

SISTEMAS EMPOTRADOS 3º Grado en Ingeniería Informática

PRÁCTICA 4

CONTROLADOR DE INTERRUPCIONES VECTORIZADAS (VIC)

4.1 Introducción

En esta práctica vamos a obtener el mismo resultado de la práctica anterior, pero programando los *timers* por interrupciones a través del controlador de interrupciones vectorizadas (VIC) del LPC2378.

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 μ s y a alta 700 μ s; la segunda tendrá una duración a baja de 200 μ s y a alta de 300 μ s. 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). Son los mismos que en la práctica anterior.

Para completar el estudio del controlador de interrupciones vectorizadas debemos mirar como referencia el capítulo 6 del manual de usuario LPC23xx de NXP Semiconductors (páginas 85–95).

Para el estudio teórico utilizamos el apartado 3.8 (páginas 76–84) 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>.

4.2 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.
- Que se observe la diferencia entre programar los *timers* por interrupciones utilizando el VIC en lugar de por *polling* (sondeo) como en las prácticas anteriores.
- Diseño de un programa de aplicación y su comprobación real con el osciloscopio.

4.3 Material utilizado

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

- Ordenador personal con *Windows* con el *software Keil μ Vision 5* instalado y el *software 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.

4.4 Desarrollo de la práctica

Los *timers* se programarán por interrupciones utilizando el VIC. El final de la cuenta del *timer* se detectará muestreando los bits de control.

Crear un nuevo proyecto (practica4) 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

Crear un nuevo fichero fuente practica4.c

```

1  /*****
2  /* Práctica 4: Timers hardware 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 "Practica4.h"
14 #include "misTipos.h"
15
16 extern void hardwareInit(void);
17
18 unsigned int signal0High = FALSE;
19 unsigned int signal1High = FALSE;
20
21 /*****
22 /* Timer0 IRQ
23 /*****
24 __irq void T0_IRQHandler (void) {
25
26     if (signal0High==TRUE)
27     {
28         signal0High=FALSE;
29         SIGNAL0_PIN_LOW;
30         delayT0Unlocked(PULSO0_LOW);
31     }
32     else
33     {
34         signal0High=TRUE;
35         SIGNAL0_PIN_HIGH;
36         delayT0Unlocked(PULSO0_HIGH);
37     }
38
39     T0IR      = 1; /* Clear interrupt flag */
40     VICVectAddr = 0; /* Acknowledge Interrupt */
41 }
42 /*****
43 /* Timer1 IRQ
44 /*****
45 __irq void T1_IRQHandler (void) {
46
47     if (signal1High==TRUE)
48     {
49         signal1High=FALSE;
50         SIGNAL1_PIN_LOW;
51         delayT1Unlocked(PULSO1_LOW);
52     }
53     else
54     {
55         signal1High=TRUE;
56         SIGNAL1_PIN_HIGH;
57         delayT1Unlocked(PULSO1_HIGH);
58     }
59
60     T1IR      = 1; /* Clear interrupt flag */
61     VICVectAddr = 0; /* Acknowledge Interrupt */
62 }
63
64 int main (void)
65 {
66     hardwareInit();
67
68     delayT0Unlocked(PULSO0_LOW);
69     delayT1Unlocked(PULSO1_LOW);
70
71     while(1) {}
72 }
73

```

Figura 4-1 Rutina principal de la aplicación.

En esta práctica la configuración de los *timers* la haremos por interrupciones, es decir, programando el circuito VIC, por lo que el fichero HAL.c quedará como en la figura 4-2.

```

1  /* ***** */
2  /* HAL.C: funciones generales que acceden al hardware */
3  /* Capa de abstracción del hardware (Hardware Abstract Layer) */
4  /* Sistemas Empotrados. Universidad de Córdoba */
5  /* ***** */
6  extern __irq void T0_IRQHandler (void);
7  extern __irq void T1_IRQHandler (void);
8  #include <LPC23xx.H> /* LPC23xx definitions */
9  /* ***** */
10 /* pinesSignalInit: Esta función configura los pines P4.24 y P4.25 */
11 /* como salida */
12 /* ***** */
13 void pinesSignalInit(void)
14 {
15     PINSEL9 = 0x00000000;
16     PINMODE9 = 0x00000000;
17     FIO4DIR3 = 0x03;
18 }
19 /* ***** */
20 /* timer0Init: Esta función configura el timer 0 con los parámetros */
21 /* que no cambian durante la aplicación */
22 /* ***** */
23 void timer0Init(void)
24 {
25     T0PR = 0x00; /* activa el preescalador a cero */
26
27     VICVectAddr4 = (unsigned long) T0_IRQHandler; /* Set Interrupt Vector */
28     VICVectCntl4 = 15; /* use it for Timer0 Interrupt */
29     VICIntEnable = (1 << 4); /* Enable Timer0 Interrupt */
30 }
31 /* ***** */
32 /* timer1Init: Esta función configura el timer 1 con los parámetros */
33 /* que no cambian durante la aplicación */
34 /* ***** */
35 void timer1Init(void)
36 {
37     PCONP |= (0 << 2); /* Habilito Timer 1, power y reloj */
38
39     T1PR = 0x00; /* activa el preescalador a cero */
40
41     VICVectAddr5 = (unsigned long) T1_IRQHandler; /* Set Interrupt Vector */
42     VICVectCntl5 = 15; /* use it for Timer1 Interrupt */
43     VICIntEnable = (1 << 5); /* Enable Timer1 Interrupt */
44 }
45 /* ***** */
46 /* hardwareInit: Esta función se llama al comienzo del programa para */
47 /* inicializar el Hardware */
48 /* ***** */
49 void hardwareInit(void)
50 {
51     pinesSignalInit(); /* Configura los pines del circuito */
52     timer0Init(); /* Inicializa el timer 0 */
53     timer1Init(); /* Inicializa el timer 1 */
54 }

```

Figura 4-2: HAL.c Funciones generales que acceden al *hardware*.

Hay una serie de registros que deben programarse obligatoriamente por cada fuente de interrupción en el VIC:

- VICVectAddrXX: coloca el vector de la fuente de interrupción (dirección de la rutina de interrupción).
- VICVectPriorityXX: coloca la prioridad de la fuente de interrupción XX. Por defecto se encuentra al valor 15.
- VICIntEnable: registro que habilita las interrupciones.

Hay que señalar que, en concreto, en el circuito de nuestra placa LPC2378, el registro que controla la prioridad es VICVectCntlXX.

El fichero cabecera Practica4.h queda, como en las prácticas anteriores, de la forma siguiente:

```

1  /*****
2  /* Practica4.h: definiciones para las prácticas de timers y algunas funciones */
3  /* Sistemas Empotrados. Universidad de Córdoba */
4  /*****
5  #define PULSO0_LOW 2 /* duración del pulso 0 a nivel bajo */
6  #define PULSO0_HIGH 3 /* duración del pulso 0 a nivel alto */
7  #define PULSO1_LOW 5 /* duración del pulso 1 a nivel bajo */
8  #define PULSO1_HIGH 7 /* duración del pulso 1 a nivel alto */
9  #define SIGNAL0_PIN_HIGH FIO4SET3 = 0x01; /* Pin señal 0 a nivel alto P4.24 */
10 #define SIGNAL0_PIN_LOW FIO4CLR3 = 0x01; /* Pin señal 0 a nivel bajo P4.24 */
11 #define SIGNAL1_PIN_HIGH FIO4SET3 = 0x02; /* Pin señal 1 a nivel alto P4.25 */
12 #define SIGNAL1_PIN_LOW FIO4CLR3 = 0x02; /* Pin señal 1 a nivel bajo P4.25 */
13
14 #include <LPC23xx.H> /* LPC23xx definitions */
15 /*****
16 /* delayT0Unlocked: Esta función arranca el timer 0 y programa el registro match0 */
17 /*****
18 void delayT0Unlocked(unsigned int delayInDecimaMiliseg)
19 {
20     T0TCR = 0x02; /* reset timer */
21     T0MR0 = delayInDecimaMiliseg * 12000000 / 10000;
22     T0MCR = 0x07; /* timer on match */
23     T0TCR = 0x01; /* inicia timer y para cuando se llegue al final de cuenta */
24 }
25 /*****
26 /* delayT1Unlocked: Esta función arranca el timer 1 y programa el registro match1 */
27 /*****
28 void delayT1Unlocked(unsigned int delayInDecimaMiliseg)
29 {
30     T1TCR = 0x02; /* reset timer */
31     T1MR0 = delayInDecimaMiliseg * 12000000 / 10000;
32     T1MCR = 0x07; /* timer on match */
33     T1TCR = 0x01; /* inicia timer y para cuando se llegue al final de cuenta */
34 }

```

Figura 4-3 Fichero *header* Practica4.h.

El fichero donde definimos los tipos de variables es igual al de las prácticas anteriores.

```

1  /*****
2  /* misTipos.h: definiciones de tipos de
3  /* variables para prácticas con LPC2378
4  /* Sistemas Empotrados. Universidad de Córdoba
5  *****/
6
7  #ifndef __misTipos_H
8  #define __misTipos_H
9
10 /* Byte */
11 #define UINT8 unsigned char
12 #define INT8 char
13
14 /*16 bits */
15 #define UINT16 unsigned short int
16 #define INT16 short int
17
18 /*32 bits WORD para el LPC2378 */
19 #define UINT32 unsigned int
20 #define INT32 int
21
22 /* Tipos para control */
23 #define STATUS UINT32
24
25 /* Booleanas */
26 #define BOOL UINT32
27 #define FALSE (unsigned int)0x00000000
28 #define TRUE (unsigned int)0x00000001
29
30 /* flags */
31 #define FLAG BOOL
32 #define ESC 0x1B
33 #define OK 0x01
34 #endif
35

```

Figura 4-4 Fichero *header* misTipos.h.

Al igual que en las prácticas anteriores, después de haber compilado, enlazado, simulado y depurado el programa, comprobaremos con el osciloscopio y el analizador lógico de *Keil μ Vision*, 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.

El resultado de la simulación en el depurador sería la siguiente:

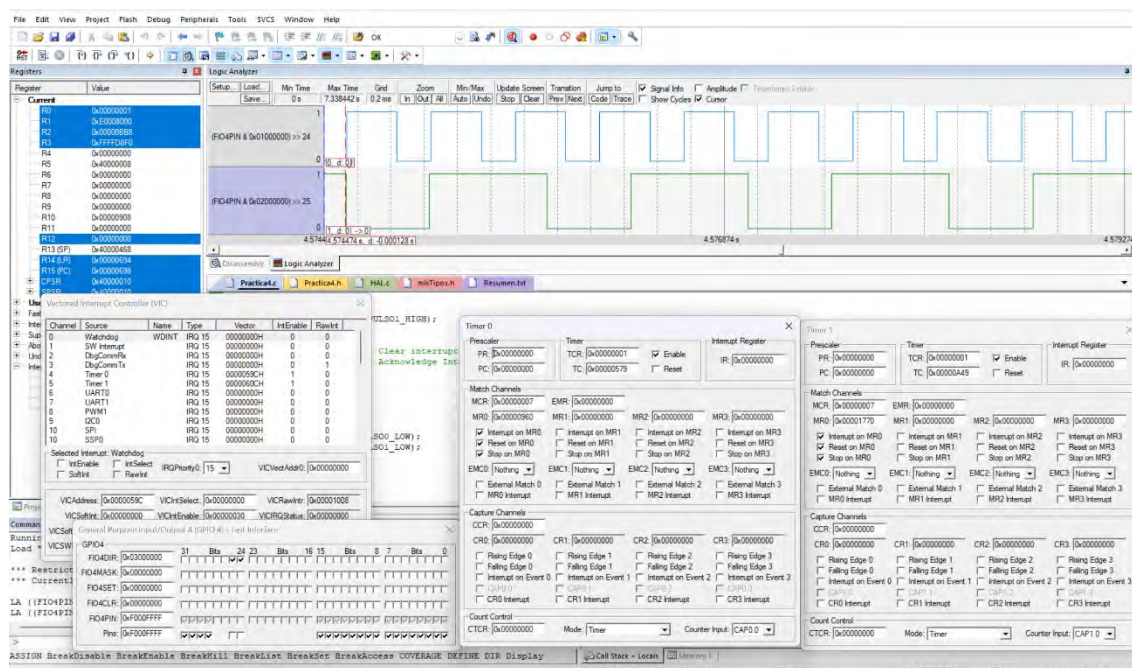


Figura 4-5 Resultado de la simulación.