PRÁCTICAS DE *INTERFACES Y PERIFÉRICOS* (CURSO 2005-2006)

PRÁCTICA 1:

INTERFAZ CON ELEMENTOS PERIFÉRICOS SIMPLES, TIPO TODO/NADA

1. **Gobierno todo/nada de una carga** con una señal TTL:

   a) Mediante excitador discreto: dárlington BC517. Calcúlese la resistencia de base.

   b) Mediante excitador integrado: ULN2003.

   La carga será, en ambos casos:

   a) Un zumbador piezoeléctrico, alimentable desde 2 hasta 30 V, con corriente máxima de 20 mA a V_{max} (a efectos de cálculos, considérese una tensión nominal de 5 V y una impedancia de 850Ω aproximadamente).

   b) Un relé electromecánico, con tensión nominal de la bobina 12V y 210 Ω. Prepárense los cálculos en previsión de que se aplique a la bobina una tensión de 12 V cc.

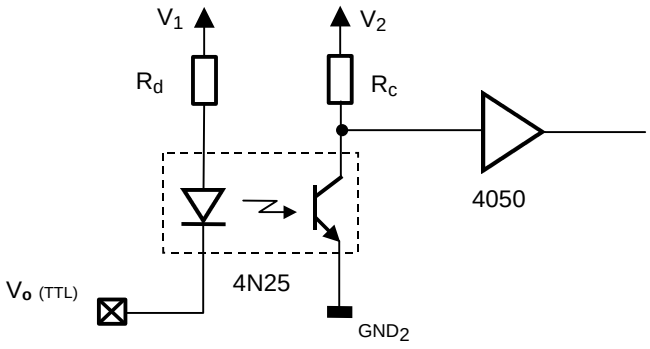
   Efectúense los cálculos oportunos, y extráiganse las consecuencias oportunas sobre el uso de excitadores discretos e integrados. En el primero de los casos contrástese los valores reales de funcionamiento con respecto a los teóricamente considerados, comentando las conclusiones que se hayan extraído.

2. **Transmisión ópticamente aislada**. Se trata de realizar el enlace físico, de manera aislada ópticamente, entre un emisor tipo TTL y un receptor tipo CMOS (tipo 4050) con V_{dd} = 12V.

   Realícense los cálculos, justificando las decisiones tomadas, para:

   a) Optoaislador **4N25**.

   b) Optoaislador **4N35**.



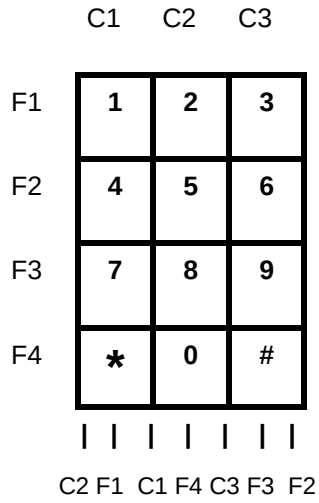
   Compárense los valores teóricos con los obtenidos en la realidad. Compruébese el comportamiento a medida que aumenta la frecuencia de transmisión. Obténgase experimentalmente los tiempos $\mathbf{t_{on}}$ y $\mathbf{t_{off}}$, así como la máxima frecuencia tolerable.

   Obsérvese la importancia de la componente capacitiva de la carga. Para ello colóquese en paralelo a élla un condensador (22 nF, por ejemplo) y compruébese el comportamiento a medida que aumenta la frecuencia de conmutación. Compárese ahora la máxima frecuencia tolerable con la obtenida anteriormente y extráiganse las conclusiones oportunas. (Sólo es necesario comprobar experimentalmente uno de los dos diseños)

PRÁCTICA 2:

INTERFAZ CON UN TECLADO MATRICIAL

Partiendo de un sistema basado en el microcontrolador **AT89S52**, impleméntese la interfaz con un teclado matricial de 3x4 teclas de tipo telefónico:



Por lo que respecta a la interfaz física, asóciese el valor lógico **0** a la tecla pulsada. Para ello añádasele al teclado los mínimos elementos necesarios, caso de que hagan falta. Conéctense las señales del teclado del siguiente modo: **F1**, **F2**, **F3** y **F4** a **P1.0**, **P1.1**, **P1.2** y **P1.3** respectivamente, y **C1**, **C2** y **C3** a **P1.4**, **P1.5** y **P1.6**.

Por lo que atañe a la interfaz lógica, gestiónese el teclado:

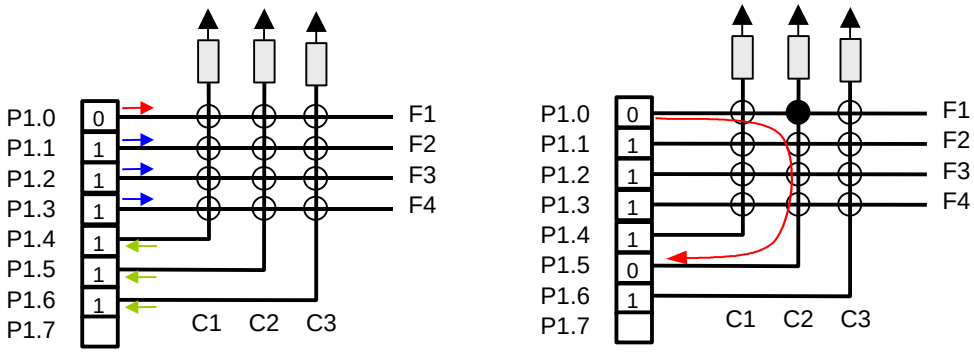
a) **Por sondeo**. La rutina implementada deberá contemplar el filtrado de los rebotes mecánicos que puedan producirse al pulsar una tecla. Para la temporización requerida por el filtrado puede o bien hacerse por programa (bucles de retardo) o por hardware (empleo de uno de los temporizadores). Para verificar el correcto funcionamiento, la rutina de gestión del teclado deberá sacar el código ASCII de la tecla pulsada por el puerto **P2.** Igualmente, cada vez que se pulse una tecla entonces se deberá actuar sobre la patilla **P3.7** cambiándola de estado.

b) **Por interrupciones**: Lo mismo, pero gestionado por interrupciones. Para ello impleméntese la capa física con una lógica oportuna que detecte la pulsación de una tecla y solicite una interrupción tipo **INT0**. Además de la rutina de servicio a la interrupción procedente del teclado, se deberá diseñar una rutina principal que genere por **P3.6** una señal periódica de frecuencia aproximadamente **1 hz** y ciclo de trabajo del **50%**. Para realizar la temporización empléense bucles software (iteración, por ejemplo, de instrucciones **NOP** el número de veces adecuado, sabiendo que esta instrucción consume un ciclo máquina, y que el sistema funcionará con un cristal de **12 Mhz**).

En función de la solución empleada en la capa física de la interfaz, realícese una crítica sobre su mayor o menor idoneidad (incluyendo el criterio fijado en lo concerniente a la capa lógica) desde un punto de vista puramente conceptual si se tiene en cuenta que se opta por una gestión del teclado por interrupciones.

Sugerencias y comentarios colaterales:

A) La implementación por programa más simple consistirá en ir barriendo cada fila una tras otra, y para cada fila sondear todas las columnas una tras otra. Puesto que se ha establecido que se debe asociar el nivel lógico 0 a una tecla pulsada, el barrido de filas consistirá en poner a cero una fila dada (y el resto a uno) para entonces sondear el estado de cada columna. Si la tecla de la fila y columna procesada no se hubiese pulsado, el estado que se obtendrá al leer la columna dada será un 1 lógico. El motivo es doble: si en esa columna C_n no hay ninguna tecla pulsada el bit del puerto P1 por el que se lee la columna estará a 1 lógico. Este nivel debería venir determinado por una oportuna resistencia de elevación (no obstante, con los microcontroladores de la familia 8051 no es precisa, en la práctica, tal resistencia puesto que ya existe una en la estructura de los puertos cuasibidireccionales; consúltese la documentación sobre la estructura de los puertos de la familia 51). Y si en esa columna hubiese pulsada una tecla pero de otra fila, igualmente el estado de esa línea C_n sería un 1 lógico puesto que se estaría llevando el estado de la línea que barre fila de la tecla pulsada a la entrada de la señal de la columna C_n, y este nivel es precisamente un 1 lógico dado que esa fila no es la barrida (recuérdese que barrer una fila "**i**" implica poner a 0 la señal F_i, y a 1 las demás).

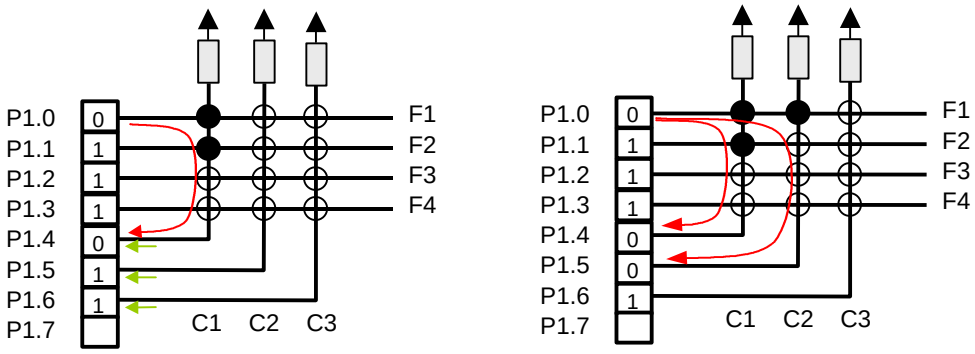


Desde el punto de vista de la programación, la estructura obedecerá a lo siguiente:

1º. Activar fila 1.
2º. Ver si está activa la columna 1: ¿sí?: hacer función asociada a la tecla 1 y continuar.
3º. Ver si está activa la columna 2: ¿sí?: hacer función asociada a la tecla 2 y continuar.
4º. Ver si está activa la columna 3: ¿sí?: hacer función asociada a la tecla 3 y continuar.
4º. Desactivar fila 1.
5º. Activar fila 2.
6º. Ver si está activa la columna 1: ¿sí?: hacer función asociada a la tecla 4 y continuar.
7º. Etcétera (..).
21º. Volver al principio para repetir el barrido del teclado.

Por supuesto, al detectar que se ha pulsado una tecla será necesario proceder al filtrado de los rebotes mecánicos. Esto se puede hacer por programa, temporizando unos cuantos milisegundos (entre cinco y diez suele ser más que suficiente). De igual modo, habrá que decidir a qué instante de la pulsación se asocia la función que representa cada tecla; es decir, si se hace a la pulsación o a la relajación. Lo habitual suele ser a la relajación, pero podría perfectamente ser al contrario. En cualquier caso, siempre es necesario proceder a la detección de la relajación y nuevamente filtrar los rebotes. Podría obviarse este segundo filtrado si la tarea es a la relajación y ésta implica un tiempo igual o superior al del filtrado de rebotes. Todo esto deberá evaluarlo el programador y optar por la solución más oportuna en función de las características operativas del sistema concreto que se esté diseñando.

Otra cuestión que cabe plantearse es el comportamiento del sistema ante anomalías como la pulsación, accidental o no, de más de una tecla. Podrían seguirse múltiples criterios; uno de ellos sería ignorar todas las pulsaciones múltiples. Otro criterio podría ser considerar la tecla más prioritaria, existiendo una jerarquía asociada al orden en que se barren las filas y dentro de una fila al orden en que se sondean las columnas.



En el primero ejemplo de la figura se han pulsado las teclas 1 y 4. Al barrer la fila 1, se tendrá que el 0 que sale por P1.0 se aplicará a la entrada P1.4 puesto que la tecla 1 cortocircuita el conductor de la fila 1 con el de la columna 1. Pero al estar también pulsada la tecla 4 esto hace que se cortocircuite la fila 2 con la columna 1; es decir, se han unido las salidas de las filas 1 y 2. ¿Supone esto algún problema? Pues depende de las características de la capa física de la interfaz. En nuestro caso, con la familia 8051, en donde los puertos son cuasibidireccionales, no sucedería nada y se tendría un efecto de *Y por conexión*, por lo que el estado 0 de salida mandaría sobre el 1, sin problema alguno de tipo eléctrico. Lo único es que se plantearía la duda lógica sobre cómo interpretar esa pulsación múltiple. Si se quisiese prever este tipo de situaciones la capa lógica debería diseñarse para detectarlas (al detectar una tecla pulsada se debería completar el rastreo del teclado para ver si hay alguna otra tecla más pulsada o no, y obrar en consecuencia).

El segundo ejemplo muestra el caso con más de una tecla pulsada en columnas distintas. Aquí puede verse que se tendría más de una entrada activa de columna.

En el caso de que en un sistema existan funciones asociadas a la pulsación simultánea y válida de más de una tecla, la capa lógica deberá ser diseñada conforme a este criterio, debiendo hacerse rastreos múltiples en los casos pertinentes

B) Para ser capaces de detectar la pulsación de una tecla, y que esto genere una petición de interrupción será necesario:

1º. En el proceso de iniciación del sistema, activar simultáneamente las cuatro filas y habilitar las interrupciones por flanco de bajada en **INT0**.

2º. Disponer una lógica externa que detecte cualquier pulsación y active la petición de interrupción. En este caso bastará con una puerta AND de tres entradas (74LS11), una por cada columna del teclado. A cada una de estas entradas se le conectará una de las columnas, de modo que si se pulsa cualquier tecla esto hace que el 0 lógico de su fila se aplique, además de a la entrada de su columna, también a la puerta AND que activará, a baja, la petición de interrupción (si no hay pulsación, las entradas a la AND estarán a 1 debido a las resistencias internas de elevación de las entradas del puerto P1 (P1.4, P1.5 y P1.6).

3º La rutina de servicio realizará el barrido habitual del teclado, teniendo la precaución, antes de retornar de la interrupción, de volver a activar las cuatro filas y borrar por programa la bandera **IE0** de evento **INT0** (el motivo es que los rebotes mecánicos, que sucederán ya dentro de la rutina de servicio, originarán una nueva petición de interrupción que quedará latente hasta que se retorne de la actual en curso).

PRÁCTICA 3:

CONTROL DE UNA IMPRESORA A TRAVÉS DE UN BUS PARALELO ASÍNCRONO

Sea un sistema basado en el μC **AT89C52** funcionando a una frecuencia externa de 12 Mhz. Diséñese una rutina que permita implementar la capa lógica elemental de la interfaz **CENTRONICS**. Hágase una implementación simplificada, utilizando sólo las señales **STROBE**, **ACK** y **BUSY**.

Para implementar la capa física de la interfaz, empléese:

- El puerto **P0** (recuérdese que es a drenador abierto, y que por tanto necesita de las oportunas resistencias de elevación) para el bus de datos CENTRONICS.
- Las señales de E/S **P3.3**, **P3.4** y **P3.5** para implementar las señales de *ocupado, reconocimiento y validación*, respectivamente.
- La señal **P3.7** se utilizará como entrada de un pulsador que permita indicar el momento a partir del que se debe enviar a la impresora un bloque de datos

La gestión de la transferencia puede elegirse entre hacerla por sondeo o por interrupciones. En este último caso la tarea principal será generar una señal de 1 Hz, aproximadamente, y ciclo de trabajo del 50% por la patilla **P3.6**. Si se opta por las interrupciones recuérdese que es necesario, al iniciar el sistema, habilitar las interrupciones por **INT1** (patilla **P3.3**) y programar su sensibilidad al flanco de bajada; lo primero se hará mediante el bit **EX1**, en el registro **IE**, y lo segundo mediante el bit **IT1**, en el registro **TCON**.

Para verificar el correcto funcionamiento de la práctica diseñada, se deberá lanzar a la impresora una cadena ASCII cualquiera. Recuérdese que la impresora sólo imprime físicamente al recibirse el código *retorno de carro* (ASCII 0Dh). Para la temporización del protocolo puede emplearse, caso de que sea necesario, o bien temporizadores internos o bucles de retardo por programa; esto se deja a la libre elección. En un segundo diseño de la capa lógica añádase además una rutina antibloqueo, de manera que si transcurren más de 5 segundos sin poderse transmitir un carácter, entonces aborte la impresión y encienda un LED a modo de error (**P1.7**).

Téngase en cuenta que como no se están empleando todas las señales del bus CENTRONICS será necesario cablear al nivel oportuno aquellas señales de control que no vayan a ser utilizadas. Es decir, **INIT** y **AUTOFEED** a su nivel no activo, mientras que **SELECT IN** deberá activarse. Por el contrario, las señales de estado no usadas se dejaran sin conectar; esto es, al aire.

Una vez hecho el montaje, utilícese un analizador lógico para observar las señales del bus y la temporización de las señales procedente de la impresora. Anótense los valores de los parámetros temporales característicos del bus y determínese cuál sería la máxima velocidad de transferencia que podría alcanzarse a la frecuencia de reloj a la que funciona el sistema patrón (microcontrolador).

PRÁCTICA 4:

COMUNICACIÓN SERIE ASÍNCRONA EIA-232

Basándose en el sistema diseñado en la práctica 2, diséñese una aplicación para la comunicación serie bidireccional, empleando la **UART** integrada del microcontrolador **AT89S52** (considérese que el sistema trabajará a una frecuencia de **11'0592 MHz**). Las características habrán de ser las siguientes:

- Se empleará un enlace serie **EIA-232 a tres hilos**. Para realizar la conversión eléctrica de niveles se empleará un circuito tipo **MAX232**.
- Al pulsar una tecla su código ASCII deberá ser transmitido con el siguiente criterio por la UART: 8 bits, 1 bit de parada, sin paridad, a **9600 baudios** (bits por segundo).
- En la recepción, usando el mismo criterio que para la transmisión, el carácter recibido deberá sacarse por el puerto **P0**. La recepción deberá gestionarse por interrupciones.

Dado que se utilizará un **PC** como sistema al que enviar y desde el que recibir información, el cable de conexión que se empleará será uno tipo *módem nulo*, en el que las señales **TxD** y **RxD** están cruzadas (este tipo de cable está pensado para enlazar dos sistemas que se comporten ambos como ***DTE***). Por tanto, en el conector **DB-9** del cable se tiene que las patillas **2**, **3** y **5** son **RxD**, **TxD** y **COMÚN**, respectivamente. En definitiva, la señal **TxD** del microcontrolador, una vez adaptada a los niveles EIA-232 se conectará a la patilla 3 del cable, y la RXD del cable deberá conectarse a la oportuna entrada EIA-232 del MAX232, cuya salida se aplicará a la entrada **RxD** del micro.

Sugerencias:

Dado que en esta práctica se va a gestionar la transmisión por programa y la recepción por interrupciones, se puede producir un efecto indeseable en la gestión puesto que en la familia 8051 no existen vectores independientes para la recepción y para la transmisión, sino un mecanismo común. Cada vez que se termine de transmitir un dato la bandera **TI** se activará para indicar tal cosa. Como la transmisión se está gestionando por programa, cada vez que haya que transmitir un dato el programa deberá ver si la bandera **TI** está activa, cosa que indica que ya se puede enviar uno nuevo. Por tanto, la rutina de gestión por programa, al ir a enviar un carácter, estará en un bucle de espera hasta que detecte que **TI** se ha activado:

```
JNB     TI, $     ; esperar a que se pueda transmitir
(…)               ; aquí el código que toma y envía el dato al puerto serie
```

El problema es que para poder gestionar por interrupciones la recepción esto implicará tener que habilitar, obligatoriamente, las interrupciones del puerto serie (bit **ES** en el registro **IE** de habilitación de interrupciones). Y esto incluye no sólo a la recepción sino también a la transmisión (esto es algo peculiar de la familia 8051, aunque no de otras familias). Puede pensarse que esto no supone ningún problema porque al diseñarse la rutina de servicio a la recepción (que realmente es de servicio al puerto serie) primero se comprobará por programa que, efectivamente, la causante de la interrupción es la recepción y no la finalización de una transmisión. Y esto es tan sencillo como evaluar la bandera **RI** (o **TI**, a gusto del programador). Por ejemplo, si **TI** está activa se retorna de la interrupción y, en caso contrario se gestiona la recepción. Esto puede apreciarse en el siguiente código:

```
SERVICIO_PUERTO_SERIE:    JBC     RI, RECEPCION
                          RETI                    ; es transmisión: retornar sin hacer nada
RECEPCION:                (…)                     ; aquí el código para tomar el dato recibido
```

¿Qué ocurrirá entonces? Pues que, a diferencia de otro tipo de interrupciones, si la rutina de servicio al puerto serie hace esto entonces se producirá un estado de cuasi bloqueo operativo del sistema. Normalmente, el mecanismo de una interrupción en un 8051 lleva aparejado el borrado automático de la bandera del evento peticionario. Sólo existen dos excepciones: interrupciones por nivel en INT0 o INT1 y por el puerto serie; en estos casos es el programador quien debe borrar *manualmente*, esto es, por programa, la bandera del evento peticionario. En nuestro caso la rutina de servicio al puerto serie tiene dos alternativas. Una, al detectar que la interrupción es por fin de la transmisión, simplemente retornar de la interrupción sin hacer nada, dejando activa la bandera **TI**. Otra, por el contrario, no hacer nada pero retornar desactivando antes, por programa, **TI**. En el primero de los casos se tendría ese estado de cuasi bloqueo; el motivo es claro: al retornarse de la interrupción se ejecutaría la siguiente instrucción al punto en donde se produjo la interrupción, para entonces, justo a continuación, desencadenarse una nueva petición espuria de interrupción. ¿Motivo? Que **TI** sigue estando activa y por tanto actuando como una nueva petición de interrupción, con lo que se estaría repitiendo indefinidamente esta situación *instrucción-interrupción-instrucción-interrupción*… En el segundo de los casos, cuando se borra **TI** antes de retornarse de la interrupción, la gestión por programa nunca detectará que la bandera **TI** se ha activado. El motivo es evidente: al activarse **TI**, automáticamente se producirá una interrupción ante la que la rutina de servicio se limita a desactivar **TI** y retornar de la interrupción, y por eso la rutina de gestión por programa, ejecutada en primer plano, no tendrá ocasión de detectar que **TI** ha llegado a activarse. Entonces, ¿cómo puede resolverse este problema? Muy sencillo: recurriendo a una *copia segura* de **TI**. El programador empleará una *variable de bit* cuya misión será retener el estado de TI cuando ésta se active. Así, ahora la rutina de servicio, al detectar que la interrupción es por transmisión, borrará **TI** pero además activará esta *pseudo bandera* ubicada en la memoria normal del sistema (cualquier *dirección de bit* en la RAM interna puede servir para implementarla; recuérdese esta peculiaridad de la familia 8051). Así, ahora la rutina de servicio podría tener el siguiente aspecto:

```
SERVICIO_PUERTO_SERIE:
                  JBC     RI, RECEPCION ; ¿es por recepción?
                  CLR     TI            ; No, es por transmisión: borrar a mano la bandera TI…
                  SETB    PSEUDO_TI     ; … pero retener su estado en una variable de bit ad hoc.
                  RETI                  ; es transmisión: retornar sin hacer nada
RECEPCION:        (…)                   ; Sí: es por recepción (código que toma el dato recibido)
```

Así, ahora la rutina que gestiona por programa la transmisión deberá evaluar no la bandera **TI**, que no puede serlo por lo ya comentado, sino la bandera **PSEUDO_TI**. Por tanto, cuando se deba transmitir un dato se hará lo que sigue:

```
                  JNB     PSEUDO_TI, $  ; esperar a que se pueda transmitir
                  (…)                   ; aquí el código que toma y envía el dato al puerto serie
```

Por supuesto, este código deberá también tratar a la variable PSEUDO_TI de idéntica manera a como lo hubiese hecho con TI (esto se haría en la parte de código mostrada con los puntos suspensivos).

PRÁCTICA 5:

INTERFAZ CON PANTALLA LCD ALFANUMÉRICA

Partiendo de un sistema basado en el μC **AT89S52** (reloj de 12 Mhz), realícese la interfaz con un visualizador alfanumérico de cristal líquido de dos filas y dieciséis caracteres por línea, de tipo retroiluminado y compatible con el estándar de este tipo de módulos (se adjuntan las hojas de características).

Este tipo de visualizadores poseen un bus de datos de ocho bits, un bus de control de dos bits (señales **E**, de habilitación, y **RW**, de lectura/escritura) y un bus de direcciones de un solo bit (señal **RS**, de selección de registro). También poseen una señal **V_o** para el control del contraste, además de las dos de alimentación **V_{dd}** y **V_{ss}** (**GND**). La manera recta de realizar la interfaz es conectando estos buses a los del sistema de control, típicamente un sistema basado en microcontrolador. Esto implica la necesidad de que el sistema disponga de los oportunos buses de dirección, datos y de control. Actualmente, cuando se utilizan microcontroladores, no se suele realizar este tipo de diseños con buses externos puesto que esto implicaría sacrificar un buen número de patillas que podrían utilizarse para funciones alternativas de E/S. Además, aunque se hiciese, existiría un problema colateral: la velocidad del sistema. Las pantallas LCD son periféricos lentos, por lo que si se conectan a los buses de un sistema *rápido* existiría la necesidad de realizar las transferencias de manera asíncrona. Pero la cuestión es que actualmente muchas familias de microcontroladores –gamas baja y media- no admiten diseños con transferencias asíncronas con la memoria. Pues bien, este tipo de módulos LCD no pueden conectarse a los buses de un sistema 8051 que funcione a más de 11 Mhz. En nuestro caso, tenemos por un lado que se trabajará a más de esa frecuencia, y por otro que no se desea sacrificar los puertos **P0** y **P2** para utilizarlos como buses de dirección/datos del sistema (esto es algo a lo que actualmente no se recurre en los diseños basados en esta familia, suponiendo una técnica totalmente obsoleta desde hace ya algunos años). Por tanto, ¿cómo puede entonces realizarse la interfaz entre un microcontrolador de la familia 8051 y un visualizador LCD alfanumérico? Muy sencillo, recurriendo a la emulación software del protocolo paralelo síncrono. Para ello tan sólo hace falta emplear las oportunas líneas de entrada y de salida digital: un puerto bidireccional para emular el bus de datos, y tres líneas de salidas para emular las señales de control **E**, **WR** y **RS**. De este modo se ahorran líneas de E/S, que incluso pueden ser todavía más si se sigue la técnica de interfaz con bus de datos de cuatro bits, que es admitido por este tipo de visualizadores (en este caso una transferencia de un octeto se lleva a cabo con dos transferencias de dos cuartetos, primero el alto y a continuación el bajo, utilizándose ahora como bus de datos las cuatro líneas de datos de mayor peso, D4 a D7).

En nuestro caso, como el microcontrolador no utilizará los puertos P0 y P2 como buses de dirección ni datos, entonces la interfaz física se hará emulando por programa el bus de datos, direcciones y control que precisa el visualizador alfanumérico (ocho líneas para datos a través del puerto **P2**, habilitación **E** con **P3.5**, lectura/escritura **RW** con **P3.6** y selección de registro **RS** con **P3.7**). Por lo que respecta a la capa lógica, debe realizarse una rutina de demostración que saque un mensaje por la pantalla y realice algún tipo de efecto visual (desplazamiento cíclico, parpadeo, etcétera). El tipo de efecto o efectos se deja a la libre elección.

Una vez realizada la práctica, hágase una versión en la que se trabaje con el LCD mediante un bus de cuatro bits en lugar de ocho, con el fin de ahorrar líneas de E/S dedicadas al control del visualizador. La manera de configurarlo con este ancho del canal de datos es en la secuencia de iniciación, con un carácter especial (ver en las hojas de características esta secuencia de iniciación). Una vez configurado con el canal de cuatro bits, cada octeto debe transferirse (sea en modo lectura o escritura) en dos mitades, la alta primero y la baja a continuación, según el cronograma general indicado en las hojas de características.

Sugerencias y comentarios colaterales:

Para emular el protocolo se sugiere diseñar cuatro rutinas básicas: una para escribir un carácter en el registro de control, otra para escribir en el registro de salida datos, otra para leer el registro de estado y, finalmente, otra para la lectura del registro de entrada de datos (esta última puede no ser necesaria). Estas cuatro rutinas serán muy parecidas y lo único que cambiará será el estado de las señales **RS** y **RW**. Se deberán activar o desactivar las señales oportunas en la secuencia indicada en el cronograma de las hojas de características, asegurándose de que se cumplen los parámetros temporales especificados (adviértase que puede suceder que algunos o todos estos parámetros se satisfagan de manera implícita por el tiempo consumido por las instrucciones que secuencian la activación desactivación de las señales).

Una vez diseñadas estas rutinas se aconseja probar la pantalla sacando un solo carácter. De esta manera se puede comprobar su correcto funcionamiento. Los fallos típicos que suelen aparecer son los siguientes:

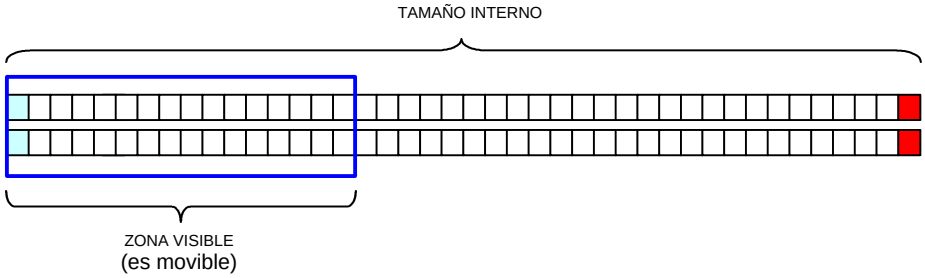
- La pantalla queda en blanco o aparece **sólo la primera línea en negro**. Esto indica un fallo en el proceso de iniciación. Posiblemente los caracteres de la secuencia están equivocados o se han enviado incumpliendo los tiempos de espera entre carácter y carácter de iniciación (consúltese las hojas de características de módulo LCD).
- Aparece basura en la pantalla o se ha borrado pero no aparece el carácter enviado sino otro. El motivo suele ser un error en el conexionado de las líneas del bus de datos (mala conexión de alguna línea debido a quedar al aire o haberse intercambiado con otra).

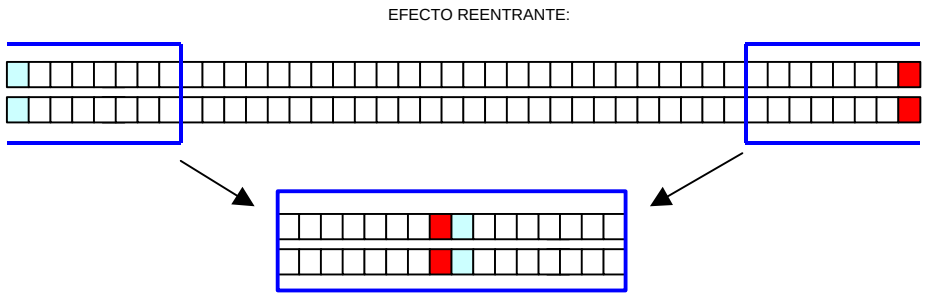
Una vez que se vea que lo básico funciona, ya puede pensarse en desarrollar rutinas de efectos más elaborados. Para ello debe tenerse en mente lo siguiente:

- Todos los módulos LCD alfanuméricos de dos líneas y dieciséis caracteres por línea tienen el mismo modelo lógico: internamente existe una RAM, con una posición por cada elemento de una línea, que siempre tiene una longitud de cuarenta caracteres. Las direcciones de posiciones consecutivas en una misma línea son a su vez consecutivas, pero la dirección del primer carácter de la segunda fila no es consecutiva a la del último en la primera (para los detalles, véase la documentación técnica). Para este tipo de módulos, con dos filas y dieciséis caracteres por fila, existen internamente dos filas pero de cuarenta caracteres cada una, con direcciones 80h a A7h para la primera y C0h a E7h para la segunda. De estos cuarenta caracteres en cada línea sólo serán visibles dieciséis.
- El cursor, sea o no visible, indica el punto (o dirección) en donde se mostrará (o guardará) el carácter enviado. Puede por tanto suceder que se envíen datos y éstos no se vean porque se están metiendo en una zona de RAM que no se está visualizando. En nuestro caso la pantalla de dieciséis caracteres puede interpretarse como una ventana aplicada a una más grande, de cuarenta.
- Mediante el envío de comandos de control el usuario puede seleccionar el modo de trabajo de la pantalla. Por ejemplo, y entre otros:

    - **Ventana estática**: se muestra siempre la misma porción del modelo interno, y los caracteres se muestran sólo si están en la parte visible.

- **Ventana dinámica**: al mismo tiempo que se envía un carácter ésta se desplaza junto con el cursor de modo conveniente (el efecto visual es como si el cursor no se moviese, sino el texto ya escrito).
- **Ubicación del cursor**: útil cuando se desea sacar información en un punto concreto distinto del que indica la actual posición del cursor.
- **Desactivado**: útil para conseguir efectos de parpadeo global, dado que al desactivar se apaga la pantalla pero no se pierde la información en la RAM de datos.
- **Desplazar la ventana**: a derecha o a izquierda; es como si la ventana que supone el visualizador se desplazase sobre el modelo interno de cuarenta caracteres. Es útil para conseguir efectos de desplazamiento del texto de una pantalla sin tener que desplazarlo físicamente (a efectos de aspecto debe considerarse la línea interna de cuarenta caracteres como una línea cíclica, es decir, que tras el último elemento de la línea se mostraría el primero de la misma.





- **Programación de caracteres**: hay modelos (entre ellos el utilizado en la práctica) que poseen ocho caracteres programables, para que el usuario defina sus propios caracteres (por ejemplo, las vocales acentuadas y la eñe). La manera de programar un carácter es muy sencilla: un carácter es un patrón de 5x7 (o 5x8) píxeles, en el que un bit a 1 representa un píxel encendido y a 0 apagado. Cada fila de la matriz se especifica por un octeto, por lo que un carácter es un conjunto de ocho octetos. Los caracteres programables tienen asociada una memoria denominada CGRAM (*character generator* RAM); por tanto, para definir un carácter sólo habrá que escribir en la posiciones oportunas de la CGRAM los octetos pertinentes (para detalles consúltese la documentación técnica del módulo LCD). Una vez hecho esto, cuando se envíe al visualizador el código ASCII de uno de los caracteres programables aparecerá en pantalla el carácter definido según lo que tenga la CGRAM.

PRÁCTICA 6:

COMUNICACIÓN SERIE SÍNCRONA INTERCIRCUITO. EL BUS I2C

Se desea emplear una **EEPROM** en un sistema basado en el μC **AT89S52**. Se optará por la **24C04**, cuyas hojas de características se acompañan. Dado que es un dispositivo **I2C**, será preciso emplear tal tipo de interfaz con élla. Como el **AT89S52** carece de un controlador **I2C** integrado, y dado que no se desea emplear uno externo, se va a realizar la interfaz empleando como capa física un par de líneas del puerto 1: **P1.0** como **SDA** y **P1.1** como **SCL**. Dado que los puertos son cuasi-bidireccionales, esto implica que se puede emplear una línea tanto como entrada (el bit del cerrojo de salida debe estar a 1) como salida; esto significa que perfectamente pueden emplearse para implementar las señales **I2C** de una manera económica en lo que a la capa física se refiere. Dado lo elemental de esta capa toda la carga del protocolo recaerá en la capa lógica, como es previsible. Ésta deberá tener las siguientes características:

- Existirá una rutina cuya misión será generar una condición de **PARADA**.
- Existirá una rutina que lance un octeto, residente en el acumulador, según el protocolo **I2C**.
- Existirá una rutina que lance un octeto, en el acumulador, pero precediéndole de una condición de **INICIO**.
- Existirá una rutina que lea un octeto a partir de la línea **SDA**, quedando en el acumulador (deberá dar, o no, el oportuno reconocimiento).

Además de estas rutinas básicas, y basándose en éllas, se construirán otras rutinas que hagan lo siguiente:

a) Permitir guardar un solo octeto en la **E2PROM**. Como condiciones de entrada, el dato se asume que habrá de residir en donde apunte **R1** dentro de la RAM interna, y la dirección de la **E^{2}PROM** en que se guardará se indicará mediante **R4**.

b) Permitir guardar un bloque de datos en la **E^{2}PROM**. Las condiciones de entrada serán: **R6** indicará el tamaño del bloque, **R1** la dirección en la RAM interna a partir de la que están los datos que se desea guardar, y **R4** la dirección en la **E^{2}PROM** a partir de la que se desea guardarlos.

c) Permitir guardar una página. Las condiciones de entrada son como en el anterior punto, salvo por lo del tamaño de bloque.

d) Permitir leer un bloque desde la **E^{2}PROM**. Las condiciones de entrada son: **R1** indicará la posición en la RAM interna a partir de la que se guardarán una vez leídos. **R4** indicará a partir de qué posición de la **E^{2}PROM** se leerán los datos y **R6** indicará el tamaño del bloque a leer.

Para ilustrar el correcto funcionamiento, se realizará una aplicación con las siguientes características: a la puesta en alimentación, leerá los 16 primeros datos en la **E^{2}PROM** y los transmitirá por medio de la UART integrada, a 9600 bps. Si se pulsa una tecla conectada a **P1.6** se guardarán los cinco últimos datos en su lugar en la **E^{2}PROM** pero incrementado su valor en una unidad. Una vez hecho esto, se esperará a que la línea **P1.6** se inactive para entonces leer esos cinco datos de la **E^{2}PROM** y obrar como cuando se leyeron los 16 primeros. Si se pulsa una tecla conectada a **P1.7** entonces los dieciséis primeros valores de la **EEPROM** se copiarán a partir de la dirección **30h** de ella.

Con un osciloscopio digital mídanse los parámetros temporales resultantes en el bus I2C.

Sugerencias y comentarios colaterales:

Al igual que se hizo con la pantalla LCD alfanumérica, aquí se emulará por programa el protocolo I2C. Esto es algo enormemente común puesto que, en aplicaciones sensibles al coste, elegir un miembro de la familia que integre el controlador I2C oportuno implicará un sobrecosto del sistema. En estos casos, y especialmente cuando la velocidad de comunicación no es un parámetro crítico, es una solución muy aceptable recurrir a emular por programa el bus I2C –al igual que otros buses de comunicación con periféricos– empleando como soporte físico unas simples señales de uno o varios puertos de E/S digital. Por supuesto, tal y como ya se ha mencionado, esto supone una drástica caída de rendimiento del bus I2C pero si la aplicación de que se trate lo admite entonces este inconveniente no será realmente tal.

En este caso la emulación por programa del protocolo **I2C** necesita tan sólo un par de líneas de E/S cuyo control se puede realizar con las oportunas instrucciones **CLR** (puesta a 0) y **SETB** (puesta a 1) si se desea controlar las señales, y si se desea evaluarlas **JB** (bifurca si el bit direccionado vale 1) o **JNB** (bifurca si el bit direccionado vale 0). Así, las rutinas que implementen el protocolo deberán realizar estas funciones cumpliendo los criterios de temporización que marque la norma. Para el modo básico I2C muy posiblemente la mayoría de estos parámetros temporales quedarán satisfechos de manera implícita debido al tiempo que consumen las diferentes instrucciones empleadas, y de no ser el caso entonces quedará resuelto el problema con alguna que otra instrucción **NOP** oportunamente intercalada. Para tener la absoluta certeza de que se cumplen estos requisitos será preciso recurrir a la especificación de la norma **I2C** y además conocer el tiempo que consumen las diversas instrucciones utilizadas (para esto bastará con conocer la frecuencia a la que trabajará el microcontrolador y los ciclos máquina que consume cada instrucción, información ésta que puede obtenerse de las hojas técnicas del microcontrolador utilizado). En nuestro caso se utilizará un controlador, el **AT89S52**, que trabaja con el ciclo máquina tradicional (esto es, doce ciclos de reloj externo), por lo que el problema de tiempos, a 12 Mhz, prácticamente será irrelevante; pero debe tenerse en cuenta que la familia 8051 ha sido objeto de innumerables mejoras en los últimos años, siendo posible encontrar actualmente derivados en los que el ciclo máquina es de sólo un ciclo de reloj externo, pudiendo funcionar excesivamente rápido para lo que el bus **I2C**, en modo estándar, necesita. Por ejemplo, **ATMEL** acaba de lanzar al mercado el derivado **AT89LP4052**, capaz de trabajar a 20 Mhz; esto equivaldría a un miembro tradicional funcionando a una frecuencia de **240 Mhz**, cuando lo habitual son frecuencias como mucho de 24 o 33 Mhz. Se comenta todo esto simplemente para que se sea consciente de los aspectos que hay que tener en cuenta en una implementación real.

En esta práctica, además, hay que tener una precaución adicional. Debido a la peculiar estructura de los puertos de E/S se ha visto que es factible utilizarlos de manera muy cómoda para emular las líneas **SDA** y **SCL** del bus **I2C**. No obstante, y justamente por ello, existe un problema colateral que habría que solucionar; se trata del valor de las resistencias de elevación que necesita el bus **I2C**. Puede pensarse en aprovechar la que ya integra cada línea de E/S, pero su valor haría que en ciertas circunstancias no se satisficiesen determinados parámetros temporales (tiempos de subida y de bajada, esencialmente). Por este motivo es necesario utilizar en la línea **SCL** una resistencia de elevación, externa, de entre **15** y **22 KÙ** que permita un adecuado trabajo con tal línea. En **SDA** no es precisa esta resistencia externa de elevación; el motivo es que para ella los tiempos de subida y bajada no son críticos.

Otra precaución adicional que es necesario tener consiste en asegurarse de que el cerrojo de salida de **SDA** está a 1 cuando se use esta línea como entrada (al hacer una lectura desde la **EEPROM**). De no hacerse así, se podría falsear la información en SDA. Recuérdese que el bus **I2C** permite la *Y por conexión* en **SDA** y **SCL**.

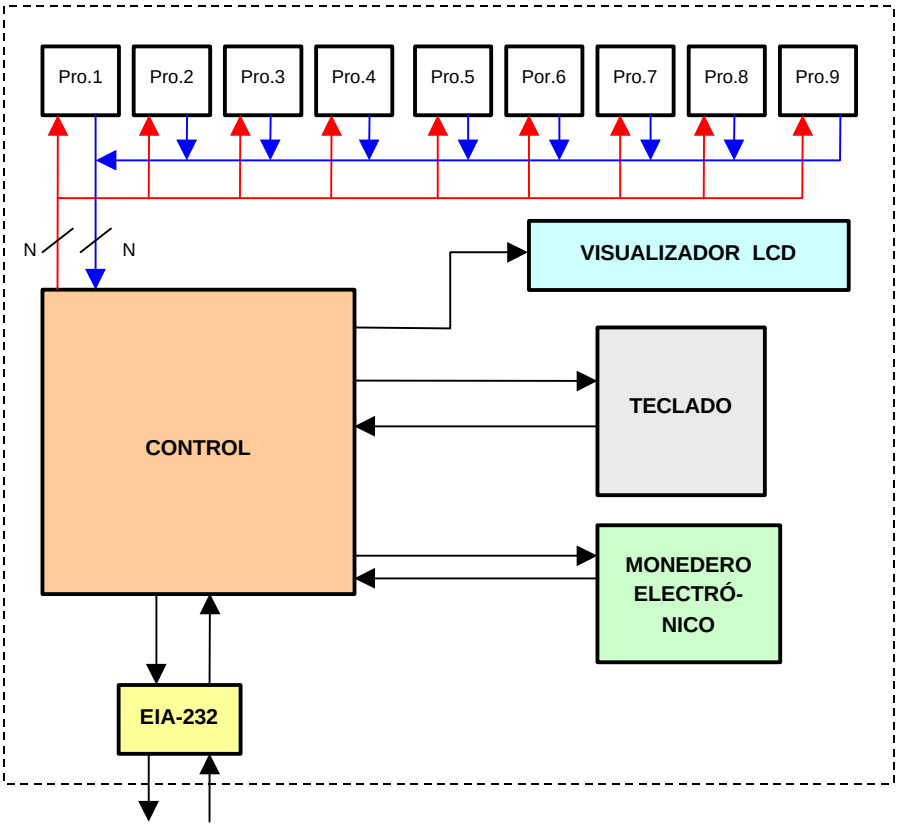
PRÁCTICA 7 VOLUNTARIA:

DISEÑO DE UN SISTEMA COMPLETO DE CONTROL EMPOTRADO

Se trata de diseñar el control de una máquina expendedora de hasta cuatro productos distintos. Constará de una pantalla LCD alfanumérica como la de la práctica 5, un teclado matricial como el de la práctica 2, y un selector de monedas cuyas características están disponible bajo petición. Con las teclas numéricas se selecciona el producto, cuya descripción y precio debe aparecer en la pantalla. Con la tecla ***** se confirma y con # se anula. Una vez seleccionado el producto se pide que se introduzca el dinero, cuya cantidad acumulada debe ir apareciendo a medida que se introducen monedas. Una vez introducida la cantidad exacta, se espera la confirmación para expulsar el producto. Esta expulsión se consigue poniendo a cero durante 100 ms una oportuna señal de control. El sistema tendrá la capacidad de ser programado a través de un enlace serie EIA-232 con un sistema local o remoto (PC o mando de programación), disponiendo de una memoria E^2PROM serie para contener los parámetros de programación. Estos parámetros serán el número de productos diferentes que se ofertan, su descripción, a qué tecla se asocia cada producto y su precio. Se deberá tener la capacidad de detectar si hay existencias de cada producto para en caso contrario rechazar la selección hecha y avisar, de algún modo, de tal eventualidad.

Caso de estarse interesado en realizar esta práctica, pídase la información más detallada que sea precisa. En este caso, no sería necesario realizar las prácticas 1 a la 6, sustituyéndose todas por este trabajo opcional.

Como puede apreciarse, en esta práctica se integran los problemas de interfaz de las anteriores, dando lugar a un sistema integral dedicado al control, de características totalmente reales.



NORMAS DE ELABORACIÓN DE LOS GUIONES DE PRÁCTICAS:

Se deberá realizar, para cada práctica, el correspondiente guión. Para su elaboración se deberá seguir la siguiente pauta:

- Enunciado.
- Cálculos, caso de ser necesarios. Aquí se justificarán, si resulta oportuno, las decisiones o criterios que se hayan seguido.
- Explicación del desarrollo, comentando los problemas encontrados en la depuración del código, a qué se han debido éstos y cómo se han resuelto.
- Esquema eléctrico del montaje físico.
- Diagrama ASM de los procesos lógicos, si se estima conveniente, y listado comentado del código.
- Comentarios y conclusiones que se hayan extraído.

La entrega de estos guiones se puede realizar hasta el día 15 de julio de 2006. Existirán siete sesiones de prácticas de laboratorio, de dos horas cada una. Se estima que con este tiempo más una hora y media de preparación para cada práctica es suficiente para su desarrollo, montaje y depuración. Caso de que algún alumno desee dedicar mayor tiempo de laboratorio, durante el período lectivo o una vez finalizado éste, para completar las prácticas o realizar apartados voluntarios tiene a su disposición el laboratorio en horario de mañana y tarde, previa consulta de su disponibilidad y acuerdo con el profesor. Esta dedicación extra, de producirse, será oportunamente valorada y podrá tener reflejo en la calificación de la asignatura.