

PM98xx 电参数测量仪 MODBUS 通讯协议

一、Modbus 规约说明

规约采用 Modbus 规约 RTU 模式，可以方便地与多种组态软件相连接，其通讯驱动与 Modicon Modbus_RTU 格式完全兼容。

字节格式

每字节含 8 位二进制码，传输时加上一个起始位(0)，一个停止位(1)，共 10 位。其传输序列如上图所示，D0 是字节的最低有效位，D7 是字节的最高有效位。先传低位，后传高位。



通讯数据格式

通讯时数据以字(WORD— 2 字节)的形式回送，回送的每个字中，高字节在前，低字节在后，如果 2 个字连续回送(如：浮点或长整形)，则高字在前，低字在后。

| 数据类型 | 寄存器数 | 字节数 | 说明 |
|------|------|-----|------------------|
| 字节数据 | | 1 | |
| 整形数据 | 1 | 2 | 一次送回，高字节在前，低字节在后 |
| 长整型数 | 2 | 4 | 分两个字回送，高字在前，低字在后 |
| 浮点数据 | | | |

帧格式

读取仪表寄存器内容（功能码 03H）

查询寄存器的采用功能码 03H。

上位机发送的帧格式：

| 顺序 | 代码 | 示例 | 说明 |
|----|-----------|------|-------------------|
| 1 | 仪表地址 | 01 H | 仪表的通讯地址（1-255 之间） |
| 2 | 03H | 03 H | 功能码 |
| 3 | 起始寄存器地址高字 | 10 H | 寄存器起始地址 |
| 4 | 起始寄存器地址低字 | 00 H | |
| 5 | 寄存器个数高字节 | 00 H | 寄存器个数 |
| 6 | 寄存器个数低字节 | 02 H | |
| 7 | 校验高字节 | C0 H | CRC 校验数据 |
| 8 | 校验低字节 | CB H | |

通信正常情况下，仪表回送的帧格式：

| 顺序 | 代码 | 说明 |
|-----|---------------|-------------------|
| 1 | 仪表地址 | 仪表的通讯地址（1-255 之间） |
| 2 | 03H | 功能码 |
| 3 | 回送数据域字节数(M) | |
| 4 | 第一个寄存器数据高字节 | |
| 5 | 第一个寄存器数据低字节 | |
| ... | | |
| | 第 N 个寄存器数据高字节 | |
| | 第 N 个寄存器数据低字节 | |
| M+4 | CRC 校验高字节 | |
| M+5 | CRC 校验低字节 | |

M=N*2

通信错误情况下, 仪表回送的帧格式:

| 顺序 | 代码 | 示例 | 说明 |
|----|-----------|-----|--------------------|
| 1 | 仪表地址 | 01H | 仪表的通讯地址 (1-255 之间) |
| 2 | 83H | 83H | 功能码 |
| 3 | 02H | 02H | 错误代码 |
| 4 | CRC 校验高字节 | C0H | |
| 5 | CRC 校验低字节 | F1H | |

以下情况为 03H 通信错误情况:

- 寄存器地址不存在。
- 寄存器个数错误。
- 上位机发送的 CRC 校验码错误。
- 干扰导致仪表接收的数据错误。

设置仪表寄存器内容 (功能码 10H)

PM98xx 仪表设置都为多寄存器 (至少为 2 个), 功能码 10H。

上位机发送的帧格式:

| 顺序 | 代码 | 示例 | 说明 |
|-------|---------------|------|-----------------------------|
| 1 | 仪表地址 | 01 H | 仪表的通讯地址 (1-255 之间) |
| 2 | 10H | 10 H | 功能码 |
| 3 | 寄存器起始地址高字节 | 15 H | 寄存器地址 1520H |
| 4 | 寄存器起始地址低字节 | 20 H | |
| 5 | 寄存器个数高字节 | 00 H | 00H |
| 6 | 寄存器个数低字节 | 02 H | 整形数据: 01H 浮点数据、长整形数: 02H |
| 7 | 字节数 (M) | 04 H | 整形数据 : 02H 浮点数、长整形数: 04H |
| 8 | 第一个寄存器数据高字节 | 42 H | 设置的浮点数据为 100 |
| 9 | 第一个寄存器数据低字节 | C8 H | |
| | | | |
| M+6 | 第 N 个寄存器数据高字节 | 00 H | |
| M+7 | 第 N 个寄存器数据低字节 | 00 H | |
| M+8 | CRC 校验高字节 | 96 H | CRC 校验数据 |
| M+9 | CRC 校验低字节 | A1 H | |

M=N*2

通信正常情况下, 仪表回送的帧格式:

| 顺序 | 代码 | 示例 | 说明 |
|----|-----------|------|--------------------|
| 1 | 仪表地址 | 01 H | 仪表的通讯地址 (1-255 之间) |
| 2 | 10H | 10 H | 功能码 |
| 3 | 起始地址高字节 | 15 H | 寄存器起始地址 1520H |
| 4 | 起始地址低字节 | 20 H | |
| 5 | 寄存器个数高字节 | 00 H | 寄存器个数 2 |
| 6 | 寄存器个数低字节 | 02 H | |
| 7 | CRC 校验高字节 | 44 H | CRC 校验数据 |
| 8 | CRC 校验低字节 | 0E H | |

通信错误情况下, 仪表回送的帧格式:

| 顺序 | 代码 | 说明 |
|----|-----------|--------------------|
| 1 | 仪表地址 | 仪表的通讯地址 (1-255 之间) |
| 2 | 90H | 功能码 |
| 3 | 03H | 错误代码 |
| 4 | CRC 校验高字节 | |
| 5 | CRC 校验低字节 | |

以下情况为 10H 通信错误情况：

- 寄存器地址不存在。
- 寄存器个数错误。
- 部分寄存器不能设置(选件部分)。
- 上位机发送的 CRC 校验码错误。
- 干扰导致仪表接收的数据错误。

注：

以上介绍中 CRC 校验为 16 位，高字节在前，低字节在后。

CRC 检验从第 1 字节开始至 CRC 校验高字节前面的字节数据结束。CRC 检验码的计算例程见附录。

二、仪表数据寄存器地址

浮点数据为单精度四字节浮点数据。以“选件”标注的寄存器是仪表的附加功能，这类寄存器能够查询，但不能设置。R：表示可读即支持 03 H 命令。W：表示可写即支持 10 H 命令。

注：03H 命令可以读取全部寄存器地址，10H 命令可以写入全部寄存器地址。

仪表参数与实时测量数据。

| 数 据 名 称 | 单位 | 数据格式 | 起始地址 | 寄存器数 | 读写 | 备 注 |
|---------------|----|-------|-------|------|-----|--|
| 仪表信息 | | | | | | |
| 仪表型号 | | ASCII | 0002H | 3 | R | 98 系列 |
| 软件版本 | | ASCII | 0005H | 3 | R | V1.00 (更新版本另作说明) |
| 硬件版本 | | ASCII | 0008H | 3 | R | V1.00 (更新版本另作说明) |
| 扩展功能 1 | | ASCII | 000BH | 3 | R | -G5(带谐波) 或 NO |
| 扩展功能 2 | | ASCII | 000EH | 3 | R | -C3(串口) |
| 扩展功能 3 | | ASCII | 0011H | 3 | R | -USB |
| 扩展功能 4 | | ASCII | 0014H | 3 | R | -C7(EHTERNET) 或 NO |
| 扩展功能 5 | | ASCII | 0017H | 3 | R | -EX1(10V)或-EX2(2V)或 NO |
| 扩展功能 6 | | ASCII | 001AH | 3 | R | -R01(继电器输出)或 NO |
| 扩展功能 7 | | ASCII | 001DH | 3 | R | |
| 扩展功能 8 | | ASCII | 0020H | 3 | R | |
| 仪表参数 | | | | | | |
| 电压电流倍率是否打开 | | ULong | 0040H | 2 | W/R | 0(关)、1(开) |
| 电压倍率 | | Float | 0042H | 2 | W/R | 0.01 ~ 999 |
| 电流倍率 | | Float | 0044H | 2 | W/R | 0.01 ~ 999 |
| 功率倍率 | | Float | 0046H | 2 | W/R | 0.01 ~ 999 |
| 外接电流传感器系数(选项) | | Float | 0048H | 2 | W/R | 0.01 ~ 999 |
| 备用 | | | 00AAH | 2 | | |
| 数据更新间隔时间 | | ULong | 004CH | 2 | W/R | 0(0.1 秒)、1(0.25 秒)、 2(0.5 秒)、3(1 秒)、4(2 秒)、 5(5 秒) |
| 备用 | | | 004EH | 2 | | |
| 备用 | | | 0050H | 2 | | |
| 备用 | | | 0052H | 2 | | |
| 备用 | | | 0054H | 2 | | |
| 备用 | | | 0056H | 2 | | |
| 备用 | | | 0058H | 2 | | |
| 备用 | | | 005AH | 2 | | |
| 备用 | | | 005CH | 2 | | |
| 备用 | | | 005EH | 2 | | |
| 备用 | | | 0060H | 2 | | |
| 备用 | | | 0062H | 2 | | |
| 备用 | | | 0064H | 2 | | |
| 备用 | | | 0066H | 2 | | |
| 备用 | | | 0068H | 2 | | |
| 备用 | | | 006AH | 2 | | |
| 备用 | | | 006CH | 2 | | |
| 按键保护 | | ULong | 006EH | 2 | W/R | 0(不保护)、1(保护) |
| 锁定数据 | | ULong | 0070H | 2 | W/R | 0(解锁)、1(锁定) |

| | | | | | | |
|-------------|-----|-------|-------|---|-----|--------------------------|
| 备用 | | | 0072H | 2 | | |
| 备用 | | | 0074H | 2 | | |
| 备用 | | | 0076H | 2 | | |
| 备用 | | | 0078H | 2 | | |
| 备用 | | | 007AH | 2 | | |
| 备用 | | | 007CH | 2 | | |
| 报警总控制 | | ULong | 007EH | 2 | W/R | 0(关)、1(开) |
| 电压报警控制 | | ULong | 0080H | 2 | W/R | 0(关)、1(开) |
| 电压报警上限值 | V | Float | 0082H | 2 | W/R | 0.0000 ~ 99999 |
| 电压报警下限值 | V | Float | 0084H | 2 | W/R | 0.0000 ~ 99999 |
| 电流报警控制 | | ULong | 0086H | 2 | W/R | 0(关)、1(开) |
| 电流报警上限值 | A | Float | 0088H | 2 | W/R | 0.0000 ~ 99999 |
| 电流报警下限值 | A | Float | 008AH | 2 | W/R | 0.0000 ~ 99999 |
| 有功功率报警控制 | | ULong | 008CH | 2 | W/R | 0(关)、1(开) |
| 有功功率报警上限值 | W | Float | 008EH | 2 | W/R | 0.0000 ~ 99999 |
| 有功功率报警下限值 | W | Float | 0090H | 2 | W/R | 0.0000 ~ 99999 |
| 视在功率报警控制 | | ULong | 0092H | 2 | W/R | 0(关)、1(开) |
| 视在功率报警上限值 | VA | Float | 0094H | 2 | W/R | 0.0000 ~ 99999 |
| 视在功率报警下限值 | VA | Float | 0096H | 2 | W/R | 0.0000 ~ 99999 |
| 无功功率报警控制 | | ULong | 0098H | 2 | W/R | 0(关)、1(开) |
| 无功功率报警上限值 | Var | Float | 009AH | 2 | W/R | 0.0000 ~ 99999 |
| 无功功率报警下限值 | Var | Float | 009CH | 2 | W/R | 0.0000 ~ 99999 |
| 功率因数报警控制 | | ULong | 009EH | 2 | W/R | 0(关)、1(开) |
| 功率因数报警上限值 | | Float | 00A0H | 2 | W/R | 0.0000 ~ 99999 |
| 功率因数报警下限值 | | Float | 00A2H | 2 | W/R | 0.0000 ~ 99999 |
| 电压频率报警控制 | | ULong | 00A4H | 2 | W/R | 0(关)、1(开) |
| 电压频率报警上限值 | Hz | Float | 00A6H | 2 | W/R | 0.0000 ~ 99999 |
| 电压频率报警下限值 | Hz | Float | 00A8H | 2 | W/R | 0.0000 ~ 99999 |
| 备用 | | | 00AAH | 2 | | |
| 备用 | | | 00ACH | 2 | | |
| 备用 | | | 00AEH | 2 | | |
| 备用 | | | 00B0H | 2 | | |
| 备用 | | | 00B2H | 2 | | |
| 备用 | | | 00B4H | 2 | | |
| 备用 | | | 00B6H | 2 | | |
| 备用 | | | 00B8H | 2 | | |
| 备用 | | | 00BAH | 2 | | |
| 备用 | | | 00BCH | 2 | | |
| 备用 | | | 00BEH | 2 | | |
| 备用 | | | 00C0H | 2 | | |
| 备用 | | | 00C2H | 2 | | |
| 备用 | | | 00C4H | 2 | | |
| 备用 | | | 00C6H | 2 | | |
| 报警延迟次数 | | ULong | 00C8H | 2 | W/R | 0 ~ 99 |
| 是否允许零点报警 | | ULong | 00CAH | 2 | W/R | 0(关)、1(开) |
| 报警继电器动作逻辑方式 | | ULong | 00CCH | 2 | W/R | 0(高低模式)、1(合格不合格模式) |
| 是否允许报警显示闪烁 | | ULong | 00CEH | 2 | W/R | 0(关)、1(开) |
| 报警时声音长度 | | ULong | 00DOH | 2 | W/R | 1 ~ 9999 |
| 常规数据 | | | | | | |
| 电压 | V | Float | 0100H | 2 | R | 数值与测量模式有关 (RMS、AC、DC) |
| 电流 | A | Float | 0102H | 2 | R | 数值与测量模式有关 (RMS、AC、DC) |
| 有功功率 | W | Float | 0104H | 2 | R | 数值与测量模式有关 (RMS、AC) |
| 无功功率 | Var | Float | 0106H | 2 | R | 数值与测量模式有关 (RMS、AC、DC) |

附录 1: CRC 校验码的计算:

/*****

本文中所有的数据及显示皆为 16 进制 本文中的 Crc 校验以查表的方式进行
本程序是 Turbo C++(Ver3.0)的格式,运行环境为 DOS 操作系统

*****/

```
#include<stdio.h>
```

```
unsigned char txd pointer; unsigned char rxd pointer;
```

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

/* CRC 低位字节值表*/
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 char uchCRCHi = 0xFF : /* 高 CRC 字节初始化 */
```

```
unsigned char uchCRCLo = 0xFF ; /* 低 CRC 字节初始化 */
```

```
unsigned uIndex ;
while (usDataLen-- ) /* 传输消息缓冲区 */
{
    uIndex = uchCRCHi ^ *puchMsg++ ; /* 计算 CRC */
    uchCRCHi = uchCRCLo ^ auchCRCHi[uIndex]
    uchCRCLo = auchCRCLo[uIndex] ;
}
return (uchCRCHi << 8 | uchCRCLo) ;}
union{unsigned int i;unsigned char c[2];}cov; union{float f;unsigned char c[4];}covf;
void main()
{
    unsigned char send[30]; unsigned int crc;
    int i;
    printf("\n\nCrc Calculate example:"); txd_pointer=0;
    send[txd_pointer++]=0x1;
    send[txd_pointer++]=0x3; send[txd_pointer++]=0x10; send[txd_pointer++]=2;
    send[txd_pointer++]=0x0; send[txd_pointer++]=0x2;
    printf("\nData:"); for(i=0;i<txd_pointer;i++)printf("%02x,", send[i]); //显示被校验的数据
    cov.i=CRC16(send,txd_pointer); //开始 CRC 校验计算
    send[txd_pointer++]=cov.c[1]; // cov.c[1]为 CRC 校验的高字节
    send[txd_pointer++]=cov.c[0]; // cov.c[0]为 CRC 校验的低字节
    printf("\nCRc=%02x,%02x", cov.c[1], cov.c[0]); //显示 CRC 校验的值
}
```

如有疑问请联系技术支持：13553893313（容生）