**概述**

与选手的PC机上的LabVIEW程序直接交互的，是电路板上的STM32单片机。STM32单片机会给LabVIEW发送**摆杆角度(ADC值)**和**电机转角(编码器)**的信息，LabVIEW处理后给STM32发送**电机转动力矩(PWM占空比)**。通信采用串口，波特率115200，8位数据、1位停止、无校验、无流控。

**STM32->LabVIEW**

**通讯协议**

STM32**每5ms**给LabVIEW发送一帧数据。通讯协议：

|  |  |
| --- | --- |
| Byte0 | 0x7A |
| Byte1 | (ADC高六位)+1 |
| Byte2 | (ADC低六位)+1 |
| Byte3 | (ENC高四位)+1 |
| Byte4 | (ENC中六位)+1 |
| Byte5 | (ENC低六位)+1 |
| Byte6 | 0xFF |

其中，**ADC范围是0~4095**，代表旋转电位器的角度值0~360°。当ADC到达4095后继续转动电位器，ADC会经历一短暂不确定期后，重新从0开始增加。摆杆处于倒立状态时的ADC值在2500~3500之间，需要自己测量。

**编码器(ENC)范围是0~65535**，电机每转一圈此值变化1040。初始值为10000，如果一直正向转动，达到65535后会从0开始增加，反之同理。

**帧格式举例**

假设某时刻**ADC值为3000，ENC值为12000**，则：

Byte1=(3000/64)+1=47=0x2E

Byte2=(3000%64)+1=57=0x39

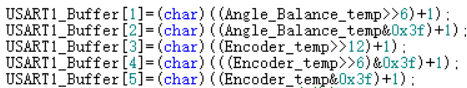
Byte3=(12000/4096)+1=3=0x03

Byte4=((12000/64)%64)+1=60=0x3C

Byte5=(12000%64)+1=33=0x21

所以这一帧数据为**0x7A 0x2E 0x39 0x03 0x3C 0x21**

附STM32原始代码：



**FAQ**

**Q:Byte,位,0x都是什么意思，看不懂**

A:Byte是字节，一个字节包含8位，一位代表一个0或者1，所以一个Byte有2^8=256种可能，可以代表0到255的数。因为ASCII编码，每个普通字符基本等价于1个Byte，所以将刚才算出来的Byte们理解为字符，就可以以字符串的形式发到串口。0x代表这个数是十六进制数，比如0xFF是十进制255。

**Q:为什么不发字符串”3000 12000”，现在这样好不直观**

A:显然地，发十进制字符串比这样要使用更多的Byte，字符串更长，从而占用更多的时间比如上述例子按字符串发送至少要用10~12个字符，所以仪器系统中只要不是要给人看，一般不用十进制字符串。

**Q:为什么要拆这么麻烦，高8位低8位不好吗**

A:我也不想，但这是LabVIEW特性导致的。VISA的串口接收函数接收的是字符串，而0x00是字符串的终止，会导致LabVIEW停止接收，所以不能允许Byte中出现0x00，这就有了+1。有了+1的话有效Byte种类只剩255个了，所以一个Byte不能发8位，只能发7位了，所以才要把ENC拆成466。至于帧尾0xFF，也是对VISA串口函数的妥协，这一点会在最后讲。

**LabVIEW->STM32**

**通讯协议**

LabVIEW每次给STM32发送数据，STM32都会在下一次发送传感器数据的同时响应。换句话说，假设LabVIEW控制算法运行时间在5ms内，那么控制的延迟就是5ms。

LabVIEW发送的数据需符合以下协议：

|  |  |
| --- | --- |
| Byte0 | 0x7A |
| Byte1 | PWM高八位 |
| Byte2 | PWM低八位 |

PWM范围是**0~14400**。7200代表静止，0代表全速反转，14400代表全速正转。

例如，想让电机停止，需要PWM=7200，而

Byte1=7200/256=28=0x1C

Byte2=7200%256=32=0x20

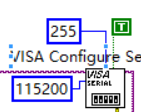
需要发送 **0x7A 0x28 0x32**

附原始代码：



**编程提示**

建议使用**帮助-查找范例-硬件输入与输出-串口-连续串口写入与读取**这个例程来改编自己的程序。

波特率设为115200，终止符设为255(0xFF)。如果采用默认终止符0x0A，可能会在数据中出现，造成LabVIEW把数据当成终止符，造成接收错误，而设为FF不会出现这种情况。



VISA读取产生字符串后，先转化为字节数组



用数组索引解码出每个Byte的数据

取数组索引后数据类型是u8，只能表示0~255的数，执行移位或乘法等操作时务必先转为u16（0~65536）类型，否则会溢出。同样地，算出PWM值准备发送时，也需要先将其转换为u16类型。



这是移位，右移n位就是整除2^n，左移n位就是乘上2^n，比乘法快

拆出PWM高低位后，先转u8再强制转换为字符类型



然后拼接为字符串，交给VISA