软件操作目录

软件启动界面

➢ CAN 参数设置：

1） 帧格式

标准帧：含有 11 位（bit）ID

扩展帧：在标准帧 ID 位数基础上，扩展出 18 位 ID，共 29 位表示扩展帧 ID

若选择标准帧，则帧 ID 就是“从机地址”。默认使用标准帧；

若选择帧格式为扩展帧，则帧 ID 的 11 位标准帧部分为“从机地址”，18 位的扩展位部分全部为 0，以此组成 29 位扩展帧 ID

2） 发送方式

正常发送：若终端因某些原因（仲裁失败、CAN 总线未正确连接等）未成功发送报文，终端将会不断重复发送报文，直至发送成功。默认使用正常发送方式

单次发送：终端只会发送一次 CAN 报文，不论发送成功还是失败。

3） 通信速率

设置终端的 CAN 收发速率。默认使用 500kbps

下拉列表中罗列了常用的从 **5kbps 至 1000kbps** 的通信速率供选择。

4） 硬件滤波

设置 Modbus-CAN 模块对其接收到的 CAN 报文 ID 进行过滤。默认无滤波

使能滤波：

勾选后点击确定，软件完成对终端滤波器的设置，同时保存滤波列表中的设置内容；

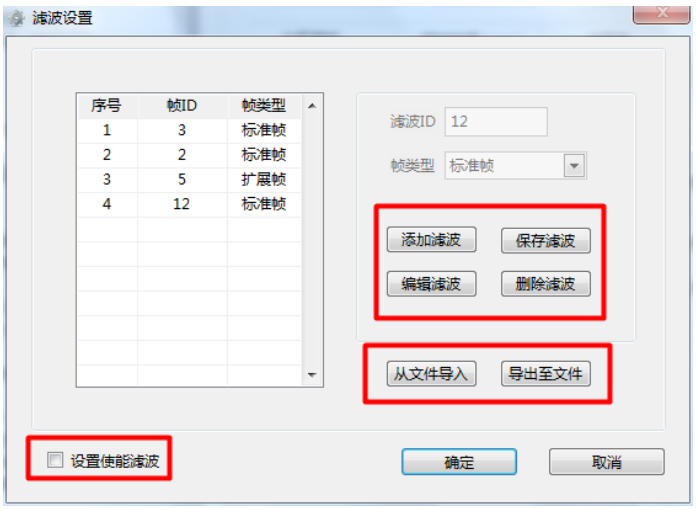
不勾选时点击确定，软件只保存滤波列表，下次打开时自动加载；

点击取消，软件将不会保存任何设置及内容。

禁能滤波：

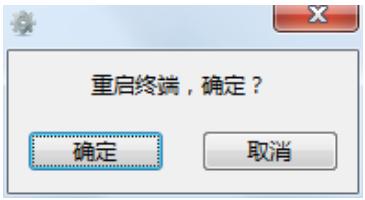
先删除滤波列表中的所有项，并勾选“设置使能滤波”，最后点击确定按钮

设置硬件滤波时，只有与列表中**帧 ID**（帧 ID 输入时为**十六进制**数字）和**帧类型**都相同的报文才能被 Modbus-CAN 模块接收到，其余报文将被舍弃。滤波设置界面如图所示



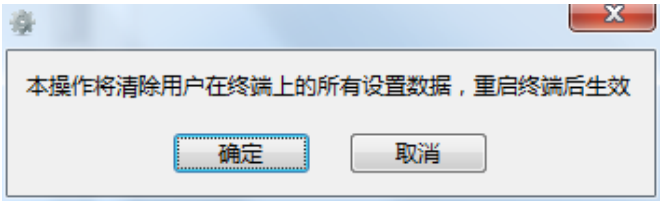
5） 终端复位

与模块重新上电相同。功能有弹窗提示，复位后终端内的参数不会丢失。如图所示。



6） 清除数据

清除保存在 Modbus-CAN 模块中的 CAN 配置数据。如图所示：



清除数据操作后，必须进行终端复位或对终端重新上电，完成数据清除过程。

数据清除后，终端将恢复至默认设置：

“500kbpsCAN 通信波特率”、“标准帧”、“正常发送”、“无硬件滤波”、 “串口 57600 波特率” 、 “无校验”

7） 关于终端

状态栏将显示当前连接终端的软件版本号信息。

软件与 Modbus-CAN 模块建立连接后，会自动读取其配置信息。

打开软件菜单的下拉列表，表中已勾选项目为终端当前的配置信息。

➢ 串口参数设置：

1） 串口号：选择与 Modbus-CAN 终端连接的串口号

2） 波特率：串口通信速率设置（建议使用高波特率，默认 57600）

3） 校验位：串口通信校验位设置

4） **MC/M 终端：**使用 Modbus-CAN 模块作为通信对象，选择“MC 终端”

使用通用 Modbus 终端作为通信对象，选择“M 终端”

点击“打开串口”，Modbus-CAN 模块**自动**与本软件设置的串口参数适配。

➢ 数据监视：

1）Modbus 数据监视

从机地址：Modbus-CAN 模块下发数据时的目标地址，默认为 1

超时时间：本软件发送 Modbus 报文到未收到终端回复的最长等待时间

实时更新显示：Modbus 显示列表是否实时显示交互的 Modbus 报文内容

自动清空显示：当 Modbus 显示列表显示数量超过 1000 条时软件自动清空显示

2）CAN 数据监视

总线状态：CAN 总线当前状态以及 Modbus-CAN 模块收发错误数

实时更新显示：本软件发送 Modbus 报文到未收到终端回复的最长等待时间

自动清空显示：CAN 显示列表显示数量超过 1000 条时软件自动清空显示

➢ 状态显示：

1）状态指示：Modbus-CAN 模块与本软件连接情况（红绿指示灯）

2）系统信息显示：系统提示信息或警告

➢ 测试界面：

1） 专项测试

2） 通用测试

点击“测试界面”右上角“专项测试”或“通用测试”文字，在两种模式间切换。

1）专项测试：（未使用我公司”CAN 驱动控制一体系统”者可以跳过）

专项测试用于对挂载在 Modbus-CAN 模块下的我公司的“CAN 驱动控制一体系统”进行测试。

基本操作说明：

① 点击不同图标，进入不同设置界面。

② 用户输入某一参数值后，需点击回车、或点击界面空白处、或直接点击其他输入框继续输入其他参数，以此确认先前的参数值输入完毕（**输入框光标不闪烁即表示当前输入完毕**）

① 基本参数：

细分：细分越大，电机运行越平滑。通过这个参数就可以知道电机转 1 圈需要的脉冲个数。默认设置 8 细分，就是 1600 个脉冲 1 圈。其他细分情况下，以此类推。

步距角：电机固有参数。默认设置 1.8。

启动频率：由静止突然启动并进入不失步的正常运行所容许的最高频率。单位 HZ，默认设置 50HZ

加减频率：从启动速度慢慢加速到运行速度的一个频率；单位 HZ，默认设置 50HZ

螺距：电机转 1 圈所对应平台移动的距离。默认设置为 1；

正反限位信号设置：可设置为 1-5(对应 IN1-IN5 输入口)；

默认设为 0，表示无输入设置。

启动信号：可设置为 1-5(对应 IN1-IN5）,用来外部启动工程。

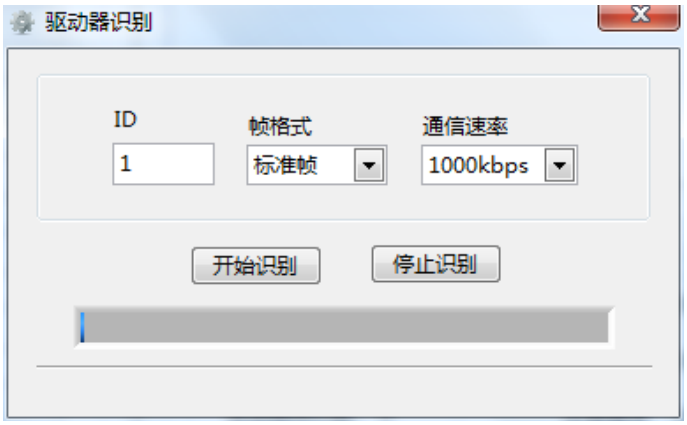
停止信号：可设置为 1-5(对应 IN1-IN5)，用来外部停止工程。

往返次数：也就是设置控制方式 3 的运行次数。默认设为 1

为 0，无限循环；为 1，循环 1 次，以此类推。

驱动器通信参数：

参数识别：



Modbus-CAN 模块对其连接的驱动器参数进行识别，识别成功后驱动器参数直接显示在“ID” “帧格式”和“通信速率”中

若忘记驱动器参数，可点击“开始识别”对驱动器进行识别，但**可能用时较长。**

参数配置：直接将主界面中的“ID” “帧格式”和“通信速率”配置给驱动器。

注意：1、配置时需设置 Modbus-CAN 模块与驱动器 CAN 通信参数一致。

2、配置成功后需重启驱动器。

② **手动测试：**

测试速度：单位是转/每分钟；一般不超过 600 转/每分钟

默认设为 200 转/分钟。

测试距离：单位可以看做是圈或 mm 等。默认设置为 1。

比如螺距为 1，运行距离为 2，则对应电机转 2 圈；

③ **工程参数：**

工程号：默认为 1

工程总步数：最大可设为 33

设定第 XX 步的参数：1 表示第 1 步参数，2 表示第 2 步参数....

输入 xx 有效停止本步：可设为 0-5，0 表示无信号控制；

1 表示，IN1 信号控制本步停止；2 表示，IN2 信号控制本步停止；

输入 xx 有效启动本步：可设为 0-5，0 表示无信号控制

1 表示，IN1 信号控制本步停止；2 表示，IN2 信号控制本步停止；

启动频率：同基本参数介绍

加减频率：同基本参数介绍

运行方向：即本步运行的方向。（0 正转，1 反转）

运行速度：即本步电机运行的速度，单位转/每分钟。

运行距离：即本步电机运行的距离。

本步输出：即本步是否输出一个信号，可用来控制电磁阀等。

(1 表示 oc1 开，2 表示 oc1 关；3 和 4 表示 OC2 控制；

5 和 6 表示 oc3 控制；7 和 8 表示 3 路全部控制；)

本步运行完毕延时：即本步运行完的延时时间，单位是毫秒。

第 xx 步到第 yy 步循环 zz 次：相当于跳转功能。

比如在第 7 步的时候设定为从第 3 步到第 5 步循环 1 次，

那么在运行完第 7 步之后会跳转到第 3 步，然后第 4，第 5 步，

然后就运行第 8 步，直到本工程最后 1 步结束。

工程循环次数：也就是第 1 步到最后 1 步循环动作的次数。

④ **自动运行：**

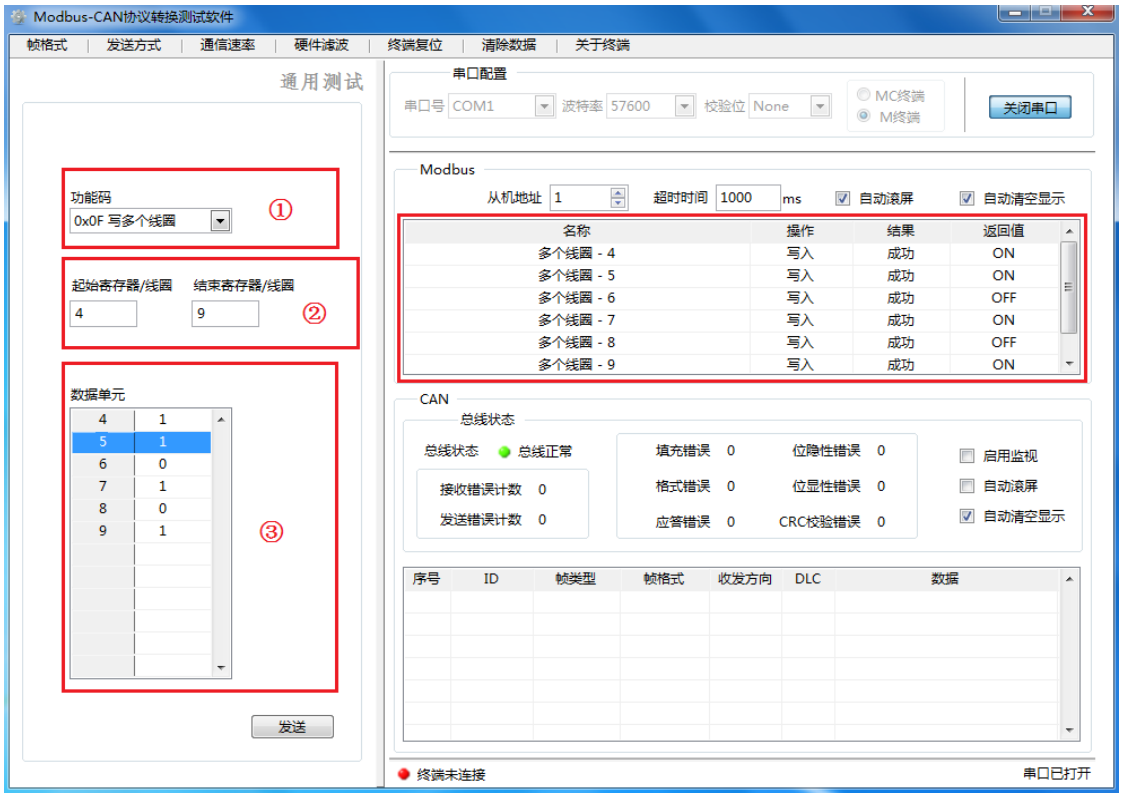
控制电机按照工程参数中的设定自动运行。

2）通用测试：

通用测试实现常用 Modbus 报文的收发测试：

0x01	**读线圈**
0x02	**读离散量输入**
0x03	**读保持寄存器**
0x04	**读输入寄存器**
0x05	**写单个线圈**
0x06	**写单个寄存器**
0x0F	**写多个线圈**
0x10	**写多个寄存器**

*软件只显示查询到的最终结果，即**寄存器值**或**线圈状态**或**离散量状态。**



① **功能选择**：选择功能码进行操作

② **起止地址**：设置读取/写入操作目标（寄存器/线圈）的起始和终止地址

③ **数据单元**：写入操作时，设置目标寄存器的值或线圈的状态。

参数输入完成后，点击“**发送**”进行终端读写。

Modbus 监视列表中：

“名称”列显示**查询类型**和**地址值**，“返回值”列显示各地址对应的**返回值**。

附 1：Modbus 常用功能码详细说明及举例

① 01（0x01）读线圈

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。因此寻址线圈 1-16 为 0-15

根据数据域的每个比特将响应报文中的线圈分成一个线圈。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。其他线圈依次类推，一直到这个字节的高位端位置，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
起始地址	1 个字节	N*
线圈数量	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

功能码	1 个字节	功能码+0x80
异常码	1 个字节	01 或 02 或 03 或 04

这是一个请求读离散量输出 20-38 的实例：

请求		响应	
域名	（十六进制）	域名	（十六进制）
功能	01	功能	01
起始地址 Hi	00	字节数	03
起始地址 Lo	13	输出状态 27-20	CD
输出数量 Hi	00	输出状态 35-28	6B
输出数量 Lo	13	输出状态 38-36	05

将输出 27-20 的状态表示为十六进制字节值 CD，或二进制 1100 1101。输出 27 是这个字节的 MSB，输出 20 是 LSB。

通常，将一个字节内的比特表示为 MSB 位于左侧，LSB 位于右侧。第一字节的输出从左至右为 27 至 20。下一个字节的输出从左到右为 35 至 28。当串行发射比特时，从 LSB 向 MSB 传输：20...27、28...35 等等。

在最后的数据字节中，将输出状态 38-36 表示为十六进制字节值 05，或二进制 0000 0101。输出 38 是左侧第六个比特位置，输出 36 是这个字节的 LSB。用零填充五个剩余高位比特。

② 02（0x02）读离散量输入

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。因此寻址输入 1-16 位 0-15.

根据数据域的每隔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。其他

输入依次类推，一直到这个字节的高位端位置，并在后续字节中从低位到高位的顺序。

如果返回的输入量不是八的倍数，将用零填充最后的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
线圈数量	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*
输入状态	N*×1 个字节	

*N=输出数量/8，如果余数不等于 0，那么 N=N+1

错误

差错码	1 个字节	0x82
异常码	1 个字节	01 或 02 或 03 或 04

这是一个请求读取离散量输入 197-218 的实例：

请求		响应	
域名	（十六进制）	域名	（十六进制）
功能	02	功能	02
起始地址 Hi	00	字节数	03
起始地址 Lo	C4	输入状态 204-297	AC

输出数量 Hi	00	输入状态 212-205	DB
输出数量 Lo	16	输入状态 218-213	35

将离散量输入状态 204-197 表示为十六进制字节值 AC，或二进制 1010 1100。输入 204 是这个字节的 MSB，输入 197 是这个字节的 LSB。

将离散量输入状态 218-213 表示为十六进制字节值 35，或二进制 0011 0101。输入 218 位于左侧第 3 比特，输入 213 是 LSB。用零填充 2 个剩余比特（一直到高位端）。

③ 03（0x03）读保持寄存器

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。因此寻址输入 1-16 位 0-15.

将响应报文中的寄存器数据分成每个寄存器有两字节，在每个字节中直接地调整二进制内容。

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

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

响应

功能码	1 个字节	0x03
字节数	1 个字节	2×N*
寄存器值	N*×2 个字节	

*N=寄存器的数量

错误

差错码	1 个字节	0x83
异常码	1 个字节	01 或 02 或 03 或 04

这是一个请求读寄存器 108-110 的实例：

请求		响应	
域名	（十六进制）	域名	（十六进制）
功能	03	功能	03
起始地址 Hi	00	字节数	06
起始地址 Lo	6B	寄存器值 Hi（108）	02
输出数量 Hi	00	寄存器值 Lo（108）	2B
输出数量 Lo	03	寄存器值 Hi（109）	00
		寄存器值 Lo（109）	00
		寄存器值 Hi（110）	00
		寄存器值 Lo（110）	64

将寄存器 108 的内容表示为两个十六进制字节值 02 2B，或十进制 555.将寄存器 109-110 的内容个分别表示为十六进制 00 00 和 00 64，或十进制 0 和 100。

④ 04（0x04）读输入寄存器

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。因此，寻址输入寄存器 1-16 为 0-15。

将响应报文中的寄存器数据分成每个寄存器为两字节 ,在每个字节中直接地调整二进制

内容。

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

功能码	1 个字节	0x04
起始地址	2 个字节	0x0000 至 0xFFFF
输入寄存器数量	2 个字节	0x0001 至 0x007D

响应

功能码	1 个字节	0x04
字节数	1 个字节	2×N*
输入寄存器	N*×2 个字节	

*N=输入寄存器的数量

错误

差错码	1 个字节	0x84
异常码	1 个字节	01 或 02 或 03 或 04

这是一个请求读输入寄存器 9 的实例：

请求		响应	
域名	（十六进制）	域名	（十六进制）
功能	04	功能	04
起始地址 Hi	00	字节数	02
起始地址 Lo	08	寄存器值 Hi（108）	00
输入寄存器数量 Hi	00	寄存器值 Lo（108）	0A

输入寄存器数量 Lo	01	

将输入寄存器 9 的内容表示为两个十六进制字节值 00 0A，或十进制 10。

⑤ 05（0x05）写单个线圈

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 FF 00 请求输出为 ON。十六进制 00 00 请求输出为 OFF。其他所有值均是非法的，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 0xFF00 请求线圈为 ON。十六进制 0x0000 请求线圈为 OFF。其他所有值均为非法的，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

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

响应

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

错误

差错码	1 个字节	0x85
异常码	1 个字节	01 或 02 或 03 或 04

这是一个请求写线圈 173 为 ON 的实例：

请求		响应	
域名	（十六进制）	域名	（十六进制）
功能	05	功能	04
输出地址 Hi	00	输出地址 Hi	02
输出地址 Lo	AC	输出地址 Lo	00
输出值 Hi	FF	输出值 Hi	FF
输出值 Lo	00	输出值 Lo	00

⑥ 06（0x06）写单个寄存器

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

功能码	1 个字节	0×06
寄存器地址	2 个字节	0x0000 至 0×FFFF
寄存器值	2 个字节	0x0000 至 0×FFFF

响应

功能码	1 个字节	0×06
寄存器地址	2 个字节	0x0000 至 0×FFFF
寄存器值	2 个字节	0x0000 至 0×FFFF

错误

差错码	1 个字节	0×86
异常码	1 个字节	01 或 02 或 03 或 04

这是一个请求将十六进制 00 03 写入寄存器 2 的实例：

请求		响应	
域名	（十六进制）	域名	（十六进制）
功能	06	功能	06
寄存器地址 Hi	00	输出地址 Hi	00
寄存器地址 Lo	01	输出地址 Lo	01
寄存器值 Hi	00	输出值 Hi	00
寄存器值 Lo	03	输出值 Lo	03

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。因此，寻找线圈 1 为 0。

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
起始地址	2 个字节	0x0000 至 0xFFFF
输出数量	2 个字节	0×0001 至 0×07B0

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
起始地址	2 个字节	0x0000 至 0xFFFF
输出数量	2 个字节	0×0001 至 0×07B0

错误

差错码	1 个字节	0x8F
异常码	1 个字节	01 或 02 或 03 或 04

这是一个请求从线圈 20 开始写入 10 个线圈的实例：

请求的数据内容为两个字节：十六进制 CD 01（二进制 1100 1101 0000 0001）。使用下列方法，二进制比特对应输出。

比特： 1 1 0 0 1 1 0 1 0 0 0 0 0 0 0 1

输出： 27 26 25 24 23 22 21 20 — — — — — — 29 28

传输的第一字节（十六进制 CD）寻址为输出 27-20，在这种设置中，最低有效比特寻址为最低输出（20）。

输出的下一字节（十六进制 01）寻址为输出 29-28，在这种设置中，最有效比特寻址为最低输出（28）。

应该用零填充最后数据字节中的未使用比特。

请求	响应

域名	（十六进制）	域名	（十六进制）
功能	0F	功能	0F
起始地址 Hi	00	起始地址 Hi	00
起始地址 Lo	13	起始地址 Lo	13
输出数量 Hi	00	输出数量 Hi	00
输出数量 Lo	0A	输出数量 Lo	0A
字节数	02		
输出值 Hi	CD		
输出值 Lo	01		

⑧ 16（0×10）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	0×10
起始地址	2 个字节	0×0000 至 0×FFFF
寄存器数量	2 个字节	0×0001 至 0×0078
字节数	1 个字节	2×N*
寄存器值	N*×2 个字节	值

*N=寄存器数量

响应 PDU

功能码	1 个字节	0×10
起始地址	2 个字节	0×0000 至 0×FFFF
寄存器数量	2 个字节	1 至 123（0×7B）

错误

差错码	1 个字节	0×90
异常码	1 个字节	1 或 02 或 03 或 04

这是一个请求将十六进制 00 0A 和 01 02 写入以 2 开始的两个寄存器的实例：

请求		响应	
域名	（十六进制）	域名	（十六进制）
功能	10	功能	10
起始地址 Hi	00	起始地址 Hi	00
起始地址 Lo	01	起始地址 Lo	01
寄存器数量 Hi	00	寄存器数量 Hi	00
寄存器数量 Lo	02	寄存器数量 Lo	02
字节数	04		
寄存器值 Hi	00		
寄存器值 Lo	0A		
寄存器值 Hi	01		
寄存器值 Lo	02		

附 2：上位机与 Modbus-CAN 模块间通信协议

协议概述：

上位机与 Modbus-CAN 模块间的交互的报文包括两种：

① 标准 Modbus 协议 ② 私有协议。

① Modbus 协议通过 Modbus-CAN 模块被转换成为 CAN 报文，与驱动器通信；

② 私有协议仅用于上位机与 Modbus-CAN 模块之间的通信。

1、 Modbus 通信协议

专项测试：使用定制的 Modbus 对驱动器进行操作，协议定义内容参见《modbus-rtu 协议地址》；

通用测试：使用标准 Modbus 协议对任意 Modbus 终端进行操作

2、私有协议

上位机对 Modbus-CAN 模块本身进行参数设置或操作时，使用**以下格式**的协议：

名称	字节数	说明
帧头	1	固定值 0xFF
帧长度	1	Len
功能码	1	Func
数据域	N	Data
校验和	1	模 256 和
帧尾	1	固定值 0x16

说明：1）帧头与帧尾为固定十六进制字节 0xFF 和 0x16

2）帧长度表示整个报文（**帧头到帧尾**）的长度

3）功能码为本帧功能含义，如下表所示：

功能码（十六进制）	含义
0x00	Modbus-CAN 模块串口参数设定
0x01	链路检测（①登陆 ②心跳）
0x02	查询终端版本号
0x03	Modbus-CAN 模块 CAN 帧格式设定
0x04	Modbus-CAN 模块 CAN 发送方式设定
0x05	Modbus-CAN 模块 CAN 通信速率设定
0x06	Modbus-CAN 模块硬件滤波设置
0x07	Modbus-CAN 模块软件复位
0x08	清除 Modbus-CAN 模块所有初始化数据
0x09	设置 Modbus-CAN 模块版本号
0x0A	启停 Modbus-CAN 模块 CAN 报文监视
0x0B	启停驱动器参数识别
0x0C	读取驱动器参数

4）数据域为本帧通信时包含的数据

5）校验和为模 256 和，加和从功能码（包含）开始到数据域（包含）为止

6）报文以字节为单位，字节流形式进行收发

报文功能码说明：

- **功能码 0x00**

设定 Modbus-CAN 模块串口通信参数

下行：

数据域内容	数据格式	字节数
串口波特率	BIN	4
校验位	BIN	1

说明：

① 串口波特率由 4 字节组成，**大端模式**，即高字节在前，低字节在后。如“115200”表示成十六进制的“00 01 C2 00”，发送时按照“00” “01” “C2” “00”的顺序发送

② 校验位由 1 字节组成，值与含义如下表所示

值	含义
0	无校验
1	奇校验
2	偶校验

③ Modbus-CAN 模块支持“115200”“57600”“38400”“19200”“9600”“4800”的波特率，若下发其他波特率值，模块会将波特率恢复为默认值“57600”

上行：数据为空。

如： 0xff 0x05 0x00 0x00 0x16

➢ **功能码 0x01**

链路检测（①登陆 ②心跳）

① 登陆帧：

下行：

数据域内容	数据格式	字节数
链路内容标识	BIN	1

说明：登陆帧下行数据域内容为“链路内容标识”，固定值 0x00，表示本帧为登陆帧

上行：

数据域内容	数据格式	字节数
链路内容标识	BIN	1
CAN 帧格式标识	BIN	1
CAN 发送方式标识	BIN	1
CAN 通信速率标识	BIN	1

说明：

① “链路内容标识”为固定一字节 0x00，表示本帧为登陆帧

② CAN 帧格式标识：0 标准帧；1 扩展帧

③ CAN 发送方式标识：0 自动重发；1 单次发送

④ CAN 通信速率标识

CAN 通信速率对应关系：

CAN 通信速率标识	对应通信速率
0	1000kbps
1	800 kbps
2	666 kbps
3	500 kbps
4	400 kbps
5	250 kbps
6	200 kbps
7	125 kbps

8	100 kbps
9	80 kbps
10	50 kbps
11	40 kbps
12	20 kbps
13	10 kbps
14	5 kbps

② 心跳帧：

下行：

数据域内容	数据格式	字节数
链路内容标识	BIN	1

说明：心跳帧下行数据域内容为“链路内容标识”，固定值 0x01，表示本帧为心跳帧

上行：

心跳报文上行帧的数据域为一字节 0x01，表示本帧为心跳报文

数据域内容	数据格式	字节数
标识码 0x01	BIN	1
接收错误计数值	BIN	1
发送错误计数值	BIN	1
总线状态	BIN	1
填充错误计数	BIN	1
格式错误计数	BIN	1

应答错误计数	BIN	1
位显性错误计数	BIN	1
位隐性错误计数	BIN	1
CRC 校验错误计数	BIN	1
模块发送超时/超次标志	BIN	1
单次发送模式连续发送时的失败次数（>2 则“模块发送超时/超次标志”置 1）	BIN	1
CAN 报文 1：帧 ID	BIN	1
CAN 报文 1：帧信息	BIN	1
CAN 报文 1：帧数据	BIN	1
CAN 报文 2：帧 ID	BIN	1
CAN 报文 2：帧信息	BIN	1
CAN 报文 2：帧数据	BIN	1
……	BIN	1
CAN 报文 n*：帧 ID	BIN	1
CAN 报文 n：帧信息	BIN	1
CAN 报文 n：帧数据	BIN	1

说明：

n 不大于 18：使能“启停 CAN 报文监视”时，若两帧心跳报文间隔时间内，CAN 总线上出现了多于 18 帧 CAN 报文的数据，则终端上报的心跳帧中将从第 19 帧 CAN 报文开始上报。（本设计意在减少 UART 单帧报文通信时间，从而保证通信实时性，建议用户使用

高 UART 波特率）

若用户使能“启停 CAN 报文监视”，则心跳报文数据域中包含“CAN 报文 X：XX”紫色区域的数据，若禁能“启停 CAN 报文监视”，则心跳报文数据内容不包含“CAN 报文 X：XX”的数据。

- **功能码 0x02**

查询终端版本号

下行：

下行报文数据体内容为空

上行：

数据域内容	数据格式	字节数
终端版本号	ASCII	N

说明：根据报文总长度确定 ASCII 数据的长度 N

- **功能码 0x03**

Modbus-CAN 模块 CAN 帧格式设定

下行：

数据域内容	数据格式	字节数
CAN 帧格式标识	BIN	1

下行报文数据域为一字节。数据内容及含义如下表所示

值	含义
0	标准帧

1	扩展帧

上行：

数据域内容	数据格式	字节数
CAN 帧格式标识	BIN	1

说明：上行报文中的“CAN 帧格式标识”为 Modbus-CAN 模块中已被正确设置的帧格式标识，值含义与下行相同

- **功能码 0x04**

Modbus-CAN 模块 CAN 发送方式设定

下行：

数据域内容	数据格式	字节数
CAN 发送方式标识	BIN	1

下行报文数据域为一字节。数据内容及含义如下表所示

值	含义
0	自动重发
1	单次发送

说明：

① 自动重发：CAN 报文由于任何原因没有成功发送，模块将对本报文不断重复发送

② 单次发送：模块对于当前报文只发送一次，不管是否成功

上行：

数据域内容	数据格式	字节数
CAN 发送方式标识	BIN	1

说明：上行报文中的“CAN 发送方式标识”为 Modbus-CAN 模块中已被正确设置的发送方式标识，值含义与下行相同

- **功能码 0x05**

Modbus-CAN 模块 CAN 通信速率设定

下行：

数据域内容	数据格式	字节数
CAN 通信速率标识	BIN	1

说明：CAN 通信速率标识见上文“登陆帧”中表格

上行：

数据域内容	数据格式	字节数
CAN 通信速率标识	BIN	1

说明：上行报文中的通信标识为 Modbus-CAN 模块当前已被正确设置的 CAN 通信速率标识

- **功能码 0x06**

Modbus-CAN 模块硬件滤波设置

下行：

数据域内容	数据格式	字节数
标准帧个数	BIN	1
扩展帧个数	BIN	1
标准帧 ID 1	BIN	4

标准帧 ID 2	BIN	4
……		
标准帧 ID m	BIN	4
扩展帧 ID 1	BIN	4
扩展帧 ID 2	BIN	4
……		
扩展帧 ID n	BIN	4

说明：

① 使用硬件滤波时，Modbus-CAN 模块将直接把不在滤波列表中的 CAN 报文滤除，不会被模块收到。下行报文中数据域内容即为滤波时允许通过报文的信息。

② 帧 ID 由 4 字节组成，**大端模式**

上行：

上行报文数据体内容为空

- **功能码 0x07**

Modbus-CAN 模块软件复位

功能码 0x07 用于对终端进行复位，复位终端后，终端程序会从头开始执行。

下行：

下行报文数据体内容为空

上行：

上行报文数据体内容为空

- **功能码 0x08**

清除 Modbus-CAN 模块所有初始化数据

下行：

下行报文数据体内容为空

上行：

上行报文数据体内容为空

- **功能码 0x09**

设置 Modbus-CAN 模块版本号

下行：

下行报文数据域为任意长度表示终端版本号的字符，表示方式为 ASCII 模式。

上行：

上行报文数据体内容为空

- **功能码 0x0A**

启停 Modbus-CAN 模块 CAN 报文监视

下行：

数据域内容	数据格式	字节数
启停标识	BIN	1

下行报文数据域为一字节。数据内容及含义如下表所示

值	含义
0	启动 CAN 报文监视
1	停止 CAN 报文监视

说明：当使能 CAN 报文监视时，Modbus-CAN 模块与上位机软件通信时的**心跳帧**中，会附加模块在 CAN 总线上监视到的 CAN 报文信息，信息格式见“功能码 0x01”中心跳帧的报文说明。

上行：

数据域内容	数据格式	字节数
启停标识	BIN	1

上行报文数据域为一字节。数据内容及含义如下表所示

值	含义
0	CAN 报文监视已启动
1	CAN 报文监视已停止

- **功能码 0x0B**

启停 Modbus-CAN 模块对驱动器的参数识别。

下行：

数据域内容	数据格式	字节数
启停标识	BIN	1

报文数据域为一字节。数据内容及含义如下表所示

值	含义
0	停止识别
1	启动识别

说明：识别的实现方式为，Modbus-CAN 模块使用不同的“**CAN 通信速率**”、“**帧格式**”、“**目标 ID**”与 CAN 设备进行握手，此过程若通信成功，则识别结束。Modbus-CAN 模块

保存 CAN 设备的“**CAN 通信速率**”、“**帧格式**”、“**目标 ID**”直至 Modbus-CAN 模块重启或断电。

上行：

数据域内容	数据格式	字节数
启停标识	BIN	1

上行报文数据域为一字节。数据内容及含义如下表所示

值	含义
0	识别已启动
1	识别已停止

- **功能码 0x0C**

读取驱动器参数

功能码 0x0C 用于向 Modbus-CAN 模块请求已经识别到的驱动器的参数，包括驱动器的“**CAN 通信速率**”、“**帧格式**”、“**驱动器 ID**”

下行：

下行报文数据体内容为空

上行：

数据域内容	数据格式	字节数
帧类型标识	BIN	1
驱动器 ID 号	BIN	1
CAN 通信速率标识	BIN	1

说明：CAN 通信速率对应关系见“功能码 0x01”中关于“CAN 通信速率标识”的介绍。

附 3：Modbus-CAN 模块与“CAN 驱动控制一体系统”通信协议（典型应用）

协议概要：

模块实现将 Modbus 报文转换为 CAN 报文，下发到 CAN 总线上，所使用的 CAN 报文类型为“**数据帧**”。由于 CAN 报文单帧最多只能包含 8 字节数据（Data），所以按照本协议规定，内容较少字节的 Modbus 报文，模块将转换为单帧 CAN 报文下发，对于内容较多的 Modbus 报文，模块将对 Modbus 报文进行拆分，将拆分后的 CAN 报文分帧后下发。协议规则如下：

1）帧 ID。

模块可使用“标准帧”或“扩展帧”进行通信。

在本协议中，针对“标准帧”或“扩展帧”，将 Modbus 报文中的“**从机地址**”的值进行转换，得到 CAN 报文的 ID 值。

Modbus 报文“从机地址”由 1 字节表示，0 表示广播地址。

对于标准帧:CAN 报文“标准帧”ID 为 11 位,地址转换时,直接将 Modbus 报文“从机地址”的值赋给 CAN“标准帧”ID。

对于扩展帧：CAN 报文“扩展帧”ID 为 29 位，其中高 11 位为标准 ID 部分，低 18 位为扩展 ID 部分。地址映射时，将 Modbus 报文“从机地址”的值赋给标准 ID（11 位）部分，将扩展 ID（18 位）部分全部填为 0。

2）帧序列域

① 将 CAN 报文 8 字节数据部分（data）的第 1 个字节作为 CAN 报文的帧序列使用。字节的 D7 和 D6 位为**帧标识**。D7D6 两位所表示值的含义如下：

值	含义
0	中间帧

1	末帧
2	首帧
3	单帧

当模块将 Modbus 报文转换为 CAN 报文时，对于较长的 Modbus 报文，模块将对其进行**分帧**，转换为几个 CAN 报文进行下发。

若 Modbus 报文转换后成为“一帧”CAN 报文，那么此 CAN 报文为“单帧”；

若 Modbus 报文转换后成为“多帧”CAN 报文 那么其中的第一帧 CAN 报文称为“**首帧**”，最后一帧 CAN 报文称为“**末帧**”，“首帧”和“末帧”之间的报文称为“**中间帧**”。

② CAN 报文 8 字节数据部分（data）的第一字节作为 CAN 报文的帧序列使用。字节的 D5 和 D4 位表示的值 D5D4 为固定值 0。

③ CAN 报文 8 字节数据部分（data）的第一字节作为 CAN 报文的帧序列使用。字节的 D3-D0 位表示模块对一帧 Modbus 报文转换后得到的 CAN 报文的序号。D3D0 值为 0-15 循环。

举例说明：若某 Modbus 报文经过模块转换后，成为 4 帧 CAN 报文，并下发到 CAN 总线上，那么这四帧 CAN 报文的 D3-D0 位表示的值依次为 0、1、2、3；若随后模块又对一帧 Modbus 报文进行了转换，转换后成为 18 帧 CAN 报文，并下发到总线上，那么这 18 帧 CAN 报文的数据区第一字节的 D3-D0 位表示的值依次为 0、1、2...15、0、1。

对于每一个新的 Modbus 报文进行转换后都会得到一个或多个数量不等的 CAN 报文，每一组 CAN 报文的 D3-D0 位表示的值都是从 0 开始的。

具体实例：（下列表示方法为 16 进制）

① 单帧：（CAN 使用**标准帧**发送）

Modbus 帧	01 05 00 00 FF 00 8C 3A
CAN 帧	Data：C0 05 00 00 FF 00 8C 3A　　ID：001

② 多帧：(CAN 使用**标准帧**发送)

Modbus 帧	01 10 00 00 00 08 10 00 B4 00 08 00 32 00 32 00 64 00 00 00 A0 00 01 EF 0F
CAN 帧	Data：80 10 00 00 00 08 10 00　　ID：001
	Data：01 B4 00 08 00 32 00 32　　ID：001
	Data：02 00 64 00 00 00 A0 00　　ID：001
	Data：43 01 EF 0F　　ID：001

③ 单帧：(CAN 使用**扩展帧**发送)

Modbus 帧	01 05 00 00 FF 00 8C 3A
CAN 帧	Data：C0 05 00 00 FF 00 8C 3A　　ID：00040000

④ 多帧：(CAN 使用**扩展帧**发送)

Modbus 帧	01 10 00 00 00 08 10 00 B4 00 08 00 32 00 32 00 64 00 00 00 A0 00 01 EF 0F
CAN 帧	Data：80 10 00 00 00 08 10 00　　ID：00040000
	Data：01 B4 00 08 00 32 00 32　　ID：00040000
	Data：02 00 64 00 00 00 A0 00　　ID：00040000
	Data：43 01 EF 0F　　ID：00040000

故障及排除：

1、连接不上 Modbus-CAN 模块

A、模块与电脑的连接是否牢靠

B、是否安装驱动

C、是否选择了正确的端口

2、向驱动器下发指令时回复失败或监视窗口不显示数据内容

A、是否打开了串口

B、上位机是否与 Modbus-CAN 模块建立了良好的连接

C、CAN 通信物理连接是否正确：如差分通信线线序，是否 H 接 H，L 接 L；

通信线是否有松动或断开

D、Modbus-CAN 模块与驱动器模块的“帧格式”“CAN 通信速率”是否一致；

下发指令时的“从机地址”是否是当前驱动器模块的 ID

3、重新配置驱动器模块的“ID”“帧格式”“CAN 通信速率”后为何通信失败

A、对驱动器模块进行这三项配置后需对驱动器进行复位

B、驱动器模块 CAN 参数被重新设置后 ,Modbus-CAN 模块的 CAN 通信参数以及“从机地址”号是否仍然与驱动器一致，若不一致，请重新选择 Modbus-CAN 模块的 CAN 参数

4、进行驱动器识别后，为何显示总线通信错误

A、通信时需要尝试不同的 CAN 通信速率，由于无应答，出现总线错误为正常现象

B、一般此类错误不影响正常通信，可忽略

C、可通过重启 Modbus-CAN 模块来消除错误

若错误消除后，收发时依然出错，请按以下步骤排除故障

5、为何会有总线错误

A、Modbus-CAN 模块与 CAN 总线的物理连线松散或断开

B、CAN 总线上有其他设备发生故障

C、CAN 总线上其他设备是否有物理连接线序不正确的现象