Sistemas Embebidos
Laboratorio 3

Felipe Mestre
Humberto Garcilazo
Leandro Shiromizu

Parte uno

2) c - Los archivos generados son:

- clock.h : Tiene implementaciones para las APIs de todos los módulos seleccionados. Inicializa el oscilador con la configuración especificada.
- exceptions.h : Tiene una función que es un general exception handler
- interrupt_manager.h : Aquí están las funciones que inicializan, activan y desactivan las interrupciones globales.
- mcc.h: No se usa
- pin_manager.h: El archivo configura la dirección de los pines, el estado inicial y la configuracion analogica.
- system.h : Tiene una función que inicializa el sistema con la configuración dada y dos más que se encargan de habilitar y deshabilitar la escritura en algunos registros.
- watchdog.h : Tiene implementaciones para habilitar y deshabilitar el wdt. Tambien tiene una función que pone a cero su tiempo.
- Dentro de la carpeta usb
    - usb.h : Tiene algunas definiciones principales de la librería de usb. Es el archivo núcleo.
    - usb_ch9.h : Define algunos tipos asociados con el chapter 9 del usb.
    - usb_common.h : Define tipos asociados con el usb host y usb device stacks pero no definidos en la especificación usb.
    - usb_device.h : Define tipos y APIs asociadas con el USB device stack
    - usb_device_cdc.h : Define algunos valores y funciones.
    - usb_device_config.h : Nos permite configurar algunas opciones del usb. Velocidad, timeouts, el usb interrupt mode, entre otros.

- usb_device_local.h : En este archivo se desactivan algunos eventos que podrían hacer uso desmedido de la cpu y que son prescindibles por el usuario.
- usb_hal.h : Define algunos data types, rutinas del usb y constantes.
- usb_hal_pic32mm.h : Define funciones, constantes y macros útiles.

La mayoría de estos .h tiene su correspondiente .c con las implementaciones.

- **d.** La función USBDeviceTasks() se ejecuta primero en un bucle (polling), esta se encarga de algunas rutinas que tienen que ver con la recepción y transmisión por usb, en la función se detectan algunos eventos que pueden suceder.
  La función CDCTxService maneja las transacciones de dispositivo a host. Al igual que la función anterior se debe ejecutar una vez por iteración en un bucle.
  Por alguna extraña razón el USB funciona cuando usamos solo la funcion CDCTxService().
  <code>

```
void main(void)
{
    CDCTxService();;
    while(1)
    {
        USBDeviceTasks();
        if((USBGetDeviceState() \< CONFIGURED_STATE) ||
          (USBIsDeviceSuspended() == true))
        {
            //Either the device is not configured or we are suspended
            //  so we don't want to do execute any application code
            continue;   //go back to the top of the while loop
        }
        else
        {
            //Keep trying to send data to the PC as required


            //Run application code.
            UserApplication();
        }
    }
}
```

Para corroborar que la comunicación serial fue configurada y hacer un eco, se puede utilizar la siguiente función:

```
int main(void) {

    while (1){

        CDCTxService();
        if ((USBGetDeviceState() < CONFIGURED_STATE) ||
                (USBIsDeviceSuspended() == true)) {
            //Either the device is not configured or we are suspended
            //  so we don't want to do execute any application code
            continue; //go back to the top of the while loop
        } else {
            if (USBUSARTIsTxTrfReady()) {
                numBytes = getsUSBUSART(buffer, sizeof (buffer) - 2);
                if (numBytes > 0) {
                    putsUSBUSART(bienvenida);
                }
            }
        }
    }
    return 1;
}
```

Bienvenida es un menú de prueba que hicimos. Para un eco como se pide en lugar de bienvenida usamos buffer, que es array de entrada.

3) b - Se generó rtcc.h y rtcc.c. Estos archivos contienen los drivers para el funcionamiento del RTCC. Funciones para obtener el tiempo y hora, sincronizando con la PC, a su vez contiene funciones para convertir de Binario a Hexadecimal y viceversa.

c -

- El struct tm es un struct creado con el objetivo de almacenar todos los datos de una fecha y hora, en binario obviamente. Es utilizado en varias funciones con distintos objetivos.

- time_t es un unasigned long.

- RTCC_TimeGet es una función con el objetivo de obtener el tiempo y fecha de RTCDATE y RTCTIME, guardando estos datos en currentTime.

- RTCC_TimeSet es una función con el objetivo de establecer un tiempo y hora determinados en RTCDATE y RTCTIME, según un struct tm mencionado antes el cual se da como parámetro de la función.