微型计算机高级图形程序设计技巧与实例

EGA VGA

MDA CGA

HGC MCG

AHG C+ Inc olor

大量的C语言和汇编语言实例

中国科学院希望高级电脑技术公司

适用于IBMPC AT 286·386·PS/2及其兼容机

EGA VGA MDA CGA MCGA HGC HGC+ Incolor

微型计算机高级图形程序设计技巧与实例

舒 青 编 译

中国科学院希望高级电脑技术公司

一九九〇年九月

内容简介

本书主要讨论了当前微机最流行的图形卡MDA、CGA、MCGA、HGC、HGC+、InColor、EGA以及VGA卡的性能及相应的程序设计技巧与实例。要求读者具有一定硬件知识以及对C语言和汇编语言有一定的了解。本书给出的例子可直接用于应用程序。

EGA VGA MDA CGA MCGA HGC HGC+ Incolor

微型计算机高级图形程序设计技巧与实例

编译　舒　青

编辑　晓　阳　人　华

■北京市新闻出版局
■准印证号：3152-109152
■订购：北京8721信箱资料部
■邮码：100080
■电话：2562329
■传真：01-2561057
■乘车：320、332、302路车至海淀黄庄下车
■办公地点：希望公司大楼101房间

目　　录

附 录

第一章 IBM图形系统概论

自从1981年IBM PC推出以后，微型计算机有了巨大发展，微机的图形显示系统也越发展越好了。同时市场对具有强大功能的显示系统硬件的需求也增加了。为了满足这些要求，IBM和其它公司已经发展了一些复杂的显示卡和显示器，同时也配套开发一些软件以配合这些硬件。

这一章主要讨论IBM PC和PS/2显示设备的发展概况，包括最常被使用的IBM和Hercules所提供的显示设备。本章也简介IBM Video BIOS，以及一组建立在所有IBM PC和PS/2的ROM中的驱动程序，它提供了图形应用程序基本的程序设计介面。

1.1 IBM PC和PS/2的显示设备

一部IBM PC或PC/AT，或80386系统的主机是没有显示设备的，必须自己来选择并且安装显示设备。最典型配置是：在一台PC中，以一个9针接头的连接线连接屏幕和显示卡。而一个典型的显示卡包含一个连接屏幕的9针接头，以及一个可以插入PC主机板的2×31连接插头。图1.1是IBM许多显示卡的IC排列以及插头位置情形，显示卡上的集成电路可以产生控制屏幕的图形信号。

当您购买PC时，必须决定使用什么显示卡以及显示器。一般最常使用的，以及大部分软件都可以应用的显示卡，是IBM的MDA（单色显示卡）、CGA（彩色图形卡）、EGA（加强型形图形卡），以及Hercules的HGC（单色图形卡）。

所有的IBM PS/2系列的计算机都配备了内建的显示系统，故不必再买一个显示卡。在PS/2 Model 25和30中的显示系统，叫做多色彩图形阵列(Multi-Color Graphics Array)，在PS/2 Model 50、60和80中的显示系统，就是著名的“显示图形阵列”（Video Graphics Array），简称VGA。现已把VGA做成一个卡插在Pc/XT、PC/AT以及80×86系统微机上。

1.1.1 IBMPC单色显示和彩色图形卡

在1981年推出IBM PC时，IBM提供了两种显示卡，一种是“单色显示卡”(简称MDA)，一种是“彩色图形卡”（简称CGA）。MDA是为单色显示器设计的，它可以显示25×80的文数字。CGA可以用在RGB的显示器上（可以显示分离的红，蓝，绿信号），也可以使用在一般家庭中的电视屏幕上（可以显示混合的视频信号）。CGA卡可以显示以点绘制的图形，以及ASCⅡ码字符。

虽然MDA和CGA都可以显示25×80的文字，但大部分的人认为MDA的绿色单色显示对人的眼睛较好，那是因为MDA的单色显示器较其它CGA所用的显示器分辨率要高，MDA的分辨率为720×350，而CGA的分辨率为640×200。

上述两种显示卡显示字符时，都是以点阵形式显示的。您可以算一下，在单色显示器中，每个字符宽是9个象素，高为14个象素；但在CGA显示器中，每个字符长宽只有8个象素。MDA的高分辨率可以产生高质量的字符。故大部人都喜欢看MDA上的文字，而不喜欢CGA上的文字。

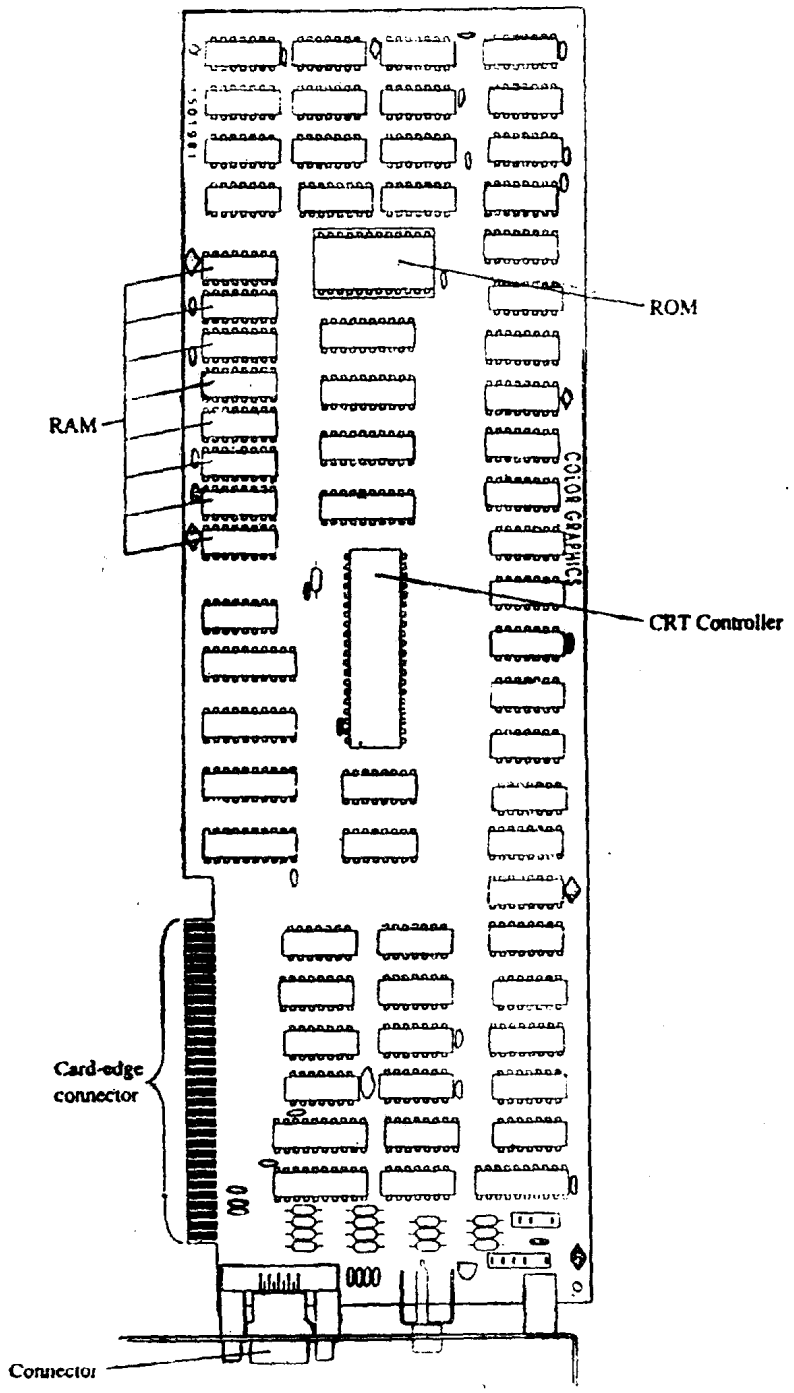

图 1.1 典型的IBM PC显示卡

另一方面，许多使用者除了要显示文本数据外，也需要显示图、图表以及其它图形数据。又因为MDA只能显示文数字，而有时候许多的应用程序也必须在屏幕上显示色彩，所以这些使用者必须折中的选择有彩色绘图功能但文字的清晰度较低的CGA卡了。

为什么不将有高分辨率的单色显示并列CGA卡中，以便两者都能兼顾呢？不幸的是，由MDA产生的图形信号，与驱动CGA显示器所需的信号不相容，显示器与显示卡不匹配会使显示器失去功能，而不会提高它的分辨率。

1.1.2 Hercules图形卡

Hercules解决要在同一个显示器显示清楚的文字和图形的方法，是在单色显示卡上加上绘图的功能。这种单色的HGC（Hercules Graphics Card）在1982年推出，它可以在同一个单色显示器上显示图形和文本数据（HGC除了能显示图形外，还保留原来的MDA的功能）。对许多用户而言，这种混合显示清晰的文字，以及单色绘图的能力是很有用的，而且选择HGC较便宜。因此HGC受到大部分软件公司的支持，HGC已经在市场建立了稳固的地位。

1.1.3 Hercules加强图形卡

HGC+是在1986年推出的，这个等级的HGC卡与原来的HGC卡最大的不同是：它可以显示用户自己设计制作且存在RAM里的文数字字形，而MDA和HGC只能显示事先烧在ROM中的文数字字形。因为文数字字形的显示较按点来画的图形字符要快，所以使用HGC+可以使一个文字的处理速度加快两倍到三倍。

1.1.4 IBM加强型图形卡（EGA）

另一个提供较高文字及绘图分辨率的卡是1985年推出的“加强型图形卡”（Enhanced Graphics Adapter（简称EGA），可以模拟MDA或CGA；EGA之所以称为“加强型”的原因是它可以做一些前面两者所做不到的事。EGA可以在单色显示器上显示以点绘制的图形。另外，EGA比CGA的分辨率要高，它可以产生16色文数字或640×350图形图象。

虽然EGA的分辨率和彩色能力较CGA有一定的提高，但重要的是EGA同时显示的文字和图形远比CGA清晰。因为EGA较便宜，又有许多高级软件充分发挥它的功能，故它成为市场上标准硬件配置。

1.1.5 Hercules InColor卡

Hercudes InColor卡是在1987年推出，它可以说是16色的HGC+。InColor卡完全模拟HGC+，所以在HGC+上可运行的程序，不需修改便可在InColor卡上使用。InColor卡的分辨率同HGC+相同，都是720×348。色彩与EGA相同，它可以从64色调色板(palette)中，同时显示16种颜色。这种卡必须在与EGA相容的彩色显示器上使用，即要求屏幕有350行的垂直分辨率。

不要将InColor Card与Hercules InColor Card混淆，后者是一种加强CGA功能的卡，并且用在HGC或HGC+可用的微机上。

1.1.6 多色彩图形阵列（MCGA）

“多色彩图形阵列”（Multi-Color Graphics Array，简称MCGA）是内建在PS/2 Model25和30上图象系统。MCGA和CGA有许多类似的地方，但MCGA有较高的分辨率（最高为640×480），并且改进了色彩的显示能力。

MCGA和上述几种显示卡的最大不同是，MCGA产生模拟RGB图形信号。数字RGB和

模拟RGB不同的地方，就象是on-off式切换与微调式切换之间的差别。用数字RGB信号的图象显示器，必须分辨某一个特别颜色（红、蓝、绿）的信号是否on或off。但用模拟RGB信号的显示器，会将每一个信号的电压值转换成符合色彩明暗的范围。只有模拟显示器可以和MCGA一起使用。

某些显示器可以使用模拟或数字图形信号。若用对了连接头，这些调成可接收模拟图形信号的显示器就可以接到MCGA上了。

使用模拟图形信号的原因是它可以显示较大范围的颜色。MCGA有一个数字到模拟转换器（DAC），可以增强系统的显示能力，它可以从262，144个颜色调色板（palette）中，同时显示256个颜色。除了模拟彩色显示器外，IBM还提供了一个模拟单色显示器。在单色屏幕上，MCGA可以显示64种灰度。

1.1.7 显示图形阵列（VGA）

Video Graphics Array（简称VGA）这个名称，是特别指PS/2 Model 50、60和80内建的视频系统的一部分。VGA实际上是一个单一的集成片，它的功能相当于EGA上几个片子的组合，而人们通常都将整个视频系统简称为VGA。

VGA的程序设计介面和EGA很相似，所以有许多对EGA写的程序不需修改便可在VGA上运行。VGA的分辨率远比EGA高（文本方式下为720×400，图形方式为640×480）。和MCGA一样，VGA也包含了一个DAC，它可以从262，144个调色板中，同时产生256个颜色。因为VGA可以产生和MCGA相同的模拟RGB信号，所以它必须配备模拟单色或彩色显示器。

1.2 ROM BIOS介面简介

ROM中的一组BIOS（Basic Input /Output System）程序是驻留在每一部IBM PC和PS/2内存中的，ROM BIOS程序提供一个使用硬件特性的介面，包括日期—时间、键盘、软盘和硬盘，以及视频系统，其中视频系统的BIOS程序，包含了一组设计基本视频程序的简单工具，例如：将字字节或字符串写到屏幕上、清除屏幕、改变颜色等等。

虽然ROM BIOS视频系统程序有时很慢，而且不复杂，但是使用它们的程序却可以在IBM PC和PS/2不同视频系统执行，另外大部分与IBM PC相兼容的计算机厂家已经将IBM BIOS的功能驻留在他们的机器中了，所以利用BIOS程序来使用显示硬件的程序具有更好的通用性。

1.2.1 中断10H

BIOS程序是用汇编语言写成的，所以用汇编语言写的程序最容易调用它们。所有的BIOS显示程序都是通过80×86的软中断INT 10H来执行的(80×86是依据Intel8086系列：8086、80286、80386，和80486的微机而来的）。

因此，ROM BIOS的显示介面就是著明的INT 10H介面。ROM BIOS提供了不少的显示功能，每一个功能都通过INT 10H来调用，并且编有号码，在执行INT 10H之前，必须将80×86的寄存器AH设成所要的功能的号码。

当INT执行时，80×86寄存器的值通常是传递给BIOS程序的参数。如果INT 10H把数据传回给程序，它是将数据放在80×86的一个或多个寄存器中，这种以寄存器传递参数的方式，是为汇编语言所写的程序用的。需知道如何使用INT 10H介面，请参考表1.1汇编语言程序SetVmode( )，这个程序可以和MicrosoftC程序连接，在程序名称前面的底线，

PROC中的关键字near，以及使用堆栈来传递参数，都是MicrosoftC的规定。这个程序的重点是调用ROM BIOS去设置显示硬件成为指定的显示方式。这些操作在第二章及附录A中有详细说明。

表 1.1 SetVmode( )

```
                TITLE       'Listing 1-1'
                NAME        setVmode
                PAGE        55,132
;
; Name,         SetVmode
;
; Function,     Call IBM ROM BIOS to set a video display mode.
;
; Caller,       Microsoft C.
;
;                           void setVmode(n);
;
;                           int ni ;              /* video mode */
;
ARGn            EQU         byte ptr [bp+4]; stack frame addressing
  EQUIP_FLAG    EQU         byte ptr ds:[10h]
  CCAbits       EQU         00100000b       ; bits for EQUIP_FLAG
  MDAbits       EQU         00110000b
  _TEXT         SEGMENT     byte public 'CODE'
                ASSUME      cs:_TEXT
                PUBLIC      _SetVmode
  _Setvmoee     PPOC        near
                push        bp              ; preserve caller registers
                mov         bp,sp
                push        ds
                mov         ax,40h
                mov         ds,ax           ; DS ->video Display Data Area
                mov         bl,CGAbits      ; BL:=bits indicating presence of CGA
                mov         al,ARGn         ; AL:=desired video mode numbor
                mov         ah,al           ; test if desired mode is monochrome
                and         ah, 7
                cmp         ah, 7
                jne         L01             ; jump if desired mode not 7 or 0Fh
                mov         bl,MDAbits      ; BL:=bits indicating presence of MDA
  L01,          and         EQUIP_FLAG,11001111b
                or          EQUIP_FLAG,bl; set bits in EQUIP_FLAG
                xor         ah,ah           ; AH:=0 (INT 10h function number)
                push        bp
                int         10h             ; call ROM BIOS to set the video mode
                pop         bp
```

```
                ???         s                   ; restore caller registers and return
                mov         sp,bp
                ret
                pop         bp
                rec
_Setvmode       ENDP
_TEXT           ENDS
                END
```

实际上要调用VideoBIOS是非常简单的，首先将所要调用的功能号码放在AH中，然后在堆栈中保存BP寄存器内容，这个程序便可以通过INT 10H来调用ROM BIOS了。

表1.2是一个Get Vmode( )的程序。它是向BIOS询问目前的显示模式，方法是通过执行INT 10H的OFH功能，ROM BIOS就会把当前模式号码放在寄存器AL中。Get Vmode( )会把号码传给调用它的程序。

表 1.2 Get Vmode( )

```
                TITLE       'Listing 1-2'
                NAME        GetVmode
                PAGE        55,132
;
; Name:         Getvmode
;
; Function:     Call IBM ROM BIOS to set a video display mode.
;
; Caller:       Microsoft C:
;
;                           int     GetVmode( );
;
_TEXT           SEGMENT byte public 'CODE'
                ASSUME      cs:_TEXT
                PUELIC      _GetVmode
_GetVmode       PROC        near
                push        bp              ; preserve caller registers
                mov         bp,sp
                mov         ah,0Fh          ; AH:=0Fh (INT 10h function number)
                push        bp
                int         10h             ; call ROM BIOS to get video mode number.
                pop         bp
                xor         ah,ah           ;     AX:=video mode number
                mov         sp,bp
                pop         bp
                ret
_GetVmode       ENDP
_TEXT           ENDS
                END
```

1.2.2 视频显示数据区

表1.1中，在实际调用BIOS之前，会修改某些表示状态的参数（global variable），这些参数会被所有的ROM BIOS显示程序更新及引用。在IBM的技术手册中．这些参数被收集到RAM的一个称为“视频显示数据区”中。这个数据区包含了两个部分，第一个部分是在内存0040:0049和0040:0066之间的位置，第二个部分是在内存0040:0084和0040:008A之间。某些Video BIOS程序也引用在0040:0010的2-bit区域（在IBM技术手册是称为EQUIP-FLAG）。其中，bit4和bit5指定了开机时所要使用的系统隐含的显示方式。SetVmode（）则更改bit区域，以便执行所选择的图形方式。例如，若在MDA中选择所需要的显示方式，则EQUIP-FLAG的bit区域就必须适当的更改。

1.2.3 用高级语言来使用Video BIOS

为了使高级语言程序能够使用ROM BIOS，必须通过汇编语言程序(如SetVmodel()或Get Vmode（））的调用来完成。表1.3和表1.4都是简短的C程序，它可以象MS-DOS指令般地被执行。表1.3的程序是调用SetVmode（）来选择显示模式的，这个程序可在批处理文件中执行，也可以在交互方式下执行。表1.4的程序是调用Get Vmode（），采用可以在批处理文件使用的方法传回显示方式号码(也就是用IF ERRORLEVEL==的方法)。

表 1.3 使用SetVmode（）的C程序

```
/* Listing 1-3 */
main (argc,argv)
int       argc;
char      **argv;
{
          int    ModeNumber;
          void   SetVmode ( );
          if (argc  1=2)                    /* verifY command line syntax */
          {
           printf ( "\nSyntax:  SETVMODE n\n" );
           exit (1);
          }
          sscanf (argv[1],  "%x" , &ModeNumber);   /* get desired mode number */
          SetVmode (ModeNumber);                   /* call ROM BIOS via INT 10h */
}
```

表 1.4 使用GetVmode（）的C程序

```
/* Listing 1-4 */
main ( )
{
int           GetVmode ( );
return (GetVmode());
  }
```

设计一个这种程序的过程是：将C程序编译，使其成为一个OBJ文件，再将汇编程序编译成目标文件，即OBJ文件，最后用LINK将两个OBJ文件连成一个可执行的文件。若表1.3

的程序是在一个名叫SM·C的文件中，表1.1的汇编程序是在SETVMODE.ASM中，则可用下列步骤建立一个SM·EXE文件。

若用Microsoft C4.0：

```
msc        sm
masm       sevmode;
link       sm+sevmode
```

若用Microsoft C5.0

```
CL /C      Sm·C
masm       sevmode
link       sm+sevmode
```

对Microsoft C5.0可直接调用库函数—setvideomode( )和getvideomode( )来完成。另外有些高级语言的编译器可以产生适当的OBJ码，来载入80×86寄存器，这时或执行INT 10H，或从寄存器中拷贝结果到调用的程序中。若您的编译器有这些功能，则您可以直接使用INT 10H介面，而不用连接汇编语言程序。例如表1.5中使用了Microsoft C的int86( )功能来获取当前显示方式的。

表 1.5 Microsoft C的int 86( )功能

```
/* Listing 1-5 */
 #include    "dos.h"
main()
{
      struct  BYTEREGS regs;      /* BYTEREGS defined in dos.h*/
      regs.ah=0XOF;               /* AH=0XOF (ROM BIOS funotion numoer) */
      int86(0×10, &regs, &regs);  /* perform interrupt 10h */
      return((int)regs.al);
```

第二章 硬件程序设计

本章将依照程序设计者的需要，介绍IBM PC和PS/2的图形硬件。包括：图形系统可编程部分，程序设计和硬件间的如何配合，以及在改变显示方式时，要做那些设置。以后几章中所用到的程序设计技巧，都是以本章的内容为基础的。

本章的目的就是要解开硬件程序设计介面之迷。因为一般的程序设计者，在他们硬件介面的程序设计中，几乎都依赖Video BIOS所提供的功能。所以软件和视频系统硬件的介面，似乎一直有神秘的气氛。当然，在您学了本章之后，可能宁愿它还是个迷，但是只要您懂的越多，您就越懂得怎样去使用视频硬件。

2.1 IBM PC和PS/2视频系统组成

当您要写一个和IBM视频硬件有关的程序时，假如您能了解IBM视频系统各部分组成之间的关系（如图2.1），那对您是非常有用的。您不需要象设计硬件电路的人一样了解的那么深入，只要知道您的程序该在何处，以及如何和视频硬件交互（interact），以便高效率的产生视频输出就可以了。

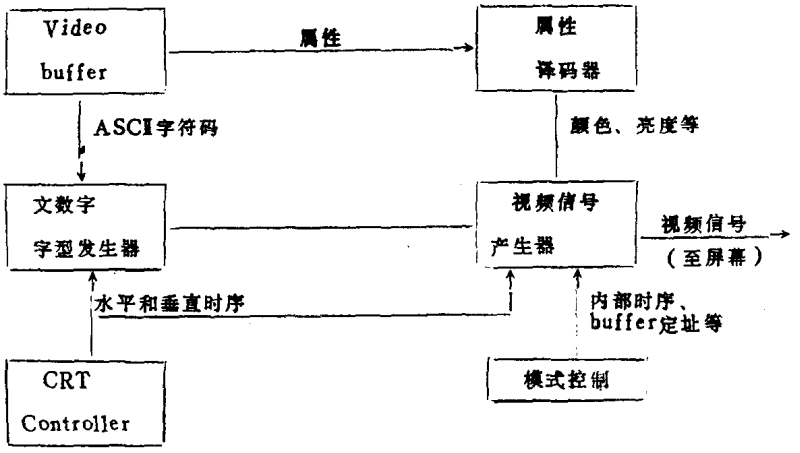

图 2.1 IBM PC和PS/2视频系统中可编程部分（Video buffer、属性控制器等）。本书提到的视频系统中，某些或全部的组成部分是可以用软件控制的。

2.1.1 显示器

在微机的视频硬件中，最显而易见的就是显示器。显示器上是没有哪个部分可以直接编程的。微机的视频系统才是可编程的，它所产生的信号则显示在屏幕上。

本书中提到的显示器，都是raster-scan装置。在显示器屏幕上的图象，实际上是由一条条称为raster（扫描线）的水平线紧密的排列在一起所形成的。电子束由屏幕的左上角开始，从左到右一行行循序地扫描。当每行扫描过后，这行中数百点象素的颜色和亮度都会改变，所有的扫描线组成一个完整的图象。

在彩色的显示器上，一条电子束实际上是包含了三道不同的电子束，各控制一种原色（红、绿、蓝）中的一种。屏幕上的每一点，实际是由一组红蓝绿的荧光所组成。这三道电子束会经过mask，使得每一道电子束各射在一个原色的荧光上。因此，扫描在这组合点上的电子束，其相对强度就决定了象素的颜色和亮度。所以，除非您用磁化眼镜或是靠的很近的看，您不会觉得红、蓝、绿点是分开的，而是看到一个混合的颜色。

2.1.2 Video Buffer

所谓Video Buffer（视频缓冲区）是指在视频系统中，一块存放显示数据的RAM。这块RAM是放在CPU可以定址到的地方，所以也可以象其它部分的RAM一样在程式中读取和写入。

视频系统的显示电路，会连续重复的读取Video Buffer中的数据，来更新屏幕的内容。在Video中的一个或一组bit，可以表示出屏幕上某一个位置的颜色和亮度。屏幕会以每秒50到70次的速度更新。所以当更改Video Buffer中的内容时，屏幕上也几乎是马上跟着改变的。

不同的视频系统，在RAM中可作为Video Buffer的空间也不一样。大部分IBM视频系统的Video Buffer，可以储存大于一个屏幕的数据，因此在任何时候都只有部份的Video buffer可以显示在屏幕上。

2.1.3 色彩和字符的显示硬件

所有IBM的视频系统都结合了一些可以在Video Buffer中读取和译码的硬件。例如，文数字字型发生器将Video Buffer中的ASCII码转换成屏幕所要显示的点阵。而属性译码

器则将Video Buffer中其它数据转换成可以产生彩色、底线等的信号。这些硬件和其它形式图形系统中的一些特殊硬件都能用软件来控制。在以后几章中将深入讨论这些程序。

2.1.4 CRT控制器

CRT控制器（简称CRTC）会产生水平和垂直时序的信号。并以和时序信号同步的速度递增Video Buffer地址计数器（address counter）的值。频频显示电路会依CRTC的地址值，由Video Buffer读取数据、译码，并随着CRTC的时序信号将译码后的色彩和亮度信号送到显示器上。所以CRTC会将Video Buffer中的数据显示和驱动视频显示的时序信号同步。

CRTC还执行了很多其它功能，包括决定由硬件产生的光标的大小和显示位置，选择要显示的Video Buffer，产生硬件底线，以及检测光笔的信号。

在MDA、CGA和Hercules卡上，CRTC是一个单片——Motorola 6845。在EGA上，CRTC是一个由IBM设计的custom LSI片子。在MCGA上，CRTC是Memory Controller Gate Array的一部分。VGA的CRTC，则是Video Graphics Array的一部分。无论这些硬件是如何做的，这些视频系统的CRTC都可由程序控制而产生不同的时序参数。在深入讨论CRTC的程序设计技巧之前，我们先来看看CRTC的时序信号是如何控制raster-scan图像的显示。

2.2 显示更新周期

视频画面是以每秒50到70次的速度，更新的，依所配备的视频系统而定。在每次的更新周期中，电子束是以“Z”字型方式，从屏幕的左上角开始扫描（如图2.2）。当由左至右扫描了一条线之后，会折返到下一条线的起点，一直到整个画面都扫描完了为止，然后再回到屏幕的左上角。这样的过程会一直重复下去。

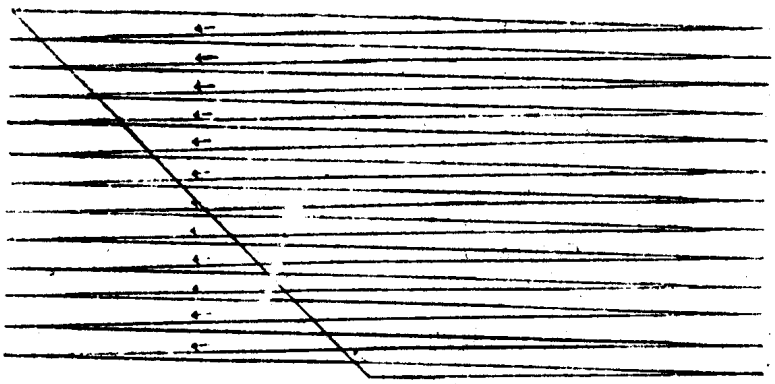

图 2.2 电子束扫描的路径

2.2.1 水平时序

在电子束扫描过画面的同时，会有一连串的事情发生。在每一行开始扫描的时候，CRTC会产生一个Display Enable的信号，让电子束开始动作。当电子束由左至右扫过一行时，视频显示电路会利用CRTC的地址计数器，由Video Buffer中读出一串byte数据。这些数据在被译码后，则用来控制显示器的色彩和亮度。也就是当电子束扫描过画面时，色彩和亮度都会跟着这些信号改变。

在接近屏幕的右端时，CRTC会将Display Enable信号设成off，Video Buffer中的数据就不再显示了。这时CRTC随即产生一个水平同步信号，使得电子束向左和向下移，折返到下一条水平线的起点。然后CRTC再把Display Enable设成on，显示下一条扫描线。

在显示完一行数据资料和下一行资料之间的短暂时间，称为水平空白区间（blanking internal）。因为水平的retrace interval（电子束折返到下一行所需的时间）比blanking internal来的短，所以每条线的两端都会有段水平的空白——Overscan产生（如图2.3）。

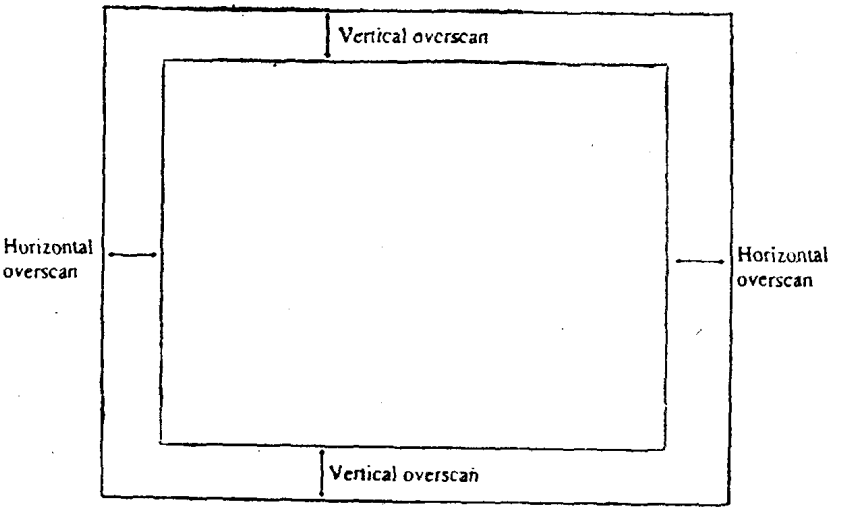

图 2.3 Overscan

在水平的Overscan期间，电子束可以是亮的，用来显示Overscan、边界或是颜色。但是设计这个Overscan最主要的原因是要用来调整边界的，让扫描线都能调整到边界的中央，这样就不会有数据被屏幕截掉了。

2.2.2 垂直时序

当所有的水平线都扫描完之后，Display Enable信号会被设成off CRTC会产生一个垂直同步信号，它会通知显示器，让电子束由屏幕最下面回到左上角。因为垂直的retrace interval（电子束由低端回到顶端的时间）比垂直的blanking interval（此时，Video Buffer中数据未被显示）来的短，所以在屏幕的顶端和底端都会有一段空白——Overscan（图2.3）。和水平的Overscan一样，垂直的Overscan也是用来调整画面，让内容调整在屏幕中央的。

2.3 CRT控制器的程序设计

CRTC的程序设计介面已经被明确定义，而且也很容易使用。所有IBM PC和PS/2所用的程序设计方法，都是相似的。

2.3.1 MDA

单色显示卡（Monochrome Display Adapter）的CRTC，Motorola 6845，有19个8-bit的专用数据寄存器。每个寄存器的值各控制由6845所产生的不同时序信号（如图2.4）。其中有一个是地址寄存器，它的值则指出其它的18个寄存器中的那一个可以被读写。大部分