产品目录

版本信息

版本	时间	修改
1.0	2023 年 12 月 26 日	初始版本，适合于产品 CC101 温湿度无线透传； CC201 RS485 无线透传
1.1	2024 年 02 月 08 日	添加 CRC-16 的计算方法和示例
1.2	2024 年 03 月 01 日	修改系统配置寄存器和输入寄存器地址，并固定 Modbus 指令备地址为 0xFF； 更新应用举例中的相关系统配置寄存器指令
1.3	2024 年 07 月 07 日	修改主动指令设置寄存器；添加参数 JSON 模板设置寄存器；
1.4	2024 年 12 月 30 日	串口 1 寄存器 1003，支持服务器 JSON 转 Modbus RTU; 串口 2 寄存器 2003，支持服务器 JSON 转 Modbus RTU
1.5	2025 年 01 月 10 日	1. 传输数据格式寄存器：增加服务器 JSON 指令转 RTU 2. 网络心跳包配置寄存器：增加服务器 JSON 指令转 RTU 3. 增加寄存器配置说明: 服务器下发 JSON 指令转 Modbus RTU
1.6	2025 年 02 月 05 日	增加寄存器：外接 Modbus 设备的 JSON 模板设置
1.7	2025 年 03 月 22 日	增加 4.1，硬件连接

参考文档

CC201, CC101-B	物联网协议网关(NBIOT)	下载
CC101-C	温湿度无线传感器(NBIOT)	下载

1. 文档简介

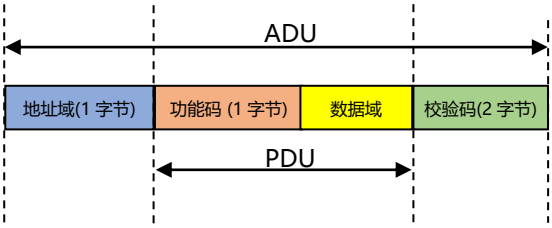
本文档描述的 Modbus 指令及 Modbus 寄存器适用于以下产品：

- CC201，RS485 无线透传模块
  - 串口 1 为 RS485 串口，电压为 3.3 伏特
  - 串口 2 为 UART TTL 串口，电压为 3.3 伏特
- CC101，温湿度无线透传模块
  - 串口 1 为 UART TTL 串口，电压为 3.3 伏特

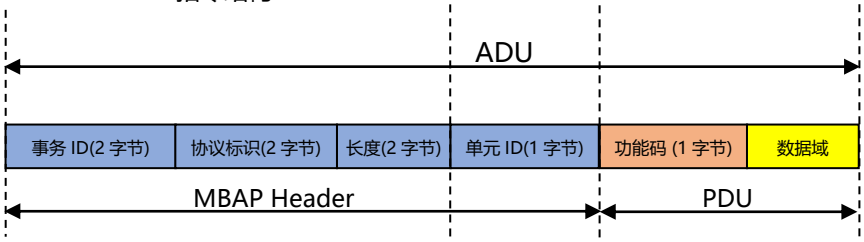
2. Modbus 指令及描述

2.1 Modbus 指令结构描述

- Modbus RTU 指令结构



- Modbus TCP 指令结构



- 校验码：为 2 字节 CRC-16 结果，其计算方法为：

计算 CRC-16 校验码的多项式为：X16+X15+X2+1（0x8005）,初始值 0xFFFF，低位在前，高位在后，结果与 0x0000 异或。示例代码参照附录 1。

2.2 Modbus 指令功能码描述

功能码 01，读线圈状态，单 bit 访问，其指令格式为：

读保持寄存器	从机地址	功能码	数据域				CRC16 校验	
主机发送	1 字节	0x01	地址 H	地址 L	数量 H	数量 L	高位	低位
从机返回	1 字节	0x01	返回字节数	字节 1	字节 2	...	高位	低位

功能码 02，读离散输入状态，单 bit 访问，其指令格式为：

读保持寄存器	从机地址	功能码	数据域				CRC16 校验	
主机发送	1 字节	0x02	地址 H	地址 L	数量 H	数量 L	高位	低位
从机返回	1 字节	0x02	返回字节数	字节 1	字节 2	...	高位	低位

功能码 05，写单个线圈，单 bit 访问，其指令格式为：

读保持寄存器	从机地址	功能码	数据域				CRC16 校验	
主机发送	1 字节	0x05	地址 H	地址 L	标志位 H	标志位 L	高位	低位
从机返回	1 字节	0x05	地址 H	地址 L	标志位 H	标志位 L	高位	低位

功能码 0F，写多个线圈，单 bit 访问，其指令格式为：

读保持寄存器	从机地址	功能码	数据域							CRC16 校验	
主机发送	1 字节	0x03	地址 H	地址 L	数量 H	数量 L	字节数	数据 1H	数据 1L	高位	低位
从机返回	1 字节	0x03	地址 H	地址 L	数量 H	数量 L				高位	低位

功能码 03，读保持寄存器，16 bit 访问，其指令格式为：

读保持寄存器	从机地址	功能码	数据域				CRC16 校验	
主机发送	1 字节	0x03	地址 H	地址 L	数量 H	数量 L	高位	低位
从机返回	1 字节	0x03	返回字节数	字 1H	字 1L	...	高位	低位

功能码 04，读输入寄存器，16 bit 访问，其指令格式为：

读输入寄存器	从机地址	功能码	数据域				CRC16 校验	
主机发送	1 字节	0x04	地址 H	地址 L	数量 H	数量 L	高位	低位
从机返回	1 字节	0x04	返回字节数	字 1H	字 1L	...	高位	低位

功能码 06，写单个保持寄存器，16 bit 访问，其指令格式为：

写单个保存寄存器	从机地址	功能码	数据域				CRC16 校验	
主机发送	1 字节	0x06	地址 H	地址 L	数据 H	数据 L	高位	低位
从机返回	1 字节	0x06	地址 H	地址 L	数据 H	数据 L	高位	低位

功能码 10，写多个保持寄存器，16 bit 访问，其指令格式为：

写多个保存寄存器	从机地址	功能码	数据域								CRC16 校验	
主机发送	1 字节	0x10	地址 H	地址 L	数量 H	数量 L	字节数	数据 1H	数据 1L	...	高位	低位
从机返回	1 字节	0x10	地址 H	地址 L	数量 H	数量 L					高位	低位

3. Modbus 寄存器

3.1 保持寄存器，系统配置

在任何模式都可执行；支持 Modbus 功能码 03、06、10；设备地址固定为 FF

寄存器(16 进制)	数据格式(16 进制)	Modbus 指令举例(16 进制)
重启设备 地址：0000 字节数：2	**字节 1**：00 **字节 2**：00	**写，重启设备：** 发送，FF 06 0000 0000 9C14 返回，设备重启
恢复出厂设置 地址：0001 字节数：2	**字节 1**：00 **字节 2**：00	**写，回复出厂设置，并重启设备：** 发送，FF 06 0001 0000 CDD4 返回，回复出厂设置，并重启
串口 1 工作模式设置 地址：0002 字节数：2	**字节 1：串口 1 工作模式** 00：配置模式(缺省) 01：透传模式 02：串口主动指令模式 **字节 2**：00，预留	**读，读取串口 1 的工作模式：** 发送，FF 03 0002 0001 3014 返回，FF 03 02 0000 9190 **写，设置串口 1 为配置模式：** 发送，FF 06 0002 0000 3DD4 返回，FF 06 0002 0000 3DD4
串口 2 工作模式设置 地址：0003 字节数：2	**字节 1：串口 2 工作模式** 00：配置模式(缺省) 01：透传模式 02：串口主动指令模式 **字节 2**：00，预留	**读，读取串口 2 的工作模式：** 发送，FF 03 0003 0001 61D4 返回，FF 03 02 0000 9190 **写，设置串口 2 为配置模式：** 发送，FF 06 0003 0000 6C14 返回，FF 06 0003 0000 6C14
串口 1 Modbus 从机地址 地址：0004 字节数：2	**字节 1：串口 1 Modbus 从机地址，** 0xC9(缺省) **字节 2**：00，预留	**读，读取串口 1 Modbus 从机地址：** 发送，FF 03 0004 0001 D015 返回，FF 03 02 C900 C7C0 **写，串口 1 Modbus 从机地址为 0x0E：** 发送，FF 06 0004 0E00 D9B5 返回，FF 06 0004 0E00 D9B5
串口 2 Modbus 从机地址 地址：0005 字节数：2	**字节 1：串口 2 Modbus 从机地址** 0xC9(缺省) **字节 2**：00，预留	**读，读取串口 2 Modbus 从机地址：** 发送，FF 03 0005 0001 81D5 返回，FF 03 02 C900 C7C0 **写，串口 2 Modbus 从机地址为 0x0E：** 发送，FF 06 0005 0E00 8875 返回，FF 06 0005 0E00 8875
设备 ID 地址：0006~0007 字节数：4	设备 ID 为 8 位 16 进制数， 举例：缺省设备号 93312190 对应的字节为： 0x93,0x31,0x21,0x90	**读，设备 ID：** 发送，FF 03 0006 0002 31D4 返回，FF 03 04 93 31 21 90 814B **写，设置设备 ID 为 0x86123451：** 发送，FF 10 0006 0002 04 86123451 1A1F 返回，FF 10 0006 0002 B417

3.2 输入寄存器，模块数据读取

在任何模式都可执行；支持 Modbus 功能码 04；设备地址固定为 FF

寄存器(16 进制)	数据格式(16 进制)	Modbus 指令举例(16 进制)
读取固件版本号 地址：0000 字节数：2	**字节 1**：模块类型 **字节 2**：固件版本	**读，固件版本：** 发送，FF 04 0000 0001 2414 返回，FF 04 02 C210 C048
读取模组 IMEI 地址：0001~0010 字节数：16	**字节 1 ~ 字节 16**：模组 IMEI 字符的 16 进制	**读，模组 IMEI：867196064805912** 发送，FF 04 0001 0008 B5D2 返回，FF 04 10 383637313936303634

		38303539313200 24FB
读取模组序列号 SN 地址：0011~001A 字节数：20	**字节 1 ~ 字节 20**：模组序列号字符的 16 进制	**读，模组序列号 SN：2316DB00235292076785** 发送，FF 04 0011 000A 35D6 返回，FF 04 14 323331364442303032 33353239323037363738335 1571
读取物联网卡 IMSI 地址：001B~002A 字节数：16	**字节 1 ~ 字节 16**：物联网卡 IMSI 字符的 16 进制	**读，物联网卡 IMSI：460082680100045** 发送，FF 04 001B 0008 9415 返回，FF 04 10 343630303832363830 31303030343500 246A
读取物联网卡 ICCID 地址：002B~0034 字节数：20	**字节 1 ~ 字节 20**：物联网卡 ICCID 字符的 16 进制	**读，物联网卡 ICCID：898604A6102181542395** 发送，FF 04 002B 000A 15DB 返回，FF 04 14 383938363034413631 3032 313831353432333935 14B1
读取网络信号强度 地址：0035 字节数：2	**字节 1：**16 进制，信号质量，10 进制值范围 0~31，越大越好 **字节 2：**16 进制，误码率，0 进制值范围 0 ~ 99	**读，网络信号强度：** 发送，FF 04 0035 0001 341A 返回，FF 04 02 1605 5E87

3.3 保持寄存器，串口 1、串口 2 配置

支持 Modbus 功能码 03、06、10

寄存器(16 进制)	**数据格式(16 进制)**	**Modbus 指令举例(16 进制)**
串口协议配置 字节数：4 寄存器地址： 串口 1: 1000~1001 串口 2: 2000~2001	**字节 1：波特率** 0x00=2400; 0x01=4800; 0x02=9600(缺省); 0x03=19200; 0x04=38400; 0x05=57600; 0x06=115200; **字节 2：字节位数** 0x00=8 位(缺省); 0x01=7 位; 0x02=6 位 **字节 3：校验位** 0x00=无(缺省); 0x01=奇校验; 0x02=偶校验 **字节 4：停止位** 0x00=1 个停止位(缺省); 0x01=1.5 个停止位; 0x02=2 个停止位	**读，串口 2 的协议配置：** 发送，C9 03 2000 0002 DF83 返回，C9 03 04 02000000 B247 **写，设置串口 2 的波特率为 115200：** 发送，C9 10 2000 0002 04 06000000 BC85 返回，C9 10 2000 0002 5A40
工作模式配置 字节数：2 寄存器地址： 串口 1: 1002 串口 2: 2002	**字节 1：**工作模式 00：配置模式(缺省) 01：透传模式 02：主动轮询模式 **字节 2：**00(预留)	**读，串口 2 的工作模式：** 发送，C9 03 2002 0001 3E42 返回，C9 03 02 0001 9854 **写，设置串口 2 为配置模式为透传：** 发送，C9 06 2002 0100 3212 返回，C9 06 2002 0100 3212
透传模式数据格式 字节数：2 寄存器地址： 串口 1: 1003 串口 2: 2003	**字节 1：设备上传的数据格式** **上传数据格式**：X = bit3 ~ bit0: X = 0：透传（缺省）; X = 1：Modbus RTU 数据转 Modbus TCP; **数据头设置**：bit 7 ~ bit 5，默认值 000: bit 7 = 1：数据头加 4 字节 UTC 时间戳信息; bit 6 = 1：数据头加 4 字节设备号; bit 5 = 1：数据头加 2 字节固件号; bit 4 = 预留 **字节 2：服务器下发的数据格式** 00：不做处理(缺省) 01：透传（缺省）; 02：Modbus TCP 数据转 Modbus RTU 03：JSON 指令转 Modbus RTU	**读，串口 2 的传输数据格式：** 发送，C9 03 2003 0001 6F82 返回，C9 03 02 0100 5804 **写，设置串口 2 的传输数据格式为 Modbus RTU/TCP 互转：** 发送，C9 06 2003 0101 A212 返回，C9 06 2003 0101 A212
服务器 IP 地址 字节数：4 寄存器地址：	默认值：7F 00 00 01，为 16 进制 对应 10 进制 IP 地址: 127.0.0.1	**读，服务器 IP 地址：** 发送，C9 03 2010 0002 DE46 返回，C9 03 04 782E2EF3 96B3

串口 1: 1010~1011 串口 2: 2010~2011		**写，设置串服务器 IP 地址，18.19.20.21：** 发送，C9 10 2010 0002 04 12131415 8773 返回，C9 10 2010 0002 5B85
服务器端口号 字节数：2 寄存器地址： 串口 1: 1012 串口 2: 2012	0x0001~0xFFFF， 默认值: 0x04D2, 对应端口号: 1234	**读，服务器端口号：** 发送，C9 03 2012 0001 3F87 返回，C9 03 02 075B 1A5F **写，设置服务器端口号为 16 进制 0x4D2：** 发送，C9 06 2012 04D2 B0DA 返回，C9 06 2012 04D2 B0DA
网络传输协议 字节数：2 寄存器地址： 串口 1: 1013 串口 2: 2013	**字节 1：** 0x00=TCP 客户端(缺省)； 0x01=UDP 客户端； 0x02=MQTT 客户端； **字节 2：** 0x00	**读，网络传输协议：** 发送，C9 03 2013 0001 6E47 返回，C9 03 02 0000 5994 **写，设置网络传输协议为 UDP：** 发送，C9 06 2013 0100 6217 返回，C9 06 2013 0100 6217
注册包设置 字节数：2 寄存器地址： 串口 1: 1014 串口 2: 2014	**字节 1, 注册包类型:** 00，自定义内容； 01，设备 ID **字节 2, 00（预留）**；	**读，注册包设置：** 发送，C9 03 2014 0001 DF86 返回，C9 03 02 0000 5994 **写，设置注册包为自定义内容：** 发送，C9 06 2014 0100 D3D6 返回，C9 06 2014 0100 D3D6
注册包自定义内容 字节数：30 寄存器地址： 串口 1: 1015~1023 串口 2: 2015~2023	**字节 1：** 注册包长度，缺省 0； **字节 2~30：** 16 进制字符串，注册包内容，长度在字节 1 定义 注册包缺省内容： 无	**读，注册包内容：** 发送，C9 03 2015 000F 0F82 返回，C9 03 1E 0543433230320000000000000000000000000000000000000000000000000000000052CE **写，设置注册包内容：** 发送，C9 10 2015 0004 08 0643433230310100 5639 返回，C9 10 2015 0004 CB86
网络心跳包时间 字节数：2 寄存器地址： 串口 1: 1024 串口 2: 2024	0x3C ~ 0xFFFF 秒，对应 60 ~ 65535 秒 缺省 00 秒：表示没有心跳包	**读，网络心跳包设置：** 发送，C9 03 2024 0001 DF89 返回，C9 03 02 0000 5994 **写，设置网络心跳包时间：** 发送，C9 06 2024 0000 D249 返回，C9 06 2024 0000 D249
网络心跳包配置 字节数：2 寄存器地址： 串口 1: 1025 串口 2: 2025	**字节 1**：0x00(预留) **字节 2：**服务器下发数据处理 0x00, 不做处理；(缺省) 0x01, 透明发送到串口； 0x02, Modbus TCP 转 RTU，发送到串口 0x03, Modbus JSON 转 RTU，发送到串口	**读，网络心跳包响应处理：** 发送，C9 03 2025 0001 8E49 返回，C9 03 02 0000 5994 **写，设置网络心跳包响应处理：** 发送，C9 06 2025 0003 C388 返回，C9 06 2025 0003 C388
网络心跳包内容 字节数：30 寄存器地址： 串口 1: 1026~1034 串口 2: 2026~2034	**字节 1：** 心跳包长度，缺省 00； **字节 2~30：** 16 进制字符串 心跳包内容，长度在**字节** 1 定义	**读，网络心跳包设置：** 发送，C9 03 2026 0005 7F8A 返回，C9 03 0A 00000000000000000000 3AFF **写，设置网络心跳包内容为 434332303102：** 发送，C9 10 2026 0004 08 0643433230310200 128D 返回，C9 10 2026 0004 3B89
Modbus 从机地址 字节数：2 寄存器地址： 串口 1: 1060 串口 2: 2060	**字节 1**： Modbus 从机地址，缺省：0xC9 **字节 2**： 0x00(预留)	**读，串口 2 的 Modbus 从机地址设置：** 发送，C9 03 2060 0001 9F9C 返回，C9 03 02 C900 0FC4 **写，设置串口 2 的 Modbus 从机地址为 0x0E：** 发送，C9 06 2060 0E00 963C 返回，C9 06 2060 0E00 963C
主动指令设置寄存器 寄存器字节数：28 寄存器地址： **主动指令 1 内容设置** 串口 1: 1080~108D 串口 2: 2080~208D	**描述：** **字节 1-2**：指令周期，单位秒，缺省 0 秒； 该值为 0 秒，表示该主动指令没有启用 **字节 3-4**：指令周期初始计时偏移，单位秒，缺省 0 **字节 5：** 设置上报数据格式和下发数据格式，: Bit/1/0：上报数据格式 =0: 为透传，**字节 6** 无意义	读，串口 2 的主动指令 1 的内容： 发送，C9 03 2080 000E DE6E 返 回 ， C9031C001E0000100009FF018700000000007800000000000000000000000000 53C7 写，设置串口 2 的主动指令 1 的内容： 发送，C9102080000E1C001E0000100009FF018700000000007800000000000000000000000000F0FE

<table>
<tr>
<td>
主动指令 2 内容设置
串口 1: 108E~109B
串口 2: 208E~209B

主动指令 3 内容设置
串口 1: 109C~10A9
串口 2: 209C~20A9

主动指令 4 内容设置
串口 1: 10AA~10B7
串口 2: 20AA~20B7

主动指令 5 内容设置
串口 1: 10B8~10C5
串口 2: 20B8~20C5

主动指令 6 内容设置
串口 1: 10C6~10D5
串口 2: 20C6~20D5</td>
<td>=1：为 JSON 格式 1，参数在字节 6 定义
=2：为 JSON 格式 2，参数在字节 6 定义
Bit3/2：下发数据处理
=0x00, 不做处理；(缺省)
=0x01, 透明发送到串口；
=0x02, Modbus TCP 转 RTU，发送到串口
=0x03, Modbus JSON 转 RTU，发送到串口
Bit4=1: 上报数据包含 UTC 时间戳；
-JSON 格式时: 键值为“time”
-透传时: 为 4 字节 16 进制数
Bit5=1: 上报数据包含 4 字节设备 ID；
-JSON 格式时: 键值为“deviceId”
-透传: 为 6 字节 10 进制数
Bit6=1: 上报数据包含 2 字节固件版本；
-JSON 格式时: 键值为“fmVer”
-透传时: 为 2 字节 16 进制数
Bit7= 预留，为 0
字节 6：RFU
字节 7：指令字节数长度，最大 20
字节 8-27：指令内容，长度为字节 7 的值
字节 28：预留</td>
<td>返回：C9 10 2080 000E 5BAD

说明：
JSON 格式 1：
{"time": 1638865162,"devid": 9331290,"fm": "C1.2","a": 23.5,"b": 46.8}

JSON 格式 2：
{"time": 1638865162,"devid": 9331290,"fm": "C1.2","data": {"a": 23.5,"b": 46.8}}</td>
</tr>
<tr>
<td>参数 JSON 模板设置
寄存器字节数：24

寄存器地址：
参数 JSON 模板 1
串口 1: 1300~130B
串口 2: 2300~230B

参数 JSON 模板 2
串口 1: 130C~1317
串口 2: 230C~2317

参数 JSON 模板 3
串口 1: 1318~1323
串口 2: 2318~2323

参数 JSON 模板 4
串口 1: 1324~132F
串口 2: 2324~232F

参数 JSON 模板 5
串口 1: 1330~133B
串口 2: 2330~233B

参数 JSON 模板 6
串口 1: 133C~1347
串口 2: 233C~2347

参数 JSON 模板 7
串口 1: 1348~1353
串口 2: 2348~2353

参数 JSON 模板 8
串口 1: 1354~135F
串口 2: 2354~235F</td>
<td>描述：
<table>
<tr><th>字节</th><th>意义</th></tr>
<tr><td>1</td><td>键值长度，最大长度 10</td></tr>
<tr><td>2-11</td><td>键值，ASCII 码，最多 10 个字节</td></tr>
<tr><td>12-13</td><td>寄存器地址</td></tr>
<tr><td>14</td><td>数据类型：
Bit7/6: 当数据为 float16, float32 时，数据的小数点位数，从 0~3；
Bit5/4: 预留, 0x0000；
Bit3/2/1/0：数据类型：
= 0: int16， 2 字节；
= 1: uint16， 2 字节；
= 2: int32， 4 字节；
= 3: uint32， 4 字节；
= 4: bool， 2 字节；
= 5: float16，2 字节；
= 6: float32，4 字节；</td></tr>
<tr><td>15</td><td>字节顺序, A 为低位，D 为高位：
= 0x00: ABCD；
= 0x01：BADC
= 0x02: CDAB;
= 0x03: DCBA</td></tr>
<tr><td>16-17</td><td>结果比例，缺省 1；符合浮点数格式 1</td></tr>
<tr><td>18-19</td><td>结果偏移，缺省 0；符合浮点数格式 1</td></tr>
<tr><td>20-21</td><td>上报阈值下限，小于等于该值，上报，否则不上报；缺省为 0。符合浮点数格式 1</td></tr>
<tr><td>22-23</td><td>上报阈值上限，大于该值，上报，否则不上报；缺省为 0。符合浮点数格式 1</td></tr>
<tr><td>24</td><td>预留，0x00</td></tr>
</table></td>
<td>读，参数 JSON 模板 1 的内容：
发送，C9 03 2300 000C 5E03
返回，C9 03 0A 02023132033334350000 4947

写，参数 JSON 模板 1 的内容：
发送，C9 10 2300 000C 18 0161000000000000000000 1060 01 00 0100 0000 0000 0000 00 DA12
返回，C9 10 23 00 00 0C DBC0

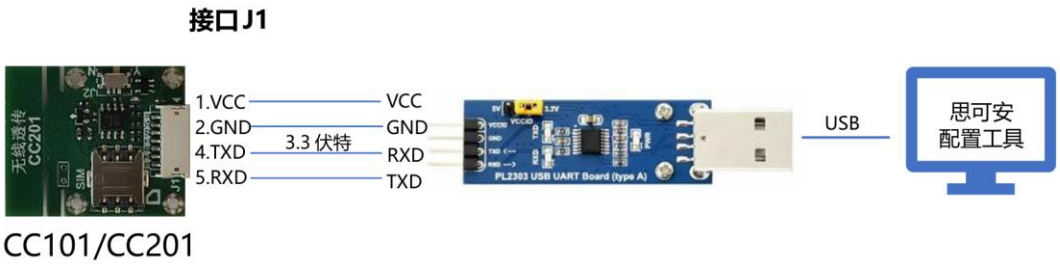
说明：
浮点数格式 1：用 2 字节表示一个浮点数
Bit 15 = 1：该数值为负数；=0：为正数；
Bit 14-13：表示该数值的小数点位数，从 0-3；
Bit 12-00：表示该数值无小数点的十进制大小，从 0~8191；
举例：0x2001=0.1(十进制)；0xC109 = -2.65(十进制)</td>
</tr>
<tr><td colspan="3"></td></tr>
<tr>
<td>Modbus 外接设备的 JSON 模板设置
寄存器字节数：2</td>
<td>字节 1：外接 Modbus 设备的地址（1~247）
字节 2：参数 JSON 格式配置，如下定义：
Bit0=1: 包含参数 JSON 模板 1
Bit1=1: 包含参数 JSON 模板 2</td>
<td>读，设备 1 的 JSON 模板设置：
发送，C9 03 2600 0001 9F0A
返回，C9 03 02 0103 1805</td>
</tr>
</table>

寄存器地址: **设备 1 的 JSON 模板** 串口 1: 1600~1601 串口 2: 2600~2601 **设备 2 的 JSON 模板** 串口 1: 1602~1603 串口 2: 2602~2603 **设备 3 的 JSON 模板** 串口 1: 1604~1605 串口 2: 2604~2605 **设备 4 的 JSON 模板** 串口 1: 1606~1607 串口 2: 2606~2607 **设备 5 的 JSON 模板** 串口 1: 1608~1609 串口 2: 2608~2609 **设备 6 的 JSON 模板** 串口 1: 160A~160B 串口 2: 260A~260B **设备 7 的 JSON 模板** 串口 1: 160C~160D 串口 2: 260C~260D **设备 8 的 JSON 模板** 串口 1: 160E~160F 串口 2: 260E~260F	Bit2=1: 包含参数 JSON 模板 3 Bit3=1: 包含参数 JSON 模板 4 Bit4=1: 包含参数 JSON 模板 5 Bit5=1: 包含参数 JSON 模板 6 Bit6=1: 包含参数 JSON 模板 7 Bit7=1: 包含参数 JSON 模板 8	**写，设备 1 的 JSON 模板设置:** 发送，C9 06 2600 0107 D298 返回，C9 06 2600 0107 D298
MQTT: 保活时间 字节数：2 寄存器地址: 串口 1: 1120 串口 2: 2120	0x1E~0x04B0 秒，对应 30 ~ 65535 秒 缺省 0x3C	**读，串口 2，MQTT 保活时间(秒):** 发送，C9 03 2120 0001 9FB4 返回，C9 03 02 00B4 59E3 **写，设置串口 2 的 MQTT 保活时间(秒):** 发送，C9 06 2120 003C 9265 返回，C9 06 2120 003C 9265
MQTT: 清理会话 字节数：2 寄存器地址: 串口 1: 1121 串口 2: 2121	**字节 1**： 0x00：不清理; 0x01：清理(缺省) **字节 2**：0x00	**读，串口 2，MQTT 清理会话:** 发送，C9 03 2121 0001 CE74 返回，C9 03 02 0001 9854 **写，设置串口 2，MQTT 清理会话:** 发送，C9 06 2121 0000 C3B4 返回，C9 06 2121 0000 C3B4
MQTT: 客户端 ID 字节数：60 寄存器地址: 串口 1: 1122~113F 串口 2: 2122~213F	客户端 ID，16 进制，以 0x00 结尾	**读，串口 2 的 MQTT 客户端 ID:** 发送，C9 03 2122 001E 7FBC 返回，C9 03 3C 636964310000000000000000000000 000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000008543 **写，设置串口 2 的 MQTT 客户端 ID 为 "cc2012":** 发送，C9 10 2122 0004 08 4343323031320000 A671 返回，C9 10 2122 0004 7BB4
MQTT: 用户名 字节数：60 寄存器地址: 串口 1: 1140~115D 串口 2: 2140~215D	用户名，16 进制，以 0x00 结尾	**读，串口 2 的 MQTT 用户名:** 发送，C9 03 2140 001E DE62 返回，C9 03 3C 7573657231323300000000000000000 000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000 90CA **写，设置串口 2 的 MQTT 用户名为 "user123":** 发送，C9 10 2140 0004 08 7573657231323300 0D13 返回，C9 10 2140 0004 DA6A
MQTT: 用户密码 字节数：60 寄存器地址: 串口 1: 115E~117B 串口 2: 215E~217B	用户密码，16 进制，以 0x00 结尾	**读，串口 2 的 MQTT 用户密码:** 发送，C9 03 215E 001E BE64 返回，C9 03 3C 313233343536000000000000000000 000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000585A **写，设置串口 2 的 MQTT 用户密码为 "123456":** 发送, C9 10 215E 0004 08 3132333435360000 A004 返回, C9 10 215E 0004 BA6C

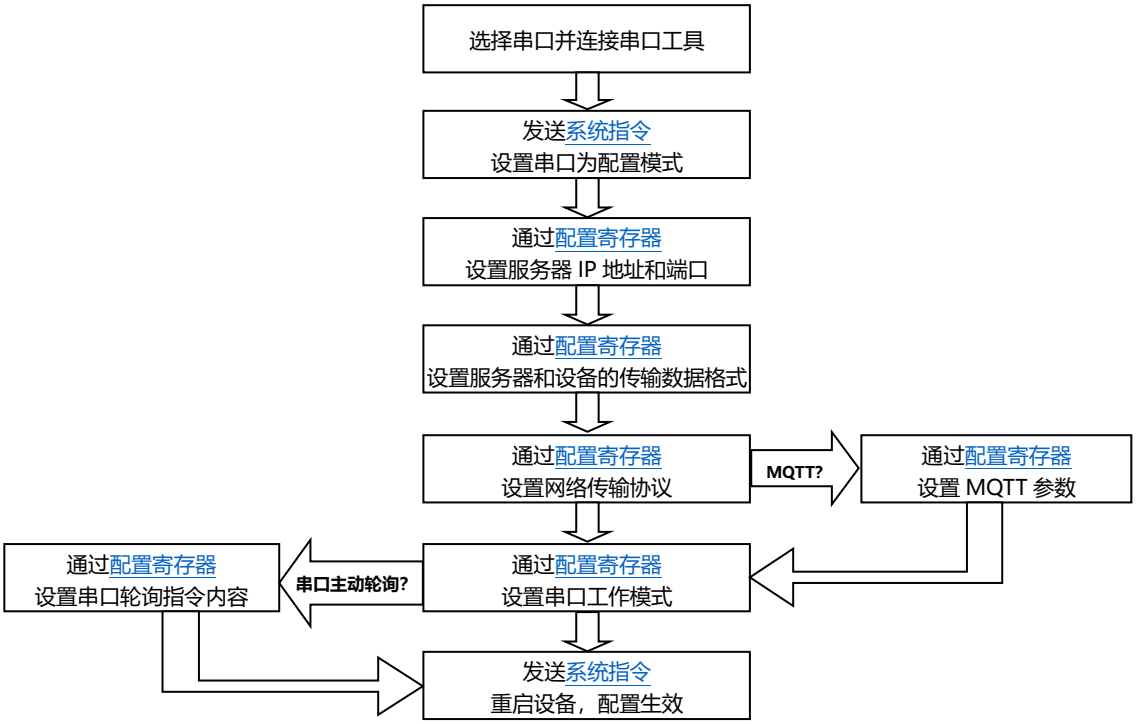
MQTT: 发布主题 字节数：100 寄存器地址: 串口 1: 117C~11DF 串口 2: 217C~21DF	发布主题，16 进制，以 0x00 结尾	**读，串口 2 的 MQTT 发布主题:** 发送，C9 03 217C 0032 1FB3 返回，C9 03 64 737562320000000000000000000000 00000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000 000000000000000000000000 2F53 **写，设置串口 2 的 MQT 发布主题为“pub2”:** 发送, C9 10 217C 0003 06 707562320000 AD6D 返回, C9 10 217C 0003 5BA4
MQTT: 订阅主题 字节数：100 寄存器地址: 串口 1: 11E0~1243 串口 2: 21E0~2243	订阅主题，16 进制，以 0x00 结尾	**读，串口 2 的 MQTT 订阅主题:** 发送，C9 03 21E0 0032 DF9D 返回，C9 03 64 7075623200000000000000000000000 00000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000 000000000000000000000000 98AA **写，设置串口 2 的 MQTT 订阅主题为“sub2”:** 发送, C9 10 21E0 0003 06 737562320000 6B1C 返回, C9 10 21E0 0003 9B8A

4. 通用配置流程

4.1 硬件连接



4.2 配置流程



附录

附录 1：计算 Modbus CRC-16 校验码的示例代码

对于 CRC-16 的实现，可参照查表法实现：

```
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, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40 } ;

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 };
```

```
unsigned short CRC16 ( unsigned char *puchMsg, unsigned short usDataLen )
{
   unsigned char uchCRCHi = 0xFF ; /* 高字节初始化值  */
    unsigned char uchCRCLo = 0xFF ; /* 低字节初始化值  */
    unsigned char uIndex ;
      while (usDataLen--)
    {
         uIndex = uchCRCLo ^ (*puchMsg++) ;
         uchCRCLo = uchCRCHi ^ auchCRCHi[uIndex] ;
         uchCRCHi = auchCRCLo[uIndex] ;
     }
        return ((uchCRCHi << 8) | ((unsigned char)uchCRCLo)) ;
}
```

联系方式

客服电话： 13651106881 (同微信号)

淘宝店销售：思可安

微信公众号：思可安

公司网站： www.cyckn.com

产品资料： www.cyckn.com/download