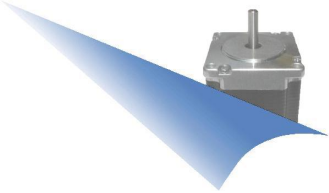
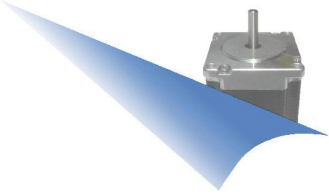


iD254-MDFA 闭环步进电机驱动器

使用说明书

版本号：1.1

目　录



1 【主要规格】

项 目	内 容	备注
型 号	iD254-MDFA	
输入电源电压	DC 24V～48V	
额定输出电流	iD254-MDFA 4.5A (0 -peak)	连续电流
最大输出电流	iD254-MDFA 4.5A (0 -peak)	瞬间电流
控制对象电机	附编码器 2 相双极性步进电机	
驱动方式	PWM 恒流驱动	
通讯界面	**输入** · 脉冲、方向输入（可配置为数字输入） · 数字输入 5 个 · 编码器输入(A，B，Z) **输出** · 数字输出 4 个 · 编码信号输出(差动 A，B，Z)	除了编码器输出为固定，其余的输入/输出均可通过通讯自由配置
数字输入详细内容	/SV ON (Servo On) /RESET (报警复位) /START (电机启动/停止) /JOG (电机点动) /HOME (回零点)	
数字输出详细内容	/IN POTISION /ALARM	
LED 指示	状态、故障	2 个指示灯
通讯 I/F	RS485，最多 32 节点	MODBUS RTU 协议，波特率：19200bps(预设) **或 根据约定**
控制方法	位置控制模式	根据脉冲定位、根据 RS485 通讯定位
外形尺寸（mm）	W77 ×D134 ×H34	不含接线端子
重量	约 350g	不含接线端子
动作温度/湿度	0～45℃，85%RH 以下	防止冷凝
保存温度	0～85℃，85%以下	防止冷凝
环境气体	防止腐蚀性气体	

2 【准备】

接通电源前请务必进行以下工作。

2.1 配线

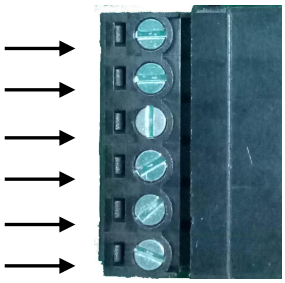
请务必确认参照叙述接头指定表进行配线。

1. CN1 ： 电源与电机的配线
   请正确连接电源与电机。若将电机输出端子连接至电源可能会导致驱动器破损。
   请注意：请使用 AWG#20 以上线材。
2. CN2 ： 编码器的配线
3. CN3 ： 接口信号的配线
   请配置必要的数字输入和数字输出信号。通用输入/输出皆以光耦合器隔离。接口用的电源(+24V)请另行准备。 （备注：编码器信号为差分输出，未用光耦合器隔离）
4. CN5 ： RS485 通讯的配线 请使用 RJ45 接头。

3 【接插件指定表】

3.1 CN1 (Power & Motor)

Pin.	信号名称
6	电机 A+
5	电机 A-
4	电机 B+
3	电机 B-
2	电源 GND
1	电源 V+ (DC24V or 48V)



接线时注意电源极性

使用电线规格：AWG20～AWG16(多股线)

使用专用工具紧固接线端子

紧固端子时，请使用刃宽 0.4×2.5 的螺丝刀。

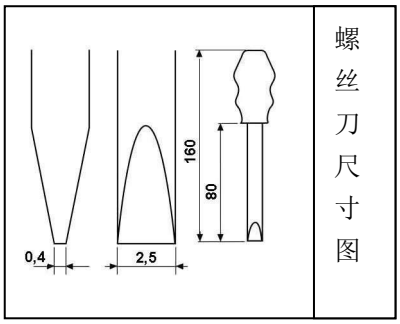
例如：Phoenix Contact 的螺丝刀

（产品编号：1205037，型号 SZS 0. 4×2. 5）

。

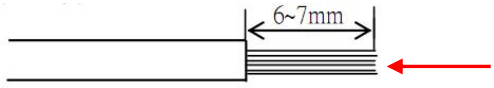
紧固力矩请选用 0.22～0.25N·m

（2.3kgf·cm～2.5kgf·cm）。



接线方法：

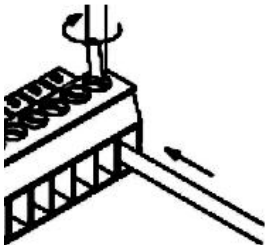
① **剥线长度：6～7mm**



请勿在线头上先上一层焊锡。
(可能会导致无法正常接线)

② **插入电线，直到碰到端子台，顺时针方向拧紧螺丝，固定电线。**

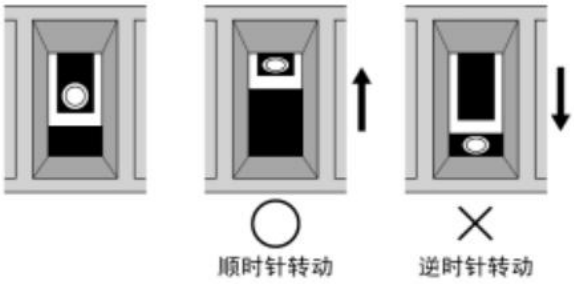
（紧固力矩选用 0.22～0.25N·m（2.3kgf·cm～2.5kgf·cm））



接线时的注意事项

遵守以下各项，注意不要断线。

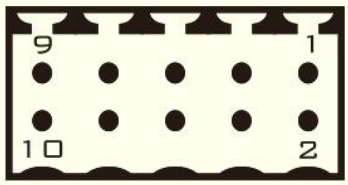
- 剥去包覆层时，不要损伤芯线。
- 接线时，注意不要使芯线扭结同时芯线不可外漏从而避免引起导线短路。
- 芯线请直接连接，不要焊接。否则有时会因振动而断线。
- 接线后，电线上不可施加压力。
- 由于端子的构造，若逆时针转动而固定电线时，会造成接触不良。请拔出电线，确认端子孔后重新接线。

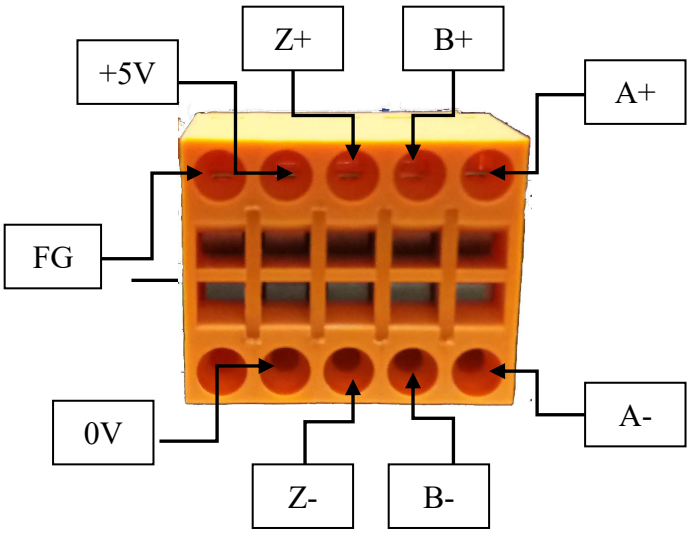


3.2 CN2 (Encoder IN)

Pin.	信号名称	Pin.	信号名称
1	A+	2	A-
3	B+	4	B-
5	Z+	6	Z-
7	+5V	8	0V
9	FG	10	NC

示意图



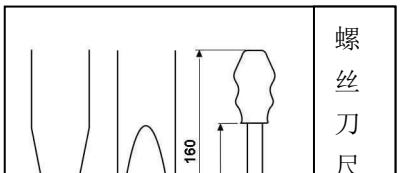


接线时注意编码器电源极性

使用电线规格：AWG28～AWG18(多股线)

端子为回拉式弹簧连接，采用正面接线方式，在使用专用螺丝刀情况下操作非常简便。

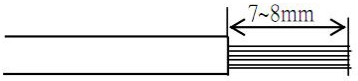
使用专用工具紧固接线端子



紧固端子时，请使用刃宽 0.4×2.5 的螺丝刀。
例如：Phoenix Contact 的螺丝刀
（产品编号：1205037，型号 SZS 0. 4×2. 5）。

接线方法：

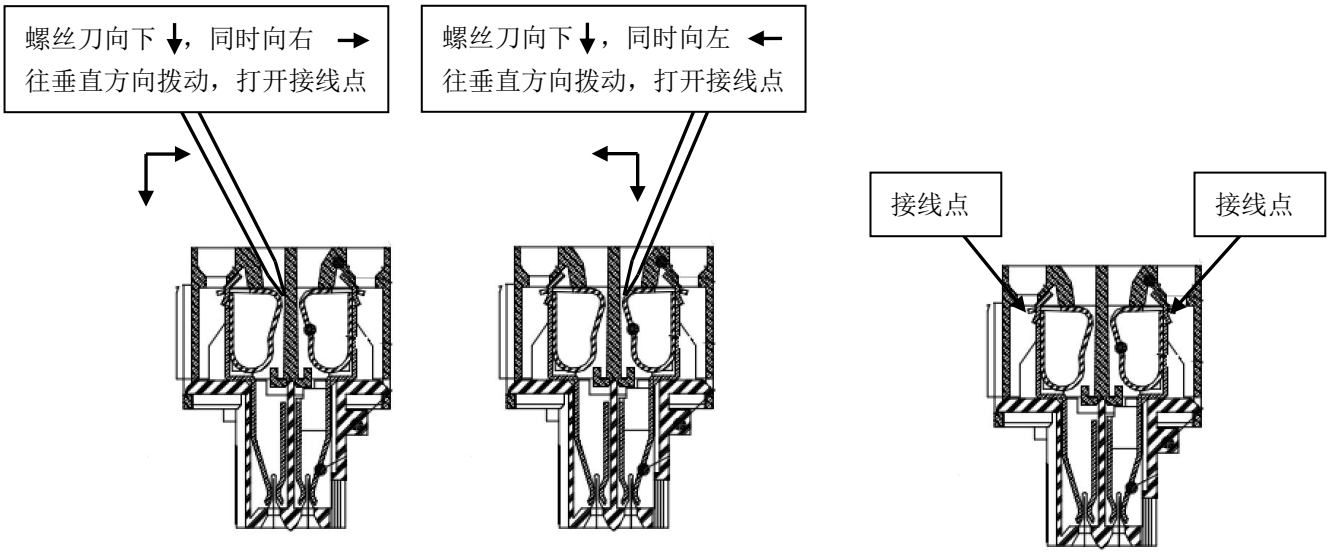
①剥线长度：7～8mm



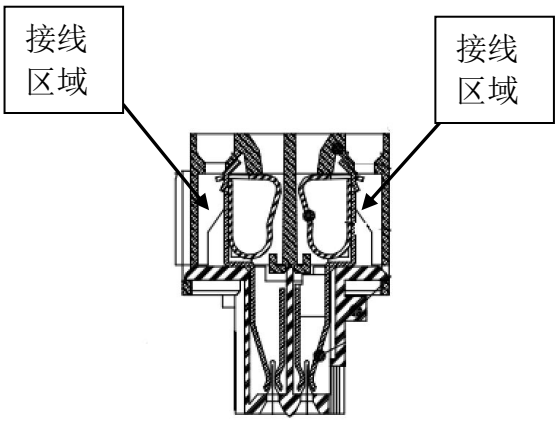
请勿在线头上先上一层焊锡。
(可能会导致无法正常接线)

端子为回拉式弹簧连接，采用正面接线方式，操作非常简便：

②您可以用标准螺丝刀打开接线点。



③将导线插入**接线区域**，然后**移开**螺丝刀。导线实现自动连接。



接线时的注意事项

遵守以下各项，注意不要断线。

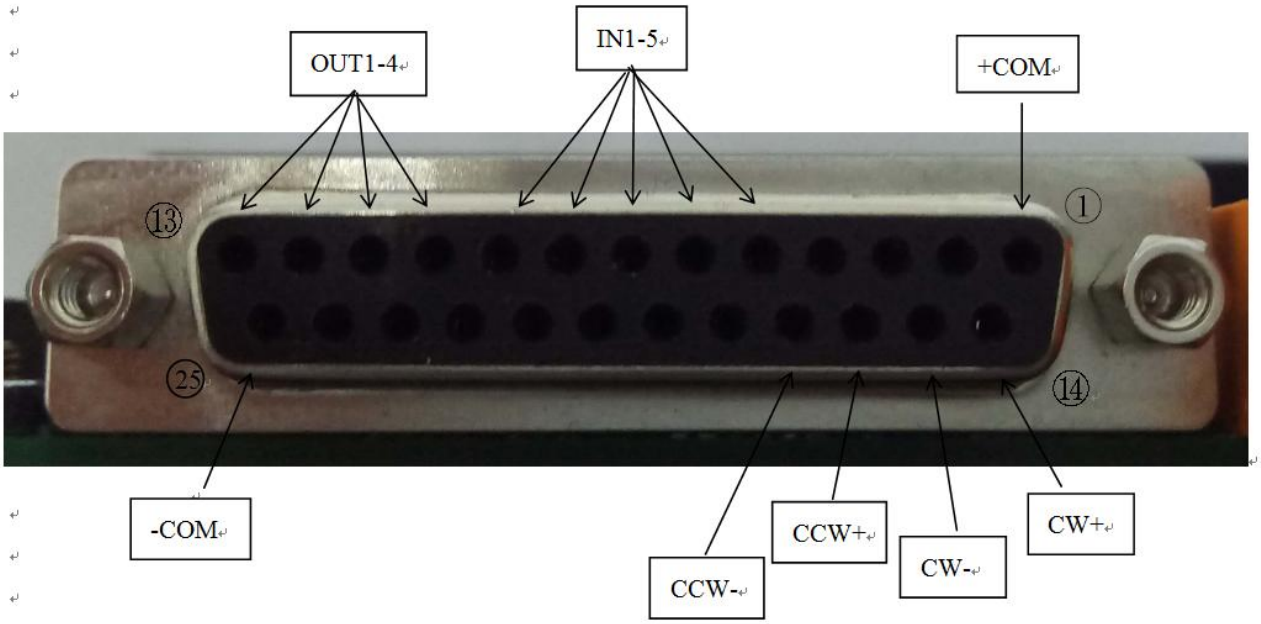
- **剥去包覆层时，不要损伤芯线。**
- **接线时，注意不要使芯线扭结，同时芯线不可外漏，避免引起导线短路。**
- **芯线请直接连接，不要焊接。否则有时会因振动而断线。**
- **接线后，电线上不可施加压力。**
- **必须使用规定尺寸、同等类型的螺丝刀，否则将存在损坏接线端子弹片的风险。**

1.1 CN3 （I/O）

（备注：18-23 脚为编码器输出{差分输出}，为选配，需订货时注明）

Pin.	信号名称	Pin.	信号名称	Pin.	信号名称	Pin.	信号名称
1	+COM(24V)	7	IN3	14	CW+	20	Encoder B+
2	NC	8	IN4	15	CW-	21	Encoder B-
3	NC	9	IN5	16	CCW+	22	Encoder Z+
4	NC	10	OUT1	17	CCW-	23	Encoder Z-
5	IN1	11	OUT2	18	Encoder A+	24	FG
6	IN2	12	OUT3	19	Encoder A-	25	-COM（0V）
		13	OUT4				

以插入面视角看各针脚位置



1.2 CN4(IN) /CN5(OUT) (RS485)

Pin.	信号名称	Pin.	信号名称
1	NC	2	GND
3	A Input (RS485)	4	NC
5	NC	6	B Input (RS485)
7	终端电阻（CN5）	8	终端电阻（CN5）

标准品：RJ45 类型 ×2

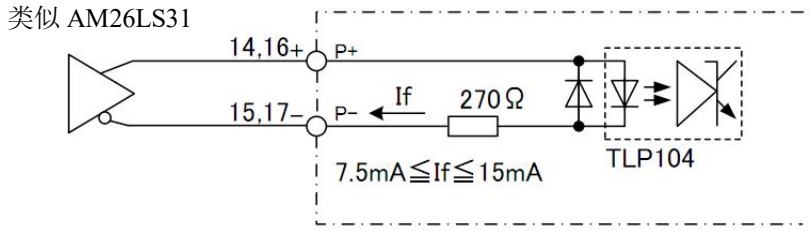
以面向插入视角看各针脚位置

注意：当使用多台串接时，在最后一台 CN5 的 3 脚和 8 脚短路、6 脚和 7 脚短路时，即为接入终端电阻。

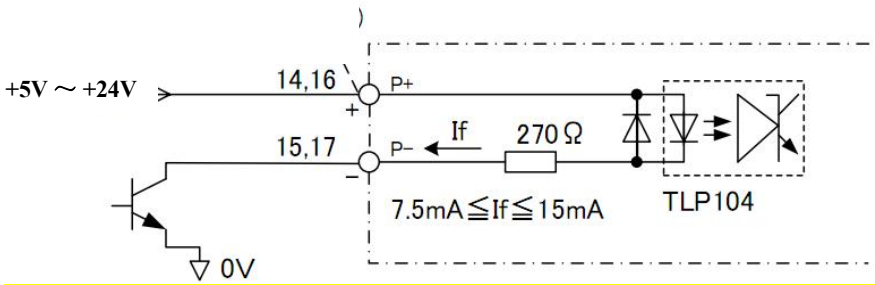
备注：CN4 不含终端电阻。

4 【输入回路图】

4.1 指令脉冲输入回路（差分驱动）

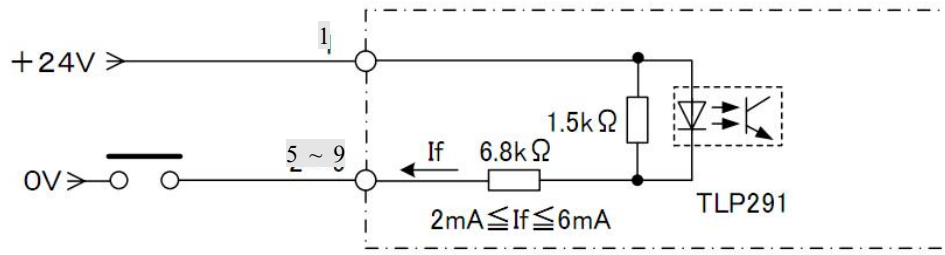


4.2 指令脉冲输入回路(集电极)

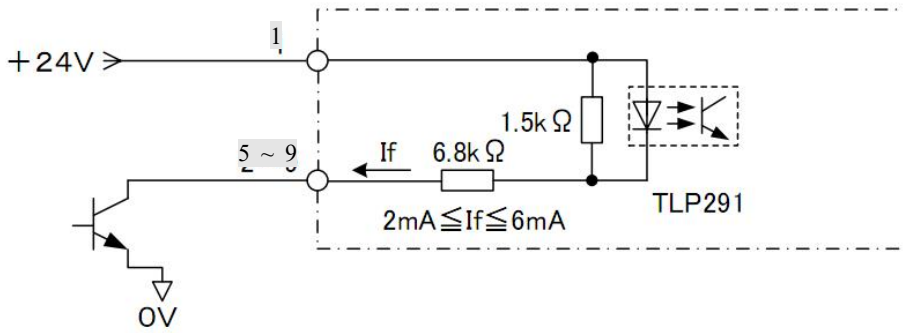


注） 本产品为+5V/+24V 信号兼容，24V 输入时无需串联限流电阻。

4.3 传感器、数字输入回路（接点）

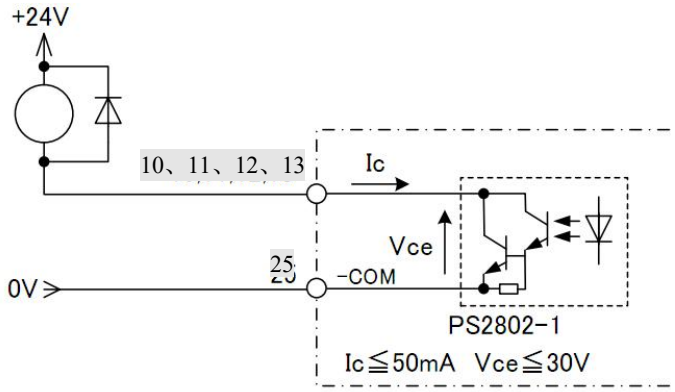


4.4 传感器、数字输入回路（集电极输出）



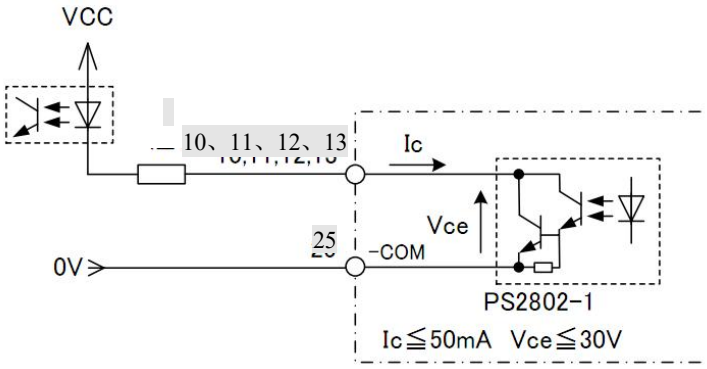
5 【输出回路图】

5.1 数字输出回路(继电器连接)

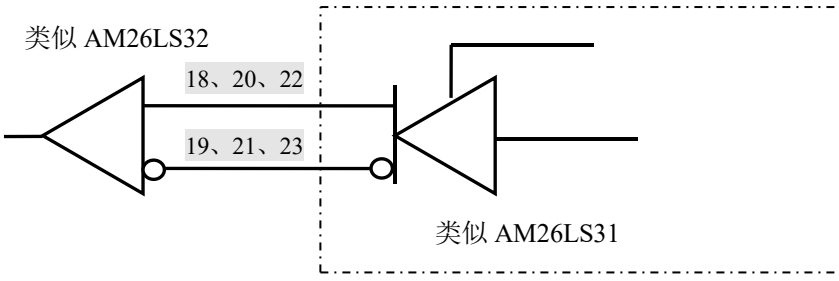


注意：继电器连接时，要求在继电器两端二极管（相当 IN4007）

5.2 数字输出回路(光耦连接)

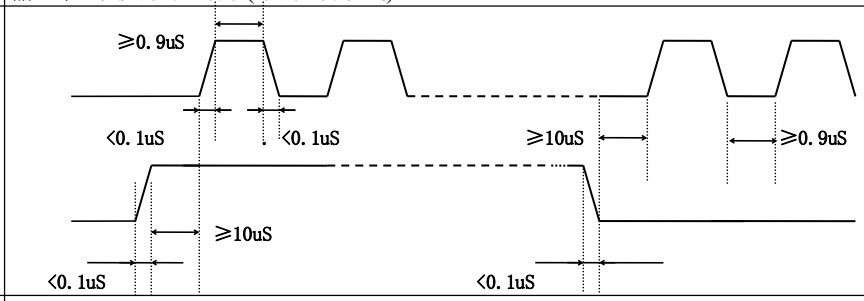
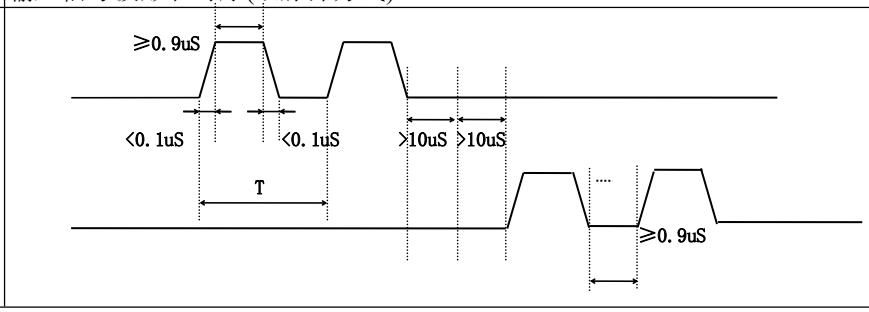


5.3 差分输出回路(编码器输出)



⚠ 注意	**编码器输出无光耦隔离，上电前请再次确认接线是否正确、有无短路情况，避免引入 CN3 端口上的 24V 电源，损坏上位机和驱动器。**

6 【脉冲/方向输入时序图】

	输入信号波形和时序(单脉冲方式)
STEP 输入口 DIR 输入口 （低电平正转， 高电平反转）	
	输入信号波形和时序(双脉冲方式)
STEP 输入口 （脉冲输入正转） DIR 输入口 （脉冲输入反转）	

7 【指示灯】

7.1 状态指示：

方式：完成不同状态下对应的闪烁（0.5 秒低电平，0.5 秒高电平）次数，完成 2 秒高电平，然后再循环。

状态功能	绿灯	通讯代码	说明
停止中	闪烁	2	开使能，电机锁相但电机未运行
运行中	常亮	3	驱动器在运行中
使能断开	闪烁	1	使能断开，电机可以自由

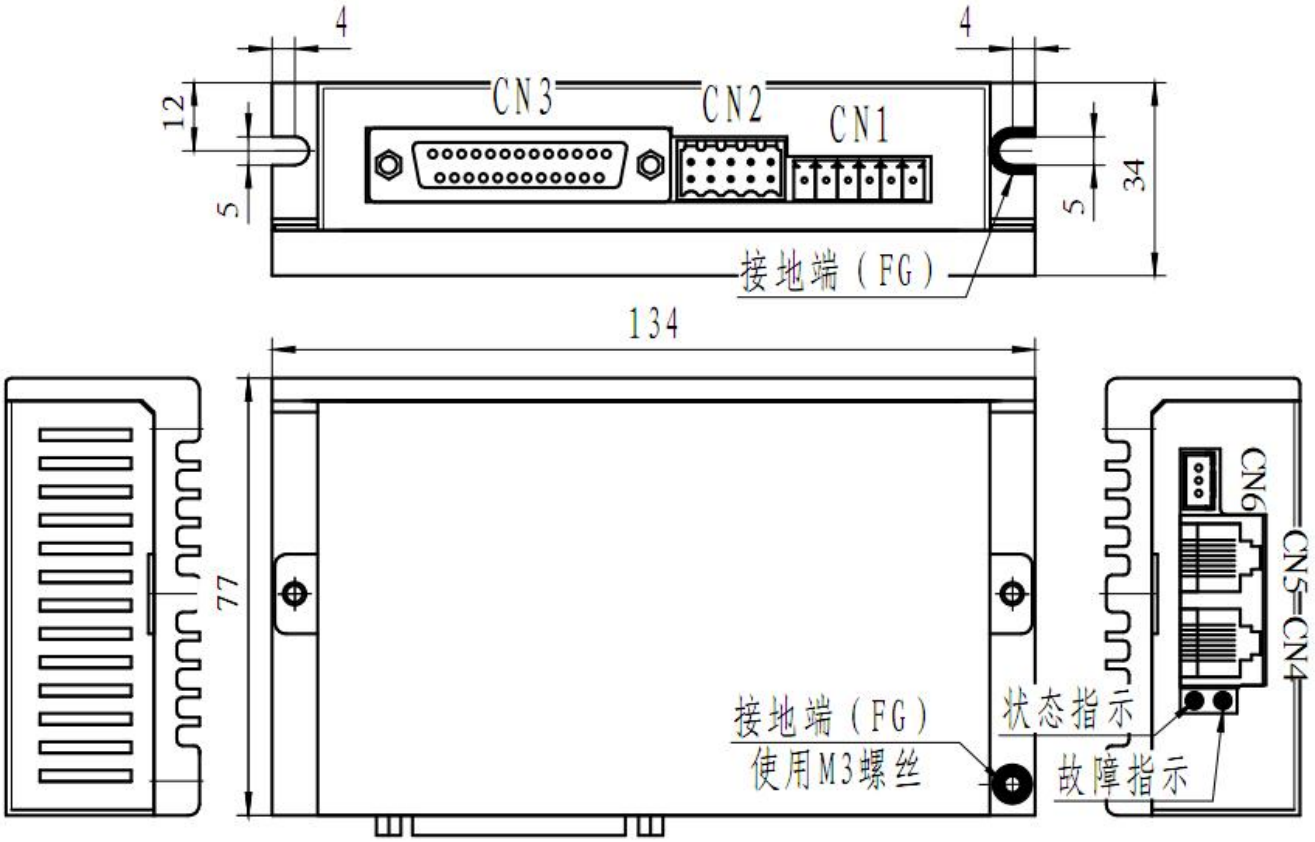
7.2 故障指示：

方式：完成不同状态下对应的闪烁（0.5 秒低电平，0.5 秒高电平）次数，完成 2 秒高电平，然后再循环。

报警功能	红灯	通讯代码	说明
电机过流	闪烁 1 次	10	电机相电流过流或驱动器故障
电机缺相	闪烁 2 次	11	电机没接
电机过流	闪烁 5 次	12	超过设定的补偿次数
欠压	闪烁 4 次	13	电源输入小于 18V
过压	闪烁 3 次	14	电源输入大于 60V
其他报警	其他	其他	

警告功能	红灯	通讯代码	说明
位置超差	常亮	25	位置偏差大于设定值

8 【外形尺寸（mm）】



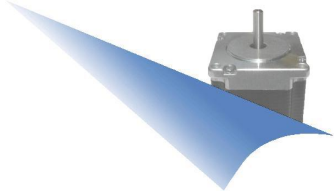
各功能表

端口号	用途
CN1	电机和电源接口
CN2	编码器输入接口
CN3	输入和输出接口
CN4	RS485 接口（IN）
CN5	RS485 接口（OUT）
CN6	备用

9 【控制参数】注：通讯参数非正式版本，部分参数固定未开放设置

9.1 控制器基本状态（分类 01）

adr	word	内容	详述	范围/单位
0100	1	电机电流	电机实时电流值	0.1%A
0101	1	输入电压	当前输入电压	1%V
0104	2	设置细分	设置细分值	ppr
0106	1	脉冲方式	1 为脉冲+方向模式、2 为双脉冲模式	1-2
0108	1	故障代码	报警时代码，内容见 1-2，显示“0”为无故障	-
0109	1	运行状态	驱动器运行状态，内容见 1-1	-
0110	1	硬件版本	驱动器硬件版本	-
0111	1	软件版本	驱动器软件版本	-
0117	2	当前位置	目标位置	pulse
0119	1	实际转速显示	-	0.01rps
0126	1	实际位置	运行实时位置	pulse
0174	1	IO 选择多段运行段落	-	-
0176	1	多段编写出错 No	-	-
0178	1	多段运行 No	-	-
0135	1	输入端口状态	数据位 Bit7 ..... Bit1 Bit0 输入端口 IN7 ..... IN2 IN1	
0136	1	输出端口状态	数据位 Bit3 Bit2 Bit1 Bit0 输出端口 OUT4 OUT3 OUT2 OUT1	



9.2 基本参数设置（分类 02）

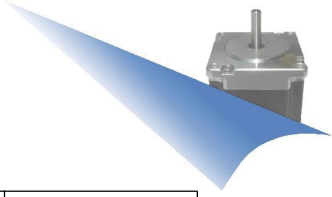
adr	word	内容	详述	范围/单位
0201	1	电机方向切换	选择电机运行方向	0～1
0213	1	半流比例	停止电流比例（开环模式有效）	10%~120%
0217	1	电机控制模式	0：开环 1：闭环 默认：1	0～1
0224	1	角度滤波	值越小，电机运行越平滑，但延迟也越高	1~700
0234	1	数字滤波	输入脉冲的滤波系数，值越大输入频响越低	1~15
0241	1	输入电流	设置电流	100~4500 0.1A~-4.5A
0242	2	设置细分	每圈脉冲数	200~102400 ppr
0244	1	脉冲方式	1：脉冲+方向模式 2：双脉冲模式	1~2
0245	1	半流时间	电机停止运行后进入半流状态的延时时间（开环模式有效）	1~32767 ms
0296	1	运行模式选择	0：外部脉冲 1：内部脉冲 默认：0 注：功能修改后需断电重启	0~1
0298	1	通讯地址	默认：1	1~255
0299	2	通讯波特率	默认：19200	1600~115200

9.3 闭环参数设置（分类 04）

adr	word	内容	详述	范围/单位
0246	1	编码器分辨率	分辨率=编码器线数 x 4	200~65535
0247	2	到位脉冲宽度	到达目标位置接近距离，输出到位信号 默认：0	1~1000 编码器分辨率
0251	1	速度环 Kp	速度环 Kp	0～30000
0252	1	速度环 Ki	速度环 Ki	0～30000
0255	1	位置环 Kp	位置环 Kp	0～30000
0258	1	位置超差阈值	以编码器分辨率为单位	0～30000 编码器分辨率

9.4 控制用参数（分类 05）

adr	word	内容	详述	范围/单位
0301	1	启动频率	默认：100	1~2000 0.01~20rps
0302	1	停止频率	默认：100	1~2000 0.01~20rps
0303	1	加速度	默认：100	5~10000 rps^2
0304	1	减速度	默认：100	5~10000 rps^2
0305	1	回原点模式	回原点模式，0：顺时针回原点 1：逆时针回原点	0~1
0306	1	定长运行速度	默认：1000	1~5000 0.01~50rps
0307	1	速度模式运行速度	速度模式时，运行方向与速度方向一致 默认：1000	-5000~5000 -50~50rps
0308	1	点动运行速度	默认：1000	1~5000 0.01~50rps
0309	1	回原点运行速度	默认：1000	1~5000

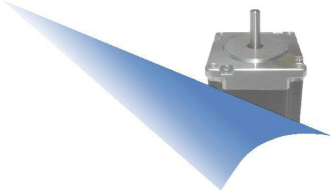


				0.01~50rps
0310	1	回原点蠕动速度	碰到原点后运行速度 默认：1000	1~5000 0.01~50rps
0311	2	回原点偏移量	默认：0	-2000000000~ 2000000000 pulse
0313	2	输出脉冲	运行行程 绝对位置模式：运行到指定位置 相对位置模式：运行设定偏移量行程 默认：0	-2000000000~ 2000000000 pulse
0317	2	正软限位	默认：2000000000 注：回原点过程中无效	-2000000000~ 2000000000 pulse
0319	2	负软限位	默认：-2000000000 注：回原点过程中无效	-2000000000~ 2000000000 pulse
0321	2	设置当前位置	默认：0	-2000000000~ 2000000000 pulse
0323	1	控制命令	0：空 1、绝对运行，运行到设定距离，运行方向由距离正负确定，速度正负值无效，在运行过程中修改目标位置有效 2、相对运行，以设定距离和运行速度运行，运行方向由距离正负确定，速度正负值无效，在运行过程中修改运动距离无效 3、速度模式 4、正向点动 5、反向点动 6、减速停止 7、急停 8、设定当前位置，只有在电机停止时才可以设置 12、回原点 13、报警清除 14：多段数据校验 15：多段数据保存 16：多段数据开始 17：多段数据暂停 18：多段数据结束 默认：0	0~29
0324	1	内部控制开关	<table><tr><td>数据位</td><td>Bit1</td><td>Bit0</td></tr><tr><td>功能</td><td>负软 限位</td><td>正软 限位</td></tr></table> 1：打开功能，0：关闭功能 默认：0	0-65535
0327	1	多段段落个数	默认：1	1~32
0328	1	多段选择	默认：0 注：若 IO 端口配置多段选择功能，以 IO 配置多段选择优先	0~31

9.5 输入块指定（分类 06）

adr	word	内容	详述	范围/单位
0400	1	IN1 功能选择	0：空	0~30

			1、绝对运行，运行到设定距离，运行方向由距离正负确定，速度正负值无效，在运行过程中修改目标位置有效 2、相对运行，以设定距离和运行速度运行，运行方向由距离正负确定，速度正负值无效，在运行过程中修改运动距离无效 3、速度模式 4、正向点动 5、反向点动 6、减速停止 7、急停 8、设定当前位置，只有在电机停止时才可以设置 9、正限位 10、负限位 11：原点信号 12、回原点 13、报警清除 14：多段数据校验 15：多段数据保存 16：多段数据开始 17：多段数据暂停 18：多段数据结束 20、使能 25：IO 端口配置多段选择 Bit0 26：IO 端口配置多段选择 Bit1 27：IO 端口配置多段选择 Bit2 28：IO 端口配置多段选择 Bit3 29：IO 端口配置多段选择 Bit4 默认：0	
0401	1	IN2 功能选择	设置内容同 IN1(默认值:0)	0~30
0402	1	IN3 功能选择	设置内容同 IN1 (默认值:0)	0~30
0403	1	IN4 功能选择	设置内容同 IN1 (默认值:0)	0~30
0404	1	IN5 功能选择	设置内容同 IN1 (默认值:0)	0~30
0405	1	IN6 功能选择（CCW 端口）	设置内容同 IN1 (默认值:0) （外部脉冲时，端口功能失效）	0~30
0406	1	IN7 功能选择（CW 端口）	设置内容同 IN1 (默认值:0) （外部脉冲时，端口功能失效）	0~30
0429	1	通用数字输入逻辑		
0410	1	伪通讯设定 IN1	0：OFF（初始值 0） 1：ON（触发 IN1 配置的动作）	0~1
0411	1	伪通讯设定 IN2	0：OFF（初始值 0） 1：ON（触发 IN1 配置的动作）	0~1
0412	1	伪通讯设定 IN3	0：OFF（初始值 0） 1：ON（触发 IN1 配置的动作）	0~1
0413	1	伪通讯设定 IN4	0：OFF（初始值 0） 1：ON（触发 IN1 配置的动作）	0~1
0414	1	伪通讯设定 IN5	0：OFF（初始值 0） 1：ON（触发 IN1 配置的动作）	0~1
0415	1	伪通讯设定 IN6	0：OFF（初始值 0） 1：ON（触发 IN1 配置的动作） （外部脉冲时，伪通讯端口功能失效）	0~1
0416	1	伪通讯设定 IN7	0：OFF（初始值 0） 1：ON（触发 IN1 配置的动作） （外部脉冲时，伪通讯端口功能失效）	0~1



输出块指定（分类 07）

<table>
<tr><th>adr</th><th>word</th><th>内容</th><th>详述</th><th>范围/单位</th></tr>
<tr><td>0420</td><td>1</td><td>OUT1 功能选择</td><td>100：通用端口
101：报警输出功能：
无报警时有输出信号，有报警时无输出信号。
102：到位信号
103：使能控制输出：
脱机时有输出信号，使能时无输出信号。
(默认值：101)</td><td>100~104</td></tr>
<tr><td>0421</td><td>1</td><td>OUT2 功能选择</td><td>设置内容同 OUT 1(默认值:100)</td><td>100~104</td></tr>
<tr><td>0422</td><td>1</td><td>OUT3 功能选择</td><td>设置内容同 OUT 1(默认值:100)</td><td>100~104</td></tr>
<tr><td>0423</td><td>1</td><td>OUT 4 功能选择</td><td>设置内容同 OUT 1(默认值:100)</td><td>100~104</td></tr>
<tr><td>0428</td><td>1</td><td>通用数字输出控制</td><td>输出端口功能选择 100<table><tr><td>数据位</td><td>Bit3</td><td>Bit2</td><td>Bit1</td><td>Bit0</td></tr><tr><td>输出端口</td><td>OUT4</td><td>OUT3</td><td>OUT2</td><td>OUT1</td></tr></table></td><td></td></tr>
<tr><td>0430</td><td>1</td><td>数字输出逻辑</td><td>对应输出端口逻辑<table><tr><td>数据位</td><td>Bit3</td><td>Bit2</td><td>Bit1</td><td>Bit0</td></tr><tr><td>输出端口</td><td>OUT4</td><td>OUT3</td><td>OUT2</td><td>OUT1</td></tr></table></td><td></td></tr>
</table>

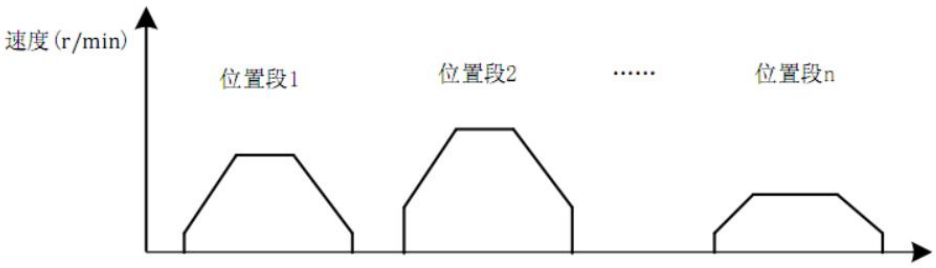
9.6 多段位置模式（分类 08）

多段地址 围 1024~1536，最多可设置 256 个数据

多段命令格式

命令码	word	内容	详述	范围/单位
1	2	绝对运行	参数 1：运行位置 默认： 0	-2147483647~ 2147483647 pulse
2	2	相对运行	参数 1：运行距离 默认： 0	-2147483647~ 2147483647 pulse
51	1	启动速度	默认：100	1~2000 0.01~20rps
53	1	停止速度	默认：100	1~2000 0.01~20rps
54	1	定长速度	默认：1000	1~5000 0.01~50rps
61	1	加速度	默认：100	5~10000 rps^2
62	1	减速度	默认：100	5~10000 rps^2
65	2	等待跳转	A（高 8 位）/B（低 8 位）/C（低 16 位）,A：固定为 0，B：跳转地址；C：等待时间	-
66	2	跳转序列	A（高 16 位）/B（低 16 位）,A：循环次数，B：跳转地址	-
100	1	多段结束	每个段落结束都需以结束码作为结束标志	-

多段位置模式功能是将多个位置段按一定顺序组合起来，通过外部 IO 信号触发运动，完成一系列位置段动作的一种工作方式。该功能可看作是位置模式的多段组合，用户可以将若干段位置段的描述参数如加减速，脉冲数等事先存储于 EEPROM 中，需要使能这些位置段时只需提供一个触发信号即可完成工作，其工作过程描述如下图所示。



多段位置工作模式

端口选择对应多段

Bit4	Bit3	Bit2	Bit1	Bit0	位置段
0	0	0	0	0	1
0	0	0	0	1	2
0	0	0	1	0	3
0	0	0	1	1	4
…	…	…	…	…	…
1	1	1	0	1	30
1	1	1	1	0	31
1	1	1	1	1	32

IO 选择端口

1、 输入端口配置多段选择功能 25~29：IO 端口配置多段选择 Bit0~ Bit4
输入端口配置多段开始功能 15：多段数据开始

2、 端口选择对应多段

例： IN1 端口功能配置 25，Bit0
IN3 端口功能配置 26，Bit1
可根据需求配置 IN1~ IN7 功能

IN3 Bit1	IN1 Bit0	位置段
0	0	1
0	1	2
1	0	3
1	1	4

注： 表格中“1”表示有效保持信号
段落选择信号需提前开始信号 20ms 以上时间完成

例： 多段参数编写、校验及保存 ***注：例中数据以 16 进制表示**

1、多段参数设置

命令 1、当前行号 0：定长速度设置 1000，即 10rps，

01	10	04 00	00 02	04	00 36	03 e8	21 DF
①	②	③	④	⑤	⑥	⑦	⑧

① :通讯地址 0x1；
② :MODBUS 写命令 0x10；
③ :通讯地址 0x400（十进制表示 1024）；
④ :写 2 个数据；
⑤ :写 4 个字节；
⑥ :数据 1，定长速度命令 0x0036(十进制表示 54)；
⑦ :数据 2，定长速度值 0x03E8(十进制表示 1000)；
⑧ :CRC 校验；

命令 2、当前行号 1：相对运行，运行距离 10000 脉冲

01	10	04 02	00 03	06	00 02	27 10 00 00	20 CB
①	②	③	④	⑤	⑥	⑦	⑧

① :通讯地址 0x1；
② :MODBUS 写命令 0x10；
③ :通讯地址 0x402（十进制表示 1026）；
④ :写 3 个数据；
⑤ :写 6 个字节；
⑥ :数据 1，相对运行命令 0x0002(十进制表示 2)；
⑦ :数据 2，参数：运行脉冲值 0x2710(十进制表示 10000)；
⑧ :CRC 校验；

命令 3、当前行号 2：等待 1000ms，

01	10	04 05	00 03	06	00 41	03 E8 00 03	1F DE
①	②	③	④	⑤	⑥	⑦	⑧

① :通讯地址 0x1；
② :MODBUS 写命令 0x10；
③ :通讯地址 0x405（十进制表示 1029）；
④ :写 3 个数据；
⑤ :写 6 个字节；
⑥ :数据 1，相对运行命令 0x0041(十进制表示 65)；
⑦ :数据 2， 数据 03 E8 00 03 转换为 00 03 03 E8 *注：4 个字节数据，低 16 位在前，高 16 位在后
A B C
参数 A：系统预留默认为 0，请勿设置值；
参数 B：等待跳转行 3，当前等待命令行为 2；
参数 C：等待时间 0x03E8(十进制表示 1000ms)；
⑧ :CRC 校验；

命令 4、当前行号 3：再循环执行相对运行 10 次，

01	10	04 08	00 03	06	00 42	00 01 00 0A	DB 92
①	②	③	④	⑤	⑥	⑦	⑧

① :通讯地址 0x1；
② :MODBUS 写命令 0x10；
③ :通讯地址 0x408（十进制表示 1032）；
④ :写 3 个数据；
⑤ :写 6 个字节；
⑥ :数据 1，相对运行命令 0x0042(十进制表示 66)；
⑦ :数据 2， 数据 00 01 00 0A 转换为 00 0A 00 01 *注：4 个字节数据，低 16 位在前，高 16 位在后
A B
参数 A：动作跳转次数 0xA(十进制表示跳转 10 次)；
参数 B：跳转行 1，再次执行相对运行；
⑧ :CRC 校验；

命令 5、当前行号 4：动作结束，

01	06	04 0B	00 64	F8 D3
①	②	③	④	⑤

① :通讯地址 0x1；
② :MODBUS 写命令 0x06；
③ :通讯地址 0x40B（十进制表示 1035）；
④ :数据 段结束 0x64(十进制表示 100)；
⑤ :CRC 校验；

2、多段参数校验

01	06	01 43	00 0E	F8 26
①	②	③	④	⑤

① :通讯地址 0x1；

② :MODBUS 写命令 0x06；

③ :通讯地址 0x0143（十进制表示 323，写通讯命令）；

④ :数据 多段数据校验 0xE(十进制表示 14)；

⑤ :CRC 校验；

3、多段参数保存 ***注：数据校验成功后才可进行数据保存，否则数据无法正常保存**

01	06	01 43	00 0F	39 E6
①	②	③	④	⑤

① :通讯地址 0x1；

② :MODBUS 写命令 0x06；

③ :通讯地址 0x0143（十进制表示 323，写通讯命令）；

④ :数据 多段数据校验 0xF(十进制表示 15)；

⑤ :CRC 校验；

【报文格式】

MODBUS 协议定义了一个与基础通信层无关的协议数据单元（PDU）和在 RS485 物理层上的 MODBUS 协议映射在应用数据单元（ADU）上。

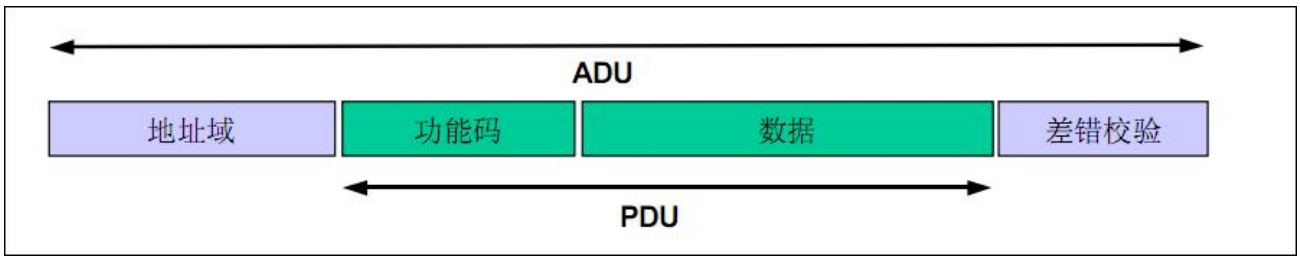


图 1 ：通用 MODBUS 帧

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 byte) + CRC (2 字节) = 256 字节。

MODBUS 协议定义了三种 PDU。它们是：

a 、MODBUS 请求 PDU，mb_req_pdu

b 、MODBUS 响应 PDU，mb_rsp_pdu

c 、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。

10 【MODBUS 事务处理】

10.1 MODBUS 事务处理的定义

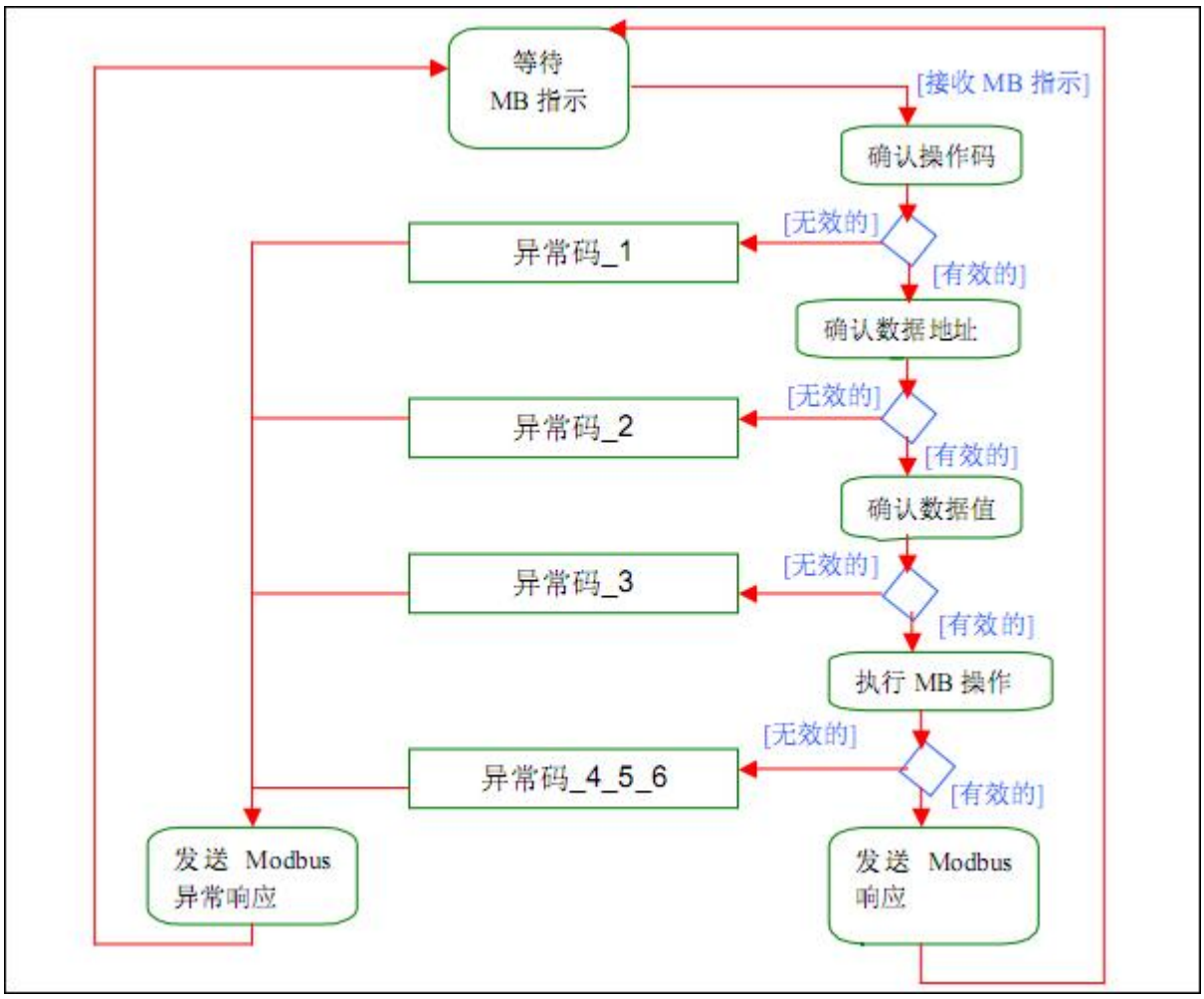


图 2：MODBUS 事务处理的状态图

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：

1 一个 MODBUS 正常响应：

响应功能码 = 请求功能码

2 一个 MODBUS 异常响应：

a、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；

b、响应功能码 = 请求功能码 + 0x80；

c、提供一个异常码来指示差错原因。

10.2 MODBUS 正常响应

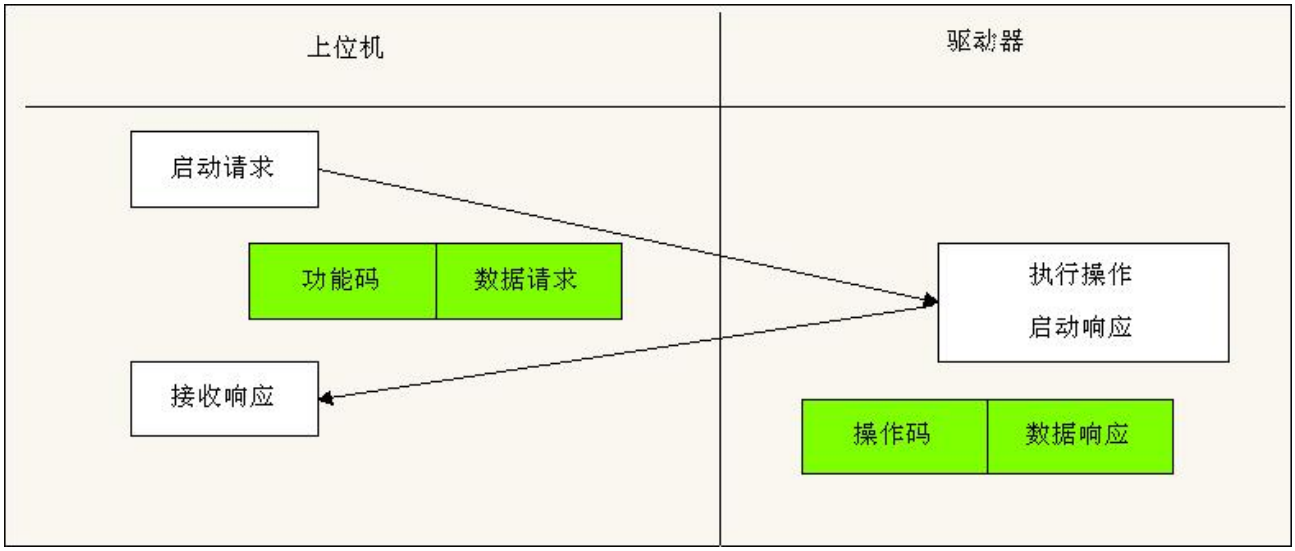


图 3：MODBUS 事务处理（无差错）

10.3 MODBUS 异常响应

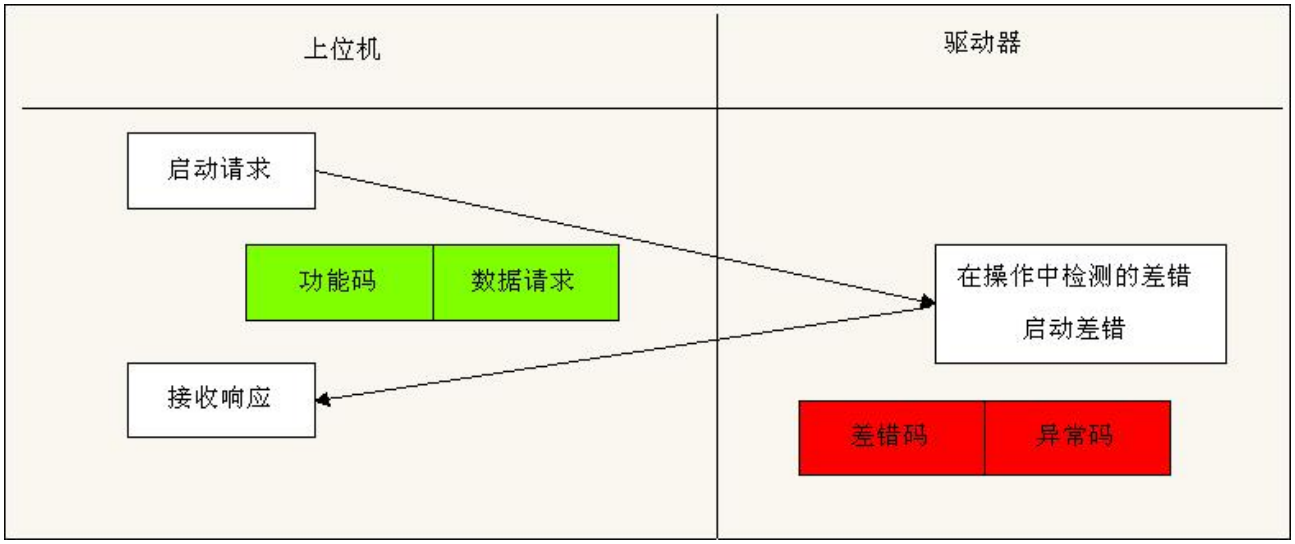
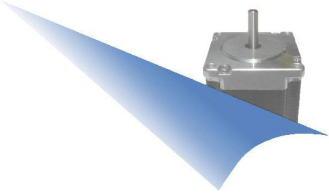


图 4：MODBUS 事务处理（异常响应）

11 【数据编码】

MODBUS 使用一个‘big-Endian’规范（即高位在前，低位在后） 表示地址和数据项。



12 【公共功能码定义及功能码描述】

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

斯达普公司根据通信需求常用功能码为上表黄色部分

03 (0x03)读保持寄存器

06 (0x06)写单个寄存器

16 (0x10) 写多个寄存器

12.1 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	03	功能	03
高起始地址	00	字节数	06
低起始地址	6B	寄存器值 Hi（108）	02
高寄存器编号	00	寄存器值 Lo（108）	2B
低寄存器编号	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。

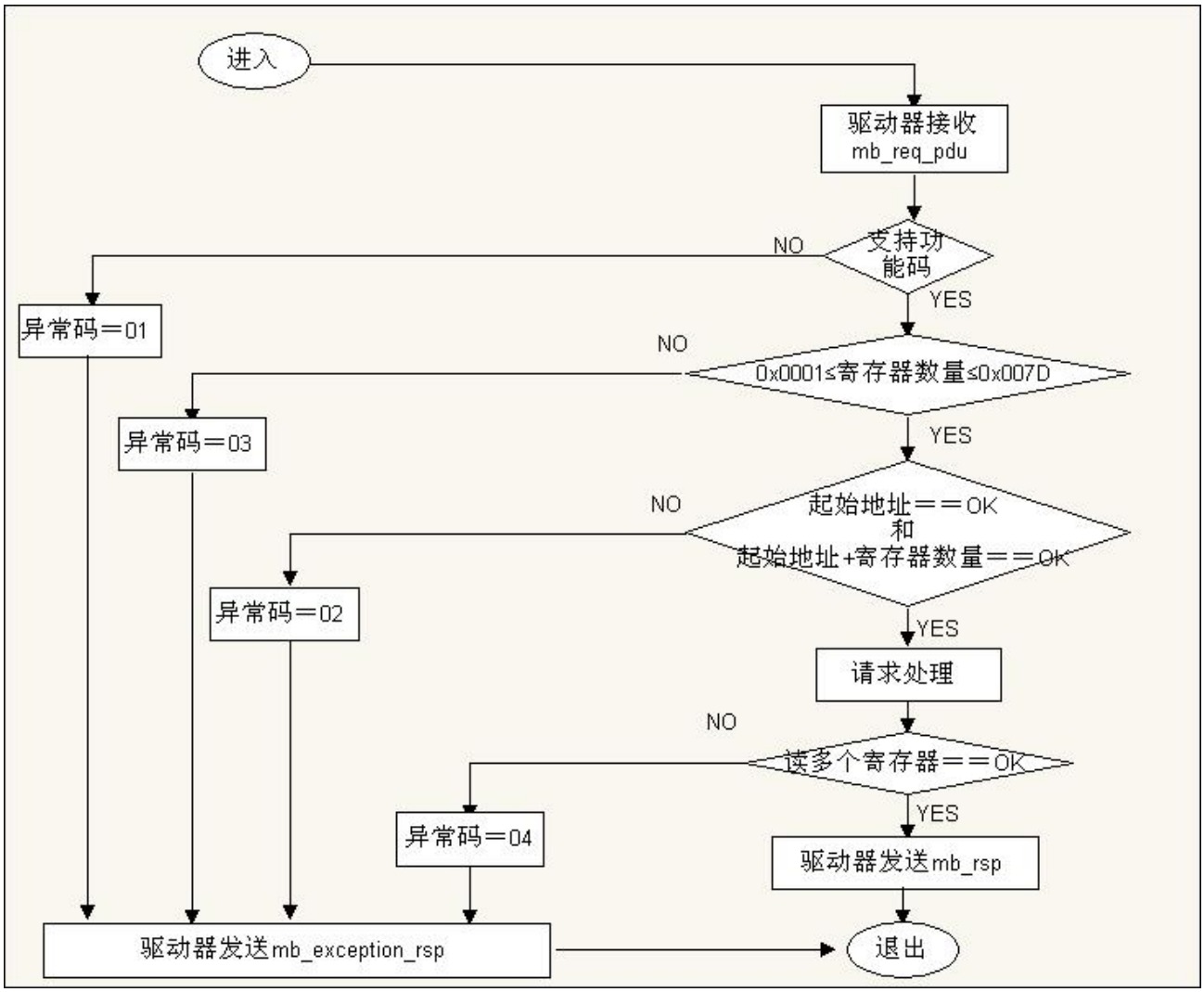


图 5：读保持寄存器的状态图

12.2 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	0x03
寄存器地址	2 个字节	0x0000 至 0xFFFF
寄存器值	2 个字节	0x0000 至 0xFFFF

*N＝寄存器的数量

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
寄存器地址 Hi	00	输出地址 Hi	00
寄存器地址 Lo	01	输出地址 Lo	01
寄存器值 Hi	00	输出值 Hi	00
寄存器值 Lo	03	输出值 Lo	03

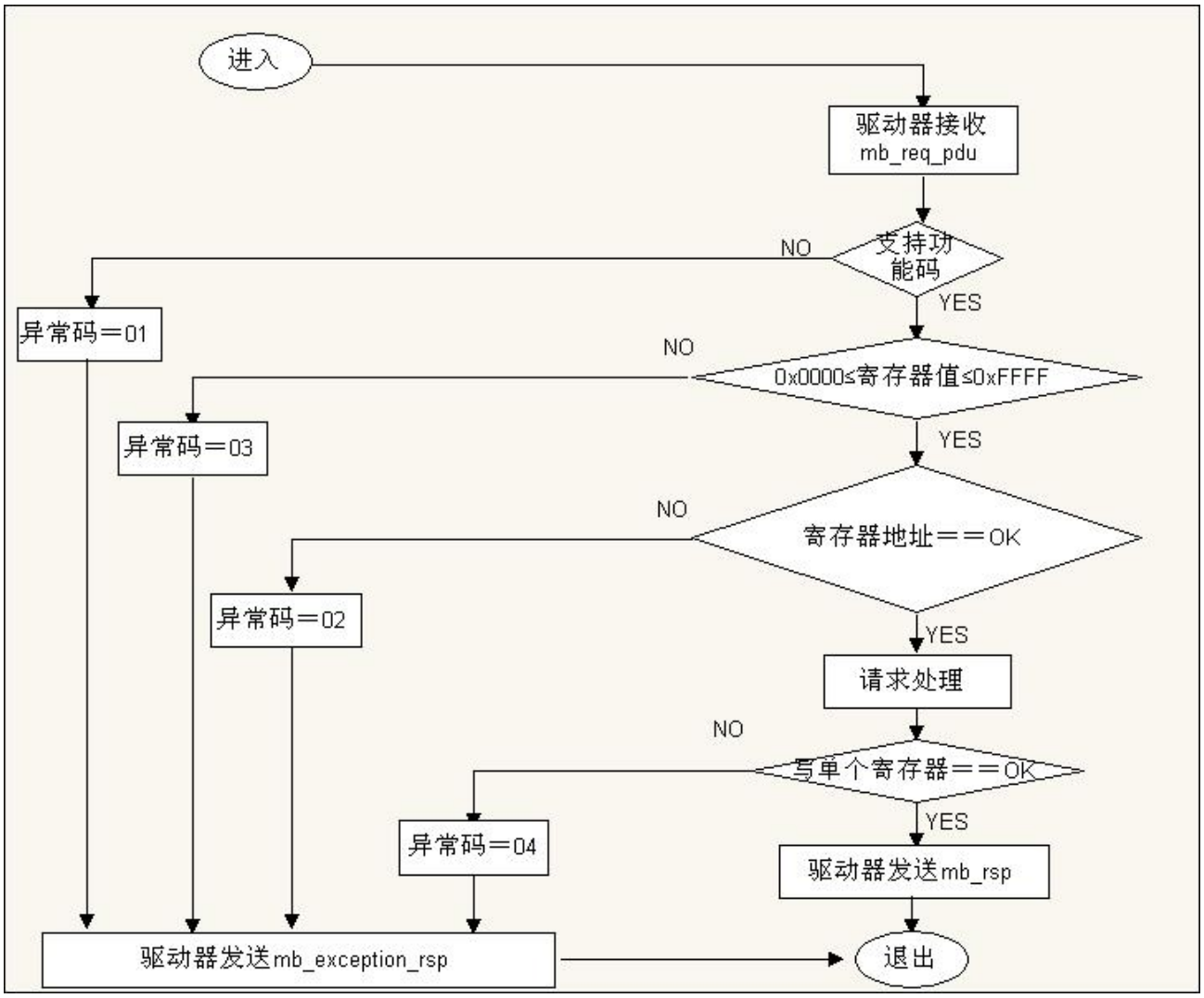


图 6：写单个寄存器状态图

12.3 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

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

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）

*N＝寄存器的数量

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		

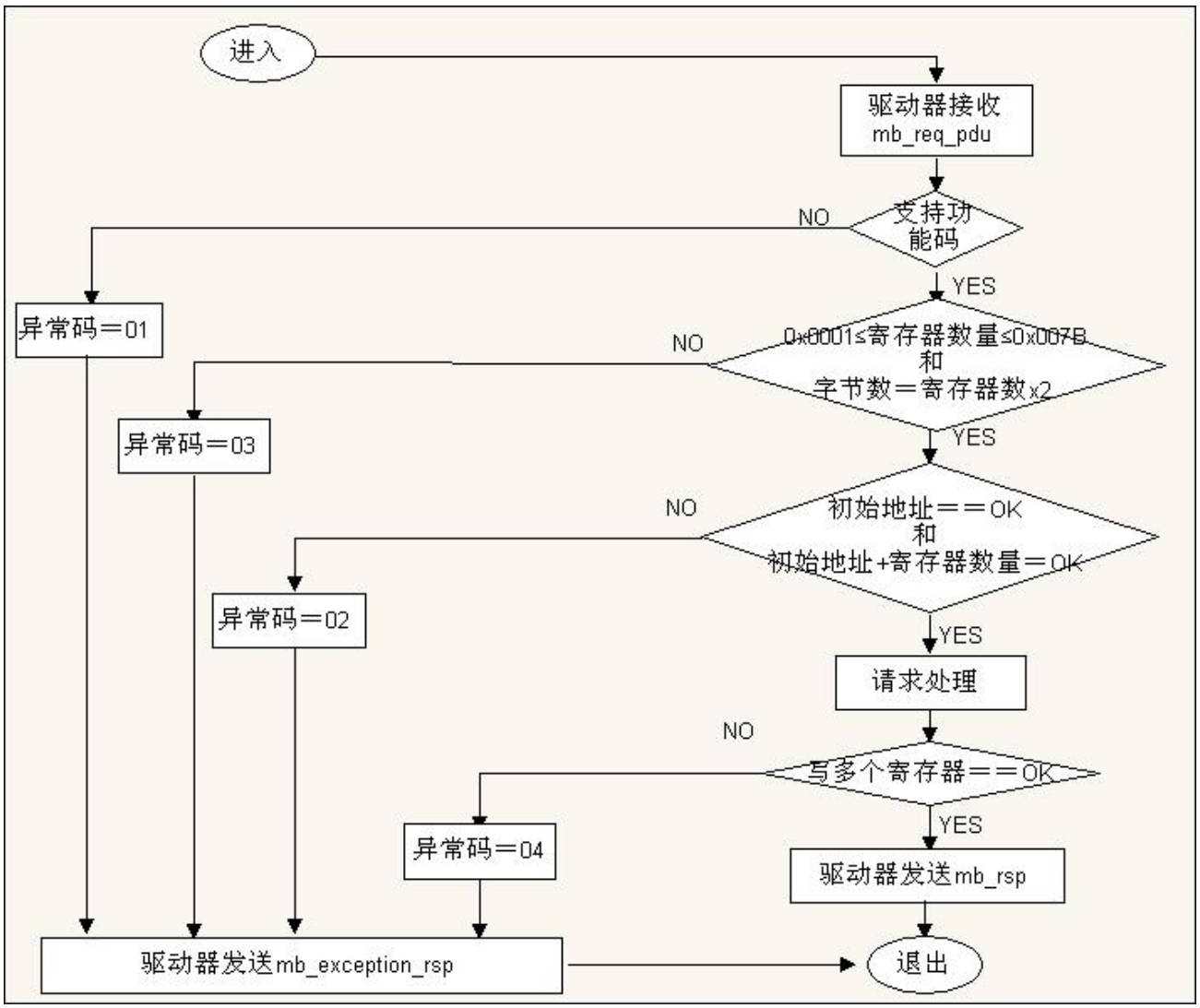


图 7：写多个寄存器状态图

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	当驱动器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。张服务器(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。
0C	字节之间发送超时	发送的同一帧数据中如果发送的上一个字节与下一个字节之间间隔的时间大于 1.5 个字符，则出错，当前这帧数据丢弃，等待 3.5 个字符内未接收到数据发送错误码，若一直有数据发送需等待到 3.5 个字符内都无数据接收再发送错误码，在此之间接收到的数据都丢弃。
0D	帧之间发送小于最小间隔时间	成功接收完一帧数据，在小于 3.5 个字符内又接收到数据，则出错，等待 3.5 个字符内未接收到数据发送错误码，若一直有数据发送需等待到 3.5 个字符内都无数据接收再发送错误码，在此之间接收到的数据都丢弃。

13 【MODBUS 主节点工作模式】

主节点以两种模式对子节点发出 Modbus 请求:

a、 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。

b、 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。

14 【MODBUS 地址规则】

Modbus 寻址空间有 256 个不同地址。

0	1-47	48-255
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。

注：通讯地址=拨码开关的值+1(驱动器地址不能为 0)

15 【主站/从站通信时序图】

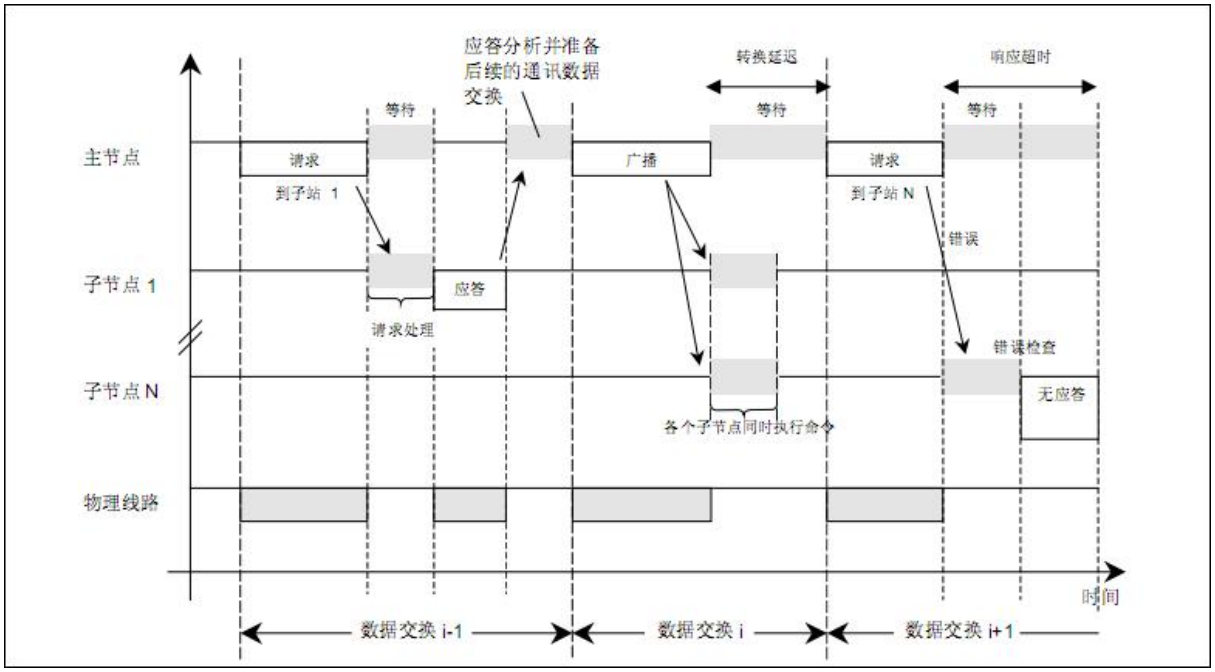


图 8：各种情形的主/从通信时序图

16 【RTU 传输模式】

RTU 模式每个字节 (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 起始位

8 数据位， 首先发送最低有效位

1 位作为奇偶校验

1 停止位

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。

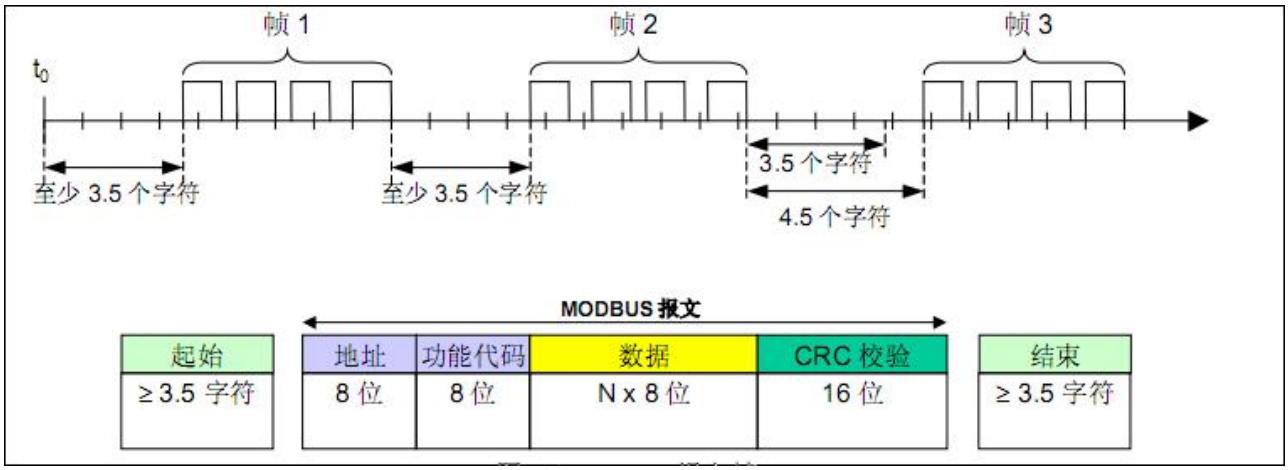


图 9：RTU 报文帧

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。

波特率	报文帧空闲分隔区	字节之间空格	响应超时	转换延迟
>=19200 bps	>=2ms	<=0.8ms	1s	200ms
14400 bps	>=2.7ms	<=1.1ms	1s	200ms
9600 bps	>=4ms	<=1.7ms	1s	200ms
4800 bps	>=8ms	<=3.4ms	1s	200ms
2400 bps	>=16ms	<=6.8ms	1s	200ms

17 【CRC 校验】

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。

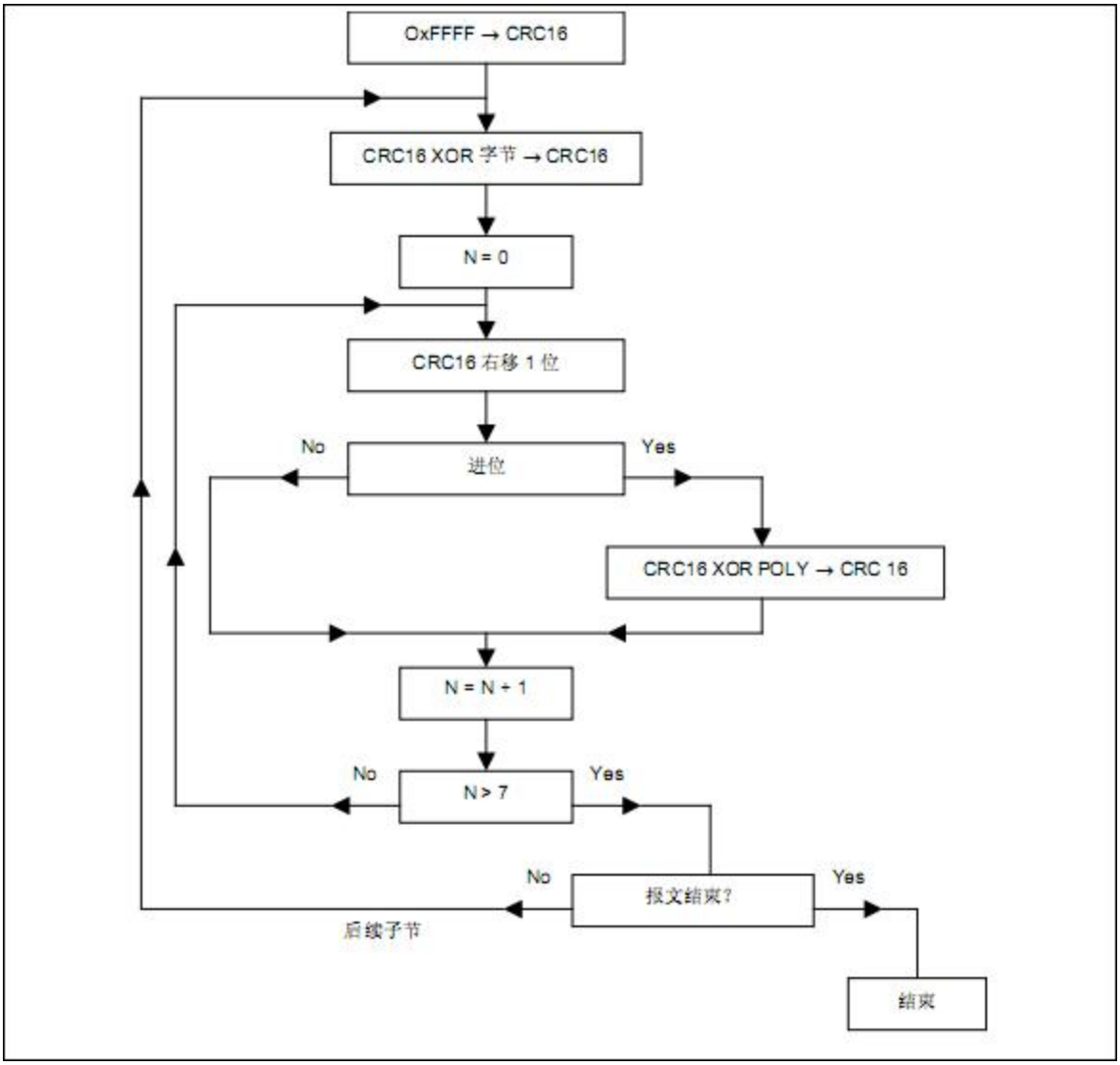


图 10：CRC 16 计算算法

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+x_2 +x_{15} +x_{16})

在 CRC 16 中， 发送的第一个字节为低字节。

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, 0 x00, 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, 0x80, 0x41, 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
};

18 【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。

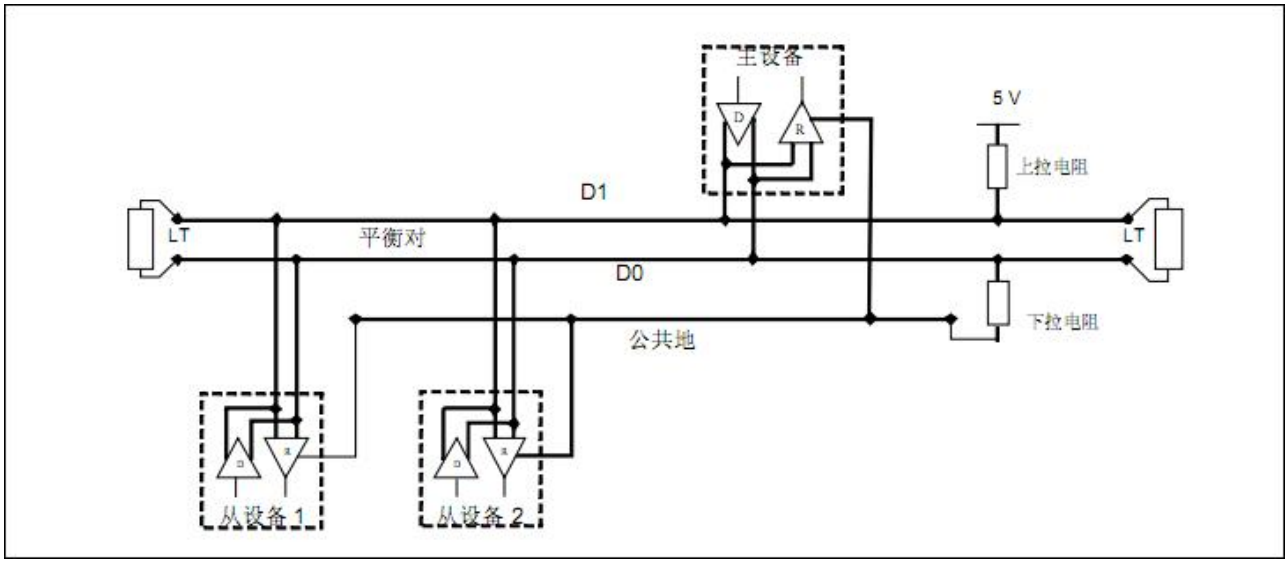


图 11：2-线制的一般拓扑结构