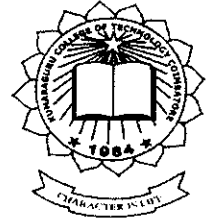


P- 3555



## **VEHICLE ACTUATED TRAFFIC SIGNAL CONTROL SYSTEM**

**By**

**G.NIKHIL**

**Reg No: 0710107065**

**R.VARUNKUMAR**

**Reg No: 0710107108**

**T.VIVEKGANESH**

**Reg No: 0710107116**

**G.VIVEKANAND**

**Reg No: 0710107117**

*of*

**KUMARAGURU COLLEGE OF TECHNOLOGY,  
COIMBATORE - 641 049.**

**(An Autonomous Institution affiliated to Anna University of Technology, Coimbatore)**

**A PROJECT REPORT**

*Submitted to the*

**FACULTY OF ELECTRONICS AND COMMUNICATION  
ENGINEERING**

*In partial fulfillment of the requirements*

*for the award of the degree*

*of*

**BACHELOR OF ENGINEERING**

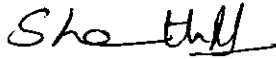
*in*

**ELECTRONICS AND COMMUNICATION ENGINEERING**

**APRIL 2011**

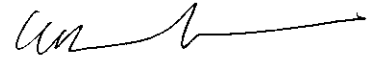
## BONAFIDE CERTIFICATE

Certified that this project report titled “**Vehicle Actuated Traffic Signal Control System**” is the bonafide work of **Mr.G.Nikhil** [Reg.No: 0710107065], **Mr.R.Varunkumar** [Reg.No:0710107108], **Mr.T.Vivekganesh** [Reg. No: 0710107116] and **Mr.G.Vivekanand** [Reg.No: 0710107117] who carried out the research under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.



**Project Guide**

Ms.M.Shanthi



**Head of the Department**

Dr. Rajeswari Mariappan

The candidate with Register No. 0710107065, 108, 116, 117..... is examined by us in the project viva-voce examination held on ..18..A..11.



18/4/11  
**Internal Examiner**



**External Examiner**

## ACKNOWLEDGEMENT

First I would like to express my praise and gratitude to the Lord, who has showered his grace and blessing enabling me to complete this project in an excellent manner.

I express my sincere thanks to our beloved Director **Dr.J.Shanmugam**, Kumaraguru College of Technology, for his kind support and for providing necessary facilities to carry out the work.

I express my sincere thanks to our beloved Principal **Dr.S.Ramachandran**, Kumaraguru College of Technology, who encouraged me in each and every steps of the project work.

I would like to express my deep sense of gratitude to our HOD, the ever active **Dr.Rajeswari Mariappan**, Department of Electronics and Communication Engineering, for her valuable suggestions and encouragement which paved way for the successful completion of the project work.

In particular, I wish to thank with everlasting gratitude to the Project Coordinator **Ms.K.Kavitha.**, Assistant Professor-SRG, Department of Electronics and Communication Engineering, for her expert counseling and guidance to make this project to a great deal of success.

I am greatly privileged to express my heartfelt thanks to my project guide **Ms.M.Shanthi**, Associate Professor, Department of Electronics and Communication Engineering, throughout the course of this project work and I wish to convey my deep sense of gratitude to all the teaching and non-teaching staffs of ECE Department for their help and cooperation.

Finally, I thank my parents and my family members for giving me the moral support and abundant blessings in all of my activities and my dear friends who helped me to endure my difficult times with their unfailing support and warm wishes.

## ABSTRACT

In city life, one of the problems people face is being late to offices and schools due to heavy traffic. Embedded systems provide an easy solution to this problem. Using embedded system, traffic signals can be controlled according to the density of the vehicles on the road so that prolonged unnecessary waiting of vehicles in each signal junction can be minimized.

Vehicle actuated traffic control system mainly involves controlling the traffic signal based on the density of traffic. IR sensors are placed in each lane to measure the density of traffic and send the corresponding information to the microcontroller. Basically, the traffic signal has a fixed cycle of direction .When the density of a particular lane is high, the microcontroller allocates more time for the 'GREEN' signal for that particular lane and the timing on the 'RED' signal for the other lanes is delayed by the extra time. The sensors are placed at three different positions along the road. If the first sensor is covered, then a previously set time delay is given.

In the second part of the project, importance is given to emergency vehicles by allowing them to pass the signal without any interference .The emergency vehicle is fit with a ZigBee device which communicates with the traffic signal controller in advance and gains complete priority for the direction of its approach. That particular lane is set to 'GREEN' no matter what the situation might be at the traffic signal. All the other lanes are set to 'RED' irrespective of its current signaling status.

## TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	<b>ACKNOWLEDGEMENT</b>	<b>iii</b>
	<b>ABSTRACT</b>	<b>iv</b>
	<b>TABLE OF CONTENTS</b>	<b>v</b>
	<b>LIST OF TABLES</b>	<b>vii</b>
	<b>LIST OF FIGURES</b>	<b>viii</b>
<b>1</b>	<b>INTRODUCTION</b>	
	1.1 GENERAL VIEWS	1
	1.2 BASIC PRINCIPLE	1
	1.3 BASIC LOGIC	2
	1.4 ADVANTAGES	4
	1.5 APPLICATIONS	4
<b>2</b>	<b>CIRCUIT DESCRIPTION</b>	
	2.1 CONTROLLING SECTION	5
	2.2 VEHICLE/AMBULANCE SECTION	7
<b>3</b>	<b>HARDWARE DESCRIPTION</b>	
	3.1 PIC 16F877A MICROCONTROLLER	9
	3.1.1 HIGH-PERFORMANCE RISC	10
	3.1.2 PERIPHERAL FEATURES	11
	3.1.3 ANALOG FEATURES	11
	3.1.4 SPECIAL FEATURES	12
	3.2 BCD TO SEVEN SEGMENT	
	DECODER IC 7447	13
	3.3 MAX 232	14

	3.4 ZIGBEE	16
	3.5 EXTERNAL CRYSTAL OSCILLATOR	16
<b>4</b>	<b>SOFTWARE DESCRIPTION</b>	
	4.1 PROGRAM DESCRIPTION	17
	4.2 ALGORITHM 1	18
	4.3 ALGORITHM 2	18
	4.4 FLOWCHART 1	19
	4.5 FLOWCHART 2	20
	<b>CONCLUSION</b>	21
	<b>ANNEXURE</b>	<b>I</b>
	<b>APPENDIX</b>	
	<b>REFERENCES</b>	<b>A</b>

## LIST OF TABLES

<b>TABLE NO</b>	<b>TITLE</b>	<b>PAGE NO</b>
3.1	IC 7447 Pin description	13
4.4	FLOWCHART 1	20
4.5	FLOWCHART 2	21

## LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
1.1	BASIC BLOCK DIAGRAM	3
1.2	SIMPLE ZIGBEE INRRERFACE BLOCK DIAGRAM	4
2.1	CONTROLLING SECTION	7
2.2	VEHICLE/AMBULANCE SECTION	8
3.1	PIN DETAILS OF PIC16F877A	10
3.2	PIN CONFIGURATION AND INTERFACING OF 7447 WITH SEVEN SEGMENT LED	13
3.3	PIN DETAILS OF MAX232	14



## CHAPTER 1

### INTRODUCTION

#### 1.1 GENERAL VIEWS

In city life, one of the problems people face is being late to offices and schools due to heavy traffic. Embedded systems provide an easy solution to this problem. Using embedded system, traffic signals can be controlled according to the density of the vehicles on the road so that prolonged unnecessary waiting of vehicles in each signal junction can be minimized to a great extent. Another critical problem is the blockage of emergency vehicles, like ambulance, fire fighting vehicle, etc in the traffic signals. In this project, priority is given to these vehicles.

Hence the project concentrates on the following issues, namely

- Traffic signaling based on traffic density.
- Giving preference to emergency vehicle

#### 1.2 BASIC PRINCIPLE

Density calculation is done by obtaining the count values from the IR sensors placed in the lanes and feeding them to the microcontroller. With this data, the time for 'RED' signal and 'GREEN' signals for each of the roads are allocated. Preference to ambulance is done by getting an alert from the vehicle through the ZigBee module at the traffic junction. The controller turns that particular road to be 'GREEN' and 'RED' signal is given to the other roads.

PIC microcontrollers are the best choice for embedded programming since they are RISC processors. Serial communication through ZigBee protocol is employed to transmit data between the vehicles and the receiver section which is the traffic signal controller.

ZigBee is a low-cost, low-power, wireless mesh networking standard. ZigBee is a specification for a suite of high level communication protocols using small, low-power digital radios based on the IEEE 802.15.4-2003 standard for Low-Rate Wireless Personal

Area Networks (LR-WPANs), such as wireless light switches with lamps, electrical meters with in-home-displays, consumer electronics equipment via short-range radio.

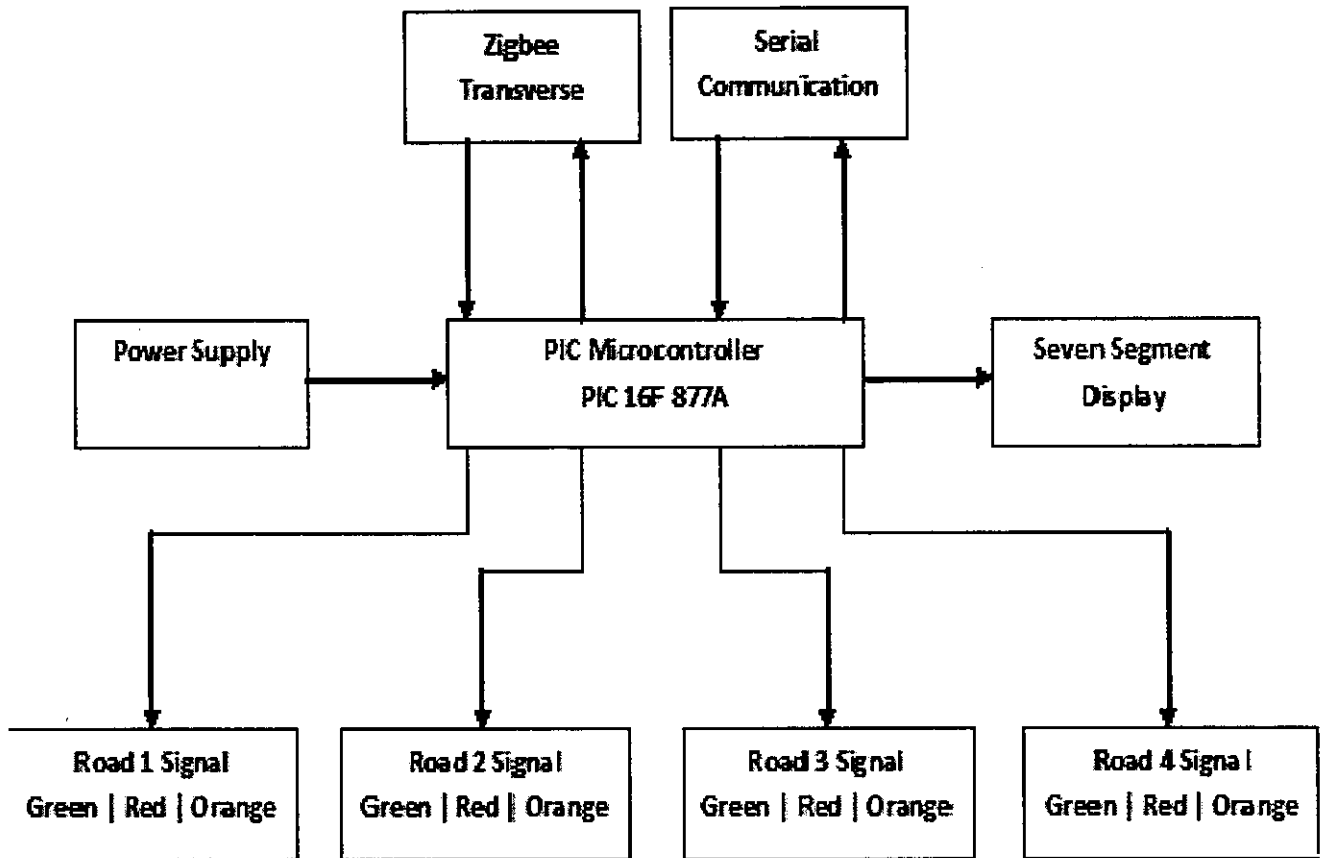
### **1.3 BASIC LOGIC**

The traffic density is calculated using the IR sensors which are placed along the sides of all the roads in a junction (for example there are three IR sensors placed one after another on each road at an approximate distance of 7-10mtrs from one another). When the IR sensor connection is not disrupted/blocked it means that there is no vehicle standing thus only the default time is allocated to it (example 10sec), like wise if there are vehicles standing between the first IR sensor then the connection is blocked and so there is a time delay given to that road's GREEN signal (example 10secs extra). Similarly if the second sensor is blocked then there is a larger time delay given to that road (example 20secs extra), finally if all three sensors are blocked then maximum time is allocated to that road's GREEN light (example 30secs extra).

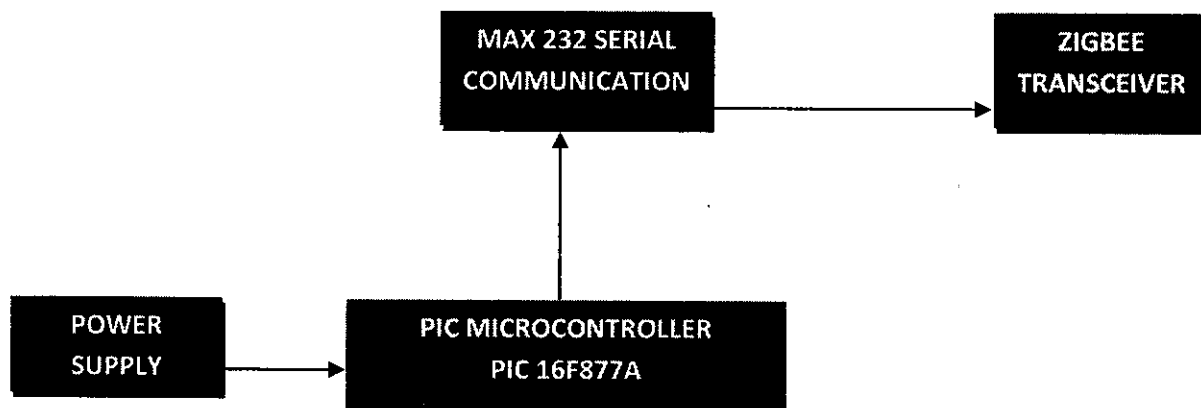
The signal will follow a cyclic process, starting from the road in the east to the road in the south, then to the road in the west and finally to the road in the north. There is a default time setting of approximately 360 seconds to complete each cycle (i.e. all four roads turning green and then back to red). During this period the preferences for each road is given based on the number of IR sensors that are blocked on that road. The cycle will not be disturbed at any point of time irrespective of the density on the roads.

Along with this, zigbee protocol is used to interface with the microcontroller and provide preference to emergency vehicles like ambulance and fire engines etc. A MAX232 IC is used to interface between the zigbee component and the PIC microcontroller. There are two zigbee transceivers used, one in the main signal junction and the other is placed in the vehicle. When the zigbee in the vehicle is turned on the signal reaches the zigbee transceiver in the main signal junction from where the signal goes to the microcontroller through the MAX232 IC which is used for TTL to CMOS conversion.

This idea is shown as a block diagram in Fig 1.1 and Fig 1.2 below:-



**Fig 1.1: Basic Block Diagram**



**Fig1.2 Simple Zigbee interface block diagram.**

#### **1.4 ADVANTAGES**

- Save time at each signal.
- Reduce manpower.
- Save traveling time for emergency vehicles.

#### **1.5 APPLICATIONS**

- Avoid congestion at a traffic signal junction.
- Life and properties are saved by providing preference to emergency vehicles.

## CHAPTER 2

### CIRCUIT DESCRIPTION

The process mainly consists of two sections the controlling section and the vehicle/ambulance section, as shown in the Fig 2.1 and Fig 2.2.

#### 2.1 CONTROLLING SECTION

The controlling section deals with the process of providing the appropriate time for each road based on the density of vehicles on that road. The main components of the controlling section are as follows:-

1. PIC16F877A Microcontroller
2. IC 7447
3. External Crystal Oscillator (4MHz)
4. IR Sensors
5. Seven Segment Display
6. MAX 232 IC
7. Zigbee

The PIC16F877A microcontroller forms the brain of the circuit.

There are 12 IR sensors used in the project, (3 on each road). The IR sensors are connected to the microcontrollers I/O Ports (RD0, RD1, RD2, RE0, RE1, RE2, RE6, RC0, RC1, RC3, RC4, RC5) there is a 5V supply through each sensor, when the vehicle disrupts the IR connection the 5V drops to 0V and so this drop in voltage is detected by the microcontroller which immediately varies the time provided to the road accordingly.

The external crystal oscillator is connected to the oscillator input ports(osc1,osc2), in the microcontroller. This is used to provide an external trigger pulse to the circuit.

In the power supply a step down transformer is used which converts the 230v input supply to 12v ,1A supply which is then passed to the Bridge rectifier which is used to convert the A.C to D.C.from which it goes through a few capacitors and a 7805 IC. The capacitors are used to filter the ripples or any remaining AC component left in the current and the 7805

IC is used to convert 12V DC to 5V DC, since the PIC microcontroller requires only 5V supply.

There are three seven segment displays used in the circuit to display the time/delay visually. A 5V supply is provided to each segment, and in order to interface the seven segment display with the microcontroller a 7447 decoder IC is used. The input ports of the 7447 IC (A,B,C,D) are interfaced with the I/O ports of the microcontroller (RBO, RB1, RB2, RB3).

Since the output from the microcontroller is in the BCD it needs to be converted into the format for the seven segment for the seven segment display, thus 7447 IC is used to do the conversion and the transfer of the data through the 8 output ports(out A, out' A, out B, out' B, out C, out' C, out D, out' D) and to the corresponding seven segment input ports (A,B,C,D,E,F,G,DP). We use a common cathode segment in this project which is in turn connected to the I/O ports (RB5, RB6, RB7).

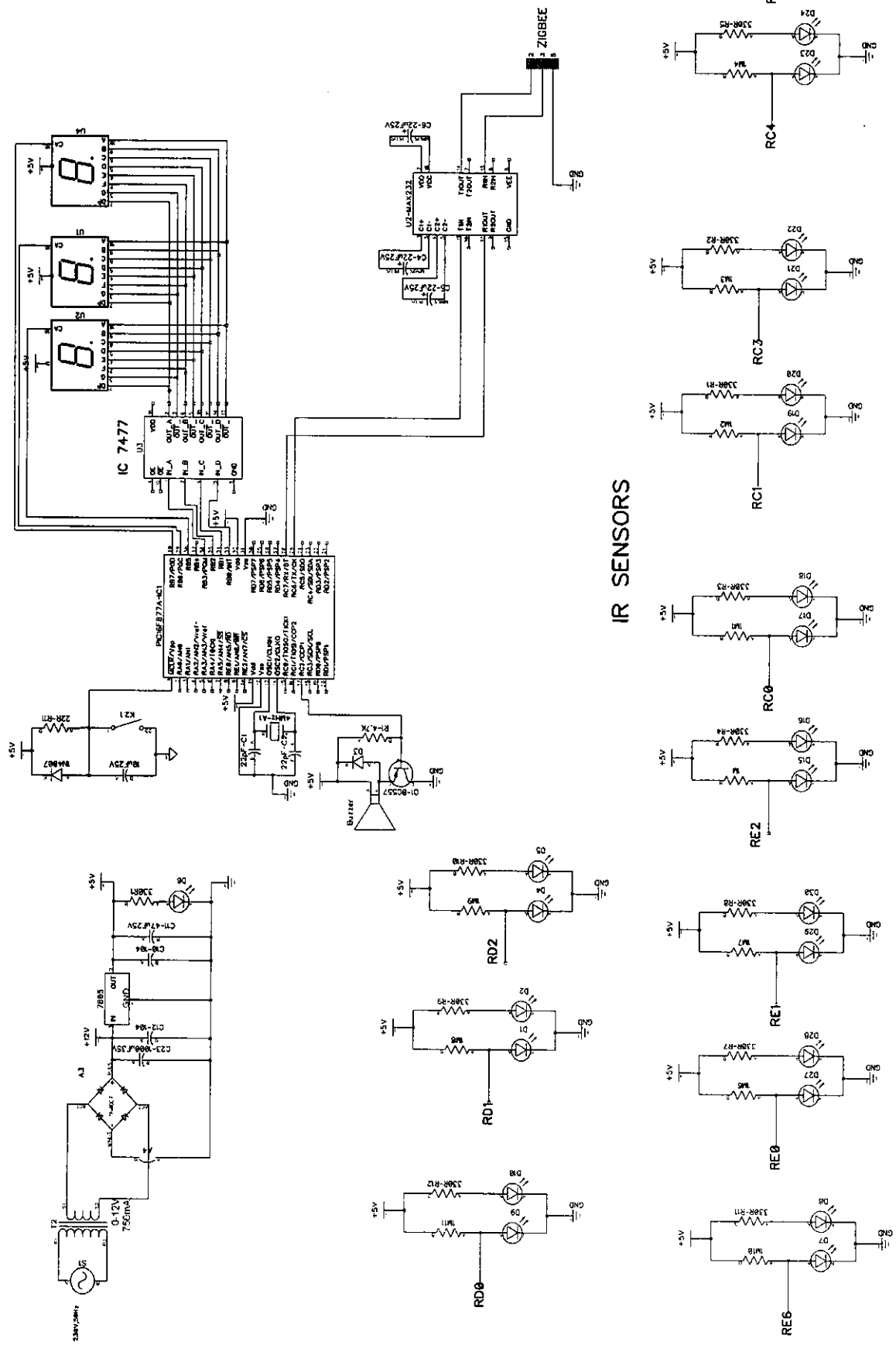
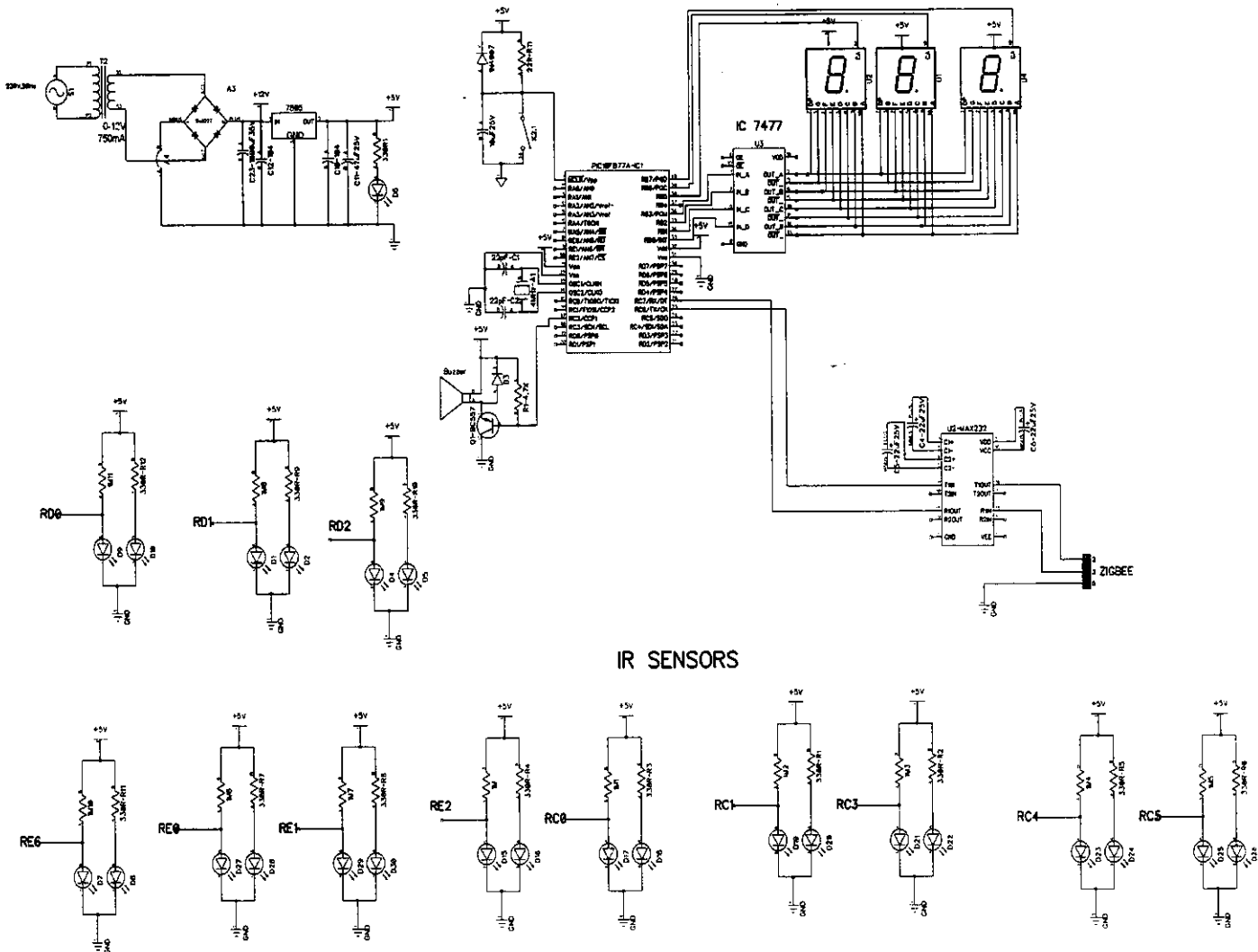


Fig 2.1 Controlling section

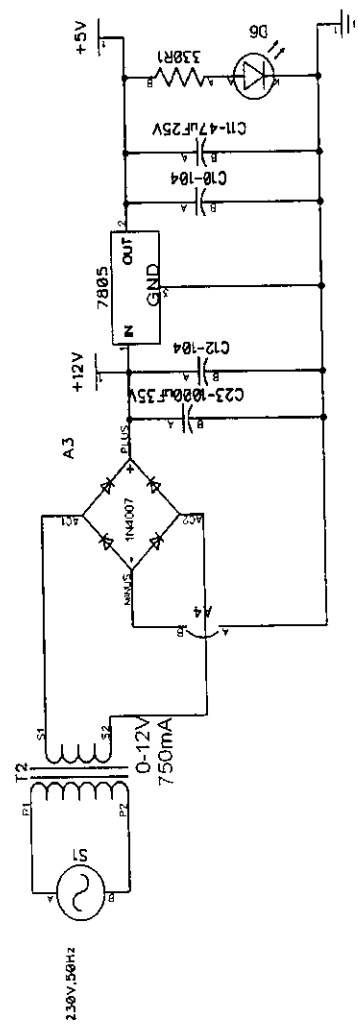
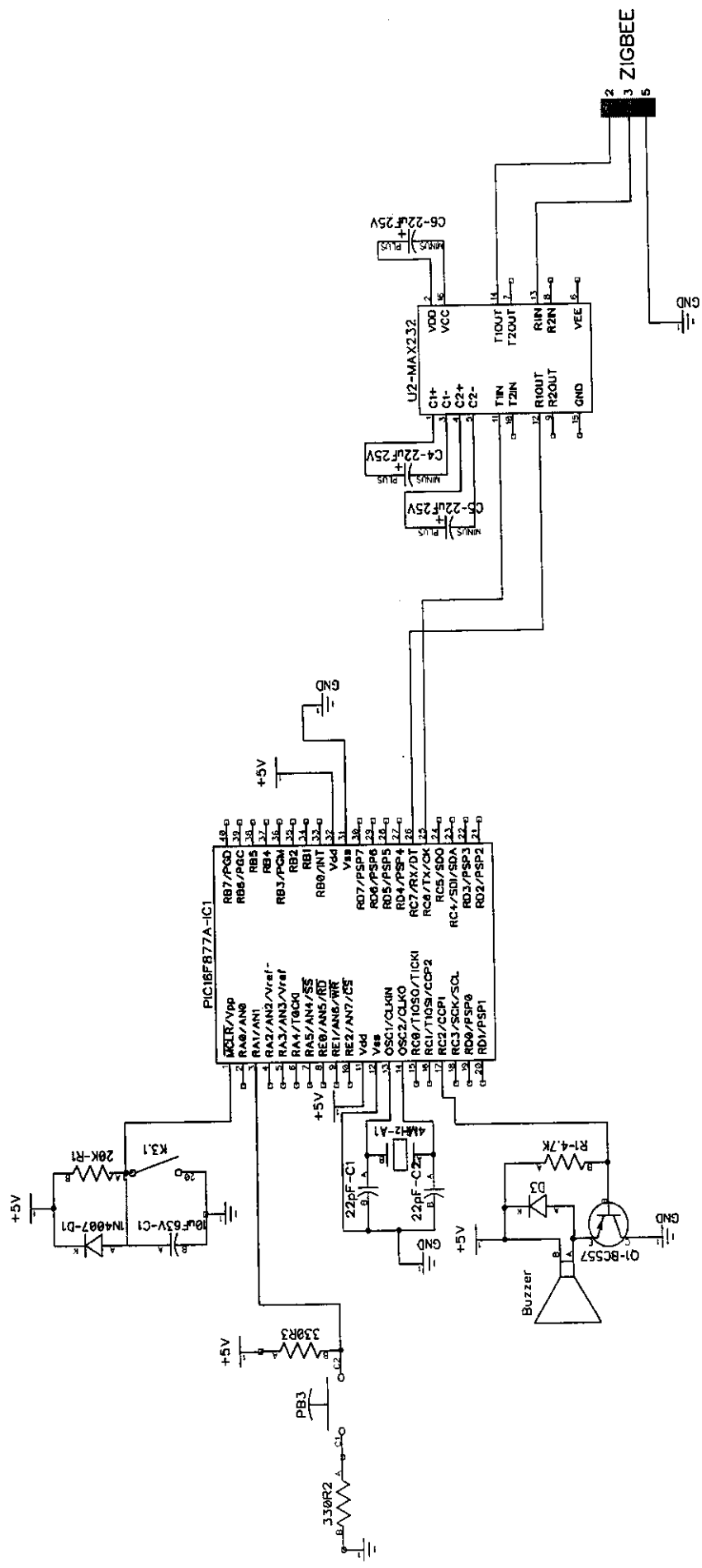


**Fig 2.1 Controlling section**

## **2.2 VEHICLE/AMBULANCE SECTION**

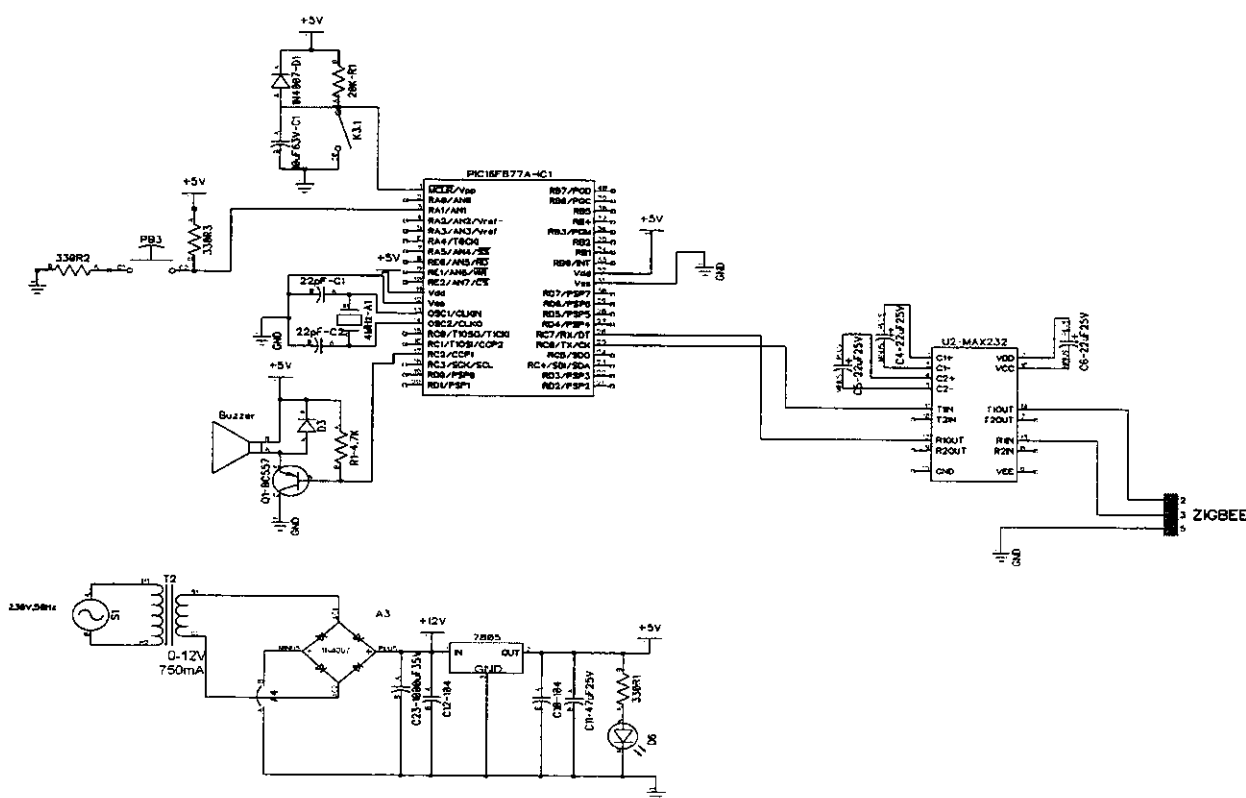
The Fig 2.2 shows the Zigbee section. The zigbee protocol is used to provide preference to the emergency vehicles such as the ambulance, fire engine and police vehicles. The zigbee controller is interfaced with the microcontroller through a MAX 232 IC. This IC is used to convert TTL/CMOS signals in the range of -3 to -25V or 3 to 25V





to the desired voltage of 5V used for serial communication in the microcontroller and vice versa.

The input port (T1IN) of the MAX232 IC is connected to the (RC6) port of the microcontroller and the output port (R1OUT) of the MAX232 IC is connected to the (RC7). Likewise the I/O ports (T1OUT and R1IN) of the MAX232 IC are in turn connected to the Zigbee module. Thus through these connections the zigbee sends a signal to the microcontroller which is considered as an emergency and then the microcontroller immediately provides preference to the direction from which the zigbee signal is obtained, and then turns the signal light on that road to green.



**Fig 2.2 Vehicle/Ambulance section**

## CHAPTER 3

### HARDWARE DESCRIPTION

The main hardware components used in this project are:-

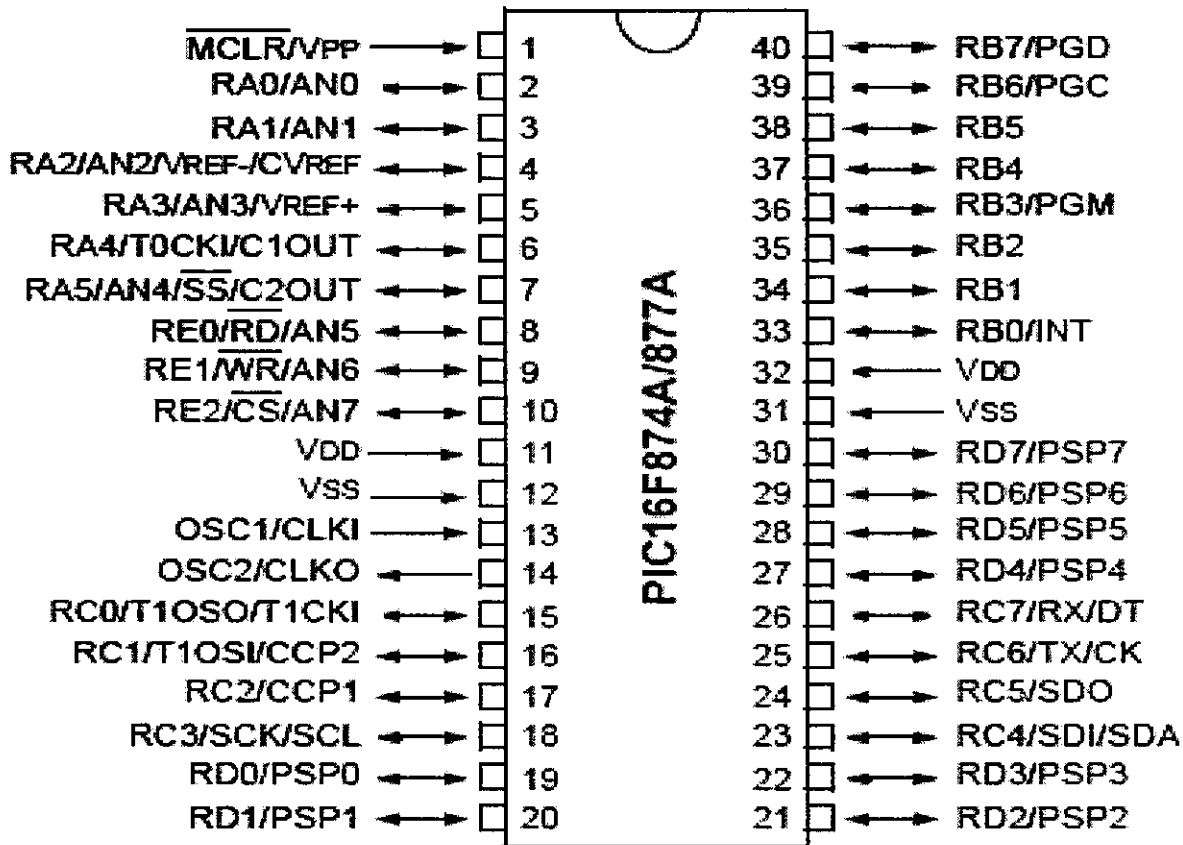
1. PIC16F877A Microcontroller
2. IC 7447
3. External Crystal Oscillator (4MHz)
4. IR Sensors
5. Seven Segment Display
6. MAX 232 IC
7. Zigbee

#### **3.1 PIC 16F877A MICROCONTROLLER**

PIC is a family of Harvard architecture microcontrollers made by Microchip Technology, derived from the PIC1640 originally developed by General Instrument's Microelectronics Division. The name PIC initially referred to "Peripheral Interface Controller". PICs are popular with both industrial developers and hobbyists alike due to their low cost, wide availability, large user base, extensive collection of application notes, availability of low cost or free development tools, and serial programming (and re-programming with flash memory) capability.

The powerful (200nanosecond instruction execution) yet easy-to-program (only 35 single word instructions) CMOS FLASH-based 8-bit *PIC 16F877A* microcontroller packs Microchip's powerful PIC architecture into an 40- or 44-pin package and is upwards compatible with the PIC16C5X, PIC12CXXX and PIC16C7X devices. The PIC16F877A features 256 bytes of EEPROM data memory, self programming, an ICD, 2 Comparators, 8 channels of 10-bit Analog-to-Digital (A/D) converter, 2 capture/compare/PWM

functions, the synchronous serial port can be configured as either 3-wire SPI or the 2-wire I<sup>2</sup>C bus and as USART. All of these features make it ideal for more advanced level A/D applications in automotive, industrial, appliances and consumer applications.



**Fig 3.1 Pin details of PIC 16F877A Microcontroller**

### 3.1.1 HIGH-PERFORMANCE RISC (Reduced Instruction Set Computing) CPU

- Only 35 single-word instructions to learn
- All single-cycle instructions except for program branches, which are two-cycle
- Operating speed: DC – 20 MHz clock input DC – 200 ns instruction cycle
- Up to 8K × 14 words of Flash Program Memory, Up to 368 × 8 bytes of Data Memory (RAM),

Up to 256 × 8 bytes of EEPROM Data Memory

- Pin out compatible to other 28-pin or 40/44-pin PIC16CXXX and PIC16FXXX microcontrollers.

### 3.1.2 PERIPHERAL FEATURES

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during Sleep via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, maximum resolution is 12.5 ns
  - Compare is 16-bit, maximum resolution is 200 ns
  - PWM maximum resolution is 10-bit
- Synchronous Serial Port (SSP) with SPI™  
(Master mode) and I2C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver  
Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) – 8 bits wide with external RD, WR and CS controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)



### 3.1.3 ANALOG FEATURES

- 10-bit, up to 8-channel Analog-to-Digital Converter (A/D)

- Brown-out Reset (BOR)
- Analog Comparator module with:
  - Two analog comparators
  - Programmable on-chip voltage reference (VREF) module
  - Programmable input multiplexing from device inputs and internal voltage reference
  - Comparator outputs are externally accessible

### 3.1.4 SPECIAL FEATURES

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

### 3.2 BCD to Seven segment Decoder (IC7447)

The IC7447 is a BCD to 7-segment decoder. Here in this case, the IC7447 takes the Binary Coded Decimal (BCD) as the input and outputs the relevant 7 segment code. The I/O ports RB of the microcontroller are connected to the seven segment display as shown in the fig 3.2. The number required to display is sent as the lower nibble of the output port of the Microcontroller. The 7447 converts the four input bits (BCD) to their corresponding 7-segment codes. The outputs of the 7447 are connected to the 7-segment display.

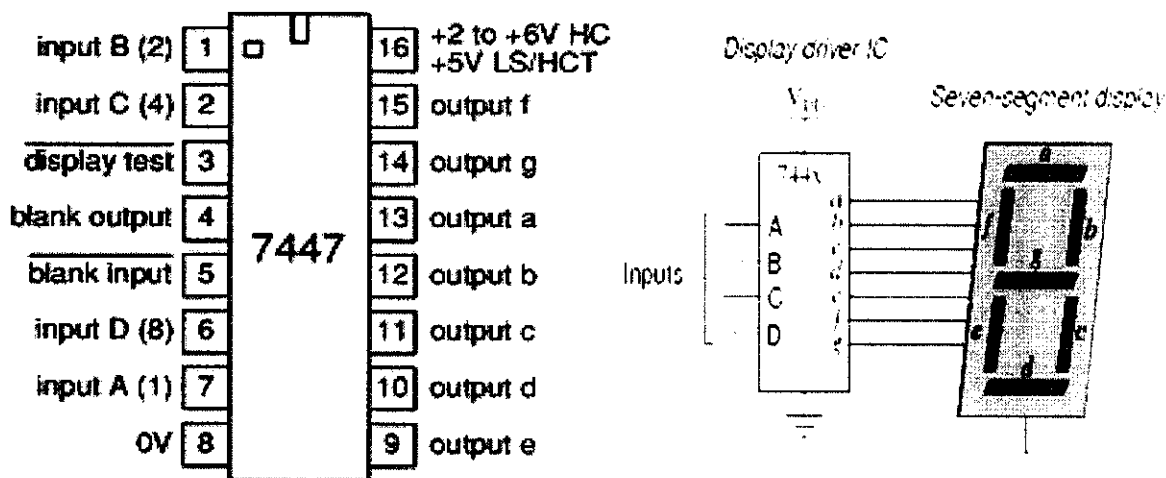


Fig 3.2 Pin configuration and interfacing of 7447 with the seven segment LED

Tab 3.1 IC 7447 Pin description

Pin Number	Description
1	BCD B Input
2	BCD C Input
3	Lamp Test
4	RB Output
5	RB Input
6	BCD D Input
7	BCD A Input
8	Ground
9	7-Segment e Output

10	7-Segment d Output
11	7-Segment c Output
12	7-Segment b Output
13	7-Segment a Output
14	7-Segment g Output
15	7-Segment f Output
16	Positive Supply

### 3.3 MAX 232 IC

The pin details of MAX 232 IC are shown in Table 3.2. The driver section consists of the pins 11 and 14 and the receiver section used comprises of the pins 13 and 12. The MAX232 is an integrated circuit that converts signals from an RS-232 serial port to signals suitable for use in TTL compatible digital logic circuits. The receivers reduce RS-232 inputs (which may be as high as  $\pm 25$  V), to standard 5 V TTL levels.

It is helpful to understand what occurs to the voltage levels. When a MAX232 IC receives a TTL level to convert, it changes a TTL Logic 0 to between +3 and +15 V, and changes TTL Logic 1 to between -3 to -15 V, and vice versa for converting from RS232 to TTL.

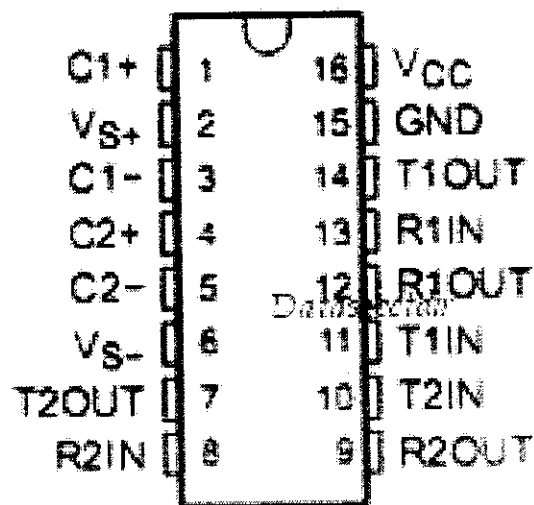


Fig 3.3 Pin diagram on MAX232 IC



Tab 3.2 Pin details of MAX 232 IC

Pin No	Function	Name
1	Capacitor connection pins	Capacitor 1 +
2		Capacitor 3 +
3		Capacitor 1 -
4		Capacitor 2 +
5		Capacitor 2 -
6		Capacitor 4 -
7 and 14	Output pin; outputs the serially transmitted data at RS232 logic level; connected to receiver pin of PC serial port	T2 Out and T1out
8 and 13	Input pin; receives serially transmitted data at RS 232 logic level; connected to transmitter pin of PC serial port	R2 In and R1 In
9 and 12	Output pin; outputs the serially transmitted data at TTL logic level; connected to receiver pin of controller.	R2 Out and R1 out
10 and 11	Input pins; receive the serial data at TTL logic level; connected to serial transmitter pin of controller.	T2 In and T1 In
15	Ground (0V)	Ground
16	Supply voltage; 5V (4.5V – 5.5V)	Vcc

### 3.4 Zigbee

ZigBee is a wireless technology developed as an open global standard to address the unique needs of low-cost, low-power wireless M2M networks. The ZigBee standard operates on the IEEE 802.15.4 physical radio specification and operates in unlicensed bands including 2.4 GHz, 900 MHz and 868 MHz. The ZigBee protocol is designed to communicate data through hostile RF environments that are common in commercial and industrial applications.

ZigBee protocols are intended for use in embedded applications requiring low data rates and low power consumption. ZigBee's current focus is to define a general-purpose,

inexpensive, self-organizing mesh network that can be used for industrial control, embedded sensing, medical data collection, smoke and intruder warning, building automation, home automation, etc. The resulting network will use very small amounts of power — individual devices must have a battery life of at least two years to pass ZigBee certification.

Typical application areas include.

- Home Entertainment and Control — Smart lighting, advanced temperature control, safety and security, movies and music
- *Wireless Sensor Networks'* — Starting with individual sensors like Telosb/Tmote and Iris from Memsic.

### **3.5 External Crystal Oscillator**

A crystal oscillator is an electronic oscillator circuit that uses the mechanical resonance of a vibrating crystal of piezoelectric material to create an electrical signal with a very precise frequency. This frequency is commonly used to keep track of time (as in quartz wristwatches), to provide a stable clock signal for digital integrated circuits, and to stabilize frequencies for radio transmitters and receivers. The most common type of piezoelectric resonator used is the quartz crystal, so oscillator circuits designed around them became known as "crystal oscillators."

In this project, there is an external crystal oscillator used in spite of an internal RC oscillator in PIC16F877A to keep track of time and provide a steady clock signal, since the internal RC oscillator has very low accuracy and it is usually sensitive to EMI and vibrations.

## CHAPTER 4

### SOFTWARE DESCRIPTION

#### 4.1 PROGRAM DESCRIPTION

The programming language used in project is Embedded C.

To start with, the codes #byte TRISA, TRISB to TRISE indicates the tri state registers used to specify whether the ports from A to E are input or output ports.

#byte PORTA=0x05 indicates that port A has the register value as 0x05. similarly it applies for all the other ports as well from port A to port to E.

The port numbering is as follows.

Port A is assigned to number 5

Port B is assigned to number 6

Port C is assigned to number 7

Port D is assigned to number 8

Port E is assigned to number 9.

The next section is for initialization of the LEDs in all the four junctions.

#bit East LED R=0x07.0 indicates the red color led placed in the eastern road and the register value 0x07.0 specifies that it is connected to 0<sup>th</sup> pin in the port C (number 7). Similarly it goes for the LEDs yellow and green as well.

The three LEDs in the south directions are connected to the pins 3 to 5 in the port C.

Likewise the three LEDs, red, yellow and green in the west and north directions are connected to the pins 0 to 2 in the ports 8 and 9 respectively.

Now coming to the IR section, first we have #bit East\_IR\_1=0x08.6 which indicates one of the three IRs placed in the eastern road connected to the port D(number 8) in the 6<sup>th</sup> pin.

Similarly the this applies to the sections south and west having the three IRs connected to the port A(number 5) to the pins 0 to 5.

Last comes the north section having their IRs connected to the port D( number 8) from pins 0 to 3.

#### **4.2 ALGORITHM 1:**

STEP 1: Initialize the process.

STEP 2: The control is given to the microcontroller which manages uniform signal cycle.

STEP 3: Incase of any alert from zigbee, and then the interrupt is given.

STEP 4: The lane from which the zigbee alert comes is given green.

STEP 5: ELSE, the traffic signal continues in the normal cycle with the time variation handled by the microcontroller.

#### **4.3 ALGORITHM 2:**

STEP 1: Initiate traffic signal cycle.

STEP 2: The green light starts with the eastern road( A).

STEP 3: Let the ambulance come from the road C.

if yes,

STEP 4: Road C gets green

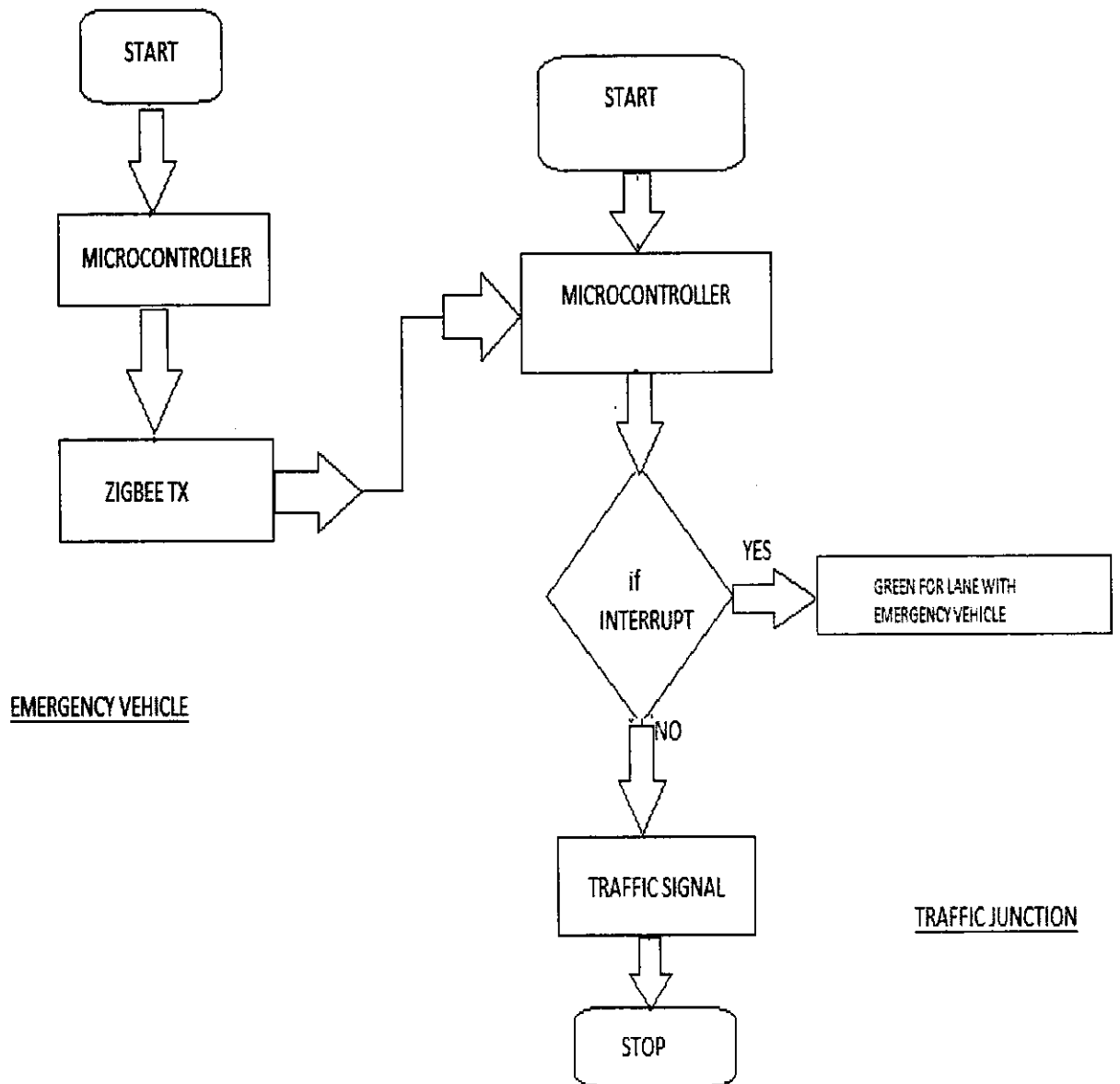
else,

STEP 5: if A= green,

then B=C=D= red.

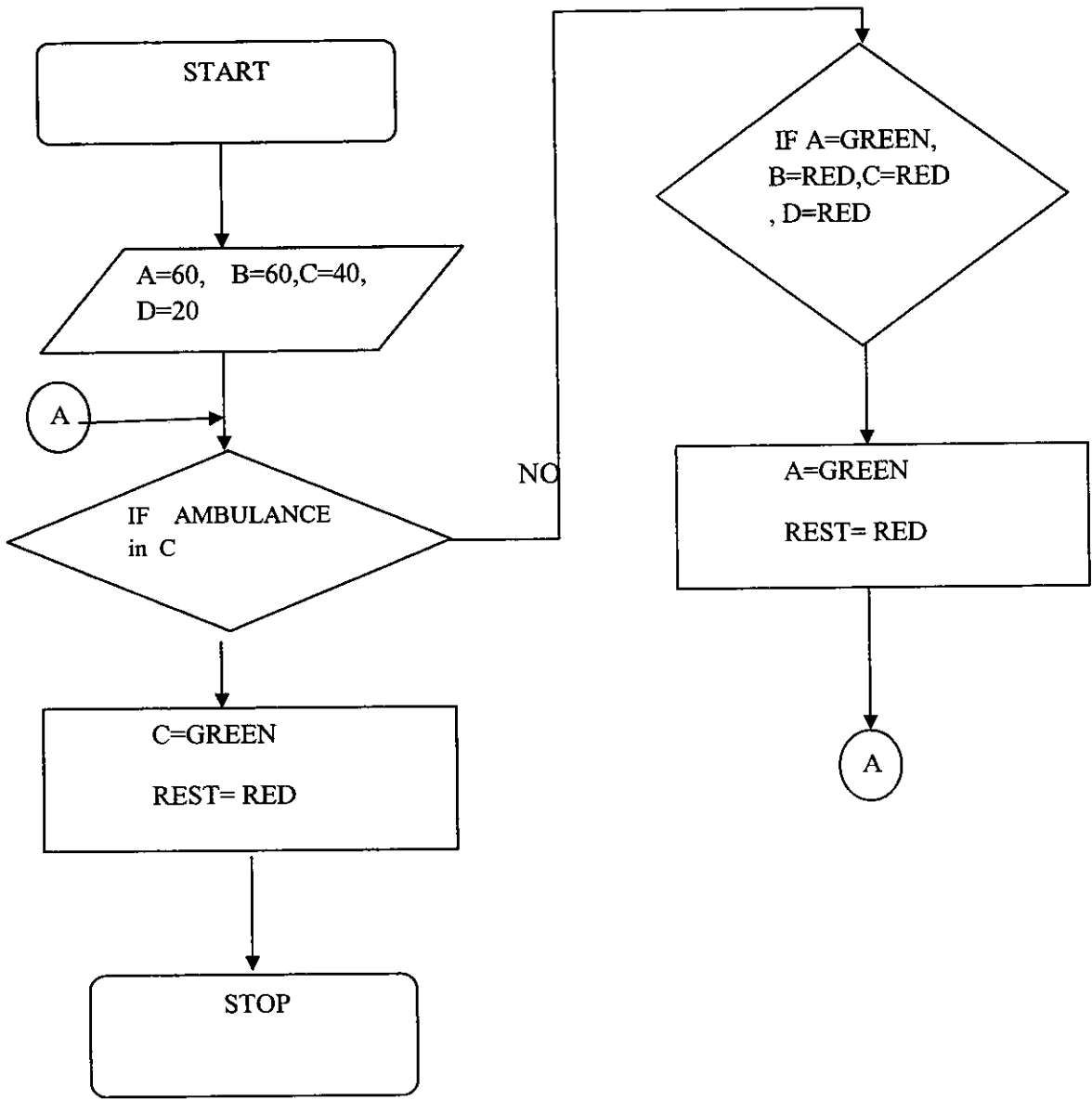
STEP 6: Pointer goes back to checking for the ambulance

STEP 7: Stop

**FLOW CHART 1**

**FLOW CHART 2**

This flow diagram is for traffic signal on one road, where road A is 'GREEN' and the other roads are 'RED' and the other conditions will vary accordingly.



## **CONCLUSION**

The objective of the project has been successfully achieved by analyzing the density of vehicles on each road in a traffic signal junction with the help of infrared sensors which is interfaced with a PIC16f877A microcontroller and there by allocating varied time intervals for each road based on the density. Also a zigbee protocol is interfaced with the microcontroller in order to provide unquestionable preference to emergency vehicles such as ambulance, fire engines, police cars etc. Thus, the human involvement in traffic signal control is minimized and certain over all benefits is provided to the public in the form of time and fuel saving, not to mention the fact that at most importance to the human life is ensured.

## **ANNEXURE**



```
#include "E:\pgms\real pic\beta\traffic light controller\main.h"
```

```
#include <lcd.c>
```

```
#byte TRISA=0x85
```

```
#byte TRISB=0x86
```

```
#byte TRISC=0x87
```

```
#byte TRISD=0x88
```

```
#byte TRISE=0x89
```

```
#byte PORTA=0x05
```

```
#byte PORTB=0x06
```

```
#byte PORTC=0x07
```

```
#byte PORTD=0x08
```

```
#byte PORTE=0x09
```

```
#bit East_led_R=0x07.0
```

```
#bit East_led_Y=0x07.1
```

```
#bit East_led_G=0x07.2
```

#bit South\_led\_R=0x07.3

#bit South\_led\_Y=0x07.4

#bit South\_led\_G=0x07.5

#bit West\_led\_R=0x08.0

#bit West\_led\_Y=0x08.2

#bit West\_led\_G=0x08.1

#bit North\_led\_R=0x09.0

#bit North\_led\_Y=0x09.1

#bit North\_led\_G=0x09.2

#bit East\_IR\_1=0x08.6

#bit East\_IR\_2=0x08.7

#bit East\_IR\_3=0x06.7//THIS IS NOT CONNECTED AS IF NOW

#bit South\_IR\_1=0x05.0

#bit South\_IR\_2=0x05.1

#bit South\_IR\_3=0x05.2

#bit West\_IR\_1=0x05.3

#bit West\_IR\_2=0x05.4

#bit West\_IR\_3=0x05.5

#bit North\_IR\_1=0x08.3

#bit North\_IR\_2=0x08.4

#bit North\_IR\_3=0x08.5

#bit Seg1Bit = 0x06.4

#bit Seg2Bit = 0x06.5

#bit Seg3Bit = 0x06.6

#byte INTCON=0x0B

#byte OPTION\_REG = 0x81

#byte TXSTA=0x98

#bit TMR0IE =0x0B.5

#bit TMR0IF = 0x0B.2

#byte TMR0=0x01

#bit INTE = 0x0B.4

#bit INTF = 0x0B.1

#bit GIE = 0x0B.7

#bit PEIE = 0x0B.6

int8 East\_IRFlag=0,West\_IRFlag=0,North\_IRFlag=0,South\_IRFlag=0;

int8 i=0,j=0,E=1,W=1,N=1,S=1;

int8 Asciiarr[5];

```
int8  
RpmReady,SegDataArr[10]={0xE0,0xF0,0xE8,0xF8,0xE4,0xF4,0xEC,0  
xFc,0xE2,0xF2};
```

```
int16 sec,TimerCunt;
```

```
//=====
```

```
void Dec2Ascii(int16 decval)
```

```
{
```

```
  Asciiarr[0]=(decval/1000);
```

```
  decval=decval%1000;
```

```
  Asciiarr[1]=(decval/100);
```

```
  decval=decval%100;
```

```
  Asciiarr[2]=(decval/10);
```

```
  decval=decval%10;
```

```
  Asciiarr[3]=decval;
```

```
}
```

```
//=====
```

```
#INT_TIMER0
```

```
void timer_isr()
```

```
{
```

```
  TimerCunt++;
```

```
  if(TimerCunt==16)
```

```
  {
```

```
    Sec--;
```

```
TimerCunt=0;
}
if(sec==0)
{
RpmReady=1;
sec=0;
TMR0IE =0;
INTE =0;
}

TMR0IF=0;

}

//=====
=====

void TimerInit()
{
TXSTA=0x20;

OPTION_REG = 0xC7; //enable portb full disable & enable rising
edge
INTCON = 0xf0; //enable GIE,INTE,INTF=0
GIE=1;
PEIE=1;
```

```
TMR0IE =1;
TMR0IF =0;
TMR0=0x0c;
INTE =0;
INTF =0;
RpmReady=0;
TimerCunt=0;
TRISB = 0x00;
}
//=====
=====

Void SegmentFunc()
{
    PortB=SegDataArr[Asciiarr[1]];
    Seg1Bit =0;
    Seg2Bit =1;
    Seg3Bit =1;
    delay_ms(4);
    PortB=SegDataArr[Asciiarr[2]];
    Seg1Bit =1;
    Seg2Bit =0;
    Seg3Bit =1;
    delay_ms(4);
    PortB=SegDataArr[Asciiarr[3]];
    Seg1Bit =1;
    Seg2Bit =1;
    Seg3Bit =0;
```

```

    delay_ms(4);
}

//=====
void segment(int8 delay)
{
    Seg1Bit =1;
    Seg2Bit =1;
    Seg3Bit =1;
    TimerInit();
    sec=(delay);
    while(rpmready==0)//this function is infnite pls check for whole
    correction
    {
        Dec2Ascii(Sec);
        SegmentFunc();
        lcd_putc("\fTO SEGMNT ");
        lcd_putc(asciiarr[1]+48);
        lcd_putc(asciiarr[2]+48);
        lcd_putc(asciiarr[3]+48);
        lcd_putc("\nDELAY ");
        lcd_putc(e+48);
        lcd_putc(s+48);
        lcd_putc(w+48);
        lcd_putc(n+48);
    }
}

```

```
//=====
//=====
int East_density()
{
    East_IRFlag=0;
    if(East_ir_1==1)
        East_IRFlag+=5;
    if(East_ir_2==1)
        East_IRFlag+=5;
    if(East_ir_3==1)
        East_IRFlag+=5;
    if(!East_IR_1&&!East_IR_2&&!East_IR_3)
        East_IRFlag=5;
    return(East_IRFlag);
}
int West_density()
{
    West_IRFlag=0;
    if(West_ir_1==1)
        West_IRFlag+=5;
    if(West_IR_2==1)
        West_IRFlag+=5;
    if(West_ir_3==1)
        West_IRFlag+=5;
    if(!West_IR_1&&!West_IR_2&&!West_IR_3)
        West_IRFlag=5;
    return(West_IRFlag);
}
```



```
}  
int North_density()  
{  
    North_IRFlag=0;  
    if(North_ir_1==1)  
        North_IRFlag+=5;  
    if(North_ir_2==1)  
        North_IRFlag+=5;  
    if(North_ir_3==1)  
        North_IRFlag+=5;  
    if(!North_IR_1&&!North_IR_2&&!North_IR_3)  
        North_IRFlag=5;  
    return(North_IRFlag);  
  
}  
int South_density()  
{  
    South_IRFlag=0;  
    if(South_ir_1==1)  
        South_IRFlag+=5;  
    if(South_ir_2==1)  
        South_IRFlag+=5;  
    if(South_ir_3==1)  
        South_IRFlag+=5;  
    if(!South_IR_1&&!South_IR_2&&!South_IR_3)  
        South_IRFlag=5;  
    return(South_IRFlag);
```

```
    }  
void East_open()  
{  
    East_led_R=0;  
    East_led_G=1;  
}  
void West_open()  
{  
    West_led_R=0;  
    West_led_G=1;  
}  
void South_open()  
{  
    South_led_R=0;  
    South_led_G=1;  
}  
void North_open()  
{  
    North_led_R=0;  
    North_led_G=1;  
}  
void East_close()  
{  
    East_led_R=1;  
    East_led_G=0;  
}  
void West_close()
```

```
{
West_led_R=1;
west_led_G=0;
}
void North_close()
{
North_led_R=1;
North_led_G=0;
}
void South_close()
{
South_led_R=1;
South_led_G=0;
}
void go()
{
E=East_density();
East_open();
west_close();
north_close();
south_close();
//for(i=0;i<=e;i++)//change to e
//for(j=0;j<=50;j++)//org=50
{
segment(E);
}
S=South_density();
```

```
East_close();
West_close();
North_close();
south_open();
//for(i=0;i<=S;i++)//change to e
    //for(j=0;j<=50;j++)
        {
            segment(S);
        }
W=West_density();
East_close();
west_open();
north_close();
south_close();
//for(i=0;i<=W;i++)//change to e
    //for(j=0;j<=50;j++)
        {
            segment(W);
        }
N=North_density();
East_close();
west_close();
north_open();
south_close();
//for(i=0;i<=N;i++)//change to n
    //for(j=0;j<=50;j++)
        {
```

```
        segment(N);  
    }  
}
```

```
void main()
```

```
{
```

```
    TRISA=0x3f;
```

```
    TRISB=0x80;
```

```
    TRISC=0x00;
```

```
    TRISD=0xf8;
```

```
    TRISE=0x00;
```

```
    PORTA=0x00;
```

```
    PORTB=0x00;
```

```
    PORTC=0x00;
```

```
    PORTD=0x00;
```

```
    PORTE=0x00;
```

```
    lcd_init();
```

```
    delay_ms(1000);
```

```
    while(true)
```

```
    {
```

```
        go();
```

```
    }
```

```
}
```

## **APPENDIX**



**MICROCHIP**

---

**PIC16F87X  
Data Sheet**

**28/40-Pin 8-Bit CMOS FLASH  
Microcontrollers**



MICROCHIP

# PIC16F87X

## 28/40-Pin 8-Bit CMOS FLASH Microcontrollers

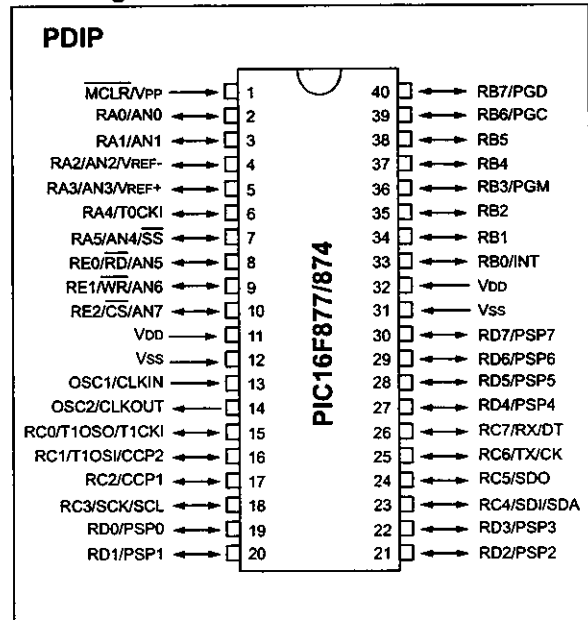
### Devices Included in this Data Sheet:

- PIC16F873
- PIC16F876
- PIC16F874
- PIC16F877

### Microcontroller Core Features:

- High performance RISC CPU
- Only 35 single word instructions to learn
- All single cycle instructions except for program branches which are two cycle
- Operating speed: DC - 20 MHz clock input  
DC - 200 ns instruction cycle
- Up to 8K x 14 words of FLASH Program Memory,  
Up to 368 x 8 bytes of Data Memory (RAM)  
Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to the PIC16C73B/74B/76/77
- Interrupt capability (up to 14 sources)
- Eight level deep hardware stack
- Direct, indirect and relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and  
Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC  
oscillator for reliable operation
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low power, high speed CMOS FLASH/EEPROM  
technology
- Fully static design
- In-Circuit Serial Programming™ (ICSP) via two  
pins
- Single 5V In-Circuit Serial Programming capability
- In-Circuit Debugging via two pins
- Processor read/write access to program memory
- Wide operating voltage range: 2.0V to 5.5V
- High Sink/Source Current: 25 mA
- Commercial, Industrial and Extended temperature  
ranges
- Low-power consumption:
  - < 0.6 mA typical @ 3V, 4 MHz
  - 20 µA typical @ 3V, 32 kHz
  - < 1 µA typical standby current

### Pin Diagram



### Peripheral Features:

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

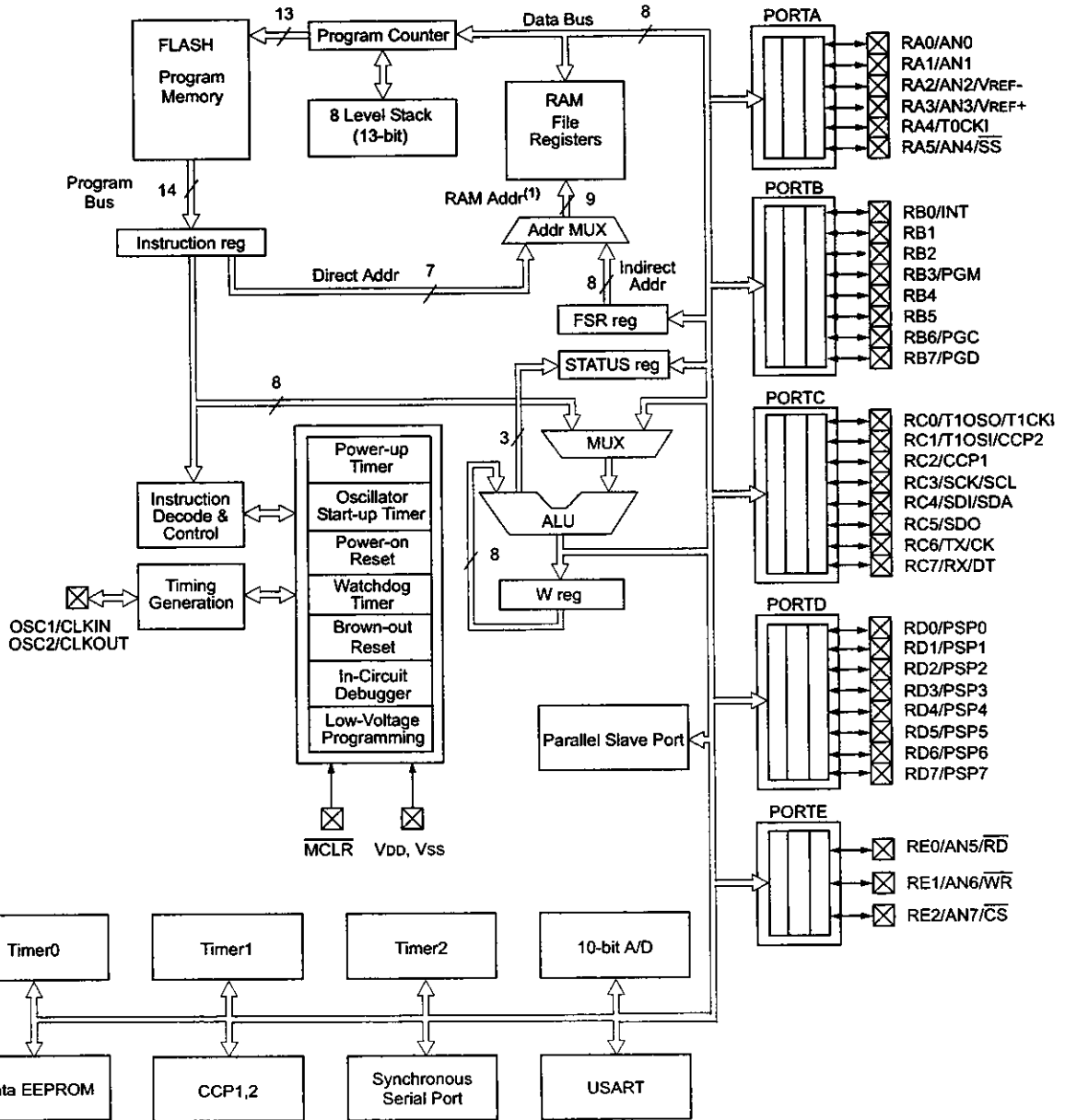


<b>Key Features PICmicro™ Mid-Range Reference Manual (DS33023)</b>	<b>PIC16F873</b>	<b>PIC16F874</b>	<b>PIC16F876</b>	<b>PIC16F877</b>
Operating Frequency	DC - 20 MHz	DC - 20 MHz	DC - 20 MHz	DC - 20 MHz
RESETS (and Delays)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)
FLASH Program Memory (14-bit words)	4K	4K	8K	8K
Data Memory (bytes)	192	192	368	368
EEPROM Data Memory	128	128	256	256
Interrupts	13	14	13	14
I/O Ports	Ports A,B,C	Ports A,B,C,D,E	Ports A,B,C	Ports A,B,C,D,E
Timers	3	3	3	3
Capture/Compare/PWM Modules	2	2	2	2
Serial Communications	MSSP, USART	MSSP, USART	MSSP, USART	MSSP, USART
Parallel Communications	—	PSP	—	PSP
10-bit Analog-to-Digital Module	5 input channels	8 input channels	5 input channels	8 input channels
Instruction Set	35 instructions	35 instructions	35 instructions	35 instructions

# PIC16F87X

FIGURE 1-2: PIC16F874 AND PIC16F877 BLOCK DIAGRAM

Device	Program FLASH	Data Memory	Data EEPROM
PIC16F874	4K	192 Bytes	128 Bytes
PIC16F877	8K	368 Bytes	256 Bytes



Note 1: Higher order bits are from the STATUS register.

# PIC16F87X

**TABLE 1-2: PIC16F874 AND PIC16F877 PINOUT DESCRIPTION**

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	13	14	30	I	ST/CMOS <sup>(4)</sup>	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	14	15	31	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
MCLR/VPP	1	2	18	I/P	ST	Master Clear (Reset) input or programming voltage input. This pin is an active low RESET to the device.
RA0/AN0	2	3	19	I/O	TTL	PORTA is a bi-directional I/O port. RA0 can also be analog input0. RA1 can also be analog input1. RA2 can also be analog input2 or negative analog reference voltage. RA3 can also be analog input3 or positive analog reference voltage. RA4 can also be the clock input to the Timer0 timer/counter. Output is open drain type. RA5 can also be analog input4 or the slave select for the synchronous serial port.
RA1/AN1	3	4	20	I/O	TTL	
RA2/AN2/VREF-	4	5	21	I/O	TTL	
RA3/AN3/VREF+	5	6	22	I/O	TTL	
RA4/T0CKI	6	7	23	I/O	ST	
RA5/SS/AN4	7	8	24	I/O	TTL	
RB0/INT	33	36	8	I/O	TTL/ST <sup>(1)</sup>	PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs. RB0 can also be the external interrupt pin.  RB3 can also be the low voltage programming input. Interrupt-on-change pin. Interrupt-on-change pin. Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming clock. Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming data.
RB1	34	37	9	I/O	TTL	
RB2	35	38	10	I/O	TTL	
RB3/PGM	36	39	11	I/O	TTL	
RB4	37	41	14	I/O	TTL	
RB5	38	42	15	I/O	TTL	
RB6/PGC	39	43	16	I/O	TTL/ST <sup>(2)</sup>	
RB7/PGD	40	44	17	I/O	TTL/ST <sup>(2)</sup>	

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

- Note 1:** This buffer is a Schmitt Trigger input when configured as an external interrupt.  
**Note 2:** This buffer is a Schmitt Trigger input when used in Serial Programming mode.  
**Note 3:** This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).  
**Note 4:** This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

**TABLE 1-2: PIC16F874 AND PIC16F877 PINOUT DESCRIPTION (CONTINUED)**

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
RC0/T1OSO/T1CKI	15	16	32	I/O	ST	PORTC is a bi-directional I/O port. RC0 can also be the Timer1 oscillator output or a Timer1 clock input. RC1 can also be the Timer1 oscillator input or Capture2 input/Compare2 output/PWM2 output. RC2 can also be the Capture1 input/Compare1 output/PWM1 output. RC3 can also be the synchronous serial clock input/output for both SPI and I <sup>2</sup> C modes. RC4 can also be the SPI Data In (SPI mode) or data I/O (I <sup>2</sup> C mode). RC5 can also be the SPI Data Out (SPI mode). RC6 can also be the USART Asynchronous Transmit or Synchronous Clock. RC7 can also be the USART Asynchronous Receive or Synchronous Data.
RC1/T1OSI/CCP2	16	18	35	I/O	ST	
RC2/CCP1	17	19	36	I/O	ST	
RC3/SCK/SCL	18	20	37	I/O	ST	
RC4/SDI/SDA	23	25	42	I/O	ST	
RC5/SDO	24	26	43	I/O	ST	
RC6/TX/CK	25	27	44	I/O	ST	
RC7/RX/DT	26	29	1	I/O	ST	
RD0/PSP0	19	21	38	I/O	ST/TTL <sup>(3)</sup>	PORTD is a bi-directional I/O port or parallel slave port when interfacing to a microprocessor bus.
RD1/PSP1	20	22	39	I/O	ST/TTL <sup>(3)</sup>	
RD2/PSP2	21	23	40	I/O	ST/TTL <sup>(3)</sup>	
RD3/PSP3	22	24	41	I/O	ST/TTL <sup>(3)</sup>	
RD4/PSP4	27	30	2	I/O	ST/TTL <sup>(3)</sup>	
RD5/PSP5	28	31	3	I/O	ST/TTL <sup>(3)</sup>	
RD6/PSP6	29	32	4	I/O	ST/TTL <sup>(3)</sup>	
RD7/PSP7	30	33	5	I/O	ST/TTL <sup>(3)</sup>	
RE0/ $\overline{RD}$ /AN5	8	9	25	I/O	ST/TTL <sup>(3)</sup>	PORTE is a bi-directional I/O port. RE0 can also be read control for the parallel slave port, or analog input5. RE1 can also be write control for the parallel slave port, or analog input6. RE2 can also be select control for the parallel slave port, or analog input7.
RE1/ $\overline{WR}$ /AN6	9	10	26	I/O	ST/TTL <sup>(3)</sup>	
RE2/ $\overline{CS}$ /AN7	10	11	27	I/O	ST/TTL <sup>(3)</sup>	
Vss	12,31	13,34	6,29	P	—	Ground reference for logic and I/O pins.
VDD	11,32	12,35	7,28	P	—	Positive supply for logic and I/O pins.
NC	—	1,17,28,40	12,13,33,34		—	These pins are not internally connected. These pins should be left unconnected.

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

- Note 1:** This buffer is a Schmitt Trigger input when configured as an external interrupt.  
**Note 2:** This buffer is a Schmitt Trigger input when used in Serial Programming mode.  
**Note 3:** This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).  
**Note 4:** This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

## 2.0 MEMORY ORGANIZATION

There are three memory blocks in each of the PIC16F87X MCUs. The Program Memory and Data Memory have separate buses so that concurrent access can occur and is detailed in this section. The EEPROM data memory block is detailed in Section 4.0.

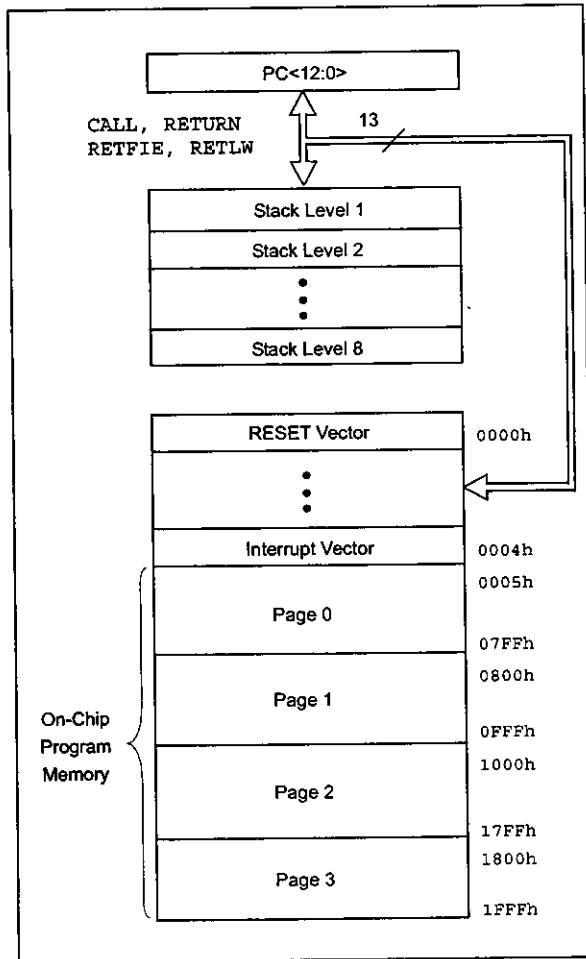
Additional information on device memory may be found in the PICmicro™ Mid-Range Reference Manual, (DS33023).

## 2.1 Program Memory Organization

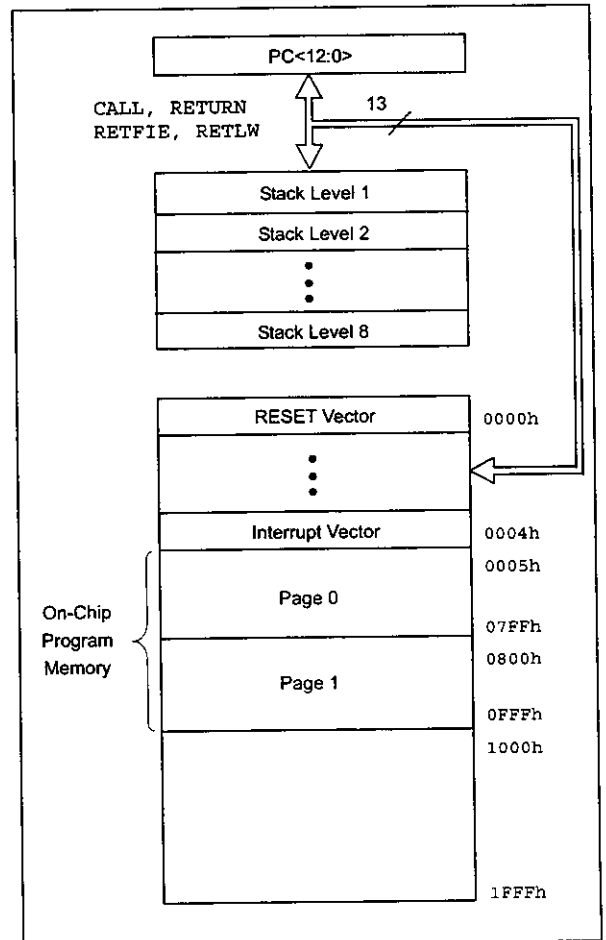
The PIC16F87X devices have a 13-bit program counter capable of addressing an 8K x 14 program memory space. The PIC16F877/876 devices have 8K x 14 words of FLASH program memory, and the PIC16F873/874 devices have 4K x 14. Accessing a location above the physically implemented address will cause a wraparound.

The RESET vector is at 0000h and the interrupt vector is at 0004h.

**FIGURE 2-1: PIC16F877/876 PROGRAM MEMORY MAP AND STACK**



**FIGURE 2-2: PIC16F874/873 PROGRAM MEMORY MAP AND STACK**



# PIC16F87X

---

## 2.2 Data Memory Organization

The data memory is partitioned into multiple banks which contain the General Purpose Registers and the Special Function Registers. Bits RP1 (STATUS<6>) and RP0 (STATUS<5>) are the bank select bits.

RP1:RP0	Bank
00	0
01	1
10	2
11	3

Each bank extends up to 7Fh (128 bytes). The lower locations of each bank are reserved for the Special Function Registers. Above the Special Function Registers are General Purpose Registers, implemented as static RAM. All implemented banks contain Special Function Registers. Some frequently used Special Function Registers from one bank may be mirrored in another bank for code reduction and quicker access.

<b>Note:</b> EEPROM Data Memory description can be found in Section 4.0 of this data sheet.
---

### 2.2.1 GENERAL PURPOSE REGISTER FILE

The register file can be accessed either directly, or indirectly through the File Select Register (FSR).

## REFERENCES

1. <http://www.commsdesign.com/showArticle.jhtml?articleID=192200323>  
ZigBee: Wireless Technology for Low-Power Sensor Networks
2. <http://www.futurlec.com/IC7447.html>
3. [www.easts.info/on-line/proceedings\\_05/1301.pdf](http://www.easts.info/on-line/proceedings_05/1301.pdf)
4. [www.wikipedia.com](http://www.wikipedia.com)
5. [www.EFY.com](http://www.EFY.com)
6. [www.microchip.com](http://www.microchip.com)