PRÁCTICA 2

Comunicación serie (UART)

María González Herrero

Santiago Molpeceres Díaz

UNIVERSIDAD NEBRIJA

INDICE

INTRODUCCIÓN

Para esta segunda práctica hemos aprendido a configurar el módulo de comunicaciones serie UART. Para poder hacer los ejercicios, primero hemos realizado la configuración de los bits y de la UART tal y como hicimos en la sesión de la explicación de dicha práctica.

Para la UART hemos establecido reciclado los pines de nuestra anterior práctica (*Practica 1: Control de GPIO*).

- LED 1 en el pin RB3 (LED_RED) y LED 2 en el pin RA0 (LED_GREEN), ambos configurados como pines de salida.

```
TRISBbits.TRISB3 = 0;
TRISAbits.TRISA0 = 0;
delay_ms(10);

LATBbits.LATB3 = 0;
LATAbits.LATA0 = 0;
delay_ms(10);
uart_config(baud_9600);
```

- UART Rx y Tx, con los pines RC0 y RC1 respectivamente. Han sido configurados como muestra la imagen.

```
TRISCbits.TRISC0 = 1;
RPINR18bits.U1RXR = 16;
RPOR8bits.RP17R = 3;
```

- A parte de la configuración de registros propia de la UART, tanto de los registros U1MODE como el de los registros U1STA. La cual vimos en la sesión de la práctica según las explicaciones del profesor.

Ejercicio 1

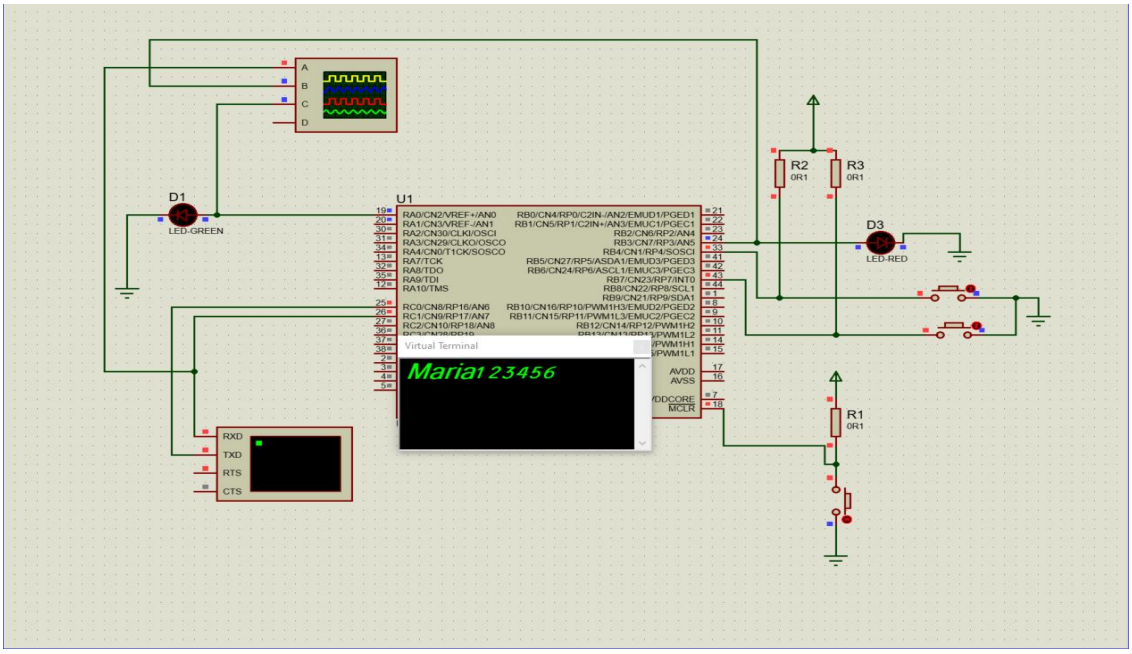
Para este primer ejercicio se pide que se imprima en la terminal virtual el nombre de un alumno y a continuación que se vaya aumentando cada 0,5 segundos.

Para ello, hemos creado una función *imprime_cadena()* en la cual observamos si el buffer de transmisión con el sub-registro de UTXBF para comprobar si está lleno o podemos escribir aun q sea un byte extra. Esta transmisión será visible desde la terminal virtual.

```
void imprime_cadena(char cadena[]) {
    int i = 0;
    char* j = &cadena[0];
    while (U1STAbits.UTXBF == 0) {
        U1TXREG = (char) (*j);
        i++;
        j++;
    }
}
```

Hemos creado una variable “str” qué como máximo puede tener 100 elementos, no es óptimo, pero así llegará a un punto que deje de contar, después imprimimos la cadena con el nombre del alumno y el contador aumenta. Cómo se pide que vaya aumentando cada 0,5 segundos, hemos hecho uso de la función creada en la anterior práctica (Práctica 1: Control del GPIO), *delay_ms()* con un delay de 500ms.

```
char str[100];
sprintf(str, "%d", contador);
imprime_cadena(str);
contador++;
delay_ms(500);
```



Ejercicio 2

Se pide encender un Led de color rojo cuando se pulse la tecla E y que se apague cuando se pulse la letra A. Por defecto, el led deberá estar apagado.

Para ello, hacemos uso del registro de estado (U1STA) y su sub-registro FERR, para comprobar si ha habido algún error y del sub-registro OERR para solucionar dicho error.

```
if (U1STAbits.FERR == 1) {
    continue;
}
if (U1STAbits.OERR == 1) {
    U1STAbits.OERR = 0;
    continue;
}
```

También hemos hecho uso del sub-registro URXDA, que nos permitirá leer si hay al menos un dato en el buffer de recepción. Si esto se cumple hay una letra ‘e’ o ‘E’, el led rojo se encenderá. Pero si las letras son ‘a’ o ‘A’, entonces el led se apagará.

```
if (U1STAbits.URXDA) {
    ReceivedChar = (char) U1RXREG;
    if (ReceivedChar == 'e' || ReceivedChar == 'E') {
        LATBbits.LATB3 = 1;
    } else if (ReceivedChar == 'a' || ReceivedChar == 'A') {
        LATBbits.LATB3 = 0;
```

Ejercicio 3

Se pide hacer parpadear un led verde cada 250 ms cuando se pulse la tecla H. Cuando se vuelva a pulsar esa tecla, se deshabilitará esta función y quedará por lo tanto apagada.

Para ello hemos creado una variable bool “h_pulsada” que nos indicará si la acción de parpadear debe cumplirse. Reutilizamos el sub-registro URXDA anterior y metiendo una condición nueva, en la que, si se ha pulsado la tecla ‘h’ o ‘H’, h_pulsada cambiará su valor, al contrario. Si h_pulsada es true, entonces el LED verde parpadeará cada 250 ms y sino, permanecerá apagado.

```
        }else if(ReceivedChar == 'h' || ReceivedChar == 'H'){
            h_pulsada=!h_pulsada;
            if(h_pulsada==0){
                LED_GREEN=0;
            }
        }
    }
if(h_pulsada){parpadeo();delay_ms(250);}
```

Ejercicio 4

Por último, se pide que cuando se pulse la barra espaciadora el contador del ejercicio 1 se tendrá que reiniciar su cuenta a 0 y seguir funcionando.

Para ello, volvemos a utilizar la lectura del registro URXDA y añadimos la condición de que si el valor es un espacio en blanco ‘ ‘, lo que contiene el buffer, el contador tendrá el valor de 0.

```
} else if (ReceivedChar == ' ') {
    contador = 0;
```

Control de Tiempos

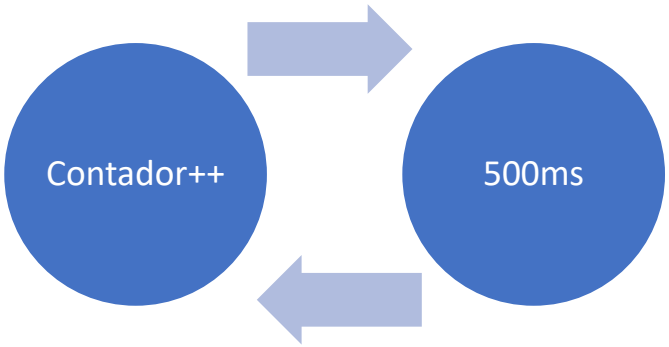
Para esta práctica se nos pide que cumplamos dos periodos.
Para ello, usaremos al función *delay_ms();*

```
void delay_ms(unsigned long time_ms)
{
    unsigned long u;
    for(u = 0; u < time_ms * 650; u++)
    {
        asm("NOP");
    }
}
```

Actualizamos el parámetro de calibración a 650, ya que nuestra frecuencia de CPU es de 40MHz.

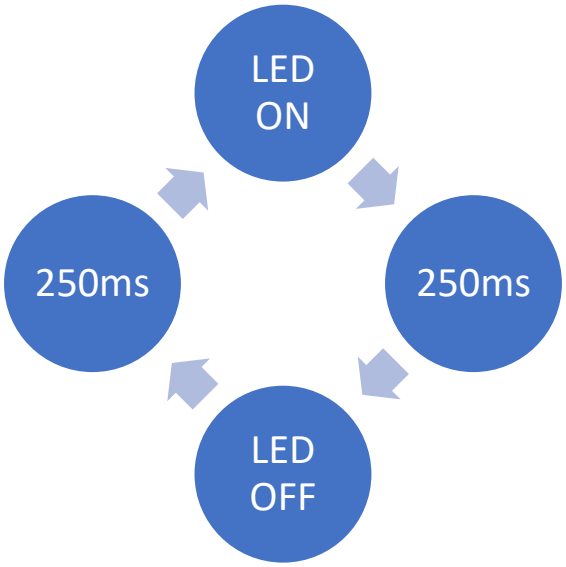
- Requisitos:
    - El contador debe actualizarse e imprimirse cada 500ms.

      Para ello simplemente debemos crear un delay de 500ms con la función *delay_ms();*

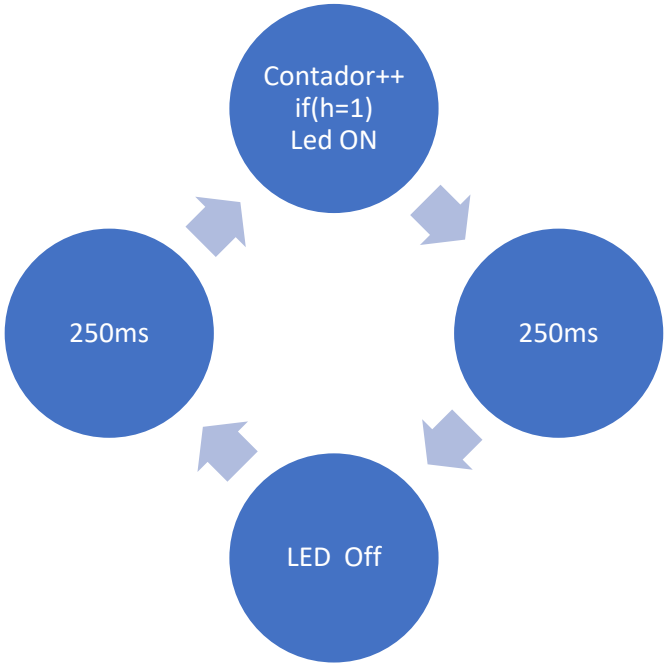


- El LED_GREEN debe parpadear cada 250ms.

  Para ello generaremos un delay de 250ms.



El problema que presenta es que cuando el contador se actualice y h esta pulsada, debemos gestionar los tiempos.



Código

```
if(h_pulsada){
 //he pulsado
    parpadeo();
    delay_ms(250);
}else{
 delay_ms(500);// genera aproximadamente un delay de 0.5 sengundos s
}
```

Como hemos comentado… si H esta pulsada… empezaremos el parpadeo y generamos el primer delay de 250ms, de lo contrario solo nos centramos en el contador y hacemos el delay de 500ms.

```
if(h_pulsada){parpadeo();delay_ms(250);}
```

Para Completar el ciclo, como hemos visto en el código del ejercicio 3, si h_pulsada está a 1, haremos el siguiente parpadeo y generaremos el ultimo delay de 250ms.

Código del bucle de ejecución

Para tener una visión completa del bucle infinito que se representa lo que se ejecuta en cada ciclo. (Se excluyen las funciones creadas fuera del while y algunas declaraciones de variables).

```
while(1)
{
    // Ejercicio 1
        char str[100];// no es muy optimo, llegará un punto que deje de contar por que no entra el numer
        sprintf(str, "%d", contador);
        imprime_cadena(str);
        contador++;
        if(h_pulsada){
         //he pulsado
            parpadeo();
            delay_ms(250);
        }else{
         delay_ms(500);// genera aproximadamente un delay de 0.5 sengundos s
        }

            /* mira si ha habido un error*/
            if(U1STAbits.FERR == 1)
            {
            continue;
            }
            /* soluciona el overrrun */
            if(U1STAbits.OERR == 1)
            {
            U1STAbits.OERR = 0;
            continue;
            }

            if(U1STAbits.URXDA)
            {
            ReceivedChar =(char)U1RXREG;
                if(ReceivedChar =='e' || ReceivedChar == 'E'){
                LATBbits.LATB3 = 1;
                }else if(ReceivedChar=='a' || ReceivedChar=='A'){
                    LATBbits.LATB3 = 0;
                }else if(ReceivedChar == ' '){
                    contador=0;
                }else if(ReceivedChar == 'h' || ReceivedChar == 'H'){
                    h_pulsada=!h_pulsada;
                    if(h_pulsada==0){
                        LED_GREEN=0;
                    }
                }
            }
        if(h_pulsada){parpadeo();delay_ms(250);};
}
return 0;
}
```