Práctica 6 & 7 – Comunicaciones serie

Jaime Arana
Manuel Ferrero
3ºA

Comunicaciones serie I

Recepción de caracteres mediante la UART1

```
#include <xc.h>
#include <stdint.h>
#include "Pic32Ini.h"

#define PIN_LED0 0
#define PIN_LED1 1
#define PIN_LED2 2
#define PIN_LED3 3

#define PIN_RB13 13

/************ CÁLCULO DE UxBRG ************/
/* UxBRG = (f_reloj / (16 * Baud Rate)) - 1 */
/* U1BRG = (5 mHz / (16 * 9600 baudios)) - 1 = 31,55 = 32 */
/******************************************/

/******* REVISAR EL VALOR CALCULADO *****/
/* Baud rate = f_reloj / (16 * (UxBRG + 1)) */
/* Baud rate = 5 mHz / (16 * (32 + 1)) = 9469,7 */
/* error = (9600 - 9469,7) / 9600 = 1,36% < 2% asi que BIEN*/
#define BAUDIOS 32

int main (void) {
    int caracter;

    int puerto = 0x0000;
    puerto |= (1 << PIN_LED0);
    puerto |= (1 << PIN_LED1);
    puerto |= (1 << PIN_LED2);
    puerto |= (1 << PIN_LED3);
```

```
31
32      ANSELC = ~puerto;
33      //ANSELC &= ~puerto;    // LEDs como digitales
34      ANSELB &= ~(1 << PIN_RB13); // Se configura RB13 como digital
35
36      TRISB |= (1 << PIN_RB13); // RB13 como entrada
37      TRISC &= ~(1 << PIN_LED0);    // LEDs como salida
38      TRISC &= ~(1 << PIN_LED1);
39      TRISC &= ~(1 << PIN_LED2);
40      TRISC &= ~(1 << PIN_LED3);
41
42      LATC = puerto;  // LEDs apagados
43
44      SYSKEY = 0xAA996655;
45      SYSKEY = 0x556699AA;
46      U1RXR = 3;  // Pin de recepción de la UART al RB13
47      SYSKEY = 0x1CA11CA1;
48      U1BRG = BAUDIOS;
49
50      // Por defecto trabaja con el formato 8N1
51      U1MODEbits.BRGH = 0;
52      U1MODEbits.PDSEL = 0;
53      U1MODEbits.STSEL = 0;
54
55      U1STAbits.URXEN = 1; // Se habilita el receptor
56      U1MODE = 0x8000; // Se arranca la UART
57
58      while(1){
59          while(U1STAbits.URXDA == 0)
60              ; // Espera un nuevo carácter
61          // Se lee el carácter
62          caracter = U1RXREG;
63          // LATC = ~(caracter & puerto);
64          LATC = ~(caracter & 0x0F);
65      }
66  }
```

Transmisión de caracteres mediante la UART1

```c
#include <xc.h>
#include <stdint.h>
#include "Pic32Ini.h"

#define PIN_PULSADOR 5
#define PIN_RB7 7

/*********** CÁLCULO DE UxBRG ***********/
/* UxBRG = (f_reloj / (16 * Baud Rate)) - 1 */
/* U1BRG = (5 mHz / (16 * 9600 baudios)) - 1 = 31,55 = 32 */
/****************************************/

/******* REVISAR EL VALOR CALCULADO *****/
/* Baud rate = f_reloj / (16 * (UxBRG + 1)) */
/* Baud rate = 5 mHz / (16 * (32 + 1)) = 9469,7 */
/* error = (9600 - 9469,7) / 9600 = 1,36% < 2% asi que BIEN*/
#define BAUDIOS 32

int main (void)
{
    int pulsador_ant, pulsador_act;
    char texto [] = "Hola mundo\r\n";
    char *punt; // tiene que ser puntero para ser de tamaño variable

    ANSELB &= ~(1 << PIN_RB7); // Se configura RB7 como digital
    TRISB &= ~(1 << PIN_RB7);
    TRISB |= (1 << PIN_PULSADOR);  // pulsador como entrada
    LATB = 0; // 1 = transmisor inhabilitado | 0 = habilitado


    // Por defecto trabaja con el formato 8N1
    U1MODEbits.BRGH = 0;
    U1MODEbits.PDSEL = 0;
    U1MODEbits.STSEL = 0;

    U1STAbits.UTXEN = 0; // Se inhabilita el transmisor
    U1MODE = 0x8000; // Se arranca la UART

    pulsador_ant = (PORTB>>PIN_PULSADOR) & 1;

    while(1){
        pulsador_act = (PORTB >> PIN_PULSADOR) & 1;
        if ((pulsador_act != pulsador_ant) && (pulsador_act == 0)){
            U1STAbits.UTXEN = 1;    // habilitar transmisor

            punt = texto;   // escribir frase
            while (*punt != '\0'){  // iteramos hasta la final de la cadena
                U1TXREG = *punt;    // enviamos caracter
                while(U1STAbits.TRMT == 0)
                    ;
                punt++;
            }
            U1STAbits.UTXEN = 0;    // inhabilitar transmisión al terminar
        }
        pulsador_ant = pulsador_act;
    }
}
```

Transmisión y recepción de caracteres

```
#include <xc.h>
#include <stdint.h>
#include "Pic32Ini.h"

#define PIN_PULSADOR 5
#define PIN_LED0 0
#define PIN_LED1 1
#define PIN_LED2 2
#define PIN_LED3 3
#define PIN_RB13 13 // recepción
#define PIN_RB7 7   // transmisión

/************ CÁLCULO DE UxBRG ************/
/* UxBRG = (f_reloj / (16 * Baud Rate)) - 1 */
/* U1BRG = (5 mHz / (16 * 9600 baudios)) - 1 = 31,55 = 32 */
/******************************************/

/******* REVISAR EL VALOR CALCULADO *****/
/* Baud rate = f_reloj / (16 * (UxBRG + 1)) */
/* Baud rate = 5 mHz / (16 * (32 + 1)) = 9469,7 */
/* error = (9600 - 9469,7) / 9600 = 1,36% < 2% asi que BIEN*/
#define BAUDIOS 32

int main (void) {
    int c;
    int pulsador_ant, pulsador_act;
    char texto [] = "Hola mundo";
    char *punt;

    int puerto = 0x0000;
    puerto |= (1 << PIN_LED0);
    puerto |= (1 << PIN_LED1);
    puerto |= (1 << PIN_LED2);
    puerto |= (1 << PIN_LED3);
```

```
35
36      ANSELC &= ~0xF; // Pines de los LEDs como digitales
37      // ANSELC &= ~puerto;
38      ANSELB &= ~((1 << 13)|(1 << 7)); // Se configura RB13 como digital
39
40      TRISB |= (1<<13) | (1<<PIN_PULSADOR);   // pulsador como entrada
41      TRISC &= ~(1 << PIN_LED0);   // LEDs como salida
42      TRISC &= ~(1 << PIN_LED1);
43      TRISC &= ~(1 << PIN_LED2);
44      TRISC &= ~(1 << PIN_LED3);
45
46      LATC = puerto;  // LEDs apagados
47
48      SYSKEY=0xAA996655;
49      SYSKEY=0x556699AA;
50      U1RXR = 3;  // pin de recepción de la UART al RB13
51      RPB7R = 1;  // pin para rb7
52      SYSKEY=0x1CA11CA1;
53      U1BRG = BAUDIOS;
54
55      // Por defecto trabaja con el formato 8N1
56      U1MODEbits.BRGH = 0;
57      U1MODEbits.PDSEL = 0;
58      U1MODEbits.STSEL = 0;
59
60      U1STAbits.URXEN = 1; // Se habilita el receptor
61      U1STAbits.UTXEN = 0; // Se inhabilita el receptor
62      U1MODE = 0x8000; // Se arranca la UART
63
64      while(1){
65          // RECEPTOR
66          if(U1STAbits.URXDA == 1){
67              c = U1RXREG;
68              LATC = ~(c & 0x0F);
69              // LATC = ~(C & puerto);
70          }
71
72          // TRANSMISOR
73          pulsador_act = (PORTB >> PIN_PULSADOR) & 1;
74          if ((pulsador_act != pulsador_ant) && (pulsador_act == 0)){
75              U1STAbits.UTXEN = 1;    // habilitar transmisor
76
77              punt = texto;   // escribir frase
78              while (*punt != '\0'){  // iteramos hasta la final de la cadena
79                  U1TXREG = *punt;    // enviamos caracter
80                  while(U1STAbits.TRMT == 0)
81                      ;
82                  punt++;
83              }
84              U1STAbits.UTXEN = 0;    // inhabilitar transmisión al terminar
85          }
86          pulsador_ant = pulsador_act;
87
88      }
89  }
```

Eco

```c
#include <xc.h>
#include <stdint.h>
#include "Pic32Ini.h"

#define PIN_PULSADOR 5
#define PIN_LED0 0
#define PIN_LED1 1
#define PIN_LED2 2
#define PIN_LED3 3
#define PIN_RB13 13 // recepción
#define PIN_RB7 7   // transmisión

/************ CÁLCULO DE UxBRG ************/
/* UxBRG = (f_reloj / (16 * Baud Rate)) - 1 */
/* U1BRG = (5 mHz / (16 * 9600 baudios)) - 1 = 31,55 = 32 */
/******************************************/

/******* REVISAR EL VALOR CALCULADO *****/
/* Baud rate = f_reloj / (16 * (UxBRG + 1)) */
/* Baud rate = 5 mHz / (16 * (32 + 1)) = 9469,7 */
/* error = (9600 - 9469,7) / 9600 = 1,36% < 2% asi que BIEN*/
#define BAUDIOS 32

int main (void) {
    int c;

    int puerto = 0x0000;
    puerto |= (1 << PIN_LED0);
    puerto |= (1 << PIN_LED1);
    puerto |= (1 << PIN_LED2);
    puerto |= (1 << PIN_LED3);

    ANSELC &= ~0xF; // Pines de los LEDs como digitales
    // ANSELC &= ~puerto;
    ANSELB &= ~((1 << 13)|(1 << 7)); // Se configura RB13 como digital
```

```
37      TRISB |= (1<<13) | (1<<PIN_PULSADOR);   // pulsador como entrada
38      TRISC &= ~(1 << PIN_LED0);   // LEDs como salida
39      TRISC &= ~(1 << PIN_LED1);
40      TRISC &= ~(1 << PIN_LED2);
41      TRISC &= ~(1 << PIN_LED3);
42
43      LATC = puerto;  // LEDs apagados
44
45      SYSKEY=0xAA996655;
46      SYSKEY=0x556699AA;
47      U1RXR = 3;  // pin de recepción de la UART al RB13
48      RPB7R = 1;  // pin para rb7
49      SYSKEY=0x1CA11CA1;
50      U1BRG = BAUDIOS;
51
52      //Por defecto trabaja con el formato 8N1
53      U1MODEbits.BRGH = 0;
54      U1MODEbits.PDSEL = 0;
55      U1MODEbits.STSEL = 0;
56
57      U1STAbits.URXEN = 1; // Se habilita el receptor
58      U1STAbits.UTXEN = 0; // Se inhabilita el transmisor
59      U1MODE = 0x8000; // Se arranca la UART
60
61      while(1){
62          if(U1STAbits.URXDA == 1){
63              c = U1RXREG;
64              LATC = ~(c & 0x0F);
65              U1STAbits.UTXEN = 1;     // habilitar transmisión
66              U1TXREG = c;
67              while(U1STAbits.TRMT == 0)
68                  ;
69              U1STAbits.UTXEN = 0;     // deshabilitar transmisión
70          }
71      }
72  }
```

Comunicaciones serie II

Módulo para gestionar la UART1 con colas e interrupciones

```c
#include <xc.h>
#include <stdint.h>
#include "Pic32Ini.h"
#include "UART1.h"

#define BAUDIOS 9600

int main (void) {
    char c[] = "12";
    InicializarUART1(BAUDIOS);

    INTCONbits.MVEC = 1;  // activar interrupciones
    asm(" ei");

    while(1){
       c[0] = getcUART();
       c[1] = '\0';
       putsUART(c);
    }

    return 0;
}
```

```
#include <xc.h>
#include <stdint.h>
#include "Pic32Ini.h"

#define TAM_COLA 100

#define FREC_RELOJ 5000000

#define RB7 7
#define RB13 13

static char cola_rx[TAM_COLA]; // cola de recepción
static int icab_rx = 0; // índice para añadir
static int icol_rx = 0; // índice para leer
static char cola_tx[TAM_COLA]; // cola de transmisión
static int icab_tx = 0; // índice para añadir
static int icol_tx = 0; // índice para leer

__attribute__((vector(32), interrupt(IPL3SOFT),nomips16))
void InterrupcionUART1(void) {
    if(IFS1bits.U1RXIF == 1){ // Ha interrumpido el receptor
        if( (icab_rx+1 == icol_rx) || (icab_rx+1 == TAM_COLA && icol_rx == 0)){
          // La cola está llena
        } else {
            cola_rx[icab_rx] = U1RXREG; // Lee carÃ¡cter de la UART
            icab_rx++;
            if(icab_rx == TAM_COLA){
                icab_rx = 0;
            }
        }
        IFS1bits.U1RXIF = 0; // Y para terminar se borra el flag
    }
    if(IFS1bits.U1TXIF == 1){ // Ha interrumpido el transmisor
    // Se extrae un carácter de la cola y se envía
        if(icol_tx != icab_tx){ // Hay datos nuevos
            U1TXREG = cola_tx[icol_tx];
            icol_tx++;
            if(icol_tx == TAM_COLA){
                icol_tx = 0;
            }
        }else{ // Se ha vaciado la cola
            IEC1bits.U1TXIE = 0; // Para evitar bucle sin fin
        }
        IFS1bits.U1TXIF = 0; // Y para terminar se borra el flag
    }
}
void InicializarUART1(int baudios){
    double valor;
    if(baudios > 38400){
        valor = (FREC_RELOJ/(4*baudios)) - 1;     //Modo de alta velocidad con BRGH = 1
        if((FREC_RELOJ%(4*9600)) > (9600*2)){
            valor++;     // aproximación hacia arriba, menor error
        }
    } else{
        valor = FREC_RELOJ/(16*baudios) - 1;
        if((FREC_RELOJ%(16*9600)) > (9600*8)){
            valor++;     // aproximación hacia arriba, menor error
        }
    }

    U1BRG = (int) valor;

```

```
64      // Activamos las interrupciones del receptor
65      IFS1bits.U1RXIF = 0; // Borro flag receptor
66      IEC1bits.U1RXIE = 1; // Habilito interrupciones
67      IFS1bits.U1TXIF = 0; // Borro flag del transmisor
68      IPC8bits.U1IP = 3; // Prioridad 3
69      IPC8bits.U1IS = 1; // Subprioridad 1
70
71      // Conectamos U1RX y U1TX a los pines RB13 y RB7 del micro
72      ANSELB &= ~((1 << RB13)|(1 << RB7)); // Pines digitales
73      TRISB |= (1 << RB13); // Y RB13 como entrada
74      LATB |= (1 << RB7); // A 1 si el transmisor está inhabilitado.
75
76      SYSKEY=0xAA996655; // Se desbloquean los registros
77      SYSKEY=0x556699AA; // de configuración.
78      U1RXR = 3; // U1RX conectado a RB13
79      RPB7R = 1; // U1TX conectado a RB7
80      SYSKEY=0x1CA11CA1; // Se vuelven a bloquear
81
82      U1STAbits.URXISEL = 0; // Interrupción cuando llegue 1 char
83      U1STAbits.UTXISEL = 2; // Interrupciçon cuando FIFO vacía
84      U1STAbits.URXEN = 1; // habilitar el receptor
85      U1STAbits.UTXEN = 1; // habilitar el transmisor
86      U1MODE = 0x8000; // Se arranca la UART
87  }
89  void putsUART(char *ps) {
90      while(*ps != '\0'){ // iteramos por la frase
91          if( (icab_tx+1 == icol_tx) || (icab_tx+1 == TAM_COLA && icol_tx == 0)){
92              // La cola está llena --> Se aborta el envío de los caracteres que restan
93              break;
94          }else{
95              cola_tx[icab_tx] = *ps; // Copia el carácter en la cola
96              ps++;  //Apunto al siguiente carácter de la cadena
97              icab_tx++;
98              if(icab_tx == TAM_COLA){
99                  icab_tx = 0;
100             }
101         }
102     }
103     // Se habilitan las interrupciones del transmisor para comenzar a enviar
104     IEC1bits.U1TXIE = 1;
105 }
107 char getcUART(void) {
108     char c;
109
110     if(icol_rx != icab_rx){ // Hay datos nuevos
111         c = cola_rx[icol_rx];
112         icol_rx++;
113         if(icol_rx == TAM_COLA){
114             icol_rx=0;
115         }
116     }else{ // no ha llegado nada
117         c = '\0';
118     }
119     return c;
120 }
```

Bluetooth bit whacker

```c
#include <xc.h>
#include <stdint.h>
#include "UART1.h"
#include "Pic32Ini.h"
#include <string.h>

#define BAUDIOS 9600

void analizarCadena(char *c);
int hex2int(char ch);

int main (void) {
    char cad[100];
    InicializarUART1(BAUDIOS);
    int i = 0;
    char c;

    INTCONbits.MVEC = 1; // habilitar interrupciones
    asm(" ei");

    while(1){
        c = getcUART();
        if(c != '\0'){
            cad[i] = c;
            i++;
            if(c == '\n'){
                analizarCadena(cad);
                i = 0;
            }
        }
    }
}
```

```
34  void analizarCadena(char *c) {
35      int pin = hex2int(c[5]);
36      char res[20] = "OK\n";
37      //char *res = malloc(20); strcpy(res, "OK\n");
38      int val_port = 0;
39      int valor = 0;
40      //Así da error el primer PD,C,0,0 que haces --> después bien
41
42      if(strncmp(c, "PD",2)==0){ //TRIS
43          valor = hex2int(c[7]);
44          if((strncmp(c+3,"A",1)==0)){
45              if(valor==0){
46                  TRISA &= ~(1 << pin);
47              }else if(valor==1){
48                  TRISA |= 1 << pin;
49              }else{
50              strcpy(res,"Error\n");
51              }
52          }else if((strncmp(c+3,"B",1)==0)){
53              if(valor==0){
54                  TRISB &= ~(1 << pin);
55              }else if(valor==1){
56                  TRISB |= 1 << pin;
57              }else{
58                  strcpy(res,"Error\n");
59              }
60          }else if((strncmp(c+3,"C",1)==0)){
61              if(valor==0){
62                  TRISC &= ~(1 << pin);
63              }else if(valor==1){
64                  TRISC |= 1 << pin;
65              }else{
66                  strcpy(res,"Error\n");
67              }
68          }else{
69              strcpy(res,"Error\n");
70          }
71      } else if(strncmp(c, "PI",2)==0){ //PORT
```

```
72          if((strncmp(c+3,"A",1)==0)){
73              if(((PORTA >> pin) & 1)==1){
74                  strncpy(res,c,3); //PI,
75                  char num[2] = "1";
76                  strcat(res,num);
77                  strcat(res,"\n");
78              }else if(((PORTA >> pin) & 1)==0){
79                  strncpy(res,c,3); //PI,
80                  char num[2] = "0";
81                  strcat(res,num);
82                  strcat(res,"\n");
83              }else{
84                  strcpy(res,"Error\n");
85              }
86          }else if((strncmp(c+3,"B",1)==0)){
87              if(((PORTB >> pin) & 1)==1){
88                  strncpy(res,c,3); //PI,
89                  char num[2] = "1";
90                  strcat(res,num);
91                  strcat(res,"\n");
92              }else if(((PORTB >> pin) & 1)==0){
93                  strncpy(res,c,3); //PI,
94                  char num[2] = "0";
95                  strcat(res,num);
96                  strcat(res,"\n");
97              }else{
98                  strcpy(res,"Error\n");
99              }
100         }else if((strncmp(c+3,"C",1)==0)){
101             if(((PORTC >> pin) & 1)==1){
102                 strncpy(res,c,3); //PI,
103                 char num[2] = "1";
104                 strcat(res,num);
105                 strcat(res,"\n");
106             }else if(((PORTC >> pin) & 1)==0){
107                 strncpy(res,c,3); //PI,
```

```
                char num[2] = "0";
                strcat(res,num);
                strcat(res,"\n");
            }else{
                strcpy(res,"Error\n");
            }
        }else{
            strcpy(res,"Error\n");
        }
    }else if(strncmp(c, "PO",2)==0){ //LAT
        valor = hex2int(c[7]);
        if((strncmp(c+3,"A",1)==0)){
            if(valor==0){
                LATA &= ~(1 << pin);
            }else if(valor==1){
                LATA |= 1 << pin;
            }else{
                strcpy(res,"Error\n");
            }
        }else if((strncmp(c+3,"B",1)==0)){
            if(valor==0){
                LATB &= ~(1 << pin);
            }else if(valor==1){
                LATB |= 1 << pin;
            }else{
                strcpy(res,"Error\n");
            }
        }else if((strncmp(c+3,"C",1)==0)){
            if(valor==0){
                LATC &= ~(1 << pin);
            }else if(valor==1){
                LATC |= 1 << pin;
            }else{
                strcpy(res,"Error\n");
            }
        }else{
            strcpy(res,"Error\n");
        }
    }else{
        strcpy(res,"Error\n");
    }
    putsUART(res);
}

int hex2int(char ch) {
    if (ch >= '0' && ch <= '9'){
        return ch - '0';
    }
    if (ch >= 'A' && ch <= 'F'){
        return ch - 'A' + 10;
    }
    if (ch >= 'a' && ch <= 'f'){
        return ch - 'a' + 10;
    }
    return -1;
}
```