上海仰邦软件科技有限公司

动态区域用户手册

Copyright

Version list:

1.0	2013-04-17	张文忠	初始版本
1.1	2013-04-23	张文忠	更新了包头数据格式，以避免出现非对齐访问
1.2	2013-04-27	张文忠	动态区域的页数更新为 2 个字节表示，以便可以存储更多的页 修改了动态区域 ImmePlay 参数的定义 为避免过多转义，动态区域命令字修改为 0xA7
1.3	2014-10-21	覃忠利	增加 5q 系列卡类型 增加 5q 系列数据编码格式
1.4	2015-03-06	张文忠	抽取了 5E 和 5Q 的共性，去除了部分无关紧要的信息 删除了更新页信息和删除页信息命令

目录

动态区域用户手册

1. 概述

1.1 *功能描述*

类型：双基色(5E)/全彩(5Q)

扫描方式：静态，1/2，1/4，1/8，1/16 扫，用命令可以修改

数据类型：点阵方式

播放方式：节目顺序播放/定长播放可选

通讯方式：RS232/RS485、GPRS、 RF、Ethernet

节目个数：5e 系列 512 个，5q 系列 1000 个

区域个数：同时支持 32 个图文区，4 个动态区

1.2 *通讯方式*

1) RS232/485 波特率 ：9600/57600, 无校验, 8 位数据, 1 停止位(5Q 系列无该通讯方式)
2) GPRS /RF 波特率 ：9600/57600, 无校验, 8 位数据, 1 停止位(5Q 系列无该通讯方式)
3) Ethernet 10/100M 自适应

1.3 动态区域概况

1) 动态区域可与异步节目同时播放，也可以单独播放
2) 控制卡支持 4 个动态区域,每个动态区域最大容量为 300K
3) 动态区域信息不能掉电保存，但可以无限次更新

1.4 *术语和缩略语*

名称	说明
MSB	高位字节（Most Significant Byte）
LSB	低位字节（Least Significant Byte ）
CRC16	16 位的 CRC 校验，校验算法参考附录
CHK	CRC 校验值

1.5 *协议说明*

◆ 本文档中十六进制数据表示为 0x？？，如 0x7E。

- 本文档中涉及到的多字节参数，均以先低字节(LSB)后高字节(MSB)顺序发送，但是对于文件名和控制器名称等字符串参数，发送时按顺序发送，如“P123”则先发送‘P’，最后发送‘3’。
- 本文档中提及的数据长度，如无特别说明，皆是以字节（byte）为单位
- 本文档中提及的时间相关的参数均采用 BCD 码
- 本文档中提及的颜色属性，均用 1Byte 来表示，其中，Bit0 表示红，bit1 表示绿，bit2 表示蓝
- 本文档中所有偏移量、块地址等参数如无特殊说明，均以 0 开始计算 。
- 本文档中区域的坐标定义按照左上角为坐标原点。横、纵坐标分别向右、向下延伸。
- 本文档中提及的“读取”和“写入”都是指上位机对控制器的动作
- 本文档中提及的保留字全部默认发送 0x00。

2. 标准通讯格式

协议结构如下：

帧头 0xA5 (8byte)	包头数据 (16byte)	数据域 (Nbyte)	包校验 (2byte)	帧尾 0x5A (1byte)

以下为协议中各项数据的说明：

1. 帧头由 8 个字节的 0xA5 组成，帧尾由一个字节的 0x5A 组成。帧头采用 8 个帧头，是为了防止 0xA5 丢失导致数据接收错误。在接收数据时，只要接收到一个 0xA5 就可认为接收到了帧头，然后等待下一个不是 0xA5 的数据，该数据为该帧的第一个有效数据。
2. 包头数据包含本包数据的一些属性，其定义参考包头数据格式定义。
3. 数据域为用户协议层数据，参考数据域定义
4. 包校验为包头数据和数据域的 CRC16 校验值，CRC16 校验算法参考附录。

2.1 *协议的分层*

协议采用分层模式，分为协议层和物理传输层两层，其中数据域属于协议层数据。物理传输层又分为 PHY0 和 PHY1 两层，其中 PHY1 层数据结构如下：

包头数据(16byte)	数据域(Nbyte)	包校验(2byte)

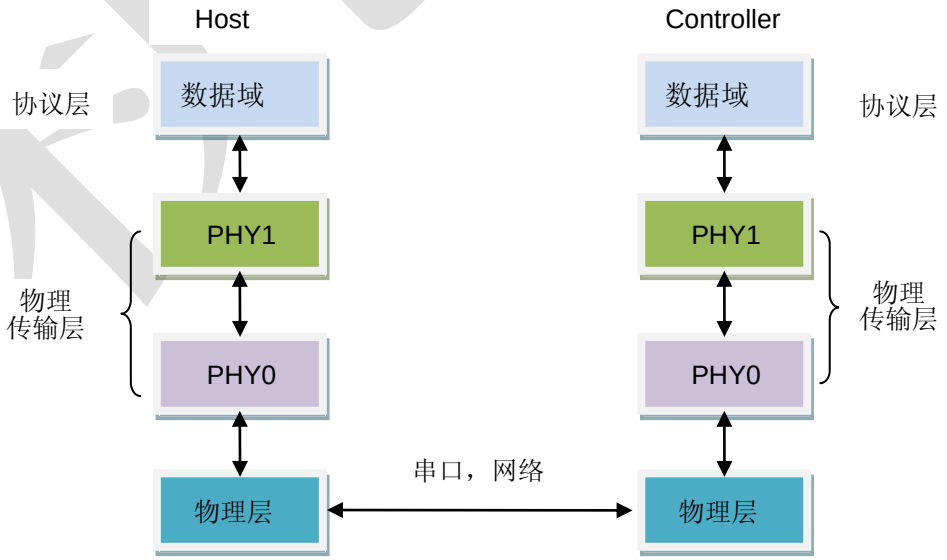
PHY1 层实现数据域的封包操作，它为数据域增加包头，并计算包数据的校验值。

PHY0 层数据结构为：

帧头(8byte)	PHY1 层数据(N + 18byte)	帧尾(1byte)

PHY0 层为 PHY1 层数据增加帧头和帧尾，并对 PHY1 层数据进行转义（参考字符转义）。

2.2 *数据流向示意图*



在发送端，协议层数据先提交到 PHY1 层，对数据域进行封包操作。然后 PHY1 层报数据提交到 PHY0 层，对 PHY1 层数据进行字符转义并增加帧头帧尾，最后数据经过物理底层发送出去。

在接收端，控制器将物理底层接收到的数据发送到 PHY0 层，PHY0 层去除帧头帧尾，并对数据进行反转义，然后将数据提交到 PHY1 层。PHY1 层将判断包数据的正确性，并去除包头和包校验值，向协议层提交有用数据。

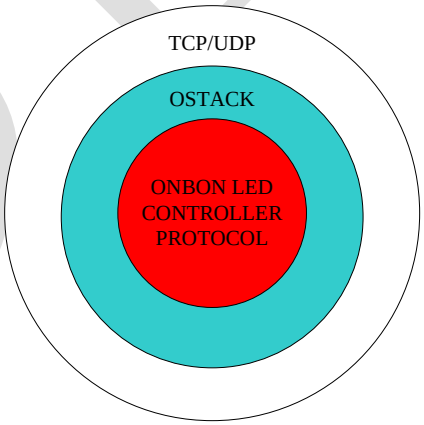
2.3 *字符转义*

- 封帧中遇到 0xA5，则将之转义为 0xA6，0x02；如遇到 0xA6，则将之转义为 0xA6，0x01 。
- 封帧中遇到 0x5A，则将之转义为 0x5B，0x02；如遇到 0x5B，则将之转义为 0x5B，0x01 。
- 解帧过程如果遇到连续两个字节为 0xA6, 0x02 ,则反转义为 0xA5 。
- 解帧过程如果遇到连续两个字节为 0xA6, 0x01 ,则反转义为 0xA6 。
- 解帧过程如果遇到连续两个字节为 0x5B, 0x02 ,则反转义为 0x5A 。
- 解帧过程如果遇到连续两个字节为 0x5B, 0x01 ,则反转义为 0x5B 。

注意：封帧过程中，所涉及校验的数据皆是转义之前的数据，所涉及的数据长度皆是转义之前的数据长度。

3. 网络通讯

网络通讯部分的通讯协议格式与串口通讯部分完全相同，不同之处仅在于，网络通讯最底层采用 UDP 或 TCP 协议。因此，此处只针对单用于以太网通讯的相关协议进行描述。

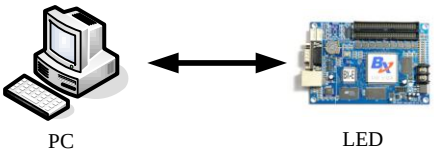


3.1 网络连接模式

为了方便用户的使用，我们将控制器与 PC 之间的连接，分为单机直连、通过以太网连接、通过 Internet 连接三种模式。

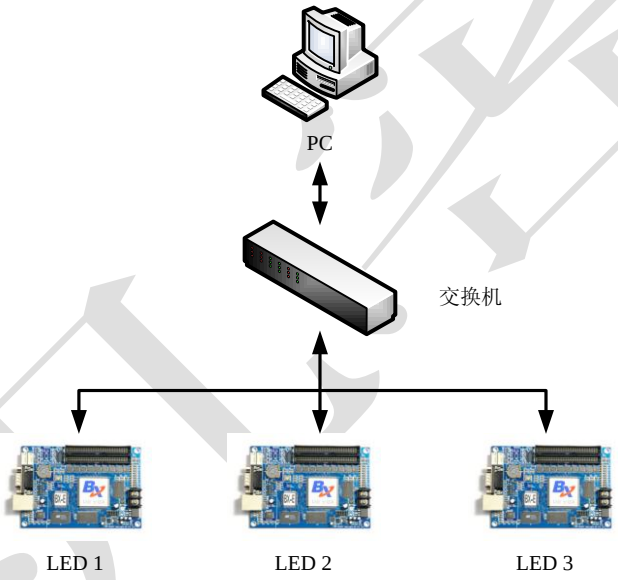
3.1.1 单机直连

单机直连是指，PC 和 LED 控制器之间不经过任何交换机或路由路而直接连接。在这种连接模式下，为了减少参数设置上的过多操作，PC 与控制器之间全部采用广播地址来进行通讯，而无须对控制器进行 IP 地址的设置。



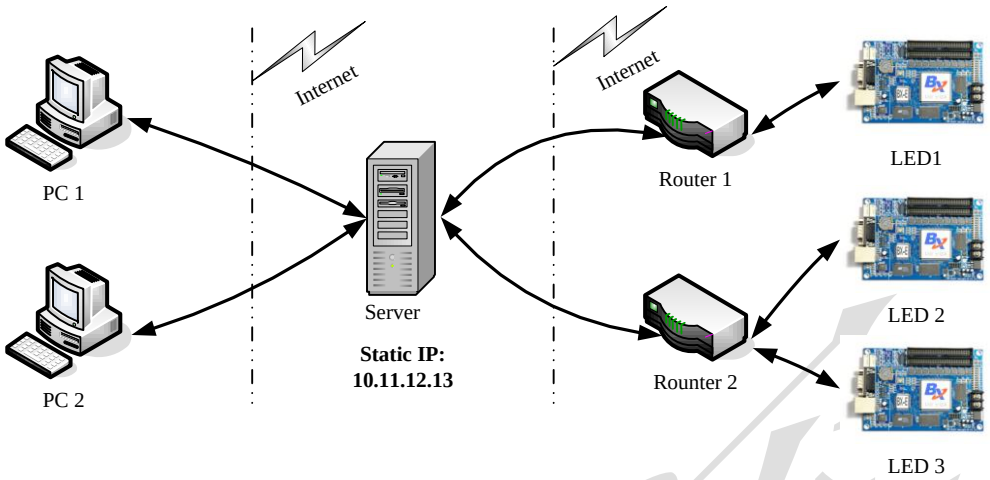
3.1.2 以太网连接

以太网连接，是指 PC 与控制器处于同一以太网内部。在这种连接模式下，控制器可以支持 DHCP 和手动设置 IP（静态 IP）两种模式。



3.1.3 跨Internet/Intranet连接

跨 Internet/Intranet 连接，是指 PC 软件和 LED 控制器之间通过 Internet/Intranet 进行连接。由于，在 Internet 上为每一个控制器分配一个固定的 IP 地址成本太高（基本上不可能实现）。因此，在此我们采用了一种基于 Server 的连接模式，具体连接模式如下：



- 架设一台具有固定 IP 的服务器（Server）
- 各控制器定时向服务器发送自身的 IP 地址和端口号
- PC 软件定时从服务器上下载各控制器的 IP 地址和端口号列表
- PC 软件通过 IP 与端口列表对 LED 控制器进行访问

注：在此种模式下，Server 和 PC 可以是同一台机器

3.2 TCP与UDP端口号

TCP 端口号，默认值为 5005，UDP 端口号为 5007。

4. 包头数据格式

Size (byte)	Name	Description
2	DstAddr	目标地址，即目标屏幕的屏号
2	SrcAddr	源地址 PC 客户端软件默认为 0x8000
1	ProtocolVer	动态区域协议版本号为 0xf0
1	Reserved	保留字
2	DeviceType	设备类型，用于区分网络中不同类型的设备，具体如下： BX_5E1 0x0154 BX_5E2 0x0254 BX_5E3 0x0354 BX_5Q0+ 0x1056 BX_5Q1+ 0x1156 BX_5Q2+ 0x1256
4	Reserved	保留字
4	DataLen	数据域的长度(仅针对数据域，不包括包头和包校验字节)

5. 数据域定义

5.1 请求与答复

信息(Message)可分为请求(Request)和答复(Response)两种，请求是指从 PC 软件到 LED 控制器的命令，答复是指从控制器到 PC 的回复。

所有的数据通讯必须由 PC 来发起。

5.1.1 Request信息格式

参数	数据长度	默认值	描述
RtnReq	1	0x01	是否要求返回状态 注：对于有些命令，此字段必须为 0x01，如：Ping 命令
CmdGroup	1	0x00	命令组
Cmd	1	0x00	命令编号
Data	N		发送的数据

5.1.2 Response信息格式

参数	数据长度	默认值	描述
Reserved	1	0x00	保留字
CmdGroup	1	0x00	命令组
Cmd	1	0x00	命令编号
Status	2		控制器状态寄存器
Error	2		错误状态寄存器
DataLen	2	N	数据长度，只包括 DATA，而不包括 Status 和 Error
Data	N		发送的数据

5.2 Status与Error

5.2.1 Status寄存器定义

位置	参数	是否可重置	描述
Bit0	NACK	N	0 –Request 被正常处理 1 –有错误存在
Bit2– Bit15	Reserved		

5.2.2 Error寄存器定义

Error Number	Name	Description
0	ERR_NO	No Error
1	ERR_OUTOFGROUP	命令组错误
2	ERR_NOCMD	此命令不存在
3	ERR_BUSY	控制器忙
4	ERR_MEMORYVOLUME	存储器容量越界
5	ERR_CHECKSUM	数据包 CRC 校验错误
6	ERR_FILENOTEXIST	此文件不存在

7	ERR_FLASH	Flash 访问错误
8	ERR_FILE_DOWNLOAD	文件下载错误
9	ERR_FILE_NAME	文件名错误
10	ERR_FILE_TYPE	文件类型错误
11	ERR_FILE_CRC16	文件校验错误
12	ERR_FONT_NOT_EXIST	字库文件不存在
13	ERR_FIRMWARE_TYPE	Firmware 与控制器类型不匹配
14	ERR_DATE_TIME_FORMAT	日期时间格式错误
15	ERR_FILE_EXIST	此文件已存在
16	ERR_FILE_BLOCK_NUM	文件 Block 号错误
17	ERR_CONTROLLER_TYPE	控制器类型不匹配
18	ERR_SCREEN_PARA	控制器参数越界或错误
19	ERR_CONTROLLER_ID	控制器 ID 错误

5.3 ACK与NACK

5.3.1 ACK答复

当一个请求信息被正常处理，没有发生任何错误，且不需要向 PC 回复任何附加内容时，需返回 ACK。其格式如下表所示：

参数	数据长度	默认值	描述
Reserved	1	0x00	保留字
CmdGroup	1	0xA0	命令组
Cmd	1	0x00	命令编号
Status	2	Bit0 = 0	控制器状态
Error	2		错误状态寄存器
DataLen	2	0x00	数据长度

5.3.2 NACK答复

参数	数据长度	默认值	描述
Reserved	1	0x00	保留字
CmdGroup	1	0xA0	命令组
Cmd	1	0x01	命令编号
Status	2	Bit0 = 1	控制器状态
Error	2		错误状态寄存器
DataLen	2	0x00	数据长度

5.4 动态区域相关命令

5.4.1 更新动态区域信息

参数	数据长度	默认值	描述
RtnReq	1	0x01	要求返回
CmdGroup	1	0XA7	命令组
Cmd	1	0x00	命令编号
Reserved	2	0x00	保留字

AreaDataLen	4		区域数据长度，即下面数据的总长度
AreaId	1	0x00	区域序号，从 0 开始
RunMode	1	0x00	动态区运行模式 0— 动态区数据循环显示。 1— 动态区数据显示完成后静止显示最后一页数据。 2— 动态区数据循环显示，超过设定时间后数据仍未更新时不再显示 3— 动态区数据循环显示，超过设定时间后数据仍未更新时显示 Logo 信息,Logo 信息即为动态区域的最后一页信息 4— 动态区数据顺序显示，显示完最后一页后就不再显示
Timeout	2		动态区数据超时时间，单位为秒
RelateAllPro	1		当该字节为 1 时，所有异步节目播放时都允许播放该动态区域；为 0 时，由接下来的规则来决定
RelateProNum	2	N	动态区域关联了多少个异步节目 一旦关联了某个异步节目，则当该异步节目播放时允许播放该动态区域，否则，不允许播放该动态区域 以下的节目编号根据 RelateProNum 的值来确定，当该值为 0 时不发送
RelateProSerial0	2		动态区域关联的第 0 个异步节目的编号
……			
RelateProSerialN-1	2		动态区域关联的第 N-1 个异步节目的编号
ImmePlay	1		是否立即播放 该字节为 0 时，该动态区域与异步节目一起播放 该字节为 1 时，异步节目停止播放，仅播放该动态区域 该字节为 2 时，暂存该动态区域，当播放完节目编号最高的异步节目后播放该动态区域 注意： 当该字节为 0 时，RelateAllPro 到 RelateProSerialN-1 的参数才有效，否则无效 当该参数为 1 或 2 时，由于不与异步节目同时播放，为控制该动态区域能及时结束，可选择 RunMode 参数为 2 或 4，当然也

			可通过删除该区域来实现
Reserved	4	0x00	保留字节
AreaType	1	0x10	区域类型
AreaX	2		区域左上角横坐标(Top Left)，单位 Pixel
AreaY	2		区域左上角纵坐标(Top Left)，单位 Pixel
AreaWidth	2		区域宽度，单位 Pixel
AreaHeight	2		区域高度，单位 Pixel
AreaFrame	N		区域边框属性
PageNum	2	0x0001 ~ 0xFFFF	数据页数，此参数不能为 0
以下为每个数据页的格式，根据页数循环			
PageDataLen	4		每页的数据长度
PageStyle	1	0x00	数据页类型
DisplayMode	1		显示方式 0x00 –随机显示 0x01 –静止显示 0x02 –快速打出 0x03 –向左移动 0x04 –向左连移 0x05 –向上移动 0x06 –向上连移 0x07 –闪烁 0x08 –飘雪 0x09 –冒泡 0x0a –中间移出 0x0b –左右移入 0x0c –左右交叉移入 0x0d –上下交叉移入 0x0e –画卷闭合 0x0f –画卷打开 0x10 –向左拉伸 0x11 –向右拉伸 0x12 –向上拉伸 0x13 –向下拉伸 0x14 –向左镭射 0x15 –向右镭射 0x16 –向上镭射 0x17 –向下镭射 0x18 –左右交叉拉幕 0x19 –上下交叉拉幕 0x1a –分散左拉 0x1b –水平百页 0x1c –垂直百页 0x1d –向左拉幕 0x1e –向右拉幕 0x1f –向上拉幕 0x20 –向下拉幕 0x21 –左右闭合

			0x22 –左右对开 0x23 –上下闭合 0x24 –上下对开 0x25 –向右移动 0x26 –向右连移 0x27 –向下移动 0x28 –向下连移
ClearMode	1		退出方式/清屏方式
Speed	1		速度等级
StayTime	2		停留时间，单位为 10ms
RepeatTime	1		重复次数
ValidLen	2		此字段只在左移、右移方式下有效，默认值为区域宽度。 注：此字段使用时一定要和区域宽度进行一下比较，如果此字段大于区域宽度，则使用区域宽度
Reserved	4		保留字
文本、图片数据格式			
PicPageData	N		点阵数据，参考点阵编码规则

5.4.2 删除动态区域区域信息

参数	数据长度	默认值	描述
RtnReq	1	0x01	要求返回
CmdGroup	1	0XA7	命令组
Cmd	1	0x01	命令编号
Reserved	2	0x00	保留字
DeleteAreaNum	1		要删除的区域个数 注意：如果该值为 0xFF，则删除所有动态区数据; 如果该值为 0x00，则不删除区域
DeleteAreaId	N		需要删除的区域 ID 号 如果要删除的区域个数为 0，则该项不发送

5.4.3 区域边框属性

参数	数据长度	默认值	描述
区域边框属性（Area Frame）			
AreaFFlag	1	0x00	区域边框标志位 注：如果此字段为 0x00，则以下区域边框属性不发送
AreaFDispStyle	1	0x00	边框显示方式： 0x00 –闪烁 0x01 –顺时针转动

			0x02 –逆时针转动 0x03 –闪烁加顺时针转动 0x04 –闪烁加逆时针转动 0x05 –红绿交替闪烁 0x06 –红绿交替转动 0x07 –静止打出
AreaFDispSpeed	1	0x01	边框显示速度
AreaFMoveStep	1	0x01	边框移动步长，单位为点，此参数范围为 1~16
AreaFWidth	1	0x01	边框组元宽度，此参数范围为 1~8 注：边框组元的长度固定为 16
AreaFBackup	2	0x00	备用字
AreaFUnitData	N		N = AreaFWidth

附录1 CRC16校验算法

For the calculation of the CRC-16 the following polynomial is used:
X^16 + X ^15 + X^2 + 1 = (x + 1) * (X^15 + x + 1).

For this polynomial efficient calculation via a table is possible. Below the algorithm is given in C:

```
#define CRC(crc,byte) (((crc) >> 8 ) ^ tabel[((crc) ^ (unsigned int) (byte)) & 0XFF])

unsigned short tabel[256] = {
  0X0000, 0XC0C1, 0XC181, 0X0140, 0XC301, 0X03C0, 0X0280, 0XC241,
  0XC601, 0X06C0, 0X0780, 0XC741, 0X0500, 0XC5C1, 0XC481, 0X0440,
  0XCC01, 0X0CC0, 0X0D80, 0XCD41, 0X0F00, 0XCFC1, 0XCE81, 0X0E40,
  0X0A00, 0XCAC1, 0XCB81, 0X0B40, 0XC901, 0X09C0, 0X0880, 0XC841,
  0XD801, 0X18C0, 0X1980, 0XD941, 0X1B00, 0XDBC1, 0XDA81, 0X1A40,
  0X1E00, 0XDEC1, 0XDF81, 0X1F40, 0XDD01, 0X1DC0, 0X1C80, 0XDC41,
  0X1400, 0XD4C1, 0XD581, 0X1540, 0XD701, 0X17C0, 0X1680, 0XD641,
  0XD201, 0X12C0, 0X1380, 0XD341, 0X1100, 0XD1C1, 0XD081, 0X1040,
  0XF001, 0X30C0, 0X3180, 0XF141, 0X3300, 0XF3C1, 0XF281, 0X3240,
  0X3600, 0XF6C1, 0XF781, 0X3740, 0XF501, 0X35C0, 0X3480, 0XF441,
  0X3C00, 0XFCC1, 0XFD81, 0X3D40, 0XFF01, 0X3FC0, 0X3E80, 0XFE41,
  0XFA01, 0X3AC0, 0X3B80, 0XFB41, 0X3900, 0XF9C1, 0XF881, 0X3840,
  0X2800, 0XE8C1, 0XE981, 0X2940, 0XEB01, 0X2BC0, 0X2A80, 0XEA41,
  0XEE01, 0X2EC0, 0X2F80, 0XEF41, 0X2D00, 0XEDC1, 0XEC81, 0X2C40,
  0XE401, 0X24C0, 0X2580, 0XE541, 0X2700, 0XE7C1, 0XE681, 0X2640,
  0X2200, 0XE2C1, 0XE381, 0X2340, 0XE101, 0X21C0, 0X2080, 0XE041,
  0XA001, 0X60C0, 0X6180, 0XA141, 0X6300, 0XA3C1, 0XA281, 0X6240,
  0X6600, 0XA6C1, 0XA781, 0X6740, 0XA501, 0X65C0, 0X6480, 0XA441,
  0X6C00, 0XACC1, 0XAD81, 0X6D40, 0XAF01, 0X6FC0, 0X6E80, 0XAE41,
  0XAA01, 0X6AC0, 0X6B80, 0XAB41, 0X6900, 0XA9C1, 0XA881, 0X6840,
  0X7800, 0XB8C1, 0XB981, 0X7940, 0XBB01, 0X7BC0, 0X7A80, 0XBA41,
  0XBE01, 0X7EC0, 0X7F80, 0XBF41, 0X7D00, 0XBDC1, 0XBC81, 0X7C40,
  0XB401, 0X74C0, 0X7580, 0XB541, 0X7700, 0XB7C1, 0XB681, 0X7640,
  0X7200, 0XB2C1, 0XB381, 0X7340, 0XB101, 0X71C0, 0X7080, 0XB041,
```

```
  0X5000, 0X90C1, 0X9181, 0X5140, 0X9301, 0X53C0, 0X5280, 0X9241,
  0X9601, 0X56C0, 0X5780, 0X9741, 0X5500, 0X95C1, 0X9481, 0X5440,
  0X9C01, 0X5CC0, 0X5D80, 0X9D41, 0X5F00, 0X9FC1, 0X9E81, 0X5E40,
  0X5A00, 0X9AC1, 0X9B81, 0X5B40, 0X9901, 0X59C0, 0X5880, 0X9841,
  0X8801, 0X48C0, 0X4980, 0X8941, 0X4B00, 0X8BC1, 0X8A81, 0X4A40,
  0X4E00, 0X8EC1, 0X8F81, 0X4F40, 0X8D01, 0X4DC0, 0X4C80, 0X8C41,
  0X4400, 0X84C1, 0X8581, 0X4540, 0X8701, 0X47C0, 0X4680, 0X8641,
  0X8201, 0X42C0, 0X4380, 0X8341, 0X4100, 0X81C1, 0X8081, 0X4040
};

unsigned short
CalcCRC(data, size)
char     *data;
int      size;
{
int i;
unsigned short crc = 0;

for (i = 0; i < size; i++) {
crc = CRC(crc, data[i]);
 }
return crc;
}
```

附录2 点阵编码规则

1. BX-5E 系列动态区编码规则

从区域的左上角开始，依次向右编码，到达区域最右端结束，然后再开始下一行的编码，依此循环。假定区域位于屏幕的左上角，编码时左上角的第一个红像素点（坐标为（0,0））对应第一个字节的 bit7,第二个红像素点坐标为（1,0）对应第一个字节的 bit6；第一个绿像素点（坐标为（0,0））对应第二个字节的 bit7,第二个绿像素点坐标为（1,0）对应第二个字节的 bit6； 以此类推只有在双色时才对绿像素进行编码，单色时不对绿像素进行编码，这样同样区域大小双色时的编码量为单色时的两倍。

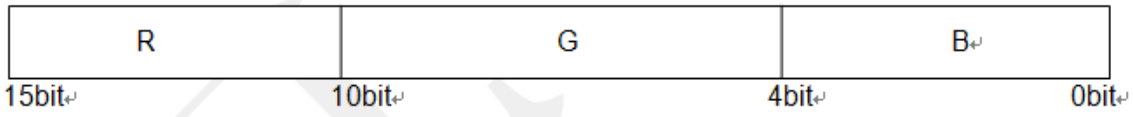
0 代表对应的像素点亮，1 代表对应的像素点灭

当区域的起始坐标不是 8 的整数倍时，编码时需要对该区域最左边像素所对应的字节靠高位以 1 进行填充，例如某区域左上角起始坐标为（35,0），则该区域第一个字节对应的二进制编码应该是 111*****。同理，当区域的结束坐标不是 8 的整数倍时，编码时需要对该区域最右边像素所对应的字节靠地位以 1 进行填充。

2. BX-5Q 系列动态区编码规则

BX-5Q 系列采用 RGB565 模式，即一个像素用两个字节共 16bit 表示。像素由左到右，由上到下依次编码。

RGB565 编码格式如下：



对应每种颜色取高位，低位舍弃。