Introducció.

Ens trobem davant la penúltima pràctica d'aquest curs. Aquesta és la 'consolidació' de les pràctiques que heu realitzat al llarg d'aquests mesos. Per a la correcta realització d'aquesta pràctica cal haver 'acabat satisfactòriament' totes les anteriors.

1- El teclat matricial.

Una estructura molt comú emprada en la configuració de teclats (de ben segur aquells perifèrics més 'antics' i que han perdurat fins als nostres dies amb pocs canvis) és la 'matriu de tecles'. Les diferents tecles (polsadors elèctrics) es troben 'curtcircuitant' diferents files/columnes d'una disposició 'més o menys quadrada'. Amb això s'aconsegueix controlar 'n*n' tecles fent servir, tan sols, '2*n' senyals. L'algorisme bàsic per detectar les tecles premudes és el següent:

> Posar un 'zero lògic' en una sola columna.
> Llegir les files en paral·lel. Els 'zeros' llegits indiquen 'tecla premuda'.
> Procedir de la mateixa forma per totes les columnes restants.

Cal afegir que la disposició fila/columna així com altres elements electrònics presents depenen de la implementació concreta davant la que us trobeu. Consulteu la documentació necessària en cada cas.

Amb aquesta estructura es pot controlar un teclat amb 'rollover' de fins a dues tecles. Això vol dir que podem detectar fins a pulsacions simultànies de qualsevol parella de tecles (i així oferir 'combinacions' del tipus 'shift+tecla', 'control+tecla'...). En cas que es polsin simultàniament tres o més tecles les conclusions que en podem treure de ben segur que seran errònies i així hem d'actuar (controlar l'error).

A) Construïu el codi necessari per tal d'efectuar l'escombrat del teclat de la placa del laboratori. Aquest codi (conjunt de rutines, 'macros'...) serà 'la base' dels propers apartats. Podeu pensar en implementar una funció 'escombra(columna)' que retorna el valor de la lectura de totes les files amb la columna que passem com a paràmetre 'activada' (amb un zero). Amb aquesta funció podrem fer les primeres proves i saber exactament quines files/columnes corresponen a cadascuna de les tecles.

Suposant correctament implementada la funció descrita anteriorment i aprofitant les rutines de pràctiques anteriors podem executar el següent codi:

```
while (cert)
{
   EscriuCadena("  Columna 0");
   EscriuCaracter(escombra(0));
...
   EscriuCaracter(escombra(3));
   EscriuCaracter(0x0d);          /* un 'retorn de carro' */
}
```

Posar determinats bits d'un port i/o examinar- ne l'estat és una cosa que heu fet en pràctiques anteriors.

B) Tal i com us hem avançat tot això 'no funciona' si hi ha més de dues tecles premudes. Afegiu el codi necessari (filtrats, rutines, estructures...) per tal de 'validar'

pulsacions de una UNICA tecla. En qualsevol altre cas, s'ha de descartar la informació recollida. Podeu fer servir dues variables globals per aquesta tasca. Una de tipus 'bit' que indicarà si hi ha una pulsació vàlida i una altre de tipus 'char' que indica (en cas correcte) quina ha estat la tecla premuda . Òbviament s'ha d'exigir que l'escombrat del teclat es segueixi efectuant. D'aquesta manera els tractaments sobre el teclat seran del tipus:

```
.... ;dins un bucle que es repeteix indefinidament
escombra(i);
if (tecla_premuda) EscriuCaracter(tecla);
```

Per simplificar el codi de més alt nivell, a demés efectuarem una 'traducció' d'un 'número de tecla' a un 'símbol' a través d'una taula (al igual que els 'scan codes' i caràcters ASCII del PC). D'aquesta manera, les rutines de més baix nivell (escombra (), tecla...) retornaran un número entre zero i setze que farem servir per indexar una taula que conté la representació ASCII d'aquestes tecles. Amb això el codi anterior queda:

```
if (tecla_premuda) EscriuCaracter(taula_teclat[tecla]);
```

... i ens apareixen sobre el terminal els caràcters correctes.

C) Un altre aspecte a considerar amb els teclats és l'efecte produït pels 'rebots' de les tecles mecàniques. Encara que en el nostre cas (el teclat de la placa és 'de membrana') aquest és mínim, s'ha d'eliminar la incertesa de si una 'detecció de tecla' és correcta o si és produïda per un 'rebot'. La estratègia a seguir és:

Un cop detectada 'l'aparició d'un zero' en una lectura d'una fila, esperem un cert temps (10 ms) i tornem a llegir. Si la segona lectura és la mateixa la pulsació és vàlida, en cas contrari ho atribuïm a un 'rebot'.

Com que l'escombrat del teclat s'ha d'efectuar periòdicament i indefinida, variarem 'el punt de vista' i 'mirarem el present, coneixent el passat' en comptes de 'esperar esdeveniments futurs'. Amb això volem dir que dins de la rutina de servei a una interrupció periòdica (produïda cada 250 microsegons, p.e.) cada 10 milisegons prendrem una lectura del teclat (que anomenarem 'actual'). Aquesta lectura es compararà amb la presa en una lectura anterior i d'aquí s'extreu la informació vàlida (no era un rebot). Això ens obliga a una 'despesa' en espai (tantes variables 'anteriors' com files/columnes tinguem) però proporciona un tractament efectiu i 'elegant'.

Arribats a aquest punt, deixem a la vostra elecció les estructures i 'serveis' (notificació de 'make/break', notificació de 'rollover'...) que implementareu en el vostre codi.

2- Visualitzador LCD.

En aquest bloc heu de programar un visualitzador LCD de 'dues línies de setze caràcters'. Aquests visualitzadors han esdevingut molt 'populars' i la seva interfície s'ha convertit en un 'standart de facto'. Encara que hi ha molts fabricants diferents que produeixen gran varietat d'aquests mòduls, TOTS ells fan servir el mateix controlador (o una 'imitació') de manera que les connexions i el protocol de transferència és el mateix per a tots ells. Aquests visualitzadors es troben des de 'una línia de setze caràcters' fins a 'quatre de vint'; amb o sense 'retroil·luminació'. Tots ells tenen una interfície de '14 pins'.

A través d'aquesta interfície es poden llegir o escriure dos registres de vuit bits. Aquests dos registres són anomenats 'instrucció' (o comanda) i 'dades'. És a través d'aquests dos registres (i només a través d'ells) que s'efectua tota la programació. Cal esmentar que dins el mòdul LCD hi ha un microcontrolador (de vegades 'més complex' que el que el governa) que s'encarrega d'interpretar aquestes 'comandes' / 'dades' i genera tots els senyals necessaris (escombrats, desplaçaments, generació de 'bitmaps'...) per 'atacar' el visualitzador físic.

Per conèixer exactament els detalls de la connexió física (senyals, cronogrames...) consulteu la documentació adjunta (això forma part de la 'preparació prèvia', imprescindible en aquesta pràctica) i vegeu els exemples.

El principal 'problema' amb el que ens enfrontem en aquesta pràctica és que el visualitzador està muntat en configuració '4 bits'. Això no porta més complicació que tots els accessos es convertiran en 'dos de quatre bits' en comptes 'd'un de vuit'. Una altre implicació és que durant la fase d'inicialització NO es pot consultar 'còmodament' el flag de 'busy'; això ens obliga a enviar les diferents comandes d'inicialització 'a cegues' esperant un temps 'suficientment gran' entre comanda i comanda.

Nota: Si algú realitza aquesta pràctica en una placa que no sigui la del laboratori l'ha d'acabar tractant el visualitzador en mode '4 bits' (amb totes les implicacions que això comporta) encara que pugui efectuar proves en mode '8 bits'.

A) Feu un programa que us permeti enviar i llegir comandes i dades al visualitzador 'a voluntat'. Aquest programa us ha de permetre llegir/escriure qualsevol valor hexadecimal (de '0' a 'ff') tant al registre de dades com al de comanda. Podeu fer servir el teclat de la placa (i les rutines que heu fet en l'apartat anterior) o bé el port sèrie (també 'controlat' en pràctiques anteriors) amb la definició de 'trama' adient.

Exemple. Defineixo una estructura de trama de la següent forma:
'caràcter_d'inici'+ 'caràcter_comanda'+ 'caràcter_valor'+ 'caràcter_final'.

Faré servir '$' per inici, 'retorn' per final, els caràcters (entre '0' i '9' i de 'a' a 'f') pel valor i el 'conveni': 'i'- lectura de comanda, 'o'– escriptura comanda, 'k'- lectura dada i 'l'- escriptura dada per a les comandes de trama.

Amb aquesta 'definició' si envio la cadena '$of<CR>' (escrita al teclat del PC) per la línia sèrie el micro ha d'escriure el valor 'f' al registre de comandes. Això us permetrà assegurar quina és la seqüència correcta d'inicialització del visualitzador. En cas que solicitem una 'lectura' el que ha de fer el programa és enviar per la línia

sèrie el valor retornat adequadament (convertit a caràcter ASCII).
B) Un cop coneguda i verificada la inicialització del visualitzador i el seu tractament feu un programa del tipus 'terminal' (ja conegut). Aquest programa ha de mostrar al visualitzador tot allò que arriba per la línia sèrie i enviar les tecles premudes. Heu d'efectuar els tractaments necessaris per fer el 'scroll' vertical i horitzontal (desplaçar de manera adient la porció de text 'que es veu' a mida que van arribant línies de caràcters).

Construïu les funcions que creieu necessàries per a la futura utilització 'amb soltura' d'aquest visualitzador. Ex. 'esborra_displai()', 'escriu_cadena_displai(cadena)', 'escriu_caracter_displai(caracter)', 'amaga_cursor()', 'situa_cursor(posicio)'...

Recomanacions:

Feu una preparació prèvia 'a consciència'. Intentar abordar aquesta pràctica 'a primera vista' davant el teclat només us portarà a perdre una sessió sencera. Ordeneu i documenteu les pràctiques anteriors, feu un anàlisi de les explicacions i els codis dels articles que teniu a fotocòpies (part VI del 'Curso bàsico de microcontrolador' de la revista 'Elector', 'Commande de LCD par uC' i 'Gestion de LCD sur 4 bits' de la revista 'Radio Plans', informació del visualitzador 'LM032').

Planifiqueu amb molta cura el codi (escriviu prèviament els algorismes). Tingueu en compte la ubicació de les variables. Segons el model 'tiny' (que fem servir) TOTES les variables s'ubiquen en memòria RAM interna. Segur que amb 128 bytes (no tots 'disponibles') no hi ha prou lloc per ubicar 'un buffer' per la línia sèrie, una taula de 'bitmaps' de pulsacions de tecles (potser 8 bytes), la taula de 'pulsacions anteriors', la taula de 'traducció de scan codes' i totes les variables (locals i globals) del programa. Un problema d'aquests deriva en la generació d'un codi d'error del tipus 'Wrong segment size or segment overlap' a l'hora d'enllaçar. Forceu la ubicació d'aquestes dades (amb els prefix 'idata' 'xdata' en la definició) a la zona de memòria adient.