RS232 串口通信协议

本篇文档适用于 RMDS-101、RMDS-102、RMDS-103 等三个版本的 RoboModule 驱动器。

本段介绍如何使用 RS232 串口的方式操作驱动器来控制电机的各种方式转动。

在 RS232 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）、
RS232 线（232T、GND、232R）。
其中，232T 要连接到主控器的 RS232 接收端（R 端），232R 要连接到主控中心的 RS232 发送端（T 端）。

对于本段协议，RS232 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。RS232 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：

由于 RS232 串口是一对一的传输方式，所以，一个主控器不能通过同一个 RS232 接口来挂接两个驱动器，如果要使用多个驱动器并且使用 RS232 串口挂接在主控器上，则要求该主控器有多个 RS232 接口。比如要控制三个驱动器，则要求使用三个 RS232 串口才能实现完整的控制。

当不具备这种多 RS232 串口资源的时候，CAN 总线和 RS485 则是更好的选择。一条 CAN 总线，可以直接挂接 120 个驱动器，一条 RS485 总线，可以支持挂接 15 个驱动器。

下面正式介绍 RS232 串口通信协议：

因为串口指令只能是一个主机（电脑作为主机或者 MCU 主控器作为主机）和一个从机（当前连接的驱动器作为从机），所以，所发送的所有指令，都只有本串口所连接的驱动器能收到，所以不存在选中哪一个驱动器

的问题，所以不需要像 CAN 总线或 RS485 那样用标识符来区分谁是谁。

每段串口命令都是由 10 个字节组成，

主控器发送给驱动器的指令，有如下 11 种：

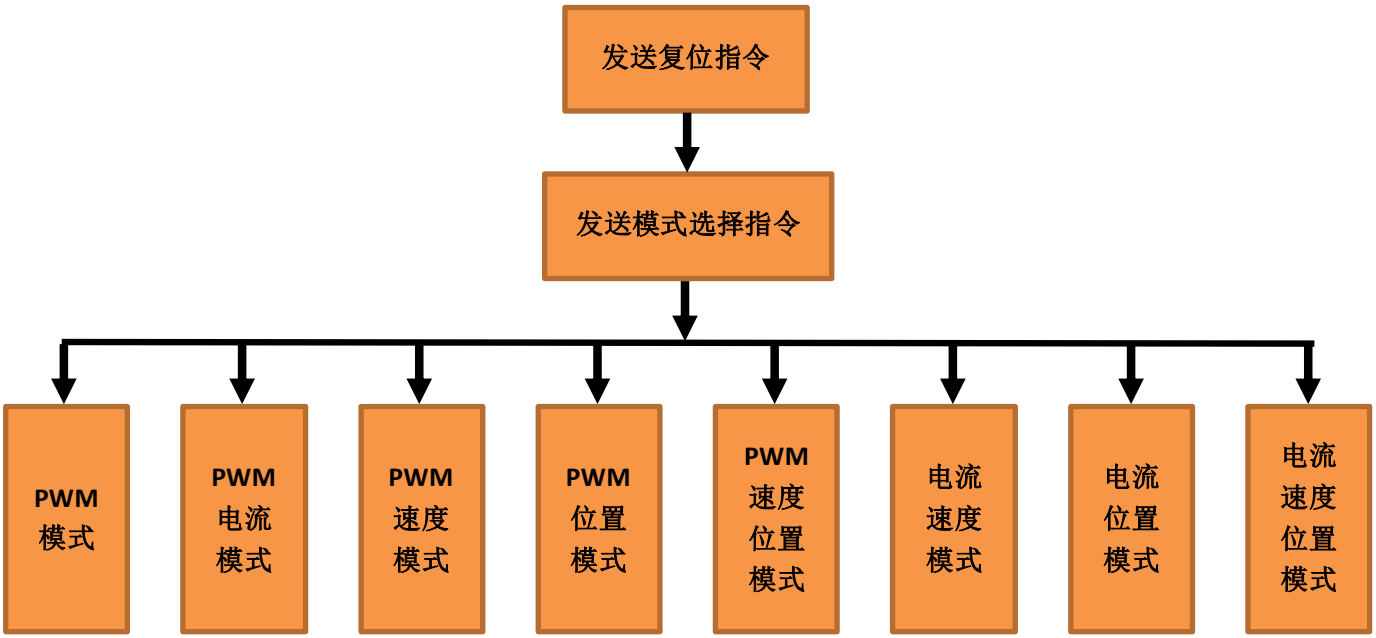
1. 让当前驱动器复位。
2. 让当前驱动器进入某种模式，有 8 种可选模式：
   ① PWM 模式
   ② PWM 电流模式
   ③ PWM 速度模式
   ④ PWM 位置模式
   ⑤ PWM 速度位置模式
   ⑥ 电流速度模式
   ⑦ 电流位置模式
   ⑧ 电流速度位置模式
3. 在 PWM 模式下给驱动器发送目标 PWM 的指令。
4. 在 PWM 电流模式下给驱动器发送限制 PWM 和目标电流的指令。（电流值只是一个相对值，没有单位）
5. 在PWM速度模式下给驱动器发送限制PWM和目标速度的指令。(速度值是指1ms时间内编码器转过的线数)
6. 在 PWM 位置模式下给驱动器发送限制 PWM 和目标位置的指令。
7. 在 PWM 速度位置模式下给驱动器发送限制 PWM、限制速度、目标位置的指令。
8. 在电流速度模式下，给驱动器发送限制电流和目标速度的指令。
9. 在电流位置模式下，给驱动器发送限制电流和目标位置的指令。
10. 在电流速度位置模式下，给驱动器发送限制电流和限制速度和目标位置的指令。
11. 配置指令，配置驱动器对外发送电流、速度、位置等信息的时间间隔和配置开启或关闭 CTL 端口在电平变化时候的发送功能。

驱动器发送给主控器的指令，有如下 2 种：

1. 驱动器对外发送电流、速度、位置等信息。
2. 驱动器对外发送 CTL1、CTL2 端口电平状态。

下面来分解每个指令的具体内容：

一、控制流程图：



使用 RS232 串口的方式控制驱动器，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：

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]
0x23	0x00	0x55	0x55	0x55	0x55	0x55	0x55	0x55	0x55

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。

三、模式选择指令：

本指令只在驱动器未进入任何模式的情况下生效。
如果驱动器已经进入某种模式，再发送此指令则会报错（报错的具体表现为红灯闪烁，驱动器的蜂鸣器不断的鸣叫）。
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]
0x23	0x01	待选择	0x55	0x55	0x55	0x55	0x55	0x55	0x55

Data[2]与所选模式的对应值如下：

模式名称	Data[2]的值
PWM 模式	0x01
PWM 电流模式	0x02
PWM 速度模式	0x03
PWM 位置模式	0x04
PWM 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

当驱动器进入上述 8 种模式中的任何一种的时候，蜂鸣器都会 Bi 的一声（持续时间为 70ms），表示进入模式成功。

四、“PWM 模式”下的数据指令：

本指令只有在驱动器进入 PWM 模式之后才生效，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]
0x23	0x02	待计算	待计算	0x55	0x55	0x55	0x55	0x55	0x55

举例说明：
让当前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[0] = 0x23;
Data[1] = 0x02;
Data[2] = (unsigned char)((temp_pwm>>8)&0xff);
Data[3] = (unsigned char)(temp_pwm&0xff);
Data[4] = 0x55;
Data[5] = 0x55;
Data[6] = 0x55;
Data[7] = 0x55;
Data[8] = 0x55;
Data[9] = 0x55;
```

其中：
temp_pwm 的取值范围为：-5000~+5000。

五、“PWM 电流模式”下的数据指令：

本指令只在驱动器进入电流环模式之后才有效，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、电流的值，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]
0x23	0x03	待计算	待计算	待计算	待计算	0x55	0x55	0x55	0x55

举例说明：
让当前驱动器连接的电机以 temp_pwm 的限制 PWM，以 temp_current 的最大电流工作：
则

```
Data[0] = 0x23;
Data[1] = 0x03;
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);
Data[6] = 0x55;
Data[7] = 0x55;
Data[8] = 0x55;
Data[9] = 0x55;
```

其中：
temp_pwm 的取值范围为：0~+5000。
temp_current 的取值范围为：-1600~+1600。

六、“PWM 速度模式”下的数据指令：

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]
0x23	0x04	待计算	待计算	待计算	待计算	0x55	0x55	0x55	0x55

举例说明：
让当前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[0] = 0x23;
Data[1] = 0x04;
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] = 0x55;
Data[7] = 0x55;
Data[8] = 0x55;
Data[9] = 0x55;
```

其中：
temp_pwm 的取值范围为：0~+5000。
temp_velocity 的取值范围为：-32768~+32767。(16 位有符号整型数的范围)

七、“PWM 位置模式”下的参数指令：

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]
0x23	0x05	待计算	待计算	0x55	0x55	待计算	待计算	待计算	待计算

举例说明：
让当前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[0] = 0x23;
Data[1] = 0x05;
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 位有符号整型数的范围）

八、“PWM 速度位置模式”下的参数指令：

本指令只在驱动器进入“PWM 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]
0x23	0x06	待计算	待计算	待计算	待计算	待计算	待计算	待计算	待计算

让当前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[0] = 0x23;
Data[1] = 0x06;
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）
temp_position 的取值范围为：-2147483648~+2147483647。（32 位有符号整型数的范围）

九、“电流速度模式”下的数据指令：

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]
0x23	0x07	待计算	待计算	待计算	待计算	0x55	0x55	0x55	0x55

举例说明：
让当前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[0] = 0x23;
Data[1] = 0x07;
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] = 0x55;
Data[7] = 0x55;
Data[8] = 0x55;
Data[9] = 0x55;
```

其中：
temp_current 的取值范围为 0~+1600。
temp_velocity 的取值范围为：-32768~+32767。（16 位有符号整型数的范围）

十、“电流位置模式”下的参数指令：

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]
0x23	0x08	待计算	待计算	0x55	0x55	待计算	待计算	待计算	待计算

举例：
让当前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[0] = 0x23;
Data[1] = 0x08;
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~+1600。
temp_position 的取值范围为：-2147483648~+2147483647。（32 位有符号整型数的范围）

十一、“电流速度位置模式”下的参数指令：

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]
0x23	0x09	待计算	待计算	待计算	待计算	待计算	待计算	待计算	待计算

让当前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[0] = 0x23;
Data[1] = 0x09;
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~+1600。

temp_velocity 的取值范围为：0~+32767。（16 位有符号整型数的正数范围）

temp_position 的取值范围为：-2147483648~+2147483647。（32 位有符号整型数的范围）

十二、配置指令：

配置指令，目前包含两个功能：
①可以决定是否让驱动器以某个固定的时间间隔通过 RS232 串口对外发送电机当前的实时电流、速度、位置值等信息。
②可以决定 CTL1 和 CTL2 端口在作为左右限位功能后，在电平发生变化时，是否对外发送当前的电平值。

本指令在任何状态下都可以生效。
但驱动器只在进入上文的 8 种运动模式之后，才会周期性的对外发送以上所述的电流、速度、位置等信息。
同理，只有进入上文的 8 种运动模式之后，才会在 CTL 端口电平变化后对外发送其电平值。

Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]	Data[8]	Data[9]
0x23	0x0A	待计算	待给定	0x55	0x55	0x55	0x55	0x55	0x55

举例：
让当前驱动器以 100 毫秒为周期的对外发送电流、速度、位置等信息的指令为：

```
Data[0] = 0x23;
Data[1] = 0x0A;
Data[2] = 0x0A; //给定数据*10 毫秒 = 发送周期，单位为毫秒。
Data[3] = 0x01; //允许发送 CTL 端口的电平状态，0x00 为不允许
Data[4] = 0x55;
Data[5] = 0x55;
Data[6] = 0x55;
Data[7] = 0x55;
Data[8] = 0x55;
Data[9] = 0x55;
```

其中：
当 Data[3] = 0x00 的时候，不允许 232 串口对外 CTL 的电平状态
当 Data[3] = 0x01 的时候，允许 232 串口对外发送 CTL 的电平状态

注意：对于 RMDS-101 和 RMDS-103 版本的驱动器，因为其无 CTL 端口，所以请将 Data[3]=0x00 即可。

十三、数据反馈：

以下是驱动器对外发送电流、速度、位置等信息的 RS232 消息的格式。
需要特别注意，这条消息是由驱动器发出，发出的周期可以通过上述的配置指令来确定。

Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]	Data[8]	Data[9]
0x23	0x0B	待接收	待接收	待接收	待接收	待接收	待接收	待接收	待接收

举例：
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 RS232 消息：

```
Data[0] = 0x23;
Data[1] = 0x0B;
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 real_current  = (Data[0]<<8)|Data[1];
int16 real_velocity = (Data[2]<<8)|Data[3];
int32 real_position = (Data[4]<<24)| (Data[5]<<16)| (Data[6]<<8)| Data[7];
```

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、左右限位反馈：

以下是驱动器检测到 CTL1、CTL2 的电平发生变化后对外发送 RS232 数据包的格式。
特别注意，这条 RS232 消息是由驱动器发出，需要满足三个条件，驱动器才会对外发出该数据包
1. 在“RoboModule 直流伺服电机驱动器调试软件”上配置好 CTL1、CTL2 作为左右限位功能。
2. 通过 RS232 配置指令，打开 CTL1、CTL2 发送功能
3. CTL1、CTL2 接口上的电平发生变化时。

Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]	Data[8]	Data[9]
0x23	0x0C	待接收	待接收	0x55	0x55	0x55	0x55	0x55	0x55

举例：

当 CTL1 变化后的电平为低电平时候，Data[2] = 0x00，为高电平时候，Data[2] = 0x01
当 CTL2 变化后的电平为低电平时候，Data[3] = 0x00，为高电平时候，Data[3] = 0x01

注意：本段的 CTL1、CTL2 电平报告，仅仅只对 RMDS-102 版本的驱动器有实际意义，RMDS-101 和 RMDS-103 版本的驱动器无此接口，则反馈的 CTL1 和 CTL2 电平无实际意义。