## 下载文件数据编码格式

下载文件通讯交互过程如下：

1）客户端向LED发送下载请求

【客户端到LED】：FLASHDOWNLOAD˽设备ID˽下载类型˽类型参数<\r\n>

说明：设备ID必须与LED的ID一致，否则LED报告错误“设备ID不正确”

|  |  |  |  |
| --- | --- | --- | --- |
| 下载类型 | 类型说明 | 类型参数 | 说明 |
| UPGRADEBIN | 下载LED的主程序升级文件 | 固件版本号 | 例如1.0-01.130213为将要升级的程序的固件版本号；下载完成后，重启设备生效 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

下载类型信息表

2）LED接受下载请求

【LED到客户端】：FLASHDOWNLOAD˽READY<\r\n>

3）客户端向LED发送下载基本信息

【客户端到LED】：DOWNLOADINFO˽大包总数˽小包总数˽文件字节数˽整个文件的CRC校验值<\r\n>

例如：

【客户端到LED】：DOWNLOADINFO˽17˽536 ˽68524˽ABCD<\r\n>

其中17为本次发送下载文件的大包总数（总包数不能超过1000个，每包数据不能超过4096个字节），536为本次发送下载文件的小包总数，65532为下载的文件的总字节数（最大长度根据下载类型的不同而不同），CRC校验值为ABCD（见“CRC校验方式”）

4）LED接收信息成功

【LED到客户端】：DOWNLOADINFO˽OK<\r\n>

5）客户端依次将各个数据包发送给LED；只有在LED回复该序号的数据包后，客户  
 端才能够发送下一个序号的数据包

【客户端到LED】：DOWNLOADBIN˽大包序号˽小包序号˽包中下载文件长度˽本包下载文件内容CRC校验值˽下载文件内容<\r\n>

小包序号=32时：大包序号加1，并回复

【LED到客户端】：DOWNLOADBIN˽大包序号˽OK<\r\n>

大包序号从1开始。

注意：只有“下载文件内容”参数包含16进制的内容，其他参数均为ASCII编码

包中下载文件长度 最大固定为128个字节。

例如：

【客户端到LED】：DOWNLOADBIN ˽1˽5˽128˽ABCD˽(下载文件内容)<\r\n>

其中1为发送下载文件的大包序号(包序号从1开始)，5为发送下载文件的小包序号(包序号从1开始)，128为当前包的下载文件内容的长度（128为下载文件内容的最大长度）。下载文件内容格式包含十六进制数据和ASCII编码或者GB2312编码的格式，ABCD为该包下载文件内容的CRC校验（见“ CRC校验方式”）

【LED到客户端】：DOWNLOADBIN˽1˽OK<\r\n>

其中1为接收到的下载文件的大包序号

6）LED接收下载文件成功

【LED到客户端】：FLASHDOWNLOADBIN˽OK<\r\n>

最后一包不返回DOWNLOADBIN˽包序号˽OK<\r\n>，而是直接返回此包或者错误代码

如果在上述过程中LED发现了错误，将会向客户端返回错误代码，此时客户端应终止传输

【LED到客户端】：FLASHDOWNLOAD˽FAIL˽错误代码<\r\n>

当客户端发现错误时，客户端应该发送命令给LED取消本次下载：

【客户端到LED】：FLASHDOWNLOADCANCLE<\r\n>

【LED到客户端】：FLASHDOWNLOADCANCLE˽OK<\r\n>

错误代码见下表

|  |  |
| --- | --- |
| 错误代码 | 错误信息 |
| 01 | 设备ID不正确 |
| 02 | 下载文件超长 |
| 03 | 包序号错误 |
| 04 | CRC校验值错误 |
| 06 | 收到了非预期的命令 |
| 07 | 总包数太多 |
| 08 | 命令格式错误 |
| 09 | DOWNLOADBIN接收到了太多的数据 |
| 10 | DOWNLOADBIN命令标示长度与实际长度不符 |
| 11 | 写入FLASH失败 |
| 12 | 下载类型不正确 |
| 13 |  |
| 14 |  |
|  |  |

下载文件错误代码表

## CRC校验方式

本协议中采用的CRC字段为两个8位字节组成的一个16位值。CRC字段作为报文的一部分附加到报文上。将CRC附加到报文上的发送设备计算CRC值。在接收报文过程中，接收设备重新计算CRC值，并将计算值与CRC字段中接受的实际CRC值相比较。如果两个值不相等，则产生错误结果。

本协议采用的CRC多项式为X16+X12+X5+1，初始值为0xFFFF 。

INT16U AddCrc16(INT16U CRCValue,INT8U \*BUF,INT16U ByteL)

{

INT16U i,temp;

for(i=0;i<ByteL;i++){

temp=CRCValue>>8;

CRCValue<<=8;

CRCValue^=(crc\_ta[temp^BUF[i]]);

}

return(CRCValue);

}

// X16+X12+X5+1 余式表

const u16 crc\_ta[256]={

0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50a5, 0x60c6, 0x70e7,

0x8108, 0x9129, 0xa14a, 0xb16b, 0xc18c, 0xd1ad, 0xe1ce, 0xf1ef,

0x1231, 0x0210, 0x3273, 0x2252, 0x52b5, 0x4294, 0x72f7, 0x62d6,

0x9339, 0x8318, 0xb37b, 0xa35a, 0xd3bd, 0xc39c, 0xf3ff, 0xe3de,

0x2462, 0x3443, 0x0420, 0x1401, 0x64e6, 0x74c7, 0x44a4, 0x5485,

0xa56a, 0xb54b, 0x8528, 0x9509, 0xe5ee, 0xf5cf, 0xc5ac, 0xd58d,

0x3653, 0x2672, 0x1611, 0x0630, 0x76d7, 0x66f6, 0x5695, 0x46b4,

0xb75b, 0xa77a, 0x9719, 0x8738, 0xf7df, 0xe7fe, 0xd79d, 0xc7bc,

0x48c4, 0x58e5, 0x6886, 0x78a7, 0x0840, 0x1861, 0x2802, 0x3823,

0xc9cc, 0xd9ed, 0xe98e, 0xf9af, 0x8948, 0x9969, 0xa90a, 0xb92b,

0x5af5, 0x4ad4, 0x7ab7, 0x6a96, 0x1a71, 0x0a50, 0x3a33, 0x2a12,

0xdbfd, 0xcbdc, 0xfbbf, 0xeb9e, 0x9b79, 0x8b58, 0xbb3b, 0xab1a,

0x6ca6, 0x7c87, 0x4ce4, 0x5cc5, 0x2c22, 0x3c03, 0x0c60, 0x1c41,

0xedae, 0xfd8f, 0xcdec, 0xddcd, 0xad2a, 0xbd0b, 0x8d68, 0x9d49,

0x7e97, 0x6eb6, 0x5ed5, 0x4ef4, 0x3e13, 0x2e32, 0x1e51, 0x0e70,

0xff9f, 0xefbe, 0xdfdd, 0xcffc, 0xbf1b, 0xaf3a, 0x9f59, 0x8f78,

0x9188, 0x81a9, 0xb1ca, 0xa1eb, 0xd10c, 0xc12d, 0xf14e, 0xe16f,

0x1080, 0x00a1, 0x30c2, 0x20e3, 0x5004, 0x4025, 0x7046, 0x6067,

0x83b9, 0x9398, 0xa3fb, 0xb3da, 0xc33d, 0xd31c, 0xe37f, 0xf35e,

0x02b1, 0x1290, 0x22f3, 0x32d2, 0x4235, 0x5214, 0x6277, 0x7256,

0xb5ea, 0xa5cb, 0x95a8, 0x8589, 0xf56e, 0xe54f, 0xd52c, 0xc50d,

0x34e2, 0x24c3, 0x14a0, 0x0481, 0x7466, 0x6447, 0x5424, 0x4405,

0xa7db, 0xb7fa, 0x8799, 0x97b8, 0xe75f, 0xf77e, 0xc71d, 0xd73c,

0x26d3, 0x36f2, 0x0691, 0x16b0, 0x6657, 0x7676, 0x4615, 0x5634,

0xd94c, 0xc96d, 0xf90e, 0xe92f, 0x99c8, 0x89e9, 0xb98a, 0xa9ab,

0x5844, 0x4865, 0x7806, 0x6827, 0x18c0, 0x08e1, 0x3882, 0x28a3,

0xcb7d, 0xdb5c, 0xeb3f, 0xfb1e, 0x8bf9, 0x9bd8, 0xabbb, 0xbb9a,

0x4a75, 0x5a54, 0x6a37, 0x7a16, 0x0af1, 0x1ad0, 0x2ab3, 0x3a92,

0xfd2e, 0xed0f, 0xdd6c, 0xcd4d, 0xbdaa, 0xad8b, 0x9de8, 0x8dc9,

0x7c26, 0x6c07, 0x5c64, 0x4c45, 0x3ca2, 0x2c83, 0x1ce0, 0x0cc1,

0xef1f, 0xff3e, 0xcf5d, 0xdf7c, 0xaf9b, 0xbfba, 0x8fd9, 0x9ff8,

0x6e17, 0x7e36, 0x4e55, 0x5e74, 0x2e93, 0x3eb2, 0x0ed1, 0x1ef0

};

参考：<https://blog.csdn.net/szullc/article/details/84946673>

## 控制LED命令格式

【客户端到LED】：LEDDISPLAY˽包号˽**总包数**˽动作˽红光PWM值˽绿光PWM值˽蓝光PWM值˽琥珀光PWM值˽冷光PWM值˽冷光PWM值<\r\n>

【LED到客户端】：LEDDISPLAY˽ OK<\r\n>

如果在上述过程中LED发现了错误，将会向客户端返回错误代码

【LED到客户端】：LEDDISPLAY˽ERROR˽错误代码<\r\n>

发送命令格式为ASCII编码

|  |  |  |
| --- | --- | --- |
| 序号 | 字段信息 | 说明 |
| 1 | LEDDISPLAY | 命令关键字，固定部分 |
| 2 | 包号 | 固定为1 |
| 3 | 总包数 | 固定为1 |
| 4 | 动作 | 固定为1 |
| **5** | 红光PWM值 | **R\_PWM\_NUMS:0-1000** |
| **6** | 绿光PWM值 | **G\_PWM\_NUMS :0-1000** |
| 7 | 蓝光PWM值 | **B\_PWM\_NUMS :0-1000** |
|  | 琥珀光PWM值 | **A\_PWM\_NUMS: 0-1000** |
|  | 冷光PWM值 | **CW\_PWM\_NUMS :0-1000** |
|  | 暖光PWM值 | **WW\_PWM\_NUMS :0-1000** |

错误代码暂略

## 控制sensor命令格式

1.打开摄像头命令

【客户端到LED】：SENSORSTARTUP<\r\n>

【LED到客户端】：SENSORSTARTUP ˽ OK<\r\n>

2.打开摄像头命令

【客户端到LED】：SENSORSHUTUP <\r\n>

【LED到客户端】：SENSORSHUTUP ˽ OK<\r\n>