

Práctica 3 - Temporizadores

Jaime Arana Manuel Ferrero 3ºA

Generación de retardos:

```
1 = #include <xc.h>
   #include "Pic32Ini.h"
2
 3
      #define BIT LED0 0
 4
5
  r⊟ int main(void) {
 6
7
          //configuramos LED 0 como salida
8
          TRISC &= ~(1 << BIT LED0);
9
          // configuramos LED 0 para que esté apagado
10
11
          LATC |= (1 << BIT LED0);
12
13
          T2CON = 0; // se para el temporizador 2
14
          TMR2 = 0; // se pone la cuenta a 0
          IFSObits.T2IF = 0; // se borra el bit de fin de cuenta
15
16
          /* CALCULO DEL VALOR PR2 */
17
18
   白
          // 1 Hz = periodo completo, por lo tanto,
19
          // LED encendido 0.5 seg
20
          // LED apagado 0.5 seg
21
          // PR2 = (retardo / (div * 200 ns)) - 1 --> <= 65535
22
          // en este caso tenemos PR2 = (0.5 seg / (64 * 200 ns)) - 1 = 39061
23
          PR2 = 39061;
24
          T2CON = 0x8060; // Timer 2 --> ON = 1 + reloj interno + prescaler = 64
25
26
27
          while(1) {
28
              if(IFSObits.T2IF == 1) {
29
                  // invertimos el valor del LED 0
30
                  LATC ^= 1;
31
                  // borramos el flag del Timer 2
                  IFSObits.T2IF = 0;
32
33
34
35
36
          return 0;
37
```



Función para generar retardos:

```
1
      #include <xc.h>
 2
3
      #define NUM MAX 16BITS 65535
      #define BITS PRESCALER 4
4
6
      int Retardo (uint16 t retardo ms)
7
  rĢ {
8
          int prescaler[] = {1, 2, 4, 8, 16, 32, 64, 256};
9
          int i = -1;
10
          int prescaler ok = 1;
11
          float res;
12
13
          // primero vamos a comprobar el valor del retardo
          if(retardo_ms <= 0) {</pre>
14
15
              // no entramos en el bucle while
              i = 8;
16
17
18
          // iteramos por cada valor de prescaler posible
19
          while ((i < 7) && (prescaler_ok == 1)){
20
21
              i++;
22
              res = (retardo ms / (prescaler[i] * 0.0002)) - 1;
              if(res <= NUM MAX 16BITS){</pre>
23
24
                  // PR2 correcto nos salimos del bucle while
25
                  prescaler ok = 0;
26
              }
27
          }
28
29 🖃
          // solo si se ha conseguido generar el retardo
          // se configura el temporizador
30
          if(prescaler ok == 0) {
31
32
              T2CON = 0; // Se para el temporizador 2
              TMR2 = 0; // Cuenta a 0
33
              IFSObits.T2IF = 0; // Se borra el bit de fin de cuenta
34
35
              PR2 = (uint16 t)res;
36
              T2CON = 0x8000; // Timer 2 encendido --> ON = 1
37
              T2CON |= (prescaler << BITS PRESCALER);
38
              while (IFSObits.T2IF == 0) {
39
                 ;// Espera el fin del temporizador
40
41
              T2CON = 0;
42
              TMR2 = 0:
43
44
              IFSObits.T2IF = 0; // Se borra el bit de fin de cuenta
45
46
47
          return prescaler ok;
48
```



```
#ifndef RETARDO H
     #define RETARDO H
2
3
   = #ifdef __cplusplus
 4
5
     extern "C" {
 6
     #endif
7
8
     int Retardo(uint16 t retardo ms);
9
10
  #ifdef __cplusplus
11
12
     }
   - #endif
13
14
   #endif /* RETARDO H */
15
```

Generación de retardos II:

```
1 = #include <xc.h>
      #include "Pic32Ini.h"
3
    #include "Retardo.h"
4
5
      #define BIT LED0 0
6
  r int main(void) {
7
8
          //configuramos LEC 0 como salida
9
10
          TRISC &= \sim (1 << BIT LED0);
11
          //configuramos LED 0 a 1 para que esté apagado
12
13
          LATC |= (1 << BIT LED0);
14
15
          // retardo en ms
          uint16 t retardo = 500;
16
17
18
          while(1) {
19
              int k = Retardo(retardo);
              // si la función crea correctamente el retardo
20
   口
21
              // devuelve un 0
              if(k == 0) {
22
23
                  LATC ^= 1;
24
              }
25
26
27
          return 0;
28
```



LED latiendo:

```
1 = #include <xc.h>
2
     #include "Pic32Ini.h"
    | #include "Retardo.h"
3
4
5
      #define BIT LED0 0
      #define MAX TEMP 20 // 20 ms
6
7
  r int main (void) {
8
9
          // configuramos LED como salida
          TRISC &= \sim (1 << BIT LED0);
10
11
12
          // configuramos LED a 1 para que esté apagado
          LATC |= (1 << BIT LED0);
13
14
15
          int cont, j;
16
17
          while(1) {
              for(j = 0; j <= MAX TEMP; j++) { // tiempo retardo ascendiendo</pre>
18
                   cont = Retardo(20-j); // retardo de 20-j ms
19
20
                   if(cont == 0) {
                       LATC ^= 1;
21
22
                   cont = Retardo(j); // retardo de j ms
23
                   if(cont == 0) {
24
                       LATC ^= 1;
25
26
27
28
              for(j = MAX TEMP-1; j >= 0; j--){ // tiempo retardo descendiendo
                   LATC ^= 1;
29
30
                   cont = Retardo(j); // retardo de j ms
31
                   if(cont == 0) {
                       LATC ^= 1;
32
33
                   }
34
35
                   // para j != 0 generamos un retardo de 20-j ms
                   // cuando j == 0 saltará a primer bucle for
36
37
                   // para empezar secuencia de nuevo
38
                   if(j != 0){
39
                       cont = Retardo(20-j);
40
41
              1
42
43
44
          return 0;
45
```