

# Práctica 5 - Interrupciones

Jaime Arana Manuel Ferrero 3ºA

### **Temporizador con interrupciones**

```
1 = #include <xc.h>
2
     #include <stdint.h>
     #include "Temporizador.h"
3
   #include "Pic32Ini.h"
5
     #define PIN PULSADOR 5
6
7
     #define PIN LED0 0
     #define PIN LED3 3
8
9
     // se encienda y apague 2 seg = 1 seg apagado + 1 seg encendido
10
11
     #define T_PARPADEO 1000 // en ms
12
13
     /* CÁLCULO DE PR2 */
14 \( \bar{-} \) // PR2 = (retardo / (divisor * 200 ns)) - 1 <= 65535
   L // PR2 = (1 ms / (1 * 200 ns)) - 1 = 4999
15
     #define PERIODO TIMER2 4999 // valor de PR2
16
     #define PRIORIDAD TIMER2 2
17
18
     #define SUBPRIORIDAD_TIMER2 1
19
     #define PRESCALER 0
20
21 = int main(void) {
         int pulsador, puerto;
22
23
         uint32 t ticks ant, ticks act;
24
25
         // configurar entradas y salidas
         TRISB |= (1 << PIN_PULSADOR);
26
         TRISC &= ~(1 << PIN LED0);
27
28
         TRISC &= ~(1 << PIN LED3);
29
         // LED 0 y LED 3 empiezan apagados
30
         puerto = 0x0000;
31
         puerto |= (1 << PIN_LED0);
32
33
         puerto |= (1 << PIN LED3);
         LATC |= puerto;
34
35
```



```
// llamamos a la función para inicializar Timer2
36
37
          InicializarTimer2(PERIODO TIMER2, PRIORIDAD TIMER2, SUBPRIORIDAD TIMER2, PRESCALER);
38
39
         // activamos las interrupciones
         INTCONbits.MVEC = 1;
40
         asm(" ei");
41
42
43
         ticks ant = TicksDesdeArr();
44
45
          while(1) {
              ticks act = TicksDesdeArr();
46
              if((ticks act - ticks ant) > T PARPADEO) { // con la resta medimos el tiempo transcurrido
47
                  ticks ant = ticks act;
48
                  LATCINV = (1 << PIN LED3); // invertimos el valor de LED
49
50
51
52
              pulsador = (PORTB >> PIN PULSADOR) & 1;
53
              if(pulsador == 0) { // está pulsado
54
                 LATCCLR = (1 << PIN_LED0); // LED = 0 --> encendido
55
56
              } else {
                  LATCSET = (1 << PIN LED0); // LED = 1 --> apagado
57
58
59
60
```

# Interrupción Timer2

```
1 = #include <xc.h>
     #include <stdio.h>
3
     #include "Pic32Ini.h"
4
5
     #define BITS PRESCALER 4
6
7
   void InicializarTimer2(int tiempo, int prioridad, int subprioridad, int valor prescaler) {
8
         T2CON = 0; // se para el temporizador
         TMR2 = 0; //se pone la cuenta a 0
9
10
         IFSObits.T2IF = 0; // borrar bit fin de cuenta (flag)
         PR2 = tiempo;
11
12
         IECObits.T2IE = 1; // enable = 1
13
         IPC2bits.T2IP = prioridad; // prioridad = 2
         IPC2bits.T2IS = subprioridad; // subprioridad = 1
14
15
         T2CON = 0x8000; // Timer2 --> ON = 1 + reloj interno seleccionado
16
17
         T2CON |= (valor prescaler << BITS PRESCALER); // añadimos el prescaler correspondiente
18
19
20
     // variable contador que se irá iterando
21
     static uint32 t ticks = 0;
22
```



```
23
       attribute ((vector( TIMER 2 VECTOR), interrupt(IPL2SOFT), nomimps16))
   void InterrupcionTimer2(void) {
24
25
          // borrar flag para que no vuelve a entrar
26
          // solo entrar cuando se produzca interrupción
          IFSObits.T2IF = 0;
27
28
29
          // rutina de atención a la interrupcion
30
          ticks++;
31
32
33
  uint32 t TicksDesdeArr(void) {
34
          uint32_t temp; // variable para hacer copia atomica
35
36
          asm(" di"); // inhabilitar interrupción
37
          temp = ticks;
          asm(" ei"); // habilitar
39
40
         return temp;
41
```

#### **Ejercicio**

En este apartado se pide modificar el programa anterior y no utilizar las instrucciones LATCCLR, LATCSET y LATCINV. Estas instrucciones consiguen hacer modificar el registro en una instrucción de escritura, haciendo que esta sea atómica. Por lo tanto, se protege la escritura del registro frente a las interrupciones. En nuestro caso no hemos detectado ningún problema ya que ha dado la casualidad de que la interrupción no ha interrumpido el proceso de escritura del registro.

Solo se muestra la sección de código donde se producen los cambios.

```
36
          // llamamos a la función para inicializar Timer2
          InicializarTimer2(PERIODO_TIMER2, PRIORIDAD_TIMER2, SUBPRIORIDAD_TIMER2, PRESCALER);
37
38
39
          // activamos las interrupciones
40
          INTCONbits.MVEC = 1:
          asm(" ei");
41
42
          ticks_ant = TicksDesdeArr();
43
44
45
          while(1) {
              ticks act = TicksDesdeArr();
46
              if((ticks act - ticks ant) > T PARPADEO) { // con la resta medimos el tiempo transcurrido
47
                  ticks ant = ticks act;
48
                  LATC ^= (1 << PIN_LED3); // invertimos el LED 0
49
50
51
52
              pulsador = (PORTB >> PIN PULSADOR) & 1;
53
54
              if(pulsador == 0) { // está pulsado
55
                  LATC &= ~(1 << PIN LED0); // ponemos el LED = 0 --> endencendido
56
57
                  LATC |= (1 << PIN LED0); // ponemos el LED = 1 --> apagado
58
59
60
```



## **Interrupciones múltiples**

```
1 = #include <xc.h>
     #include <stdint.h>
3
     #include "TemporizadorTimer2.h" // Timer2
     #include "TemporizadorTimer3.h" // Timer3
4
   #include "Pic32Ini.h"
5
6
7
     #define PIN PULSADOR 5
     #define PIN LED0 0
8
     #define PIN_LED2 2
9
     #define PIN LED3 3
10
11
12
     #define T PARPADEO LED3 1000 // en ms
13
     #define T PARPADEO LED2 500 // en ms
14
15
     /* CÁLCULO DE PR2 */
16 ☐ // PR2 = (retardo / (divisor * 200 ns)) - 1 <= 65535
   L // PR2 = (1 ms / (1 * 200 ns)) - 1 = 4999
17
     #define PERIODO_TIMER2 4999 // valor de PR2
18
     #define PRIORIDAD_TIMER2 2
19
20
     #define SUBPRIORIDAD TIMER2 1
21
     #define PRESCALER_TIMER2 0
22
    /* CÁLCULO DE PR3 */
23
24 - // PR3 = (retardo / (divisor * 200 ns)) - 1 <= 65535
   L // PR3 = (1 ms / (1 * 200 ns)) - 1 = 4999
25
     #define PERIODO TIMER3 4999 // valor de PR3
26
     #define PRIORIDAD TIMER3 4
27
     #define SUBPRIORIDAD TIMER3 0
28
     #define PRESCALER_TIMER3 0
29
30
31  int main(void) {
32
         int pulsador, puerto;
         uint32_t ticks_ant, ticks_act, ticks_ant2, ticks_act2;
33
34
35
         // configurar entradas y salidas
36
         TRISB |= (1 << PIN_PULSADOR);
37
         TRISC &= \sim (1 << PIN LED0);
38
         TRISC &= \sim (1 << PIN LED2);
39
         TRISC &= ~(1 << PIN LED3);
40
41
         // LEDs empiezan apagados
42
         puerto = 0x00000;
43
         puerto |= (1 << PIN_LED0);
44
         puerto |= (1 << PIN_LED2);
         puerto |= (1 << PIN LED3);
45
         LATC |= puerto;
46
47
48
          // llamamos a las funciones para inicializar los timers
49
         InicializarTimer2(PERIODO TIMER2, PRIORIDAD TIMER2, SUBPRIORIDAD TIMER2, PRESCALER TIMER2);
         InicializarTimer3(PERIODO TIMER3, PRIORIDAD TIMER3, SUBPRIORIDAD TIMER3, PRESCALER TIMER3);
50
52
         INTCONbits.MVEC = 1; // ponemos CPU en modo multvector
53
         asm(" ei"); // activamos interrupciones
54
         ticks ant = TicksDesdeArrTimer2();
55
56
         ticks ant2 = TicksDesdeArrTimer3();
57
```



```
58
          while(1) {
59
              // para el LED3 se utiliza el TIMER2
60
              ticks_act = TicksDesdeArrTimer2();
61
              if( (ticks_act - ticks_ant) > T_PARPADEO_LED3){ // con la resta medimos el tiempo transcurrido
                   ticks ant = ticks act;
62
                   LATCINV = (1 << PIN LED3);
63
64
65
              // para el LED2 se utiliza TIMER3
66
              ticks act2 = TicksDesdeArrTimer3();
67
68
              if( (ticks_act2 - ticks_ant2) > T_PARPADEO_LED2){
                   ticks ant2 = ticks act2;
69
70
                  LATCINV = (1 << PIN LED2);
71
72
73
              pulsador = (PORTB >> PIN PULSADOR) & 1;
74
75
              if(pulsador == 0) { // está pulsado
                 LATCCLR = (1 << PIN_LED0); // LED = 0 --> encendido
76
77
              } else {
78
                 LATCSET = (1 << PIN_LED0); // LED = 1 --> apagado
79
80
          }
81
```



## Interrupción Timer2

```
1 = #include <xc.h>
     #include <stdint.h>
3
   #include "Pic32Ini.h"
4
     #define BITS PRESCALER 4
5
6
7
   void InicializarTimer2(int tiempo, int prioridad, int subprioridad, int valor prescaler) {
         T2CON = 0;
8
         TMR2 = 0;
9
10
         IFSObits.T2IF = 0;
         PR2 = tiempo;
11
         IECObits.T2IE = 1; // enable = 1
13
         IPC2bits.T2IP = prioridad; // prioridad = 2
         IPC2bits.T2IS = subprioridad; // subprioridad = 1
14
15
16
         T2CON = 0x8000; // Timer2 --> ON = 1 + reloj interno seleccionado
17
         T2CON |= (valor_prescaler << BITS_PRESCALER); // añadimos el prescaler correspondiente
18
19
20
     // variable contador que se irá iterando
21
     static uint32_t ticks = 0;
22
23
       attribute ((vector( TIMER 2 VECTOR), interrupt(IPL2SOFT), nomimps16))
24 - void InterrupcionTimer2(void) {
25
         // Se borra el flag de interrupción para no volver a
26
         // entrar en esta rutina hasta que se produzca una nueva
27
         // interrupción.
         IFSObits.T2IF = 0;
28
29
         // rutina de atención a la interrupcion
30
31
          ticks++;
32
33
   uint32 t TicksDesdeArrTimer2(void) {
          uint32_t temp; // variable para hacer copia atomica
35
36
37
          asm(" di"); // inhabilitar interrupción
          temp = ticks;
38
          asm(" ei"); // habilitar
40
41
         return temp;
42
```



### Interrupción Timer3

```
1 = #include <xc.h>
2
     #include <stdint.h>
3
     #include "Pic32Ini.h"
4
5
     #define BITS PRESCALER 4
6
  void InicializarTimer3(int tiempo, int prioridad, int subprioridad, int valor prescaler) {
8
         T3CON = 0;
9
         TMR3 = 0;
10
         IFSObits.T3IF = 0;
11
         PR3 = tiempo;
         IECObits.T3IE = 1; // enable = 1
13
         IPC3bits.T3IP = prioridad; // prioridad = 4
         IPC3bits.T3IS = subprioridad; // subprioridad = 0
14
15
         T3CON = 0x8000; // Timer3 --> ON = 1 + reloj interno seleccionado
16
17
         T3CON |= (valor prescaler << BITS PRESCALER); // añadimos el prescaler correspondiente
18
19
20
21
     // variable contador que se irá iterando
22
     static uint32 t ticks = 0;
24
      _attribute__((vector(_TIMER_3_VECTOR), interrupt(IPL4SOFT), nomimps16))
25 - void InterrupcionTimer3 (void) {
26
          // Se borra el flag de interrupción para no volver a
         // entrar en esta rutina hasta que se produzca una nueva
27
28
         // interrupción.
29
         IFSObits.T3IF = 0;
30
         // rutina de atención a la interrupcion
31
         ticks++;
32
33
34
35
   uint32_t TicksDesdeArrTimer3(void) {
36
         uint32_t temp; // variable para hacer copia atomica
37
38
         asm(" di"); // inhabilitar interrupción
39
         temp = ticks;
         asm(" ei"); // habilitar
40
41
42
         return temp;
43
```



#### Juego de velocidad

#### Primero se adjunta el código sin la sección opcional del ejercicio.

```
1 = #include <xc.h>
      #include <stdint.h>
2
     #include "TemporizadorSeccion4Simple.h"
3
     #include "Pic32Ini.h"
4
5
6
     #define PIN PULSADOR 5
7
     #define PIN LED 1
8
9
     #define MAX PULSACIONES 5
10
     /* CÁLCULO DE PR2 */
11
12 \( \begin{aligned} \text{// PR2} = (retardo / (divisor * 200 ns)) - 1 <= 65535 \)
   // PR2 = (1 seg / (256 * 200 ns)) - 1 = 19531
13
     #define PERIODO TIMER2 19531 // le damos al jugador l seg para pulsar x5 veces el pulsador
14
15
     #define PRIORIDAD TIMER2 2
     #define SUBPRIORIDAD TIMER2 1
16
     #define PRESCALER 256
17
18
   int main(void) {
19
20
          int puls ant, puls act;
21
          // configurar entradas y salidas
          TRISB |= (1 << PIN PULSADOR);
23
          TRISC &= ~(1 << PIN LED);
24
25
          LATCSET = (1 << PIN_LED); // poner a 1 --> apagado
26
27
28
          // llamamos a la función para inicializar Timer2
29
          InicializarTimer2 (PERIODO TIMER2, PRIORIDAD TIMER2, SUBPRIORIDAD TIMER2, PRESCALER);
30
31
          INTCONbits.MVEC = 1; // ponemos la CPU en modo multivector
32
          asm(" ei"); // activamos las interrupciones
33
          puls_ant = (PORTB >> PIN_PULSADOR) & 1;
34
35
36
          while(1) {
37
              puls act = (PORTB >> PIN PULSADOR) & 1;
38
39
              if((puls ant != puls act) && (puls act == 0)) { // detectar flanco bajada pulsador
                  asm( "di"); // desactivamos interupciones
40
                  num_pulsaciones++;
41
42
                  asm(" ei"); // activamos interrupciones
43
44
                  if(num pulsaciones >= MAX PULSACIONES) {
45
                      LATCCLR = (1 << PIN LED); // poner a 0 --> encendido
46
47
48
              puls_ant = puls_act;
49
50
```



```
1 = #include <xc.h>
     #include <stdint.h>
 2
    #include "Pic32Ini.h"
 3
 4
 5
     #define BITS PRESCALER 4
 6
 7 p void InicializarTimer2 (int tiempo, int prioridad, int subprioridad, int valor prescaler) {
         T2CON = 0; // se para el temporizador
8
         TMR2 = 0; //se pone la cuenta a 0
9
10
          IFSObits.T2IF = 0; // borrar bit fin de cuenta (flag)
11
         PR2 = tiempo;
12
         IECObits.T2IE = 1; // enable = 1
13
         IPC2bits.T2IP = prioridad; // prioridad = 2
         IPC2bits.T2IS = subprioridad; // subprioridad = 1
14
15
16
         T2CON = 0x8000; // Timer2 --> ON = 1 + reloj interno seleccionado
17
         T2CON |= (valor_prescaler << BITS_PRESCALER); // añadimos el prescaler correspondiente
18
19
20
     // variable global compartida que cuenta las pulsaciones
21
     static uint16 t num pulsaciones = 0;
22
      _attribute__((vector(_TIMER_2_VECTOR), interrupt(IPL2SOFT), nomips16))
23
24 - void InterrupcionTimer2 (void) {
25
         // Se borra el flag de interrupción para no volver a
         // entrar en esta rutina hasta que se produzca una nueva
26
27
         // interrupción.
28
         IFSObits.T2IF = 0;
29
30
         // rutina de atención interrupción
31
         // se ponen a 0 las pulsaciones al saltar la interrupción
32
         num pulsaciones = 0;
33
```



#### Código con la opción de esperar 4 segundos y apagar el LED

```
1 = #include <xc.h>
 2
      #include <stdint.h>
 3
     #include "TemporizadorSeccion4.h"
 4
      #include "Pic32Ini.h"
 5
 6
     #define PIN PULSADOR 5
 7
     #define PIN LED 1
 8
 9
     #define MAX PULSACIONES 5
10
11
     /* CÁLCULO DE PR2 */
12 - // PR2 = (retardo / (divisor * 200 ns)) - 1 <= 65535
    L // PR2 = (1 seg / (256 * 200 ns)) - 1 = 19531
13
     #define PERIODO TIMER2 19531 // le damos al jugador l seg para pulsar x5 veces el pulsador
14
     #define PRIORIDAD TIMER2 2
15
16
     #define SUBPRIORIDAD TIMER2 1
     #define PRESCALER 256
17
18
     #define RETARDO 4
19
20
21
      extern uintl6 t fin partida;
22
23 = int main(void) {
24
         int puls_ant, puls_act;
25
26
         // configurar entradas y salidas
         TRISB |= (1 << PIN PULSADOR);
27
28
         TRISC &= ~(1 << PIN LED);
29
30
         LATCSET = (1 << PIN LED); // poner a 1 --> apagado
31
32
          // llamamos a la función para inicializar Timer2
33
          InicializarTimer2 (PERIODO TIMER2, PRIORIDAD TIMER2, SUBPRIORIDAD TIMER2, PRESCALER);
34
         INTCONbits.MVEC = 1; // ponemos la CPU en modo multivector
35
36
          asm(" ei"); // activamos las interrupciones
37
         puls ant = (PORTB >> PIN PULSADOR) & 1;
38
39
          while(1) {
40
41
              puls act = (PORTB >> PIN PULSADOR) & 1;
42
43
              if((puls ant != puls act) && (puls act == 0)) { // detectar flanco bajada pulsador
                  asm( "di"); // desactivamos interupciones
44
45
                  num pulsaciones++;
                  asm(" ei"); // activamos interrupciones
46
47
                  if (num pulsaciones >= MAX PULSACIONES) {
48
49
                      fin partida = 1;
                      LATCCLR = (1 << PIN LED); // poner a 0 --> encendido
50
51
52
53
              puls ant = puls act;
54
55
```



```
1 = #include <xc.h>
     #include <stdint.h>
   #include "Pic32Ini.h"
3
4
5
     #define BITS PRESCALER 4
    #define PIN LED 1
 6
8
   void InicializarTimer2(int tiempo, int prioridad, int subprioridad, int valor prescaler) {
9
          T2CON = 0; // se para el temporizador
10
          TMR2 = 0; //se pone la cuenta a 0
         IFSObits.T2IF = 0; // borrar bit fin de cuenta (flag)
11
12
         PR2 = tiempo;
13
         IECObits.T2IE = 1; // enable = 1
         IPC2bits.T2IP = prioridad; // prioridad = 2
14
15
         IPC2bits.T2IS = subprioridad; // subprioridad = 1
16
17
         T2CON = 0x8000; // Timer2 --> ON = 1 + reloj interno seleccionado
18
         T2CON |= (valor prescaler << BITS PRESCALER); // añadimos el prescaler correspondiente
19
20
21
      // variable global compartida que cuenta las pulsaciones
     static uint16_t num_pulsaciones = 0;
22
23
     uintl6_t fin_partida = 0;
24
       attribute ((vector( TIMER 2 VECTOR), interrupt(IPL2SOFT), nomips16))
25
26
   void InterrupcionTimer2(void) {
27
         // Se borra el flag de interrupción para no volver a
         // entrar en esta rutina hasta que se produzca una nueva
28
29
         // interrupción.
30
         static int contador = 0;
31
         IFSObits.T2IF = 0;
32
33
         // rutina de atención interrupción
         // se ponen a 0 las pulsaciones al saltar la interrupción
34
35
         if (fin partida == 0) {
36
             num pulsaciones = 0;
          } else {
37
38
             contador++;
39
             if (contador == 4) {
40
                 fin_partida = 0;
                 LATCSET = (1 << PIN LED);
41
42
43
         }
44
```