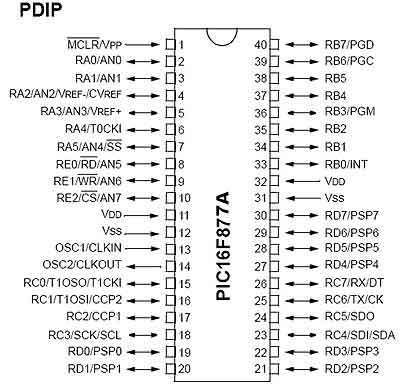
# PIC16F877 MICROCONTROLLER STUDY

https://www.alldatasheet.com/datasheet-pdf/pdf/75016/MICROCHIP/PIC16F877.html



**- SOFTWARE IDE Compiler Install**

[**MPLAB® XC Compilers | Microchip Technology**](https://www.microchip.com/en-us/tools-resources/develop/mplab-xc-compilers)

Three popular compiler which are used to program pic microcontrollers are MPLAB XC8,  Mikro C for pic, PIC CCS compiler and Hi-Tech compiler.

The official compiler is [MPLAB XC8 compiler](https://www.microchip.com/mplab/compilers) which is developed by manufactures of PIC16F877A.

We generally recommend Mikro C for pic compiler for beginners and MPLAB XC8 compiler for those who want to learn pic microcontrollers programming from register level bare metal concepts.

You can go through our [list pic microcontroller compilers](http://microcontrollerslab.com/pic-microcontroller-compiler/) article for further details.

**C Code Program and Proteus Simulation Topic**

**- GPIO**

PIN 1: MCLR: The first pin is the master clear pin of this IC. It resets the microcontroller and is active low, meaning that it should constantly be given a voltage of 5V and if 0 V are given then the controller is reset. Resetting the controller will bring it back to the first line of the program that has been burned into the IC.

A push button and a resistor is connected to the pin. The pin is already being supplied by constant 5V. When we want to reset the IC we just have to push the button which will bring the MCLR pin to 0 potential thereby resetting the controller.

PIN 2: RA0/AN0: PORTA consists of 6 pins, from pin 2 to pin 7, all of these are bidirectional input/output pins. Pin 2 is the first pin of this port. This pin can also be used as an analog pin AN0. It is built in [analog to digital converter](http://microcontrollerslab.com/analog-to-digital-adc-converter-working/).

PIN 3: RA1/AN1: This can be the analog input 1.

PIN 4: RA2/AN2/Vref- : It can also act as the analog input2. Or negative analog reference voltage can be given to it.

PIN 5: RA3/AN3/Vref+: It can act as the analog input 3. Or can act as the analog positive reference voltage.

PIN 6: RA0/T0CKI: To timer0 this pin can act as the clock input pin, the type of output is open drain.

PIN 7: RA5/SS/AN4: This can be the analog input 4. There is synchronous serial port in the controller also and this pin can be used as the slave select for that port.

PIN 8: RE0/RD/AN5:PORTE starts from pin 8 to pin 10 and this is also a bidirectional input output port. It can be the analog input 5 or for parallel slave port it can act as a ‘read control’ pin which will be active low.

PIN 9: RE1/WR/AN6:It can be the analog input 6. And for the parallel slave port it can act as the ‘write control’ which will be active low.

PIN 10: RE2/CS/A7: It can be the analog input 7, or for the parallel slave port it can act as the ‘control select’ which will also be active low just like read and write control pins.

PIN 11 and 32: VDD:These two pins are the positive supply for the input/output and logic pins. Both of them should be connected to 5V.

PIN 12 and 31: VSS:These pins are the ground reference for input/output and logic pins. They should be connected to 0 potential.

PIN 13: OSC1/CLKIN:This is the oscillator input or the external clock input pin.

PIN 14: OSC2/CLKOUT:This is the oscillator output pin. A crystal resonator is connected between pin 13 and 14 to provide external clock to the microcontroller. ¼ of the frequency of OSC1 is outputted by OSC2 in case of RC mode. This indicates the instruction cycle rate.

PIN 15: RC0/T1OCO/T1CKI: PORTC consists of 8 pins. It is also a bidirectional input output port. Of them, pin 15 is the first. It can be the clock input of timer 1 or the oscillator output of timer 2.

PIN 16: RC1/T1OSI/CCP2: It can be the oscillator input of timer 1 or the capture 2 input/compare 2 output/ PWM 2 output.

PIN 17: RC2/CCP1: It can be the capture 1 input/ compare 1 output/ PWM 1 output.

PIN 18: RC3/SCK/SCL: It can be the output for SPI or I2C modes and can be the input/output for synchronous serial clock.

PIN 23: RC4/SDI/SDA: It can be the SPI data in pin. Or in I2C mode it can be data input/output pin.

PIN 24: RC5/SDO: It can be the data out of SPI in the SPI mode.

PIN 25: RC6/TX/CK: It can be the synchronous clock or USART Asynchronous transmit pin.

PIN 26: RC7/RX/DT: It can be the synchronous data pin or the USART receive pin.

PIN 19,20,21,22,27,28,29,30: All of these pins belong to PORTD which is again a bidirectional input and output port. When the microprocessor bus is to be interfaced, it can act as the parallel slave port.

PIN 33-40: PORT B: All these pins belong to PORTB. Out of which RB0 can be used as the external interrupt pin and RB6 and RB7 can be used as in-circuit debugger pins.

Note: All pins have multiple functionalities. Like PORTC pins can be used as digital input pins, digital output pins, for UART communication, I2C communication. But, we can use one function of each port at a time or you have to program it smartly so that you can switch between different functionalities. Like once you need one pin to read analog signal, define it for analog purpose and then switch to other functions like digital output etc. For beginners, we suggest to use one pin for single functionality and you can use multi-function mode, once you become an expert in programming.

**- External Interrupt**

Interrupts have wonderful applications in embedded systems field. If you don’t know about interrupts, I suggest you to get complete understanding about them, you will not get command on embedded programming them. PIC16F877A microcontroller provides 8 types of interrupts namley; External interrupts, timer interrupts, PORT state change interrupts, UART  interrupt, I2C, PWM interrupts. you can read this guide on [pic microcontroller interrupts](http://microcontrollerslab.com/how-use-pic-microcontroller-timers-interrupt/) for additional information.

**- Timer**

It provides three timers timer0, timer1 and timer2.  All these timers can be used either in timer mode or in counter mode.  These timers are used to generate delays, pulse width modulation, counting external events and timer interrupts. TIMER0 is a 8 bit timer and it can operate with internal or external clock frequency. When we use Timer0 in timer mode, we usually operate it with internal frequency and in counter mode, we trigger it with external clock source.  Similarly, TIMER1 is a 16-bit timer and it can also operate in both modes. TIMER2 is also of 8-bit. It is used with PWM as a time base for CCP module.

**- Proteus**

A diagram of a circuit board

Description automatically generated

A computer screen shot of a circuit board

Description automatically generated

A computer screen shot of a circuit board

Description automatically generatedA computer circuit board with a screen

Description automatically generated with medium confidence

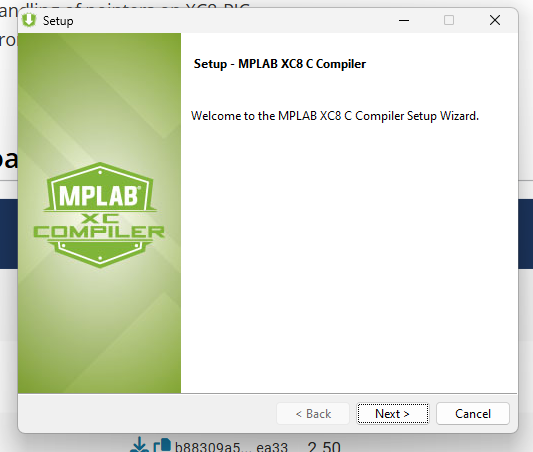
A computer circuit board with many wires

Description automatically generated with medium confidenceA computer screen shot of a circuit board

Description automatically generatedA computer screen shot of a circuit board

Description automatically generated

**- Compiler**



A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**- Exercise**

A computer screen shot of a circuit board

Description automatically generated

Code:

#include <xc.h> // Include the required header for your device

#define \_XTAL\_FREQ 4000000 // Define the clock frequency (adjust as needed)

// Segment display patterns for digits 0-9

unsigned char seg\_display[] = {0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0x7f, 0x6f};

void main() {

int i, tens, ones;

TRISB = 0x00; // Set PORTB as output (for tens digit)

TRISD = 0x00; // Set PORTD as output (for ones digit)

PORTB = 0x00; // Initialize PORTB to 0

PORTD = 0x00; // Initialize PORTD to 0

while (1) {

for (i = 0; i <= 99; i++) {

// Split the number into tens and ones places

tens = i / 10; // Tens digit

ones = i % 10; // Ones digit

// Display tens place on PORTB (first 7-segment display)

PORTB = seg\_display[tens]; // Display tens digit on PORTB

\_\_delay\_ms(100); // Short delay for tens digit update

// Display ones place on PORTD (second 7-segment display)

PORTD = seg\_display[ones]; // Display ones digit on PORTD

\_\_delay\_ms(100); // Short delay for ones digit update

}

}

}

A screenshot of a computer

Description automatically generated

BUILD SUCCESSFUL

A screenshot of a computer

Description automatically generated

CLICK 2nd BUTTON TO COMPILE HEX FILE