

Creació de Funcions de Retard i un Rellotge en Temps Real mitjançant l'ús de *Timers* i *Interrupcions* - Informe

Quins són els objectius de la pràctica?

A l'anterior pràctica, vam veure com podíem utilitzar i perquè funcionaven els *ports GPIO* (entrades/sortides digitals), a més vam utilitzar les subrutines a *interrupcions* (les funcions anomenades *PORTX_IRQHandler(void)*). En aquesta pràctica l'objectiu és veure la configuració i el funcionament d'un altre tipus de perifèrics, els anomenats *Timers*. Alhora veurem com preparar les *interrupcions* (necessitem habilitar-les a 3 nivells) i com actuen de forma interna, és a dir com el processador interacciona amb el nostre programa. Tot això ho veurem però, a través de dos exercicis pràctics; en el primer generarem una base de temps mitjançant un tipus de rellotge (*ACLK* o *SMCLK*), i en el segon programarem una alarma.

Quins recursos farem servir?

Per a fer els exercicis partirem del codi de la pràctica 2. En l'anterior pràctica vam fer servir els diferents botons (direccions del *joystick* i botons *S1/S2*), els *leds*, tant de la placa del microcontrolador (els *leds RGB*) com els de la placa secundària acoblada (8 *leds* que emeten llum vermella). També vam fer servir la pantalla per a veure en quin estat ens trobàvem (segons la taula de l'enunciat de la pràctica). Els recursos que farem servir (a part dels *GPIO* anomenats) aquest cop seran els *Timers*, comptadors, que compten polsos de rellotge del sistema. Programarem també les seves *interrupcions*.

Configuració dels diferents recursos

Els diferents recursos que havíem utilitzat a l'anterior pràctica estaven ja "*configurats*", tant les inicialitzacions del *joystick*, la dels botons *S1/S2*, *leds* de la placa principal (*leds RGB*), i *leds* de la placa secundària, com les seves *interrupcions*. Tot això ho podem veure al codi de la pràctica, en les funcions *init_botons(void)* i *init_interrupciones()*, respectivament. Per al *Timer*, però, farem ús d'un altre mètode, ja que ens cal;

- Seleccionar una font de rellotge per al *Timer*. Farem servir el registre *TAxCTL* per fer aquesta tasca.
- Donar-li el mode (*UP*, *UP_DOWN*, *CONTINUOUS* o *STOP*)
- Establir el valor de la funció comparador del *Timer* en qüestió. Farem servir el registre *TAxCCRx*.

d) Habilitar les interrupcions per a conduir-les a les subrutines a través dels registres `TAXCCTLx`.

El mètode que he fet servir per a dur a terme les tres primeres necessitats és `init_Timers()`, es pot apreciar que en aquest mètode establim; (mitjançant el registre `TA0CTL` i `TA1CTL`) la font de rellotge que volem (d'entre 4) i el mode inicial (en el nostre cas seleccionarem `SMCLK`), amb els registres `TA0CCTL0` i `TA0CCTL1` habilitem les *interrupcions* a nivell de dispositiu, a través dels registres `TA0CCR0` i `TA1CCR0` posem el valor que ens interessa per a la funció comparador que té el *Timer* (en el nostre cas 24000, a causa de la freqüència de rellotge que l'`SMCLK` utilitza 24 MHz), i per últim he establert el mode `UP` per al *Timer 1* (El *Timer* que he fet servir per a l'exercici 2). D'altra banda, per a poder treballar amb els *Timers* cal habilitar les *interrupcions* a nivell del micro, mitjançant el `NVIC`, a través dels `ICPR[]` (per a assegurar-nos que no queda cap *interrupció* residual pendent per al port en qüestió) i dels `ISER[]`.

Com i per a que farem servir els recursos de la pràctica?

Un cop ho tenim tot configurat, passem a la realització dels exercicis pràctics. Com que hem agafat el codi de la pràctica anterior, hem conservat les diferents funcionalitats que ja hi havien implementades, és a dir els diferents estats segons si movíem el joystick a una direcció o un altre o si pressionàvem algun dels botons, tant els `S1` i `S2`, com el botó del joystick. També cal recordar que teníem altres funcions, com en el meu cas, l'anomenada `knightRiderEffect(uint8_t est, uint32_t ret)`, que movia d'esquerra a dreta o a l'inrevés, segons si movíem el joystick cap a la dreta o cap a l'esquerra respectivament. A més d'aquestes funcionalitats, ara hem de realitzar unes altres fent ús dels *Timers*.

En el primer dels exercicis havíem de reimplementar una de les funcionalitats de la pràctica anterior. Hem de tenir la possibilitat de regular la velocitat del parpadeig dels *leds* de la placa secundària en moure el joystick, establint uns límits, tant superior com inferior (*maxretard amb valor de 1050 i minretard amb valor 100*) per al retard del "*knight rider effect*". Doncs bé, en aquesta ocasió hem implementat una alternativa fent ús de la funció anomenada anteriorment, mitjançant l'ús de la font de rellotge `SMCLK`, tot generant una interrupció cada mil·lisegon (aprofitant que a `init_Timers()` havíem establert al registre `TA0CCR0` el valor de 24000), a la subrutina `TA0_0_IRQHandler(void)` controlem, segons si hem mogut el joystick cap a una banda un altre, el "*moviment*" dels *leds* tenint en compte també el retard que hi ha establert. Recordem que

aquest retard es modificarà (sumarem o restarem de 100 en 100) dins el *switch* de la funció principal *main*. En tot moment sabrem quin és el retard que hi tenim en el moment a través de la pantalla.

En el segon exercici pràctic se'ns demanava fer servir un altre *Timer* (en el meu cas A1) per a programar una alarma. En el meu cas, com he dit, he fet servir un altre *Timer* i a la subrutina *TA1_0_IRQHandler (void)* aprofitant que cada mil·lisegon es produïa una interrupció he fet que un cop transcorreguts 1000 mil·lisegons cada segon s'anés augmentant, de la mateixa manera per a minuts i segons. Tot això ho he aconseguit mitjançant variables globals inicialitzades a l'inici del codi. A continuació, he programat també amb variables globals, l'hora de l'alarma, la qual es podia modificar gràcies a les funcions *addTimeAlm()*, *substTimeAlm()* i *resetAlm()*. En la mateixa subrutina he controlat, que quan els enters per a hores, minuts i segons de l'alarma coincideixin amb els del clock (el qual recordem es va modificant gràcies a la subrutina) salti un missatge d'avís (que duri tan sols 5 segons) perquè l'usuari pugui veure que ha saltat l'alarma. En aquest exercici a més, podem ajustar l'hora que nosaltres vulguem si està en mode clock (ja que tenim tres funcions anàlogues a es de l'alarma per a modificar hh/mm/ss, d'unitat en unitat). Cal dir que addicionalment el mode clock i el mode alarma es poden canviar mitjançant el *joystick*, l'alarma es pot activar o desactivar amb el botó *S1* i tant l'hora de l'alarma com l'hora del rellotge es poden resetejar mitjançant el botó *S2*. Totes les funcionalitats i la lògica del programa les podrem veure més endavant en el diagrama de flux.

Problemes que han sorgit durant la pràctica

Per a dur a terme aquesta pràctica ha calgut molt de treball autònom i un treball de lectura del *Technical Reference Manual*, per tal d'adquirir els conceptes més teòrics. Per part meua a casa, al principi, abans de la primera sessió de laboratori no tenia ni idea de com utilitzar els registres del *Timer*, no vaig poder preparar-me amb antelació i no sabia com fer les inicialitzacions prèvies per a la configuració d'aquest recurs.

Després de l'última sessió de laboratori he modificat les funcionalitats requerides per al primer exercici, i no les he testejat, tot i que en principi haurien de funcionar de forma correcta.

A més d'això no puc dir que tingués cap problema en especial que no es resolgués fent *debugging* amb l'entorn de desenvolupament CCS, ja que l'enunciat era força clar i en cada exercici sabíem el que havíem de fer.

Conclusions

Tot i que al principi va ser difícil d'entendre, gràcies a l'ajuda del professorat i al suport tècnic que es pot trobar al Campus Virtual de l'assignatura, puc afirmar que durant aquesta pràctica he après i he resolt tots els dubtes que tenia sobre com funcionaven les *interrupcions* i en particular els *Timers*, a més d'apreciar els avantatges de treballar amb aquests.

Val a dir també que el codi es troba degudament comentat. A continuació es troba els diagrames de flux dels exercicis 1 i 2, els quals he decidit separar per tal que siguin més clars i entenedors. Les imatges dels diagrames de flux venen adjunts al .zip.

Diagrames de flux

Exercici 1)

Tal i com està a l'enunciat de la pràctica, en aquest exercici havíem d'extendre les funcionalitats de la pràctica anterior fent ús del *Timer*. Havíem d'incrementar/decrementar el retard per a que el "knight rider effect" tingués lloc de forma més ràpida o més lenta.



