Usart GPU 使用手册

文档更新日期	更新内容
2014-10-30	根据咨询修改一些描述和添加新产品
2014-10-16	增加有关程序框架方面的描述(11 节)
2014-9-10	C 编程 sprintf 问题
2014-8-8	版本程序 1.0，升级了自定义波特率部分
------	原始版本

第一部分：基础应用

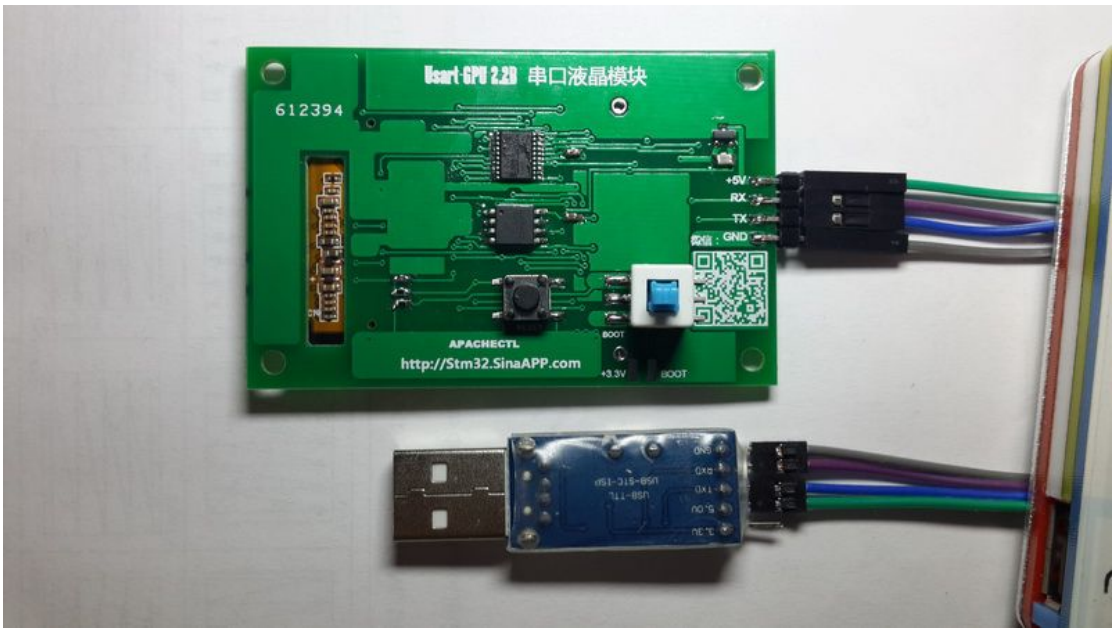
概述：

Usart 是串口的意思，GPU 是图形处理器的意思，产品的含义是做一个单片机使用的专用图形处理器，或者称之为串口液晶显示模块。

产品列表：

型号	类型	尺寸	分辨率	模块	价格	备注
GPU22B	非触摸	2.2 吋	220X176		33 元	购买
GPU22A	非触摸	2.2 吋	320X240		42 元	购买
GPU22C	带 4 个按钮	2.2 吋	320X240		45 元	购买
GPU24A	非触摸	2.4 吋	320X240		研发中	
GPU26A	非触摸	2.6 吋	400X240		研发中	
GPU28A	非触摸	2.8 吋	400X240		60 元	购买
GPU28B(TP)	带触摸	2.8 吋	400X240		72 元	购买
GPU30B(TP)	带触摸	3.0 吋	400X240		研发中	
GPU32A	非触摸	3.2 吋	400X240		研发中	
GPU35A	非触摸,核心板	3.5 吋	480X272	4M 存储器	68 元	购买
GPU35B	非触摸	3.5 吋	480X320		研发中	

一、 接线



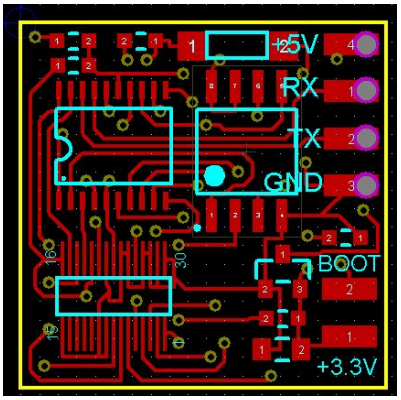
开箱后，可以将串口输出的 4 根引脚焊上排插，使用杜邦线将串口接到 USB 转 TTL 线上，即可接到电脑 USB 口上上电，屏幕即会显示第一屏的 Hello 界面；

说明下：照片中是我调试用的，因此增加了 RESET 按钮和运行程序刷机切换的 BOOT 自锁开关，正常使用和产品中不带这两个东东；

主板中使用 XC6206 接到 5V 的，6206 是一个低压差稳压器，输出 3.3V，160mV 的低压差，让板子在 3.46V 即可正常供电，实际使用中，电压低到 3V，6206 也可以正常输出电压但是不稳压；由于 STM32 最低 2V 即可工作，因此本板子可以直接接单节锂电池即可工作；

如果接不通，建议 RX TX 反一下，有些下载线是指接入单片机端的标志，不是自身标示；

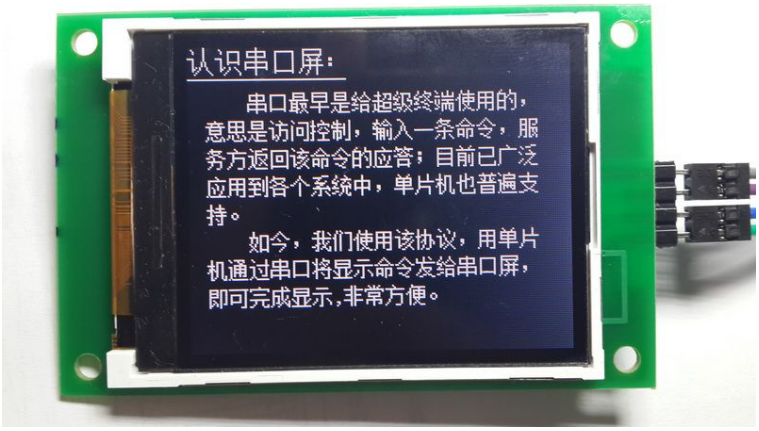
其他产品都有 这 4 根线，3.5 吋的核心板太小，导致字体印刷不清，请参考：

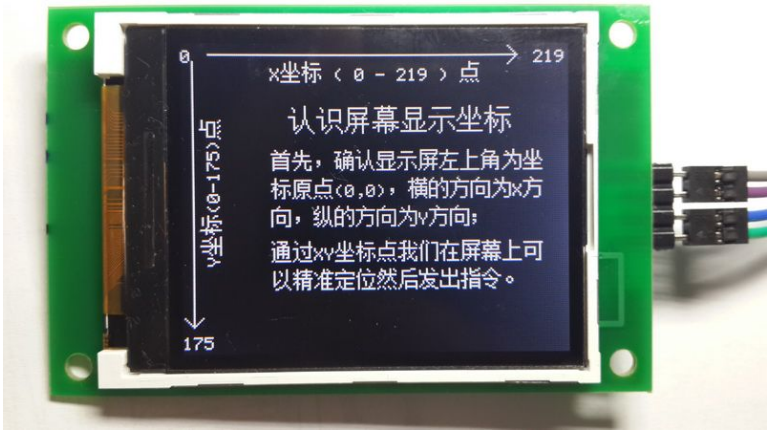


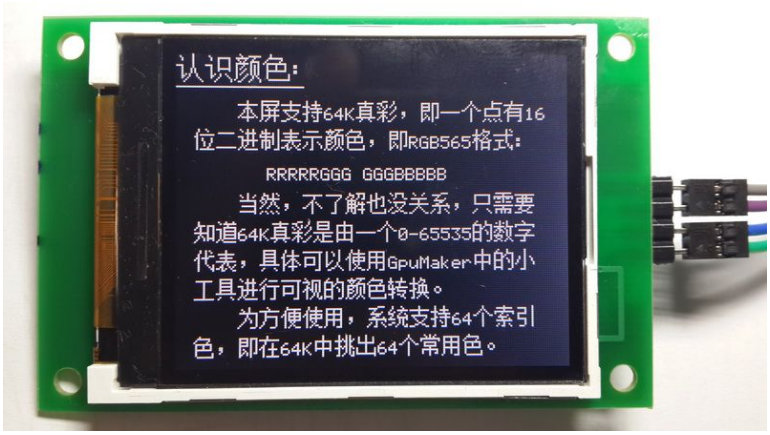
二、 上电，观看演示

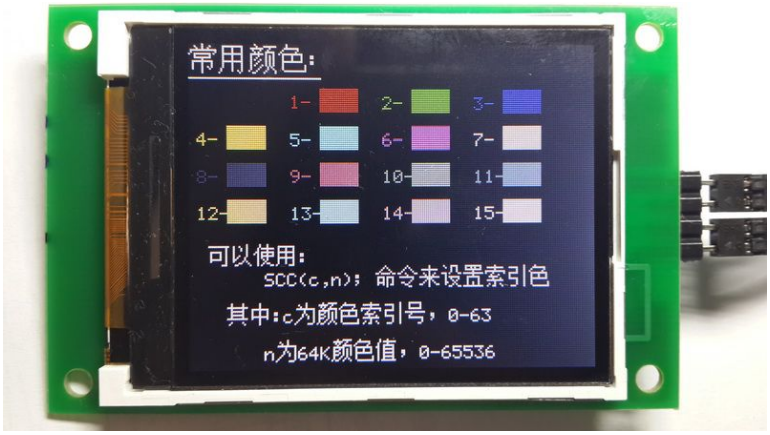
这是上电后的 Hello 界面，俗称欢迎界面，此界面属于第一个批界面，可以有上位机程序在 PC 下自由设计，用户可以在这个界面上设计自己产品的名字和公司的图标；

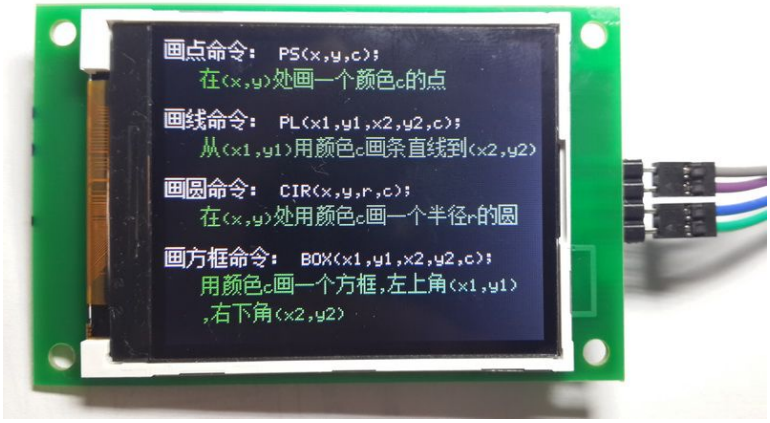
开机界面十秒种，如果收不到串口命令，就会进入演示状态。正常的量产之后，单片机需要在上电十秒内给串口液晶屏发送指令，只要一发送指令，就自动的进入串口命令状态。



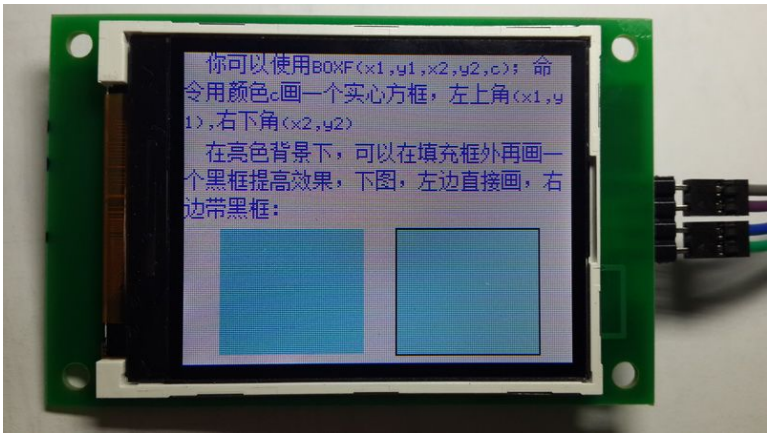

0
219
X坐标（0 - 219）点
Y坐标（0-175）点
175
认识屏幕显示坐标
首先，确认显示屏左上角为坐标原点(0,0)，横的方向为x方向，纵的方向为y方向；
通过xy坐标点我们在屏幕上可以精准定位然后发出指令。

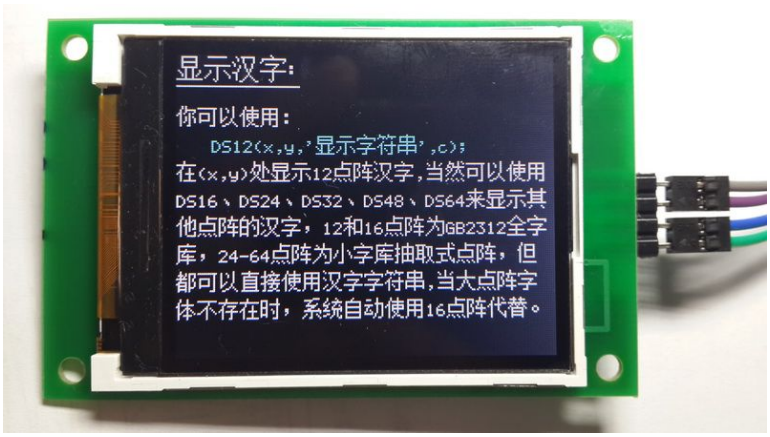

认识颜色：
本屏支持64K真彩，即一个点有16位二进制表示颜色，即RGB565格式：
RRRRRGGG GGGBBBBB
当然，不了解也没关系，只需要知道64K真彩是由一个0-65535的数字代表，具体可以使用GpuMaker中的小工具进行可视的颜色转换。
为方便使用，系统支持64个索引色，即在64K中挑出64个常用色。

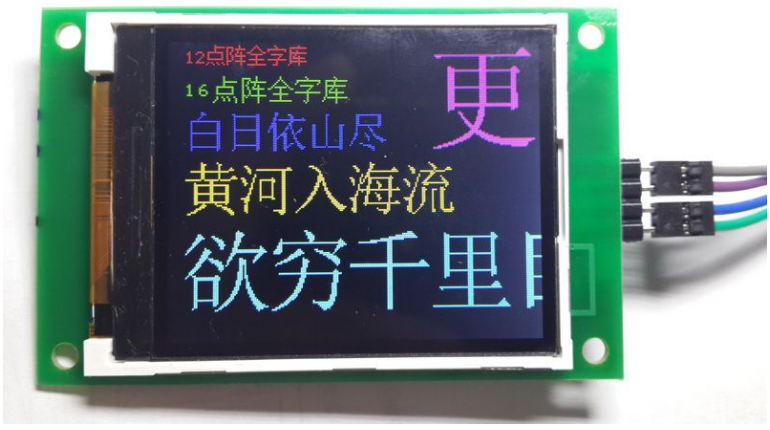

常用颜色：
1-
2-
3-
4-
5-
6-
7-
8-
9-
10-
11-
12-
13-
14-
15-
可以使用：
SCC(c,n)；命令来设置索引色
其中：c为颜色索引号，0-63
n为64K颜色值，0-65536

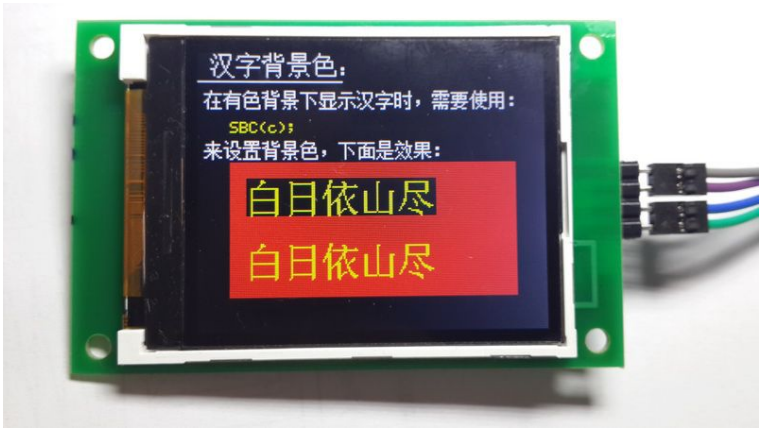

画点命令： PS(x,y,c)；
在(x,y)处画一个颜色c的点
画线命令： PL(x1,y1,x2,y2,c)；
从(x1,y1)用颜色c画条直线到(x2,y2)
画圆命令： CIR(x,y,r,c)；
在(x,y)处用颜色c画一个半径r的圆
画方框命令： BOX(x1,y1,x2,y2,c)；
用颜色c画一个方框，左上角(x1,y1)
，右下角(x2,y2)

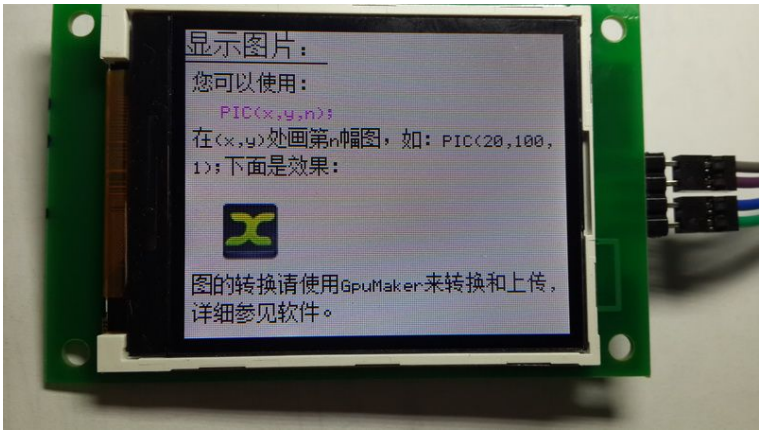

上面的图形执行以下命令：
PS(10,10,1);PL(20,20,200,60,2)
CIR(50,50,50,3);BOX(30,30,180,80,4);

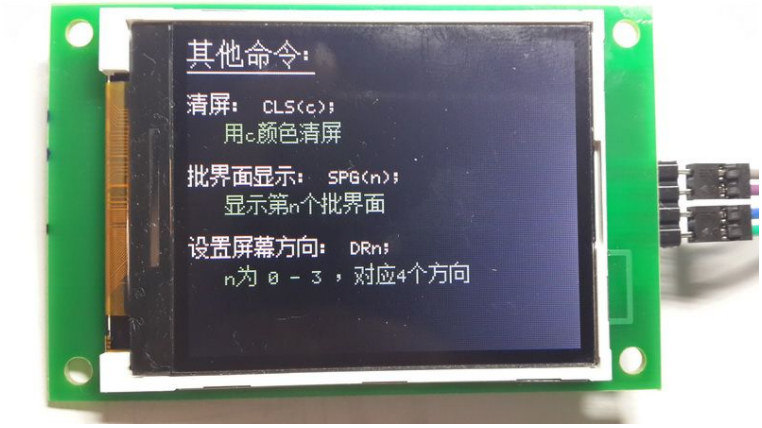

你可以使用BOXF(x1,y1,x2,y2,c)；命令用颜色c画一个实心方框，左上角(x1,y1),右下角(x2,y2)
在亮色背景下，可以在填充框外再画一个黑框提高效果，下图，左边直接画，右边带黑框：

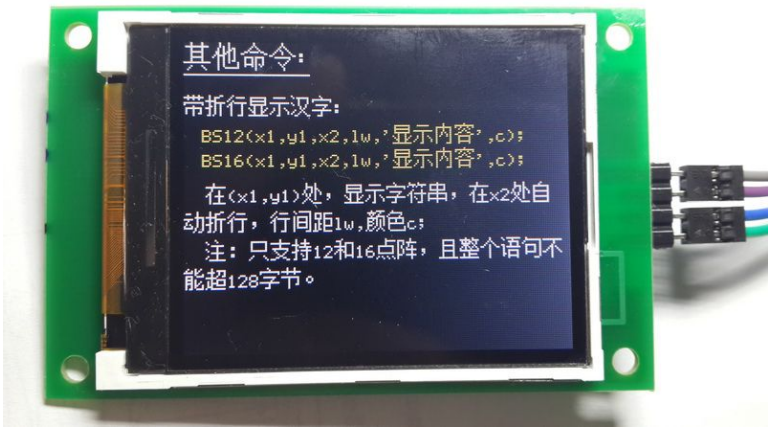

显示汉字：
你可以使用：
DS12(x,y,'显示字符串',c);
在(x,y)处显示12点阵汉字，当然可以使用DS16、DS24、DS32、DS48、DS64来显示其他点阵的汉字，12和16点阵为GB2312全字库，24-64点阵为小字库抽取式点阵，但都可以直接使用汉字字符串，当大点阵字体不存在时，系统自动使用16点阵代替。


12点阵全字库
16点阵全字库
白日依山尽
更
黄河入海流
欲穷千里


汉字背景色：
在有色背景下显示汉字时，需要使用：
SBC(c);
来设置背景色，下面是效果：
白日依山尽
白日依山尽


显示图片：
您可以使用：
PIC(x,y,n);
在(x,y)处画第n幅图，如：PIC(20,100,1);下面是效果：
图的转换请使用GpuMaker来转换和上传，详细参见软件。


其他命令：
清屏： CLS(c);
用c颜色清屏
批界面显示： SPG(n);
显示第n个批界面
设置屏幕方向： DRn;
n为 0 - 3 ，对应4个方向


其他命令：
带折行显示汉字：
BS12(x1,y1,x2,lw,'显示内容',c);
BS16(x1,y1,x2,lw,'显示内容',c);
在(x1,y1)处，显示字符串，在x2处自动折行，行间距lw,颜色c;
注：只支持12和16点阵，且整个语句不能超128字节。

三、 接 GpuMaker

将 USB 转 TTL 接入计算机，注意由于程序的原因串口号不能超过 10，即 COM1~COM9 方可使用；

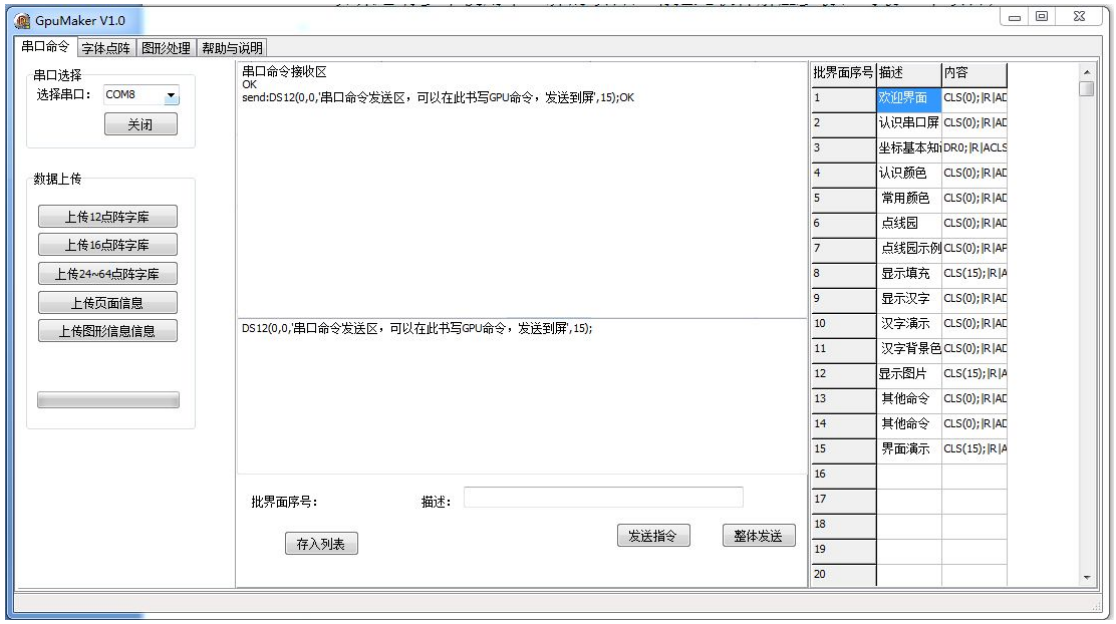
从：

http://stm32.sinaapp.com/dn/d.php?n=g7

下载 GpuMaker 程序，程序是一个 ZIP 包，绿色软件，解压到硬盘中即可使用；

如果您有多个使用串口屏的项目，请把此软件解压多份，每份一个项目；

运行解开目录中的 GpuMaker.exe，系统进入：



选择，左上角的串口号，点击“打开按钮”，串口连接成功；此时点击“发送指令”，液晶屏即可显示表示连接正常；

【备注：】如果没有反应，请自查

1、TTL 串口线是否接好；拔下 TTL 线，COM 口消失，插上，就出现，就表示 TTL 对应的那个 COM 口正常

2、注意不要选错串口号，和波特率，默认波特率（拿到新品未设置时），波特率都是 115200

3、COM 口不能超过 10

4、如果命令接收区不出现 OK，请把 TX 和 RX 两根线颠倒一下接；GPU 上电会从串口传出序列号数据，如果接收区有，表示正常。

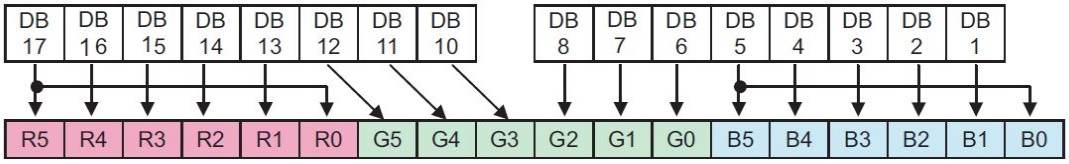
5、如果还不行，考虑换一个 USB 口，或换台机器或者换一根 USB 转 TTL 的转换线。

四、 命令表

命令	说明	示例
CLS(c);	用 c 颜色清屏	CLS(0);
SCC(c,n);	自定义 c 颜色，颜色值 n 由上位机提供计算	SCC(15,65535);
SBC(c);	设置背景色 C，显示汉字等时无点阵时填的颜色	SBC(1);
PS(x,y,c);	在(x,y)的地方画一个颜色 c 的点	PS(100,100,1);
PL(x1,y1,x2,y2,c);	从(x1,y1)用颜色 c 画一条直线到(x2,y2)	PL(0,0,100,0,2);
BOX(x1,y1,x2,y2,c);	用颜色 c 画一个方框，左上角(x1,y1),右下角(x2,y2)	BOX(0,0,100,100,2);
BOXF(x1,y1,x2,y2,c);	用颜色 c 画一个实心方框，左上角(x1,y1),右下角(x2,y2)	BOXF(0,0,100,100,2);
PIC(x,y,n);	在(x,y)处画第 n 幅图	PIC(0,0,1);
CIR(x,y,r,c);	在(x,y)处用颜色 c 画一个半径 r 的园	CIR(100,100,50,1);
SPG(n);	显示第 n 个批界面	SPG(1);
DS12(x,y,'显示内容字符串',c);	在(x,y)处用颜色 c 显示一行 12 点阵字	DS12(0,0,'显示字符串',1);
DS16(x,y,'显示内容字符串',c);	在(x,y)处用颜色 c 显示一行 16 点阵字	DS16(0,0,'显示字符串',1);
DS24(x,y,'显示内容字符串',c);	在(x,y)处用颜色 c 显示一行 24 点阵字	DS24(0,0,'显示字符串',1);
DS32(x,y,'显示内容字符串',c);	在(x,y)处用颜色 c 显示一行 32 点阵字	DS32(0,0,'显示字符串',1);
DS48(x,y,'显示内容字符串',c);	在(x,y)处用颜色 c 显示一行 48 点阵字	DS48(0,0,'显示字符串',1);
DS64(x,y,'显示内容字符串',c);	在(x,y)处用颜色 c 显示一行 64 点阵字	DS64(0,0,'显示字符串',1);
DRn	设置屏幕显示的方向；n 为 0~3，分别对应屏的 4 个方向，可以使用此调整横竖屏显示；另外此命令不清屏，因此可以显示在横屏下显示部分竖显汉字。	DR0; 横屏显示 DR1; 竖屏显示 DR2; 横屏倒立 DR3; 竖屏倒立
BS12(x1,y1,x2,lw,'显示内容',c);	在(x1,y1)处，显示 12 点阵字符串，在 x2 处自动折行，行间距 lw,颜色 c;	BS12(0,0,219,4,'显示内容...很多字',c);
BS16(x1,y1,x2,lw,'显示内容	在(x1,y1)处，显示 16 点阵字	BS16(0,0,219,4,'显示内容...很

',c);	符串，在 x2 处自动折行，行间距 lw,颜色 c;	多字',c);
INF;	传回系统序列号等信息	INF;

五、 定义颜色



颜色是由 RGB 构成的，系统支持的 64K 色其实是 65536 中颜色，使用 16 位二进制（2 字节）组成，16 位，分成：R 红色 5 位；G 绿色 6 位；B 蓝色 5 位，就是俗称 RGB565 模式；常规的计算机颜色描述是由 3 字节组成，每字节一色，比如红色描述为：0xFF0000;绿色描述为 0x00FF00;而蓝色描述为 0x0000FF

看不懂也没关系，只需要进入 GpuMaker，到“帮助与说明”找到：

点击“颜色”就可以出现：

选择一个颜色：点击确定，系统就会显示颜色的 16 进制值；点击“转换”

选中的颜色就可被计算出 GPU 的颜色值，如上例是 13507；
您就可以使用 SCC(1,13507);命令将 1 号颜色设置成刚才选择的颜色；

六、 截取汉字点阵

【重要概念 :】

在进行本节的讲解之前，先理解一下 24~64 点阵使用的是小字库，所谓的大字库（全字库）就是把用到的 7000 多个汉字所有的点阵都存储下来，这样什么字都可以直接使用，这样很方便，但是占用的空间也就很大，有些应用只用到几个字，也得讲 7000 多字全部存储下来，这样在大点阵字体下尤其不划算（很多大点阵字往往是在标题中显示几个字），因此就有了大点阵字使用的“小字库”之说，就是用到哪几个字，系统抽取那几个字的点阵存到 GPU 中，不用的不存，在本产品中，每种大小的字体（含全角和半角）最多能存 512 个字（字符）；

当您看到 GPU 在大点阵字显示的时候发现个别字很小（16 点阵）就表示该字没有录入到“字体点阵”设置中的汉字区中，也就是没有生成小字库，或生成后没有上传的 GPU 中。

系统支持 12、16 点阵的全 GB2312 的字库，含符号区；因此 12、16 点阵无需使用软件截取可以直接使用，但是 24、32、48、64 点阵需要使用 GpuMaker 进行点阵的截取转换；

启动 GpuMaker 进入：“字体点阵”

可以看到 4 种字体分全角汉字和半角字符共 8 类，我们以 32 点阵汉字为例讲述使用方法：

第一步：点击点阵序号 3 后面的栏目，进入 32 点阵汉字的编辑状态：

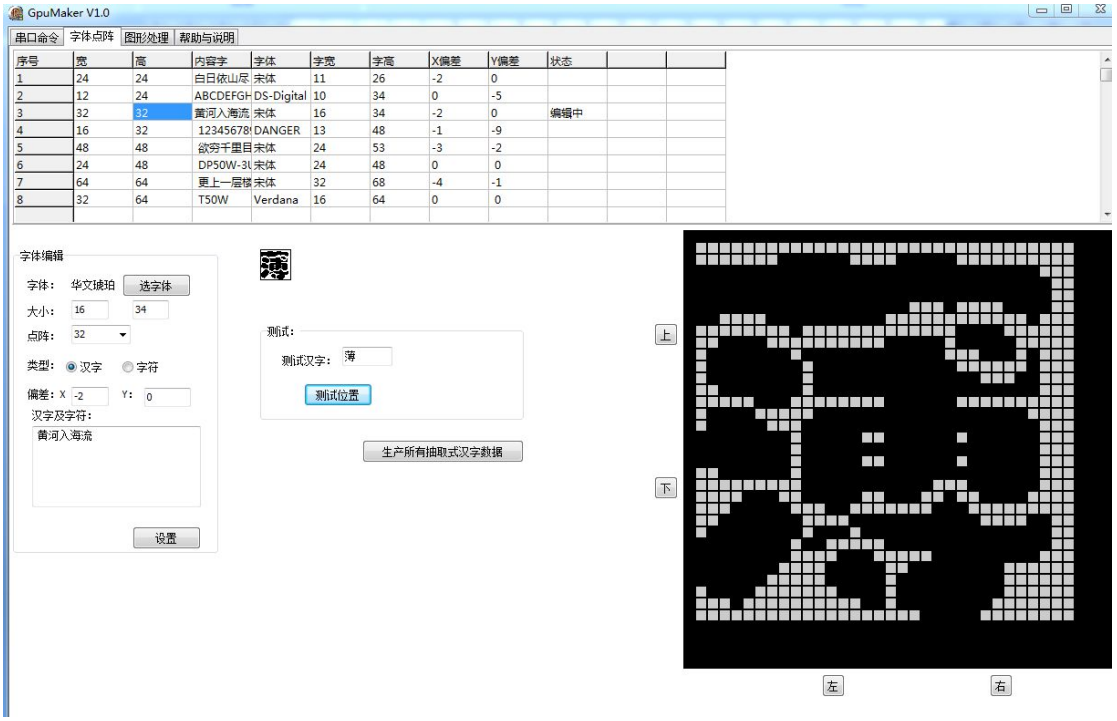
此时，字体编辑区可选；32 点阵那一行后面的状态变为“编辑中”；

【注意】

这 8 组数据分别对应 4 种大小字体的全角和半角，不要把这些数据调乱，只需要调后面的文字即可

第二步：选择字模的属性。

点击“字体”按钮选择相应的字体，点击“测试位置”，右边的显示区就可以显示汉字的点阵：



在测试汉字选择上，我们一般选择“薄、餐”等复杂汉字，以免出界；

如果汉字较小，可以增大图中“大小：”后面 34 的值，汉字就会放大，反之缩小；

如果汉字偏向一边，可以使用 上下左右 按钮进行调整，使汉字尽可能大的填满方框；

将需要显示的汉字放到“汉字及字符”输入框中，无需查重，系统会自动查重，可以直接输入需要显示的每句话；

调整完成之后；点击“设置”按钮，将设置信息存好；

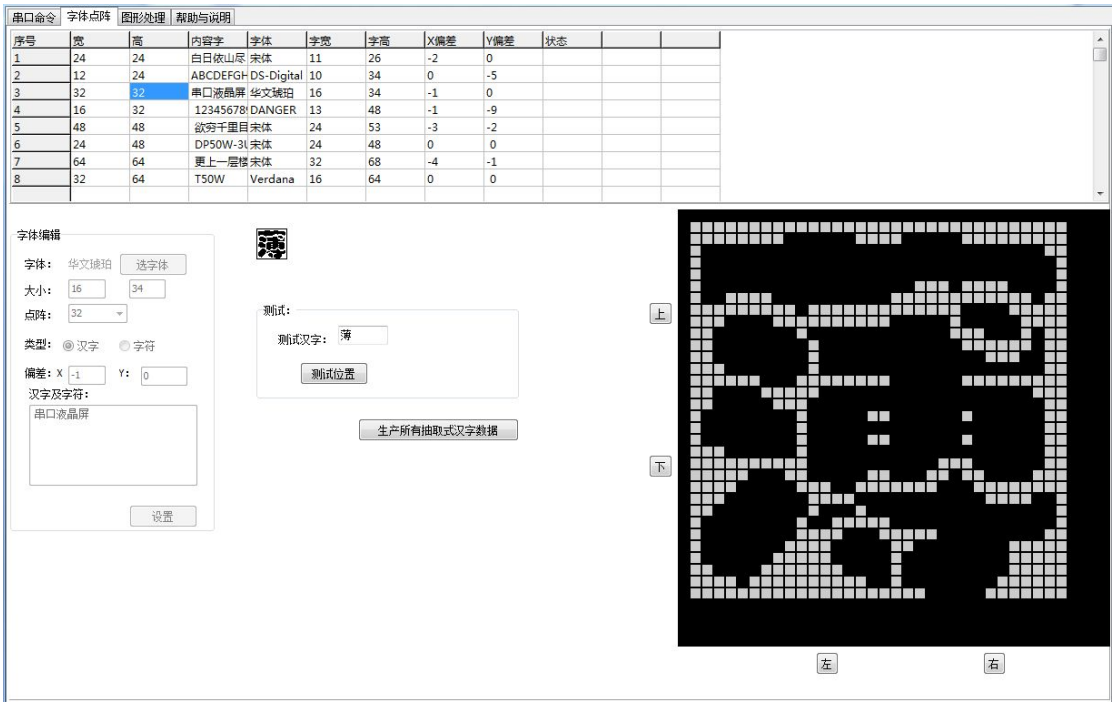
依次设置完别的字体；当设置半角字体的时候，需要注意：

1、半角可以通过“大小：”标签后的 2 个参数设置横宽比，这点不同于汉字，汉字只认后一个参数；测试的字符一般使用 W 等超宽字符；

2、选择半角字体的时候，如果要显示全部字母，建议找等宽字体，否则很难调好比例；

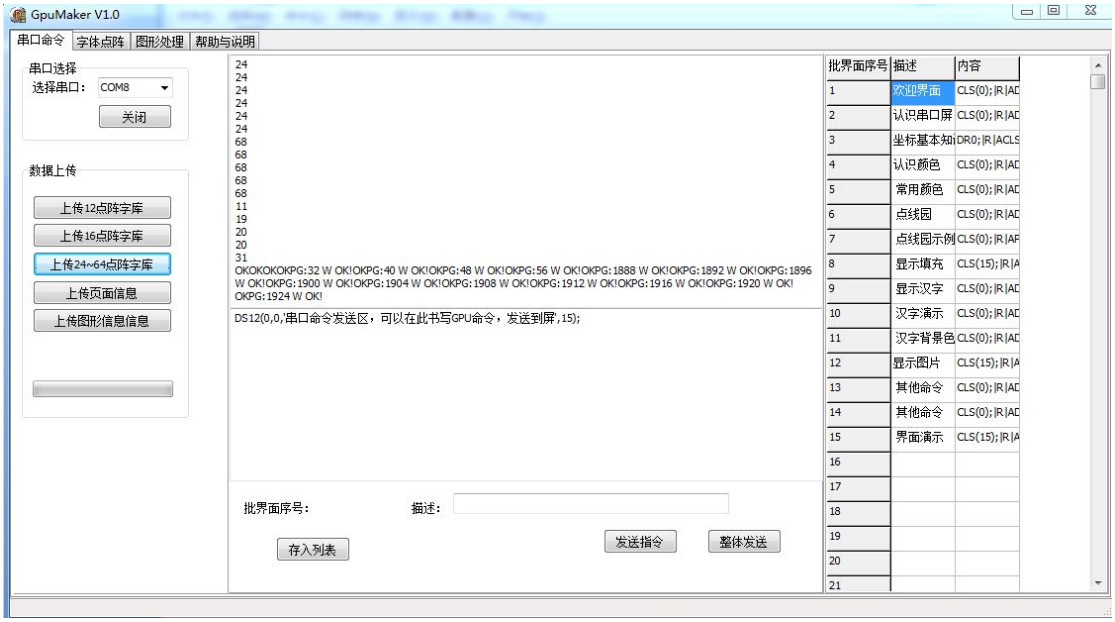
所需的字体可以百度下字体资源网站得到；

第三步：生产点阵；



点击“生产所有抽取式汉字数据”，系统就会将所有汉字循环一边，生成数据；

第四步：上传数据到 GPU 串口液晶屏

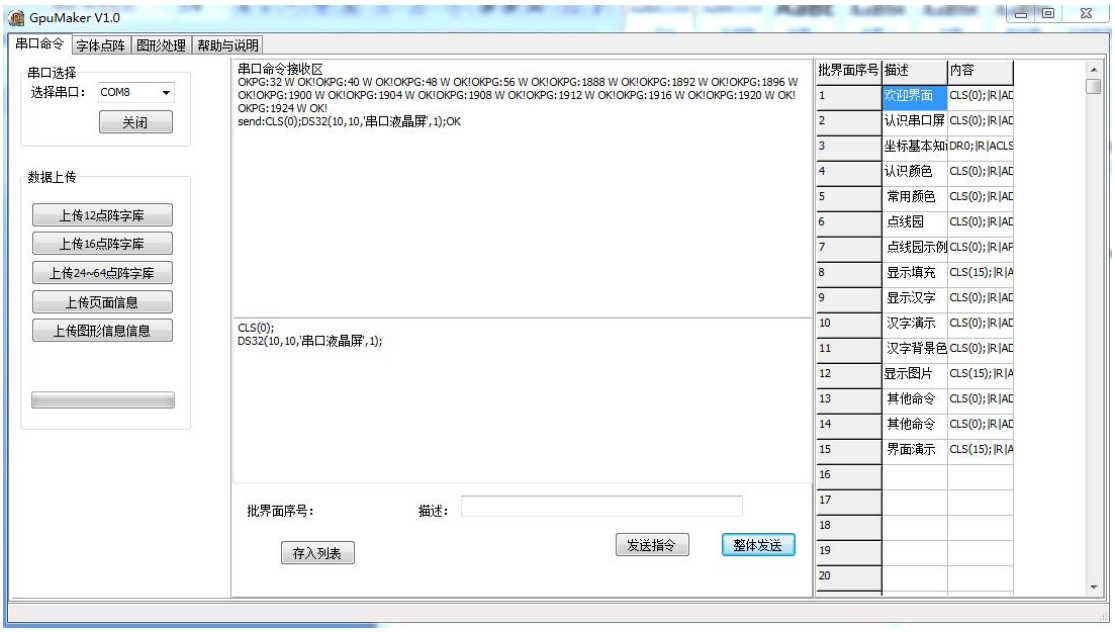


回到串口命令界面，使用 USB-TTL 串口线接上 GPU 模块，然后点击“连接”，再点击“上传 24~64 点阵字库”；如图，系统将字库上传到 GPU 模块，完成界面如上图；

然后发送命令：

```
CLS(0);
DS32(10,10,'串口液晶屏',1);
```

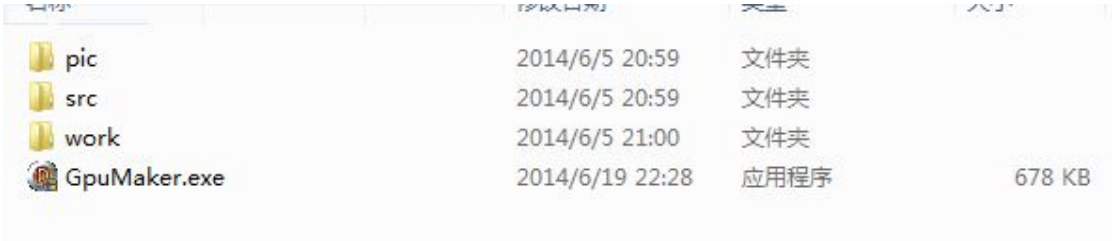
发送完成之后界面：



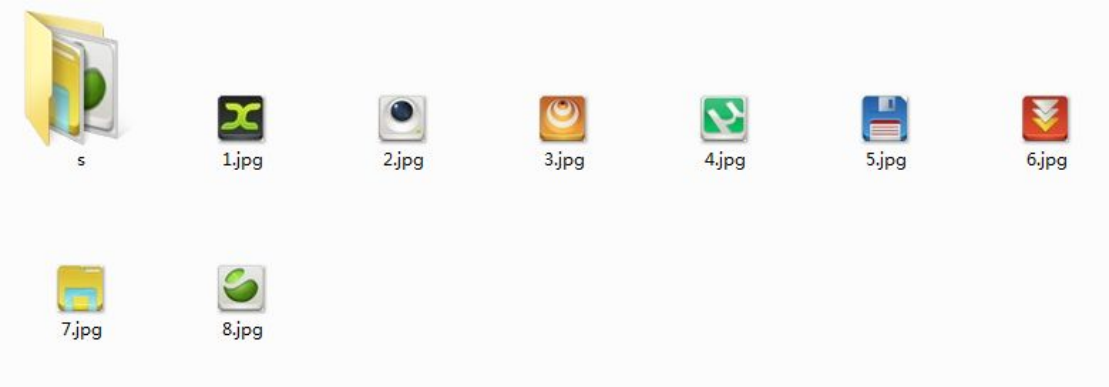
此时液晶屏显示：

七、 自定义图形

GpuMaker 目录如下：

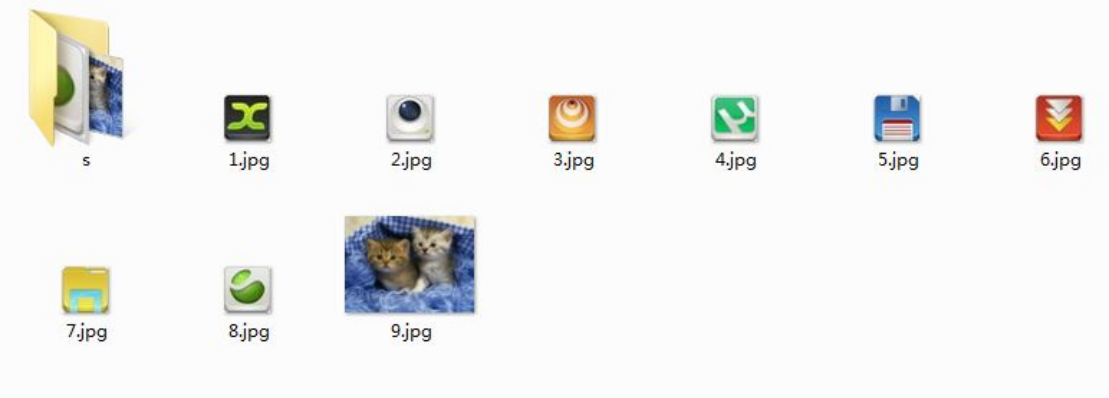


进入 pic 目录，选择缩略图显示：



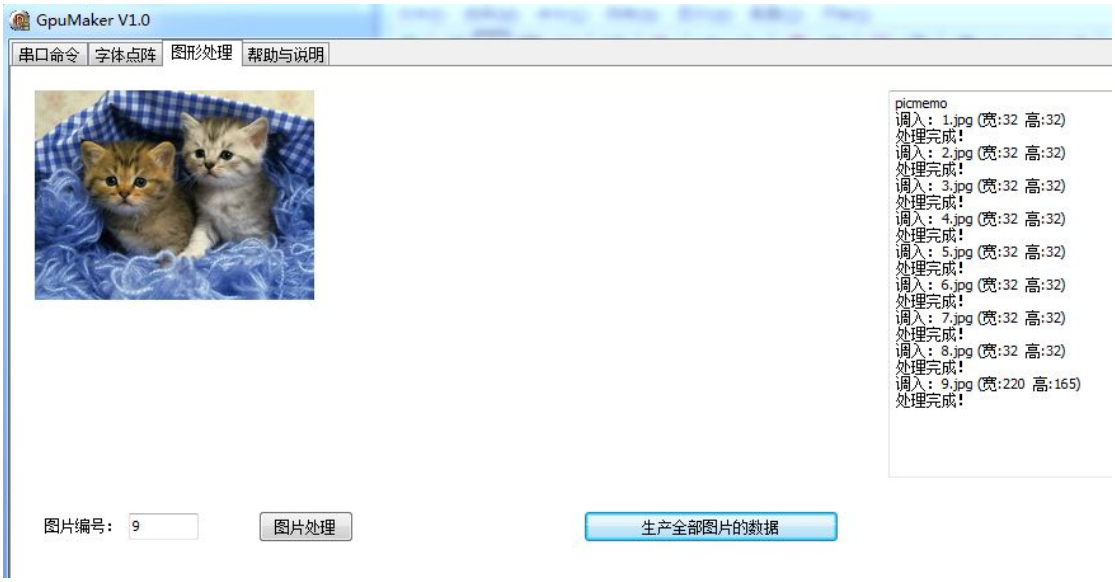
这样很容易看出图形对应的序号；

添一张图进去：



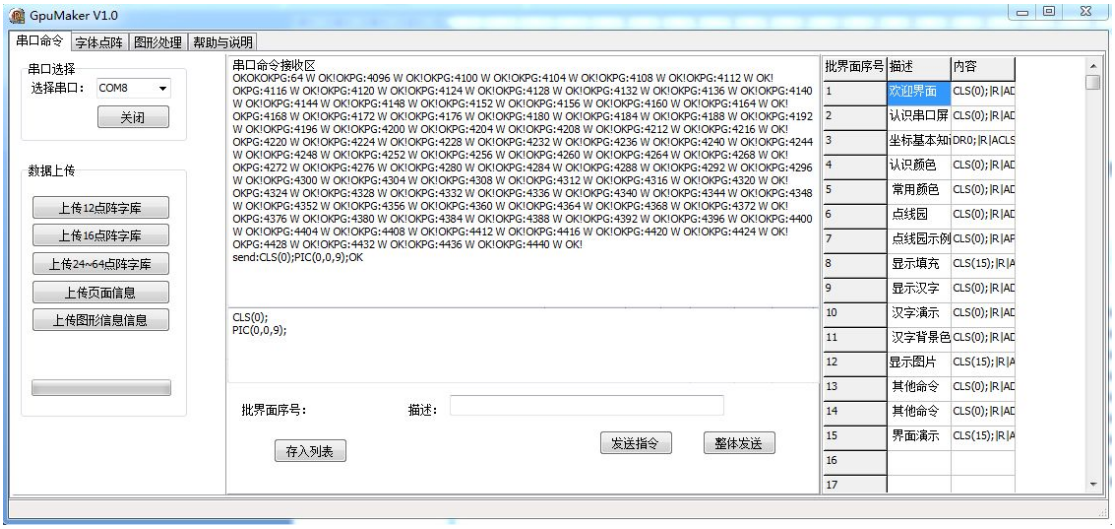
使用图形处理软件，将图缩到 220*176 点以下，如果是别的格式的图，可以用图形处理软件转换下；

启动 GpuMaker，进入：图形处理页面：



点击“生产全部图片的数据”按钮，系统自动将 pic 目录下的所有图片处理好；

进入“串口命令”界面，连上串口，点击“上传图形信息”按钮，系统将图片信息传入 GPU：

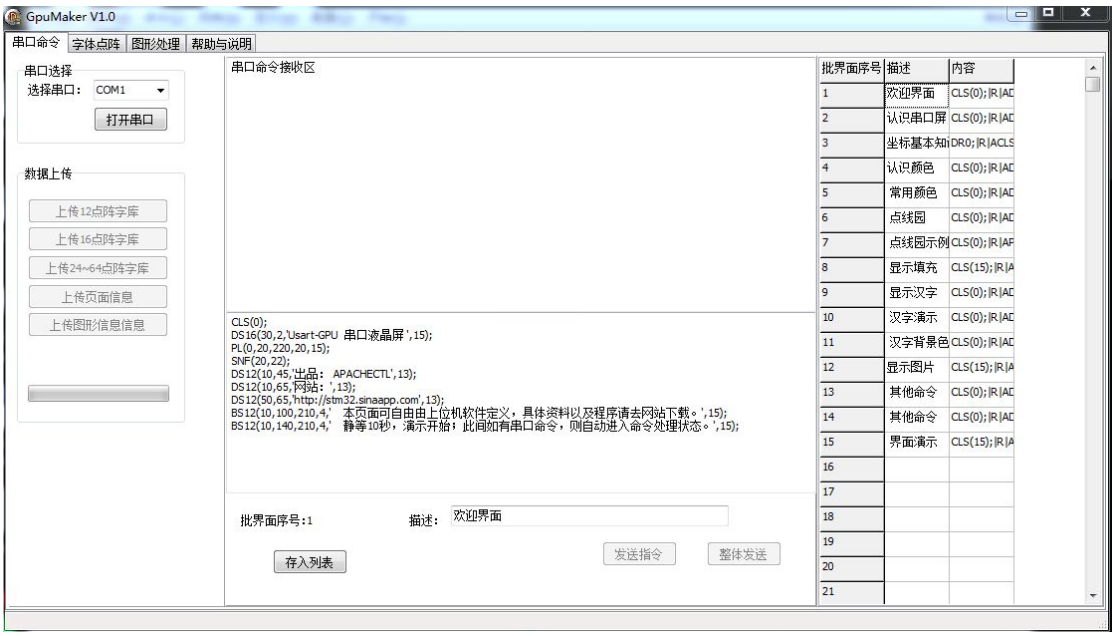


输入命令：
CLS(0);
PIC(0,0,9);

液晶屏显示：

八、 使用批界面

我们使用的单片机，大多数都不能提供充足的内存，因此对于复杂一些的 UI 有时就得考虑内存方面的问题，因此，能不能像 DOS 下批处理一样，将一群复杂的 UI 界面语句组合起来，这就是批界面；



在这个界面，右边就是批界面，系统允许有 127 个批界面，点击右边序号后面的格子，就可以修改描述以及批界面的语句，点击存入列表，就可以将批界面语句存入数据库；点击“上传页面信息”就可以将批界面语句传入 GPU 中；

使用的时候，可以使用串口传入命令：SPG（批界面序号）；即可显示该界面，无需将复杂的 UI 语句放置在单片机内存，再用串口传到 GPU；

另外：序号为 1 的批界面我们称之为 HELLO 界面，即 GPU 上电后立即显示的第一界面，因此此界面需要设计为产品的名称，公司的 LOGO 之类的，上电后，第一界面会显示 10 秒中，在这 10 秒钟内，主系统的单片机需要向 GPU 传送第一条指令，否则 GPU 就会进入演示模式，挨个将批界面依次显示，直到接收到串口指令；

技巧：

1、任何一个界面都分为背景元素和前台元素，背景元素是从界面创立起就一成不变的，因此非常适合放在批界面中；显示界面的时候，先使用批界面显示背景界面，再由串口指令刷新前台数据显示；

2、批界面中可以不加 CLS 清屏指令，这样可以使用多个批界面组合成一个更复杂的 UI 界面，这种情况下，某几个批界面可以成为某个背景界面的前台元素；比如：锂电池充电界面，左侧是充电数据，右侧是充电进度数据，我们可以使用另外一个批界面将右侧换成电池的图形，显示电池容量；

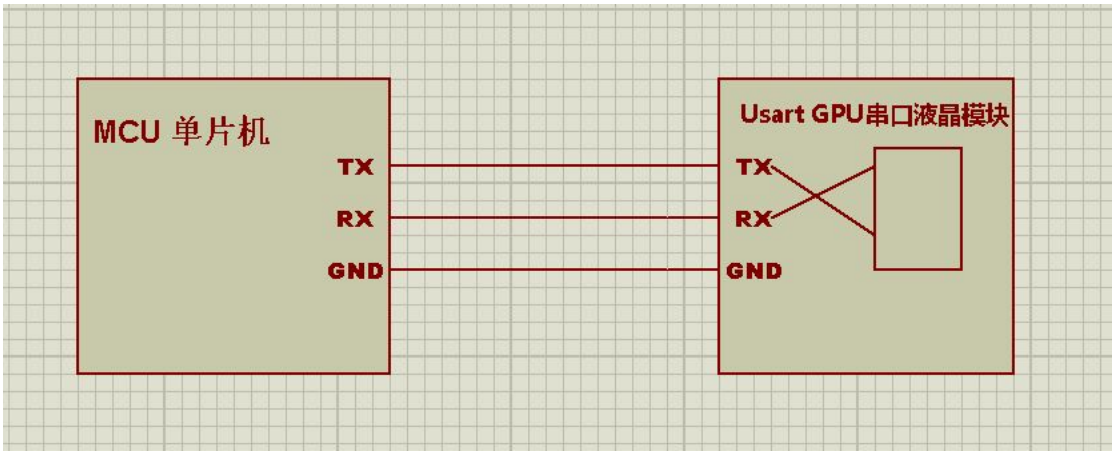
3、第一个批页面含有初始参数的设置，比如波特率的调整，详见串口波特率调整一节；

九、 界面示例

以上界面使用：

```
CLS(15);
BOX(0,0,219,175,15);
BOX(1,1,218,174,0);
BOXF(2,2,217,17,3);
PL(2,18,218,18,0);
SBC(3);
DS12(60,4,'菜单演示界面',15);
SBC(15);
PIC(20,40,1);DS12(25,75,'电压',0);
PIC(70,40,2);DS12(75,75,'电流',0);
PIC(120,40,3);DS12(125,75,'充电',0);
PIC(170,40,4);DS12(175,75,'输入',0);
PIC(20,110,5);DS12(25,145,'输出',0);
PIC(70,110,6);DS12(75,145,'测试',0);
PIC(120,110,7);DS12(125,145,'关闭',0);
PIC(170,110,8);DS12(175,145,'设置',0);
```

十、 接单片机



接法非常简单，如图接好即可；

TTL 电平是 0~5V 的电平，因此 TTL 串口不存在 5V 和 3.3V 单片机电平转换的问题，可以直接接入使用；但不可直接与 RS232 的串口接入，因为 RS232 的串口电平标准是 12V 以上，直接接入会烧掉 GPU 上的单片机；

在程序驱动来说，所有的单片机程序对于发送串口指令无在乎就三个要点：

1、初始化串口，由于目前 GPU 只支持统一的 115200 的串口波特率，因此初始化得初始化成此波特率，其余的参数均为默认；不熟悉的话，可以按照串口助手默认参数定；

2、将一个 BYTE 发送到串口发送端；

3、判断发送标示等待发送结束，结束后继续发送下一个字节；

因此，对于 STM32 来说可以使用下面的语句：

```
GpuSend("CLS(13);BOX(0,0,219,175,15);BOX(1,1,218,174,0);BOXF(2,2,217,17,3);PL(2,18,218,18,0
);\r\n");

void GpuSend(char * buf1)
{   u8 i=0;
    while (1)
    {   if (buf1[i]!=0)
        {   USART_SendData(USART1, buf1[i]);   //发送一个 byte 到串口
            while(USART_GetFlagStatus(USART1, USART_FLAG_TXE) == RESET){}; //等待发送结束
            i++;
        }
        else return;
    }
}
```

其他单片机请参考单片机手册自行书写相应语句；一般来说单片机对串口编

程的例子是很多的，可以 baidu 下或直接参考开发板例程；

【C 语言参考】：不少用户都是从汇编语言转到 C 语言的，对于 C 语言的字符串处理完全没有概念，有不少用户都问：AD 获取的电压值如何用串口屏显示的问题，在这里统一回答下：

要解决这个问题，要使用 C 语言的 sprintf 这个语句，具体语句的详细内容可以自行百度下，这里仅提供简单使用方法：

1、 sprintf 是需要 stdio.h 来声明的，因此需要在程序开头使用：

```
#include "stdio.h"
```

此函数大约需要 3K 左右的空间；

2、 声明一个存储空间，用于存放需要显示的字符串

```
char buf[100];        //要求命令串长度不超 100 字符
```

3、 假设由 AD 取回并转换成电压的浮点数 vol

```
float vol;     //vol 变量是浮点数
vol=1.253;     //vol 为 1.253V，可由 AD 采样在此步赋值
```

4、 产生送给 gpu 的命令字符串

```
sprintf(buf,"DS12(0,0,'电压：%.3fV',1);\r\n",vol);
```

5、 发送给 gpu

```
GpuSend（buf）;
```

【重要说明】：

1、由于 GPU 系统允许接收命令组，因此串口传入的名利必须以 0x0d,0x0a 结束（就是常说的回车换行,字符串中的\r\n),不发送这个，系统会一直等待下去，表现为发送命令不起作用！

2、如果接不通，建议 RX TX 反一下，有些下载线是指接入单片机端的标志，不是自身标示；

3、GPU 执行指令需要时间，因此快速发送指令时，需要按需求区分两种情况处理：

情况一：重要界面确保显示；需要延时足够的时间，或延时到串口收到“OK”字符为止；

情况二：数据刷新，宁丢勿慢；常用在 UI 界面上数值调整，比如有+ - 键，按住不放，数值不停的增加或减少，此时直接不停的发就可以，漏点无所谓，但最后一次传的一定可以正确显示。

十一、 程序框架与编程思路

见到很多用户写的程序,以及咨询的问题,所以增加此节来详细讲解串口屏的编程思路;

首先,串口屏不是一般的 TFT 显示屏,一般的显示屏需要自己一个点一个点的操作屏显，因此需要讲各种应用写出函数，然后再调用函数，这样操作难度非常

大，需要了解硬件资源以及各种显示技巧，需要非常高速的 MCU，优点当然是 MCU 与屏的通讯带宽很高，可以做大幅度的实时显示；但是分析常用的界面，其实实时高速显示并非单片机常用的需求；

比如，作为仪表显示，实际刷新速度，最快也就是 1 秒 3 次左右，再快人眼也反映不过来，因此，串口屏的优势就出来了，串口屏的特点是：功能封装，调用简单，但传输带宽小

因此，请把需要显示的界面分成固定的与需要刷新的，如 Arduino 的编程思想一样，讲每个界面都分成 setup()和 loop() 两个结构，再 setup 中完成界面以及各个变量的初始化，再 loop()中循环刷新显示测量的数据；

由于串口屏有 1K 的传输缓存区，因此最好每个结构中调用串口屏可以将语句陆续传出，不传最后的 0D 0A,串口屏就不会执行，然后最后传一个 0D 0A 就可以一下显示出来，一点都不拖泥带水；

比如本屏带的示例：1~6S 的锂电池电压显示仪中的代码：

http://pan.baidu.com/share/link?shareid=2710107915&uk=3204894695

主函数：

```
int main(void)
{
    system_init(); //系统初始化 GPIO，串口等与硬件有关的
    Delayms(200000);//延时 200ms,等待串口屏确保启动
    setup();//显示背景界面

    while(1)
    {   loop(); //显示每次刷新测试的值

    }
}
```

可以看到，主函数中，构造出来类似 arduino 一样的 setup 与 loop 结构

Setup 程序：

```
void setup(void)
{   GpuSend("DR2;CLS(0);DS24(4,0,' 锂 电 池 电 压 ',1);DS24(160,0,' 总 电压:',4);BOX(0,30,319,130,11);\r\n");
    Delayms(200000);
}
```

固定显示一些内容，延时 200ms；如果显示的固定信息比较复杂，建议使用批页面显示，每个批页面可存储 1K 语句，一个批页面不够时，可以在批页面最后一个语句使用 SPG(下一个批页面号)；讲批页面级连起来完成负载的页面显示；

Loop 程序：

```
void loop(void)
{       u8 i,y,c;
        u16 x1,x2;
```

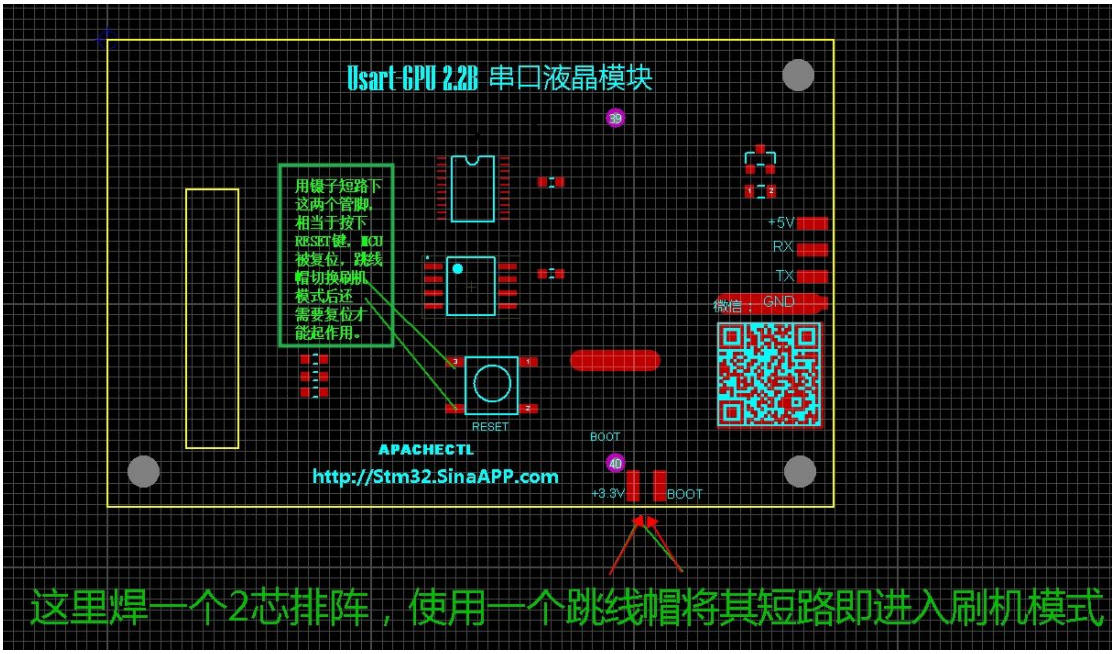
```
        CompADC();
        CompVol();
        CompLi();
        CompCellPer();
        min=12;max=0;argv=0;
        js=0;
        for (i=0;i<6;i++)
        {   if (Li[i]>1)
            {   argv+=Li[i];
                js++;
                if (Li[i]<=min) min=Li[i];
                if (Li[i]>max) max=Li[i];
            }
        }
        argv=argv/js;
        for (i=0;i<6;i++)
        {   x1=i*54+10;x2=x1+30;
            if (Li[i]>1)
            {
    sprintf(buf,"ICON(%d,165,1,11,1,%d);DS16(%d,224,'%.2f',4);",54*i,CellPer[i]
,54*i+4,Li[i]);
                GpuSend(buf);
                y=(Li[i]-argv)*100;
                y=80-y;
                c=2;
                if (y>130) {y=130;c=1;};
                if (y<30) {y=30;c=1;};

    sprintf(buf,"BOXF(%d,31,%d,%d,0);BOXF(%d,%d,%d,129,%d);",x1,x2,y,x1,y,x2,c)
;
                GpuSend(buf);
            }
            else
            {   sprintf(buf,"ICON(%d,165,1,11,1,10);DS16(%d,224,'                ---
',4);BOXF(%d,31,%d,79,0);BOXF(%d,81,%d,129,0);",54*i,54*i+4,x1,x2,x1,x2);
                GpuSend(buf);

            }
        }
        sprintf(buf,"DS24(240,0,'%.2f',4);",LV[js-1]);
        GpuSend(buf);
        GpuSend("PL(0,80,320,80,11);\r\n");
        Delayms(1000000);
}
```

Loop 程序比较复杂，通过 AD 测量的值，决定各个图标以及柱状图的显示颜色和比例，这些都不用管，重要的**是每个 GpuSend 后面都不带 0d 0a，直到最后一条 GpuSend 语句才发送 0d 0a,发送后延时 1 秒**，这样确

保显示连贯，不闪烁，且 1s 刷新一次测量值，效果很好；

十二、 升级程序

第一步：按本文第一节中将 GPU 模块与计算机相连；
第二步：认识 GPU 模块上的和刷机相关的接口：



第三步：下载刷机软件： Flash Loader Demonstrator

您可以去官网下载 V2.6 以上的版本，也可以去下我准备的绿色版本：
http://stm32.sinaapp.com/dn/d.php?n=g8

第四步：查看序列号：
在 GPUMake 中，使用 INF 命令获取序列号：

```
send:INF;

SN:U53AC89A3 FN:EF14 RC:320X240
```

U53AC89A3 即是序列号；
备注：早期的版本可能不支持 INF 命令，可以在接好 GPUMake 的，短路下 GPU 上的 RESET，GPU 即传回：

```
Usart-GPU 2.2B  Ver 0.9b B0617
[FFFFXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXFC0]
53AC89A3
```

其中 53AC89A3 前加 U 即 U53AC89A3 就是序列号；
第五步：下载新 ROM；
去网站：http://stm32.sinaapp.com/gpu22b/，有历史的各个版本可以下载，

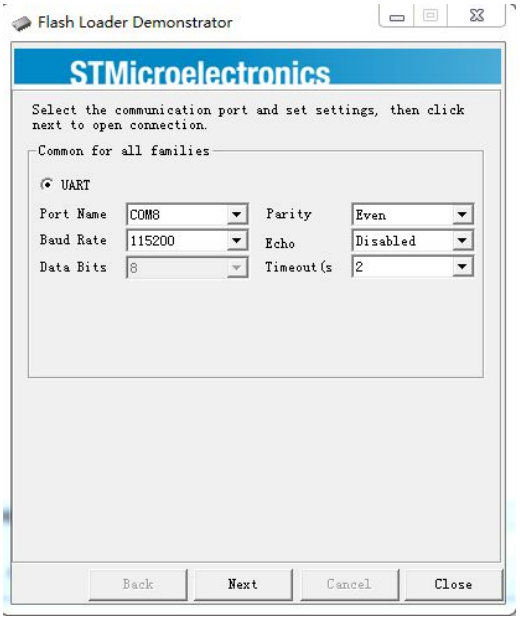
输入序列号，即可下载 bin 文件；

第六步：拔下 USB 刷机线（GPUMaker 的串口程序可能与刷机软件的自动波特率检测冲突，需要拔下设备，再接入 USB 刷机线时会初始化串口）；跳线帽短路 BOOT 后，将 USB 刷机线重新接入电脑；

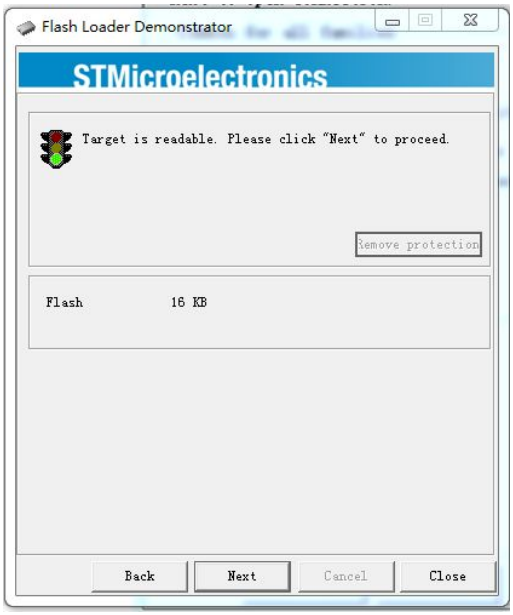
第七步：运行：

STMicroelectronics flash loader.exe

出现：

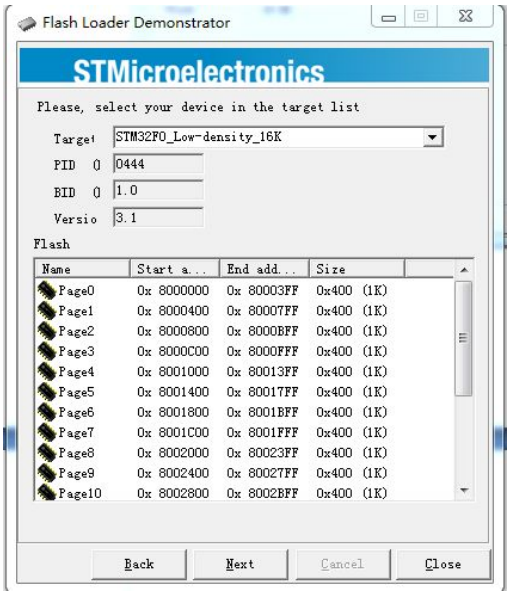


点击 next：

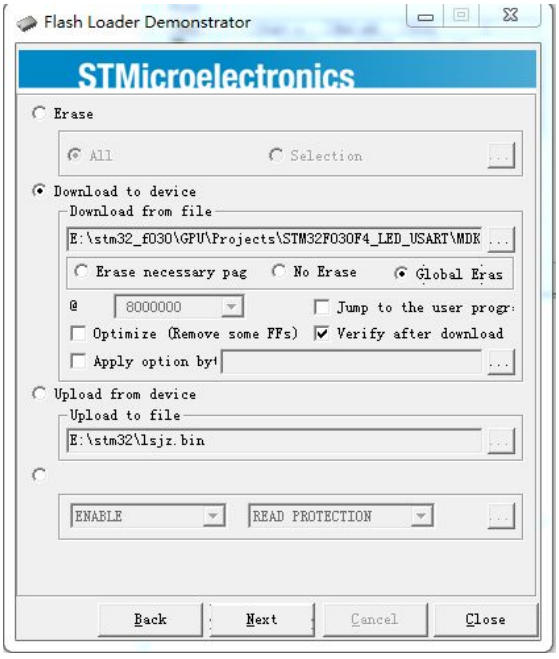


如果不出现此界面，请用镊子短路下 RESET，然后再试一次；如果还不行，请检查一下 BOOT 跳线是否接好；

然后出现：



继续点 next:

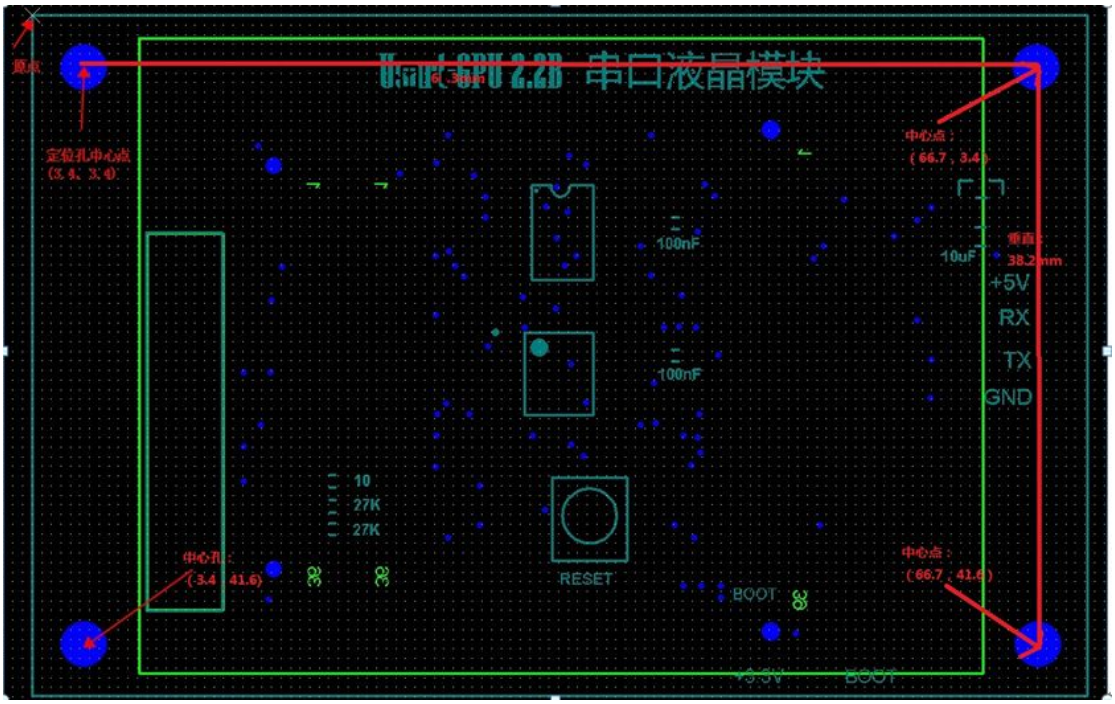


此界面，请按图中设置，选择 Download，且文件选择刚下载的 ROM 文件（BIN 后缀的文件），点击 next;进度条走 2 遍，不显示红色进度条的话，表示刷机成功。

第八步：去掉 BOOT 跳线，重新上电即可启动新版本程序；

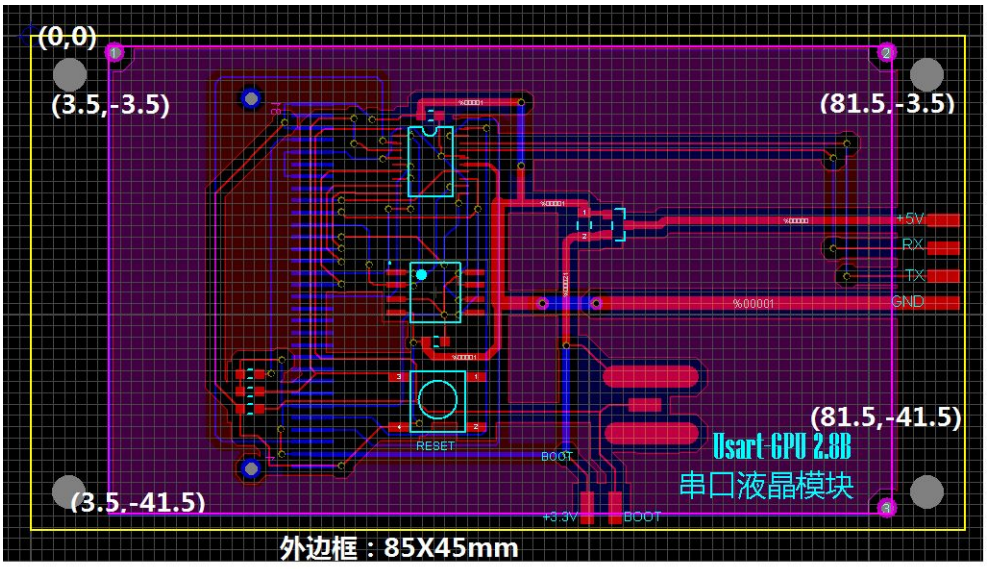
十三、 外形尺寸：

1、2.2 寸外形尺寸：

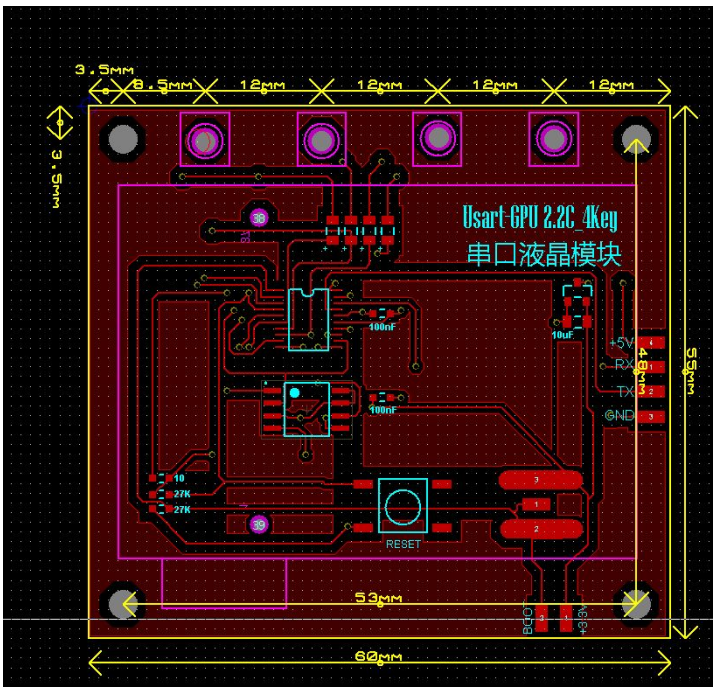


定位孔是为ø3mm 的螺丝设计，PCB 整体长 70mm,宽 45mm;
孔间距:横向：63.3mm，垂直：38.2mm;
液晶屏外框：56X42mm

2、2.8 吋外形尺寸（GPU28A 和 GPU28BTP 带触摸屏的板子定位孔完全一致，只是触摸屏稍厚）



定位孔是为ø3mm 的螺丝设计，PCB 整体长 85mm,宽 45mm;
孔间距:横向：78mm，垂直：38mm;
液晶屏外框：72X42.5mm(约，液晶屏为窄边框设计)



GPU22C4key 面板尺寸。

十四、 量产方案

Usart GPU 模块使用的 W25Q16 存储器数据和 MCU 芯片无关，因此，当调试好一个样品后，少量的可以通过 GPUMaker 写入，但量产这个效率非常低；建议直接使用在线编程器在线烧写或从板子上焊下 25Q16，直接使用片对片拷贝的方式用编程器烧写；大多数编程器全编程（檫除、写入、校验）可以在 20 秒内完成；

如果您将串口屏应用到产品上，请先使用标准屏调试，形成产品后，获取存储器数据可以采用 OEM 订单的方式获取批量产品，甚至可将电路嵌入您产品的 PCB 中；

十五、 QA

1、乱码问题：

程序里书写的汉字到串口屏中，显示乱码，但是英文字母正常；

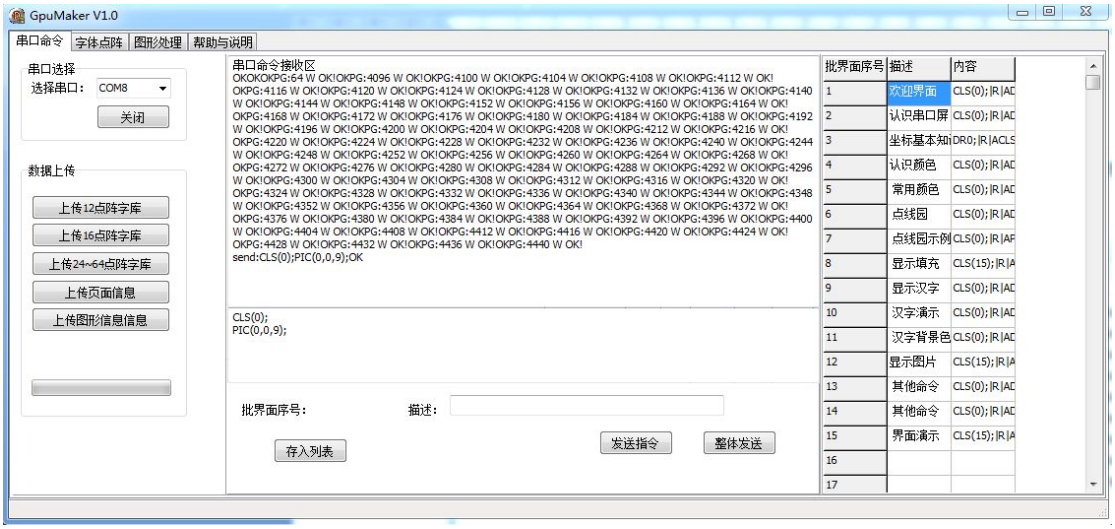
原因：你的程序编辑环境是 UTF8 的，因此写入程序的汉字时 UTF8 字符集的，需要找到程序编辑器设置为 GB2312 或 GBK 格式就正常了；或者用外部的编辑器存为普通格式或 GBK 格式即可；

2、连接没有反应

检测第一串口是不是 RX TX 接反了；第二检测送的语句是否已 0d 0a 结尾；

3、上传的数据不正常

表现为：上传的大字体点阵显示不正常或者图片不正常，请上传的时候关注输出：



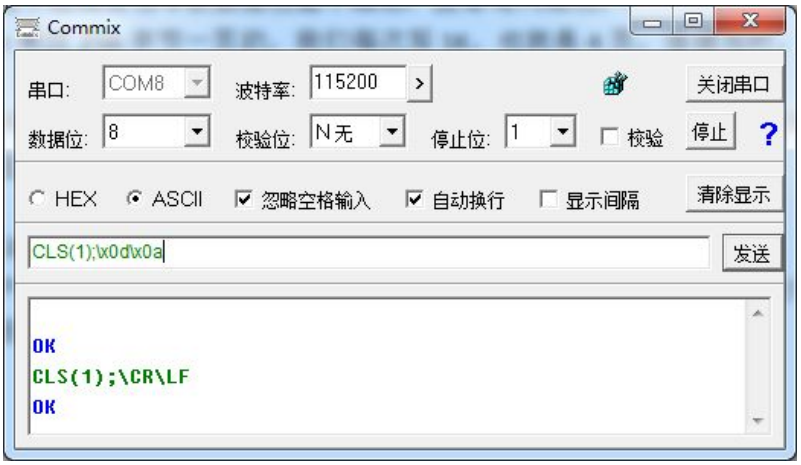
如果是如图输出：中间出现 PG：xxxx 就表示正常，如果出现一堆 OK，没有 PG：xxx，表示传输数据中出现丢包导致数据校验不成功，没有写入成功；

SPI Flash 是没 256 字节一页的，我们每次写 1K，也就是 4 页，连续写时，PG 是每次增加 4；

遇见此问题的童鞋可以换一根 TTL 线试试，或者换台电脑；TTL 线电平较低，不能传输很远距离，且容易受到干扰，一般接线不超过 20cm 最好；

4、关于使用串口助手的问题

很多用户反应使用 GPUMaker 通信没问题，但使用串口助手发送指令却没有反应，原因是串口助手没有发送结尾的 0d 0A，我的串口组手是这样使用的：



需要输入\x0d\x0a　这样转义才可以，具体您使用的串口助手需要如何输入 0d 0a ，请参考串口助手的说明书.

第二部分：高级应用

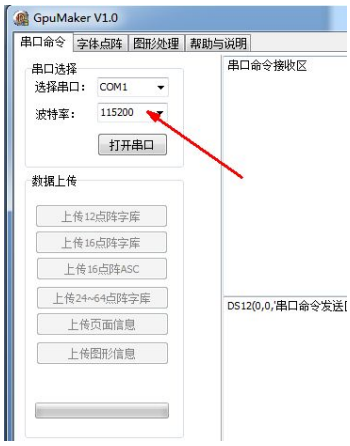
说明：高级应用属于高手使用的部分，这部分会用到比较多的技巧和基本知识，因此我们无法提供更多客服，请自行参考使用；

一、 改变产品的波特率

很多 51,52 的用户提出，89C51,89C52 之类大家熟悉的单片机由于设计过于久远，不能提供 115200 波特率，最高只能提供 9600 的波特率，因此，不能使用串口液晶屏；因此对于需要使用 9600 波特率的用户，需要按下列步骤使用：

1、 将串口屏刷到 V1.0 以上的版本，刷机方法参见第一部分 第 11 节：升级程序；

2、 下载新版本的 GPUmaker，新版本增加了波特率选择；



3、 在 1 号批界面顶头加 U3;(本例中将波特率设置成 19200) 三个字母表示设置波特率；

波特率支持：2400,4800,9600,19200,38400,57600,115200,256000

对应关系：

U0; //2400

U1; //4800

U2; //9600

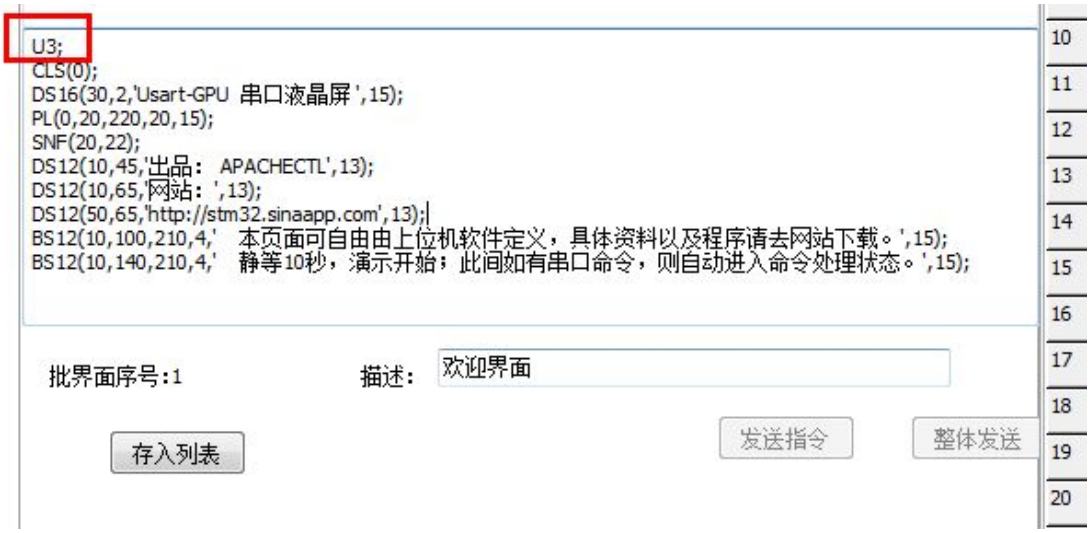
U3; //19200

U4; //38400

U5; //57600

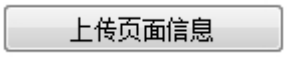
U6; //115200

U7; //256000



U3；请务必第一行顶头书写；

请不要忘记点击： 

，然后点击 

将设置好的上传到 GPU；

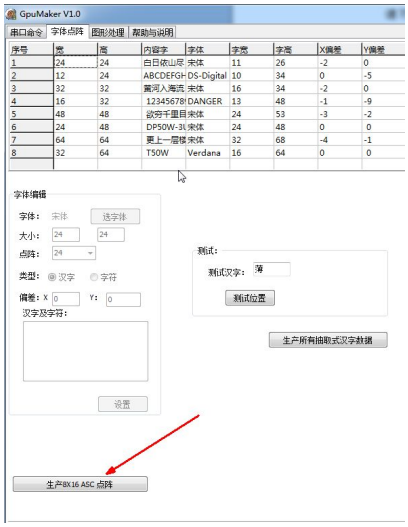
4、 重新给串口屏 GPU 上电，此时开机界面的序列号后面显示：B:19200 表示当前波特率为 19200；此时用单片机的 19200 的串口就可以正常使用 GPU 串口液晶屏了；
5、 重新设置了 GPU 波特率了，相应的 GPUMaker 程序也必须使用新的波特率才能正常连通；
6、 如果波特率设置的较高，超出了电脑 TTL 串口的波特率限制，无法使用 GPUMaker 连通以降低波特率，请使用刷机软件刷回 0.9 版，0.9 版本固定波特率 115200；
7、 9600 下，gpuMaker 传输图形等大量数据需要花费更长的时间，请做好心里准备；

【重点强调：】U3；不是命令，因为你不能在已经按 115200 波特率连接下的串口用串口命令修改波特率，U3；只是一个存储标志，GPU 是在开机的时候检测这个标志，然后按标志对应的波特率初始化串口，**因此需要上传批页面才能起效；**

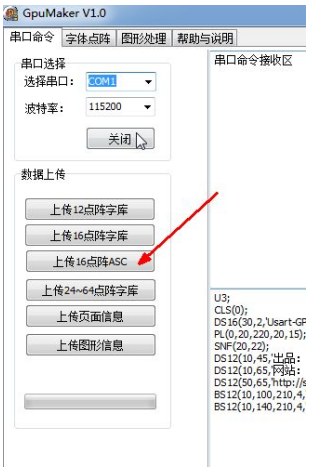
二、 关于 asc8 点阵问题

当您升级为 V1.0 后，对于 220X176 的版本，会发现在 DS16 输出英文字符的时候会显示空白方框，此现象为 asc8X16 点阵缺失导致，解决方法是：

1、 将新版本的 gpumake.exe 覆盖原有目录下的 gpumaker.exe;
2、 将新版本 work 目录下的 asc 文件拷入源 gpumake/work/ 目录下
3、 点击：



4、 点击：



将 asc8X16 点阵数据上传

5、 重启 GPU 串口屏

三、 关于 Arduino 如何使用串口屏，请参见文档：

http://pan.baidu.com/share/link?shareid=2873136112&uk=3204894695