SISTEMAS BASADOS EN MICROPROCESADOR
PRÁCTICA 3. MÓDULO DE RELOJ

Grado en Ingeniería Electrónica Industrial (3er curso)
Curso 2023/2024

1 INTRODUCCIÓN

Todos los microcontroladores de la serie MSP430 tienen un sofisticado módulo de reloj que permite definir varios relojes con distintas velocidades y características. En esta práctica se va a profundizar en este módulo.

1.1 Objetivos

- Experimentar con el sistema de reloj del MSP430.
- Familiarizarse con la documentación técnica.
- Realizar un módulo de gestión de pulsadores y leds de usuario.
- Realizar un módulo de gestión del reloj.

2 ESTUDIO TEÓRICO

Busque y estudie el *datasheet* del MSP430FR6989, la hoja de datos del *Launchpad*, así como la documentación del *Clock System* (`cs.pdf`) en la información de examen de la página web de la asignatura. En ella se recogen los registros de control del módulo de reloj del FR6989. Indique cuáles son las condiciones de arranque en el reset de los siguientes elementos:

- Velocidad del DCO: _____
- Fuente y velocidad del MCLK: _____
- Fuente y velocidad del SMCLK: _____
- Fuente y velocidad del ACLK: _____
- Estado del oscilador LFXT: _____
- Estado del oscilador HFXT: _____
- Estado del oscilador VLO: _____
- Tolerancia de la velocidad del DCO respecto de la nominal: _____

Busque en la hoja de datos del FR6989 en qué puertos están los terminales del oscilador LFXT:

- Anote aquí los puertos: _____
- Anote la función por defecto de los mismos: _____

Busque en la hoja de datos del *Launchpad* en qué pines de los puertos de expansión están las señales de reloj siguientes[1]:

- MCLK: _____
- SMCLK: _____
- ACLK: _____

1. Los puertos de expansión son las tiras de pines situadas en los laterales del *Launchpad*. La caja de protección impide ver la serigrafía de los pines. Las tiras se llaman, de izquierda a derecha, J1, J3, J4 y J2, Cada una es de 10 pines, estando el 1 en la parte superior. Puede comprobar en el manual que el J1.1 es +3'3V, el J3.1 es +5V, el J4.1 es P2.7 y el J2.1 es GND.

3 ESTUDIO PRÁCTICO

3.1 Preparando el proyecto

Copie el proyecto de la práctica anterior y llámelo `P3.1`. Para ello, selecciónelo en el Explorador de Proyectos y pulse `Control-C` (copiar) y `Control-V` (pegar). Se va a reaprovechar el módulo creado en ella, `lcd.asm`, así como parte del programa principal en C.

3.2 Módulo de gestión de la E/S del Launchpad `iuLaunchPad.asm`

3.2.1 Creación del módulo

Seleccione el proyecto en el *Project Explorer* y cree un nuevo fichero para albergar el módulo pulsando sobre *File/New/File from template...*. Llame al fichero `iuLaunchpad.asm` y seleccione "*Ensamblador*" en "*Template*" (modelo)[2]. El nuevo fichero se abre automáticamente en el editor. Estudie el contenido:

- La directiva `.cdecls` permite incluir un fichero de cabecera de C y lo traduce a ensamblador. En nuestro caso carga las definiciones del MSP430.
- La directiva `.global` declara una serie de etiquetas que se van a referenciar desde fuera. En nuestro caso nombra las subrutinas que se van a definir en este módulo. Si en el futuro añade subrutinas a este módulo, tendrá que añadir aquí sus etiquetas si van a ser accedidas desde otro módulo.

Todo módulo debe tener un fichero de cabecera que recoja las declaraciones de tipos y los prototipos del mismo, para poder incluirlo en aquéllos que lo utilicen. Seleccione el proyecto en el *Project Explorer* y cree un nuevo fichero para albergar el módulo pulsando sobre *File/ New/Header File*. Llame al fichero `iuLaunchpad.h` y seleccione "*Default C header template*" en "*Template*".

Recuerde que por cada servicio (subrutina, función) que añada a un módulo y que quiera que sea accesible desde el exterior, deberá añadir su nombre en la directiva `.global` y su prototipo en el fichero de cabecera.

3.2.2 Inicialización del módulo `iuIni`

Inicializa los pulsadores y leds de usuario del *Launchpad*. Los puertos de los pulsadores serán entradas con resistencia de *pullup* y los leds como salida y apagados:

```
void iuIni (void);
```

3.2.3 Establecimiento del estado del led `iuLedSetx`

Apaga (si `estado`=0) o enciende (en caso contrario) el led correspondiente:

```
void iuLedSet1 (int estado);
void iuLedSet2 (int estado);
```

2. Si el modelo no aparece en la lista, añádalo a partir del fichero `TemplateEnsamblador.xml` (que puede encontrar en la web de la asignatura) pulsando en "*Configure...*" y después en "*Import...*". Seleccione el fichero descargado y pulse "*Abrir*".

3.2.4 Lectura del estado de un pulsador `iuSwGetx`

Devuelve 0 si el pulsador correspondiente NO está pulsado y un número distinto de 0 en caso contrario:

```
int iuSwGet1 (void);
int iuSwGet2 (void);
```

3.3 Módulo de gestión del reloj `cs.asm`

El módulo de reloj recogerá aquellas funciones relacionadas con los relojes del MSP430. Haga un nuevo módulo para albergarlas. Para ello seleccione el proyecto en el *Project Explorer* y pulse sobre *File/New/Source File...*. Llame al fichero `cs.asm` y seleccione "*Ensamblador*" en "*Template*". No olvide hacer también el `cs.h` e incluya en él los prototipos de las funciones de `cs.asm`. Recuerde también incluir las etiquetas de las funciones accesibles desde el exterior en la directiva `.globals` y los prototipos en `cs.h`.

3.3.1 Inicializa el oscilador de baja frecuencia `csIniLFXT`

```
void csIniLFXT (void);
```

Las tareas que realiza son:

- Configurar los pines de E/S para que tengan la función de los pines del oscilador.
- Habilitar el oscilador LFXT.
- Esperar hasta que el oscilador se estabilice[3]. Para ello debe borrar el flag `LFXTOFFG` de `CSCTL5` que indica error en el oscilador LFXT, así como el flag `OFIFG` del registro `SFRIFG1` que indica error en *un* oscilador. Hacer un bucle que repita este paso mientras `LFXTOFFG` siga activo.

3.3.2 Exporta relojes por los pines de E/S `csCLKOut`

```
void csCLKOut (void);
```

Configura los pines de E/S para que puedan salir las señales de reloj al exterior. Dependiendo el modelo de microcontrolador, puede que no todos los relojes sean exportables. Los pines de salida dependen del microntrolador. Consulte su estudio teórico para ver dichos pines.

3.3.3 Configura la velocidad del DCO `csConfDCO`

```
void csConfDCO (uint8_t mhz);
```

Configura la velocidad del DCO. `mhz` es un entero sin signo de 8 bits que codifica las distintas posibilidades de este reloj. Los valores permitidos son los siguientes (defina las constantes en mayúsculas en el fichero `cs.h` para mejorar la claridad de los programas):

- `CS_DCO1_00` = 0 (Velocidad 1'00 MHz).
- `CS_DCO2_67` = 1 (Velocidad 2'67 MHz).
- `CS_DCO3_50` = 2 (Velocidad 3'50 MHz).
- `CS_DCO4_00` = 3 (Velocidad 4'00 MHz).

3. Cuando se activa un oscilador de cuarzo, éste no oscila de forma estable hasta pasado un tiempo. El FR6989 dispone de un circuito que monitoriza cada uno de los osciladores externos y activa un flag en caso de que el mismo esté en fallo (oscilación no estable). Hay un flag para el oscilador de baja frecuencia (`LFXTOFFG`), otro para el de alta frecuencia (`HFXTOFFG`) y otro que se activa cuando alguno ha fallado (`OFIFG`).

- `CS_DCO5_33` = 4 (Velocidad 5'33 MHz).
- `CS_DCO7_00` = 5 (Velocidad 7'00 MHz).
- `CS_DCO8_00` = 6 (Velocidad 8'00 MHz).
- `CS_DCO16_00` = 7 (Velocidad 16'00 MHz).
- `CS_DCO21_00` = 8 (Velocidad 21'00 MHz).
- `CS_DCO24_00` = 9 (Velocidad 24'00 MHz).

Si el parámetro está fuera de rango, la función no hace nada.

3.3.4 Configura reloj `csConfClk`

```
int  csConfClk (uint8_t clk, uint8_t fuente, uint8_t divisor);
```

Configura el reloj `clk` con una `fuente` y un `divisor` determinados. Son números sin signo de 8 bits que codifican las distintas posibilidades. Defina las constantes en mayúsculas en el fichero `cs.h` para mejorar la claridad de los programas.

Relojes disponibles:

- `CS_MCLK` = 0.
- `CS_SMCLK` = 1.
- `CS_ACLK` = 2.

Fuentes disponibles:

- `CS_LFXT` = 0.
- `CS_VLO` = 1.
- `CS_LFMOD` = 2.
- `CS_DCO` = 3.
- `CS_MOD` = 4.
- `CS_HFXT` = 5.

Divisores disponibles:

- `CS_DIV1` = 0.
- `CS_DIV2` = 1.
- `CS_DIV4` = 2.
- `CS_DIV8` = 3.
- `CS_DIV16` = 4.
- `CS_DIV32` = 5.

Cuando se configura el `ACLK` sólo están disponibles las fuentes de baja velocidad `CS_LFXT`, `CS_VLO` y `CS_LFMOD`.

Devuelve 0 si todo va bien y -1 en caso de error.

Ejemplos:

```
csConfClk (CS_ACLK, CS_LFXT, CS_DIV1); //ACLK con fuente LFXT, divisor 1
csConfClk (CS_MCLK, CS_DCO, CS_DIV8); //MCLK con fuente DCO, divisor 8
csConfClk (CS_ACLK, CS_DCO, CS_DIV8); //No hace nada porque DCO no es válido
```

3.4 Programa principal en C

Para poder usar el módulo anterior desde C, debe incluir los ficheros de cabecera correspondientes:

```
#include "cs.h"
#include "iuLaunchpad.h"
```

Escriba el `main()` para que haga lo siguiente:

1. Muestre el mensaje "`Inicia`" en la pantalla.
2. Encienda el led1, inicialice el oscilador de baja frecuencia y apague el led1.
3. Espere a que el usuario pulse y suelte el pulsador 1. Encienda el led1.
4. Muestre el mensaje "`CLKout`" en la pantalla.
5. Exporte las señales de reloj que pueda al exterior.
6. Espere a que el usuario pulse y suelte el pulsador 1. Apague el led1.
7. Haga un bucle que cambie el divisor del ACLK cada vez que se pulse (y suelte) sw2, empezando por 1. Cuando pase de 32, empiece por 1 de nuevo. Se sale de este modo pulsando (y soltando) sw1. El led2 debe permanecer encendido mientras se tenga pulsado sw2 y el led1 mientras se tenga pulsado sw1. Cada vez que se cambie el divisor se mostrará un mensaje "`ACK/nn`" donde nn es el divisor.
8. Haga un bucle que cambie el divisor del MCLK cada vez que se pulse (y suelte) sw2, empezando por 1. Cuando pase de 32, empiece por 1 de nuevo. Se sale de este modo pulsando (y soltando) sw1. El led2 debe permanecer encendido mientras se tenga pulsado sw2 y el led1 mientras se tenga pulsado sw1. Cada vez que se cambie el divisor se mostrará un mensaje "`MCK/nn`" donde nn es el divisor.
9. Haga un bucle que cambie la velocidad del DCO cada vez que se pulse (y suelte) sw2, empezando por 1MHz. Cuando pase de 8MHz, empiece por 1MHz de nuevo. Se sale de este modo pulsando (y soltando) sw1. Asegúrese de poner al principio el divisor de MCLK a 1. El led2 debe permanecer encendido mientras se tenga pulsado sw2 y el led1 mientras se tenga pulsado sw1. Cada vez que se cambie la velocidad del DCO se mostrará un mensaje "`DCO nn`" donde nn son los dos primeros dígitos de la velocidad (10 para 1 MHz, 26 para 2'67MHz,...).
10. Vuelva al paso 7.

3.5 Midiendo las señales de reloj

Una vez que el programa funcione correctamente y con la ayuda del osciloscopio, realice las siguientes medidas:

- El led1 se enciende mientras dura la puesta en marcha del oscilador de baja frecuencia. Anote aquí el tiempo aproximado (puede medirlo con exactitud con la ayuda del osciloscopio): _____
- Conecte el canal 1 del osciloscopio a MCLK y el canal 2 a ACLK. No olvide conectar la masa del osciloscopio a GND en los puertos de expansión.
- Pulse sw1 y las señales deberían aparecer y el led1 encenderse. Anote las frecuencias de ambos relojes:
  - f_{MCLK}: _____
  - f_{ACLK}: _____
- Pulse sw1. El led1 debe apagarse.
- Conecte el canal 1 del osciloscopio en ACLK. Pulse sw2 para cambiar el divisor. Pulse sw1 para pasar al siguiente punto. Anote las frecuencias medidas:
  - DIV1: _____
  - DIV2: _____
  - DIV4: _____
  - DIV8: _____
  - DIV16: _____
  - DIV32: _____
- ¿Ha notado algo raro durante las medidas anteriores? Emita una hipótesis de la causa:_____.

- Conecte el canal 1 del osciloscopio en MCLK. Pulse sw2 para cambiar el divisor. Pulse sw1 para pasar al siguiente punto. Anote las frecuencias medidas:
  - DIV1: _____
  - DIV2: _____
  - DIV4: _____
  - DIV8: _____
  - DIV16: _____
  - DIV32: _____
- Siga con el canal 1 del osciloscopio en MCLK. Pulse sw2 para cambiar la velocidad del DCO. Pulse sw1 para pasar terminar. Anote las frecuencias medidas e indique si está dentro del margen de tolerancia:
  - 1’00 MHz: _____
  - 2’67 MHz: _____
  - 3’50 MHz: _____
  - 4’00 MHz: _____
  - 5’33 MHz: _____
  - 7’00 MHz: _____
  - 8’00 MHz: _____