

Microcontroladores

Sesión de Laboratorio

Semana 6

2021-0

Profesor: Kalun Lau

1

Preguntas previas:

2

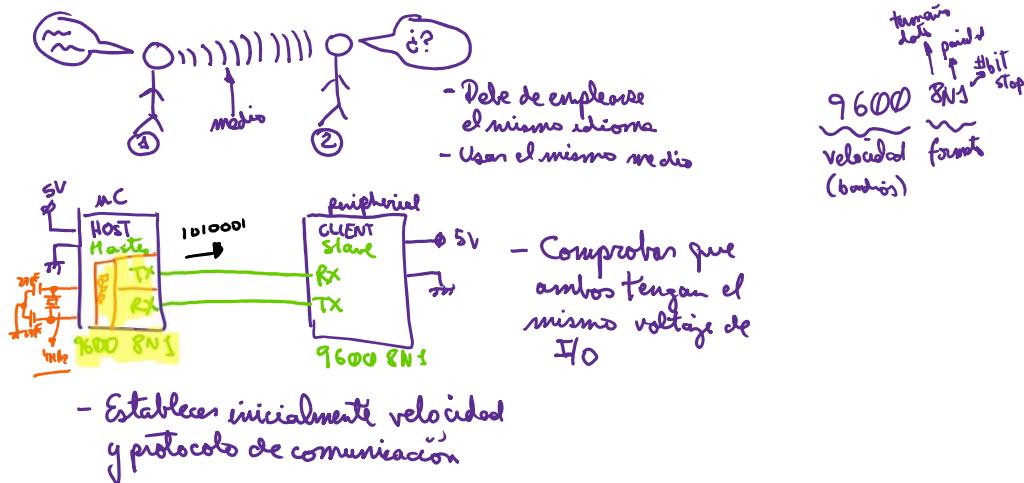
Agenda:

- Comunicación serial asíncrona UART
- El módulo EUSART del microcontrolador PIC18F4550
 - Módulo generador de baudios (BRG ó SPBRG)
 - Módulo transmisor (TX)
 - Módulo receptor (RX)

3

Comunicación asíncrona UART

- Asíncrona: No hay una señal dedicada de reloj

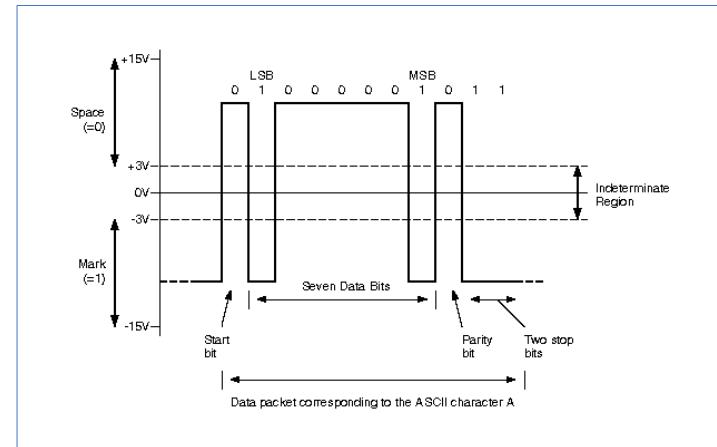
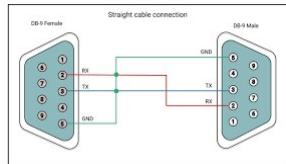
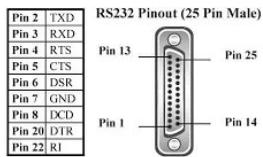


4

Sobre niveles de voltaje en comunicación UART

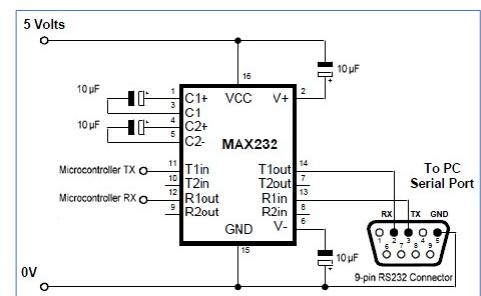
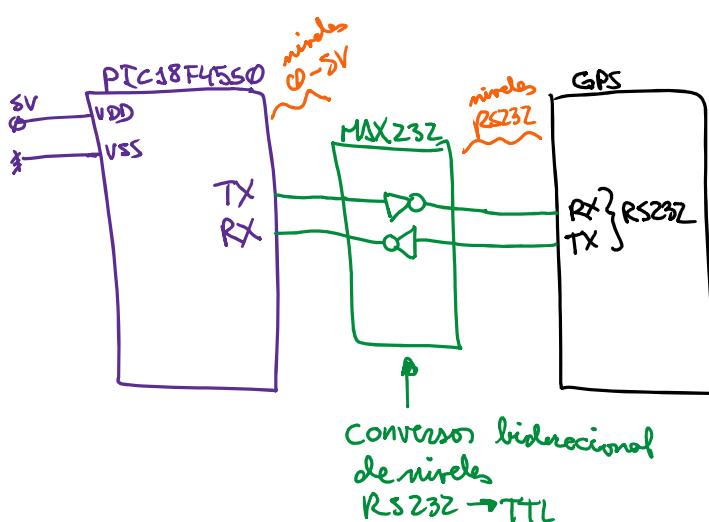
- RS232 (EIA/TIA 232) – Comunicación a distancias medianas (hasta 15 metros)

RS232 25 Pin



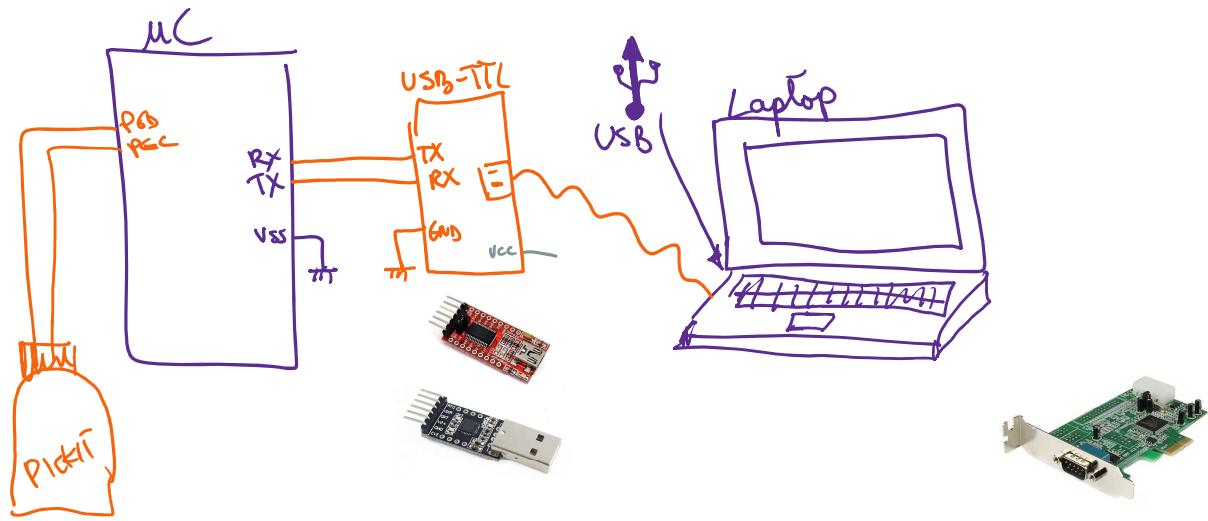
5

¿Cómo hago para conectar un dispositivo RS232 al PIC18F4550?



6

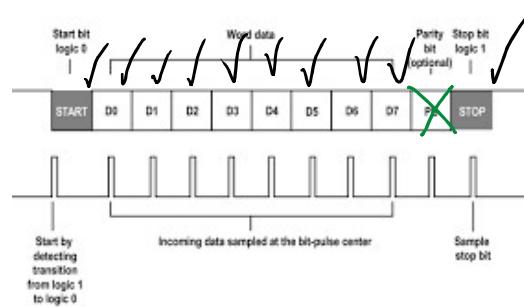
Cómo conectar un microcontrolador PIC hacia un computador mediante USB



7

Cálculo de la velocidad en comunicación UART

- Formato completo es
 - Velocidad – Protocolo
 - Ej. 9600 - 8N1
- Protocolo común: 8N1
 - 8: El dato es de 8 bits
 - N: no paridad (O:paridad impar, I:paridad par)
 - 1: un bit de stop



- En total se están enviando 10 bits por cada dato de 8 bits

8

Cálculo de la velocidad en comunicación UART

- Si tengo $\overset{V_{TX}}{9600}$ 8N1:

$$\text{Tiempo de bit: } T_{\text{bit}} = \frac{1}{V_{TX}} = \frac{1}{9600} = 1.04 \times 10^{-4} \text{ s} \\ = 0.1 \text{ ms}$$

- Si para enviar un dato de 8 bits usamos 10 bits:

$$\text{Tiempo para enviar un dato de 8bit} = 1.04 \times 10^{-4} \text{ s} = 1.04 \text{ ms.}$$

9

Ejemplo de cálculo de cuánto tiempo demora en enviar datos vía comm. serial:

- ¿Cuánto demoro en enviar un archivo de 50KByte por un canal de comunicación serial a 4800 8N1?

$$\begin{aligned} & 50 \text{ Kbyte} \\ & 1 \text{ byte} = 8 \text{ bits} \\ & 1 \text{ Kbyte} = 1024 \text{ bytes} \\ & \frac{50 \times 1024}{512000 \text{ bytes}} \times (8+2) \text{ bits} \\ & T_{\text{bit}} = \frac{1}{4800} = \\ & T_{\text{bit}} = 0.208 \text{ ms} \\ & T_{\text{Total}} = 106.67 \text{ segundos} \end{aligned}$$

10

Ejemplo de cálculo de cuánto tiempo demora en enviar datos vía comm. serial:

- ¿Cuánto demoro en enviar un archivo de música de 6MByte por una canal de comm. serial a 57600 8N1?

Aprox 20 minutos

11

El modulo EUSART

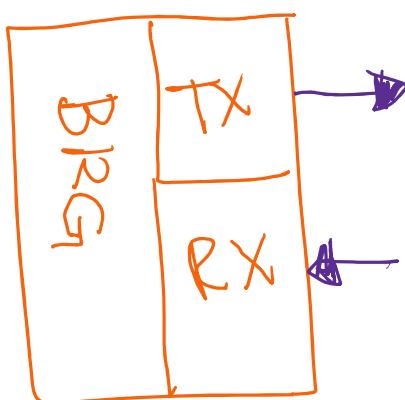


FIGURE 20-3: EUSART TRANSMIT BLOCK DIAGRAM

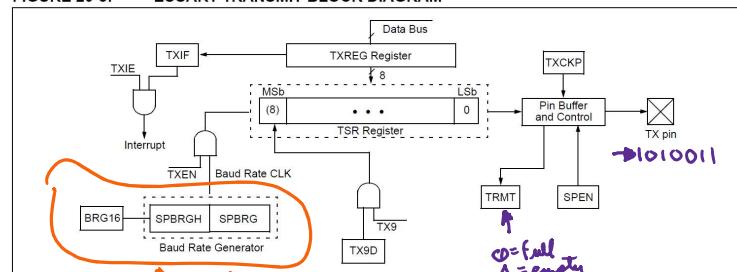
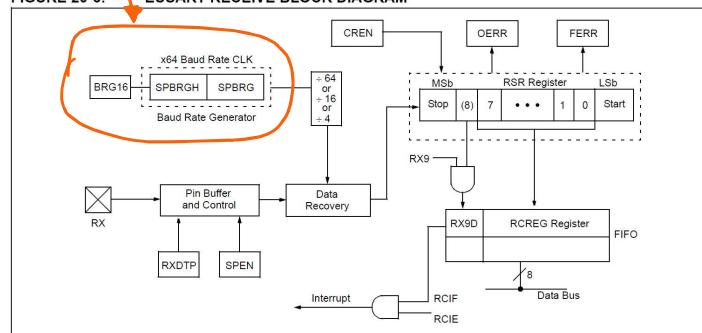


FIGURE 20-6: EUSART RECEIVE BLOCK DIAGRAM



12

Para transmitir un dato:

1. Initialize the SPBRGH:SPBREG registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate. ✓
 2. Enable the asynchronous serial port by clearing bit, SYNC, and setting bit, SPEN. ✓
 3. If the signal from the TX pin is to be inverted, set the TXCKP bit. ✗
 4. If interrupts are desired, set enable bit, TXIE. ✗
 5. If 9-bit transmission is desired, set transmit bit, TX9. Can be used as address/data bit. ✗
 6. Enable the transmission by setting bit, TXEN, which will also set bit, TXIF. ✓
 7. If 9-bit transmission is selected, the ninth bit should be loaded in bit, TX9D. ✗
 8. Load data to the TXREG register (starts transmission). ✓
 9. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set ✗

10. Esperar a que TRMT cambie de estado

| Configuration Bits | | | BRG/EUSART Mode | Baud Rate Formula |
|--------------------|-------|------|--------------------|--------------------|
| SYNC | BRG16 | BRGH | | |
| 0 | 0 | 0 | 8-bit/Asynchronous | $Fosc/[64(n + 1)]$ |

$$\text{Si } V_{TX} = 96\phi_0 \Rightarrow B_R G = ? \quad \text{despu\'es de 'n'}$$

$$m = \frac{(\text{FOSC} / \text{Bitrate})}{64} - 1 \quad n = \underbrace{\text{SPBRGH:SPBRG}}_{\text{multi-F: 16-bit}}$$

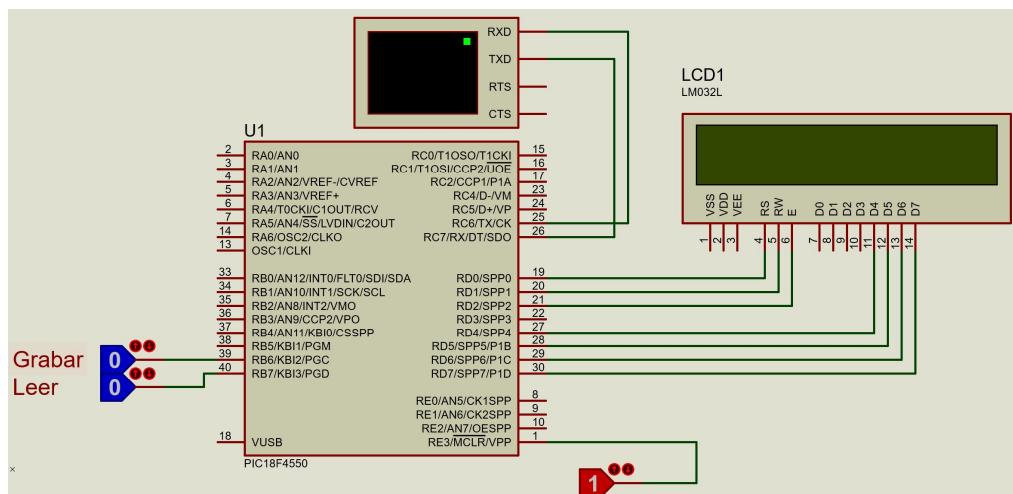
$$m = \frac{\frac{48000000}{9600}}{64} - 1 \quad \text{pero como bit } BRG16 = 0 \\ \Leftrightarrow PRBGH = 0$$

$$m = 77.125 \rightarrow SPBRG = 77$$

$$\text{Band rate actual} = \frac{FOSC}{[64(m+1)]} = \frac{48000000}{[64(77+1)]} = 9615$$

$$\text{Bit rate error: } \frac{9615 - 9600}{9600} \times 100\% = 0.16\%$$

Círculo de prueba



Código ejemplo:

```

12 #include <xc.h>
13 #define _XTAL_FREQ 48000000UL //frecuencia
14
15 unsigned char mensaje1[] = {"Boton presionado"};
16 unsigned char mensaje2[] = {"Boton soltado "};
17 unsigned char indicador = 0;
18
19 void init_conf(void){
20     TRISChbits.RC6 = 0; //Salida para
21 }
22
23 void EUSART_conf(){
24     SPBRG = 77; //Vtx = 9600
25     RCSTAbits.SPEN = 1; //Habilitamos el
26     TXSTAbits.TXEN = 1; //Habilitamos la
27 }

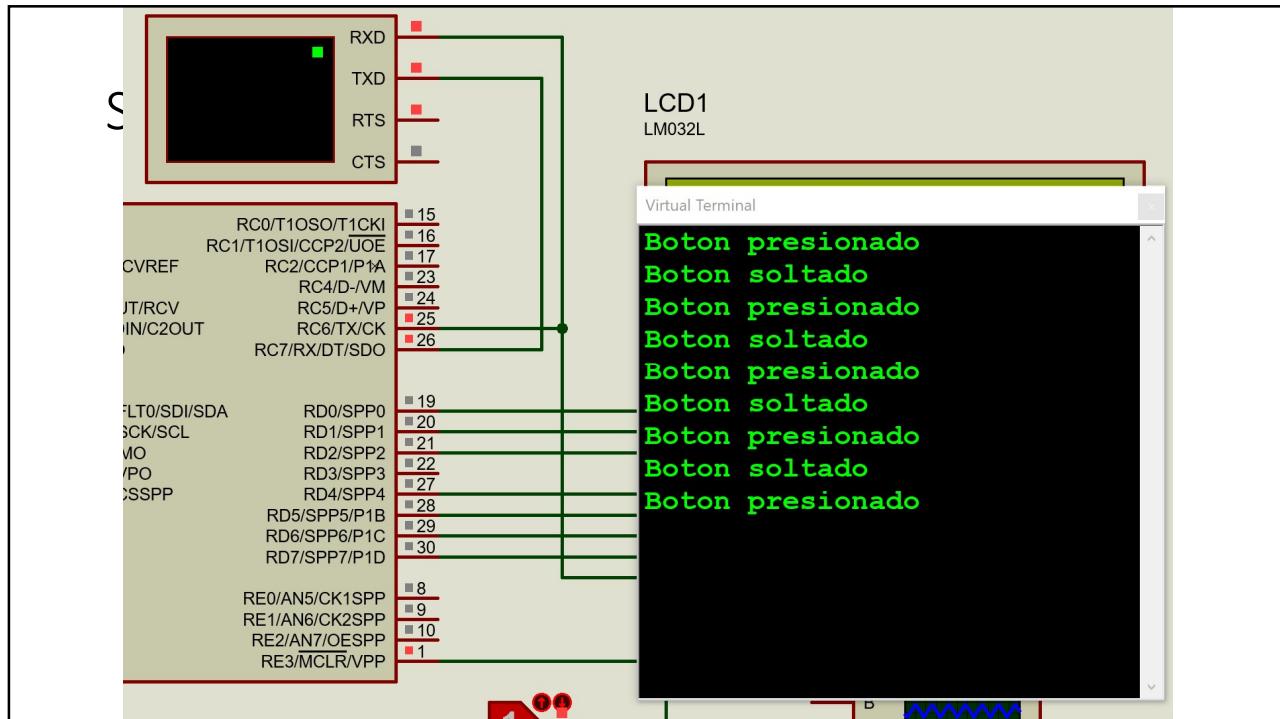
```

```

29 void main(void){
30     init_conf();
31     EUSART_conf();
32     while(1){
33         if(PORTBbits.RB6 == 1 && indicador == 0){
34             for(unsigned char x=0;x<16;x++){
35                 TXREG = mensaje1[x];
36                 while(TXSTAbits.TRMT == 0); //Esperar a
37             }
38             TXREG = 0x0A; //Comando para nueva linea
39             while(TXSTAbits.TRMT == 0); //Esperar a que
40             TXREG = 0x0D; //Comando para retorno de
41             while(TXSTAbits.TRMT == 0); //Esperar a que
42             indicador = 1;
43         }
44         else if(PORTBbits.RB6 == 0 && indicador == 1){
45             for(unsigned char x=0;x<16;x++){
46                 TXREG = mensaje2[x];
47                 while(TXSTAbits.TRMT == 0); //Esperar a
48             }
49             TXREG = 0x0A; //Comando para nueva linea
50             while(TXSTAbits.TRMT == 0); //Esperar a que
51             TXREG = 0x0D; //Comando para retorno de
52             while(TXSTAbits.TRMT == 0); //Esperar a que
53             indicador = 0;
54         }
55     }
56 }

```

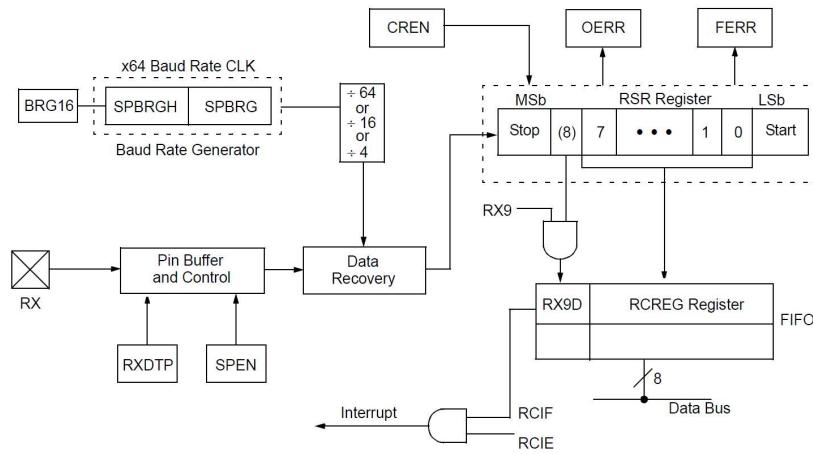
15



16

Receptor del EUSART

- Diagrama de bloques



17

Receptor del EUSART

- Emplea el mismo SPBRG que el transmisor.
- Seguir el procedimiento de configuración de la hoja técnica
- Se recomienda el uso de interrupciones en esta etapa de recepción ya que el receptor carece de FIFO.

18

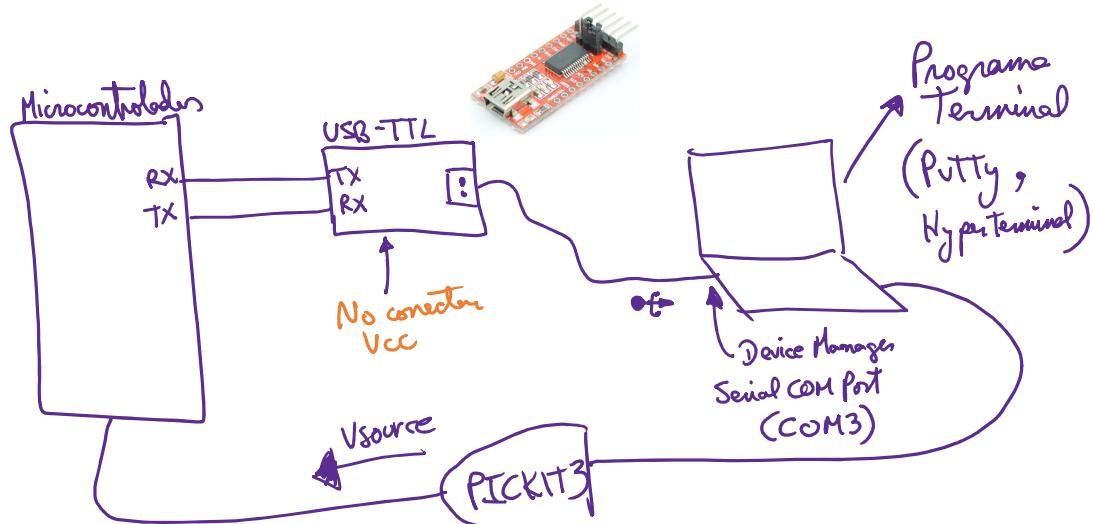
Receptor del EUSART

- Procedimiento para recibir un dato (según hoja técnica)

- To set up an Asynchronous Reception:
1. Initialize the SPBRGH:SPBRG registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
 2. Enable the asynchronous serial port by clearing bit, SYNC, and setting bit, SPEN.
 3. If the signal at the RX pin is to be inverted, set the RXDTP bit.
 4. If interrupts are desired, set enable bit, RCIE.
 5. If 9-bit reception is desired, set bit, RX9.
 6. Enable the reception by setting bit, CREN.
 7. Flag bit, RCIF, will be set when reception is complete and an interrupt will be generated if enable bit, RCIE, was set.
 8. Read the RCSTA register to get the 9th bit (if enabled) and determine if any error occurred during reception.
 9. Read the 8-bit received data by reading the RCREG register.
 10. If any error occurred, clear the error by clearing enable bit, CREN.
 11. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

19

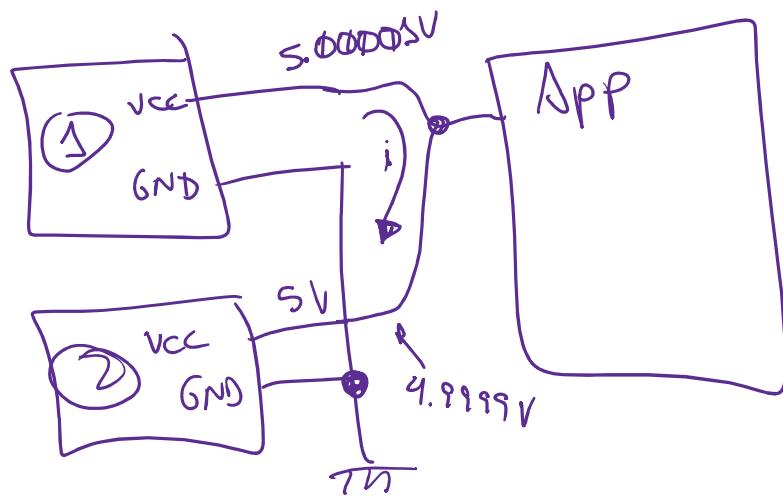
Ejemplo de comunicación UART entre el microcontrolador y un terminal serial



20

Nota: Tener cuidado en no utilizar fuente de alimentación en paralelo.

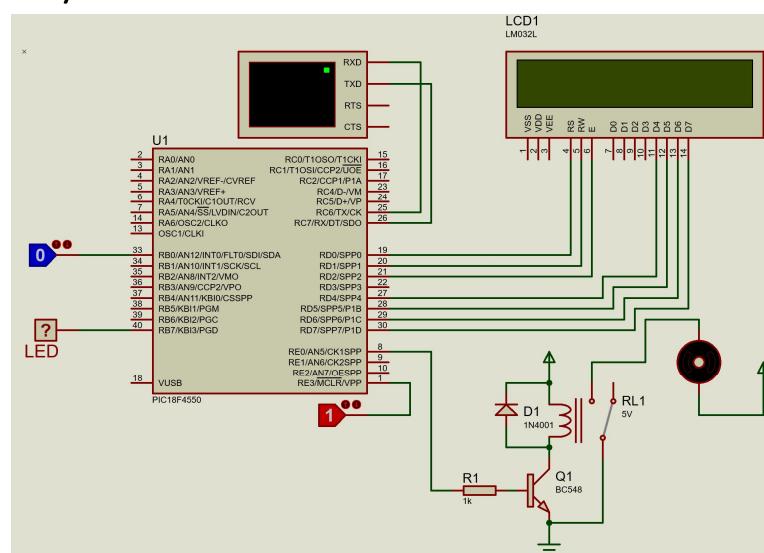
- El pickit3 suministra la alimentación al circuito de prueba, el conversor USB-TTL también tiene una línea de alimentación que viene directo del puerto USB por lo que solo deben de usar una fuente, o la del pickit3 o la del USB-TTL



21

Ejemplo de comunicación UART entre el microcontrolador y un terminal serial

- El PIC18F4550 enviará el menú de opciones vía EUSART hacia el terminal virtual a una tasa de 9600 8N1, se tendrá las opciones de encender el LED conectado en RB7



22

Ejemplo de comunicación UART entre el microcontrolador y un terminal serial

```

1 #pragma config PLLDIV = 1           // PLL Prescaler Selection
2 #pragma config CPUDIV = OSC1_PLL2 // System Clock Selection
3 #pragma config FOSC = XTPLL_XT // Oscillator Selection
4 #pragma config FWRT = ON          // Power-up Timer Enable
5 #pragma config BOR = OFF           // Brown-out Reset
6 #pragma config WDT = OFF           // Watchdog Timer Enable
7 #pragma config CCP2MX = ON         // CCP2 MUX bit (CCP2M0)
8 #pragma config PBADEN = OFF        // PORTB A/D Enable
9 #pragma config MCIRE = ON          // MCLR Pin Enable
10 #pragma config LVP = OFF           // Single-Supply ICSP
11
12 #include <xc.h>
13 #define _XTAL_FREQ 48000000UL      //frecuencia de cristal
14
15 unsigned char menu1[] = {"Bienvenidos al ejemplo"};
16 unsigned char menu2[] = {"Selecione una opcion"};
17 unsigned char menu3[] = {"(1) Enciende LED"};
18 unsigned char menu4[] = {"(2) Apaga LED"};
19 unsigned char menu5[] = {"(m) Muestra el menu"};
20 unsigned char ledon[] = {"LED encendido"};
21 unsigned char ledoff[] = {"LED apagado"};
22
23 unsigned char indicador = 0;
24
25 void PORT_config(void){
26     TRISBbits.RB7 = 0;           //Salida en RB7
27 }
28
29 void EUSART_config(void){
30     SPBRGH = 0;                //Ignorado debido a la velocidad
31     SPBRG = 7;                 //Vtx es 9600 baudios
32     TRISEcbits.RC6 = 0;          //Puerto RC6 como salida
33     RCREcbits.SREN = 1;          //Encendemos el puerto
34     TXSTAbits.TXEN = 1;          //Encendemos el transmisor
35     RCSTAcbits.CREN = 1;          //Encendemos el receptor
36 }
37
38 void INT_config(void){
39     INTCONbits.GIE = 1;          //Interruptor global habilitado
40     INTCONbits.PIE1 = 1;          //Interruptor de perifericos habilitado
41     PIE1bits.RCIE = 1;           //Habilitado interrupcion por recepcion
42 }
43
44 void EUSART_siguientelinea(void){
45     TXREG = 0xA;
46     while(TXSTAbits.TRMT == 0);
47     TXREG = 0xD;
48     while(TXSTAbits.TRMT == 0);
49 }
50
51 void EUSART_enviacadena(const unsigned char *vector,unsigned char pos){
52     for (unsigned char x=0;x<pos;x++){
53         TXREG = vector[x];
54         while(TXSTAbits.TRMT == 0);
55     }
56 }
57
58 void show_menu(void){
59     EUSART_enviacadena(menu1,22);
60     EUSART_siguientelinea();
61     EUSART_enviacadena(menu2,22);
62     EUSART_siguientelinea();
63     EUSART_enviacadena(menu3,22);
64     EUSART_siguientelinea();
65     EUSART_enviacadena(menu4,22);
66     EUSART_siguientelinea();
67     EUSART_enviacadena(menu5,22);
68     EUSART_siguientelinea();
69 }
70
71 void main(void) {
72     PORT_config();
73     EUSART_config();
74     INT_config();
75     show_menu();
76     while(1);
77 }
78
79
80 void _interrupt(high priority) RC_Ier(void){
81     PIR1bits.RC1F = 0;
82     if(RCREG == '1'){
83         LATBbits.LB7 = 1;
84         EUSART_enviacadena(ledon,22);
85         EUSART_siguientelinea();
86     }
87     else if(RCREG == '0'){
88         LATBbits.LB7 = 0;
89         EUSART_enviacadena(ledoff,22);
90         EUSART_siguientelinea();
91     }
92     else if(RCREG == 'D'){
93         show_menu();
94     }
95 }
96

```

23

Ejemplo de comunicación UART entre el microcontrolador y un terminal serial

- Agregando una opción para que se ingrese una cadena de caracteres desde el teclado y devuelva por el terminal virtual dicha cadena.
- Cambiando la opción de RB7 por la de RE0 donde esta conectado un relay y un motor DC.

24

Ejemplo de comunicación UART entre el microcontrolador y un terminal serial

```

1 #pragma config PLLDIV = 1           // FLL Prescaler Selection bits (0x
2 #pragma config FUDIV = OSC1_PLL2 // System Clock Postscaler Select
3 #pragma config FOSC = XTPLL_XT // Oscillator Selection bits (XT
4 #pragma config FCKSM = OSC1_PRI // Power-up Timer Enable bit (PWU)
5 #pragma config BOD = OFF          // Brown-out Detector Enable bit (BOD
6 #pragma config WDT = OFF          // Watchdog Timer Enable bit (WDT
7 #pragma config C2CKMX = ON         // C2CK MX2 Mix bit (C2CK input/output
8 #pragma config PBADEN = OFF        // PORTB A/D Enable bit (PORTBx4:10
9 #pragma config MCLE = ON          // MCLR Pin Enable bit (MCLR pin a
10 #pragma config LVP = OFF          // Single-Supply ICSP Enable bit (0
11
12 #include <xc.h>
13 #define _XTAL_FREQ 48000000UL      //frecuencia de trabajo
14
15 unsigned char menu1[] = {"Bienvenido al ejemplo de la semana 13"};
16 unsigned char menu2[] = {"Elige la opcion a ejecutar"};
17 unsigned char menu3[] = {"r" - Enciende el motor"};
18 unsigned char menu4[] = {"p" - Apaga el motor"};
19 unsigned char menu5[] = {"i" - Ingresar cadena"};
20 unsigned char menu6[] = {"No debes de pasar de 20 caracteres"};
21 unsigned char menu7[] = {"m" - muestra menu"};
22 unsigned char motoron[] = {"Motor funcionando"};
23 unsigned char motoroff[] = {"Motor detenido"};
24 unsigned char errores[] = {"Tecla incorrecta, intente de nuevo"};
25 unsigned char opt_1[] = {"El mensaje ingresado fue:"};
26
27 unsigned char ingreso[] = {" "};
28
29
30 unsigned char indicador = 0;
31 unsigned char indice = 0;
32
33 void PORT_config(void){
34     TRISBbits.RB0 = 0;           //REO salida
35     ADCON1 = 0x0F;              //Puertos ANx en digital
36 }
37
38 void EUSART_config(void){
39     SPBRGH = 0;                 //Ignorado debido a que BRG1=0
40     SPBRGL = 0;
41     TRISDbits.RC6 = 0;          //Puerto RC6 como salida, no es m
42     RCREAbits.RCEN = 1;          //Encendemos el puerto serial
43     TXSTAbits.TXEN = 1;          //Encendemos el transmisor
44     RXSTAbits.CREN = 1;          //Encendemos el receptor
45 }
46
47 void INT_config(void){
48     INTCONbits.GIE = 1;          //Interruptor general habilitado
49     INTCONbits.PIE1 = 1;          //Interruptor de perifericos habilitado
50     PIE1bits.RCIE = 1;            //Habilitador de interrupciones de rezo
51 }
52
53 void EUSART_siguientelinea(void){
54     TXREG = 0xA;
55     while(TXSTAbits.TRMT == 0);
56     TXREG = 0x0D;
57     while(TXSTAbits.TRMT == 0);
58 }
59
60 void EUSART_enviacadena(const unsigned char *vector,unsigned char pos){
61     for (unsigned char x=0;x<pos;x++){
62         TXREG = vector[x];
63         while(TXSTAbits.TRMT == 0);
64     }
65 }
66
67 void show_menu(void){
68     EUSART_enviacadena(menu1,37);
69     EUSART_siguientelinea();
70     EUSART_enviacadena(menu2,37);
71     EUSART_siguientelinea();
72     EUSART_enviacadena(menu3,37);
73     EUSART_siguientelinea();
74     EUSART_enviacadena(menu4,37);
75     EUSART_siguientelinea();
76     EUSART_enviacadena(menu5,37);
77     EUSART_siguientelinea();
78     EUSART_enviacadena(menu6,37);
79     EUSART_siguientelinea();
80 }
81
82 void main(void) {
83     EUSART_config();
84     INT_config();
85     PORT_config();
86     show_menu();
87     while(1);
88 }
89
90 void __interrupt(high_priority) RC_Isr(void){
91     PIR1bits.RC1F = 0;
92     if(indicador == 0){
93         if(opt_1[0] == 'r'){
94             LATBbits.LED = 1;
95             EUSART_enviacadena(motoron,37);
96             EUSART_siguientelinea();
97         }
98         else if(opt_1[0] == 'p'){
99             LATBbits.LED = 0;
100            EUSART_enviacadena(motorof,37);
101            EUSART_siguientelinea();
102        }
103        else if(opt_1[0] == 'i'){
104            EUSART_enviacadena(rmenu6,37);
105            EUSART_siguientelinea();
106            indicador = 1;
107            indice = 0;
108        }
109        else if(opt_1[0] == 'm'){
110            show_menu();
111        }
112        else{
113            EUSART_enviacadena(errores,37);
114            EUSART_siguientelinea();
115        }
116    }
117    else{
118        //Partida para recibir la cadena y
119        if(opt_1[0] == '0x00'){
120            //Finalizo el ingreso de la cadena
121            EUSART_enviacadena(opt_1,37);
122            EUSART_siguientelinea();
123            EUSART_enviacadena(ingreso,20);
124            EUSART_siguientelinea();
125            indicador = 0;
126        }
127        else{
128            ingreso[indice] = RCREG;
129            indice++;
130        }
131    }
132 }
133

```

25

Fin de la sesión

26