Device控制器通信规范

|  |  |  |
| --- | --- | --- |
| **版本** | **修订内容** | **日期** |
| 1.0 |  | 2020/8/18 |
| 1.1 | 1. 增加指令的详细说明 2. 耗时指令直接返回忙碌状态 | 2020/8/18 |
| 1.2 | 1. 区分请求和应答 2. 只计算STX,ETX之间内容的校验和 | 2020/8/21 |
| 1.2.1 | 1. 防止校验和与分隔字符冲突 | 2020/8/26 |
| 1.2.2 | 1. Fix bug，30改为0x30 2. 增加一个回复内容 3. Features给出明确定义 | 2020/8/27 |
| 1.2.3 | 1. 废弃Features，降低复杂度 | 2020/11/3 |

Chiv

# 概述

举凡我司生产的各类可控制器件都应遵循本规范。本规范是一种协议层规范，规定主从设备之间的通信方式。可控类器件一般都由低端单片机实现，性能相对较弱，因此本规范的主要目的是提供一致性的通信协议，其它如加密、压缩等，都不在本规范的考虑范围内。

# 特性

规范的目的是制定统一的通信协议，降低设备维护难度，提高稳定性。鉴于此，本规范支持以下特性。

* 纯文本
* 主从式，强应答

规定，使用本规范时必须存在主机，只有主机能够发起通信。强应答，表示从机收到数据后必须给与应答。

* 支持特性查询

主机能够查询从机的状态，取得从机的特性。

# 规范

## 定义

在介绍规范之前，我们定义几个术语，以方便描述。

1. 指令

一个指令由以下ASCII控制字符和可打印字符构成。尖括号内为必须项，中括号内为可选项。

<**ENQ**><**STX**>[part]<**US**><action><**US**>[parameters,delim is **US**]<**ETX**>[chksum]<**EOT**>

chksum为校验和，校验和只针对STX和ETX之间的字符，包括US。校验和为8位，它由如下方式计算而得：

1. STX和ETX之间的所有字符异或，得到S0
2. S0，若S0的值小于0x30，则S0 = S0 + 0x30；S0二进制后若位1的个数为偶数，则S0 = S0 | (S0 + 1)

[part]：用于制定指令操作的对象，例如LED、ADC等

<action>：用于制定行为，例如，对于LED，可以是On，Off；对于彩色LED，可以是SetColor

[parameters]：参数，以**US**隔开，例如对于彩色LED，可以是0,0,0，Green等。

这些设计是类的抽象，例如，我们可以把原型

Led::Set(enum ColorName cr)；

Led::Set(const Color& rgb);

设计为如下指令

<**ENQ**><**STX**>Led6<**US**>Set<**US**>[it depends]<**ETX**>[chksum]<**EOT**>，直接通过颜色名字设置可以发送<**ENQ**><**STX**>Led6<**US**>Set<**US**>Green<**ETX**>[chksum]<**EOT**>，通过RGB设置可以发送

<**ENQ**><**STX**>Led6<**US**>Set<**US**>[it depends]<**ETX**>0,255,0<**EOT**>

在设计设备的指令时应当灵活应对，最佳的设计即等效设计，在从机中设计好某个设备的控制模块，然后将此模块按照以上的方式抽象成指令。例如：

<**ENQ**><**STX**>ColorLed<**US**>Set<**US**>6, Green<**ETX**>[chksum]<**EOT**>，表示将第6个彩色Led设置为绿色。

1. 指令包

指令包中包括一组指令。

<SOH>[指令]<EOB>

1. 应答

一条应答由以下ASCII控制字符和可打印字符构成。

<**ACK**><**STX**>[JSON]<**ETX**>[chksum]<**EOT**>，当没有JSON字符串时可以不传送<**STX**>、<**ETX**>，如：<**ACK**>[chksum]<**EOT**>，因为没有字符串，故而chksum的值为1。

校验和计算方式见“见上方”。应答的主体部分为一个JSON格式的字符串，UTF-8编码。若没有JSON，则可简单回复<**ACK**><**EOT**>，下文简称ACK。

## 强制指令

所谓强制指令，是指必须实现的指令；只有实现了必须实现指令的设备才能声称符合本规范，也只有符合本规范的设备才允许出厂。

### 设备ID请求

指令

<**ENQ**><**STX**>RequestID<**ETX**>[chksum]<**EOT**>

应答的JSON必须包含”Name"和”ID”键，举例如下：

<**ACK**><**STX**>

{

“Name”: “V8 Interface/Switch”,

“ID”: “QN9ST8U001”,

“SSVersion”: “1.2.1”

}

<**ETX**>[chksum]<**EOT**>

### 请求设备特性

WARNING：本节介绍的特征从1.2.3开始不再使用。

*指令*

*<****ENQ****><****STX****>RequestFeatures<****ETX****>[chksum]<****EOT****>*

*设备特性是指设备支持的特征，我们在ss中将这些特征简化，即用一个字符串表达一种特征，设备收到该字符串后就执行该特征对应的操作，字符串和对应操作都由设备生产商自行定义，以下是一个样例*

*样例：*

*[*

*“WDT”,*

*“LowPower”,*

*“Reset”*

*]*

*在收到这样的回复后，主控设备可以将命令的part部分用特征取代，如：*

*<ENQ><STX>Reset<ETX>[chksum><EOT>，若指令与其它部分有冲突，那么设备的默认实现都应该是优先执行Feature定义的命令。*

### 查询指令

指令

<**ENQ**><**STX**>RequestStatus<**ETX**>[chksum]<**EOT**>

应答

若从设备处于Standby状态，则返回ACK，否则返回JSON字串，并在Status键值中描述状态。

### 异常应答

对于任何异常（无论是错误的指令还是从机本身的异常），都返回一个JSON字串，并在键值Status中描述异常状态。如：

{

“Status”: “Unsupported”

}

### 耗时指令处理

若某条指令需要消耗较长时间，则从机应遵循如下方式执行：

1. 指令错误，返回异常应答；
2. 返回忙碌状态，
3. 进入忙碌状态，此时对所有的指令返回异常应答。

## 设备特有指令

设备特有指令不能与3.2节中规定的指令重复，除此之外，只需要遵循指令和应答的规范即可。