

串口通信协议说明书



www.rockmong.com

上海岩獾科技有限公司

V1.1

目录

串口通信协议说明书.....	1
一、 串口配置.....	3
二、 帧格式介绍.....	3
三、 IO 功能指令.....	4
3.1 初始化配置 IO 口 (0x01 0x00).....	4
3.2 读取输入/输出状态 (0x01 0x01).....	5
3.3 写输入/输出状态 (0x01 0x02).....	5
四、 SPI 功能指令.....	6
4.1 初始化配置 SPI 通道 (0x05 0x00).....	6
4.2 写数据 (0x05 0x01).....	7
4.3 读数据 (0x05 0x02).....	7
4.4 写读数据 (0x05 0x03).....	7
五、 I2C 功能指令.....	8
5.1 初始化配置 I2C 通道 (0x06 0x00).....	8
5.2 写数据 (0x06 0x01).....	9
5.3 读数据 (0x06 0x02).....	9
5.4 写读数据 (0x06 0x03).....	10
附录——ModbusCRC16 算法.....	11

一、串口配置

波特率：2Mbps

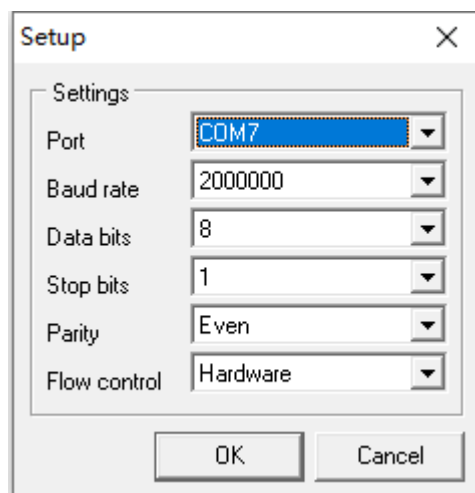
数据位：8

停止位：1

奇偶校验：偶校验（Even）

硬件流控制：RTS/CTS

下图是 SSCOM 串口工具的串口设置示例：



二、帧格式介绍

帧格式：

帧头 1	帧头 2	功能码 1	功能码 2	数据长度	数据域	校验码
1 Byte	1 Byte	1 Byte	1 Byte	2 Bytes	n Bytes	2 Bytes

帧数据说明：

帧数据	说明
帧头 1	电脑发送写 0xFA 模块返回 0xFB
帧头 2	0x02
功能码 1	0x01: IO 功能
功能码 2	详见各功能码 1 模块中的介绍
数据长度	数据域的数据长度
数据域	详见各功能码 2 模块中的介绍
校验码	检验数据，使用 ModbusCRC16，具体算法见附录

三、IO 功能指令

IO 功能码:

功能码 1: 0x01

功能码 2:

0x00: 初始化配置 IO 口

0x01: 读取输入 IO 口状态

0x03: 写入输出 IO 口状态

0x04: 读取输出 IO 口状态

3.1 初始化配置 IO 口 (0x01 0x00)

在读写 IO 口之前，需要在程序初始化的时候和模块上电后，初始配置一次 IO。

发送数据域数据:

数据	说明
Byte 1	引脚编号: P0 口, 写 0 P1 口, 写 1 依次类推...
Byte 2	设置模式: 输入写 0 输出写 1 开漏写 2
Byte 3	内部上拉下拉电阻配置。仅在输入模式下有效。 不配置写 0 配置上拉写 1 下拉写 2

接收数据域数据:

数据	说明
Byte 1	0 表示成功。小于 0 为异常

示例 1, 配置 P0 口为上拉输入:

发送: FA 02 01 00 00 03 00 00 01 20 C9

接收: FB 02 01 00 00 01 00 6C 7D

示例 2, 配置 P2 口为输出:

发送: FA 02 01 00 00 03 02 01 00 41 59

接收: FB 02 01 00 00 01 00 6C 7D

3.2 读取输入口状态 (0x01 0x01)

发送数据域数据:

数据	说明
Byte 1	引脚编号: P0 口, 写 0 P1 口, 写 1 依次类推...

接收数据域数据:

数据	说明
Byte 1	0 表示成功。小于 0 为异常
Byte 2	0: 低电平 1: 高电平

示例 1, 读取 P0 口输入状态, 模块返回高电平:

发送: FA 02 01 01 00 01 00 7D 41

接收: FB 02 01 01 00 02 00 01 71 2D

3.3 写输出口状态 (0x01 0x02)

发送数据域数据:

数据	说明
Byte 1	引脚编号: P0 口, 写 0 P1 口, 写 1 依次类推...
Byte 2	0: 低电平 1: 高电平

接收数据域数据:

数据	说明
Byte 1	0 表示成功。小于 0 为异常

示例 1, 写 P2 口为低电平:

发送: FA 02 01 02 00 02 02 00 34 41

接收: FB 02 01 02 00 01 00 6D C5

示例 2, 写 P2 口为高电平:

发送: FA 02 01 02 00 02 02 01 F5 81

接收: FB 02 01 02 00 01 00 6D C5

四、SPI 功能指令

SPI 功能码：

功能码 1：0x05

功能码 2：

0x00: 初始化配置 SPI 通道

0x01: 写数据

0x03: 读数据

0x04: 写读数据

4.1 初始化配置 SPI 通道 (0x05 0x00)

在读写 SPI 之前，需要在程序初始化的时候和模块上电后，初始配置一次 SPI。

发送数据域数据：

数据	说明
Byte 1	通道编号： 通道 0，写 0 通道 1，写 1 依次类推...
Byte 2	设置模式： 主机模式，写 0 从机模式，写 1
Byte 3~6	波特率。小端格式。 例如，375KHz，写 D8 B8 05 00
Byte 7	CPHA：时钟相位 第一边沿采样，写 0 第二边沿采样，写 1
Byte 8	CPOL：时钟极性 时钟空闲时为低电平，写 0 时钟空闲时为高电平，写 1
Byte 9	片选极性： 低电平使能，写 0 高电平使能，写 1

接收数据域数据：

数据	说明
Byte 1	0 表示成功。小于 0 为异常

4.2 写数据 (0x05 0x01)

写入指定长度的数据

发送数据域数据:

数据	说明
Byte 1	通道编号: 通道 0, 写 0 通道 1, 写 1 依次类推...
Byte 2~5	要写入数据的长度。小端格式。 例如, 长度为 10 给字节, 写 0A 00 00 00
Byte 6~n	要写入的数据

接收数据域数据:

数据	说明
Byte 1	0 表示成功。小于 0 为异常

4.3 读数据 (0x05 0x02)

读取指定长度的数据

发送数据域数据:

数据	说明
Byte 1	通道编号: 通道 0, 写 0 通道 1, 写 1 依次类推...
Byte 2~5	要读取数据的长度。小端格式。 例如, 长度为 10 给字节, 写 0A 00 00 00

接收数据域数据:

数据	说明
Byte 1	0 表示成功。小于 0 为异常
Byte 2~n	读取返回的数据

4.4 写读数据 (0x05 0x03)

写入指定长度数据, 并读取指定长度的数据

发送数据域数据:

数据	说明
Byte 1	通道编号: 通道 0, 写 0 通道 1, 写 1 依次类推...
Byte 2~5	要写入数据的长度。小端格式。 例如, 长度为 10 给字节, 写 0A 00 00 00
Byte 6~9	要读取数据的长度。小端格式。 例如, 长度为 10 给字节, 写 0A 00 00 00
Byte 10~n	要写入的数据

接收数据域数据:

数据	说明
Byte 1	0 表示成功。小于 0 为异常
Byte 2~n	读取返回的数据

五、I2C 功能指令

I2C 功能码:

功能码 1: 0x06

功能码 2:

0x00: 初始化配置 I2C 通道

0x01: 写数据

0x03: 读数据

0x04: 写读数据

5.1 初始化配置 I2C 通道 (0x06 0x00)

在读写 I2C 之前, 需要在程序初始化的时候和模块上电后, 初始配置一次 I2C。

发送数据域数据:

数据	说明
Byte 1	通道编号: 通道 0, 写 0 通道 1, 写 1 依次类推...
Byte 2	设置模式: 主机模式, 写 0 从机模式, 写 1

Byte 3	地址长度。 7bit, 写 0
Byte 4~5	本机地址。 主机模式下, 写 00 00 从机模式下, 写从机地址
Byte 6~9	时钟频率。小端格式。 例如: 100KHz, 写 0A 86 01 00
Byte 10	内部上拉。 不需要, 写 0 需要配置为上拉, 写 1

接收数据域数据:

数据	说明
Byte 1	0 表示成功。小于 0 为异常

5.2 写数据 (0x06 0x01)

写入指定长度的数据

发送数据域数据:

数据	说明
Byte 1	通道编号: 通道 0, 写 0 通道 1, 写 1 依次类推...
Byte 2~3	从机地址。小端格式 例如从机地址为 0x50, 写 50 00
Byte 4~7	超时时间。单位毫秒。小端格式。 例如, 超时设置为 1s, 写入 E8 03 00 00
Byte 8~9	要写入数据的长度。小端格式。 例如, 长度为 10 给字节, 写 0A 00
Byte 10~n	要写入的数据

接收数据域数据:

数据	说明
Byte 1	0 表示成功。小于 0 为异常

5.3 读数据 (0x06 0x02)

读取指定长度的数据

发送数据域数据：

数据	说明
Byte 1	通道编号： 通道 0，写 0 通道 1，写 1 依次类推...
Byte 2~3	从机地址。小端格式 例如从机地址为 0x50，写 50 00
Byte 4~7	超时时间。单位毫秒。小端格式。 例如，超时设置为 1s，写入 E8 03 00 00
Byte 8~9	要读取数据的长度。小端格式。 例如，长度为 10 给字节，写 0A 00

接收数据域数据：

数据	说明
Byte 1	0 表示成功。小于 0 为异常
Byte 2~n	读取返回的数据

5.4 写读数据 (0x06 0x03)

写入指定长度数据，并读取指定长度的数据

发送数据域数据：

数据	说明
Byte 1	通道编号： 通道 0，写 0 通道 1，写 1 依次类推...
Byte 2~3	从机地址。小端格式 例如从机地址为 0x50，写 50 00
Byte 4~7	超时时间。单位毫秒。小端格式。 例如，超时设置为 1s，写入 E8 03 00 00
Byte 8~9	要写入数据的长度。小端格式。 例如，长度为 10 给字节，写 0A 00
Byte 10~11	要读取数据的长度。小端格式。 例如，长度为 10 给字节，写 0A 00
Byte 11~n	要写入的数据

接收数据域数据：

数据	说明
Byte 1	0 表示成功。小于 0 为异常

Byte 2~n	读取返回的数据
----------	---------

附录——ModbusCRC16 算法

这里仅以 C 语言为例。其他语言类似。

1、查表法（推荐）：

```
uint16_t const CRC16MODBUS_Table[256]= {
    /* 16: 8005 reflected */
    0x0000 0xc0c1 0xc181 0x0140 0xc301 0x03c0 0x0280 0xc241 0xc601 0x06c0 0x0780 0xc7
    41 0x0500 0xc5c1 0xc481 0x0440
    0xcc01 0x0cc0 0x0d80 0xcd41 0x0f00 0xcfc1 0xce81 0x0e40 0x0a00 0xcac1 0xcb81 0x0b4
    0 0xc901 0x09c0 0x0880 0xc841
    0xd801 0x18c0 0x1980 0xd941 0x1b00 0xdbc1 0xda81 0x1a40 0x1e00 0xdec1 0xdf81 0x1
    f40 0xdd01 0x1dc0 0x1c80 0xdc41
    0x1400 0xd4c1 0xd581 0x1540 0xd701 0x17c0 0x1680 0xd641 0xd201 0x12c0 0x1380 0x
    d341 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 0xef
    41 0x2d00 0xedc1 0xec81 0x2c40
    0xe401 0x24c0 0x2580 0xe541 0x2700 0xe7c1 0xe681 0x2640 0x2200 0xe2c1 0xe381 0x2
    340 0xe101 0x21c0 0x2080 0xe041
    0xa001 0x60c0 0x6180 0xa141 0x6300 0xa3c1 0xa281 0x6240 0x6600 0xa6c1 0xa781 0x6
    740 0xa501 0x65c0 0x6480 0xa441
    0x6c00 0xacc1 0xad81 0x6d40 0xaf01 0x6fc0 0x6e80 0xae41 0xaa01 0x6ac0 0x6b80 0xab
    41 0x6900 0xa9c1 0xa881 0x6840
    0x7800 0xb8c1 0xb981 0x7940 0xbb01 0x7bc0 0x7a80 0xba41 0xbe01 0x7ec0 0x7f80 0xb
    f41 0x7d00 0xbdc1 0xbc81 0x7c40
    0xb401 0x74c0 0x7580 0xb541 0x7700 0xb7c1 0xb681 0x7640 0x7200 0xb2c1 0xb381 0x
    7340 0xb101 0x71c0 0x7080 0xb041
    0x5000 0x90c1 0x9181 0x5140 0x9301 0x93c0 0x5280 0x9241 0x9601 0x96c0 0x5780 0x9
    741 0x5500 0x95c1 0x9481 0x5440
    0x9c01 0x5cc0 0x5d80 0x9d41 0x5f00 0x9fc1 0x9e81 0x5e40 0x5a00 0x9ac1 0x9b81 0x5b
    40 0x9901 0x59c0 0x5880 0x9841
    0x8801 0x48c0 0x4980 0x8941 0x4b00 0x8bc1 0x8a81 0x4a40 0x4e00 0x8ec1 0x8f81 0x4f
    40 0x8d01 0x4dc0 0x4c80 0x8c41
    0x4400 0x84c1 0x8581 0x4540 0x8701 0x47c0 0x4680 0x8641 0x8201 0x42c0 0x4380 0x8
    341 0x4100 0x81c1 0x8081 0x4040
};
```

```
uint16_t rshiftu16(uint16_t value, int nb)
{
    return (uint16_t)((value >> nb) & ~((( uint16_t) 0x8000) >> (nb-1)));
}

uint16_t CRC16MODBUS_Calc(uint8_t *q, uint32_t len)
{
    uint16_t crc = 0xffff;
    while (len-- > 0)
        crc=(rshiftu16(crc,8) ^ CRC16MODBUS_Table[(crc ^ *q++) & 0xff]);
    return crc;
}
```

2、计算法

```
uint16_t CRC16MODBUS_Calc(const uint8_t *data, uint16_t length) {
    uint16_t crc = 0xFFFF;

    for (uint16_t i = 0; i < length; ++i) {
        crc ^= (uint16_t)data[i];

        for (uint8_t j = 0; j < 8; ++j) {
            if (crc & 0x0001) {
                crc = (crc >> 1) ^ 0xA001;
            } else {
                crc >>= 1;
            }
        }
    }

    return crc;
}
```