Chapter 5b: Timers

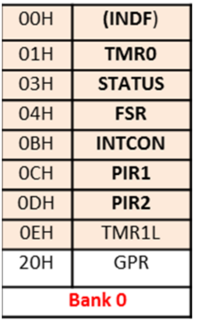
Australia's Global University | Faculty of Engineering | School of Electrical Engineering and Telecommunications

Professor Eliathamby Ambikairajah

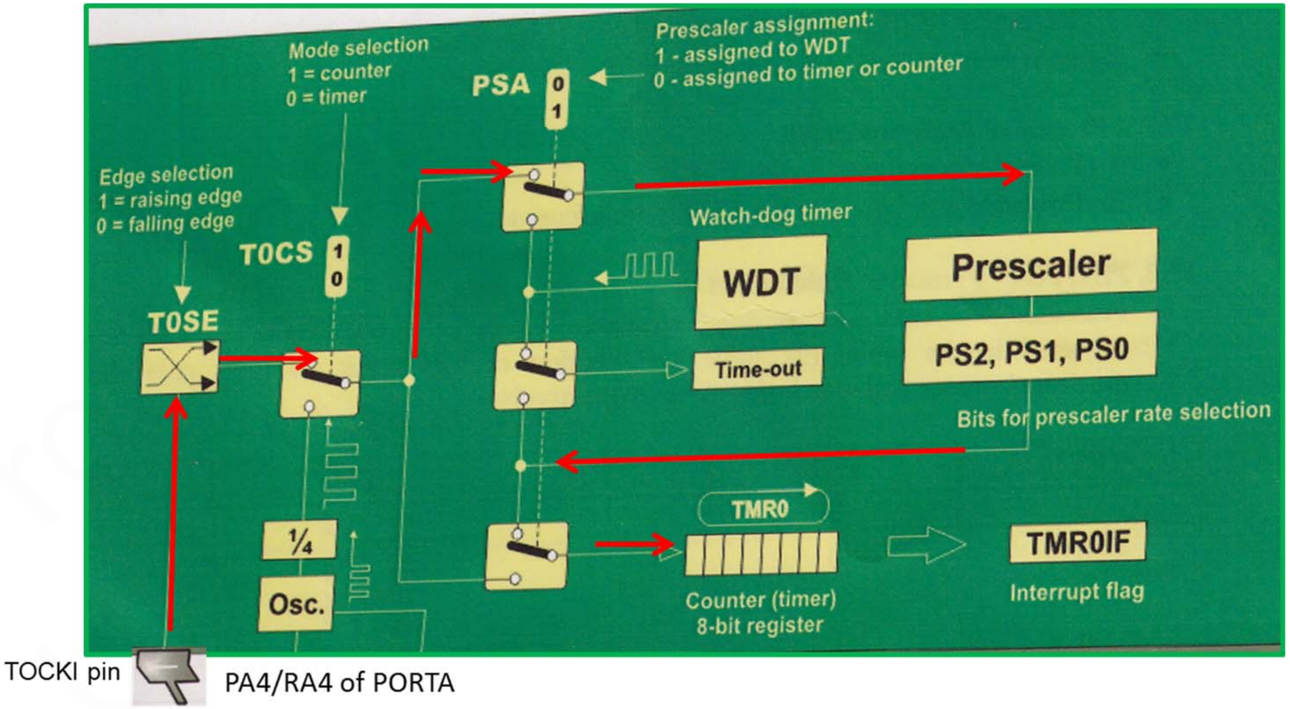
Head of School of Electrical Engineering and Telecommunications, UNSW, Sydney

Timers

- ✓ There are three completely independent timers/counters available in PIC16f886 micro controllers and they are marked as **TMR0**, TMR1 and TMR2
- ✓ TMR0 has a wide range of applications in practice: (a) Time measurement (b) Counting external pulses (c) Generating pulses of arbitrary duration
- ✓ **TMR0 operation**: When used as a timer, the Timer0 module can be used as either an **8-bit timer** or an **8-bit counter**
- ✓ Timer0 appears as register TMR0 at memory location H’01’ in data memory bank 0
- ✓ TMR0 is configurable and controlled by a number of bits that appear in the OPTION register



00H	(INDF)
01H	**TMR0**
03H	**STATUS**
04H	**FSR**
0BH	**INTCON**
0CH	**PIR1**
0DH	**PIR2**
0EH	TMR1L
20H	GPR
Bank 0	



Timers

- ✓ When used as a 8-bit timer, the Timer0 module will increment every instruction cycle.
  - Timer mode is selected by clearing the TOCS bit (bit5) of OPTION register to ‘0’
- ✓ When used as a 8-bit counter, the Timer0 module will increment on every rising or falling edge of the TOCKI pin (PA4/RA4 of PORTA).
  - The incrementing edge is determined by the T0SE bit of the OPTION register.
  - The counter mode is selected by setting the T0CS bit (bit5) of Option register to ‘1’.

OPTION Register – Data memory bank 1 and 3

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

R/W Read/Writable bit; (1) After rest, bit is set;

T0CS – TMR0 Clock Select bit
1 – Transition on TOCK1 pin (PA4/RA4 of PORTA); i.e clock mode
0 – Internal instruction cycle clock (Fosc/4) i.e timer mode

TOSE – TMR0 Source Edge Select bit
1 – increment on high to low transition on TOCKI pin;
0 – increment on low to high transition on TOCKI pin;

Option Register

OPTION Register – Data memory bank 1 and 3

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

R/W Read/Writable bit; (1) After rest, bit is set;

$\overline{\text{RBPU}}$ – PORTB Pull-up Enable bit
1 – PORTB pull-ups are disabled
0 – PORTB pull-ups are enabled by individual PORT latch values

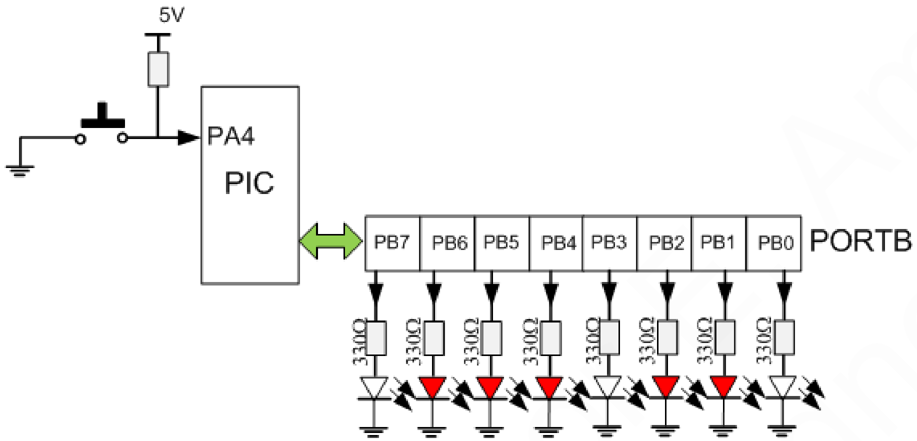
PSA – Prescaler Assignment bit
1 – Prescaler is assigned to the WDT (watchdog Timer);
0 – Prescaler is assigned to the Timer0 module

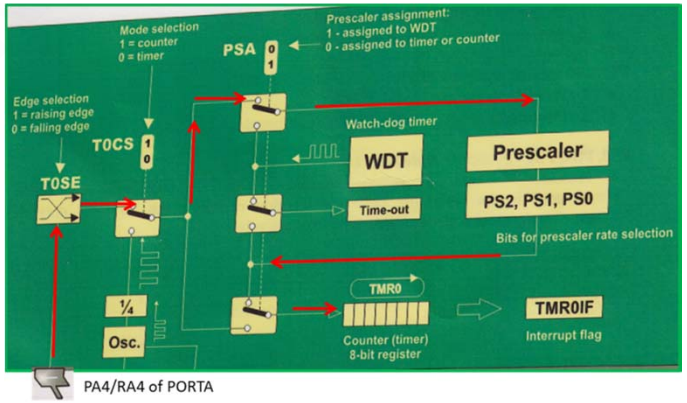
PS0,PS1,PS2 – Prescaler Rate Select Bits
Inspection of these bits in table below shows that they allow a choice of frequency divisions of incoming clock signal.

PS2	PS1	PS0	TMR0	WDT
0	0	0	1:2	1:1
0	0	1	1:4	1:2
0	1	0	1:8	1:4
0	1	1	1:16	1:8
1	0	0	1:32	1:16
1	0	1	1:64	1:32
1	1	0	1:128	1:64
1	1	1	1:256	1:128

TMR0: Counter Mode

- A push button is connected to the PA4 pin of PORTA as shown in the diagram below.
- Write an assembly language program to continuously count (use TMR0 to count the pulses) the pressing and releasing of the button (one count) and continuously display the counter value on the LEDs connected to PORTB.
- You may assume that the switch is debounced.

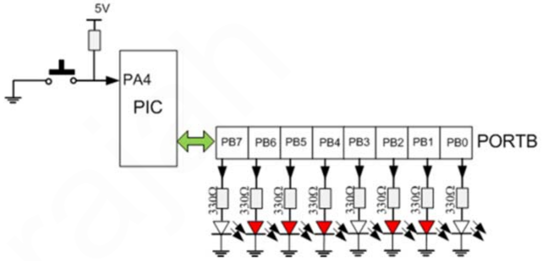




OPTION Register – Data memory bank 1 and 3

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

- To configure Timer0 we need to select its external input (i.e TOCS of Option Reg must be 1)
- We will count at the rising edge (i.e TOSE of the Option Reg must be 0)
- We do not want the prescaler as we want to count the exact number of switch presses. i.e PSA of the Option Reg = 1; Hence the values of PS2, PS1, PS0 do not matter as we do not need WDT in this program.
- The final value of the Option Reg is B'00101000'



; Event Counting using Timer0

```
;Intialisation

start       BANKSEL     ANSEL       ;select memory bank containing ANSEL Register
            clrf        ANSEL       ;set PORTA to digital by clearing ANSEL Register
            clrf        ANSELH      ;set PORTA B to digital by clearing ANSELH Register
;
            BANKSEL     TRISA       ;select memory bank containing TRISA and TRISB Registers
            movlw       B'00010000' ;PA4 =1
            movwf       TRISA       ;make PORTA  (PA4 as input)
            clrf        TRISB       ;make PORTB all outputs
            BANKSEL     PORTB       ;select memory bank containing PORTA and PORTB
                                    ;Registers
            clrf        PORTB       ;reset PORTB (turn off all LEDS)
            clrf        TMR0        ;As we are in Bank 0, we may as well clear the contents of the
                                    ;Timer0
;
            BANKSEL     OPTION_REG ;select Option register to use TMR0
            Movlw       B'00101000'  ;setup TMR0 for external (+) edge (TOCS=1) input and no
                                     ;Prescaler
            movwf       OPTION_REG
            BANKSEL     PORTA       ;select memory bank containing PORTA Register (bank 0)
;
;Main program starts here
Main        movf        TMR0,0      ; move the content of TMR0 to W register
            movwf       PORTB       ;sent the content of W register to LEDS
            goto        main        ;
            END
```

00H	Indirect address **(INDF)**
01H	**TMR0**
02H	PCL
03H	STATUS
04H	FSR
05H	PORTA
06H	PORTB
07H	PORTC
20H: 7FH	(GPR) 96 Bytes
Bank 0	

180H	Indirect address
181H	Option_REG
182H	PCL
183H	**STATUS**
184H	FSR
185H	SRCON
186H	TRISB
187H	BAUDCTL
188H	**ANSEL**
189H	ANSELH
18AH	PCLATH
190H - 1FFH	GPR 16 bytes
Bank 3	

OPTION Register – Data memory bank 1 and 3

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

R/W Read/Writable bit; (1) After rest, bit is set;

TMR0: Timer Mode

- ✓ Timer mode is selected by clearing the TOCS bit (OPTION register, bit 5 =0).
- ✓ In timer mode, the TMR0 register increments every instruction cycle. As an 8-bit register, TMR0 can count from 00 to FF (255). When it reaches its maximum value, FF, and is incremented further, it rolls over to 00.
- ✓ This register overflow is recorded by the T0IF (Timer0 Interrupt Flag) bit of the INTCON (Interrupt Control) register by being set to 1.
- ✓ The T0IF bit set can trigger an interrupt (known as Timer0 Interrupt), if enabled.
- ✓ The Timer0 interrupt is enabled by setting the T0IE bit (Timer0 Interrupt Enable) of the INTCON register along with the Global Interrupt Enable (GIE) bit.
- ✓ This interrupt would be the indication of the time out and will occur on the every overflow of the TMR0 register.
- ✓ The TOIF bit must be cleared by the interrupt service routine so that the timer interrupt can take place again.

OPTION Register – Data memory bank 1 and 3

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

T0CS – TMR0 Clock Select bit
1 – Transition on TOCK1 pin (PA4/RA4 of PORTA); i.e clock mode
0 – Internal instruction cycle clock (Fosc/4) i.e timer mode

INTCON Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(x)

TMR0: Timer Mode

- ✓ If the clock frequency of PIC16f886 is 4 MHz clock, then the instruction clock will be 1 MHz (1 instruction cycle = 4 clock cycles, for PIC).
- ✓ The counter would then be clocked every 1 μs exactly.
- ✓ Therefore the Timer0 will take 256 μs to count from 00 to FF and then 00.
- ✓ By preloading the TMR0 register with a suitable value, a smaller timer interval (delay) could be selected, with time out indicated by the timer interrupt.
- ✓ For example, if you preload the TMR0 register with the value 200, the Timer0 overflow would occur after 56 μs. (256 μs – 200 μs)
- ✓ An eight bit programmable divider(prescaler) is also available and we can make use of this.
- ✓ The prescaler divides the input frequency by one of eight binary values between 2 and 256. With 1 MHz instruction cycle, the maximum timer period would be 256 x 256 μs = 65.536 ms, corresponding to the prescaler value of 256.
- ✓ The prescaler values are software selectable through PS0, PS1, and PS2 bits of the OPTION register as explained in the previous slides
- ✓ In order to use the prescaler with the Timer0 module, the PSA bit of the OPTION register must be cleared. If the PSA bit is set, no prescaler will be assigned to the Timer0 module.

$$\boldsymbol{f_{out}} = \frac{\boldsymbol{f_{clock}/4}}{\boldsymbol{(Prescaler) * (256 - TMR0)}} \qquad \boldsymbol{T_{out}} = \frac{\boldsymbol{1}}{\boldsymbol{f_{out}}} \qquad f_{clock} = 4\ MHz;$$

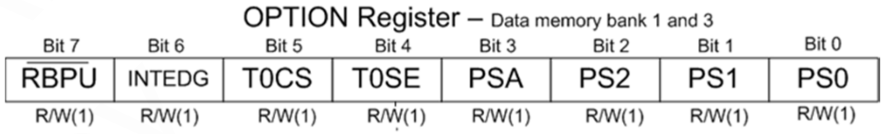
$TMR0\text{:}\ timer\ register\ value;$ $Prescalar\text{:}\ 1\ to\ 256$

Generating 10 ms Delays usingTMR0

- For example, if you preload the TMR0 register with D’100’, the Timer0 overflow would occur after 156 μs. (256 μs –156 μs) . The maximum timer period with a prescaler value of 64 would be 64 x 156 μs = 9.984 ms, + we need a delay of another 16 μs.

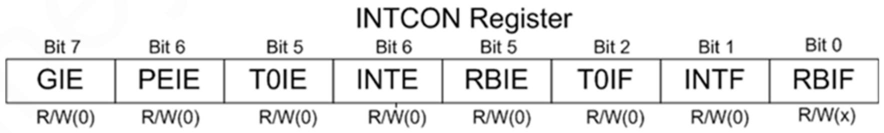
; The Timer0 interrupt is should not be enabled (i.e T0IE bit = 0 of the INTCON register) and the Global Interrupt Enable (GIE) bit should disabled (GIE=0).

```
          ; initialisation
                    BANKSEL  OPTION_REG          ;select Option register to use TMR0
                    Movlw    B’00000101’         ; setup TMR0 for internal input (TOCS =0)
;                                                ;and Prescaler =64; PS2=1, PS1=0, PS0=1;
                    movwf    OPTION_REG
                    -------

;Main
                    call     delay10ms
                    ------

;delay sub_routine
delay10ms           movlw    D’100’
                    movwf    TMR0                ;preload counter with D’100”
loop_ms             btfss    INTCON,2            ;test for if timer overflow has happened i.e T0IF = 1
                    goto     loop_ms             ; loop if not set i.e. wait
                    bcf      INTCON,2            ;clear timer overflow flag i.e. T0IF = 0;
                    return
```



OPTION Register – Data memory bank 1 and 3

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

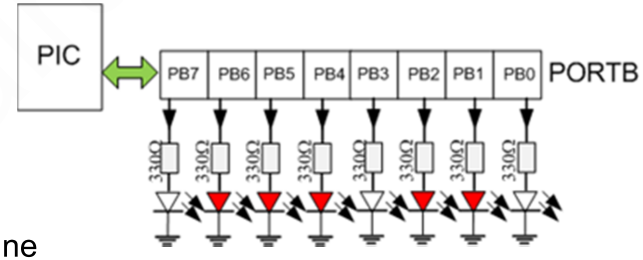


INTCON Register

Bit 7	Bit 6	Bit 5	Bit 6	Bit 5	Bit 2	Bit 1	Bit 0
GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(x)

Using TMR0 interrupts

- Write an interrupt service routine to increment the number in PORTB by 1, when TMR0 register overflows and causes an interrupt. The maximum timer period would be 256 x 256 μs = 65.536 ms, corresponding to the prescaler value of 256.



```
w_temp              equ         H'20'
status_temp  equu   H'21
                    org         H'00'
                    goto        start
                    org         H'04'
                    goto        TMR0_ISR    ;go to Interrupt service routine
;Intialisation
start        BANKSEL    ANSEL       ;select memory bank containing ANSEL Register
             clrf       ANSELH      ;set PORTA B to digital by clearing ANSELH Register
             BANKSEL    TRISB       ;select memory bank containing TRISB Register
             clrf       TRISB       ;make PORTB all outputs
;
             BANKSEL    OPTION_REG     ;select Option register to use TMR0
             bcf        OPTION_REG,5  ;TOCS=0; timer mode is selected
             bcf        OPTION_REG,3   ;Prescaler selected
             bsf        OPTION_REG,0   ;PS0=1 ; prescaler = 1:256
             bsf        OPTION_REG,1   ;PS1=1
             bsf        OPTION_REG,2   ;PS2=1
;
;            BANKSEL    INTCON         ;select INTCON register to enable interrupt
             bsf        INTCON,5       ; T0IE=1 implies TMR0 overflow interrupt is  enabled
             bsf        INTCON,7        ; Global interrupt enabled
             BANKSEL    PORTB       ;select memory bank containing PORTB Register
             clrf       PORTB       ;reset PORTB (turn off all LEDS)

;main program
wait_loop    goto       wait_loop
             END
```

INTCON Register

Bit 7	Bit 6	Bit 5	Bit 6	Bit 5	Bit 2	Bit 1	Bit 0
GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(x)

R/W Read/Writable bit; (0) After rest, bit is cleared; (x) After reset, bit is unknown

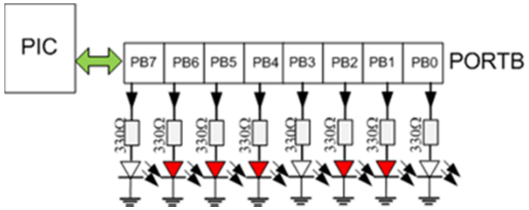
OPTION Register – Data memory bank 1 and 3

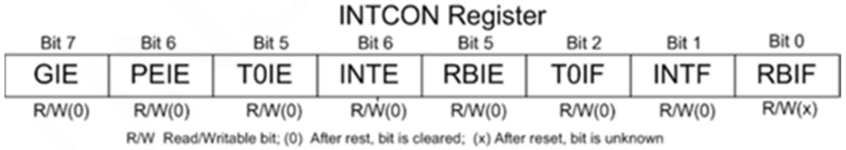
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

R/W Read/Writable bit; (1) After rest, bit is set;

Interrupt service routine

```
;Interrupt service routine  (first save all registers)
TMR0_ISR    movwf        w_temp       ;save current contents of w register in a temporary location ‘w_temp’
            movf         STATUS,W     ;move the contents of the  status register into W register
            movwf        status_temp  ;save the contents of  STATUS register in a temporary location ‘status_temp’
;
;
;
;main function
            BANKSEL      PORTB        ;select the databank memory containing PORTB
            incf         PORTB        ; increment the content of PORTB by 1
            BANKSEL      INTCON       ;select the databank memory containing INTCON
            bcf          INTCON,2     ; clear interrupt flag bit T0IF
;
;
;Restore all registers
            movf         status_temp,W ;retrieve the STATUS register content saved in temporary location
            movwf        STATUS       ; put it back in the STATUS register
;
;it is important that  any further  instructions , before returning from the service routine should not affect the STATUS register.
So we use swapf and movwf instructions as they do not affect the STSTUS register (see instruction set sheet)
;
            swapf        w_temp,1     ;the upper and lower nibbles are exchanged in w-temp and result placed in w-temp
            swapf        w_temp,0     ;the upper and lower nibbles are exchanged in w-temp and result placed in W reg.
;
            bsf          INTCON,7                     ;Global interrupt enabled
            retfie                    ;return from the interrupt service routine
;
            END
```

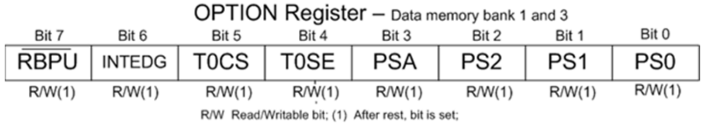




INTCON Register

Bit 7	Bit 6	Bit 5	Bit 6	Bit 5	Bit 2	Bit 1	Bit 0
GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(x)

R/W Read/Writable bit; (0) After rest, bit is cleared; (x) After reset, bit is unknown



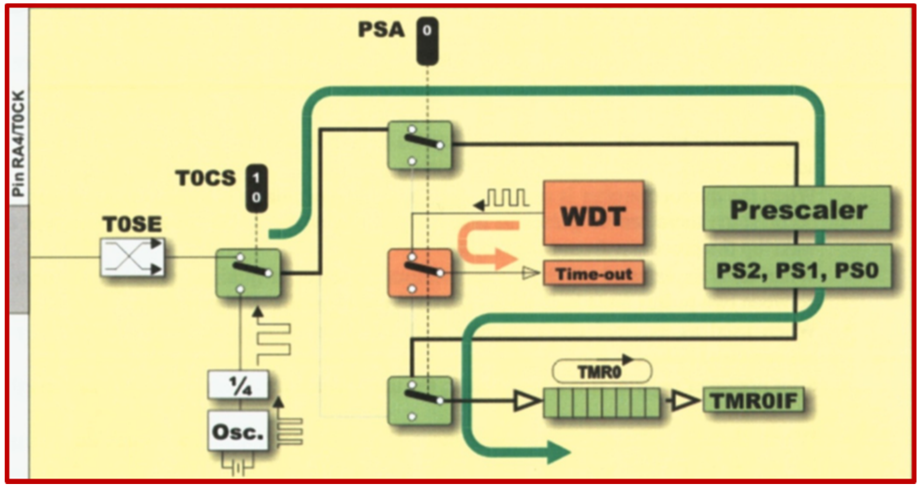
OPTION Register – Data memory bank 1 and 3

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

R/W Read/Writable bit; (1) After rest, bit is set;

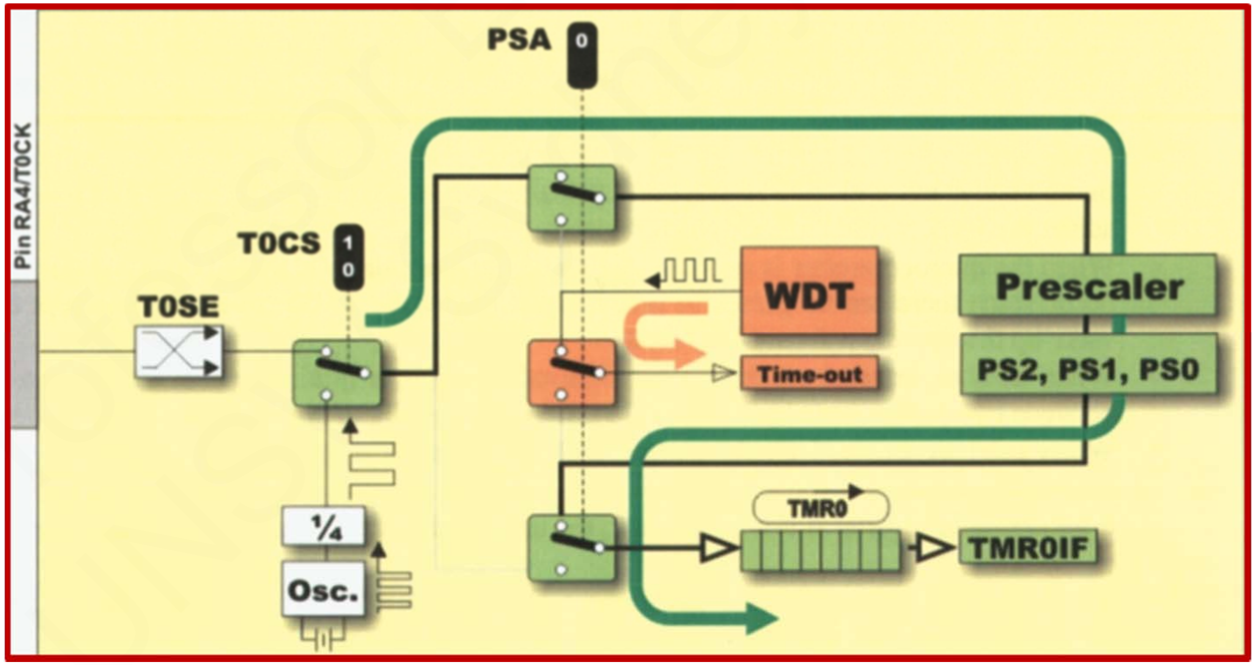
Watchdog Timer (WDT)

- ✓ One of the major functions of a watchdog timer is to automatically reset the microcontroller in the event of a crash.
- ✓ The watch-dog timer built-in to the PIC16f886 runs with its own RC oscillator (independent of external clock) and has a typical minimum time-out period of 18 milliseconds. There is a programmable prescaler (which divides the RC clock) that can multiply this period by 128 (max) to give a total time-out period of 2.3 secs, which is good for most applications.
- ✓ When the Watchdog Timer (WDT) is enabled, a counter starts at 00 and increments by 1 until it reaches FF. When it goes from FF to 00, the PIC micro will be reset.
- ✓ The only way we can stop the WDT from resetting the PIC is to periodically reset the WDT back to 00 within the program.
- ✓ The instruction for clearing the WDT is ‘clrwdt’
- ✓ If the program does get stuck for some reason, the WDT will then reset the PIC, causing our program to restart from the beginning.



Watchdog Timer (WDT)

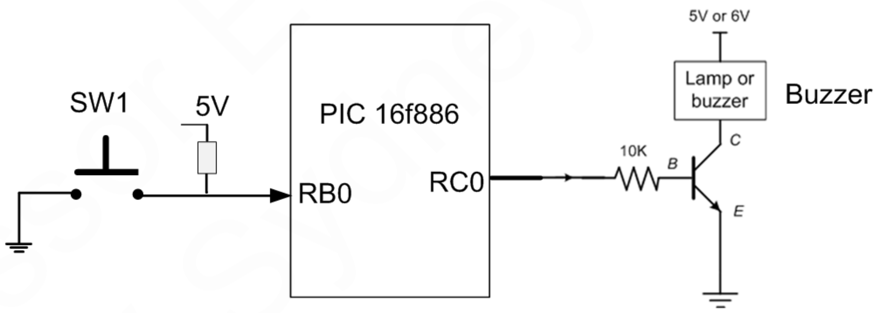
- ✓ Hopefully when the PIC restarts whatever condition led to the crash will have gone away and the PIC will resume its normal operation.
- ✓ Note that the prescaler is not readable or writeable. When assigned to the TMR0 module, all instructions writing to the TMR0 register will clear the prescaler
- ✓ When the prescaler is assigned to WDT, a 'clrwdt' instruction will clear the prescaler along with the WDT.
- ✓ When changing the prescaler from Timer0 to the WDT module or vice versa care must be taken and a sequence of instructions must be executed (Ref: microchip data sheet)

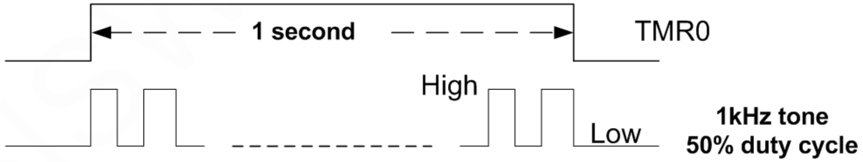


Laboratory Activities

Activity 8: Intrusion Warning System

- Write an assembly language program for an intrusion warning system, that uses interrupts on the PIC16F886 microcontroller to sound an 1000 Hz tone for 1 second, whenever a door sensor (SW1) connected to RB0 is closed (i.e., there is an intrusion through the door) (see figure below).
- The 1 second time-out must be implemented using the TMR0 register overflow interrupt. Also calculate the total amount of program memory space required for the interrupt routine , in terms of bytes, occupied .
- Note that you are required to include a switch de-bouncing routine.





ELEC2117: References

1. Designing Embedded Systems with PIC Microcontrollers – Tim Wilmshurst, Elsevier, 2010
2. PIC Microcontrollers –Free online book – mikroElektronika ; http://www.mikroe.com/products/view/11/book-pic-microcontrollers/
3. PIC 16F886 Data Sheet (2007), Microchip Technology; www.microchip.com