SISTEMAS EMPOTRADOS
3º Grado en Ingeniería Informática

PRÁCTICA 5

TIMERS *SOFTWARE* CON INTERRUPCIONES (VIC)

5.1. Introducción

Los recursos de los microcontroladores son limitados. En el caso del LPC2378 disponemos solamente de 4 temporizadores (*timers*) que, para el desarrollo de las prácticas que hemos realizado hasta ahora han sido suficientes ya que sólo hemos utilizado dos. Pero estos ejercicios pueden ser parte de algún proyecto mayor con lo que puede que tengamos que utilizar todos los temporizadores.

En esta práctica vamos a realizar lo mismo que en la anterior, es decir, que vamos a obtener dos señales por los puertos P4.24 y P4.25, pero utilizando sólo un *timer hardware*, el *timer* 0. Por lo que para obtener dos temporizaciones para cada señal tendremos que definir por *software* dos *timers* que acceden físicamente a un único temporizador *hardware* (T0).

Por otro lado, vamos a gestionar la temporización de la señal por interrupciones, con el objetivo de liberar al microcontrolador para que pueda dedicarse a otras gestiones, al igual que en la práctica anterior. Para ello tendremos que programar convenientemente los registros del VIC y definir las rutinas de servicio a la interrupción (ISR). En esta práctica vamos a separar esta rutina de interrupción del *timer* 0 en un fichero separado que llamaremos IRQ.c.

5.2. Objetivos

Los objetivos marcados en esta práctica son los siguientes:

- Que el alumnado estudie y aprenda a configurar el controlador de interrupciones vectorizadas VIC del LPC2378.

- Diseño de un programa de aplicación utilizando un solo timer *hardware* del microcontrolador y su comprobación real con el osciloscopio.

5.3. Material utilizado

El material necesario para la realización de esta práctica es el siguiente:

- Ordenador personal con *Windows* con el *software Keil µVision 5* instalado y el *pack* correspondiente a nuestra placa.
- Placa de desarrollo *Keil* MCB2300.
- Adaptador USB–JTAG de la familia ULINK para depurar programas.
- Dos cables USB A–B conectados a dos puertos USB disponibles del ordenador.
- Osciloscopio.

5.4. Desarrollo de la práctica

√ Esta práctica se realizará a partir de la anterior, pero utilizando un sólo *timer* (T0) del microcontrolador LPC2378.

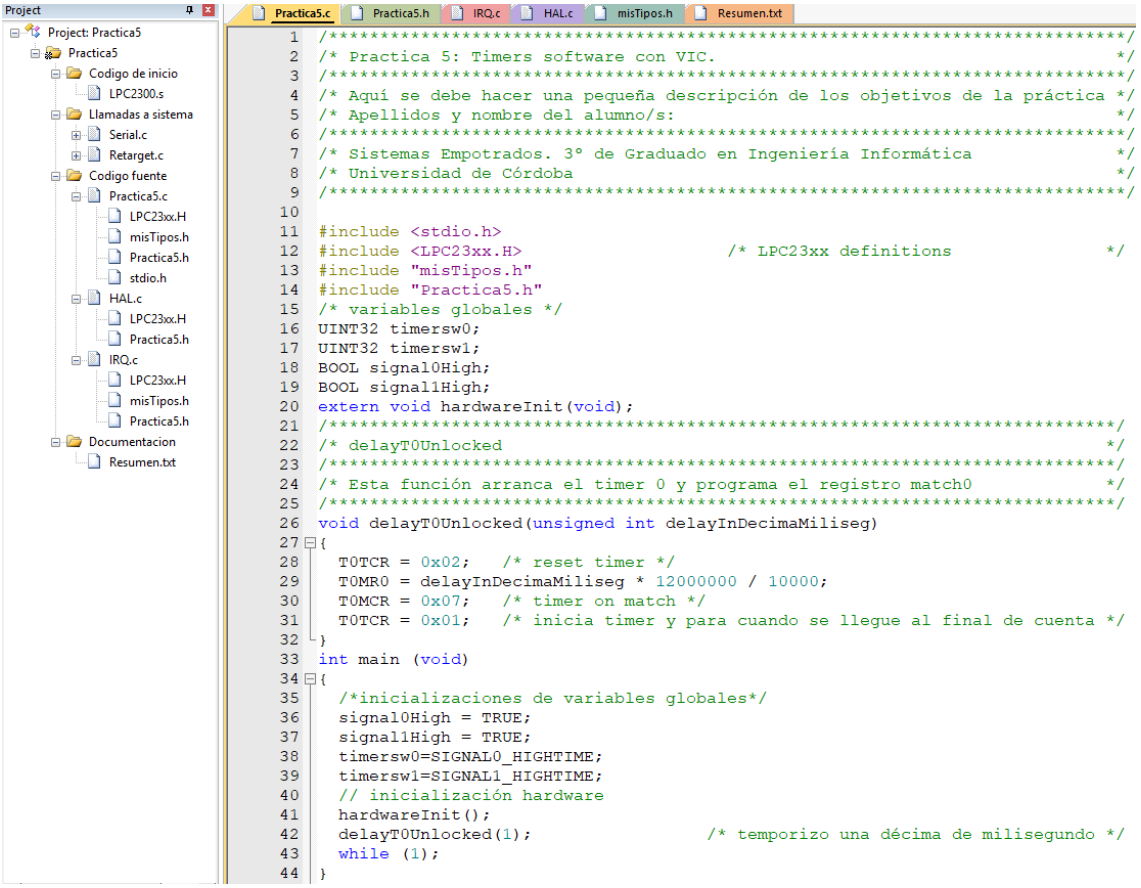
√ Trataremos de obtener dos señales periódicas de distinta anchura, igual que en las prácticas anteriores, pero programando convenientemente el *timer* T0 del micro.

√ Generar dos señales digitales por un puerto GPIO con distinta anchura a baja que a alta. La primera tendrá una duración a baja de 500 µs y a alta 700 µs y la segunda tendrá una duración a baja de 200 µs y a alta de 300 µs. Se comprobará con el osciloscopio que estas señales se generan correctamente.

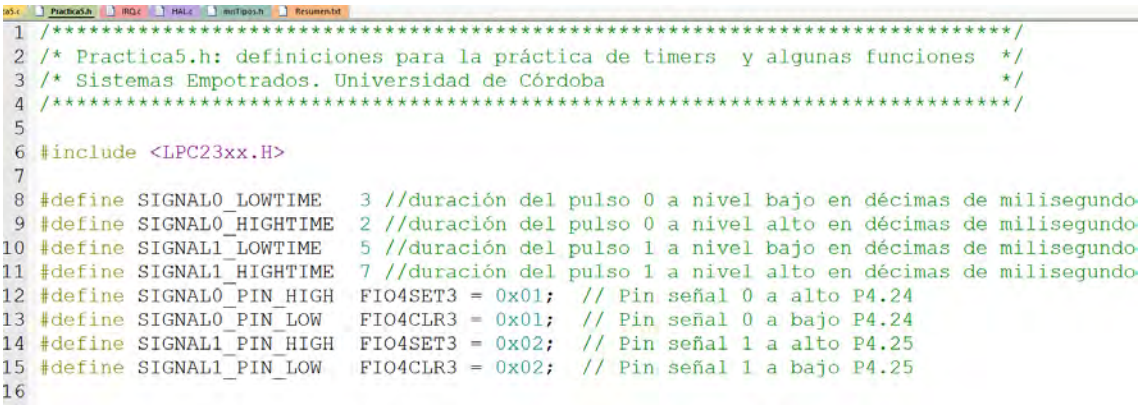
√ Crear un nuevo proyecto (practica5) en una carpeta personalizada para cada práctica. Copiar en esa carpeta los ficheros:

- LPC2300.s
- retarget.c (para configurar el microcontrolador, entradas/salidas, estándar de C, stdio.h, etc.)
- serial.c
- Practica5.h
- HAL.c
- IRQ.c

√ Crear un nuevo fichero fuente practica5.c



√ El fichero cabecera para las definiciones practica5.h queda de la forma siguiente:



√ Crear un fichero IRQ.c donde estarán las rutinas de servicio a la interrupción (ISR) con el que se controlan las interrupciones que quedaría así:

Practica5.c | Practica5.h | IRQ.c | HAL.c | misTipos.h | Resumen.txt

```
/*****************************************************************************/
/* IRQ.C: Rutinas de servicio a la interrupción (ISR)                        */
/* Sistemas Empotrados. Universidad de Córdoba                               */
/*****************************************************************************/
#include <LPC23xx.H>                        /* LPC23xx definitions           */
#include "Practica5.h"
#include "misTipos.h"
extern UINT32 timersw0;
extern UINT32 timersw1;
extern BOOL signal0High;
extern BOOL signal1High;
extern void delayT0Unlocked(unsigned int delayInDecimaMiliseg);
/*****************************************************************************/
/* Timer0 IRQ: Rutina de servicio a la interrupción del timer 0              */
/*****************************************************************************/
__irq void T0_IRQHandler (void) {
  timersw0--;
  timersw1--;
  if (timersw0==0)   /* comprueba si el timer software 0 ha finalizado */
    {
      if (signal0High==TRUE)
      {
        SIGNAL0_PIN_LOW;
        signal0High=FALSE;
        timersw0=SIGNAL0_LOWTIME;
      }
      else
      {
        SIGNAL0_PIN_HIGH;
        signal0High=TRUE;
        timersw0=SIGNAL0_HIGHTIME;
      }
    }
  if (timersw1==0)   /* comprueba si el timer software 1 ha finalizado */
  {
    if (signal1High==TRUE)
    {
      SIGNAL1_PIN_LOW;
      signal1High=FALSE;
      timersw1=SIGNAL1_LOWTIME;
    }
    else
    {
      SIGNAL1_PIN_HIGH;
      signal1High=TRUE;
      timersw1=SIGNAL1_HIGHTIME;
    }
  }
  T0IR        = 1;    /* Clear interrupt flag                              */
  VICVectAddr = 0;    /* Acknowledge Interrupt                             */
  delayT0Unlocked(1); /* Vuelvo a arrancar el timer una vez programado */
}
```

√ En este caso, el fichero HAL.c para inicializar el *hardware* utiliza solamente un *timer hardware*, T0:

Practica5.c | Practica5.h | IRQ.c | HAL.c | misTipos.h | Resumen.txt

```
/*****************************************************************************/
/* HAL.C: funciones generales que acceden al hardware                        */
/* Sistemas Empotrados. Universidad de Córdoba                               */
/*****************************************************************************/

#include <LPC23xx.H>                                  /* LPC23xx definitions */
#include "Practica5.h"
/* Import external IRQ handlers from IRQ.c file                              */
extern __irq void T0_IRQHandler  (void);
/*****************************************************************************/
/* pinesSignalInit                                                           */
/*****************************************************************************/
/* Esta función configura los pines   P4.24 y P4.25 como salida              */
/*****************************************************************************/
void pinesSignalInit(void)
{
  PINSEL9 = 0x00000000;
  PINMODE9 = 0x000000000;
  FIO4DIR3  = 0x03;
}
/*****************************************************************************/
/* timer0Init                                                                */
/*****************************************************************************/
/* Esta función configura el timer 0 con los parámetros que no cambian       */
/* durante la aplicación                                                     */
/*****************************************************************************/
void timer0Init(void)
{
  T0PR  = 0x00;   /* activa el preescalador a cero */
  //controlador de interrupciones vectorizadas
  VICVectAddr4  = (unsigned long)T0_IRQHandler;       /* Set Interrupt Vector */
  VICVectPriority4  = 15;             /* Priority use it for Timer0 Interrupt */
  VICIntEnable  = (1  << 4);                        /* Enable Timer0 Interrupt */
}
/*****************************************************************************/
/* hardwareInit                                                              */
/*****************************************************************************/
/* Esta función se llama al comienzo del programa para inicializar el Hardware*/
/*****************************************************************************/
void hardwareInit(void)
{
  pinesSignalInit();  // Configura los pines del circuito
  timer0Init();       // Inicializa el timer 0
 }

```

- √ Las dos señales deberán observarse en el analizador lógico de *Keil µVision* y en los dos canales del osciloscopio.
- √ Hay que observar que tanto en el analizador lógico del *Keil µVision* como en el osciloscopio va a ser más inexacto que en la práctica 4.