S5301

24 路隔离开关量输入

16 路隔离集电极开路输出

使用说明书

上海世杰电子有限责任公司

销售：**michael@shjelectronic.com**

技术支持：**support@shjelectronic.com**

一、概述

S5301 是 24 路干接点或湿节点开关量隔离输入 16 路集电极开路输出模块，输入输出均光耦隔离，可以对单路小于 1000Hz 的信号计数，计数长度为 4 字节，24 路输入每路可以使能或禁能，这样只使能一路时，最大可采集 5000Hz 信号。输出总线为 RS485，标准 Modbus RTU 协议，可以和组态软件和 PLC 通讯。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。外壳有白色，黑色 2 种，并且支持 DIN 导轨安装。

主要特点

- RS485 总线,最多 254 个设备，支持 Modbus RTU
- 可以和组态王软件，PLC 直接通讯
- 光耦隔离数字量输入，可作为计数器，32 位
- 可测频率，频率范围 0 到 1000Hz,分辨率 0.1Hz
- 输入可以为干簧管输出的表，比如水表
- 光耦隔离集电极开路输出，直接驱动继电器
- 大量 FLASH 可以用作存储用户数据，需要用户提出需求
- Led 用于指示系统和通讯状态
- DIN 导轨安装
- 白色，黑色外壳可选

应用:

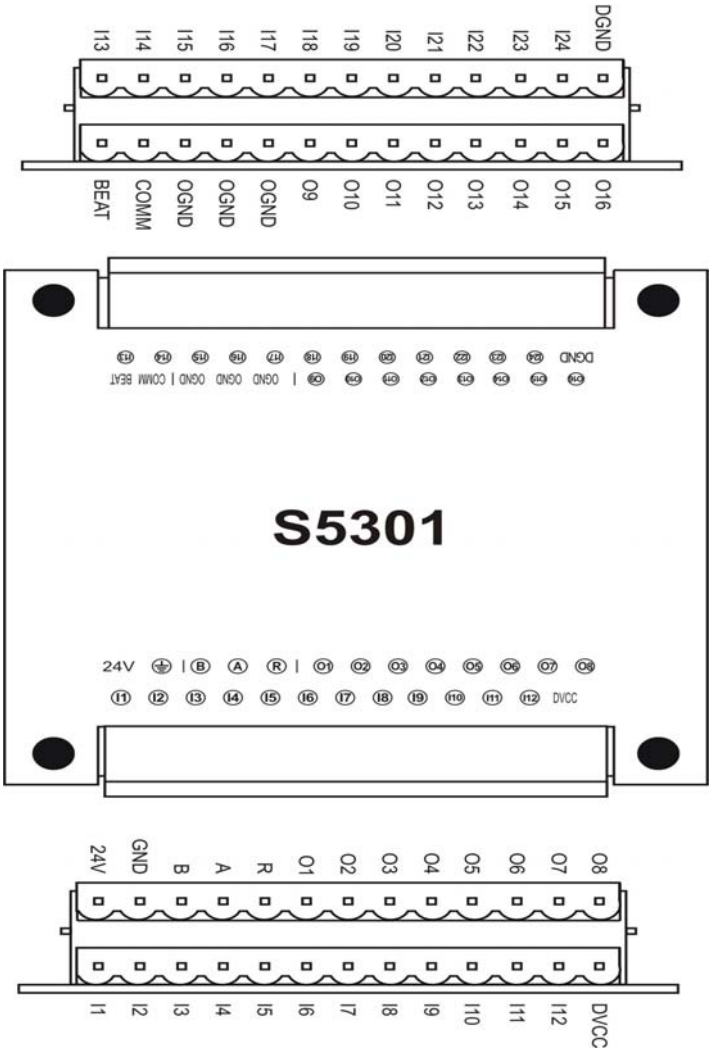
- ✓ 远程数据采集
- ✓ 过程监控
- ✓ 工业过程控制
- ✓ 能源管理
- ✓ 安全系统
- ✓ 工厂自动化
- ✓ 建筑自动化
- ✓ 产品测试
- ✓ 直接数字控制

二、技术参数

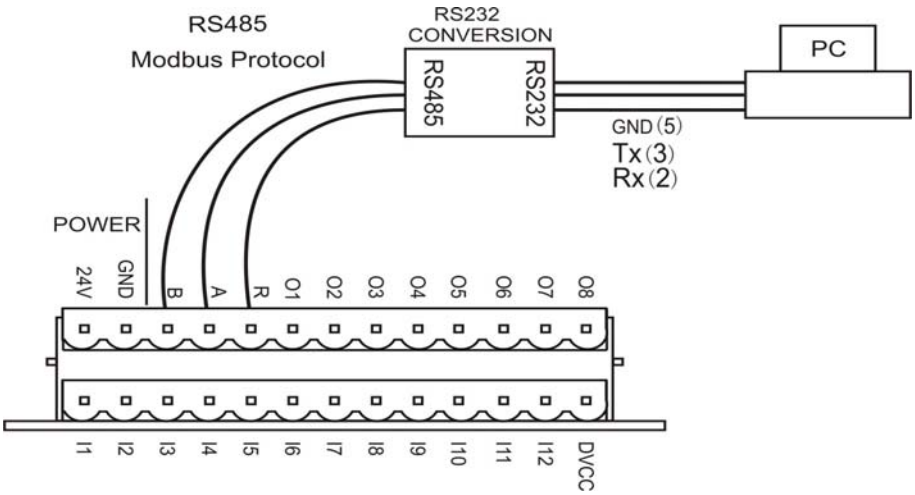
输入通道----------24
输入信号----------+4~+36VDC
输入保护----------防雷，静电
输入类型----------隔离干接点，湿节点共阳极，集电极开路
计数频率----------1000Hz（24 通道）、1000Hz（1 通道）
计数字长----------4 字节
输出通道----------16
输出类型----------隔离集电极开路，可直接驱动继电器
隔离电压---------- > 3000V
输出总线----------光耦隔离 RS485
输出保护----------防雷，静电

电源---------- 9~24V(AC/DC),标准 24VAC
功耗---------- <0.6W
工作温度---------- -20~85℃(-4~185℉)
存储温度---------- -40~125℃(-40~257℉)
相对湿度---------- 5%~95%RH (无凝露)
尺寸----------115*90*43mm

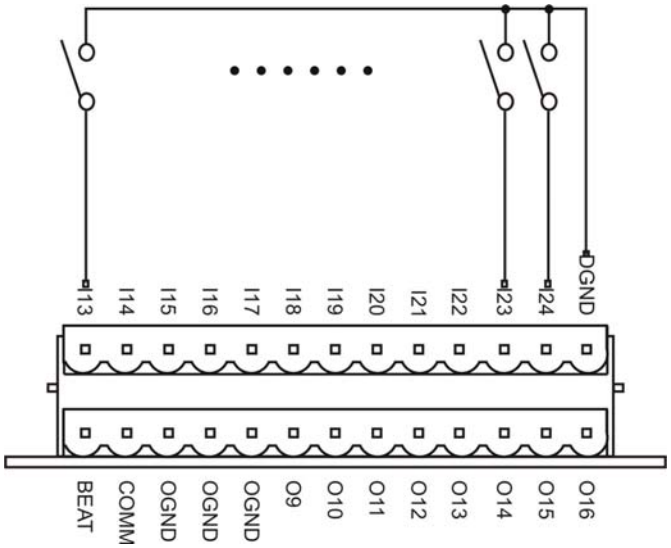
三、**接线说明**



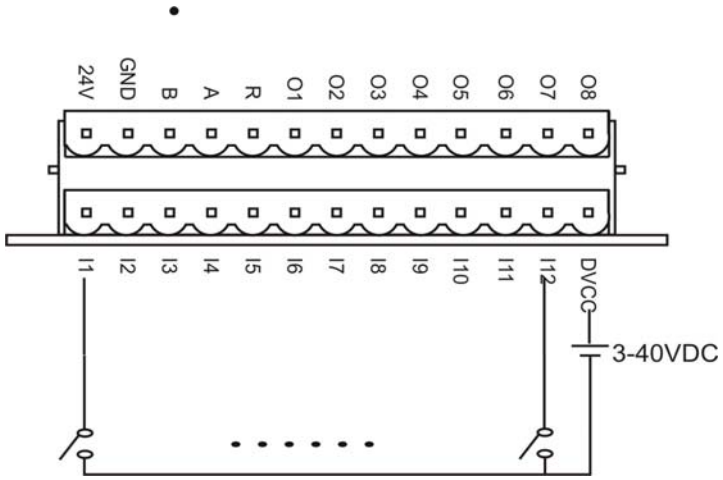
顶视图

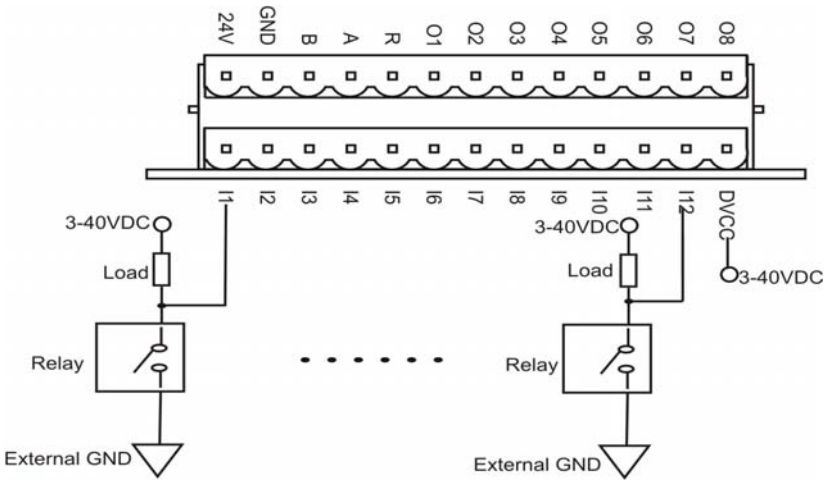


RS485 接线图

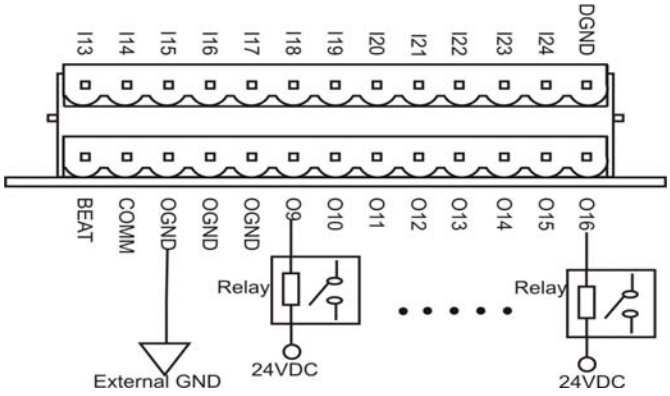


干接点输入





湿接点输入



集电极开路输出，R 为限流电阻，电流应小于 40mA

1、 输入

DVCC：数字量输入 1 到 24 共源电压输入端

I1~I24: 数字量输入通道 1 ~24

DGND:数字量输入通道 1 到 24 地端，只有在干接点输入时使用

2、 输出

O1~O16:集电极开路输出通道 1~16

OVCC:集电极输出共源端

OGND:集电极输出地端

3、 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

4、 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

5、 参数复位

跳线跳在 GND 和 INIT 端，下面这些参数恢复为出厂值。

- 地址：254
- 波特率：19200
- 通道：使能所有通道
- 计数滤波时间：200us

跳线跳在 NULL 端，使用用户配置参数

6、 人机界面

BEAT： 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>5301</td><td>5301</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>开关量输入通道 1 到 16 状态，0 = 触点闭合，1 = 触点断开。第 0 位对应输入 1，第 1 位对应输入 2，以此类推。</td><td>只读</td></tr>
<tr><td>101</td><td>1</td><td>0</td><td>255</td><td>开关量输入通道 17 到 24 状态，0 = 触点闭合，1 = 触点断开。第 0 位对应输入 17，第 1 位对应输入 18，以此类推。</td><td>只读</td></tr>
</table>

102	2	0	65535	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。	读写
103	1	1	100	串口通讯模块响应命令间隔，单位 2.5 毫秒，默认 10 毫秒	读写
104	2	1	30000	计数模式时对输入脉冲滤波时间，单位 10 微妙，默认为 20（200us）	读写
105	2	0	65535	使能/禁能输入通道,0 = 禁能，1 = 使能。Bit 0 对应通道 1，bit2 对应通道 2……	读写
106	1	0	255	使能/禁能输入通道,0 = 禁能，1 = 使能。Bit 0 对应通道 17，bit2 对应通道 18……	读写
107	1	0	1	输入状态选择。0 = ON/OFF, 1 = OFF/ON	读写

待续…

续表：

地址	字节数	数值范围		描述	属性
		最小值	最大值		
108	1	0	1	上升沿计数或者下降沿计数。0 = 上升沿,1 = 下降沿，默认式上升沿	读写
109,111,113….	2	0	65535	开关量输入通道 1~24 计数高字	读写
110,112,114…	2	0	65535	开关量输入通道 1~24 计数低字	读写

02 modbus 命令，读数字输入状态

地址	字节数	数值范围		描述	属性
		最小值	最大值		
0-99	1	0	0	Reserved	读
100-123	1	0	1	输入 1 到 24 状态,1 = 触点打开,0 = 触点闭合	读
124-65535	1	0	0	Reserved	读

01modbus 命令，读线圈状态

地址	字节数	数值范围		描述	属性
		最小值	最大值		
0-123	1	0	0	Reserved	读
124-139	1	0	1	数字输出 1 到 16 状态,1 = 触点打开,0 = 触点闭合	读
140-65535	1	0	0	Reserved	读

15 modbus 命令，多写线圈状态

地址	字节数	数值范围		描述	属性
		最小值	最大值		
0-123	1	0	0	Reserved	读
124-139	1	0	1	写数字输出 1 到 16,1 = 触点打开,0 = 触点闭合	读
140-65535	1	0	0	Reserved	读

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) ;
}
```