpose of this project is to control the movement of a vehicle using the input from the joystick. The vehicle can move forward, reverse, and turn left and right. The joystick is controlling the vehicle using radio frequency. The potentiometer is used as the joystick.

The hardware contains 2 parts that is the transmitter (Figure 1.1) and the receiver (Figure 1.2). In the transmitter part, the joystick will be connected to the PIC 16F877A and the PIC 16F877A will send the signal to the encoder (TWS-434).

The receiver will receive the signal from the transmitter using the decoder (RWS-434) and sent the signal to PIC 16F877A. The PIC 16F877A will send a signal to the motor driver and the motor will run according to the joystick input.

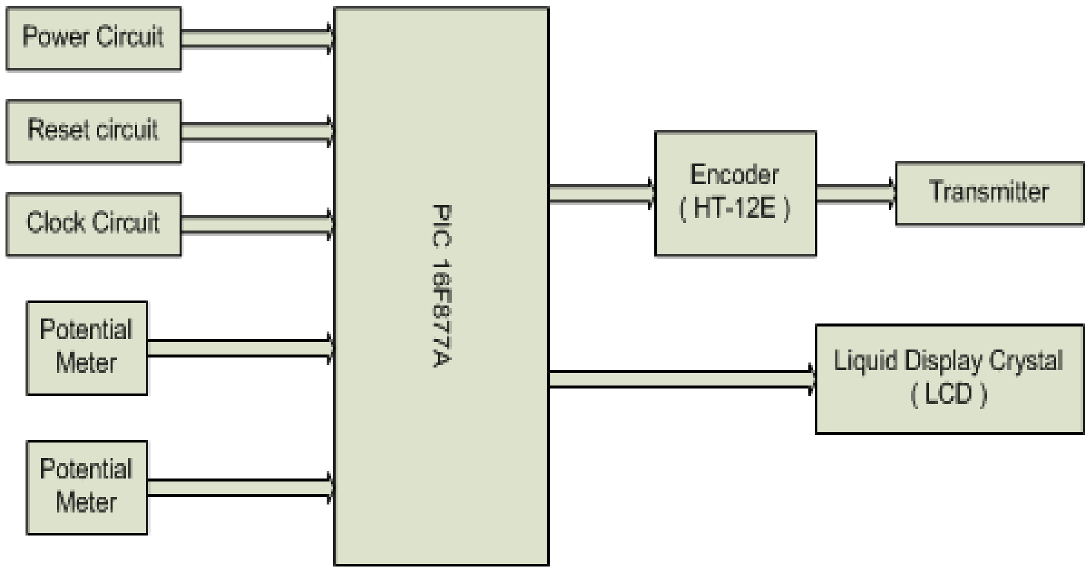


Figure 1.1: Transmitter Block Diagram

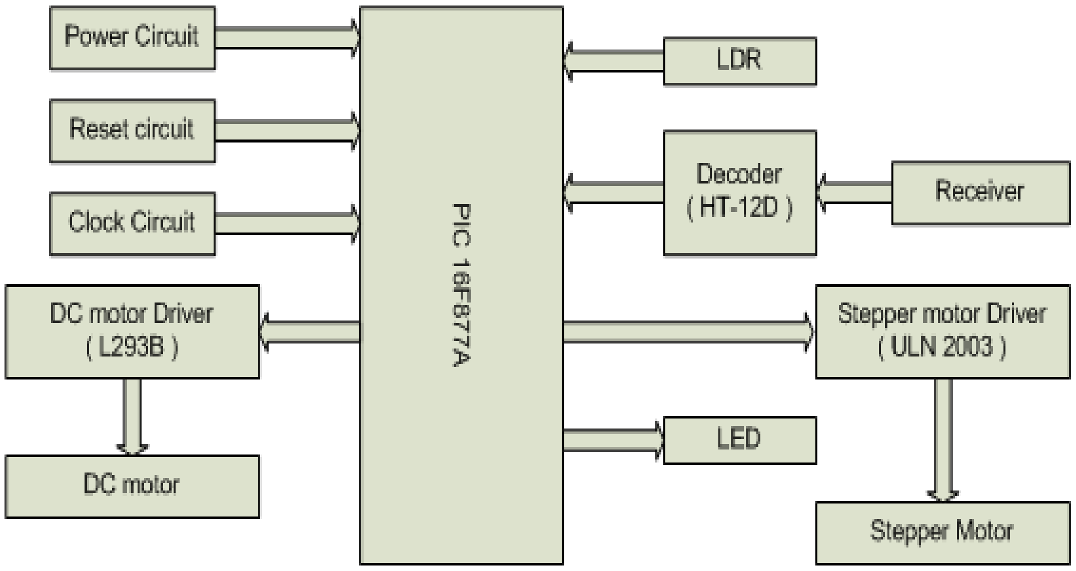


Figure 1.2: Receiver Block Diagram

1.6 Review of the Thesis Content

This thesis consists of five chapters. This chapter discuss about overview of project, problem statement, objective, project scope, methodology and thesis organization.

Chapter 2 will describe about the input, controller, the output of the system and the previous similar project. It will explain about the concept of the components that are used in the project.

Chapter 3 includes the project methodology. It will explain how the project is organized and the flow of process in completing this project. Also in this topic discusses the methodology of the system, circuit design, software design and the mechanical design.

Chapter 4 will be discussing about the result obtained in this project and a discussion about the result. This chapter also discuss about the experimental result, expected performance and performance limit that can be archive.

Finally, the conclusions for this project are presented in chapter 5. This chapter also discusses about the recommendation or future development of the project and cost that involved in the project

CHAPTER 2

THEORY AND LITERATURE REVIEW

2.1 Overview

This chapter reviews about previous system that has been developed and has similarities with the remote control of a moving vehicle. This topic will also discuss about the component that will be used in developing this systems.

2.2 Available Remote Control Car

2.2.1 Ruf Bot 1.1

The article is about constructing a car that can be controlled using joystick. The project is called the Ruf Bot. It is made from a 4 x 8 piece of plastic sheet and has modified servos for motors. The servos are modified in a way that makes them DC gear head motors. The servos internal electronics have been completely removed and only the motor and gears remain (hence the need for an H-Bridge).

The project uses TX/RX pair and the serial communication built into the PICBasic programming language for the PIC's. The actual programming couldn't be easier since it is written in Basic and uses premade serial communication routines.

The position of the Potentiometer in the joystick can be determine using the PICBasic 'POT' command and the result is store in memory at location 'B0'. From there, the contents of 'B0' are sent using the 'SEROUT' command to pin 6 of the TWS 434 transmitter **[1].** On the receiver end, the 'SERIN' command is use to read the incoming data from pin 3 on the RWS 434 and the result is store in 'B0'. The value in 'B0' directly correlates to joystick position, above 150 is right, below 106 is left, and in between is center. By using these numbers a dead zone can be define.

The implication is easier at this point. From the number that is transmitted we can determine the movement of the vehicle. By using the antenna that is made for 900 MHz cordless telephones, the vehicle can be control within the range of 350 feet. The circuit and the model of the project are shown in Figure 2.1, 2.2 and 2.3 below.

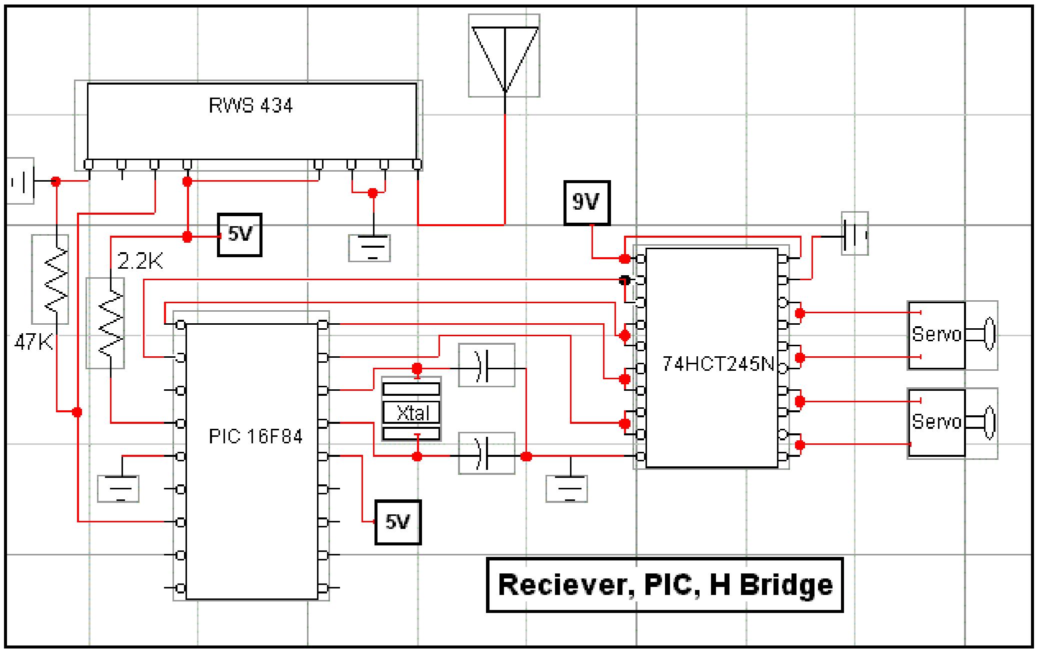


Figure 2.1: Transmitter Circuit

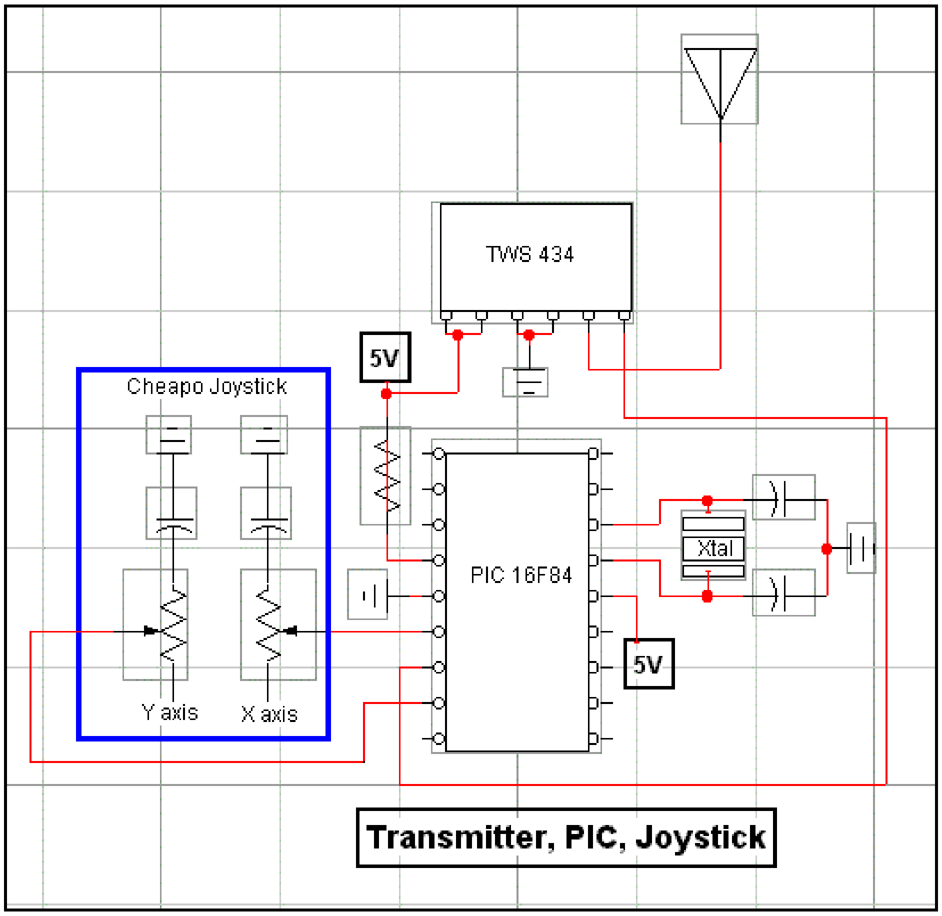


Figure 2.2: Receiver Circuit

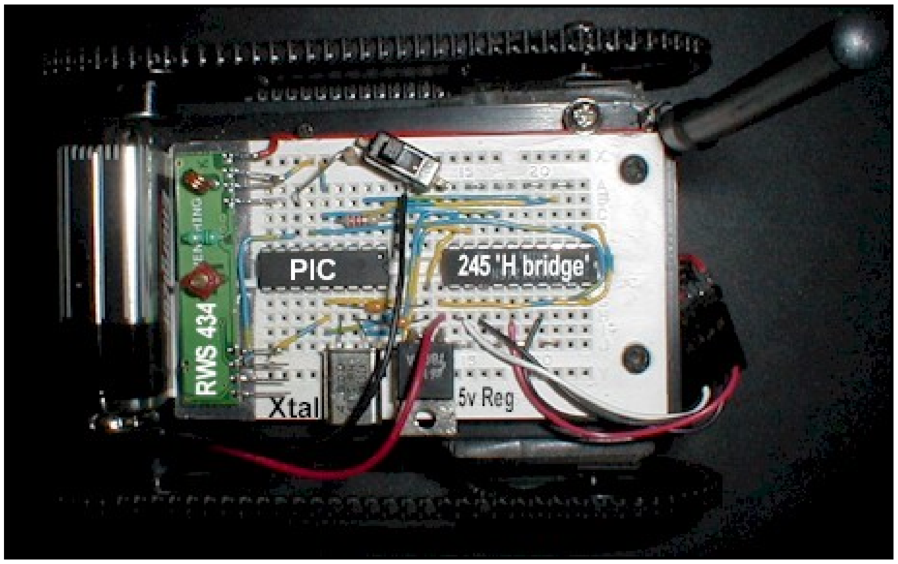


Figure 1.3: Car model

2.2.2 Toy Car Hack - "Synthetic Rodent Development"

This article is about making a light follower toy car. The toy car will follow the light that is beam to the car. It will determine whether the light came from the left, right or center. The car will stop when an object has been hit. The car is built in 5 1/2 inches in length. It is powered by 4 AAA batteries and it has a small dc motor. After gutting the original electronics, a piece of circuit board was cut to size and mounted with one screw. The circuit board is pad-per-hole type and the wiring on the bottom is done with tiny pieces of 30 gauge (wire wrap) wire soldered between points [2].

Visual detection is done using a pair of (matched) photocells. This type of cell works at even very low light levels. The light sources can be determined to be from left, right, or center using a pair in series feeding a 3 level comparator circuit. The photo cells are physically located to 'look' through a hole drilled in the black plastic windshield.

The motor drive circuit has an adjustable current level detector. This detects motor 'stalls' and is used to determine when an object has been hit. To save power in 'sleep' mode, the PIC powers down the entire external circuit (the op-amp and associated resistor networks) when not needed. A permanent magnet 'floats' on a pair of pivots and is held centered by being attracted to a fixed set of metal pole pieces. When power is applied to the solenoid, the pole pieces move the magnet to the left or right [3].

The toy car hack model is shown in Figure 2.4 below.

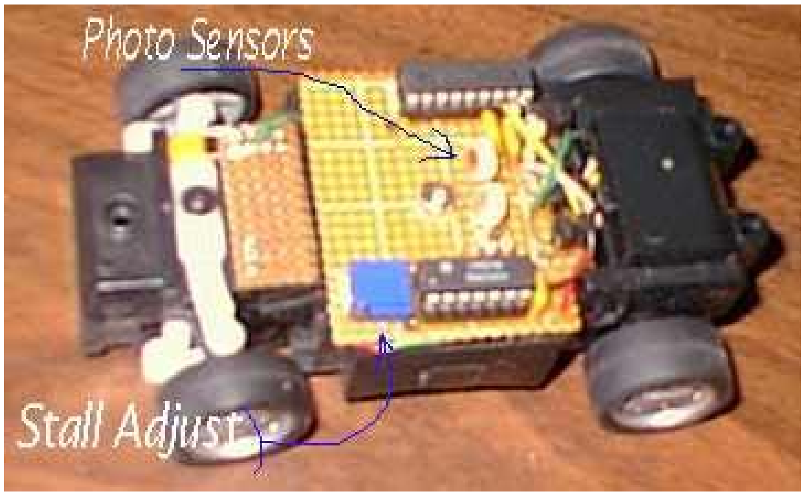


Figure 2.4: Toy Car Hack model

2.2.3 Final Project - RC Car Controller

The article is about building a transmitter and receiver modules for a radio-controlled (RC) car, as well as implements variable-speed motor control and a continuous steering function. The block diagram of the project is shown in Figure 2.6. The original implementation of speed control in the car consists of a servo which mechanically moves the arm of a simple high-power potentiometer. While the motion of the servo is continuous, the circuit only produces six discrete levels: three forward, two reverse, and neutral.

A simple communication protocol was established to send messages from the controller to the car. Because the connection is serial, we encode each command into a byte-long packet. The top nibble denotes the command, and the lower nibble represents the level at which the command is to be executed. For discrete operations (like headlights), the second nibble determines which functions are to be toggled. Communication travels one way from the controller to the car, which cuts down on the hardware required for either unit.

The original controller used spring centered potentiometers to produce analog signals which controlled the speed and direction of the car. The analog signals were transmitted to the car via a 75 MHz AM radio link.

The steering of the car is controlled by a servo. Rather than supplying a simple voltage, servos operate on a fixed voltage source and a control line that is pulsed to dictate the turn angle. The servo is connected to the MCU. A Timer interrupt subroutine is used in conjunction with user input from the transmitter to determine the pulse width to be applied and sent through the MCU's port pins. Figure 2.5 show the steering control circuit.

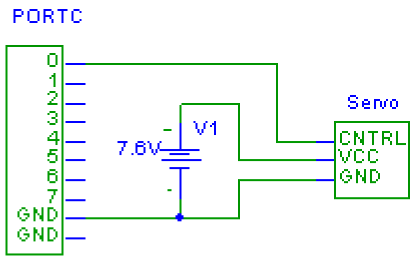


Figure 2.5: Steering Control Circuit

The received four-bit digital signal denotes the desired speed of the motor. The motor is drive at full voltage and pulse width modulation is used to control speed. Several products exist which accomplish this task; however, it is a simple matter to implement PWM on the microcontroller. A timer interrupt is used to count 16 “ticks”, and the given magnitude determined how many of those ticks the motor will be driven.

The joystick is able to move the car forwards and back, and turn left and right. The pulse width modulation, as it was originally conceived, was far too fast for the motor to turn at all. Slowing down the period to approximately one second, up from 1 millisecond, solved the problem. There was some initial irregularity in the pulse itself; instead of regular intervals, it seemed as though the pulses were being interrupted by some external stimulus. Careful programming to avoid register clobbering solved much of the problem, but irregularities occasionally appear at unpredictable times [4].

Turning the car is nicely variable but not particularly smooth. The same irregularities found in the speed control manifest themselves to a greater extent in the servo, causing the wheels to jerk slightly left and right of the desired turn angle. The problem to be extremely difficult to solve with the Atmel MCU, as pulse widths for the servo vary from 1 to 2 milliseconds; at those small periods, it is difficult to get very accurate timing with the Timer0 interrupt [5]. Perhaps a 555 circuit would have solved this problem.

The software portion of the project was easy to design, and the implementation is quite simple. By avoiding complicated code, we are able to concentrate on hardware issues. Because most of this project relies on carefully designed circuitry, the reliability of the program greatly simplifies the debugging process.

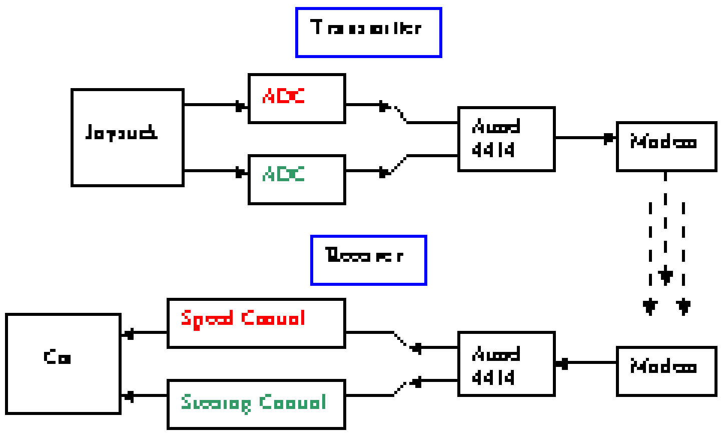


Figure 2.6: RC-Car Block Diagram

2.3 Components

A system is constructed with a certain components which has it own function integrated to each others for completing the whole system. The system should have an input, a controller and the output. In the transmitter board, the input is potential meter, the controller is PIC 16F877A and the output is the transmitter and LCD. For the receiver board, the input is receiver, the controller is PIC 16F877A and the output is stepper motor and dc motor.

2.3.1 Input - Potentiometer

A potentiometer is a three-terminal resistor with a sliding contact that forms an adjustable voltage divider. If only two terminals are used (one side and the wiper), it acts as a variable resistor or Rheostat. Potentiometers are commonly used to control electrical devices such as a volume control of a radio. Potentiometers operated by a mechanism can be used as position transducers, for example, in a joystick. Figure 2.7 shows the construction of a wire-wound circular potentiometer.

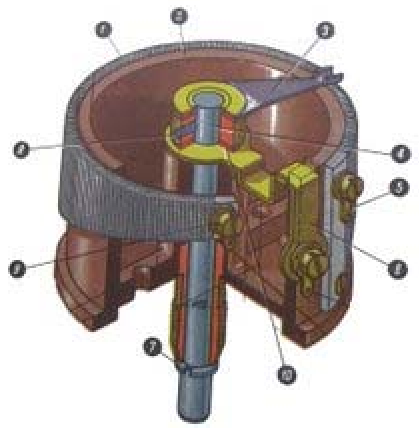


Figure 2.7: Construction of a wire-wound circular potentiometer

Key: The resistive element (1) of the shown device is trapezoidal, giving a non-linear relationship between resistance and turn angle. The wiper (3) rotates with the axis (4), providing the changeable resistance between the wiper contact (6) and the fixed contacts (5) and (9). The vertical position of the axis is fixed in the body (2) with the ring (7) (below) and the bolt (8) (above).

A potentiometer (colloquially called a "pot") is constructed using a semi-circular resistive element with a sliding contact (wiper). The resistive element, with a terminal at one or both ends, is flat or angled, and is commonly made of graphite, although other materials maybe used instead. The wiper is connected through another sliding contact to another terminal. On panel pots, the wiper is usually the center terminal of three. For single-turn pots, this wiper typically travels just under one revolution around the contact. "Multiturn" potentiometers also exist, where the resistor element may be helical and the wiper may move 10, 20, or more complete revolutions, though multiturn pots are usually constructed of a conventional resistive element wiped via a worm gear. Besides graphite, materials used to make the resistive element include resistance wire, carbon particles in plastic, and a ceramic/metal mixture called cermet. Figure 2.8 shows an example of typical single turn potentiometer.

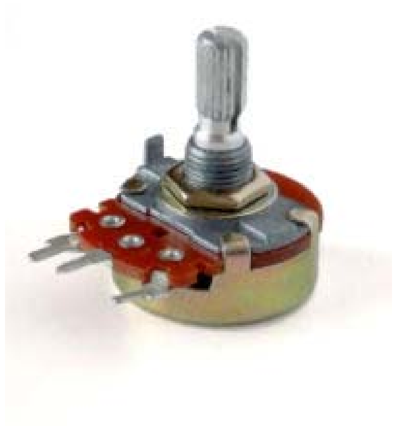


Figure 2.8: A typical single turn potentiometer

One form of rotary potentiometer is called a String potentiometer. It is a multi-turn potentiometer operated by an attached reel of wire turning against a spring. It is used as a position transducer. In a linear slider pot, a sliding control is provided instead of a dial control. The resistive element is a rectangular strip, not semi-circular as in a rotary potentiometer. Because of the large opening for the wiper and knob, this type of pot has a greater potential for getting contaminated. Potentiometers can be obtained with either linear or logarithmic relations between the slider position and the resistance (potentiometer laws or "tapers").

A linear taper potentiometer has a resistive element of constant cross-section, resulting in a device where the resistance between the contact (wiper) and one end terminal is proportional to the distance between them. Linear taper describes the electrical characteristic of the device, not the geometry of the resistive element. Linear taper potentiometers are used when an approximately proportional relation is desired between shaft rotation and the division ratio of the potentiometer; for example, controls used for adjusting the centering of (an analog) cathode-ray oscilloscope.

A logarithmic taper potentiometer has a resistive element that either 'tapers' in from one end to the other, or is made from a material whose resistivity varies from one end to the other. This results in a device where output voltage is a logarithmic (or inverse logarithmic depending on type) function of the mechanical angle of the pot.

Most (cheaper) "log" pots are actually not logarithmic, but use two regions of different, but constant, resistivity to approximate a logarithmic law. A log pot can also be simulated with a linear pot and an external resistor. True log pots are significantly more expensive. Logarithmic taper potentiometers are often used in connection with audio amplifiers.

2.3.2 Controller - PIC

Controller is the main part of the system where all the process flow will be controlled by this hardware accordingly to the embedded programming in it. PIC Microcontroller is chosen for the system as the controller. The functions of the PIC are limited by the manufacturer or the types of certain model. PIC is a family of Harvard architecture microcontrollers made by Microchip Technology, derived from the PIC1640 originally developed by General Instrument's Microelectronics Division. The name PIC initially referred to "Programmable Interface Controller", but shortly thereafter was renamed "Programmable Intelligent Computer".

The PIC architecture is distinctively minimalist. It is characterized by the following features:

I. separate code and data spaces (Harvard architecture)
II. a small number of fixed length instructions
III. most instructions are single cycle execution (4 clock cycles), with single delay cycles upon branches and skips
IV. a single accumulator (W), the use of which (as source operand) is implied (i.e. is not encoded in the opcode)
V. All RAM locations function as registers as both source and/or destination of math and other functions.[1]
VI. a hardware stack for storing return addresses
VII. a fairly small amount of addressable data space (typically 256 bytes), extended through banking
VIII. data space mapped CPU, port, and peripheral registers
IX. the program counter is also mapped into the data space and writable (this is used to implement indirect jumps)

Unlike most other CPUs, there is no distinction between "memory" and "register" space because the RAM serves the job of both memory and registers, and the RAM is usually just referred to as the register file or simply as the registers.

Baseline Core Devices

These devices feature a 12-bit wide code memory, a 32-byte register file, and a tiny two level deep call stack. They are represented by PIC10 series, as well as some PIC12 and PIC16 devices. Baseline devices are available in 6-pin to 40-pin packages.

Generally the first 7 to 9 bytes of the register file are special-purpose registers, and the remaining bytes are general purpose RAM. If banked RAM is implemented, the bank number is selected by the high 3 bits of the FSR. This affects register numbers 16–31; registers 0–15 are global and not affected by the bank select bits.

The ROM address space is 512 words (12 bits each), which may be extended to 2048 words by banking. CALL and GOTO instructions specify the low 9 bits of the new code location; additional high-order bits are taken from the status register. Note that a CALL instruction only includes 8 bits of address, and may only specify addresses in the first half of each 512-word page.

The instruction set is as follows. Register numbers are referred to as "f", while constants are referred to as "k". Bit numbers (0–7) are selected by "b". The "d" bit selects the destination: 0 indicates W, while 1 indicates the result is written back to source register f.

Mid-Range Core Devices

These devices feature a 14-bit wide code memory, and an improved 8 level deep call stack. The instruction set differs very little from the baseline devices, but the increased opcode width allows 128 registers and 2048 words of code to be directly addressed. The mid-range core is available in the majority of devices labeled PIC12 and PIC16.

The first 32 bytes of the register space are allocated to special-purpose registers; the remaining 96 bytes are used for general-purpose RAM. If banked RAM is used, the high 16 registers (0x70–0x7F) are global, as are a few of the most important special-purpose registers, including the STATUS register which holds the RAM bank select bits. (The other global registers are FSR and INDF, the low 8 bits of the program counter PCL, the PC high preload register PCLATH, and the master interrupt control register INTCON.)

The PCLATH register supplies high-order instruction address bits when the 8 bits supplied by a write to the PCL register, or the 11 bits supplied by a GOTO or CALL instruction, is not sufficient to address the available ROM space.

PIC17 High End Core Devices

The 17 series never became popular and has been superseded by the PIC18 architecture. It is not recommended for new designs, and may be in limited availability. Improvements over earlier cores are 16-bit wide opcodes (allowing many new instructions), and a 16 level deep call stack. PIC17 devices were produced in packages from 40 to 68 pins. The 17 series introduced a number of important new features:

- a memory mapped accumulator
- read access to code memory (table reads)
- direct register to register moves (prior cores needed to move registers through the accumulator)
- an external program memory interface to expand the code space
- an 8bit x 8bit hardware multiplier
- a second indirect register pair

- auto-increment/decrement addressing controlled by control bits in a status register (ALUSTA)

PIC18 High End Core Devices

Microchip introduced the PIC18 architecture in 2002 [2], and unlike the 17 series, it has proven to be very popular, with a large number of device variants presently in manufacture. In contrast to earlier devices, which were more often than not programmed in assembly, C has become the predominant development language [3]. The 18 series inherits most of the features and instructions of the 17 series, while adding a number of important new features:

- much deeper call stack (31 levels deep)
- the call stack may be read and written
- conditional branch instructions
- indexed addressing mode (PLUSW)
- extending the FSR registers to 12 bits, allowing them to linearly address the entire data address space
- the addition of another FSR register (bringing the number up to 3)

The auto increment/decrement feature was improved by removing the control bits and adding four new indirect registers per FSR. Depending on which indirect file register is being accessed it is possible to postdecrement, postincrement, or preincrement FSR; or form the effective address by adding W to FSR. In more advanced PIC18 devices, an "extended mode" is available which makes the addressing even more favourable to compiled code:

- a new offset addressing mode; some addresses which were relative to the access bank are now interpreted relative to the FSR2 register
- the addition of several new instructions, notable for manipulating the FSR registers.

These changes were primarily aimed at improving the efficiency of a data stack implementation. If FSR2 is used either as the stack pointer or frame pointer, stack items may be easily indexed -- allowing more efficient re-entrant code. Microchip C18 chooses to use FSR2 as a frame pointer.

PIC24 and PIC 16-bit Microcontrollers

Microchip introduced the dsPIC series of chips in 2001, and they entered mass production in late 2004. They are Microchip's first inherently 16-bit microcontrollers. PIC24 devices are designed as general purpose microcontrollers. PIC devices include digital signal processing capabilities in addition.

Architecturally, although they share the PIC moniker, they are very different from the 8-bit PICs. The most notable differences are:

- they feature a set of 16 working registers
- they fully support a stack in RAM, and do not have a hardware stack
- bank switching is not required to access RAM or special function registers
- data stored in program memory can be accessed directly using a feature called Program Space Visibility
- interrupt sources may be assigned to distinct handlers using an interrupt vector table

Some features are:

- hardware MAC (multiply-accumulate)
- barrel shifting
- bit reversal
- (16x16)-bit multiplication and other DSP operations.
- hardware support for loop indexing
- Direct Memory Access

To summarize, a microcontroller contains (in one chip) two or more of the following elements in order of importance:

i. Includes Powerful Microchip PIC16F877 Microcontroller with 8kb Internal Flash program memory (Figure 2.9)
ii. Operating Speed at 10MHz
iii. Direct In-Circuit Programming for Easy Program Updates
iv. Up to 28 I/O points with easy to connect standard headers
v. Internal EEPROM
vi. 8 Channel 10-bit A/D Converter
vii. One 16-bit Timer with Two 8-bit Timers
viii. Serial port
ix. Reset Button

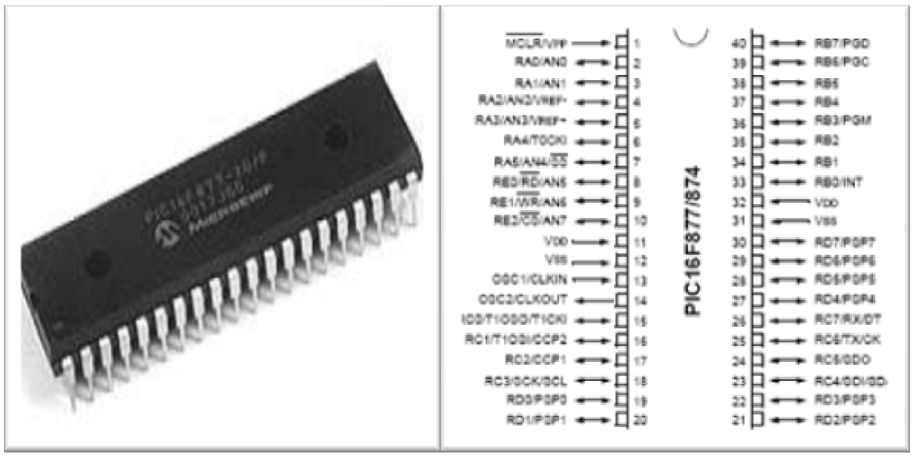


Figure 2.9: PIC 16F877A and it’s Schematic

2.3 Output

2.3.1 LCD

A liquid crystal display (LCD) is an electro-optical amplitude modulator realized as a thin, flat display device made up of any number of color or monochrome pixels arrayed in front of a light source or reflector. It is often utilized in battery-powered electronic devices because it uses very small amounts of electric power. Figure 2.10 shows the example of LCD. Important factors to consider when evaluating an LCD monitor:

- Resolution: The horizontal and vertical size expressed in pixels (e.g., 1024x768). Unlike monochrome CRT monitors, LCD monitors have a native-supported resolution for best display effect.
- Dot pitch: The distance between the centers of two adjacent pixels. The smaller the dots pitch size, the fewer granularities are present, resulting in a sharper image. Dot pitch may be the same both vertically and horizontally, or different (less common).
- Viewable size: The size of an LCD panel measured on the diagonal (more specifically known as active display area).
- Response time: The minimum time necessary to change a pixel's color or brightness. Response time is also divided into rise and fall time. For LCD Monitors, this is measured in btb (black to black) or gtg (gray to gray). These different types of measurements make comparison difficult.
- Refresh rate: The number of times per second in which the monitor draws the data it is being given. A refresh rate that is too low can cause flickering and will be more noticeable on larger monitors. Many high-end LCD televisions now have a 120 Hz refresh rate (current and former NTSC countries only) [citation needed]. This allows for less distortion

when movies filmed at 24 frames per second (fps) are viewed due to the elimination of telecine (3:2 pulldown). The rate of 120 was chosen as the least common multiple of 24 fps (cinema) and 30 fps (TV).

- Matrix type: Active TFT or Passive.
- Viewing angle: (coll., more specifically known as viewing direction).
- Color support: How many types of colors are supported (coll., more specifically known as color gamut).
- Brightness: The amount of light emitted from the display (coll., more specifically known as luminance).
- Contrast ratio: The ratio of the intensity of the brightest bright to the darkest dark.
- Aspect ratio: The ratio of the width to the height (for example, 4:3, 16:9 or 16:10).
- Input ports (e.g., DVI, VGA, LVDS, Display Port, or even S-Video and HDMI).

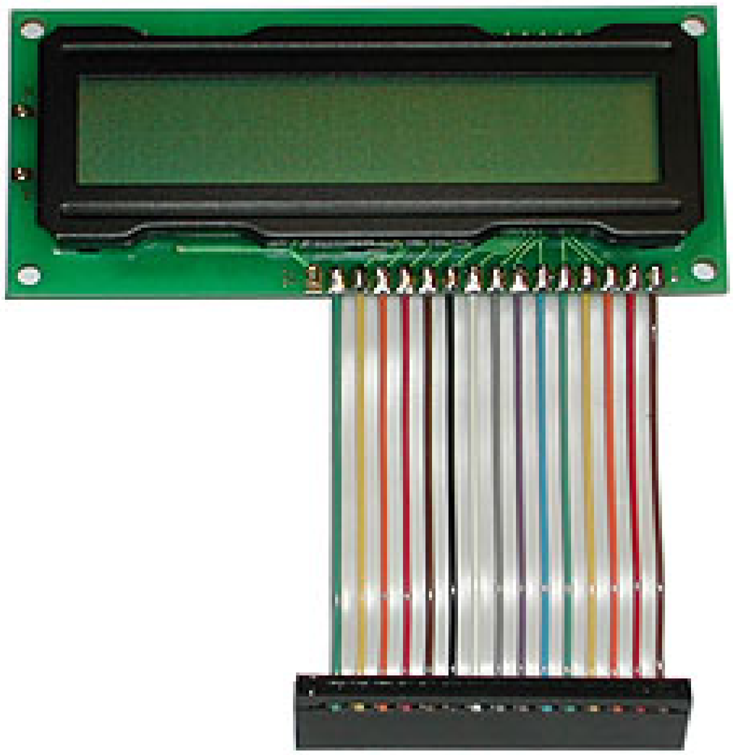


Figure 2.10: Example of LCD

2.3.2 Stepper Motor

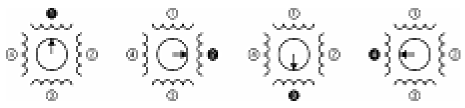
A stepper motor (or step motor) is a brushless, synchronous electric motor that can divide a full rotation into a large number of steps. The motor's position can be controlled precisely, without any feedback mechanism. Stepper motors are similar to switched reluctance motors, which are very large stepping motors with a reduced pole count, and generally are closed-loop commutated.

Stepper motors operate differently from normal DC motors, which rotate when voltage is applied to their terminals. Stepper motors, on the other hand, effectively have multiple "toothed" electromagnets arranged around a central gear-shaped piece of iron. The electromagnets are energized by an external control circuit, such as a microcontroller. To make the motor shaft turn, first one electromagnet is given power, which makes the gear's teeth magnetically attracted to the electromagnet's teeth. When the gear's teeth are thus aligned to the first electromagnet, they are slightly offset from the next electromagnet. So, when the next electromagnet is turned on and the first is turned off, the gear rotates slightly to align with the next one, and from there the process is repeated. Each of those slight rotations is called a "step." In that way, the motor can be turned to a precise angle.

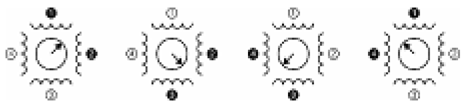
Stepper motors are constant-power devices (power = angular velocity x torque). As motor speed increases, torque decreases. The torque curve may be extended by using current limiting drivers and increasing the driving voltage.

Steppers exhibit more vibration than other motor types, as the discrete step tends to snap the rotor from one position to another. This vibration can become very bad at some speeds and can cause the motor to lose torque. The effect can be mitigated by accelerating quickly through the problem speed range, physically damping the system, or using a micro-stepping driver. Motors with a greater number of phases also exhibit smoother operation than those with fewer phases. Figure 2.11 shows the examples of stepper motors.

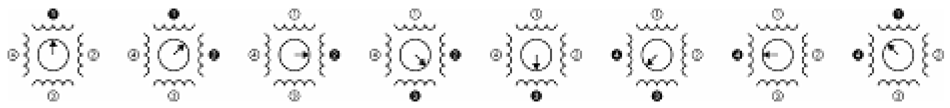
Full Step, Low Torque



Full Step, High Torque (standard application)



Half Step (best precision):



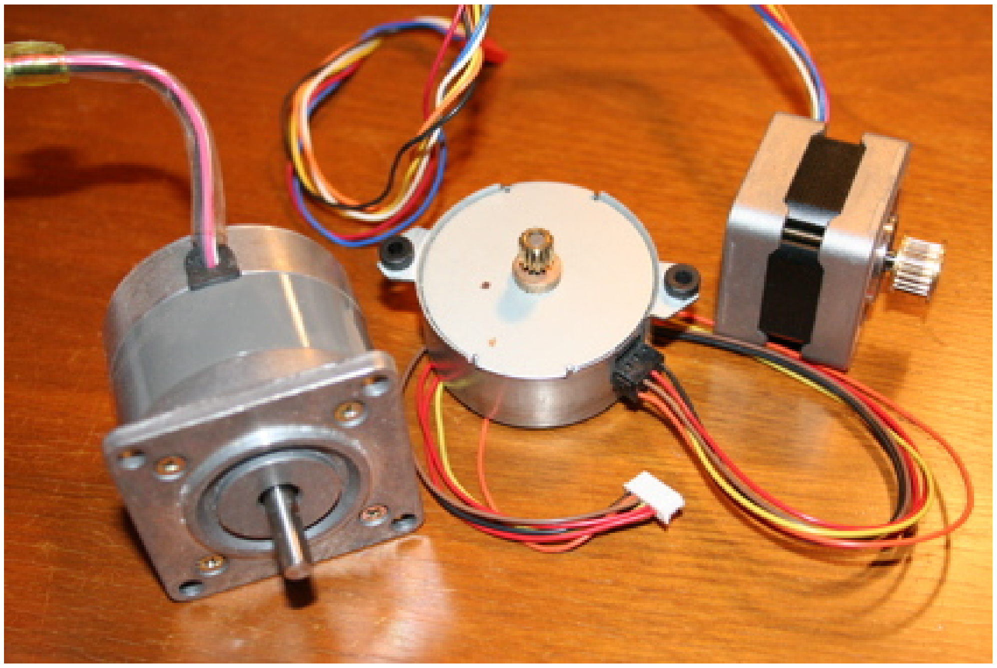


Figure 2.11: Examples of stepper motor

2.3.3 DC motor

Electric motors are everywhere. In a house, almost every mechanical movement that you see around you is caused by a DC (direct current) electric motor. An electric motor is a device that transforms electrical energy into mechanical energy by using the motor effect. Figure 2.13 shows the example of DC motor.

Every DC motor has six basic parts -- axle, rotor (a.k.a., armature), stator, commutator, field magnet(s), and brushes. In most common DC motors, the external magnetic field is produced by high-strength permanent magnets. The stator is the stationary part of the motor -- this includes the motor casing, as well as two or more permanent magnet pole pieces. The rotor rotates with respect to the stator. The rotor consists of windings (generally on a core), the windings being electrically connected to the commutator. Figure 2.12 shows an example of part of an Electric Motor. Industrial applications use dc motors because the speed-torque relationship can be varied to almost any useful form -- for both dc motor and regeneration applications in either direction of rotation. Continuous operation of dc motors is commonly available over a speed range of 8:1. Infinite range (smooth control down to zero speed) for short durations or reduced load is also common.

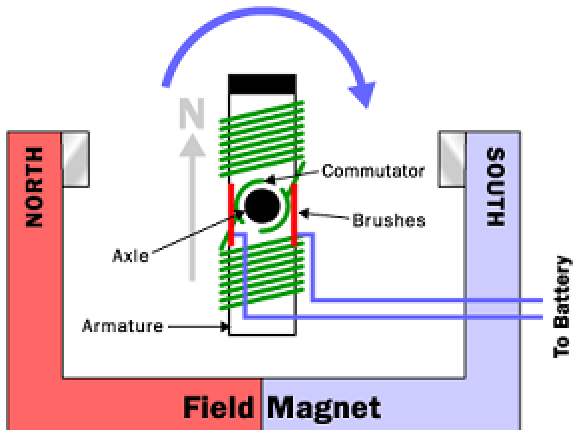


Figure 2.12: Part of an Electric Motor

Dc motors are often applied where they momentarily deliver three or more times their rated torque. In emergency situations, dc motors can supply over five times rated torque without stalling (power supply permitting). Dc motors feature a speed, which can be controlled smoothly down to zero, immediately followed by acceleration in the opposite direction -- without power circuit switching. And dc motors respond quickly to changes in control signals due to the dc motor's high ratio of torque to inertia.

Dc motors feature a speed, which can be controlled smoothly down to zero, immediately followed by acceleration in the opposite direction -- without power circuit switching. And dc motors respond quickly to changes in control signals due to the dc motor's high ratio of torque to inertia. The greatest advantage of DC motors may be speed control. Since speed is directly proportional to armature voltage and inversely proportional to the magnetic flux produced by the poles, adjusting the armature voltage and/or the field current will change the rotor speed.

Today, adjustable frequency drives can provide precise speed control for AC motors, but they do so at the expense of power quality, as the solid-state switching devices in the drives produce a rich harmonic spectrum. The DC motor has no adverse effects on power quality. The drawbacks of Dc motors are:

- Power supply, initial cost, and maintenance requirements are the negatives associated with DC motors
- Rectification must be provided for any DC motors supplied from the grid. It can also cause power quality problems.
- The construction of a DC motor is considerably more complicated and expensive than that of an AC motor, primarily due to the commutator, brushes, and armature windings. An induction motor requires no commutator or brushes, and most use cast squirrel-cage rotor bars instead of true windings — two huge simplifications.

Figure 2.13: Example of DC motor

CHAPTER 3

METHODOLOGY AND DESIGN

3.1 Overview

This chapter explains about the systems design through construction of the hardware and development of the software. In addition, the chapter elaborates the hardware and software stage by stage. All the operations of the hardware and software are included in this chapter. Before looking into the details of designing this project, it is best to start with brief review of the system design. Figure 3.1 and 3.2 shows the complete system design of a transmitter and a receiver board. The project is divided into 4 parts that is hardware testing, hardware implementation, software implementation and application.

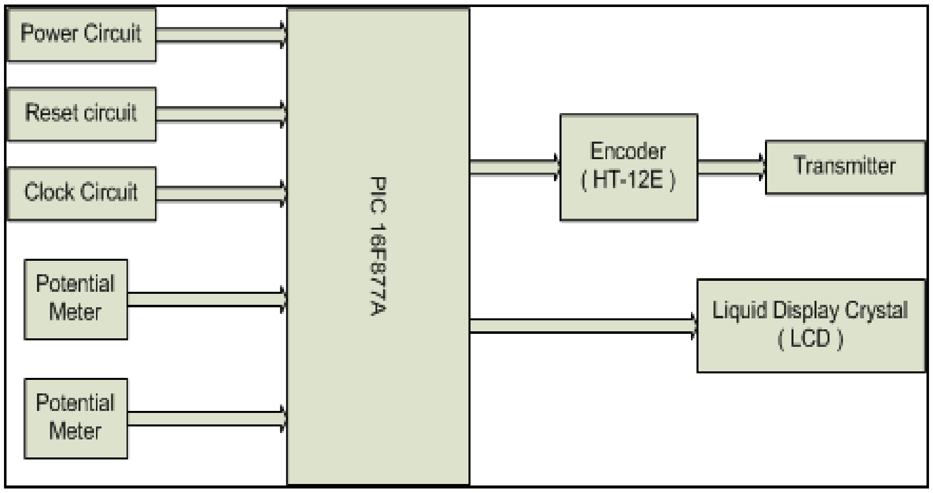


Figure 3.1: Transmitter Block Diagram

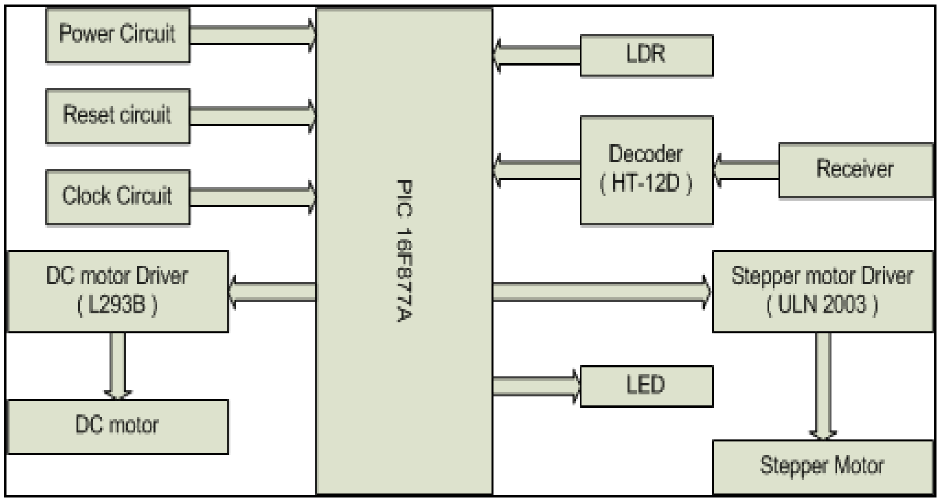


Figure 3.2: Receiver Block Diagram

3.2 Hardware Testing

The whole idea of this part is to test the functionality of the main component of the project. After this part has fully passed, the project will continue into hardware implementation part. In this part, it is divided into 5 stages that are PIC free running test, transmitter and receiver, LCD, stepper motor and DC motor.

3.2.1 PIC free running test

In the first phase, the circuit is connected as the diagram below (Figure 3.3). The purpose of this phase is to test the functionality of the PIC. There are 4 parts of connection in the first phase which is the power supply, clock circuit, reset circuit and also the free running test.

Before testing the circuit, the PIC is removed. Then, the power supply is set to 6v (without switching on the supply). Using the digital millimeters, only 5V can passes through the whole circuit and PIC. Now, the PIC can be placed back in the circuit making sure that the PIC was not plugged in backward. When the power supply is on, the clock is monitored by using the signal test probe which will show a high low signal alternatively. If the test probe shows the exact result, the LED connected to PORTB 7 will blink. This shows that the PIC is working promptly.

In addition to make sure that all the connections are correct, then RESET button is pushed. The circuit then will halts at its state before reset. When the RESET button is released, the system will reset itself. A reset circuit is essential is this design as it is useful when the system hangs. The function of the regulator in the circuit is to maintain the voltage output no matter how much the input voltage is. We are using two PIC and both PICs are tested for free running test. Figure 3.4 shows the free running test.

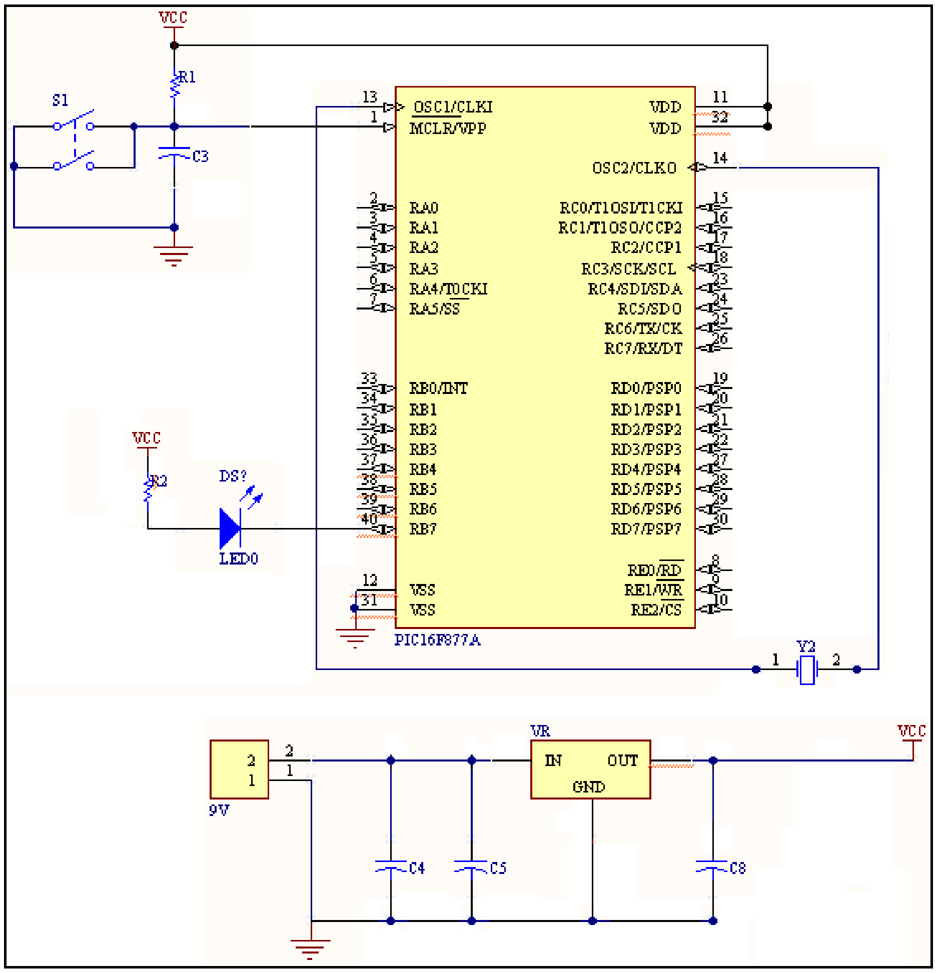


Figure 3.3: Free Running Circuit

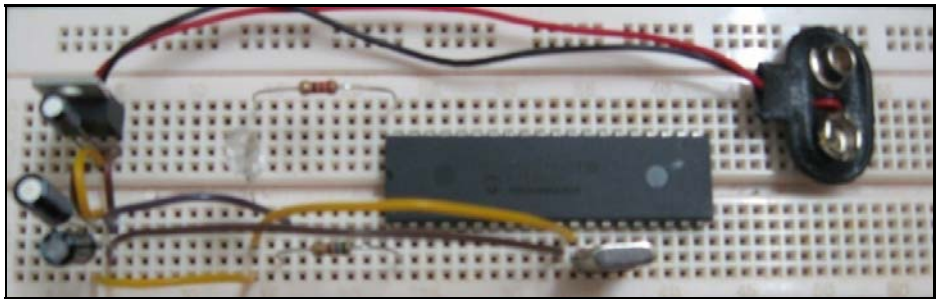


Figure 3.4: Free Running Test

3.2.2 Transmitter and receiver

In this part, we have to test the functionality of the transmitter and receiver. We can test the functionality using a function generator and an oscilloscope. The TX pin from the transmitter is connected to a function generator and the antenna is connected to the oscilloscope. The result is shown in the next chapter.

For the receiver part, the antenna pin is connected to the function generator and the RX pin is connected to the oscilloscope. The result is shown in the next chapter. To test the functionality of both transmitter and receiver when using together, the TX pin of the transmitter is connected to the function generator and the RX pin of the receiver is connected to the oscilloscope. We can determine the range of the transmitter and receiver from the oscilloscope. The result is shown in the next chapter.

The transmitter and receiver are connected using an IC to the PIC. For the transmitter, we use encoder to encode the data from the PIC to the transmitter. The data from the PIC is a digital data, and the encoder is function to send the digital data to the transmitter serially. For the receiver, we use decoder to decode the serial data that is received from the receiver and the data is sent to the PIC.

The encoder is connected to PORTB 0 – PORTB 3 in the transmitter board and the decoder is connected to PORTB 4 – PORTB 7 in the receiver board (Figure 3.5 and 3.6). The enable pin (TE) for the encoder is an active low, the pin is connected to a switch and from the switch to ground. The enable pin allowed the data from the PIC to be encoded and sent to the transmitter.

The encoder and decoder uses resistor to oscillate frequency according to the datasheet (Figure 3.7 and 3.8). The decoder frequency has to be 50 times the encoder frequency.

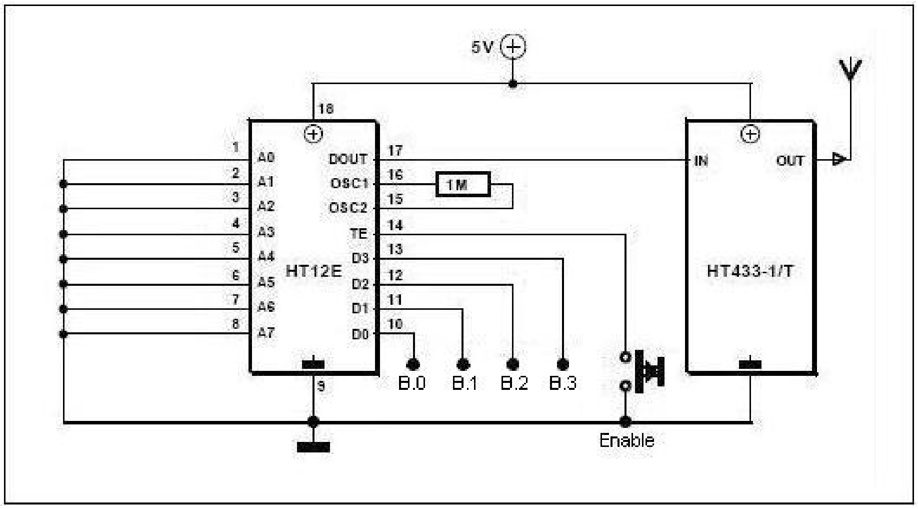


Figure 3.5: Encoder Circuit

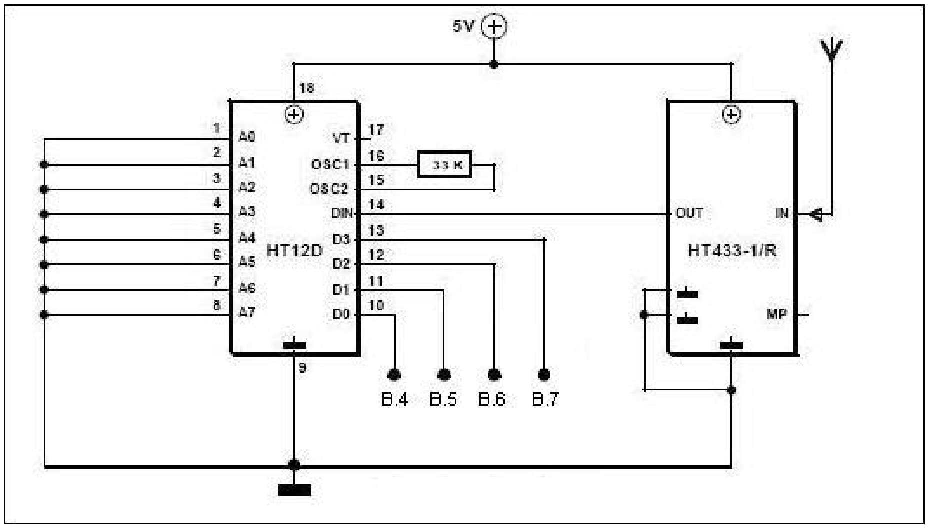


Figure 3.6: Decoder Circuit

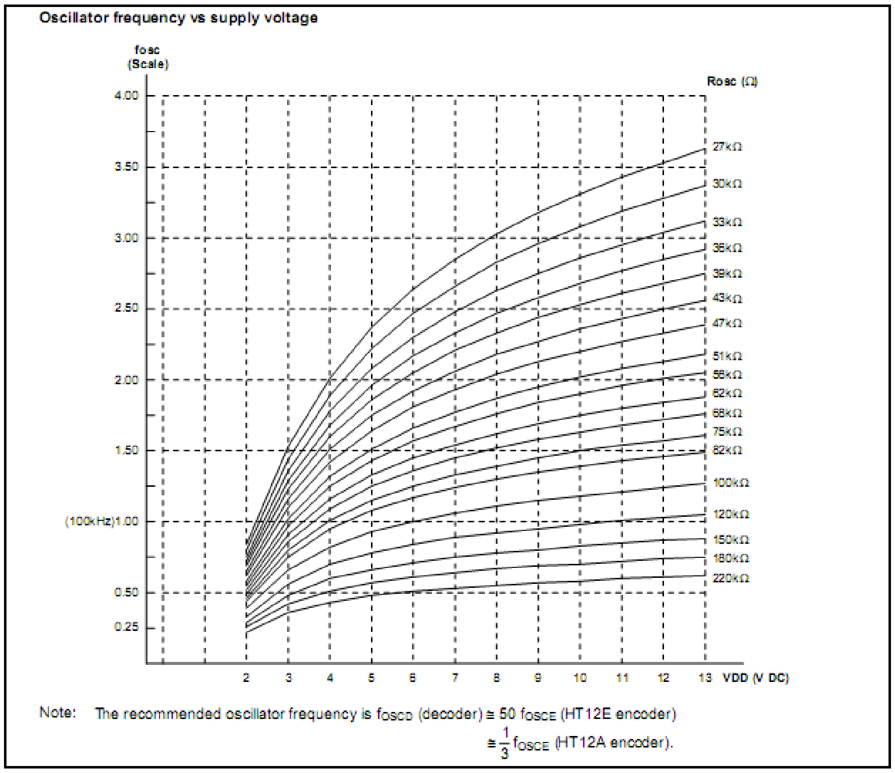


Figure 3.7: Decoder Frequency

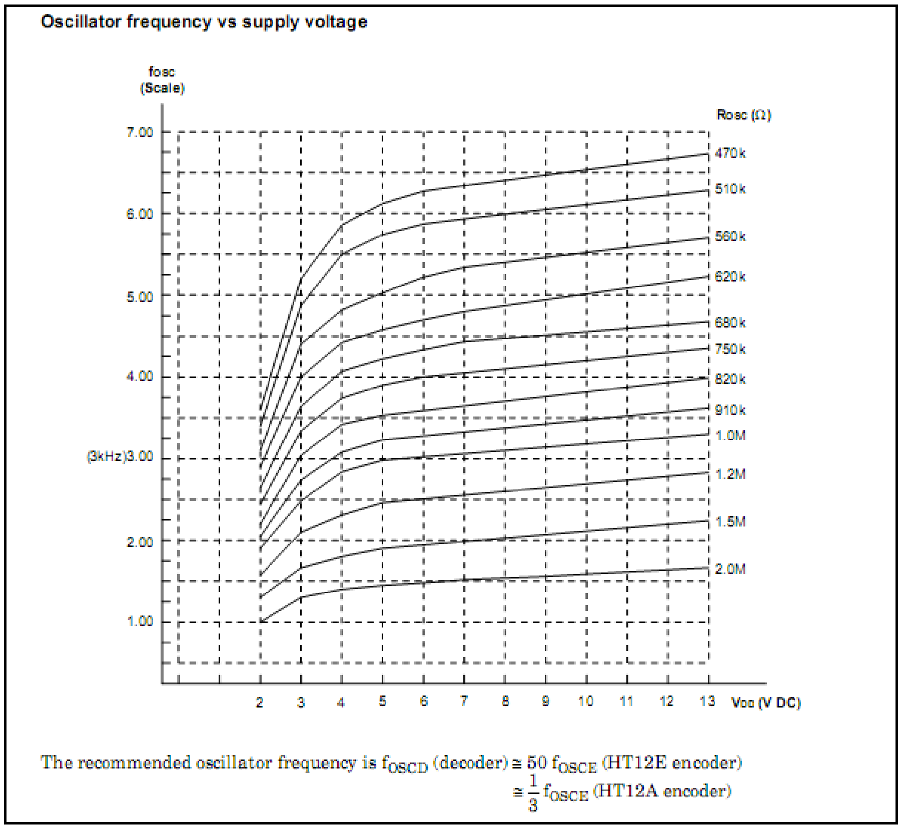


Figure 3.8: Encoder Frequency

3.2.3 LCD

The LCD is use in the transmitter board as angle and movement indicator. Most LCD conforms to a standard interface. A 14-pin access is provided having eight data lines, three control lines and three power lines.

Pin 1 and 2 are the power supply lines, V_{GND} and V_{CC}. Pin 3 is a control pin, V_{EE} which is used to alter the contrast of the display. Pin 3 is connected to a potentiometer. The potentiometer will control the contrast of the LCD. Pin 4 is the Register Select (RS) line, the first of the three command control input. When this line is low, data bytes transfer to the display are treated as command, and the bytes read from the display indicates its status. By setting the RS line high, character data can be transferred to and from the LCD. Pin 4 is connected to PORTE 1 of the PIC.

Pin 5 is the Read/Write (R/W) line. This line is set low in order to write commands or character to the LCD, or set high to read character data or status information from its register. In this project, R/W line is connected to ground as it only is used for transmitted data from PIC16F877A to LCD. Pin 6 is the Enable (E) line. This input is used to initiate the actual transfer of commands or character data between the LCD and data lines. When writing to the display, data is transferred only on the high to low transition of this signal. However, when reading from the display, data will become available shortly after the low to high transition and remain available until the signal falls to low again. The pin 6 is connected to PORTE 2 of the PIC.

Pin7 to pin 14 are the eight data bus lines (D0 to D7). Data can be transferred to and from the display, either as a single 8-bit byte or as two 4-bit “nibbles”. For this project, the entire pin data bus line is connected to PORTD.

The pin D0-D7 is connected to PORTD 0 – PORTD 7 of the PIC. The connection of the LCD is shown is Figure 3.9 below.

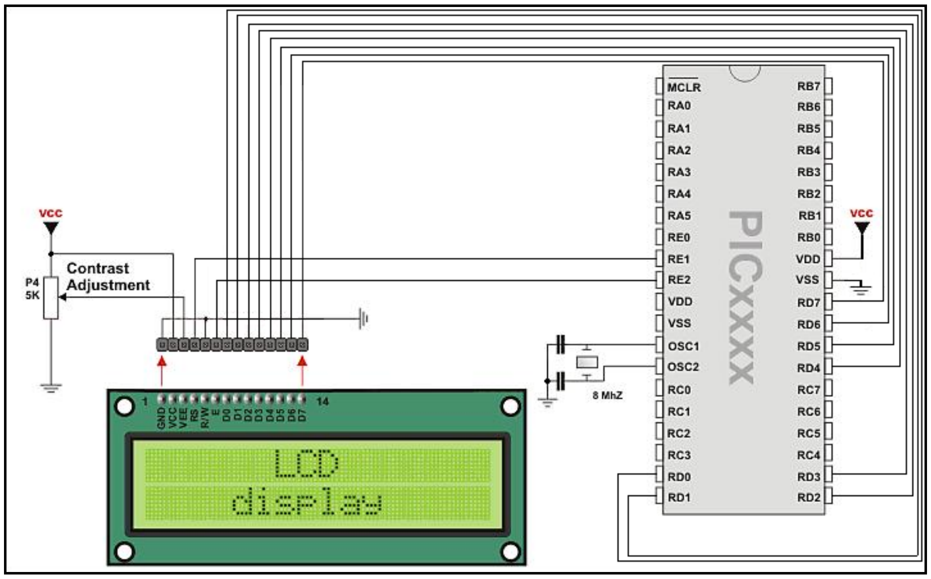


Figure 3.9: LCD circuit

3.2.4 Stepper Motor

In this part, we have to test functionality of the stepper motor. The stepper motor is connected to a stepper motor IC driver that is ULN 2003A. By using the ULN 2003A we can control the stepper motor. Figure 3.10 shows the connection of the stepper motor to the stepper motor driver and PIC.

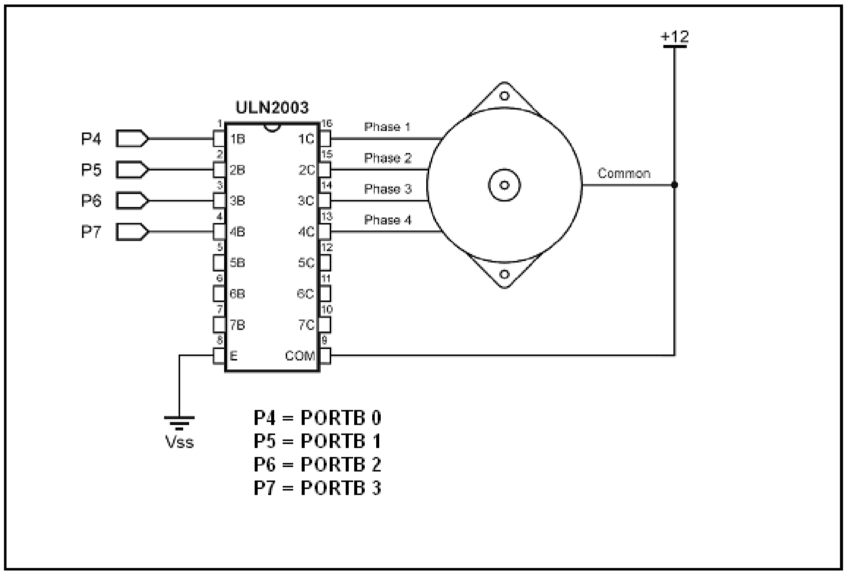


Figure 3.10: Stepper Motor Circuit

The PIC will send the sequence of data to the ULN 2003A and according to the data the stepper will move clockwise or anti clockwise. There are several stepping modes that can use to drive the stepper motor. The stepping modes are different according to the data that sent to the ULN 2003A.

Single stepping is the simplest mode that turns one coil ON at a time. 48 pulses are needed to complete one revolution. Each pulse moves rotor by 7.5 degrees. The following sequence has to be repeated 12 times to complete one revolution (Table 3.1).

Half Step = L, One Phase = H				
Step	**A**	**B**	**C**	**D**
POR	ON	OFF	OFF	OFF
1	ON	OFF	OFF	OFF
2	OFF	ON	OFF	OFF
3	OFF	OFF	ON	OFF
4	OFF	OFF	OFF	ON

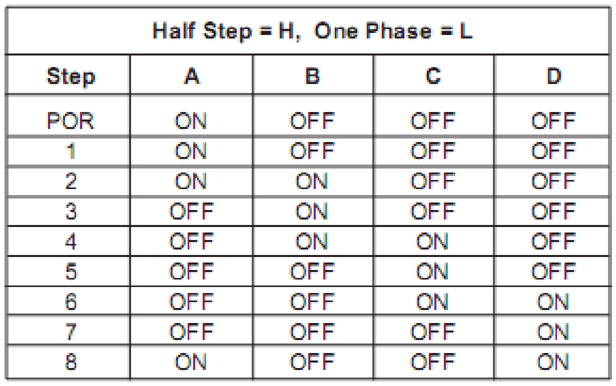
Table 3.1: Single Stepping Sequence

High torque stepping is a high power / precision that mode turns ON two coils at a time. 48 pulses are needed to complete one revolution. Each pulse moves rotor by 7.5 degrees. The following sequence has to be repeated 12 times for motor to complete one revolution (Table 3.2).

Half Step = L, One Phase = L				
Step	**A**	**B**	**C**	**D**
POR	ON	OFF	OFF	ON
1	ON	OFF	OFF	ON
2	ON	ON	OFF	OFF
3	OFF	ON	ON	OFF
4	OFF	OFF	ON	ON

Table 3.2: High torque Stepping Sequence

Half stepping is modes that doubled the stepping so that the motor needed 96 pulses to complete one revolution. Each pulse moves rotor by approximately 3.75 degrees. It is the mix of single stepping mode and high torque mode (Table 3.3).



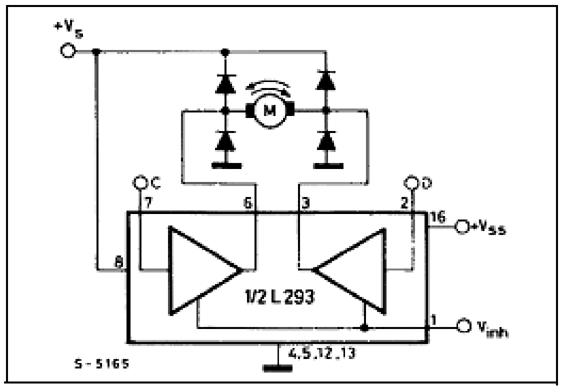
Half Step = H, One Phase = L				
Step	A	B	C	D
POR	ON	OFF	OFF	OFF
1	ON	OFF	OFF	OFF
2	ON	ON	OFF	OFF
3	OFF	ON	OFF	OFF
4	OFF	ON	ON	OFF
5	OFF	OFF	ON	OFF
6	OFF	OFF	ON	ON
7	OFF	OFF	OFF	ON
8	ON	OFF	OFF	ON

Table 3.3: Half Stepping Sequence

From the above data, we have found the suitable sequence data to move the stepper motor to archive the objective of the project. Result is shown in the next chapter.

3.2.5 Direct Current Motor

The DC motor is connected to the DC driver that is L 293B. The driver has a separate supply input for the logic so that it may be run off a lower voltage to reduce dissipation. The low voltage from the PIC can drive the high voltage DC motor. Figure 3.11 shows the connection of the DC motor to the DC driver motor.



Inputs	Function	
V_{inh} = H	C = H ; D = L	Turn Right
	C = L ; D = H	Turn Left
	C = D	Fast Motor Stop
V_{inh} = L	C = X ; D = X	Free Running Motor Stop

L = Low H = High X = Don't Care

Figure 3.11: DC motor Circuit

3.3 Hardware Implementation

The hardware implementation of the project can be divided into 5 main processes. They are:

I. Printed Circuit Board (PCB) Layout
II. PCB Layout Ironing
III. PCB Etching
IV. PCB Holes Drilling
V. PCB Soldering

3.3.1 Printed Circuit Board (PCB) Layout

The PCB layout for the project is design using Altium DXP. It is an electronics design software which provides a complete and diverse set of capabilities for electronic product development. Altium allows to better communicate the breadth of technologies integrated on its DXP platform – board-level system design and verification, FPGA-level system design and verification, embedded software development, CAM engineering, and design data, document and library management. Protel 2004's schematic editor provides full support for designs that contain repeated blocks of circuitry. Figure below shows that the example circuit and PCB development using DXP Protel. Figure 3.12 shows the software that is used for this project.

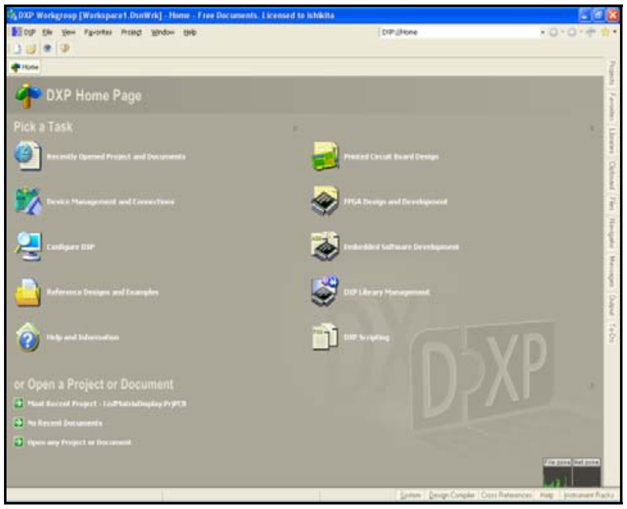
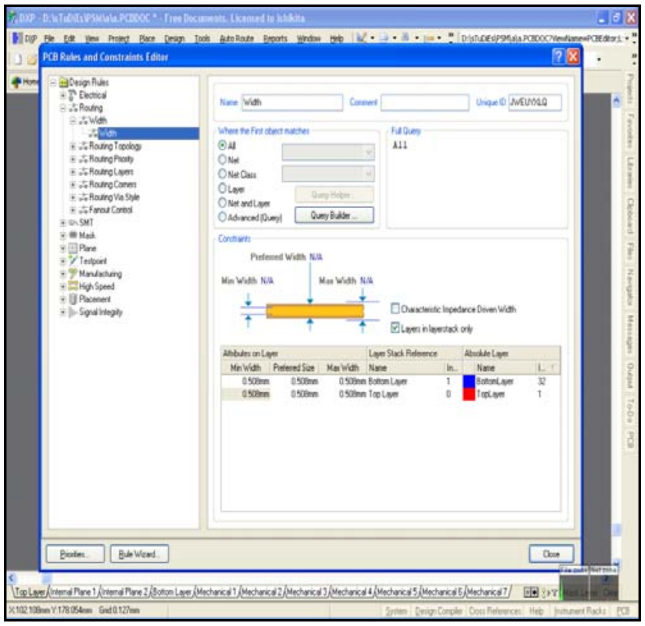

DXP Workgroup [Workspace1.DsnWrk] - Home - Free Documents. Licensed to ishikita
DXP File View Favorites Project Window Help
DXP://Home
Home
DXP Home Page
Pick a Task
Recently Opened Project and Documents
Printed Circuit Board Design
Device Management and Connections
FPGA Design and Development
Configure DXP
Embedded Software Development
Reference Designs and Examples
DXP Library Management
Help and Information
DXP Scripting
or Open a Project or Document
Most Recent Project - LedMatrixDisplay.PrjPCB
No Recent Documents
Open any Project or Document
DXP
System
Design Compiler
Cross References
Help
Instrument Racks

Figure 3.12: DXP Protel

The size of the board and the track should be setting as the same as the required by the components that will be soldered on the board. Figure 3.13 shows the board setting exactly the same specification with the strip board.



Name: Width Comment: Unique ID: JWEUYXLQ

Where the First object matches: [✓ All] Net, Net Class, Layer, Net and Layer, Advanced (Query)

Full Query: All

Constraints

Preferred Width N/A Min Width N/A Max Width N/A

☐ Characteristic Impedance Driven Width

☑ Layers in layerstack only

Attributes on Layer			Layer Stack Reference		Absolute Layer		
Min Width	Preferred Size	Max Width	Name	In...		Name	I...
0.508mm	0.508mm	0.508mm	Bottom Layer	1	🟦	BottomLayer	32
0.508mm	0.508mm	0.508mm	Top Layer	0	🟥	TopLayer	1

Figure 3.13: Track size setting

The board layout will be designed with the toolbars that has been provided by the software. The complete PCB layout for transmitter and receiver board is shown in figure 3.13 and figure 3.14.

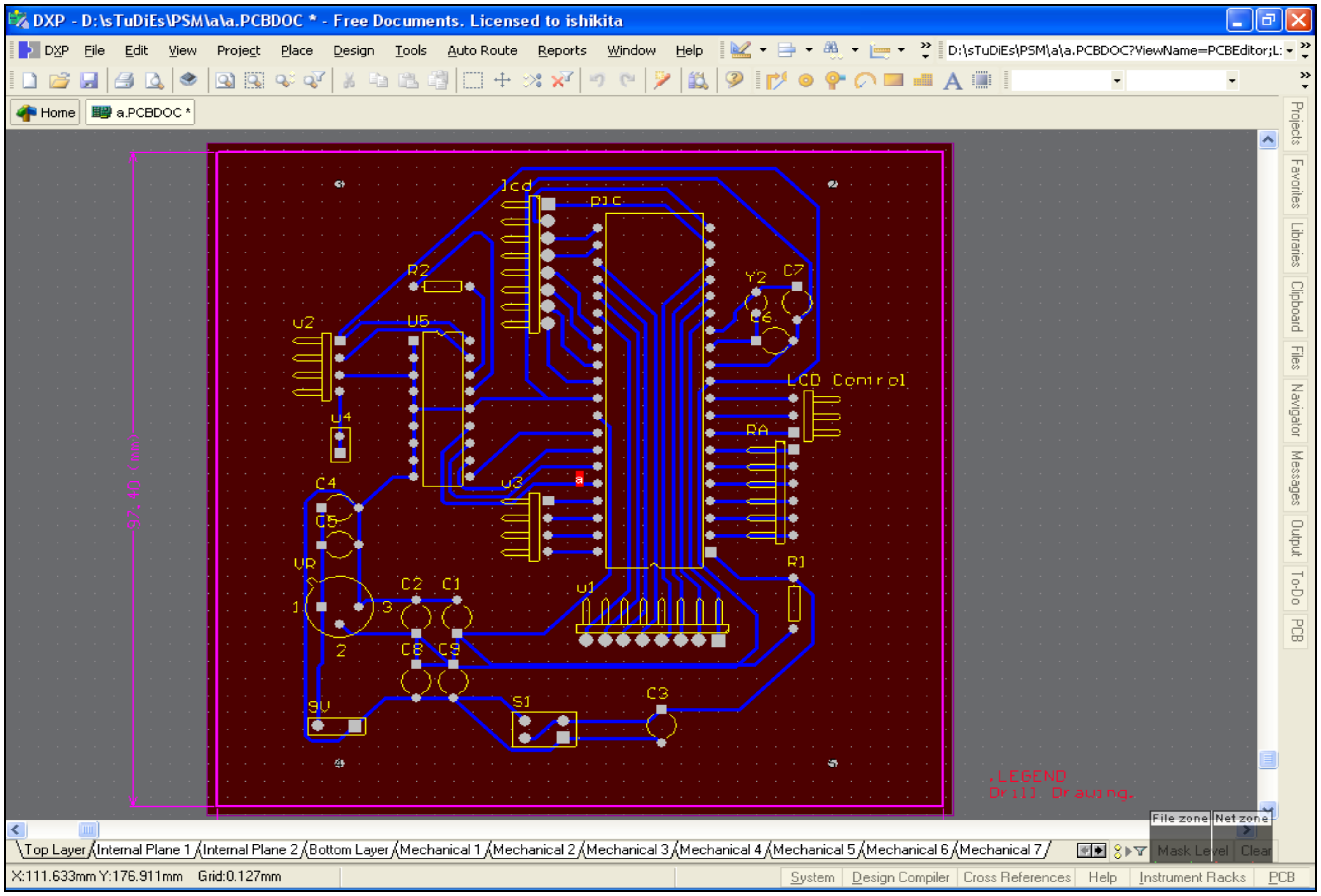


Figure 3.14: Complete Transmitter PCB layout

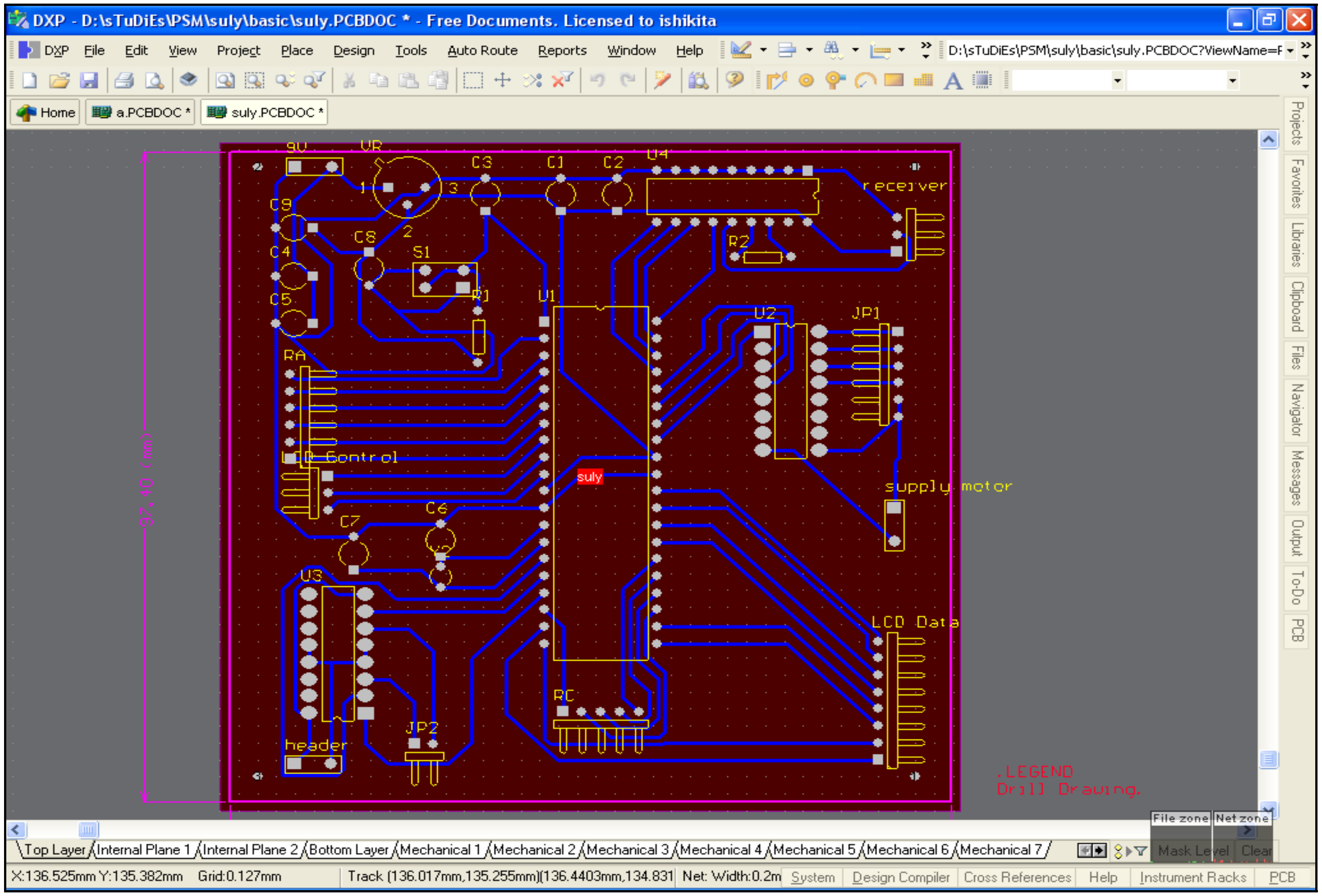


Figure 3.15: Complete Receiver PCB layout

3.3.2 PCB Layout Ironing

The complete PCB layout will be printed on the special paper that can be ironed into the PCB. The paper only can be printed by laser jet printer. There are certain things that should be confirm before the board will be ironed:

I. Clean the oxidize surface of the PCB
II. Any dirt on the surface should be removed to avoid the printed part from the paper will not stuck on the PCB.
III. The heat of the iron should be 75 percent of the total heat.
IV. Each surface that being ironed should be in duration of 3 – 4 minutes.
V. The ironed surface should not rub with the iron to avoid the tracks size change.

After the ironing process, the tracks of the layout on the PCB should be check if any of the tracks on the PCB is lost. ‘Touch Up’ process is required if any of the tracks is lost. Permanent marker is used to connect the lost of the tracks.

3.3.3 PCB Etching

Etching is the process where the excess copper is removed to leave the individual tracks or traces. Many different chemical solutions can be used to etch circuit boards. The etching also can be fastening by increasing the temperature of the chemical solution and increase the saturation of the reagent.

Certain acid are used to do the process. Normal copper etching reacted with the initial reaction, which results in build up of cuprous ions. The basic etching reaction is the same as cupric chloride etching but to work the copper (II) ions require complexing with ammonium chloride and ammonia. Fastening the etching process should be considered because if the process occurred to fast, it sometimes will remove the tracks of traces on the PCB surface. Personal Protection Equipment must always be used, spent solutions should always be disposed of properly and not down local drains, where they pollute local sewage works and rivers. The etching process for this project took about 2 hours.

3.3.4 PCB Holes Drilling

The process will produce the holes for the components before been soldered. The holes required should suitable enough with the component legs. The drill lead also should be suitable in size for avoiding the holes drilled to large or sometimes will remove the tracks of the cuprum. The drill lead use for the project is 0.8 mm in diameter.

The black printed tracks on the PCB will be clean for viewing the cuprum tracks that is not removed while the etching process been conducted. The black printed should be removed by using sand paper or pen eraser. The connection on the PCB is check for it continuity on the same connection or path. Figure 3.16 shows the complete PCB transmitter board after drilling.

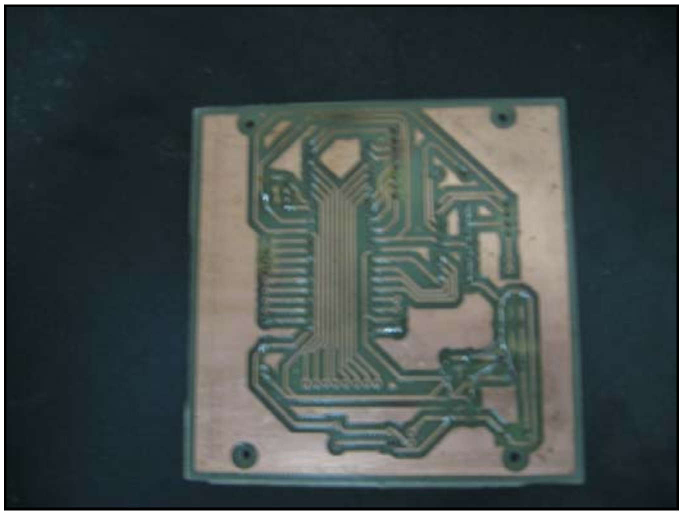


Figure 3.16: Complete Transmitter board after drilling

3.3.5 PCB Soldering

This is the last stage of the hardware implementation where the whole components of the project are soldered on the PCB. The components will be soldered and resoldered is avoided to prevent the cuprum tracks from damages. The soldered PCB is encouraged to be coated with the layer that will prevent the cuprum from being deoxidize. Figure 3.17 and 3.18 shows the PCB board after being soldered with the component of the project.

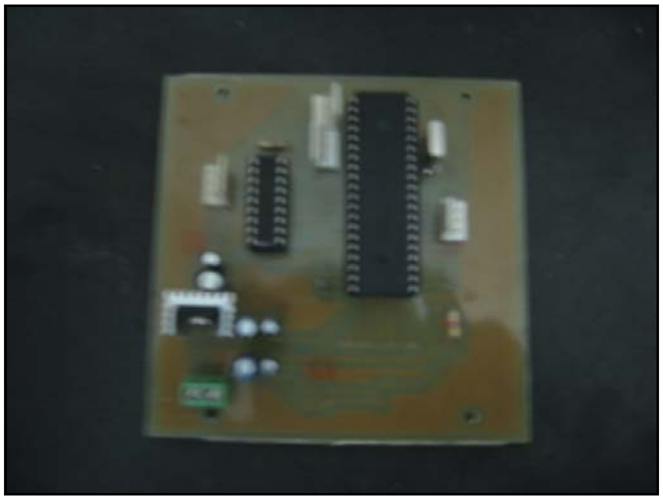


Figure 3.17: Front view of the PCB transmitter board after being soldered

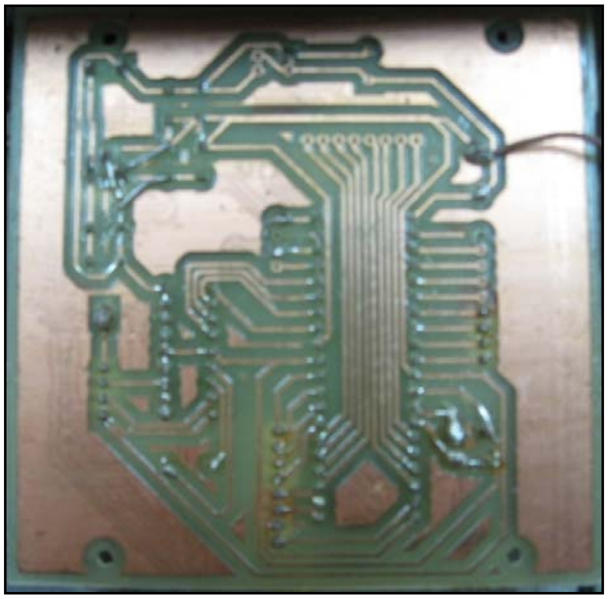


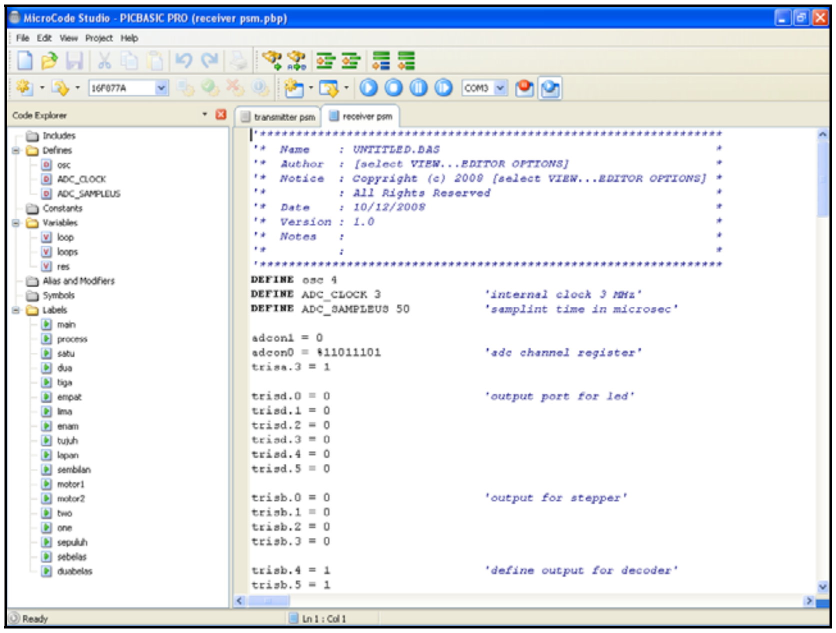
Figure 3.18: The PCB transmitter board after being soldered

3.4 Software Implementation

This section will discuss about the software that has been implemented in this project. There are two main software used which is PICkit™ 2 Microcontroller Programmer and PicBasic Pro Compilers (Micro Code Studio). The PICkit™ 2 Microcontroller Programmer is used for uploading the programming into the PIC and PicBAsic Pro Compilers is used to design the programming.

3.4.1 Programming the PIC16F877A Microcontroller

Micro Code Studio is a powerful, visual Integrated Development Environment (IDE) with In Circuit Debugging (ICD) capability designed specifically for microEngineering Labs PICBasic PRO compiler. The code explorer allows you to automatically jump to include files, defines, constants, variables, aliases and modifiers, symbols and labels that are contained within your source code. It's easy to set up the compiler, assembler and programmer options or the Micro Code Studio can do it automatically with its built in auto search feature. Compilation and assembler errors can easily be identified and corrected using the error results window. Figure 3.19 shows the PicBasic Pro Compilers (Micro Code Studio) interface.



```
'****************************************************************
'*  Name    : UNTITLED.BAS                                      *
'*  Author  : [select VIEW...EDITOR OPTIONS]                    *
'*  Notice  : Copyright (c) 2008 [select VIEW...EDITOR OPTIONS] *
'*          : All Rights Reserved                               *
'*  Date    : 10/12/2008                                        *
'*  Version : 1.0                                               *
'*  Notes   :                                                   *
'*          :                                                   *
'****************************************************************
DEFINE osc 4
DEFINE ADC_CLOCK 3                  'internal clock 3 MHz'
DEFINE ADC_SAMPLEUS 50              'samplint time in microsec'

adcon1 = 0
adcon0 = %11011101                  'adc channel register'
trisa.3 = 1

trisd.0 = 0                         'output port for led'
trisd.1 = 0
trisd.2 = 0
trisd.3 = 0
trisd.4 = 0
trisd.5 = 0

trisb.0 = 0                         'output for stepper'
trisb.1 = 0
trisb.2 = 0
trisb.3 = 0

trisb.4 = 1                         'define output for decoder'
trisb.5 = 1
```

Figure 3.19: PICBasic Pro Compilers (Micro Code Studio) interface

The programming of the PIC is divided into 7 part that is programming the free running test, programming the transmitter and receiver part, programming the potentiometer, programming the LCD, programming the dc motor, programming the stepper motor and combining all 6 programming parts. The result will be shown in the next chapter.

Figure 3.20 shows the transmitter board flowchart meanwhile figure 3.21 shows the receiver board flowchart.

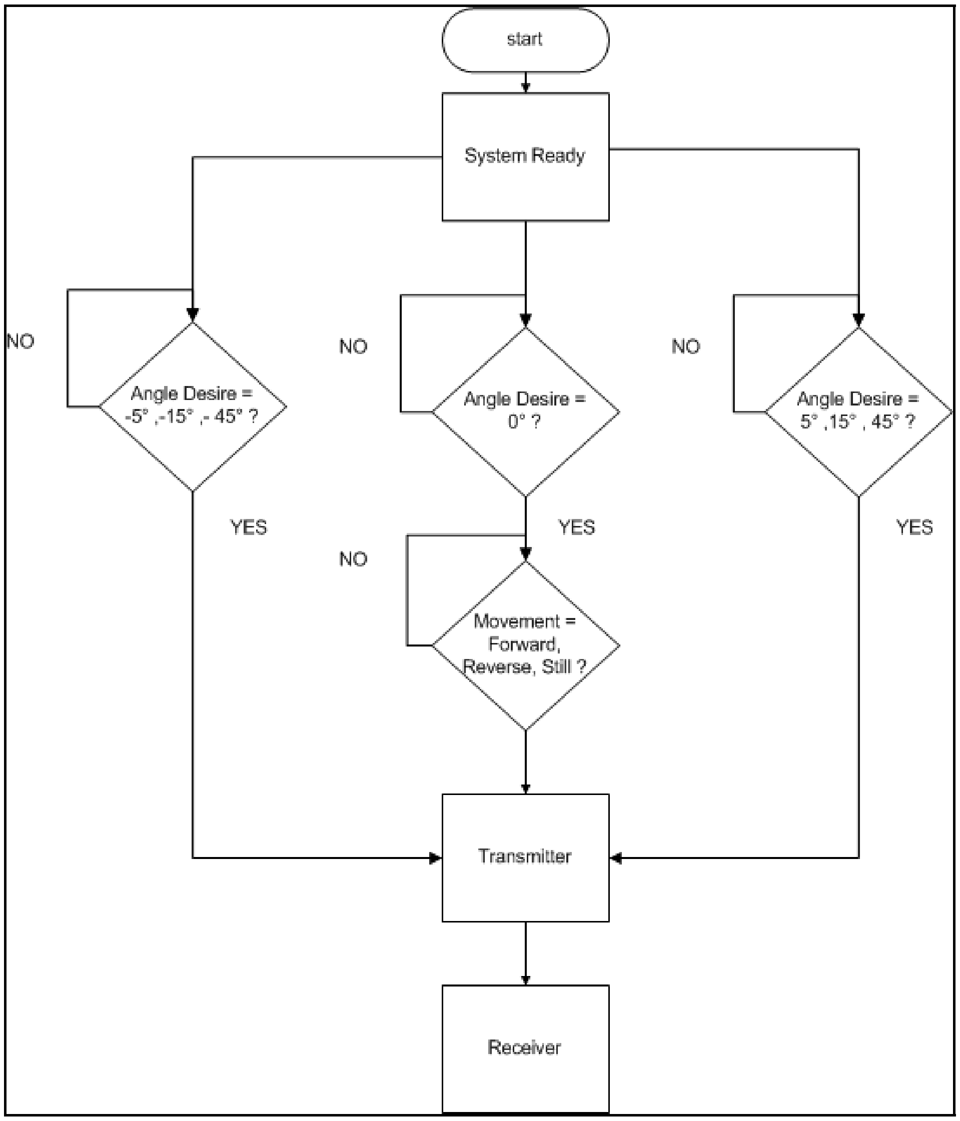

start
System Ready
NO
Angle Desire = -5° ,-15° ,- 45° ?
NO
Angle Desire = 0° ?
NO
Angle Desire = 5° ,15° , 45° ?
YES
YES
YES
NO
Movement = Forward, Reverse, Still ?
Transmitter
Receiver

Figure 3.20: Flowchart of the Transmitter board

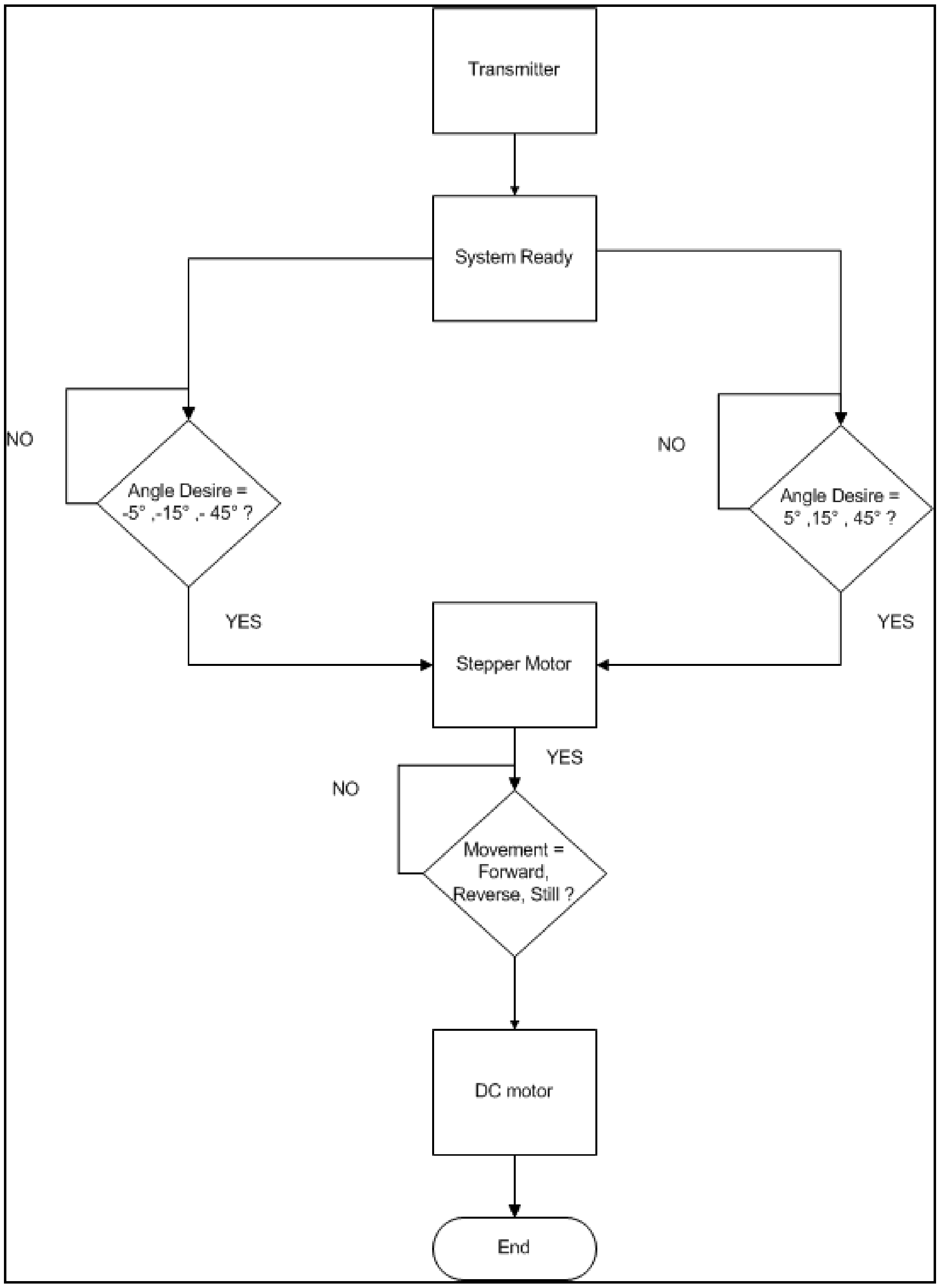
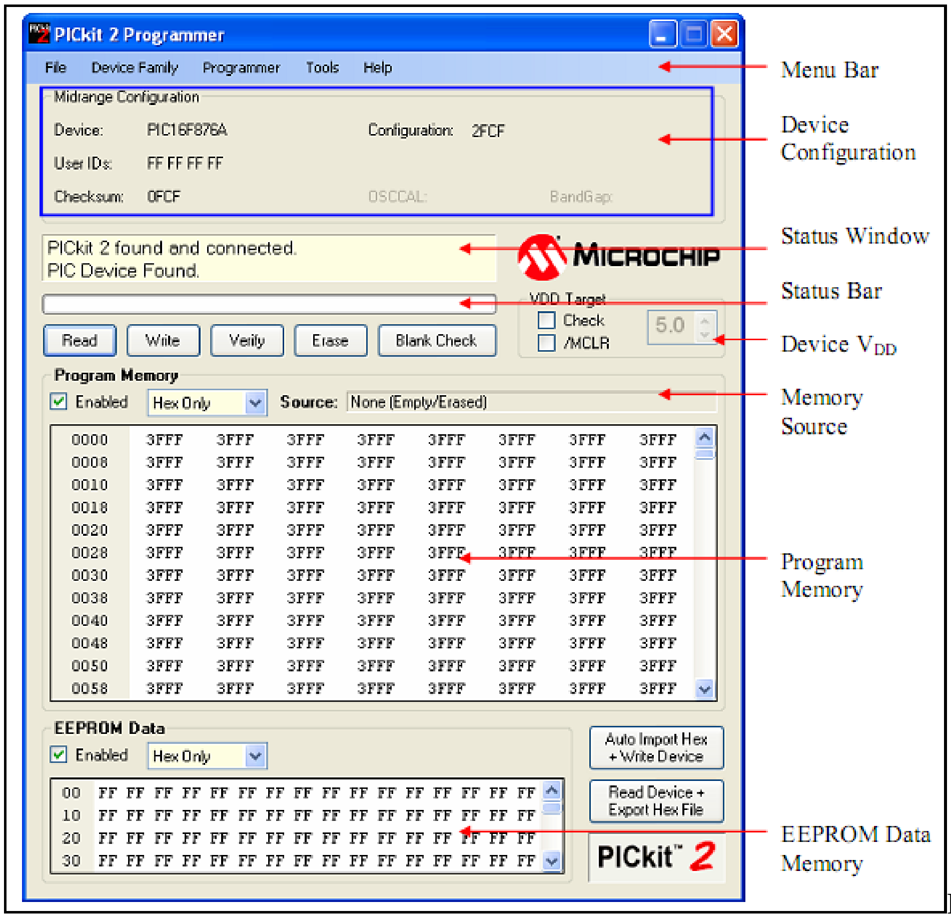

Transmitter
System Ready
NO
Angle Desire = -5° ,-15° ,- 45° ?
NO
Angle Desire = 5° ,15° , 45° ?
YES
YES
Stepper Motor
YES
NO
Movement = Forward, Reverse, Still ?
DC motor
End

Figure 3.21: Flowchart of the Receiver board

3.4.2 The PICkit™ 2 Microcontroller Programmer

The PICkit™ 2 Microcontroller Programmer is a low cost development programmer. It is capable of programming most of Microchip's Flash microcontrollers. This programmer is able to automatically detect PIC from connected target and display it in the Device Configuration window. Figure 3.22 shows the PICkit 2 programmer interface.



```
Midrange Configuration
Device:     PIC16F876A          Configuration:  2FCF
User IDs:   FF FF FF FF
Checksum:   0FCF                OSCCAL:                 BandGap:

PICkit 2 found and connected.
PIC Device Found.

VDD Target
[ ] Check      5.0
[ ] /MCLR

Program Memory
[✓] Enabled   Hex Only   Source: None (Empty/Erased)

0000  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
0008  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
0010  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
0018  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
0020  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
0028  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
0030  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
0038  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
0040  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
0048  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
0050  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
0058  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF

EEPROM Data
[✓] Enabled   Hex Only

00  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
20  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
30  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

Auto Import Hex + Write Device
Read Device + Export Hex File
```

Figure 3.22: PICkit 2 Microcontroller Programmer Interface

When the PIC is detected, we can check whether the PIC is blank or already have a program using *Blank Check* function. It helps us to check the status of the PIC. For the *Erase*, it helps us to delete the programming that is in the PIC. We can clear the programming in the PIC and the PIC is ready to be uploaded with a new programming. We can upload the programming to the PIC using *Write* function.

Verify function verify the device program to the imported Hex file. *Read* function is to view the code written to the PIC. The code will be display in the Program Memory and Data EEPROM Memory. If all zero display, it is possible that the target device is code-protected.

Auto Import Hex + Write Device allows the programmer to automatically import and write the Hex file to the connected device when the Hex file is updated, for an example on a new firmware build. By clicking this button, it will bring up an Import Hex File dialog.

The PICkit 2 programmer transfer the programming to the UIC00A (Figure 3.23). The UIC00A is a hardware that transfers the programming from the PICkit 2 programmer into the PIC. The UIC00A contain 5 main components that is mini USB socket, switch to initiate write device programming, main power supply indicator, busy indicator and box header for programming connector (Table 3.4).

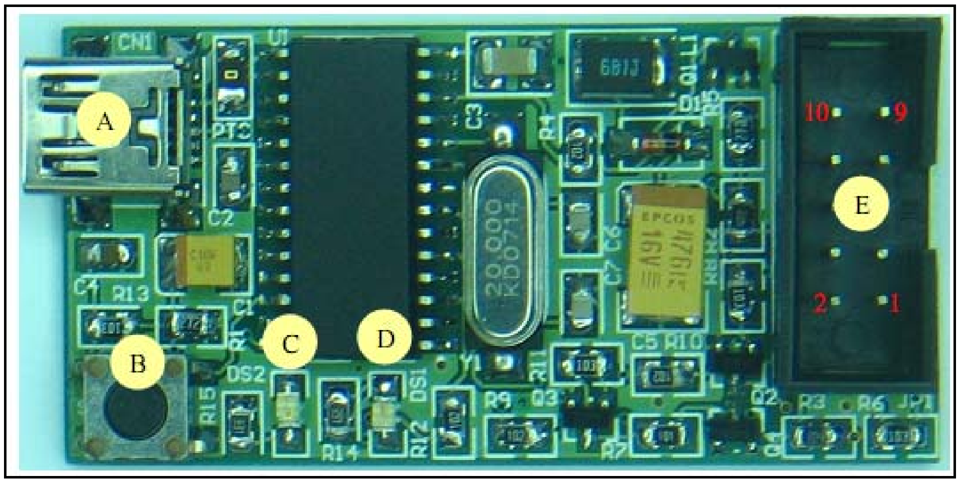
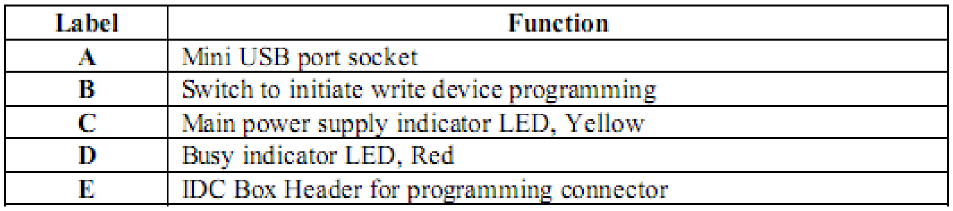


Figure 3.23: UIC00A Board Layout



Label	Function
A	Mini USB port socket
B	Switch to initiate write device programming
C	Main power supply indicator LED, Yellow
D	Busy indicator LED, Red
E	IDC Box Header for programming connector

Table 3.4: PICkit board component

The mini USB port socket is for USB connection to PC desktop or laptop. The mini USB is used to transfer the programming HEX into the board. The switch is used to initiate the write device function when the *Write* function on the PICkit programmer is checked. The yellow LED indicates the main power supply of UIC00A. It should be ON once USB connection from UIC00A to computer or Laptop is ready. The red LED indicates busy function such as UIC00A is in program mode or is alerting that a function is in progress. The box header is used for connecting programming cable to the development board.

The development board is connected to the UIC00A through the ribbon cable and the ribbon cable is connected to the box header in UIC00A board. The circuit for the development is shown in Figure 3.24.

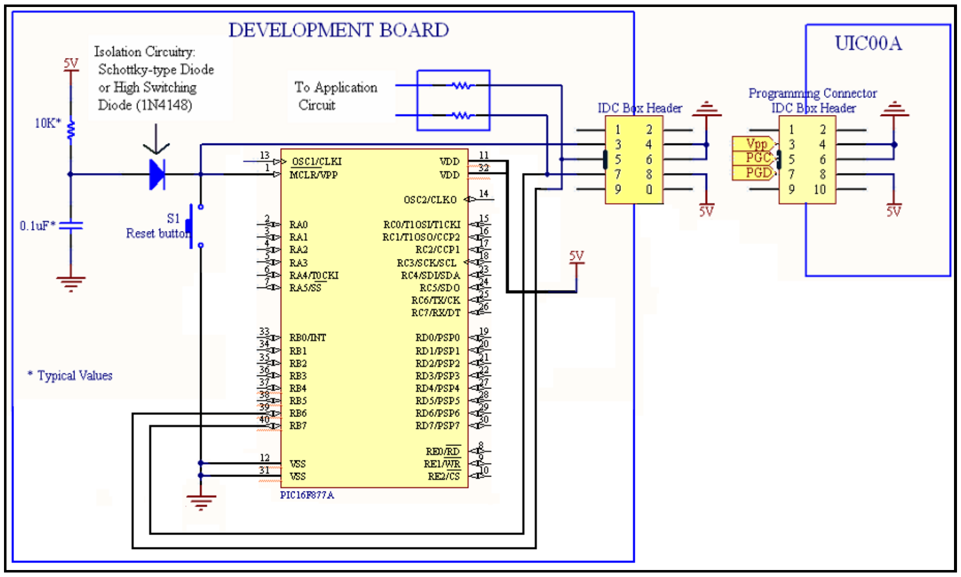


Figure 3.24: Development Board Circuit

3.5 Application

After all the above stage is complete, the process will continue into making the model of the remote control and the car. The remote control model is build is using a box and the model car is built from the recycle toy car (Figure 3.25).

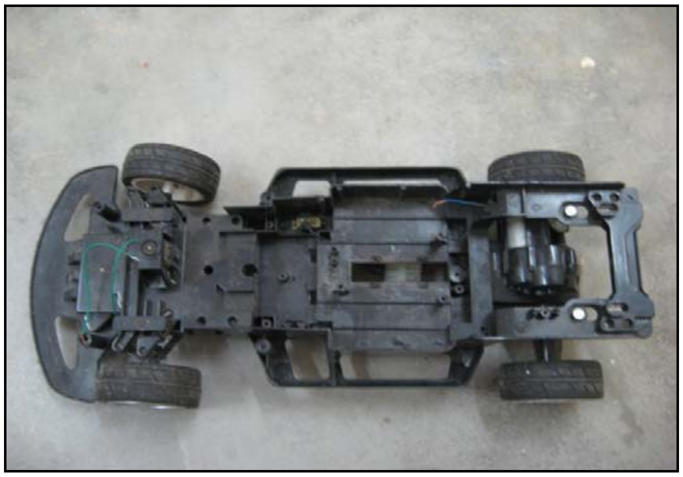


Figure 3.25: The recycle toy car

The box is spray into black colour to make it more realistic (Figure 3.26). The used toy car board (Figure 3.27) is removed and replace with the receiver board (Figure 3.28) that had been successfully passed in the hardware implementation stage. The toy car uses DC motor and gears to maneuver the car. There are 3 gears and one spring in the car maneuver compartment. The spring is used to move the tyres to the original position.

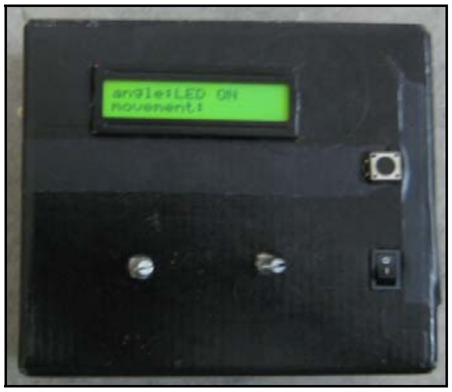


Figure 3.26: Remote control is spray into black colour

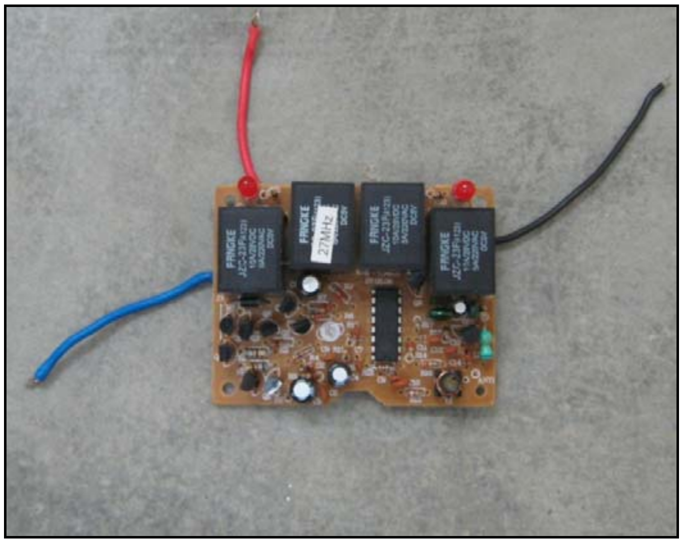


Figure 3.27: The original board

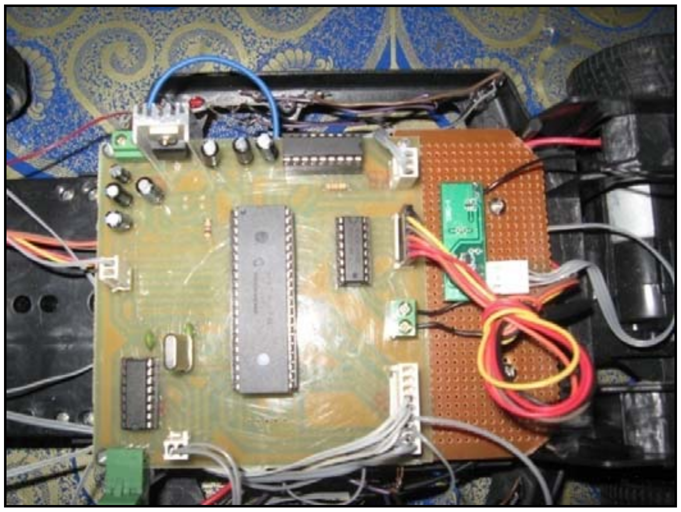


Figure 3.28: Implementing Receiver Board into the model

For this project, we are using stepper motor instead of DC motor to control the maneuver of the car. The DC motor is change to the stepper motor because stepper motor have more precise angle than DC motor. The gears are modified to suit the stepper motor. The spring is removed because the stepper motor will control all the maneuver of the car.

The DC motor holder is small to hold the stepper motor, so the holder is modified to hold the stepper motor. The stepper motor gear is modified because it did not reach the gears in the maneuver compartment. Figure 2.39 shows the stepper motor after being modified.

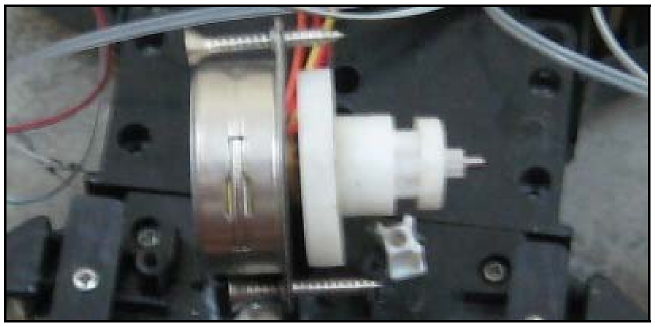


Figure 3.29: Modified stepper motor

Each pair of a little (12-tooth) and large (24-tooth) gears steps the speed down by a factor of one quarter (12/24). There are 3 pairs of gears (Figure 3.30), so the whole trains steps the speed down by $\frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} = \frac{1}{8}$. Now for every 8 turn of the stepper motor the wheel turns only one.

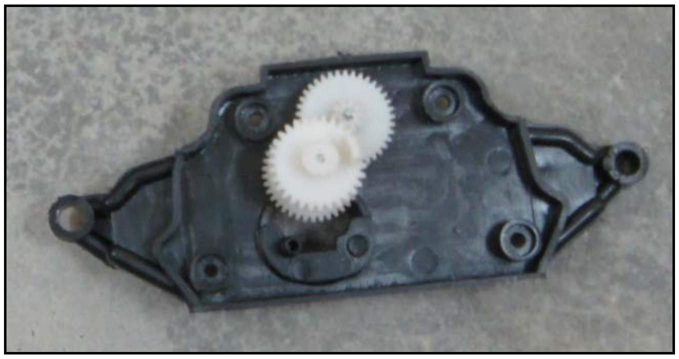


Figure 3.30: Gears position

The toy car have 2 motor that is the stepper motor that is used to control the maneuver of the car and the DC motor that is used for movement of the car. The full complete model will be shown in the next chapter.

CHAPTER 4

RESULTS & DISCUSSION

4.1 Introduction

In this chapter, the function of circuit and operating of the transmitter and receiver will be discussed according to the flow of program and the result that obtained from this project. The analysis is divided into 7 parts:

i. Power supply output voltages
ii. Receiver and transmitter testing
iii. LCD
iv. Stepper Motor
v. DXP protel
vi. PICBAsic Pro
vii. Application

4.2 Power Supply Output Voltages

The analysis of supply voltage is important to ensure the voltage is not exceeding the required value. Electronic component such as PIC just need 5V to operate. If the component is supplied more than that, the component will be hot and in some cases it can blow. So it is important to analyze the input voltage to all components to ensure it get the required supply voltages.

Figure 4.1 shows the outputs of a power supply. This supply came from LM 7805 voltage regulator, enters the PIC. All the circuits in this project required 5V power supply including LCD. From the figure we can see that the output is 5.12V which match the supply voltages that PIC and other components needed. It is very smooth and the voltage is near to the desired value.

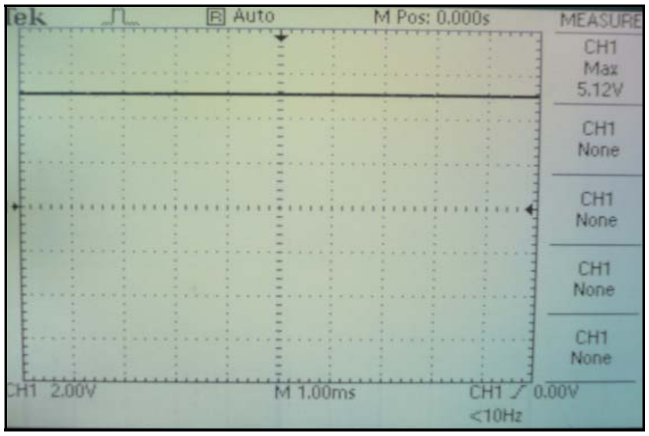


Figure 4.1: Power Supply Output Voltage

4.3 Transmitter and receiver

The result from the oscilloscope for transmitter testing is shown in Figure 4.2.

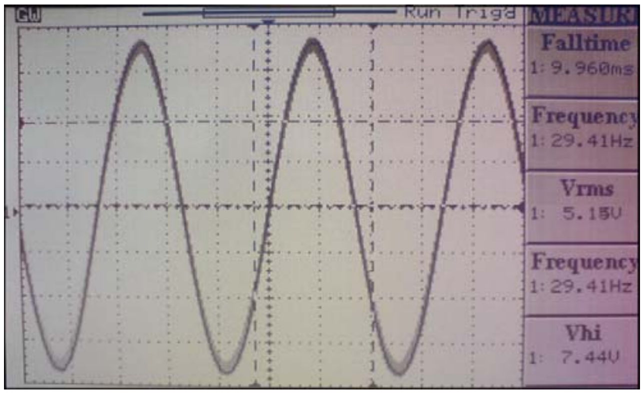


Figure 4.2: Transmitter Reading

The result from the oscilloscope for receiver testing is shown in Figure 4.3.

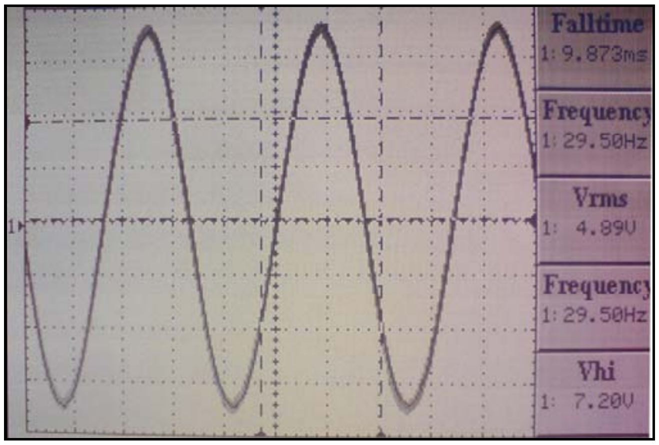


Figure 4.3: Receiver Reading

The result for the range of the transmitter is shown in Figure 4.4.

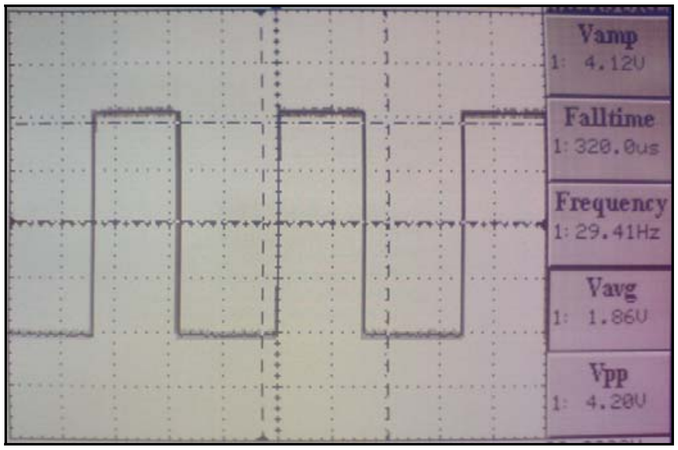


Figure 4.4: Transmitter and Receiver Reading

Range (cm)	Voltage (V)
50	2.90
100	2.50
150	2.20
200	1.90
300	1.50

Table 4.1: Transmitter and Receiver Range

From the testing result, the transmitter and the receiver can be used in a range of 5 meter without using an antenna. The antenna can lengthen the range up to 7 meter. We are using a straight antenna to get to the fullest range.

The encoder is used to send data serially to the receiver. The data is sent according to the frequency of the encoder. We choose to use a 1MΩ resistor and according to the datasheet it will produce around 1.9 kHz – 3.5 kHz. The transmitter and the receiver must be in the same frequency so that it will function properly. To choose the frequency for the receiver, we must follow the following rule:

$$f_{OSCD} \text{ (HT12D decoder)} = 50\, f_{OSCE} \text{ (HT12E encoder)}$$

4.4 LCD

This part will discuss about the operation of LCD and result obtained from it. The potentiometer is use to control the brightness of the LCD. The potentiometer is set to the desired value according to the brightness of the LCD. Before programming, we can only see the upper line display of the LCD (Figure 4.5). After the programming is done, we can program the upper and lower line display of the LCD (Figure 4.6).

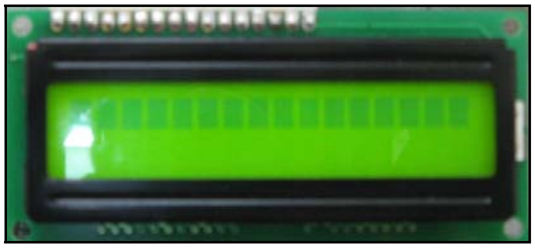


Figure 4.5: The Upper line visible

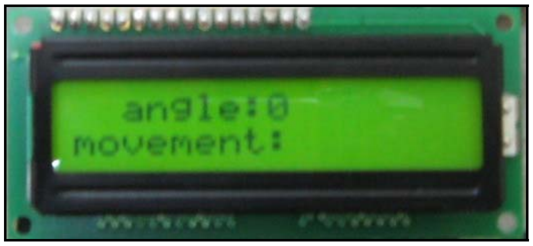


Figure 4.6: Upper and lower line Display

The LCD is program to display the angle and the movement according to the single turn potentiometers that acts as joysticks for the system. There are two single turn potentiometer that are used, one is used to control angle and the other one is use to control movement. When the potentiometer for the angle is turned, the angle will change (Figure 4.7) and when the potentiometer for the movement is turned, the movement will change (Figure 4.8). Figure 4.9 shows the LCD is inserted into the slot at the remote control box.

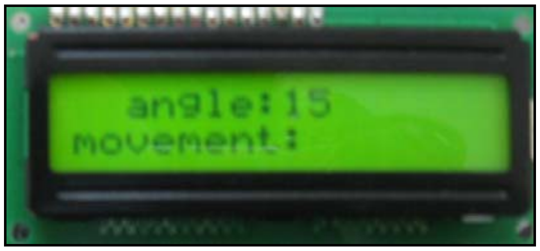


Figure 4.7: Changing the angle

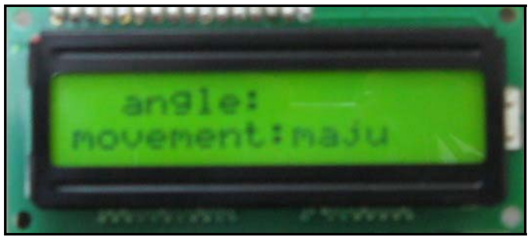


Figure 4.8: Changing movement

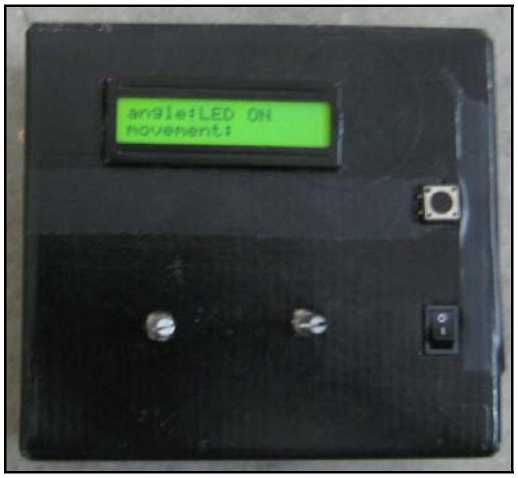


Figure 4.9: LCD is inserted into the slot at the remote control box

4.5 STEPPER MOTOR

The stepper motor needs a data sequence to work properly. For the hardware testing part, we found that the sequence in Table 4.1 is the most suitable one for the project.

Half Step = L, One Phase = L				
Step	**A**	**B**	**C**	**D**
POR	ON	OFF	OFF	ON
1	ON	OFF	OFF	ON
2	ON	ON	OFF	OFF
3	OFF	ON	ON	OFF
4	OFF	OFF	ON	ON

Table 4.2: High torque Stepping Sequence

High torque stepping is a high power / precision mode that turns ON two coils at a time. 48 pulses are needed to complete one revolution. Each pulse moves rotor by 7.5 degrees. The following sequence has to be repeated 12 times for motor to complete one revolution.

During the testing time, a problem has occur, the ULN 2003 and the stepper motor cannot withstand the current flow from the battery. The stepper motor and the ULN 2003A always heated if use in a long period. To protect the component from damage, the usage of the stepper motor and the ULN 2003A have been minimized.

4.6 DXP protel

DXP Protel is used to design PCB board. During this time there are many problems occurred. The connection of the PCB is cannot be made because the path of the component is cross into one another. The path have been reconstructed and compiled. Figure 4.10 shows the successful transmitter board layout and Figure 4.11 shows the successful receiver board layout.

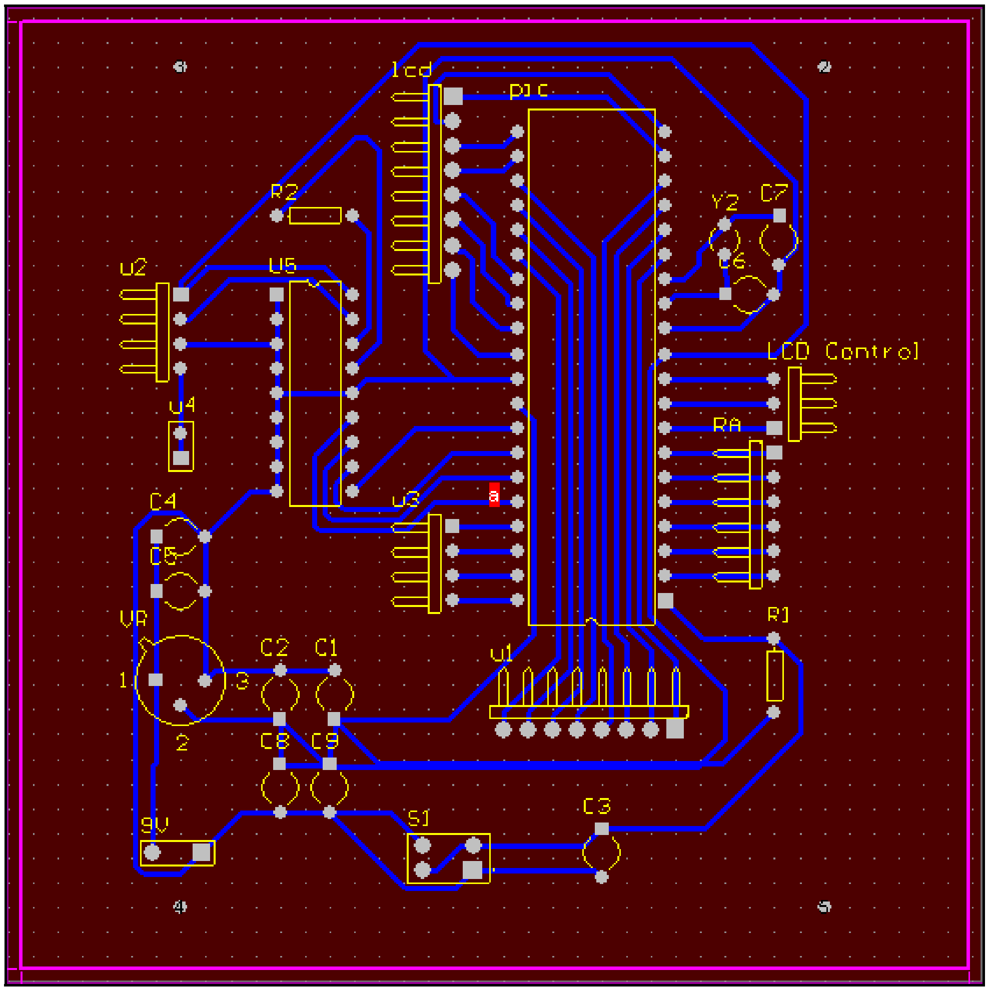


Figure 4.10: Successful Transmitter Board Layout

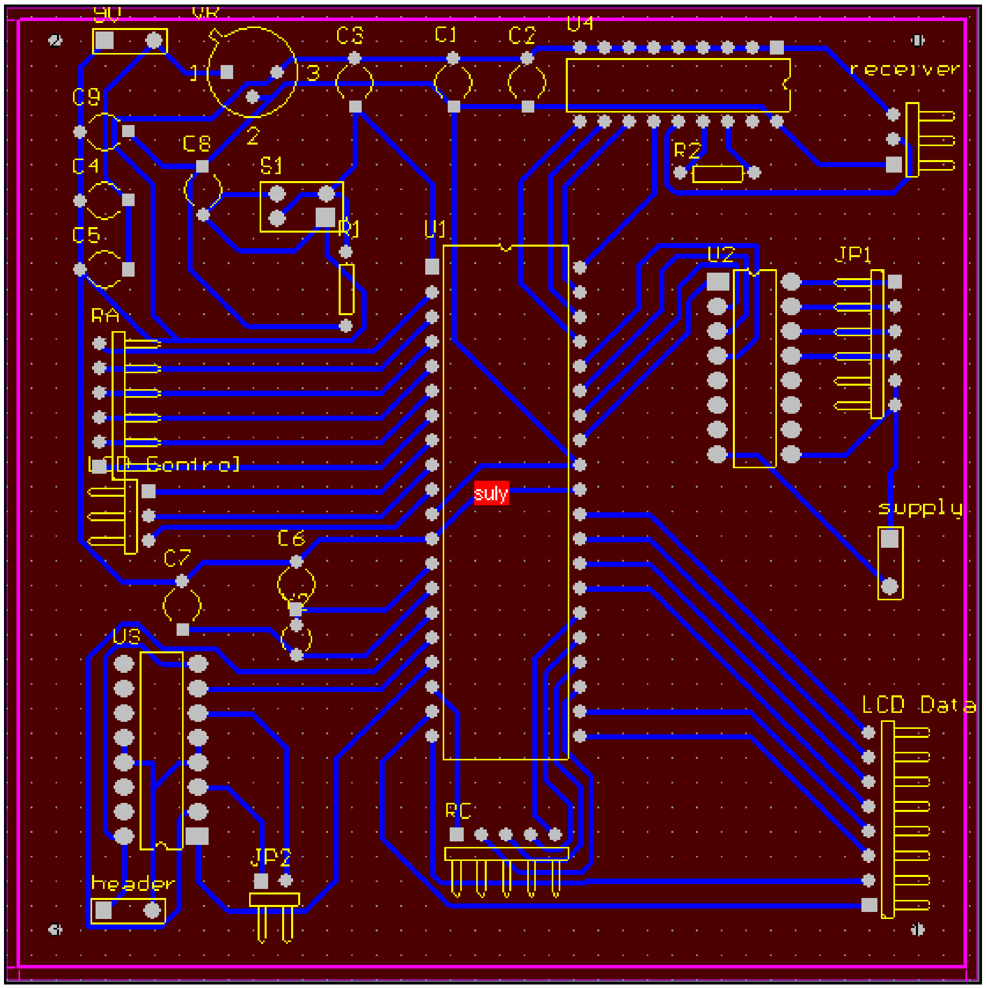


Figure 4.11: Successful Receiver Board Layout

4.7 PICBasic Pro

Programming is divided into 7 modules that are programming the free running test, programming the transmitter and receiver, programming the potential meter and photo resistor (LDR), programming the LCD, programming the stepper motor, programming the DC motor and combining all 6 modules. For combining all 6 modules, the programming is shown in Appendix C.

4.7.1 Programming the free running

The free running programming is the simplest programming in the whole system. The programming is a basic programming that can help us familiarize with defining the register and setting the clock of the system. We have to define the clock and register of the system like below.

```
Define osc 4
trisb.7 = 0
```

The clock of the system is define to 4MHz and the PORTB 7 is define as an output. The program will flow according to the line. After defining the register, the program will flow to the main program. The main program will be set as below:

```
Main:
High portb.7
pause 1000
Low port.7
pause 1000
goto Main
```

The LED will be ON for one seconds and follow by OFF for one seconds. The system will loops until it is shut down or the reset button is pressed.

4.7.2 Programming the transmitter and receiver

For the transmitter and receiver programming, it is nearly the same as the free running program. We have to define the register to set the output or input of the system. For example, we use a push button to ON LED using transmitter and receiver function. The transmitter has 4 outputs and one input, the register is define as below:

```
Define osc 4
trisb.0 = 0
trisb.1 = 0
trisb.2 = 0
trisb.3 = 0
trisb.4 = 1
```

There are 4 port that has been define for output because the 4 output from the PIC will be connected to the encoder and will be became the input for the encoder. The encoder will sent data serially to the transmitter. The data will be sent when the push button is press, the command is as below:

```
main:
if portb.4 = 1 then LED ON
Goto main
LED ON:
portb = %00000001
goto main
```

The program will loop until the system is shut down or the reset button is pressed. For the receiver programming, we have to define the register to allow the PIC to process the data that received from the transmitter. The register define is as below:

```
Define osc 4
trisb.4 = 1
trsib.5 = 1
trisb.6 = 1
trisb.7 = 1
trisb.3 = 0
```

There are four inputs because the receiver is connected to the decoder. The data that are received from the transmitter are sent to the decoder serially and the decoder sent the data to the PIC. The PIC processed the data and implemented it to the output. The main program for the receiver is as below:

```
main:
if portb.3 = 1 then LED ON
goto main

LED ON:
HIGH portb.3
pause 1000
goto main
```

The LED will be continuously ON until the puss button is released. The system will loops until it is shut down or the reset button is pressed.

4.7.3 Programming the potentiometer

For programming the potentiometer, we use ADCIN function in the PIC to ease our programming. ADCIN allow the PIC to read the on-chip analog to digital converter Channel and store the result in a variable. Before the ADCIN can be used, the appropriate register must be set to make the desired pin input. ADCON1 also needs to be set to assign the desired pins to analog inputs and in some cases to set the result format and clock source (Refer to the data sheet in the Appendix).The PIC 16F877A has an 8-bit ADC.

Several DEFINEs may also be used. The defaults are shown below:

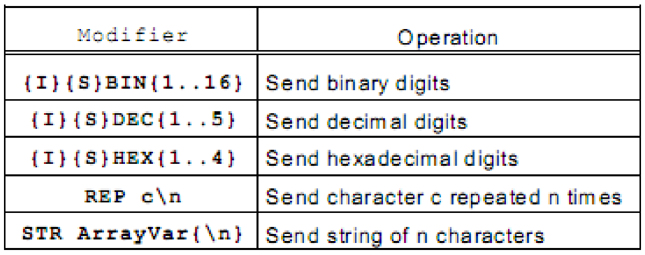
```
DEFINE ADC_BITS 8            'Set number of bits in result 8'
DEFINE ADC_CLOCK 3           'Set clock source (rc =3)
DEFINE ADC_SAMPLEUS 50       'Set sampling time in
                               microseconds'
```

ADC_SAMPLEUS is the number of microseconds the program waits between setting the Channel and starting the analog to digital conversion. This is the sampling time. The minimum number of microseconds usable is determined by the minimum time for PAUSEUS.'

```
TRISA = 255         'Set PORTA to all input'
ADCON1 = 0          'PORTA is analog'
ADCIN 0, B0         'Read channel 0 to B0'
```

4.7.4 Programming the LCD

PICBasic Pro supports LCD modules with a Hitachi 44780 controller or equivalent. These LCDs usually have a 14- or 16-pin single- or dual-row header at one edge. If a pound sign (#) precedes an Item, the ASCII representation for each digit is sent to the LCD. Table 4.3 shows the specified modifier for LCD.



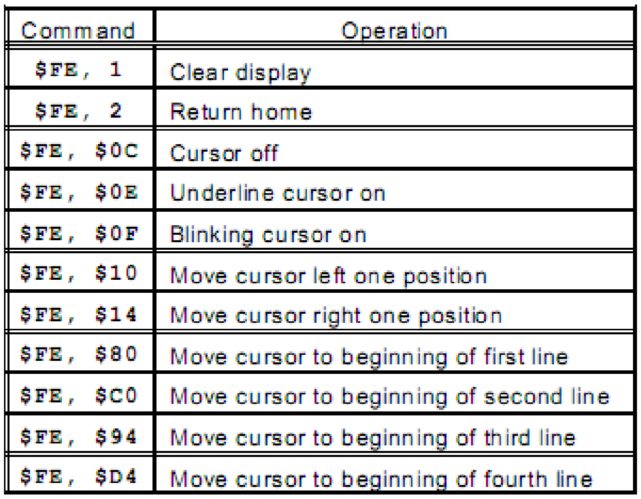
Modifier	Operation
{I}{S}BIN{1..16}	Send binary digits
{I}{S}DEC{1..5}	Send decimal digits
{I}{S}HEX{1..4}	Send hexadecimal digits
REP c\n	Send character c repeated n times
STR ArrayVar{\n}	Send string of n characters

Table 4.3: Specified modifier

A program should wait for at least half a second before sending the first command to an LCD. It can take quite a while for an LCD to start up. The LCD is initialized the first time any character or command is sent to it using LCDOUT. If it is powered down and then powered back up for some reason during operation, an internal flag can be reset to tell the program to reinitialize it the next time it uses LCDOUT:

FLAGS = 0

Commands are sent to the LCD by sending a $FE followed by the command. Some useful commands are listed in the following table:



Command	Operation
$FE, 1	Clear display
$FE, 2	Return home
$FE, $0C	Cursor off
$FE, $0E	Underline cursor on
$FE, $0F	Blinking cursor on
$FE, $10	Move cursor left one position
$FE, $14	Move cursor right one position
$FE, $80	Move cursor to beginning of first line
$FE, $C0	Move cursor to beginning of second line
$FE, $94	Move cursor to beginning of third line
$FE, $D4	Move cursor to beginning of fourth line

Table 4.4: LCD command

There are commands to move the cursor to the beginning of the different lines of a multi-line display. For most LCDs, the displayed characters and lines are not consecutive in display memory - there can be a break in between locations. For most 16x2 displays, the first line starts at $80 and the second line starts at $C0. The command:

LCDOUT $FE, $80 + *4*

Sets the display to start writing characters at the forth position of the first line. 16x1 displays are usually formatted as 8x2 displays with a break between the memory locations for the first and second 8 characters. 4-line displays also have a mixed up memory map, as shown in the table above.

```
LCDOUT $FE,1,“Hello”            ‘ Clear display and show
                                  “Hello”’
LCDOUT $FE,$C0,“World”          ‘ Jump to second line and show
                                  “World”’
LCDOUT B0,#B1                   ‘ Display B0 and decimal ASCII
                                  value of B1’
```

The LCD may be connected to the PIC using either a 4-bit bus or an 8-bit bus. If an 8-bit bus is used, all 8 bits must be on one port. If a 4-bit bus is used, the top 4 LCD data bits must be connected to either the bottom 4 or top 4 bits of one port. Enable and Register Select may be connected to any port pin. R/W may be tied to ground if the LCDIN command is not used.

PBP assumes the LCD is connected to specific pins unless told otherwise using DEFINEs. It assumes the LCD will be used with a 4-bit bus with data lines DB4 - DB7 connected to PIC PORTD.0 - PORTD.3, Register Select to PORTE.1and Enable to PORTE.2. It is also preset to initialize the LCD to a 2 line display.

```
        ‘Set LCD Data port’
DEFINE LCD_DREG PORTD
‘Set starting Data bit (0 or 4) if 4-bit bus’
DEFINE LCD_DBIT 4
‘Set LCD Register Select port’
DEFINE LCD_RSREG PORTE
‘Set LCD Register Select bit’
DEFINE LCD_RSBIT 1
‘Set LCD Enable port’
DEFINE LCD_EREG PORTE
‘Set LCD Enable bit’
DEFINE LCD_EBIT 2
‘Set LCD bus size (4 or 8 bits)’
DEFINE LCD_BITS 4
‘Set number of lines on LCD’
```

```
DEFINE LCD_LINES 2
‘Set command delay time in us’
DEFINE LCD_COMMANDUS 2000
‘ Set data delay time in us’
DEFINE LCD_DATAUS 50
```

4.7.5 Programming the Stepper Motor

The programming for the stepper motor is nearly the same as free running program. We have to define the register to set the output or input of the system. For example, we want to run the stepper motor clock-wise continuously. The stepper motor program has 4 output, the register is define as below:

```
Define osc 4
trisb.0 = 0
trisb.1 = 0
trisb.2 = 0
trisb.3 = 0
```

There are 4 port that has been define for output because the 4 output from the PIC will be connected to the stepper motor driver and will be sending the sequential data to the stepper motor. The stepper motor will run according to the sequential data. There are three main stepping modes that are single stepping, high torque and half stepping. We are using the high stepping mode for this project. For each 4 of the sequential data is sent, the stepper motor will rotate one step. The stepper motor can rotate either clock-wise or anti clock-wise. The clock-wise program for stepper motor is as below:

```
main:
portb = %00001001
pause 100
portb = %00001100
```

```
pause 100
portb = %00000110
pause 100
portb = %00000011
pause 100
goto main
```

The data order of the sequential data will determine the direction of the stepper motor rotation. When the sequential data is sent in inverse order of the clock-wise, the rotation of the stepper motor will be anti clock-wise. The program is as below:

```
main:
portb = %00000011
pause 100
portb = %00000110
pause 100
portb = %00001100
pause 100
portb =%00001001
pause 100
goto main
```

4.7.6 Programming the DC motor

Programming the DC motor is also nearly the same as the free running program. We have to define the register to set the output or input of the system. For example, we want to run the DC motor clock-wise continuously. . The DC motor program has 3 output, the register is define as below:

```
Define osc 4
trisc.0 = 0
trisc.1 = 0
trisc.2 = 0
```

The DC motor has 3 output because the DC driver has 3 input to drive the motor. PORTC 1 and PORTB 2 are use to control the rotation of the motor and PORTC 0 is to enable the motor to run. The motor can rotate clock-wise or anti clock-wise according to the input that are sent from the PIC. For the clock-wise rotation, the program is as below:

```
main:
LOW portc.0
HIGH portc.1
LOW portc.2
goto main
```

The motor will continuously rotate clock-wise until the system is shut down or the reset button is push. The motor will rotate anti clock-wise if the data for PORTC 1 and PORTC 2 are sent inversely from the clock-wise data. The motor will run anti clock-wise for the program below:

```
main:
LOW portc.0
LOW portc.1
HIGH portc.2
goto main
```

PORTC 0 is the enabling port that allow the driver to drive the motor. If the enabling port is reset to HIGH the motor will stop.

```
HIGH portc.0
```

4.8 Application

The remote control and the car model are successfully built. Figure 4.12 and Figure 4.13 shows the complete model of the remote control and the complete model of the car. 4 LEDs (Figure 4.14) are added at the front of the car and 3 LEDs (Figure 4.15) at the bottom of left and right side of the car. The 2 LEDs at the front will automatically ON when the surrounding became a little bit dark and the LEDs at the bottom of the car will automatically ON when the surrounding is completely dark. The other 2 LEDs at the front will ON when received the signal from the remote control.

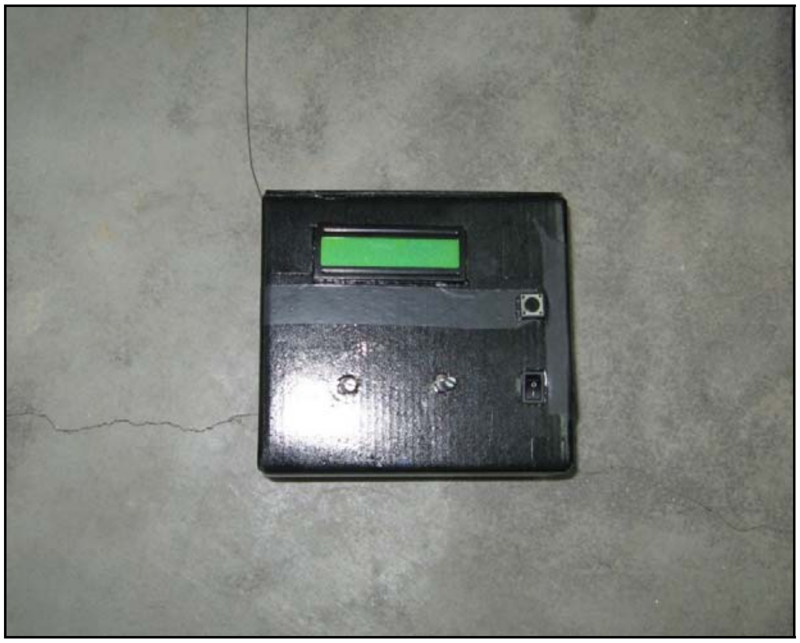


Figure 4.12: Complete remote control

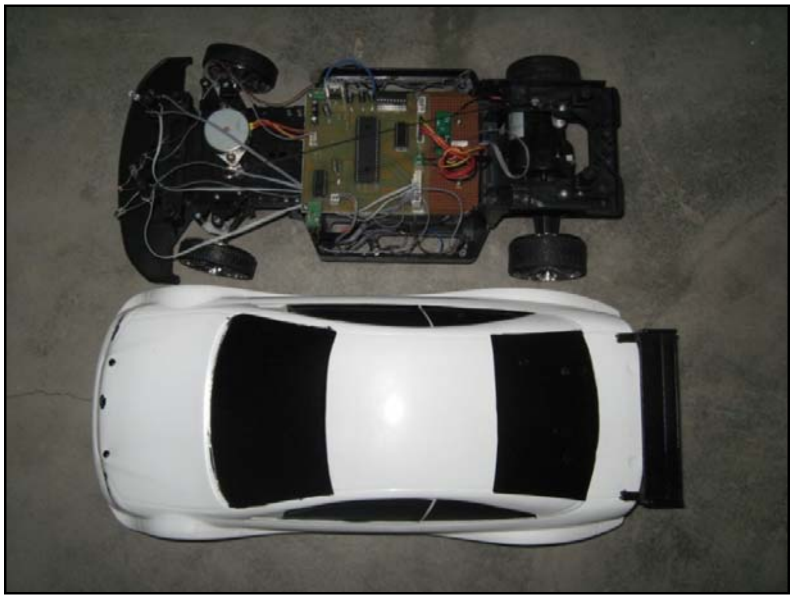


Figure 4.13: Complete car model

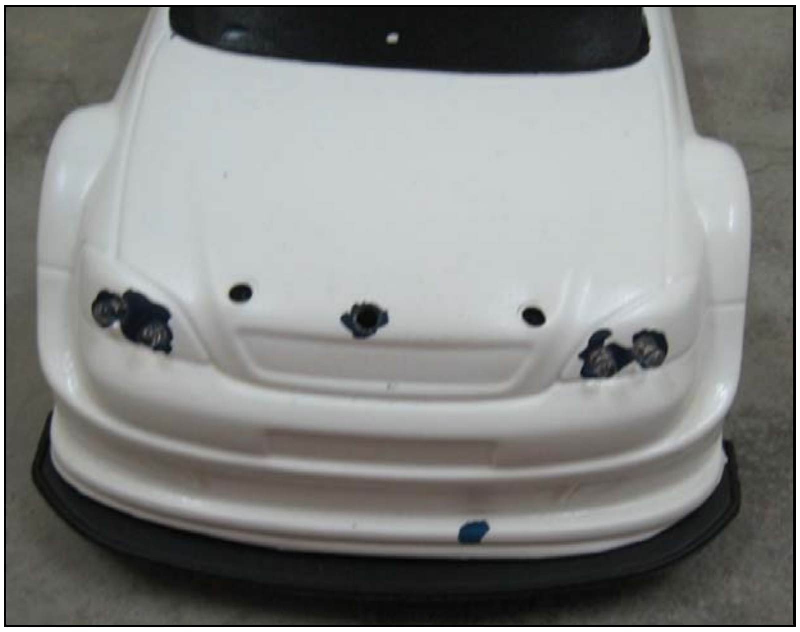


Figure 4.14: LEDs placed at the front of the car

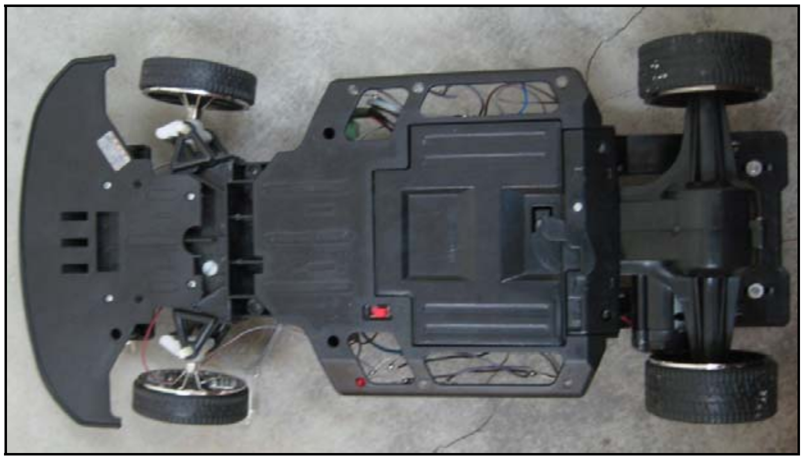


Figure 4.15: LEDs placed at the bottom of the car

CHAPTER 5

CONCLUSION AND FUTURE DEVELOPMENT

5.1 Overview

This thesis has discussed on remote control of a moving vehicle. This project is design to build a device that can control a vehicle movement and maneuver within a range of 7 meter. This project is divided into 3 scopes that are the vehicle can be control in a range of 7 meter, the vehicle can move forward and reverse, and the degree of the vehicle maneuver can be control precisely.

This project has been able to achieve its objectives. The remote control of a moving vehicle is successfully built. This chapter will discuss about the problem faced during the development of this project and future recommendation that can be made to the project.

5.2 Problem And Solution

There are some problems occur during the development of the project:

(i) Implementing stepper motor into the model:

- The original motor of the recycle toy car is a DC motor.
- The maneuver compartment has gears.

Solution:
The maneuver compartment is modified using recycle component and the stepper motor is modified to suit the gears. (Refer chapter 3)

(ii) Programming the transmitter and receiver to transmit and receive data

- The transmitter needed a serial data as an input and the receiver output is also serial.
- The PICBasic Pro has a little bit complex command for sending and receiving serial data.

Solution:
The PIC is connected to the encoder to send serial data to the transmitter and the PIC is connected the decoder when receiving data from the receiver. There are two type of encoder and decoder that are suitable that is HT-12E and PT 2262, and HT-12D and PT 2272. HT-12E and HT-12D is chosen because it only uses 16 inputs for encoder and 16 outputs for decoder. The PT 2262 and PT 2272 is has more inputs and outputs. This project just used 12 inputs and 12 outputs. (Refer chapter 3 and chapter 4).

(iii) Implementing PCB

- The route on the PCB is overlap with one another.
- There are some libraries for the DXP Protel software is not available.

Solution:
The track size for the PCB route is narrows a little bit to ensure the route will not overlap again. The libraries are made such as libraries for encoder and decoder.

5.3 Conclusion

The remote control of a moving vehicle has been presented in this project. This project has achieved all the objectives and scopes. The results of the output are the same as stated in reference studied. The remote control of a moving vehicle can be used as a surveillance car. It car help us scout the area that are unreachable to us.

5.4 Future Recommendation

Even though this project is successfully done, however there are some enhancements that can be applied to the project. This addition somehow can improve performance of the project. Below are some suggestions for the future development:

(i) Installed a motion camera.

- Using a motion camera, we can detect movement in dark places.

(ii) Use GUI to control the car.

- Replace the potentiometer as the joystick with a GUI interface such as visual basic.

(iii) Produce a smaller car

- Built a smaller car that can enter a small area without any difficulties.

(iv) Range of the transmitter and receiver.

- Control the car in wider range.
- Using a higher frequency transmitter and receiver
- Using a coil antenna.

5.5 Costing and Commercialization

This part explains about the costing of this project. The total project cost for all components is estimated to be RM 317.11. The highest cost among the component is the price of the transmitter and receiver. Even though the price of these components is expensive, it is still a necessary item and the less expensive substitutes are nonexistent. The component chosen based on the performance of the component, means that the chosen component rating is above designed value. The table of component cost is on Table 5.1 below.

No	**Device**	**Specification**	**Unit Cost**	**Quantity**	**Extended Cost**
1	Capacitor	4.7 µF	RM0.07	1	RM0.07
2	Capacitor	1 µF	RM0.07	2	RM0.14

3	Capacitor	100 μF	RM0.10	1	RM0.10
4	Capacitor	22 pF	RM0.07	2	RM0.14
5	Resistor	220 Ω	RM0.05	8	RM0.40
6	Resistor	560 Ω	RM0.05	8	RM0.40
7	Resistor	2.2 kΩ	RM0.06	1	RM0.06
8	Resistor	4.7 kΩ	RM0.07	8	RM0.56
9	Resistor	10 kΩ	RM0.06	1	RM0.06
10	Resistor	33 kΩ	RM0.06	1	RM0.06
11	Resistor	47 kΩ	RM0.06	1	RM0.06
12	Resistor	1 MΩ	RM0.04	8	RM0.32
13	Strip board	Independent	RM5.00	0	RM5.00
14	LCD	16 x 2	RM30.00	1	RM30.00
15	Potentiometer	10 kΩ	RM0.60	3	RM1.80
16	Ribbon Cable	16 x 2m	RM3.00	3	RM9.00
17	Voltage Regulator	LM 7805	RM1.00	3	RM3.00
18	Crystal	4 MHz	RM1.90	2	RM3.80
19	Transmitter	RF transmitter	RM50.00	1	RM50.00
20	Receiver	RF receiver	RM50.00	1	RM50.00
21	Encoder	HT-12E	RM4.50	1	RM4.50

22	Decoder	HT-12D	RM4.50	1	RM4.50
23	IC Base	16 kaki	RM0.20	2	RM0.40
24	IC Base	18 kaki	RM0.20	2	RM0.40
25	IC Base	48 kaki	RM0.20	2	RM0.40
26	PIC 16F877A	Microcontroller	RM25.00	2	RM50.00
27	Stepper Motor		RM35.00	1	RM35.00
28	Stepper Motor Driver	ULN 2003A	RM4.90	1	RM4.90
29	DC Motor		RM35.00	1	RM35.00
30	DC Motor Driver	L 293B	RM7.20	1	RM7.50
31	Connector	9 Vdc	RM0.02	2	RM0.04
32	LED	Crystal white	RM0.50	3	RM1.50
33	LED	Crystal Red	RM0.50	3	RM1.50
34	LED	Green	RM0.50	3	RM1.50
35	Wire Wrapping		RM15.00	1	RM15.00
				TOTAL	**RM317.11**

Table 5.1: Total Cost of the Project

This project can be commercialized after some improvement is being made. The car has to be attached with a camera to ease user to monitor the movement of the car. The car can be used as a surveillance car. The total cost of this project was estimated to be RM 350. Based on it, it is still relevant to commercialize this project.

REFERENCES

[1] Ruf Bot 1.1
http://mysite.verizon.net/res8dbeh/

[2] Toy Car Hack - "Synthetic Rodent Development
http://mondo-technology.com/toycar.html

[3] Toy Car Hack - "Synthetic Rodent Development
http://mondo-technology.com/toycar.html

[4] Final Project - RC Car Controller
http://instruct1.cit.cornell.edu/courses/ee476/FinalProjects/s1999/blair/RCcar.html

[5] Final Project - RC Car Controller
http://instruct1.cit.cornell.edu/courses/ee476/FinalProjects/s1999/blair/RCcar.html

APPENDIX A

PIC MICROCONTROLLER CIRCUIT

TRANSMITTER CIRCUIT

RECEIVER CIRCUIT

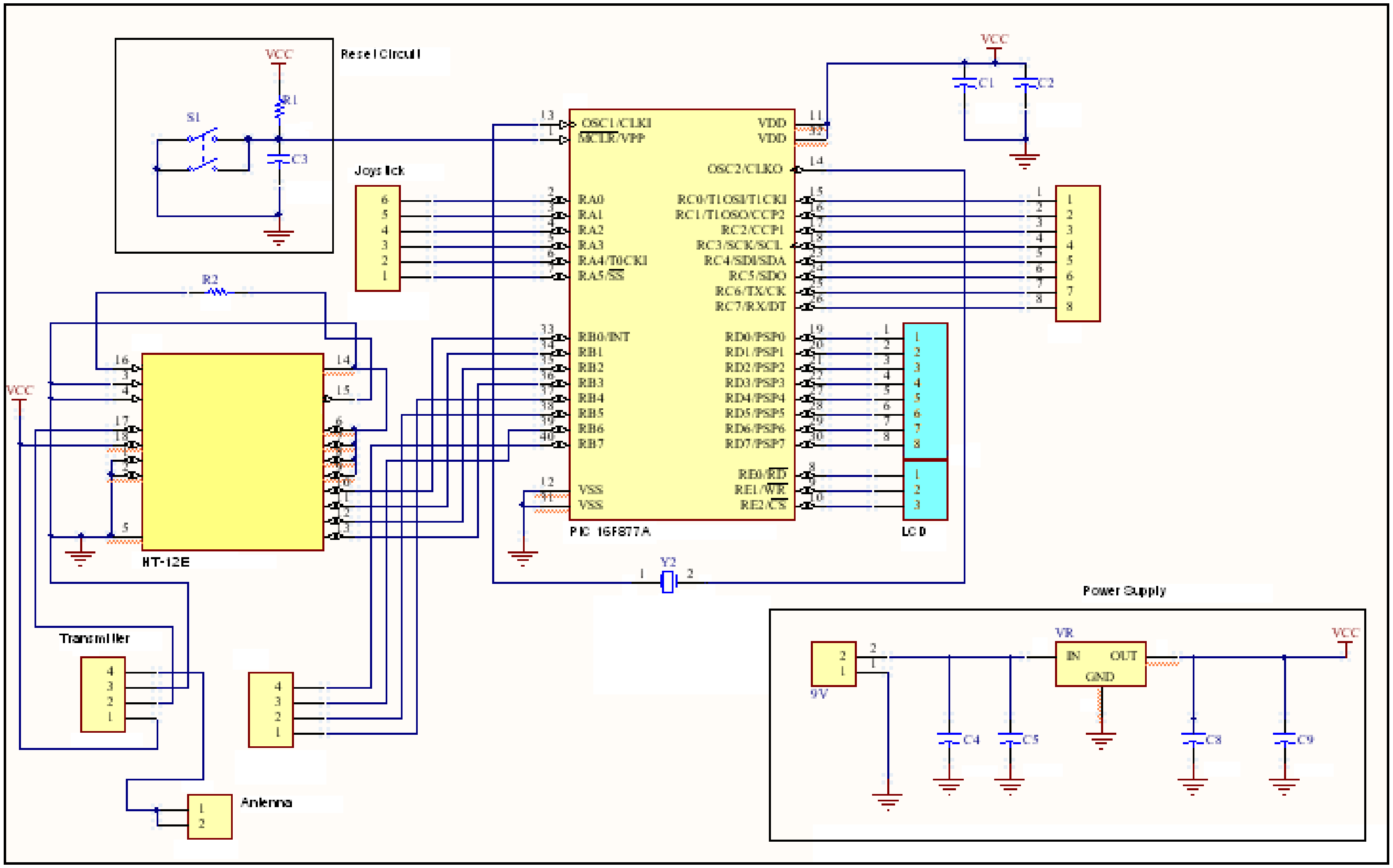


Figure A.1: Full Transmitter Schematic of the Project

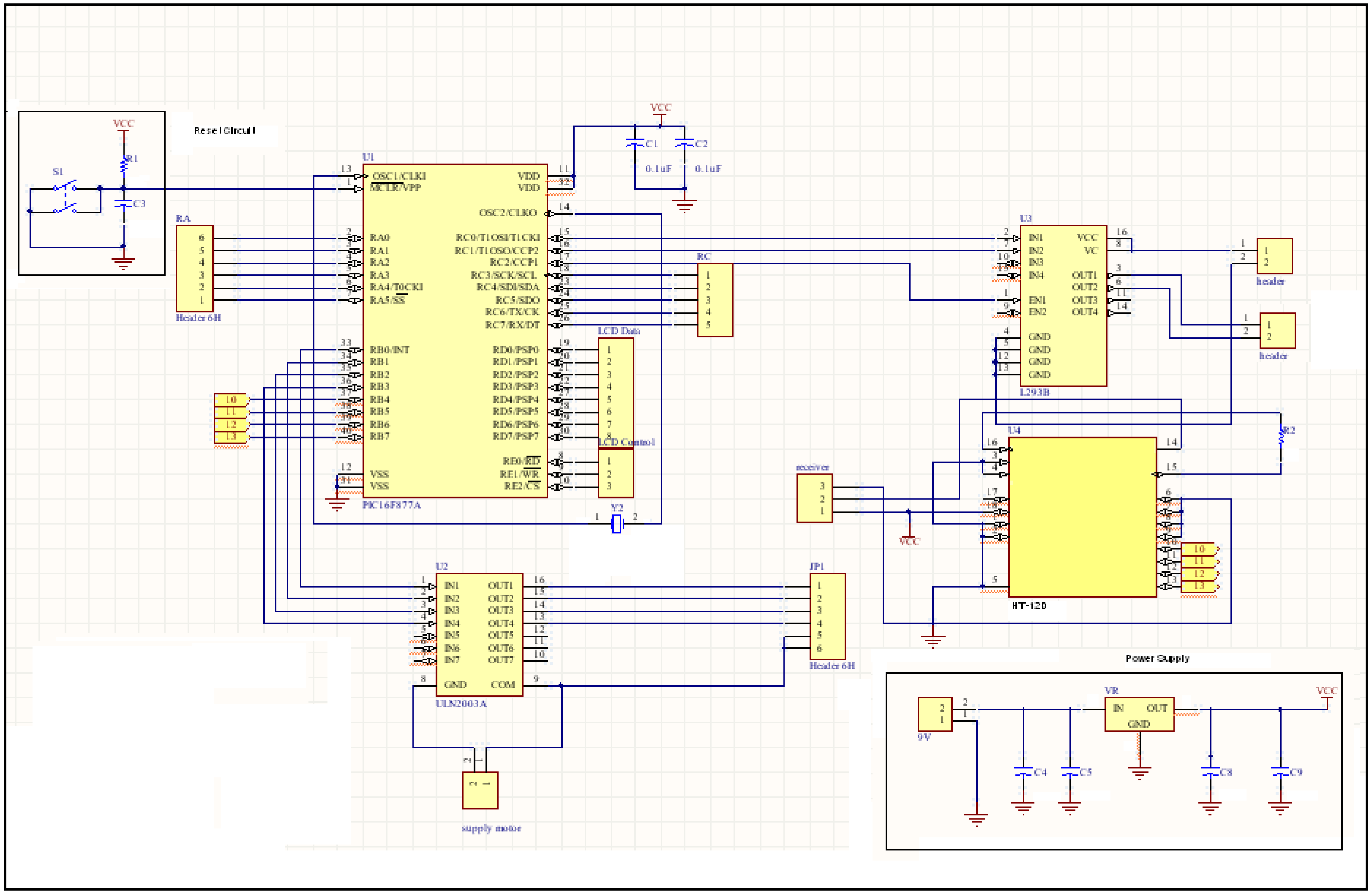


Figure A.2: Full Receiver Schematic of the project

APPENDIX B

LIST OF COMPONENT

Table B.1: List of Component 1

No	Component	Specification	Quantity
1	Capacitor	4.7 µF	1
2	Capacitor	1 µF	2
3	Capacitor	100 µF	1
4	Capacitor	22 pF	2
5	Resistor	220 Ω	8
6	Resistor	560 Ω	8
7	Resistor	2.2 kΩ	1
8	Resistor	4.7 kΩ	8
9	Resistor	10 kΩ	1
10	Resistor	33 kΩ	1
11	Resistor	47 kΩ	1
12	Resistor	1 MΩ	8
13	Strip board	Independent	0
14	LCD	16 x 2	1
15	Potentiometer	10 kΩ	3
16	Ribbon Cable	16 x 2m	3
17	Voltage Regulator	LM 7805	3
18	Crystal	4 MHz	2
19	Transmitter	RF transmitter	1
20	Receiver	RF receiver	1
21	Encoder	HT-12E	1
22	Decoder	HT-12D	1
23	IC Base	16 kaki	2
24	IC Base	18 kaki	2
25	IC Base	48 kaki	2
26	PIC 16F877A	Microcontroller	2
27	Stepper Motor		1
28	Stepper Motor Driver	ULN 2003A	1

29	DC Motor		1
30	DC Motor Driver	L 293B	1
31	Connector	9 Vdc	2
32	LED	Crystal white	3
33	LED	Crystal Red	3
34	LED	Green	3
35	Wire Wrapping		1

APPENDIX C

PICBASIC PROGRAMMING

TRANSMITTER PROGRAMMING

RECEIVER PROGRAMMING

TRANSMITTER PROGRAMMING

```
DEFINE osc 4
Define ADC_BITS 8                                         'bit adc'
Define ADC_CLOCK 3                                        'internal clock 3 MHz'
Define ADC_SAMPLEUS 50                                    'samplint time in microsec'

DEFINE LCD_DREG                 PORTD                     'define port for lcd'
DEFINE LCD_DBIT                 4
DEFINE LCD_RSREG                 PORTE
DEFINE LCD_RSBIT                1
DEFINE LCD_EREG                  PORTE
DEFINE LCD_EBIT                  2
DEFINE LCD_BITS                 4
DEFINE LCD_LINES                2
DEFINE LCD_COMMANDUS            2000
DEFINE LCD_DATAUS     50

adcon0 = %11101101                                         'adc channel register'
adcon0 = %11100101

adcon1 = %00001001                                        'adc port register'

trisd   = 0                                               'output port for lcd'
trise.1 = 0
trise.2 = 0

trise.0 = 1                                               'input for adc'
trisa.5 = 1

trisb.0 = 0                                               'output port for encoder'
trisb.1 = 0
trisb.2 = 0
trisb.3 = 0

trisb.5 = 1                                               'suis'

res1 var word                                             'adc variable'
res2 var word

main:
pause 100
    LCDOUT $FE,1
    lcdout $fe,$80+2, "angle:"
    lcdout $fe,$C0, "movement:"
pause 100
goto process
```

```
process:
adcin 4, res1
pause 100
if res1 < 28  then satu
if res1 < 56  then dua
if res1 < 84  then tiga
if res1 < 112 then empat
if res1 < 140 then lima
if res1 < 168 then enam
if res1 < 196 then tujuh
if res1 < 224 then lapan
if res1 < 256 then sembilan
goto process

satu:
pause 100
    LCDOUT $FE,1
    lcdout $fe,$80+2, "angle:-45"
    lcdout $fe,$C0, "movement:"
pause 100
if portb.5 = 1 then process
portb = %00000001
goto process

dua:
pause 100
    LCDOUT $FE,1
    lcdout $fe,$80+2, "angle:-15"
    lcdout $fe,$C0, "movement:"
pause 100
if portb.5 = 1 then process
portb = %00000010
goto process

tiga:
pause 100
    LCDOUT $FE,1
    lcdout $fe,$80+2, "angle:-5"
    lcdout $fe,$C0, "movement:"
pause 100
if portb.5 = 1 then process
portb = %00000011
goto process
```

```
empat:
pause 100
    LCDOUT $FE,1
    lcdout $fe,$80, "angle:LED OFF"
    lcdout $fe,$C0, "movement:"
pause 100
if portb.5 = 1 then process
portb = %00000100
goto process

lima:
pause 100
    LCDOUT $FE,1
    lcdout $fe,$80+2, "angle:0"
    lcdout $fe,$C0, "movement:"
pause 100
adcin 5, res2
pause 100
if res2 < 85 then maju
if res2 < 170 then senyap
if res2 < 255 then undur
if portb.6 = 1 then process
portb = %00000101
goto process

enam:
pause 100
    LCDOUT $FE,1
    lcdout $fe,$80, "angle:LED ON"
    lcdout $fe,$C0, "movement:"
pause 100
if portb.5 = 1 then process
portb = %00000110
goto process

tujuh:
pause 100
    LCDOUT $FE,1
    lcdout $fe,$80+2, "angle:5"
    lcdout $fe,$C0, "movement:"
pause 100
if portb.5 = 1 then process
portb = %00000111
goto process
```

```
lapan:
pause 100
    LCDOUT $FE,1
    lcdout $fe,$80+2, "angle:15"
    lcdout $fe,$C0, "movement:"
pause 100
if portb.5 = 1 then process
portb = %00001000
goto process

sembilan:
pause 100
    LCDOUT $FE,1
    lcdout $fe,$80+2, "angle:45"
    lcdout $fe,$C0, "movement:"
pause 100
if portb.5 = 1 then process
portb = %00001001
goto process

maju:
pause 100
    LCDOUT $FE,1
    lcdout $fe,$80+2, "angle:"
    lcdout $fe,$C0, "movement:maju"
pause 100
portb = %00001010
goto process

senyap:
pause 100
    LCDOUT $FE,1
    lcdout $fe,$80+2, "angle:"
    lcdout $fe,$C0, "movement:senyap"
pause 100
portb = %00001011
goto process

undur:
pause 100
    LCDOUT $FE,1
    lcdout $fe,$80+2, "angle:"
    lcdout $fe,$C0, "movement:undur"
pause 100
portb = %00001100
goto process
```

RECEIVER PROGRAMMING

```
DEFINE osc 4
Define ADC_BITS 8                          'bit adc'
Define ADC_CLOCK 3                         'internal clock 3 MHz'
Define ADC_SAMPLEUS 50                     'samplint time in microsec'

adcon0 = %11011101                         'adc channel register'
adcon1 = 0                                 'adc port register'

trisa.3 = 1                                ‘adc input’

trisd.0 = 0                                'output port for led'
trisd.1 = 0
trisd.2 = 0
trisd.3 = 0
trisd.4 = 0
trisd.5 = 0

trisb.0 = 0                                 'output for stepper'
trisb.1 = 0
trisb.2 = 0
trisb.3 = 0

trisb.4 = 1                                'define output for decoder'
trisb.5 = 1
trisb.6 = 1
trisb.7 = 1

trisc.0 = 0                                 'output for dc motor'
trisc.1 = 0
trisc.2 = 0

loop var word                              'stepper motor variable'
loops var word

res var word                               'adc variable'

main:
loops = 0
adcin 3, res
pause 100
        if res < 5 then two
        if res < 10 then one
pause 100
        high portd.0
        low portd.1
```

```
          low portd.2
          high portd.3
pause 100
goto process

process:
portb = %00000000
if portb = %00010000 then satu
if portb = %00100000 then dua
if portb = %00110000 then tiga
if portb = %01000000 then empat
if portb = %01010000 then lima
if portb = %01100000 then enam
if portb = %01110000 then tujuh
if portb = %10000000 then lapan
if portb = %10010000 then sembilan
if portb = %10100000 then sepuluh
if portb = %10110000 then sebelas
if portb = %11000000 then duabelas
goto main

satu:                                   'loop 1 step clockwise (30degree)'
loop = 15
loops = 0
goto motor1                             'motor1 = clockwise'

dua:
loop = 10
loops = 0
goto motor1

tiga:
loop = 5
loops = 0
goto motor1

empat:
high portd.4
low portd.5
goto process

lima:
loop = 0
loops = 0
goto motor1
```

```
enam:
low portd.4
high portd.5
goto process

tujuh:
loop = 5
loops = 0
goto motor2

lapan:
loop = 10
loops = 0
goto motor2

sembilan:
loop = 15
loops = 0
goto motor2

motor1:                                 'rotate 1 step clockwise'

portb = %00001001
pause 100
portb = %00001100
pause 100
portb = %00000110
pause 100
portb = %00000011
pause 100

if loops = loop then process
loops = LOOPS + 1                       'add 1 at variable loops'
pause 100
goto motor1                             'loop back until loop = loops'

motor2:                                 'rotate 1 step counterclockwise'

portb = %00000011
pause 100
portb = %00000110
pause 100
portb = %00001100
pause 100
portb = %00001001
pause 100
```

```
if loops = loop then process
loops = loops + 1
pause 100
goto motor2

two:
pause 100
high portd.1
low portd.0
high portd.2
low portd.3
pause 100
goto process

one:
pause 100
high portd.1
low portd.0
low portd.2
high portd.3
pause 100
goto process

sepuluh:
high portc.1
low portc.0
high portc.2
goto main

sebelas:
high portc.1
low portc.0
low portc.2
goto main

duabelas:
high portc.0
low portc.1
high portc.2
goto main
```

APPENDIX D

DATA SHEET

PIC 16F877A

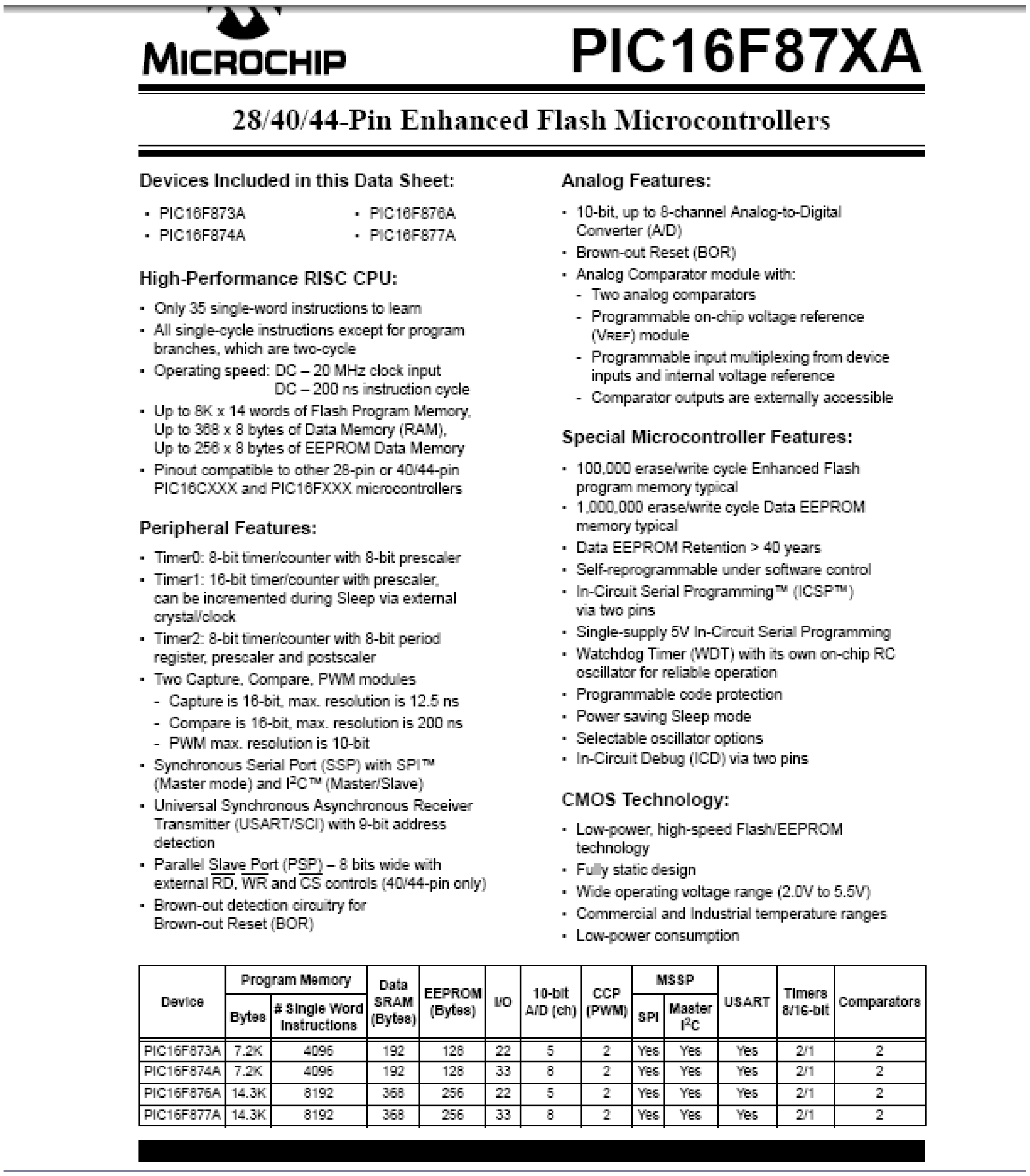
ULN 2003A

L 293B

HT-12E

HT12D

PIC 16F877A



MICROCHIP

PIC16F87XA

28/40/44-Pin Enhanced Flash Microcontrollers

Devices Included in this Data Sheet:

- PIC16F873A
- PIC16F874A
- PIC16F876A
- PIC16F877A

High-Performance RISC CPU:

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

Peripheral Features:

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

Analog Features:

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

Special Microcontroller Features:

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

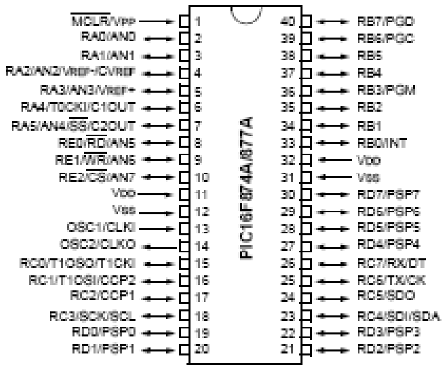
CMOS Technology:

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

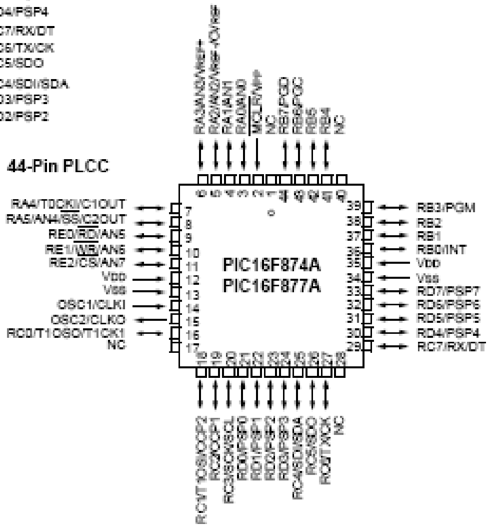
Device	Program Memory		Data SRAM (Bytes)	EEPROM (Bytes)	I/O	10-bit A/D (ch)	CCP (PWM)	MSSP		USART	Timers 8/16-bit	Comparators
	Bytes	# Single Word Instructions						SPI	Master I^{2}C			
PIC16F873A	7.2K	4096	192	128	22	5	2	Yes	Yes	Yes	2/1	2
PIC16F874A	7.2K	4096	192	128	33	8	2	Yes	Yes	Yes	2/1	2
PIC16F876A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

Pin Diagrams (Continued)

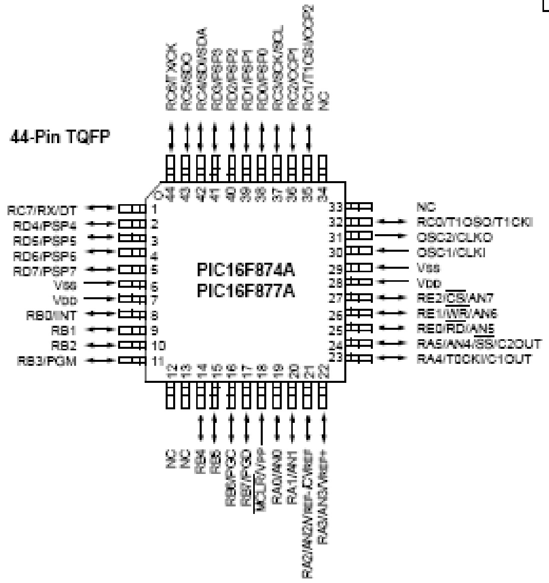
40-Pin PDIP



44-Pin PLCC



44-Pin TQFP



1.0 DEVICE OVERVIEW

This document contains device specific information about the following devices:

- PIC16F873A
- PIC16F874A
- PIC16F876A
- PIC16F877A

PIC16F873A/876A devices are available only in 28-pin packages, while PIC16F874A/877A devices are available in 40-pin and 44-pin packages. All devices in the PIC16F87XA family share common architecture with the following differences:

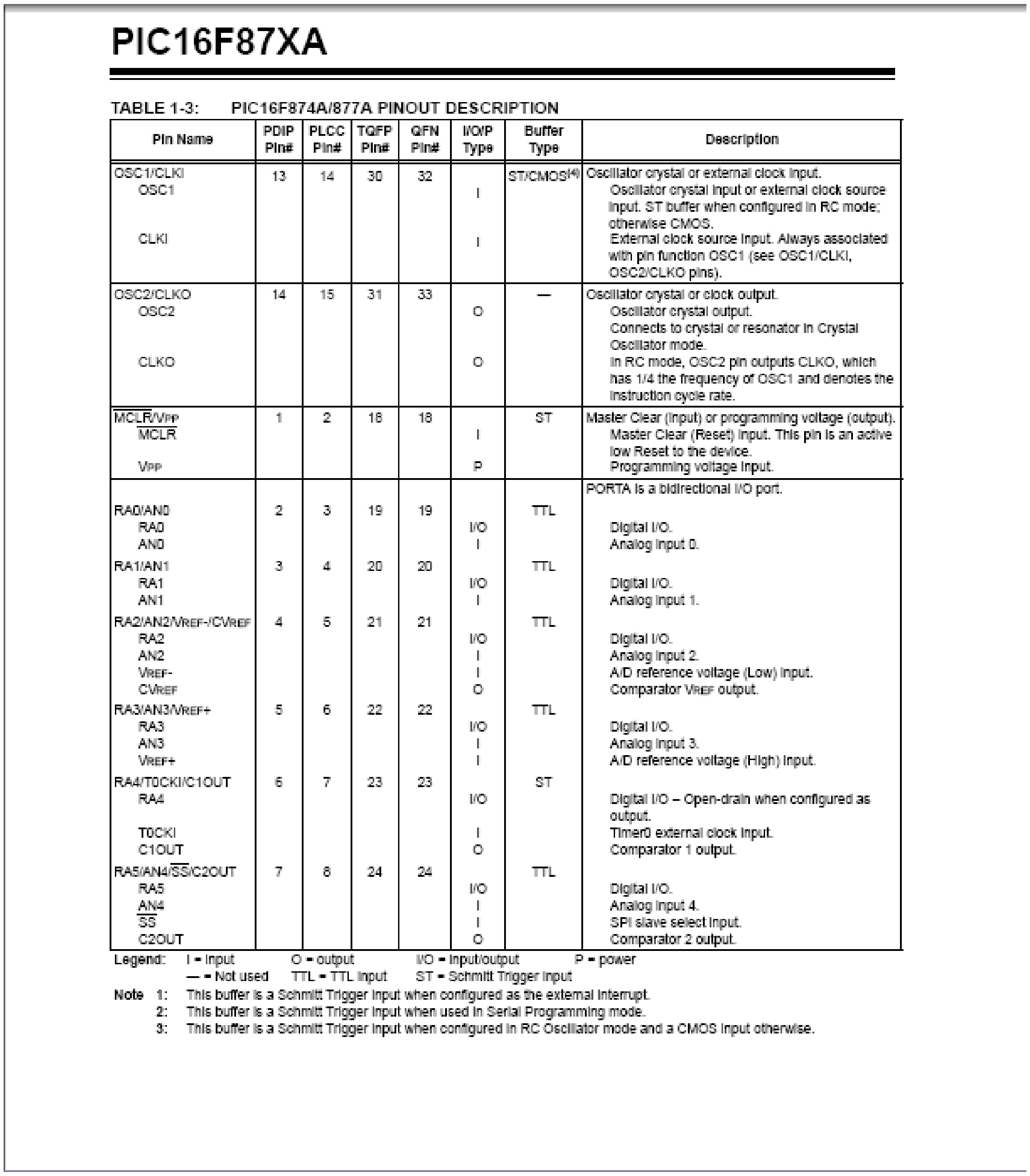
- The PIC16F873A and PIC16F874A have one-half of the total on-chip memory of the PIC16F876A and PIC16F877A
- The 28-pin devices have three I/O ports, while the 40/44-pin devices have five
- The 28-pin devices have fourteen interrupts, while the 40/44-pin devices have fifteen
- The 28-pin devices have five A/D input channels, while the 40/44-pin devices have eight
- The Parallel Slave Port is implemented only on the 40/44-pin devices

The available features are summarized in Table 1-1. Block diagrams of the PIC16F873A/876A and PIC16F874A/877A devices are provided in Figure 1-1 and Figure 1-2, respectively. The pinouts for these device families are listed in Table 1-2 and Table 1-3.

Additional information may be found in the PICmicro® Mid-Range Reference Manual (DS33023), which may be obtained from your local Microchip Sales Representative or downloaded from the Microchip web site. The Reference Manual should be considered a complementary document to this data sheet and is highly recommended reading for a better understanding of the device architecture and operation of the peripheral modules.

TABLE 1-1: PIC16F87XA DEVICE FEATURES

Key Features	PIC16F873A	PIC16F874A	PIC16F876A	PIC16F877A
Operating Frequency	DC – 20 MHz	DC – 20 MHz	DC – 20 MHz	DC – 20 MHz
Resets (and Delays)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)
Flash Program Memory (14-bit words)	4K	4K	8K	8K
Data Memory (bytes)	192	192	368	368
EEPROM Data Memory (bytes)	128	128	256	256
Interrupts	14	15	14	15
I/O Ports	Ports A, B, C	Ports A, B, C, D, E	Ports A, B, C	Ports A, B, C, D, E
Timers	3	3	3	3
Capture/Compare/PWM modules	2	2	2	2
Serial Communications	MSSP, USART	MSSP, USART	MSSP, USART	MSSP, USART
Parallel Communications	—	PSP	—	PSP
10-bit Analog-to-Digital Module	5 input channels	8 input channels	5 input channels	8 input channels
Analog Comparators	2	2	2	2
Instruction Set	35 Instructions	35 Instructions	35 Instructions	35 Instructions
Packages	28-pin PDIP 28-pin SOIC 28-pin SSOP 28-pin QFN	40-pin PDIP 44-pin PLCC 44-pin TQFP 44-pin QFN	28-pin PDIP 28-pin SOIC 28-pin SSOP 28-pin QFN	40-pin PDIP 44-pin PLCC 44-pin TQFP 44-pin QFN



PIC16F87XA

TABLE 1-3: PIC16F874A/877A PINOUT DESCRIPTION

Pin Name	PDIP Pin#	PLCC Pin#	TQFP Pin#	QFN Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKI	13	14	30	32		ST/CMOS[4]	Oscillator crystal or external clock input.
OSC1					I		Oscillator crystal input or external clock source input. ST buffer when configured in RC mode; otherwise CMOS.
CLKI					I		External clock source input. Always associated with pin function OSC1 (see OSC1/CLKI, OSC2/CLKO pins).
OSC2/CLKO	14	15	31	33		—	Oscillator crystal or clock output.
OSC2					O		Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
CLKO					O		In RC mode, OSC2 pin outputs CLKO, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
$\overline{\text{MCLR}}$/VPP	1	2	18	18		ST	Master Clear (input) or programming voltage (output).
$\overline{\text{MCLR}}$					I		Master Clear (Reset) input. This pin is an active low Reset to the device.
VPP					P		Programming voltage input.
							PORTA is a bidirectional I/O port.
RA0/AN0	2	3	19	19		TTL	
RA0					I/O		Digital I/O.
AN0					I		Analog input 0.
RA1/AN1	3	4	20	20		TTL	
RA1					I/O		Digital I/O.
AN1					I		Analog input 1.
RA2/AN2/VREF-/CVREF	4	5	21	21		TTL	
RA2					I/O		Digital I/O.
AN2					I		Analog input 2.
VREF-					I		A/D reference voltage (Low) input.
CVREF					O		Comparator VREF output.
RA3/AN3/VREF+	5	6	22	22		TTL	
RA3					I/O		Digital I/O.
AN3					I		Analog input 3.
VREF+					I		A/D reference voltage (High) input.
RA4/T0CKI/C1OUT	6	7	23	23		ST	
RA4					I/O		Digital I/O – Open-drain when configured as output.
T0CKI					I		Timer0 external clock input.
C1OUT					O		Comparator 1 output.
RA5/AN4/$\overline{\text{SS}}$/C2OUT	7	8	24	24		TTL	
RA5					I/O		Digital I/O.
AN4					I		Analog input 4.
$\overline{\text{SS}}$					I		SPI slave select input.
C2OUT					O		Comparator 2 output.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured in RC Oscillator mode and a CMOS input otherwise.

TABLE 1-3: PIC16F874A/877A PINOUT DESCRIPTION (CONTINUED)

Pin Name	PDIP Pin#	PLCC Pin#	TQFP Pin#	QFN Pin#	I/O/P Type	Buffer Type	Description
							PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	33	36	8	9		TTL/ST(1)	
RB0					I/O		Digital I/O.
INT					I		External interrupt.
RB1	34	37	9	10	I/O	TTL	Digital I/O.
RB2	35	38	10	11	I/O	TTL	Digital I/O.
RB3/PGM	36	39	11	12		TTL	
RB3					I/O		Digital I/O.
PGM					I		Low-voltage ICSP programming enable pin.
RB4	37	41	14	14	I/O	TTL	Digital I/O.
RB5	38	42	15	15	I/O	TTL	Digital I/O.
RB6/PGC	39	43	16	16		TTL/ST(2)	
RB6					I/O		Digital I/O.
PGC					I		In-circuit debugger and ICSP programming clock.
RB7/PGD	40	44	17	17		TTL/ST(2)	
RB7					I/O		Digital I/O.
PGD					I/O		In-circuit debugger and ICSP programming data.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured in RC Oscillator mode and a CMOS input otherwise.

TABLE 1-3: PIC16F874A/877A PINOUT DESCRIPTION (CONTINUED)

Pin Name	PDIP Pin#	PLCC Pin#	TQFP Pin#	QFN Pin#	I/O/P Type	Buffer Type	Description
							PORTC is a bidirectional I/O port.
RC0/T1OSO/T1CKI	15	16	32	34		ST	
RC0					I/O		Digital I/O.
T1OSO					O		Timer1 oscillator output.
T1CKI					I		Timer1 external clock input.
RC1/T1OSI/CCP2	16	18	35	35		ST	
RC1					I/O		Digital I/O.
T1OSI					I		Timer1 oscillator input.
CCP2					I/O		Capture2 input, Compare2 output, PWM2 output.
RC2/CCP1	17	19	36	36		ST	
RC2					I/O		Digital I/O.
CCP1					I/O		Capture1 input, Compare1 output, PWM1 output.
RC3/SCK/SCL	18	20	37	37		ST	
RC3					I/O		Digital I/O.
SCK					I/O		Synchronous serial clock input/output for SPI mode.
SCL					I/O		Synchronous serial clock input/output for I^{2}C mode.
RC4/SDI/SDA	23	25	42	42		ST	
RC4					I/O		Digital I/O.
SDI					I		SPI data in.
SDA					I/O		I^{2}C data I/O.
RC5/SDO	24	26	43	43		ST	
RC5					I/O		Digital I/O.
SDO					O		SPI data out.
RC6/TX/CK	25	27	44	44		ST	
RC6					I/O		Digital I/O.
TX					O		USART asynchronous transmit.
CK					I/O		USART1 synchronous clock.
RC7/RX/DT	26	29	1	1		ST	
RC7					I/O		Digital I/O.
RX					I		USART asynchronous receive.
DT					I/O		USART synchronous data.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured in RC Oscillator mode and a CMOS input otherwise.

TABLE 1-3: PIC16F874A/877A PINOUT DESCRIPTION (CONTINUED)

Pin Name	PDIP Pin#	PLCC Pin#	TQFP Pin#	QFN Pin#	I/O/P Type	Buffer Type	Description
							PORTD is a bidirectional I/O port or Parallel Slave Port when interfacing to a microprocessor bus.
RD0/PSP0	19	21	38	38		ST/TTL(3)	
RD0					I/O		Digital I/O.
PSP0					I/O		Parallel Slave Port data.
RD1/PSP1	20	22	39	39		ST/TTL(3)	
RD1					I/O		Digital I/O.
PSP1					I/O		Parallel Slave Port data.
RD2/PSP2	21	23	40	40		ST/TTL(3)	
RD2					I/O		Digital I/O.
PSP2					I/O		Parallel Slave Port data.
RD3/PSP3	22	24	41	41		ST/TTL(3)	
RD3					I/O		Digital I/O.
PSP3					I/O		Parallel Slave Port data.
RD4/PSP4	27	30	2	2		ST/TTL(3)	
RD4					I/O		Digital I/O.
PSP4					I/O		Parallel Slave Port data.
RD5/PSP5	28	31	3	3		ST/TTL(3)	
RD5					I/O		Digital I/O.
PSP5					I/O		Parallel Slave Port data.
RD6/PSP6	29	32	4	4		ST/TTL(3)	
RD6					I/O		Digital I/O.
PSP6					I/O		Parallel Slave Port data.
RD7/PSP7	30	33	5	5		ST/TTL(3)	
RD7					I/O		Digital I/O.
PSP7					I/O		Parallel Slave Port data.
							PORTE is a bidirectional I/O port.
RE0/$\overline{\text{RD}}$/AN5	8	9	25	25		ST/TTL(3)	
RE0					I/O		Digital I/O.
$\overline{\text{RD}}$					I		Read control for Parallel Slave Port.
AN5					I		Analog input 5.
RE1/$\overline{\text{WR}}$/AN6	9	10	26	26		ST/TTL(3)	
RE1					I/O		Digital I/O.
$\overline{\text{WR}}$					I		Write control for Parallel Slave Port.
AN6					I		Analog input 6.
RE2/$\overline{\text{CS}}$/AN7	10	11	27	27		ST/TTL(3)	
RE2					I/O		Digital I/O.
$\overline{\text{CS}}$					I		Chip select control for Parallel Slave Port.
AN7					I		Analog input 7.
VSS	12, 31	13, 34	6, 29	6, 30, 31	P	—	Ground reference for logic and I/O pins.
VDD	11, 32	12, 35	7, 28	7, 8, 28, 29	P	—	Positive supply for logic and I/O pins.
NC	—	1, 17, 28, 40	12,13, 33, 34	13	—	—	These pins are not internally connected. These pins should be left unconnected.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured in RC Oscillator mode and a CMOS input otherwise.

2.2.2.2 OPTION_REG Register

The OPTION_REG Register is a readable and writable register, which contains various control bits to configure the TMR0 prescaler/WDT postscaler (single assignable register known also as the prescaler), the external INT interrupt, TMR0 and the weak pull-ups on PORTB.

Note:	To achieve a 1:1 prescaler assignment for the TMR0 register, assign the prescaler to the Watchdog Timer.

REGISTER 2-2: OPTION_REG REGISTER (ADDRESS 81h, 181h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 7 **$\overline{\text{RBPU}}$**: PORTB Pull-up Enable bit
1 = PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6 **INTEDG**: Interrupt Edge Select bit
1 = Interrupt on rising edge of RB0/INT pin
0 = Interrupt on falling edge of RB0/INT pin

bit 5 **T0CS**: TMR0 Clock Source Select bit
1 = Transition on RA4/T0CKI pin
0 = Internal instruction cycle clock (CLKO)

bit 4 **T0SE**: TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on RA4/T0CKI pin
0 = Increment on low-to-high transition on RA4/T0CKI pin

bit 3 **PSA**: Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2-0 **PS2:PS0**: Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Note:	When using Low-Voltage ICSP Programming (LVP) and the pull-ups on PORTB are enabled, bit 3 in the TRISB register must be cleared to disable the pull-up on RB3 and ensure the proper operation of the device

2.2.2.3 INTCON Register

The INTCON register is a readable and writable register, which contains various enable and flag bits for the TMR0 register overflow, RB port change and external RB0/INT pin interrupts.

Note:	Interrupt flag bits are set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>). User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

REGISTER 2-3: INTCON REGISTER (ADDRESS 0Bh, 8Bh, 10Bh, 18Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF
bit 7							bit 0

bit 7 **GIE**: Global Interrupt Enable bit
1 = Enables all unmasked interrupts
0 = Disables all interrupts

bit 6 **PEIE**: Peripheral Interrupt Enable bit
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts

bit 5 **TMR0IE**: TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4 **INTE**: RB0/INT External Interrupt Enable bit
1 = Enables the RB0/INT external interrupt
0 = Disables the RB0/INT external interrupt

bit 3 **RBIE**: RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **TMR0IF**: TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INTF**: RB0/INT External Interrupt Flag bit
1 = The RB0/INT external interrupt occurred (must be cleared in software)
0 = The RB0/INT external interrupt did not occur

bit 0 **RBIF**: RB Port Change Interrupt Flag bit
1 = At least one of the RB7:RB4 pins changed state; a mismatch condition will continue to set the bit. Reading PORTB will end the mismatch condition and allow the bit to be cleared (must be cleared in software).
0 = None of the RB7:RB4 pins have changed state

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

2.2.2.4 PIE1 Register

The PIE1 register contains the individual enable bits for the peripheral interrupts.

Note:	Bit PEIE (INTCON<6>) must be set to enable any peripheral interrupt.

REGISTER 2-4: PIE1 REGISTER (ADDRESS 8Ch)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIE[1]	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

bit 7 **PSPIE**: Parallel Slave Port Read/Write Interrupt Enable bit[1]
1 = Enables the PSP read/write interrupt
0 = Disables the PSP read/write interrupt

Note 1: PSPIE is reserved on PIC16F873A/876A devices; always maintain this bit clear.

bit 6 **ADIE**: A/D Converter Interrupt Enable bit
1 = Enables the A/D converter interrupt
0 = Disables the A/D converter interrupt

bit 5 **RCIE**: USART Receive Interrupt Enable bit
1 = Enables the USART receive interrupt
0 = Disables the USART receive interrupt

bit 4 **TXIE**: USART Transmit Interrupt Enable bit
1 = Enables the USART transmit interrupt
0 = Disables the USART transmit interrupt

bit 3 **SSPIE**: Synchronous Serial Port Interrupt Enable bit
1 = Enables the SSP interrupt
0 = Disables the SSP interrupt

bit 2 **CCP1IE**: CCP1 Interrupt Enable bit
1 = Enables the CCP1 interrupt
0 = Disables the CCP1 interrupt

bit 1 **TMR2IE**: TMR2 to PR2 Match Interrupt Enable bit
1 = Enables the TMR2 to PR2 match interrupt
0 = Disables the TMR2 to PR2 match interrupt

bit 0 **TMR1IE**: TMR1 Overflow Interrupt Enable bit
1 = Enables the TMR1 overflow interrupt
0 = Disables the TMR1 overflow interrupt

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

8.0 CAPTURE/COMPARE/PWM MODULES

Each Capture/Compare/PWM (CCP) module contains a 16-bit register which can operate as a:

- 16-bit Capture register
- 16-bit Compare register
- PWM Master/Slave Duty Cycle register

Both the CCP1 and CCP2 modules are identical in operation, with the exception being the operation of the special event trigger. Table 8-1 and Table 8-2 show the resources and interactions of the CCP module(s). In the following sections, the operation of a CCP module is described with respect to CCP1. CCP2 operates the same as CCP1 except where noted.

CCP1 Module:

Capture/Compare/PWM Register 1 (CCPR1) is comprised of two 8-bit registers: CCPR1L (low byte) and CCPR1H (high byte). The CCP1CON register controls the operation of CCP1. The special event trigger is generated by a compare match and will reset Timer1.

CCP2 Module:

Capture/Compare/PWM Register 2 (CCPR2) is comprised of two 8-bit registers: CCPR2L (low byte) and CCPR2H (high byte). The CCP2CON register controls the operation of CCP2. The special event trigger is generated by a compare match and will reset Timer1 and start an A/D conversion (if the A/D module is enabled).

Additional information on CCP modules is available in the PICmicro® Mid-Range MCU Family Reference Manual (DS33023) and in application note *AN594, "Using the CCP Module(s)"* (DS00594).

TABLE 8-1: CCP MODE – TIMER RESOURCES REQUIRED

CCP Mode	Timer Resource
Capture	Timer1
Compare	Timer1
PWM	Timer2

TABLE 8-2: INTERACTION OF TWO CCP MODULES

CCPx Mode	CCPy Mode	Interaction
Capture	Capture	Same TMR1 time base
Capture	Compare	The compare should be configured for the special event trigger which clears TMR1
Compare	Compare	The compare(s) should be configured for the special event trigger which clears TMR1
PWM	PWM	The PWMs will have the same frequency and update rate (TMR2 interrupt)
PWM	Capture	None
PWM	Compare	None

REGISTER 8-1: CCP1CON REGISTER/CCP2CON REGISTER (ADDRESS 17h/1Dh)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CCPxX	CCPxY	CCPxM3	CCPxM2	CCPxM1	CCPxM0
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5-4 **CCPxX:CCPxY**: PWM Least Significant bits

<u>Capture mode</u>:
Unused.

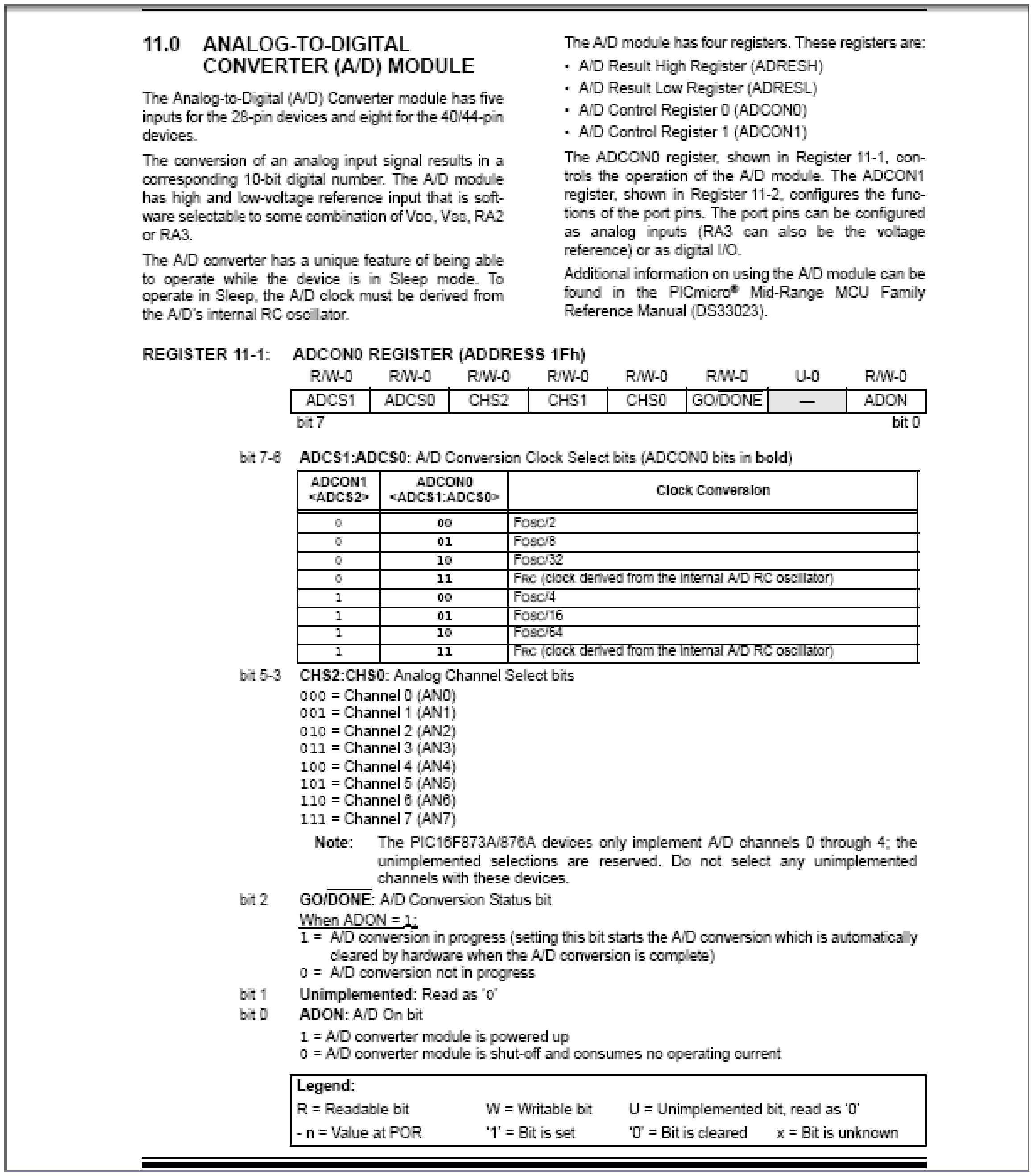
<u>Compare mode:</u>
Unused.

<u>PWM mode:</u>
These bits are the two LSbs of the PWM duty cycle. The eight MSbs are found in CCPRxL.

bit 3-0 **CCPxM3:CCPxM0**: CCPx Mode Select bits

`0000` = Capture/Compare/PWM disabled (resets CCPx module)
`0100` = Capture mode, every falling edge
`0101` = Capture mode, every rising edge
`0110` = Capture mode, every 4th rising edge
`0111` = Capture mode, every 16th rising edge
`1000` = Compare mode, set output on match (CCPxIF bit is set)
`1001` = Compare mode, clear output on match (CCPxIF bit is set)
`1010` = Compare mode, generate software interrupt on match (CCPxIF bit is set, CCPx pin is unaffected)
`1011` = Compare mode, trigger special event (CCPxIF bit is set, CCPx pin is unaffected); CCP1 resets TMR1; CCP2 resets TMR1 and starts an A/D conversion (if A/D module is enabled)
`11xx` = PWM mode

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown



11.0 ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE

The Analog-to-Digital (A/D) Converter module has five inputs for the 28-pin devices and eight for the 40/44-pin devices.

The conversion of an analog input signal results in a corresponding 10-bit digital number. The A/D module has high and low-voltage reference input that is software selectable to some combination of VDD, VSS, RA2 or RA3.

The A/D converter has a unique feature of being able to operate while the device is in Sleep mode. To operate in Sleep, the A/D clock must be derived from the A/D's internal RC oscillator.

The A/D module has four registers. These registers are:

- A/D Result High Register (ADRESH)
- A/D Result Low Register (ADRESL)
- A/D Control Register 0 (ADCON0)
- A/D Control Register 1 (ADCON1)

The ADCON0 register, shown in Register 11-1, controls the operation of the A/D module. The ADCON1 register, shown in Register 11-2, configures the functions of the port pins. The port pins can be configured as analog inputs (RA3 can also be the voltage reference) or as digital I/O.

Additional information on using the A/D module can be found in the PICmicro® Mid-Range MCU Family Reference Manual (DS33023).

REGISTER 11-1: ADCON0 REGISTER (ADDRESS 1Fh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	—	ADON
bit 7							bit 0

bit 7-6 **ADCS1:ADCS0:** A/D Conversion Clock Select bits (ADCON0 bits in **bold**)

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	Clock Conversion
0	**00**	FOSC/2
0	**01**	FOSC/8
0	**10**	FOSC/32
0	**11**	FRC (clock derived from the internal A/D RC oscillator)
1	**00**	FOSC/4
1	**01**	FOSC/16
1	**10**	FOSC/64
1	**11**	FRC (clock derived from the internal A/D RC oscillator)

bit 5-3 **CHS2:CHS0:** Analog Channel Select bits
000 = Channel 0 (AN0)
001 = Channel 1 (AN1)
010 = Channel 2 (AN2)
011 = Channel 3 (AN3)
100 = Channel 4 (AN4)
101 = Channel 5 (AN5)
110 = Channel 6 (AN6)
111 = Channel 7 (AN7)

Note: The PIC16F873A/876A devices only implement A/D channels 0 through 4; the unimplemented selections are reserved. Do not select any unimplemented channels with these devices.

bit 2 **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit
When ADON = 1:
1 = A/D conversion in progress (setting this bit starts the A/D conversion which is automatically cleared by hardware when the A/D conversion is complete)
0 = A/D conversion not in progress

bit 1 **Unimplemented:** Read as '0'

bit 0 **ADON:** A/D On bit
1 = A/D converter module is powered up
0 = A/D converter module is shut-off and consumes no operating current

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 11-2: ADCON1 REGISTER (ADDRESS 9Fh)

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

bit 7 **ADFM:** A/D Result Format Select bit
1 = Right justified. Six (6) Most Significant bits of ADRESH are read as '0'.
0 = Left justified. Six (6) Least Significant bits of ADRESL are read as '0'.

bit 6 **ADCS2:** A/D Conversion Clock Select bit (ADCON1 bits in shaded area and in **bold**)

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	Clock Conversion
0	00	FOSC/2
0	01	FOSC/8
0	10	FOSC/32
0	11	FRC (clock derived from the internal A/D RC oscillator)
1	00	FOSC/4
1	01	FOSC/16
1	10	FOSC/64
1	11	FRC (clock derived from the internal A/D RC oscillator)

bit 5-4 **Unimplemented:** Read as '0'

bit 3-0 **PCFG3:PCFG0:** A/D Port Configuration Control bits

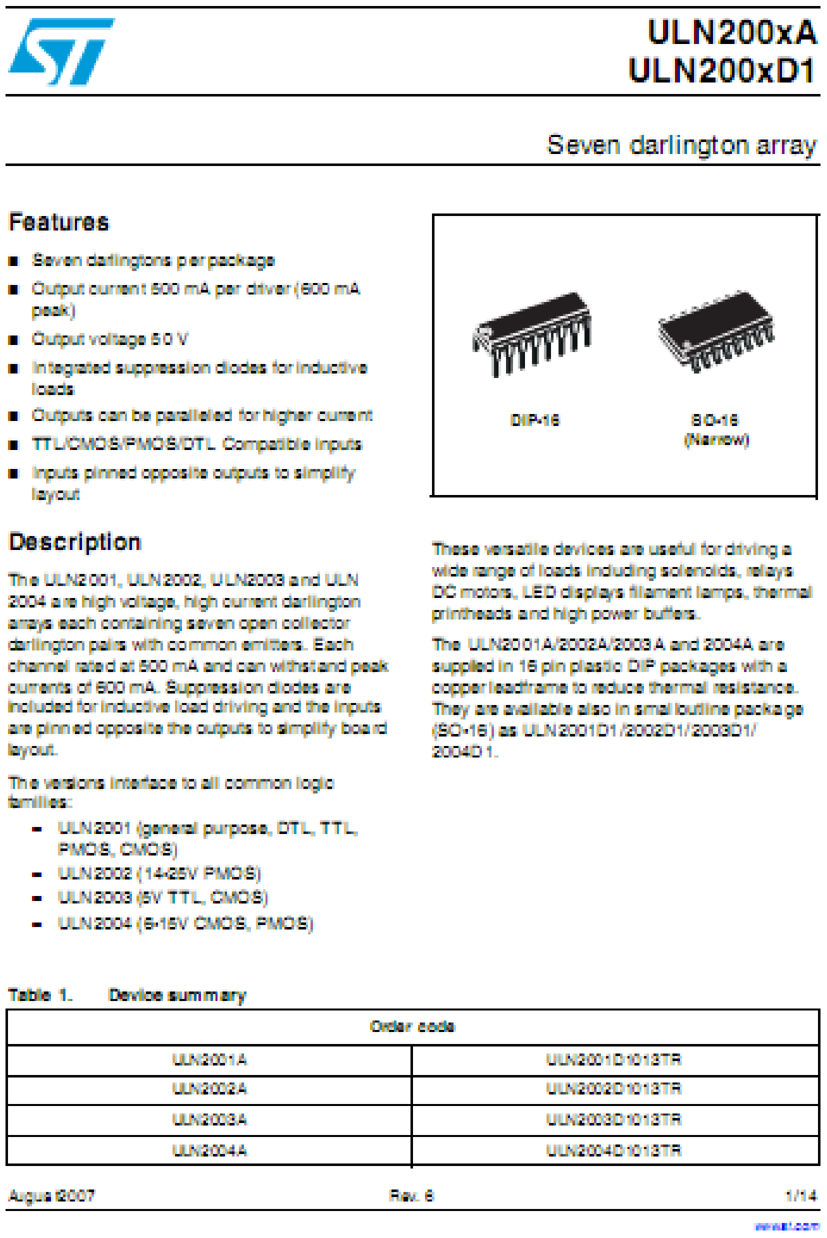
PCFG <3:0>	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	VREF+	VREF-	C/R
0000	A	A	A	A	A	A	A	A	VDD	VSS	8/0
0001	A	A	A	A	VREF+	A	A	A	AN3	VSS	7/1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5/0
0011	D	D	D	A	VREF+	A	A	A	AN3	VSS	4/1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3/0
0101	D	D	D	D	VREF+	D	A	A	AN3	VSS	2/1
011x	D	D	D	D	D	D	D	D	—	—	0/0
1000	A	A	A	A	VREF+	VREF-	A	A	AN3	AN2	6/2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6/0
1010	D	D	A	A	VREF+	A	A	A	AN3	VSS	5/1
1011	D	D	A	A	VREF+	VREF-	A	A	AN3	AN2	4/2
1100	D	D	D	A	VREF+	VREF-	A	A	AN3	AN2	3/2
1101	D	D	D	D	VREF+	VREF-	A	A	AN3	AN2	2/2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1/0
1111	D	D	D	D	VREF+	VREF-	D	A	AN3	AN2	1/2

A = Analog input D = Digital I/O
C/R = # of analog input channels/# of A/D voltage references

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared x = Bit is unknown

Note: On any device Reset, the port pins that are multiplexed with analog functions (ANx) are forced to be an analog input.

ULN 2003A

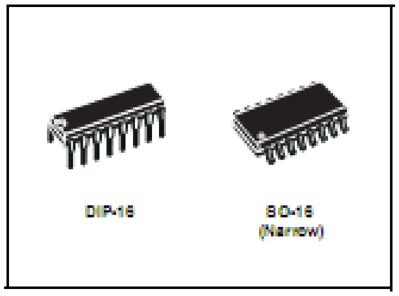


ULN200xA
ULN200xD1

Seven darlington array

Features

- Seven darlingtons per package
- Output current 500 mA per driver (600 mA peak)
- Output voltage 50 V
- Integrated suppression diodes for inductive loads
- Outputs can be paralleled for higher current
- TTL/CMOS/PMOS/DTL Compatible inputs
- Inputs pinned opposite outputs to simplify layout



Description

The ULN2001, ULN2002, ULN2003 and ULN 2004 are high voltage, high current darlington arrays each containing seven open collector darlington pairs with common emitters. Each channel rated at 500 mA and can withstand peak currents of 600 mA. Suppression diodes are included for inductive load driving and the inputs are pinned opposite the outputs to simplify board layout.

The versions interface to all common logic families:

- ULN2001 (general purpose, DTL, TTL, PMOS, CMOS)
- ULN2002 (14-25V PMOS)
- ULN2003 (5V TTL, CMOS)
- ULN2004 (6-15V CMOS, PMOS)

These versatile devices are useful for driving a wide range of loads including solenoids, relays DC motors, LED displays filament lamps, thermal printheads and high power buffers.

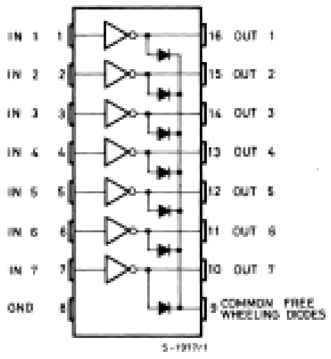
The ULN2001A/2002A/2003A and 2004A are supplied in 16 pin plastic DIP packages with a copper leadframe to reduce thermal resistance. They are available also in small outline package (SO-16) as ULN2001D1/2002D1/2003D1/2004D1.

Table 1. Device summary

Order code	
ULN2001A	ULN2001D1013TR
ULN2002A	ULN2002D1013TR
ULN2003A	ULN2003D1013TR
ULN2004A	ULN2004D1013TR

2 Pin configuration

Figure 2. Pin connections (top view)



3 Maximum ratings

Table 2. Absolute maximum ratings

Symbol	Parameter	Value	Unit
V_O	Output voltage	50	V
V_I	Input voltage (for ULN2002A/D - 2003A/D - 2004A/D)	30	V
I_C	Continuous collector current	500	mA
I_B	Continuous base current	25	mA
T_A	Operating ambient temperature range	- 20 to 85	°C
T_{STG}	Storage temperature range	- 55 to 150	°C
T_J	Junction temperature	150	°C

Table 3. Thermal data

Symbol	Parameter	DIP-16	SO-16	Unit
R_{thJA}	Thermal resistance junction-ambient Max.	70	120	°C/W

4 Electrical characteristics

Table 4. Electrical characteristics
(T_A = 25°C unless otherwise specified).

Symbol	Parameter	Test condition	Min.	Typ.	Max.	Unit
I_{CEX}	Output leakage current	V_{CE} = 50 V, (Figure 3.)			50	µA
		T_A = 70°C, V_{CE} = 50 V (Figure 3.)			100	
		T_A = 70°C for ULN2002, V_{CE} = 50 V, V_I = 6 V (Figure 4.)			500	
		T_A = 70°C for ULN2002, V_{CE} = 50 V, V_I = 1V (Figure 4.)			500	
$V_{CE(SAT)}$	Collector-emitter saturation voltage (Figure 5.)	I_C = 100 mA, I_B = 250 µA		0.9	1.1	V
		I_C = 200 mA, I_B = 350 µA		1.1	1.3	
		I_C = 350 mA, I_B = 500 µA		1.3	1.6	
$I_{I(ON)}$	Input current (Figure 6.)	for ULN2002, V_I = 17 V		0.82	1.25	mA
		for ULN2003, V_I = 3.85V		0.93	1.35	
		for ULN2004, V_I = 5 V		0.35	0.5	
		V_I = 12 V		1	1.45	
$I_{I(OFF)}$	Input current (Figure 7.)	T_A = 70°C, I_C = 500 µA	50	65		µA
$V_{I(ON)}$	Input voltage (Figure 8.)	V_{CE} = 2 V, for ULN2002 I_C = 300 mA for ULN2003 I_C = 200 mA I_C = 250 mA I_C = 300 mA for ULN2004 I_C = 125 mA I_C = 200 mA I_C = 275 mA I_C = 350 mA			13 2.4 2.7 3 5 6 7 8	V
h_{FE}	DC Forward current gain (Figure 5.)	for ULN2001, V_{CE} = 2 V, I_C = 350 mA	1000			
C_I	Input capacitance			15	25	pF
t_{PLH}	Turn-on delay time	0.5 V_I to 0.5 V_O		0.25	1	µs
t_{PHL}	Turn-off delay time	0.5 V_I to 0.5 V_O		0.25	1	µs
I_R	Clamp diode leakage current (Figure 9.)	V_R = 50 V			50	µA
		T_A = 70°C, V_R = 50 V			100	
V_F	Clamp diode forward voltage (Figure 10.)	I_F = 350 mA		1.7	2	V

5 Test circuits

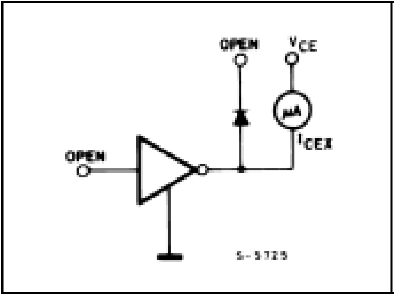


Figure 3. Output leakage current

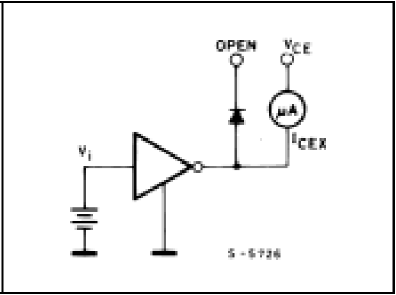


Figure 4. Output leakage current (for ULN2002 only)

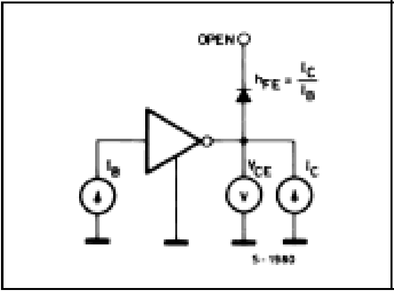


Figure 5. Collector-emitter saturation voltage

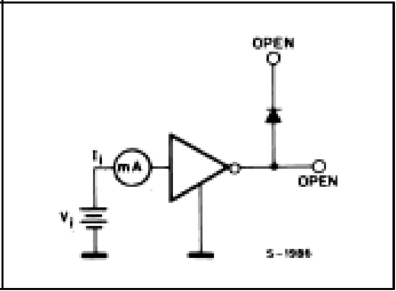


Figure 6. Input current (ON)

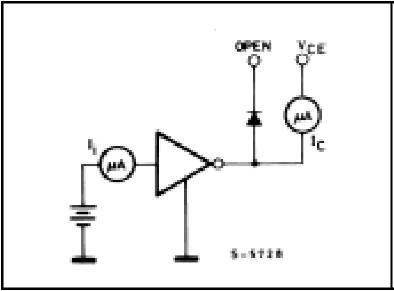


Figure 7. Input current (OFF)

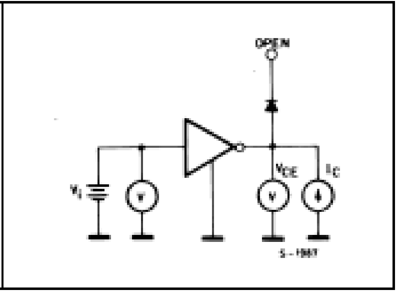
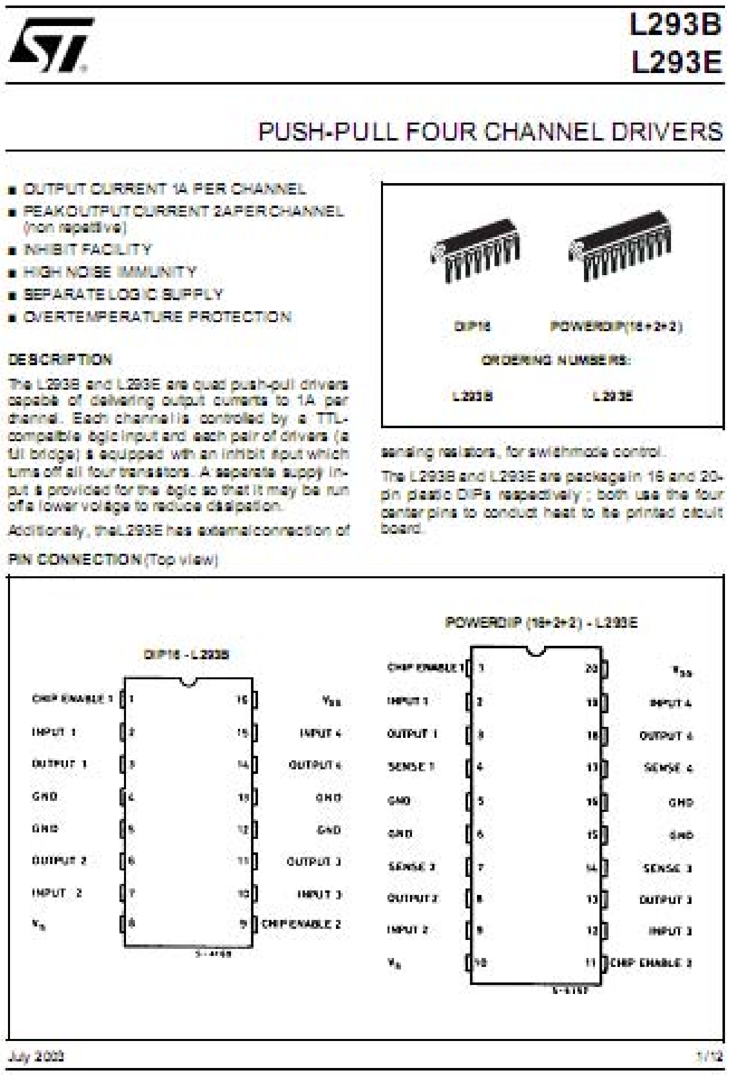


Figure 8. Input voltage

<u>L 293B</u>



L293B
L293E

PUSH-PULL FOUR CHANNEL DRIVERS

- OUTPUT CURRENT 1A PER CHANNEL
- PEAK OUTPUT CURRENT 2A PER CHANNEL (non repetitive)
- INHIBIT FACILITY
- HIGH NOISE IMMUNITY
- SEPARATE LOGIC SUPPLY
- OVERTEMPERATURE PROTECTION

DIP16 POWERDIP (16+2+2)

ORDERING NUMBERS:

L293B L293E

DESCRIPTION

The L293B and L293E are quad push-pull drivers capable of delivering output currents to 1A per channel. Each channel is controlled by a TTL-compatible logic input and each pair of drivers (a full bridge) is equipped with an inhibit input which turns off all four transistors. A separate supply input is provided for the logic so that it may be run off a lower voltage to reduce dissipation.

Additionally, the L293E has external connection of sensing resistors, for switchmode control.

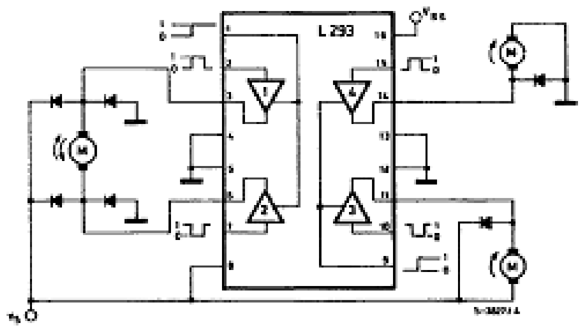
The L293B and L293E are package in 16 and 20-pin plastic DIPs respectively ; both use the four center pins to conduct heat to the printed circuit board.

PIN CONNECTION (Top view)

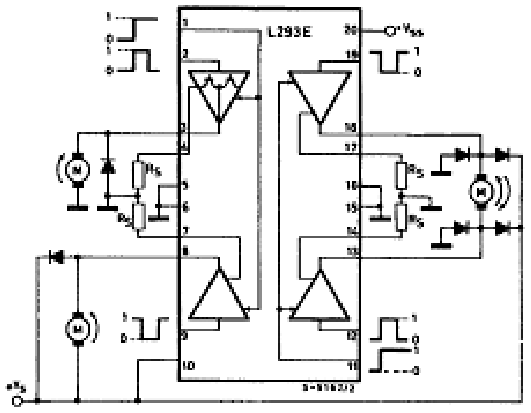
July 2003 1/12

BLOCK DIAGRAMS

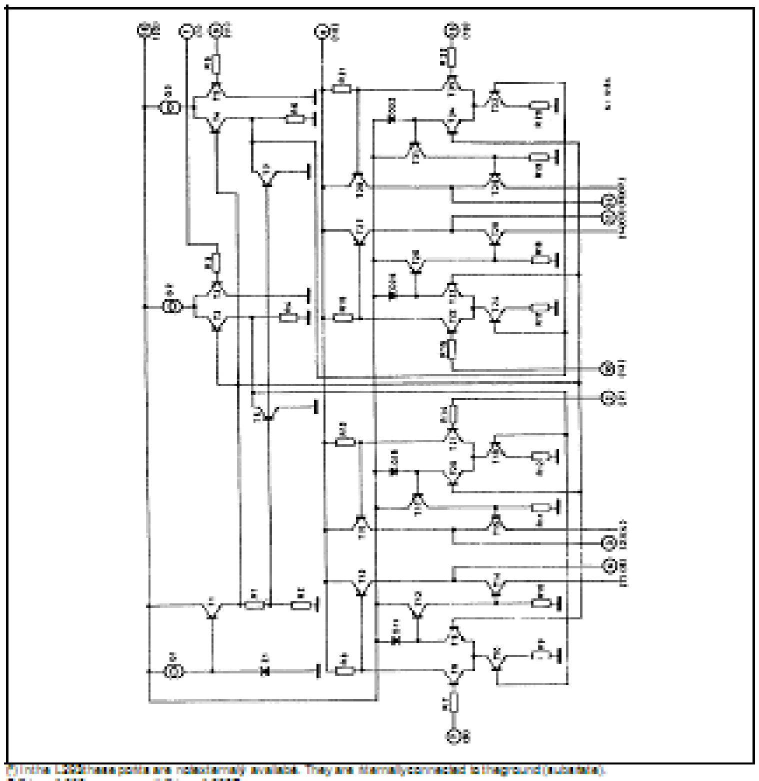
DIP16 - L293B



POWERDIP (16+2+2) - L293E



SCHEMATIC DIAGRAM



(*) In the L293 these points are not externally available. They are internally connected to the ground (substrate).
O Pins of L293 () Pins of L293E.

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_s	Supply Voltage	36	V
V_{ss}	Logic Supply Voltage	36	V
V_i	Input Voltage	7	V
V_{inh}	Inhibit Voltage	7	V
I_{out}	Peak Output Current (non repetitive t = 5ms)	2	A
P_{tot}	Total Power Dissipation at $T_{ground-pins}$ = 80°C	5	W
T_{stg}, T_j	Storage and Junction Temperature	–40 to +150	°C

THERMAL DATA

Symbol	Parameter		Value	Unit
$R_{th\ j\text{-}case}$	Thermal Resistance Junction-case	Max.	14	°C/W
$R_{th\ j\text{-}amb}$	Thermal Resistance Junction-ambient	Max.	80	°C/W

ELECTRICAL CHARACTERISTICS

Symbol	Parameter	Test Condition	Min.	Typ.	Max.	Unit
V_s	SupplyVoltage		V_{ss}		36	V
V_{ss}	Logic Suppy Voltage		4.5		36	V
I_s	Total Quiescent Supply Current	V_i = L; I_O = 0; V_{inh} = H		2	6	mA
		V_i = h; I_O = 0; V_{inh} = H		16	24	mA
		V_{inh} = L			4	mA
I_{ss}	Total Quiescent Logic Supply Current	V_i = L; I_O = 0; V_{inh} = H		44	60	mA
		V_i = h; I_O = 0; V_{inh} = H		16	22	mA
		V_{inh} = L		16	24	mA
V_{iL}	Input Low Voltage		-0.3		1.5	V
V_{iH}	Input High Voltage	$V_{SS} \leq 7V$	2.3		V_{ss}	V
		$V_{SS} > 7V$	2.3		7	V
I_{iL}	Low Voltage Input Current	V_i = 1.5V			-10	μA
I_{iH}	High Voltage Input Current	$2.3V \leq V_{iH} \leq V_{ss} - 0.6V$		30	100	μA
V_{inhL}	Inhibit Low Voltage		-0.3		1.5	V
V_{inhH}	Inhibit High Voltage	$V_{SS} \leq 7V$	2.3		V_{ss}	V
		$V_{SS} > 7V$	2.3		7	V
I_{inhL}	Low Voltage Inhibit Current	V_{inhL} = 1.5V		-30	-100	μA
I_{inhH}	High Voltage Inhibit Current	$2.3V \leq V_{inhH} \leq V_{ss} - 0.6V$			±10	μA
V_{CEsatH}	Source Output Saturation Voltage	I_O = -1A		14	1.8	V
V_{CEsatL}	Sink Output Saturation Voltage	I_O = 1A		12	1.8	V
V_{SENS}	Sensing Voltage (pins 4,7, 14 ,17) (**)				2	V
t_r	Rise Time	0.1 to 0.9 V_O (*)		250		ns
t_f	Fall Time	0.9 to 0.1 V_O (*)		250		ns
t_{on}	Turn-on Delay	0.5 V_i to 0.5 V_O (*)		750		ns
t_{off}	Turn-off Delay	0.5 V_i to 0.5 V_O (*)		200		ns

*See figure 1
** Referred to L293E

TRUTH TABLE

Vi (each channel)	Vo	V_{inh} (**)
H	H	H
L	L	H
H	X (*)	L
L	X (*)	L

(*) High output impedance
(**) Relative to the considerate channel

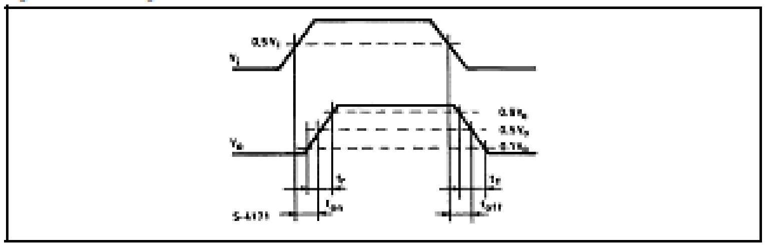


Figure 1. Switching Timers

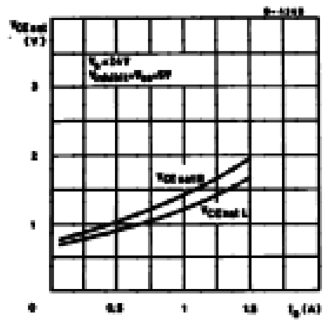


Figure 2. Saturation voltage versus Output Current

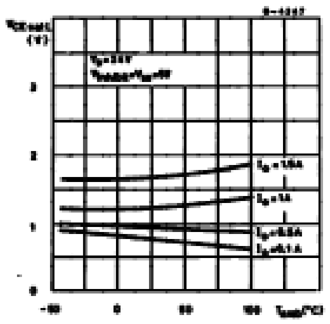


Figure4. Sink Saturation Voltage versus Ambient Temperature

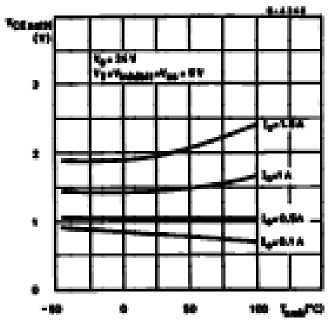


Figure 3. Source Saturation Voltage versus Ambient Temperature

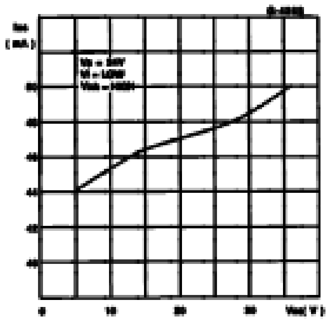


Figure 5. Quiescent Logic Supply Current versus Logic Supply Voltage

Figure 6. Output Voltage versus Input Voltage

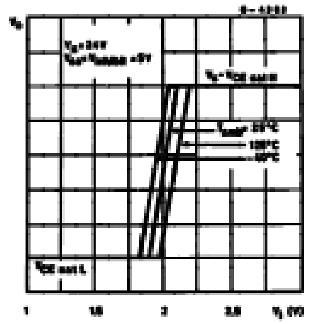
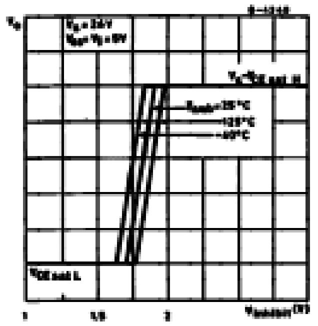
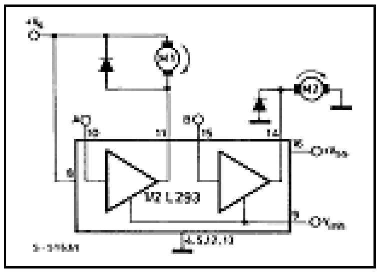


Figure 7. Output Voltage versus Inhibit Voltage



APPLICATION INFORMATION

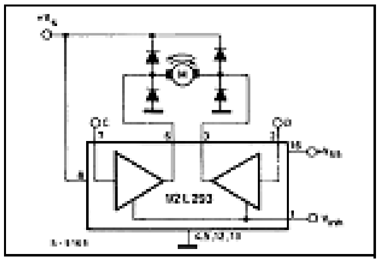
Figure 8. DC Motor Controls
(with connection to ground and to the supply voltage)



V_{inh}	A	M1	B	M2
H	H	Fast Motor Stop	H	Run
H	L	Run	L	Fast Motor Stop
L	X	Free Running	X	Free Running
		Motor Stop		Motor Stop

L = Low H = High X = Don't Care

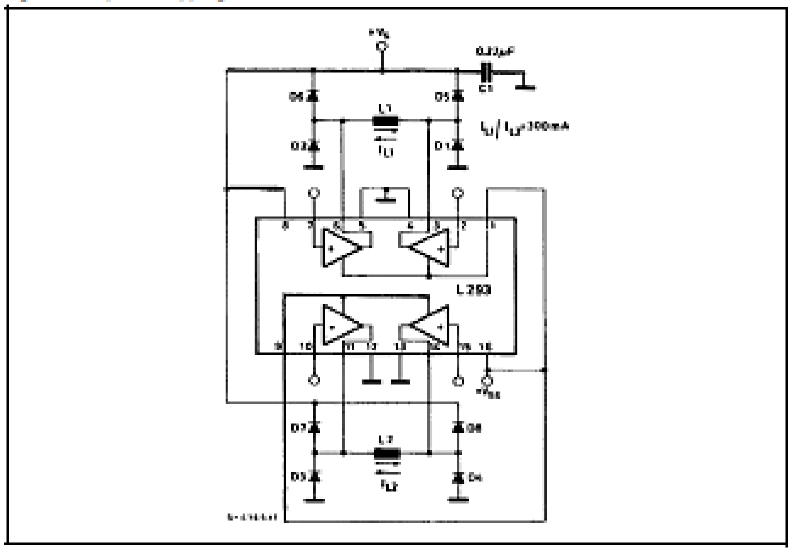
Figure 9. Bidirectional DC Motor Control



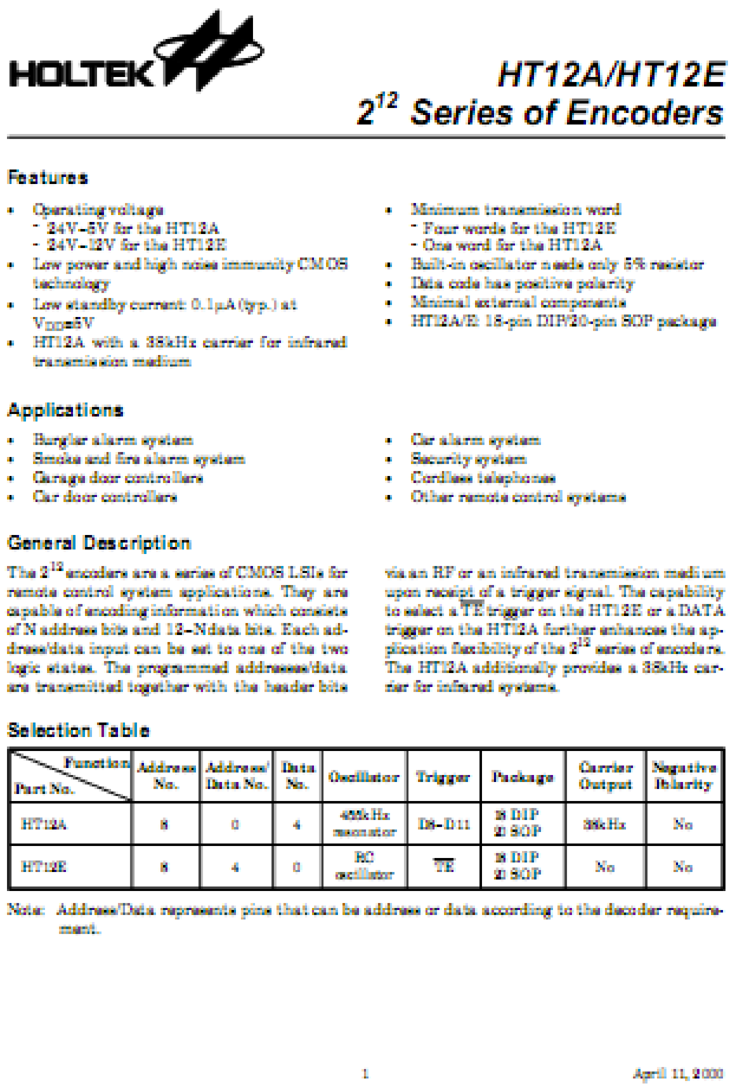
Inputs	Function	
V_{inh} = H	C = H ; D = L	Turn Right
	C = L ; D = H	Turn Left
	C = D	Fast Motor Stop
V_{inh} = L	C = X; D = X	Free Running Motor Stop

L = Low H = High X = Don't Care

Figure 10. Bipolar Stepping Motor Control



HT-12E



HOLTEK

HT12A/HT12E
2^{12} Series of Encoders

Features

- Operating voltage
  - 2.4V~5V for the HT12A
  - 2.4V~12V for the HT12E
- Low power and high noise immunity CMOS technology
- Low standby current: 0.1μA (typ.) at V_{DD}=5V
- HT12A with a 38kHz carrier for infrared transmission medium
- Minimum transmission word
  - Four words for the HT12E
  - One word for the HT12A
- Built-in oscillator needs only 5% resistor
- Data code has positive polarity
- Minimal external components
- HT12A/E: 18-pin DIP/20-pin SOP package

Applications

- Burglar alarm system
- Smoke and fire alarm system
- Garage door controllers
- Car door controllers
- Car alarm system
- Security system
- Cordless telephones
- Other remote control systems

General Description

The 2^{12} encoders are a series of CMOS LSIs for remote control system applications. They are capable of encoding information which consists of N address bits and 12−N data bits. Each address/data input can be set to one of the two logic states. The programmed addresses/data are transmitted together with the header bits via an RF or an infrared transmission medium upon receipt of a trigger signal. The capability to select a $\overline{TE}$ trigger on the HT12E or a DATA trigger on the HT12A further enhances the application flexibility of the 2^{12} series of encoders. The HT12A additionally provides a 38kHz carrier for infrared systems.

Selection Table

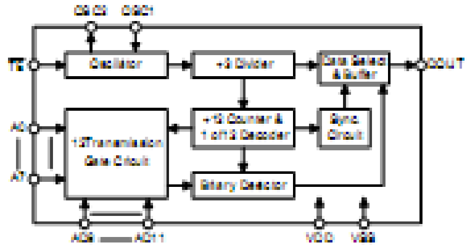
Function / Part No.	Address No.	Address/ Data No.	Data No.	Oscillator	Trigger	Package	Carrier Output	Negative Polarity
HT12A	8	0	4	455kHz resonator	D8~D11	18 DIP 20 SOP	38kHz	No
HT12E	8	4	0	RC oscillator	$\overline{TE}$	18 DIP 20 SOP	No	No

Note: Address/Data represents pins that can be address or data according to the decoder requirement.

Block Diagram

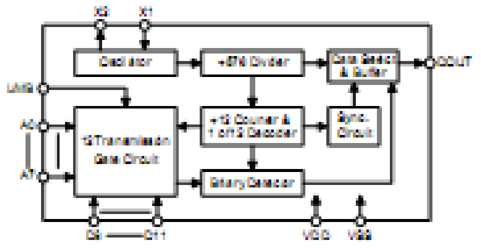
$\overline{\text{TE}}$ trigger

HT12E



DATA trigger

HT12A



Note: The address data pins are available in various combinations (refer to the address/data table).

Electrical Characteristics

HT12A

Ta=25°C

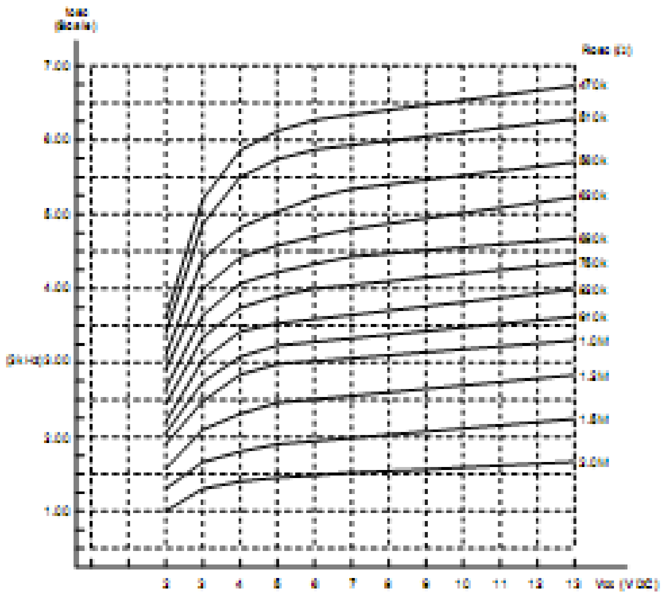
Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	2.4	3	5	V
I_{STB}	Standby Current	3V	Oscillator stops	—	0.1	1	μA
		5V		—	0.1	1	μA
I_{DD}	Operating Current	3V	No load f_{OSC}=455kHz	—	200	400	μA
		5V		—	400	800	μA
I_{DOUT}	Output Drive Current	5V	V_{OH}=0.9V_{DD} (Source)	-1	-1.6	—	mA
			V_{OL}=0.1V_{DD} (Sink)	2	3.2	—	mA
V_{IH}	"H" Input Voltage	—	—	0.8V_{DD}	—	V_{DD}	V
V_{IL}	"L" Input Voltage	—	—	0	—	0.2V_{DD}	V
R_{DATA}	D8~D11 Pull-high Resistance	5V	V_{DATA}=0V	—	150	300	kΩ

HT12E

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	2.4	5	12	V
I_{STB}	Standby Current	3V	Oscillator stops	—	0.1	1	μA
		12V		—	2	4	μA
I_{DD}	Operating Current	3V	No load f_{OSC}=3kHz	—	40	80	μA
		12V		—	150	300	μA
I_{DOUT}	Output Drive Current	5V	V_{OH}=0.9V_{DD} (Source)	-1	-1.6	—	mA
			V_{OL}=0.1V_{DD} (Sink)	1	1.6	—	mA
V_{IH}	"H" Input Voltage	—	—	0.8V_{DD}	—	V_{DD}	V
V_{IL}	"L" Input Voltage	—	—	0	—	0.2V_{DD}	V
f_{OSC}	Oscillator Frequency	5V	R_{OSC}=1.1MΩ	—	3	—	kHz
$R_{\overline{TE}}$	$\overline{TE}$ Pull-high Resistance	5V	$V_{\overline{TE}}$=0V	—	1.5	3	MΩ

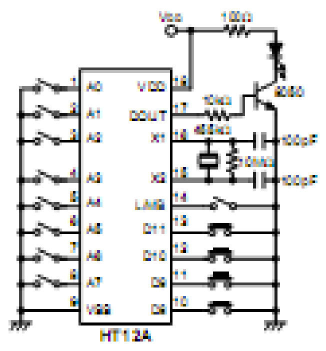
Oscillator frequency vs supply voltage

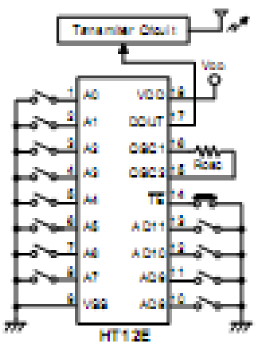


The recommended oscillator frequency is f_{OSCD} (decoder) $\cong$ 50 f_{OSCE} (HT12E encoder)

$\cong \frac{1}{3} f_{OSCE}$ (HT12A encoder)

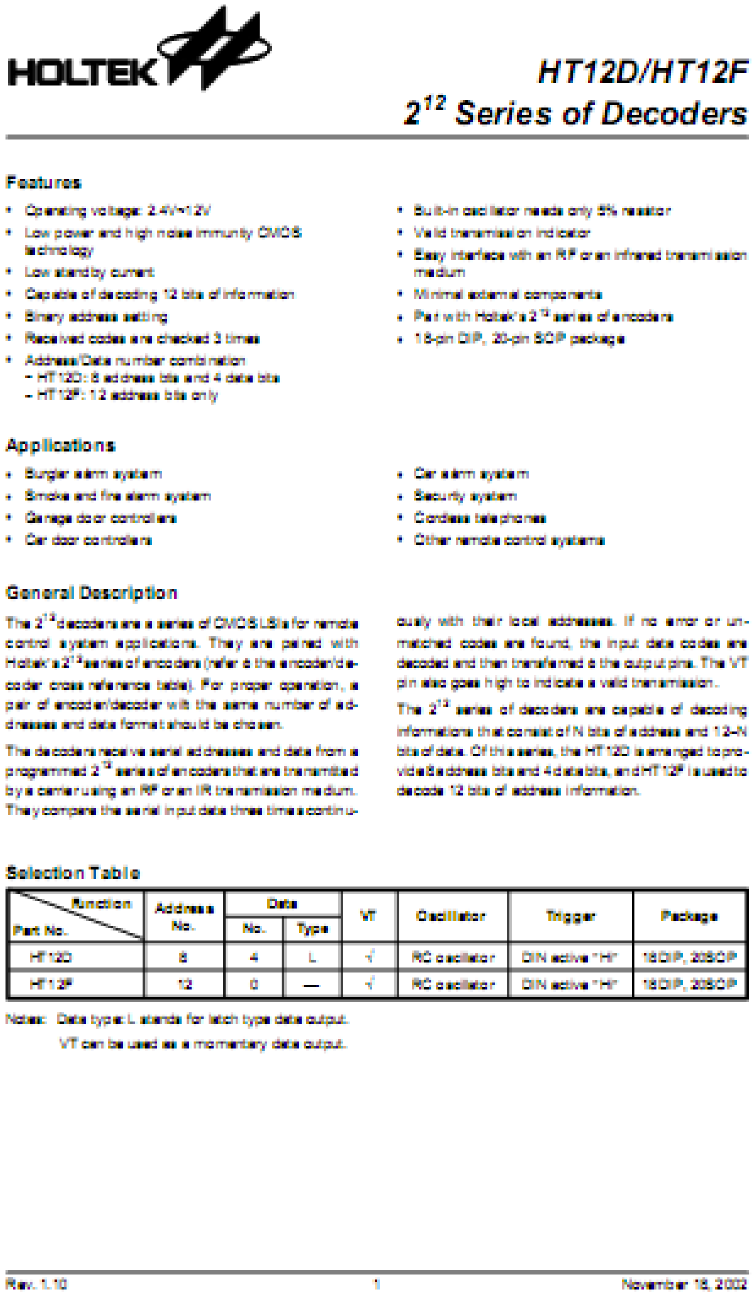
Application Circuits





Note: Typical infrared diode: EL-1L2 (KODENSHI CORP.)
Typical RF transmitter: JR-220 (JUWA CORP.)

HT-12D



HOLTEK

HT12D/HT12F
2^{12} Series of Decoders

Features

- Operating voltage: 2.4V~12V
- Low power and high noise immunity CMOS technology
- Low standby current
- Capable of decoding 12 bits of information
- Binary address setting
- Received codes are checked 3 times
- Address/Data number combination
  - HT12D: 8 address bits and 4 data bits
  - HT12F: 12 address bits only
- Built-in oscillator needs only 5% resistor
- Valid transmission indicator
- Easy interface with an RF or an infrared transmission medium
- Minimal external components
- Pair with Holtek's 2^{12} series of encoders
- 18-pin DIP, 20-pin SOP package

Applications

- Burglar alarm system
- Smoke and fire alarm system
- Garage door controllers
- Car door controllers
- Car alarm system
- Security system
- Cordless telephones
- Other remote control systems

General Description

The 2^{12} decoders are a series of CMOS LSIs for remote control system applications. They are paired with Holtek's 2^{12} series of encoders (refer to the encoder/decoder cross reference table). For proper operation, a pair of encoder/decoder with the same number of addresses and data format should be chosen.

The decoders receive serial addresses and data from a programmed 2^{12} series of encoders that are transmitted by a carrier using an RF or an IR transmission medium. They compare the serial input data three times continuously with their local addresses. If no error or unmatched codes are found, the input data codes are decoded and then transferred to the output pins. The VT pin also goes high to indicate a valid transmission.

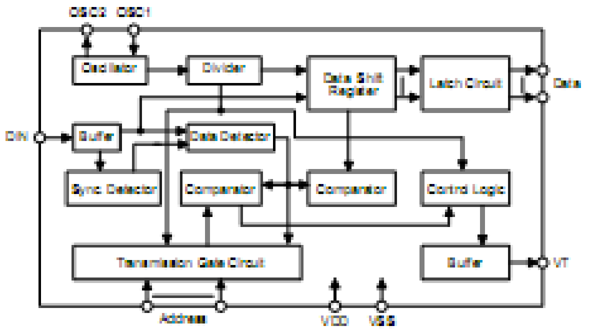
The 2^{12} series of decoders are capable of decoding informations that consist of N bits of address and 12–N bits of data. Of this series, the HT12D is arranged to provide 8 address bits and 4 data bits, and HT12F is used to decode 12 bits of address information.

Selection Table

Function / Part No.	Address No.	Data		VT	Oscillator	Trigger	Package
		No.	Type				
HT12D	8	4	L	√	RC oscillator	DIN active "Hi"	18DIP, 20SOP
HT12F	12	0	—	√	RC oscillator	DIN active "Hi"	18DIP, 20SOP

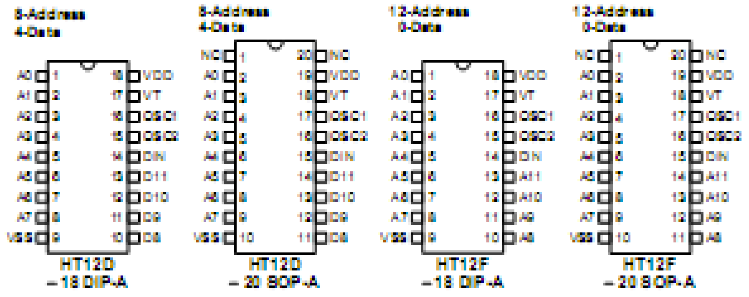
Notes: Data type: L stands for latch type data output.
VT can be used as a momentary data output.

Block Diagram



Note: The address/data pins are available in various combinations (see the address/data table).

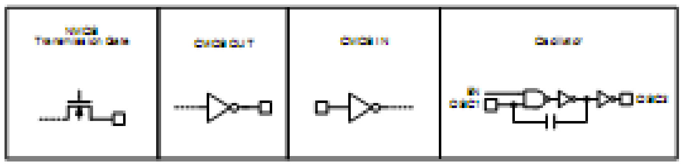
Pin Assignment


8-Address 4-Data
A0 1, A1 2, A2 3, A3 4, A4 5, A5 6, A6 7, A7 8, VSS 9
18 VDD, 17 VT, 16 OSC1, 15 OSC2, 14 DIN, 13 D11, 12 D10, 11 D9, 10 D8
HT12D – 18 DIP-A
8-Address 4-Data
NC 1, A0 2, A1 3, A2 4, A3 5, A4 6, A5 7, A6 8, A7 9, VSS 10
20 NC, 19 VDD, 18 VT, 17 OSC1, 16 OSC2, 15 DIN, 14 D11, 13 D10, 12 D9, 11 D8
HT12D – 20 SOP-A
12-Address 0-Data
A0 1, A1 2, A2 3, A3 4, A4 5, A5 6, A6 7, A7 8, VSS 9
18 VDD, 17 VT, 16 OSC1, 15 OSC2, 14 DIN, 13 A11, 12 A10, 11 A9, 10 A8
HT12F – 18 DIP-A
12-Address 0-Data
NC 1, A0 2, A1 3, A2 4, A3 5, A4 6, A5 7, A6 8, A7 9, VSS 10
20 NC, 19 VDD, 18 VT, 17 OSC1, 16 OSC2, 15 DIN, 14 A11, 13 A10, 12 A9, 11 A8
HT12F – 20 SOP-A

Pin Description

Pin Name	I/O	Internal Connection	Description
A0~A11 (HT12F)	I	NMOS Transmission Gate	Input pins for address A0~A11 setting These pins can be externally set to VSS or left open.
A0~A7 (HT12D)			Input pins for address A0~A7 setting These pins can be externally set to VSS or left open.
D8~D11 (HT12D)	O	CMOS OUT	Output data pins, power-on state is low.
DIN	I	CMOS IN	Serial data input pin
VT	O	CMOS OUT	Valid transmission, active high
OSC1	I	Oscillator	Oscillator input pin
OSC2	O	Oscillator	Oscillator output pin
VSS	—	—	Negative power supply, ground
VDD	—	—	Positive power supply

Approximate internal connection circuits



Absolute Maximum Ratings

Supply Voltage ....−0.3V to 13V

Input Voltage ....V_{SS}−0.3 to V_{DD}+0.3V

Storage Temperature ....−50°C to 125°C

Operating Temperature ....−20°C to 75°C

Note: These are stress ratings only. Stresses exceeding the range specified under "Absolute Maximum Ratings" may cause substantial damage to the device. Functional operation of this device at other conditions beyond those listed in the specification is not implied and prolonged exposure to extreme conditions may affect device reliability.

Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	2.4	5	12	V
I_{STB}	Standby Current	5V	Oscillator stops	—	0.1	1	μA
		12V		—	2	4	μA
I_{DD}	Operating Current	5V	No load, f_{OSC}= 150kHz	—	200	400	μA
I_O	Data Output Source Current (D8~D11)	5V	V_{OH}=4.5V	−1	−1.6	—	mA
	Data Output Sink Current (D8~D11)	5V	V_{OL}=0.5V	1	1.6	—	mA
I_{VT}	VT Output Source Current	5V	V_{OH}=4.5V	−1	−1.6	—	mA
	VT Output Sink Current		V_{OL}=0.5V	1	1.6	—	mA
V_{IH}	"H" Input Voltage	5V	—	3.5	—	5	V
V_{IL}	"L" Input Voltage	5V	—	0	—	1	V
f_{OSC}	Oscillator Frequency	5V	R_{OSC}=51kΩ	—	150	—	kHz

Encoder/Decoder cross reference table

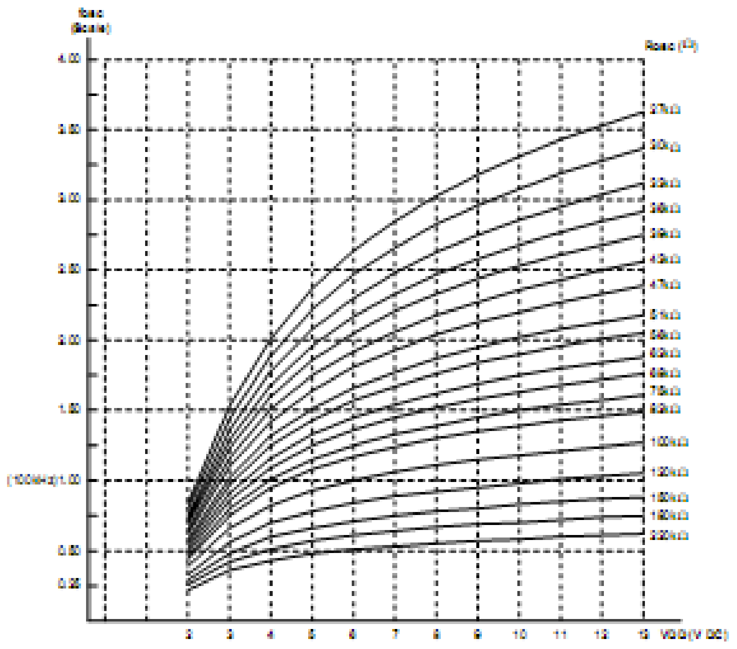
Decoders Part No.	Data Pins	Address Pins	VT	Pair Encoder	Package			
					Encoder		Decoder	
					DIP	SOP	DIP	SOP
HT12D	4	8	√	HT12A HT12E	18	20	18	20
HT12F	0	12	√	HT12A HT12E	18	20	18	20

Address/Data sequence

The following table provides address/data sequence for various models of the 2^{12} series of decoders.

Part No.	Address/Data Bits											
	0	1	2	3	4	5	6	7	8	9	10	11
HT12D	A0	A1	A2	A3	A4	A5	A6	A7	D8	D9	D10	D11
HT12F	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11

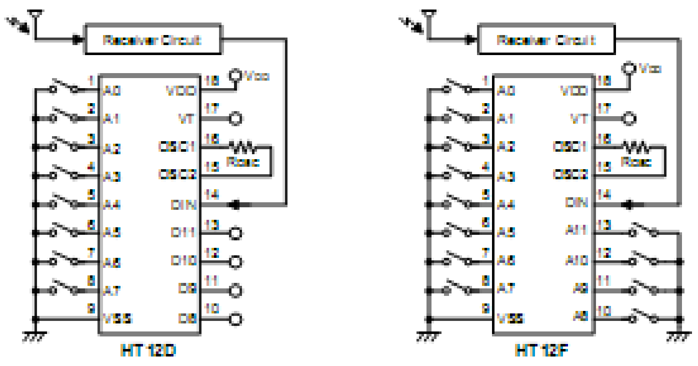
Oscillator frequency vs supply voltage



Note: The recommended oscillator frequency is f_{OSCD} (decoder) $\cong 50\ f_{OSCE}$ (HT12E encoder)

$\cong \frac{1}{3} f_{OSCE}$ (HT12A encoder).

Application Circuits


Receiver Circuit
A0
A1
A2
A3
A4
A5
A6
A7
VSS
VDD
VT
OSC1
OSC2
DIN
D11
D10
D9
D8
Vcc
Rosc
HT 12D
Receiver Circuit
A11
A10
A9
A8
Vcc
Rosc
HT 12F