

**Modbus协议栈**

**编程手册**

作者：木南

版本：V2.0

# 1、概述

Modbus是全球第一个真正用于工业现场的总线协议。Modbus网络是一个工业通信系统，由带智能终端的可编程序控制器和计算机通过公用线路或局部专用线路连接而成。其系统结构既包括硬件、亦包括软件。它可应用于各种数据采集和过程监控。

现在Modbus支持串行链路和以太网链路两种硬件形式，从协议上说有Modbus RTU、Modbus ASCII、Modbus TCP。现在用的最多的就是基于串行链路的Modbus RTU和基于以太网的Modbus TCP两种。虽然在国标中规定，在串行链路上，Modbus RTU协议是默认项，而Modbus ASCII为可选项，出于协议站的完整性，我们一并实现之。

前一个版本我们已经实现了基于串行链路的Modbus RTU和基于以太网的Modbus TCP两种。在这一个版本中，我们添加了基于串行链路的Modbus ASCII方式。并对上一个版本中已知的问题做了修正。

# 2、函数描述

在本节中我们将对全部封装并开放的函数进行描述，包括功能，参数，返回值等。

## 2.1、应用数据单元

在应用数据单元我们处理Modbus数据通讯的基本报文格式，这些基本是通用的，目前版本只支持01、02、03、04、05、06、15、16功能码。

### 2.2.1、功能码枚举

/\*定义Modbus的操作功能码，支持01、02、03、04、05、06、15、16功能码\*/

typedef enum {

ReadCoilStatus=0x01, /\*读线圈状态（读多个输出位的状态）\*/

ReadInputStatus=0x02, /\*读输入位状态（读多个输入位的状态）\*/

ReadHoldingRegister=0x03, /\*读保持寄存器（读多个保持寄存器的数值）\*/

ReadInputRegister=0x04, /\*读输入寄存器（读多个输入寄存器的数值）\*/

WriteSingleCoil=0x05, /\*强制单个线圈（强制单个输出位的状态）\*/

WriteSingleRegister=0x06, /\*预制单个寄存器（设定一个寄存器的数值）\*/

WriteMultipleCoil=0x0F, /\*强制多个线圈（强制多个输出位的状态）\*/

WriteMultipleRegister=0x10 /\*预制多个寄存器（设定多个寄存器的数值）\*/

}FunctionCode;

### 2.1.2、Modbus状态枚举

/\*定义接收到指令检测错误时的错误码\*/

typedef enum{

MB\_OK=0x00,

InvalidFunctionCode=0x01,//不合法功能代码

IllegalDataAddress=0x02,//非法的数据地址

IllegalDataValues=0x03,//非法的数据值或者范围

OperationFail=0x04

}ModbusStatus;

### 2.1.3、站信息结构

/\*定义用于传递要访问从站（服务器）的信息\*/

typedef struct{

uint8\_t unitID;

FunctionCode functionCode;

uint16\_t startingAddress;

uint16\_t quantity;

}ObjAccessInfo;

定义用于传递从站（服务器）访问的各种信息，包括站号、功能码、起始地址和数量。

### 2.1.4、生成访问从站PDU单元函数

/\*作为RTU主站（TCP客户端）时，生成读写RTU从站（TCP服务器）对象的命令\*/

uint16\_t GenerateReadWriteCommand(ObjAccessInfo objInfo,bool \*statusList,uint16\_t \*registerList,uint8\_t commandBytes[]);

**参数：**ObjAccessInfo objInfo，要访问的从站对象

bool \*statusList，需要往下写的状态量对象数据列表

uint16\_t \*registerList，需要往下写的寄存器量对象的数据列表

uint8\_t commandBytes[]，生成的访问命令

**返回值：**生成的访问命令的长度，以字节为单位。

### 2.1.5、解析读取的返回数据函数

/\*解析主站（客户端）从服务器读取的数据\*/

void TransformClientReceivedData(uint8\_t \* receivedMessage,uint16\_t quantity,bool \*statusList,uint16\_t \*registerLister);

**参数：**uint8\_t \* receivedMessage，接收到的信息报文

uint16\_t quantity，读取的对象数量

bool \*statusList，解析出来的读取到的状态对象的数据列表

uint16\_t \*registerLister，解析出来的读取到的寄存器对象的数据列表

**返回值：**无

### 2.1.6、从站生成读数据命令的响应报文函数

/\*生成主站读访问的响应，包括0x01、0x02、0x03、0x04功能码,返回相应信息的长度\*/

uint16\_t GenerateMasterAccessRespond(uint8\_t \*receivedMesasage,bool \*statusList,uint16\_t \*registerList,uint8\_t \*respondBytes);

**参数：**uint8\_t \*receivedMesasage，接收到的主战读操作报文

bool \*statusList，读取的从站状态对象数据列表

uint16\_t \*registerList，读取到的从站寄存器对象数据列表

uint8\_t \*respondBytes，生成的从站响应报文信息

**返回值：**响应报文的长度，以字节为单位

### 2.1.7、功能码检验函数

/\*检查功能码是否正确\*/

ModbusStatus CheckFunctionCode(FunctionCode fc);

**参数：**FunctionCode fc，功能码，由枚举定义

**返回值：**Modbus操作状态，由枚举定义

### 2.1.8、生成TCP客户端命令函数

//生成读写服务器对象的命令

uint16\_t SyntheticReadWriteTCPServerCommand(ObjAccessInfo objInfo,bool \*statusList,uint16\_t \*registerList,uint8\_t \*commandBytes);

**参数：**ObjAccessInfo objInfo，TCP服务器访问信息对象

bool \*statusList，写服务器状态量对象数据列表

uint16\_t \*registerList，写服务器寄存器量对象数据列表

uint8\_t \*commandBytes，生成的命令信息报文

**返回值：**命令的长度，以字节为单位

### 2.1.9、生成TCP服务器响应函数

/\*合成对服务器访问的响应\*/

uint16\_t SyntheticServerAccessRespond(uint8\_t \*receivedMesasage,bool \*statusList,uint16\_t \*registerList,uint8\_t \*respondBytes);

**参数：**uint8\_t \*receivedMesasage，接收到的客户端命令信息报文

bool \*statusList，读TCP服务器的状态量对象数据列表

uint16\_t \*registerList，读TCP服务器的寄存器量对象数据列表

uint8\_t \*respondBytes，生成的响应客户端命令的响应信息报文

**返回值：**响应信息报文长度，以字节为单位

### 2.1.10、CRC校验检查函数

/\*通过CRC校验接收的信息是否正确\*/

bool CheckRTUMessageIntegrity (uint8\_t \*message,uint8\_t length);

**参数：**uint8\_t \*message，待校验的数据报文

uint8\_t length，带教言的数据报文长度，以字节为单位

**返回值：**返回校验是否通过的状态信号

### 2.1.11、生成读写RTU从站的主站命令函数

/\*生成读写从站数据对象的命令,命令长度包括2个校验字节\*/

uint16\_t SyntheticReadWriteSlaveCommand(ObjAccessInfo slaveInfo,bool \*statusList,uint16\_t \*registerList,uint8\_t \*commandBytes);

**参数：**ObjAccessInfo slaveInfo，从站访问信息对象，由结构体定义

bool \*statusList，写从站状态量数据列表

uint16\_t \*registerList，写从站寄存器量数据列表

uint8\_t \*commandBytes，生成的访问命令报文

**返回值：**访问命令的长度，以字节为单位

### 2.1.12、生成从站应答信息报文函数

/\*生成从站应答主站的响应\*/

uint16\_t SyntheticSlaveAccessRespond(uint8\_t \*receivedMesasage,bool \*statusList,uint16\_t \*registerList,uint8\_t \*respondBytes);

**参数：**uint8\_t \*receivedMesasage，接受到主站报文信息

bool \*statusList，主站命令要读取的状态量值列表

uint16\_t \*registerList，主站命令要读取的寄存器量值列表

uint8\_t \*respondBytes，生成的响应信息报文

**返回值：**生成的响应信息报文的长度，以字节为单位

### 2.1.13、LRC校验检查函数

/\*判断ASCII数据信息是否正确\*/

bool CheckASCIIMessageIntegrity(uint8\_t \*usMsg, uint16\_t usLength)

**参数：**uint8\_t \*usMsg，待校验的消息列表，16进制数据

uint16\_t usLength，待校验的消息列表的长度

**返回值：**校验结果，校验正确返回“true”，否则返回“false”

### 2.1.14、将ASCII消息转换为16进制

/\*接收到的ASCII消息转换为16进制\*/

bool CovertAsciiMessageToHex(uint8\_t \*aMsg, uint8\_t \*hMsg, uint16\_t aLen);

**参数：**uint8\_t \*aMsg，ASCII码信息

uint8\_t \*hMsg，转换后的16进制信息

uint16\_t aLen，ASCII码信息的长度

**返回值：**转换是否正确，转换正确返回“true”，否则返回“false”

### 2.1.15、生成读写ASCII从站的命令函数

/\*生成读写从站的命令，应用于主站，含校验及起始结束符\*/

uint16\_t SyntheticReadWriteAsciiSlaveCommand(ObjAccessInfo slaveInfo, bool \*statusList, uint16\_t \*registerList, uint8\_t \*commandBytes)

**参数：**ObjAccessInfo slaveInfo，从站访问信息对象，由结构体定义

bool \*statusList，写从站状态量数据列表

uint16\_t \*registerList，写从站寄存器量数据列表

uint8\_t \*commandBytes，生成的访问命令报文

**返回值：**访问命令的长度，以字节为单位

### 2.1.16、生成ASCII从站应答主站命令的函数

/\*生成应答主站的响应，应用于从站\*/

uint16\_t SyntheticAsciiSlaveAccessRespond(uint8\_t \*receivedMessage, bool \*statusList, uint16\_t \*registerList, uint8\_t \*respondBytes)

**参数：**uint8\_t \*receivedMesasage，接受到主站报文信息

bool \*statusList，主站命令要读取的状态量值列表

uint16\_t \*registerList，主站命令要读取的寄存器量值列表

uint8\_t \*respondBytes，生成的响应信息报文

**返回值：**生成的响应信息报文的长度，以字节为单位

## 2.2、应用封装

对应用级封装，我们只封装RTU主站应用，RTU从站应用、TCP客户端应用和TCP服务器端应用四类。

### 2.2.1、生成RTU主站访问从站命令函数

/\*生成访问服务器的命令\*/

uint16\_t CreateAccessSlaveCommand(ObjAccessInfo objInfo,void \*dataList,uint8\_t \*commandBytes);

**参数：**ObjAccessInfo objInfo，从站访问信息对象，由结构体定义

void \*dataList，写从站对象的数据列表

uint8\_t \*commandBytes，生成的命令报文

**返回值：**生成的访问命令的长度，以字节为单位

### 2.2.2、RTU主站解析从站的响应函数

/\*解析收到的服务器响应信息\*/

void ParsingSlaveRespondMessage(uint8\_t \*recievedMessage,uint8\_t \*command);

**参数：**uint8\_t \*recievedMessage，接收到的返回信息

uint8\_t \*command，已经下发到从站的命令列表

**返回值：**无

### 2.2.3、RTU主站解析从站返回消息命令匹配函数

/\*接收到返回信息后，判断是否是发送命令列表中命令的返回信息\*/

int FindCommandForRecievedMessage(uint8\_t \*recievedMessage,uint8\_t (\*commandList)[8],uint16\_t commandNumber);

**参数：**uint8\_t \*recievedMessage，接收的从站返回信息

uint8\_t (\*commandList)[8]，主站下发过的命令列表

uint16\_t commandNumber，存储的命令的条数

**返回值：**返回匹配命令所在的位置，无匹配时，返回“-1”

本函数只在RTU主站中，访问的数据量大且频繁，或者访问多个从站时使用，便于作正确的数据解析。

### 2.2.4、RTU从站解析主站访问命令函数

/\*解析接收到的信息，并返回合成的回复信息和信息的字节长度，通过回调函数\*/

uint16\_t ParsingMasterAccessCommand(uint8\_t \*receivedMesasage,uint8\_t \*respondBytes,uint16\_t rxLength);

**参数：**uint8\_t \*receivedMesasage，接收到的主站报文信息

uint8\_t \*respondBytes，生成响应信息报文

uint16\_t rxLength，接收到的报文的长度

**返回值：**生成的响应信息的长度，以字节为单位

### 2.2.5、生成TCP客户端访问服务器命令函数

/\*生成访问服务器的命令\*/

uint16\_t CreateAccessServerCommand(ObjAccessInfo objInfo,void \*dataList,uint8\_t \*commandBytes);

**参数：**ObjAccessInfo objInfo，访问服务器信息对象

void \*dataList，些服务器对象数据列表

uint8\_t \*commandBytes，生成的访问服务器命令报文

**返回值：**访问命令的长度，以字节为单位

### 2.2.6、TCP客户端解析服务器返回信息函数

/\*解析收到的服务器相应信息\*/

void ParsingServerRespondMessage(uint8\_t \*recievedMessage);

**参数：**uint8\_t \*recievedMessage，接受到的服务器响应信息

**返回值：**无

### 2.2.7、TCP服务器解析客户端命令函数

/\*解析接收到的信息，返回响应命令的长度\*/

uint16\_t ParsingClientAccessCommand(uint8\_t \*receivedMessage,uint8\_t \*respondBytes);

**参数：**uint8\_t \*receivedMessage，接受到的客户端访问命令

uint8\_t \*respondBytes，生成的服务器响应信息

**返回值：**生成的响应信息报文的长度，以字节为单位

### 2.2.8、生成访问ASCII从站的命令

/\*生成访问ASCII从站的命令\*/

uint16\_t CreateAccessAsciiSlaveCommand(ObjAccessInfo objInfo,void \*dataList,uint8\_t \*commandBytes);

**参数：**ObjAccessInfo objInfo，从站访问信息对象，由结构体定义

void \*dataList，写从站对象的数据列表

uint8\_t \*commandBytes，生成的命令报文

**返回值：**生成的访问命令的长度，以字节为单位

### 2.2.9、ASCII主站解析从站的响应函数

/\*解析收到的服务器相应信息\*/

void ParsingAsciiSlaveRespondMessage(uint8\_t \*recievedMessage, uint8\_t \*command,uint16\_t rxLength);

**参数：**uint8\_t \*recievedMessage，接收到的返回信息

uint8\_t \*command，已经下发到从站的命令列表

uint16\_t rxLength，接收到的消息的长度

**返回值：**无

### 2.2.10、ASCII主站解析从站返回消息命令匹配函数

/\*接收到返回信息后，判断是否是发送命令列表中命令的返回信息\*/

int FindAsciiCommandForRecievedMessage(uint8\_t \*recievedMessage,uint8\_t (\*commandList)[17],uint16\_t commandNumber);

**参数：**uint8\_t \*recievedMessage，接收的从站返回信息

uint8\_t (\*commandList)[17]，主站下发过的命令列表

uint16\_t commandNumber，存储的命令的条数

**返回值：**返回匹配命令所在的位置，无匹配时，返回“-1”

本函数只在RTU主站中，访问的数据量大且频繁，或者访问多个从站时使用，便于作正确的数据解析。

### 2.2.11、ASCII从站解析主站访问命令函数

/\*解析接收到的信息，并返回合成的回复信息和信息的字节长度，通过回调函数\*/

uint16\_t ParsingAsciiMasterAccessCommand(uint8\_t \*receivedMessage, uint8\_t \*respondBytes, uint16\_t rxLength, uint8\_t StationAddress);

**参数：**uint8\_t \*receivedMesasage，接收到的主站报文信息

uint8\_t \*respondBytes，生成响应信息报文

uint16\_t rxLength，接收到的报文的长度

**返回值：**生成的响应信息的长度，以字节为单位

## 2.3、数据处理函数

具体应用中还涉及到对数据对想的处理，不同的应用数据结构与处理方式千差万别，所以定义一些回调函数（实际使用弱化的函数\_\_weak关键字）用于数据的处理。

### 2.3.1、获取线圈量值的函数

/\*获取想要读取的Coil量的值\*/

\_\_weak void GetCoilStatus(uint16\_t startAddress,uint16\_t quantity,bool \*statusList)

{

//如果需要Modbus TCP Server/RTU Slave应用中实现具体内容

}

**参数：**uint16\_t startAddress，要读取对象的起始地址

uint16\_t quantity，要读取对象的数量

bool \*statusList，获取的线圈量的数据值列表

**返回值：**无

### 2.3.2、获取输入状态值的函数

/\*获取想要读取的InputStatus量的值\*/

\_\_weak void GetInputStatus(uint16\_t startAddress,uint16\_t quantity,bool \*statusValue)

{

//如果需要Modbus TCP Server/RTU Slave应用中实现具体内容

}

**参数：**uint16\_t startAddress，要读取对象的起始地址

uint16\_t quantity，要读取对象的数量

bool \*statusValue，获取的输入状态量的数据值列表

**返回值：**无

### 2.3.3、获取保持寄存器值的函数

/\*获取想要读取的保持寄存器的值\*/

\_\_weak void GetHoldingRegister(uint16\_t startAddress,uint16\_t quantity,uint16\_t \*registerValue)

{

//如果需要Modbus TCP Server/RTU Slave应用中实现具体内容

}

**参数：**uint16\_t startAddress，要读取的保持寄存器对象的起始地址

uint16\_t quantity，要读取的保持寄存器的数量

uint16\_t \*registerValue，获取的保持寄存器的值

**返回值：**无

### 2.3.4、获取输入寄存器值的函数

/\*获取想要读取的输入寄存器的值\*/

\_\_weak void GetInputRegister(uint16\_t startAddress,uint16\_t quantity,uint16\_t \*registerValue)

{

//如果需要Modbus TCP Server/RTU Slave应用中实现具体内容

}

**参数：**uint16\_t startAddress，要读取的输入寄存器的起始地址

uint16\_t quantity，要读取的输入寄存器的数量

uint16\_t \*registerValue，获取的输入寄存器的值

**返回值：**无

### 2.3.5、设置单个线圈的值函数

/\*设置单个线圈的值\*/

\_\_weak void SetSingleCoil(uint16\_t coilAddress,bool coilValue)

{

//如果需要Modbus TCP Server/RTU Slave应用中实现具体内容

}

**参数：**uint16\_t coilAddress，所需要预置值的线圈地址

bool coilValue，所预置的值

**返回值：**无

### 2.3.6、设置单个寄存器值的函数

/\*设置单个寄存器的值\*/

\_\_weak void SetSingleRegister(uint16\_t registerAddress,uint16\_t registerValue)

{

//如果需要Modbus TCP Server/RTU Slave应用中实现具体内容

}

**参数：**uint16\_t registerAddress，所预置的保持寄存器的地址

uint16\_t registerValue，所预置的保持寄存器的值

**返回值：**无

### 2.3.7、设置多个线圈值的函数

/\*设置多个线圈的值\*/

\_\_weak void SetMultipleCoil(uint16\_t startAddress,uint16\_t quantity,bool \*statusValue)

{

//如果需要Modbus TCP Server/RTU Slave应用中实现具体内容

}

**参数：**uint16\_t startAddress，要预置的多个线圈的起始地址

uint16\_t quantity，所要预知的线圈的数量

bool \*statusValue，要预知的线圈值的列表

**返回值：**无

### 2.3.8、设置多个寄存器值的函数

/\*设置多个寄存器的值\*/

\_\_weak void SetMultipleRegister(uint16\_t startAddress,uint16\_t quantity,uint16\_t \*registerValue)

{

//如果需要Modbus TCP Server/RTU Slave应用中实现具体内容

}

**参数：**uint16\_t startAddress，所预置的多个保持寄存器的起始地址

uint16\_t quantity，所要预置的保持寄存器的数量

uint16\_t \*registerValue，预知的保持寄存器值的列表

**返回值：**无

### 2.3.9、从站更新获取到的线圈量值的函数

/\*更新读回来的线圈状态\*/

\_\_weak void UpdateCoilStatus(uint16\_t startAddress,uint16\_t quantity,bool \*stateValue)

{

//在客户端（主站）应用中实现

}

**参数：**uint16\_t startAddress，读取值的线圈对象的起始地址

uint16\_t quantity，所读取的线圈的数量

bool \*stateValue，读取的线圈地址的列表

**返回值：**无

### 2.3.10、从站更新获取到的输入状态值的函数

/\*更新读回来的输入状态值\*/

\_\_weak void UpdateInputStatus(uint16\_t startAddress,uint16\_t quantity,bool \*stateValue)

{

//在客户端（主站）应用中实现

}

**参数：**uint16\_t startAddress，所读取的输入状态对象的起始地址

uint16\_t quantity，所读取的输入状态对象的数量

bool \*stateValue，读取到的输入状态对象的值

**返回值：**无

### 2.3.11、从站更新保持寄存器值的函数

/\*更新读回来的保持寄存器\*/

\_\_weak void UpdateHoldingRegister(uint16\_t startAddress,uint16\_t quantity,uint16\_t \*registerValue)

{

//在客户端（主站）应用中实现

}

**参数：**uint16\_t startAddress，所读取的保持寄存器对象的起始地址

uint16\_t quantity，所读取的保持寄存器对象的数据

uint16\_t \*registerValue，读取到的保持寄存器对象值列表

**返回值：**无

### 2.3.12、从站更新输入寄存器值得函数

/\*更新读回来的输入寄存器\*/

\_\_weak void UpdateInputResgister(uint16\_t startAddress,uint16\_t quantity,uint16\_t \*registerValue)

{

//在客户端（主站）应用中实现

}

**参数：**uint16\_t startAddress，所读取的输入寄存器对象的起始地址

uint16\_t quantity，所读取的输入寄存器对象的数量

uint16\_t \*registerValue，读取到的输入寄存器值得列表

**返回值：**无

# 3、应用说明

本协议mbconfig.h