一、 简介

Modbus 协议最初由 Modicon 公司开发出来，在 1979 年末该公司成为施耐德自动化（Schneider Automation）部门的一部分，现在 Modbus 已经是工业领域全球最流行的协议。此协议支持传统的 RS-232、RS-485 和以太网设备等。

在 Modbus 网络中，可有 2～248 个设备，即 1 个主设备和 247 个从设备（最多）。每个从设备被赋予 1～247 范围中的地址。Modbus 通信总是由主设备发起，从设备在没有收到来自主设备的请求时，从不会发送数据。从设备之间从不会互相通信。主设备在同一时刻只会发起一个 Modbus 事务处理。

主设备通过将从设备的地址放到报文的地址域，对从设备寻址。当从设备返回应答时，它将自己的地址放到应答报文的地址域，以让主设备知道哪个从设备在回答。

本手册用于通过 Modbus 与质量流量计变送器进行通讯；主要阐述变送器与上层系统的通讯。此通讯基于 PI-MBUS-300，Rev.B 定义的标准 Modbus 协议（RTU 模式）。

二、 通讯参数

传输模式：RTU 模式
波特率：9600（默认）（可修改）
校验位：无校验（默认）（可修改）
起始位：1
数据位：8
停止位：1（默认）（可修改）
TX/RX 模式：半双工 支持 RTS，CTS，DTR，DSR 及 DCD
转向时间：>=150ms

三、 通讯过程

MODBUS 协议定义了一个控制器能认识使用的消息结构，而不管它们是经过何种网络进行通信的。它描述了控制器请求访问其它设备的过程，如何回应来自其它设备的请求，以及怎样侦测错误并记录。它制定了消息域格局和内容的公共格式。

使用 RTU 模式，消息发送至少要以 3.5 个字符时间的停顿间隔开始。在多样的网络波特率的字符时间下，这是最容易实现的(如下表的 T1-T2-T3-T4 所示)。传输的第一个域是设备地址。可以使用的传输字符是十六进制的 0…9，A…F。网络设备不断侦测网络总线，包括停顿间隔时间内。当第一个域（地址域）接收到，每个设备都进行解码以判断是否发往自己的。在最后一个传输字符之后，一个至少 3.5 个字符时间的停顿标定了消息的结束。一个新的消息可在此停顿后开始。

整个消息帧必须作为连续的流传输。如果在帧完成之前有超过 3.5 个字符时间的停顿时间，接收设备将刷新不完整的消息并假定下一字节是一个新消息的地址域。同样地，如果一个新消息在小于 3.5 个字符时间内接着前个消息开始，接收的设备将

认为它是前一消息的延续。这将导致一个错误，因为在最后的 CRC 域的值不可能是正确的。典型的消息帧如下所示：

起始	设备地址	功能码	数据	CRC 校验	结束
T1T2T3T4	8 bits	8 bits	n*8 bits	16 bits	T1T2T3T4

✧ 起　始：

3.5 个字符的间隔时间

✧ 地址域：

消息帧的地址域包含 8Bits（RTU）。可能的从设备地址是 0～247 (十进制)。单个　　设备的地址范围是 1～247。主设备通过将要联络的从设备的地址放入消息中的地址域，来选择从设备。当从设备发送响应消息时，它把自己的地址放入回应的地址域中，以便主设备知道是哪一个设备作出回应。地址 0 是用作广播地址，以使所有的从设备都能认识。当 Modbus 协议用于更高水准的网络，广播可能不允许或以其它方式代替。

✧ 功能码：

消息帧中的功能代码域包含了两个字符（ASCII）或 8Bits（RTU）。可能的代码范围是十进制的 1～255。当然，有些代码是适用于所有控制器，有些是应用于某种控制器，还有些保留以备后用。

当消息从主设备发往从设备时，功能代码域将告之从设备需要执行哪些行为。例如去读取输入的开关状态，读一组寄存器的数据内容，读从设备的诊断状态，允许调入、记录、校验在从设备中的程序等。

当从设备响应时，它使用功能码域来指示是正常响应(无误)还是有某种错误发生（称作异常响应）。对正常响应，从设备仅响应相应的功能代码。对异常响应，从设备返回等同于正常代码的代码，但最高位为逻辑 1。

✧ 数据域：

数据域是由一系列 2 个 8 位字节集合构成的，范围 0000～FFFF。主设备发给从设备消息的数据域包含附加的信息，从设备必须用于进行执行由功能码所定义的所为。这包括：例如不连续的寄存器地址，要处理项的数目，域中实际数据字节数。

如果没有错误发生，从设备返回的数据域包含请求的数据。如果有错误发生，此域包含异常代码，主设备应用程序可以用来判断采取下一步行动。

在某种消息中数据域可以是不存在的（0 长度）。功能码足以声明命令内容。

✧ CRC 校验：

当选用 RTU 模式作字符帧，错误检测域包含一个 16Bits 值(用两个 8 位的字符来实现)。错误检测域的内容是通过对消息内容进行循环冗长检测方法得出的。CRC 域附加在消息的最后，添加时先是低字节然后是高字节。故 CRC 的高位字节是发送消息的最后一个字节。

✧ 结　　束：

3.5 个字符的间隔时间

3.1 功能码

功能码　04：读取多寄存器

取得从设备多个保持寄存器的二进制值 。不支持广播。最大请求寄存器数量为FF(255)。

请求：请求信息定义了要读取的保持寄存器的起始地址和寄存器数量。

从设备地址	功能码 0x04	起始地址	寄存器数量	CRC 校验
8 bits	8 bits	16 bits	16 bits	16 bits

响应：响应信息数据域的每两个字节表示一个寄存器，每个字节的二进制内容向右对齐。

从设备地址	功能码 0x04	字节数（N）	数据域	CRC 校验
8 bits	8 bits	8 bits	(N)*8 bits	16 bits

相关寄存器定义如下：

读写状态	数据类型	地址范围	描述	数据范围
RO	F32	247-248	瞬时质量流速	
RO	F32	249-250	密度	
RO	F32	251-252	温度	
RO	F32	253-254	瞬时体积流速	
RO	F32	259-260	质量总量	
RO	F32	261-262	体积总量	

注：数据类型 F32：使用 32bit 存储单精度 IEEE754 格式的浮点数。每个浮点数包含 4 个字节，具体定义如下：

SEEEEEEE EMMMMMMM MMMMMMMM MMMMMMMM

S：符号位 0->正 1->负

E：阶码

M：尾数的小数部分

例如：0x 4148 0000=12.5，

注意：我们程序中使用的 IEEE754 数据采用的是反序(3-2-1-0)表示，即用 00004841 表示 12.5。

本例数据为读取瞬时质量流速的数据帧，仪表地址=1（HEX 码为 0x01）。

注：瞬时质量流速的寄存器起始地址=247-1=246（HEX 码为 0xf6）。

请求帧：上位机->仪表

数据场名称	RTU 示例数据（HEX）
帧头	NONE
仪表地址	01
功能码	04
寄存器起始地址	00,F6
寄存器数量	00,02
CRC 校验	91,F9
帧尾	NONE

应答帧：仪表->上位机

数据场名称	RTU 示例数据（HEX）
帧头	NONE
仪表地址	01
功能码	04
字节数	04
数据域	00,00,48,41
CRC 校验	0D,B4
帧尾	NONE

本应答帧返回为 IEEE754 格式的瞬时流量数据 00 00 48 41=12.5

3.2 MODBUS 异常响应

当主设备给从设备发送了一个请求，期望收到正常的响应。主设备的请求有可能出现下列四种情况之一：

- ✧ 从设备接收到主设备的请求，没有检测到通信错误，并正常处理，则从设备返回正常响应；
- ✧ 从设备因为通信故障没有接收到主设备的请求，则从设备不响应。主设备程序将最终处于请求超时状态；
- ✧ 从设备接收到主设备的请求，但是检测到通信错误（校验错误），则从设备不响应。主设备程序将最终处于请求超时状态；
- ✧ 从设备接收到主设备的请求，没有检测到通信错误，但是无法正常处理（例如主设备要求读取一个并不存在的寄存器地址），从设备将返回一个异常响应，告诉主设备异常的本质。

标准的异常响应码格式：

从设备地址	功能码	异常码	CRC 校验
8 bits	8 bits	8 bits	16 bits

异常响应信息与正常响应信息有以下两点不同：

功能码：

正常响应的功能码与请求的功能码相同。异常响应的功能码是把请求的功能码的最高位置为'1'。

数据域：

正常响应的数据域为请求的信息。异常响应的数据域为异常码，该异常码声明引起从设备异常的条件。

✧ 异常码 0x01：非法功能

请求中的功能码对于从设备来说是不允许的操作。该异常码在下列情况下产生：

- 当一个无效的功能码被请求
- 本协议仅支持功能码 01 、02、03、04、05、06、07、08、15、16

✧ 异常码 0x02：非法数据地址

请求中的数据地址对于从设备来说是不允许。该异常码在下列情况下产生：

- 超出自定义线圈或寄存器的地址范围
- 对只读线圈或寄存器进行写操作

✧ 异常码 0x03：非法数据值

发送数据数量与协议不符：

- ✧ 异常码 0x06：从设备忙

  从设备正在执行持续长时间的程序命令。主设备需在从设备空闲时重新发送指令。该异常码在下列情况下产生：

  - 从设备忙
  - 从设备不能产生响应
  - 从设备不能在超时时间内产生正确的响应

本例假设读取瞬时质量流速，但从设备读取到为非法数据值，所以将返回如下故障应答帧。

故障应答帧：仪表->上位机

数据场名称	RTU 示例数据（HEX）
帧头	NONE
仪表地址	01
功能码	84
故障码	03
CRC 校验	03,01
帧尾	NONE