Universidad Católica del Uruguay
Facultad de Ingeniería y Tecnologías
Sistemas Embebidos - 2022

Laboratorio 1

Introducción a la placa de desarrollo del curso y al MPLAB X IDE

Objetivo

Interiorizarse con el hardware de la placa de desarrollo del curso, el PIC32, y el entorno de desarrollo MPLAB X IDE. Comenzar a escribir código en lenguaje de programación C para aplicaciones embebidas. Desarrollar funciones básicas de E/S. Emplear las herramientas de compilación y depuración que ofrece el entorno de desarrollo.

Evaluación y metodología de trabajo

A partir de este laboratorio, y para todos los laboratorios restantes y el proyecto final, se trabajará en grupos (siempre el mismo). El docente asistirá a los estudiantes constantemente realizando sugerencias, devoluciones y aclarando conceptos cada vez que se necesite.
Se evaluará el trabajo de cada grupo durante el laboratorio, y se considerarán para dicha evaluación los siguientes criterios:

A. Organización del grupo, involucramiento de cada uno de los integrantes.
B. Completitud y correctitud de las soluciones a las tareas indicadas en el laboratorio en el tiempo de clase.
C. Respuestas a las preguntas planteadas en el laboratorio y/o preguntas que los docentes puedan realizar.
D. Utilización de buenas prácticas de programación.
    a. El código entregado **debe** compilar sin errores ni warnings. En caso de que por algún motivo excepcional sea necesario ignorar un warning, explicar porqué en el informe.
    b. En el código no deben haber funciones, variables, comentarios, etc. que no se utilicen. Por ejemplo, si al crear un archivo hay funciones de ejemplo, estas se deben quitar antes de entregar el código.
E. Entrega en fecha de las soluciones propuestas.

Ejercicios

Primera Parte

Introducción al MPLAB X IDE, creación de proyectos y primeros pasos.

1. **Creación de un proyecto en MPLAB X IDE**

   a. Abrir el MPLAB X IDE y crear un nuevo proyecto vacío (Standalone Project) para el PIC32MM0256GPM064, Hardware Tool: Simulator, y compilador XC32. Darle un nombre al proyecto, por ejemplo “Laboratorio” y guardarlo en el directorio que deseen.

   b. Analizar el árbol de directorios creado en el navegador del proyecto (vista “Projects”, no “Files”).

   c. Agregar un archivo ***main.c*** con formato xc32. Para ello, hacer click derecho en la carpeta *Source Files -> New xc32_newfile.c* y llamarlo main.c.

   d. Analizar el archivo generado: ¿qué secciones tiene? ¿cuales van a usar en el main.c? ¿cuales no y por qué?

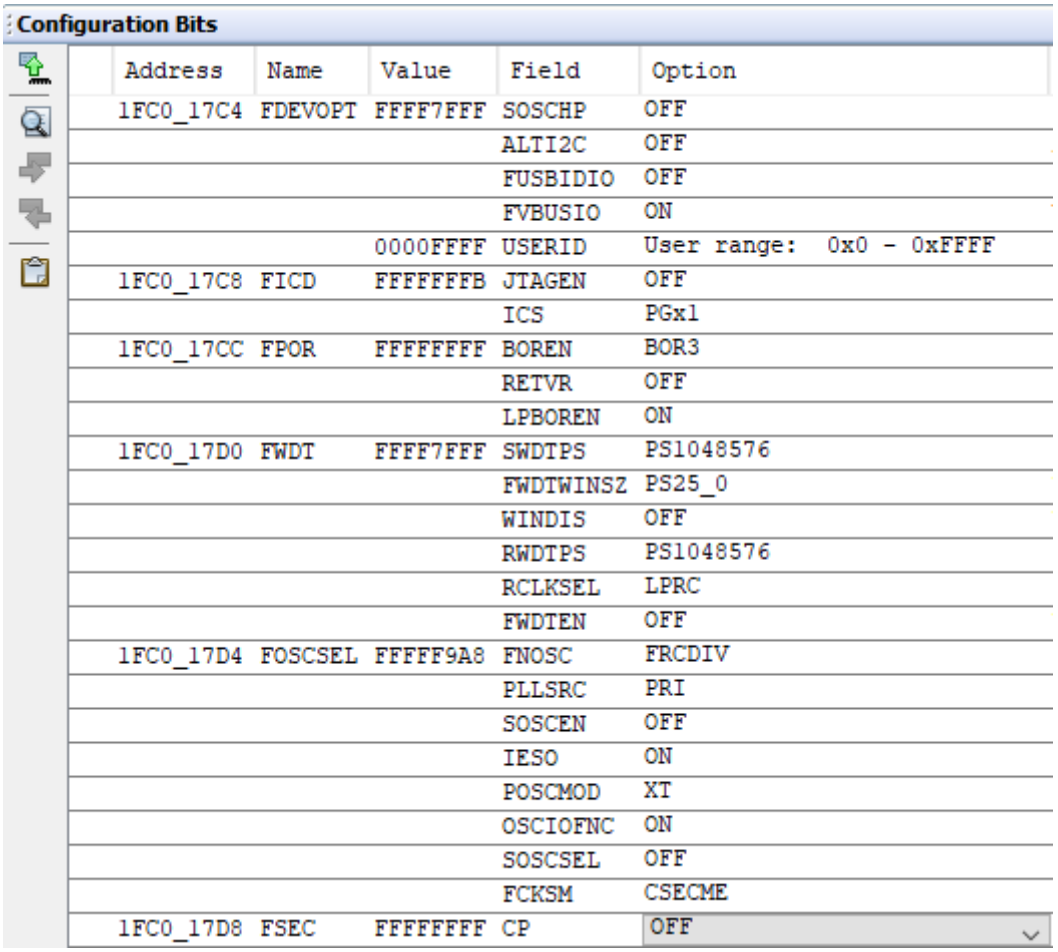
   e. Reemplazar la función **`ExampleInterfaceFunction`** por la **`main`** con la siguiente estructura:

```
int main( void ){
    while(1){
        Nop();
    }
    return 0;
}
```

2. **Bits de Configuración**

   a. Los dispositivos PIC32 tienen varios registros que contienen los ***bits de configuración***. Estos bits especifican la operación fundamental del dispositivo, como el modo del oscilador, el watchdog timer, el modo de programación y la protección de código. Si no se configuran correctamente los bits pueden provocar un error de código o un dispositivo que no se ejecuta correctamente.
   Estos ***bits de configuración*** se deben configurar en el código a través de la directiva ***`#pragma config setting_name = value_name.`***
   El nombre de configuración y valor son específicos del dispositivo y pueden determinarse mediante el documento *PIC32ConfigSet.html* ubicado en el directorio de instalación del compilador, carpeta ***docs***.
   Ver el archivo PIC32ConfigSet.html y analizar los distintos parámetros configurables.

   b. Afortunadamente, MPLAB X ofrece la configuración de los bits de configuración a través de una interfaz gráfica. Ir al menú Window -> Target Memory Views -> Configuration Bits, y en la ventana que se abre realizar la siguiente configuración:



Configuration Bits

Address	Name	Value	Field	Option
1FC0_17C4	FDEVOPT	FFFF7FFF	SOSCHP	OFF
			ALTI2C	OFF
			FUSBIDIO	OFF
			FVBUSIO	ON
		0000FFFF	USERID	User range: 0x0 - 0xFFFF
1FC0_17C8	FICD	FFFFFFFB	JTAGEN	OFF
			ICS	PGx1
1FC0_17CC	FPOR	FFFFFFFF	BOREN	BOR3
			RETVR	OFF
			LPBOREN	ON
1FC0_17D0	FWDT	FFFF7FFF	SWDTPS	PS1048576
			FWDTWINSZ	PS25_0
			WINDIS	OFF
			RWDTPS	PS1048576
			RCLKSEL	LPRC
			FWDTEN	OFF
1FC0_17D4	FOSCSEL	FFFFF9A8	FNOSC	FRCDIV
			PLLSRC	PRI
			SOSCEN	OFF
			IESO	ON
			POSCMOD	XT
			OSCIOFNC	ON
			SOSCSEL	OFF
			FCKSM	CSECME
1FC0_17D8	FSEC	FFFFFFFF	CP	OFF

Luego de seteados los parámetros, dar click en *Generate Source Code to Output* y copiar en el ***main.c*** ,antes de cualquier directiva `#include`, el código generado.

c. Analizar qué significa la configuración elegida.

3. **Compilación**

a. Agregar al archivo **main.c** la siguiente definición y las siguientes variables:

```
#define ARRAY_SIZE 12

int exampleData;
char exampleArray[ARRAY_SIZE];
```

b. Compilar el proyecto creado (en caso de errores corregirlos) y ver los archivos generados en la carpeta del proyecto *Laboratorio.X*. Analizar particularmente las carpetas *build* y *dist*.
¿Qué tipos de archivos se encuentran en la carpeta *dist*?

c. Buscar las variables antes declaradas en el archivo *.map* dentro de la carpeta *dist*. ¿Qué tamaño ocupan en la memoria y en qué direcciones están alojadas?

d. Sustituir ARRAY_SIZE por 10 y ver qué ocurre con `exampleArray` en la memoria.

4. **Ejecución, Depuración**

   a. En las propiedades del proyecto verificar que el SNAP esté asignado como herramienta de hardware.

   b. Agregar un breakpoint en la línea de `Nop()`, y clickear en *Debug Project.*

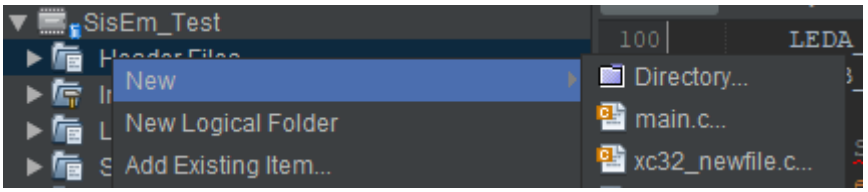
   c. Ver qué sucede, avanzar paso a paso, observar el contenido de las variables a través de Watch, y el código generado en *Execution Memory.*

Segunda Parte

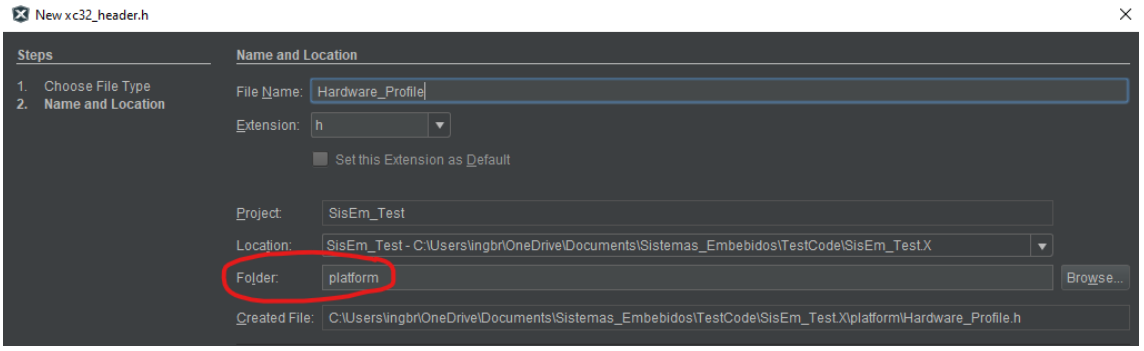
En esta parte del laboratorio se realizará el "Hola Mundo" de los Sistemas Embebidos: Encender un LED.
Luego se realizará el "parpadeo" de ambos LEDs de la placa, implementando una función de Delay (bloqueante).

1. **Creación de archivo de configuración del hardware**

   a. Para comenzar a trabajar sobre la programación modular, crear una carpeta lógica llamada ***platform*** en el directorio raíz del proyecto, y luego un archivo llamado ***HardwareProfile.h*** en dicha carpeta. Para ello, en el árbol de directorios del proyecto en el IDE, hacer click derecho sobre header files y crear un nuevo directorio y llamarlo platform:



   Luego, crear el archivo HardwareProfile.h, haciendo click derecho sobre Header Files, y creando un ***xc32_header.h***. Agregarlo dentro de la carpeta "física" platform:



   b. En el nuevo archivo, reemplazar, en las directivas ***#ifndef*** y ***#define***, la etiqueta *_EXAMPLE_FILE_NAME_H* por *_HARDWARE_PROFILE_H*. Discutir para que se utilizan estas directivas.

   c. Dado que ambos LEDs, LEDA y LEDB, están conectados al PIC a sus pines RA7 y RB14 respectivamente, se tienen que configurar ambos pines como salidas. Para ello agregar los siguientes macros en ***HardwareProfile.h***

```
#define LEDA_SetDigitalOutput() (TRISAbits.TRISA7 = 0)
```

```
#define LEDB_SetDigitalOutput() (TRISBbits.TRISB14 = 0)
```

d. Analizar la estructura de datos TRISAbits y ver otra forma de utilizarla.
¿Cómo sería la otra forma de utilizar el union?

e. Basados en los macros anteriores, y sabiendo que los puertos de salida pueden ser controlados mediante el registro *LAT*, crear/definir los siguientes macros:

```
#define LEDX_SetHigh() // Pone en nivel alto
#define LEDX_SetLow()  // Pone en nivel bajo
#define LEDX_Toggle()  // Invierte de nivel
```

Reemplazando X por A y B para ambos LEDs.

2. **Creación de una función de delay**

a. Crear una nueva carpeta en el directorio raíz del proyecto llamada ***utils***, y dentro de ella se van a crear dos archivos: ***utils.c*** y ***utils.h***.
Dentro de utils.c, incluir a ***utils.h,*** y crear la siguiente función:

```
void UT_delay( void ){
    int n=UT_DELAY_CYCLES;

    while(n>0){
       n--;
    }
}
```

Mientras que en utils.h se define la constante **UT_DELAY_CYCLES** utilizada en ***utils.c*** como:

```
#define UT_DELAY_CYCLES 500
```

Y el encabezado de la función:

```
void UT_delay( void );
```

3. **Implementación del parpadeo de LEDs**

a. Incluyendo a ***HardwareProfile.h*** y ***utils.h***, modificar el ***main.c*** para que en la placa queden ambos leds parpadeando indefinidamente.