

深圳市君同科技有限公司	文档编号	版本号	密级
	202210151010	V1.0	对外公开
文档名称	七彩显示屏串口交互协议	日期	2022-10-15

七彩显示屏串口交互协议

文档作者: Yangshun 日期: 2022-10-15
审 核: 日期:
批 准: 日期:



深圳市君同科技有限公司 版权所有

文档修订记录

序号	版本号	变化状态	变更 (+/-) 说明	作者	日期
1	V1.0	C	创建	Yangshun	2022-10-15
2	V1.1	M	拆分免费版本协议	Zhaojun	2023-2-21
3	V1.2	A	增加文字例程	ZhaoJun	2023-3-12

*变化状态：C——创建，A——增加，M——修改，D——删除

目 录

1、 简介.....	- 1 -
1.1 文档目的.....	- 1 -
1.2 适用范围（可选）.....	- 1 -
2、 调试验证.....	- 2 -
2.1 硬件准备：.....	- 2 -
2.2 软件准备：.....	- 2 -
2.3 硬件连接：.....	- 3 -
2.4 软件调试.....	- 3 -
3、 设备概述.....	- 6 -
3.1 设备简介.....	- 6 -
3.2 硬件连接.....	- 6 -
4、 串口通信.....	- 6 -
5、 供电要求.....	- 6 -
6、 数据格式.....	- 6 -
7、 数据分包处理.....	- 7 -
8、 数据取模方式.....	- 10 -
9、 数据排列方式.....	- 10 -
10、 应用交互协议.....	- 11 -
10.1 音乐律动模式（麦克风模式）.....	- 11 -
10.2 设置文字数据.....	- 12 -
10.3 设置涂鸦数据.....	- 15 -
10.4 设置动画数据.....	- 15 -
10.5 设置显示模式.....	- 15 -
10.6 设置显示速度.....	- 16 -
10.7 设置显示亮度.....	- 16 -
10.8 设置开关状态.....	- 16 -
10.9 设置镜像状态.....	- 17 -
11、 启动阶段协议.....	- 17 -
11.1 屏端上电上报.....	- 17 -
11.2 屏幕参数上报.....	- 18 -
11.3 屏幕名称上报.....	- 19 -
11.4 启动屏幕.....	- 19 -
12、 文字发送例程.....	- 19 -
12.1 文字指令构成顺序.....	- 19 -
12.2 中文粗体“欢迎”文字发送例程.....	- 20 -

1、简介

1.1 文档目的

1. 梳理大致功能
2. 提供显示屏交互协议，供开发人员使用

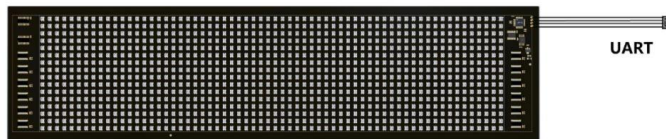
1.2 适用范围（可选）

开发人员

2、调试验证

2.1 硬件准备：

- 1) 我司柔性屏设备一个；



- 2) UART 转 RS232 串口小板一个；

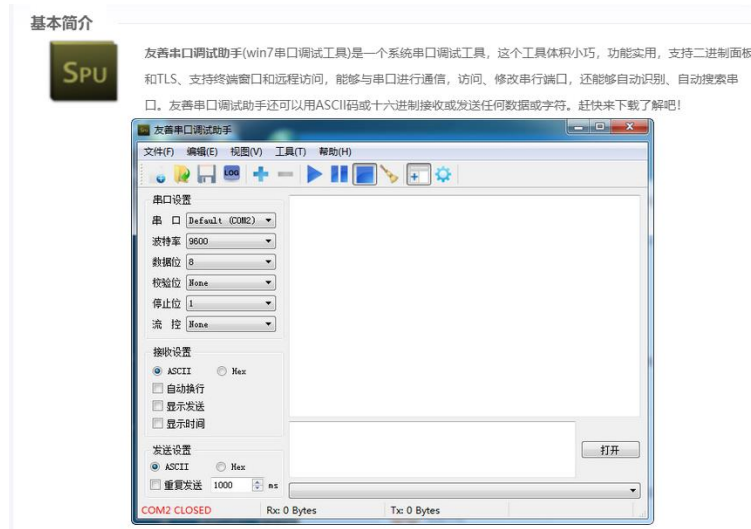


- 3) USB 转串口线一根（台式机可不用）；



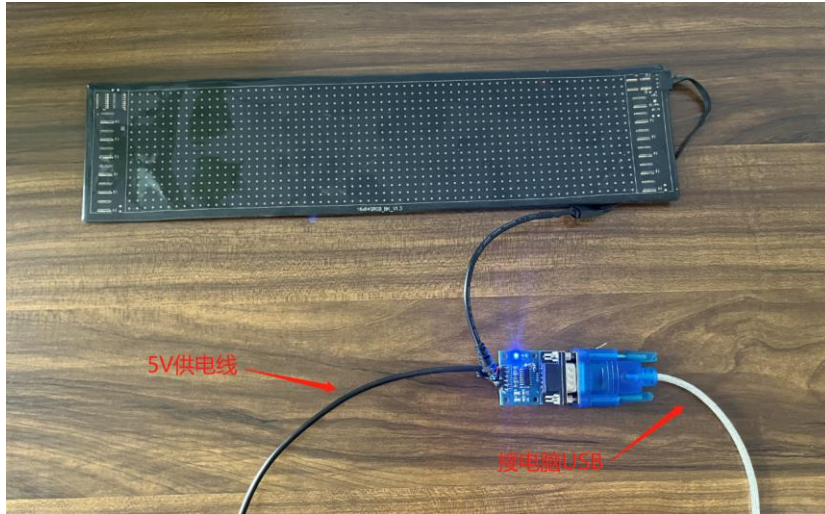
2.2 软件准备：

从网上下载串口调试助手，任意可用的均可，这里推荐“友善串口调试助手”



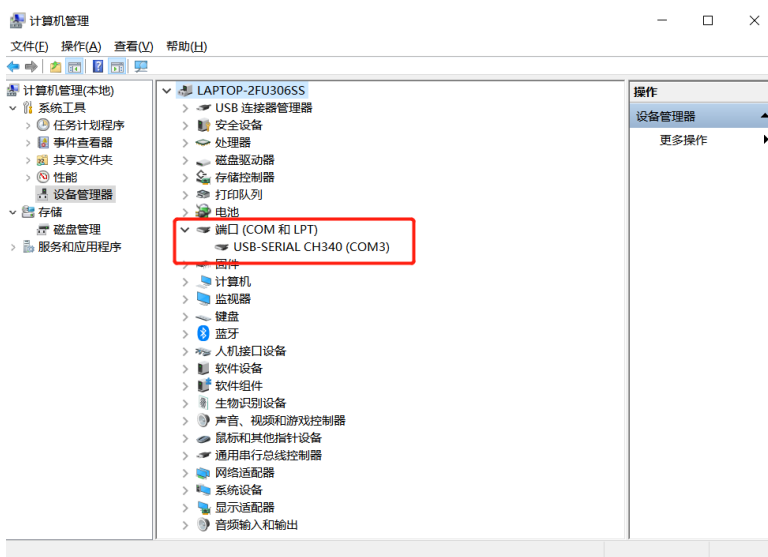
2.3 硬件连接：

将上述硬件设备按照串口连接的基本电气特性连接，即 $RX \rightarrow TX$ ， $TX \rightarrow RX$ 。并将供电线引出，可外接一根 USB 线或者用直流稳压电源均可。电压为 5V。



2.4 软件调试

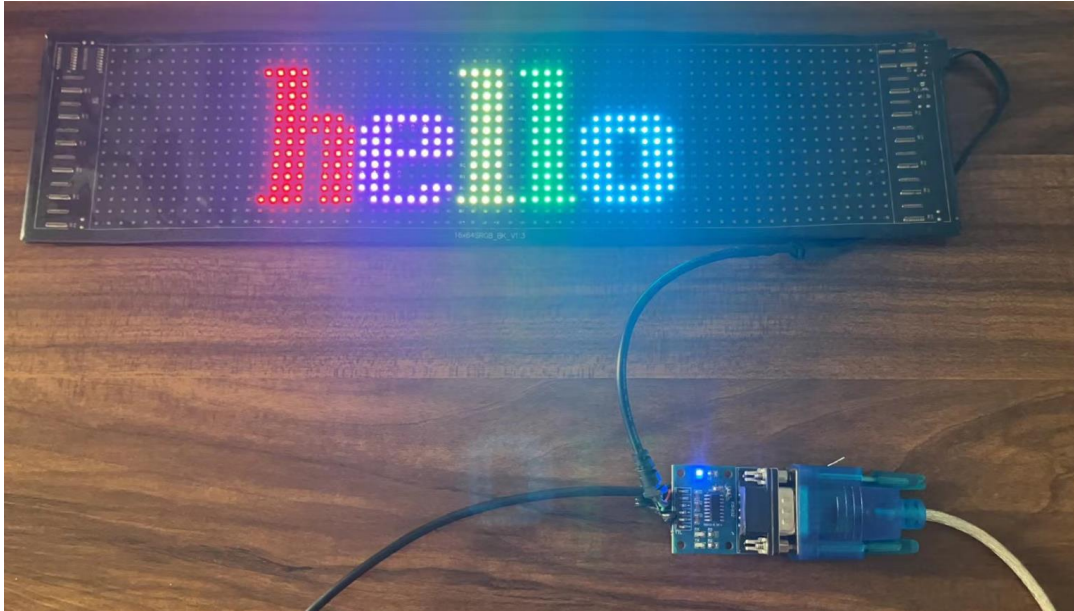
连接成功后，打开电脑，“计算机”→“设备管理器”→“端口”查看串口编号，这里是 COM3。



打开串口调试助手，选择对应的串口编号，波特率 38400，数据位 8，校验位 None，停止位 1，流控 None。接收设置与发送设置均为 Hex。

打开串口，这时，串口会不断重复接收到来自屏幕发送的请求开机命令，01000206320403。如图所示：

此时，观察屏幕，屏幕已经正常工作起来。



接下来，可以发送更多的命令尝试，比如：

静态指令： 0100020606020503
向左： 0100020606020603
向右： 0100020606020703
向上： 01000206060403

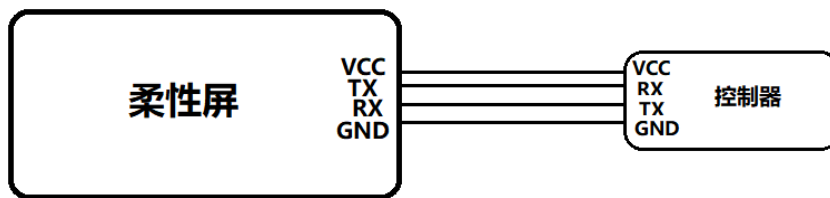
协议具体构成与数据内容部分请看一下内容。

3、设备概述

3.1 设备简介

七彩显示屏，采用柔性屏，滴胶工艺，可直接粘贴，并且能够无缝贴合，不漏光，轻便、轻薄、安装方便。支持设置文字、动画、涂鸦、音乐律动等常见功能。兼顾信息展示和娱乐功能。不同的类型的显示屏，功能可能会有差异。

3.2 硬件连接



4、串口通信

串口采用双向通信方式，串口波特率：38400，停止位 1 位，数据位 8 位，奇偶校验无。

如果不能控制，请交换 RX 和 TX。

5、供电要求

显示屏供要求 5V/2A，使用电压请勿超出范围，否则设备不能正常工作甚至损坏。电压越高，显示屏越亮。

6、数据格式

数据发送需要按照固定的数据格式发送，否则设备不能识别数据。

1	2	3	4	5				n+3	n+4
0x01	n		data					0x03	
起始字节	数据长度		数据					结束字节	

➤ 起始字节

固定值为 0x01，并占用一个字节；在交互的数据中，除了起始位为 0x01，其他数据不可能为该值。

➤ 数据长度

占用两个字节，表示了本次传输的数据段的长度，不包括起始位、结束位和数据长度。一个长度代表一个字节，高字节在前。比如：数据长度为 0x0017，第二个字节对应的数据就为：0x00，第三个字节对应的数据就为：0x17。

➤ 数据

需要发送的数据

➤ 结束字节

固定值为 0x03，并占用一个字节；在交互的数据中，除了起始位为 0x03，其他数据不可能为该值。

➤ 数据强制转换

除起始位和结束位，其他数据在发送之前都需要做一个强制转换，转换要求如下：

如果发送的数据中的某个字节小于 0x04，并且大于 0x00，那么就需要在该字节数据之前添加一个值为 0x02 的字节，并把该字节和 0x04 异或，然后再发送。添加的 0x02 不计入在数据长度之内。

参考：

```

test.js
1 function getData(ch) {
2     var out = [];
3     if (ch > 0x00 && ch < 0x04) {
4         out.push("02");
5         ch ^= 0x04;
6     }
7
8     out.push(ch.toString(16));
9
10    return out;
11 }
    
```

示例：

比如需要发送一段数据为：0x00 0x01 0x05

数据长度为：0x0003，由于数据长度的高字节 0x00 不在强制装换范围内，所以保持不变，而 0x03 需要强制转换为：0x02 0x07

最终发送的数据为：0x01 0x00 0x02 0x07 0x00 0x02 0x05 0x05 0x03

7、数据分包处理

由于数据太长，并且设备 RAM 空间有限，一次不能接受太长的数据，并且为了提高传输的可靠性，所以需要某些发送的数据做分包处理。

package 数据格式：

包类型	数据总长度	包 id	当前包数据长度	当前包数据	校验
1byte	2byte	2byte	1byte		1byte

预留字节，默认为 0	0~0xFFFF 指的是未分包前的原始数据的总长度。	0~0xFFFF 每次设置新的文字，包 id 从 0 开始，0 为该次传输的起始包，然后依次递增	取值范围：1~128 字节（建议每包的数据长度为 128 字节，最后一包除外）		0x00 和前面的所有数据，按字节，依次取异或，所得到的最终值
------------	----------------------------	--	---	--	---------------------------------

注意：

1. 每个包数据需要附带消息类型。相当于发送的数据为：**type+package**。
2. 发送的包数据（**type+package**）都需要通过[数据格式](#)转换，设备才能正常识别。
3. 每次发送的包数据（**type+package**），设备都会进行回复，APP 需要处理回复的消息。
4. 收到回复后，app 才能开始下一个动作（传输下一包、重传数据、显示传输失败等等）

应答：

<type> 消息类型，占一个字节。

<ack> 对应的包 id 传输结果，组成如下：

包类型	包 id	error_code
1byte	2byte	1byte
预留字节，默认为 0	0~0xFFFF	取值如下： 0x00 传输成功 0x01 传输失败 0x02 设备异常 0x03 数据有误 0x04 数据长度错误 0x05 数据 ID 错误 0x06 数据校验错误

APP 在传输的时候，得考虑超时，每个包发送后如果 1s 都没有收到回复，说明传输超时，需要重新传输（重新开始传输），如果【连续】重传超过 3 次，都传输失败，表示该次设置涂鸦数据失败。

注意：回复的数据也满足[数据格式](#)的要求，所以对回复的数据进行反解析。

反解析参考代码 C 语言：

```

#define PT_START_CHAR      0x01 //起始标志
#define PT_ESC_CHAR       0x02 //协议标志
#define PT_ESC_XOR_CHAR   0x04 //协议异或值
#define PT_END_CHAR       0x03 //结束标志
#define PT_MAX_LEN        1024 //接收的最大数据长度
#define UART_PARSE_BUFF_LEN 10 //串口解析数据的buff长度

typedef enum UART_REC_STATUS_E
{
    E_UART_REC_STATUS_DATA = 0, //接收数据
    E_UART_REC_STATUS_START, //接收起始信号
    E_UART_REC_STATUS_DATALEN_H, //接收数据长度高位
    E_UART_REC_STATUS_DATALEN_L, //接收数据长度低位
} UART_REC_STATUS_E;

static UART_REC_STATUS_E s_eUartRecStatus = E_UART_REC_STATUS_START;
static uint16 s_ul6UartRecIndex = 0;
static uint16 s_ul6UartDataLen = 0;
static uint8 s_u8Action = E_ACTION_UNKOWN;
static uint8 DATA s_au8UartParseBuff[UART_PARSE_BUFF_LEN] = {0};
static BOOL_t s_bInEsc = FALSE;

static void iInitUartRec(void)
{
    s_eUartRecStatus = E_UART_REC_STATUS_START;
    s_ul6UartRecIndex = 0;
    s_ul6UartDataLen = 0;
    s_u8Action = E_ACTION_UNKOWN;
}

//反解析数据
void JTUART_RecByteCallback(uint8 byte)
{
    switch (byte)
    {
        case PT_START_CHAR: //开始标志
            s_ul6UartDataLen = 0;
            s_bInEsc = FALSE;
            s_eUartRecStatus = E_UART_REC_STATUS_DATALEN_H; //等待接收数据长度
            break;
        case PT_ESC_CHAR: //0x02
            s_bInEsc = TRUE;
            break;
        case PT_END_CHAR: //结束标志
            iParseUartComplete(); //解析完成
            s_eUartRecStatus = E_UART_REC_STATUS_START; //等待接收开始标志
            break;
        default:
            if(s_bInEsc) //0x02
            {
                byte ^= PT_ESC_XOR_CHAR; //异或操作
                s_bInEsc = FALSE;
            }

            switch(s_eUartRecStatus)
            {
                case E_UART_REC_STATUS_DATA: //接收数据
                {
                    if (s_ul6UartRecIndex < s_ul6UartDataLen)
                    {
                        iParseUartData(byte);
                        s_ul6UartRecIndex++;
                    }
                }
                break;
                case E_UART_REC_STATUS_DATALEN_H: //接收数据长度高字节
                {
                    s_ul6UartDataLen = byte << 8;
                    s_eUartRecStatus = E_UART_REC_STATUS_DATALEN_L;
                }
                break;
            }
    }
}

```

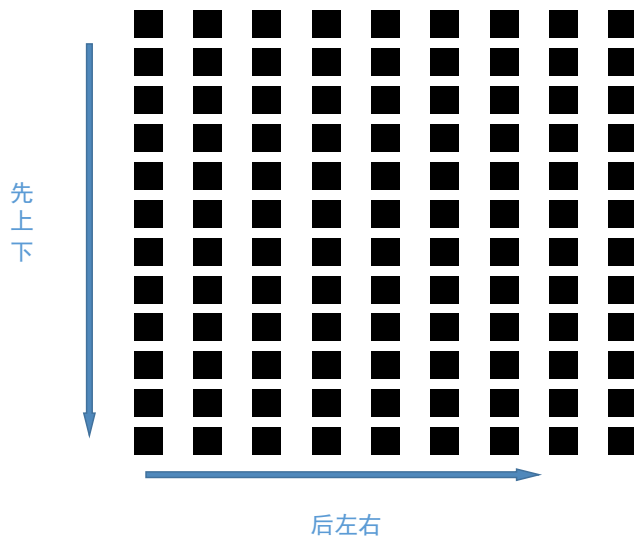
```

        case E_UART_REC_STATUS_DATALEN_L: //接收数据长度低字节
        {
            s_ul6UartDataLen |= byte;

            if (s_ul6UartDataLen > PT_MAX_LEN)
            {
                s_eUartRecStatus = E_UART_REC_STATUS_START;
                s_ul6UartDataLen = 0;
            }
            else
            {
                s_eUartRecStatus = E_UART_REC_STATUS_DATA;
                s_ul6UartRecIndex = 0;
            }
        }
        break;
    default:
        break;
    }
    break;
}
}
}

```

8、数据取模方式



取模方式采用横向-先上下，后左右，高位在前的方式。数据依次排列。比如单色的 12x48 的分辨率，每列就需要两个字节表示，一屏一共需要 2x48=96 个字节。

9、数据排列方式



所有数据都是由 RGB 三个分量组成，每个点的每个分量由一个位表示。比如 16x64 分辨率的七彩屏幕，每列需要 2 个字节，所以红色分量需要 128 个字节，绿色分量需要 128 个字节，蓝色分量需要 128 个字节。一屏数据一共就是 384 个字节。数据排列方式按照先排列红色分量的 128 个字节，再是绿色分量的 128 个字节，最后是蓝色分量的 128 个字节。

10、应用交互协议

该部分的协议，只有在屏幕启动完成之后，才有效。

10.1 音乐律动模式（麦克风模式）

命令说明：应用播放音乐或者处于麦克风状态的时候，提取音乐节奏，转换为柱状数据，并配上每列的颜色，发送给模块。

命令格式：

Direction	Type	Params	Response
APP→LIGHT	0x01	<第一列高度><第二列高度>.....<第八列高度><第一列颜色><第二列颜色>.....<第八列颜色>	无

数据说明：

<0x01> 消息类型，占一个字节，并且一定是一段蓝牙数据的第一个字节

<第 X 列高低> 第 x 列柱状数据的高度，每列一个字节

<第 X 列颜色> 第 x 列柱状数据的颜色，每列用一个字节表示。

红色表示方式：0x04

绿色表示方式：0x02

蓝色表示方式：0x01

黄色表示方式：0x06

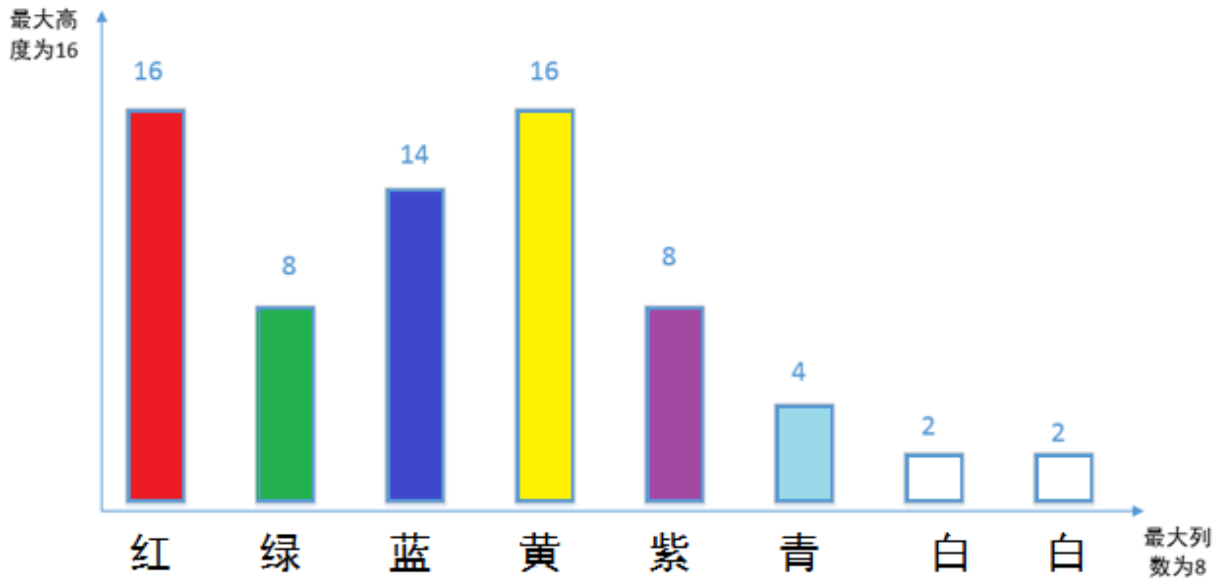
紫色表示方式：0x05

青色表示方式：0x03

白色表示方式：0x07

应答：无

➤ 示例：音乐律动模式需要生产柱状图的数据，如下图：



只需要生成最大高度为 16，一共 8 列的数据，按照顺序排列。后面附上颜色，加上消息类型，那么需要发送的数据如下：

```
01 10 08 0E 10 08 04 02 02 04 02 01 06 05 03 07 07
```

然后把上面的数据按照数据格式转换一次，得到最终的发送数据为：

```
01 00 11 02 05 10 08 0e 10 08 04 02 06 02 06 04 02 06 02 05 06 05 02 07 07 07 03
```

备注：

为了让颜色变化有更好的节奏感，个人建议方案：

1. 如果某一列音乐柱高度大于了 12（相当于重音），那么这一列就随机切换一次颜色。反之颜色保存不变。（颜色切换不能过快，两次颜色切换最好有个几十毫秒的间隔，否则不能区分颜色）
2. 开始的时候，默认音乐柱颜色为红色。

10.2 设置文字数据

命令说明：通过该命令，设置设备当前显示的文本内容。

命令格式：

Direction	Type	Params				Response
APP→LIGHT	0x02	参数名称	长度	描述	备注	<0x02><ack>
		<reserved>	23byte	预留字节	默认为 0	
		<font-num>	2byte	需要显示的文字个数	1. 12x48 最大文字个数为 123 个 2. 16x32 七彩显示屏，最大文字个数为 105 个 3. 16x64 七彩显示屏，最大文字个数 318 个	

					4. 16x96 七彩显示屏，最大文字个数 318 个 5. 16x128 七彩显示屏，最大文字个数 318 个 6. 16x192 七彩显示屏，最大文字个数 318 个	
		<e-font-size>	80byte	预留字节	默认为 0	
		<data-size>	2byte	文字点阵数据的总长度		
		<data>	nbyte	对应的文字的点阵显示数据	不同设备类型，数据组成方式不一样	

数据说明：

<0x02> 消息类型，占一个字节，并且一定是一段蓝牙数据的第一个字节

< Params > 参数和数据。该数据需要进行分包处理，详情请见[数据分包处理](#)。

应答：

参考分包应答数据[数据分包处理](#)。

备注：

1. 字库可采用点阵字库或者采用矢量转点阵的方式，16x64 的分辨率的屏幕，建议采用 16x16 的点阵字库。字库数据排列方式要满足上文的数据排列方式就行。
2. 文字数据生成规则
 由于是 7 彩，每个点就有黑、红、绿、蓝、黄、紫、青、白，一共 8 种颜色，所以每个点的数据需要用 3 个位（0~7）去表示。并且数据格式按照『所有点的红色 R 分量』『所有点的绿色 G 分量』『所有点的蓝色 B 分量』顺序排列，如下图所示：

欢迎光临！ 欢迎光临！	欢迎光临！ 欢迎光临！	欢迎光临！ 欢迎光临！
-------------	-------------	-------------

R分量

G分量

B分量

因为一个点的颜色需要用 3 个位去表示，那么：

黑色表示方式：000（所有分量都为 0）

红色表示方式：100（红色分量有效，其他分量为 0）

绿色表示方式：010（绿色分量有效，其他分量为 0）

蓝色表示方式：001（蓝色分量有效，其他分量为 0）

黄色表示方式：110（红色和绿色分量有效，其他分量为 0）

紫色表示方式：101（红色和蓝色分量有效，其他分量为 0）

青色表示方式：011（绿色和蓝色分量有效，其他分量为 0）

白色表示方式：111（所有分量都有效）

所以，只需要根据文字颜色，先从字库中读取数据，然后根据颜色，拷贝一份到对应的分量。比如当前文字为黄色，黄色是红色和绿色分量都有效，其他分量为 0。那么文字的红色分量=绿色分量=文字数据，蓝色分量全为 0。

示例:



如上图所示，字库为 UNICODE12，每个文字的数据如下：

- 『欢』，颜色为红色

该文字的字库数据为：

```
0x50,0x20,0x4c,0xc0,0x43,0x00,0x4c,0xc0,  
0x78,0x10,0x10,0x20,0xe0,0xc0,0x27,0x00,  
0x20,0xc0,0x28,0x20,0x30,0x10,0x00,0x00,
```

根据颜色规则，那么红色分量数据就为字库数据，绿色和蓝色分量数据全为 0。

红色分量：

```
0x50,0x20,0x4c,0xc0,0x43,0x00,0x4c,0xc0,  
0x78,0x10,0x10,0x20,0xe0,0xc0,0x27,0x00,  
0x20,0xc0,0x28,0x20,0x30,0x10,0x00,0x00,
```

绿色分量：

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
```

蓝色分量：

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
```

- 『迎』，颜色为黄色

该文字字库数据为：

```
0x88,0x10,0x4f,0xe0,0x00,0x10,0x7f,0x90,  
0x41,0x10,0x82,0x10,0x00,0x10,0x7f,0xf0,  
0x40,0x10,0x40,0x90,0x7f,0x90,0x00,0x00,
```

根据颜色规则，那么红色分量数据=绿色分量数据=字库数据，蓝色分量数据全为 0。

红色分量：

```
0x88,0x10,0x4f,0xe0,0x00,0x10,0x7f,0x90,  
0x41,0x10,0x82,0x10,0x00,0x10,0x7f,0xf0,  
0x40,0x10,0x40,0x90,0x7f,0x90,0x00,0x00,
```

绿色分量：

```
0x88,0x10,0x4f,0xe0,0x00,0x10,0x7f,0x90,  
0x41,0x10,0x82,0x10,0x00,0x10,0x7f,0xf0,  
0x40,0x10,0x40,0x90,0x7f,0x90,0x00,0x00,
```

蓝色分量：

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
```

数据按照『所有点的红色 R 分量』『所有点的绿色 G 分量』『所有点的蓝色 B 分量』顺序排列，最终的文字数据为：

//所有文字的红色分量数据

```
0x50,0x20,0x4c,0xc0,0x43,0x00,0x4c,0xc0,  
0x78,0x10,0x10,0x20,0xe0,0xc0,0x27,0x00,  
0x20,0xc0,0x28,0x20,0x30,0x10,0x00,0x00,  
0x88,0x10,0x4f,0xe0,0x00,0x10,0x7f,0x90,  
0x41,0x10,0x82,0x10,0x00,0x10,0x7f,0xf0,  
0x40,0x10,0x40,0x90,0x7f,0x90,0x00,0x00,
```

//所有文字的绿色分量数据

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x88,0x10,0x4f,0xe0,0x00,0x10,0x7f,0x90,  
0x41,0x10,0x82,0x10,0x00,0x10,0x7f,0xf0,  
0x40,0x10,0x40,0x90,0x7f,0x90,0x00,0x00,
```

//所有文字的蓝色分量数据

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
```

10.3 设置涂鸦数据

此协议为收费协议，请联系我司业务协调处理。

10.4 设置动画数据

此协议为收费协议，请联系我司业务协调处理。

10.5 设置显示模式

命令说明：通过该命令，通过该命令可以设置图案或者文字显示的模式。

命令格式：

Direction	Type	Params	Response
APP→LIGHT	0x06	<mode>	无

数据说明:

<0x06> 消息类型, 占一个字节, 并且一定是一段蓝牙数据的第一个字节。

< mode > 显示模式, 占一个字节, 取值如下:

模式名称	取值	备注
静态	1	
向左	2	
向右	3	
向上	4	
向下	5	
雪花	6	
画卷	7	
镭射	8	

应答: 无

10.6 设置显示速度

命令说明: 通过该命令可设置显示模式的切换速度。

命令格式:

Direction	Type	Params	Response
APP→LIGHT	0x07	<speed>	无

数据说明:

<0x07> 消息类型, 占一个字节, 并且一定是一段蓝牙数据的第一个字节。

< speed > 显示速度, 占一个字节, 取值范围: 0~255

应答: 无

10.7 设置显示亮度

命令说明: 通过该命令可设置显示的亮度。

命令格式:

Direction	Type	Params	Response
APP→LIGHT	0x08	<bn>	无

数据说明:

<0x08> 消息类型, 占一个字节, 并且一定是一段蓝牙数据的第一个字节。

< bn > 显示亮度, 占一个字节, 取值范围: 0~255

应答: 无

10.8 设置开关状态

命令说明: 通过该命令可设置设备的开关状态

命令格式:

Direction	Type	Params	Response
APP→LIGHT	0x09	<status>	无

数据说明:

<0x09> 消息类型, 占一个字节, 并且一定是一段蓝牙数据的第一个字节。

< status > 开关状态, 占一个字节, 取值: 0-关闭, 1-打开

应答: 无

10.9 设置镜像状态

命令说明: 通过该命令可设置设备的镜像状态

命令格式:

Direction	Type	Params	Response
APP→LIGHT	0x0C	<status>	

数据说明:

<0x0C> 消息类型, 占一个字节, 并且一定是一段蓝牙数据的第一个字节。

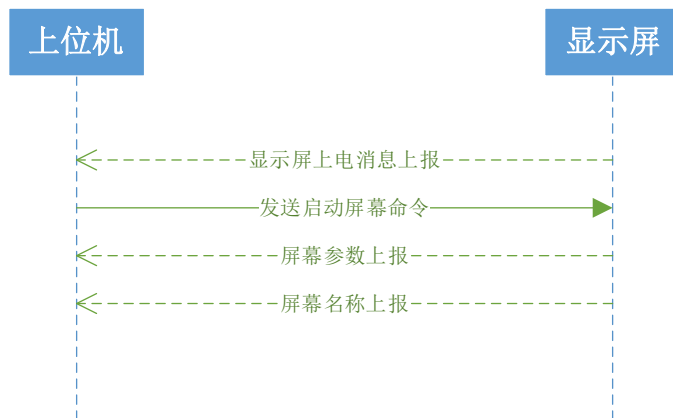
< status > 镜像状态, 占一个字节, 取值: 0-关闭, 1-打开

应答:

无

11、 启动阶段协议

该阶段的命令是屏幕启动阶段的交互命令, 需要按照指定的顺序发送命令, 屏幕才能正常启动。



11.1 屏端上电上报

命令说明: 该命令是屏端上电或者重启后, 主动上报给上位机的。

命令格式:

Direction	Type	Params	Response
LIGHT→APP	0x32	<0x04>	<0x32><0x04><ack>

数据说明:

<0x32> 消息类型, 占一个字节, 并且一定是一段数据的第一个字节。

<0x04> 参数, 值为: 0x04

应答:

<0x32> 消息类型, 占一个字节, 并且一定是一段数据的第一个字节。

<0x04> 参数, 值为: 0x04

<ack> 应答状态, 0-解析成功, 其他-解析失败

备注: 上位机在收到该命令后, 如果需要启动屏幕, 需要发送“启动屏幕”命令。

11.2 屏幕参数上报

命令说明: 屏端在收到上位机的“启动屏幕”命令后, 会在短时间(大约为 50ms)内上报屏幕参数。

命令格式:

Direction	Type	Params	Response
LIGHT→APP	0x30	<ID-H><ID-L><RES-ROW><RES-C-H><RES-C-L><DEV-TYPE><DEV-VERSION>	<0x30><ack>

数据说明:

<0x30> 消息类型, 占一个字节, 并且一定是一段数据的第一个字节。

<ID-H> 设备 ID 的高字节, 占一个字节

<ID-L> 设备 ID 的低字节, 占一个字节

<RES-ROW> 屏幕分辨率行数, 占一个字节

<RES-C-H> 屏幕分辨率列数的高字节, 占一个字节

<RES-C-L> 屏幕分辨率列数的低字节, 占一个字节

<DEV-TYPE> 设备类型, 占一个字节, 取值如下:

值	设备类型	说明
0x00	单色	
0x01	七彩	
0x02	全彩	

<DEV-VERSION> 设备版本号, 占一个字节, 取值: 0~255

应答:

<0x30> 消息类型, 占一个字节, 并且一定是一段数据的第一个字节。

<ack> 应答状态, 0-解析成功, 其他-解析失败

11.3 屏幕名称上报

命令说明：屏端在收到上位机的“启动屏幕”命令后，会在短时间（大约为 50ms）内上报屏幕名称。上位机可通过该名称去区分不同的产品。这一版设备的名称为：CoolLEDX

命令格式：

Direction	Type	Params	Response
LIGHT→APP	0x31	<name-arr>	<0x31><ack>

数据说明：

<0x31> 消息类型，占一个字节，并且一定是一段数据的第一个字节。

<name-arr> 屏幕名称，最长 20 个字节。

应答：

<0x31> 消息类型，占一个字节，并且一定是一段数据的第一个字节。

<ack> 应答状态，0-解析成功，其他-解析失败

11.4 启动屏幕

命令说明：通过该命令，可以启动屏幕。上位机发送该命令之后，屏幕端才会正常启动开机。

命令格式：

Direction	Type	Params	Response
APP→LIGHT	0x23	<0x01>	<0x23><FLAG>

数据说明：

<0x23> 消息类型，占一个字节，并且一定是一段数据的第一个字节。

<0x01> 参数，值为：0x01

应答：

<0x23> 消息类型，占一个字节，并且一定是一段数据的第一个字节。

<FLAG> 标志，占一个字节，上位机不用关心该变量的具体值。

备注：设备启动后，会在很短时间内开始设置屏幕相关参数。

12、文字发送例程

12.1 文字指令构成顺序

第一步：从字库取出字库点阵数据，通常字库取出来只有一组数据，没有颜色数据的区别；

第二步：构建颜色分量数据，并合并数据；

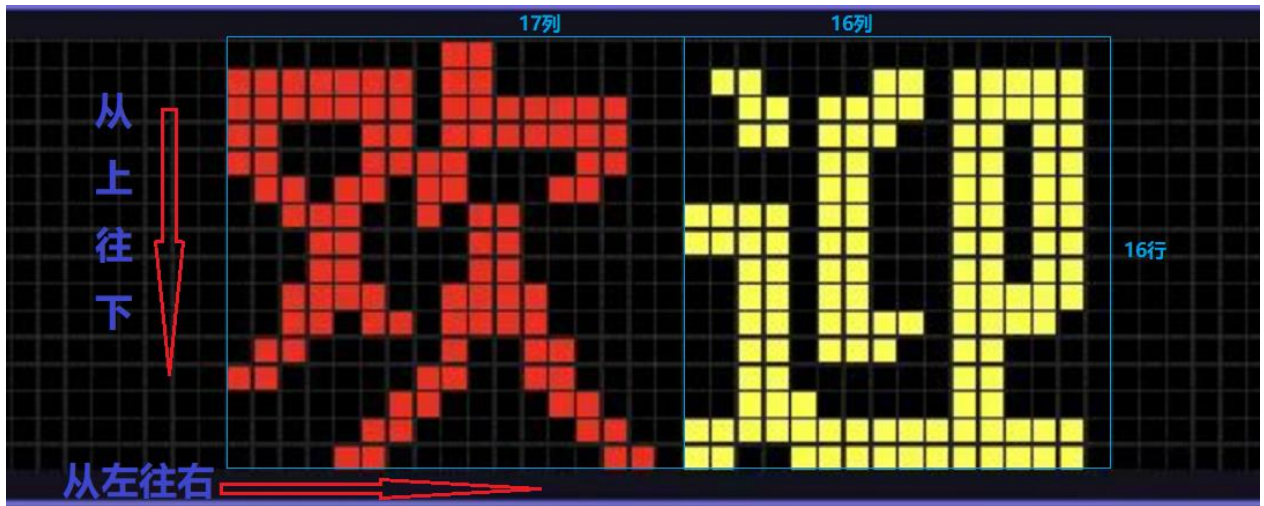
第三步：增加文字参数字节，构建基础数据；

第四步：对基础数据进行分包，并增加校验字节；

第五步：对分包基础数据增加指令类型字节；

第六步：将分包基础数据构建标准格式指令，并强制转换。

12.2 中文粗体“欢迎”文字发送例程



进行第一步之前，我们先回顾一下文字数据的构成，对字库数据进行一些预测。字库从根本上也是点阵数据，遵循上文第 8 节描述的取模方式。上图为我司的粗体字库（我司字库是基于网上字库进行了修整的字库），可以看出“欢”字有 17 列，“迎”是 16 列，均为 16 行。根据从上往下，从左往右的取模规则。“欢”字红色分量 $2*17=34$ 个字节，其它绿色、蓝色也为 34 字节。“迎”字红色、绿色、蓝色均为 $2*16=32$ 个字节。这里和第 8 节的字库不同，因此数据异不同。

注意：我司可提供免费字库，该字库可通过网上下载，或者向我司业务人员获取。亦可提供收费字库，该字库是我司在免费字库进行了修正处理，特别对粗体字进行了多达 2 万多个字的处理。

第一步：取出字库数据

“欢”的 34 字库数据为：

```
78 08 7C 18 66 70 63 E0
67 C1 7C 63 78 26 0E 0C
FC 78 F3 E0 33 E0 30 78
34 1C 3C 06 38 03 00 01
00 00
```

“迎”的 32 字库数据为：

```
03 03 43 03 73 FE 33 FE
00 07 3F F3 3F F3 70 33
60 23 00 03 7F FF 7F FF
60 63 7F E3 7F C3 00 00
```

第二步：构建三色分量数据，第一个“欢”为红色，第二个“迎”是有红色和绿色构成的黄色。

因此，这两个字的红色分量为：

```
78 08 7C 18 66 70 63 E0
67 C1 7C 63 78 26 0E 0C
FC 78 F3 E0 33 E0 30 78
34 1C 3C 06 38 03 00 01
00 00
```