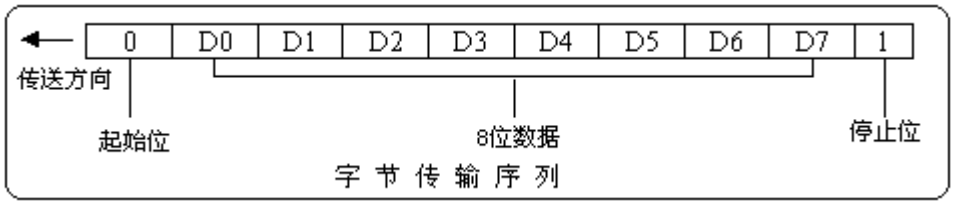
概述：本规约采用 **Modbus** 规约 **RTU** 模式，可以方便地与多种组态软件相连接，其通讯驱动与 Modicon Modbus_RTU 格式完全兼容。

1、 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。

2、 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	1	
整形数据	1	2	一次送回，高字节在前，低字节在后
长整形数	2	4	分两个字回送，高字在前，低字在后
浮点数据			

3、 帧格式：

3.1 **读取仪表寄存器内容**（功能码 03H）

3.1.1 **上位机发送的帧格式：**

顺序	代　码	示例	说　　明
1	仪表地址	1	仪表的通讯地址（1-255 之间）
2	03H	03H	功能码
3	起始寄存器地址高字节	10H	寄存器起始地址
4	起始寄存器地址低字节	00H	
5	寄存器个数高字节	00H	寄存器个数
6	寄存器个数低字节	02H	
7	CRC16 校验高字节	C0H	CRC 校验数据
8	CRC16 校验低字节	CBH	

3.1.2 **仪表回送的帧格式**（数据正常）

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	
……	……	
	第 N 个寄存器数据	
M+4	CRC 校验高字节	
M+5	CRC 校验低字节	

3.1.3 如果起始寄存器地址或寄存器个数错误，仪表回送：

顺序	代　码	示 例	说　　明
1	仪表地址	1	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	

3.2 **设置仪表寄存器内容**（功能码 16H 或 10H 或 06H）

3.2.1.1 功能码 06H 写单路，将一个字（2 字节）数据写入仪表寄存器中，上位机发送的帧格式：

顺序	代　码	示 例	说　　明
1	仪表地址	1	仪表的通讯地址（1-255 之间）
2	06H	06H	功能码
3	寄存器地址高字节	10H	寄存器地址 1000H
4	寄存器地址低字节	00H	
5	写入数据高字节	00H	写入数据 0CH
6	写入数据低字节	0CH	
7	CRC 校验高字节	8DH	CRC 校验数据 8D0FH
8	CRC 校验低字节	0FH	

3.2.1.2 仪表回送：**如果写入正确，则仪表回送相同的数据**。

3.2.2 .1 功能码 16H 或 10H 写多路寄存器，**上位机发送的帧格式**

顺序	代　码	示 例	说　　明
1	仪表地址	1	仪表的通讯地址（1-255 之间）
2	16H 或 10H	10H	功能码
3	寄存器起始地址高字节	1FH	寄存器地址 1F02H
4	寄存器起始地址低字节	02H	
5	寄存器个数高字节	00H	00H
6	寄存器个数低字节	02H	字节数据、整形数据：01H 浮点数据、长整形数：02H
7	字节数（M）	4	字节数据　　：01H 整形数据　　：02H 浮点数、长整形数：04H
8	数据低字节	00H	设置的浮点数据为 100
	数据次低字节	00H	
	数据高字节	42H	
	数据次高字节	C8H	
M+8	CRC 校验高字节	6BH	CRC 校验数据 6BC0H
M+9	CRC 校验低字节	C0H	

3.2.2.2 **仪表回送：**(写入成功)

顺序	代　码	示 例	说　　明
1	仪表地址	1	仪表的通讯地址（1-255 之间）
2	16H　或 10H	10H	功能码
3	起始地址高字节	1FH	寄存器起始地址 1F02H
4	起始地址低字节	02H	
5	寄存器个数高字节	00H	寄存器个数 2
6	寄存器个数低字节	02H	
7	CRC 校验高字节	E7H	CRC 校验数据 E7DCH
8	CRC 校验低字节	DCH	

3.2.3 **仪表回送：**(地址或数据错误)

顺序	代　码	说　　明
1	仪表地址	仪表的通讯地址（1-255 之间）
2	96H 或 90H 或 86H	功能码——针对 16H，10H，06H
3	03H	错误代码
4	CRC 校验高字节	
5	CRC 校验低字节	

注：以上介绍中 CRC 校验为 16 位，高字节在前，低字节在后。

4、 通讯波特率：通讯波特率可以在 300、600、1200、2400、4800、9600 之间选择。出厂时，仪表已设置某一波特率。

5、 仪表地址：仪表地址可以在 1-247 之间选择。仪表出厂时，已设置某一地址。

6、 通讯功能码：03H(召测数据)　16H（10H 或 06H）（数据设置）

7、 通讯数据 CRC 校验：

7.1 校验多项式：$X^{16}+X^{12}+X^{5}+1$

7.2 CRC 检验码的计算例程见附录 B 和。附录 C

7.3 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。

附录 A1：IEEE754 单精度浮点格式：

IEEE 单精度格式由三个字段组成：23 位小数 f； 8 位偏置指数 e ；以及 1 位符号 s。这些字段连续存储在一个 32 位字中（如附录 A 表 1 所示）。0:22 位包含 23 位小数 f，其中第 0 位是小数的最低有效位，第 22 位是最高有效位； 23:30 位包含 8 位偏置指数 e，第 23 位是偏置指数的最低有效位，第 30 位是最高有效位；最高的第 31 位包含符号位 s。

格式	**S**	**e[30:23]**	**f[22:0]**
位	**31**	**30－23**	**22－0**

附录 A 表 1：单精度浮点数据存储格式

附录 A 表 2 显示一侧的三个组成字段 s、e 和 f 的值与另一侧的单精度格式位模式表示的值之间的对应关系；u 意味着无关，即指示字段的值与确定特定单精度格式位模式的值无关。

单精度格式位模式	值
0 < e < 255	$(-1)^S \times 2^{e-127} \times 1.f$ （正规数）
e = 0;f ≠ 0（f 中至少有一位不为零）	$(-1)^S \times 2^{-126} \times 0.f$ （次正规数）
e = 0; f = 0（f 中的所有位均为零）	$(-1)^S \times 2^{-126} \times 0.0$（有符号的零）
s = 0; e = 255; f = 0（f 中的所有位均为零）	+INF （正无穷大）
s = 1; e = 255; f = 0（f 中的所有位均为零）	－INF （负无穷大）
s = u; e = 255;f ≠ 0（f 中至少有一位不为零）	NaN （非数）

附录 A 表 2：单精度格式位模式表示的值

注意，当 e < 255 时，为单精度格式位模式分配的值是使用以下方法构成的：将二进制基数点插入到紧邻小数最高有效位的左侧，将一个隐含位插入到紧邻二进制点的左侧，因而以二进制位置表示法来表示一个带分数 （整数加小数，其中 0 ≤ ò3 小数 < 1）

如此构成的带分数称为单精度格式有效数字。之所以称为隐含位的原因是，在单精度格式位模式中没有显式地指定其值，但偏置指数字段的值隐式指定了该值。

对于单精度格式，正规数和次正规数的差别在于正规数有效数字的前导位 （二进制点左侧的位）为 1，而次正规数有效数字的前导位为 0。在 IEEE 754 标准中，单精度格式次正规数称为单精度格式非规格化数。

在单精度格式正规数中 23 位小数加上隐含前导有效数位提供了 24 位精度。

附录 A 表 3 中给出了重要的单精度存储格式位模式的示例。最大正正规数是以 IEEE 单精度格式表示的最大有限数。最小正次正规数是以 IEEE 单精度格式表示的最小正数。最小正正规数通常称为下溢阈值。（最大和最小正规数和次正规数的十进制值是近似的；对于所示的数字来说，它们是正确的。）

通用名称	位模式（十六进制）	十进制值
+0	**0 0 0 0 0 0 0 0**	**0.0**
-0	**8 0 0 0 0 0 0 0**	**-0.0**
1	**3 f 8 0 0 0 0 0**	**1.0**
2	**4 0 0 0 0 0 0 0**	**2.0**
最大正规数	**7 f 7 f f f f f**	**3.40282347e+38**
最小正正规数	**0 0 8 0 0 0 0 0**	**1.17549435e-38**
最大次正规数	**0 0 7 f f f f f**	**1.17549421e-38**
最小正次正规数	**0 0 0 0 0 0 0 1**	**1.40129846e-45**
+∞	**7 f 8 0 0 0 0 0**	无穷
-∞	**F f 8 0 0 0 0 0**	负无穷
非数	**7 f c 0 0 0 0 0**	**NaN**

附录 A 表 3:单精度存储格式位模式及其 IEEE 值

附录 A2：IEEE754 单精度浮点手工转换样例：

一、**概述：**单精度浮点数据由四个字节组成，我公司的仪表在通讯中，回送这四个字节时分 2 种方式：一种是高字节在前，低字节在后(使用 Modbus 规约的仪表——盘装类仪表)；另一种是低字节在前，高字节在后(使用青智规约的仪表——台式仪表)，下面手工转换，都假设四字节的浮点数据高字节在前，低字节在后。四字节共 32 位

二、**IEEE754 单精度浮点格式及计算**（共四字节 32 位）

1、IEEE754 单精度浮点格式（共四字节 32 位，从高到低）

二进制位	32	31　　24	23　　1
说明	符　号(1 位) 其数值用 S 表示	指　数（8 位） 其数值用 E 表示	尾　数（23 位） 其数值用 F 表示

2、格式说明：

A、第 32 bit 为符号位，为 0 则表示正数，反之为负数，其读数值用 S 表示；

B、第 31～24 bit 共 8 位为幂数(2 的幂数)，其读数值用 E 表示；

C、第 23～1 bit 共 23 位作为系数，视为二进制纯小数，假定该小数的十进制值为 F

D、转换后的十进制浮点数据以 FData 表示；

3、转为为十进制浮点数据公式为：

$$FData=(-1)^{S} * (1 + F) * 2^{(E - 127)}$$

三、**浮点数据实例：**（高字节在前）

十进制	十六进制	二进制数据
220.5	43-5C-80-00	0100 0011 0101 1100 1000 0000 0000 0000
380.6	43-BE-4C-CD	0100 0011 1011 1110 0100 1100 1100 1101
50.25	42-49-00-00	0100 0010 0100 1001 0000 0000 0000 0000
0.999	3F-7F-BE-77	0011 1111 0111 1111 1011 1110 0111 0111
1.0	3F-80-00-00	0011 1111 1000 0000 0000 0000 0000 0000

四、手工转换实例：

1、**220.5=43-5C-80-00 进行转换**

A、220.5=43-5C-80-00 用二进制表示如下：

0	1 0 0 0 0 1 1 0	1 0 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
32	31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
		1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
S	幂数部分 E	纯小数部分 F

B 说明：

a、符号位：为 0，即 S=0

b、指数部分：E = 86H = 134

c、纯小数部分：$F=\frac{1}{2^1}+\frac{1}{2^3}+\frac{1}{2^4}+\frac{1}{2^5}+\frac{1}{2^8}$=0.5+0.125+0.0625+0.03125+0.00390625

=0.72265625 注：只有小数部分的 1，3，4，5，8 位为 1，其他为 0

d、转换后的数据 Fdata=$(-1)^S*(1+f)*2^{(E-127)}=(-1)^0*(1+0.72265625)*2^{(134-127)}=$ $1.72265625*2^7=1.72265625*128=220.5$

2、**50.25＝42-49-00-00 进行转换**

A、50.25＝42-49-00-00 用二进制表示如下：

0	1 0 0 0 0 1 0 0	1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
32	31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
		1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
S	幂数部分 E	纯小数部分 F

B 说明：

a、号位：为 0，即 S=0

b、指数部分：E = 84H = 132

c、纯小数部分：F＝$\frac{1}{2^1}+\frac{1}{2^4}+\frac{1}{2^7}$=0.5+0.0625+0.0078125=0.5703125

d、转换后的数据 Fdata=$(-1)^S*(1+f)*2^{(E-127)}=(-1)^0*(1+0.5703125)*2^{(132-127)}=$ $1.5703125*2^5=1.72265625*32=50.25$

3、**0.999＝3F-7F-BE-77 进行转换**

A、0.999＝3F-7F-BE-77 用二进制表示如下：

0	0 1 1 1 1 1 1 0	1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 0 1 1 1 0 1 1 1
32	31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
		1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
S	幂数部分 E	纯小数部分 F

B 说明：

a、位：为 0，即 S=0

b、指数部分：E = 7EH = 126

c、纯小数部分：

$$F=\frac{1}{2^1}+\frac{1}{2^2}+\frac{1}{2^3}+\frac{1}{2^4}+\frac{1}{2^5}+\frac{1}{2^6}+\frac{1}{2^7}+\frac{1}{2^8}+\frac{1}{2^{10}}+\frac{1}{2^{11}}+\frac{1}{2^{12}}+\frac{1}{2^{13}}+\frac{1}{2^{14}}+\frac{1}{2^{17}}+\frac{1}{2^{18}}+\frac{1}{2^{19}}+\frac{1}{2^{21}}+\frac{1}{2^{22}}++\frac{1}{2^{23}}$$

=0.5+0.25+0.125+0.0625+0.03125+0.015625+0.0078125+0.00390625+0.0009765625+0.00048828125+0.000244140625+0.0001220703125+0.00006103515625+0.00000762939453125+0.000003814697265625+0.000001907348632812 5+0.000000476837158203125+0.000000238418579101562 5+0.000000119209289550781 25=0.998000025749206542968 75

d、转换后的数据

Fdata=$(-1)^S*(1+f)*2^{(E-127)}=(-1)^0*(1+0.99800002574920654296875)*2^{(126-127)}=$ $1.99800002574920654296875/2=0.99900001287460327148437 5\approx0.999000$

注：之所以在最后添加了“≈”符号，因为单精度浮点数据有效位数为 7 位

附录 B：CRC 校验码的计算——计算法

```
#include "stdio.h"
/*==========================
    CrC 计算子程序
==========================*/
unsigned int CrcCal(unsigned int Data, unsigned int GenPoly, unsigned int CrcData)
{
        unsigned int TmpI;
        Data*=2;
        for(TmpI=8;TmpI>0;TmpI--)
        {
                Data=Data/2;
                if((Data ^ CrcData)&1)CrcData=(CrcData/2)^ GenPoly;
                else CrcData/=2;

        }
        return CrcData;
}
/*==================================
            主程序
==================================*/

main()
{
        unsigned int CRC;
        unsigned char tmpi;

        /*==================================
                将 1,3,10H,00H,00,02 进行 CrC 校验
        ==================================*/
        static Buf[]={1,3,0x10,0,0,2};
        CRC=0xffff;
        for(tmpi=0;tmpi<6;tmpi=tmpi+1)CRC=CrcCal(Buf[tmpi],0xa001,CRC);

        printf("CRCDA = %02xH,%02xH",CRC%256,CRC/256);/*CRC 即为计算的结
          果:COH.CBH*/
}
```

附录 C：CRC 校验码的计算——查表法

1、 CRC 校验简单函数(C 语言)

```
/**********************************************
 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, 0x40,0x00, 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 short CRC16(unsigned char *puchMsg, unsigned short usDataLen)
{

 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                QINGDAO QINGZHI INSTRUMENTS Co.Ltd.                ");
 printf("\n
==========================================================================
==========");
 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
}
```

附录 D：数据转换例程

一、概述：本文是以 C51(Franklin C)的格式来给出“**浮点转换为字节数据**”、“**字节转换为浮点数据**”，“**长整形转换为字节数据**”、“**字节数据转换为长整形数据**”等，本文使用的编译器为 Franklin C，使用其他类型的编译器时(例 Keil C)，只须修改浮点数据转换例程中填入字节的顺序即可。同时，本文的例程，同样可以在 Turbo C++上直接运行。产生同样的效果

二、数据转换例程：

```
/*=========================
          公共变量
============================*/
union
   {
     unsigned char uc[4];
     long          lda;
     unsigned long  ul;
     float         fda;
   }un_4b;

   union
   {
     unsigned char uc[2];
     int           ida;
     unsigned int  ui;
   }un_2b;

   long             lda;
   int              ida;
   float            real;
   unsigned char     uca[4];
   unsigned char     ucb[2];
```

1、浮点数据转换为字节数据

```
/*=====================================================
浮点数据转换为字节数据
入口数据：real 放入要转换的浮点数据;
出口数据：转换的四字节数据在 uca[]中
          顺序是从低(uca[0])到高(uca[3])

real=1.0;转换为字节 uca[0]-uca[3]=0,0,0x80,0x3f
=====================================================*/
void  FtoB(void)
{
   un_4b.fda=real;
   uca[0]=un_4b.uc[0];
   uca[1]=un_4b.uc[1];
   uca[2]=un_4b.uc[2];
   uca[3]=un_4b.uc[3];

}
```

2、 **字节数据转换为浮点数据**

```
/*=====================================================
字节数据转换为浮点数据
入口数据：要转换的四字节数据在 uca[]中
          顺序是从低(uca[0])到高(uca[3])
出口数据：real 存放的为已转换的浮点数据;

数据 uca[0]-uca[3]=0,0,0x80,0x3f 转换为浮点 real=1.0
=====================================================*/
void BtoF(void)
{
  un_4b.uc[0]=uca[0];
  un_4b.uc[1]=uca[1];
  un_4b.uc[2]=uca[2];
  un_4b.uc[3]=uca[3];
  real=un_4b.fda;
}
```

3、 **长整形数据转换为字节数据**

```
/*=============================================================
长整形数据转换为字节数据
入口数据：要转换的长整形放在 lda 中
出口数据：转换完的四字节数据在 uca[]中
          顺序是从高(uca[0])到第(uca[3])

长整数据 lda=1000 转换的字节数据 uca[0]-uca[3]=0xe8,0x03,0,0
=============================================================*/
void LtoB(void)
{
  un_4b.lda=lda;
  uca[0]=un_4b.uc[0];
  uca[1]=un_4b.uc[1];
  uca[2]=un_4b.uc[2];
  uca[3]=un_4b.uc[3];

}
```

4、 **字节数据转换为长整形数据**

```
/*=============================================================
字节数据换为长整形数据转
入口数据：转换完的四字节数据在 uca[]中
          顺序是从高(uca[0])到第(uca[3])
出口数据：转换完毕的长整形放在 lda 中

字节数据 uca[0]-uca[3]=0xe8,0x03,0,0   转换的长整形数据 lda=1000
=============================================================*/
void BtoL(void)
{
  un_4b.uc[0]=uca[0];
  un_4b.uc[1]=uca[1];
  un_4b.uc[2]=uca[2];
  un_4b.uc[3]=uca[3];
  lda=un_4b.lda;
}
```

5、 **整形数据转换为字节数据**

```
/*==============================================================
整形数据换为字节数据
入口数据：要转换的整形放在 ida 中
出口数据：转换完的 2 字节数据在 ucb[]中
          顺序是从高(ucb[0])到第(ucb[1])


要转换的整形数据 ida=1000,转换的字节数据 ucb[0]-ucb[1]=0xe8,0x03
==============================================================*/
void ItoB(void)
{
  un_2b.ida=ida;
  ucb[0]=un_2b.uc[0];
  ucb[1]=un_2b.uc[1];

}
```

6、 **字节数据转换为整形数据**

```
/*==============================================================
字节数据转换为整形数据
入口数据：要转换的 2 字节数据在 ucb[]中
          顺序是从高(ucb[0])到第(ucb[1])
出口数据：转换完毕的整形放在 ida 中

字节数据 ucb[0]-ucb[1]=0xe8,0x03   转换的整形数据 ida=1000
==============================================================*/
void BtoI(void)
{
  un_2b.uc[0]=ucb[0];
  un_2b.uc[1]=ucb[1];

  ida=un_2b.ida;
}
```

附录 E：通讯连接线制作

一、我公司仪表通讯串口定义：

通讯方式	接口形式	定　　义	说　　明
RS-232	DB9 针	2－RXD，3－TXD，5－GND	采用 3 线连接方式
	DB25 针	2－TXD，3－RXD，7－GND	
RS-4 8 5	DB9 针	1－A，4－B	

二、RS-232 通讯连接线制作：

1、DB9 座对 DB9 座

仪表串口 DB9	计算机 DB9 串口
2	3
3	2
5	5

2、RS-232 方式通讯时，仪表与计算机的通讯连接线长度应小于 15m

三、多台 RS-485 接口仪表连接方式：当多台带 RS-485 接口仪表连接在 1 条总线上时，应按下图方式进行连接：