第九章习题解答

1. 根据表 9- 13 显示的不同模式下的 VGA 时序要求，计算各种模式下像素时钟频率，各段要求的像素个数以及实际显示区域的像素范围。

解答：

图像格式	行时序（像素个数）				像素时钟频率	显示区域列	显示区域行
	a	b	c	d			
1024*768（75Hz）	96	176	1024	16	78.750MHz	272~1295	31~798
1024*768（60Hz）	136	160	1024	24	65MHz	296~1319	35~802
800*600（75Hz）	80	160	800	16	49.5MHz	240~1039	24~623
800*600（60Hz）	128	88	800	40	40MHz	216~1015	27~626

表 9- 1 VGA 参考时序数据

图像格式	行时序（us）				帧时序（行数）			
	a	b	c	d	a	b	c	d
1024*768（75Hz）	1.2（1.219）	2.2（2.235）	13.0（13.003）	0.2（0.203）	3	28	768	1
1024*768（60Hz）	2.1（2.092）	2.5（2.462）	15.8（15.754）	0.4（0.369）	6	29	768	3
800*600（75Hz）	1.6（1.616）	3.2（3.232）	16.2（16.162）	0.3（0.323）	3	21	600	1
800*600（60Hz）	3.2	2.2	20.0	1.0	4	23	600	1

2. 针对表 9- 13 所示不同图像显示分辨率，计算分别具有 8 位（3、3、2），16 位（5，6，5），24 位（8，8，8）像素颜色（红、绿、兰）需要的显示缓冲区存储容量大小。

解答：

8 位： 1024*768 时，显示缓冲区存储容量为 1024*768B=786432B

800*600 时，显示缓冲区存储容量为 800*600B=480000B

16 位：1024*768 时，显示缓冲区存储容量为 1024*768*2B=1572864B

800*600 时，显示缓冲区存储容量为 800*600*2B=960000B

24 位：1024*768 时，显示缓冲区存储容量为 1024*768*3B=2359296B

800*600 时，显示缓冲区存储容量为 800*600*3B=1440000B

3. 基于三基色原理（红色+绿色=黄色;绿色+蓝色=青色;红色+蓝色=品红;红色+绿色+蓝色=白色），试编程控制例 9. 2 所示计算机系统显示器显示黄屏、青屏、品红屏的程序。

解答：

像素在显示存储器中的表示

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

```
#include"xparameters.h"
#include"xstatus.h"
#include"xil_io.h"
#include"xtft.h"
#define TFT_FRAME_ADDR0     XPAR_MPMC_0_MPMC_HIGHADDR - 0x001FFFFF
#define TFT_FRAME_ADDR1     TFT_FRAME_ADDR0 - 0x00200000
#define XTFT_AR_OFFSET      0
#define XTFT_CR_OFFSET      4
#define XTFT_IESR_OFFSET 8
#define XTFT_CR_TDE_MASK 0x01
#define XTFT_CR_DPS_MASK 0x02
#define XTFT_IESR_VADDRLATCH_STATUS_MASK 0x01
#define XTFT_IESR_IE_MASK       0x08
#define XTFT_DISPLAY_WIDTH 640
#define XTFT_DISPLAY_HEIGHT 480
intmain()
{
    int Status;
    inti;
    //关闭显示
    Status=Xil_In32(XPAR_TFT_0_BASEADDR+XTFT_CR_OFFSET);
    Status &= (~XTFT_CR_TDE_MASK);
    Xil_Out32(XPAR_TFT_0_BASEADDR+XTFT_CR_OFFSET,Status);
    //修改显示存储区0内的像素颜色为白色
    for(i=0;i<XTFT_DISPLAY_BUFFER_WIDTH*XTFT_DISPLAY_HEIGHT;i++)
        Xil_Out32(TFT_FRAME_ADDR0+(4 * i),0x00FFFFFF);
    //查询单屏扫描是否结束
    while ((Xil_In32(XPAR_TFT_0_BASEADDR+XTFT_IESR_OFFSET)&
    XTFT_IESR_VADDRLATCH_STATUS_MASK) !=
                        XTFT_IESR_VADDRLATCH_STATUS_MASK);
    //修改显示存储区首地址为显示存储区0的首地址
    Xil_Out32(XPAR_TFT_0_BASEADDR+XTFT_AR_OFFSET,TFT_FRAME_ADDR0);
    //打开显示
    Status=Xil_In32(XPAR_TFT_0_BASEADDR+XTFT_CR_OFFSET);
    Status |= XTFT_CR_TDE_MASK;
    Xil_Out32(XPAR_TFT_0_BASEADDR+XTFT_CR_OFFSET,Status);
    //修改显示存储区1内的像素颜色为红色+绿色
    for(i=0;i<XTFT_DISPLAY_BUFFER_WIDTH*XTFT_DISPLAY_HEIGHT;i++)
    Xil_Out32(TFT_FRAME_ADDR1+(4 * i),0x00FFFF00);
```

```
    //查询单屏扫描是否结束
    while ((Xil_In32(XPAR_TFT_0_BASEADDR+XTFT_IESR_OFFSET)&
                XTFT_IESR_VADDRLATCH_STATUS_MASK) !=
                       XTFT_IESR_VADDRLATCH_STATUS_MASK);
    //修改显示存储区首地址为显示存储区1的首地址
    Xil_Out32(XPAR_TFT_0_BASEADDR+ XTFT_AR_OFFSET,TFT_FRAME_ADDR1);
    //修改显示存储区0内的像素颜色为绿色+蓝色
    for(i=0;i<XTFT_DISPLAY_BUFFER_WIDTH*XTFT_DISPLAY_HEIGHT;i++)
    Xil_Out32(TFT_FRAME_ADDR0+(4 * i), 0x0000FFFF);
    //查询单屏扫描是否结束
    while ((Xil_In32(XPAR_TFT_0_BASEADDR+XTFT_IESR_OFFSET)&
                  XTFT_IESR_VADDRLATCH_STATUS_MASK) !=
                         XTFT_IESR_VADDRLATCH_STATUS_MASK);
    //修改显示存储区首地址为显示存储区0的首地址
    Xil_Out32(XPAR_TFT_0_BASEADDR+ XTFT_AR_OFFSET,TFT_FRAME_ADDR0);
    //修改显示存储区1内的像素颜色为蓝色+红色
    for(i=0;i<XTFT_DISPLAY_BUFFER_WIDTH*XTFT_DISPLAY_HEIGHT;i++)
    Xil_Out32(TFT_FRAME_ADDR1+(4 * i),0x00FF00FF);
    //查询单屏扫描是否结束
    while ((Xil_In32(XPAR_TFT_0_BASEADDR+XTFT_IESR_OFFSET)&
                    XTFT_IESR_VADDRLATCH_STATUS_MASK) !=
                           XTFT_IESR_VADDRLATCH_STATUS_MASK);
    //修改显示存储区首地址为显示存储区0的首地址
    Xil_Out32(XPAR_TFT_0_BASEADDR+ XTFT_AR_OFFSET,TFT_FRAME_ADDR1);
    return XST_SUCCESS;
}
```

4. 试编程控制例 9.2 所示计算机系统显示器分别从像素（40，60），（100，160）以及（200，260）开始显示长度为 200 个像素的红色、绿色、蓝色水平线的程序。要求背景色为黑色。

解答：由于 TFT 内存一行对应显示器 4 行像素，因此显示器第 60 行，160 行，260 行分别对应显示内存的 15 行，40 行以及 65 行

```
#include"xparameters.h"
#include"xstatus.h"
#include"xil_io.h"
#include"xtft.h"
#define TFT_FRAME_ADDR0     XPAR_EMC_MEM0_HIGHADDR - 0x001FFFFF
#define TFT_FRAME_ADDR1     TFT_FRAME_ADDR0 - 0x00200000
#define XTFT_AR_OFFSET      0
#define XTFT_CR_OFFSET      4
#define XTFT_IESR_OFFSET 8
#define XTFT_CR_TDE_MASK 0x01
#define XTFT_CR_DPS_MASK 0x02
#define XTFT_IESR_VADDRLATCH_STATUS_MASK 0x01
#define XTFT_IESR_IE_MASK       0x08
```

```
#define XTFT_DISPLAY_WIDTH 640
#define XTFT_DISPLAY_HEIGHT 480
intmain()
{
    inti,j;
    Xil_Out32(XPAR_TFT_0_BASEADDR + XTFT_AR_OFFSET, TFT_FRAME_ADDR0);
    for (i = 0; i<= 120; i++) {
              for (j = 0; j <= 640;j++) {
                 Xil_Out32(TFT_FRAME_ADDR0+(4 * ((i) *
XTFT_DISPLAY_BUFFER_WIDTH + j)),0x0);
              }
           }
    for(i=39;i<240;i++)
       Xil_Out32(TFT_FRAME_ADDR0+4
*(15*XTFT_DISPLAY_BUFFER_WIDTH+i),0x003c0000);
    for(i=99;i<300;i++)
          Xil_Out32(TFT_FRAME_ADDR0+4
*(40*XTFT_DISPLAY_BUFFER_WIDTH+i),0x0000FF00);
    for(i=199;i<400;i++)
          Xil_Out32(TFT_FRAME_ADDR0+4
*(65*XTFT_DISPLAY_BUFFER_WIDTH+i),0x000000FF);
    return XST_SUCCESS;
}
```

5. 试编程控制例 9.2 所示计算机系统显示器显示左上角像素位置为（40，60），右下角像素为（160，480）的黄色矩形程序。要求背景色为黑色。

解答：

```
#include"xparameters.h"
#include"xstatus.h"
#include"xil_io.h"
#include"xtft.h"
#define TFT_FRAME_ADDR0     XPAR_EMC_MEM0_HIGHADDR - 0x001FFFFF
#define XTFT_AR_OFFSET      0
#define XTFT_CR_OFFSET      4
#define XTFT_IESR_OFFSET 8
#define XTFT_CR_TDE_MASK 0x01
#define XTFT_CR_DPS_MASK 0x02
#define XTFT_IESR_VADDRLATCH_STATUS_MASK 0x01
#define XTFT_IESR_IE_MASK       0x08
intmain()
{
    inti,j;
    Xil_Out32(XPAR_TFT_0_BASEADDR + XTFT_AR_OFFSET, TFT_FRAME_ADDR0);
    for (i = 0; i<= 120; i++) {
              for (j = 0; j <= 640;j++) {
```

```
                    if((i>14)&&(i<121)&&(j>39)&&(j<161))
                        Xil_Out32(TFT_FRAME_ADDR0+(4 * ((i) *
XTFT_DISPLAY_BUFFER_WIDTH + j)),0x00ffff00);
                    else
                    Xil_Out32(TFT_FRAME_ADDR0+(4 * ((i) *
XTFT_DISPLAY_BUFFER_WIDTH + j)),0x0);
                }
            }
    return XST_SUCCESS;
}
```

6. PS2 通信协议的时钟信号由谁提供？分别描述主机到设备以及设备到主机的一帧数据格式。

解答：PS2 通信协议的时钟信号由 PS2 设备提供

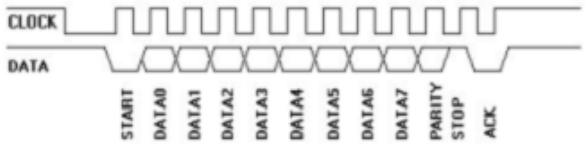


图 9- 1 主机到设备的通讯

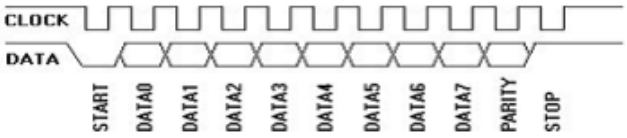


图 9- 2 设备到主机的通讯

7. 试描述 PS2 通信协议中主机到设备的通信过程和设备到主机的通信过程。

解答：

从 PS2 设备向 PC 机发送一个字节按照下面的步骤进行：

1) 检测时钟线电平，如果时钟线为低，则延时 50μs；

2) 检测判断时钟信号是否为高，为高，则向下执行，为低，则转到(1)；

3) 检测数据线是否为高，如果为高则继续执行；如果为低，则放弃发送（此时 PC 机在向 PS2 设备发送数据，所以 PS2 设备要转移到接收程序处接收数据）；

4) 延时 20μs（如果此时正在发送起始位，则应延时 40μs）；

5) 输出起始位(0)到数据线上。需要注意的是：在送出每一位后都要检测时钟线，以确保 PC 机没有抑制 PS2 设备，如果有则中止发送；

6) 输出 8 个数据位到数据线上；

7) 输出校验位；

8) 输出停止位(1)；

9) 延时 30μs（如果在发送停止位时释放时钟信号则应延时 50μs ）；

主机必须按下面的步骤发送数据到 PS/2 设备。

1）把时钟线拉低至少 100 微秒

2）把数据线拉低

3）释放时钟线

4）等待设备把时钟线拉低

5）设置/复位数据线发送第一个数据位

6）等待设备把时钟拉高

7）等待设备把时钟拉低
8）重复 5）-7）步发送剩下的 7 个数据位和校验位
9）释放数据线
10）等待设备把数据线拉低
11）等待设备把时钟线拉低
12）等待设备释放数据线和时钟线

8. 已知在某字符处理软件中，键盘给主机发送的数据序列为：12h，34h，F0h，34h，F0h，12h，66h，F0h，66h，试说明字符处理软件在屏幕上的显示过程。
解答：数据序列代表的动作：
12h，按下 shift
34h，按下 G
F0h，34h，释放 G
F0h，12h，释放 shift
66h，按下 backspace
F0h，66h，释放 backspace
由此可知字符处理软件在屏幕上首先显示大写的 G，然后又删除了该字符。

9. 已知鼠标的缩放比例为 1:1，且发送给主机的三字节数据为：0x38，0x56，0x78，试说明鼠标的移动方向和在水平以及垂直方向上的位移量。
解答：
表 9- 2PS/2 鼠标发送的 3 字节数据包格式

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							

0X38 表示鼠标的位移无论 XY 都是负方向，即向左下方移动
x 方向的位移量为：0X56，左移 86 列，缩放比例 1：1 时，4 个计数单位 1mm，因此鼠标左移了 21.5mm
y 方向的位移量为：0x78，下移 120 行，鼠标下移了 30mm。

10. 基于例 9. 7 所示计算机系统，利用 standalone 操作系统提供的 PS2 设备驱动，试编程控制 PS2 键盘通过 Caps Lock 键实现大小写字符输入。要求按下 Caps Lock 键时，Caps Lock 指示灯亮，再次按下 Caps Lock 键时，Caps Lock 灯灭，灯的亮、灭分别表示输入大、小写字符。
解答：
shift 键可以实现大小写控制，键盘缓冲区的数据包括按键码和释放码，因此基于例 9.7 需要增加 caps lock 按键按下的识别，并且控制 LED 灯，因此需要发送数据给键盘，键盘的数据发送为一次一个字节。

```
#include<stdio.h>
#include"xparameters.h"
#include"xil_exception.h"
#include"xintc.h"
#include"xps2.h"
```

```
#include"xstatus.h"
#include"key.h"
#include"xil_io.h"
staticXPs2 ps2;
staticXIntcintc;
unsignedchar word[3]={0,0xa,0xd};
unsignedcharrecv;
staticintreceivefinish,release,shift,caps=0;
intreadchar(void);
u8sendbuffer[2]={0xed,0x0};
voidhandle(void *CallBackRef, u32IntrMask, u32NumBytes);
intmain(){
//  XIntcintc;
    XPs2_Config *ps2_config;
    int Status;
    ps2_config=XPs2_LookupConfig(XPAR_XPS_PS2_0_0_DEVICE_ID);
    Status=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);
    Status = XIntc_Initialize(&intc, XPAR_INTC_0_DEVICE_ID);
//初始化中断控制器数据结构
    if (Status != XST_SUCCESS) {
           return XST_FAILURE;
       }
Status =
XIntc_Connect(&intc,XPAR_AXI_INTC_0_XPS_PS2_0_IP2INTC_IRPT_1_INTR,(XI
nterruptHandler)XPs2_IntrHandler,(void *)&ps2);
    if (Status != XST_SUCCESS) {
           return XST_FAILURE;
       }
    XIntc_Enable(&intc,XPAR_AXI_INTC_0_XPS_PS2_0_IP2INTC_IRPT_1_INTR)
;
    //使能PS2接口对应Intr的中断请求
    Status = XIntc_Start(&intc, XIN_REAL_MODE);
    if (Status != XST_SUCCESS) {
           return XST_FAILURE;
       }

    microblaze_register_handler((XInterruptHandler)XIntc_InterruptHan
dler, &intc);
    microblaze_enable_interrupts();
    while(1){
```

```
            release=0;
            shift=0;
            do{
                Status=readchar();
                }
            while(Status!=1);
            if (word[0]==0xd6){//判断是否为CAPS lock按键
        sendbuffer[1]=sendbuffer[1]^0x4;//caps lockLED按一次取一次反
                caps=caps^1;//设置标志位，按一次取一次反
            XPs2_Send(&ps2, sendbuffer, 1);//发送caps lockLED灯亮灭的命令
                XPs2_Recv(&ps2, recv, 1);
                XPs2_Send(&ps2, sendbuffer+1, 1);
                XPs2_Recv(&ps2, recv, 1);
                }
            print("The character that you have typed is ");//显示字符
            print(word);

            }
    return XST_SUCCESS;
}
voidhandle(void *CallBackRef, u32IntrMask, u32NumBytes)
{
    switch (IntrMask){
    case XPS2_IPIXR_RX_FULL:
        receivefinish=1;
        break;
    default: break;
    }
}
intreadchar(void){

    intrecvchar,i;
    recvchar=0;
    receivefinish=0;
    XPs2_Recv(&ps2, &recv, 1);
    while(receivefinish==0);
    if(recv==0xf0){
        release=1;
    }
    elseif (release==1){
        release=0;
        if(recv==0x12)
            shift=0;
        }
```

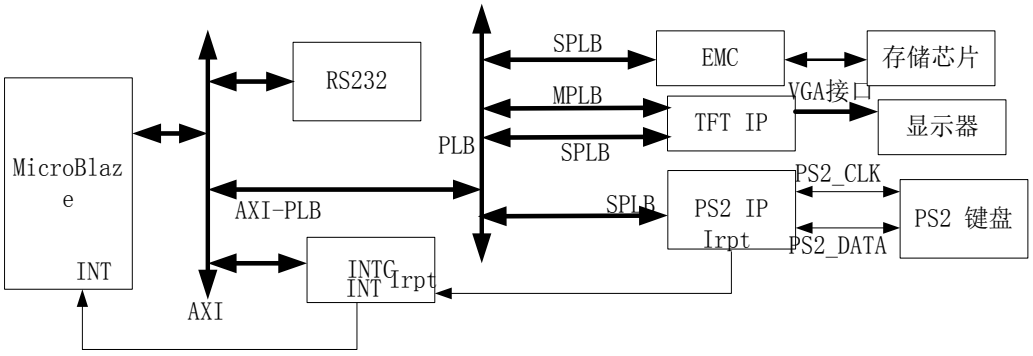
```
    elseif(recv==0x12){
        shift=1;
            }
    elseif((shift==1)||(caps==1)){//判断是否按下shift键或者caps lock亮
        for(i = 0;(shifted[i][0] != recv) && shifted[i][0];i++);//查表找到对应的字符大写码值
        if (shifted[i][0] == recv)
            {
                recvchar = 1; //解码成功标志
                word[0]= shifted[i][1];
            }
        }
    else {
        for(i = 0;(unshifted[i][0] != recv) &&unshifted[i][0];i++);//查表找到对应的字符小写码值
        if (unshifted[i][0] == recv)
            {
                recvchar = 1;//解码成功标志
                word[0]=unshifted[i][1];
            }
        }
returnrecvchar;
}
```

keyboard.c

keyboard.h 仍然为大小写编码数组，此处忽略。

11. 基于例 9.5 和例 9.7，试编程控制 VGA 显示器在屏幕的正中间显示 PS2 键盘键入的字符。

解答：硬件电路连接框图如下：



由于屏幕的中间位置为（320，240）；但是显示器缓存一点对应4行中同一列的4个点，因此显示缓存的中间坐标为（320，60）；再加上字本身占据一定的行数，因此实际的中间位置还需要减掉字的一半的大小，此处采用（320，45）为屏幕的大致中间位置。

```
#include<stdio.h>
#include"xparameters.h"
#include"xil_exception.h"
```

```
#include"xintc.h"
#include"xps2.h"
#include"xstatus.h"
#include"key.h"
#include"xil_io.h"
#include"xtft.h"
#define TFT_DEVICE_ID    XPAR_TFT_0_DEVICE_ID
#define TFT_FRAME_ADDR0     XPAR_EMC_MEM0_HIGHADDR - 0x001FFFFF
#define FGCOLOR_VALUE       0x00ff0000 /**< Foreground Color - red */
#define BGCOLOR_VALUE       0x0    /**< Background Color - Black */
staticXPs2 ps2;
staticXIntcintc;
staticXTftTftInstance;
XTft_Config *TftConfigPtr;
char word[3]={0,0xa,0xd};
u8recv;
staticintreceivefinish,release,shift;
intreadchar(void);
u8sendbuffer[2]={0xed,0x0};
voidhandle(void *CallBackRef, u32IntrMask, u32NumBytes);
intmain(){
//  XIntcintc;
    XPs2_Config *ps2_config;
    int Status;
    TftConfigPtr = XTft_LookupConfig(TFT_DEVICE_ID);
    Status = XTft_CfgInitialize(&TftInstance, TftConfigPtr,
                    TftConfigPtr->BaseAddress);
    XTft_SetFrameBaseAddr(&TftInstance, TFT_FRAME_ADDR0);
    XTft_ClearScreen(&TftInstance);
    XTft_SetColor(&TftInstance, FGCOLOR_VALUE, BGCOLOR_VALUE);
    XTft_SetPosChar(&TftInstance, 320, 45);
    XTft_Write(&TftInstance, 'A');
    ps2_config=XPs2_LookupConfig(XPAR_XPS_PS2_0_0_DEVICE_ID);
    Status=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);
    Status = XIntc_Initialize(&intc, XPAR_INTC_0_DEVICE_ID);
//初始化中断控制器数据结构
    if (Status != XST_SUCCESS) {
           return XST_FAILURE;
       }
Status =
```

```
XIntc_Connect(&intc,XPAR_AXI_INTC_0_XPS_PS2_0_IP2INTC_IRPT_1_INTR,(XI
nterruptHandler)XPs2_IntrHandler,(void *)&ps2);
    if (Status != XST_SUCCESS) {
            return XST_FAILURE;
        }
    XIntc_Enable(&intc,XPAR_AXI_INTC_0_XPS_PS2_0_IP2INTC_IRPT_1_INTR)
;
    //使能PS2接口对应Intr的中断请求
    Status = XIntc_Start(&intc, XIN_REAL_MODE);
    if (Status != XST_SUCCESS) {
            return XST_FAILURE;
        }

    microblaze_register_handler((XInterruptHandler)XIntc_InterruptHan
dler, &intc);
    microblaze_enable_interrupts();
    while(1){
            release=0;
            shift=0;
            do{
                Status=readchar();
                }
            while(Status!=1);
            print("The character that you have typed is ");//显示字符
            print(word);
            XTft_SetPosChar(&TftInstance, 320, 45);
            XTft_Write(&TftInstance, word[0]);
            }
    return XST_SUCCESS;
}
voidhandle(void *CallBackRef, u32IntrMask, u32NumBytes)
{
    switch (IntrMask){
    case XPS2_IPIXR_RX_FULL:
        receivefinish=1;
        break;
    default: break;
    }
}
intreadchar(void){

    intrecvchar,i;
    recvchar=0;
    receivefinish=0;
```

```
    XPs2_Recv(&ps2, &recv, 1);
    while(receivefinish==0);
    if(recv==0xf0){
        release=1;
    }
    elseif (release==1){
        release=0;
        if(recv==0x12)
            shift=0;
        }
    elseif(recv==0x12){
        shift=1;
            }
    elseif(shift==1){//判断是否按下shift键
        for(i = 0;(shifted[i][0] != recv) && shifted[i][0];i++);//查表找到对应的字符大写码值
        if (shifted[i][0] == recv)
            {
                recvchar = 1; //解码成功标志
                word[0]= shifted[i][1];
            }
        }
    else {
        for(i = 0;(unshifted[i][0] != recv) &&unshifted[i][0];i++);//查表找到对应的字符小写码值
        if (unshifted[i][0] == recv)
            {
                recvchar = 1;//解码成功标志
                word[0]=unshifted[i][1];
            }
        }
returnrecvchar;
}
```

12. 假定鼠标在显示器的初始位置为（320，240），且一个计数单位对应一个像素，显示器大小为 640*480，修改例 9.8 程序计算任何时刻鼠标在显示器上的相对位置信息。

解答：

```
/*
 * mouse.c
 *
*  Created on: 2013-9-22
 *      Author: Administrator
 */
#include<stdio.h>
```

```
#include"xparameters.h"
#include"xil_exception.h"
#include"xps2.h"
#include"xstatus.h"
#include"xintc.h"
#include"xio.h"
staticXPs2 ps2;
staticXIntcintCtrl;
staticintreceivefinish;
#defineprintfxil_printf     /* A smaller footprint printf */
shortintcposx=320;
shortintcposy=240;
shortintxchange,ychange;
voidhandle(void *CallBackRef, u32IntrMask, u32NumBytes);
unsignedcharmousesr(unsignedchar *command,unsignedchar
*buffer,intlbf);
intmain(){
    XPs2_Config *ps2_config;
    int Status;
    unsignedcharcommand,answer,pos[4];
//PS2接口实例化
    ps2_config=XPs2_LookupConfig(XPAR_XPS_PS2_0_0_DEVICE_ID);
    Status=XPs2_CfgInitialize(&ps2, ps2_config,
           XPAR_XPS_PS2_0_0_BASEADDR);
//设置中断处理回调函数
    XPs2_SetHandler(&ps2, (XPs2_Handler)handle,(void *)&ps2);
    XIntc_Initialize(&intCtrl,XPAR_AXI_INTC_0_DEVICE_ID );
    XIntc_Enable(&intCtrl,XPAR_AXI_INTC_0_XPS_PS2_0_IP2INTC_IRPT_1_IN
TR);
    XIntc_Connect(&intCtrl,XPAR_AXI_INTC_0_XPS_PS2_0_IP2INTC_IRPT_1_I
NTR,
                  (XInterruptHandler)XPs2_IntrHandler,(void *)&ps2);
//中断控制器实例化
    XPs2_IntrGlobalEnable(&ps2);
    XPs2_IntrEnable(&ps2, XPS2_IPIXR_RX_FULL);
    microblaze_register_handler((XInterruptHandler)XIntc_InterruptHan
dler, (void *)&intCtrl);
    microblaze_enable_interrupts();
    XIntc_Start(&intCtrl, XIN_REAL_MODE);//启动中断控制器
    command =0xff;
    answer=mousesr(&command,pos,0);
    for(Status=0;Status<5000000;Status++);
    XPs2_Recv(&ps2,pos,3);
    command =0xff;
```

```
answer=mousesr(&command,pos,0);
for(Status=0;Status<5000000;Status++);
XPs2_Recv(&ps2,pos,3);
command =0xff;
answer=mousesr(&command,pos,0);
for(Status=0;Status<5000000;Status++);
XPs2_Recv(&ps2,pos,3);
command =0xf2;
answer=mousesr(&command,pos,2);
command =0xf3;
answer=mousesr(&command,pos,1);
command =0x0a;
answer=mousesr(&command,pos,1);
command =0xf2;
answer=mousesr(&command,pos,2);
command =0xe8;
answer=mousesr(&command,pos,1);
command =0x03;
answer=mousesr(&command,pos,1);
command =0xe6;
answer=mousesr(&command,pos,1);
command =0xf3;
answer=mousesr(&command,pos,1);
command =0x28;
answer=mousesr(&command,pos,1);
command =0xf4;
answer=mousesr(&command,pos,1);
        while(1)
        {
            receivefinish=0;//循环接收streaming模式下鼠标数据报告
            XPs2_Recv(&ps2,pos,4);
            while(receivefinish==0);
            printf("%x,%x,%x,%x\r\n",pos[0],pos[1],pos[2],pos[3]);
            if((pos[0]&0x8)!=0x8){//判断鼠标数据的格式，第一个数据无效
            if ((pos[1]&0x10)==0x10)
                xchange=0xff00|pos[2];//x符号位扩展
            elsexchange=0x0000|pos[2];
            if ((pos[1]&0x20)==0x20)
                ychange=0xff00|pos[3];//y符号位扩展
            elseychange=0x0000|pos[3];
            }
            else {//第一个数据有效
                if ((pos[0]&0x10)==0x10)
                                    xchange=0xff00|pos[1];
```

```
                                elsexchange=0x0000|pos[1];
                                if ((pos[0]&0x20)==0x20)
                                    ychange=0xff00|pos[2];
                                elseychange=0x0000|pos[2];
                }
                cposx=cposx+xchange;//x向右为正
                cposy=cposy-ychange;//y向上为正，因此减法计算坐标
                if(cposx<0)
                    cposx=0;
                if (cposx>639)
                    cposx=639;
                if(cposy<0)
                    cposy=0;
                if(cposy>479)
                    cposy=479;
                printf("cposx %d,cposy %d\r\n",cposx,cposy);
                }
        return XST_SUCCESS;
}
voidhandle(void *CallBackRef, u32IntrMask, u32NumBytes)
{
    switch (IntrMask){
    case XPS2_IPIXR_RX_FULL:
        receivefinish=1;//若是接收缓冲区满产生的中断，则置receivefinish为1
        break;
    default: break;
    }
    }
unsignedcharmousesr(unsignedchar *command,unsignedchar
*buffer,intlbf){
    if(command[0]!=0)
        XPs2_Send(&ps2,command,1);
    if(lbf!=0){
        receivefinish=0;
        XPs2_Recv(&ps2,buffer,1);
        while(receivefinish==0);
        if(*buffer!=0xfa)
            return *buffer;
        elseif((lbf-1)>0)
        {
            receivefinish=0;
            XPs2_Recv(&ps2,buffer+1,lbf-1);
            while(receivefinish==0);
        }
```

```
    }
    return buffer[0];

}
```

13. 在题8以及例9.2的基础上，编程控制鼠标移动过程中点击了左键的显示器位置画白点。

解答：

```
#include<stdio.h>
#include"xparameters.h"
#include"xil_exception.h"
#include"xps2.h"
#include"xstatus.h"
#include"xintc.h"
#include"xio.h"
#include"xil_io.h"
#include"xtft.h"
#define TFT_FRAME_ADDR0     XPAR_EMC_MEM0_HIGHADDR - 0x001FFFFF
#define XTFT_AR_OFFSET      0
#define XTFT_CR_OFFSET      4
#define XTFT_IESR_OFFSET 8
#define XTFT_CR_TDE_MASK 0x01
#define XTFT_CR_DPS_MASK 0x02
#define XTFT_IESR_VADDRLATCH_STATUS_MASK 0x01
#define XTFT_IESR_IE_MASK       0x08
staticXPs2 ps2;
staticXIntcintCtrl;
staticintreceivefinish,leftclicked=0;
#defineprintfxil_printf     /* A smaller footprint printf */
shortintcposx=320;
shortintcposy=240;
shortintxchange,ychange;
voidhandle(void *CallBackRef, u32IntrMask, u32NumBytes);
unsignedcharmousesr(unsignedchar *command,unsignedchar
*buffer,intlbf);
intmain(){
    XPs2_Config *ps2_config;
    int Status;
    intdispy;
    unsignedcharcommand,answer,pos[4];
    inti,j;
        Xil_Out32(XPAR_TFT_0_BASEADDR + XTFT_AR_OFFSET,
TFT_FRAME_ADDR0);
//PS2接口实例化
    ps2_config=XPs2_LookupConfig(XPAR_XPS_PS2_0_0_DEVICE_ID);
```

```
    Status=XPs2_CfgInitialize(&ps2, ps2_config,
           XPAR_XPS_PS2_0_0_BASEADDR);
//设置中断处理回调函数
    XPs2_SetHandler(&ps2, (XPs2_Handler)handle,(void *)&ps2);
    XIntc_Initialize(&intCtrl,XPAR_AXI_INTC_0_DEVICE_ID );
    XIntc_Enable(&intCtrl,XPAR_AXI_INTC_0_XPS_PS2_0_IP2INTC_IRPT_1_IN
TR);
    XIntc_Connect(&intCtrl,XPAR_AXI_INTC_0_XPS_PS2_0_IP2INTC_IRPT_1_I
NTR,
                  (XInterruptHandler)XPs2_IntrHandler,(void *)&ps2);
//中断控制器实例化
    XPs2_IntrGlobalEnable(&ps2);
    XPs2_IntrEnable(&ps2, XPS2_IPIXR_RX_FULL);
    microblaze_register_handler((XInterruptHandler)XIntc_InterruptHan
dler, (void *)&intCtrl);
    microblaze_enable_interrupts();
    XIntc_Start(&intCtrl, XIN_REAL_MODE);//启动中断控制器
    command =0xff;
    answer=mousesr(&command,pos,0);
    for(Status=0;Status<5000000;Status++);
    XPs2_Recv(&ps2,pos,3);
    command =0xff;
    answer=mousesr(&command,pos,0);
    for(Status=0;Status<5000000;Status++);
    XPs2_Recv(&ps2,pos,3);
    command =0xff;
    answer=mousesr(&command,pos,0);
    for(Status=0;Status<5000000;Status++);
    XPs2_Recv(&ps2,pos,3);
    command =0xf2;
    answer=mousesr(&command,pos,2);
    command =0xf3;
    answer=mousesr(&command,pos,1);
    command =0x0a;
    answer=mousesr(&command,pos,1);
    command =0xf2;
    answer=mousesr(&command,pos,2);
    command =0xe8;
    answer=mousesr(&command,pos,1);
    command =0x03;
    answer=mousesr(&command,pos,1);
    command =0xe6;
    answer=mousesr(&command,pos,1);
    command =0xf3;
```

```
answer=mousesr(&command,pos,1);
command =0x28;
answer=mousesr(&command,pos,1);
command =0xf4;
answer=mousesr(&command,pos,1);
        while(1)
        {
            receivefinish=0;//循环接收streaming模式下鼠标数据报告
            XPs2_Recv(&ps2,pos,4);
            while(receivefinish==0);
            printf("%x,%x,%x,%x\r\n",pos[0],pos[1],pos[2],pos[3]);
            if((pos[0]&0x8)!=0x8){
            if ((pos[1]&0x10)==0x10)
                xchange=0xff00|pos[2];
            elsexchange=0x0000|pos[2];
            if ((pos[1]&0x20)==0x20)
                ychange=0xff00|pos[3];
            elseychange=0x0000|pos[3];
            if((pos[1]&0x1)==0x1)
                                leftclicked=1;
                            elseleftclicked=0;
            }
            else {
                if ((pos[0]&0x10)==0x10)
                                leftclicked=1;
                            elsexchange=0x0000|pos[1];
                            if ((pos[0]&0x20)==0x20)
                                ychange=0xff00|pos[2];
                            elseychange=0x0000|pos[2];
                if((pos[0]&0x1)==0x1)
                    leftclicked=1;
                elseleftclicked=0;
            }
            cposx=cposx+xchange;
            cposy=cposy-ychange;
            if(cposx<0)
                cposx=0;
            if (cposx>639)
                cposx=639;
            if(cposy<0)
                cposy=0;
            if(cposy>479)
                cposy=479;
            printf("cposx %d,cposy %d\r\n",cposx,cposy);
```

```
                if (leftclicked){
                    dispy=cposy/4;//控制鼠标位置在可显示范围之内
                    if (dispy>95)
                        dispy=95;
                for (i = 0; i<= 120; i++) {
                            for (j = 0; j <= 640;j++) {

    if((i==dispy)&&(j>cposx-2)&&(j<cposx+2))
                                    Xil_Out32(TFT_FRAME_ADDR0+(4 * ((i)
* XTFT_DISPLAY_BUFFER_WIDTH + j)),0x00ffffff);//画一个小白矩形
                                else
                                Xil_Out32(TFT_FRAME_ADDR0+(4 * ((i) *
XTFT_DISPLAY_BUFFER_WIDTH + j)),0x0);
                                }
                            }
                }
                }
        return XST_SUCCESS;
}
voidhandle(void *CallBackRef, u32IntrMask, u32NumBytes)
{
    switch (IntrMask){
    case XPS2_IPIXR_RX_FULL:
        receivefinish=1;//若是接收缓冲区满产生的中断，则置receivefinish为1
        break;
    default: break;
    }
    }
unsignedcharmousesr(unsignedchar *command,unsignedchar
*buffer,intlbf){
    if(command[0]!=0)
        XPs2_Send(&ps2,command,1);
    if(lbf!=0){
        receivefinish=0;
        XPs2_Recv(&ps2,buffer,1);
        while(receivefinish==0);
        if(*buffer!=0xfa)
            return *buffer;
        elseif((lbf-1)>0)
        {
            receivefinish=0;
            XPs2_Recv(&ps2,buffer+1,lbf-1);
            while(receivefinish==0);
        }
```

```
    }
    return buffer[0];

}
```

14. 编程设置 PS2 鼠标采样率为 40 采样点/秒，并以 steam 模式读取鼠标数据，基于例 9.2 所示计算机系统在 VGA 显示器上显示一个跟随鼠标移动的小白方块，要求该小白方块的大小为 20*40 个像素，小白方块的初始值为屏幕正中间，当鼠标移动到屏幕边缘时，不再往屏幕外移动。

解答：

```
#include<stdio.h>
#include"xparameters.h"
#include"xil_exception.h"
#include"xps2.h"
#include"xstatus.h"
#include"xintc.h"
#include"xio.h"
#include"xil_io.h"
#include"xtft.h"
#define TFT_FRAME_ADDR0     XPAR_EMC_MEM0_HIGHADDR - 0x001FFFFF
#define XTFT_AR_OFFSET      0
#define XTFT_CR_OFFSET      4
#define XTFT_IESR_OFFSET 8
#define XTFT_CR_TDE_MASK 0x01
#define XTFT_CR_DPS_MASK 0x02
#define XTFT_IESR_VADDRLATCH_STATUS_MASK 0x01
#define XTFT_IESR_IE_MASK       0x08
staticXPs2 ps2;
staticXIntcintCtrl;
staticintreceivefinish;
#defineprintfxil_printf     /* A smaller footprint printf */
shortintcposx=320;
shortintcposy=240;
shortintxchange,ychange;
voidhandle(void *CallBackRef, u32IntrMask, u32NumBytes);
unsignedcharmousesr(unsignedchar *command,unsignedchar
*buffer,intlbf);
intmain(){
    XPs2_Config *ps2_config;
    int Status;
    intdispy;
    unsignedcharcommand,answer,pos[4];
    inti,j;
        Xil_Out32(XPAR_TFT_0_BASEADDR + XTFT_AR_OFFSET,
```

```
TFT_FRAME_ADDR0);
//PS2接口实例化
    ps2_config=XPs2_LookupConfig(XPAR_XPS_PS2_0_0_DEVICE_ID);
    Status=XPs2_CfgInitialize(&ps2, ps2_config,
           XPAR_XPS_PS2_0_0_BASEADDR);
//设置中断处理回调函数
    XPs2_SetHandler(&ps2, (XPs2_Handler)handle,(void *)&ps2);
    XIntc_Initialize(&intCtrl,XPAR_AXI_INTC_0_DEVICE_ID );
    XIntc_Enable(&intCtrl,XPAR_AXI_INTC_0_XPS_PS2_0_IP2INTC_IRPT_1_IN
TR);
    XIntc_Connect(&intCtrl,XPAR_AXI_INTC_0_XPS_PS2_0_IP2INTC_IRPT_1_I
NTR,
                  (XInterruptHandler)XPs2_IntrHandler,(void *)&ps2);
//中断控制器实例化
    XPs2_IntrGlobalEnable(&ps2);
    XPs2_IntrEnable(&ps2, XPS2_IPIXR_RX_FULL);
    microblaze_register_handler((XInterruptHandler)XIntc_InterruptHan
dler, (void *)&intCtrl);
    microblaze_enable_interrupts();
    XIntc_Start(&intCtrl, XIN_REAL_MODE);//启动中断控制器
    command =0xff;
    answer=mousesr(&command,pos,0);
    for(Status=0;Status<5000000;Status++);
    XPs2_Recv(&ps2,pos,3);
    command =0xff;
    answer=mousesr(&command,pos,0);
    for(Status=0;Status<5000000;Status++);
    XPs2_Recv(&ps2,pos,3);
    command =0xff;
    answer=mousesr(&command,pos,0);
    for(Status=0;Status<5000000;Status++);
    XPs2_Recv(&ps2,pos,3);
    command =0xf2;
    answer=mousesr(&command,pos,2);
    command =0xf3;
    answer=mousesr(&command,pos,1);
    command =0x28;//设置采样速率为40
    answer=mousesr(&command,pos,1);
    command =0xf2;
    answer=mousesr(&command,pos,2);
    command =0xe8;
    answer=mousesr(&command,pos,1);
    command =0x03;
    answer=mousesr(&command,pos,1);
```

```
command =0xe6;
answer=mousesr(&command,pos,1);
command =0xf3;
answer=mousesr(&command,pos,1);
command =0x28;
answer=mousesr(&command,pos,1);
command =0xf4;
answer=mousesr(&command,pos,1);
       while(1)
       {
           receivefinish=0;//循环接收streaming模式下鼠标数据报告
           XPs2_Recv(&ps2,pos,4);
           while(receivefinish==0);
           printf("%x,%x,%x,%x\r\n",pos[0],pos[1],pos[2],pos[3]);
           if((pos[0]&0x8)!=0x8){
           if ((pos[1]&0x10)==0x10)
               xchange=0xff00|pos[2];
           elsexchange=0x0000|pos[2];
           if ((pos[1]&0x20)==0x20)
               ychange=0xff00|pos[3];
           elseychange=0x0000|pos[3];
                        }
           else {
               if ((pos[0]&0x10)==0x10)
                                xchange=0xff00|pos[1];
                           elsexchange=0x0000|pos[1];
                           if ((pos[0]&0x20)==0x20)
                               ychange=0xff00|pos[2];
                           elseychange=0x0000|pos[2];
                        }
           cposx=cposx+xchange;
           cposy=cposy-ychange;
           if(cposx<0)
               cposx=0;
           if (cposx>639)
               cposx=639;
           if(cposy<0)
               cposy=0;
           if(cposy>479)
               cposy=479;
           printf("cposx %d,cposy %d\r\n",cposx,cposy);
           dispy=cposy/4;//控制显示区域
               if (dispy>90)
                   dispy=90;
```

```
                for (i = 0; i<= 120; i++) {
                                for (j = 0; j <= 640;j++) {

    if((i>dispy)&&(i<dispy+5)&&(j>cposx)&&(j<cposx+20))
                                    Xil_Out32(TFT_FRAME_ADDR0+(4 * ((i)
* XTFT_DISPLAY_BUFFER_WIDTH + j)),0x00ffffff);
                                else
                                Xil_Out32(TFT_FRAME_ADDR0+(4 * ((i) *
XTFT_DISPLAY_BUFFER_WIDTH + j)),0x0);
                                }
                            }

                }
        return XST_SUCCESS;
}
voidhandle(void *CallBackRef, u32IntrMask, u32NumBytes)
{
    switch (IntrMask){
    case XPS2_IPIXR_RX_FULL:
        receivefinish=1;//若是接收缓冲区满产生的中断，则置receivefinish为1
        break;
    default: break;
    }
    }
unsignedcharmousesr(unsignedchar *command,unsignedchar
*buffer,intlbf){
    if(command[0]!=0)
        XPs2_Send(&ps2,command,1);
    if(lbf!=0){
        receivefinish=0;
        XPs2_Recv(&ps2,buffer,1);
        while(receivefinish==0);
        if(*buffer!=0xfa)
            return *buffer;
        elseif((lbf-1)>0)
        {
            receivefinish=0;
            XPs2_Recv(&ps2,buffer+1,lbf-1);
            while(receivefinish==0);
        }
    }
    return buffer[0];

}
```