Maqueta: Juego de ping-pong en un osciloscopio

1. Introducción

Los osciloscopios convencionales disponen de un modo de presentación denominado XY mediante el cual se pueden controlar las deflexiones horizontal (X) y vertical (Y) del punto luminoso (spot) generado por el haz de electrones. De esta forma es posible realizar dibujos en la pantalla mediante la técnica de representación vectorizada, conocida también como método vectorial (raster, en ingés).

El método vectorial se basa en mover el spot por la pantalla de forma que dibuje los elementos gráficos necesarios de foma cíclica. La modificación conjunta y coordinada de ambas deflexiones permite realizar cualquier dibujo. La ciclicidad es necesaria para mantener el dibujo en la pantalla. Este método, muy utilizado hace varias décadas, ha sido desplazado por el método de exploración (scan) empleado en los monitores de PC y los televisores, debido a la mayor versatilidad de este último. El método vectorial, sin embargo, es el más adecuado para realizar esta práctica.

El juego de ping-pong propuesto tendrá las siguientes características:

1) Se dibujará el contorno del campo en las coordenadas extremas representables, es decir 0 y FFh.

2) Existirá una sola raqueta que se moverá utilizando un ratón por toda la mitad derecha del campo (en dos dimensiones).

3) La pelota saldrá del centro del borde izquierdo del campo con una trayectoria ascendente que forme un ángulo de 45º con el borde. Se empleará una velocidad razonable.

4) Cada vez que *choque* con el borde izquierdo (frontón), superior o inferior del campo rebotará de forma especular. Si alcanza el borde derecho (punto perdido) saldrá una nueva pelota por el borde izquierdo según se indica en el punto 3.

5) Cuando la pelota choque con la raqueta, saldrá rebotada de forma especular siempre hacia la izquierda.

2. Implementación del método vectorial

El método vectorial se puede realizar de forma bastante sencilla si se limta al dibujo de puntos. Para representar cualquier otro elemento geométrico, es preciso definirlo como una tabla de puntos. Se recomienda emplear este método, que se describe en mayor detalle a continuación.

El ciclo de representación debe realizar las siguientes tareas:

1) Recorrer una tabla de objetos a representar.

2) Para cada objeto recorrer la tabla de puntos que definen su forma.

3) Para cada punto, enviar sus coordenadas a los puertos (por ejemplo, *x* al puerto 4 e *y* al puerto 5). Las coordenadas de cada punto son las del objeto más las relativas del punto dentro de su forma.

Para conseguir una presentación visual adecuada es preciso mantener las coordenadas de cada punto durante un cierto tiempo t_p antes de pasar al siguiente. De esta forma se obtiene un brillo suficiente de los puntos y los trazos entre puntos apenas son visibles. Para ello se recomienda utilizar un temporizador que genere interrupciones periódicas, por ejemplo cada 100 μs. La ISR actualiza los puertos con las coordenadas *x* e *y* del siguiente punto. Después del último punto del último objeto se vuelve al primer punto del primer objeto.

2.1 Definición de una forma.

Para definir una forma se puede utilizar una tabla de valores terminada en un marcador. Por ejemplo:

```
linea_vertical:
db 0,0
db 0,1
db 0,2
db 0,3
db 0ffh
```

define una línea vertical formada por 4 puntos. Se reserva el valor FF para indicar que no hay más puntos. Esto no debe ser un problema ya que se utilizarán objetos pequeños. Se supone que las formas no cambian (no se deforman), por eso el lícito ubicarlas en ROM, como constantes , empleando la directiva *db*.

2.2 Definición de un objeto.

Para definir un objeto es preciso indicar su forma y su ubicación. En esta aplicación no es necesario gestionar giros de objetos. Se supone que cada objeto tiene asignada una forma permanentemente. Sin embargo, es preciso poder variar su posición (objetos móviles). Por este motivo se recomienda definir los objetos en RAM externa y acceder a ellos utilizando las instrucciones de tipo MOVX.

Esta forma de acceso es compatible con la configuración HW estándar en la que las memorias de programa y datos emplean mapas de memoria separados. También es compatible con la configuración de mapas combinados empleada en la placa Altair. Dado que la configuración de la placa Altair combina el código de la aplicación y los datos en la misma memoria, sería posible también definir los objetos mediante directivas *db* (menos recomendable por su falta de portabilidad).

Si optamos por la alternativa recomendada debemos asignar una zona de XRAM para las tablas de objetos. Se supone que el número de los objetos es fijo y predefinido. Por ejemplo:

```
raqueta xdata 0A000h      ; de A000 a A003 para raqueta
otracosa xdata 0A004h     ; etc.
(..)
yanohaymas xdata ????     ; para marcar el final de la lista
```

```
; Se define raqueta con forma de linea_vertical, situada
; inicialmente en las coordenadas (10,20).

mov dptr,#raqueta
mov a,#LOW linea_vertical
movx @dptr,a      ; Dirección de la forma (LSB)
inc dptr
mov a,#HIGH linea_vertical
movx @dptr,a      ; Dirección de la forma (MSB)
mov a,#10
movx @dptr,a      ; Coordenada X
mov a,#20
movx @dptr,a      ; Coordenada Y
(...)
mov dptr,#yanohaymas
mov a,#0FFh       ; marca de final de tabla de objetos
movx @dptr,a
(...)

; Para mover raqueta se modifican sus coordenadas.
; La ISR del temporizador conoce la estructura de datos
; utilizada y puede así dibujar los objetos en la
; posición correcta.
```

Alternativamente, se puede definir una variable con el número total de objetos y no emplear la marca de final de tabla de objetos.

3. Gestión del ratón.

Para mover la raqueta se emplea un ratón serie estándar de PC, en el modo *mouse system*. El ratón envía tramas asíncronas a 1200 bps con 8 bits de datos. Mediante una placa adaptadora, las tramas llegan al puerto P3 del microcontrolador. Se utilizará la UART SW desarrollada previamente para recibir estas tramas.

Cada vez que se mueve el ratón o se pulsa un botón, se genera una trama de 5 bytes con el formato:

1º byte.- 1 0 0 0 0 I C D.

(I=0 si el botón izquierdo está pulsado, idem central y derecho)

2º byte.- Coordenada X del ratón.

3º byte.- Coordenda Y del ratón.

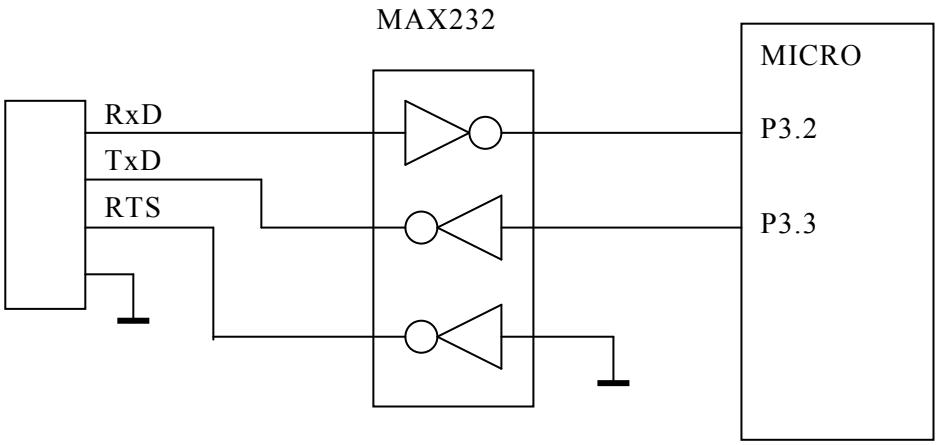
4º y 5º bytes.- Coordenadas x e y relativas al último envío.

Se recomienda utilizar solamente la información de los 3 primeros bytes. Aún así, los dos últimos deben recibirse para no perder el sncronismo. Cuando se espere el primer byte de una trama, se debe comprobarse que los 5 bits más significativos son 10000 y descartar el byte si no se cumple esta condición, es decir seguir esperando la llegada de un 1º byte válido.

La forma más simple de recibir una trama se basa en adaptar la ISR del temporizador empleada en la UART SW para recibir bits. En la ISR original existe un punto en el que se considera que ha llegado un byte completo. En ese punto se puede añadir el código necesario para capturar la información (pulsadores, coordenadas) según el valor de un contador nuevo de posición del byte dentro de la trama (inicalmente 0= byte de pulsadores). Cuando el contador el contador alcanza el valor 4 (último byte de la trama), se debe reponer a cero.

4. Circuito del adaptador.

En primer lugar se muestra el esquema de la interfaz del ratón. La línea RTS se utiliza para alimentar el ratón. El MAX232 entrega aproximadamente 8 V en esta línea. La línea TxD se emplea para reiniciar (reset) el ratón. Cuando el micro pone P3.3 a '0' el reset está activo. Cuando lo pone a '1' el ratón funcina normalmente además de recibir -8V del MAX232.



A continuación se muestra el equema de la parte del circuito dedicada a generar las tensiones que se envian al osciloscopio (Vx y Vy). Utiliza 2 DACs de 8 bits con sus correspondientes amplificadores operacionales.