Universidad del Valle de Guatemala — Natalia de León Bercián

Electrónica Digital 2 — carne: 18193

Mayo 2021 — sección: 20

Laboratorio # 7

ISR – TMR0 y UART

Link del video:

https://youtu.be/167QisWAL5g

Link del GitHub:

https://github.com/nataliadlb/LABS_REPOSITORIO.git

Código

```
//**************************************************************************
/**
 Laboratorio # 7
 Natalia de León Bericán
 carne: 18193
 Digital 2
 Sección 20
 Mayo 2021
*/
//**************************************************************************

#include <stdint.h>
#include <stdbool.h>
#include "inc/tm4c123gh6pm.h"
#include "inc/hw_memmap.h"
#include "driverlib/debug.h"
#include "inc/hw_types.h"
#include "inc/hw_ints.h"
#include "driverlib/sysctl.h"
#include "driverlib/systick.h"
#include "driverlib/interrupt.h"
#include "driverlib/gpio.h"
#include "driverlib/timer.h"
#include "driverlib/uart.h"
#include "driverlib/rom.h"
#include <string.h>
#include "grlib/grlib.h"
#include "driverlib/pin_map.h"

//#define XTAL 16000000
#define LED_ROJO GPIO_PIN_1
#define LED_VERDE GPIO_PIN_3
#define LED_AZUL GPIO_PIN_2

//**************************************************************************
//
// VARIABLES
```

```
//
//*****************************************************************************
uint32_t ui32Period; //periodo para TMR0
uint32_t ui32Status; //Status de UART

uint8_t Ban_rojo = 0;
uint8_t Ban_verde = 0;
uint8_t Ban_azul = 0;
char letra;

//*****************************************************************************
//
// PROTOTIPOS DE FUNCIONES
//
//*****************************************************************************
void Timer0IntHandler(void);
void InitUART(void);
void UARTIntHandler(void);


//*****************************************************************************
//
// The error routine that is called if the driver library encounters an error.
//
//*****************************************************************************
#ifdef DEBUG
void
__error__(char *pcFilename, uint32_t ui32Line)
{
    while(1);
}
#endif

//*****************************************************************************
//
// CONFIGURACION
//
//*****************************************************************************

int main(void){

    //Config de reloj a 40MHz
    SysCtlClockSet(SYSCTL_SYSDIV_5 | SYSCTL_USE_PLL | SYSCTL_XTAL_16MHZ |
SYSCTL_OSC_MAIN);

    // Config reloj para habilitar puerto F
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);

    // Verificar que se habilitó puerto F
    while(!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOF)){
     }

    // Configurar OUTPUTS/INPUTS
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3);

    /**
        CONFIGURACION UART
    */
    // Se inicializa la comunicación UART
    InitUART();
```

```
    /**
        CONFIGURACION TIMER
    */
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
    while(!SysCtlPeripheralReady(SYSCTL_PERIPH_TIMER0)){
    }

    // Configuración del Timer 0 como temporizador periódico
    TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);

    // Se calcula el período para el temporizador (1 seg)
    ui32Period = (SysCtlClockGet()) / 2;

    // Establecer el periodo del temporizador
    TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period - 1);

    TimerEnable (TIMER0_BASE, TIMER_A|TIMER_B);

    // Se establece que exista la interrupción por Timeout
    TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);

    //Establecemos la ISR
    TimerIntRegister(TIMER0_BASE, TIMER_A, Timer0IntHandler);

    // Se habilita la interrupción por el TIMER0A
    IntEnable(INT_TIMER0A);

    // Se habilita el Timer
    TimerEnable(TIMER0_BASE, TIMER_A);


    // Se manda mensajes por UART
    UARTCharPut(UART0_BASE, 'C');
    UARTCharPut(UART0_BASE, 'o');
    UARTCharPut(UART0_BASE, 'm');
    UARTCharPut(UART0_BASE, 'e');
    UARTCharPut(UART0_BASE, 'n');
    UARTCharPut(UART0_BASE, 'z');
    UARTCharPut(UART0_BASE, 'a');
    UARTCharPut(UART0_BASE, 'm');
    UARTCharPut(UART0_BASE, 'o');
    UARTCharPut(UART0_BASE, 's');
    UARTCharPut(UART0_BASE, 10);
    UARTCharPut(UART0_BASE, 13);


    //*****************************************************************************
    // Loop forever.
    //*****************************************************************************

    while(1){

    }
}

//*****************************************************************************
//
// FUNCIONES
//
//*****************************************************************************
```

```
//*****************************************************************************
**********************
// Handler de la interrupcion del TIMER 0 - Recordar modificar el archivo
tm4c123ght6pm_startup_css.c
//*****************************************************************************
**********************
void Timer0IntHandler(void){
    // Clear the timer interrupt
    TimerIntClear(TIMER0_BASE, TIMER_TIMA_TIMEOUT);

    //LED ROJO
    if (letra == 'r' & Ban_rojo == 1){ //activar toggle led rojo
        if (GPIOPinRead(GPIO_PORTF_BASE, LED_ROJO)){
            GPIOPinWrite(GPIO_PORTF_BASE, LED_ROJO, 0);
        }
        else {
            GPIOPinWrite(GPIO_PORTF_BASE, LED_ROJO , LED_ROJO);
        }
    }
    else if (letra == 'r' & Ban_rojo == 2){ //desactivar toggle led rojo
        GPIOPinWrite(GPIO_PORTF_BASE, LED_ROJO , 0);
        Ban_rojo = 0;
    }
    //LED VERDE
    else if (letra == 'g' & Ban_verde == 1){ //activar toggle led verde
        if (GPIOPinRead(GPIO_PORTF_BASE, LED_VERDE)){
            GPIOPinWrite(GPIO_PORTF_BASE, LED_VERDE, 0);
        }
        else {
            GPIOPinWrite(GPIO_PORTF_BASE, LED_VERDE , LED_VERDE);
        }
    }
    else if (letra == 'g' & Ban_verde == 2){ //desactivar toggle led verde
        GPIOPinWrite(GPIO_PORTF_BASE, LED_VERDE , 0);
        Ban_verde = 0;
    }

    //LED AZUL
    if (letra == 'b' & Ban_azul == 1){ //activar toggle led azul
        if (GPIOPinRead(GPIO_PORTF_BASE, LED_AZUL)){
            GPIOPinWrite(GPIO_PORTF_BASE, LED_AZUL, 0);
        }
        else {
            GPIOPinWrite(GPIO_PORTF_BASE, LED_AZUL , LED_AZUL);
        }
    }
    else if (letra == 'b' & Ban_azul == 2){ //desactivar toggle led azul
        GPIOPinWrite(GPIO_PORTF_BASE, LED_AZUL , 0);
        Ban_azul = 0;
    }

}

//*****************************************************************************
**********************
// Handler de la interrupcion del UART - Recordar modificar el archivo
tm4c123ght6pm_startup_css.c
//*****************************************************************************
**********************
void UARTIntHandler(void){
        // Get the interrupt status
```

```
        ui32Status = UARTIntStatus(UART0_BASE, true);

        // Clear the asserted interrupts.
        UARTIntClear(UART0_BASE, ui32Status);

        // Loop while there are characters in the receive FIFO.
        while(UARTCharsAvail(UART0_BASE)){
            letra = UARTCharGet(UART0_BASE);
            UARTCharPutNonBlocking(UART0_BASE,letra); //manda a un puerto especifico

            if (letra == 'r'){ //depende el caracter se aumenta la bandera de ese led
                Ban_rojo++;
            }
            else if (letra == 'g'){
                Ban_verde++;
            }
            else if (letra == 'b'){
                Ban_azul++;
            }

        }
}


//*****************************************************************************
**********************
// Inicialización de UART - Tomado del ejemplo que subio Pablo Mazariegos
//*****************************************************************************
**********************
void InitUART(void){

    /*Enable the peripheral UART Module 0*/
    SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
    while(!SysCtlPeripheralReady(SYSCTL_PERIPH_UART0)){
    }

    /*Enable the GPIO Port A*/
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);

    GPIOPinConfigure(GPIO_PA0_U0RX);
    GPIOPinConfigure(GPIO_PA1_U0TX);

    // Se habilitan las interrupciones Globales
    IntMasterEnable();

    /* Make the UART pins be peripheral controlled. */
    GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);

    UARTDisable(UART0_BASE);
    /* Sets the configuration of a UART. */
    UARTConfigSetExpClk(
            UART0_BASE, SysCtlClockGet(), 115200,
            (UART_CONFIG_WLEN_8 | UART_CONFIG_STOP_ONE | UART_CONFIG_PAR_NONE));
    IntEnable (INT_UART0);

    UARTIntEnable(UART0_BASE, UART_INT_RX | UART_INT_RT);
    UARTEnable (UART0_BASE);

}
```