Cristopher Sagastume 18640
Experimento 2: Tiva C ISRs Timers y UART

Utilizando la plataforma Tiva C

Este experimento servirá para que el estudiante se familiarice con la configuración de las ISR, Timers y el módulo UART. Utilizando programación en C con la ayuda de la librería DriverLib de TivaWare o CMSIS. Si no tuviera la Tiva C, podrá simularlo observando el comportamiento de los registros correspondientes.

Parte1

Copie las siguientes líneas para incluir los archivos header necesarios

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/tm4c123gh6pm.h"
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "inc/hw_ints.h"
#include "driverlib/sysctl.h"
#include "driverlib/interrupt.h"
#include "driverlib/gpio.h"
#include "driverlib/timer.h"
```

Parte 2.

Configure el reloj del sistema.

Para más información sobre el reloj puede irse a partir de la página 219 de la hoja de datos de la Tiva C en la sección del Reloj y verifique las tablas 5-4, 5-5 y 5-6 para ver las posibles opciones del reloj del sistema. Configure el reloj a 40 MHz.

Configure el reloj para habilitar el puerto F

Para más información puede irse a la página 340 de la hoja de datos y observe las opciones del registro RCGCGPIO.

Configure los pines de los Leds Rojo, Verde y Azul como salidas.

Configure el Timer 0

Configure el Timer 0, como un timer de 32-bits, en forma periódica, para ver más información sobre el timer irse a partir de la página 704 de la hoja de datos de la Tiva C.

Parte 3

Cree una rutina en la cual haga un toggle de un GPIO a 1Hz con un duty cycle del 50%.

Parte 4

Configure una interrupción a la mitad del periodo de la parte 3, es decir, a 0.5Hz.

Configure la Interrupción

Recuerde habilitar todas las interrupciones utilizando la instrucción IntMasterEnable().

Cree un Handler para la interrupción del Timer

```
void Timer0IntHandler(void){


}
```

Recuerde irse al archivo *tm4c123gh6pm_startup_ccs.c* y buscar donde está el comentario *// Timer 0 subtimer A*

Reemplazar IntDefaultHandler, por el nombre del handler de su función del Timer 0, en este caso sería **Timer0IntHandler,**

También tendrá que declarar la función al principio de este archivo de forma externa.

Buscar la línea donde se encuentra
extern void _c_int00(void);

y pegue el nombre de su función:

```
 extern void
Timer0IntHandler(void);
```

Parte 5.

Habilitar el módulo UART0 y los periféricos del GPIOA correspondientes al módulo UART.

Inicialice el módulo UART con los siguientes parámetros: 115200, 8 data bits, 1 stop bit, None parity.

Parte 6.

Habilite la interrupción para el módulo UART.

Recuerde irse al archivo *tm4c123gh6pm_startup_ccs.c* y buscar donde está el comentario *// UART0 Rx and Tx*

Reemplazar IntDefaultHandler, por el nombre del handler de su función del Timer 0, en este caso sería **UARTIntHandler,**

También tendrá que declarar la función al principio de este archivo de forma externa.

Buscar la línea donde se encuentra
```
extern void _c_int00(void);
```

y pegue el nombre de su función:
```
 extern void
UARTIntHandler(void);
```

Parte 7.

Haga una rutina donde recopile un carácter desde el módulo UART utilizando la interrupción.

Con esta rutina, utilice las letras “r”, “g” y “b”, recibidos desde el módulo UART para habilitar/deshabilitar el toggle de cada uno de los leds respectivos de la Tiva C.

Es decir, si recibe la letra “r”, habilita el toggle del led rojo, si vuelve a enviar la letra “r” lo deshabilita.

Pseudocodigo:

- Configurar Reloj del sistema
- Configurar TRM0 y sus interrupciones
- Configurar UART0, sus puertos y su interrupcion

- Configurar el puerto F para los led
- Verificar si se recibió un dato esto con un handler de UART
    - Si se recibió un dato se guarda en una variable
    - Si no se recibió un dato no pasara nada.
- Verificar que dato se recibió para encender los leds con un handler de TMR0
    - Si se recibió “r” encender led rojo
    - Si se recibió “g” encender led verde
    - Si se recibió “b” encender led azul
    - Si no se recibió alguna de las anteriores apagar led.

Código Comentado

```
//Cristopher Sagastume 18640

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


uint32_t dato = 0x67;
bool c = 0;


int main (void){
    //Se configura el reloj a 40Mhz
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);

    //Se habilitan los led como salidas (Puerto F)
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1);
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_2);
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_3);

    //Se configura el TMR0
```

```
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
    TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC); //configuro el TMR0 de
manera periodica
    TimerLoadSet(TIMER0_BASE, TIMER_BOTH, 20000000-1); //Se confitura el TMR0
a 32 bits medio ciclo
    TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT); //habilitamos la
interrupcion cada timeout del tmr0
    IntEnable(INT_TIMER0A);
    TimerEnable(TIMER0_BASE, TIMER_A); //habilitamos el TMR0

    //Se configura la comunicacion UART
    SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);//Se habilita el UART0
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);//Se habilita el puerto del
UART0
    GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 |GPIO_PIN_1); //definicion de
los pines del UART0
    //Configuracion de UART0 a 115200 con 8 bits de datos y 1 de stop sin
pariedad
    UARTConfigSetExpClk(UART0_BASE, SysCtlClockGet(), 115200,
(UART_CONFIG_WLEN_8 | UART_CONFIG_STOP_ONE |UART_CONFIG_PAR_NONE));

    //Se configuran las interrupciones de UART
    IntEnable(INT_UART0);
    UARTIntEnable(UART0_BASE, UART_INT_RX);
    UARTIntDisable(UART0_BASE, UART_INT_9BIT|UART_INT_TX
|UART_INT_OE|UART_INT_BE|UART_INT_PE|UART_INT_FE|UART_INT_RT|UART_INT_DSR|UART
_INT_DCD |UART_INT_CTS | UART_INT_RI);
    UARTFIFOLevelSet(UART0_BASE ,  UART_FIFO_TX1_8,  UART_FIFO_RX1_8);
    UARTEnable(UART0_BASE);

    IntMasterEnable();
    while(1){}

}

void Timer0IntHandler(void){
    //Se verifica el dato recibido
    if (c==0){
        //Si se recibe la letra r se enciende el led rojo
        if (dato == 'r'){

GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_3|GPIO_PIN_2|GPIO_PIN_1,0x02);
        }
        //Si se recibe la letra r se enciende el led azul
        if (dato == 'b'){// MANDO AZUL

GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_3|GPIO_PIN_2|GPIO_PIN_1,0x04);
        }
        //Si se recibe la letra r se enciende el led verde
        if (dato == 'g'){
```

```
GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_3|GPIO_PIN_2|GPIO_PIN_1,0x08);
        }
    c = 1;
    }else{
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3,
0x00);
        c = 0;
        }
    //si se genero una interrupcion se limpia la fuente de interrupcion para
evitar que vuelva a entrar a la misma generando un buble infinito
        TimerIntClear(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
}
void UART0IntHandler(void){
        //se verifica si hubo una recepcion de dato en el UART0
        if (UARTCharsAvail(UART0_BASE)==true){
            //Se guarda el dato recibido en la variable dato
            dato = UARTCharGet(UART0_BASE);
        }
        UARTRxErrorClear(UART0_BASE);
        //Se limpia la fuente de interrupcion de la interrpcion de la UART
            UARTIntClear(UART0_BASE, UART_INT_RX);
}
```