S5200

16 位 8 路模拟量输入

10 路继电器常开输出

使用说明书

（Rev1.2　　2007.08.28）

上海世杰电子有限公司

销售：**shjelectronic@gmail.com**

技术支持：**shjsupport@gmail.com**

一、概述

S5200 是一款中高档的高速模拟量采集，继电器输出模块，在设计过程中我们不断与客户沟通，采纳了客户的很多宝贵意见，使这款产品具有高品质并且人性化。在这里我感谢那些提出意见的客户，希望新客户也能够反馈我们建议。S5200 具有八通道模拟量输入，10 路继电器常开输出，输入采用 100k 采样速率的 16 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 0-10V,0-5V,4-20mA 常用信号，也可以是 NTC 热敏电阻和干接点，并且这些信号可以任意组合，同时输入；继电器输出可以手动设置闭合，打开和自动，方便现场调试与特殊场合应用；输出总线为 RS485，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。

二、技术参数

分辨率----------16 位

输入通道---------- 8

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/℃

采集速率----------95 次/秒（8 通道），710 次/秒（1 通道）。

【通道数可配置】

继电器输出----------1A@30VDC,0.5A@125VAC

输出----------光隔 RS485

输出保护----------防雷，静电

电源----------12~24V(AC/DC),标准 24AC

【DC 有反接保护】

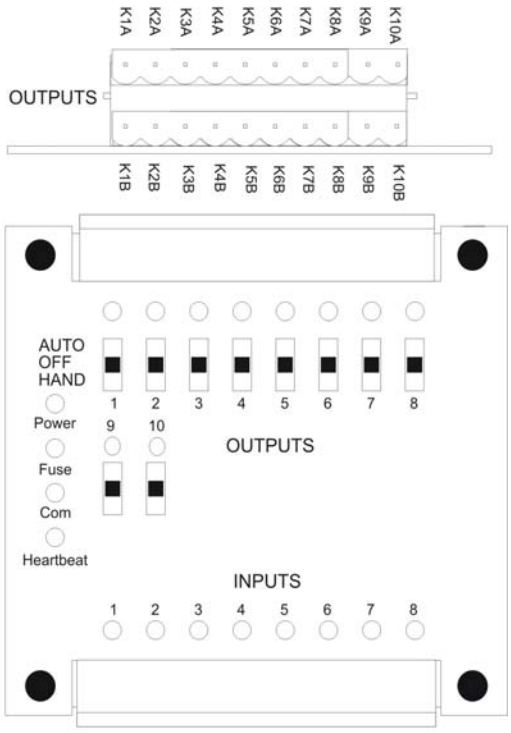
功耗---------- <0.6W

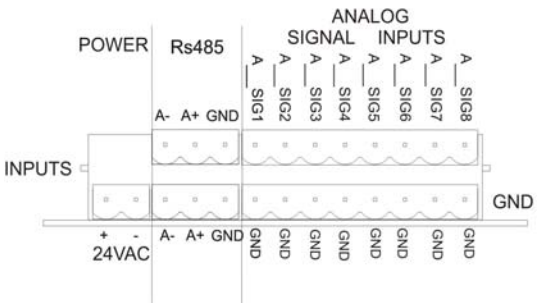
工作温度----------0℃~+70℃

存储温度---------- -20℃~+85℃

相对湿度---------- 5%~95%RH (不凝露)

尺寸----------115*90*40mm





三、接线说明

图 1 为电源输入，串口通讯和信号输入示意图

图 2 为继电器输出示意图

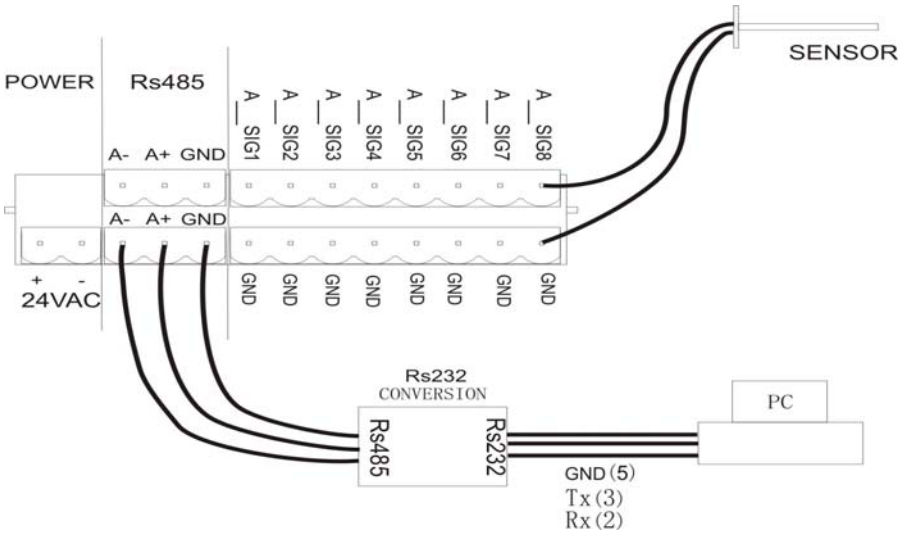


图 1

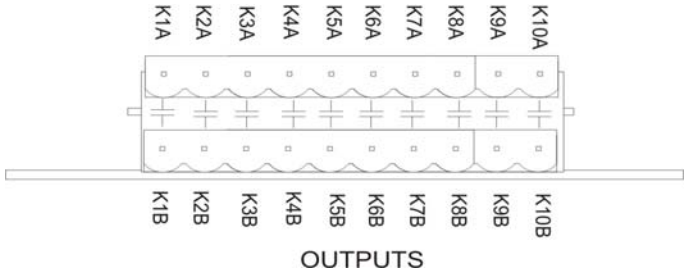


图 2

四、人机界面

1、 输入指示：当输入信号超过满量程一半时，相应通道 LED 点亮，否则熄灭。

2、 输出指示：当继电器闭合时，相应输出 LED 点亮，否则熄灭。

3、 Power： 电源指示灯，当电源正常工作时，LED 点亮，否则熄灭。

4、 Fuse： 电源保险丝指示灯，当正常时，此 LED 熄灭；保险丝烧毁时,此 LED 点亮。

5、 Comm： 串口通讯时此 LED 闪烁，否则熄灭。

6、 Heartbeat: 系统正常工作时，此 LED 闪烁。

五、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	5200	5200	产品型号	只读
8	1	1	255	硬件版本号	只读

待续…

续表：

<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>9</td><td>2</td><td>12</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>读写</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>通道 1 读数，单位由 110 决定</td><td>读写</td></tr>
<tr><td>101</td><td>2</td><td>0</td><td>65535</td><td>通道 2 读数，单位由 111 决定</td><td>读写</td></tr>
<tr><td>102</td><td>2</td><td>0</td><td>65535</td><td>通道 3 读数，单位由 112 决定</td><td>读写</td></tr>
<tr><td>103</td><td>2</td><td>0</td><td>65535</td><td>通道 4 读数，单位由 113 决定</td><td>读写</td></tr>
<tr><td>104</td><td>2</td><td>0</td><td>65535</td><td>通道 5 读数，单位由 114 决定</td><td>读写</td></tr>
<tr><td>105</td><td>2</td><td>0</td><td>65535</td><td>通道 6 读数，单位由 115 决定</td><td>读写</td></tr>
<tr><td>106</td><td>2</td><td>0</td><td>65535</td><td>通道 7 读数，单位由 116 决定</td><td>读写</td></tr>
<tr><td>107</td><td>2</td><td>0</td><td>65535</td><td>通道 8 读数，单位由 117 决定</td><td>读写</td></tr>
<tr><td>108</td><td>2</td><td>0</td><td>65535</td><td>输出控制寄存器，0* = 触点断开，1 = 触点闭合。3 端开关应拨到“AUTO”，Bit0 对应出 1，Bit1 对应输出 2，以此类推。</td><td>读写</td></tr>
<tr><td>109</td><td>1</td><td>1</td><td>255</td><td>使能/禁能相应通道，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 109 寄存器。</td><td>读写</td></tr>
<tr><td>110</td><td>1</td><td>0</td><td>8</td><td>通道 1 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)。</td><td>读写</td></tr>
<tr><td>111</td><td>1</td><td>0</td><td>8</td><td>通道 2 单位设置寄存器。设置参数与 110 寄存器相同。</td><td>读写</td></tr>
</table>

待续…

续表：

地址	字节数	数值范围		描述	属性
		最小值	最大值		
112	1	0	8	通道 3 单位设置寄存器。设置参数与 110 寄存器相同。	读写
113	1	0	8	通道 4 单位设置寄存器。设置参数与 110 寄存器相同。	读写
114	1	0	8	通道 5 单位设置寄存器。设置参数与 110 寄存器相同。。	读写
115	1	0	8	通道 6 单位设置寄存器。设置参数与 110 寄存器相同。	读写
116	1	0	8	通道 7 单位设置寄存器。设置参数与 110 寄存器相同。	读写
117	1	0	8	通道 8 单位设置寄存器。设置参数与 110 寄存器相同。	读写
118	1	0	100	通道 1 滤波系数，0 为无滤波，10*为出厂值。	读写
119	1	0	100	通道 2 滤波系数，0 为无滤波，10*为出厂值。	读写
120	1	0	100	通道 3 滤波系数，0 为无滤波，10*为出厂值。	读写
121	1	0	100	通道 4 滤波系数，0 为无滤波，10*为出厂值。	读写
122	1	0	100	通道 5 滤波系数，0 为无滤波，10*为出厂值。	读写
123	1	0	100	通道 6 滤波系数，0 为无滤波，10*为出厂值。	读写
124	1	0	100	通道 7 滤波系数，0 为无滤波，10*为出厂值。	读写
125	1	0	100	通道 8 滤波系数，0 为无滤波，10*为出厂值。	读写
126	2	0	65535	校准时， 通道 1 在输入 0V 时读数	读写
127	2	0	65535	校准时， 通道 1 在输入满量程时读数	读写
128	2	0	65535	校准时， 通道 2 在输入 0V 时读数	读写
129	2	0	65535	校准时， 通道 2 在输入满量程时读数	读写
130	2	0	65535	校准时， 通道 3 在输入 0V 时读数	读写
131	2	0	65535	校准时， 通道 3 在输入满量程时读数	读写
132	2	0	65535	校准时， 通道 4 在输入 0V 时读数	读写
133	2	0	65535	校准时， 通道 4 在输入满量程时读数	读写
134	2	0	65535	校准时， 通道 5 在输入 0V 时读数	读写
135	2	0	65535	校准时， 通道 5 在输入满量程时读数	读写
136	2	0	65535	校准时， 通道 6 在输入 0V 时读数	读写
137	2	0	65535	校准时， 通道 6 在输入满量程时读数	读写
138	2	0	65535	校准时， 通道 7 在输入 0V 时读数	读写
139	2	0	65535	校准时， 通道 7 在输入满量程时读数	读写

待续…

地址	字节数	数值范围		描述	属性
		最小值	最大值		
140	2	0	65535	校准时， 通道 8 在输入 0V 时读数	读写
141	2	0	65535	校准时， 通道 8 在输入满量程时读数	读写
142	1	2	100	串口响应延时，单位为 2.5 毫秒	读写
143	2	0	65535	3 端开关位置信息，00 = ‘OFF’,01 = ‘HAND’,10 = ‘AUTO’.bit15,bit14 对应输入 1 看关状态，bit1,bit0 对应输入 8 开关状态	只读
144	2	0	65535	3 端开关位置信息，00 = ‘OFF’,01 = ‘HAND’,10 = ‘AUTO’.bit15,bit14 对应输入 9 看关状态，bit13,bit12 对应输入 10 开关状态	只读
145-154	2	0	65535	通道 1 在 100 毫秒内的 10 个原始采样数据	只读
155-164	2	0	65535	通道 2 在 100 毫秒内的 10 个原始采样数据	只读
165-174	2	0	65535	通道 3 在 100 毫秒内的 10 个原始采样数据	只读
175-184	2	0	65535	通道 4 在 100 毫秒内的 10 个原始采样数据	只读
185-194	2	0	65535	通道 5 在 100 毫秒内的 10 个原始采样数据	只读
195-204	2	0	65535	通道 6 在 100 毫秒内的 10 个原始采样数据	只读
205-214	2	0	65535	通道 7 在 100 毫秒内的 10 个原始采样数据	只读
215-224	2	0	65535	通道 8 在 100 毫秒内的 10 个原始采样数据	只读
225	1	0	1	0 = 禁能拨段开关，1 = 使能拨段开关，默认为使能。	读写

六、输入

每个输入通道可以通过模块内部跳线设置为：

- 0 – 5V（不需跳线）
- 0 – 10V（跳线跳到 0 –10V）
- 4- 20 mA（跳线跳到 4 –20mA）
- 干节点，10K 热敏电阻（跳线跳到 0 –5V）

每个输入通道有 LED 指示，当输入信号超过满量程一半时，相应通道 LED 点亮，否则熄灭。

七、输出

每路输出状态由其对应的开关位置决定，开关有 3 个状态，HAND,OFF,AUTO。开关拨到 HAND 时，继电器触点吸合；拨到 OFF 时，继电器触点断开；当拨到 AUTO 时，继电器闭合或断开由相应输出寄存器决定，寄存器可通过串口读写。写入 0 到寄存器继电器断开，写入 1 继电器吸合，寄存器值存储在 RAM 里，断电后数据丢失，重启后默认为继电器断开。每个输出通道有 LED 指示，当继电器闭合时，相应输出 LED 点亮，否则熄灭。

八、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) ;
}
```