CO2-5000二氧化碳传感器
基本通讯协议

1 基本规范

本应用基本遵守 Modbus 规范，即客户机/服务器采用请求/应答的方式进行通讯。
本协议与 Modbus 不同的地方是,协议中全部数据顺序**统一为小端结构**.

1.1 传输模式

本应用采用 RTU(Remote Terminal Unit)传输模式。即报文中每个 8 位的字节含有两个 4 位的十六进制字符(0~F)。

RTU 信息(11 位)格式

编码系统	8 位二进制
字节中位的排列	1 起始位，8 数据位，1 奇偶校验位，1 停止位 (本应用奇偶校验位为"无")
波特率	默认为 9600，推荐 19200

1.2 帧结构

俗称通讯包，一个帧总长度不能超过 256 字节，包含 4 个区域：

地址域	功能码	数据域	校验域

- **地址域（1 字节）**
  地址域指的是从机的地址，有效的地址范围为 0~247。其中地址 0 只适用于广播命令。
  在本应用中，有如下约定：
  a. 用户可以通过专用工具一对一地改写从机的地址
  b. 从机可以响应主机请求中地址域为 0xFE 或 0x00 的命令，其中地址域为 0x00 的是广播命令，地址域为 0xFE 的只适用于一对一的情况

- **功能码（1 字节）**
  功能码详细定义和使用方法,见下文中的指令详解
  此处与标准 Modbus 协议略有不同,功能码后面紧跟的寄存器地址**全部采用小端结构**

- **数据域（0 ~ 252 字节）**

数据域采用"**LITTLE ENDIAN**"(小端)模式，即是低位字节在前高位字节在后。
例一：
1 个 16 位整形数 0x12AB ，数值发送顺序为→低位字节 0xAB，高位字节 0x12
例二：
1 个 32 位长整型数 0x11223344 ，数值发送顺序为→0x44，0x33，0x22，0x11

例三：

1 个 32 位数浮点数 100.3（0x42C8999A），数值发送顺序为→0x9A，0x99，0xC8，0x42

- **校验域（2 字节）**

校验域为 2 字节的 CRC，传输顺序为[CRC 低位], [CRC 高位]，顺序与数据域相同。

CRC 计算示例:

```
uint16_t CO2ModbusComm::modbus_calcuCRC(uint8_t *dataarray, uint16_t datalen)
{
    uint8_t uchCRCHi = 0xFF ; /* CRC 的高字节初始化*/
    uint8_t uchCRCLo = 0xFF ; /* CRC 的低字节初始化*/
    uint16_t uIndex ; /* CRC 查询表索引*/
    uint16_t crc;

    const uint8_t 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
        };
    const uint8_t 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,
```

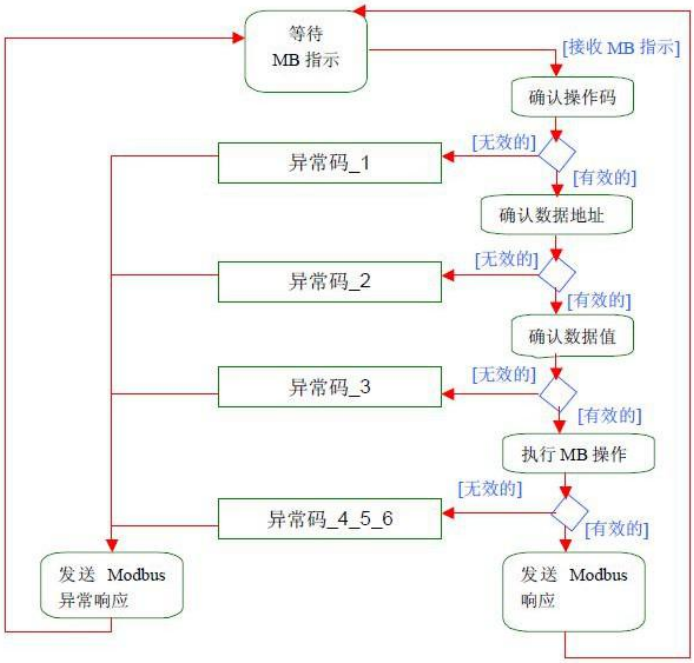
```
        0x62, 0x66, 0xA6, 0xA7, 0x67, 0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F,
        0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68, 0x78, 0xB8, 0xB9, 0x79, 0xBB,
        0x7B, 0x7A, 0xBA, 0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
        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
        };

    while (datalen--) /* 完成整个报文缓冲区*/
    {
        uIndex = uchCRCLo ^ *dataarray++;/* 计算 CRC */
        uchCRCLo = uchCRCHi ^ auchCRCHi[uIndex];
        uchCRCHi = auchCRCLo[uIndex];
    }
    crc = (uint16_t)uchCRCHi *256;
    crc += (uint16_t)uchCRCLo;
    return crc;
}
```

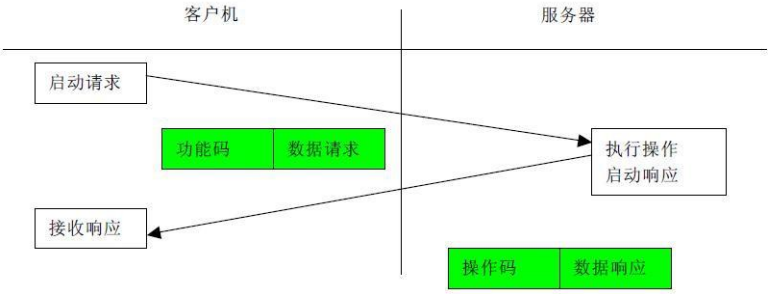
1.3 查询与响应

根据 MODBUS 规范，通讯事务总是由客户机（主机）发起，服务器（从机）要做出相应的响应。

从机通讯事务的处理流程如下：



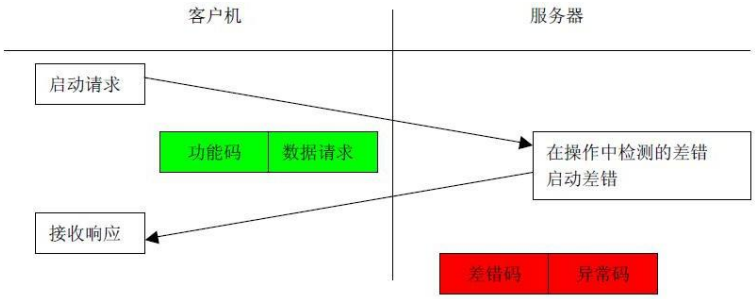
下图是一个无异常的通讯事务过程。



这里要强调两点：

- **异常响应**

当功能码或者数据请求为非法时，从机要反馈错误信息给主机。如下图所示：



异常响应的格式：

地址域	差错码	异常码	校验域

差错码为”请求的功能码+0x80”。
异常码定义如下：

Exception Code	Definition	Description
0x01	Illegal Function	The message received is not an allowable action for the addressed device.
0x02	Illegal Data Address	The address referenced in the function-dependent data section of the message is not valid in the addressed device
0x03	Illegal Data Value	The value referenced at the addressed device location is no within range.
0x04	Slave Device Failure	The addressed device has not been able to process a valid message due to a bad device state
0x05	Acknowledge	The slave has accepted a request and is processing it, but a long duration of time is required. This response is returned to prevent a timeout error from occurring in the master. The master can next issue a poll program complete message to determine if processing is completed.
0x06	Slave Device Busy	The addressed device has ejected a message due to a busy state. Retry later
0x07	NAK, Negative Acknowledge	The addressed device cannot process the current message. Issue a PROGRAM POLL to obtain device dependent error data.
0x08	Memory parity error	The slave attempted to read extended memory, but detected a parity error in the memory. The master can retry the request, but service may be required on the slave device.
0x09	Buffer Overflow	The data to be returned for the requested number of registers is greater than the available buffer space.
0x0A	CRC error	

2 功能码详解

本应用涉及到五类主要指令，下面一一详解。

2.1 设置类指令

- **读取系统设置**

Query (Host → Slave)		Response (Slave → Host)	
Slave Address	NN(1~247)	Slave Address	NN(1~247)
Function code	0x03	Function code	0x03
Address Low	M	Data length	n*2(2n bytes)
Address High	0	Data [2n]	
Register length Low	N	CRC Low	
Register length High	0	CRC High	
CRC Low			
CRC High			

目前支持地址读取: 0x0004
示例:
发送: FE 03 04 00 01 00 51 65
返回: FE 03 02 64 00 86 90
默认地址: 0x64

- **写入系统设置**

Query (Host → Slave)		Response (Slave → Host)	
Slave Address	NN(0,1~247)	Slave Address	NN(0,1~247)
Function code	0x10	Function code	0x10
Address Low	M	Address Low	M
Address High	0	Address High	0
Register length Low	N	Register length Low	N
Register length High	0	Register length High	0
Data length	n*2(2n bytes)	CRC Low	
Data [2n]		CRC High	
CRC Low			
CRC High			

目前支持地址写入:0x0004
示例:（更改地址为 100， 之前地址为 108）
发送: 6C 10 04 00 01 00 01 64 00 05 FE
返回: 6C 10 04 00 01 00 C8 14

➢ **设置测量参数**

Query (Host → Slave)		Response (Slave → Host)	
Slave Address	NN(1~247)	Slave Address	NN(1~247)
Function code	0x67	Function code	0x67
Parameter(1byte)	K	Parameter(1byte)	K
Data length(1byte)	m floats	Data length(1byte)	n floats
Para Data [4m bytes]	m*4 bytes	Receive Data [4n]	n*4 bytes
CRC Low		CRC Low	
CRC High		CRC High	

导入的参数也遵循 LITTLE ENDIAN 模式，低位字节在前。这些参数为单精度浮点型数据。
读取出来的测量值，遵循 LITTLE ENDIAN 模式，低位字节在前。被读取的数据也约定为单精度浮点型。
K 是参数代码:
Pressure: 0x01

这条指令是给 MQ 系列 CO2 传感器设置一个计算用的参数,例如大气
压示例: (设置气压 1013hpa:00 40 7D 44)当前地址为 100
发送: 64 67 01 01 00 40 7D 44 4D B0
返回: 64 67 01 01 00 40 7D 44 4D B0

➢ **读取测量参数**

Query (Host → Slave)		Response (Slave → Host)	
Slave Address	NN(1~247)	Slave Address	NN(1~247)
Function code	0x68	Function code	0x68
Parameter(1byte)	K	Parameter(1byte)	K
CRC Low		Data length(1byte)	n floats
CRC High		Receive Data [4n]	n*4 bytes
		CRC Low	
		CRC High	

导入的参数也遵循 LITTLE ENDIAN 模式，低位字节在前。这些参数为单精度浮点型数据。
读取出来的测量值，遵循 LITTLE ENDIAN 模式，低位字节在前。被读取的数据也约定为单精度浮点型。
K 是参数代码:
Pressure: 0x01

示例: (返回气压 1013hpa: 00 40 7D 44) 当前地址为 100
发送: 64 68 01 DE 1F
返回: 64 68 01 01 00 40 7D 44 B2 B0

- **读取测量值**

  K 是测量值代码:

  CO2PPM: 0x01

  CO2Temp: 0x02

Query (Host → Slave)		Response (Slave → Host)	
Slave Address	NN(1~247)	Slave Address	NN(1~247)
Function code	0x69	Function code	0x69
Parameter(1byte)	0x01 or 0x02	Parameter(1byte)	0x01 or 0x02
CRC Low		Data length(1byte)	n floats
CRC High		Receive Data [4n]	n*4 bytes
		Data status	4bytes (0x00 or 0xFF)
		CRC Low	
		CRC High	

导入的参数也遵循 LITTLE ENDIAN 模式，低位字节在前。这些参数为单精度浮点型数据。

读取出来的测量值，遵循 LITTLE ENDIAN 模式，低位字节在前。被读取的数据也约定为单精度浮点型。

示例：

1. 读取 CO2ppm 值，返回值有效，返回 522ppm

   发送：64 69 01 DF 8F

   返回：64 69 01 01 D5 9E 02 44 00 00 00 00 DA C2

2. 读取 CO2ppm 值，返回数据无效，无效数据值 500000

   发送：FE 69 01 FF A0

   返回：FE 69 01 01 00 24 F4 48 FF 00 00 00 E3 70

浮点数转化方式：

可以用联合体转换，定义一个包含四字节数组和一个浮点数的联合体

```
typedef union
{
    float fvalue;
    uint8_t byvalue[4];
}UNFLOATTOBYTES;

UNFLOATTOBYTES unPPMValue;
For( i = 0; i < 4; i++)
{
    unPPMValue. Byvalue[i] = recebuff[4+i];
}
```

注意：上述代码适合小端系统，如果使用大端系统（8051…）,联合体中 byvalue，需要从大到小赋值

CO2PPMInt16: 0x03

Query (Host → Slave)		Response (Slave → Host)	
Slave Address	NN(1~247)	Slave Address	NN(1~247)
Function code	0x69	Function code	0x69
Parameter(1byte)	0x03	Parameter(1byte)	0x03
CRC Low		Data length(1byte)	n
CRC High		Receive Data [4n]	n*4 bytes
		Data status	4bytes (0x00 or 0xFF)
		CRC Low	
		CRC High	

K= 0x03 时，功能跟 K=0x01 时一样，只是返回数据时 16bit 整形数，所以可以不需要联合体转换

示例：

3. 读取 CO2ppm 值，返回值有效，返回 522ppm

   发送：64 69 03 5E 4E

   返回：64 69 03 01 0A 02 00 00 00 00 00 00 9B F0

   0x0A 是 16bit 整形数的低字节，0x02 是高字节，合在一起就是 522

   红色部分是四字节状态

4. 读取 CO2ppm 值，返回数据无效，无效数据值 50000

   发送：FE 69 03 5E 4E

   返回：FE 69 03 01 50 C3 00 00 FF 00 00 00 FE 6B

1. 客户一点校准功能使能（带参考浓度） K= 0x80

Query (Host → Slave)		Response (Slave → Host)	
Slave Address	NN(0~247)	Slave Address	NN(0~247)
Function code	0x27	Function code	0x27
Sub Func Code	0x80	Sub Func Code	0x80
Ref CO2 ppm （ 4 bytes float）	4bytes	Ref CO2 ppm （ 4 bytes float）	4bytes
CRC Low	CRC Low	State byte	0x01(start),0xFF(error)
CRC High	CRC High	CRC Low	CRC Low
		CRC High	CRC High

MCU 接收到这个指令后，检查参考浓度是否在范围内，（限制在 5000ppm 以内）。如果参考浓度在范围内，则开始一点校准，返回的包中 state byte = 0x00，如果参考浓度有问题，则 state byte = 0xFF，一点校准不启动。一旦一点校准启动后，在没有完成前，查询状态返回“已启动”。

示例： 输入 400ppm 参考浓度,启动自校准

发送： FE 27 80 00 00 C8 43 15 F7

(400 ppm) CRCL CRCH

2. 客户一点校准功能，查询状态 K=0x81

Query (Host → Slave)		Response (Slave → Host)	
Slave Address	NN(0~247)	Slave Address	NN(0~247)
Function code	0x27	Function code	0x27
Sub Func Code	0x81	Sub Func Code	0x81
CRC Low	CRC Low	State byte	0x01(start),0x00(finished)
CRC High	CRC High	CRC Low	CRC Low
		CRC High	CRC High

示例: 查询当前自校准进程

发送: FE 27 81 CB A0

3. 使能或者禁止自校准 K = 0x66

Query (Host → Slave)		Response (Slave → Host)	
Slave Address	NN(0~247)	Slave Address	NN(0~247)
Function code	0x27	Function code	0x27
Sub Func Code	0x66	Sub Func Code	0x66
State	0xFF(Disable) 0x00(Enable)	State	0xFF(Disable) 0x00(Enable)
CRC Low	CRC Low	CRC Low	CRC Low
CRC High	CRC High	CRC High	CRC High

示例：

1、使能传感器自动校准功能

发送：64 27 66 00 84 BF

返回：64 27 66 00 84 BF

2、禁止自动校准功能

发送：64 27 66 FF C4 FF

返回：64 27 66 FF C4 FF

4、读取自校准状态 K = 0x67

Query (Host → Slave)		Response (Slave → Host)	
Slave Address	NN(0~247)	Slave Address	NN(0~247)
Function code	0x27	Function code	0x27
Sub Func Code	0x67	Sub Func Code	0x67
CRC Low	CRC Low	State	0xFF(Disable) 0x00(Enable)
CRC High	CRC High	CRC Low	CRC Low
		CRC High	CRC High

示例：读取自校准状态

发送：64 27 67 6A 05

返回：64 27 67 00 85 2F（开启）

64 27 67 FF C5 6F（关闭）

5、读取自校准周期(小时数) K = 0x69

Query (Host → Slave)		Response (Slave → Host)	
Slave Address	NN(0~247)	Slave Address	NN(0~247)
Function code	0x27	Function code	0x27
Sub Func Code	0x69	Sub Func Code	0x69
CRC Low	CRC Low	Period(2bytes)	N hours
CRC High	CRC High	CRC Low	CRC Low
		CRC High	CRC High

示例：读取自校准周期

发送：64 27 69 EB C1

返回：64 27 69 18 00 85 60 （24 小时）

6、写入自校准周期(小时数) K = 0x6A (最小 24hours, 最多 720hours)

Query (Host → Slave)		Response (Slave → Host)	
Slave Address	NN(0~247)	Slave Address	NN(0~247)
Function code	0x27	Function code	0x27
Sub Func Code	0x6A	Sub Func Code	0x6A
Period(2bytes)	N hours	State(1byte)	0x00 OK 0x01 less than 24 0x02 more than 720
CRC Low	CRC Low	CRC Low	CRC Low
CRC High	CRC High	CRC High	CRC High

示例：写入自动校准周期（168 小时）

发送：64 27 6A A8 00 00 A0

返回：64 27 6A 00 81 BF（写入成功）