Microprocessors and Microcontrollers

Timers & Counters

EE3954

by
Maarten Uijt de Haag, Tim Bambeck
Timers and Counters

- PIC Microcontroller has three Timers/Counters:
  - **Timer 0**: 8-bit register TMR0
    - Described in section 11 of RM
  - **Timer 1**: 16-bit register TMR1H | TMR1L
    - Described in section 12 of RM
  - **Timer 2**: 8-bit register TMR2
    - Described in section 13 of RM
Timer 0

- **Timer mode:**
  - a register (TMRO) will be incremented every instruction cycle (without prescaler)

- **Counter mode:**
  - a register (TMRO) will be incremented every time either a rising or falling edge occurs at pin TOCKI

- **Register TMRO (0x01 in banks 0 & 2):**
  - can be read by the user at any time in the program
**Timer 0 Architecture**

*Figure 11-1: Timer0 Block Diagram*

Note 1: T0CS, T0SE, PSA, PS2:PS0 (OPTION_REG<5:0>).
2: The prescaler is shared with Watchdog Timer (refer to Figure 11-6 for detailed block diagram).

**Register 11-1: OPTION_REG Register**  
(at address 0x01 in Bank 1 & 3)

<table>
<thead>
<tr>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>RBPU</td>
<td>INTEDG</td>
<td>T0CS</td>
<td>T0SE</td>
<td>PSA</td>
<td>PS2</td>
<td>PS1</td>
<td>PS0</td>
</tr>
</tbody>
</table>

bit 7

bit 0
Timer 0

Writing to TMRO (0x01 banks 0 & 2)

A write to TMRO will cause a 2 instruction cycle ($2T_{CY}$) inhibit !!!!!
A TIMER 0 interrupt is generated when the TIMER 0 overflows from 0xFF to 0x00.
**Timer 0**

**Interrupt**

See Reference Manual pp.171

<table>
<thead>
<tr>
<th>Register 8-1: INTCON Register</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>GIE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PEIE (3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T0IE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTE (2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RBIE (1, 2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T0IF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTF (2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RBIF (1, 2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(at location 0x0B in all banks)
Timer 0

Interrupt - Example

Blink InfraRed (IR) LED at a rate of 5kHz

5kHz => T = 0.2msec => ON for 0.1 ms  
OFF for 0.1 ms

If run at maximum speed and selecting the microcontroller’s oscillator as the timer’s input, the counter/timer gets incremented every 1μsec.

T = 0.1msec = 100 x 1μsec
Timer 0
Interrupt - Example

\[ T = 0.1 \text{msec} = 100 \times 1 \mu\text{sec} \]

So make sure that it takes Timer 0 100 counts before it overflows:

Pick the initial value of Timer 0 as 256 - 100 = 156 = 0x9C
( add 2 to this to account for two-cycle delay) 156+2 = 158 = 0x9E
**Timer 0**

*Example - Setup*

**Register 11-1: OPTION_REG Register**

<table>
<thead>
<tr>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>RBPU</td>
<td>INTEDG</td>
<td>T0CS</td>
<td>T0SE</td>
<td>PSA</td>
<td>PS2</td>
<td>PS1</td>
<td>PS0</td>
<td></td>
</tr>
</tbody>
</table>

bit 7

**Figure 11-1: Timer0 Block Diagram**

Note 1: T0CS, T0SE, PSA, PS2:PS0 (OPTION_REG<5:0>).

2: The prescaler is shared with Watchdog Timer (refer to Figure 11-6 for detailed block diagram).

**OPTION_REG:** \(B' \text{XX0X1XXX}': B'1101\ 1111'\)
Timer 0
Interrupt - Example - Step 1 & 3

org 0x000
goto INIT
org 0x004
goto T0_ISR

INIT:
clrf STATUS ; access
bsf STATUS,5 ; bank 1 (for OPTION and TRIS)
clrf TRISB, F ; pin 0 (and others) of PORTB: output
movlw B'11011111'; setup the OPTION register
movwf OPTION_REG
bcf STATUS,5 ; access bank 0
movlw b'00100000'; enable the Timer 0 interrupt
movwf INTCON ; clears TOIF also
movlw D'158'; initialize the Timer 0 counter
movwf TMR0
bsf INTCON,7 ; enable global interrupts
...
Timer 0
Interrupt - Example - Step 2

MAIN:

... 

... 

goto MAIN

TO_ISR:

    comf PORTB,F ; Toggles LED on RB0
    movlw D'158' ; make sure to put the original
    movwf TMRO ; value back in the TMRO register
    bcf INTCON,2 ; clear the Timer 0 interrupt flag
    retfie ; return to whatever you were doing
A Prescaler divides the input clock (Fosc/4 or the clock on the pin) by another factor:

<table>
<thead>
<tr>
<th>PS2:PS0: Prescaler Rate Select bits</th>
<th>TMR0 Rate</th>
<th>WDT Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>1 : 2</td>
<td>1 : 1</td>
</tr>
<tr>
<td>001</td>
<td>1 : 4</td>
<td>1 : 2</td>
</tr>
<tr>
<td>010</td>
<td>1 : 8</td>
<td>1 : 4</td>
</tr>
<tr>
<td>011</td>
<td>1 : 16</td>
<td>1 : 8</td>
</tr>
<tr>
<td>100</td>
<td>1 : 32</td>
<td>1 : 16</td>
</tr>
<tr>
<td>101</td>
<td>1 : 64</td>
<td>1 : 32</td>
</tr>
<tr>
<td>110</td>
<td>1 : 128</td>
<td>1 : 64</td>
</tr>
<tr>
<td>111</td>
<td>1 : 256</td>
<td>1 : 128</td>
</tr>
</tbody>
</table>

Figure 11-1: Timer0 Block Diagram

Note 1: T0CS, T0SE, PSA, PS2:PS0 (OPTION_REG<5:0>).

2: The prescaler is shared with Watchdog Timer (refer to Figure 11-6 for detailed block diagram).
Suppose we want to change our earlier example’s interrupt rate from 10kHz (= 0.1msec) to 2.5kHz (0.4msec).

This could be accomplished by dividing the input clock by four or selecting bit value 001 for PS2, PS1, and PS0. Make sure to also select the pre-scalar by clearing bit 3 of OPTION_REG (PSA).

```
org 0x0000
goto INIT
org 0x0004
goto T0_ISR

INIT:
clrf STATUS ; access
bsf STATUS,5 ; bank 1
clrf TRISB, F ; pin 0 (an others) of PORTB: output
movlw B’00000001’ ; setup the OPTION register
movwf OPTION_REG
bcf STATUS,5 ; access bank 0
...
```
Timer 0

Prescaler

• The Prescaler is shared with the WatchDog timer ... but more about that one later.
Timer 0

Counter Mode of Operation

Figure 11-1: Timer0 Block Diagram

Note 1: T0CS, T0SE, PSA, PS2:PS0 (OPTION_REG<5:0>).
2: The prescaler is shared with Watchdog Timer (refer to Figure 11-6 for detailed block diagram).
Timer 0

Counter Mode: Synchronization

Figure 11-5: Timer0 Timing with External Clock

External Clock Input or Prescaler output (2)

External Clock/Prescaler Output after sampling

Increment Timer0 (Q4)

Timer0

Q1 | Q2 | Q3 | Q4
---|----|----|----
T0

Q1 | Q2 | Q3 | Q4
----|----|----|----
T0 + 1

Q1 | Q2 | Q3 | Q4
----|----|----|----
T0 + 2

Note 1: Delay from clock input change to Timer0 increment is 3Tosc to 7Tosc. (Duration of Q = Tosc).
Therefore, the error in measuring the interval between two edges on Timer0 input = ±4Tosc max.
2: External clock if no prescaler selected, Prescaler output otherwise.
3: The arrows indicate the points in time where sampling occurs.
Timer 0
Counter Mode of Operation

OPTION_REG

<table>
<thead>
<tr>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>R8PU</td>
<td>INTEDG</td>
<td>T0CS</td>
<td>T0SE</td>
<td></td>
</tr>
</tbody>
</table>

bit 7

T0SE = 0:

or

T0SE = 1:

External Clock

TOCKI

PIC16F877

OSC1

CPU Clock
Timer 0

Counter Mode Example

Count the number of times someone pushes a button given the following hardware setup.

If button is pushed the voltage on the TOCKI pin goes from $V_{DD}$ to $V_{SS}$.

In other words we want to detect a falling edge: TOSE = 1
Timer 0

Counter Mode Example

Register 11-1: OPTION_REG Register

<table>
<thead>
<tr>
<th>RBPU&lt;1&gt;</th>
<th>INTEDG</th>
<th>T0CS</th>
<th>T0SE</th>
<th>PSA</th>
<th>PS2</th>
<th>PS1</th>
<th>PS0</th>
</tr>
</thead>
<tbody>
<tr>
<td>R/W-1</td>
<td>R/W-1</td>
<td>R/W-1</td>
<td>R/W-1</td>
<td>R/W-1</td>
<td>R/W-1</td>
<td>R/W-1</td>
<td>R/W-1</td>
</tr>
</tbody>
</table>

bit 7

Figure 11-1: Timer0 Block Diagram

Note 1: T0CS, T0SE, PSA, PS2:PS0 (OPTION_REG<5:0>).
2: The prescaler is shared with Watchdog Timer (refer to Figure 11-6 for detailed block diagram).

OPTION_REG: B’ XX111XXX’ : B’ 11111111’
Timer 0
Counter Mode Example

```
org 0x000

INIT:
   clrf TMRO,F ; clear the counter (2Tcy inhibit)
   clrf STATUS ; access
   bsf STATUS,RPO ; bank 1
   movlw B'11111111' ; setup the OPTION register
   movwf OPTION_REG
   bcf STATUS,RPO ; access bank 0

MAIN:
   ... 
   ... 
   ... 
   movf TMRO,W ; read the number of button pushes 
   ... 
   ... 
   ... 
```
Timer 0

During microcontroller SLEEP
Timer 0 is shutdown
Timer 1

• **Timer mode:**
  - a register \((TMR1H \mid TMR1L)\) will be incremented every instruction cycle (without prescaler)

• **Counter mode:**
  - a register \((TMR1H \mid TMR1L)\) will be incremented every time a rising edge occurs at pin T1CKI

• **Registers** \(TMR1H \& TMR1L\):
  - can be read by the user at any time in the program
Timer 1
Architecture

Figure 12-1: Timer1 Block Diagram

Set TMR1IF flag bit on Overflow

TMR1

CLR

CCP Special Trigger

TMR1ON on/off

Synchronized clock input

0

1

T1SYNC

Prescaler
1, 2, 4, 8

Synchronize det

SLEEP input

T1OSO/ T1CKI

T1OSC

T1OSI

T1OSCE
able Oscillator(1)

Fosc/4 Internal Clock

T1CKPS1:T1CKPS0

TMR1CS

Note 1: When the T1OSCEN bit is cleared, the inverter and feedback resistor are turned off. This eliminates power drain.

Register 12-1: T1CON: Timer1 Control Register

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>U-0</td>
<td>U-0</td>
</tr>
<tr>
<td>T1CKPS1</td>
<td>T1CKPS0</td>
</tr>
<tr>
<td>T1OSCE</td>
<td>T1SYNC</td>
</tr>
<tr>
<td>TMR1CS</td>
<td>TMR1ON</td>
</tr>
</tbody>
</table>

Timers.24
Timer 1

Architecture

- **Timer**
- **Counter**
  - **Synchronous**: The external clock input is synchronized with the rising edge of the CPU clock just like with Timer0
  - **Asynchronous**: No synchronization takes place, just increment counter at any rising edge of external clock input
Timer 1

What about reading out the Timer’s 2 Bytes

<table>
<thead>
<tr>
<th>TMR1</th>
<th>Sequence 1</th>
<th>Sequence 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Action</td>
<td>TMPH:TMPL</td>
</tr>
<tr>
<td>04FFh</td>
<td>READ TMR1L</td>
<td>xxxxh</td>
</tr>
<tr>
<td>0500h</td>
<td>Store in TMPL</td>
<td>xxFFh</td>
</tr>
<tr>
<td>0501h</td>
<td>READ TMR1H</td>
<td>xxFFh</td>
</tr>
<tr>
<td>0502h</td>
<td>Store in TMPH</td>
<td>05FFh</td>
</tr>
</tbody>
</table>

Oops

Oops

So, what is the solution?
Timer 1

What about reading out the Timer’s 2 Bytes

; All interrupts are disabled
movf TMR1H, W       ; Read high byte
movwf TMPH
movf TMR1L, W       ; Read low byte
movwf TMPL
movf TMR1H, W       ; Read high byte
subwf TMPH, W       ; Sub 1st read with 2nd read
btfsc STATUS, Z     ; Is result = 0
goto CONTINUE       ; Good 16-bit read

; TMR1L may have rolled over between the read of the high and low bytes.
; Reading the high and low bytes now will read a good value.

movf TMR1H, W       ; Read high byte
movwf TMPH
movf TMR1L, W       ; Read low byte
movwf TMPL

; Re-enable the Interrupt (if required)
CONTINUE ...         ; Continue with your code
Timer 1

What about writing out the Timer’s 2 Bytes

; All interrupts are disabled
clr TMR1L ; Clear Low byte, Ensures no rollover into TMR1H
movlw HI_BYTE ; Value to load into TMR1H
movwf TMR1H, F ; Write High byte
movlw LO_BYTE ; Value to load into TMR1L
movwf TMR1L, F ; Write Low byte

; Re-enable the Interrupt (if required)
CONTINUE ... ; Continue with your code
# Timer 1

## Associated Registers

### Table 6-2: Registers Associated with Timer1 as a Timer/Counter

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Value on: POR, BOR</th>
<th>Value on: all other RESETS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0Bh, 8Bh, 10Bh, 18Bh</td>
<td>INTCON</td>
<td>GIE</td>
<td>PEIE</td>
<td>T0IE</td>
<td>INTE</td>
<td>RBIE</td>
<td>T0IF</td>
<td>INTF</td>
<td>RBIF</td>
<td>0000 000x</td>
<td>0000 000u</td>
</tr>
<tr>
<td>0Ch</td>
<td>PIR1</td>
<td>ADIF</td>
<td>RCIF</td>
<td>TXIF</td>
<td>SSPIF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
<td></td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
<tr>
<td>8Ch</td>
<td>PIE1</td>
<td>ADIE</td>
<td>RCIE</td>
<td>TXIE</td>
<td>SSPIE</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1IE</td>
<td></td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
<tr>
<td>0Eh</td>
<td>TMR1L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxxxxxx</td>
<td>uuuuu uuuuu</td>
</tr>
<tr>
<td>0Fh</td>
<td>TMR1H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxxxxxx</td>
<td>uuuuu uuuuu</td>
</tr>
<tr>
<td>10h</td>
<td>T1CON</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>T1CKPS1</td>
<td>T1CKPS0</td>
<td>T1OSCN</td>
<td>T1SYNC</td>
<td>TMR1CS</td>
<td>TMR1ON</td>
</tr>
</tbody>
</table>

**Legend:**  
- `x` = unknown, `u` = unchanged, `-` = unimplemented, read as '0'. Shaded cells are not used by the Timer module.

**Note 1:** Bits PSPIE and PSPIF are reserved on the PIC16F873/876; always maintain these bits clear.

To turn the Timer 1 On
**Timers.30**

# Timer 1

**Associated Registers (Interrupts)**

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Value on: POR, BOR</th>
<th>Value on all other RESETS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0Bh, 8Bh, 10Bh, 18Bh</td>
<td>INTCON</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>GIE, PEIE</td>
<td>0000 000x</td>
<td>0000 000u</td>
</tr>
<tr>
<td>0Ch</td>
<td>PIR1</td>
<td>ADIF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>T0IE</td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
<tr>
<td>8Ch</td>
<td>PIE1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RCIE</td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
<tr>
<td>0Eh</td>
<td>TMR1L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TXIF</td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
<tr>
<td>0Fh</td>
<td>TMR1H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SSPIF</td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
<tr>
<td>10h</td>
<td>T1CON</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>T1CKPS1</td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
</tbody>
</table>

Legend:  
- $x$ = unknown  
- $u$ = unchanged  
- $-$ = unimplemented, read as '0'. Shaded cells are not used by the Timer module.  

**Note 1:** Bits PSPIE and PSPIF are reserved on the PIC16F873/876; always maintain these bits clear.

---

**Peripheral Interrupt Enable**  
**Flag & Enable bits**
FIGURE 12-9: INTERRUPT LOGIC

For PIC16F877
(See section 12 of DS)
Timer 1
Example with Interrupt

We want to blink the light at a frequency of 1 Hz without using a Timing Loop (Delay)

1 Hz => $T = 1.0$ seconds => ON for 0.5 seconds
OFF for 0.5 seconds

@ $f_{osc} = 4$MHz => $f_{cy} = 1$MHz => $T_{cy} = 1$µsec

0.5 seconds = 500,000 instruction cycles

Timer 1 can count from 0 to 65,536 (=64k)

How do we increase this?

Use the pre-scaler !!
Timer 1

Example with Interrupt

65,536 * 1 = 65,536 (1us)
65,536 * 2 = 131,072 (2us)
65,536 * 4 = 262,144 (4us)
65,536 * 8 = 524,288 (8us)

500,000/8 = 62,500

(62,500 * 8us = 0.5s)

So, if we can cause Timer 1 to overflow after 62,500 increments we will cause an Interrupt every ~0.5 seconds

Thus, start Timer 1 at 65,536 - 62,500 = 3036 = 0x0BDC

Initialize TMR1H with 0x0B and TMR1L with 0xDC
**Timer 1**

**Setup**

**Figure 12-1: Timer1 Block Diagram**

Note 1: When the T1OSCEN bit is cleared, the inverter and feedback resistor are turned off. This eliminates power drain.

<table>
<thead>
<tr>
<th>T1CON:</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td></td>
<td></td>
<td>T1CKPS1</td>
<td>T1CKPS0</td>
<td>T1OSCEN</td>
<td>T1SYNC</td>
<td>TMR1CS</td>
<td>TMR1ON</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Timer 1

Interrupt - Example - Step 1

```
org 0x000
goto INIT
org 0x004
goto T1_ISR

INIT:
clrf STATUS ; access
bsf STATUS,RPO ; bank 1
bcf TRISB, 0 ; make pin 0 of PORTB: output
bcf STATUS,RPO ; access bank 0

; Initialize Timer 1
movlw B'00110100' ; setup the T1CON register
movwf T1CON ; leave Timer 1 off for now
movlw 0x0B ; Value to load into TMR1H
movwf TMR1H, F ; Write High byte
movlw 0xDC ; Value to load into TMR1L
movwf TMR1L, F
...
```
Timer 1
Interrupt - Example - Step 1 (Continued)

; Initialize the interrupts:

bsf STATUS, RP0 ; select bank 1
bsf PIE1, TMR1IE ; enable Timer 1 interrupt
bcf STATUS, RP0 ; select bank 0 again
bcf PIR1, TMR1IF ; clear the interrupt flag
movlw B’11000000’ ; Turn on GIE and PEIE
movwf INTCON
bsf T1CON, TMR1ON ; TURN on Timer 1

Main:

goto Main

end
**Timer 1**

*Interrupt - Example - Step 2*

**MAIN:**

```
    goto MAIN
```

**T1_ISR:**

```
    comf PORTB,F ; Toggles LED on RB0
    clrf TMR1L ; Clear Low byte, Ensures no rollover into TMR1H
    movlw 0x0B ; Value to load into TMR1H
    movwf TMR1H, F ; Write High byte
    movlw 0xDC ; Value to load into TMR1L
    movwf TMR1L, F
    bcf PIR1, TMR1IF ; clear the interrupt flag
    retfie ; return to whatever you were doing
```
Timer 1

During microcontroller SLEEP
Timer 1 is ON (in asynchronous mode)
and can wake up the microcontroller
Timer 2

- Timer: 8-bit
- With prescaler and postscaler
- Shutdown during SLEEP

- Timer 2 increments until it matches register PR2 at which time it resets to 0x00, and increments the postscaler.
Figure 13-1: Timer2 Block Diagram

Note: TMR2 register output can be software selected by the SSP Module as a baud clock.

Register 13-1: T2CON: Timer2 Control Register

<table>
<thead>
<tr>
<th>bit 7</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>TOUTPS3</td>
<td>TOUTPS2</td>
<td>TOUTPS1</td>
<td>TOUTPS0</td>
<td>TMRON</td>
<td>T2CKPS1</td>
<td>T2CKPS0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Timers.40
Timer 2

In PIC16F877

**TABLE 7-1: REGISTERS ASSOCIATED WITH TIMER2 AS A TIMER/COUNTER**

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Value on: POR, BOR</th>
<th>Value on all other RESETS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0Bh,8Bh, 10Bh,18Bh</td>
<td>INTC0</td>
<td>GIE</td>
<td>PEIE</td>
<td>T0IE</td>
<td>INTE</td>
<td>RBIE</td>
<td>T0IF</td>
<td>INTF</td>
<td>RBIF</td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
<tr>
<td>0Ch</td>
<td>PIR1</td>
<td>PSPIE(1)</td>
<td>ADIF</td>
<td>RCIF</td>
<td>TXIF</td>
<td>SSPIF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
<tr>
<td>8Ch</td>
<td>PIE1</td>
<td>PSPIE(1)</td>
<td>ADIE</td>
<td>RCIE</td>
<td>TXIE</td>
<td>SSPIE</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1IE</td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
<tr>
<td>11h</td>
<td>TMR2</td>
<td>Timer2 Module’s Register</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
<tr>
<td>12h</td>
<td>T2CON</td>
<td>TOUTPS3</td>
<td>TOUTPS2</td>
<td>TOUTPS1</td>
<td>TOUTPS0</td>
<td>TMR2ON</td>
<td>T2CKPS1</td>
<td>T2CKPS0</td>
<td>-000 0000</td>
<td>-000 0000</td>
<td></td>
</tr>
<tr>
<td>92h</td>
<td>PR2</td>
<td>Timer2 Period Register</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1111 1111</td>
<td>1111 1111</td>
</tr>
</tbody>
</table>

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by the Timer2 module.

**Note 1:** Bits PSPIE and PSPIF are reserved on the PIC16F873/876; always maintain these bits clear.

Prescaler and Postscaler Counters clear when:
- any write occurs to TMR2 register.
- any write occurs to T2CON register. (TMR2 does not clear).