海悟MODBUS通讯协议

(风/水冷机房精密空调)

-Ver02

|  |  |  |  |
| --- | --- | --- | --- |
|  | 姓名 | 签字 | 日期 |
| 编制： |  |  |  |
| 校对： |  |  |  |
| 批准： |  |  |  |

**版本控制表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 日期 | 更改内容 | | 备注 |
| 1 | 2017-2-17 | 1 | 新版发布 | 新归档 |
| 2 | 2017-10-9 | 1 | 修订B.3中的地址0x8113的参数为“风机过热保护” |  |
| 2 | 修订B.4中的地址4X8103的参数为“风冷/风串水：室外温度” |
| 3 | 修订B.4中的地址4X8104的参数为“风冷/风串水：回水水温” |
| 4 | 修订B.5中的地址0x8145的参数为“风机过热保护” |
| 5 | 修订B.6中的地址4X8111的参数为“风冷/风串水：室外温度” |
| 6 | 修订B.6中的地址4X8112的参数为“风冷/风串水：回水水温” |
|  |  | 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |
| 6 |  |  |

***目录***

[1 协议简介 4](#_Toc478476688)

[2 物理层 5](#_Toc478476689)

[2.1 引言 5](#_Toc478476690)

[2.2 数据信号发送速率 5](#_Toc478476691)

[2.3 电气接口 5](#_Toc478476692)

[2.3.1 多点串行总线结构 5](#_Toc478476693)

[2.3.2 2线-MODBUS 定义 6](#_Toc478476694)

[2.3.3 RS232-MODBUS定义 7](#_Toc478476695)

[2.3.4 RS232-MODBUS要求 8](#_Toc478476696)

[2.4 多点系统需求 8](#_Toc478476697)

[2.4.1 拓扑结构 8](#_Toc478476698)

[2.4.2 长度 8](#_Toc478476699)

[2.4.3 接地形式 9](#_Toc478476700)

[2.4.4 线路终端 9](#_Toc478476701)

[2.5 电缆 9](#_Toc478476702)

[2.6 可视诊断 10](#_Toc478476703)

[3 数据链路层 10](#_Toc478476704)

[3.1 Modbus主站/从站协议原理 10](#_Toc478476705)

[3.2 Modbus帧描述 11](#_Toc478476706)

[3.3 主站/从站状态图 12](#_Toc478476707)

[3.3.1 主站状态图 12](#_Toc478476708)

[3.3.2 从站状态图 14](#_Toc478476709)

[3.3.3 主站/从站通信时序图 14](#_Toc478476710)

[3.4 两种串行传输模式 15](#_Toc478476711)

[3.4.1 RTU传输模式 15](#_Toc478476712)

[3.5 差错检验方法 19](#_Toc478476713)

[3.5.1 帧检验 19](#_Toc478476714)

[4 应用层 19](#_Toc478476715)

[4.1 协议描述 19](#_Toc478476716)

[4.2 数据编码 20](#_Toc478476717)

[4.3 MODBUS数据模型 21](#_Toc478476718)

[4.4 MODBUS事务处理的定义 22](#_Toc478476719)

[5 支持的功能码 23](#_Toc478476720)

[5.1 功能码描述 24](#_Toc478476721)

[5.1.1 01 (0x01)读线圈 24](#_Toc478476722)

[5.1.2 02 (0x02)读离散量输入 26](#_Toc478476723)

[5.1.3 03 (0x03)读保持寄存器 28](#_Toc478476724)

[5.1.4 04(0x04)读输入寄存器 30](#_Toc478476725)

[5.1.5 05 (0x05)写单个线圈 32](#_Toc478476726)

[5.1.6 06 (0x06)写单个寄存器 34](#_Toc478476727)

[5.1.7 15 (0x0F) 写多个线圈 35](#_Toc478476728)

[5.1.8 16 (0x10) 写多个寄存器 37](#_Toc478476729)

[6 MODBUS异常响应 39](#_Toc478476730)

[附录A —— CRC循环冗余校验的生成 41](#_Toc478476731)

[附录B —— MODBUS地址表 46](#_Toc478476732)

[B.1 室内机组状态-线圈(功能码：01) 46](#_Toc478476733)

[B.2 室内机组状态-寄存器(功能码: 03) 51](#_Toc478476734)

[B.3 室外机1状态-线圈(功能码：01) 53](#_Toc478476735)

[B.4 室外机1状态-寄存器(功能码: 03) 54](#_Toc478476736)

[B.5 室外机2状态-线圈(功能码：01) 54](#_Toc478476737)

[B.6 室外机2状态-寄存器(功能码: 03) 55](#_Toc478476738)

[B.7 泵柜1状态-线圈(功能码：01) 55](#_Toc478476739)

[B.8 泵柜2状态-线圈(功能码：01) 57](#_Toc478476740)

[B.9 参数设定(功能码：03, 06，16) 58](#_Toc478476741)

# 协议简介

MODBUS[协议](http://baike.baidu.com/view/36190.htm)是应用于[电子控制器](http://baike.baidu.com/view/2066657.htm)上的一种通用语言。通过此协议，[控制器](http://baike.baidu.com/view/122229.htm)相互之间、控制器经由网络（例如[以太网](http://baike.baidu.com/view/848.htm)）和其它设备之间可以通信。它已经成为一通用工业标准。有了它，不同厂商生产的控制设备可以连成[工业网络](http://baike.baidu.com/view/3392413.htm)，进行集中监控。此[协议](http://baike.baidu.com/view/36190.htm)定义了一个[控制器](http://baike.baidu.com/view/122229.htm)能认识使用的消息结构，而不管它们是经过何种网络进行通信的。它描述了一[控制器](http://baike.baidu.com/view/122229.htm)请求访问其它设备的过程，如何回应来自其它设备的请求，以及怎样侦测错误并记录。它制定了消息域格局和内容的公共格式。

当在一MODBUS网络上通信时，此[协议](http://baike.baidu.com/view/36190.htm)决定了每个[控制器](http://baike.baidu.com/view/122229.htm)须要知道它们的设备地址，识别按地址发来的消息，决定要产生何种行动。如果需要回应，[控制器](http://baike.baidu.com/view/122229.htm)将生成反馈信息并用MODBUS[协议](http://baike.baidu.com/view/36190.htm)发出。在其它网络上，包含了MODBUS[协议](http://baike.baidu.com/view/36190.htm)的消息转换为在此网络上使用的帧或包结构。这种转换也扩展了根据具体的网络解决节地址、路由路径及错误检测的方法。

MODBUS具有以下几个特点：

（1）标准、开放，用户可以免费、放心地使用MODBUS[协议](http://baike.baidu.com/view/36190.htm)，不需要交纳许可证费，也不会侵犯知识产权。目前，支持MODBUS的厂家超过400家，支持MODBUS的产品超过600种。

（2）MODBUS可以支持多种电气接口，如RS-232、RS-485等，还可以在各种介质上传送，如双绞线、光纤、无线等。

（3）MODBUS的帧格式简单、紧凑，通俗易懂。用户使用容易，厂商开发简单。

在MODBUS系统中有2种传输模式可选择：一种模式是ASCII（[美国](http://baike.baidu.com/view/2398.htm)信息[交换码](http://baike.baidu.com/view/5481005.htm)），另一种模式是RTU（远程[终端设备](http://baike.baidu.com/view/34654.htm)）。本协议采用标准的RTU协议。

详细的定义请参照MODBUS协议版本<http://www.modicon.com/techpubs/TechPubNew/PI_MBUS_300.pdf>。

# 物理层

## 引言

新的串行链路上的MODBUS解决方案应该按照EIA/TIA-485(即已知的RS485标准)实现电气接口。该标准允许“两线结构”的点对点和多点系统。此外，某些设备可能能实现“四线”RS485接口。

设备也可能能实现RS232接口。

在这种MODBUS系统中，一个主站和一个或几个从站在一个无源串行链路路上通信。

在标准的MODBUS系统中，所有设备(并行)连结在一条由3条导线组成的干线电缆上。其中两条导线(“两线”结构)形成一对平衡双绞线，双向数据在其上传送，典型比特率为每秒9600比特。

每台设备可能连结(见图1)：

或是双向连到主干电缆上，形成菊花链，

或是经分支电缆连到一个无源接头上，

或是经特种电缆连到一个有源接头上。

在设备上可用螺钉端子，RJ45，或9芯D-型连接器与电缆相接。

## 数据信号发送速率

要求9600bps波特率(控制器必须实现)。

部分控制器可选择来实现：1200, 2400, 4800, 19200, 38400bps

每种波特率，对发送方，要求其精度必须高于1%，而对接收方，必须允许2%误差。

## 电气接口

### 多点串行总线结构

图1展现的是MODBUS 多点串行链路系统中串行总线结构的总貌。

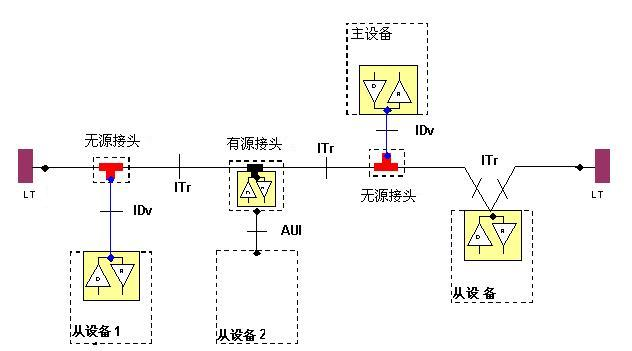


图1: 串行总线基本结构

一个MODBUS 多点串行链路系统是由主电缆(主干),和一些可能的分支电缆组成。

在主干电缆的两端需要有线路终端以使阻抗匹配 。

如图1所示，不同的设备可以在同一个MODBUS串行链路系统中运行：

集成有通信收发器的设备通过无源接头和分支电缆连接到主干上(例如从站1和主站);

没有集成通信收发器的设备通过有源接头和分支电缆连接到主干上(有源接头集成有收发器)(例如从站2)；

设备以菊花链形式直接连接到主干电缆上(例如从站n)

我们采用下列规定:

主干间的接口称为ITr(主干接口)

设备和无源接头间的接口称为 IDv (分支接口)

设备和有源接头间的接口称为 AUI (附加单元接口)

注:

某些情况下,接头可能直接连接到设备的IDv-插槽或AUI-插槽上，而不使用分支电缆。

2．一个接头可能有几个IDv插槽以连接几台设备。当它是无源接头时,称为分配器。

3．当使用有源接头时，可以通过接头的AUI或ITr接口向其提供电源。

### 2线-MODBUS 定义

串行链路上的MODBUS解决方案应当依照EIA/TIA-485标准实现“2-线”电气接口。

在这个2线-总线上，在任何时候只有一个驱动器有权发送信号。

实际上，还有第三条导线把总线上所有设备相互连接：公共地。



图2: 2-线制的一般拓扑结构

2线-MODBUS电路定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 所需电路 | | 设备 | 设备需求 | EIA/TIA-485 的命名 | 说 明 |
| 在ITr上 | 在 IDv上 |
| D1 | D1 | I/O | X | B/B’ | 收发器端子1，V1电压  （V1 > V0 表示二进制1[OFF]状态） |
| D0 | D0 | I/O | X | A/A’ | 收发器端子0，V0电压  ( V0 > V1表示二进制0[ON]状态) |
| 公共地 | 公共地 | -- | X | C/C’ | 信号和可选的电源公共地 |

注:

对于线路终端（TL），即上拉和下拉电阻，请参考“多点系统要求”部分。

在与设备和接头有关的文件(用户指南，连线指南，…)中，必须使用D0，D1和公共地的电路名字，以提高互操作能力。

### RS232-MODBUS定义

某些设备是应用RS232接口以实现DCE和DTE通信。

RS232-MODBUS 的电路定义

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 信号 | DCE | DCE(1)要求 | DTE(1)要求 | 备 注 |
| 公共地 | -- | X | X | 信号地 |
| CTS | In |  |  | 为发送而清除 |
| DCD | -- |  |  | 被侦测数据载波 (从 DCE 到 DTE ) |
| DSR | In |  |  | 数据设置就绪 |
| DTR | Out |  |  | 数据终端就绪 |
| RTS | Out |  |  | 请求发送 |
| RXD | In | X | X | 接收的数据 |
| TXD | Out | X | X | 发送的数据 |

注：

* 标有“X”的信号只在选择执行RS232-MODBUS 时才需要。
* 信号都要符合EIA/TIA-232标准。
* 每个TXD都与另一设备的RXD连接。
* RTS可以与另一设备的CTS连接。
* DTR可以与另一设备的DSR连接。

### RS232-MODBUS要求

这种可选串行链路系统上的MODBUS只应用于短距离（一般小于20m）的点到点的互连。

其次，必须遵守EIA/TIA-232标准：

* 电路定义。
* 最大线路对地电容量（2500pF，对100pF/m的电缆，则长为25m）。

关于屏蔽和使用第5类电缆的可能性，请参阅“电缆”章节。

## 多点系统需求

### 拓扑结构

没有配置中继器的RS-485 MODBUS有一个主干电缆，所有的设备沿着它直接（菊花链）或通过短的分支电缆连接起来。

主干电缆，又称总线，可以很长（见下面），它的两端必须连接在线路终端上。

在多个RS-485 MODBUS 之间使用中继器也是可以的。

### 长度

主干电缆端到端的长度必须有限制。其长度由波特率，电缆（规格，电容或特征阻抗），菊花链上的负载数，以及网络配置（2线或4线制）所决定。

对于最高波特率为9600，AWG26（或更粗）规格的电缆，其最大长度为1000m。

分支必须短，不能超过20m 。如果使用n分支的多口接头，每个分支最大长度必须限制为40m除以n。

### 接地形式

《公共地》电路（信号与可选电源公共地）必须直接连到保护地上，最好是整条总线只接在一点。通常该点可选在主站上或其接头上。

### 线路终端

沿线路传播的移动信号波遇到不连续的阻抗，造成在传输线路中的反射。为了使在RS-485电缆终端的反射最小，需要在接近总线两端点处放置线路终端。

由于传播是双向的，故在线路两端都加置终端是非常重要的，但在一个无源D0-D1平衡对线上，加的LT不能超过2个。也不能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D0和D1的两条导线之间。

线路终端可以是150欧姆（0.5W）的电阻。

当对线极性偏置时（见下述），最好选择电容（1nF，最低10V）与120欧姆（0.25W）电阻串联。

在RS232互联中，可以不用连接终端。

## 电缆

MODBUS串行链路电缆必须屏蔽。在电缆两端，其屏蔽必须接到保护地上。若在这个端部使用了连接器，该连接器外壳要连在电缆屏蔽上。

RS485-MODBUS必须使用一对平衡对线(用于D0-D1)和第三根导线(用于公共地)。

对RS485-MODBUS，必须选择足够大的连线直径以达到最大长度（1000m）。AGW24对MODBUS数据总是满足的。

在RS485-MODBUS中使用第5类电缆，最大长度可达600m.

对在RS485-系统中使用的平衡对线，建议特征阻抗高于100欧姆，导线的分布电容要小于100pF/m，如果使用屏蔽双绞线，导线与屏蔽层之间的分布电容应该小于200pF/m。RS-485的理论通讯距离为1200m（@9.6Kbps）,下表为RS-485通讯距离经验值：

|  |  |  |
| --- | --- | --- |
| （@9.6kbps） | 电缆导体截面积 | 接线端子 |
| 0～100m | 0.12mm²(26AWG) | RJ45、RJ11 |
| 0～200m | 0.20mm²(24AWG) | RJ45、RJ11 |
| 200～500m | 0.34 mm²(22AWG) | DB插座焊接 |
| 500～1000m | 0.50 mm²(20AWG) | 螺栓紧固压接 |
| 1200～1800m | 0.828mm²(18AWG) | 螺栓紧固压接 |
| 1200～1900m | 1.309mm²(16AWG) | 螺栓紧固压接 |

## 可视诊断

为可视诊断，必须用LED（发光两极管）指示通信状态和设备状态：

|  |  |  |  |
| --- | --- | --- | --- |
| 发光两极管 | 级别要求 | 说 明 | 推荐色彩 |
| 通信 | 必须 | 在帧接收或发送期间置于ON.  (两个LED表示帧接收和帧发送，或一个LED表示这两个意思。) | 黄 |
| 故障 | 推荐 | 置于ON：内部故障  闪烁：其它故障（通信故障或配置故障） | 红 |
| 设备状态 | 可选 | 置于ON：设备通电 | 绿 |

# 数据链路层

## Modbus主站/从站协议原理

MODBUS串行链路协议是一个主-从协议。 在同一时刻，只有一个主节点连接于总线，一个或多个子节点 (最大编号为 247 ) 连接于同一个串行总线。Modbus 通信总是由主节点发起。子节点在没有收到来自主节点的请求时，从不会发送数据。子节点之间从不会互相通信。主节点在同一时刻只会发起一个Modbus 事务处理。

主节点以两种模式对子节点发出 Modbus 请求:

🡺 在单播模式，主节点以特定地址访问某个子节点，子节点接到并处理完请求后，子节点向主节点返回一个报文(一个 '应答')。

在这种模式， 一个 Modbus 事务处理包含 2 个报文： 一个来自主节点的请求， 一个来自子节点的应答。

每个子节点必须有唯一的地址 (1 到 247)，这样才能区别于其它节点被独立的寻址。

🡺 在广播模式，主节点向所有的子节点发送请求。

对于主节点广播的请求没有应答返回。 广播请求一般用于写命令。所有设备必须接受广播模式的写功能。地址 0 是专门用于表示广播数据的。

单播和广播模式的区别在一个多点的结构下(如RS485)更加易于理解。



图 3: 单播模式



图 4: 广播模式

**Modbus地址规则**

Modbus 寻址空间有 256 个不同地址。

|  |  |  |
| --- | --- | --- |
| 0 | 1 ~247 | 248 ~ 255 |
| 广播地址 | 子节点单独地址 | 保留 |

地址0 保留为广播地址。 所有的子节点必须识别广播地址。

Modbus 主节点没有地址， 只有子节点必须有一个地址。该地址必须在 Modbus 串行总线上唯一。

## Modbus帧描述

Modbus 应用协议 [1] 定义了简单的独立于其下面通信层的协议数据单元(PDU - Protocol Data Unit)：



图5：Modbus协议数据单元

在不同总线或网络的Modbus 协议映射在协议数据单元之外引入了一些附加的域。发起 Modbus 事务处理的客户端构造 Modbus PDU，然后添加附加的域以构造适当的通信PDU。



图6：串行链路上的Modbus帧

* 在 Modbus 串行链路，地址域只含有子节点地址。

如前文所述，合法的子节点地址为十进制 0 – 247。 每个子设备被赋予1 – 247 范围中的地址。 主节点通过将子节点的地址放到报文的地址域对子节点寻址。当子节点返回应答时， 它将自己的地址放到应答报文的地址域以让主节点知道哪个子节点在回答。

* 功能码指明服务器要执行的动作。功能码后面可跟有表示含有请求和响应参数的数据域。
* 错误检验域是对报文内容执行 "冗余校验" 的计算结果。根据不同的传输模式 (RTU or ASCII) 使用两种不同的计算方法。

## 主站/从站状态图

Modbus 由两个不同的子层组成 :

* 主/ 从协议
* 传输模式 ( RTU 和 ASCII 模式)

下面的章节描述了主节点和子节点与传输模式无关的状态图。

RTU 和 ASCII 传输模式在下一章用两个状态图具体说明。描述了一个帧的接收和发送。

状态图词法 :

下面的状态图使用与 UML 标准标记法绘制。标记法要点如下：



当一个系统处于 "状态\_A" 时发生"触发"事件，只有当 "临界条件" 为真时系统会转换到 "状态\_B"，然后，一个"动作"被执行。

### 主站状态图

下图描述了主节点的状态特征 :



图 7: 主节点状态图

对上面的状态图的一些解释 :

* 状态 "空闲" = 无等待的请求。 这是电源上电后的初始状态。 只有在"空闲"状态请求才能被发送。发送一个请求后，主节点离开"空闲"状态， 而且不能同时发送第二个请求。
* 当单播请求发送到一个子节点，主节点将进入"等待应答" 状态， 同时一个临界超时定时启动。这个超时称为 "响应超时"。 它避免主节点永远处于"等待应答" 状态。 响应超时的时间依赖于具体应用。
* 当收到一个应答时，主节点在处理数据之前检验应答。在某些情况下，检验的结果可能为错误。如收到来自非期望的子节点的应答，或接收的的帧错误。在收到来自非期望子节点的应答时， 响应超时继续计时；当检测到帧错时，可以执行一个重试。
* 响应超时但没有收到应答时，则产生一个错误。那么主节点进入”空闲” 状态， 并发出一个重试请求。重试的最大次数取决于主节点 的设置。
* 当广播请求发送到串行总线上，没有响应从子节点返回。然而主节点需要进行延迟以便使子节点在发送新的请求处理完当前请求。该延迟被称作 "转换延迟"。因此，主节点会在返回能够发送另一个请求的“空闲”状态之前，到" 等待转换延迟"状态。
* 在单播方式，响应超时必须设置到足够的长度以使任何子节点都能处理完请求并返回响应。而广播转换延迟必须有足够的长度以使任何子节点都能只处理完请求而可以接收新的请求。 因此，转换延迟应该比响应超时要短。典型的响应超时在 9600 bps 时从1秒到几秒，而转换延迟从100 ms 到 200ms。
* 帧错误包括 : 1) 对每个字符的奇偶校验; 2) 对整个帧的冗余校验。详细解释参见 §2.6 "差错检验方法" 。

状态图以简洁的方式绘出。它没有包含对线路的访问、报文帧及传输错误重试等等。有关帧传输的细节，请参见 2.5 中的图， "两种串行式"。

### 从站状态图

下图描述了子节点的状态特征:



图 8: 子节点状态图

对上面的状态图的一些解释 :

* 状态 “空闲” = 没有等待的请求。 这是电源上电后的初始状态。
* 当收到一个请求时，子节点在处理请求中要求的动作前检验报文包。不同的错误可以发生于: 请求的格式错，非法动作，…… 当检测到错误时，必须向主节点发送应答。
* 当要求的动作完成后，单播报文要求必须格式化一个应答并发往主节点。
* 如果子节点在接收到的帧中检测到错误， 则没有响应返回到主节点。
* 任何子节点均应该定义并管理Modbus 诊断计数器以提供诊断信息。通过使用Modbus诊断功能码，可以得到这些计数值。( 参见 附录 A， 和 Modbus 应用协议规范 [1])。

### 主站/从站通信时序图

下面的图显示了主/从通信的 3 种典型情况。



图 9: 各种情形的主/从通信时序图

注 :

* 请求， 应答， 广播阶段的持续时间依赖于通信特征 (帧长度和吞吐量)。
* 等待和处理阶段的持续时间取决于子节点应用的请求处理时间。

## 两种串行传输模式

有两种串行传输模式被定义: RTU 模式 和 ASCII 模式。

它定义了报文域的位内容在线路上串行的传送。它确定了信息如何打包为报文和解码。

Modbus 串行链路上所有设备的传输模式 (和串行口参数) 必须相同。

本协议仅采用RTU传输模式。

### RTU传输模式

当设备使用RTU (Remote Terminal Unit) 模式在 Modbus 串行链路通信， 报文中每个8位字节含有两个4位十六进制字符。这种模式的主要优点是较高的数据密度，在相同的波特率下比ASCII 模式有更高的吞吐率。每个报文必须以连续的字符流传送。

本协议每个字节 ( 11 位 ) 的格式为 :

编码系统: 8–位二进制

报文中每个8位字节含有两个4位十六进制字符(0–9， A–F)

Bits per Byte: 1位起始位

8 位数据位， 首先发送最低有效位

无奇偶校验位

2位停止位



图10: RTU 模式位序列 (无校验的特殊情况)

帧检验域: 循环冗余校验 (CRC)

帧描述 :



图 11: RTU 报文帧

🡺 Modbus RTU 帧最大为256字节。

#### Modbus报文RTU帧

由发送设备将Modbus 报文构造为带有已知起始和结束标记的帧。这使设备可以在报文的开始接收新帧，并且知道何时报文结束。不完整的报文必须能够被检测到而错误标志必须作为结果被设置。

在 RTU 模式，报文帧由时长至少为3.5 个字符时间的空闲间隔区分。在后续的部分，这个时间区间被称作t3.5。





图 10: RTU 报文帧

整个报文帧必须以连续的字符流发送。

如果两个字符之间的空闲间隔大于1.5 个字符时间，则报文帧被认为不完整应该被接收节点丢弃。



注 :

RTU 接收驱动程序的实现，由于t1.5 和 t3.5 的定时，隐含着大量的对中断的管理。在高通信速率下，这导致 CPU 负担加重。因此，在通信速率等于或低于 19200 Bps 时，这两个定时必须严格遵守；对于波特率大于19200 Bps 的情形，应该使用2 个定时的固定值：建议的字符间超时时间(t1.5)为750µs，帧间的超时时间 (t1.5) 为 1.750ms。

下图表示了对RTU 传输模式状态图的描述。 "主节点" 和 "子节点" 的不同角度均在相同的图中表示：



图 11: RTU 传输模式状态图

上面状态图的一些解释:

* 从 "初始" 态到 “空闲” 态转换需要t3.5 定时超时: 这保证帧间延迟
* “空闲” 态是没有发送和接收报文要处理的正常状态。
* 在 RTU 模式， 当没有活动的传输的时间间隔达3.5 个字符长时，通信链路被认为在 “空闲” 态。
* 当链路空闲时， 在链路上检测到的任何传输的字符被识别为帧起始。 链路变为 "活动" 状态。 然后， 当链路上没有字符传输的时间间个达到t3.5 后，被识别为帧结束。
* 检测到帧结束后，完成CRC 计算和检验。然后，分析地址域以确定帧是否发往此设备，如果不是，则丢弃此帧。 为了减少接收处理时间，地址域可以在一接到就分析，而不需要等到整个帧结束。这样，CRC 计算只需要在帧寻址到该节点 (包括广播帧) 时进行。

#### CRC校验

在RTU 模式包含一个对全部报文内容执行的，基于循环冗余校验 (CRC - Cyclical Redundancy Checking) 算法的错误检验域。CRC 域检验整个报文的内容。不管报文有无奇偶校验，均执行此检验。

CRC 包含由两个8位字节组成的一个16位值。

CRC 域作为报文的最后的域附加在报文之后。计算后，首先附加低字节，然后是高字节。CRC 高字节为报文发送的最后一个子节。

附加在报文后面的CRC 的值由发送设备计算。接收设备在接收报文时重新计算 CRC 的值，并将计算结果于实际接收到的CRC 值相比较。如果两个值不相等，则为错误。

CRC 的计算, 开始对一个16位寄存器预装全1。 然后将报文中的连续的8位子节对其进行后续的计算。只有字符中的8个数据位参与生成CRC 的运算，起始位，停止位和校验位不参与 CRC 计算。

CRC 的生成过程中， 每个 8–位字符与寄存器中的值异或。然后结果向最低有效位(LSB)方向移动(Shift) 1位，而最高有效位(MSB)位置充零。 然后提取并检查 LSB：如果LSB 为1， 则寄存器中的值与一个固定的预置值异或；如果LSB 为 0， 则不进行异或操作。

这个过程将重复直到执行完8次移位。完成最后一次（第8次）移位及相关操作后，下一个8位字节与寄存器的当前值异或，然后又同上面描述过的一样重复8次。当所有报文中子节都运算之后得到的寄存器忠的最终值，就是CRC。

当CRC 附加在报文之后时，首先附加低字节，然后是高字节。在附录 A 含有CRC 生成的详细示例。

## 差错检验方法

标准 Modbus 串行链路的可靠性基于两种错误检验:

* 奇偶校验 (偶或奇) 应该被每个字符采用。
* 帧检验 (LRC or CRC) 必须运用于整个报文。

由设备 (主节点或子节点) 生成的字符检验和帧检验发送前附加于报文体。设备 (子节点或主节点) 在接收时检验每个字符和整个报文。

主节点被用户配置为在放弃事务处理前等待一个预定的超时间隔 (响应超时)。这个间隔被设置成任何子节点有足够的时间正常响应 (单播请求)。如果子节点检测到错误，则报文不起作用。子节点将不会构造对主节点的响应。因此，将达到超时时间能使主节点的程序处理错误。注意，当寻址到不存在的子设备的报文也会导致超时错误。

### 帧检验

在RTU 模式，包含一个对全部报文内容执行的，基于循环冗余校验 (CRC - Cyclical Redundancy Checking) 算法的错误检验域。CRC 域检验整个报文的内容。不管报文有无奇偶校验，均执行此检验。

# 应用层

## 协议描述

MODBUS协议定义了一个与基础通信层无关的简单协议数据单元（PDU）。特定总线或网络上的MODBUS协议映射能够在应用数据单元（ADU）上引入一些附加域。



图12：通用MODBUS帧

启动MODBUS事务处理的客户机创建MODBUS应用数据单元。功能码向服务器指示将执行哪种操作。

MODBUS协议建立了客户机启动的请求格式。

用一个字节编码MODBUS数据单元的功能码域。有效的码字范围是十进制1-255（128-255为异常响应保留）。当从客户机向服务器设备发送报文时，功能码域通知服务器执行哪种操作。

向一些功能码加入子功能码来定义多项操作。

从客户机向服务器设备发送的报文数据域包括附加信息，服务器使用这个信息执行功能码定义的操作。这个域还包括离散项目和寄存器地址、处理的项目数量以及域中的实际数据字节数。

在某种请求中，数据域可以是不存在的（0长度），在此情况下服务器不需要任何附加信息。功能码仅说明操作。

如果在一个正确接收的MODBUS ADU中，不出现与请求MODBUS功能有关的差错，那么服务器至客户机的响应数据域包括请求数据。如果出现与请求MODBUS功能有关的差错，那么域包括一个异常码，服务器应用能够使用这个域确定下一个执行的操作。

例如，客户机能够读一组离散量输出或输入的开/关状态，或者客户机能够读/写一组寄存器的数据内容。

当服务器对客户机响应时，它使用功能码域来指示正常（无差错）响应或者出现某种差错（称为异常响应）。对于一个正常响应来说，服务器仅对原始功能码响应。



图13：MODBUS事务处理（无差错）

对于异常响应，服务器返回一个与原始功能码等同的码，设置该原始功能码的最高有效位为逻辑1。



图14 MODBUS事务处理（异常响应）

注释：需要管理超时，以便明确地等待可能不会出现的应答。

串行链路上第一个MODBUS执行的长度约束限制了MODBUS PDU大小（最大RS485ADU=256字节）。

因此，对串行链路通信来说，MODBUS PDU=256-服务器地址（1字节）-CRC（2字节）＝253字节。

## 数据编码

* MODBUS使用大端对齐模式来表示地址和数据项。这意味着当发射多个字节时，首先发送最高有效位。例如：

寄存器大小 值

16 – 比特 0x1234 发送的第一字节为 0x12 然后0x34

## MODBUS数据模型

MODBUS以一系列具有不同特征表格上的数据模型为基础。四个基本表格为：

|  |  |  |  |
| --- | --- | --- | --- |
| 基本表格 | 对象类型 | 访问类型 | 内容 |
| 离散量输入 | 单个比特 | 只读 | I/O系统提供这种类型数据 |
| 线圈 | 单个比特 | 读写 | 通过应用程序改变这种类型数据 |
| 输入寄存器 | 16-比特字 | 只读 | I/O系统提供这种类型数据 |
| 保持寄存器 | 16-比特字 | 读写 | 通过应用程序改变这种类型数据 |

输入与输出之间以及比特寻址的和字寻址的数据项之间的区别并没有暗示任何应用操作。如果这是对可疑对象核心部分最自然的解释，那么这种区别是可完全接受的，而且很普通，以便认为四个表格全部覆盖了另外一个表格。

对于基本表格中任何一项，协议都允许单个地选择65536个数据项，而且设计那些项的读写操作可以越过多个连续数据项直到数据大小规格限制，这个数据大小规格限制与事务处理功能码有关。

很显然，必须将通过MODBUS处理的所有数据放置在设备应用存储器中。但是，存储器的物理地址不应该与数据参考混淆。要求仅仅是数据参考与物理地址的链接。

MODBUS功能码中使用的MODBUS逻辑参考数字是以0开始的无符号整数索引。

* MODBUS模型实现的实例

下例实例示出了两种在设备中构造数据的方法。可能有不同的结构，这个文件中没有全部描述出来。每个设备根据其应用都有它自己的数据结构。

实例1：有4个独立块的设备

下例实例示出了设备中的数据结构，这个设备含有数字量和模拟量、输入量和输出量。由于不同块中的数据不相关，每个块是相互独立。按不同MODBUS功能码访问每个块。



图15：带有独立块的MODBUS数据模型

实例2：仅有1个块的设备

在这个实例中，设备仅有1个数据块。通过几个MODBUS功能码可能得到一个相同数据，或者通过16比特访问或1个访问比特。



图16：仅带有1个块的MODBUS数据模型

## MODBUS事务处理的定义

下列状态图描述了在服务器侧MODBUS事务处理的一般处理过程。



图17：MODBUS事务处理的状态图

一旦服务器处理请求，使用合适的MODBUS服务器事务建立MODBUS响应。

根据处理结果，可以建立两种类型响应：

* 一个正MODBUS响应：
  + 响应功能码 = 请求功能码
* 一个MODBUS异常响应：
* 用来为客户机提供处理过程中与被发现的差错相关的信息；
* 响应功能码 = 请求功能码 + 0x80；
* 提供一个异常码来指示差错原因。

# 支持的功能码

本协议支持的功能码如下表所示：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 含义 | 功能码 | | 操作对象 | 起始地址 |
| 十进制 | 十六进制 |
| 位访问 | 读输入离散量 | 02 | 02 | 开关量输入或只读位变量 | 10001~19999 |
| 读线圈 | 01 | 01 | 可读写位变量或继电器输出 | 00001~09999 |
| 写单个线圈 | 05 | 05 |
| 写多个线圈 | 15 | 0F |
| 字(16位)访问 | 读输入寄存器 | 04 | 04 | 输入存储器(只读) | 30001~39999 |
| 读多个寄存器 | 03 | 03 | 内部存储或物理输出 | 40001~49999 |
| 写单个寄存器 | 06 | 06 |
| 写多个寄存器 | 16 | 10 |

## 功能码描述

### 01 (0x01)读线圈

在一个远程设备中，使用该功能码读取线圈的1至2000连续状态。请求PDU详细说明了起始地址，即指定的第一个线圈地址和线圈编号。从零开始寻址线圈。因此寻址线圈1-16为0-15。

根据数据域的每个比特将响应报文中的线圈分成为一个线圈。指示状态为1= ON 和0= OFF。第一个数据字节的LSB（最低有效位）包括在询问中寻址的输出。其它线圈依次类推，一直到这个字节的高位端为止，并在后续字节中从低位到高位的顺序。

如果返回的输出数量不是八的倍数，将用零填充最后数据字节中的剩余比特（一直到字节的高位端）。字节数量域说明了数据的完整字节数。

请求PDU

|  |  |  |
| --- | --- | --- |
| 功能码 | 1个字节 | 0x01 |
| 起始地址 | 2个字节 | 0x0000 至0xFFFF |
| 线圈数量 | 2个字节 | 1至2000（0x7D0） |

响应PDU

|  |  |  |
| --- | --- | --- |
| 功能码 | 1个字节 | 0x01 |
| 字节数 | 1个字节 | N\* |
| 线圈状态 | N个字节 | n＝N或N+1 |

\*

N＝输出数量/8，如果余数不等于0，那么N = N+1

错误

|  |  |  |
| --- | --- | --- |
| 功能码 | 1个字节 | 功能码＋0x80 |
| 异常码 | 1个字节 | 01或02或03或04 |

这是一个请求读离散量输出20-38的实例：

|  |  |  |  |
| --- | --- | --- | --- |
| 请求 | | 响应 | |
| 域名 | （十六进制） | 域名 | （十六进制） |
| 功能  起始地址Hi  起始地址Lo  输出数量Hi  输出数量Lo | 01 | 功能  字节数  输出状态27-20  输出状态35-28  输出状态38-36 | 01 |
| 00 | 03 |
| 13 | CD |
| 00 | 6B |
| 13 | 05 |

将输出27-20的状态表示为十六进制字节值CD，或二进制1100 1101。输出27是这个字节的MSB，输出20是LSB。

通常，将一个字节内的比特表示为MSB位于左侧，LSB位于右侧。第一字节的输出从左至右为27至20。下一个字节的输出从左到右为35至28。当串行发射比特时，从LSB向MSB传输： 20 . . . 27、28 . . . 35等等。

在最后的数据字节中，将输出状态38-36表示为十六进制字节值05，或二进制0000 0101。输出38是左侧第六个比特位置，输出36是这个字节的LSB。用零填充五个剩余高位比特。

注：用零填充五个剩余比特（一直到高位端）。



图18：读取线圈状态图

### 02 (0x02)读离散量输入

在一个远程设备中，使用该功能码读取离散量输入的1至2000连续状态。请求PDU详细说明了起始地址，即指定的第一个输入地址和输入编号。从零开始寻址输入。因此寻址输入1-16为0-15。

根据数据域的每个比特将响应报文中的离散量输入分成为一个输入。指示状态为1= ON 和0= OFF。第一个数据字节的LSB（最低有效位）包括在询问中寻址的输入。其它输入依次类推，一直到这个字节的高位端为止，并在后续字节中从低位到高位的顺序。

如果返回的输入数量不是八的倍数，将用零填充最后数据字节中的剩余比特（一直到字节的高位端）。字节数量域说明了数据的完整字节数。

请求PDU

|  |  |  |
| --- | --- | --- |
| 功能码 | 1个字节 | 0x02 |
| 起始地址 | 2个字节 | 0x0000至0xFFFF |
| 输入数量 | 2个字节 | 1至2000（0x7D0） |

响应PDU

|  |  |  |
| --- | --- | --- |
| 功能码 | 1个字节 | 0x82 |
| 字节数 | 1个字节 | N\* |
| 输入状态 | N\*×1个字节 |  |

\*N＝输出数量/8，如果余数不等于0，那么N = N+1

错误

|  |  |  |
| --- | --- | --- |
| 差错码 | 1字节 | 0x82 |
| 异常码 | 1字节 | 01或02或03或04 |

这是一个请求读取离散量输入197-218的实例：

|  |  |  |  |
| --- | --- | --- | --- |
| 请求 | | 响应 | |
| 域名 | （十六进制） | 域名 | （十六进制） |
| 功能  起始地址Hi  起始地址Lo  输出数量Hi  输出数量Lo | 02 | 功能  字节数  输入状态204-197  输入状态212-205  输入状态218-213 | 02 |
| 00 | 03 |
| C4 | AC |
| 00 | DB |
| 16 | 35 |

将离散量输入状态204-197表示为十六进制字节值AC，或二进制1010 1100。输入204是这个字节的MSB，输入197是这个字节的LSB。

将离散量输入状态218-213表示为十六进制字节值35，或二进制0011 0101。输入218位于左侧第3比特，输入213是LSB。

注：用零填充2个剩余比特（一直到高位端）。



图19：读离散量输入的状态图

### 03 (0x03)读保持寄存器

在一个远程设备中，使用该功能码读取保持寄存器连续块的内容。请求PDU说明了起始寄存器地址和寄存器数量。从零开始寻址寄存器。因此，寻址寄存器1-16为0-15。

将响应报文中的寄存器数据分成每个寄存器有两字节，在每个字节中直接地调整二进制内容。

对于每个寄存器，第一个字节包括高位比特，并且第二个字节包括低位比特。

请求

|  |  |  |
| --- | --- | --- |
| 功能码 | 1个字节 | 0x03 |
| 起始地址 | 2个字节 | 0x0000至0xFFFF |
| 寄存器数量 | 2个字节 | 1至125（0x7D） |

响应

|  |  |  |
| --- | --- | --- |
| 功能码 | 1个字节 | 0x03 |
| 字节数 | 1个字节 | 2×N\* |
| 寄存器值 | N\*×2个字节 |  |

\*N＝寄存器的数量

错误

|  |  |  |
| --- | --- | --- |
| 差错码 | 1个字节 | 0x83 |
| 异常码 | 1个字节 | 01或02或03或04 |

这是一个请求读寄存器108-110的实例：

|  |  |  |  |
| --- | --- | --- | --- |
| 请求 | | 响应 | |
| 域名 | （十六进制） | 域名 | （十六进制） |
| 功能  高起始地址  低起始地址  高寄存器编号  低寄存器编号 | 03 | 功能  字节数  寄存器值Hi（108）  寄存器值Lo（108）  寄存器值Hi（109）  寄存器值Lo（109）  寄存器值Hi（110）  寄存器值Lo（110） | 03 |
| 00 | 06 |
| 6B | 02 |
| 00 | 2B |
| 03 | 00 |
|  | | 00 |
| 00 |
| 64 |

将寄存器108的内容表示为两个十六进制字节值02 2B，或十进制555。将寄存器109-110的内容分别表示为十六进制00 00和00 64，或十进制0和100。



图20：读保持寄存器的状态图

### 04(0x04)读输入寄存器

在一个远程设备中，使用该功能码读取1至大约125的连续输入寄存器。请求PDU说明了起始地址和寄存器数量。从零开始寻址寄存器。因此，寻址输入寄存器1-16为0-15。

将响应报文中的寄存器数据分成每个寄存器为两字节，在每个字节中直接地调整二进制内容。

对于每个寄存器，第一个字节包括高位比特，并且第二个字节包括低位比特。

请求

|  |  |  |
| --- | --- | --- |
| 功能码 | 1个字节 | 0x04 |
| 起始地址 | 2个字节 | 0x0000至0xFFFF |
| 输入寄存器数量 | 2个字节 | 0x0001至0x007D |

响应

|  |  |  |
| --- | --- | --- |
| 功能码 | 1个字节 | 0x04 |
| 字节数 | 1个字节 | 2×N\* |
| 输入寄存器 | N\*×2个字节 |  |

\*N＝输入寄存器的数量

错误

|  |  |  |
| --- | --- | --- |
| 差错码 | 1个字节 | 0x84 |
| 异常码 | 1个字节 | 01或02或03或04 |

这是一个请求读输入寄存器9的实例：

|  |  |  |  |
| --- | --- | --- | --- |
| 请求 | | 响应 | |
| 域名 | （十六进制） | 域名 | （十六进制） |
| 功能  起始地址Hi  起始地址Lo  输入寄存器数量Hi  输入寄存器数量Lo | 04 | 功能  字节数  输入寄存器9 Hi  输入寄存器9 Lo | 04 |
| 00 | 02 |
| 08 | 00 |
| 00 | 0A |
| 01 |  |

将输入寄存器9的内容表示为两个十六进制字节值00 0A，或十进制10。



图21：读输入寄存器的状态图

### 05 (0x05)写单个线圈

在一个远程设备上，使用该功能码写单个输出为ON或OFF。

请求数据域中的常量说明请求的ON/OFF状态。十六进制值FF 00请求输出为ON。十六进制值00 00请求输出为OFF。其它所有值均是非法的，并且对输出不起作用。

请求PDU说明了强制的线圈地址。从零开始寻址线圈。因此，寻址线圈1为0。线圈值域的常量说明请求的ON/OFF状态。十六进制值0XFF00请求线圈为ON。十六进制值0X0000请求线圈为OFF。其它所有值均为非法的，并且对线圈不起作用。

正常响应是请求的应答，在写入线圈状态之后返回这个正常响应。

请求

|  |  |  |
| --- | --- | --- |
| 功能码 | 1个字节 | 0x05 |
| 输出地址 | 2个字节 | 0x0000至0xFFFF |
| 输出值 | 2个字节 | 0x0000至0x00 |

响应

|  |  |  |
| --- | --- | --- |
| 功能码 | 1个字节 | 0x05 |
| 输出地址 | 2个字节 | 0x0000至0xFFFF |
| 输出值 | 2个字节 | 0x0000至0xFF00 |

错误

|  |  |  |
| --- | --- | --- |
| 差错码 | 1个字节 | 0x85 |
| 异常码 | 1个字节 | 01或02或03或04 |

这是一个请求写线圈173为ON的实例：

|  |  |  |  |
| --- | --- | --- | --- |
| 请求 | | 响应 | |
| 域名 | （十六进制） | 域名 | （十六进制） |
| 功能  输出地址Hi  输出地址Lo  输出值Hi  输出值Lo | 05 | 功能  输出地址Hi  输出地址Lo  输出值Hi  输出值Lo | 05 |
| 00 | 00 |
| AC | AC |
| FF | FF |
| 00 | 00 |



图22：写单个输出状态图

### 06 (0x06)写单个寄存器

在一个远程设备中，使用该功能码写单个保持寄存器。

请求PDU说明了被写入寄存器的地址。从零开始寻址寄存器。因此，寻址寄存器1为0。

正常响应是请求的应答，在写入寄存器内容之后返回这个正常响应。

请求

|  |  |  |
| --- | --- | --- |
| 功能码 | 1个字节 | 0x06 |
| 寄存器地址 | 2个字节 | 0x0000至0xFFFF |
| 寄存器值 | 2个字节 | 0x0000至0xFFFF |

响应

|  |  |  |
| --- | --- | --- |
| 功能码 | 1个字节 | 0x06 |
| 寄存器地址 | 2个字节 | 0x0000至0xFFFF |
| 寄存器值 | 2个字节 | 0x0000至0xFFFF |

错误

|  |  |  |
| --- | --- | --- |
| 差错码 | 1个字节 | 0x86 |
| 异常码 | 1个字节 | 01或02或03或04 |

这是一个请求将十六进制00 03写入寄存器2的实例：

|  |  |  |  |
| --- | --- | --- | --- |
| 请求 | | 响应 | |
| 域名 | （十六进制） | 域名 | （十六进制） |
| 功能  寄存器地址Hi  寄存器地址Lo  寄存器值Hi  寄存器值Lo | 06 | 功能  输出地址Hi  输出地址Lo  输出值Hi  输出值Lo | 06 |
| 00 | 00 |
| 01 | 01 |
| 00 | 00 |
| 03 | 03 |



图23：写单个寄存器状态图

### 15 (0x0F) 写多个线圈

在一个远程设备中，使用该功能码强制线圈序列中的每个线圈为ON或OFF。请求PDU说明了强制的线圈参考。从零开始寻址线圈。因此，寻址线圈1为0。

请求数据域的内容说明了被请求的ON/OFF状态。域比特位置中的逻辑“1”请求相应输出为ON。域比特位置中的逻辑“0”请求相应输出为OFF。

正常响应返回功能码、起始地址和强制的线圈数量。

请求PDU

|  |  |  |
| --- | --- | --- |
| 功能码 | 1个字节 | 0x0F |
| 起始地址 | 2个字节 | 0x0000至0xFFFF |
| 输出数量 | 2个字节 | 0x0001至0x07B0 |
| 字节数 | 1个字节 | N\* |
| 输出值 | N\*×1个字节 |  |

\*N＝输出数量/8，如果余数不等于0，那么N = N+1

响应PDU

|  |  |  |
| --- | --- | --- |
| 功能码 | 1个字节 | 0x0F |
| 起始地址 | 2个字节 | 0x0000至0xFFFF |
| 输出数量 | 2个字节 | 0x0001至0x07B0 |

错误

|  |  |  |
| --- | --- | --- |
| 差错码 | 1个字节 | 0x8F |
| 异常码 | 1个字节 | 01或02或03或04 |

这是一个请求从线圈20开始写入10个线圈的实例：

请求的数据内容为两个字节：十六进制CD 01 (二进制1100 1101 0000 0001)。使用下列方法，二进制比特对应输出。

比特：1 1 0 0 1 1 0 1 0 0 0 0 0 0 0 1

输出：27 26 25 24 23 22 21 20 – – – – – – 29 28

传输的第一字节(十六进制CD)寻址为输出27-20，在这种设置中，最低有效比特寻址为最低输出（20）。

传输的下一字节(十六进制01)寻址为输出29-28，在这种设置中，最低有效比特寻址为最低输出（28）。

应该用零填充最后数据字节中的未使用比特。

|  |  |  |  |
| --- | --- | --- | --- |
| 请求 | | 响应 | |
| 域名 | （十六进制） | 域名 | （十六进制） |
| 功能  起始地址Hi  起始地址Lo  输出数量Hi  输出数量Lo  字节数  输出值Hi  输出值Lo | 0F | 功能  起始地址Hi  起始地址Lo  输出数量Hi  输出数量Lo | 0F |
| 00 | 00 |
| 13 | 13 |
| 00 | 00 |
| 0A | 0A |
| 02 |  |
| CD |
| 01 |



图24：写多个输出的状态图

### 16 (0x10) 写多个寄存器

在一个远程设备中，使用该功能码写连续寄存器块(1至约120个寄存器)。

在请求数据域中说明了请求写入的值。每个寄存器将数据分成两字节。

正常响应返回功能码、起始地址和被写入寄存器的数量。

请求PDU

|  |  |  |
| --- | --- | --- |
| 功能码 | 1个字节 | 0x10 |
| 起始地址 | 2个字节 | 0x0000至0xFFFF |
| 寄存器数量 | 2个字节 | 0x0001至0x0078 |
| 字节数 | 1个字节 | 2×N\* |
| 寄存器值 | N\*×2个字节 | 值 |

\*N＝寄存器数量

响应PDU

|  |  |  |
| --- | --- | --- |
| 功能码 | 1个字节 | 0x10 |
| 起始地址 | 2个字节 | 0x0000至0xFFFF |
| 寄存器数量 | 2个字节 | 1至123（0x7B） |

错误

|  |  |  |
| --- | --- | --- |
| 差错码 | 1个字节 | 0x90 |
| 异常码 | 1个字节 | 01或02或03或04 |

这是一个请求将十六进制00 0A和01 02写入以2开始的两个寄存器的实例：

|  |  |  |  |
| --- | --- | --- | --- |
| 请求 | | 响应 | |
| 域名 | （十六进制） | 域名 | （十六进制） |
| 功能  起始地址Hi  起始地址Lo  寄存器数量Hi  寄存器数量Lo  字节数  寄存器值Hi  寄存器值Lo  寄存器值Hi  寄存器值Lo | 10 | 功能  起始地址Hi  起始地址Lo  寄存器数量Hi  寄存器数量Lo | 10 |
| 00 | 00 |
| 01 | 01 |
| 00 | 00 |
| 02 | 02 |
| 04 |  |
| 00 |
| 0A |
| 01 |
| 02 |



图25：写多个寄存器的状态图

# MODBUS异常响应

当客户机设备向服务器设备发送请求时，客户机希望一个正常响应。从主站询问中出现下列四种可能事件之一：

* 如果服务器设备接收到无通信错误的请求，并且可以正常地处理询问，那么服务器设备将返回一个正常响应。
* 如果由于通信错误，服务器没有接收到请求，那么不能返回响应。客户机程序将最终处理请求的超时状态。
* 如果服务器接收到请求，但是检测到一个通信错误（奇偶校验、LRC、CRC、...），那么不能返回响应。客户机程序将最终处理请求的超时状态。
* 如果服务器接收到无通信错误的请求，但不能处理这个请求（例如，如果请求读一个不存在的输出或寄存器），服务器将返回一个异常响应，通知用户错误的本质特性。

异常响应报文有两个与正常响应不同的域：

功能码域：在正常响应中，服务器利用响应功能码域来应答最初请求的功能码。所有功能码的最高有效位（MSB）都为0（它们的值都低于十六进制80）。在异常响应中，服务器设置功能码的MSB为1。这使得异常响应中的功能码值比正常响应中的功能码值高十六进制80。

通过设置功能码的MSB，客户机的应用程序能够识别异常响应，并且能够检测异常码的数据域。

数据域：在正常响应中，服务器可以返回数据域中数据或统计表（请求中要求的任何报文）。在异常响应中，服务器返回数据域中的异常码。这就定义了产生异常的服务器状态。

客户机请求和服务器异常响应的实例：

|  |  |  |  |
| --- | --- | --- | --- |
| 请求 | | 响应 | |
| 域名 | （十六进制） | 域名 | （十六进制） |
| 功能 | 01 | 功能 | 81 |
| 起始地址Hi | 04 | 异常码 | 02 |
| 起始地址Lo | A1 |  |  |
| 输出数量Hi | 00 |  |
| 输出数量Lo | 01 |  |

在这个实例中，客户机对服务器设备寻址请求。功能码(01)用于读输出状态操作。它将请求地址1245(十六进制04A1)的输出状态。值得注意的是，象输出域(0001)号码说明的那样，只读出一个输出。

如果在服务器设备中不存在输出地址，那么服务器将返回异常码(02)的异常响应。这就说明从站的非法数据地址。

从下页开始异常码的列表。

|  |  |  |
| --- | --- | --- |
| MODBUS异常码 | | |
| 代码 | 名称 | 含义 |
| 01 | 非法功能 | 对于服务器(或从站)来说，询问中接收到的功能码是不可允许的操作。这也许是因为功能码仅仅适用于新设备而在被选单元中是不可实现的。同时，还指出服务器(或从站)在错误状态中处理这种请求，例如：因为它是未配置的，并且要求返回寄存器值。 |
| 02 | 非法数据地址 | 对于服务器(或从站)来说，询问中接收到的数据地址是不可允许的地址。特别是，参考号和传输长度的组合是无效的。对于带有100个寄存器的控制器来说，带有偏移量96和长度4的请求会成功，带有偏移量96和长度5的请求将产生异常码02。 |
| 03 | 非法数据值 | 对于服务器(或从站)来说，询问中包括的值是不可允许的值。这个值指示了组合请求剩余结构中的故障，例如：隐含长度是不正确的。并不意味着，因为MODBUS协议不知道任何特殊寄存器的任何特殊值的重要意义，寄存器中被提交存储的数据项有一个应用程序期望之外的值。 |
| 04 | 从站设备故障 | 当服务器(或从站)正在设法执行请求的操作时，产生不可重新获得的差错。 |

# 附录A —— CRC循环冗余校验的生成

循环冗余校验 (CRC) 域为两个字节，包含一个二进制 16 位值。附加在报文后面的CRC 的值由发送设备计算。接收设备在接收报文时重新计算 CRC 的值，并将计算结果于实际接收到的CRC 值相比较。如果两个值不相等，则为错误。

CRC 的计算, 开始对一个16位寄存器预装全1. 然后将报文中的连续的8位子节对其进行后续的计算。只有字符中的8个数据位参与生成CRC 的运算，起始位，停止位和校验位不参与 CRC 计算。

CRC 的生成过程中， 每个 8–位字符与寄存器中的值异或。然后结果向最低有效位 (LSB) 方向移动(Shift) 1位，而最高有效位 (MSB) 位置充零。 然后提取并检查 LSB：如果LSB 为1， 则寄存器中的值与一个固定的预置值异或；如果LSB 为 0， 则不进行异或操作。

这个过程将重复直到执行完8次移位。完成最后一次（第8次）移位及相关操作后，下一个8位字节与寄存器的当前值异或，然后又同上面描述过的一样重复8次。当所有报文中子节都运算之后得到的寄存器中的最终值，就是CRC.

生成 CRC 的过程为:

1. 将一个 16位寄存器装入十六进制 FFFF (全1). 将之称作 CRC 寄存器.

2. 将报文的第一个8位字节与16位CRC寄存器的低字节异或，结果置于CRC 寄存器.

3. 将 CRC 寄存器右移1位 (向 LSB 方向)， MSB 充零. 提取并检测 LSB.

4. (如果LSB 为 0): 重复步骤 3 (另一次移位).

(如果LSB 为1): 对CRC 寄存器异或多项式值 0xA001 (1010 0000 0000 0001).

5. 重复步骤 3 和 4，直到完成 8 次移位。当做完此操作后，将完成对8位字节的完整操作。

6. 对报文中的下一个字节重复步骤2 到5，继续此操作直至所有报文被处理完毕。

7. CRC 寄存器中的最终内容为CRC 值.

8. 当放置CRC 值于报文时，如下面描述的那样，高低字节必须交换。

将 CRC 放置于报文

当16位 CRC (2 个 8 位字节) 在报文中传送时，低位字节首先发送，然后是高位字节。

例如， 如果 CRC 值为十六进制1241 (0001 0010 0100 0001):



图 26: CRC 字节序列

**CRC 16 计算算法**



XOR = 异或

N = 字节的信息位

POLY = CRC 16 多项式计算 = 1010 0000 0000 0001

(生成多项式 = 1 + x2 + x 15 + x 16)

在 CRC 16 中， 发送的第一个字节为低字节.

CRC 计算示例 (帧 02 07)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CRC 寄存器初始化 |  | 1111 | 1111 | 1111 | 1111 |
| XOR 第一个字符 |  | 0000 | 0000 | 0000 | 0000 |
|  |  | 1111 | 1111 | 1111 | 1101 |
|  | 移位 1 | 0111 | 1111 | 1111 | 11101 |
|  |  | 1010 | 0000 | 0000 | 0001 |
| 标志 1， XOR 多项式 |  | 1101 | 1111 | 1111 | 1111 |
|  | 移位2 | 0110 | 1111 | 1111 | 11111 |
| 标志1， XOR多项式 |  | 1010 | 0000 | 0000 | 0001 |
|  |  | 1100 | 1111 | 1111 | 1110 |
|  | 移位3 | 0110 | 0111 | 1111 | 11100 |
|  | 移位4 | 0011 | 0011 | 1111 | 11111 |
|  |  | 1010 | 0000 | 0000 | 0001 |
|  |  | 1001 | 0011 | 1111 | 1110 |
|  | 移位5 | 0100 | 1001 | 1111 | 11110 |
|  | 移位6 | 0010 | 0100 | 1111 | 11111 |
|  |  | 1010 | 0000 | 0000 | 0001 |
|  |  | 1000 | 0100 | 1111 | 1110 |
|  | 移位7 | 0100 | 0010 | 0111 | 11110 |
|  | 移位8 | 0010 | 0001 | 0011 | 11110 |
|  |  | 1010 | 0000 | 0000 | 0001 |
|  |  |  |  |  |  |
|  |  | 1000 | 0001 | 0011 | 1110 |
|  |  | 0000 | 0000 | 0000 | 0111 |
| XOR 第二个字符 |  | 1000 | 0001 | 0011 | 1001 |
|  | 移位1 | 0100 | 0000 | 1001 | 11001 |
|  |  | 1010 | 0000 | 0000 | 0001 |
|  |  | 1110 | 0000 | 1001 | 1101 |
|  | 移位2 | 0111 | 0000 | 0100 | 11101 |
|  |  | 1010 | 0000 | 0000 | 0001 |
|  |  | 1101 | 0000 | 0100 | 1111 |
|  | 移位3 | 0110 | 1000 | 0010 | 01111 |
|  |  | 1010 | 0000 | 0000 | 0001 |
|  |  | 1100 | 1000 | 0010 | 0110 |
|  | 移位4 | 0110 | 0100 | 0001 | 00110 |
|  | 移位5 | 0011 | 0010 | 0000 | 10011 |
|  |  | 1010 | 0000 | 0000 | 0001 |
|  |  | 1001 | 0010 | 0000 | 1000 |
|  | 移位6 | 0100 | 1001 | 0000 | 01000 |
|  | 移位7 | 0010 | 0100 | 1000 | 00100 |
|  | 移位8 | 0001 | 0010 | 0100 | 00010 |

|  |  |  |
| --- | --- | --- |
|  |  |  |

最高有效 最低有效

帧的 CRC 16 则为: 4112

例

*执行 CRC 生成的C 语言的函数在下面示出。所有的可能的CRC 值都被预装在两个数组中，当计算报文内容时可以简单的索引即可。一个数组含有16位CRC 域的所有256个可能的高位字节，另一个数组含有地位字节的值。*

这种索引访问CRC 的方式提供了比对报文缓冲区的每个新字符都计算新的CRC 更快的方法。

注意: 此函数内部执行高/低 CRC 字节的交换。此函数返回的是已经经过交换的 CRC 值。

也就是说，从该函数返回的 CRC 值可以直接放置于报文用于发送。

函数使用两个参数:

unsigned char \*puchMsg; 指向含有用于生成 CRC 的二进制数据报文缓冲区的指针

unsigned short usDataLen; 报文缓冲区的字节数.

CRC 生成函数

unsigned short CRC16 ( puchMsg， usDataLen ) /\* 函数以 unsigned short 类型返回 CRC \*/

unsigned char \*puchMsg ; /\* 用于计算 CRC 的报文 \*/

unsigned short usDataLen ; /\* 报文中的字节数 \*/

{

unsigned char uchCRCHi = 0xFF ; /\* CRC 的高字节初始化 \*/

unsigned char uchCRCLo = 0xFF ; /\* CRC 的低字节初始化 \*/

unsigned uIndex ; /\* CRC 查询表索引 \*/

**while (usDataLen--) /\* 完成整个报文缓冲区 \*/**

{

uIndex = uchCRCLo ^ \*puchMsgg++ ; /\* 计算 CRC \*/

uchCRCLo = uchCRCHi ^ auchCRCHi[uIndex} ;

uchCRCHi = auchCRCLo[uIndex] ;

**}**

return (uchCRCHi << 8 | uchCRCLo) ;

}

高字节表

/\* 高位字节的CRC 值 \*/

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, 0xC1,0x81,0x40,0x01,0xC0,0x800x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81, 0x40

} ;

低字节表

/\* 低位字节的CRC 值 \*/

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

};

# 附录B —— MODBUS地址表

## B.1 室内机组状态-线圈(功能码：01)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 地址 | 参数 | 备注 | 默认值 | 最小值 | 最大值 | 精度 | 单位 |
| 0X8001 | 开/关状态 | 0：关机；1：开机 |  | 0 | 1 |  |  |
| 0X8002 | 保留 |  |  | 0 | 1 |  |  |
| 0X8003 | 保留 |  |  | 0 | 1 |  |  |
| 0X8004 | 保留 |  |  | 0 | 1 |  |  |
| 0X8005 | 保留 |  |  | 0 | 1 |  |  |
| 0X8006 | 保留 |  |  | 0 | 1 |  |  |
| 0X8007 | 保留 |  |  | 0 | 1 |  |  |
| 0X8008 | 保留 |  |  | 0 | 1 |  |  |
| 0X8009 | 保留 |  |  | 0 | 1 |  |  |
| 0X8010 | 远程开关 | 0：停止；1：开启 |  | 0 | 1 |  |  |
| 0X8011 | 保留 |  |  | 0 | 1 |  |  |
| 0X8012 | 机械/节能制冷 | 0：机械制冷；1：节能制冷 |  | 0 | 1 |  |  |
| 0X8013 | 保留 |  |  | 0 | 1 |  |  |
| 0X8014 | 保留 |  |  | 0 | 1 |  |  |
| 0X8015 | 保留 |  |  | 0 | 1 |  |  |
| 0X8016 | 保留 |  |  | 0 | 1 |  |  |
| 0X8017 | 电源故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8018 | 相序错误 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8019 | 缺相错误 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8020 | 电加热保护 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8021 | 加湿器报警 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8022 | 烟雾报警 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8023 | 水浸报警 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8024 | 湿膜低水位报警 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8025 | 湿膜水位开关故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8026 | EEPROM故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8027 | 远程通讯故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8028 | 手操器通讯故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8029 | 本地联网通讯故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8030 | 扩展板通讯故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8031 | 远程内温探头故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8032 | 滤网堵塞报警 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8033 | 风机1转速反馈故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8034 | 风机2转速反馈故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8035 | 风机3转速反馈故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8036 | 风机4转速反馈故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8037 | 风机5转速反馈故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8038 | 风机6转速反馈故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8039 | 风机7转速反馈故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8040 | 风机8转速反馈故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8041 | 风机过载故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8042 | 气流丢失故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8043 | 高温报警 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8044 | 低温报警 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8045 | 高湿报警 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8046 | 低湿报警 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8047 | 送风高温报警 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8048 | 送风低温报警 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8049 | 回风温度探头1故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8050 | 回风温度探头2故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8051 | 回风温度探头3故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8052 | 回风湿度探头故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8053 | 送风温度1探头故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8054 | 送风温度2探头故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8055 | 送风温度3探头故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8056 | 送风湿度探头故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8057 | 保留 |  |  | 0 | 1 |  |  |
| 0X8058 | 保留 |  |  | 0 | 1 |  |  |
| 0X8059 | 保留 |  |  | 0 | 1 |  |  |
| 0X8060 | 保留 |  |  | 0 | 1 |  |  |
| 0X8061 | 保留 |  |  | 0 | 1 |  |  |
| 0X8062 | 保留 |  |  | 0 | 1 |  |  |
| 0X8063 | 保留 |  |  | 0 | 1 |  |  |
| 0X8064 | 保留 |  |  | 0 | 1 |  |  |
| 0X8065 | 系统1高压保护 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8066 | 系统1低压保护 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8067 | 高压开关1故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8068 | 低压开关1故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8069 | 内管温1探头故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8070 | 排气温度1探头故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8071 | 系统1蒸发温度过低 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8072 | 系统1排气高温 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8073 | 系统1变频驱动故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8074 | 系统1变频驱动通讯故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8075 | 系统1压力传感器故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8076 | 保留 |  |  | 0 | 1 |  |  |
| 0X8077 | 保留 |  |  | 0 | 1 |  |  |
| 0X8078 | 保留 |  |  | 0 | 1 |  |  |
| 0X8079 | 压缩机1 | 0：停止；1：开启 |  | 0 | 1 |  |  |
| 0X8080 | 保留 |  |  | 0 | 1 |  |  |
| 0X8081 | 系统2高压保护 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8082 | 系统2低压保护 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8083 | 高压开关2故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8084 | 低压开关2故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8085 | 内管温2探头故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8086 | 排气温度2探头故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8087 | 系统2蒸发温度过低 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8088 | 系统2排气高温 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8089 | 系统2变频驱动故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8090 | 系统2变频驱动通讯故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8091 | 系统2压力传感器故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8092 | 保留 |  |  | 0 | 1 |  |  |
| 0X8093 | 保留 |  |  | 0 | 1 |  |  |
| 0X8094 | 保留 |  |  | 0 | 1 |  |  |
| 0X8095 | 压缩机2 | 0：停止；1：开启 |  | 0 | 1 |  |  |
| 0X8096 | 保留 |  |  | 0 | 1 |  |  |

## B.2 室内机组状态-寄存器(功能码: 03)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 地址 | 参数 | 备注 | 默认值 | 最小值 | 最大值 | 精度 | 单位 |
| 4X8001 | 系统运行模式 | 0：停机  B0：制冷；  B1：制热；  B2：加湿；  B3：除湿；  B4：待机；(1：待机)  B5：故障停机（0：无停机故障；1：有停机故障）  B6：手动关机（远程关机/线控器）  B7：主/备（0：主；1：备）  B8：断网  B9：干接点关机 |  | 0 |  |  |  |
| 4X8002 | 烟雾报警 |  |  | 0 | 1 |  |  |
| 4X8003 | 送风温度注1 | =实际值\*10 |  | -300 | 700 | 0.1 | ℃ |
| 4X8004 | 室内温度注1 | =实际值\*10 |  | -300 | 700 | 0.1 | ℃ |
| 4X8005 | 回风温度注1 | =实际值\*10 |  | -300 | 700 | 0.1 | ℃ |
| 4X8006 | 回风湿度注1 | =实际值\*10 |  | 100 | 950 | 0.1 | % |
| 4X8007 | 机组累计运行时间 |  |  | 0 | 65535 | 1 | 小时 |
| 4X8008 | 保留 |  |  |  |  |  |  |
| 4X8009 | 保留 |  |  |  |  |  |  |
| 4X8010 | 保留 |  |  |  |  |  |  |
| 4X8011 | 保留 |  |  |  |  |  |  |
| 4X8012 | 保留 |  |  |  |  |  |  |
| 4X8013 | 保留 |  |  | 0 | 300 | 1 | V |
| 4X8014 | 电源电压 |  |  | 0 | 100 | 1 | % |
| 4X8015 | 送风湿度注1 | =实际值\*10 |  | 100 | 950 | 0.1 | % |
| 4X8016 | 保留 |  |  |  |  |  |  |
| 4X8017 | 保留 |  |  |  |  |  |  |
| 4X8018 | 保留 |  |  |  |  |  |  |
| 4X8019 | 保留 |  |  |  |  |  |  |
| 4X8020 | 保留 |  |  |  |  |  |  |
| 4X8021 | 保留 |  |  |  |  |  |  |
| 4X8022 | 保留 |  |  |  |  |  |  |
| 4X8023 | 保留 |  |  |  |  |  |  |
| 4X8024 | 保留 |  |  |  |  |  |  |
| 4X8025 | 平均温度注1 | =实际值\*10 |  | -300 | 700 | 0.1 | ℃ |
| 4X8026 | 平均湿度注1 | =实际值\*10 |  | 100 | 950 | 0.1 | % |
| 4X8027 | 内风机风速 | =0:停止；>0:开启 |  | 0 | 100 | 1 |  |
| 4X8028 | 保留 |  |  |  |  |  |  |
| 4X8029 | 保留 |  |  |  |  |  |  |
| 4X8030 | 保留 |  |  |  |  |  |  |
| 4X8031 | 保留 |  |  |  |  |  |  |
| 4X8032 | 保留 |  |  |  |  |  |  |
| 4X8033 | 保留 |  |  |  |  |  |  |
| 4X8034 | 保留 |  |  |  |  |  |  |
| 4X8035 | 保留 |  |  |  |  |  |  |
| 4X8036 | 保留 |  |  |  |  |  |  |
| 4X8037 | 保留 |  |  |  |  |  |  |
| 4X8038 | 保留 |  |  |  |  |  |  |
| 4X8039 | 保留 |  |  |  |  |  |  |
| 4X8040 | 保留 |  |  |  |  |  |  |
| 4X8041 | 保留 |  |  |  |  |  |  |
| 4X8042 | 压机1运行频率 |  |  | 0 |  | 1 | Hz |
| 4X8043 | 保留 |  |  |  |  |  |  |
| 4X8044 | 保留 |  |  |  |  |  |  |
| 4X8045 | 保留 |  |  |  |  |  |  |
| 4X8046 | 保留 |  |  |  |  |  |  |
| 4X8047 | 保留 |  |  |  |  |  |  |
| 4X8048 | 保留 |  |  |  |  |  |  |
| 4X8049 | 保留 |  |  |  |  |  |  |
| 4X8050 | 保留 |  |  |  |  |  |  |
| 4X8051 | 压机2运行频率 |  |  | 0 |  | 1 | Hz |
| 4X8052 | 保留 |  |  |  |  |  |  |
| 4X8053 | 保留 |  |  |  |  |  |  |
| 4X8054 | 保留 |  |  |  |  |  |  |
| 4X8055 | 保留 |  |  |  |  |  |  |

注1：-32768表示传感器故障；-32767表示还未检测到有效值

## B.3 室外机1状态-线圈(功能码：01)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 地址 | 参数 | 备注 | 默认值 | 最小值 | 最大值 | 精度 | 单位 |
| 0X8101 | 保留 |  |  |  |  |  |  |
| 0X8102 | 水流开关 | 0：关闭；1：开启 |  | 0 | 1 |  |  |
| 0X8103 | 保留 |  |  |  |  |  |  |
| 0X8104 | 变频驱动通讯故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8105 | 室外温度传感器故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8106 | 压力传感器故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8107 | 系统高压保护 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8108 | 系统低压保护 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8109 | 保留 |  |  |  |  |  |  |
| 0X8110 | 保留 |  |  |  |  |  |  |
| 0X8111 | 保留 |  |  |  |  |  |  |
| 0X8112 | 变频驱动故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8113 | 风机过热保护 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8114 | 保留 |  |  |  |  |  |  |
| 0X8115 | 保留 |  |  |  |  |  |  |
| 0X8116 | 保留 |  |  |  |  |  |  |
| 0X8117 | 保留 |  |  |  |  |  |  |
| 0X8118 | 保留 |  |  |  |  |  |  |
| 0X8119 | 保留 |  |  |  |  |  |  |
| 0X8120 | 保留 |  |  |  |  |  |  |
| 0X8121 | 保留 |  |  |  |  |  |  |
| 0X8122 | 保留 |  |  |  |  |  |  |
| 0X8123 | 保留 |  |  |  |  |  |  |
| 0X8124 | 保留 |  |  |  |  |  |  |
| 0X8125 | 保留 |  |  |  |  |  |  |
| 0X8126 | 保留 |  |  |  |  |  |  |
| 0X8127 | 保留 |  |  |  |  |  |  |
| 0X8128 | 保留 |  |  |  |  |  |  |
| 0X8129 | 保留 |  |  |  |  |  |  |
| 0X8130 | 保留 |  |  |  |  |  |  |
| 0X8131 | 保留 |  |  |  |  |  |  |
| 0X8132 | 与室内机通讯故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |

## B.4 室外机1状态-寄存器(功能码: 03)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 地址 | 参数 | 备注 | 默认值 | 最小值 | 最大值 | 精度 | 单位 |
| 4X8101 | 出水水温注1 | 水冷有效 |  | -300 | 700 | 0.1 | ℃ |
| 4X8102 | 外风机转速 | =0:停止；>0:开启 |  | 0 | 50 | 1 | HZ |
| 4X8103 | 风冷/风串水：室外温度注1 | =实际值\*10 |  | -300 | 700 | 0.1 | ℃ |
| 4X8104 | 水冷/风串水：回水水温注1 | =实际值\*10 |  | -300 | 700 | 0.1 | ℃ |
| 4X8105 | 保留 |  |  |  |  |  |  |
| 4X8106 | 比例水阀开度 | 水冷有效 |  | 0 | 100 | 1 | % |

注1：-32768表示传感器故障；-32767表示还未检测到有效值

## B.5 室外机2状态-线圈(功能码：01)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 地址 | 参数 | 备注 | 默认值 | 最小值 | 最大值 | 精度 | 单位 |
| 0X8133 | 保留 |  |  |  |  |  |  |
| 0X8134 | 水流开关 | 0：关闭；1：开启 |  | 0 | 1 |  |  |
| 0X8135 | 保留 |  |  |  |  |  |  |
| 0X8136 | 变频驱动通讯故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8137 | 室外温度传感器故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8138 | 压力传感器故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8139 | 系统高压保护 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8140 | 系统低压保护 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8141 | 保留 |  |  |  |  |  |  |
| 0X8142 | 保留 |  |  |  |  |  |  |
| 0X8143 | 保留 |  |  |  |  |  |  |
| 0X8144 | 变频驱动故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8145 | 风机过热保护 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8146 | 保留 |  |  |  |  |  |  |
| 0X8147 | 保留 |  |  |  |  |  |  |
| 0X8148 | 保留 |  |  |  |  |  |  |
| 0X8149 | 保留 |  |  |  |  |  |  |
| 0X8150 | 保留 |  |  |  |  |  |  |
| 0X8151 | 保留 |  |  |  |  |  |  |
| 0X8152 | 保留 |  |  |  |  |  |  |
| 0X8153 | 保留 |  |  |  |  |  |  |
| 0X8154 | 保留 |  |  |  |  |  |  |
| 0X8155 | 保留 |  |  |  |  |  |  |
| 0X8156 | 保留 |  |  |  |  |  |  |
| 0X8157 | 保留 |  |  |  |  |  |  |
| 0X8158 | 保留 |  |  |  |  |  |  |
| 0X8159 | 保留 |  |  |  |  |  |  |
| 0X8160 | 保留 |  |  |  |  |  |  |
| 0X8161 | 保留 |  |  |  |  |  |  |
| 0X8162 | 保留 |  |  |  |  |  |  |
| 0X8163 | 保留 |  |  |  |  |  |  |
| 0X8164 | 与室内机通讯故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |

## B.6 室外机2状态-寄存器(功能码: 03)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 地址 | 参数 | 备注 | 默认值 | 最小值 | 最大值 | 精度 | 单位 |
| 4X8109 | 出水水温注1 | 水冷有效 |  | -300 | 700 | 0.1 | ℃ |
| 4X8110 | 外风机转速 | =0:停止；>0:开启 |  | 0 | 50 | 1 | HZ |
| 4X8111 | 风冷/风串水：室外温度注1 | =实际值\*10 |  | -300 | 700 | 0.1 | ℃ |
| 4X8112 | 水冷/风串水：回水水温注1 | =实际值\*10 |  | -300 | 700 | 0.1 | ℃ |
| 4X8113 | 保留 |  |  |  |  |  |  |
| 4X8114 | 比例水阀开度 | 水冷有效 |  | 0 | 100 | 1 | % |

注1：-32768表示传感器故障；-32767表示还未检测到有效值

## B.7 泵柜1状态-线圈(功能码：01)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 地址 | 参数 | 备注 | 默认值 | 最小值 | 最大值 | 精度 | 单位 |
| 0X8201 | 氟泵 | 0：关闭；1：开启 |  | 0 | 1 |  |  |
| 0X8202 | 保留 |  |  |  |  |  |  |
| 0X8203 | 保留 |  |  |  |  |  |  |
| 0X8204 | 保留 |  |  |  |  |  |  |
| 0X8205 | 保留 |  |  |  |  |  |  |
| 0X8206 | 保留 |  |  |  |  |  |  |
| 0X8207 | 保留 |  |  |  |  |  |  |
| 0X8208 | 保留 |  |  |  |  |  |  |
| 0X8209 | 保留 |  |  |  |  |  |  |
| 0X8210 | 保留 |  |  |  |  |  |  |
| 0X8211 | 保留 |  |  |  |  |  |  |
| 0X8212 | 保留 |  |  |  |  |  |  |
| 0X8213 | 保留 |  |  |  |  |  |  |
| 0X8214 | 氟泵高扬程锁定告警 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8215 | 氟泵低扬程锁定告警 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8216 | 氟泵流量不足锁定告警 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8217 | 氟泵进口温感故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8218 | 氟泵出口温感故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8219 | 氟泵进口压力传感器故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8220 | 氟泵出口压力传感器故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8221 | 相序错误 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8222 | 氟泵进口过冷度过小锁定告警 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8223 | 氟泵出口温度过低锁定告警 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8224 | EEPROM故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8225 | 保留 |  |  |  |  |  |  |
| 0X8226 | 保留 |  |  |  |  |  |  |
| 0X8227 | 保留 |  |  |  |  |  |  |
| 0X8228 | 保留 |  |  |  |  |  |  |
| 0X8229 | 保留 |  |  |  |  |  |  |
| 0X8230 | 保留 |  |  |  |  |  |  |
| 0X8231 | 保留 |  |  |  |  |  |  |
| 0X8232 | 与室内机通讯故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |

## B.8 泵柜2状态-线圈(功能码：01)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 地址 | 参数 | 备注 | 默认值 | 最小值 | 最大值 | 精度 | 单位 |
| 0X8233 | 氟泵 | 0：关闭；1：开启 |  | 0 | 1 |  |  |
| 0X8234 | 保留 |  |  |  |  |  |  |
| 0X8235 | 保留 |  |  |  |  |  |  |
| 0X8236 | 保留 |  |  |  |  |  |  |
| 0X8237 | 保留 |  |  |  |  |  |  |
| 0X8238 | 保留 |  |  |  |  |  |  |
| 0X8239 | 保留 |  |  |  |  |  |  |
| 0X8240 | 保留 |  |  |  |  |  |  |
| 0X8241 | 保留 |  |  |  |  |  |  |
| 0X8242 | 保留 |  |  |  |  |  |  |
| 0X8243 | 保留 |  |  |  |  |  |  |
| 0X8244 | 保留 |  |  |  |  |  |  |
| 0X8245 | 保留 |  |  |  |  |  |  |
| 0X8246 | 氟泵高扬程锁定告警 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8247 | 氟泵低扬程锁定告警 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8248 | 氟泵流量不足锁定告警 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8249 | 氟泵进口温感故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8250 | 氟泵出口温感故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8251 | 氟泵进口压力传感器故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8252 | 氟泵出口压力传感器故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8253 | 相序错误 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8254 | 氟泵进口过冷度过小锁定告警 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8255 | 氟泵出口温度过低锁定告警 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8256 | EEPROM故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |
| 0X8257 | 保留 |  |  |  |  |  |  |
| 0X8258 | 保留 |  |  |  |  |  |  |
| 0X8259 | 保留 |  |  |  |  |  |  |
| 0X8260 | 保留 |  |  |  |  |  |  |
| 0X8261 | 保留 |  |  |  |  |  |  |
| 0X8262 | 保留 |  |  |  |  |  |  |
| 0X8263 | 保留 |  |  |  |  |  |  |
| 0X8264 | 与室内机通讯故障 | 0：正常；1：故障 |  | 0 | 1 |  |  |

## B.9 参数设定(功能码：03, 06，16)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 地址 | 参数 | 备注 | 默认值 | 最小值 | 最大值 | 精度 | 单位 |
| 4X3501 | 回风温度设定值 | =实际值\*10 | 240 | 100 | 400 | 0.1 | ℃ |
| 4X3502 | 出风温度设定值 | =实际值\*10 | 180 | 100 | 400 | 0.1 | ℃ |
| 4X3503 | 远程温度设定值 | =实际值\*10 | 240 | 100 | 400 | 0.1 | ℃ |
| 4X3504 | 设定湿度 | =实际值\*10 | 500 | 200 | 800 | 0.1 | % |
| 4X3505 | 高温报警 | =实际值\*10 | 320 | 100 | 500 | 0.1 | ℃ |
| 4X3506 | 低温报警 | =实际值\*10 | 150 | 0 | 300 | 0.1 | ℃ |
| 4X3507 | 高湿报警 | =实际值\*10 | 700 | 100 | 950 | 0.1 | % |
| 4X3508 | 低湿报警 | =实际值\*10 | 300 | 100 | 950 | 0.1 | % |
| 4X3509 | 开/关机 | 0：关机；1：开机 | 1 | 0 | 1 |  |  |
| 4X3510 | 本地操作 | 0：允许；1：禁止 | 0 | 0 | 1 |  |  |
| 4X3511 | 用户密码 |  | 0 | 0 | 9999 |  |  |
| 4X3512 | 启动延时 |  | 2 | 0 | 240 | 1 | 秒 |