CREACIÓ DE LES FUNCIONS BÀSIQUES DE COMUNICACIÓ I D'UNA LLIBRERIA DE FUNCIONS PER CONTROLAR EL ROBOT

En aquesta pràctica tenim dos objectius:

- Generar els recursos bàsics per comunicar la placa del microcontrolador amb el Robot.
  - Fer una funció per configurar les comunicacions amb els mòduls del Robot.
  - Fer una funció per Enviar "comandaments" als mòduls del Robot.
  - Fer una funció per Rebre informació dels mòduls del Robot.
- Fer una llibreria de funcions que permetin controlar el robot.

I la dividirem en dos parts:

- Versió emulada: no ens comunicarem amb els motors/sensor Dynamixel real sinó en una versió emulada en el PC. Per aconseguir-ho ens caldrà usar la UART del microcontrolador que va cap el PC. És recomana començar per aquesta donat que serà molt més senzill detectat errors i corregir-los.
- Versió real: ens comunicarem amb els motors/sensors Dynamixel real. Caldrà canviar a un altre UART que és la que realment està connectada a aquests dispositius. A diferencia del cas anterior, cal usar el pin "DIRECTION_PORT" per alternar entre la recepció i transmissió.

RECURSOS BÀSICS PER COMUNICAR-SE

Com ja sabem, la comunicació entre el microcontrolador i els mòduls del robot (AX-12: Motors; AX-S1: Sensor) és fa mitjançant un protocol sèrie asíncron. El recurs del processador que s'encarrega d'això es la USCI (Universal Serial Communication Interface) programada com a UART (Universal Asincronous Receiver Transmiter). El nostre microcontrolador disposa de 4 USCIs que poden funcionar com a UART.

VERSIÓ EMULADA

En el cas de la versió emulada els motors tindran IDs 1 i 2 i el sensor ID 3. L'emulació d'aquests s'executarà en una aplicació python que emularà el seu funcionament (vegeu la secció Emulador PAE del campus per instruccions detallades de la seva instal·lació i ús). Aquesta aplicació s'executarà en el ordinador on connecteu la placa d'avaluació i és comunicarà pel USB amb el microcontrolador MSP432. Per part del microcontrolador, la comunicació és realitzarà amb fins el *debugger* integrat en la placa d'avaluació, i aquest farà la translació entre UART i USB. En resum, per la part del codi que heu de desenvolupar vosaltres, el protocol de comunicació serà UART i haurà de funcionar a 115200 bps.

En el cas de la versió emulada farem servir l'anomenada "Backchannel UART", que és la UCA0. Està al port 1 (pins P1.2 = RX, P1.3 = TX).

VERSIÓ REAL

La USCI a usar és la UCA2, que està al port 3 (pins P3.2 = RX, P3.3 = TX). En el cas de la versió real els motors tindran IDs 1 i 2 i el sensor ID 100. Els identificadors poden ser uns altres depenent del robot. En tot cas disposen d'etiquetes que, excepte que s'hagin accidentalment reprogramat, mostren quin hauria de ser el ID correcte del dispositiu.

NOTA: En cap cas és pretén tenir dos codis diferents. L'objectiu és basa en fer ús de *defines* i/o variables per tal de poder seleccionar en el codi amb quina de les dos versions és vol treballar.

Necessitareu diverses funcions que heu d'introduir al vostre programa, que són les bàsiques per fer la comunicació:

A. Configuració del sistema de rellotge del microcontrolador UCS (Unified Clock System) per que la UART treballi a la freqüència requerida pels mòduls Dynamixel.
B. Configuració de la UART.
C. Funcions de transmissió i recepció d'una trama tal com la requereixen els mòduls Dynamixel.

CONFIGURACIÓ DE LA UCS

La UCS configura 3 fonts de rellotge:

- **MCLK**: Master Clock per a la CPU.
- **SMCLK**: Secondary Master Clock disponible pels perifèrics.
- **ACLK**: Auxiliary Clock disponible pels perifèrics.

Com ja ho hem estat fent en les pràctiques anteriors, utilitzarem la funció "init_ucs_24MHz()", disponible a la llibreria "lib_PAE2.h". Amb aquesta funció, l'ACLK ens queda a 2^{15}Hz, i l'SMCLK funciona a 24MHz.

Penseu que la configuració de la UCS repercuteix en el funcionament de tot el sistema, per tant es recomana que aquesta configuració es faci al començament de l'execució del programa (just després d'aturar el "Watchdog Timer" tal com hem anat fent en les pràctiques anteriors).

CONFIGURACIÓ DE LA UART

La configuració de la UART hauria d'ajustar-se al sistema de comunicació dels mòduls Dynamixel. El protocol que fan servir aquests mòduls és sèrie, asíncron de 8 bits, 1 bit de stop i sense paritat.

A més, per nosaltres, les comunicacions amb l'emulador de la pràctica funcionen amb un *baud rate* de 500 kbps. No obstant, per millorar la detecció dels bits en recepció, farem treballar la nostra UART amb sobremostreig (oversampling), agafant 16 mostres per cada bit transmès. Això vol dir que una UART que vulgui treballar a aquest *baud rate* ha d'estar alimentada amb un rellotge 16 vegades mes ràpid.

Tenint en compte tots aquest requeriments, estudieu la configuració de la UART a partir dels valors assignats als registres de configuració adients en la funció "`init_uart()`" proporcionada a classe de teoria, i expliqueu-lo a l'informe (recordeu que com a part de les eines de depuració, el CCS disposa d'una pestanya "Registers", que ens permet inspeccionar el continguts dels registres del microcontrolador).

Els mòduls Dynamixel són *half-duplex* al contrari de les UARTs habituals. Per la versió emulada és podria ignorar aquesta particularitat, però si que és imprescindible per la versió real. La recomanació és fer-ho sempre per evitar problemes. Per tant, hem de tenir en compte que cada cop que enviem un comandament a algun mòdul (TXD des del punt de vista del microcontrolador), aquest casi sempre torna una resposta (RXD des del punt de vista del microcontrolador), com a mínim amb un missatge informant de que ha rebut les dades sense error. És un punt important que haurem de tenir en compte als nostres programes, per evitar col·lisions a la línia *Data*.

Hem de tenir en compte que ara des del microcontrolador, per programa, hem de controlar el senyal "DIRECTION_PORT". A la nostra placa l'hem connectat al pin GPIO P3.0. Hem d'inicialitzar aquest senyal i gestionar-ho en funció de si volem enviar o rebre missatges a les funcions corresponents de transmetre o rebre. Per això farem un parell de funcions que controlin aquest pin i les farem servir just quan comencem una transmissió o una recepció. Finalment, aquesta línia "Data", l'haurem de mantenir sempre per defecte

en l'estat RXD. La canviarem a TXD només quan vulguem transmetre-li dades a algun mòdul, i la tornarem a RXD quan s'hagi acabat de transmetre l'últim bit de la trama a enviar.

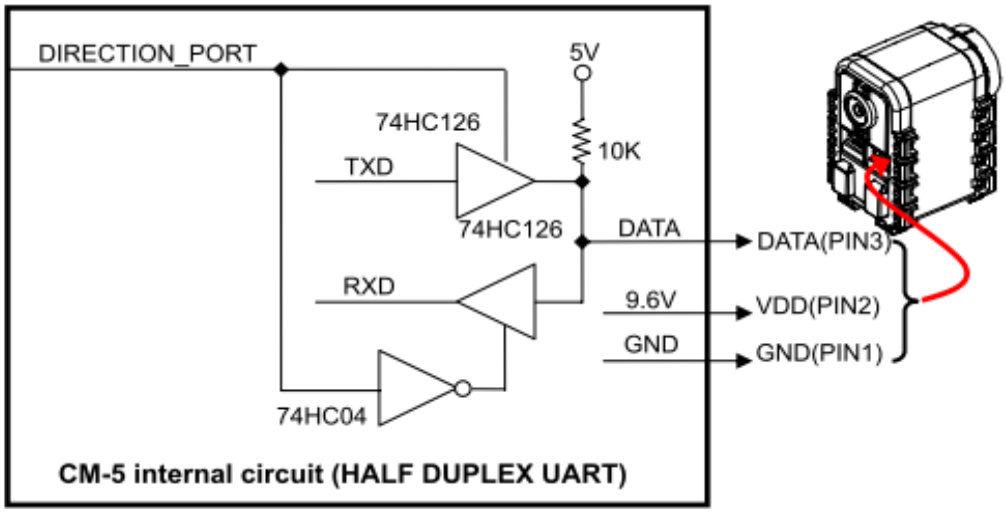

FIGURA 1 ADAPTACIÓ DE FULL DUPLEX A HALF DUPLEX

FUNCIÓ DE TRANSMISSIÓ D'UNA TRAMA TAL COM LA REQUEREIXEN ELS MÒDULS DYNAMIXEL

El microcontrolador es comunica amb els mòduls Dynamixel enviant i rebent paquets de dades, o trames. Hi ha dos tipus de paquets: "paquets d'instruccions" (enviat des del microcontrolador als mòduls Dynamixel, TXD) i "paquets d'estat" (enviat des dels mòduls Dynamixel al microcontrolador, RXD.)

Les funcions proporcionades a classe de teoria serveixen per transmetre una trama d'instruccions amb el format dels mòduls Dynamixel.

- A partir del que s'ha estudiat a la classe de teoria, i dels Manuals del AX-12 o el AX-S1, analitzeu les funcions de transmissió de dades (TXD) i estudieu el seu funcionament. Comenteu les instruccions que no ho estan i expliqueu tot això a l'informe de la pràctica.
- Feu una funció similar per tal de rebre dades (RXD) dels mòduls Dynamixel, utilitzant en aquest cas la capacitat de la USCI de generar interrupcions en recepció.

Hi ha un conjunt de adreces de memòria que en cas d'escriptura accidental poden deixar el mòdul inutilitzat. Per això, heu d'afegir un codi per tal que en cap cas és pugui escriure en les 6 (adreça 0x05 inclosa) primeres adreces.

NOTA: Penseu en encapsular adequadament en funcions els conjunts d'instruccions que repetiu sovint. Com a nota general, si esteu escrivint dos cops la mateixa seqüència d'operacions, agrupeu-ho en una funció.

FER UNA LLIBRERIA DE FUNCIONS PER CONTROLAR EL ROBOT

Es tracta de fer un conjunt de funcions per tal de controlar el robot. Per exemple, moure cap endavant, moure cap enrere... canviar velocitat, moure contínuament, moure uns determinats graus, capturar els diferents tipus de sensors... La vostra llibreria haurà de tenir com a **mínim** les funcions necessàries per configurar el robot en mode moviment continu, moure el robot en totes les direccions, pivotar sobre si mateix i llegir els tres sensors de distància.

Per fer aquesta llibreria feu servir les funcions de l'apartat anterior i la documentació dels mòduls Dynamixel AX-12 i AX-S1 que teniu al camps virtual.

NOTA: En el emulador no tenim disponible els dos modes de moviment del motor (moviment angular o moviment continu), sinó que només el mode continu. En aquest mode, referit com a *endless turn* en el manual, definim una velocitat i un sentit de gir pel motor i aquest el continuarà aplicant contínuament fins que s'indiqui un altre valor. Aquest és l'únic mode disponible pel emulador i el que ja ve configurat per defecte. És també el mode de funcionament recomanat per la pràctica i el projecte.

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, el capítol corresponent a la USCI que feu servir al Technical Reference Manual del nostre microcontrolador, ja que haureu de configurar els registres adients.