### 基础实验1 LCD绘图实验

1. 实验内容

(1) LCD显示系统

LCD显示系统如图2.6.1所示。

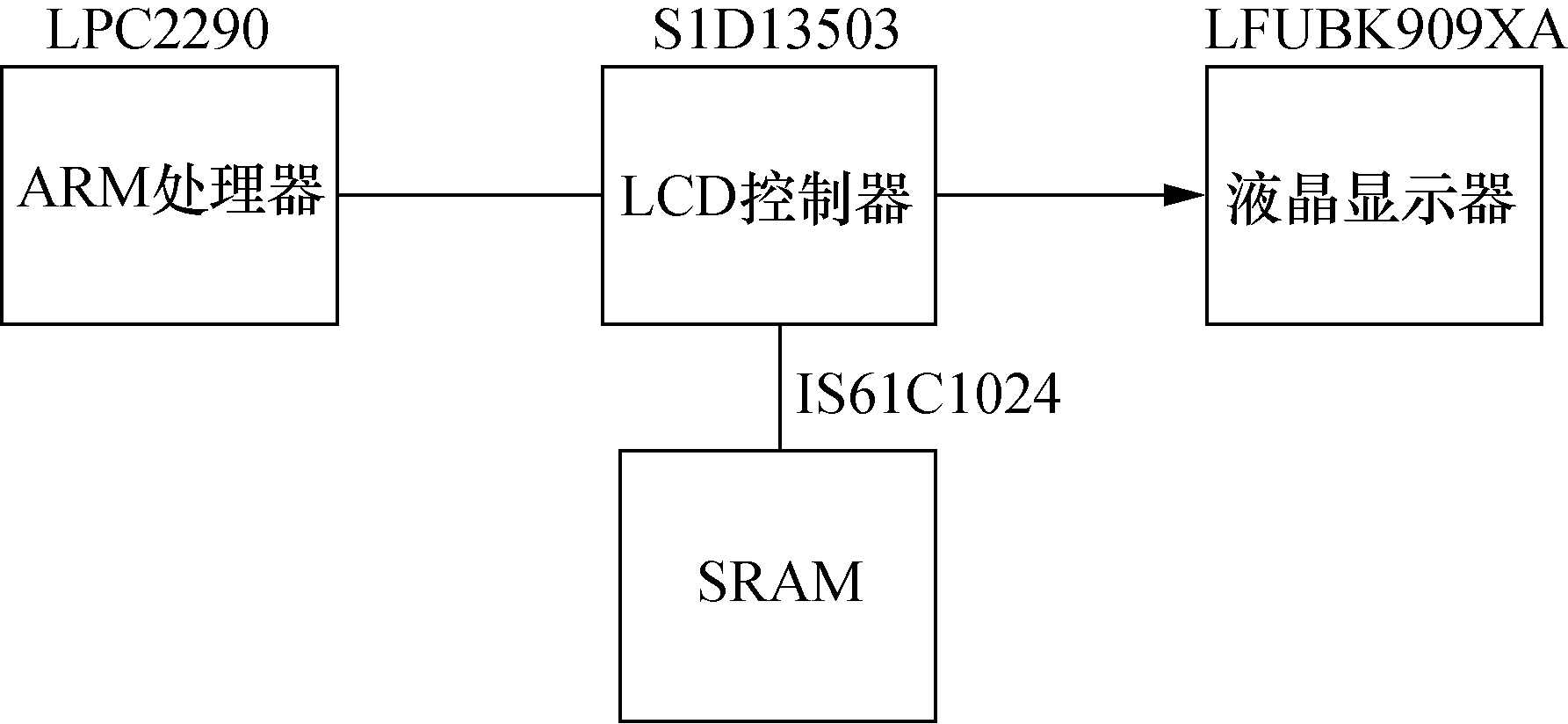


图2.6.1 LCD显示系统

(2) 实验内容

编程使用LCD驱动程序，在LCD上绘制两个波形。

① 波形一，坐标按如下表达式计算，x取值0、1、2…319：

M=-PI+2\*PI\*x/320；

y=239-(uint16)(120\*sin(M)+120)；

② 波形二，坐标按如下表达式计算，x取值0、1、2…319：

M=-PI/2+2\*PI\*x/320；

y=239-(uint16)(120\*sin(M)+120)；

(3) 编程要求

LCD配置为图形方式，分辨率320×240，彩色模式为256色。

#### 2. 实验指导

(1) LCD技术

LCD系统是一个功能强大的显示系统，它由LCD控制器、液晶显示器、视频缓冲区组成，在ARM处理器的控制器下完成文字、图像等信息显示，如图2.6.1所示。

① 液晶显示器(LCD)

液晶显示器又称LCD，是一种以液晶为材料的超薄的显示设备。液晶分子在电场作用下会发生排列上的变化，从而影响通过它的光线变化。在偏光片的作用下，使光线表现为明暗变化，从而达到显示图像的目的。

② LCD控制器

LCD控制器有很多功能，最主要的功能有两个：一是用于确定显示模式(图形模式、文本模式、图形文本模式)；二是用于定期从视频缓冲区提取显示数据，依据显示数据驱动液晶显示器显示，这就是所谓的屏幕刷新，它是由LCD控制器自动完成。

③ 视频缓冲区

视频缓冲区用于存储当前屏幕的显示数据。液晶显示器依据像素显示图像文字信息，一旦显示模式确定，屏幕水平方向、垂直方向上的可用像素也随之确定。通过控制图像和文字覆盖区域的像素发光从而达到显示图像和文字的目的，控制像素显示的依据就是显示数据。屏幕上的每个像素点在视频缓冲区内都固定位置与之对应，这个位置保存的就是显示数据。

(2) S1D13503编程

S1D13503是一个点阵彩色图像显示控制器，它可以工作在图形模式、图形文本混合模式下。工作模式可以是“1024×768黑白 1bit”、“640×480 4彩色2bits”、“640×400 16彩色4bits”、“320×240 256彩色8bits”；外接存储芯片作为视频缓冲区，接口为8位或16位，存储器容量为64K或128K。

S1D13503总共有16个控制寄存器，由AUX[n]来标记，其中n取值为0…15。每个寄存器都有一个访问地址，而且这些地址是连续的。

① AUX[0]测试寄存器

芯片S1D13503可工作在普通模式、测试模式下。它是通过配置AUX[0]位7实现的，其中，位7=0为普通模式；位7=1为测试模式。AUX[0]寄存器的地址为0x83800000，配置语句如下：

uint16 \*REG\_Point；

REG\_Point=(void \*)0x83800000；

\*(REG\_Point)=0；

语句中，REG\_Point为16位地址指针，语句“\*(REG\_Point)=0”将寄存器AUX[0]位7配置为0，使S1D13503工作在普通模式下。

② AUX[01]模式寄存器0

AUX[01]为模式寄存器0，其地址为0x83800000+1。经常使用的配置包括以下3项。

a. 开/关液晶显示器

当寄存器AUX[01]位7=0时，关闭显示器；当位7=1时，打开显示器，默认为打开显示器。

b. 显示模式

显示模式由AUX[01]位3、AUX[03]位2、AUX[03]位1共同确定，如表2.6.1所示。

表2.6.1 显示模式

|  |  |  |  |
| --- | --- | --- | --- |
| 显 示 模 式 | 灰度/彩色  AUX[01]位3 | BW/256色  AUX[03]位2 | 彩色模式  AUX[03]位1 |
| 256色 | x | 1 | 1 |
| 16色 | 1 | 0 | 1 |
| 4色 | 0 | 0 | 1 |
| 16级灰度 | 1 | 0 | 0 |
| 4级灰度 | 0 | 0 | 0 |
| BW | x | 1 | 0 |

c. LCD数据宽度

LCD数据宽度指的是LCD控制器与液晶显示器之间传送数据的宽度。LCD数据宽度由AUX[03]位3和AUX[01]位2决定，如表2.6.2所示。

表2.6.2 LCD数据线宽度

|  |  |  |  |
| --- | --- | --- | --- |
| 显 示 器 | LCD数据宽位1  AUX[03]位3 | LCD数据宽位2  AUX[01]位2 | 功 能 |
| 单色 | x | 0 | LCD数据 4位宽 |
| 单色 | x | 1 | LCD数据 8位宽 |
| 彩色 | 0 | 0 | LCD数据 4位宽 |
| 彩色 | 0 | 1 | LCD数据 8位宽，格式1 |
| 彩色 | 1 | 0 | LCD数据 16位宽(外接电路) |
| 彩色 | 1 | 1 | LCD数据 8位宽，格式2 |

d. 存储器接口LCD Data Width Bit0

存储器接口指的是与外接存储芯片的连接接口。S1D13503芯片与存储芯片之间采用8位数据线或者16位数据线连接。由AUX[01]位1配置，配置如下：

位1=0为16位存储器接口；

位1=1为8位存储器接口。

e. 寻址范围RAMS

当总线接口为8位时，此位用于确定寻址范围是8Kx8还是32Kx8，由寄存器AUX[1]位0配置。

位1=0寻址范围8Kx8；位1=1寻址范围32Kx8。

③ AUX[02]行字节计数寄存器

此寄存器用于配置水平分辨率，其地址为0x83800000+2。寄存器AUX[03]位0作为配置数据的最高位，寄存器AUX[02]各位作为配置数据的低8位。计算公式如公式2.6.1所示。

行字节数=(每像素位数/存储器接口宽度×行像素数)-1 (2.6.1)

例如，彩色模式为256色，即每像素显示数据为8位，存储器接口采用16位接口，分辨率为320×240。参考公式2.6.1，行字节数计算如下：

行字节数(十进制)=(8/16\*320)-1=159

依据公式计算结果，将AUX[03]位0配置为0，AUX[02]的位[7:0]配置为159，则显示器水平分辨率为320。

④ AUX[03]模式寄存器1

此寄存器主要用于配置节能模式和彩色模式，其地址为0x83800000+3。

a. 位[7:6]即PS位[1:0]用于配置节能模式。如表2.6.3所示。

表2.6.3 节能模式

|  |  |  |
| --- | --- | --- |
| PS1 | PS0 | 模 式 |
| 0 | 0 | 普通模式 |
| 0 | 1 | 节能模式1 |
| 1 | 0 | 节能模式2 |
| 1 | 1 | 保留 |

b. 位[2]用于彩色模式下的颜色选择。

位2=0 4色、16色

位2=1 256 色

c. 位[1]用于选择显示模式。

位[1]=0 黑白模式(BW Mode)

位[1]=1 彩色模式(Color Mode)

⑤ AUX[04]、AUX[05]显示行计数寄存器

用于配置LCD垂直分辨率，寄存器AUX[04]的地址为0x83800000+4, 寄存器AUX[05]的地址为0x83800000+5。数据一共为10位，寄存器AUX[04]位[7:0]为数据中的低8位，寄存器AUX[05]位[1:0]为数据中的高2位，计算公式如公式2.6.2所示。

像素行计数值=垂直分辨率-1 (2.6.2)

例如，显示分辨率为320×240。参考公式2.6.2，像素行计数值计算如下，像素行计数值(十进制)=240-1=239。依据计算结果，将AUX[05]位[1:0]配置为0，AUX[04]的位[7:0]配置为239，则显示器垂直分辨率为240。

⑥ AUX[06]、AUX[07]屏1视频缓冲区起始地址寄存器

屏1显示数据保存在视频缓冲区中，其中数据区16位起始地址的低8位保存在寄存器AUX[06]中，高8位保存在寄存器AUX[07]中。寄存器AUX[06]的地址为0x83800000+6，寄存器AUX[07]的地址为0x83800000+7。

⑦ AUX[08]、AUX[09]屏2视频缓冲区起始地址寄存器

在双屏模式下，屏1在显示器的上半部；屏2在显示器的下半部分。屏1、屏2显示数据都保存在视频缓冲区中，其中屏2数据缓冲区16位起始地址的低8位保存在寄存器AUX[08]中，高8位保存在寄存器AUX[09]中。屏2的起始地址计算依赖屏1，也就是说在视频区内，屏2数据的存放紧随屏1数据，屏2起始地址计算如公式2.6.3所示。

|  |  |
| --- | --- |
| 屏2起始地址=屏1起始地址+(屏1水平分辨率×屏1垂直分辨率×每像素字节数)/  (2×存储器接口宽度/8) | (2.6.3) |

寄存器AUX[8]的地址为0x83800000+8，寄存器AUX[9]的地址为0x83800000+9。

⑧ AUX[0A]、AUX[0B]屏1显示行计数寄存器

用来配置屏1的像素行数，共10位，其中AUX[0A]的位[7:0]为10位中的低8位；AUX[0B]的位[1:0]为10位中的高2位。

寄存器AUX[0A]的地址为0x83800000+0A，寄存器AUX[0B]的地址为0x83800000+0B

⑨ AUX[0C]水平非显示周期寄存器

用于调整水平非显示周期，其地址为0x83800000+0C。

⑩ AUX[0D]地址锁存调节寄存器

此寄存器用于控制虚拟显示，其地址为0x83800000+0D。图2.6.2为虚拟显示的一个实例。

虚拟显示就是保存在视频缓冲区中的一个虚拟镜像，以图2.6.2为例，它的分辨率为640×480。并且使用当前的液晶显示器作为视口(分辨率为320×240)来显示虚拟镜像的一部分。通过将视口向下、向右滚动，就可以显示全部的虚拟镜像。

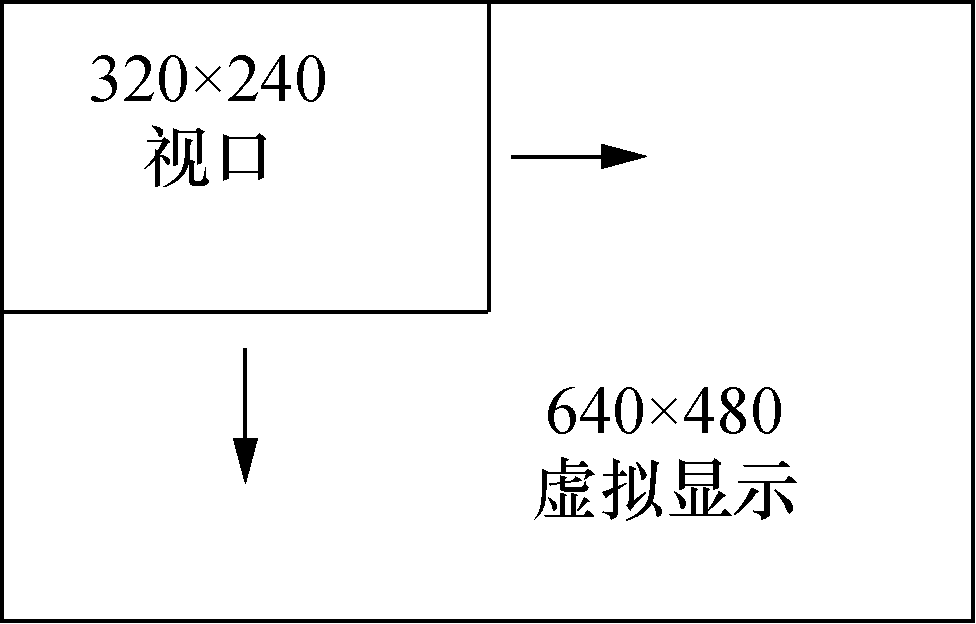


图2.6.2 虚拟显示实例

下边以实例讲述创建虚拟显示的过程。

第一，通过配置S1D13503寄存器，将LCD分辨率配置为320×240。

第二，存储器接口为16位，则以字为操作单位。

第三，确定字数据包含的像素数。

假设，彩色模式为16色，则一个像素的显示数据为4位。这样，一个字节包含2个像素数据，一个字包含4个像素数据。

第四，计算水平方向不可见的像素数，不可见像素数=640-320=320。

第五，地址锁存调节寄存器配置值=水平不可见像素数/每字像素数=320/4=80字=0×50字。

关于滚动部分，请查阅S1D13503参考手册。

⑪ AUX[0E]查找表地址寄存器

S1D13503有三个查找表，分别是红色查找表、绿色查找表、蓝色查找表，每个查找表内部有16个位置，每个位置保存一个4位数据。从每个表取出一个数据，送颜色译码器，控制按配置的颜色显示一个点。

寄存器AUX[0E]位[3:0]用于指示查找表16个位置中的一个位置，寄存器AUX[0E]的地址为0x83800000+0E。

⑫ AUX[0F]查找表数据寄存器

寄存器AUX[0F]的地址为0x83800000+0F。以256色为例，说明显示数据与三种查找表的关系，如图2.6.3所示。

256色彩色模式下，一个像素的显示数据为一个字节。其中，位[7:5]为红色数据；位[4:2]为绿色数据；位[1:0]为蓝色数据。

红色查找表包括两个BANK，分别是BANK0、BANK1。每个BANK包含8个数据项，每个数据项保存在一个位置，位置序号为0到7。显示数据中位[7:5]的值就是位置序号。AUX[0F]位6用于选择BANK，位6=0，选择BANK0；位6=1，选择BANK1。

由AUX[0F]位6确定BANK，显示数据中的位[7:5]确定数据位置，从中取出4位RED数据，送液晶显示器，控制红基色像素显示。

绿色查找表包括两个BANK，分别是BANK0、BANK1。每个BANK包含8个数据项，每个数据项保存在一个位置，位置序号为0到7。显示数据中位[4:2]的值就是位置序号。AUX[0E]位6用于选择BANK，位6=0，选择BANK0；位6=1，选择BANK1。

由AUX[0F]位6确定BANK，显示数据中的位[4:2]确定数据位置，从中取出4位GREEN数据，送液晶显示器，控制绿基色像素显示。

蓝色查找表包括4个BANK，分别是BANK0、BANK1、BANK2、BANK3。每个BANK包含4个数据项，每个数据项保存在一个位置，位置序号为0到3。显示数据中位[1:0]的值就是位置序号。AUX[0F]的位[5:4]用于选择BANK，位[5:4]=00，选择BANK0；位[5:4]=01，选择BANK1；位[5:4]=10，选择BANK2；位[5:4]=11，选择BANK3。

由AUX[0F]位[5:4]确定BANK，显示数据中的位[1:0]确定数据位置，从中取出4位BLUE数据，送液晶显示器，控制蓝基色像素显示。

至此，S1D13503依据显示数据，控制三基色像素显示，从而达到显示文字与图像的目的。



图2.6.3 显示数据与查找表

(3) LCD驱动程序

一个嵌入式开发板包括许多硬件功能模块，每个模块都有很多功能，为了实现这些功能，需编写若干子程序，将这些子程序以及常量、变量分别存放到两个文件，一个是头文件(\*.h)；一个是源程序文件(\*.c)，这就是最简单的硬件驱动程序。

文件lcddriver.h和lcddriver.c就是S1D13503LCD控制器的驱动程序，实现的主要常量、函数包括以下几个方面。

① 常量，定义在lcddriver.h文件中

文件内容如下。

#define TCOLOR uint8 //颜色数据类型

#define GUI\_LCM\_XMAX 320 //水平方向分辨率

#define GUI\_LCM\_YMAX 240 //垂直方向分辨率

#define S1D13503\_REG 0x83800000 //S1D13503寄存器组首地址

#define S1D13503\_DAT 0x83400000 //视频缓冲区首地址

//======== 颜色表 ========

#define BLACK 0x00 //黑色 0, 0, 0

#define NAVY 0x02 //深蓝色 0, 0, 128

#define DGREEN 0x10 //深绿色 0, 128, 0

#define DCYAN 0x12 //深青色 0, 128, 128

#define MAROON 0x80 //深红色 128, 0, 0

#define PURPLE 0x82 //紫色 128, 0, 128

#define OLIVE 0x90 //橄榄绿 128, 128,0

#define LGRAY 0xCA //灰白色 192, 192, 192

#define DGRAY 0x92 //深灰色 128, 128, 128

#define BLUE 0x03 //蓝色 0, 0, 255

#define GREEN 0x1C //绿色 0, 255, 0

#define CYAN 0x1F //青色 0, 255, 255

#define RED 0xE0 //红色 55, 0, 0

#define MAGENTA 0xE3 //品红： 255, 0, 255

#define YELLOW 0xFC //黄色： 255, 255, 0

#define WHITE 0xFF //白色： 255, 255, 255

② LCM初始化数据，定义在lcddriver.c文件中

文件内容如下。

uint8 const INIT\_TAB[14] =

{

0x00, // AUX[0]=00000000B(正常工作模式，即非测试模式)

0xBD,//AUX[1]=10111101B(模式控制：打开显示，单屏，屏蔽XSCL水平非显示周期输出，使能LCDENB，彩色8位LCD数据--无XCSL2，16位显示存储数据总线方式)

159, // AUX[2]=159每行显示占用的存储器字节数，即列参数

0x0E, // AUX[3]= 00001110B(非节功模式，256彩色方式)

239, // AUX[4]= 239 (总行数)

0, // AUX[5]= 0(WF翻转周期，为0时表示每一帧WF输出翻转一次)

0x00, //AUX[6]=00H(屏1视频缓冲区首地址低8位)

0x00, //AUX[7]=00H(屏1视频缓冲区首地址高8位)

0x00, //AUX[8] =0H(屏2视频缓冲区首地址低8位)

0x00, //AUX[9]=00H(屏2视频缓冲区首地址高8位)

239, //AUX[A]=239(行数，低8位)

0, //AUX[B]=0(行数，高2位)

10, //AUX[C]= 10(水平默认不显示周期)

0 //AUX[D]= 0(显示域冗余宽度，正常操作模式)

}；

③ 函数GUI\_Initialize()

此函数用于初始化LCM。它调用两个函数，分别是LCD\_Initialize()、GUI\_FillSCR(0x00)。

④ 函数LCD\_Initialize()

此函数用于配置显示模式与调色板，它调用两个函数，分别是S1D13503\_Init()、S1D13503\_LutInit()。

⑤ 函数S1D13503\_Init()

此函数将表INIT\_TAB的值写入以0x83800000为首地址的连续14个单元中，即分别存入寄存器AUX[0]…AUX[D]中，从而将LCD显示模式配置为分辨率为320×240，彩色模式为256色。

⑥ 函数S1D13503\_LutInit()

此函数用于配置调色板。调色板包括红色调色板、绿色调色板和蓝色调色板，每个调色板都有16个位置，每个位置的数据都是4位二进制数据。配置过程为，使用AUX[0E]寄存器确定调色板位置，然后使用AUX[0F]寄存器依次向红色调色板、绿色调色板、蓝色调色板写入数据。

红、绿、蓝三色数据，共12位，包括212=4096种。使用调色板可以输出28=256色。也就是说调色板一旦配置完成，可使用的颜色为256种，它为4096种颜色的一部分。

⑦ 函数Coordinate

此函数的功能是绘制一个坐标系。调用函数GUI\_HLine()画一条水平线，线起始点坐标(0,120)，结束点坐标(319, 120)。调用函数GUI\_RLine()画一条垂直线，线起始点坐标(160, 0)，结束点坐标(160, 239)。这样坐标系绘制完成，坐标原点为点(160, 120)。

⑧ 函数GUI\_RLine()

此函数的功能是画一条垂直线。

a. 函数声明

代码如下：

void GUI\_RLine(uint32 x0, uint32 y0, uint32 y1, TCOLOR color)；

b. 入口参数

(x0,y0)为直线起始点坐标。(x0,y1)为直线结束点坐标。

c. 出口参数

无。

⑨ 函数GUI\_HLine()

此函数的功能是画一条水平线。

a. 函数声明

代码如下：

void GUI\_HLine(uint32 x0, uint32 y0, uint32 x1, TCOLOR color)；

b. 入口参数

(x0, y0)为直线起始点坐标。(x1, y0)为直线结束点坐标。

c. 出口参数

无。

⑩ 函数GUI\_FillSCR()

此函数的功能是用选定的颜色填充整个屏幕。

a. 函数声明

代码如下：

void GUI\_FillSCR(TCOLOR dat)；

b. 入口参数

dat为填充颜色。

c. 出口参数

无。

⑪ 函数GUI\_Point()

此函数用于在显示屏上画一个点。

a. 函数声明

代码如下：

uint8 GUI\_Point(uint32 x, uint32 y, TCOLOR color)；

b. 入口参数

(x,y)为点的坐标；color为画点颜色。

c. 出口参数

无。

⑫ 驱动程序安装

为LCD控制器S1D13503开发的功能子程序，它的声明部分在文件lcddriver.h内，实现部分在文件lcddriver.c内。如果编程要使用这些函数，则必须先将驱动程序加入工程，安装驱动程序的步骤如下。

a. 将驱动程序文件lcddriver.h、lcddriver.c复制到工程存储目录的“/src”子目录下。

b. 打开工程管理窗口，参考图1.2.6。在工程管理窗口中右击，弹出菜单。然后，选择Add Files命令，显示文件选择窗口。最后，通过文件选择窗口，选中路径“工程存储目录\src”下的文件lcddriver.h、lcddriver.c，单击“打开”按钮，将驱动程序加入工程。

c. 在工程管理窗口内，打开文件“config.h”，在文件末尾加入驱动程序引用“#include "lcddriver.h"”的声明。

至此，驱动程序安装完毕。随后，在编程时就可以使用驱动程序中的函数了。

(4) 编程策略

主程序以while死循环为程序框架。在程序框架内，首先，调用函数Coordinate()绘制坐标系。然后，绘制两条正弦波曲线。第一条曲线，线上点的坐标为(x,y)，其中，x取值为0到239；y的取值为239-(uint16)(120\*sin(-PI+2\*PI\*i/320)+120)。第二条曲线，线上点的坐标(x,y)，其中，x取值为0到239；y的取值为239-(uint16)(120\*sin(-PI/2+2\*PI\*i/320)+120)。

(5) 调试策略

在语句“GUI\_Point(x,y, BLACK)；”处设置一断点，记为断点1。单击按钮，执行程序，程序停在断点1处。单击按钮，单步执行程序，跟踪进入函数GUI\_Point内部。随后，使用单步执行的方法调试程序。注意观察两个问题，一是观察点坐标(x,y)与保存点显示数据的视频缓冲区地址对应关系；二是显示数据写入视频缓冲区时，屏幕如何显示点。

### 基础实验2 LCD汉字显示实验

#### 1. 实验内容

(1) 实验内容

16×16点阵汉字“之”的字模信息如下：

0×02，0×00，0×01，0×00，0×00，0×88，0×7F，0×FC，0×00，0×08，0×00，0×10，0×00，0×20，0×00，0×40，0×00，0×80，0×01，0×00，0×06，0×00，0×18，0×00，0×28，0×00，0×E7，0×00，0×40，0×FE，0×00，0×00。

16×8点阵字符“A”的字模信息如下：

0×00，0×00，0×00，0×10，0×10，0×18，0×28，0×28，0×24，0×3C，0×44，0×42，0×42，0×E7，0×00，0×00。

依据字模信息，编写汉字字符显示程序。

(2) 编程要求

除了以正常方式显示汉字外，还必须实现按逆时针旋转90度、180度、270度的方式显示汉字。

#### 2. 实验指导

(1) 汉字显示技术

以点阵字库HZK16为例说明汉字显示技术。

通常情况下，在文件中使用2字节数据保存一个汉字，这2字节数据就是汉字的机内码。机内码中每个字节的最高位都是“1”，用来区别ASCII字符，也就是说，最高位为“1”的数据为汉字机内码数据，最高位为“0”的数据为ASCII字符数据。

点阵字库HZK16以区位来组织汉字字模信息，字库按区来划分，每个区都有一个编号，称为区号。区中包括94个汉字，每个汉字都有一个固定位置，称为位号。在字库中，是通过区号、位号来确定汉字字模信息位置的，确定汉字字模信息位置如公式2.6.3、公式2.6.4所示。

汉字字模信息位置=((区号-1)×94+位号-1)×32 (2.6.3)

其中，区号小于16；区号=机内码高字节-160；位号=机内码低字节-160。

汉字字模信息位置=((区号-7)×94+位号-1)×32 (2.6.4)

其中，区号大于等于16。区号=机内码高字节-160；位号=机内码低字节-160。

下边介绍一下汉字字模信息，如图2.6.4所示。



图2.6.4 汉字“之”字模信息

显示屏内，选择一正方形区域，共16行，每行16个像素。在这个区域内，控制“之”字覆盖的像素发光，其他像素不发光，结果会在屏幕上显示一个“之”字，这就是16×16点阵汉字的显示原理。如图2.6.4所示，每行16个像素，记录为2字节数据。第一字节记录左侧8个像素信息；第二个字节记录右侧8个像素信息。数据中的每一位对应一个像素，如果像素发光，则数据中对应像素的位记录为1，否则，记录为0。比如，第一行，从左向右数，第7个像素发光，其他像素不发光，则2字节数据为0x02、0x00，其中，8个像素与数据位的对应关系是像素从左向右，依次对应位7、位6、…、位0。将所有的行数据按像素行从上到下的排列顺序组合在一起，共32字节，这就是“之”字的字模信息。

对于嵌入式系统来说，如果系统有足够资源存储汉字字库，则汉字显示过程如下：首先，根据机内码计算汉字字模信息位置；然后，依据字模信息位置提取汉字字模信息；最后，依据字模信息显示汉字。

如果系统没有足够资源存储汉字字库，则先要编写一个台式机程序，从字模库中提取系统用到的汉字字模信息，组成一个小字库，将小字库存入系统中；创建一个对应关系表，通过汉字机内码查询此表，可获取汉字在小字库中字模信息的位置；然后，依据字模信息位置提取汉字字模信息；最后，依据字模信息显示汉字。

(2) 编程策略

本程序主要依据“之”字字模信息显示汉字“之”；依据字符“A”字模信息显示字符“A”。程序由一个主函数和6个功能子函数组成。

① 函数point()

此函数的功能是在屏幕(x,y)位置显示一个点。

本例屏幕的分辨率配置为320×240，彩色模式配置为256色。这样，每个像素的显示数据就是一个字节。视频缓冲区内，按照屏幕像素从左至右、从上至下的顺序存储显示数据。像素与显示数据存储地址的对应关系如公式2.6.5所示。

存储地址=视频缓冲区首址+y×320+x (2.6.5)

其中，视频缓冲区首地址为0x83400000。只要将显示数据写入此地址，在屏幕上(x,y)位置就会显示一个点，函数实现查阅参考程序。

② 函数ShowChinese16()

a. 函数功能

是在屏幕上依据字模信息显示汉字。

b. 入口参数

参数x,y为汉字在屏幕上的位置；buf为汉字字模信息；color为汉字显示颜色。

c. 出口参数

无。

d. 显示策略

首先，使用一个for循环控制显示16行；然后，在循环体内，先提取行的第一字节数据，依据数据在屏幕显示当前行的前8个点；最后，再提取行的第二字节数据，依据数据在屏幕显示当前行的后8个点。

字节数据的提取顺序为从位7到位0。当位值=1时，显示点；当位值=0时，不显示点。第一字节数据中位值对应屏幕点的位置坐标计算为x=x+7-位序；y=y+行号；第二字节数据中位值对应屏幕点的位置坐标计算为x=x+15-位序；y=y+行号。函数实现查阅参考程序。

③ 函数ShowChar16()

a. 函数功能

是在屏幕上依据字模信息显示字符。

b. 入口参数

参数x,y为汉字在屏幕上的位置；buf为字符字模信息；color为字符显示颜色。

c. 出口参数

无。

d. 显示策略

首先，使用一个for循环控制显示16行；然后，在循环体内，提取行的字节数据，依据数据在屏幕显示当前行的8个点。

字节数据的提取顺序为从位7到位0。当位值=1时，显示点；当位值=0时，不显示点。字节数据中位值对应屏幕点的位置坐标计算为x=x+7-位序；y=y+行号。函数实现查阅参考程序。

④ 函数ShowChinese16\_Rotate90

a. 函数功能

是在屏幕上依据字模信息按逆时针方向旋转90度显示汉字。

b. 入口参数

参数x,y为汉字在屏幕上的位置；buf为汉字字模信息；color为汉字显示颜色。

c. 出口参数

无。

d. 显示策略

将汉字按逆时针旋转90度，使汉字原来的第一行变成第一列；第二行变成第二列；…；第16行变成第16列。

首先，使用一个for循环控制显示16列；然后，在循环体内，先提取列的第一字节数据，依据数据在屏幕纵向显示当前列的前8个点；最后，再提取行的第二字节数据，依据数据在纵向屏幕显示当前列的后8个点。

字节数据的提取顺序为从位7到位0。当位值=1时，显示点；当位值=0时，不显示点。第一字节数据中位值对应屏幕点的位置坐标计算为x=x+列号；y=y+8+位序；第二字节数据中位值对应屏幕点的位置坐标计算为x=x+列号；y=y+位序。函数实现查阅参考程序。

其他函数的实现请查阅参考程序。

⑤ 主程序main()

首先，调用函数GUI\_Initialize()对LCD控制器初始化，将显示模式配置为320x240、256色；并且配置画图使用的调色板。

然后，调用GUI\_FillSCR (WHITE)将屏幕背景填充为白色。

最后，调用函数ShowChinese16()显示“之”字；调用函数ShowChar16()显示“A”字符；调用函数ShowChinese16\_Rotate90逆时针旋转90度显示“之”字；调用函数ShowChinese16\_Rotate180逆时针旋转180度显示“之”字；调用函数ShowChinese16\_Rotate270逆时针旋转270度显示“之”字。

(3) 调试策略

在语句“ShowChinese16(30,30,(unsigned char \*)fontchinese,RED)；”处设置一断点，记为断点1。单击按钮，执行程序，程序会停在断点1处。单击按钮，单步执行程序，跟踪进入函数ShowChinese16内部。随后，使用单步执行的方法调试程序。注意观察两个问题，一是计算观察字模数据与点坐标(x,y)的对应关系；二是显示时，观察显示点相对于汉字的位置。

其他函数调试方法类似。

### 基础实验3 触摸屏实验

#### 1、实验内容

(1) 触摸屏接口电路

触摸屏接口电路如图2.7.1所示。

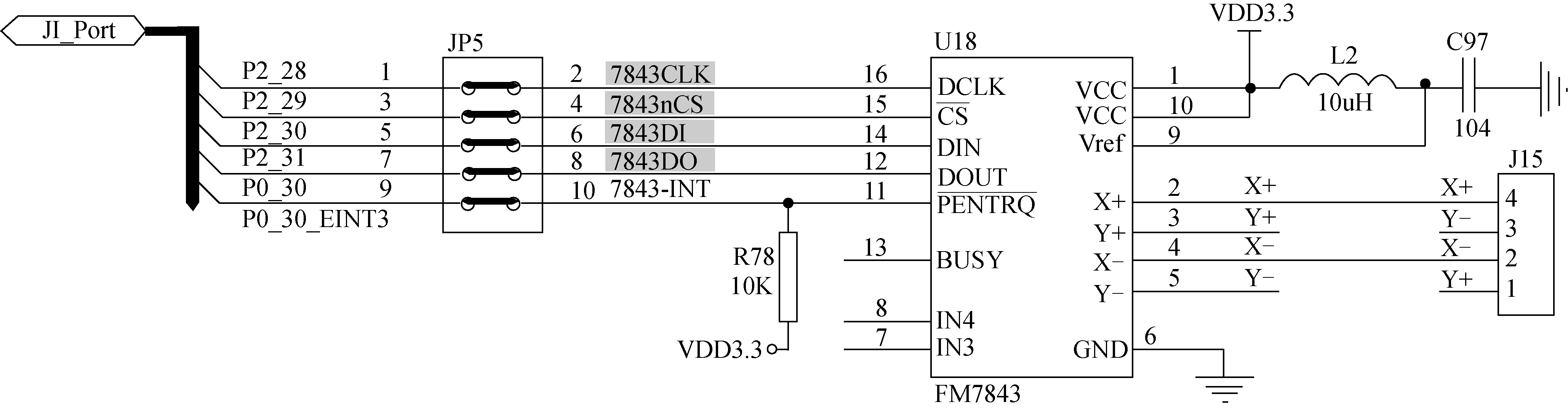


图2.7.1 触摸屏接口电路

(2) 实验内容

使用触摸笔单击触摸屏，触摸点垂直对应LCD上的一个点。在LCD上围绕此点显示一个方块，即除了显示此点外，还在上、下、左、右、右上、右下、左上、左下各显示一个点。

(3) 编程要求

LCD配置为图形方式，分辨率320×240，彩色模式为256色。

#### 2、实验指导

(1) 触摸屏技术

触摸屏系统是一个屏幕触摸定位系统，类似鼠标的功能。它由触摸屏控制器、触摸屏组成，如图2.7.1所示。它可在ARM处理器的控制器下实现显示器坐标定位。

① 触摸屏

触摸屏是一种透明的绝对定位系统，它被放置在LCD表面。当用触摸笔或手指接触触摸屏时，它的位置信息经信号线X+、X-、Y+、Y-输出，信号线X+、X-、Y+、Y-输出的是模拟直流电压信号。

② 触摸屏控制器FM7843

触摸屏控制器的作用是在ARM处理器的控制下，对触摸屏输出信号进行A/D转换，将用电压表示的触摸位置信息转换成用数字表示的位置信息，并将数字信息发给ARM处理器。

(2) 触摸屏接口电路

① 接口电路组成

参考图2.7.1。触摸屏接口电路中，包括触摸屏控制器芯片FM7843；短路块JP5；电阻R78、电感L2、无极性电容C97；连接插头J15；电源正VDD3.3；地GND；P2口的引脚P2.28、P2.29、P2.30、P2.31；P0口的引脚P0.30。

② 信号连接

参考图2.1.7。LPC2290的引脚P2.28经过短路块JP5与FM7843的DCLK连接，用于传送数据时，提供时钟信号；引脚P2.29经过短路块JP5与FM7843的连接，用于控制选中芯片FM7843；引脚P2.30经过短路块JP5与FM7843的DIN连接,用于向FM7843发送读取触摸屏X(或Y)坐标命令；引脚P2.31经过短路块JP5与FM7843的DOUT连接，用于接收触摸屏采集的X(或Y)坐标数据；引脚P1.30经过短路块JP5与FM7843的连接，用于识别FM7843的触摸中断信号。

FM7843的X+、X-、Y+、Y-引脚与触摸屏的X+、X-、Y+、Y-连接，用于采集触摸屏的位置信号。

③ 工作原理

首先，ARM处理器LPC2290通过引脚P2.29选中芯片FM7843；其次，通过P2.28、P2.30引脚向FM7843发送采集触摸屏X方向(或Y方向)的坐标命令；FM7843收到命令后，对引脚X+、X-、Y+、Y-进行A/D转换，将触摸点电压位置信号转换为数字位置信号；然后，ARM处理器通过引脚P2.31、P2.28接收FM7843采集的数据；最后，通过引脚P2.29撤销对芯片FM7843的选中。

(3) 触摸屏驱动程序

为FM7843编写的功能函数分别放在文件ads7843.h、ads7843.c中，其中函数声明部分放在文件ads7843.h中；函数实现部分放在文件ads7843.c中。

① 函数ADS7843\_WriteRead(uint8 data)

a. 功能

将读取触摸屏X(或Y)坐标命令发送至FM7843，然后，接收来自FM7843的触摸屏坐标数据。

b. 入口参数

data为入口参数，是读取触摸屏X(或Y)坐标的命令。

c. 出口参数

出口参数为12位二进制坐标数据。

② 函数GPIO\_Initialize()

a. 功能

将引脚P0.7配置为GPIO输出；将引脚P2.28、P2.29、P2.30配置为GPIO输出；将引脚P2.31、P0.30配置为GPIO输入。

b. 入口参数

无。

c. 出口参数

无。

③ 函数Calibration()

a. 功能

触摸屏与显示器是两个独立的设备，各自具有自己的坐标定位体系。实际应用过程中，功能选择和数据录入都是看着显示器，操作触摸屏完成的，输入大都借助触摸屏实现的。这种情况下，需要确定触摸屏坐标与显示器坐标之间的转换关系。此函数就是获取转换的已知条件，获取的过程如下，首先在LCD左上角(19，19)处，显示一个“十”字，使用触摸笔透过触摸屏单击“十”字，获取触摸屏坐标(errx1,erry1)；然后，在LCD左上角(299，219)处，显示一个“十”字，使用触摸笔透过触摸屏单击“十”字，获取触摸屏坐标(errx2, erry2)。得到已知条件为：触摸屏上的点(errx1,erry1)垂直对应LCD上的点(19,19)；触摸屏上的点(errx2, erry2)垂直对应LCD上的点(299,219)。这样，任意位置单击触摸屏(errx, erry)，都可以将其转换为与它垂直对应的LCD坐标(x,y)。

b. 入口参数

无。

c. 出口参数

触摸点(errx1,erry1)、触摸点(errx2, erry2)。

④ 函数GetLCD\_XY()

a. 功能

获取触摸屏坐标。

b. 入口参数

无。

c. 出口参数

触摸屏坐标(\*vx,\*vy)。

⑤ 函数Beep()

a. 功能

控制蜂鸣器响一声。

b. 入口参数

无。

c. 出口参数

无。

⑥ 主函数main()

首先，调用函数GPIO\_Initialize()初始化管脚；调用函数GUI\_Initialize()初始化LCD；调用函数GUI\_FillSCR(WHITE)，将整个屏幕填充为白色。

然后，调用函数Calibration()获取两个已知条件：一个是垂直对应LCD点(19,19)的触摸屏坐标(errx1,erry1)；另一个是垂直对应LCD点(299,219)的触摸屏坐标(errx2,erry2)。

最后，为while死循环。循环体内先调用函数GetLCD\_XY()获取当前触摸屏上触摸点坐标(errx,erry)；其次，将其转换成显示器坐标(x,y)，转换如公式2.7.1所示，

x=(errx-errx1)×(299-19)/(errx2-errx1)+20

y=(erry-erry1)×(219-19)/(erry2-erry1)+20 (2.7.1)

最后，围绕点(x,y)，左上、左下、上、下、左、右、右上、右下显示9个点(包含此点)。

(4) 调试策略

在语句“while(Calibration() != 1)；”处设置一断点，记为断点1。单击按钮，执行程序，程序停在断点1处。单击按钮,单步执行程序，跟踪进入函数Calibration内部。随后，使用分块结合单步执行的方法调试程序，注意观察两个问题：一是观察读取的触摸屏坐标(sumx,sumy)；二是判断触摸屏坐标是否在允许范围内，从而理解函数Calibration()的功能。

在语句“GetLCD\_XY(&errx, &erry)；”处设置一断点，记为断点2。单击按钮，执行程序，程序停在断点2处。单击按钮，单步执行程序，跟踪进入函数GetLCD\_XY内部。随后，使用分块结合单步执行的方法调试程序。注意观察如何读取触摸屏坐标。