第1章

概　　述

1.1　循环冗余校验简介

在远距离数据通信中，为确保高效而无差错地传输数据，必须对数据检错，即差错控制。有多种检错的方法，如奇偶校验、算术累加和校验及循环冗余校验。而循环冗余校验是其中最好的一种。

循环冗余校验(Cyclic Redundancy Check，CRC)是目前运用非常广泛的一种数据校验方式。其特点是：检错能力极强，开销小，易于用编码器及检测电路实现。从检错能力来看，其不能发现错误的几率在0.0047%以下；从性能和开销上考虑，其远远优于奇偶校验及算术累加和校验等方式。因而，在计算机网络、磁盘存储和数据通信等领域，CRC无处不在。

Modbus通信协议下有两种通信模式：一种是ASCII(美国标准信息交换代码)通信模式，另一种是RTU(远程终端单元)通信模式，后者就采用了CRC校验方法。著名的通信协议X.25的FCS(帧检错序列)采用的是CRC-CCITT；磁盘驱动器的读写

采用的是CRC16；ARJ、LHA等压缩工具软件采用的是CRC32；一些半导体厂商生产的单总线芯片用CRC8校验其芯片固有编号，如芯片DS18B20的8字节的序列号，最后1个字节是前面7个字节的CRC码，这是为了保证序列号的唯一性与正确性；此外通用的图像存储格式GIF、TIFF等都用CRC作为检错手段。

CRC校验的基本思想是利用线性编码理论，在发送端根据要传送的k位二进制码序列，以一定的规则产生一个校验用的r位监督码(即CRC码)，并附在信息后面，构成一个新的二进制码序列数，共$(k+r)$位，最后发送出去。在接收端，则根据信息码和CRC码之间所遵循的规则进行检验，以确定传送中是否出错。

CRC校验可以简单地描述为：例如我们要发送一些数据(信息字段)，为了避免一些干扰并在接收端判断接收的是否是真实的数据，这时就要加上校验数据(即CRC校验码)，以判断接收的数据是否正确。在发送端，根据要传送的k位二进制码序列，以一定的规则(CRC校验有不同的规则，规则一词，在差错控制理论中称为“生成多项式”)产生一个校验用的r位校验码(CRC码)，附在原始信息后边，构成一个新的二进制码序列数，共$k+r$位，然后发送出去。在接收端，根据信息码和CRC码之间所遵循的规则(即与发送时生成CRC校验码相同的规则)进行检验，校验采用计算机的模2除法，即除数和被除数(即生成多项式)做异或运算，进行异或运算时除数和被除数最高位对齐，进行按位异或运算，若最终的数据能被除尽，则传输正确；否则，传输错误。

生成CRC码的多项式$g(x)$又叫生成多项式(Generation Polynomial)，生成多项式有多种(见表1-1)。生成多项式不同，产生的CRC码也不同。这就类似作除法时被除数相同，除数不同，所求得商和余数不同。生成多项式如果是(4+1)位的，则产生4位的CRC4码；如果是(8+1)位的，则产生8位的CRC8码；如果是(12+1)位的，则产生12位的CRC12码；如果是(16+1)位的，则产生16位的CRC16码；如果是(32+1)位的，则产生32位的CRC32码。

表1-1 标准CRC码生成多项式

名　　称	生成多项式	简记式	标准引用
CRC8	$x^8+x^5+x^3+1$	0x129	
CRC8	x^8+x^2+x+1	0x107	
CRC8	$x^8+x^6+x^4+x^3+x^2+x$	0x15E	
CRC12	$x^{12}+x^{11}+x^3+x^2+x+1$	0x180F	
CRC16	$x^{16}+x^{15}+x^2+1$	0x18005	IBM SDLC
CRC16-CCITT	$x^{16}+x^{12}+x^5+1$	0x11021	ISO HDLC, ITU X.25

续表

名　　称	生成多项式	简记式	标 准 引 用
CRC16-REV	$x^{15}+x^{13}+1$	0xA001	
CRC32	$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$	0x04C11DB7	ZIP，RAR，IEEE 802

CRC 校验码的产生的大致步骤是：把要发送数据(通常是由多个字节组成的数组)以二进制的格式排列起来，拿生成多项式所对应的二进制数去做不借位的除法运算(相当于按位异或)，所得的余数就是 CRC 校验码。

从表 1-1 可以看出 CRC 码有 8 位到 32 位的，常用的是 8 位(一个字节)的 CRC8 码、16 位(两个字节)的 CRC16 码和 32 位(四个字节)的 CRC32 码。以下主要介绍 CRC8 码、CRC16 码和 CRC32 码的生成方法。

1.2 循环冗余校验的基本过程

CRC 校验的基本过程：

采用 CRC 校验时，通信的发送方和接收方用同一个生成多项式 $g(x)$，多项式正序反序，初始值也相同，并且 $g(x)$的首位和最后一位的系数必须为 1。

CRC 检验的处理方法是：发送方用发送数据的二进制多项式 $t(x)$除以 $g(x)$，得到余数 $y(x)$作为 CRC 校验码。校验时，以计算的校正结果是否为 0 为依据，判断数据帧是否出错。设生成多项式是 r 阶的(最高位是 $x\hat{}r$)，具体步骤描述如下。

发送方：

(1) 在发送的 m 位数据的二进制多项式 $t(x)$后添加 r 个 0，扩展到 $m+r$ 位，以容纳 r 位的校验码，追加 0 后的二进制多项式为 $T(x)$。

(2) 用 $T(x)$除以生成多项式 $g(x)$，得到 r 位的余数 $y(x)$，它就是 CRC 校验码。

(3) 把 $y(x)$追加到 $t(x)$后面，此时的数据 $s(x)$就是包含了 CRC 校验码的待发送字符串；由于 $s(x)=t(x)y(x)$，因此 $s(x)$肯定能被 $g(x)$除尽。

接收方：

(1) 接收数据 $n(x)$，这个 $n(x)$就是包含了 CRC 校验码的 $m+r$ 位数据；

(2) 计算 $n(x)$除以 $g(x)$，如果余数为 0 则表示传输过程没有错误，否则表示有错误。从 $n(x)$去掉尾部的 r 位数据，得到的就是原始数据。

1.3 使用循环冗余校验码的一个例子

Modbus 通信协议是工业控制领域实现单个电子控制器互联的通信协议，实际上它已成为一种通用的工业标准。硬件采用 RS232 串行口或 RS485 串行口，通信方式为主从式半双工通信，主机呼叫从机地址，从机应答通信。数据帧共 10 位，1 个起始位，8 个数据位，1 个停止位，无校验。波特率：9600 或 19200。图 1-1 是采用 Modbus 通信协议的总站、分站通信系统结构示意图。以下介绍 Modbus 通信协议的通信格式。

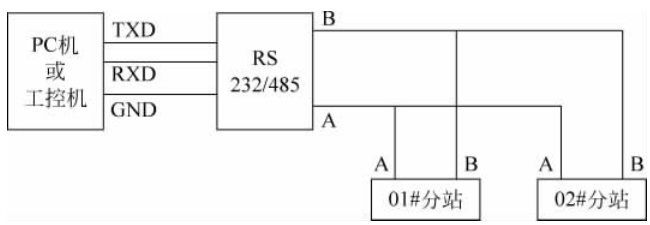


图 1-1 采用 Modbus 通信协议的总站、分站通信系统结构示意图

1.3.1 Modbus 通信协议通信格式说明

1. 功能 03H：读寄存器值

主机发送：

1	2	3	4	5	6	7	8
ADR	03H	起始寄存器高位	起始寄存器低位	寄存器数高位	寄存器数低位	CRC 低位	CRC 高位

第 1 字节 ADR : 从机地址码(= 001～254)
第 2 字节 03H : 读寄存器值(功能码)
第 3、4 字节 : 要读的寄存器开始地址
第 5、6 字节 : 要读的寄存器数量
第 7、8 字节 : 从字节 ADR 起的数据校验和

从机回送：

1	2	3	4、5	6、7		$M-1$、M	$M+1$	$M+2$
ADR	3H	字总数	寄存器数据 1	寄存器数据 2	…	寄存器数据 M	CRC 低位	CRC 高位

第 1 字节　ADR　　：　从机地址码(= 001～254)
第 2 字节　03H　　：　读寄存器值(功能码)
第 3 字节　字总数　　：　从 4～M(包括 4 及 M)的字总数
第 4～M 字节　　：　寄存器数据
第 $M+1$、$M+2$ 字节　　：　从字节 ADR 起的数据校验和

当从机接收错误时，从机回送：

1	2	3	4	5
ADR	83H	错误信息码	CRC 低位	CRC 高位

第 1 字节　ADR　　：　从机地址码(= 001～254)
第 2 字节　83H　　：　读寄存器值出错
第 3 字节　错误信息码　　：　见错误信息码表
第 4、5 字节　　：　从字节 ADR 起的数据校验和

2. 功能 04H：读寄存器值

主机发送：

1	2	3	4	5	6	7	8
ADR	04H	起始寄存器高位	起始寄存器低位	寄存器数高位	寄存器数低位	CRC 低位	CRC 高位

从机回送：

1	2	3	4、5	6、7		$M-1$、M	$M+1$	$M+2$
ADR	4H	字总数	寄存器数据 1	寄存器数据 2	…	寄存器数据 M	CRC 低位	CRC 高位

当从机接收错误时，从机回送：

1	2	3	4	5
ADR	84H	错误信息码	CRC 低位	CRC 高位

第 1 字节　ADR　　：　从机地址码(= 001～254)
第 2 字节　84H　　：　读寄存器值出错
第 3 字节　错误信息码　：　见错误信息码表
第 4、5 字节　　：　从字节 ADR 起的数据校验和

3. 功能 06H：写单个寄存器值

主机发送：

1	2	3	4	5	6	7	8
ADR	06H	起始寄存器高位	起始寄存器低位	数据高字节	数据低字节	CRC 低位	CRC 高位

从机回送：

1	2	3	4	5	6	7	8
ADR	06H	起始寄存器高位	起始寄存器低位	数据高字节	数据低字节	CRC 低位	CRC 高位

当从机接收错误时，从机回送：

1	2	3	4	5
ADR	86H	错误信息码	CRC 低位	CRC 高位

第 1 字节　ADR　　：　从机地址码(= 001～254)
第 2 字节　86H　　：　读寄存器值出错
第 3 字节　错误信息码　：　见错误信息码表
第 4、5 字节　　：　从字节 ADR 起的数据校验和

1.3.2　循环冗余码 CRC16 码的用法

根据上面的 Modbus 通信协议，主机发送 8 个字节，先把前 6 个字节的 CRC 校验码算出，依照低位在前的原则，将其添到后两个字节中，然后发送出去。从机收到 8 个字节后，也是先把前 6 个字节的 CRC 校验码(仍用同一生成多项式)算出，与后两个字节相比，若相同，说明 CRC 校验通过，可执行下一步。若不同，则 CRC 校验未通过，从机应回送错误信息码，要求主机重发。

第2章

CRC8码的计算

2.1 用手工计算 CRC8 码

常用标准 CRC8 码生成多项式如表 2-1 所示。

表 2-1 常用标准 CRC8 码生成多项式

编号	名称	生成多项式	简记式 *	标准引用
1	CRC8	$x^8+x^5+x^4+1$	0x131	
2	CRC8	$x^8+x^5+x^3+1$	0x129	
3	CRC8	x^8+x^2+x+1	0x107	
4	CRC8	$x^8+x^6+x^4+x^3+x^2+x$	0x15E	

在上述 4 个常用标准 CRC8 码生成多项式中，本章着重分析编号为 1 和 3 的两个生成多项式。

CRC8 最终生成的 CRC 校验码为 1 字节，假如其生成多项式 $g(x)=x^8+x^5+x^4+1$，相当于 $g(x)=1\cdot x^8+0\cdot x^7+0\cdot x^6+1\cdot x^5+1\cdot x^4+0\cdot x^3+0\cdot x^2+0\cdot$

$x^1+1\cdot x^0$，即对应的二进制数为 0x131(100110001B)。

CRC8 校验算法：

1. CRC8 正序校验法或常规算法

例题 1* 已知信息码为 0x01，生成多项式 $g(x)=x^8+x^5+x^4+1$，求对应的 CRC8 码。

解：第一步：0x01 右边补 8 个 0，变为 1 0 0 0 0 0 0 0 0，多项式为 1 0 0 1 1 0 0 0 1B=0x131。

第二步：100000000 异或 100110001 得余数 1 1 0 0 0 1，因余数小于生成多项式，那么，余数即为 CRC 码，110001 的十六进制数为 0x31。其计算竖式为

```
   100000000
^  100110001
   ----------
      110001
```

现在我们再用普通的算术除法，计算上面的式子。100000000/100110001 = 0x100/0x131，因为被除数小于除数，商为 0，余数就是被除数 0x100。可见，求 CRC8 码所用的除法(模 2 除法)，并不是普通算术除法，两者的计算结果也大不相同。

例题 2 已知，信息字段代码为 00000001 00000010，对应 $m(x)=x^8+x$；生成多项式 $g(x)=x^8+x^5+x^4+1$，对应 $g(x)$的二进制代码为 100110001。求包括对应的 CRC8 码在内的待发送的数据。

解：现在我们将要对 2 字节数据 0x01，0x02 生成 CRC8 校验码，并最终将生成的 1 字节 CRC 校验码跟在 0x01，0x02 的后面，即 0x01，0x02，＃＃(＃＃即 8 位 CRC 码)，最终生成的 3 字节数据就是待发送的数据。

先计算 $x^8m(x)=x^{16}+x^9$，对应的二进制数为 100000010 00000000。可以看到运算所得结果其实就是将信息字段代码的数左移 8 位。因为最终要将生成的 8 位 CRC8 校验码附在信息字段的后面，所以要将信息字段的数左移 8 位。最后用 $x^8m(x)$得到的二进制数对生成多项式 $g(x)$进行模 2 运算，最终的余数(其二进制数的位数一定比生成多项式 $g(x)$的位数小)就是所要的 CRC8 校验码。

```
  100000010 00000000
^ 100110001
  --------------------
```

---

* 手工计算例题编号为例题 1、2、3，与后面程序计算例题编号例 2.1、2.2 区分。

```
  000110011 00000000
^  100110 001
--------------------
    010101 00100000
^    10011 0001
--------------------
      00110 00110000
^      100 110001
--------------------
        010 11110100
^         10 0110001
--------------------
          00 10010110
```

对 $x^8m(x)$做模 2 运算取余得 10010110(0x96)，这个 8 位的二进制数就是 CRC8 校验码。所以，经 CRC8 校验后待发送的数据就是 0x01，0x02，0x96。

我们再用通常的算术除法，计算上面的式子。100000010 00000000/100110001＝0x10200/0x131＝66048/305＝216，余数为 168＝0xa8。这再次说明，求 CRC8 码所用的除法(模 2 除法)，并不是普通算术除法，两者的计算结果也大不相同。

【说明】 “模 2 除法”与“算术除法”类似，但它既不向上位借位，也不比较除数和被除数的相同位数值的大小，只要以相同位数进行相除即可。模 2 加法运算为：1＋1＝0，0＋1＝1，0＋0＝0，无进位，也无借位；模 2 减法运算为：1－1＝0，0－1＝1，1－0＝1，0－0＝0，也无进位，无借位。相当于二进制中的逻辑异或运算。也就是比较后，两者对应位相同则结果为“0”，不同则结果为“1”。这里举一个“模 2 除法”的例子，如 100101 除以 1110，结果得到商为 11，余数为 1，如图 2-1 所示。此例的算术除法结果为 100101B/1110B＝25H/0EH＝37/14，商为 2，余数为 9。两者显然不同。

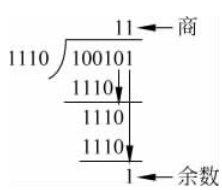


图 2-1　“模 2 除法”的一个例子

2. 逆序(或反序)CRC8 校验法

DS18B20 是一种由达拉斯(DALLAS)公司生产的温度传感器芯片。在 DS18B20 中，有两处用到 CRC8 码，一处是 DS18B20 的 8 字节的序列号，最后一字节是前面 7

个字节的 CRC 码，这是为了保证序列号的唯一性与正确性；另一处是在 DS18B20 内部 9 字节的高速温度存储器，其第 9 字节是前面 8 个字节的 CRC 校验码，这是为了温度数据传输的正确性。而在 DS18B20 中生成 CRC 码所用到的方法不同于常规生成算法，它采用的是逆序 CRC 信息单元编码算法，该 CRC 码是由 DS18B20 中的多项式寄存器通过其中所包含的移位寄存器以及异或门对输入该多项式寄存器的每一位二进制数做一定的运算所得到的。所谓逆序 CRC 算法，就是把原生成多项式的反转作为计算逆序 CRC 码的生成多项式，从低字节低位计算 CRC 码的算法，即从右端到左端计算 CRC 码的算法。

DS18B20 的内部有一个 64 位光刻 ROM 区，是出厂前被光刻好的，它可以看作该 DS18B20 的地址序列号，不同的器件地址序列号不同。64 位闪速 ROM 的结构如下：

8 位产品系列号	48 位产品序号	8 位 CRC 编码

开始 8 位是产品类型的编号，中间的是每个器件的唯一的序号，共有 48 位，最后 8 位是前 56 位的 CRC 校验码，这也是多个 DS18B20 可以采用一线进行通信的原因。

不仅 DS18B20 芯片是这样，由达拉斯(DALLAS)公司生产的 DS 系列芯片，诸如 DS1990A，DS2430，DS2431，DS2432，DS2433 等都是这样。这些芯片数据通信采用 1-Wire 总线协议，仅占用一条信号线和一条地线。为了保证序列号的唯一性与正确性，这些芯片的 8 字节的序列号，最后 1 个字节是前面 7 个字节的 CRC 码。

DS18B20 计算 CRC 码为 1 个字节长，其生成多项式 $g(x)=x^8+x^5+x^4+1$，反转 $g'(x)=1+x^4+x^5+x^8$，略去最右边的“1”后的反转码为 10001100B=0x8c。下面，我们用手工计算生成多项式 $g(x)=x^8+x^5+x^4+1$(反序)时，一个字节的 CRC 码。

例题 3 已知信息码为 0x01；生成多项式 $g'(x)=1+x^4+x^5+x^8$，求对应的 CRC8 码。

解：第一步：把 0x01 写成二进制形式，使 crc＝00000001B，生成多项式为 10001100B=0x8c。

第二步：看 crc=00000001 的末位是否为 1，若为 1，crc 右移 1 位后，与 10001100 异或；否则，crc 右移 1 位。就这样，共进行 8 次。最后变量 crc 中的数即为 CRC 码。其计算竖式为

```
  00000001              ;待求字节
  00000000              ;因待求字节末位为 1,右移 1 位后,变成 0x00
^ 10001100              ;与 0x8c 异或
  ------------------
```

```
  10001100              ;结果为 0x8c
  01000110              ;因末位为 0,右移 1 位后,变成 0x46
  00100011              ;因末位为 0,右移 1 位后,变成 0x23
  00010001              ;因末位为 1,右移 1 位后,变成 0x11
^ 10001100              ;与 0x8c 异或
  ------------------
  10011101              ;结果为 0x9d
  01001110              ;因末位为 1,右移 1 位后,变成 0x4E
^ 10001100              ;与 0x8c 异或
  ------------------
  11000010              ;结果为 0xc2
  01100001              ;因末位为 0,右移 1 位后,变成 0x61
  00110000              ;因末位为 1,右移 1 位后,变成 0x30
^ 10001100              ;与 0x8c 异或
  ------------------
  10111100              ;结果为 0xbc
  01011110              ;因末位为 0,右移 1 位后,变成 0x5e
```

0x5e 就是 01 这个数在生成多项式 $g'(x)=1+x^4+x^5+x^8$ 时的 CRC8 码。当所求的数值不止一个字节(绝大部分情况都是多个字节)时,用手工计算太麻烦,一般要用程序来计算。

当生成多项式 $g(x)=x^8+x^5+x^4+1$,取反序时,采用 C 语言计算一个字节的 CRC 码的实现方式如下:

```
#include <stdio.h>                    //C语言包含文件
unsigned char crc8;
unsigned char cal_table_low_first(unsigned char value)
{
    unsigned char i, crc;
    crc = value;
    for (i = 8; i > 0; -- i)          // 同样需要计算 8 次
    {
        if (crc & 0x01)               // 判断最低位是否为 1
           crc = (crc >> 1) ^ 0x8C;   // 若为 1,先右移 1 位,再和 0x8C 异或
        else
           crc = (crc >> 1);          // 否则,只右移 1 位
    }
    return crc;
}
void main()
{
  crc8 = cal_table_low_first(0x01);
```

```
    printf(" crc8 = %3x \n",crc8);
}
```

此程序是计算 01 这个数的 CRC 码的，程序执行后，所得结果为 crc8＝0x5e。这与上面用手工计算的结果一致。

2.2 用程序计算 CRC8 码：生成多项式 $g(x)=x^8+x^5+x^4+1$（正序）

生成多项式 $g(x)=x^8+x^5+x^4+1$，当正序时 $x^8+x^5+x^4+1=1\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 1B=0x131$。

2.2.1 C 语言

2.2.1.1 用计算法求 CRC8 码

求 CRC8 码共有两种方法，一种是计算法，一种查表法，两种方法各有优缺点。计算法速度较慢，但占用最小的内存空间；查表法运行速度较快，但占用较大的内存。实际应用时，应酌情选择。这里将提供这两种求 CRC8 码的方法的 C 语言程序。

1. 程序清单

```
/*****************************************************************/
* 用计算法求 CRC8 码的 C 语言程序，生成多项式 g(x) = x^8 + x^5 + x^4 + 1(正序)
/*****************************************************************/
#include <stdio.h>                    /*C 语言包含文件*/
#include <math.h>
#define uchar unsigned char
#define uint unsigned int

uchar crc8,crc,mm;
uchar SerialNum[8] = {0x01,0x02,0x03,0x04,0x05,0x06,0x07};
uchar calcrc_1byte(uchar abyte)
{
   uchar i;
   crc = crc ^ abyte;
   for(i = 8; i > 0; --i)
```

```
    {
       if (crc&0x80)
       {
           crc = (crc << 1)^ 0x31;
        }
        else
        crc = (crc << 1);
    }
    return crc;
}
void main( )
{
     uchar i;
     crc = 0;
     for(i = 0;i < 7;i++){
       mm = SerialNum[i];
       crc8 = calcrc_1byte(mm);
     }
     printf(" % 3x \n",crc8);          //以上 7 个数的 CRC 校验为 0xaf
}
```

2. 程序运行

例 2.1　生成多项式 $g(x)=x^8+x^5+x^4+1$(正序),初始值为 0x00,求 01 这个数的 CRC8 码。

解：打开“第 2 章\2.2 用程序计算 CRC8 码—生成多项式 131H\C 语言程序\用计算法求 CRC8 码”文件夹,双击扩展名为.dsw 的文件*,将进入 VC6.0 编译环境,将程序中的“for(i=0;i<7;i++){”改为“for(i=0;i<1;i++){”,编译并执行程序,运行结果如下：

```
crc8 = 31
```

可知,01 这个数的 CRC8 码是 0x31,这与我们在前面用手工计算的 01 这个字节的 CRC8 码是一致的。

例 2.2　生成多项式 $g(x)=x^8+x^5+x^4+1$(正序),初始值为 0x00,求 01,02 这两个数的 CRC8 码。

---

* 当双击扩展名为.dsw 的文件不能进入 VC6.0 编译环境时,可以用右键单击该文件→打开方式→选择 Microsoft(R) Developer Studio,单击“确定”,即可进入 VC6.0 环境。

解：打开“第 2 章\2.2 用程序计算 CRC8 码—生成多项式 131H\C 语言程序\用计算法求 CRC8 码”文件夹，双击扩展名为 DSW 的文件，将进入 VC6.0 编译环境，将程序中的“for(i=0;i<7;i++){”改为“for(i=0;i<2;i++){”，编译并执行程序，运行结果如下：

```
crc8 = 96
```

可知，01，02 这两个数的 CRC8 码是 0x96，这与我们在前面用手工计算的 01，02 这两个字节的 CRC8 码是一致的。

例 2.3 生成多项式 $g(x)=x^8+x^5+x^4+1$(正序)，初始值为 0x00，求 01，02，03，04，05，06，07 这 7 个数的 CRC8 码。

解：打开“第 2 章\2.2 用程序计算 CRC8 码—生成多项式 131H\C 语言程序\用计算法求 CRC8 码”文件夹，双击扩展名为.dsw 的文件，将进入 VC6.0 编译环境，编译并执行程序，运行结果如下：

```
crc8 = af
```

可知，01，02，03，04，05，06，07 这 7 个数的 CRC8 码是 0xaf。

2.2.1.2 用查表法求 CRC8 码

1. 程序清单

```
/*****************************************************************/
*用查表法求 CRC8 码的 C 语言程序,生成多项式 g(x) = x^8 + x^5 + x^4 + 1(正序)
/*****************************************************************/
#include <stdio.h>                    /*C 语言包含文件*/
#include <math.h>
#define uchar unsigned char
#define uint unsigned int
uchar tab[] =
{
    0x00, 0x31, 0x62, 0x53, 0xc4, 0xf5, 0xa6, 0x97, 0xb9, 0x88, 0xdb, 0xea, 0x7d, 0x4c,
0x1f,0x2e,
    0x43, 0x72, 0x21, 0x10, 0x87, 0xb6, 0xe5, 0xd4, 0xfa, 0xcb, 0x98, 0xa9, 0x3e, 0x0f,
0x5c,0x6d,
    0x86, 0xb7, 0xe4, 0xd5, 0x42, 0x73, 0x20, 0x11, 0x3f, 0x0e, 0x5d, 0x6c, 0xfb, 0xca,
0x99,0xa8,
    0xc5, 0xf4, 0xa7, 0x96, 0x01, 0x30, 0x63, 0x52, 0x7c, 0x4d, 0x1e, 0x2f, 0xb8, 0x89,
0xda,0xeb,
    0x3d, 0x0c, 0x5f, 0x6e, 0xf9, 0xc8, 0x9b, 0xaa, 0x84, 0xb5, 0xe6, 0xd7, 0x40, 0x71,
```

```
0x22,0x13,
    0x7e, 0x4f, 0x1c, 0x2d, 0xba, 0x8b, 0xd8, 0xe9, 0xc7, 0xf6, 0xa5, 0x94, 0x03, 0x32,
0x61,0x50,
    0xbb, 0x8a, 0xd9, 0xe8, 0x7f, 0x4e, 0x1d, 0x2c, 0x02, 0x33, 0x60, 0x51, 0xc6, 0xf7,
0xa4,0x95,
    0xf8, 0xc9, 0x9a, 0xab, 0x3c, 0x0d, 0x5e, 0x6f, 0x41, 0x70, 0x23, 0x12, 0x85, 0xb4,
0xe7,0xd6,
    0x7a, 0x4b, 0x18, 0x29, 0xbe, 0x8f, 0xdc, 0xed, 0xc3, 0xf2, 0xa1, 0x90, 0x07, 0x36,
0x65,0x54,
    0x39, 0x08, 0x5b, 0x6a, 0xfd, 0xcc, 0x9f, 0xae, 0x80, 0xb1, 0xe2, 0xd3, 0x44, 0x75,
0x26,0x17,
    0xfc, 0xcd, 0x9e, 0xaf, 0x38, 0x09, 0x5a, 0x6b, 0x45, 0x74, 0x27, 0x16, 0x81, 0xb0,
0xe3,0xd2,
    0xbf, 0x8e, 0xdd, 0xec, 0x7b, 0x4a, 0x19, 0x28, 0x06, 0x37, 0x64, 0x55, 0xc2, 0xf3,
0xa0,0x91,
    0x47, 0x76, 0x25, 0x14, 0x83, 0xb2, 0xe1, 0xd0, 0xfe, 0xcf, 0x9c, 0xad, 0x3a, 0x0b,
0x58,0x69,
    0x04, 0x35, 0x66, 0x57, 0xc0, 0xf1, 0xa2, 0x93, 0xbd, 0x8c, 0xdf, 0xee, 0x79, 0x48,
0x1b,0x2a,
    0xc1, 0xf0, 0xa3, 0x92, 0x05, 0x34, 0x67, 0x56, 0x78, 0x49, 0x1a, 0x2b, 0xbc, 0x8d,
0xde,0xef,
    0x82, 0xb3, 0xe0, 0xd1, 0x46, 0x77, 0x24, 0x15, 0x3b, 0x0a, 0x59, 0x68, 0xff, 0xce,
0x9d,0xac
};
uchar crc(uchar * p,uchar len){
    uchar crc = 0x00;
    uchar i;
    for(i = 0;i < len;i++) crc = tab[crc ^ * p++];
    return crc;
}
int main(void)
{
        uchar crc8;
        uchar buf[ ] = {0x01,0x02,0x03,0x04,0x05,0x06,0x07};
        crc8 = crc(buf,7);
        printf(" crc8 = %x \n",crc8);
}
```

2. 程序运行

例 2.4　生成多项式 $g(x)=x^8+x^5+x^4+1$(正序),初始值为 0x00,求 01,02,03,04,05,06,07 这 7 个数的 CRC8 码。

解：打开“第 2 章\2.2 用程序计算 CRC8 码—生成多项式 131H\C 语言程序\用查表法求 CRC8 码”文件夹,双击扩展名为.dsw 的文件,将进入 VC6.0 编译环境,编译并执行程序,运行结果如下：

```
crc8 = af
```

可知,01,02,03,04,05,06,07 这 7 个数的 CRC8 码是 0xaf。

2.2.2 VB 语言

VB 语言和 C 语言有一个共同点,它们都是高级语言,都是在个人电脑或笔记本电脑上运行的。

2.2.2.1 用计算法求 CRC8 码

1. 程序清单

用计算法求 CRC8 码的 VB 语言程序:

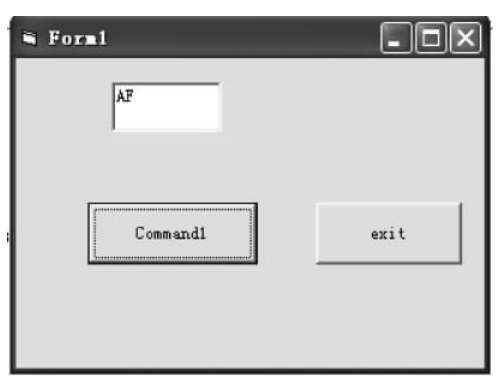
```
Private Sub Command1_Click()
       Dim CRC() As Byte
       Dim bData() As Byte '待传输数据
       ReDim d(6) As Byte
       d(0) = 1 '1
       d(1) = 2 '2
       d(2) = 3 '3
       d(3) = 4 '4
       d(4) = 5 '5
       d(5) = 6 '6
       d(6) = 7 '7
       CRC = CRC8(d)      '调用 CRC8 计算函数
       Text1 = Hex(CRC(0))
End Sub
Private Function CRC8(bData() As Byte) As String
    Dim CRC_Temp As Byte
    Dim CRC_Temp1 As Byte
    Dim i As Integer
    Dim j As Integer
    CRC_Temp = 0
    For j = LBound(bData) To UBound(bData)
     CRC_Temp = CRC_Temp Xor bData(j)
      For i = 8 To 1 Step - 1
        CRC_Temp1 = CRC_Temp And &H80
        If CRC_Temp1 >= &H80 Then
          CRC_Temp = ((CRC_Temp And &H7F) * 2) Xor &H31
```

```
            Else
              CRC_Temp = CRC_Temp * 2
            End If
          Next i
        Next j
        Dim ReturnData(1) As Byte
            ReturnData(0) = CRC_Temp
            CRC8 = ReturnData
End Function
Private Sub Command2_Click()
End
End Sub
```

2. 程序运行

例 2.5　生成多项式 $g(x)=x^8+x^5+x^4+1$(正序),初始值为 0x00,求 01,02,03,04,05,06,07 这 7 个数的 CRC8 码。

解：打开“第 2 章\2.2 用程序计算 CRC8 码—生成多项式 131H\VB 程序\用计算法求 CRC8 码”文件夹,双击扩展名为.vpb 的文件,将进入 VB 编译环境,执行菜单命令“运行”→“启动”,将出现一个对话框,再单击对话框中的 Command1 按钮,将显示出一个字节的 CRC8 码,如下所示：



可知,01,02,03,04,05,06,07 这 7 个数的 CRC8 码是 0AFH。

2.2.2.2　用查表法求 CRC8 码

1. 程序清单

用查表法求 CRC8 码的 VB 语言程序：

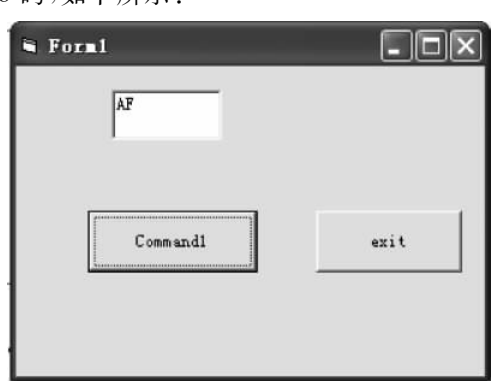
```
Private Sub Command1_Click()
      Dim CRC() As Byte
      Dim bData() As Byte '待传输数据
      ReDim d(6) As Byte
      d(0) = 1 '1
      d(1) = 2 '2
      d(2) = 3 '3
      d(3) = 4 '4
      d(4) = 5 '5
      d(5) = 6 '6
      d(6) = 7 '7
      CRC = CRC8(d)      '调用 CRC8 计算函数
      Text1 = Hex(CRC(0))
End Sub
Private Function CRC8(bData() As Byte) As String
    Dim CRC_Temp As Byte
    Dim CRC_Index As Byte
    Dim i As Long
    CRC_Temp = 0
    For i = LBound(bData) To UBound(bData)
        CRC_Index = CRC_Temp Xor bData(i)
        CRC_Temp = GetCRC8(CRC_Index)
    Next
    Dim ReturnData(1) As Byte
      ReturnData(0) = CRC_Temp
      CRC8 = ReturnData
End Function

Private Function GetCRC8(Index As Byte) As Byte
    GetCRC8 = Choose(Index + 1, _
    &H0, &H31, &H62, &H53, &HC4, &HF5, &HA6, &H97, &HB9, &H88, &HDB, &HEA, &H7D, &H4C,
&H1F, &H2E, _
    &H43, &H72, &H21, &H10, &H87, &HB6, &HE5, &HD4, &HFA, &HCB, &H98, &HA9, &H3E, &HF,
&H5C, &H6D, _
    &H86, &HB7, &HE4, &HD5, &H42, &H73, &H20, &H11, &H3F, &HE, &H5D, &H6C, &HFB, &HCA,
&H99, &HA8, _
    &HC5, &HF4, &HA7, &H96, &H1, &H30, &H63, &H52, &H7C, &H4D, &H1E, &H2F, &HB8, &H89,
&HDA, &HEB, _
    &H3D, &HC, &H5F, &H6E, &HF9, &HC8, &H9B, &HAA, &H84, &HB5, &HE6, &HD7, &H40, &H71,
&H22, &H13, _
    &H7E, &H4F, &H1C, &H2D, &HBA, &H8B, &HD8, &HE9, &HC7, &HF6, &HA5, &H94, &H3, &H32,
&H61, &H50, _
    &HBB, &H8A, &HD9, &HE8, &H7F, &H4E, &H1D, &H2C, &H2, &H33, &H60, &H51, &HC6, &HF7,
&HA4, &H95, _
    &HF8, &HC9, &H9A, &HAB, &H3C, &HD, &H5E, &H6F, &H41, &H70, &H23, &H12, &H85, &HB4,
```

```
&HE7, &HD6, _
    &H7A, &H4B, &H18, &H29, &HBE, &H8F, &HDC, &HED, &HC3, &HF2, &HA1, &H90, &H7, &H36,
&H65, &H54, _
    &H39, &H8, &H5B, &H6A, &HFD, &HCC, &H9F, &HAE, &H80, &HB1, &HE2, &HD3, &H44, &H75,
&H26, &H17, _
    &HFC, &HCD, &H9E, &HAF, &H38, &H9, &H5A, &H6B, &H45, &H74, &H27, &H16, &H81, &HB0,
&HE3, &HD2, _
    &HBF, &H8E, &HDD, &HEC, &H7B, &H4A, &H19, &H28, &H6, &H37, &H64, &H55, &HC2, &HF3,
&HA0, &H91, _
    &H47, &H76, &H25, &H14, &H83, &HB2, &HE1, &HD0, &HFE, &HCF, &H9C, &HAD, &H3A, &HB,
&H58, &H69, _
    &H4, &H35, &H66, &H57, &HC0, &HF1, &HA2, &H93, &HBD, &H8C, &HDF, &HEE, &H79, &H48,
&H1B, &H2A, _
    &HC1, &HF0, &HA3, &H92, &H5, &H34, &H67, &H56, &H78, &H49, &H1A, &H2B, &HBC, &H8D,
&HDE, &HEF, _
    &H82, &HB3, &HE0, &HD1, &H46, &H77, &H24, &H15, &H3B, &HA, &H59, &H68, &HFF, &HCE,
&H9D, &HAC)
End Function
Private Sub Command2_Click()
 End
End Sub
```

2. 程序运行

例 2.6　生成多项式 $g(x)=x^8+x^5+x^4+1$(正序),初始值为 0x00,求 01,02,03,04,05,06,07 这 7 个数的 CRC8 码。

解：打开“第 2 章\2.2 用程序计算 CRC8 码—生成多项式 131H\VB 程序\用查表法求 CRC8 码”文件夹,双击扩展名为.vpb 的文件,将进入 VB 编译环境,执行菜单命令“运行”→“启动”,将出现一个对话框,再单击对话框中的 Command1 按钮,将显示出一个字节的 CRC8 码,如下所示：



可知,01,02,03,04,05,06,07 这 7 个数的 CRC8 码是 0AFH。

2.2.3 C51 语言

C51 语言和 C 语言大同小异,C 语言一般是在个人电脑或笔记本电脑上运行的,C51 语言则是在 51 系列单片机上运行的。

2.2.3.1 用计算法求 CRC8 码

1. 程序清单

```
/*****************************************************************/
* 用计算法求 CRC8 码的 C51 语言程序,生成多项式 g(x) = x^8 + x^5 + x^4 + 1(正序)
/*****************************************************************/
#include<reg52.h>                                    //包含单片机寄存器的头文件
#include<intrins.h>                                  //包含_nop_()函数定义的头文件
#define uchar unsigned char
#define uint unsigned int
uchar crc8,crc,mm;
uchar SerialNum[8] = {0x01,0x02,0x03,0x04,0x05,0x06,0x07};     //0xaf
unsigned char calcrc_1byte(unsigned char abyte)
{
   unsigned char i;
   crc = crc ^ abyte;
   for(i = 8; i>0; --i)
   {
      if (crc&0x80)
      {
          crc = (crc<<1)^0x131;
      }
      else
      crc = (crc<<1);
   }
   return crc;
}
void delay1(uint x)                                  //延时 x ms 程序
{
   uchar tw;
   while (x-->0){
     for (tw = 0;tw<125;tw++){;}
   }
}
void main( )                                         //主程序
{
   uchar i;
```