自定义通信协议（2018）

版本：18.0.0.1

最后修改时间：2018-04-14

修改记录

|  |  |  |
| --- | --- | --- |
| 版本 | 时间 | 修改内容 |
| 18.0.0.1 | 2018-04-14 | 初稿 |

# 1 总述

我想自己设定一个数据传输协议，使用自己制定的规则传输数据。

参考了《输电线路I1接口规约》2015版

# 2 报文格式分类

## 2.1 数据

## 2.2 控制及配置

## 2.3 文件

## 2.4工作状态

3 报文基本规范

## 3.1 数据通信方式

采用UDP协议。

## 3.2 数据传输方式

数据传输采用数据帧模式，传输序列为二进制字节流。

## 3.3 校验算法

校验方法为CRC16校验算法。

## 3.4 帧结构及帧数据排列格式

报文使用数据帧模式，对数据帧定义报文头、报文长度、设备ID、帧类型、报文类型、报文内容、校验位。以下所有值均为16进制数据格式。

### 3.4.1 帧结构

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 报文头 | 报文长度 | 设备ID | 帧类型 | 报文类型 | 帧序列号 | 报文内容 | 校验位 | 报文尾 |
| 1Byte | 2 Byte | 17Byte | 1Byte | 1Byte | 1Byte | 变长 | 2Byte | 1Byte |

各参数定义：

——报文头：标识报文，以十六进制7AB7（10进制31415）表示。

——报文长度：帧结构中“报文内容”的数据长度，单位：字节（Byte）。

——设备ID：给上传数据的设备分配的唯一标识。

——帧类型：按功能对数据帧进行区分、标识。

——报文类型：按不同数据类型对数据帧进行区分、标识。

——帧序列号：设备或上级设备主动发送的报文顺序流水号，以无符号整数表示，在确认或响应报文中应返回该序列号。

——报文内容：数据的字节长度不固定。

——校验位：数据通信领域中最常用的一种差错校验码，其特征是通过CRC16校验算法换算得出，校验的内容包括除去报文头、校验位、报文尾外所有报文数据（包括报文长度+设备ID+帧类型+报文类型+帧序列号+报文内容）。

——报文尾：标识报文数据结束的标识以16进制整数值96（10进制值150）表示。

### 3.4.3 帧数据排列格式

——除特殊说明，整形（占2Byte）、长整型（占4Byte）均采用低位在前高位在后方式存储：即字节由低B1 到高Bn 上下排列，字结位有高b7 到低 b0 左右排列，格式如下表所示：

|  |  |
| --- | --- |
| b7 b6 b5 b4 b3 b2 b1 b0 | B1 字节 |
| b7 b6 b5 b4 b3 b2 b1 b0 | B2 字节 |
| b7 b6 b5 b4 b3 b2 b1 b0 | B3 字节 |
| …… | …… |

以报文头为例：16进制7AB7占用两个字节，排列格式为：

|  |  |
| --- | --- |
| 1 0 1 1 0 1 1 1 | 字节1（B7） |
| 0 1 1 1 1 0 1 0 | 字节2（7A） |

以采集时间为例：16进制值01020304 占4个字节，排列格式为：

|  |  |
| --- | --- |
| 0 0 0 0 0 1 0 0 | 字节1（0x04） |
| 0 0 0 0 0 0 1 1 | 字节2（0x03） |
| 0 0 0 0 0 0 1 0 | 字节3（0x02） |
| 0 0 0 0 0 0 0 1 | 字节4（0x01） |

——控制数据报文中，IP地址以高位在前方式存储，如设备IP为192.168.1.1，排列格式为：

|  |  |
| --- | --- |
| 1 1 0 0 0 0 0 0 | 字节1（0xC0） |
| 1 0 1 0 1 0 0 0 | 字节2（0xA8） |
| 0 0 0 0 0 0 0 1 | 字节3（0x01） |
| 0 0 0 0 0 0 0 1 | 字节4（0x01） |

3.5 采集时间定义

表示方法：世纪秒法，使用32bit 长整型表示。世纪秒是指从1970年1月1日0时0分到指定时间过去的秒数，应用程序通过相应的处理函数实现世纪秒与实际的年月日时分秒的转换。

3.6 重发机制定义

被定义为数据的报文，如在4秒内没有收到相应数据报，或响应标明接收失败，则继续发送该数据报文，直至收到成功响应数据报。如发送5次不成功后则停止发送。

# 4 数据类报文格式

# 5 控制数据报文格式

# 6 文件数据报文格式

# 7 设备工作状态报文格式

## 7.1 心跳数据报

设备胡心跳数据报文格式如下表，默认心跳数据报上送周期为5分钟：

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 报文名称 | 长度（Byte） | 含义 |
| 1 | Sync | 2 | 报文头：B77A |
| 2 | Packet\_Length | 2 | 报文长度 |
| 3 | Device\_Id | 17 | 设备ID |
| 4 | Frame\_Type | 1 | 帧类型 |
| 5 | Packet\_Type | 1 | 报文类型 |
| 6 | Frame\_No | 1 | 帧序列号(无符号整数) |
| 7 | Clocktime\_Stamp | 4 | 当前时间 |
| 8 | Protocol\_Version | 4 | 通信协议版本(带小数) 四个部分，每一个字节代表一段。举例：版本号1.2.4.10的四个字节表示为：01 02 04 0A |
| 9 | CRC16 | 2 | 校验位 |
| 10 | End | 1 | 报文尾：0x96 |
| 11 |  |  |  |
| 12 |  |  |  |

响应方式胡数据报文格式如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 报文名称 | 长度(Byte) | 含义 |
| 1 | Sync | 2 | 报文头：B77A |
| 2 | Packet\_Length | 2 | 报文长度 |
| 3 | Device\_Id | 17 | 设备ID |
| 4 | Frame\_Type | 1 | 帧类型 |
| 5 | Packet\_Type | 1 | 报文类型 |
| 6 | Frame\_No | 1 | 帧序列号 |
| 7 | Command\_Status | 1 | 数据发送状态：0xFF 成功，0x00 失败 |
| 8 | Clocktime\_Stamp | 4 | 上位机时间（世纪秒，当值为0时，标明设备与上位机时间一致;当值非0时，设备需要进行校时） |
| 9 | CRC16 | 2 | 校验位 |
| 10 | End | 1 | 报文尾：0x96 |

8 CRC校验算法

为了提高运算效率，校验采用查表法：

// 高位字节表

static const unsigned char auchCRCHi[] = {

0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,

0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,

0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,

0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,

0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x01,

0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,

0x80,0x41,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,

0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,

0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x00,

0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,

0x80,0x41,0x00,0xC1,0x81, 0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,

0x41,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,

0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,

0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x00,0xC1,

0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,

0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,

0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,

0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,

0x81,0x40,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,

0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40};

// 低位字节表

static const unsigned char auchCRCLo[] = {

0x00,0xC0,0xC1,0x01,0xC3,0x03,0x02,0xC2,0xC6,0x06,0x07,0xC7,

0x05,0xC5,0xC4,0x04,0xCC,0x0C,0x0D,0xCD,0x0F,0xCF,0xCE,0x0E,0x0A,

0xCA,0xCB,0x0B,0xC9,0x09,0x08,0xC8,0xD8,0x18,0x19,0xD9,0x1B,0xDB,

0xDA,0x1A,0x1E,0xDE,0xDF,0x1F,0xDD,0x1D,0x1C,0xDC,0x14,0xD4,0xD5,

0x15,0xD7,0x17,0x16,0xD6,0xD2,0x12,0x13,0xD3,0x11,0xD1,0xD0,0x10,

0xF0,0x30,0x31,0xF1,0x33,0xF3,0xF2,0x32,0x36,0xF6,0xF7,0x37,0xF5,

0x35,0x34,0xF4,0x3C,0xFC,0xFD,0x3D,0xFF,0x3F,0x3E,0xFE,0xFA,0x3A,

0x3B,0xFB,0x39,0xF9,0xF8,0x38,0x28,0xE8,0xE9,0x29,0xEB,0x2B,0x2A,

0xEA,0xEE,0x2E,0x2F,0xEF,0x2D,0xED,0xEC,0x2C,0xE4,0x24,0x25,0xE5,

0x27,0xE7,0xE6,0x26,0x22,0xE2,0xE3,0x23,0xE1,0x21,0x20,0xE0,0xA0,

0x60,0x61,0xA1,0x63,0xA3,0xA2,0x62,0x66,0xA6,0xA7,0x67,0xA5,0x65,

0x64,0xA4,0x6C,0xAC,0xAD,0x6D,0xAF,0x6F,0x6E,0xAE,0xAA,0x6A,0x6B,

0xAB,0x69,0xA9,0xA8,0x68,0x78,0xB8,0xB9,0x79,0xBB,0x7B,0x7A,0xBA,

0xBE,0x7E,0x7F,0xBF,0x7D,0xBD,0xBC,0x7C,0xB4,0x74,0x75,0xB5,0x77,

0xB7,0xB6,0x76,0x72,0xB2,0xB3,0x73,0xB1,0x71,0x70,0xB0,0x50,0x90,

0x91,0x51,0x93,0x53,0x52,0x92,0x96,0x56,0x57,0x97,0x55,0x95,0x94,0x54,

0x9C,0x5C,0x5D,0x9D,0x5F,0x9F,0x9E,0x5E,0x5A,0x9A,0x9B,0x5B,0x99,

0x59,0x58,0x98,0x88,0x48,0x49,0x89,0x4B,0x8B,0x8A,0x4A,0x4E,0x8E,

0x8F,0x4F,0x8D,0x4D,0x4C,0x8C,0x44,0x84,0x85,0x45,0x87,0x47,0x46,

0x86,0x82,0x42,0x43,0x83,0x41,0x81,0x80,0x40

};

// 校验函数

nodebug unsigned short RTU\_CRC(unsigned char\*puchMsg, unsigned short usDataLen){

unsigned char uchCRCHi; // high byte of CRC initialized

unsigned char uchCRCLo; // low byte of CRC initialized

unsigned uIndex; // will index into CRC lookup table

uchCRCHi = 0xFF;

uchCRCLo = 0xFF;

while(usDataLen--){

// calculate the CRC

uIndex = uchCRCHi ^ (unsigned char)(\*puchMsg++);

uchCRCHi = uchCRCLo ^ auchCRCHi[uIndex];

uchCRCLo = auchCRCLo[uIndex];

}

return (uchCRCHi<<8|uchCRCLo);

}

# 9 附表

## 9.1 帧类型表

|  |  |  |
| --- | --- | --- |
| 序号 | 类型值 | 含义 |
| 1 | 0x01 | 数据报（设备→上位机） |
| 2 | 0x02 | 数据响应报（上位机→设备） |
| 3 | 0x03 | 控制数据报（上位机→设备） |
| 4 | 0x04 | 控制响应报（设备→上位机） |
| 5 | 0x05 | 文件控制报（上位机→设备） |
| 6 | 0x06 | 文件控制响应报（设备→上位机） |
| 7 | 0x07 | 工作状态报（设备→上位机） |
| 8 | 0x08 | 工作状态响应报（上位机→设备） |

## 9.2 报文类型表

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 报文分类 | 类型值 | 含义 |
| 1 | 数据报  (0x01~0x2F) | 0x01 |  |
| 2 | 0x02 |  |
| 3 | 0x03 |  |
| 4 | 控制数据报  (0x30~0x5F) | 0x30 |  |
| 5 | 0x31 |  |
| 6 | 0x32 |  |
| 7 | 文件报  (0x60~0x8F) | 0x60 |  |
| 8 | 0x61 |  |
| 9 | 0x62 |  |
| 10 | 工作状态报  (0x90~0xBF) | 0x90 | 心跳数据报 |
| 11 | 0x91 | 故障信息报 |
| 12 | 0x92 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |