

S5138

8 路模拟量输入 6 路数字量输入

16 路集电极输出模块

使用说明书



上海世杰电子有限公司

销售: michael@shjelectronic.com

技术支持: support@shjelectronic.com

一、概述

S5138 具有 8 通道模拟量输入，采用 100k 采样速率的 12 位 AD 转换器，每输入通道有防雷、静电保护，输入信号不但可以为 0-5V,0-10V,4-20mA 常用信号，也可以是 NTC 热敏电阻，集电极输出和干接点，干簧管或霍尔传感器输出的水表的脉冲信号，脉冲计数值存在 FLASH 里。这些信号可以任意组合，同时输入。同时有 6 路光耦隔离数字量输入，每路输入有 32 位长度计数功能，16 路光耦隔离集电极开路输出。输出总线为工业以太网和 RS485，支持 MODBUS TCP 和 MODBUS RTU,输出高速光耦隔离并有防雷、静电保护，有效降低通讯对数据采集的干扰。设计上还通过使用外部看门狗，表面贴装工艺和单点共地技术提高系统稳定性。

主要特点

- 工业以太网，MODBUS TCP 协议
- RS485 总线,最多 254 个设备，支持 Modbus RTU
- 网络和 RS485 都可与组态王软件，PLC 直接通讯
- 8 路模拟量输入可以为 0-5V,0-10V,4-20mA,NTC 10K 热敏电阻，干接点
- 输入可以为干簧管输出的表，比如水表
- 6 路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
- 16 路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，需要用户提出要求
- 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:

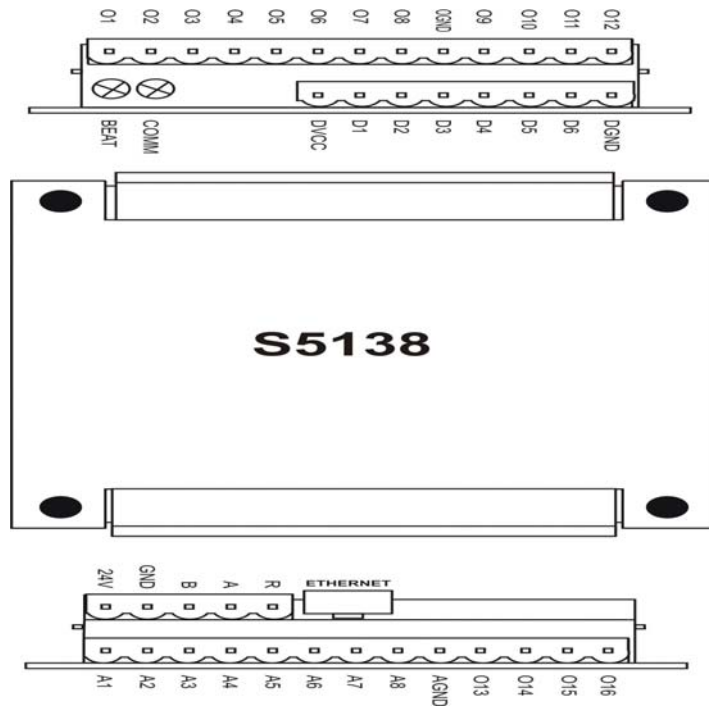
- ✓ 远程数据采集
- ✓ 过程监控
- ✓ 工业过程控制
- ✓ 能源管理
- ✓ 安全系统
- ✓ 工厂自动化
- ✓ 建筑自动化
- ✓ 产品测试
- ✓ 直接数字控制

二、技术参数

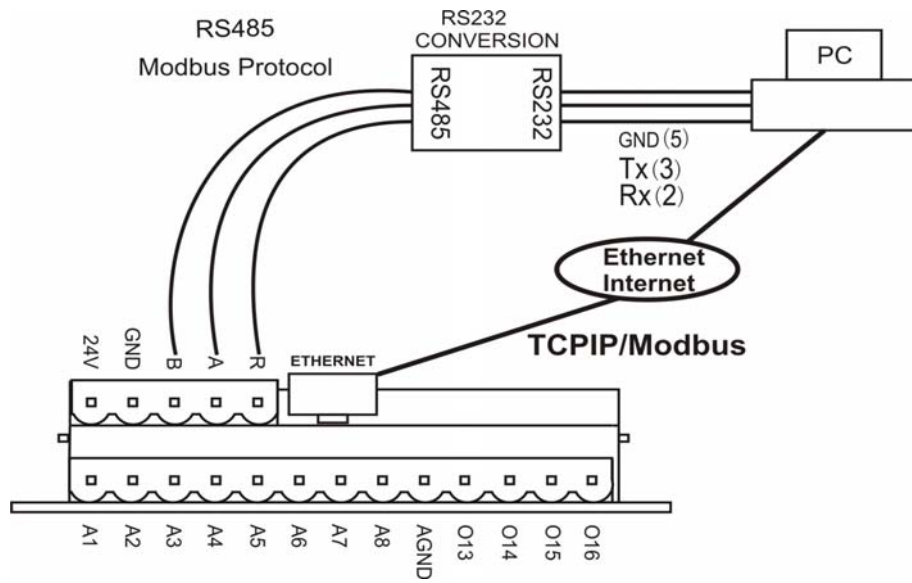
分辨率-----	12 位
输入通道-----	40
输入信号-----	0~5V,0~10V,4~20mA
-----	NTC 10K 热敏电阻，干接点
输入保护-----	防雷，静电
准确度-----	±0.1%
零漂移-----	±3uV/°C

采集速率-----	60 次/秒 (8 通道), 710 次/秒 (1 通道)。【通道数可配置】
输入通道-----	6
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 (6 通道)、1000Hz (1 通道)
计数字长-----	4 字节
数字输出通道-----	16
数字输出信号-----	光耦隔离集电极开路
数字输出电流-----	maximum 40mA
输出总线-----	Ethernet/RS485
RS485 protocol-----	MODBUS/RTU
Ethernet protocol-----	MODBUS/TCPIP
输出保护-----	防雷, 静电
电源-----	12~24V(AC/DC),标准 24VAC
功耗-----	<0.6W
工作温度-----	-20~85℃(-4~185°F)
存储温度-----	-40~125℃(-40~257°F)
相对湿度-----	5%~95%RH (不凝露)
尺寸-----	15*90*43 mm

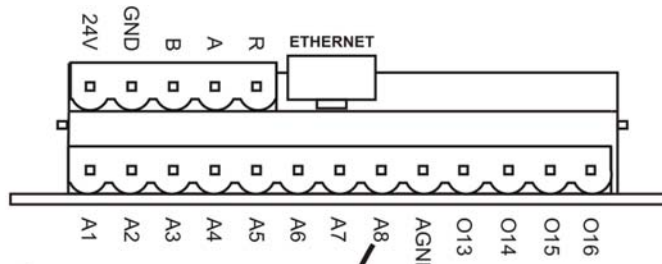
三、接线说明



顶视图



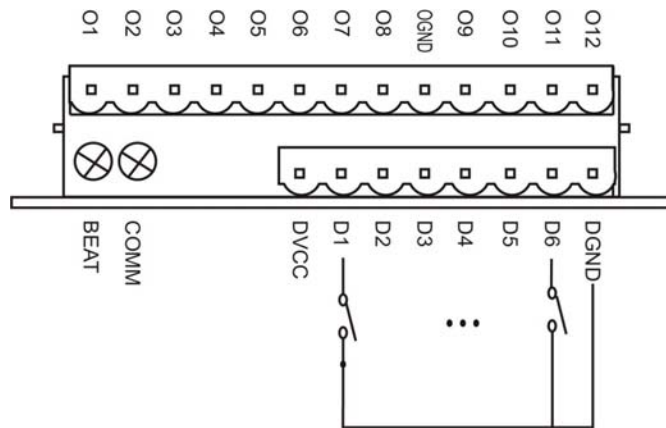
以太网、RS485 接线图



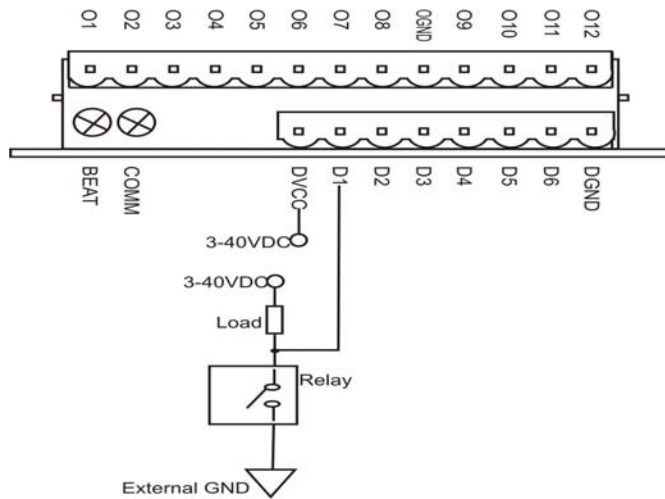
Thermistor
 0-5V
 0-10V
 4-20mA

SENSOR

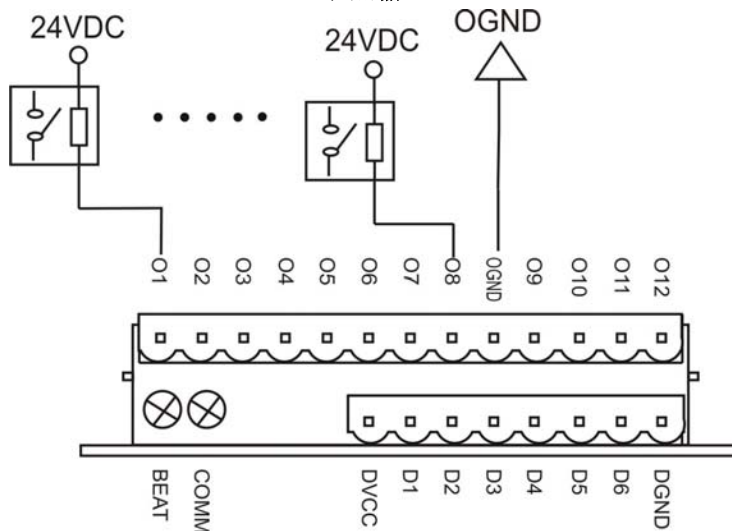
模拟输入



干接点输入



湿节点输入



光耦隔离集电极开路输出

1、模拟输入

A1 ~ A8: 通道 1 到 8 信号输入

AGND:公共端，内部连在一起

2、数字输入

DVCC: 数字输入 1 到 6，湿节点输入时的电源端

D1 ~ D6: 数字输入 1 到 6

DGND: 干接点输入时，输入 1 到 6 的地端

3、数字量输出

O1~O16: :16 路光耦隔离集电极开路输出

4、电源

直流: 12~24V 接正极

GND 接负极

注: 有反接保护

交流: 12~24VAC 不分正负极

5、RS485 输出

A 接 485 总线 A 端

B 接 485 总线 B 端

7、参数复位

跳线跳在 GND 和 INIT 端，下面这些参数恢复为出厂值。

a) 地址: 254

b) 波特率: 19200

c) 通道: 使能所有通道

d) 单位: AD 采集原始数据值

e) 滤波系数: 10

跳线跳在 NULL 端，使用用户配置参数

8、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。

地址	字节数	数值范围		描述	属性
		最小值	最大值		
0-3	4	1	4294967295	产品序列号，每个产品唯一。	只读
4-5	2	100	65535	固件版本号	只读
6	1	1	254	MODBUS 通讯地址，254*为出厂值。	读写
7	2	5138	5138	产品型号	只读
8	1	1	255	硬件版本号	只读

地址	字节数	数值范围		描述	属性	
		最小值	最大值			
9	2	12	1152	波特率设置寄存器.		读写
				数值	波特率	
				12	1200	
				24	2400	
				48	4800	
				96	9600	
				192*	19200	
				384	38400	
				576	57600	
				1152	115200	
10-99	-	-	-	保留		-
100~107	2	0	4095	模拟输入通道 1 到 8 读数,单位由 113~120 决定	读写	
108	1	0	255	输入 1 到 6 的状态,0 = 触点闭合,1 = 触点断开。Bit0 对应输入 1,bit1 对应输入 2 etc.	读写	
109	2	0	65535	16 路集电极开路输出。每一位对应一个输出。BIT0 对应输出 1, BIT1 对应输出 2。。。。	读写	
110	-	-	-	Reserved	-	
111~117	-	-	-	Reserved	-	
118	1	0	255	使能/禁能相应模拟输入通道,最低位对应通道 1,最高位对应通道 8,0 = 禁能,1* = 使能。例:使能通道 1,2,禁能通道 3 到 8,应该写 0x03 到 118 寄存器。	读写	
119~126	1	0	8	通道 1~8 单位设置寄存器。0* = 原始 AD 采样数据,1 = 0-5V(实际值 = 读数 /100,比如读数为 288,则实际值为 2.88V),2 = 0-10V(实际值 = 读数 /100),3 = 4-20mA(实际值 = 读数 /100),4 = 0 - 100%,5 = ON/OFF,6 = OFF/ON,7 = 10K 热敏电阻,摄氏度,(实际值 = 读数 /10),8 = 10K 热敏电阻,华氏度(实际值 = 读数 /10)。	读写	
127-134	1	0	100	通道 1~8 滤波系数,0 为无滤波,10*为出厂值。	读写	
135,137,139 ...	2	0	4095	校准时,通道 1~8 在输入 0V 时读数	读写	

136,138,140 ...	2	0	4095	校准时, 通道 1~8 在输入满量程时读数	读写
151	-	-	-	Reserved	-
152	-	-	-	Reserved	-
153,155...	-	-	-	Reserved	-
154,156...	-	-	-	Reserved	-
167~174	2	0	1000	模拟输入 1 到 8 NTC 热敏电阻偏移值校准	读写
175	2	0	65535	开关量输入通道计数高字	读写
176	2	0	65535	开关量输入通道计数低字	读写
177	2	0	65535	开关量输入通道计数高字	读写
178	2	0	65535	开关量输入通道计数低字	读写
179	2	0	65535	开关量输入通道计数高字	读写
180	2	0	65535	开关量输入通道计数低字	读写
187	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
188	1	0	255	使能/禁能相应数字输入通道, 最低位对应通道 1, 最高位对应通道 6, 0 = 禁能, 1* = 使能。	读写
189	1	0	1	输入状态选择。0 = ON/OFF, 1 = OFF/ON	读写
190	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
191	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
192-199	-	-	-	reserved	-
200-203	1	0	255	设备 IP 地址, 默认是 192.168.0.X	读写
204-207	1	0	255	设备网关, 默认是 192.168.0.1	读写
208-211	1	0	255	子网掩码, 默认是 255.25.255.0	读写
212-217	1	0	255	MAC 地址	读写
218	2	0	65535	端口号, 默认是 502	读写

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, 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, 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) ;
}
```