Laboratorio 3

USB, RTC y LEDs RGB.

Objetivo

Haciendo uso del MCC, se configurará el USB para rápidamente poder enviar y recibir mensajes a través de un software terminal (por ej: Hercules que se encuentra en webasignatura). Se trabajará con el RTC (Real Time Clock) para tener la hora y fecha configurada en el sistema, y se podrán encender y apagar los LEDs RGB con algunos colores prefijados.
Este laboratorio está pensado para 2 semanas de trabajo.

Evaluación y metodología de trabajo

Los docentes asistirán a los estudiantes constantemente realizando sugerencias, devoluciones y aclarando conceptos cada vez que se necesite y sea pertinente.
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.

Ejercicios

Primera Parte

Integración del MCC (Microchip Code Configurator).

1. **Reconfigurar el clock del sistema para luego poder utilizar el USB.**

    a. Dado que el periférico del USB interno al microcontrolador necesita de una frecuencia del sistema bastante mayor a lo que se venía trabajando, se pasará a utilizar como señal de reloj un cristal externo de 8MHz, que será multiplicado y

dividido a través de los PLLs (Phase Locked Loops) del sistema, para obtener una señal de mayor frecuencia. Para ello:

Configurar *System Module* con los parámetros conocidos:

- Primary Oscillator de 8 MHz
- PLL multiplica por 12 y divide por 4
- No se usa Secondary Oscillator
- 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

2. **Agregar el periférico de control de USB**

   a. En el MCC, en System Module -> Registers, cambiar la configuración del bit FVBUSIO de "VBUS pin is controlled by USB module" a "VBUS pin is controlled by port function". Esto para configurar que el pin RB14 se comporte como pin I/O (comportamiento deseado dado de que está conectado al LED B) y no como parte del USB.

   b. Agregar para el USB, el recurso *MLA USB Device Lite.*

   c. Dejar la configuración por defecto (a efectos prácticos no se va a analizar los parámetros configurados por la complejidad de la librería de USB) y hacer click en **Generate.**
   Ver los archivos generados.

   d. Buscar la función USBDeviceTasks y analizar la documentación comentada de dicha función.
   ¿Es necesario utilizarla? ¿Por qué?
   También buscar la función CDCTxService y determinar si es necesario utilizarla, y cómo utilizarla en caso de ser necesaria.
   En función de ese ejemplo, y teniendo en cuenta las funciones *getsUSBUSART*, *putUSBUSART* y *putsUSBUSART*, implementar un programa que verifique que el USB esté configurado, y si recibe algo lo vuelve a enviar (Función Eco).
   Para lo anterior, notar que al conectar el USB con la placa del curso alimentada, se genera un puerto COM en el PC. Ese puerto COM puede ser abierto a través de un software terminal como lo es el Hercules que se encuentra en webasignatura.

3. **Agregar el periférico RTCC (Real Time Clock and Calendar)**

   a. Agregar el recurso del RTC desde el *Resource Manager.* y habilitar (en caso de que no lo esté) el RTC.
   NOTA: El RTC toma su señal de reloj desde el oscilador interno de baja potencia (LPRC), ya que la placa no cuenta con cristal para la utilización del Oscilador Secundario.

   b. **Generar** y ver los archivos nuevos generados.

c. Analizar la estructura ***struct tm*** y el tipo ***time_t*** de t*ime.h,* y las funciones *RTCC_TimeGet* y *RTCC_TimeSet* de *rtcc.c.*

Segunda Parte

La idea de esta parte es implementar al menos dos tareas en máquinas de estados, que trabajen "en paralelo" para resolver problemas simples.

Se deberá escribir un programa que implemente al menos 2 tareas utilizando máquinas de estado:

1. **Encendido de LED A**

   a. Modificar la función UT_delayms del módulo utils, para que sea "no bloqueante". Se sugiere definir un nuevo tipo de dato ut_tmrDelay_t que sea de la siguiente forma:

```
typedef struct
{
  uint32_t startValue;
  UT_TMR_DELAY_STATE state;
}ut_tmrDelay_t;
```

   Donde TMR_DELAY_STATE se define como:

```
typedef enum
{
    UT_TMR_DELAY_INIT,
    UT_TMR_DELAY_WAIT
}UT_TMR_DELAY_STATE;
```

   y modificar el encabezado de UT_delayms a

```
bool UT_delayms(ut_tmrDelay_t* p_timer, uint32_t p_ms).
```

   Esta función recibe como parámetro el puntero al timer a utilizar y la cantidad de ms que se quiere esperar; y va a ejecutarse según la variable de estado del parámetro. Debe retornar **false** mientras el tiempo no haya transcurrido y **true** una vez se cumpla el tiempo.

   b. Con el delay implementado, definir una tarea sencilla que luego se llamará en la función main, que estará constantemente encendiendo y apagando el led rojo (LED A). Lo hará de tal forma que esté 400ms encendido y 800ms apagado. Realizarla utilizando máquinas de estado.

2. **Interfaz de usuario**

a. La segunda tarea se encargará de implementar la interfaz de usuario de la placa, mostrando y recibiendo opciones a través del USB utilizando un software terminal. Esta interfaz debe permitir:
   - Fijar la fecha y hora del reloj de tiempo real (RTC) del PIC, validando el ingreso de datos (por ejemplo, no puede aceptar mes 13)
   - Encender/Apagar un led particular de un color fijo.
   - Consultar el estado y fecha y hora del último led que sufrió una modificación.

Sugerencias

Para almacenar el último cambio realizado a un led, se sugiere definir una estructura con la información, por ejemplo:

```
typedef struct
{
    uint8_t led;
    uint8_t color;
    uint32_t time;
} app_register_t;
```

- "led" indica cuál fue el led modificado (de 1 a 8).
- “color” indica el color a encender en el led:
  - 0 es blanco
  - 1 es rojo
  - 2 es azul
  - 3 es verde
  - 4 es negro (Apagado)
- "time" es el momento en el que se modificó el estado del led.

Para trabajar con el reloj de tiempo real, y convertir la variable uint32_t de tiempo en horas, minutos, segundos, etc, y viceversa, se deben utilizar las funciones *mktime* y *gmtime.*

Para controlar los LEDs RGB, descargar de webasignatura los códigos fuente a incluir dentro de la carpeta platform. Hay que configurar el pin de control de los LEDs a través del MCC con el mismo nombre que figura en el módulo WS2812, esto es LED_CTRL, de lo contrario el módulo no compila.
Se sugiere generar un array de *ws2812_t* de 8 elementos (uno por LED) y utilizar la función *`WS2812_send`* para encender cada uno.
También se sugiere implementar una función para setear los colores R, G y B de un led.