CAN 通信协议

本篇文档适用于 RMDS 全系列带 CAN 总线的驱动器（105/105+不带 CAN 总线）。

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

使用 CAN 通信时候，同一条 CAN 总线上挂接最多 120 个 RMDS 驱动器。采用分组的形式进行编号，一共分成 8 组（第 0 组到第 7 组），每组有 15 个成员（01 号到 15 号）。

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:

可以设置 CAN 总线的波特率,CAN 的波特率支持以下数值:1000kbps、800kbps、500kbps、250kbps、125kbps、100kbps、50kbps、20kbps、10kbps、5kbps 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:

出厂默认的 CAN 的波特率为 1000kbps。如需改变，请直接在<RoboModule 直流伺服电机驱动器调试软件>

上操作修改即可。

另外，RMDS 系列驱动器的所有 CAN 消息都是数据帧、标准帧、帧长度为 8，这些不可修改。

使用 CAN 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或者位置控制。
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. 设置驱动器的编组和编号。
4. 设置 CAN 的波特率。（默认为 1000kbps）

在 CAN 通信协议下，

主控器对驱动器的操作命令有如下 12 种：

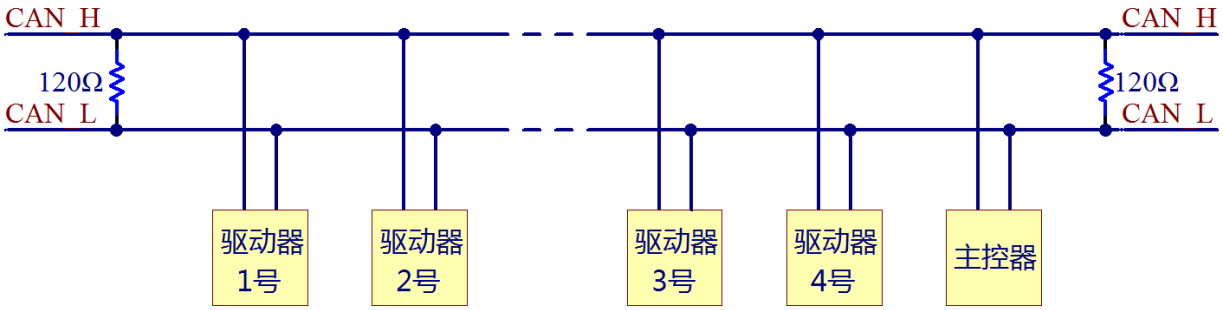
1. 让驱动器复位。
2. 让驱动器进入以下的 8 个运动模式的其中一个
3. 开环模式下，给驱动器发送数据指令
4. 电流模式下，给驱动器发送数据指令
5. 速度模式下，给驱动器发送数据指令
6. 位置模式下，给驱动器发送数据指令
7. 速度位置模式下，给驱动器发送数据指令
8. 电流速度模式下，给驱动器发送数据指令
9. 电流位置模式下，给驱动器发送数据指令
10. 电流速度位置模式下，给驱动器发送数据指令
11. 配置驱动器对外发送电流、速度、位置等数据的周期和对外发送 CTL1、CTL2 的电平状态的周期。
12. 在线检测指令

驱动器对外发送的指令有如下 3 种：

1. 在进入任意的运动模式后，对外发送当前电流、当前速度、当前位置值。
2. 在进入任意的运动模式后，对外发送 CTL1、CTL2 的电平值。（此功能仅对带有 CTL1、CTL2 的驱动器有实际意义）
3. 接收到在线检测指令的消息包后，对外发送一条同样的消息包，证明自己的存在。

CAN 总线组网连线示意图：

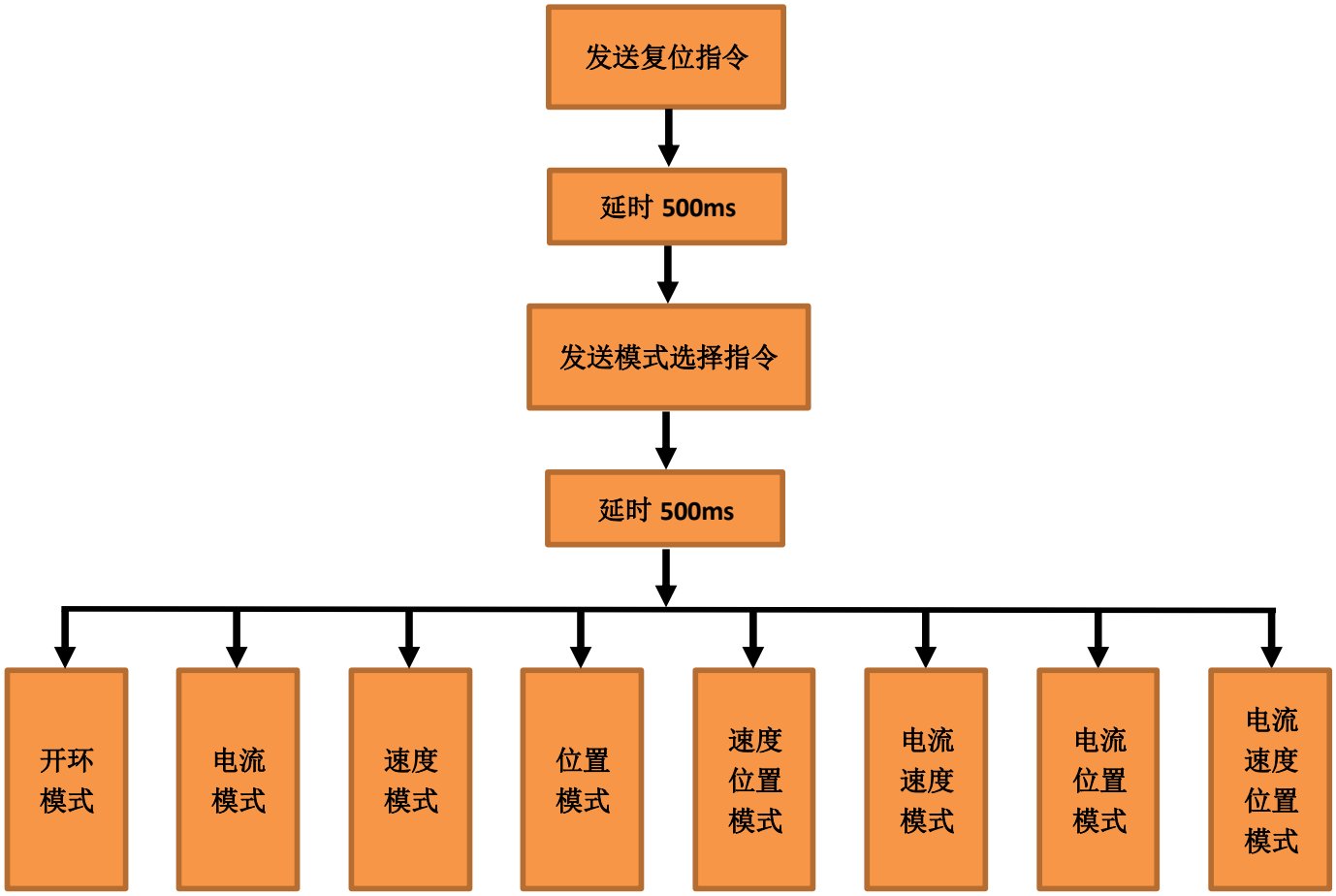
根据 CAN 总线的硬件特性，当一条 CAN 总线上挂接多个驱动器的时候，应当按照如下示意图进行连接布线。需要在线的起点和终点上各连上一个 120Ω 的电阻。并且 CAN 总线只能有一条主干线，分支的线不宜过长。



RMDS 系列的所有驱动器的内部都没有焊接上 120Ω 的电阻。所以用户需要多个驱动器一起使用 CAN 总线的时候，请在起点和末尾分别补上 120Ω 的电阻，直接跨接在 CANH 和 CANL 的外接端口上即可。

组网时候，每个驱动器的独立个体还需要连接各自的电机和编码器，以及电源等，对应如下：
电源线（24V/36V/48V、GND）、
编码器（CHA、CHB、GND、+5V）、
电机（MT1、MT2）、

控制流程图：



使用 CAN 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，对同一个驱动器之间的发间隔最短为 1ms，对多个驱动器发送消息，不同的驱动器间无需延时）

比如总线上一共有 4 个驱动器：

```
    程序初始化
    发送复位指令
    延时 500ms
    发送模式选择，进入速度模式
    延时 500ms
While（1）
{
    发送 1 号驱动器的速度；
    发送 2 号驱动器的速度；
    发送 3 号驱动器的速度；
    发送 4 号驱动器的速度；
    延时 1ms
}
```

零、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。

CAN_ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
待选择	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响一声，持续时间为 300ms。

驱动器的当前指令的独有 CAN_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编组	驱动器的编号	功能序号
0x	范围是 0-7	范围是 1-F	0

驱动器的当前指令的广播 CAN_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编组	驱动器的编号	功能序号
0x	范围是 0-7	0	0

举例说明：
出厂默认的驱动器编号是 0 组 1 号，
当前指令的独有 CAN_ID 是 0x010，也可以写成 0x10，
当前指令的广播 CAN_ID 是 0x000，也可以写成 0x00。
再比如：
0 组的第 10 号成员，当前指令的独有 CAN_ID 是 0x0A0，当前指令的广播 CAN_ID 是 0x000
6 组的第 02 号成员，当前指令的独有 CAN_ID 是 0x620，当前指令的广播 CAN_ID 是 0x600

备注：独有 CAN_ID 的指令发送后，只有当前指定的驱动器能接收到并响应。
广播 CAN_ID 的指令发送后，同组的所有驱动器都能接收到到并响应。

再次解释，独有 CAN_ID 和广播 CAN_ID 的问题：
假如有 15 个驱动器，都在 0 组，编号为 1 号到 15 号，那么现在要对 15 个驱动器进行复位，
两种方法：
第一种，发送以下 15 条 CAN 消息：（独有 CAN_ID 发送方法）
CAN_ID=0x10，标准帧,数据帧,内容为 55 55 55 55 55 55 55 55;//对 1 号驱动器复位
CAN_ID=0x20，标准帧,数据帧,内容为 55 55 55 55 55 55 55 55;//对 2 号驱动器复位
。。。。。。
。。。。。。
CAN_ID=0xE0，标准帧,数据帧,内容为 55 55 55 55 55 55 55 55;//对 14 号驱动器复位
CAN_ID=0xF0，标准帧,数据帧,内容为 55 55 55 55 55 55 55 55;//对 15 号驱动器复位
这种方式比较麻烦，要发送 15 个 CAN 消息，才能实现对 15 个驱动器的复位。
第二种，发送以下 1 条 CAN 消息：（广播 CAN_ID 发送方法）
CAN_ID=0x00，标准帧,数据帧,内容为 55 55 55 55 55 55 55 55;//对 0 组全部驱动器复位
这种方式比较快，这就是广播的优势。

一、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。

CAN_ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
待选择	待选择	0x55	0x55	0x55	0x55	0x55	0x55	0x55

驱动器的当前指令的 CAN_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编组	驱动器的编号	功能序号
0x	范围是 0-7	范围是 1-F	1

驱动器的当前指令的广播 CAN_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编组	驱动器的编号	功能序号
0x	范围是 0-7	0	1

举例说明：
出厂默认的驱动器编号是 0 组 1 号，
当前指令的独有 CAN_ID 是 0x011，也可以写成 0x11，
当前指令的广播 CAN_ID 是 0x001，也可以写成 0x01。
再比如：
0 组的第 10 号成员，当前指令的独有 CAN_ID 是 0x0A1，当前指令的广播 CAN_ID 是 0x001
6 组的第 02 号成员，当前指令的独有 CAN_ID 是 0x621，当前指令的广播 CAN_ID 是 0x601

备注：独有 CAN_ID 的指令发送后，只有当前指定的驱动器能接收到并响应。
广播 CAN_ID 的指令发送后，同组的所有驱动器都能接收到到并响应。

Data[0]与所选模式的对应值如下：

模式名称	Data[0]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

正确发送完本指令后，驱动器上的蜂鸣器会响一声，持续时间为 70ms，表示已经进入指定模式。

二、“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。

CAN_ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
待选择	待计算	待计算	0x55	0x55	0x55	0x55	0x55	0x55

驱动器的当前指令的 CAN_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编组	驱动器的编号	功能序号
0x	范围是 0-7	范围是 1-F	2

驱动器的当前指令的广播 CAN_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编组	驱动器的编号	功能序号
0x	范围是 0-7	0	2

举例说明：
出厂默认的驱动器编号是 0 组 1 号，
当前指令的独有 CAN_ID 是 0x012，也可以写成 0x12，
当前指令的广播 CAN_ID 是 0x002，也可以写成 0x02。
再比如：
0 组的第 10 号成员，当前指令的独有 CAN_ID 是 0x0A2，当前指令的广播 CAN_ID 是 0x002
6 组的第 02 号成员，当前指令的独有 CAN_ID 是 0x622，当前指令的广播 CAN_ID 是 0x602

备注：独有 CAN_ID 的指令发送后，只有当前指定的驱动器能接收到并响应。
广播 CAN_ID 的指令发送后，同组的所有驱动器都能接收到到并响应。

让 xx 号驱动器连接的电机在“开环模式”下，让它以 temp_pwm 的占空比转动：

```
则  Data[0] = (unsigned char)((temp_pwm>>8)&0xff);
    Data[1] = (unsigned char)(temp_pwm&0xff);
```

其中
temp_pwm 的取值范围为：-5000~+5000。

关于 temp_pwm 限制值，在
3. 电流模式
4. 速度模式
5. 位置模式
6. 速度位置模式
的取值问题：

当驱动器供电为 48V，电机的额定电压是 12V，此时 temp_pwm 限制值给定 1250 即可。
当驱动器供电为 48V，电机的额定电压是 24V，此时 temp_pwm 限制值给定 2500 即可。
当驱动器供电为 48V，电机的额定电压是 36V，此时 temp_pwm 限制值给定 3750 即可。
当驱动器供电为 48V，电机的额定电压是 48V，此时 temp_pwm 限制值给定 5000 即可。

当驱动器供电为 24V，电机的额定电压是 12V，此时 temp_pwm 限制值给定 2500 即可。
当驱动器供电为 24V，电机的额定电压是 24V，此时 temp_pwm 限制值给定 5000 即可。
当驱动器供电为 24V，电机的额定电压是 36V，此时 temp_pwm 限制值给定 5000 即可。
当驱动器供电为 24V，电机的额定电压是 48V，此时 temp_pwm 限制值给定 5000 即可。

当驱动器供电为 12V，电机的额定电压是 12V，此时 temp_pwm 限制值给定 5000 即可。
当驱动器供电为 12V，电机的额定电压是 24V，此时 temp_pwm 限制值给定 5000 即可。

temp_pwm 限制值在这里只起到一个按比例降压输出的作用，而无法实现升压。

temp_pwm 限制占空比，实际上就是限制输出电压，用于限制电机线圈所得电压不大于其额定电压，以此限制电机工作在额定电压范围之内。

三、“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 1 毫秒，建议以 10 毫秒为周期。给不同驱动器发送，可以连续发，不需要延时。

CAN_ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
待选择	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 CAN_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编组	驱动器的编号	功能序号
0x	范围是 0-7	范围是 1-F	3

驱动器的当前指令的广播 CAN_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编组	驱动器的编号	功能序号
0x	范围是 0-7	0	3

举例说明：
出厂默认的驱动器编号是 0 组 1 号，
当前指令的独有 CAN_ID 是 0x013，也可以写成 0x13，
当前指令的广播 CAN_ID 是 0x003，也可以写成 0x03。
再比如：
0 组的第 10 号成员，当前指令的独有 CAN_ID 是 0x0A3，当前指令的广播 CAN_ID 是 0x003
6 组的第 02 号成员，当前指令的独有 CAN_ID 是 0x623，当前指令的广播 CAN_ID 是 0x603

备注：独有 CAN_ID 的指令发送后，只有当前指定的驱动器能接收到并响应。
广播 CAN_ID 的指令发送后，同组的所有驱动器都能接收到到并响应。

让 xx 号驱动器连接的电机在“电流模式”下，设置 temp_pwm 的限制占空比，输出 temp_current 的电流：
则
```
Data[0] = (unsigned char)((temp_pwm>>8)&0xff);
Data[1] = (unsigned char)(temp_pwm&0xff);
Data[2] = (unsigned char)((temp_current>>8)&0xff);
Data[3] = (unsigned char)(temp_current&0xff);
```

其中
temp_pwm 的取值范围为：0~+5000，具体如何取值请参考本篇第 9 页。
temp_current 的取值范围为：-32768~+32767。(16 位有符号整型数的取值范围，单位 mA)

四、“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 1 毫秒，建议以 10 毫秒为周期。给不同驱动器发送，可以连续发，不需要延时。

CAN_ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
待选择	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 CAN_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编组	驱动器的编号	功能序号
0x	范围是 0-7	范围是 1-F	4

驱动器的当前指令的广播 CAN_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编组	驱动器的编号	功能序号
0x	范围是 0-7	0	4

举例说明：
出厂默认的驱动器编号是 0 组 1 号，
当前指令的独有 CAN_ID 是 0x014，也可以写成 0x14，
当前指令的广播 CAN_ID 是 0x004，也可以写成 0x04。
再比如：
0 组的第 10 号成员，当前指令的独有 CAN_ID 是 0x0A4，当前指令的广播 CAN_ID 是 0x004
6 组的第 02 号成员，当前指令的独有 CAN_ID 是 0x624，当前指令的广播 CAN_ID 是 0x604

备注：独有 CAN_ID 的指令发送后，只有当前指定的驱动器能接收到并响应。
广播 CAN_ID 的指令发送后，同组的所有驱动器都能接收到到并响应。

让 xx 号驱动器连接的电机在“速度模式”下以 temp_pwm 的限制占空比，以 temp_velocity 的速度转动：
则
```
Data[0] = (unsigned char)((temp_pwm>>8)&0xff);
Data[1] = (unsigned char)(temp_pwm&0xff);
Data[2] = (unsigned char)((temp_velocity>>8)&0xff);
Data[3] = (unsigned char)(temp_velocity&0xff);
```
其中
temp_pwm 的取值范围为：0~+5000，具体如何取值请参考本篇第 9 页。
temp_velocity 的取值范围为：-32768~+32767。(16 位有符号整型数的范围，单位 RPM)

五、“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 1 毫秒，建议以 10 毫秒为周期。给不同驱动器发送，可以连续发，不需要延时。给不同驱动器发送，可以连续发，不需要延时。

CAN_ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
待选择	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 CAN_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编组	驱动器的编号	功能序号
0x	范围是 0-7	范围是 1-F	5

驱动器的当前指令的广播 CAN_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编组	驱动器的编号	功能序号
0x	范围是 0-7	0	5

举例说明：
出厂默认的驱动器编号是 0 组 1 号，
当前指令的独有 CAN_ID 是 0x015，也可以写成 0x15，
当前指令的广播 CAN_ID 是 0x005，也可以写成 0x05。
再比如：
0 组的第 10 号成员，当前指令的独有 CAN_ID 是 0x0A5，当前指令的广播 CAN_ID 是 0x005
6 组的第 02 号成员，当前指令的独有 CAN_ID 是 0x625，当前指令的广播 CAN_ID 是 0x605

备注：独有 CAN_ID 的指令发送后，只有当前指定的驱动器能接收到并响应。
广播 CAN_ID 的指令发送后，同组的所有驱动器都能接收到到并响应。

让 xx 号驱动器连接的电机在“位置模式”下以 temp_pwm 的限制占空比，转动到 temp_position 的位置：
则
```
Data[0] = (unsigned char)((temp_pwm>>8)&0xff);
Data[1] = (unsigned char)((temp_pwm)&0xff);
Data[2] = 0x55;
Data[3] = 0x55;
Data[4] = (unsigned char)((temp_position>>24)&0xff);
Data[5] = (unsigned char)((temp_position>>16)&0xff);
Data[6] = (unsigned char)((temp_position>>8)&0xff);
Data[7] = (unsigned char)(temp_position&0xff);
```
其中
temp_pwm 的取值范围为：0~+5000，具体如何取值请参考本篇第 9 页。
temp_position 的取值范围为：-2147483648~+2147483647。(32 位有符号整型数的范围，单位 qc)

六、“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 1 毫秒，建议以 10 毫秒为周期。给不同驱动器发送，可以连续发，不需要延时。

CAN_ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
待选择	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 CAN_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编组	驱动器的编号	功能序号
0x	范围是 0-7	范围是 1-F	6

驱动器的当前指令的广播 CAN_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编组	驱动器的编号	功能序号
0x	范围是 0-7	0	6

举例说明：
出厂默认的驱动器编号是 0 组 1 号，
当前指令的独有 CAN_ID 是 0x016，也可以写成 0x16，
当前指令的广播 CAN_ID 是 0x006，也可以写成 0x06。
再比如：
0 组的第 10 号成员，当前指令的独有 CAN_ID 是 0x0A6，当前指令的广播 CAN_ID 是 0x006
6 组的第 02 号成员，当前指令的独有 CAN_ID 是 0x626，当前指令的广播 CAN_ID 是 0x606

备注：独有 CAN_ID 的指令发送后，只有当前指定的驱动器能接收到并响应。
广播 CAN_ID 的指令发送后，同组的所有驱动器都能接收到到并响应。

让 xx 号驱动器连接的电机在“速度位置模式”下以 temp_pwm 的限制占空比，temp_velocity 的限制速度，转动到 temp_position 的位置：

```
则  Data[0] = (unsigned char)((temp_pwm>>8)&0xff);
    Data[1] = (unsigned char)((temp_pwm)&0xff);
    Data[2] = (unsigned char)((temp_velocity>>8)&0xff);
    Data[3] = (unsigned char)(temp_velocity&0xff);
    Data[4] = (unsigned char)((temp_position>>24)&0xff);
    Data[5] = (unsigned char)((temp_position>>16)&0xff);
    Data[6] = (unsigned char)((temp_position>>8)&0xff);
    Data[7] = (unsigned char)(temp_position&0xff);
```

其中
temp_pwm 的取值范围为：0~+5000，具体如何取值请参考本篇第 9 页。
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）

七、“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 1 毫秒，建议以 10 毫秒为周期。给不同驱动器发送，可以连续发，不需要延时。

CAN_ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
待选择	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 CAN_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编组	驱动器的编号	功能序号
0x	范围是 0-7	范围是 1-F	7

驱动器的当前指令的广播 CAN_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编组	驱动器的编号	功能序号
0x	范围是 0-7	0	7

举例说明：
出厂默认的驱动器编号是 0 组 1 号，
当前指令的独有 CAN_ID 是 0x017，也可以写成 0x17，
当前指令的广播 CAN_ID 是 0x007，也可以写成 0x07。
再比如：
0 组的第 10 号成员，当前指令的独有 CAN_ID 是 0x0A7，当前指令的广播 CAN_ID 是 0x007
6 组的第 02 号成员，当前指令的独有 CAN_ID 是 0x627，当前指令的广播 CAN_ID 是 0x607

备注：独有 CAN_ID 的指令发送后，只有当前指定的驱动器能接收到并响应。
广播 CAN_ID 的指令发送后，同组的所有驱动器都能接收到到并响应。

让 xx 号驱动器连接的电机在“电流速度模式”下，以 temp_current 的限制电流，以 temp_velocity 的速度转动：

```
则  Data[0] = (unsigned char)((temp_current>>8)&0xff);
    Data[1] = (unsigned char)(temp_current&0xff);
    Data[2] = (unsigned char)((temp_velocity>>8)&0xff);
    Data[3] = (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)

八、“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电流的限制值，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 1 毫秒，建议以 10 毫秒为周期。给不同驱动器发送，可以连续发，不需要延时。

CAN_ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
待选择	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 CAN_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编组	驱动器的编号	功能序号
0x	范围是 0-7	范围是 1-F	8

驱动器的当前指令的广播 CAN_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编组	驱动器的编号	功能序号
0x	范围是 0-7	0	8

举例说明：
出厂默认的驱动器编号是 0 组 1 号，
当前指令的独有 CAN_ID 是 0x018，也可以写成 0x18，
当前指令的广播 CAN_ID 是 0x008，也可以写成 0x08。
再比如：
0 组的第 10 号成员，当前指令的独有 CAN_ID 是 0x0A8，当前指令的广播 CAN_ID 是 0x008
6 组的第 02 号成员，当前指令的独有 CAN_ID 是 0x628，当前指令的广播 CAN_ID 是 0x608

备注：独有 CAN_ID 的指令发送后，只有当前指定的驱动器能接收到并响应。
广播 CAN_ID 的指令发送后，同组的所有驱动器都能接收到到并响应。

举例：
让 xx 号驱动器连接的电机在“电流位置模式”下以 temp_current 的限制电流,转动到 temp_position 的位置：
则
```
Data[0] = (unsigned char)((temp_current>>8)&0xff);
Data[1] = (unsigned char)((temp_current)&0xff);
Data[2] = 0x55;
Data[3] = 0x55;
Data[4] = (unsigned char)((temp_position>>24)&0xff);
Data[5] = (unsigned char)((temp_position>>16)&0xff);
Data[6] = (unsigned char)((temp_position>>8)&0xff);
Data[7] = (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）

九、“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电流的限制值，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 1 毫秒，建议以 10 毫秒为周期。给不同驱动器发送，可以连续发，不需要延时。

CAN_ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
待选择	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 CAN_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编组	驱动器的编号	功能序号
0x	范围是 0-7	范围是 1-F	9

驱动器的当前指令的广播 CAN_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编组	驱动器的编号	功能序号
0x	范围是 0-7	0	9

举例说明：
出厂默认的驱动器编号是 0 组 1 号，
当前指令的独有 CAN_ID 是 0x019，也可以写成 0x19，
当前指令的广播 CAN_ID 是 0x009，也可以写成 0x09。
再比如：
0 组的第 10 号成员，当前指令的独有 CAN_ID 是 0x0A9，当前指令的广播 CAN_ID 是 0x009
6 组的第 02 号成员，当前指令的独有 CAN_ID 是 0x629，当前指令的广播 CAN_ID 是 0x609

备注：独有 CAN_ID 的指令发送后，只有当前指定的驱动器能接收到并响应。
广播 CAN_ID 的指令发送后，同组的所有驱动器都能接收到到并响应。

让 xx 号驱动器连接的电机在“电流速度位置模式”下以 temp_current 的限制电流，temp_velocity 的限制速度，转动到 temp_position 的位置：

```
则  Data[0] = (unsigned char)((temp_current>>8)&0xff);
    Data[1] = (unsigned char)((temp_current)&0xff);
    Data[2] = (unsigned char)((temp_velocity>>8)&0xff);
    Data[3] = (unsigned char)(temp_velocity&0xff);
    Data[4] = (unsigned char)((temp_position>>24)&0xff);
    Data[5] = (unsigned char)((temp_position>>16)&0xff);
    Data[6] = (unsigned char)((temp_position>>8)&0xff);
    Data[7] = (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、配置指令：（功能序号为 A） 2018 年 07 月 28 日修改

配置指令，目前包含两个功能：
①Data[0]可以决定是否让驱动器以某个固定的时间间隔通过 CAN 总线对外发送电机当前的实时电流、速度、位置值等信息。详细内容参考第 11 节。
②Data[1]可以决定是否让驱动器以某个固定的时间间隔通过CAN总线对外发送CTL1/CTL2/DSIN/CHZ的电平状态，和 ASIN 端口的输入电压，以及此刻 PWM 的输出值。详细内容参考第 12 节。

本指令在任何状态下都可以发送并生效。
但驱动器只在进入上文的 8 种运动模式之后，且配置了固定周期后，才会周期性的对外发送上述信息。

CAN_ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
待选择	待给定	待给定	0x55	0x55	0x55	0x55	0x55	0x55

驱动器的当前指令的 CAN_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编组	驱动器的编号	功能序号
0x	范围是 0-7	范围是 1-F	A

驱动器的当前指令的广播 CAN_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编组	驱动器的编号	功能序号
0x	范围是 0-7	0	A

举例说明：
出厂默认的驱动器编号是 0 组 1 号，
当前指令的独有 CAN_ID 是 0x01A，也可以写成 0x1A，
当前指令的广播 CAN_ID 是 0x00A，也可以写成 0x0A。
再比如：
0 组的第 10 号成员，当前指令的独有 CAN_ID 是 0x0AA，当前指令的广播 CAN_ID 是 0x00A
6 组的第 02 号成员，当前指令的独有 CAN_ID 是 0x62A，当前指令的广播 CAN_ID 是 0x60A

备注：独有 CAN_ID 的指令发送后，只有当前指定的驱动器能接收到并响应。
广播 CAN_ID 的指令发送后，同组的所有驱动器都能接收到到并响应。

举例：
让 0 组的 02 号驱动器
以 10 毫秒为周期的对外发送电流、速度、位置等信息，
以 10 毫秒为周期的对外发送 CTL1/CTL2/DSIN/CHZ 的电平状态，和 ASIN 端口的输入电压，以及此刻 PWM 的输出值。
的指令为：

```
CAN_ID : 0x02A
Data[0] = 0x0A; //电流速度位置实际值对外发送周期，单位为毫秒，0x00 为不发送。
Data[1] = 0x0A; //对外发送周期，单位为毫秒，0x00 为不发送。（2018 年 07 月 28 日修改）
Data[2] = 0x55;
```

```
Data[3] = 0x55;
Data[4] = 0x55;
Data[5] = 0x55;
Data[6] = 0x55;
Data[7] = 0x55;
```

十一、电流速度位置的定时反馈：(功能序号为B)

以下是驱动器对外发送电流、速度、位置等信息的CAN消息的格式。特别注意，这条CAN消息是由驱动器发出。

CAN_ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
待接收	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CAN_ID的组成遵循如下规则：

十六进制数开头	驱动器的编组	驱动器的编号	功能序号
0x	范围是0-7	范围是1-F	B

本功能不支持广播指令。

举例：
0组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，则驱动器则会对外发送如下的CAN消息：

```
CAN_ID:0x02B
Data[0] = (unsigned char)((real_current>>8)&0xff);
Data[1] = (unsigned char)(real_current&0xff);
Data[2] = (unsigned char)((real_velocity>>8)&0xff);
Data[3] = (unsigned char)(real_velocity&0xff);
Data[4] = (unsigned char)((real_position>>24)&0xff);
Data[5] = (unsigned char)((real_position>>16)&0xff);
Data[6] = (unsigned char)((real_position>>8)&0xff);
Data[7] = (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[0]<<8)|Data[1];
int16_t real_velocity = (Data[2]<<8)|Data[3];
int32_t real_position = (Data[4]<<24)| (Data[5]<<16)| (Data[6]<<8)| Data[7];
```

易错点：
real_current 和 real_velocity 必须定义为16bit的有符号整型数，否则当其为负数的时候，拼接还原会出错。一般定义为short都不会错，short在大部分编译环境下都表示为16bit有符号整型数。在库函数支持的情况下，直接定义为int16_t，是最安全的。
real_position 必须定义为32bit有符号整型数，否则当其为负数的时候，拼接还原会出错。一般定义为int或者long。但int和long在16bit/32bit/64bit的单片机或者编译器或者操作系统中会表示为不同bit数，这个问题比较复杂，要具体情况具体分析。在库函数支持的情况下，直接定义为int32_t，是最安全的。
（一般用sizeof函数可以获知int或者long具体表示多少bit，具体如何操作不是本篇应该讨论的，请自行查资料）

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。

十二、混杂项的定时反馈：(功能序号为 C)　2018 年 07 月 28 日修改

本节介绍驱动器定时对外发送：

1. CTL1 电平状态
2. CTL2 电平状态
3. DSIN 电平状态
4. ASIN 端口的电压
5. CHZ 端口的电平状态（仅对 RMDS-40x 系列有意义）
6. 驱动器此刻对外输出的 PWM 值

等 6 种信息的 CAN 数据包。

特别注意，这条 CAN 消息是由驱动器发出，需要满足两个条件，驱动器才会对外定时发出该数据包

1. 通过 CAN 配置指令，设置了不为 0 的周期。
2. 驱动器进入了任意一种运动模式。

CAN_ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
待接收	CTL1	CTL2	DSIN	ASIN 高	ASIN 低	CHZ	PWM 高	PWM 低

驱动器的当前指令的 CAN_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编组	驱动器的编号	功能序号
0x	范围是 0-7	范围是 1-F	C

本功能不支持广播指令。

举例说明

0 组的第 10 号成员，当前指令的 CAN_ID 是 0x0AC

6 组的第 02 号成员，当前指令的 CAN_ID 是 0x62C

当 CTL1 端口的电平为低电平或者悬空，或光耦为熄灭状态时，Data[0] = 0x00，为高电平时候或光耦为点亮状态时，Data[0] = 0x01。无 CTL1 端口的驱动器，此数值无意义。

当 CTL2 端口的电平为低电平或者悬空，或光耦为熄灭状态时，Data[1] = 0x00，为高电平时候或光耦为点亮状态时，Data[1] = 0x01。无 CTL2 端口的驱动器，此数值无意义。

当 DSIN 端口的电平为低电平或者悬空，Data[2] = 0x00,为高电平时候，Data[2] = 0x01。无 DSIN 端口的驱动器，此数值无意义。

当 ASIN 端口的电压为 N 毫伏时候，Data[3]=(unsigned char)(N>>8),Data[4] = (unsigned char)(N),注意，ASIN 只支持输入正电压，不支持输入负电压。比如对 ASIN 输入 5V 的电压时候，也就是 5000mV，此时，Data[3]=0x13,Data[4]=0x88,当由于 AD 采样的波动，Data[4]的输出值可能会跳动。无 ASIN 端口的驱动器，Data[3]和 Data[4]的数值无意义。

当 CHZ 端口的输入电平为低电平时，Data[5]=0x00,输入电平为高电平或者悬空时，Data[5]=0x01。无 CHZ 端口版本的驱动器，Data[5]数值无意义。

Data[6]和 Data[7]则表示此时驱动器输出的 PWM 的值，当驱动器的频率设置为 14.4KH 时，此处输出的 PWM 范围为-5000~+5000，当驱动器的频率设置为 28.8KHz，此处输出的 PWM 的范围为-2500~+2500。此处的正负符号，只区分正反转的 PWM 输出，未经标定，仅供参考。

十三、错误输出：（功能序号为 D）

以下是驱动器发生错误后，对外发送数据包的指令格式。
特别注意，这条 CAN 消息是由驱动器发出，在发生错误后，每隔 100ms 对外发送一次如下数据包。

CAN_ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
待接收	待接收	0x55	0x55	0x55	0x55	0x55	0x55	0x55

驱动器的当前指令的 CAN_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编组	驱动器的编号	功能序号
0x	范围是 0-7	范围是 1-F	D

本功能不支持广播指令。

举例说明
0 组的第 10 号成员，当前指令的 CAN_ID 是 0x0AD
6 组的第 02 号成员，当前指令的 CAN_ID 是 0x62D

其中，当发生堵转保护时候，
Data[0] = 0x01

其他暂未添加

十四、设置 OUT1 和 OUT2 输出状态(功能序号为 E) 2018 年 07 月 28 日修改

RMDS-201 驱动器，提供了一个 F+/F-的输出口，实际为 OUT1。
RMDS-40x 驱动器，提供了 OUT1 和 OUT2 的接口。
可以通过此命令，来设置 OUT1 和 OUT2 的状态。

CAN_ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
待选择	OUT1	OUT2	0x55	0x55	0x55	0x55	0x55	0x55

驱动器的当前指令的 CAN_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编组	驱动器的编号	功能序号
0x	范围是 0-7	范围是 1-F	E

驱动器的当前指令的广播 CAN_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编组	驱动器的编号	功能序号
0x	范围是 0-7	0	E

举例说明：
出厂默认的驱动器编号是 0 组 1 号，
当前指令的独有 CAN_ID 是 0x01E，也可以写成 0x1E，
当前指令的广播 CAN_ID 是 0x00E，也可以写成 0x0E。
再比如：
0 组的第 10 号成员，当前指令的独有 CAN_ID 是 0x0AE，当前指令的广播 CAN_ID 是 0x00E
6 组的第 02 号成员，当前指令的独有 CAN_ID 是 0x62E，当前指令的广播 CAN_ID 是 0x60E

备注：独有 CAN_ID 的指令发送后，只有当前指定的驱动器能接收到并响应。
广播 CAN_ID 的指令发送后，同组的所有驱动器都能接收到到并响应。

Data[0]的取值，有效值为 0x00 和 0x01,除此之外的任何值，都会被驱动器忽略。
Data[1]的取值，有效值为 0x00 和 0x01,除此之外的任何值，都会被驱动器忽略。

假设需要点亮 OUT1 的光耦，请将 Data[0]=0x00,需要熄灭 OUT1 的光耦，请将 Data[0]=0x01;
假设需要点亮 OUT2 的光耦，请将 Data[1]=0x00,需要熄灭 OUT2 的光耦，请将 Data[1]=0x01;

另外，驱动器复位后的默认的 OUT1 和 OUT2 状态，为光耦熄灭。

十五、在线检测：（功能序号为 F）

如果主控需要检查某个驱动器是否连接在 CAN 总线上，则发送以下数据包即可。本条 CAN 消息在任何时候都会生效。

CAN_ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
待选择	0x55	0x55	0x55	0x55	0x55	0x55	0x55	0x55

驱动器的当前指令的 CAN_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编组	驱动器的编号	功能序号
0x	范围是 0-7	范围是 1-F	F

驱动器的当前指令的广播 CAN_ID 的组成遵循如下规则：

十六进制数开头	驱动器的编组	驱动器的编号	功能序号
0x	范围是 0-7	0	F

举例说明：
出厂默认的驱动器编号是 0 组 1 号，
当前指令的独有 CAN_ID 是 0x01F，也可以写成 0x1F，
当前指令的广播 CAN_ID 是 0x00F，也可以写成 0x0F。
再比如：
0 组的第 10 号成员，当前指令的独有 CAN_ID 是 0x0AF，当前指令的广播 CAN_ID 是 0x00F
6 组的第 02 号成员，当前指令的独有 CAN_ID 是 0x62F，当前指令的广播 CAN_ID 是 0x60F

备注：独有 CAN_ID 的指令发送后，只有当前指定的驱动器能接收到并响应。
广播 CAN_ID 的指令发送后，同组的所有驱动器都能接收到到并响应。

当驱动器成功接收到本条 CAN 消息，会返回一条一模一样的 CAN 消息。

使用 USBCAN 工具测试计算机与驱动器的 CAN 通信

所有的 USBCAN 的测试软件都差不多，下面以周立功的 CANTest 的为例：
（举例的驱动器编号为默认的为 0 组 1 号，驱动器不需要经过任何配置即可测试）
界面配置为：（波特率为 1M）

1. 发送方式：正常发送
2. 帧类型：标准帧
3. 帧格式：数据帧
4. 发送次数：1
5. 每次发送间隔：0
6. 设置为每次发送单帧

然后现在测试在线检测指令：
帧 ID（HEX）： 1F
数据（HEX）：55 55 55 55 55 55 55 55
然后点击发送即可，发送成功后，将接受到一帧一模一样的 CAN 消息，同时在界面上会有显示，如下图。

另外，简单介绍，如何使用 USBCAN 让电机动起来。

一、发送复位指令
帧 ID：10　数据：55 55 55 55 55 55 55 55

二、发送模式选择，这里以开环模式为例
帧 ID：11　数据：01 55 55 55 55 55 55 55

三、让电机在开环模式下，以 PWM=500 转起来
帧 ID：12　数据：01 F4 55 55 55 55 55 55

四、让电机在开环模式下，以 PWM=0 停下
帧 ID：12　数据：00 00 55 55 55 55 55 55

五、让电机在开环模式下，以 PWM=-500 反向转起来
帧 ID：12　数据：FE 0C 55 55 55 55 55 55

六、让电机在开环模式下，以 PWM=0 停下
帧 ID：12　数据：00 00 55 55 55 55 55 55

七、结束测试，让驱动器复位
帧 ID：10　数据：55 55 55 55 55 55 55 55

如果不知道上述 01 F4 和 FE 0C 是怎么算出来的，请参考说明书，常见疑问解答的 11-15 页。