Taller virtual de reforzamiento de Microcontroladores

Profesor: Kalun José Lau Gan

2023-2

Agenda

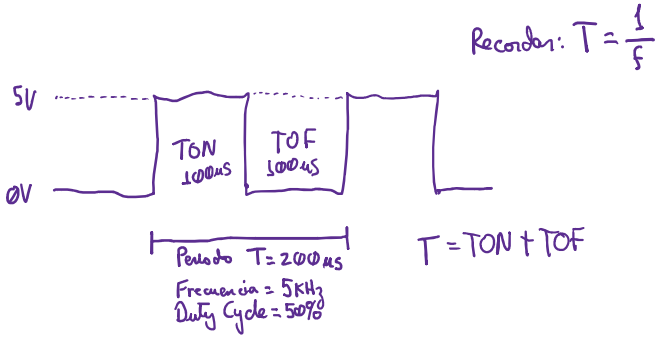
- Reforzar en el tema de interrupciones del PIC18F57Q43
- PWM
- Uso del Timer0
- UART
- Modo SLEEP

Las interrupciones vectorizadas

- Cuando el bit de configuración MVECEN = ON (nos basamos en la tabla de interrupciones vectorizadas – IVT). Ref cap. 11 datasheet.
- La prioridad la define el orden natural según IVT.
- Los habilitadores de cada interrupción se encuentran en los registros PIEx
- Las banderas de cada interrupción se encuentran en los registros PIRx
- No olvidar de habilitar el habilitador global de las interrupciones GIE (INTCON0 bit 7)

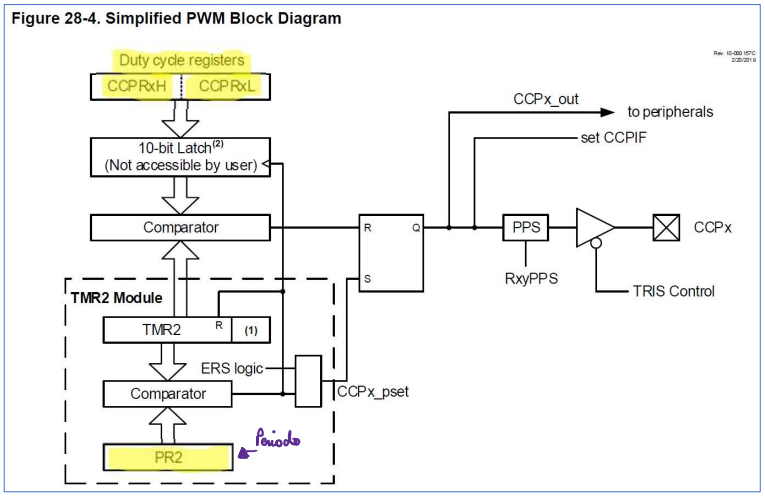
¿Señal PWM?

- PWM = Pulse Width Modulation
- Es una señal cuadrada con frecuencia fija y ancho de pulso variable.



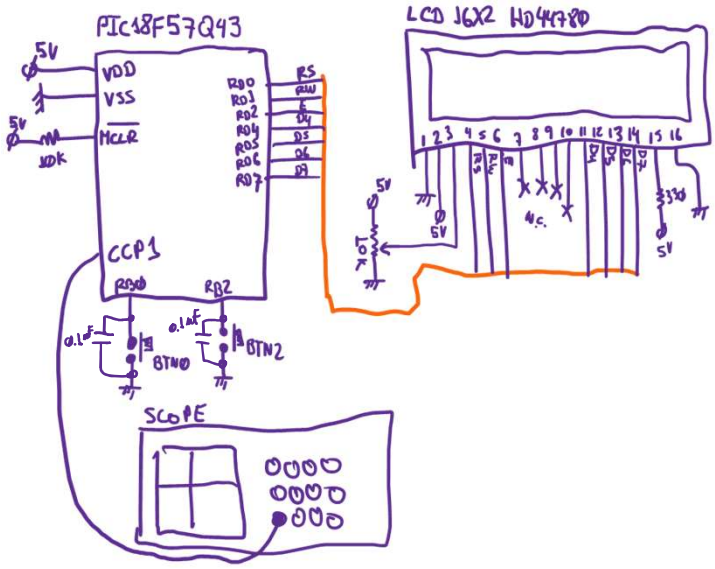
El periférico CCP1 en modo PWM

- Referencia capítulo 28 del datasheet
- Emplea como base de tiempo el Timer2
- En el gráfico de la derecha, PR2 (T2PR en el PIC18F57Q43) define el periodo.
- En el gráfico de la derecha, CCPRxH:CCPRxL define el duty cycle.



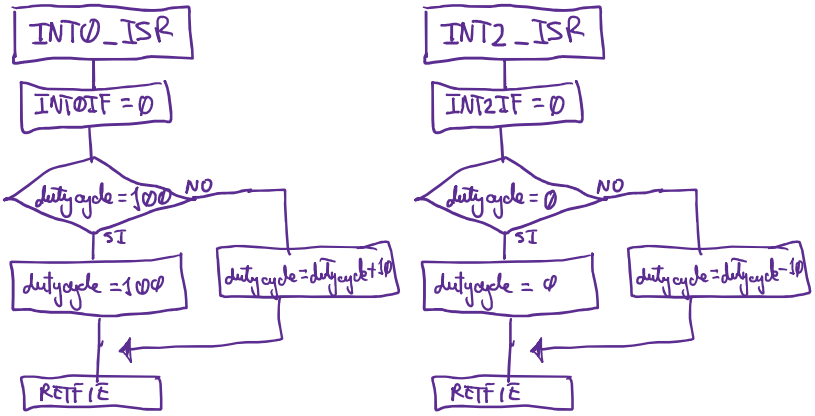
Circuito de prueba

- LCD – Emplear librería LCD proporcionado en el curso
- Botones activos en bajo – activar las weak pullup
- Salida PWM a través del CCP1
- La frecuencia de PWM será 5KHz.
- Los botones servirán para modificar el duty cycle del PWM. RB0 incrementará el duty cycle y el RB2 decrementar el duty cycle.
- En el display se visualizará el valor de duty cycle actual



Función de los pulsadores

- BTN0 (INT0) servirá para incrementar el Duty Cycle de 10 en 10
- BTN2 (INT2) servirá para decrementar el duty cycle de 10 en 10



Cálculo del periodo del PWM

- Según datasheet:

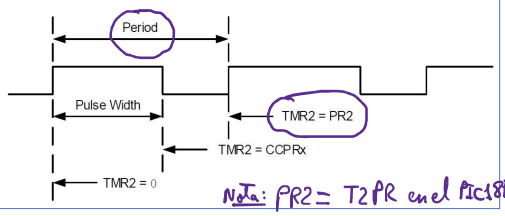
28.4.1 Standard PWM Operation

The standard PWM function described in this section is available and identical for all CCP modules. It generates a Pulse-Width Modulation (PWM) signal on the CCPx pin with up to ten bits of resolution. The period, duty cycle, and resolution are controlled by the following registers:

- Even numbered TxPR registers (T2PR, T4PR, etc.)
- Even numbered TxCON registers (T2CON, T4CON, etc.)
- 16-bit CCPRx registers
- CCPxCON registers

It is required to have $F_{OSC}/4$ as the clock input to TxTMR for correct PWM operation. The following figure shows a simplified block diagram of the PWM operation.

Figure 28-3. CCP PWM Output Signal



Nota: PR2 = T2PR en el PIC18F57Q43

Equation 28-1. PWM Period

$$PWM\ Period = [(T2PR + 1)] \bullet 4 \bullet T_{OSC} \bullet (TMR2\ Prescale\ Value)$$

where $T_{OSC} = 1/F_{OSC}$

Despejando T2PR:

$$T2PR = \left(\frac{\text{Periodo PWM}}{4 \cdot Tosc \cdot (\text{Prescale value})}\right) - 1$$

Nota: Tener en cuenta que TMR2 debe de tener como fuente de reloj: Fosc/4

- Frecuencia solicitada: 5 KHz ($T = 200\,\mu s$)
- Selección de Prescaler del TMR2: 1:16
- Fosc: 16MHz

$$\Rightarrow T2PR = \left(\frac{(\text{Periodo PWM})(Fosc)}{4 \cdot \text{Prescale value}}\right) - 1$$

$$T2PR = \left(\frac{(200 \times 10^{-6})(16 \times 10^{6})}{4 \cdot 16}\right) - 1$$

$$T2PR = 49$$

Nota: Tener en consideración que T2PR al ser un registro de 8 bits, solo puede contener valores de entre 0 y 255.

Ajustar el valor del divisor de frecuencia del Timer2 para lograr cumplir con lo anterior.

Cálculo del duty cycle del PWM

- Según datasheet:

Figure 28-3. CCP PWM Output Signal

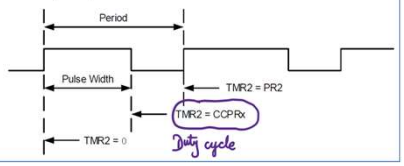
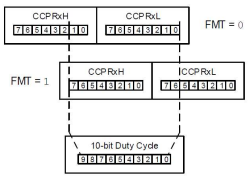


Figure 28-5. PWM 10-Bit Alignment



Equation 28-2. Pulse Width

$$Pulse\ Width = (CCPRxH{:}CCPRxL\ register\ value) \bullet T_{OSC} \bullet (TMR2\ Prescale\ Value)$$

Equation 28-3. Duty Cycle

$$DutyCycleRatio = \frac{(CCPRxH{:}CCPRxL\ register\ value)}{4(T2PR+1)}$$

- La resolución máxima del duty cycle es de 10 bits
- El duty cycle se carga en el par de registros CCPRxH:CCPRxL
- Podemos ajustar entre justificación a la izquierda y justificación a la derecha en el valor de duty cycle (10bits) en el par de registros CCPRxH:CCPRxL.
- Si es que se opta por solo usar 8bits de resolución en el duty cycle, justificar a la izquierda de tal modo que los 8 bits mas significativos se encuentren en el registro CCPRxH y así usar un solo registro para la manipulación del duty cycle.
- Tal como se muestra en el diagrama anterior, si CCPRxH es igual a PR2 (T2PR) el duty cycle será 100%, si CCPRxH es la mitad de PR2 (T2PR) el duty cycle será de 50% (onda cuadrada simétrica).

⇒ De la diapositiva anterior: T2PR=49

Aproximando:

Duty cycle	CCPR1H
0%	0
50%	25
100%	50

9

Código propuesto:

```c
#include <xc.h>
#include "cabecera.h"
#include "LCD.h"
#define _XTAL_FREQ 16000000UL

unsigned char dutycycle = 0;
unsigned char centena, decena, unidad;

void configuro(void){
    //configuracion del oscilador
    OSCCON1 = 0x60;      //HFINTOSC y PSC 1:1
    OSCFRQ = 0x05;       //HFINTOSC a 16MHz
    OSCEN = 0x40;        //HFINTOSC enabled
    //configuracion de las E/S
    TRISBbits.TRISB0 = 1;   //RB0 entrada
    ANSELBbits.ANSELB0 = 0; //RB0 digital
    WPUBbits.WPUB0 = 1;     //RB0 pullup enabled
    TRISBbits.TRISB2 = 1;   //RB0 entrada
    ANSELBbits.ANSELB2 = 0; //RB0 digital
    WPUBbits.WPUB2 = 1;     //RB0 pullup enabled
    //configuracion de las interrupciones
    PIE1bits.INT0IE = 1;        //INT0 enabled
    PIE10bits.INT2IE = 1;       //INT2 enabled
    INTCON0bits.INT0EDG = 0;    //INT0 falling edge detect
    INTCON0bits.INT2EDG = 0;    //INT2 falling edge detect
    PIR1bits.INT0IF = 0;        //INT0 flag en cero
    PIR10bits.INT2IF = 0;       //INT2 flag en cero
    INTCON0bits.GIE = 1;        //global interrupt enabled
    //configuracion de PWM 5KHz y 50%DC
    TRISCbits.TRISC2 = 0;       //RC2 como salida
    ANSELCbits.ANSELC2 = 0;     //RC2 como digital
    T2PR = 49;                  //freq del PWM a 5KHz
    RC2PPS = 0x15;              //asignacion PPS de CCP1 para que salga por RC2
    CCP1CON = 0x9C;              //CCP1 on, PWM mode, left justified
    CCPR1H = 25;
    CCPR1L = 0;                 //duty cycle a 50%
    T2CLKCON = 0x01;            //TMR2 con FOSC/4
    T2CON = 0xC0;               //TMR2 ON pres 1:16 posts 1:1
}
```

```c
void lcd_init(void){
    TRISD = 0x00;
    ANSELD = 0x00;
    LCD_CONFIG();
    __delay_ms(23);
    BORRAR_LCD();
    CURSOR_HOME();
    CURSOR_ONOFF(OFF);
}

void convierte(unsigned char numero){
    centena = numero / 100;
    decena = (numero % 100) / 10;
    unidad = numero % 10;
}

void main(void) {
    configuro();
    lcd_init();
    POS_CURSOR(1,1);
    ESCRIBE_MENSAJE("PWM Generator", 13);
    while(1){
        POS_CURSOR(2, 1);
        ESCRIBE_MENSAJE("Duty Cycle:", 11);
        convierte(dutycycle);
        ENVIA_CHAR(centena+0x30);
        ENVIA_CHAR(decena+0x30);
        ENVIA_CHAR(unidad+0x30);
        ENVIA_CHAR('%');
        CCPR1H = dutycycle / 2;
    }
}
```

```c
void __interrupt(irq(IRQ_INT0)) INT0_ISR(void){
    PIR1bits.INT0IF = 0;
    if(dutycycle == 100){
        dutycycle = 100;
    }
    else{
        dutycycle = dutycycle + 10;
    }
}

void __interrupt(irq(IRQ_INT2)) INT2_ISR(void){
    PIR10bits.INT2IF = 0;
    if(dutycycle == 0){
        dutycycle = 0;
    }
    else{
        dutycycle = dutycycle - 10;
    }
}
```

10