DIGITAL CONTROLLER

3rd STAGE

CHAPTER: 3
ON – CHIP CCP

Lecturer : Ali Salman Kurji

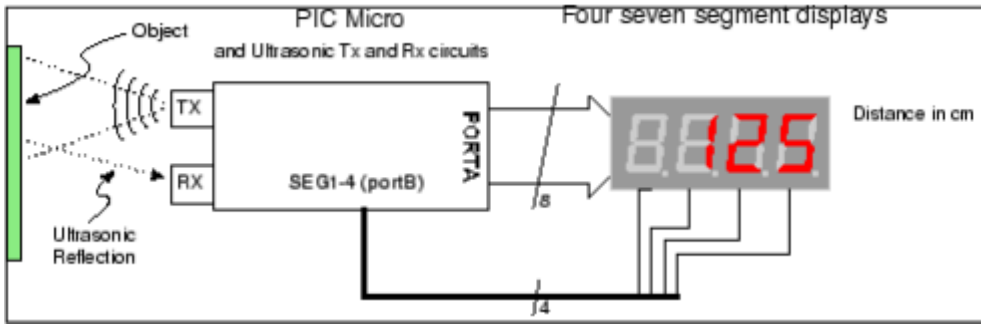
Chapter 3

On-Chip CCP

Capture | Compare | PWM

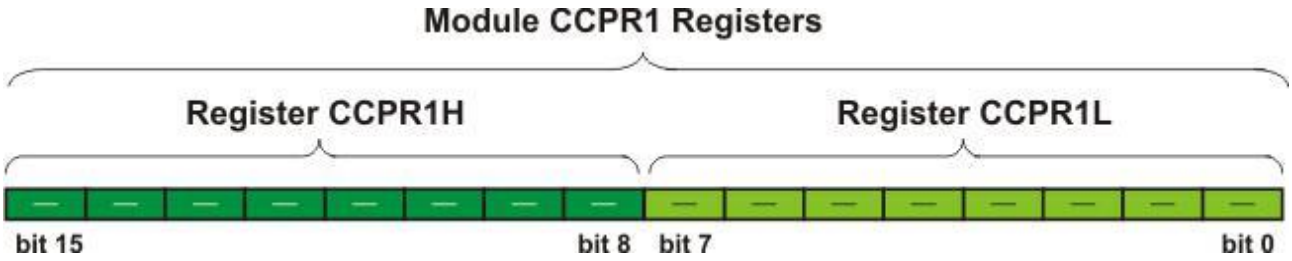
This chapter will discuss, CCP which stands for Capture, Compare and Pulse width modulation is one of the most complicated modules in PIC microcontrollers. I will go through this module only briefly, as not overburden the beginner. In fact this module does three functions, or it has three modes. There are two such modules present in PIC18F452.

Now consider an example of ultrasonic range finder. The project consists of a few driving transistors, and standard 40KHz ultrasonic transducers. The microcontroller uses its CCP module in capture mode. So the value of timer1 is noted and an impulse is given on Tx transducer, when the impulse is returned the CCP module stores the new value of Timer1 in CCP register, subtracting the previous value from current value gives you the time to echo.



From this you can easily calculate distance. (See projects section).

If we talk about CCP1, it is the same as CCP2. The central point in CCP module is CCPR register. This is a 16 bit register and consists of a H and L parts. This register contains the values captured or compared with



Timer1. Remember when Capture is initiated, Timer1, is not initialized. And at the time of capture, whatever the value of Timer1 register is copied into this register.

In **Capture Mode**, the peripheral allows timing of duration of an event. This circuit gives insight into the current state of some register which constantly changes its value. In this case, it is the timer TMR1 register. Thus with this mode we can measure for how long a pin remained in logic 1 or 0.

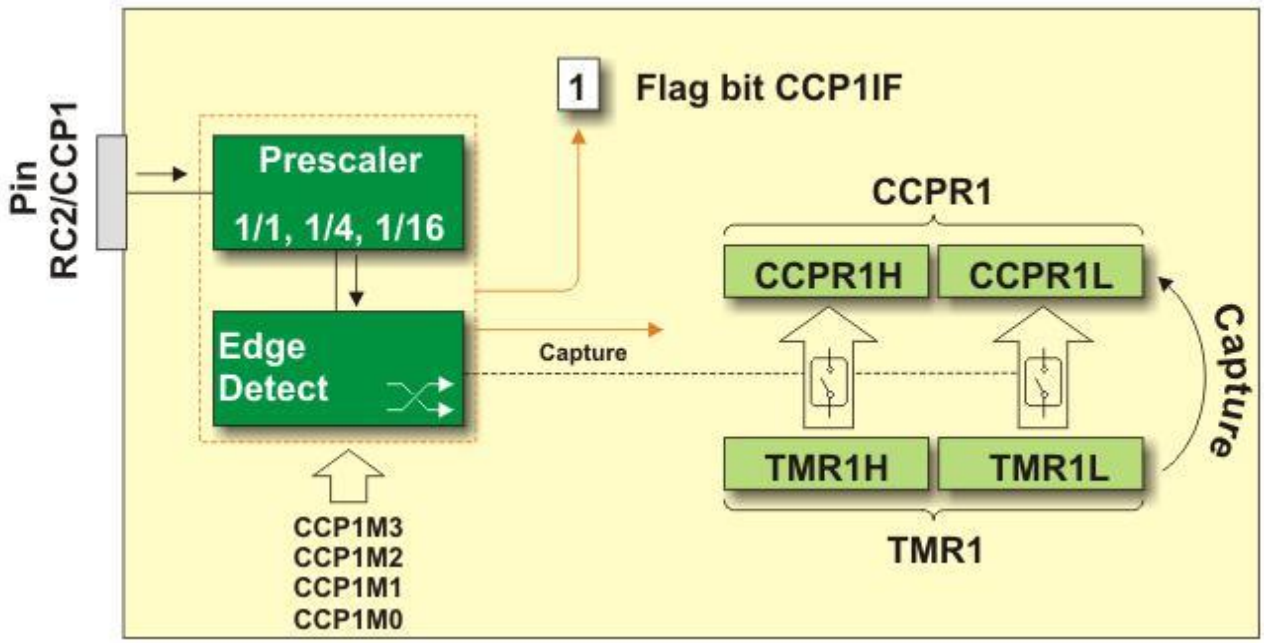


Fig: Capture Mode Operation Diagram

The **Compare Mode** compares values contained in two registers at some point. One of them is the timer TMR1 register. This circuit also allows the user to trigger an external event when a predetermined amount of time has expired. Thus if you set a compare register to some value, when Timer1 reaches that value, a capture event takes place and an interrupt signal is fired indicating that a predefined time period has been elapsed.

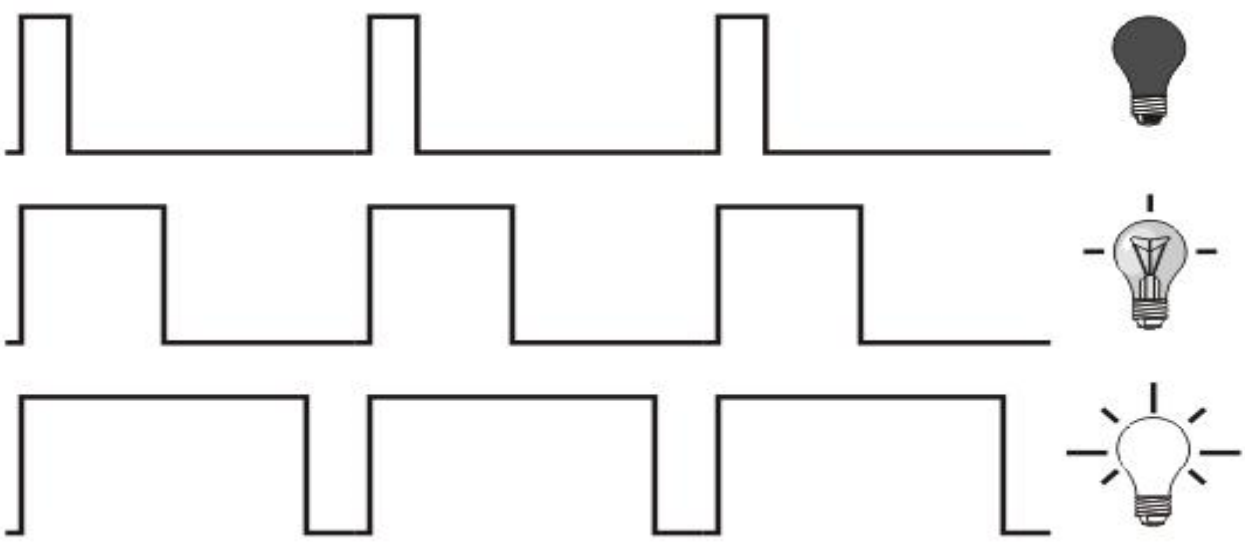
PWM - *Pulse Width Modulation* can generate signals of varying frequency and duty cycle. The ratio between ON and OFF states of the pulse determines the amount of energy transferred to device. This method is useful in controlling the speed of motors, or brightness of LEDs etc.

Pulse Width Modulation

Pulse width modulation is a technique where digital data is used to control the energy transfer to a device.

Whenever a digital signal is high it is powering the target device, like a transistor, or LED. When it is Low it is not powering the device. If the line is constantly kept high, full energy (100%) is being transferred to the device and when it is constantly Low, there is No energy transfer. In between if the line is On for some time and Off for some time, the energy delivered depends upon the ON time / Off time ratio as well as the frequency of pulses. As you can see in this figure when On time is small and Off time long, Bulb hardly gets any time to turn ON, As the ON time is increased and

OFF time decreased it gets brighter. This is called the Duty Cycle. So the duty cycle is ratio between ON and OFF. A 50% duty cycle is equal ON and Equal OFF time per cycle.

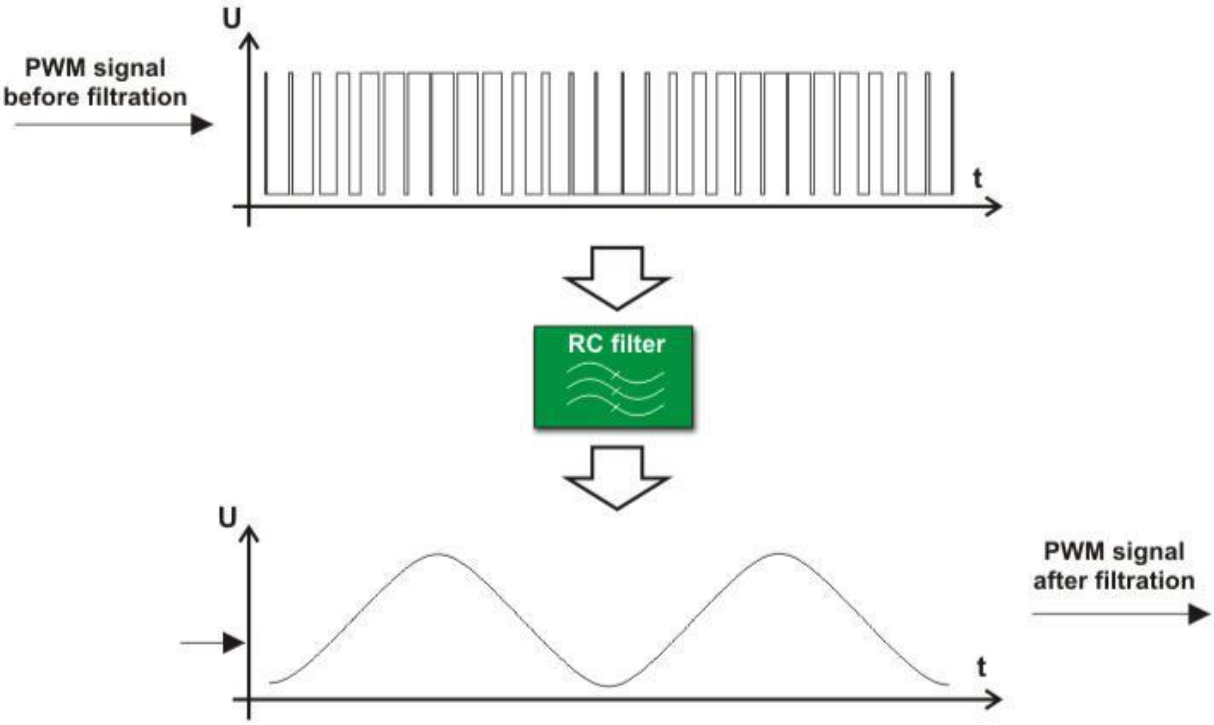


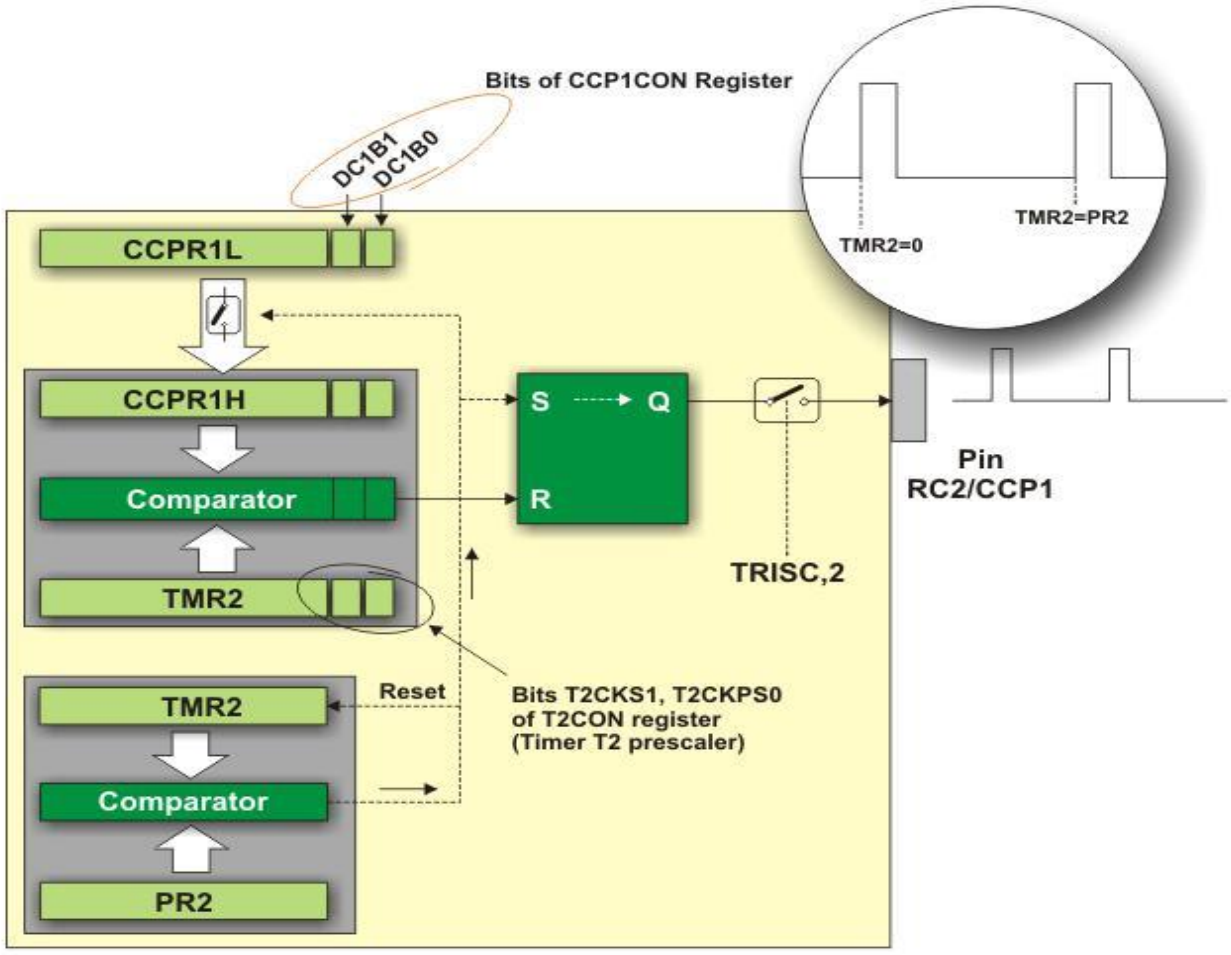
Another common usage of PWM is creation of various kinds of waveforms, like sine wave. If you recall the sounds chapter, various types of sound waves are formed by internally using PWM.

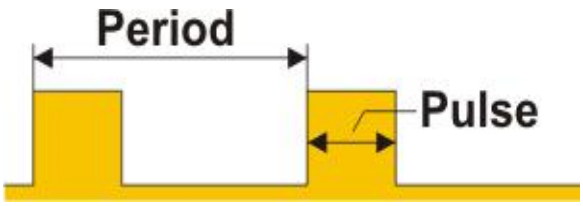
In order to produce a waveform from the digital circuit, we have to include some sort of filter. This filter can be as simple as an RC-Filter, which charges the capacitor, and gradually discharges. The rate and speed of charging is influenced by the width of pulse.

Notice when pulses are wide, the waveform reaches peak, and when the pulses are narrow, with smaller duty cycle the waveform falls down. Devices which operate in this way are often used in practice as switching regulators which control the operation of motors (speed, acceleration, deceleration etc.).

The figure above shows block diagram of the CCP1 module setup in PWM mode. In order to generate a pulse of arbitrary form on its output pin, it is necessary to determine only two values- pulse frequency and duration.
Although a lot needs to be discussed about these various modules, and various registers that set these parameters,


U
PWM signal
before filtration
t
RC filter
U
t
PWM signal
after filtration


Bits of CCP1CON Register
DC1B1
DC1B0
TMR2=0
TMR2=PR2
CCPR1L
CCPR1H
Comparator
TMR2
S
Q
R
TRISC,2
Pin
RC2/CCP1
Reset
Bits T2CKS1, T2CKPS0
of T2CON register
(Timer T2 prescaler)
TMR2
Comparator
PR2



I think it will be rather confusing for a beginner. So now let's come to the business, and see how our Proton Basic is going to help. Just like other commands, as we have seen, all the register level details are managed by the Proton Basic itself, and we are left with a neat easy to use code. Nevertheless a sound understanding of the things behind the scene makes a real difference.

As you know Pulse width Modulation as such is a technique, if you can produce On/OFF wave on any pin, it can be used as PWM output. In this example we are going to produce a PWM output on pin PORTC.0 which is also connected to the LED on PIC Lab-II board. So that we can see the effect. You can make a waveform simply by changing the ON and OFF times. In this example we have used a command offered by

```
Device=18F452                          Output LED
XTAL=20                                Dim x As Byte
ALL_DIGITAL true                       x=100
Symbol LED PORTC.0                     loop:
Symbol SW3 PORTE.0                     If SW3=0 Then x=x-10:DelayMS 200
Symbol SW4 PORTE.1                     If SW4=0 Then x=x+10:DelayMS 200
Input SW3                              PWM LED,x,1000

Input SW4                              GoTo loop
```

Proton Basic called PWM, accepts a pin as parameter and duty cycle as second parameter, the number of pulses to be sent as last parameter. Duty is a variable or a constant which specifies the analog level required. It ranges from 0-255. 255 produce full 5V.

PWM emits a burst of 1s and 0s whose ratio is proportional to the *duty* value you specify. If *duty* is 0, then the pin is continuously low (0); if *duty* is 255, then the pin is continuously high. For values in between, the proportion is *duty*/255. For example, if *duty* is 100, the ratio of 1s to 0s is 100/255 = 0.392, approximately 39 percent. When such a burst is used to charge a capacitor arranged, the voltage across the capacitor is equal to:-

(duty/ 255) * 5.

So if *duty* is 100, the capacitor voltage is

(100/255) * 5 = 1.96 volts.

This voltage will drop as the capacitor discharges through whatever load it is driving. The rate of discharge is proportional to the current drawn by the load; more current = faster discharge. You can reduce this effect in software by refreshing the capacitor's charge with frequent use of the **PWM** command. You can also buffer the output using an op-amp to greatly reduce the need for frequent **PWM** cycles.

So we have used a standard I/O line for PWM that is fairly good. However to keep the PWM going on the instruction must be executed continuously.

The hardware PWM module eliminates this need and continuously gives PWM pulses on the specific PWM pin, while the program continues doing something else. This is really a sort of multitasking. The output of CCP1 and CCP2 modules are hard- wired to specific pins and they may vary among PICs, so always read the datasheet. On PIC18F452 CCP1 is on RC2 pin, and fortunately we have LED on that pin as well, so we can test Hardware PWM right on board.

Notice that after declaring the HPWM statement, the processor is busy in an endless loop to display some data on LCD, while the CCP1 module is producing PWM pulses on the specified channel. Since CCP modules pins vary among processors, it is advisable to declare the CCP pin in program.

```
Device=18F452
XTAL=20
ALL_DIGITAL true
LCD_DTPIN PORTD.4
LCD_RSPIN PORTD.3
LCD_ENPIN PORTD.2
Symbol LED PORTC.2
Symbol SW3 PORTE.0
Symbol SW4 PORTE.1
Input SW3
Input SW4
Output PORTC
CCP1_PIN PORTC.2
Dim x As Byte
PORTC=0
HPWM 1,50,1000
Cls
Print At 2,1,"PWM ON"
loop:
For x=0 To 255
Print At 1,1,DEC3,x
DelayMS 200
Next x
GoTo loop
```