S3303

16 路隔离集电极开路输出

使用说明书

（Rev1.2　　2007.12.10）

上海世杰电子有限责任公司

销售：**shjelectronic@gmail.com**

技术支持：**shjsupport@gmail.com**

一、概述

S3303 是 16 路集电极开路输出模块,输出光耦隔离， 输出总线为 RS485，标准 Modbus 协议，输出高速光偶隔离并有防雷、静电保护，有效降低通讯对数据采集的干扰。设计上还通过使用内，外部双看门狗，表面贴装工艺提高系统稳定性。

二、技术参数

输出通道----------16
输出类型----------隔离集电极开路，可直接驱动继电器
隔离电压---------- > 3000V
输出总线----------光耦隔离 RS485
输出保护----------防雷，静电
电源---------- 9~24V(AC/DC),标准 24VAC
功耗---------- <0.6W
工作温度----------0℃~+70℃
存储温度---------- -20℃~+85℃
相对湿度---------- 5%~95%RH (无凝露)
尺寸----------100*69*25mm

三、接线说明

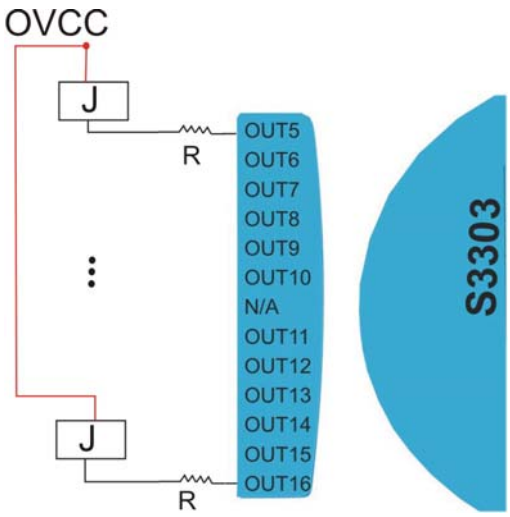

图 1　输出驱动继电器

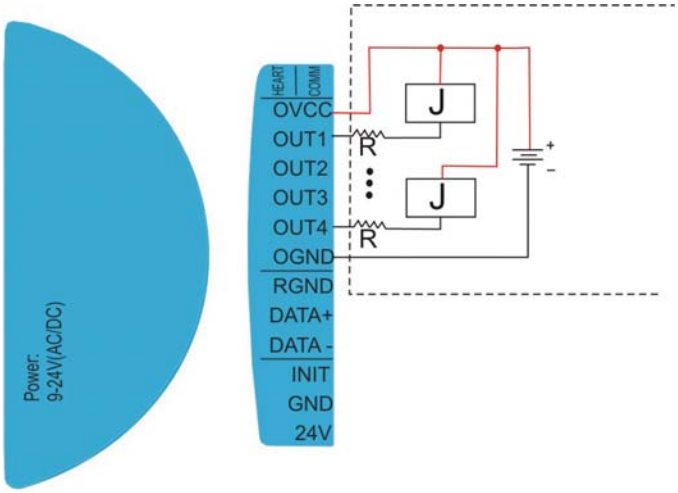

图 2 输出接线，R 为限流电阻，电流小于 40mA

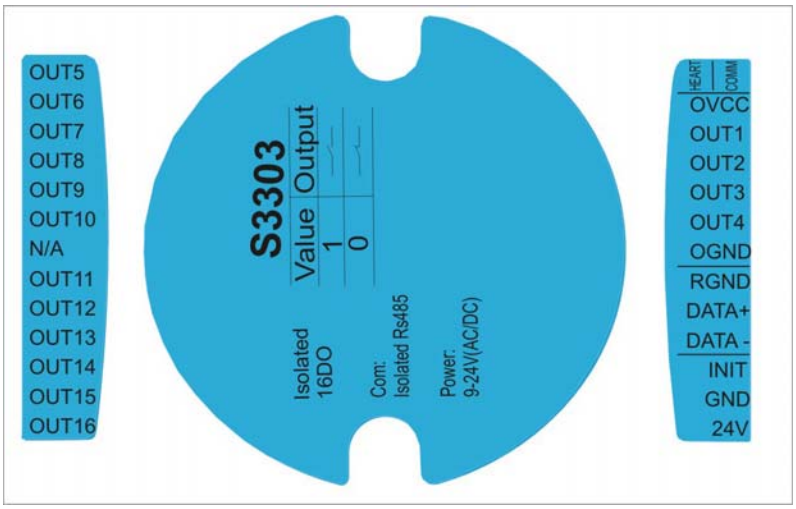

图 3 整体端子定义框图

1、 输出

OUT1 ~ OUT16:集电极开路输出通道 1 到 16

OVCC:集电极输出共源端

OGND:集电极输出地端

2、 电源

直流：24V 接正极

GND 接负极

注： 有反接保护

交流：不分正负极

3、 RS485 输出

DATA+接 485 总线 A 端

DATA-接 485 总线 B 端

RGND：悬空或接 RS485 屏蔽地

4、 参数复位

跳线跳在 INIT 端，下面这些参数恢复为出厂值。

- 地址：254
- 波特率：19200

跳线跳在 NULL 端，使用用户配置参数

5、 人机界面

Heart： 系统工作时这个 LED 闪烁，代表活着。

Comm：通讯时这个 LED 闪烁

四、寄存器列表

注：带*号的数值为出厂值。

<table>
<tr><th rowspan="2">地址</th><th rowspan="2">字 节数</th><th colspan="2">数值范围</th><th rowspan="2">描述</th><th rowspan="2">属性</th></tr>
<tr><th>最小值</th><th>最大值</th></tr>
<tr><td>0-3</td><td>4</td><td>1</td><td>4294967295</td><td>产品序列号，每个产品唯一。</td><td>只读</td></tr>
<tr><td>4-5</td><td>2</td><td>100</td><td>65535</td><td>固件版本号</td><td>只读</td></tr>
<tr><td>6</td><td>1</td><td>1</td><td>254</td><td>MODBUS 通讯地址，254*为出厂值。</td><td>读写</td></tr>
<tr><td>7</td><td>2</td><td>3303</td><td>3303</td><td>产品型号</td><td>只读</td></tr>
<tr><td>8</td><td>1</td><td>1</td><td>255</td><td>硬件版本号</td><td>只读</td></tr>
<tr><td>9</td><td>2</td><td>2</td><td>1152</td><td>波特率设置寄存器.
<table>
<tr><th>数值</th><th>波特率</th></tr>
<tr><td>12</td><td>1200</td></tr>
<tr><td>24</td><td>2400</td></tr>
<tr><td>48</td><td>4800</td></tr>
<tr><td>96</td><td>9600</td></tr>
<tr><td>192*</td><td>19200</td></tr>
<tr><td>384</td><td>38400</td></tr>
<tr><td>576</td><td>57600</td></tr>
<tr><td>1152</td><td>115200</td></tr>
</table></td><td></td></tr>
<tr><td>10-99</td><td>-</td><td>-</td><td>-</td><td>保留</td><td>-</td></tr>
<tr><td>100</td><td>2</td><td>0</td><td>65535</td><td>集电极输出，0 = 三极管导通，1 = 三极管截至。 第 0 位对应输出 1，第 1 位对应输出 2，以此类推。</td><td>只读</td></tr>
<tr><td>101</td><td>1</td><td>1</td><td>100</td><td>串口通讯模块响应命令间隔，单位 2.5 毫秒，默认 10 毫秒</td><td>读写</td></tr>
</table>

五、MODBUS 通信规约

概述

ModBus 协议是 Modicon 公司于 1978 年发明的一种用于电子控制器进行控制和通讯的通讯协议。通过此协议，控制器相互之间、控制器经由网络（例如以太网）和其它设备之间可以进行通信。它的开放性、可扩充性和标准化使它成为一个通用工业标准。ModBus 有 27 种命令，SHJ-3100 只用了 READ,WRITE 两种 ，物理层为 RS485 或 RS232, 串口数据格式为一个起始位，8 个数据位，1 个停止位。

ModBus 标准数据格式为：

字节 1：从节点地址，地址范围为 1-254，255 为广播地址
字节 2：命令，读或写
字节 3：读或写寄存器起始地址的高字节
字节 4：读或写寄存器起始地址的低字节
字节 5：读或写寄存器数据长度的高字节
字节 6：读或写寄存器数据长度的低字节
字节 7：CRC 高字节
字节 8：CRC 低字节

命令示例：

1、读命令（0x03）

这个命令用来读取多个寄存器的内容，主节点需要指明要操作的从节点的地址，起始寄存器地址和要读取寄存器的个数。如果寄存器内容是整型，则高字节在前，低字节在后。例：读取从节点 18，起始寄存器为 100，读 3 个寄存器，主节点应发送如下数据.

字节 1：从节点地址	0x12
字节 2：读命令字	0x03
字节 3：寄存器起始地址的高字节	0x00
字节 4：寄存器起始地址的低字节	0x64
字节 5：寄存器个数的高字节	0x00
字节 6：寄存器个数的低字节	0x03
字节 7：CRC 校验高字节	0x46
字节 8：CRC 校验低字节	0xb7

从节点在几毫秒内返回如下数据。

字节 1： 从节点地址	0x12
字节 2： 读命令字	0x03
字节 3 ：数据个数（寄存器数*2）	0x06
字节 4： 数据 1 的高字节	0xff
字节 5： 数据 1 的低字节	0xff
字节 6： 数据 2 的高字节	0xff
字节 7： 数据 2 的低字节	0xff
字节 8： 数据 3 的高字节	0xff

字节 9： 数据 3 的低字节 0xff
字节 10：CRC 的高字节 0xXX
字节 11：CRC 的低字节 0xXX

2、写命令（0x06）

这个命令用来向单个寄存器写入数据， 主节点需要指明要操作的从节点的地址，寄存器地址和要写入的数据。 例：写从节点 18，寄存器为 100，数据为 512，主节点应发送如下数据.

字节 1：从节点地址 0x12
字节 2：写命令字 0x06
字节 3：寄存器地址的高字节 0x00
字节 4：寄存器地址的低字节 0x64
字节 5：写入数据的高字节 0x02
字节 6：写入数据的低字节 0x00
字节 7：CRC 校验高字节 0xcb
字节 8：CRC 校验低字节 0xd6

从节点在几毫秒内返回如下数据。

字节 1：从节点地址 0x12
字节 2：写命令字 0x06
字节 3：寄存器地址的高字节 0x00
字节 4：寄存器地址的低字节 0x64
字节 5：写入数据的高字节 0x02
字节 6：写入数据的低字节 0x00
字节 7：CRC 校验高字节 0xcb
字节 8：CRC 校验低字节 0xd6

从节点返回数据和发送数据相同，代表成功收到数据。

CRC 校验

下面表格为 ModBus 的 CRC 校验查找表，为了帮助软件工程师快速完成 CRC 程序编写，我们提供示例程序，有需要请通知我们，我们会把如下代码发给你。

CRC 高字节查找表

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
} ;
```

CRC 低字节查找表

```
static unsigned 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
} ;
```

例：计算存储在*puchMsg 里的 usDataLen 个数据的 CRC.

```
unsigned short CRC16 (unsigned char *puchMsg, unsigned char usDataLen)
{
        unsigned char uchCRCHi = 0xFF ; /* CRC 高字节初始化  */
        unsigned char uchCRCLo = 0xFF ; /* CRC 低字节初始化*/
        unsigned uIndex ;
        while (usDataLen--)
```

```
        {
                uIndex = uchCRCHi ^ *puchMsg++ ; /* calculate the CRC */
                uchCRCHi = uchCRCLo ^ auchCRCHi[uIndex] ;
                uchCRCLo = auchCRCLo[uIndex] ;
        }
        return (uchCRCHi << 8 | uchCRCLo) ;
}
```