EC-310 Microprocessor and Microcontroller Based Design

Chapter - 9

PIC Timers

Nazar Abbas Saqib
nazar.abbas@ceme.nust.edu.pk

Objective

- List the Timers of PIC18 and their associated registers
- Describe the various modes of the PIC18 timers
- Program the PIC18 timers in Assembly to generate time delays
- Program the PIC18 timers in Assembly as event counters

Introduction

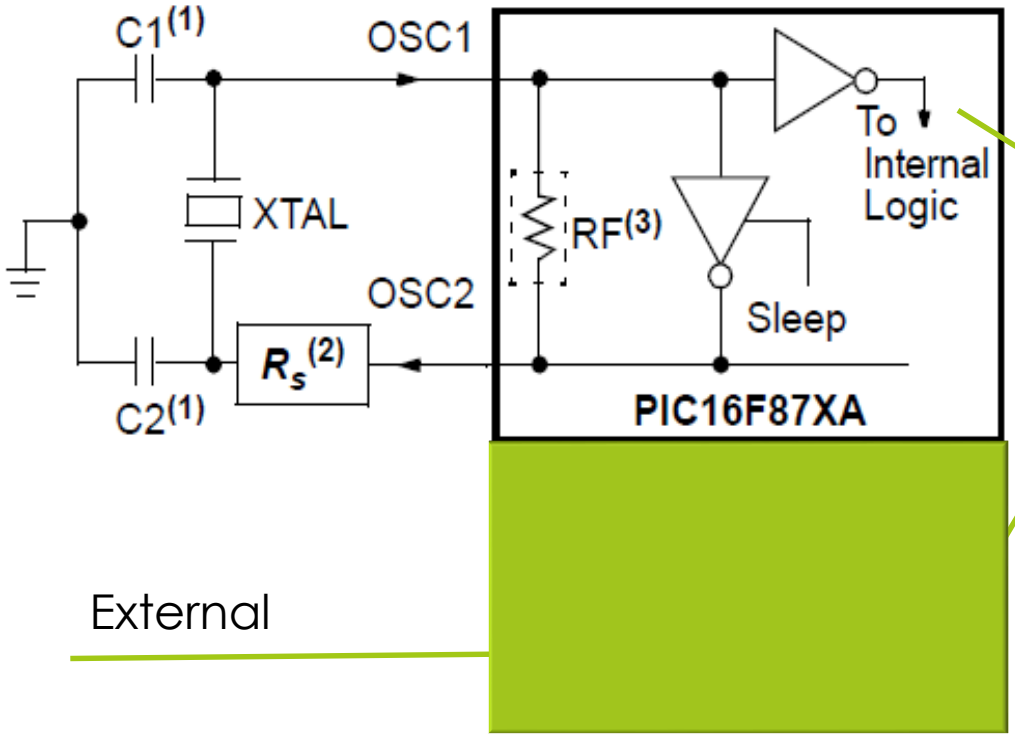
- PIC18 has two to five timers
  - Timers 0,1,2,3 and 4
  - Depending on the family number

- These timers can be used as
  - **Timers** to generate a time delay
  - **Counters** to count events happening outside the uC

Programming Timers 0 and 1

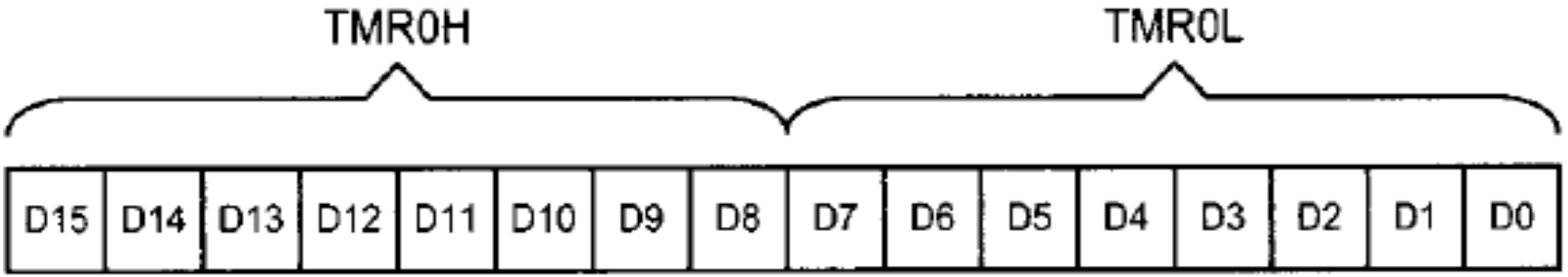
- Every timer needs a clock pulse to tick
- Clock source can be
  - **Internal** → 1/4th of the frequency of the crystal oscillator on OSC1 and OSC2 pins (Fosc/4) is fed into timer
  - **External:** pulses are fed through one of the PIC18's pins (RA4) → Counter

Programming Timers 0 and 1



Timer0 registers and programming

- Timers are 16-bit wide
  - Can be accessed as two separate reg. (TMRxL & TMRxH)
- TMR0L & TMR0H are 8-bit Reg.
  - MOVWF TMR0L
  - MOVFF TMR0L, PORTB



T0CON Reg (Timer0 Control)

Each timer has TCON (timer Control) reg.

8-bit register

TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0

TMR0ON D7 Timer0 ON and OFF control bit
1 = Enable (start) Timer0
0 = Stop Timer0

T08BIT D6 Timer0 8-bit/16-bit selector bit
1 = Timer0 is configured as an 8-bit timer/counter.
0 = Timer0 is configured as a 16-bit timer/counter.

T0CS D5 Timer0 clock source select bit
1 = External clock from RA4/T0CKI pin
0 = Internal clock (Fosc/4 from XTAL oscillator)

T0CON Reg (Timer0 Control)

8-bit register

TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0

T0SE D4 Timer0 source edge select bit
1 = Increment on H-to-L transition on T0CKI pin
0 = Increment on L-to-H transition on T0CKI pin

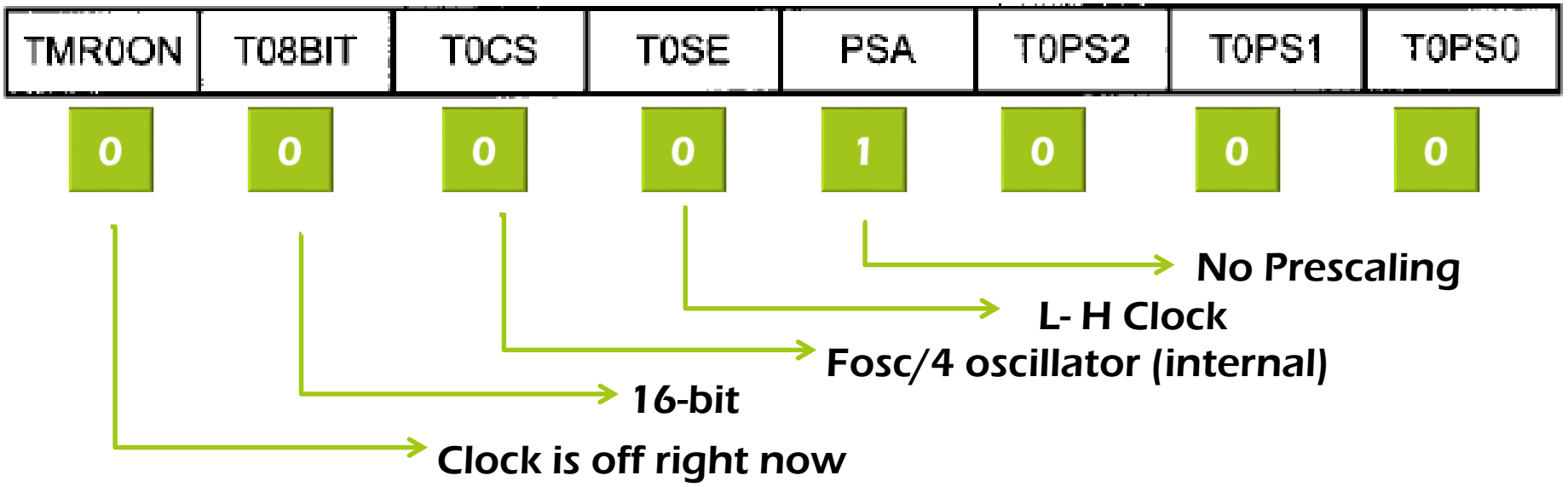
PSA D3 Timer0 prescaler assignment bit
1 = Timer0 clock input bypasses prescaler.
0 = Timer0 clock input comes from prescaler output.

T0PS2: T0PS0 D2 D1 D0 Timer0 prescaler selector

0 0 0 = 1:2	Prescale value (Fosc / 4 / 2)
0 0 1 = 1:4	Prescale value (Fosc / 4 / 4)
0 1 0 = 1:8	Prescale value (Fosc / 4 / 8)
0 1 1 = 1:16	Prescale value (Fosc / 4 / 16)
1 0 0 = 1:32	Prescale value (Fosc / 4 / 32)
1 0 1 = 1:64	Prescale value (Fosc / 4 / 64)
1 1 0 = 1:128	Prescale value (Fosc / 4 / 128)
1 1 1 = 1:256	Prescale value (Fosc / 4 / 256)

T0CON Reg (Timer0 Control)

8-bit register



Example 9-1

Find the value for T0CON if we want to program Timer0 in 16-bit mode, no prescaler. Use PIC18's Fosc/4 crystal oscillator for the clock source, increment on positive-edge.

Solution:

T0CON = 0000 1000 16-bit, Fosc/4 clock source, no prescaler, Timer0 off

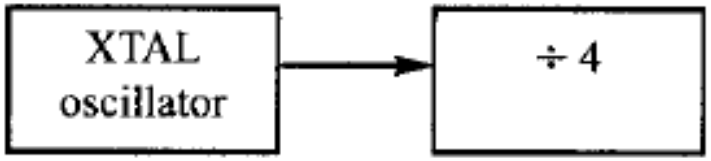
T0CS (Timer0 Clock Source)

- T0CS = 0
- Fosc/4 is the clock source
- Timer in this mode is used for time delay generation

Example 9-2

Find the timer's clock frequency and its period for various PIC18-based systems, with the following crystal frequencies. Assume that no prescaler is used.
(a) 10 MHz (b) 16 MHz (c) 4 MHz

Solution:
(a) 1/4 × 10 MHz = 2.5 MHz and T = 1/2.5 MHz = 0.4 µs
(b) 1/4 × 16 MHz = 4 MHz and T = 1/4 MHz = 0.25 µs
(c) 1/4 × 4 MHz = 1 MHz and T = 1/1 MHz = 1 µs



NOTE: PIC18 TIMERS USE 1/4 OF THE CRYSTAL FREQUENCY, IN ADDITION TO PRESCALER.

TMR0IF bit (Timer0 interrupt flag)

- Part of INTCON (Interrupt Control) register
- When timer reaches to maximum value of FFFF, it rolls over to 0000, and Timer0IF is set to 1

					TMR0IF		

TMR0IF D2 Timer0 interrupt overflow flag bit
0 = Timer0 did not overflow
1 = Timer0 has overflowed (FFFF to 0000, or FF to 00 in 8-bit mode).

The importance of TMR0IF: In 16-bit mode, when TMR0H:TMR0L overflows from FFFF to 0000 this flag is raised. In 8-bit, it is raised when the timer goes from FF to 00. We monitor this flag bit before we reload the TMR0H:TMR0L registers.

The other bits of this register are discussed in Chapter 11.

Timer0 Overflow Flag

- When timer reaches to maximum value of FFFF, it rolls over to 0000, and Timer0IF is set to 1, otherwise it is 0

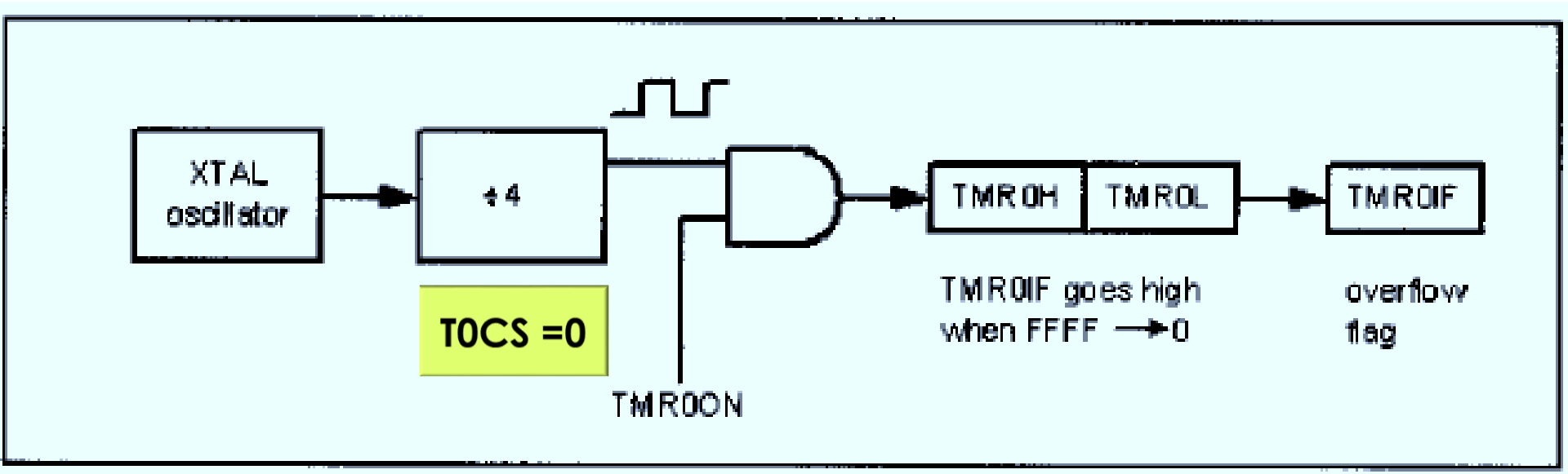


Figure 9-4. Timer0 Overflow Flag

Characteristics and operations of 16-bit mode

1. 16-bit timer, 0000 to FFFFH to be loaded into the registers TMR0H & TMR0L
2. After loading TMR0H and TMR0L, the timer must be started
3. After the timer started, it counts up until it reaches to its limit of FFFFH
4. When it rolls over from FFFFH to 0000, it sets high TMR0IF flag. The flag can be monitored. If raised, one option is, to stop the timer
5. When the timer rolls over, to repeat the process, TMR0H and TMR0L must be reloaded with the original value and TMR0IF flag must be set to 0.

Steps to program Timer0 in 16-bit mode to generate time delay

1. Load the value into the T0CON register
2. Load reg. TMR0H followed by reg. TMR0L with initial value (always load TMR0H first)
3. Start the timer with instruction

   BSF T0CON, TMR0ON
4. Keep monitoring the timer flag (TMR0IF) to see if it is raised.
5. Stop the timer
6. Clear the TMR0IF flag 3
7. Go Back to step 2

Timer0 Programming

Example 9-3

In the following program, we are creating a square wave of 50% duty cycle (with equal portions high and low) on the PORTB.5 bit. Timer0 is used to generate the time delay. Analyze the program.

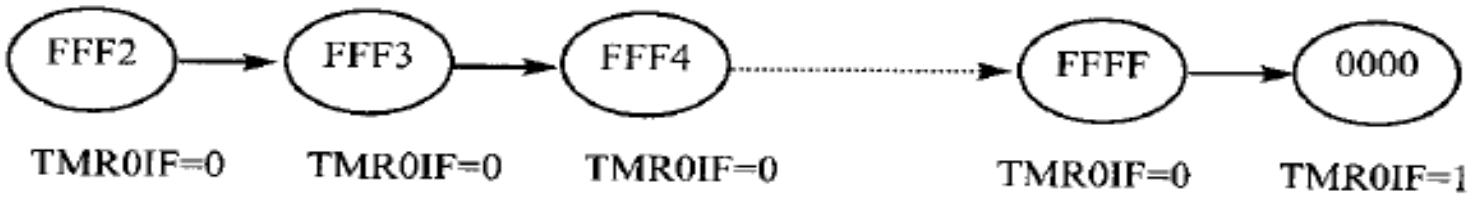

```
        BCF    TRISB,5          ;PB5 as an output
        MOVLW  0x08             ;Timer0,16-bit,int clk,no prescale
        MOVWF  T0CON            ;load T0CON reg.
HERE    MOVLW  0xFF             ;TMR0H = FFH, the high byte
        MOVWF  TMR0H            ;load Timer0 high byte
        MOVLW  0xF2             ;TMR0L = F2H, the low byte
        MOVWF  TMR0L            ;load Timer0 low byte
        BCF    INTCON, TMR0IF   ;clear timer interrupt flag bit
        BTG    PORTB,5          ;toggle PB5
        BSF    T0CON, TMR0ON    ;start Timer0
AGAIN   BTFSS  INTCON, TMR0IF   ;monitor Timer0 flag until
        BRA    AGAIN            ;it rolls over
        BCF    T0CON, TMR0ON    ;stop Timer0
        BRA    HERE             ;load TH, TL again
```

Timer0 Programming

In the above program notice the following steps:

1. T0CON is loaded.
2. FFF2H is loaded into TMR0H–TMR0L.
3. The Timer0 interrupt flag is cleared by the "`BCF INTCON, TMR0IF`" instruction.
4. PORTB.5 is toggled for the high and low portions of the pulse.
5. Timer0 is started by the "`BSF T0CON, TMR0ON`" instruction.
6. Timer0 counts up with the passing of each clock, which is provided by the crystal oscillator. As the timer counts up, it goes through the states of FFF3, FFF4, FFF5, FFF6, FFF7, FFF8, FFF9, FFFA, FFFB, and so on until it reaches FFFFH. One more clock rolls it to 0, raising the Timer0 flag (TMR0IF = 1). At that point, the "`BTFSS INTCON, TMR0IF`" instruction bypasses the "`BRA AGAIN`" instruction.
7. Timer0 is stopped by the instruction "`BCF T0CON, TMR0ON`", and the process is repeated.

Notice that to repeat the process, we must reload the TMR0L and TMR0H registers, and start the timer again.



Time Delay calculation using Timer

Example 9-4

In Example 9-3, calculate the amount of time delay generated by the timer. Assume that XTAL = 10 MHz.

Solution:

The timer works with the Fosc/4 clock; therefore, we have 10 MHz / 4 = 2.5 MHz as the timer frequency. As a result, each clock has a period of T = 1 /2.5 MHz = 0.4 μs. In other words, Timer0 counts up each 0.4 μs resulting in delay = number of counts × 0.4 μs.
The number of counts for the rollover is FFFFH – FFF2H = 0DH (13 decimal). However, we add one to 13 because of the extra clock needed when it rolls over from FFFF to 0 and raises the TMR0IF flag. This gives 14 × 0.4 μs = 5.6 μs for half the pulse. For the entire period the time delay generated by the timer is T = 2 × 5.6 μs = 11.2 μs.

Figure 9-6. Timer Delay Calculation for XTAL = 10 MHz with No Prescaler

- General formula for delay calculation
  - T = 4/(10MHz) = 0.4 usecond

(a) in hex	(b) in decimal
(FFFF - YYXX + 1) × 0.4 μs where YYXX are the TMR0H, TMR0L initial values respectively. Notice that YYXX values are in hex.	Convert YYXX values of the TMR0H, TMR0L register to decimal to get a NNNNN decimal number, then (65536 - NNNNN) × 0.4 μs

Example 9-6

Find the delay generated by Timer0 in the following code, using both of the methods of Figure 9-6. Do not include the overhead due to instructions.

```
       BCF   TRISB,5            ;PB5 as an output
       MOVLW 0x80               ;Timer0,16-bit,int clk, no prescale
       MOVWF T0CON
       BCF   INTCON,TMR0IF      ;clear Timer0 interrupt
HERE   MOVLW 0xB8               ;TMR0H = B8, the high byte
       MOVWF TMR0H
       MOVLW 0x3E               ;TMR0L = 3E, the low byte
       MOVWF TMR0L
       BSF   T0CON, TMR0ON      ;start Timer0
AGAIN  BTFSS INTCON, TMR0IF     ;monitor Timer0 flag until
       BRA   AGAIN              ;it rolls over
       BCF   T0CON, TMR0ON      ;stop Timer0
       BCF   INTCON, TMR0IF     ;clear Timer0 interrupt
       BTG   PORTB,5            ;toggle PB5
       BRA   HERE               ;load TH, TL again
```

Solution:

(a) (FFFF - B83E + 1) = 47C2H = 18,370 in decimal and 18,370 × 0.4 μs = 7.348 ms.

(b) Because TMR0H : TMR0L = B83EH = 47166 (in decimal) we have 65,536 - 47,166 = 18,370. This means that the timer counts from B83EH to FFFFH. This plus rolling over to 0 goes through a total of 18,370 clock cycles, where each clock is 0.4 μs in duration. Therefore, we have 18,370 × 0.4 μs = 7.348 ms as the width of the pulse.

Example 9-8

- Write a program to generate a square wave with a period of 10 ms on pin PORTB.3 (XALT=10 Mhz)
  - T = 10 ms
  - Time delay = 10ms/2 = 5 ms.
  - We need 5ms/0.4us = 12500 clocks
  - 12500 (Decimal) = 30D4 (Hex)
  - FFFF - 30D4 +1 =CF2C
  - TMR0H = CFH
  - TMR0L= 2CH

Example 9-8

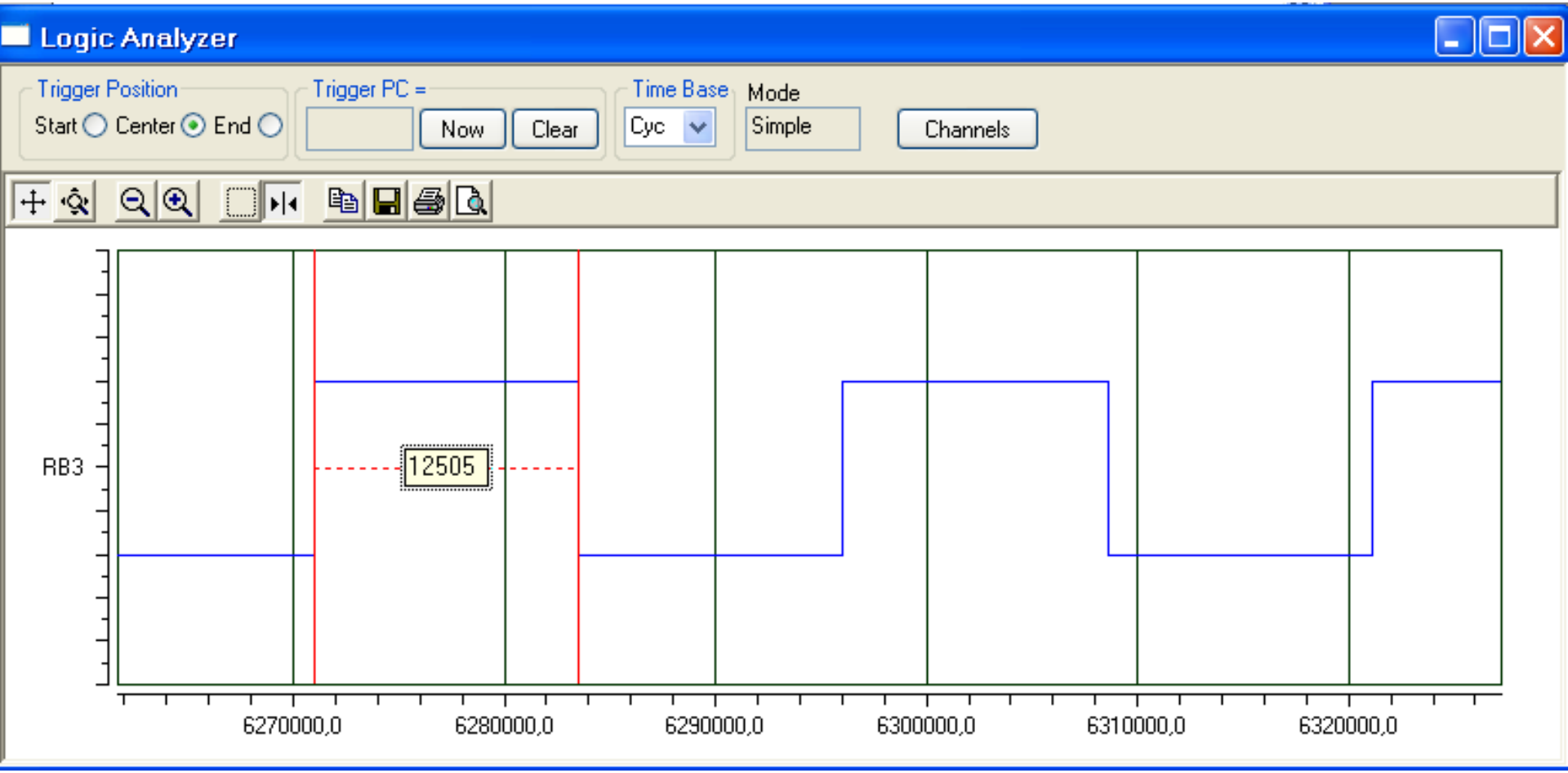
Assuming that XTAL = 10 MHz, write a program to generate a square wave with a period of 10 ms on pin PORTB.3.

Solution:

For a square wave with T = 10 ms we must have a time delay of 5 ms. Because XTAL = 10 MHz, the counter counts up every 0.4 μs. This means that we need 5 ms / 0.4 μs = 12,500 clocks. 65,536 – 12,500 = 53,036 = CF2CH. Therefore, we have TMR0H = CF and TMR0L = 2C.

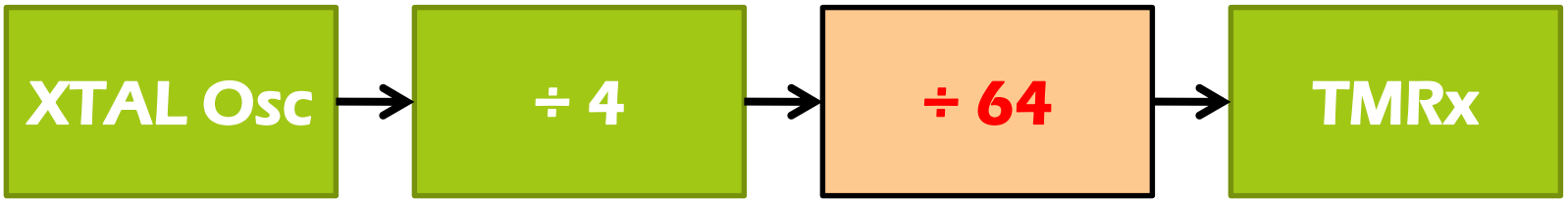
```
        BCF    TRISB,3              ;PB3 as an output
        MOVLW  0x08                 ;Timer0,16-bit,int clk,no prescale
        MOVWF  T0CON                ;load T0CON reg
HERE    MOVLW  0xCF                 ;TMR0H = CFH, the high byte
        MOVWF  TMR0H                ;load Timer0 high byte
        MOVLW  0x2C                 ;TMR0L = 2CH, the low byte
        MOVWF  TMR0L                ;load Timer0 low byte
        BCF    INTCON,TMR0IF        ;clear timer interrupt flag bit
        CALL   DELAY
        BTG    PORTB,3              ;toggle PB3
        BRA    HERE                 ;load TH, TL again
;----------delay using Timer0
DELAY   BSF    T0CON,TMR0ON         ;start Timer0
AGAIN   BTFSS  INTCON,TMR0IF        ;monitor Timer0 flag until
        BRA    AGAIN                ;it rolls over
        BCF    T0CON,TMR0ON         ;stop Timer0
        RETURN
```

Example 9-8, Cont.



Prescaler and generating larger delay

- The size of delay depend on
  - The Crystal frequency
  - The timer's 16-bit register.
- The largest timer happens when TMR0L=TMR0H=0
- Prescaler option is used to duplicate the delay by dividing the clock by a factor of 2,4, 8,16, 32,64 ,128,256
  - If T0CON=0000 0101, then T = 4*64/f

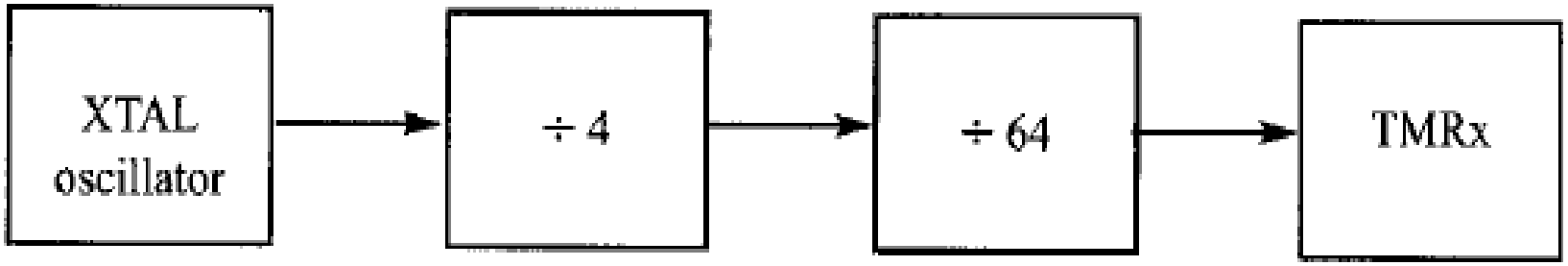


Prescaler and generating larger delay

Example 9-12

Find the timer's clock frequency and its period for various PIC18-based systems, with the following crystal frequencies. Assume that a prescaler of 1:64 is used.
(a) `10 MHz` (b) `16 MHz`

Solution:



(a) 1/4 × 10 MHz = 2.5 MHz and 1/64 × 2.5 MHz = 39062.5 Hz due to 1:64 prescaler and T = 1/39062.5Hz = 25.6 µs

(b) 1/4 × 16 MHz = 4 MHz and 1/64 × 4 MHz = 62500 Hz due to prescaler and T = 1/62500 Hz = 16 µs

Example 9-13

Examine the following program and find the time delay in seconds. Exclude the overhead due to the instructions in the loop. Assume XTAL = 10 MHz.

```
        BCF    TRISB,2              ;PB2 as an output
        MOVLW  0x05                 ;Timer0,16-bit,int clk,prescaler 64
        MOVWF  T0CON                ;load T0CON reg.
HERE    MOVLW  0x01                 ;TMR0H = 01H, the high byte
        MOVWF  TMR0H                ;load Timer0 high byte
        MOVLW  0x08                 ;TMR0L = 08H, the low byte
        MOVWF  TMR0L                ;load Timer0 low byte
        BCF    INTCON,TMR0IF        ;clear timer interrupt flag bit
        CALL   DELAY
        BTG    PORTB,2              ;toggle PB2
        BRA    HERE                 ;load TH, TL again
;----------delay using Timer0
DELAY   BSF    T0CON,TMR0ON         ;start Timer0
AGAIN   BTFSS  INTCON,TMR0IF        ;monitor Timer0 flag until
        BRA    AGAIN                ;it rolls over
        BCF    T0CON,TMR0ON         ;stop Timer0
        RETURN
```

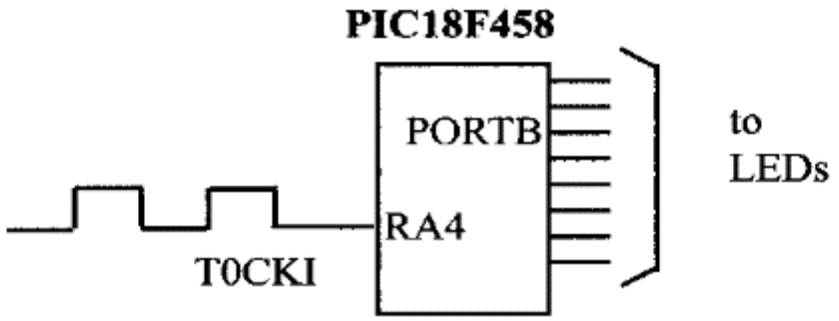
Solution:

TMR0H:TMR0L = 0108H = 264 in decimal and 65,536 – 264 = 65,272. Now 65,272 × 64 × 0.4 μs = 1.671 seconds, or from Example 9-12, we have 65,272 × 25.6 μs = 1.671 seconds.

SECTION 9.2: Counter Programming

- Used to counts event outside the PIC
  - Increments the TMR0H and TMR0L registers
- T0CS in T0CON reg determines the clock source
  - If T0CS = 1, the timer is used as a counter
  - Counts up as pulses are fed from pin RA4 (T0CKI)
  - What does T0CON=0110 1000 mean?
- If TMR1CS=1, the timer 1 counts up as clock pulses are fed into pin RC

PORTB is connected to 8 LEDs and input T0CKI to pulse.



Example 9-23

Assuming that clock pulses are fed into pin T0CKI, write a program for counter 0 in 8-bit mode to count the pulses and display the state of the TMR0L count on PORTB.

Solution:

```
      BSF   TRISA,RA4          ;PORTA.4 as an input for clock
      CLRF  TRISB              ;PORTB as an output
      MOVLW 0x68               ;Timer0, 8-bit,ext clk,no prescale
      MOVWF T0CON              ;load T0CON reg
HERE  MOVLW 0x0                ;TMR0L = 0
      MOVWF TMR0L              ;load Timer0
      BCF   INTCON,TMR0IF      ;clear timer interrupt flag bit
      BSF   T0CON,TMR0ON       ;start Timer0
AGAIN MOVFF TMR0L,PORTB        ;display the count on PORTB
      BTFSS INTCON,TMR0IF      ;monitor Timer0 flag until
      BRA   AGAIN              ;it rolls over
      BCF   T0CON,TMR0ON       ;stop Timer0
      GOTO  HERE
```

Example 9-28

Write a C18 program to toggle all the bits of PORTB continuously with some delay. Use Timer0, 16-bit mode, and no prescaler options to generate the delay.

Solution:

```
#include <p18f4580.h>
void T0Delay(void);
void main(void)
  {
  TRISB=0;                          //PORTB output port
  while(1)                          //repeat forever
    {
      PORTB=0x55;                   //toggle all bits of Port B
      T0Delay();                    //delay size unknown
      PORTB=0xAA;                   //toggle all bits of Port B
      T0Delay();
    }
  }

void T0Delay()
  {
    T0CON=0x08;                     //Timer0, 16-bit mode, no prescaler
    TMR0H=0x35;                     //load TH0
    TMR0L=0x00;                     //load TL0
    T0CONbits.TMR0ON=1;                   //turn on T0
    while(INTCONbits.TMR0IF==0);          //wait for TF0 to roll over
    T0CONbits.TMR0ON=0;                   //turn off T0
    INTCONbits.TMR0IF=0;            //clear TF0
  }
```

Example 9-29

Write a C18 program to toggle only the PORTB.4 bit continuously every 50 ms. Use Timer0, 16-bit mode, the 1:4 prescaler to create the delay. Assume XTAL = 10 MHz.

Solution:

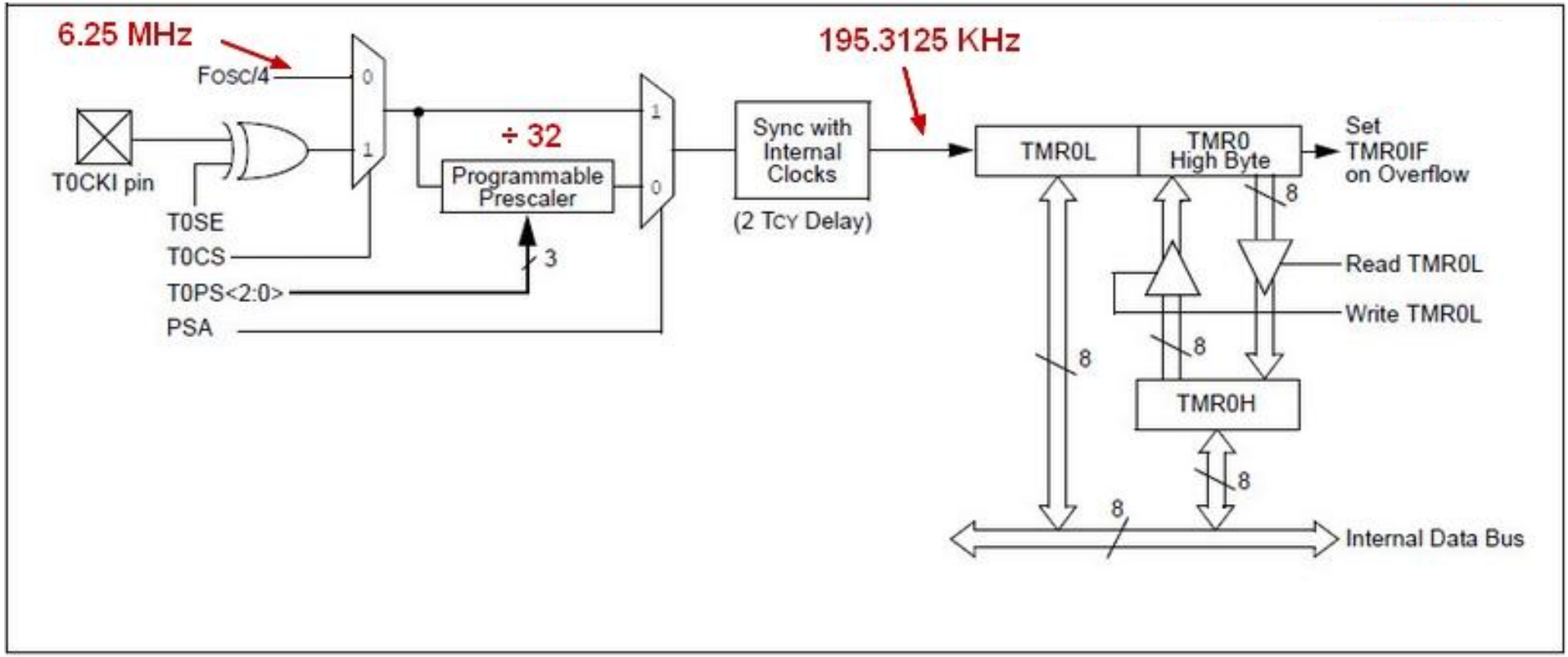
```
#include <p18f4580.h>
void T0Delay(void);
#define mybit PORTBbits.RB4
void main(void)
  {
   TRISBbits.TRISB4=0;
   while(1)
    {
      mybit^=1;                          //toggle PORTB.4
      T0Delay();                         //Timer0, mode 1 (16-bit)
    }
  }

void T0Delay()
  {
    T0CON=0x01;                  //Timer0, 16-bit mode, 1:4 prescaler
    TMR0H=0x85;                        //load TH0
    TMR0L=0xEE;                        //load TL0
    T0CONbits.TMR0ON=1;                //turn on Timer0
    while(INTCONbits.TMR0IF==0);       //wait for TF0 to roll over
    T0CONbits.TMR0ON=0;                //turn off Timer0
    INTCONbits.TMR0IF=0;               //clear TF0
  }
```

FFFFh–85EEH = 7A11H = 31249 + 1 = 31250

Timer delay = 31250 × 4 × 0.4 µs = 50 ms

Figure 9-5. Timer0 16-bit Block Diagram



Time Delay calculation using Timer

Example 9-5

Calculate the frequency of the square wave generated on pin PORTB.5.

Solution:

To get a more accurate timing, we need to add clock cycles due to the instructions in the loop.

```
                                        Cycles
            BCF   TRISB,5
            MOVLW 0x08
            MOVWF T0CON
            BCF   INTCON, TMR0IF

HERE        MOVLW 0xFF                    1
            MOVWF TMR0H                   1
            MOVLW -D'48'                  1
            MOVWF TMR0L                   1
            CALL  DELAY                   1
            BTG   PORTB,5                 1
            BRA   HERE                    1
;----------delay using Timer0
DELAY       BSF   T0CON, TMR0ON           1
AGAIN       BTFSS INTCON, TMR0IF          1
            BRA   AGAIN                   1
            BCF   T0CON, TMR0ON           1
            BCF   INTCON, TMR0IF          1
            RETURN                        1
                                         13
```

48 cycles delayed

$T = 2 \times (48 + 13) \times 0.4\ \mu s = 48.8\ \mu s$ and $F = 20.491$ kHz.