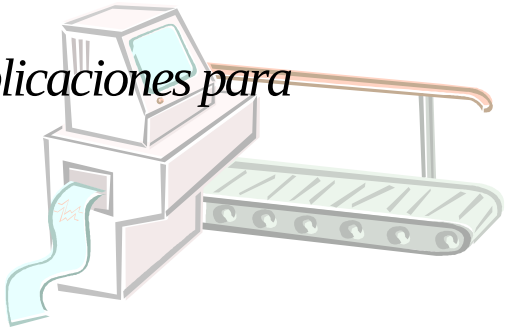
e-mail: eliseo.garcia@uah.es
Despacho: N-245
Departamento de Automática

Práctica 4

Interrupciones

Laboratorio de Desarrollo de Aplicaciones para Sistemas Industriales



Interrupciones

Introducción

En la presente practica se profundizará en el sistema de Gestión de Excepciones de los microprocesadores. Para eso, se trabajará con un conjunto de excepciones concreto, que son las interrupciones externas, que nos permitirán conocer cuestiones tanto de su activación como del mecanismo de atención a la excepción implementado para que puedan ser atendidas. Este mecanismo será común para todas las excepciones, por lo que las conclusiones que se extraigan podrán ser aplicadas a cualquier tipo de excepción.

Interrupciones externas del LPC1768

Las interrupciones externas permiten que un cambio del nivel de tensión en un pin de entrada del microcontrolador, provoque la ejecución de una rutina de atención a esa interrupción. Las interrupciones externas suelen utilizarse para que el procesador responda a eventos de elementos de hardware externos al chip.

Las interrupciones externas se pueden configurar activas por nivel (alto o bajo) o activas por flanco (bajada o subida). Una interrupción activa por nivel implica que permanecerá activa mientras el nivel de tensión en la patilla asociada se mantenga. Normalmente las interrupciones activas por nivel se utilizan para atender un evento de un dispositivo externo que se encarga de desactivar la interrupción cuando finaliza su atención.

El LPC1768 dispone de 4 entradas de interrupción externas específicas (/EINT0 ... /EINT3) situadas en los pines P2.10 … P2.13 como primera función alternativa, por lo que para seleccionarlas es necesario escribir un 01 en los bits adecuados del registro PINSEL4, tal como se muestra en la siguiente Tabla. La sintaxis que utiliza el entorno de desarrollo Keil uVision para acceder a dicho registro es LPC_PINCON->PINSEL4.

PINSEL4	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
1:0	P2.0	GPIO Port 2.0	PWM1.1	TXD1	Reserved	00
3:2	P2.1	GPIO Port 2.1	PWM1.2	RXD1	Reserved	00
5:4	P2.2	GPIO Port 2.2	PWM1.3	CTS1	Reserved [2]	00
7:6	P2.3	GPIO Port 2.3	PWM1.4	DCD1	Reserved [2]	00
9:8	P2.4	GPIO Port 2.4	PWM1.5	DSR1	Reserved [2]	00
11:10	P2.5	GPIO Port 2.5	PWM1.6	DTR1	Reserved [2]	00
13:12	P2.6	GPIO Port 2.6	PCAP1.0	RI1	Reserved [2]	00
15:14	P2.7	GPIO Port 2.7	RD2	RTS1	Reserved	00
17:16	P2.8	GPIO Port 2.8	TD2	TXD2	ENET_MDC	00
19:18	P2.9	GPIO Port 2.9	USB_CONNECT	RXD2	ENET_MDIO	00
21:20	P2.10	GPIO Port 2.10	$\overline{\text{EINT0}}$	NMI	Reserved	00
23:22	P2.11 [1]	GPIO Port 2.11	$\overline{\text{EINT1}}$	Reserved	I2STX_CLK	00
25:24	P2.12 [1]	GPIO Port 2.12	$\overline{\text{EINT2}}$	Reserved	I2STX_WS	00
27:26	P2.13 [1]	GPIO Port 2.13	$\overline{\text{EINT3}}$	Reserved	I2STX_SDA	00
31:28	-	Reserved	Reserved	Reserved	Reserved	0

Las interrupciones externas /EINT0 - /EINT3 tienen asociados los números de interrupción IRQ18 – IRQ21 (Tabla 50 del LPC17xx User Manual). Como el resto de las interrupciones, se configuran con el acceso a los registros generales del NVIC (6.5 del LPC17xx User Manual), pero adicionalmente dispone de un conjunto de registros propios para establecer los elementos particulares de este tipo de interrupciones.

a) Registros de Configuración de Interrupciones (comunes para toda Interrupción)

Para la configuración de cualquier interrupción en Cortex-M3, es necesario gestionar un conjunto de registros relacionado cada uno de ellos con un aspecto del tratamiento de Interrupciones.

- Habilitación y deshabilitación. Esta habilitación se puede realizar mediante un conjunto de funciones que nos entrega CMSIS para poder utilizar en nuestro programa.
  - void NVIC_EnableIRQ(IRQn_Type IRQn);
  - void NVIC_DisableIRQ(IRQn_Type IRQn);

- Configuración del nivel de prioridad mediante un conjunto de funciones que nos entrega CMSIS para poder utilizar en nuestro programa.
  - void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority);
  - uint32_t NVIC_GetPriority(IRQn_Type IRQn)

- Consulta o modificación del estado de interrupción. Se puede realizar mediante el uso de funciones de CMSIS, en concreto
  - uint32_t NVIC_GetPendingIRQ(IRQn_Type IRQn);
  - void NVIC_SetPendingIRQ(IRQn_Type IRQn);
  - void NVIC_ClearPendingIRQ(IRQn_Type IRQn);
  - uint32_t NVIC_GetActive(IRQn_Type IRQn);

b) Registros particulares de Configuración de Interrupciones Externas

El tipo de interrupción externa (activo por nivel o por flanco) y el nivel activo (activa por nivel alto o bajo, o flanco de bajada o subida) se configura con los registros EXTMODE y EXTPOLAR respectivamente (3.6 del LPC17xx User Manual). La sintaxis para el acceso a dichos registros en el entorno de desarrollo Keil uVision es LPC_SC->EXTMODE y LPC_SC->EXTPOLAR

Table 11. External Interrupt Mode register (EXTMODE - address 0x400F C148) bit description

Bit	Symbol	Value	Description	Reset value
0	EXTMODE0	0	Level-sensitivity is selected for $\overline{\text{EINT0}}$.	0
		1	$\overline{\text{EINT0}}$ is edge sensitive.	
1	EXTMODE1	0	Level-sensitivity is selected for $\overline{\text{EINT1}}$.	0
		1	$\overline{\text{EINT1}}$ is edge sensitive.	
2	EXTMODE2	0	Level-sensitivity is selected for $\overline{\text{EINT2}}$.	0
		1	$\overline{\text{EINT2}}$ is edge sensitive.	
3	EXTMODE3	0	Level-sensitivity is selected for $\overline{\text{EINT3}}$.	0
		1	$\overline{\text{EINT3}}$ is edge sensitive.	
31:4	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Bit	Symbol	Value	Description	Reset value
0	EXTPOLAR0	0	$\overline{\text{EINT0}}$ is low-active or falling-edge sensitive (depending on EXTMODE0).	0
		1	$\overline{\text{EINT0}}$ is high-active or rising-edge sensitive (depending on EXTMODE0).	
1	EXTPOLAR1	0	$\overline{\text{EINT1}}$ is low-active or falling-edge sensitive (depending on EXTMODE1).	0
		1	$\overline{\text{EINT1}}$ is high-active or rising-edge sensitive (depending on EXTMODE1).	
2	EXTPOLAR2	0	$\overline{\text{EINT2}}$ is low-active or falling-edge sensitive (depending on EXTMODE2).	0
		1	$\overline{\text{EINT2}}$ is high-active or rising-edge sensitive (depending on EXTMODE2).	

Table 12. External Interrupt Polarity register (EXTPOLAR - address 0x400F C14C) bit description

Bit	Symbol	Value	Description	Reset value
3	EXTPOLAR3	0	$\overline{\text{EINT3}}$ is low-active or falling-edge sensitive (depending on EXTMODE3).	0
		1	$\overline{\text{EINT3}}$ is high-active or rising-edge sensitive (depending on EXTMODE3).	
31:4	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Cuando la entrada está habilitada y se recibe un flanco o nivel activo, se pone a 1 el flag de interrupción asociado que se puede consultar leyendo el registro EXTINT y que, inicia el proceso de interrupción pasando su estado a PENDING, y siendo ejecutado o no dependiendo de si están o no habilitadas y tienen la prioridad adecuada.

Dentro de la rutina de atención a las interrupciones externas, se deben borrar los flags de interrupción que las provocaron. Esto se realiza escribiendo un 1 en el bit correspondiente del registro EXTINT. Esto es muy importante porque si no se hace, no se detectarán eventos futuros. Si la interrupción es activa por nivel, el flag de interrupción no se podrá borrar mientras permanezca activo el nivel, relanzando la interrupción si termina la rutina antes de que cambie el nivel de activación. La sintaxis para el acceso a dichos registros en el entorno de desarrollo Keil uVision es LPC_SC->EXTINT.

Bit	Symbol	Description	Reset value
0	EINT0	In level-sensitive mode, this bit is set if the EINT0 function is selected for its pin, and the pin is in its active state. In edge-sensitive mode, this bit is set if the EINT0 function is selected for its pin, and the selected edge occurs on the pin. This bit is cleared by writing a one to it, except in level sensitive mode when the pin is in its active state.[1]	0
1	EINT1	In level-sensitive mode, this bit is set if the EINT1 function is selected for its pin, and the pin is in its active state. In edge-sensitive mode, this bit is set if the EINT1 function is selected for its pin, and the selected edge occurs on the pin. This bit is cleared by writing a one to it, except in level sensitive mode when the pin is in its active state.[1]	0
2	EINT2	In level-sensitive mode, this bit is set if the EINT2 function is selected for its pin, and the pin is in its active state. In edge-sensitive mode, this bit is set if the EINT2 function is selected for its pin, and the selected edge occurs on the pin. This bit is cleared by writing a one to it, except in level sensitive mode when the pin is in its active state.[1]	0
3	EINT3	In level-sensitive mode, this bit is set if the EINT3 function is selected for its pin, and the pin is in its active state. In edge-sensitive mode, this bit is set if the EINT3 function is selected for its pin, and the selected edge occurs on the pin. This bit is cleared by writing a one to it, except in level sensitive mode when the pin is in its active state.[1]	0
31:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

En el manual de usuario se recomienda mantener siempre la interrupción desactivada cuando se modifique EXTMODE y borrar el flag asociado antes de habilitarla.

Practica Propuesta

En la presente practica se propone modificar la funcionalidad al programa desarrollado en la practica 3. Así, se añadirán una serie de interrupciones al código para implementar el siguiente comportamiento.

a) El diodo LD2 se encontrará parpadeando con un periodo de parpadeo inicial. Cuando se produzca una pulsación completa (pulsado y despulsado) del pulsador **KEY1**, se aumentará el periodo de parpadeo del diodo LD2, y cuando se produzca la siguiente pulsación completa, se decrementará el periodo de parpadeo al valor inicial. Este comportamiento se podrá repetir tantas veces como se quiera.

b) Similar comportamiento se debe establecer con el diodo LD3 y el pulsador KEY2 . El diodo LD3 se encontrará parpadeando con un periodo de parpadeo inicial. Cuando se produzca una pulsación completa (pulsado y despulsado) del pulsador **KEY2**, se aumentará el periodo de parpadeo del diodo LD3, y cuando se produzca la siguiente pulsación completa, se decrementará el periodo de parpadeo al valor inicial. Este comportamiento se podrá repetir tantas veces como se quiera.

c) La temporización se deberá realizar mediante la programación del Timer SYSTICK y no mediante la función delay() que se utilizaba en la práctica anterior. El valor inicial del periodo de parpadeo deber ser 0.1 segundos, y tras la primera pulsación se aumentará a 0.5 segundos

Realización de la Práctica

Una vez conocidos los requisitos que se plantean en esta práctica, se procederá a realizar las funciones necesarias para su correcto funcionamiento.

Paso 1. Configuración de las interrupciones

Realícese un proyecto que realice la configuración de las interrupciones que va a manejar el ejercicio solicitado. Para eso, créese en el entorno de desarrollo un proyecto con las siguientes funciones:

- Funcion *"uint8_t SetEintIrq(uint8_t extirq)"*. Establece que la funcionalidad de una determinada patilla será como interrupción externa EINT. La interrupción externa en concreto que se quiere configurar se especifica en el parámetro de entrada *extirq*, que podrá tener los valores 0, 1, 2 y 3 para configurar las interrupciones externas EINT0, EINT0, EINT0 y EINT0, respectivamente. La función retornará 1 si la configuración se realizó correctamente y 0 en caso contrario.

  Compruebese, mediante depuración y situando breakpoints, que utilizando esta función las patillas quedan configuradas correctamente. Para eso, se puede utilizar, una vez comenzada la depuración, el panel Peripherals→Pin Connect Block.

- Funcion *"uint8_t SetEintConfig(uint8_t extirq, uint8_t configuration)"*. Establece la configuración de una determinada interrupción externa. Admite dos parámetros de entrada
  - Como primer parámetro se define la interrupción externa en concreto que se quiere configurar utilizando el parámetro de entrada *extirq*. Este podrá tener los valores 0, 1, 2 y 3 para configurar las interrupciones externas EINT0, EINT0, EINT0 y EINT0, respectivamente.
  - Como segundo parámetro, *configuration,* se especifica la configuración concreta que se quiere asignar a la interrupción. Se admiten cuatro posibles valores, que son los que se muestran a continuación:
    - Valor '0' → Configuracion a nivel bajo
    - Valor '1' → Configuracion a nivel alto
    - Valor '2' → Configuracion en flanco de bajada
    - Valor '3' → Configuracion en flanco de subida

La función retornará 1 si la configuración se realizó correctamente y 0 en caso contrario.

Compruebese, mediante depuración y situando breakpoints, que utilizando esta función las patillas quedan configuradas correctamente. Para eso, se puede utilizar, una vez comenzada la depuración, el panel Peripherals→System Control Block → External Interrupts.

- Función *"uint8_t configIRQ()"*, que cuando se invoque, realice la configuración de las interrupciones, modificando los registros mencionados en esta guía de la práctica para conseguir la configuración solicitada.

```
#include "lpc17XX.h"

void configIRQ (){
…
…
}

int main(){
…
configIRQ();
while(1);
}
```

Compruébese que se consigue la configuración solicitada en la práctica. Para tal fin, obsérvese los resultados que se obtienen tras la simulación en el PC del proyecto creado, observando los valores de las pestañas:

- *Peripherals->Pin Connect Block*
- *Peripherals-> Core peripherals->Nested Vectored Interrupt Controller*
- *Peripherals->System Control Block->External Interrupts*
- *Peripherals-> Core peripherals->System Tick Timer.*

Paso 2. Creación de las Rutinas de Atención a la interrupción.

Añádase al proyecto las rutinas de atención a la interrupción que gestionan los pulsadores KEY1 (*void EINT1_IRQHandler*) y KEY2 (*void EINT2_IRQHandler*) que contendrán el código de respuesta a las interrupciones que va a manejar el ejercicio solicitado. Para eso, créese en el entorno de desarrollo un proyecto donde, añadiendo las funciones *"uint8_t SetEintIrq(uint8_t extirq)", uint8_t SetEintConfig(uint8_t extirq, uint8_t configuration)* y *void configIRQ()"* obtenidas en el paso anterior, se cree además la estructura de las rutinas que se deben invocar cuando se produzcan los eventos señalados en el enunciado.

```
#include "lpc17XX.h"

void EINT1_IRQHandler(){
…
}

void EINT2 IRQHandler(){
…
}


// Funcion que genera la temporizacion solicitada
…
void configIRQ (){
…
…
}
```

```
int main(){
…
configIRQ();
while(1);
}
```

Compruébese el funcionamiento de este proyecto. En concreto, compruébese que cuando se produzcan cada uno de los eventos reseñados en el enunciado, la ejecución del proyecto desemboca en la ejecución de la rutina que tiene asociada. Para tal fin, sitúe *breakpoints* en cada una de las funciones a analizar, y observe si el programa se detiene en ellos cuando se activa el evento asociado

Esta comprobación se debe realizar de dos formas:

- En primer lugar, utilizando el Simulador que nos ofrece nuestro entorno de programación Keil uVision, provocando en él los cambios en las patillas que generan los eventos. Utilícese para tal fin el menú Peripherals->GPIO Fast Interface
- En segundo lugar, configúrese el proyecto para realizar la ejecución en la placa, usando el entorno de desarrollo para monitorizar lo que en ella ocurra *(Project->Optios for Target*… y a continuación, pestaña *Debug* y opción *Use J-link/J-Trace Cortex*). Compruébese en este caso que la pulsación de los pulsadores de la placa desemboca en la ejecución de las rutinas asociadas al evento que producen.

Paso 3. Crear la rutina para la temporización.

Realícese un nuevo proyecto que incluya todas las funciones desarrolladas en los apartados anteriores y las funciones desarrolladas en la practica 2. Modifique/elimine asimismo todo lo relacionado con la temporización que se realizaba en la practica 2 (función delay y las llamadas a esta función). Inserte la rutina creada para generar la temporización. Este código debe permitir que el comportamiento de la aplicación en cuanto a tiempos sea el solicitado en la propuesta de la práctica, pero utilizando la interrupción de la temporización en lugar de la función delay(). Simule el proyecto en el entorno µVisión, para comprobar su correcto funcionamiento.

Compruébese también que la rutina que se encarga de la temporización se ejecuta periódicamente con el intervalo de tiempo que se necesita en nuestro programa.

Una vez comprobado el funcionamiento por separado de todas las rutinas de atención a las interrupciones solicitadas, compruébese en la placa miniDK2 que el programa realiza correctamente todas las acciones solicitadas en esta práctica.