

## 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

Project      Práctica5.c    Práctica5.h    IRQ.c    HAL.c    misTipos.h    Resumen.txt

```

1  /*****
2  * Practica 5: Timers software con VIC.
3  ****
4  /* Aquí se debe hacer una pequeña descripción de los objetivos de la práctica */
5  /* Apellidos y nombre del alumno/s:
6  ****
7  /* Sistemas Empotrados. 3º de Graduado en Ingeniería Informática
8  /* Universidad de Córdoba
9  ****
10
11 #include <stdio.h>
12 #include <LPC23xx.H>           /* LPC23xx definitions */
13 #include "misTipos.h"
14 #include "Práctica5.h"
15 /* variables globales */
16 UINT32 timersw0;
17 UINT32 timersw1;
18 BOOL signal0High;
19 BOOL signal1High;
20 extern void hardwareInit(void);
21 ****
22 /* delayT0Unlocked
23 /* Esta función arranca el timer 0 y programa el registro match0
24 */
25 ****
26 void delayT0Unlocked(unsigned int delayInDecimaMiliseg)
27 {
28     TOTCR = 0x02; /* reset timer */
29     TOMR0 = delayInDecimaMiliseg * 12000000 / 10000;
30     TOMCR = 0x07; /* timer on match */
31     TOTCR = 0x01; /* inicia timer y para cuando se llegue al final de cuenta */
32 }
33 int main (void)
34 {
35     /*inicializaciones de variables globales*/
36     signal0High = TRUE;
37     signal1High = TRUE;
38     timersw0=SIGNAL0_HIGHTIME;
39     timersw1=SIGNAL1_HIGHTIME;
40     // inicialización hardware
41     hardwareInit();
42     delayT0Unlocked(1);           /* temporizo una décima de milisegundo */
43     while (1);
44 }
```

- ✓ El fichero cabecera para las definiciones práctica5.h queda de la forma siguiente:

```

1  ****
2  /* Práctica5.h: definiciones para la práctica de timers y algunas funciones */
3  /* Sistemas Empotrados. Universidad de Córdoba */
4  ****
5
6  #include <LPC23xx.H>
7
8 #define SIGNAL0_LOWTIME 3 //duración del pulso 0 a nivel bajo en décimas de milisegundo
9 #define SIGNAL0_HIGHTIME 2 //duración del pulso 0 a nivel alto en décimas de milisegundo
10 #define SIGNAL1_LOWTIME 5 //duración del pulso 1 a nivel bajo en décimas de milisegundo
11 #define SIGNAL1_HIGHTIME 7 //duración del pulso 1 a nivel alto en décimas de milisegundo
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 bajo P4.25
16
```

- ✓ 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
1  /***************************************************************************** /
2  /* IRQ.C: Rutinas de servicio a la interrupción (ISR)                      */
3  /* Sistemas Empotrados. Universidad de Córdoba                                */
4  /***************************************************************************** /
5  #include <LPC23xx.H>                                         /* LPC23xx definitions */
6  #include "Practica5.h"
7  #include "misTipos.h"
8  extern UINT32 timersw0;
9  extern UINT32 timerswl;
10 extern BOOL signal0High;
11 extern BOOL signallHigh;
12 extern void delayT0Unlocked(unsigned int delayInDecimaMiliseg);
13 /***************************************************************************** /
14 /* Timer0 IRQ: Rutina de servicio a la interrupción del timer 0          */
15 /***************************************************************************** /
16 __irq void T0_IRQHandler (void) {
17     timersw0--;
18     timerswl--;
19     if (timersw0==0)    /* comprueba si el timer software 0 ha finalizado */
20     {
21         if (signal0High==TRUE)
22         {
23             SIGNAL0_PIN_LOW;
24             signal0High=FALSE;
25             timersw0=SIGNAL0_LOWTIME;
26         }
27         else
28         {
29             SIGNAL0_PIN_HIGH;
30             signal0High=TRUE;
31             timersw0=SIGNAL0_HIGHTIME;
32         }
33     }
34     if (timerswl==0)    /* comprueba si el timer software 1 ha finalizado */
35     {
36         if (signallHigh==TRUE)
37         {
38             SIGNAL1_PIN_LOW;
39             signallHigh=FALSE;
40             timerswl=SIGNAL1_LOWTIME;
41         }
42         else
43         {
44             SIGNAL1_PIN_HIGH;
45             signallHigh=TRUE;
46             timerswl=SIGNAL1_HIGHTIME;
47         }
48     }
49     TOIR      = 1;      /* Clear interrupt flag
50     VICVectAddr = 0;    /* Acknowledge Interrupt
51     delayT0Unlocked(1); /* Vuelvo a arrancar el timer una vez programado */
52 }
53 }
```

- ✓ 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
1  /************************************************************************
2  * HAL.C: funciones generales que acceden al hardware
3  * Sistemas Empotrados. Universidad de Córdoba
4  ****
5
6  #include <LPC23xx.H>                                /* LPC23xx definitions */
7  #include "Practica5.h"
8  /* Import external IRQ handlers from IRQ.c file */
9  extern __irq void T0_IRQHandler (void);
10 /* ****
11 /* pinesSignalInit
12 /* Esta función configura los pines P4.24 y P4.25 como salida
13 /* ****
14 void pinesSignalInit(void)
15 {
16     PINSEL9 = 0x00000000;
17     PINMODE9 = 0x00000000;
18     FIO4DIR3 = 0x03;
19 }
20 /* ****
21 /* timer0Init
22 /* Esta función configura el timer 0 con los parámetros que no cambian
23 /* durante la aplicación
24 /* ****
25 void timer0Init(void)
26 {
27     T0PR = 0x00; /* activa el preescalador a cero */
28     //controlador de interrupciones vectorizadas
29     VICVectAddr4 = (unsigned long)T0_IRQHandler;      /* Set Interrupt Vector */
30     VICVectPriority4 = 15;                            /* Priority use it for Timer0 Interrupt */
31     VICIntEnable = (1 << 4);                      /* Enable Timer0 Interrupt */
32 }
33 /* ****
34 /* hardwareInit
35 /* Esta función se llama al comienzo del programa para inicializar el Hardware
36 /* ****
37 void hardwareInit(void)
38 {
39     pinesSignalInit(); // Configura los pines del circuito
40     timer0Init();    // Inicializa el timer 0
41 }
```

- ✓ 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.