通信模块RS422接口定义

1. 修订历史

日期	说明	作者
	create this	wangzhuang
2025.1.17	增加写文件说明	zhangfeng、cuijingtao

2. 通信速率

波特率：115200

启动位：0bit

停止位：1bit

数据位：8bit

校验位：无

硬件流控制：无

3. 消息格式

3.1. 消息格式定义

消息参考modus实现，但是也不全一样，内通信的消息格式定义如下

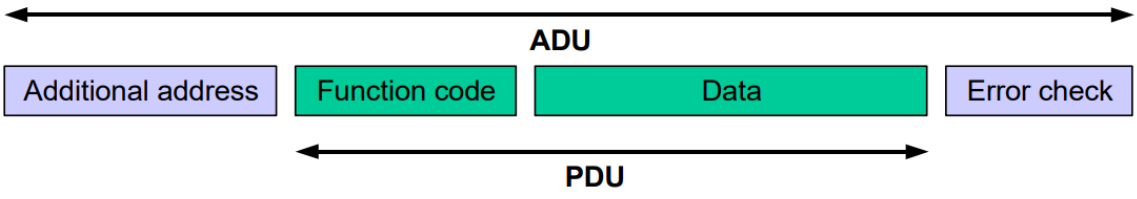


Figure 3: General MODBUS frame

从站地址（）	Function Code	Data	Error check
1Byte	1Byte	由FunctionCode确定	2Byte CRC校验

消息由主控板创建，功能码用于知识从机执行的动作。

消息交互流程如下：

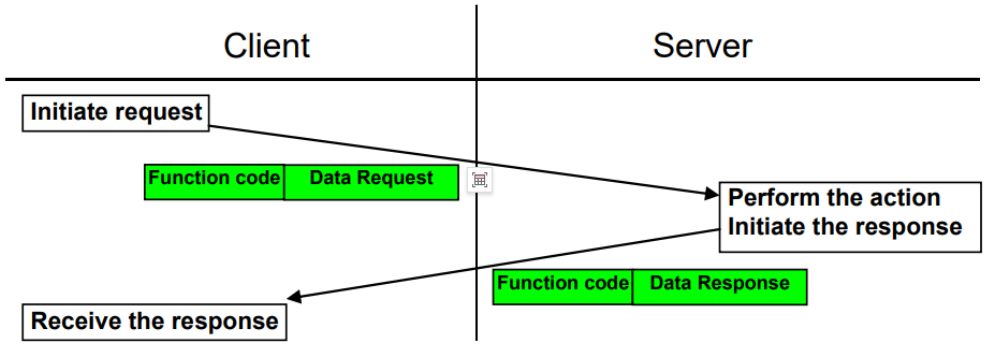


Figure 4: MODBUS transaction (error free)

For an exception response, the server returns a code that is equivalent to the original function code from the request PDU with its most significant bit set to logic 1.

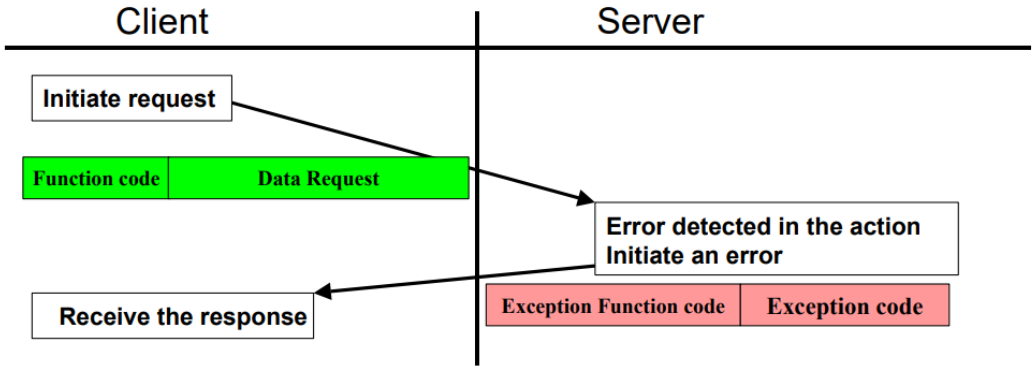


Figure 5: MODBUS transaction (exception response)

通信模块的从站地址

从站地址	说明
0x40	通信模块使用0x40作为从机地址

消息的长度：

RS422 ADU = 从机地址（1byte） 功能码(1Byte) + 数据：252byte + + CRC(2Bytes) = 256Byte

数据编码：消息内容全部采用**大端形式**发送

3.1.1. Function Code描述

Function Code(10进制)	说明
65 (0x41)	读数据操作
66 (0x42)	写数据操作

3.1.1.1. 0x41 读数据操作

该功能码用于读数据操作，关于读那一项数据由具体的数据地址（2Byte）确定，关于数据地址对应哪个数据，由从机和主控开发人员确定；

Request

Field	大小	取值说明
Function Code	1 Byte	0x41
Data Address	2 Byte	0x0000 to 0xFFFF 关于数据地址对应数据结构详见后续章节
Data Len	1 Byte	配置的预期大小Byte

Response

Field	大小	取值说明
Function Code	1 Byte	0x41
Data Len	1 Byte	N
Data	N Byte	N * Bytes

Error Response

Field	大小	取值说明
Error Code	1 Byte	0x41 + 0x80
异常码	1 Byte	由从机自定义，如果出现错误则必须返回详细区分的错误码以方便问题定位

读数据举例

假设某数据的地址为0x0016，其数据内容结构体如下所示：

```
typedef struct {
    U8 d8;    // 0x00
    U8 d81;   // 0x01
    U16 d16;  // 0x0203
    U32 d32;  // 0x04050607
}Device5Data0x0016;
```

Request		Response	
Field Name	(Hex)	Field Name	(Hex)
从站地址	0x05	从站地址	0x05
Function	0x41	Function	0x41
Data Address Hi	0x00	Data Address Hi	0x00
Data Address Lo	0x16	Data Address Lo	0x16
Data Len	0x08（8字节）	Data Len	0x08（8字节数据大小）
CRC Hi	xx	Data[0]	0x00
CRC Lo	xx	Data[1]	0x01
		Data[2]	0x02 (d16 hi)
		Data[3]	0x03 (d16 lo)
		Data[4]	0x04 (d32 [31:24])
		Data[5]	0x05 (d32 [23:16])
		Data[6]	0x06 (d32[15:8])
		Data[7]	0x07 (d32 [7:0])
		CRC Hi	y
		CRC Lo	y

3.1.1.2. 0x42 写数据操作

该功能码用于写数据操作，关于写那一项数据由具体的数据地址（2Byte）定义，关于数据地址对应哪个数据由从机和主控板开发人员确定；

Request

Field	大小	取值说明
Function Code	1 Byte	0x42
Data Address	2 Byte	0x0000 to 0xFFFF
Data size	1 Byte	N， 数据大小Byte
Data	N * Byte	数据内容

Response

Field	大小	取值说明
Function Code	1 Byte	0x42
Data Address	2 Byte	0x0000 to 0xFFFF
Data size	1 Byte	N， 数据大小Byte
Data	N * Byte	数据内容

Error

Field	大小	取值说明
Error Code	1 Byte	0x42 + 0x80
异常码	1 Byte	由从机自定义，如果出现错误则必须返回详细区分的错误码以方便问题定位

写数据举例

假设主机要向从机写一个数据地址为0x0016的数据，假设数据内容结构体如下所示：

```
typedef struct {
    U8 d8;    // 0x00
    U8 d81;   // 0x01
    U16 d16;  // 0x0203
    U32 d32;  // 0x04050607
}Device5Data0x0016;
```

写的响应为写的请求的内容，需要原封不动的传回，以便确认数据正确写入

Request/Response	
Field Name	(Hex)
从站地址	0x05
Function	0x42
Data Address Hi	0x00
Data Address Lo	0x16
Data Len	0x8（8字节）
Data[1]	0x01
Data[2]	0x02 (d16 hi)
Data[3]	0x03 (d16 lo)
Data[4]	0x04 (d32 [31:24])
Data[5]	0x05 (d32 [23:16])
Data[6]	0x06 (d32[15:8])
Data[7]	0x07 (d32 [7:0])
CRC Hi	y
CRC Lo	y

3.1.1.3. 0x64 写文件操作

写文件操作需要配置数据读写操作进行，主要涉及

- 写文件操作开始
- 写文件操作终止
- 写文件操作结束

该指令码用于文件传输，文件传输流程需要配合读写数据指令进行，其流程如下所示

流程详解：

步骤1：开始

步骤2：主机发送写文件操作开始请求，从机收到请求后，保存相关信息，进入写文件操作状态，申请缓存；回复响应

步骤3：主机接收写文件操作开始响应，主机根据响应结果决定后续步骤

步骤4：主机发送写文件操作请求，PDU格式详见下文定义，从机收到请求后，回复响应，详见下文定义

步骤5：主机接收写文件操作响应，，主机根据响应结果决定后续步骤

步骤6，7：可选，当主机在文件传输中遇到异常或者收到终止指令后，通知从机终止文件传输，此种情况从机应该释放缓存

步骤8，9：同步骤3，4 主机持续发送文件直到文件全部写入到从机

步骤10： 当文件写入完毕，主机发送消息通知从机写操作结束，从机根据文件的类型，决策文件的处理原则，eg：如果文件配置类，其配置行为由从机自行完成；如果文件为软件版本，则主机会根据响应情况，决策是否发起软件重构流程（详见后文）

步骤11：主机接收写操作完结束响应，根据响应情况和约定的文件处理原则执行后续步骤；

步骤12： 结束

写文件请求的PDU格式（由主机发送给从机）如下所示：

Request PDU

Field	大小	取值说明
Function Code	1byte	0x64
Sequence	1byte	帧序号 0x00~0xFF循环，文件收包首帧以0x00开始，如果从机收到的序号相同，则表明为主机的重传
Data Len	1byte	当前帧传输文件数据的长度L
Data 0	1byte	
Data 1	1byte	
…		
Data L-1	1byte	

Response PDU， 即如果从机正确接收PDU则原封不动的把ADU传回，否则发送异常响应ADU

Field	大小	取值说明
function Code	1byte	0x64
Sequence	1byte	同发送
Data Len	1byte	同发送
Data 0	1byte	同发送
Data 1	1byte	同发送
…		同发送
Data L-1	1byte	同发送

Error PDU

Field	大小	取值说明
Function Code	1 byte	0x64 + 0x80
异常码	1 Byte	由从机自定义，如果出现错误则必须返回详细区分的错误码以方便问题定位

3.1.1.4. 0x65 读文件操作

读文件操作需要需要配合相关数据读写功能实现，主要涉及

- 读文件操作开始
- 读文件操作终止
- 读文件操作结束

此功能可选，当从机有文件被读的诉求（eg：日志或者计数器信息）时，可以按照协议实现读文件操作；

读文件操作**Request PDU**

Field	大小	取值说明
Function Code	1byte	0x65- 读文件功能码
Sequence	1byte	帧序号 0x00~0xFF循环，首帧以0x00开始，如果从机收到的序号相同，则表明主机希望从机重传该序号的数据

读文件操作**Response PDU**

Field	大小	取值说明
Function Code	1byte	0x65- 读文件功能码
Sequence	1byte	和读请求一致，如果收到重复的请求则重传该序号对应的数据，否则发送新数据
DataLen	1byte	从机传输Sequence对应的数据长度，N-Byte， 取值范围： 0~250，如果长度不为0则后续数据域应和长度指示保持一致，如果长度为0则说明当前从机无后续数据可读（即从机表明整个文件已经被读走）。
Data 0	1byte	当DataLen>=1时有效
Data 1	1byte	当DataLen>=2时有效
…		
Data N-1	1byte	当DataLen==N时有效
CRC-Flag	1byte	当DataLen==0时有效，标识传输的文件是否需要执行CRC校验; 0-不需要CRC校验；1-需要CRC校验,
CRC-hi	1byte	当DataLen==0且CRC-Flag置1时有效，当前已传输文件的CRC高8byte，整个文件的CRC16-CCITT-FALSE校验
CRC-lo	1byte	当DataLen==0且CRC-Flag置1时有效，当前已传输文件的CRC低8byte，整个文件的CRC16-CCITT-FALSE校验

读文件操作 **Error PDU**

Field	大小	取值说明
Function Code	1 byte	0x65 + 0x80
异常码	1 Byte	由从机自定义，如果出现错误则必须返回详细区分的错误码以方便问题定位

注：主机一次只会读一种类型的文件；

3.1.1.5. 异常响应

读写操作的功能码+0x80即为异常码

从机地址	功能码	异常码	CRC
0x40	请求功能码+0x80	U8- 用户自定义	U16-详见CRC校验说明

3.1.1.6. CRC校验说明

CRC校验ADU头开始直到数据结束为，CRC校验结果放在ADU的尾部，高字节在前，低字节在后；

C1C校验采用下述算法进行

```
static const uint8_t aucCRCHi[] = {
    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, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40
};

static const uint8_t aucCRCLo[] = {
    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
};

uint16_t WX_Modbus_Crc16( uint8_t *pFrame, uint16_t len )
{
    uint8_t           ucCRCHi = 0xFF;
    uint8_t           ucCRCLo = 0xFF;
    int             iIndex;

    while( len-- )
    {
        iIndex = ucCRCLo ^ *( pFrame++ );
        ucCRCLo = ( uint8_t )( ucCRCHi ^ aucCRCHi[iIndex] );
        ucCRCHi = aucCRCLo[iIndex];
    }
    return ( uint16_t )( ucCRCHi << 8 | ucCRCLo );
}
```

3.2. 读数据地址及数据结构定义

3.2.1. 地址0x0001 读取设备自检结果

序号	参数	参数大小	参数说明
1	自检结果	U8	0为自检成功，1为自检失败

3.2.2. 地址0x0002 读取快遥数据

序号	参数	参数大小	单位	参数说明
1	解调锁定	U8	/	
2	帧同步	U8	/	
3	译码锁定	U8	/	
4	主以太网业务接口连接状态	U8	/	
5	备以太网业务接口连接状态	U8	/	
6	激光链路纠前误码率	S16	/	bit15~bit8:底数部分，最高位符号位；bit7~bit0:指数部分，最高位符号位
7	激光链路纠后误码率	S16	/	bit15~bit8:底数部分，最高位符号位；bit7~bit0:指数部分，最高位符号位
8	发送测距帧出发时间	U72	ps	时间零点为UTC：2006年1月1日0时0分0秒
9	发送测距帧到达时间	U72	ps	时间零点为UTC：2006年1月1日0时0分0秒
10	测距结果	U32	mm	0~0xFFFFFFFF（单位：mm）
11	钟差结果	S32	ns	
12	测距时间标识	U32	s	
13	测距状态	U8	/	
14	当前空口链路状态	U8	/	

3.2.3. 地址0x0003 读取慢遥数据

序号	参数	参数大小	单位	参数说明
1	空口侧光模块温度	S8	℃	/
2	空口侧光模块电压	U16	mV	/
3	空口侧光模块发射电流	U16	mA	/
4	空口侧光模块发射功率	U16	dBm	/
5	空口侧光模块接收功率	U16	dBm	/
6	路由侧光模块温度	S8	℃	/
7	路由侧光模块电压	U16	mV	/
8	通信成功时间戳	U32	s	时间零点为UTC：2006年1月1日0时0分0秒
9	通信结束时间戳	U32	s	时间零点为UTC：2006年1月1日0时0分0秒
10	数据源选择	U8	/	/
11	通信速率	U8	/	/
12	接收到路由包计数	U64	/	/
13	发送到路由包计数	U64	/	/
14	接收到路由错包计数	U32	/	/
15	发送到路由错包计数	U32	/	/

3.2.4. 地址0x0004 查询重构结果

响应数据数据内容定义如下

序号	参数	参数大小	参数说明
1	重构结果	U8	0x00 重构成功 0x11 重构中 0x22 未收到启动重构 0xFF 重构失败

3.2.5. 地址0x0004 查询固件版本号

响应数据格式定义如下：

序号	参数	参数大小	参数说明
1	固件版本号	32*CHAR	固件版本号，格式: ”主版本号 . 子版本号.修正版本号. 编译版本号 \0“，不可超过32字符，其中: 主版本号: 由年份表示，2025年则为25； 子版本号：表示当年的第几个版本，如果为第一个版本则为0，第二个版本则为1，依次类推； 修正版本号：用于标识当前版本的补丁版本号，0表示为基线版本，1表示为.1补丁版本，依次类推，如不涉及补丁则固定填写0 编译版本号：基线值为0，每当版本的内容发生变化时，如需对外发布，则需要变更编译版本号，以便厂商识别版本差异； eg：25.1.0.7 表示25年第二个基线版本，第8次编译的版本。

3.3. 写数据地址及数据结构定义

3.3.1. 地址0x8001 设置设备自检

序号	参数	参数大小	参数说明
1	无	0	无

3.3.2. 地址0x8002 设置工作模式

序号	参数	参数大小	参数说明
1	工作模式	U8	0为待机模式，1为通信模式，2为测距模式

3.3.3. 地址0x8003 设置IP地址配置

序号	参数	参数大小	下限	上限	参数说明
1	IP类别	U8	0x00	0x7F	配置指定APID（高7位）的本地链路IP，自身APID则为配置自身本地IP 0xAA：配置本机外网IP 0xFF：配置对方网管IP
2	接口ID	U16	/	/	0x0000
3	VLAN ID	U16	0x0000	0xFFFF	0x00000000表示无VLAN
4	IPv6地址	U8[16]	全0	全F	本地按以太网协议（星务管理），外网由星网系统总体统一分配
5	掩码	U8	0x00	0x7F	数值代表地址掩码位数，从最高位开始

3.3.4. 地址0x8004 设置通信参数配置

序号	参数	参数大小	参数说明
1	通信速率	U8	0x00: 10Gbps 0x03: 5Gbps 0x0C: 2.5Gbps 0x0F:1.25Gbps(可选) 0x30:0.625Mbps(可选) 0xFF: 不更新
2	数据源控制	U8	0x00: 自产生PRBS 0x03: 业务数据主份 0x0C: 业务数据备份 0xFF: 不更新

3.3.5. 地址0x8005 设置测距配置

序号	参数	参数类型	参数说明
1	测距开关	U8	0X00:测距开； 0xFF:测距关
2	测距模式	U8	0x00:测距模式1； 0x03:测距模式2;0x0C:测距模式3;0x36:测距模式4;0xFF:不操作

3.3.6. 地址0x8006 设置秒脉冲来源

序号	参数	参数大小	参数说明
1	秒脉冲来源	U16	0x55: 来源导航增强 0xAA：来源平台

3.3.7. 地址0x8007 写文件操作开始

主机开始写文件给从机前会先发送写文件操作请求，给从机，指示指示接下来传输文件的相关信息，从机在收到请求进入写文件操作状态

序号	参数	参数大小	参数说明
1	文件类型	U8	从机根据自身需求自定义
2.	文件长度	U32	接下来要传输文件的大小
3.	文件CRC校验	U16	传输文件的CRC，采用CRC16-CCITT-FALSE校验

3.3.8. 地址0x8008 写文件操作终止

当主机需要终止文件传输时候，发送该请求到从机，从机退出写文件操作状态。

号	参数	参数大小	参数说明
1	无参数	0	无

3.3.9. 地址0x8009 写文件操作结束

主机文件传输完毕后，会发送写文件操作结束给从机，从机在响应中告知主机文件接收和校验情况，发送响应并退出写文件操作状态；

序号	参数	参数大小	参数说明
1	文件接收结果	U8	0x00 - 文件接收正常； 其他 - 文件接收异常，异常值及含义由从机自定义

3.3.10. 地址0x800A 启动重构

通知从机启动重构，主机发送启动重构请求前，会先通过写文件操作，把要重构的软件发送给从机；从机在收到该请求后，执行重构

序号	参数	参数大小	参数说明
1	文件类型	U8	从机根据自身需求自定义

3.3.11. 地址0x800B 复位

主机通知从机复位，**注：从机在收到复位请求后需回复复位响应结束后，才能执行复位；**

序号	参数	参数大小	参数说明
1	无	0	无

3.3.12. 地址0x800C 读文件操作开始

当主机需要读取从机的文件时候，通过对地址0x800C写如下述数据告诉从机，当前主机想要读取从机的某个文件，发送写入读文件操作开始给从机，从机进入读文件操作状态，按照后续的读文件操作进行交互，详见3.1.1.4章节

序号	参数	参数大小	参数说明
1	文件类型	U8	根据从机需求自定义

3.3.13. 地址0x800D 读文件操作终止

当主机需要进行读文件操作时候，发送该请求ADU到从机，从机退出文件操作状态；

序号	参数	参数大小	参数说明
1	无	0	无

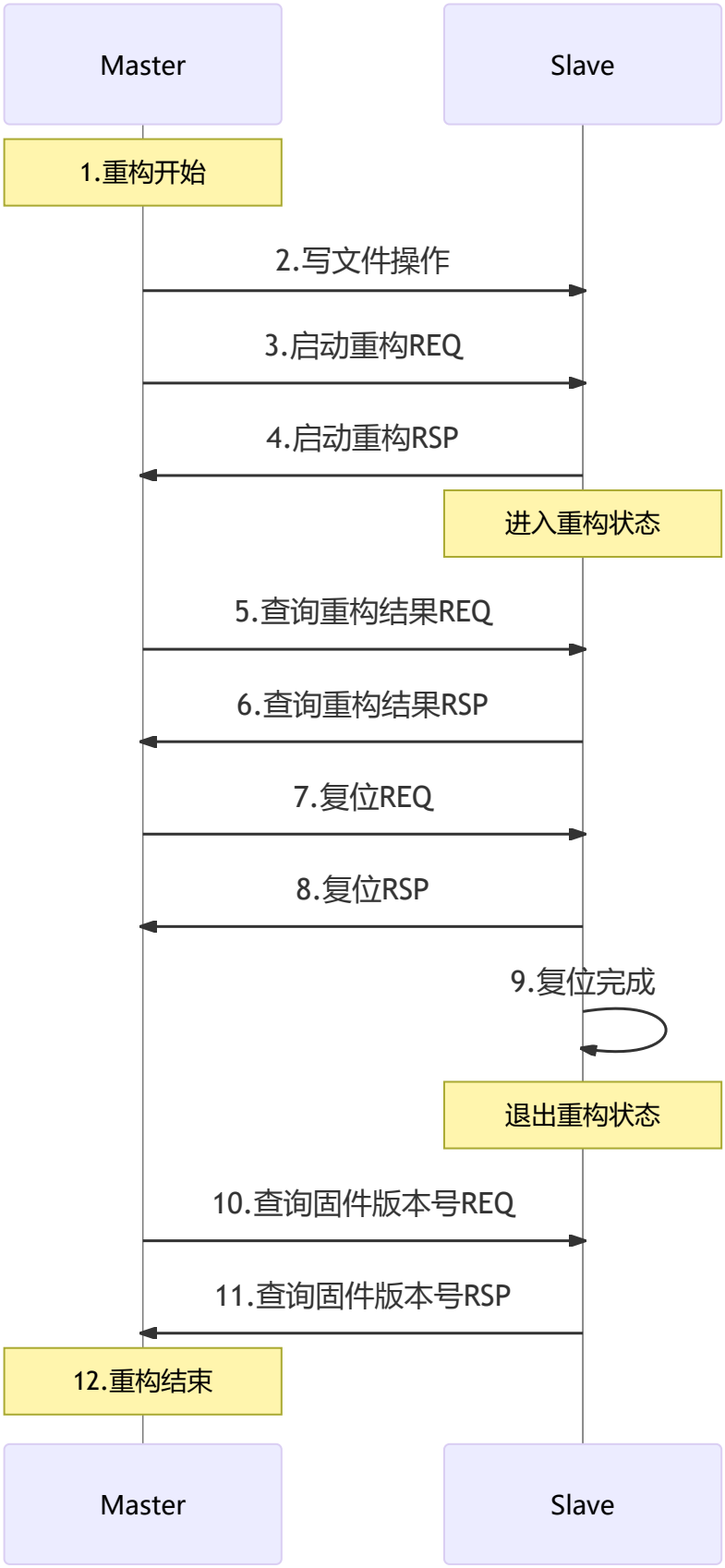
3.3.14. 地址0x800E 读文件操作结束

当主机读取从机的某个文件结束后，发送该请求到从机，从机退出读文件操作状态

序号	参数	参数大小	参数说明
1	无	0	无

3.4. 软件重构交互流程

软件重构主要由主机通过modbus协议交互完成，其交互流程如下所示



流程详解：

步骤1：主机重构开始；

步骤2：主机通过**写文件操作**把要重构的软件发送给从机，详见3.1.1.3章节描述，如果从机正确接收文件，则进入步骤3，否则流程结束；

步骤3：主机发送**启动重构请求**，请求中包含了需要重构的文件，从机找到该文件，进入重构状态(把文件加载到固件)，如果从机无法进入重构状态，则回复异常响应，否则回复正常响应；

步骤4：主机接收**启动重构响应**，如果为正常响应，则进入步骤5，否则重复步骤2，直到超过规定次数，流程结束；

步骤5：主机发送**查询重构结果请求**，查询重构结果，根据自身文件的状态，按照要求回复查询重构结果状态

步骤6：主机接收**查询重构结果响应**：

- 如果响应为超时，则重复步骤5，直到超过规定次数，记录异常，流程结束；否则
- 如果响应为异常，则重复步骤5，直到超过规定次数，记录异常，流程结束，否则

- 如果响应为重构成功，则进入步骤7，否则
- 如果响应为重构中，则延迟一定时间，重复步骤5，如果等待重构的时间超出规定时间（从机需要给出软件重构的最大等待时间），记录异常，流程结束，否则
- 如果响应为未收到启动重构，则重复步骤3，如果重复次数超过规定的次数，记录异常，流程结束；否则
- 如果响应为重构失败，记录异常，流程结束；否则，记录异常，流程结束

步骤7：主机发送**复位请求**，通知从机复位以便生效软件，从机收到请求后，**回复响应**，启动复位（需要等待响应所有bit全部发送完毕后，启动复位）；

步骤8：主机接收**复位响应**：

- 如果响应为异常，则重复步骤8，直到超过最大次数，记录异常流程结束；否则
- 如果响应为超时，则记录场景E，进入步骤10；否则
- 如果响应为正常，则记录场景N，进入步骤10

步骤9：从机复位完成，自动退出重构状态，此时固件版本号变更为最新版本号；

步骤10：主机发送**查询固件版本号请求**，主机根据步骤8的场景，决策发送请求的时机，从机复位起来后如果接收到查询请求后，则回复当前的固件版本号；

步骤11：主机接收**查询固件版本号响应**，如果响应版本号和升级的预期的版本号一直，则进入步骤12，否则根据响应结果执行相关策略；

步骤12： 重构结束。