Programació d'Arquitectures Encastades Pràctica 3
Creació de Funcions de Retard i un Rellotge en Temps Real mitjançant l'ús de *Timers* i *Interrupcions*
(2 sessions)

En aquesta pràctica farem servir dos dels recursos més importants dels que sol disposar qualsevol microcontrolador:

- ***Timers*** (Temporitzadors/Comptadors).
- ***Interrupcions*.**

Ja coneixem els perifèrics més senzills que són els ports GPIO (entrades/sortides digitals). Els ***Timers*** (Temporitzadors/Comptadors) són un altre tipus de perifèric. Son comptadors, que compten polsos de rellotge del sistema. Els podem configurar per fixar el límit fins al que volem que comptin, i executar alguna acció quan s'arribi a aquest límit. També poden generar interrupcions cada cop que es compleix un comptatge. El nostre microcontrolador té 4 timers (TAx), amb 5 comptadors (TAx.n) cadascun. Consulteu la documentació ***Technical Reference Manual***, capítol 17, especialment els apartats "17.2.3 Timer Mode Control" y "17.2.6 Timer_A Interrupts".

Les ***interrupcions*** permeten interaccions entre programa i usuari mitjançant els perifèrics (pantalla, teclat, UART, Timers, botons, etc...). En el nostre microcontrolador, podem decidir per a cada pin individualment (per exemple el pin P1.3) i per a cada perifèric (per exemple el Timer TA0.0) si volem activar o no que puguin generar interrupcions. Quan un perifèric sol·licita (o genera) una interrupció, el processador interromp el programa per atendre-la:

- desa l'estat del programa,
- identifica la interrupció,
- salta a la subrutina (Interrupt Service Routine, ISR) corresponent que hàgim programat,
- l'executa,
- recupera l'estat del programa,
- torna al punt on s'havia interromput y segueix amb el curs normal del programa.

Recordeu que les interrupcions són totalment asíncrones, és a dir que poden produir-se en qualsevol punt del programa a partir de la instrucció amb la que les habilitem. Això també vol dir que per fer servir les interrupcions les hem d'habilitar prèviament a la seva utilització.
Sovint, ens resultarà útil programar una ISR per modificar el valor d'alguna variable global nostra, de manera que al tornar d'executar la ISR, el curs del programa segueixi amb el nou valor.

L'habilitació/inhabilitació de les interrupcions es fa a 3 nivells.

- Primer nivell: cada font d'interrupcions té un bit d'habilitació (Enable). Per exemple, pel port GPIO x, tenim el registre PxIE que permet habilitar/inhabilitar cada pin com a font d'interrupció del port. En el cas d'un timer, ho configurarem amb el registre TaxCCTLn (on "x" és el numero del timer, i "n" és la interrupció que volem configurar). Estudieu la funció **init_botons()** proporcionada amb el programa de partida de la pràctica 2 per veure un exemple de com s'han de configurar aquestes interrupcions. Consulteu la documentació ***Technical Reference Manual***, apartat 10.2.7 i 17.2.6.
- Segon nivell: a nivell de perifèric, mitjançant l'anomenat "controlador NVIC". Cada perifèric hi té una entrada per inhabilitar/habilitar el conjunt de totes les seves interrupcions. Per defecte, el microcontrolador s'inicialitza amb les interrupcions de tots els perifèrics inhabilitades a nivell de l'NVIC. Estudieu la funció **init_interrupciones()** proporcionada amb el programa de partida de la pràctica 2 per veure un exemple de com s'ha de configurar aquest controlador. Consulteu la documentació esmentada en els comentaris d'aquesta mateixa funció: Datasheet ("Table

6-39. NVIC Interrupts", apartat "6.7.2 Device-Level User Interrupts", i Technical Reference Manual, apartat "2.4.3 NVIC Registers").

- Tercer nivell: a nivell global del microcontrolador. En qualsevol moment podem inhabilitar/habilitar temporalment totes les interrupcions que hàgim activat en els altres dos nivells, amb les funcions "**_disable_interrupt()**" i "**_enable_interrupt()**". Per defecte, el microcontrolador s'inicialitza amb les interrupcions inhabilitades a nivell global. Això ens permet configurar totes les interrupcions que vulguem a nivell individual, sense que se'ns dispari cap abans d'hora, i terminarem habilitant-les globalment quan ho tinguem tot a punt.

Important:

- Per a que funcioni una font d'interrupció, l'hem d'habilitar a tots 3 nivells.
- Cada cop que decidim activar una interrupció, no ens hem d'oblidar d'escriure la seva ISR. En cas contrari, al disparar-se una interrupció sense ISR, el programa saltaria automàticament a una ISR predefinida, **Default_Handler()**, que conté un bucle infinit, on es quedaria "atrapat" i controlat per evitar mals majors.
- A diferencia de les altres funcions "normals", a una ISR, NO la cridem nosaltres, ja que lògicament, no sabem mai quan es produiran les respectives interrupcions. Es el microcontrolador qui s'encarrega de desviar el curs del programa cap a la ISR pertinent.

Els nom de les rutines d'atenció a les interrupcions (ISRs) ens venen predefinits en la taula

```
/* Interrupt vector table.
```

dins l'arxiu

startup_msp432p401r_ccs.c

que se'ns genera automàticament al crear un projecte nou.
Les ISR no reben cap paràmetre ni tornen cap valor:

```
void Nom_predefinit_de_la_ISR(void){

      //instruccions…

}
```

Per practicar amb aquests recursos ho farem en dues parts:

1. Generant una base de temps de l'ordre d'una mil·lèsima de segon (o si es prefereix de 1/100 de segon).

2. Programant un rellotge horari en temps real.

1.- BASE DE TEMPS 1/1000 DE SEGON.

Ho farem partint de l'últim programa de la pràctica 2, on es demanava que els LEDs del ***port7*** s'il·luminessin segons unes seqüències que depenien del *joystick*. A més, podíem variar la velocitat en funció de que moguéssim el *joystick* cap amunt o cap avall.

Es tracta de fer el mateix però controlant amb més precisió els temps. Es demana:

- Programar un *timer* per a que generi una interrupció cada 1/1000 de segon. Disposem de dos rellotges de sistema que poden servir com a base per als timers, segons en vagi millor, configurats amb la funció "init_ucs_24MHz()" de la següent manera:
    - ACLK: freqüència 2^{15}Hz.
    - SMCLK: configurat a 24MHz.
- Al moure el *joystick* cap amunt o cap avall farem increments/decrements temporals basats en aquesta base de temps (bé de un en un , de 10 en 10... com creieu millor).
- Posant uns límits per la velocitat més lenta y la més ràpida de canvi dels LEDs.
- Mostrant al LCD el temps que hi ha en cada moment entre canvi de LEDs.

2.- RELLOTGE EN TEMPS REAL AMB ALARMA.

Es tracta ara d'afegir al nostre programa un rellotge amb les següents característiques:

- Mostrar a l'LCD l'hora amb el format: **hh:mm:ss**
- Poder ajustar/modificar l'hora amb els botons i/o *joystick*.
- Poder introduir una alarma per a que a una hora i minuts, que podem modificar, surti un missatge d'avís per l'LCD.

Per tal de poder mantenir els dos recursos, el rellotge i la base de temps del punt 1, es recomana fer el rellotge amb un *timer* diferent dels que hem fet servir abans.

Per fer aquesta pràctica, estudieu amb detall el que s'ha explicat a la classe de teoria i que teniu al campus virtual. I, evidentment, els capítols corresponents als *Timers* que feu servir al *Technical Reference Manual,* ja que haureu de configurar els registres adients.