PRÁCTICAS

Sistemas Empotrados

ROCIO RUIZ E IGNACIO ORTEGA

mie. 14 oct. 2020

PRACTICA 1

En esta práctica hemos aprendido a configurar el módulo de comunicaciones serie UART. Para la realización de estos ejercicios primero hemos realizado la configuración de los bits, y de la UART tal y como se realizó en clase.

Para la UART hemos establecido nuevos pines:

- LEDs 1 y 2 en los pin RB7 y RA7, ambos configurados como pin de salida.

```
TRISBbits.TRISB7 = 0;
TRISAbits.TRISA7 = 0;

delay_ms(10);

LATBbits.LATB7 = 0;
LATAbits.LATA0 = 0;

delay_ms(10);

uart_config(baud_9600);
```

Imagen 1

- UART Rx y Tx, con los pines RC7 y RC8 respectivamente. Han sido configurados tal como muestra la imagen 2.

```
void uart_config(unsigned int baud){

    // CONFIGURACION de pines TX y RX
    TRISCbits.TRISC0  = 1;

    RPINR18bits.U1RXR = 23;

    RPOR12bits.RP24R  = 3;
```

Imagen 2

Ejercicio 1

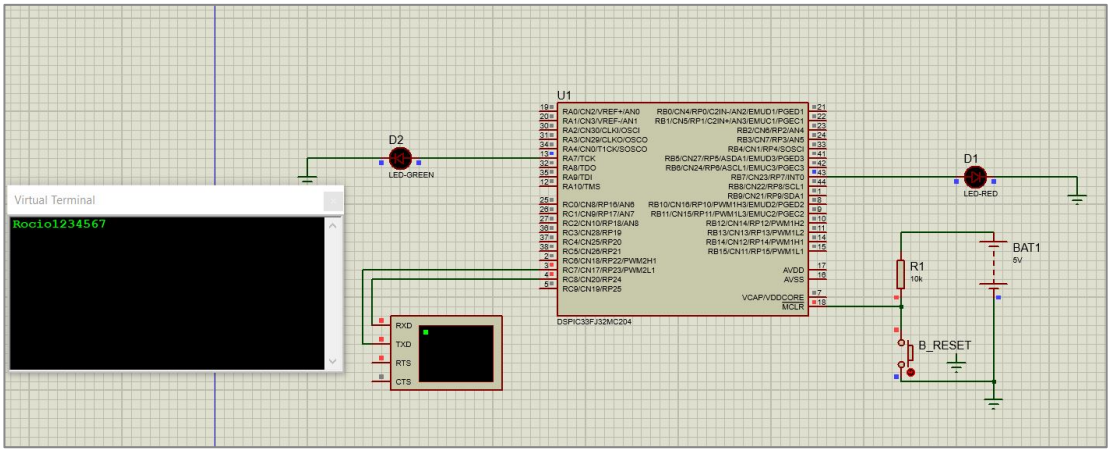
Se pide imprimir en la terminal virtual el nombre de un alumno y a continuación un contador que vaya aumentando cada 0,5 segundos.

Para ello, creamos un contador y se lo asignamos a el buffer de transmisión U1TXREG. Esta transmisión será visible en la terminal virtual. Como tiene que ser transmitido es ascii para ser visible, hemos implementado contador como un carácter. Al ser cada 0,5 segundos, haríamos uso de la función creada en clase delay_ms() con parámetro de 500 ms. Pero, en el tercer ejercicio se nos pedirá implementar una acción cada 0,25 segundos. Por ello, lo hemos implementado como muestra la imagen 3 y hemos puesto un delay_ms() de 200 ms.

```
int contador = '0';
char nombre[] = "Rocio";
int j = 0;
while(j < sizeof(nombre)){
    U1TXREG = nombre[j];
    j++;
}
while(1){
    //PRIMER EJERCICIO. (LO HEMOS INTENTADO CON SPRINTF PERO NO IMPRIME NADA
    if(ms500 == true){
        U1TXREG = contador;
       (int)contador++;
    }
    ms500 = !ms500;
```

Imagen 3

"ms500" es una variable tipo booleano que cada ciclo cambia de valor a su opuesto, de esa manera, el contador solo se imprimirá cada dos ciclos de 250 ms, es decir 500ms.



Ejercicio 2

Se pide encender la LED 1 roja cuando se pulse la tecla E del teclado y apagarla cuando se pulse la letra A. Por defecto, debe estar apagada.

Para ello, hacemos uso de registro de estado (U1STA) y su sub-registro URXDA, que nos permitirá leer si hay al menos un dato en el buffer de recepción. Si esto es así, y hay una letra 'e' o 'E' en ASCII (69 o 101), el LED rojo se encenderá. Si es así, pero la letra es 'a' o 'A' (65 o 97 en ASCII) se apagará.

```
//SEGUNDO EJERCICIO
if(U1STAbits.URXDA == 1){
    if(U1RXREG == 69 || U1RXREG == 101 ){ // TECLA 'E' o 'e'
        led_D1red = 1;          // ENCENDER D2
    }
    else if(U1RXREG == 65 || U1RXREG == 97){ //TECLA 'A' O 'a'
        led_D1red = 0;          // APAGAR D2
    }
```

Imagen 4

Ejercicio 3

Se pide hacer parpadear el LED 2 verde cada 250 ms cuando se pulse la tecla H. Cuando se vuelva a pulsar esta tecla, se deshabilitará esta función y quedará apagada.

Para ello, vamos a necesitar una variable bool que nos indique si el LED está encendido o apagado ("Hpressed"). Nos aprovechamos del uso del sub-registro URXDA anterior, y metemos otra condición nueva, en la que si se ha pulsado la tecla 'h' o 'H' (72 y 104 en ASCII) Hpressed cambiará a su valor contrario. Si Hpressed es true, el LED parpaderá cada 250 ms, y sino, permanecerá apagado.

```
        //TERCER EJERCICIO
        if(U1RXREG == 72 || U1RXREG == 104){ // TECLA 'H' o 'h'
            Hpressed = !Hpressed;
        }
    }
    if(Hpressed == true){
            led_D2green = !PORTAbits.RA7;
        }else if(Hpressed == false){
            led_D2green = 0;
        }
    delay_ms(250);
}
```

Imagen 5

Ejercicio 4

Se pide que cuando se pulse la barra espaciadora el contador del ejercicio 1 debe reiniciar su cuenta a 0 y seguir funcionando.

Para ello, nos volvemos a aprovechar de la lectura al registro URXDA y añadimos la condición de que si es el valor asci de 32 lo que contiene el buffer, el contandor tendrá el valor de '0'.

```
//CUARTO EJERCICIO
if(U1RXREG == 32){
    contador = '0';
}
```

Imagen 6

Código del bucle de ejecución

Para tener una visión completa, la imagen 6 muestra el bucle infinito que representa lo que se ejecuta cada ciclo. (Excluye la impresión del nombre delalumno porque eso solo se imprime en el primer ciclo de programa/ ocasión de reset).

```
while(1){
    //PRIMER EJERCICIO. (LO HEMOS INTENTADO CON SPRINTF PERO NO IMPRIME NADA)
    if(ms500 == true){
        U1TXREG = contador;
       (int)contador++;
    }
    ms500 = !ms500;

    //SEGUNDO EJERCICIO
    if(U1STAbits.URXDA == 1){
        if(U1RXREG == 69 || U1RXREG == 101 ){ // TECLA 'E' o 'e'
            led_D1red = 1;          // ENCENDER D2
        }
        else if(U1RXREG == 65 || U1RXREG == 97){ //TECLA 'A' O 'a'
            led_D1red = 0;          // APAGAR D2
        }

        //CUARTO EJERCICIO
        if(U1RXREG == 32){
            contador = '0';
        }
        //TERCER EJERCICIO
        if(U1RXREG == 72 || U1RXREG == 104){ // TECLA 'H' o 'h'
            Hpressed = !Hpressed;
        }
    }
    if(Hpressed == true){
            led_D2green = !PORTAbits.RA7;
        }else if(Hpressed == false){
            led_D2green = 0;
        }
    delay_ms(250);
}
```