## 参考资料

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [1] | 36.212 Multiplexing and channel coding | V8.8.0 | 2009-12 | 3GPP |
|  |  |  |  |  |

### LTE系统中定义

从参考文献[6]，可以看出，LTE系统共定义4种的CRC生成多项式。相对WCDMA/TD-SCDMA系统，其中第gCRC24A(D)是新定义的，其他3个都是WCDMA/TD-SCDMA系统中存在的。其中在LTE系统中，余式多项式只有一种情况为全0。

表 3 CRC生成多项式定义

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 名称 | 定义 | 描述 | 参考文献及章节 |
| 1 | gCRC24A(D) | [D24 + D23 + D18 + D17 + D14 + D11 + D10 + D7 + D6 + D5 + D4 + D3 + D + 1] | [1 1 0 0 0 0 1 1 0 0 1 0 0 1 1 0 0 1 1 1 1 1 0 1 1] | [6] 5.1.1 |
| 2 | CRC-24 | G24(D)=D24+D23+D6+D5+D+1 | [1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 ] | [4][5] 4.2.1.1，[6]5.1.1 |
| 3 | CRC-16 | g16(D)=D16+D12+D5+1 | [1 1 0 0 1 1 0 1 1][1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 1] | [4] [5]4.2.1.1，[6]5.1.1 |
| 4 | CRC-8 | g8(D)=D8+D7+D4+D3+D+1 | [1 1 0 0 1 1 0 1 1] | [4] [5]4.2.1.1，[6]5.1.1 |

## CRC原理

### 基本原理

循环校验码（Cyclic Redundancy Check，简称CRC码），它是利用除法及余数的原理来做错误检测（Error Detecting）的。它将要发送的数据比特序列当作一个多项式的系数，发送时用双方预先约定的生成多项式去除，求得一个余数多项式，将余数多项式加到数据多项式之后发送到接收端，接收端用同样生成多项式去除接收到的数据，进行计算，然后把计算结果和实际接收到的余数多项式数据进行比较，相同的话表示传输正确。CRC校验检错能力强，容易实现，是目前数据通信系统中应用最广的检错码编码方式之一。

CRC是一种线性分组码，具有较强的检错能力并有许多特殊的代数性质，前k位为信息码元，后r位为校验码元，他除了具有线性分组码的封闭性之外，还具有循环性。

任意一个由二进制位串组成的代码都可以和一个系数仅为‘0’和‘1’的多项式一一对应。例如：代数1010111对应的多项式。

若设码字长度为N，信息字段为K位，校验字段为R位（），则对于CRC码集中的任一码字，存在且仅存在一个R次多项式，使得

 （1）



图 1 CRC 结构图

其中：为K次信息多项式，为R-1次校验多项式，称为生成多项式：

 （2）

发送方通过指定的产生CRC码字，接收方则通过该来验证收到的CRC码字。

### CRC分类

余式多项式初始状态为全0或全1，分为两种情况。相应的，余式多项式为互补关系。例如（假设待发送信息为：1 0 0 1 1 0 0 1）：

情况1：若余式初始状态全为0，假设此时发送比特为100（即校验字段为：100）。即发送方发出的传输字段为：

1 0 0 1 1 0 0 1 1 0 0

信息字段 校验字段

情况2：若余式初始状态全为1，假设此时发送比特为011（即校验字段为：011）。即发送方发出的传输字段为：

1 0 0 1 1 0 0 1 0 1 1

信息字段 校验字段

## CRC算法实现

### 基于多项式除法CRC算法

CRC码是由两部分组成，前部分是信息码，就是需要校验的信息，后部分是校验码，如果CRC码共长n个bit,信息码长k个bit，它的编码规则是：

1、首先将原信息码（k bit）左移r位（k+r=n）,对应多项式为。

2、运用一个生成R次多项式g(x) (也可看成二进制) 用模2除上面的式子，得到的余数就是校验码r=R。

下面按照二进制多项式除法（异或操作，即xor）进行计算CRC校验比特。下面以最简单的生成多项式为例，进行详细说明。

1、假设信息字段代码为：10011001；对应。

2、假设生成多项式：，对应的代码为：1011。

3、移位后信息字段代码：，对应的代码为：10011001000；

4、采用多项式除法，如图 2所示：



图2 多项式除法示例

### 基于DSP的CRC算法

下面按照DSP算法过程，对上述示例进行详细描述。假设DSP的处理顺序是从右向左，采用原位操作，其中C为溢出寄存器，具体操作过程如图 3所示。

图 3 CRC校验比特生成过程

从图 3‑3可以看出，输入数据与生成多项式的异或操作，分为两步完成：首先根据寄存器、输入比特和固定值1进行异或操作得到C值，然后根据C值决定是否进行a0寄存器的更新过程（当C=0时，需要对寄存器a0值与后续生成多项式进行异或操作并保存该值；否则，不需要更新寄存器值。由于寄存器承载的是寄存器值与后续生成多项式异或值，因此与除法多项式的判决条件是异或关系）。

从图 3还可以看出，根据寄存器、输入比特和固定值1进行异或操作得到C值时，其中1是固定值，如果只根据寄存器和输入比特进行异或操作得到C值时，此时判决条件正好是相反的（与多项式除法过程一致），即，。

### CRC检测算法

对于接收方，使用相同的生成多项式进行校验（即CRC-3）。接收到的消息字段+校验码除以生成多项式（或DSP算法过程），如果能够除尽（全0/或全1），则正确；否则接收消息字段有误，如图 4、图 5所示。

SS

图 4 初始状态为全0检测过程



图 5 初始状态为全1检测过程

在图5过程中，也可以首先保存校验比特，将对应状态使用全1进行填充，再进行多项式除法，最后将结果与保存校验比特进行比较即可。

参考程序：

function [Parity\_Bits] = CRC\_Code(CRC\_code\_in, CRC\_g, CRC\_State\_T)

A = length(CRC\_code\_in); % A is tne length of input data

L = length(CRC\_g) - 1; % L is the length of CRC

B = A + L; % B is the length of output data

if CRC\_State\_T == 0

State = zeros(1,L);

else

State = ones(1,L);

end

data = [CRC\_code\_in State]; % count the CRC bits by the input data

if B >= 0

for k = 1: A % k is the cycle variable

if data(k) ~= 0

data(k : k+L) = xor(data(k : k+L), CRC\_g);

end

end % Adding the CRC bits

Parity\_Bits = data(A+1 : A+L);

% Parity\_Bits = [CRC\_code\_in data(A+1 : A+L)];

end