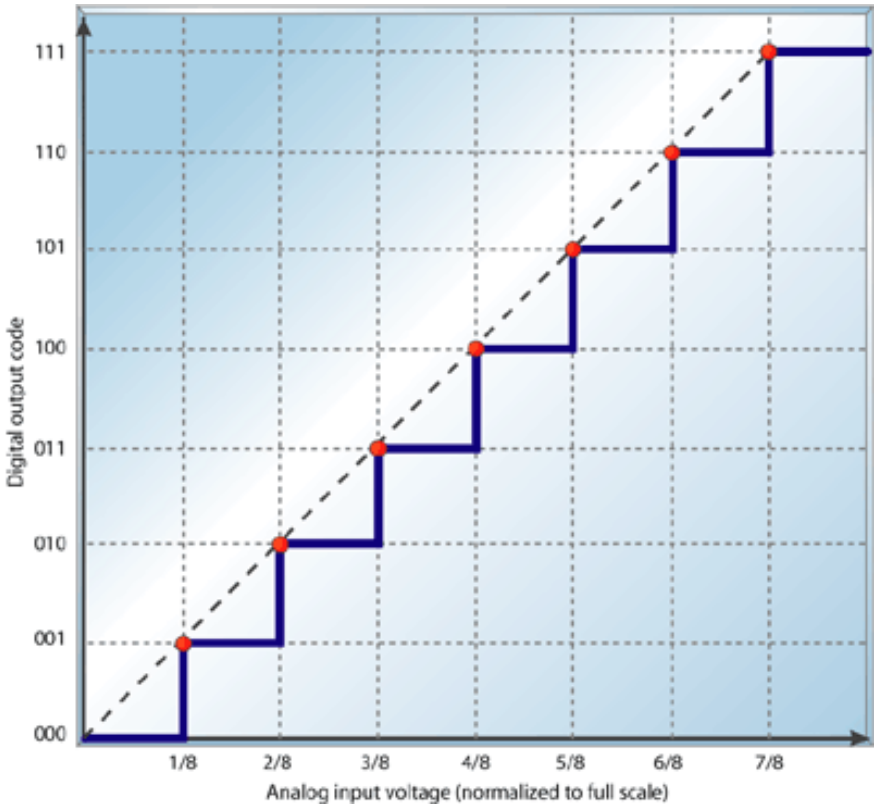
Bài 04

ADC & PWM

(ADC: Analog to Digital Converter
PWM: Pulse Width Modulation)

I. ADC:

❖ *Phân biệt độ phân giải và tần số lấy mẫu*



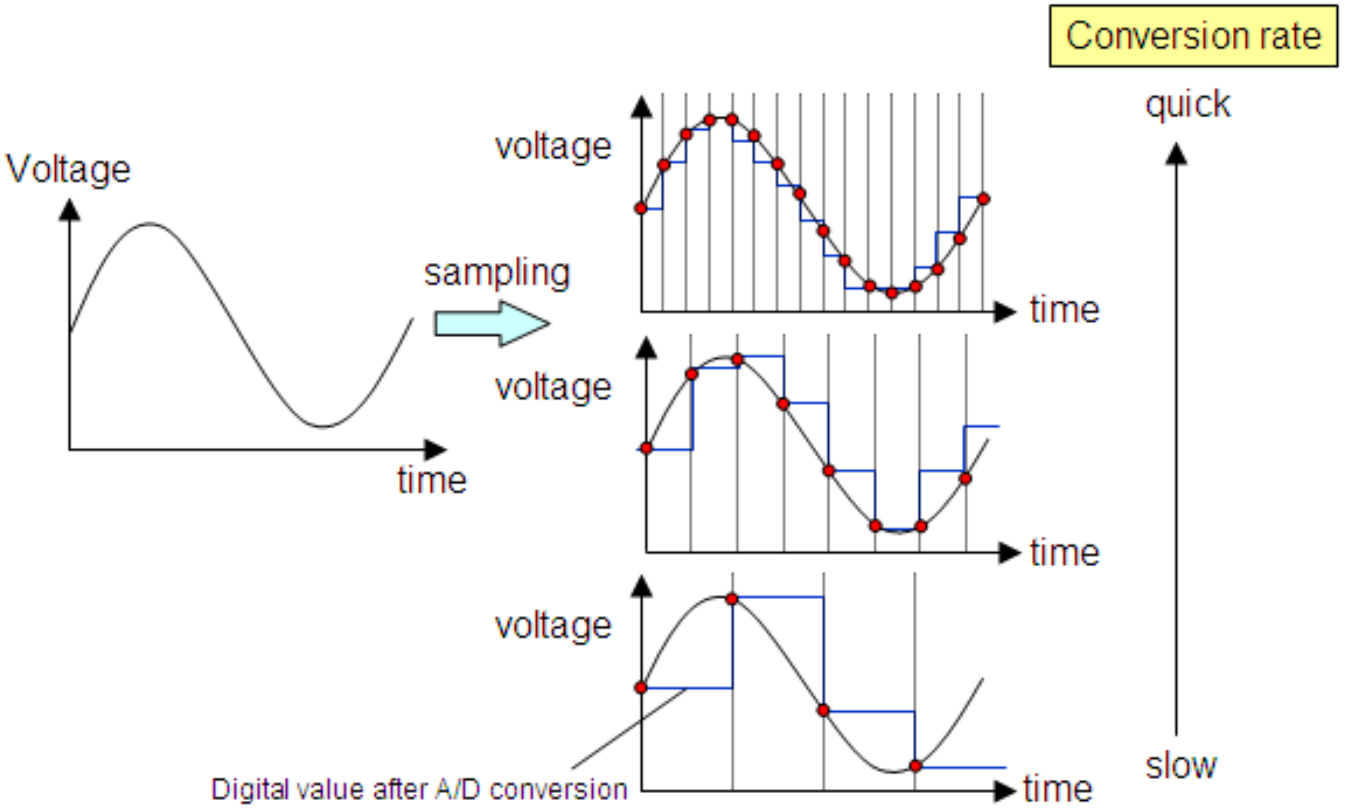
$$N = \frac{V_i * (2^n - 1)}{V_{ref}}$$

N: giá trị số của ngõ ra ADC

V_i: điện áp analog tại thời điểm lấy mẫu

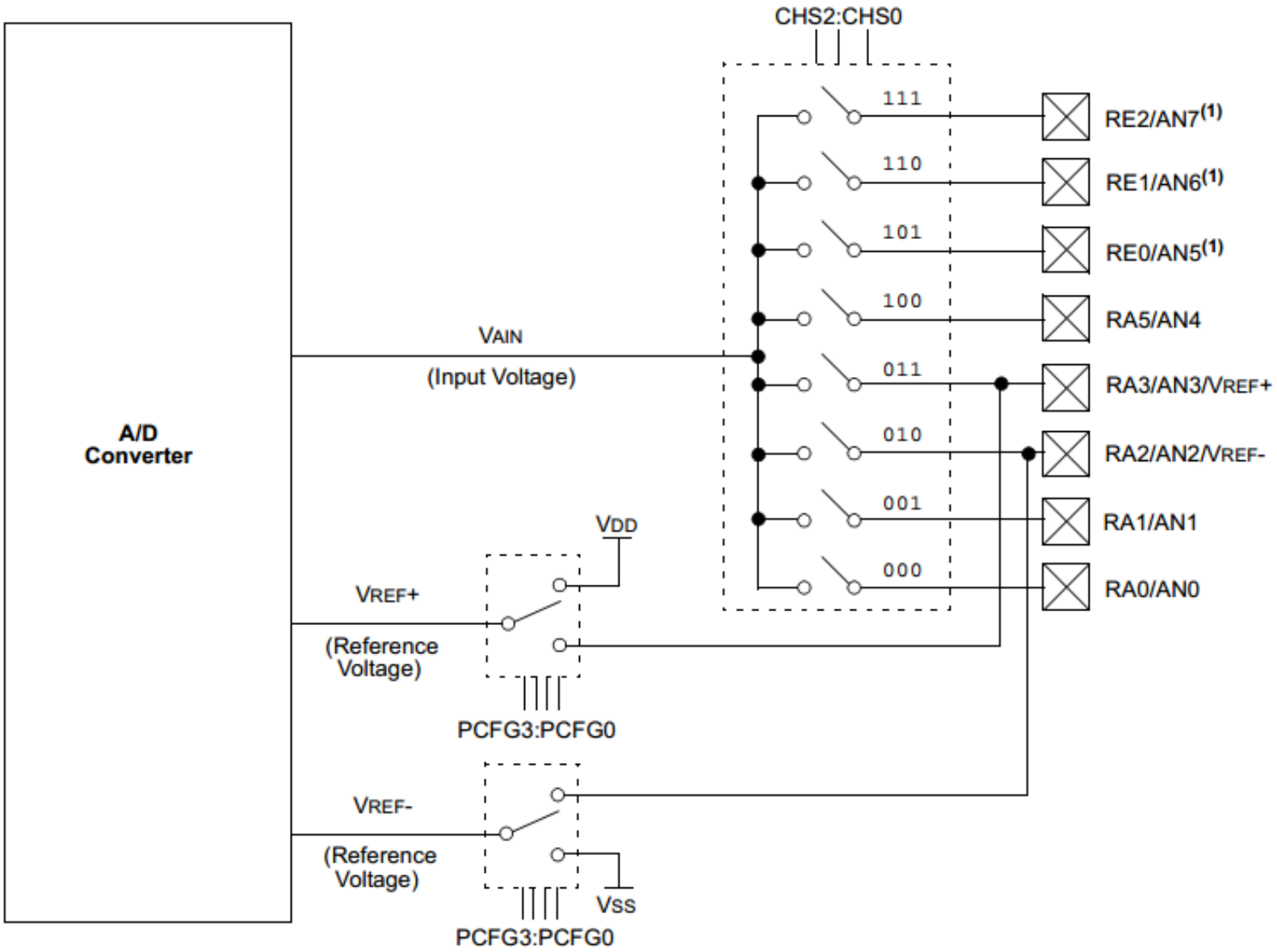
n: số bit (độ phân giải)

V_{ref} : điện áp chuẩn ($\geq V_i$)



- Tần số lấy mẫu càng cao thì tín hiệu analog sau khi được hồi giải mã càng trung thực.
- Một thông số khác của bộ ADC là: tốc độ biến đổi analog sang digital.

- Cấu trúc ADC của PIC16F877A



- **Các lệnh CCS liên quan ADC:**

1. #DEVICE ADC = 8/10/16
2. Setup_adc(ADC_CLOCK_INTERNAL)
3. Setup_adc_ports(**value**)
4. Set_adc_channel(0-7)
5. Digital_Value = read_adc()

Những giá trị của value:

ALL_ANALOG // A0 A1 A2 A3 A5 E0 E1 E2

AN0_AN1_AN2_AN4_AN5_AN6_AN7_VSS_VREF

AN0_AN1_AN2_AN3_AN4

AN0_AN1_AN2_AN4_VSS_VREF

AN0_AN1_AN3

AN0_AN1_VSS_VREF

AN0_VREF_VREF // A0 VRefh=A3 VRefl=A2

Ví dụ: LM35 là cảm biến nhiệt độ có ngõ ra là 10mV/1^{0}C

Tính toán độ phân giải:

Các ngõ ra điện áp tham chiếu: $V_{REF-}=V_{SS}=0V$; $V_{REF+}=V_{DD}=5V$

Nên độ phân giải (Step Size - giá trị điện áp ngõ vào để ngõ ra thay đổi 1 đơn vị) là:

$$SS= V_{REF+}- V_{REF-} / (2^{10}-1) = 5000mV / 1023 = 4,887mV$$

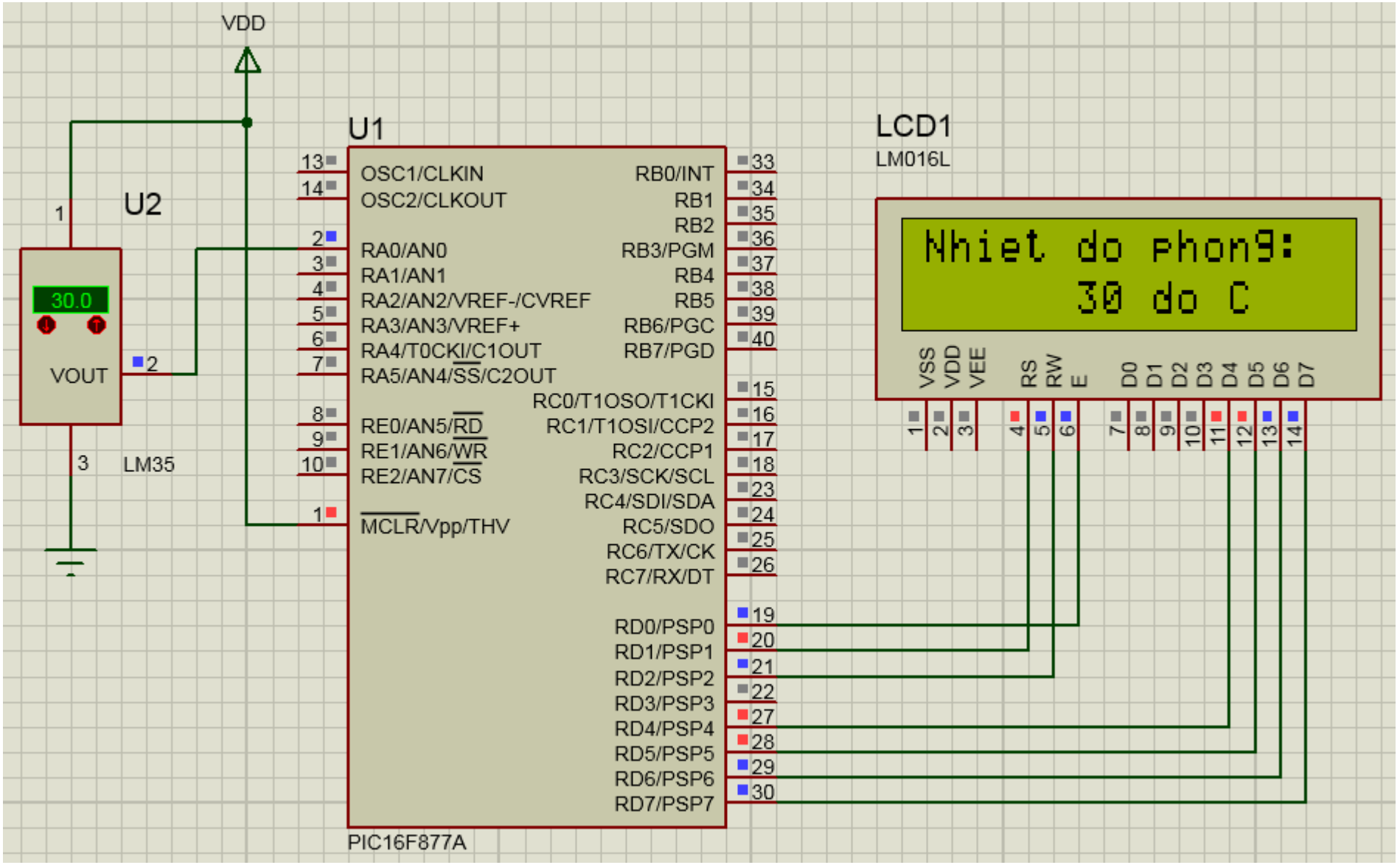
Thông số 2^{10} là do ADC 10 bit

Điện áp toàn giai (Full Scale): FS = SS x 1023 = 4,887mV x 1023 = 5000mV = 5V

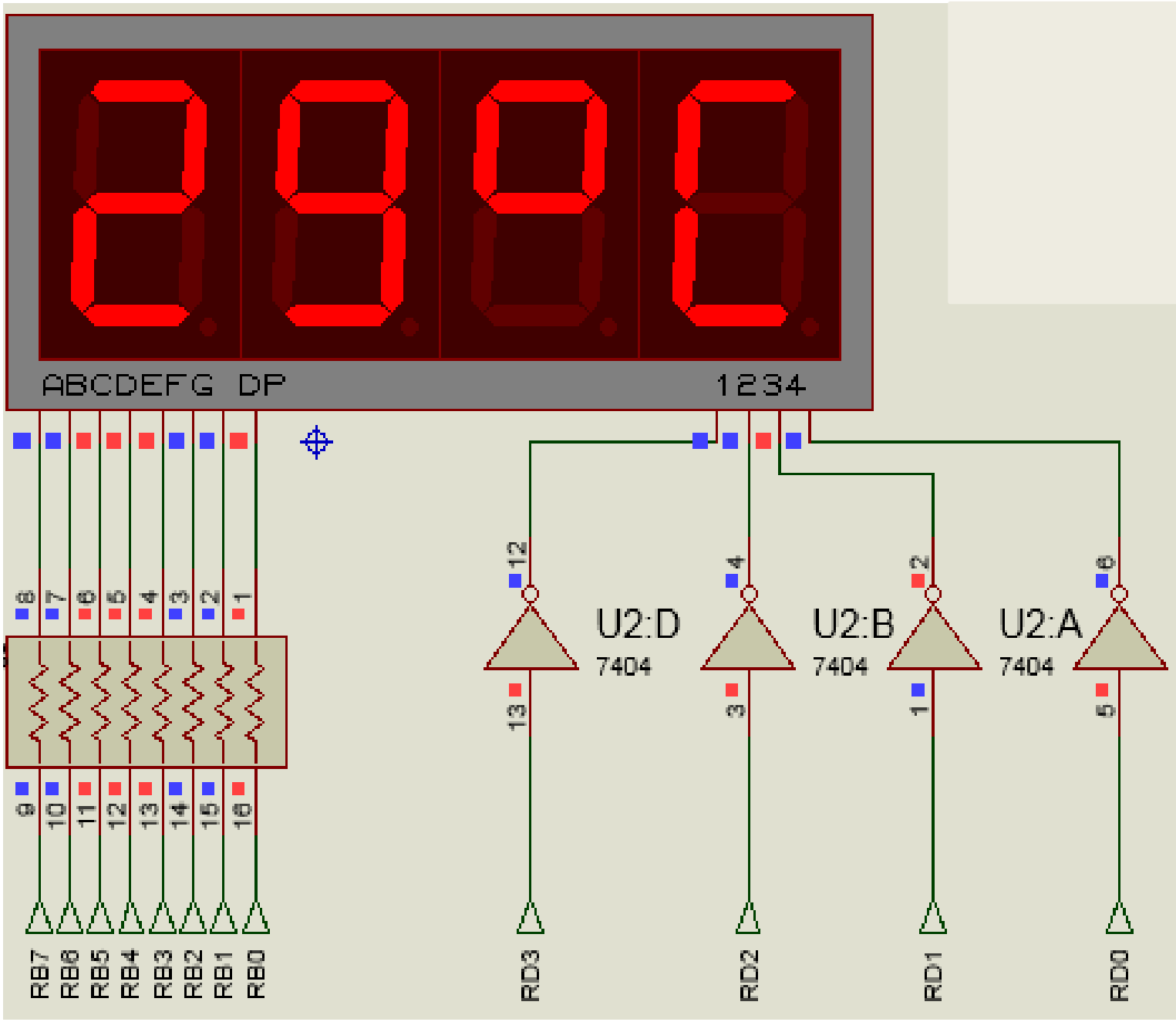
Độ phân giải ADC là 4,887mV không tương thích với độ phân giải của cảm biến LM35 bằng 10mV.

Vậy ta có tỷ lệ chênh lệch: 10mV / 4,887mV = 2,046.

Để có kết quả hiển thị đúng thì lấy kết quả chuyển đổi chia cho tỷ lệ chênh lệch.


VDD
U1
LCD1
LM016L
U2
30.0
VOUT
LM35
PIC16F877A
OSC1/CLKIN
OSC2/CLKOUT
RA0/AN0
RA1/AN1
RA2/AN2/VREF-/CVREF
RA3/AN3/VREF+
RA4/T0CKI/C1OUT
RA5/AN4/SS/C2OUT
RE0/AN5/RD
RE1/AN6/WR
RE2/AN7/CS
MCLR/Vpp/THV
RB0/INT
RB1
RB2
RB3/PGM
RB4
RB5
RB6/PGC
RB7/PGD
RC0/T1OSO/T1CKI
RC1/T1OSI/CCP2
RC2/CCP1
RC3/SCK/SCL
RC4/SDI/SDA
RC5/SDO
RC6/TX/CK
RC7/RX/DT
RD0/PSP0
RD1/PSP1
RD2/PSP2
RD3/PSP3
RD4/PSP4
RD5/PSP5
RD6/PSP6
RD7/PSP7
Nhiet do phong:
30 do C
VSS
VDD
VEE
RS
RW
E
D0
D1
D2
D3
D4
D5
D6
D7

```
#include <16F877A.h>
#device adc=10
#FUSES NOWDT, HS, NOPUT, PROTECT, NODEBUG, NOBROWNOUT, NOLVP
#use delay(clock=8000000)
#include <lcd.c>

//=================================================================
void main()
{
   INT temp;
   Setup_adc(ADC_CLOCK_INTERNAL);
   Setup_adc_ports(AN0);
   Set_adc_channel(0);

   //---------------------------------------
   lcd_init () ;
   temp = 0;
   lcd_gotoxy (1, 1);
   printf (lcd_putc, "Nhiet do phong:");

   //-------------------------
   WHILE (TRUE)
   {
      temp = read_adc () / 2.046;
      lcd_gotoxy (6, 2);
      printf (lcd_putc, " %02u do C", temp) ;
   }
}//end main
```

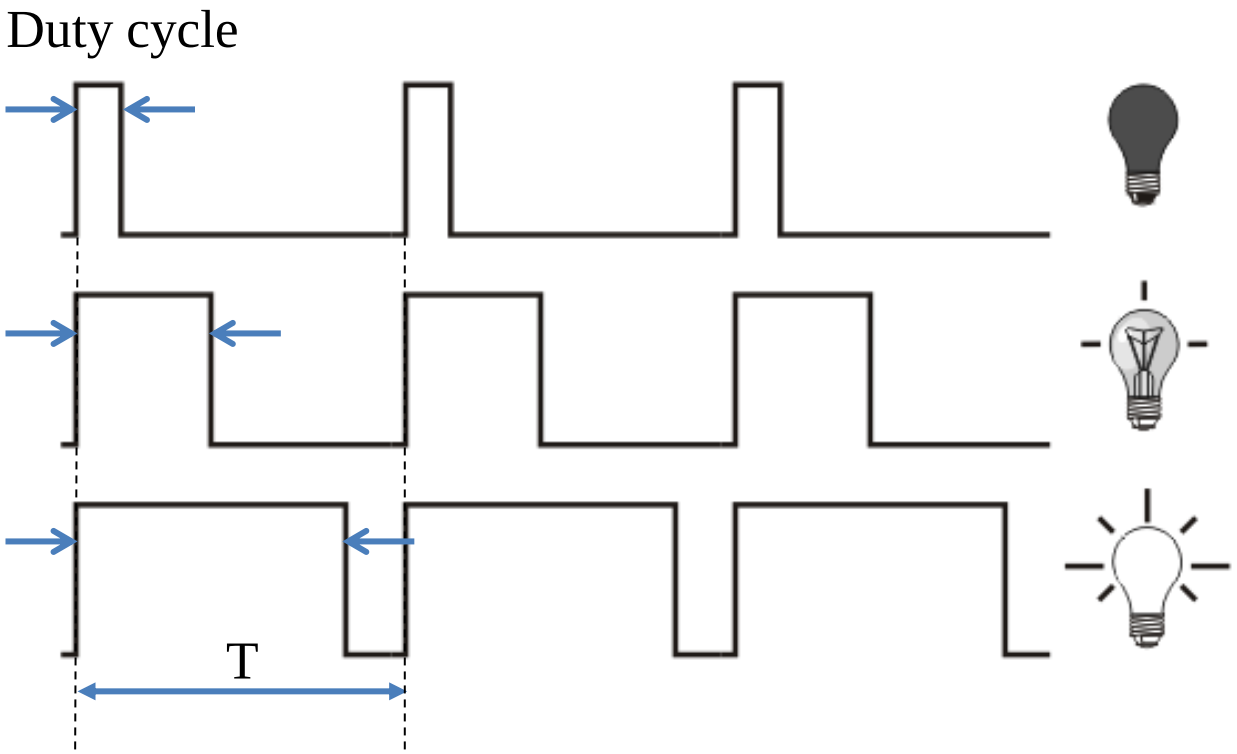

ABCDEFG DP
1234
U2:D
7404
U2:B
7404
U2:A
7404
RB7
RB6
RB5
RB4
RB3
RB2
RB1
RB0
RD3
RD2
RD1
RD0

```
#include <16F877A.h>
#device ADC=10// ADC = 8/10/16
#FUSES NOWDT, HS, NOPUT, PROTECT, NODEBUG, NOBROWNOUT, NOLVP
#use delay(clock=8000000)

CONST INT8 a[10] = {0x03, 0x9f,0x25,0x0d,0x99,0x49,0x41,0x1f,0x01,0x09 };
int8 chuc,dvi;
int16 temp;
// Chuong trinh Ngat timer0
#INT_TIMER0
void qled()
{
   Output_d (0xff); // tat các led
   Output_b (a[chuc]); output_low (PIN_D3); delay_ms (3); // "chuc: nhiet do
   Output_d (0xff); // tat các led
   Output_b (a[dvi]); output_low (PIN_D2); delay_ms (3); // "Don vi: nhiet do
   Output_d (0xff); // tat các led
   Output_b (0x39); output_low (PIN_D1); delay_ms (3); // "dau: o
   Output_d (0xff); // tat các led
   Output_b (0x63); output_low (PIN_D0); delay_ms (3); // "C
}
```

```
//-------CHUONG TRÌNH CHÍNH---------
Void main()
{
   // cho phep ngat timer 0
   enable_interrupts (GLOBAL) ;
   enable_interrupts (INT_TIMER0) ;
   Setup_timer_0 (RTCC_DIV_4|RTCC_INTERNAL) ;
   set_timer0(200);

   // Cau hinh ADC
   setup_adc (ADC_CLOCK_INTERNAL) ;
   setup_adc_ports (AN0) ;
   set_adc_channel (0); // 0 -- >  7
   WHILE (1)
   {
      set_timer0(200);
      temp = read_adc () / 2.046;
      chuc = temp / 10;
      dvi = temp % 10;
   }

}
```

II. PWM (Pulse Width Modulation):

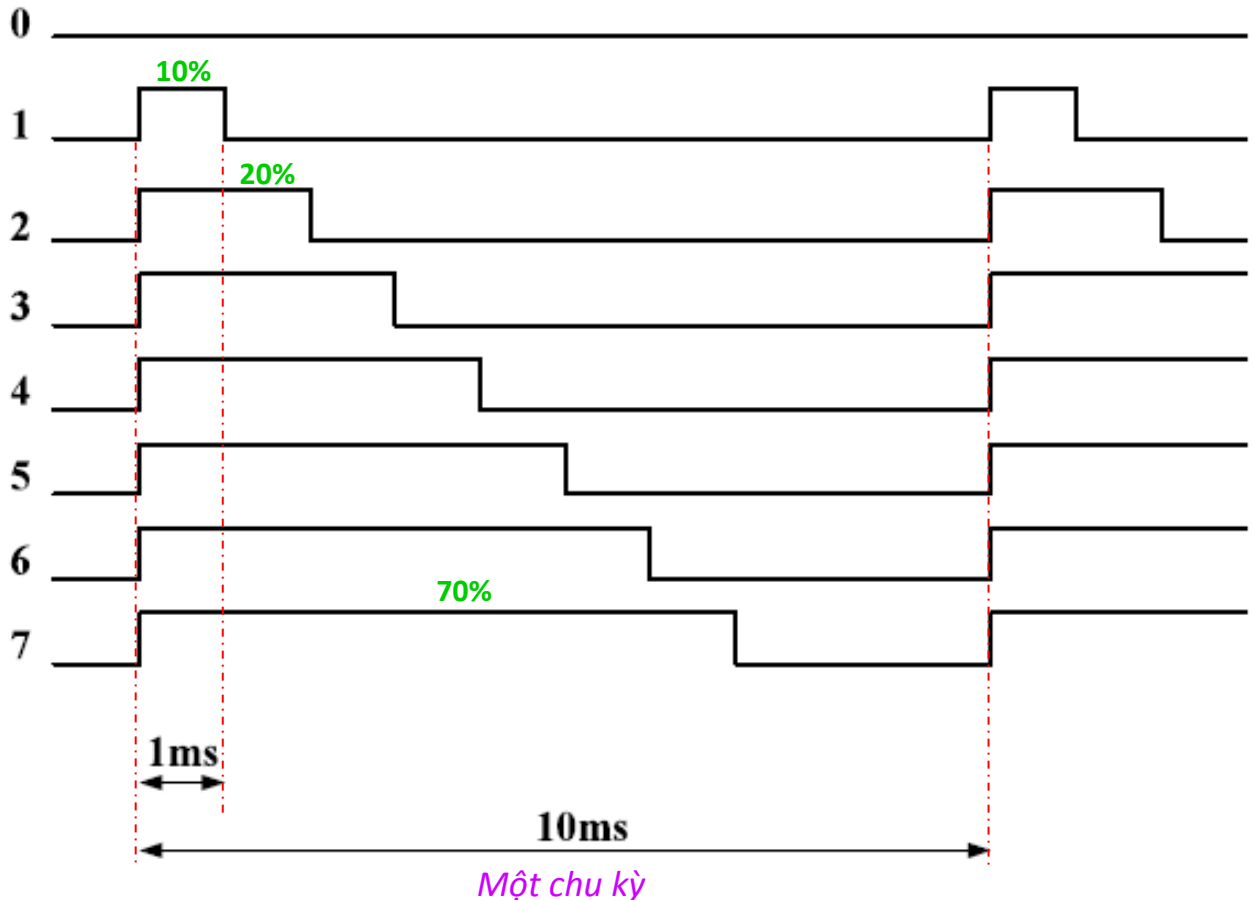
❖ Tần số không thay đổi, chu kì duty thay đổi theo mục đích điều khiển



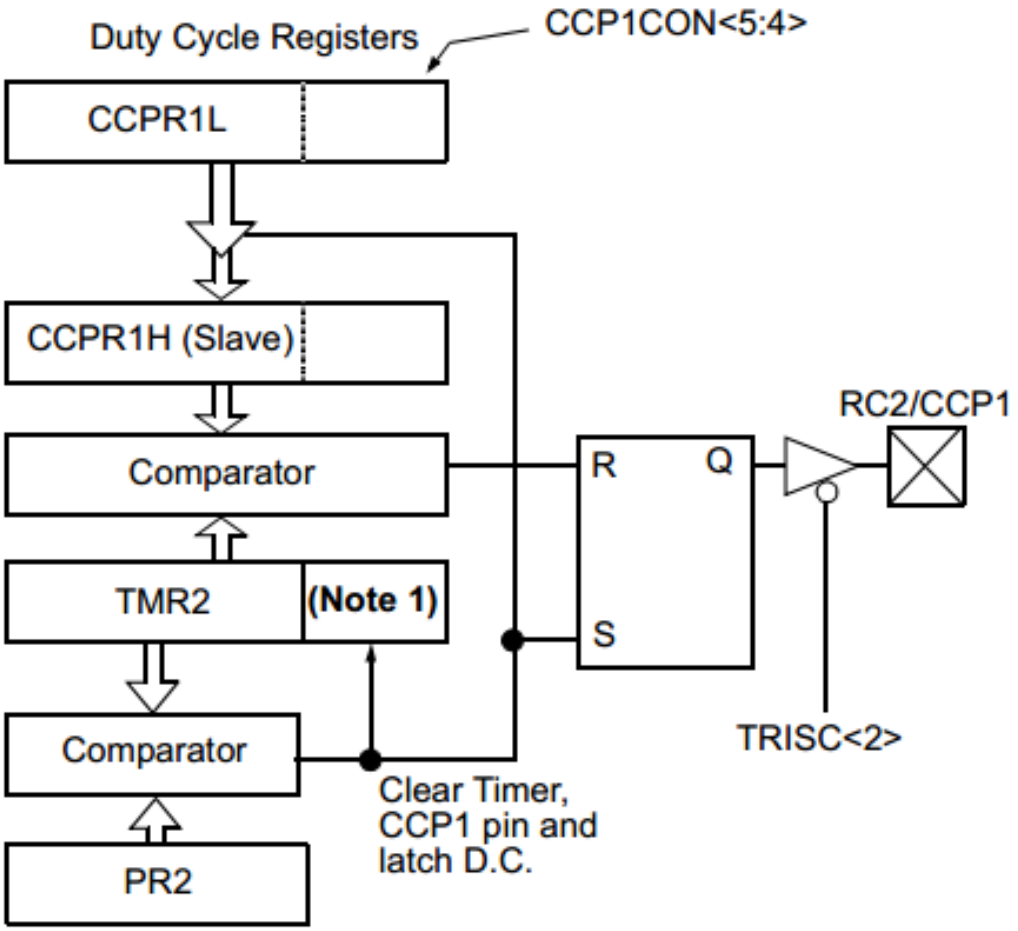
Tần số: $f = \frac{1}{T}$

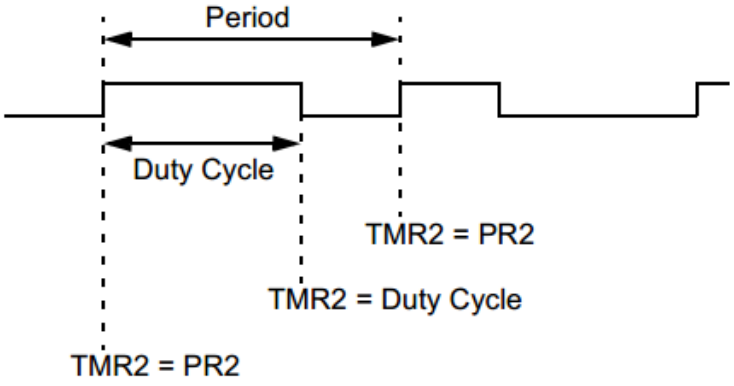
❖ Nguyên lý điều chế độ rộng xung PWM:

Nguyên lý điều chế độ rộng xung là mạch tạo ra xung vuông có chu kỳ là hằng số nhưng hệ số công tác (hệ số chu kỳ - duty cycle) có thể thay đổi được. Sự thay đổi hệ số chu kỳ làm thay đổi điện áp trung bình hoặc dòng điện trung bình. Từ đó, dùng để điều khiển các tải như động cơ DC (làm thay đổi tốc độ động cơ), điều khiển bóng đèn (làm thay đổi độ sáng bóng đèn), …



❖ **PIC16F877A có 2 bộ PWM, ngõ ra của PWM tại chân RC2 (CCP1) và RC1 (CCP2)**





❖ Các lệnh liên quan PWM:

1. setup_ccp1(CCP_PWM); // Đặt CCP1 là PWM chân RC2
2. setup_ccp2(CCP_PWM); // Đặt CCP2 là PWM chân RC1
3. setup_timer_2(PVTMR2, PR2, Postscale); // Đặt timer_2
4. set_pwm1_duty(value); // Đặt duty_cycle cho PWM1
5. set_pwm2_duty(value); // Đặt duty_cycle cho PWM2

Tính toán:

setup_timer_2(PV$_{TMR2}$, PR2, Postscale);

set_pwm1_duty(value);

► **Tính setup_timer_2 để có chu kỳ PWM mong muốn:**

Tần số thạch anh: Fosc nên có chu kỳ là Tosc=1/Fosc

Chu kỳ PWM: T_{PWM}=[PR2+ **Postscale**]*4*T_{osc}*PV_{TMR2} ***Tần số PWM:*** $F_{PWM}=1/T_{PWM}$

PV_{TMR2}: có 3 giá trị chia **1**, chia **4** và chia **16**

PR2: có giá trị 8 bit *(0,1,2…255)* ---> đây cũng là giá trị đặt period cho timer2

Postscale: Timer 2 tràn bộ đếm **1** *(có thể từ **1** đến **16**)* lần thì bit báo tràn sẽ báo 1 lần.

► **Tính hệ số chu kỳ:** Duty_cycle= $T_{PWM}/(T_{osc}*PV_{TMR2})$

VD: Cho thạch anh 20MHz, có chu kỳ PWM là T_{PWM}=0.8ms hay tần số PWM là F_{PWM}=1.25KHz

Giải: Ta chọn PV_{TMR2}=16 theo T_{PWM}=[PR2+ **Postscale**]*4*T_{osc}*PV_{TMR2}

=> [PR2+ **Postscale**] = 250 ---> chọn **Postscale=1** => PR2 = 249

setup_timer_2(T2_DIV_BY_16,249,1);

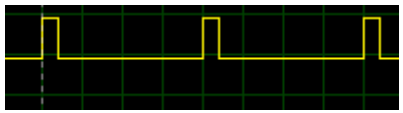
Duty_cycle= T_{PWM} / (T_{osc}*PV_{TMR2}) = 800.000ns / (50ns*16) = 1000 <---> 100%

Vậy 500 <---> 50%

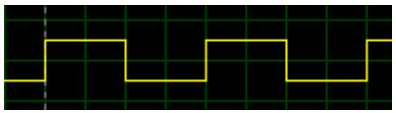
set_pwm1_duty(500); // 50%

```
#include <16F877A.h>
#FUSES NOWDT, HS, NOPUT, PROTECT, NODEBUG, NOBROWNOUT, NOLVP
#use delay(clock=20000000)


//=====================================================

UNSIGNED INT16 TOC_DO;
void main()
{
   SET_TRIS_C (0X00) ;
   SETUP_CCP1 (CCP_PWM); // RC2
   setup_timer_2 (T2_DIV_BY_16, 249, 1) ; //16-249-1 <-> TPWM=0.8ms
   TOC_DO=500;// 50% -> 500; // 100% -> 1000
   set_pwm1_duty(TOC_DO); // 50% -> 500
   WHILE(TRUE){}//DUNG LAI
}//end main
```

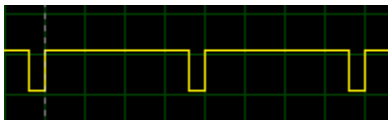
10% -> 100



50% -> 500



90% -> 900



```c
#include <16F877A.h>
#FUSES NOWDT, HS, NOPUT, PROTECT, NODEBUG, NOBROWNOUT, NOLVP
#use delay(clock=4000000)
#include <lcd.c>

//=====================================================

UNSIGNED INT16 TOC_DO;
void main()
{
   SET_TRIS_C (0X00) ;
   SETUP_CCP1 (CCP_PWM); // RC2
   SETUP_CCP2 (CCP_PWM); // RC1
   setup_timer_2 (T2_DIV_BY_4, 99, 1) ;
   TOC_DO=350;// 100% -> 400
   set_pwm1_duty(TOC_DO);
   set_pwm2_duty(50);
   WHILE(TRUE){}//DUNG LAI
}//end main
```