

Laboratorio 2

MCC, Puertos de entrada, Interrupciones, Timers

Objetivo

Ya con algunas nociones básicas del IDE y la placa de desarrollo del curso, se seguirá avanzando en el control del hardware, implementación de macros, y mejorando ciertas implementaciones del laboratorio anterior haciendo uso de interrupciones.

Se comenzará a utilizar la herramienta brindada por Microchip para configuración de registros de periféricos a través de una interfaz gráfica. Con esta herramienta, se configurarán puertos de entrada para el uso de botones y Timers para el control de tiempos.

El laboratorio se divide en dos partes, la primera para trabajo con el simulador, la segunda para trabajo con la placa del curso.

Evaluación y metodología de trabajo

Al igual que en el resto de los laboratorios y el proyecto final, se trabajará en grupos (siempre el mismo). Los docentes asistirán 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.
- E. Entrega en fecha de las soluciones propuestas.

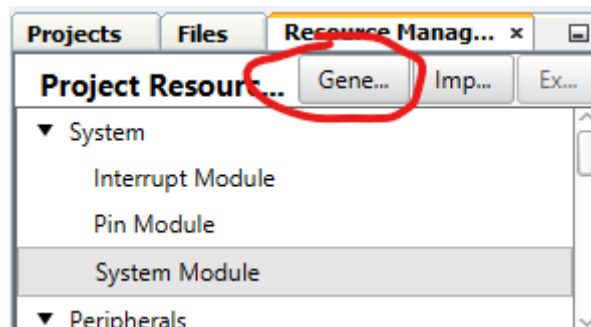
Ejercicios

Primera Parte

Integración del MCC (Microchip Code Configurator) e integración de los botones a través de la placa a la solución planteada en el Laboratorio 1..

1. Rehacer la configuración de los LEDs del laboratorio anterior a través del MCC.

- a. En el MPLAB X ir al menú *Tools-->Plugins-->Available Plugins* y descargar el Microchip Code Configurator (MCC)
- b. Abrir el proyecto del Laboratorio 1, y hacer click en el botón del MCC que se encuentra en la barra de herramientas . Esperar a que se abra el nuevo configurador, puede demorar varios segundos.
- c. Configurar *System Module* con los siguientes parámetros:
 - Clock: FRC Oscillator
 - FRC Postscaler 1:1
 - No se usa Secondary Oscillator
 - No se usa Reference Oscillator Output
 - Habilitar Clock Switching, Fail-Safe Monitor y PSOC Errata Workaround.
 - Se utiliza PGEC1 y PGED1 como señales de clock y datos para programación ICD.
 - No se utiliza el Watchdog Timer
- d. Abrir el *Pin Manager: Package View* para configurar los pines. Mediante click derecho se puede asignar a los pines, donde se encuentran los LEDs como General Purpose Input Output (GPIO) como salidas.
- e. Configurar el *Pin Module* agregando un nombre en el campo *Custom Name* para cada pin (por ejemplo LEDA).
- f. En el *Resource Manager* dar click en **Generate** y ver como queda el árbol del proyecto y el main.c. En este paso el MCC les va a marcar varios conflictos con el archivo main anterior. La recomendación es aceptar el main nuevo.



- g. Analizar los archivos generados en MCC Generated Files, tanto en Source Files como en Header Files. Particularmente, ver los archivos *pin_manager.c* y *.h...*
 ¿A qué les hace acordar del laboratorio anterior?
 ¿Sigue siendo necesario el HardwareProfile.h?
 - h. Probar que el laboratorio 1 sigue funcionando correctamente.
2. **Lectura de botones mediante polling**
- a. Utilizando las ventajas de la configuración con interfaz gráfica que proporciona el MCC, se configurará ambos botones de la placa del curso. Para ello, analizar el circuito esquemático de la placa del curso (y cualquier otro documento que sea

necesario) para determinar en qué puertos y pines están conectados (elementos S2 y S3 del esquemático).

¿Se deben configurar como puertos de entrada o salida?

- b. Para que la lectura de los botones funcione correctamente, el pin de entrada debe estar en algún estado bien definido, alto o bajo (1 o 0). Para ello se emplean configuraciones con resistencias de pull-up o pull-down, dependiendo del valor que se quiera tener para un botón sin presionar y como se halle el botón conectado en el circuito.

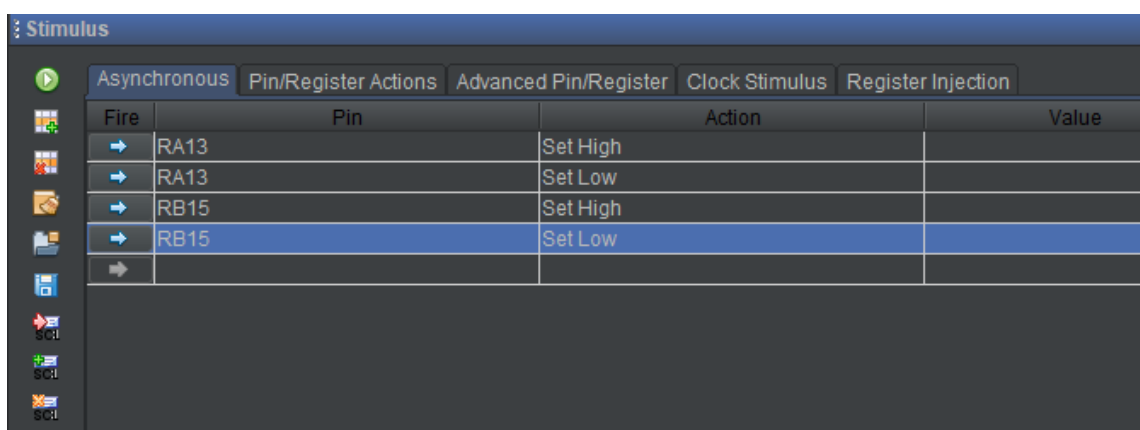
Analizando el circuito esquemático de la placa de desarrollo, definir si es necesaria una resistencia de pull-up o pull-down, en cada botón, para que los botones funcionen correctamente.

- c. Configurar en MCC la dirección de los pines, las resistencias de pull-up o pull-down (seteando los cuadros WPU o WPD respectivamente), y dar los nombres BTN1 y BTN 2 en el *Pin Module*, para luego generar el código del MCC.
- d. Analizar los macros y funciones creadas para el manejo de los botones.
- e. Utilizar los macros, creados por el MCC, en la función main de forma tal que, utilizando una arquitectura Round Robin, mientras se presiona un botón se enciende el led correspondiente (BTN1 con LEDA y BTN2 con LEDB) y cuando se suelta el botón el led se apaga.

Correr con el simulador.

El simulador ofrece una herramienta para incluir estímulos, con el cual se podrán simular los botones. La herramienta se llama Stimulus y puede ser encontrada en el menú Windows -> Simulator - Stimulus.

Una vez abierto, bajo la pestaña de estímulos asíncronos, vamos a agregar 4 líneas: 2 para subir y bajar el pin RA13 y 2 para subir y bajar el pin RB15, como puede verse en la siguiente imagen:



Utilizando el botón Fire, se puede introducir el estímulo. De esta forma se podrá debuggear si se detecta correctamente el botón presionado.

- f. Modificar el código anterior para que al presionar un botón y soltarlo, se invierta el estado del led.

Correr con el simulador.

Segunda Parte

Utilización de la placa: Integración de los botones a través de interrupciones y control de tiempo con timers.

1. Lectura de botones mediante interrupciones

- a. Para mejorar el tiempo de respuesta ante la presión de uno de los botones, se reemplazará la solución polling (Round Robin), por interrupciones (Round Robin con Interrupciones). Para ello se va a utilizar la funcionalidad que ofrecen los pines de entrada llamada Change Notice (CN). Dicha funcionalidad provee la posibilidad de generar interrupciones cuando hay un cambio de flanco en el pin seleccionado. Para ello, marcar en *Pin Module* la posibilidad de interrupción por flanco de subida en el campo IOC (Interrupt on Change).
- b. Antes de generar el nuevo código para manejo de interrupciones con MCC, es necesario configurar las prioridades y sub-prioridades, así como también habilitar las interrupciones del sistema.
Configurar Prioridad 2 para ambos botones, sub-prioridad 0 a través del *Interrupt Module* del MCC, y también habilitar las interrupciones del sistema.
- c. Generar el código del MCC.
¿Qué código fue agregado?
¿Cuáles registros son configurados para las interrupciones de los pines de los botones?
- d. En el archivo *pin_manager.c*, buscar las rutinas de atención interrupción (ISR) para cada botón.
Explicar qué es lo que hacen.
- e. Falta agregar el control de una variable booleana para indicar si el botón correspondiente fue presionado.
Para ello, definir un módulo *buttons.c* y *buttons.h* en la carpeta **platform**, que defina ambas variables y las funciones de interface para Setear, Resetear y leer el valor de dichas variables.
Luego, a través de las funciones *BTNX_SetInterruptHandler* definidas por el MCC, configurar las funciones de seteo para que sean llamadas al momento de la interrupción.
- f. Modificar la función *main* para lograr el manejo de los botones con los nuevos flags y comprobar funcionamiento.
Verificar con el debugger que efectivamente se está ingresando a las ISRs cuando se presiona alguno de los botones.

2. Utilización de Timer 2

- a. Configurar a través del MCC, el Timer 2 para que genere una interrupción cada 1 ms.
Analizar los macros y funciones creadas.
- b. Utilizando las funciones anteriores, modificar la función UT_delay del módulo **utils**, para que el tiempo de demora sea configurable y múltiplo de 1 ms. Ya que la nueva función cuenta milisegundos, renombrar dicha función como UT_delayms (Utilizar la herramienta **Refactor** del IDE).
- c. Modificar el main.c para que luego de presionar un botón se encienda el led correspondiente durante 2 segundos y después se apague.