Modbus RTU Learning based on RS485

倪煜晖

2025 年 4 月 12 日

目录

1 说明

记录 Modbus RTU 通讯协议学习过程，用于实现 xArm6 机械臂与知行机器人夹爪之间的通信，以备后续查看使用。

2 Modbus RTU 与 RS485 初探

2.1 简介

Modbus 是一种应用协议，RTU 是一种通信模式，而 RS485 是总线串行标准。前二者工作在应用层与链路层，而 RS485 工作在物理层。

2.2 联系与区别

- Modbus RTU: 一种主从通信协议。它定义了数据传输的规则，包括数据帧的格式、帧的开始和结束标志、地址域、功能码、数据区和错误检测域等。例如，在一个 Modbus RTU 帧中，地

址域用于标识从设备的地址，功能码用于指定主设备希望从设备执行的操作，如读取寄存器、写入寄存器等。

- RS485: 一种电气接口标准，它规定了数据传输的物理层特性，如信号电平、传输速率、传输距离等。RS - 485 支持多点通信，能够在长距离和高噪声环境下可靠地传输数据。

在我们的任务中，RS - 485 提供了硬件层面的通信通道，Modbus RTU 则是在这个通道上运行的协议，规定了数据的传输格式、帧结构等内容，二者相互配合来实现设备之间的通信。

2.3 重点

RS485 使用差分传输模式, 使用双绞线 A, B 之间的电位差来实现通信。它的核心是一个主机与多个从机的通讯。这里需要注意的是，这和 **I/O** 通信完全没有关系，也就是说，我们 **I/O** 的五根线大概是没用的。由于 RS485 协议对电位敏感，建议在之后断开对这五根线的连接，保证接地唯一。

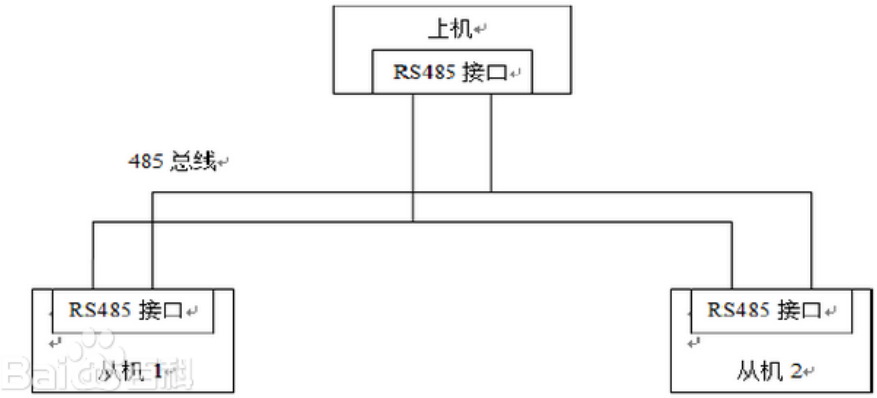

图 1: 一主多从

确认接线之后，我们把注意力更多放在 Modbus RTU 上。

3 使用机器码实现 Modbus RTU 通信

3.1 参数配置

3.1.1 夹爪要求

- 波特率：115200
- ID: 默认为 1
- 数据格式：默认的数据格式为无检验
- 校验模式：使用 16 进制 CRC 校验码，低字节在前

这里 highlight 校验模式。

使用手册中给出了示例通信码 01 06 01 02 00 64 28 1D。具体解释会在后面章节注明，也可以直接参照表格(1)。注意到，这里的最后两位是校验码，由前六位决定。我们遇到的报文无效问题由一部分应该是这个原因。

表 1: 数据帧格式说明

数据	字节	数据说明	备注
01	1	从机地址	0x01 为设备 ID 号，0x00 为广播地址（无回应）
06	1	功能码	单个保持寄存器的写入
01 02	2	数据地址	0x0102 为需要执行功能码的数据地址（执行器临时区运动位置）
00 64	2	数据值	0x0064 为 16 进制的 100，即将执行器运动位置（临时区）的值设定为 100
28 1D	2	CRC 校验码	16 进制 CRC 校验码，低字节在前

为了生成正确的校验码，我找到了生成网站并且正确生成了校验码。

图 2: 校验码

值得注意的是，xArm 手册中提到自动生成 CRC 校验码，不知道使用的是否为 16 位，也不知道采用的是低位在前还是高位在前，可能成为核心问题。

3.1.2 xArm 要求

- 波特率：默认 200000
- 12 位 6 字节十六进制编码，**自动生成 CRC 校验码**

之前一直无法执行很有可能是位数不对与设备 ID 不对，应当只输入 6 字节，选定正确设备 ID

也就是 1，后续尝试。

3.2 通用 Modbus RTU 指令 (TO DO)

TO DO

3.3 夹爪执行器特有参数 (TO DO)

TO DO

4 使用 Python 调用 Modbus RTU(TO DO)

TO DO

5 可能的替代方案——I/O 控制 (TO DO)

TO DO