CAN-USB（Modbus-rtu）HUB 产品说明书

一、产品介绍

本模块主要完成 CAN 与 PC 端 USB（驱动装好后识别为 COM 串口）双向通信的功能，COM 接口通信协议采用了工业通用的 Modbus-rtu 协议格式和私有自定义协议（目的是配置本模块的基本参数，如：串口通信速率、CAN 通信帧格式等参数），CAN 接口通信协议为对应转换的规约；因此本模块又名为 CAN-USB（Modbus-rtu）集中器。内部电路及通信格式等设计有以下特点：

1、 CAN 接口采用工业设计隔离模块，做了全抗干扰处理，隔离电压达 1KV；

2、 USB 端采用了 FT230 工业芯片（完成 USB 转 COM 功能），需装驱动；

3、 COM、CAN 端通信速率及相关帧格式可设定，实现协议参考附件二；

4、 模块具体接线方式参考相应的接线说明图（参考附件三）；

5、 出厂默认参数配置，串口：波特率为 57600，8 位数据，1 位停止，无校验；CAN 口：500KBPS，单次触发，标准帧。

二、USB 端（COM 口）通信规约介绍

1、Modbus-rtu 报文格式

Modbus-rtu 报文，MODBUS-CAN 的实现报文。为方便广大客户朋友应用，本公司现将相应部分内容列举如下：

（1-1）MODBUS-RTU 功能码（最常用功能码）：

01 (0x01)　　读线圈
02 (0x02)　　读离散量输入
03 (0x03)　　读保持寄存器
04 (0x04)　　读输入寄存器
05 (0x05)　　写单个线圈
06 (0x06)　　写单个寄存器
15 (0x0F)　　写多个线圈
16 (0x10)　　写多个寄存器

上面“线圈”、“寄存器”其实分别指的就是“位变量”和“16 位变量”。

（1-2）MODBUS-RTU 的地址说明

数据类型	modbus 地址，寄存器编号	读功能码	写功能码	数据大小
数字量输出(线圈)	00001——09999	01H	05H，0FH	位，1bit
数字量输入(触点)	10001——19999	02H		位，1bit
输入寄存器	30001——39999	04H		字，16bit
保持寄存器	40001——49999	03H	06H，10H	字，16bit

modbus 地址使用 0xxxx；1xxxx；3xxxx；4xxxx 十进制形式。通信协议里的地址使用的是十六进制，需要进行如下转换：比如 modbus 地址 00001，对应协议地址为 0x00；00009，对应协议地址 0x08；同理，modbus 地址 40001；代表协议地址 0x00；40010 代表协议地址 0x09；40011 代表协议地址 0x0a。依次类推！

（1-3）具体协议的其他格式情况，查看本说明书的附件一。

2、私有协议（配置模块本身内部参数的协议，我公司自定义的协议）：这部分的内容详细请参考本说明书的附件二。

三、CAN 报文：即通过 CAN 通信接口发送出或接收到的一串数据。

一帧 CAN 报文最多包含 8 个字节。以下用 D0—D7 顺序表示这 8 个字节。而对于一帧较长的 modbus-rtu 报文（如大于 8 个字节数据）时，需要将它拆分为多帧 CAN 报文。

所以要对 CAN 报文进行分类；规则如下：

1、单帧报文：小于等于 8 个字节的一帧报文。

2、首帧报文，用于在多帧报文中区分先后顺序。

3、中间帧报文，用于在多帧报文中区分先后顺序。

4、末帧报文，用于在多帧报文中区分先后顺序。

帧序列：将一帧 CAN 报文的 8 个字节的数据的第一个字节（D0）来描述以上 4 种报文。该字节（B0-B7 共 8 位）的 B7(第 8 位，最高位)、 B6(第 7 位)来表示，如下表：

含义	B7	B6
中间帧	0	0
末帧	0	1
首帧	1	0
单帧	1	1

该字节的 B5(第 6 位) 、B4(第 5 位) 、B3(第 4 位)未使用，默认值都为 0。

该字节的 B2(第 3 位) 、B1(第 2 位) 、B0(第 1 位)来表示一帧较长的 modbus-rtu 报文(大于 8 个字节)拆分为多帧 CAN 报文的序号。（注意：通过对 MODBUS-RTU 协议报文分析，最多可能拆分为 4 帧）。

举例说明：比如一帧 modbus-rtu 报文(大于 8 个字节)需要拆分为 4 帧 CAN 报文，

那么第 1 帧 CAN 报文的第一个字节的 B3—B0 位的值为 0；

那么第 2 帧 CAN 报文的第一个字节的 B3—B0 位的值为 1；

那么第 3 帧 CAN 报文的第一个字节的 B3—B0 位的值为 2；

那么第 4 帧 CAN 报文的第一个字节的 B3—B0 位的值为 3；

四、帧 ID (即 CAN 报文 ID)

本转换协议中，CAN 报文 ID 值对应 Modbus-RTU 报文中的设备 ID 值（地址）。由一个字节表示。

1、对于 CAN 标准帧：CAN 报文 ID 为 11 位，所以可以直接将 Modbus 报文中的设备 ID 的值赋给 CAN 报文 ID。

2、对于 CAN 扩展帧：CAN 报文 ID 为 29 位，其中高 11 位标准 ID 部分，低 18 位为扩展 ID 部分；所以可以直接将 Modbus 报文中的设备 ID 的值赋给高 11 位标准 ID 部分；低 18 位的扩展 ID 部分全部填为 0。

五、分析几条 Modbus-rtu 报文转换为 CAN 报文的示例。

1、读寄存器、写线圈、写一个寄存器、写多个寄存器的反馈只需一帧 CAN 报文

2、读寄存器的反馈、写多个寄存器需要多帧报文。

3、比如：（读多个连续的寄存器），**建议一次读的个数不要超过 5 个。**

（1）举例：以读寄存器的值为例。modbus 报文与 CAN 报文对应发送如下：

Modbus	设 备	无	功 能	寄 存	寄 存	读 寄	读 寄	CRC	CRC

	号		码	器起始地址高8位	器起始地址低8位	存器个数高8位	存器个数低8位	低8位	高8位
CAN	帧 ID	D0	D1	D2	D3	D4	D5	D6	D7

PC 端通过 USB(COM)口发送给模块的 Modbus-rtu 发送报文: 01 03 00 00 00 02 c4 0B；

通过本模块后，从 CAN 接口端发出来的报文如下（标准帧）: 01 C0 03 00 00 00 02 C4 0B;

帧 ID	D0	D1	D2	D3	D4	D5	D6	D7
01	C0	03	00	00	00	02	C4	0B

对上面报文分析：PC 端通过 COM 口发送读取从地址 00 、00 开始的 2 个字节寄存器的数据，通过本模块后，CAN 通信接口输出用一帧数据就可以发出。同理，若 CAN 接口端收到 01 C0 03 00 00 00 02 C4 0B；则从 USB(COM)端发出 01 03 00 00 00 02 c4 0B。

特注：上面的 01 、03 等实际为 01H，03H（十六进制）；0X01，0X03（C 语言写法）。

modbus 报文与 CAN 报文对应接收格式如下：

Modbus	设备号	无	功能码	字节个数	寄存器 1 值高 8 位	寄存器 1 值低 8 位	寄存器 2 值高 8 位	寄存器 2 值低 8 位	CRC 低 8 位	CRC 高 8 位
CAN	帧 ID	D0	D1	D2	D3	D4	D5	D6	D7	D1（第二帧）

以收到 2 个寄存器的数据为例，若 CAN 接收报文如下: 大于 8 个字节，所以 2 帧数据。

帧 ID	D0	D1	D2	D3	D4	D5	D6	D7
01	80	03	04	00	B4	00	08	bb

帧 ID	D0	D1
01	41	D3

CAN 报文分成 2 帧收到数据，第一帧：01 80 03 04 00 B4 00 08 BB；第二帧为：01 41 D3。

模块内部转换后，Modbus 发出报文为: 01 03 04 00 b4 00 08 bb d3；

对上面报文分析：本模块 CAN 端收到第一帧：01 80 03 04 00 B4 00 08 BB；第二帧为：01 41 D3 后；通过 USB（COM 口）发给 PC 端 01 03 04 00 b4 00 08 bb d3。同理，若 USB 端收到数据，则通过本模块后从 CAN 端发出相应的数据帧。

（2）举例：发一条线圈设定命令。

modbus 报文与 CAN 报文对应发送如下：

Modbus	设备号	无	功能码	线圈起始地址高8位	线圈起始地址低8位	输出值高8位	输出值低8位	CRC 低8位	CRC 高8位
CAN	帧 ID	D0	D1	D2	D3	D4	D5	D6	D7

Modbus 发送报文: 01 05 00 07 ff 00 3d fb

CAN 发送报文如下: 一帧

帧 ID	D0	D1	D2	D3	D4	D5	D6	D7
01	C0	05	00	07	ff	00	3d	fb

对上面报文分析：若 USB 端收到报文：01 05 00 07 ff 00 3d fb；则从 CAN 端发出报文：01 C0 05 00 07 FF 00 3D FB；反之，若 CAN 端收到报文：01 C0 05 00 07 FF 00 3D FB，则从 USB 端发出：01 05 00 07 ff 00 3d fb。（**注意：十六进制数据不区分大小写！**）

modbus 报文与 CAN 报文对应接收如下：（和发送的数据一样）

Modbus	设备号	无	功能码	线圈起始地址高 8 位	线圈起始地址低 8 位	输出值高 8 位	输出值低 8 位	CRC 低 8 位	CRC 高 8 位
CAN	帧 ID	D0	D1	D2	D3	D4	D5	D6	D7

Modbus 接收报文: 01 05 00 07 ff 00 3d fb

CAN 接收报文如下：一帧，

帧 ID	D0	D1	D2	D3	D4	D5	D6	D7
01	c0	05	00	07	ff	00	3d	fb

（3）举例：写一个寄存器的值命令。

modbus 报文与 CAN 报文对应发送如下：

Modbus	设备号	无	功能码	寄存器起始地址高 8 位	寄存器起始地址低 8 位	写入值高 8 位	写入值低 8 位	CRC 低 8 位	CRC 高 8 位
CAN	帧 ID	D0	D1	D2	D3	D4	D5	D6	D7

Modbus 发送报文: 01 06 00 01 00 04 d9 c9

CAN 发送报文如下: 一帧

帧 ID	D0	D1	D2	D3	D4	D5	D6	D7
01	C0	06	00	01	00	04	D9	C9

对上面报文分析：若 USB 端收到报文：01 06 00 01 00 04 d9 c9；则从 CAN 端发出报文：01 C0 06 00 01 00 04 d9 c9；反之，若 CAN 端收到报文：01 C0 06 00 01 00 04 d9 c9，则从 USB 端发出：01 06 00 01 00 04 d9 c9。

modbus 报文与 CAN 报文对应接收如下：（和发送的数据一样）

Modbus	设备号	无	功能码	寄存器起始地址高 8 位	寄存器起始地址低 8 位	写入值高 8 位	写入值低 8 位	CRC 低 8 位	CRC 高 8 位
CAN	帧 ID	D0	D1	D2	D3	D4	D5	D6	D7

Modbus 接收报文: 01 06 00 01 00 04 d9 c9

CAN 接收报文如下：一帧

帧 ID	D0	D1	D2	D3	D4	D5	D6	D7
01	C0	06	00	01	00	04	D9	C9

（4）举例：写多个连续的寄存器的值命令。

modbus 报文与 CAN 报文对应发送如下：

Modbus	设备号	无	功能码	寄存器起始地址高 8 位	寄存器起始地址低 8 位	寄存器个数高 8 位	寄存器个数低 8 位	字节个数	写入值 1 高 8 位	写入值 1 低 8 位	写入值 1 高 8 位	写入值 1 低 8 位	CRC 低 8 位	CRC 高 8 位
CAN	帧 ID	D0	D1	D2	D3	D4	D5	D6	D7	D1	D2	D3	D4	D5

Modbus 发送报文: 01 10 00 09 00 02 04 00 c8 00 00 b2 3b

CAN 发送报文如下: 大于 8 个字节，所以需要做 2 帧发送。

帧 ID	D0	D1	D2	D3	D4	D5	D6	D7
01	80	10	00	09	00	02	04	00

帧 ID	D0	D1	D2	D3	D4	D5
01	41	C8	00	00	B2	3B

对上面报文分析：若 USB 端收到报文：01 10 00 09 00 02 04 00 c8 00 00 b2 3b；则从 CAN 端发出报文第一帧：01 80 10 00 09 00 02 04 00 ；第二帧：01 41 c8 00 00 b2 3b；反之，若 CAN 端收到报文第一帧：01 80 10 00 09 00 02 04 00 ；第二帧：01 41 c8 00 00 b2 3b，则从 USB 端发出：01 10 00 09 00 02 04 00 c8 00 00 b2 3b。

modbus 报文与 CAN 报文对应接收如下：

Modbus	设备号	无	功能码	寄存器起始地址高 8 位	寄存器起始地址低 8 位	寄存器的个数高 8 位	寄存器的个数低 8 位	CRC 低 8 位	CRC 高 8 位
CAN	帧 ID	D0	D1	D2	D3	D4	D5	D6	D7

Modbus 接收报文: 01 10 00 09 00 02 91 ca

CAN 接收报文如下：一帧

帧 ID	D0	D1	D2	D3	D4	D5	D6	D7
01	C0	10	00	09	00	02	91	Ca

六、注意事项

注意事项 1：修改本模块的串口通信速率、帧 ID、CAN 帧格式等采用了私有协议；

注意事项 2：MODBUS 通信时，读/写一个 32 位的参数(即占 2 个寄存器的参数)时，低

16 位在前（低寄存器地址内的数据：其中高 8 位在前，低 8 位在后），高 16 位在后（高寄存器地址内的数据：其中高 8 位在前，低 8 位在后）；

注意事项 3：本模块的应用主要针对工业总线 MODBUS-CAN 的开发技术人员使用。

七、测试方法

1、本 CAN-USB 模块的 USB 端接线插入 PC 的 USB 口后，需要安装对应 FT232 芯片的驱动程序，驱动程序包我公司会提供。安装好驱动后，计算机会识别到相应的 COM 号，在计算机的设备管理器里面会识别到，如下图所示：

2、测试软件

（1）采用串口调试精灵进行测试：我公司可以提供串口调试精灵软件下载，运行界面如下；由于 CAN-USB 模块出厂配置的串口通信格式为：波特率=57600，无奇偶校验位、8 位数据格式，1 位停止格式，所以上面串口调试精灵打开请按照此参数格式配置。

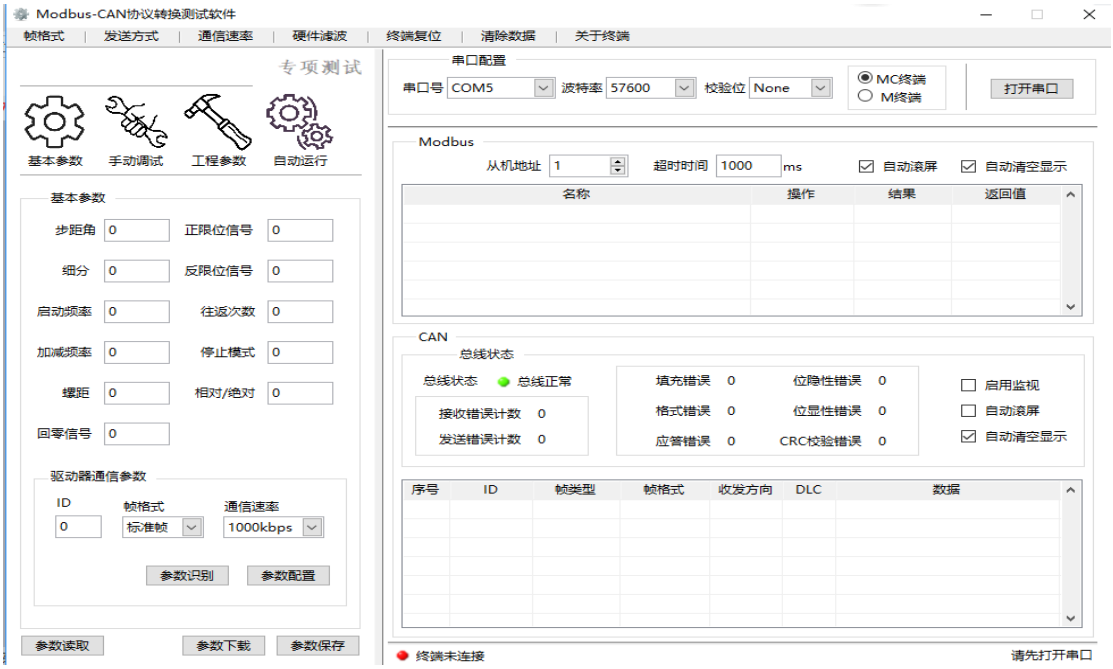
接下来，可以将 MODBUS 的各种命令帧通过上述串口调试软件发送给本模块，本模块通过 USB 端收到数据帧后，CAN 端将发出对应 CAN 数据帧。反之，若模块 CAN 接口接收相应数据帧（按照我方协议构建的 CAN 数据格式）后，则通过 USB 端返回在上面软件显示。

（2）Labview 应用软件：我公司针对 CAN 集中器模块产品，自行开发了 Labview 的串口调试应用软件，并提供源代码；此软件代码可以从我公司网站下载得到。使用我方此测试软件方法：

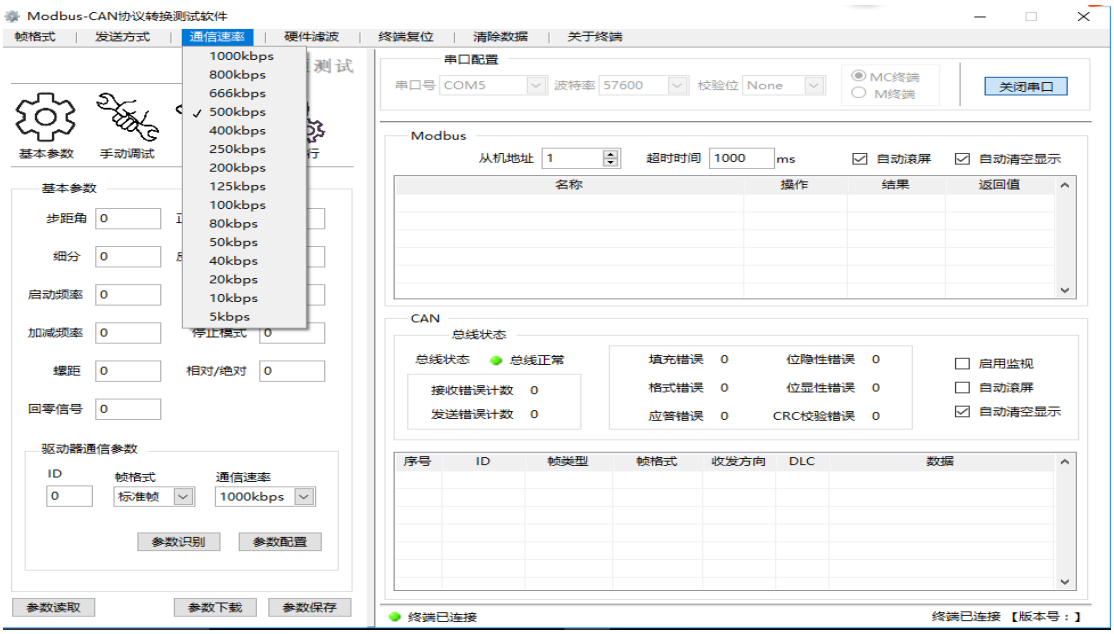
（2.1）针对熟悉 Labview 软件开发人员，电脑中已经安装好 Labview 2015 及以上版本的软件开发环境，只要将我方提供源代码和执行文件，拷贝到电脑后直接打开运行就可。

（2.2）针对不熟悉 Labview 的开发人员，可以用前面介绍的串口调试精灵测试，也可以将我方提供的 Labview 应用安装文件，安装到电脑中后运行使用，安装过程稍比较复杂。

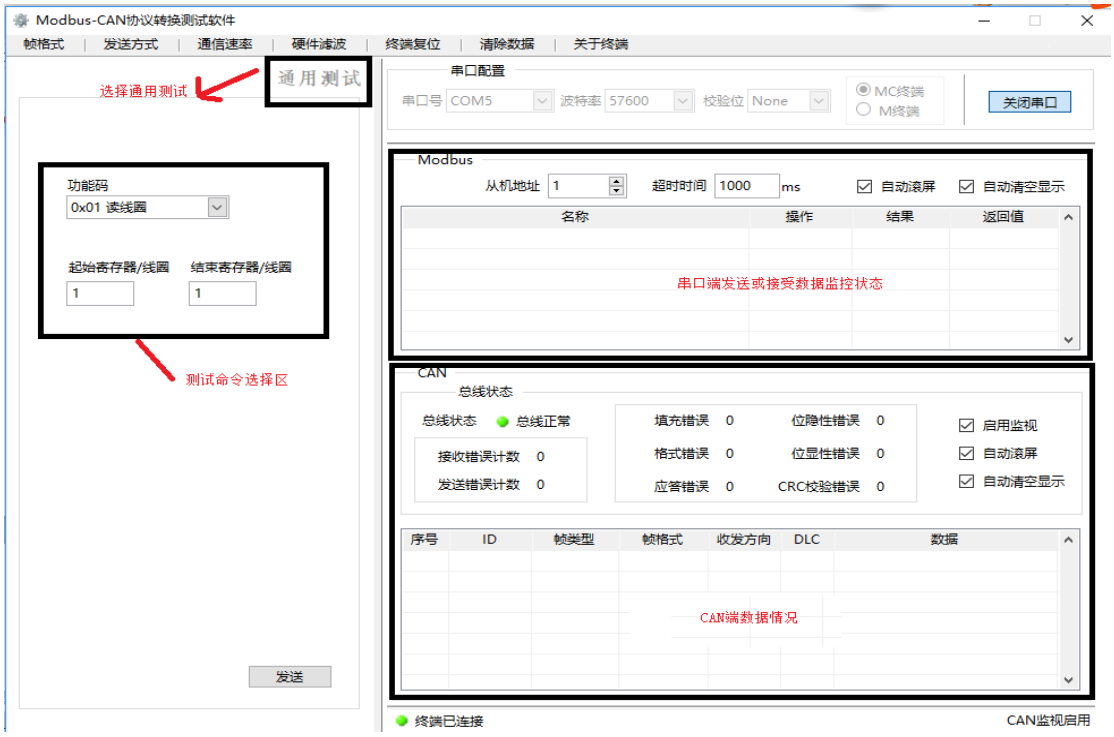
软件运行后界面如下：



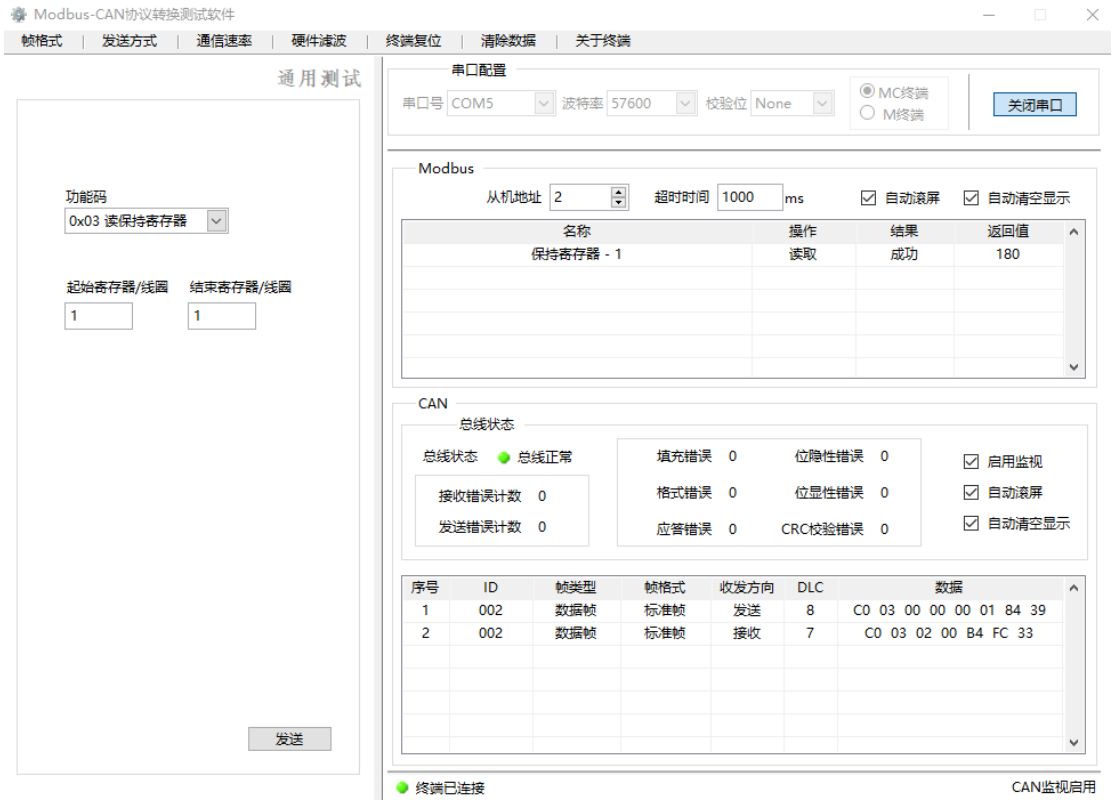
点击右上角的“打开串口”，默认模块的通信格式为：波特率=57600，无奇偶校验位、8 位数据格式，1 位停止位；同时默认为 CAN 帧格式为：标准帧、通信速率为 500KPBS。



接下来，可以在软件上进行 MODBUS 各种命令模式的测试。

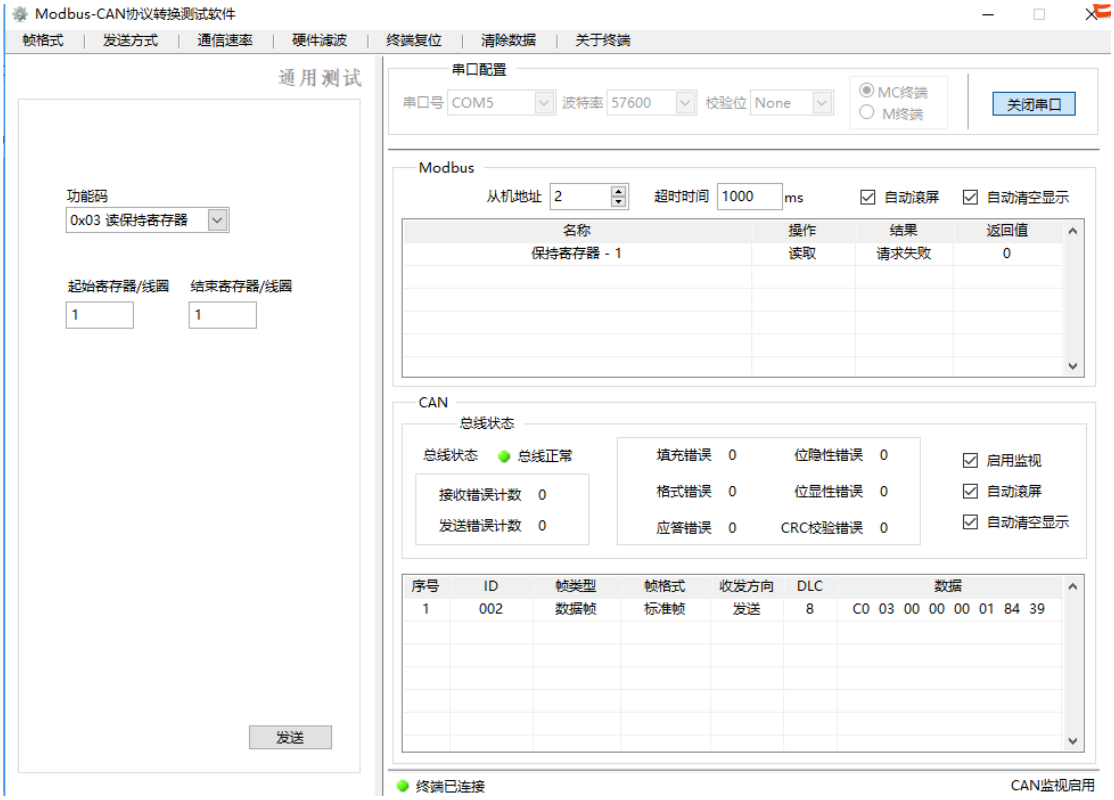


选择好对应的从机地址，若从机设备的地址为 2，则更改为 2.测试效果图如下：



上述测试分析：发送功能码 03 对应读取从机地址为 2 的设备（此设备已经通过 CAN 总线连接到总线上，并且已经正常运行），起始寄存器为 1 的 1 个单元的数据；返回对应寄存器的数据为 180.在 CAN 端口上监测到了发出去的数据帧和接收到的数据帧。

若 CAN 总线没有连接地址为 2 号的设备，则发送下去响应的结果如下图：



分析上述结果：发现请求读取 2 号设备寄存器 1 的数据失败，监测到 CAN 总线上已经通过 CAN 口发出了标准帧数据，只是 CAN 总线没有 2 号设备的数据反馈。

具体本软件的操作说明可以参考我公司相应的软件操作说明书。

附一：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 个连续状态。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		

附件二：上位机与 Modbus-CAN 模块间通信协议（私有协议）

上位机通过 COM(或 232、485 口)对 CAN-Modbus 模块本身进行内部参数设置或操作时，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：串口波特率和校验模式；

下行：

数据域内容	数据格式	字节数
串口波特率	BIN	4
校验模式	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”。

- **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 通信速率对应关系如下表：

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：（初学者建议先不用这个模式）

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 CAN 总线上其他设备的参数识别(对外接 CAN 总线上的设备进行识别，过程比较久，设备本模块+ 1 台 CAN 设备)

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**

  功能：读取 CAN 总线上其他设备的参数（适合本模块+1 个其他 CAN 设备）

功能码 0x0C 用于向 Modbus-CAN 模块请求已经识别到的其他设备的参数，包括相应 CAN 设备的“**CAN 通信速率**”、“**帧格式**”、“**驱动器 ID**”。

下行：

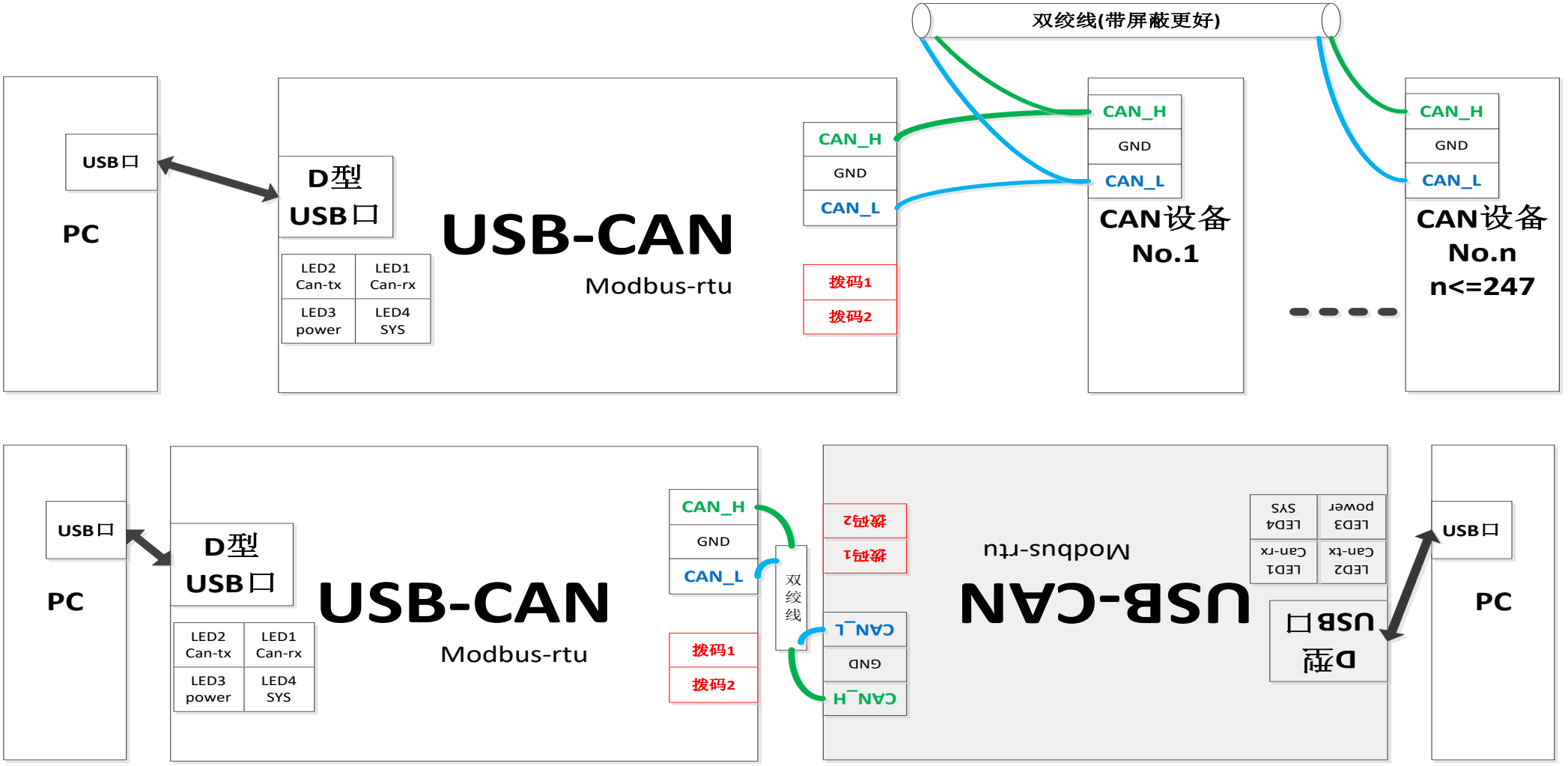
下行报文数据体内容为空

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。

附件三：产品接线图



说明：

1、USB-CAN集线器的4个led灯，can-rx表示CAN总线接收数据时闪，can-tx表示发送数据灯闪（一般在CAN发送数据时can-rx灯也会闪），power灯为电源指示灯，SYS灯为系统故障灯（公司产品出厂调试用）。

2、拨码1和拨码2：必须同时拨下或不拨，在构建CAN网络时，首、尾对应的两个CAN模块所对应的拨码开关要拨下；网络中间链路的CAN模块所对应的拨码开关不用拨。（系统内部对应为 120欧姆的线路匹配电阻）