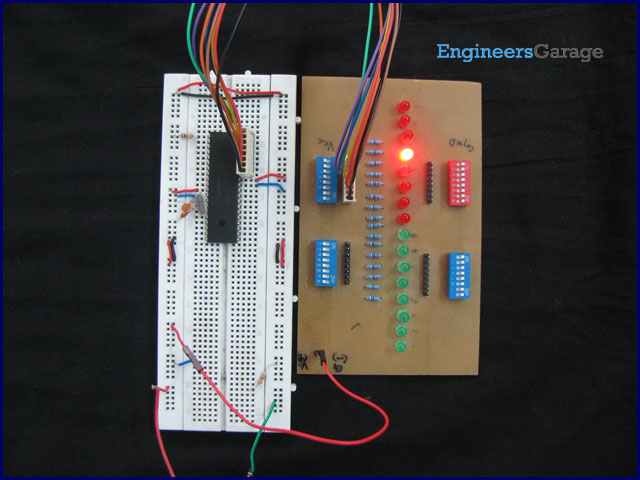
**[How to use Timers in PIC18F4550 Microcontroller](http://www.engineersgarage.com/embedded/pic-microcontroller-projects/timer-circuit" \o "How to use Timers in PIC18F4550 Microcontroller)**



[Timers](http://www.engineersgarage.com/tutorials/timers-8051-timer-programming-tutorial) as the name suggests pertain to time-related operations. They are mostly used for exact delay generation. Timers are also used in various other operations like PWM signal generation, auto-triggering of several other peripherals etc. The working and configuration of [PIC18F4550](http://www.engineersgarage.com/electronic-components/pic18f4550-microcontroller) Timers have been explained in this article.

Timers are the most essential peripheral for a [microcontroller](http://www.engineersgarage.com/microcontroller) and every controller provides a provision for using them. Beginners are advised to go through the [tutorial on Timers](http://www.engineersgarage.com/tutorials/timers-8051-timer-programming-tutorial) before going any further.

For basic Timer operations, refer the [Tutorial on Timers](http://www.engineersgarage.com/tutorials/timers-8051-timer-programming-tutorial). [PIC18F4550](http://www.engineersgarage.com/electronic-components/pic18f4550-microcontroller) is equipped with four Timers namely, Timer0, Timer1, Timer2 and Timer3. Before going for the details of Timer configurations, it is important to learn how time delay is calculated by the timer since exact delay generation is the most common application of Timers.

**Time Delay calculation with Timer:**

***Example:*** Given that a time delay of 1 sec is to be generated and a 12MHz crystal oscillator is connected with PIC. Please note that this example considers external clock source for the controller, however, PIC18F4550 has provision for both external as well as internal clock source.

Timer is related to the internal frequency which is always Fosc/4.

Clock source frequency (Crystal)

**Fosc = 12 MHz = 12000000 Hz**

Therefore, Timer frequency :

**FTimer = Fosc / 4 = 12000000 / 4 = 3000000 Hz = 3 MHz**

If Prescaler = 1:256, then

**FTimer = 3000000 / 256 = 11718.75 Hz**

So, the time interval :

**TTimer = 1 / FTimer = 1 / 11718.75 = 85 us (microseconds)**

This means that when Timer runs, it will take 85 ?secs to increment its value at every count.

To calculate the value to be filled in Timer rolling over register to generate 1 sec delay :

**No. of count for 1 sec Delay = 1sec / 85 us = 11718.75 = 2DC6H**

**So the value to be filled in Timer’s 16 bit register                  =          FFFF – 2DC6**

**=         D239**

These values are filled in the Timer register and it rolls over up to FFFF. The values are reloaded again to start timer for same delay. Before filling this value timer registers should be configured as we shall see.

Each of the four Timers of PIC18F4550 has certain special features some of which are explained below. The detailed list of these features can be obtained from [PIC18F4550](http://www.engineersgarage.com/electronic-components/pic18f4550-microcontroller) datasheet.

**Timer0:**

·         Timer0 can work as Timer/Counter in both 8-bit and 16-bit modes

·         Dedicated 8-bit, software programmable prescaler

·         Selectable clock source (internal or external)

·         Interrupt on overflow

**Timer1:**

·         Timer1 can work as 16-bit timer or counter

·         Readable and writable 8-bit registers (TMR1H and TMR1L)

·         Selectable clock source (internal or external)

·         Alternate clock source can be provided at Timer1 oscillator pins (T1OSO & T1OSI)

·         Interrupt on overflow

·         Timer1 can be multiplexed with other peripherals like ADC etc. and generates special event triggering for CCP (Capture, Compare and PWM) events.

**Timer2:**

·         8-bit Timer and Period registers (TMR2 and PR2, respectively)

·         Software programmable prescaler (1:1, 1:4 and 1:16)

·         Software programmable postscaler (1:1 - 1:16)

·         Interrupt on TMR2 to PR2 match

·         Optional use as the shift clock for the MSSP (Master Synchronous Serial Port) module

**Timer3:**

·         Timer3 can work as 16-bit timer or counter

·         Readable and writable 8-bit registers (TMR3H and TMR3L)

·         Selectable clock source (internal or external)

·         Alternate clock source can be provided at Timer1 oscillator pins (T1OSO & T1OSI)

·         Interrupt on overflow

·         Timer3 can be multiplexed with other peripherals like ADC etc. and generates special event triggering for CCP (Capture, Compare and PWM) events.

**PIC18F4550 Timer0 Registers:**

Every Timer has certain registers related to it which must be configured for desired operations. The registers of Timer0 have been explained below.

**1. T0CON (Timer0 Control Register)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit 7** | **Bit 6** | **Bit 5** | **Bit 4** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** |
| TMR0ON | T08BIT | T0CS | T0SE | PSA | T0PS2 | T0PS1 | T0PS0 |

**T0PS2: T0PS0:** These are prescaler selection bits. The bits setting is shown in the following table to choose desired prescaler.

|  |  |
| --- | --- |
| **T0PS2:T0PS0** | **Prescale Value** |
| 000 | 1:256 |
| 001 | 1:128 |
| 010 | 1:64 |
| 011 | 1:32 |
| 100 | 1:16 |
| 101 | 1:8 |
| 110 | 1:4 |
| 111 | 1:2 |

**PSA:** This bit is set to high if there is no need to assign a prescaler value.

        1 = Timer0 prescaler is not assigned. Timer0 clock input bypasses prescaler.

0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.

**T0SE:** This bit is used when external source is selected for the Timer. This bit is used to select external clock edge to increment the Timer.

1 = Increment on high-to-low transition on T0CKI pin (Pin6)

0 = Increment on low-to-high transition on T0CKI pin (Pin6)

**T0CS:** This bit is used to select the proper clock source for Timer0.

1 = Transition on T0CKI pin (Pin6)

0 = Internal instruction cycle clock (CLKO)

**T08BIT:** This bit selects the Timer mode.

1 = Timer0 is configured as an 8-bit timer/counter.

0 = Timer0 is configured as a 16-bit timer/counter.

**TMR0ON:** This bit is set to high (1) to enable the Timer0 and set to low (0) to stop it.

**2. INTCON (Interrupt Control Register)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit 7** | **Bit 6** | **Bit 5** | **Bit 4** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** |
| GIE/GIEH | PEIE/GIEL | **TMR0IE** | INT0IE | RBIE | **TMR0IF** | INT0IF | RBIF |

**TMR0IE:** This bit is used to enable/disable the Timer0 overflow interrupt.

1 = Enables the Timer0 overflow interrupt

0 = Disables the Timer0 overflow interrupt

**TMR0IF:** This is Timer0 overflow flag bit. This bit is set when TMR0 register overflows. This bit cleared by the software.

**3. TMR0 (Timer0 Register)**

This register is divided into registers TMR0H and TMR0L. Both registers are accessible separately thus Timer0 can work in both 8-bit and 16-bit modes. In these registers, pre-calculated value for delay is filled.

**Objective:**

**To configure the Timer0 and generate 1 second delay.**

**Programming Steps:**

1.         Select the Prescaler, Clock option, Mode of Timer0 with the T0CON register.

2.         Fill the higher byte of Timer value in TMR0H and then fill lower byte value in TMR0L register.

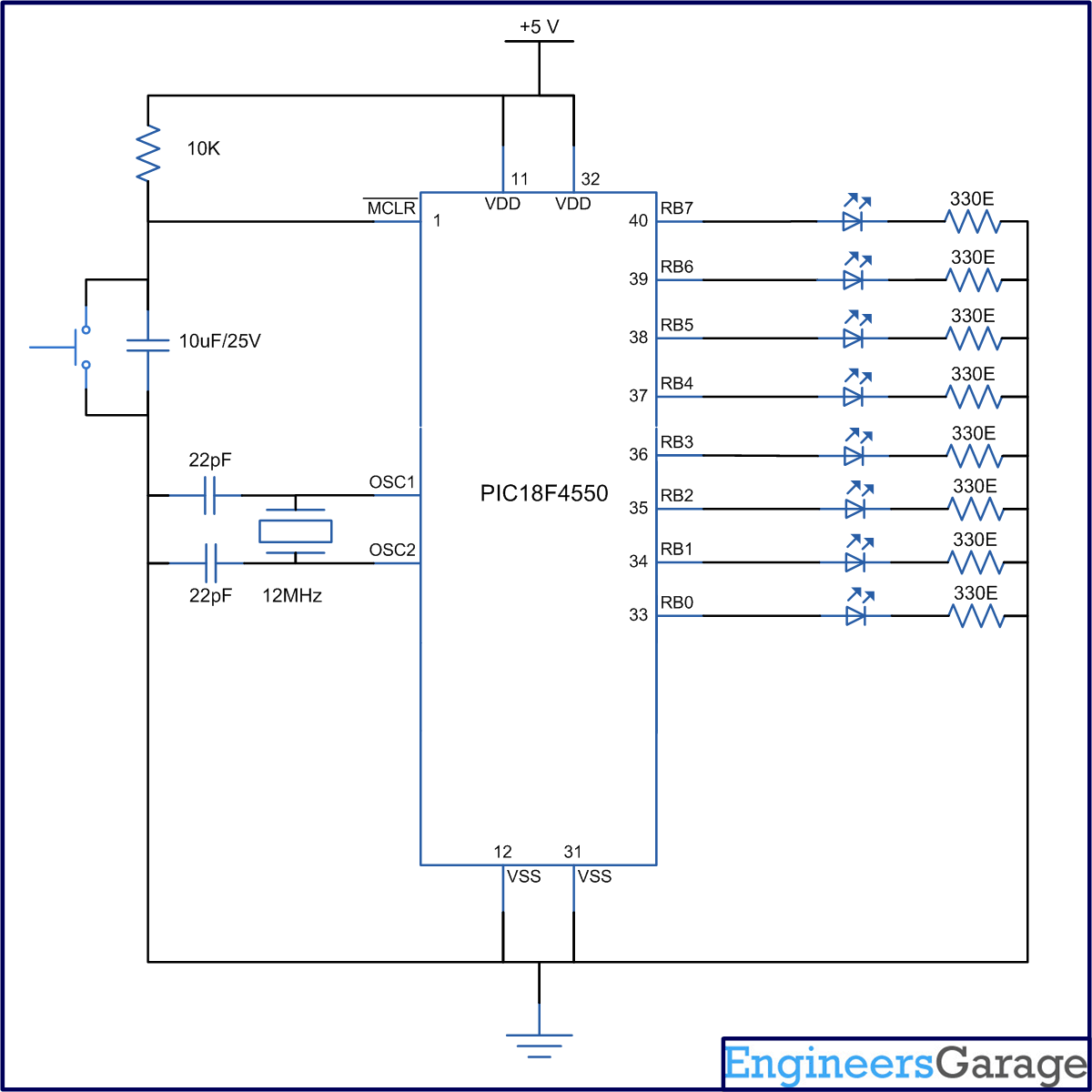
3.         Set the TMR0ON bit to start the timer.

4.         Wait until the TMR0IF flag gets high.

5.         As TMR0IF gets high, set it to zero and stop the timer by clearing the TMR0ON bit.

6.         To start the Timer0 again repeat the process from step2.

The time delay has been demonstrated by glowing a set of 8 LEDs one by one with a delay of 1 sec which can be seen the adjoining video. The circuit diagram and code for the same is also given.



// Program to use Timer0 of PIC18F4550 Microcontroller  
  
void T0\_init();  
void main()  
{  
 TRISB=0; // COnfigure PortB as output Port.  
 LATB=0x01;  
 T0CON=0x07; // Prescaler= 1:256, 16-bit mode, Internal Clock  
 while(1)  
 {  
 T0\_init(); // Initialize Timer0  
 LATB=(LATB<<1)|(LATB>>7); // Circular right shift at PortB  
 }  
}  
  
void T0\_init()  
{  
 TMR0H=0xD2; // Values calculated for 1 second delay with 12MHz crystal  
 TMR0L=0x39;  
 T0CON.TMR0ON=1; // Timer0 On  
 while(INTCON.TMR0IF==0); // Wait until TMR0IF gets flagged  
 T0CON.TMR0ON=0; // Timer0 Off  
 INTCON.TMR0IF=0; // Clear Timer0 interrupt flag  
}