The University of Texas at Arlington

Lecture 12
Timers and CCP
(Capture/Compare/PWM)

CSE 3442/5442

Embedded Systems 1

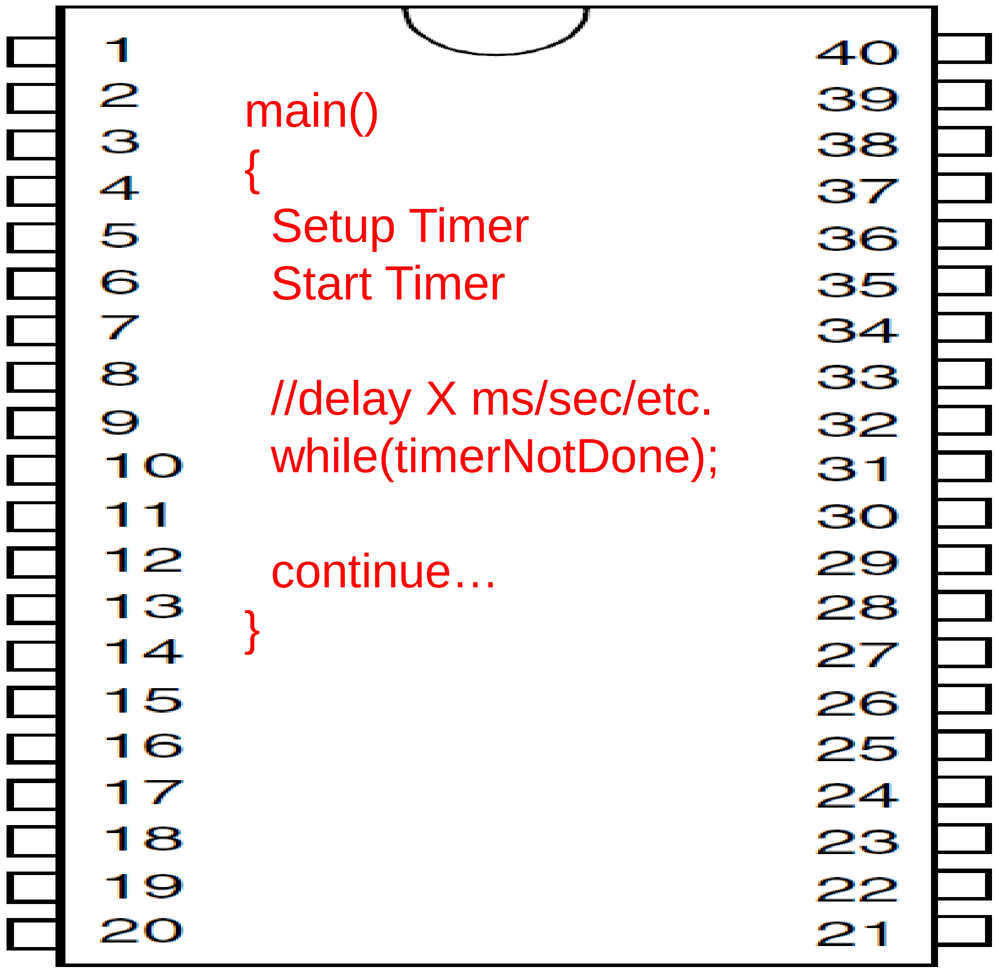
Based heavily on slides by Dr. Gergely Záruba and Dr. Roger Walker

PIC Timers

- PIC18 family microcontrollers have 2 to 5 timers on-board
- Timers can be used to **generate time delays** or to **count (outside) events** happening “in the background”
- Some timers can also be used to control timing of other peripherals (the designer needs to pay attention to that)
- Every timer needs a clock that will make it to count
- PIC18 timers have the option to use at most ¼ of the main clock’s frequency or use a separate external signal for clocking
  - **Timer**: uses internal clock source (F_{osc} /4)
    - “Wait this amount of fixed time”
  - **Counter**: fed pulses through one of the PIC’s pins
    - “Count how many events/pulses occur on a pin”

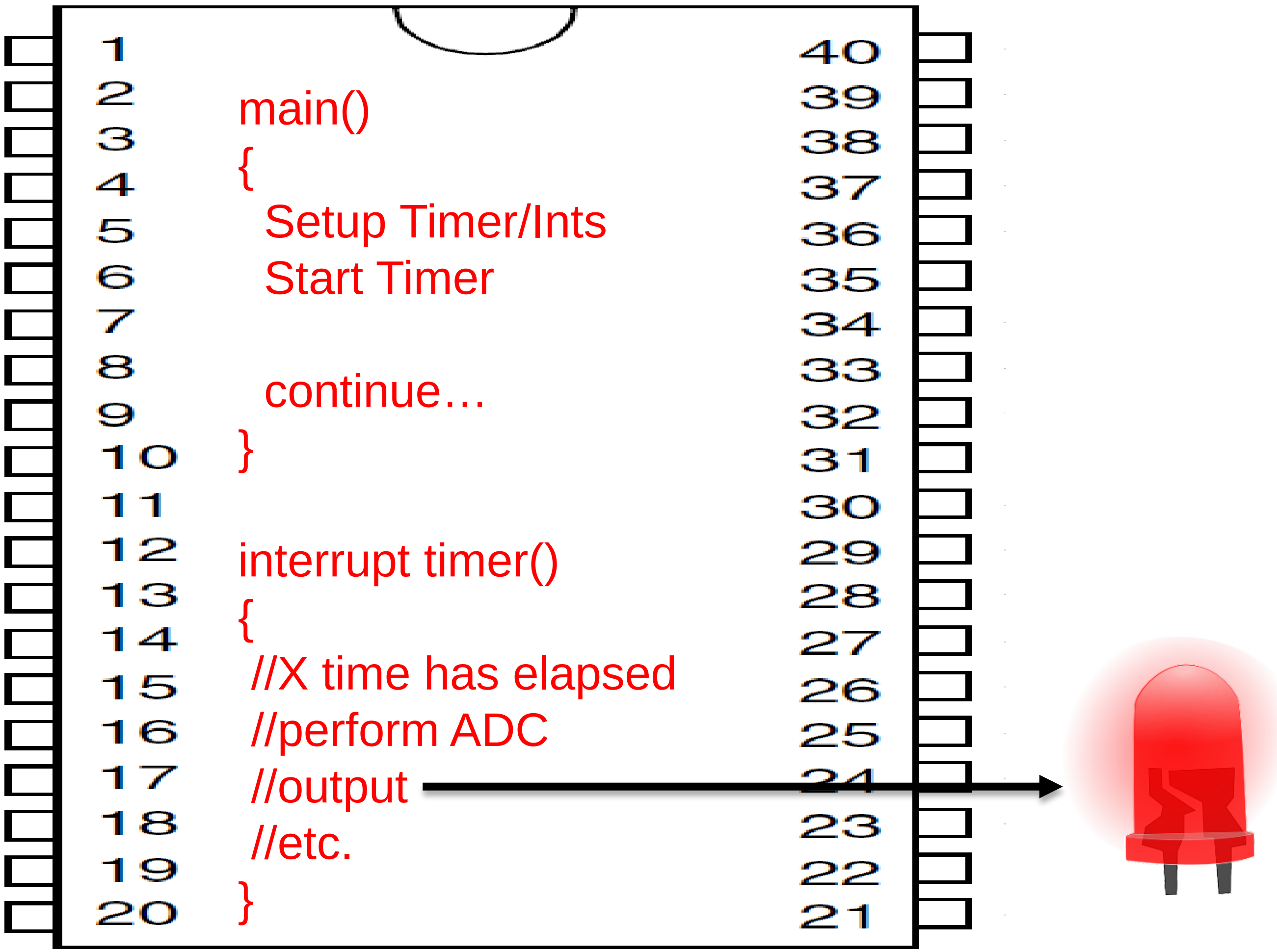
Timer

- Software specified time delay or “background” time elapsed



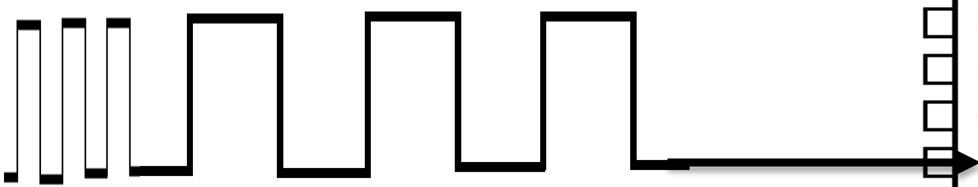
Timer

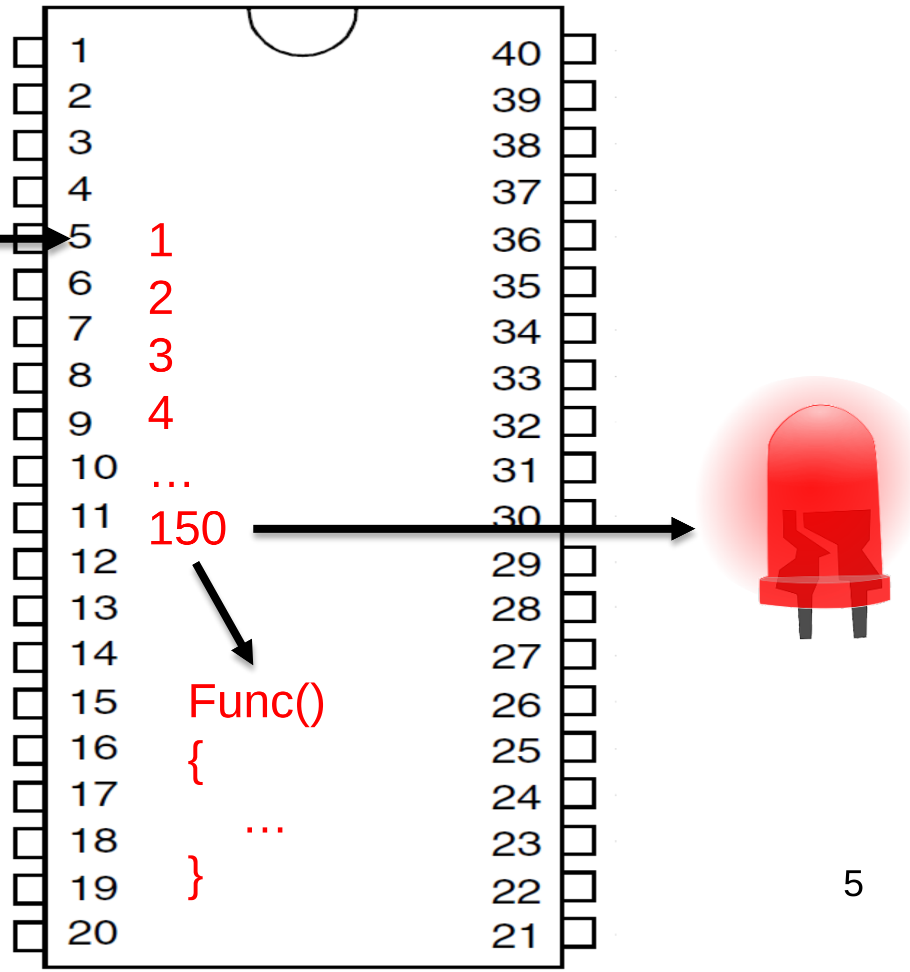
- Software specified time delay or “background” time elapsed



Counter

- Count external/outside events and pulses





- IR Sensor
- Encoder
- Button
- Water Flow Sensor

Timer Length (Width or Mode) and Preload

- 8-bit timers: Can count from 0 – 255
- 16-bit timers: Can count from 0 – 65,536
- 32-bit timers: Can count from 0 – 4,294,967,296
- Can start counting at 0 or any **preload** within range

- Ex. 8-bit Overflow:

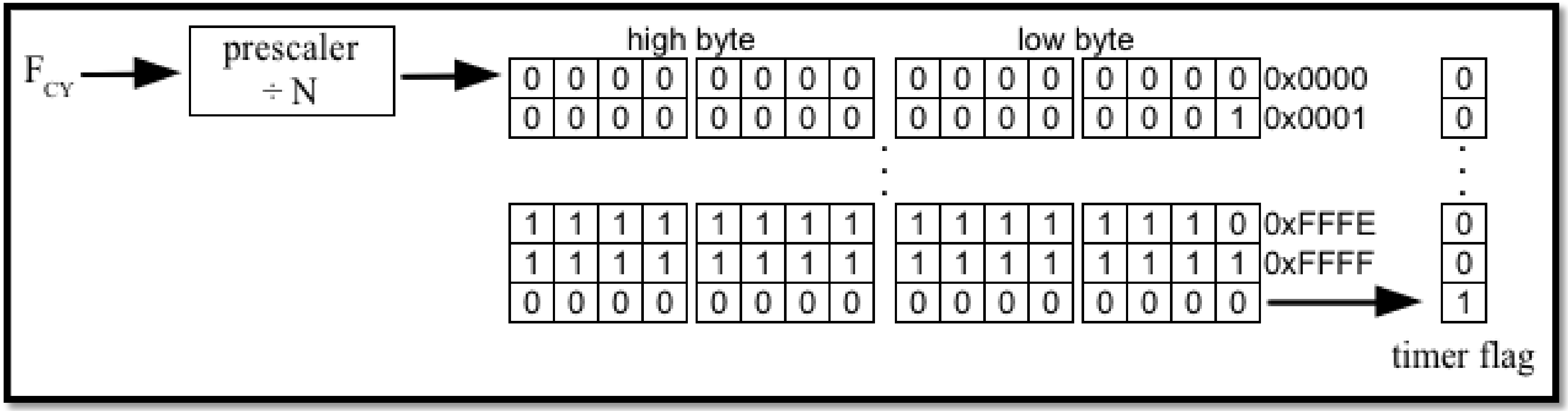
OV

  - 0 → 1 → 2 → … → 254 → 255 → 0 → 1 → 2…

OV → Reload Preload

  - 200 → 201 → 202 → … → 254 → 255 → 0 → 200 → 201…

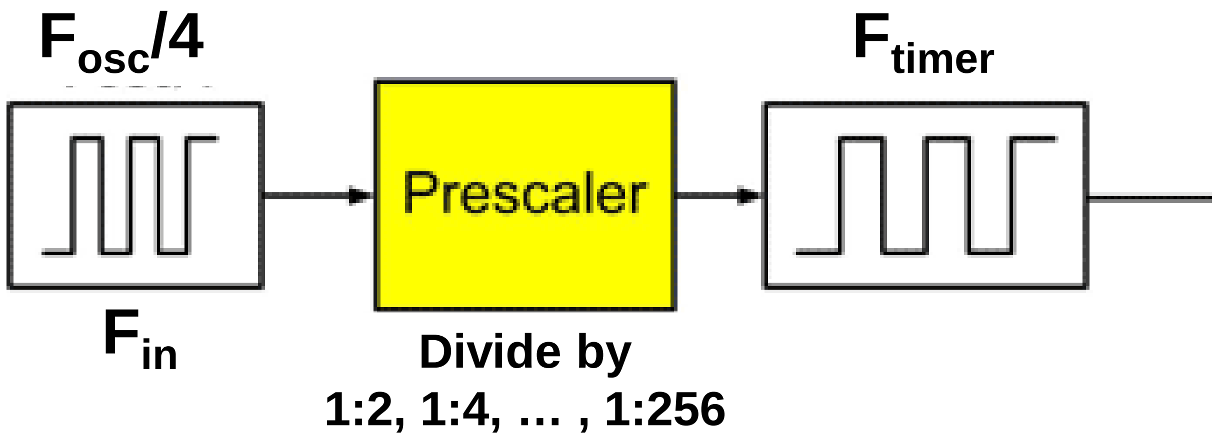
Timer Overflow



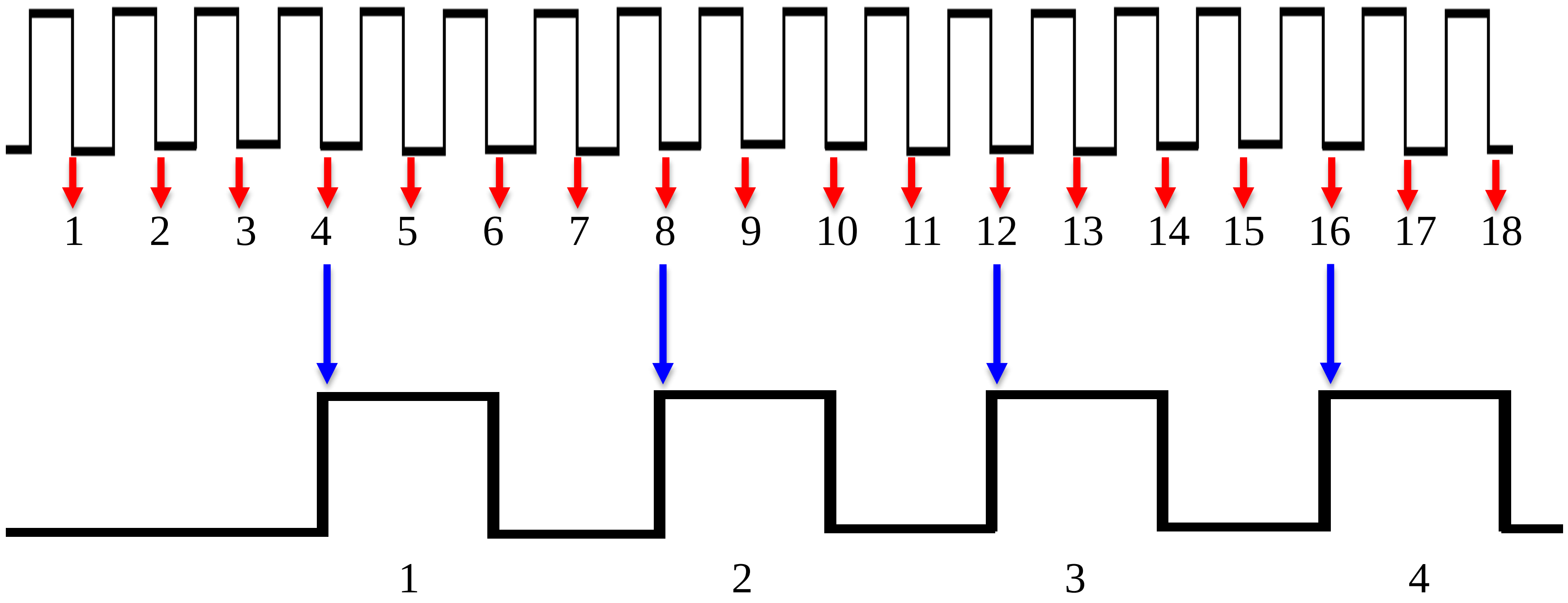
Source: http://roberthall.net/PIC18F4550_Timers

Prescaler

- Sometimes the frequency is too fast
- A prescaler divides the clock source to obtain a smaller frequency (less frequent)
  - 1, 2, 4, 8, 16, 32, 64, 128, 256…



Ex: F/4



How to Calculate Example (Want 1.2 sec Delay)

- d = 1.2sec (time period)
- F_{osc} = 10MHz
- $F_{in} = F_{osc} / 4$ = 2.5MHz
- 16-bit Timer: 0 – 65,536

- $X = d * F_{in}$ = 1.2s * 2.5Mhz = 3,000,000
  - **3,000,000 cycles/ticks occur in a 1.2s time span**

How to Calculate Example (Want 1.2 sec Delay)

- $X = d * F_{in} = 1.2s * 2.5Mhz = 3{,}000{,}000$
  - 3,000,000 cycles/ticks occur in a 1.2s time span
- Use **prescaler** to bring down X to fit into the 16-bit Timer register (0 – 65,536)
  - 3,000,000 / 4 = 750,000 (> 65,536)
  - 3,000,000 / 16 = 187,500 (> 65,536)
  - 3,000,000 / 32 = 93,750 (> 65,536)
  - 3,000,000 / 64 = 46,875 (< 65,536)

→Use Prescaler 1:64

How to Calculate Example (Want 1.2 sec Delay)

- $X = d * F_{in} = 1.2s * 2.5Mhz = 3,000,000$
  - 3,000,000 cycles/ticks occur in a 1.2s time span
- Using **Prescaler** 1:64 to find **Preload** value
  - Now 46,875 ticks/cycles will occur in 1.2s span
- **Preload** = 65,536 – 3,000,000/64

  = 65,536 – 46,875

  = 18,661

Instead of counting 0 → 65,536

Now count from **18,661 → 65,536**

How to Calculate Example (Want 1.2 sec Delay)

- So if we want a 1.2 second delay when using a 10MHz oscillator…
  1. We select a 16-bit Timer
  2. Select the prescaler 1:64
  3. Load the timer register with 18,661 (dec)
  4. Turn on the Timer
  5. When the Timer overflows, we know that exactly 1.2s has passed

Four PIC18F452 Timers

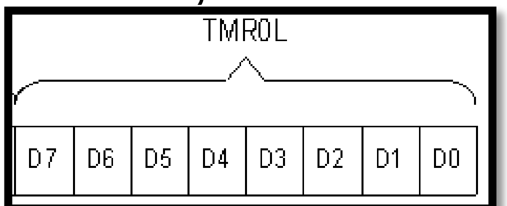
- **Timer0**: 8 or 16-bit timer/counter
  - T0CON, TMR0H:TMR0L
  - Prescalers: 1:2, 1:4, … ,1:128, 1:256
- **Timer1**: 16-bit timer/counter
  - T1CON, TMR1H:TMR1L
  - Prescalers: 1:1, 1:2, 1:4, 1:8
- **Timer2**: 8-bit timer
  - T2CON, TMR2L
  - Prescalers: 1:1, 1:4, 1:16 and Postscalers: 1:1 … 1:16
- **Timer3**: 16-bit timer/counter
  - T3CON, TMR3H:TMR3L
  - Prescalers: 1:1, 1:2, 1:4, 1:8

TABLE 4-1: SPECIAL FUNCTION REGISTER MAP

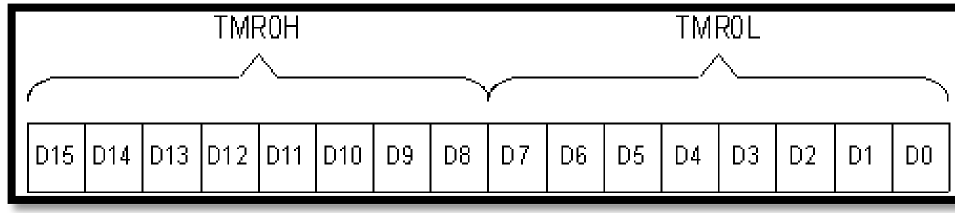
Address	Name	Address	Name	Address	Name	Address	Name
FFFh	TOSU	FDFh	INDF2[3]	FBFh	CCPR1H	F9Fh	IPR1
FFEh	TOSH	FDEh	POSTINC2[3]	FBEh	CCPR1L	F9Eh	PIR1
FFDh	TOSL	FDDh	POSTDEC2[3]	FBDh	CCP1CON	F9Dh	PIE1
FFCh	STKPTR	FDCh	PREINC2[3]	FBCh	CCPR2H	F9Ch	—
FFBh	PCLATU	FDBh	PLUSW2[3]	FBBh	CCPR2L	F9Bh	—
FFAh	PCLATH	FDAh	FSR2H	FBAh	CCP2CON	F9Ah	—
FF9h	PCL	FD9h	FSR2L	FB9h	—	F99h	—
FF8h	TBLPTRU	FD8h	STATUS	FB8h	—	F98h	—
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	—	F97h	—
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	—	F96h	TRISE[2]
FF5h	TABLAT	FD5h	T0CON	FB5h	—	F95h	TRISD[2]
FF4h	PRODH	FD4h	—	FB4h	—	F94h	TRISC
FF3h	PRODL	FD3h	OSCCON	FB3h	TMR3H	F93h	TRISB
FF2h	INTCON	FD2h	LVDCON	FB2h	TMR3L	F92h	TRISA
FF1h	INTCON2	FD1h	WDTCON	FB1h	T3CON	F91h	—
FF0h	INTCON3	FD0h	RCON	FB0h	—	F90h	—
FEFh	INDF0[3]	FCFh	TMR1H	FAFh	SPBRG	F8Fh	—
FEEh	POSTINC0[3]	FCEh	TMR1L	FAEh	RCREG	F8Eh	—
FEDh	POSTDEC0[3]	FCDh	T1CON	FADh	TXREG	F8Dh	LATE[2]
FECh	PREINC0[3]	FCCh	TMR2	FACh	TXSTA	F8Ch	LATD[2]
FEBh	PLUSW0[3]	FCBh	PR2	FABh	RCSTA	F8Bh	LATC
FEAh	FSR0H	FCAh	T2CON	FAAh	—	F8Ah	LATB
FE9h	FSR0L	FC9h	SSPBUF	FA9h	EEADR	F89h	LATA
FE8h	WREG	FC8h	SSPADD	FA8h	EEDATA	F88h	—
FE7h	INDF1[3]	FC7h	SSPSTAT	FA7h	EECON2	F87h	—
FE6h	POSTINC1[3]	FC6h	SSPCON1	FA6h	EECON1	F86h	—
FE5h	POSTDEC1[3]	FC5h	SSPCON2	FA5h	—	F85h	—
FE4h	PREINC1[3]	FC4h	ADRESH	FA4h	—	F84h	PORTE[2]
FE3h	PLUSW1[3]	FC3h	ADRESL	FA3h	—	F83h	PORTD[2]
FE2h	FSR1H	FC2h	ADCON0	FA2h	IPR2	F82h	PORTC
FE1h	FSR1L	FC1h	ADCON1	FA1h	PIR2	F81h	PORTB
FE0h	BSR	FC0h	—	FA0h	PIE2	F80h	PORTA

Timer0

- Timer0 can be used as an **8-bit** or as a **16-bit** timer
- Thus, two **SFR**s are used to contain the count:

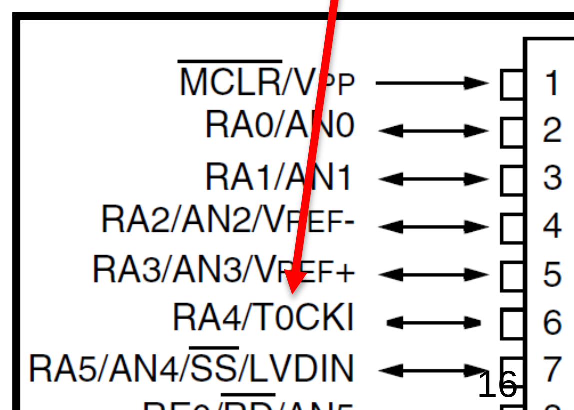


or



- **T0CON** is the control register
- **TMR0IF** is the interrupt flag in the **INTCON** register
- The clock source for Timer0 may be internal or external

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TMR0L	Timer0 Module Low Byte Register							
TMR0H	Timer0 Module High Byte Register							
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
TRISA	—	PORTA Data Direction Register						



Timer0 Control Register T0CON

10-1: T0CON: TIMER0 CONTROL REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
bit 7							bit 0

bit 7 **TMR0ON:** Timer0 On/Off Control bit
1 = Enables Timer0
0 = Stops Timer0

bit 6 **T08BIT**: Timer0 8-bit/16-bit Control bit
1 = Timer0 is configured as an 8-bit timer/counter
0 = Timer0 is configured as a 16-bit timer/counter

bit 5 **T0CS**: Timer0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKO)

bit 4 **T0SE**: Timer0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA**: Timer0 Prescaler Assignment bit
1 = TImer0 prescaler is NOT assigned. Timer0 clock input bypasses prescaler.
0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.

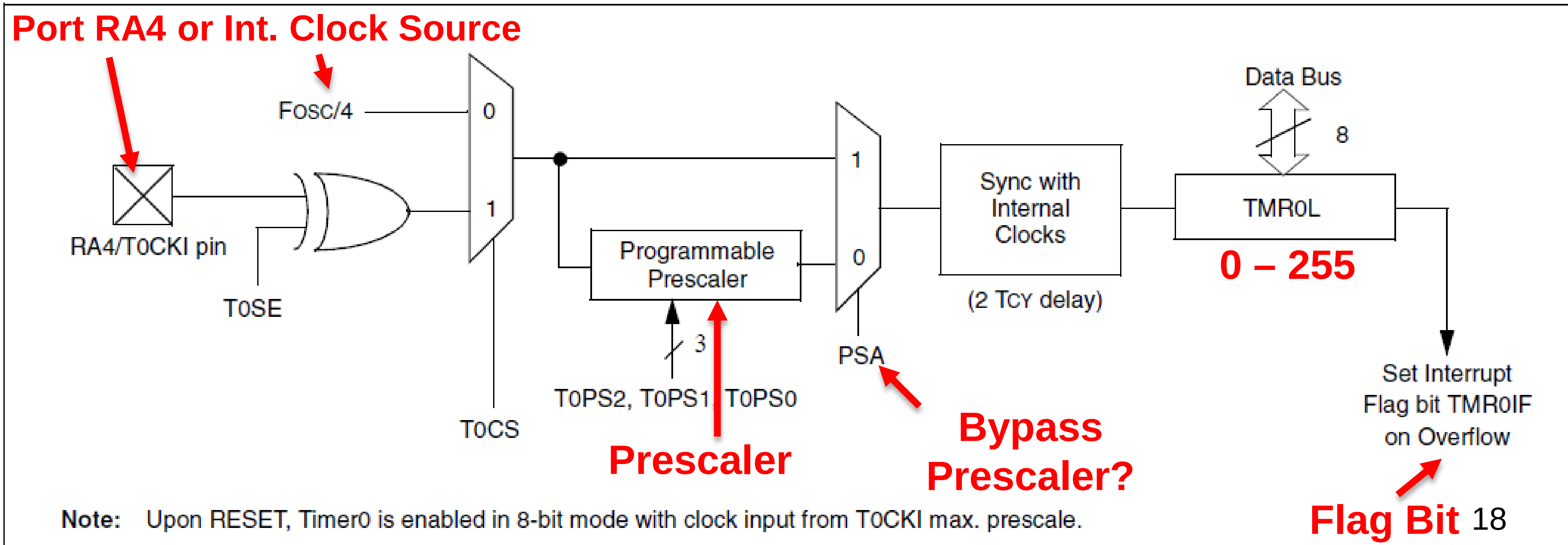
bit 2-0 **T0PS2:T0PS0**: Timer0 Prescaler Select bits
111 = 1:256 prescale value
110 = 1:128 prescale value
101 = 1:64 prescale value
100 = 1:32 prescale value
011 = 1:16 prescale value
010 = 1:8 prescale value
001 = 1:4 prescale value
000 = 1:2 prescale value

- Note that timer interrupt enable/flag bits are in registers related to interrupts (e.g., INTCON)
- When the timer overflows, TMR0IF is set.
- 16- vs. 8-bit timer
- Prescalers are useful for large time delays

Timer0 8-bit Programming

1. Select **8-bit** mode and **prescaler**
2. Load **TMR0L** with preload value (ignore **TMR0H**)
3. Start timer (**TMR0ON = 1**)
4. Monitor **TMR0IF** (or set interrupt on it)
5. When **TMR0IF** is set, stop the timer, reset the flag (and if needed go to step 2)

FIGURE 10-1: TIMER0 BLOCK DIAGRAM IN 8-BIT MODE

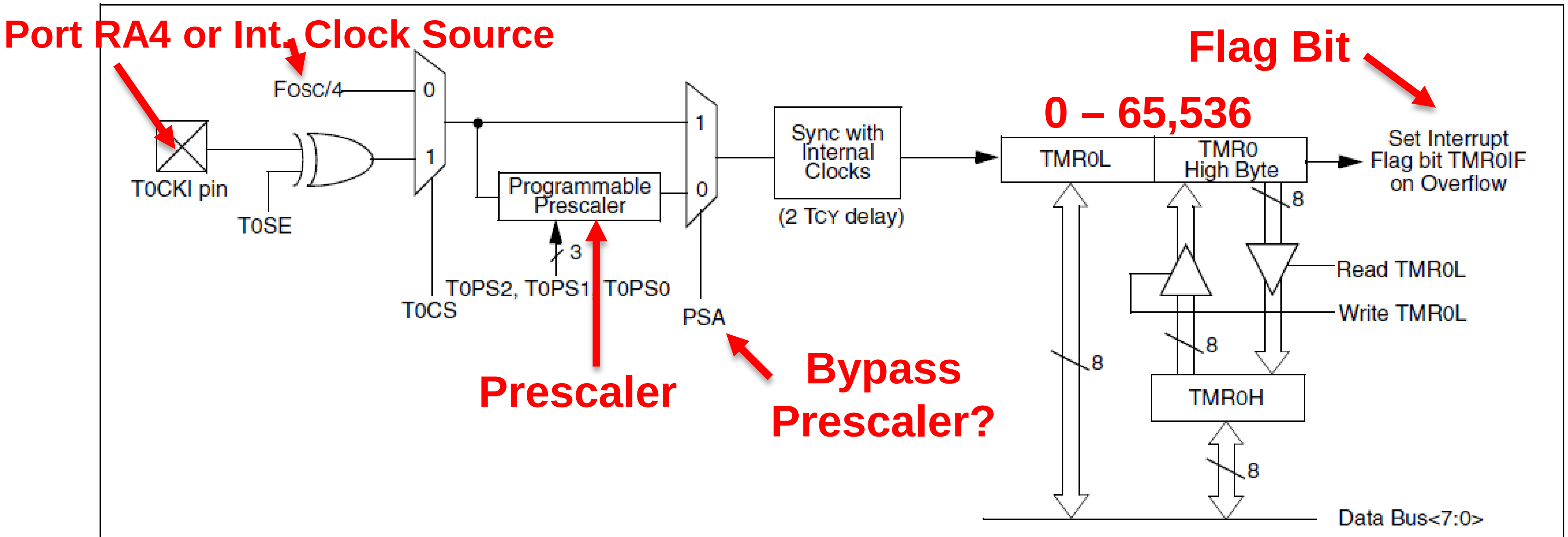


Note: Upon RESET, Timer0 is enabled in 8-bit mode with clock input from T0CKI max. prescale.

Timer0 16-bit Programming

1. Select **16-bit** mode and **prescaler**
2. Load **TMR0H** and then **TMR0L** with preload values (load HIGH first!!)
   – Ex: 18,661 dec = 0x48E5 → TMR0H = 0x48 and TMR0L = 0xE5
3. Start timer (**TMR0ON = 1**)
4. Monitor **TMR0IF** (or set interrupt on it)
5. When **TMR0IF** is set, stop the timer, reset the flag (and if needed go to step 2)

FIGURE 10-2: TIMER0 BLOCK DIAGRAM IN 16-BIT MODE



Note: Upon RESET, Timer0 is enabled in 8-bit mode with clock input from T0CKI max. prescale.

CSE@UTA

Timer1

TMR1H								TMR1L							
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

- Timer1 is **16-bit** only
- **T1CON** is the control register
- **TMR1IF** is the interrupt flag in the **PIR1** register
- Prescaler does not support divisions above 1:8
- Timer1 has 2 <u>external</u> clock sources and 1 regular internal
  - Clock fed into T1CK1 pin (RC0)
  - Crystal (typically 32-kHz) connected between the T1CKI and T1OSI PINS (RC0&RC1) – for saving power during sleep mode. Timer1 is not shut down allowing use a clock that can be used for waking up

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
IPR1	PSPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP
TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register							
TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register							
T1CON	RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON

Timer1 Control Register T1CON

CSE@UTA

ER 11-1: T1CON: TIMER1 CONTROL REGISTER

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
bit 7							bit 0

bit 7 **RD16:** 16-bit Read/Write Mode Enable bit
1 = Enables register Read/Write of Timer1 in one 16-bit operation
0 = Enables register Read/Write of Timer1 in two 8-bit operations

bit 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 bit
1 = Timer1 Oscillator is enabled
0 = Timer1 Oscillator is shut-off
The oscillator inverter and feedback resistor are turned off to eliminate power drain.

bit 2 **$\overline{\text{T1SYNC}}$:** Timer1 External Clock Input Synchronization Select 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/T13CKI (on the rising edge)
0 = Internal clock (Fosc/4)

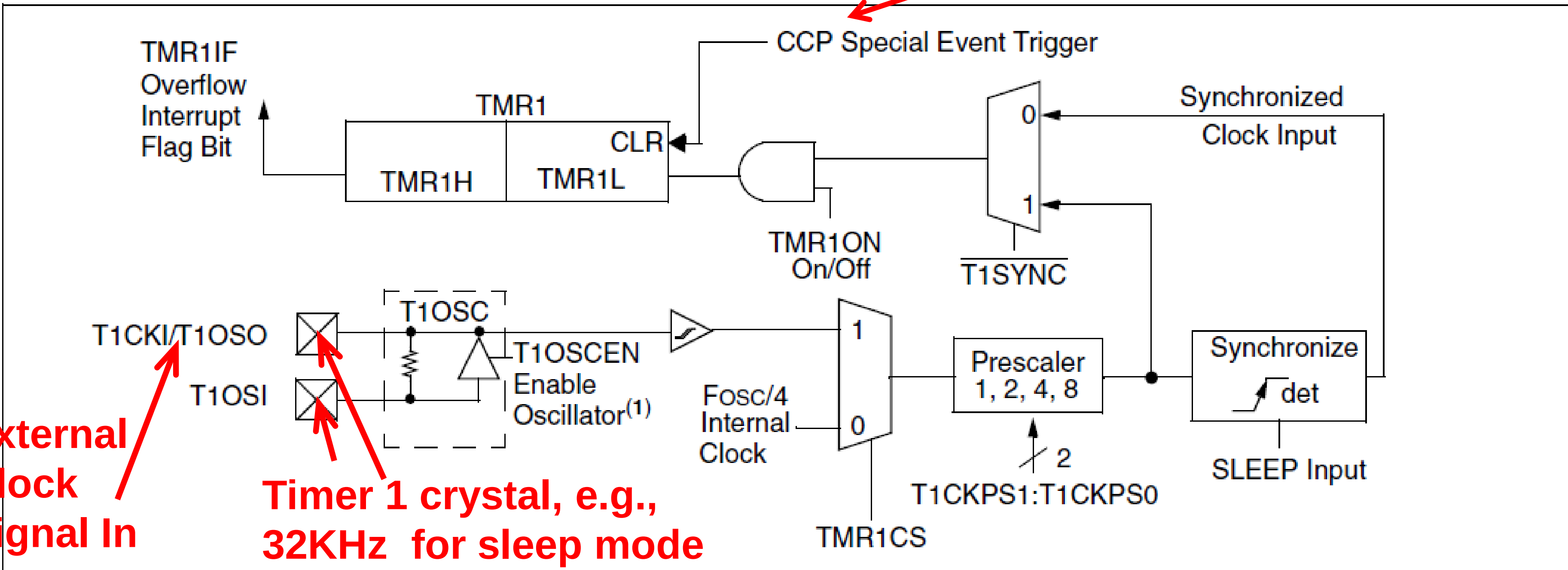
bit 0 **TMR1ON:** Timer1 On bit
1 = Enables Timer1
0 = Stops Timer1

- 16-bit mode only
- Smaller prescaler range
- Timer1 can be used as
  1. timer
  2. synchronous counter (T1SYNC)
  3. asynchronous counter

Timer1 Block Diagram

Used with
CCP option

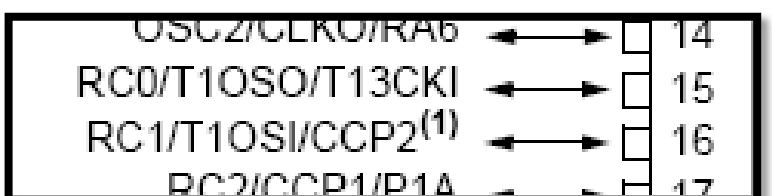
FIGURE 11-1: TIMER1 BLOCK DIAGRAM



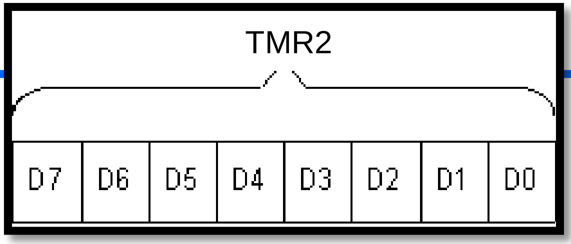
External
Clock
Signal In

Timer 1 crystal, e.g.,
32KHz for sleep mode

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



Timer2



- Timer2 is an **8-bit** only
- **T2CON** is the control register
- **TMR2IF** is the interrupt flag in the **PIR1** register
- Timer2 has a period register **PR2**; Timer2 can be set to count only to **PR2** and set **TMR2IF** then
- Clock source is only Fosc/4 (Timer2 cannot be a counter)
- Has both a **prescaler** and a **postscaler**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
IPR1	PSPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP
TMR2	Timer2 Module Register							
T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
PR2	Timer2 Period Register							

Timer2 Control Register T2CON

REGISTER 12-1: T2CON: TIMER2 CONTROL REGISTER

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
•
•
•
1111 = 1:16 Postscale

bit 2 **TMR2ON**: Timer2 On bit

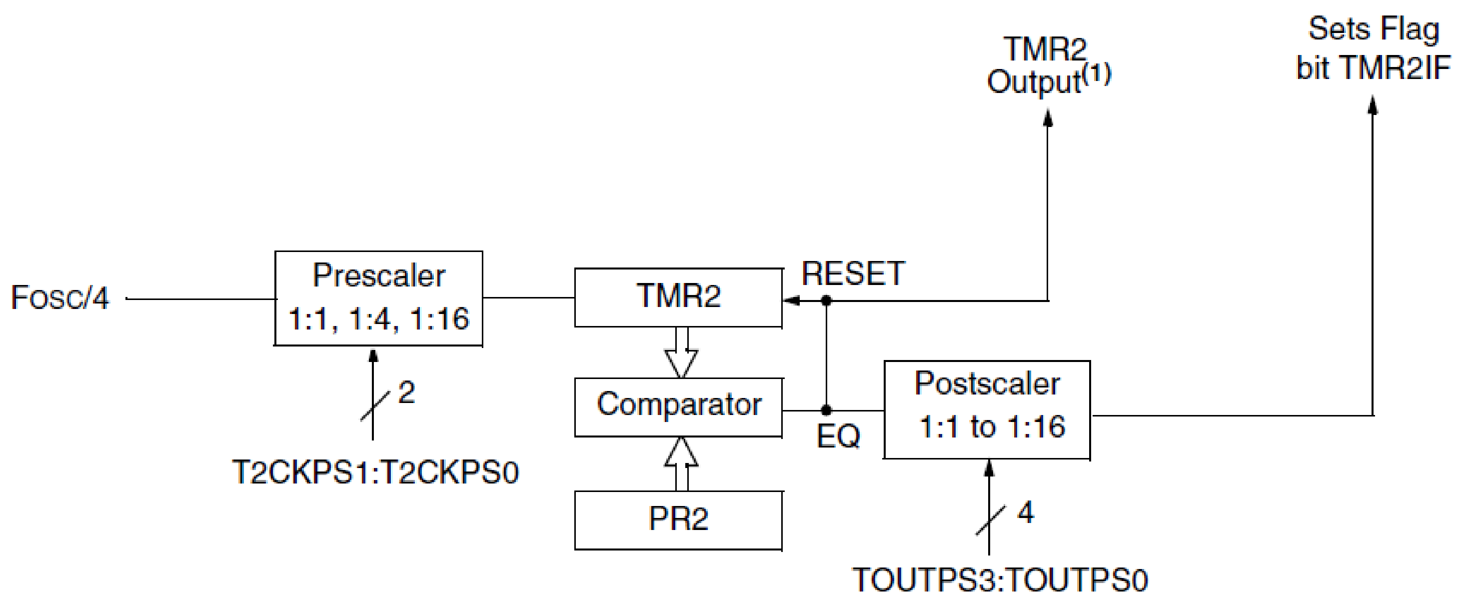
1 = Timer2 is on
0 = Timer2 is off

bit 1-0 **T2CKPS1:T2CKPS0**: Timer2 Clock Prescale Select bits

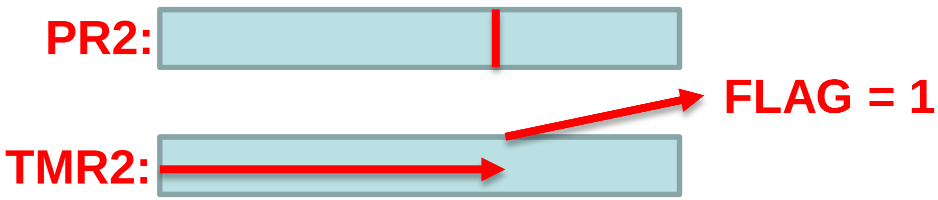
00 = Prescaler is 1
01 = Prescaler is 4
1x = Prescaler is 16

Timer2 Block Diagram

FIGURE 12-1: TIMER2 BLOCK DIAGRAM



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



Timer3

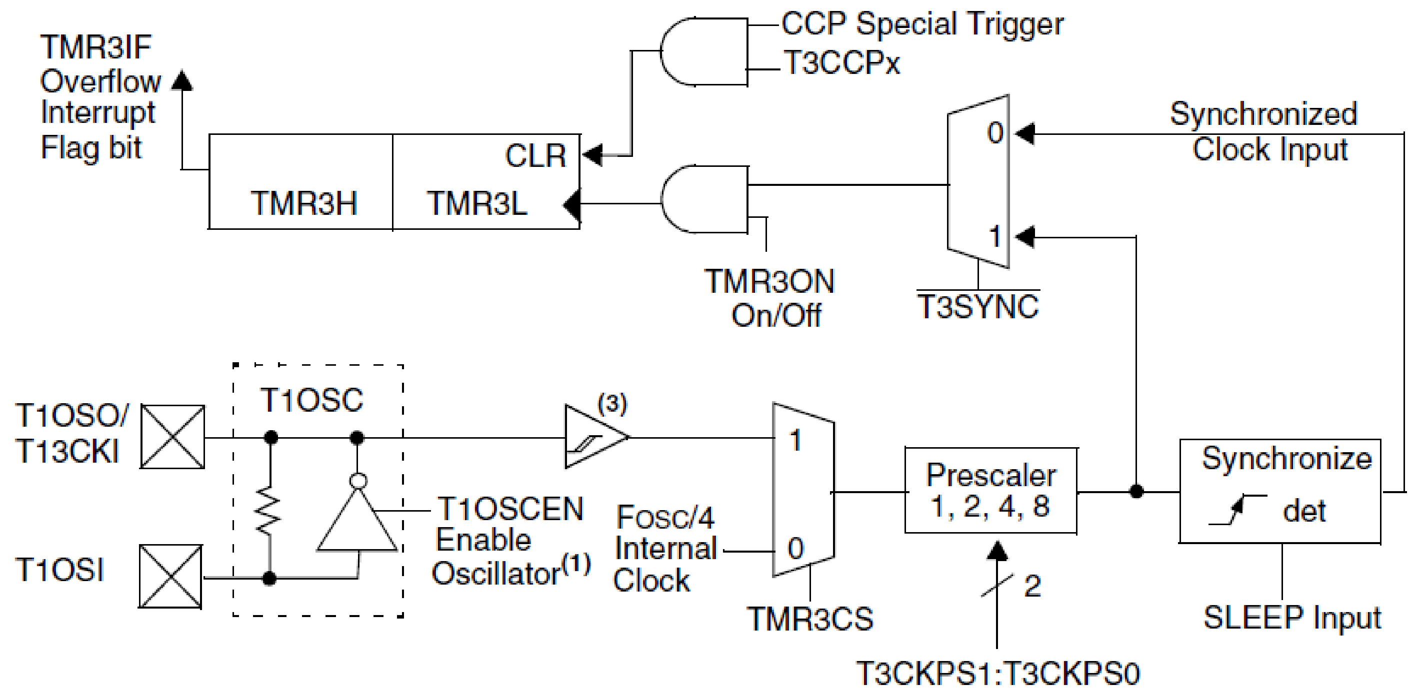
TMR3H								TMR3L							
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

- Timer3 is **16-bit** only
- **T3CON** is the control register
- **TMR3IF** is the interrupt flag in the **PIR2** register
- Can work with CCP peripheral (later)
- Timer3 has 2 <u>external</u> clock sources and 1 regular internal
  - Same external source(s) as timer1
- Can be used as timer, ascynchronous, or synchronous counter

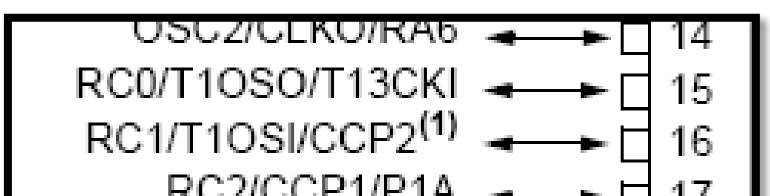
Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/ GIEH	PEIE/ GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
PIR2	—	—	—	EEIF	BCLIF	LVDIF	TMR3IF	CCP2IF
PIE2	—	—	—	EEIE	BCLIE	LVDIE	TMR3IE	CCP2IE
IPR2	—	—	—	EEIP	BCLIP	LVDIP	TMR3IP	CCP2IP
TMR3L	Holding Register for the Least Significant Byte of the 16-bit TMR3 Register							
TMR3H	Holding Register for the Most Significant Byte of the 16-bit TMR3 Register							
T1CON	RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
T3CON	RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON

Timer3 Block Diagram

FIGURE 13-1: TIMER3 BLOCK DIAGRAM



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



Timer3 Control Register T3CON

13-1: T3CON: TIMER3 CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON
bit 7							bit 0

bit 7 **RD16:** 16-bit Read/Write Mode Enable bit
1 = Enables register Read/Write of Timer3 in one 16-bit operation
0 = Enables register Read/Write of Timer3 in two 8-bit operations

bit 6-3 **T3CCP2:T3CCP1:** Timer3 and Timer1 to CCPx Enable bits
1x = Timer3 is the clock source for compare/capture CCP modules
01 = Timer3 is the clock source for compare/capture of CCP2,
Timer1 is the clock source for compare/capture of CCP1
00 = Timer1 is the clock source for compare/capture CCP modules

bit 5-4 **T3CKPS1:T3CKPS0**: Timer3 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 2 **$\overline{\text{T3SYNC}}$:** Timer3 External Clock Input Synchronization Control bit
(Not usable if the system clock comes from Timer1/Timer3)
When TMR3CS = 1:
1 = Do not synchronize external clock input
0 = Synchronize external clock input
When TMR3CS = 0:
This bit is ignored. Timer3 uses the internal clock when TMR3CS = 0.

bit 1 **TMR3CS:** Timer3 Clock Source Select bit
1 = External clock input from Timer1 oscillator or T1CKI
(on the rising edge after the first falling edge)
0 = Internal clock (Fosc/4)

bit 0 **TMR3ON:** Timer3 On bit
1 = Enables Timer3
0 = Stops Timer3

Timer0 Interrupt Example

1. In **T0CON**…
    1. Select **16-bit** mode
    2. Select **internal** or **external clock source**
    3. Allow **prescaler option** if desired
    4. Select desired **prescaler**
2. Load **TMR0H** and **TMR0L** with preloads (**load HIGH first!!**)
    – Ex: 18,661 dec = 0x48E5 → TMR0H = 0x48 and TMR0L = 0xE5
3. In **INTCON…**
    1. Enable the **TMR0IE** interrupt bit
    2. Enable the **PEIE** peripheral interrupt bit
    3. Enable the **GIE** global interrupt bit
4. Start timer (**T0CONbits.TMR0ON = 1**)
5. Monitor **TMR0IF** (if only polling)
6. When overflow occurs (1.2s has passed) **TMR0IF** is set to 1
7. In the ISR…
    1. **Identify** the interrupt source
    2. **Stop** the timer (disable Timer0)
    3. **Reset the flag** (if needed go to step 2 of writing preload values)

Using PIC18 Timers for CCP (Capture, Compare, and PWM)

- **Next lecture: CCP and motor/servo driving**
- Timer0 is usually just for generic timing
- Timers 1 and 3 can be used for capture and compare features
  - T3CON is used to chose the timer for CCP
- Timer2 is used for PWM
  - Note: These rules do not always apply, have to check the specific PIC18 datasheet

Using PIC18 Timers for CCP

(Capture, Compare, and PWM)

CCP

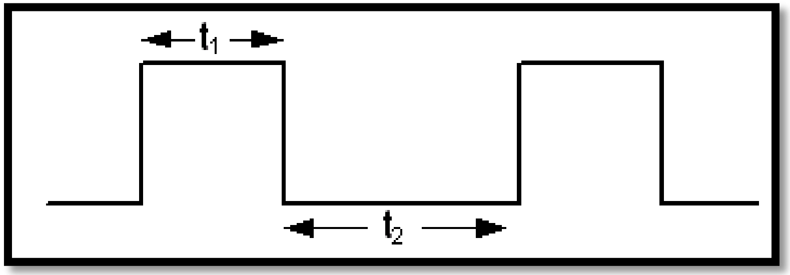
- **Compare** (input)
  - Count outside events (incoming to the PIC's pins)
  - When X have occurred → do something
- **Capture** (input)
  - Measure an unknown signal's frequency (period) or PWM Duty Cycle
- **PWM** (output)
  - Send a precise signal out of the PIC

PWM Basics (Pulse Width Modulation)

- Digital signals have two distinct levels: **high** and **low**
- These levels are usually represented by a voltage
  - e.g., in PIC low is 0V and high is VCC (5V)
- A temporal digital signal changes with time from low to high and back
- Thus we can describe temporal digital signals with a series of values representing the **time for which they stay in one state**
- Periodic temporal digital signals have a distinct frequency
  - the inverse of the time between two consecutive rising edges

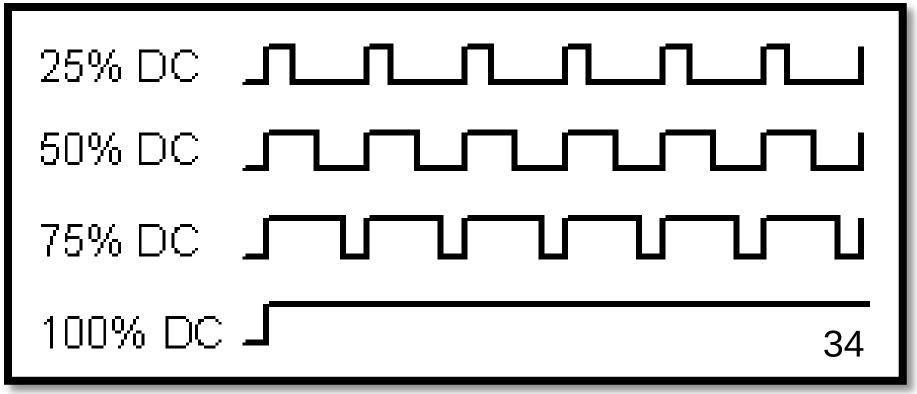
PWM Basics (cont'd)

- If $t_1 + t_2$ remain constant → frequency remains constant

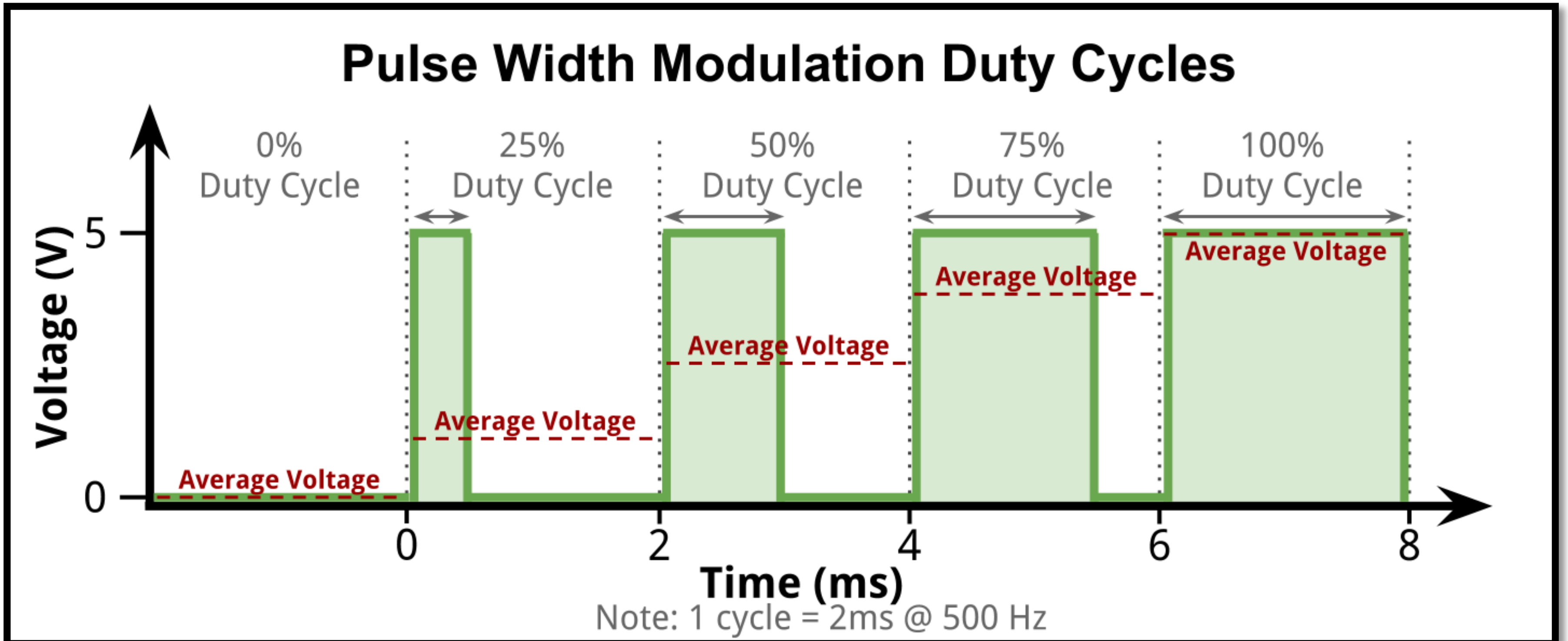


- Such periodic signals can still have varying times they spend in high vs. low state
- **PWM Duty Cycle** is the portion of the pulse that stays **HIGH** relative to the entire period

$$DC[\%] = 100 * \frac{t_1}{t_1 + t_2}$$



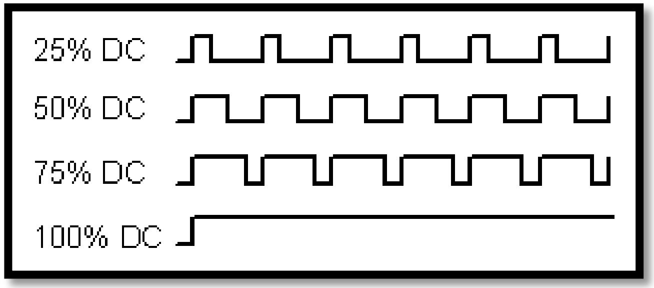
PWM Basics (cont'd)



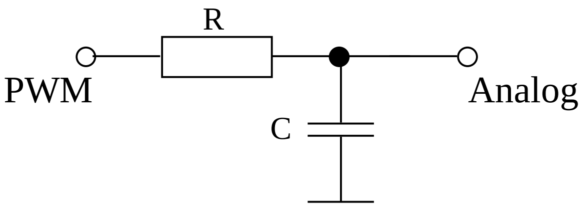
Source: http://www.hho4free.com/pulse_width_modulator__pwm.html

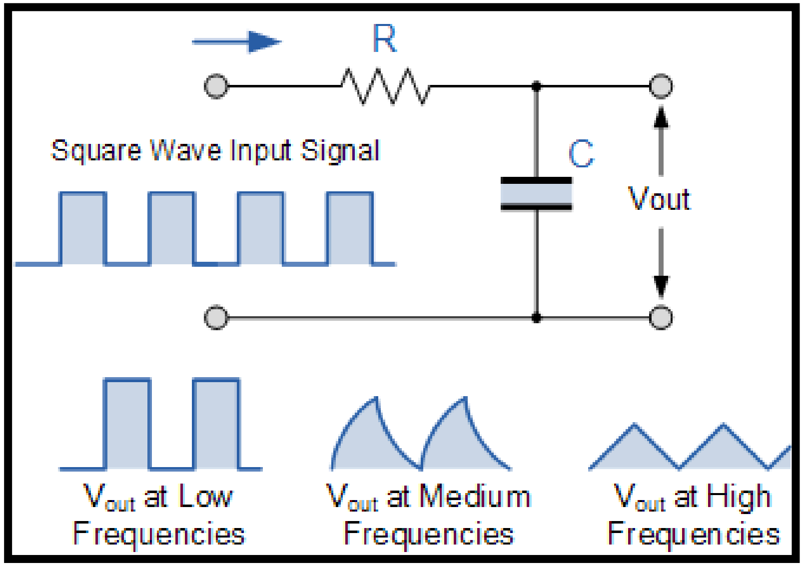
PWM Basics (cont'd)

- There are various sensors that provide their output as PWM signals, where the DC corresponds to the reading
- There are various actuators that work well with a PWM input



- An appropriate RC filter (Integrator) can make an analog signal out of a PWM digital signal





Source: http://www.electronics-tutorials.ws/

PIC18's CCP Modules

- PIC18s have **0 – 5 CCP** modules on-board (CCPx) with 3 modes
- **Capture**
  - can use an external **input** to copy timer values into a 16-bit register
  - provides the capability of **measuring** the period of a pulse
- **Compare**
  - enables the **counter** value of timers to be compared to a 16-bit register
  - if equal, then perform an action
- **PWM**
  - can be used as a quasi-analog **output** (timed digital output with duty cycle setting)
- These are great for driving motors, reading encoders, IR comm.
- For DC motor control some of the CCPs have been enhanced and are called **ECCP**
- The PIC18F452 has 2 CCP Modules: **CCP1 & CCP2**
  - can be used at the same time but only 1 mode per CCP at a time

Timers and CCP Association

TABLE 14-1: CCP MODE - TIMER RESOURCE

CCP Mode	Timer Resource
Capture	Timer1 or Timer3
Compare	Timer1 or Timer3
PWM	Timer2

T3CON: TIMER3 CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON
bit 7							bit 0

These rules do not always apply – have to check the specific PIC18 datasheet

T3CON

REGISTER 13-1: T3CON: TIMER3 CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON
bit 7							bit 0

bit 7 **RD16:** 16-bit Read/Write Mode Enable bit

1 = Enables register Read/Write of Timer3 in one 16-bit operation
0 = Enables register Read/Write of Timer3 in two 8-bit operations

bit 6-3 **T3CCP2:T3CCP1:** Timer3 and Timer1 to CCPx Enable bits

1x = Timer3 is the clock source for compare/capture CCP modules
01 = Timer3 is the clock source for compare/capture of CCP2, Timer1 is the clock source for compare/capture of CCP1
00 = Timer1 is the clock source for compare/capture CCP modules

bit 5-4 **T3CKPS1:T3CKPS0**: Timer3 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 2 **$\overline{\text{T3SYNC}}$:** Timer3 External Clock Input Synchronization Control bit
(Not usable if the system clock comes from Timer1/Timer3)

<u>When TMR3CS = 1:</u>
1 = Do not synchronize external clock input
0 = Synchronize external clock input

<u>When TMR3CS = 0:</u>
This bit is ignored. Timer3 uses the internal clock when TMR3CS = 0.

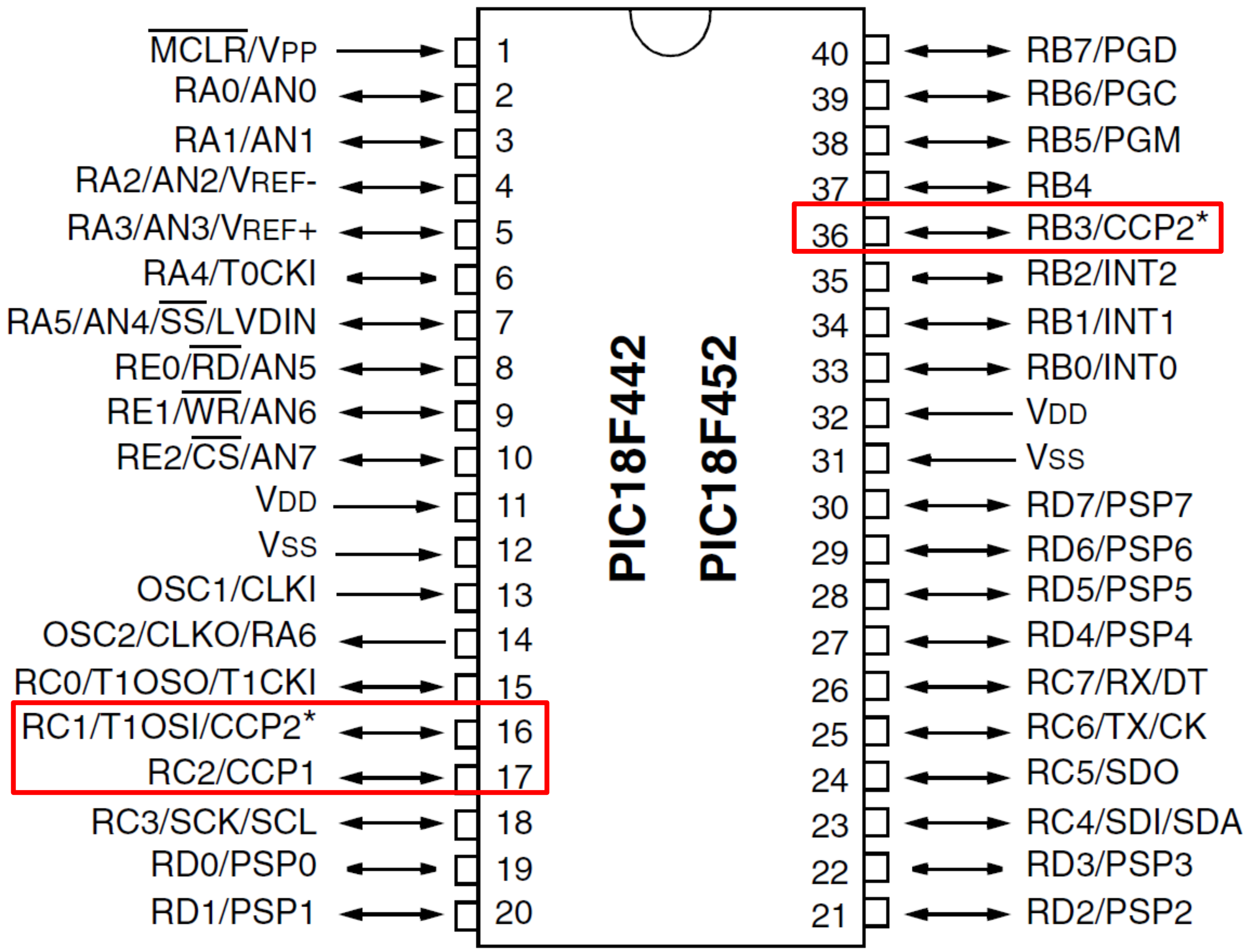
bit 1 **TMR3CS:** Timer3 Clock Source Select bit

1 = External clock input from Timer1 oscillator or T1CKI (on the rising edge after the first falling edge)
0 = Internal clock (FOSC/4)

bit 0 **TMR3ON:** Timer3 On bit

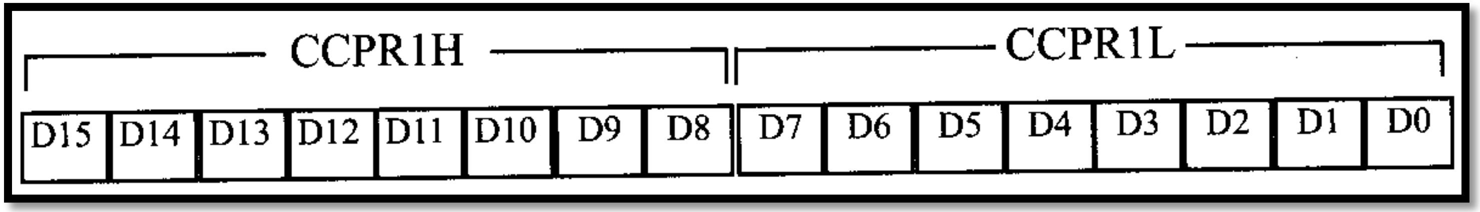
1 = Enables Timer3
0 = Stops Timer3

CCPx Pins

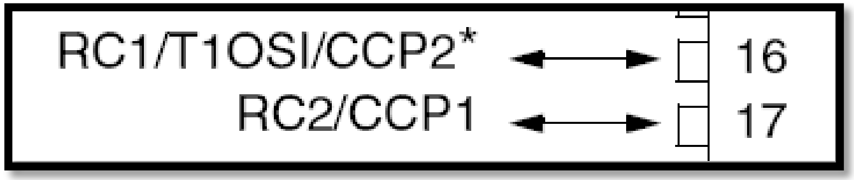

PIC18F442
PIC18F452
MCLR/VPP 1
RA0/AN0 2
RA1/AN1 3
RA2/AN2/VREF- 4
RA3/AN3/VREF+ 5
RA4/T0CKI 6
RA5/AN4/SS/LVDIN 7
RE0/RD/AN5 8
RE1/WR/AN6 9
RE2/CS/AN7 10
VDD 11
VSS 12
OSC1/CLKI 13
OSC2/CLKO/RA6 14
RC0/T1OSO/T1CKI 15
RC1/T1OSI/CCP2* 16
RC2/CCP1 17
RC3/SCK/SCL 18
RD0/PSP0 19
RD1/PSP1 20
40 RB7/PGD
39 RB6/PGC
38 RB5/PGM
37 RB4
36 RB3/CCP2*
35 RB2/INT2
34 RB1/INT1
33 RB0/INT0
32 VDD
31 VSS
30 RD7/PSP7
29 RD6/PSP6
28 RD5/PSP5
27 RD4/PSP4
26 RC7/RX/DT
25 RC6/TX/CK
24 RC5/SDO
23 RC4/SDI/SDA
22 RD3/PSP3
21 RD2/PSP2

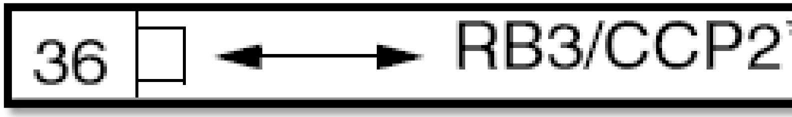
CCP Module Basics

- Each CCP module has 3 associated registers
  - **CCPxCON** controlling the modes
  - **CCPxL** and **CCPxH** as a 16-bit compare/capture/PWM duty cycle register



- Each CCP module has a pin associated with it (input or output)





CCP 1 & 2 Module Control CCPxCON

REGISTER 14-1: CCP1CON REGISTER/CCP2CON REGISTER

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	DCxB1	DCxB0	CCPxM3	CCPxM2	CCPxM1	CCPxM0
bit 7							bit 0

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

bit 5-4 **DCxB1:DCxB0**: PWM Duty Cycle bit1 and bit0

Capture mode:
Unused

Compare mode:
Unused

PWM mode:
These bits are the two LSbs (bit1 and bit0) of the 10-bit PWM duty cycle. The upper eight bits (DCx9:DCx2) of the duty cycle are found in CCPRxL.

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

0000 = Capture/Compare/PWM disabled (resets CCPx module)
0001 = Reserved
0010 = Compare mode, toggle output on match (CCPxIF bit is set)
0011 = Reserved
0100 = Capture mode, every falling edge
0101 = Capture mode, every rising edge
prescalers
0110 = Capture mode, every 4th rising edge
0111 = Capture mode, every 16th rising edge
1000 = Compare mode,
Initialize CCP pin Low, on compare match force CCP pin High (CCPIF bit is set)
1001 = Compare mode,
Initialize CCP pin High, on compare match force CCP pin Low (CCPIF bit is set)
1010 = Compare mode,
Generate software interrupt on compare match (CCPIF bit is set, CCP pin is unaffected)
1011 = Compare mode,
Trigger special event (CCPIF bit is set)
11xx = PWM mode

Relevant Registers

TABLE 14-3: REGISTERS ASSOCIATED WITH CAPTURE, COMPARE, TIMER1 AND TIMER3

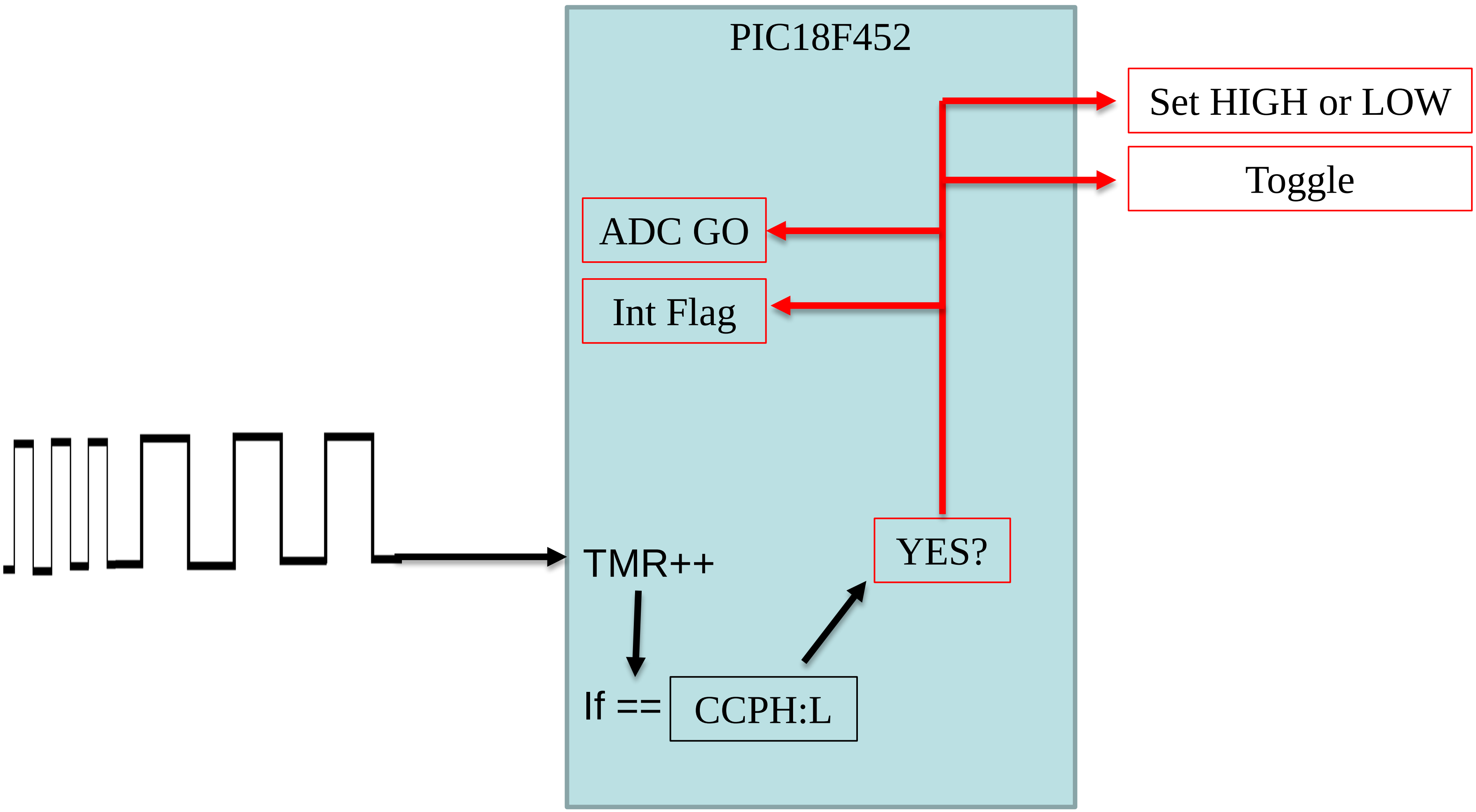
Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on All Other RESETS
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF[1]	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE[1]	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP[1]	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
TRISC	PORTC Data Direction Register								1111 1111	1111 1111
TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
T1CON	RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	0-00 0000	u-uu uuuu
CCPR1L	Capture/Compare/PWM Register1 (LSB)								xxxx xxxx	uuuu uuuu
CCPR1H	Capture/Compare/PWM Register1 (MSB)								xxxx xxxx	uuuu uuuu
CCP1CON	—	—	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	--00 0000
CCPR2L	Capture/Compare/PWM Register2 (LSB)								xxxx xxxx	uuuu uuuu
CCPR2H	Capture/Compare/PWM Register2 (MSB)								xxxx xxxx	uuuu uuuu
CCP2CON	—	—	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	--00 0000
PIR2	—	—	—	EEIE	BCLIF	LVDIF	TMR3IF	CCP2IF	---0 0000	---0 0000
PIE2	—	—	—	EEIF	BCLIE	LVDIE	TMR3IE	CCP2IE	---0 0000	---0 0000
IPR2	—	—	—	EEIP	BCLIP	LVDIP	TMR3IP	CCP2IP	---1 1111	---1 1111
TMR3L	Holding Register for the Least Significant Byte of the 16-bit TMR3 Register								xxxx xxxx	uuuu uuuu
TMR3H	Holding Register for the Most Significant Byte of the 16-bit TMR3 Register								xxxx xxxx	uuuu uuuu
T3CON	RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON	0000 0000	uuuu uuuu

Compare Mode

~ ***IF(CCP == TMR1/3) THEN…***

- The CCPRxH:CCPRxL is loaded by the user
- If Timer1 TMR1H:TMR1L (or Timer3 – T3CON) count becomes equal to the above set value then the Compare module can:
  1. Drive the CCPx pin high (CCPx config'd as out)
  2. Drive the CCPx pin low (CCPx config'd as out)
  3. Toggle the CCPx pin (CCPx config'd as out)
  4. Trigger a CCPxIF interrupt and clear the timer
  5. CCP2 can be used to kick off the A/D converter

Compare Mode

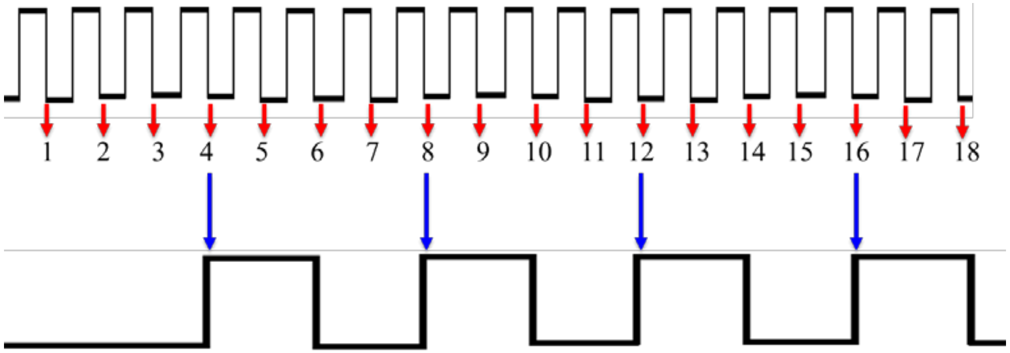


Compare Mode Programming

0. Set up CCP interrupt if needed
1. Initialize CCPxCON for compare
2. Pick timer source (T3CON)
3. Initialize the CCPRxH:CCPRxL 16-bit value
4. Make sure CCPx pin is **output** if used
   – setting appropriate TRISbits
5. Initialize Timer1 (or Timer3)
6. Start Timer1 (or Timer3)
7. Poll CCPxIF flag or make sure interrupt is handled

Capture Mode

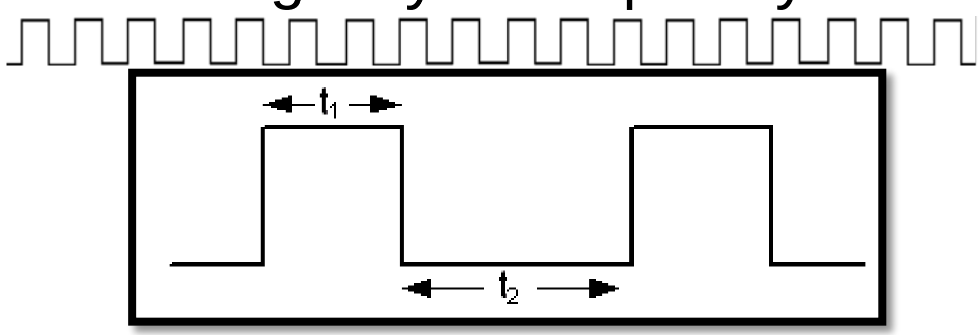
- The CCPx pin is set as **input** (set TRISbits)
- When an external event triggers the CCPx pin, then the TMR1H:TMR1L (or Timer3) values will be loaded into CCPRxH:CCPRxL
- Four options for CCPx pin triggering:
  - Every falling edge
  - Every rising edge
  - Every 4th rising edge
  - Every 16th rising edge



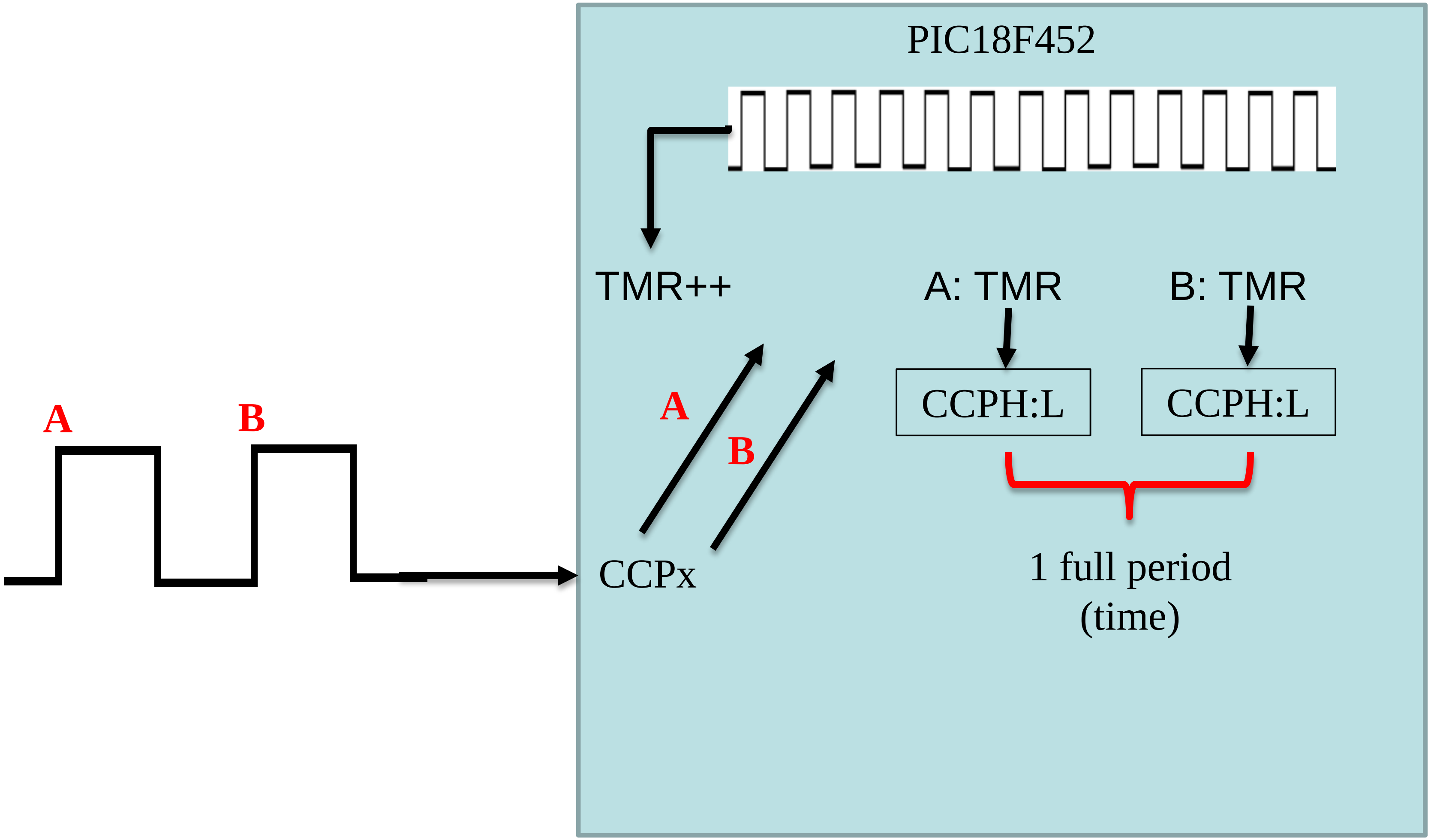
- Typical applications are measuring frequency or pulsewidth

Capture Mode Programming for Frequency Measurement

1. Initialize CCPxCON for capture
2. Make CCPx pin an **input** pin (TRISB/TRISC)
3. Pick timer source (T3CON)
4. On first rising edge, Timer1/3 is loaded into CCPRxH:CCPRxL
   – remember values
5. On next rising edge, Timer1/3 is loaded into CCPRxH:CCPRxL
   – subtract previous values from current values
6. You have now the **period of the signal** captured by timer ticks. Some basic math will give you frequency

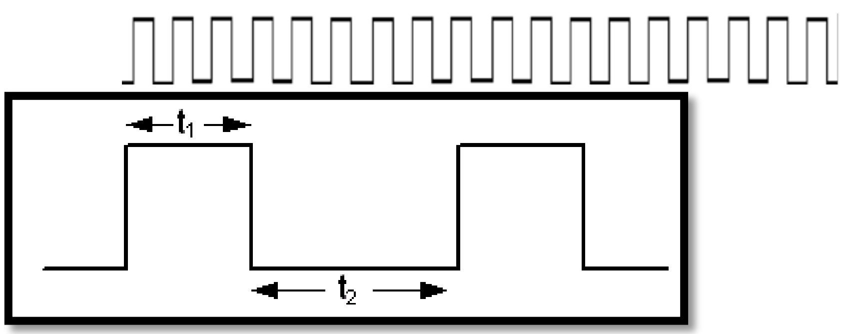


Capture Mode Programming for Frequency Measurement

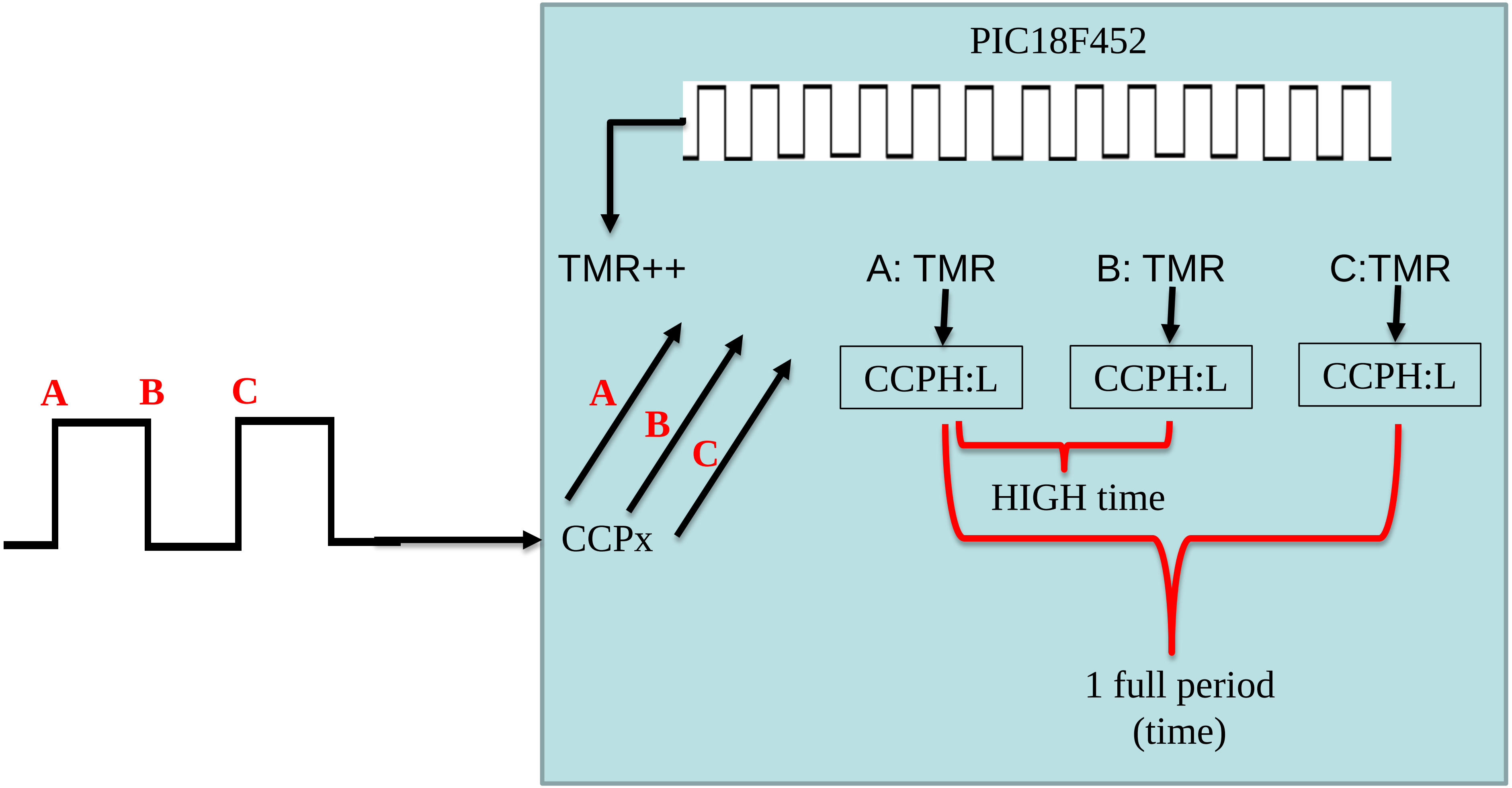


Capture Mode Programming for Measuring PWM Duty Cycle

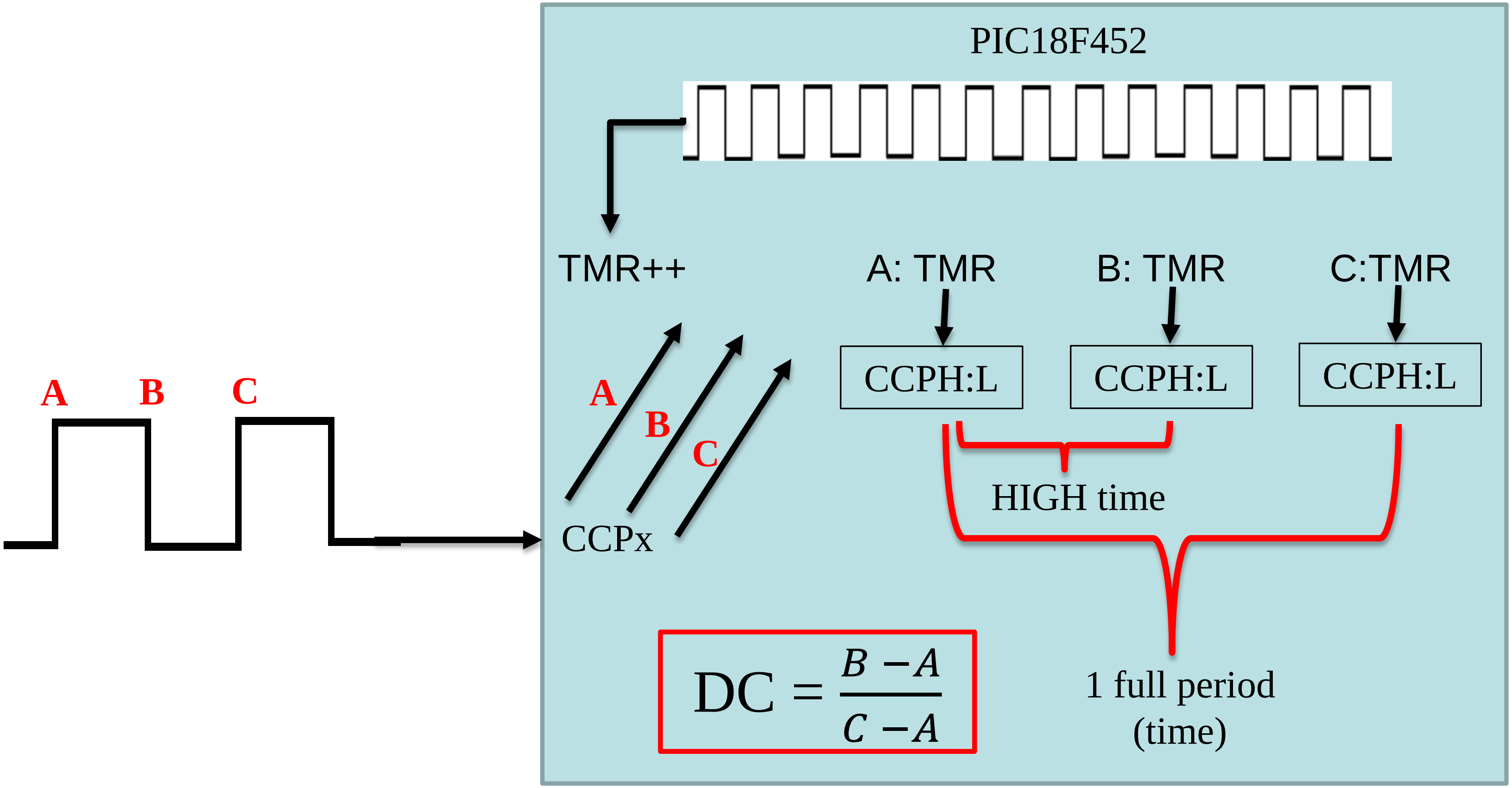
1. Initialize CCPxCON for capture
2. Make CCPx pin an input pin (TRISB/TRISC)
3. Pick timer source (T3CON)
4. On rising edge, Timer is started & mode set to falling edge
5. On falling edge the CCPRxH:CCPRxL should be saved, CCP should be set to rising edge
6. On rising edge CCPRxH:CCPRxL is saved
   – Now we have measurements for t_1 and t_2
7. DC can be calculated while new measurement is prepared



Capture Mode Programming for Measuring PWM Duty Cycle

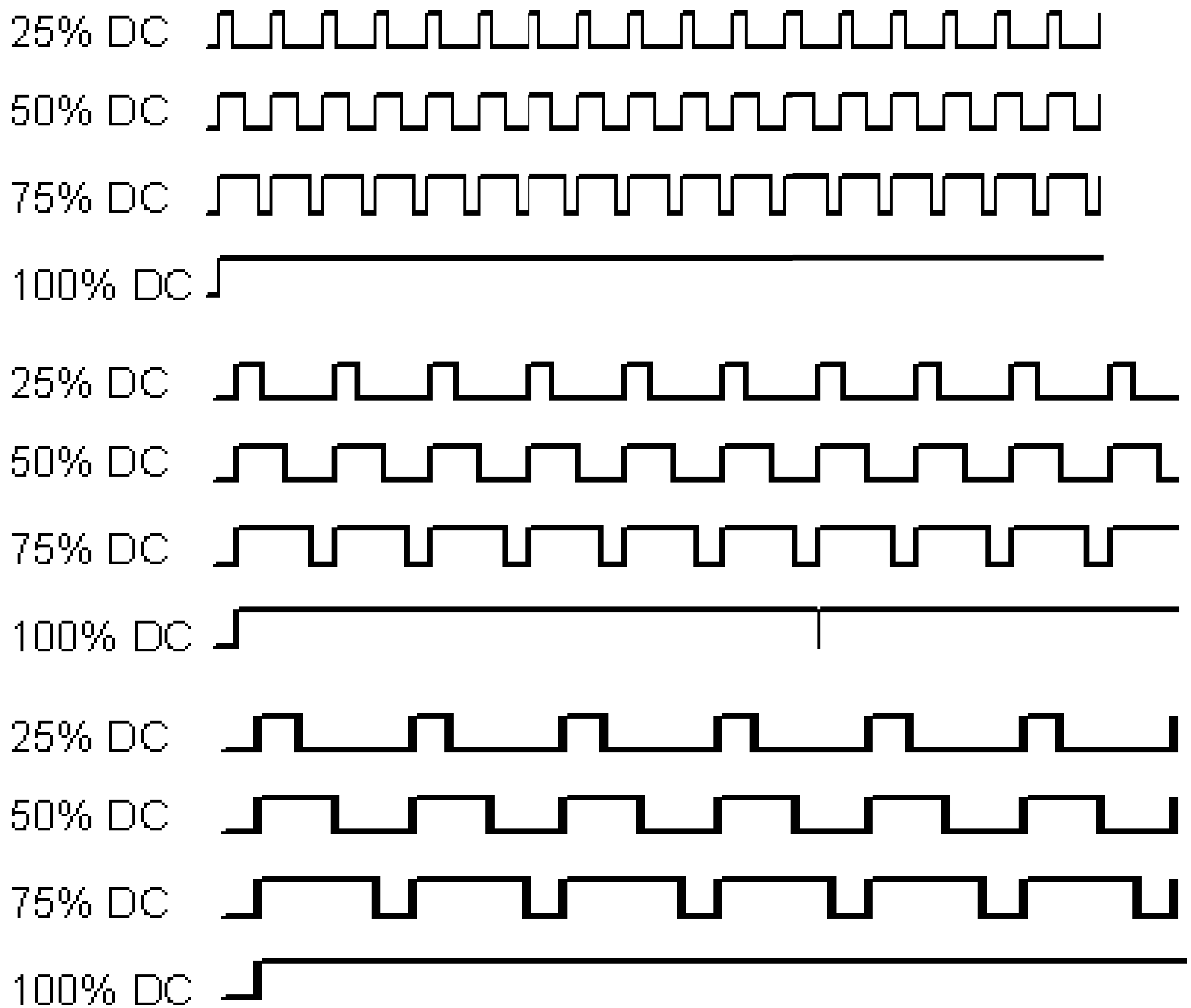


Capture Mode Programming for Measuring PWM Duty Cycle



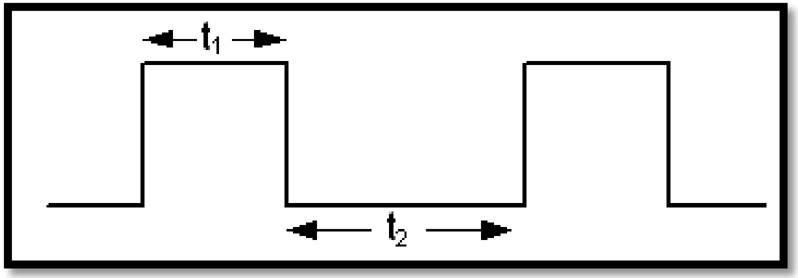
$$\boxed{\mathrm{DC} = \frac{B - A}{C - A}}$$

PWM Mode
(Generate Precise Output)



PWM Mode

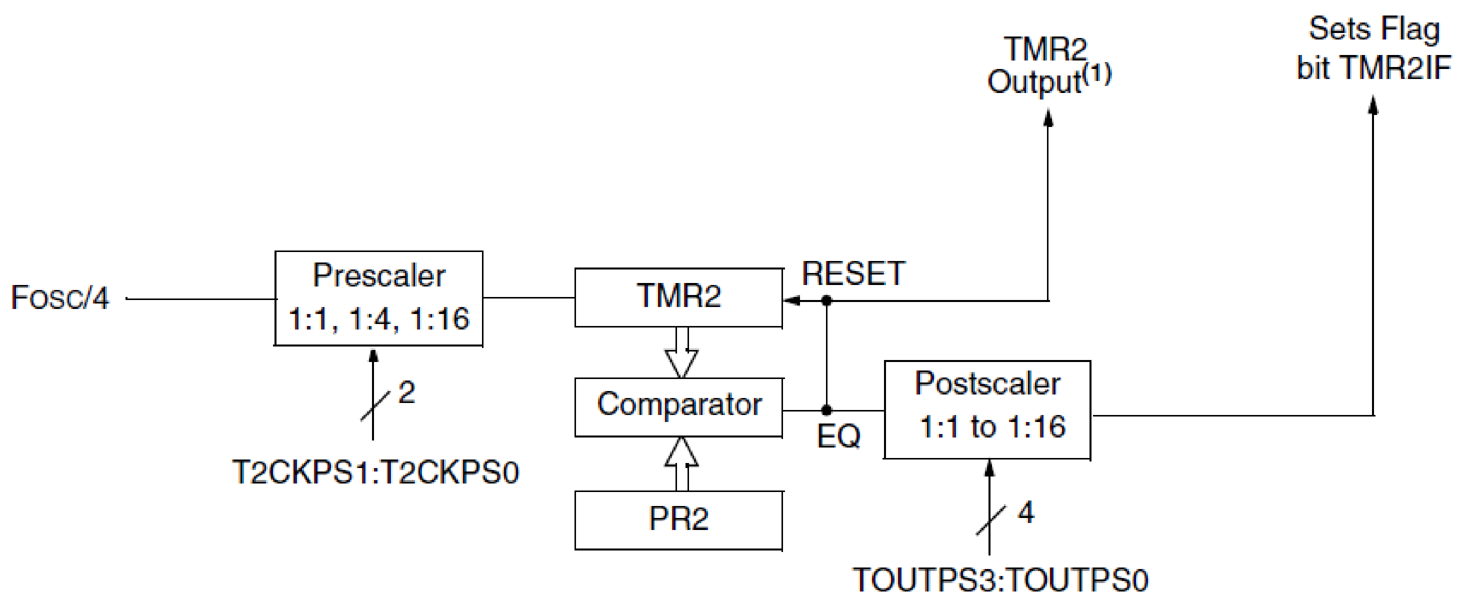
- PWM output can be created without tedious programming of the compare mode or timers
- ECCP’s PWM mode enables generating temporal digital signals of varying frequencies and varying DC
  - recall: width of the pulse indicates some measured quantity
- Recall, that the **PWM Duty Cycle** is the portion of the pulse at **HIGH** relative to the entire period
  - $DC[\%] = 100 * t_1/(t_1+t_2)$



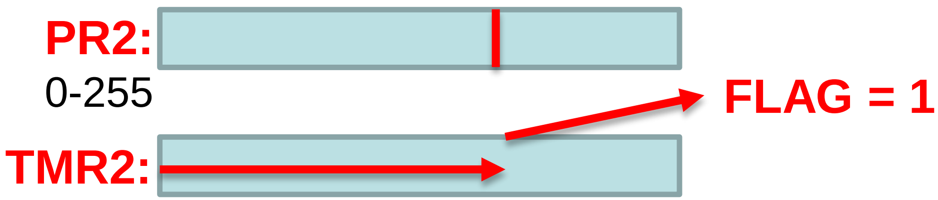
- For PWM, **Timer2** is used
- Recall, that Timer 2 has a period register **PR2**

Timer2 Block Diagram

FIGURE 12-1: TIMER2 BLOCK DIAGRAM



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



PWM
Specify Two Things

14.5.1 PWM PERIOD

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

Unit of time (ms, us, etc.)

$$\text{PWM period} = (PR2) + 1] \cdot 4 \cdot T_{OSC} \cdot (\text{TMR2 prescale value})$$

14.5.2 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:

Unit of time (ms, us, etc.)

$$\text{PWM duty cycle} = (\text{CCPR1L:CCP1CON}<5:4>) \cdot T_{OSC} \cdot (\text{TMR2 prescale value})$$

PWM Mode
Desired Period and Frequency

- $\mathbf{T_{pwm}}$ = desired PWM period (time, secs/cycle)
- $\mathbf{F_{pwm}}$ = desired PWM freq. (rate, cycles/sec)
  - $T_{pwm} = 1 / F_{pwm}$
- PR2: 0 – 255 (from TMR2)
- $T_{osc} = 1 / F_{osc}$

- $T_{pwm} = 4*N*(PR2+1) / F_{osc}$ or….
- $T_{pwm} = 4*N*(PR2+1) * T_{osc}$
  – where N is the prescaler of TMR2 (1, 4, 16)

PWM Mode
Desired Period and Frequency

- **Fastest Rate**
  - Min $T_{pwm} = 4*1*(0+1) * T_{osc} = 4\ T_{osc}$
  - Max $F_{pwm} = 1 / T_{pwm} = F_{osc} / 4$

- **Slowest Rate**
  - Max $T_{pwm} = 4*16*(255+1) * T_{osc} = 16{,}384\ T_{osc}$
  - Min $F_{pwm} = 1 / T_{pwm} = F_{osc} / 16{,}384$

PWM Mode Ex. Desired Period and Frequency

Find the PR2 value and the prescaler needed to get the following PWM frequencies. Assume XTAL = 20 MHz.
(a) 1.22 kHz, (b) 4.88 kHz, (c) 78.125 kHz

Solution:

(a) PR2 value = $[(20\text{ MHz} / (4 \times 1.22\text{ kHz})] - 1 = 4{,}097$, which is larger than 255, the maximum value allowed for the PR2. Now choosing the prescaler of 16 we get
PR2 value = $[(20\text{ MHz} / (4 \times 1.22\text{ kHz} \times 16)] - 1 = 255$

(b) PR2 value = $[(20\text{ MHz} / (4 \times 4.88\text{ kHz})] - 1 = 1{,}023$, which is larger than 255, the maximum value allowed for the PR2. Now choosing the prescaler of 4 we get
PR2 value = $[(20\text{ MHz} / (4 \times 4.88\text{ kHz} \times 4)] - 1 = 255$

(c) PR2 value = $[(20\text{ MHz} / (4 \times 78.125\text{ kHz})] - 1 = 63$

PWM Mode Ex.
Desired Period and Frequency

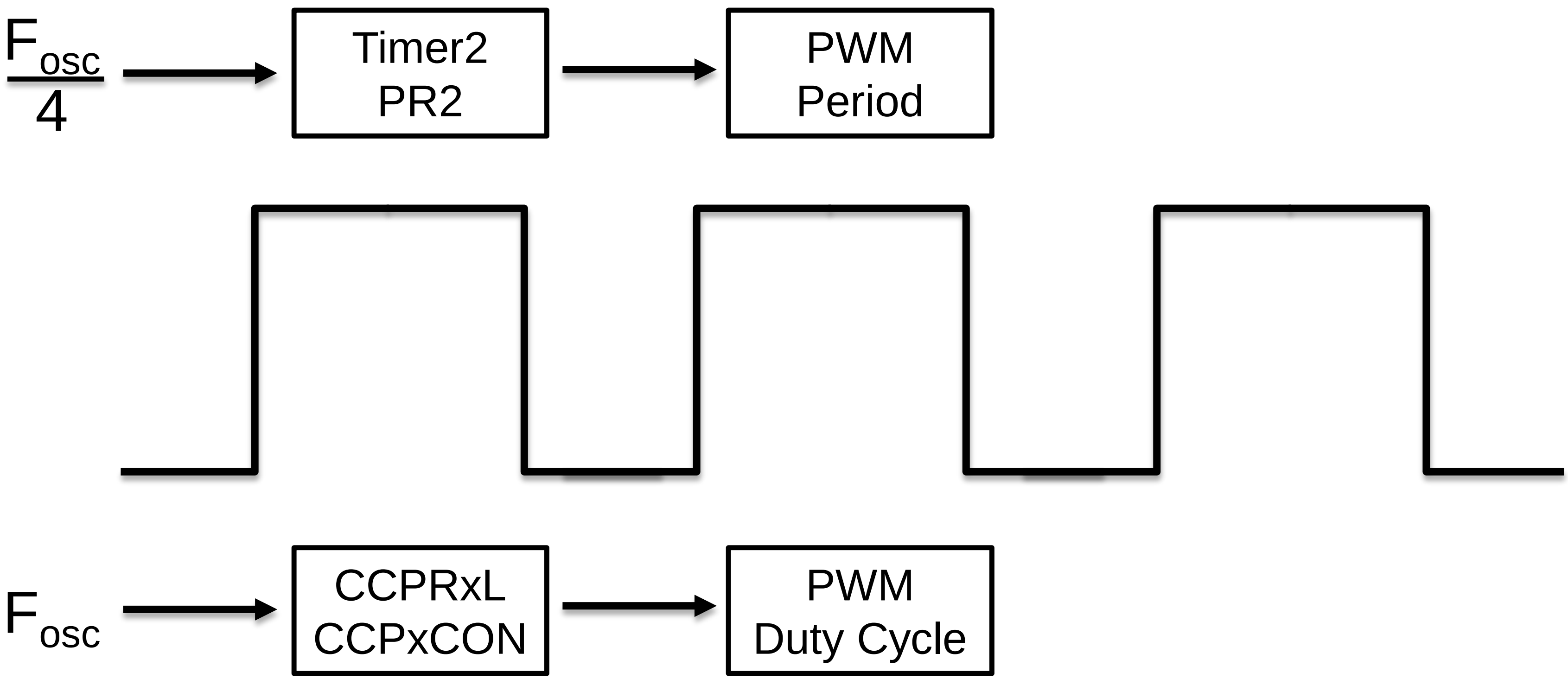
Find the minimum and maximum Fpwm frequency allowed for XTAL = 10 MHz. State the PR2 and prescaler values for the minimum and maximum Fpwm.

Solution:

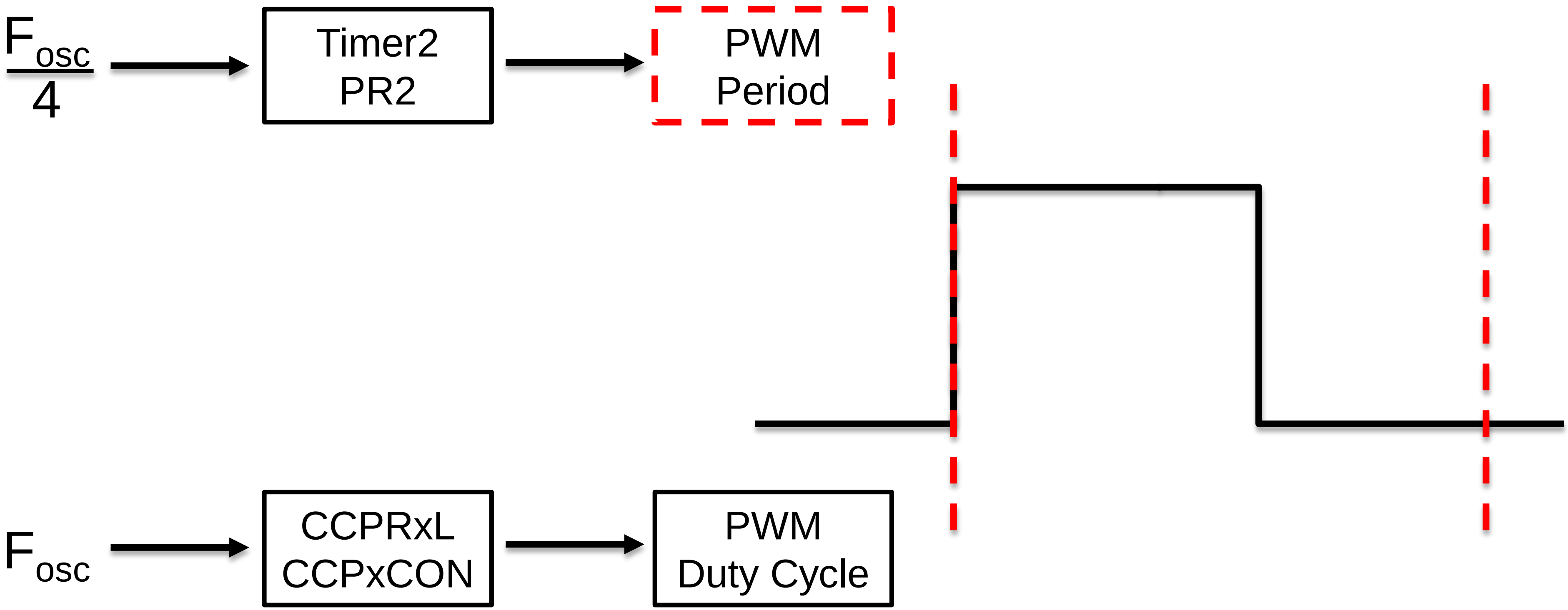
We get the minimum Fpwm by making PR2 = 255 and prescaler = 16, which gives us 10 MHz / (4 × 16 × 256) = 610 Hz.

We get the maximum Fpwm by making PR2 = 1 and prescaler = 1, which gives us 10 MHz / (4 × 1 × 1) = 2.5 MHz.

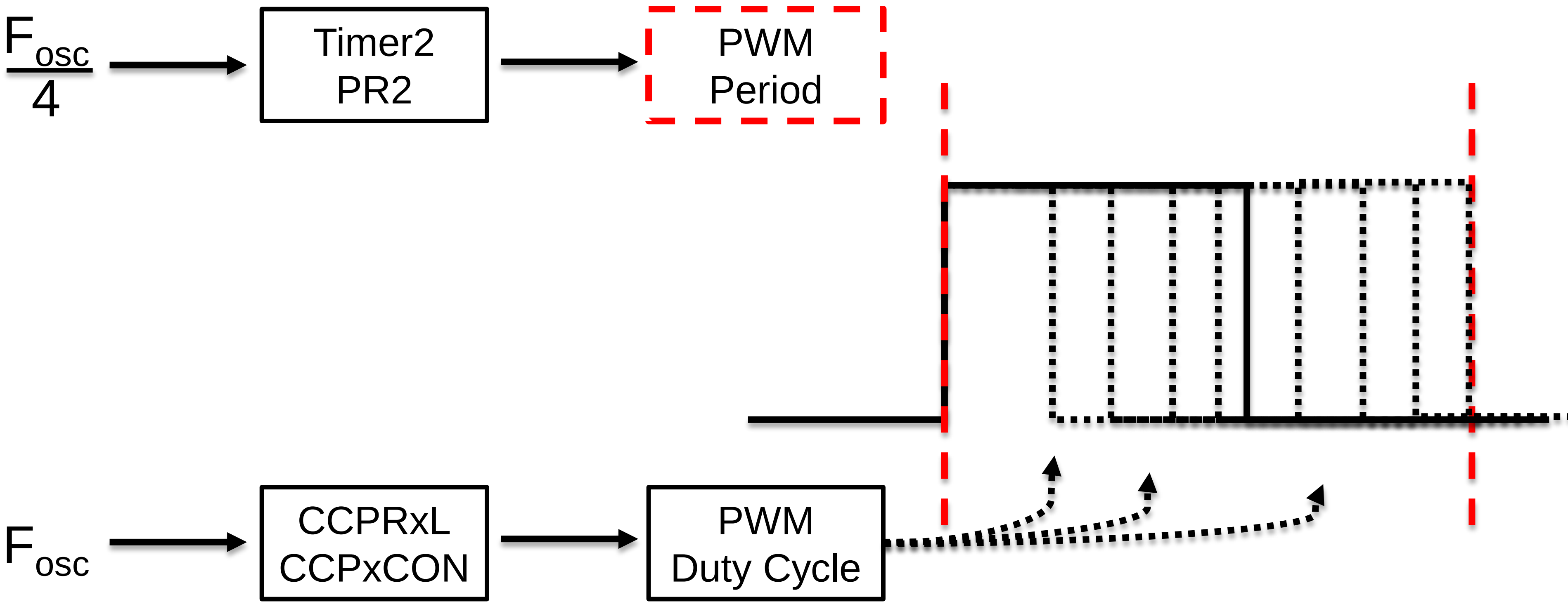
PWM Period and Duty Cycle



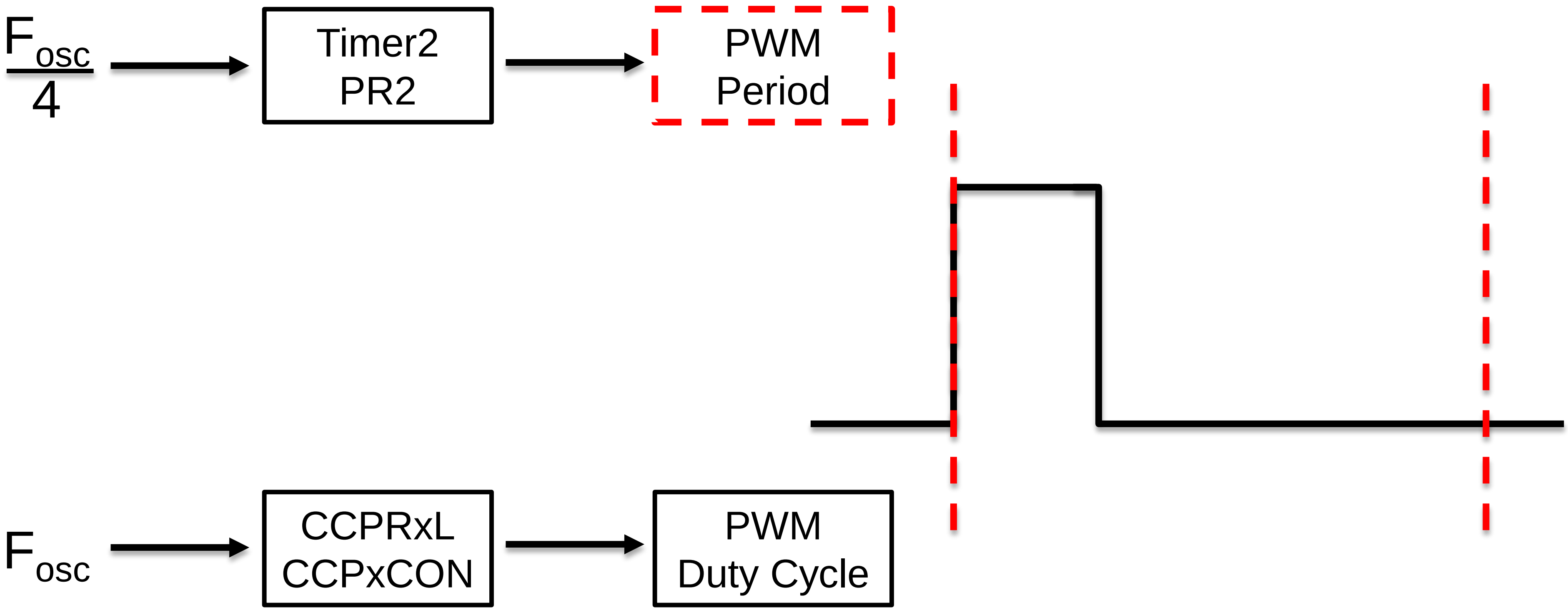
PWM Period and Duty Cycle



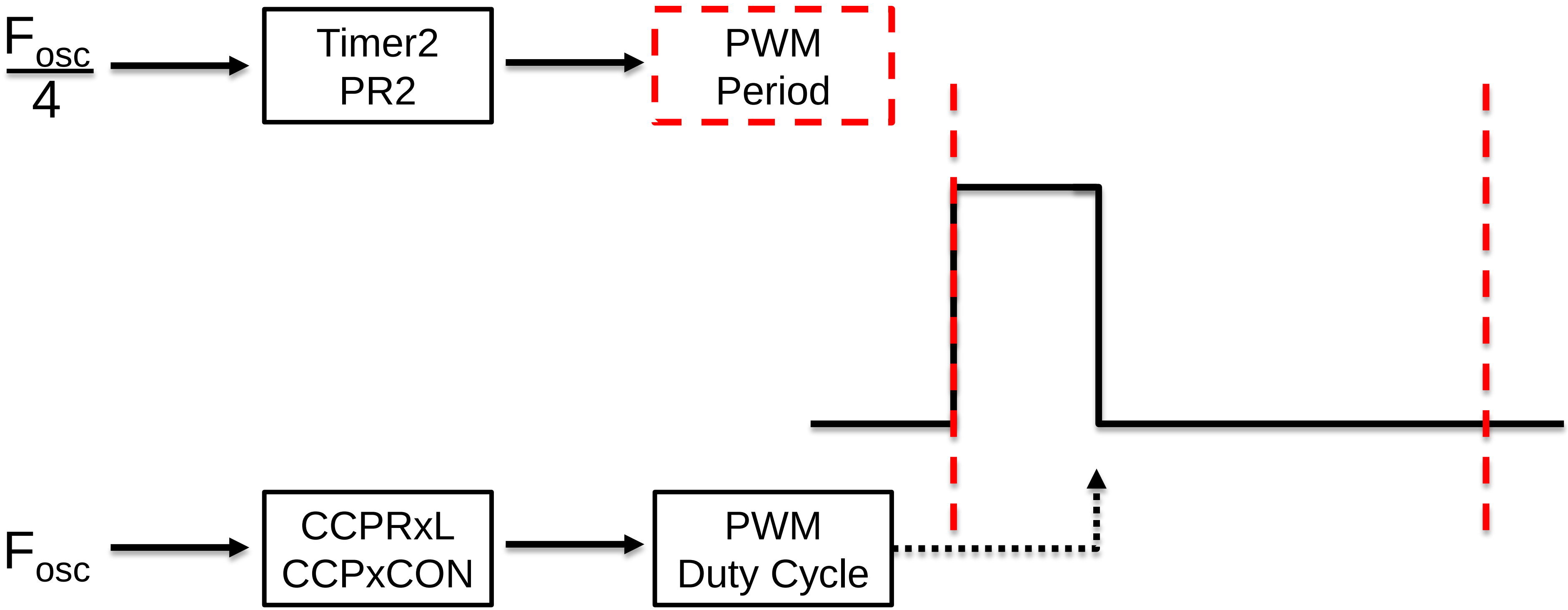
PWM Period and Duty Cycle



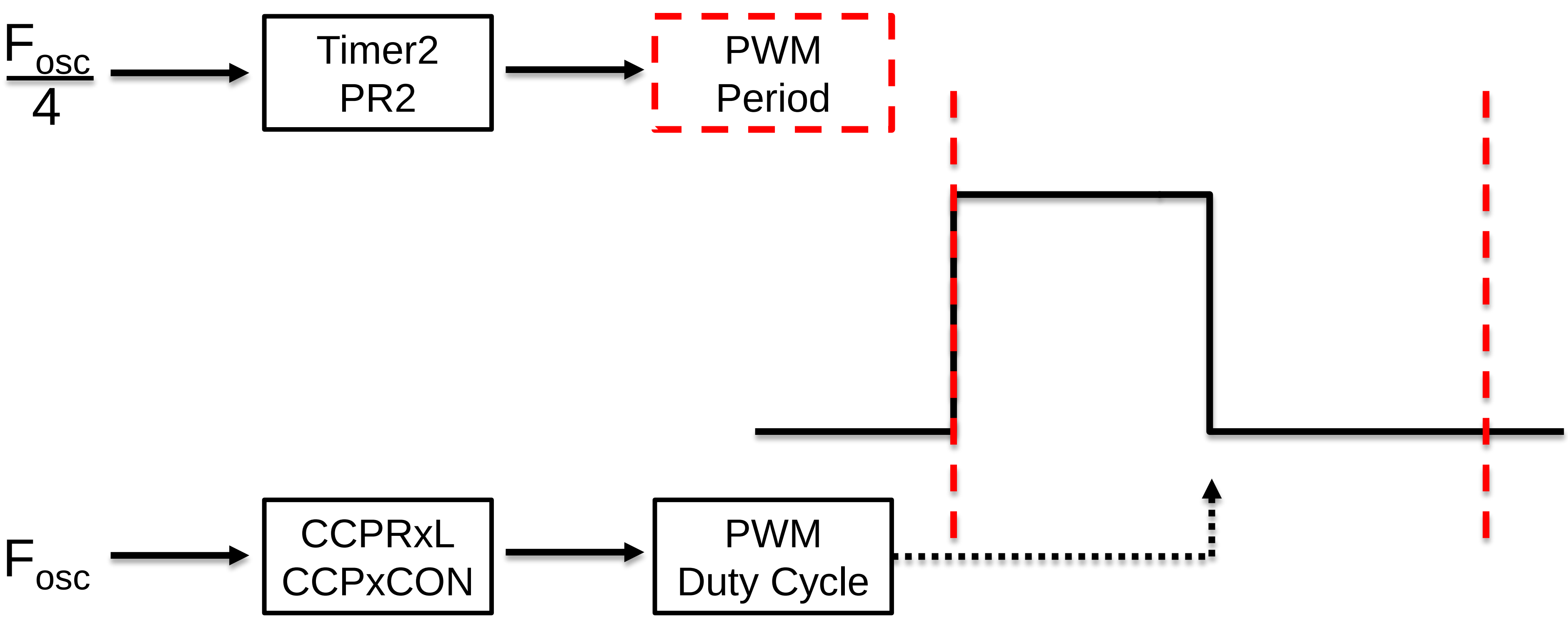
PWM Period and Duty Cycle



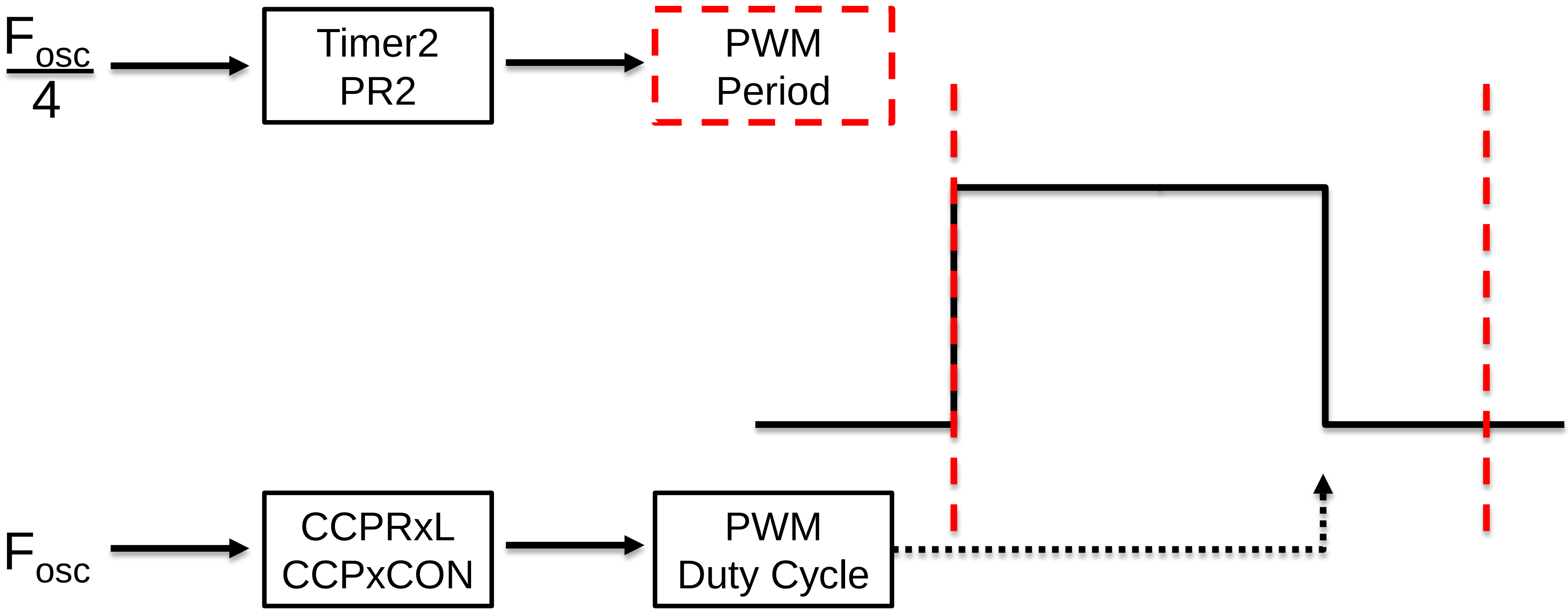
PWM Period and Duty Cycle



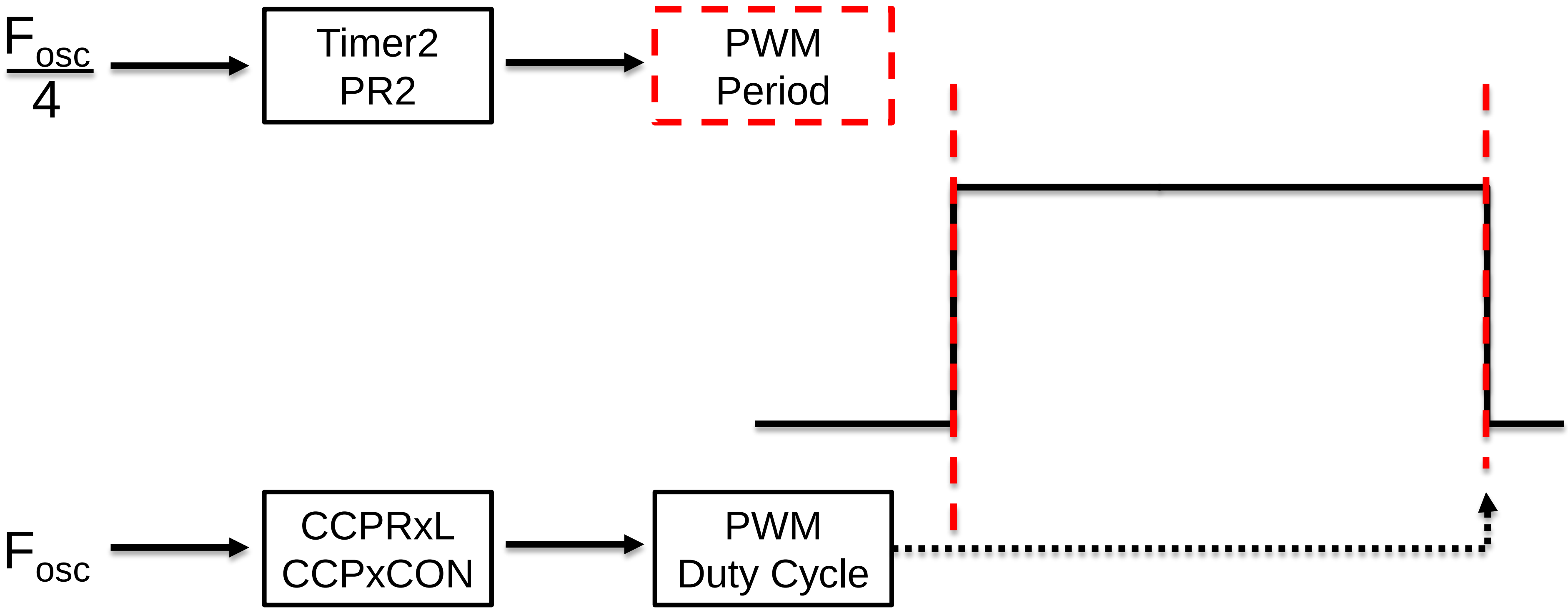
PWM Period and Duty Cycle



PWM Period and Duty Cycle



PWM Period and Duty Cycle



PWM Mode
Desired Duty Cycle

- PIC18F452 has “10-bit” duty cycle resolution
  – Remember, DC is just a percentage of the period

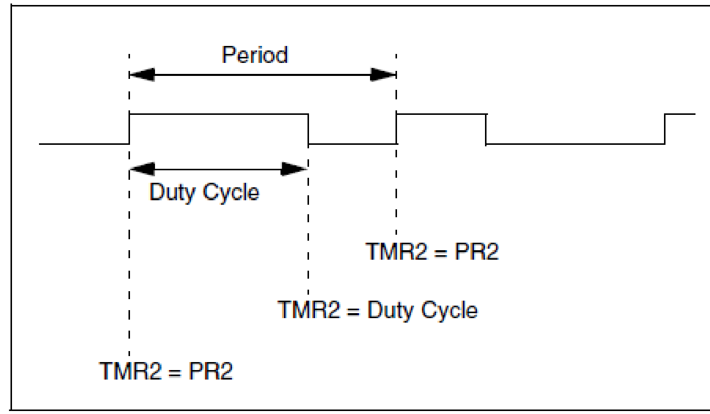
Ex:

– DC[%] = 75% = .75

– T_{PWM} = .4ms

– T_{DC} = (.75)(.4ms) = .3ms

FIGURE 14-4: PWM OUTPUT



Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
----------------------------------CCPRxL--------------------------------								CCPxCON DCxB1	CCPxCON DCxB0

CCPR1L	Capture/Compare/PWM Register1 (LSB)							
CCP1CON	—	—	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0

PWM Example

- **Knowns**
  - F_{OSC}
  - F_{PWM}
  - DC(%)
- **So we also know**
  - T_{OSC}
  - T_{PWM}
  - T_{DC}
- **Unknowns to Calculate**
  - PR2 register value
    - To set T_{PWM} (PWM period)
  - CCPR1L:CCP1CON<5:4> register value
    - To set T_{DC} (Duty Cycle period)

CSE@UTA

PWM Example

- F_{OSC} = 10 MHz
- F_{PWM} = 2.5 kHz
- DC(%) = 75%

- Note
  PR2 (8-bit): 0-255
  CCPR… (10-bit): 0-1023

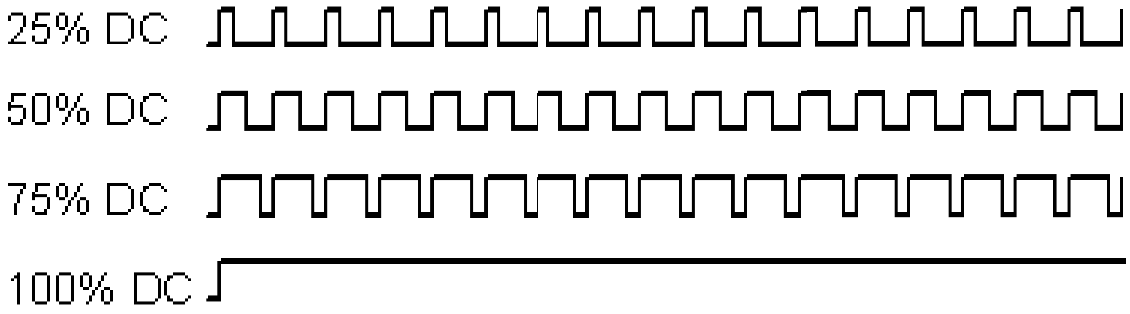
$$\text{PWM period} = (PR2) + 1] \bullet 4 \bullet T_{OSC} \bullet (\text{TMR2 prescale value})$$

Calculate these values

$$\text{PWM duty cycle} = (CCPR1L{:}CCP1CON<5{:}4>) \bullet T_{OSC} \bullet (\text{TMR2 prescale value})$$

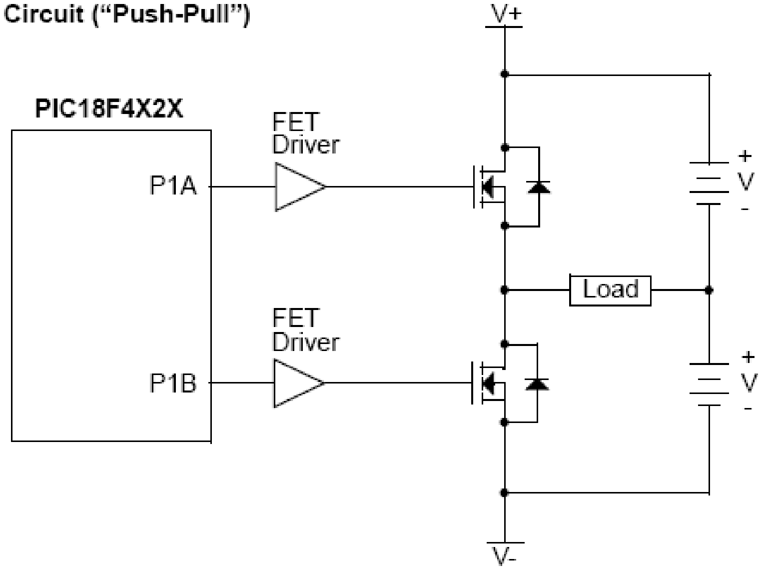
PWM Mode - Programming

1. Set PWM **period** by setting PR2 and T2CON (prescaler)
2. Set PWM **duty cycle** by calculating and writing **top 8 bits** to CCPRxL and the **remainder** 2 to CCPxCON<5:4> bits
3. Set the CCPx as output (TRIS)
4. Clear TMR2
5. Set CCPx to **PWM mode**
6. Start Timer 2
7. CCPx output pin will constantly keep outputting your signal at the set period and DC until you turn it off/disable it

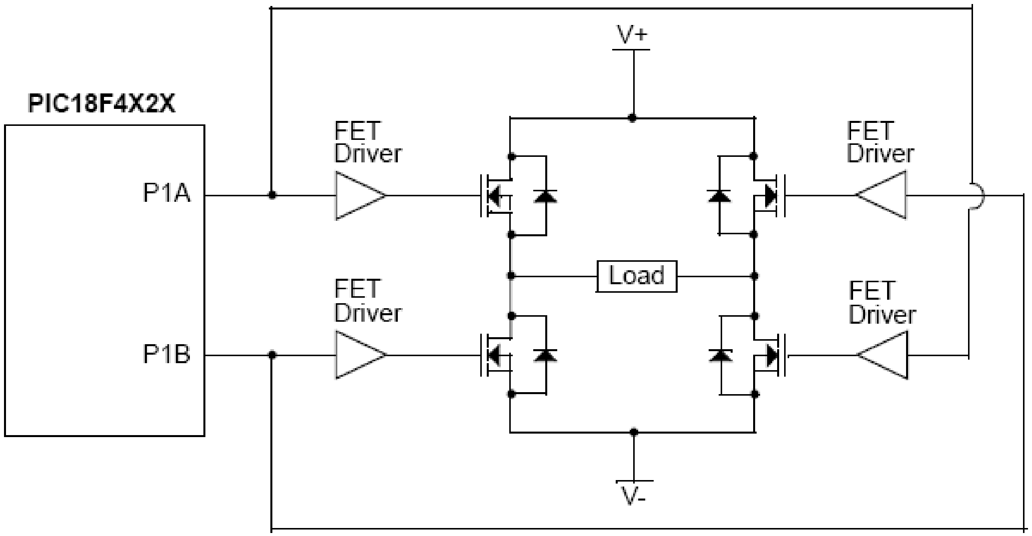


DC Motor Drive Half bridge

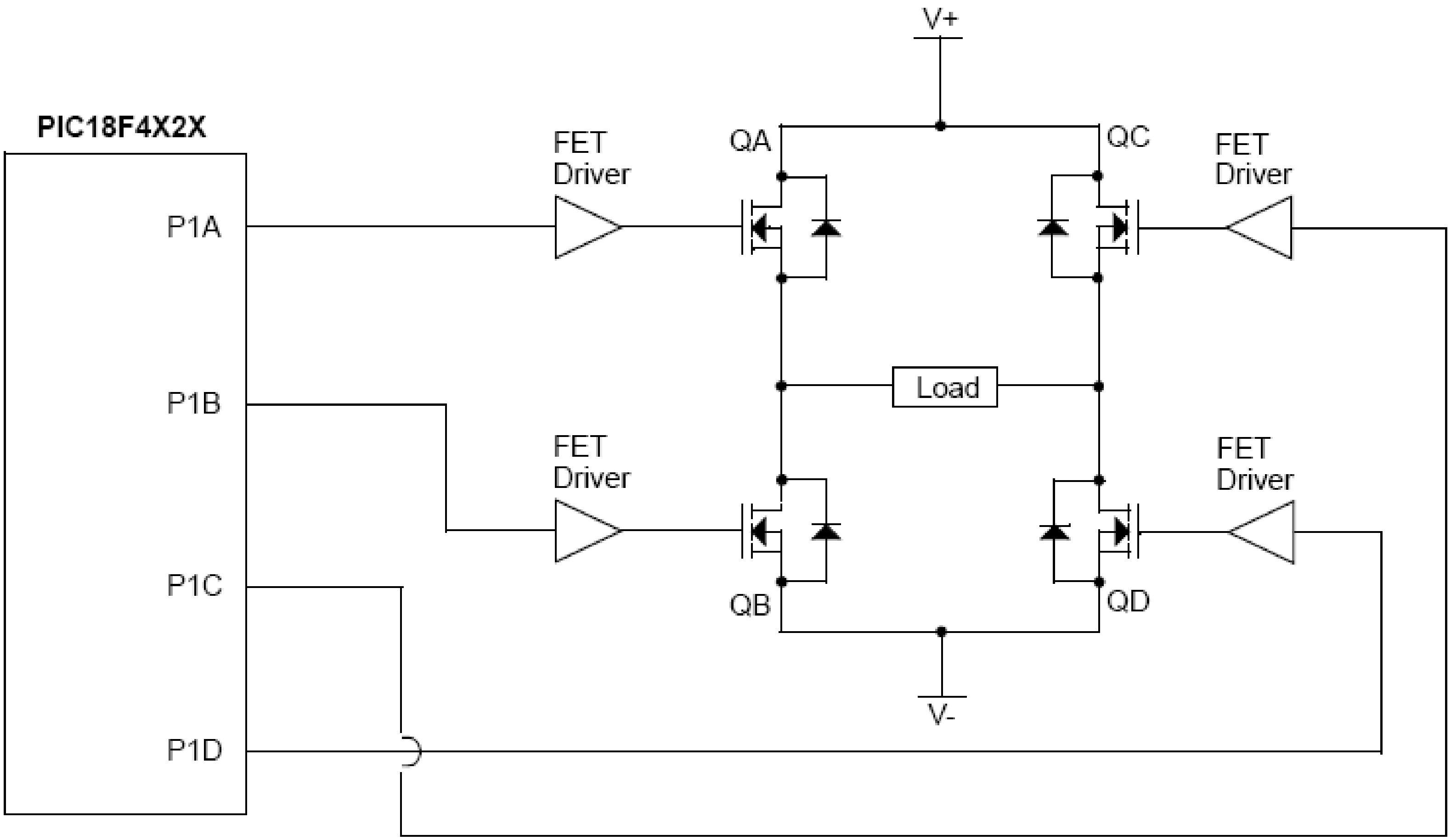
Standard Half-Bridge Circuit ("Push-Pull")



Half-Bridge Output Driving a Full-Bridge Circuit

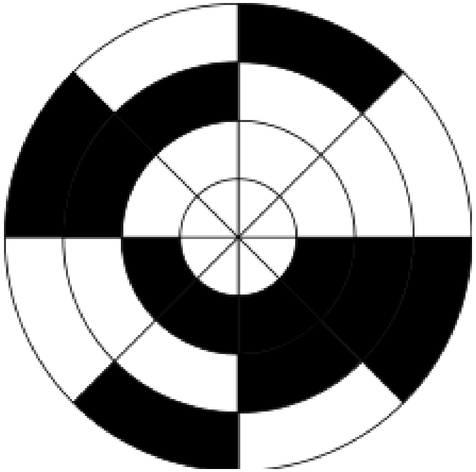


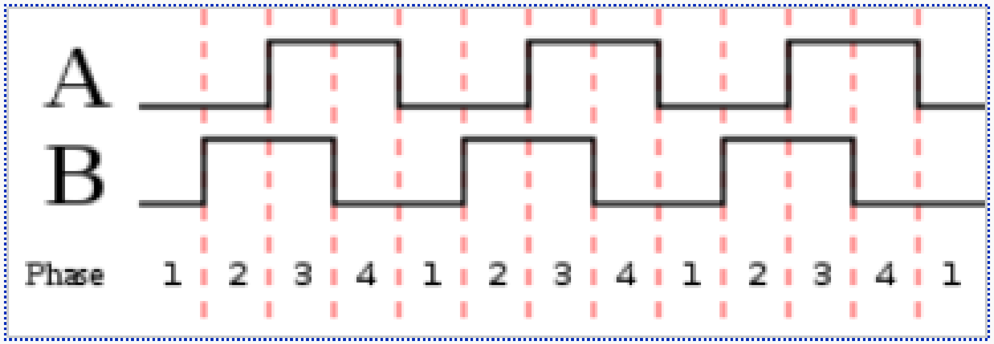
DC Motor Drive Full H Bridge



Rotary Encoders

- Rotary encoders are rotational sensors (one component of servos), they can provide precise readings (PWM) of shafts turning (flow valves, etc.)
- Internally they can be mechanical, magnetic (induction) based or optical
- Optical encoders are usually of high precision, contain encoder wheels
- Encoders can be absolute or incremental
- They can be read using timers but will tie up microcontroller; there are special purpose circuitry to read them, which have parallel or serial interfaces to microcontrollers





Two square waves in quadrature (clockwise rotation).

CCP Questions?

- Lab 7 will be detailed and given out soon
  - Take-home lab/project
  - Explained in lab and class
- Remaining Lectures…
  - Hardware Connections (ICSP, .hex details, etc.)
  - Communication (MSSP, SPI, USART, I2C, etc.)