PSZ 19:16 (Pind. 1/07)

UNIVERSITI TEKNOLOGI MALAYSIA

DECLARATION OF THESIS / UNDERGRADUATE PROJECT PAPER AND COPYRIGHT

Author's full name : **SUHAILA BINTI MOHD NAJIB**

Date of birth : **12th MAY 1988**

Title : **MICROCONTROLLER BASED REMOTE LOCATOR USING RADIO FREQUENCY SIGNAL**

Academic Session : **2009/2010**

I declare that this thesis is classified as :

☐	**CONFIDENTIAL**	(Contains confidential information under the Official Secret Act 1972)*
☐	**RESTRICTED**	(Contains restricted information as specified by the organisation where research was done)*
☑	**OPEN ACCESS**	I agree that my thesis to be published as online open access (full text)

I acknowledged that Universiti Teknologi Malaysia reserves the right as follows :

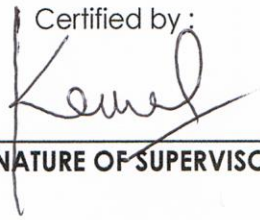
1. The thesis is the property of Universiti Teknologi Malaysia.
2. The Library of Universiti Teknologi Malaysia has the right to make copies for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

SIGNATURE

880512-13-5194
(NEW IC NO. /PASSPORT NO.)

Date : 30th April 2010

Certified by :



SIGNATURE OF SUPERVISOR

EN.KAMAL BIN KHALIL
NAME OF SUPERVISOR

Date : April 2010

NOTES : * If the thesis is CONFIDENTIAL or RESTRICTED, please attach with the letter from the organisation with period and reasons for confidentiality or restriction.

"I hereby declare that I have read this thesis and in my opinion this thesis is sufficient in terms of scope and quality for the award of the degree of Bachelor of Engineering (Computer)"

Signature : 

Name : EN.KAMAL BIN KHALIL

Date : APRIL 2010

MICROCONTROLLER BASED REMOTE LOCATOR USING RADIO FREQUENCY SIGNAL

SUHAILA BINTI MOHD NAJIB

THIS THESIS IS SUBMITTED IN PART FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE BACHELOR OF ENGINEERING (COMPUTER)

FACULTY OF ELECTRICAL ENGINEERING

UNIVERSITI TEKNOLOGI MALAYSIA

APRIL 2010

I declare that this thesis entitled "Microcontroller *Based Remote Locater Using Radio Frequency Signal"* is the result of my own research except as cited in the references. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature : 

Name : SUHAILA BINTI MOHD NAJIB

Date : 30th APRIL 2010

Special dedication to my beloved parents:-

Hj. Mohd Najib Bin Abd Majid and Hjh. Mariah Bt Abd Kadir

for their advice and concern

Brother and sisters:-

Mohd Noor Halimy Bin Mohd Najib

Mohd Noor Ideal Bin Mohd Najib

Anis Binti Ismail

Suhana Bt Mohd Najib

Surizza Bt Mohd Najib

Sofya Suraya Bt Mohd Najib

for their support

Thank you to my best friend:-

Mohd Asrul Azri Bin Husaini

for their guides and inspiration

ACKNOWLEDGEMENT

Alhamdulillah, all praise and gratitude to Allah S.W.T for giving me strength to finish my Final Year Project and a good health for the last two semesters. Without blessed from Him, I would not have been able to be at this stage.

I would like to express my appreciation to my supervisor, En. Kamal Bin Khalil for his valuable guidance, suggestion and full support in all aspect during my Final Year Project.

My deepest appreciation, thanks and love goes to my family members who have always been very supportive and willing to share all my joy and pain.

Appreciation is further extended to all my friends for their support and ideas.

Finally, my heartfelt appreciation goes to all, who have directly or indirectly helped me in completing my thesis.

ABSTRACT

This thesis describes the system for finding lost object. It is briefly explains the remote locater which will operates in the radio frequency range. In this project, Radio Frequency range used is 315MHz. At this frequency, the operating distance can be up to 100 meters. The operating distance is really depends on the supply voltage and the antenna. This project will be covered both software and hardware. Hardware is implemented using microcontrollers manufactured by Microchip. Algorithm code for both transmitter and receivers were done using MPLAB IDE Assembler. Circuit schematic is designed and verified using ExpressSCH. This remote locater will be a great gadget as it will save users' precious time.

ABSTRAK

Tesis ini menerangkan tentang satu sistem untuk mencari barang yang hilang. Secara ringkas, ia menjelaskan tentang pengesan objek yang akan beroperasi pada julat frekuensi radio. Dalam projek ini, frekuensi radio yang digunakan adalah 315MHz. Pada frekuensi ini, jarak operasi boleh mencapai sehingga 100 meter. Jarak operasi sangat bergantung pada bekalan voltan dan antena. Projek ini akan meliputi kedua-dua perkakas dan perisian. Perkakasan dilaksanakan dengan menggunakan mikropengawal yang dihasilkan oleh Microchip. Kod algoritma untuk kedua-dua pemancar dan penerima dilakukan menggunakan IDE MPLAB Assembler. Litar skematik dirancang dan disahkan menggunakan ExpressSCH. Pengesan objek ini akan menjadi alat yang terbaik kerana ia akan menjimatkan masa berharga pengguna.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATION

ICSP	⇨	In-Circuit Serial Programming
PIC	⇨	Peripheral Interface Controller
RF	⇨	Radio Frequency
TX	⇨	Transmitter
RX	⇨	Receiver
LCD	⇨	Liquid Crystal Display
IDE	⇨	Integrated Development Environment
EPROM	⇨	Electricity Programmable Read Only Memory
EEPROM	⇨	Electricity Erasable Programmable Read Only Memory
PAL	⇨	Programmable Array Logic
FPGA	⇨	Full Programmable Gate Array

CHAPTER 1

INTRODUCTION

1.1 Background

Time is now become more precious. Finding the missing item sometimes make our life miserable. In Malaysia, there is no such thing to locate the missing objects such as keys, remote control and etc. Therefore, the focus of this project is to build up a remote locater using radio frequency signal as a medium of transmission.

In this project, I will use two microcontrollers which are PIC16F877A and PIC16F876A because two parts needed to be developed. These two parts are Control unit and Remote unit which are the special name for transmitter board and receiver board respectively. These PIC's had been chosen because the architecture is easy to understand, have a relation with the project and different input output needed for different part.

This project will be divided into two parts which are control unit and base unit. In control unit, it will contain a transmitter while in remote unit, it will contain a receiver. The base unit will send the address using radio frequency signal and it will be received by remote unit. Audible sound and LED will be generated and blinking respectively.

1.2 Objectives

Generally, the main objective is to develop a communication between two microcontrollers using radio frequency signal in order to help the people to have a easier way of life. In Malaysia, there are no remote locaters that are already manufactured by any industries. This is the reason, why this project is being developed. The user just need to push a button and the missing item will be located.

This product will benefit the people who have multiple items in their home such as keys, remote control, spectacles, walking sticks and mobile phone. People will no longer have to spend their times in searching their missing items.

The target users are people at the range of 10 year and above. They can users can easily used this remote locater easily because it is user-friendly and light. Other that, the cost for this remote locater is cheaper compared to the international products.

1.3 Scope of Work

This project separated into two parts which are hardware and software design. Microcontroller will be used as a behavior controller for this project. In this project PIC16F877A and PIC16F876A will be used for transmitter and receiver respectively.

In the hardware design, the board can be divided into two parts which are Control Unit and Remote Unit. Another major component is Radio Frequency Module manufactured by the Cytron Technologies. The frequency for this RF module has been chosen to be 315MHz.

For the preliminary design, both Control unit and Remote unit will be implemented on the breadboard.

1.4 Chapter Overview

The following chapter will be discussed about the design specification, overview of hardware and software and project implementation. Chapter 2 and Chapter 3 will be discussed about the hardware and software overview respectively. For Chapter 4, the project implementation will be further discussed using the hardware and software that has been discussed in Chapter 2 and Chapter 3. On this chapter, further discussion will be made on how the system is being developed. Chapter 5 will show the result for the overall project. In addition, Chapter 6 is the conclusion of the project and the possible further improvement for the project.

CHAPTER 2

HARDWARE OVERVIEW

2.1 Introduction

Embedded systems which are referred to the concept of microcontroller require running an application. Microcontroller with the integration of the input output device makes the embedded system more valuable. Input device means any hardware component that allows users to enter data and instructions into a computer. Besides, the output device is any hardware component that can display information to a user. Examples of input devices are keypad, push buttons and sensors. While LED, LCD, LCD and buzzer are the examples of the output devices.

2.2 Microcontroller

2.2.1 Introduction

Microcontroller is an embedded system. Embedded system is one computer system on a single-circuit designed to perform one or few functions [1] [2]. Some input output devices can be attached to the microcontroller such as sensor, motor, LCD, keypad, solenoid, push buttons and etc. This project used PIC's family will be used. PIC stands for Peripheral Interface Controller. PIC is a family of Harvard architecture microcontrollers manufactured by Microchip Technology [3]. Nowadays, PIC is the advanced version of PIC1640. This microcontroller have inbuilt 8K bytes of flash ROM, 368 bytes of RAM, 256 bytes of EEPROM data memory and 15 vectored interrupts[4].

In this project, PIC16F876A and PIC16F877A will be used for receivers and transmitter respectively. PIC16F876A and PIC16F877A is the modern version of PIC16F876 and PIC16F877. These types of PIC is useful because it only can update the bootloaders and program code from a serial port without removing PIC from the circuit and put it in the programmer[5].

2.2.2 PIC16F876A

2.2.2.1 Pin description

Table 2.1, Table 2.2 and Table 2.3 are the function of PORTA, PORTB and PORTC respectively. PORT is the pin where the input and output device will be attached. Figure 2.1 shows the pin diagram for PIC16F876A.

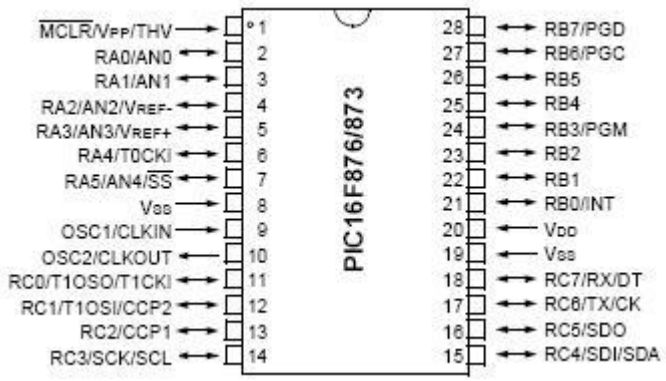


Figure 2.1: Pin diagram for PIC16F876A

Table 2.1: PORTA Function

Name	Bit	Buffer	Function
RA0/AN0	bit0	TTL	Input/output or analog input.
RA1/AN1	bit1	TTL	Input/output or analog input.
RA2/AN2	bit2	TTL	Input/output or analog input.
RA3/AN3/VREF	bit3	TTL	Input/output or analog input or VREF.
RA4/T0CKI	bit4	ST	Input/output or external clock input for Timer0. Output is open drain type.
RA5/SS/AN4	bit5	ST	Input/output or slave select input for synchronous serial port or analog input.

Legend: TTL: TTL input ST: Schmitt Trigger

Table 2.2: PORTB Function

Name	Bit	Buffer	Function
RB0	bit0	TTL/ $ST_{(1)}$	Input/output pin or external interrupt input. Internal software programmable weak pull-up.
RB1	bit1	TTL	Input/output pin. Internal software programmable weak pull-up.
RB2	bit2	TTL	Input/output pin. Internal software programmable weak pull-up.
RB3/ $PGM_{(3)}$	bit3	TTL	Input/output pin or programming pin in LVP mode. Internal software programmable weak pull-up.
RB4	bit4	TTL	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up.
RB5	bit5	TTL	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up.
RB6/ PGC	bit6	TTL/ $ST_{(2)}$	Input/output pin (with interrupt-on-change) or In-Circuit Debugger pin. Internal software programmable weak pull-up. Serial programming clock.
RB7/ PGD	bit7	TTL/ $ST_{(2)}$	Input/output pin (with interrupt-on-change) or In-Circuit Debugger pin. Internal software programmable weak pull-up. Serial programming data.

Legend: 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. Low Voltage ICSP Programming (LVP) is enabled by default, which disables the RB3 I/O function. LVP must be disabled to enable RB3 as an I/O pin and allow maximum compatibility to the other 28-pin and 40-pin mid-range devices.

Table 2.3: PORTC Function

Name	Bit	Buffer	Type Function
RC0/ T1OSO/ T1CKI	bit0	ST	Input/output port pin or Timer1 oscillator output/Timer1 clock input.
RC1/ T1OSI/ CCP2	bit1	ST	Input/output port pin or Timer1 oscillator input or Capture2 input/Compare2 output/PWM2 output.
RC2/ CCP1	bit2	ST	Input/output port pin or Capture1 input/Compare1 output/PWM1 output.
RC3/ SCK/SCL	bit3	ST	RC3 can also be the synchronous serial clock for both SPI and I2C modes.
RC4/ SDI/SDA	bit4	ST	RC4 can also be the SPI Data In (SPI mode) or data I/O (I^2C mode).
RC5/ SDO	bit5	ST	Input/output port pin or Synchronous Serial Port data output.
RC6/ TX/CK	bit6	ST	Input/output port pin or USART Asynchronous Transmit or Synchronous Clock.
RC7/ RX/DT	bit7	ST	Input/output port pin or USART Asynchronous Receive or Synchronous Data.

Legend: ST = Schmitt Trigger input

2.2.3 PIC16F877A

2.2.3.1 Pin description

Table 2.4 and Table 2.5 show the function of PORTD and PORTE while the other PORTs are the same with the PIC16F876A. Because of this advantage, it is used as the controller at the Control Unit. Figure 2.2 shows the pin diagram for PIC16F877A.

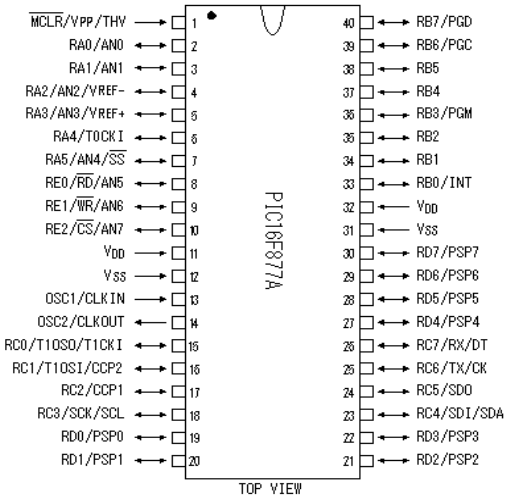


Figure 2.2: Pin diagram for PIC16F877A

Table 2.4: PORTD Function

Name	Bit	Buffer	Type Function
RD0/PSP0	bit0	ST/TTL(1)	Input/output port pin or parallel slave port bit0
RD1/PSP1	bit1	ST/TTL(1)	Input/output port pin or parallel slave port bit1.
RD2/PSP2	bit2	ST/TTL(1)	Input/output port pin or parallel slave port bit2.
RD3/PSP3	bit3	ST/TTL(1)	Input/output port pin or parallel slave port bit3.
RD4/PSP4	bit4	ST/TTL(1)	Input/output port pin or parallel slave port bit4.
RD5/PSP5	bit5	ST/TTL(1)	Input/output port pin or parallel slave port bit5.
RD6/PSP6	bit6	ST/TTL(1)	Input/output port pin or parallel slave port bit6.
RD7/PSP7	bit7	ST/TTL(1)	Input/output port pin or parallel slave port bit7.

Legend: ST = Schmitt Trigger input, TTL = TTL input

Note

1. Input buffers are Schmitt Triggers when in I/O mode and TTL buffers when in Parallel Slave Port mode.

Table 2.5: PORTE Function

Name	Bit	Buffer Type	Function
RE0/ RD/ AN5	bit0	ST/TTL(1)	I/O port pin or read control input in Parallel Slave Port mode or analog input: (RD) 1 = Idle 0 = Read operation. Contents of PORTD register are output to PORTD I/O pins (if chip selected)
RE1/ WR/ AN6	bit1	ST/TTL(1)	I/O port pin or write control input in Parallel Slave Port mode or analog input: (WR) 1 = Idle 0 = Write operation. Value of PORTD I/O pins is latched into PORTD register (if chip selected)
RE2/ CS/ AN7	bit2	ST/TTL(1)	I/O port pin or chip select control input in Parallel Slave Port mode or analog input: CS 1 = Device is not selected 0 = Device is selected

2.2.4 Clock generator

There are 4 different methods of clocking the PIC microcontrollers. The different options are designed based on different requirement such as cost, speed and accuracy [8]. Four clock/ oscillator are:

1. RC oscillator (resistor/capacitor)
2. XT oscillator (crystal/ceramic resonator)
3. HS oscillator (high sped crystal/ceramic resonator)
4. LP oscillator (low power crystal)

Table 2.6: Capacitor selection for crystal oscillator [4]

Osc Type	Crystal Freq.	Cap. Range C1	Cap. Range C2
LP	32 kHz	33 pF	33 pF
	200 kHz	15 pF	15 pF
XT	200 kHz	47-68 pF	47-68 pF
	1 MHz	15 pF	15 pF
	4 MHz	15 pF	15 pF
HS	4 MHz	15 pF	15 pF
	8 MHz	15-33 pF	15-33 pF
	20 MHz	15-33 pF	15-33 pF

In this project, HS oscillator has been used because the operating frequency range is up to 20MHz [6]. Figure 2.3 shows the connection of the crystal oscillator with the two capacitors. Higher capacitors help to assure to increase stability of oscillator operation and start-up [8] [4]. The use of crystal oscillator is, it will oscillate at a fixed frequency, with an accuracy around 50ppm (parts per million) [9]. This will allow the hardware timer to measure the exact intervals and accurate output signal will be produced.

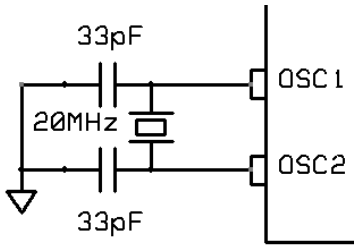


Figure 2.3: Oscillator connection

2.2.5 Reset switch

Reset acts as a “behavior controller”. Reset is used in order to put the microcontroller into the known condition. It means reset can prevent the microcontroller function into undesirable condition. Other than that, reset also can be used as an interrupt in program execution. In order to bring the microcontroller into a proper function, reset button must be pushed and it will bring all the registers into a starting positions.

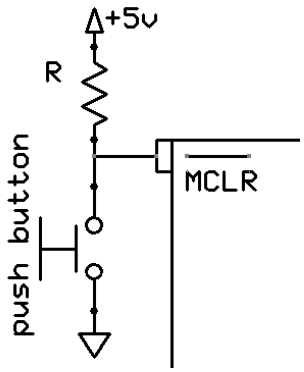


Figure 2.4: RESET pin connection

Figure 2.4 shows the connection at MCLR pin. To prevent the MCLR from bringing logical zero, MCLR must be connected into with resistor to the V_{dd} with range 5kΩ-10kΩ. This resistor called as a pull up resistor.

2.2.6 USART configuration

2.2.6.1 Introduction

The PIC microcontroller's Universal Synchronous Asynchronous Receiver Transmitter (USART) allows interfacing with serial devices such as RS-232 or synchronous serial device because PIC will provide the clock or having an external clock drive rate. The USART module is the best to asynchronous serial data transmission [6]. USART can be configured asynchronous full-duplex device, as a synchronous half-duplex master, or as a synchronous half-duplex slave. Asynchronous mode is used mostly in communication between analog-to-digital and digital-to-analog for serial EEPROM interfacing.

Five registers is used in USART module which are RCSTA (Receive Status and Control Register), TXREG (Read/Write Transmit Buffer), RCREG, TXSTA (Transmit Status and Control Register) and SPBRG (Baud Rate Generator Register).

2.2.6.2 The USART Baud Rate Generator

The baud rate generator is used in both synchronous and asynchronous mode in USART. In the clock generator circuit, the SPBRG register is used as a comparison value for the counter. When the counter is equal to the SPBRG register's value, a clock 'tick' output is made, and the counter is reset.

For asynchronous operation, the data speed can be known by the formula

BRGH = 0 (Low Speed) Baud rate $= \frac{f_{OSC}}{64([SPBRG]+1)}$ (2.1)

BRGH = 1(High Speed) Baud rate $= \frac{f_{OSC}}{16([SPBRG]+1)}$ (2.2)

For asynchronous operation, the data speed can be known by the formula

BRGH = don't care Baud rate $= \frac{f_{OSC}}{4([SPBRG]+1)}$ (2.3)

For baud rate selection, refer to *Appendix **C***

2.2.6.3 USART Asynchronous mode

In asynchronous mode, standard non return to zero (NRZ) formats is used. Non return to zero is a binary code which 1's represented by a positive voltage and 0's represented by negative voltage with no other neutral or rest condition.

Non-return to zero encoding is commonly used in slow speed communications interfaces for both synchronous and asynchronous. Figure 2.5 shows the packet of NRZ data which consists of start bit, data and stop bit. If nine (9) bits are transmitted, parity bit will be included before stop bit.

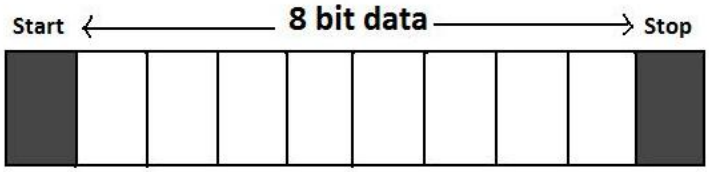


Figure 2.5: Asynchronous NRZ packet

2.2.6.4 USART Asynchronous Transmitter

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D
bit 7							bit 0

bit 7 **CSRC**: Clock Source Select bit
Asynchronous mode:
Don't care
Synchronous mode:
1 = Master mode (clock generated internally from BRG)
0 = Slave mode (clock from external source)

bit 6 **TX9**: 9-bit Transmit Enable bit
1 = Selects 9-bit transmission
0 = Selects 8-bit transmission

bit 5 **TXEN**: Transmit Enable bit
1 = Transmit enabled
0 = Transmit disabled

Note: SREN/CREN overrides TXEN in SYNC mode.

bit 4 **SYNC**: USART Mode Select bit
1 = Synchronous mode
0 = Asynchronous mode

bit 3 **Unimplemented**: Read as '0'

bit 2 **BRGH**: High Baud Rate Select bit
Asynchronous mode:
1 = High speed
0 = Low speed
Synchronous mode:
Unused in this mode

bit 1 **TRMT**: Transmit Shift Register Status bit
1 = TSR empty
0 = TSR full

bit 0 **TX9D**: 9th bit of Transmit Data, can be parity bit

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

Figure 2.6: Bitmap of the TXSTA register

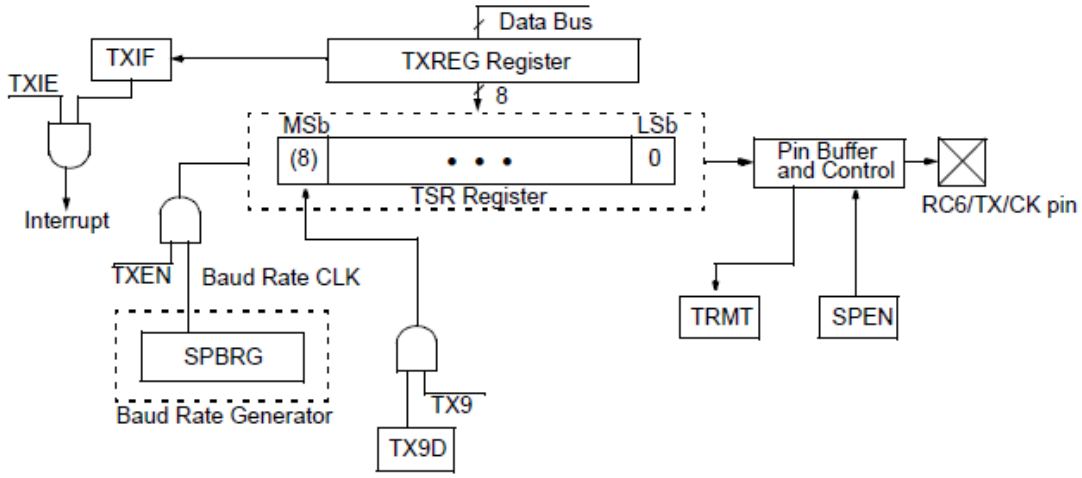


Figure 2.7: USART transmit block diagram

Figure 2.7 shows the block diagram for USART transmitter. The heart of the transmitter is Transmit (serial) Shift Register (TSR). The TSR obtain the data from the TXREG. The data is loaded to the TXREG by the software. The TSR will not start the loading process until the Stop bit has not been transmitted from the previous load. Once the Stop bit has been transmitted, TSR will be loaded by new data from TXREG. Once TXREG is empty, flag bit in TXIF is set. TRMT is set when the TSR register is empty. Transmission is enabled by setting enable bit for TXEN. Clearing enable bit TXEN during transmission will caused the transmission to be aborted and will reset the transmitter. As the result, RC6/TX/CK pin will result to high-impedance.

2.2.6.5 USART Asynchronous Receiver

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

bit 7 **SPEN**: Serial Port Enable bit
1 = Serial port enabled (configures RC7/RX/DT and RC6/TX/CK pins as serial port pins)
0 = Serial port disabled

bit 6 **RX9**: 9-bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5 **SREN**: Single Receive Enable bit
Asynchronous mode:
Don't care
Synchronous mode - master:
1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.
Synchronous mode - slave:
Don't care

bit 4 **CREN**: Continuous Receive Enable bit
Asynchronous mode:
1 = Enables continuous receive
0 = Disables continuous receive
Synchronous mode:
1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)
0 = Disables continuous receive

bit 3 **ADDEN**: Address Detect Enable bit
Asynchronous mode 9-bit (RX9 = 1):
1 = Enables address detection, enables interrupt and load of the receive buffer when RSR<8> is set
0 = Disables address detection, all bytes are received, and ninth bit can be used as parity bit

bit 2 **FERR**: Framing Error bit
1 = Framing error (can be updated by reading RCREG register and receive next valid byte)
0 = No framing error

bit 1 **OERR**: Overrun Error bit
1 = Overrun error (can be cleared by clearing bit CREN)
0 = No overrun error

bit 0 **RX9D**: 9th bit of Received Data (can be parity bit, but must be calculated by user firmware)

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

Figure 2.8: Bitmap of the RCSTA register

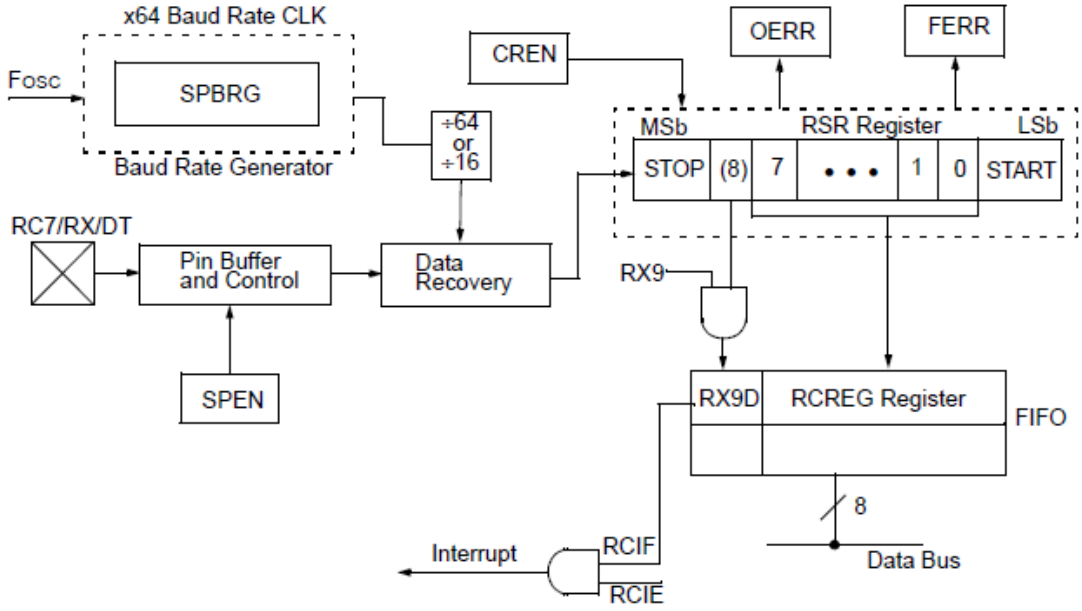


Figure 2.9: USART receiver block diagram

The data will be received by RC7/RX/DT pin and drives the data recovery block. The data recovery block is actually a high-speed shifter, operating at x16 times the baud rate. Reception is enabled by setting bit CREN. The heart of receiver is Receiver (Serial) Shift Register (RSR). The data receives by RC7/R/DT pin will be sampled three times by a majority detect to determine if a high or low level is present at the RX pin. After sampling the Stop bit, the data in the RSR will be transferred to the RCREG register. The RCIF will be set if the transfer is finished. Flag RCIF is a read-only bit which is cleared by the hardware. It is cleared after the data in the RCREG has been read and empty.

RCREG is a double-buffered register which it can stores two bytes of data once in a time. After the third byte is received, one byte in the RCREG will be transferred into RSR register. Flag bit in the Overrun Error bit (OERR) will be set if RCREG register is still full. Flag bit of OERR can be cleared by the software. This is done by resetting the flag bit of CREN. After OERR is set, no further data will be received.

2.3 RF module

2.3.1 Introduction

Radio frequency (RF) transmitters are widely used in radio frequency communications systems. With the increasing availability of efficient, low cost electronic modules, mobile communication systems are becoming more and more widespread. For this project, RF module manufactured by Cytron Technologies is used. This module comes ready to plug into application and only need simple interfaces circuits. This project used 315MHz of frequency. According to ISM band, 315MHz is in licensed free frequency range. ISM band means industrial, scientific and medical radio bands which it is reserved for international uses of RF electromagnetic fields in that area other than communication.[13]

This RF module only cost RM15 and RM25 for transmitter and receiver respectively. I am using ready-made RF module because of the complexity of the circuit. Since I am just using it, buying from nearby shop is enough.

2.3.2 RF-RX-315

2.3.2.2 Specification

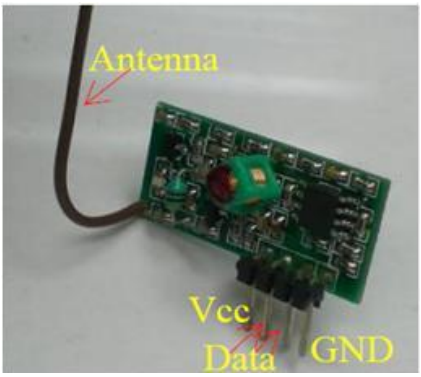


Figure 2.10: 315MHz Receiver Module

This low cost RF Receiver can be used to receive RF signal from any 315MHz transmitter. Super regeneration design ensure sensitive to weak signal. This other advantages of this module is, easy to integrate and low power consumption (4mA).

Table 2.7: RF-RX-315 specification

No	Specifications	RF Receiver
1	Operating Voltage	5.0V ± 0.5V
2	Operating Current	≤5.5mA @5.0V
3	Operating Principle	Monolithic super heterodyne receiving
4	Modulation	OOK/ASK
5	Frequency	315MHz
6	Bandwidth	2MHz
7	Sensitivity	-100dBm
8	Rate	< 9.6Kbps (315MHz @-95dBm)
9	Data Output	TTL
10	Antenna Length	24cm

2.3.3 RF-TX-315

2.3.3.1 Specification

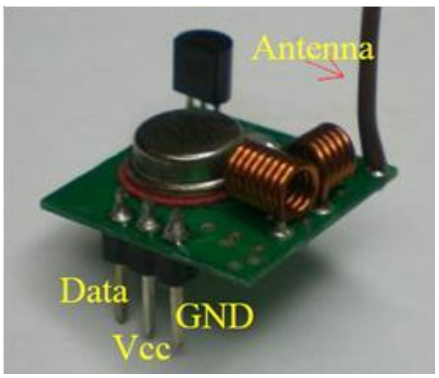


Figure 2.11: 315MHz Transmitter Module

This RF Transmitter Modules is very small in dimension and has a wide operating voltage range (3V-12V). The low cost RF Transmitter can be used to transmit signal up to 100 meters (the antenna design, working environment and supply voltage will seriously impact the effective distance). It is good for short distance.

Table 2.8: RF-TX-315 specification

No	Specifications	RF Transmitter
1	Operating Voltage	3V to 12 V
2	Operating Current	Max ≤ 40mA (12V), Min ≤ 9mA (3V)
3	Oscillator	SAW (Surface Acoustic Wave) oscillator
4	. Frequency	315MHz
5	. Frequency error	±150kHz(max)
6	Modulation	ASK/OOK
7	Transfer Rate	≤10Kbps
8	Transmitting power	25mW (315MHz@12V)
9	Antenna Length	24cm

2.4 Speaker

Figure 2.12: Piezo-buzzer, 12vDC

Piezo buzzer is used to indicate where the receiver is. If the address transmitted by the transmitter is matched with the address in the receiver, a loud distinctive sound will be produced by this buzzer.

Table 2.9: Piezo-buzzer specification

No	Specification	Description
1	Supply Voltage	1.5 V
2	Max Supply Voltage DC	20V dc
3	Min Supply Voltage DC	1.5V dc
4	Self Resonant Frequency	2.8kHz
5	Sound Level SPL	96dB
6	Current Consumption	10mA
7	External Depth	16mm
8	External Diameter	50mm
9	Weight	15g

2.5 Voltage regulator

A voltage regulator is an electrical regulator designed to automatically maintain a constant voltage level. Since a power supply frequently produces raw current that would otherwise damage one of the components in the circuit, voltage regulators is needed in order to regulate input voltage relatively close to a desired value.

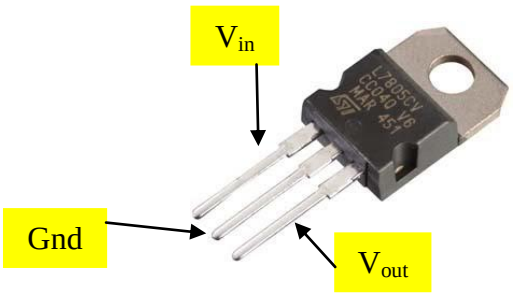


Figure 2.13: Voltage regulator

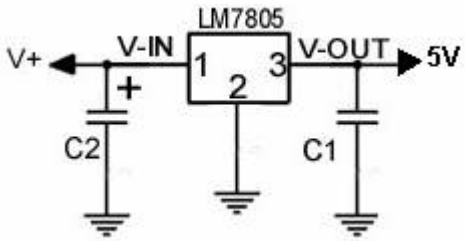


Figure 2.14: Typical connection

Figure 2.14 shows a connection for 3 terminals voltage regulator. The voltage range of power supply (V_{in}) should be between 7V and 15V. Higher input voltage will produce more heat at the voltage regulator. The purpose of connected capacitor C1 and C2 is use to stabilize the voltage input and output of the LM7805.

2.6 Liquid Crystal Display (LCD)

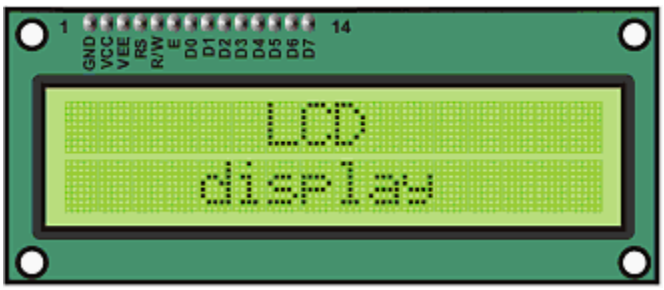


Figure 2.15: 2x20 LCD pin

This project will used LCD with 2x20 configurations. It has 16 pins. Table 2.10 shows the function for each pin.

Table 2.10: LCD Configuration

Pin No.	Name	Description
1	Vss	Power supply (GND)
2	Vcc	Power supply (+5V)
3	D7	Data line 7
4	D6	Data line 6
5	D5	Data line 5
6	D4	Data line 4
7	D3	Data line 3
8	D2	Data line 2
9	D1	Data line 1
10	D0	Data line 0
11	En	Enable signal for row 0 and 1
12	RW	0 = Write to LCD module 1 = Read from LCD module
13	RS	0 = Instruction input 1 = Data input
14	V_{EE}	Contrast adjust
15	V_{CC}	Power supply (+5V)
16	V_{SS}	Power supply (GND)

2.7 USB ICSP PIC Programmer

Figure 2.16: USB ICSP PIC Programmer and 8-40 Pin ZIF Socket

UIC00A offers low cost PIC USB programmer yet reliable and user friendly. It is designed to program popular Flash PIC microcontroller which includes PIC12F, PIC16F and PIC18F family. It supports on board programming which eliminate the frustration of plug-in and plug-out of PIC microcontroller. This also allow user to quickly program and debug the source code while the target PIC is on the development board.

CHAPTER 3

SOFTWARE OVERVIEW

3.1 Introduction

In this chapter, program the PIC is a must in order to make the system work as what had been planned. Program the PIC is another tough task in develop an embedded system. To program it, we must use assembler software. Assembler software is software that converts the instructions into a pattern of bits that the PIC can understand and do its job [12]. This pattern bits is called machine language. This software will also generate many files but one of the important file is .hex file. This file will be used by PIC to do its job.

To load this .hex file, programmer and emulator are needed. Programmer function as to a hardware device that configures programmable non-volatile circuits such as EPROMs, EEPROMs, Flashs, PALs, FPGAs or programmable logic circuits. Meanwhile, a simulator is a tool that helps the assembler to load the .hex file into the PIC.

3.2 PIC Language

Language is the source of communication among human beings. Different countries have different languages. It is similar to the communication happens in the PIC. Language that is use must be understood by the PIC. There are two major types of programming languages which are low level languages and high level languages. Low level languages are further divided into machine language and assembly language. Examples of high level language are C, C++, MicroC and Basic. For this project, C language will be used in order to program the PIC.

3.2.1 Advantages of high level language

High level language enables people to write the programs in their own understanding. High level language actually is the symbolic language that is use English words and/or mathematical symbols rather than mnemonic codes. Each instruction in the high level language will be translated into machine language instructions. Below are the advantages of using high level language:

- User-friendly
- Similar to English with vocabulary of words and symbols
- Easier to learn.
- Require less time to write.
- Easier to maintain.
- Problem oriented rather than 'machine' based.
- Program written in a high-level language can be translated into many machine language and therefore can run on any computer for which there exists an appropriate translator.
- It is independent of the machine on which it is used. Programs developed in high level language can be run on any Computer

3.3 MPLAB IDE Assembler

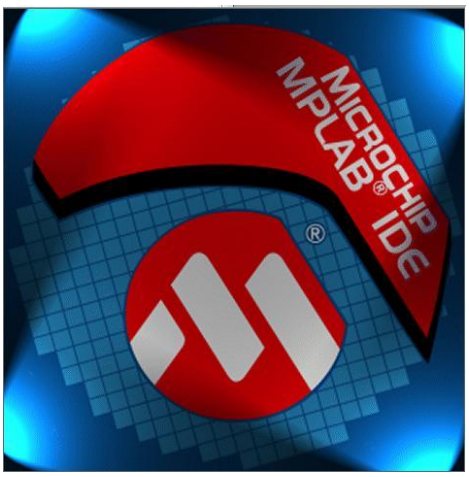


Figure 3.1: MPLAB IDE assembler

Assembler is a programming language that is part of the toolset used in embedded systems programming. It comes with its own distinct set of rules and techniques. It is essential to adopt and learn IDE (Integrated Development Environment) when developing programs. MPLAB IDE is an excellent tool for PIC microcontrollers, both for learners and professionals. In addition, it is easy to get and free. People can download MPLAB assembler for microchip website which is www.microchip.com. The main software tools and files created and used by MPLAB during the development process [7] are as Table 3.1.

Table 3.1: Components of MPLAB development system

Software tool	Tool function	Files produced or used	File description
Text editor	Used to create and modify source code text file	.asm/.c	Source code text file
Assembler	Generates machine code from source code	.hex	Executable machine code
	reports syntax errors	.err	Error massages
	generates list files	.lst	List files with source code and machine code
	generate symbol files	.cod	Symbol and debug information
Simulator	Allows program to be tested in software before downloading	.hex .cod	
Programmer	Downloads machine code to chip	.hex	

3.4 Creating and building a project

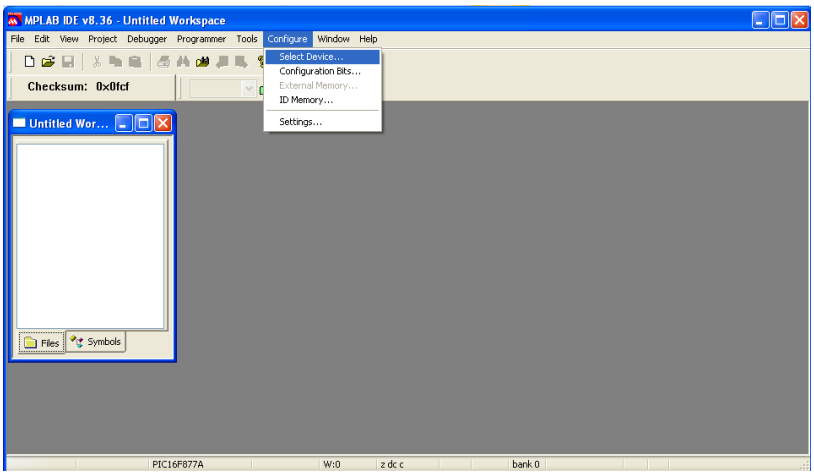


Figure 3.2: Select device

Open MPLAB IDE application. Firstly, we need to select the desired PIC by clicking Configure>Select device>choose your device from the list>Ok.

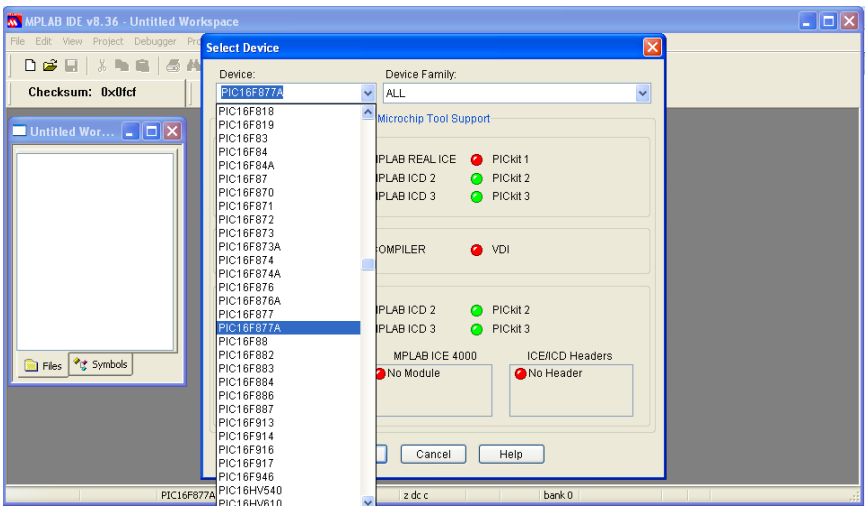


Figure 3.3: Choose the type of PIC

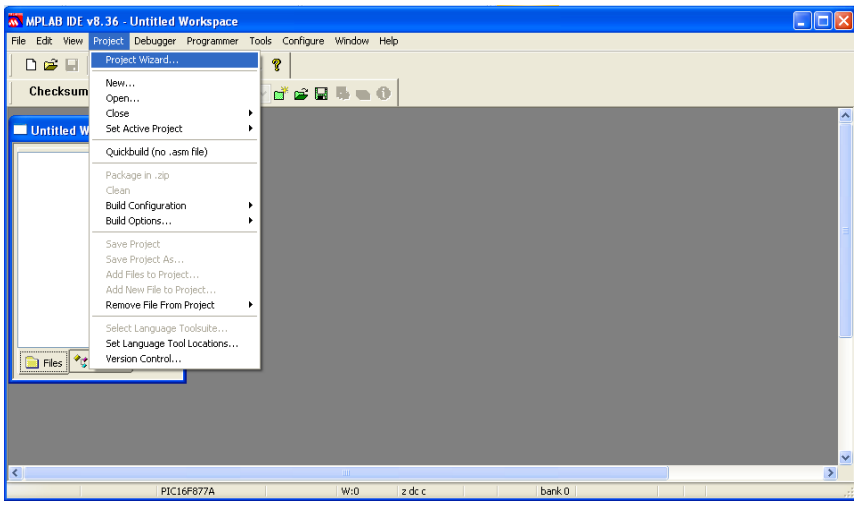


Figure 3.4: Project wizard

Before start typing the program, project wizard must be existed. Firs, click Project>Project Wizard. One project wizard interface will be pop out. Click Next>Next. In step two, user requires to select their active toolsuit. As in the project, C language is used. Hi-Tech Universal Toolsuite is select. Then click Next.

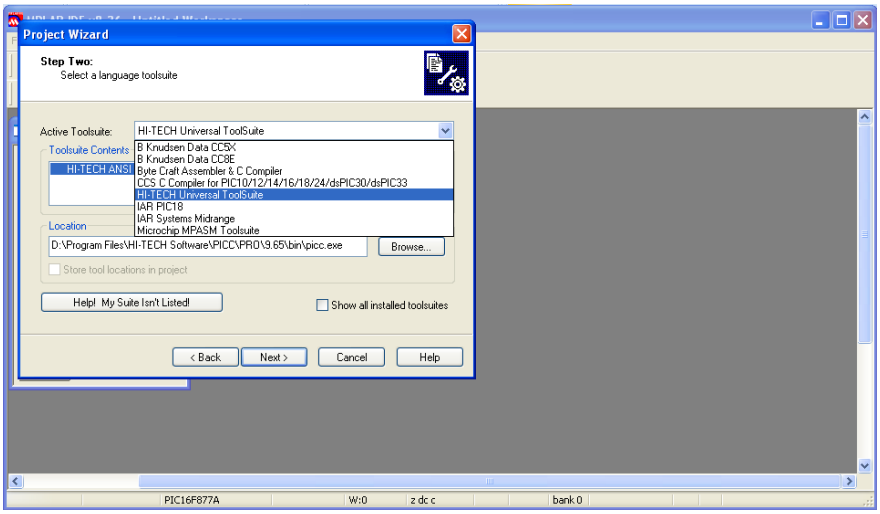


Figure 3.5: Select active toolsuit

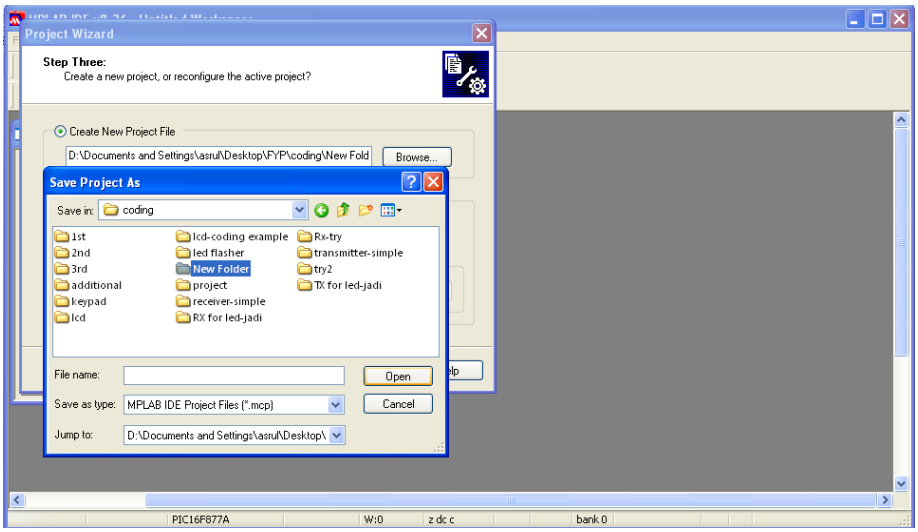


Figure 3.6: Create project path

After that, in step three, project path must be selected by clicking Browse. Lastly, click Next>Finish. To start typing the program, click New. A workspace will be pop out and writing the coding can be started.

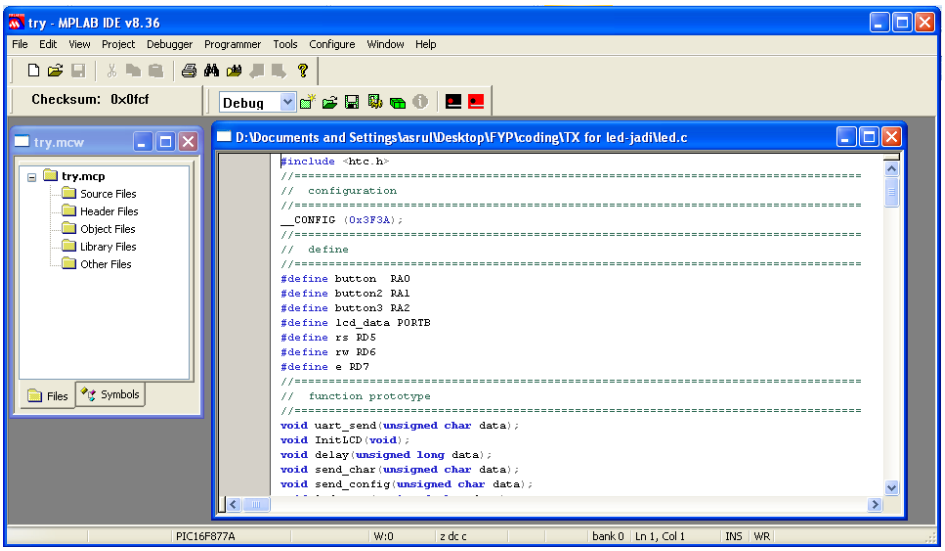


Figure 3.7: Workspace

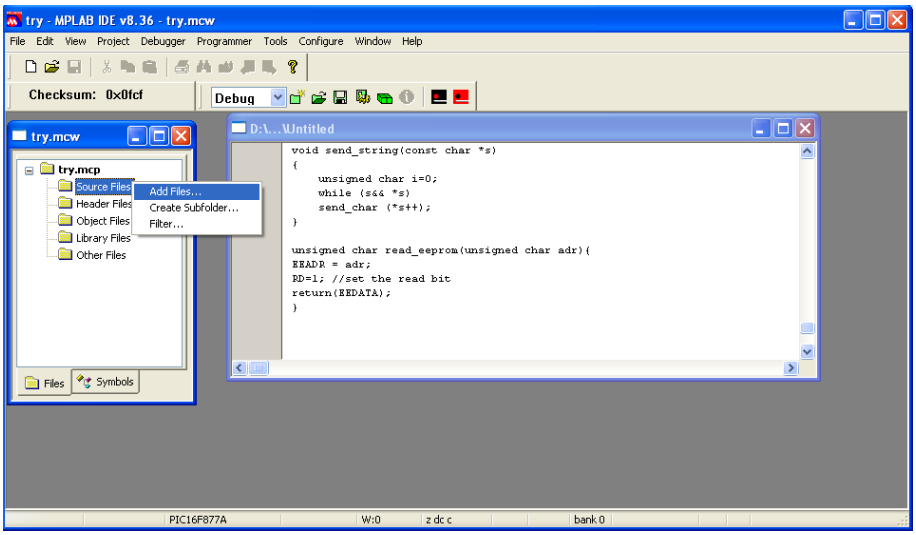


Figure 3.8: Include source file

After finish the writing, click Save as file *.c*. Before a coding can be build, a source file must be included. The source file is the file that was saved previously. See the following figure.

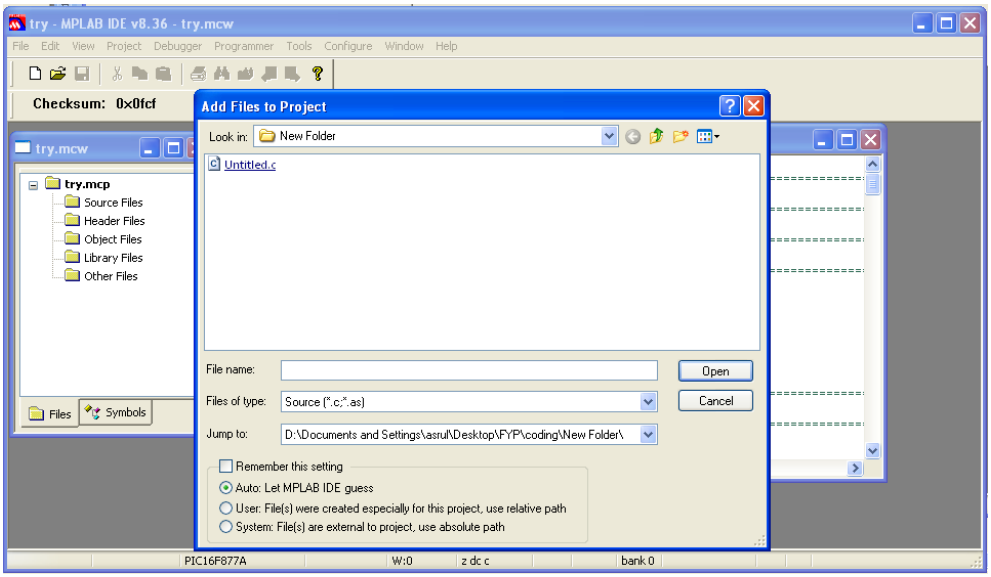


Figure 3.9: Add file to the project.

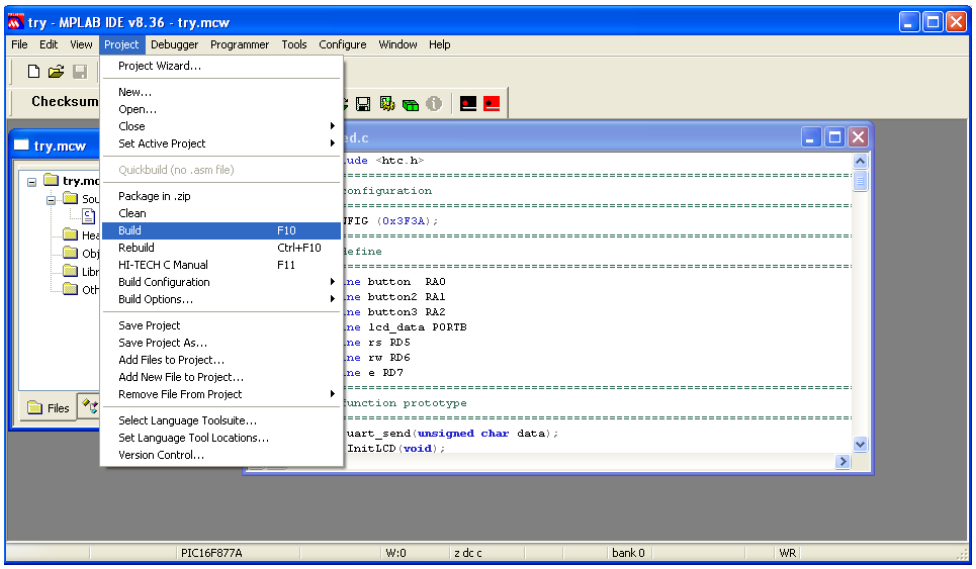


Figure 3.10: Build the project

The coding can be build by clicking Project>Build on the toolbar. The result can be seen in the Figure 3.11.

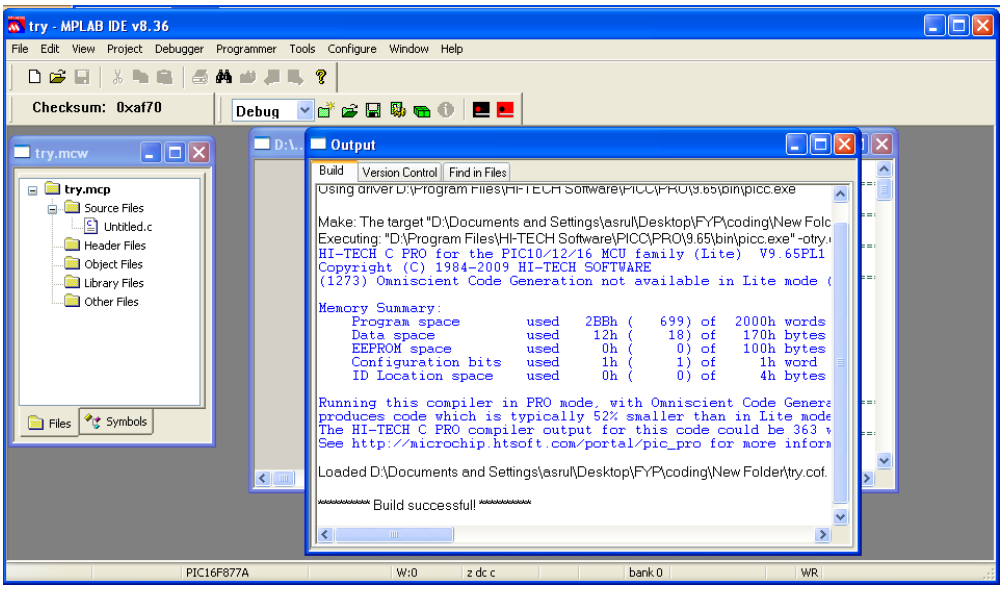


Figure 3.11: The build file

3.5 PICKIT 2 v2.55 emulator

Figure 3.12: PICKIT2 logo

PICkit 2 is a tool that is used to load .hex file into the microcontroller. PICkit 2 is manufactured by Microchip Technology. Figure 3.13 shows the interface of PICkit 2. It is a user-friendly tool because it is easy to learn it.

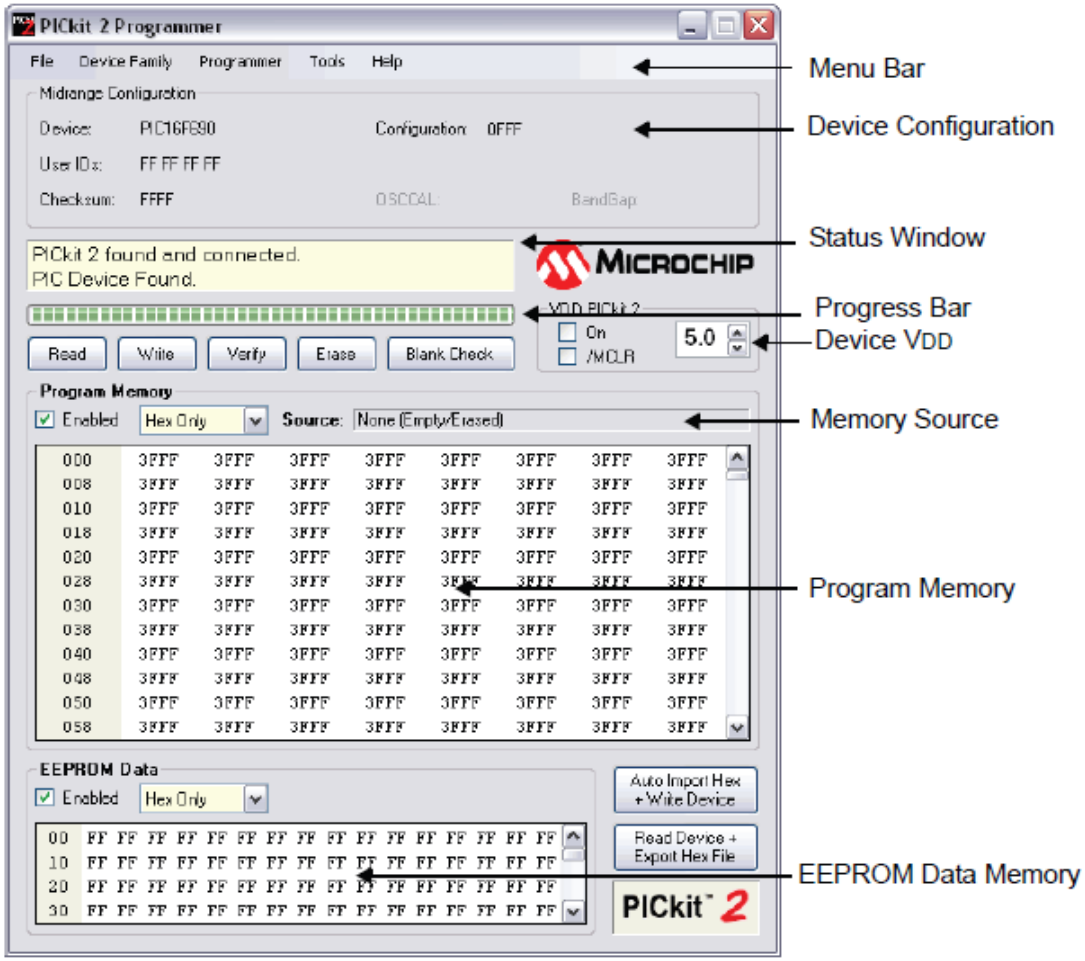


Figure 3.13: PICKIT2 interface

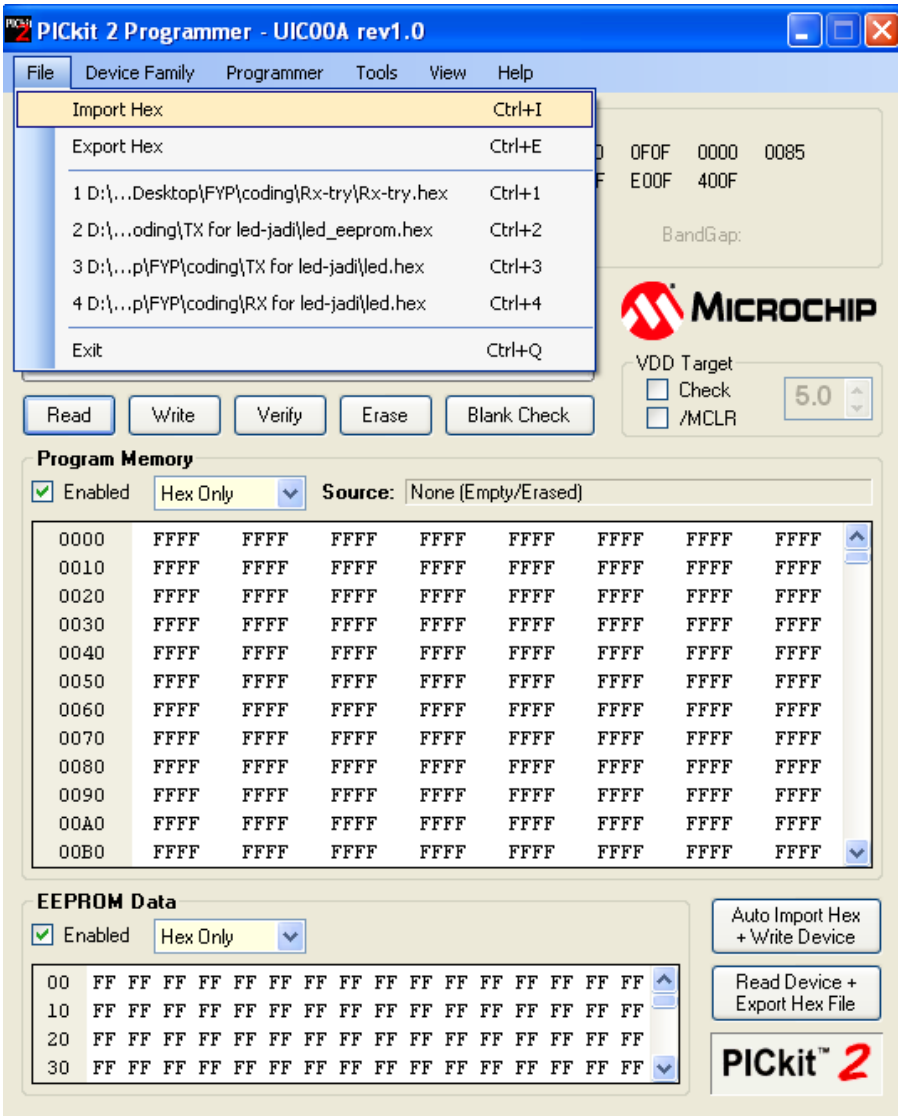


Figure 3.14: Load .hex file

To load the program, we need the .hex file. This file automatically exists after we build the project. Firstly, click File>Import Hex and then, choose .hex file for the project. After the .hex file is successfully loaded, click Write>Verify. If the user wants the previously .hex file to be deleted, click Erase and Blank Check to make sure that the file in PIC is totally erased.

3.6 Circuit drawing software

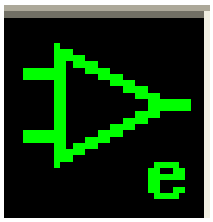


Figure 3.15: ExpressSCH

Circuit drawing tool is one of the important tool that is needed to design the electronic circuit. ExpressSCH is a freeware tool. It is easy to get on the internet. Other than that, it is easy to learn. The user does not need a long time to learn how to use it because it is a user-friendly tool.

CHAPTER 4

PROJECT IMPLEMANTATION

4.1 Introduction

In this chapter, further description is about system development. As stated earlier, this project consists of two parts, software and hardware. The hardware are consists of PIC16F87A and PIC16F877A, radio frequency module, LCD and piezo buzzer. The program for this remote locator is created in C language using MPLAB IDE.

4.2 Control unit

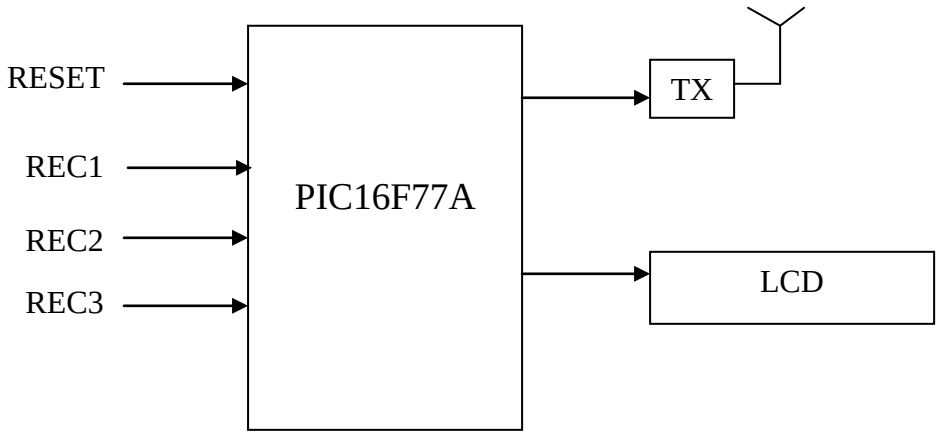


Figure 4.1: Control unit block diagram

Figure 4.1 shows the block diagram for the Control unit. The control unit contains a PIC16F877A, a transmitter, four push buttons which are RESET button and three alert buttons for the receiver and also the LCD.

4.2.1 Device function

Table 4.1: Hardware description used in Control Unit

No.	Hardware	Description
1	PIC16F877A	When the receiver button is pressed, the microcontroller in the Control Unit will sent the desired address to the respective remote unit. Other than that, PIC16F877A is stored the menu for the LCD display.
2	RF-RX-315	Transmit the address using radio frequency signal
3	LCD	Display the menu item
4	REC1 button	Provide the address for receiver 1 once the button is pressed.
5	REC2 button	Provide the address for receiver 2 once the button is pressed.
6	REC3 button	Provide the address for receiver 3 once the button is pressed.
7	RESET button	Reset the program to the starting point.

4.3 Remote unit

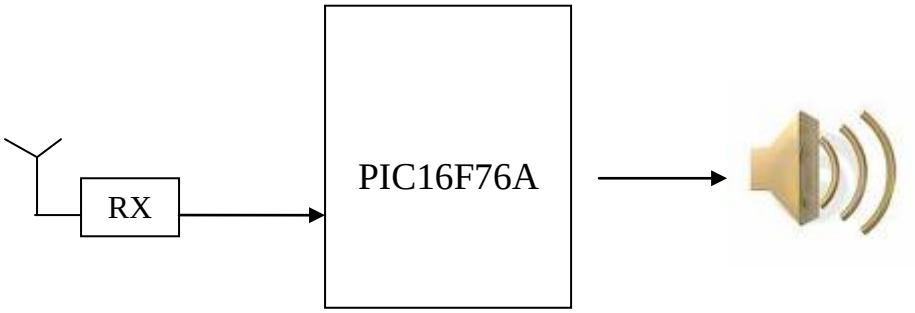


Figure 4.2: Remote unit block diagram

Figure 4.3 shows the block diagram for the Remote unit. The remote unit contains a PIC16F876A, a receiver, and a buzzer.

4.3.1 Device function

Table 4.2: Hardware description used in Remote Unit

No.	Hardware	Description
1	PIC16F876A	This microcontroller will receive the address and matched them with the existing address.
2	RF-RX-315	Received the address from the Control unit every time the receiver button in Control unit is pressed
3	Buzzer	Every time the correct addressed is matched, the buzzer will produce an alert sound

4.4 Process flowchart

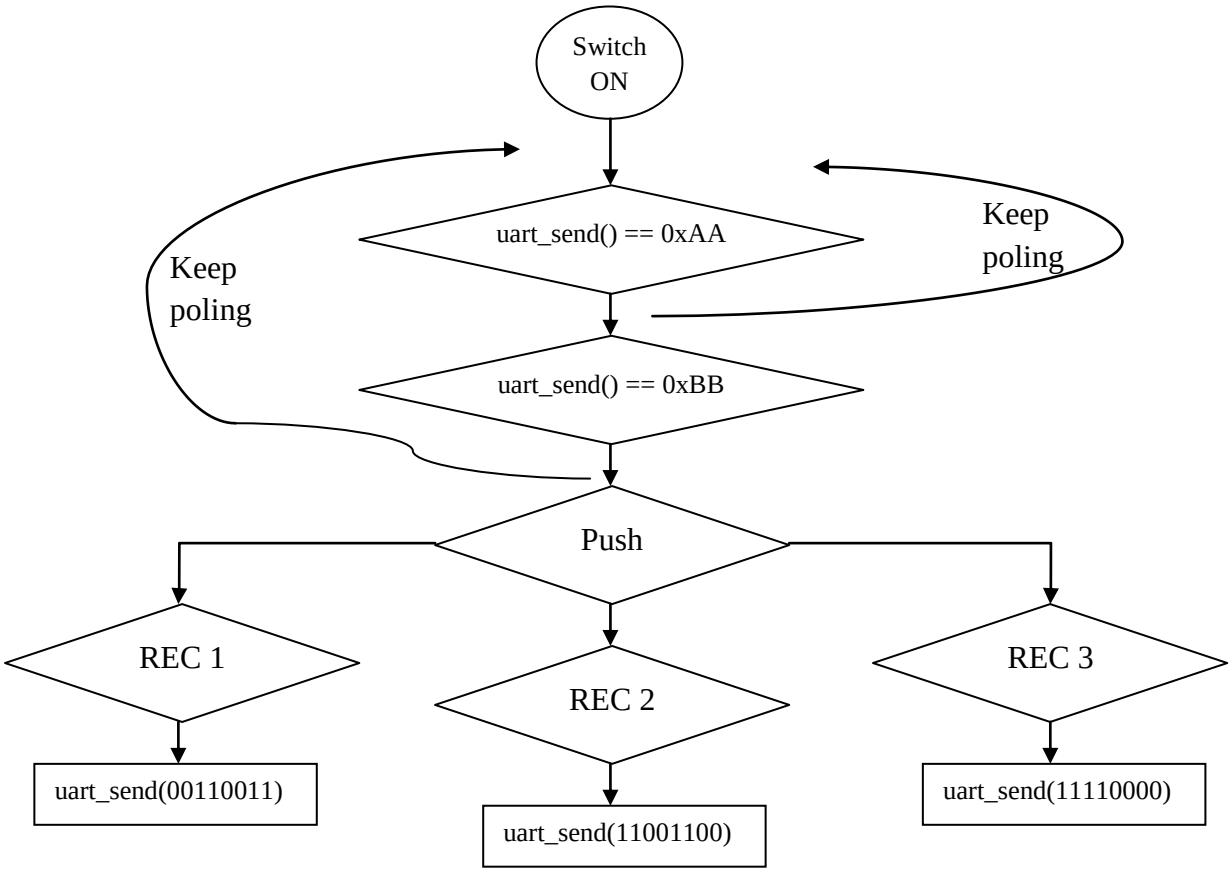


Figure 4.3: Process flowchart in Control Unit

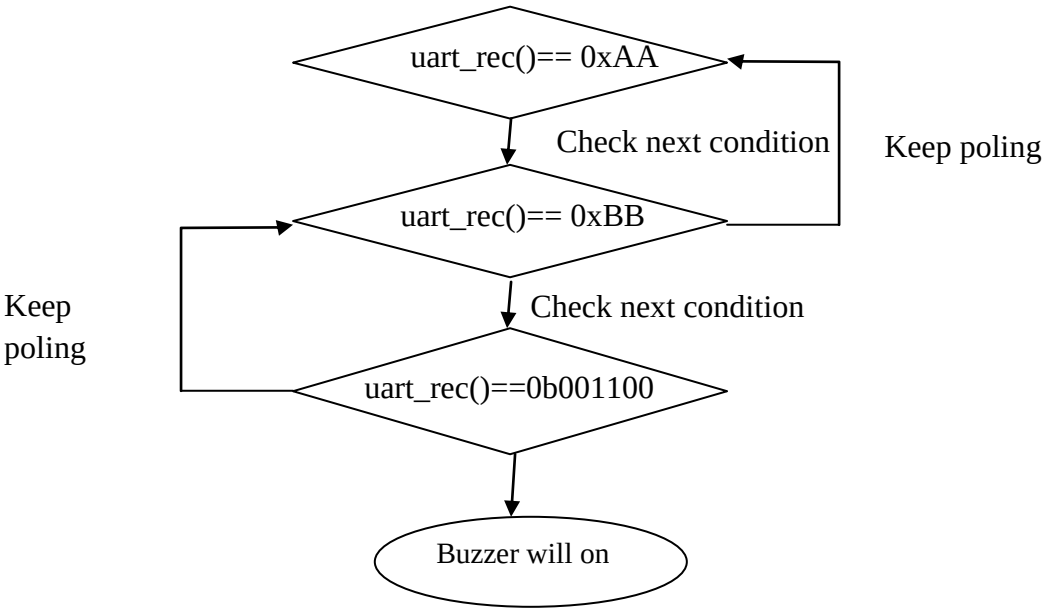


Figure 4.4: Process flowchart in Remote Unit

Figure 4.3 and Figure 4.4 shows the flowchart for overall processes in the Control Unit and Remote unit. Once the power on, the transmitter will automatically send the enable signals which are 0xAA and 0xBB. The enable signal will stop sending the data when the receiver button is being pushed. For example, if REC1 is being presses, address "00110011" will be transmitted. At the Remote unit, all the receiver will receives the address and microcontroller will differentiate which address is matches with them. When the address is matched, the loud distinctive sound will be produced by the buzzer.

4.5 Hardware setup

4.5.1 Power supply

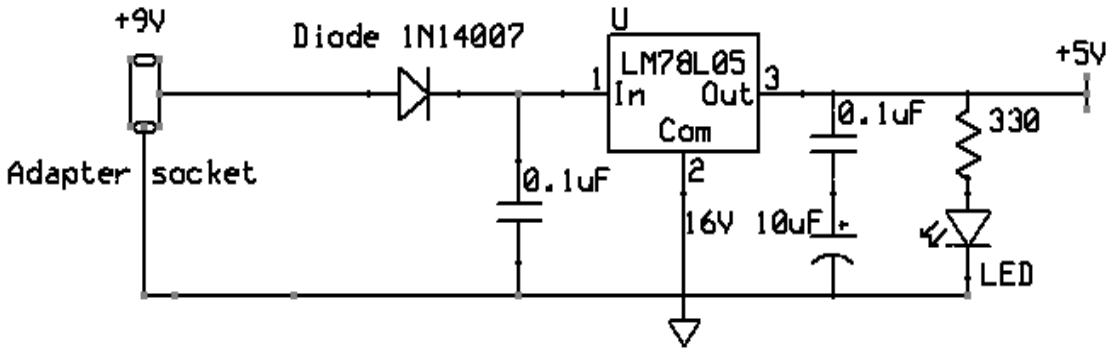


Figure 4.5 Power supply using AC-DC power supply

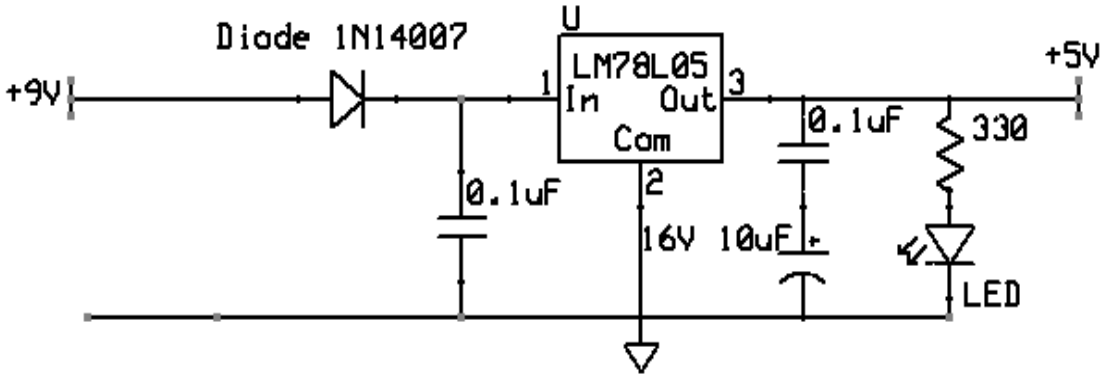


Figure 4.6: Power supply using battery

Figure 4.5 and Figure 4.6 shows the schematic of the power supply for the microcontroller. Figure 4.5 is the power supply for Control unit while Figure 4.6 is the power supply for the Remote unit. The input supply is +9V which is generated from the battery and the output is +5V. The function of LM7805 is to regulate the input voltage become +5V. The purpose of using diode 1N14007 is for circuit protection in case the polarity of the power source is incorrect. Capacitor (0.1uF) is use to stabilize the voltage input and output of the LM7805.

4.5.2 PIC interfacing

4.5.2.1 Interface PIC with RF- TX-315

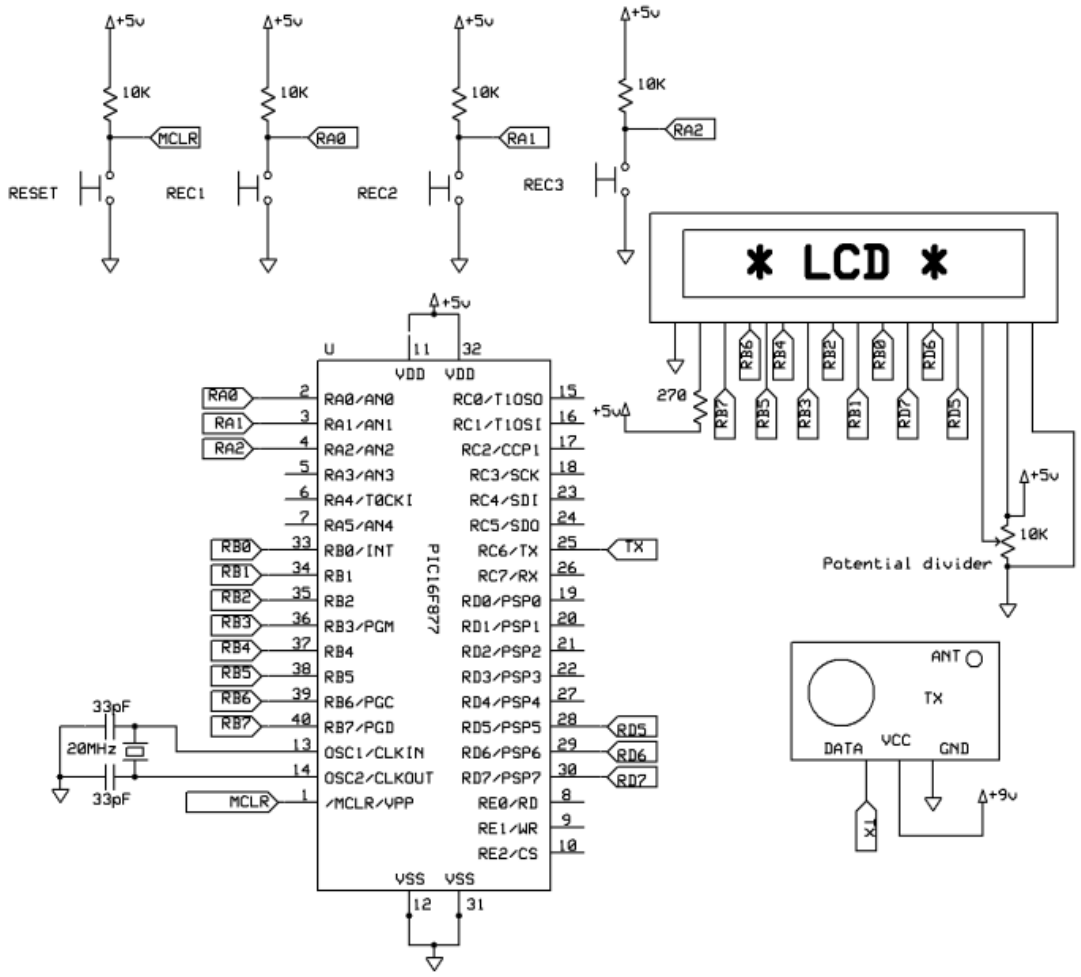


Figure 4.7: Control unit schematic diagram

Figure 4.7 shows how the microcontroller is being attached with the transmitter and other hardware.

4.5.2.2 Interface PIC with RF-RX-315

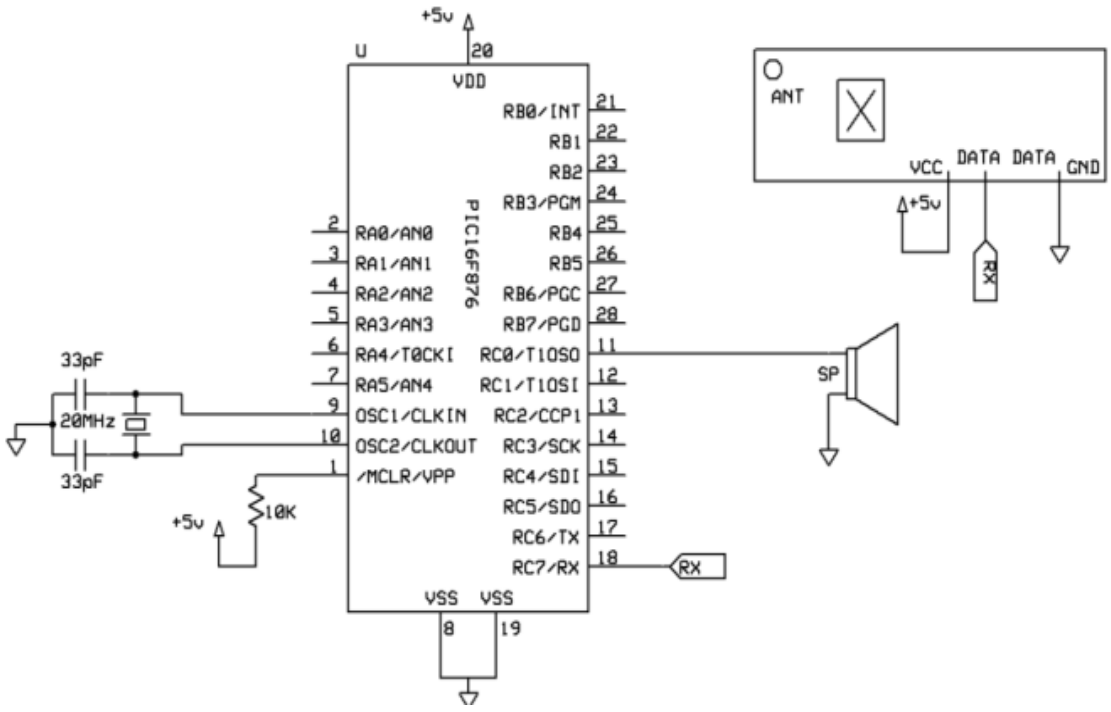


Figure 4.8: Remote unit schematic diagram

Figure 4.8 shows how the microcontroller is being attached with the receiver and other hardware.

4.6 Software function

4.6.1 Function for data transmitting

```
if(button==0)                                   1
{
    {
        no=0b00110011;                          2     //send the id number
        lcd_clr();
        lcd_goto(0);
        send_string("RECEIVER 1--> ON");
        lcd_goto(20);
        send_string("Searching >>>");
    }
        while(button==0)
            uart_send(no);                      //continuous send data
}                                               3

void uart_send(unsigned char data)
{
    while(TXIF==0);                 //only send the new data after the previous is sent
    TXREG=data;
}
```

Figure 4.9: Data transmission function

Figure 4.9 shows the function for data transmitting. Refer to this figure, once the user pressed the REC1, no. will set as address “00110011”. Circle three shows, while the button is still pushed, the transmitter will always send the data to the receiver. While the TXIF is equal to zero, the data will be transmitter only when the previous data already transmitted.

4.6.2 Function for data receiving

```
while(1)                        //infinity loop
{
    CREN=1;                                          1
    if(OERR==0)
    {
        if(uart_rec()==0xAA)                                  2
        {
            if(uart_rec()==0xBB)
            {
                if(uart_rec()==0b00110011)                          3
                buzzer=1;
                else
                    buzzer=0;
            }
        }
    }
    else
        CREN=0;
}

unsigned char uart_rec(void)     //receive uart value
{
    unsigned char rec_data;
    while(RCIF==0);              //wait for data
    rec_data = RCREG;
    return rec_data;             //return the received data
}
```

Figure 4.10: Data receiving function

Figure 4.10 shows how the data is going to receive. Firstly, CREN will be equal to one means sat data to be sent continuously. Other than that, OERR condition also will equal to zero means RCREG still do not full. Secondly, enable signal will be checked in order to eliminate noise. Thirdly, comparison will be made to the address. If the receiving address is equal to “00110011”, then the buzzer will

be on. Otherwise, it will be off. RCIF will be cleared by the software if the data in RCREG is empty and has been read.

4.6.3 Noise elimination

```
//=======================================================
//Send enable signal
//=======================================================
        no=0xAA;
        uart_send(no);
        no=0xBB;
        uart_send(no);
```

Figure 4.11: Data receiving function

Figure 4.11 shows the coding for noise elimination. Noise means the disturbance signal or unwanted signal occur in the system. To eliminate this noise, the enable signal will be sent by the transmitter continuously. At the receiver part, microcontroller will detect the address 0xAA first and 0xBB for the second address. If it is in sequence, then the address received will be compared by their address.

4.7 Preliminary work

Before starts doing the real project, the previous project is doing again. The purpose is to familiarize how the hardware is being connected. Other than that, I want to build a program for this design again in the C language. Figure 4.12 shows the transmitter unit and Figure 4.13 is the receiver part.

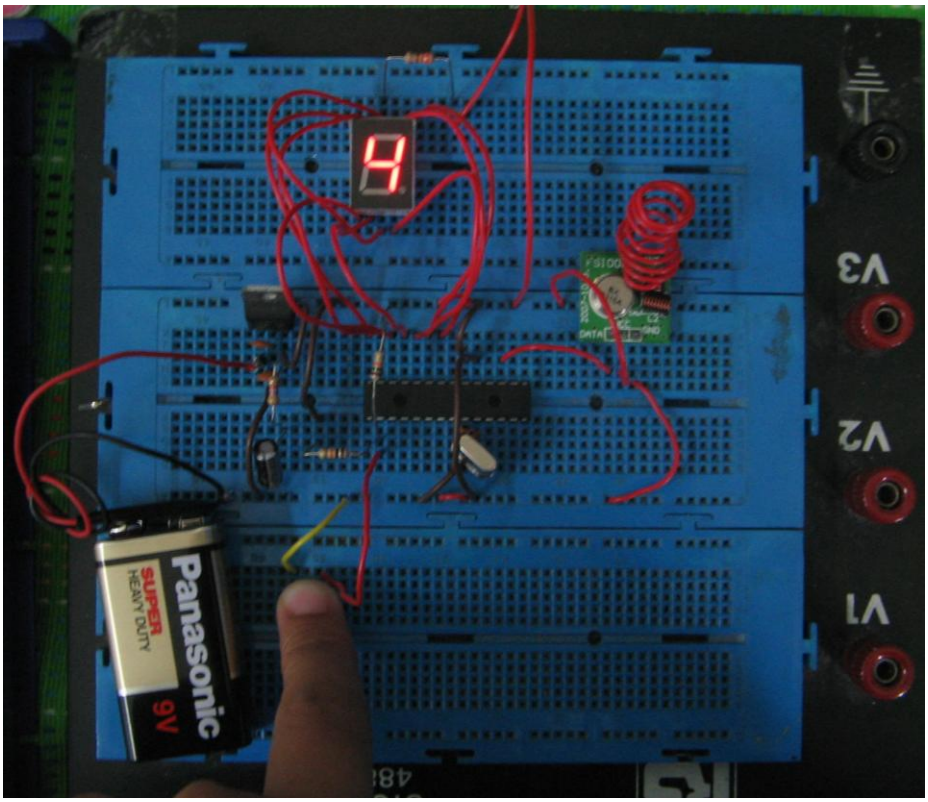


Figure 4.12: Transmitter unit

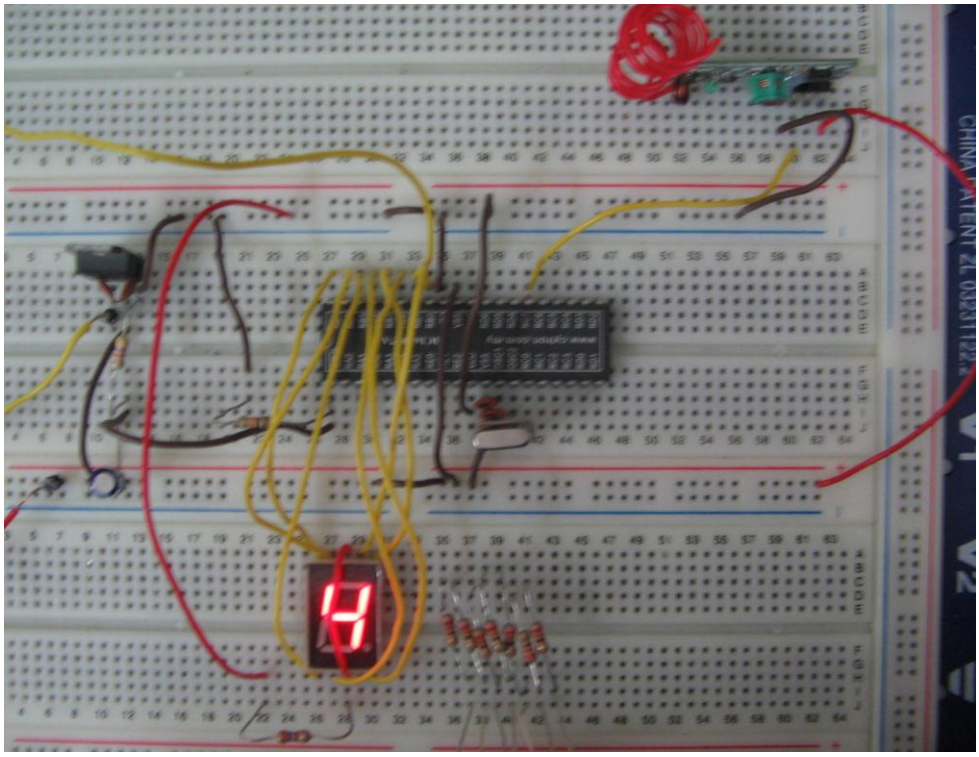


Figure 4.13: Receiver unit

When the user pushed the button, the number will be increased and being transmits by the transmitter. The correctness can be checked if the number at the transmitter unit is the same with the number at the receiver part.

CHAPTER 5

RESULT

5.1 Introduction

This chapter will be discussed about the results obtained from the project. There are two types of outcomes which are the hardware and the experiment that has been done for test its capability.

5.2 Hardware part

5.2.1 Control unit board

Figure 5.1 shows the main board of this project. The main board consists of PIC16F877A, voltage regulator, RF-TX-315, voltage regulator, LCD and push buttons.

Figure 5.1: Control unit board

5.2.2 Remote unit board

Figure 5.2 shows the sub board of this project which are the Remote unit. The figure only shows one of the receiver units. The sub board consists of PIC16F876A, voltage regulator, RF-RX-315, voltage regulator and piezo buzzer.

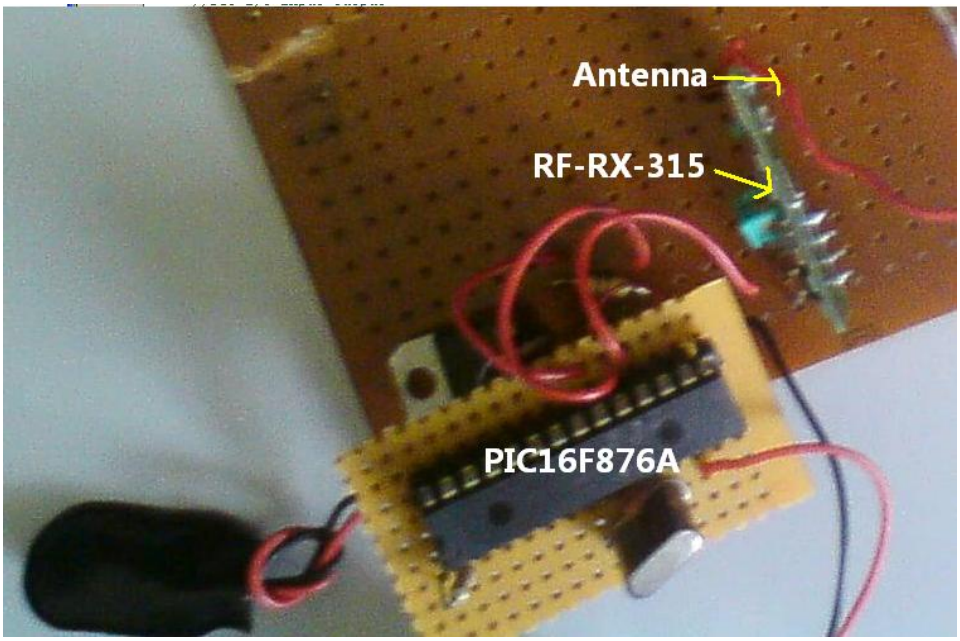


Figure 5.2: Remote unit board

5.3 Remote locator capability

Table 5.1: The capability of the remote locator

Condition	Outcome
With antenna in closed surrounding	30 meters
With antenna in opened surrounding	43 meters
In the drawer	Yes
Below the stairs	Yes
2^{nd} floor	Yes
In other room	Yes

Table 5.1 shows the capability of this project. In the open area, the transmission can be done successfully in the range of 43 meters. While in the closed area such as in the room or in the house, the buzzer can be detected about 30 meters in range. In addition, the remote locater also can even respond when it caught between the sofa, through the wall, under the stairs or in other room.

CHAPTER 6

CONCLUSION

This thesis has been discussed about the process development of the project titled “Microcontroller based Remote Locator using Radio Frequency Signal”. Generally, the main objective stated on Chapter 1 has been achieved. The main objective of this project is to develop a communication between two microcontrollers using radio frequency signal. Other than that, this project is also the improvement from the previous project because this project was using less hardware. In addition, the weight and the size of this remote locator also light and small.

This chapter also will discussed about the problems occur during the implementation of this project and the improvement that can be made for this project.

6.1 Introduction

As a conclusion, the project is successfully implemented using inexpensive and less hardware used. To sum up the overall project, the main component and their function are:

- ✓ PIC16F877A and PPIC16F876A is work as a main controller which are used in the both Control unit and Remote unit.
- ✓ RF module is used as a Radio Frequency signal generator to create radio frequency signal as a medium of transmission.

6.2 Problem occurs

While developing the project, many problems were occurring. One of them is noise during transmission. Data transmission which contains a noise will disturb the process. While the button was not pushed, the buzzer will also produce an audible sound. This sound is caused by the unwanted signal that was transmitted by the transmitter. As we know, the transmitter will always send the signal, but it is receiver job to differentiate which are the actual data that it must receive.

The other problem occurs during hardware design. A lot of time was spent during the hardware implementation because there receivers need to be designed plus with the transmitter. That means, in this project, four board of circuit need to be developed. In addition, all the circuit was done one by one to make sure it will much easier to troubleshoot.

6.3 Project improvement

To make this remote locator more precise and user-friendly, RF module can be changed to RFID. This modification can fit the size and make the overall project much smaller and lighter. Other than that, when using RFID, much application can be implemented on the project.

Another improvement is to change the receiver and transmitter to the transceiver. Many applications can be implemented if the transceiver is being used such as distance calculation.

Many menus can be added such as the list of items and distance calculation for the LCD application.

REFERENCES

1. Microcontroller: http://en.wikipedia.org/wiki/Microcontroller
2. Embedded system : http://en.wikipedia.org/wiki/Embedded _system
3. PIC Microcontroller : http://en.wikipedia.org/wiki/PIC_mirocontroller
4. PIC16F87XA Data Sheet 28/40/44-Pin Enhanced Flash Microcontrollers , Microchip Technology Inc. , 2003
5. PIC : http://www.tinel.cat/~fmco/pic_sp.html#876differ
6. Myke Predko, (2008), "Programming and Costumizing the PIC Microcontroller", The McGraw.Hill Companies.Inc
7. Marlin Bates, (2004), "PIC Microcontroler An Introduction to Microelectronics", Elsevier Ltd
8. PIC Basic Hardware Structure : http.//www.scrid.com/doc/25901509/PIC-Basic-Hardware-Structure
9. W.M Hafidz Bin Mohd Jaafar,(2005), " Microcontroller Based Remote Locater Using Radio Frequency Signal", Universiti Teknologi Malaysia , Skudai
10. PIEZO BUZZER 12VDC,LEADS : http://cpc.farnell.com/_/abi-023-rc/Piezo-buzzer-12vdc-leads/dp/LS02470
11. Non Return to Zero : http://en.wikipedia.org/wiki/Non-Return-to-zero
12. Programmer : http://en.wikipedia.org/wiki/Programmer_(hardware)
13. ISM Band : http://en.wikipedia.org/wiki/ISM_Band
14. Julio Sanchez,Maria P. Canton, (2007), "Microcontroller Programming The Microchip PIC ", Taylor & Francis Group, USA ,

15. RF Communication between microcontroller(Part2) : http://extremeelectronics.co.in/avr-tutorials/rf-communication-between-microcontrollers-part-ii/

APPENDIX A

4.0 I/O PORTS

Some pins for these I/O ports are multiplexed with an alternate function for the peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin.

Additional information on I/O ports may be found in the PICmicro™ Mid-Range Reference Manual (DS33023).

4.1 PORTA and the TRISA Register

PORTA is a 6-bit wide, bidirectional port. The corresponding data direction register is TRISA. Setting a TRISA bit (= 1) will make the corresponding PORTA pin an input (i.e., put the corresponding output driver in a High-Impedance mode). Clearing a TRISA bit (= 0) will make the corresponding PORTA pin an output (i.e., put the contents of the output latch on the selected pin).

Reading the PORTA register reads the status of the pins, whereas writing to it will write to the port latch. All write operations are read-modify-write operations. Therefore, a write to a port implies that the port pins are read, the value is modified and then written to the port data latch.

Pin RA4 is multiplexed with the Timer0 module clock input to become the RA4/T0CKI pin. The RA4/T0CKI pin is a Schmitt Trigger input and an open-drain output. All other PORTA pins have TTL input levels and full CMOS output drivers.

Other PORTA pins are multiplexed with analog inputs and the analog VREF input for both the A/D converters and the comparators. The operation of each pin is selected by clearing/setting the appropriate control bits in the ADCON1 and/or CMCON registers.

Note:	On a Power-on Reset, these pins are configured as analog inputs and read as '0'. The comparators are in the off (digital) state.

The TRISA register controls the direction of the port pins even when they are being used as analog inputs. The user must ensure the bits in the TRISA register are maintained set when using them as analog inputs.

EXAMPLE 4-1: INITIALIZING PORTA

```
BCF     STATUS, RP0     ;
BCF     STATUS, RP1     ; Bank0
CLRF    PORTA           ; Initialize PORTA by
                        ; clearing output
                        ; data latches
BSF     STATUS, RP0     ; Select Bank 1
MOVLW   0x06            ; Configure all pins
MOVWF   ADCON1          ; as digital inputs
MOVLW   0xCF            ; Value used to
                        ; initialize data
                        ; direction
MOVWF   TRISA           ; Set RA<3:0> as inputs
                        ; RA<5:4> as outputs
                        ; TRISA<7:6>are always
                        ; read as '0'.
```

FIGURE 4-1: BLOCK DIAGRAM OF RA3:RA0 PINS

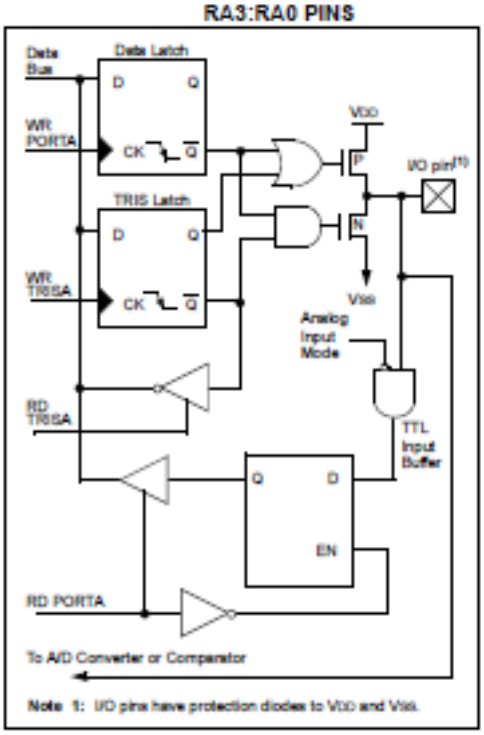
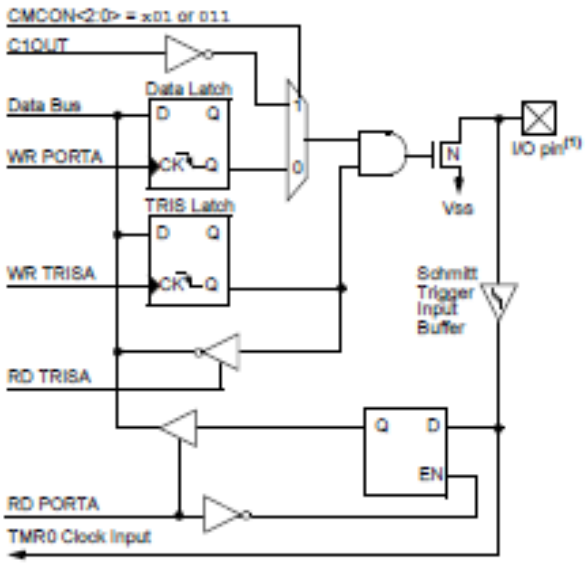
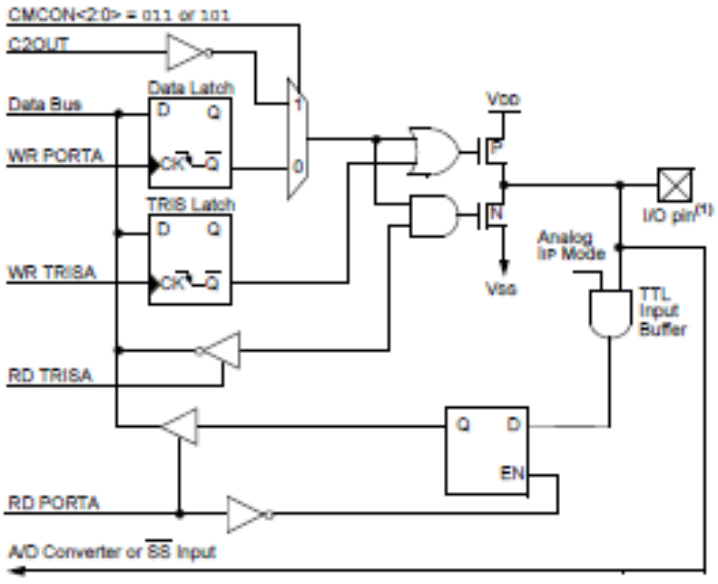


FIGURE 4-2: BLOCK DIAGRAM OF RA4/T0CKI PIN



Note 1: I/O pin has protection diodes to Vss only.

FIGURE 4-3: BLOCK DIAGRAM OF RA5 PIN



Note 1: I/O pin has protection diodes to Vdd and Vss.

TABLE 4-1: PORTA FUNCTIONS

Name	Bit#	Buffer	Function
RA0/AN0	bit 0	TTL	Input/output or analog input.
RA1/AN1	bit 1	TTL	Input/output or analog input.
RA2/AN2/VREF-/CVREF	bit 2	TTL	Input/output or analog input or VREF- or CVREF.
RA3/AN3/VREF+	bit 3	TTL	Input/output or analog input or VREF+.
RA4/T0CKI/C1OUT	bit 4	ST	Input/output or external clock input for Timer0 or comparator output. Output is open-drain type.
RA5/AN4/$\overline{SS}$/C2OUT	bit 5	TTL	Input/output or analog input or slave select input for synchronous serial port or comparator output.

Legend: TTL = TTL input, ST = Schmitt Trigger input

TABLE 4-2: SUMMARY OF REGISTERS ASSOCIATED WITH PORTA

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
05h	PORTA	—	—	RA5	RA4	RA3	RA2	RA1	RA0	--0x 0000	--0u 0000
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111	--11 1111
9Ch	CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	0000 0111	0000 0111
9Dh	CVRCON	CVREN	CVROE	CVRR	—	CVR3	CVR2	CVR1	CVR0	000- 0000	000- 0000
9Fh	ADCON1	ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0	00-- 0000	00-- 0000

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'. Shaded cells are not used by PORTA.

Note: When using the SSP module in SPI Slave mode and $\overline{SS}$ enabled, the A/D converter must be set to one of the following modes, where PCFG3:PCFG0 = 0100, 0101, 011x, 1101, 1110, 1111.

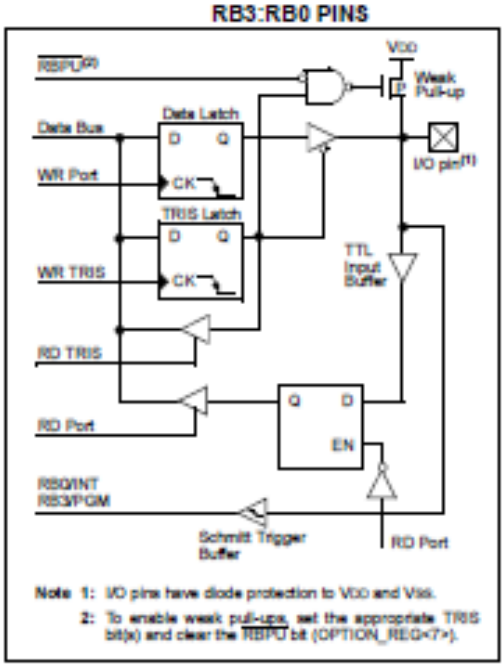
4.2 PORTB and the TRISB Register

PORTB is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISB. Setting a TRISB bit (= 1) will make the corresponding PORTB pin an input (i.e., put the corresponding output driver in a High-Impedance mode). Clearing a TRISB bit (= 0) will make the corresponding PORTB pin an output (i.e., put the contents of the output latch on the selected pin).

Three pins of PORTB are multiplexed with the In-Circuit Debugger and Low-Voltage Programming function: RB3/PGM, RB6/PGC and RB7/PGD. The alternate functions of these pins are described in **Section 14.0 "Special Features of the CPU"**.

Each of the PORTB pins has a weak internal pull-up. A single control bit can turn on all the pull-ups. This is performed by clearing bit $\overline{\text{RBPU}}$ (OPTION_REG<7>). The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on a Power-on Reset.

FIGURE 4-4: BLOCK DIAGRAM OF RB3:RB0 PINS



Note 1: I/O pins have diode protection to VDD and VSS.
2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the $\overline{\text{RBPU}}$ bit (OPTION_REG<7>).

Four of the PORTB pins, RB7:RB4, have an interrupt-on-change feature. Only pins configured as inputs can cause this interrupt to occur (i.e., any RB7:RB4 pin configured as an output is excluded from the interrupt-on-change comparison). The input pins (of RB7:RB4) are compared with the old value latched on the last read of PORTB. The "mismatch" outputs of RB7:RB4 are OR'ed together to generate the RB port change interrupt with flag bit RBIF (INTCON<0>).

This interrupt can wake the device from Sleep. The user, in the Interrupt Service Routine, can clear the interrupt in the following manner:

a) Any read or write of PORTB. This will end the mismatch condition.

b) Clear flag bit RBIF.

A mismatch condition will continue to set flag bit RBIF. Reading PORTB will end the mismatch condition and allow flag bit RBIF to be cleared.

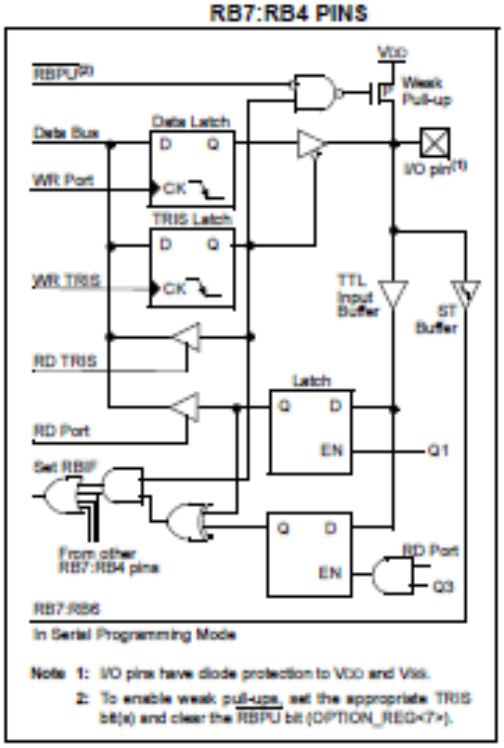
The interrupt-on-change feature is recommended for wake-up on key depression operation and operations where PORTB is only used for the interrupt-on-change feature. Polling of PORTB is not recommended while using the interrupt-on-change feature.

This interrupt-on-mismatch feature, together with software configurable pull-ups on these four pins, allow easy interface to a keypad and make it possible for wake-up on key depression. Refer to the application note, *AN662*, "*Implementing Wake-up on Key Stroke*" (DS00552).

RB0/INT is an external interrupt input pin and is configured using the INTEDG bit (OPTION_REG<6>).

RB0/INT is discussed in detail in **Section 14.11.1 "INT Interrupt"**.

FIGURE 4-5: BLOCK DIAGRAM OF RB7:RB4 PINS



Note 1: I/O pins have diode protection to VDD and VSS.
2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the $\overline{\text{RBPU}}$ bit (OPTION_REG<7>).

TABLE 4-3: PORTB FUNCTIONS

Name	Bit#	Buffer	Function
RB0/INT	bit 0	TTL/ST(1)	Input/output pin or external interrupt input. Internal software programmable weak pull-up.
RB1	bit 1	TTL	Input/output pin. Internal software programmable weak pull-up.
RB2	bit 2	TTL	Input/output pin. Internal software programmable weak pull-up.
RB3/PGM(3)	bit 3	TTL	Input/output pin or programming pin in LVP mode. Internal software programmable weak pull-up.
RB4	bit 4	TTL	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up.
RB5	bit 5	TTL	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up.
RB6/PGC	bit 6	TTL/ST(2)	Input/output pin (with interrupt-on-change) or in-circuit debugger pin. Internal software programmable weak pull-up. Serial programming clock.
RB7/PGD	bit 7	TTL/ST(2)	Input/output pin (with interrupt-on-change) or in-circuit debugger pin. Internal software programmable weak pull-up. Serial programming data.

Legend: 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 or in-circuit debugger.

3: Low-Voltage ICSP Programming (LVP) is enabled by default which disables the RB3 I/O function. LVP must be disabled to enable RB3 as an I/O pin and allow maximum compatibility to the other 28-pin and 40-pin mid-range devices.

TABLE 4-4: SUMMARY OF REGISTERS ASSOCIATED WITH PORTB

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
06h, 106h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	uuuu uuuu
86h, 186h	TRISB	PORTB Data Direction Register								1111 1111	1111 1111
81h, 181h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

Legend: x = unknown, u = unchanged. Shaded cells are not used by PORTB.

4.3 PORTC and the TRISC Register

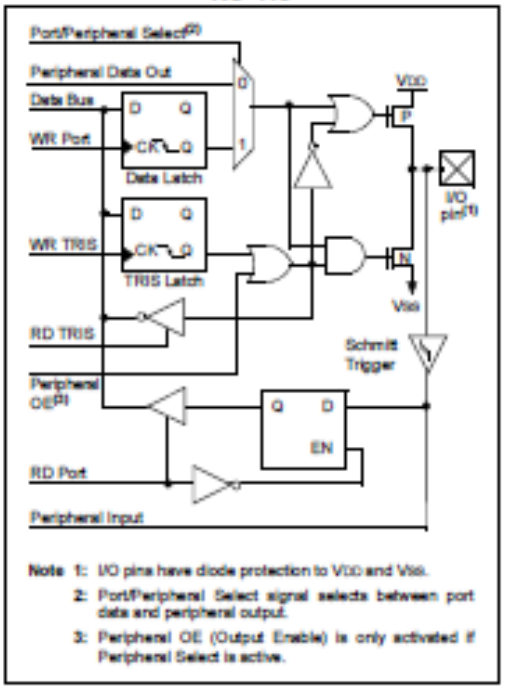
PORTC is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISC. Setting a TRISC bit (= 1) will make the corresponding PORTC pin an input (i.e., put the corresponding output driver in a High-Impedance mode). Clearing a TRISC bit (= 0) will make the corresponding PORTC pin an output (i.e., put the contents of the output latch on the selected pin).

PORTC is multiplexed with several peripheral functions (Table 4-5). PORTC pins have Schmitt Trigger input buffers.

When the I^2C module is enabled, the PORTC<4:3> pins can be configured with normal I^2C levels, or with SMBus levels, by using the CKE bit (SSPSTAT<6>).

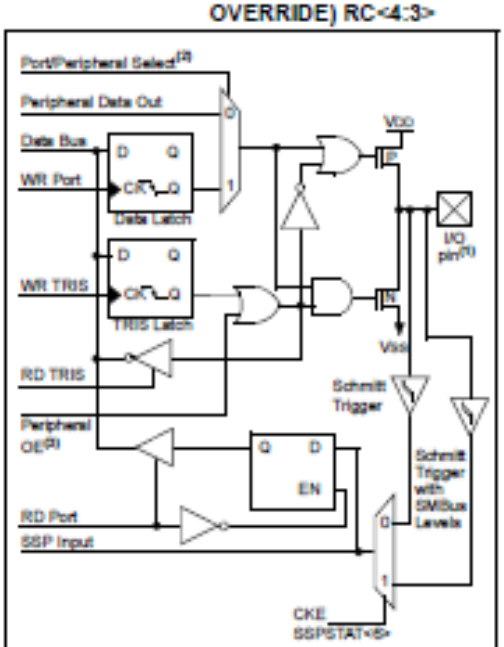
When enabling peripheral functions, care should be taken in defining TRIS bits for each PORTC pin. Some peripherals override the TRIS bit to make a pin an output, while other peripherals override the TRIS bit to make a pin an input. Since the TRIS bit override is in effect while the peripheral is enabled, read-modify-write instructions (BSF, BCF, XORWF) with TRISC as the destination, should be avoided. The user should refer to the corresponding peripheral section for the correct TRIS bit settings.

FIGURE 4-6: PORTC BLOCK DIAGRAM (PERIPHERAL OUTPUT OVERRIDE) RC<2:0>, RC<7:5>



Note 1: I/O pins have diode protection to VDD and VSS.
2: Port/Peripheral Select signal selects between port data and peripheral output.
3: Peripheral OE (Output Enable) is only activated if Peripheral Select is active.

FIGURE 4-7: PORTC BLOCK DIAGRAM (PERIPHERAL OUTPUT OVERRIDE) RC<4:3>



Note 1: I/O pins have diode protection to VDD and VSS.
2: Port/Peripheral Select signal selects between port data and peripheral output.
3: Peripheral OE (Output Enable) is only activated if Peripheral Select is active.

TABLE 4-5: PORTC FUNCTIONS

Name	Bit#	Buffer Type	Function
RC0/T1OSO/T1CKI	bit 0	ST	Input/output port pin or Timer1 oscillator output/Timer1 clock input.
RC1/T1OSI/CCP2	bit 1	ST	Input/output port pin or Timer1 oscillator input or Capture2 input/ Compare2 output/PWM2 output.
RC2/CCP1	bit 2	ST	Input/output port pin or Capture1 input/Compare1 output/ PWM1 output.
RC3/SCK/SCL	bit 3	ST	RC3 can also be the synchronous serial clock for both SPI and I^2C modes.
RC4/SDI/SDA	bit 4	ST	RC4 can also be the SPI data in (SPI mode) or data I/O (I^2C mode).
RC5/SDO	bit 5	ST	Input/output port pin or Synchronous Serial Port data output.
RC6/TX/CK	bit 6	ST	Input/output port pin or USART asynchronous transmit or synchronous clock.
RC7/RX/DT	bit 7	ST	Input/output port pin or USART asynchronous receive or synchronous data.

Legend: ST = Schmitt Trigger input

TABLE 4-6: SUMMARY OF REGISTERS ASSOCIATED WITH PORTC

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
07h	PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxx xxxx	uuuu uuuu
87h	TRISC	PORTC Data Direction Register								1111 1111	1111 1111

Legend: x = unknown, u = unchanged

4.4 PORTD and TRISD Registers

Note:	PORTD and TRISD are not implemented on the 28-pin devices.

PORTD is an 8-bit port with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output.

PORTD can be configured as an 8-bit wide microprocessor port (Parallel Slave Port) by setting control bit, PSPMODE (TRISE<4>). In this mode, the input buffers are TTL.

FIGURE 4-8: PORTD BLOCK DIAGRAM (IN I/O PORT MODE)

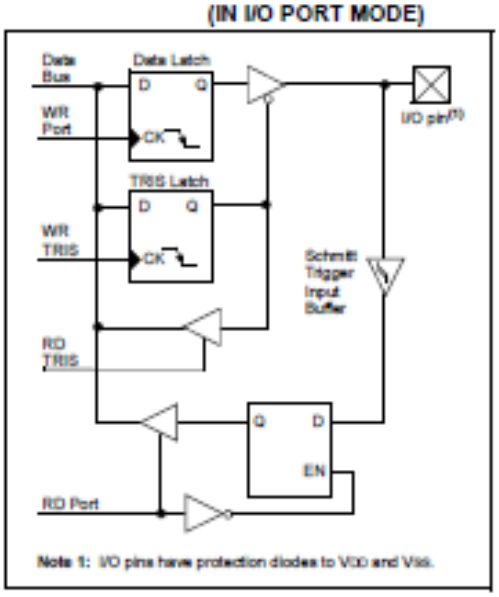


TABLE 4-7: PORTD FUNCTIONS

Name	Bit#	Buffer Type	Function
RD0/PSP0	bit 0	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 0.
RD1/PSP1	bit 1	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 1.
RD2/PSP2	bit2	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 2.
RD3/PSP3	bit 3	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 3.
RD4/PSP4	bit 4	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 4.
RD5/PSP5	bit 5	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 5.
RD6/PSP6	bit 6	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 6.
RD7/PSP7	bit 7	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 7.

Legend: ST = Schmitt Trigger input, TTL = TTL input

Note 1: Input buffers are Schmitt Triggers when in I/O mode and TTL buffers when in Parallel Slave Port mode.

TABLE 4-8: SUMMARY OF REGISTERS ASSOCIATED WITH PORTD

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
08h	PORTD	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	xxxx xxxx	uuuu uuuu
88h	TRISD	PORTD Data Direction Register								1111 1111	1111 1111
89h	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction Bits			0000 -111	0000 -111

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by PORTD.

4.5 PORTE and TRISE Register

Note:	PORTE and TRISE are not implemented on the 28-pin devices.

PORTE has three pins (RE0/$\overline{RD}$/AN5, RE1/$\overline{WR}$/AN6 and RE2/$\overline{CS}$/AN7) which are individually configurable as inputs or outputs. These pins have Schmitt Trigger input buffers.

The PORTE pins become the I/O control inputs for the microprocessor port when bit PSPMODE (TRISE<4>) is set. In this mode, the user must make certain that the TRISE<2:0> bits are set and that the pins are configured as digital inputs. Also, ensure that ADCON1 is configured for digital I/O. In this mode, the input buffers are TTL.

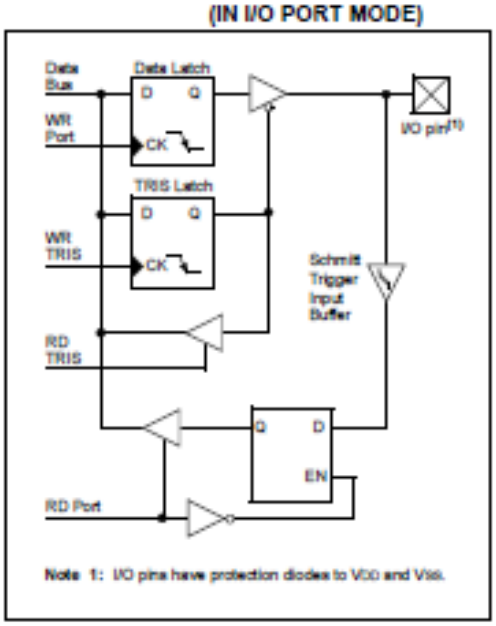
Register 4-1 shows the TRISE register which also controls the Parallel Slave Port operation.

PORTE pins are multiplexed with analog inputs. When selected for analog input, these pins will read as '0's.

TRISE controls the direction of the RE pins, even when they are being used as analog inputs. The user must make sure to keep the pins configured as inputs when using them as analog inputs.

Note:	On a Power-on Reset, these pins are configured as analog inputs and read as '0'.

FIGURE 4-9: PORTE BLOCK DIAGRAM (IN I/O PORT MODE)



Note 1: I/O pins have protection diodes to VDD and VSS.

TABLE 4-9: PORTE FUNCTIONS

Name	Bit#	Buffer Type	Function
RE0/$\overline{RD}$/AN5	bit 0	ST/TTL(1)	I/O port pin or read control input in Parallel Slave Port mode or analog input: $\overline{RD}$ 1 = Idle 0 = Read operation. Contents of PORTD register are output to PORTD I/O pins (if chip selected).
RE1/$\overline{WR}$/AN6	bit 1	ST/TTL(1)	I/O port pin or write control input in Parallel Slave Port mode or analog input: $\overline{WR}$ 1 = Idle 0 = Write operation. Value of PORTD I/O pins is latched into PORTD register (if chip selected).
RE2/$\overline{CS}$/AN7	bit 2	ST/TTL(1)	I/O port pin or chip select control input in Parallel Slave Port mode or analog input: $\overline{CS}$ 1 = Device is not selected 0 = Device is selected

Legend: ST = Schmitt Trigger input, TTL = TTL input

Note 1: Input buffers are Schmitt Triggers when in I/O mode and TTL buffers when in Parallel Slave Port mode.

TABLE 4-10: SUMMARY OF REGISTERS ASSOCIATED WITH PORTE

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
09h	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	---- -uuu
89h	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction bits			0000 -111	0000 -111
9Fh	ADCON1	ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0	00-- 0000	00-- 0000

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by PORTE.

REGISTER 4-1: TRISE REGISTER (ADDRESS 89h)

R-0	R-0	R/W-0	R/W-0	U-0	R/W-1	R/W-1	R/W-1
IBF	OBF	IBOV	PSPMODE	—	Bit 2	Bit 1	Bit 0
bit 7							bit 0

Parallel Slave Port Status/Control Bits:

bit 7 **IBF**: Input Buffer Full Status bit
1 = A word has been received and is waiting to be read by the CPU
0 = No word has been received

bit 6 **OBF**: Output Buffer Full Status bit
1 = The output buffer still holds a previously written word
0 = The output buffer has been read

bit 5 **IBOV**: Input Buffer Overflow Detect bit (in Microprocessor mode)
1 = A write occurred when a previously input word has not been read (must be cleared in software)
0 = No overflow occurred

bit 4 **PSPMODE**: Parallel Slave Port Mode Select bit
1 = PORTD functions in Parallel Slave Port mode
0 = PORTD functions in general purpose I/O mode

bit 3 **Unimplemented**: Read as '0'

PORTE Data Direction Bits:

bit 2 **Bit 2**: Direction Control bit for pin RE2/$\overline{CS}$/AN7
1 = Input
0 = Output

bit 1 **Bit 1**: Direction Control bit for pin RE1/$\overline{WR}$/AN6
1 = Input
0 = Output

bit 0 **Bit 0**: Direction Control bit for pin RE0/$\overline{RD}$/AN5
1 = Input
0 = Output

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

APPENDIX B

2. PRODUCT SPECIFICATION

2.1 The Specifications of RF Transmitter Module

Except for the frequency and antenna length, RF_TX_315 and RF_TX_433 share the same product specifications as shown in table below:

No.	Specifications	RF Transmitter Module
1.	Operating Voltage	3V to 12 V
2.	Operating Current	Max ≤ 40mA (12V), Min ≤ 9mA (3V)
3.	Oscillator	SAW (Surface Acoustic Wave) oscillator
4.	Frequency	315MHz~433.92MHz
5.	Frequency error	±150kHz(max)
6.	Modulation	ASK/OOK
7.	Transfer Rate	≤10Kbps
8.	Transmitting power	25mW (315MHz@12V)
9.	Antenna Length	24cm (315MHz), 18cm (433.92MHz)

2.2 Antenna

1. User may use any soft or hard wire (likes Drawbars antenna) as antenna. The frequency is determined by the length of antenna, **please select the correct length with refer to specification of RF Transmitter above** (Section 2.1, No. 9). If a soft wire is used, please make sure it is fully extended.

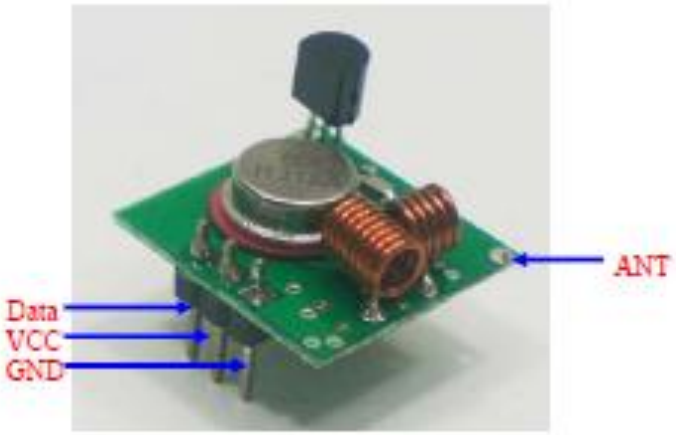
2. If the transmitter module is molded in a metal casing, please use an external antenna. For better result, use A 50 Ohm coaxial cable can be used as antenna to the module.

Created by Cytron Technologies Sdn. Bhd. – All Rights Reserved

3. PRODUCT LAYOUT

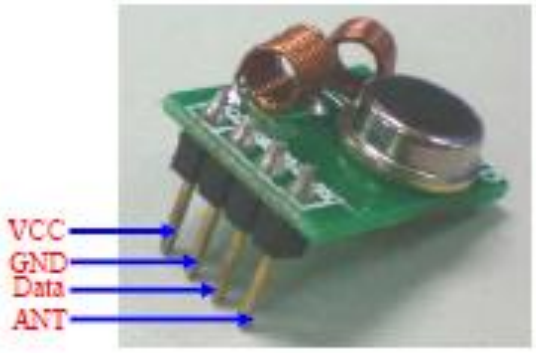
3.1 RF_TX_315MHz

3.1.1 RF_TX_315MHz Type1



Label	Description
Data	The Data pin of the transmitter.
VCC	The power supply to the transmitter.
GND	The Ground of the transmitter.
ANT	The hole to solder and connect antenna. (Please select the correct antenna length, which is 24cm)

3.1.2 RF_TX_315MHz Type2



Label	Description
Data	The Data pin of the transmitter.
VCC	The power supply to the transmitter.
GND	The Ground of the transmitter.
ANT	The pin connect antenna. (Please select the correct antenna length, which is 24cm)

Created by Cytron Technologies Sdn. Bhd. – All Rights Reserved

2. PRODUCT SPECIFICATION

2.1 The Specifications of RF Receiver

Except for the frequency and antenna length, RF_RX_315 and RF_RX_433 share the same product specifications as shown in table below:

No.	Specifications	RF Receiver
1.	Operating Voltage	5.0V ± 0.5V
2.	Operating Current	≤5.5mA @5.0V
3.	Operating Principle	Monolithic super heterodyne receiving
4.	Modulation	OOK/ASK
5.	Frequency	315MHz, 433.92MHz
6.	Bandwidth	2MHz
7.	Sensitivity	-100dBm
8.	Rate	<9.6Kbps (315MHz @-95dBm)
9.	Data Output	TTL
10.	Antenna Length	24cm (315MHz), 18cm (433.92MHz)

2.2 Antenna

1. User may use any soft or hard wire (likes Drawbars antenna) as antenna. If the soft wire is used, do make sure it is fully extended. The distance of reception will be influence by the length of antenna; **please select the correct length with refer to specifications of RF Receiver above.** (Section 2.1, No. 10). Please keep the RF Receiver Module away from metal objects.

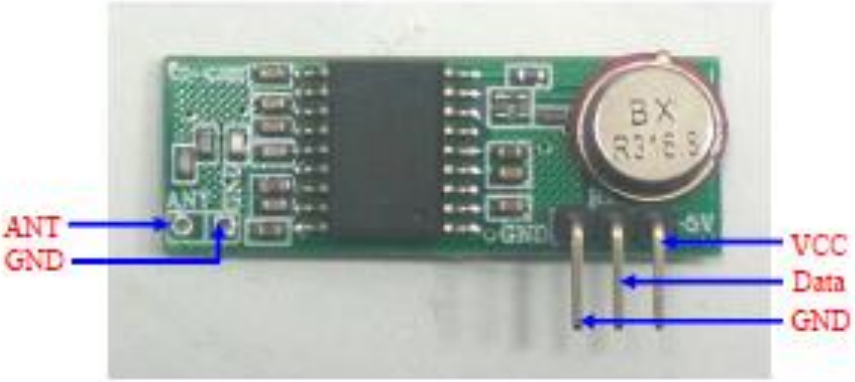
2.3 Important Notes

1. If the module is used with microcontroller, the clock frequency should be under 4MHz. Please try to keep a distance between oscillator and the RF Receiver module to avoid the disturbance from oscillator.
2. The voltage supply need to stable and the ripple voltage need to be as low as possible, multi-level filtering are needed. (For example, add ferrite bead, inductor and capacitor.)

Created by Cytron Technologies Sdn. Bhd. – All Rights Reserved

3. PRODUCT LAYOUT

3.1 RF_RX_315MHz



Label	Description
ANT	The hole to solder and connect antenna. (Please select the correct antenna length, which is 24cm)
VCC	The power supply (5V) to the receiver.
GND	The Ground of the receiver. (The 2 GND are internally connected each other.)
Data	The Data pin of the receiver.

Label	Description
ANT	The hole to solder and connect antenna. (Please select the correct antenna length, which is 24cm)
VCC	The power supply (5V) to the receiver.
GND	The Ground of the receiver.
Data	The Data pin of the receiver. (The 2 Data pins are internally connected each other.)

10.1 USART Baud Rate Generator (BRG)

The BRG supports both the Asynchronous and Synchronous modes of the USART. It is a dedicated 8-bit baud rate generator. The SPBRG register controls the period of a free running 8-bit timer. In Asynchronous mode, bit BRGH (TXSTA<2>) also controls the baud rate. In Synchronous mode, bit BRGH is ignored. Table 10-1 shows the formula for computation of the baud rate for different USART modes which only apply in Master mode (internal clock).

Given the desired baud rate and Fosc, the nearest integer value for the SPBRG register can be calculated using the formula in Table 10-1. From this, the error in baud rate can be determined.

It may be advantageous to use the high baud rate (BRGH = 1) even for slower baud clocks. This is because the Fosc/(16 (X + 1)) equation can reduce the baud rate error in some cases.

Writing a new value to the SPBRG register causes the BRG timer to be reset (or cleared). This ensures the BRG does not wait for a timer overflow before outputting the new baud rate.

10.1.1 SAMPLING

The data on the RC7/RX/DT pin is sampled three times by a majority detect circuit to determine if a high or a low level is present at the RX pin.

TABLE 10-1: BAUD RATE FORMULA

SYNC	BRGH = 0 (Low Speed)	BRGH = 1 (High Speed)
0	(Asynchronous) Baud Rate = Fosc/(64 (X + 1))	Baud Rate = Fosc/(16 (X + 1))
1	(Synchronous) Baud Rate = Fosc/(4 (X + 1))	N/A

Legend: X = value in SPBRG (0 to 255)

TABLE 10-2: REGISTERS ASSOCIATED WITH BAUD RATE GENERATOR

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
18h	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	0000 000x
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented, read as '0'. Shaded cells are not used by the BRG.

TABLE 10-3: BAUD RATES FOR ASYNCHRONOUS MODE (BRGH = 0)

BAUD RATE (K)	Fosc = 20 MHz			Fosc = 16 MHz			Fosc = 10 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	-	-	-	-	-	-	-	-	-
1.2	1.221	1.75	255	1.202	0.17	207	1.202	0.17	129
2.4	2.404	0.17	129	2.404	0.17	103	2.404	0.17	64
9.6	9.766	1.73	31	9.615	0.16	25	9.766	1.73	15
19.2	19.531	1.72	15	19.231	0.16	12	19.531	1.72	7
28.8	31.250	8.51	9	27.778	3.55	8	31.250	8.51	4
33.6	34.722	3.34	8	35.714	6.29	6	31.250	6.99	4
57.6	62.500	8.51	4	62.500	8.51	3	52.083	9.58	2
HIGH	1.221	-	255	0.977	-	255	0.610	-	255
LOW	312.500	-	0	250.000	-	0	156.250	-	0

BAUD RATE (K)	Fosc = 4 MHz			Fosc = 3.6864 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	0.300	0	207	0.3	0	191
1.2	1.202	0.17	51	1.2	0	47
2.4	2.404	0.17	25	2.4	0	23
9.6	8.929	6.99	6	9.6	0	5
19.2	20.833	8.51	2	19.2	0	2
28.8	31.250	8.51	1	28.8	0	1
33.6	-	-	-	-	-	-
57.6	62.500	8.51	0	57.6	0	0
HIGH	0.244	-	255	0.225	-	255
LOW	62.500	-	0	57.6	-	0

TABLE 10-4: BAUD RATES FOR ASYNCHRONOUS MODE (BRGH = 1)

BAUD RATE (K)	Fosc = 20 MHz			Fosc = 16 MHz			Fosc = 10 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	-	-	-	-	-	-	-	-	-
1.2	-	-	-	-	-	-	-	-	-
2.4	-	-	-	-	-	-	2.441	1.71	255
9.6	9.615	0.16	129	9.615	0.16	103	9.615	0.16	64
19.2	19.231	0.16	64	19.231	0.16	51	19.531	1.72	31
28.8	29.070	0.94	42	29.412	2.13	33	28.409	1.36	21
33.6	33.784	0.55	36	33.333	0.79	29	32.895	2.10	18
57.6	59.524	3.34	20	58.824	2.13	16	56.818	1.36	10
HIGH	4.883	-	255	3.906	-	255	2.441	-	255
LOW	1250.000	-	0	1000.000		0	625.000	-	0

BAUD RATE (K)	Fosc = 4 MHz			Fosc = 3.6864 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	-	-	-	-	-	-
1.2	1.202	0.17	207	1.2	0	191
2.4	2.404	0.17	103	2.4	0	95
9.6	9.615	0.16	25	9.6	0	23
19.2	19.231	0.16	12	19.2	0	11
28.8	27.798	3.55	8	28.8	0	7
33.6	35.714	6.29	6	32.9	2.04	6
57.6	62.500	8.51	3	57.6	0	3
HIGH	0.977	-	255	0.9	-	255
LOW	250.000	-	0	230.4	-	0

APPENDIX D

```
//=====================================================================
// Coding for Transmitter
//=====================================================================
#include <htc.h>
//=====================================================================
//configuration
//=====================================================================
__CONFIG (0x3F3A);
//=====================================================================
//     define
//=====================================================================
#define      button RA0
#define button2     RA1
#define button3     RA2
#define lcd_data    PORTB
#define rs RD5
#define rw RD6
#define e RD7
//=====================================================================
//     function prototype
//=====================================================================
void uart_send(unsigned char data);
void InitLCD(void);
void delay(unsigned long data);
void send_char(unsigned char data);
void send_config(unsigned char data);
void lcd_goto(unsigned char data);
void lcd_clr(void);
void send_string(const char *s);
unsigned char read_eeprom(unsigned char adr);
//=====================================================================
//     main function
//=====================================================================
void main(void)
```

```
{
	//assign variable
	unsigned char no,ptr,data;


	ADCON1= 0x06;			//configure PortA as digital I/O
	TRISA = 0b111111;		//configure PORTA input
	TRISB = 0;
	TRISD = 0;

	InitLCD();

	//setup USART
	BRGH = 0;			//baud rate for low speed option
	SPBRG = 255;			//set boud rate to 1200bps for 20Mhz
	TX9 = 0;			//8-bit transmission
	TXEN = 1;			//enable transmission
	SYNC = 0;			//asynchronous
	SPEN = 1;			//enable serial port

	lcd_goto(0);
	data=0;
	for(ptr=0x00;ptr<=0x0F;ptr++)
		{
			if(data<=15)
			{
			send_config(0x80 + data);
			send_char(read_eeprom(ptr));
			data=data+1;
			delay(5000);
			}
		}
	lcd_goto(20);
	data=0;
	for(ptr=0x10;ptr<=0x1F;ptr++)
		{
```

```
                    if(data<=15)
                    {
                    send_config(0xC0 + data);
                    send_char(read_eeprom(ptr));
                    data=data+1;
                    delay(5000);
                    }
      }
while(1)                                        //infinity loop
{
//=============================================================================
//Send enable signal
//=============================================================================
             no=0xAA;
             uart_send(no);
             no=0xBB;
             uart_send(no);

if(button==0)
      {
             {

                    no=0b00110011;                    //send the id number
                    lcd_clr();
                    lcd_goto(0);
                    send_string("RECEIVER 1--> ON");
                    lcd_goto(20);
                    send_string("Searching >>>");
             }
                    while(button==0)
                           uart_send(no);             //continuous send data
      }
      else if(button2==0)
      {
             {
                    no=0b11001100;                    //send the id number
```

```
                lcd_clr();
                lcd_goto(0);
                send_string("RECEIVER 2--> ON");
                lcd_goto(20);
                send_string("Searching >>>");
        }
                while(button2==0)
                        uart_send(no);              //continuous send data
    }
        else if(button3==0)
    {
        {
                no=0b11110000;                      //send the id number
                lcd_clr();
                lcd_goto(0);
                send_string("RECEIVER 3--> ON");
                lcd_goto(20);
                send_string("Searching >>>");
        }
                while(button3==0)
                        uart_send(no);              //continuous send data
    }
    }
}
//=============================================================================
//     functions
//=============================================================================
void uart_send(unsigned char data)
{
while(TXIF==0);     //only send the new data after the previous is sent
TXREG=data;
}

void InitLCD(void)
{
    send_config(0b00000001);  //clear display at lcd
```

```
	send_config(0b00000010);  //Lcd Return to home
	send_config(0b00000101);  //entry mode-cursor increase 1
	send_config(0b00001100);  //diplay on, cursor off and cursor blink
off
	send_config(0b00111000);  //function
}

void delay(unsigned long data)
{
	for( ;data>0;data-=1);
}

void send_char(unsigned char data)		//send lcd character
{
	rw=0;                                                   //set lcd to
write mode
	rs=1;                                                   //set lcd to
display mode
	lcd_data=data;                                          //lcd data port
= data
	e=1;                                                    //pulse e to
confirm the data
	delay(10);
	e=0;
	delay(10);
}

void send_config(unsigned char data)
{
	rw = 0;
	rs = 0;
	lcd_data = data;
	e=1;
	delay(50);
	e=0;
	delay(50);
}

void lcd_goto(unsigned char data)
```

```
{
	if(data<16)
	{send_config(0x80+data);}
	else
	{
		data = data - 20;
		send_config(0xC0 + data);
	}
}

void lcd_clr(void)
{
	send_config(0x01);
	delay(600);
}

void send_string(const char *s)
{
	unsigned char i=0;
	while (s&& *s)
	send_char (*s++);
}

unsigned char read_eeprom(unsigned char adr){
EEADR = adr;
RD=1; //set the read bit
return(EEDATA);
}
```

```
//=====================================================================
// Coding for REC1
//=====================================================================
#include <htc.h>
//=====================================================================
//     configuration
//=====================================================================
__CONFIG (0x3F3A);
//=====================================================================
//     define
//=====================================================================
#define buzzer      RC1
//=====================================================================
//     function
//=====================================================================
unsigned char uart_rec(void);
//=====================================================================
//     main function
//=====================================================================
void main(void)
{
       //assign variable
       unsigned char no;

       //set I/O input output
       TRISC= 0b10000000;

       //setup USART
       BRGH = 0;                  //baud rate for low speed option
       SPBRG = 255;               //set baud rate to 1200bps for 20Mhz
       SPEN = 1;                  //enable serial port
       RX9 = 0;                   //8-bit reception
       CREN = 1;                  //enable reception

       while(1)                   //infinity loop
```

```
	{
		CREN=1;
		if(0ERR==0)
		{
			if(uart_rec()==0xAA)
			{
				if(uart_rec()==0xBB)
				{
					if(uart_rec()==0b00110011)
					buzzer=1;
					else
						buzzer=0;
				}
		}
		}
	  else
				CREN=0;
	}
}
//=============================================================================
//     functions
//=============================================================================
unsigned char uart_rec(void)     //receive uart value
{
	unsigned char rec_data;
	while(RCIF==0);                         //wait for data
	rec_data = RCREG;
	return rec_data;                 //return the received data
}
```

APPENDIX E

Overall Project View