

## SISTEMAS EMPOTRADOS

### 3º Grado en Ingeniería Informática

## PRÁCTICA 3

### MANEJO DE GPIO Y TIMERS II

---

#### 3.1 *Introducción*

En esta práctica vamos a poner en práctica los conocimientos adquiridos en la práctica anterior, utilizando dos pines del puerto y dos temporizadores.

Queremos 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 microsegundos y a alta 700 microsegundos; la segunda tendrá una duración a baja de 200 microsegundos y a alta de 300 microsegundos. Se comprobará con el osciloscopio que estas señales se generan correctamente.

Los pines escogidos para visualizar las dos señales son P4.24 y P4.25 que se corresponden con los pines 127 y 124 respectivamente, como se puede observar en el documento donde se incluye el esquemático de la placa MCB2300 (disponible en la plataforma moodle).

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.

#### 3.2 *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 comprobación real con el osciloscopio.

### 3.3 *Material utilizado*

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

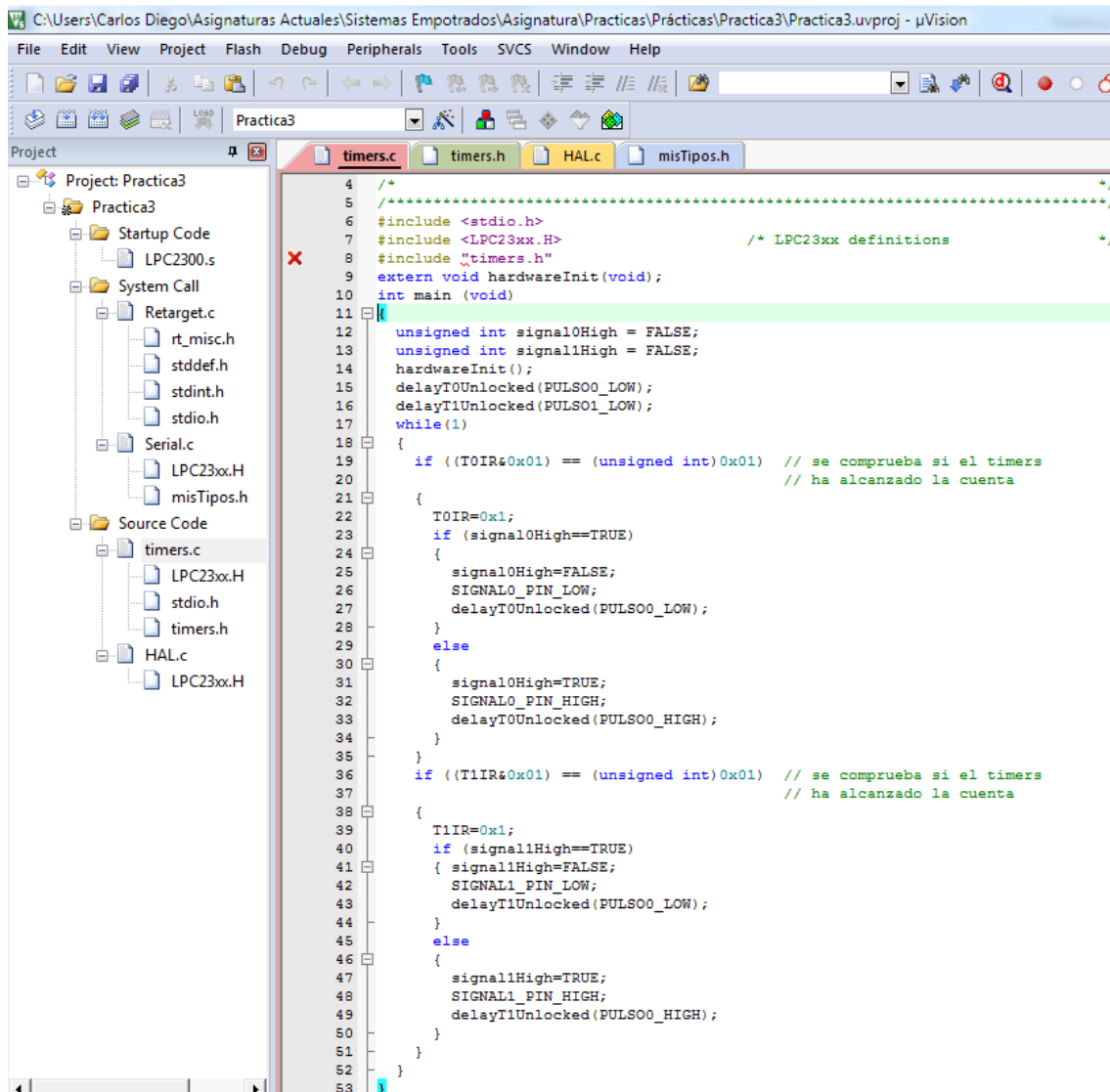
- Ordenador personal con Windows XP (mínimo) con el software Keil  $\mu$  Vision 5 instalado y el *pack* correspondiente a nuestra placa.
- Placa de desarrollo Keil MCB2300.
- Adaptador USB–JTAG de la familia ULINK para depurar programas.
- Dos cables USB A–B conectados a dos puertos USB disponibles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

### 3.4 *Desarrollo de la práctica*

Crear un nuevo proyecto (practica3) 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
- § timers.h
- § HAL.c

Estos ficheros los tienen ustedes en la plataforma Moodle.



```

4  /*
5  /*.....*/
6  #include <stdio.h>
7  #include <LPC23xx.H>          /* LPC23xx definitions */
8  #include "timers.h"
9  extern void hardwareInit(void);
10 int main (void)
11 {
12     unsigned int signal0High = FALSE;
13     unsigned int signal1High = FALSE;
14     hardwareInit();
15     delayT0Unlocked(PULSO0_LOW);
16     delayT1Unlocked(PULSO1_LOW);
17     while(1)
18     {
19         if ((T0IR&0x01) == (unsigned int)0x01) // se comprueba si el timers
20                                                 // ha alcanzado la cuenta
21         {
22             T0IR=0x1;
23             if (signal0High==TRUE)
24             {
25                 signal0High=FALSE;
26                 SIGNAL0_PIN_LOW;
27                 delayT0Unlocked(PULSO0_LOW);
28             }
29             else
30             {
31                 signal0High=TRUE;
32                 SIGNAL0_PIN_HIGH;
33                 delayT0Unlocked(PULSO0_HIGH);
34             }
35         }
36         if ((T1IR&0x01) == (unsigned int)0x01) // se comprueba si el timers
37                                                 // ha alcanzado la cuenta
38         {
39             T1IR=0x1;
40             if (signal1High==TRUE)
41             {
42                 signal1High=FALSE;
43                 SIGNAL1_PIN_LOW;
44                 delayT1Unlocked(PULSO1_LOW);
45             }
46             else
47             {
48                 signal1High=TRUE;
49                 SIGNAL1_PIN_HIGH;
50                 delayT1Unlocked(PULSO1_HIGH);
51             }
52         }
53     }
  
```

**Figura 3-1 Programa principal de la aplicación.**

El fichero `timers.h` para configurar los dos temporizadores tendría que ser algo como el de la figura 3-2. Hay que observar la configuración de los registros `FIO4SET3` y `FIO4CLR3` en el manual del fabricante. Así mismo también debemos estudiar la configuración de ambos temporizadores `T0` y `T1`.

```

timers.c | timers.h | HAL.c | misTipos.h
1  /*****
2  /* timers.h: definiciones para la practica de timers y algunas funciones */
3  /* Sistemas Empotrados Universidad de Cordoba */
4  /*****
5  /*
6  /*****
7
8  #define PULSO0_LOW 2 //duracion del pulso a nivel bajo
9  #define PULSO0_HIGH 3 //duracion del pulso a nivel alto
10 #define PULSO1_LOW 5 //duracion del pulso a nivel bajo
11 #define PULSO1_HIGH 7 //duracion del pulso a nivel alto
12 #define SIGNAL0_PIN_HIGH FIO4SET3 = 0x01; // Pin señal 0 a alto P4.24
13 #define SIGNAL0_PIN_LOW FIO4CLR3 = 0x01; // Pin señal 0 a bajo P4.24
14 #define SIGNAL1_PIN_HIGH FIO4SET3 = 0x02; // Pin señal 1 a alto P4.25
15 #define SIGNAL1_PIN_LOW FIO4CLR3 = 0x02; // Pin señal 1 a alto P4.25
16 #define FALSE (unsigned int)0x00000000
17 #define TRUE (unsigned int)0x00000001
18
19 /*****
20 /* delayT0Unlocked */
21 /*****
22 /* Esta funcion arranca el timer 0 y programa el registro match0 */
23 /*****
24 void delayT0Unlocked(unsigned int delayInDecimaMiliseg)
25 {
26     TOTCR = 0x02; /* reset timer */
27     TOMR0 = delayInDecimaMiliseg * (12000000 / 10000-1);
28     TOMCR = 0x07; /* timer on match */
29     TOTCR = 0x01; /* inicia timer y para cuando se llegue al final de cuenta */
30 }
31 /*****
32 /* delayT1Unlocked */
33 /*****
34 /* Esta funcion arranca el timer 1 y programa el registro match0 */
35 /*****
36 void delayT1Unlocked(unsigned int delayInDecimaMiliseg)
37 {
38     T1TCR = 0x02; /* reset timer */
39     T1MR0 = delayInDecimaMiliseg * (12000000 / 10000-1);
40     T1MCR = 0x07; /* timer on match */
41     T1TCR = 0x01; /* inicia timer y para cuando se llegue al final de cuenta*/
42 }
43
  
```

Figura 3-2 Fichero de configuración de los temporizadores.

En este caso, el fichero HAL.c para inicializar el hardware quedaría de la forma siguiente:

```

timers.c  timers.h  HAL.c  misTipos.h
1  /**
2  /* HAL.C: funciones generales que acceden al hardware */
3  /* Sistemas Empotrados Universidad de Cordoba */
4  /**
5  /*
6  /**
7  #include <LPC23xx.H> /* LPC23xx definitions */
8  /**
9  /* pinesSignalInit */
10 /**
11 /* Esta funcion configura los pines P4.24 y P4.25 como salida */
12 /**
13 void pinesSignalInit(void)
14 {
15     PINSEL9 = 0x00000000;
16     PINMODE9 = 0x00000000;
17     FIO4DIR3 = 0x03;
18 }
19 /**
20 /* timer0Init */
21 /**
22 /* Esta funcion configura el timer 0 con los parámetros que no cambian durante*/
23 /* la aplicacion */
24 /**
25 void timer0Init(void)
26 {
27     TOPR = 0x00; /* activa el preescalador a cero */
28 }
29 /**
30 /* timer1Init */
31 /**
32 /* Esta funcion configura el timer 1 con los parámetros que no cambian durante*/
33 /* la aplicacion */
34 /**
35 void timer1Init(void)
36 {
37     PCONP |= (0 << 2); // Habilito Timer 1, power y reloj
38     T1PR = 0x00; /* activa el preescalador a cero */
39 }
40 /**
41 /* hardwareInit */
42 /**
43 /* Esta funcion se llama al comienzo del programa para inicializar el Hardware*/
44 /**
45 void hardwareInit(void)
46 {
47     pinesSignalInit(); // Configura los pines del circuito
48     timer0Init(); // Inicializa el timer 0
49     timer1Init(); // Inicializa el timer 1
50 }

```

Figura 3-3 Fichero HAL.c para acceder al hardware.

El fichero misTipos.h para definir los tipos de variables, se corresponde con el siguiente:

```
1  /*****  
2  /* misTipos.h: definiciones de tipos de variables para practicas con LPC2378 */  
3  /* Sistemas Empotrados Universidad de Cordoba */  
4  /*****  
5  /*  
6  /*****  
7  #ifndef __misTipos_H  
8  #define __misTipos_H  
9  /* Byte */  
10 #define UINT8 unsigned char  
11 #define INT8 char  
12 /*16 bits */  
13 #define UINT16 unsigned short int  
14 #define INT16 short int  
15 /*32 bits WORD para el LPC2378 */  
16 #define UINT32 unsigned int  
17 #define INT32 int  
18 /* Tipos para control */  
19 #define STATUS UINT32  
20 /* Booleanas */  
21 #define BOOL UINT32  
22 #define FALSE (unsigned int)0x00000000  
23 #define TRUE (unsigned int)0x00000001  
24 /* flags */  
25 #define FLAG BOOL  
26 #endif
```

**Figura 3-4** Fichero para la definición de tipos de variables.

Al igual que en la práctica anterior, después de haber compilado, enlazado, simulado y depurado el programa, comprobaremos con el osciloscopio, que ambas señales se generan correctamente, observando y midiendo sus características. Los pines son P4.24 (127) y P4.25 (124) que tendrán que localizarlos en la placa.