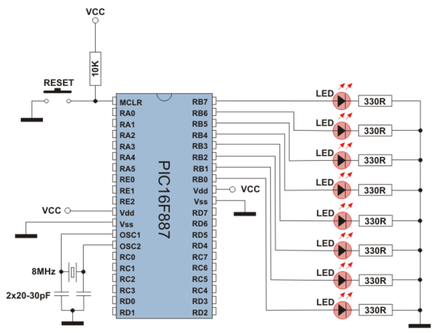
4.6 Ejemplo 4

Utilizar los temporizadores Timer0, Timer1 y Timer2. Utilizar interrupciones, declarar nuevas funciones...

Al analizar los ejemplos anteriores, es probable que se haya fijado en la desventaja de proporcionar tiempo de retardo por medio de la función Delay. En estos casos, el microcontrolador se queda 'estático' y no hace nada. Simplemente espera que transcurra una cierta cantidad de tiempo. Tal pérdida de tiempo es un lujo inaceptable, por lo que se deberá aplicar otro método. ¿Se acuerda usted del capítulo de los temporizadores? ¿Se acuerda de lo de interrupciones? Este ejemplo los conecta de una manera práctica. El esquema se queda inalterada, y el desafío sigue siendo presente. Es necesario proporcionar un tiempo de retardo suficiente largo para darse cuenta de los cambios en el puerto. Para este propósito se utiliza el temporizador Timer0 con el pre-escalador asignado. Siempre que se genere una interrupción con cada desbordamiento en el registro del temporizador, la variable cnt se aumenta automáticamente en 1 al ejecutarse cada rutina de interrupción. Cuando la variable llega al valor 400, el puerto PORTB se incrementa en 1. Todo el procedimiento se lleva a cabo "entre bastidores", lo que habilita al microcontrolador hacer otra tarea.



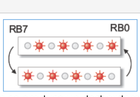

```
/*Cabecera*****************************************************/

unsigned cnt;          // Definir la variable cnt

void interrupt() {
  cnt++;               // Con una interrupción la cnt se incrementa en 1
  TMR0 = 96;           // El valor inicial se devuelve en el temporizador TMR0
  INTCON = 0x20;       // Bit T0IE se pone a 1, el bit T0IF se pone a 0
}

void main(){
  OPTION_REG = 0x84;   // Pre-escalador se le asigna al temporizador TMR0
  ANSEL = 0;           // Todos los pines de E/S se configuran como digitales
  ANSELH = 0;
  TRISB = 0;           // Todos los pines de puerto PORTB se configuran

                       // como salidas
  PORTB = 0x0;         // Reiniciar el puerto PORTB
  TMR0 = 96;           // Temporizador T0 cuenta de 96 a 255
  INTCON = 0xA0;       // Habilitada interrupción TMR0
  cnt = 0;             // A la variable cnt se le asigna un 0

  do {                 // Bucle infinito
    if (cnt == 400) {  // Incrementar el puerto PORTB después 400 interrupciones
      PORTB = PORTB++; // Incrementar número en el puerto PORTB en 1
      cnt = 0;         // Reiniciar la variable cnt
    }
  } while(1);

}
```

Siempre que se produzca un desbordamiento en el registro del temporizador TRM0, ocurre una interrupción.

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

unsigned short cnt;   // Definir la variable cnt

void interrupt() {
  cnt++ ;             // Con una interrupción la cnt se incrementa en 1
  PIR1.TMR1IF = 0;    // Reiniciar el bit TMR1IF
  TMR1H = 0x80;       // El valor inicial se devuelve en los registros
  TMR1L = 0x00;       // del temporizador TMR1H y TMR1L
}

void main() {
  ANSEL = 0;          // Todos los pines de E/S se configuran como digitales
  ANSELH = 0;
  PORTB = 0xF0;       // Valor inicial de los bits del puerto PORTB
  TRISB = 0;          // Pines del puerto PORTB se configuran como salidas
  T1CON = 1;          // Configurar el temporizador TMR1
  PIR1.TMR1IF = 0;    // Reiniciar el bit TMR1IF
  TMR1H = 0x80;       // Ajustar el valor inicial del temporizador TMR1
  TMR1L = 0x00;
  PIE1.TMR1IE = 1;    // Habilitar la interrupción al producirse un desbordamiento
  cnt = 0;            // Reiniciar la variable cnt
  INTCON = 0xC0;      // Interrupción habilitada (bits GIE y PEIE)

  do {                // Bucle infinito
    if (cnt == 76) {  // Cambiar el estado del puerto PORTB después de 76 interrupciones
      PORTB = ~PORTB; // Número en el puerto PORTB está invertido
      cnt = 0;        // Reiniciar la variable cnt
    }
  } while (1);
}
```

En este caso, una interrupción se habilita después de que se produzca un desbordamiento en el registro del temporizador TMR1 (TMR1H, TMR1L). Además, la combinación de los bits que varía en el puerto POTRB difiere de la en el ejemplo anterior.

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

unsigned short cnt;   // Definir la variable cnt

void Reemplazar() {
  PORTB = ~PORTB;     // Definir nueva función 'Reemplazar'
}                     // Función invierte el estado del puerto

void interrupt() {
  if (PIR1.TMR2IF) {  // Si el bit TMR2IF = 1,
    cnt++ ;           // Incrementar variable la cnt en 1
    PIR1.TMR2IF = 0;  // Reiniciar el bit y
    TMR2 = 0;         // Reiniciar el registro TMR2
  }
}

// main
void main() {
  cnt = 0;            // Reiniciar la variable cnt
  ANSEL = 0;          // Todos los pines de E/S se configuran como digitales
  ANSELH = 0;
  PORTB = 0b10101010; // Estado lógico en los pines del puerto PORTB
  TRISB = 0;          // Todos los pines del puerto PORTB se configuran como salidas
  T2CON = 0xFF;       // Configurar el temporizador T2
  TMR2 = 0;           // Valor inicial del registro del temporizador TMR2
  PIE1.TMR2IE = 1;    // Interrupción habilitada
  INTCON = 0xC0;      // Bits GIE y PEIE se ponen a 1

  while (1) {         // Bucle infinito
    if (cnt > 30) {   // Cambiar el estado del puerto PORTB después de
                      // más de 30 interrupciones
      Reemplazar();   // Función Reemplazar invierte el estado del puerto PORTB
      cnt = 0;        // Reiniciar la variable cnt
    }
  }
}
```



En este ejemplo, una interrupción ocurre después de que se produce un desbordamiento en el registro del temporizador TMR2. Para invertir el estado lógico de los pines del puerto se utiliza la función Reemplazar, que normalmente no pertenece al lenguaje C estándar.