
基于 arm 的两机通讯（含界面）

目录

一、	实验目的	1
二、	实验要求	1
三、	实验设备	1
四、	实验设计思路和步骤	1
1、	相关硬件连接.....	1
2、	实验软件架构设计.....	2
2.1	实验思路设计	2
2.2	软件设计流程图	3
3、	所需原理简介	4
五、	程序调试详解	5
1、	创建文件.....	5
2、	配置文件.....	5
3、	代码详述.....	6
六、	代码调试	9
1、	编辑窗口文件重叠.....	9
2、	头像图片无法任意设置位置.....	9
3、	Keypad 无反应	9
4、	图片出现 image2lcd.....	9
5、	出现两张图片，并且位置出错.....	9
6、	Uart 串口每次只能发一个数据	9
七、	实验总结	10
八、	附录	11
1、	Lcd_test.c	11
2、	keypad.c	14
3、	uart.c.....	15
4、	glib.c.....	16

一、 实验目的

通过实验掌握 ARM 嵌入式软件开发与调试技术。

通过实验掌握 S3C2410X 的中断控制寄存器的使用，掌握 S3C2410X 处理器的中断响应过程，掌握 ARM 处理器的中断方式和中断处理过程，掌握 ARM 处理器中断处理的软件编程方法。

通过实验掌握使用 μ Vision IDE 辅助信息窗口来分析判断调试过程和结果，学会查找软件调试时的故障或错误，掌握使用 μ Vision IDE 开发工具进行软件开发与调试的常用技巧。

通过实验了解 S3C2410X 处理器 UART 相关控制寄存器的使用；掌握 ARM 处理器串行通信的软件编程方法；了解 IIC 串行数据通信协议的使用。

通过实验初步掌握液晶屏的使用及其电路设计方法；掌握 S3C2410X 处理器的 LCD 控制器的使用；掌握液晶显示文本及图形的方法与程序设计。

二、 实验要求

实现两台 arm 机的通讯，具体要求如下：

1. 第一页 LCD 屏幕上显示本人图像、学号、姓名等信息。
2. 换屏以微信或 QQ 界面的对话方式呈现，里面分输入框和输出框。
3. 每次键盘输入，只有当按了“发送”键后，才能在输出框中显示，且在另一台机中显示发送的内容。
4. 两台机执行同一程序，不能出现死机现象。

三、 实验设备

硬件：两台全模块 arm 实验平台，两套 ULINK2 仿真器套件，PC 机，串口。

软件： μ Vision IDE for ARM 集成开发环境，Windows 98/2000/NT/XP。

四、 实验设计思路和步骤

1、 相关硬件连接

使用串口连接两台 arm 实验平台。串口硬件电路如下图 1：

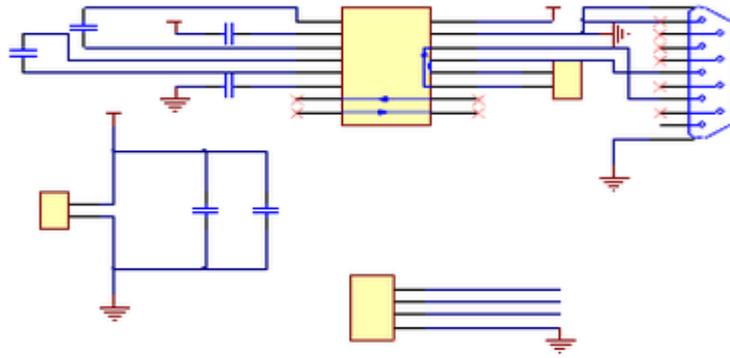


图 1 RS232 电路原理图

2、实验软件架构设计

2.1 实验思路设计

- 程序启动后，首先开启 LED 液晶显示屏，显示一张带有本人姓名学号等信息的本人照片。
- 在该界面停留五秒后自动进入聊天对话框界面，该界面分为编辑文字区，发送文字区和接收文字区。
- 当使用 keypad 输入键值时，触发中断，将该键值发送至对话框编辑文字区显示，若按下 0 键，则在屏幕上显示删除前一位输入的字符，当按下 1 键时，则发送目前在编辑区编辑的字符串至发送区和另一个机器。
- 发送区显示的字符串周边带有文字框，并且每一次发送，字符串显示在原字符串的下一行，接收区同理。
- 字符串接收功能通过 UART 中断触发，对象机使用 UART 的发送函数发送字符串时中断触发，将该字符串赋值给一个新的数组，并且将该数组发送至对话框接收文字区显示。

2.2 软件设计流程图

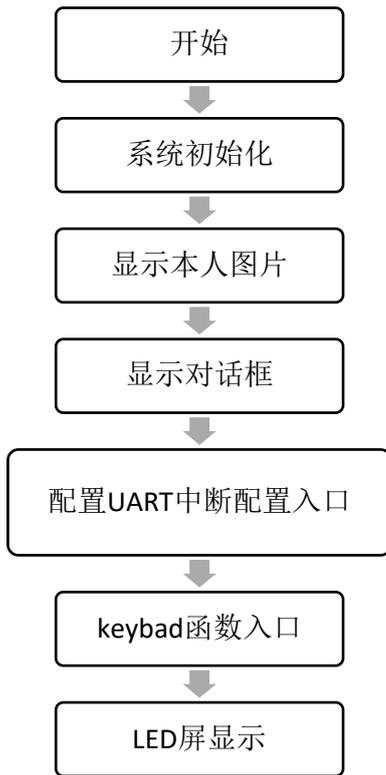


图2 主程序流程图

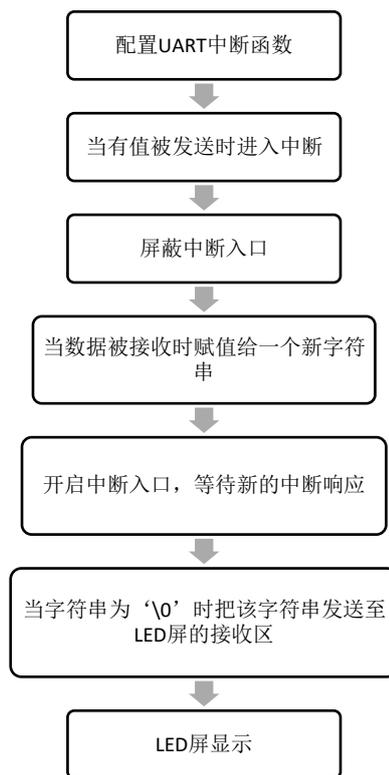


图3 接收字符串流程图

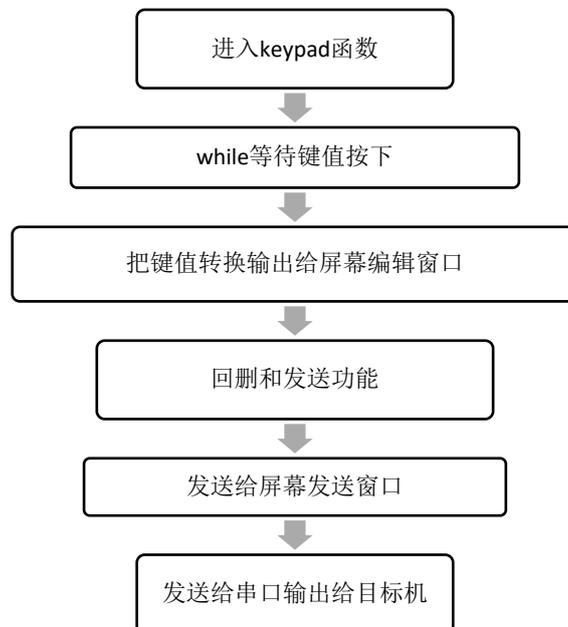


图4 发送字符串流程图

3、所需原理简介

- TFT LCD

S3C2410X LCD 控制器用于传输显示数据和产生控制信号。我们可以通过 LCD 显示我们自己设置的背景图片，通过 LCD 制作我们想要的 UI 界面，将我们从串口和 Keyboard 接收到的字符传到指定的位置显示。

- UART

我们使用 UART 单元提供的 UART1 异步串行通信接口，用于数据的接收。

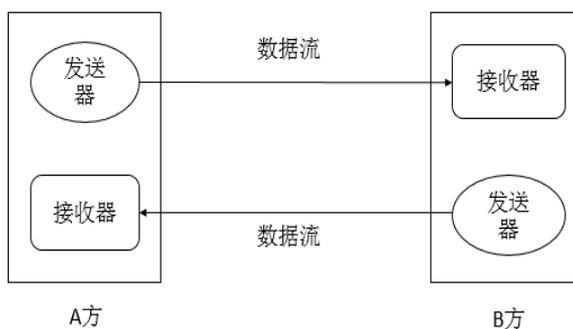


图 5 UART 通信操作

- 中断

S3C2410X 的中断控制器可以接受多达 56 个中断源的中断请求。S3C2410X 的中断源可以由片内外设提供，比如 DMA、UART、IIC 等，其中 UARTn 中断和 EINTn 中断是逻辑或的关系，它们共用一条中断请求线。本次我们使用 UART 中断。

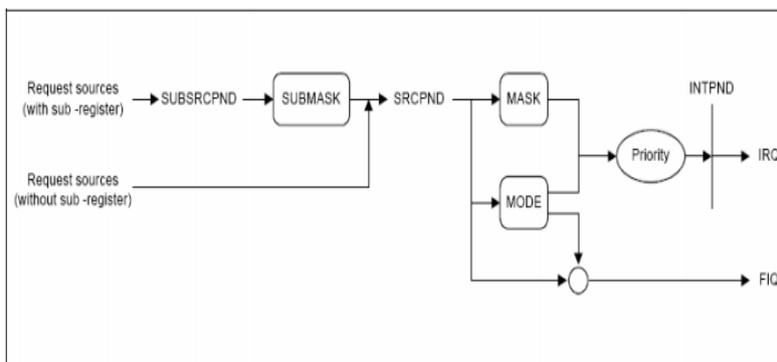


图 6 中断处理图

Register	Address	R/W	Description	Reset Value
SRCPND	0x4A000000	R/W	中断标志寄存器	0x00000000
INTMOD	0x4A000004	R/W	中断模式寄存器	0x00000000
INTMSK	0x4A000008	R/W	中断屏蔽寄存器	0xFFFFFFFF
PRIORITY	0x4A00000C	R/W	中断优先级寄存器	0x7F
INTPND	0x4A000010	R/W	中断服务寄存器	0x00000000
INTOFFSET	0x4A000014	R	中断偏移寄存器	0x00000000
SUBSRCPND	0x4A000018	R/W	子源挂起寄存器	0x00000000
INTSUBMSK	0x4A00001C	R/W	中断子源屏蔽寄存器	0x7FF

图 7 中断寄存器

五、 程序调试详解

1、 创建文件

根据实验要求,用到的主要程序有几个部分分别是 UART、Keypad 以及 LCD 显示,并且使用 UART 中断。因为需要用到 UART、Keypad、LCD 主要的库文件,因此将官方提供的 UART、Keypad、LCD 源文件和头文件内容以及系统文件如 S3C2410.s、SDRAM.ini、2410lib.c、sys_init.c 集成到当前新的文件夹。

2、 配置文件

在程序编译前需要给程序配置路径以及存储空间、地址、烧写方式等。

- 起始地址及储存空间: 在 keil4 目录中点击 Option for Target, ROM 起始为 0x30000000, SIZE 为 0x300000; RAM 起始为 0x30300000, SIZE 为 0x4000000; IRAM1 起始为 0x40000000, SIZE 为 0x1000。
- 源文件调用路径: 在 C/C++选项中, include paths 为源文件路径,需在此配置本机源文件的地址。
- 烧写方式: Debug 选项中,点击 Load Application at Startup、Run to main、ULINK ARM Debugger。

配置以上路径以及各个选项后,程序便能按照要求运行编译。其中 RAM 空间配置、include paths 配置最为重要,影响着硬件运行情况。配置界面如图 10 所示。

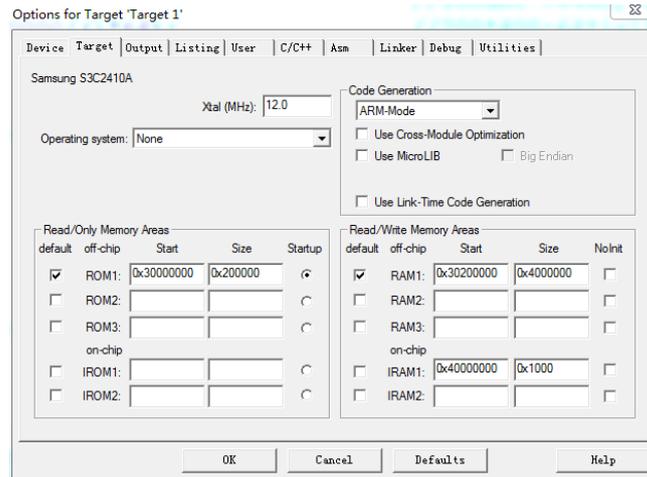


图 8 路径配置

根据实验要求，配置好了地址和存储地址，根据需求将各个子程序放在一个文件夹，以自己名字 lsc 命名，如图 2-3 所示，需要更改以下三个位置。

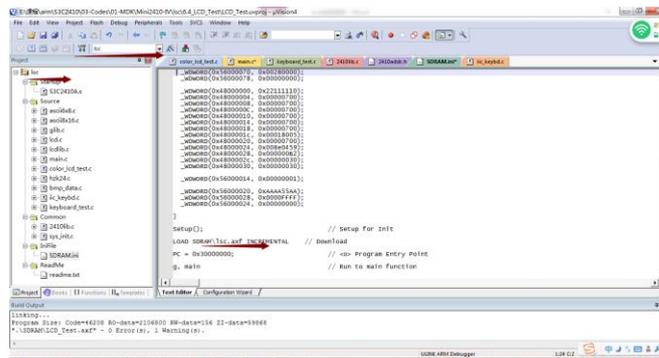


图 9 文件名设置

3、代码详述

主函数先系统初始化所有函数，配置好 UART 端口为 COM1，波特率 115200 等。配置中断函数的内容如清除所有中断、建立中断地址表、设置触发方式为下降沿触发，使能 UART1 中断，屏蔽所有子中断，使能 UART1 的 RXD 中断。在中断地址表中，当外部有信息通过 UART 通讯时候，触发中断函数，中断函数首先将子中断关闭，对接收到数据进行分析、发送给 LCD 中显示。然后清除中断、挂起、开启中断，从而继续接收外部信息这样周而复始的中断响应。在中断函数下面是 keyboard_test 函数对 keyboard 初始化，等键盘给响应。

程序大致如下内容所示：

- 主函数

- 开始

- 系统初始化
 - 显示本人图片
 - 显示对话框
 - 配置 UART 中断配置入口
 - keyboard 函数入口
 - LED 屏显示
- **UART 中断具体配置方式如下所示：**
- 清除所有中断挂起
 - 将中断函数入口地址送入中断向量表
 - 清除所有外部中断挂起
 - 使能 uart1 中断
 - 设置触发方式为下降沿触发
 - 使能 UART1 中断
 - 屏蔽所有子中断
 - 使能 UART1 的 RXD 中断
- **UART1 内容**
- 配置 UART 中断函数
 - 当有值被发送时进入中断
 - 屏蔽中断入口
 - 当数据被接收时赋值给一个新字符串
 - 开启中断入口，等待新的中断响应
 - 当字符串为 ‘\0’ 时把该字符串发送至 LED 屏的接收区
- **Keypad 内容**
- 等待按键
 - 转换键值

- 发送给 LCD 屏幕
- 本人信息图片 LCD 屏显示
 - 图片显示
 - 汉字显示
- 聊天界面 LCD 屏显示
 - 图片显示
 - 使用直线划分编辑区和非编辑区
- 字符串发送 LCD 屏显示
 - 在 keypad 处取得键值并赋值给一个字符串 str1 发送至编辑区
 - 回删功能——判断是否为字母 ‘E’，若是则回删
 - 发送功能——判断是否为字母 ‘FUN’，若是则把 str1 赋值给 str2 发送至发送区和对方机器，并且将 str1 清零
 - 发送区显示字符串以及字符串框，并且其位置依次下移
- 字符串接收 LCD 屏显示
 - 对方机器发送字符串并触发 UART 中断
 - 通过中断程序把该字符串发送至 LED 屏的接收区
 - 接收区显示字符串以及字符串框，并且其位置依次下移



图 10 效果图

六、 代码调试

1、 编辑窗口文件重叠

- 问题详述：当发送字符串给 LCD 屏幕编辑区时，之前发送的文字无法删除，不断的重叠，包括回删功能，无法删除文字，但光标位置正确
- 问题思考：光标位置正确说明代码正确，只能是屏幕显示问题，屏幕没有更新，之前放上去的字符串会一直存在
- 问题解决：把编辑区窗口部分图片单独列出一个矩阵，按下发送就重新显示编辑窗口，此时编辑窗口为无文字状态

2、 头像图片无法任意设置位置

- 问题详述：图片只能通过函数放在固定位置，不能改动
- 问题思考：改变函数内循环
- 问题解决：改变函数内循环

3、 Keypad 无反应

- 问题详述：按下键值无法显示到屏幕上，keychar 不能接受到值
- 问题思考：有可能是 keypad 本身损坏，也有可能是代码错误
- 问题解决：断点法，查看是否进入 color_lcd_test 函数，尝试更换 keypad

4、 图片出现 image2lcd

- 问题详述：图片出现 image2lcd 字样
- 问题思考：软件问题
- 问题解决：注册软件

5、 出现两张图片，并且位置出错

- 问题详述：图片出现两张图片
- 问题思考：图片像素点为 16 位，需用两个值来表示，图片为按行扫描，所以行取值时需除以 2
- 问题解决：在显示图片代码中，行取值时除以 2

6、 Uart 串口每次只能发一个数据

- 问题详述：串口每次只能发送一个字符，若发送多个，只能接收第一个字符，并且下一次无法发送
- 问题思考：每次只发送一个字符是因为中断接收一个字符后关闭，此时字符只有第一个被赋值并发送给 LCD；发送多个字符下一次无法发送没有找到原因。
- 问题解决：中断部分基础薄弱，需要加强学习

七、 实验总结

本次试验实现了两台 arm 机的通讯;实现了在 LCD 屏幕上显示图像文字;实现了在屏幕上编辑图像和文字;学会了使用中断函数控制程序,学会了对寄存器的使用和控制。

在这次实验中,初步掌握了 ARM 嵌入式 U 软件开发与调试技术。对嵌入式的寄存器,中断,通讯等有了一定的认识,学会了利用 keil4 开发 C 和 ARM 汇编代码。学会了在 keil4 中进行代码调试的方法和过程,对 keil4 的调试以及中断配置有了初步的了解。

在整个课程学习和课程设计中,我懂得了很多东西,夯实了嵌入式学习的基础,把上课学的知识融会贯通,十分感谢王老师一直以来仔细严谨的教学和耐心的为我们解答各种问题,使我收获良多,同时也意识到了很多不足的地方,如编程语言和嵌入式基础薄弱,以前学习的东西不实践就容易忘掉,明白了要投入去思考和实践才能更好的收获知识,以后我也会抱着这种严谨认真的态度对待我接下来的课业和项目。学习是一个长期积累的过程,在以后的工作和生活中都应该不断的学习,努力提高自己的知识和综合素质。



八、 附录

1、 Lcd_test.c

```
void color_lcd_test3(void)
{
    BitmapViewTft16Bit_8004801((UINT8T*)(g_ucBitmap6));
    Glib_Rectangle(235,90+25*(q+times),295,110+25*(q+times),BLACK);
    Lcd_DspAscII8x16(240,92+25*(q+times),BLUE,str);// 接收字体
    BitmapViewTft16Bit_8004803((UINT8T*)(g_ucBitmap7));
    Lcd_DspHz24(205,82,GREEN,"对方机");
}
/*****
*****
void color_lcd_test2(void)
{

    lcd_init_app();
    str1[num]=keychar;
    if(keychar!='1') //不是回车
    {
        if(keychar=='0')
        {
            str1[num-1]=' ';
            str1[num]=' ';
            num-=2;
            BitmapViewTft16Bit_640480((UINT8T*)(g_ucBitmap4));
            for(j=0;j<2;j++)
            {
                Glib_Rectangle(560+j,360+j,620-j,390-j,GREEN);
            }
            Lcd_DspHz24(565,365,GREEN,"发送");
        }
    }
}
```

```

}
else //回车
{
    BitmapViewTft16Bit_640480((UINT8T*)(g_ucBitmap4));
    for(j=0;j<2;j++)
    {
        Glib_Rectangle(560+j,360+j,620-j,390-j,GREEN);
    }
    Lcd_DspHz24(565,365,GREEN,"发送");
    for(j=0;j<num;j++)
    {
        str2[o]=str1[j];
        o=o+1;
    }
    str2[o]='\0';
    q=q+1;
    BitmapViewTft16Bit_8004802((UINT8T*)(g_ucBitmap5));

    Glib_Rectangle(545,90+25*(q+times),605,110+25*(q+times),BLACK);
    Lcd_DspAscII8x16(550,92+25*(q+times),BLUE,str2);// 发送字体
    uart_sendstring(str2);
    uart_printf("\n The words that you input are: %s\n",str2);
    o=0;
    //num=0;
    for(j=0;j<10;j++)
    {
        str1[j]=' ';
    }
    //memset(str1,0,sizeof(str1));
    num=-1;

}

Lcd_DspAscII8x16(200,310,RED,str1);// 未发送字体
num=num+1;

```

```

}
/*****
*****

void color_lcd_test1(void)
{
    int i=30;
    uart_printf("\n LCD display Test Example (please look at LCD screen)\n");
    lcd_init_app();
    BitmapViewTft16Bit_800480((UINT8T*)(g_ucBitmap2));
    BitmapViewTft16Bit_240320((UINT8T*)(g_ucBitmap3));
    BitmapViewTft16Bit_8004803((UINT8T*)(g_ucBitmap7));
    Lcd_DspHz24(205,82,GREEN,"对方机");
    Glib_Rectangle(199,79,641,401,BLACK);
    Lcd_DspHz24(50,i+=20,RED,"说明");
    Lcd_DspHz12(10,i+=30,GREEN,"中间部分为聊天界面");
    Lcd_DspHz12(10,i+=20,GREEN,"绿色直线上面右边部分为发");
    Lcd_DspHz12(10,i+=20,GREEN,"送区，左边部分为接收区");
    Lcd_DspHz12(10,i+=20,GREEN,"绿色直线下面部分为编辑区");
    Lcd_DspHz12(10,i+=20,GREEN,"按");
    Lcd_DspAscII8x16(26,i,GREEN,"0");
    Lcd_DspHz12(34,i,GREEN,"键为回删，");
    Lcd_DspAscII8x16(114,i,GREEN,"1");
    Lcd_DspHz12(122,i,GREEN,"键为发送");
    for(j=0;j<5;j++)
    {
        Glib_Line(200,300+1*j,640,300+1*j,GREEN);
    }
    for(j=0;j<2;j++)
    {
        Glib_Rectangle(560+j,360+j,620-j,390-j,GREEN);
    }
    Lcd_DspHz24(565,365,GREEN,"发送");
}

```

```

*****
*****/
void color_lcd_test(void)
{
    int i=290;
    lcd_init_app();
    BitmapViewTft16Bit_800480((UINT8T*)(g_ucBitmap1));
    Lcd_DspHz24(460,i+=30,BLACK,"卢思岑");
    Lcd_DspAscII8x16(460,i+=30,BLACK,"2150160409");
    Lcd_DspHz24(460,i+=30,BLUE,"机电工程学院");
}

```

2、 keypad.c

```

void keypad_test(void)
{

    if(keyscan())
    {
        switch(KeyNo)
        {
            case 0x0000: keychar = 'U'; break;           // FUN key
            case 0x0001: keychar = 'D'; break;
            case 0x0002: keychar = '-'; break;
            case 0x0003: keychar = '0'; break;
            case 0x0004: keychar = '+'; break;
            case 0x0100: keychar = '*'; break;
            case 0x0101: keychar = 'C'; break;
            case 0x0102: keychar = '3'; break;
            case 0x0103: keychar = '2'; break;
            case 0x0104: keychar = '1'; break;
            case 0x0200: keychar = 'F'; break;
            case 0x0201: keychar = 'B'; break;
            case 0x0202: keychar = '6'; break;
            case 0x0203: keychar = '5'; break;
            case 0x0204: keychar = '4'; break;
            case 0x0300: keychar = 'E'; break;

```

```

        case 0x0301: keychar = 'A'; break;
        case 0x0302: keychar = '9'; break;
        case 0x0303: keychar = '8'; break;
        case 0x0304: keychar = '7'; break;
        default: break;
    }
    if( keychar == 'U')
        uart_printf(" You have pressed key < FUN >\n");
    else
        uart_printf(" You have pressed key < %c >\n", keychar);
    color_lcd_test2();
}
}

```

3、uart.c

```

void __irq uart1(void)
{
    int j;
    rINTSUBMSK=0x7ff;
    if((rUTRSTAT1&0x01 >0)&&(rUERSTAT1==0)) //发送缓存区没有值
    并且接收没有中断正在被请求
    {
        rINTSUBMSK=0x7ff;
        str[dx]=rURXH1;
        dx=dx+1;
    }
    rSUBSRCPND=rSUBSRCPND;
    ClearPending(BIT_UART1);
    rINTSUBMSK&= ~(BIT_SUB_RXD1); //使能 UART1 的 RXD 中断
    BitmapViewTft16Bit_8004803((UINT8T*)(g_ucBitmap7));
    Lcd_DspHz24(205,82,GREEN,"对方机");
    Lcd_DspHz12(280,85,GREEN,"正在输入...");

    uart_printf(str);
    times=times+1;
    color_lcd_test3());

```

```

dx=0;
}

```

```

void uartinit(void)
{
    rSRCPND = rSRCPND;           // Clear all interrupt
    rINTPND = rINTPND;         // Clear all interrupt
    // pISR_EINT8_23=(UINT32T)int_int; //将中断函数入口地址送入
中断向量表
    pISR_UART1  =(UINT32T)uart1;
    rEINTPEND = 0xfffff;        //清除所有外部中断挂起
    rEXTINT1 &= ~((0x7<<4));
    rEXTINT1 |= ((0x2<<4));     //设置触发方式为下降沿触发
    rINTMSK &= ~(~(BIT_UART1)); //中断屏蔽控制 开放
uart1 中断
    rINTSUBMSK = 0x7ff;        //屏蔽所有子中断
    rINTSUBMSK&= ~(BIT_SUB_RXD1); //使能 UART1 的 RXD 中断

```

4、glib.c

```

void BitmapViewTft16Bit_8004801(UINT8T *pBuffer)
{
    UINT32T i, j;
    UINT32T *pView = (UINT32T*)frameBuffer16BitTft800480;
    pView+=(90+25*(q+times))*800;
    for (i = 90+25*(q+times); i < 20+90+25*(q+times); i++)
    {
        for (j =100; j < 10+100 ; j++)
        {
            pView[j] = ((*pBuffer+1) << 24) + ((*pBuffer) << 16) +
            ((*pBuffer+3)) << 8) + (*pBuffer+2));

            pBuffer += 4;
        }
    }
}

```

```

        pView+=LCD_XSIZE_TFT_800480;
    }

}

void BitmapViewTft16Bit_8004802(UINT8T *pBuffer)
{

    UINT32T i, j;
    UINT32T *pView = (UINT32T*)frameBuffer16BitTft800480;
    pView+=(90+25*(q+times))*800;
    for (i =90+25*(q+times); i < 20+90+25*(q+times); i++)
    {
        for (j = 305; j < 10+305 ; j++)
        {

            pView[j] = ((*pBuffer+1) << 24) + ((*pBuffer) << 16) +
            ((*pBuffer+3) << 8) + (*pBuffer+2));

            pBuffer += 4;

        }
        pView+=LCD_XSIZE_TFT_800480;
    }

}

void BitmapViewTft16Bit_8004803(UINT8T *pBuffer)
{
    UINT32T i, j;
    UINT32T *pView = (UINT32T*)frameBuffer16BitTft800480;
    pView+=80*800;
    for (i =80; i < 110; i++)
    {
        for (j = 100; j <220+100 ; j++)
        {

```

```

        pView[j] = ((*pBuffer+1) << 24) + ((*pBuffer) << 16) +
        ((*pBuffer+3) << 8) + (*pBuffer+2));

```

```

        pBuffer += 4;

```

```

    }

```

```

    pView+=LCD_XSIZE_TFT_800480;

```

```

}

```

```

}

```

```

void BitmapViewTft16Bit_240320(UINT8T *pBuffer)

```

```

{

```

```

    UINT32T i, j;

```

```

    UINT32T *pView =(UINT32T*)frameBuffer16BitTft800480;

```

```

    pView+=64000;

```

```

    for (i = 80; i <320+80; i++)

```

```

    {

```

```

        for (j = 100; j <220+100 ; j++)

```

```

        {

```

```

            pView[j] = ((*pBuffer+1) << 24) + ((*pBuffer) << 16) +
            ((*pBuffer+3) << 8) + (*pBuffer+2));

```

```

            pBuffer += 4;

```

```

        }

```

```

        pView+=LCD_XSIZE_TFT_800480;

```

```

    }

```

```

}

```

```

void BitmapViewTft16Bit_640480(UINT8T *pBuffer)

```

```

{

```

```

    UINT32T i, j;

```

```

    UINT32T *pView = (UINT32T*)frameBuffer16BitTft800480;

```

```

pView+=244000;
for (i =305; i <305+95; i++)
{
    for (j = 100; j < 220+100 ; j++)
    {

        pView[j] = ((*pBuffer+1) << 24) + ((*pBuffer) << 16) +
        ((*pBuffer+3) << 8) + (*pBuffer+2));

        pBuffer += 4;

    }
    pView+=LCD_XSIZE_TFT_800480;
}
}

```