SISTEMAS EMPOTRADOS
3º Grado en Ingeniería Informática

PRÁCTICA 2

Manejo de los puertos básicos de Entrada/Salida y Timers (I)

2.1. Introducción

El CI LPC2378 es un microcontrolador y por tanto multiplexa los pines como entradas y salidas. Cada pin maneja de una a cuatro funciones. Los pines de A0 a A15 y D0 a D8 son exclusivos para uso del controlador externo de memoria (EMC).

A continuación se muestran los registros de configuración de los pines de I/O:

- (F)IOxDIR (x = [0,4]), selecciona en cada *bit* si un pin es entrada o salida, con 0 es una entrada y con 1 es una salida.
- (F)IOxSET (x = [0,4]): escribe uno a los *bits* seleccionados de un registro IO.
- (F)IOxCLR (x = [0,4]): escribe cero a los *bits* seleccionados de un registro IO
- (F)IOxPIN (x = [0,4]): permite escribir o leer a un registro IO.
- IOxMASK (x = [0,4]): cuando se escriben unos a los *bits* se inhibe cualquier acción de escritura en los registros de entrada salida.
- PINSELx (x = [0,10]): estos registros escogen una de cuatro funciones que tiene cada pin del microcontrolador.

Para completar el estudio de los puertos GPIO debemos mirar como referencia los capítulos 8, 9 y 10 del manual de usuario LPC23xx de NXP *Semiconductors*. Para estudiar la configuración de los temporizadores el capítulo 23 de dicho manual.

Para el estudio teórico utilizamos el apartado 3.8.1 del libro de referencia de la asignatura "*The Insider's guide to the NXP LPC2300/2400 Based Microcontrollers. An engineer's introduction to the LPC2300 & LPC2400 series*". **http://www.hitex.co.uk.** Además de los apartados 4.2 y 4.3 para el estudio de los GPIO y timers.

2.2. Objetivos

Los objetivos marcados en esta práctica son los siguientes:

- Que el alumnado estudie y aprenda a configurar los puertos básicos de entrada/salida (GPIO) del LPC2378.
- Estudio y programación de los temporizadores (*timers*) de que dispone dicho microcontrolador.
- Diseño de un programa de aplicación y su simulación con el depurador del *Keil µVision* 5 y comprobación real con el osciloscopio.

2.3. 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.

2.4. Desarrollo de la práctica

Queremos generar una señal digital por un puerto GPIO con distinta anchura a nivel bajo que a nivel alto. Tendrá una duración a baja de 500 µs y a alta de 700 µs. Se comprobará con el osciloscopio que esta señal se genera correctamente.

Se programarán los timers por *polling* (sondeo). El final de la cuenta del *timer* se detectará muestreando los *bits* de control.

Crear un nuevo proyecto (practica2) en una nueva carpeta. Al igual que en la práctica anterior, copiar en esta carpeta los siguientes ficheros:

- LPC2300.s
- retarget.c
- serial.c
- HAL.c : para definir las funciones generales que acceden al *hardware*. En el caso de esta práctica al puerto de entrada/salida y al *timer* 0.)
- Practica2.h : definiciones necesarias para esta práctica y algunas funciones.

En el fichero HAL.c vamos a definir las funciones generales que acceden al *hardware*. En esta práctica solamente vamos a necesitar configurar un pin del puerto 4 (el pin P4.24) como salida, y el *timer* 0.

Practica2.c | Practica2.h | HAL.c | misTipos.h | Resumen.txt

```
1  /*****************************************************************************/
2  /* HAL.C: funciones generales que acceden al hardware.                       */
3  /* Capa de abstraccción del hardware (Hardware Abstract Layer)               */
4  /* Sistemas Empotrados. Universidad de Córdoba                               */
5  /*****************************************************************************/
6   #include <LPC23xx.H>                    /* LPC23xx definitions             */
7  /*****************************************************************************/
8  /* pinesSignalInit                                                           */
9  /*****************************************************************************/
10 /* Esta función configura el pin P4.24 como salida                           */
11 /*****************************************************************************/
12 void pinesSignalInit(void)
13 {
14   PINSEL9 = 0x00000000;
15   PINMODE9 = 0x000000000;
16   FIO4DIR3  = 0x01; //00000001b
17 }
18 /*****************************************************************************/
19 /* timer0Init                                                                */
20 /*****************************************************************************/
21 /* Esta función configura el timer 0 con los parámetros que no cambian       */
22 /* durante la aplicación                                                     */
23 /*****************************************************************************/
24 void timer0Init(void)
25 {
26   T0PR  = 0x00;                          /* activa el preescalador a cero */
27 }
28 /*****************************************************************************/
29 /* hardwareInit                                                              */
30 /*****************************************************************************/
31 /* Esta función se llama al comienzo del programa para inicializar el Hardware*/
32 /*****************************************************************************/
33 void hardwareInit(void)
34 {
35   pinesSignalInit();  // Configura los pines del circuito
36   timer0Init();       // Inicializa el timer 0
37 }
38
```

Figura 2-1 Fichero HAL.c para acceder al *hardware*

PINSEL9 = 0X00000000 selecciona el pin P4.24 como GPIO (bits 17 y 16)

PINMODE9 = 0X00000000 lo habilita como salida resistencia *pull–up*

FIO4DIR3 = 0x03 selecciona los pines P4.24 y P4.25

T0PR = 0x00 activa el preescalador del *timer* 0 a un valor cero.

El fichero Practica2.h lo utilizaremos para incorporar las definiciones necesarias.

```
1  /*****************************************************************************/
2  /* Practica2.h: definiciones para la práctica de timers  y algunas funciones  */
3  /* Sistemas Empotrados. Universidad de Córdoba                                */
4  /*****************************************************************************/
5
6  #define PULSO0_LOW  5                       // duración del pulso a nivel bajo
7  #define PULSO0_HIGH 7                       // duración del pulso a nivel alto
8  #define SIGNAL0_PIN_HIGH  FIO4SET3 = 0x01;  // Pin señal 0 a alto   P4.24
9  #define SIGNAL0_PIN_LOW   FIO4CLR3 = 0x01;  // Pin señal 0 a bajo   P4.24
10 #include <LPC23xx.H>
11 /*****************************************************************************/
12 /* delayT0Unlocked                                                           */
13 /*****************************************************************************/
14 /* Esta función arranca el timer 0 y programa el registro match0             */
15 /*****************************************************************************/
16 void delayT0Unlocked(unsigned int delayInDecimaMiliseg)
17 {
18   T0TCR = 0x02;   /* reset timer */
19   T0MR0 = delayInDecimaMiliseg * 12000000 / 10000;
20   T0MCR = 0x07;   /* timer on match */
21   T0TCR = 0x01;   /* inicia timer y para cuando se llegue al final de cuenta*/
22 }
23
```

Figura 2-2 Fichero timers.h para las definiciones.

FIO4SET3 = 0x01 coloca el pin P4.24 a nivel alto

FIO4CLR3 = 0x01 coloca el pin P4.24 a nivel bajo

T0TCR = 0x02 Registro de control del timer 0. Resetea el contador

T0MR0 = delayInDecimalMiliseg * (12000000/10000–1) Coloca el registro *match* en el valor deseado. El reloj del LPC2378 es 12 MHz.

T0MCR = 0x07 Registro de control del *match*. Genera una interrupción cuando coincida la cuenta del *timer* con el valor del registro *match*.

T0TCR = 0x05 Habilita la cuenta

En la programación de sistemas empotrados, es muy útil disponer de un fichero donde vamos añadiendo los tipos de variables que vamos a utilizar durante todo el diseño. Este fichero sería el misTipos.h que podría ser algo así:

Practica2.c | Practica2.h | HAL.c | misTipos.h | Resumen.txt

```
1  /*****************************************************************************/
2  /* misTipos.h: definiciones de tipos de variables para prácticas con LPC2378  */
3  /* Sistemas Empotrados. Universidad de Córdoba                                */
4  /*****************************************************************************/
5
6  #ifndef __misTipos_H
7  #define __misTipos_H
8
9  /* Byte */
10 #define UINT8 unsigned char
11 #define INT8 char
12
13 /*16 bits */
14 #define UINT16 unsigned short int
15 #define INT16  short int
16
17 /*32 bits WORD para el LPC2378 */
18 #define UINT32 unsigned int
19 #define INT32  int
20
21 /* Tipos para control */
22 #define STATUS UINT32
23
24 /* Boolenas */
25 #define BOOL   UINT32
26 #define FALSE (unsigned int)0x000000000
27 #define TRUE  (unsigned int)0x000000001
28
29 /* flags */
30 #define FLAG  BOOL
31 #define ESC 0x1B
32 #define OK 0x01
33 #endif
34
```

Figura 2-3 Fichero para la definición de tipos de variables.

Finalmente el programa principal para obtener una señal digital no periódica quedaría, por ejemplo, de la manera siguiente:

T0IR: Registro de interrupción del *timer* 0. Se lee para ver cuál de las interrupciones posibles están pendientes y se escribe en él para borrar esta interrupción.

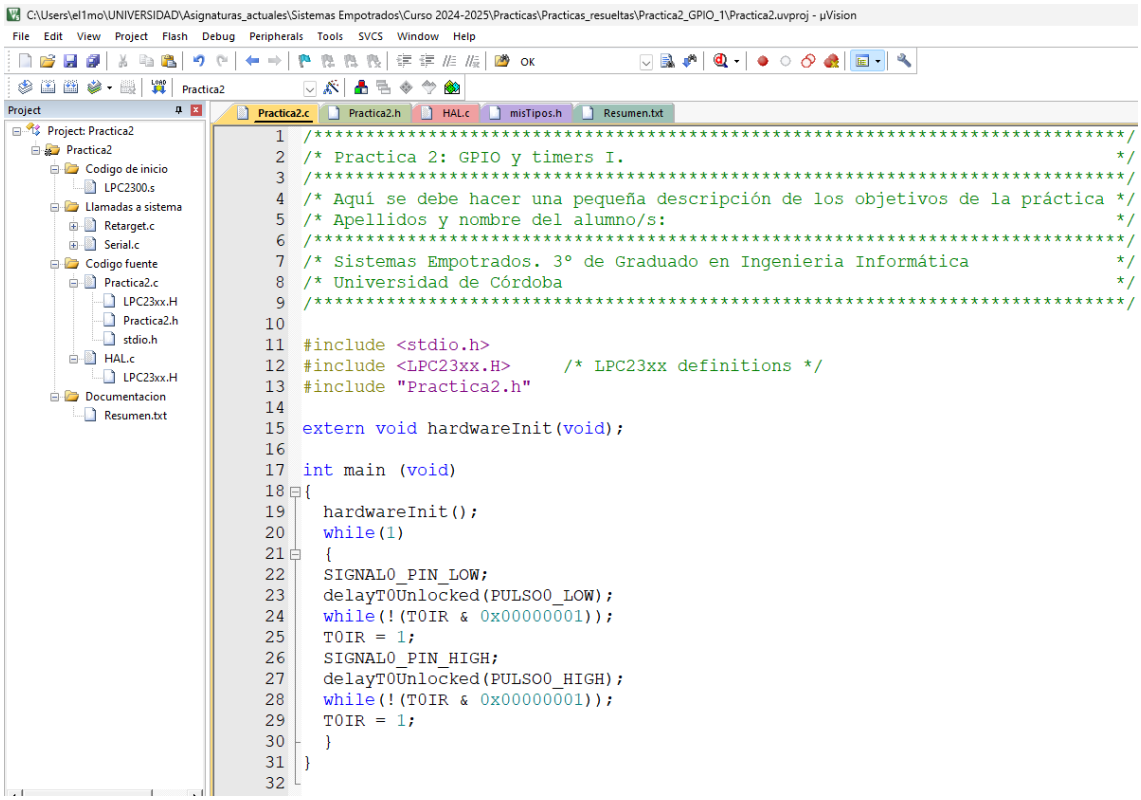

Figura 2-4 Programa principal de la aplicación.

2.5. Pruebas y resultados de la práctica

Una vez escrito el programa y añadidos los archivos convenientemente, habrá que guardar el programa  , compilarlo  y pasaremos a la depuración. Si han surgido errores o avisos (*warning*) habrá que corregirlos.

Para la simulación y depuración de la práctica utilizaremos la sesión *debugger* que nos proporciona el *Keil µVision*, pulsando Ctrl + F5, o en el menú *Debug* → *Start/Stop Debug Session* o, la forma más rápida con la herramienta rápida  .

- **Visualización de los registros de los periféricos:**

Una vez estemos en modo depuración, podremos observar los registros internos de nuestro microcontrolador. Podremos ejecutar el programa paso a paso o en ejecución rápida. Tendremos que observar los periféricos que hemos utilizado en esta práctica que son el *Timer* 0 y el GPIO P4.24.

Para ver los periféricos debemos hacerlo mediante el menú *Peripherals* → *Timer* → *Timer0*, y *Peripherals* → *GPIO* → *GPIO Fast Interface* → *Port4*. Podremos observar en la ejecución los registros internos del *timer* 0 y en el Port4 que cambia el pin p4.24.

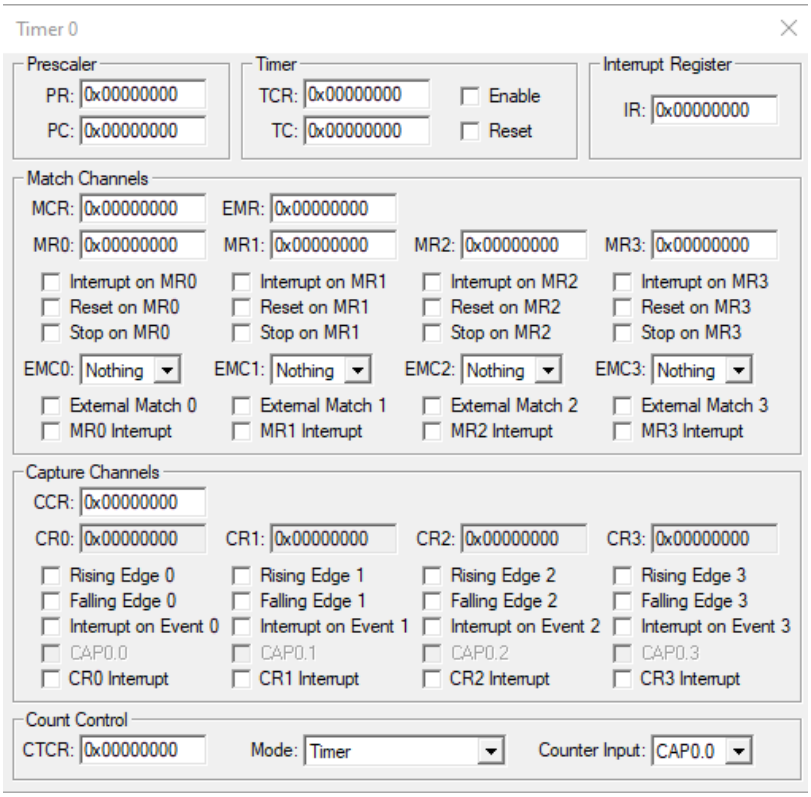

Figura 2-5 Visualización de los registros internos del periférico *Timer* 0.

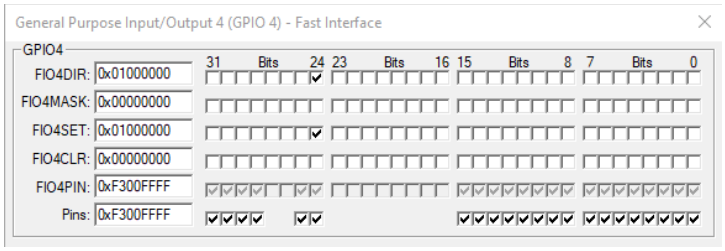

Figura 2-6 Visualización de los registros internos del periférico GPIO4.

A continuación ejecutaremos el programa con el menú *Debug* → *Run* o bien F5 o con la herramienta rápida  .

- **Visualización de la señal en el analizador lógico:**

A continuación vamos a visualizar la señal de salida P4.24 en el analizador lógico. Para ello pulsaremos la herramienta *Analysis Windows* → *Logic Analyzer* o de forma alternativa con el menú *View* → *Analysis Windows* → *Logic Analyzer*

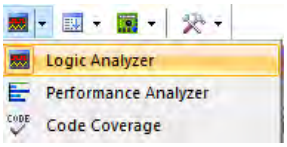

Figura 2-7 Herramienta rápida del analizador lógico virtual de *Keil µVision*.

Dentro de la ventana del analizador lógico, para seleccionar las señales que queremos visualizar pulsaremos en *Setup*.

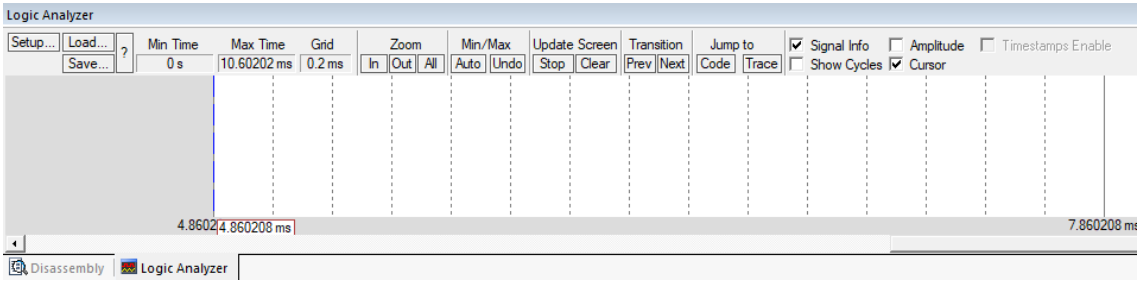

Figura 2-8 Vista inicial del analizador lógico virtual de *Keil µVision*.

Aquí tendremos que añadir nuestra señal en la pestaña *New (Insert)* donde añadiremos la señal que queremos visualizar, que en nuestro caso es FIO4PIN.24 (P4.24).

Figura 2-9 Inserción de las señales que queremos visualizar en el analizador lógico.

La herramienta *software* nos hará una máscara en el pin 24. En *Display Type* seleccionaremos el tipo de señal que en este caso es tipo *Bit*.

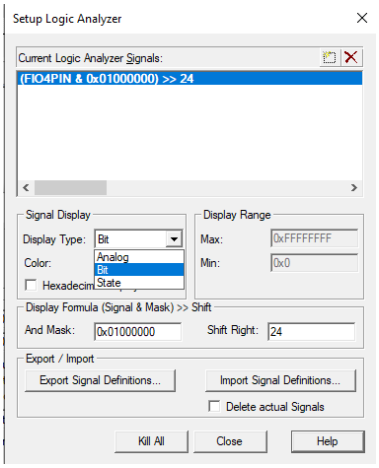

Figura 2-10 Cambio del tipo de visualización de la señal.

El analizador lógico nos permite medir los valores temporales de la señal marcando los ítems *Cursor* y *Signal Info*

Figura 2-11 Activación de los cursores y de la información de la señal.

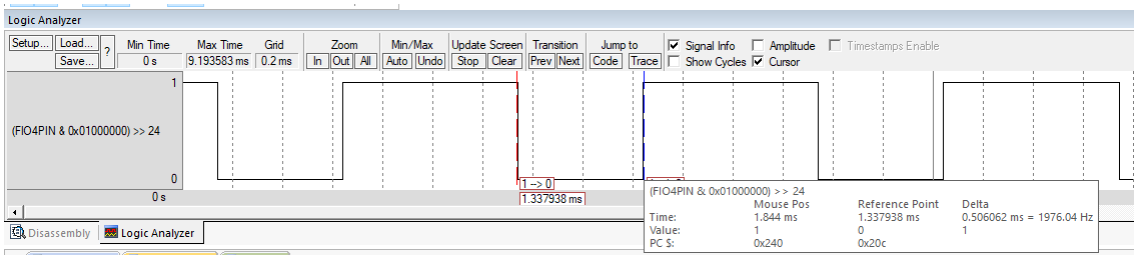

Figura 2-12 Tiempo de la señal a valor bajo (500 µs).

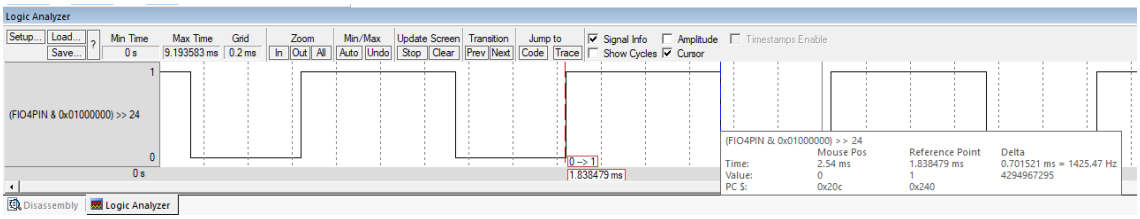

Figura 2-13 Tiempo de la señal a valor alto (700 µs).

- **Visualización de la señal en el osciloscopio:**

Después de haber compilado, enlazado, simulado y depurado el programa, comprobaremos con el osciloscopio, que la señal se genera correctamente, observando y midiendo sus características.

El pin escogido es el P4.24, tenemos que irnos al esquemático de la placa MCB2300 y ver que este se corresponde con el pin 127 y localizarlo en la placa para conectar el osciloscopio.

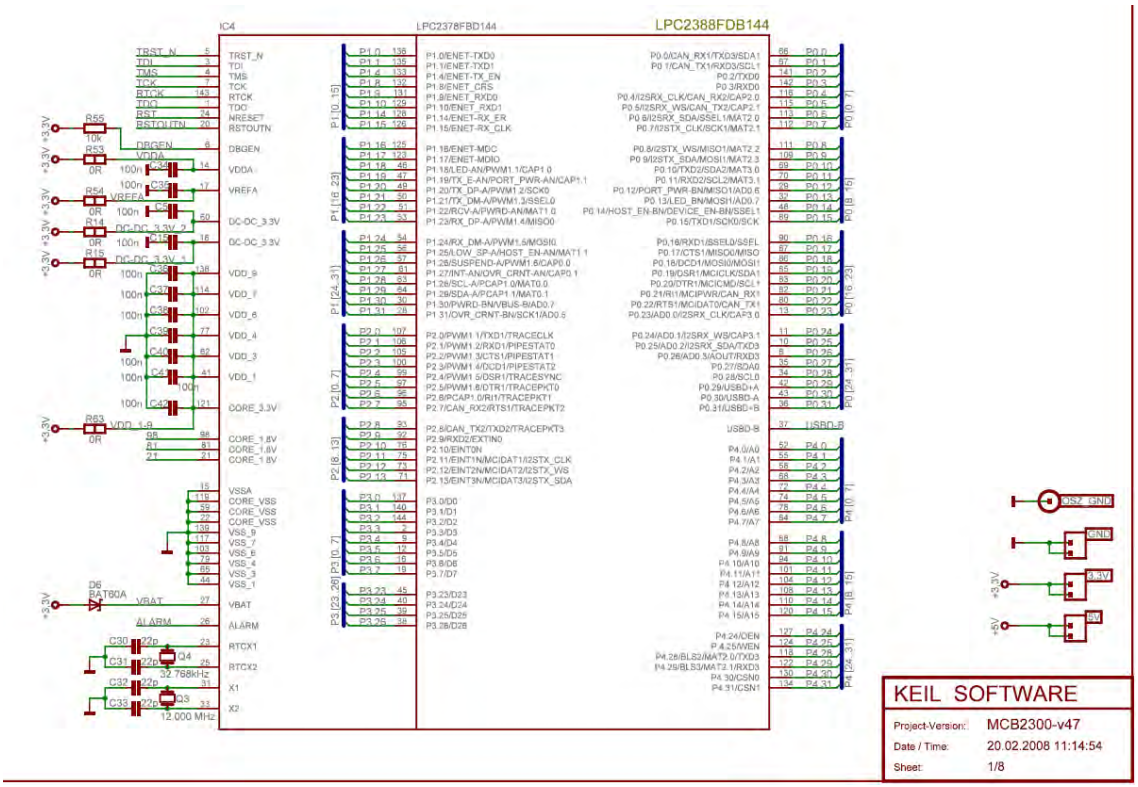

Figura 2-14 Esquemático de la placa MCB2300.