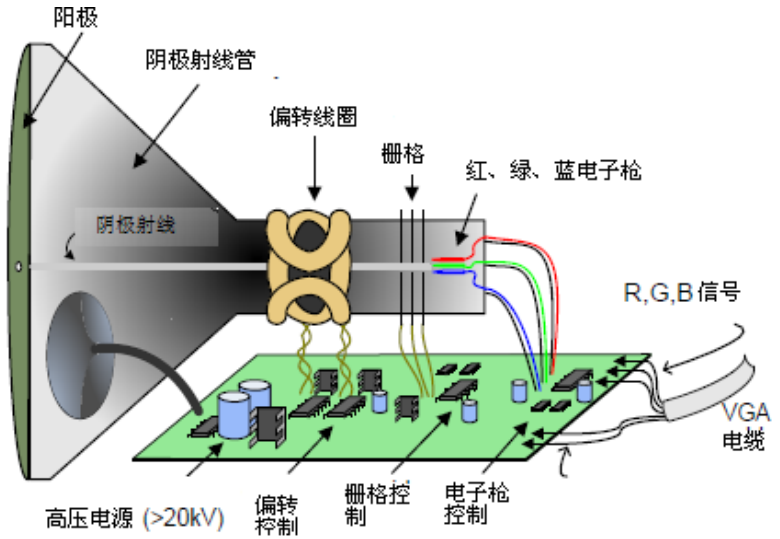
第九章 人机接口

学习目标

- 了解显示器、键盘、鼠标的基本工作原理
- 了解VGA接口基本原理
- 了解PS2接口工作原理
- 掌握VGA接口设计
- 掌握显示器图形以及字符显示原理，并能编程控制显示不同的图形以及字符
- 掌握键盘接口程序设计
- 掌握鼠标接口程序设计

显示器工作原理简介

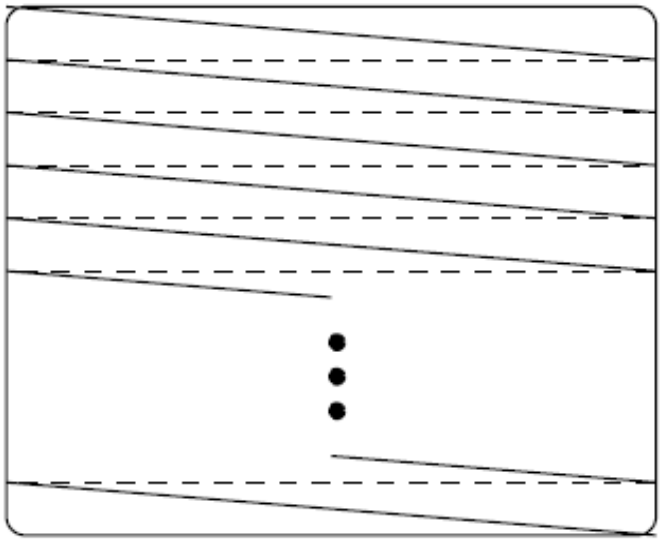
CRT显示器显示原理



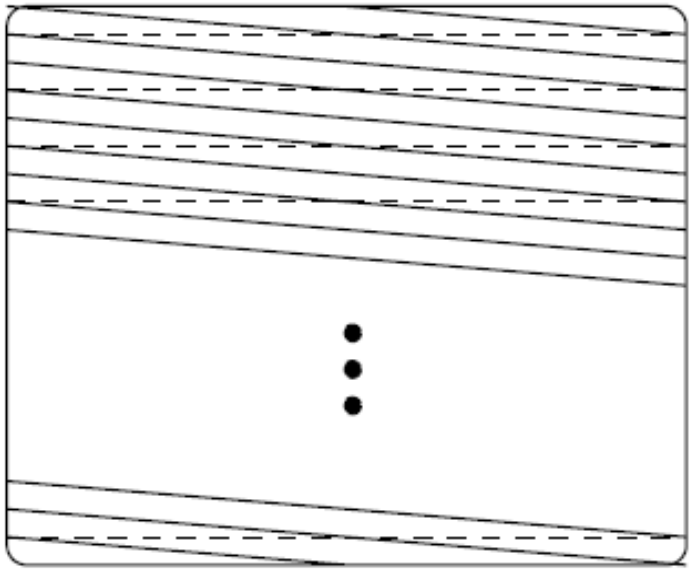
LCD显示器显示原理

在彩色LCD面板中，每一个像素都是由三个液晶单元格构成，其中每一个单元格前面都分别有红色，绿色，或蓝色的过滤器。

显示器成像原理

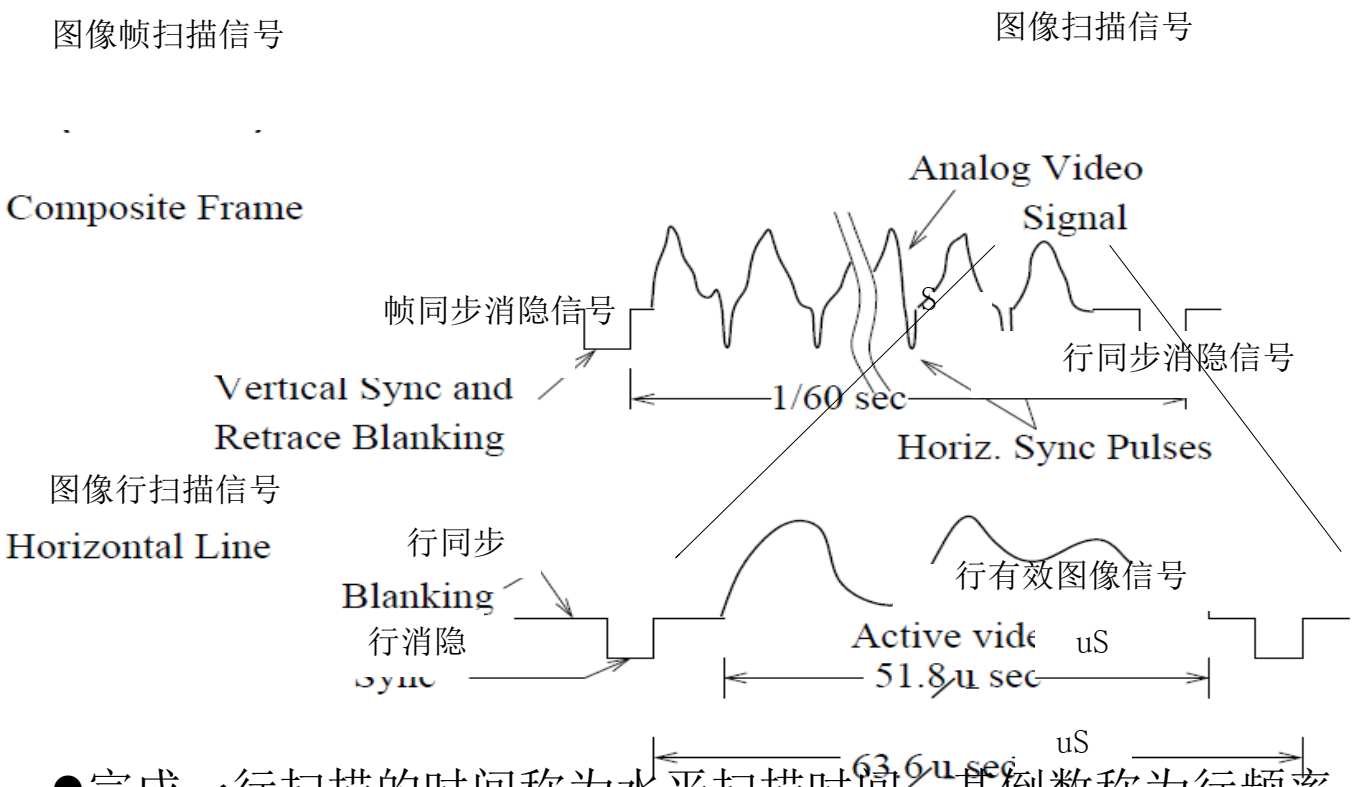


逐行扫描



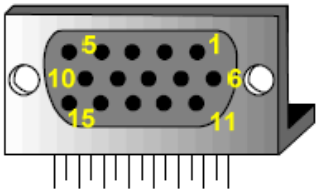
隔行扫描

显示器扫描信号



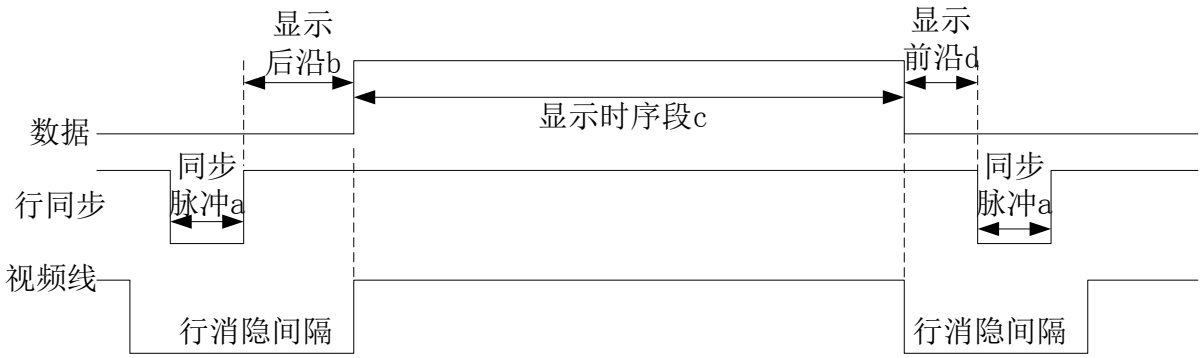
- 完成一行扫描的时间称为水平扫描时间，其倒数称为行频率；
- 完成一帧（整屏）扫描的时间称为垂直扫描时间，其倒数称为场频率

VGA接口标准



Pin 1: Red
Pin 2: Grn
Pin 3: Blue
Pin 13: HS
Pin 14: VS
Pin 5: GND
Pin 6: Red GND
Pin 7: Grn GND
Pin 8: Blu GND
Pin 10: Sync GND

VGA行时序

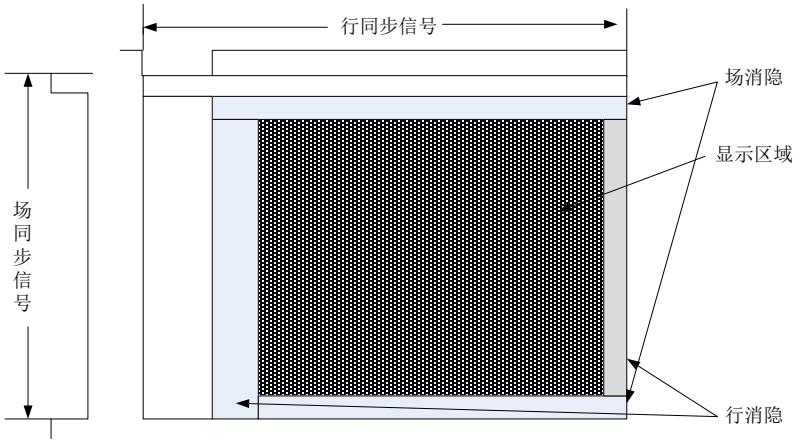


分辨率	信号类型	同步脉冲a	显示后沿b	显示时序c	显示前沿d	一个周期
800*600	行同步	128像素	88像素	800像素	40像素	1056像素
	场同步	4行	23行	600行	1行	628行
640*480	行同步	96像素	48像素	640像素	16像素	800像素
	场同步	2行	29行	480行	10行	521行

行、场同步信号与显示区域之间的关系

- 如果显示器采用逐行扫描方式60Hz的刷新频率，且分辨率为640*480，那么控制每个像素的显示时间周期为：，因此像素的时钟频率为25MHz。

$$T_{clk} = \frac{1}{60 * 800 * 521} = 0.04us$$

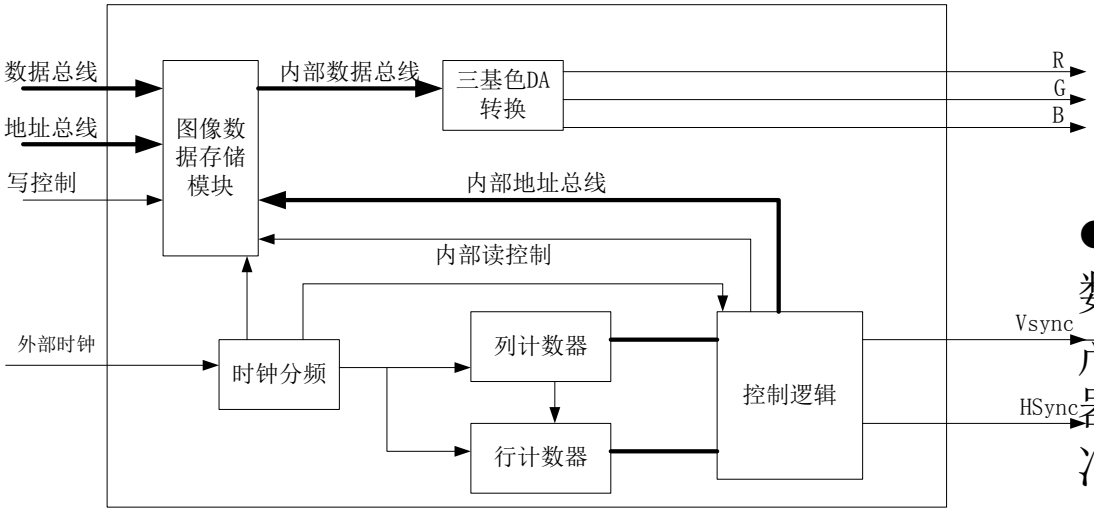


- 如果显示器采用逐行扫描方式要求60Hz的刷新频率，且分辨率为800*600，那么控制每个像素的显示时间周期为：，因此像素的时钟频率为40MHz。

$$T_{clk} = \frac{1}{60 * 1056 * 628} = 0.025us$$

如果每个像素采用8位来表示颜色信息，则要求显示控制器与显示存储器之间的带宽为：$f_{clk} * 8$

简单VGA显示控制器设计



●显示器分辨率为640*480，每个像素的颜色信息采用8位来描述，那么该图像数据存储区的大小则为640*480个字节，即307200个字节存储空间。

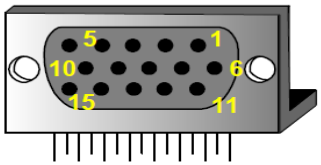
●分辨率640*480，列计数器每计数800个时钟脉冲时复位，并同时产生一个脉冲给行计数器，行计数器每计数521个列计数器送来的脉冲时复位。

●列计数器和行计数器同时送出计数输出给控制逻辑单元。当列计数器的输出在0~95之间时，HSync输出低电平；列计数器的输出在96~799之间时，HSync输出高电平。当行计数器的输出在0~1之间时，VSync输出低电平；行计数器的输出在2~520之间时，VSync输出高电平。

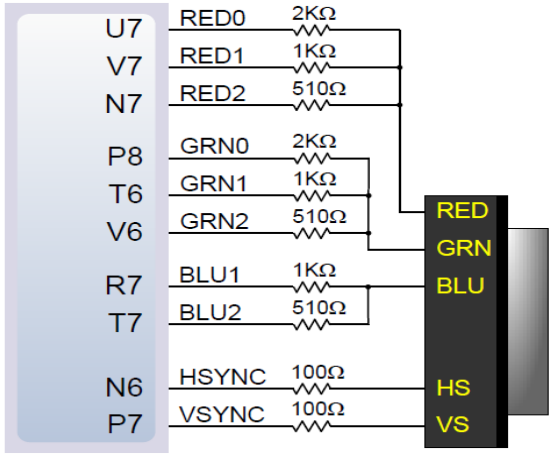
- 只有当列计数器计数范围在144~783之间，且行计数器计数范围在30~429之间时，显示器才能显示图像，内部地址才能发生改变，
- 地址的变化规律与颜色信息的位数相关。
  - 如果8位颜色信息表示一个像素，每个时钟作用下存储器地址都需要增加1；
  - 如果16位颜色信息表示一个像素，每个时钟作用下存储器地址都需要增加2；
  - 如果4位颜色信息表示一个像素，每两个时钟作用下存储器地址才需要增加1。

VGA控制器设计举例

- 试基于FPGA verilog语言设计该VGA显示接口控制器，要求输出VGA标准为640*480@60Hz，颜色信息为8位。



Pin 1: Red　　Pin 5: GND
Pin 2: Grn　　Pin 6: Red GND
Pin 3: Blue　　Pin 7: Grn GND
Pin 13: HS　　Pin 8: Blu GND
Pin 14: VS　　Pin 10: Sync GND



行、列计数器采用verilog语言描述

```
module counter(
input clk_25,
   output [9:0] hcnt,//列计数器
   output [9:0] vcnt,//行计数器
input rst
   );
reg [9:0] hcnt;
reg [9:0] vcnt;
always @(posedge clk_25 )
begin
                if(rst==1)//复位时清除计数器
                begin
                hcnt<=0;
                vcnt<=0;
                end
                else begin
                if(hcnt<799)
                                hcnt<=hcnt+1;
                                else begin
                                hcnt<=0;
                                if (vcnt<520)
                                                vcnt<=vcnt+1;
                                                else
                                                vcnt<=0;
                                                end
                end
end
endmodule
```

行、场同步信号verilog语言描述

```
module sync(input [9:0] hcnt,input [9:0] vcnt,output hsync,output vsync,output [14:0] addr,input clk,input rst  );
reg hsync;
reg vsync;
reg [14:0] addr;
always @(posedge clk )
begin
      if(hcnt>=96)
                hsync<=1;
                else
                hsync<=0;
end
always @(posedge clk )
begin
      if(vcnt>1)
                vsync<=1;
                else
                vsync<=0;
end
endmodule
```

地址生成逻辑verilog语言描述

- RAM存储区大小为160*120B，即采用RAM存储单元内的一个字节对应显示器上一个4*4的像素块，那么RAM地址只需要160*120=19200个不同的地址，因此只需要14位地址引脚

```
always @(posedge clk  )
begin
            if ((rst==1) | (vcnt<31))
                        addr<=14'h00000;
else if ((vcnt>30) & (vcnt<511) & (hcnt>143) & (hcnt<784) & (hcnt[1:0]==2'b11) )
                                    if (addr>=19199)
                                    addr<=14'h00000;
else if ((hcnt==783) & ((vcnt[1:0]==2'b00) |(vcnt[1:0]==2'b01)|(vcnt[1:0]==2'b11)))
                                                addr<=addr-159;
                                                else
                                    addr<=addr+1;
end
```

- 消隐期间RGB引脚必须输出0电平，只有在正常显示区域才显示RAM内的图像数据

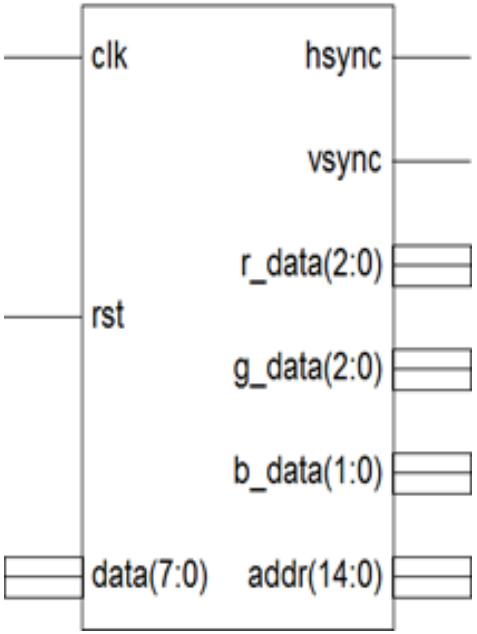
```
always @(posedge clk_25) begin
        if (rst==1)
                datatemp<= 8'h00;
                else if ((vcnt>30) & (vcnt<511) & (hcnt>143) & (hcnt<784))
                        datatemp<=data;//来自RAM的数据输出端
                        else datatemp<=8'h00;//非显示区域必须把RGB赋0
        end
```

RAM存储区内字节表示的颜色信息格式

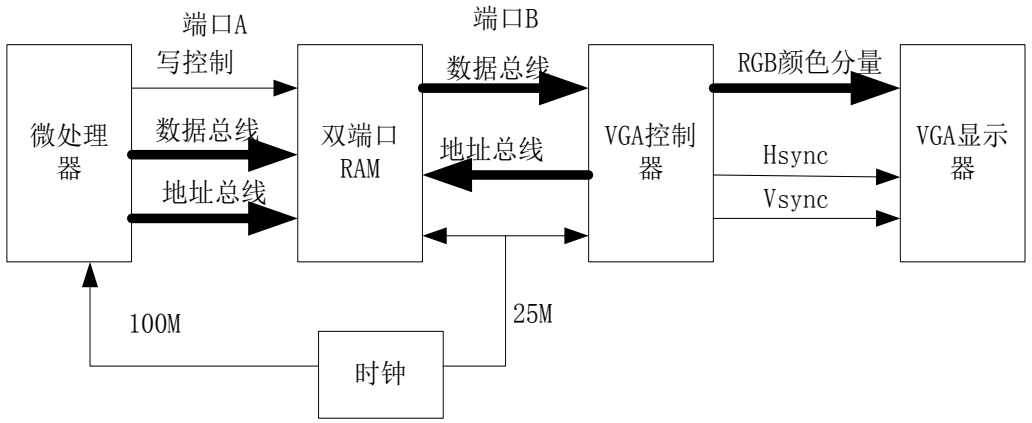
B1	B0	G2	G1	G0	R2	R1	R0

```
ssign r_data[2:0]=datatemp[2:0];//红色分量
assign g_data[2:0]=datatemp[5:3];//绿色分量
assign b_data[1:0]=datatemp[7:6];//蓝色分量
```

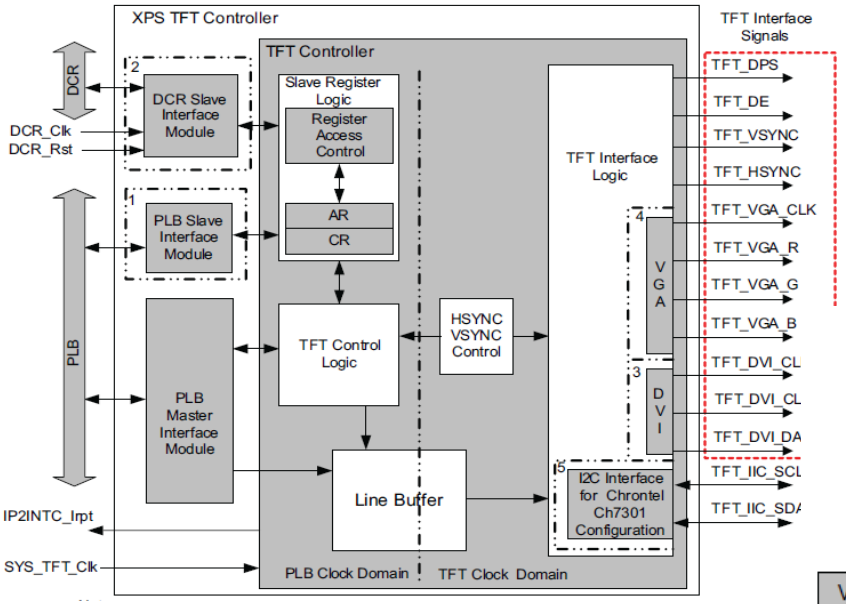
VGA控制器外部引脚结构



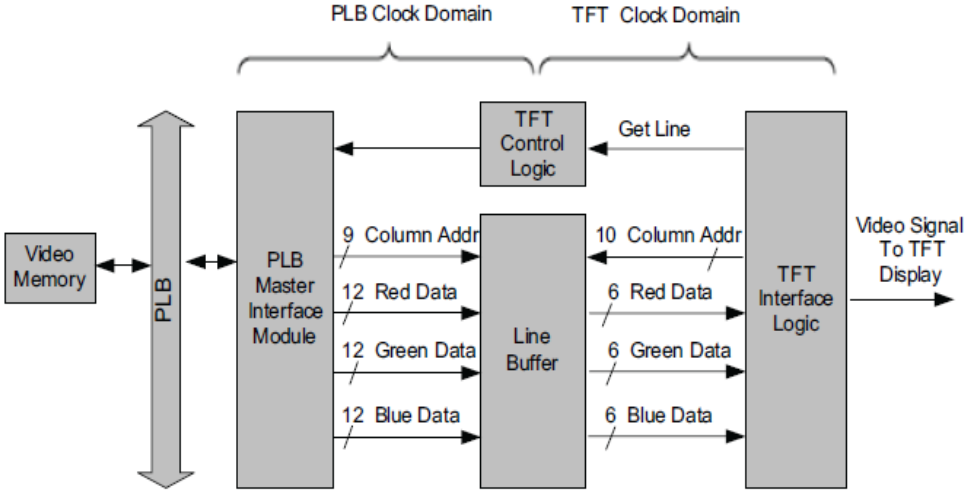
简单VGA控制器应用案例



Xilinx XPS TFT显示控制器简介



像素内存地址偏移	数据位	含义
（行*1024+列）*4 （每行1024个像素，每个像素4个字节）	[0:7]	未定义
	[8:13]	红色
	[14:15]	未定义
	[16:21]	绿色
	[22:23]	未定义
	[24:29]	蓝色
	[30:31]	未定义



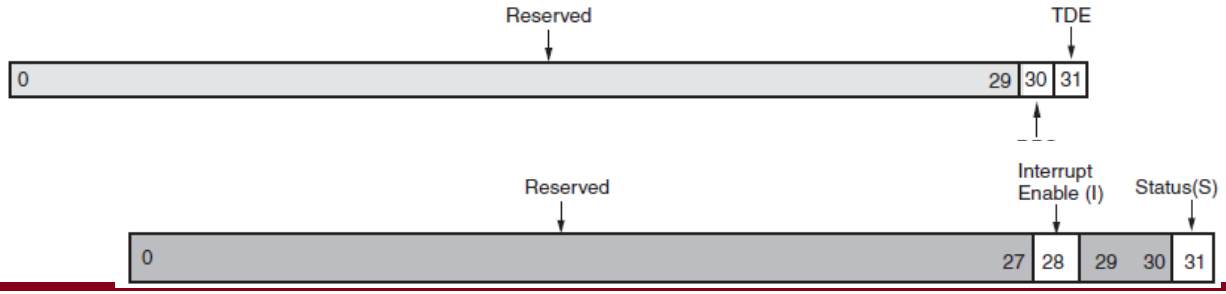
编程控制

寄存器名称	偏移地址	含义
AR	0	显示存储器基地址
CR	4	显示属性控制
IESR	8	VSYNC中断使能标志、中断状态标志

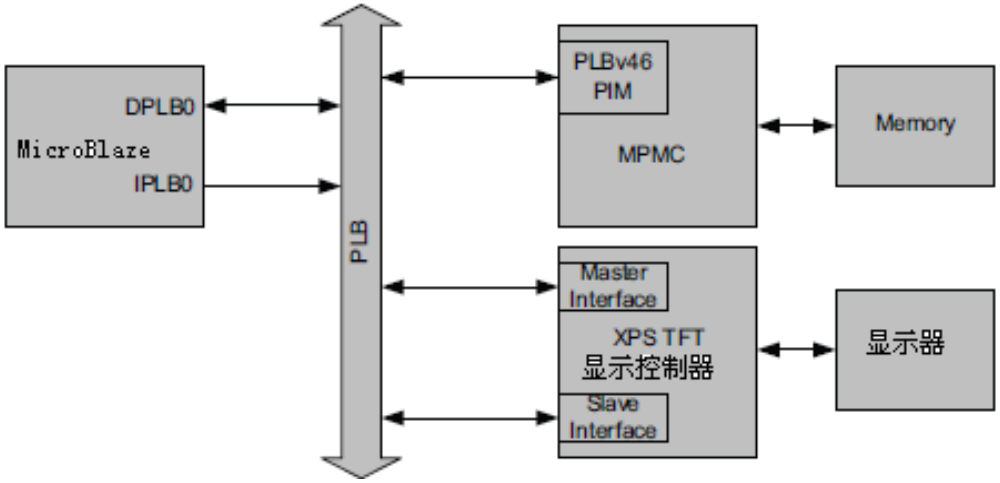
- AR寄存器仅需存储显示存储器的高11位地址，低21位为0

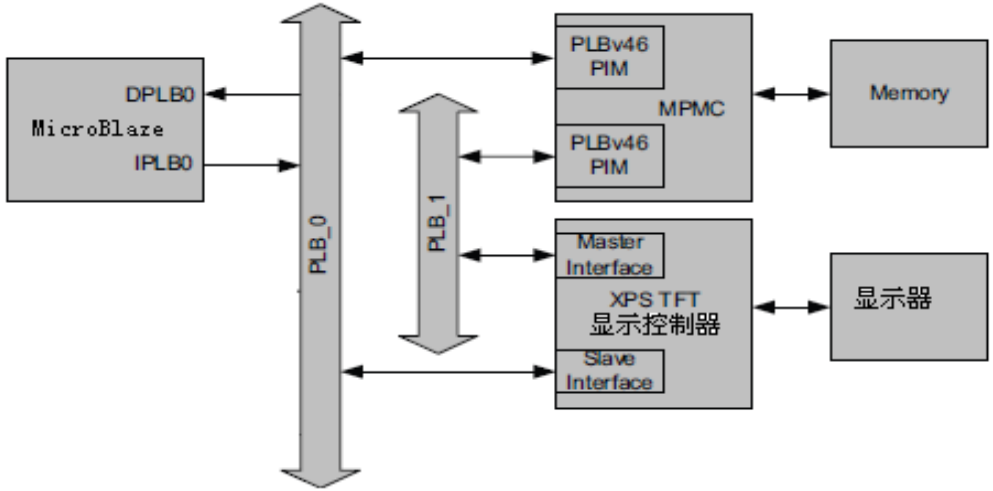
- IE为1表示中断使能，IS为1表示产生了中断

- TDE表示控制显示还是关闭，1表示正常显示，0表示关闭显示；DPS表示控制扫描方式，0正常扫描（从左到右），1表示反方向的扫描（屏幕旋转180^0），



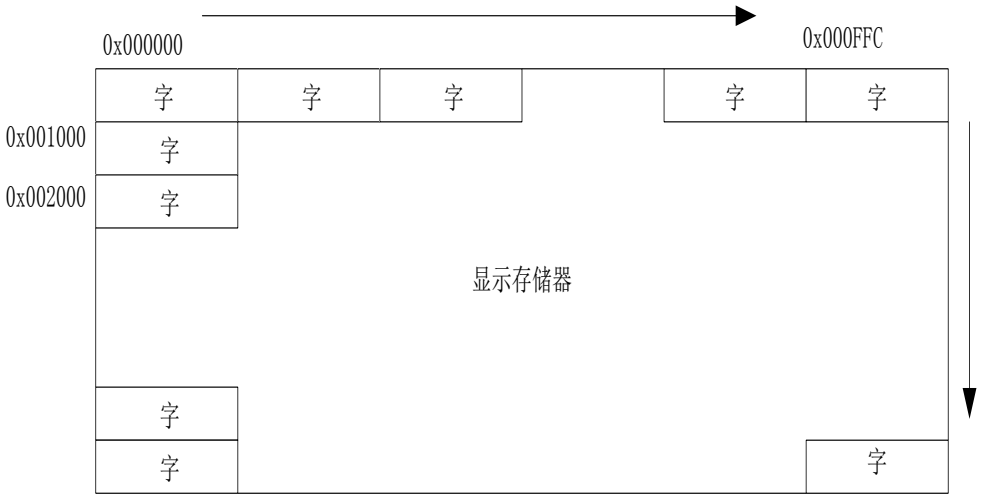
使用案例

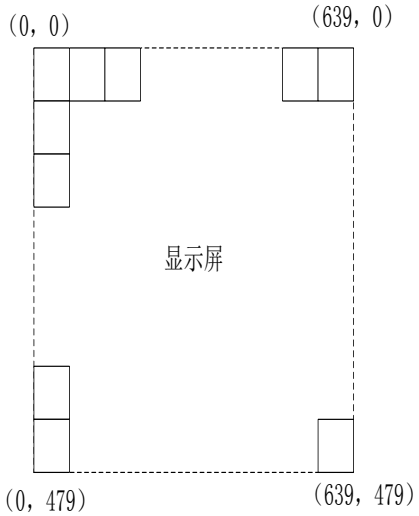

DPLB0
MicroBlaze
IPLB0
PLB
PLBv46
PIM
MPMC
Memory
Master
Interface
XPS TFT
显示控制器
Slave
Interface
显示器


DPLB0
MicroBlaze
IPLB0
PLB_0
PLB_1
PLBv46
PIM
MPMC
PLBv46
PIM
Memory
Master
Interface
XPS TFT
显示控制器
Slave
Interface
显示器

VGA图像及字符显示编程控制

- 显示存储器空间与显示屏幕像素之间的对应关系



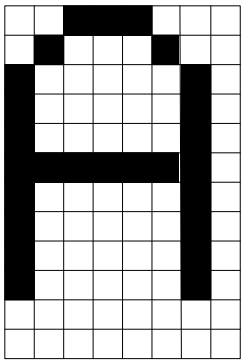


- 将坐标为（3，4）的像素写入红色,就要对地址单元（显示存储器基地址+4099）*4的内存单元写入字数据0x00FF0000
  - 4099=1024*4(行)+3(列)
- 在坐标（3，4）到坐标（400，4）之间画一条绿色的直线

```
For(int i=3;i<=400;i++)
Xil_Out32(VideoBase+(4*1024+i)*4,0x0000FF00);
```

字符显示控制

- 12*8



```
{
 GenPixels( 0, 0, 1, 1, 1, 0, 0, 0),
 GenPixels( 0, 1, 0, 0, 0, 1, 0, 0),
 GenPixels( 1, 0, 0, 0, 0, 0, 1, 0),
 GenPixels( 1, 0, 0, 0, 0, 0, 1, 0),
 GenPixels( 1, 0, 0, 0, 0, 0, 1, 0),
 GenPixels( 1, 1, 1, 1, 1, 1, 1, 0),
 GenPixels( 1, 0, 0, 0, 0, 0, 1, 0),
 GenPixels( 1, 0, 0, 0, 0, 0, 1, 0),
 GenPixels( 1, 0, 0, 0, 0, 0, 1, 0),
 GenPixels( 1, 0, 0, 0, 0, 0, 1, 0),
 GenPixels( 0, 0, 0, 0, 0, 0, 0, 0),
 GenPixels( 0, 0, 0, 0, 0, 0, 0, 0)
},
```

- 字模

```
#define  GenPixels(a7, a6, a5, a4, a3, a2, a1, a0) ( ((a7) << 7) |\
        ((a6) << 6) | ((a5) << 5) | ((a4) << 4) | ((a3) << 3) | \
                ((a2) << 2) | ((a1) << 1) | (a0)   )//一行的像素表示
```

图像显示

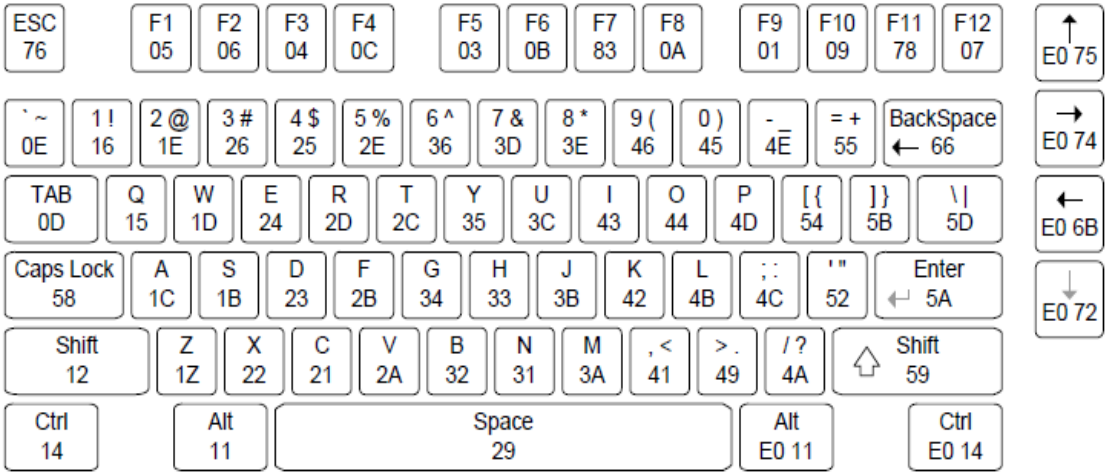
- 编程控制显示器显示一副分辨率为640*480，颜色为24位的位图，且该位图的数据表示形式为：蓝色、绿色、红色各8位表示一个像素。假设表示该位图的数组首地址为gImage_24caise。
- 由于位图分辨率为640*480，颜色为24位，因此表示该位图图像数据的数组为640*480*3=921600个字节。
- 定义为const unsigned char gImage_24caise[921600]，

```
void TftDrawPicture(int videobase,unsignedchar *p)
{
      int i,j;
      u32 PixelVal;
      u32 a,b,c;
      for (j = 0; j < 480;j++)
      {
            for (i = 0; i <640; i++)
            {
                  a=((u32)(*(p++)));//蓝色
                  b=((u32)(*(p++)))<<8;//绿色
                  c=((u32)(*(p++)))<<16;//红色
                  PixelVal=a+b+c;
            Xil_Out32(videobase+(j*1024+i)*4,PixelVal);
            }
      }
}
```

GUI

- 显示
  - 图形，文字，图像的组合
- 输入

键盘工作原理



当有键按下时，键盘分两次将位置扫描码发送给键盘接口；

按下发一次，叫接通扫描码（通码）；释放时再发一次，叫断开扫描码（断码）。

字符“G”出现在字处理软件里 次序按下“Shift”键，按下“G”键，释放“G”键，释放“Shift”键

键盘发送到计算机主机的数据是“12h，34h，F0h，34h，F0h，12h”

键盘按键数据发送规律

- 按下
  - 通码
- 释放
  - 段码(F0，通码）
- 按住不放
  - 机打着
  - 连续发送通码
  - 机打延时（第一个通码与第二个通码之间的延时）
  - 机打速率（在第二个通码之后的通码发送速率）
- 电子琴按键模拟

命令集

- **0xED (Set/Reset LEDs)**——主机在本命令后跟随一个参数字节，用于指示键盘上Num Lock, Caps Lock 和 Scroll Lock LED的状态。

Always0	Always0	Always0	Always0	Always0	Caps Lock	Num Lock	ScrollLock

- 相应位为1表示LED亮，为0表示LED灭。

鼠标工作原理

- 标准的PS/2鼠标,支持输入X（左右)位移、Y（上下）位移、左键、中键和右键鼠标以一个固定的频率读取这些输入，并更新不同的计数器，然后标记出反映的移动和按键状态。
- 标准的鼠标有两个计数器保持位移的跟踪：X位移计数器和Y位移计数器
- 位移计数器是一个9位2进制的补码整数。

字节	D7	D6	D5	D4	D3	D2	D1	D0
字节1	Y溢出	X溢出	Y符号位	X符号位	1	中间键	右键	左键
字节2	X 位移							
字节3	Y 位移							

●鼠标在没收到“使能数据报告”（0xF4）命令之前不会发送任何位移数据包给主机

鼠标在屏幕中的位置

- 例9.6 如果用一个计数单位表示显示器的一个像素，那么在缺省4个计数单位/毫米的分辨率下，鼠标的实际移动范围上下、左右分别为多少毫米就表示了分辨率为640*480整个显示器的范围？

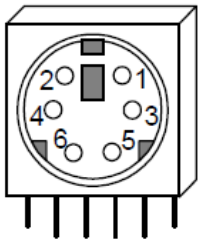
解：假设鼠标移动范围的中心位置为显示器中心，鼠标左右分别移动320个计数单位就表示了整个显示器的水平范围，而320个计数单位在缺省分辨率的情况下，需要移动80毫米，即左右各8厘米。

鼠标上下各移动240个计数单位就表示了整个显示器的垂直范围，而240个计数单位在缺省分辨率的情况下，需要移动60毫米，即上下各6厘米。

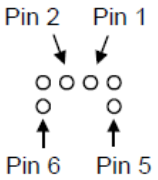
鼠标有四种标准的工作模式。

- Reset——鼠标在上电或收到“Reset”（0xFF）命令后进入Reset 模式
  - 鼠标在没收到“使能数据报告”（0xF4）命令之前不会发送任何位移数据包给主机。
- Stream 模式中，一旦鼠标检测到位移或发现一个或多个鼠标键的状态改变了就发送位移数据包。数据报告的最大速率是采样速率。
- Remote——在这个模式下鼠标以当前的采样速率读取输入，并更新它的计数器和标志。但是它只在主机请求数据的时候才报告给主机位移和按键状态。
- wrap 回声模式，鼠标收到的每个字节都会被发回主机，甚至收到的是一个有效的命令鼠标都不会应答这条命令，它只把这个字节回送给主机。

PS/2通讯接口



PS/2
Connector



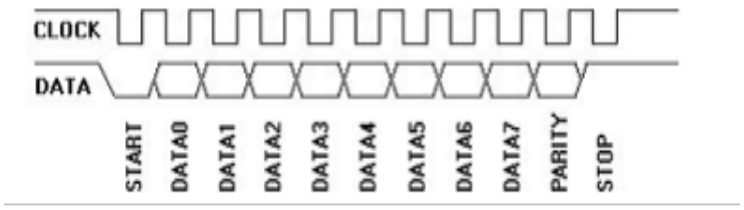
Bottom-up
hole pattern

Pin Definitions

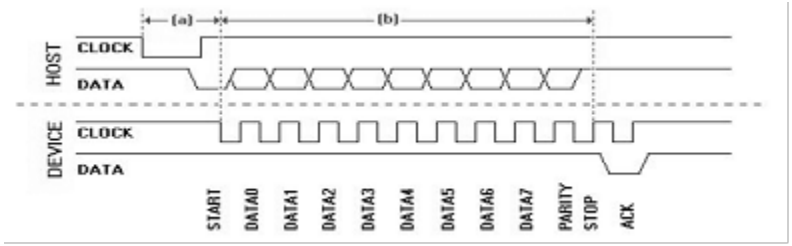
Pin	Function
1	Data
2	Reserved
3	GND
4	Vdd
5	Clock
6	Reserved

- PS/2通讯数据帧格式

1个起始位	总是逻辑0
8个数据位	（LSB）低位在前
1个奇偶校验位	奇校验
1个停止位	总是逻辑1
1个应答位	仅用在主机对设备的通讯中

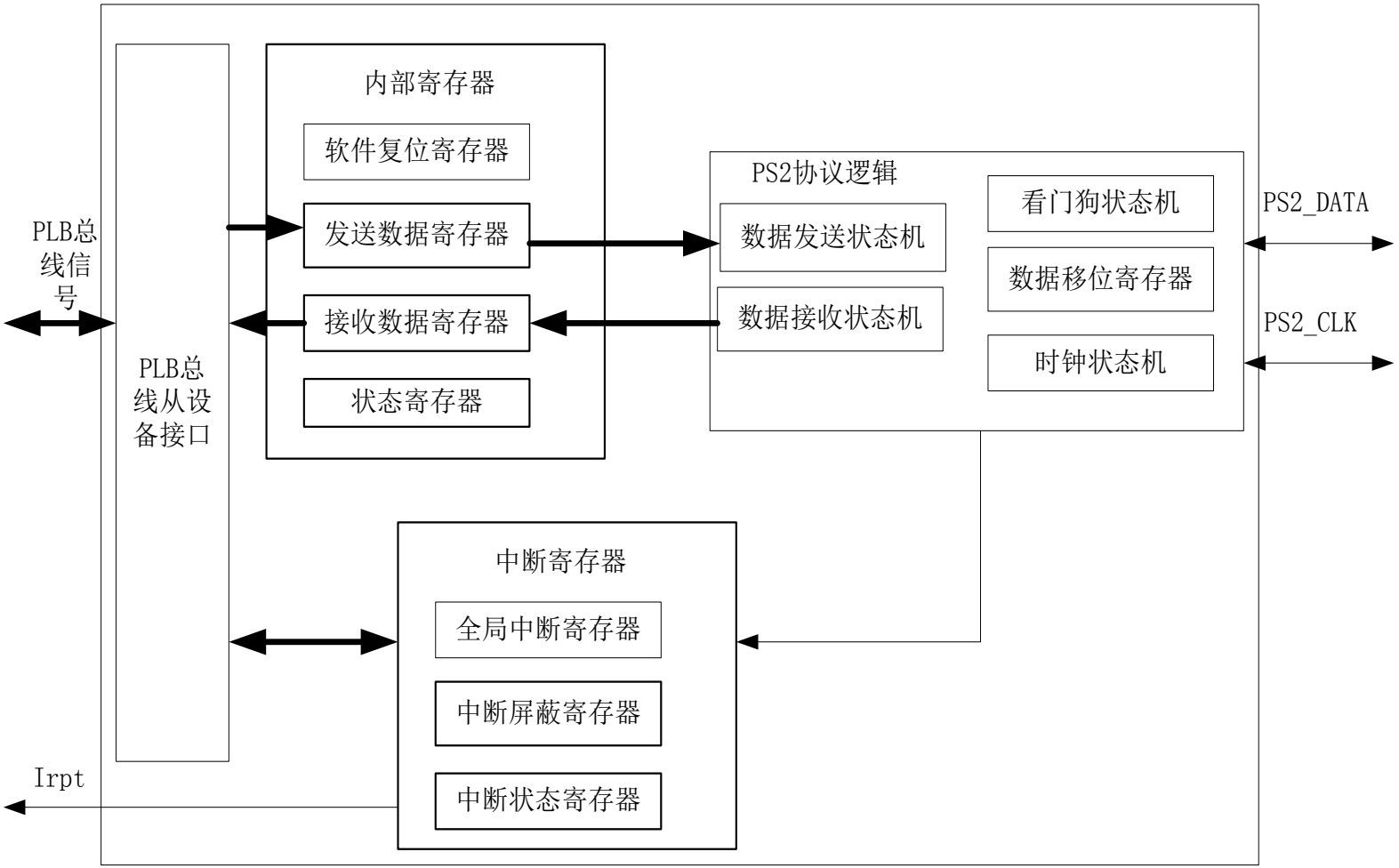


PS/2设备到主机的通讯



主机到设备的通讯

Xilinx XPS PS2 IP核简介



PS2接口内寄存器

寄存器名称	功能	偏移地址	读写权限
SRST_1	PS2接口1软件复位	0x0000	写0x0000000A复位接口
STATUS_REG1	PS2接口1的状态	0x0004	读
RX1_DATA	PS2接口1接收的数据	0x0008	读
TX1_DATA	PS2接口1发送的数据	0x000C	写
SRST_2	PS2接口2软件复位	0x1000	写0x0000000A复位接口
STATUS_REG2	PS2接口2的状态	0x1004	读
RX2_DATA	PS2接口2接收的数据	0x1008	读
TX2_DATA	PS2接口2发送的数据	0x100C	写
GIE_1	PS2接口1全局中断允许	0x002c	写最高位为1允许中断
IPISR_1	PS2接口1中断状态	0x0030	读/写1清除
IPIER_1	PS2接口1中断屏蔽	0x0038	写
GIE_2	PS2接口2全局中断允许	0x102c	写最高位为1允许中断
IPISR_2	PS2接口2中断状态	0x1030	读/写1清除
IPIER_2	PS2接口2中断屏蔽	0x1038	写

键盘输入小写ASCII 与扫描码的对应关系

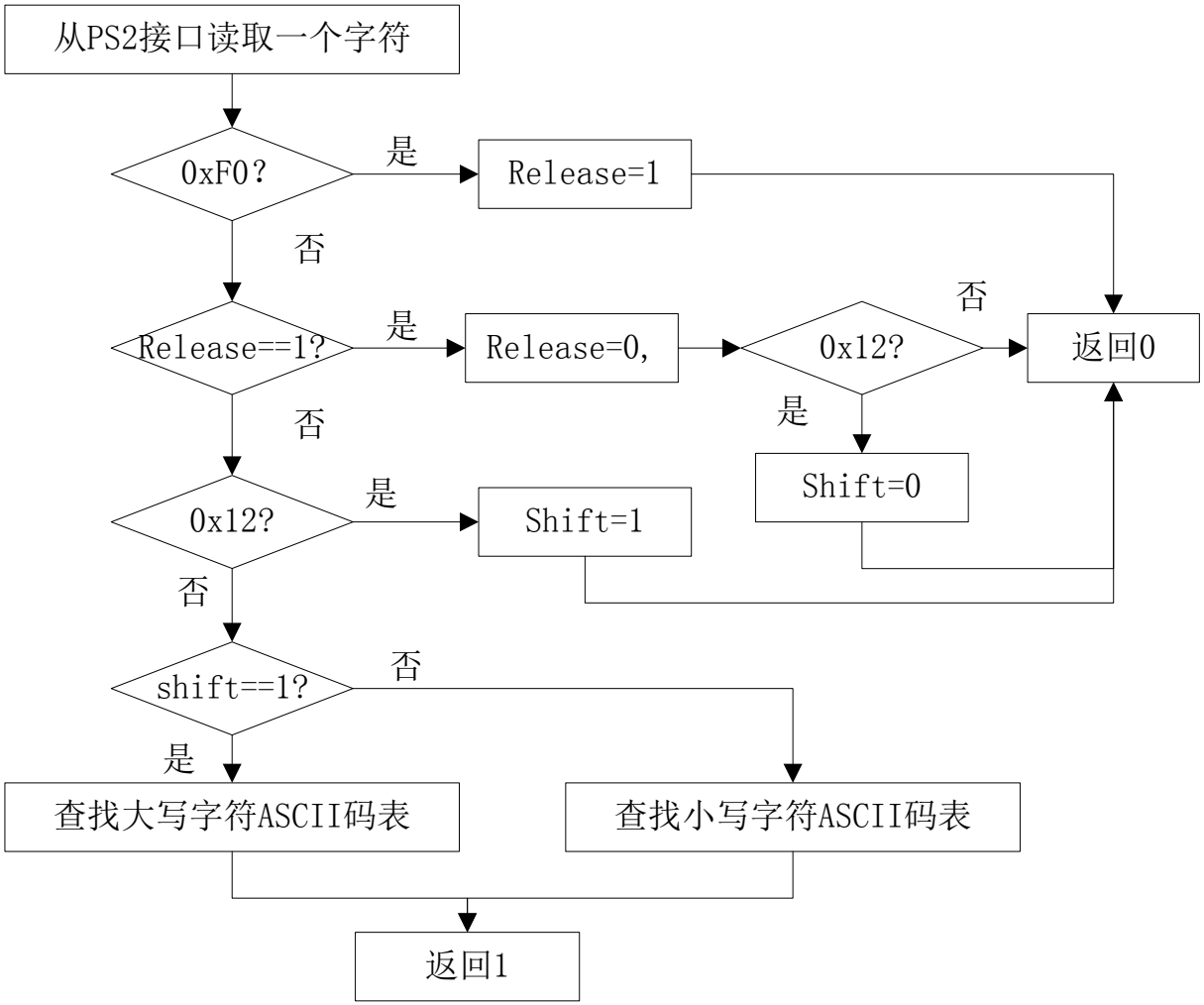
```
const unsigned char unshifted[][2] =
                    {
                  0x76, 200, //ESC
                  0x05, 201, //F1
                    0x06, 202,
                    0x04, 203,
                    0x0c, 204,
                    0x03, 205,
                    0x0b, 206,
                    0x83, 207,
                    0x0a, 208,
                    0x01, 209,
                   0x09, 0xd2,
                   0x78, 0xd3,
                   0x07, 0xd4,
                    0x0e, '`',
                    0x16, '1',
                    0x1e, '2',
                    0x26, '3',
```

```
                    0x25, '4',
                    0x2e, '5',
                    0x36, '6',
                    0x3d, '7',
                    0x3e, '8',
                    0x46, '9',
                    0x45, '0',
                    0x4e, '-',
                    0x55, '=',
                   0x66, 0xd5,
                   0x0d, 0xd,
                    0x15, 'q',
                    0x1d, 'w',
                    0x24, 'e',
                    0x2d, 'r',
                    0x2c, 't',
                    0x35, 'y',
                    0x3c, 'u',
                    0x43, 'i',
```

```
                    0x44, 'o',
                    0x4d, 'p',
                    0x54, '[',
                    0x5b, ']',
                   0x5d, '\\',
                   0x58, 0xd6,
                    0x1c, 'a',
                    0x1b, 's',
                    0x23, 'd',
                    0x2b, 'f',
                    0x34, 'g',
                    0x33, 'h',
                    0x3b, 'j',
                    0x42, 'k',
                    0x4b, 'l',
                    0x4c, ';',
                   0x52, '\'',
                   0x5a, 0xa,
                   0x12, 0xd7,
```

```
                    0x1a, 'z',
                    0x22, 'x',
                    0x21, 'c',
                    0x2a, 'v',
                    0x32, 'b',
                    0x31, 'n',
                    0x3a, 'm',
                    0x41, ',',
                    0x49, '.',
                    0x4a, '/',
                   0x59, 0xd8,
                   0x14, 0xd9,
                   0x11, 0xda,
                    0x29, ' ',
                   0x75, 0xdb,
                   0x74, 0xdc,
                   0x6b, 0xdd,
                   0x72, 0xde,
                    };
```

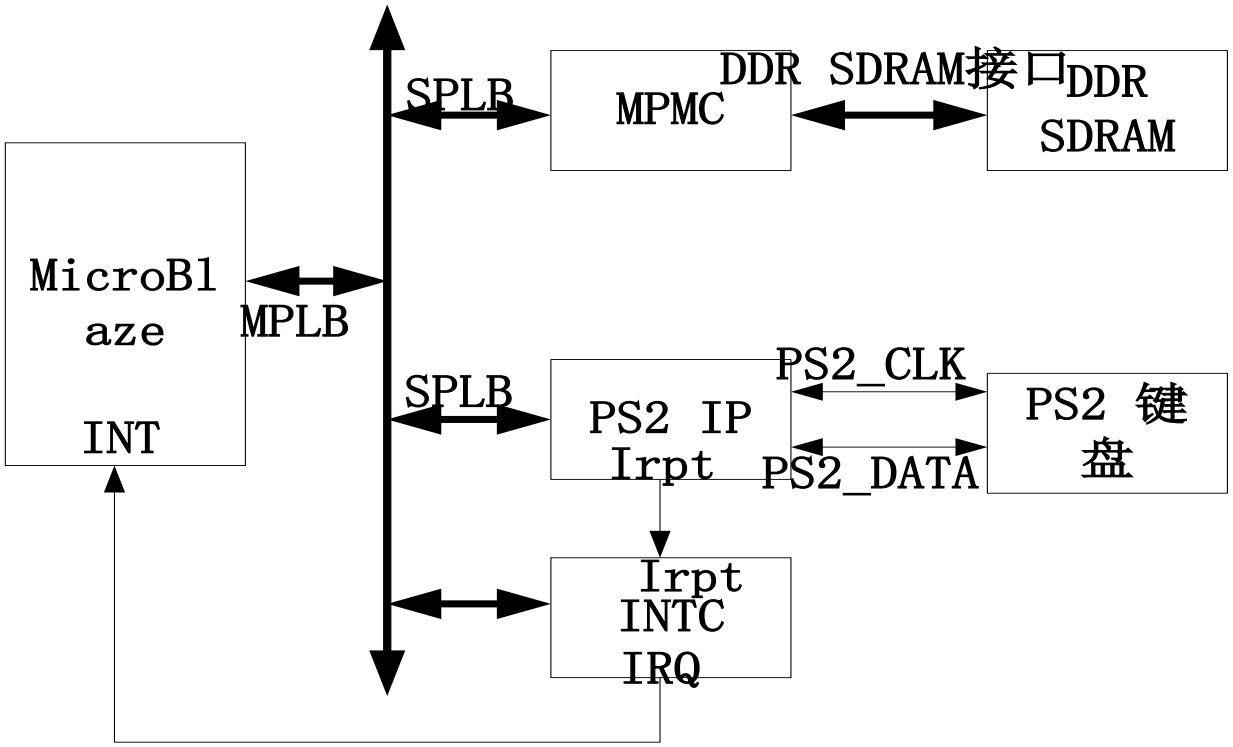
大小写识别的程序流程


从PS2接口读取一个字符
0xF0?
是
Release=1
否
Release==1?
是
Release=0,
0x12?
否
返回0
是
Shift=0
否
0x12?
是
Shift=1
否
shift==1?
否
是
查找大写字符ASCII码表
查找小写字符ASCII码表
返回1

PS2 API

XPs2_Reset
XPs2_GetStatus
XPs2_IntrGlobalEnable
XPs2_IntrGlobalDisable
XPs2_IntrEnable
XPs2_IntrDisable
XPs2_IntrGetEnabled
XPs2_IntrGetStatus
XPs2_IntrClear
XPs2_LookupConfig(u16) : XPs2_Config*
XPs2_CfgInitialize(XPs2*, XPs2_Config*, u32) : int
XPs2_Send(XPs2*, u8*, u32) : u32
XPs2_Recv(XPs2*, u8*, u32) : u32
XPs2_SetHandler(XPs2*, XPs2_Handler, void*) : void
XPs2_IntrHandler(XPs2*) : void
XPs2_SelfTest(XPs2*) : int

- 例9.7已知某基于Microblaze微处理器PLB总线的计算机系统，要求为该计算机系统PS2接口键盘设计中断方式接口电路，并采用中断方式编程读入任意按键扫描码



main

```
ps2_config=XPs2_LookupConfig(XPAR_XPS_PS2_0_0_DEVICE_ID);
XPs2_CfgInitialize(&ps2, ps2_config,XPAR_XPS_PS2_0_0_BASEADDR);
XPs2_IntrGlobalEnable(&ps2);
XPs2_IntrEnable(&ps2, XPS2_IPIXR_ALL);
XPs2_SetHandler(&ps2, (XPs2_Handler)handle, (void *)&ps2);
 XIntc_Initialize(&intc, XPAR_INTC_0_DEVICE_ID);
//初始化中断控制器数据结构
 XIntc_Connect(&intc, XPAR_XPS_INTC_0_XPS_PS2_0_IP2INTC_IRPT_1_INTR,
      (XInterruptHandler)XPs2_IntrHandler,(void *)&ps2);
 XIntc_Start(&intc, XIN_REAL_MODE);
XIntc_Enable(&intc,XPAR_XPS_INTC_0_XPS_PS2_0_IP2INTC_IRPT_1_INTR);
//使能PS2接口对应Intr的中断请求
microblaze_register_handler((XInterruptHandler)XIntc_InterruptHandler, &intc);
microblaze_enable_interrupts();
while(1){
      do{
            receivefinish=0;
            XPs2_Recv(&ps2, &recv, 1);
            }
            while(receivefinish==0);
            }
      return XST_SUCCESS;
}
```

```
void handle(void *CallBackRef, u32 IntrMask, u32
    NumBytes)
{
    switch (IntrMask){
    case XPS2_IPIXR_RX_FULL:
          receivefinish=1;
          break;
    default: break;
    }
}
```

- 由于鼠标数据报告包含3个字节，因此接收鼠标数据报告时总是三个字节。
- **XPs2_Recv(&ps2, pos, 3);**

主机与键盘之间的初始化通信过程举例

Keyboard: AA 自检通过
Host: ED 设置/复位状态指示器
Keyboard: FA 键盘应答
Host: 00 关闭所有指示灯
Keyboard: FA 键盘应答
Host:F2 读ID
Keyboard: FA 键盘应答
Keyboard: AB ID的第一个字节
Host: ED 设置/复位状态指示器；
Keyboard: FA 键盘应答
Host: 02 点亮 Num Lock指示灯
Keyboard: FA 键盘应答
Host: F3 设置机打速率和延时；
Keyboard: FA 键盘应答
Host: 20 500 ms / 30.0 reports/sec
Keyboard: FA 键盘应答
Host: F4 使能键盘
Keyboard: FA 键盘应答
Host: F3 设置机打速率和延时
Keyboard: FA 键盘应答
Host: 00 250 ms / 30.0 reports/sec
Keyboard: FA 键盘应答

主机与鼠标之间的初始化通信过程举例

- Mouse: AA　自我测检通过
- Mouse: 00　鼠标ID
- Host: FF　主机发送复位命令
- Mouse: FA　鼠标应答
- Mouse: AA　自我测检通过
- Mouse: 00　鼠标ID
- Host: FF　主机发送复位命令
- Mouse: FA　鼠标应答
- Mouse: AA　自我测检通过
- Mouse: 00　鼠标ID
- Host: FF　主机发送复位命令
- Mouse: FA　鼠标应答
- Mouse: AA　自我测检通过
- Mouse: 00　鼠标ID
- Host: F3　设置采样速率,识别鼠标是否带滚轮
- Mouse: FA　鼠标应答
- Host: C8　采样速率 200点/秒
- Mouse: FA　鼠标应答
- Host: F3　设置采样速率
- Mouse: FA　鼠标应答
- Host:64　采样速率 100点/秒
- Mouse: FA　鼠标应答

主机与鼠标之间的初始化通信过程举例

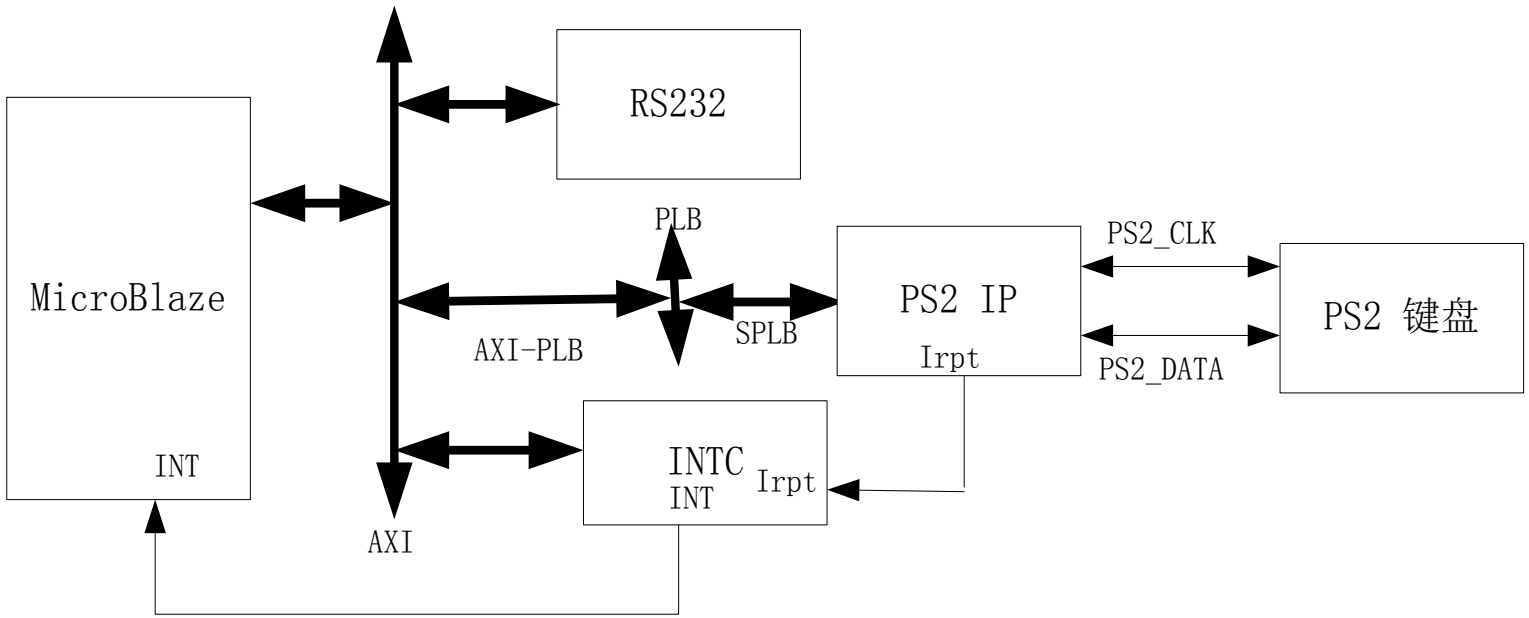
- Host: F3　设置采样速率
- Mouse: FA　鼠标应答
- Host:50　采样速率 80点/秒
- Host: F2　获取设备ID
- Mouse: FA　鼠标应答
- Mouse: 00　鼠标ID,如果回应的是03带滚轮的三键鼠标
- Mouse: FA　鼠标应答
- Host: F3　设置采样速率
- Mouse: FA　鼠标应答
- Host:0A　采样速率 10点/秒
- Mouse: FA　鼠标应答
- Host: F2　获取设备ID
- Mouse: FA　鼠标应答
- Mouse: 00　鼠标ID
- Host: E8　设置分辨率
- Mouse: FA　鼠标应答
- Host: 03　8个计数值/毫米
- Mouse: FA　鼠标应答
- Host: E6　缩放比例1:1
- Mouse: FA　鼠标应答
- Host: F3　设置采样速率
- Mouse: FA　鼠标应答
- Host28　采样速率 40点/秒
- Mouse: FA　鼠标应答
- Host: F4　使能数据报告
- Mouse: FA　鼠标应答

带滚轮4字节的PS/2协议数据格式

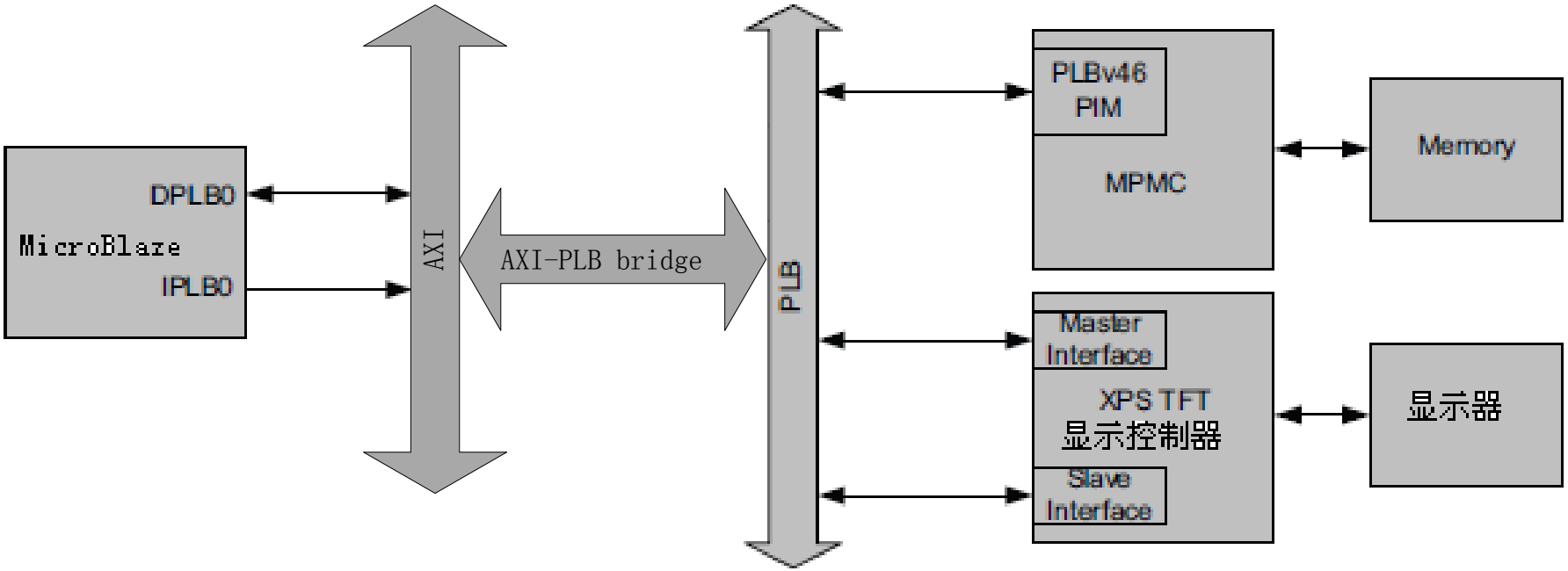
Y Overflow	X Overflow	Y Sign	X Sign	1	Middle Button	Right Button	Left Button
X movement							
Y movement							
ZH movement				ZL movement			

ZH（横向）和**ZL**（纵向）都采用二进制补码表示，范围为**-8～7**。
单一纵向滚轮仅使用低**4**位

PS2实验


RS232
MicroBlaze
INT
PLB
AXI-PLB
SPLB
PS2 IP
Irpt
PS2_CLK
PS2_DATA
PS2 键盘
INTC
INT
Irpt
AXI

VGA实验


DPLB0
MicroBlaze
IPLB0
AXI
AXI-PLB bridge
PLB
PLBv46
PIM
MPMC
Memory
Master
Interface
XPS TFT
显示控制器
Slave
Interface
显示器

作业

- 1，2，5，9