LA PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO

SÓLO PROGRAMADORES

por desarrolladores ESCRITA para desarrolladores

Revista especializada para usuarios de PC

AÑO 3. Nº 23
1250 PTAS.

ARGENTINA 9'50 $
CHILE 3000 $
PORTUGAL 1500$

EL PODER DE LOS 32 BITS

WATCOM C++

CD-ROM DE REGALO
COMPILADOR DE JAVA Y FICHEROS DE AYUDA

ESTUDIO DE LOS APPLETS DE JAVA

LINUX: CÓMO CREAR DISCOS DE EMERGENCIA A MEDIDA

Y además:

- Redes locales: Interconexión y segmentación
- Grandes sistemas: Objeto natural
- RedesTCP/IP bajo UNIX

CURSOS PRÁCTICOS DE:

Creación de un compilador
Visual Basic 4.0
Visual C++ 4.0
Programación de una demo

TOWER COMMUNICATIONS, S.R.L.

8 413042 303299 00023

INTRODUCCIÓN A LOS REGISTROS DE LA VGA

Santiago Romero, Miguel Cubas y Vicente Cubas

En el número anterior se estudio cómo inicializar y usar el modo X, pero sin adentrarnos en las posibilidades que nos ofrece la tarjeta VGA. En este número vamos a tomar el control nuestra tarjeta de una forma profesional gracias a los registros que la VGA usa para controlar sus parámetros internos

El PC es un ordenador extraño. En sistemas como el Amiga, por ejemplo, es indispensable el uso de los chips que la acompañan (tanto gráficos como sonoros) mediante ordenes hardware debidamente documentadas.

Los programadores del PC no disponen de estos chips ni de esta documentación y se tienen que arreglar con la utilización de los registros internos que la VGA usa para almacenar datos necesarios para su funcionamiento. Cosas como saber la localización del haz de electrones que redibuja la pantalla podría parecer algo meramente informativo pero puede ser usado para realizar vistosos efectos de paleta como las *Copper Bars* en modo texto (ver COPPER.EXE en los ejemplos del disco) y es indispensable para llevar una precisa sincronización y evitar parpadeos y "nieve" en las operaciones gráficas que requieran una gran velocidad.

Estos registros internos, que llevan parámetros importantes para la VGA, pueden ser accedidos mediante los puertos E/S a los que están conectados, que van desde el 3b0h al 3dfh y nos permiten su modificación y lectura, aunque aquí vamos a ver sólo los que nos interesan para la programación de los modos *unchained*. Hay que puntualizar que las tarjetas Super VGA disponen de nuevas funciones no estándar de control cuyo uso debe evitarse para asegurar la compatibilidad de nuestros programas en otras tarjetas, aunque aquí sólo se describirán aquellas funciones que puedan ser utilizadas en cualquier ordenador que disponga de una tarjeta VGA sin que haya ningún tipo de problema de compatibilidad.

LOS CONTROLADORES ASOCIADOS A LA VGA

Veamos en la tabla 1 los nombres de los componentes que son manejados mediante cada puerto, dándonos acceso a los registros de la VGA mediante los controladores de cada uno de ellos. En esta tabla todos los sufijos *_ADDR* son abreviatura de ADDRESS (dirección). Comencemos con la definición y explicación de cada uno de ellos:

Attribute Controller *(ATTR_ADDR, puerto 3c0h):* es el controlador de atributos, relacionado con todos los temas de colores, atributos, intensidad y parpadeo, paleta, etc... así como de control del color tanto del borde como de los planos y que está situado en el puerto 3c0h.

VGA Sequencer *(SEQU_ADDR, puerto 3c4h):* es el secuenciador de la VGA y maneja las funciones más críticas e importantes de la VGA tales como el reseteo de la tarjeta, el generador de carácteres, el *Memory Mode Register* (el cual contiene el bit CHAIN-4 que desencadenaba los 4 planos), etc... Su control se obtiene mediante el puerto 3c4h.

Graphics Controller *(GRAC_ADDR, puerto 3c3h):* es el controlador gráfico de la VGA, que estando situado en el puerto 3ceh da acceso a la lectura de planos (Ver GetPixelX( );), al *Miscellaneous Register*, al *Mode Control Register* y a funciones de operaciones lógicas y reseteo de registros.

Cathode Ray Tube Controller *(CRTC_ADDR, puerto 3d4h):* maneja todos los datos referentes al tubo de rayos catódicos que refresca o redibuja la pantalla, en un sistema asociado al monitor. Mediante este controlador, situado en el puerto 3d4h, tenemos acceso a información como el comien-

zo y fin del retrazo (tanto vertical como horizontal), el número de líneas a redibujar (*Total Vertical Register*), a la posición del cursor y a otras muchas funciones de pantalla. Este chip tiene una protección contra escritura que debe ser eliminada antes de modificar cualquier registro que sea accedido mediante el CRTC_ADDR o puerto 3d4h, aunque se pueden leer sus registros sin manipular la protección. La manera de eliminar la protección la veremos en la sección dedicada a la programación de los controladores.

VGA Enabler / Disabler *(VGAENABLE_ADDR, puerto 3c3h):* situado en el puerto 3c3h, el *least significant bit* (LSB = bit 0) de este puerto puede anular el acceso a los puertos VGA.

Miscellaneous Output *(MISC_ADDR, puerto 3c2h):* mediante este importante puerto, el 3c2h, pueden realizarse funciones de selección de polaridad, reloj, etc... Su uso lo encontramos, por ejemplo, en la rutina de inicialización del modo X, forzando al monitor a refrescar a 60 Hz mediante la utilización de este registro.

El ***SVGA Chip Controller*** *(CHIPSTECH_ADDR, puerto 3d6h)* no es estándar y por ello no va a ser usado para nuestros ejemplos, pero es interesante que los lectores conozcan su dirección E/S (3d6h) y que sepan que es el puerto mediante el que se accede a los registros específicos de cada tarjeta Super VGA.

PROGRAMACIÓN DE LOS REGISTROS DE LA VGA

La programación completa de la VGA requeriría una sección por sí misma y de hecho los lectores de Sólo Programadores ya disponen de la sección «Programación de la VGA» en algunos números de la revista, donde los registros son analizados con más complejidad y extensión, así que se van a introducir sólo sus fundamentos de una manera asequible y siempre pensando en los modos *Unchained*.

La programación de los registros es sencilla gracias a la conexión de estos a los puertos E/S, pero debido a la cantidad de registros y funciones de que disponemos hay que introducir un concepto denominado *índice de puerto* (*Port Index*).

ÍNDICES DE PUERTO

Para escribir o leer de un registro hay que saber el puerto en el que está y el índice de puerto que tiene. El número de índice de puerto lo podríamos comparar al concepto de función de una interrupción. Como ejemplo, al imprimir textos mediante la interrupción 21h del MS-DOS escribimos:

```
MOV DX, offset Cadena
MOV AH, 9          ; número de función
INT 21h            ; llamada
```

Pero la misma interrupción la podemos usar para, por ejemplo, salir al MS-DOS:

```
MOV AH, 4Ch          ; número de
funciónINT 21h       ; llamada
```

El significado que tiene AH al llamar a la interrupción se puede comparar con el que tiene el índice de puerto, porque al igual que la interrupción 21h tiene diversas funciones, un puerto VGA puede controlar más de un registro, y necesitamos indicarle cuál de ellos queremos leer o modificar. Así pues, el índice de puerto le indica al puerto al que se escribe cuál es el registro que queremos modificar de todos los que ese puerto controla. Únicamente hemos de enviar los parámetros tal y como los requiere cada controlador, y ese es el principal problema de la programación a bajo nivel de la VGA.

En la VGA existen varias maneras distintas de leer o modificar los registros. Esto es debido al sistema que sigue cada periférico:

ATTRIBUTE CONTROLLER, ATTR_ADDR (3c0h)

- ESCRITURA:

Antes de poder acceder al Attribute Controller debemos limpiar un "*flip-flop*" interno de la VGA que controla la lectura y escritura. Esto se hace simplemente leyendo del puerto de *STATUS*, al que llamaremos *STATUS_ADDR*, situado en el puerto 3dah. Este registro se comentará al final de este artículo por su gran importancia. Después de leer del puerto de estado (STATUS_ADDR), se le envía al Attribute Controller el número de índice de puerto (el registro que se quiere leer o modificar) y más tarde el valor que se desea asignar al registro. De entre todos sus registros y funciones, el index port enviado nos asegura que el registro leído/modificado es el deseado.

```
inportb ( STATUS_ADDR );
outportb( ATTR_ADDR , index_port ); //
index port
outportb( ATTR_ADDR , valor );      //
valor
```

Después de cada acceso al Attribute Controller se desactiva el refresco de la pantalla y, como se puede ver en el **listado 2**, al finalizar las operaciones de escritura activamos de nuevo este refresco mediante:

```
outportb( ATTR_ADDR, 0x20 );
//reactivar refresco
```

- LECTURA:

Al igual que en la escritura, debe leerse primero del puerto de estado (3dah) y después enviar el índice de puerto al ATTR_ADDR. El valor del registro que queremos leer lo obtendremos del puerto ATTR_ADDR + 1:

```
inportb ( STATUS_ADDR );
outportb( ATTR_ADDR , index_port );
// enviar index-port
valor = inportb( ATTR_ADDR+1 );     //
leer valor
```

IDENTIFICADOR:	PUERTO:	NOMBRE:
ATTR_ADDR	3c0h	Attribute Controller.
SEQU_ADDR	3c4h	VGA Sequencer.
GRAC_ADDR	3ceh	Graphics Controller.
CRTC_ADDR	3d4h	Cathode Ray Tube Controller.
VGAENABLE_ADDR	3c3h	VGA Enabler/Disabler.
MISC_ADDR	3c2h	Miscellaneous Output.
CHIPSTECH_ADDR	3d6h	SVGA Specific Registers.

Tabla 1. Componentes controlados por la VGA.

Para probar el ATTR_ADDR disponemos de un ejemplo de cómo leer el color del borde de la pantalla VGA usando el Attribute Controller:

```
inportb ( STATUS_ADDR );        // limpiar flip-flop interno
outportb( ATTR_ADDR , 11h );    // función leer borde
color = inportb( ATTR_ADDR+1 );  // leer valor del puerto
```

Otra manera de no desactivar el refresco de pantalla cuando se accede al Attribute Controller es activar el bit 5 del índice, que si está a 0 hace que la VGA deje de redibujar la pantalla. Esto lo podemos hacer de muchas maneras, aunque la más sencilla es sumarle 32 al índice (El valor que indica el bit 5: 32 = 100000b). Si nos olvidamos de activar el bit 5 no ocurrirá nada gracias a la línea añadida al final de las funciónes *RegisterIn* y *RegisterOut* que asegura el correcto uso de este controlador, al igual que ocurre en el anterior apartado de escritura.

GRAPHICS CONTROLLER, GRAC ADDR (3ceh) CATHODE RAY TUBE CONTROLLER, CRTC_ADDR (3d4h) SVGA CHIPS CONTROLLER, CHIPSTECH_ADDR (3d6h)

- ESCRITURA:

En estos tres controladores la escritura se hace de similar manera. Se envía el Index Port (registro a modificar) al puerto correspondiente y se escribe el valor que se le desea asignar en el puerto asociado, PUERTO+1.

```
outportb( CRTC_ADDR, index );   // índice de puerto
outportb( CRTC_ADDR +1 , valor ); // valor a enviar
```

Estas dos llamadas a outportb también se pueden concretar en un solo outport gracias a que, como se comentó en el número anterior, outport(), outpw() y out dx, ax envían el byte bajo del valor de 16 bits que se le pasa al puerto especificado y después envían a Puerto+1 el byte bajo, de manera que pueden reducirse estas dos órdenes a:

```
outport( CRTC_ADDR, index | valor<<8 ); //enviar word
```

De esta manera, colocamos *index* en el byte bajo y *valor* en el byte alto (valor << 8) para que outport haga el trabajo en una sola orden. Mediante el OR (|) fundimos los dos bytes en un word.

Este sistema de programación de controladores debería sernos familiar ya que es el mismo que se usaba en el número anterior al seleccionar el plano de lectura de la VGA, controlado mediante la función 4 (Index port 4) del Graphics Controller:

```
outportb( GRAC_ADDR, 0x04 );    // índex 4 = selec. plano
outportb( GRAC_ADDR +1, ( x&3 ) ); // plano a leer
```

Este sistema es usado en muchas operaciones de inicialización, reseteo y control de la tarjeta, y nos permite controlar los 3 componentes de la misma manera, aunque hay que hacer notar que el CRTC Controller dispone de una protección contra escritura de registros que debe ser eliminada antes de intentar modificar cualquier registro que sea accedido mediante el puerto 3d4h. En operaciones de lectura no necesita ser eliminado. Su desprotección se realiza poniendo a 0 el bit 7 del índice o registro 11h, realizándose primero la lectura por el método ya descrito y después la desprotección. Este bit (bit 7) cuando está a uno impide la modificación de los registros y debe ser limpiado (bit 7 = 0) para realizar nuestras operaciones de escritura:

```
void UnProtectCRTC( void )
{
int v ;
outportb( CRTC_ADDR, 0x11 );    // Índice = 11h
v = inportb( CRTC_ADDR + 1 );   // 3d5h
v = v & 0x7F;   // bit 7 = 0
outportb( CRTC_ADDR, 0x11 );    // Índice = 11h
outport( CRTC_ADDR, 0x11 | ( v << 8 ) );   // desproteger
}
```

- LECTURA

La lectura es muy sencilla y consiste en envíar el índice de puerto al controlador correspondiente y leer del puerto contiguo, tal y como hace el siguiente código:

```
outportb( GRAC_ADDR, index );   // index port
v = inportb( GRAC_ADDR +1 );    // valor
```

VGA SEQUENCER, SEQU_ADDR (3c4h)

La programación del secuenciador de la VGA es igual que la de los tres anteriores salvo la excepción de escritura cuando el puerto de índice es 1. La función 1 significa la modificación del

ATTRIBUTE CONTROLLER | <ATTR_ADDR> | PUERTO: 3c0h

ÍNDICE:	NOMBRE:	DESCRIPCIÓN DEL REGISTRO:
10h	Mode Register. (Read/Write)	Bits: (los más importantes) -------- 0 Si es 1, modo gráfico. Si es 0, modo alfanumérico. 1 Modo monocromo si es 1. Modo de color si es 0. 2 Si 1, carácteres de 9 bits. 3 Si es uno, el bit 7 del byte de atributo significa parpadeo. Si es 0 = alta intensidad. 5 (VGA) Si 0, se ignora la comparación de línea. 6 (VGA) Si 1, anchura del pixel = 8. Para modos de 256 col.
11h	Border Color Register. (Read/Write)	Bits: -------- 0-5 Color del borde de la pantalla VGA.
12h	Color Plane Enable Register. (Read/Write)	Bits: -------- 0 Si vale 1, Plano 0 activado. 1 Si vale 1, Plano 1 activado. 2 Si vale 1, Plano 2 activado. 3 Si vale 1, Plano 3 activado.
13h	Horizontal Pixel Panning Register. (Read/Write)	Bits: -------- 0-3 Indican el nº de píxels a desplazar la pantalla hacia la izquierda. Ver tabla de HPPR.
14h	Color Select Register. (Read/Write)	Bits: -------- 0-7 Indican modo de acceso a la tabla DAC de la VGA

VALOR	TEXT	256 color
0	1	0
1	2	n/a
2	3	1
3	4	n/a
4	5	2
5	6	n/a
6	7	3
7	8	n/a
8	9	n/a

Tabla HPPR.

Tabla 2. Registros del Attribute Controller.

GRAPHICS CONTROLLER	(GRAC_ADDR)	PUERTO: 3ceh

ÍNDICE:	NOMBRE:	DESCRIPCIÓN DEL REGISTRO:
05h	Mode Register. (Read/Write)	Para una completa descricpción del mode register, ver sección **Programación de la VGA** de Solo Programadores nº 20. Bits: (los más importantes) -------- 0-1 Modo de escritura. Controla cómo los datos de la CPU son transformados antes de ser escritos a la Video RAM: 0: El Modo 0 trabaja realizando escrituras en las que los datos pueden ser escritos de dos formas, comentadas en la Solo Prog. nº 20. 1: El Modo 1 es usado en transferencias de video a video. 2: El Modo 2 escribe un color a todos los pixels en el byte que escribamos en la Video RAM. El bit 0 es copiado en el plano 0, etc... 3: (VGA) El Modo 3 es usado para rellenar un area con un color o patrón. 3 Modo de lectura: 0: Los datos son leídos desde el plano seleccionado en el Read Map Select Register. (3CEh index 4). 1: Comparación entre los 8 pixels que ocupan el byte de lectura y el color del Color Compare Register (3CEh index 2). 6 Permite 256 colores si está a 1.
06h	Miscellaneous Mode Register. (Read/Write)	Bits: -------- 0 Indica modo gráfico si está a 1. Es un modo alfanumérico si está a 0. 2-3 Modo de memoria: 0: Usa A000h - BFFFh 1: Usa A000h - AFFFh Modos EGA/VGA. 2: Usa B000h - B7FFh Modos monocromos. 3: Usa B800h - BFFFh Modos CGA.

Tabla 3. Registros del Graphics Controller.

registro de polaridad del reloj de la VGA, y es un caso especial porque antes de modificar este registro se debe resetear el secuenciador. Esto se controlaría de la siguiente manera:

```
if ( index == 1 )
{
outport( SEQU_ADDR, 0x0100 );
outport( SEQU_ADDR, valor << 8 | 1 );
outport( SEQU_ADDR, 0x0300 );
}
else
outport( SEQU_ADDR, index | ( valor << 8 ) );
```

El sistema de lectura es igual que el de los tres anteriores sea cual sea el registro al que se desea acceder.

OTROS CONTROLADORES

El resto de los registros o controladores, tanto VGAENABLE_ADDR, STATUS_ADDR y MISC_ADDR como los que no han sido nombrados, se programan de la misma manera, la estándar del PC:

La modificación o lectura consiste en leer o enviar el registro directamente al puerto sin preocuparse del Index Port. Esto se hace cuando un puerto controla un solo registro, por lo que no puede existir ambigüedad sobre a cual nos referimos, que es como decir que ese puerto cumple una sola función. Son puertos de acceso directo porque el

Index Port es ignorado y nos ocupamos únicamente de escribir o leer en el puerto correspondiente. La mayoría de puertos del PC son usados de esta manera, más sencilla y simplificada:

LECTURA: valor = inportb(puerto);
ESCRITURA: outportb(puerto, valor);

En los *listados 1* y *2* podemos ver nuestras nuevas funciones de modificación de registros *RegisterIn* y *RegisterOut*, implementadas en la nueva librería *VGAREGS.H*, a modo de resumen sobre la programación de los diferentes componentes de la VGA. Además disponemos de funciones de lectura y escritura en puertos de manera que no haya que incluir la librería <dos.h> en nuestros programas.

LOS REGISTROS VGA

En las tablas 2, 3, 4 y 5 podemos ver los registros más interesantes en la programación del modo X, el significado de sus bits y el índice y puerto al que están asociados. En este número se incluyen los registros del Attribute Controller, Graphics Controller, VGA Sequencer y Miscellaneous Outport Register, además del STATUS_ADDR. Para el siguiente número dejamos el CRTC Controller, que nos permitirá hacer verdaderas "virguerías" con la VGA gracias a su control sobre todos los parámetros de retrazo, anchura y offset.

A continuación se describirá un registro con el que crearemos una *copper bar* como ejemplo de las posibilidades que nos brinda la programación de la VGA.

EL REGISTRO DE ESTADO STATUS_ADDR

Su descripción puede verse en la tabla 5. El significado de sus bits es sencillo de interpretar y proporciona acceso a los retrazos verticales y horizontales. Sobre el retrazo hay que saber que en el monitor hay un haz de electrones que bombardea el fósforo del monitor de manera que no se apague su tonalidad. Esto ocurre entre 60 y 70 veces por segundo, en las que el haz comienza a leer bytes de la VideoRam y a pintarlos empezando en la esquina (0,0), incluido borde, del monitor. Al final de cada línea horizontal el haz vuelve al principio de la línea siguiente. Este corto lapso de tiempo en que va de una línea a otra se llama *Retrazo Horizontal*.

El *Retrazo Vertical* es el período de tiempo en el que el haz vuelve a (0,0)

SEQUENCER (SEQU_ADDR) PUERTO: 3c4h

ÍNDICE:	NOMBRE:	DESCRIPCIÓN DEL REGISTRO:
01h	Clocking Mode Register. (Read/Write)	Bits: (los más importantes) -------- 0 Si está a 1 los clocks de carácter serán de ancho = 8. 1 (EGA) Si está a 1 el CRTC usará sólo 2 / 5 de los ciclos de reloj. Si es 0, usará 4 / 5. 2 Si es 1, carga otro ciclo de reloj. 3 Si es 1, el Dot Clock es Master Clock / 2. Si 0 usa la misma frec. que Master Clock (3c2h bits 2-3). (Dobla pixels) 4 (VGA) Cada 4 ciclos de reloj carga los serializadores. Si es 0 lo carga cada ciclo. 5 Si es uno deja de refrescar la pantalla y le da a la CPU el control de todos los ciclos de reloj, pues ya no los usa.
02h	Map Mask Register. (Read/Write)	Controla los planos en los que se puede escribir. Podemos desactivar planos para impedir su modificación, etc... Bits: -------- 0 Permite escritura a plano 0 si está activado. 1 Permite escritura a plano 1 si está activado. 2 Permite escritura a plano 2 si está activado. 3 Permite escritura a plano 3 si está activado.
04h	Memory Mode Register. (Read/Write)	Bits: -------- 0 A 1 en modos alfanuméricos. 0 en modos gráficos. 1 Activado si hay más de 64k de memoria en el adaptador. 2 Si está a 1 activa modo de direcciones Odd/Even (impar-par). El modo Odd/Even coloca los bytes impares en los planos 1 y 3, y los pares en los planos 0 y 2. 3 (VGA) Si está activado los bytes 0 y 1 seleccionan planos de video memoria (modo de 256 colores).

Tabla 4. Registros del VGA Sequencer.

MISCELLANEOUS R. (MISC_ADDR) PUERTO: 3c2h

ÍNDICE:	NOMBRE:	DESCRIPCIÓN DEL REGISTRO:
01h	Miscellaneous Output Register. (Read: 3cch Write: 3c2h)	Bits: -------- 0 Si está a 1, emulación de color y Dirección Base = 3Dxh. Si está a 0, emulación Mono y Dirección Base = 3Bxh. 1 Si es 1, permite el acceso de la CPU a la VideoRam. 2-3 Modo de reloj de video: 0: 14 Mhz (EGA) 25 Mhz (VGA) 1: 16 Mhz (EGA) 28 Mhz (VGA) 2: Externo (EGA) Reservado (VGA) 4 (EGA sólo) Desactivar drivers internos si está a 1. 5 Seleccionar bancos de 64k altos si está a 1. 6 Horizontal Sync Polarity, negativa si está a 1. 7 Vertical Sync Polarity, negativa si está a 1. Estos 2 bits (6-7) indican el número de líneas de visualización. 0: 200 (EGA) Reservado (VGA) 1: 400 (VGA) 2: 350 (EGA) 350 (VGA) 3: 480 (VGA)

CRTC STATUS REGISTER (STATUS_ADDR) PUERTO: 3dah

ÍNDICE:	NOMBRE:	DESCRIPCIÓN DEL REGISTRO:
----	Input Status Register (Read only)	Bits (los más importantes): -------- 0 Si está a 1 el retrazo horizontal está en progreso. Si está a 0 el haz está redibujando la pantalla. 3 Si está a 1 está ocurriendo un retrazo vertical. Si está a 0 aún está refrescando la pantalla.

Tabla 5. Miscellaneous OutPut Register y Status Register.

en diagonal desde la esquina inferior derecha tras haber terminado el refresco de todas las líneas de la pantalla y durante el cual los accesos a video memoria no aparecen en pantalla hasta que los redibuje en el siguiente ciclo, cosa que debemos aprovechar para realizar las operaciones gráficas.

Como se puede ver en la tabla, el bit 3 cuando está a 0 nos indica que el haz de electrones está redibujando la pantalla y que no debemos realizar ninguna operación de escritura en VideoRam, por lo que debemos esperar a que esté a 1 para dibujar o volcar bitmaps y gráficos en pantalla.

Lo más práctico que se puede hacer es esperar a que esté a 0 (a que esté redibujando la pantalla), y luego esperar a que acabe de hacerlo (hasta que pase a estar a 1). Si sólo se comprobara esta última condición podríamos encontrarnos con el haz a punto de llegar a (0,0) con el consiguiente parpadeo en las operaciones gráficas. Veamos como sería el código de espera al retrazo vertical:

```
        mov dx, 3dah
vert1:
        in al,dx
        test al,8
        jnz vert1
vert2:
        in al,dx
        test al,8
        jz vert2
```

El retrazo horizontal lo controlamos de igual manera gracias al bit 0 de este registro. Mediante estos dos retrazos ya podemos evitar los parpadeos en nuestras operaciones gráficas y realizar efectos como el que vamos a ver a continuación.

UN EJEMPLO PRÁCTICO: LA COPPER BAR

Las *copper bars* se basan en los retrazos y en el cambio de paleta para cambiar el color del fondo y simular franjas horizontales. Para eso debemos saber cómo esperar un retrazo vertical y horizontal, cosas que ya podemos hacer gracias a nuestros nuevos conocimientos sobre los registros VGA y sobre el STATUS_ADDR.

LISTADO 1

```
unsigned char RegisterIn( int Controlador, char
index )
{
 unsigned char valor;

  switch ( Controlador )
   {
     case MISC_ADDR:
     valor = InPortb(0x3cc);
     break;
    // 0x3c2 es de escritura y 0x3cc de lectura

     case ATTR_ADDR:
InPortb(STATUS_ADDR);  // reseteamos el flip-flop
OutPortb(ATTR_ADDR, index);
valor = InPortb(ATTR_ADDR+1);
break;

case SEQU_ADDR:
case GRAC_ADDR:
case CRTC_ADDR:
case CHIPSTECH_ADDR:
            OutPortb(Controlador, index);
            valor = InPortb(Controlador+1);
            break;

case VGAENABLE_ADDR:
default:    // resto de puertos
     valor = InPortb(Controlador);  break;
   }

 OutPortb( ATTR_ADDR, 0x20 );  // permitir refres-
co VGA
 return(valor);          // devolvemos el valor
}
```

Lectura de los registros VGA.

LISTADO 2

```
void RegisterOut( int Controlador, char index,
unsigned char valor )
{
  switch (Controlador)
    {
      case ATTR_ADDR:
  InPortb(STATUS_ADDR);   // resetear flip-flop
  OutPortb(ATTR_ADDR, index);
  OutPortb(ATTR_ADDR, valor);
  break;

     case SEQU_ADDR:
            if (index == 1)
        {
    OutPortw(SEQU_ADDR, 0x0100);   // resetear
el secuenc.
    OutPortw(SEQU_ADDR, valor<<8 | 1);
    OutPortw(SEQU_ADDR, 0x0300);
    break;
   };
case GRAC_ADDR:
case CRTC_ADDR:
case CHIPSTECH_ADDR:
  OutPortw( Controlador , index | valor<<8);
  break;

case MISC_ADDR:
case VGAENABLE_ADDR:
default:
  OutPortb(Controlador, valor);
  break;
}

OutPortb( ATTR_ADDR, 0x20 );  // permitir refres-
co VGA
}
```

Modificación de los registros VGA.

Para crear la Copper Bar se ha de seguir el siguiente algoritmo, que repitiéndose hasta que se pulse una tecla:

1. Esperar un retrazo vertical para cerciorarnos de que el haz de electrones está situado en la esquina superior izquierda de la pantalla.

2. Si la variable *Y* controla la altura de la barra en la pantalla de texto, esperarse *Y* retrazos horizontales significaría que el haz está situado ahora donde debe estar la primera línea de la barra. Lo hacemos mediante un bucle y pasamos al paso 3.

3. Para cada una de las líneas de la barra, cambiar la paleta del color de fondo con una tonalidad distinta de la anterior de manera que el haz dibuja la línea con el nuevo tono, pero como aún no ha llegado a redibujar la parte superior e inferior de la pantalla, estas líneas siguen con el color negro de fondo.

4. Incrementar o decrementar la variable *Y* según suba o baje la barra.

5. Cuando se hayan acabado todas las líneas de la barra, para que el rayo continue dibujando la pantalla con el color de fondo normal cambiamos la tonalidad para dejarla igual que el color de fondo, por ejemplo negro (0,0,0) y saltamos de nuevo al paso 1.

En los ejemplos del CD tenemos las nuevas funciones que nos serán útiles en cuestiones gráficas: *Ejemplo2.c* contiene la manera de cambiar la paleta utilizando el hardware de la VGA mientras que *Ejemplo3.c* y *Ejemplo4.c* contienen funciones de retrazos verticales y horizontales. El listado *Copperc.c* contiene una versión C de la Copper Bar, más asequible a los programadores que no dominan ensamblador.

EN LA PRÓXIMA ENTREGA

En la próxima entrega terminaremos la programación de los registros de la VGA mediante el CRTC Controller, y prepararemos nuestra librería modox.h para inicializar el modo X con distintas resoluciones de pantalla (1x4, 2x2, 4x1). Se incluirá además información extra sobre todos los demás registros de la VGA y una utilidad para examinarlos con exactitud.