Modubs通信软件规格书

目录

[第一章 引言 1](#_Toc54782453)

[1.1 编写目的 1](#_Toc54782454)

[第二章 需求概述 1](#_Toc54782455)

[2.1 项目背景 1](#_Toc54782456)

[2.2 需求概述 1](#_Toc54782457)

[2.3 系统结构 1](#_Toc54782458)

[第三章 系统功能需求 2](#_Toc54782459)

[3.1 Modbus协议 2](#_Toc54782460)

[3.1.1 功能码描述 2](#_Toc54782461)

[3.1.2 地址域 6](#_Toc54782462)

[3.1.2 异常码描述 6](#_Toc54782463)

[3.5 Modbus RTU主站 7](#_Toc54782464)

[3.5.1 Modbus RTU串口配置 7](#_Toc54782465)

[3.5.2 Modbus RTU消息帧格式 7](#_Toc54782466)

[3.5.3 Modbus RTU异常响应报文 7](#_Toc54782467)

[3.5.4 CRC校验 8](#_Toc54782468)

[3.6 Modbus TCP从站 8](#_Toc54782469)

[3.6.1 Modbus TCP 网络配置 8](#_Toc54782470)

[3.6.2 Modbus TCP消息帧格式 9](#_Toc54782471)

[3.6.3 Modbus TCP异常响应报文 9](#_Toc54782472)

[第四章 接口需求 9](#_Toc54782473)

[4.1 用户界面 9](#_Toc54782474)

[4.2 运行环境 10](#_Toc54782475)

# 第一章 引言

## 1.1 编写目的

为了明确软件需求、安排项目规划与进度、组织软件开发与测试，撰写本文档。

# 第二章 需求概述

## 2.1 项目背景

Modbus是Modicon公司为其PLC通讯而开发的一种通讯协议。从1979年问世至今，已经成为工业通讯领域的业界标准，最初的Modbus仅支持串口，分为RTU和ACSII两种信号传输模式（一般基于RS485串口通信媒介）。而随着时代的变迁，Modbus新增了TCP版本，可以通过以太网进行通讯。和其他工业通信协议相比，Modbus主要的优点包括内容公开，无版权要求，不用支付额外的费用，硬件简单，容易部署。本软件通过Modbus协议进行通信，便于Modbus协议的学习与使用。

## 2.2 需求概述

本软件主要是遵守Modbus主/从或客户端/服务器架构的协议，根据协议规定去读取（或写入）从机线圈和寄存器的值。其中主要实现的功能码有以下几种：

（0x01）读取线圈/离散量输出状态

（0x03）读取保持寄存器值

（0x0F）写多个线圈

（0x10）写多个保持寄存器

## 2.3 系统结构



图2.3 系统结构

# 第三章 系统功能需求

## 3.1 Modbus协议

Modbus协议是一种单主/多从的通信协议，其特点是在同一时间，总线上只能有一个主设备，但可以有一个或者多个（最多247个）从设备。Modbus通信总是由主设备发起，当从设备没有收到来自主设备的请求时，不会主动发送数据。从设备之间不能相互通信，主设备同时只能启动一个Modbus访问事务处理。正常的事务处理如图3.1所示。



图3.1 Modbus 事务处理（无差错）

### 3.1.1 功能码描述

（0x01）读取线圈/离散量输出状态

在一个远程设备中，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为一个线圈。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。其它线圈依次类推，一直到这个字节的高位端为止，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 |
| 字节数 | 1个字节 | N\* |
| 线圈状态 | 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。

（0x03）读取保持寄存器值

在一个远程设备中，使用该功能码读取保持寄存器连续块的内容。请求 PDU 说明了起始寄存器地址和寄存器数量。从零开始寻址寄存器。因此，寻址寄存器 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 |
| 寄存器数量 | 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＝寄存器数量

错误

|  |  |  |
| --- | --- | --- |
| 功能码 | 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 |
| 高起始地址 | 00 | 字节数 | 06 |
| 低起始地址 | 6B | 寄存器值Hi | 02 |
| 高寄存器编号 | 00 | 寄存器值Lo | 2B |
| 低寄存器编号 | 03 | 寄存器值Hi | 00 |
|  |  | 寄存器值Lo | 00 |
|  |  | 寄存器值Hi | 00 |
|  |  | 寄存器值Lo | 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 容分别表示为十六进制 00 00 和 00 64，或十进制 0 和 100。

（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，寻址线圈 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 | 0x0001 至 0x07B0 |
| 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 | 0x0001 至 0x07B0 |

\*N＝寄存器数量

错误

|  |  |  |
| --- | --- | --- |
| 功能码 | 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。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 值 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 线圈 | 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）。 传输的下一字节(十六进制 01)寻址为输出 29-28，在这种设置中，最低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 |  |  |

（0x10）写多个保持寄存器

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 | 0x10 |
| 起始地址 | 2个字节 | 0x0000至0xFFFF |
| 寄存器数量 | 2个字节 | 0x0001 至 0x0078 |
| 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 | 0x10 |
| 起始地址 | 2个字节 | 0x0000至0xFFFF |
| 寄存器数量 | 2个字节 | 1 至 123（0x7B） |

错误

|  |  |  |
| --- | --- | --- |
| 功能码 | 1个字节 | 0x90 |
| 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 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 |  |  |

### 3.1.2 地址域

所谓地址域，指的是Modbus通信帧中的地址字段，其内容为从设备地址。消息帧中可能的从设备地址是0～247（十进制），单个设备的实际地址范围是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通从设备。当从设备发送回应消息时，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做出回应。

### 3.1.2 异常码描述

异常响应报文由从站地址、功能码以及异常码构成。其中，功能码与正常响应报文不同，在异常响应报文中，功能码最高位（即MSB）被设置为1。因为Modbus协议中功能码占用一个字节，故用表达式描述为：异常功能码=正常功能码+0x80。异常码如表3-1所示。

表3-1 Modbus异常码

|  |  |  |
| --- | --- | --- |
| 代码 | 名称 | 含义 |
| 01 | 非法功能 | 从站设备不支持此功能码 |
| 02 | 非法数据地址 | 指定的数据地址在从站设备中不存在 |
| 03 | 非法数据值 | 指定的数据超过范围或者不允许使用 |
| 04 | 从站设备故障 | 从站设备处理响应的过程中，出现未知错误等 |

## 3.2 Modbus RTU主站

Modbus RTU主站主要功能是通过向从站发送报文（数据请求），从站接收到并且处理完成请求后，从站会向主站返回一个报文（数据响应），主站对返回的报文进行解析可以获得相关信息。

Modbus RTU主站主要有四种功能码，分别是（0x01）读取线圈/离散量输出状态、（0x03）读取保持寄存器值、（0x0F）写多个线圈和（0x10）写多个保持寄存器，每种功能对应不同的数据格式。

### 3.2.1 Modbus RTU串口配置

串口配置如表3-2所示。

表3-2 串口配置

|  |  |
| --- | --- |
| 端口 | 自动获取端口 |
| 波特率 | 1200、4800、9600、19200、38400、57600、115200 |
| 数据位 | 7、8 |
| 停止位 | 1、2、1.5 |
| 校验位 | None、Odd、Even |

### 3.2.2 Modbus RTU消息帧格式

一帧正常的Modbus RTU数据帧包含的内容有：地址域 + 功能码 + 数据 + CRC校验。

|  |  |  |  |
| --- | --- | --- | --- |
| 地址域 | 功能码 | 数据 | CRC校验 |
| 1个字节 | 1个字节 | N个字节 | 2个字节 |

### 3.2.3 Modbus RTU异常响应报文

Modbus RTU异常响应报文格式：地址域+异常功能码（0x80+功能码）+异常码+CRC校验。

|  |  |  |  |
| --- | --- | --- | --- |
| 地址域 | 异常功能码 | 异常码 | CRC校验 |
| 1个字节 | 1个字节 | 1个字节 | 2个字节 |

### 3.2.4 CRC校验

在ModbusRTU传输模式下，通信报文（帧）包括一个基于循环冗余校验（CRC）方法的差错校验字段。

Modbus协议中，采用了CRC-16标准校验方法。在RTU模式下，CRC自身由两个字节组成，即CRC是一个16位的值。CRC字段校验整个报文的内容，无论报文中的单个字节采用何种奇偶校验方式，整个通信报文均可应用CRC-16校验算法。CRC字段作为报文的最后字段添加到整个报文末尾。

接收设备在接收信息时，会通过CRC算法重新计算，并把计算值与CRC字段中接收的实际值进行比较。若两者不同，则产生一个错误，并返回一个异常响应报文（帧）告知发送设备。

Modbus协议中的RTU校验码（CRC）计算，运算规则（即CRC计算方法）如下：

（1）预置一个值为0xFFFF的16位寄存器，此寄存器为CRC寄存器。

（2）把第1个8位二进制数据（即通信消息帧的第1个字节）与16位的CRC寄存器的相异或，异或的结果仍存放于该CRC寄存器中。

（3）把CRC寄存器的内容右移一位，用0填补最高位，并检测移出位是0还是1。

（4）如果移出位为零，则重复步骤（3）（再次右移一位）；如果移出位为1，则CRC寄存器与0xA001进行异或。

（5）重复步骤（3）和（4），直到右移8次，这样整个8位数据全部进行了处理。

（6）重复步骤（2）～（5），进行通信消息帧下一个字节的处理。

（7）将该通信消息帧所有字节按上述步骤计算完成后，得到的16位CRC寄存器的高、低字节进行交换。即发送时首先添加低位字节，然后添加高位字节。

（8）最后得到的CRC寄存器内容即为CRC校验码。

## 3.3 Modbus TCP从站

Modbus TCP从站主要是对主站发送的请求数据进行解析，然后进行对应的操作，之后返回数据给到主站。

### 3.3.1 Modbus TCP 网络配置

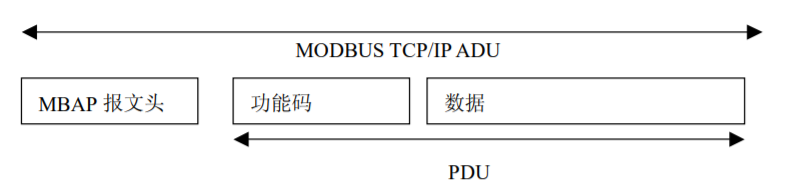
网络配置如表3-9所示。

表3-9 网络配置

|  |  |
| --- | --- |
| 互联网通信协议 | IPv4 |
| IP地址 | 自动获取 |
| 端口号 | 用户输入 |

### 3.3.2 Modbus TCP消息帧格式

ModbusTCP的数据帧包含的内容有：MBAP+PUD。如下图所示。



MBAP报文头，长度为7字节，组成如表3-10所示

表3-10 MBAP报文头

|  |  |  |  |
| --- | --- | --- | --- |
| 事务处理标识 | 协议标识符 | 长度 | 单元标识符 |
| 2个字节 | 2个字节 | 2个字节 | 1个字节 |

事务处理标识：可以理解为报文的序列号，一般每次通信之后就要加1以区别不同的通信数据报文。

协议标识符：00 00表示ModbusTCP协议。

长度：表示接下来的数据长度，单位为字节。

单元标识符：可以理解为设备地址。

### 3.3.3 Modbus TCP异常响应报文

Modbus TCP异常响应报文格式：MBAP+异常功能码（0x80+功能码）+异常码。

|  |  |  |
| --- | --- | --- |
| MBAP | 异常功能码 | 异常码 |
| 7个字节 | 1个字节 | 1个字节 |

# 第四章 接口需求

## 4.1 用户界面

本设计将为用户提供美观,大方,直观,操作简单，软件运行提供两种模式的选择。

## 4.2 运行环境

Window10