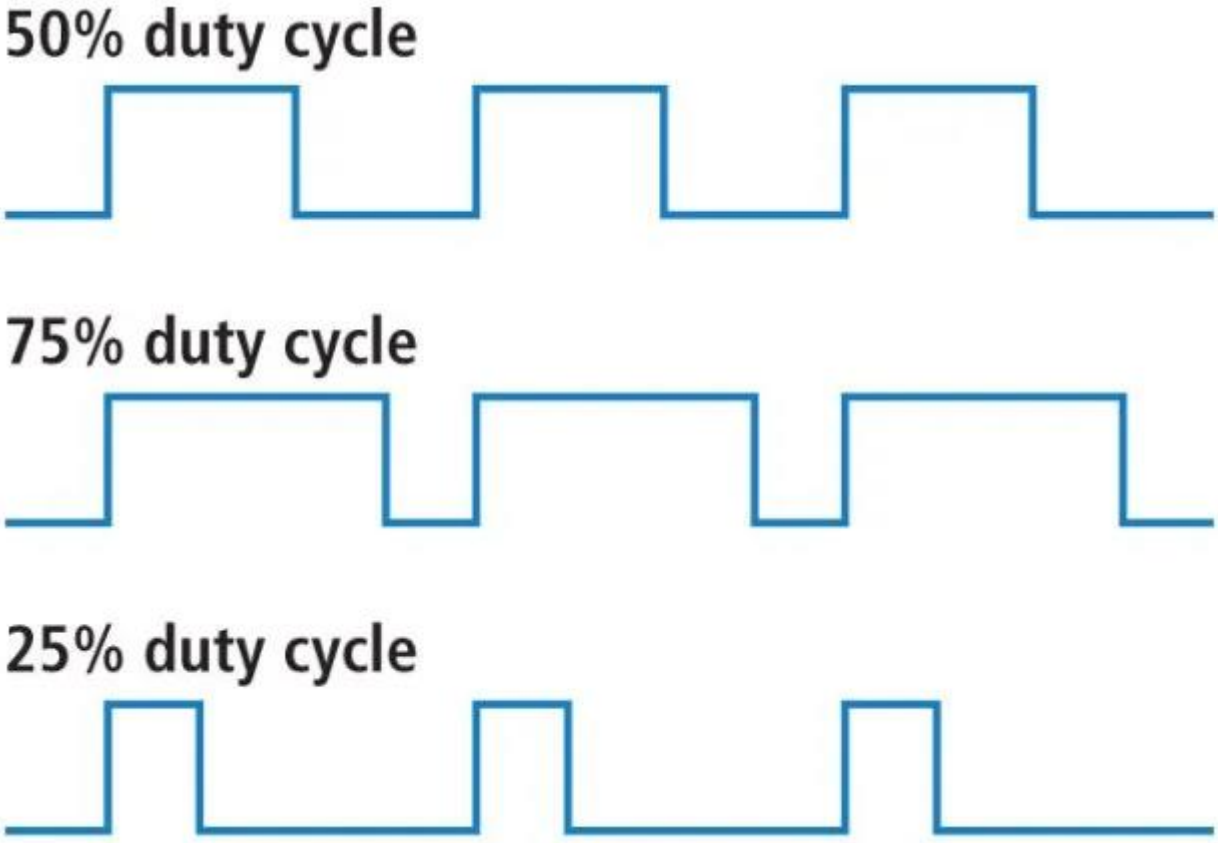
PWM - Hướng dẫn điều chế độ rộng xung | Mô-đun CCP

Tín hiệu PWM là gì?

PWM là viết tắt của Điều chế độ rộng xung. Tính năng này cung cấp vi điều khiển tính năng, xuất ra các giá trị tương tự của điện áp giữa (0-5) v. Thay vì xuất ra các giá trị kỹ thuật số **Thấp** (0v) hoặc **Cao** (5v). Biểu đồ tín hiệu PWM trông giống như trong sơ đồ bên dưới.



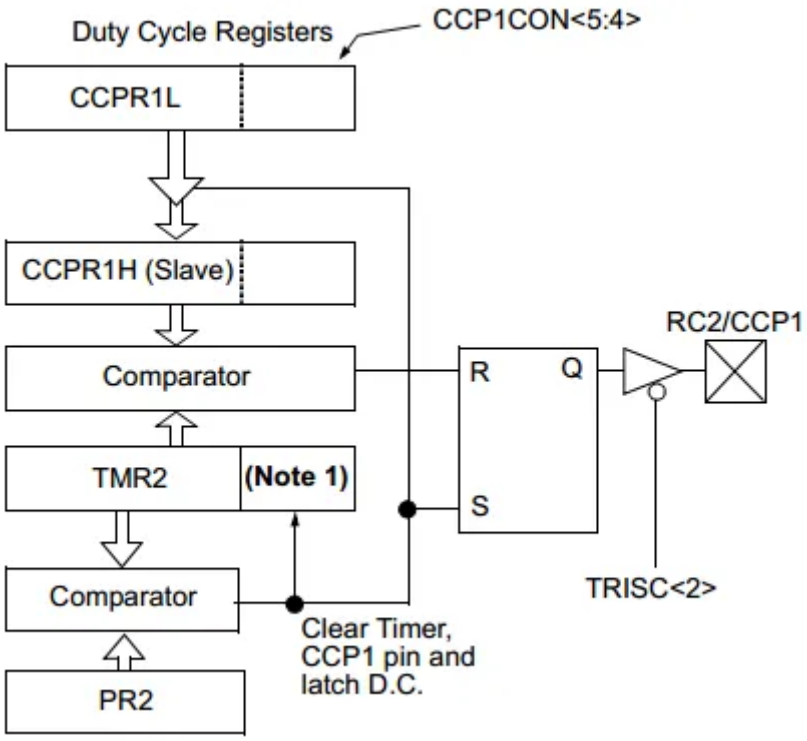
Keyword chính cho tín hiệu này nắm bắt hai tính năng: (Tần số &Chu kỳ nhiệm vụ). Và chúng ta sẽ thảo luận chi tiết hơn sau đây.

Ở Chế độ PWM, chân CCPx (CCP1 = RC2, CCP2 = RC1) tạo ra tín hiệu đầu ra PWM có độ phân giải lên đến 10 Bit. Vì chân CCP1 được ghép với chốt dữ

liệu PORTC, chân RC2 phải được định cấu hình làm chân đầu ra bằng cách xóa bit TRISC2. Và điều tương tự cũng xảy ra với chân CCP2 (RC1).

Sơ đồ chế độ PWM

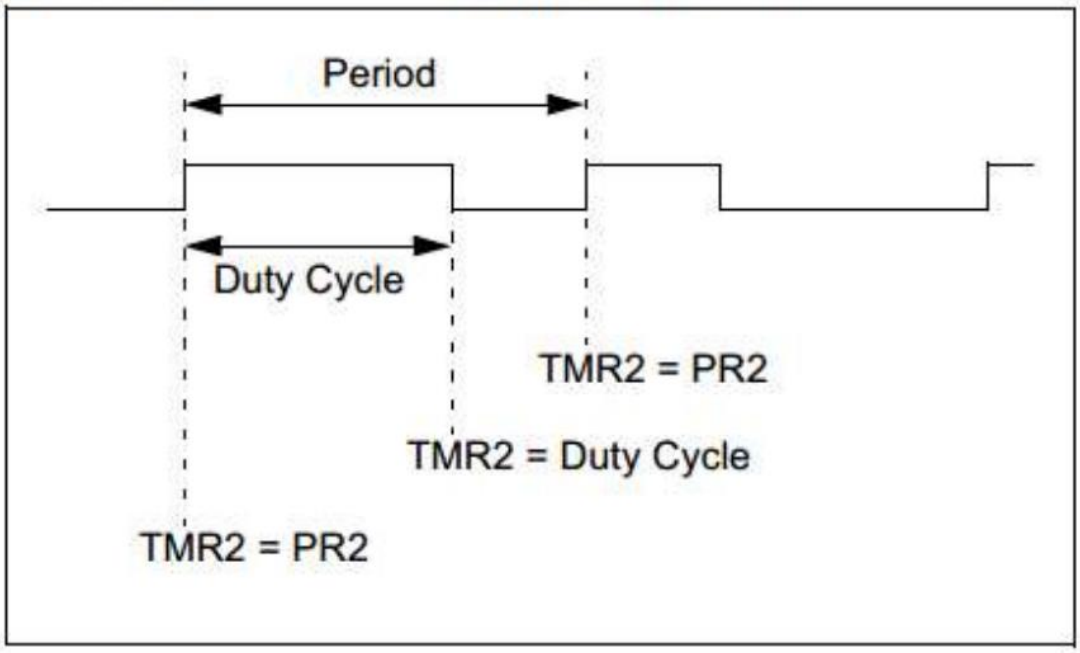
Đây là sơ đồ logic (Khối) cho chế độ PWM như được tìm thấy trong biểu dữ liệu (8.3 trang-67) (datasheet)



- Khối PWM gồm có 2 mạch so sánh: mạch so sánh 8 bit với mạch so sánh 10 bit. Mạch so sánh 8 bit sẽ so sánh giá trị đếm của timer2 với giá trị của thanh ghi PR2 (period register),giá trị trong timer2 tăng từ giá trị đặt trước cho đến khi bằng giá trị của PR2 thì mạch so sánh sẽ set đầu ra của mạch flip-flop-RS RS làm ngõ ra CCPx lên mức 1. Đồng thời nạp giá trị 10 bit từ thanh ghi CCPRxL 10 sang thanh ghi CCPRxH, timer2 bị reset và bắt đầu đếm lại cho đến khi giá trị của timer2 bằng giá trị của CCPRxH thì mạch so sánh sẽ reset flip flop RS làm ngõ ra CCPx về mức 0. Quá trình này lặp lại.

Các tham số của PWM

FIGURE 8-4: PWM OUTPUT



- **Chu kì xung(pwm period)**

PWM period = [(PR2)+1]*4*T_{OSC}*(giá trị bộ chia tần số của TMR2).

TOSC : là chu kỳ của thạch anh tạo dao động=20M
Khi giá trị của timer 2 (TMR2) bằng giá trị của thanh ghi PR2 thì 3 sự kiện theo sau sẽ xảy ra:
· Thanh ghi TMR2 bị xóa
· Tín hiệu ngõ ra CCPx lên mức 1, ngoại trừ hệ số chu kỳ bằng 0% thì CCPx vẫn ở mức 0.
· Hệ số chu kỳ PWM được chuyển từ thanh ghi CCPRxL sang thanh ghi CCPRxH

- **Độ rộng xung (pwm duty cycle)**
  Hệ số chu kỳ được thiết lập bởi giá trị lưu trong thanh ghi 10 bit gồm CCPRxL 8 bit và 2 bit còn lại là
  bit thứ 4 và thứ 5 ở trong thanh ghi CCPxCON – kí hiệu là CCPxCON<5:4>.
  Giá trị của hệ số chu kỳ là 10 bit nên có thể thay đổi từ 0 đến 1023 tạo ra 1024 cấp giá trị điều khiển.
  Giá trị 10 bit thì 8 bit có trọng số lớn lưu trong thanh ghi CCPRxL và 2 bit còn lại có trọng số thấp thì ở CCPxCON<5:4>.
  Hệ số chu kỳ của PIC16F877A được tính theo công thức:

PWM duty cycle = (CCPRxL:CCPxCON<5:4>)*T_{OSC}*(giá trị bộ chia tần số TMR2)

- Thanh ghi CCPxCON: (CCP1CON VÀ CCP2CON)

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: chưa được thực hiện, cài đặt =0
bit 5-4 CCPxX:CCPxY: PWM Least Significant bits
2bit MSB chứa giá trị tính độ rộng xung(duty cycle) của pwm. (8bit còn lại chứa trong thanh ghi CCPRxL)
bit 3-0 CCPxM3:CCPxM0: CCPx chọn chế độ hoạt động
0000 = Capture/Compare/PWM disabled
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 isunaffected)
1011 = Compare mode, trigger special event (CCPxIF bit is set, CCPx pin is unaffected); CCP1resets TMR1; CCP2 resets TMR1 and starts an A/D conversion (if A/D module is enabled)
11xx = PWM mode
ở đây, ta dùng chế độ pwm nên pwm mode= 11xx. Nghĩa là ép buộc CCPxM3=1;CCPxM2=1; còn CCPxM1,CCPxM0 có thể thay đổi tùy ý mà không ảnh hưởng tới chế độ pwm

- **Thanh ghi T2CON**

ở chế độ pwm, ta cần chú ý tới giá trị bộ chia tần số prescaler của Timer2

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

Bit7: không quan tâm và mặc định bằng 0
bit 6-3 TOUTPS3:TOUTPS0: Timer2 Output Postscale Select bits: các bit này chọn tỷ số chia tần cho postscale
0000 = 1:1 Postscale
0001 = 1:2 Postscale
0010 = 1:3 Postscale
•••
1111 = 1:16 Postscale
bit 2 :**TMR2ON**: chọn chế độ timer
1 = Timer2 is on
0 = Timer2 is off

bit 1-0 :T2CKPS1:T2CKPS0: chọn bộ chia tần số Prescaler
00 = Prescaler is 1
01 = Prescaler is 4
1x = Prescaler is 16

Cấu hình mô-đun CCP1 cho hoạt động PWM

Bước 1 - Định cấu hình mô-đun CCP cho hoạt động PWM

- Như đã đề cập trong biểu dữ liệu trong mô tả của sổ đăng ký CCP1CON. Viết 11xx vào CCP1M0: CCP1M3 4-Bits chọn chế độ hoạt động PWM. Lưu ý rằng xx có nghĩa là Bit0,1 không quan tâm! đó là lý do tại sao chúng sẽ bỏ qua chúng trong khi viết mã.

Bước 2 - Định cấu hình chân CCP1 (RC2) thành chân đầu ra bằng cách xóa bit TRISC2 tương ứng

- Chân CCP1 / RC2 phải được định cấu hình làm chân đầu ra.

Bước 3 - Xác định tần số PWM và nhận giá trị thời gian PWM

- Giả sử chúng ta cần có tín hiệu PWM với tần số 5kHz. Điều này có nghĩa là khoảng thời gian phải là = 1 / F = 1/5000 = 200µs

Bước 4 - Sử dụng period PWM, Tính toán giá trị mà chúng ta sẽ tải vào **Thanh ghi PR2**

- Bước này liên quan đến việc chọn một giá trị cho bộ định dạng trước Timer2 trước khi xác định giá trị của thanh ghi PR2. Đối với bước này, chúng ta sẽ sử dụng phương trình sau
- Bây giờ có PWM, chữ $T_{Osc} = 1/F_{Osc}$và chúng ta sẽ chọn giá trị ban đầu cho Timer2Prescaler (giả sử 1 cho tỷ lệ 1:1). Nó sẽ dễ dàng giải quyết để **PR2** có được giá trị của nó. Nguyên tắc chung là thanh ghi PR2 là một thanh ghi 8 bit có nghĩa là giá trị của nó nằm trong khoảng từ (0 đến 255). Kết quả của bạn không bao giờ được vượt quá giới hạn 255 này. Nếu không, bạn nên thử một giá trị khác cho Timer2Prescaler. Cuối cùng, sau khi nhận được giá trị PR2, chúng ta nên chuyển số này sang thanh ghi PR2.

Bước 5 - Đặt prescaler của Timer2

- Điều chỉnh giá trị prescaler Timer2 để khớp với giá trị mà bạn đã sử dụng ở bước trước. Điều này được thực hiện bằng cách ghi vào thanh ghi T2CON T2CKPSx bit.

Bước 6 - Bước Chu trình nhiệm vụ PWM, bằng cách ghi vào thanh ghi CCPR1L > CCP1CON<5: 4> Bits

- Để đặt hoặc thay đổi Chu kỳ nhiệm vụ của tín hiệu đầu ra PWM, bạn nên tính toán và ghi giá trị DC vào thanh ghi bộ đệm 10 Bit bao gồm CCPR1L: CCP1CON<5: 4>. CCPR1L chứa tám MSB và CCP1CON<5: 4> chứa hai LSb. Giá trị 10 bit này được thể hiện bằng CCPR1L:CCP1CON<5:4>.
- Phương trình sau đây được sử dụng để tính chu kỳ nhiệm vụ PWM kịp thời:

- Giả sử bạn sẵn sàng nhận 40% DC cho đầu ra PWM của mình. Sau đó, bạn nên nhân 0,40 với PWM. Để có được PWM kịp thời. Bây giờ, thay thế cho PWM, T_{Osc}, $TMR2_{prescaler}$ trong phương trình trước và giải cho giá trị 10-Bit giữa ( )
- Sau khi nhận được chu kỳ nhiệm vụ 10-Bit từ các phép tính của bạn, chỉ cần ghi giá trị này vào bộ đệm CCPR1L: CCP1CON<5: 4>

Bước 7 - BẬT Hẹn giờ2

- Đừng quên BẬT mô-đun Timer2! rõ ràng đó là con ngựa làm việc của hệ thống của chúng tôi. Tuy nhiên, nó tình cờ phát hiện ra rằng bạn đã bỏ lỡ việc làm như vậy và đó là lý do tại sao không có gì hoạt động cả.

Code mẫu

```
long pwm = 8000;
void PWM_init(){
   PR2 = (_XTAL_FREQ/(pwm*4*PRESCALE)) - 1; // CHON GIA TRI CHO PR2 TIMER2
   CCP1CON |= 0X0C; // KICH CCP1 HOAT DONG VOI CHE DO PWM
   T2CON   |= 0X05; // KICH TIMER2 ON VA TAN SO CHIA LA 4 (PHU THUOC VAO PRESCALE MINH CHON)
   TRISC    = 0b000000000; // CHAN CCP1
}
void PWM_DUTY(unsigned int duty)
{
   if(duty <1024)
   {
      duty = ((float)duty/1023)*(_XTAL_FREQ /(pwm*PRESCALE));
      CCP1X = duty & 1;
      CCP1Y = duty & 2;
      CCPR1L = duty>>2;
   }
}
```

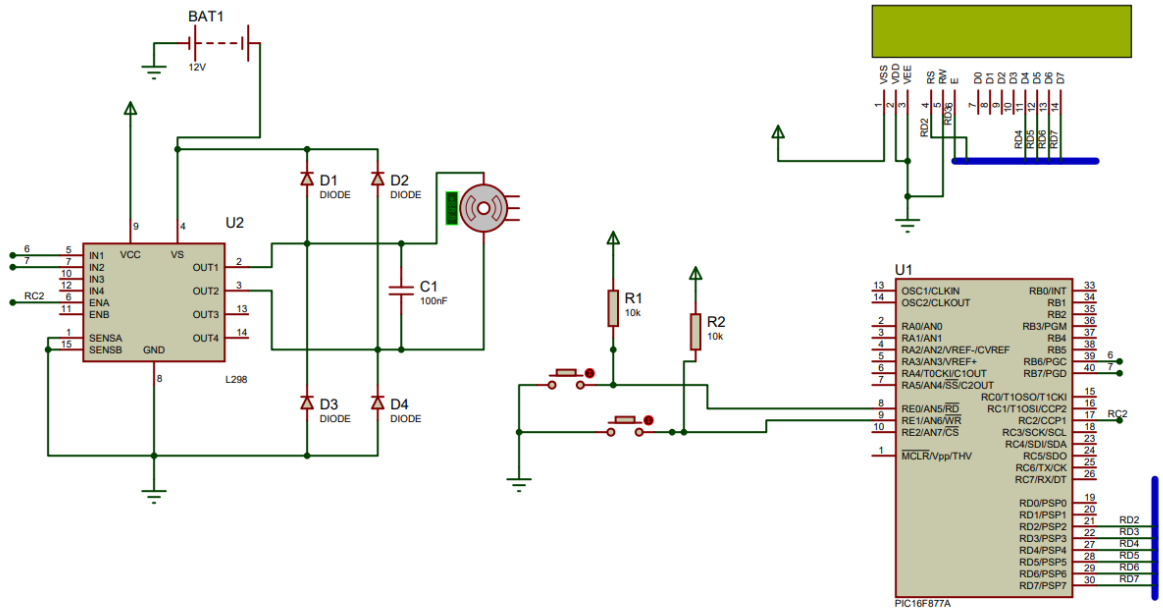
Example

```c
#include <stdio.h>
#include <stdlib.h>
#define _XTAL_FREQ 20000000
#define RS RD2
#define EN RD3
#define D4 RD4
#define D5 RD5
#define D6 RD6
#define D7 RD7
#include <xc.h>
#include "lcd.h"
// BEGIN CONFIG
#pragma config FOSC = HS // Oscillator Selection bits (HS oscillator)
#pragma config WDTE = OFF // Watchdog Timer Enable bit (WDT enabled)
#pragma config PWRTE = OFF // Power-up Timer Enable bit (PWRT disabled)
#pragma config BOREN = ON // Brown-out Reset Enable bit (BOR enabled)
#pragma config LVP = OFF // Low-Voltage (Single-Supply) In-Circuit Serial
Programming Enable bit (RB3 is digital I/O, HV on MCLR must be used for
programming)
#pragma config CPD = OFF // Data EEPROM Memory Code Protection bit (Data
EEPROM code protection off)
#pragma config WRT = OFF // Flash Program Memory Write Enable bits (Write
protection off; all program memory may be written to by EECON control)
#pragma config CP = OFF // Flash Program Memory Code Protection bit (Code
protection off)
//END CONFIG
//
#define PRESCALE 16
#define plus RE0
#define sub RE1

//
int duty_counter;
float value_pwm ;
int level,speed;
char s[30];
char x[30];
//
long pwm = 8000; // cai tan so PWM o day
void PWM_init(){
    PR2 = (_XTAL_FREQ/(pwm*4*PRESCALE)) - 1; // CHON GIA TRI CHO PR2 TIMER2
    CCP1CON |= 0X0C; // KICH CCP1 HOAT DONG VOI CHE DO PWM
    T2CON |= 0X07; // KICH TIMER2 ON VA TAN SO CHIA LA 16 (PHU THUOC VAO
PRESCALE MINH CHON)
    TRISC= 0b000000000; // CHAN CCP1
}
void PWM_DUTY(unsigned int duty)
{
```

```
    if(duty <1024)
    {
        duty = ((float)duty/1023)*(_XTAL_FREQ /(pwm*PRESCALE));
        CCP1X = duty & 1;
        CCP1Y = duty & 2;
        CCPR1L = duty>>2;
    }
}//=======================================
void setup_interrupt(){
    INTCONbits.GIE   = 1;
   INTCONbits.PEIE       =1;
    PIE1bits.TMR1IE      =1;

    T1CONbits.TMR1CS     = 0;
    T1CONbits.T1CKPS1    = 1;
    T1CONbits.T1CKPS0    = 0;
    T1CONbits.T1SYNC     = 0;
    T1CONbits.TMR1ON     = 1;
   TMR1                 = 535;
}

//================
void main(void) {

    ADCON1 = 0x07; // set up cho AN0-7 la digital
    TRISE = 0b111;
    TRISB= 0x00;
    PORTB = 0x40;
    TRISC = 0x00;
    TRISD = 0x00;
    setup_interrupt();
    Lcd_Init();
    PWM_init();
    duty_counter =0;
    while(1){
        PWM_DUTY(duty_counter);
      Lcd_Set_Cursor(1,1);
      Lcd_Write_String("Muc quat :");
      speed = duty_counter/341;
      sprintf(s,"%3u",(speed));
         Lcd_Set_Cursor(1,14);
      Lcd_Write_String(s);
      Lcd_Set_Cursor(2,5);
      Lcd_Write_String("Nhom ABC XYZ ");

    }
}
```

```
//============
void __interrupt() timer1(void){

     if(PIR1bits.TMR1IF == 1)
{
      PIR1bits.TMR1IF = 0;
      TMR1 = 535;
      if(plus != 1)
        {
          if(duty_counter <683)
            duty_counter = duty_counter + 341;


        }
        if(sub != 1 )
        {
          if(duty_counter >340)
          duty_counter = duty_counter - 341;
        }
}
}
```

Simulation



Tài liệu tham khảo

PIC Microcontroller PWM Tutorial using MPLAB and XC8 (circuitdigest.com)