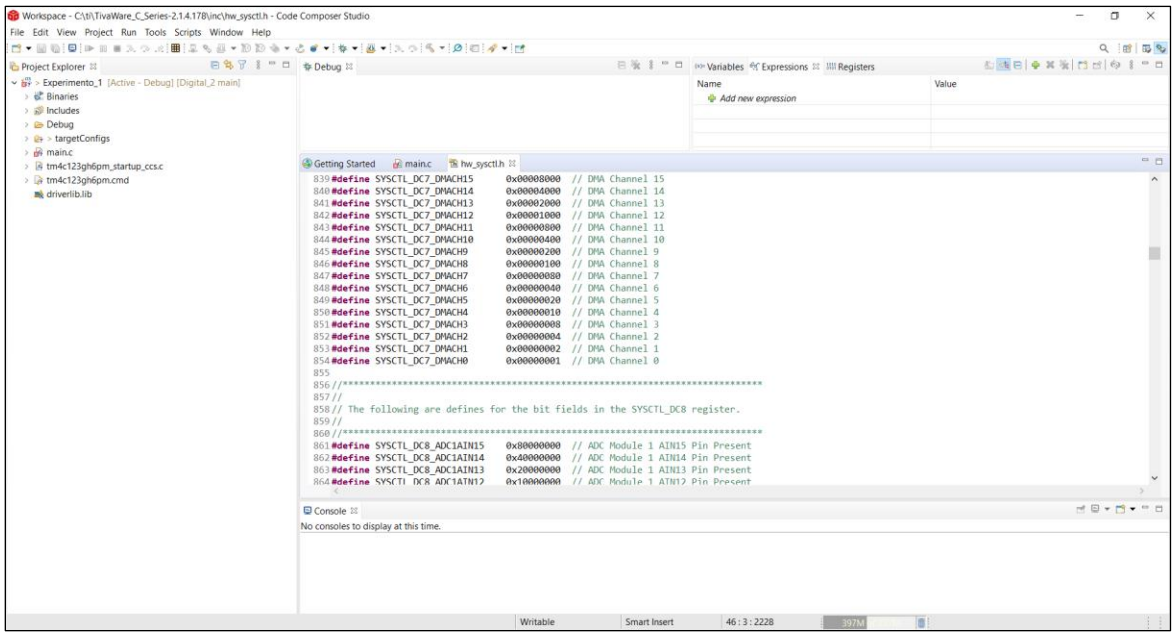
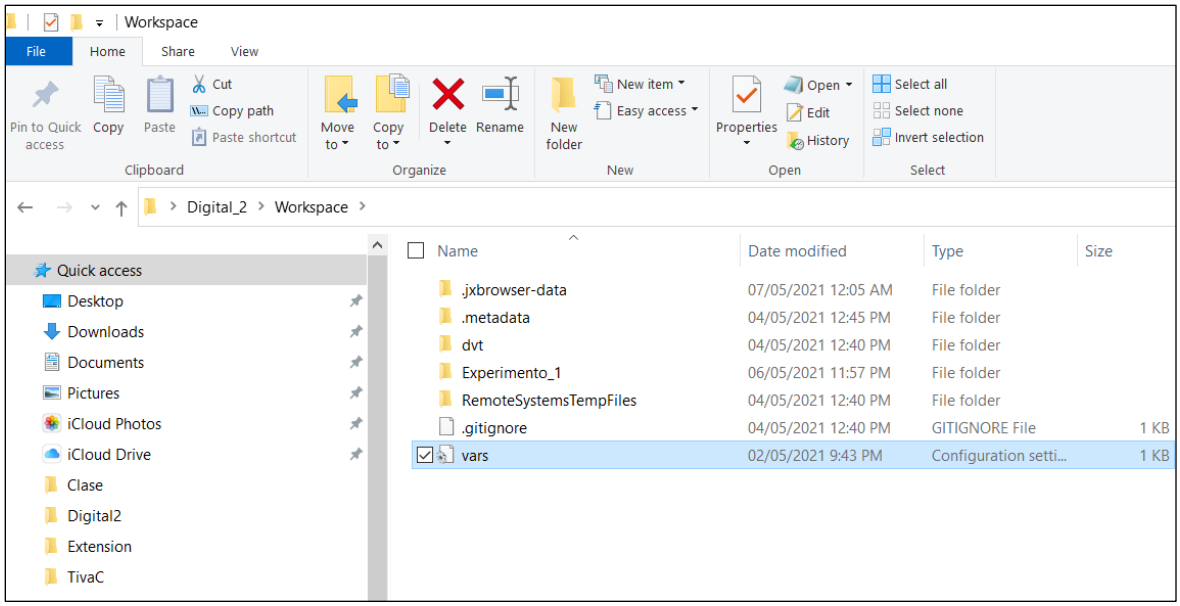
Universidad del Valle de Guatemala
Digital 2
Kurt Kellner
Pablo Rene Arellano Estrada

EXPERIMENTO 1 Tiva_C TIVAWARE

Parte 1:





PARTE 2 - RELOJ:

SysCtlClockSet(SYSCTL_OSC_MAIN | SYSCTL_XTAL_16MHZ | SYSCTL_USE_PLL | SYSCTL_SYSDIV_5);
SysCtlClockSet(SYSCTL_OSC_MAIN | SYSCTL_XTAL_16MHZ | SYSCTL_USE_OSC | SYSCTL_SYSDIV_1);

En esta función se establece el reloj a diferentes frecuencias utilizando el PLL. Se establece el reloj principal como el Main, a una oscilación de 16MHz, con el PLL activado el cual es (400MHz/2), y luego dividido 5, lo cual da como resultado 40MHz. Al cambiar el SYSCTL_SYSDIV se pueden obtener diferentes frecuencias. También la fuente de reloj se puede escoger con SYSCTL_OSC_MAIN, SYSCTL_OSC_INT, YSCTL_OSC_INT4, SYSCTL_OSC_INT30, y SYSCTL_OSC_EXT32. El cristal externo se puede escoger con SYSCTL_XTAL_4MHZ, SYSCTL_XTAL_4_09MHZ, SYSCTL_XTAL_4_91MHZ, entre otras opciones.

PARTE 2 – RELOJ PARA PUERTO F:

SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);

Función para habilitar un periférico. Existen distintos habilitadores de periféricos, pero los usados más frecuentes son los siguientes:

Periféricos	Registros Necesarios
Timer	SYSCTL_PERIPH_TIMER0, … SYSCTL_PERIPH_TIMER7
UART	SYSCTL_PERIPH_UART0, … SYSCTL_PERIPH_UART7
USB	SYSCTL_PERIPH_USB0
PWM	SYSCTL_PERIPH_PWM0, SYSCTL_PERIPH_PWM1
ADC	SYSCTL_PERIPH_ADC0, SYSCTL_PERIPH_ADC1

PARTE 2 – LEDS COMO SALIDA:

GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);

Con esta función se pueden utilizar los pines descritos como leds. Dependiendo del valor asignado. Si se coloca un valor de encendido decimal que corresponda a un número binario, se pueden realizar distintas combinaciones de colores con los pines ya habilitados.

GPIO Pin	Pin Function	Dispositivo
PF4	GPIO	SW1
PF0	GPIO	SW2
PF1	GPIO	RGB RED
PF2	GPIO	RGB BLUE
PF3	GPIO	RGB GREEN

PARTE 3 – COMBINACION DE COLORES:

GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, GPIO_PIN_1);
GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, GPIO_PIN_2);
GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, GPIO_PIN_3);
GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 6);

El primero enciende el color rojo, el segundo enciende el color azul y el tercero enciende el verde. Ya que el numero 6 en binario corresponde al azul y al rojo, se enciende el color morado. Esto se logra usando la tabla anterior para combinar colores.

PARTE 3 – SEMAFORO:

```
while(1)
{
    // Encender Led: Puerto, Pines, Valor
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, GPIO_PIN_3);

    // Delay de 4*(40MHz/3) = 4 segundos
    SysCtlDelay (4*40000000/3);

    // Encender y apagar por un segundo el Led verde
    for(ui32Loop = 0; ui32Loop < 3; ui32Loop++)
    {

        // Encender Led: Puerto, Pines, Valor
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);

        // Delay de (40MHZ/3) = 1 segundo
        SysCtlDelay (40000000/3);

        // Encender Led: Puerto, Pines, Valor
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, GPIO_PIN_3);

        // Delay de (40MHZ/3) = 1 segundo
        SysCtlDelay (40000000/3);
    }

    // Encender Led: Puerto, Pines, Valor
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 10);

    // Delay de 2*40MHz/3) = 2 segundos
    SysCtlDelay (2*40000000/3);

    // Encender Led: Puerto, Pines, Valor
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, GPIO_PIN_1);

    // Delay de (40MHz/3) = 1 segundos
    for(ui32Loop = 0; ui32Loop < 10000000; ui32Loop++)
    {
    }
```

PARTE 3 – DELAY:

1. SysCtlDelay (40000000/3);
2. for(ui32Loop = 0; ui32Loop < 10000000; ui32Loop++)

El primero se obtiene debido a que el valor que se pone en el Delay tarda tres ciclos de reloj y la frecuencia de 40MHz para un segundo, por eso se pone el valor allí mostrado. Para el segundo delay se usa un for que cuenta hasta 10 milloes para tres segundos.

PARTE 4 - BOTON:

En esta parte se desbloquea primero el SW2 y el SW1. Se establecen asimismo los pines de entrada y se configura la corriente y si es Weak PullUp o Weak PullDown. Finalmente se lee con la función **GPIOPinRead** los pines seleccionados por las mascaras para determinar el valor y luego tomar decisiones en base a la opción seleccionada.

```
// Desbloquear PF0 SW2 y SW1
GPIO_PORTF_LOCK_R = GPIO_LOCK_KEY;
GPIO_PORTF_CR_R = 0x0f;

// Enable GPIO para el puerto F. Pines de entrada
GPIOPinTypeGPIOInput(GPIO_PORTF_BASE, GPIO_PIN_0|GPIO_PIN_4);

// Configuracion: Puerto, Pines, Corriente, Pull Up de la Tiva
GPIOPadConfigSet(GPIO_PORTF_BASE,  GPIO_PIN_0|GPIO_PIN_4, GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD_WPU);

// Loop
//
while(1)
{
    Push = GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_0|GPIO_PIN_4);
    if (Push == 17)
    {
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
    }

    else if ((Push == 1) || (Push == 16))
    {
        // Encender Led: Puerto, Pines, Valor  para       VERDE
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, GPIO_PIN_3);
```

PARTE 4 - ANTIREBOTE:

Esta función permite presionar un botón un tiempo suficiente como para ejecutar un proceso, pero no para repetirlo infinitamente al leer valores demasiado pequeños o falsos.

```
void boton (void)
{
    boton();
    if (presionado == 1)                        // Boton presionado?
    {
       pressed_ok = pressed_ok + 1;         // Contador presionado con rango de seguridad
       released_ok = 0;                     // Contador suelto en cero
       if (pressed_ok > 500)                // Desborde contador?
       {
           if (presionado == 0)             // Estuvo suelto antes?
           {
               i = i + 1;                   // Incrementa contador
               presionado = 1;              // Se presiono

           }
           pressed_ok = 0;                  // Reinicia contador
        }
    }
    else if (presionado == 17)              // No presinado?
    {
       released_ok = released_ok + 1;       // Contador suelto con rango de seguridad
       pressed_ok = 0;                      // Contador presioando en cero
       if (released_ok > 500)               // Desborde contador?
       {
          presionado = 17;                  // Se solto
          released_ok = 0;                  // Reinicia contador
       }
    }
```