

```

/*****
* File: serie_rx.c
* Author: Malakabot
* Description: Recibe palabras de 8 bits a través del puerto serie (PIN_C7)
* Para probarlo se ha usado un módulo bluetooth HC-05 conectado a PIN_C6 y PIN_C7
* Se recomienda leer el manual pdf acerca de la configuración de los baudios
* EL PLL no se ha configurado, ya que a 48MHz no sincroniza bien. El PIC trabaja a 20MHz
* Hardware: PIC18F2550, Cristal 20Mhz, MKBot v.3
* Created: 1 septiembre 2017
*****/

#include <xc.h>
#pragma config FOSC = HS      // Oscillator Selection bits (HS oscillator (HS))
#pragma config LVP = OFF      // Single-Supply ICSP Enable bit (Single-Supply ICSP disabled)
#pragma config WDT = OFF      // Watchdog Timer Enable bit (WDT disabled (control is placed on the SWDTEN bit))
#define _XTAL_FREQ 2000000    // 20MHz

/* Valores para SPBRG, a 20MHz, alta velocidad (BRGH), 8 bits de resolución (BRG16) */
/* SPBRG = (int) (_XTAL_FREQ/16*baud_rate)-1 */
#define BAUD_9600    129    // 9615bps, error=0.16%
#define BAUD_19200   64     // 19231bps, error=0.16%
#define BAUD_57600   21     // 56818bps, error=1.36%
#define BAUD_115200  10     // 113636bps, error=1.36%

/* Configura el puerto serie para recibir caracteres, sin bit de paridad, baud_rate=9600bps */
void USART_conf()
{
    TRISCbits.RC6=0;        // Configura PIN_C6 (Tx), de salida
    TRISCbits.RC7=1;        // Configura PIN_C7 (Rx), de entrada
    TXSTAbits.SYNC=0;      // USART modo asíncrono
    RCSTAbits.SPEN=1;      // Habilita puerto serie
    RCSTAbits.CREN=1;      // Habilita la recepción continua
    TXSTAbits.BRGH=1;      // Baudios: Alta velocidad
    BAUDCONbits.BRG16=0;   // Baudios: SPRG usa 8 bits, baja resolución
    SPBRG=BAUD_9600;      // Baudios: 9615bps, error=0.16%
}

/* Devuelve el carácter recibido */
char USART_getch()
{
    while(PIR1bits.RCIF==0) // Espera a que salte el flag de recepción de datos
        continue;
    return(RCREG);          // Devuelve el dato de 8 bits recibido, de tipo char
}

```

```
/* Si recibe 'F': enciende LED. Si recibe 'B': apaga LED. Resto de caracteres: indiferente */
void main()
{
    char data_in;
    TRISB=0x00;           // PUERTO B de salida
    LATB=0x00;
    USART_conf();
    while(1)
    {
        data_in=USART_getch();
        switch (data_in)
        {
            case 'F':
                LATB=0xFF;
                break;
            case 'B':
                LATB=0x00;
                break;
            default:
                break;
        }
    }
}
```