Práctica 3 – Temporizadores

Jaime Arana
Manuel Ferrero
3ºA

Generación de retardos:

```c
#include <xc.h>
#include "Pic32Ini.h"

#define BIT_LED0 0

int main(void) {
    //configuramos LED 0 como salida
    TRISC &= ~(1 << BIT_LED0);

    // configuramos LED 0 para que esté apagado
    LATC |= (1 << BIT_LED0);

    T2CON = 0; // se para el temporizador 2
    TMR2 = 0; // se pone la cuenta a 0
    IFS0bits.T2IF = 0; // se borra el bit de fin de cuenta

    /* CALCULO DEL VALOR PR2 */
    // 1 Hz = periodo completo, por lo tanto,
    // LED encendido 0.5 seg
    // LED apagado 0.5 seg
    // PR2 = (retardo / (div * 200 ns)) - 1 --> <= 65535
    // en este caso tenemos PR2 = (0.5 seg / (64 * 200 ns)) - 1 = 39061

    PR2 = 39061;
    T2CON = 0x8060; // Timer 2 --> ON = 1 + reloj interno + prescaler = 64

    while(1) {
        if(IFS0bits.T2IF == 1) {
            // invertimos el valor del LED 0
            LATC ^= 1;
            // borramos el flag del Timer 2
            IFS0bits.T2IF = 0;
        }
    }

    return 0;
}
```

Función para generar retardos:

```c
#include <xc.h>

#define NUM_MAX_16BITS 65535
#define BITS_PRESCALER 4

int Retardo(uint16_t retardo_ms)
{
    int prescaler[] = {1, 2, 4, 8, 16, 32, 64, 256};
    int i = -1;
    int prescaler_ok = 1;
    float res;

    // primero vamos a comprobar el valor del retardo
    if(retardo_ms <= 0) {
        // no entramos en el bucle while
        i = 8;
    }

    // iteramos por cada valor de prescaler posible
    while ((i < 7) && (prescaler_ok == 1)){
        i++;
        res = (retardo_ms / (prescaler[i] * 0.0002)) - 1;
        if(res <= NUM_MAX_16BITS){
            // PR2 correcto nos salimos del bucle while
            prescaler_ok = 0;
        }
    }

    // solo si se ha conseguido generar el retardo
    // se configura el temporizador
    if(prescaler_ok == 0) {
        T2CON = 0; // Se para el temporizador 2
        TMR2 = 0; // Cuenta a 0
        IFS0bits.T2IF = 0; // Se borra el bit de fin de cuenta
        PR2 = (uint16_t)res;
        T2CON = 0x8000; // Timer 2 encendido --> ON = 1
        T2CON |= (prescaler << BITS_PRESCALER);

        while(IFS0bits.T2IF == 0){
            ;// Espera el fin del temporizador
        }
        T2CON = 0;
        TMR2 = 0;
        IFS0bits.T2IF = 0; // Se borra el bit de fin de cuenta
    }

    return prescaler_ok;
}
```

```h
#ifndef RETARDO_H
#define RETARDO_H

#ifdef __cplusplus
extern "C" {
#endif

int Retardo(uint16_t retardo_ms);


#ifdef __cplusplus
}
#endif

#endif /* RETARDO_H */
```

Generación de retardos II:

```c
#include <xc.h>
#include "Pic32Ini.h"
#include "Retardo.h"

#define BIT_LED0 0

int main(void) {

    //configuramos LEC 0 como salida
    TRISC &= ~(1 << BIT_LED0);

    //configuramos LED 0 a 1 para que esté apagado
    LATC |= (1 << BIT_LED0);

    // retardo en ms
    uint16_t retardo = 500;

    while(1) {
        int k = Retardo(retardo);
        // si la función crea correctamente el retardo
        // devuelve un 0
        if(k == 0) {
            LATC ^= 1;
        }
    }

    return 0;
}
```

LED latiendo:

```c
#include <xc.h>
#include "Pic32Ini.h"
#include "Retardo.h"

#define BIT_LED0 0
#define MAX_TEMP 20 // 20 ms

int main (void) {
    // configuramos LED como salida
    TRISC &= ~(1 << BIT_LED0);

    // configuramos LED a 1 para que esté apagado
    LATC |= (1 << BIT_LED0);

    int cont, j;

    while(1) {
        for(j = 0; j <= MAX_TEMP; j++){ // tiempo retardo ascendiendo
            cont = Retardo(20-j); // retardo de 20-j ms
            if(cont == 0) {
                LATC ^= 1;
            }
            cont = Retardo(j); // retardo de j ms
            if(cont == 0) {
                LATC ^= 1;
            }
        }
        for(j = MAX_TEMP-1;j >= 0; j--){ // tiempo retardo descendiendo
            LATC ^= 1;
            cont = Retardo(j); // retardo de j ms
            if(cont == 0) {
                LATC ^= 1;
            }

            // para j != 0 generamos un retardo de 20-j ms
            // cuando j == 0 saltará a primer bucle for
            // para empezar secuencia de nuevo
            if(j != 0){
                cont = Retardo(20-j);
            }
        }
    }

    return 0;
}
```