M4 PLC 和工业设备对接

M4 PLC 和工业设备对接

3.1 Modbus TCP

Modbus TCP 是一种常见的工业对接协议。比较简单。简单说，就是通过对设备变量（地址）的读写，实现功能。 例如一个三色灯设备，暴露一个变量（地址）。写 1 表示亮红色，写 2 表示亮绿色，写 3 表示亮黄色，写 0 表示关灯。再比如一个滚筒线对接口，暴露两个变量（地址）。第一个变量供读取，1 表示可以放货，0 表示不能放货。第二个变量供写入，1 表示放货完成，0 表示未放货。 Modbus 协议其实有两种，TCP 和 RTU（串口）。这里只讨论 TCP 协议，即基于以太网的 Modbus 协议。 一个设备（作为 Slave）有一个 IP 和一个端口。M4 连接到这个设备（作为 Master）。 Modbus 有两种变量类型。一种是布尔量（也称线圈量）1 个 bit（比特），只有两个值，真或假，或者说 0 和 1。另一种是寄存器变量，是一个整数，短整数，占用 1 个 word（字），能表示 0 ~ 65535。

有两种看待 Modbus 设备的角度，从存储看或者从读写看。 从存储看，Modbus 协议规定了 4 个存储区，分别是 0 1 3 4 区，其中 0 区和 4 区是可读可写，1 区和 3 区是只读。

但在与客户或第三方沟通时，更直接且准确的方式是从读写角度看。问：“用什么功能码、读哪个地址。特别是当非 PLC 对 PLC 通讯，如 PLC 对 C# 或 Java 代码，不同系统设备对上面存储定义是有细微差异的，但功能码一定是准的。 协议规定了很多功能码，常用的有（注意此功能码用十六进制表示）：

例如，在对接协议时，可以约定“使用 01 功能码，读取地址 1，表示货物是否就绪”、“当取货完成，使用 05 功能码，写入地址 10”。

3.1.1 字符串

Modbus 原生只支持线圈（bool）和寄存器（word），不支持字符串、浮点型、长整型，为此提几个标准，项目中优先按照此标准。 此标准基于 寄存器 设计；对于多字节，暂时只支持大端。 暂时只支持 ASCII 字符串。即一个字符用一个字节表示。 方案协议中约定字符串的首末位地址、解析模式。

每个地址的解析模式有 2 种（需在方案协议中约定）：

一个地址表示一个字符，如 0x0041 表示 A， 0x0061 表示 a。 优点：方便处理，每个地址视为一个字符，无需拆分。 缺点：不使用左侧字节，只使用地址的右侧字节，会浪费一定空间。 一个地址表示两个字符，如 0x4161 表示 Aa， 0x3132 表示 12。 地址中第一个字节表示第一个字符，第二个字节表示第二个字符。 此模式 不支持 只读一个地址中的第一个字节/第二个字节，如：400001 ~ 400002 的 2 个地址只读前 3 个字节的值，忽略最后一个字节。 如果字符串实际长度小于定义长度，则右侧无实际值的字节写 00。 优点：充分利用空间。 缺点：每个地址要拆开处理，不便于操作；必须要填满所有字节。

暂不支持：

一个地址表示字符串长度，然后连续读 N 个地址，每个表示一个字符串。 非 ASCII 编码的字符。 例： 400001 ~ 400003 的 3 个地址表示一个定长的容器号，每个地址表示一个字节。Modbus 中的 0x0041 0x0042 0x0043 表示字符串 ABC。

3.1.2 长整数

Modbus 的寄存器（word）占 2 个字节，对应 Java 的短整数（Short）类型，原生不支持四字节整数（Integer）、八字节整数（Long）。 此标准基于 寄存器 设计；对于多字节，暂时只支持大端。 方案协议中约定长整数的首末位地址、整数占用字节数量。 例：在 400001 ~ 400002 写一个四字节的长整数，Modbus 地址中的 0x1234 0x23EF 表示十进制的值 305406959。

支持：

四字节整数

连续两个地址。第一个地址表示高位字节，第二个自制表示低位字节。

八字节整数

连续四个地址。第一个地址表示最高位字节，第四个自制表示最低位字节。

3.1.3 浮点型

对于 N 位小数，乘以 10 的 N 次方写入 Modbus。如约定 3 位小数。如果是 3.14，则 Modbus 地址里写 3140；如果是 234.123，写入 234123。如果数值范围两字节整数表示不了，则按上述大整数处理。 方案协议中约定 N 的值、占用字节数、首末位地址。 此标准基于 寄存器 设计；对于多字节，暂时只支持大端。

例 1：在 400001 写一个 2 字节浮点数，将原始值乘 10 的 2 次方写入。原始值为 1.2，则在 Modbus 地址里写 120。 例 2：在 400001 ~ 400002 写一个 4 字节的浮点数，将原始值乘 10 的 5 次方写入。原始值为 3.14159，则在 Modbus 地址里写 314159。

支持：

两字节浮点数

四字节浮点数

八字节浮点数

3.2 S7

3.2.1 基础

S7 是西门子 Smart 系列 PLC 的内部集成的一种通讯协议，封装了更丰富的数据类型，如整型、浮点型、字符型，其本质也是读写地址空间上的字节。 目前只讨论 Client/Server 的单边通信，Client 记录配置并主动访问，Server 准备被访问的数据。PLC 作为 Server，M4 作为 Client。 PLC 可将地址空间上的数个字节打包为一个 S7 节点，用于存储变量值，供 M4/PLC 读写。

链接到 Server 需要的参数

读写时需要的参数

西门子常见 PLC 型号 S1200、S1500。

3.2.2 数据类型

PLC 将地址空间的字节定义为数个 S7 节点。例：将下图的 0000~000F 地址定义为一个 S7 STRING，将 0010～0011 定义为一个 S7 INT16。 NGAPbSDjVo8Bz8xMjPPcNx12nld.png

S7 数据类型取值范围 & 与 Java 数据类型的映射关系

S7 BOOL

S7 将一个字节拆为 8 个 BOOL 节点，偏移量如 123.1、123.2，但不推荐这样用（Java 的库有可能要按照字节来，所以可能需要单独处理，所以不推荐用，推荐用 INT16）。

S7 STRING

S7 的 STRING 分为 3 段：定义长度（1 字节）a + 实际长度（1 字节）b + 实际值（N 字节，占用 a 个字节，字符串存储在前 b 个字节）。 例：字符串 ABCDEF 定义长 10，但实际长 6，则第一个字节为 10，第二个字节为 6，第三个字节开始为 ABCDEF，后面 4 个字节无效，需要忽略。

3.2.3 S7 与 Modbus 的区别

Modbus 只能读某个线圈（1 bit）或者某个寄存器（2 字节），而 S7 一般是读写一个或多个字节；所以 Modbus 的取值范围较小且只能是布尔型 Boolean 或者短整型 Short，S7 额外支持整数 Integer、长整数 Long、浮点型 Float、字符串 String。 使用上，Modbus 只能写 true/false 或者 0～65535 的整数，不能写小数、字符串；S7 支持 true/false、较大范围的整数、小数、字符串。 例：要将容器编号 “BOX-A-023” 传给 PLC， S7 可以直接传 “BOX-001”；Modbus 需要与 PLC 定义一套转换规则 “去掉‘BOX-’前缀，并将 A 映射为 1，再去掉前面的 0”，然后再将处理过的数据 “1”，“23” 传给 PLC。

3.2.4 仿真工具 snap7

Server

PZOsbcPdqo99O1xSgQscP75Snqc.png

双击 server.exe 启动 S7 Server 模拟器，用于仿真 PLC，提供地址空间。
HOaWbCQbooGUxCxkgydcAhaRnaf.png

将 “0.0.0.0” 改为 “127.0.0.1”。

点击【start】，即可启动 S7 仿真 Server

Client

GoZobKVMSo7FWnxeZ2ycgEexnzb.png

双击 clientd 启动 S7 Client 模拟器，等价于 M4，常用于读取 PLC 地址空间的值。
OptCb16VDopKzbxEt8acJFqvnSe.png

修改 IP 为 “127.0.0.1”。 如果链接真实 S7 Server 的话，需将 Rack、Slot 改为 PLC 真实的值。

点击【Connect】即可建立链接

XCyRb3bOToUZ8Rx9Y0qcVjvAnjh.png

看到红框中的 “OK” 就代表链接成功。 点击 【Data read/write】 切换到读写区域。
XnvDbyIcPoY1YSxlkvUcnikBnRb.png

读节点的操作步骤

修改【DB Number】为实际的 dbId

修改【Start】为节点的偏移量 offset

修改【Amount】为节点的长度（单位：字节）

点击【Read】按钮

读取蓝框中的值，并将其转为实际的格式

写节点的操作步骤

按照 “读节点” 都出该节点的值

修改上图蓝框中的值

这里写的是 16 进制，不是字符，也不是 10 进制的数字

注意是大端还是小端

点击【Write】按钮