

# A PROJECT REPORT ON FUZZY BASED ROOM AUTOMAT

A Thesis Submitted In Partial Fulfillment of the  
for the Degree

of  
BACHELOR OF TECHNOLOGY  
in  
INSTRUMENTATION ENGINEER

by  
BISHWARAJ DHAR (Roll No. Gau-C-1  
BLIJIT RANJAN DAS (Roll No. Gau-C-  
BWHWIMA NARZARY (Roll No. Ga

Under the supervision  
**MR. JEET DUTTA**  
Asst. Professor  
Dept. of IE, CIT, KOKRAJHA



DEPARTMENT OF INSTRUMENTA  
केन्द्रीय प्रौद्योगिकी संस्थान  
CENTRAL INSTITUTE OF TECHN  
(A Centrally Funded Institute under Mi  
BODOLAND TERRITORIAL AREAS DISTRICT  
Website: [www.cit.kokrajha](http://www.cit.kokrajha)  
MAY 20

# A PROJECT REPORT ON

## FUZZY BASED ROOM AUTOMATION SYSTEM

*A Thesis Submitted In Partial Fulfillment of the Requirements*

*for the Degree*

*of*

**BACHELOR OF TECHNOLOGY**

**in**

**INSTRUMENTATION ENGINEERING**

**by**

**BISHWARAJ DHAR (Roll No.Gau-C-11/L-211)**

**BIJIT RANJAN DAS (Roll No. Gau-C-11/127)**

**BWHWIMA NARZARY (Roll No. Gau-C-11/79)**

Under the supervision of

**MR. JEET DUTTA**

Asst. Professor

Dept. of IE, CIT, KOKRAJHAR



**DEPARTMENT OF INSTRUMENTATION ENGINEERING**

**केन्द्रीय प्रौद्योगिकी संस्थान कोकराझार**

**CENTRAL INSTITUTE OF TECHNOLOGY KOKRAJHAR**

**(A Centrally Funded Institute under Ministry of HRD, Govt. of India)**

**BODOLAND TERRITORIAL AREAS DISTRICTS :: KOKRAJHAR :: ASSAM :: 783370**

**Website: [www.cit.kokrajhar.in](http://www.cit.kokrajhar.in), [www.cit.ac.in](http://www.cit.ac.in)**

**MAY 2015**



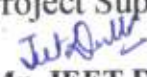
DEPARTMENT OF INSTRUMENTATION ENGINEERING  
केन्द्रीय प्रौद्योगिकी संस्थान कोकराझार  
CENTRAL INSTITUTE OF TECHNOLOGY, KOKRAJHAR  
(An Autonomous Institute under MHRD)  
Kokrajhar – 783370, BTAD, Assam, India

### CERTIFICATE OF PROJECT SUBMISSION

This is to certify that the work embodied in this project entitled “A FUZZY BASED ROOM AUTOMATION SYSTEM” submitted by Student **Bishwaraj Dhar, Bijit Ranjan Das** and **Bwhwima Narzary** to the Department of Instrumentation Engineering, is carried out under our direct supervisions and guidance.

The project work has been prepared as per the regulations of Central Institute of Technology and I strongly recommend that this project work be accepted in partial fulfillment of the requirement for the degree of B.Tech.

Project Supervisor

  
(Mr. JEET DUTTA)

HoD (i/c)

Department of IE, CIT Kokrajhar



**DEPARTMENT OF INSTRUMENTATION ENGINEERING**  
**केन्द्रीय प्रौद्योगिकी संस्थान कोकराझार**  
**CENTRAL INSTITUTE OF TECHNOLOGY, KOKRAJHAR**  
*(An Autonomous Institute under MHRD)*  
Kokrajhar – 783370, BTAD, Assam, India

**CERTIFICATE BY THE BOARD OF EXAMINERS**

This is to certify that the project work entitled “**A FUZZY BASED ROOM AUTOMATION SYSTEM**” submitted by **Bishwaraj Dhar, Bijit Ranjan Das** and **Bwhwima Narzary** to the Department of Instrumentation Engineering of Central Institute of Technology, Kokrajhar has been examined and evaluated.

The project work has been prepared as per the regulations of Central Institute of Technology and qualifies to be accepted in partial fulfillment of the requirement for the degree of B. Tech.

**Project Coordinator**

**Board of Examiners**



## CERTIFICATE

This is to certify that the thesis entitled, "**A FUZZY BASED ROOM AUTOMATION SYSTEM**" submitted by **Bishwaraj Dhar, Bijit Ranjan Das** and **Bwhwima Narzary** in partial fulfillment of the requirements for the award of Bachelor of Technology in Instrumentation Engineering during session 2011-2015 at Central Institute of Technology, Kokrajhar is an authentic work carried out by him under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other university/institute for the award of any degree.



Mr. Jeet Dutta

Assistant Professor and HOD (i/c)

Department of Instrumentation Engineering  
Central Institute of Technology, Kokrajhar

Date:

Place:

## DECLARATION

We hereby declare that the project work entitled "A FUZZY BASED ROOM AUTOMATION SYSTEM" is an authenticated work carried out by us under the guidance of Mr. Jeet Dutta for the partial fulfillment of the award of the Bachelor of Technology degree in Instrumentation Engineering and this work has not been submitted for similar purpose anywhere else except to department of IE, Central Institute of Technology, Kokrajhar.

Date:

Place: Kokrajhar

*Bishwaraj Dhar*  
BISHWARAJ DHAR

University Roll:Gau-c-11/I-211

Registration No: 081769

*Bijit Ranjan Das*  
BIJIT RANJAN DAS

University Roll:Gau-c-11/127

Registration No: 015817

*Bahuwima Narzary*  
BHOWIMA NARZARY

University Roll:Gau-c-11/79

Registration No: 014645

## ABSTRACT

The present work concentrates on the optimize control of home appliances like fan, lamp, heater etc. The control is based on the room temperature and the number of persons present in a room. The room temperature is sensed by LM35 and the number of persons present is determined by a counter circuit, both interfaced with PIC microcontroller .The control logic for controlling the output based on the inputs is specified by fuzzy logic implemented in MATLAB environment. A provision for monitoring the sensor or counter inputs is done on a window created in Visual C# software. The logic is specified for each and every input being fed to the microcontroller and accordingly the user can control the output like speed of the fan, heater temperature etc. manually from another window for output control designed in visual C#.

## ACKNOWLEDGEMENT

First of all, we would like to express our deep sense of respect and gratitude towards our advisor and guide Sir Jeet Dutta, who has been the guiding force behind this work. We are greatly indebted to him for his constant encouragement, invaluable advice and for propelling us further in every aspect of our academic life. We consider it our good fortune to have an opportunity to work with such a wonderful person.

We also extend our thanks to Mr Dipankar Sutradhar, Mr Dipen Deka, Mr Ganesh Roy and all the faculty members and staff of the Department of Instrumentation Engineering, Central Institute of Technology, Kokrajhar who have encouraged us for completing the project work.

Date:

Place: Kokrajhar

*Bishwaraj Dhar*  
Bishwaraj Dhar

Roll No: Gau-C-11/L-211

Dept. of IE

CIT, Kokrajhar

*Bijit Ranjan Das*  
Bijit Ranjan Das

Roll No: Gau-C-11/127

Dept. of IE

CIT, Kokrajhar

*Bwhwima Narzary*  
Bwhwima Narzary

Roll No: Gau-C-11/79

Dept. of IE

CIT, Kokrajhar



## CONTENTS

TITLE	i
CERTIFICATE OF PROJECT SUBMISSION	ii
CERTIFICATE BY THE BOARD OF EXAMINERS	iii
CERTIFICATE BY PROJECT SUPERVISOR	iv
DECLARATION	v
ABSTRACT	vi
ACKNOWLEDGEMENT	vii
CONTENTS	viii
LIST OF FIGURES	x

### CHAPTER 1.

<b>INTRODUCTION</b> .....	<b>1</b>
1.1. INTRODUCTION ABOUT THE PROJECT .....	1
1.2. OBJECTIVE OF THE PROJECT .....	2
1.3. LITERARY REVIEW .....	2

### CHAPTER 2.

<b>BACKGROUND THEORY</b> .....	<b>3</b>
--------------------------------	----------

### CHAPTER 3

<b>EXPERIMENTAL STUDY</b> .....	<b>5</b>
3.1 HARDWARE DESIGN .....	5
3.1.1 BLOCK DIAGRAM OF THE PROJECT .....	5
3.1.2 CIRCUIT DIAGRAM OF THE PROJECT .....	6
3.1.3 CODE WRITTEN IN MIKROC FOR DAQ, CONTROL AND COUNTER .....	6
3.2 SOFTWARE DESIGN .....	8
3.2.1 DESIGN OF WINDOWS IN VISUAL C# .....	8
3.2.1.1 CODE FOR CREATING DAQ WINDOW .....	10
3.2.1.2 CODE FOR CREATING CONTROL WINDOW .....	10
3.2.2 DESIGN OF FUZZY LOGIC IN MATLAB .....	12

**CHAPTER 4**  
**RESULTS AND DISCUSSION.....16**

**CHAPTER 5.**  
**CONCLUSION AND FUTURE CONCERN.....18**  
    5.1 CONCLUSION.....18  
    5.2 FUTURE CONCERN.....18

**BIBLIOGRAPHY.....19**

**APPENDIX**

Data sheet of PIC18F458

## LIST OF FIGURES

	TITLE	PAGE NO
Figure 3.1	Block diagram of project .....	5
Figure 3.2	Circuit diagram of project.....	6
Figure 3.3	Operating window for monitoring process variable.....	9
Figure 3.4	Control window for controlling the speed of fan.....	9
Figure 3.5	Fuzzy logic editor panel in MATLAB .....	12
Figure 3.6	Selecting the temperature value as input.....	13
Figure 3.7	Selecting the counter values as input.....	13
Figure 3.8	output range for controlling the speed of fan.....	14
Figure 3.9	Rules window of fuzzy controller in MATLAB.....	14
Figure 4.1	Data acquired from the sensor LM35.....	16
Figure 4.2	Displaying the number of persons entered or exit.....	16
Figure 4.3	Window of output to be selected according to the input.....	17
Figure 4.4	controlling the speed of fan manually by using fuzzy logic.....	17

# CHAPTER 1

## INTRODUCTION

### 1.1 INTRODUCTION ABOUT THE PROJECT

As the world is moving towards rapid modernization in the twenty first century, the demand for electricity has become a never ending phenomena. Consequently the world is facing a severe crisis of electricity. In this context, we have planned to develop a system which can optimize the household as well as the commercial consumption of electricity. Sometimes it is seen that in living rooms as well as in offices and in commercial establishments, the wastage of energy is inevitable in conditions when all the electrical equipment's are running continuously and that too at the full speed in the absence of or presence of less numbers of persons in the room. We can optimize this usage of electricity to a great extent and help in saving power. This data acquisition and control system is based on fuzzy logic.

Fuzzy Logic Controllers are generally implemented on microprocessors and microcontrollers which offer a smooth design surface. Fuzzy logic concept comes from the establishments that the Boolean variable is not designed for the representation of the majority of the current phenomenon. While, the classic logic considers a statement as true or false, the fuzzy logic considers many cases between 0 and 1. The fuzzy logic has proved its superiority in complex systems control containing non-linearity and including human description or intuitive think.

In our project we have obtained a set of database for controlling the electrical appliances like fan based on input conditions like number of persons in the room and the room temperature. The database is prepared by implementing fuzzy logic in MATLAB software for a given set of inputs or membership functions and the relationship with the outputs. We have designed a system based on PIC microcontroller and Visual C# to interface the inputs obtained from the sensors with a serial PC, so that an operator can view the inputs in the PC screen. The inputs about the number of persons present in a room is acquired by using a LED-LDR based circuit and the temperature of the room is sensed by using a LM35 temperature sensor based circuit, both interface with PIC258. Further, a separate window is created by using Visual C#, which will enable the operator to control the appliances based on the database obtained from the fuzzy logic controller.



## 1.2 OBJECTIVE OF THE PROJECT

- 1) We developed a window by visual C# software to acquire the room temperature by interfacing LM35 with PIC Microcontroller.
- 2) Counter circuit is interfaced with a PIC Microcontroller and the number of persons entered and exit in a room is shown in the window.
- 3) We developed another window for controlling the speed of fan.
- 4) We developed a fuzzy logic by using MATLAB software. By using those logic we control the speed of fan.
- 5) The controlling section includes DAC through which it is easier to control the home appliances by a computer window.

## 1.3 LITERARY REVIEW

We have reviewed from many papers about the fuzzy logic based controller. Studying those paper we came through our designing of project. we have observed that some redundancies exist in the usage of motion, force and flow sensors and that could be controlled by reducing the number of sensor nodes to minimize the electrical load and cost [1].Based on a simple and fast learning algorithm that emulates the human learning process, the fuzzy controller self-adapts its control surface at a rate of 150 updates per second. The present implementation was based on the inexpensive and standard 8-bit microcontroller Intel 8031, which makes the self-learning fuzzy controller very cost effective [2]. With the help of fuzzy logic, manufacturers of home appliances are today embedding intelligence inside individual products. The application of fussy logic has also transformed industrial process control and enabled new product development strategies [2]. An implementation of this artificial intelligent based controller under the MATLAB/SIMULINK development environment is shown. It consists of functions that upgrade MATLAB/SIMULINK to a tool with hardware[3]

## CHAPTER 2

### BACKGROUND THEORY

Fuzzy logic control is a methodology bridging artificial intelligence and traditional control theory. This methodology is usually applied in the only cases when accuracy is not of high necessity or importance. On the other hand, as it is stated in (TI SPRA028, Jan.1993), "Fuzzy Logic can address complex control problems, such as robotic arm movement, chemical or manufacturing process control, antiskid braking systems or automobile transmission control with more precision and accuracy, in many cases, than traditional control techniques. Fuzzy Logic is a methodology for expressing operational laws of a system in linguistic terms instead of mathematical equations."

The Fuzzy Logic methodology (Yager & Zadeh, 1992; Klir & Yuan, 1996) comprises three phases:

1. The fuzzification is a transformation of analog (continuous) input variables to linguistic ones, e.g., transformation of temperature into the terms cool, warm, hot or transformation of speed into the terms negative big (NB), negative small (NS), zero (Z)", positive small (PS), positive big (PB). Such transformation is realized by introduction of so-called membership functions, which define both a range of value and a degree of membership. For linguistic variables it is important not only which membership function a variable belongs to, but also a relative degree (weight) to which it is a member. A variable can have a weighted membership in several membership functions at the same time.
2. The fuzzy inference maps input linguistic variables onto output linguistic variables on the base a system of fuzzy rules of the type "IF A THEN B" For instance: "IF the temperature is worm THEN the speed is Positive Small (PS)" or "IF the speed is Negative Big (NB) THEN force is ZERO". Since input linguistic variables are weighted, the output linguistic variables can be obtained-weighted as well. Traditional fuzzy logic approach comprises Mamdani- type and Sugeno-type inference methods. The Mamdani-type method is more intuitive and assumes the output variables as a fuzzy set. Fuzzy rules in it contain a fuzzy precondition part (after IF) and a fuzzy consequence part (after THEN).The Sugeno-type method expects the output variables to be singletons or dealing with consequents that are equations. So it is better suited for mathematical analysis, nonlinear system modeling and interpolation.



In the defuzzification phase, the weighted values of output linguistic variables obtained as a result of fuzzy inference have to be transformed to analogue (continuous) variables. This procedure is also based on membership functions. The maximum defuzzification method, wherein an output value is determined by the linguistic variable with the maximum weight. The centroid calculation defuzzification method, wherein an output value is determined by the weighted influence of all the active output membership functions.

As a rule, or at least in a great part of applications, a fuzzy controller is a transformer of input analog signals into an analog output signal. A linguistic variable is a subjective characteristic of an input analog variable, values of which are transformed on bases of given membership functions into a set of weighted values of corresponding linguistic variables. This procedure is called a fuzzification and it contains as its composite part the analog digital transformation.

A set of combinations of weighted linguistic variables corresponds to each value combination of input analog variables. On bases of a system of fuzzy inference rules it is possible to receive the set of weighted output linguistic variables. Using these variables and their membership functions, with help of one of well-known defuzzification methods it is possible to form values of the analog output variable. The defuzzification procedure also includes digital-analog transformation

## CHAPTER 3

### EXPERIMENTAL STUDY

#### 3.1 HARDWARE DESIGN

##### 3.1.1 BLOCK DIAGRAM OF THE PROJECT

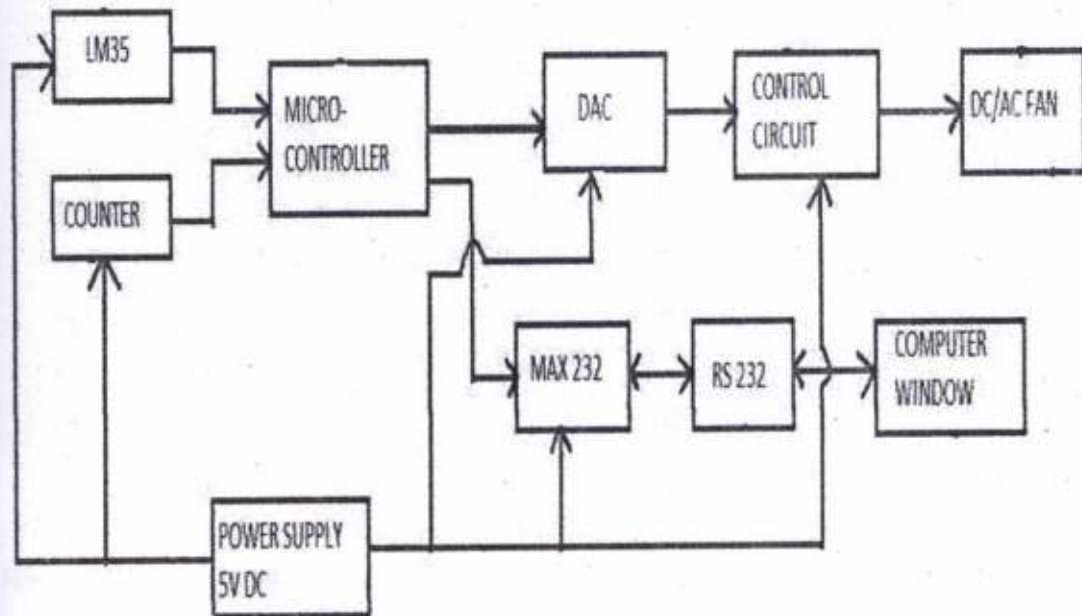


Figure 3.1: Block diagram of project

The above block diagram shows our total scheme of the project. Temperature sensor LM35 is interfaced with microcontroller to sense the room temperature and the counter circuit is also interfaced with it to detect the number of person entered in a room. The controlling part of our project comprises of DAC, control circuit and fuzzy logic. The temperature value and the number of person in a room will be monitored continuously on a computer window created by Visual C#. The fuzzy logic and Visual C# will be explained in the later sub headings. The values will be transferred serially through MAX232 and RS232. Through the logic implemented by fuzzy control we will then control the output manually by the computer window.



### 3.1.2 CIRCUIT DIAGRAM OF THE PROJECT

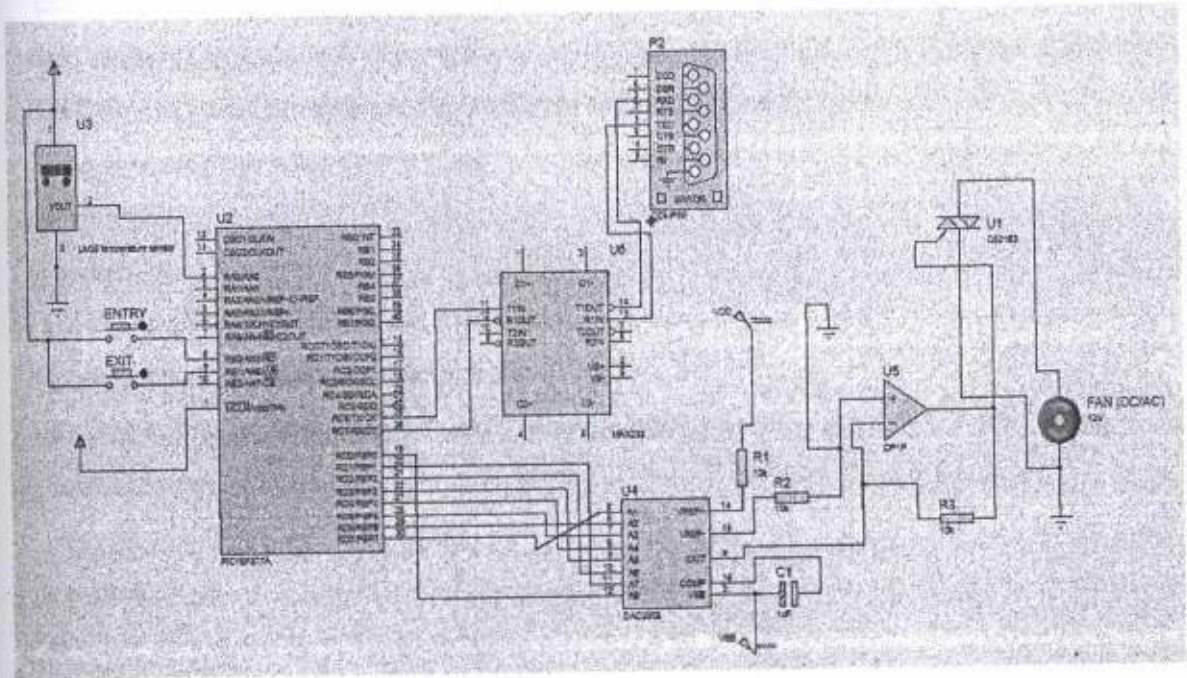


Figure 3.2: Circuit diagram of project

Here we used the sensor (LM35) which is connected to the analog pin of PIC16F877A. There is an inbuilt ADC in PIC which converts the analog input to the digital. The UART pin of PIC is connected to the MAX232 to send the data to the computer through RS232. As the voltage level of PIC is not compatible with RS232, MAX232 is used. The data of the sensor is sent serially to the computer. And the data can be seen in a window created by Visual C#. The counter circuit is fetched to port E of microcontroller which counts the number of persons entered in a room. The controlling is done by the windows which send the signal back to the microcontroller and fetched to the DAC 0808 for controlling purposes. The digital input from control window is then transformed into analog output which is fetched to the TRIAC to control the speed of the AC fan or heater. The fuzzy logic created in MATLAB is implemented to microcontroller through which we can control the output.

### 3.1.3 CODE WRITTEN IN MIKROC FOR DAQ, CONTROL AND COUNTER

```
void newline() {
    UART1_write(13);
    UART1_write(10);}

```

```

char txt [20];
float x,y;
char uart_rd;
void main() {
UART1_init(9600); //initializing the UART pin to baud rate 9600
UART1_write_text("temperature =");
newline();
newline();
while(1) {
x=adc_read(0); // reading the analog input
y=(x*500)/1024; // converting the LM35 value to degree celsius
floattostr(y,txt);
delay_ms(10000);
newline();
char uart_rd ; // for controlling the output
void main() {
TRISB=0;//selecting port b as output which will be fed to DAC
PORTB=0;
TRISE.F0=1; // counter input (entry)
TRISE.F1=1; // counter input (exit)
UART1_init(9600);
delay_ms(100);
while(1)
{
if (UART1_Data_Ready()) { // data recieved by window to control
uart_rd = UART1_READ();
if (uart_rd== 'a') { PORTB=0x00; ;}
if (uart_rd== 'b') {PORTB=0x01;}
if (uart_rd== 'c') {PORTB=0x02;}
if (uart_rd== 'd') {PORTB=0x14;}
if (uart_rd== 'e') {PORTB=0x26;}
if (uart_rd== 'f') {PORTB=0x48;}
if (uart_rd== 'g') {PORTB=0x5A;}

```

```

if (uart_rd== 'h') {PORTB=0x7C;}
if (uart_rd== 'i') {PORTB=0x7E;}
if (uart_rd== 'j') {PORTB=0xAF;}
if (uart_rd== 'k') {PORTB=0xFF;}
}}
if (PORTE.F0==1)
    {i=i++;
    delay_ms(100);
    UART1_Write((i/10)+0x30); // entry counter values in VC#
    UART1_Write((i%10)+0x30);
    newline();
    delay_ms(500); }
if (PORTE.F1==1)
    {i=i--;
    delay_ms(100);
    UART1_Write((i/10)+0x30); // exit counter values in VC#
    UART1_Write((i%10)+0x30);
    newline();
    delay_ms(100);
    }
}}

```

## 3.2 SOFTWARE DESIGN

### 3.2.1 DESIGN OF WINDOWS IN VISUAL C#

C# (pronounced "C sharp") is a programming language that is designed for building a variety of applications that run on the .NET Framework. C# is simple, powerful, type-safe, and object-oriented. The many innovations in C# enable rapid application development while retaining the expressiveness and elegance of C-style languages. Visual C# is an implementation of the C# language by Microsoft. Visual Studio supports Visual C# with a full-featured code editor, compiler, project templates, designers, code wizards, a powerful and easy-to-use debugger, and other tools. The .NET Framework class library provides access to many operating system services and other useful, well-designed classes that speed up the development cycle significantly.



We created a window using Visual C# where we can see the output of the sensors and various process parameters. Figure 3.3 shows the operating window for monitoring temperature of the process parameters

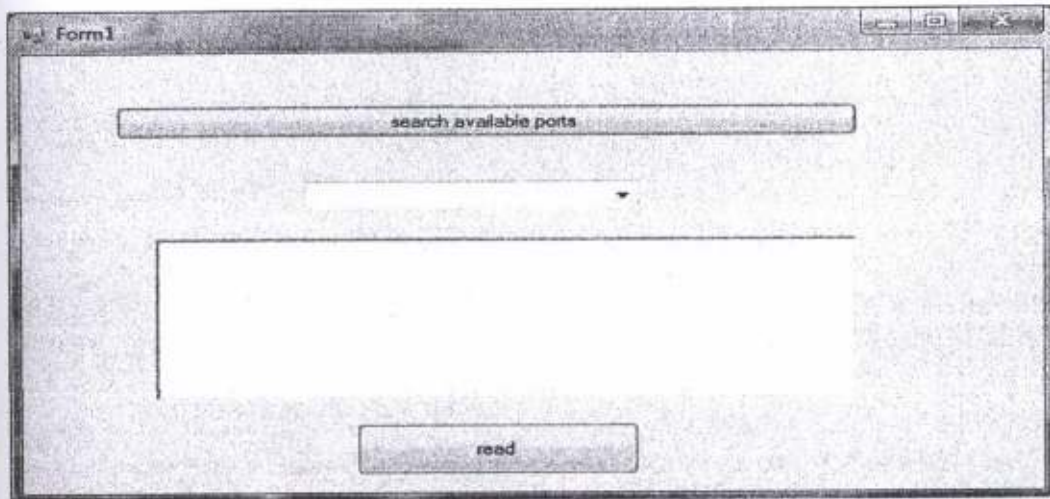


Figure 3.3: Operating window for monitoring process variable

We also created a control window by the software Visual C#. This window consists of several button to control the speed of ceiling fan. The control logic will be provided by fuzzy logic which will be described later.

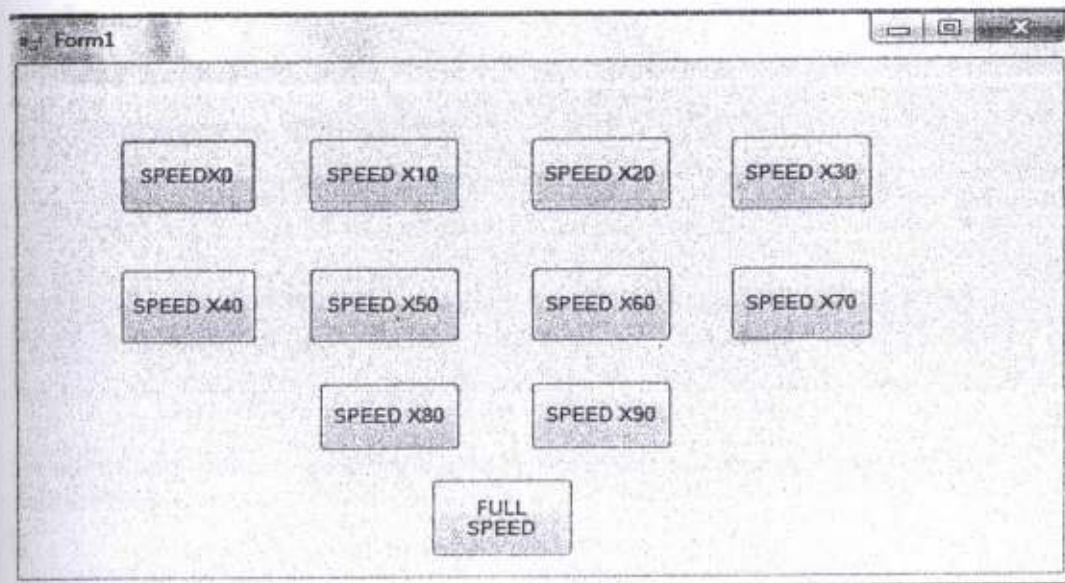


Figure 3.4: Control window for controlling the speed of fan



### 3.2.1.1 CODE FOR CREATING DAQ WINDOW

#### Code of search button:

```
string[] ports = SerialPort.GetPortNames();
    foreach (string port in ports)
    {comboBox1.Items.Add(ports);
    }
```

#### Code for read button:

```
t= comboBox1.Text.ToString();
sErial(t);
```

#### Code for receiving data:

```
SerialPort sp;
void sErial(string Port_name)
{sp = new SerialPort(Port_name, 9600 ,Parity . None, 8, StopBits.One);
    sp.DataReceived+=new
SerialDataReceivedEventHandler(DataReceivedHandler);
    sp.Open();
}
ReceivedHandler(object sender,SerialDataReceivedEventArgs e)
{
SerialPort sp = (SerialPort)sender;
string w = sp.ReadLine();
//string msg = sp. readExisting();
    if (w != String.Empty)
    {
        Invoke(new Action(() => richTextBox1.AppendText(w)));}
}
```

### 3.2.1.2 CODE FOR CREATING CONTROLLING WINDOW

```
SerialPort port = new SerialPort("COM1", 9600, Parity.None, 8, StopBits.One);
public Form1()
{
    InitializeComponent();
}
private void button1_Click(object sender, EventArgs e)
{
    port.Open();
}
```

```

port.Write("a");
port.Close();
}
private void button2_Click(object sender, EventArgs e)
{
port.Open();
port.Write("b");
port.Close();
}
private void button3_Click(object sender, EventArgs e)
{
port.Open();
port.Write("c");
port.Close();
}
private void button5_Click(object sender, EventArgs e)
{
port.Open();
port.Write("g");
port.Close();
}
private void button6_Click(object sender, EventArgs e)
{
port.Open();
port.Write("d");
port.Close();
}
private void button9_Click(object sender, EventArgs e)
{
port.Open();
port.Write("e");
port.Close();
}
private void button10_Click(object sender, EventArgs e)
{
port.Open();
port.Write("f");
port.Close();
}
private void button4_Click(object sender, EventArgs e)
{
port.Open();
port.Write("h");
port.Close();
}
private void button8_Click(object sender, EventArgs e)
{
port.Open();
port.Write("i");
port.Close();
}

```

```

}
private void button7_Click(object sender, EventArgs e)
{
    port.Open();
    port.Write("j");
    port.Close();
}
private void button11_Click(object sender, EventArgs e)
{
    port.Open();
    port.Write("k");
    port.Close();
}
private void Form1_Load(object sender, EventArgs e)
{
}
}
}

```

### 3.2.2 DESIGN OF FUZZY LOGIC IN MATLAB

A basic application might characterize various sub-ranges of a continuous variable. For instance, a temperature measurement for anti-lock brakes might have several separate membership functions defining particular temperature ranges needed to control the brakes properly. Each function maps the same temperature value to a truth value in the 0 to 1 range. These truth values can then be used to determine how the brakes should be controlled. The below figure shows the editor panel where you can select the input and output in MATLAB by just writing fuzzy in the front panel.

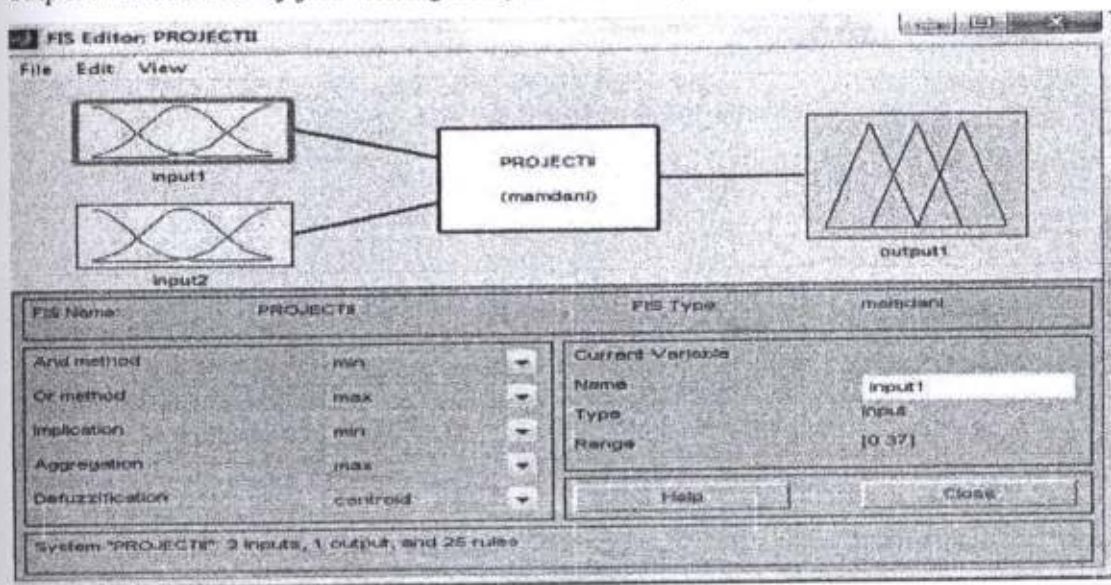


Figure 3.5: Fuzzy logic editor panel in MATLAB

By clicking the input we can select the range of the input in which way we will control the output. The below figure shows the input range chosen for our project. The input



chosen here is the room temperature and accordingly we will control the fan. The temperature range we chosen is up to 0-45

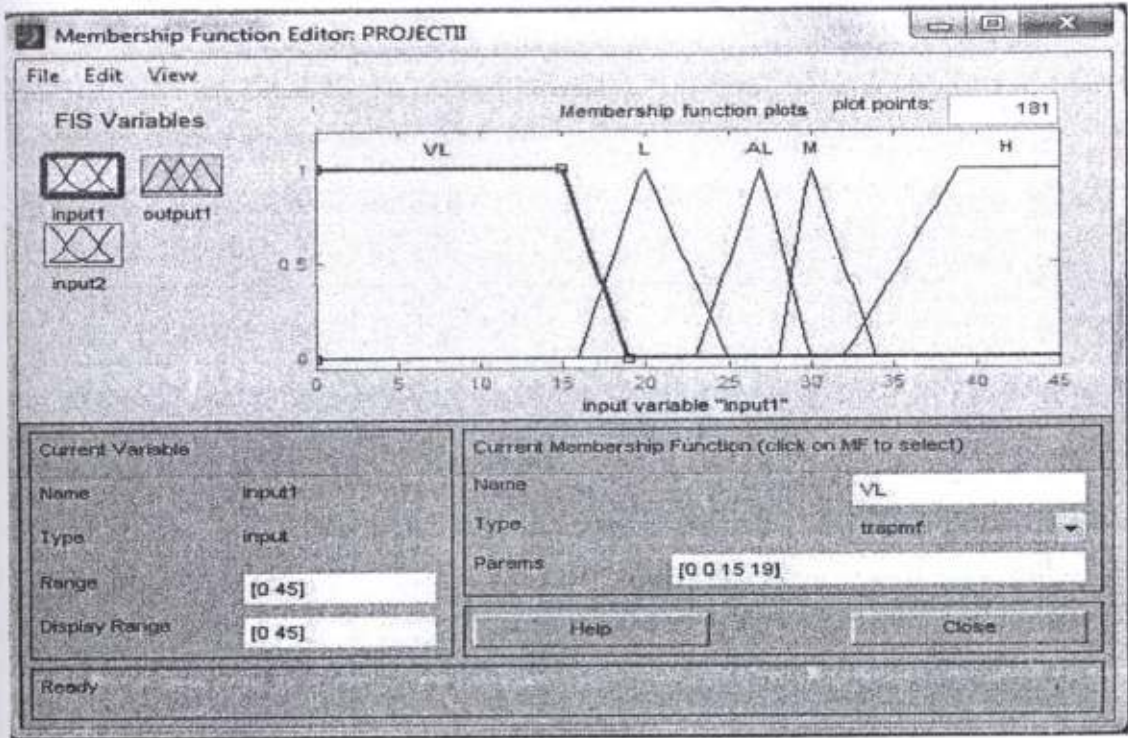


Figure 3.6: Selecting the temperature values as input

Fig 3.7 shown below is another input given to the microcontroller i.e. the number of persons in a room. By selecting the appropriate range we will next select the output range. The range of the number of person we chosen for our project is up to 0-30

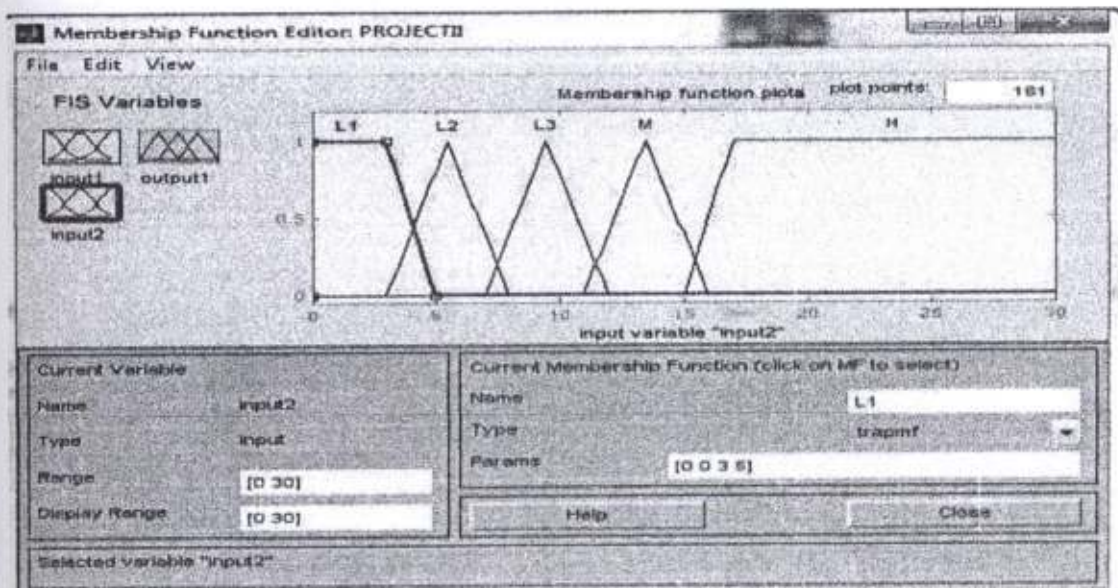


Figure 3.7: Selecting the counter values as input



The output range is then selected by clicking into the output. In our project we selected the output as a speed of fan according to the input received by a microcontroller. Fig 3.8 shows the selection of output and its range. The output range is up to 0-5 and it is the output voltage by which we can control the speed of fan according to the logic or rules created for each of the inputs provided

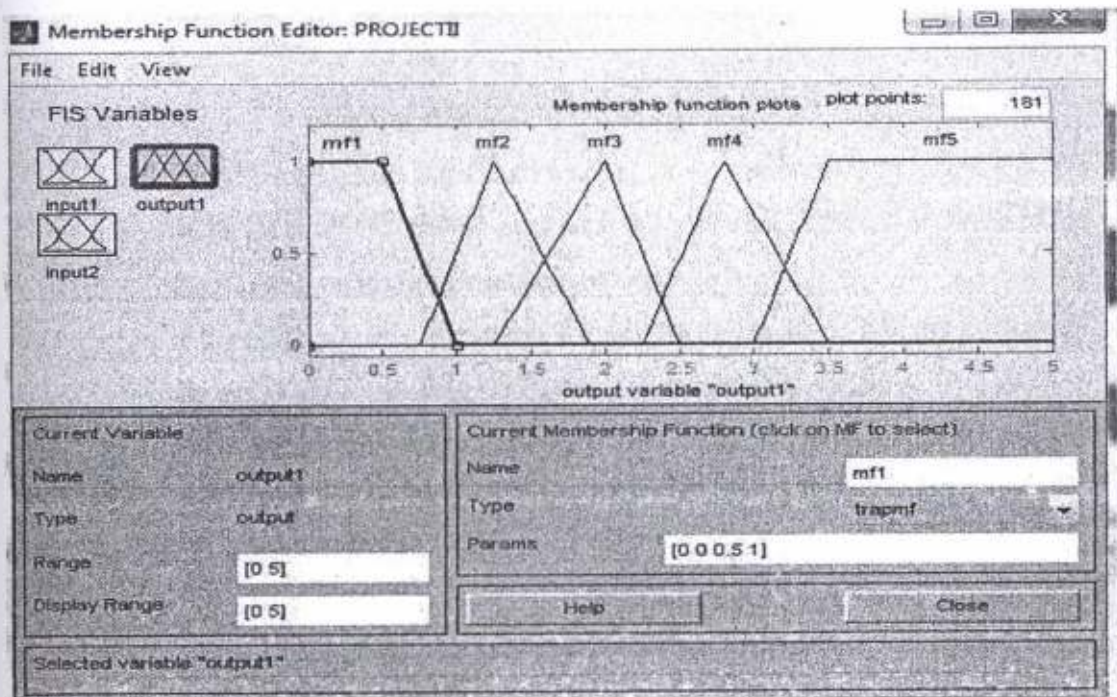


Figure 3.8: Output range for controlling the speed of fan

The rules is then selected which is shown in fig 3.9 below. Accordingly we can control the output using the logic we have made in the fuzzy control.

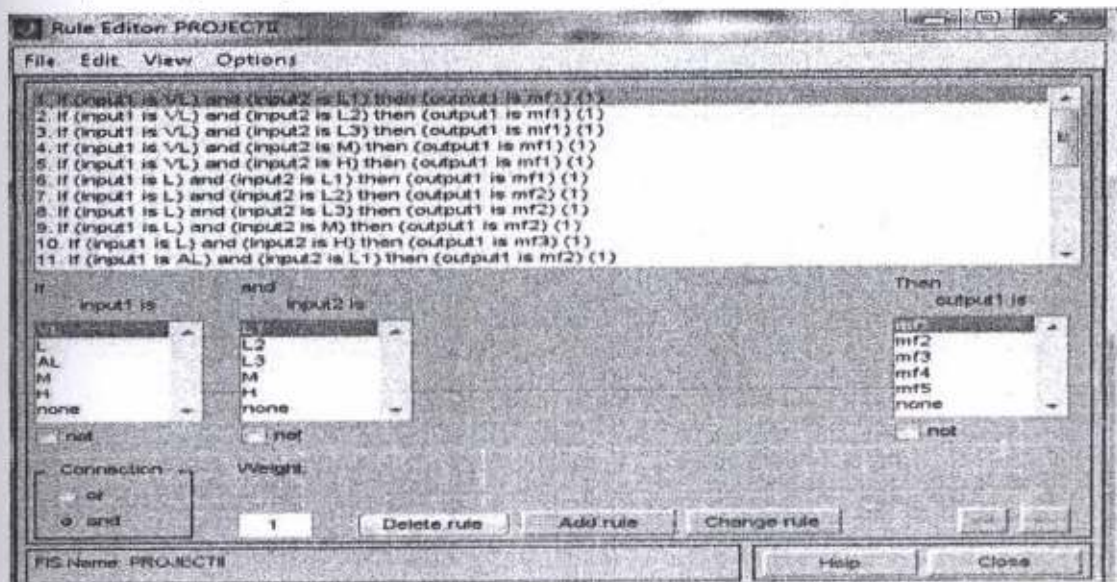


Figure 3.9: Rules window of fuzzy controller in MATLAB

The output which to be selected can be viewed from window after creating a rules in fuzzy logic. By creating those logic it is very simple to control the output for specific input. The rules and the logic to control are as follows:

- If the temperature is very low say between 0-20 and the number of persons in a room in less then the speed of fan is low
- If the temperature is low say above 20 -25 and the number of persons in a room is also moderate say up to 3-7 persons in a room the fan will be moving in an average speed
- If the temperature is quite above low say in our logic we created it in the range up to 25-30 and the persons in a room also higher then the speed of the fan will be higher than the previous rule
- If the temperature of the room is medium above 30-35 and the persons in a room is above 15 then the fan will move in a full speed.
- If the room temperature is higher above 35 degree centigrade and the number of person more or less the fan will move on full speed



## CHAPTER 4

### RESULTS AND DISCUSSIONS

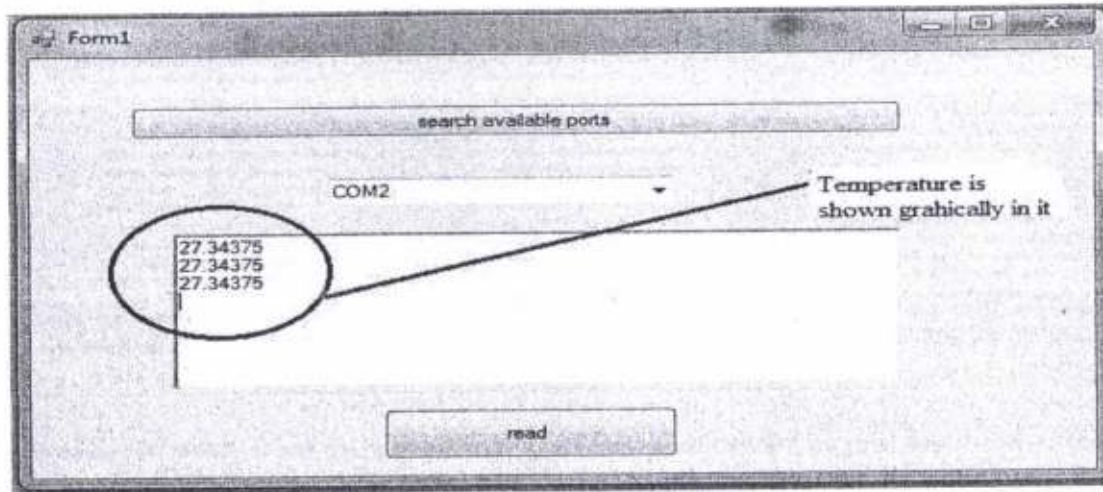


Figure 4.1: Data acquired from the sensor LM35

The fig shown above is the result of data acquired from the temperature sensor LM35 interfaced with PIC16F877A. The data shown will be then used to control the output manually. There is an option to select the ports by selecting the port connected to the microcontroller we can acquired the sensor value.

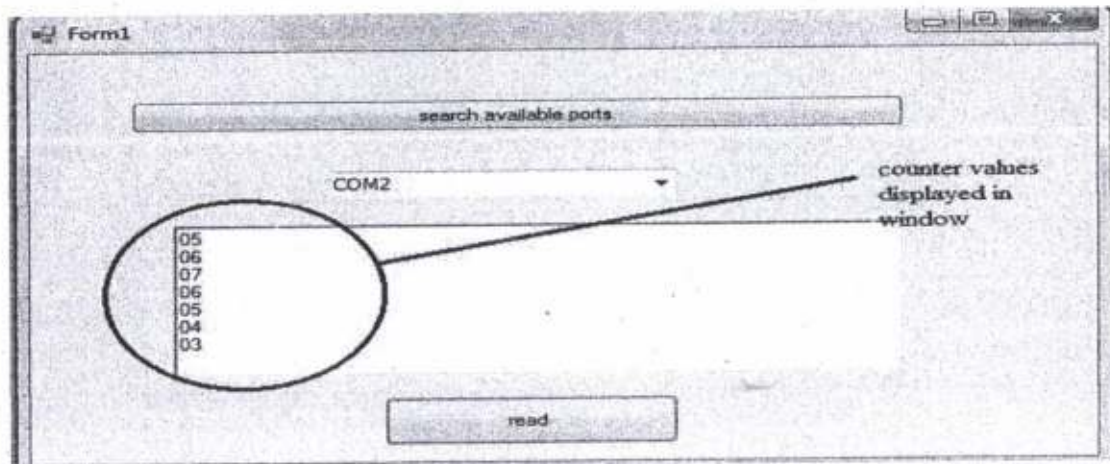


Figure 4.2: Displaying the number of persons entered or exit

The counter circuit placed in the door will give us the number of persons entered or exit, which will act as a second input for controlling the output. Above fig shows the window in which the number of persons entered or exit. Accordingly we can use both the above input according to fuzzy logic to control the speed of fan



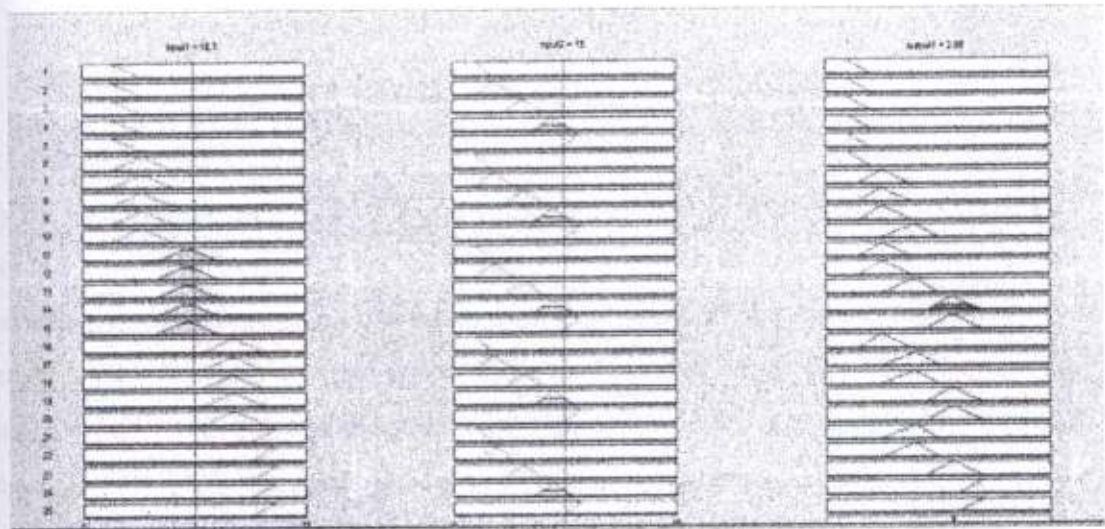


Figure 4.3: Window of output to be selected according to input

For each and every input there is output by seeing these we can control the speed of fan. Thus finally we control the output by controlling window by the logic provided by the fuzzy control designed in MATLAB. By using those two inputs interfaced with microcontroller we can manually control the fan. The above figure shows the logic and output for every input to be given to the microcontroller. This logic is created by giving rules for both the inputs with respect to its output generated

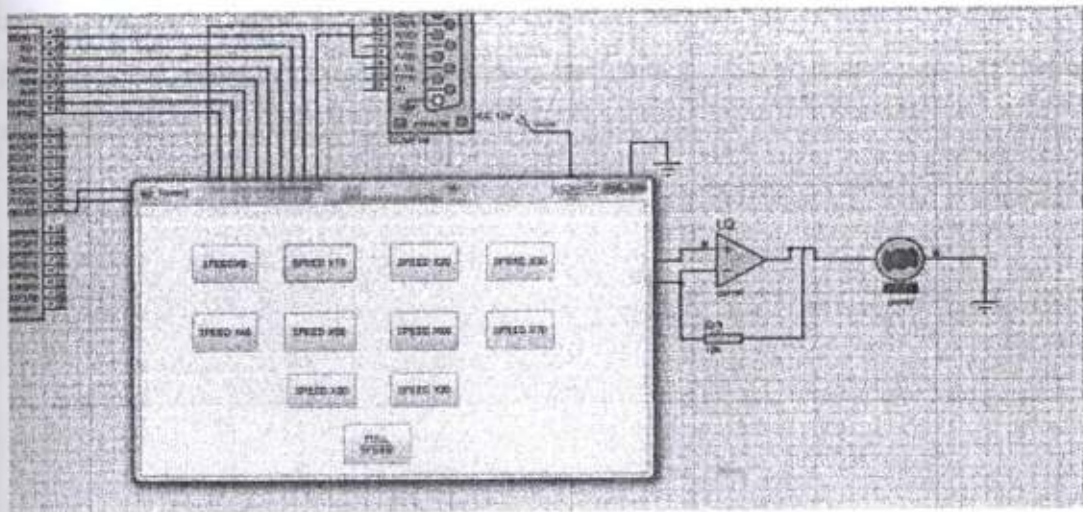


Figure 4.4: Controlling the speed of fan manually by using fuzzy logic

The controlling of motor is shown in the fig 4.3 by controlling window according to the logic provided by the fuzzy control. Each and every button of the controlling window is based on the logic created by fuzzy control. The logic shown in the window created by fuzzy control leads us to the controlling of fan in various speed and thus the power is optimized to a great extent.

## **CHAPTER 5**

### **CONCLUSION AND FUTURE CONCERN**

#### **5.1 CONCLUSION**

By the use of fuzzy logic control along with manual control options using a controlling window created in Visual C# we have been able to obtain the variable speed of the fan according to the logic. The conventional method required the human interruption to decide upon what should be the speed of the fan. In other words this situation analysis ability has been incorporated in a room which makes the appliances much more optimize and represents the decision taking power of the new arrangement. Though the analysis in this project has been very basic, but this clearly depicts the advantage of adding the fuzzy logic controller in controlling the speed of fan or other appliances.

#### **5.2 FUTURE CONCERN**

A more fully automatic control of speed of fan is straightforward to design using fuzzy logic technology. Moreover, the design process mimics human intuition, which adds to the ease of development, future maintenance and optimize the use of power. Although this particular example controls only the speed of fan, the design process can be extended without undue complications to other appliances such as AC, room heater etc. The formulation and implementation of membership functions and rules is similar to that shown for the controlling of fan speed.

## BIBLIOGRAPHY

1. Dutta, S. (2014). Fuzzy logic applications: Technological and strategic issues. *IEEE*, 10.
2. Samanta, N. D., & RoyChaudhuri, C. (2013). Multi sensor wireless system optimized for elderly health monitoring. *IEEE*, 7.
3. Zainzinger, H. (2014). An artificial intelligence based tool for home automation using MATLAB. *IEEE*, 13.
4. Dr.A.Murugan, D. (1999). *programming with Visual C#*. Andhra Pradesh: L K publisher.
5. Mazidi, M. A. (2003). *PIC microcontroller and embedded system using c program*. delhi: muhaamad ali publisher.
6. Smith. (n.d.). *Programming The Pic Microcontroller With Mbasic* . New Delhi: J R publisher.



MICRO

Device ID:

16C877A

16C877A

16C877A

16C877A

16C877A

16C877A

16C877A

16C877A

16C877A

16C877A

16C877A

16C877A

16C877A

16C877A

16C877A

16C877A

16C877A

16C877A

16C877A

16C877A

16C877A

16C877A

16C877A

16C877A

16C877A

16C877A

16C877A

16C877A

16C877A

16C877A

16C877A

16C877A

16C877A

16C877A

## APPENDIX

### DATASHEET OF PIC16F877A



# PIC16F87XA

## 28/40/44-Pin Enhanced Flash Microcontrollers

### Devices Included in this Data Sheet:

- PIC16F873A
- PIC16F874A
- PIC16F876A
- PIC16F877A

### High-Performance RISC CPU:

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

### Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler,  
can be incremented during Sleep via external  
crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period  
register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- Synchronous Serial Port (SSP) with SPI  
(Master mode) and I<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  $\overline{RD}$ ,  $\overline{WR}$  and  $\overline{CS}$  controls (40/44-pin only)
- Brown-out detection circuitry for  
Brown-out Reset (BOR)

### Analog Features:

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

### Special Microcontroller Features:

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

### CMOS Technology:

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

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

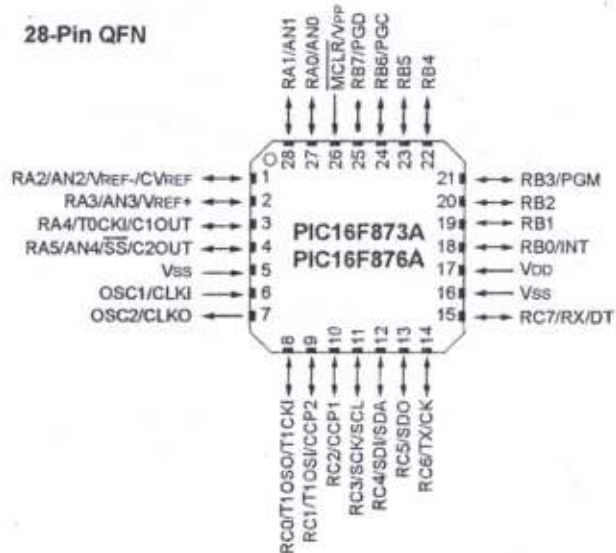
# PIC16F87XA

## Pin Diagrams

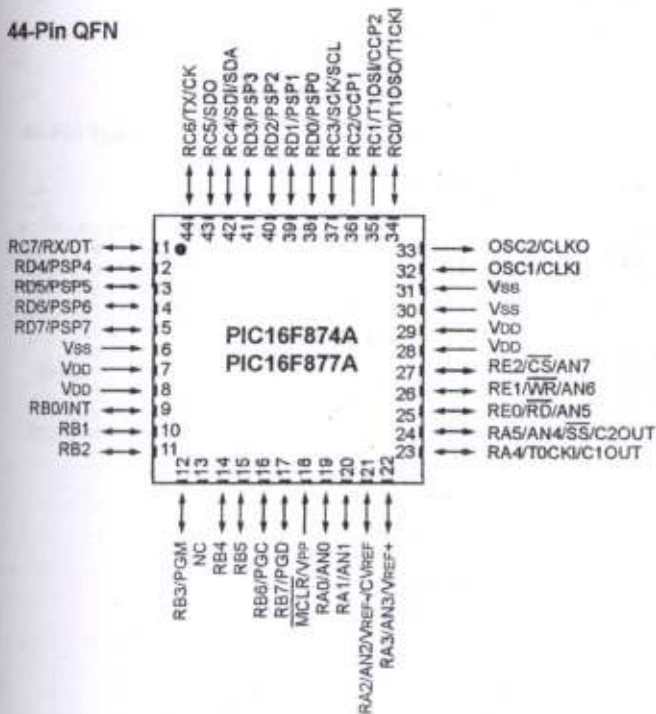
### 28-Pin PDIP, SOIC, SSOP



### 28-Pin QFN



### 44-Pin QFN



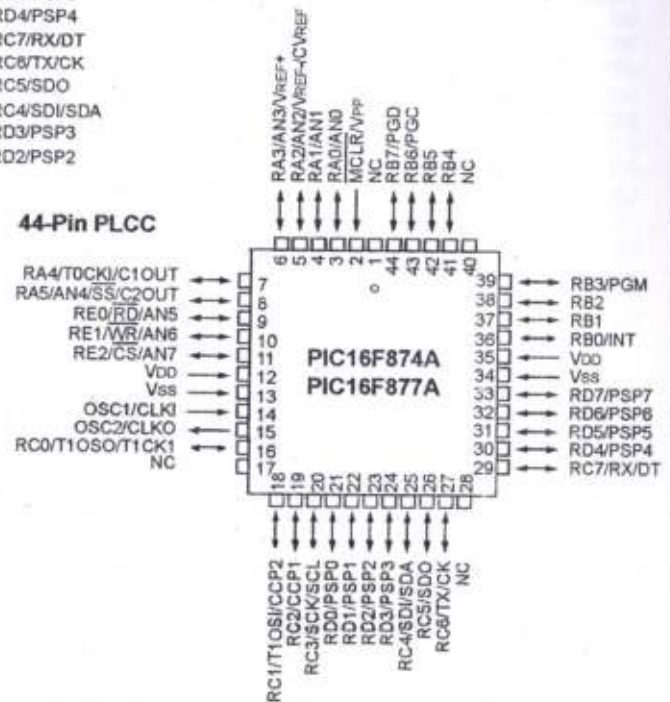


## Pin Diagrams (Continued)

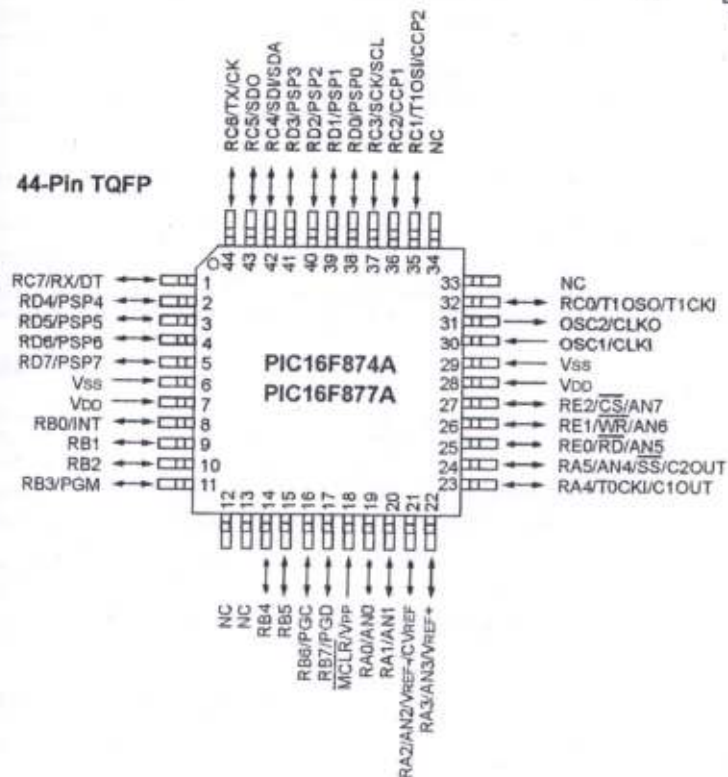
### 40-Pin PDIP



### 44-Pin PLCC



### 44-Pin TQFP



# PIC16F87XA

## Table of Contents

1.0	Device Overview	5
2.0	Memory Organization	15
3.0	Data EEPROM and Flash Program Memory	33
4.0	I/O Ports	41
5.0	Timer0 Module	53
6.0	Timer1 Module	57
7.0	Timer2 Module	61
8.0	Capture/Compare/PWM Modules	63
9.0	Master Synchronous Serial Port (MSSP) Module	71
10.0	Addressable Universal Synchronous Asynchronous Receiver Transmitter (USART)	111
11.0	Analog-to-Digital Converter (A/D) Module	127
12.0	Comparator Module	135
13.0	Comparator Voltage Reference Module	141
14.0	Special Features of the CPU	143
15.0	Instruction Set Summary	159
16.0	Development Support	167
17.0	Electrical Characteristics	173
18.0	DC and AC Characteristics Graphs and Tables	197
19.0	Packaging Information	209
	Appendix A: Revision History	219
	Appendix B: Device Differences	219
	Appendix C: Conversion Considerations	220
	Index	221
	On-Line Support	229
	Systems Information and Upgrade Hot Line	229
	Reader Response	230
	PIC16F87XA Product Identification System	231

## TO OUR VALUED CUSTOMERS

It is our intention to provide our valued customers with the best documentation possible to ensure successful use of your Microchip products. To this end, we will continue to improve our publications to better suit your needs. Our publications will be refined and enhanced as new volumes and updates are introduced.

If you have any questions or comments regarding this publication, please contact the Marketing Communications Department via E-mail at [docerrors@mail.microchip.com](mailto:docerrors@mail.microchip.com) or fax the Reader Response Form in the back of this data sheet to (480) 792-4150. We welcome your feedback.

### Most Current Data Sheet

To obtain the most up-to-date version of this data sheet, please register at our Worldwide Web site at:

<http://www.microchip.com>

You can determine the version of a data sheet by examining its literature number found on the bottom outside corner of any page. The last character of the literature number is the version number, (e.g., DS30000A is version A of document DS30000).

### Errata

An errata sheet, describing minor operational differences from the data sheet and recommended workarounds, may exist for current devices. As device/documentation issues become known to us, we will publish an errata sheet. The errata will specify the revision of silicon and revision of document to which it applies.

To determine if an errata sheet exists for a particular device, please check with one of the following:

- Microchip's Worldwide Web site; <http://www.microchip.com>
- Your local Microchip sales office (see last page)
- The Microchip Corporate Literature Center; U.S. FAX: (480) 792-7277

When contacting a sales office or the literature center, please specify which device, revision of silicon and data sheet (include literature number) you are using.

### Customer Notification System

Register on our Web site at [www.microchip.com/cn](http://www.microchip.com/cn) to receive the most current information on all of our products.



## 1.0 DEVICE OVERVIEW

This document contains device specific information about the following devices:

- PIC16F873A
- PIC16F874A
- PIC16F876A
- PIC16F877A

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

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

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

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

**TABLE 1-1: PIC16F87XA DEVICE FEATURES**

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



# PIC16F87XA

FIGURE 1-1: PIC16F873A/876A BLOCK DIAGRAM

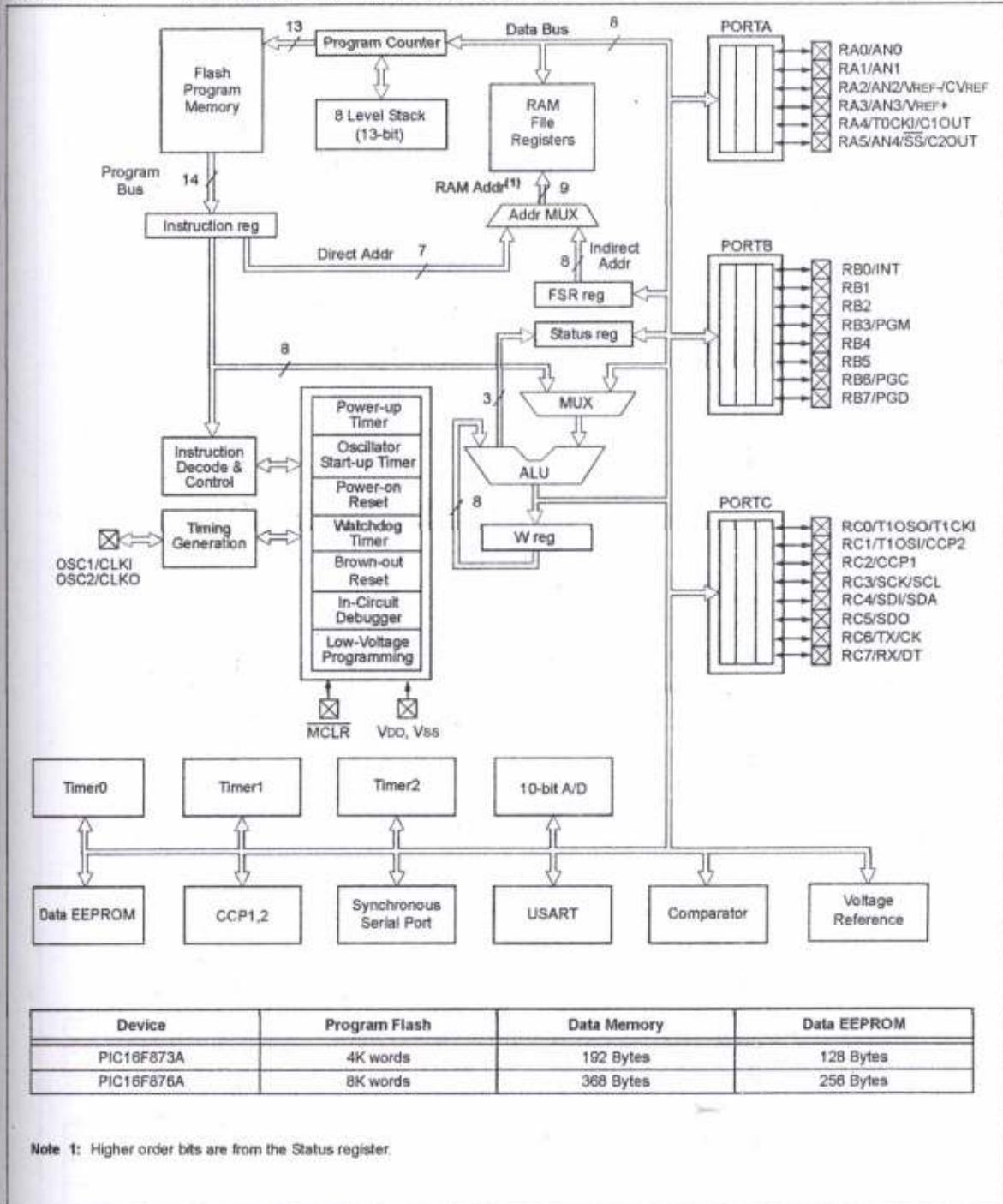
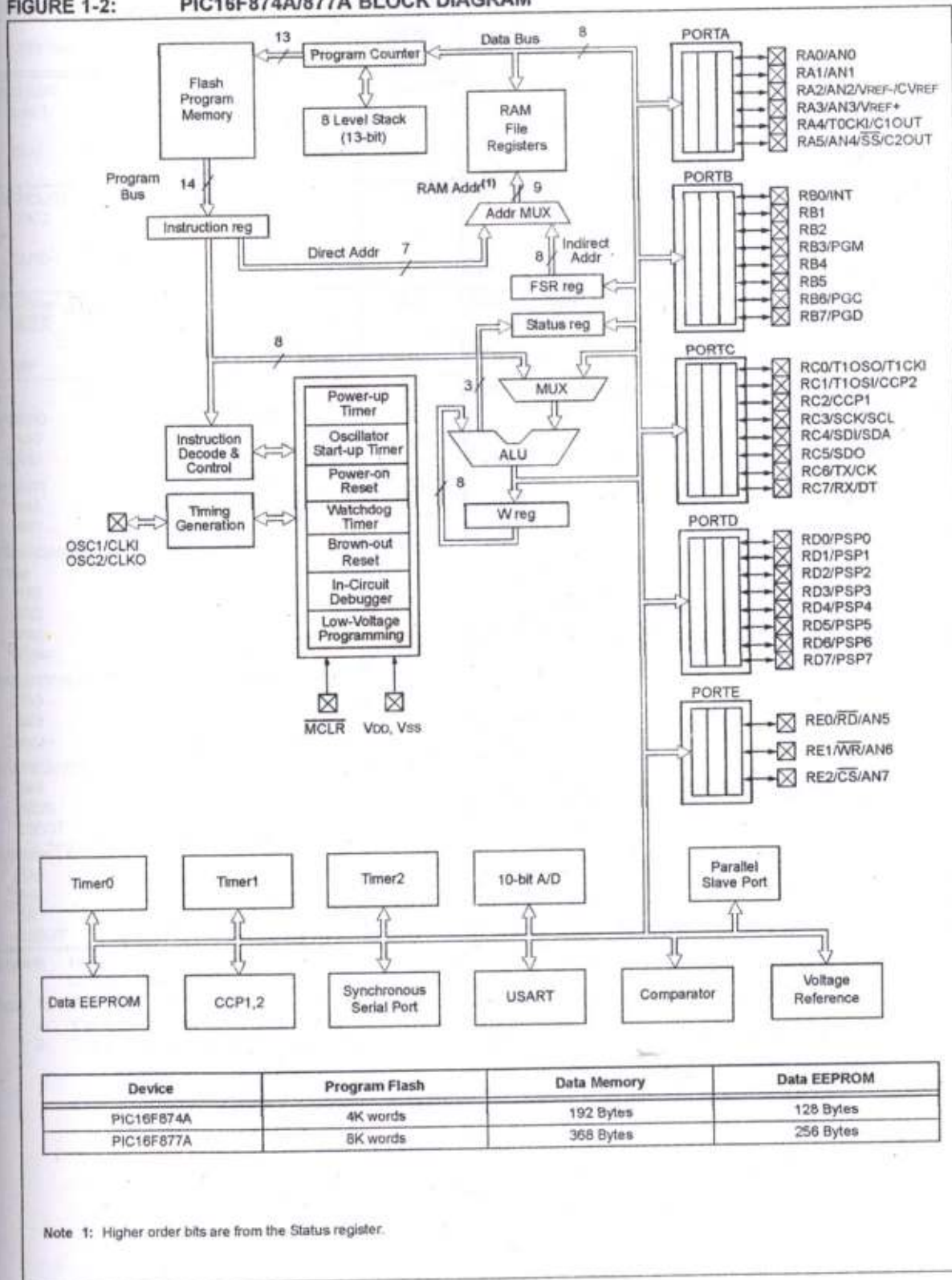


FIGURE 1-2: PIC16F874A/877A BLOCK DIAGRAM



# PIC16F87XA

TABLE 1-2: PIC16F873A/876A PINOUT DESCRIPTION

Pin Name	PDIP, SOIC, SSOP Pin#	QFN Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKI OSC1 CLKI	9	6	I I	ST/CMOS <sup>(3)</sup>	Oscillator crystal or external clock input. Oscillator crystal input or external clock source input. ST buffer when configured in RC mode; otherwise CMOS. External clock source input. Always associated with pin function OSC1 (see OSC1/CLKI, OSC2/CLKO pins).
OSC2/CLKO OSC2 CLKO	10	7	O O	—	Oscillator crystal or clock output. Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. In RC mode, OSC2 pin outputs CLKO, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
MCLR/VPP MCLR VPP	1	26	I P	ST	Master Clear (input) or programming voltage (output). Master Clear (Reset) input. This pin is an active low Reset to the device. Programming voltage input.
RA0/AN0 RA0 AN0	2	27	I/O I	TTL	PORTA is a bidirectional I/O port.  Digital I/O. Analog input 0.
RA1/AN1 RA1 AN1	3	28	I/O I	TTL	Digital I/O. Analog input 1.
RA2/AN2/VREF-/ CVREF RA2 AN2 VREF- CVREF	4	1	I/O I I O	TTL	Digital I/O. Analog input 2. A/D reference voltage (Low) input. Comparator VREF output.
RA3/AN3/VREF+ RA3 AN3 VREF+	5	2	I/O I I	TTL	Digital I/O. Analog input 3. A/D reference voltage (High) input.
RA4/T0CKI/C1OUT RA4 T0CKI C1OUT	6	3	I/O I O	ST	Digital I/O – Open-drain when configured as output. Timer0 external clock input. Comparator 1 output.
RA5/AN4/SS/C2OUT RA5 AN4 SS C2OUT	7	4	I/O I I O	TTL	Digital I/O. Analog input 4. SPI slave select input. Comparator 2 output.

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

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



# PIC16F87XA

**TABLE 1-2: PIC16F873A/876A PINOUT DESCRIPTION (CONTINUED)**

Pin Name	PDIP, SOIC, SSOP Pin#	QFN Pin#	I/O/P Type	Buffer Type	Description
RB0/INT RB0 INT	21	18	I/O I	TTL/ST <sup>(1)</sup>	PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs. Digital I/O. External interrupt.
RB1	22	19	I/O	TTL	Digital I/O.
RB2	23	20	I/O	TTL	Digital I/O.
RB3/PGM RB3 PGM	24	21	I/O I	TTL	Digital I/O. Low-voltage (single-supply) ICSP programming enable pin.
RB4	25	22	I/O	TTL	Digital I/O.
RB5	26	23	I/O	TTL	Digital I/O.
RB6/PGC RB6 PGC	27	24	I/O I	TTL/ST <sup>(2)</sup>	Digital I/O. In-circuit debugger and ICSP programming clock.
RB7/PGD RB7 PGD	28	25	I/O I/O	TTL/ST <sup>(2)</sup>	Digital I/O. In-circuit debugger and ICSP programming data.
RC0/T1OSO/T1CKI RC0 T1OSO T1CKI	11	8	I/O O I	ST	PORTC is a bidirectional I/O port. Digital I/O. Timer1 oscillator output. Timer1 external clock input.
RC1/T1OSI/CCP2 RC1 T1OSI CCP2	12	9	I/O I I/O	ST	Digital I/O. Timer1 oscillator input. Capture2 input, Compare2 output, PWM2 output.
RC2/CCP1 RC2 CCP1	13	10	I/O I/O	ST	Digital I/O. Capture1 input, Compare1 output, PWM1 output.
RC3/SCK/SCL RC3 SCK SCL	14	11	I/O I/O I/O	ST	Digital I/O. Synchronous serial clock input/output for SPI mode. Synchronous serial clock input/output for I <sup>2</sup> C mode.
RC4/SDI/SDA RC4 SDI SDA	15	12	I/O I I/O	ST	Digital I/O. SPI data in. I <sup>2</sup> C data I/O.
RC5/SDO RC5 SDO	16	13	I/O O	ST	Digital I/O. SPI data out.
RC6/TX/CK RC6 TX CK	17	14	I/O O I/O	ST	Digital I/O. USART asynchronous transmit. USART1 synchronous clock.
RC7/RX/DT RC7 RX DT	18	15	I/O I I/O	ST	Digital I/O. USART asynchronous receive. USART synchronous data.
Vss	8, 19	5, 6	P	—	Ground reference for logic and I/O pins.
Vcc	20	17	P	—	Positive supply for logic and I/O pins.

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

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



# PIC16F87XA

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

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

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

- Note 1:** This buffer is a Schmitt Trigger input when configured as the external interrupt.  
**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 in RC Oscillator mode and a CMOS input otherwise.



# PIC16F87XA

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

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

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

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

# PIC16F87XA

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

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

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

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

# PIC16F87XA

---

---

6F87XA

NOTES:



## 2.0 MEMORY ORGANIZATION

There are three memory blocks in each of the PIC16F87XA devices. 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 3.0 "Data EEPROM and Flash Program Memory".

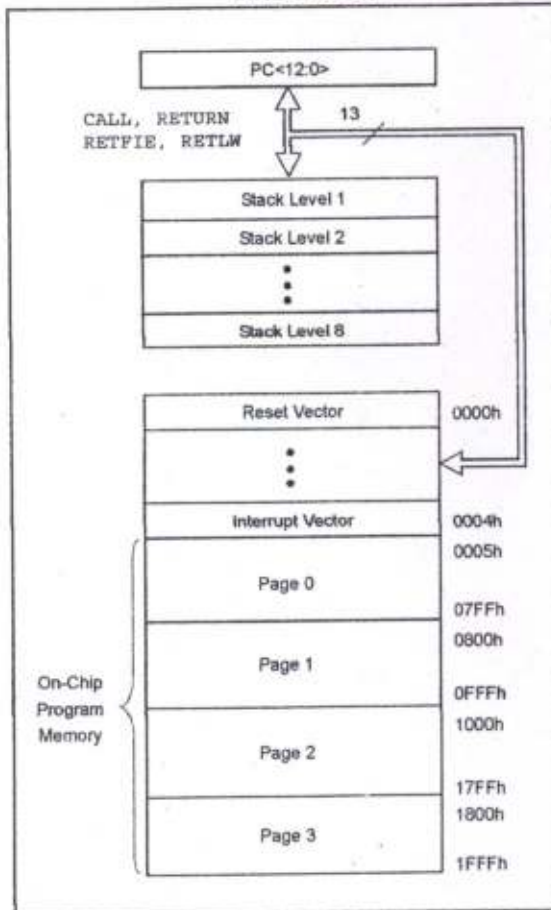
Additional information on device memory may be found in the PIC® Mid-Range MCU Family Reference Manual (DS33023).

## 2.1 Program Memory Organization

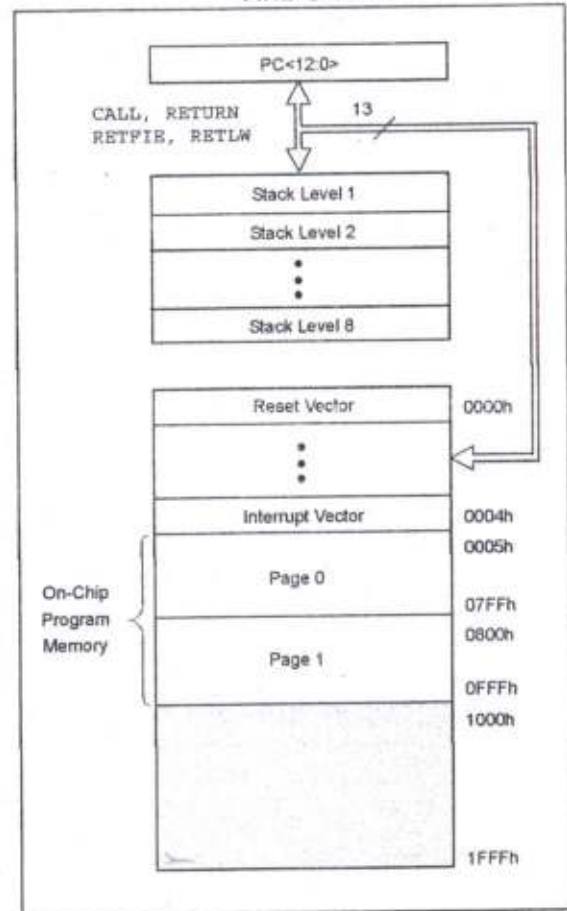
The PIC16F87XA devices have a 13-bit program counter capable of addressing an 8K word x 14 bit program memory space. The PIC16F876A/877A devices have 8K words x 14 bits of Flash program memory, while PIC16F873A/874A devices have 4K words x 14 bits. Accessing a location above the physically implemented address will cause a wraparound.

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

**FIGURE 2-1: PIC16F876A/877A PROGRAM MEMORY MAP AND STACK**



**FIGURE 2-2: PIC16F873A/874A PROGRAM MEMORY MAP AND STACK**



# PIC16F87XA

## 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.

**Note:** The EEPROM data memory description can be found in Section 3.0 "Data EEPROM and Flash Program Memory" 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).





# PIC16F87XA

FIGURE 2-4: PIC16F873A/874A REGISTER FILE MAP

File Address		File Address		File Address		File Address	
Indirect addr. <sup>(*)</sup>	00h	Indirect addr. <sup>(*)</sup>	80h	Indirect addr. <sup>(*)</sup>	100h	Indirect addr. <sup>(*)</sup>	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h		105h		185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h		107h		187h
PORTD <sup>(1)</sup>	08h	TRISD <sup>(1)</sup>	88h		108h		188h
PORTE <sup>(1)</sup>	09h	TRISE <sup>(1)</sup>	89h		109h		189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDATA	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh	Reserved <sup>(2)</sup>	18Eh
TMR1H	0Fh		8Fh	EEADRH	10Fh	Reserved <sup>(2)</sup>	18Fh
T1CON	10h		90h		110h		190h
TMR2	11h	SSPCON2	91h				
T2CON	12h	PR2	92h				
SSPBUF	13h	SSPADD	93h				
SSPCON	14h	SSPSTAT	94h				
CCPR1L	15h		95h				
CCPR1H	16h		96h				
CCP1CON	17h		97h				
RCSTA	18h	TXSTA	98h				
TXREG	19h	SPBRG	99h				
RCREG	1Ah		9Ah				
CCPR2L	1Bh		9Bh				
CCPR2H	1Ch	CMCON	9Ch				
CCP2CON	1Dh	CVRCON	9Dh				
ADRESH	1Eh	ADRESL	9Eh				
ADCON0	1Fh	ADCON1	9Fh				
	20h		A0h		120h		1A0h
General Purpose Register 96 Bytes		General Purpose Register 96 Bytes		accesses 20h-7Fh		accesses A0h - FFh	
	7Fh		FFh		16Fh 170h		1EFh 1F0h
Bank 0		Bank 1		Bank 2	17Fh	Bank 3	1FFh

Unimplemented data memory locations, read as '0'.  
 \* Not a physical register.

**Note 1:** These registers are not implemented on the PIC16F873A.  
**Note 2:** These registers are reserved; maintain these registers clear.



# PIC16F87XA

## 2.2.2 SPECIAL FUNCTION REGISTERS

The Special Function Registers are registers used by the CPU and peripheral modules for controlling the desired operation of the device. These registers are implemented as static RAM. A list of these registers is given in Table 2-1.

The Special Function Registers can be classified into two sets: core (CPU) and peripheral. Those registers associated with the core functions are described in detail in this section. Those related to the operation of the peripheral features are described in detail in the peripheral features section.

**TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY**

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:		
<b>Bank 0</b>													
00h <sup>(3)</sup>	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)									0000 0000	31, 150	
01h	TMR0	Timer0 Module Register											
02h <sup>(3)</sup>	PCL	Program Counter (PC) Least Significant Byte									xxxx xxxx	55, 150	
03h <sup>(3)</sup>	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	0000 0000	30, 150		
04h <sup>(3)</sup>	FSR	Indirect Data Memory Address Pointer											
05h	PORTA	—	—	PORTA Data Latch when written: PORTA pins when read								xxxx xxxx	31, 150
06h	PORTB	PORTB Data Latch when written: PORTB pins when read										--0x 0000	43, 150
07h	PORTC	PORTC Data Latch when written: PORTC pins when read										xxxx xxxx	45, 150
08h <sup>(4)</sup>	PORTD	PORTD Data Latch when written: PORTD pins when read										xxxx xxxx	47, 150
09h <sup>(4)</sup>	PORTE	—	—	—	—	—	RE2	RE1	RE0	xxxx xxxx	48, 150		
0Ah <sup>(1,3)</sup>	PCLATH	—	—	Write Buffer for the upper 5 bits of the Program Counter								---0 xxxx	49, 150
0Bh <sup>(3)</sup>	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	30, 150		
0Ch	PIR1	PSPIF <sup>(3)</sup>	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	24, 150		
0Dh	PIR2	—	CMIF	—	EEIF	BCLIF	—	—	CCP2IF	0-0 0--0	26, 150		
0Eh	TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register										xxxx xxxx	28, 150
0Fh	TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register										xxxx xxxx	60, 150
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	xxxx xxxx	60, 150		
11h	TMR2	Timer2 Module Register											
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	0000 0000	57, 150		
13h	SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register										0000 0000	62, 150
14h	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	xxxx xxxx	61, 150		
15h	CCPR1L	Capture/Compare/PWM Register 1 (LSB)										0000 0000	82, 82, 150
16h	CCPR1H	Capture/Compare/PWM Register 1 (MSB)										xxxx xxxx	63, 150
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	xxxx xxxx	63, 150		
18h	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	--00 0000	64, 150		
19h	TXREG	USART Transmit Data Register										0000 000x	112, 150
1Ah	RCREG	USART Receive Data Register										0000 0000	118, 150
1Bh	CCPR2L	Capture/Compare/PWM Register 2 (LSB)										0000 0000	118, 150
1Ch	CCPR2H	Capture/Compare/PWM Register 2 (MSB)										xxxx xxxx	63, 150
1Dh	CCP2CON	—	—	CCP2X	CCP2Y	CCP2M3	CCP2M2	CCP2M1	CCP2M0	xxxx xxxx	63, 150		
1Eh	ADRESH	A/D Result Register High Byte										--00 0000	64, 150
1Fh	ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	—	ADON	xxxx xxxx	133, 150		
										0000 00-0	127, 150		

**Legend:** x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved. Shaded locations are unimplemented, read as '0'.

- Note 1:** The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8>, whose contents are transferred to the upper byte of the program counter.
- 2:** Bits PSPIE and PSPIF are reserved on PIC16F873A/876A devices; always maintain these bits clear.
- 3:** These registers can be addressed from any bank.
- 4:** PORTD, PORTE, TRISD and TRISE are not implemented on PIC16F873A/876A devices, read as '0'.
- 5:** Bit 4 of EEADRH implemented only on the PIC16F876A/877A devices.



# PIC16F87XA

**TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)**

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:	
<b>Bank 1</b>												
80h <sup>(3)</sup>	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	31, 150	
81h	OPTION_REG	RBPV	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	23, 150	
82h <sup>(3)</sup>	PCL	Program Counter (PC) Least Significant Byte								0000 0000	30, 150	
83h <sup>(3)</sup>	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxxx	22, 150	
84h <sup>(3)</sup>	FSR	Indirect Data Memory Address Pointer								xxxxx xxxxx	31, 150	
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111	43, 150	
86h	TRISB	PORTB Data Direction Register								1111 1111	45, 150	
87h	TRISC	PORTC Data Direction Register								1111 1111	47, 150	
88h <sup>(4)</sup>	TRISD	PORTD Data Direction Register								1111 1111	48, 151	
89h <sup>(4)</sup>	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction bits				0000 -111	50, 151
8Ah <sup>(1,2)</sup>	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter						--0 0000	30, 150
8Bh <sup>(3)</sup>	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBF	0000 000x	24, 150	
8Ch	PIE1	PSPIE <sup>(2)</sup>	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	25, 151	
8Dh	PIE2	—	CMIE	—	EEIE	BCLIE	—	—	CCP2IE	-0-0 0--0	27, 151	
8Eh	PCON	—	—	—	—	—	—	POR	BOR	---- --qq	29, 151	
8Fh	—	Unimplemented								—	—	
90h	—	Unimplemented								—	—	
91h	SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000	83, 151	
92h	PR2	Timer2 Period Register								1111 1111	62, 151	
93h	SSPADD	Synchronous Serial Port (I <sup>2</sup> C mode) Address Register								0000 0000	79, 151	
94h	SSPSTAT	SMP	CKE	D/A	P	S	R/W	UA	BF	0000 0000	79, 151	
95h	—	Unimplemented								—	—	
96h	—	Unimplemented								—	—	
97h	—	Unimplemented								—	—	
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	111, 151	
99h	SPBRG	Baud Rate Generator Register								0000 0000	113, 151	
9Ah	—	Unimplemented								—	—	
9Bh	—	Unimplemented								—	—	
9Ch	CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	0000 0111	135, 151	
9Dh	CVRCON	CVREN	CVROE	CVRR	—	CVR3	CVR2	CVR1	CVR0	000- 0000	141, 151	
9Eh	ADRESL	A/D Result Register Low Byte								xxxxx xxxxx	133, 151	
9Fh	ADCON1	ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0	00-- 0000	128, 151	

**Legend:** x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved.  
Shaded locations are unimplemented, read as '0'.

- Note** 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8>, whose contents are transferred to the upper byte of the program counter.  
2: Bits PSPIE and PSPIF are reserved on PIC16F873A/876A devices; always maintain these bits clear.  
3: These registers can be addressed from any bank.  
4: PORTD, PORTE, TRISD and TRISE are not implemented on PIC16F873A/876A devices, read as '0'.  
5: Bit 4 of EEADRH implemented only on the PIC16F876A/877A devices.



# PIC16F87XA

**TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)**

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:
<b>Bank 2</b>											
100h <sup>(3)</sup>	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	31, 150
101h	TMR0	Timer0 Module Register								xxxx xxxx	55, 150
102h <sup>(3)</sup>	PCL	Program Counter's (PC) Least Significant Byte								0000 0000	30, 150
103h <sup>(3)</sup>	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxxx	22, 150
104h <sup>(3)</sup>	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	31, 150
105h	—	Unimplemented								—	—
106h	PORTB	PORTB Data Latch when written; PORTB pins when read								xxxx xxxx	45, 150
107h	—	Unimplemented								—	—
108h	—	Unimplemented								—	—
109h	—	Unimplemented								—	—
10Ah <sup>(1,3)</sup>	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	30, 150
10Bh <sup>(3)</sup>	INTCON	GIE	PEIE	TMROIE	INTE	RBIE	TMROIF	INTF	RBIF	0000 000x	24, 150
10Ch	EEDATA	EEPROM Data Register Low Byte								xxxx xxxx	39, 151
10Dh	EEADR	EEPROM Address Register Low Byte								xxxx xxxx	39, 151
10Eh	EEDATH	—	—	EEPROM Data Register High Byte					---x xxxx	39, 151	
10Fh	EEADRH	—	—	—	— <sup>(5)</sup>	EEPROM Address Register High Byte				---- xxxx	39, 151
<b>Bank 3</b>											
180h <sup>(3)</sup>	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	31, 150
181h	OPTION_REG	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	23, 150
182h <sup>(3)</sup>	PCL	Program Counter (PC) Least Significant Byte								0000 0000	30, 150
183h <sup>(3)</sup>	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxxx	22, 150
184h <sup>(3)</sup>	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	31, 150
185h	—	Unimplemented								—	—
186h	TRISB	PORTB Data Direction Register								1111 1111	45, 150
187h	—	Unimplemented								—	—
188h	—	Unimplemented								—	—
189h	—	Unimplemented								—	—
18Ah <sup>(1,3)</sup>	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	30, 150
18Bh <sup>(3)</sup>	INTCON	GIE	PEIE	TMROIE	INTE	RBIE	TMROIF	INTF	RBIF	0000 000x	24, 150
18Ch	EECON1	EEPGD	—	—	—	WRERR	WREN	WR	RD	x--- x000	34, 151
18Dh	EECON2	EEPROM Control Register 2 (not a physical register)								---- ----	39, 151
18Eh	—	Reserved; maintain clear								0000 0000	—
18Fh	—	Reserved; maintain clear								0000 0000	—

**Legend:** x = unknown, u = unchanged, c = value depends on condition, - = unimplemented, read as '0', r = reserved.  
Shaded locations are unimplemented, read as '0'.

- Note 1:** The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8>, whose contents are transferred to the upper byte of the program counter.
- 2:** Bits PSPIE and PSPIF are reserved on PIC16F873A/876A devices; always maintain these bits clear.
- 3:** These registers can be addressed from any bank.
- 4:** PORTD, PORTE, TRISD and TRISE are not implemented on PIC16F873A/876A devices, read as '0'.
- 5:** Bit 4 of EEADRH implemented only on the PIC16F876A/877A devices.

# PIC16F87XA

## 2.2.2.1 Status Register

The Status register contains the arithmetic status of the ALU, the Reset status and the bank select bits for data memory.

The Status register can be the destination for any instruction, as with any other register. If the Status register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. These bits are set or cleared according to the device logic. Furthermore, the  $\overline{TO}$  and  $\overline{PD}$  bits are not writable, therefore, the result of an instruction with the Status register as destination may be different than intended.

For example, `CLRF STATUS`, will clear the upper three bits and set the Z bit. This leaves the Status register as `000u u1uu` (where u = unchanged).

It is recommended, therefore, that only `BCF`, `BSF`, `SWAPF` and `MOVWF` instructions are used to alter the Status register because these instructions do not affect the Z, C or DC bits from the Status register. For other instructions not affecting any status bits, see Section 15.0 "Instruction Set Summary".

**Note:** The C and DC bits operate as a borrow and digit borrow bit, respectively, in subtraction. See the `SUBLW` and `SUBWF` instructions for examples.

### REGISTER 2-1: STATUS REGISTER (ADDRESS 03h, 83h, 103h, 183h)

	R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7								bit 0

- bit 7     **IRP:** Register Bank Select bit (used for indirect addressing)  
 1 = Bank 2, 3 (100h-1FFh)  
 0 = Bank 0, 1 (00h-FFh)
- bit 6-5   **RP1:RP0:** Register Bank Select bits (used for direct addressing)  
 11 = Bank 3 (180h-1FFh)  
 10 = Bank 2 (100h-17Fh)  
 01 = Bank 1 (80h-FFh)  
 00 = Bank 0 (00h-7Fh)  
 Each bank is 128 bytes.
- bit 4      **$\overline{TO}$ :** Time-out bit  
 1 = After power-up, `CLRWDT` instruction or `SLEEP` instruction  
 0 = A WDT time-out occurred
- bit 3      **$\overline{PD}$ :** Power-down bit  
 1 = After power-up or by the `CLRWDT` instruction  
 0 = By execution of the `SLEEP` instruction
- bit 2     **Z:** Zero bit  
 1 = The result of an arithmetic or logic operation is zero  
 0 = The result of an arithmetic or logic operation is not zero
- bit 1     **DC:** Digit carry/borrow bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions)  
 (for borrow, the polarity is reversed)  
 1 = A carry-out from the 4th low order bit of the result occurred  
 0 = No carry-out from the 4th low order bit of the result
- bit 0     **C:** Carry/borrow bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions)  
 1 = A carry-out from the Most Significant bit of the result occurred  
 0 = No carry-out from the Most Significant bit of the result occurred

**Note:** For borrow, the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (`RRF`, `RLF`) instructions, this bit is loaded with either the high, or low order bit of the source register.

#### Legend:

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



## 2.2.2.2 OPTION\_REG Register

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

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

### REGISTER 2-2: OPTION\_REG REGISTER (ADDRESS 81h, 181h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
<b>RBP</b>	<b>U</b>	<b>T</b>	<b>T</b>	<b>P</b>	<b>P</b>	<b>P</b>	<b>P</b>
<b>PU</b>	<b>EDG</b>	<b>CS</b>	<b>SE</b>	<b>SA</b>	<b>S2</b>	<b>S1</b>	<b>S0</b>
							bit 0

- bit 7 **RBP**: PORTB Pull-up Enable bit  
1 = PORTB pull-ups are disabled  
0 = PORTB pull-ups are enabled by individual port latch values
- bit 6 **U**: Interrupt Edge Select bit  
1 = Interrupt on rising edge of RB0/INT pin  
0 = Interrupt on falling edge of RB0/INT pin
- bit 5 **T**: TMR0 Clock Source Select bit  
1 = Transition on RA4/T0CKI pin  
0 = Internal instruction cycle clock (CLKO)
- bit 4 **T**: TMR0 Source Edge Select bit  
1 = Increment on high-to-low transition on RA4/T0CKI pin  
0 = Increment on low-to-high transition on RA4/T0CKI pin
- bit 3 **P**: Prescaler Assignment bit  
1 = Prescaler is assigned to the WDT  
0 = Prescaler is assigned to the Timer0 module
- bit 2-0 **P**: Prescaler Rate Select bits

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

#### Legend:

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

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



# PIC16F87XA

## 2.2.2.3 INTCON Register

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

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

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

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

- bit 7 **GIE:** Global Interrupt Enable bit  
1 = Enables all unmasked interrupts  
0 = Disables all interrupts
- bit 6 **PEIE:** Peripheral Interrupt Enable bit  
1 = Enables all unmasked peripheral interrupts  
0 = Disables all peripheral interrupts
- bit 5 **TMR0IE:** TMR0 Overflow Interrupt Enable bit  
1 = Enables the TMR0 interrupt  
0 = Disables the TMR0 interrupt
- bit 4 **INTE:** RB0/INT External Interrupt Enable bit  
1 = Enables the RB0/INT external interrupt  
0 = Disables the RB0/INT external interrupt
- bit 3 **RBIE:** RB Port Change Interrupt Enable bit  
1 = Enables the RB port change interrupt  
0 = Disables the RB port change interrupt
- bit 2 **TMR0IF:** TMR0 Overflow Interrupt Flag bit  
1 = TMR0 register has overflowed (must be cleared in software)  
0 = TMR0 register did not overflow
- bit 1 **INTF:** RB0/INT External Interrupt Flag bit  
1 = The RB0/INT external interrupt occurred (must be cleared in software)  
0 = The RB0/INT external interrupt did not occur
- bit 0 **RBIF:** RB Port Change Interrupt Flag bit  
1 = At least one of the RB7:RB4 pins changed state; a mismatch condition will continue to set the bit. Reading PORTB will end the mismatch condition and allow the bit to be cleared (must be cleared in software).  
0 = None of the RB7:RB4 pins have changed state

#### Legend:

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

## 2.2.2.4 PIE1 Register

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

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

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

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIE <sup>(1)</sup>	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
							bit 0
							bit 7

- bit 7 **PSPIE:** Parallel Slave Port Read/Write Interrupt Enable bit<sup>(1)</sup>  
 1 = Enables the PSP read/write interrupt  
 0 = Disables the PSP read/write interrupt  
**Note 1:** PSPIE is reserved on PIC16F873A/876A devices; always maintain this bit clear.
- bit 6 **ADIE:** A/D Converter Interrupt Enable bit  
 1 = Enables the A/D converter interrupt  
 0 = Disables the A/D converter interrupt
- bit 5 **RCIE:** USART Receive Interrupt Enable bit  
 1 = Enables the USART receive interrupt  
 0 = Disables the USART receive interrupt
- bit 4 **TXIE:** USART Transmit Interrupt Enable bit  
 1 = Enables the USART transmit interrupt  
 0 = Disables the USART transmit interrupt
- bit 3 **SSPIE:** Synchronous Serial Port Interrupt Enable bit  
 1 = Enables the SSP interrupt  
 0 = Disables the SSP interrupt
- bit 2 **CCP1IE:** CCP1 Interrupt Enable bit  
 1 = Enables the CCP1 interrupt  
 0 = Disables the CCP1 interrupt
- bit 1 **TMR2IE:** TMR2 to PR2 Match Interrupt Enable bit  
 1 = Enables the TMR2 to PR2 match interrupt  
 0 = Disables the TMR2 to PR2 match interrupt
- bit 0 **TMR1IE:** TMR1 Overflow Interrupt Enable bit  
 1 = Enables the TMR1 overflow interrupt  
 0 = Disables the TMR1 overflow interrupt

**Legend:**

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