# CRC 循环冗余校验算法

**(注：本文档只为提供协议CRC计算过程的参考说明与案例参考代码，由于协议的CRC计算为通用的计算方法，具体的计算过程本公司不负责技术指导，可参阅更多的modbus-rtu协议的CRC16计算文献)**

循环冗余校验CRC区为2字节，含一个16位二进制数据。由发送设备计算CRC值，并把计算值附在信息中，接收设备在接收信息时，重新计算CRC值，并把计算值与接收的在CRC区中实际值进行比较，若两者不相同，则产生一个错误。

CRC开始时先把寄存器的16位全部置成“1”,然后把相邻2个8位字节的数据放入当前寄存器中，只有每个字符的8位数据用作产生CRC，起始位，停止位和奇偶校验位不加到CRC中。

产生CRC期间，每8位数据与寄存器中值进行异或运算，其结果向右移一位(向LSB方向)，并用“0”填入MSB，检测LSB，若LSB为“1”则与预置的固定值异或，若LSB为“0”则不作异或运算。

重复上述处过程，直至移位8次，完成第8次移位后，下一个8位数据，与该寄存器的当前值异或，在所有信息处理完后，寄存器中的最终值为CRC值。
产生CRC的过程：

1. 把16位CRC寄存器置成FFFFH.
2. 第一个8位数据与CRC寄存器低8位进行异或运算，把结果放入CRC寄存器。
3. CRC寄存器向右移一位，MSB填零，检查LSB.
4. (若LSB为0):重复3，再右移一位。
   (若LSB为1):CRC寄存器与A001H(二进制1010 0000 0000 0001) 进行异或运算
5. 重复3和4直至完成8次移位，完成8位字节的处理。
6. 重复2至5步，处理下一个8位数据，直至全部字节处理完毕。
7. CRC寄存器的最终值为CRC值。
8. 把CRC值放入信息时，高8位和低8位应分开放置。

## 把CRC值放入信息中

发送信息中的16 位CRC值时，先送低8位，后送高8位。
若CRC值为1241(0001 0010 0100 0001):

| Addr | Func | Data Count | Data | Data | Data | Data | CR $C_{Lo}$ | CR $C_{Hi}$ |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | 41 | 12 |

**图48：CRC字节顺序**

**例：**

各种可能的CRC值，按两列装入，一列在16 位CRC的高8位区，为(0-256的)CRC值，另一类为低8位区，为CRC的低位值。

用这种方法得到的CRC其执行速度快于计算缓冲器中每个新字符得到一个新CRC值的方法。

※ **注意：**该功能内部交换CRC中的高/低字节，返回的CRC值中，其字节已交换。
因此，由功能码返回的CRC值，能直接放在信息中传送。

# CRC生成

**例：**

**功能取2个自变量：**

unsigned char *puchMsg ; 为生成CRC值，把指针指向含有二进制的数据的缓冲器

unsigned short usDataLen ; 缓冲器中的字节数。

该功能返回CRC作为一种类型“unsigned short”。

**CRC产生的功能**

```
unsigned short CRC16(puchMsg, usDataLen)
unsigned char *puchMsg ; /*按信息的字节数计算CRC */
unsigned short usDataLen ; /* quantity of bytes in message */
{
  unsigned char uchCRCHi = 0xFF ; /* 初始化高字节*/
  unsigned char uchCRCLo = 0xFF ; /* 初始化低字节*/
  unsigned uIndex ; /*把CRC表*/
  while (usDataLen--) /*通过数据缓冲器*/
     {
     uIndex = uchCRCHi ^ *puchMsgg++ ; /*计算CRC */
     uchCRCHi = uchCRCLo ^ auchCRCHi[uIndex] ;
     uchCRCLo = auchCRCLo[uIndex] ;
     }
   return (uchCRCHi << 8 | uchCRCLo) ;
}
```

**高位字节表**

```
/* Table of CRC values for high-order byte */
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
} ;
```

## 低位字节表

```
/* Table of CRC values for low-order byte */
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
} ;
```

## 代码参考2：C语言程序

```
uint GetCRC16(uchar *modbusBuf,uchar num)   //modbusBuf为要参与计算的数据数组；num为字节个数
{
      uint data crc_r;
      uint data temp_c;
      uchar data k,i;
      crc_r=0xffff;
      for(k=0;k<num;k++)
      {
            crc_r=crc_r^modbusBuf[k];
            for(i=0;i<8;i++)
            {
                temp_c=crc_r&0x0001;
                crc_r=crc_r>>1;
                if(temp_c==0x0001)crc_r=crc_r^0xa001;
            }
      }
      return  crc_r;
}
```

# 计算过程：

| 地址 | 功能码 | 起始寄存器 | | 寄存器个数 | | CRC校验码 |
|---|---|---|---|---|---|---|
| 64 | 03 | 00 | 0a | 00 | 01 | To be calculated |

| Step | Byte | Bits Shifted | Action | 16-Bit Register | Bit Shifted Out |
|---|---|---|---|---|---|
| 2 | | | Initial Value | 111 1111 1111 1111 | |
| | 1 | | Load First Data Byte | 0000 0000 0110 0100 | |
| 3 | | | Exclusive OR | 1111 1111 1001 1011 | |
| 4 | | 1 | Shift 1 bit to the Right | 0111 1111 1100 1101 | 1 |
| | | | Generator Polynomial | 1010 0000 0000 0001 | |
| 5a | | | Exclusive OR | 1101 1111 1100 1100 | |
| 4 | | 2 | Shift 1 bit to the Right | 0110 1111 1110 0110 | 0 |
| 4 | | 3 | Shift 1 bit to the Right | 0011 0111 1111 0011 | 0 |
| 4 | | 4 | Shift 1 bit to the Right | 0001 1011 1111 1001 | 1 |
| | | | Generator Polynomial | 1010 0000 0000 0001 | |
| 5a | | | Exclusive OR | 1011 1011 1111 1000 | |
| 4 | | 5 | Shift 1 bit to the Right | 0101 1101 1111 1100 | 0 |
| 4 | | 6 | Shift 1 bit to the Right | 0010 1110 1111 1110 | 0 |
| 4 | | 7 | Shift 1 bit to the Right | 0001 0111 0111 1111 | 0 |
| 4 | | 8 | Shift 1 bit to the Right | 0000 1011 1011 1111 | 1 |
| | | | Generator Polynomial | 1010 0000 0000 0001 | |
| 5a | | | Exclusive OR | 1010 1011 1011 1110 | |
| | 2 | | Load 2 nd Data Byte | 0000 0000 0000 0011 | |
| **Step** | **Byte** | **Bits Shifted** | **Action** | **16-Bit Register** | **Bit Shifted Out** |
| | | | Exclusive OR | 1010 1011 1011 1101 | |
| 4 | | 1 | Shift 1 bit to the Right | 0101 0101 1101 1110 | 1 |
| | | | Generator Polynomial | 1010 0000 0000 0001 | |
| 5a | | | Exclusive OR | 1111 0101 1101 1111 | |
| 4 | | 2 | Shift 1 bit to the Right | 0111 1010 1110 1111 | 1 |
| | | | Generator Polynomial | 1010 0000 0000 0001 | |
| 5a | | | Exclusive OR | 1101 1010 1110 1110 | |
| 4 | | 3 | Shift 1 bit to the Right | 0110 1101 0111 0111 | 0 |
| 4 | | 4 | Shift 1 bit to the Right | 0011 0110 1011 1011 | 1 |
| | | | Generator Polynomial | 1010 0000 0000 0001 | |
| 5a | | | Exclusive OR | 1001 0110 1011 1010 | |
| 4 | | 5 | Shift 1 bit to the Right | 0100 1011 0101 1101 | 0 |
| 4 | | 6 | Shift 1 bit to the Right | 0010 0101 1010 1110 | 1 |
| | | | Generator Polynomial | 1010 0000 0000 0001 | |
| 5a | | | Exclusive OR | 1000 0101 1010 1111 | |
| 4 | | 7 | Shift 1 bit to the Right | 0100 0010 1101 0111 | 1 |
| | | | Generator Polynomial | 1010 0000 0000 0001 | |
| 5a | | | Exclusive OR | 1110 0010 1101 0110 | |
| 4 | | 8 | Shift 1 bit to the Right | 0111 0001 0110 1011 | 0 |
| | 3 | | Load 3rd Data Byte | 0000 0000 0000 0000 | |
| 7 | | | Exclusive OR | 0111 0001 0110 1011 | |

| | | | | | |
|---|---|---|---|---|---|
| 4 | | 1 | Shift 1 bit to the Right | 0011 1000 1011 0101 | 1 |
| | | | Generator Polynomial | 1010 0000 0000 0001 | |
| 5a | | | Exclusive OR | 1001 1000 1011 0100 | |
| 4 | | 2 | Shift 1 bit to the Right | 0100 1100 0101 1010 | 0 |
| 4 | | 3 | Shift 1 bit to the Right | 0010 0110 0010 1101 | 0 |
| 4 | | 4 | Shift 1 bit to the Right | 0001 0011 0001 0110 | 1 |
| | | | Generator Polynomial | 1010 0000 0000 0001 | |
| 5a | | | Exclusive OR | 1011 0011 0001 0111 | |
| 4 | | 5 | Shift 1 bit to the Right | 0101 1001 1000 1011 | 1 |
| | | | Generator Polynomial | 1010 0000 0000 0001 | |
| 5a | | | Exclusive OR | 1111 1001 1000 1010 | |
| 4 | | 6 | Shift 1 bit to the Right | 0111 1100 1100 0101 | 0 |
| 4 | | 7 | Shift 1 bit to the Right | 0011 1110 0110 0010 | 1 |
| | | | Generator Polynomial | 1010 0000 0000 0001 | |
| 5a | | | Exclusive OR | 1001 1110 0110 0011 | |
| 4 | | 8 | Shift 1 bit to the Right | 0100 1111 0011 0001 | 1 |
| | | | Generator Polynomial | 1010 0000 0000 0001 | |
| 5a | | | Exclusive OR | 1110 1111 0011 0000 | |
| 7 | 4 | | Load 4th Data Byte | 0000 0000 0000 1010 | |
| | | | Exclusive OR | 1110 1111 0011 1010 | |
| 4 | | 1 | Shift 1 bit to the Right | 0111 0111 1001 1101 | 0 |
| 4 | | 2 | Shift 1 bit to the Right | 0011 1011 1100 1110 | 1 |
| | | | Generator Polynomial | 1010 0000 0000 0001 | |
| 5a | | | Exclusive OR | 1001 1011 1100 1111 | |
| 4 | | 3 | Shift 1 bit to the Right | 0100 1101 1110 0111 | |
| | | | Generator Polynomial | 1010 0000 0000 0001 | 1 |
| 5a | | | Exclusive OR | 1110 1101 1110 0110 | |
| 4 | | 4 | Shift 1 bit to the Right | 0111 0110 1111 0011 | 0 |
| 4 | | 5 | Shift 1 bit to the Right | 0011 1011 0111 1001 | 1 |
| | | | Generator Polynomial | 1010 0000 0000 0001 | |
| 5a | | | Exclusive OR | 1001 1011 0111 1000 | |
| 4 | | 6 | Shift 1 bit to the Right | 0100 1101 1011 1100 | 0 |
| 4 | | 7 | Shift 1 bit to the Right | 0010 0110 1101 1110 | 0 |
| 4 | | 8 | Shift 1 bit to the Right | 0001 0011 0110 1111 | 0 |
| 7 | 5 | | Load 5th Data Byte | 0000 0000 0000 0000 | |
| | | | Exclusive OR | 0001 0011 0110 1111 | |
| 4 | | 1 | Shift 1 bit to the Right | 0000 1001 1011 0111 | 1 |
| | | | Generator Polynomial | 1010 0000 0000 0001 | |
| 5a | | | Exclusive OR | 1010 1001 1011 0110 | |
| 4 | | 2 | Shift 1 bit to the Right | 0101 0100 1101 1011 | 0 |
| 4 | | 3 | Shift 1 bit to the Right | 0010 1010 0110 1101 | 1 |
| | | | Generator Polynomial | 1010 0000 0000 0001 | |
| 5a | | | Exclusive OR | 1000 1010 0110 1100 | |
| 4 | | 4 | Shift 1 bit to the Right | 0100 0101 0011 0110 | 0 |
| **Step** | **Byte** | **Bits Shifted** | **Action** | **16-Bit Register** | **Bit Shifted Out** |
| 4 | | 5 | Shift 1 bit to the Right | 0010 0010 1001 1011 | 0 |
| 4 | | 6 | Shift 1 bit to the Right | 0001 0001 0100 1101 | 1 |

| | | | | | |
|---|---|---|---|---|---|
| | | | Generator Polynomial | 1010 0000 0000 0001 | |
| 5a | | | Exclusive OR | 1011 0001 0100 1100 | |
| 4 | | 7 | Shift 1 bit to the Right | 0101 1000 1010 0110 | 0 |
| 4 | | 8 | Shift 1 bit to the Right | 0010 1100 0101 0011 | 0 |
| 7 | 6 | | Load 6th Data Byte | 0000 0000 0000 0001 | |
| | | | Exclusive OR | 0010 1100 0101 0010 | |
| 4 | | 1 | Shift 1 bit to the Right | 0001 0110 0010 1001 | 0 |
| 4 | | 2 | Shift 1 bit to the Right | 0000 1011 0001 0100 | 1 |
| | | | Generator Polynomial | 1010 0000 0000 0001 | |
| 5a | | | Exclusive OR | 1010 1011 0001 0101 | |
| 4 | | 3 | Shift 1 bit to the Right | 0101 0101 1000 1010 | 1 |
| | | | Generator Polynomial | 1010 0000 0000 0001 | |
| 5a | | | Exclusive OR | 1111 0101 1000 1011 | |
| 4 | | 4 | Shift 1 bit to the Right | 0111 1010 1100 0101 | 1 |
| | | | Generator Polynomial | 1010 0000 0000 0001 | |
| 5a | | | Exclusive OR | 1101 1010 1100 0100 | |
| 4 | | 5 | Shift 1 bit to the Right | 0110 1101 0110 0010 | 0 |
| 4 | | 6 | Shift 1 bit to the Right | 0011 0110 1011 0001 | 0 |
| 4 | | 7 | Shift 1 bit to the Right | 0001 1011 0101 1000 | 1 |
| | | | Generator Polynomial | 1010 0000 0000 0001 | |
| 5a | | | Exclusive OR | 1011 1011 0101 1001 | |
| 4 | | 8 | Shift 1 bit to the Right | 0101 1101 1010 1100 | 1 |
| | | | Generator Polynomial | 1010 0000 0000 0001 | |
| 5a | | | Exclusive OR | 1111 1101 1011 1100 | |
| | | | RESULT | Hex FD Hex AD | |

读1号模块发送的数据命令：

01 03 00 10 00 0E C5 CB

返回的数据说明

01 03 1E 27 15 27 15 27 1E 27 19 27 15 27 15 27 25 00 00 27 16 3E C7 00 96 0E 2E 00 95 F6 D0 30 2B

1E:代表数据长度28个字节；

27 15 27 15 27 1E 27 19 27 15 27 15 27 25 00 00 27 16 3E C7：两个字节为一个参数，先后顺序为A相电压、A相电流、B相电压、B相电流、C相电压、C相电流，有功功率，无功功率，功率因数，频率共10个参数，20个字节；

00 96 0E 2E：四个字节代表有功电度；

00 95 F6 D0：四个字节代表无功电度；

30 2B：代表校验码；