

SISTEMAS EMPOTRADOS

3º Grado en Ingeniería Informática

PRÁCTICA 5

TIMERS SOFTWARE CON INTERRUPCIONES

5.1. 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 micro y su comprobación real con el osciloscopio.

5.2. Material utilizado

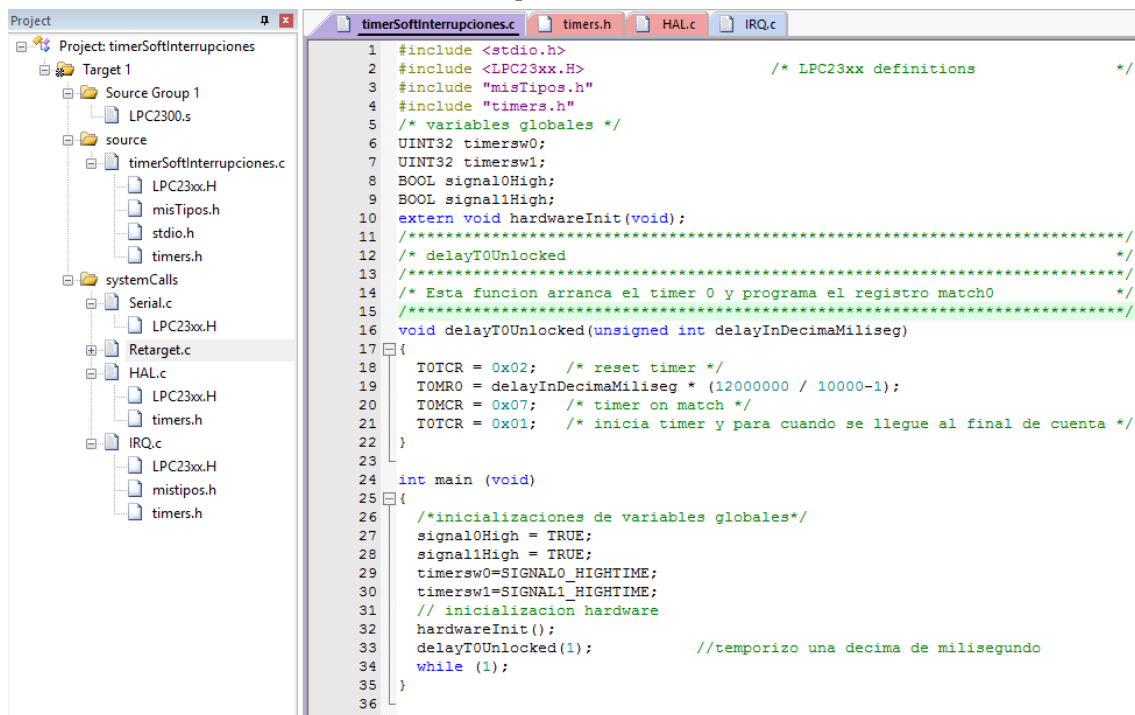
El material necesario para la realización de esta práctica es el siguiente:

- Ordenador personal con Windows XP (mínimo) con el software Keil μ 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.

5.3. 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 microsegundos y a alta 700 microsegundos y 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.
- ✓ 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
 - timers.h
 - HAL.c
 - IRQ.c
- ✓ Crear un nuevo fichero fuente practica5.c



```
1 #include <stdio.h>
2 #include <LPC23xx.H> /* LPC23xx definitions */
3 #include "misTipos.h"
4 #include "timers.h"
5 /* variables globales */
6 UINT32 timersw0;
7 UINT32 timersw1;
8 BOOL signal0High;
9 BOOL signal1High;
10 extern void hardwareInit(void);
11 /* delayT0Unlocked */
12 /* Esta funcion arranca el timer 0 y programa el registro match0 */
13 /* Este fichero se encarga de generar las interrupciones */
14 void delayT0Unlocked(unsigned int delayInDecimaMiliseg)
15 {
16     TOTCR = 0x02; /* reset timer */
17     TOMR0 = delayInDecimaMiliseg * (12000000 / 10000-1);
18     TOMCR = 0x07; /* timer on match */
19     TOTCR = 0x01; /* inicia timer y para cuando se llegue al final de cuenta */
20 }
21
22 int main (void)
23 {
24     /*inicializaciones de variables globales*/
25     signal0High = TRUE;
26     signal1High = TRUE;
27     timersw0=SIGALO_HIGHTIME;
28     timersw1=SIGALI_HIGHTIME;
29     // inicializacion hardware
30     hardwareInit();
31     delayT0Unlocked(1); /*temporizo una decima de milisegundo
32     while (1);
33 }
34
35
36
```

- ✓ El fichero IRQ.c con el que se controlan las interrupciones quedaría así:

```

timerSoftInterrupciones.c  timers.h  HAL.c  IRQ.c
2  /* IRQ.C: manejadores de interrupcion */
3  /* Sistemas Empotrados Universidad de Cordoba */
4  /****** */
5  #include <LPC23xx.H> /* LPC23xx definitions */
6  #include "timers.h"
7  #include "mistipos.h"
8  extern UINT32 timersw0;
9  extern UINT32 timersw1;
10 extern BOOL signal0High;
11 extern BOOL signal1High;
12 extern void delayT0Unlocked(unsigned int delayInDecimaMiliseg);
13 /* Timer0 IRQ */
14 irq void T0_IRQHandler (void) {
15     // codigo ISR
16     timersw0--;
17     timersw1--;
18     if (timersw0==0) // comprueba si el timer sw 0 ha finalizado
19     {
20         if (signal0High==TRUE)
21         {
22             SIGNAL0_PIN_LOW;
23             signal0High=FALSE;
24             timersw0=SIGNAL0_LOWTIME;
25         }
26         else
27         {
28             SIGNAL0_PIN_HIGH;
29             signal0High=TRUE;
30             timersw0=SIGNAL0_HIGHTIME;
31         }
32     }
33     if (timersw1==0) // comprueba si el timer sw 1 ha finalizado
34     {
35         if (signal1High==TRUE)
36         {
37             SIGNAL1_PIN_LOW;
38             signal1High=FALSE;
39             timersw1=SIGNAL1_LOWTIME;
40         }
41         else
42         {
43             SIGNAL1_PIN_HIGH;
44             signal1High=TRUE;
45             timersw1=SIGNAL1_HIGHTIME;
46         }
47     }
48     TOIR = 1; /* Clear interrupt flag */
49     VICVectAddr = 0; /* Acknowledge Interrupt */
50     delayT0Unlocked(1); // vuelvo a arrancar el timer una vez programado
51 }
  
```

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

```

timerSoftInterrupciones.c  timers.h  HAL.c  IRQ.c
1  /*****/
2  /* HAL.C: funciones generales que acceden al hardware */
3  /* Sistemas Empotrados Universidad de Cordoba */
4  /*****/
5  #include <LPC23xx.H> /* LPC23xx definitions */
6  #include "timers.h"
7  /* Import external IRQ handlers from IRQ.c file */
8  extern __irq void T0_IRQHandler (void);
9  /*****/
10 /* pinesSignalInit */
11 /*****/
12 /* Esta funcion 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 /*****/
23 /* Esta funcion configura el timer 0 con los parámetros que no cambian */
24 /* durante la aplicacion */
25 /*****/
26 void timer0Init(void)
27 {
28     TOPR = 0x00; /* activa el preescalador a cero */
29     //controlador de interrupciones
30     VICVectAddr4 = (unsigned long)T0_IRQHandler; /* Set Interrupt Vector */
31     VICVectCntl4 = 15; /* Priority use it for Timer0 Interrupt */
32     VICIntEnable = (1 << 4); /* Enable Timer0 Interrupt */
33 }
34 /*****/
35 /* hardwareInit */
36 /*****/
37 /* Esta funcion se llama al comienzo del programa para inicializar el Hardware*/
38 /*****/
39 void hardwareInit(void)
40 {
41     pinesSignalInit(); // Configura los pines del circuito
42     timer0Init(); // Inicializa el timer 0
43 }
  
```

✓ El fichero timers.h es igual al de la práctica anterior:

```

timerSoftInterrupciones.c  timers.h*  HAL.c  IRQ.c
1  /*****/
2  /* timers.h: definiciones para la practica de timers y algunas funciones */
3  /* Sistemas Empotrados Universidad de Cordoba */
4  /*****/
5  #define SIGNAL0_LOWTIME 3 //duracion del pulso 0 a nivel bajo en decimas de milisecondo
6  #define SIGNAL0_HIGHTIME 2 //duracion del pulso 0 a nivel alto en decimas de milisecondo
7  #define SIGNAL1_LOWTIME 5 //duracion del pulso 1 a nivel bajo en decimas de milisecondo
8  #define SIGNAL1_HIGHTIME 7 //duracion del pulso 1 a nivel alto en decimas de milisecondo
9  #define SIGNAL0_PIN_HIGH FIO4SET3 = 0x01; // Pin señal 0 a alto P4.24
10 #define SIGNAL0_PIN_LOW FIO4CLR3 = 0x01; // Pin señal 0 a bajo P4.24
11 #define SIGNAL1_PIN_HIGH FIO4SET3 = 0x02; // Pin señal 1 a alto P4.25
12 #define SIGNAL1_PIN_LOW FIO4CLR3 = 0x02; // Pin señal 1 a bajo P4.25
  
```

✓ Las dos señales deberán observarse en los dos canales del osciloscopio.