电子台签调试文档

Debug Doc

Powered By Coogle

华中科技大学 2008 级种子班

陈曦骏 李海涛 龚小聪

目录

图目录

表目录

1 简介

电子台签设计到硬件部分和软件部分，软件部分又分为 PC 端上位机和 MCU 端下位机。相应的调试也分为硬件和软件部分。上位机与单片机通过串口通信，可以用串口调试助手模拟调试，不需要与硬件部分直接交互。下位机的软件和硬件调试是相辅相成的，在证明软件方案正确性的同时，也说明硬件部分没有大的问题。我们小组的调试时间大多用在软硬件联调上。

2 小组印刷版电路概览

小组印刷版电路为两层。概览如下：

2.1 电路板总体情况

表格 2-1 电路板总体情况

项目	参数
印制电路板层数	2 层板
板卡数目	1（显示+主控）
板卡尺寸	10975mil*4290mil

2.2 正面 PCB

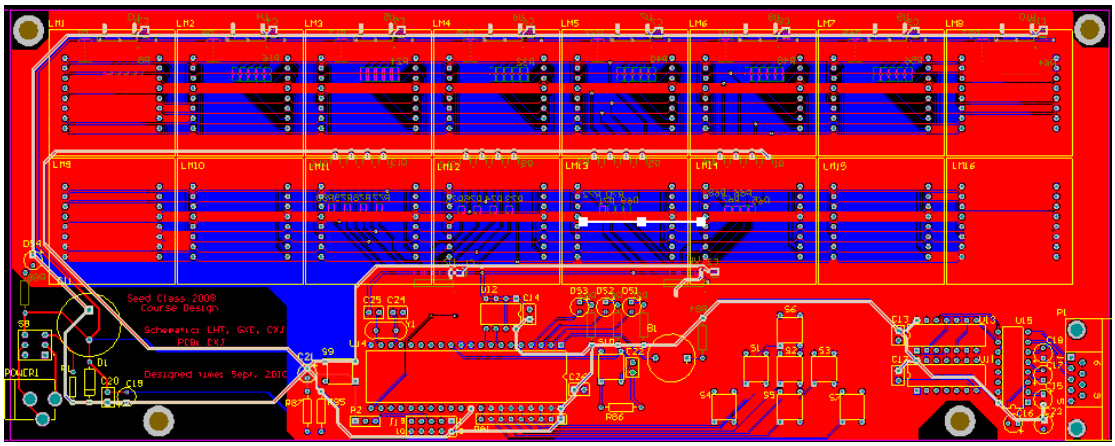

图 2-1 正面 PCB

2.3 反面 PCB

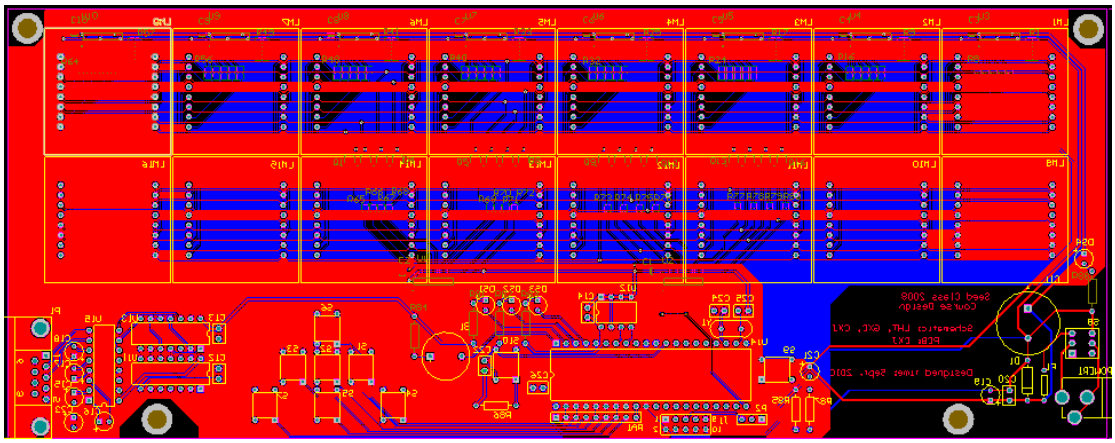

图 2-2 反面 PCB

3 硬件焊接

3.1 电子台签所需器材

表格 3-1 器材列表

Footprint	Comment	Description	Quantity
SOT-23B	8550	PNP 贴片三极管	16
DIP-8	AT24C04	EEProm	1
ATMEL89S52-DIP40	Atmel89S52		1

Buzzer	BEEP	蜂鸣器	1
VP32-3.2	Cap	Capacitor	8
CAP-LED	CAP POL	1uF 瓷片电容	7
CAPPR7.5-16x35	Cap Pol1	220uF 点解电容	1
C1206	Cap Semi	1uF 贴片电容	10
DIODE-0.4	Diode 1N4001	二极管	1
AXIAL-0.3	FUSE1	800mA 保险丝	1
HDR1X3	Header 3	3 脚跳线	1
HDR2X5	Header 5X2	5X2 接头	1
LEDMatrix8	JM-M1088A-B	LED 点阵	16
CAP-LED	LED	普通 LED	4
DIP-16	MAX232CPE		1
Power	POWER	电源插孔	1
0805	RES	200 欧 8050 贴片电阻	64
0805	RES	1K 欧 8050 贴片电阻	16
AXIAL-0.4	RES	Resistor	8
HDR1X9	Res Arry	4.7K x 8	1
DSUB1.385-2H9	RS232	串口母头	1
N014	SN74AC04N	逻辑非门	1
N014	SN74HC125N	3 态门	1
D016_N	SN74HC138D	3-8 译码器	2
D016_N	SN74HC595D	8 位 3 态移位寄存器	8
SWITCH-6	SW-DPDT	电源开关	1
key	SW-PB	按键，8 长 1 短	9
XTAL	XTAL	12M 晶振	1

由于电子台签的电路板比较大，为了节约空间，我们小组在设计的时候，尽可能的考虑多用贴片代替直插的原件。小组的成员也没有焊接贴片元件的经验，这样一来，为以后的焊接埋下了一定的隐患。

3.2 焊接顺序

- 由于各个元件的大小高低不尽相同，焊接的顺序也不一样。
- 我们采取的策略是先难后易，先低后高，先贴片后直插。
- 宗旨：焊接方便，节省时间。
- 先焊接难度大的，这主要是指管脚密集的贴片式集成芯片。如果把这些难度大的放于最后焊接，一旦焊接失败把焊盘搞坏，那就会前功尽弃。

- 先低后高，先贴片后直插。这样焊接起来方便，如果先把高的元件焊接了，有可能妨碍其他元件的焊接，尤其是高大的元件密集众多的时候。如果先焊接插装的元件，电路板就会在焊台上放不平，影响焊接心情和效率。

3.3 焊接注意事项

3.3.1 电阻、电容等两引脚的贴片元件

- 批量将同侧的一端焊盘镀上适量焊锡
- 依据元件明细表批量将元件正确焊接在镀锡焊盘上
- 批量焊接元件另一端
- 修复优化焊点，并做清理工作

3.3.2 晶体管、集成电路等多引脚贴片元件

- 批量将元件的其中一脚焊盘镀上焊锡
- 依据元件明细表，按由小到大，由低到高，方便焊接的原则批量将元件的一个脚或两个脚固定
- 在电路板上，要求固定可靠。
- 批量焊接元件剩余引脚，引脚间距允许时可以依次单个焊接引脚，引脚间距较小的集成电路可
- 采用堆锡法焊接：在元件所有引脚上镀锡，暂不考虑引脚间的桥接
- 元件焊接完成后，去掉元件引脚上多余和连接的焊锡。
- 修复优化引脚焊点，并做清理工作

3.3.3 直插元件

- 直插元件比较容易焊接，主要要做到美观准确

3.3.4 焊接后续工作

- 依据元件清单核对元器件，确保所有元件焊接位置正确
- 确认二极管、蜂鸣器、钽电容等有极性要求的元器件焊接正确
- 确认集成电路引脚排列与电路板上一致
- 优化修复焊点，确认所有元件焊点光滑无毛刺、无漏焊、无虚焊

4 硬件调试

4.1 最小系统测试

在所有的硬件焊接工作完成后，我们的电子台签看起来如下：

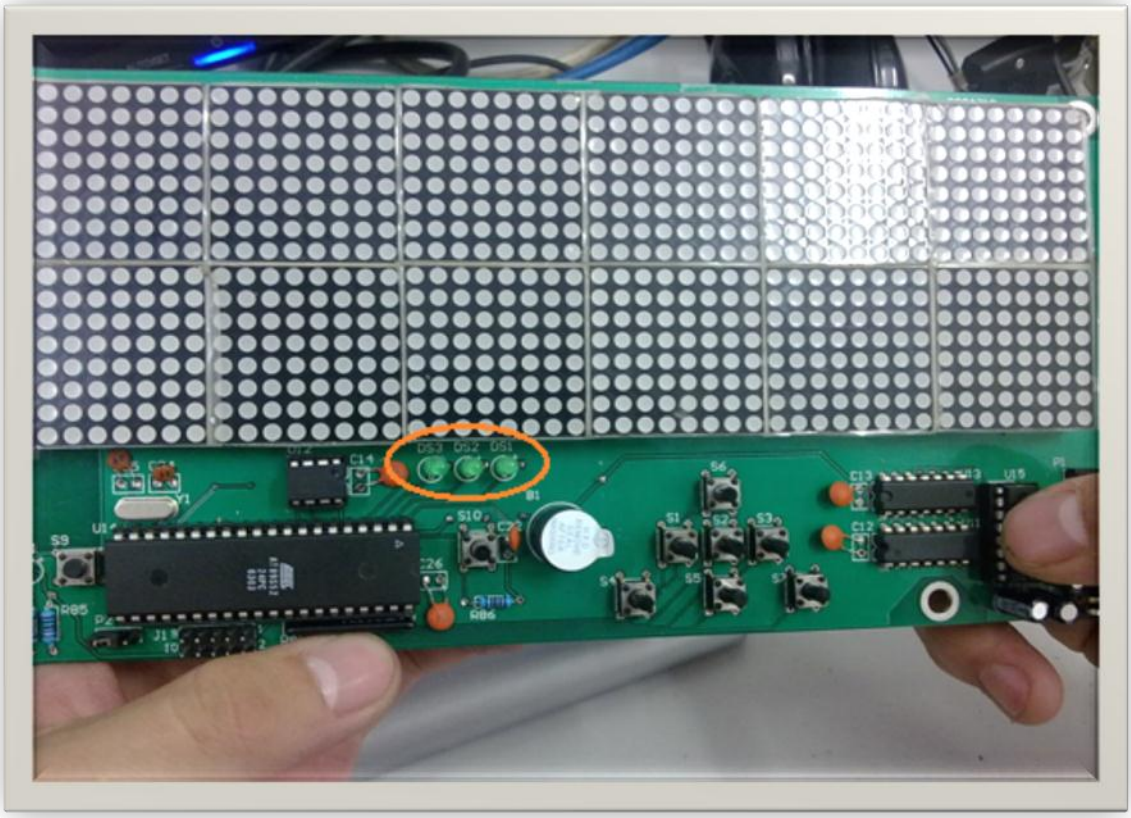

图 4-1 焊接后的台签

遵循最小系统原则，我们先编写了跑马灯程序，目的是使上图中被圈中的 3 个 LED 依次点亮。以此来验证两个问题：是否能成功将.hex 文件下载到单片机中，硬件的最小部分是否能够正常工作。

实验结果：经测试，LED 灯能够正常工作。这个结果也让我们看到了整个大系统成功运行的希望。

4.2 LED 点阵测试

Coogle 编写了测试 LED 点阵的程序，目的是以行为单位测试 16 行 LED 的状态。预期的现象是 16 行 LED 从上到下一次点亮，但是结果却是下面 8 行一次点亮，上面的没有反应。但是下面的 8 行点亮过后，要等待一定的时间，即本应该是上面 8 行点亮的时间。同时，上面的 8 行虽然没有依次点亮，但是第 3 行完全不亮，且倒数第 2 列对应的 LED 比周围的要亮。

针对上述问题，我们小组成员进行了细致的分析，我们用示波器测了上面 8 行点阵对应的三极管，发现这八片三极管全都被烧毁了。追究原因，发现是在焊接的时候，先焊了这八片三极管，然后直接用万用表的测三极管两段电压，导致全部烧毁。找了几片三极管，重新焊接之后，发现一切正常。

教训:测试电容的时候一定不能直接把电压表两极加在三极管的引脚上，这样会导致电流过大，三极管容易烧毁。

在测试 LED 点阵的过程中，我们发现在相同的程序控制条件下，有的行不亮，经过万用表的测试，发现是行选对应的电阻虚焊，重新焊接之后，问题解决。

教训：电路焊接完毕真的并不代表一切就绪，哪怕看起来没有问题了。

4.3 串口测试

在我们的设计中，串口是复用的，我们先测试了串口作为传输 LED 点阵显示数据的部分功能。我们为此编写了程序，测试程序的目标是通过串口传送数据，控制 LED 点阵的列选。在确定测试程序无误后，下载程序，但是 LED 点阵没有反应，经过一步步排查，用万用表测试 SN74AC04N 被使用的两端电压，竟然发现本来应该是一高一低的电压全部都是高电压!! 初步确定 SN74AC04N 挂掉了，然后拆卸，换上一块新的经过测试的 SN74AC04N，重新焊接，正常！ LED 点阵

按照一定规律有一半的列亮，另一半不亮。至此，串口测试通过。

教训：新买的元件也不一定是好的，在调试的过程中要考虑各种因素。

4.4 EEProm 测试程序的烧写情况

两个汉字+IICRead	成功
半个，一个，两个汉字+IICWrite	失败

经检查后发现，是因为递归函数的问题。在单片机里面，貌似不能使用递归函数，更改算法后成功。

5 下位机软件测试

5.1 串口测试

下位机的软件和硬件测试是分不开的，在此不一一列举个软件测试的详细情况。在我们的设计中，最终要实现的是通过串口复用来传输显示数据和下载点阵数据，传输 LED 点阵数据在上面已经叙述过，测试通过。而在利用串口模式 1 从上位机下载点阵数据的时候，我们遇到了麻烦，经过小组成员的反复调试、对比，得出了以下结论：

（前提条件:开串口中断，使用模式 1，关闭其他如 LED 点阵的显示以减少干扰）

表格 5-1 电压情况

对比项	电子台签(V)	51 开发板(V)
1(C1+)	5~10 矩形波	5~10 矩形波
2(V+)	10	10
3(C1-)	0~5 矩形波(261.9kHz)	0~5 矩形波(142.3kHz)
4(C2+)	0~10 矩形波(262kHz)	0~10 矩形波(142.3kHz)
5(C2-)	-10~0 矩形波(261.5kHz)	-10~0 矩形波(142.2kHz)
6(V-)	-10	-10

7(T2out)	~10，不稳定	-10
8(R2in)	0	0
9(R2out)	0	5
10(T2in)	0	5
11(T1in)	0	5
12(R1out)	5	5
13(R1in)	0	0
14(T1out)	~10,不稳定	-10
15(GND)	0	0
16(VCC)	5	5

经查证(实测),频率的区别是 Max232 型号的区别导致,并且实测后,发现该指标不会对传输结果造成直接影响，而电压的区别是板子电路造成的.

5.1.1 直接在程序中添加点阵数据

在串口模式 1 调试成功之前，我们将点阵数据写在了程序的代码中，直接显示，经过点阵映射，终于成功了：

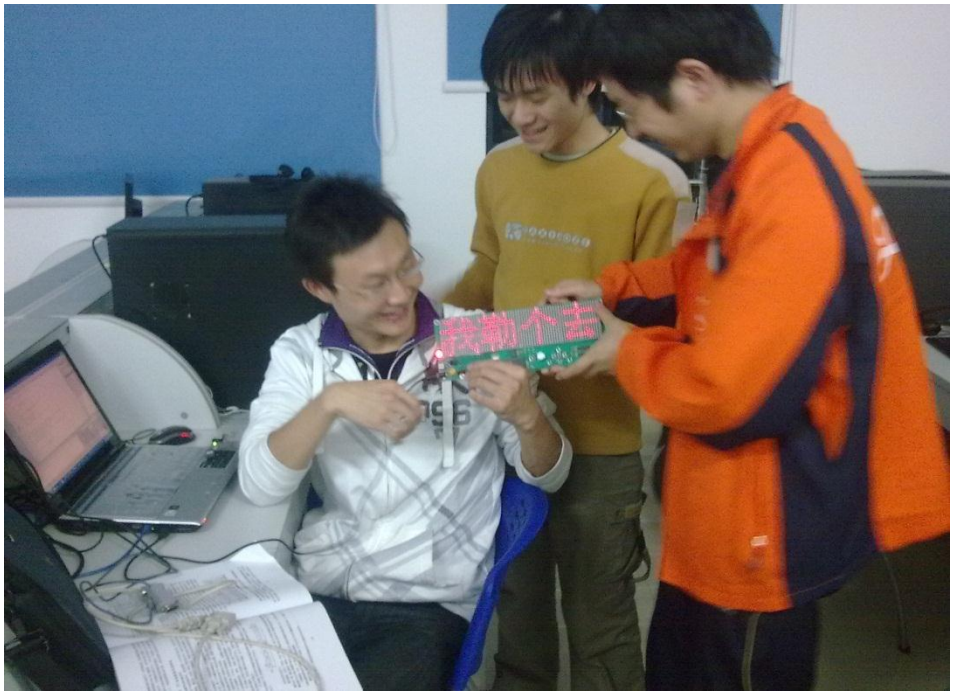

图 5-1 凌晨 2：45 分，A711 房间

6 上位机软件测试：跨平台引起的各个问题

6.1 说明

我们的上位机是使用 QT 编写的，拥有两套工程，一套是使用 VS2008 生成的解决方案，只适用于 Windows 系统，而另一套是基于 QT 的 pro 文件工程，可以跨平台编译。使用了一个串口库，按其官方文档说明，可以在 Windows，Linux，MacOS 上使用，但只在 Windows 上验证过。

因为设计之初就是朝着跨平台努力的，所以使用 QT，但是并不是使用了 QT 就能跨平台的，跨平台是一个非常复杂的问题。

6.2 工程的跨平台问题

在 Windows 下全部编译通过后，移植到 Linux 下（感谢石葆光的帮助），发现出现了文件未找到的错误，经检查，发现是 lib 文件的问题。

lib 文件是编译过程中生成的中间文件，在 Windows 下的格式是“xxx.lib”，而我在 pro 文件中指定 lib 文件时使用的如下方式：

```
LIBS    += dian_matrix_libd.lib
```

后来发现，这样的写法是非常不具有跨平台特性的。可以说是硬编码，因为 Linux 平台下的 lib 文件是 xxx.o 的形式，这样看来，直接指定名称及后缀是非常不可取的。

避免方法：使用如下形式：

```
LIBS    += -ldian_matrix_libd
```

QT 是个跨平台框架，自然会提供全套的跨平台服务，-l 前缀是 QT 工程文件中表征 lib 文件的标记，加上这个前缀，QT 工具会自动寻找对应 lib 文件，完成跨平台的任务。

针对这个问题，我想，在使用一个工具时，应该了解它的设计宗旨，如果某个功能在它的宗旨之内，而这个工具又是一个优秀工具的话，它一定会以某种形式提供这个功能。

6.3 编码的跨平台问题

在 Linux 系统下编译完成后，发现运行结果异常，表现为汉字显示的乱码，大致如下图所示：

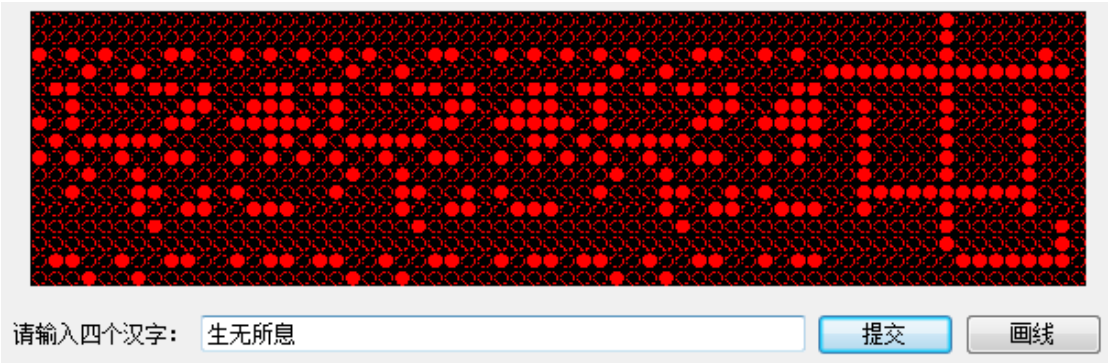

图 6-1 linux 下上位机乱码

之前在 Windows 平台下其实也遇到过类似问题，通过 VS 调试后发现是编码错误，因为 HZK16 文件是基于 ascii 码编制的字库文件，使用如下方式获取文件的偏移量：

```
// get the word's zone/bit code
unsigned char zoneCode= word[0] - 0xa0;
unsigned char bitCode = word[1] - 0xa0;
long offset = (94*(zoneCode-1) + (bitCode-1))*32;
```

然后使用随机读取的方式获取文字点阵信息。

这种方法严重依赖字符的编码方式，只能使用 ascii 编码才能正确访问，wstring 使用 unicode 编码，ubuntu 下使用 utf-8 编码，造成了显示乱码的现象。

解决方案：

使用 QT 库提供的 QTextCodec 类完成各类编码的转换工作，使其统一转换成“GB18030”编码方式，就可以正常的访问汉字字模信息了。

图 6-2 编码转换后乱码消失

注：此部分两张图片为 Windows 下的截图，只为更清晰的说明，不是证明。

6.4 其他跨平台问题

6.4.1 文件路径问题

在 Linux 下运行编译结果时，发现找不到 HZK16 文件，但是明明已经放在目录下了。经分析，最终确定是文件路径的分隔符“\”和“/”的问题。

使用 QDir::toNativeSeparators 函数可以解决此问题。

6.4.2 串口端口问题

在 Windows 下使用的是 COM1~COMn 的形式，而 Linux 下使用的 ttySn 的形式，如何编写跨平台的程序？

目前为止，我还没有找到一个非常好的方式，只是使用了条件编译的方式来限定不同的字符，如下：

```
    // add serial device name to the combo box.
    QString commonName=
#ifdef _WINDOWS
        "COM";
#else
        "/dev/ttyS";
#endif
```

```
    for (int i= 0; i < 4; i++)
    {
        QString fullName= commonName +
#ifdef _WINDOWS
            QString::number(i+1);
#else
            QString::number(i);
#endif
        serialDevice->addItem(fullName);
}
```

7 依然存在的问题

7.1 关于 PCB

- PCB 的空间利用不充分，很多控制部分，如单片机，逻辑控制器等元件完全可以考虑设计成贴片放到 LED 点阵后面，将按键，串口等外接部分单独设计成板，或使用通用版，减小板子面积。
- 三级管封装有问题，引脚错误，导致焊接很纠结很麻烦，以后一定要确认每一个元件的封装。
- 排阵没有考虑到牛角座的空间，而且放在那个缝里非常不美观，空间很大，应该放出来。
- 很多 IC 的引脚都是悬空的，使得电路不是很稳定，以后这种情况都要接地或接电源。
- 很多器件，如电容，串口母头等较高的器件可以考虑放在后面，到时候壳子准备好了，LED，按键之类的元件一扣就可以装好了。