上海仰邦软件科技有限公司

Onbon G6 Protocol

2019-7-24

Copyright

All rights reserved. No part of this publication may be reproduced in any form by print, photo print, microfilm or any other means without written permission by ONBON.

© 2019-2022 Onbon

Version list:

Version	Date	Author	Comment, Description
1.0	2019-8-12	Css	1. 从 v19.07.04.00 移植过来的初始版本

机密

目录

目录 3
1. 概述 4
1.1 目的 4
1.2 参考资料 4
1.3 术语与缩略词 4
1.4 其它 4
2. 数据链路层定义 5
2.1 帧结构定义 5
2.2 数据转义 6
3. 信息模式及其分类 7
3.1 请求与答复 7
3.2 单包模式与分包模式 7
3.2.1 单包模式 7
3.2.2 分包模式 7
3.3 Request 信息格式 8
3.4 Response 信息格式 8
4. ACK 与 NACK 10
4.1 ACK 答复 10
4.2 NACK 答复 10
5. 控制器维护命令组 11
5.1 命令列表 错误!未定义书签。
5.2 节目锁定 错误!未定义书签。
附录 2 CRC16 校验算法 12

Onbon G6 Protocol V1.0

1. 概述

此文档主要定义了 ONBON LED 控制器的通讯协议，仰邦所有的通用控制器都必须遵循此协议进行开发。本协议 PHY 层采用 O'Stack，此协议为协议层的定义。

1.1 目的

此协议可用于基于 RS232/RS485/GPRS 的点对点单机通讯，多机通讯及广播通讯，也可扩展至支持 TCP/IP 以太网通讯。此协议具有灵活的扩展性，主要用于 LED 控制器的数据通讯。

1.2 参考资料

编号	文档名称	类型	日期
1	VPL Protocol V1.0		
2	Modbus Protocol		
3	O'Stack Protocol V1.0	Internal	
4	BX-C-II 通讯协议 V2.3	Internal	

1.3 术语与缩略词

缩写	说明
MSB	Most Significant Byte
LSB	Least Significant Byte
OFS	Onbon File System
O'Stack	Onbon Stack

1.4 其它

- 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
- 本文档中提及的“读取”和“写入”都是指 Host 主机对控制器的动作
- 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 download 是指从 PC 到控制器的操作，upload 是指从控制器到 PC
- 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，对于每一个 Bit，0 表示灭，1 表示亮
- 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

2. 数据链路层定义

2.1 帧结构定义

Size (byte)	Name	Description
2	DstAddr	目标地址，即控制器的屏号。 0xFFFF 为广播地址,PC 端软件发送这个地址表示,访问所有 LED 控制器，此时应用层软件不用回复应答信号给源端； 0xFFFE 为地址通配符，用于 ping 命令； 注： 0xffff 为广播地址，其可用于广播校时等命令。 注意：是否返回数据，由应用层协议来控制
2	SrcAddr	源地址，PC 客户端软件默认为 0x8000。 注：PC 客户端软件可用源地址范围为 0x8000~0xDFFF，用来代表不同客户端软件，0xE000~0xFFFE 为保留地址。
1	ProtocolVer	协议栈版本，默认值为 0x07。 注：三代控制器：0x00； 四代控制器：0x04； 五代控制器：0x05； 六代控制器：0x07； 0xF0~0xFF 为保留值。
1	ProtocolType	协议类型，无灰度系统为 0x00，带灰度系统为 0x01， 另 0xF0~0xFF 为保留值。
2	DeviceType	设备类型，用于区分网络中不同类型的设备，在协议栈的实现中。此项应为可配置项，并且可根据此项配置对数据包进行过滤。 其中低位字节表示设备系列，其具体定义如下： 0x66 – BX-6Q 系列 0xFE– 特殊类型 而高位字节表示设备编号，例如，BX-6Q2 应表示为[0x66, 0x02]，其它型号依此类推。 注：通配符为[0xfe, 0xff] 全彩控制器：BX-6QL/BQ-6Q0 型号的数据格式为 565，另外 6Q2 版本有 888 和 565 两种模式，其他版本均为 888 模式
1	Reserved	0x00
1	Reserved	备用字
1	Reserved	0x00

2	MsgSeq	消息序号，用于区分各条消息 其有效范围为 0x0000 – 0x7fff 中的偶数，其相对应的返回帧对应的消息序号为此序号 0x0000 – 0x7fff+0x8000
1	FrameLenMax	0x10
1	Reserved	0x96
1	Reserved	0x00
6	Reserved	此字段全填充为 0x00
2	DataLen	数据域的长度
2	DataChecksum	数据域校验，此较验采用 CRC16 算法，详细算法请参考附录
1	Reserved	0x00
1	HeaderChecksum	为整个 P1Header 除 HeaderChecksum 外的各字节异或

2.2 数据转义

封装过程：

a) PHY1 中遇到 0xA5，则将之转义为 0xA6，0x02，如遇到 0xA6，则将之转义为 0xA6，0x01

b) PHY1 中遇到 0x5A，则将之转义为 0x5B，0x02，如遇到 0x5B，则将之转义为 0x5B，0x01

解包过程：

a) 如果遇到连续两个字节为 0xA6, 0x02 ,则反转义为 0xA5

b) 如果遇到连续两个字节为 0xA6, 0x01 ,则反转义为 0xA6

对 0X5A 的解包同 0XA5

3. 信息模式及其分类

3.1 请求与答复

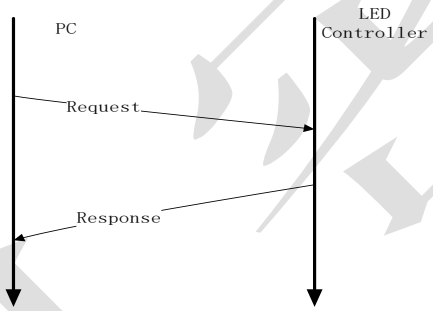
信息(Message)可分为请求(Request)和答复(Response)两种，请求是指从 PC 软件到 LED 控制器的命令，答复是指从控制器到 PC 的回复（是否答复由 Request 中的 RtnReq 字段决定）。所有的数据通讯必须由 PC 来发起。

3.2 单包模式与分包模式

根据信息包长不同，其发送模式可分为单包发送模式和分包发送模式。

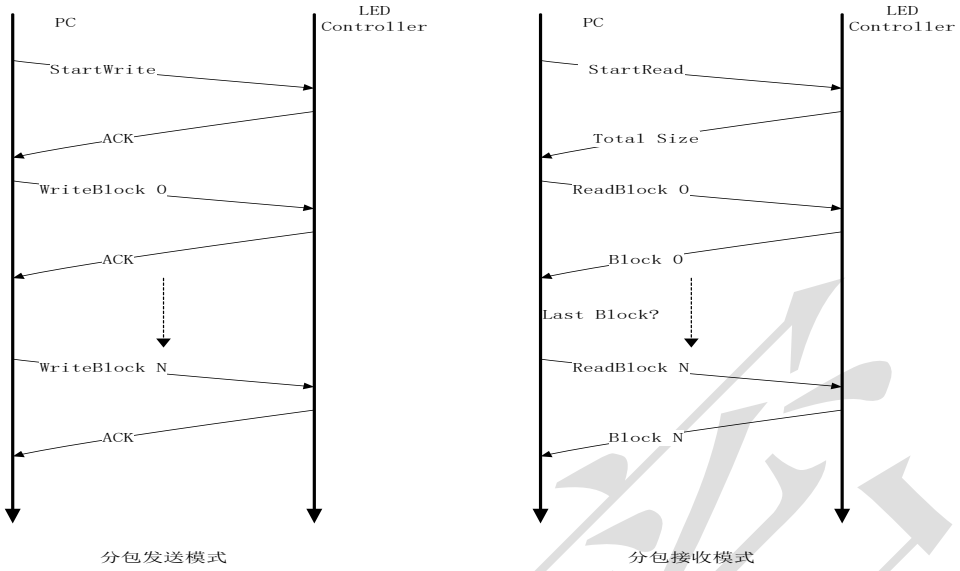
3.2.1 单包模式

单包模式是指请求和回复信息数据量都比较小，一条信息即可完成信息的发送或接收。所以单包模式是最简单的一种通讯模式。其通讯过程如下图所示：



3.2.2 分包模式

当请求或回复信息的数据量比较大时，由于数据链路层中数据帧长度的限制，单条信息无法完成数据的发送或接收，此时就需采用信息的分包模式。而根据发送和接收两种情况，又具体分为分包发送和分包接收两种模式。



其通讯流和如上图所示。

3.3 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 0x0，如 ping 命令，且 **4 代控制器无此字段**
CmdGroup	1	0x00	命令组
Cmd	1	0x00	命令编号
DataLen	2	N	数据长度
Data	N		发送的数据

3.4 Response信息格式

参数	数据长度	默认值	描述
Reserved	1	0x00	保留字　**注：4 代控制器无此字段**
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

4. ACK与NACK

4.1 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
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

4.2 NACK答复

参数	数据长度	默认值	描述
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. 0FS控制器维护命令组

5.1 校时命令（6th）

参数	数据长度	默认值	描述
CmdGroup	1	0xA2	命令组
Cmd	1	0x02	命令编号
DataLen	2	0x08	数据长度
Year	2		年（BCD 码）
Month	1		月（BCD 码）
Date	1		日（BCD 码）
Hour	1		时（BCD 码）
Minute	1		分（BCD 码）
Second	1		秒（BCD 码）
Week	1		星期，范围为 1~7，7 表示周日

返回值 为 ACK 或 NACK。

5.2 校时命令序列示例

示例 1、6 代卡（6M\6A）校时命令序列：

a5 a5 a5 a5 a5 a5 a5 a5 fe ff 00 80 07 00 fe ff 00 00 00 00 84 00 96 00 00 00 00 00 00 00 0d 00 29 9a 00 2b 01 a2 02 08 00 19 20 08 12 18 01 30 01 5a

其中，

1.蓝色字体 29 9a 为深蓝色字 01 到深蓝色字 01 的字经过 CRC16 位校验得到的值；

2.绿色字 2b 为红色字 fe 到红色字体 00 间的字经过按位异或得到的校验和；

3.数据域为 01 a2 02 08 00 19 20 08 12 18 01 30 01 ，校时时间为 2019-08-12 18：01：30，周一

示例 1、6M1 校时命令序列：

a5 a5 a5 a5 a5 a5 a5 a5 fe ff 00 80 07 00 62 01 00 00 00 00 84 00 96 00 00 00 00 00 00 00 0d 00 6C 55 00 C3 01 a2 02 08 00 19 20 08 12 18 30 00 01 5a

解释如示例一，校时时间：2019/8/12 18：30：00，周一

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;

}
```

机密