大然机器人无刷伺服驱动器
CAN 通讯协议

DrEmpower

大然机器人

Daran Robot

Dr Sci-Tech
Daran Robot.

目录

大然机器人

Daran Robot

第一章　电机驱动板 CAN 通讯协议概要

电机通过 CAN 组网时允许以并联方式连接到 CAN 上。网络中的每个电机均有独立的 ID 号（可自行设置、默认为 0）。电机会响应自己 ID 的指令。电机通讯使用 CAN 协议，默认 1Mbps 波特率。（还可支持 125 kbps，250 kbps，500 kbps，可自行设置）

硬件部分：

设备假定 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。硬件端包括一个焊接的 120 欧姆终端电阻通过内部跳接线是否短接来确定是否连接。CAN 端口使用 3.3v 作为高输出，符合 CAN PHY 的要求，即差分电压>1.5V 代表“0”，与标准 5V 总线架构兼容。

电机协议解析

1.1　标准控制 CAN 包（标准帧）组成

参数	帧信息	帧 ID 高位	帧 ID 低位	帧数据
字长	8 位	8 位	8 位	64 位

表 1-1

指令参数含义解释：

帧信息　　其值为帧数据的字节数，在该 CAN 协议中帧数据永远为 64 位，即帧信息永远为 8。

帧 ID 高位　和帧低位共同构成 CAN 包 ID 识别号，只有 ID 正确对应电机才会执行对应动作，否则数据将被丢弃。

帧 ID 低位　和帧高位共同构成 CAN 包 ID 识别号，只有 ID 正确对应电机才会执行对应动作，否则数据将被丢弃。

帧数据　　64 位数据，用于在设备交换数据。

电机 ID 及命令 ID 在帧 ID 中分布：

帧 ID 高位							
\	\	\	\	\	电机	电机	电机

					ID	ID	ID
帧 ID 低位							
电机 ID	电机 ID	电机 ID	命令 ID	命令 ID	命令 ID	命令 ID	命令 ID

表 1-2

CAN 包中帧 ID 由电机 ID 和命令 ID 共同构成，电机 ID 指定需要响应该条指令的电机，而命令 ID 决定电机具体执行什么命令。列如：一个电机的 ID 为 3，其会响应的 CAN 包为帧 ID 为 0b000011*****的 CAN 包（*代表任意值），而其具体执行的命令将根据*****的不同来执行。如果你发送读取命令，那么返回的数据包帧 ID 将与发送的相同（实时状态返回指令除外）。

例如我们需要控制 3 号电机急停，则对应的帧 ID 计算过程如下：

(1) 电机 ID = 0x03;

(2) 查询《命令 ID 查询表》得到急停命令位于系统控制指令下，其命令 ID = 0x008；

(3) 将命令 ID 与(1)得到的帧 ID 相加可得

帧 ID = 电机 ID<<5 + 命令 ID = 0x60 + 0x008 = 0x68

(4) 此时得到的数为 8 位，将其变换为帧 ID 的 16 位 0x0068，其中 0x68 为帧 ID 低位，0x00 位帧 ID 高位。

帧数据排序逻辑：

帧 ID 低位	帧数据		
8 位	第 1 位	…	第 64 位

帧数据段长度为 64 位，靠近帧 ID 的为低位，末尾为最高位，符合 Intel 排序方法。如果指令中的数据为不足 64 位，则空余位用 0 补齐。

查询《命令 ID 查询表》可知，系统控制指令对应的帧数据段有两个数据，分别为系统命令编号 sys_num 和系统命令数据 sys_data，其中急停命令对应的 sys_num=0x06, sys_data = 0x00。则完整的急停控制 CAN 包为

帧信息	帧 ID 高位	帧 ID 低位	data1	data2	data3	data4	data5	data6	data7	data8
0x08	0x00	0x68	0x06	0x00	0x00	0x00	0x00	0x00	0x00	0x00
			sys_num = 0x06 = 0x00000006				sys_data=0x00=0x0000			

1.2 《命令 ID 查询表》

命令 ID	帧型及有无返回	参数名称		起始字节	数据类型	位	数量	字节顺序
0x019	标准帧	目标位置	T 轨迹跟踪模式	0	IEEE 754 float	32	1	intel
		目标速度		4	Signed short	16	0.01	
		角度输入滤波带宽		6	Signed short	16	0.01	
0x01A	标准帧	目标位置	Q 梯形轨迹模式	0	IEEE 754 float	32	1	intel
		目标速度		4	Signed short	16	0.01	
		目标加速度		6	Signed short	16	0.01	
0x01B	标准帧	目标位置	P 前馈控制模式	0	IEEE 754 float	32	1	intel
		速度前馈		4	Signed short	16	0. 01	
		扭矩前馈		6	Signed short	16	0. 01	
0x01C	标准帧	目标转速	V 速度控制指令	0	IEEE 754 float	32	1	intel
		模式参数		4	Signed short	16	0. 01	
		目标模式		6	unsigned short	16	1	
速度控制指令的目标模式用来指定速度控制模式， 1 表示速度前馈模式，此时模式参数为前馈扭矩（Nm） 2 表示速度爬升模式，此时模式参数为速度爬升速率（(r/min)/s）								
0x01D	标准帧	目标扭矩	C 扭矩控制指令	0	IEEE 754 float	32	1	intel
		模式参数		4	Signed short	16	0. 01	

		目标模式		6	unsigned short	16	1	
扭矩控制指令的目标模式用来指定扭矩控制模式， 1 表示直接控制模式，此时模式参数无作用 6 表示扭矩爬升模式，此时模式参数为扭矩爬升速率（Nm/s）								
0x00C	标准帧	参数 1	预设参数指令（多个电机）	0	IEEE 754 float	32	1	intel
		参数 2		4	Signed short	16	0.01	
		参数 3		6	Signed short	16	0.01	
参数预设指令中的参数 1~3 具体由**起始指令**（系统指令命令 ID=0x008）类型 sys_num 决定								
sys_num		参数 1	参数 2			参数 3		
0x10		目标位置	目标速度（无作用）			目标角度输入滤波带宽		
0x11		目标位置	目标时间			目标加速度		
0x12		目标位置	前馈速度			前馈扭矩		
0x13		目标速度	模式参数			目标模式		
0x14		目标扭矩	模式参数			目标模式		
0x15		目标位置	目标速度			前馈扭矩		
0x008	标准帧	系统命令编号 sys_num	系统控制指令	0	Unsigned Int	32	1	intel
		系统命令数据 sys_data1		4	Signed short	16	0.01	intel
		系统命令数据 sys_data2		6	Signed short	16	0.01	intel

系统命令编号 sys_num 对应表：

sys_num = 0x00　恢复初始配置 init_config； sys_num = 0x01　保存配置 save_config； sys_num = 0x02　擦除配置 erase_config； sys_num = 0x03　电机重启 reboot； sys_num = 0x04　清除错误 clear_errors； sys_num = 0x05　设置零点 set_zero_position； sys_num = 0x06　电机急停 estop； sys_num = 0x07　进入 dfu 升级模式 enter_dfu_mode；	sys_num = 0x10　t 模式起始指令 t_start_move； sys_num = 0x11　q 模式起始指令 q_start_move； sys_num = 0x12　p 模式起始指令 p_start_move； sys_num = 0x13　v 模式起始指令 v_start_move； sys_num = 0x14　c 模式起始指令 c_start_move； sys_num = 0x15　d 阻抗控制模式起始指令 d_start_move **注意：** **起始指令需与参数预设指令（命令 ID=0x00C）配合使用。**

系统命令数据 sys_data 对应表：								
sys_num= 0x10~0x12 (角度控制起始指令：库函数中将此处数据类型定义成了 u16) sys_data1 = 0, sys_data2 = 0 表示绝对角度，目标角度相对于输出轴零点 sys_data1 = 1, sys_data2 = 0 表示相对角度，目标角度相对于输出轴当前位置 sys_num= 0x15 (阻抗控制起始指令) sys_data1: 刚度系数 Kp (Nm/rad) sys_data2: 阻尼系数 Kd (Nm/rad/s) 注意此处的刚度和阻尼系数单位借鉴了 MIT 方案，但是指令中的位置、速度单位不是 rad 及 rad/s，而是 degree 和 r/min。 sys_num = 其他值，sys_data1 = 0，sys_data2=0 无系统命令数据。								
0x01E	标准帧有返回	Address <u16>	参数读取指令	0	unsigned short	16	1	intel
		Data_type <u16>		2	unsigned short	16	1	
0x01F	标准帧	Address <u16>	参数设置指令	0	unsigned short	16	1	intel
		Data_type <u16>		2	unsigned short	16	1	
		目标值 Value		4	根据 Data_type 确定	16 或 32	1	

表 1-3

1、数量为具体输入到电机控制器的量，1 为等量输入，最终输入值=数据帧中的值，0.01 表示最终输入值=数据帧中值/100, 0.001 表示最终输入值=数据帧中的值/1000（因为 CAN 标准帧中数据长度只有 64 位，如果需要同时传输三个浮点数，后两位需要通过 16 位 signed int 进行传输）

2、参数读取指令（命令 ID=0x01E）数据格式

data1	data2	data3	data4	data5	data6	data7	data8
Address<u16>		Data_type<u16>		0x00	0x00	0x00	0x00

注意：CAN 远程帧不能传递数据，所以参数读取指令也要用数据帧进行发送。

3、参数设置指令（命令 ID=0x01F）及参数读取指令电机返回的数据格式

data1	data2	data3	data4	data5	data6	data7	data8
Address<u16>		Data_type<u16>		Vlaue<类型和长度由 Data_type 确定，低位在前>			

4、Address: 参数地址见《常用参数及其地址表》,例如查表可得

电机 ID 号(axis.config.can_node_id)对应的地址 Address = 31001

CAN 通信波特率(can.config.baud_rate)对应的 Address = 21001

5、Data_type：数据类型，有以下 5 种数据类型，分别对应 0-4

Data_type=0, 浮点数（float）,数据长度 32 位,符号‘f’;

Data_type=1, 无符号短整数（unsigned short int）,数据长度 16 位,符号‘u16’;

Data_type=2, 有符号短整数（short int）,数据长度 16 位,符号‘s16’;

Data_type=3, 无符号短整数（unsigned int）,数据长度 32 位,符号‘u32’;

Data_type=4, 有符号短整数（int）,数据长度 32 位,符号‘s32’;

示例：读取 4 号电机输出轴当前输出轴位置（axis0.output_shaft.pos_estimate）

帧信息：0x08

帧 ID： 电机 ID 为 0x04，向左移动 5 位空出命令 ID 位，此时帧 ID 为 0x80，查询《命令 ID 查询表》得到读取参数指令命令 ID=0x01E，将命令 ID 与刚刚得到的帧 ID 相加得到 0x9E，此时得到的数为 8 位，将其变换为帧 ID 的 16 位 0x009E，其中 9E 为帧 ID 低位，00 位帧 ID 高位

帧数据：查《命令 ID 查询表》可知，读取参数指令数据帧有两个参数 Address 和 Data_type，查询《常用参数及其地址》表 axis0.output_shaft.pos_estimate 对应的 Address = 38007 = 0x9477，数据类型为 float32, 则 Data_type = 0 = 0x0000。

最后得到的 CAN 包从包头到包尾为：

帧信息	帧 ID 高位	帧 ID 低位	data1	data2	data3	data4	data5	data6	data7	data8
0x08	0x00	0x9E	0x77	0x94	0x00	0x00	0x00	0x00	0x00	0x00
			Address unsigned short		Data_type unsigned short					

电机返回的 CAN 包如下：

帧信息	ID 高位	ID 低位	data1	data2	data3	data4	data5	data6	data7	data8
0x08	0x00	0x9E	0x77	0x94	0x00	0x00	0x38	0xF0	0xB8	0xC2
			Address unsigned short		Data_type unsigned short		返回值 Value float32			

电机返回的 CAN 包中 data5~data8 为电机输出轴当前输出轴位置（axis0.output_shaft.pos_estimate），其数据类型为 float32，则先将数据取出， Pos Estimate 为 0xC2B8F038，经过转换后为-92.47 度，至此通讯结束。

设置参数指令时帧数据和读取返回值是一致的，首先查表确定其位于数据包的起始字节，再将需要写入数据的每个字节倒序取出，从数据帧起始字节低位开始放置，例如目标值为 unsigned int 数 0x 324B550A，起始字节为 4，则数据包应该写为：

data1	data2	data3	data4	data5	data6	data7	data8
*	*	*	*	0x0A	0x55	0x4B	0x32

特别需要注意的是，对于数据类型是 signed int 的数据负数是以补码进行传输。

6、实时状态返回指令数据格式

帧 ID 低位								data1	data2	data3	data4	data5	data6	data7	data8
8	7	6	5	4	3	2	1	0x77	0x94	0x00	0x00	0x38	0xF0	0xB8	0xC2
			*	*	E	T	1	当前位置（°） Float32				当前速度（r/min） Signed short-0.01		当前扭矩（Nm） Signed short-0.01	

实时状态返回是指在 axis0.config.extra_setting.enable_reply_state 置为 1 后，关节电机在接收到控制指令（命令 ID= 0x019~0x01D）或预设指令（命令 ID=0x00C）后，会自动立即回复一条电机实时状态指令，该指令里包含了关节电机的当前位置、速度、扭矩信息，同时在指令的命令 ID 中加入了一些关节电机状态信息，包括 T-是否到达目标位置（axis0.controller.trajectory_done）、E-是否报错（axis0.error），在命令 ID 的最低位被置位 1，用于区分正常参数读取指令的返回值（命令 ID=0x01E,最低位为 0）；

7、快速读取指令数据格式

除此之外，正常情况下由于参数读取指令每次只能读取一个变量，如果需要同时读取关节电机的实时位置、速度、扭矩信息（很多实时控制场景需要），则需要 3 条读取指令，导致通信效率很低；为此在固件版本号>1.1 的版本中增加了一条快速读取指令接口。

该指令仍采用参数读取指令接口（命令 ID=0x01E），只是其 Address 比较特殊为 0x00（正常读取指令的 Address 都不为零），Data_type 也为 0x00；完整的指令如下所示：

帧信息	帧 ID 高位	帧 ID 低位	data1	data2	data3	data4	data5	data6	data7	data8
0x 08	((电机 ID<<5 +0x1E)>>8) &0xFF	(电机 ID<<5 +0x1E) &0xFF	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
			Address unsigned short		Data_type unsigned short					

该指令的返回值就是 6 中的实时状态返回指令，其中包括了实时位置、速度、扭矩信息及是否到达目标位置、电机是否报错等信息。实时位置信息位于返回指令数据段前 4 位，数据类型为浮点数(Float32)；实时速度和扭矩位于返回指令数据段的后 4 位，分别占用 2 个字节，数据类型为有符号短整型（Signed short），数量为 0.01；是否到达目标位置（T）位于返回指令命令 ID 倒数第 2 位，1 表示已到达，0 表示未到达；电机是否报错（E）位于返回指令命令 ID 倒数第 3 位，1 表示关节电机报错，状态异常，0 表示状态正常；

第二章 CAN 通讯协议示例

2.1 控制指定编号电机转动到指定角度

控制 ID 为 1 的以轨迹跟踪模式转动到 90°位置，同时指定角度输入滤波带宽为 10。

CAN 端口发送数据：

0x08 0x00 0x39 0x00 0x00 0xB4 0x42 0x00 0x00 0xE8 0x03

0x08 为数据头

0x00 0x39 为电机 ID 与命令 ID 组合：1 的 16 进制为 0x01，向左移动 5 位空出命令 ID 位，此时帧 ID 为 0x20，查询《命令 ID 查询表》得到轨迹跟踪模式命令 ID 为 0x019，将命令 ID 与刚刚得到的帧 ID 相加得到 0x39，此时得到的数为 8 位，将其变换为帧 ID 的 16 位 0x0039，其中 39 为帧 ID 低位，00 为帧 ID 高位

0x00 0x00 0xB4 0x42 0x00 0x00 0xE8 0x03 的由来：

轨迹跟踪模式下有三个参数，分别为目标位置，目标速度(无作用)，角度输入滤波带宽。其中角度输入滤波带宽参数数量为 0.01，则传输过程中需要换算为 10/0.01=1000。所以最终传输的三个数分别为 90,0,1000。

再利用库函数中的 format_data([90,0,1000],’f s16 s16’,’encode’)转换得到最终的 16 进制：[0x00 0x00 0xB4 0x42 0x00 0x00 0xE8 0x03]。

（换算函数 format_data 见章末）

最终完整的 CAN 包为

帧信息	帧 ID 高位	帧 ID 低位	data1	data2	data3	data4	data5	data6	data7	data8
0x08	0x00	0x39	0x00	0x00	0xB4	0x42	0x00	0x00	0xE8	0x03
			目标位置: float32 数量：1				目标速度: short int 数量：0.01		角度输入滤波带宽： short int 数量：0.01	

控制 1 号电机以梯形轨迹模式转动到-150 度位置，加速度为 5((r/min)/s)，最大速度为 10(r/min)。

CAN 端口发送数据：

0x08 0x00 0x3a 0x00 0x00 0x16 0xC3 0xE8 0x03 0xF4 0x01

0x08 为数据头

0x00 0x3a 为电机 ID 与命令 ID 组合：1 的 16 进制为 0x01，向左移动 5 位空出命令 ID 位，此时帧 ID 为 0x20，查询表格得到梯形轨迹模式消息的命令 ID 为 0x01a，将命令 ID 与刚刚得到的帧 ID 相加得到 0x3a，此时得到的数为 8 位，将其变换为帧 ID 的 16 位 0x003a，其中 3a 为帧 ID 低位，00 为帧 ID 高位

0x00 0x00 0x16 0xC3 0xE8 0x03 0xF4 0x01 的由来：

梯形轨迹模式下有三个参数，分别为目标位置，目标速度及目标加速度。其中目标速度和目标加速度的数量为 0.01，则传输过程中需将速度换算为 10/0.001=1000, 5/0.001=500。所以最终传输的三个数为-150,1000,500。

再利用库函数中的 format_data9([-150,1000,500],'f s16 s16','encode')转换为 16 进制：[0x00 0x00 0x16 0xC3 0xE8 0x03 0xF4 0x01]。

最终完整的 CAN 包为

帧信息	帧 ID 高位	帧 ID 低位	data1	data2	data3	data4	data5	data6	data7	data8
0x08	0x00	0x3a	0x00	0x00	0x16	0xC3	0xE8	0x03	0xF4	0x01
			目标位置: float32 数量：1				目标速度: short int 数量：0.01		目标加速度： short int 数量：0.01	

2.2 控制指定电机以指定速度运行

控制 1 号电机以速度前馈模式以速度 20(r/min)转动，前馈扭矩为 1.5Nm

CAN 端口发送数据：

0x08 0x00 0x3c 0x00 0x0 0xA0 0x41 0x96 0x00 0x01 0x00

0x08 为数据头

0x00 0x3c 为电机 ID 与命令 ID 组合：1 的 16 进制为 0x01，向左移动 5 位空出命令 ID 位，此时帧 ID 为 0x20，查询表格得到电机速度指令消息的命令 ID 为 0x01c，将命令 ID 与刚刚得到的帧 ID 相加得到 0x3c，此时得到的数为 8 位，将其变换为帧 ID 的 16 位 0x003c，其中 3c 为帧 ID 低位，00 为帧 ID 高位

0x00 0x0 0xA0 0x41 0x96 0x00 0x01 0x00 的由来：

速度控制指令有三个参数，分别为目标速度，模式参数及目标模式。其中速度前馈模式对应的目标模式为 1，对应的模式参数为前馈扭矩，数量为 0.01，则传输过程中需将前馈扭矩换算为 1.5/0.001=150。

再利用库函数中的 format_data([20,150,1], 'f s16 u16', 'encode')转换为 16 进制：[0x00 0x00 0xA0 0x41 0x96 0x00 0x01 0x00]。

最终完整的 CAN 包为

帧信息	帧 ID 高位	帧 ID 低位	data1	data2	data3	data4	data5	data6	data7	data8
0x08	0x00	0x3c	0x00	0x00	0xA0	0x41	0x96	0x00	0x01	0x00
			目标速度: float32 数量：1				前馈扭矩: short int 数量：0.01		目标模式： unsigned short int 数量：1	

控制 1 号电机以速度爬升模式以速度爬升速率（加速度）5((r/min)/s)加速至 20(r/s)转动

CAN 端口发送数据：

0x08 0x00 0x3C 0x00 0x00 0xA0 0x41 0xf4 0x01 0x02 0x00

0x08 为数据头

0x00 0x3C 为电机 ID 与命令 ID 组合：1 的 16 进制为 0x01，向左移动 5 位空出命令 ID 位，此时帧 ID 为 0x20，查询表格得到电机速度控制指令命令 ID 为 0x01c，将命令 ID 与刚刚得到的帧 ID 相加得到 0x3c，此时得到的数为 8 位，将其变换为帧 ID 的 16 位 0x003c，其中 3c 为帧 ID 低位，00 为帧 ID 高位

0x00 0x00 0xA0 0x41 0xf4 0x01 0x02 0x00 的由来：

速度控制指令有三个参数，分别为目标速度，模式参数及目标模式。其中速度爬升模式对应的目标模式为 2，对应的模式参数为速度爬升速率，数量为 0.01，则传输过程中需将速度爬升速率换算为 5/0.001=500。

再利用库函数中的 format_data([20,500,2],'f s16 u16', 'encode')转换为 16 进制：[0x00 0x00 0xA0 0x41 0xf4 0x01 0x02 0x00]。

最终完整的 CAN 包为

帧信息	帧 ID 高位	帧 ID 低位	data1	data2	data3	data4	data5	data6	data7	data8

0x08	0x00	0x3c	0x00	0x00	0xA0	0x41	0xF4	0x01	0x02	0x00
			目标速度: float32 数量：1				速度爬升速率: short int 数量：0.01		目标模式： unsigned short int 数量：1	

2.3 设置电机 ID 号

设置电机 ID 号，将 1 号电机的 ID 号(axis0.config.can_node_id)修改为 5

CAN 端口发送数据：

0x08 0x00 0x3f 0x19 0x79 0x03 0x00 0x05 0x00 0x00 0x00

0x08 为数据头

0x00 0x3f 为电机 ID 与命令 ID 组合：电机 ID 号为 1，向左移动 5 位空出命令 ID 位，此时帧 ID 为 0x20，查询《命令 ID 查询表》得到参数设置指令的命令 ID 为 0x01f，将命令 ID 与刚刚得到的帧 ID 相加得到 0x3f，此时得到的数为 8 位，将其变换为帧 ID 的 16 位 0x003f，其中 3f 为帧 ID 低位，00 为帧 ID 高位

0x19 0x79 0x03 0x00 0x05 0x00 0x00 0x00 的由来：

查询《常用参数及其地址》表 axis0.config.can_node_id 对应的 Address = 31001 = 0x7919，数据类型为 uint32，则 Data_type = 3 = 0x0003。参数设置指令的第三个参数为目标值 Value，这里 Value=5。

再利用库函数中的 format_data([31001,3,5],'u16 u16 u32', 'encode')转换为 16 进制[0x19 0x79 0x03 0x00 0x05 0x00 0x00 0x00]。

最终完整的 CAN 为

帧信息	帧 ID 高位	帧 ID 低位	data1	data2	data3	data4	data5	data6	data7	data8
0x08	0x00	0x3F	0x19	0x79	0x03	0x00	0x05	0x00	0x00	0x00
			Address u16		Data_type u16		Value u32			

2.4 函数 format_data()

函数使用说明：

该函数位于 yf_14_can.py 库函数文件中；

函数原型：def format_data(data=[], format="f f", type='decode')

调用函数时：

若使用数据转为 byte 类型，在 data 处写入 float，int，unsigned int，short，unsigned short，用逗号“，”隔开，在 format 处输入对应类型名：’f，s32，u32，s16，u16’，多个数据时用空格“ ”隔开，在 type 处输入’encode’；（建议只输入两个数据）

若使用函数将 byte 类型转为普通数据类型，在 data 处输入二进制数，用逗号“，”隔开，在 format 处输入需要转化的类型的对应名，分别为：’f，s32，u32，s16，u16’，多个数据时，用空格“ ”隔开，在 type 处输入’decode’。（data 只能输入八位）

其中，s16、u16 只占用两位 byte 位，f、s32、u32 占用四位。