CAN-109QC 驱动控制一体系统通信协议说明书

一、通信协议概述

本模块通信接口采用工业 CAN 通信接口，出厂的 CAN 帧格式设定为：500KBPS，单次触发，标准帧，数据帧模式；本说明书主要介绍 CAN 通信规约的具体内容和实现方式。

由于目前工控设备主流采用了串口通信模式（RS232、RS485：基于此两种接口，我公司另有相应的驱控产品），串口通信一般采用了 MODBUS 规约；所以，考虑产品的一致性和符合工业规范，公司这款基于 CAN 接口的驱动模块，内部通信规约实际方案是：软件实现了 CAN 规约转 MODBUS-RTU 的模式。因此，若对 MODBUS-RTU 熟悉的用户，阅读本模块的 CAN 通信协议的话，就比较轻松；若不熟悉 MODBUS-RTU，请耐心了解本通信协议的具体内容，同时熟悉一下相应的规约。

同时，本公司还提供了 CAN-USB（COM）、CAN-232、CAN-485 的集中器产品，用本 CAN-109Q 驱控模块直接通过 CAN 接口与这三款集中器相连的话，那么你就可以在 PC 端，PLC 端或工业触摸屏（文本屏）通过串口来操作此款 CAN 驱控模块。因此，需要应用我们此款 CAN-109Q 驱控模块的用户，有下面两种应用开发方式：

1、自行开发基于 CAN 总线上位机的客户，即客户自己开发的上位机端就是 CAN 接口，则客户需要详细了解我们 CAN 规约。

2、采用 PC 机、PLC、工业屏或其他终端的客户，希望采用串口（MODBUS-RTU 协议）开发上位机系统，那么客户只需要另外购置我们公司的 CAN-USB（COM）、CAN-232、CAN-485 三款集中器的一款，构建 N 个 CAN-109Q 驱控模块 + CAN 集中器 + PC（或 PLC 或工业屏）方案。

注意：本模块 CAN 接口采用工业设计隔离模块，做了抗干扰处理，隔离电压达 1KV。

二、CAN-109Q 驱控模块内部寄存器参数地址分配及说明

注意：1 个寄存器是 16 位的无符号数，占 2 个字节。

1、数据保持寄存器

PLC、或工业屏内部对应寄存器编号（modbus-rtu）	定义(对应参数)	读/写	本 CAN-109QC 内对应 16 位数据寄存器地址
40001	对应步进电机的步距角 (比如步距角是 1.8，写的时候需要扩大 100 倍，即设置为 180；读出来也是 180，除以 100 就对应 1.8)	R/W	0x0000
40002	对应电机驱动器的细分(驱动器上是多少细分，就设为多少细分)	R/W	0X0001
40003	对应电机运行的启动频率(单位：HZ)，步进电机运行一般有启动、加速、平速、减速、停止的过程。	R/W	0x0002
40004	对应电机运行的加减频率(单位：HZ)	R/W	0x0003
40005 40006	螺距 (电机转一圈对应的距离)：没有小数，若电机一圈运行的是 5.5MM，那么你就设定 55；后面运行距离同比 X 10 设定；如想运行一圈，运行距离就设定 55。	R/W	0x04、0x05（以下都缩写成一个字节）

40007	机械零点输入信号(有效值 0--5)：本模块有 5 个输入信号口（24V 负极有效），电机运行若需要回机械原点，则设备上接一个机械触点开关，或 NPN 常开的接近开关做原点反馈信号，信号反馈接到本模块 IN1-IN5 的任意一个，对应这里就设置 1-5 的值；设置 0 表示无机械原点。	R/W	0x06
40008	停止模式(0 缓慢停；1 立即停)：若设置为 0 表示减速停止，1 表示电机在收到停止命令时立刻停止。	R/W	0x07
40009	测试速度(单位：RPM 转/每分钟)：设定电机测试调试运行时的速度。	R/W	0x08
40010 40011	**测试距离(即单次运行的距离)**：设定电机测试调试运行时的距离，具体设定方式参考前面螺距设定说明。	R/W	0x09、0x0a
40012	测试方向 (0 正向，1 反向)：电机测试调试时运行方向设置	R/W	0x0b
40013	**单次运行到位反馈**：0 表示没到位 ，1 表示运行到位。通过读取这个寄存器来获取单次运行是否到位。**（只针对单次运行命令）**	R	0x0c
40014	备用		
40015	正限位信号(有效值 0--5):参考电机机械零点输入信号设置方法。	R/W	0x0e
40016	反限位信号(有效值 0--5):参考电机机械零点输入信号设置方法。	R/W	0x0f
40017	备用	R/W	0x10
40018	工程启动信号(有效值 0--5)：可以外接一个开关按钮来启动本模块设置的工程参数运行。对应输入 IN1-IN5，哪个输入接了开关按钮，就对应设定对应的数值，若设定为 0，则表示无需外接开关启动工程。	R/W	0x11
40019	工程停止信号(有效值 0--5)：可以外接一个开关按钮来停止本模块设置的工程参数运行。对应输入 IN1-IN5，哪个输入接了开关按钮，就对应设定对应的数值，若设定为 0，则表示无需外接开关来停止工程运行。	R/W	0x12
40020	备用		0x13
40021	系统工作次数计数：本寄存器存放的是系统工程运行的次数。	R/W	0x14
40022	备用		0x15

40023 40024	当前坐标存放寄存器，可以通过读取这两个寄存器的值来显示当前电机的运行坐标。	R	0x16、0x17
40025	工程号（只能为 1）：本模块只能存储一个工程数据，此工程最多能够设定 33 步。	R/W	0x18
40026	工程总步数(有效值 1--33)：工程的步数最多可设定 33.	R/W	0x19
40027	本步启动口信号(有效值 0--5)：请参考上面限位输入信号的设定方法。	R/W	0x1A
40028	工程本步电机运行启动频率，单位 HZ	R/W	0x1B
40029	工程本步电机运行加减频率，单位 HZ	R/W	0x1C
40030	工程本步电机运行方向(0 正向，1 反向)	R/W	0x1D
40031	工程本步运行速度 (单位：转/每分钟)	R/W	0x1E
40032 40033	工程本步运行距离：请参考前面测试调试电机运行距离设定的方法。	R/W	0x1F、0x20
40034	本步输出口(有效值 0--8)：对应有 3 个输出口，0 表示无设置；1 表示 OC1 开，2 表示 OC1 关；3 表示 OC2 开，4 表示 OC2 关；5 表示 OC3 开，6 表示 OC3 关。	R/W	0x21
40035 40036	本步运行完后延时时间(单位：毫秒)	R/W	0x22、0x23
40037	段循环起始步	R/W	0x24
40038	段循环结束步	R/W	0x25
40039	段循环次数	R/W	0x26
40040	设定工程当前步号(有效值 1--33)	R/W	0x27
40041	备用		0x28
40042	工程实时步号显示：（有效值 0—33），工程在运行时，通过查看此数据寄存器了解运行到第几步。	R	0x29
40043	第 1 路输入信号状态显示数据寄存器(1--ON；0--OFF)	R	0x2A
40044	第 2 路输入信号状态显示数据寄存器(1--ON； 0--OFF)	R	0x2B
40045	第 3 路输入信号状态显示数据寄存器(1--ON； 0--OFF)	R	0x2C
40046	第 4 路输入信号状态显示数据寄存器(1--ON； 0--OFF)	R	0x2D
40047	第 5 路输入信号状态显示数据寄存器(1--ON； 0--OFF)	R	0x2E
40048	第 1 路输出信号状态显示数据寄存器(1--ON； 0--OFF)	R	0x2F

40049	电机运行状态显示(1--ON 0--OFF)：1 表示电机正在运行，0 表示电机处于停滞状态。	R	0x30
40050	工程循环次数:设定工程运行循环的次数，若设定为 0，表示无限循环；若设定为 1 表示就运行一次。	R/W	0x31
40051	本步停止口信号(有效值 0--5)：请参考上面限位输入信号的设定方法。	R/W	0x32
40052	CAN 通信速率值，范围为 0-14 可设定和读取，0 对应 1000K；1 对应 800K；2 对应 666K；3 对应 500K；4 对应 400K；5 对应 250K；6 对应 200K；7 对应 125K；8 对应 100K；9 对应 80K；10 对应 50K；其他的设置速率更小。	R/W	0x33
40053	CAN 帧 ID，默认为 1。ID 识别号，驱控模块依靠自身 ID 进行硬件滤波。	R/W	0x34
40054	CAN 帧类型（0 数据帧 ；1 遥控帧）	R/W	0x35
40055	CAN 帧格式（0 标准格式 ；1 扩展格式）	R/W	0x36
40056	第 2 路输出信号状态显示(1--ON 0--OFF)	R	0x37
40057	第 3 路输出信号状态显示(1--ON 0--OFF)	R	0x38
40058	往返运行的次数	R/W	0x39
40076	相对/绝对运行(0 相对；1 绝对)，**只针对单次运行命令。**	R/W	0x3A
40077	运行方式选择（0 位置 ；1 速度触发；2 速度点动）：**主要是针对正反启动信号控制的。**	R/W	0x3B
40078	正转启动信号 (0 表示无设置) (1—5 对应 IN1—IN5 输入)**信号有效时，按上面选择的运行方式正转**	R/W	0x3C
40079	反转启动信号 (0 表示无设置) (1—5 对应 IN1—IN5 输入)**信号有效时，按上面选择的运行方式反转**	R/W	0x3D

备注：在修改了 CAN 帧格式的相关数据，如 ID，通信速率，帧类型和帧格式时，请上位机发数据保存命令给驱控模块，然后再断电重启模块，这些修改参数才生效。

2、线圈功能寄存器 (一般用来控制操作)

线圈寄存器对应驱控板相应功能为 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。

PLC、或工业屏内部对应线圈寄存器编号（modbus-rtu）	定义(对应参数)	本 CAN-109QC 内对应 16 位线圈寄存器地址

00001	数据保存：将本线圈寄存器设定为 0xff00,那么 CAN 驱控设备内部将完成数据存储的动作，**有些参数需要发保存命令后重新启动模块才有效。**	0x0000
00002	工程参数读取：将 CAN 驱控模块内部的工程参数从内部存储器读取到上面表格介绍的各工程类型数据保持寄存器中。	0x0001
00003	工程参数清零：将工程类型数据保持寄存器的值清 0，并不清上面表格的基本参数。更改后需要发保存命令，这样才真正将驱控模块内部存储器的工程参数清掉。	0x0002
00004	(工程)停止/急停：工程参数在运行时，一旦收到此命令，将立即停止运行。	0x0003
00005	正转点动：置 ON，电机一直正转 置 OFF，电机停止。	0x0004
00006	反转点动：置 ON，电机一直反转 置 OFF，电机停止。	0x0005
00007	回数据零：电机运行到坐标零点	0x0006
00008	**单次运行(按 40009 的速度和 40010 40011 的距离运行 1 次)：可选择相对/绝对运行 2 种方式运行。**	0x0007
00009	工程启动：按设定好的每一个步骤运行。直到所有步骤完毕或急停。	0x0008
00010	回机械零：电机一直反转，碰到机械零点信号停止。	0x0009
00011	坐标清零：将数据保持寄存器 40023 40024 的值设为 0，对应电机坐标。	0x000A
00012	输出 1 开：控制 OC1 输出低电平	0x000B
00013	输出 1 关：控制 OC1 输出高电平	0x000C
00014	输出 2 开：控制 OC2 输出低电平	0x000D
00015	输出 2 关：控制 OC2 输出高电平	0x000E
00016	输出 3 开：控制 OC3 输出低电平	0x000F
00017	输出 3 开：控制 OC3 输出高电平	0x0010
00018	工程上一步：将数据保持寄存器 40040 内的值减 1。	0x0011
00019	工程下一步：将数据保持寄存器 40040 内的值加 1。	0x0012

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（B7-B0 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、分析几条收到的 CAN 报文对应内部 Modbus-rtu 报文

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。**

（1-1）MODBUS-RTU 功能码（本 109QC 使用到的功能码有）：
03 (0x03)：读保持寄存器；05 (0x05)：写单个线圈 ；
06 (0x06)：写单个寄存器；16 (0x10)：写多个寄存器。

（1-2）MODBUS-RTU 的地址说明

数据类型	modbus 地址，PLC 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		05H	位，1bit
保持寄存器	40001——49999	03H	06H，10H	字，16bit

（1）举例：以读寄存器的值为例（功能码为 03）

1.1　以 CAN 接口收到单帧，modbus 报文与 CAN 报文对应发送如下：

Modbus 帧格式	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	读寄存器个数高 8 位	读寄存器个数低 8 位	CRC 低 8 位	CRC 高 8 位
CAN 帧	帧 ID	D0	D1	D2	D3	D4	D5	D6	D7

说明：比如本 109QC 模块从 CAN 接口收到帧：　01(地址 ID) + 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：01 表示地址 ID；C0 对应二进制（1100 0000），表示单帧；03 表示功能码，读数据寄存器；00 00 表示要读的数据寄存器首地址；00 02 表示要读取 2 个寄存器；C4　0B 两个字节对应的是 01（地址）、03（功能码）、00、00（这两个字节表示首地址）、00、02（这两个字节表示寄存器个数）共 6 个字节的 16 位 CRC 校验码，C4 表示低 8 位校验，0B 表示高 8 位校验。

CAN 接口的数据：01 C0 03 00 00 00 02 C4 0B（我们称为 CAN 帧）；实际我们驱动控制模块内部转换为 01 03 00 00 00 02 c4 0B（这就是 MODBUS-RTU 帧）。

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

1.2 以发出 2 个寄存器的数据为例，若 109QC 模块的 CAN 发出报文如下:

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 接口发出 2 帧数据，第一帧：01 80 03 04 00 B4 00 08 BB；第二帧为：01 41 D3。

第一帧分析：01 表示地址；80 对应二进制（1000 0000）表示 CAN 帧为多帧，并且这是首帧；03 表示功能码-读寄存器数据；04 表示发出的数据为 4 个字节，因为 2 个寄存器，每个寄存器 2 个字节；00 B4 表示第一个寄存器内的数据；00 08 表示的是第二个寄存器内的数据；bb 表示 16 位 CRC 校验的其中低 8 位。

第二帧数据分析：01 表示地址；41 对应二进制（0100 0001）表示 CAN 帧的末帧，第二帧；D3 表示的 16 位 CRC 校验的高 8 位。

模块内实际对应的 Modbus 报文为:　01 03 04 00 b4 00 08 bb d3。

Modbus 帧数据格式	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	帧 ID	D0	D1	D2	D3	D4	D5	D6	D7	D1

（2）举例：一条线圈设定命令（05 设定线圈命令）

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

CAN 接口收到的报文如下： 01 C0 05 00 07 FF 00 3D FB.

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：01 表示地址；C0 对应二进制（1100 0000），表示单帧；05 表示命令码，写单个线圈功能；00 07 表示线圈寄存器地址；FF 00 表示设定线圈为 ON；3D FB 表示 16 位 CRC 校验位。

CAN 接口收到报文后实际内部转换成的 MODBUS 报文为：01 05 00 07 ff 00 3d fb。

（**注意：十六进制数据不区分大小写！**）

（3）举例：写一个寄存器的值命令（06 设定单数据寄存器命令）

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

CAN 接口收到的报文数据如下: 01 C0 06 00 01 00 04 d9 c9

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：01 表示地址；C0 对应二进制（1100 0000），表示单帧；06 表示命令码，写单个数据寄存器；00 01 对应的是数据寄存器的地址；00 04 是写入的值；d9 c9 为 CRC 校验码。

对应内容由 CAN 转换成的 MODBUS 帧为：01 06 00 01 00 04 d9 c9。

（4）举例：写多个连续的寄存器的值命令（16 设定多个寄存器命令）

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

CAN 接口收到 2 帧报文如下， 第一帧：01 80 10 00 09 00 02 04 00；第二帧：01 41 c8 00 00 b2 3b。

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，第一帧：01 80 10 00 09 00 02 04 00；01 表示地址；80 对应二进制（1000 0000）表示 CAN 帧为多帧，并且这是首帧；10 表示功能码，写多个数据寄存器；00 09 表示寄存器的起始地址；00 02 表示寄存器的个数；04 表示数据的字节个数；

第二帧：01 41 c8 00 00 b2 3b；01 表示地址；41 对应二进制（0100 0001）表示末帧，第二帧；第一帧的最末位字节 00，与本帧的 C8，对应 00 c8 表示写入第一个寄存器（地址为 00 09）的数据值；00 00 表示写入第二个寄存器（地址为 00 0A）的数据值。B2 3B 表示 16 位 CRC 校验码。

驱动控制模块内容转换成的 MODBUS 帧为：01 10 00 09 00 02 04 00 c8 00 00 b2 3b。

备注：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，高 16 位在后。 比如测试距离和工程运行距离、螺距。

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