# Modbus协议

* Modbus驱动程序有两个作用：

第一，从通讯管理机读取数据，这些要读取的数据类型包括：AI、DI、ACC

第二，向通讯管理机发送数据，这些要发送的数据类型包括：AO、DO

* Modbus协议规定与通许管理机读取和发送数据需要不同的功能码，不同数据对应不同的功能码，见下表

|  |  |
| --- | --- |
| 功能码 | 数据类型 |
| 01 | DI |
| 03 | AI |
| 03 | ACC |
| 05 | DO |
| 06 | AO |

注意：

由于读取AI和ACC数据使用相同的功能码，所以需要规定

用03功能码读取数据时，本人写的Modbus驱动程序中规定：通许管理机返回的数据中数据段0-5999是AI数据，数据段6000以后是 ACC数据.

|  |  |  |
| --- | --- | --- |
| 数据类型 | 在ModbusTCPCfg.xml中  的第一个寄存器号码 | 通许管理机中的第一个寄存器 |
| DI | **10001** | **00000** |
| AI | **40001** | **00000** |
| ACC | **46001** | **06000** |
| AO | **80001** | **00000** |
| DO | **90001** | **00000** |

如上表所示，从ModbusTCPCfg.xml中读取的寄存器号还需要转换成实际中通讯管理机中的寄存器号；这两者之间的转换被封装在CmeManager.cs文件中的如下函数：

private int ConvertXmlResIdToConmandResId(UpDataBase.RTWriteProxy.MType type, int xmlResId)

private int ConvertConmandResIdToXmlResId(UpDataBase.RTWriteProxy.MType type, int cmdResId)

* Modbus驱动程序向通许管理机读取和下发数据，需要一定的命令格式，即所谓的Modbus协议，它们进行交互都是使用字节流的形式。对命令格式的具体解释将分布在后面的各个小节中分别详细讲解。

## 读取DI数据—01功能码

* Modbus发送01命令，向通讯管理机请求DI数据，命令格式如下：

**16进制**

第0字节 00

第1字节 00

第2字节 00

TCP包头，占6个字节

第3字节 00

第4字节 00

这2个字节表示除TCP包头剩下的字节数，根据本人的理解，其值为6

第5字节 06

第6字节 00

开始地址，取值范围0-9

第7字节 01

01功能码

第8字节 00

起始寄存器地址，这里的0000 0004，表示从第4号寄存器开始读取DI数据

第9字节 04

第10字节 00

读取的寄存器个数，这里的0000 0010表示读取2个寄存器

第11字节 10

**特别提醒：**

“这2个字节表示除TCP包头剩下的字节数，根据本人的理解，其值为6”，之所以这么说，是因为请求数据包的长度必须是12个字节，否则通讯管理机返回错误的数据，但是本人向通许管理机发送了13个字节的请求命名，结果返回错误的数据，很难排查，折腾了很久，深受其苦。

* 通讯管理机返回DI数据格式：

必须铭记通讯管理机返回DI数据格式：

1. 返回的DI字节流数据中，1位（bit）表示一个寄存器的值；
2. 返回的每个字节的从低位开始表示寄存器的值；
3. 返回的DI数据是以字节为单位，而不是以bit为单位，剩下的bit不表示任何寄存器数据，例如：

当请求2个寄存器的值，返回一个字节，而不是2个Bit，剩下6个bit数据不表示任何寄存器的值；

当请求9个寄存器的值，返回两个字节，剩下的bit不表示任何寄存器的值。

如果Modbus驱动发送的命名是：00 00 00 00 00 06 00 01 00 04 00 10，通讯管理机返回DI数据格式会是：00 00 00 00 00 06 00 01 01

示意图，见下图：

**16进制**

第0字节 00

第1字节 00

第2字节 00

TCP包头，占6个字节

第3字节 00

第4字节 00

这2个字节表示除TCP包头剩下的字节数，根据本人的理解，其值为6

第5字节 06

开始地址，取值范围0-9

第6字节 00

第7字节 01

01功能码

第8字节 01

通讯管理机返回的字节数，

第9字节 01

从该字节的低位开始，每个bit分别表示1个寄存器的值

特别描述：

第9字节 存在01（16进制），展开为二进制即：

0000 0001

根据通讯管理机返回DI数据格式的第2条：

“返回的每个字节的从低位开始表示寄存器的值 ”

0 0 0 0 0 0 0 0 1 第9字节

本示例的第5号寄存器

起始寄存器的值，（本示例中起始寄存器是第4号寄存器）

本示例中：由于只读取2个寄存器的值，剩下的这些bit不表示任何寄存器的值

## 读取AI数据—03功能码

## 读取ACC数据—03功能码

## 发送DO数据—05功能码

## 发送DO数据—06功能码

# 地方