4.4 Ejemplo 2

Utilizar instrucciones en ensamblador y el oscilador interno LFINTOSC...

En realidad, esto es una continuación del ejemplo anterior, pero se ocupa de un problema un poco más complicado... El propósito era hacer los LEDs en el puerto PORTB parpadear lentamente. Se puede realizar al introducir un valor suficiente grande para el parámetro del tiempo de retardo en la función Delay. No obstante, hay otra manera más eficiente para ejecutar el programa lentamente. Acuérdese de que este microcontrolador tiene un oscilador incorporado LFINTOSC que funciona a una frecuencia de 31kHz. Ahora llegó la hora de “darle una oportunidad”. El programa se inicia con el bucle do-while y se queda allí por 20 ciclos. Después da cada iteración, llega el tiempo de retardo de 100ms, indicado por un parpadeo relativamente rápido de los LEDs en el puerto PORTB. Cuando el programa salga de este bucle, el microcontrolador se inicia al utilizar el oscilador LFINTOSC como una fuente de señal de reloj. Los LEDs parpadean más lentamente aunque el programa ejecuta el mismo bucle do-while con un tiempo de retardo 10 veces más corto. Con el propósito de hacer evidentes algunas situaciones potencialmente peligrosas, se activan los bits de control por medio de las instrucciones en ensamblador. Dicho de manera sencilla, al entrar o salir una instrucción en ensamblador en el programa, el compilador no almacena los datos en un banco actualmente activo de la RAM, lo que significa que en esta sección de programa, la selección de bancos depende del registro SFR utilizado. Al volver a la sección de programa escrito en C, los bits de control RP0 y RP1 deben recuperar el estado que tenían antes de ‘la aventura en ensamblador’. En este programa, el problema se resuelve al utilizar la variable auxiliar saveBank, lo que guarda el estado de estos dos bits.

```
/* Cabecera ********************************************/

int k = 0;                          // Variable k es de tipo int
char saveBank;                      // Variable saveBank es de tipo char

void main() {
  ANSEL = 0;                        // Todos los pines de E/S se configuran como digitales
  ANSELH = 0;
  PORTB = 0;                        // Todos los pines del puerto PORTB se ponen a 0
  TRISB = 0;                        // Pines del puerto PORTB se configuran como salidas

  do {
  PORTB = ~PORTB;                   // Invertir el estado lógico del puerto PORTB
  Delay_ms(100);                    // Tiempo de retardo de 100mS
  k++;                              // Incrementar k en 1
  }
  while(k<20);                      // Quedarse en bucle hasta que k<20

  k=0;                              // Reiniciar variable k
  saveBank = STATUS & 0b01100000;   // Guardar el estado de los bits RP0 y RP1

                                    // (bits 5 y 6 del registro STATUS)
  asm {                             // Inicio de una secuencia en ensamblador
    bsf STATUS,RP0                  // Seleccionar el banco de memoria que contiene el
    bcf STATUS,RP1                  // registro OSCCON
    bcf OSCCON,6                    // Seleccionar el oscilador interno LFINTOSC
    bcf OSCCON,5                    // de frecuencia de 31KHz
    bcf OSCCON,4
    bsf OSCCON,0                    // Microcontrolador utiliza oscilador interno
  }                                 // Final de la secuencia en ensamblador

  STATUS &= 0b10011111;             // Bits RP0 y RP1 recuperan el estado original
  STATUS |= saveBank;

  do {
    PORTB = ~PORTB;                 // Invertir el estado lógico del puerto PORTB
    Delay_ms(10);                   // Tiempo de retardo de 10 mS
    k++;                            // Incrementar k en 1
  }
  while(k<20);                      // Quedarse en el bucle hasta que k<20
}
```