Spansion® 模拟和微控制器产品

本文档包含有关 Spansion 模拟和微控制器产品的信息。尽管本文档内有原来开发该产品规格的公司名称“富士通”或“Fujitsu”， 该产品将由 Spansion 提供给现有客户和新客户。

规格的延续

本文档内容并不因产品供应商的改变而有任何修改。文档内容的其他更新，均为改善文档而进行，并已记录在文档更改摘要。日后如有需要更改文档，其更改内容也将记录在文档更改摘要。

型号的延续

Spansion 将继续提供型号以“MB”开始的现有产品。如欲订购该类产品，敬请使用本文档内列出的产品型号。

查询更多信息

如欲查询更多关于 Spansion 存储器、模拟产品和微控制器产品及其解决方案的信息，请联系您当地的销售办事处。

书末出版说明

本文档介绍的产品，其设计、开发和制造均基于一般用途，包括但不限于普通工业使用、普通办公使用、个人使用及家庭使用，不应用于：(1) 存在严重风险或危险，除非能够保证极高的安全性，否则可能对公众造成严重影响，甚至可能直接造成死亡、人员伤害、物品损坏或其他损失的用途（如核设施的核反应控制、飞机飞行控制、空中交通控制、公共交通控制、医学生命支持系统、武器系统的导弹发射控制），或者(2) 不允许出现故障的用途（如潜艇中继器和人造卫星）。请注意，对于您和 / 或任何第三方由于将产品用于上述用途而造成的任何索赔和损失，Spansion 不承担任何责任。任何半导体设备都可能发生故障。您必须在自己的设施和装置中加入安全设计措施，如冗余、防火、防止电流过载及其他异常运行情形等，以防由于此类故障而造成伤害、损坏或损失。如果根据日本 Foreign Exchange and Foreign Trade Law、美国US Export Administration Regulations 或其他国家（地区）的适用法律的规定，本文档中介绍的任何产品是在出口方面受到特别限制的商品或技术，则这些产品的出口必须预先得到相关政府的许可。

商标和声明

本文档的内容如有变更，恕不另行通知。本文档可能包含Spansion 正在开发的 Spansion 产品的相关信息。 Spansion 保留变更任何产品或停止其相关工作的权利，恕不另行通知。本文档中的信息 “ 按原样 ” 提供，对于其精确性、完整性、可操作性、对特定用途的适用性、适销性、不侵犯第三方权利等不提供任何担保或保证，也不提供任何明确的、隐含的或法定的其他担保。对于因使用本文档中的信息而造成的任何形式的任何损失，Spansion 不承担任何责任。

富士通半导体（上海）有限公司
应用笔记

MCU-AN-500032-Z-10

F²MC-8FX 家族

8 位微型控制器

MB95200 系列

CRC 的生成与校验

应用笔记

FUJITSU

修改记录

版本	日期	作者	修改记录
1.0	2009-04-08	Folix	初稿

本手册包含19页。

1. 本文档记载的产品信息及规格说明如有变动，恕不预先通知。如需最新产品信息和/或规格说明，联系富士通销售代表或富士通授权经销商。
2. 基于本文档记载信息或示意图的使用引起的对著作权、工业产权或第三方的其他权利的侵害，富士通不承担任何责任。
3. 未经富士通明文批准，不得对本文档的记载内容进行转让、拷贝。
4. 本文档所介绍的产品并不旨在以下用途: 需要极高可靠性的设备，诸如航空航天装置、海底中继器、核控制系统或维系生命的医用设施。
5. 本文档介绍的部分产品可能是“外汇及外贸管理法”规定的战略物资(或专门技术)，出口该产品或其中部分元件前，应根据该法获得正式批准。

目录

1 概要

本文档介绍了什么是 CRC 以及如何通过软件实现 CRC。

2 什么是 CRC

2.1 背景介绍

物理损坏的传输或存储介质可能给数据的传输或存储过程带来不必要的数据修改（不包括第三方的修改，比如恶意攻击者）。要检测这些错误，需要编写一些错误检测或错误校正代码，用于从数据集计算出一个值并与数据一起传输或存储。很多功能可以在一定程度上用于检测错误，其中一个是循环冗余校验（CRC）。它并不绝对安全，不能检测传输数据中的恶意修改，但是可以检测一些常见的错误，如1/2位或突发性错误，并且非常有效。不同种类的CRC有不同大小的计算值。最常见的是计算32 位值的CRC32，但是我们有时使用CRC16 或CRC8.

大多数时候，用户希望计算一个给定数据集的CRC。有时候CRC已经给出，用户希望修改数据，以便后来计算CRC值。这些情况包括固件校验和或者计算数据集的CRC，后者可以创建一个ZIP文件。接下来的章节将介绍和分析计算这些修改的方法。

2.2 CRC原理

2.2.1 CRC算法

循环冗余校验提供了一种误差控制编码方式，通过受控方式在数据中引入若干冗余来保护数据。它是一种常用的且非常有效的在各种网络传输中检测传输错误的方式。常见的 CRC 多项式可检测以下错误类型。

所有单位错误

所有双位错误

所有奇数错误，假设限制长度足够

任何脉冲时间比多项式长度小的错误

多数突发性错误

式 1 描述了 CRC 编码原理。

$$V(x) = S(x) + x^{n-k}U(x) \tag{1}$$

V(x) 是传输的 n 位长的数据，它包括原始数据，U(x) 以及代码字 S(x) （也称作 CRC-和）。S(x) 是额外添加至信息的位，用于提供冗余，以便在传输过程中检测错误。S(x) 的长度表示限制长度。最常见的 CRC 多项式的限制长度在 8-32 位之间。式 3 可用于计算 S(x) 。

$$X^{n-k}U(x) = a(x)g(x) + S(x) \tag{2}$$

$$\frac{X^{n-k}U(x)}{g(x)} = a(x) + \frac{S(x)}{g(x)} \tag{3}$$

换句话说，S (x) 是数据流相除产生的余数，并生成多项式 g (x)。 因为所有代码字都可以被g(x)整除，对于实际的代码字，式（3）左侧的余数 S(x) 必须为零。

实际编码程序在这条线路的接收和发送端是相同的。图 2-1 说明了 CRC 的编码/解码原则。

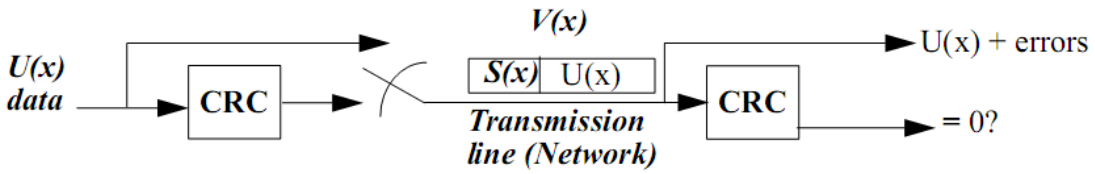

图 2-1： 使用 CRC 算法检测错误的原则

如图 2-1 所示，NT 对接收的消息进行 CRC 校验。如果 S(x) 为零，传输没有错误。另一个解决方法是只计算消息 U（x）第一部分的 CRC，然后在传输端用计算的 S（x）进行余数相加。如果结果非零，接收方将向发送方请求重新传输。

2.2.2 模数二进制除法

如果要最大化“有效信息包的最小加重平均距离”，很明显基于二进制加法的简单校验和算法缺少必需的属性。在加法中，改变其中一个消息位并不能对校验和位产生足够的影响。 但是我们不用单独开发一个更好的校验和。

研究员很早以前就发现模数二进制除法是为检验和提供必要属性的最简单的数学运算。

所有的 CRC 公式都是基于模数二进制除法的简单校验和算法。虽然不同 CRC 公式有所不同，其基本数学运算过程是相同的。

消息位添加 c 零位；添加的消息被除以一个除数。

一个预先确定的 c+1 位二进制顺序（也称做“生成多项式”）是除数。

校验和是除法运算的 c 位余数。

换句话说，生成多项式用于除以添加的消息，弃用商数，使用余数作为校验和。除法最吸引人的地方就是当消息中很小数量的位变化时，余数快速变化。和，乘，商都没有这个特性。图 2-2 举例说明了模数 2 除法。

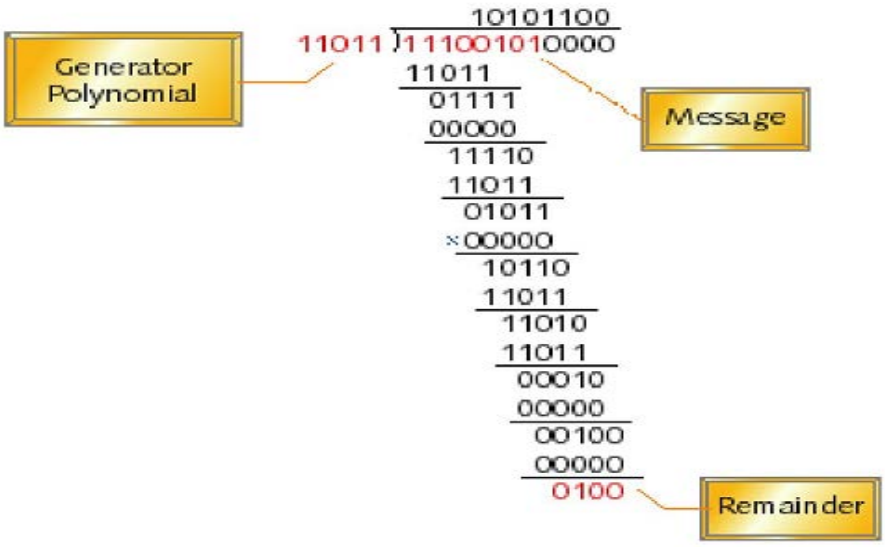

图 2-2：模数 2 二进制除法实例

在本实例中，消息包括八位，而校验和有四位。进行除法运算后，余数为 0111， 1111，0101， 1011， 1101， 0001， 0010。最后的余数就是该信息的校验和。

对大多数人来说，关于 CRC 最难的事就是如何实现。仅仅知道所有的 CRC 算法都是简单的除法并没有多大帮助。模数 2 二进制除法并不能很好地映射通用处理器的指令集，基于加法的校验和算法更直接。以 m+c 位为分子，c+1 位为分母的二进制除法算法不能解决问题，因为没有用于保存操作数的 m+c 或 c+1 位寄存器。我们应看到它们各自作为校验和的优缺点。

2.2.3 生成多项式

为什么预先设定的用于计算 CRC 的 c+1 位除数叫做生成多项式？许多关于 CRC 的解释实际上都是在回答这个问题。这让其作者和读者了解到无数关于多项式算法和 CRC 应用的数学基础的信息。但是学术上的东西对于充分理解并使用 CRC 用处不大，相反会造成混淆。

这里我们只能说除数有时也称作生成多项式，不能随意编造。很多有名的生成多项式被用作各种国际通信标准的组成部分。如果你对多项式算法有所了解，应该知道在生成检验和方面部分生成多项式要比其他好。推荐使用国际上通用的那些生成多项式。

表 2-1 列出了某些 16 的最常用的生成器多项式 - 和 32 位 CRCs。 切记这个除数的宽度更宽比余数总是一。 因此，例如，您会使用 17 位生成器多项式，每当需要 16 位校验和。

表2-1. 国际标准 CRC 多项式

名称	生成多项式	符号
CRC-CCITT	G (x)=x16+x12+x5+1	0x1021
CRC-16	G (x)=x16+x15+x2+1	0x8005
CRC-32	G (x)=x32+x26+x23+x22+x16+x12+x10+x8+x7+x5+x4+x2+x+1	0x104C11DB7

和其他类型的校验和一样，CRC 宽度在算法的错误检测功能方面扮演着重要角色。忽略由特殊校验和算法检测到的特殊类型的错误，我们发现校验和的宽度严格制约了可检测错误的百分比。c 位校验和只能有一个 2c 值。因为可能的消息数明显要大很多，两个或多个消息可能有相同的校验和。如果在传输中，其中一个消息被转换为其他消息，校验和看上去正确，而接收端将不知道接收的消息是否有误。该错误的发生几率与校验和的宽度直接相关，发生几率达 $1/2^c$。因此，任意误差被检测的概率为 $1-1/2^c$。

重复一遍，检测任意误差的概率随着校验和宽度的增加而增加。16 位校验和可检测 99.9985%的错误，而 8 位校验和的错误检测率仅为 99.6094%。32 位校验和的检测率最高，达到 99.9999%，这对于 CRC 和基于加法的校验和都一样。不一样的是，CRC 的特殊错误检测率为 100%。例如，加法校验和无法检测两个相反的位反向（一个变为 0，一个变为 1）造成的错误，而 CRC 却不一样。

通过使用表 1 中所列出的生成多项式来计算校验和，可无误检测以下错误类型。

有任何一位错误的消息

有任何两位错误的消息（不管离得多远，在哪一列，等等）。

有任何奇数位错误的消息（不管在哪里）。

有与校验和等宽的突发性错误的信息。

可检测错误的第一级由基于加法的校验和或者甚至一个简单的校验位检测。但是，中间两级的错误代表比其他类型校验和更强的检测能力。第四级可检测错误看起来与基于加法的校验和检测的错误类似，但 CRC 将检测任何奇数位的错误。因此宽错误群的检测将限于有偶数位错误的消息。所有其他类型的错误的检测概率高达 $1-1/2^c$。

3 CRC 软件实现

CRC-16 用于描述 CRC 软件。

3.1 逐位实现

逐位实现CRC计算的方法有很多种。一旦一个位被转移至左边，计算将被执行。

计算次数=总位数 – 16

下图说明了逐位实现CRC 计算的流程。

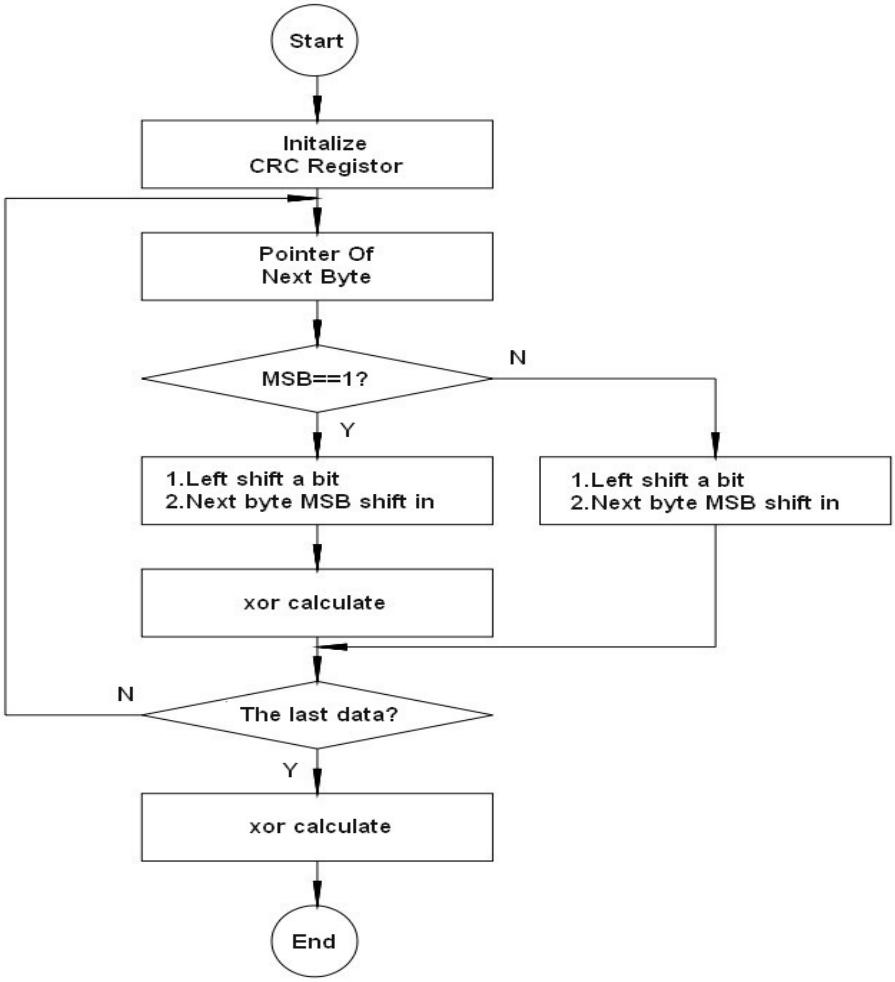

图 3-1：逐位 CRC 计算

UWORD CRC16_Bitbybit(UCHAR *pMsg,int len)

返回　　: CRC value.

参数　　: pMsg,data buffer.

　　len,length of data.

描述　　: Bit by bit calculate CRC.

示例　　: UWORD Val;

　　UCHAR datbuf[16];

　　Val=CRC16_Bitbybit(datbuf,16);

```
UWORD CRC16_Bitbybit(UCHAR *pMsg,int len)
{
  UCHAR i,j;
  UWORD CRC16=0;

  for(j=0;j<len;j++)
  {
    pMsg++;
    for(i=0;i<8;i++,*pMsg<<=1)
    {
      if(CRC16 & 0x8000)
      {
        CRC16<<=1;
        CRC16|=(*pMsg & 0x80)>>7;
        CRC16^=0x8005;
      }
      else
      {
        CRC16<<=1;
        CRC16|=0x8005;
      }
    }
  }
```

3.2 表查找实现

基于8位表查找的CRC计算可显著减少MCU的占空比，使其运行速度更快。

下图显示了表查找 CRC 计算的流程。

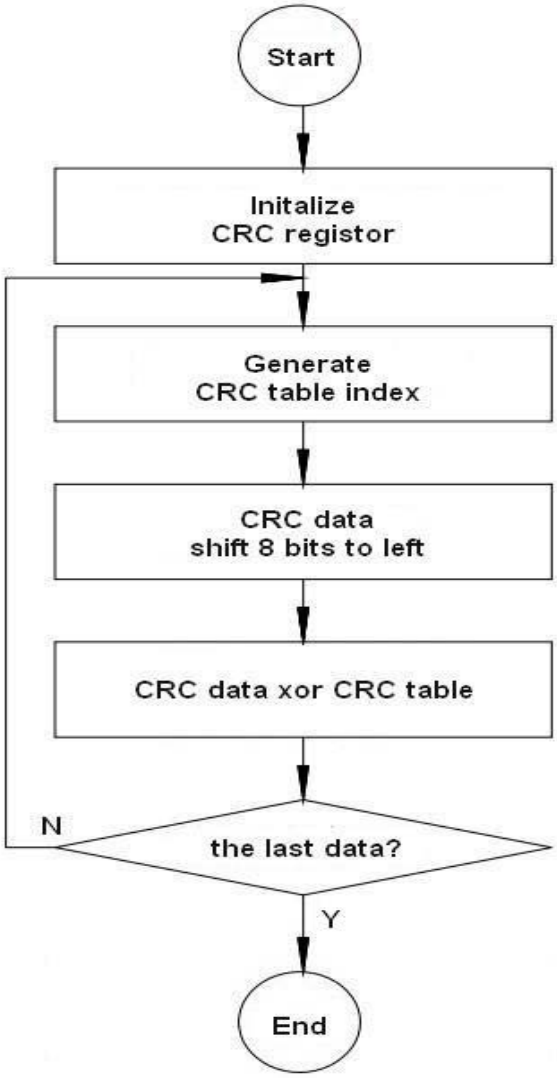

图 3-2：表查找实现

3.2.1 CRC API

UWORD CRC16_Table(UCHAR *pMsg,int len)

返回 : CRC value.

参数 : pMsg,data buffer.

len,length of data.

描述 : Table Look calculate CRC.

示例 : UWORD Val;

UCHAR datbuf[16];

Val=CRC16_Table(datbuf,16);

```
UWORD CRC16_Table(UCHAR *pMsg, int len)
{
    UCHAR indx;
    UWORD CRC16=0;
    while(len--)
    {
        indx = CRC16 >> 8;
        CRC16 <<= 8;
        CRC16 ^= CRCTable[indx ^ *pMsg++];
    }
    return CRC16;
}
```

void CRC16TableGen ()

返回　　: None.

参数　　: None.

描述　　: Generate CRC table.

示例　　: CRC16TableGen();

```
void CRC16TableGen()
{
  int i,j;
  UWORD CRC16;

  for(i=0;i<256;i++)
  {
    CRC16=i;
    CRC16<<=8;
    for(j=0;j<8;j++)
    {
      if(CRC16&0x8000)
      {
        CRC16<<=1;
        CRC16^=0x8005;
      }
      else
        CRC16<<=1;
    }
    CRCTable[i]=CRC16;
  }
}
```

3.2.2 CRC表

```
UWORD CRCTable[256]=
{
    0x0000,0x8005,0x800F,0x000A,0x801B,0x001E,0x0014,0x8011,
    0x8033,0x0036,0x003C,0x8039,0x0028,0x802D,0x8027,0x0022,
    0x8063,0x0066,0x006C,0x8069,0x0078,0x807D,0x8077,0x0072,
    0x0050,0x8055,0x805F,0x005A,0x804B,0x004E,0x0044,0x8041,
    0x80C3,0x00C6,0x00CC,0x80C9,0x00D8,0x80DD,0x80D7,0x00D2,
    0x00F0,0x80F5,0x80FF,0x00FA,0x80EB,0x00EE,0x00E4,0x80E1,
    0x00A0,0x80A5,0x80AF,0x00AA,0x80BB,0x00BE,0x00B4,0x80B1,
    0x8093,0x0096,0x009C,0x8099,0x0088,0x808D,0x8087,0x0082,
    0x8183,0x0186,0x018C,0x8189,0x0198,0x819D,0x8197,0x0192,
    0x01B0,0x81B5,0x81BF,0x01BA,0x81AB,0x01AE,0x01A4,0x81A1,
    0x01E0,0x81E5,0x81EF,0x01EA,0x81FB,0x01FE,0x01F4,0x81F1,
    0x81D3,0x01D6,0x01DC,0x81D9,0x01C8,0x81CD,0x81C7,0x01C2,
    0x0140,0x8145,0x814F,0x014A,0x815B,0x015E,0x0154,0x8151,
    0x8173,0x0176,0x017C,0x8179,0x0168,0x816D,0x8167,0x0162,
    0x8123,0x0126,0x012C,0x8129,0x0138,0x813D,0x8137,0x0132,
    0x0110,0x8115,0x811F,0x011A,0x810B,0x010E,0x0104,0x8101,
    0x8303,0x0306,0x030C,0x8309,0x0318,0x831D,0x8317,0x0312,
    0x0330,0x8335,0x833F,0x033A,0x832B,0x032E,0x0324,0x8321,
    0x0360,0x8365,0x836F,0x036A,0x837B,0x037E,0x0374,0x8371,
    0x8353,0x0356,0x035C,0x8359,0x0348,0x834D,0x8347,0x0342,
    0x03C0,0x83C5,0x83CF,0x03CA,0x83DB,0x03DE,0x03D4,0x83D1,
    0x83F3,0x03F6,0x03FC,0x83F9,0x03E8,0x83ED,0x83E7,0x03E2,
    0x83A3,0x03A6,0x03AC,0x83A9,0x03B8,0x83BD,0x83B7,0x03B2,
    0x0390,0x8395,0x839F,0x039A,0x838B,0x038E,0x0384,0x8381,
    0x0280,0x8285,0x828F,0x028A,0x829B,0x029E,0x0294,0x8291,
    0x82B3,0x02B6,0x02BC,0x82B9,0x02A8,0x82AD,0x82A7,0x02A2,
    0x82E3,0x02E6,0x02EC,0x82E9,0x02F8,0x82FD,0x82F7,0x02F2,
    0x02D0,0x82D5,0x82DF,0x02DA,0x82CB,0x02CE,0x02C4,0x82C1,
    0x8243,0x0246,0x024C,0x8249,0x0258,0x825D,0x8257,0x0252,
    0x0270,0x8275,0x827F,0x027A,0x826B,0x026E,0x0264,0x8261,
    0x0220,0x8225,0x822F,0x022A,0x823B,0x023E,0x0234,0x8231,
    0x8213,0x0216,0x021C,0x8219,0x0208,0x820D,0x8207,0x0202
};
```

3.3 CRC 检查

CRC校验和生成的方法几乎相同。在CRC校验中，如果接收的数据块的所有数据（包括CRC字节）生成的CRC校验代码为零，表明数据全部正确。

4 注意事项

在应用中，用户可以定义不同的CRC生成多项式来实现CRC-8、 CRC-16 或者 CRC-32。

5 更多信息

关于富士通半导体更多的产品信息，请访问以下网站：

英文版本地址：

http://www.fujitsu.com/cn/fsp/services/mcu/mb95/application_notes.html

中文版本地址：

http://www.fujitsu.com/cn/fss/services/mcu/mb95/application_notes.html

6 附录