MODBUS 串行线规范与实现指南

MODBUS over Serial Line Specification and Implementation Guide

版本：V1.02

日期：2006年12月20日

中文翻译版 | www.modbus.cn

本文件为 Modbus.org
官方规范文档的中文翻译版，仅供学习参考之用。官方英文原版请访问 www.modbus.org 获取。如翻译内容与英文原版存在歧义，以英文原版为准。

文档修改记录

版本	月份-年份	修改说明
1.0	2002年11月	创建文档。本文档包含对主从协议（Master/Slave）和两种传输模式（RTU、ASCII）的描述。提供了物理层的主要特性（RS485、RS232）及一些建议。提出了实现类以指导实现。
1.01	2006年8月30日	少量澄清和拼写纠正。
1.02	2006年12月20日	少量澄清和拼写纠正。

1 引言

1.1 本文档的范围

MODBUS 标准定义了一个应用层报文协议，位于 OSI 模型的第 7 层，提供连接在不同类型总线或网络上的设备之间的”客户端/服务器”（client/server）通信。它还标准化了一个串行线上的专用协议，用于在主站（Master）和一个或多个从站（Slave）之间交换 MODBUS 请求。

本文档的目标是介绍串行线上的 MODBUS 协议，供所有系统设计者在其串行线产品上实现 MODBUS 协议时使用。因此，本文档将促进使用 MODBUS 协议的设备之间的互操作性。本文档是对”MODBUS 应用协议规范”（MODBUS Application Protocol Specification）文档的补充。

在第 5 章中，为”MODBUS 串行线”定义了不同的实现类。一个类的规范是设备为属于该类而必须遵守的要求的总和。

图 1：MODBUS 文档总体概览
MODBUS 应用协议规范 —— MODBUS 应用协议（OSI 第 7 层）
MODBUS 串行线规范与实现指南 —— 串行线规范（OSI 第 1 和 2 层）【本文档】

1.2 协议概述

本文档描述了串行线上的 MODBUS 协议。MODBUS 串行线协议是一个主从协议（Master-Slave）。该协议位于 OSI 模型的第 2 层。

主从类型系统有一个节点（主站节点）向某个”从站”节点发出显式命令并处理响应。从站节点通常不会在主站节点没有请求的情况下发送数据，也不会与其他从站通信。在物理层面，MODBUS 串行线系统可使用不同的物理接口（RS485、RS232）。TIA/EIA-485（RS485）双线接口是最常见的。作为附加选项，也可实现 RS485 四线接口。当仅需短距离点对点通信时，也可使用

TIA/EIA-232-E（RS232）串行接口。（参见”物理层”章节）

下图给出了 MODBUS 串行通信协议栈与 OSI 模型 7 层的对比示意图。

层	ISO/OSI 模型	
7	应用层	MODBUS 应用协议
6	表示层	空
5	会话层	空
4	传输层	空
3	网络层	空
2	数据链路层	MODBUS 串行线协议
1	物理层	EIA/TIA-485（或 EIA/TIA-232）

图 2：MODBUS 协议与 ISO/OSI 模型

MODBUS 应用层报文协议位于 OSI 模型的第 7 层，提供连接在总线或网络上的设备之间的客户端/服务器通信。在 MODBUS 串行线上，客户角色由串行总线的主站承担，从站节点充当服务器。

1.3 约定

本文档中使用以下词语来定义各项特定要求的重要性。

- "MUST" / "REQUIRED"（必须/要求）：所有包含"MUST"的要求均为强制性要求。MUST 一词或形容词"REQUIRED"表示该项是实现的一项绝对要求。这些词以下划线标注。
- "SHOULD" / "RECOMMENDED"（应该/推荐）：所有包含"SHOULD"或形容词"RECOMMENDED"的建议被视为期望行为。在选择不同方案来实现功能时，这些建议应作为指导方针。在特定情况下可能有正当理由忽略该项，但在选择不同方案之前，应充分理解其全部影响并仔细权衡。这些词以下划线标注。
- "MAY" / "OPTIONAL"（可以/可选）："MAY"一词或形容词"OPTIONAL"表示该项是真正可选的。一个设计者可能因为特定市场需求或因为它增强了产品而选择包含该项；另一个设计者可能省略同一项。

1.4 合规性

如果实现未能满足其实现类中一个或多个 MUST 要求，则该实现不符合规范。

满足所有 MUST 要求和所有 SHOULD 建议的实现被称为”无条件合规”（unconditionally compliant）。

满足所有 MUST 要求但未满足所有 SHOULD 建议的实现被称为”有条件合规”（conditionally compliant）。

1.5 术语表

本文档中使用的特定词语、符号和缩写的定义。

术语	说明
2W	在”电气接口”章节中定义的双线配置，或其接口之一。
4W	在”电气接口”章节中定义的四线配置，或其接口之一。
AUI	Attachment Unit Interface，附加单元接口
AWG	American Wire Gauge，美国线规，表示线径的标准方法；参见附录 E - 参考文献。
Common	EIA/TIA 标准中的信号公共端。在 2W 或 4W RS485 MODBUS 网络中，为信号和可选电源公共端。
DCE	MODBUS 设备，例如可编程控制器适配器，实现 RS232 数据电路端接设备，也称数据通信设备。参见”MODBUS 设备”定义。
Driver	驱动器，发生器或发送器。
DTE	MODBUS 设备，例如编程面板或 PC，实现 RS232 数据终端设备。
ITr	干线侧物理总线接口。
IDv	分支侧（抽头或设备分支）物理总线接口。
LT	Line Termination，线路终端。
MODBUS Device	实现串行线 MODBUS 并遵守本技术说明的设备。
RS232	EIA/TIA-232 标准。
RS485	EIA/TIA-485 标准。
RS485-MODBUS	符合本技术说明的 2W 或 4W 网络。
Transceiver	收发器，一个发送器和一个接收器（或驱动器和接收器）。

2 MODBUS 数据链路层

2.1 MODBUS 主从协议原理

MODBUS 串行线协议是一个主从协议。同一时间只有一个主站连接到总线，一个或多个（最多 247 个）从站节点也连接到同一串行总线。MODBUS 通信始终由主站发起。从站节点不会在没有收到主站请求的情况下发送数据。从站节点之间不会互相通信。主站节点同一时间只发起一个 MODBUS 事务。

主站节点以两种模式向从站节点发出 MODBUS 请求：

- 单播模式（Unicast mode）：主站向单个从站寻址。从站在收到并处理请求后，向主站返回一条消息（”应答”）。在此模式下，一个 MODBUS 事务由 2 条消息组成：来自主站的请求和来自从站的应答。每个从站必须有一个唯一地址（1 到 247），以便能够独立于其他节点被寻址。
- 广播模式（Broadcast mode）：主站可以向所有从站发送请求。主站发送的广播请求不会返回任何响应。广播请求必须是写命令。所有设备必须接受写功能的广播。地址 0 保留用于标识广播交换。

图 3：单播模式 — 主站向某个从站发送请求，从站返回应答。
图 4：广播模式 — 主站向所有从站发送请求，无应答。

2.2 MODBUS 寻址规则

MODBUS 寻址空间包含 256 个不同地址。

地址范围	用途
0	广播地址
1 到 247	从站独立地址
248 到 255	保留

地址 0 保留为广播地址。所有从站节点必须识别广播地址。

MODBUS 主站节点没有特定地址，只有从站节点必须有地址。该地址在 MODBUS 串行总线上必须是唯一的。

2.3 MODBUS 帧描述

MODBUS 应用协议 [1] 定义了一个简单的协议数据单元（PDU），独立于底层通信层：

图 5：MODBUS 协议数据单元 — 由功能码（Function code）和数据（Data）组成。

MODBUS 协议映射到特定总线或网络时，在协议数据单元上引入一些附加字段。发起 MODBUS 事务的客户端构建 MODBUS PDU，然后添加字段以构建适当的通信 PDU。

图 6：串行线上的 MODBUS 帧
MODBUS SERIAL LINE PDU = [地址字段] [功能码] [数据] [CRC（或 LRC）]
其中包含 MODBUS PDU = [功能码] [数据]

- 在 MODBUS 串行线上，地址字段仅包含从站地址。如前所述，有效的从站节点地址范围为 0–247（十进制）。各个从站设备分配的地址范围为 1–247。主站通过在消息的地址字段中放置从站地址来寻址从站。当从站返回响应时，将其自身地址放入响应地址字段中，让主站知道是哪个从站在响应。
- 功能码指示服务器要执行的操作类型。功能码后面可以跟一个数据字段，包含请求和响应参数。
- 错误校验字段是对消息内容执行”冗余校验”计算的结果。根据所使用的传输模式（RTU 或 ASCII），使用两种计算方法。（参见 2.5 节”两种串行传输模式”）

2.4 主从状态图

MODBUS 数据链路层包含两个独立子层：

- 主从协议
- 传输模式（RTU 与 ASCII 模式）

以下各节描述了主站和从站的状态图，这些状态图独立于所使用的传输模式。RTU 和 ASCII 传输模式在后续章节中使用两个状态图进行规范。描述了帧的接收和发送。

状态图语法：
以下状态图按照 UML 标准符号绘制。符号简要回顾如下：
State_A —[触发事件 [守卫条件] / 动作]→ State_B
当处于”State_A”的系统发生”触发事件”时，仅当”守卫条件”为真时，系统进入”State_B”。然后执行动作”action”。

2.4.1 主站状态图

图 7：主站状态图

主站状态图包含以下状态和转换：

空闲（Idle）→ 等待应答（Waiting for reply）：向从站发送请求 / 启动响应超时计时器

等待应答 → 处理应答（Processing reply）：收到应答 [期望的从站] / 停止响应超时

处理应答 → 空闲：应答处理结束

等待应答 → 处理错误（Processing error）：帧错误 / 响应超时到期

处理错误 → 空闲：错误处理结束

空闲 → 等待周转延迟（Waiting turnaround delay）：以广播模式发送请求 / 启动周转延迟

等待周转延迟 → 空闲：周转延迟到期

等待应答 → 等待应答：收到应答 [非期望的从站]（保持响应超时运行）

关于上述状态图的一些说明：

- 状态”空闲”（Idle） = 无待处理请求。这是上电后的初始状态。只有在”空闲”状态下才能发送请求。发送请求后，主站离开”空闲”状态，不能同时发送第二个请求。
-

当单播请求发送给从站时，主站进入”等待应答”状态，并启动”响应超时”（Response Time-out）。它防止主站无限期停留在”等待应答”状态。响应超时值取决于应用。

- 收到应答后，主站在开始数据处理前检查应答。检查可能导致错误，例如来自非期望从站的应答或接收帧中的错误。如果收到来自非期望从站的应答，响应超时继续运行。如果检测到帧错误，可以执行重试。
- 如果未收到应答，响应超时到期并产生错误。然后主站回到”空闲”状态，允许重试请求。最大重试次数取决于主站设置。

关于广播请求和超时设置的补充说明：

- 当广播请求发送到串行总线时，从站不返回响应。但主站仍需遵守一个延迟，以允许任何从站在发送新请求前处理当前请求。此延迟称为”周转延迟”（Turnaround delay）。因此主站在回到”空闲”状态之前进入”等待周转延迟”状态，然后才能发送另一个请求。
- 在单播模式下，响应超时必须设置得足够长，以便任何从站都能处理请求并返回响应；在广播模式下，周转延迟必须足够长，以便任何从站都能处理请求并能够接收新请求。因此，周转延迟应短于响应超时。典型值：在 9600 bps 时响应超时为 1 秒到数秒；周转延迟为 100 ms 到 200 ms。
- 帧错误包括：1) 应用于每个字符的校验检查（Parity checking）；2) 应用于整个帧的冗余校验（Redundancy checking）。参见 §2.6”错误校验方法”了解更多说明。

状态图被故意简化。它没有考虑总线访问、消息组帧或传输错误后的重试等。有关帧传输的更多细节，请参见 2.5 节”两种串行传输模式”。

2.4.2 从站状态图

图 8：从站状态图

从站状态图包含以下状态和转换：

空闲（Idle）→ 检查请求（Checking request）：收到来自主站的请求

检查请求 → 处理所需操作（Processing required action）：检查通过（check OK）

处理所需操作 → 格式化正常应答（Formatting normal reply）：处理结束 [单播模式]

处理所需操作 → 空闲：处理结束 [广播模式]

格式化正常应答 → 空闲：正常应答已发送

处理所需操作 → 格式化错误应答（Formatting error reply）：处理中出错

格式化错误应答 → 空闲：错误应答已发送

检查请求 → 格式化错误应答：请求数据中的错误 → 发送错误应答

检查请求 → 空闲：帧检查错误，或帧不是发给本从站的

关于上述状态图的一些说明：

- 状态”空闲”（Idle）= 无待处理请求。这是上电后的初始状态。
- 收到请求后，从站在执行请求中要求的操作之前检查数据包。可能发生不同的错误：请求中的格式错误、无效操作等。如果发生错误，必须向主站发送一个应答。
- 一旦所需操作完成，单播消息要求必须格式化一个应答并发送给主站。
- 如果从站在接收到的帧中检测到错误，不会向主站返回任何响应。
- MODBUS 诊断计数器已定义，任何从站都应管理这些计数器以提供诊断信息。这些计数器可通过诊断 MODBUS 功能获取（参见附录 A 和 MODBUS 应用协议规范 [1]）。

2.4.3 主从通信时序图

图 9：主从通信场景时序图

下图展示了主从通信的 3 个典型场景的时序图：

交换 i-1：主站发送 REQUEST → 从站处理请求 → 从站返回 REPLY → 主站分析应答并准备下一次交换

交换 i：主站发送 BROADCAST REQUEST → 所有从站同时执行命令（无 REPLY）→ 周转延迟（Turnaround delay）

交换 i+1：主站发送 REQUEST → 从站无响应 → 响应超时（Response time out）错误 → 错误检测

备注：

- REQUEST、REPLY、BROADCAST 阶段的持续时间取决于通信特性（帧长度和吞吐量）。
- WAIT 和 TREATMENT 阶段的持续时间取决于从站应用所需的请求处理时间。

2.5 两种串行传输模式

定义了两种不同的串行传输模式：RTU 模式和 ASCII 模式。

它定义了在串行线上传输的消息字段的位内容。它决定了信息如何被打包到消息字段中以及如何解码。

MODBUS 串行线上所有设备的传输模式（和串口参数）必须相同。

虽然在某些特定应用中需要 ASCII 模式，但只有在每个设备具有相同传输模式时才能实现 MODBUS 设备之间的互操作性：所有设备必须实现 RTU 模式。ASCII 传输模式是可选的。

用户应将设备设置为所需的传输模式（RTU 或 ASCII）。默认设置必须为 RTU 模式。

2.5.1 RTU 传输模式

当设备使用 RTU（Remote Terminal Unit，远程终端单元）模式在 MODBUS 串行线上通信时，消息中的每个 8 位字节包含两个 4 位十六进制字符。此模式的主要优点是其更高的字符密度允许在相同波特率下比 ASCII 模式有更好的数据吞吐量。每条消息必须以连续的字符流传输。

RTU 模式中每个字节的格式（11 位）：

项目	说明
编码系统	8 位二进制
每字节的位数	1 个起始位 8 个数据位，最低有效位先发送 1 个校验位 1 个停止位

偶校验（Even parity）是必需的，其他模式（奇校验、无校验）也可使用。为确保与其他产品的最大兼容性，建议也支持无校验模式。默认校验模式必须为偶校验。

> 注意：使用无校验时需要 2 个停止位。

字符的串行传输方式：

每个字符或字节按以下顺序发送（从左到右）：最低有效位（LSB）... 最高有效位（MSB）

图 10：RTU 模式位序列（带校验检查）
Start | 1 2 3 4 5 6 7 | Par | Stop | 8

设备可通过配置接受偶校验、奇校验或无校验。如果实现无校验，则传输一个额外的停止位以将字符帧填充为完整的 11 位异步字符：

图 11：RTU 模式位序列（无校验的特殊情况）
Start | 1 2 3 4 5 6 7 | Stop | Stop | 8

帧校验字段：循环冗余校验（CRC，Cyclical Redundancy Checking）

帧描述：

图 12：RTU 消息帧

从站地址	功能码	数据	CRC
1 字节	1 字节	0 到 252 字节	2 字节

CRC 由 CRC Hi（高字节）和 CRC Low（低字节）组成。

- MODBUS RTU 帧的最大大小为 256 字节。

2.5.1.1 MODBUS 消息 RTU 组帧

MODBUS 消息由发送设备放入一个具有已知起点和终点的帧中。这使得接收新帧的设备能够从消息开头开始，并知道消息何时完成。必须检测不完整的消息并因此设置错误。

在 RTU 模式中，消息帧之间由至少 3.5 个字符时间的静默间隔分隔。在以下章节中，此时间间隔称为 t3.5。

图 13：RTU 消息帧

帧结构：⩾3.5 char [起始] [地址 8 bits] [功能码 8 bits] [数据 N x 8 bits] [CRC 16 bits] [结束] ⩾3.5 char

整个消息帧必须作为连续的字符流传输。

如果两个字符之间出现超过 1.5 个字符时间的静默间隔，则消息帧被声明为不完整，接收方应将其丢弃。

> 注意：RTU 接收驱动的实现可能意味着由于 t1.5 和 t3.5 定时器而需要管理大量中断。在高通信波特率下，这会导致较重的 CPU 负载。因此，当波特率等于或低于 19200 Bps 时，这两个定时器必须严格遵守。对于高于 19200 Bps 的波特率，应使用两个定时器的固定值：建议字符间超时（t1.5）使用 750µs 的值，帧间延迟（t3.5）使用 1.750ms 的值。

图 14：RTU 传输模式状态图

下图描述了 RTU 传输模式状态图。”主站”和”从站”的观点在同一图中表达。

状态图包含以下状态和转换：

初始状态（Initial State）→ 空闲（Idle）：t3.5 超时到期（确保帧间延迟）

空闲 → 接收（Reception）：收到第一个字符 / 初始化并启动 t1.5、t3.5

接收 → 接收：收到字符 / 初始化并启动 t1.5、t3.5

接收 → 控制与等待（Control and Waiting）：t3.5 到期 → 控制帧（CRC、校验、从站地址）→ 设置标志 frame OK 或 NOK

控制与等待 → 空闲：t3.5 到期（如果帧 OK → 处理帧；如果帧 NOK → 删除整个帧）

空闲 → 发送（Emission）：发送请求 → 发送字符 → [如果是最后发送的字符] / 初始化并启动 t3.5 → 空闲

图例：t1.5、t3.5 为定时器；t3.5 为 3.5 个字符时间；t1.5 为 1.5 个字符时间。

关于上述状态图的说明：

- 从”初始状态”到”空闲”状态的转换需要 t3.5 超时到期：这确保了帧间延迟。
- “空闲”状态是既无发送也无接收活动时的正常状态。
- 在 RTU 模式中，当无传输活动的时间间隔等于至少 3.5 个字符时，通信链路被声明为”空闲”状态。
- 当链路处于空闲状态时，在链路上检测到的每个传输字符被标识为帧的开始。链路进入”活动”状态。然后，当时间间隔 t3.5 后链路上不再有字符传输时，标识帧结束。
- 检测到帧结束后，完成 CRC 计算和校验。然后分析地址字段以确定帧是否发往本设备。如果不是，则丢弃帧。为了减少接收处理时间，地址字段可在收到后立即分析，无需等待帧结束。在这种情况下，只有当帧发往本从站（包括广播帧）时才计算和校验 CRC。

2.5.1.2 CRC 校验

RTU 模式包含一个基于对消息内容执行循环冗余校验（CRC）方法的错误校验字段。

CRC 字段校验整个消息的内容。它独立于消息各个字符所使用的任何校验检查方法。

CRC 字段包含一个 16 位值，实现为两个 8 位字节。CRC 字段作为消息中的最后一个字段附加到消息中。附加时，先附加该字段的低字节，然后附加高字节。CRC 高字节是消息中发送的最后一个字节。

CRC 值由发送设备计算，发送设备将 CRC 附加到消息中。接收设备在接收消息期间重新计算 CRC，并将计算值与在 CRC 字段中接收到的实际值进行比较。如果两个值不相等，则产生错误。

CRC 计算首先将一个 16 位寄存器预装载为全 1。然后开始将消息的连续 8 位字节应用到寄存器当前内容的过程。只有每个字符中的 8 位数据用于生成 CRC。起始位、停止位和校验位不参与 CRC 计算。

在 CRC 生成期间，每个 8 位字符与寄存器内容进行异或（XOR）运算。然后结果向最低有效位（LSB）方向移位，最高有效位（MSB）位置填零。提取并检查 LSB。如果 LSB 为 1，则寄存器与预设的固定值进行异或。如果 LSB 为 0，则不进行异或。

此过程重复直到执行了 8 次移位。在最后一次（第 8 次）移位后，下一个 8 位字节与寄存器当前值进行异或，然后按上述方式重复 8 次移位过程。消息的所有字节都应用完毕后，寄存器的最终内容即为 CRC 值。当 CRC 附加到消息时，先附加低字节，然后附加高字节。CRC 生成的详细示例见附录 B。

2.5.2 ASCII 传输模式

当设备设置为使用 ASCII（American Standard Code for Information Interchange，美国信息交换标准代码）模式在 MODBUS 串行线上通信时，消息中的每个 8 位字节作为两个 ASCII 字符发送。当物理通信链路或设备能力不允许满足 RTU 模式关于定时器管理的要求时，使用此模式。

> 注意：此模式比 RTU 模式效率低，因为每个字节需要两个字符。示例：字节 0X5B 被编码为两个字符：0x35 和 0x42（0x35 = "5"，0x42 = "B" 的 ASCII 编码）。

ASCII 模式中每个字节的格式（10 位）：

项目	说明
编码系统	十六进制，ASCII 字符 0-9、A-F。每个 ASCII 字符中包含 4 位数据。
每字节的位数	1 个起始位 7 个数据位，最低有效位先发送 1 个校验位 1 个停止位

偶校验是必需的，其他模式（奇校验、无校验）也可使用。为确保与其他产品的最大兼容性，建议也支持无校验模式。默认校验模式必须为偶校验。

> 注意：使用无校验时需要 2 个停止位。

字符的串行传输方式：

每个字符或字节按以下顺序发送（从左到右）：最低有效位（LSB）... 最高有效位（MSB）

图 15：ASCII 模式位序列（带校验检查）
Start | 1 2 3 4 5 6 7 | Par | Stop

设备可通过配置接受偶校验、奇校验或无校验。如果实现无校验，则传输一个额外的停止位以填充字符帧：

图 16：ASCII 模式位序列（无校验的特殊情况）
Start | 1 2 3 4 5 6 7 | Stop | Stop

帧校验字段：纵向冗余校验（LRC，Longitudinal Redundancy Checking）

2.5.2.1 MODBUS 消息 ASCII 组帧

MODBUS 消息由发送设备放入一个具有已知起点和终点的帧中。这使得接收新帧的设备能够从消息开头开始，并知道消息何时完成。必须检测不完整的消息并因此设置错误。

消息帧的地址字段包含两个字符。

在 ASCII 模式中，消息由特定字符作为帧起始和帧结束进行分隔。消息必须以冒号（:）字符（ASCII 3A hex）开始，以回车-换行（CRLF）对（ASCII 0D 和 0A hex）结束。

> 注意：LF 字符可使用特定的 MODBUS 应用命令进行更改（参见 MODBUS 应用协议规范）。

所有其他字段传输的允许字符为十六进制 0-9、A-F（ASCII 编码）。设备持续监控总线上的冒号字符。收到此字符后，每个设备解码下一个字符，直到检测到帧结束。

消息中字符之间可间隔最多 1 秒。除非用户配置了更长的超时，大于 1 秒的间隔意味着发生了错误。某些广域网应用可能需要 4 到 5 秒范围的超时。典型消息帧如下所示：

图 17：ASCII 消息帧

起始	地址	功能码	数据	LRC	结束
:	2 chars	2 chars	0 到 2x252 chars	2 chars	CR,LF

> 注意：每个数据字节需要两个字符进行编码。因此，为确保 ASCII 模式和 RTU 模式在 MODBUS 应用层面的兼容性，ASCII 数据字段的最大数据大小（2x252）是 RTU 数据字段最大数据大小（252）的两倍。因此，MODBUS ASCII 帧的最大大小为 513 个字符。

图 18：ASCII 传输模式状态图

ASCII 组帧要求综合在以下状态图中。”主站”和”从站”的观点在同一图中表达。

状态图包含以下状态和转换：

空闲（Idle）→ 接收（Reception）：收到":"字符 / 清空接收缓冲区

接收 → 等待帧结束（Waiting "End of Frame"）：收到字符 / 将字符连接到接收缓冲区

等待帧结束 → 等待帧结束：收到字符 / 继续连接字符

等待帧结束 → 控制与等待：收到"CR"字符 → 收到"LF"字符 / 控制帧（LRC、校验、从站地址）

控制与等待 → 空闲：如果帧 OK → 处理帧；如果帧 NOK → 删除整个帧

空闲 → 发送（Emission）：发送请求 → 发送":"→ 发送所有字符 → 发送"CR"→ 发送"LF"→ 空闲

关于上述状态图的说明：

- “空闲”状态是既无发送也无接收活动时的正常状态。
- 每次收到":"字符意味着新消息的开始。如果在收到此字符时正在接收消息，则当前消息被声明为不完整并被丢弃。然后分配新的接收缓冲区。
- 检测到帧结束后，完成 LRC 计算和校验。然后分析地址字段以确定帧是否发往本设备。如果不是，则丢弃帧。为了减少接收处理时间，地址字段可在收到后立即分析，无需等待

帧结束。

2.5.2.2 LRC 校验

在 ASCII 模式中，消息包含一个基于对消息内容执行纵向冗余校验（LRC）计算的错误校验字段，计算不包括起始冒号和终止 CRLF 对字符。它独立于消息各个字符所使用的任何校验检查方法。

LRC 字段为一个字节，包含一个 8 位二进制值。LRC 值由发送设备计算，发送设备将 LRC 附加到消息中。接收设备在接收消息期间计算 LRC，并将计算值与在 LRC 字段中接收到的实际值进行比较。如果两个值不相等，则产生错误。

LRC 的计算方法是将消息中连续的 8 位字节相加，丢弃任何进位，然后对结果取二补码。它在消息的字节上执行，在将每个字节编码为与每个半字节十六进制表示对应的两个 ASCII 字符之前进行。计算不包括消息起始的冒号字符，也不包括消息结束的 CRLF 对。结果 LRC 被 ASCII 编码为两个字节，放置在 ASCII 模式帧中 CRLF 之前。

LRC 生成的详细示例见附录 B。

2.6 错误校验方法

标准 MODBUS 串行线的安全性基于两种错误校验：

- 校验检查（Parity checking，偶校验或奇校验）应应用于每个字符。
- 帧校验（Frame checking，LRC 或 CRC）必须应用于整个消息。

字符校验和消息帧校验都在发送设备（主站或从站）中生成，并在传输前应用到消息内容上。接收设备（从站或主站）在接收期间检查每个字符和整个消息帧。

用户配置主站等待预定的超时间隔（响应超时，Response time-out）后中止事务。此间隔设置得足够长，以便任何从站都能正常响应（单播请求）。如果从站检测到传输错误，消息将不会被处理。从站不会向主站构造响应。因此超时将到期并允许主站程序处理错误。注意，发往不存在的从站设备的消息也会导致超时。

2.6.1 校验检查

用户可将设备配置为偶校验（必需）、奇校验或无校验（推荐）。这将决定每个字符中校验位的设置方式。

如果指定了偶校验或奇校验，则计算每个字符数据部分中 1 的位数（ASCII 模式为 7 个数据位，RTU 模式为 8 个数据位）。然后校验位设置为 0 或 1，使 1 的总位数为偶数或奇数。

例如，RTU 字符帧中包含以下 8 个数据位：

```
1100 0101
```

帧中 1 的总位数为 4。如果使用偶校验，帧的校验位将为 0，使 1 的总位数仍为偶数（4）。如果使用奇校验，校验位将为 1，使 1 的总位数为奇数（5）。

发送消息时，校验位被计算并应用到每个字符的帧上。接收设备计算 1 的位数，如果与设备配置的校验方式不同则设置错误（MODBUS 串行线上的所有设备必须配置为使用相同的校验检查方法）。

注意，校验检查只能在传输期间字符帧中拾取或丢失奇数个位时检测到错误。例如，如果使用奇校验检查，并且包含 3 个 1 位的字符丢失了两个 1 位，结果仍然是 1 的奇数计数。

如果指定了无校验，则不传输校验位，也不进行校验检查。传输一个额外的停止位以填充字符帧。

2.6.2 帧校验

根据传输模式（RTU 或 ASCII），使用两种帧校验：

- 在 RTU 模式中，消息包含一个基于循环冗余校验（CRC）方法的错误校验字段。CRC 字段校验整个消息的内容。它独立于消息各个字符所使用的任何校验检查方法。

- 在 ASCII 模式中，消息包含一个基于纵向冗余校验（LRC）方法的错误校验字段。LRC 字段校验消息的内容，不包括起始冒号和结束 CRLF 对。它独立于消息各个字符所使用的任何校验检查方法。

关于错误校验方法的详细信息包含在前面的章节中。

3 物理层

3.1 前言

新的串行线 MODBUS 解决方案应实现符合 EIA/TIA-485 标准（也称为 RS485 标准）的电气接口。该标准允许在”双线配置”中实现点对点和多点系统。此外，某些设备可实现”四线”RS485 接口。

设备也可实现 RS232 接口。

在此类 MODBUS 系统中，主站设备和一个或多个从站设备在无源串行线上通信。

在标准 MODBUS 系统中，所有设备并联连接在由 3 根导线组成的干线上。其中两根导线（”双线”配置）构成一个平衡双绞线对，在其上双向传输数据，典型比特率为 9600 bps。每个设备可通过以下方式连接（见图 19）：

- 直接连接到干线上，形成菊花链（daisy-chain）
- 通过无源抽头（Passive Tap）和分支电缆连接
- 通过有源抽头（Active Tap）和专用电缆连接

设备上可使用螺钉端子、RJ45 或 D 型 9 针连接器来连接电缆（参见”机械接口”章节）。

3.2 数据信令速率

9600 bps 和 19.2 Kbps 是必需的，19.2 是要求的默认值。

可选实现其他波特率：1200、2400、4800、... 38400 bps、56 Kbps、115 Kbps、...

每个实现的波特率在发送情况下必须优于 1% 的精度，在接收情况下必须接受 2% 的误差。

3.3 电气接口

3.3.1 多点串行总线基础设施

图 19：串行总线基础设施

图 19 给出了 MODBUS 多点串行线系统中串行总线基础设施的总体概览。图中包含：干线（Trunk）、线路终端（LT）、无源抽头（Passive TAP）、有源抽头（Active Tap）、以及主站和多个从站设备。

多点 MODBUS 串行线总线由一条主干电缆（干线）和可能的分支电缆组成。干线电缆两端需要线路终端（LT）进行阻抗匹配（参见”双线 MODBUS 定义”和”可选四线 MODBUS 定义”了解详情）。

如图 19 所示，不同的实现可在同一 MODBUS 串行线系统中运行：

- 设备集成通信收发器，通过无源抽头和分支电缆连接到干线（从站 1 和主站的情况）；
- 设备不集成通信收发器，通过有源抽头和分支电缆连接到干线（有源抽头集成收发器）（从站 2 的情况）；
- 设备以菊花链方式直接连接到干线电缆（从站 n 的情况）。

采用以下约定：

- 与干线的接口命名为 ITr（干线接口，Trunk Interface）
- 设备与无源抽头之间的接口命名为 IDv（分支接口，Derivation Interface）
- 设备与有源抽头之间的接口命名为 AUI（附加单元接口，Attachment Unit Interface）

备注：

- 1. 在某些情况下，抽头可直接连接到设备的 IDv 插座或 AUI 插座，无需使用分支电缆。
- 2. 一个抽头可以有多个 IDv 插座以连接多个设备。当抽头为无源时，称为分配器（Distributor）。
- 3. 使用有源抽头时，抽头的电源可通过其 AUI 或 ITr 接口提供。

ITr 和 IDv 接口在以下章节中描述（参见”双线 MODBUS 定义”和”四线 MODBUS 定义”）。

3.3.2 双线 MODBUS 定义

MODBUS 串行线解决方案应实现符合 EIA/TIA-485 标准的”双线”电气接口。在此类 2W 总线上，任何时候只有一个驱动器有权发送。

实际上，还必须有第三根导线互连总线上所有设备：公共端（Common）。

图 20：通用双线拓扑 — 主站和从站通过平衡双绞线对（D0-D1）连接，带有上拉（Pull Up）、下拉（Pull Down）和线路终端（LT）。

2W-MODBUS 电路定义

ITr 上的必需电路	IDv 上的必需电路	设备方向	设备上必需	EIA/TIA-485 名称	说明
D1	D1	I/O	X	B/B'	收发器端子 1，V1 电压（V1 > V0 为二进制 1 [OFF] 状态）
D0	D0	I/O	X	A/A'	收发器端子 0，V0 电压（V0 > V1 为二进制 0 [ON] 状态）
Common	Common	--	X	C/C'	信号和可选电源公共端

备注：

- 关于线路终端（LT）、上拉和下拉电阻，请参见”多点系统要求”章节。

- D0、D1 和 Common 电路名称必须在设备相关文档中使用（用户指南、布线指南等），以促进互操作性。
- 可选的电气接口可添加，例如：
- 电源供电：5..24 V D.C.
- 端口模式控制：PMC 电路（TTL 兼容）。需要时，端口模式可通过此外部电路和/或其他方式（例如设备上的开关）控制。在前一种情况下，PMC 开路将请求 2W-MODBUS 模式，PMC 上的低电平将根据实现将端口切换为 4W-MODBUS 或 RS232-MODBUS 模式。

3.3.3 可选四线 MODBUS 定义

可选地，此类 MODBUS 设备还允许实现单向数据的双对总线（4 线）。主站对（RXD1-RXD0）上的数据仅被从站接收；从站对（TXD1-TXD0）上的数据仅被主站接收。

实际上，还必须有第五根导线互连 4W 总线上所有设备：公共端（Common）。

与 2W-MODBUS 相同，任何时候只有一个驱动器有权发送。

此类设备必须为每个平衡对实现符合 EIA/TIA-485 的驱动器和收发器。

（有时此方案被称为”RS422”，这是不正确的：RS422
标准不支持一个平衡对上有多个驱动器。）

图 21：通用四线拓扑 — 主站对和从站对，带有上拉、下拉和线路终端。

可选 4W-MODBUS 电路定义

ITr 上的必需电路	IDv 上的必需电路	设备方向	设备上必需	EIA/TIA-485 名称	IDv 说明
TXD1	TXD1	Out	X	B	发生器端子 1，Vb 电压（Vb > Va 为二进制 1 [OFF] 状态）
TXD0	TXD0	Out	X	A	发生器端子 0，Va 电压（Va > Vb 为二进制 0 [ON] 状态）
RXD1	RXD1	In	(1)	B'	接收器端子 1，Vb' 电压（Vb' > Va' 为二进制 1 [OFF] 状态）
RXD0	RXD0	In	(1)	A'	接收器端子 0，Va' 电压（Va' > Vb' 为二进制 0 [ON] 状态）
Common	Common	--	X	C/C'	信号和可选电源公共端

备注：

- 关于线路终端（LT）、上拉和下拉电阻，请参见”多点系统要求”章节。

- 电路 (1) 仅在实现 4W-MODBUS 选项时才需要。
- 5 个必需电路的名称必须在设备和抽头相关文档中使用（用户指南、布线指南等），以促进互操作性。
- 可选的电气接口可添加，例如：
- 电源供电：5..24 V D.C.
- PMC 电路：参见上述（2W-MODBUS 电路定义中）关于此可选电路的备注。

3.3.3.1 四线布线系统重要事项

在此类 4W-MODBUS 中，主站设备和从站设备具有包含相同 5 个必需电路的 IDv 接口。

由于主站需要：

- 从从站接收从站对（TXD1-TXD0）上的数据，
- 在主站对（RXD1-RXD0，由从站接收）上发送，

因此 4W 布线系统必须在主站的 ITr 和 IDv 之间交叉两对总线：

主站 IDv 上的信号名称	类型	EIA/TIA-485 名称	ITr 上的电路
RXD1	In	B'	TXD1（从站对）
RXD0	In	A'	TXD0（从站对）
TXD1	Out	B	RXD1（主站对）
TXD0	Out	A	RXD0（主站对）
Common	--	C/C'	Common

此交叉可通过交叉电缆实现，但在 2 线系统中连接此类交叉电缆可能造成损坏。要连接 4W 主站设备（具有 MODBUS 连接器），更好的解决方案是使用包含交叉功能的抽头。

3.3.3.2 四线与双线布线的兼容性

为将实现双线物理接口的设备连接到已有的四线系统，可按如下方式修改四线布线系统：

- TxD0 信号应与 RxD0 信号连接，变为 D0 信号
- TxD1 信号应与 RxD1 信号连接，变为 D1 信号
- 上拉、下拉和线路终端电阻应重新排列以正确适配 D0、D1 信号。

图 22：将四线布线系统改为双线布线系统 — 从站 2 和 3 使用双线接口可与使用四线接口的主站和从站 1 一起工作。

为将实现四线物理接口的设备连接到已有的双线系统，可按如下方式排列新接入设备的四线接口：

在每个四线设备接口上：

- TxD0 信号应与 RxD0 信号连接，然后连接到干线的 D0 信号；
- TxD1 信号应与 RxD1 信号连接，然后连接到干线的 D1 信号。

图 23：将四线接口设备连接到双线布线系统 — 从站 2 和 3

使用四线接口可与使用双线接口的主站和从站 1 一起工作。

3.3.4 RS232-MODBUS 定义

某些设备可在 DCE 和 DTE 之间实现 RS232 接口。

可选 RS232-MODBUS 电路定义

信号	DCE 方向	DCE 上必需	DTE 上必需	说明
Common	--	X	X	信号公共端
CTS	In			Clear to Send（允许发送）
DCD	--			Data Carrier Detected（数据载波检测，从 DCE 到 DTE）
DSR	In			Data Set Ready（数据集就绪）
DTR	Out			Data Terminal Ready（数据终端就绪）
RTS	Out			Request to Send（请求发送）
RXD	In	X	X	Received Data（接收数据）
TXD	Out	X	X	Transmitted Data（发送数据）

备注：

- 标有"X"的信号仅在实现 RS232-MODBUS 选项时才需要。
- 信号符合 EIA/TIA-232 标准。
- 每个 TXD 必须与对方设备的 RXD 连接；
- RTS 可与对方设备的 CTS 连接；
- DTR 可与对方设备的 DSR 连接。
- 可选的电气接口可添加，例如：

- 电源供电：5..24 V D.C.
- PMC 电路：参见上述（2W-MODBUS 电路定义中）关于此可选电路的备注。

3.3.5 RS232-MODBUS 要求

此可选的串行线 MODBUS 系统应仅用于短距离（通常小于 20m）点对点互连。

然后，必须遵守 EIA/TIA-232 标准：

电路定义，

对地最大导线电容（2500 pF，即 100 pF/m 电缆时 25 m）。

关于屏蔽层以及使用 Category 5 电缆的可能性，请参见”线缆”章节。

设备文档必须指明：

设备应被视为 DCE 还是 DTE，

可选电路（如有）应如何工作。

3.4 多点系统要求

对于任何 EIA/TIA-485 多点系统，无论是 2 线还是 4 线配置，以下要求均适用。

3.4.1 不使用中继器的最大设备数

在任何无中继器的 RS485-MODBUS 系统上，始终允许 32 台设备。

根据以下因素：

- 所有可能的地址
- 设备使用的 RS485 单元负载数量
- 以及所需的线路偏置

RS485 系统可实现更多设备。某些设备允许在无中继器的情况下实现超过 32 台设备的 RS485-MODBUS 串行线。在这种情况下，这些 MODBUS 设备必须在文档中说明无中继器时允许的设备数量。

在两个重负载 RS485-MODBUS 之间使用中继器（repeater）也是可能的。

3.4.2 拓扑

无中继器的 RS485-MODBUS 配置有一条干线电缆，设备沿干线直接连接（菊花链）或通过短分支电缆连接。干线电缆（也称为”总线”，Bus）可以很长（见下文）。其两端必须连接线路终端。

在多个 RS485-MODBUS 之间使用中继器也是可能的。

3.4.3 长度

干线电缆的端到端长度必须受限。最大长度取决于波特率、电缆（线规、电容或特性阻抗）、菊花链上的负载数量和网络配置（2 线或 4 线）。

对于最大 9600 波特率和 AWG26（或更粗）线规，最大长度为 1000m。在图 22 所示的特定情况下（四线布线用作双线布线系统），最大长度必须除以 2。

分支必须短，绝不超过 20m。如果使用带 n 个分支的多端口抽头，每个分支必须遵守 40m/n 的最大长度。

3.4.4 接地安排

“Common”电路（信号和可选电源公共端）必须直接连接到保护地，最好在整个总线上仅一点接地。通常此点选择在主站设备或其抽头上。

3.4.5 线路终端

传输线中的反射是行波沿线传播时遇到的阻抗不连续性的结果。为最小化 RS485 电缆端部的反射，要求在总线的两个端部附近各放置一个线路终端（LT）。

重要的是线路两端都要终端匹配，因为传播是双向的，但不允许在一个无源 D0-D1 平衡对上放置超过 2 个 LT。切勿在分支电缆上放置任何 LT。

每个线路终端必须连接在平衡线的两个导体 D0 和 D1 之间。

线路终端可以是 150 欧姆（0.5 W）的电阻。

当需要实现对的偏置时，串联电容（1 nF，最小 10 V）与 120 欧姆（0.25 W）电阻的组合是更好的选择（见下文）。在 4W 系统中，每对必须在总线的每一端进行终端匹配。

在 RS232 互连中，不应布线终端匹配。

3.4.6 线路偏置

当 RS-485 平衡对上无数据活动时，线路不被驱动，因此容易受到外部噪声或干扰的影响。为确保其接收器在没有数据信号时保持恒定状态，某些设备需要对网络进行偏置。

每个 MODBUS 设备必须在文档中说明：

- 设备是否需要线路偏置
- 设备是否实现或可以实现此类线路偏置

如果一个或多个设备需要偏置，则必须在 RS-485 平衡对上连接一对电阻：

- 在 D1 电路上连接上拉电阻到 5V 电压
- 在 D0 电路上连接下拉电阻到公共端电路

这些电阻的值必须在 450 欧姆到 650 欧姆之间。650 欧姆电阻值可允许串行线总线上更多设备。

在这种情况下，必须在整个串行总线的一个位置实现对的偏置。通常此点选择在主站设备或其抽头上。其他设备不得实现任何偏置。

在此类 MODBUS 串行线上允许的最大设备数比无偏置的 MODBUS 减少 4 个。

3.5 机械接口

螺钉端子可用于 IDv 和 ITr 连接。必须向用户提供每个信号确切位置的所有信息，名称应与前面”电气接口”章节一致。

如果在设备上使用 RJ45（或 mini-DIN 或 D 型）连接器作为 MODBUS 机械接口，必须选择屏蔽的母连接器。然后电缆端必须具有屏蔽的公连接器。

3.5.1 2W-MODBUS 连接器引脚定义

图 24：RJ45 连接器上的 2W-MODBUS（必需引脚定义）— 设备侧母连接器

图 25：D 型 9 针连接器

也可使用螺钉型连接器。

如果标准 MODBUS 设备使用 RJ45 或 9 针 D 型连接器，则每个已实现电路必须遵守以下引脚定义。

RJ45 引脚	D9-shell 引脚	要求级别	IDv 电路	ITr 电路	EIA/TIA-485 名称	IDv 说明
3	3	可选	PMC	--	--	端口模式控制
4	5	必需	D1	D1	B/B'	收发器端子 1，V1 电压（V1 > V0 为二进制 1 [OFF] 状态）
5	9	必需	D0	D0	A/A'	收发器端子 0，V0 电压（V0 > V1 为二进制 0 [ON] 状态）
7	2	推荐	VP	--	--	正 5...24 V D.C. 电源
8	1	必需	Common	Common	C/C'	信号和电源公共端

3.5.2 可选 4W-MODBUS 连接器引脚定义

图 26：RJ45 连接器上的 4W-MODBUS（必需引脚定义）— 设备侧母连接器

图 27：D 型 9 针连接器

也可使用螺钉型连接器。

如果 4W-MODBUS 设备使用 RJ45 或 9 针 D 型连接器，则每个已实现电路必须遵守以下引脚定义。

RJ45 引脚	D9-shell 引脚	要求级别	IDv 信号	ITr 信号	EIA/TIA-485 名称	IDv 说明
1	8	必需	RXD0	RXD0	A'	接收器端子 0，Va' 电压（Va' > Vb' 为二进制 0 [ON] 状态）
2	4	必需	RXD1	RXD1	B'	接收器端子 1，Vb' 电压（Vb' > Va' 为二进制 1 [OFF] 状态）
3	3	可选	PMC	--	--	端口模式控制
4	5	必需	TXD1	TXD1	B	发生器端子 1，Vb 电压（Vb > Va 为二进制 1 [OFF] 状态）
5	9	必需	TXD0	TXD0	A	发生器端子 0，Va 电压（Va > Vb 为二进制 0 [ON] 状态）
7	2	推荐	VP	--	--	正 5...24 V DC 电源
8	1	必需	Common	Common	C/C'	信号和电源公共端

注意：当同一端口上同时实现 2 线和 4 线配置时，必须使用 4W 标注。

3.5.3 可选 RS232-MODBUS 的 RJ45 和 9 针 D 型引脚定义

如果 RS232-MODBUS 设备使用 RJ45 或 9 针 D 型连接器，则每个已实现电路必须遵守以下引脚定义。

RJ45 引脚	D9-shell 引脚	要求级别	名称	说明	RS232 来源	DTE 要求级别	DTE RJ45	DTE D9
1	2	必需	TXD	发送数据	DTE	必需	2	3
2	3	必需	RXD	接收数据	DCE	必需	1	2
3	7	可选	CTS	允许发送	DCE	可选	6	8
6	8	可选	RTS	请求发送	DTE	可选	3	7

RJ45 引脚	D9-shell 引脚	要求级别	名称	说明	RS232 来源	DTE 要求级别	DTE RJ45	DTE D9
8	5	必需	Common	信号公共端	--	必需	8	5

重要注意：某些 DCE 引脚定义与同名的 DTE 引脚定义交叉：在 DTE（例如 PC）和 DCE（例如 PLC）之间必须使用直通引脚到引脚的电缆（无任何交叉）。

3.6 线缆

MODBUS 串行线电缆必须屏蔽。在每根电缆的一端，其屏蔽层必须连接到保护地。如果该端使用连接器，则连接器外壳连接到电缆屏蔽层。

RS485-MODBUS 必须使用一个平衡对（用于 D0-D1）和第三根导线（用于 Common）。此外，在 4W-MODBUS 系统中必须使用第二个平衡对（用于 RXD0-RXD1）。

如果使用带连接器的 4 对 Category 5 电缆，请在用户指南中提醒用户："在 2 线 MODBUS 系统中连接交叉电缆可能造成损坏"。

为最小化布线错误，推荐 RS485-MODBUS 电缆中的导线使用颜色编码：

信号名称	推荐颜色
D1 / TXD1	黄色（yellow）
D0 / TXD0	棕色（brown）
Common	灰色（grey）
4W（可选）RXD0	白色（white）
4W（可选）RXD1	蓝色（blue）

图 28：RS485-MODBUS 导线颜色编码

注意：Category 5 电缆使用其他颜色。

对于 RS485-MODBUS，必须选择足够粗的线规以允许最大长度（1000m）。AWG 24 对 MODBUS 数据始终足够。

Category 5 电缆可用于 RS485-MODBUS，最大长度为 600m。

对于 RS485 系统中使用的平衡对，可能优选特性阻抗值大于 100 欧姆的电缆，特别是在 19200 及更高波特率时。

3.7 视觉诊断

对于视觉诊断，通信状态和设备状态必须通过 LED 指示：

LED	要求级别	状态	推荐颜色
通信	必需	在帧接收或发送期间点亮。（2 个 LED 分别用于帧接收和帧发送，或 1 个 LED 兼顾两者。）	黄色
错误	推荐	常亮：内部故障；闪烁：其他故障（通信故障或配置错误）	红色
设备状态	可选	常亮：设备已上电	绿色

4 安装与文档

4.1 安装

产品供应商应注意向 MODBUS 系统或 MODBUS 设备的用户提供所有有用信息，以防止他们在布线或使用布线配件时出现错误：

- 某些其他现场总线，例如 CANopen，使用相同类型的连接器（D 型、RJ45 等）。
- 正在研究以太网在同一平衡对电缆上供电的方案。
- 某些产品在 I/O 电路中使用相同类型的连接器（D 型、RJ45 等）。

在这些连接器上，大多数情况下没有防呆装置（极化卡扣或其他实现）。

4.2 用户指南

任何 MODBUS 设备或布线系统组件的用户指南必须以非穷尽方式包含一种或两种类型的信息：

4.2.1 对于任何 MODBUS 产品

以下信息应被记录：

- 所有已实现的请求。
- 操作模式。
- 视觉诊断。
- 可访问的寄存器和支持的功能码。
- 安装规则。
- 以下章节中要求的信息也应被记录：

”双线 MODBUS 定义”（提及必需电路）；

”可选四线 MODBUS 定义”（提及必需电路）；

”线路偏置”（提及可能的需求或实现）；

”线缆”（特别注意交叉电缆）。

- 关于设备地址的特定指示，应以重要警告的形式书写：

> “在设备寻址过程中，确保没有两台设备具有相同地址是非常重要的。在这种情况下，整个串行总线可能出现异常行为，主站将无法与总线上所有存在的从站通信。”

- 强烈推荐包含一个”入门指南”（Getting Started）章节，其中记录典型应用示例的描述，以便于快速上手。

4.2.2 对于实现了选项的 MODBUS 产品

不同的可选参数必须清楚详细说明：

可选串行传输模式；

可选校验检查；

可选波特率；

可选电路：电源供电、端口配置；

可选接口；

无中继器时的最大设备数（如果大于 32）。

5 实现类

MODBUS 串行线上的每个设备必须遵守同一实现类的所有强制性要求。

以下参数用于分类 MODBUS 串行线设备：

- 寻址
- 广播
- 传输模式
- 波特率
- 字符格式
- 电气接口参数

提出了两种实现类：基本类（Basic）和常规类（Regular）。

常规类必须提供配置能力。

参数	BASIC	REGULAR	默认值
寻址	从站：可配置地址 1 到 247 主站：能够寻址 1 到 247 的从站	同 BASIC	--
广播	是	是	--
波特率	9600（也推荐 19200）	9600、19200 + 额外可配置波特率	19200（如已实现，否则 9600）
校验	EVEN	EVEN + 可配置 NO 和 ODD 校验	EVEN
模式	RTU	RTU + ASCII	RTU
电气接口	RS485 2W 布线或 RS232	RS485 2W 布线（4W 布线作为附加选项）或 RS232	RS485 2W 布线
连接器类型	RJ45（推荐）	--	--

6 附录

6.1 附录 A - 串行线诊断计数器管理

6.1.1 一般描述

MODBUS 串行线定义了一系列诊断计数器，用于性能和错误管理。

这些计数器可通过 MODBUS 应用协议及其诊断功能（功能码 08）访问。每个计数器可通过绑定到计数器编号的子功能码获取。所有计数器可使用子功能码 0x0A 清除。诊断功能的格式在 MODBUS 应用协议规范中描述。以下是串行线设备支持的诊断和关联子功能码列表。

子功能码 Hex	子功能码 Dec	计数器编号	计数器名称	说明
0x0B	1	1	Return Bus Message Count	远程设备自上次重启、清除计数器操作或上电以来在通信系统上检测到的消息数量。CRC 错误的消息不计入。
0x0C	2	2	Return Bus Communication Error Count	远程设备自上次重启、清除计数器操作或上电以来遇到的 CRC 错误数量。如果检测到字符级错误（溢出、校验错误），或消息长度 < 3 字节，接收设备无法计算 CRC。在这些情况下，此计数器也会递增。
0x0D	3	3	Return Slave Exception Error Count	远程设备自上次重启、清除计数器操作或上电以来检测到的 MODBUS 异常错误数量。它还包括在广播消息中检测到的错误，即使在这种情况下不返回异常消息。异常错误在”MODBUS 应用协议规范”文档中描述和列出。
0x0E	4	4	Return Slave Message Count	远程设备自上次重启、清除计数器操作或上电以来处理的消息数量（包括广播消息），这些消息地址指向该远程设备。
0x0F	5	5	Return Slave No Response Count	远程设备自上次重启、清除计数器操作或上电以来接收到的、未返回任何响应（既非正常响应也非异常响应）的消息数量。此计数器也计算接收到的广播消息数量。

子功能码 Hex	子功能码 Dec	计数器编号	计数器名称	说明
0x10	6	6	Return Slave NAK Count	远程设备自上次重启、清除计数器操作或上电以来接收到的、返回否定确认（NAK）异常响应的消息数量。异常响应在”MODBUS 应用协议规范”文档中描述和列出。
0x11	7	7	Return Slave Busy Count	远程设备自上次重启、清除计数器操作或上电以来接收到的、返回从站设备忙（Slave Device Busy）异常响应的消息数量。异常响应在”MODBUS 应用协议规范”文档中描述和列出。
0x12	8	8	Return Bus Character Overrun Count	远程设备自上次重启、清除计数器操作或上电以来接收到的、由于字符溢出状况而无法处理的消息数量。字符溢出由数据字符到达端口的速度超过存储速度，或由于硬件故障导致字符丢失引起。

6.1.2 计数器管理流程图

以下流程图描述了每个前述计数器何时必须递增。

计数器管理流程图说明：

帧接收过程中的计数器递增逻辑：

- 帧结束时 CPT1 递增（总线消息计数）
- 帧错误时 CPT2 递增（总线通信错误计数）
- 地址不匹配或帧错误时，不递增处理计数器
- 异常错误时 CPT3 递增（从站异常错误计数）
- 处理消息时 CPT4 递增（从站消息计数）
- 无响应时 CPT5 递增（从站无响应计数）
- 字符溢出时 CPT8 递增（总线字符溢出计数）

流程图包含以下关键判断节点：

1. 帧接收 → 字符错误？→ 长度 < 3 字节？→ CRC 是否正确？→ 从站号是否为 0？→ 从站号是否匹配？→ 功能码是否已知？→ 数据是否正确？

2. 异常处理：异常 n° 1（功能码未知）、异常 n° 2（地址不正确）、异常 n° 3（数据不正确）

3. 广播判断 → 是否为广播 → 是否有响应

6.2 附录 B - LRC/CRC 生成

6.2.1 LRC 生成

纵向冗余校验（LRC）字段为一个字节，包含一个 8 位二进制值。LRC 值由发送设备计算，发送设备将 LRC 附加到消息中。接收设备在接收消息期间重新计算 LRC，并将计算值与在 LRC 字段中接收到的实际值进行比较。如果两个值不相等，则产生错误。

LRC 的计算方法是将消息中连续的 8 位字节相加，丢弃任何进位，然后对结果取二补码。LRC 是一个 8 位字段，因此每次添加字符导致值超过 255（十进制）时，字段的值简单地从零”翻转”。由于没有第 9 位，进位被自动丢弃。

生成 LRC 的步骤：

1. 将消息中所有字节相加（不包括起始冒号和结束 CRLF）。将它们加到一个 8 位字段中，使进位被丢弃。
2. 从 FF hex（全 1）中减去最终字段值，产生一补码。
3. 加 1 产生二补码。

将 LRC 放入消息

当 8 位 LRC（2 个 ASCII 字符）在消息中传输时，高字节字符先发送，然后是低字节字符。例如，如果 LRC 值为 61 hex（0110 0001）：

图 29：LRC 字符序列

Colon | Addr | Func | Data Count | Data | Data | Data | Data | LRC Hi | LRC Lo | CR | LF

"6" | "1" | 0x36 | 0x31

示例：以下是执行 LRC 生成的 C 语言函数示例。

该函数接受两个参数：

unsigned char *auchMsg; — 指向包含用于生成 LRC 的二进制数据的消息缓冲区的指针

unsigned short usDataLen; — 消息缓冲区中的字节数

LRC 生成函数

```
static unsigned char LRC(auchMsg, usDataLen)
/* the function returns the LRC as a type unsigned char */
unsigned char *auchMsg ;          /* message to calculate LRC upon */
unsigned short usDataLen ;        /* quantity of bytes in message */
```

```
{
  unsigned char uchLRC = 0 ;      /* LRC char initialized */
  while (usDataLen--)             /* pass through message buffer */
    uchLRC += *auchMsg++ ;        /* add buffer byte without carry */
  return ((unsigned char)(-((char)uchLRC))) ; /* return twos complement */
}
```

6.2.2 CRC 生成

循环冗余校验（CRC）字段为两个字节，包含一个 16 位二进制值。CRC 值由发送设备计算，发送设备将 CRC 附加到消息中。接收设备在接收消息期间重新计算 CRC，并将计算值与在 CRC 字段中接收到的实际值进行比较。如果两个值不相等，则产生错误。

CRC 首先将一个 16 位寄存器预装载为全 1。然后开始将消息的连续 8 位字节应用到寄存器当前内容的过程。只有每个字符中的 8 位数据用于生成 CRC。起始位、停止位和校验位不参与 CRC 计算。

在 CRC 生成期间，每个 8 位字符与寄存器内容进行异或运算。然后结果向最低有效位（LSB）方向移位，最高有效位（MSB）位置填零。提取并检查 LSB。如果 LSB 为 1，则寄存器与预设的固定值进行异或。如果 LSB 为 0，则不进行异或。

此过程重复直到执行了 8 次移位。在最后一次（第 8 次）移位后，下一个 8 位字节与寄存器当前值进行异或，然后按上述方式重复 8 次移位过程。消息的所有字符都应用完毕后，寄存器的最终内容即为 CRC 值。

生成 CRC 的步骤：

1. 将 16 位寄存器装载为 FFFF hex（全 1）。称此为 CRC 寄存器。
2. 将消息的第一个 8 位字节与 16 位 CRC 寄存器的低字节进行异或，结果放入 CRC 寄存器。
3. 将 CRC 寄存器右移一位（向 LSB 方向），MSB 填零。提取并检查 LSB。
4. （如果 LSB 为 0）：重复步骤 3（再移位一次）。（如果 LSB 为 1）：将 CRC 寄存器与多项式值 0xA001（1010 0000 0000 0001）进行异或。
5. 重复步骤 3 和 4 直到完成 8 次移位。完成后，一个完整的 8 位字节即被处理。
6. 对消息的下一个 8 位字节重复步骤 2 到 5。继续直到所有字节处理完毕。
7. CRC 寄存器的最终内容即为 CRC 值。
8. 将 CRC 放入消息时，其高字节和低字节必须按如下所述交换。

将 CRC 放入消息

当 16 位 CRC（两个 8 位字节）在消息中传输时，低字节先发送，然后是高字节。例如，如果 CRC 值为 1241 hex（0001 0010 0100 0001）：

图 30：CRC 字节序列

Addr | Func | Data Count | Data | Data | Data | Data | CRC Lo | CRC Hi

0x41 | 0x12

CRC 16 计算算法流程图

算法流程：

```
0xFFFF → CRC16
CRC16 XOR BYTE → CRC16
N = 0
┌─→ Move to the right CRC16
│   Carry over
│   ┌─→ CRC16 XOR POLY → CRC16 (if carry = 1)
│   │   N = N + 1
│   └──N > 7? — No → back to "Move to the right"
│       Yes
└──End of message? — No → Following BYTE → back to "CRC16 XOR BYTE"
    Yes → END

XOR = exclusive or
N = number of information bits
POLY = calculation polynomial of the CRC 16 = 1010 0000 0000 0001
       (Generating polynomial = 1 + x^2 + x^15 + x^16)

In the CRC 16, the 1st byte transmitted is the least significant one.
```

CRC 计算示例（帧 02 07）

```
CRC register initialization    1111 1111 1111 1111
XOR 1st character              0000 0000 0000 0010
                               1111 1111 1111 1101
Move 1  0111 1111 1111 1110 |1  Flag to 1, XOR polynomial 1010 0000 0000 0001
                               1101 1111 1111 1111
Move 2  0110 1111 1111 1111 |1  Flag to 1, XOR polynomial 1010 0000 0000 0001
                               1100 1111 1111 1110
Move 3  0110 0111 1111 1111 |0
Move 4  0011 0011 1111 1111 |1  XOR polynomial 1010 0000 0000 0001
                               1001 0011 1111 1110
Move 5  0100 1001 1111 1111 |0
Move 6  0010 0100 1111 1111 |1  XOR polynomial 1010 0000 0000 0001
                               1000 0100 1111 1110
Move 7  0100 0010 0111 1111 |0
Move 8  0010 0001 0011 1111 |1  XOR polynomial 1010 0000 0000 0001
                               1000 0001 0011 1110
XOR 2nd character              0000 0000 0000 0111
                               1000 0001 0011 1001
Move 1  0100 0000 1001 1100 |1  XOR polynomial 1010 0000 0000 0001
                               1110 0000 1001 1101
Move 2  0111 0000 0100 1110 |1  XOR polynomial 1010 0000 0000 0001
                               1101 0000 0100 1111
Move 3  0110 1000 0010 0111 |1  XOR polynomial 1010 0000 0000 0001
                               1100 1000 0010 0110
```

```
Move 4  0110 0100 0001 0011 |0
Move 5  0011 0010 0000 1001 |1  XOR polynomial 1010 0000 0000 0001
                              1001 0010 0000 1000
Move 6  0100 1001 0000 0100 |0
Move 7  0010 0100 1000 0010 |0
Move 8  0001 0010 0100 0001 |0

Most significant                 least significant
The CRC 16 of the frame is then: 4112
```

示例

以下是执行 CRC 生成的 C 语言函数示例。所有可能的 CRC 值预加载到两个数组中，函数遍历消息缓冲区时只需索引即可。一个数组包含 16 位 CRC 字段高字节的所有 256 个可能 CRC 值，另一个数组包含低字节的所有值。以这种方式索引 CRC 比为消息缓冲区中每个新字符计算新 CRC 值提供更快的执行速度。

> 注意：此函数在内部执行 CRC 高/低字节的交换。函数返回的 CRC 值中字节已经交换。因此，函数返回的 CRC 值可以直接放入消息中进行传输。

该函数接受两个参数：

unsigned char *puchMsg; — 指向包含用于生成 CRC 的二进制数据的消息缓冲区的指针

unsigned short usDataLen; — 消息缓冲区中的字节数

CRC 生成函数

```
unsigned short CRC16(puchMsg, usDataLen)
/* The function returns the CRC as a unsigned short type */
unsigned char *puchMsg ;          /* message to calculate CRC upon */
unsigned short usDataLen ;        /* quantity of bytes in message */
{
  unsigned char uchCRCHi = 0xFF ; /* high byte of CRC initialized */
  unsigned char uchCRCLo = 0xFF ; /* low byte of CRC initialized */
  unsigned uIndex ;               /* will index into CRC lookup table */
  while (usDataLen--)             /* pass through message buffer */
  {
    uIndex = uchCRCLo ^ *puchMsg++ ;  /* calculate the CRC */
    uchCRCLo = uchCRCHi ^ auchCRCHi[uIndex] ;
    uchCRCHi = auchCRCLo[uIndex] ;
  }
  return (uchCRCHi << 8 | uchCRCLo) ;
}
```

高字节表（High-Order Byte Table）

```
/* Table of CRC values for high-order byte */
static unsigned char auchCRCHi[] = {
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
```

```
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01,
0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40
} ;
```

低字节表（Low-Order Byte Table）

```
/* Table of CRC values for low-order byte */
static char auchCRCLo[] = {
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07, 0xC7, 0x05, 0xC5, 0xC4,
0x04, 0xCC, 0x0C, 0x0D, 0xCD, 0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09,
0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A, 0x1E, 0xDE, 0xDF, 0x1F, 0xDD,
0x1D, 0x1C, 0xDC, 0x14, 0xD4, 0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF2, 0x32, 0x36, 0xF6, 0xF7,
0x37, 0xF5, 0x35, 0x34, 0xF4, 0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A,
0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29, 0xEB, 0x2B, 0x2A, 0xEA, 0xEE,
0x2E, 0x2F, 0xEF, 0x2D, 0xED, 0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60, 0x61, 0xA1, 0x63, 0xA3, 0xA2,
0x62, 0x66, 0xA6, 0xA7, 0x67, 0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F,
0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68, 0x78, 0xB8, 0xB9, 0x79, 0xBB,
0x7B, 0x7A, 0xBA, 0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0, 0x50, 0x90, 0x91,
0x51, 0x93, 0x53, 0x52, 0x92, 0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C,
0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B, 0x99, 0x59, 0x58, 0x98, 0x88,
0x48, 0x49, 0x89, 0x4B, 0x8B, 0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83, 0x41, 0x81, 0x80,
0x40
};
```

6.3 附录 E - 参考文献

参考文献	说明
ANSI/TIA/EIA-232-F-1997	Interface Between Data Terminal Equipment and Data Circuit-Terminating Equipment Employing Serial Binary Data Interchange.（数据终端设备与数据电路端接设备之间采用串行二进制数据交换的接口。）
ANSI/TIA/EIA-485-A-1998	Electrical Characteristics of Generators and Receivers for Use in Balanced Digital Multipoint Systems.（用于平衡数字多点系统的发生器和接收器的电气特性。）
AWG	“American Wire Gauge”（美国线规）是表示线径的标准方法，在美国和其他国家使用；线规号越大，导线参数越小。参见例如 D.G. Fink 和 H.W. Beaty, Standard Handbook for Electrical Engineers, 13th Edition, McGraw-Hill, 1993。
MODBUS.org	MODBUS 应用协议规范（MODBUS application protocol specification）。