OIDelec 欧艾迪®

智能FOC无刷电机驱动器

（RS485 系列）编程手册 V1.0

深圳欧艾迪科技有限公司
www.oidelec.com

修订历史

版本	日期	备注
V1.0	2022-10-10	首次创建

目 录

1. 文档名词说明

ADU	应用数据单元
HDLC	高级数据链路控制
HMI	人机界面
IETF	因特网工程工作组
I/O	输入/输出设备
IP	互连网协议
MAC	介质访问控制
MB	MODBUS 协议
MBAP	MODBUS 协议
PDU	协议数据单元
PLC	可编程逻辑控制器
TCP	传输控制协议
进制	文中指令未做特别说明的数字皆采用十六进制。

2. Modbus 协议

2.1. 协议描述

MODBUS 协议定义了一个与基础通信层无关的简单协议数据单元（PDU）。特定总线或网络上 的 MODBUS 协议映射能够在应用数据单元（ADU）上引入一些附加域。

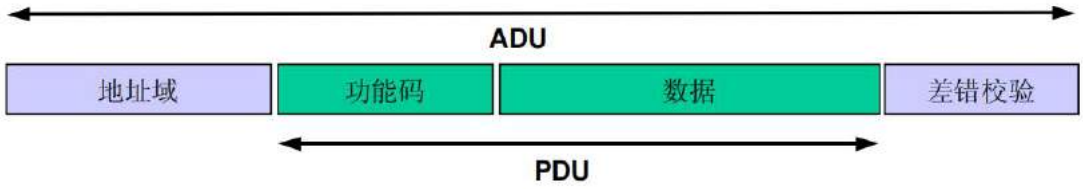


图 2-1 通用 MODBUS 帧

启动 MODBUS 事务处理的客户机创建 MODBUS 应用数据单元。功能码向服务器指示将执行哪种操作。

MODBUS 协议建立了客户机启动的请求格式。

用一个字节编码 MODBUS 数据单元的功能码域。有效的码字范围是十进制 1-255（128-255 为异常响应保留）。当从客户机向服务器设备发送报文时，功能码域通知服务器执行哪种操作。

向一些功能码加入子功能码来定义多项操作。

从客户机向服务器设备发送的报文数据域包括附加信息，服务器使用这个信息执行功能码定义的操作。这个域还包括离散项目和寄存器地址、处理的项目数量以及域中的实际数据字节数。

在某种请求中，数据域可以是不存在的（0 长度），在此情况下服务器不需要任何附加信息。功能码仅说明操作。

如果在一个正确接收的 MODBUS ADU 中，不出现与请求 MODBUS 功能有关的差错，那么服务器至客户机的响应数据域包括请求数据。如果出现与请求 MODBUS 功能有关的差错，那么域包括一个异常码，服务器应用能够使用这个域确定下一个执行的操作。

例如，客户机能够读一组离散量输出或输入的开/关状态，或者客户机能够读/写一组寄存器的数据内容。当服务器对客户机响应时，它使用功能码域来指示正常（无差错）响应或者出现某种差错（称为异常响应）。对于一个正常响应来说，服务器仅对原始功能码响应。

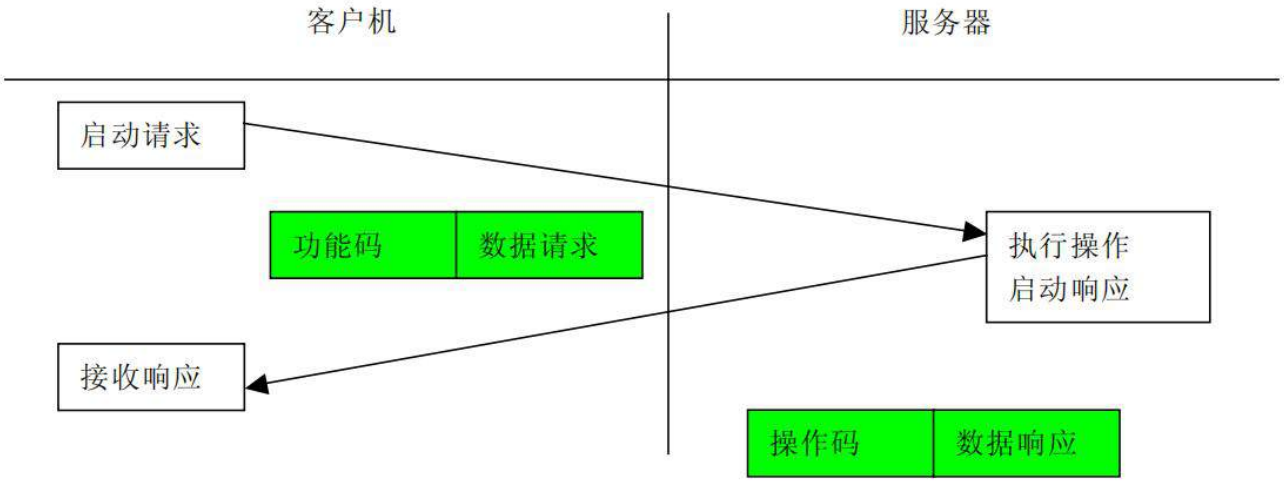


图 2-2 事务处理（无异常）

对于异常响应，服务器返回一个与原始功能码等同的码，设置该原始功能码的最高有效位为逻辑 1。

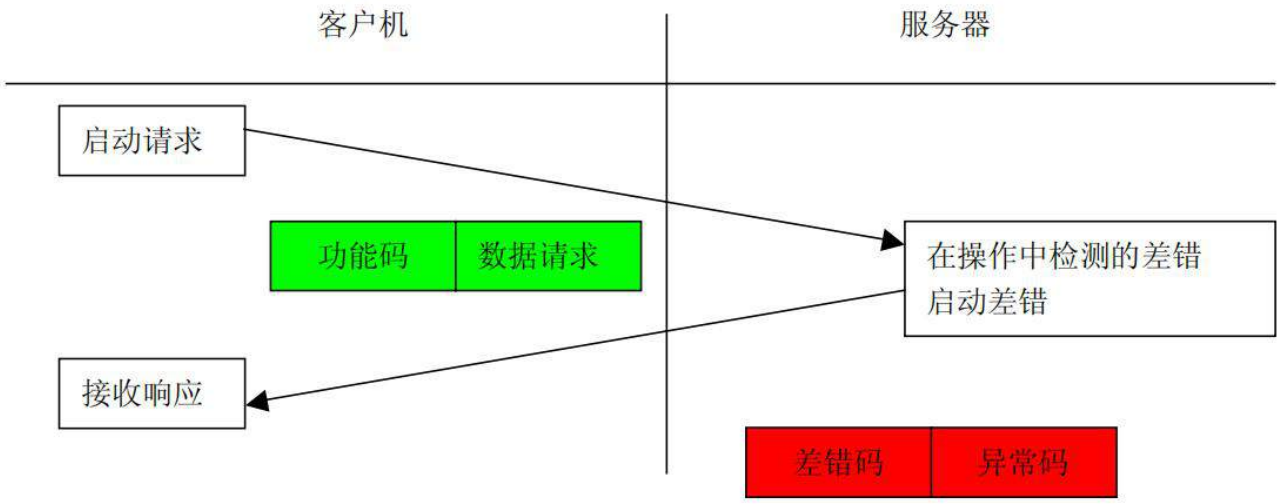


图 2-3 事务处理（异常）

2.2. 数据编码

MODBUS 使用一个**大端**表示地址和数据项。这意味着当发射多个字节时，首先发送最高有效位。例如：我们想发送 0x1234 发送的第一字节为 0x12 然后 0x34。

2.3. MODBUS 数据模型

MODBUS 以一系列具有不同特征表格上的数据模型为基础。四个基本表格为：

基本表格	对象类型	访问类型	内容
离散量输入	单个比特	只读	I/O 系统提供这种类型数据
线圈	单个比特	读写	通过应用程序改变这种类型数据
输入寄存器	16-比特字	只读	I/O 系统提供这种类型数据
保持寄存器	16-比特字	读写	通过应用程序改变这种类型数据

表 2-1 基本数据模型

2.4. 功能码定义

			功能码（十进制）	功能码（十六进制）
数据访问	bit 访问	读输入离散量	02	02
		读线圈	01	01
		写单个线圈	05	05
		写多个线圈	15	0F
	16 bits 访问	读输入寄存器	04	04
		读多个寄存器	03	03
		写单个寄存器	06	06
		写多个寄存器	16	10
		读/ 写多个寄存器	23	17

表 2-2 公共功能码定义

2.5. 功能码描述

2.5.1. 01 (0x01) 读线圈

在一个远程设备中，使用该功能码读取线圈的 1 至 2000 连续状态。请求 PDU 详细说明了起始地址，即指定的第一个线圈地址和线圈编号。从零开始寻址线圈。

根据数据域的每个比特将响应报文中的线圈分成为一个线圈。指示状态为 1= ON 和 0= OFF。

第一个数据字节的 LSB（最低有效位）包括在询问中寻址的输出。其它线圈依次类推，一直到这个字节的高位端为止，并在后续字节中从低位到高位的顺序。

如果返回的输出数量不是八的倍数，将用零填充最后数据字节中的剩余比特（一直到字节的高位端）。字节数量域说明了数据的完整字节数。

- 请求 PDU

功能码	1 个字节	0x01
起始地址	2 个字节	0x0000 至0xFFFF
线圈数量	2 个字节	1 至 2000（0x7D0）

- 响应 PDU

功能码	1 个字节	0x01
字节数	1 个字节	N
线圈状态	N 个字节	n = N 或 N+1

N = 输出数量/8，如果余数不等于 0，那么 N = N+1

- 错误

错误码	1 个字节	功能码 + 0x80（**0x81**）
异常码	1 个字节	01、02、03、04

2.5.2. 02 (0x02) 读离散量输入

在一个远程设备中，使用该功能码读取离散量输入的 1 至 2000 连续状态。请求 PDU 详细说明了起始地址，即指定的第一个输入地址和输入编号。从零开始寻址输入。

根据数据域的每个比特将响应报文中的离散量输入分成为一个输入。指示状态为 1= ON 和 0= OFF。第一个数据字节的 LSB（最低有效位）包括在询问中寻址的输入。其它输入依次类推，一直到这个字节的高位端为止，并在后续字节中从低位到高位的顺序。

如果返回的输入数量不是八的倍数，将用零填充最后数据字节中的剩余比特（一直到字节的高位端）。字节数量域说明了数据的完整字节数。

- 请求 PDU

功能码	1 个字节	0x02
起始地址	2 个字节	0x0000 至0xFFFF
数量	2 个字节	1 至 2000（0x7D0）

- 响应 PDU

功能码	1 个字节	0x02
字节数	1 个字节	N
状态	N 个字节	n = N 或 N+1

N = 输出数量/8，如果余数不等于 0，那么 N = N+1

- 错误

错误码	1 个字节	功能码 + 0x80（**0x82**）
异常码	1 个字节	01、02、03、04

2.5.3. 03 (0x03) 读保持寄存器

在一个远程设备中，使用该功能码读取保持寄存器连续块的内容。请求 PDU 说明了起始寄存器地址和寄存器数量。从零开始寻址寄存器。

将响应报文中的寄存器数据分成每个寄存器有两字节。

对于每个寄存器，第一个字节包括高位比特，并且第二个字节包括低位比特。

- 请求 PDU

功能码	1 个字节	0x03
起始地址	2 个字节	0x0000 至0xFFFF
数量	2 个字节	1 至 125（0x7D）

- 响应 PDU

功能码	1 个字节	0x03
字节数	1 个字节	2*N
状态	N*2 个字节	

N = 寄存器数量

- 错误

错误码	1 个字节	功能码 + 0x80（**0x83**）
异常码	1 个字节	01、02、03、04

2.5.4. 04 (0x04) 读输入寄存器

在一个远程设备中，使用该功能码读取 1 至大约 125 的连续输入寄存器。请求 PDU 说明了起始地址和寄存器数量。从零开始寻址寄存器。

将响应报文中的寄存器数据分成每个寄存器有两字节。

对于每个寄存器，第一个字节包括高位比特，并且第二个字节包括低位比特。

- 请求 PDU

功能码	1 个字节	0x04
起始地址	2 个字节	0x0000 至 0xFFFF
数量	2 个字节	1 至 125（0x7D）

- 响应 PDU

功能码	1 个字节	0x04
字节数	1 个字节	2*N
状态	N*2 个字节	

N = 寄存器数量

- 错误

错误码	1 个字节	功能码 + 0x80（**0x84**）
异常码	1 个字节	01、02、03、04

2.5.5. 05 (0x05) 写单个线圈

在一个远程设备上，使用该功能码写单个输出为 ON 或 OFF。

请求数据域中的常量说明请求的 ON/OFF 状态。十六进制值 FF 00 请求输出为 ON。十六进制值 00 00 请求输出为 OFF。其它所有值均是非法的，并且对输出不起作用。

请求 PDU 说明了强制的线圈地址。从零开始寻址线圈。因此，寻址线圈 1 为 0。线圈值域的常量说明请求的 ON/OFF 状态。十六进制值 0XFF00 请求线圈为 ON。十六进制值 0X0000 请求线圈为 OFF。其它所有值均为非法的，并且对线圈不起作用。

正常响应是请求的应答，在写入线圈状态之后返回这个正常响应。

- 请求 PDU

功能码	1 个字节	0x05
地址	2 个字节	0x0000 至0xFFFF
输出值	2 个字节	0x0000 至 0xFF00

- 响应 PDU

功能码	1 个字节	0x05
地址	2 个字节	0x0000 至0xFFFF
输出值	2 个字节	0x0000 至 0xFF00

- 错误

错误码	1 个字节	功能码 + 0x80（**0x85**）
异常码	1 个字节	01、02、03、04

2.5.6. 06 (0x06) 写单个寄存器

在一个远程设备中，使用该功能码写单个保持寄存器。

请求 PDU 说明了被写入寄存器的地址。从零开始寻址寄存器。因此，寻址寄存器 1 为 0。正常响应是请求的应答，在写入寄存器内容之后返回这个正常响应。

- 请求 PDU

功能码	1 个字节	0x06
地址	2 个字节	0x0000 至0xFFFF
输出值	2 个字节	0x0000 至 0xFFFF

- 响应 PDU

功能码	1 个字节	0x06
地址	2 个字节	0x0000 至0xFFFF
输出值	2 个字节	0x0000 至 0xFFFF

- 错误

错误码	1 个字节	功能码 + 0x80（**0x86**）
异常码	1 个字节	01、02、03、04

2.5.7. 15 (0x0F) 写多个线圈

在一个远程设备中，使用该功能码强制线圈序列中的每个线圈为 ON 或 OFF。请求 PDU 说明了强制的线圈参考。从零开始寻址线圈。

请求数据域的内容说明了被请求的 ON/OFF 状态。域比特位置中的逻辑“1”请求相应输出为 ON。域比特位置中的逻辑“0”请求相应输出为 OFF。

正常响应返回功能码、起始地址和强制的线圈数量。

- 请求 PDU

功能码	1 个字节	0x0F
地址	2 个字节	0x0000 至0xFFFF
输出数量	2 个字节	0x0001 至 0x07B0
字节数	1 个字节	N
输出值	N*1 个字节	

N = 输出数量/8，如果余数不等于 0，那么 N = N+1

- 响应 PDU

功能码	1 个字节	0x0F
地址	2 个字节	0x0000 至0xFFFF
输出值	2 个字节	0x0001 至 0x07B0

- 错误

错误码	1 个字节	功能码 + 0x80（**0x8F**）
异常码	1 个字节	01、02、03、04

2.5.8. 16 (0x10) 写多个寄存器

在一个远程设备中，使用该功能码写连续寄存器块(1 至约 120 个寄存器)。

在请求数据域中说明了请求写入的值。每个寄存器将数据分成两字节。

正常响应返回功能码、起始地址和被写入寄存器的数量。

- 请求 PDU

功能码	1 个字节	0x10
地址	2 个字节	0x0000 至 0xFFFF
输出数量	2 个字节	0x0001 至 0x007B
字节数	1 个字节	2*N
寄存器值	N*2 个字节	

N = 寄存器数量

- 响应 PDU

功能码	1 个字节	0x10
地址	2 个字节	0x0000 至 0xFFFF
输出值	2 个字节	0x0001 至 0x007B

- 错误

错误码	1 个字节	功能码 + 0x80（**0x90**）
异常码	1 个字节	01、02、03、04

2.6. Modbus-RTU

2.6.1. 描述

由发送设备将 Modbus 报文构造为带有已知起始和结束标记的帧。这使设备可以在报文的开始接收

新帧，并且知道何时报文结束。不完整的报文必须能够被检测到而错误标志必须作为结果被设置。在 RTU 模式，报文帧由时长至少为 3.5 个字符时间的空闲间隔区分。在后续的部分，这个时间区间被称作 t3.5。

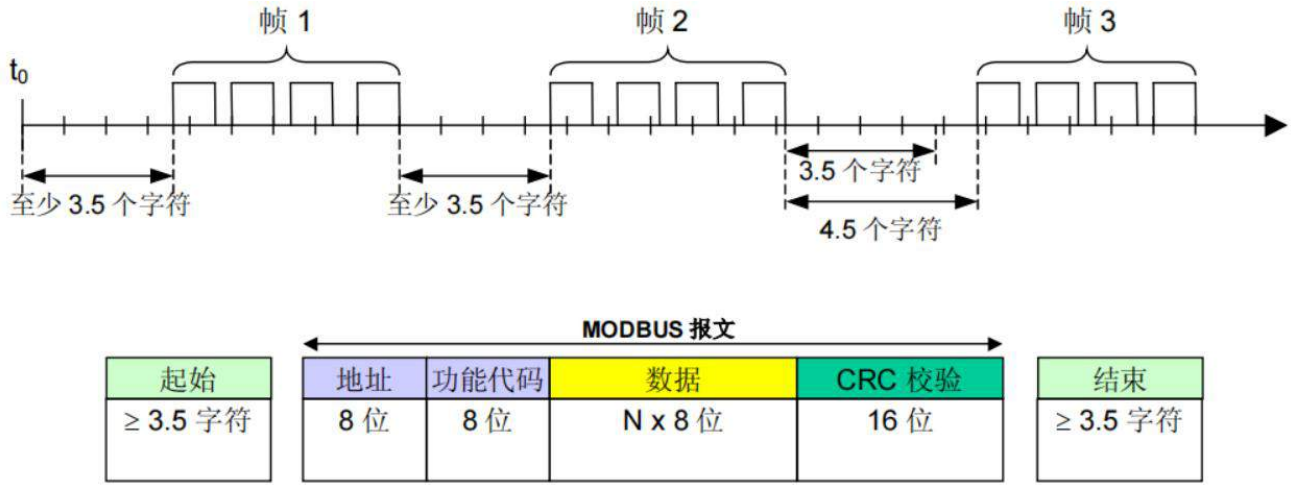
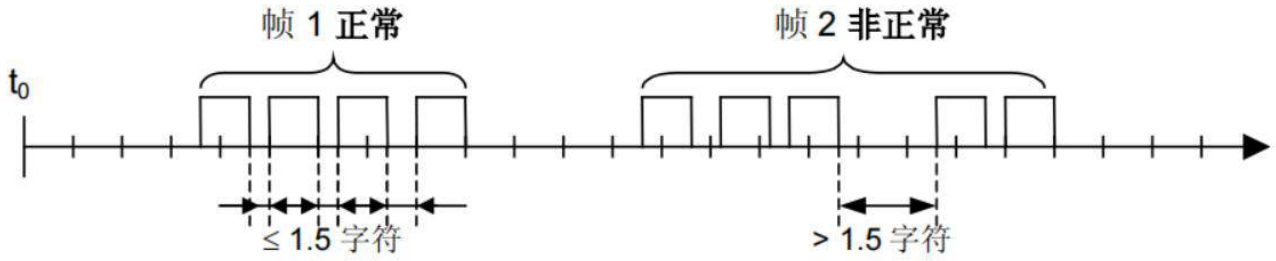


图 2-4 RTU 报文帧

整个报文帧必须以连续的字符流发送。

如果两个字符之间的空闲间隔大于 1.5 个字符时间，则报文帧被认为不完整应该被接收节点丢弃。



2.6.2. CRC 校验

在 RTU 模式包含一个对全部报文内容执行的，基于循环冗余校验（CRC - Cyclical Redundancy Checking）算法的错误检验域。CRC 域检验整个报文的内容。不管报文有无奇偶校验，均执行此检验。

CRC 包含由两个 8 位字节组成的一个 16 位值。

CRC 域作为报文的最后的域附加在报文之后。计算后，首先附加低字节，然后是高字节。CRC 高字节为报文发送的最后一个子节。

附加在报文后面的 CRC 的值由发送设备计算。接收设备在接收报文时重新计算 CRC 的值，并将计算

结果于实际接收到的 CRC 值相比较。如果两个值不相等，则为错误。

CRC 的计算，开始对一个 16 位寄存器预装全 1。然后将报文中的连续的 8 位子节对其进行后续的计算。只有字符中的 8 个数据位参与生成 CRC 的运算，起始位，停止位和校验位不参与 CRC 计算。

CRC 的生成过程中，每个 8–位字符与寄存器中的值异或。然后结果向最低有效位(LSB)方向移动(Shift) 1 位，而最高有效位(MSB)位置充零。 然后提取并检查 LSB，如果 LSB 为 1， 则寄存器中的值与一个固定的预置值异或；如果 LSB 为 0， 则不进行异或操作。

这个过程将重复直到执行完 8 次移位。完成最后一次（第 8 次）移位及相关操作后，下一个 8 位字节与寄存器的当前值异或，然后又同上面描述过的一样重复 8 次。当所有报文中子节都运算之后得到的寄存器中的最终值，就是 CRC。

当 CRC 附加在报文之后时，首先附加低字节，然后是高字节。

2.6.3. 常见 RTU 帧格式

- **0x03 读保持寄存器**

举例：读取 1 号站，6000~6001 2 个保持寄存器的数值。

Modbus RTU 主站设备请求帧格式：

设备地址 [1 字节]	功能码 [1 字节]	起始地址 [2 字节]		寄存器数量 [2 字节]		校验 [2 字节]	
		高 8 位	低 8 位	高 8 位	低 8 位	低 8 位	高 8 位
0x01	0x03	0x17	0x70	0x00	0x02	0xC0	0x64

1 号站设备响应帧格式：

设备地址 [1 字节]	功能码 [1 字节]	字节数 [1 字节]	寄存器值 [寄存器数量*2 字节]					校验 [2 字节]	
			寄存器 1 高 8 位	寄存器 1 低 8 位	…	寄存器 N 高 8 位	寄存器 N 低 8 位	低 8 位	高 8 位
0x01	0x03	0x04	0x00	0x64		0x13	0x88	0xB6	0xBA

寄存器 6000 值	寄存器 6001 值

- **0x04 读输入寄存器**

举例：读取 1 号站，5000~5001 2 个输入寄存器的数值。

Modbus RTU 主站设备请求帧格式：

设备地址 [1 字节]	功能码 [1 字节]	起始地址 [2 字节]		寄存器数量 [2 字节]		校验 [2 字节]	
		高 8 位	低 8 位	高 8 位	低 8 位	低 8 位	高 8 位
0x01	0x04	0x13	0x88	0x00	0x02	0xF5	0x65

1 号站设备响应帧格式：

设备地址 [1 字节]	功能码 [1 字节]	字节数 [1 字节]	寄存器值 [寄存器数量*2 字节]					校验 [2 字节]	
			寄存器 1 高 8 位	寄存器 1 低 8 位	…	寄存器 N 高 8 位	寄存器 N 低 8 位	低 8 位	高 8 位
0x01	0x04	0x04	0x00	0x64		0x13	0x88	0xB7	0x0D
			寄存器 5000 值			寄存器 5001 值			

- **0x06 写单个寄存器**

举例：写入 1 号站，6000 保持寄存器的数值。

Modbus RTU 主站设备请求帧格式：

设备地址 [1 字节]	功能码 [1 字节]	寄存器地址 [2 字节]		写入值 [2 字节]		校验 [2 字节]	
		高 8 位	低 8 位	高 8 位	低 8 位	低 8 位	高 8 位
0x01	0x06	0x17	0x70	0x00	0x02	0x0C	0x64

1 号站设备响应帧格式：

设备地址 [1 字节]	功能码 [1 字节]	寄存器地址 [2 字节]	写入值 [2 字节]	校验 [2 字节]

		高 8 位	低 8 位	高 8 位	低 8 位	低 8 位	高 8 位
0x01	0x06	0x17	0x70	0x00	0x02	0x0C	0x64

- **0x10 写多个寄存器**

举例：写入 1 号站，6000~6001 2 个保持寄存器的数值 0x0064、0x1234。

Modbus RTU 主站设备请求帧格式：

设备地址 [1 字节]	功能码 [1 字节]	起始地址 [2 字节]		寄存器数量 [2 字节]		写入字节 [1 字节]	寄存器值 [寄存器数量*2 字节]					校验 [2 字节]	
		地址高 8 位	地址低 8 位	数量高 8 位	数量低 8 位		寄存器 1 高 8 位	寄存器 1 低 8 位	...	寄存器 N 高 8 位	寄存器 N 低 8 位	低 8 位	高 8 位
0x01	0x10	0x17	0x70	0x00	0x02	0x04	0x00	0x64		0x12	0x34	0x53	0xD3
							寄存器 6000 值			寄存器 6001 值			

1 号站设备响应帧格式：

设备地址 [1 字节]	功能码 [1 字节]	寄存器地址 [2 字节]		寄存器数量 [2 字节]		校验 [2 字节]	
		高 8 位	低 8 位	高 8 位	低 8 位	低 8 位	高 8 位
0x01	0x10	0x17	0x70	0x00	0x02	0x45	0xA7

3. 上位机设置 485 通信

设置通信控制之前请先参照《入门学习视频》识别电机参数和正确设置电机最大速度、磁极对数等。

注意：设置为通信控制后，上位机仅用作监控数据，控制无效。

设置步骤如下：

1、设置应用类型为**串口（485）**。

2、配置串口（485）参数

3、写入应用参数到驱动器

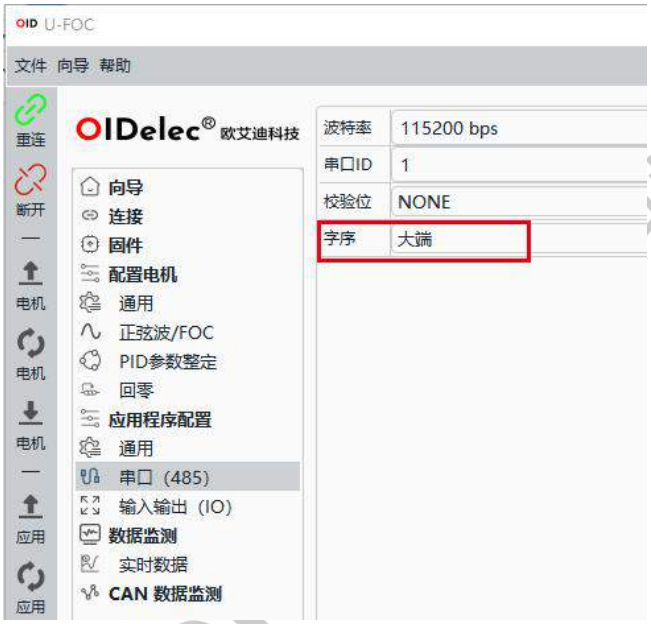
4、**驱动器会自动重启。（如果驱动器未自动重启，需用户手动重启驱动生效）**

4. 更改 modbus 字序

modbus-rtu 通信下传输 16 位（2 字节）数据时，规定大端传输。

而传输 32 位（4 字节）数据时，由上位机配置传输字序。

注意：文档中示意指令如未说明均在大端模式下发送。



- **大端**

对于一个 32 位的寄存器值（4 个字节），如果其值为 0x01020304，根据 Modbus RTU 的大端字节顺序，将会以如下方式发送：

发送字节：0x01

发送字节：0x02

发送字节：0x03

发送字节：0x04

- **小端**

对于一个 32 位的寄存器值（4 个字节），如果其值为 0x01020304，根据 Modbus RTU 的大端字节顺

序，将会以如下方式发送：

发送字节：0x03

发送字节：0x04

发送字节：0x01

发送字节：0x02

5. 驱动器寄存器地址

寄存器地址	描述	寄存器类型	读写属性	备注
5000	故障信息	输入寄存器	只读	见故障信息说明
5001-5002	实时转速	输入寄存器	只读	此转速为电角度转速，erpm=rpm*磁极对数
5003	实时占空比	输入寄存器	只读	无量纲，范围-1000~1000
5004	实时功率	输入寄存器	只读	单位 W
5005	实时输入电压	输入寄存器	只读	单位 V
5006	实时电机电流	输入寄存器	只读	单位 10mA
5007	实时总线电流	输入寄存器	只读	单位 10mA
5008	实时温度	输入寄存器	只读	单位度
5009	实时角度	输入寄存器	只读	单位 0.01 度
5010-5011	实时位置	输入寄存器	只读	单位 0.01 度。位置即为角度的累加
5012	回零状态	输入寄存器	只读	高 8 位：回零状态；0 代表回零中，1 代表回零完成。 低 8 位：回零代码；0 代表回零成功；-1 代表相应 IO 未设置功能；-2 代表中止回零。
6000	心跳包	保持寄存器	读写	需要在心跳时间内改变该寄存器的值，否则会停止电机

6001	控制模式	保持寄存器	读写	0: 电流控制模式 1：转速控制模式 2：占空比控制模式 3：绝对位置控制模式 4：相对上一次目标位置控制模式 5：相对当前位置控制模式 6：刹车电流控制模式 7：手刹电流控制模式 8：回零模式 9：回零停止 10： 电流爬升模式 0xFFFF：空模式
6002	设定电流	保持寄存器	读写	单位 10mA
6003-6004	设定转速	保持寄存器	读写	此转速为电角度转速，RPM 单位下的转速需要除以磁极对数
6005	设定占空比	保持寄存器	读写	无量纲，范围-1000~1000
6006-6007	设定绝对位置	保持寄存器	读写	设置绝对位置，单位 0.01 度。
6008-6009	设定相对位置	保持寄存器	读写	设置相对位置，单位 0.01 度。相对于上个目标值的增量
6010-6011	设定相对位置	保持寄存器	读写	设置相对位置，单位 0.01 度。相对于当前位置的增量
6012-6013	设定当前位置	保持寄存器	读写	设置当前位置，单位 0.01 度。设置为 0 即设零点。
6014	设定刹车电流	保持寄存器	读写	单位 10mA
6015	设定手刹电流	保持寄存器	读写	单位 10mA
6016-6017	设置速度环加速度	保持寄存器	读写	此加速度为每秒增加的电角度转速《仅速度环有效》

6018-6019	设置轨迹最大速度	保持寄存器	读写	转速为电角度转速，RPM 单位下的转速需要除以磁极对数《仅轨迹位置有效》
6020-6021	设置轨迹最大加速度	保持寄存器	读写	此加速度为每分钟增加的电角度转速《仅轨迹位置有效》
6022-6023	设置轨迹最大减速度	保持寄存器	读写	此减速度为每分钟增加的电角度转速《仅轨迹位置有效》
6024	设置当前生效的电机配置表	保持寄存器	读写	开机默认使用的 0 配置。掉电不保存。
6025-6026	设置速度环减速度	保持寄存器	读写	此加速度为每秒减少的电角度转速《仅速度环有效》
6027	回零模式	保持寄存器	读写	详情见上位机
6028	闭环模式最大电流值	保持寄存器	读写	满足调速情况下可以调整扭矩大小
6029	电流爬升加速度	保持寄存器	读写	单位每秒增加的千分比最大电流。如果值为 1000，即 1S 增加到最大电流值。
6030	电流爬升模式下设定电流值	保持寄存器	读写	单位千分比最大电流。

6. 故障信息说明

错误值	错误信息	备注
0	无错误	
1	过压	
2	欠压	
3	绝对值超过最大电流	
4	MOS管过温	
5	MCU欠压	
6	看门狗触发后启动	
7	SPI接口驱动器错误	
8	FLASH损毁	
9	U相电流传感器偏移过大	
10	V相电流传感器偏移过大	
11	W相电流传感器偏移过大	
12	三相电流不平衡	
13	FLASH中电机配置损毁	
14	FLASH中应用配置损毁	

7. 控制模式

7.1. 心跳保护机制

7.1.1. 描述

驱动器为保证通信安全。增加了心跳保护机制。主机需要周期更改心跳寄存器的值。周期和上位机设置的超时时间有关系。一般发送周期为超时时间的 1/2；如果更改驱动器通信参数需要写入到驱动器生效，见《上位机设置 485 通信》写入应用参数到驱动器。

注意：心跳和控制模式无关，建议单独新增线程，周期更改心跳寄存器的值。不管处于任何模式，都需要更新心跳的值。否则驱动器会放空电机，不能正确响应控制指令。

7.1.2. 指令示例

例：如果上位机配置心跳超时时间为 1000ms，主机以 500ms 周期更新心跳寄存器的值。

驱动器 ID：1

时间（ms）	指令	说明
0	01 06 17 70 00 01 4C 65	设置心跳寄存器值为 1
500	01 06 17 70 00 02 0C 64	设置心跳寄存器值为 2
1000	01 06 17 70 00 01 4C 65	设置心跳寄存器值为 1

1500	01 06 17 70 00 02 0C 64	设置心跳寄存器值为 2
	....	

7.2. 电流控制

7.2.1. 描述

电流控制即恒扭矩控制（电流闭环）。此时速度根据负载很变化。负载大转速就低，负载小转速就高。

正电流电机就正转，负电流电机就反转。

注：控制的前提是心跳一直在更新。见心跳保护机制。

7.2.2. 指令示例

例：如果驱动器 ID：1，电流 1A 控制电机。

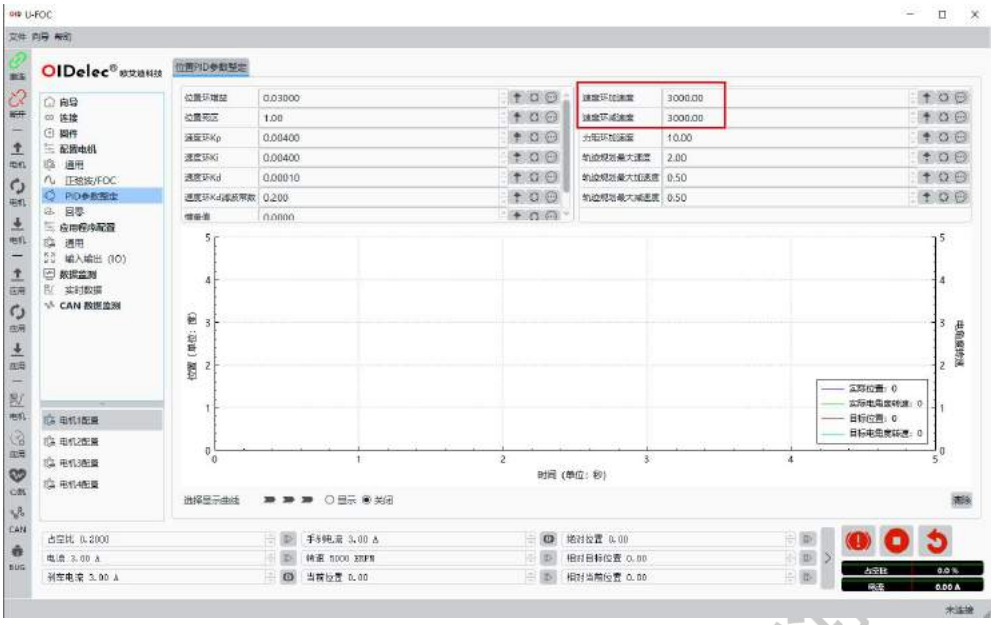
指令	说明
01 06 17 72 00 64 2D 8E	设置目标电流 1A。因为电流单位是 10mA。所以 1A=100*10mA
01 06 17 71 00 00 DC 65	设置控制模式为电流控制。默认控制模式为 0，所以第一次控制可能不需要发送更改控制模式。

7.3. 转速控制

7.3.1. 描述

转速控制即恒转速控制（速度闭环）。此时速度根据目标给定运行，不受外界负载影响（负载在额定扭矩范围内）。

驱动器自身带斜坡加减速功能，如果通信不设置，使用的就是驱动保存的配置值。可以上位机查看和修改。



正数就正转，负数就反转。

注：控制的前提是心跳一直在更新。见心跳保护机制。

驱动器内部使用的速度是 erpm，rpm=erpm/磁极对数

7.3.2. 指令示例

例：如果驱动器 ID：1，电机是 4 对级，控制电机以 1000rpm 运行。使用驱动器内部保存的加减速。

指令	说明
01 10 17 73 00 02 04 00 00 0F A0 5A E6	设置目标转速 4000erpm。
01 06 17 71 00 01 1D A5	设置控制模式为转速控制。

例：如果驱动器 ID：1，电机是 4 对级，控制电机以 1000rpm 运行。加速度 500rpm/s²，减速度 200rpm/s²。

指令	说明
01 10 17 73 00 02 04 00 00 0F A0 5A E6	设置目标转速 4000erpm。
01 10 17 80 00 02 04 00 00 07 D0 13 93	设置加速度 2000erpm/s²。
01 10 17 89 00 02 04 00 00 03 20 D1 7D	设置减速度 800erpm/s²。
01 06 17 71 00 01 1D A5	设置控制模式为转速控制。

7.4. 占空比控制

7.4.1. 描述

占空比控制也可以基于电流环控制转速，和转速控制的区别是占空比控制转速低，扭矩就小。转速高扭矩就大。速度控制是可以做到低速大扭矩的。

占空比正数就正转，负数就反转。量程是-1000~1000，对应反向最大转速和正向最大转速。

注：控制的前提是心跳一直在更新。见心跳保护机制。

7.4.2. 指令示例

例：如果驱动器 ID：1，设置占空比控制 100。即 10%转速。

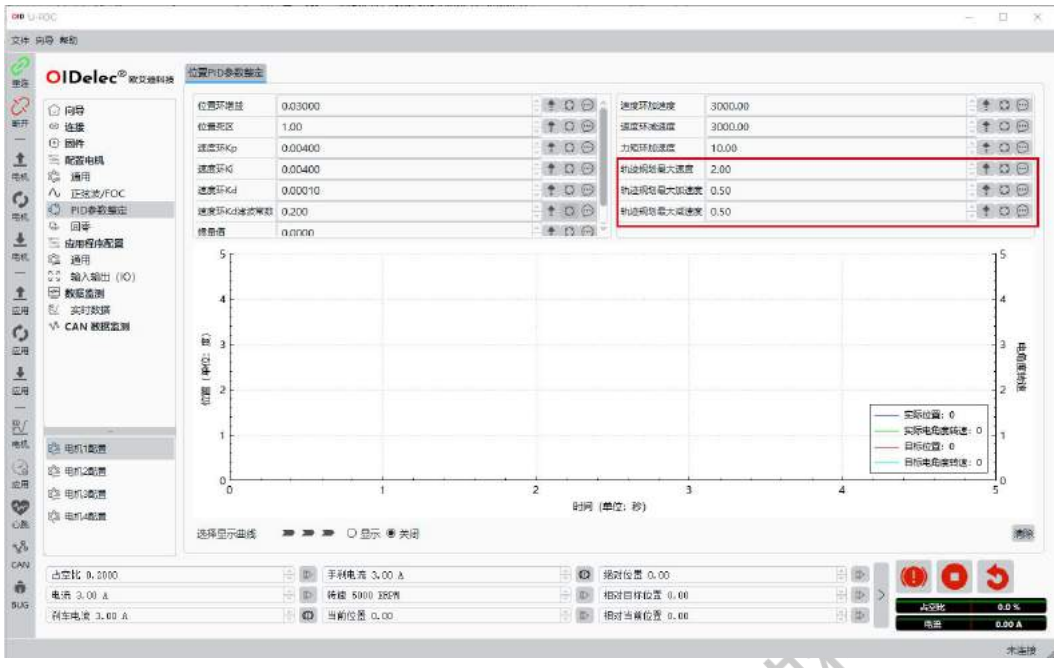
指令	说明
01 06 17 75 00 64 9C 4F	设置目标占空比 100。
01 06 17 71 00 02 5D A4	设置控制模式为占空比控制。

7.5. 绝对位置控制

7.5.1. 描述

绝对位置控制是使用的全局位置，每次给定的目标都是全局唯一的。可以通过轨迹最大速度、轨迹加速度、轨迹减速度控制位置控制的轨迹。

如果通信不设置，使用的就是驱动保存的配置值。可以上位机查看和修改。



注：控制的前提是心跳一直在更新。见心跳保护机制。

7.5.2. 指令示例

例：如果驱动器 ID：1，设置到绝对位置 36000。单位是 0.01 度，即 360 度。如果是从零点开始转动，此时电机应该转动 1 圈。最大速度设置为 5000erpm，加速度和减速度都设置为 5000erpm/s²。

注：位置指令仅在模式切换的那一刻生成一段路径规划。即控制位置之前需要把模式切换到空模式（或非目标模式皆可）。

指令	说明
01 06 17 71 FF FF DD D5	设置为空模式。
01 10 17 76 00 02 04 00 00 8C A0 FB E9	设置目标位置
01 10 17 82 00 02 04 00 00 13 88 9C B0	设置轨迹最大速度
01 10 17 84 00 02 04 00 00 13 88 1C 9A	设置轨迹加速度
01 10 17 86 00 02 04 00 00 13 88 9D 43	设置轨迹减速度
01 06 17 71 00 03 9C 64	设置控制模式为绝对位置控制。

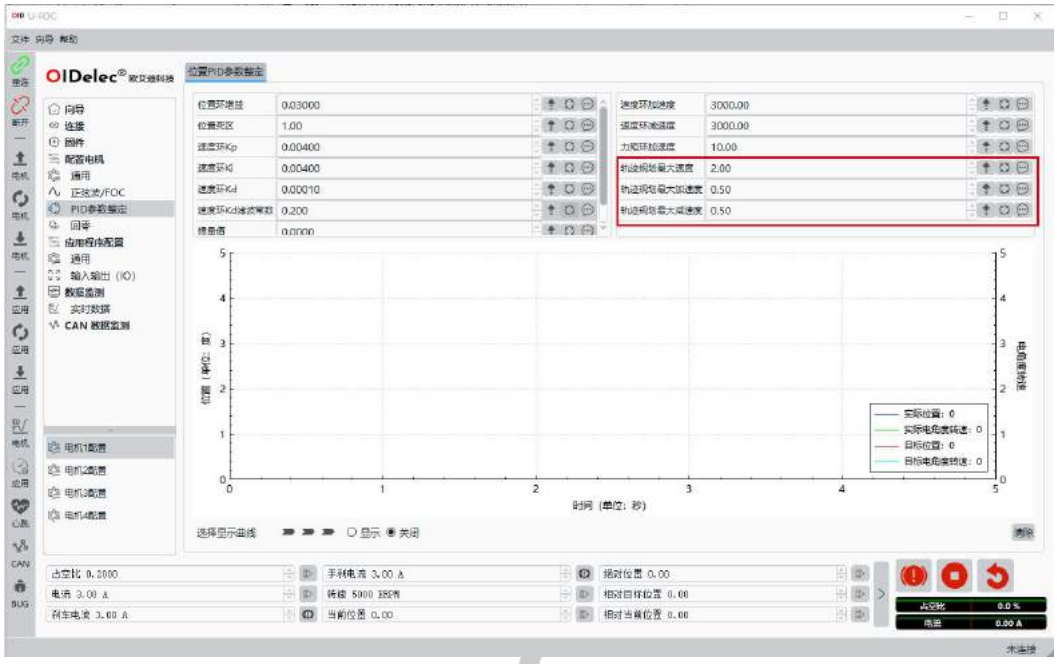
7.6. 相对上一次目标位置控制

7.6.1. 描述

相对位置控制是基于某个位置点走相对的增量。比如上次发送的指令是走绝对位置到 36000 位置，在还

未到目标点时候，使用此指令，又相对走 36000。这样电机最终会走 72000。即相对零点转动 2 圈。可以通过轨迹最大速度、轨迹加速度、轨迹减速度控制位置控制的轨迹。

如果通信不设置，使用的就是驱动保存的配置值。可以上位机查看和修改。



注：控制的前提是心跳一直在更新。见心跳保护机制。

7.6.2. 指令示例

例:如果驱动器ID:1,相对上一次目标值走36000。单位是0.01度,即360度。最大速度设置为5000erpm,加速度和减速度都设置为 5000erpm/s²。

注：位置指令仅在模式切换的那一刻生成一段路径规划。即控制位置之前需要把模式切换到空模式（或非目标模式皆可）。

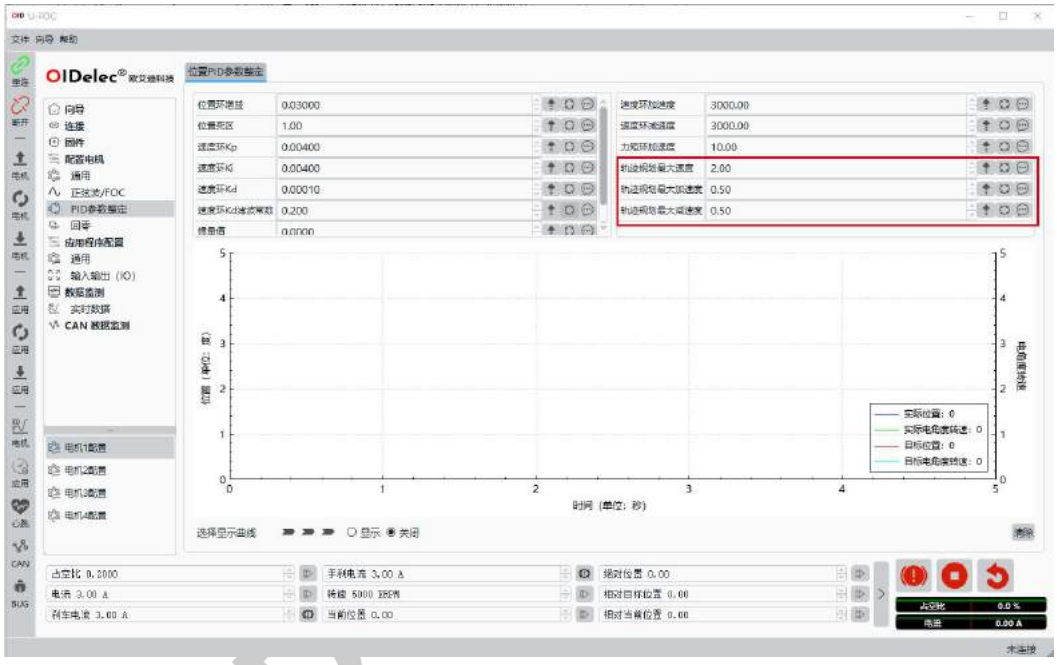
指令	说明
01 06 17 71 FF FF DD D5	设置为空模式。
01 10 17 78 00 02 04 00 00 8C A0 7A 65	设置目标位置
01 10 17 82 00 02 04 00 00 13 88 9C B0	设置轨迹最大速度
01 10 17 84 00 02 04 00 00 13 88 1C 9A	设置轨迹加速度
01 10 17 86 00 02 04 00 00 13 88 9D 43	设置轨迹减速度
01 06 17 71 00 04 DD A6	设置控制模式为相对上一次目标位置控制。

7.7. 相对当前位置控制

7.7.1. 描述

相对当前位置控制是基于当前接收到指令这一刻的位置走相对的增量。可以通过轨迹最大速度、轨迹加速度、轨迹减速度控制位置控制的轨迹。

如果通信不设置，使用的就是驱动保存的配置值。可以上位机查看和修改。



注：控制的前提是心跳一直在更新。见心跳保护机制。

7.7.2. 指令示例

例：如果驱动器 ID：1，相对当前位置走 36000。单位是 0.01 度，即 360 度。最大速度设置为 5000erpm，加速度和减速度都设置为 5000erpm/s²。

注：位置指令仅在模式切换的那一刻生成一段路径规划。即控制位置之前需要把模式切换到空模式（或非目标模式皆可）。

指令	说明
01 06 17 71 FF FF DD D5	设置为空模式。
01 10 17 7A 00 02 04 00 00 8C A0 FB BC	设置目标位置
01 10 17 82 00 02 04 00 00 13 88 9C B0	设置轨迹最大速度

01 10 17 84 00 02 04 00 00 13 88 1C 9A	设置轨迹加速度
01 10 17 86 00 02 04 00 00 13 88 9D 43	设置轨迹减速度
01 06 17 71 00 05 1C 66	设置控制模式为相对当前位置控制。

7.8. 设置当前位置

7.8.1. 描述

设置当前位置为设定值。典型应用是设置零点。比如机构运行到某个关键点时，设置当前位置为 0，即完成了零点的设置。

7.8.2. 指令示例

例：如果驱动器 ID：1，设置当前位置为零点。

指令	说明
01 10 17 7C 00 02 04 00 00 00 00 1F 2E	设置当前位置为零点。

7.9. 刹车控制

7.9.1. 描述

刹车为电子刹车，刹车力度和刹车电流大小呈正相关性。和转速也呈正相关性。即速度越大，刹车电流越大，刹车效果越明显。

仅正值有效。

注：控制的前提是心跳一直在更新。见心跳保护机制。

注：刹车方式为再生制动，会引起电源电压抬升。如果是开关电源供电，需增加能量耗散单元或斩波器等装置吸收刹车能量。电池系统则自身拥有吸收能量的能力，可不用额外增加。

7.9.2. 指令示例

例：如果驱动器 ID：1，使用刹车电流 1A 刹停电机。

指令	说明
01 06 17 7E 00 64 ED 8D	设置刹车电流 1A。因为电流单位是 10mA。所以 1A=100*10mA
01 06 17 71 00 06 5C 67	设置控制模式为刹车控制。

7.10. 手刹控制

7.10.1. 描述

手刹实际为强行对准到某个电角度。可以达到类似于手刹的效果。

仅正值有效。

注：控制的前提是心跳一直在更新。见心跳保护机制。

注：一定在极低速或者 0 速使用手刹功能，否则会损坏电机或驱动器！

注：一定在极低速或者 0 速使用手刹功能，否则会损坏电机或驱动器！

注：一定在极低速或者 0 速使用手刹功能，否则会损坏电机或驱动器！

7.10.2. 指令示例

例：如果驱动器 ID：1，使用手刹电流 1A 使电机制动。

指令	说明
01 06 17 7F 00 64 BC 4D	设置手刹电流 1A。因为电流单位是 10mA。所以 1A=100*10mA
01 06 17 71 00 07 9D A7	设置控制模式为手刹控制。

7.11. 更改电机当前使用的配置表

7.11.1. 描述

驱动器内部一共可以保存 4 份配置表。（有效值是 0~3）通过此命令可更换到其他编号的配置表。配置表的参数需要在上位机上配置识别好后才能使用。

上位机切换配置的接口如下图：

注意：切换指令不保存，开机默认为 1 号配置。所以正常使用请默认为 1 号配置。

7.11.2. 指令示例

例：如果驱动器 ID：1，使用 2 号配置表。

指令	说明
01 06 17 88 00 01 CD 94	使用 2 号配置表。因为内部索引计数是从 0 开始的，所以 2 号配置表索引值为 1。

7.12. 回零

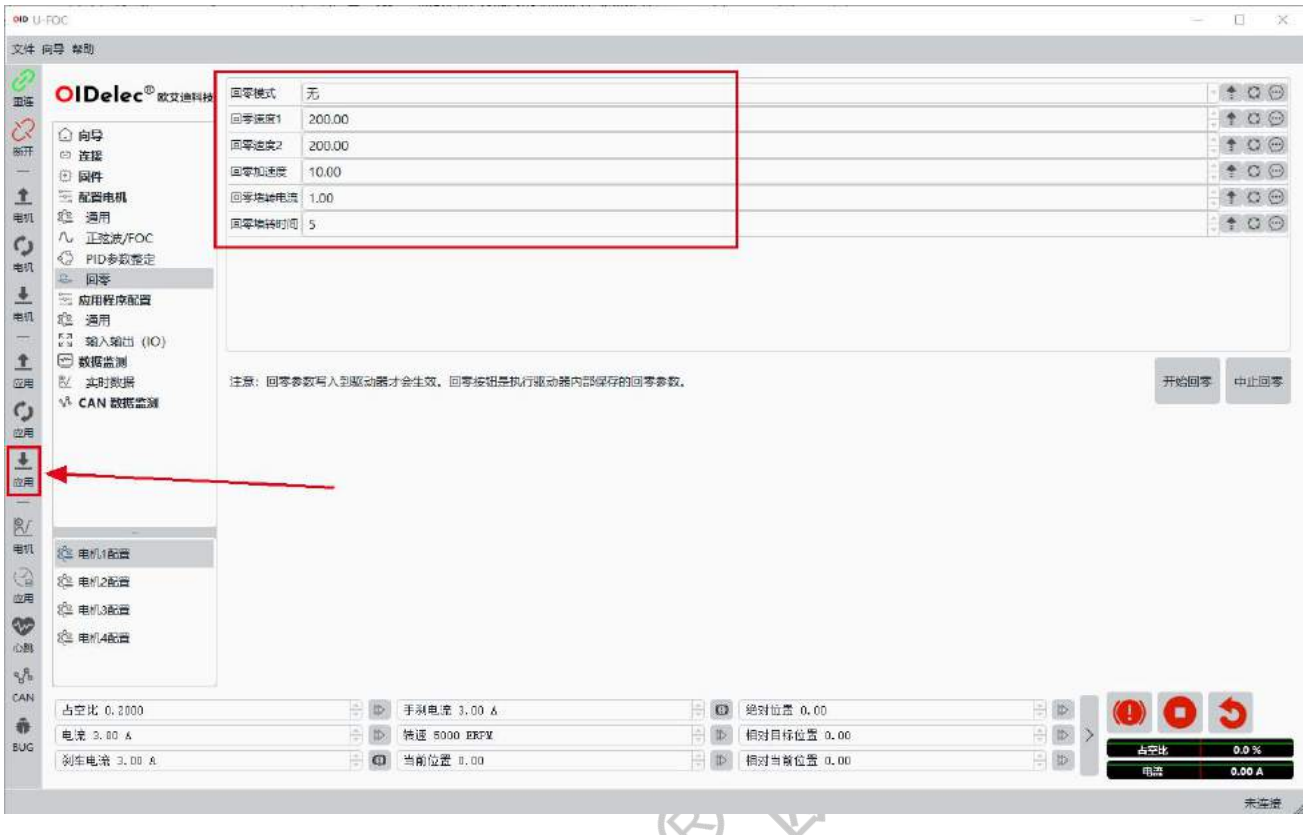
7.12.1. 描述

可以使用的传感器包括负极限开关、正极限开关、零点开关、堵转。

注意：如果使用传感器开关作为回零方式，需要在上位机上配置 IO 为相应的功能。否则回零失败。

回零的速度、堵转电流、堵转时间等在上位机上设置。**保存应用配置到驱动器才生效。**

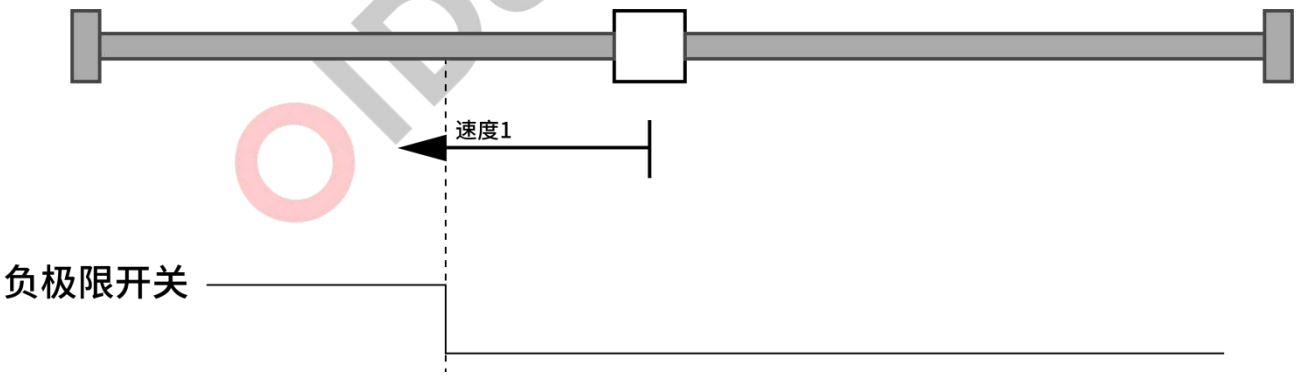
注：控制的前提是心跳一直在更新。见心跳保护机制。



7.12.2. 回零方法图形示意

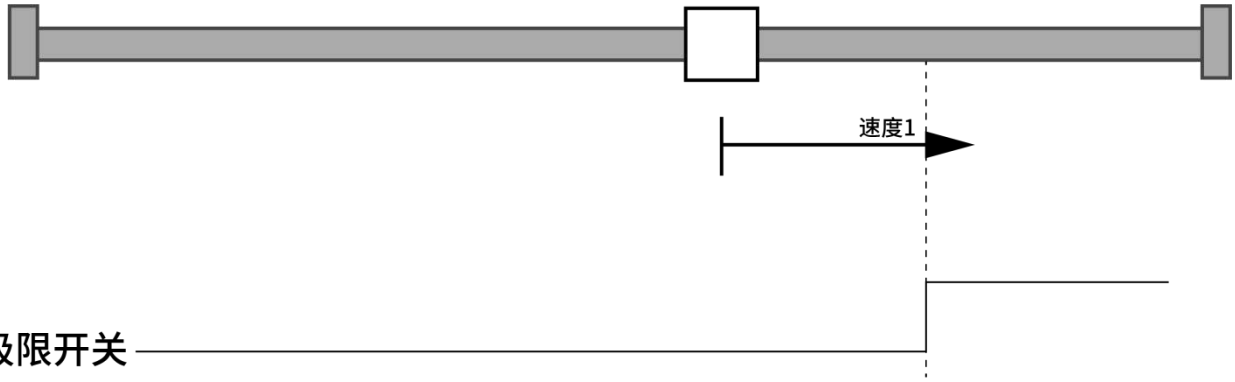
- 负极限开关触发有效电平为零点(6027=0x01)

  电机此时按照速度 1 朝负方向转动，负极限开关有效电平时设置为零点，并停止电机。零点设置完成。



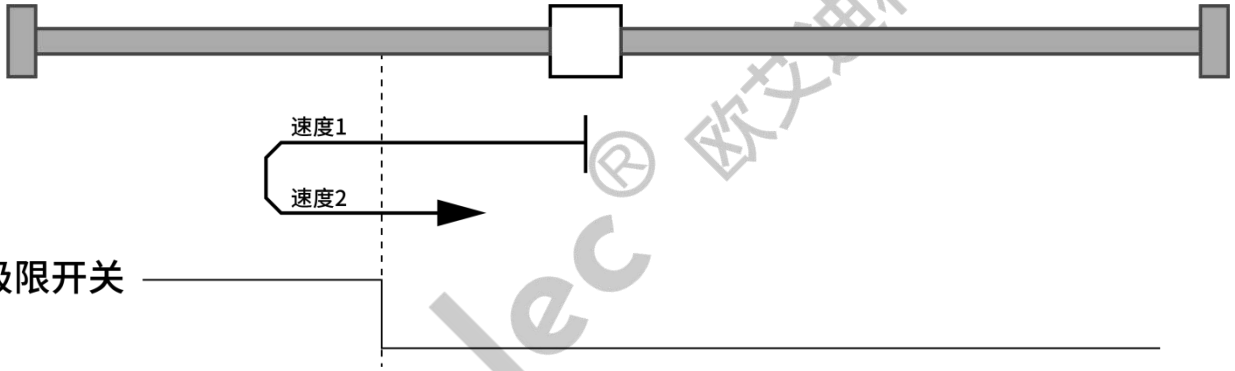
- 正极限开关触发有效电平为零点(6027=0x02)

  电机此时按照速度 1 朝正方向转动，正极限开关有效电平时设置为零点，并停止电机。零点设置完成。



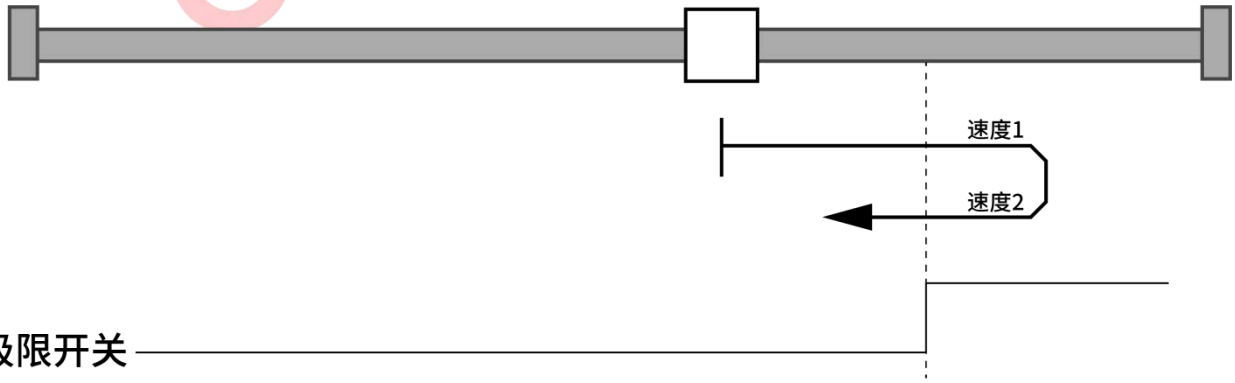
- 负极限开关触发有效电平后反向转动到无效电平为零点(6027=0x03)

电机此时按照速度 1 朝负方向转动，负极限开关有效电平后，按速度 2 朝正方向转动，负极限开关无效电平时设置为零点，并停止电机。零点设置完成。



- 正极限开关触发有效电平后反向转动到无效电平为零点(6027=0x04)

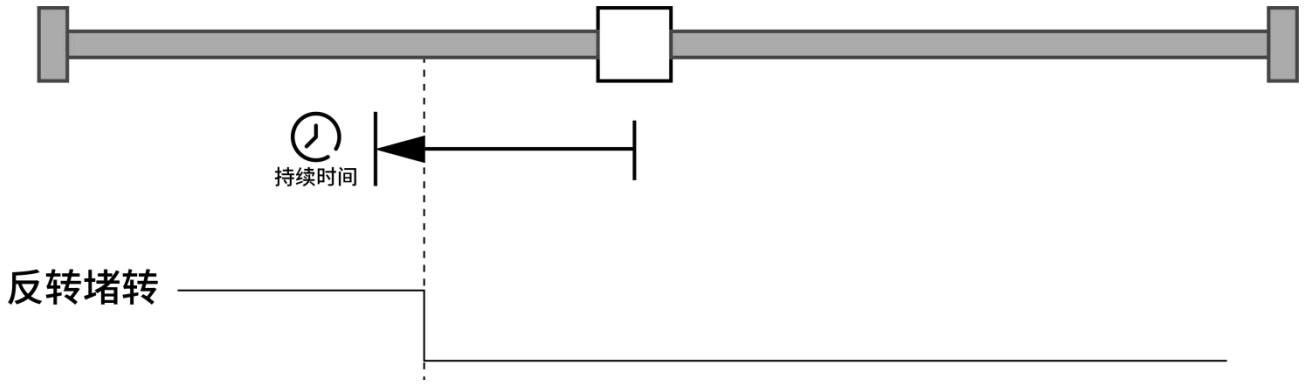
电机此时按照速度 1 朝正方向转动，正极限开关有效电平后，按速度 2 朝负方向转动，正极限开关无效电平时设置为零点，并停止电机。零点设置完成。



- 负方向堵转为零点(6027=0x05)

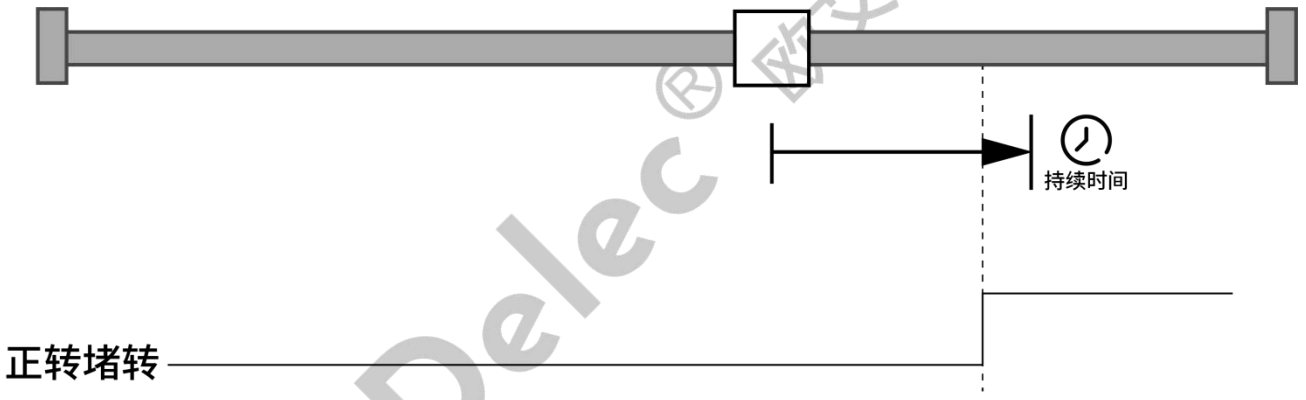
电机此时按照速度 1 朝负方向转动，堵转后在堵转电流满足设置值保持设置时间后设置为零点，并停止

电机。零点设置完成。（堵转电流和堵转时间在上位机设置后固化在驱动器内部，不可实时修改）



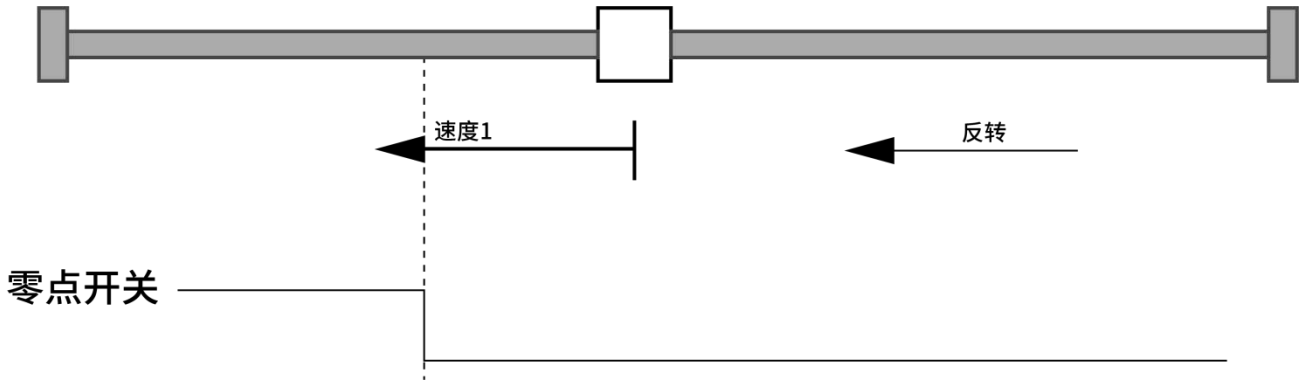
- 正方向堵转为零点(6027=0x06)

电机此时按照速度 1）朝正方向转动，堵转后在堵转电流满足设置值保持设置时间后设置为零点，并停止电机。零点设置完成。（堵转电流和堵转时间在上位机设置后固化在驱动器内部，不可实时修改）



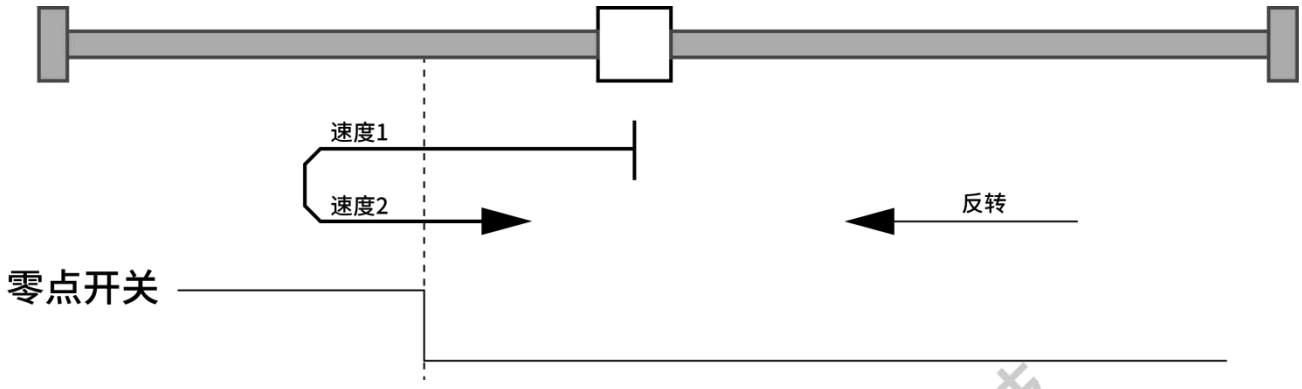
- 零点开关在负方向触发有效电平为零点(6027=0x07)

电机此时按照速度 1 朝负方向转动，零点开关有效电平时设置为零点，并停止电机。零点设置完成。



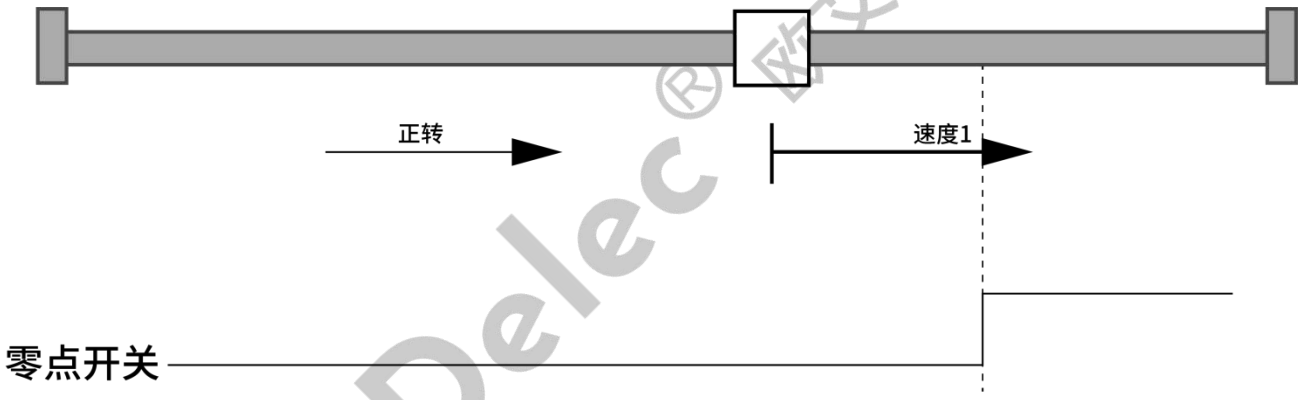
- 零点开关在负方向触发有效电平后反向转动到无效电平为零点(6027=0x08)

电机此时按照速度 1 朝负方向转动，零点开关有效电平后，按速度 2 朝正方向转动，零点开关无效电平时设置为零点，并停止电机。零点设置完成。



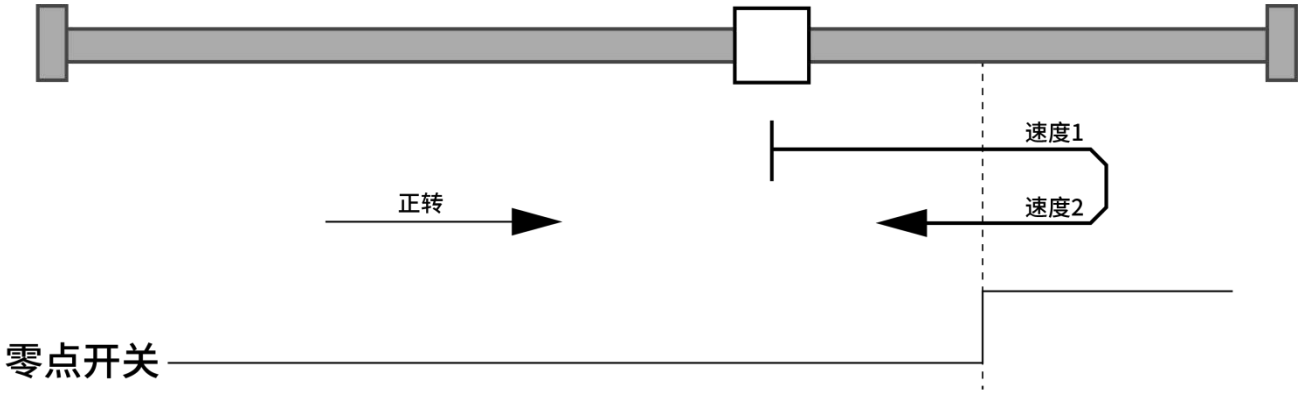
- 零点开关在正方向触发有效电平为零点(6027=0x09)

  电机此时按照速度 1 朝正方向转动，零点开关有效电平时设置为零点，并停止电机。零点设置完成。



- 零点开关在正方向触发有效电平后反向转动到无效电平为零点(6027=0x0A)

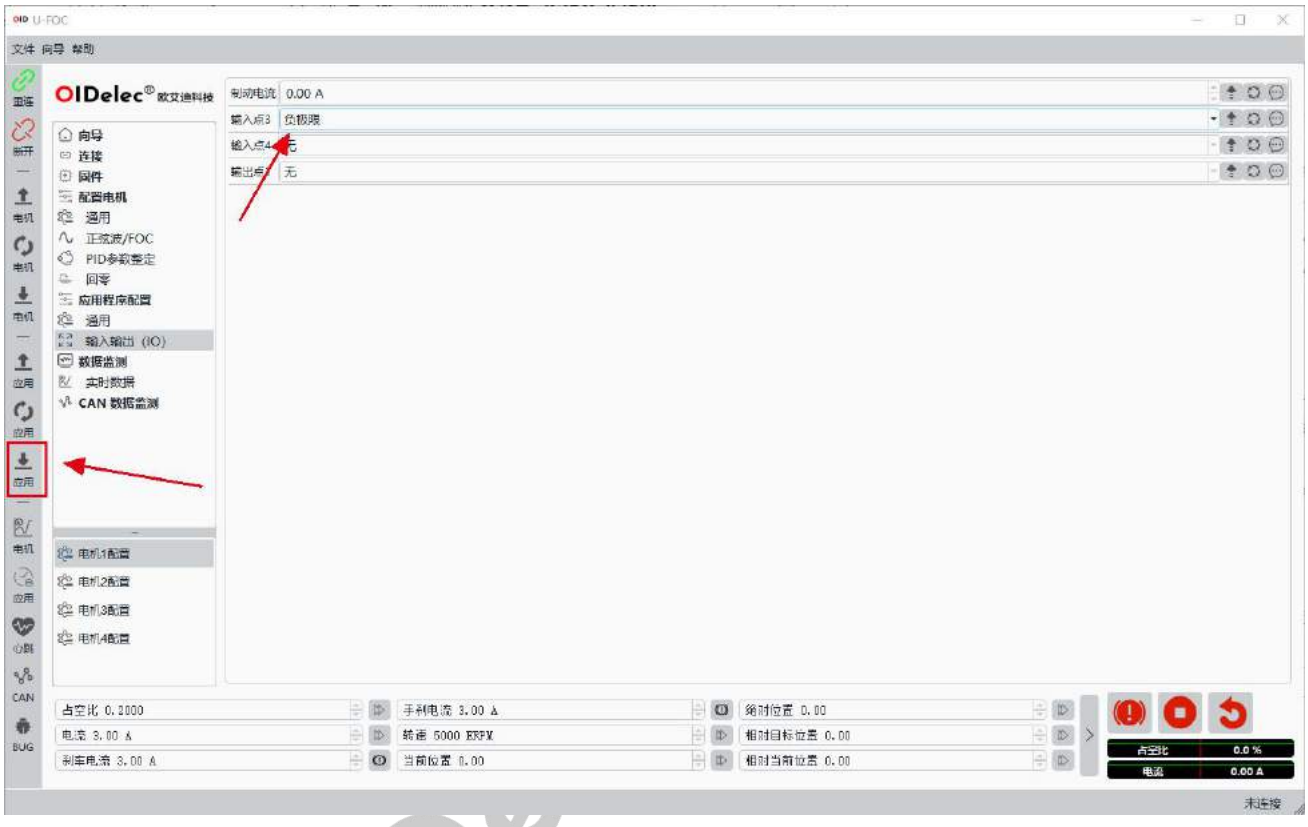
  电机此时按照速度 1 朝正方向转动，零点开关有效电平后，按速度 2 朝负方向转动，零点开关无效电平时设置为零点，并停止电机。零点设置完成。



7.12.3. 指令示例

例：如果驱动器 ID：1，使用 1 号负极限开关触发有效电平为零点回零。

首先需要在上位机配置 IO 功能为负极限开关。



注：回零指令仅在模式切换的那一刻开始执行。即控制位置之前需要把模式切换到空模式（或非目标模式皆可）。

指令	说明
01 06 17 71 FF FF DD D5	设置为空模式。
01 06 17 8B 00 01 3D 94	设置回零模式 01
01 06 17 71 00 08 DD A3	设置控制模式为回零。

如果需要中止回零

指令	说明
01 06 17 71 00 09 1C 63	中止回零。

7.13. 闭环模式设置最大扭矩

7.13.1. 描述

如果速度闭环、位置闭环的时候又想控制扭矩的大小，可以设置此寄存器动态修改扭矩大小。

7.13.2. 指令示意

例：如果驱动器 ID：1，设置闭环状态下最大输出电流 1A。

指令	说明
01 06 17 8C 00 64 4C 7E	闭环状态下最大输出电流 1A。因为电流单位是 10mA。所以 1A=100*10mA

7.14. 电流爬升控制

7.14.1. 描述

电流控制即恒扭矩控制（电流闭环）。此时速度根据负载很变化。负载大转速就低，负载小转速就高。

正电流电机就正转，负电流电机就反转。

爬升电流的区别是有个斜坡加扭矩的过程。

注：控制的前提是心跳一直在更新。见心跳保护机制。

7.14.2. 指令示例

例：如果驱动器 ID：1，以 10%最大电流控制电机。爬升速度为 1S 爬升到 10%电流值。

指令	说明
01 06 17 8D 00 64 1D BE	设置电流爬升加速度为 100。单位为千分比最大电流每秒，100/1000=10%。
01 06 17 8E 00 64 ED BE	设置目标电流值 100。单位为千分比千分比最大电流，100/1000=10%。
01 06 17 71 00 0A 5C 62	设置控制模式为电流爬升控制。

8. 查询驱动器信息指令示意

8.1. 描述

驱动器自身的信息可通过读取相关只读寄存器获取。见驱动器寄存器地址。

8.2. 读取故障信息

例：地址为 1，无故障 0

方向	指令	说明
发送	01 04 13 88 00 01 B5 64	读取故障信息
接收	01 04 02 00 00 B9 30	返回无故障 0

8.3. 读取转速

例：地址为 1，读取速度为 1345，如果电机为 4 对极，此时转速为 1345÷4=336.25rpm

方向	指令	说明
发送	01 04 13 89 00 02 A4 A5	读取转速
接收	01 04 04 00 00 05 41 38 E4	返回 1345erpm

8.4. 读取实际占空比

例：地址为 1，读取占空比数值为 100

方向	指令	说明
发送	01 04 13 8B 00 01 45 64	读取实际占空比
接收	01 04 02 00 64 B8 DB	返回 100，即 10%占空比

8.5. 读取功率

例：地址为 1，读取功率数值为 0

方向	指令	说明
发送	01 04 13 8C 00 01 F4 A5	读取实际功率
接收	01 04 02 00 00 B9 30	返回 0，单位 W

8.6. 读取输入电压

例：地址为 1，读取输入电压数值为 48

方向	指令	说明
发送	01 04 13 8D 00 01 A5 65	读取输入电压
接收	01 04 02 00 30 B9 24	返回 48，单位 V

8.7. 读取电机电流

例：地址为 1，读取电机电流数值为 423

方向	指令	说明
发送	01 04 13 8E 00 01 55 65	读取电机电流
接收	01 04 02 01 A7 F9 1A	返回 423，单位 10mA。即 423*10mA=4.23A。

8.8. 读取总线电流

例：地址为 1，读取输入电流数值为 0

方向	指令	说明
发送	01 04 13 8F 00 01 04 A5	读取总线电流
接收	01 04 02 00 00 B9 30	返回 0，单位 10mA。

8.9. 读取温度

例：地址为 1，读取温度数值为 27

方向	指令	说明
发送	01 04 13 90 00 01 35 63	读取温度
接收	01 04 02 00 1B F9 3B	返回 27，单位度

8.10. 读取角度

例：地址为 1，读取角度数值为 90，实际值放大了 100 倍，所以实际值为 0.9 度

方向	指令	说明
发送	01 04 13 91 00 01 64 A3	读取角度
接收	01 04 02 00 5A 39 0B	返回 90，单位 0.01 度。即 90*0.01=0.9 度

8.11. 读取位置

例：地址为 1，读取角度数值为 28890，实际值放大了 100 倍，所以实际值为 288.9 度

方向	指令	说明
发送	01 04 13 92 00 02 D4 A2	读取位置，即角度的积分
接收	01 04 04 00 00 70 DA 5F DF	返回 28890，单位 0.01 度。即 28890*0.01=288.9 度

8.12. 回零状态

例：地址为 1，读取到回零成功。

方向	指令	说明
发送	01 04 13 94 00 01 74 A2	读取回零状态
接收	01 04 02 01 00 B8 A0	高 8 位：回零状态；0 代表回零中，1 代表回零完成。 低 8 位：回零代码；0 代表回零成功；-1 代表相应 IO 未设置功能；-2 代表中止回零。

9. 附录

9.1. CRC 示例 C 语言代码

```
static const uint8_t aucCRCHi[] = {
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40
};

static const uint8_t aucCRCLo[] = {
    0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07, 0xC7,
    0x05, 0xC5, 0xC4, 0x04, 0xCC, 0x0C, 0x0D, 0xCD, 0x0F, 0xCF, 0xCE, 0x0E,
    0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09, 0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9,
    0x1B, 0xDB, 0xDA, 0x1A, 0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC,
    0x14, 0xD4, 0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
    0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF2, 0x32,
    0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4, 0x3C, 0xFC, 0xFD, 0x3D,
    0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A, 0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38,
    0x28, 0xE8, 0xE9, 0x29, 0xEB, 0x2B, 0x2A, 0xEA, 0xEE, 0x2E, 0x2F, 0xEF,
    0x2D, 0xED, 0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
    0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60, 0x61, 0xA1,
```

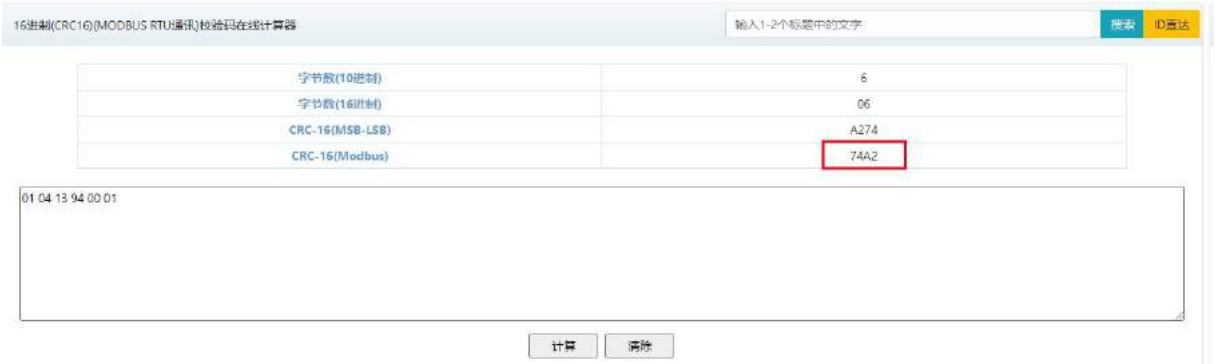
```
    0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67, 0xA5, 0x65, 0x64, 0xA4,
    0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F, 0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB,
    0x69, 0xA9, 0xA8, 0x68, 0x78, 0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA,
    0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
    0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0,
    0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92, 0x96, 0x56, 0x57, 0x97,
    0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C, 0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E,
    0x5A, 0x9A, 0x9B, 0x5B, 0x99, 0x59, 0x58, 0x98, 0x88, 0x48, 0x49, 0x89,
    0x4B, 0x8B, 0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
    0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83,
    0x41, 0x81, 0x80, 0x40
};

uint16_t
usMBCRC16( uint8_t* pucFrame, uint16_t usLen )
{
    uint8_t           ucCRCHi = 0xFF;
    uint8_t           ucCRCLo = 0xFF;
    int             iIndex;

    while( usLen-- )
    {
        iIndex = ucCRCLo ^ *( pucFrame++ );
        ucCRCLo = ( uint8_t)( ucCRCHi ^ aucCRCHi[iIndex] );
        ucCRCHi = aucCRCLo[iIndex];
    }
    return ( uint16_t)( ucCRCHi << 8 | ucCRCLo );
}
```

9.2. 在线计算 CRC 网站

https://www.23bei.com/tool/59.html



欧艾迪驱动器快速 入门教学视频

官网二维码

联系我们

深圳欧艾迪科技有限公司

网址: www.oidelec.com

电话: 400-166-0195

15226313566 余经理(微信同号)

邮箱: support@oidencoder.com

地址： 深圳市宝安区西乡街道盐田社区银田工业区 B9 栋 3 层