Calificación P5-P6: 10

MICROPROCESADORES

INFORME PRÁCTICAS 5 Y 6

JORGE CANDIA

21 de marzo de 2023

Práctica 5/6 (UART mediante polling)

main3.c

```
#include <stdio.h>
#include <stdlib.h>

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

#define PIN_RECEPCION 13

static uint32_t lectura = 0;

void initUART1(void){
    //Configuro el pin receptor (Rx)
    SYSKEY = 0xAA996655;
    SYSKEY = 0x556699AA;
    U1RXR = 3;  //Receptor de la UART1 a RB13
    SYSKEY = 0x1CA11CA1;

    //Ahora configuro los parámetros de funcionamiento de la UART
    U1BRG = 32; //9600 baudios
    U1STAbits.URXEN = 1; //Siempre habilitado para reibir
    U1MODE = 0x8000; //El on a la UART1, PDSEL a 00 para modo 8N1
}



int main(void){

    ANSELB &= ~(1 << PIN_RECEPCION);

    TRISA = 0;
    TRISB = (1 << PIN_RECEPCION);
    TRISC = 0;  //RC0-3 LEDS

    LATC = ~0;

    initUART1();


    while(1){ //Gestiono la recepción por polling
        if(U1STAbits.URXDA == 1){
            lectura = U1RXREG; //Push and dump el valor más alto de la
```

```
FIFO
        }

        //Actualizo los LEDs en cada iteración del bucle
        //lectura |= ~(0xF); //Pongo a 1 todo lo q no me interesa
        LATC = ~lectura;

    }
                                                                BIEN
}
```

main4.c

```
#include <stdio.h>
#include <stdlib.h>

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

#define PIN_TRANSMISION 7
#define PIN_PULSADOR 5

static char mensaje[] = "Hola ICAI!"; //Lleva el '\0' al final
automáticamente

void initUART1(void){
    //Configuro el pin receptor (Rx)
    SYSKEY = 0xAA996655;
    SYSKEY = 0x556699AA;
    RPB7R = 1;  //Emisor de la UART1 a RB7
    SYSKEY = 0x1CA11CA1;

    //Ahora configuro los parámetros de funcionamiento de la UART
    U1BRG = 32; //9600 baudios
    LATB |= (1 << PIN_TRANSMISION); //A 1 al empezar, bit de stop
    U1MODE = 0x8000; //El on a la UART1, PDSEL a 00 para modo 8N1
}


void transmitir(void){
    U1STAbits.UTXEN = 1;
    for(int i=0; mensaje[i] != '\0'; i++){
```

```
        U1TXREG = mensaje[i];
        while(U1STAbits.TRMT == 0);
    }
    U1STAbits.UTXEN = 0;
}



int main(void){

    ANSELB &= ~(1 << PIN_TRANSMISION);

    TRISA = 0;
    TRISB = ((1 << PIN_TRANSMISION) | (1 << PIN_PULSADOR));
    TRISC = 0;  //RC0-3 LEDS

    LATC = ~3;

    initUART1();

    int pulsador_ant , pulsador_act ;
    pulsador_ant = (PORTB >> PIN_PULSADOR ) & 1;

    while(1){ //Gestiono la transmisión por polling



        pulsador_act = ( PORTB >> PIN_PULSADOR ) & 1;
        if( ( pulsador_act != pulsador_ant ) && ( pulsador_act== 0) ) {
            transmitir();

        }
        pulsador_ant = pulsador_act;

    }

}
```

BIEN

main5.c

```
#include <stdio.h>
```

```c
#include <stdlib.h>

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

#define PIN_RECEPCION 13
#define PIN_TRANSMISION 7
#define PIN_PULSADOR 5

static uint32_t lectura = 0;
static char mensaje[] = "Hola ICAI!"; //Lleva el '\0' al final
automáticamente

void initUART1(void){
    //Configuro el pin receptor (Rx)
    SYSKEY = 0xAA996655;
    SYSKEY = 0x556699AA;
    U1RXR = 3;  //Receptor de la UART1 a RB13
    RPB7R = 1;  //Emisor de la UART1 a RB7
    SYSKEY = 0x1CA11CA1;

    //Ahora configuro los parámetros de funcionamiento de la UART
    U1BRG = 32; //9600 baudios
    U1STAbits.URXEN = 1; //Siempre habilitado para reibir
    LATB |= (1 << PIN_TRANSMISION); //A 1 al empezar, bit de stop
    U1MODE = 0x8000; //El on a la UART1, PDSEL a 00 para modo 8N1
}

void transmitir(void){
    U1STAbits.UTXEN = 1;
    for(int i=0; mensaje[i] != '\0'; i++){
        U1TXREG = mensaje[i];
        while(U1STAbits.TRMT == 0);
    }
    U1STAbits.UTXEN = 0;
}



int main(void){

    ANSELB &= ~((1 << PIN_RECEPCION) | (1 << PIN_TRANSMISION) | (1 <<
PIN_PULSADOR));

    TRISA = 0;
```

```
    TRISB = ((1 << PIN_RECEPCION) | (1 << PIN_TRANSMISION) | (1 <<
PIN_PULSADOR));
    TRISC = 0;  //RC0-3 LEDS

    LATC = ~0;

    initUART1();

    int pulsador_ant , pulsador_act ;
    pulsador_ant = (PORTB >> PIN_PULSADOR ) & 1;

    while(1){ //Gestiono la recepción por polling

        ////////// RECEPCIÓN //////////
        if(U1STAbits.URXDA == 1){
            lectura = U1RXREG; //Push and dump el valor más alto de la
FIFO
        }

        //Actualizo los LEDs en cada iteración del bucle
        LATC = ~lectura;


        ////////// TRANSMISIÓN //////////
        pulsador_act = ( PORTB >> PIN_PULSADOR ) & 1;
        if( ( pulsador_act != pulsador_ant ) && ( pulsador_act== 0) ) {
            transmitir();
        }
        pulsador_ant = pulsador_act;

    }
                                                                BIEN
}
```

Práctica 6/7 (UART mediante interrupciones)

Main1.c

```c
#include <xc.h>
#include <stdint.h>
#include "Pic32Ini.h"
#include "DriverUART1.h"

#define baudios 9600


int main(void){

    InicializarUART1(9600);

    char eco[2];
    eco[0] = '\0';
    eco[1] = '\0';

    while(1){

        char c = getcUART();

        if(c != '\0'){
            eco[0] = c;
            putsUART(eco);
        }
    }

                                                    BIEN
}
```

DriverUART1.c

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

#define PIN_RX 13 //RB13
#define PIN_TX 7  //RB7


#define TAM_COLA 100 //Tamaño de nuestra cola en nº de caracteres/bytes
typedef struct cola {
```

```
    int icabeza;
    int icola;
    char cola[TAM_COLA];
} cola_t;

static cola_t cola_tx, cola_rx;



void InicializarUART1(int baudios){

    //Inicializo pines
    ANSELB = 0;

    TRISA = 0;
    TRISB = (1 << PIN_RX);
    TRISC = 0;

    LATA = 0;
    LATB = (1 << PIN_TX); //Bit de stop
    LATC = 0;


    //Inicializo UART1
    if(baudios < 38400){
        U1MODE = 0x8000;
        U1BRG = round(5000000/(16*baudios));
    } else {
        U1MODE = 0x8008;
        U1BRG = round(5000000/(4*baudios));
    }

    U1STA = 0x9400; //UTXISEL a 2, URXISEL a 0 y ambos enables RX y TX

    //PARA PROBAR, BORRAR LUEGO
    //U1MODE = 0x8000;
    //U1BRG = 32;


    //Remapeo de pines
    SYSKEY = 0xAA996655;
    SYSKEY = 0x55669955;
    U1RXR = 3;
    RPB7R = 1;
    SYSKEY = 0X1CA11CA1;
```

```
    //Inicializo interrupciones
    IFS1CLR = 3<<8; //Bajo flag de RX y TX
    IEC1SET = 1<<8;
    IEC1CLR = 1<<9; //ENABLE (mask) de TX desactivado
    IPC8SET = 3<<2; //Prioridad a 3
    IPC8CLR = 1<<4;
    IPC8SET = 1<<0; //Subprioridad a 1
    IPC8CLR = 1<<1;

    INTCON |= 1<<12;

    asm("ei");


    //Inicializo colas
    cola_rx.icabeza=0, cola_rx.icola=0;
    cola_tx.icabeza=0, cola_tx.icola=0;
}




//RX -- Gestiono los caracteres de la cola de RX
char getcUART(void){
    char c;
    if(cola_rx.icola != cola_rx.icabeza){ //Miro si hay datos nuevos,
cabeza == cola
        c = cola_rx.cola[cola_rx.icola]; //Cargo el valor de la cola de
RX
        cola_rx.icola++; //Aumento en 1 posición la cola
        if(cola_rx.icola == TAM_COLA){
            cola_rx.icola = 0;
        }
    } else {
        c = '\0'; //No ha llegado nada -> devuelvo \0
    }
    return c;
}



//TX -- Lleno la cola de TX
void putsUART(char *s[]){ //s[] es la 1ª posi de memoria del vector s[]
```

```
    while(*s != '\0'){ //Mientras la cadena no haya llegado a su fin
        if((cola_tx.icabeza + 1 == cola_tx.icola) || ((cola_tx.icabeza +
1 == TAM_COLA) && cola_tx.icabeza == 0)){
            break; //Cola llena -> salgo del bucle
        } else {
            cola_tx.cola[cola_tx.icabeza] = *s;
            s++; //Sólo sumo 1? Por qué no 8??
            cola_tx.icabeza++;
            if(cola_tx.icabeza == TAM_COLA){
                cola_tx.icabeza = 0;
            }
        }
    }

    //Activo interrupciones de TX (mask) para que empiece a transmitir
    //Se desactivan en la interrupción cuando se ha transmitido todo
(cabeza == cola)
    IEC1SET = 1<<9;
}




__attribute__((vector(32), interrupt(IPL3SOFT), nomips16))
void InterrupcionUART1(void){

    //RX -- Lleno la cola de RX
    if((IFS1>>8)&1 == 1){
        if((cola_rx.icabeza + 1 == cola_rx.icola) || (cola_rx.icabeza +
1 == TAM_COLA && cola_rx.icola == 0)){
            //Cola llena, no hago nada
        } else {
            cola_rx.cola[cola_rx.icabeza] = U1RXREG;
            cola_rx.icabeza++; //La cabeza reposa sobre el siguiente
caracter a sobreescribir
            if(cola_rx.icabeza == TAM_COLA){
                cola_rx.icabeza = 0;
            }
        }
        IFS1CLR = 1<<8; //En comunicaciones bajo la flag al final
    }
```

```
    //TX -- Gestiono el envío de TX
    if((IFS1>>9)&1 == 1){ //Interrupción cuando FIFO vacía
        if(cola_tx.icola != cola_tx.icabeza){ //Compruebo que haya datos
para transmititr
            U1TXREG = cola_tx.cola[cola_tx.icola];
            cola_tx.icola++;
            if(cola_tx.icola == TAM_COLA){
                cola_tx.icola = 0;
            }
        } else { //No queda nada para transmitir, bajo ENABLE (mask) de
TX
            IEC1CLR = 1<<9;
        }
        IFS1CLR = 1<<9;
    }

}
```

DriverUART1.h

```
#ifndef UART_H
#define UART_H
#include <stdint.h> // Define uint32_t

void InicializarUART1 (int baudios );
void putsUART ( char *s[]) ;
char getcUART ( void ) ;


#endif
```

Main2.c

```
#include <xc.h>
#include <stdint.h>
#include "Pic32Ini.h"
#include "DriverUART2.h"

#define baudios 9600
```

```
int main(void){

    InicializarUART12(baudios);
    char limpiezaInit = getcUART2();

    while(1){

    }

}
```

DriverUART2.c

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

#define PIN_RX 13 //RB13
#define PIN_TX 7  //RB7


static int trigger = 0;

#define TAM_COLA 100 //Tamaño de nuestra cola en nº de caracteres/bytes
typedef struct cola {
    int icabeza;
    int icola;
    char cola[TAM_COLA];
} cola_t;

static cola_t cola_tx, cola_rx;



void InicializarUART12(int baudios){

    //Inicializo pines
    ANSELB = 0;

    TRISA = 0;
    TRISB = (1 << PIN_RX);
    TRISC = 0;

    LATA = 0;
```

```
    LATB = (1 << PIN_TX); //Bit de stop
    LATC = 0;


    //Inicializo UART1
    if(baudios < 38400){
        U1MODE = 0x8000;
        U1BRG = round(5000000/(16*baudios));
    } else {
        U1MODE = 0x8008;
        U1BRG = round(5000000/(4*baudios));
    }

    U1STA = 0x9400; //UTXISEL a 2, URXISEL a 0 y ambos enables RX y TX

    //PARA PROBAR, BORRAR LUEGO
    //U1MODE = 0x8000;
    //U1BRG = 32;


    //Remapeo de pines
    SYSKEY = 0xAA996655;
    SYSKEY = 0x55669955;
    U1RXR = 3;
    RPB7R = 1;
    SYSKEY = 0X1CA11CA1;


    //Inicializo interrupciones
    IFS1CLR = 3<<8; //Bajo flag de RX y TX
    IEC1SET = 1<<8;
    IEC1CLR = 1<<9; //ENABLE (mask) de TX desactivado
    IPC8SET = 3<<2; //Prioridad a 3
    IPC8CLR = 1<<4;
    IPC8SET = 1<<0; //Subprioridad a 1
    IPC8CLR = 1<<1;

    INTCON |= 1<<12;

    asm("ei");


    //Inicializo colas
    cola_rx.icabeza=0, cola_rx.icola=0;
    cola_tx.icabeza=0, cola_tx.icola=0;
}
```

```
//RX -- Gestiono los caracteres de la cola de RX
char getcUART2(void){
    char c;
    if(cola_rx.icola != cola_rx.icabeza){ //Miro si hay datos nuevos,
cabeza == cola
        c = cola_rx.cola[cola_rx.icola]; //Cargo el valor de la cola de
RX
        cola_rx.icola++; //Aumento en 1 posición la cola
        if(cola_rx.icola == TAM_COLA){
            cola_rx.icola = 0;
        }
    } else {
        c = '\0'; //No ha llegado nada -> devuelvo \0
    }
    return c;
}



//TX -- Lleno la cola de TX
void putsUART2(char *s[]){ //s[] es la 1ª posi de memoria del vector s[]
    while(*s != '\0'){ //Mientras la cadena no haya llegado a su fin
        if((cola_tx.icabeza + 1 == cola_tx.icola) || ((cola_tx.icabeza +
1 == TAM_COLA) && cola_tx.icabeza == 0)){
            break; //Cola llena -> salgo del bucle
        } else {
            cola_tx.cola[cola_tx.icabeza] = *s;
            s++; //Sólo sumo 1? Por qué no 8??
            cola_tx.icabeza++;
            if(cola_tx.icabeza == TAM_COLA){
                cola_tx.icabeza = 0;
            }
        }
    }

    //Activo interrupciones de TX (mask) para que empiece a transmitir
    //Se desactivan en la interrupción cuando se ha transmitido todo
(cabeza == cola)
    IEC1SET = 1<<9;
}
```

```
void evaluarCadena(void){
    char cadena[] = getcUART2();

    char tipo = cadena[1];
    char puerto = cadena[3];
    int pin = cadena[5];
    int direccion = cadena[7];


    if(tipo == 'D'){
        if(puerto == 'A'){
            if(pin==1){
                TRISASET = 1<<pin;
                putsUART2("Ok\n");
            } else {
                TRISACLR = 1<<pin;
                putsUART2("Ok\n");
            }
        } else if(puerto == 'B'){
            if(pin==1){
                TRISBSET = 1<<pin;
                putsUART2("Ok\n");
            } else {
                TRISBCLR = 1<<pin;
                putsUART2("Ok\n");
            }
        } else if(puerto == 'C'){
            if(pin==1){
                TRISCSET = 1<<pin;
                putsUART2("Ok\n");
            } else {
                TRISCCLR = 1<<pin;
                putsUART2("Ok\n");
            }
        } else {
            putsUART2("ERROR\n");
        }
    }
}
```

```
__attribute__((vector(32), interrupt(IPL3SOFT), nomips16))
void InterrupcionUART12(void){

    //RX -- Lleno la cola de RX
    if((IFS1>>8)&1 == 1){
        char c = U1RXREG;

        if((cola_rx.icabeza + 1 == cola_rx.icola) || (cola_rx.icabeza +
1 == TAM_COLA && cola_rx.icola == 0)){
            //Cola llena, no hago nada
        } else if(c != '\n') {
            cola_rx.cola[cola_rx.icabeza] = c;
            cola_rx.icabeza++; //La cabeza reposa sobre el siguiente
caracter a sobreescribir
            if(cola_rx.icabeza == TAM_COLA){
                cola_rx.icabeza = 0;
            }
        }

        if(c == '\n'){
            evaluarCadena();
        }

        IFS1CLR = 1<<8; //En comunicaciones bajo la flag al final
    }


    //TX -- Gestiono el envío de TX
    if((IFS1>>9)&1 == 1){ //Interrupción cuando FIFO vacía
        if(cola_tx.icola != cola_tx.icabeza){ //Compruebo que haya datos
para transmititr
            U1TXREG = cola_tx.cola[cola_tx.icola];
            cola_tx.icola++;
            if(cola_tx.icola == TAM_COLA){
                cola_tx.icola = 0;
            }
        } else { //No queda nada para transmitir, bajo ENABLE (mask) de
TX
            IEC1CLR = 1<<9;
        }
        IFS1CLR = 1<<9;
    }

}
```

DriverUART2.h

```
#ifndef UART_H
#define UART_H
#include <stdint.h> // Define uint32_t

void InicializarUART12 (int baudios );
void putsUART2 ( char *s[]) ;
char getcUART2 ( void ) ;


#endif
```