Guia de Ejercitación

Programación de la EDU-CIAA en lenguaje C

Objetivos

- Estudiar el hardware de la EDU-CIAA-NXP.
- Analizar características de la arquitectura ARM Cortex M4 sobre el microcontrolador LPC4337.
- Desarrollar experiencia en la instalacióny uso del CIAA-IDE.

1. Manejo de Puertos I/O

Para la realización de todos los trabajos prácticos de este curso, se utilizará la librería LPCOpen para el acceso a los periféricos específicos del LPC4337.

Esta biblioteca, se incluye en el Firmware de la CIAA, en:
$...CIAA\backslash Firmware\backslash externals\backslash drivers\backslash cortexM4\backslash lpc43xx$

Es destacable, que para cambiar de procesador, esta bibioteca se debe también cambiar por la biblioteca correspondiente, por ejemplo, para trabajar con la CIAA_FSL (CIAA Freescale), es necesario cargar los drives que se encuentran en:
$...CIAA\backslash Firmware\backslash externals\backslash drivers\backslash cortexM4\backslash k60\_120$.

1.1. Puertos de Salida: LEDS

Para el manejo de puertos de entrada salida de uso general (GPIO), el archivo gpio_18xx_43xx.c, contiene todos los drivers para manejo del mismo. Ver documentación de LPCOpen.

Para configurar un GPIO, lo primero es llamar a la función `Chip_GPIO_Init(LPC_GPIO_T * pGPIO)`, a la hay que pasarle como parámetro la dirección base del periférico GPIO definida ya en chip_lpc 43xx.h como LPC_GPIO_PORT.

Luego hay que configurar la System Control Unit (SCU), para indicarle las características eléctricas de cada pin empleado y remapearlos como puertos GPIO. Hay que recordar que en este procesador, se puede elegir entre varias funciones disponibles para cada pin (ver Tabla 189 en la página 397 del User Manual):

```
   Chip_SCU_PinMux(2,0,MD_PUP,FUNC4); /* mapea P2_0 en GPIO5[0], LED0R y habilita el
pull up*/

   Chip_SCU_PinMux(2,1,MD_PUP,FUNC4); /* mapea P2_1 en GPIO5[1], LED0G y habilita el
pull up */
   Chip_SCU_PinMux(2,2,MD_PUP,FUNC4); /* mapea P2_2 en GPIO5[2], LED0B y habilita el
pull up */
   Chip_SCU_PinMux(2,10,MD_PUP,FUNC0); /* remapea P2_10 en GPIO0[14], LED1 y habilita
el pull up */
```

```
   Chip_SCU_PinMux(2,11,MD_PUP,FUNC0); /* remapea P2_11 en GPIO1[11], LED2 y habilita
el pull up */
   Chip_SCU_PinMux(2,12,MD_PUP,FUNC0); /* remapea P2_12 en GPIO1[12], LED3 y habilita
el pull up */
```

A continuación, se debe seleccionar el modo (entrada o salida) de cada pin con la función:

```
Chip_GPIO_SetDir(LPC_GPIO_T * pGPIO, uint8_t portNum,uitn32_t portValue,uint8_t out);
```

Para setear y resetear los pines, existen numerosas funciones, entre ellas:

```
Chip_GPIO_ClearValue();
Chip_GPIO_SetValue();
Chip_GPIO_SetPinOutLow();
Chip_GPIO_SetPinOutHigh();
Chip_GPIO_SetPortOutHigh();
Chip_GPIO_SetPinToggle();
Chip_GPIO_SetPortToggle();
```

A las que siempre hay que pasarles como parámetro la dirección base del periférico GPIO (LPC_GPIO_PORT), el número de puerto y el bit a modificar. En caso de las funciones que hacen referencia a un solo GPIO (contienen la palabra "Pin") se le indica el número de bit del puerto que se desea modificar.

En caso de funciones que acceden a todo el puerto (Port), se le pasa una máscara del tipo uint32_T con los bits a modificar en 1.

1.2. Puerto de Entrada: Pulsadores

Para trabajar con los puertos de salida disponibles en la placa se deben configurar las entradas digitales correspondientes a las teclas.

```
Chip_SCU_PinMux(1,0,MD_PUP|MD_EZI|MD_ZI,FUNC0);/* mapea P1_0 en GPIO 0[4], SW1 */
Chip_SCU_PinMux(1,0,MD_PUP|MD_EZI|MD_ZI,FUNC0);/* mapea P1_0 en GPIO 0[4], SW1 */
Chip_SCU_PinMux(1,1,MD_PUP|MD_EZI|MD_ZI,FUNC0);/* mapea P1_1 en GPIO 0[8], SW2 */
Chip_SCU_PinMux(1,2,MD_PUP|MD_EZI|MD_ZI,FUNC0);/* mapea P1_2 en GPIO 0[9], SW3 */
Chip_SCU_PinMux(1,6,MD_PUP|MD_EZI|MD_ZI,FUNC0);/* mapea P1_6 en GPIO 1[9], SW4 */
```

Además, habilita para cada pin el buffer de entrada y deshabilita el filtro de glitch (ver figura 41 del User Manual). Es interesante como re realiza esta acción en el driver de GPIO del Firmware de la CIAA: "ciaaDriverDio.c", que se encuentra en
Luego setear estos pines como entrada (`Chip_GPIO_SetDir`), y para leer, se pueden utilizar las funciones:

```
Chip_GPIO_ReadValue()
Chip_GPIO_ReadPortBit()
```

1.3. Consignas

a) Diseñe e implemente un firmware sobre la EDU-CIAA que haga parpadear el led amarillo con un periodo que permita visualizar el proceso. Emplear retardo por software.

b) Diseñe e implemente un firmware sobre la EDU-CIAA que haga parpadear un led con un periodo que permita visualizar el proceso. Mediante las cuatro teclas disponibles se debe poder seleccionar el led activo. Emplear retardo por software.

2. Manejo de Temporizadores e Interrupciones

En este trabajo práctico se puede utilizar el Timer de Interrupciones Repetitivas (pag. 1073 del User Manual) cuyas funciones están implementadas en el archivo *ritimer_18xx_43xx.c*, de la biblioteca LPCOpen.

También disponemos de una función de inicialización del temporizador:

```
Chip_RIT_Init(LPC_RITIMER_T* pRITimer);
```

A la misma hay que pasarle como parámetro la dirección base del periférico RIT definida ya en `chip_lpc 43xx.h` como LPC_RITIMER.

Se debe configurar el intervalo (en ms.) de interrupción empleando la función:

```
Chip_RIT_SetTimer(LPC_RITIMER_T* pRITimer, uint32_T intervalo);
```

Para escribir el código de servicio de la interrupción, se debe generar una función:

```
void NombreDeLaIRutinaDeServicio(void);
```

y enlazar ese nombre en el vector de interrupciones definido en `vector.c`.

Para habilitar la interrupción se utiliza la función:

`NVIC_EnableIRQ(IRQn_Type IRQn);` definida en `core_cm4.h`.

Para borrar el flag de interrupción pendiente, existe en `ritimer_18xx_43xx.c` una función para esto:

```
Chip_RIT_ClearInt();
```

2.1. Consignas

a) Diseñe e implemente un firmware sobre la EDU-CIAA que encienda de a un led por vez y de manera secuencial. El tiempo de encendido de cada led ser 250ms. Se deberá temporizar mediante interrupciones sin usar funciones de retardo por software.

b) Diseñe e implemente un firmware sobre la EDU-CIAA que haga parpadear un led con un periodo de 250 ms. El sistema debe permitir seleccionar uno de entre 4 de los leds disponibles empleando una tecla para cada led.

- Tec 1: Selecciona LED RGB (uno de los tres colores)
- Tec 2: Selecciona LED 1.
- Tec 3: Selecciona LED 2.
- Tec 4: Selecciona LED 3.

c) **Consigna 2.3**:
Incorpore al ejercicio anterior la funcionalidad de variar el periodo de parpadeo del led activo.

- Tec 1: Selecciona el LED a la izquierda del actual.
- Tec 2: Selecciona LED a la derecha del actual.
- Tec 3: Disminuye el periodo de parpadeo.
- Tec 4: Aumenta el periodo de parpadeo.

3. Generación de Señales Analógicas (D/A)

- **Consigna 3.1**: Diseñe e implemente un firmware sobre la EDU-CIAA que genera una señal tipo diente de sierra de periodo 100 ms y excursión de 0 a 3V.
- **Cosigna 3.2**: Incorpore al ejercicio anterior la funcoinalidad de variariar el periodo y la amplitud de la señal.

- Tec 1: Aumenta la amplitud de la señal.
- Tec 2: Disminuye la amplitu de la señal.
- Tec 3: Aumenta el periodo de la señal.
- Tec 4: disminuye el periodo de la señal.

4. Adquisición de datos

4.1. Conversión D-A

Para la generación de señales analógica a través del conversor D-A, se deben empolear todas las funciones presentadas hasta este momento: de GPIO, RITimer e IRQ. Adicionalemtne se deben incorporar las funciones de uso del conversor Digital a Analógico (pag 1350 del User Manual).

En primer lugar se debe configurar la System Control Unit (SCU):
Algunos pines soportan el multiplexado de funciones digitales y analógicas, sin embargo, todas las entradas y salidas analógicas del ADC y DAC se encuentran ruteadas, además, a pines de función analógica sin necesidad de multiplexado.

Es necesario entonces indicarle al microcontrolador que vamos a utilizar el conversor DA mediante la función:

```
Chip_SCU_DAC_Analog_Config();
```

Luego se podrán utilizar las funciones de LPOpen para el manejo del conversor DA, incorporadas en `dac_18xx_43xx.h`:

```
Chip_DAC_Init(LPC_DAC_T *pDAC);

Chip_DAC_UpdateValue((LPC_DAC_T *pDAC, uint32_t dac_value);
```

4.2. Conversión A-D

En edición...

4.3. Consignas

a) : Diseñe e implemente un firmware sobre la EDU-CIAA que permita adquirir una señal analógica de excursión entre 0 y 3.3V, presente en el pin XX. El sistema debe encender el led rojo se la señal toma su valor máximo y led verde si la señal toma su valor mínimo.

b) : Incorpore al ejercicio anterior la funcionalidad de variariar los umbrales máximo y mínimo.

- Tec 1: Aumenta el valor del umbral.
- Tec 2: Disminuye el valor del umbral.

5. Transmisión de datos adquiridos a través del Puerto Serie

5.1. Consignas

a) Diseñe e implemente un firmware sobre la EDU-CIAA que envíe por el puerto serie la cadena *Hola Mundo* a tasa de transferencia de 9600 baudios. La cadena debe enviarse cada vez que el usuario presiona la tecla 1.

b) Diseñe e implemente un firmware sobre la EDU-CIAA que lea el puerto serie a la espera del caracter ascii 'a'. En respuesta el sistema debe enviar por el mismo puerto la cadena "Hola Mundoz cambiar el estado del LED 2 de la placa.

c) Diseñe e implemente un firmware sobre la EDU-CIAA que lea el puerto serie a la espera del caracteres ascii. El sistema solo responderá a los carácteres 'a', 'r' y 'v' según el siguiente detalle:

- Caracter 'a': Se debe cambiar el estado del LED 1 y enviar por el puerto serie la cadena: *Hola Mundo*
- Caracter 'r': Se debe cambiar el estado del LED 2 y enviar por el puerto serie la cadena: *Hola Mundo*
- Caracter 'v': Se debe cambiar el estado del LED 3 y enviar por el puerto serie la cadena: *Hola Mundo*

d) Diseñe e implemente un firmware sobre la EDU-CIAA que envíe por el puerto serie, cada vez qu se presiona la tecla 1, el valor de un contador de 256 cuentas. El valor del contador debe modificarse mediante las teclas 4 y 5.

e) Diseñe e implemente un firmware sobre la EDU-CIAA que permita adquirir una señal analógica de excursión entre 0 y 3.3V, presente en el CH1. El sistema debe enviar por el puerto serie una cadena de caracteres con el valor en decimal del dato convertido.

6. Manejo de periféricos con POSIX

Repita las consignas 1.1, 1.2 y 2.1 usando las funciones POSIX definidas en el Firmware de la CIAA.