Universidad de Valladolid

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Electrónica Industrial y Automática

Generador de disparos configurable

Anejo I

Autor:

Nozal Fernández, Sara

Tutor:

Diez Muñoz, Pedro Luis
Tecnología Electrónica

Valladolid, Abril 2018

ÍNDICE

1. MAIN.C 4
2. BOTON.C 7
3. OUTPUT_COMPARE.C 8
4. TIMER.C 12
5. UART.C 14
6. BOTON.H 21
7. OUTPUT_COMPARE.H 23
8. TIMER.H 25
9. UART.H 27

1. MAIN.C

```
/*
 * File:   main.c
 * Author: Sara Nozal Fernández
 * Proyecto: Disparador de Pulsos
 *
 * Created on 19 de diciembre de 2017, 19:26
 */

// PIC24FJ128GA010 Configuration Bit Settings

// 'C' source line config statements

// CONFIG2
#pragma config POSCMOD = XT                 // Primary Oscillator Select (XT Oscillator mode selected)
#pragma config OSCIOFNC = OFF           // Primary Oscillator Output Function (OSC2/CLKO/RC15 functions as CLKO (FOSC/2))
#pragma config FCKSM = CSDCMD           // Clock Switching and Monitor (Clock switching and Fail-Safe Clock Monitor are disabled)
#pragma config FNOSC = PRI              // Oscillator Select (Primary Oscillator (XT, HS, EC))
#pragma config IESO = OFF               // Internal External Switch Over Mode (IESO mode (Two-Speed Start-up) disabled)

// CONFIG1
#pragma config WDTPS = PS32768          // Watchdog Timer Postscaler (1:32,768)
#pragma config FWPSA = PR128            // WDT Prescaler (Prescaler ratio of 1:128)
#pragma config WINDIS = ON              // Watchdog Timer Window (Standard Watchdog Timer enabled,(Windowed-mode is disabled))
```

```
#pragma config FWDTEN = OFF          // Watchdog Timer Enable (Watchdog Timer is disabled)
#pragma config ICS = PGx2            // Comm Channel Select (Emulator/debugger uses EMUC2/EMUD2)
#pragma config GWRP = OFF            // General Code Segment Write Protect (Writes to program memory are allowed)
#pragma config GCP = OFF             // General Code Segment Code Protect (Code protection is disabled)
#pragma config JTAGEN = OFF          // JTAG Port Enable (JTAG port is disabled)
// #pragma config statements should precede project file includes.



#include "xc.h"
#include<stdio.h>
#include <p24fj128ga010.h>
#include "uart.h"
#include "timer.h"
#include "output_compare.h"
#include "boton.h"
#include <libpic30.h>  // para usar __C30_UART=2;
#define UART_H
//int espera=0;



int main(void) {

   TRISA= 0;


   init_T2 ();
   init_uart ();
   init_Botones();
```

```
    LATAbits.LATA7=1; //Activamos el LED D10

    long i = 0;
    while (1) {

        printf("Si se desea crear un pulso mande una S");
        printf("\n");
        printf("Sino pulse E");
        printf("\n");
        U2TXREG = 0x0A; // Debug caracter de nueva linea (\n))
        for (i = 0; i < 20000000; i++);

    }
    return 0;
}
```

2. BOTON.C

```
/*
 * File:   output_compare.c
 * Author: Sara Nozal Fernández
 * Proyecto: Disparador de Pulsos
 *
 * Created on 19 de diciembre de 2017, 19:26
 */


#include "xc.h"
#include<stdio.h>
#include <p24fj128ga010.h>
#include <stdbool.h>

#define S3_PORT  PORTDbits.RD6

/*******Rutina del servicio de inicialización del Boton S3*********/
void init_Botones(void){
   TRISDbits.TRISD6 = 1;
}

/*************** Comprobacion de si se pulsa el boton S3*******************/
bool boton_presionado (void){
   return ( ( S3_PORT == 0 ) ? true : false ) ;
}
```

3. OUTPUT_COMPARE.C

```
/*
 * File:   output_compare.c
 * Author: Sara Nozal Fernández
 * Proyecto: Disparador de Pulsos
 *
 * Created on 19 de diciembre de 2017, 19:26
 */



#include "xc.h"
#include<stdio.h>
#include "output_compare.h"
#include <p24fj128ga010.h>
#include <libpic30.h>  // para usar __C30_UART=2;
#include "uart.h"



/*******Rutina del servicio de inicialización del OutputCompare*********/
void init_OutputCompare(char tipo)
{
    printf("Inicialización del módulo Output Compare");


      switch (tipo){
       case '1':
            TRISDbits.TRISD0 = 0; //El pin 0 del puerto D como salida RD0 (OC1)
            OC1CON = 0x0000; // Apaga la salida del comparador 1
           // OC1R = 0x3000; // Iniciaiza el registro del comparador
           // OC1RS = 0x3003; //Inicializa el registro del segundo comparador
```

```
            OC1CONbits.OCM2 = 1; // Carga el modo dual de generación de un
pulso
            OC1CONbits.OCM1 = 0;
            OC1CONbits.OCM0 = 0;
            OC1CONbits.OCTSEL = 0; //Indica que se va a utilizar el Timer2

            IPC0bits.OC1IP0 = 1; //Configuración de salida del Compare 1
interrupt para
            IPC0bits.OC1IP1 = 1; // la prioridad deseada
            IPC0bits.OC1IP2 = 1; // nivel 7 de prioridad
            IFS0bits.OC1IF = 0; // Limpia la flag de interrupción de la salida del
comparador 1
            IEC0bits.OC1IE = 1; // Activa la interrupcion del comprador 1

            T2CONbits.TON = 1; //Activamos el timer2 para que empiece a
contar
            break;

        case '2':
            TRISDbits.TRISD1= 0; //El pin 1 del puerto D como salida RD1
(OC2)
            OC2CON = 0x0000; // Apaga la salida del comparador 2
            OC2R = 0x3000; // Iniciaiza el registro del comparador
            OC2RS = 0x3003; //Inicializa el registro del segundo comparador
            OC2CONbits.OCM2 = 1; // Carga el modo dual de generación de un
pulso
            OC2CONbits.OCM1 = 0;
            OC2CONbits.OCM0 = 0;
            OC2CONbits.OCTSEL = 0; //Indica que se va a utilizar el Timer2

            IPC1bits.OC2IP0 = 1; //Configuración de salida del Compare 1
interrupt para
            IPC1bits.OC2IP1 = 1; // la prioridad deseada
```

```
        IPC1bits.OC2IP2 = 1; // nivel 7 de prioridad
        IFS0bits.OC2IF = 0; // Limpia la flag de interrupción de la salida del comparador 1
        IEC0bits.OC2IE = 1; // Activa la interrupcion del comprador 1

        T2CONbits.TON = 1; //Activamos el timer2 para que empiece a contar
        break;

    }

}

/*****************************************************/
/*                INTERRUPCIÓN DE LA FUNCIÓN         */
/*****************************************************/

/*************** Rutina de interrupcion del OC1*******************/
void __attribute__ ((__interrupt__, auto_psv)) _OC1Interrupt(void)
{
  LATAbits.LATA1^=1 ;//Debug (el led D4 cambia de estado con cada pulso generado)
  U2TXREG = 0x0A; // Debug caracter de nueva linea (\n))
  printf("Se crea el pulso deseado en OC1");
  printf("\n");

IFS0bits.OC1IF = 0;
}
```

```
/*************** Rutina de interrupcion del OC2*******************/
void __attribute__ ((__interrupt__, auto_psv)) _OC2Interrupt(void)
{
    LATAbits.LATA0^=1 ;//Debug (el led D3 cambia de estado con cada pulso generado)
    U2TXREG = 0x0A; // Debug caracter de nueva linea (\n))
    printf("Se crea el pulso deseado en OC2");
    printf("\n");

IFS0bits.OC2IF = 0;
}
```

4. TIMER.C

```
*
 * File:   timer.c
 * Author: Sara Nozal Fernández
 * Proyecto: Disparador de Pulsos
 *
 * Created on 19 de diciembre de 2017, 19:26
 */


#include "xc.h"
#include "timer.h"
#include <p24fj128ga010.h>
extern unsigned long espera;

/******Rutina del servicio de inicialización del Timer2******/
void init_T2 (void)
{
   T2CONbits.TCS = 0; // Reloj interno Fosc/2
   T2CONbits.TCKPS1 = 1; // Preescala 1:64
   T2CONbits.TCKPS0 = 0;
   T2CONbits.TGATE = 0;

   TMR2 = 0X0000; //Limpiamos el registro de T2
   PR2 = 0x3006; // Periodo de T2

   IFS0bits.T2IF = 0; //Baja la bandera de interrupción
   IEC0bits.T2IE = 1; // Habilita la interrupción del Timer 2
   IPC1bits.T2IP = 5; //Nivel de interrupción del timer T2 =5

 }
```

```
/*****************************************************/
/*                INTERRUPCIÓN DE LA FUNCIÓN         */
/*****************************************************/

/********** Rutina del servicio de interrupción del T2 timer mode**********/
void __attribute__ ( (interrupt, shadow, no_auto_psv) ) _T2Interrupt( void )
{
    LATAbits.LATA2^=1 ;//Debug (el led D5 cambia de estado con cada vez que se resetea el contador)
    IFS0bits.T2IF = 0; //Baja la bandera de interrupción
}
```

5. UART.C

```
/*
 * File:   uart.c
 * Author: Sara Nozal Fernández
 * Proyecto: Disparador de Pulsos
 *
 * Created on 19 de diciembre de 2017, 19:26
 */


#include "xc.h"
#include <p24fj128ga010.h>
#include <libpic30.h>  // para usar __C30_UART=2;
#include "uart.h"
#include "output_compare.h"
#include "boton.h"
#include<stdio.h>



// Rutina del servicio de inicialización del UART2
// y tanto para recepción como transmisión

void init_uart (void)
{
    printf("\n");
    printf("Inicialización del módulo UART");

     __C30_UART=2; //seleciona la UART2 para la función printf
    _TRISF13= 0; //El pin 13 del puerto F como salida RF13 (U2RTS)
```

```
    _LATF13=0; //Este pin esta conectado al RS232 señal RTS (Ready To
Send)
    _TRISF12=1; //El pin 12 del puerto F como entrada RF12 (U2CTS))


  U2MODEbits.UARTEN = 0; //limpia los registros
  U2MODEbits.RTSMD = 1; //modo simplex
  U2MODEbits.PDSEL0 = 0; //8 bits y no paridad
  U2MODEbits.PDSEL1 = 0;
  U2MODEbits.STSEL = 0; //1 bit parada


  U2STAbits.UTXISEL0 = 0;
  U2STAbits.UTXISEL1 = 0;

  IFS1bits.U2RXIF = 0; //Baja la bandera de interrupción
  IFS1bits.U2TXIF = 0; //Baja la bandera de interrupción
  IEC1bits.U2RXIE = 1; //Habilita la interrupción de Recepción
  IEC1bits.U2TXIE = 1; //Habilita la interrupción de Transmisión
  IPC7bits.U2RXIP = 6;  //Nivel de interrupción del receptor del UART1 =6
  IPC7bits.U2TXIP = 6; //Nivel de interrupción del transmisor del UART1 =6

   U2BRG = 25;


  U2MODEbits.UARTEN = 1; //Activamos la UART1
  U2STAbits.UTXEN = 1; // Activamos la transmision
}
```

```
/*****************************************************/
/*                INTERRUPCIÓN DE LA FUNCIÓN         */
/*****************************************************/


// Rutina del servicio de interrupción del UART2 Recepción
void __attribute__ ( (interrupt, no_auto_psv) ) _U2RXInterrupt( void )
{
   _LATA3 ^=1;  //Debug (el led D6 cambia de estado con cada byte recibido)
   int aux;

   switch (aux){
      case 1:
            printf("Valor del registro OC1: ");
            OC1R = U2RXREG;
            U2TXREG = OC1R;
            printf("\n");
            aux = 3;
            printf("Para indicar el valor de OC1RS pulse T");
            printf("\n");
            break;
      case 2:
            printf("\n");
            printf("Valor para registro OC1RS: ");
            OC1RS = U2RXREG;
            U2TXREG = OC1RS;
            printf("\n");
            aux = 3;
            printf("Indique que tipo de pulso desea crear");
            printf("\n");
            printf("Pulso por una patilla de ancho variable: Marca A");
```

```
            printf("\n");
            printf("Dos pulsos por una misma patilla y diferencia entre ellos variable: Marca B");
            printf("\n");
            printf("Dos pulsos, cada uno por una patilla y diferencia entre ellos variable: Marca C");
            printf("\n");

            break;

        case 0:

            OC1R = 0x3000; // Iniciaiza el registro del comparador
            OC1RS = 0x3003; //Inicializa el registro del segundo comparador
            aux = 3;

            break;
        case 3:
           break;
    }


    switch (U2RXREG){

        case 'S':

            printf("\n");
            _LATA0 = 1;  // Enciende el LED D3
            printf("Indique valores para los tiempos, pulse 'R' para establecer OC1");
            printf("\n");
            printf("Sino se establecera el valor por defecto pulsando 'D'");
```

```
        printf("\n");
        break;

    case 'A':
        printf("\n");
        init_OutputCompare ('1');
        break;

    case 'B':
        printf("\n");
        init_OutputCompare ('1');
        printf("\n");
        printf("Pulsa el boton S3 cuando se desee crear el segundo pulso");
        printf("\n");
        while (!boton_presionado( )) ;
        init_OutputCompare ('1');
        break;

    case 'C':
        printf("\n");
        init_OutputCompare ('1');
        printf("\n");
        printf("Pulsa el boton S3 cuando se desee crear el segundo pulso");
        printf("\n");
        while (!boton_presionado( )) ;
        init_OutputCompare ('2');
        break;

    case 'R':
        aux = 1;
        printf("\n");
```

```
            printf("Valor para registro OC1");
            printf("\n");
            break;

        case 'T':
            aux = 2;
            printf("\n");
            printf("Valor para registro OC1RS");
            printf("\n");
            break;

        case 'D':
            aux = 0;
            printf("Indique que tipo de pulso desea crear");
            printf("\n");
            printf("Pulso por una patilla de ancho variable: Marca A");
            printf("\n");
            printf("Dos pulsos por una misma patilla y diferencia entre ellos variable: Marca B");
            printf("\n");
            printf("Dos pulsos, cada uno por una patilla y diferencia entre ellos variable: Marca C");
            printf("\n");
           break;

       }

  // espera = 1;
   U2TXREG = 0x0A;
   IFS1bits.U2RXIF = 0;   //Baja la bandera de interrupción
```

```
}


// Rutina del servicio de interrupción del UART1 Transmisión
// Limpia la bandera de interrupción
void __attribute__ ( (interrupt, no_auto_psv) ) _U2TXInterrupt( void )
{
    _LATA4 ^=1;   //Debug (el led D7 cambia de estado con cada byte transmitido)
   IFS1bits.U2TXIF = 0;  //Baja la bandera de interrupción
}
```

6. BOTON.H

```
/*
 * File:  output_compare
 * Author: Sara Nozal Fernández
 * Proyecto: Disparador de Pulsos
 *
 * Definición: Se definen las funciones relacionadas
 * con el Output Compare.
 * Created on 19 de diciembre de 2017, 19:26
 */

#ifndef BOTON_H
#define         BOTON_H

#include <xc.h> // include processor files - each processor file is guarded.
#include <p24fj128ga010.h>
#include <stdbool.h>

/**
 * @Nombre de la funcion
 * init_botones
 *
 * @Parametros
   ninguno
 * @Retuns
   ninguno
 * @Description
   Rutina del servicio de inicialización del boton1 S3

 */
void init_Botones(void);
```

```
/**
 * @Nombre de la funcion
 * init_botones
 *
 * @Parametros
   ninguno
 * @Retuns
   Verdadero o falso
 * @Description
   Rutina para identificar si el botón S3 ha sido presionado

 */
bool boton_presionado (void);

#endif /*BOTON_H */
```

7. OUTPUT_COMPARE.H

```
/*
 * File:  output_compare
 * Author: Sara Nozal Fernández
 * Proyecto: Disparador de Pulsos
 *
 * Definición: Se definen las funciones relacionadas
 * con el Output Compare.
 * Created on 19 de diciembre de 2017, 19:26
 */

#ifndef OUTPUT_COMPARE_H
#define         OUTPUT_COMPARE_H

#include <xc.h>
#include <p24fj128ga010.h>

/**
 * @Nombre de la funcion
 * init_OutputCompare
 *
 * @Parametros
   ninguno
 * @Retuns
   ninguno
 * @Description
   Rutina del servicio de inicialización del OC1

 */
void init_OutputCompare(char tipo);
```

```
/**
 * @Nombre de la funcion
 *  _OC1Interrupt
 *
 * @Parametros
   interrupt,auto_psv
 * @Retuns
   ninguno
 * @Description
   Rutina de interrupcion del OC1

 */
void __attribute__ ((__interrupt__, auto_psv)) _OC1Interrupt(void);



/**
 * @Nombre de la funcion
 *  _OC2Interrupt
 *
 * @Parametros
   interrupt,auto_psv
 * @Retuns
   ninguno
 * @Description
   Rutina de interrupcion del OC2

 */
void __attribute__ ((__interrupt__, auto_psv)) _OC2Interrupt(void);

#endif          /* OUTPUT_COMPARE_H */
```

8. TIMER.H

```
/*
 * File:  timer
 * Author: Sara Nozal Fernández
 * Proyecto: Disparador de Pulsos
 *
 * Created on 19 de diciembre de 2017, 19:26
 */

#ifndef TIMER_H
#define        TIMER_H

#include <xc.h>
#include <p24fj128ga010.h>

/**
 * @Nombre de la funcion
 * init_T2
 *
 * @Parametros
   ninguno
 * @Retuns
   ninguno
 * @Description
   Rutina del servicio de inicialización del Timer2

 */
void init_T2 (void);

/**
 * @Nombre de la funcion
```

```
 *  _T2Interrupt
 *
 * @Parametros
   interrupt,shadow
 * @Retuns
   ninguno
 * @Description
   Rutina de interrupcion del Timer2

 */
void __attribute__ ( (interrupt,  shadow) ) _T2Interrupt( void );

#endif /* TIMER_H */
```

9. UART.H

```
/*
 * File:  uart
 * Author: Sara Nozal
 * Proyecto: Disparador de Pulsos
 *
 * Created on 19 de diciembre de 2017, 19:26
 */


#ifndef UART_H
#define         UART_H
extern int espera;
#include <xc.h>


/**
 * @Nombre de la funcion
 * init_uart
 *
 * @Parametros
   ninguno
 * @Retuns
   ninguno
 * @Description
   Rutina del servicio de inicialización del UART1
   y tanto para recepción como transmisión

 */
void init_uart (void);
```

```
/**
 * @Nombre de la funcion
 *  _U1RXInterrupt
 *
 * @Parametros
   interrupt, no_auto_psv
 * @Retuns
   ninguno
 * @Description
   Rutina de interrupcion del UART1 cuando recibe datos

 */
void __attribute__ ( (interrupt, no_auto_psv) ) _U1RXInterrupt( void );

/**
 * @Nombre de la funcion
 *  _U1TXInterrupt
 *
 * @Parametros
   interrupt, no_auto_psv
 * @Retuns
   ninguno
 * @Description
   Rutina de interrupcion del UART1 cuando transmite datos

 */
void __attribute__ ( (interrupt, no_auto_psv) ) _U1TXInterrupt( void );

#endif /* UART_H */
```