发送 02 就可以自动上传数据了 若要设置速率可以发送 02 00(01)(02)(03)
可以只发送 02 不设置采样速率,按仪表手动设置的多少上传
需要注意的是自动上传也不是永久上传,当 USB 断开后再连接需要重新发送 02 命令,当时间超过了 1 小时也会停止,为了保证持续上传,可以一小时内周期性的发送 02 或 01 命令维持数据不中断
发送 03 则停止自动上传数

| 包头 | |
|---|---|
| 1-4 字节 | |

为了格式简单化,请求的包头可以是 1-4 个字节

| type | zero | att | | obj | crc |
|---|---|---|---|---|---|
| 01 周期同步<br>02 ADC 自动上传<br>03 ADC 停止自动上传 | 保持 0 | 高 4 位 | 0:只上传 ADC<br>1:ADC 和 PD 协议数据<br>2: ADC 和 UFCS 协议数据<br>3: ADC 和 PD/UFCS 协议数据 | 5:<br>5*20 个字节 | 8 位 crc |
| | | 低 4 位 | 0:每秒 4 次<br>1:每秒 10 次<br>2:每秒 50 次<br>3:每秒 1 千次(数据长度变大)<br>4:每秒 1 万次(保留) | | |

接收的数据格式也简单
结构体为

| 包头 | 数据 1 | 数据 2 | 数据 3 | 数据 4 | 数据 n |
|---|---|---|---|---|---|
| 4 字节 | 20 字节 | 20 字节 | 20 字节 | 20 字节 | 20 字节 |

采集速率慢的时候只有数据 1,但是采集速度快了以后, 由于数据来不及上传则会保存在缓存区,等 USB 空闲了一次性发走,所以会出现有的数据包长度是变化的.

| type | zero | att | | obj | crc |
|---|---|---|---|---|---|
| 02 ADC 自动上传 | 保持 0 | 高 4 位 | 0:只上传 ADC<br>1:ADC 和 PD 协议数据<br>2: ADC 和 UFCS 协议数据<br>3: ADC 和 PD/UFCS 协议数据 | 5:<br>5*20 个字节 | 8 位 crc |
| | | 低 4 位 | 0:每秒 4 次<br>1:每秒 10 次<br>2:每秒 50 次<br>3:每秒 1 千次(数据长度变大)<br>4:每秒 1 万次(保留) | | |

```
struct
{
    uint32_t type       : 7;    /*!<                          */
    uint32_t zero       : 1;    /*!<  保持为 0                */
    uint32_t att        : 8;    /*!<  回复数据置 1            */
    uint32_t obj        : 8;    /*!<  数据包长度 4 字节大小    */
```

```
    uint32_t crc                : 8;        /*!< crc8                          */
}cdc_cmd_t; //包头
typedef struct
{
    uint32_t        Time;
    int32_t         Vbus; //uV
    int32_t         Ibus; //uA
    uint16_t        Vcc1;
    uint16_t        Vcc2;
    uint16_t        Vdp;
    uint16_t        Vdm;
}adc_queue_data_t;
```

例如收到

02 12 05 6A 20 2A 0B 00 68 F9 88 00 2F DF EF FF 79 02 72 01 0D 01 16 01

02 02 06 F2 CB 2D 00 00 40 1C 89 00 97 37 F7 FF 03 00 02 03 BC 0C 27 02

数据都是小端格式,即地位在前,把数据直接用结构体保存就能用

68 F9 88 00 = 0x0088F968 = 8976744 = 8.976744V

2F DF EF FF = 0xFFEFDF2F = -1056977 = -1.056977A 表示电流是从母座流向公头

79 02 = 0.633V

72 01 = 0.370V

0D 01 = 0.269V

16 01 = 0.278V