[返回主页](https://www.cnblogs.com/yzl050819/)

[**MODBUS TCP/IP协议规范详细介绍**](https://www.cnblogs.com/yzl050819/p/7591428.html)

1．该规范的发展概况

    原始版本1997年9月3日作为公共评论的草案。

    再版1999年3月29日，即修订版1.0。

    没有大的技术改动，仅作了补充说明。增加了附录A和B作为对一些常用执行问题的回应。

该Modbus/TCP规范在万维网上公开发行。它表明开发者的意愿是把它作为工业自动化领域具有互用性的标准。

既然MODBUS和MODBUS/TCP作为事实上的“实际”标准，而且很多生产商已经实现了它的功能，此规范主要是阐述在互连网上具有普遍可用性的基于TCP通讯协议的MODBUS报文的特殊编码。

2.  概    述

      MODBUS/TCP是简单的、中立厂商的用于管理和控制自动化设备的MODBUS系列通讯协议的派生产品。显而易见，它覆盖了使用TCP/IP协议的 “Intranet”和“Internet”环境中MODBUS报文的用途。协议的最通用用途是为诸如PLC，I/O模块，以及连接其它简单域总线或 I/O模块的网关服务的。

    MODBUS/TCP协议是作为一种（实际的）自动化标准发行的。既然MODBUS已经广为人知，该规范只将别处没有收录的少量信息列入其中。然而，本规 范力图阐明MODBUS中哪种功能对于普通自动化设备的互用性有价值，哪些部分是MODBUS作为可编程的协议交替用于PLC’s的“多余部分”。

    它通过将配套报文类型“一致性等级”，区别那些普遍适用的和可选的，特别是那些适用于特殊设备如PLC’s的报文。

2.1 面向连接

    在MODBUS中，数据处理传统上是无国界的，使它们对由噪音引起的中断有高的抵抗力，而且在任一端只需要最小的维护信息。

    编程操作，另一方面，期望一种面向连接的方法。这种方法对于简单变量通过唯一的“登录”符号完成，对于Modbus Plus变量，通过明确的“程序路径”容量来完成，而“程序路径”容量维持了一种双向连接直到被彻底击穿。

    MODBUS/TCP处理两种情况。连接在网络协议层很容易被辨认，单一的连接可以支持多个独立的事务。此外，TCP允许很大数量的并发连接，因而很多情况下，在请求时重新连接或复用一条长的连接是发起者的选择。

    熟悉MODBUS的开发者会感到惊讶：为什么面向连接TCP协议比面向数据报的UDP要应用广泛。主要原因是通过封装独立的“事务”在一个连接中，此连接 可被识别，管理和取消而无须请求客户和服务器采用特别的动作。这就使进程具有对网络性能变化的适应能力，而且容许安全特色如防火墙和代理可以方便的添加。

    类似的推理被最初的万维网的开发者所采用，他们选用TCP及端口80去实现一个作为单一事务的最小的环球网询问。

2.2 数据编码

    MODBUS 采用“big-endian”来表示地址和数据对象。

    这就意味着当一个数字表示的数量大于所传输的单一字节，最大有效字节将首先被发送。例如：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 16 | bits | 0x1234 | 将为 | 0x12 | 0x34 |
| 32 | bits | 0x12345678L | 将为 | 0x12 | 0x34 |
|  |  |  |  | 0x56 | 0x78 |

2.3参考编号的解释

    MODBUS将其数据模型建立在一系列具有不同特征的表的基础之上。这四个基本表如下

l         离散输入        单比特，由I/O系统提供，只读

l         离散输出        单比特，由应用程序更改，读写

l         输入寄存器      16比特，数值，由I/O系统提供  ，只读

l         输出寄存器      16比特，数值，由应用程序更改，读写

    输入和输出之间以及可寻址位和可寻址代码的数据对象之间的差别并不意味着任何应用性能的不同。如果这是我们所讨论的目标机械的最自然的解释，那么认为所有的四个基本表是相互覆盖的看法也是非常普通而完全可以接受的。

    对于每一个基本表，协议允许单独选择65536个数据对象中的任何一个，而且对那些对象的读写操作可以跨越多个连续的数据对象，直到达到基于处理事务功能代码的数据大小限制。

    这儿没有假定数据对象代表一种真正邻接的数据阵列，而这是大多数简单PLC’s的解释。

    “读写常用参考”功能代码被定义为携带32位的参考值并且能允许在“非常”大的空间里可以直接访问数据对象。现在没有可以利用这一特点的PLC设备。

    一个易造成混乱的潜在来源是用于MODBUS功能的参考值和用于Modicon                                PLC’s的“寄存器值”之间的关系。由于历史原因，用户参考值使用从1开始的十进制数表示。而MODBUS采用更普通的从0开始的无符号整数进行软件数 据整理分析。

    于是，请求从0读取寄存器的Modbus消息将已知值返回建立在寄存器4：00001（存储类型4=输出寄存器，参考值00001）中的应用程序。

2.4隐含长度基本原则

    所有的MODBUS 请求和响应都被设计成在此种方法下工作，即接收者可确认消息的完整性。对于请求和响应为固定长度的功能代码，仅发送功能代码就足够了。对于在请求和响应中 携带不定长数据的功能代码，数据部分前将加上一个字节的数据统计。

    当 Modbus通过TCP运送，前缀中携带附加的长度信息以便接收者识别消息的边界，甚至消息被分成若干组进行传输。外在的和隐含的长度准则的存在，以及 CRC-32检错代码（以太网）的使用使请求和响应消息中发生未被识别的错误的机率减至无限小。

3. 一致性等级概述

    当从草稿开始定义一种新的协议，有可能加强编码方式和阐述的一致性。MODBUS由于其先进的特性，已经在很多地方得到了实施，必须避免破坏它已经存在的实施。

    因此，已经存在的成套的处理类型被划分出一致性等级：等级0代表普遍使用且总体上一致的功能；等级2代表有用的功能，但带有某些特性。现存装置的不适应于互用性的功能也已确认。

    必须注意到，将来对该标准的扩充将定义附加的功能代码来处理现存事实标准不适用的情形。然而，被提议扩充的详细资料出现在本手册中将会另人误解。通过将代 码“随机的”发送或者即便是通过检查异常响应的类型来确定特别的目标装置是否支持特别的功能代码总是可能的，而且该方法将保证引入这些扩充的现使用的 MODBUS设备的连续的互用性。事实上，这就是当前功能代码的分级原则。

3.1等级0

     这是最小的有用功能，对主站和从站来说。

         读乘法寄存器 (fc 3)

         写乘法寄存器 (fc 16)

3.2等级 1

    这是附加的被普遍实现的和能共同使用的成套功能，正如前面介绍过的，许多从站把输入，输出，离散值和寄存器值作为同等的进行处理。

         读线圈 (fc 1)

         读离散输入 (fc 2)

         读寄存器输入 (fc 4)

         写线圈 (fc 5)

         写单一寄存器 (fc 6)

         读异常状态字 (fc 7)

    此功能对于每一个从站系列显然具有不同的含义。

3.3等级 2

    这些是需要HMI和管理等例行操作的数据传送功能。

         强制型多路线圈 (fc 15)

         读一般参考值 (fc 20)

    该功能可以处理并发的多个请求，而且能接收32位的参考数值。当前的584和984PLC’s仅使用此功能接收类型6的参考值（扩展的寄存器文件）。

    该功能最适于扩充以处理大的寄存器空间和缺少诸如“未定位”变量的参考值的数据对象。

         写一般参考值 (fc 21)

    此功能可以处理并发的多个请求，也可接收32位的参考数值。当前的584和984PLC’s仅使用此功能接收类型6的参考值（扩展的寄存器文件）。

    该功能最适于扩充以处理大的寄存器空间和缺少诸如“未定位”变量的参考值的数据对象。

         掩膜写寄存器 (fc 22)

         读/写寄存器 (fc 23)

    此功能把一定范围的寄存器输入和输出当作单一的处理事务。使用MODBUS是执行规则的带有I/O模块的状态影象交换的最好办法。

    如此，高性能的通用的数据采集装置可以执行功能3，16和23，从而把快捷的数据规则交换（23）和执行特殊数据对象的需求询问或更新的能力结合起来（3和16）。

         读FIFO队列 (fc 24)

     一个有点专用的功能，打算将表结构的数据象FIFO（用到584/984上的FIN和FOUT功能模块）一样传送到主机。对于某种事件录入软件很有用。

3.4机器/厂家/网络的特殊功能

    以下所有的功能，虽然在MODBUS协议手册中提到，但由于它们有很强的机器依赖性，因而不适于互用性的目的。

         诊断 (fc 8)

         编程 (484) (fc 9)

         轮询 (484) (fc 10)

         获取通讯事件计数器值(Modbus) (fc 11)

         获取通讯事件记录(Modbus) (fc 12)

         编程 (584/984) (fc 13)

         轮询(584/984) (fc 14)

         通告从站 ID (fc 17)

         编程 (884/u84) (fc 18)

         恢复通讯连接 (884/u84) (fc 19)

         编程 (原理) (fc 40)

         固件置换 (fc 125)

         编程 (584/984) (fc 126)

         通告本地地址 (Modbus) (fc 127)

4. 协议结构

    本部分阐述了通过MODBUS/TCP网络携带的MODBUS请求和或响应封装的一般格式。必须注意到请求和响应本体（从功能代码到数据部分的末尾）的结构和其它MODBUS变量具有完全相同的版面格式和含义，如：

MODBUS 串行端口 - ASCII 编码

MODBUS 串行端口 - RTU (二进制) 编码

MODBUS PLUS 网络 – 数据通道

    这些其它案例仅在组帧次序，检错模式和地址描述等格式有所不同。

     所有的请求通过TCP从寄存器端口502发出。 请求通常是在给定的连接以半双工的方式发送。也就是说，当单一连接被响应所占用，就不能发送其它的请求。有些装置采用多条TCP连接来维持高的传输速率。

     然而一些客户端设备尝试“流水线式”的请求。允许服务器以这种方式工作的技术在附录A中阐述。

MODBUS “从站地址”字段被单字节的“单元标识符”替换，从而用于通过网桥和网关等设备的通讯，这些设备用单一IP地址来支持多个独立的终接单元。

请求和响应带有六个字节的前缀，如下：

      byte 0:     事务处理标识符 –由服务器复制 –通常为 0

      byte 1:     事务处理标识符 –由服务器复制 –通常为 0

      byte 2:     协议标识符= 0

      byte 3:     协议标识符= 0

      byte 4:     长度字段 （上半部分字节） = 0 (所有的消息长度小于256)

      byte 5:     长度字段  (下半部分字节)   = 后面字节的数量

      byte 6:     单元标识符 (原“从站地址”)

      byte 7:     MODBUS 功能代码

      byte 8 on:   所需的数据

因而处理示例“以4的偏移从UI 9读1寄存器”返回5的值将是

     请求：00  00  00  00  00  06  09  03  00  04  00  01

     响应：00  00  00  00  00  05  09  03  02  00  05

一致性等级0-2的功能代码的应用的例子见后续部分

    熟悉MODBUS的设计师将注意到MODBUS/TCP中不需要“CRC-16”或“LRC”检查字段。而是采用TCP/IP和链路层（以太网）校验和机制来校验分组交换的准确性。

5. 一致性等级的协议参考值

    注意到在例子中，请求和响应列在功能代码字节的前面。如前所述，在MODBUS/TCP案例中有一个依赖传输的包含7个字节的前缀。

ref  ref  00  00  00  len  unit前面两个字节的“ref  ref”在服务器中没有具体的值，只是为方便客户端而从请求和响应中逐字的复制过来。单客户机通常将该值置为0。

    在这个例子中，请求和响应的格式如下（例子是“读寄存器”请求，详述见后面部分）。

    03 00 00 00 01  =>   03 02 12 34

    这表示给前缀加上一个十六进制的串联的字节，这样，TCP连接上的整个消息将是（假设单元标识符还是09）

    请求：  00 00 00 00 00 06 09 03 00 00 00 01

    响应：   00 00 00 00 00 05 09 03 02 12 34

    (所有的这些请求和响应通过查询Modicon Quantum PLC规范采用自动工具来进行校验。

5.1等级0指令详述

5.1.1读乘法寄存器(FC 3)

请求

     Byte 0:           FC = 03

     Byte 1-2:     参考数值

     Byte 3-4:     指令数(1-125)

响应

     Byte 0:           FC = 03

     Byte 1:           响应的字节数 (B=2 x指令数)

     Byte 2-(B+1):       Register values

异常

     Byte 0:           FC = 83 (hex)

Byte 1:           异常代码 = 01 or 02

示例:

    读参考值为0 (Modicon 984中为40001)时的1寄存器得到十六进制的值1234

        03 00 00 00 01  =>   03 02 12 34

5.1.2 写乘法寄存器(FC 16)

请求

     Byte 0:           FC = 10 (hex)

     Byte 1-2:     参考数值

     Byte 3-4:        指令数 (1-100)

     Byte 5:           字节数 (B=2 x word count)

     Byte 6-(B+5):       寄存器值

响应

     Byte 0:           FC = 10 (hex)

     Byte 1-2:        参考数值

     Byte 3-4:        指令数

异常

     Byte 0:           FC = 90 (hex)

     Byte 1:           异常代码 = 01 or 02

示例:

    读参考值为0(Modicon 984中为40001)时的1寄存器得到十六进制的值1234

    10 00 00 00 01 02 12 34  =>   10 00 00 00 01

5.2等级1指令详述

5.2.1 读线圈 (FC 1)

请求

     Byte 0:           FC = 01

     Byte 1-2:        参考数值

     Byte 3-4:        比特数(1-2000)

响应

     Byte 0:           FC = 01

     Byte 1:           响应的字节数 (B=(比特数+7)/8)

     Byte 2-(B+1):       比特值(最小意义位首先绕线圈!)

异常

     Byte 0:           FC = 81 (hex)

     Byte 1:           exception code = 01 or 02

示例

读参考值为0 (Modicon 984中为00001)时的1线圈得到的值1

     01 00 00 00 01  =>   01 01 01

注意到返回的数据的格式和big-endian体系结构不同。而且此请求如果调用乘法指令字且这些指令不以16位为界排列，那么该请求将在从站得到计算强化。

5.2.2读离散输入 (FC 2)

请求

     Byte 0:           FC = 02

     Byte 1-2:        参考数值

     Byte 3-4:        比特数 (1-2000)

响应

     Byte 0:           FC = 02

     Byte 1:           响应的字节数 (B=(比特数+7)/8)

     Byte 2-(B+1):       比特值 (最小意义位首先绕线圈!)

异常

     Byte 0:           FC = 82 (16进制)

     Byte 1:           异常代码 = 01 or 02

示例

读参考值为0 (Modicon 984中为10001)时的1离散输入得到的值1

     02 00 00 00 01  =>   02 01 01

注意到返回的数据的格式和big-endian体系结构不同。而且此请求如果调用乘法指令字且这些指令不以16位为界排列，那么该请求将在从站得到计算强化。

5.2.3 读输入寄存器 (FC 4)

请求

     Byte 0:           FC = 04

     Byte 1-2:        参考数值

     Byte 3-4:        指令数 (1-125)

响应

     Byte 0:           FC = 04

     Byte 1:           响应的比特数 (B=2 x 指令数)

     Byte 2-(B+1):       寄存器值

异常

     Byte 0:           FC = 84 (hex)

     Byte 1:           异常代码 = 01 or 02

示例

     读参考值为0 (Modicon 984中为30001)时的1输入寄存器得到十六进制的值1234

     04 00 00 00 01  =>   04 02 12 34

5.2.4 写线圈 (FC 5)

请求

     Byte 0:           FC = 05

     Byte 1-2:        参考数值

     Byte 3:           = FF 打开线圈, =00 关闭线圈

     Byte 4:           = 00

响应

     Byte 0:           FC = 05

     Byte 1-2:        参考数值

     Byte 3:           = FF 打开线圈, =00 关闭线圈(回波)

     Byte 4:           = 00

异常

     Byte 0:           FC = 85 (16进制)

     Byte 1:           异常代码 = 01 or 02

示例

将值1在参考值为0（Modicon 984中为00001）时写入1线圈

     05 00 00 FF 00  =>   05 00 00 FF 00

5.2.5 写单一寄存器(FC 6)

请求

     Byte 0:           FC = 06

     Byte 1-2:        参考数值

     Byte 3-4:        寄存器值

响应

     Byte 0:           FC = 06

     Byte 1-2:        参考数值

     Byte 3-4:        寄存器值

异常

     Byte 0:           FC = 86 (16进制)

     Byte 1:           异常代码= 01 or 02

示例

将十六进制值1234在参考值为0（Modicon 984中为40001）时写入1线圈

     06 00 00 12 34  =>   06 00 00 12 34