University of San Carlos | Department of
COMPUTER ENGINEERING

CpE 3201
Embedded Systems

Timers

Basic Concepts in Counters & Timers

Counter

- simply a **register**
- can be loaded with a binary count (can incremented or decremented at every clock pulse)
- **Time** is calculated by subtracting the beginning count from the end count and multiplying the difference by the clock period

Basic Concepts in Counters & Timers

Example: if a counter is loaded with 100_{10} and counts down to 00 and the clock frequency is 2 MHz, the **time** is calculated as follows:

- **Clock Period** = (1/f) = $1/(2\text{x}10^6)$ = 0.5 us
- **Time** = (Difference in Counts) x Clock Period
  = (100 - 0) x 0.5 us = **50 us**

Basic Concepts in Counters & Timers

Event Counter

- the counter is triggered by an **event** [instead of a clock] such as a button push that increments or decrements the counter

Types of Counters and Timers

Up-counter

- this counter counts up at every clock cycle and when it **reaches the maximum count**, it resets back to 0 (overflow) which may set a flag or generate an interrupt signal; readable/writable

Down-counter

- counts down at every clock cycle and when it **reaches 0**, it may set a flag or generate an interrupt signal; readable/writable

Types of Counters and Timers

Free-Running Counter

- this counter runs continuously and it is **only readable**; may set flags or generate an interrupt signal when it reach its maximum count

Timer Applications

Time Delay

- timer register can be loaded with a certain value to produce a delay; in the case of free-running counter the program needs to record the timer reading at a starting point, add the delay count to that reading and keep comparing the readings until a match is found

Timer Applications

Pulse Wave Generation

- to **produce** a periodic wave (square wave @ 50% duty cycle for example), the time delay for ON-time and OFF-time is the same

Pulse Width Measurement

- to **measure** a pulse-width, the program begins the counter on the rising edge and stops at the falling edge

Timer Applications

Timer in the Counter Mode

- the timer register can be set up as a **counter** to count pulses representing events
- the timer is not incremented by an internal clock but by an **external pulse** which may include:
  - external clocks
  - sensors (mechanical, optical, etc.)
  - signals from other hardware modules

Timers in PIC16F877A

Timer0 Module

- an 8-bit readable and writable timer/counter
- 8-bit software programmable prescaler
- Interrupt on overflow from FFh to 00h
- Internal or external clock select
- Edge select for external clock

Timer1 Module

- a 16-bit timer/counter consisting of two 8-bit registers (**TMR1H** and **TMR1L**) which are readable and writable.
- TMR1 interrupt is generated on overflow, if enabled
- Has two modes: **Timer** or **Counter**

Timers in PIC16F877A

- **Timer2 Module**
  - is an 8-bit timer with a prescaler and a postscaler. It can be used as the **PWM time base** for the **PWM mode of the CCP module(s)**
  - **TMR2** register is readable and writable and is cleared on any device Reset
  - has an 8-bit period register, **PR2**
  - Timer2 increments from 00h until it matches PR2 and then resets to 00h on the next increment cycle
  - the match output of TMR2 goes through a 4-bit postscaler (which gives a 1:1 to 1:16 scaling inclusive) to generate a TMR2 interrupt

Timers in PIC16F877A

- **Capture, Compare & PWM Module**
  - Each Capture/Compare/PWM (CCP) module contains a 16-bit register which can operate as a:
    - 16-bit Capture register
    - 16-bit Compare register
    - PWM Master/Slave Duty Cycle register

In this unit…

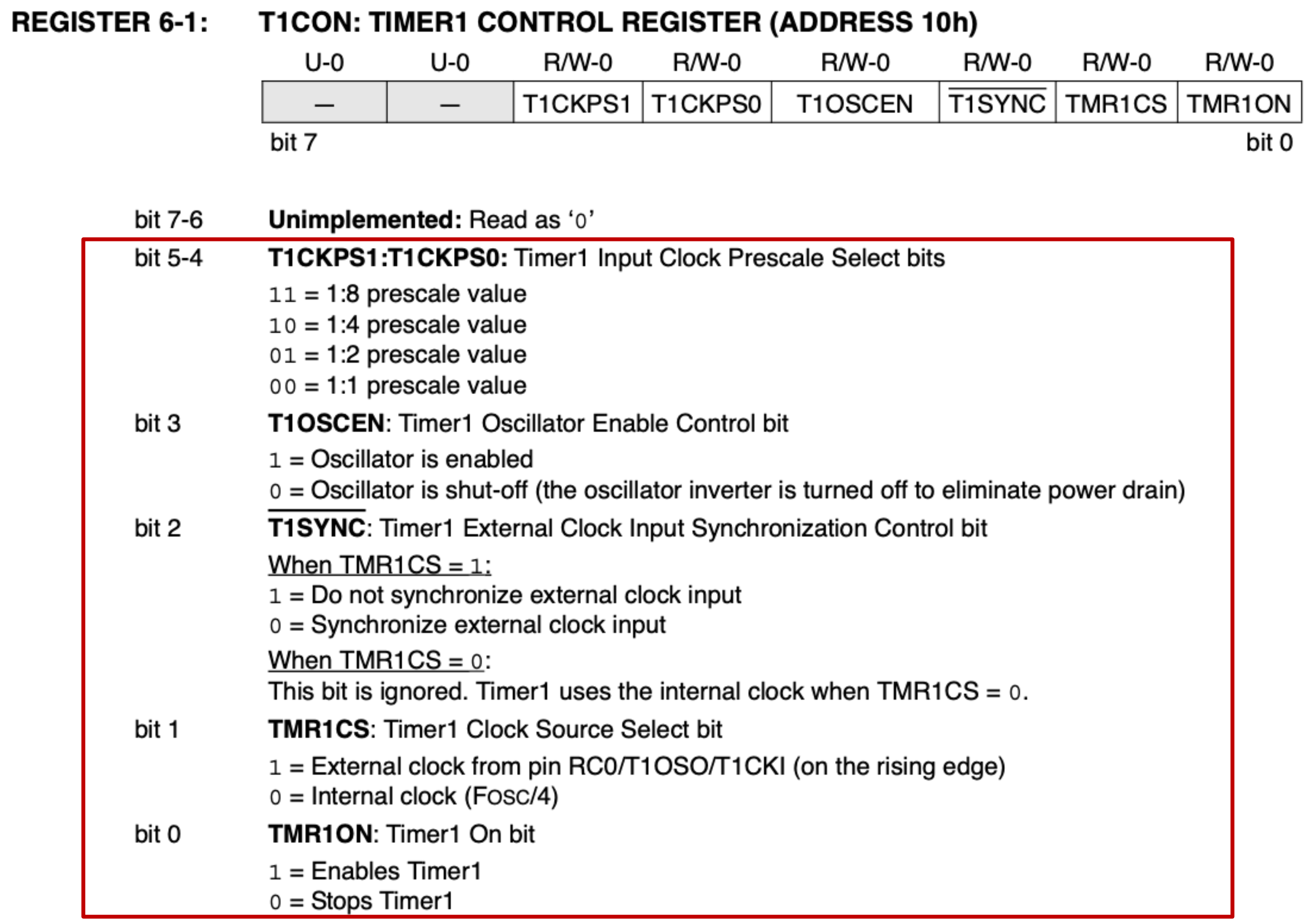
This unit deals with the following timers:

- Timer1 (Timer Mode)
- Timer2
- CCP Module

Timer1

- In **Timer mode**, Timer1 increments every instruction cycle. In **Counter mode**, it increments on every rising edge of the external clock input.
- Timer1 can be enabled/disabled by setting or clearing **control bit**, **TMR1ON** (T1CON<0>).
- Timer1 can be enabled/disabled by setting/clearing **TMR1 interrupt enable bit**, **TMR1IE** (PIE1<0>).
- Timer1 register is **TMR1** (TMR1H:TMR1L).

Timer1 Control Register (T1CON)



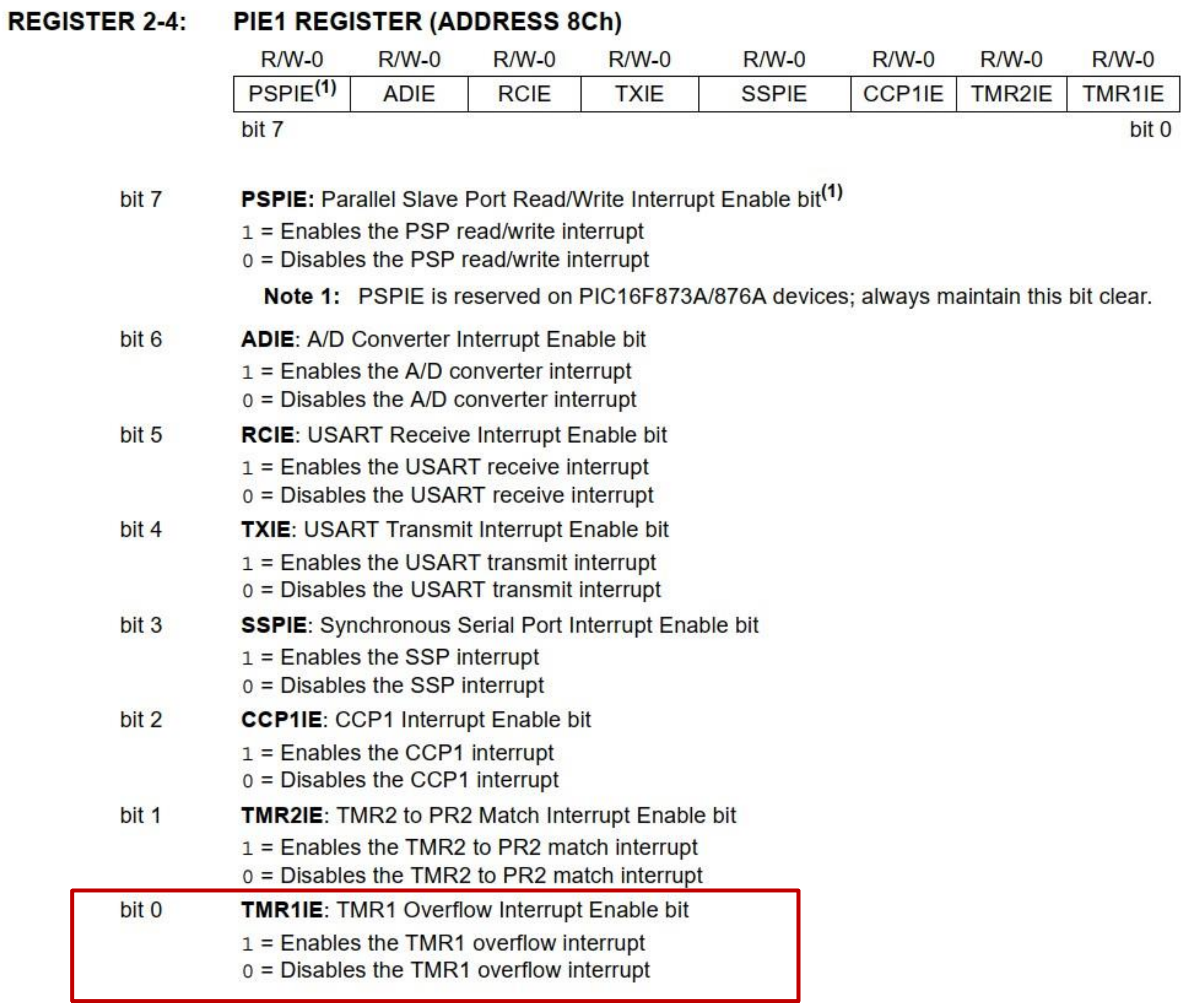
REGISTER 6-1: T1CON: TIMER1 CONTROL REGISTER (ADDRESS 10h)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
bit 7							bit 0

bit 7-6	**Unimplemented:** Read as '0'
bit 5-4	**T1CKPS1:T1CKPS0:** Timer1 Input Clock Prescale Select bits 11 = 1:8 prescale value 10 = 1:4 prescale value 01 = 1:2 prescale value 00 = 1:1 prescale value
bit 3	**T1OSCEN**: Timer1 Oscillator Enable Control bit 1 = Oscillator is enabled 0 = Oscillator is shut-off (the oscillator inverter is turned off to eliminate power drain)
bit 2	**$\overline{\text{T1SYNC}}$**: Timer1 External Clock Input Synchronization Control bit When TMR1CS = 1: 1 = Do not synchronize external clock input 0 = Synchronize external clock input When TMR1CS = 0: This bit is ignored. Timer1 uses the internal clock when TMR1CS = 0.
bit 1	**TMR1CS**: Timer1 Clock Source Select bit 1 = External clock from pin RC0/T1OSO/T1CKI (on the rising edge) 0 = Internal clock (FOSC/4)
bit 0	**TMR1ON**: Timer1 On bit 1 = Enables Timer1 0 = Stops Timer1

Interrupt enable is at PIE1 register while the **flags at PIR1 register**.
See PIC16F87X data sheet for more details.

Peripheral Interrupt Enable1 Register (PIE1)



REGISTER 2-4: PIE1 REGISTER (ADDRESS 8Ch)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIE[(1)]	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

bit 7 **PSPIE:** Parallel Slave Port Read/Write Interrupt Enable bit[(1)]
1 = Enables the PSP read/write interrupt
0 = Disables the PSP read/write interrupt

Note 1: PSPIE is reserved on PIC16F873A/876A devices; always maintain this bit clear.

bit 6 **ADIE**: A/D Converter Interrupt Enable bit
1 = Enables the A/D converter interrupt
0 = Disables the A/D converter interrupt

bit 5 **RCIE**: USART Receive Interrupt Enable bit
1 = Enables the USART receive interrupt
0 = Disables the USART receive interrupt

bit 4 **TXIE**: USART Transmit Interrupt Enable bit
1 = Enables the USART transmit interrupt
0 = Disables the USART transmit interrupt

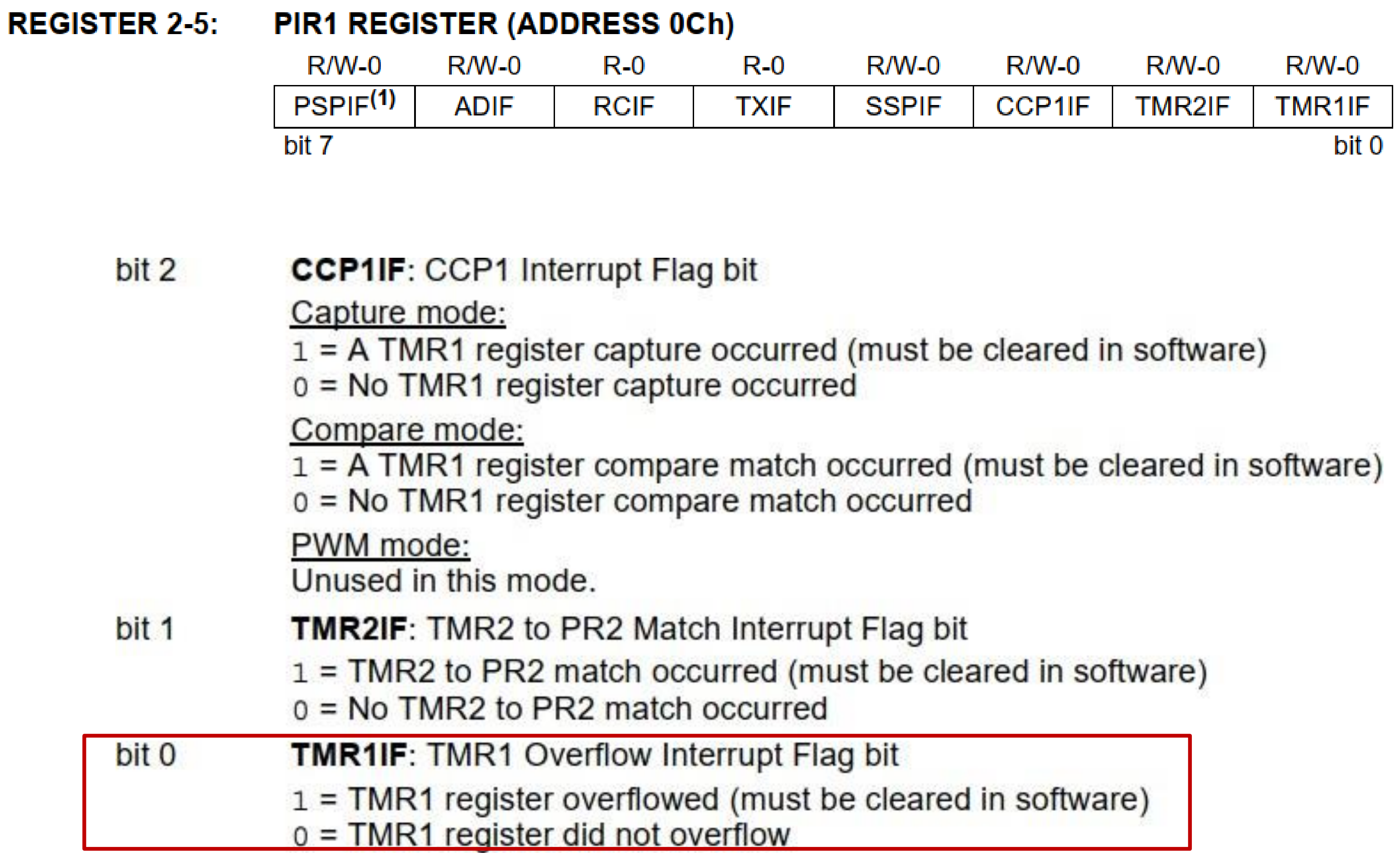
bit 3 **SSPIE**: Synchronous Serial Port Interrupt Enable bit
1 = Enables the SSP interrupt
0 = Disables the SSP interrupt

bit 2 **CCP1IE**: CCP1 Interrupt Enable bit
1 = Enables the CCP1 interrupt
0 = Disables the CCP1 interrupt

bit 1 **TMR2IE**: TMR2 to PR2 Match Interrupt Enable bit
1 = Enables the TMR2 to PR2 match interrupt
0 = Disables the TMR2 to PR2 match interrupt

bit 0 **TMR1IE**: TMR1 Overflow Interrupt Enable bit
1 = Enables the TMR1 overflow interrupt
0 = Disables the TMR1 overflow interrupt

Peripheral Interrupt Register1 (PIR1)



REGISTER 2-5: PIR1 REGISTER (ADDRESS 0Ch)

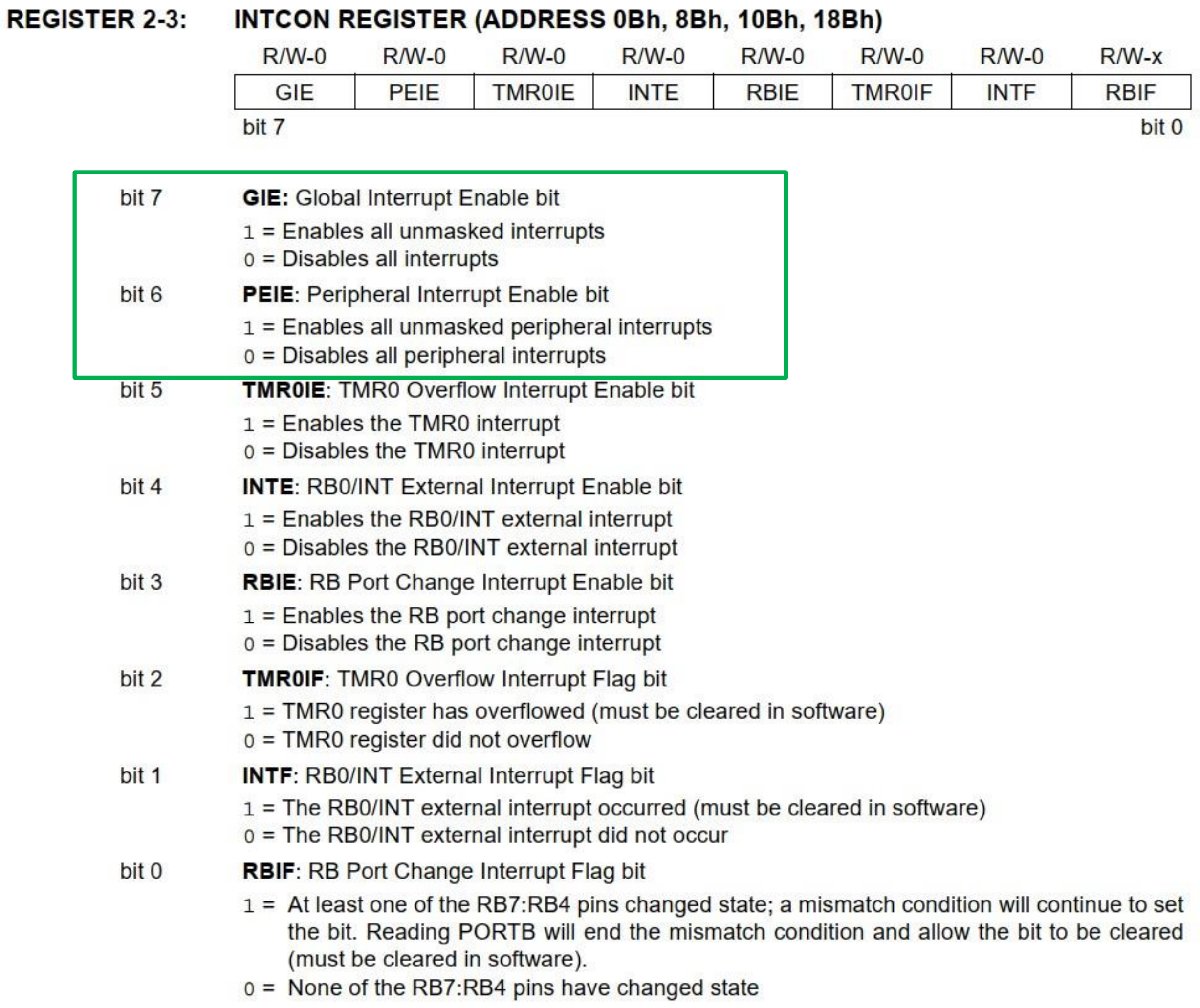
R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

bit 2 **CCP1IF**: CCP1 Interrupt Flag bit
<u>Capture mode:</u>
1 = A TMR1 register capture occurred (must be cleared in software)
0 = No TMR1 register capture occurred
<u>Compare mode:</u>
1 = A TMR1 register compare match occurred (must be cleared in software)
0 = No TMR1 register compare match occurred
<u>PWM mode:</u>
Unused in this mode.

bit 1 **TMR2IF**: TMR2 to PR2 Match Interrupt Flag bit
1 = TMR2 to PR2 match occurred (must be cleared in software)
0 = No TMR2 to PR2 match occurred

bit 0 **TMR1IF**: TMR1 Overflow Interrupt Flag bit
1 = TMR1 register overflowed (must be cleared in software)
0 = TMR1 register did not overflow

Interrupt Control Register (INTCON)



REGISTER 2-3: INTCON REGISTER (ADDRESS 0Bh, 8Bh, 10Bh, 18Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF
bit 7							bit 0

bit 7	**GIE:** Global Interrupt Enable bit 1 = Enables all unmasked interrupts 0 = Disables all interrupts
bit 6	**PEIE**: Peripheral Interrupt Enable bit 1 = Enables all unmasked peripheral interrupts 0 = Disables all peripheral interrupts
bit 5	**TMR0IE**: TMR0 Overflow Interrupt Enable bit 1 = Enables the TMR0 interrupt 0 = Disables the TMR0 interrupt
bit 4	**INTE**: RB0/INT External Interrupt Enable bit 1 = Enables the RB0/INT external interrupt 0 = Disables the RB0/INT external interrupt
bit 3	**RBIE**: RB Port Change Interrupt Enable bit 1 = Enables the RB port change interrupt 0 = Disables the RB port change interrupt
bit 2	**TMR0IF**: TMR0 Overflow Interrupt Flag bit 1 = TMR0 register has overflowed (must be cleared in software) 0 = TMR0 register did not overflow
bit 1	**INTF**: RB0/INT External Interrupt Flag bit 1 = The RB0/INT external interrupt occurred (must be cleared in software) 0 = The RB0/INT external interrupt did not occur
bit 0	**RBIF**: RB Port Change Interrupt Flag bit 1 = At least one of the RB7:RB4 pins changed state; a mismatch condition will continue to set the bit. Reading PORTB will end the mismatch condition and allow the bit to be cleared (must be cleared in software). 0 = None of the RB7:RB4 pins have changed state

Summary of Registers for TMR1

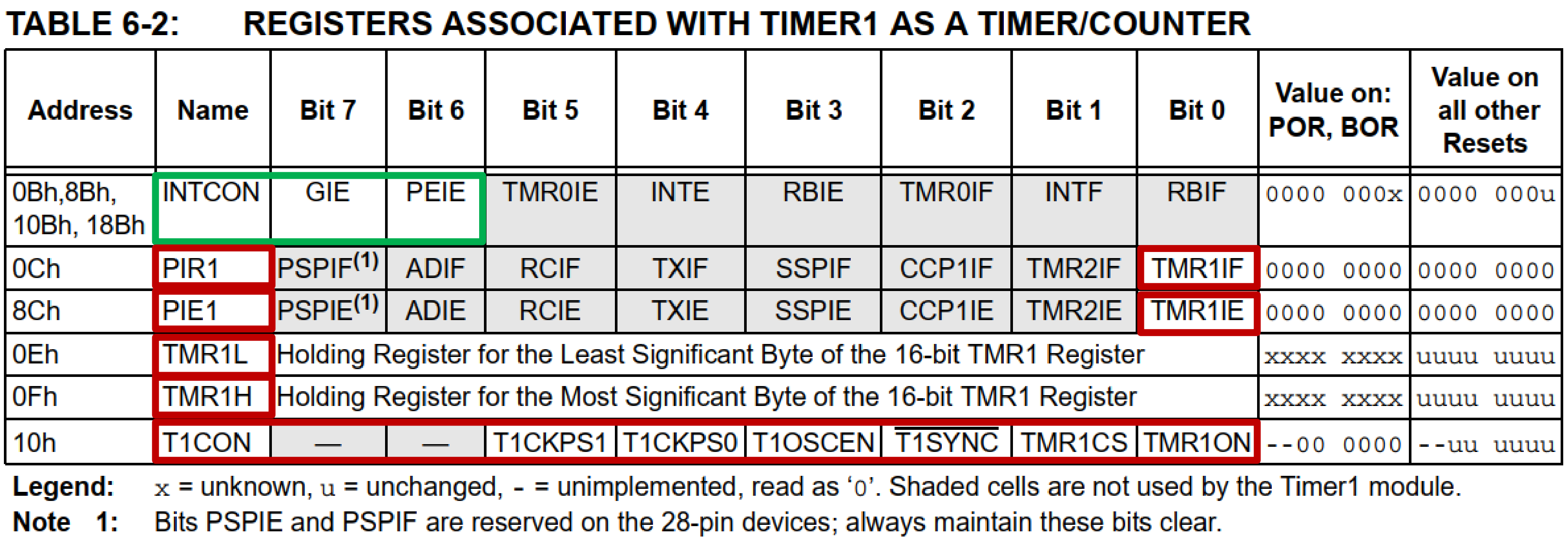


TABLE 6-2: REGISTERS ASSOCIATED WITH TIMER1 AS A TIMER/COUNTER

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
0Bh,8Bh, 10Bh, 18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF[1]	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
8Ch	PIE1	PSPIE[1]	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
0Eh	TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
0Fh	TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	--00 0000	--uu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by the Timer1 module.

Note 1: Bits PSPIE and PSPIF are reserved on the 28-pin devices; always maintain these bits clear.

Timer1 (Timer Mode)

- **Timer mode** is selected by clearing the **TMR1CS** (T1CON<1>) bit.
- In this mode, the input clock to the timer is FOSC/4.
- The timeout can be calculated given the oscillator frequency (Fosc) of the MCU and a prescaler value.

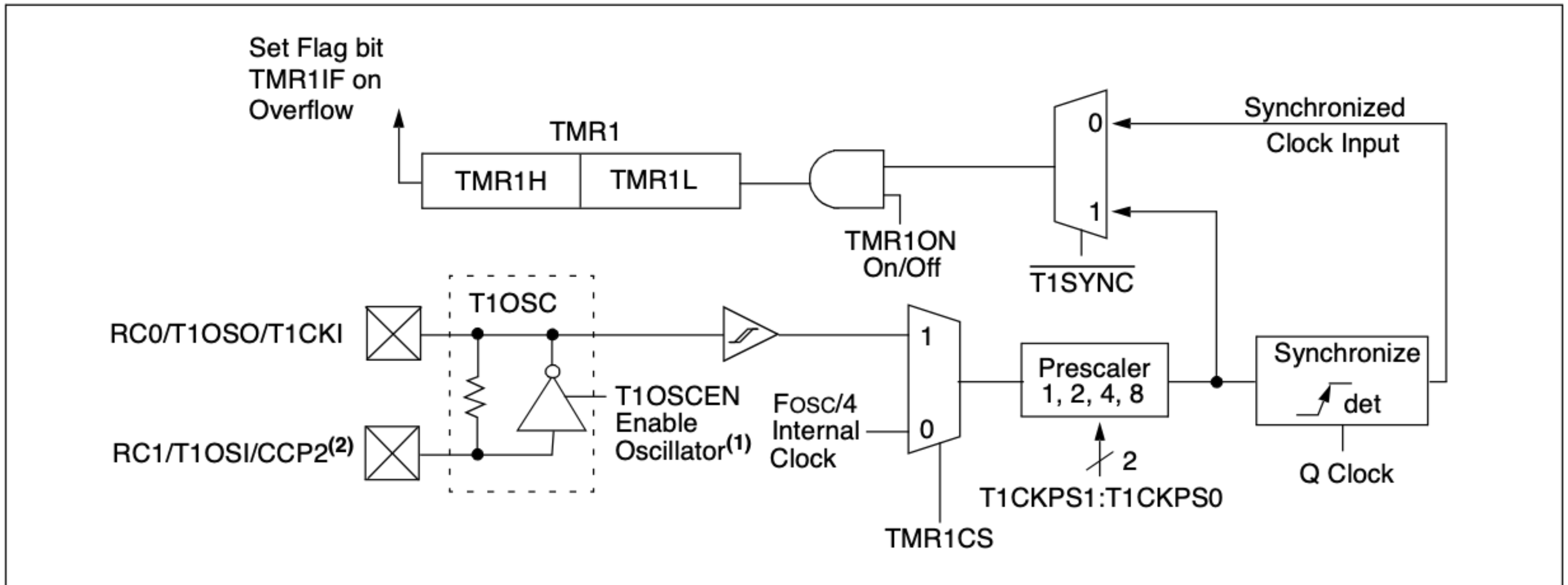
$$timeout = \frac{1}{\frac{Fosc}{4}} \text{ x } prescaler \text{ x } TimerMaxCount \quad (1)$$

For example, given Fosc = 4MHz and a prescaler of 1:4

$$timeout = \frac{1}{\frac{4\text{MHz}}{4}} \text{ x } 4 \text{ x } 65536 = \frac{1}{1\text{x}10^6} \text{ x } 262144 = 0.2621\text{s}$$

Timer1 Module

FIGURE 6-2: TIMER1 BLOCK DIAGRAM



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

Timer1 Module Block Diagram

Example (Timer1)

Configure Timer1 to generate an interrupt every 0.5 seconds. Create an ISR for Timer1 overflow interrupt where it blinks the LED connected at RA0. Assume Fosc = 4MHz and the timer operates in Timer Mode.

Using a prescaler of 1:8 and since the timeout is given at 0.5s

From Equation 1:

$$0.5s = \frac{1}{\frac{4\text{MHz}}{4}} \text{ x } 8 \text{ x } TimerMaxCount$$

Timer Max Count must be determined in order to have a timeout of 0.5s

$$TimerMaxCount = 62500$$

The value of Timer Max Count will be used to reload the timer after interrupt.

```
void main(void)
{
  ADCON1 = 0x6;           // set all pins in PORTA as digital I/O
  TRISA = 0x00;           // sets all of PORTA to output
  RA0 = 0;                // initialize RA0 to 0 (LED off)
  T1CON = 0x30;           // 1:8 prescaler, internal clock, Timer1 off
  TMR1IE = 1;             // enable Timer1 overflow interrupt (PIE1 reg)
  TMR1IF = 0;             // reset interrupt flag (PIR1 reg)
  PEIE = 1;               // enable all peripheral interrupt (INTCON reg)
  GIE = 1;                // enable all unmasked interrupts (INTCON reg)
  TMR1 = 0x0BDC;          // counter starts counting at 0x0BDC (3036)
  TMR1ON = 1;             // Turns on Timer1 (T1CON reg)

  for(;;)                 // foreground routine
  {


  }
}
```

The value of 0x0BDC (3036) was derived from 65536-62500. By starting the counter at this value, the 0.5s timeout can be achieved.

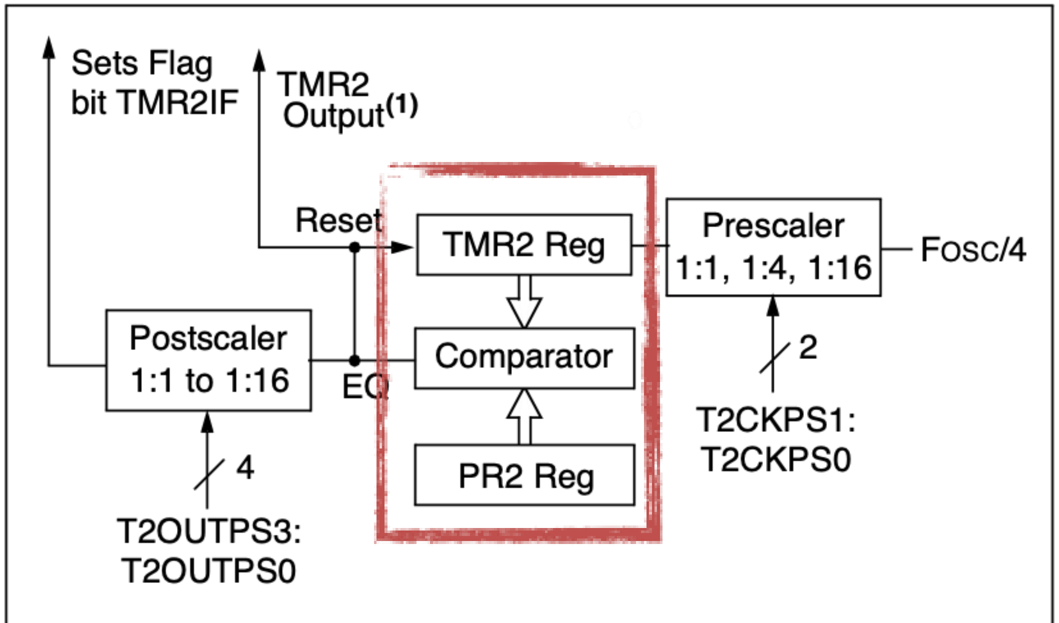
```
void interrupt ISR(void)
{
  GIE = 0;            // disable all unmasked interrupts (INTCON reg)
  if(TMR1IF==1)       // checks Timer1 interrupt flag
  {
    TMR1IF = 0;       // clears interrupt flag
    TMR1 = 0x0BDC;    // timer will start counting at 0x0BDC (3036)
    RA0 = RA0^1;      // toggles the LED at RA0
  }
  GIE = 1;            // enable all unmasked interrupts (INTCON reg)
}
```

Timer2

- Unlike Timer0 and Timer1, this timer module does not generate interrupt upon overflow.
- It generates an interrupt when its value will be **equal to the period register (PR2)** and resets to 00h.
- **PR2** can be read or written to and is also an 8-bit register.
- Timer2 register is **TMR2**.

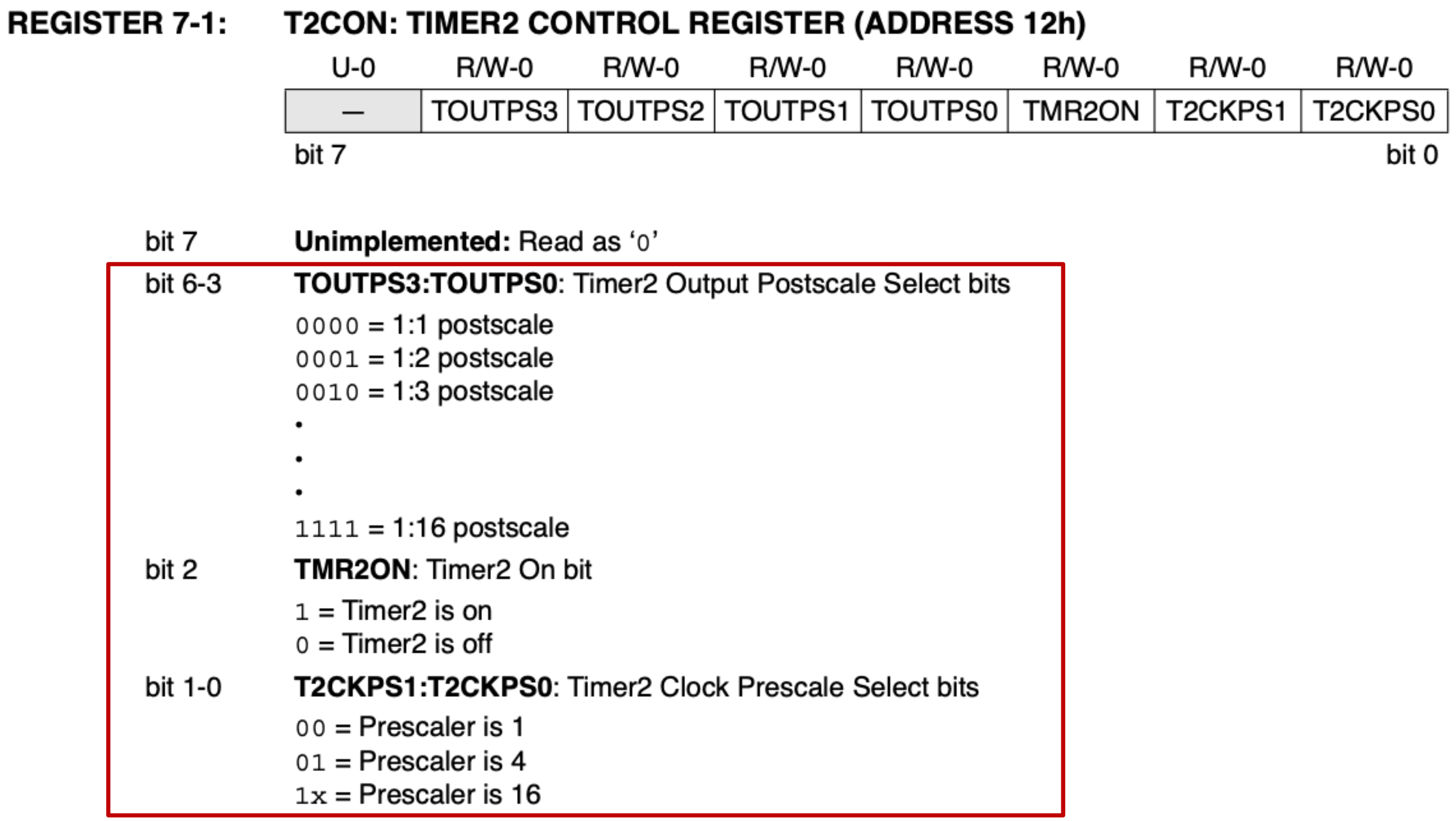
Timer2

FIGURE 7-1: TIMER2 BLOCK DIAGRAM



Note 1: TMR2 register output can be software selected by the SSP module as a baud clock.

Timer2 Control Register (T2CON)



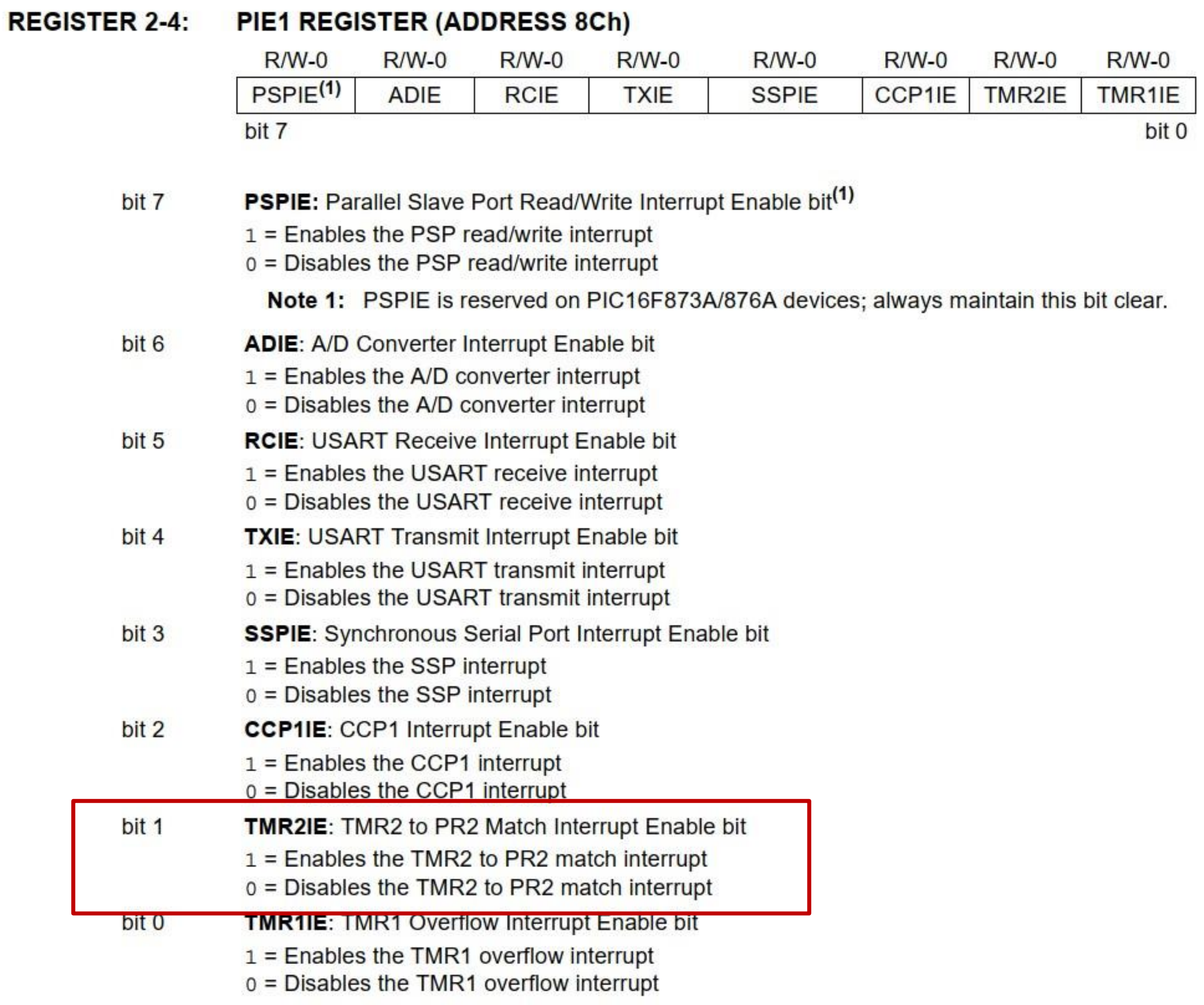
REGISTER 7-1: T2CON: TIMER2 CONTROL REGISTER (ADDRESS 12h)

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit 7							bit 0

bit 7	**Unimplemented:** Read as ‘0’
bit 6-3	**TOUTPS3:TOUTPS0**: Timer2 Output Postscale Select bits 0000 = 1:1 postscale 0001 = 1:2 postscale 0010 = 1:3 postscale • • • 1111 = 1:16 postscale
bit 2	**TMR2ON**: Timer2 On bit 1 = Timer2 is on 0 = Timer2 is off
bit 1-0	**T2CKPS1:T2CKPS0**: Timer2 Clock Prescale Select bits 00 = Prescaler is 1 01 = Prescaler is 4 1x = Prescaler is 16

Interrupt enable is at PIE1 register while the **flags at PIR1 register**. See PIC16F87X data sheet for more details.

Peripheral Interrupt Enable1 Register (PIE1)

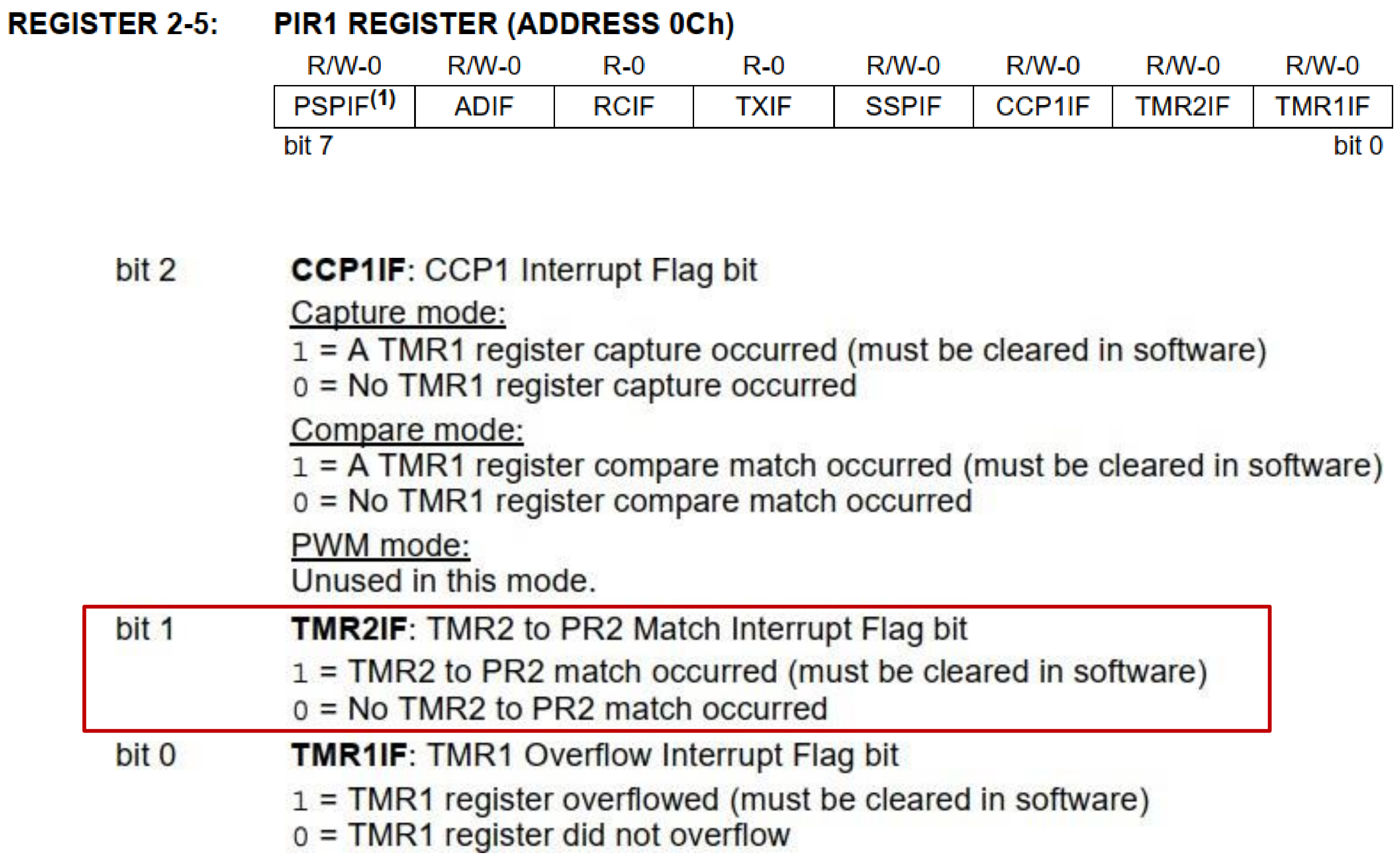


REGISTER 2-4: PIE1 REGISTER (ADDRESS 8Ch)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIE[(1)]	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

bit 7	**PSPIE:** Parallel Slave Port Read/Write Interrupt Enable bit[(1)] 1 = Enables the PSP read/write interrupt 0 = Disables the PSP read/write interrupt **Note 1:** PSPIE is reserved on PIC16F873A/876A devices; always maintain this bit clear.
bit 6	**ADIE**: A/D Converter Interrupt Enable bit 1 = Enables the A/D converter interrupt 0 = Disables the A/D converter interrupt
bit 5	**RCIE**: USART Receive Interrupt Enable bit 1 = Enables the USART receive interrupt 0 = Disables the USART receive interrupt
bit 4	**TXIE**: USART Transmit Interrupt Enable bit 1 = Enables the USART transmit interrupt 0 = Disables the USART transmit interrupt
bit 3	**SSPIE**: Synchronous Serial Port Interrupt Enable bit 1 = Enables the SSP interrupt 0 = Disables the SSP interrupt
bit 2	**CCP1IE**: CCP1 Interrupt Enable bit 1 = Enables the CCP1 interrupt 0 = Disables the CCP1 interrupt
bit 1	**TMR2IE**: TMR2 to PR2 Match Interrupt Enable bit 1 = Enables the TMR2 to PR2 match interrupt 0 = Disables the TMR2 to PR2 match interrupt
bit 0	**TMR1IE**: TMR1 Overflow Interrupt Enable bit 1 = Enables the TMR1 overflow interrupt 0 = Disables the TMR1 overflow interrupt

Peripheral Interrupt Register1 (PIR1)



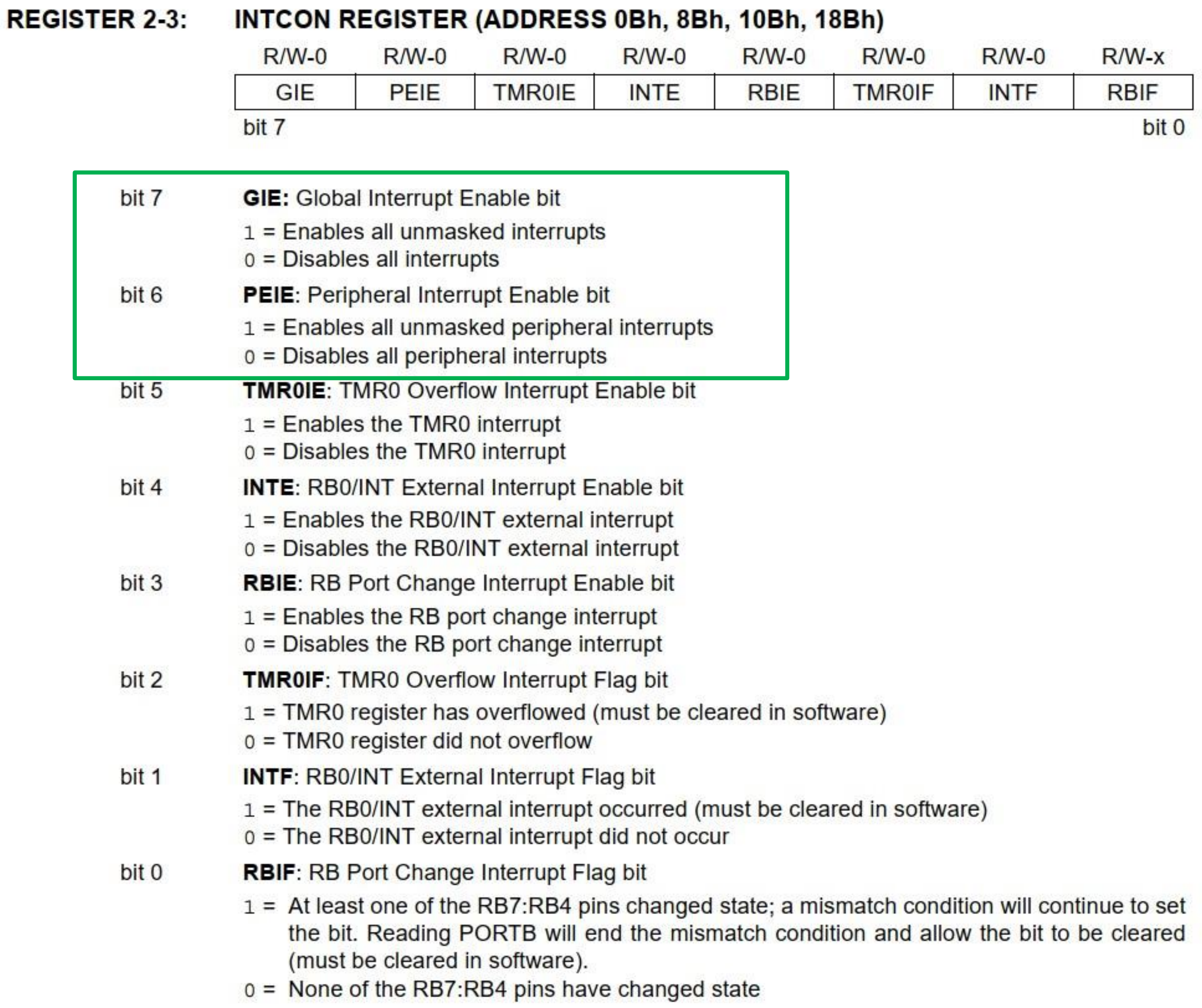
REGISTER 2-5: PIR1 REGISTER (ADDRESS 0Ch)

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIF[1]	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

bit 2	**CCP1IF**: CCP1 Interrupt Flag bit Capture mode: 1 = A TMR1 register capture occurred (must be cleared in software) 0 = No TMR1 register capture occurred Compare mode: 1 = A TMR1 register compare match occurred (must be cleared in software) 0 = No TMR1 register compare match occurred PWM mode: Unused in this mode.
bit 1	**TMR2IF**: TMR2 to PR2 Match Interrupt Flag bit 1 = TMR2 to PR2 match occurred (must be cleared in software) 0 = No TMR2 to PR2 match occurred
bit 0	**TMR1IF**: TMR1 Overflow Interrupt Flag bit 1 = TMR1 register overflowed (must be cleared in software) 0 = TMR1 register did not overflow

UNIVERSITY of SAN CARLOS
SCIENTIA • VIRTUS • DEVOTIO

Interrupt Control Register (INTCON)



REGISTER 2-3: INTCON REGISTER (ADDRESS 0Bh, 8Bh, 10Bh, 18Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF
bit 7							bit 0

bit 7 **GIE:** Global Interrupt Enable bit
1 = Enables all unmasked interrupts
0 = Disables all interrupts

bit 6 **PEIE**: Peripheral Interrupt Enable bit
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts

bit 5 **TMR0IE**: TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4 **INTE**: RB0/INT External Interrupt Enable bit
1 = Enables the RB0/INT external interrupt
0 = Disables the RB0/INT external interrupt

bit 3 **RBIE**: RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **TMR0IF**: TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INTF**: RB0/INT External Interrupt Flag bit
1 = The RB0/INT external interrupt occurred (must be cleared in software)
0 = The RB0/INT external interrupt did not occur

bit 0 **RBIF**: RB Port Change Interrupt Flag bit
1 = At least one of the RB7:RB4 pins changed state; a mismatch condition will continue to set the bit. Reading PORTB will end the mismatch condition and allow the bit to be cleared (must be cleared in software).
0 = None of the RB7:RB4 pins have changed state

Summary of Registers for TMR2

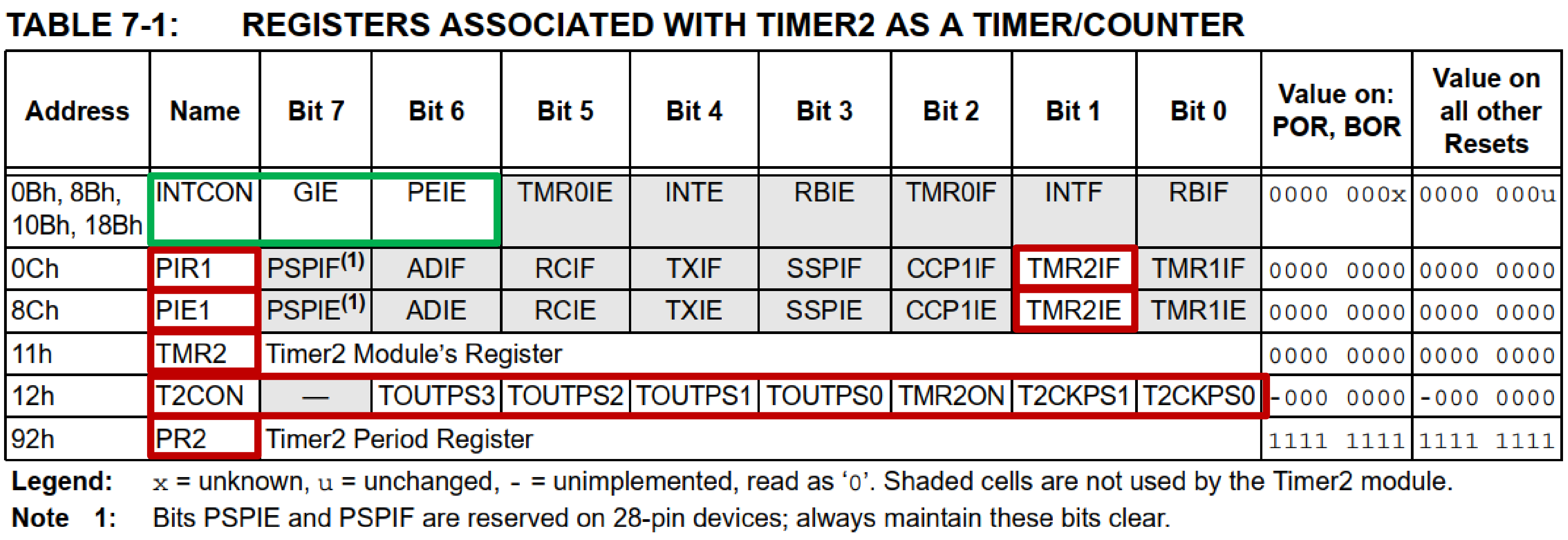


TABLE 7-1: REGISTERS ASSOCIATED WITH TIMER2 AS A TIMER/COUNTER

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
0Bh, 8Bh, 10Bh, 18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	`0000 000x`	`0000 000u`
0Ch	PIR1	PSPIF[(1)]	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	`0000 0000`	`0000 0000`
8Ch	PIE1	PSPIE[(1)]	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	`0000 0000`	`0000 0000`
11h	TMR2	Timer2 Module's Register								`0000 0000`	`0000 0000`
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	`-000 0000`	`-000 0000`
92h	PR2	Timer2 Period Register								`1111 1111`	`1111 1111`

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 28-pin devices; always maintain these bits clear.

Timer2 Timeout

- Timer2 increments every clock cycle (Fosc/4).
- The timeout can be calculated given the oscillator frequency (Fosc) of the MCU and a prescaler value similar to Timer0.

$$timeout = \frac{1}{\frac{Fosc}{4}} \text{ x } prescaler \text{ x } TimerMaxCount \quad (1)$$

For example, given Fosc = 4MHz and a prescaler of 1:4

$$timeout = \frac{1}{\frac{4\text{Mhz}}{4}} \text{ x } 4 \text{ x } 256 = \frac{1}{1\text{x}10^6} \text{ x } 1024 = 1.024\text{ms}$$

Example (Timer2)

- Generate a pulse-wave with a frequency of 1000Hz at 50% duty cycle. The output pulse-wave will be at RA0. Assume Fosc = 4MHz.

Determining the period given the frequency of 1000Hz:

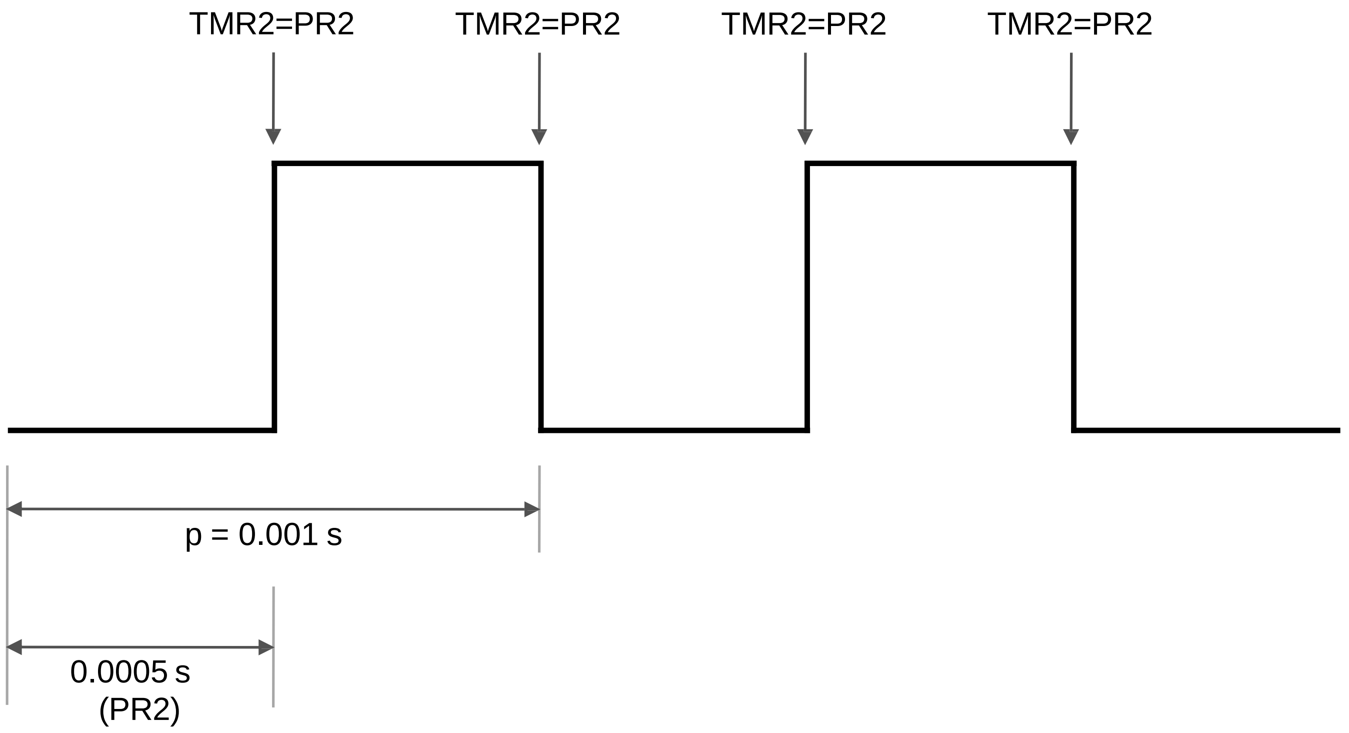
$$p = \frac{1}{f} = \frac{1}{1000} = 0.001s$$

Using a prescaler of 1:4 for Timer2, we can calculate the Timer Max Count at <u>half-cycle</u>:

From Equation 1: $$0.0005s = \frac{1}{\frac{4\text{Mhz}}{4}} \text{ x } 4 \text{ x } TimerMaxCount$$

$$TimerMaxCount = 125$$

Therefore, <u>the value of PR2 must be set to 125 to match it to TMR2</u>. When TMR2=PR2, the timeout for the half cycle is achieved then an interrupt can be generated.



Every time TMR2 would be equal to PR2, interrupt is generated and the output of RA0 is toggled.

```
void main(void)
{
   ADCON1 = 0x6;     // set all pins in PORTA as digital I/O
   TRISA = 0x00;     // sets all of PORTA to output
   RA0 = 0;          // initialize RA0 to 0
   T2CON = 0x01;     // 1:4 prescaler, Timer2 off
   TMR2IE = 1;       // enable Timer2/PR2 match interrupt (PIE1 reg)
   TMR2IF = 0;       // reset interrupt flag (PIR1 reg)
   PEIE = 1;         // enable all peripheral interrupt (INTCON reg)
   GIE = 1;          // enable all unmasked interrupts (INTCON reg)
   PR2 = 0x7D;       // match value for TMR2(125)at half cycle
   TMR2ON = 1;       // Turns on Timer2 (T2CON reg)

   for(;;)           // foreground routine
   {


   }
}
```

The value of 0x7D (125) was derived from Timer Max Count at half-cycle timeout for the frequency of 1000Hz.

```
void interrupt ISR(void)
{
  GIE = 0;              // disable all unmasked interrupts (INTCON reg)
  if(TMR2IF==1)         // checks Timer2 interrupt flag (TMR2=PR2)
  {
    TMR2IF = 0;         // clears interrupt flag
    RA0 = RA0^1;        // toggles the output signal at RA0
  }
  GIE = 1;              // enable all unmasked interrupts (INTCON reg)
}
```

University of San Carlos | Department of
COMPUTER ENGINEERING

10 minute health break

CCP Module

- There are **2 CCP modules**: **CCP1** and **CCP2**
- Both the CCP1 and CCP2 modules are identical in operation, with the exception being the operation of the special event trigger.

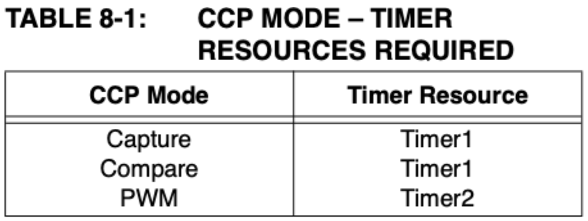


TABLE 8-1: CCP MODE – TIMER RESOURCES REQUIRED

CCP Mode	Timer Resource
Capture	Timer1
Compare	Timer1
PWM	Timer2

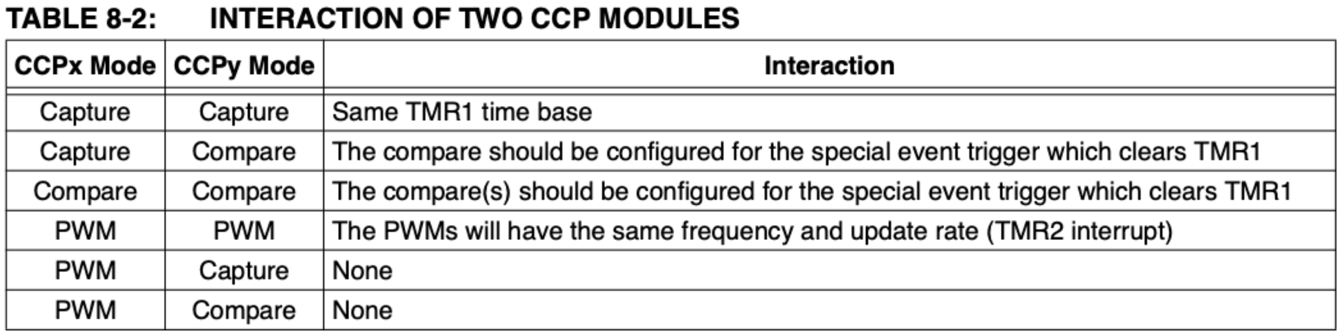


TABLE 8-2: INTERACTION OF TWO CCP MODULES

CCPx Mode	CCPy Mode	Interaction
Capture	Capture	Same TMR1 time base
Capture	Compare	The compare should be configured for the special event trigger which clears TMR1
Compare	Compare	The compare(s) should be configured for the special event trigger which clears TMR1
PWM	PWM	The PWMs will have the same frequency and update rate (TMR2 interrupt)
PWM	Capture	None
PWM	Compare	None

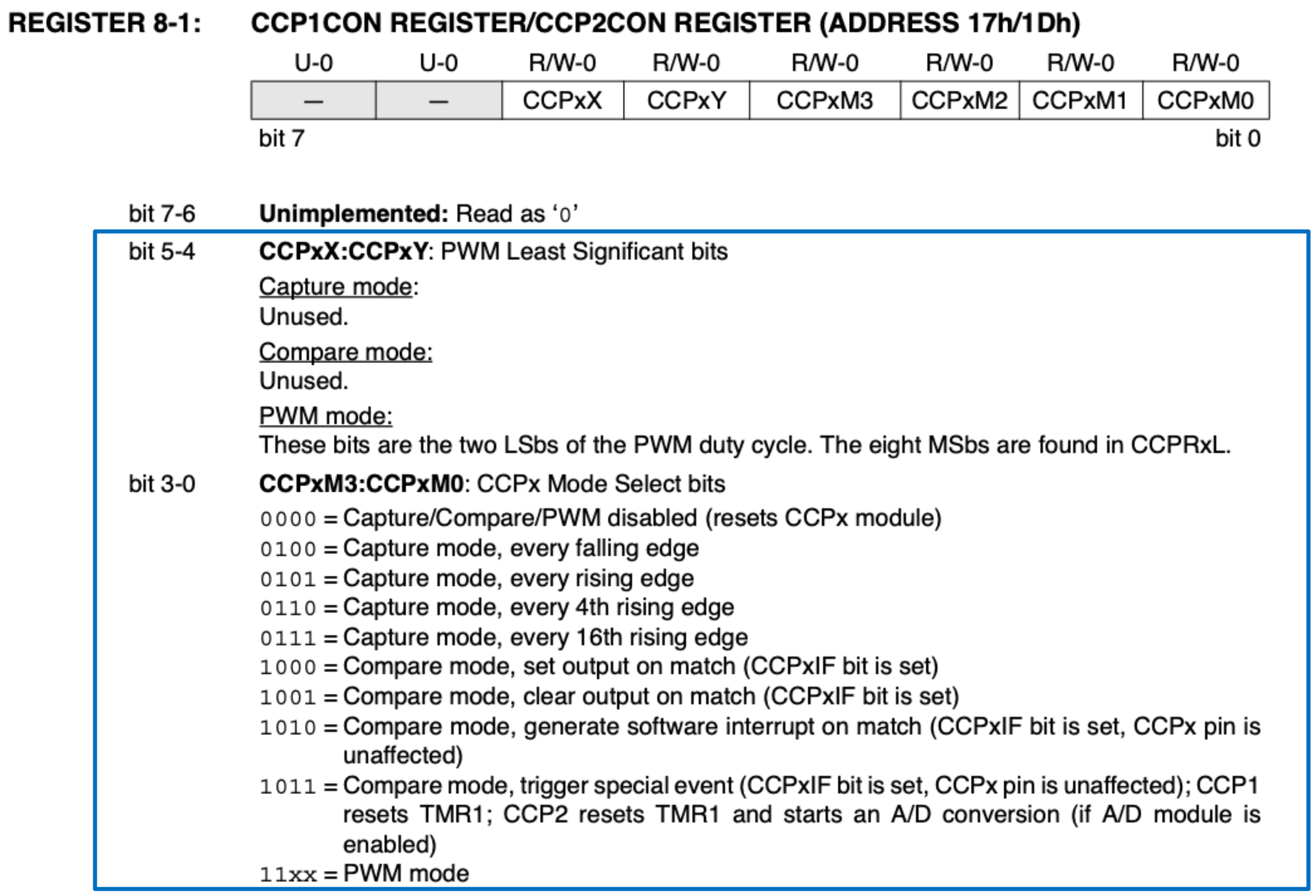
CCP1 Module

- **Capture/Compare/PWM Register 1 (CCPR1)** is comprised of two 8-bit registers: **CCPR1L** (low byte) and **CCPR1H** (high byte).
- The **CCP1CON register** controls the operation of CCP1.
- The special event trigger is generated by a compare match and will reset Timer1.

CCP2 Module

- **Capture/Compare/PWM Register 2 (CCPR2)** is comprised of two 8-bit registers: **CCPR2L** (low byte) and **CCPR2H** (high byte).
- The **CCP2CON register** controls the operation of CCP2.
- The special event trigger is generated by a compare match and will reset Timer1 and start an A/D conversion (if the A/D module is enabled).

CCP1/CCP2 Control Register



REGISTER 8-1: CCP1CON REGISTER/CCP2CON REGISTER (ADDRESS 17h/1Dh)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CCPxX	CCPxY	CCPxM3	CCPxM2	CCPxM1	CCPxM0
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5-4 **CCPxX:CCPxY**: PWM Least Significant bits

Capture mode:
Unused.

Compare mode:
Unused.

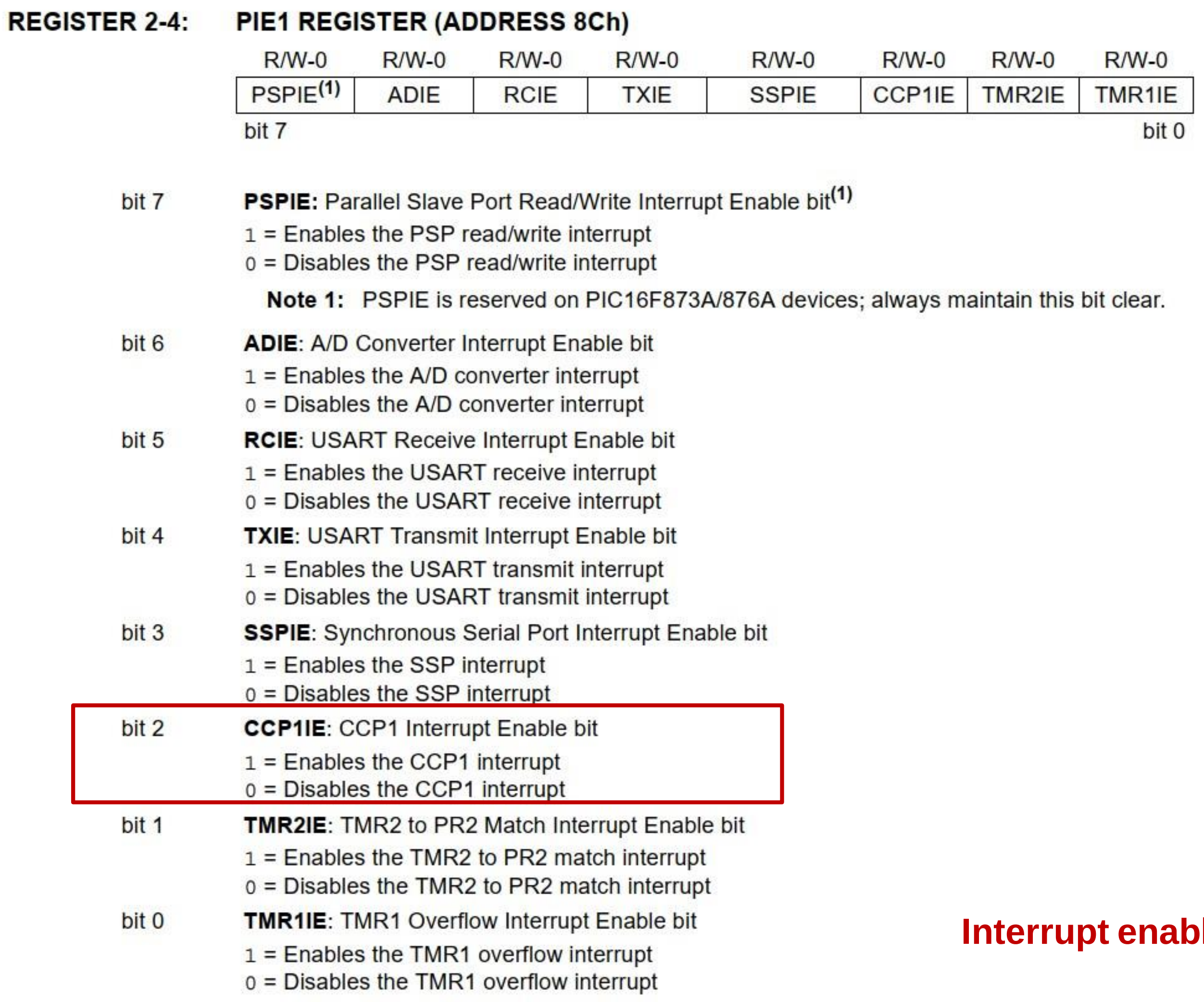
PWM mode:
These bits are the two LSbs of the PWM duty cycle. The eight MSbs are found in CCPRxL.

bit 3-0 **CCPxM3:CCPxM0**: CCPx Mode Select bits

0000 = Capture/Compare/PWM disabled (resets CCPx module)
0100 = Capture mode, every falling edge
0101 = Capture mode, every rising edge
0110 = Capture mode, every 4th rising edge
0111 = Capture mode, every 16th rising edge
1000 = Compare mode, set output on match (CCPxIF bit is set)
1001 = Compare mode, clear output on match (CCPxIF bit is set)
1010 = Compare mode, generate software interrupt on match (CCPxIF bit is set, CCPx pin is unaffected)
1011 = Compare mode, trigger special event (CCPxIF bit is set, CCPx pin is unaffected); CCP1 resets TMR1; CCP2 resets TMR1 and starts an A/D conversion (if A/D module is enabled)
11xx = PWM mode

Interrupt enable is at PIE1 register while the **flag at PIR1 register** for CCP1 while **interrupt enable and flag for CCP2 are at PIE2 and PIR2 respectively.** See PIC16F87X data sheet for more details.

Peripheral Interrupt Enable1 Register (PIE1)



REGISTER 2-4: PIE1 REGISTER (ADDRESS 8Ch)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIE[(1)]	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

bit 7	**PSPIE:** Parallel Slave Port Read/Write Interrupt Enable bit[(1)] 1 = Enables the PSP read/write interrupt 0 = Disables the PSP read/write interrupt **Note 1:** PSPIE is reserved on PIC16F873A/876A devices; always maintain this bit clear.
bit 6	**ADIE**: A/D Converter Interrupt Enable bit 1 = Enables the A/D converter interrupt 0 = Disables the A/D converter interrupt
bit 5	**RCIE**: USART Receive Interrupt Enable bit 1 = Enables the USART receive interrupt 0 = Disables the USART receive interrupt
bit 4	**TXIE**: USART Transmit Interrupt Enable bit 1 = Enables the USART transmit interrupt 0 = Disables the USART transmit interrupt
bit 3	**SSPIE**: Synchronous Serial Port Interrupt Enable bit 1 = Enables the SSP interrupt 0 = Disables the SSP interrupt
bit 2	**CCP1IE**: CCP1 Interrupt Enable bit 1 = Enables the CCP1 interrupt 0 = Disables the CCP1 interrupt
bit 1	**TMR2IE**: TMR2 to PR2 Match Interrupt Enable bit 1 = Enables the TMR2 to PR2 match interrupt 0 = Disables the TMR2 to PR2 match interrupt
bit 0	**TMR1IE**: TMR1 Overflow Interrupt Enable bit 1 = Enables the TMR1 overflow interrupt 0 = Disables the TMR1 overflow interrupt

Interrupt enable for CCP2 is at PIE2.

Peripheral Interrupt Register1 (PIR1)

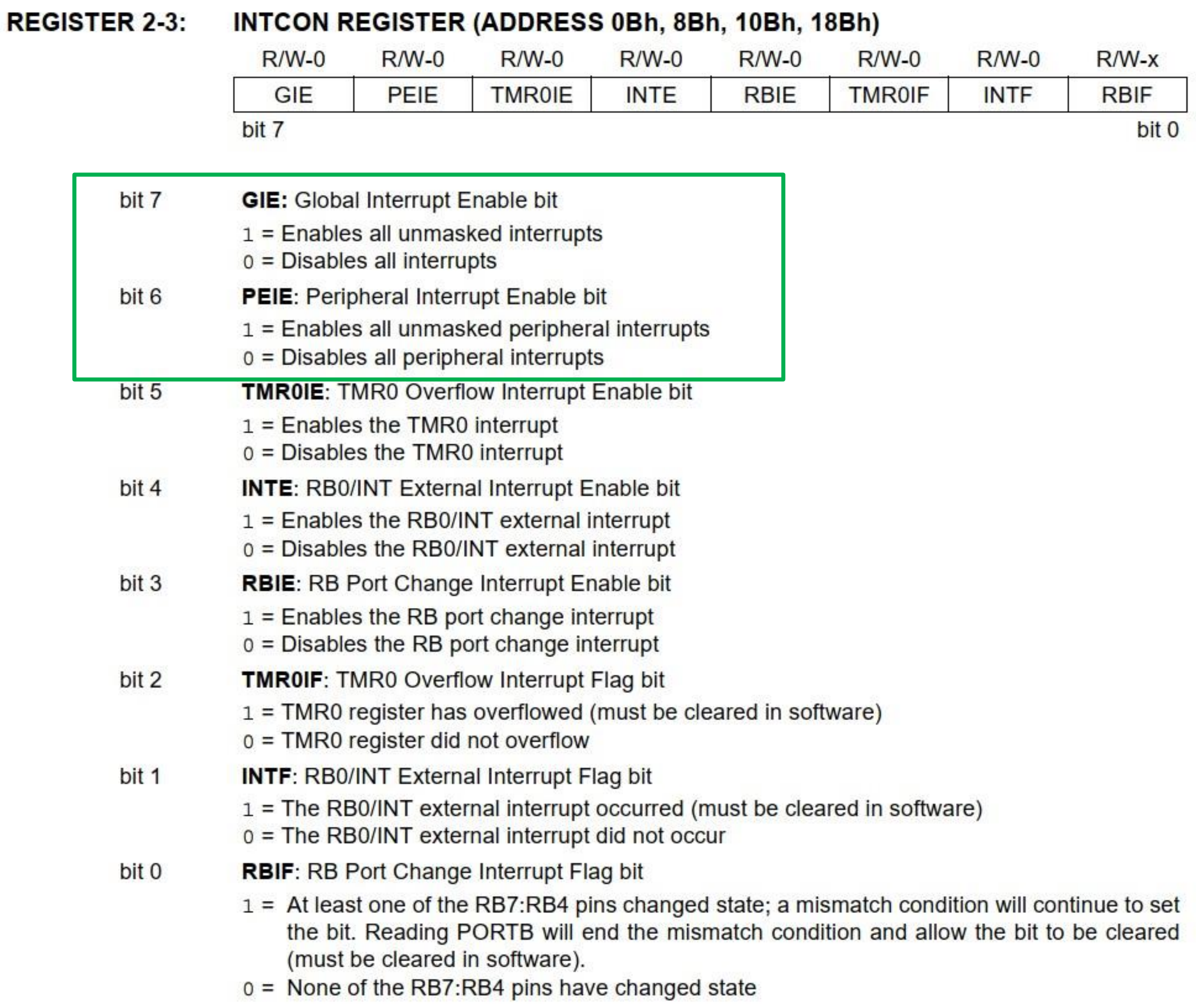
REGISTER 2-5: PIR1 REGISTER (ADDRESS 0Ch)

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIF[(1)]	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

bit 2	**CCP1IF**: CCP1 Interrupt Flag bit Capture mode: 1 = A TMR1 register capture occurred (must be cleared in software) 0 = No TMR1 register capture occurred Compare mode: 1 = A TMR1 register compare match occurred (must be cleared in software) 0 = No TMR1 register compare match occurred PWM mode: Unused in this mode.
bit 1	**TMR2IF**: TMR2 to PR2 Match Interrupt Flag bit 1 = TMR2 to PR2 match occurred (must be cleared in software) 0 = No TMR2 to PR2 match occurred
bit 0	**TMR1IF**: TMR1 Overflow Interrupt Flag bit 1 = TMR1 register overflowed (must be cleared in software) 0 = TMR1 register did not overflow

Interrupt flag for CCP2 is at PIR2.

Interrupt Control Register (INTCON)



REGISTER 2-3: INTCON REGISTER (ADDRESS 0Bh, 8Bh, 10Bh, 18Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF
bit 7							bit 0

bit 7 **GIE:** Global Interrupt Enable bit
1 = Enables all unmasked interrupts
0 = Disables all interrupts

bit 6 **PEIE**: Peripheral Interrupt Enable bit
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts

bit 5 **TMR0IE**: TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4 **INTE**: RB0/INT External Interrupt Enable bit
1 = Enables the RB0/INT external interrupt
0 = Disables the RB0/INT external interrupt

bit 3 **RBIE**: RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **TMR0IF**: TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

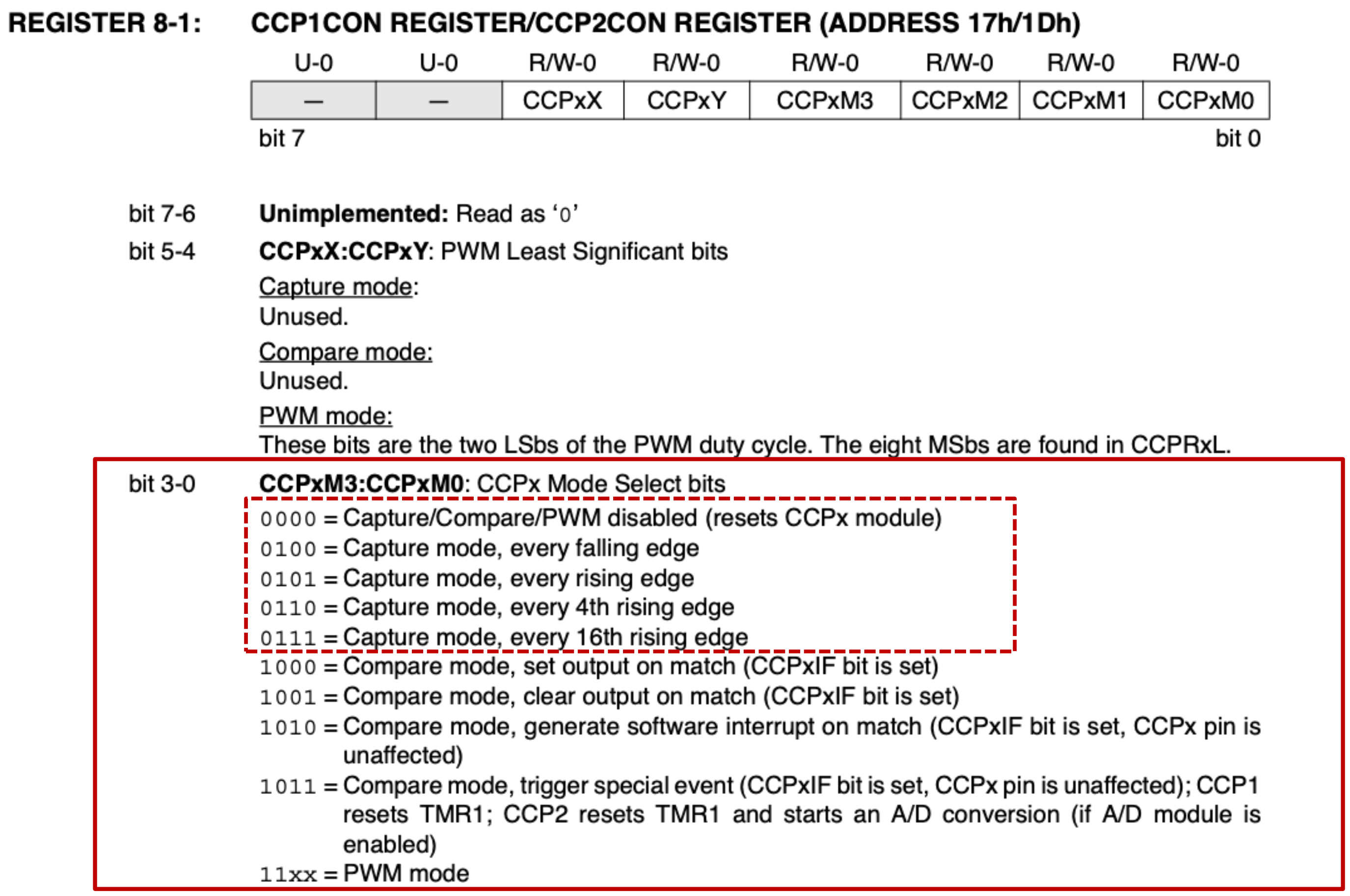
bit 1 **INTF**: RB0/INT External Interrupt Flag bit
1 = The RB0/INT external interrupt occurred (must be cleared in software)
0 = The RB0/INT external interrupt did not occur

bit 0 **RBIF**: RB Port Change Interrupt Flag bit
1 = At least one of the RB7:RB4 pins changed state; a mismatch condition will continue to set the bit. Reading PORTB will end the mismatch condition and allow the bit to be cleared (must be cleared in software).
0 = None of the RB7:RB4 pins have changed state

Capture Mode

- In **Capture mode**, CCPR1H:CCPR1L captures the 16-bit value of the TMR1 register when an event occurs on pin RC2/CCP1.
- An **event** is defined as one of the following:
  - Every falling edge
  - Every rising edge
  - Every 4th rising edge
  - Every 16th rising edge
- The type of event is configured by control bits, **CCP1M3:CCP1M0** (CCPxCON<3:0>).

CCP1/CCP2 Control Register (Capture)



REGISTER 8-1: CCP1CON REGISTER/CCP2CON REGISTER (ADDRESS 17h/1Dh)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CCPxX	CCPxY	CCPxM3	CCPxM2	CCPxM1	CCPxM0
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5-4 **CCPxX:CCPxY**: PWM Least Significant bits

Capture mode:
Unused.

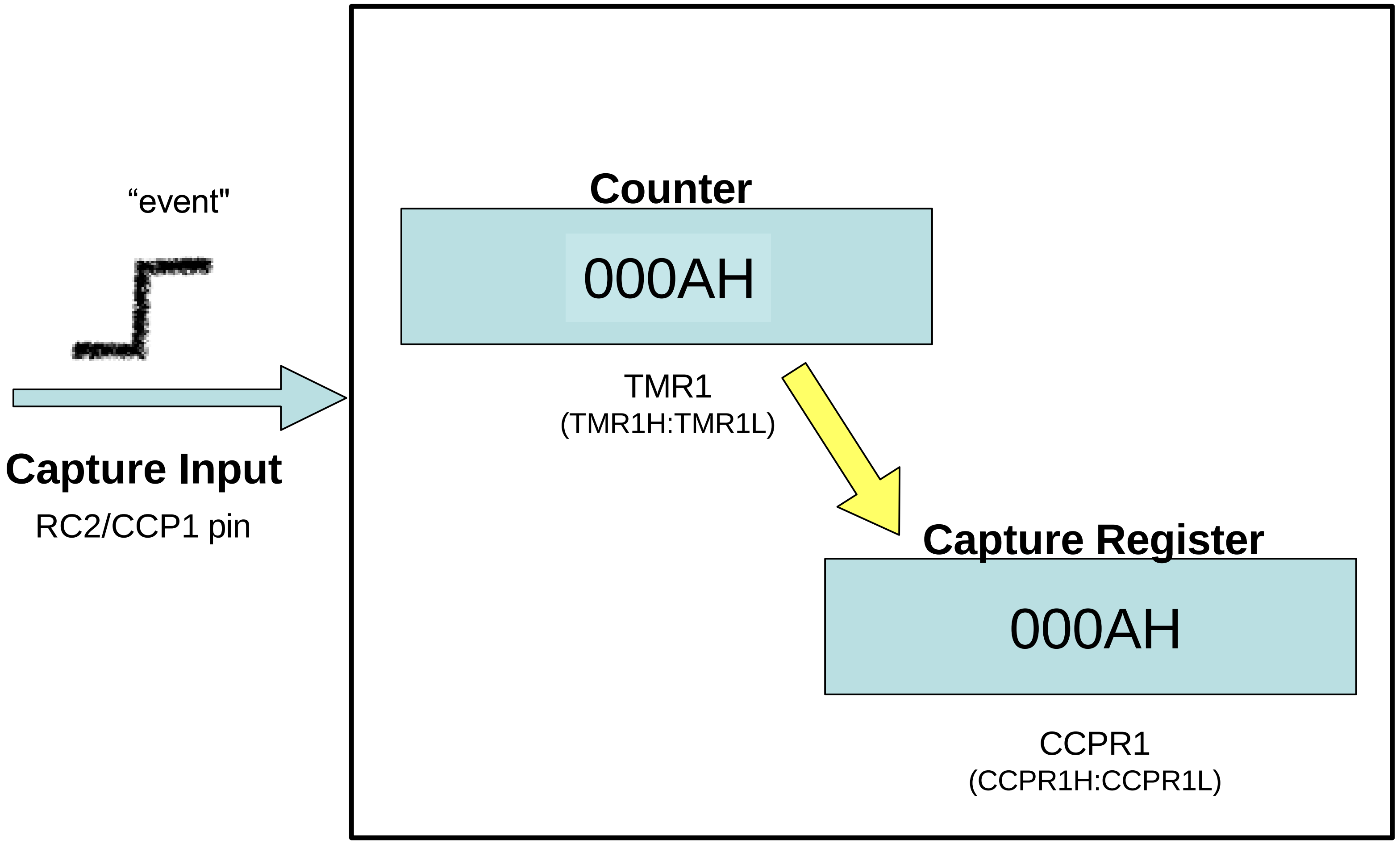
Compare mode:
Unused.

PWM mode:
These bits are the two LSbs of the PWM duty cycle. The eight MSbs are found in CCPRxL.

bit 3-0 **CCPxM3:CCPxM0**: CCPx Mode Select bits

`0000` = Capture/Compare/PWM disabled (resets CCPx module)
`0100` = Capture mode, every falling edge
`0101` = Capture mode, every rising edge
`0110` = Capture mode, every 4th rising edge
`0111` = Capture mode, every 16th rising edge
`1000` = Compare mode, set output on match (CCPxIF bit is set)
`1001` = Compare mode, clear output on match (CCPxIF bit is set)
`1010` = Compare mode, generate software interrupt on match (CCPxIF bit is set, CCPx pin is unaffected)
`1011` = Compare mode, trigger special event (CCPxIF bit is set, CCPx pin is unaffected); CCP1 resets TMR1; CCP2 resets TMR1 and starts an A/D conversion (if A/D module is enabled)
`11xx` = PWM mode

Interrupt enable is at PIE1 register while the **flag at PIR1 register** for CCP1 while **interrupt enable and flag for CCP2 are at PIE2 and PIR2 respectively.** See PIC16F87X data sheet for more details.


"event"
Capture Input
RC2/CCP1 pin
Counter
000AH
TMR1
(TMR1H:TMR1L)
Capture Register
000AH
CCPR1
(CCPR1H:CCPR1L)

Example (Capture)

- Determine the **period (ms)** of a pulse-wave signal input at RC2. Assume Fosc = 4Mhz.

Using a prescaler of 1:8 for **Timer1**, we can calculate the time per increment:

From Equation 1*:

$$timeout = \frac{1}{\frac{4\text{MHz}}{4}} \text{ x } 8 \text{ x } 1$$

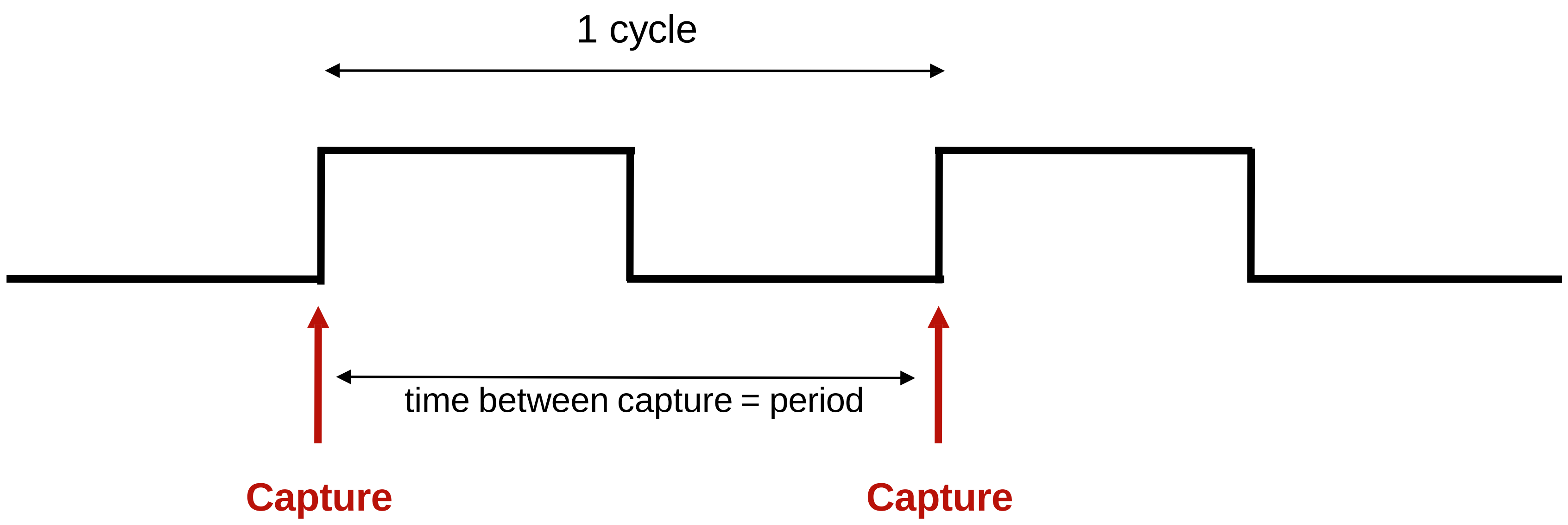
$$timeout = 8\text{x}10^{-6}\text{s}$$

Normalizing the timeout to make it a whole number:

$$timeout_{\text{n}} = 8\text{x}10^{-6}\text{s } (1\text{x}10^{6}) = 8\text{s}$$

The normalization is used to eliminate working with real (floating point) numbers.

* Timer Max Count is set to 1 since we need to determine the timeout per Timer1 increment.


1 cycle
time between capture = period
Capture
Capture

```
void main(void)
{
   TRISC = 0x04;       // set RC2 to input
   T1CON = 0x30;       // 1:8 prescaler, Timer1 off
   CCP1CON = 0x05;     // capture mode: every rising edge
   CCP1IE = 1;         // enable TMR1/CCP1 match interrupt (PIE1 reg)
   CCP1IF = 0;         // reset interrupt flag (PIR1 reg)
   PEIE = 1;           // enable all peripheral interrupt (INTCON reg)
   GIE = 1;            // enable all unmasked interrupts (INTCON reg)
   TMR1ON = 1;         // Turns on Timer1 (T1CON reg)

   for(;;)             // foreground routine
   {


   }
}
```

CCP1CON must be set to capture mode which generates an interrupt when a **rising edge event at RC2** occurs.

```
void interrupt ISR(void)
{
   int period = 0;
   GIE = 0;            // disable all unmasked interrupts (INTCON reg)
   if(CCP1IF==1)       // checks CCP1 interrupt flag
   {
     CCP1IF = 0;               // clears interrupt flag
     TMR1 = 0;                 // resets TMR1
     period = CCPR1/1000;      // transfers captured TMR1 value
                               // normalize the value (make the number smaller)
     period = period*8;        // multiply by the normalized TMR1 timeout
   }
   GIE = 1;            // enable all unmasked interrupts (INTCON reg)
}
```

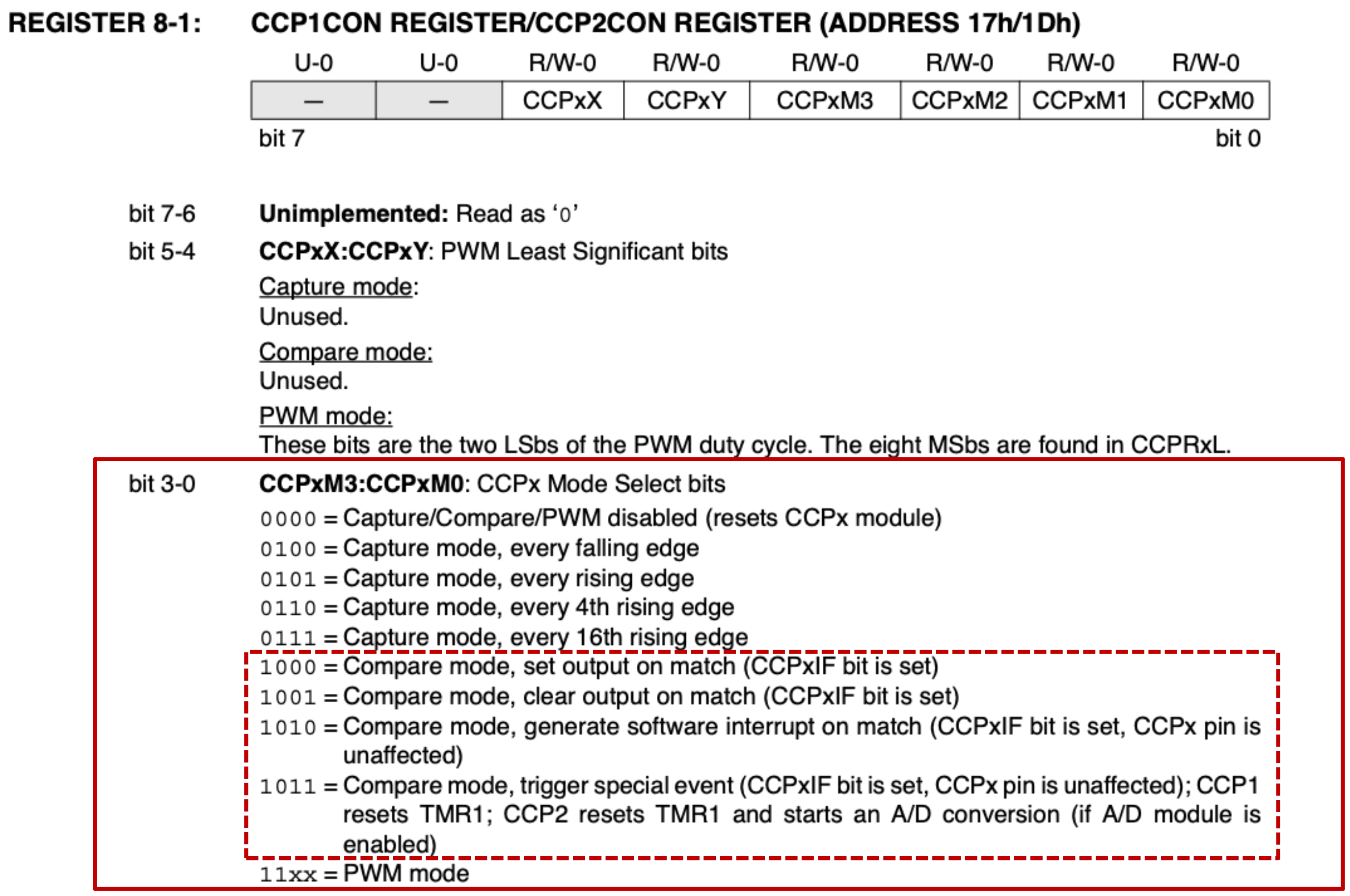
The value of the variable *period* is an integer which is in ms. For example, if the value is 500 then it is equivalent to 500 ms.

Frequency above 1000Hz will result in a value of 0.

Compare Mode

- In **Compare** mode, the 16-bit CCPR1 register value is <u>constantly compared against the TMR1 register pair</u> value.
- **When a match occurs**, the RC2/CCP1 pin is:
  - Driven high
  - Driven low
  - Remains unchanged
- The action on the pin is based on the value of control bits, **CCP1M3:CCP1M0** (CCP1CON<3:0>). At the same time, **interrupt flag bit CCP1IF** is set.

CCP1/CCP2 Control Register (Compare)



REGISTER 8-1: CCP1CON REGISTER/CCP2CON REGISTER (ADDRESS 17h/1Dh)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CCPxX	CCPxY	CCPxM3	CCPxM2	CCPxM1	CCPxM0
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5-4 **CCPxX:CCPxY**: PWM Least Significant bits

Capture mode:
Unused.

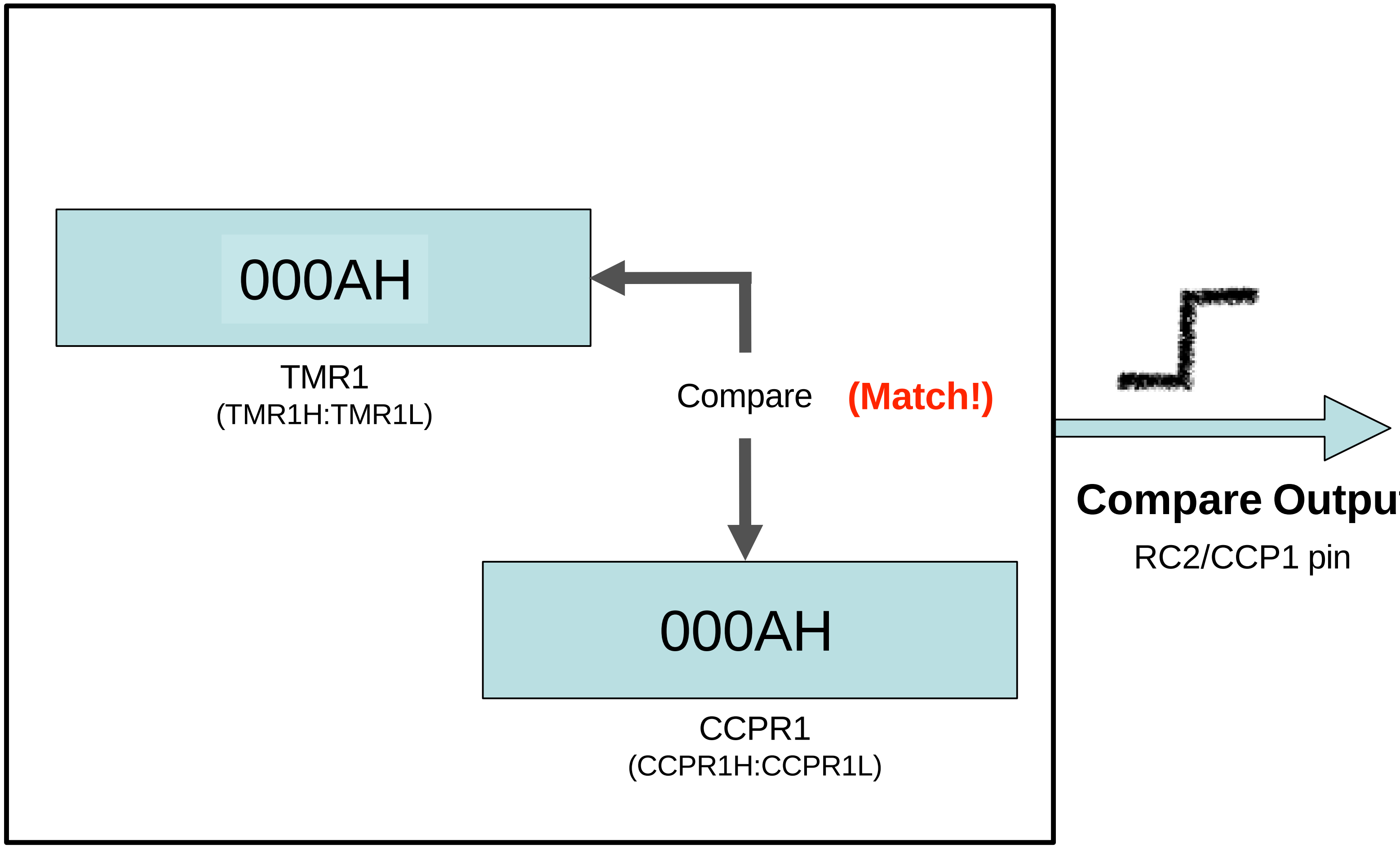
Compare mode:
Unused.

PWM mode:
These bits are the two LSbs of the PWM duty cycle. The eight MSbs are found in CCPRxL.

bit 3-0 **CCPxM3:CCPxM0**: CCPx Mode Select bits

- `0000` = Capture/Compare/PWM disabled (resets CCPx module)
- `0100` = Capture mode, every falling edge
- `0101` = Capture mode, every rising edge
- `0110` = Capture mode, every 4th rising edge
- `0111` = Capture mode, every 16th rising edge
- `1000` = Compare mode, set output on match (CCPxIF bit is set)
- `1001` = Compare mode, clear output on match (CCPxIF bit is set)
- `1010` = Compare mode, generate software interrupt on match (CCPxIF bit is set, CCPx pin is unaffected)
- `1011` = Compare mode, trigger special event (CCPxIF bit is set, CCPx pin is unaffected); CCP1 resets TMR1; CCP2 resets TMR1 and starts an A/D conversion (if A/D module is enabled)
- `11xx` = PWM mode

Interrupt enable is at PIE1 register while the **flag at PIR1 register** for CCP1 while **interrupt enable and flag for CCP2 are at PIE2 and PIR2 respectively.** See PIC16F87X data sheet for more details.


000AH
TMR1
(TMR1H:TMR1L)
Compare
(Match!)
000AH
CCPR1
(CCPR1H:CCPR1L)
Compare Output
RC2/CCP1 pin

Example (Compare)

- Generate a pulse-wave with a frequency of 100Hz at 50% duty cycle. The output pulse-wave will be at RA0. Assume Fosc = 4MHz.

Determining the period given the frequency of 100Hz:

$$p = \frac{1}{f} = \frac{1}{100} = 0.01s$$

Using a prescaler of 1:4 for **Timer1**, we can calculate the Timer Max Count at half-cycle:

From Equation 1:

$$0.005s = \frac{1}{\frac{4\text{MHz}}{4}} \text{ x } 4 \text{ x } TimerMaxCount$$

$$TimerMaxCount = 1250 \text{ or } 0\text{x}4\text{E}2$$

Therefore, CCPR1 must be set to 0x4E2. When TMR1=CCPR1, then it generates an interrupt and toggles the output at RA0.

```
void main(void)
{
  ADCON1 = 0x6;        // set all pins in PORTA as digital  I/O
  TRISA = 0x00;        // sets all of PORTA to output
  RA0 = 0;             // initialize RA0 to 0
  T1CON = 0x20;        // 1:4 prescaler, Timer1 off
  CCP1CON = 0x0A;      // compare mode: generate interrupt  on match
  CCP1IE = 1;          // enable TMR1/CCP1 match interrupt  (PIE1 reg)
  CCP1IF = 0;          // reset interrupt flag (PIR1 reg)
  CCPR1 = 0x4E2;       // set the match value to TMR1
  PEIE = 1;            // enable all peripheral interrupt (INTCON reg)
  GIE = 1;             // enable all unmasked interrupts (INTCON reg)
  TMR1ON = 1;          // Turns on Timer1 (T1CON reg)

  for(;;)              // foreground routine
  {


  }
}
```

CCP1CON must be set to **compare mode** which generates an interrupt when **TMR1=CCPR1**. This example does not use the CCP1 output at RC2.

```
void interrupt ISR(void)
{
    GIE = 0;            // disable all unmasked interrupts (INTCON reg)
    if(CCP1IF==1)       // checks CCP1 interrupt flag
    {
        CCP1IF = 0;     // clears interrupt flag
        RA0 = RA0^1;    // toggles the output signal at RA0
    }
    GIE = 1;            // enable all unmasked interrupts (INTCON reg)
}
```

Summary of Registers for Capture, Compare, TMR1

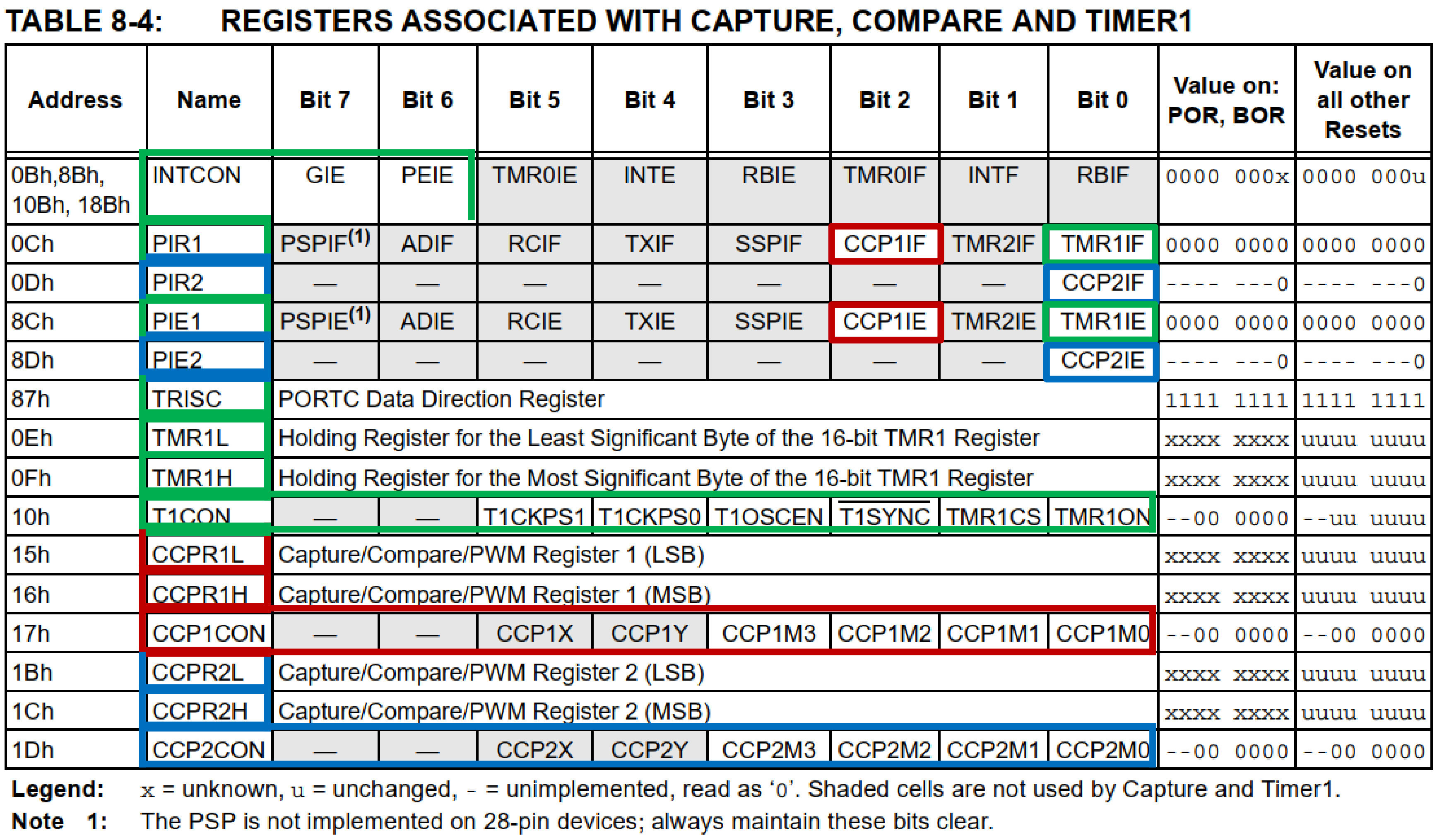


TABLE 8-4: REGISTERS ASSOCIATED WITH CAPTURE, COMPARE AND TIMER1

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
0Bh,8Bh, 10Bh, 18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF[1]	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
0Dh	PIR2	—	—	—	—	—	—	—	CCP2IF	---- ---0	---- ---0
8Ch	PIE1	PSPIE[1]	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
8Dh	PIE2	—	—	—	—	—	—	—	CCP2IE	---- ---0	---- ---0
87h	TRISC	PORTC Data Direction Register								1111 1111	1111 1111
0Eh	TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
0Fh	TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	--00 0000	--uu uuuu
15h	CCPR1L	Capture/Compare/PWM Register 1 (LSB)								xxxx xxxx	uuuu uuuu
16h	CCPR1H	Capture/Compare/PWM Register 1 (MSB)								xxxx xxxx	uuuu uuuu
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	--00 0000
1Bh	CCPR2L	Capture/Compare/PWM Register 2 (LSB)								xxxx xxxx	uuuu uuuu
1Ch	CCPR2H	Capture/Compare/PWM Register 2 (MSB)								xxxx xxxx	uuuu uuuu
1Dh	CCP2CON	—	—	CCP2X	CCP2Y	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	--00 0000

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by Capture and Timer1.

Note 1: The PSP is not implemented on 28-pin devices; always maintain these bits clear.

🟥 For CCP1 module

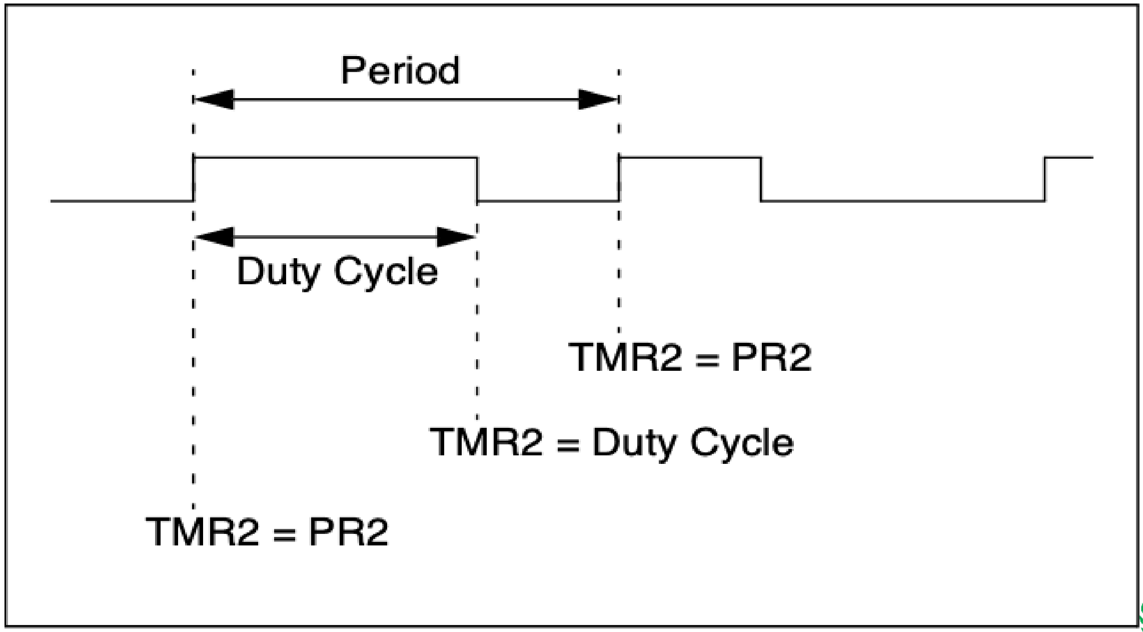
🟦 For CCP2 module

🟩 Common to both, including TMR1

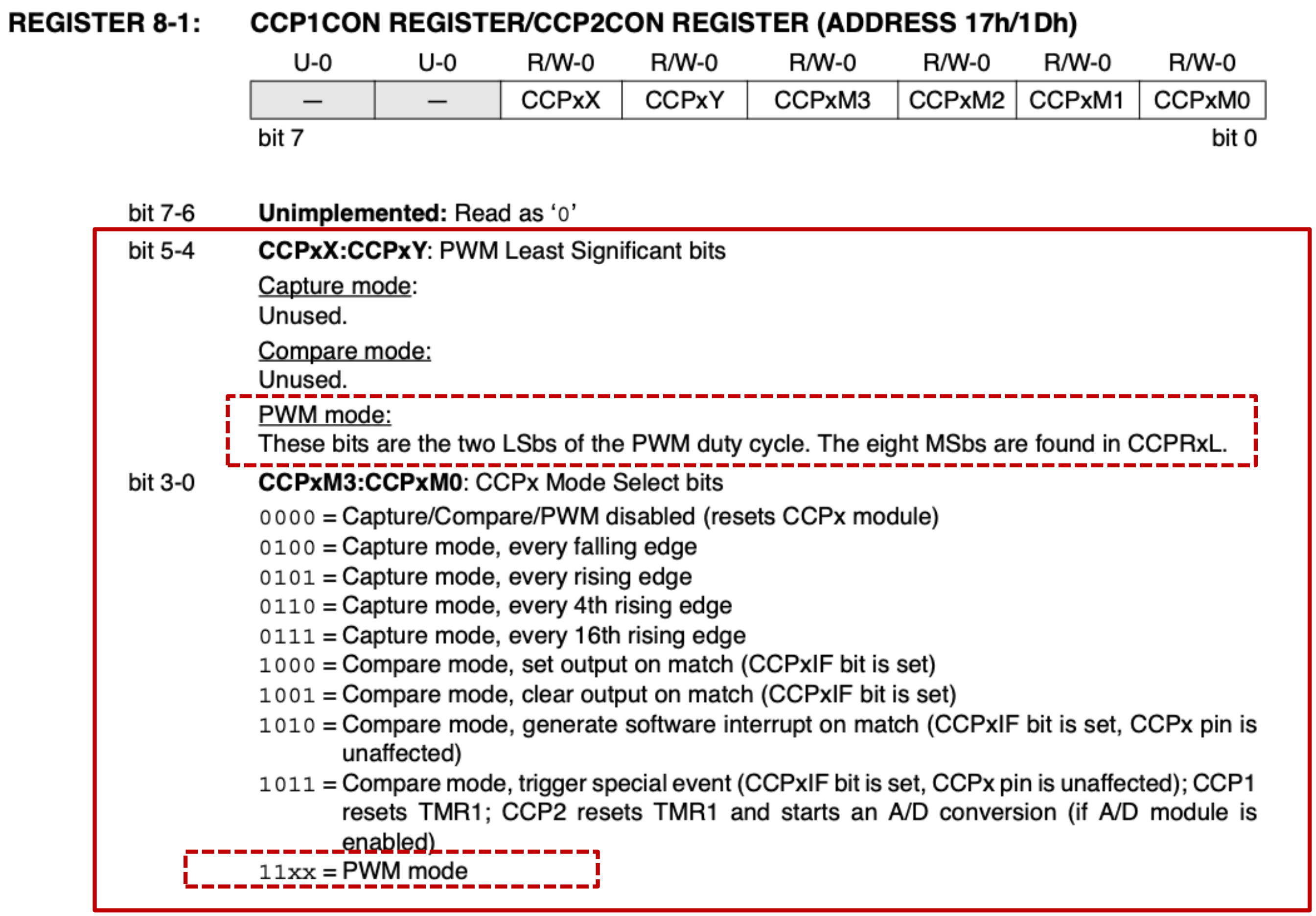
PWM

- In **Pulse Width Modulation (PWM) mode**, the CCPx pin produces up to a 10-bit resolution PWM output. Since the CCP1 pin is multiplexed with the PORTC data latch, the **TRISC<2>** bit must be cleared to make the CCP1 pin an output.

FIGURE 8-4: PWM OUTPUT



CCP1/CCP2 Control Register (PWM)



REGISTER 8-1: CCP1CON REGISTER/CCP2CON REGISTER (ADDRESS 17h/1Dh)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CCPxX	CCPxY	CCPxM3	CCPxM2	CCPxM1	CCPxM0
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as ‘0’

bit 5-4 **CCPxX:CCPxY**: PWM Least Significant bits

Capture mode:
Unused.

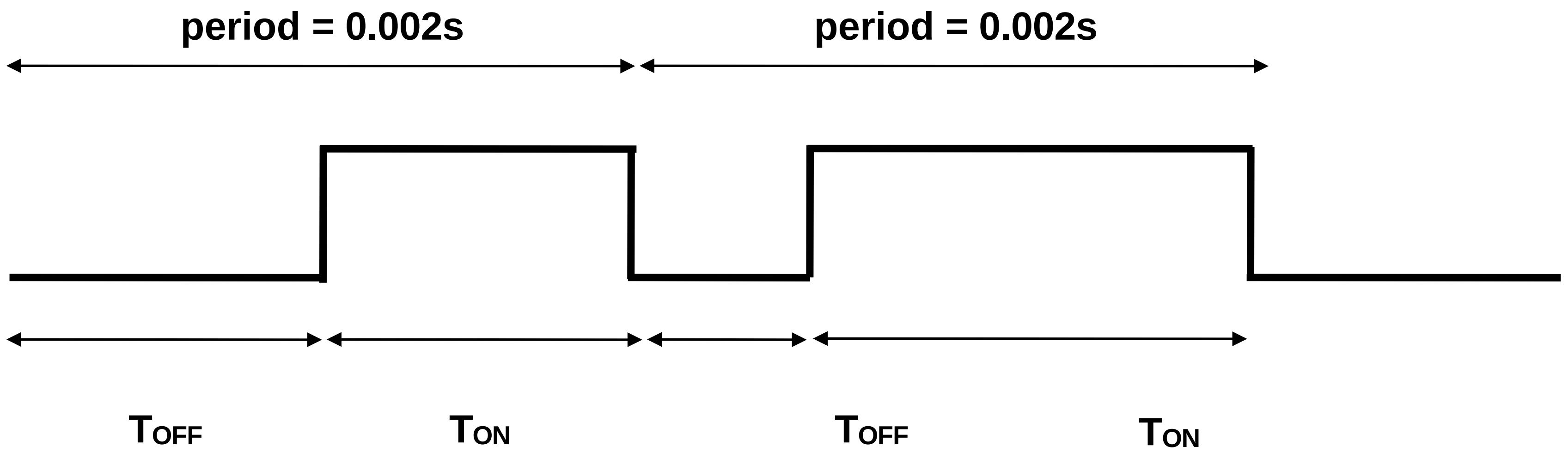
Compare mode:
Unused.

PWM mode:
These bits are the two LSbs of the PWM duty cycle. The eight MSbs are found in CCPRxL.

bit 3-0 **CCPxM3:CCPxM0**: CCPx Mode Select bits

0000 = Capture/Compare/PWM disabled (resets CCPx module)
0100 = Capture mode, every falling edge
0101 = Capture mode, every rising edge
0110 = Capture mode, every 4th rising edge
0111 = Capture mode, every 16th rising edge
1000 = Compare mode, set output on match (CCPxIF bit is set)
1001 = Compare mode, clear output on match (CCPxIF bit is set)
1010 = Compare mode, generate software interrupt on match (CCPxIF bit is set, CCPx pin is unaffected)
1011 = Compare mode, trigger special event (CCPxIF bit is set, CCPx pin is unaffected); CCP1 resets TMR1; CCP2 resets TMR1 and starts an A/D conversion (if A/D module is enabled)
11xx = PWM mode

Interrupt enable is at PIE1 register while the **flag at PIR1 register** for CCP1 while **interrupt enable and flag for CCP2 are at PIE2 and PIR2 respectively.** See PIC16F87X data sheet for more details.



period = $T_{OFF} + T_{ON}$

@ 50% duty cycle: $T_{OFF} = T_{ON}$

@ 70% duty cycle: $T_{ON} = 0.70 * \text{period}$

PWM Period

- The **PWM period** is specified by writing to the **PR2** register. The PWM period can be calculated using the following formula:

$$\text{PWM Period} = [(\text{PR2}) + 1] \cdot 4 \cdot T_{OSC} \cdot (\text{TMR2 Prescale Value}) \qquad (2)$$

- **When TMR2 is equal to PR2**, the following three events occur on the next increment cycle:
  - **TMR2** is cleared
  - The **CCP1** pin is set (*exception*: *if PWM duty cycle = 0%, the CCP1 pin will not be set*)
  - The **PWM duty cycle** is latched from **CCPR1L** into **CCPR1H**

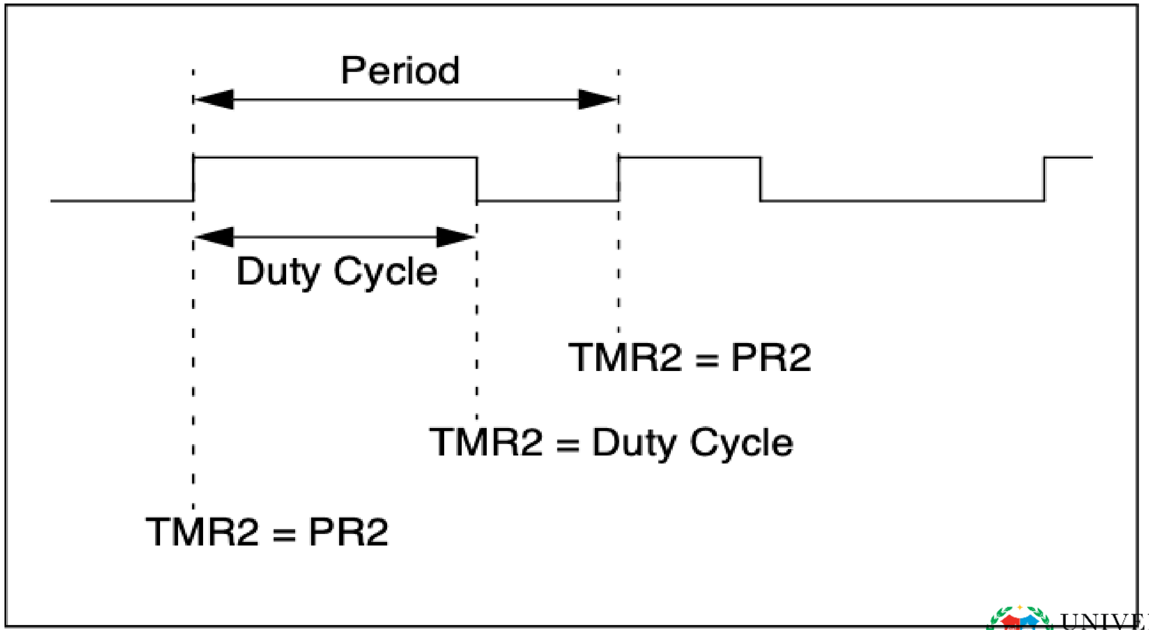
Note: Tosc = 1/Fosc where is Fosc is oscillator frequency.

PWM Period

- **When TMR2 is equal to PR2**, the following three events occur on the next increment cycle:
  - **TMR2** is cleared
  - The **CCP1** pin is set (*exception*: *if PWM duty cycle = 0%, the CCP1 pin will not be set*)
  - The **PWM duty cycle** is latched from **CCPR1L** into **CCPR1H**

(2)

FIGURE 8-4: PWM OUTPUT



PWM Duty Cycle

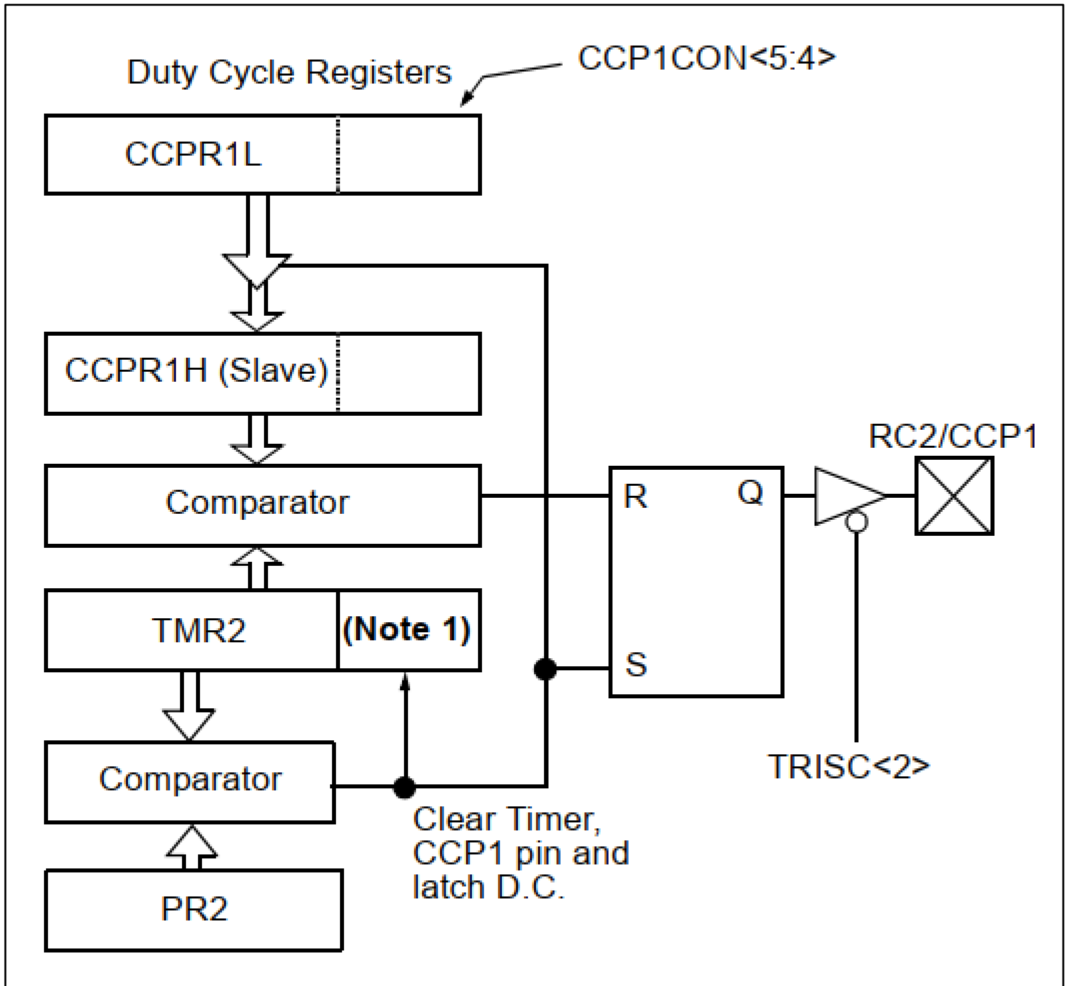
- The **PWM duty cycle** is specified by writing to the **CCPR1L register and to the CCP1CON<5:4> bits**.
- Up to 10-bit resolution is available.
- The **CCPR1L** contains the eight MSBs and the **CCP1CON<5:4>** contains the two LSBs.
- This 10-bit value is represented by **CCPR1L:CCP1CON<5:4>**. The following equation is used to calculate the **PWM duty cycle in time**:

$$\text{PWM Duty Cycle} = (\text{CCPR1L:CCP1CON<5:4>}) \bullet \text{TOSC} \bullet (\text{TMR2 Prescale Value}) \quad (3)$$

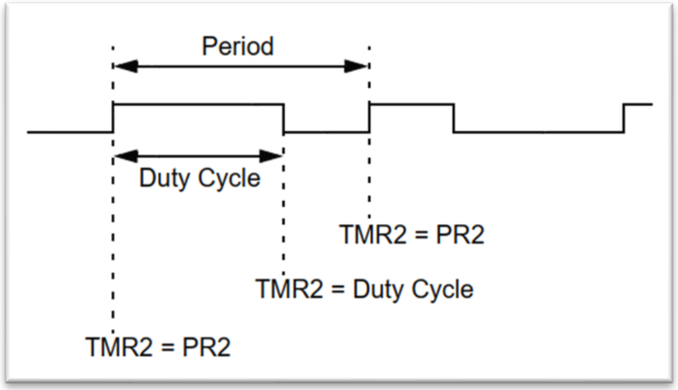
- **CCPR1L and CCP1CON<5:4>** can be written to at any time, but the duty cycle value is not latched into CCPR1H until after a match between PR2 and TMR2 occurs (i.e., the period is complete).
- In **PWM mode**, **CCPR1H** is a read-only register.

Note: Tosc = 1/Fosc where is Fosc is oscillator frequency.

Simplified PWM Block Diagram



Note 1: The 8-bit timer is concatenated with 2-bit internal Q clock, or 2 bits of the prescaler, to create 10-bit time base.



Setup for PWM

The following steps should be taken when configuring the CCP module for **PWM operation**:

1. Set the **PWM period** by writing to the **PR2 register**.
2. Set the **PWM duty cycle** by writing to the **CCPR1L register and CCP1CON<5:4> bits**.
3. Make the **CCP1 pin** an **output** by clearing the **TRISC<2> bit**.
4. Set the **TMR2 prescale value** and enable **Timer2** by writing to **T2CON**.
5. Configure the **CCP1 module** for PWM operation.

Example (PWM)

Configure the CCP1 Module to operate in **PWM mode** which outputs a signal with a 70% duty cycle at a frequency of 500Hz. Assume that Fosc = 4MHz.

Calculating the value of **PR2** given a period of 1/500Hz (0.002s) and **Timer2 prescaler** at 1:16:

From Equation 2:

$$0.002s = [(PR2) + 1] \text{ x } 4 \text{ x } (2.5\text{x}10^{-7}) \text{ x } 16$$

$$PR2 = 124\ (0\text{x}7C)$$

Calculating the **10-bit resolution value** given a **duty cycle of 70%.** The **PWM duty cycle** should be given in <u>time</u> that is 0.7 x 0.002s = 1.4ms.

From Equation 3:

$$1.4\text{x}10^{-3} = (CCPR1L{:}CCP1CON < 5{:}4 >) \text{ x } \frac{1}{4\text{MHz}} \text{ x } 16$$

$$(CCPR1L{:}CCP1CON < 5{:}4 >) = 350 \text{ or } 0101\ 0111\ 10_2$$

$$CCPR1L = 0101\ 0111_2 \text{ or } 0\text{x}57$$

$$CCP1CON < 5{:}4 >= 10_2 \text{ or } 0\text{x}2$$

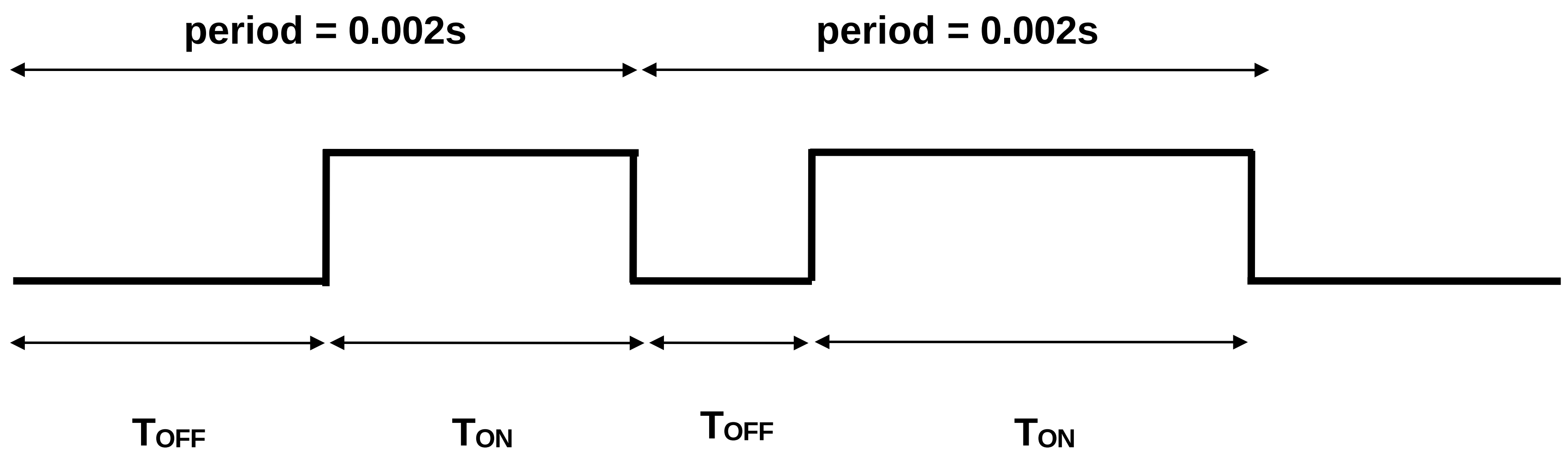
UNIVERSITY of SAN CARLOS
SCIENTIA • VIRTUS • DEVOTIO

```
void main(void)
{
  /* following the steps in setting up PWM */
  PR2 = 0x7C;          // set value for PR2
  CCPR1L = 0x57;       // set value for (8 MSbs)
  CCP1CON = 0x2C;      // set value for (2 LSbs), PWM mode
  TRISC = 0x00;        // sets all of PORTC (RC2) to output
  RC2 = 0;             // initialize RC2 to 0
  T2CON = 0x06;        // 1:16 prescaler, Timer2 on

  for(;;)              // foreground routine
  {


  }
}
```

CCP1CON sets the **2-bit LSB of the PWM resolution** as well as **setting the mode to PWM**. See the PIC16F87X data sheet for more information.



period = $T_{OFF} + T_{ON}$

@ 50% duty cycle: $T_{OFF} = T_{ON}$

@ 70% duty cycle: T_{ON} = 0.70 * period

Summary of Registers for PWM & TMR2

TABLE 8-5: REGISTERS ASSOCIATED WITH PWM AND TIMER2

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
0Bh,8Bh, 10Bh, 18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF[1]	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
0Dh	PIR2	—	—	—	—	—	—	—	CCP2IF	---- ---0	---- ---0
8Ch	PIE1	PSPIE[1]	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
8Dh	PIE2	—	—	—	—	—	—	—	CCP2IE	---- ---0	---- ---0
87h ✓	TRISC	PORTC Data Direction Register								1111 1111	1111 1111
11h	TMR2	Timer2 Module's Register								0000 0000	0000 0000
92h ✓	PR2	Timer2 Module's Period Register								1111 1111	1111 1111
12h ✓	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	-000 0000
15h ✓	CCPR1L	Capture/Compare/PWM Register 1 (LSB)								xxxx xxxx	uuuu uuuu
16h	CCPR1H	Capture/Compare/PWM Register 1 (MSB)								xxxx xxxx	uuuu uuuu
17h ✓	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	--00 0000
1Bh	CCPR2L	Capture/Compare/PWM Register 2 (LSB)								xxxx xxxx	uuuu uuuu
1Ch	CCPR2H	Capture/Compare/PWM Register 2 (MSB)								xxxx xxxx	uuuu uuuu
1Dh	CCP2CON	—	—	CCP2X	CCP2Y	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	--00 0000

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

Note 1: Bits PSPIE and PSPIF are reserved on 28-pin devices; always maintain these bits clear.

🟥 For CCP1 module

🟦 For CCP2 module

🟩 Common to both, including TMR2

✓ For PWM setup

UNIVERSITY of SAN CARLOS
SCIENTIA • VIRTUS • DEVOTIO

University of San Carlos | Department of
COMPUTER ENGINEERING

CpE 3201
Embedded Systems

End of Lecture

References:

- De Leon, Hilary L., Microcontroller Programming and Interfacing, 2006.
- Goankar, Ramesh, Fundamentals of Microcontrollers and Applications in Embedded Systems (with the PIC18 MCU Family),2007.
- PIC16F87X Data Sheet, Microchip Technology Inc. 2003.