				功能码			
				码	子码	（十六进制）	页
数据访问	比特访问	物理离散量输入	读输入离散量	02		02	11
		内部比特或物理线圈	读线圈	01		01	10
			写单个线圈	05		05	16
			写多个线圈	15		0F	37
	16 比特访问	输入存储器	读输入寄存器	04		04	14
		内部存储器或物理输出存储器	读多个寄存器	03		03	13
			写单个寄存器	06		06	17
			写多个寄存器	16		10	39
			读/写多个寄存器	23		17	47
			屏蔽写寄存器	22		16	46
	文件记录访问		读文件记录	20	6	14	42
			写文件记录	21	6	15	44
封装接口			读设备识别码	43	14	2B	

一、MODBUS通讯协议完整介绍（全文10000字以上）

曾小庆

愿一切的美好都如期而至！！！

相关专栏：

串口传输，通信协议

www.zhihu.com/column/c_125154259226...

1、引言

1.1 范围

MODBUS 是 OSI 模型第 7 层上的应用层报文传输协议，它在连接至不同类型总线或网络的设备之间提供客户机/服务器通信。

自从 1979 年出现工业串行链路的事实标准以来，MODBUS 使成千上万的自动化设备能够通信。目前，继续增加对简单而雅观的 MODBUS 结构支持。互联网组织能够使 TCP/IP 栈上的保留系统端口502 访问 MODBUS。

MODBUS 是一个请求/应答协议，并且提供功能码规定的服务。MODBUS 功能码是 MODBUS 请求/应答 PDU 的元素。本文件的作用是描述 MODBUS 事务处理框架内使用的功能码。

1.2 规范性引用文件

1．RFC791，互联网协议，Sep81 DARPA

2．MODBUS 协议参考指南 Rev J,MODICON，1996 年 6 月，doc#PI_MBUS_300

MODBUS 是一项应用层报文传输协议，用于在通过不同类型的总线或网络连接的设备之间的客户机/服务器通信。

目前，使用下列情况实现 MODBUS：

以太网上的 TCP/IP。

各种媒体（有线：EIA/TIA-232-E、EIA-422、EIA/TIA-485-A；光纤、无线等等）上的异步串行传输。

MODBUS PLUS，一种高速令牌传递网络。

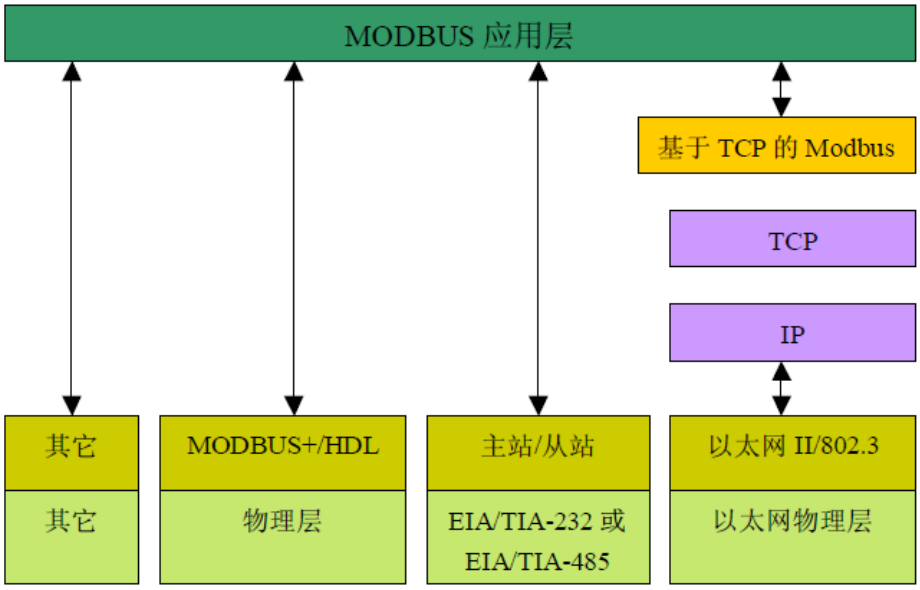

图 1：MODBUS 通信栈

2、缩略语

ADU 应用数据单

HDLC 高级数据链路控制

HMI 人机界面

IETF 因特网工程工作组

I/O 输入/输出设备

IP 互连网协议

MAC 介质访问控制

MB MODBUS 协议

MBAP MODBUS 协议

PDU 协议数据单元

PLC 可编程逻辑控制器

TCP 传输控制协议

3、背景概要

MODBUS 协议允许在各种网络体系结构内进行简单通信。

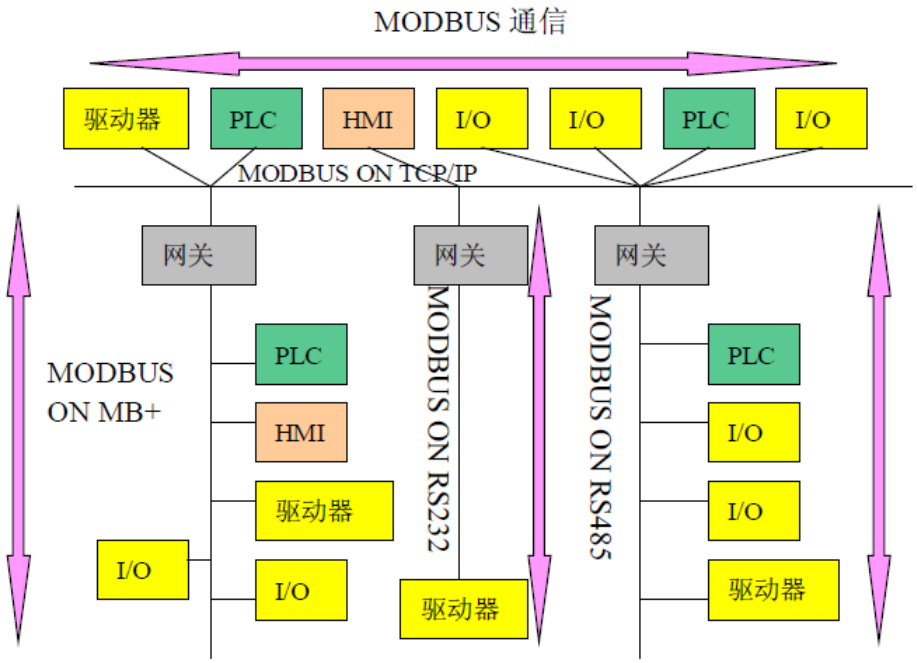

图 2：MODBUS 网络体系结构的实例

每种设备（PLC、HMI、控制面板、驱动程序、动作控制、输入/输出设备）都能使用 MODBUS协议来启动远程操作。

在基于串行链路和以太 TCP/IP 网络的 MODBUS 上可以进行相同通信。

一些网关允许在几种使用 MODBUS 协议的总线或网络之间进行通信。

4、总体描述

4.1 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。

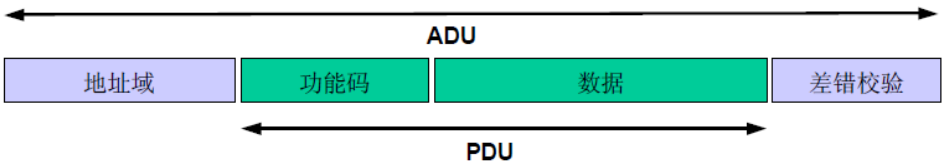

图 3：通用 MODBUS 帧

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。

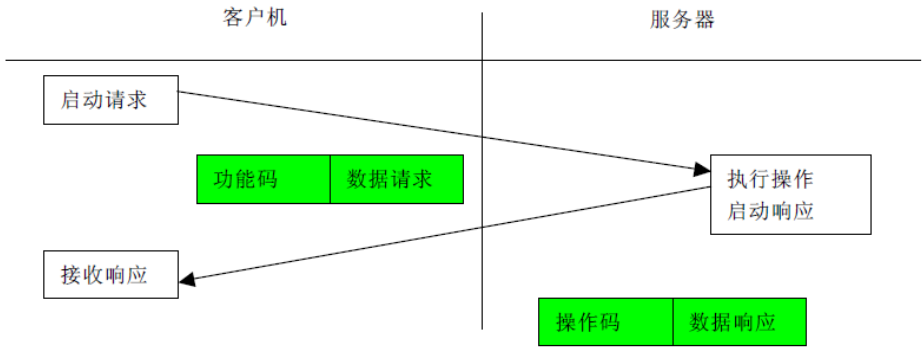

图 4：MODBUS 事务处理（无差错）

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。

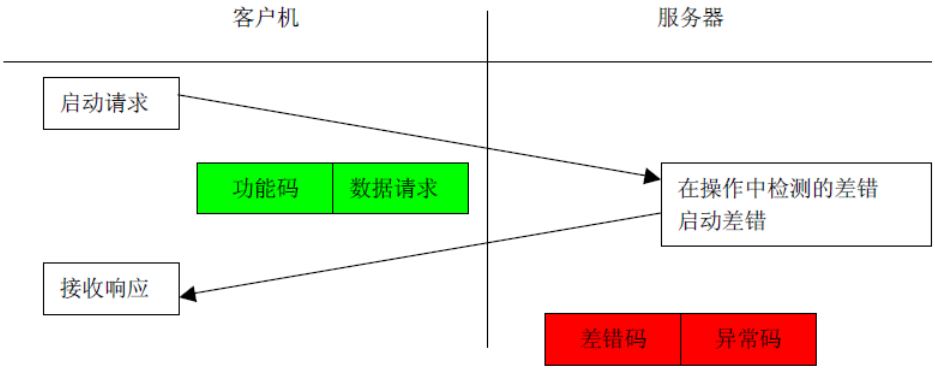

图5 MODBUS事务处理（异常响应）

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。

从而：

RS232 / RS485 ADU = 253 字节+服务器地址(1字节) + CRC (2 字节) = 256 字节。

TCP MODBUS ADU = 249 字节+ MBAP (7 字节) = 256 字节。

MODBUS 协议定义了三种 PDU。它们是：

- MODBUS 请求 PDU，mb_req_pdu
- MODBUS 响应 PDU，mb_rsp_pdu
- MODBUS 异常响应 PDU，mb_excep_rsp_pdu
- 定义 mb_req_pdu 为：

mb_req_pdu = { function_code, request_data}，其中

function_code - [1 个字节] MODBUS 功能码

request_data - [n 个字节]，这个域与功能码有关，并且通常包括诸如可变参考、变量、数据偏移量、子功能码等信息。

- 定义 mb_rsp_pdu 为：

mb_rsp_pdu = { function_code, response_ data}，其中

function_code - [1 个字节] MODBUS 功能码

response_data - [n 个字节]，这个域与功能码有关，并且通常包括诸如可变参考、变量、数据偏移量、子功能码等信息。

- 定义 mb_excep_rsp_pdu 为：

mb_excep_rsp_pdu = { function_code, request_data}，其中

function_code - [1 个字节] MODBUS 功能码 + 0x80

exception_code - [1 个字节]，在下表中定义了 MODBUS 异常码。

4.2 数据编码

MODBUS 使用一个‘big-Endian’ 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

4.3 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。

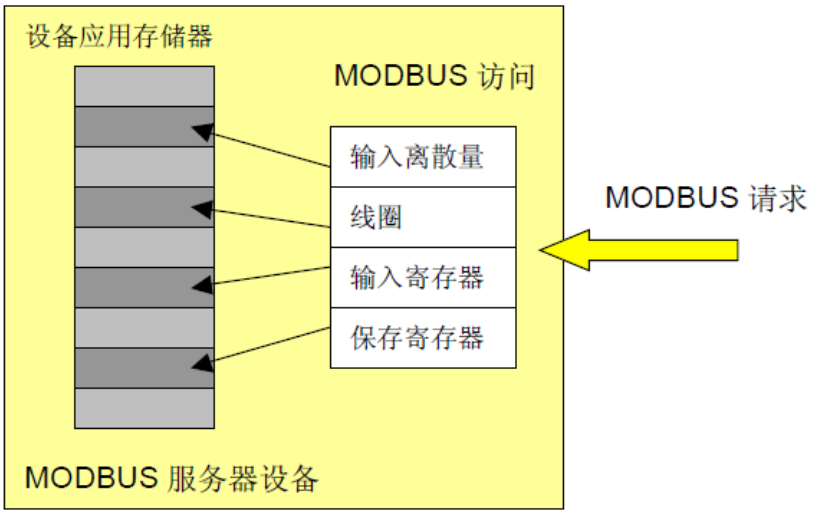

图 6：带有独立块的 MODBUS 数据模型

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。

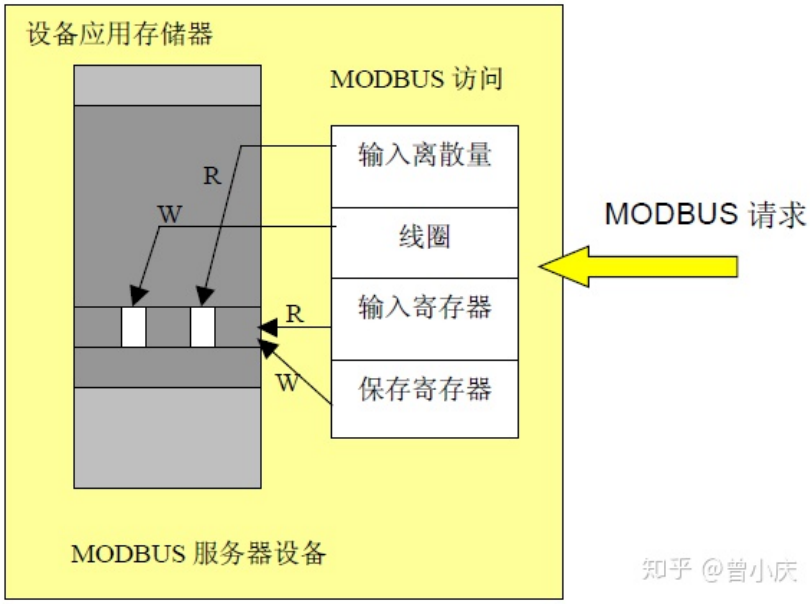

图 7：仅带有 1 个块的 MODBUS 数据模型

4.4 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。

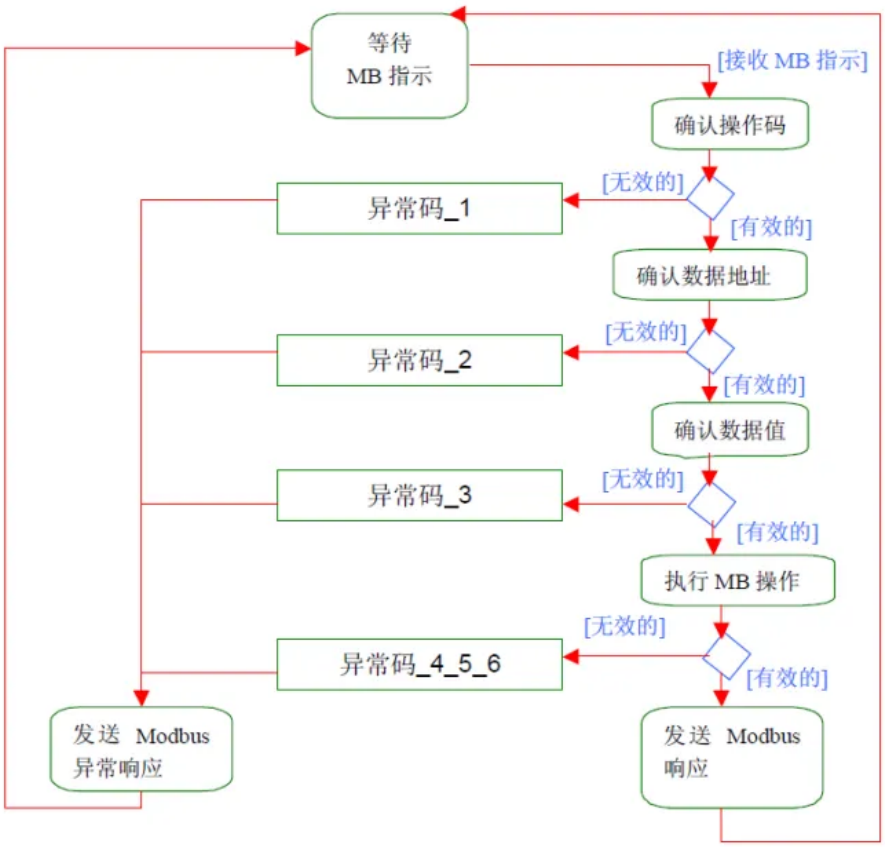

图 8：MODBUS 事务处理的状态图

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：

01、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；

02、响应功能码 = 请求功能码 + 0x80；

03、提供一个异常码来指示差错原因。

5、功能码分类

有三类 MODBUS 功能码。它们是：

公共功能码

- 是较好地被定义的功能码，
- 保证是唯一的，
- MODBUS 组织可改变的，
- 公开证明的，
- 具有可用的一致性测试，
- MB IETF RFC 中证明的，
- 包含已被定义的公共指配功能码和未来使用的未指配保留供功能码。

用户定义功能码

- 有两个用户定义功能码的定义范围，即 65 至 72 和十进制 100 至 110。
- 用户没有 MODBUS 组织的任何批准就可以选择和实现一个功能码
- 不能保证被选功能码的使用是唯一的。
- 如果用户要重新设置功能作为一个公共功能码，那么用户必须启动 RFC，以便将改变引入公共分类中，并且指配一个新的公共功能码。

保留功能码

- 一些公司对传统产品通常使用的功能码，并且对公共使用是无效的功能码。

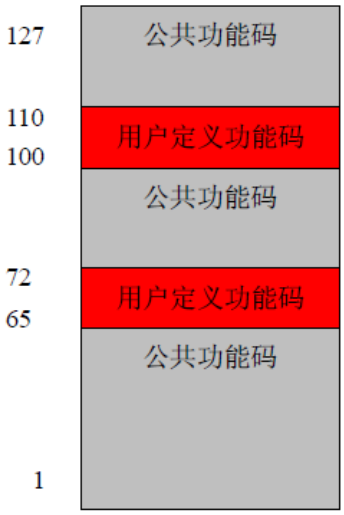

图 9：MODBUS 功能码分类

5.1 公共功能码定义

				功能码			
				码	子码	（十六进制）	页
数据访问	比特访问	物理离散量输入	读输入离散量	02		02	11
		内部比特或物理线圈	读线圈	01		01	10
			写单个线圈	05		05	16
			写多个线圈	15		0F	37
	16 比特访问	输入存储器	读输入寄存器	04		04	14
		内部存储器或物理输出存储器	读多个寄存器	03		03	13
			写单个寄存器	06		06	17
			写多个寄存器	16		10	39
			读/写多个寄存器	23		17	47
			屏蔽写寄存器	22		16	46
	文件记录访问		读文件记录	20	6	14	42
			写文件记录	21	6	15	44
封装接口			读设备识别码	43	14	2B	

6、功能码描述

6.1 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

差错码	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	01	功能	01
起始地址 Hi	00	字节数	03
起始地址 Lo	13	输出状态 27-20	CD
输出数量 Hi	00	输出状态 35-28	6B
输出数量 Lo	13	输出状态 38-36	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）。

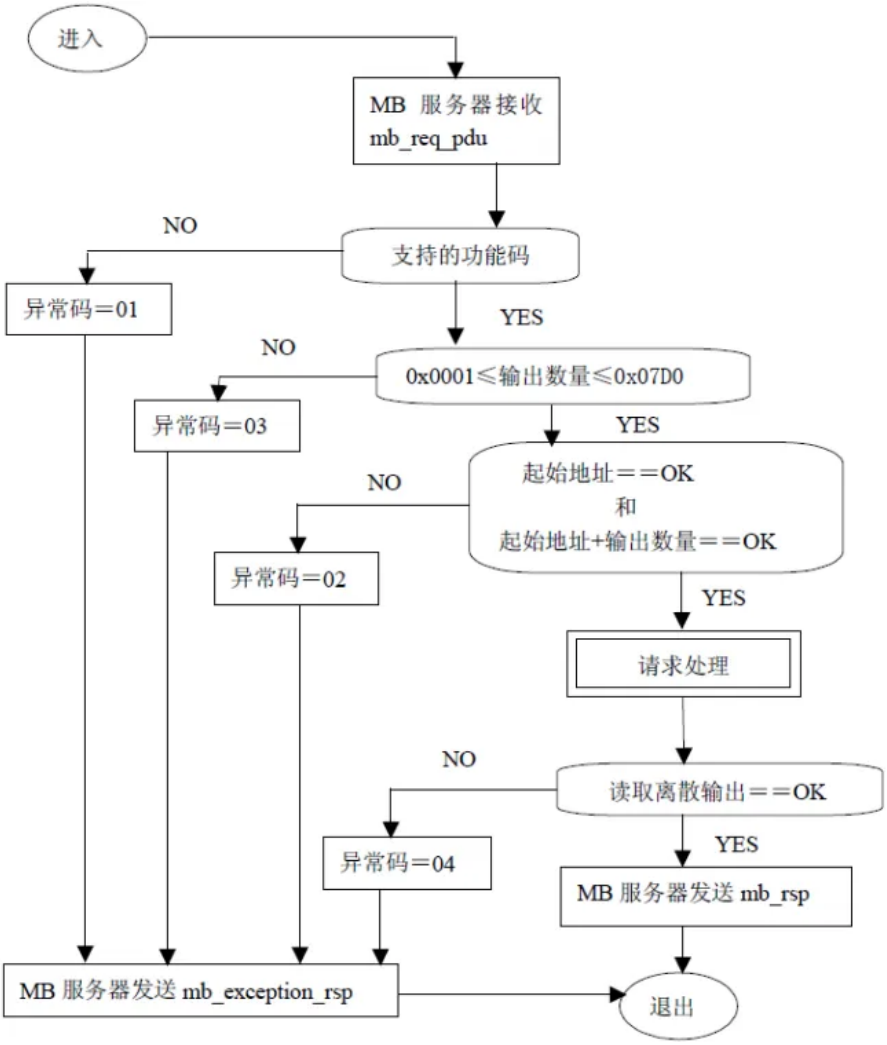

图 10：读取线圈状态图

6.2 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	**0x02**
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 个字节	0x82
异常码	1 个字节	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	02	功能	02
起始地址 Hi	00	字节数	03
起始地址 Lo	C4	输出状态 204-197	AC
输出数量 Hi	00	输出状态 212-205	DB
输出数量 Lo	16	输出状态 218-213	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）。

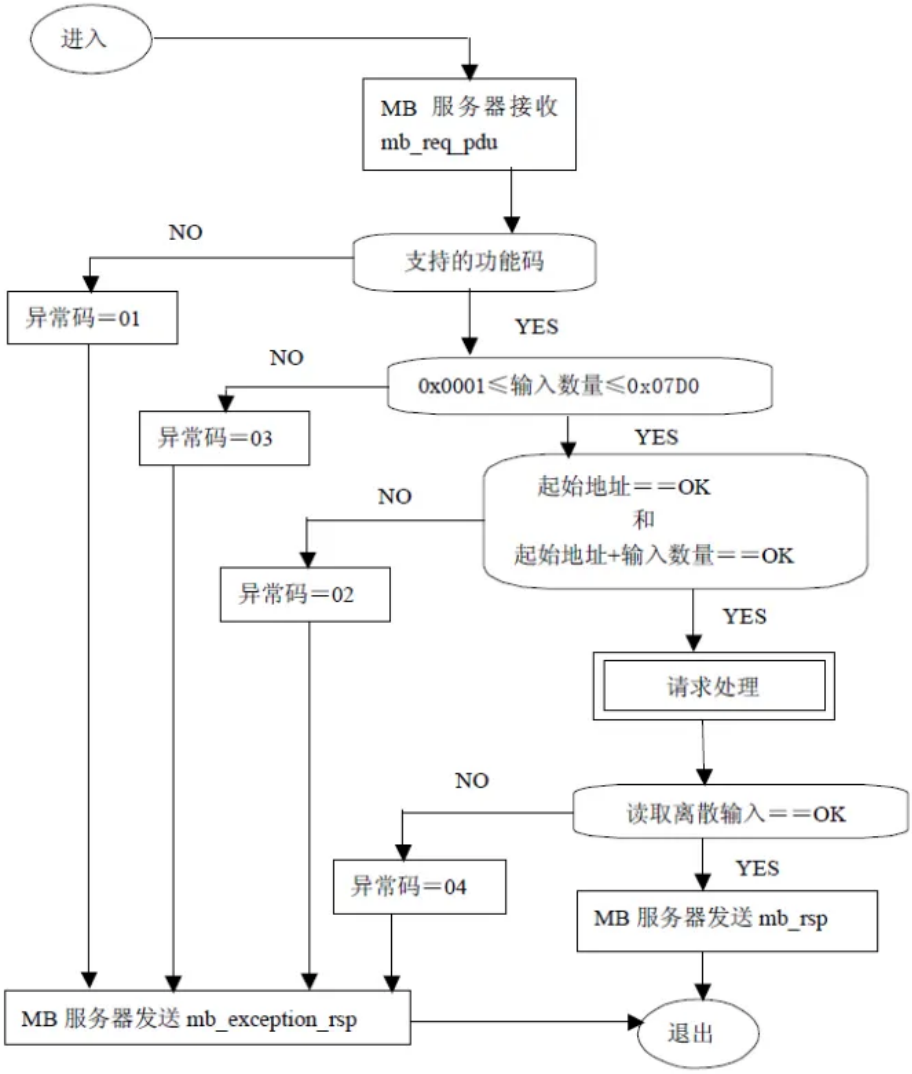

图 11：读离散量输入的状态图

6.3 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 PDU

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 PDU

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

这是一个请求读保持寄存器 108-110 的实例：

请求		响应	
域名	（十六进制）	域名	（十六进制）
功能	03	功能	03
起始地址 Hi	00	字节数	06
起始地址 Lo	6B	寄存器值 Hi（108）	02
寄存器编号 Hi	00	寄存器值 Lo（108）	2B
寄存器编号 Lo	03	寄存器值 Hi（109）	00
		寄存器值 Lo（109）	00
		寄存器值 Hi（110）	00
		寄存器值 Lo（110）	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。

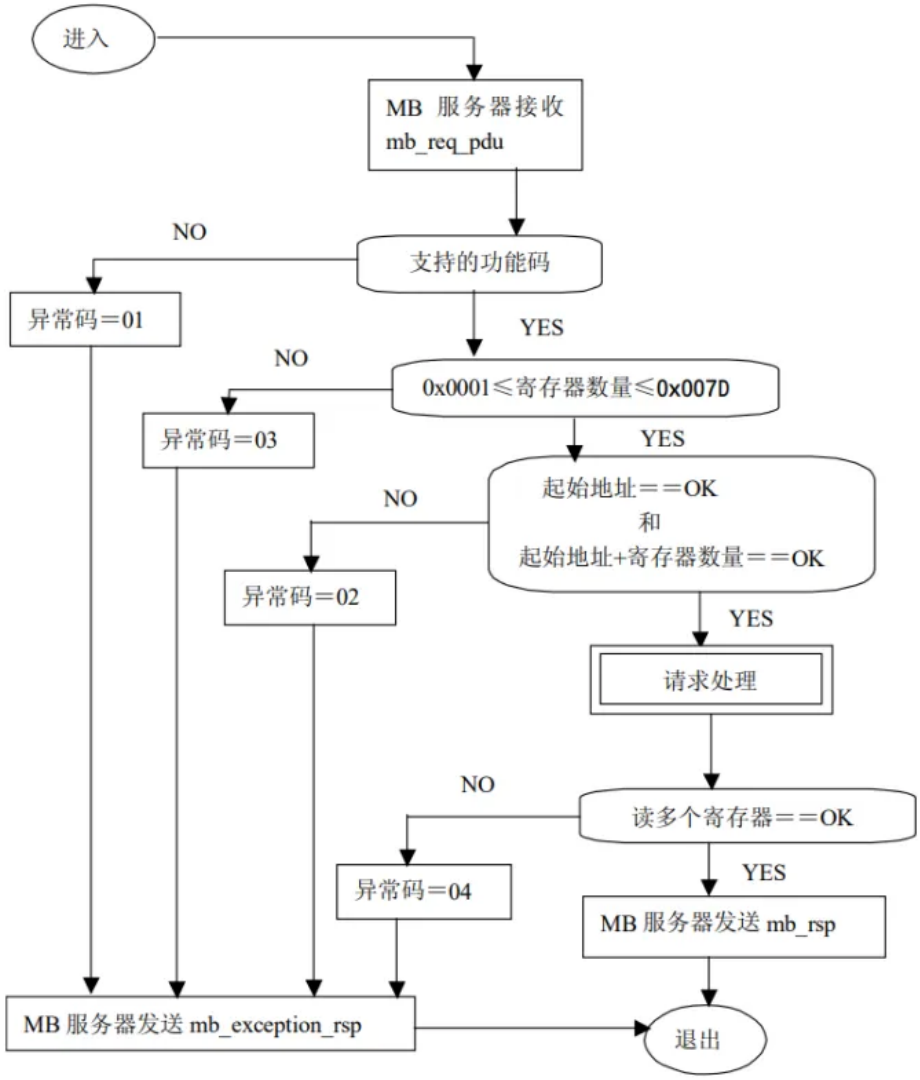

图 12：读保持寄存器的状态图

6.4 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 PDU

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

响应 PDU

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*＝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 09 的实例：

请求		响应	
域名	（十六进制）	域名	（十六进制）
功能	04	功能	04
起始地址 Hi	00	字节数	02
起始地址 Lo	08	输入寄存器 Hi（09）	00
输入寄存器数量 Hi	00	输入寄存器 Lo（09）	0A
输入寄存器数量 Lo	01		

将输入寄存器 09 的内容表示为两个十六进制字节值 00 0A，或十进制 10。

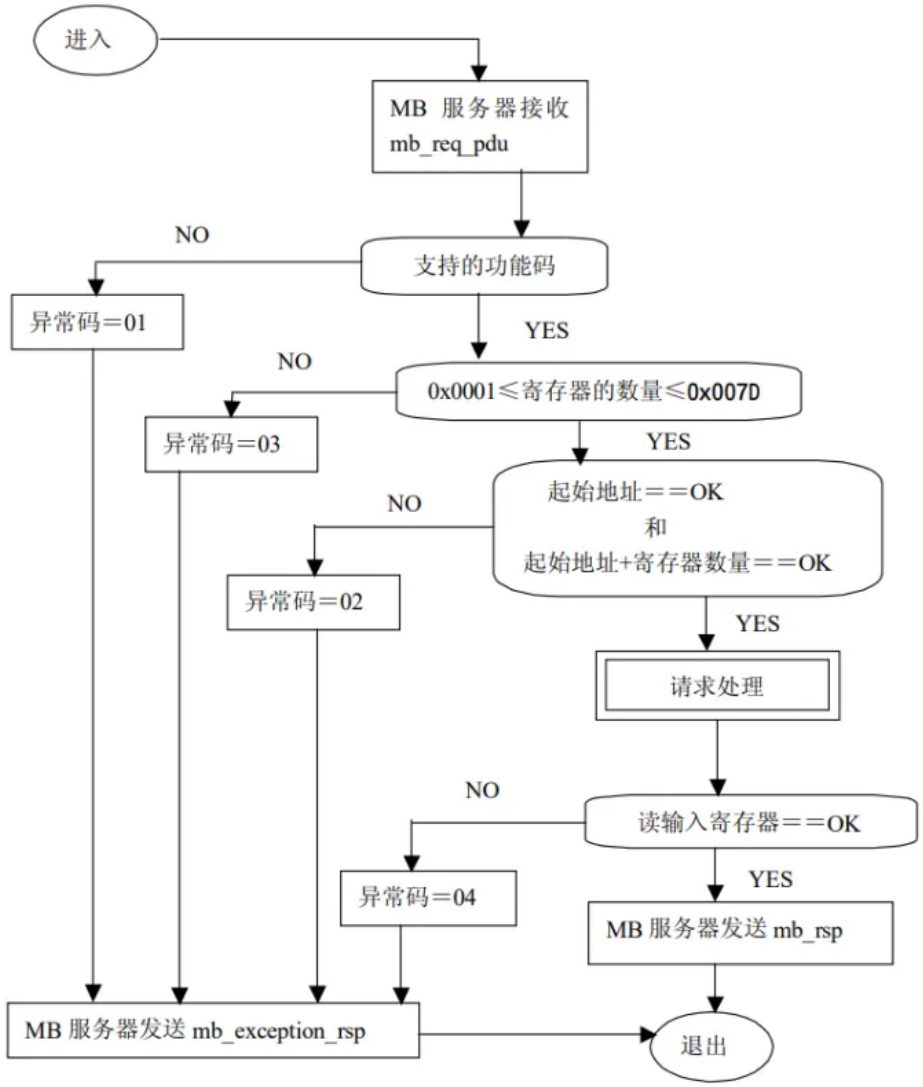

图 13：读输入寄存器的状态图

6.5 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 PDU

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 PDU

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	05	功能	05
输出地址 Hi	00	输出地址 Hi	00
输出地址 Lo	AC	输出地址 Lo	AC
输出值 Hi	FF	输出值 Hi	FF
输出值 Lo	00	输出值 Lo	00

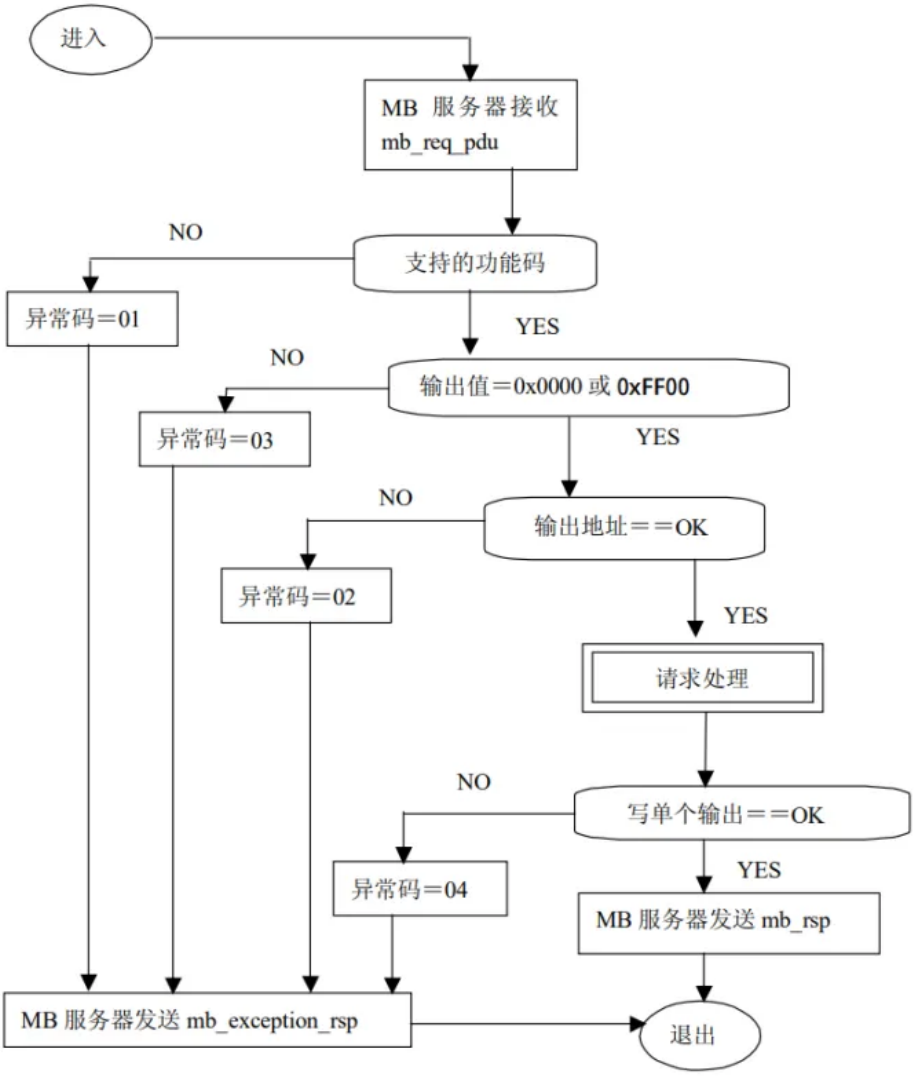

图 14：写单个输出状态图

6.6 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 PDU

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 PDU

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	06	功能	06
输出地址 Hi	00	输出地址 Hi	00
输出地址 Lo	01	输出地址 Lo	01
输出值 Hi	00	输出值 Hi	00
输出值 Lo	03	输出值 Lo	03

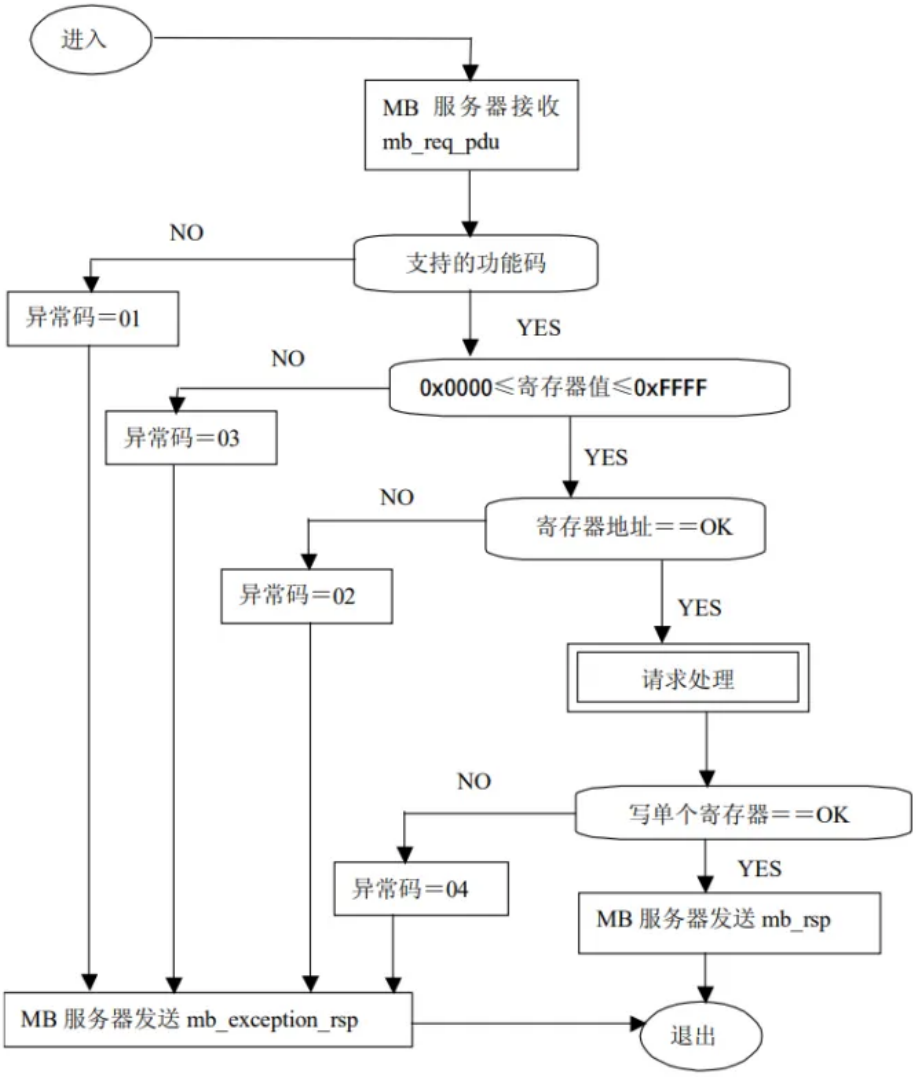

图 15：写单个寄存器状态图

6.7 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	0F	功能	0F
起始地址 Hi	00	起始地址 Hi	00
起始地址 Lo	13	起始地址 Lo	13
输出数量 Hi	00	输出数量 Hi	00
输出数量 Lo	0A	输出数量 Lo	0A
字节数	02		
输出值 Hi	CD		
输出值 Lo	01		

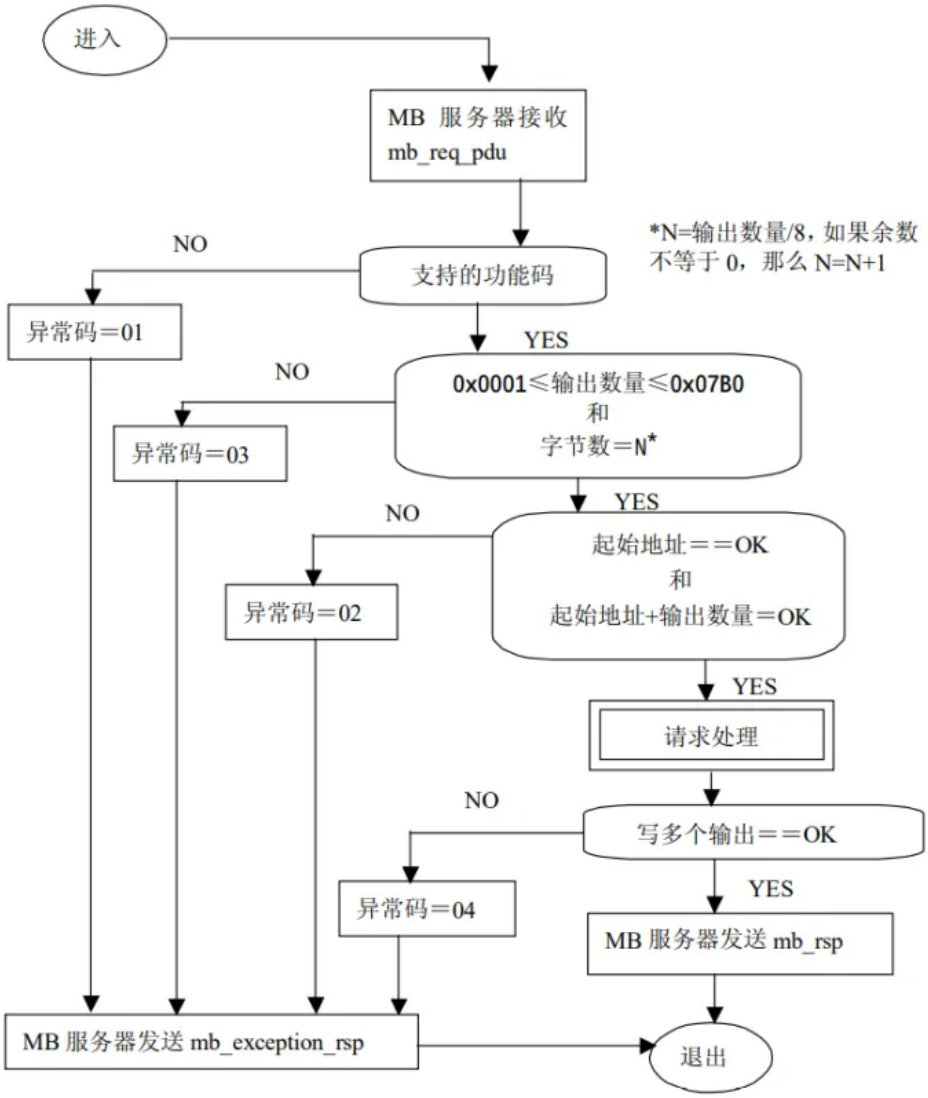

图 16：写多个输出的状态图

6.8 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	**N*** ×2
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	10	功能	10
起始地址 Hi	00	起始地址 Hi	00
起始地址 Lo	01	起始地址 Lo	01
寄存器数量 Hi	00	寄存器数量 Hi	00
寄存器数量 Lo	02	寄存器数量 Lo	02
字节数	04		
寄存器值 Hi	00		
寄存器值 Lo	0A		
寄存器值 Hi	01		
寄存器值 Lo	02		

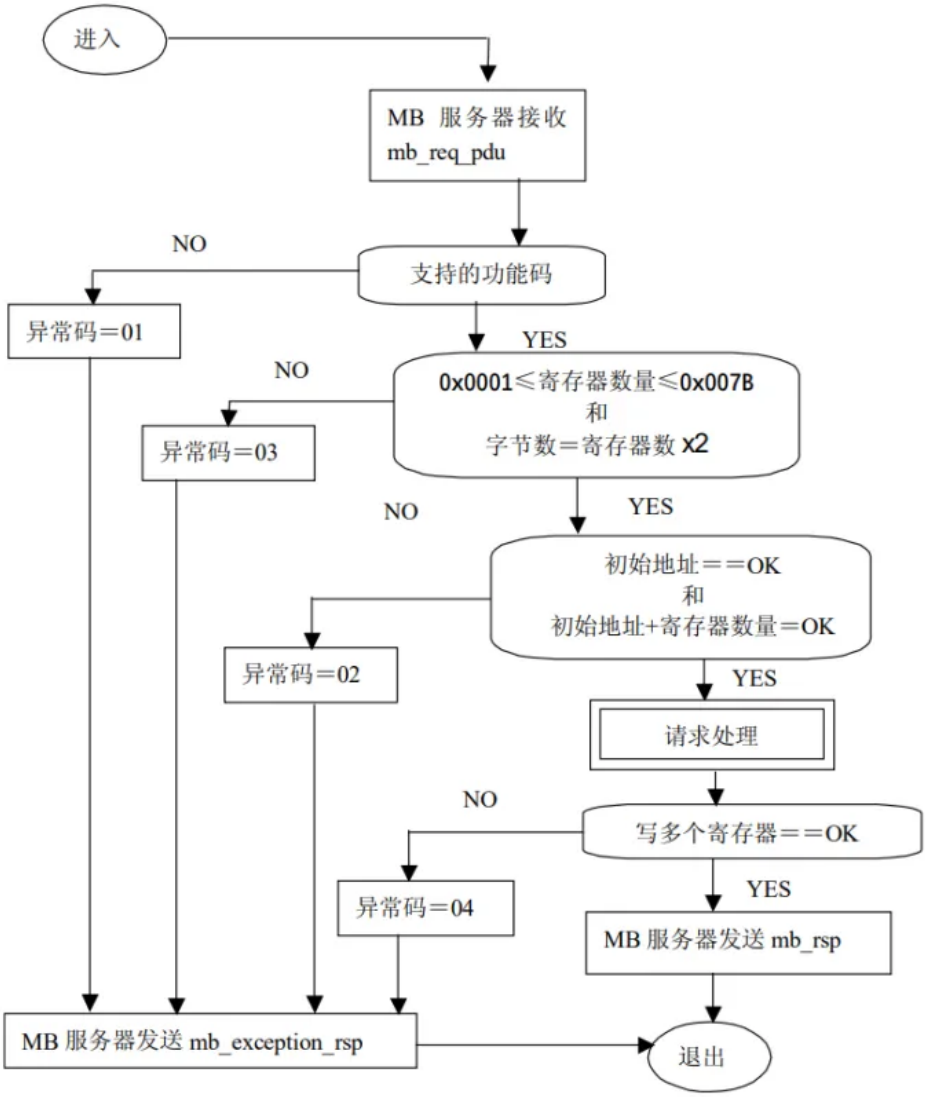

图 17：写多个寄存器状态图

6.9.1 20 (0x14) 读文件记录

使用该功能码进行文件记录读取。根据字节数量提供所有请求数据长度，并且根据寄存器提供所有记录长度。

文件是记录的结构。每个文件包括 10000 个记录，寻址这些记录为十进制 0000 至 9999 或十六进制 0X0000 至 0X270F，例如寻址记录 12 为 12。

该功能可以读取多个参考组。这些组可以是分散的(不连续的)，但每组中的参考必须是连续的。

用含有 7 个字节的独立“子请求”域定义每个组：

参考类型：1 个字节(必须规定为 6)

文件号：2 个字节

文件中的起始记录号：2 个字节

被读出的记录长度：2 个字节

被读取的寄存器数量不能超过 MODBUS 报文允许的长度：256 个字节，这个寄存器数量与预期响应中的所有其它域组合。

正常响应是一系列“子响应”，与“子请求”——对应。字节数域是所有“子响应”中的全部组合字节数。另外，每个“子响应”都包括一个表示自身字节数的域。

请求 PDU

功能码	1 个字节	**0x14**
字节数	1 个字节	0x07 至 0xF5
子请求 x，参考类型	1 个字节	0x06
子请求 x，文件号	2 个字节	0x0000 至 0xFFFF
子请求 x，记录号	2 个字节	0x0000 至 0x270F
子请求 x，记录长度	2 个字节	**N**
子请求 x+1，…		

响应 PDU

功能码	1 个字节	**0x14**
响应数据长度	1 个字节	0x07 至 0xF5
子请求 x，文件响应长度	1 个字节	0x07 至 0xF5
子请求 x，参考类型	1 个字节	06
子请求 x，记录数据	N×2 个字节	
子请求 x+1，…		

错误

差错码	1 个字节	0x94
异常码	1 个字节	01 或 02 或 03 或 04

这是一个请求从远程设备读取两个参考组的实例:

组1 包括文件 4 中的 2 个寄存器，以寄存器 1 开始（地址 0001）。

组2 包括文件 3 中的 2 个寄存器，以寄存器 9 开始（地址 0009）。

请求		响应	
域名	（十六进制）	域名	（十六进制）
功能	14	功能	14
字节数	0C	响应数据长度	0E
子请求 1，参考类型	06	子请求 1，文件响应长度	05
子请求 1，文件号 Hi	00	子请求 1，参考类型	06
子请求 1，文件号 Lo	04	子请求 1，记录数据 Hi	0D
子请求 1，记录号 Hi	00	子请求 1，记录数据 Lo	FE
子请求 1，记录号 Lo	01	子请求 1，记录数据 Hi	00
子请求 1，记录长度 Hi	00	子请求 1，记录数据 Lo	20
子请求 1，记录长度 Lo	02	子请求 2，文件响应长度	05
子请求 2，参考类型	06	子请求 2，参考类型	06
子请求 2，文件号 Hi	00	子请求 2，记录数据 Hi	33
子请求 2，文件号 Lo	03	子请求 2，记录数据 Lo	CD
子请求 2，记录号 Hi	00	子请求 2，记录数据 Hi	00
子请求 2，记录号 Lo	09	子请求 2，记录数据 Lo	40
子请求 2，记录长度 Hi	00		
子请求 2，记录长度 Lo	02		

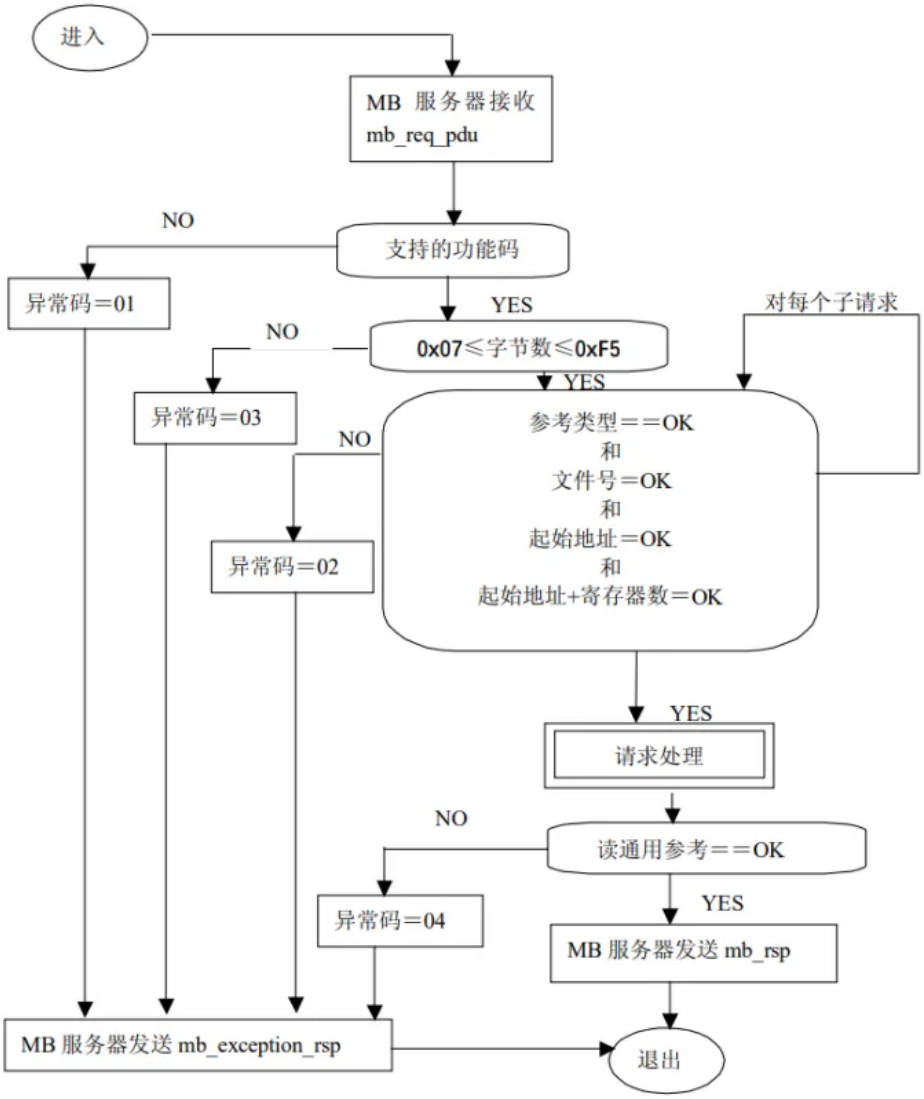

图 18：读文件记录状态图

6.9.2 21 (0x15) 写文件记录

使用该功能码进行文件记录写入。根据字节数量提供所有请求数据长度，并且根据 16 比特字的数量提供所有记录长度。

文件是记录的结构。每个文件包括 10000 个记录，寻址这些记录为十进制 0000 至 9999 或十六进制 0X0000 至 0X270F，例如寻址记录 12 为 12。

该功能可以写多个参考组。这些组可以是分散的，即不连续的，但每组内的参考必须是连续的。

用含有 7 个字节和数据的独立“子请求”域定义每个组：

参考类型：1 个字节(必须规定为 6)

文件号：2 个字节

文件中的起始记录号：2 个字节

被写入的记录长度：2 个字节

被写入的数据：每个寄存器为 2 字节。

被写入的寄存器数量不能超过 MODBUS 报文允许的长度：256 个字节，这个寄存器数量与询问中的所有其它域组合。

正常响应是请求的应答。

请求 PDU

功能码	1 个字节	**0x15**
请求数据长度	1 个字节	0x07 至 0xF5
子请求 x，参考类型	1 个字节	0x06
子请求 x，文件号	2 个字节	0x0000 至 0xFFFF
子请求 x，记录号	2 个字节	0x0000 至 0x270F
子请求 x，记录长度	2 个字节	**N**
子请求 x，记录数据	N×2 个字节	
子请求 x+1，…		

响应 PDU

功能码	1 个字节	**0x15**
响应数据长度	1 个字节	
子请求 x，参考类型	1 个字节	06
子请求 x，文件号	2 个字节	0x0000 至 0xFFFF
子请求 x，记录号	2 个字节	0x0000 至 0xFFFF
子请求 x，记录长度	2 个字节	**N**
子请求 x，记录数据	N×2 个字节	
子请求 x+1，…		

错误

差错码	1 个字节	0x95
异常码	1 个字节	01 或 02 或 03 或 04

这是一个请求将一个参考组写入远程设备的实例：

组包括文件4 中的3 个寄存器，以寄存器7 开始（地址0007）。

请求		响应	
域名	（十六进制）	域名	（十六进制）
功能	15	功能	15
请求数据长度	0D	请求数据长度	0D
子请求 1，参考类型	06	子请求 1，参考类型	06
子请求 1，文件号 Hi	00	子请求 1，文件号 Hi	00
子请求 1，文件号 Lo	04	子请求 1，文件号 Lo	04
子请求 1，记录号 Hi	00	子请求 1，记录号 Hi	00
子请求 1，记录号 Lo	07	子请求 1，记录号 Lo	07
子请求 1，记录长度 Hi	00	子请求 1，记录长度 Hi	00
子请求 1，记录长度 Lo	03	子请求 1，记录长度 Lo	03
子请求 1，记录数据 Hi	06	子请求 1，记录数据 Hi	06
子请求 1，记录数据 Lo	AF	子请求 1，记录数据 Lo	AF
子请求 1，记录数据 Hi	04	子请求 1，记录数据 Hi	04
子请求 1，记录数据 Lo	BE	子请求 1，记录数据 Lo	BE
子请求 1，记录数据 Hi	10	子请求 1，记录数据 Hi	10
子请求 1，寄存器数据 Lo	0D	子请求 1，寄存器数据 Lo	0D

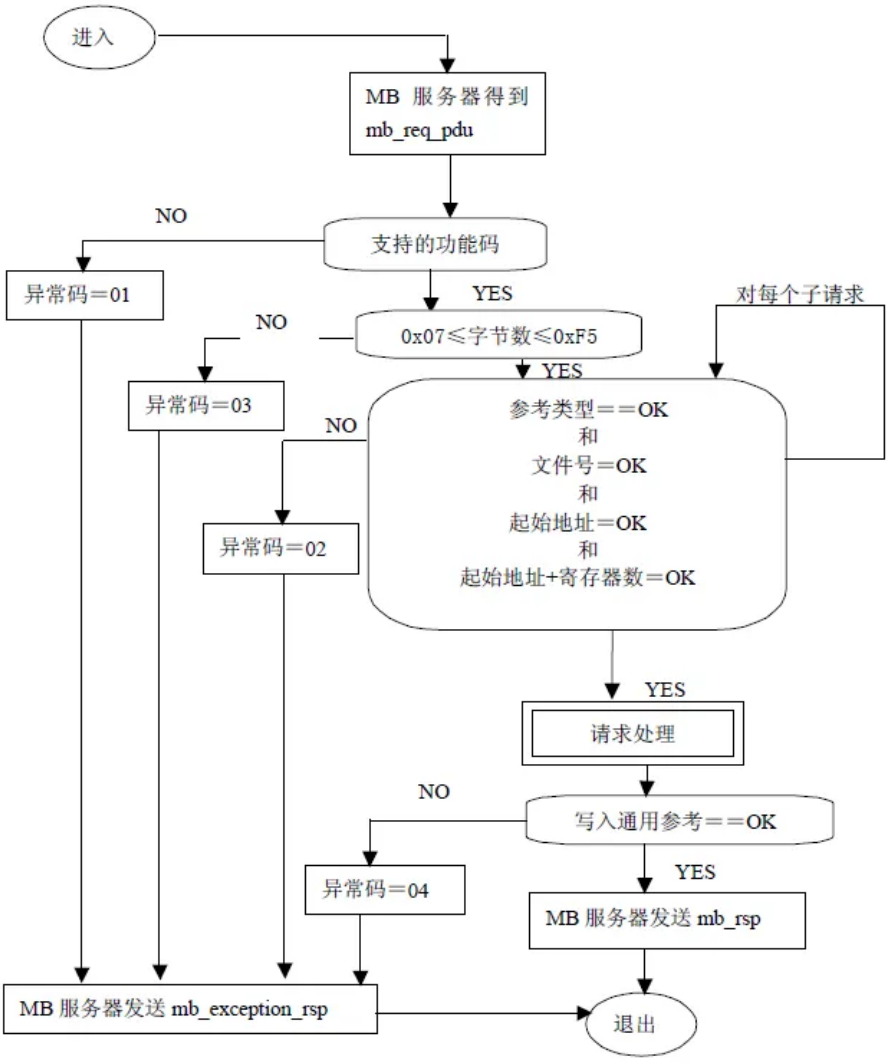

图 19：写文件记录状态图

6.10 22 (0x16) 屏蔽写寄存器

该功能码用于通过利用 AND 屏蔽、OR 屏蔽以及寄存器内容的组合来修改特定保持寄存器的内容。使用这个功能设置或清除寄存器中的单个比特。

请求说明了被写入的保持寄存器、AND屏蔽使用的数据以及 OR 屏蔽使用的数据。

从0 开始寻址寄存器。因此，寻址寄存器 1-16为0-15。功能的算法为：

结果= (当前内容 AND And_Mask) OR (Or_Mask AND And_Mask)

例如：

	十六进制	二进制
当前内容 =	12	0001 0010
And_Mask =	F2	1111 0010
Or_Mask =	25	0010 0101
And_Mask =	0D	0000 1101
结果=	17	0001 0111

注：如果Or_Mask 值为零，那么结果是当前内容和 And_Mask的简单逻辑 AND（与）。如果 And_Mask 值为零，结果等于 Or_Mask值。可以使用读保持寄存器功能（功能码03）读出寄存器

的内容。于是，当控制器扫描它的用户逻辑程序时，随后可以改变寄存器的内容。

正常的响应是请求的应答。在已经写入寄存器之后，返回响应。

请求 PDU

功能码	1 个字节	**0x16**
参考地址	2 个字节	0x0000 至 0xFFFF
And_Mask	2 个字节	0x0000 至 0xFFFF
Or_Mask	2 个字节	0x0000 至 0xFFFF

响应 PDU

功能码	1 个字节	**0x16**
参考地址	2 个字节	0x0000 至 0xFFFF
And_Mask	2 个字节	0x0000 至 0xFFFF
Or_Mask	2 个字节	0x0000 至 0xFFFF

错误

差错码	1 个字节	0x96
异常码	1 个字节	01 或 02 或 03 或 04

这是一个利用上述屏蔽值在远程设备中对寄存器 5 的屏蔽写入实例

请求		响应	
域名	（十六进制）	域名	（十六进制）
功能	16	功能	16
参考地址 Hi	00	参考地址 Hi	00
参考地址 Lo	04	参考地址 Lo	04
And_Mask Hi	00	And_Mask Hi	00
And_Mask Lo	F2	And_Mask Lo	F2
Or_Mask Hi	00	Or_Mask Hi	00
Or_Mask Lo	25	Or_Mask Lo	25

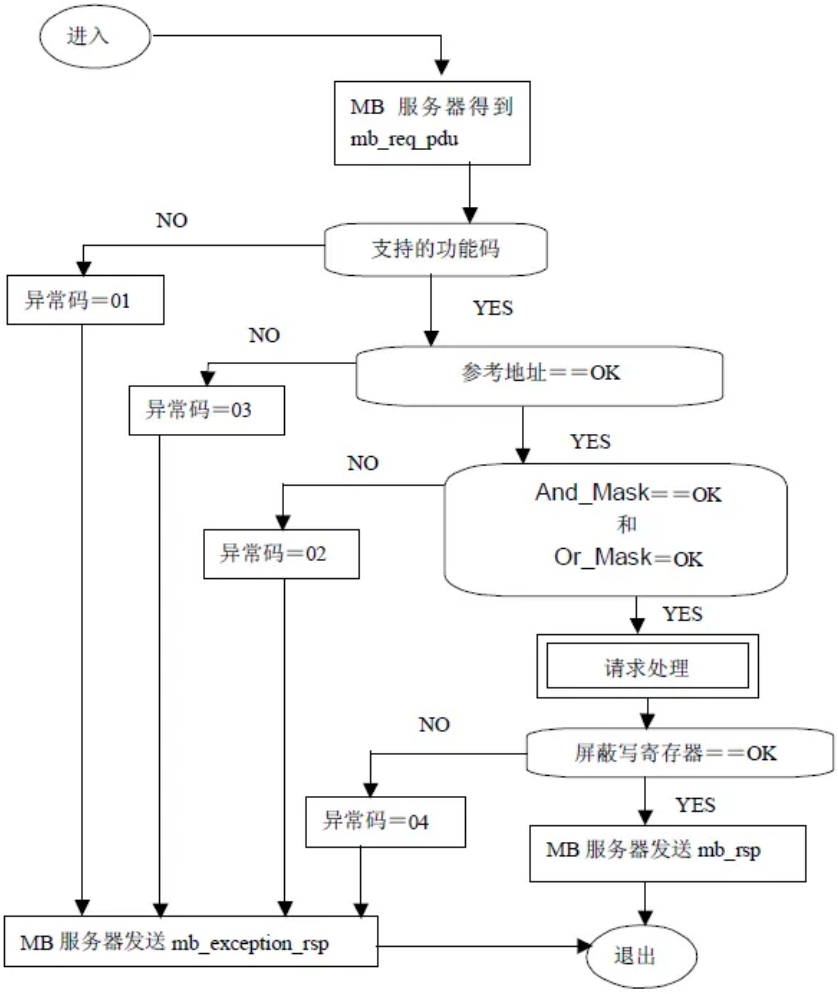

图 20：屏蔽写保持寄存器状态图

6.11 23 (0x17) 读/写多个寄存器

在一个单独 MODBUS 事务中，这个功能码实现了一个读操作和一个写操作的组合。从零开始寻址保持寄存器。因此，寻址保持寄存器 1-16为0-15。

请求说明了起始地址、被读取的保持寄存器号和起始地址、保持寄存器号以及被写入的数据。在写数据域中，字节数说明随后的字节号。

正常响应包括被读出的寄存器组的数据。在读数据域中，字节数域说明随后的字节数量。

请求 PDU

功能码	1 个字节	**0x17**
读起始地址	2 个字节	0x0000 至 0xFFFF
读的数量	2 个字节	0x0001 至近似 0x0076
写的起始地址	2 个字节	0x0000 至 0xFFFF
写的数量	2 个字节	0x0001 至近似 0x0076
写的字节	1 个字节	**N*** ×2
写寄存器值	**N*** ×2 个字节	

N*=写的数量

响应 PDU

功能码	1 个字节	**0x17**
字节数	2 个字节	**N*** ×2
读寄存器值	**N*** ×2 个字节	

N*=读的数量

错误

差错码	1 个字节	0x97
异常码	1 个字节	01 或 02 或 03 或 04

这是一个请求从寄存器4开始读六个寄存器并且从寄存器15开始读三个寄存器的实例：

请求		响应	
域名	（十六进制）	域名	（十六进制）
功能	17	功能	17
读起始地址 Hi	00	字节数	0C
读起始地址 Lo	03	读寄存器值 Hi	00
读的数量 Hi	00	读寄存器值 Lo	FE
读的数量 Lo	06	读寄存器值 Hi	0A
写起始地址 Hi	00	读寄存器值 Lo	CD
写起始地址 Lo	0E	读寄存器值 Hi	00
写的数量 Hi	00	读寄存器值 Lo	01
写的数量 Lo	03	读寄存器值 Hi	00
写字节数	06	读寄存器值 Lo	03
写寄存器值 Hi	00	读寄存器值 Hi	00
写寄存器值 Lo	FF	读寄存器值 Lo	0D
写寄存器值 Hi	00	读寄存器值 Hi	00
写寄存器值 Lo	FF	读寄存器值 Lo	FF
写寄存器值 Hi	00		
写寄存器值 Lo	FF		

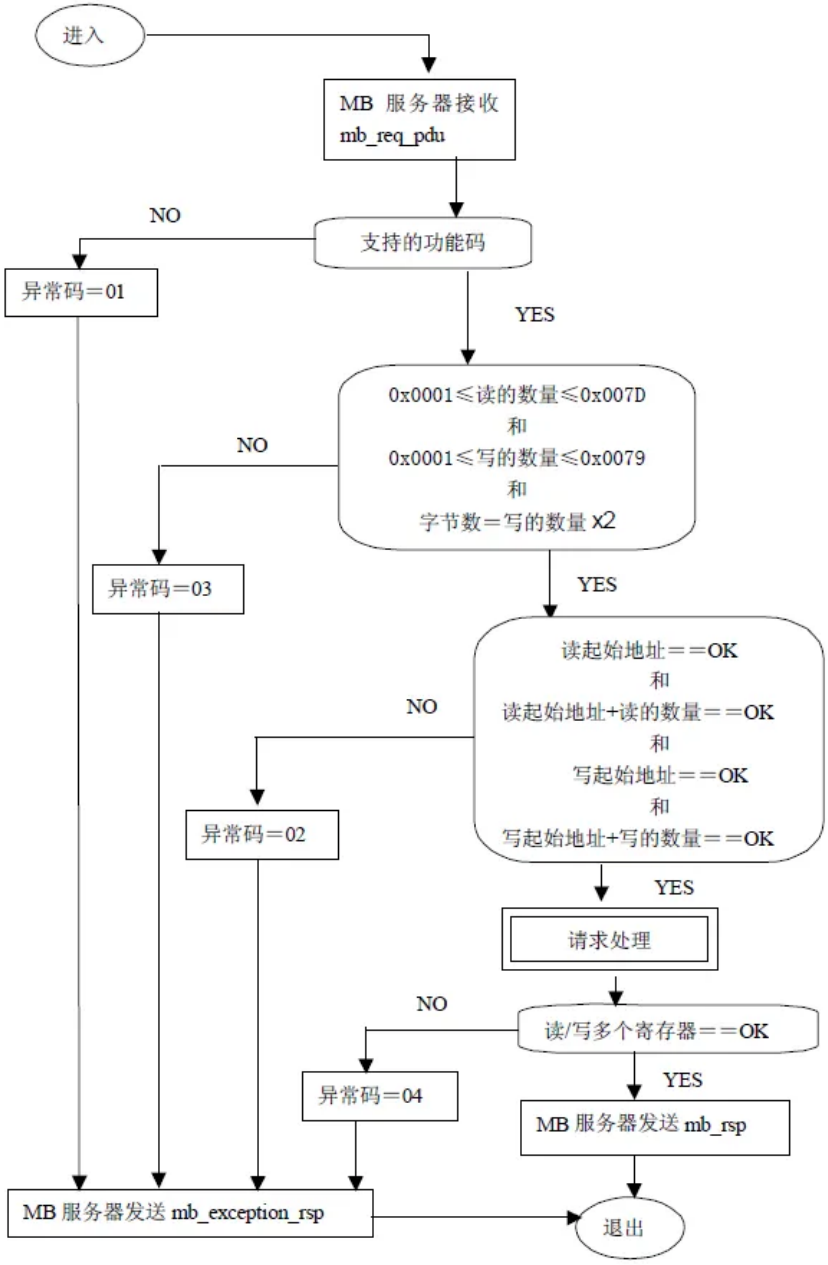

图 21：读/写多个寄存器状态图

6.12 43 (0x2B)读设备识别码

这个功能码允许读取与远程设备的物理描述和功能描述相关的识别码和附加报文。

将读设备识别码接口模拟为一个地址空间，这个地址空间由一组可寻址数据元素组成。数据元素是被叫对象，并且对象Id确定这个数据元素。

接口由 3 种对象组成：

- 基本设备识别码。所有此种对象都是必备的：厂商名称、产品代码和修订本号。
- 正常设备识别码。除基本数据对象以外，设备提供了附加的和可选择的识别码以及数据对象描述。按标准定义所有种类的对象，但是这种对象的执行是可选的。
- 扩展设备识别码。除正常数据对象以外，设备提供了附加的和可选的识别码以及专用数据描述。所有这些数据都是与设备有关的。

对象 Id	对象名称/描述	类型	M/O	种类
0x00	厂商名称	ASCII 字符串	强制的	基本
0x01	产品代码	ASCII 字符串	强制的	
0x02	主要修订本	ASCII 字符串	强制的	
0x03	VendorUrl	ASCII 字符串	可选的	规则
0x04	产品名称	ASCII 字符串	可选的	
0x05	模式名称	ASCII 字符串	可选的	
0x06	用户应用名称	ASCII 字符串	可选的	
0x07 … 0x7F	保留		可选的	
0x80 … 0xFF	可选择地定义专用对象 范围[0x80—0xFF]与产品有关	相关设备	可选的	扩展

请求 PDU

功能码	1 个字节	**0x2B**
MEI 类型	1 个字节	0x0E
ReadDevId 码	1 个字节	01/02/03/04
对象 id	1 个字节	0x00 至 0xFF

响应 PDU

功能码	1 个字节	**0x2B**
MEI 类型	1 个字节	0x0E
ReadDevId 码	1 个字节	01/02/03/04
一致性等级	1 个字节	
随后更多	1 个字节	00/FF
下一个对象 Id	1 个字节	对象 ID 号
对象号	1 个字节	
对象 ID 的列表	1 个字节	
对象长度	1 个字节	
对象值	1 个字节	

错误

差错码	1 个字节	0xAB（Fc 0x2B+0x80）
MEI 类型	1 个字节	14
异常码	1 个字节	01、02、03、04

请求参数描述：

指配号为14的MODBUS封装接口识别读识别码请求。定义四种访问类型：

01：请求获得基本设备识别码（流访问）

02：请求获得正常设备识别码（流访问）

03：请求获得扩展设备识别码（流访问）

04：请求获得特定识别码对象（专用访问）

在识别码数据不适合单独响应的情况下，可以需要几个请求/响应事务处理。对象id字节给出了获得的第一个对象识别码。对于第一个事物处理来说，客户机必须设置对象id为0，以便获得设备识别码数据的开始。对于下列事务来说，客户机必须设置对象id为前面响应中服务器的返回值。

如果对象id不符合任何已知对象，那么服务器象指向对象0那样响应（从头开始）。

在单个访问的情况下：ReadDevId代码04，请求中的对象id给出了获得的对象识别码。

如果对象id不符合任何已知对象，那么服务器返回一个异常码 = 02（非法数据地址）的异常响应。

响应参数描述：

功能码： 功能码 43（十进制）0x2B (十六进制)

MEI 类型： 为设备识别码接口指配号的 14 (0x0E) MEI 类型

ReadDevId 码： 与请求 ReadDevId 码相同：01、02、03 或 04

一致性等级： 设备的识别码一致性等级和支持访问的类型

01：基本识别码(仅流访问)

02：正常识别码(仅流访问)

03：扩展识别码(仅流访问)

81：基本识别码(流访问和单个访问)

82：正常识别码(流访问和单个访问)

83：扩展识别码(流访问和单个访问)

随后更多： **在 ReadDevId 码 01、02或03(流访问)的情况下，**

如果识别码数据不符合单个响应，那么需要几个请求/响应事务处理。

00：对象不再是可利用的

FF：其它识别码对象是可利用的，并且需要更多 MODBUS 事务处理

在 ReadDevId码04(单个访问)的情况下，

必须设置这个域为00。

下一个对象 Id： 如果“随后更多=FF”，那么请求下一个对象的识别码

如果“随后更多=00”，那么必须设置为00(无用的)对象号

在响应中返回的对象识别码号

(对于单个访问，对象号码= 1)

对象 0.id： PDU 中返回的第一个对象识别码(流访问)或请求对象的识别码（单个访问）

Object0.长度： 第一个对象的字节长度

Object0.值： 第一个对象的值(对象0.长度字节)

…

ObjectN.id： 最后对象的识别码(在响应中)

ObjectN.长度： 最后对象的字节长度

ObjectN.值： 最后对象的值(对象N.长度字节)

“基本设备识别码”的读设备识别码请求的实例：在这个实例中，一个响应PDU中发送所有的报文。

请求		响应	
域名	值	域名	值
功能	2B	功能	2B
MEI 类型	0E	MEI 类型	0E
ReadDevId 码	01	ReadDevId 码	01
对象 id	00	一致性等级	01
		更多继续	00
		下一个对象 id	00
		对象号	03
		对象 id	00
		对象长度	16
		对象值	“公司识别码”
		对象 id	01
		对象长度	0A
		对象值	“产品代码”
		对象 id	02
		对象长度	05
		对象值	“V2.11”

如果一个设备需要几个事务处理发送响应，那么启动下列事务处理。

第一个事务处理：

请求		响应	
域名	值	域名	值
功能	2B	功能	2B
MEI 类型	0E	MEI 类型	0E
ReadDevId 码	01	ReadDevId 码	01
对象 id	00	一致性等级	01
		更多继续	FF
		下一个对象 id	02
		对象号	03
		对象 id	00
		对象长度	16
		对象值	“公司识别码”
		对象 id	01
		对象长度	1A
		对象值	“产品代码” XXXXXXXXXXX

第二个事务处理：

请求		响应	
域名	值	域名	值
功能	2B	功能	2B
MEI 类型	0E	MEI 类型	0E
ReadDevId 码	01	ReadDevId 码	01
对象 id	02	一致性等级	01
		更多继续	00
		下一个对象 id	00
		对象号	03
		对象 id	02
		对象长度	05
		对象值	V2.11

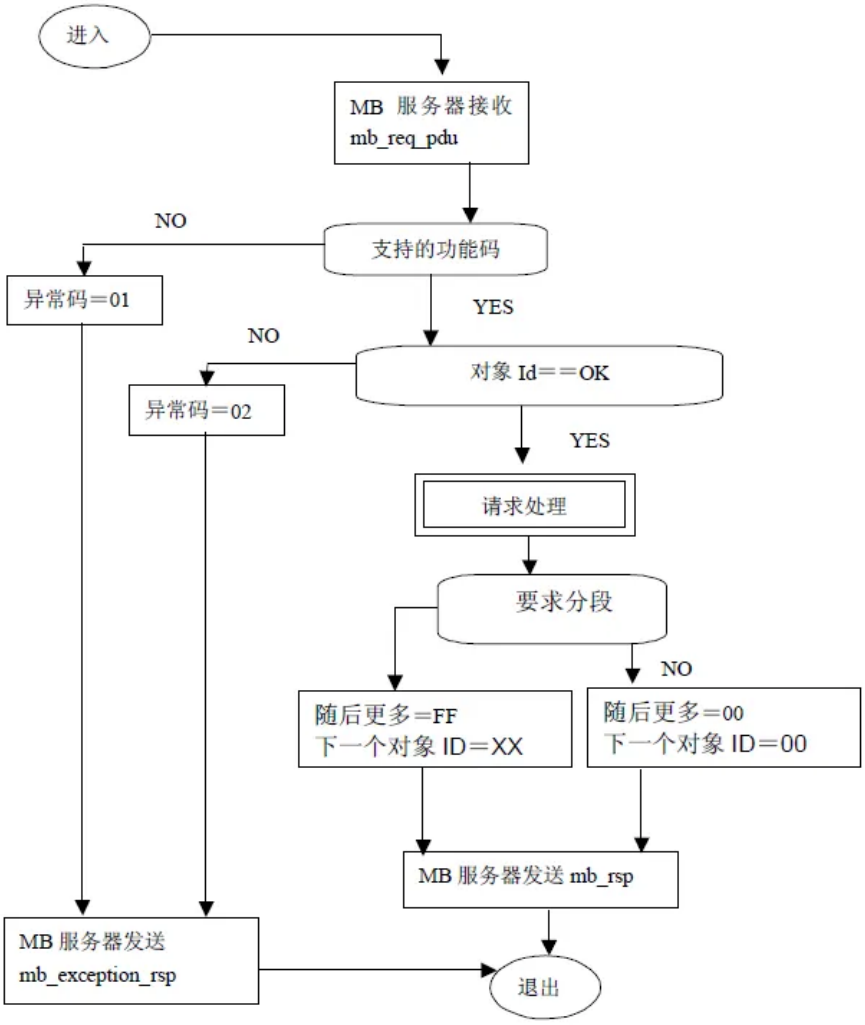

图 22：读设备识别码状态图

7、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	值	域名	值
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。

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。
05	确认	与编程命令一起使用。服务器(或从站)已经接受请求，并切正在处理这个请求，但是需要长的持续时间进行这些操作。返回这个响应防止在客户机(或主站)中发生超时错误。客户机(或主站)可以继续发送轮询程序完成报文来确定是否完成处理。
06	从属设备忙	与编程命令一起使用。服务器(或从站)正在处理长持续时间的程序命令。服务器(或从站)空闲时，用户(或主站)应该稍后重新传输报文。
08	存储奇偶性差错	与功能码20和21 以及参考类型6一起使用，指示扩展文件区不能通过一致性校验。 服务器(或从站)设法读取记录文件，但是在存储器中发现一个奇偶校验错误。客户机(或主方)可以重新发送请求，但可以在服务器(或从站)设备上要求服务。
0A	不可用网关路径	与网关一起使用，指示网关不能为处理请求分配输入端口至输出端口的内部通信路径。通常意味着网关是错误配置的或过载的。
0B	网关目标设备响应失败	与网关一起使用，指示没有从目标设备中获得响应。通常意味着设备未在网络中。

编辑于 2021-03-19 10:10

MODBUS协议　通讯协议

写下你的评论...

8 条评论　默认　最新

孙建希
这是机翻吗?
2020-10-26 4

方言
主要是里面专有名词，Discrete Output Coils, Discrete Input Contacts。文中翻译成“线圈”和“输入离散量”。我觉得Discrete翻译成“开关”好点，因为一般指的就是开关0or1这种离散值。所以上边两个翻译成
Discrete Output Coils：开关输出线圈
Discrete Input Contacts：开关输入触点
这样子，可能会好一些。
这些名词可能因为该协议是施耐德早期设计用在自己产品上的，所以在协议里带了明显的指向性。做为一个通用协议，私以为起名字时还是应该更抽象一些得好。
2022-04-29 3

Strong Tu
我觉得是。。
2021-02-27 2

展开其他 1 条回复 >

Bruce
推荐使用MThings工具，辅助Modbus调试调测
2020-07-30 1

格物
CRC校验没有谈啊
2022-11-11 赞

飞云客青松
MBAP MODBUS 协议，这个搞错了应该?
2022-10-20 赞

DF express
好多字😓
2021-06-03 赞

文章被以下专栏收录

串口传输，通信协议
串口数据交互

串口通信协议

协议类技术知识
转载，硬核知识分享。