

1. 协议解析

1.1 CAN-FD 相关说明

1. CAN-FD 波特率：

- 仲裁段：1 Mbps
- 数据段：最高支持 5 Mbps，也可用 1Mbps（可变波特率）。

2. ID：由 16 位构成，其中 0x7F 是广播地址。

- 高 8 位：表示**源地址**：
 - 最高位为 1：需要回复，相当于一个**总开关**，开启但不发生查询指令会返回一个数据段长度为 0 的帧。
 - 最高位为 0：无需回复。
 - 其余 7 位：信号源地址。
- 低 8 位：表示**目的地址**：
 - 最高为 0。
 - 其余 7 位表示目的地址。

例如：

1. ID: 0x8001

- 信号源地址为 0。
- 目的地址为 1。
- 最高位为 1，表示需要回复，即打开回复**总开关**。

2. ID: 0x100

- 信号源地址为 1。
- 目的地址为 0。
- 最高位为 0，表示无需回复，即关闭回复**总开关**。

1.2 协议基本说明

1. 最小单位为子帧，每条子帧可以向**一个或多个**寄存器写入值，或读取数据。
2. 一条 CAN-FD 帧可由**一个或多个**子帧组成。
3. 电机各个功能实现通过在一条 FDCAN 帧中写入**一个或多个**对应功能的寄存器的值实现。
4. 可向任意寄存器写入 `int8_t`、`int16_t`、`int32_t`、`float` 四种基本数据类型。
5. 在**同一子帧**中的数据类型必需相同，**不同子帧**中的数据类型可以不同。
6. 所有基本类型的传输都为**小端模式**，即先发送低字节的数据，再发送高字节的数据。
7. 在一条 CAN-FD 中需要在**尾部填充**字节是应使用 `0x50`，表示无任何操作（NOP）。
8. 各种数据类型的**无限制**：
 - `int8_t`： `0x80`
 - `int16_t`： `0x8000`
 - `int32_t`： `0x80000000`
 - `float`： `NAN`
9. 模式三（一拖多模式）用 FDCAN 的 ID 来识别。
 - ID 低 8 位大于等于 `0x80` 时，为模式三（一拖多模式）。

- ID 低 8 位小于 0x80 时，为模式一或模式而（普通模式）。

1.3 子帧解析

1.3.1 发送协议

1.3.1.1 模式一

```
uint8_t tdata[] = {cmd, addr, a1, a2, b1, b2...};
```

1. cmd：表示读写、数据类型和个数。

1. 高四位 cmd[7:4] 表示读、写、回读。

1. 0000：写
2. 0001：读
3. 0010：回复

2. 低四位 cmd[3:0] 表示数据类型和个数：

1. 高两位 cmd[3:2] 表示数据类型：

1. 00：int8_t
2. 01：int16_t
3. 10：int32_t
4. 11：float

2. 低两位 cmd[1:0] 表示数据个数。

1. 01：一个数据。
2. 01：两个数据。
3. 11：三个数据。
4. 00：模式二标志。

2. addr：起始寄存器地址。

3. a1, a2, b1, b2...：向寄存器中写入的数据，注意：需满足 1.2 协议基本说明的 4、5 项。

1. a1, a2：需要向 addr 中写入的数据。
2. b1, b2：需要向 addr + 1 中写入的数据。
3. ...：需要向 add + n 中写入的数据。

1.3.1.2 模式二

```
uint8_t tdata[] = {cmd, num, addr, a1, a2, b1, b2...};
```

1. cmd：表示读写、数据类型和个数。

1. 高四位 cmd[7:4] 表示读、写、回读。

1. 0000：写
2. 0001：读
3. 0010：回复

2. 低四位 cmd[3:0] 表示数据类型和个数：

1. 高两位 cmd[3:2] 表示数据类型：

1. 00：int8_t
2. 01：int16_t
3. 10：int32_t

4. 11: float

2. 低两位 `cmd[1:0]` 固定为 00, 表示模式二, 后一字节表示数据个数。

2. `num`: 数据个数。

3. `addr`: 起始寄存器地址。

4. `a1, a2, b1, b2...`: 向寄存器中写入的数据, 注意: 需满足 1.2 协议基本说明的 4、5 项。

1. `a1, a2`: 需要向 `addr` 中写入的数据。

2. `b1, b2`: 需要向 `addr + 1` 中写入的数据。

3. `...`: 需要向 `addr + n` 中写入的数据。

模式二实质: 在模式一的 `xxx` 中数据个数写为 0 时, 使用后一字节表示数据个数, 其余字节都往后移动一位。

1.3.1.3 模式三 (一拖多模式)

- 模式三由 FDCAN-ID 的低 8 位决定, FDCAN-ID 大于 0x80 即为一拖多模式 (具体模式见例程)
- 协议格式为: 电机 1 数据、电机 2 数据、电机 3 数据... 最后 2 字节为查询电机状态代码。
- 电机 1、电机 2...: 指 id 为 1 的电机、id 为 2 的电机....
- 一拖多模式发送的数据类型均为 **int16 类型**。

例 1: ID 为 0x8090

```
uint8_t cmd[] = {pos11, pos12, val11, val12, tqe11, tqe12, pos21, pos22, val21, val22, tqe21, tqe22, 占位字节, 0x14, 0x04, 0x00}
```

- ID 为 0x8090 时, 为一拖多的位置、速度、力矩控制。
- 第 1 字节到第 6 字节分别为位置、速度、力矩控制。
- 第 7 字节到第 12 字节分别为位置、速度、力矩控制。
- 第 13 字节到第 18 字节分别为位置、速度、力矩控制。
- 以此类推....
- 最后 3 字节为查询电机状态的指令。

例 2: ID 为 0x8093

```
uint8_t cmd[] = {pos11, pos12, val11, val12, tqe11, tqe12, rkp11, rkp12, rkd11, rkd12, pos21, pos22, val21, val22, tqe21, tqe22, rkp21, rkp22, rkd21, rkd22, 占位字节, 0x14, 0x04, 0x00}
```

- ID 为 0x8093 时, 为一拖多的位置、速度、力矩、PD 控制。
- 第 1 字节到第 10 字节分别为位置、速度、力矩、PD 控制。
- 第 11 字节到第 20 字节分别为位置、速度、力矩、PD 控制。
- 第 21 字节到第 30 字节分别为位置、速度、力矩、PD 控制。
- 以此类推....
- 最后 3 字节为查询电机状态的指令。

1.3.2 接收协议

- 接受协议模式是由发送协议的模式决定的。
- 接收协议模式和发送协议模式是相同的。

1.4.2.1 模式一

假设获取的数据是 `int16_t`

```
uint8_t rdata[] = {cmd, addr, a1, a2, b1, b2, ..., cmd1, addr1, c1, c2, c3, c4}
```

- `cmd` :
 - 高四位 `cmd[7, 4]` : 0010 表示回复。
 - 2~3 位 `cmd[3, 2]` : 表示类型。
 - 00 : `int8_t` 类型。
 - 01 : `int16_t` 类型。
 - 10 : `int32_t` 类型。
 - 11 : `float` 类型。
 - 低 2 位 `cmd[1, 0]` : 表示数量。
 - 00 : 表示模式二。
 - 01 : 一个数据。
 - 10 : 两个数据。
 - 11 : 三个数据。
- `addr` : 开始获取的地址。
- `a1, a2` : 数据 1, 小端模式。
- `b1, b2` : 数据 2, 小端模式。

1.4.2.2 模式二

```
uint8_t rdata[] = {cmd, addr, num, a1, a2, b1, b2, ..., cmd1, addr1, c1, c2, c3, c4}
```

- `cmd` :
 - 高四位 `cmd[7, 4]` : 0010 表示回复。
 - 2~3 位 `cmd[3, 2]` : 表示类型。
 - 00 : `int8_t` 类型。
 - 01 : `int16_t` 类型。
 - 10 : `int32_t` 类型。
 - 11 : `float` 类型。
 - 低 2 位 `cmd[1, 0]` : 表示数量。
 - 00 : 表示模式二, 后一个字节表示数据数量。
- `num` : 数据个数。
- `addr` : 开始获取的地址。
- `a1, a2` : 数据 1, 小端模式。
- `b1, b2` : 数据 2, 小端模式。

1.3.3 示例

1.3.1 发送协议示例

1.4.1.1 模式一

```
uint8_t cmd[] = {0x01, 0x00, 0x0A, 0x0A, 0x20, 0x00, 0x00, 0x00, 0x80, 0x10,  
0x27, 0x00, 0x00, 0x50, 0x50, 0x50};
```

该 CAN-FD 帧由两个子帧构成：

1. 子帧 1：整体意思是电机进入位置模式。

- 0x01：第一个子帧的开头
 - 高四位为 0000，表示写操作。
 - 低四位：
 - 高 2 位为：00，表示 int8_t 类型。
 - 低 2 位为：01，表示 1 个数据。
- 0x00：起始寄存器地址：查表可知，0x00 寄存器表示电机模式设置。
- 0x0A：往 0x00 寄存器写入 0x0A。

2. 子帧 2：整体意思是位置不限制，速度为 0.1 转/秒

- 0x0A：第 2 个子帧的开头
 - 高 8 位为 0x0，表示写操作。
 - 低 8 位：
 - 高 2 位为：10，表示 int32_t 类型。
 - 低 2 位为：10，表示 2 个数据。
- 0x20：起始寄存器地址：查表可知，0x20 寄存器表示位置，0x21 寄存器表示速度。
- 0x00、0x00、0x00、0x80：小端模式，即 0x80000000 写入 0x20 寄存器，即表示电机位置无限制。
- 0x10、0x27、0x00、0x00：小端模式，即 0x2710 写入 0x21 寄存器，表示电机速度设置为 0.1 转/秒。

3. 0x50, 0x50, 0x50：由于 CAN-FD 发送字节数最近的两个是 12 和 16，固还需加上 3 个字节的占位字节。

1.4.1.2 模式二

```
uint8_t cmd[] = {0x01, 0x00, 0x0A, 0x08, 0x02, 0x20, 0x00, 0x00, 0x00, 0x80,  
0x10, 0x27, 0x00, 0x00, 0x50, 0x50};
```

该 CAN-FD 帧由两个子帧构成：

1. 子帧 1：整体意思是电机进入位置模式。

- 0x01：第一个子帧的开头
 - 高 8 位为 0001，表示写操作。
 - 低 8 位：
 - 高 2 位为：00，表示 int8_t 类型。
 - 低 2 位为：01，表示 1 个数据。

- 0x00：起始寄存器地址：查表可知，0x00 寄存器表示电机模式设置。
- 0x0A：往 0x00 寄存器写入 0x0A。

2. 子帧 2：整体意思是位置不限制，速度为 0.1 转/秒

- 0x08：第 2 个子帧的开头
 - 高 8 位为 0x0，表示写操作。
 - 低 8 位：
 - 高 2 位为：10，表示 int32_t 类型。
 - 低 2 位为：00，表示模式二，后一个字节表示数据数量。
- 0x02：2 个数据。
- 0x20：起始寄存器地址：查表可知，0x20 寄存器表示位置，0x21 寄存器表示速度。
- 0x00、0x00、0x00、0x80：小端模式，即 0x80000000 写入 0x20 寄存器，即表示电机位置无限制。
- 0x10、0x27、0x00、0x00：小端模式，即 0x2710 写入 0x21 寄存器，表示电机速度设置为 0.1 转/秒。

3. 0x50, 0x50：由于 CAN-FD 发送字节数最近的两个是 12 和 16，固还需加上 2 个字节的占位字节。

1.3.2 接收协议示例

- 接受协议模式是由发送协议的模式决定的。
- 接收协议模式和发送协议模式是相同的。

1.3.2.1 模式一

```
uint8_t rdata[] = {0x28, 0x04, 0x00, 0x0A, 0x00, 0x00, 0x00, 0x96, 0x1D, 0x04, 0x00, 0x54, 0x40, 0x02, 0x00, 0x86, 0x01, 0x00, 0x00, 0x50};
```

该帧由一个子帧构成：

1. 子帧 1：整体意思是电机进入位置模式。

- 0x28：
 - 高 8 位为 0010，表示回复操作。
 - 低 8 位：
 - 高 2 位为：10，表示 int32_t 类型。
 - 低 2 位为：00，表示模式二，后一个字节表示数据数量。
- 0x04：4 个数据。
- 0x00：起始寄存器地址
- 0x0A, 0x00, 0x00, 0x00：0x00 寄存器的值，查表可知为电机模式，十进制为 10，查表可知为位置模式。
- 0x96, 0x1D, 0x04, 0x00：小端模式，十进制为 269718，即电机当前位置是 2.69718 转处。
- 0x54, 0x40, 0x02, 0x00：小端模式，十进制为 147540，即当前电机速度为 1.4754 转/秒。
- 0x86, 0x01, 0x00, 0x00：小端模式，十进制为 390，即当前实际输出力矩为：0.0039 NM。

- 0x50：占位符。

由前三个字符可知：此 CAN-FD 是回复 0x18, 0x04, 0x00 的。

1.3.2.2 模式二

```
uint8_t rdata[] = {0x2B, 0x01, 0x0A, 0x00, 0x00, 0x00, 0x96, 0x1D, 0x04, 0x00, 0x54, 0x40, 0x02, 0x00, 0x86, 0x01, 0x00, 0x00, 0x50};
```

该帧由一个子帧构成：

1. 子帧 1：整体意思是电机进入位置模式。

- 0x2B：
 - 高 8 位为 0x2，表示回复操作。
 - 低 8 位：
 - 高 2 位为：10，表示 int32_t 类型。
 - 低 2 位为：11，表示 3 个数据。
- 0x01：起始寄存器地址
- 0x0A, 0x00, 0x00, 0x00：0x00 寄存器的值，查表可知为电机模式，十进制为 10，查表可知为位置模式。
- 0x96, 0x1D, 0x04, 0x00：小端模式，十进制为 269718，即电机当前位置是 2.69718 转处。
- 0x54, 0x40, 0x02, 0x00：小端模式，十进制为 147540，即当前电机速度为 1.4754 转/秒。
- 0x86, 0x01, 0x00, 0x00：小端模式，十进制为 390，即当前实际输出力矩为：0.0039 NM。
- 0x50：占位符。

由前三个字符可知：此 CAN-FD 是回复 0x1B, 0x01 的。

2. 常用类型（单位）说明

2.1 电流（A）

数据类型	LSB	实际 (A)
int8	1	1
int16	1	0.1
int32	1	0.001
float	1	1

2.2 电压 (V)

数据类型	LSB	实际 (V)
int8	1	0.5
int16	1	0.1
int32	1	0.001
float	1	1

2.3 扭矩 (Nm)

- 真实扭矩 = $k * tqe + d$

2.3.1 5046 扭矩 (Nm)

数据类型	斜率 (k)	偏移量 (d)
int16	0.005397	-0.455107
int32	0.000528	-0.414526
float	0.533654	-0.519366

2.3.2 4538 扭矩 (Nm)

数据类型	斜率 (k)	偏移量 (d)
int16	0.004587	-0.290788
int32	0.000445	-0.234668
float	0.493835	-0.473398

2.3.2 5047/6056 (双极, 36 减速比) 扭矩 (Nm)

数据类型	斜率 (k)	偏移量 (d)
int16	0.004563	-0.493257
int32	0.000462	-0.512253
float	0.465293	-0.554848

2.3.3 5047 (单极, 9 减速比) 扭矩 (Nm)

数据类型	斜率 (k)	偏移量 (d)
int16	0.005332	-0.072956
int32	0.000533	-0.034809

数据类型	斜率 (k)	偏移量 (d)
float	0.547474	-0.150232

2.3.4 4438 (双极, 32减速比) 扭矩 (Nm)

数据类型	斜率 (k)	偏移量
int16	0.004855	-0.083
int32	0.000486	-0.083
float	0.485565	-0.083

2.4 温度 (°C)

数据类型	LSB	实际 (°C)
int8	1	1
int16	1	0.1
int32	1	0.001
float	1	1

2.5 时间 (s)

数据类型	LSB	实际 (s)
int8	1	0.01
int16	1	0.001
int32	1	0.000001
float	1	1

2.6 位置 (转)

数据类型	LSB	实际 (转)	实际 (°)
int8	1	0.01	3.6
int16	1	0.0001	0.036
int32	1	0.00001	0.0036
float	1	1	360

2.7 速度 (转/秒)

数据类型	LSB	实际 (转/秒)
int8	1	0.01
int16	1	0.00025
int32	1	0.00001
float	1	1

2.8 加速度 (转/秒^2)

数据类型	LSB	实际 (转/秒^2)
int8	1	0.05
int16	1	0.001
int32	1	0.00001
float	1	1

2.9 PWM 标度 (无单位)

数据类型	LSB	实际
int8	1	1/127 - 0.007874
int16	1	1/32767 - 0.000030519
int32	1	(1/2147483647) - 4.657^10
float	1	1

2.10 rKp、rKd 标度 (无单位, 寄存器0x23和0x24的)

数据类型	LSB	实际
int8	1	1/127 - 0.007874
int16	1	1/32767 - 0.000030519
int32	1	(1/2147483647) - 4.657^10
float	1	1

2.11 Kp、Kd 标度（无单位，例程中用的是这个）

数据类型	LSB	实际
int8	1	1
int16	1	0.1
int32	1	0.001
float	1	1

3. 函数示例

说明：

- 下面介绍示例程序中提供的电机控制模式的简要说明。
- 详情请看提供的示例工程。
- 本电机所能实现的功能不止于此，如需自定义特殊功能，可根据寄存器表发挥想象。

3.1 普通模式

3.1.1 DQ 电压控制

说明：

- D 相电压默认为 0。
- Q 相电压由用户控制。

提供函数：

```
void set_dq_volt_float(FDCAN_HandleTypeDef *fdcanHandle, uint8_t id, float volt);
void set_dq_volt_int32(FDCAN_HandleTypeDef *fdcanHandle, uint8_t id, int32_t volt);
void set_dq_volt_int16(FDCAN_HandleTypeDef *fdcanHandle, uint8_t id, int16_t volt);
```

3.1.2 DQ 电流控制

说明：

- D 相电流默认为 0。
- Q 相电流由用户控制。

提供函数：

```
void set_dq_current_float(FDCAN_HandleTypeDef *fdcanHandle, uint8_t id, float current);
void set_dq_current_int32(FDCAN_HandleTypeDef *fdcanHandle, uint8_t id, int32_t current);
void set_dq_current_int16(FDCAN_HandleTypeDef *fdcanHandle, uint8_t id, int16_t current);
```

3.1.3 位置控制

说明:

- 以最大速度和力矩转动到指定位置。
- 正负表示方向。

提供函数:

```
void set_pos_float(FDCAN_HandleTypeDef *fdcanHandle, uint8_t id, float pos);
void set_pos_int32(FDCAN_HandleTypeDef *fdcanHandle, uint8_t id, int32_t pos);
void set_pos_int16(FDCAN_HandleTypeDef *fdcanHandle, uint8_t id, int16_t pos);
```

3.1.4 速度控制

说明:

- 以设置速度转动。
- 正负表示方向。

提供函数:

```
void set_vel_float(FDCAN_HandleTypeDef *fdcanHandle, uint8_t id, float vel);
void set_vel_int32(FDCAN_HandleTypeDef *fdcanHandle, uint8_t id, int32_t vel);
void set_vel_int16(FDCAN_HandleTypeDef *fdcanHandle, uint8_t id, int16_t vel);
```

3.1.5 力矩控制

说明:

- 以给定的力矩转动。

提供函数:

```
void set_torque_float(FDCAN_HandleTypeDef *fdcanHandle, uint8_t id, float torque);
void set_torque_int32(FDCAN_HandleTypeDef *fdcanHandle, uint8_t id, int32_t torque);
void set_torque_int16(FDCAN_HandleTypeDef *fdcanHandle, uint8_t id, int16_t torque);
```

3.1.6 位置、速度、最大力矩控制

说明:

- 以给定的速度转动到指定位置，并限制输出的最大力矩。
- 不想控制力矩可直接给一个很大的值或者无限制宏（无限制宏在例程中有宏定义）。

提供函数:

```
void set_pos_vel_tqe_float(FDCAN_HandleTypeDef *fdcanHandle, uint8_t id, float pos, float vel, float torque);
void set_pos_vel_tqe_int32(FDCAN_HandleTypeDef *fdcanHandle, uint8_t id, int32_t pos, int32_t vel, int32_t torque);
void set_pos_vel_tqe_int16(FDCAN_HandleTypeDef *fdcanHandle, uint8_t id, int16_t pos, int16_t vel, int16_t torque);
```

3.1.7 位置、速度、力矩、PD 控制

说明:

- 输出力矩 = 位置偏差 * Mkp + 速度偏差 * Mkd + tqe (Mkp 表示电机内部 kp, Mkd 表示电机内部 kd)
- 并可调节内部 Kp、Kd 的比例。

提供函数:

```
void set_pos_vel_tqe_pd_float(FDCAN_HandleTypeDef *fdcanHandle, uint8_t id, float pos, float vel, float tqe, float kp, float kd);
void set_pos_vel_tqe_pd_int32(FDCAN_HandleTypeDef *fdcanHandle, uint8_t id, int32_t pos, int32_t vel, int32_t tqe, int32_t kp, int32_t kd);
void set_pos_vel_tqe_pd_int16(FDCAN_HandleTypeDef *fdcanHandle, uint8_t id, int16_t pos, int16_t vel, int16_t tqe, int16_t kp, int16_t kd);
```

3.1.8 速度、速度限幅控制

说明:

- 以指定速度转动、若速度大于速度限幅，则以速度限幅转动。

提供函数:

```
void set_vel_velmax_int16(FDCAN_HandleTypeDef *fdcanHandle, uint8_t id, int16_t vel, int16_t vel_max);
```

3.1.9 位置、速度、加速度控制（梯形控制）

说明:

- 指定电机转动到某个位置，并限制转动过程中的最大速度和加速度。

提供函数:

```
void set_pos_velmax_acc_float(FDCAN_HandleTypeDef *fdcanHandle, uint8_t id, float pos, float vel_max, float acc);
void set_pos_velmax_acc_int32(FDCAN_HandleTypeDef *fdcanHandle, uint8_t id, int32_t pos, int32_t vel_max, int32_t acc);
void set_pos_velmax_acc_int16(FDCAN_HandleTypeDef *fdcanHandle, uint8_t id, int16_t pos, int16_t vel_max, int16_t acc);
```

3.1.10 速度、加速度控制

说明:

- 以指定的加速度加速到指定速度。

提供函数:

```
void set_vel_acc_float(FDCAN_HandleTypeDef *fdcanHandle, uint8_t id, float vel, float acc);
void set_vel_acc_int32(FDCAN_HandleTypeDef *fdcanHandle, uint8_t id, int32_t vel, int32_t acc);
void set_vel_acc_int16(FDCAN_HandleTypeDef *fdcanHandle, uint8_t id, int16_t vel, int16_t acc);
```

3.1.11 重设零点

说明:

- 将当前位置设为零点。

提供函数:

```
void set_pos_rezero(FDCAN_HandleTypeDef *fdcanHandle, uint8_t id);
```

3.1.12 保存设置

说明:

- 保存电机配置信息。
- 目前只用在重设零点。

提供函数:

```
void set_conf_write(FDCAN_HandleTypeDef *fdcanHandle, uint8_t id);
```

3.1.13 电机刹车

说明:

- 电机所有相都短接到地，产生被动的“刹车”效果。

提供函数:

```
void set_motor_brake(FDCAN_HandleTypeDef *fdcanHandle, uint8_t id);
```

3.1.14 电机停止

说明:

- 进入停止状态。

提供函数:

```
void set_motor_stop(FDCAN_HandleTypeDef *fdcanHandle, uint8_t id);
```

3.1.15 读取电机状态

说明:

- 提供了读取电机运行状态、位置、速度、转矩四种数据的例程。
- 这里的函数是让电机发送这四种数据回来，具体解析数据的例程请看示例代码。

提供函数:

```
void read_motor_state_float(FDCAN_HandleTypeDef *fdcanHandle, uint8_t id);
void read_motor_state_int32(FDCAN_HandleTypeDef *fdcanHandle, uint8_t id);
void read_motor_state_int16(FDCAN_HandleTypeDef *fdcanHandle, uint8_t id);
```

3.1.16 周期返回电机状态数据

说明:

1. 周期返回电机位置、速度、力矩数据。
2. 返回数据格式和使用 0x17, 0x01 指令获取的格式一样
3. 周期单位为 ms。
4. 最小周期为 1ms, 最大周期 32767ms。
5. 如需停止周期返回数据, 将周期给 0 即可, 或者给电机断电。

```
void timed_return_motor_status_int16(uint8_t id, int16_t t_ms);
```

3.2 一拖多模式

3.2.1 一拖多位置控制

说明:

- 提供了一个 fdcan 帧控制 4 个电机位置的例程。
- 用户可自行增加或减少电机数量。
- ID 固定为 0X8080。

提供函数:

```
void set_many_pos_int16(FDCAN_HandleTypeDef *fdcanHandle, int16_t pos1, int16_t pos2, int16_t pos3, int16_t pos4);
```

3.2.2 一拖多速度控制

说明:

- 提供了一个 fdcan 帧控制 4 个电机速度的例程。
- 用户可自行增加或减少电机数量。
- ID 固定为 0X8081。

提供函数:

```
void set_many_vel_int16(FDCAN_HandleTypeDef *fdcanHandle, int16_t vel1, int16_t vel2, int16_t vel3, int16_t vel4);
```

3.2.3 一拖多力矩控制

说明:

- 提供了一个 fdcan 帧控制 4 个电机力矩的例程。
- 用户可自行增加或减少电机数量。
- ID 固定为 0X8082。

提供函数:

```
void set_many_tqe_int16(FDCAN_HandleTypeDef *fdcanHandle, int16_t tqe1, int16_t tqe2, int16_t tqe3, int16_t tqe4);
```

3.2.4 一拖多 DQ 电压控制

说明:

- 提供了一个 fdcan 帧控制 4 个电机 Q 相电压控制的例程。
- 用户可自行增加或减少电机数量。
- ID 固定为 0X8083。

提供函数:

```
void set_many_volt_int16(FDCAN_HandleTypeDef *fdcanHandle, int16_t volt1, int16_t volt2, int16_t volt3, int16_t volt4);
```

3.2.5 一拖多 DQ 电流控制

说明:

- 提供了一个 fdcan 帧控制 4 个电机 Q 相电流的例程。
- 用户可自行增加或减少电机数量。
- ID 固定为 0X8084。

提供函数:

```
void set_many_volt_int16(FDCAN_HandleTypeDef *fdcanHandle, int16_t volt1, int16_t volt2, int16_t volt3, int16_t volt4);
```

3.2.6 一拖多位置、速度、最大力矩控制

说明:

- 提供了一个 fdcan 帧控制 2 个电机位置、速度、最大力矩的例程。
- 即电机按指定速度运行到指定位置，但若电机输出力矩达到最大力矩限制任不能达到指定位置或速度，则输出指定的最大力矩。
- 若不想控制最大力矩，可将最大力矩设为 0xFFFF，意为不限制输出力矩大小。
- 用户可自行增加或减少电机数量。
- ID 固定为 0X8090。

提供函数:

```
void set_many_pos_vel_tqe_int16(FDCAN_HandleTypeDef *fdcanHandle, int16_t pos1, int16_t vel1, int16_t tqe1, int16_t pos2, int16_t vel2, int16_t tqe2);
```

3.2.7 一拖多位置、速度、力矩、PD 控制

说明:

- 提供了一个 fdcan 帧控制 2 个电机位置、速度、力矩、PD 的例程。
- 用户可自行增加或减少电机数量。
- ID 固定为 0X8093。

提供函数:


```
void set_many_pos_vel_tqe_pd_int16(FDCAN_HandleTypeDef *fdcanHandle, int16_t pos1, int16_t vel1, int16_t tqe1, int16_t rkp1, int16_t rkd1, int16_t pos2, int16_t vel2, int16_t tqe2, int16_t rkp2, int16_t rkd2);
```

3.2.8 一拖多位置、速度、PD 控制

说明:

- 提供了一个 fdcan 帧控制 2 个电机位置、速度、PD 的例程。
- **输出力矩 = 位置偏差 * Mkp + 速度偏差 * Mkd + tqe** (Mkp 表示电机内部 kp, Mkd 表示电机内部 kd)
- 用户可自行增加或减少电机数量。
- ID 固定为 0X809E。

提供函数:

```
void set_many_pos_vel_pd_int16(FDCAN_HandleTypeDef *fdcanHandle, int16_t pos1, int16_t vel1, int16_t rkp1, int16_t rkd1, int16_t pos2, int16_t vel2, int16_t rkp2, int16_t rkd2);
```

3.2.9 一拖多位置、速度、加速度（梯形控制）

说明:

- 提供了一个 fdcan 帧控制 2 个电机位置、速度、加速度（梯形控制）的例程。
- 用户可自行增加或减少电机数量。
- ID 固定为 0X80AD。

提供函数:

```
void set_many_pos_vel_acc_int16(FDCAN_HandleTypeDef *fdcanHandle, int16_t pos1, int16_t vel1, int16_t acc1, int16_t pos2, int16_t vel2, int16_t acc2);
```

4. 示例程序说明

- 电机示例函数都在 libelybot. c 和 libelybot. h 文件内。
- 示例程序默认不会运行任何控制代码，需测试某一功能请在 main. c 文件内解注释或自行编写。

需要注意的宏定义:

1. `MOTOR_TORQUE_RATIO`: 用于选择电机型号，用于修正输入力矩。
2. `POS_FLAG`: 用于**测试位置模式**，共三种模式，三种模式只是数据类型不同，效果都是 -0.5 转~0.5 转来回旋转。
 1. `POS_FLAG = 1`: float
 2. `POS_FLAG = 2`: int32
 3. `POS_FLAG = 3`: int16
3. `READ_MOTOR_FLAG`: 用于改变读取**电机状态的数据类型**。
 1. `READ_MOTOR_FLAG = 1`: float
 2. `READ_MOTOR_FLAG = 2`: int 32
 3. `READ_MOTOR_FLAG = 3`: int 16
4. `POS_REZERO`: 用于测试电机的**重置零点**功能，
 1. 效果是电机上电 3 秒后将当前位置设为零点。

2. 注意：需让电机停止后再重置零位，否则无效
3. 需测试此功能将 POS_REZERO 解注释即可。
5. **MOTOR_STOP**：用于测试电机**停止**功能。
 1. 需测试此功能将 MOTOR_STOP 解注释即可。
 2. 效果是电机上电 3 秒后将停止。
 3. 注意：需配合电机控制函数使用，启用 MOTOR_STOP 宏不会改变任何电机控制函数。
6. **MOTOR_BRAKE**：用于测试电机**刹车**功能功能。
 1. 需测试此功能将 MOTOR_BRAKE 解注释即可。