

CAN 协议

硬件设定

ODrive 假定 CAN PHY 是线性总线配置中的标准差分双绞线，其两端均具有 120 欧姆的终端电阻。小于 V3.5 的 ODrive 版本包括一个焊接的 120 欧姆终端电阻，但是 03.5 和更高版本的 ODrive 则带有一个拨码开关来切换终端。ODrive 使用 3.3v 作为高输出，但符合 CAN PHY 的要求，即差分电压 > 1.5V 代表 “0”。因此，它与标准 5V 总线架构兼容。

ODrive 当前支持以下 CAN 波特率：

- 125 kbps
- 250 kbps（默认）
- 500 kbps
- 1000 kbps

传输协议

我们已经实现了一个非常基本的 CAN 协议，我们将其称为 “CAN Simple”，以使用户可以用 CAN 控制 ODrive。CAN 协议已经足够抽象，因此将来可以直接添加其他协议，例如 CANopen，J1939 或基于 ISO-TP 的光纤。不幸的是，实现这些协议是一项繁重的工作，我们希望为用户提供一种尽快通过 CAN 控制 ODrive 基本功能的方法。

CAN 框架

简单来说，CAN Simple 框架如下所示：

- 高 6 位-节点 ID-最大值 0x3F（或使用扩展 CAN ID 时为 0xFFFFF）
- 低 5 位-命令 ID-最大值 0x1F

为了解节点 ID 和命令 ID 的交互方式，我们来看一个示例：

```
odrv0.axis0.can_node_id = 0x010-保存信息 0x200 至 0x21F
odrv0.axis1.can_node_id = 0x018-保存信息 0x300 至 0x31F
```

可能并不明显，但这可以与 CANopen 兼容。尽管地址空间 0x200 和 0x300 对应于接收 PDO 基本地址，但是我们可以保证，如果所有 CANopen 节点 ID >= 32，它们都不会冲突。例如：

CANopen nodeID = 35 = 0x23

接收 PDO 0x200 + nodeID = 0x223，这与范围[0x200: 0x21F]不冲突。

请注意，不要为每个 PDO 组分配太多的 nodeID。四个 CAN Simple 节点（32 * 4）是单个 PDO 的所有可用地址空间。如果总线都是 ODrive CAN Simple 节点，则简单的顺序节点 ID 分配就可以正常工作。

主要内容

命令 ID	名称	信号	起始字节	信号类型	位	数量	补位	字节顺序
0x000	CANopen NMT 消息**	-	-	-	-	-	-	-
0x001	ODrive 心跳消息	Axis Error	0	Unsigned Int	32	1	0	Intel
		Axis Current State	4	Unsigned Int	32	1	0	Intel
0x002	ODrive 急停消息	-	-	-	-	-	-	-
0x003	获取电机错误*	Motor Error	0	Unsigned Int	32	1	0	Intel
0x004	获取编码器错误*	Encoder Error	0	Unsigned Int	32	1	0	Intel
0x005	获取无传感器错误*	Sensorless Error	0	Unsigned Int	32	1	0	Intel
0x006	设置 Axis 节点 ID	Axis CAN Node ID	0	Unsigned Int	32	1	0	Intel
0x007	设置 Axis 请求状态	Axis Requested State	0	Unsigned Int	32	1	0	Intel
0x008	设置 Axis 启动配置	- Not yet implemented -	-	-	-	-	-	-
0x009	获取编码器估算值*	Encoder Pos Estimate	0	IEEE 754 Float	32	1	0	Intel
		Encoder Vel Estimate	4	IEEE 754 Float	32	1	0	Intel
0x00A	获取编码器计数*	Encoder Shadow Count	0	Signed Int	32	1	0	Intel

命令 ID	名称	信号	起始字节	信号类型	位	数量	补位	字节顺序
		Encoder Count in CPR	4	Signed Int	32	1	0	Intel
0x00B	设置控制器模式	Control Mode	0	Signed Int	32	1	0	Intel
		Input Mode	4	Signed Int	32	1	0	Intel
0x00C	设定输入位置	Input Pos	0	IEEE 754 Float	32	1	0	Intel
		Vel FF	4	Signed Int	16	0.001	0	Intel
		Torque FF	6	Signed Int	16	0.001	0	Intel
0x00D	设定输入速度	Input Vel	0 4	IEEE 754 Float	32	1	0	Intel
		Torque FF	4	IEEE 754 Float	32	1	0	Intel
0x00E	设定输入扭矩	Input Torque	0	IEEE 754 Float	32	1	0	Intel
0x00F	设定速度极限	Velocity Limit	0	IEEE 754 Float	32	1	0	Intel
0x010	开启防嵌齿	-	-	-	-	-	-	-
0x011	设置梯形轨迹模式速度限值	Traj Vel Limit	0	IEEE 754 Float	32	1	0	Intel
0x012	设置梯形轨迹模式加速度极限	Traj Accel Limit	0	IEEE 754 Float	32	1	0	Intel
		Traj Decel Limit	4	IEEE 754 Float	32	1	0	Intel
0x013	设置梯形轨迹模式惯性	Traj Inertia	0	IEEE 754 Float	32	1	0	Intel

命令 ID	名称	信号	起始字节	信号类型	位	数量	补位	字节顺序
0x014	获取 IQ*	Iq Setpoint	0	IEEE 754 Float	32	1	0	Intel
		Iq Measured	4	IEEE 754 Float	32	1	0	Intel
0x015	获取无传感器估算值*	Sensorless Pos Estimate	0	IEEE 754 Float	32	1	0	Intel
		Sensorless Vel Estimate	4	IEEE 754 Float	32	1	0	Intel
0x016	重新启动 ODrive ***	-	-	-	-	-	-	-
0x017	获取 Vbus 电压 ***	Vbus Voltage	0	IEEE 754 Float	32	1	0	Intel
0x018	清除错误	-	-	-	-	-	-	-
0x700	CANOpen 心跳消息**	-	-	-	-	-	-	-

*注意：这些消息是呼叫和回复。主节点发送一条带有 RTR 位置 1 的消息，并且 Axis 以相同的 ID 和指定的有效负载进行响应。

**注意：保留这些 CANOpen 消息是为了避免总线与 CANOpen 设备发生冲突。CAN Simple 不使用它们。

***注意：这些消息可以发送到给定 ODrive 板上的任一地址。

为 CAN 配置 ODrive

将设备放置在总线上之前，应通过 USB 完成 CAN 参数的配置。

要设置所需的波特率，请使用 `<odrv>.can.set_baud_rate(<value>)`。无需重新启动设备即可完成波特率设置。如果您想保存波特率，只需在重启前保存设置 `<odrv>.save_configuration()`。

每个 Axis 看起来像总线上的一个单独节点。因此，它们都具有两个属性 `can_node_id` 和 `can_node_id_extended`。节点 ID 可以为 0 到 63 (0x3F) (含)，如果使用扩展的 CAN ID，则可以为 0 到 16777215 (0xFFFFFFFF)。如果要在 CAN 总线上连接多个 ODrive，则必须为第二个 ODrive 设置不同的节点 ID，否则它们将发生冲突并使总线崩溃。

配置示例

```
odrv0.axis0.config.can_node_id = 3
```

```
odrv0.axis1.config.can_node_id = 1
```

```
odrv0.can.set_baud_rate(250000)
```

```
odrv0.save_configuration()
```

```
odrv0.reboot()
```

