Informe pràctica 3

Eduard Martín Graells
Joan Peracaula Prat

March 2018

1 Introducció

En aquesta pràctica ens acabarem de familiaritzar amb les interrupcions i aprendrem el que són els *Timer*'s i quina és la seva funció. Aquests dos recursos són dels més importants que disposa un microcontrolador, que ens permeten des de fer un bucle en temps "real" fins programar una alarma. Que precisament seran les funcionalitats que programarem en aquesta pràctica.

És a dir, mitjançant les interrupcions i els *Timer*'s, modificarem l'entrega anterior per tal que el retard entre els LEDs de la placa inferior sigui basant-se en un rellotge del microcontrolador i no en els cicles del rellotge intern del microcontrolador. A més, afegirem la funcionalitat de programar una alarma amb el *Joystick* i el botó *S2*. Per tal que l'alarma tingui més joc, també mostrarem per la pantalla LCD un rellotge i l'usuari podrà modificar l'hora d'aquest amb el *Joystick* i el botó *S1*.

2 Recursos utilitzats

Els recursos que hem utilitzat han estat: el *Joystick*, els botons *S1* i *S2*, la pantalla LCD, els LEDs RGB de la placa superior i els 7 LEDs de la placa inferior i els *Timer*'s TA0 i TA1, que els utilitzarem amb la font de rellotge ACLK(*Auxiliary Clock*.

Excepte els *Timer*'s, els recursos són els mateixos que els utilitzats en la pràctica anterior i s'utilitzen de la mateixa manera, ja que a l'afegir funcionalitats segons les diferents accions, el que farem serà modificar el *main*. No canviarem el mode en el que funcionen aquestes components hardware ni modificarem les interrupcions; així que per no fer més farragós el text, no els inclourem en els següents apartats a no ser que es treballin directament (p.e. en el diagrama), però que sàpiga el lector que pot trobar tota la informació sobre aquests recursos en la memòria de l'entrega anterior; de manera que no ens deixem res.

En particular sobre els *Timer*'s, dir que hem utilitzat el *TA0* i el *TA1*, que són de 16 bits.

3 Configuració dels recursos

La configuració dels *Timer*'s l'hem encapsulat en una funció d'inicialització per cadascun d'ells. Pel *Timer* TA0, que serà el que utilitzarem pel *delay* dels *LED*s del port 7, hem seleccionat la font de rellotge ACLK

mitjançant el registre TA0CTL i hem setejat el valor fins el que volem que compti a TA0CCR0, 32; aquest valor correspon als hertzs que equivalen a un milisegon (que és el temps que ens demanava l'enunciat), en funció de la freqüència a la que va el rellotge que utilitzem com a font (el ACLK va a 2^{15}Hz), per calcular el valor hem utilitzat una regla de tres. Un cop fet això, hem habilitat les interrupcions i hem netejat el vector d'interrupcions per seguretat amb els camps CCIE i CCIFG (negat) respectivament en el registre TA0CCTL0. Com que aquest rellotge no l'utilitzem immediatament al iniciar-lo, hem decidit engegar el *Timer* posteriorment (en concret en la funció *delay_t*).

Per l'altre *Timer*, TA1, els passos són els mateixos però pels seus registres corresponents, amb la diferència del valor que assignem a TA1CCR0 és 32000, ja que volem que compti fins a un segon (al utilitzar la mateixa font que TA0, clarament si un mili-segon són 32Hz, 1 segon són 32000Hz). En aquest *Timer*, al voler que treballi com un rellotge real, l'iniciem a la mateixa funció, amb el camp *MC__UP* al registre TA1CTL. A part d'engegar el timer, el mode de treball *MC__UP* configura el timer a comptar de forma continuada fins al valor del registre TA1CCR0 i després passar a zero.

També fer esment de la configuració de les interrupcions dels *Timer*'s. Com hem explicat més a dalt, l'habilitació a primer nivell de les interrupcions es fa en les mateixes funcions d'inicialització. En el següent nivell, el NVIC (*Nested Vectored Interrupt Controller*, ens hem d'assegurar que no quedi cap interrupció pendent i habilitar les interrupcions a aquest nivell; això ho hem fet amb els registres ICPR i ISER respectivament seguint les indicacions del *datasheet*: *BIT*8 pel TA0 i *BIT*A pel TA1. Per activar les interrupcions en l'últim nivell no hem hagut d'afegir res degut a que ja es cridava a la funció *__enable_interrupt*.

4 Funcions dels recursos

Hem implementat dues funcionalitats noves: un rellotge en temps real i una alarma programable.

De manera que l'usuari vegi aquestes dues funcionalitats el que fem es mostrar a la pantalla LCD ambdues coses, refrescant-la de manera que quan es modifiqui l'alarma o el rellotge real vagi avançant sempre es mostri el valor correcte. I quan 'soni" l'alarma escrivim per pantalla *ALARMA!!!*, per tal d'avisar l'usuari. Tant el valor del rellotge real com l'activació de l'alarma es fan utilitzant les rutines d'interrupció dels diferents *Timer*'s, on s'actualitza una variable global (*comptador* en TA0 i *seg* en TA1) que posteriorment tenim en compte en el *main* per modificar correctament l'hora i saber si hem d'activar l'alarma o no.

D'altra banda, donem a l'usuari les eines per modificar el rellotge real i l'alarma. Això ho fem aprofitant les rutines d'interrupcions del *Joystick* i els botons i el *switch* que teníem de la pràctica anterior que ens ajuden a separar i tractar les diferents accions que fa l'usuari, afegint alguns booleans de control (en realitat són variables enteres amb valors 0 i 1) i cridant funcions auxiliars. De manera que per modificar i deixar de modificar el rellotge o l'alarma l'usuari ha de prémer *S1* o *S2* respectivament. Un cop estigui en el mode editable, podrà avançar el rellotge de minut en minut movent el *Joystick* cap a dalt o endarrerint-lo movent el *Joystick* cap abaix; de la mateixa manera amb l'alarma.

5 Problemes

Apart dels diferents problemes que comporta començar a treballar amb un concepte nou, com són els *Timer*'s, també ens hem donat compte de certes limitacions que té el hardware.

Sobre els *Timer*'s, destacar que vam tardar una sessió completa a entendre com funcionaven i es relacionaven

els diferents registres associats, tot i tenir les diapositives de teoria i el manual, no acabàvem de veure com fer el que volíem. També ens vam quedar atrapats al intentar utilitzar el recurs RTC (Real Time Clock) en comptes d'un *Timer* per implementar el rellotge, fins que ens vam adonar que no podíem controlar interrupcions en funció dels segons i el professor ens va dir que utilitzéssim un *Timer* com en l'exercici anterior.

Pel que fa el tema *hardware* també vam veure que tot i en teoria controlar el rebot dels botons, al canviar l'estat d'un boolean al prémer un botó, no canviava ja que la senyal del botó rebotava. És a dir, polsant una sola vegada es generaven dues interrupcions i el valor de la variable booleana és negava dues vegada i quedava com el principi, provocant que no es realitzes el tractament que volíem. Era necessari, doncs, prémer el botó molt delicadament.

Una solució software per aquest últim problema, podria ser afegir un *delay* d'uns quants mili-segons dins de la rutina d'interrupció del component hardware que genera rebots, ja que dins de la ruta d'interrupció deshabilitem totes les interrupcions del port, i durant aquest temps per molt que es generin rebots, no saltaran interrupcions.

6 Conclusió

En aquesta pràctica hem estudiat i tractat amb *Timer*'s. Això ens ha servit per aprendre a programar la seva configuració, inicialització i freqüència de les interrupcions per tal d'obtenir els resultats que esperem. Hem hagut de seguir treballant amb interrupcions i la gestió d'aquestes.

En particular hem après que quan es dóna una interrupció es para el flux del programa principal per tractar-la. I per tal que aquest esdeveniment aïllat tingui una repercussió en el programa principal el que fem és actualitzar el valor d'una variable global que analitzem en el bucle infinit del *main*. Un altre aspecte important és que quan es dóna una interrupció cal desactivar la resta d'interrupcions d'aquell port perquè quan estiguem tractant-la no se'n doni cap altra. Al final del tractament cal reactivar les interrupcions d'aquell port. Actualitzar una variable global és una acció ràpida que ens permet tenir les interrupcions aturades una fracció de temps molt petita, permetent que es puguin generar altres interrupcions en aquest port de nou el més ràpid possible.

Pel que fa la metodologia, hem experimentat la importància d'utilitzar *breakpoints* i visualitzar els valors de variables i expressions durant el *debug*. Aquestes eines ens han permès agilitzar el procés de detecció i correcció d'errors.