102685 Arquitectura de Computadors i Perifèrics
PRÀCTICA Nº 5, CURS 2020/2021

ENTRADA/SORTIDA UTILITZANT EL MÈTODE D'INTERRUPCIONS

Objectius de la pràctica

En aquestes dues ultimes sessions de pràctiques tractarem els següents temes:
1) Com treballar amb els registres d'E/S dels dispositius perquè puguin interrompre al microprocessador Cortex M0.
2) Com dissenyar una RSI i com situar la seva adreça en la taula de vectors d'interrupcions.

Gestió d'interrupcions en la placa FRDM-KL25Z

Com la majoria dels microprocessadors actuals, el Cortex M0 que incorpora la placa **FRDM-KL25Z** suporta l'E/S gestionada per interrupcions. Aquest microprocessador ofereix la gestió de 48 fonts d'interrupció diferents, identificades de la 0 fins a la 47. Per la gestió de les RSI's (rutines de tractament d'interrupció) associades a aquestes interrupcions, el sistema utilitza una taula amb 48 vectors d'interrupció, un per cada font d'interrupció. Les primeres 16 interrupcions (de la 0 a la 15) estan reservades per a us intern del microprocessador (excepcions o trucades al sistema) i les últimes 32 (del 16 al 47) estan dedicades a les que generen els dispositius externs (com el Touch, Timers, DMA's, etc).
En el microprocessador Cortex M0, utilitzat en pràctiques, la taula de vectors d'interrupció se situa a partir de l'adreça 0 de l'espai d'adreçament. Com que cadascun dels seus vectors d'interrupció conté l'adreça d'una RSI, que ocupa 4 bytes, la mida d'aquesta taula de vectors d'interrupció és de 192 bytes.
Les 32 interrupcions dedicades a dispositius externs s'anomenen IRQ's i com comencen a partir de la interrupció 16 (la IRQ0 correspon a la font d'interrupció 16), tindran associats els vectors d'interrupció compresos entre el 16 i el 47.
Per a dissenyar programes que realitzen una operació E/S utilitzant aquestes IRQ's, s'han de seguir els següents 3 passos importants:

1) Permetre (o activar) la generació de l'IRQ pel part del dispositiu, procés conegut com *l'activació local* de la IRQ. Per a realitzar aquesta tasca, normalment s'ha d'activar (escriure un 1 o un 0) un bit dins d'un dels registres d'E/S associats al dispositiu. Per exemple i en el cas del Touch, si en el bit 6 (**TSIIEN**) del registre **TSI0_GENCS** s'ha escrit 1, llavors es genera una IRQ (la 26, associada al vector d'interrupció 42) cada cop que s'acaba de realitzar l'escaneig dels elèctrodes, això és, cada cop que s'activa el flag **EOSF** (bit 2 del mateix registre d'E/S).

2) Situar l'adreça de la RSI associada a l'IRQ en la posició adequada dins la taula de vectors d'interrupció.

3) Permetre que el microprocessador accepti l'IRQ, procés conegut com *l'activació global* de l'IRQ. Per realitzar la gestió de les IRQ's, el sistema disposa d'un mòdul de control

d'interrupcions anomenat **NVIC**. Una de les funcions d'aquest mòdul és controlar quines IRQ's seran acceptades pel microprocessador. Per realitzar aquesta tasca, el mòdul **NVIC** ofereix un conjunt de registres d'E/S de 32 bits, un per cada possible IRQ. Per realitzar aquestes dues sessions pràctiques, d'aquests registres només s'utilitzaran els dos següents:

- **ISER:** Escriure un 1 en el bit *i* d'aquest registre, permet que el microprocessador accepti l'*IRQi.* Escriure un 0 fa que el microprocessador no accepti l'*IRQi*.
- **ICPR:** Escriure un 1 en el bit *i* d'aquest registre, elimina l'estat pendent de l'*IRQi.* Escriure un 0 no l'elimina.

El procediment que se segueix per activar globalment una IRQ es: Primer esborrar el seu possible estat pendent (registre **ICPR**) i després permetre que el microprocessador l'accepti (registre **ISER**).

Es important destacar que abans de realitzar el pas 3, s'ha de comprovar que els passos 1 i 2 s'han realitzat correctament per evitar possibles problemes. Per exemple, si no s'ha situat l'adreça de la RSI en la posició correcta de la taula de vectors d'interrupció, el reconeixement de l'IRQ per part del microprocessador, provocarà la trucada a una RSI invalida (o a una adreça que no pertany a cap RSI).

Suport a la gestió d'interrupcions ofert per l'entorn CodeWarrior

A la pràctica anterior s'ha explicat el suport que dóna l'entorn CodeWarior, utilitzant el llenguatge C, per als registres d'E/S. Però aquest no es l'únic suport a l'E/S que ofereix CodeWarrior, també dóna suport a la gestió d'interrupcions. Com s'ha explicat en l'apartat anterior, per poder dissenyar programes que treballin en l'E/S gestionada per IRQ's, s'han de seguir 3 passos, els quals estan suportats per l'entorn CodeWarrior de la següent manera:

Pas 1, activació local de l'IRQ: Com s'ha vist en l'apartat anterior, aquesta activació consisteix en l'escriptura d'un bit en un registre d'E/S. Per tant, s'haurà de realitzar el mateix procediment que l'explicat a la pràctica anterior respecte com accedir als bits d'un registre d'E/S definit en el fitxer **MKL25Z4.h (***estructura DISPOSITIU_MemMapPtr i define DISPOSITIU_BASE_PTR)* **.**

Pas 2, situar l'adreça de la RSI en la taula de vectors d'interrupció: Per fer aquesta tasca, l'entorn CodeWarrior ofereix diverses estructures de dades. La primera és el *enum* **IRQInterruptIndex (**situat en el fitxer **MKL25Z4.h)** amb el qual es defineixen els noms de les 48 fonts d'interrupció reconegudes pel processador Cortex M0. La segona estructura de dades es la definició dels prototips predeterminats (nom i arguments) de les RSI associades a cada IRQ (el de les primeres 16 fonts d'interrupció és diferent). Aquests prototips es troben definits en el fitxer **kinetis_sysinit.c** i tenen la següent sintaxi**:**

void IdIRQ_IRQHandler() __attribute__ ((weak, alias("Default_Handler")));

A on ***IdIRQ*** es el nom de l'IRQ donat en l'enum **IRQInterruptIndex**, sense el prefixe **"INT_"**.
Per exemple, la IRQ associada al **PORTA** (IRQ 30, vector d'interrupció 46), rep el nom de **INT_PORTA** en el fitxer **MKL25Z4.h** i en el fitxer **kinetis_sysinit.c** té el prototipus de la seva RSI definit com: **void PORTA_IRQHandler() __attribute__ ((weak, alias("Default_Handler"))).**

L'entorn CodeWarrior situa automàticament les adreces d'aquestes RSI's predefinides en les seves posicions corresponents dins la taula de vectors d'interrupció. Aquest fet es pot comprovar mirant l'estructura de dades **void (* const InterruptVector[])(),** definida en el fitxer **kinetis_sysinit.c.** En aquesta estructura es pot observar que a cada vector d'interrupció de la taula (associat a una IRQ) es situa l'adreça de la seva RSI. Per exemple, en la posició 46 (començant per la zero) de **void (* const InterruptVector[])()** es situa l'adreça de la RSI associada a l'IRQ del **PORTA:**

```
void (* const InterruptVector[])(void) __attribute__ ((section(".vectortable"))) = {
.....
    //Posició 46
    PORTA_IRQHandler, /* Port A interrupt */
.....
}
```

Si no s'ha definit el cos de la RSI en el codi que es compilarà (per exemple en el main.c de les pràctiques), llavors, per defecte, es situa com a cos de la RSI el codi de la funció **Default_Handler** (això vol dir el **__attribute__ ((weak, alias ("Default_Handler"))** de la definició dels prototipus**).**
Per tant, si es vol codificar una RSI, només s'ha de definir el seu codi en el fitxer font (en el fitxer main.c de les pràctiques) de la següent manera:

```
void IdIRQ_IRQHandler() __attribute__((interrupt("IRQ")));
void IdIRQ_IRQHandler(void) {
    /* Definició del codi de la RSI */
}
```

Per exemple, per codificar la RSI associada al PORTA, només hem de definir el seu codi en el fitxer font de la següent manera:

```
void PORTA_IRQHandler() __attribute__((interrupt("IRQ")));
voidPORTA_IRQHandler (void) {
    /* Definició del codi de la RSI del PORTA */
}
```

Pas 3, activació global de l'IRQ: Aquesta activació consisteix en l'escriptura de bits en els registres d'E/S **ISER** i **IPCR** del mòdul **NVIC**. Per fer això, se segueix el mateix procediment que en la pràctica anterior, tenint en compte que aquests dos registres estan definits en l'estructura ***NVIC_MemMapPtr*** i que aquesta és accessible a través del apuntador ***NVIC_BASE_PTR*** (definides en el fitxer **MKL25Z4.h**) .
Per exemple, per activar globalment la interrupció del PORTA es podria utilitzar el següent codi:

```
NVIC_BASE_PTR→ISER = NVIC_BASE_PTR→ISER | 1<< (46-16);
```

Exercici

Realitzar la mateixa funció que a la pràctica anterior (la 4), que canvia el color del led del verd al vermell depenent si hi ha o no activitat al Touch, però utilitzant la IRQ associada a aquest touch. Com a condició important s'ha de tenir en compte que si el led ja està encès i depenent de l'activitat del touch, aquest led ha de seguir encès, llavors no s'ha de tornar a encendre per indicar aquest fet. Per exemple, si el led verd ja està encès i no hi ha activitat el touch, no l'hem de tornar a encendre per indicar aquesta falta activitat en aquest touch (ja que el led verd ha de continuar encès).
Per realitzar aquestes accions s'hauran de seguir els següents passos:

1) Inicialitzar el **Touch** i el **PORT B** com es va fer a la pràctica 4. També s'ha de fer l'activació local de la **IRQ** del **Touch** associada al final d'un escaneig, activant el bit **TSIIEN** del registre **TSI0_GENCS (**la informació d'aquest bit es pot trobar en el document: *KL25 Sub-Family Reference Manual)*.
2) Programar la **RSI associada a la IRQ del Touch** (finalització de l'escaneig) perquè com a la pràctica 4, il·lumini els leds verd o vermell depenent si hi ha activitat sobre el Touch: Si l'hi ha i el led vermell està apagat, llavors s'il·lumina només aquest led, si no hi ha activitat i el led verd està apagat, llavors només s'il·lumina aquest led verd. Important, abans de sortir d'aquesta RSI, s'ha de llançar un nou escaneig
3) Activar globalment la **IRQ's** del **Touch.**
4) Esperar indefinidament, però posant el microprocessador en baix consum entre execucions de les RSI's (utilitzar les instruccions assemblador que més convinguin: **WFI** o **WFE).**

IMPORTANT: Abans de **sortir de la RSI S'ha d'esborrar el flag** (**Touch: bit EOSF** del registre **GENCS)** que ha generat la interrupció associada a la RSI.

Documentació

- Al Campus Virtual: Dispositius i registres d'E/S de placa FRDM-KL25Z: *FRDM-KL25Z User's Manual i KL25 Sub-Family Reference Manual.*
- Operadors binaris amb C: http://www.zator.com/Cpp/E4_9_3.htm