Chapter 9. Timer Modules and Digital Clock Application

In 16F877, there are three timer modules: Timer0, Timer1, and Timer2 modules. The Timer0 module is a readable/writable 8-bit timer/counter consisting of one 8-bit register, TMR0. It triggers an interrupt when it overflows from FFh to 00h.

The Timer1 module is a readable/ writable 16-bit timer/counter consisting of two 8-bit registers (TMR1H and TMR1L). The TMR1 Register pair (TMR1H:TMR1L) increments from 0000h to FFFFh and rolls over to 0000h. The Timer1 Interrupt is generated on overflow.

The Timer2 is an 8-bit timer with a prescaler, a postscaler, and a period register. Using the prescaler and postscaler at their maximum settings, the overflow time is the same as a 16-bit timer. Timer2 is the PWM time-base when the CCP module(s) is used in the PWM mode. Detailed description and application of each timer, except Timer2 module, follow.

1. Timer 0

Timer0 module can work as a timer and a counter, however, in this section of Timer0, we use it as a timer only. In Timer1 module, we use it, instead, as a counter. So, for counter purpose, see the section for Timer1 module.

Timer mode is selected by clearing the T0CS bit (OPTION_REG<5>). In timer mode, the Timer0 module will increment every instruction cycle (without prescaler). Prescaler concept comes from the too-fast instruction cycle of the microcontroller. Think about the Timer0 register, TMR0. If the content is incremented by one every instruction (i.e., 0.2 μs with 20 MHz crystal oscillator), it takes, from 00h to FFh ,only 255x0.2μs=51μs. Then, how many overflow would we need, if we want to have an exact 1 second time delay? It would be over 19500 overflows. A mere 1ms delay would require about 20 overflows. Prescaler then is to give multiple instructions cycles for the increment of TMR0 register. Prescaler value of 1:4 would take 4 instruction cycles to increment TMR0 by 1. On the other hand, prescaler value of 1:256 requires 256 instruction cycles for the increment. With prescaler value of 1:256, one over flow would take 255x256x0.2μs=13056μs. Therefore, with 1:256, it would take only 76 overflows to have an exact 1 second timing. The prescaler is not readable or writable. Instead, The prescaler assignment is controlled in software by the PSA control bit (OPTION_REG<3>). Clearing the PSA bit will assign the prescaler to the Timer0 module.

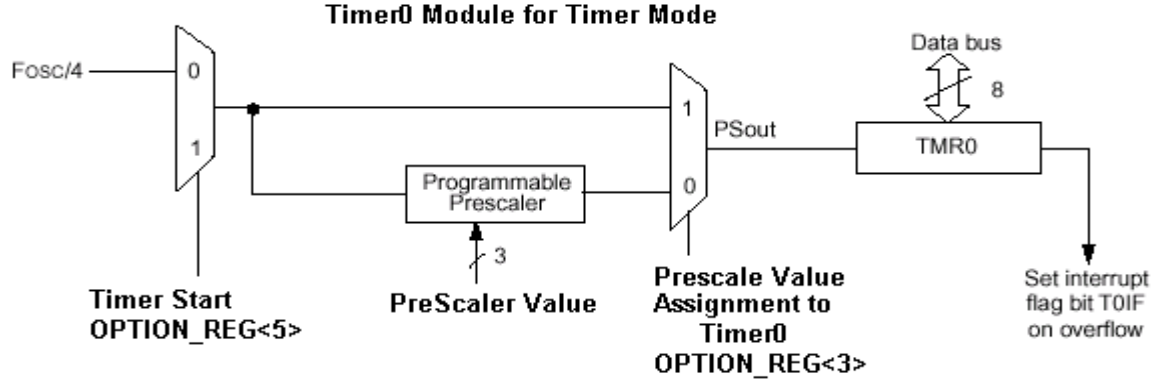

Fig. 70 Timer0 Module for Timer Mode

Timer0 starts or stops by the T0CS bit of OPTION_REG. Once it is started, the incremental signal comes to the TMR0 register based on the value selected for a prescaler. When TMR0 register is overflow, the T0IF flag is set to indicate the overflow. There are two ways to monitor the overflow event of TMR0: polling the T0IF flag and Triggering the Timer0 interrupt. In our example, we explore both the methods.

As you notice, we already talked about one register heavily, OPTION_REG register, while explaining the Timer0 module. The main control action of OPTION_REG register is to assign a prescaler value to Timer0 and start/stop the timer. Clearing T0CS bit starts the timer increment based on the prescaler value, assigned by clearing PSA bit and selected by the PS2:PS0 bits.

OPTION_REG (81h) For Timer Operation

$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0

T0CS: TMR0 Clock
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock

PSA: Prescaler Assignment
1 = Prescaler is to WDT
0 = Prescaler is to the Timer0

PS2:PS0: Prescaler Rate Select

TMR0 Rate			
1:2	0	0	0
1:4	0	0	1
1:8	0	1	0
1:16	0	1	1
1:32	1	0	0
1:64	1	0	1
1:128	1	1	0
1:256	1	1	1

The only other file register for the Timer0 module operation is INTCON register. INTCON register allows, in principle, interrupt for all interrupt enabled devices and modules. For the polling method, we may be able to enable the global interrupt by setting the GIE bit, but disable the T0IE bit of Timer0 module interrupt. Therefore, to use the interrupt method for Timer0 application, we have set both the bits: GIE and T0IE. If interrupt method is not used, just clearing GIE bit would do. In polling method, the pin T0IF bit must be monitored for the overflow of TMR0. In interrupt method, this is not necessary. However, for both the method, once a overflow event occurs, the T0IF must be cleared by software, i.e., in the code.