UNIVERSIDAD NEBRIJA

MEMORIA PRACTICA 2
SISTEMAS EMPOTRADOS

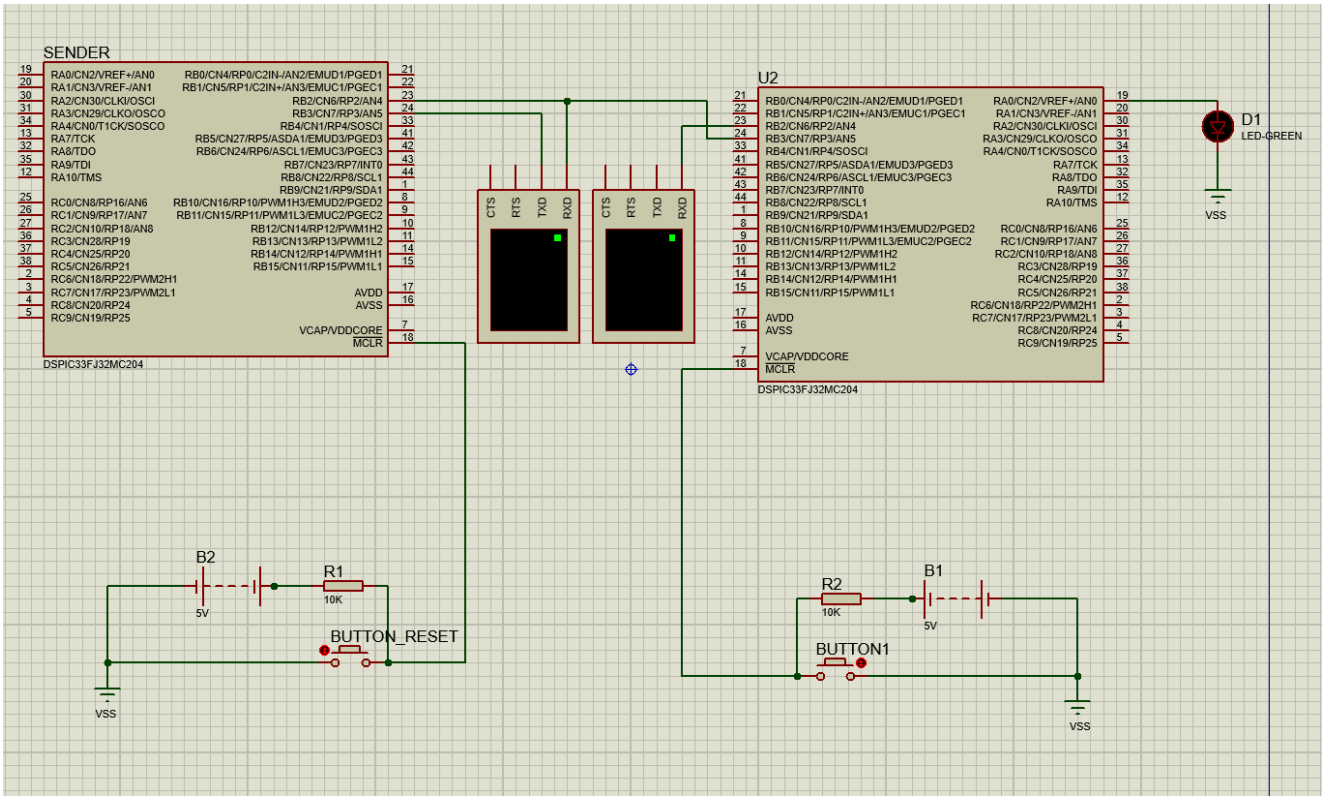
José María Fernández Gómez
Pablo Rayón Zapater
Fernando Pérez Ballesteros

OBJETIVOS

En esta práctica se pretende estudiar comunicación entre dos microcontroladores por medio del protocolo de comunicaciones serial UART, se pretende enviar un comando a través de un terminal virtual a la unidad master o Sender (izquierda) la cual reenviará este comando a la unidad slave o receptora (Derecha).

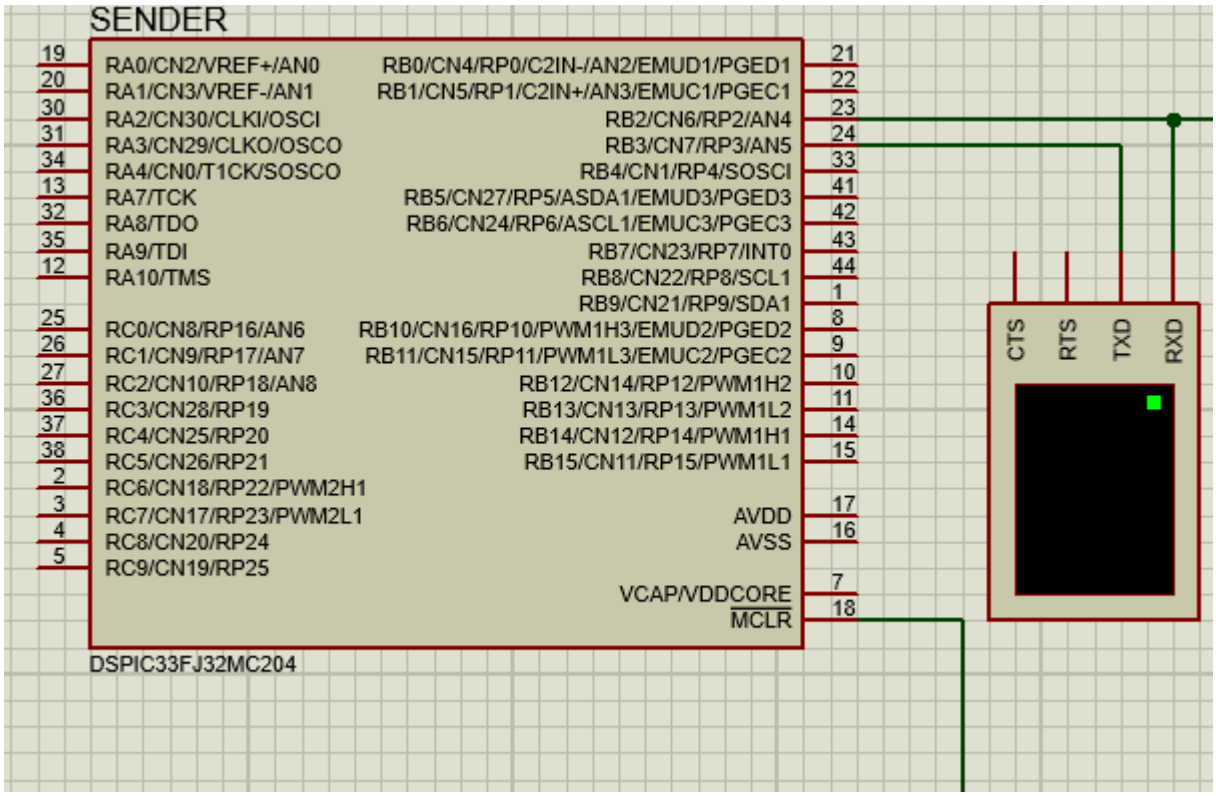
Los pines que se han empleado para este protocolo son iguales tanto para el master como para el slave, siendo estos el pin RB2 para el envío de datos y el RB3 para la recepción.

A continuación se muestra el diseño del esquemático trabajado.

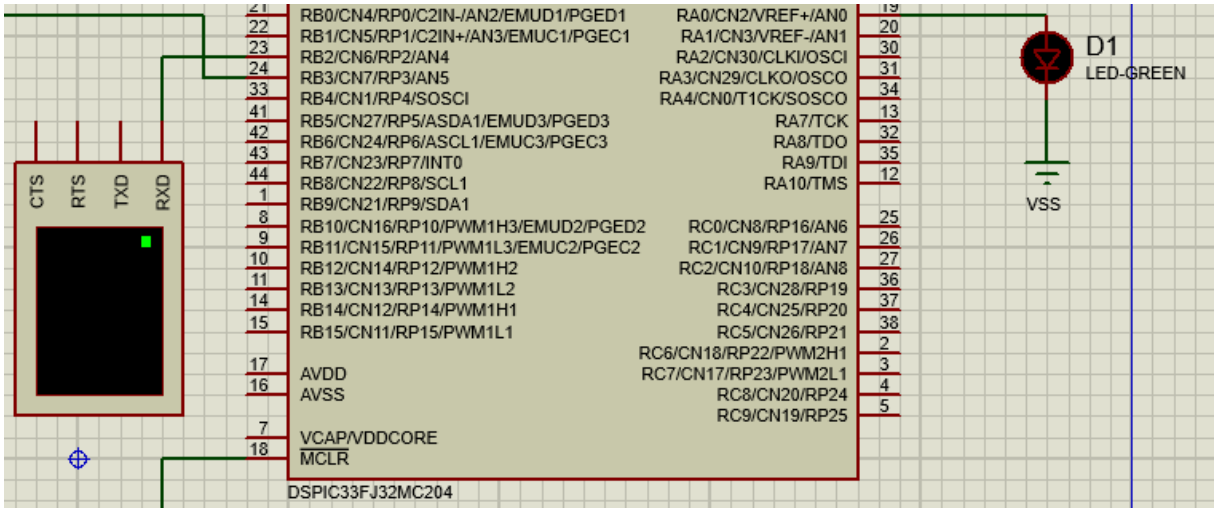


CONFIGURACIÓN

A continuación, se muestra la configuración a nivel de hardware para cada unidad.



En esta primera imagen se muestra el Sender, que tiene conectado a su pin de recepción de la UART un terminal virtual por el que el usuario escribirá los comandos, estos serán re-enviados por el pin de transmisión de esta misma unidad hacia el microcontrolador sender, este pin también tiene conectado una terminal virtual para poder ver el output que se dirige al segundo controlador.



En este segundo esquemático observamos la unidad receptora, que por su pin de recepción le llegan los comandos que han sido previamente enviados por la unidad Master. Por su pin de envío vemos un terminal virtual, este situado ahí para poder imprimir el contador que también será controlado por comandos. Po último tenemos un diodo LED en el pin A0, este diodo es encendido y apagado por comando recibidos a través de UART.

CODIGO

```
void uart_send_byte(char c) {
    while (U1STAbits.UTXBF); //mientras el buffer no este lleno
    U1TXREG = c;
}

void uart_send_text(char *s) {
    while (*s != '\0') uart_send_byte(*s++);
}
```

Estas son las funciones empleadas para el envío de caracteres y cadenas de texto a través del protocolo UART. Son iguales para los dos microcontroladores.

```
int main(void) {
    // Fcpu = Fosc/2 = 20mhz
    // Fosc = 40mhz
    // Fin  = 4mhz
    // Fosc = Fin *(M/N1*N2)
    PLLFBD = 38; //M  = PLLFBD + 2
    CLKDIVbits.PLLPOST = 0; //N1 = PLLPOST + 2
    CLKDIVbits.PLLPRE = 0; //N1 = PLLPOST + 2
    while (OSCCONbits.LOCK != 1);

    AD1PCFGL = 0xFFFF; // Todos los pines configurados como pines digitales
    uart_config(baud_4800);

    while (1) {

        if (U1STAbits.URXDA) {
            uart_send_byte(U1RXREG);
        }
        delay_ms(500);
    }

    return 0;
}
```

En la imagen de la izquierda observamos el código del controlador master o sender, el cual se encarga de reenviar por su puerto UART de transmisión el comando recibido a través de su puerto de recepción. Este puerto de transmisión esta dirigido al puerto de recepción del controlador slave.

```c
int main(void) {
    // Fcpu = Fosc/2 = 20mhz
    // Fosc = 40mhz
    // Fin  = 4mhz
    // Fosc = Fin *(M/N1*N2)
    PLLFBD = 38; //M  = PLLFBD + 2
    CLKDIVbits.PLLPOST = 0; //N1 = PLLPOST + 2
    CLKDIVbits.PLLPRE = 0; //N1 = PLLPOST + 2
    while (OSCCONbits.LOCK != 1);

    AD1PCFGL = 0xFFFF; // Todos los pines configurados como pines digitales
    TRISAbits.TRISA0 = 0;
    LATAbits.LATA0 = 0;

    uart_config(baud_4800);
    int countBool = 1;
    while (1) {
        if (U1STAbits.URXDA) {
            var_recv = U1RXREG; //este dato va a salir por RB2
            for (int i = 0; i < 10; i++) {
                if (txbuffer[i] == NULL) {
                    if (var_recv == 8) {
                        txbuffer[i - 1] = NULL;
                    } else {
                        txbuffer[i] = var_recv;
                        break;
                    }
                }
```

En esta imagen tenemos la primera parte del código del receptor, se ha desarrollado una función que se pone a la escucha de caracteres a través del puerto de recepción UART siempre que el buffer de recepción no esté vacío. Esta función itera a través de un buffer creado que se encargará de almacenar los comandos deseados, sanitizando la entrada por si se recibe algún carácter no deseado (8), para luego almacenar en este buffer carácter a carácter el comando recibido.

En esta última porción de código se realiza la lógica detrás del reconocimiento de comandos y la actuación en consecuencia del microcontrolador. En primer lugar, se determina cuando el usuario ha enviado un comando, esto se hace a través de la detección del carácter \r, que equivale a pulsar la tecla enter, concluyendo así la escritura en una línea y por lo tanto la escritura de ese comando. Después se realiza la comparación del buffer que contiene el comando con el comando predefinido correspondiente, cada uno ejecutando las acciones que le corresponde en caso de ser reconocido.

```c
                if (var_recv == '\r') {
                    txbuffer[strlen(txbuffer) - 1] = NULL;
                    if (strcmp(txbuffer, "lon") == 0) {
                        LATAbits.LATA0 = 1;
                        uart_send_text("lon");
                    }
                    if (strcmp(txbuffer, "loff") == 0) {
                        LATAbits.LATA0 = 0;
                        uart_send_text("loff");
                    }
                    if (strcmp(txbuffer, "play") == 0) {
                        countBool = 1;
                        uart_send_text("play");
                    }
                    if (strcmp(txbuffer, "pause") == 0) {
                        countBool = 0;
                        uart_send_text("pause");
                    }
                    for (int i = 0; i < 10; i++) {
                        txbuffer[i] = NULL;
                    }
                }
            }
            if (countBool!=0) {
                sprintf(countbuffer, "contador: %d\r\n", contador++);
                uart_send_text(countbuffer);
            }
            delay_ms(1000);
        }
```