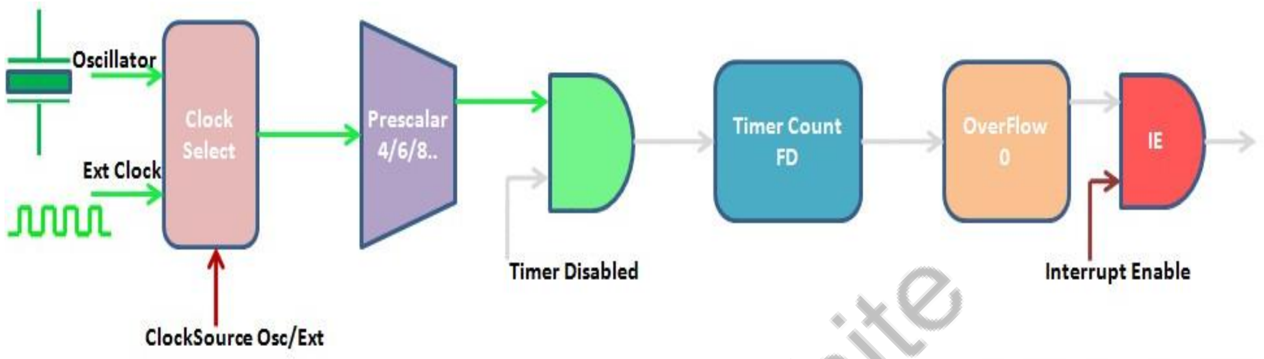
UNIT – IV: Peripheral Support in PIC 18FXXXX

Q. 1 Draw Functional Diagram of PIC Timer and Explain concept of prescaler.

- The timers are used to measure the time or generate the accurate time delay.
- The timer takes the internal clock as a reference clock, while the counter counts external clocks or pulses applied through port pins.



- PIC18F Family has 2 to 5 has independent timers which can be used as timer, Counters or for PWM generation.
- Below table provides the details of the twoTimers

Timer	Size	Control Register	Count Register	Flag Bit	Modes
TIMER0	16-bit	T0CON	TMR0H,TMR0L	TMR0IF	8 Bit and 16 Bit Mode
TIMER1	16-bit	T1CON	TMR1H,TMR1L	TMR1IF	16 Bit Mode only

- 8-bit Mode capable of counting 0-255
- 16-bit Mode – capable of counting 0-65535

Prescalar Concept

Prescalar is a configurable clock-divider circuit.

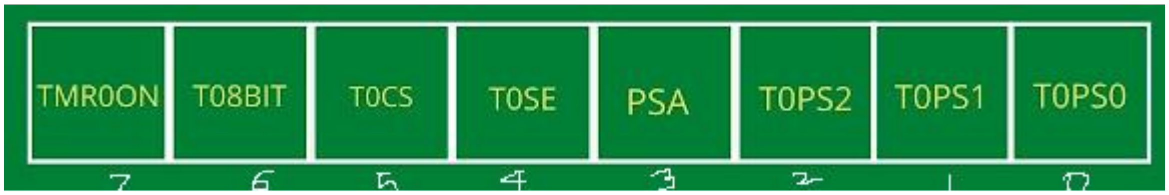
It can be used to divide the clock frequency input to the timer module.

For example, if the instruction clock is 5MHz and we use a prescaler of 2 to divide it which effectively makes the clock 2.5MHz.

So each counting time will increase from 0.2 µs to 0.4 µs

Q. 2 Explain T0CON and T1CON Register in PIC 18Fxxx

T0CON is an 8-bit register used to Control the Timer 0 Operations



Explanation of all 8 Bits is Given Below,

TMR0ON (Timer0 on/off control bit)

1 = Enable timer 0
0 = Disable timer 0

T08BIT (Timer0 8-bit/16-bit control bit)

1 = Configured as an 8-bit timer/counter
0 = Configured as an 16-bit timer/counter

T0CS (Timer0 clock source select bit)

1 = Transition on T0CKL pin
0 = Internal instruction cycle clock (CLK0)

T0SE (Timer0 Source Edge select bit)

1 = High to Low transition on T0CKL pin
0 = Low to High transition on T0CKL pin

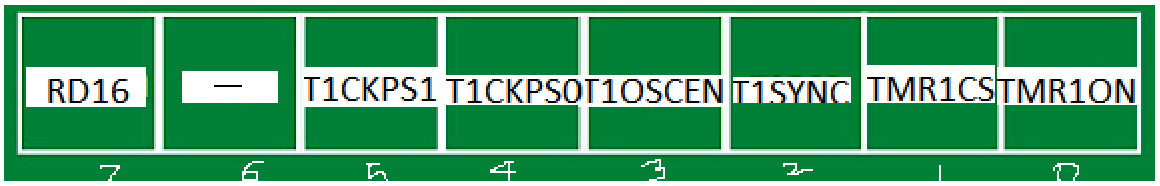
PSA (Timer0 Prescaler assignment bit)

1 = Timer 0 Prescaler is not assigned
0 = Timer 0 Prescaler is assigned

T0PS2 - T0PS0 (Timer0 prescaler select bit)

111 - 1:256 prescaler value
110 - 1:128 prescaler value
101 - 1:64 prescaler value
100 - 1:32 prescaler value
011 - 1:16 prescaler value
010 - 1:8 prescaler value
001 - 1:4 prescaler value
000 - 1:2 prescaler value

T1CON is an 8-bit register used to Control the Timer 1 Operations



Explanation of all 8 Bits is Given Below,

TMR1ON (Timer1 on/off control bit)

1 = Enable timer 1
0 = Disable/STOP timer 1

TMR1CS (Timer1 clock source select bit)

1 = External Clock
0 = Internal Clock

T1sync (Timer1 synchronization bit)

1 = Counter mode (for external clock only)
0 = otherwise

T1OSCEN (Timer1 Oscillator enable bit)

1 = Timer1 Oscillator enabled
0 = Timer1 Oscillator off

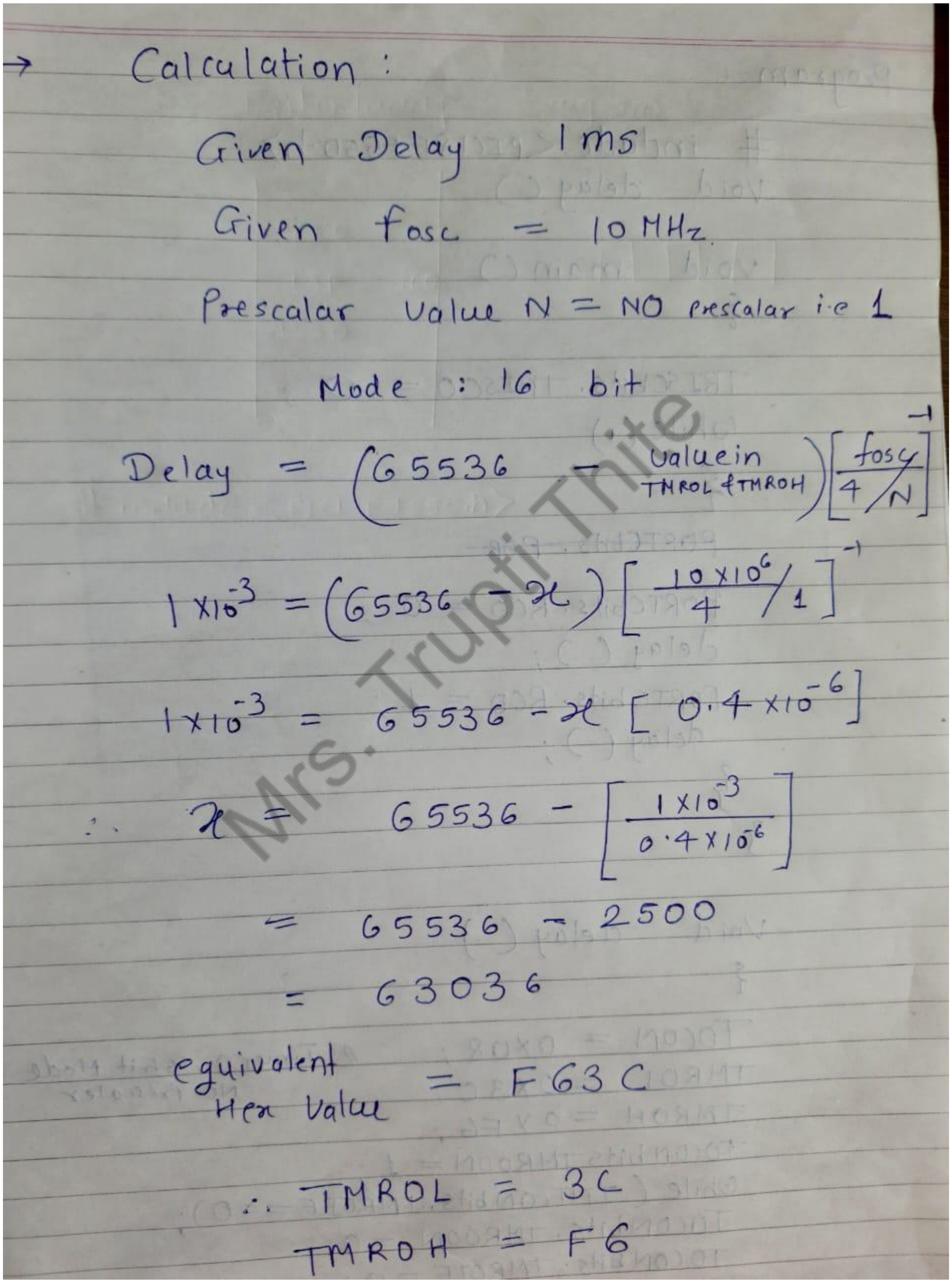
T1CKPS1:T1CKPS0 (Timer1 Prescaler SELECTOR bit)

00 – 1:1 prescaler value
01 - 1:2 prescaler value
10 - 1:4 prescaler value
11 - 1:8 prescaler value

RD16 (16 BIT Enable bit)

1 = One 16 Bit Operation
0 = two 8 bit operation

Q. 3 Write C code to toggle RC0 bit after 1m second, 16-bit operation Timer 0, XTAL frequency = 10MHz (without Prescaler)



→ Calculation :

Given Delay 1 ms

Given fosc = 10 MHz

Prescalar Value N = NO prescalar i.e 1

Mode : 16 bit

$$\text{Delay} = \left(65536 - \underset{\text{TMR0L \& TMR0H}}{\text{value in}}\right)\left[\frac{fosc}{4}\Big/N\right]^{-1}$$

$$1\times10^{-3} = (65536 - x)\left[\frac{10\times10^{6}}{4}\Big/1\right]^{-1}$$

$$1\times10^{-3} = 65536 - x\,[0.4\times10^{-6}]$$

$$\therefore\ x = 65536 - \left[\frac{1\times10^{-3}}{0.4\times10^{-6}}\right]$$

$$= 65536 - 2500$$

$$= 63036$$

equivalent Hex Value = F63C

$\therefore$ TMR0L = 3C

TMR0H = F6

Program :

```
# include <PIC18F4550.h>
Void delay ()

Void main ()
{

TRISCbits. TRISC0 = 0 ;
While (1)
 {
 ~~PORTEbits. POR~~
 PORTCbits. RC0 = 0 ;
 delay ( );
 PORTCbits. RC0 = 1;
  delay ( );
 }
}

Void delay ( )
 {
 T0CON = 0x08 ;      // Timer0, 16bit Mode
                        No Prescaler
 TMR0L = 0x3C ;
 TMR0H = 0xF6 ;
 T0CONbits.TMR0ON = 1 ;
 While ( INTCONbits. TMR0IF == 0);
 T0CONbits. TMR0ON = 0;
 T0CONbits. TMR0IF = 0;

 }
```

Q. 4 Explain interrupt structure of PIC18F

- ❑ Interrupts are mechanisms which enable instant response to events
- ❑ In normal mode, microcontroller executes the main program using Polling
- ❑ Upon interrupt, microcontroller stops the execution of main program and commences the special part of the program(ISR) which will analyze and handle the interrupt.

The PIC18F devices have multiple interrupt sources and an interrupt priority feature that allows each interrupt source to be assigned a high priority level or a low-priority level

PIC18F Interrupts Sources

- Timers 0, 1
- Pins RB0, RB1, RB2 for external hardware interrupts INT0,INT1, INT2
- PORTB-Change interrupt (any one of the upper four Port B pins)
- Serial communication's USART interrupts: receive and transmit, respectively
- ADC (analog-to-digital converter)
- CCP (compare capture pulse-width-modulation)
- There are ten registers which are used to control interrupt operation.

These registers are:

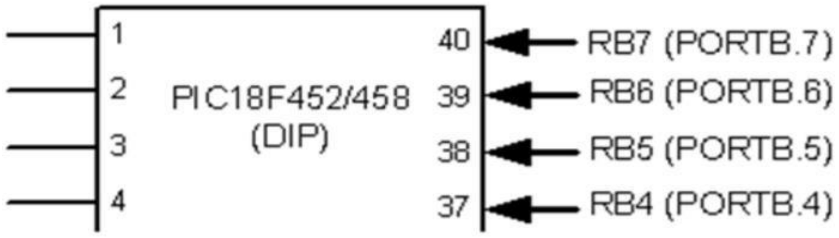
- RCON
- INTCON
- INTCON2
- INTCON3
- PIR1, PIR2
- PIE1, PIE2
- IPR1, IPR2

- **Interrupt vector table for PIC18 (ROM location)**

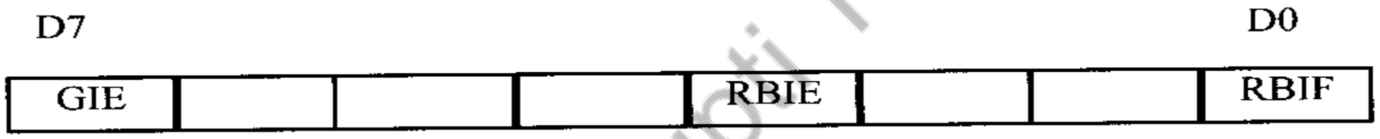
➢ **Power-on reset**	**0x0000**
➢ **High priority interrupt**	**0x0008**
➢ **Low priority interrupt**	**0x0018**

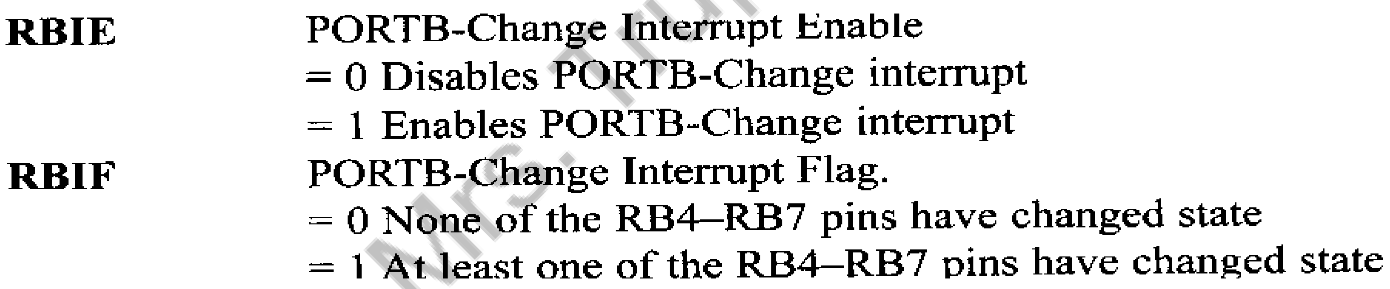
Q. 5 Explain PORTB Change Interrupt of PIC18F

- Four pins of PORTB can cause an interrupt when any changes detected on them, They are referred as PORTB change interrupt
- An input change on PORTB<7:4> sets flag bit, RBIF (INTCON<0>).
- The interrupt can be enabled/disabled by setting/clearing enable bit, RBIE and it has flag RBIF Located in INTCON



INTCON Register



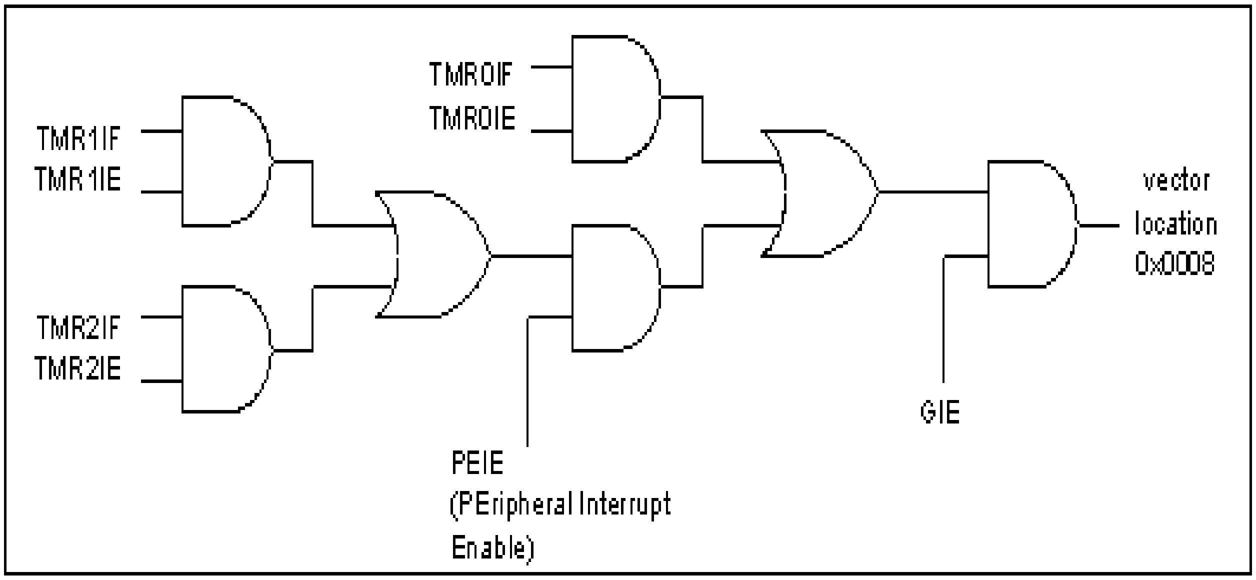


RBIE	PORTB-Change Interrupt Enable = 0 Disables PORTB-Change interrupt = 1 Enables PORTB-Change interrupt
RBIF	PORTB-Change Interrupt Flag. = 0 None of the RB4–RB7 pins have changed state = 1 At least one of the RB4–RB7 pins have changed state

Q. 6 Elaborate Timer Interrupts and Steps for Programming PIC18F4550 Timer using Interrupt.

Timer Interrupt Flag Bits and Associated Registers are given below

Interrupt	Flag Bit	Register	Enable Bit	Register
• Timer0	• TMR0IF	• INTCON	• TMR0IE	• INTCON
• Timer1	• TMR1IF	• PIR1	• TMR1IE	• PIE1

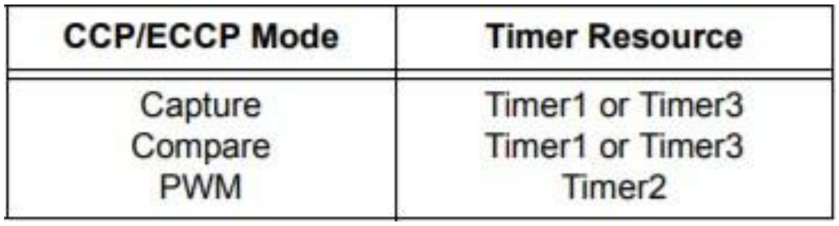


Steps for Programming PIC18F4550 Timer using interrupt

1. Enable GIE, PEIE, TMR1IE.
2. Configure the T1CON register.
3. Clear TMR1IF Timer1 interrupts flag.
4. Load the count in TMR1H (higher byte) andTMR1L (lower byte).
5. Set TMR1ON to start the Timer1 operation.
6. When TMR1IF = 1, code will jump to ISR to execute it, and after execution control returns to the main program.

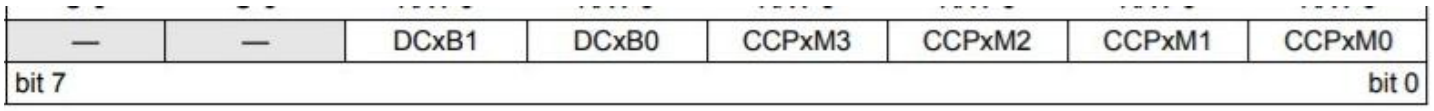
Q. 9 Describe CCP module in PIC 18.

- PIC 18 has 0 and 5 CCP modules inside it as CCP1, CCP2 and CCP so on.
- PWM feature of the CCP has been enhanced for better DC motor control. It is called enhanced CCP [ECCP].
- Each Capture/Compare/PWM module is associated with a control register (generically, CCPxCON) and a data register (CCPRx).
- The data register, in turn, is comprised of two 8-bit registers: CCPRxL (low byte) and CCPRxH (high byte).



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

CCPxCON Register



—	—	DCxB1	DCxB0	CCPxM3	CCPxM2	CCPxM1	CCPxM0
bit 7							bit 0

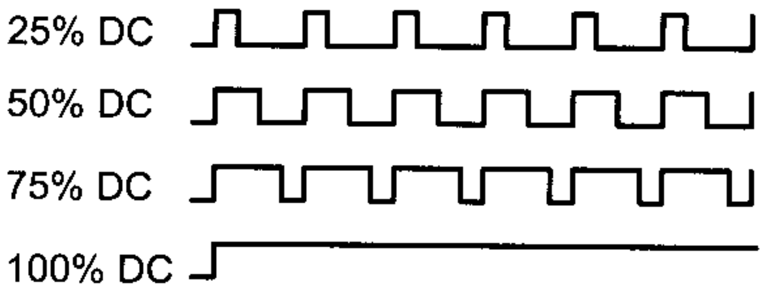
DCXB1:DCXB0:-Used to select duty cycle in PWM Mode

DC1B2	DC1B1	Decimal points
0	0	0
0	1	0.25
1	0	0.5
1	1	075

CCxM[3:0]:- Used to specify the mode of operation of CCPx module

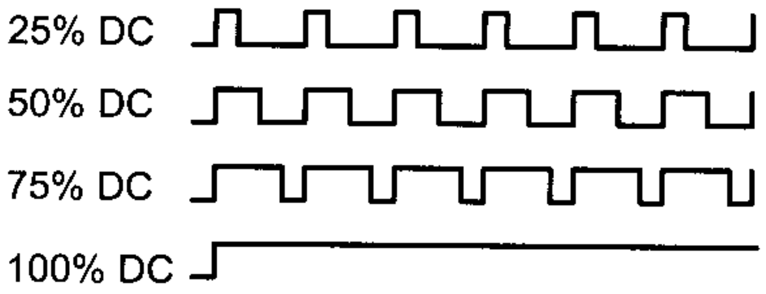
0000 : Capture/Compare/PWM disabled (resets CCPx module)
0010: Compare mode: Toggle output on match (CCPxIF bit is set)
0100: Capture mode: Every falling edge
0101: Capture mode: Every rising edge
0110: Capture mode: Every 4th rising edge
0111: Capture mode: Every 16th rising edge
1000: Compare mode
1001: Compare mode
1010: Compare mode
1011: Compare mode:
11xx: PWM mode

- **Capture Mode** provides access to the current state of a register which constantly changes its value. In this case, it is the timer TMR1 register. The Capture Mode is used to time a particular event
- **Compare Mode** constantly compares values of two registers. 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.
- **PWM** (*Pulse Width Modulation*) This mode allow us to generate pulses with variable width through Program. PWM is widely used in DC Motor control



10. Explain Generation of PWM using CCP Module

PIC 18 CCP Module has PWM feature, which allows programmer to create pulses of variable widths.
Pulse Width Modulation (PWM) is a technique by which the width of a pulse is varied while keeping the frequency of the wave constant.



In creating pulses with variable widths for PWM two factors are important

1. Setting Period of Pulse
2. Setting Duty Cycle of Pulse

Steps to set two factors are given below

1. Setting Period of PWM
   To set period Calculate PR2 Register Value using below formula

$$PR2 = [Fosc / (Fpwm \times 4 \times N)] - 1$$

Here, Fpwm =Expected PWM frequency
Fosc= Oscillator frequency
N=Prescalar value

2. Setting Duty Cycle of PWM
   A. Multiply PR2 Value by expected duty cycle.
   B. Load whole number of answer in CCPRL1 Register
   C. to set factional part (Decimal Point Part) of answer use DC1B1 and DC1B2 bit as shown in below table

DC1B2	DC1B1	Decimal points
0	0	0
0	1	0.25
1	0	0.5
1	1	075

10. Write Steps to Program PWM Module

Following steps are taken to program PWM Module of CCP,

1. Set the PWM period by writing to the PR2 register.
2. Set the PWM duty cycle by writing to CCPR1L for the higher 8 bits.
3. Set the CCP pin as an output.
4. Using the T2CON register, set the prescale value. See Figure 15-14.
5. Clear the TMR2 register.
6. Configure the CCP1CON register for PWM and set DC1B2:DC1B1 bits for the decimal portion of the duty cycle.
7. Start Timer2.

11. Explain Interfacing of PIC 18 F with DC Motor for speed control using CCP Module

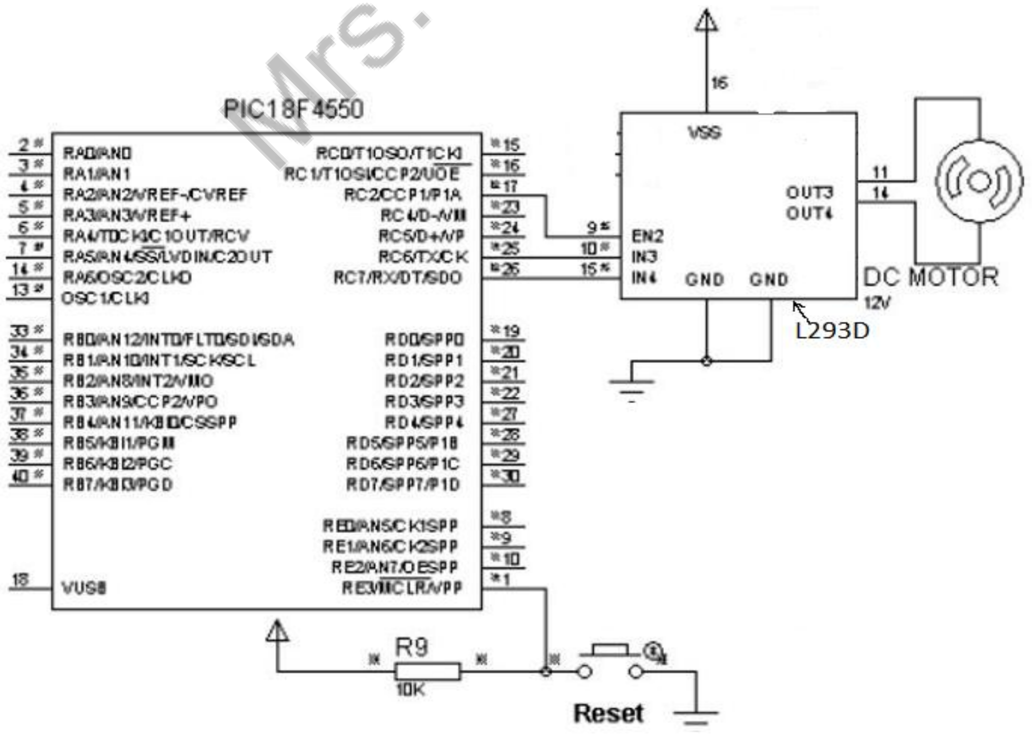
Or

Write code of DC Motor Control Using PWM Module

Or

Write a program for 2.5KHz and 75% duty cycle PWM generation with N=4. Assume Fosc = 10MHz

Answer:

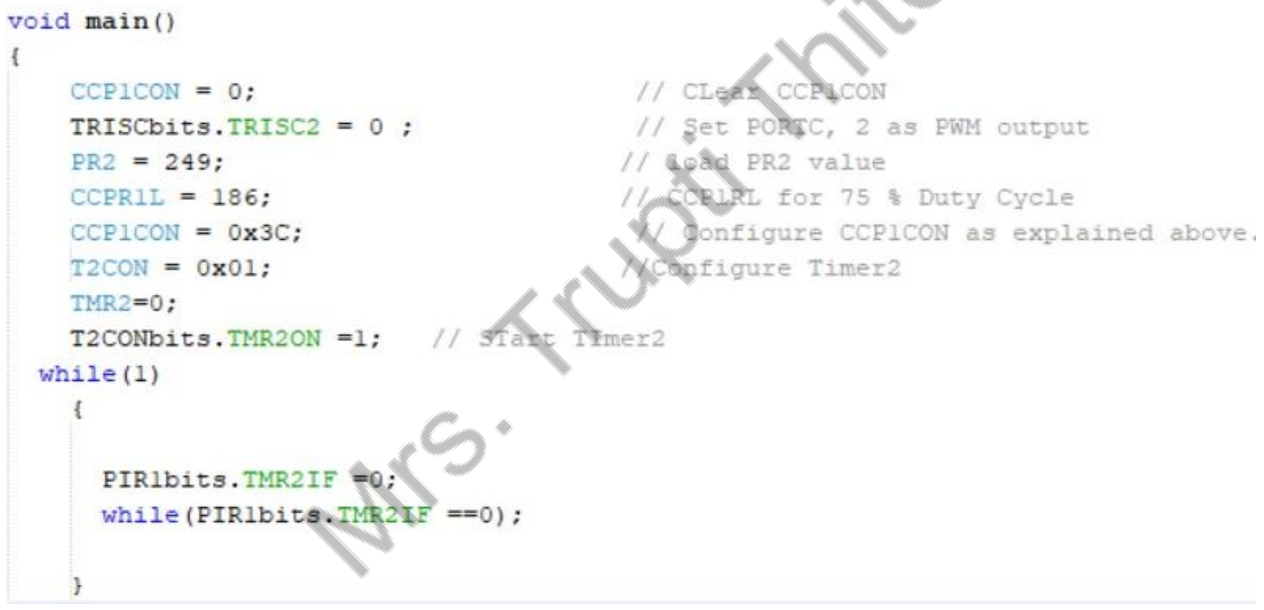


Following are steps,

1. Find value of PR2
    - PR2 = [Fosc/Fpwm*4*N]-1 = [10Mhz/2.5KHz*4*4]-1 = 249
2. Find value of CCPR1L
    - CCPR1L = PR2*Duty Cycle = 249 * 0.75 = 186.75 ≈ 186
3. Set the TMR2 pre-scalar value, and Enable TMR2 by writing T2CON
    - T2CON = 0x01 (00000001)
4. Configure CCPx module for PWM and set DC1B2 and DC1B1 for decimal portion of duty cycle 0.75 , S0 11 will be loaded into CCP1CON 5:4

Program:

```
#include<PIC18F4550.h>
```



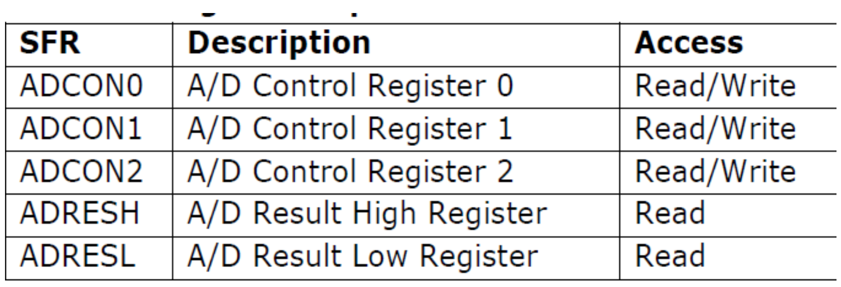
```
void main()
{
    CCP1CON = 0;                          // CLear CCP1CON
    TRISCbits.TRISC2 = 0 ;                // Set PORTC, 2 as PWM output
    PR2 = 249;                           // Load PR2 value
    CCPR1L = 186;                        // CCP1RL for 75 % Duty Cycle
    CCP1CON = 0x3C;                       // Configure CCP1CON as explained above.
    T2CON = 0x01;                        //Configure Timer2
    TMR2=0;
    T2CONbits.TMR2ON =1;    // STart TImer2
  while(1)
    {

      PIR1bits.TMR2IF =0;
      while(PIR1bits.TMR2IF ==0);

    }
```

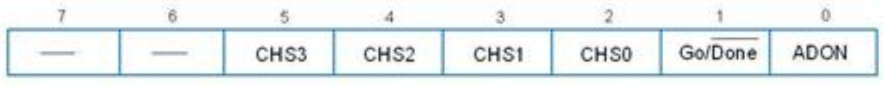
12. Explain PIC 18F ADC

Analog to digital converters are mostly used for data acquisition to convert analog sensor output into Digital.
PIC 18 F has inbuilt on chip ADC with 10 bit resolution and having 16 Channels
Following table gives details about registers required for ADC programming

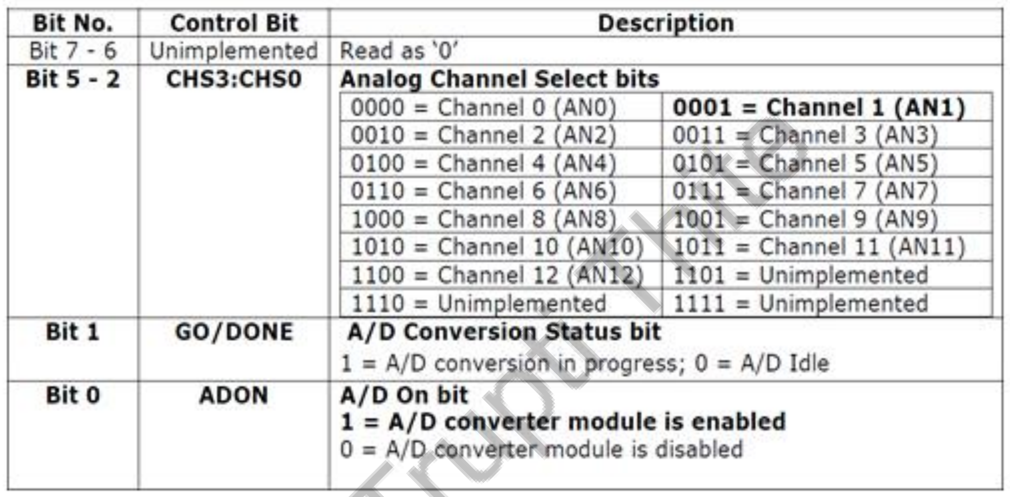


SFR	Description	Access
ADCON0	A/D Control Register 0	Read/Write
ADCON1	A/D Control Register 1	Read/Write
ADCON2	A/D Control Register 2	Read/Write
ADRESH	A/D Result High Register	Read
ADRESL	A/D Result Low Register	Read

A/D Control Register (ADCON0) is used for channel selection as below



7	6	5	4	3	2	1	0
—	—	CHS3	CHS2	CHS1	CHS0	Go/$\overline{\text{Done}}$	ADON



Bit No.	Control Bit	Description	
Bit 7 - 6	Unimplemented	Read as '0'	
Bit 5 - 2	**CHS3:CHS0**	**Analog Channel Select bits**	
		0000 = Channel 0 (AN0)	**0001 = Channel 1 (AN1)**
		0010 = Channel 2 (AN2)	0011 = Channel 3 (AN3)
		0100 = Channel 4 (AN4)	0101 = Channel 5 (AN5)
		0110 = Channel 6 (AN6)	0111 = Channel 7 (AN7)
		1000 = Channel 8 (AN8)	1001 = Channel 9 (AN9)
		1010 = Channel 10 (AN10)	1011 = Channel 11 (AN11)
		1100 = Channel 12 (AN12)	1101 = Unimplemented
		1110 = Unimplemented	1111 = Unimplemented
Bit 1	**GO/DONE**	**A/D Conversion Status bit** 1 = A/D conversion in progress; 0 = A/D Idle	
Bit 0	**ADON**	**A/D On bit** **1 = A/D converter module is enabled** 0 = A/D converter module is disabled	

Steps for Programming ADC

1. Configure ADCON1 Register to select Reference voltage using VCFG1: VCFG0 bits and also configure port pins which we require as an analog input using PCFG3: PCFG0 bits.
2. Configure ADCON2 Register to select A/D result format, A/D clock, and acquisition time.

A/D conversion and Read digital values

1. Configure ADCON0 Register to select a channel that we require using CHS3: CHS0.
2. Start A/D conversion by setting ADON bit and Go/done' bit of ADCON0 Register.
3. Wait for G0/done' bit which is cleared when the conversion is completed. 4.Then copy Digital data which is stored in ADRESH and ADRESL Register

Program is as below,

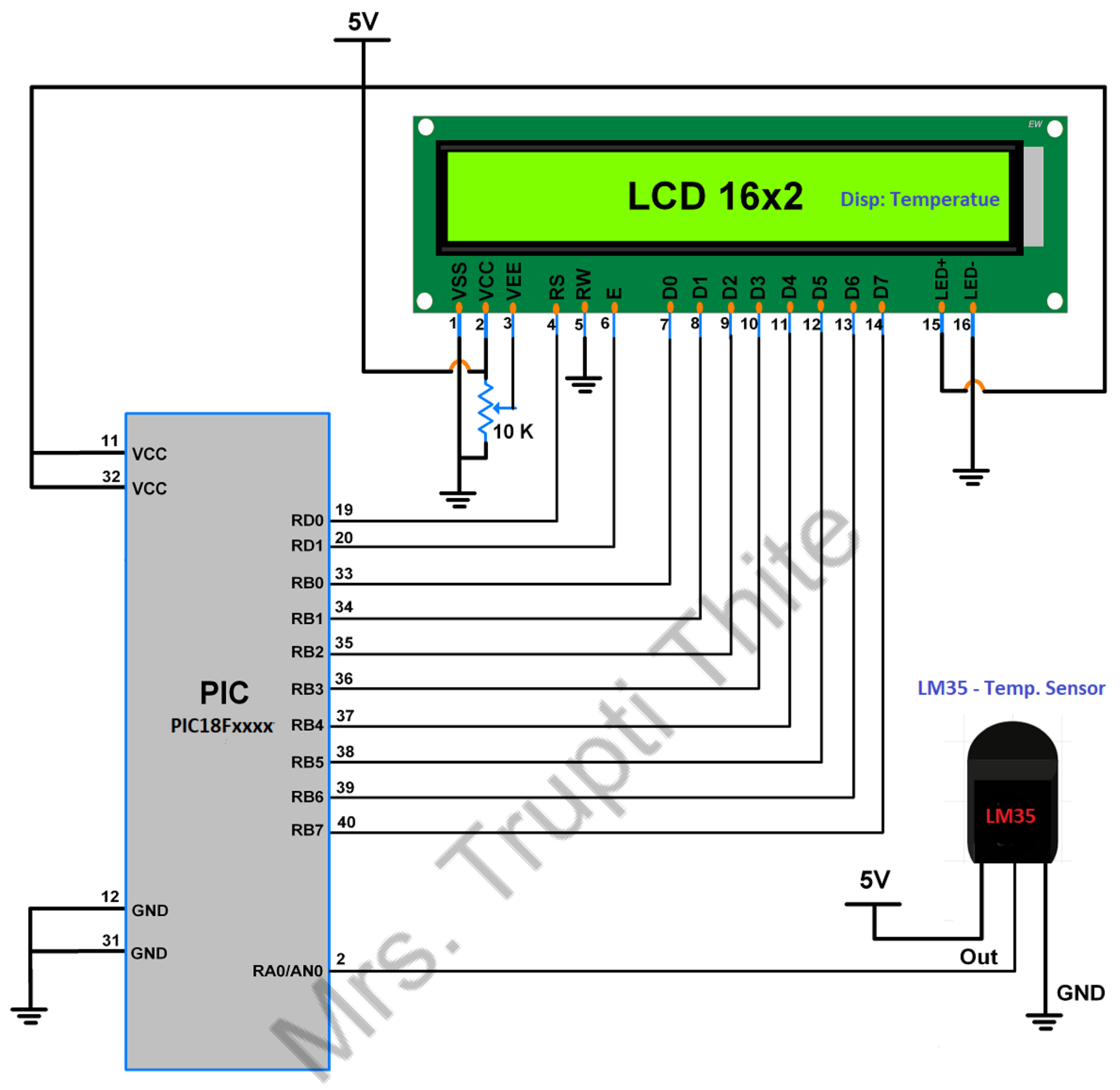
```
void main(void)
  {
     TRISC=0;              //make PORTC output port
     TRISD=0;              //make PORTD output port
     TRISAbits.TRISA0=0;   //RA0 = INPUT for analog input
     ADCON0 = 0x81;  //Fosc/64, channel 0, A/D is on
     ADCON1 = 0xCE;        //right justified, Fosc/64,
                           //AN0 = analog
     while(1)
     {
      DELAY(1); //give A/D channel time to sample
      ADCON0bits.GO = 1;   //start converting
      while(ADCON0bits.DONE == 1);
      PORTC=ADRESL;        //display low byte on PORTC
      PORTD=ADRESH;        //display high byte on PORTD
      DELAY(250);          //wait for one quarter of a
                           //second before trying again
     }
  }
```

13. Explain LM 35 Interfacing toPIC18F using ADC

- LM35 is a temperature sensor that can measure temperature in the range of -55°C to 150°C.
- It is a 3-terminal device that provides an analog voltage proportional to the temperature.
- The higher the temperature, the higher is the output voltage.
- The output analog voltage can be converted to digital form using ADC so that a microcontroller can process it.
- Let's interface the LM35 temperature sensor with PIC18FXX and display the surrounding temperature on the LCD16x2 display.
- LM35 gives output in the analog form so connect out pin of a sensor to one of the ADC channels of PIC18FXX.

$$V_{OUT} = (10\ \text{mv} / °\text{C}) \times T$$

Interfacing Diagram is as shown below,



Algorithm:

(1) The LM34 (or LM35) is connected to channel 0 (RA0 pin).

(2) The channel AN3 (RA3 pin) is connected to the Vref of 2.56 V. That makes PCFG = 0010 for the ADCON1 register.

(3) The 10-bit output of the A/D is divided by 4 to get the real temperature.

The algorithm is as follows: (a) Shift right the ADRESL 2 bits, (b) rotate the ADRESH 2 bits, and (c) OR the ADRESH with ADRESL together to get the 8-bit output for temperature.