无锡超强伟业科技有限公司

送丝机通信协议

V1.5

修改记录：

版本号	主要变更内容	拟制/修改日期	备注
V1.0	首次发布，无变更。	2021/10/29	
V1.1	（1）更新了波特率； （2）更新了寄存器地址定义表； （3）更新了脉冲模式相关参数定义； （4）更新了相关示例。	2022/05/16	从软件版本601开始导入
V1.2	更新了“脉冲周期”、“平滑度”、“平均速度”这3个参数的限定范围。	2022/07/05	适用于601及以上版本
V1.3	（1）增加了写单个寄存器的功能码0x06； （2）将CRC改为低字节在前，高字节在后。	2022/07/30	适用于602及以上版本
V1.4	（1）增加了“断丝模式”、“手动送丝”、“手动回抽”这3个参数； （2）“运行状态”由只读改为了可读可写。	2023/05/11	适用于602及以上版本
V1.5	(1)增加有回执的手动送丝、手动回抽指令	2024/11/19	适用于直流送丝机v607以

			上版本，或步进电机 v704 以上版本

目录

1. 硬件接口定义.....1
2. 串口通信格式.....1
2.1 通信模块基本参数.....1
2.2 寄存器地址定义.....2
2.3 数据帧格式.....4
2.4 读寄存器指令—功能码 0x03.....6
2.4.1 功能码 0x03 指令结构介绍.....6
2.4.2 功能码 0x03 指令示例.....7
2.5 写寄存器指令—功能码 0x06 和功能码 0x10.....9
2.5.1 功能码 0x06 指令结构介绍.....9
2.5.2 功能码 0x06 指令示例.....10
2.5.3 功能码 0x10 指令结构介绍.....11
2.5.4 功能码 0x10 指令示例.....11
附录 A：CRC 算法.....13

1. 硬件接口定义

我司送丝机产品对外通信接口新款为端子排，老款为 RS232 DB9 母头插座。
端子排 TX/RX/GND 定义详细见端子排电路丝印。老款 DB9 插座接口定义见表 1.1。

表 1.1 DB9 接口定义（送丝机端）

引脚序号	信号	功能说明	数据方向
1	NC	不使用	
2	TXD	送丝机数据发送	送丝机→主机
3	RXD	送丝机数据接收	主机→送丝机
4	NC	不使用	
5	GND	信号地	
6	NC	不使用	
7	NC	不使用	
8	NC	不使用	
9	+5V	不使用	

2. 串口通信格式

本协议兼容 Modbus RTU 规范。

2.1 通信模块基本参数

通信模块基本参数见表 2.1：

表 2.1 通信模块基本参数

编码	8 位二进制
数据位	8 位
奇偶校验位	无
停止位	1 位
波特率	115200 bit/s

2.2 寄存器地址定义

寄存器地址定义见表 2.2：

表 2.2 送丝机寄存器地址定义表

NO.	功能名称	数据长度(Bytes)	数据类型	数据范围	寄存器地址	R/W 属性
1	自动送丝速度	2	无符号	15~600 cm/min	0x0100	R/W
2	手动送丝速度	2	无符号	15~600 cm/min	0x0101	R/W
3	手动回抽速度	2	无符号	15~600 cm/min	0x0102	R/W
4	启动延时	2	无符号	0~2000 ms	0x0103	R/W
5	回抽长度	2	无符号	0~100 mm	0x0104	R/W
6	补丝长度	2	无符号	0~100 mm	0x0105	R/W
7	补丝延时	2	无符号	0~2000 ms	0x0106	R/W
8	脉冲周期	2	无符号	100~1000 ms	0x0107	R/W
9	平滑度	2	无符号	25 %~80 %	0x0108	R/W
10	平均速度	2	无符号	15~150 cm/min	0x0109	R/W
11	语言选择	2	无符号	0～8	0x010A	R/W
12	预留	2	无符号		0x010B	R/W
13	模式设置	2	无符号	0：连续模式 1：脉冲模式	0x010C	R/W
14	断丝模式	2	无符号	0：常规 1：高速 注：仅步进送丝机有效	0x010D	R/W
15	预留	2	无符号		0x010E	R/W
16	预留	2	无符号		0x010F	R/W
17	预留	2	无符号		0x0110	R/W
18	预留	2	无符号		0x0111	R/W
19	过流告警	2	无符号	1：告警 2：正常	0x0112	R
20	硬件版本	2	无符号		0x0113	R
21	软件版本	2	无符号		0x0114	R
22	运行状态	2	无符号	0:停止 1:运行	0x0115	R/W(0x06/0x10单写)
23	保存当前送丝机数据	2	无符号	XX（任意值）	0x013F	W(0x06/0x10单写)
24	手动送丝	2	无符号	0：停止手动送丝 1：开启手动送丝 注：无回执	0x0140	W(0x06/0x10单写)
25	手动回抽	2	无符号	0：停止手动回抽 1：开启手动回抽	0x0141	W(0x06/0x10单写)

				注：无回执		
26	保留	2	无符号		0x0142	
27	手动送丝 2	2	无符号	0：停止手动送丝 1：开启手动送丝 注：有回执	0x0143	W(0x06/0x10 单写)
28	手动回抽 2	2	无符号	0：停止手动送丝 1：开启手动送丝 注：有回执	0x0144	W(0x06/0x10 单写)
29	保留	2	无符号		0x0145	

注 1：R/W—表示该参数能读能写；R—表示该参数只能读；W—表示该参数只能写。

注 2：地址 0x0115~0x0144 不能连续写多个，只能通过 0x06 或 0x10 写单个。

各寄存器功能定义见表 2.3：

表 2.3 送丝机各寄存器功能定义

NO.	功能名称	功能定义
1	自动送丝速度	表示自动送丝的速度，其范围为：15～600cm/min。
2	手动送丝速度	表示手动送丝的速度，其范围为：15～600cm/min。
3	手动回抽速度	表示手动回抽的速度，其范围为：15～600cm/min。
4	启动延时	表示送丝开关按下到开始送丝中间的延时时间，其范围为：0～2000ms。
5	回抽长度	表示送丝开关松开后回抽焊丝的长度（帮助断丝），其范围为：0～100mm。
6	补丝长度	表示回抽焊丝后补充焊丝的长度，其范围为：0～100mm。
7	补丝延时	表示回抽和补丝之间的等待时间，其范围为：0～2000ms。（用于增强断丝效果）
8	脉冲周期	用于脉冲控制模式，它体现了单个鱼鳞纹的大小，数值越大，鱼鳞纹越大。其范围为：100～1000ms。
9	平滑度	用于脉冲控制模式，它体现了鱼鳞纹路的平滑程度，数值越大纹路越平滑，鱼鳞效果越不明显。其范围为：25%～80%。
10	平均速度	用于脉冲控制模式，表示脉冲模式的平均送丝速度，其范围为：15～150cm/min。
11	语言选择	该寄存器值范围为：0～8，依次对应以下 9 种语言： 中文简体、英语、中文繁体、日语、韩语、俄语、德语、法语、拉丁语。
12	预留	

13	模式设置	用于设置当前送丝模式： 0：连续模式 1：脉冲模式
14	断丝模式	表示回抽补丝速度的两种模式： 0：常规（700cm/min） 1：高速（1200cm/min） 注：仅步进送丝机有效
15~18	预留	
19	过流告警	送丝机产生过流告警后会自动停止送丝，检查确认无异常后重启送丝机，可恢复正常运行。
20	硬件版本	当前使用的送丝机硬件版本号。
21	软件版本	当前使用的送丝机主控板软件版本号。
22	运行状态	送丝机的运行状态，可设置为运行或停止状态。注：该状态与送丝机的送丝使能输入口都处于运行状态时，才会运行。
23	保存当前送丝机数据	该指令将保存送丝机所有参数当前值。建议：非必要不保存，以免影响 flash 寿命。
24	手动送丝	手动送丝开关（手动送丝开关开启并且运行状态设置为“运行”时，开启手动送丝）： 0：停止手动送丝 1：开启手动送丝 注：该参数只能通过写单个寄存器指令设置，并且主控板不会回复该指令。
25	手动回抽	手动回抽开关（手动回抽开关开启并且运行状态设置为“运行”时，开启手动回抽）： 0：停止手动回抽 1：开启手动回抽 注：该参数只能通过写单个寄存器指令设置，并且主控板不会回复该指令。
27	手动送丝 2	手动送丝开关（手动送丝开关开启并且运行状态设置为“运行”时，开启手动送丝）： 0：停止手动送丝 1：开启手动送丝 注：该参数只能通过写单个寄存器指令设置，该指令有回复。
28	手动回抽 2	手动回抽开关（手动回抽开关开启并且运行状态设置为“运行”时，开启手动回抽）： 0：停止手动回抽 1：开启手动回抽 注：该参数只能通过写单个寄存器指令设置，该指令有回复。

2.3 数据帧格式

（1）发送、回执（成功响应）基本数据格式

发送与回执基本数据格式见表 2.4：

表 2.4 发送与回执基本数据格式

从机地址	功能码	数据区	错误校验码	错误校验码

（1 Byte）	（1 Byte）	（n Bytes）			(低字节)	(高字节)
0x0A	0x03/0x06/0x10	0xXX	……	0xXX	0xXX	0xXX

从机地址：送丝机默认地址为 0x0A，暂不支持修改。

功能码：见表 2.5。

0x03 读 1 个或多个寄存器，对应的异常回执功能码为 0x83；

0x06 写单个寄存器，对应的异常回执功能码为 0x86。

0x10 写多个寄存器，对应的异常回执功能码为 0x90。

数据区：n 个字节，双字节数据时，高字节在前，低字节在后。

错误校验码：校验方式采用 CRC16(Modbus)，低字节在前，高字节在后。CRC 计算包含从“从机地址”开始，直到 CRC 校验码前的所有数据，计算方法见附录 A。

CRC 校验过程：发送方计算待发送数据的 CRC 后，将计算出的 CRC 值加入发送帧。接收方接收到数据帧后，重新计算接收到的数据的 CRC 值，并与接收到的 CRC 值比较，如果两个值相同，认为正常，否则异常。

表 2.5 送丝机功能码列表

功能码 (发送)	功能	回执功能码 (正常)	回执功能码 (异常)
0x03	读 1 个或多个寄存器	0x03	0x83
0x06	写单个寄存器	0x06	0x86
0x10	写多个寄存器	0x10	0x90

（2）错误回执基本数据格式：

错误回执基本数据格式见表 2.6：

表 2.6 错误回执基本数据格式

从机地址 （1 Byte）	功能码 （1 Byte）	错误代码 （1 Byte）	CRC16 (低字节)	CRC16 (高字节)
0x0A	0x83/0x86/0x90	0x01/0x02/0x03	0XX	0xXX

异常功能码对应回执关系：

0x03—0x83 当功能值异常时，回执功能码高位置 1，即：0x83。

0x06—0x86 当功能值异常时，回执功能码高位置 1，即：0x86。

0x10—0x90 当功能值异常时，回执功能码高位置 1，即：0x90。

错误代码：

0x01—非法功能；

0x02—非法寄存器地址；

0x03—非法寄存器值。

注：从机地址错误、CRC 校验错误、功能码错误时，送丝机不响应。

2.4 读寄存器指令—功能码 0x03

2.4.1 功能码 0x03 指令结构介绍

（1）功能码 0x03 可同时读取 1 个或多个寄存器的值，其指令结构见表 2.7：

表 2.7 功能码 0x03 指令结构

读寄存器 (主机请求)								
数据方向	从机地址 (1 Byte)	功能码 (1 Byte)	数据区				错误校验码	
主机→从机			寄存器起始地址 (2 Bytes)		寄存器数量 (2 Bytes)		CRC16 (2 Bytes)	
			寄存器起始地址 (高字节)	寄存器起始地址 (低字节)	寄存器数量 (高字节)	寄存器数量 (低字节)	CRC16 (低字节)	CRC16 (高字节)
	0x0A	0x03	0xXX	0xXX	N		0xXX	0xXX

注：当读取多个寄存器时，它的含义是读取以寄存器起始地址开始的连续 N 个寄存器，这里的连续是指寄存器地址连续。

（2）功能码 0x03 成功响应，指令结构见表 2.8：

表 2.8 功能码 0x03 成功响应指令结构

读寄存器 (从机应答)										
数据方向	从机地址 (1 Byte)	功能码 (1 Byte)	数据区						错误校验码	
从机→主机 (成功响应)			返回字节数 (1 Byte)	Data1 (2 Bytes)		……	DataN (2 Bytes)		CRC16 (2 Bytes)	
				Data1 (高字节)	Data1 (低字节)	……	DataN (高字节)	DataN (低字节)	CRC16 (低字节)	CRC16 (高字节)
	0x0A	0x03	2*N	0xXX	0xXX		0xXX	0xXX	0xXX	0xXX

注：

① “返回字节数”表示“Data1”～“DataN”总的字节数。

② Data1 为起始地址对应的寄存器的值，Data2 为起始地址+1 对应的寄存器的值，如此依次递增。

（3）错误响应指令结构见表 2.6。

2.4.2 功能码 0x03 指令示例

（1）读“自动送丝速度”指令示例见表 2.9：

表 2.9 读“自动送丝速度”示例

数据方向	指令内容
主机→从机	0A 03 01 00 00 01 84 8D
从机→主机	0A 03 02 00 3C 1D 94

示例说明见表 2.10：

表 2.10 读“自动送丝速度”示例说明

数据方向	从机地址 (1 Byte)	功能码 (1 Byte)	寄存器地址 (2 Bytes)	寄存器数量 (2 Bytes)	CRC16 (2 Bytes)
主机→从机	0x0A	0x03	0x0100	0x0001	0x848D
数据方向	从机地址 (1 Byte)	功能码 (1 Byte)	返回字节数 (1 Byte)	寄存器值 (2 Bytes)	CRC16 (2 Bytes)
从机→主机	0x0A	0x03	0x02	0x003C	0x1D94

如表 2.10 所示：

主机→从机：读寄存器功能码为 0x03，“自动送丝速度”地址为 0x0100，只读 1 个寄存器的值，因此寄存器数量为 1。

从机→主机：读寄存器成功响应，功能码不变，为 0x03；返回 1 个寄存器的值，返回字节数=2*1=2；寄存器的值 0x003C，对应的 10 进制形式为 60，表示当前自动送丝速度为 60cm/min。

（2）读所有可读寄存器指令示例见表 2.11：

表 2.11 读所有寄存器示例

数据方向	指令内容
主机→从机	0A 03 01 00 00 16 C4 83
从机→主机	0A 03 2C 00 3C 01 2C 01 5E 00 00 00 0A 00 08 01 90 00 C8 00 32 00 32 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 02 00 DC 02 5A 00 00 C1 A0

示例说明见表 2.12：

表 2.12 读所有寄存器示例说明

数据方向	从机地址 (1 Byte)	功能码 (1 Byte)	寄存器地址 (2 Bytes)	寄存器数量 (2 Bytes)	CRC16 (2 Bytes)
主机→从机	0x0A	0x03	0x0100	0x0016	0xC483
数据方向	从机地址 (1 Byte)	功能码 (1 Byte)	返回字节数 (1 Byte)	Data1 (2 Bytes)	Data2 (2 Bytes)
从机→主机	0x0A	0x03	0x2C	0x003C	0x012C
	Data3 (2 Bytes)	Data4 (2 Bytes)	Data5 (2 Bytes)	Data6 (2 Bytes)	Data7 (2 Bytes)
	0x015E	0x0000	0x000A	0x0008	0x0190
	Data8 (2 Bytes)	Data9 (2 Bytes)	Data10 (2 Bytes)	Data11 (2 Bytes)	Data12 (2 Bytes)
	0x00C8	0x0032	0x0032	0x0000	0x0000
	Data13 (2 Bytes)	Data14 (2 Bytes)	Data15 (2 Bytes)	Data16 (2 Bytes)	Data17 (2 Bytes)

0x0000	0x0000	0x0000	0x0000	0x0000
Data18 (2 Bytes)	Data19 (2 Bytes)	Data20 (2 Bytes)	Data21 (2 Bytes)	Data22 (2 Bytes)
0x0000	0x0002	0x00DC	0x025A	0x0000
CRC16 (2 Bytes)				
0xC1A0				

如表 2.12 所示：

主机→从机：送丝机所有可读寄存器一共 22 个，其 16 进制形式为 0x0016；首地址为 0x0100，所有寄存器地址是连续的，因此可以通过表 2.12 中这一条指令读出送丝机所有可读寄存器的值。

从机→主机：返回 22 个寄存器的值，返回字节数=2*22=44，其对应的 16 进制形式为 0x2C；从寄存器地址 0x0100 开始，寄存器的值依次为 Data1～Data22。Data1 为 0x003C，对应的 10 进制形式为 60，表示当前自动送丝速度为 60 cm/min。依次解析出示例中所有寄存器的值如表 2.13 所示：

表 2.13 示例（2）中所有寄存器值列表

自动送丝速度	手动送丝速度	手动回抽速度	启动延时	回抽长度	补丝长度
60 cm/min	300 cm/min	350 cm/min	0 ms	10 mm	8 mm
补丝延时	脉冲周期	平滑度	平均速度	语言选择	预留
400 ms	200 ms	50	50 cm/min	中文简体	0
模式设置	预留	预留	预留	预留	预留
连续模式	0	0	0	0	0
过流告警	硬件版本	软件版本	运行状态		
正常	220	602	停止		

（3）错误响应示例：

如表 2.14 所示，当主机向从机发送读一个不存在的寄存器地址 0x0120 指令时，从机回复错误功能码 0x83，错误代码 0x02（表示非法寄存器地址）。

表 2.14 读寄存器错误响应示例

数据方向	指令内容
主机→从机	0A 03 01 20 00 01 85 47
从机→主机	0A 83 02 B1 33

2.5 写寄存器指令—功能码 0x06 和功能码 0x10

2.5.1 功能码 0x06 指令结构介绍

（1）功能码 0x06 可写单个寄存器，其指令结构见表 2.15：

表 2.15 功能码 0x06 指令结构

写单个寄存器（主机请求）								
数据方向	从机地址 (1 Byte)	功能码 (1 Byte)	数据区				错误校验码	
主机→从机			寄存器地址 (2 Bytes)		寄存器值 (2 Bytes)		CRC16 (2 Bytes)	
			寄存器地址 (高字节)	寄存器地址 (低字节)	寄存器值 (高字节)	寄存器值 (低字节)	CRC16 (低字节)	CRC16 (高字节)
	0x0A	0x06	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX

如表 2.15 所示，通过功能码 0x06 可向单个寄存器地址中写入数据。

（2）功能码 0x06 成功响应，指令结构见表 2.16：

表 2.16 功能码 0x06 成功响应指令结构

写单个寄存器（从机应答）								
数据方向	从机地址 (1 Byte)	功能码 (1 Byte)	数据区				错误校验码	
从机→主机 (成功响应)			寄存器地址 (2 Bytes)		寄存器值 (2 Bytes)		CRC16 (2 Bytes)	
			寄存器地址 (高字节)	寄存器地址 (低字节)	寄存器值 (高字节)	寄存器值 (低字节)	CRC16 (低字节)	CRC16 （高字节）
	0x0A	0x06	0xXX	0xXX	N		0xXX	0xXX

（3）错误响应指令结构见表 2.6。

2.5.2 功能码 0x06 指令示例

（1）写“补丝延时”示例见表 2.17：

表 2.17 写“补丝延时”示例

数据方向	指令内容
主机→从机	0A 06 01 06 01 2C 69 01
从机→主机	0A 06 01 06 01 2C 69 01

示例说明见表 2.18：

表 2.18 写“补丝延时”示例说明

数据方向	从机地址 (1 Byte)	功能码 (1 Byte)	寄存器地址 (2 Bytes)	寄存器值 (2 Bytes)	CRC16 (2 Bytes)
主机→从机	0x0A	0x06	0x0106	0x012C	0x6901
数据方向	从机地址 (1 Byte)	功能码 (1 Byte)	寄存器地址 (2 Bytes)	寄存器值 (2 Bytes)	CRC16 (2 Bytes)
从机→主机	0x0A	0x06	0x0106	0x012C	0x6901

如表 2.18 所示，“补丝延时”寄存器地址为 0x0106，示例中向地址 0x0106 中写入数值 0x012C（对应的 10 进制数为 300），表示将送丝机的“补丝延时”设置为 300ms。

2.5.3 功能码 0x10 指令结构介绍

（1）功能码 0x10 可同时写多个寄存器，其指令结构见表 2.19：

表 2.19 功能码 0x10 指令结构

写寄存器（主机请求）														
数据方向	从机地址 (1 Byte)	功能码 (1 Byte)	数据区										错误校验码	
主机→从机			寄存器起始地址		寄存器数量		数据字节数 (1 Byte)	Data1		.....	DataN		CRC16	
			寄存器起始地址 (高字节)	寄存器起始地址 (低字节)	寄存器数量 (高字节)	寄存器数量 (低字节)		Data1 (高字节)	Data1 (低字节)	.....	DataN (高字节)	DataN (低字节)	CRC16 (低字节)	CRC16 (高字节)
	0x0A	0x10	0xXX	0xXX	N		2*N	0xXX	0xXX		0xXX	0xXX	0xXX	0xXX

如表 2.19 所示，功能码 0x10 可同时向以“寄存器起始地址”开始的连续 N 个寄存器写入数据。“数据字节数”表示“Data1”～“DataN”总的字节数量。

（2）功能码 0x10 成功响应，指令结构见表 2.20：

表 2.20 功能码 0x10 成功响应指令结构

写寄存器 (从机应答)								
数据方向	从机地址 (1 Byte)	功能码 (1 Byte)	数据区				错误校验码	
从机→主机 (成功响应)			寄存器起始地址 (2 Bytes)		寄存器数量 (2 Bytes)		CRC16 (2 Bytes)	
			寄存器起始地址 (高字节)	寄存器起始地址 (低字节)	寄存器数量 (高字节)	寄存器数量 (低字节)	CRC16 (低字节)	CRC1 (高字节)
	0x0A	0x10	0xXX	0xXX	N		0xXX	0xXX

（3）错误响应指令结构见表 2.6。

2.5.4 功能码 0x10 指令示例

（1）写“回抽长度”和“补丝长度”示例见表 2.21：

表 2.21 写“回抽长度”和“补丝长度”示例

数据方向	指令内容
主机→从机	0A 10 01 04 00 02 04 00 0A 00 08 FB 2C
从机→主机	0A 10 01 04 00 02 00 8E

示例说明见表 2.22：

表 2.22 写“回抽长度”和“补丝长度”示例说明

数据方向	从机地址 (1 Byte)	功能码 (1 Byte)	寄存器地址 (2 Bytes)	寄存器数量 (2 Bytes)	字节数量 (1 Bytes)	DATA1 (2 Bytes)	DATA2 (2 Bytes)	CRC16 (2 Bytes)
主机→从机	0x0A	0x10	0x0104	0x0002	0x04	0x000A	0x0008	0xFB2C
数据方向	从机地址 (1 Byte)	功能码 (1 Byte)	寄存器地址 (2 Bytes)	寄存器数量 (2 Bytes)	CRC16 (2 Bytes)			
从机→主机	0x0A	0x10	0x0104	0x0002	0x008E			

如表 2.22 所示，“回抽长度”寄存器地址为 0x0104，示例中向地址 0x0104 中写入数值 0x000A（对应的 10 进制数为 10），表示将送丝机的“回抽长度”设置为 10mm；向地址 0x0105 中写入数值 0x0008，表示将送丝机的“补丝长度”设置为 8mm。

（2）错误响应示例：

如表 2.19 所示，主机向寄存器地址 0x0105 和 0x0106 写入数据，其中，向 0x0106 写入 0x09C4 时（即：将“补丝延时”设置为 2500ms，超出了其设置范围：0~2000 ms），从机回复错误功能码 0x86，错误代码 0x03（表示非法寄存器值）。

表 2.19 写寄存器错误响应示例

数据方向	指令内容
主机→从机	0A 10 01 05 00 02 04 00 08 09 C4 9D 25
从机→主机	0A 90 03 7D C3

附录 A：CRC 算法

（1）查表法：

```
uint16_t modbus_CRC16_Table (uint8_t *nData, uint16_t wLength)
{
    static const uint16_t wCRCTable[] = {
    0X0000, 0XC0C1, 0XC181, 0X0140, 0XC301, 0X03C0, 0X0280, 0XC241,
    0XC601, 0X06C0, 0X0780, 0XC741, 0X0500, 0XC5C1, 0XC481, 0X0440,
    0XCC01, 0X0CC0, 0X0D80, 0XCD41, 0X0F00, 0XCFC1, 0XCE81, 0X0E40,
    0X0A00, 0XCAC1, 0XCB81, 0X0B40, 0XC901, 0X09C0, 0X0880, 0XC841,
    0XD801, 0X18C0, 0X1980, 0XD941, 0X1B00, 0XDBC1, 0XDA81, 0X1A40,
    0X1E00, 0XDEC1, 0XDF81, 0X1F40, 0XDD01, 0X1DC0, 0X1C80, 0XDC41,
    0X1400, 0XD4C1, 0XD581, 0X1540, 0XD701, 0X17C0, 0X1680, 0XD641,
    0XD201, 0X12C0, 0X1380, 0XD341, 0X1100, 0XD1C1, 0XD081, 0X1040,
    0XF001, 0X30C0, 0X3180, 0XF141, 0X3300, 0XF3C1, 0XF281, 0X3240,
    0X3600, 0XF6C1, 0XF781, 0X3740, 0XF501, 0X35C0, 0X3480, 0XF441,
    0X3C00, 0XFCC1, 0XFD81, 0X3D40, 0XFF01, 0X3FC0, 0X3E80, 0XFE41,
    0XFA01, 0X3AC0, 0X3B80, 0XFB41, 0X3900, 0XF9C1, 0XF881, 0X3840,
    0X2800, 0XE8C1, 0XE981, 0X2940, 0XEB01, 0X2BC0, 0X2A80, 0XEA41,
    0XEE01, 0X2EC0, 0X2F80, 0XEF41, 0X2D00, 0XEDC1, 0XEC81, 0X2C40,
    0XE401, 0X24C0, 0X2580, 0XE541, 0X2700, 0XE7C1, 0XE681, 0X2640,
    0X2200, 0XE2C1, 0XE381, 0X2340, 0XE101, 0X21C0, 0X2080, 0XE041,
    0XA001, 0X60C0, 0X6180, 0XA141, 0X6300, 0XA3C1, 0XA281, 0X6240,
    0X6600, 0XA6C1, 0XA781, 0X6740, 0XA501, 0X65C0, 0X6480, 0XA441,
    0X6C00, 0XACC1, 0XAD81, 0X6D40, 0XAF01, 0X6FC0, 0X6E80, 0XAE41,
    0XAA01, 0X6AC0, 0X6B80, 0XAB41, 0X6900, 0XA9C1, 0XA881, 0X6840,
    0X7800, 0XB8C1, 0XB981, 0X7940, 0XBB01, 0X7BC0, 0X7A80, 0XBA41,
    0XBE01, 0X7EC0, 0X7F80, 0XBF41, 0X7D00, 0XBDC1, 0XBC81, 0X7C40,
    0XB401, 0X74C0, 0X7580, 0XB541, 0X7700, 0XB7C1, 0XB681, 0X7640,
    0X7200, 0XB2C1, 0XB381, 0X7340, 0XB101, 0X71C0, 0X7080, 0XB041,
    0X5000, 0X90C1, 0X9181, 0X5140, 0X9301, 0X53C0, 0X5280, 0X9241,
    0X9601, 0X56C0, 0X5780, 0X9741, 0X5500, 0X95C1, 0X9481, 0X5440,
    0X9C01, 0X5CC0, 0X5D80, 0X9D41, 0X5F00, 0X9FC1, 0X9E81, 0X5E40,
    0X5A00, 0X9AC1, 0X9B81, 0X5B40, 0X9901, 0X59C0, 0X5880, 0X9841,
    0X8801, 0X48C0, 0X4980, 0X8941, 0X4B00, 0X8BC1, 0X8A81, 0X4A40,
    0X4E00, 0X8EC1, 0X8F81, 0X4F40, 0X8D01, 0X4DC0, 0X4C80, 0X8C41,
    0X4400, 0X84C1, 0X8581, 0X4540, 0X8701, 0X47C0, 0X4680, 0X8641,
    0X8201, 0X42C0, 0X4380, 0X8341, 0X4100, 0X81C1, 0X8081, 0X4040 };

    uint8_t nTemp;
    uint16_t wCRCWord = 0xFFFF;
    uint16_t wCRCResult = 0;
```

```
    while (wLength--)
    {
        nTemp = *nData++ ^ wCRCWord;
        wCRCWord >>= 8;
        wCRCWord ^= wCRCTable[nTemp];
    }
    //将 CRC 高低字节交换
    wCRCResult = wCRCWord>>8;
    wCRCResult = wCRCResult | (wCRCWord << 8);
    return wCRCResult;
}
```

（2）直接计算：

```
uint16_t modbus_CRC16_Calculate(uint8_t *ndata, uint16_t wLength)
{
    uint16_t wCRCWord = 0xFFFF;
    uint16_t wCRCResult = 0;
    uint16_t polynomial = 0xA001;
    uint8_t i,j;

    for(i = 0;i < wLength;i++)
    {
        wCRCWord ^= ndata[i];
        for(j = 0;j < 8;j++)
        {
            if((wCRCWord & 0x0001) != 0)
            {
                wCRCWord >>= 1;
                wCRCWord ^= polynomial;
            }
            else
            {
                wCRCWord >>= 1;
            }
        }
    }
    //将 CRC 高低字节交换
    wCRCResult = wCRCWord>>8;
    wCRCResult = wCRCResult | (wCRCWord << 8);
    return wCRCResult;
}
```