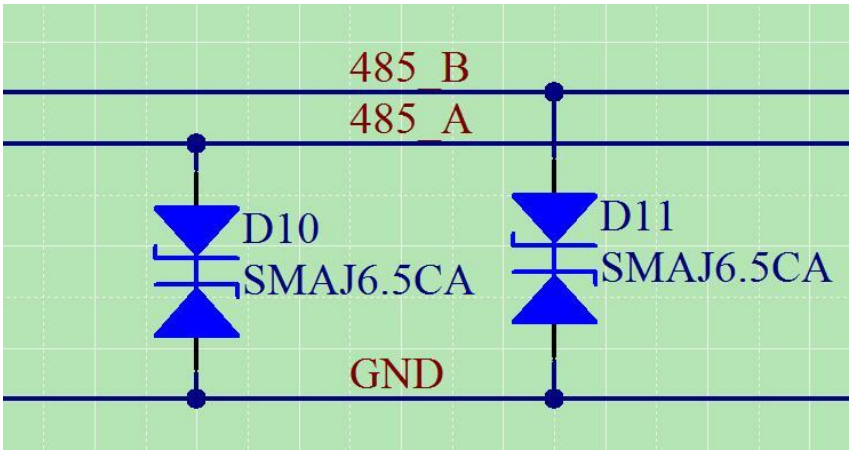
RS485 总线通信协议说明

本篇文档适用于 RMDS 系列 101、103、106、106+、302、303、401、402 等多个版本的 RoboModule 驱动器。其中，302 和 303 版本的驱动器，需要确保电路板背面跳线为 485A/485B 才可以使用 RS485 功能。

对于型号为：
101、103、106、106+、
302、303、
401、（303+和 402 的内部 485 电路，已经内含瞬态抑制二极管，可以直接跳过这一段）
由于内部 485 部分电路不含瞬态抑制二极管，在实际使用中，如遇人体静电或者碰触车架，或者线上共模电压超过 16V，则可能损坏。
因此，如需使用 485 总线，总线布线时候，最好走 3 根线，即：485A 485B GND，此处，即使主控与驱动器已经共地，也要额外的加一条 GND 的线，或者缠绕在 485A 485B 上，或者排线，或者屏蔽线皆可。
并且，
对 485A 与 GND 之间，增加一个 6.5V 的双向 TVS 二极管，
对 485B 与 GND 之间，增加一个 6.5V 的双向 TVS 二极管
此 TVS 二极管型号可以是：SMAJ6.5CA,SMBJ6.5CA 等等。



虽然此篇文档介绍 485 功能，但还是强烈推荐使用 CAN 总线来替代。

CAN 总线相比 485，无论哪方面，都是优势。

1. CAN 支持的通信速率可高达 1M；485 通信速率高了极不稳定。
2. CAN 数据包由硬件打包，不可能产生接收错位；485 数据包只能软件处理，产生错位可能性很大。
3. CAN 总线共模耐压高，为 58V，很难损坏；485 总线共模耐压只有 12V，容易坏。
4. CAN 总线为全双工，支持多主多从，硬件自动仲裁各设备的数据包优先级；485 是半双工，只能存在一个主机，其他设备都只能是从机，从机不能主动对外发数据。
5. CAN 的发送接收由硬件自动无缝切换，不会出错；485 需要软件干预切换，时序要求严格，否则一定出错。
6. CAN 数据包有硬件校验，出错率极低；485 无校验，走线好坏对出错率影响很大。
7. CAN 带有硬件过滤器，可以将不属于自己的数据包从硬件上过滤掉，不需要软件干预。对于多个驱动器一起使用的情况下，不会被别的驱动器的数据包打断并占用 CPU 资源。485 总线没有硬件过滤器。
8. CAN 可支持 120 个驱动器和多个主控。485 最多只能接 15 个驱动器，以及最多只能有 1 个主控。

本段介绍如何使用 RS485 总线的方式来操作本驱动器来控制电机的各种方式转动。

在 RS485 控制方式下需要连接的线有：

电源线（24V、GND）、

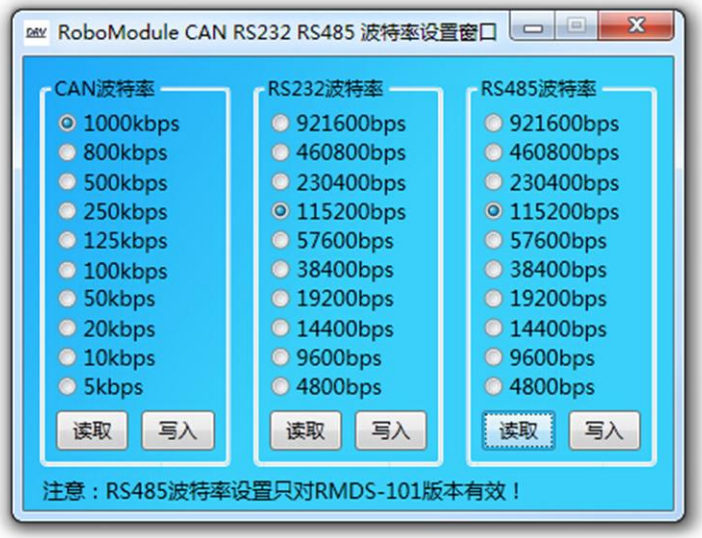
编码器（CHA、CHB、GND、+5V）、

电机（MT1、MT2）、

RS485 接口（485A、485B）。

对于本段协议，RS485 的串口波特率默认为 115200，数据位 8，停止位 1，无校验，无流控制。RS485 波特率支持在<RoboModule 直流伺服电机驱动器调试软件>上修改，支持如下 10 种波特率：921600、460800、230400、115200、57600、38400、19200、14400、9600、4800 等。

在<RoboModule 直流伺服电机驱动器调试软件>中对应的调试界面如下：



如上图所示是在<RoboModule 直流伺服电机驱动器调试软件上>对驱动器进行编号的界面。在 RS485 模式下，最多可以支持 15 个驱动器一起在同一条 RS485 总线上使用。

需要特别注意的，此界面的编组，只对 CAN 总线有效。对 RS485 有效的仅仅只有编号，01 号到 15 号。

另外：使用 RS485 总线来操作驱动器之前，必须先使用 RS232 串口线将驱动器连接至电脑，来进行参数调试，所涉及的调试内容有：

1. 调节电机和编码器的方向，确定电机转动的正方向，并使驱动器能够正常的进行调速。
2. 调节驱动器三个环路的 PID 参数，使驱动器最大程度的匹配所连接的电机和编码器。
3. 设置驱动器的编号。
4. 设置 RS485 的波特率。

在 RS485 通信协议下，

主控器对驱动器的操作命令有如下 12 种：

1. 让驱动器复位。
2. 让驱动器进入以下的 8 个运动模式的其中一个
3. 开环模式下，给驱动器发送数据指令
4. 电流模式下，给驱动器发送数据指令
5. 速度模式下，给驱动器发送数据指令
6. 位置模式下，给驱动器发送数据指令
7. 速度位置模式下，给驱动器发送数据指令
8. 电流速度模式下，给驱动器发送数据指令
9. 电流位置模式下，给驱动器发送数据指令
10. 电流速度位置模式下，给驱动器发送数据指令
11. 配置驱动器对外发送电流、速度、位置等数据的周期。
12. 在线检测指令。

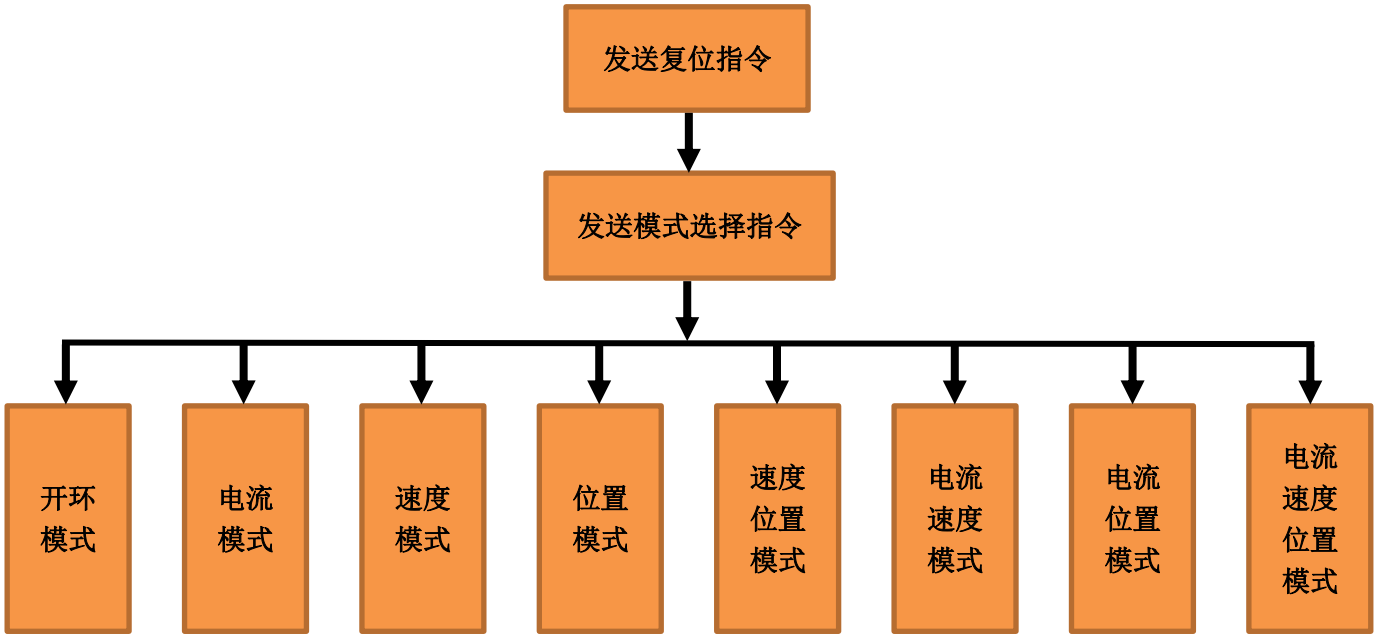
驱动器对外发送的指令只有如下 2 种：

1. 在进入 8 种运动模式后，对外发送当前电流、当前速度、当前位置值。
2. 在线检测指令的反馈。

PS:只有复位指令、模式选择指令、8 种运动模式下的数据指令等 10 种指令支持广播发送，其他指令都不支持广播发送。

PS:8 种运动模式是指：开环模式、电流模式、速度模式、位置模式、速度位置模式、电流速度模式、电流位置模式、电流速度位置模式

一、控制流程图：



使用 RS485 的通信方式来控制驱动器，控制流程如下：

1. 发送复位指令
2. 等待 500ms
3. 发送模式选择指令，使驱动器进入某种模式
4. 等待 500ms
5. 在已经进入的模式下发送数据指令。（周期性发送本条指令，间隔最短为 2ms，推荐间隔 10ms）

二、复位指令：（功能序号为 0）

本指令在任何状态下都会直接生效。
发送本指令后，驱动器会立即复位，即程序从头开始运行。

Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]	Data[8]	Data[9]
0x48	485_ID	0x55	0x55	0x55	0x55	0x55	0x55	0x55	0x55

正确发送完本指令后，驱动器上的蜂鸣器会 Bi 的一声，持续时间为 300ms。

驱动器的当前指令的 485_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编号	功能序号
0x	范围是 1-F	0

另外，本条指令的广播 485_ID = 0x00

举例说明：
02 号驱动器的当前指令的 485_ID 是 0x20
10 号驱动器的当前指令的 485_ID 是 0xA0

三、模式选择指令：（功能序号为 1）

本指令只在驱动器未进入任何模式的情况下生效,即复位之后。
如果驱动器已经进入某种模式，再发送此指令则会报错。
所以在发送本指令前，建议先发送复位指令，等待驱动器复位完成（大约 500ms），再发送本指令。

Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]	Data[8]	Data[9]
0x48	485_ID	待选择	0x55	0x55	0x55	0x55	0x55	0x55	0x55

驱动器的当前指令的 485_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编号	功能序号
0x	范围是 1-F	1

另外，本条指令的广播 485_ID = 0x01

举例说明：
02 号驱动器的当前指令的 485_ID 是 0x21
10 号驱动器的当前指令的 485_ID 是 0xA1

Data[2]与所选模式的对应值如下：

模式名称	Data[2]的值
开环模式	0x01
电流模式	0x02
速度模式	0x03
位置模式	0x04
速度位置模式	0x05
电流速度模式	0x06
电流位置模式	0x07
电流速度位置模式	0x08

正确发送完本条指令后，驱动器上蜂鸣器会 Bi 的一声，持续时间为 70ms，表示已经进去所选模式。

四、“开环模式”下的数据指令：（功能序号为 2）

本指令只有在驱动器进入“开环模式”之后才有效，其他任何状态下发送本指令都会让驱动器报错。
支持连续发送本指令来修改 PWM 的值，但连续发送的时间间隔不能小于 2 毫秒，建议以 10 毫秒为周期。

Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]	Data[8]	Data[9]
0x48	485_ID	待计算	待计算	0x55	0x55	0x55	0x55	0x55	0x55

驱动器的当前指令的 485_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编号	功能序号
0x	范围是 1-F	2

另外，本条指令的广播 485_ID = 0x02

举例说明：
02 号驱动器的当前指令的 485_ID 是 0x22
10 号驱动器的当前指令的 485_ID 是 0xA2

让 xx 号驱动器连接的电机在“PWM 模式”下，让它以 temp_pwm 的占空比转动：
则 Data[2] = (unsigned char)((temp_pwm>>8)&0xff);
Data[3] = (unsigned char)(temp_pwm&0xff);
其中
temp_pwm 的取值范围为：-5000~+5000。

五、“电流模式”下的数据指令：（功能序号为 3）

本指令只在驱动器进入“电流模式”之后才有效，其他任何状态下发送本指令都会让驱动器报错。
支持连续发送本指令来修改 PWM 的限制值和目标电流值，但连续发送的时间间隔不能小于 2 毫秒，建议以 10 毫秒为周期。

Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]	Data[8]	Data[9]
0x48	485_ID	待计算	待计算	待计算	待计算	0x55	0x55	0x55	0x55

驱动器的当前指令的 485_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编号	功能序号
0x	范围是 1-F	3

另外，本条指令的广播 485_ID = 0x03

举例说明：
02 号驱动器的当前指令的 485_ID 是 0x23
10 号驱动器的当前指令的 485_ID 是 0xA3

让 xx 号驱动器连接的电机在“PWM 电流模式”下，设置 temp_pwm 的限制占空比，输出 temp_current 的电流：
则 Data[2] = (unsigned char)((temp_pwm>>8)&0xff);
Data[3] = (unsigned char)(temp_pwm&0xff);
Data[4] = (unsigned char)((temp_current>>8)&0xff);
Data[5] = (unsigned char)(temp_current&0xff);
其中
temp_pwm 的取值范围为：0~+5000。
temp_current 的取值范围是：-32768~+32767。（16 位整型数取值范围，单位是 mA）

六、“速度模式”下的数据指令：（功能序号为 4）

本指令只在驱动器进入“速度模式”之后才有效，其他任何状态下发送本指令都会让驱动器报错。
支持连续发送本指令来修改 PWM 的限制值和给定的速度值，但连续发送的时间间隔不能小于 2 毫秒，建议以 10 毫秒为周期。

Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]	Data[8]	Data[9]
0x48	485_ID	待计算	待计算	待计算	待计算	0x55	0x55	0x55	0x55

驱动器的当前指令的 485_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编号	功能序号
0x	范围是 1-F	4

另外，本条指令的广播 485_ID = 0x04

举例说明：
02 号驱动器的当前指令的 485_ID 是 0x24
10 号驱动器的当前指令的 485_ID 是 0xA4

让 xx 号驱动器连接的电机以 temp_pwm 的限制 PWM，以 temp_velocity 的速度转动：

```
则  Data[2] = (unsigned char)((temp_pwm>>8)&0xff);
    Data[3] = (unsigned char)(temp_pwm&0xff);
    Data[4] = (unsigned char)((temp_velocity>>8)&0xff);
    Data[5] = (unsigned char)(temp_velocity&0xff);
```

其中
temp_pwm 的取值范围为 0~+5000。
temp_velocity 的取值范围为：-32768~+32767。（16 位整型数取值范围，单位是 RPM）

七、“位置模式”下的参数指令：（功能序号为 5）

本指令只在驱动器进入“位置模式”之后才有效，其他任何状态下发送本指令都会让驱动器报错。
支持连续发送本指令来修改 PWM 的限制值和给定的目标位置值，但连续发送的时间间隔不能小于 2 毫秒，建议以 10 毫秒为周期。

Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]	Data[8]	Data[9]
0x48	485_ID	待计算	待计算	0x55	0x55	待计算	待计算	待计算	待计算

驱动器的当前指令的 485_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编号	功能序号
0x	范围是 1-F	5

另外，本条指令的广播 485_ID = 0x05

举例说明：
02 号驱动器的当前指令的 485_ID 是 0x25
10 号驱动器的当前指令的 485_ID 是 0xA5

让 xx 号驱动器连接的电机以 temp_pwm 的限制电流，转动到 temp_position 的位置：

```
则  Data[2] = (unsigned char)((temp_pwm>>8)&0xff);
    Data[3] = (unsigned char)((temp_pwm)&0xff);
    Data[4] = 0x55;
    Data[5] = 0x55;
    Data[6] = (unsigned char)((temp_position>>24)&0xff);
    Data[7] = (unsigned char)((temp_position>>16)&0xff);
    Data[8] = (unsigned char)((temp_position>>8)&0xff);
    Data[9] = (unsigned char)(temp_position&0xff);
```

其中
temp_pwm 的取值范围为：0~+5000。
temp_position 的取值范围为：-2147483648~+2147483647。（32 位有符号整型数取值范围，单位 qc）

八、“速度位置模式”下的参数指令：（功能序号为 6）

本指令只在驱动器进入“速度位置模式”之后才有效，其他任何状态下发送本指令都会让驱动器报错。
支持连续发送本指令来修改 PWM 的限制值，运行速度值和给定的目标位置值，但连续发送的时间间隔不能小于 2 毫秒，建议以 10 毫秒为周期。

Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]	Data[8]	Data[9]
0x48	485_ID	待计算	待计算	待计算	待计算	待计算	待计算	待计算	待计算

驱动器的当前指令的 485_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编号	功能序号
0x	范围是 1-F	6

另外，本条指令的广播 485_ID = 0x06

举例说明：
02 号驱动器的当前指令的 485_ID 是 0x26
10 号驱动器的当前指令的 485_ID 是 0xA6

让 xx 号驱动器连接的电机以 temp_pwm 的限制 PWM，temp_velocity 的限制速度，转动到 temp_position 的位置：

```
则  Data[2] = (unsigned char)((temp_pwm>>8)&0xff);
    Data[3] = (unsigned char)((temp_pwm)&0xff);
    Data[4] = (unsigned char)((temp_velocity>>8)&0xff);
    Data[5] = (unsigned char)(temp_velocity&0xff);
    Data[6] = (unsigned char)((temp_position>>24)&0xff);
    Data[7] = (unsigned char)((temp_position>>16)&0xff);
    Data[8] = (unsigned char)((temp_position>>8)&0xff);
    Data[9] = (unsigned char)(temp_position&0xff);
```

其中
temp_pwm 的取值范围为：0~+5000。
temp_velocity 的取值范围为：0~+32767。（16 位有符号整型数的正数范围，单位 RPM）
temp_position 的取值范围为：-2147483648~+2147483647。（32 位有符号整型数的范围，单位 qc）

九、“电流速度模式”下的数据指令：（功能序号为 7）

本指令只在驱动器进入“电流速度模式”之后才有效，其他任何状态下发送本指令都会让驱动器报错。
支持连续发送本指令来修改电流的限制值和给定的速度值，但连续发送的时间间隔不能小于 2 毫秒，建议以 10 毫秒为周期。

Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]	Data[8]	Data[9]
0x48	485_ID	待计算	待计算	待计算	待计算	0x55	0x55	0x55	0x55

驱动器的当前指令的 485_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编号	功能序号
0x	范围是 1-F	7

另外，本条指令的广播 485_ID = 0x07

举例说明：
02 号驱动器的当前指令的 485_ID 是 0x27
10 号驱动器的当前指令的 485_ID 是 0xA7

让 xx 号驱动器连接的电机以 temp_current 的限制电流，以 temp_velocity 的速度转动：

```
则  Data[2] = (unsigned char)((temp_current>>8)&0xff);
    Data[3] = (unsigned char)(temp_current&0xff);
    Data[4] = (unsigned char)((temp_velocity>>8)&0xff);
    Data[5] = (unsigned char)(temp_velocity&0xff);
```

其中
temp_current 的取值范围为：0~+32767。（16 位有符号整型数的正数范围，单位 mA）
temp_velocity 的取值范围为：-32768~+32767。（16 位有符号整型数的范围，单位 RPM）

十、“电流位置模式”下的参数指令：（功能序号为 8）

本指令只在驱动器进入“电流位置模式”之后才有效，其他任何状态下发送本指令都会让驱动器报错。
支持连续发送本指令来修改 PWM 的限制值，运行速度值和给定的目标位置值，但连续发送的时间间隔不能小于 2 毫秒，建议以 10 毫秒为周期。

Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]	Data[8]	Data[9]
0x48	485_ID	待计算	待计算	0x55	0x55	待计算	待计算	待计算	待计算

驱动器的当前指令的 485_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编号	功能序号
0x	范围是 1-F	8

另外，本条指令的广播 485_ID = 0x08

举例说明：
02 号驱动器的当前指令的 485_ID 是 0x28
10 号驱动器的当前指令的 485_ID 是 0xA8

举例：
让 xx 号驱动器连接的电机以 temp_current 的限制电流，转动到 temp_position 的位置：

```
则  Data[2] = (unsigned char)((temp_current>>8)&0xff);
    Data[3] = (unsigned char)((temp_current)&0xff);
    Data[4] = 0x55;
    Data[5] = 0x55;
    Data[6] = (unsigned char)((temp_position>>24)&0xff);
    Data[7] = (unsigned char)((temp_position>>16)&0xff);
    Data[8] = (unsigned char)((temp_position>>8)&0xff);
    Data[9] = (unsigned char)(temp_position&0xff);
```

其中
temp_current 的取值范围为：0~+32767（16 位有符号整型数的正数范围，单位 mA）
temp_position 的取值范围为：-2147483648~+2147483647。（32 位有符号整型数的范围，单位 qc）

十一、“电流速度位置模式”下的参数指令：（功能序号为 9）

本指令只在驱动器进入“电流速度位置模式”之后才有效，其他任何状态下发送本指令都会让驱动器报错。支持连续发送本指令来修改 PWM 的限制值，运行速度值和给定的目标位置值，但连续发送的时间间隔不能小于 2 毫秒，建议以 10 毫秒为周期。

Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]	Data[8]	Data[9]
0x48	485_ID	待计算	待计算	待计算	待计算	待计算	待计算	待计算	待计算

驱动器的当前指令的 485_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编号	功能序号
0x	范围是 1-F	9

另外，本条指令的广播 485_ID = 0x09

举例说明：
02 号驱动器的当前指令的 485_ID 是 0x29
10 号驱动器的当前指令的 485_ID 是 0xA9

让 xx 号驱动器连接的电机以 temp_current 的限制电流，temp_velocity 的限制速度，转动到 temp_position 的位置：

```
则  Data[2] = (unsigned char)((temp_current>>8)&0xff);
    Data[3] = (unsigned char)((temp_current)&0xff);
    Data[4] = (unsigned char)((temp_velocity>>8)&0xff);
    Data[5] = (unsigned char)(temp_velocity&0xff);
    Data[6] = (unsigned char)((temp_position>>24)&0xff);
    Data[7] = (unsigned char)((temp_position>>16)&0xff);
    Data[8] = (unsigned char)((temp_position>>8)&0xff);
    Data[9] = (unsigned char)(temp_position&0xff);
```

其中
temp_current 的取值范围为：0~+32767。（16 位有符号整型数的正数范围，单位 mA）
temp_velocity 的取值范围为：0~+32767。（16 位有符号整型数的正数范围，单位 RPM）
temp_position 的取值范围为：-2147483648~+2147483647。（32 位有符号整型数的范围，单位 qc）

十二、运行参数获取指令：（功能序号为 10）

运行参数获取指令，可以让主控得知 485 总线上挂接的驱动器当前的电流、速度、位置等信息。

在 CAN 总线的操作模式下，所有的连接设备都可以随时对外发送消息，也就是说，CAN 总线是一个多主多从的总线结构。在任意的时间，任何一个挂接的设备都可以对外发送消息，而总线的仲裁由 CAN 总线的硬件自动完成，所以不会造成总线冲突。

而 485 总线是一主多从的总线结构，所以，一条 485 总线上只能有一个主机，其他挂接的设备都只能是从机，从机不能主动的对外发送消息。也就是说，主控器控制着整个 485 总线的运行，从机只能被动的对外发送消息。一句话说，就是，主机叫你发，你就发，主机没让你发，你就不能发。

主控器想要获取 1 号驱动器的电流、速度、位置信息，要发一条特定指令给 1 号驱动器，然后 1 号驱动器才对外发送电流、速度、位置信息。这个特定指令是单次生效的，如果主控器在某下一个时间还要获取电流、速度、位置信息，还要进行重复操作。而在等待 1 号驱动器返回电流、速度、位置信息的过程中，主控需要安静的等待数据的到来，不能再发送消息来占用 485 总线，否则驱动器发送出来的消息就错乱了。

主控器要获取某个驱动器的电流、速度、位置信息的指令如下：

Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]	Data[8]	Data[9]
0x48	485_ID	0x01	0x55	0x55	0x55	0x55	0x55	0x55	0x55

驱动器的当前指令的 485_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编号	功能序号
0x	范围是 1-F	A

另外，本条指令没有广播指令。

举例说明：
02 号驱动器的当前指令的 485_ID 是 0x2A
10 号驱动器的当前指令的 485_ID 是 0xAA

发送完本条指令，主控器需要停止对外发送消息，直到指定的驱动器对外发送完毕电流、速度、位置信息后才能继续去霸占 485 总线，否则有可能造成总线冲突。

当然，此过程中，超时判断的程序必不可少，否则某个节点一旦发生故障，则整条 485 总线都会陷入瘫痪。

十三、数据反馈：（功能序号为 11）

驱动器收到主机的参数获取指令后，对外发送一次当前的电流、速度、位置信息。格式如下：

Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]	Data[8]	Data[9]
0x48	485_ID	待接收	待接收	待接收	待接收	待接收	待接收	待接收	待接收

驱动器的当前指令的 485_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编号	功能序号
0x	范围是 1-F	B

另外，本条指令没有广播指令。

举例说明：
02 号驱动器的当前指令的 485_ID 是 0x2B
10 号驱动器的当前指令的 485_ID 是 0xAB

举例：
02 号驱动器当前的电流值是 real_current，当前的速度值是 real_velocity，当前的位置是 real_position，在收到主控器的获取参数指令后，2 号驱动器则会对外发送如下的 485 消息：

```
Data[0] = 0x48;
Data[1] = 0x2B;
Data[2] = (unsigned char)((real_current>>8)&0xff);
Data[3] = (unsigned char)(real_current&0xff);
Data[4] = (unsigned char)((real_velocity>>8)&0xff);
Data[5] = (unsigned char)(real_velocity&0xff);
Data[6] = (unsigned char)((real_position>>24)&0xff);
Data[7] = (unsigned char)((real_position>>16)&0xff);
Data[8] = (unsigned char)((real_position>>8)&0xff);
Data[9] = (unsigned char)(real_position&0xff);
```

对于主控而言，还原电流、速度、位置的反馈值，可以如下：

```
int16_t real_current  = (Data[2]<<8)|Data[3];
int16_t real_velocity = (Data[4]<<8)|Data[5];
int32_t real_position = (Data[6]<<24)|(Data[7]<<16)|(Data[8]<<8)|Data[9];
```

用户可以利用此项功能进行检测驱动器工作状态，例如如下所述：
1. 可以利用电流反馈值来监测电机母线电流的值，以此可以在主控上设计一个长时堵转保护功能。
2. 可以利用速度反馈，来分析带负载情况下速度的变化曲线。
3. 可以利用位置反馈，来检测位置环的执行程度，监测位置是否到位，以便设计一个时间紧凑的执行流程。

十四、在线检测：（功能序号为 15）

本指令在任何状态下都会直接生效。
成功发送本指令后，驱动器会返回一条一模一样的指令，证明自己在线。

Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]	Data[8]	Data[9]
0x48	485_ID	0x55	0x55	0x55	0x55	0x55	0x55	0x55	0x55

驱动器的当前指令的 485_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编号	功能序号
0x	范围是 1-F	F

本条指令不支持广播。

举例说明：
02 号驱动器的当前指令的 485_ID 是 0x2F
10 号驱动器的当前指令的 485_ID 是 0xAF