Universidad del Valle de Guatemala

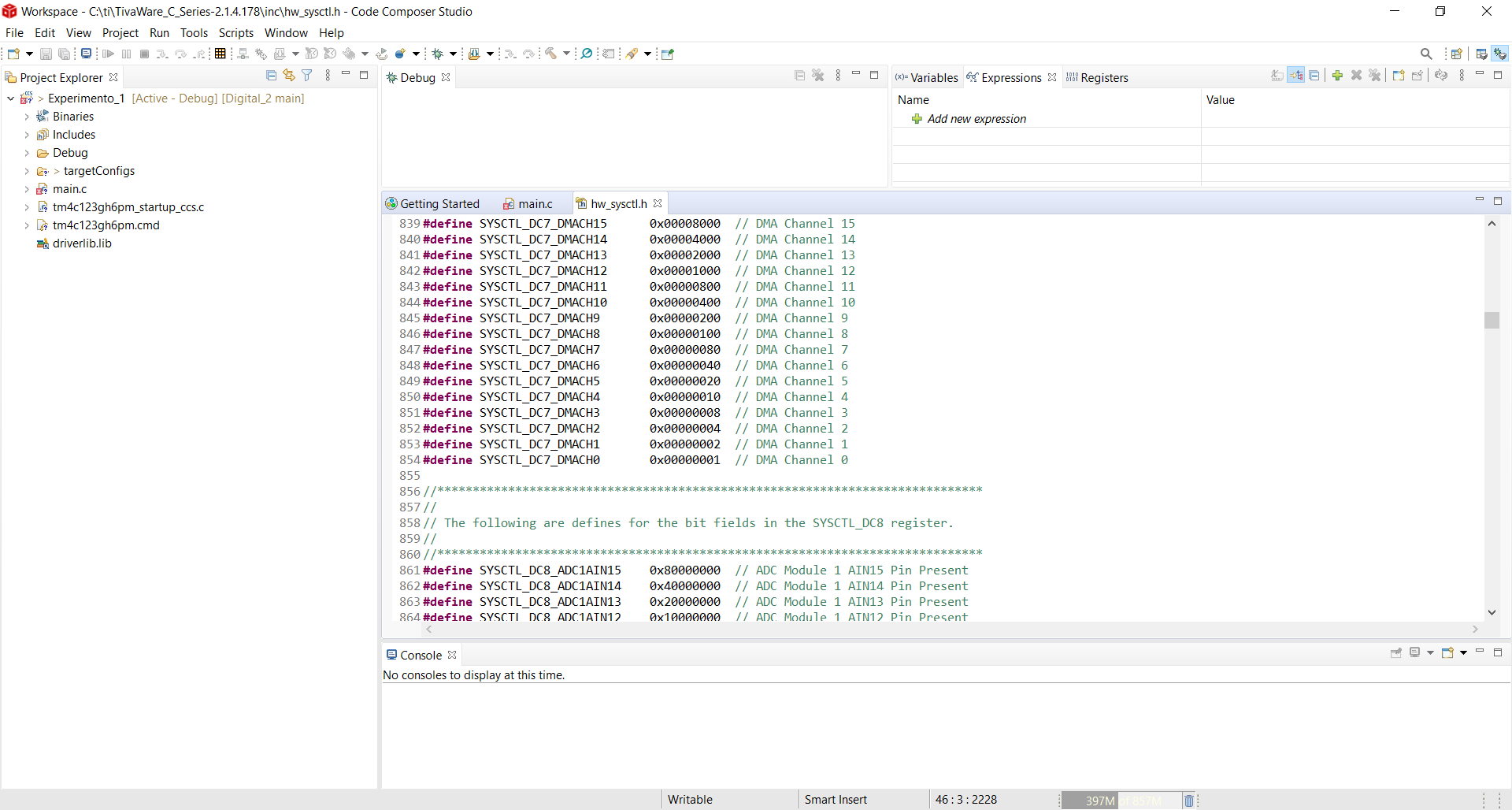
Digital 2

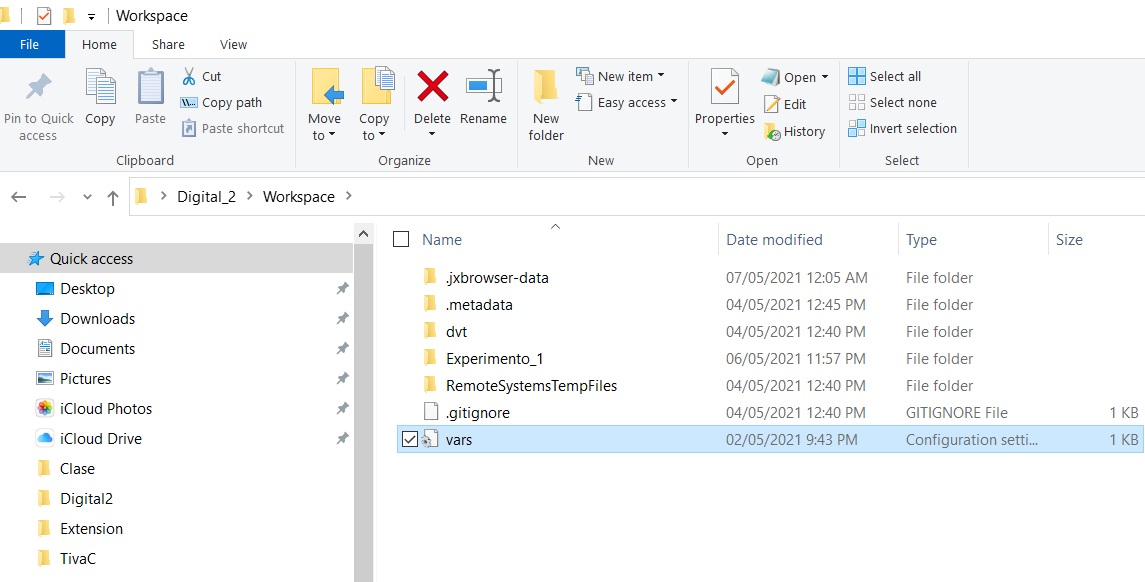
Kurt Kellner

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);**

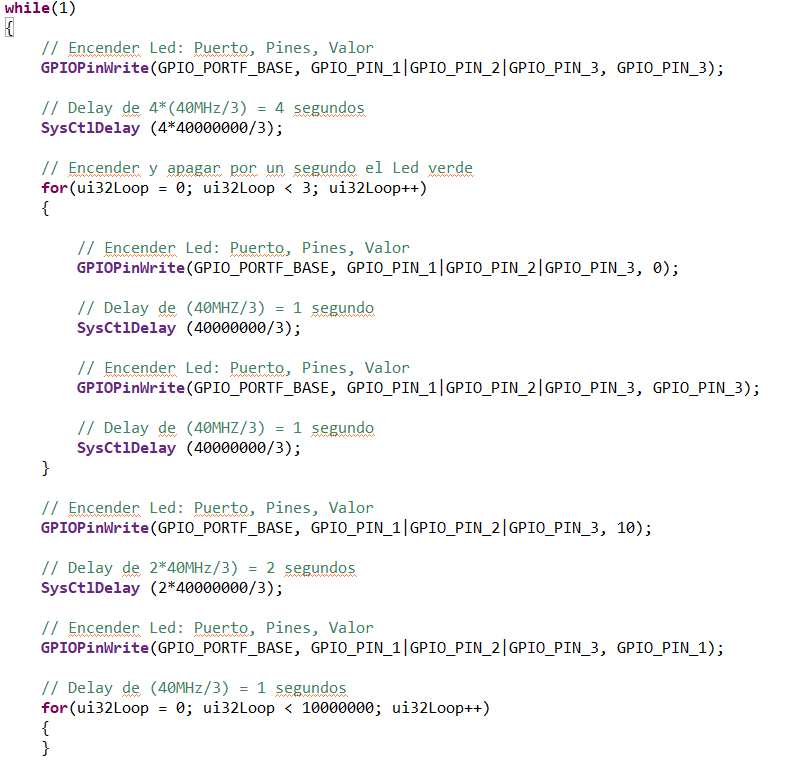
**GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3, GPIO\_PIN\_2);**

**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:**



**PARTE 3 – DELAY:**

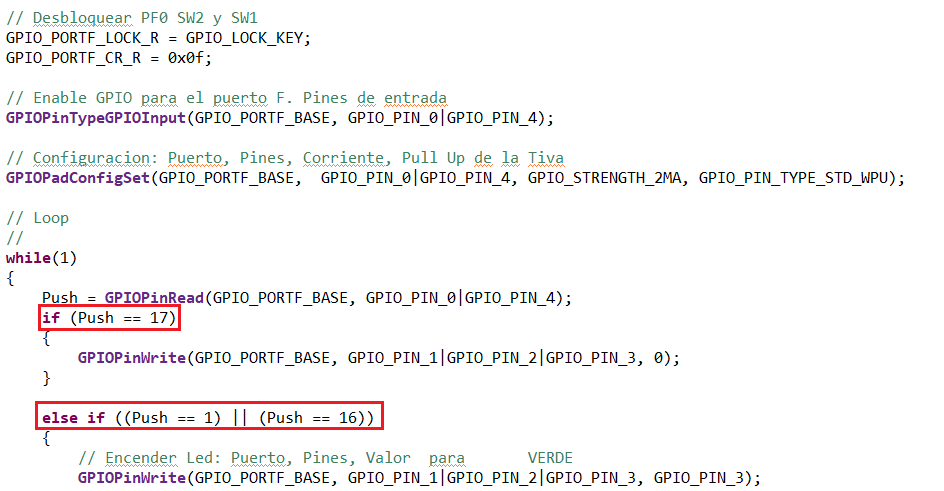
**1. SysCtlDelay (40000000/3);**

**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.

****

**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.

