Experimental 5: Implementation of Cyclic Redundancy Check

Xie Sangma
School of Automation

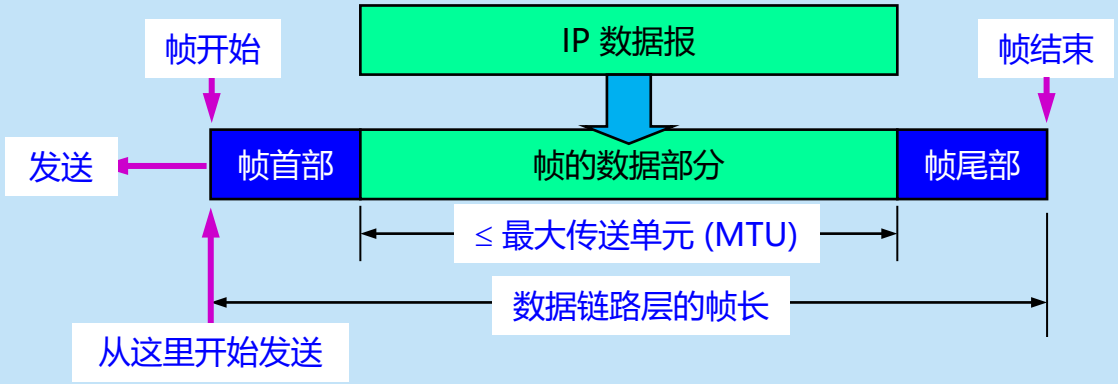
The three basic problems of the Data Link Layer.

- There are many types of data link layer protocols, but there are three basic problems that are common. These three basic problems are::
  1. Framing encapsulation
  2. Transparent transmission
  3. Error detection

三个基本问题 – 封装成帧

1. 封装成帧

- **封装成帧** (framing) 就是在一段数据的前后分别添加首部和尾部，然后就构成了一个帧。
- 首部和尾部的一个重要作用就是进行**帧定界**，还包括许多必要的控制信息。



用帧首部和帧尾部封装成帧

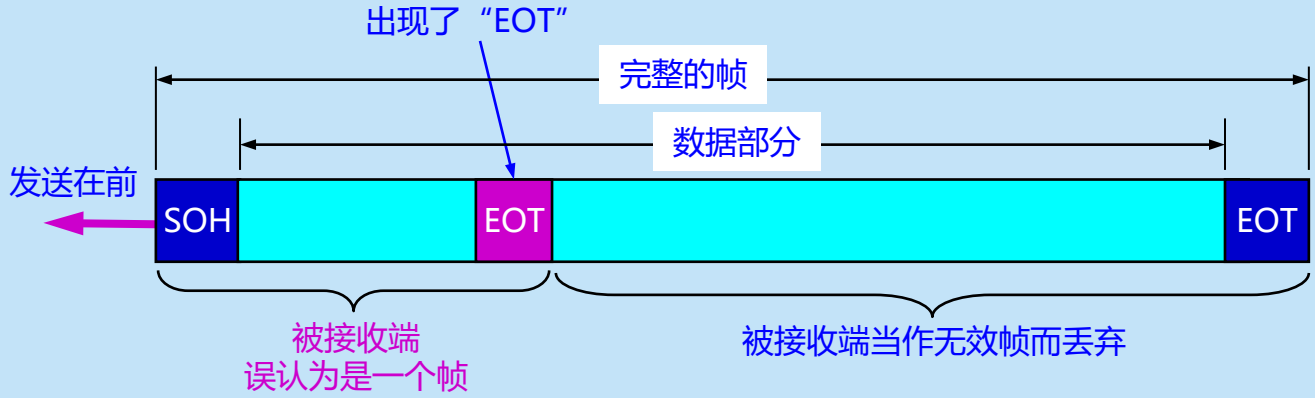
- 每一种链路层协议都规定了所能传送的帧的**数据部分长度上限——最大传送单元MTU（Maximum Transfer Unit）.**

三个基本问题 – 透明传输

2. 透明传输

透明

- 指某一个实际存在的事物看起来却好像不存在一样。
- “在数据链路层透明传送数据”表示无论发送什么样的比特组合的数据，这些数据都能够按照原样没有差错地通过这个数据链路层。
- 如果数据中的某个字节的二进制代码恰好和 SOH 或 EOT 一样，数据链路层就会错误地“找到帧的边界”。

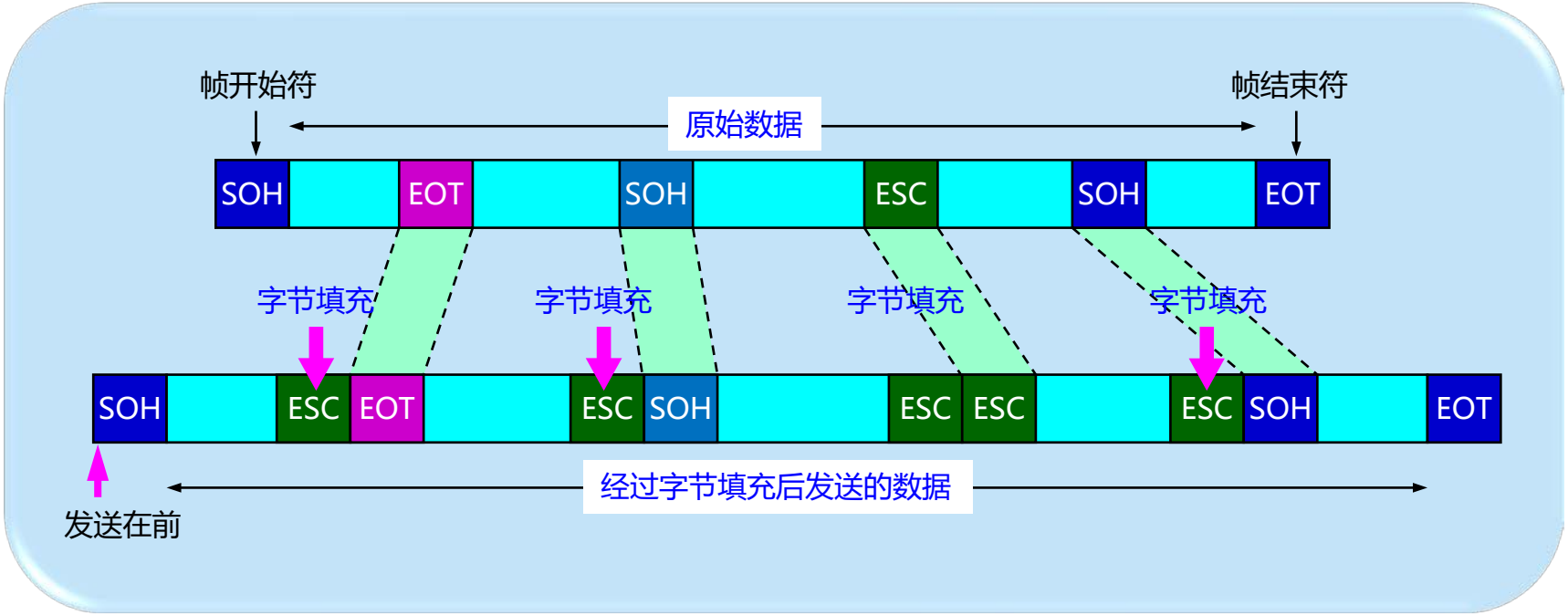


数据部分恰好出现与 EOT 一样的代码

三个基本问题 – 透明传输

解决透明传输问题：字节填充 (byte stuffing) 或字符填充 (character stuffing)

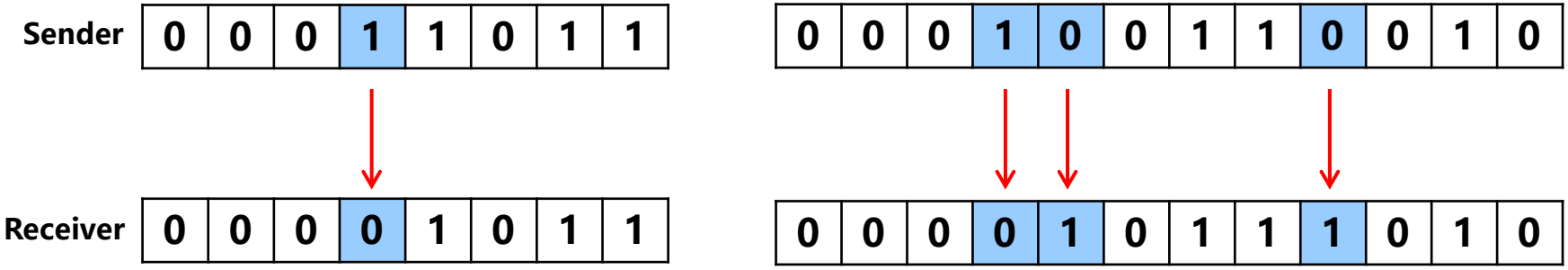
- **发送端的数据链路层在数据中出现控制字符“SOH”或“EOT”的前面插入一个转义字符“ESC”(其十六进制编码是1B)。接收端的数据链路层在将数据送往网络层之前删除插入的转义字符。**



Three basic problems - Error detection

3. Error detection

During the transmission process, bit errors may occur: 1s may become 0s and 0s may become 1s.



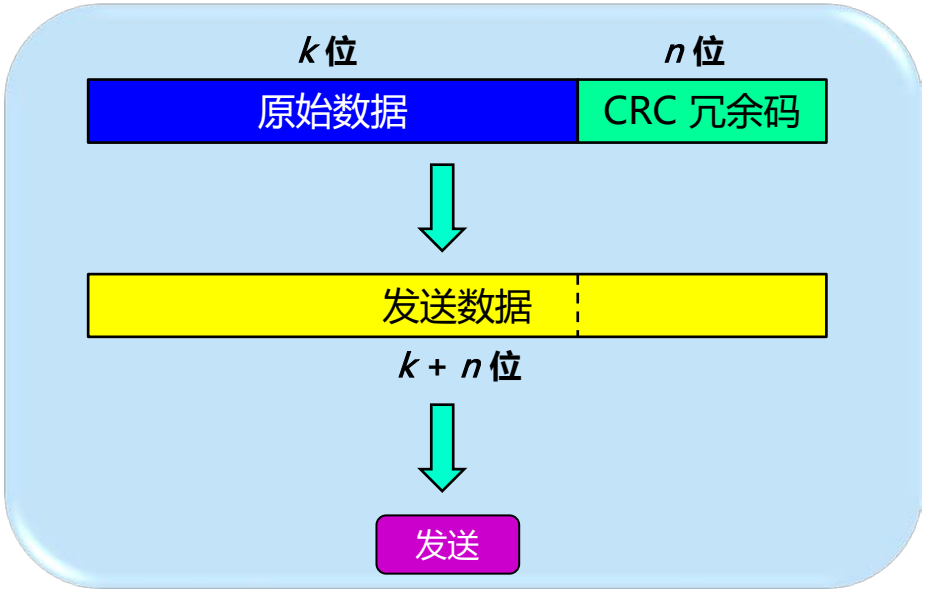
Three basic problems - Error detection

3. Error detection

- To ensure the reliability of data transmission, various error detection measures must be used when transmitting data in computer networks.
- The error detection technique widely used in the frames transmitted in the data link layer is the Cyclic Redundancy Check (CRC) method.

三个基本问题 – 差错检测

循环冗余检验的原理



- 在发送端，先把数据划分为组。假定每组 k 个比特。
- 在每组 M 后面再添加供差错检测用的 n 位冗余码，然后一起发送出去。

- 模2加法：不考虑进位的加法
  - 0+0=0
  - 1+0=1
  - 0+1=1
  - 1+1=0

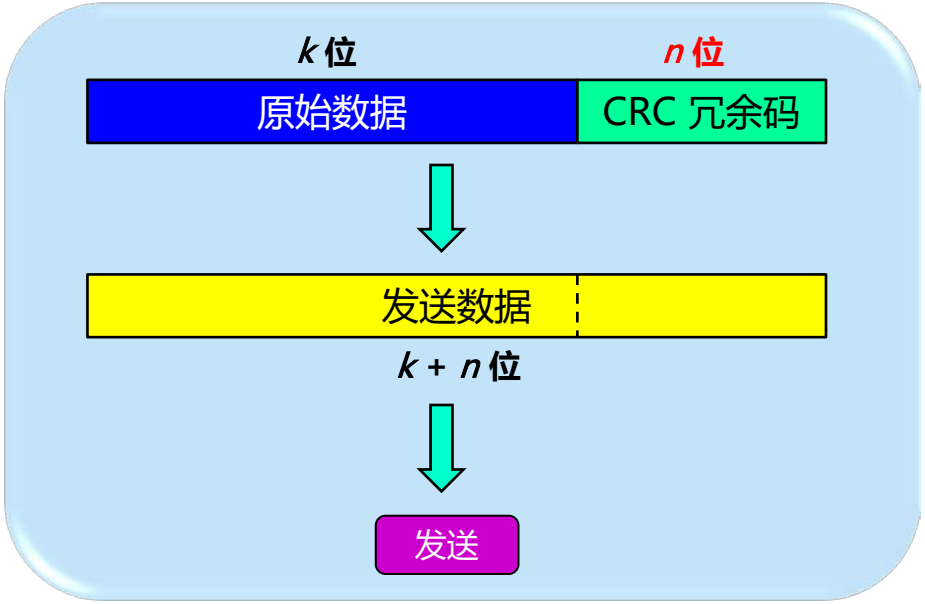
异或操作

- 模2减法：不考虑借位的加法
  - 0-0=0
  - 1-0=1
  - 0-1=1
  - 1-1=0

三个基本问题 – 差错检测

冗余码的计算

- 假设选定的除数P长度为（n+1）位，则在原始数据*M*后面添加 *n* 个 0（假设原始数据为k位）。
- 得到的 (*k* + *n*) 位的数除以事先选定好的除数 *P*，得出商是 *Q* 而余数是 *R*，余数 R 比除数 *P* 少 1 位，即 *R* 是 *n* 位。
- 将余数 *R* 作为冗余码拼接在数据 *M* 后面，一起发送出去。



三个基本问题 – 差错检测

接收端对收到的每一帧进行 CRC 检验：把收到的每一个帧都除以同样的除数P（模2运算），然后检查得到的余数R。

- (1) 若得出的余数 $R = 0$，则判定这个帧没有差错，就接受 (accept)。
- (2) 若余数 $R \neq 0$，则判定这个帧有差错，就丢弃。
- 但这种检测方法并不能确定究竟是哪一个或哪几个比特出现了差错。
- 只要经过严格的挑选，并使用位数足够多的除数 P，那么出现检测不到的差错的概率就很小很小。

三个基本问题 – 差错检测

冗余码的计算举例

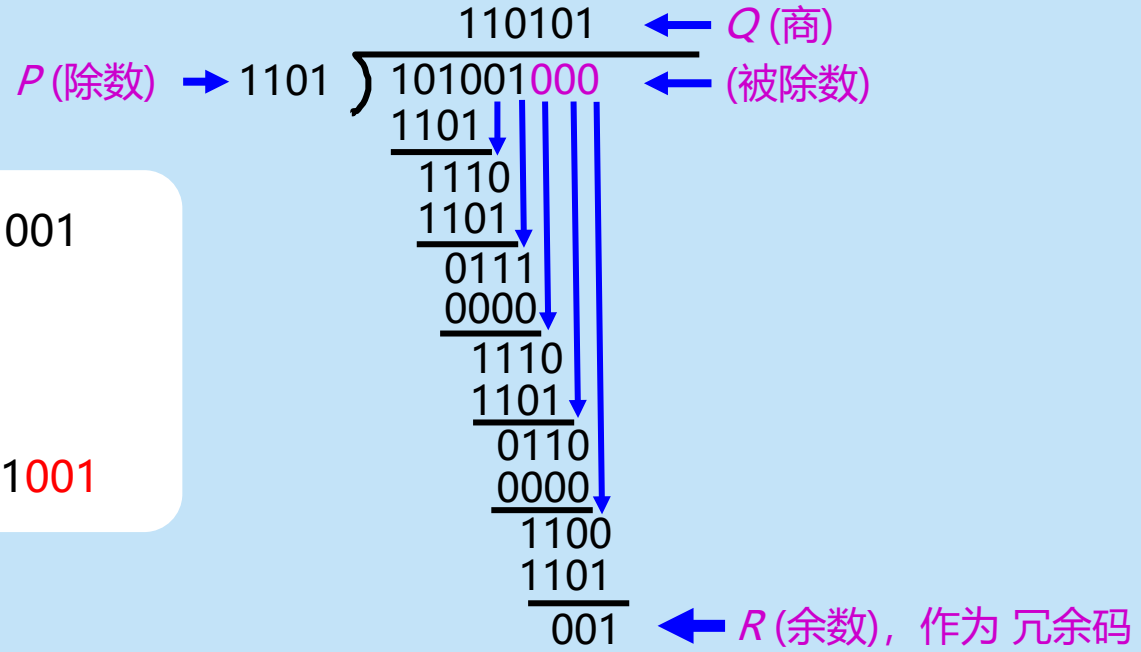
- 现在 $k = 6$, $M = 101001$。
- 设 除数 $P = 1101$，则$n = 3$
- 被除数是 101001000。
- 模 2 运算的结果是：商 $Q = 110101$，余数 $R = 001$。
- 把余数 R 作为冗余码添加在数据 M 的后面发送出去。发送的数据是：101001001，共 $(k + n)$ 位。
- 余数必须比除数少且只少一位，不够就补0

三个基本问题 – 差错检测

循环冗余检验的原理说明

原始数据 M = 101001
除数 P = 1101

得到:
发送数据 = 101001001



- **模2除法具有下列三个性质**
- **当最后余数的位数小于除数位数时，除法停止。**
- **当被除数的位数小于除数位数时，则商数为0，被除数就是余数。**
- **只要被除数或部分余数的位数与除数一样多，且最高位为1，不管其他位是什么数，皆可商1。**

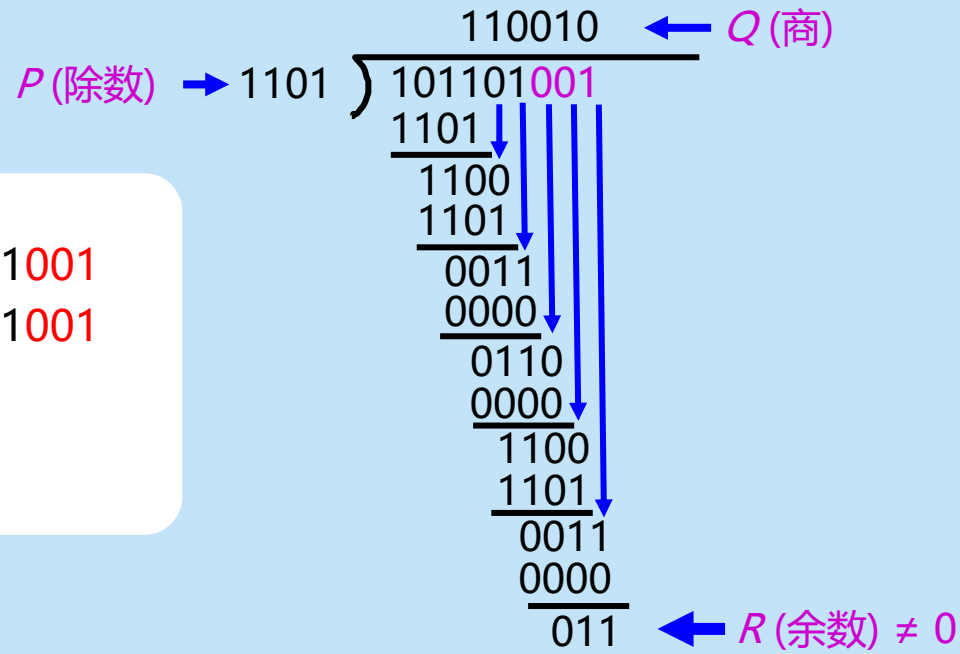
三个基本问题 – 差错检测

循环冗余检验的原理说明

- 若发送的数据101001001中途出现比特差错，变成了101101001，则循环冗余检验如何检测出数据出现比特差错?

发送数据 = 101001001
得到数据 = 101101001

除数 P = 1101



三个基本问题 – 差错检测

循环冗余检验的原理说明

- 另一种较方便的方法是用多项式来表示循环冗余检验过程。
- 在上面的例子中，可以用多项式$P(X) = X^3+X^2+1 = 1\times X^3 + 1\times X^2+ 0\times X^1 +1\times X^0$，表示上面的除数1101。
- 比如多项式$P(X)=X^8+X^2+X+1$，则表示除数为100000111。
- 生成多项式的选取是个很有难度的问题，如果选的不好，那么检出错误的概率就会低很多

CRC参数模型

- **国际常用的模型表**

CRC算法名称	多项式公式	宽度	多项式	初始值	结果异或值	输入值反转	输出值反转
CRC-4/ITU	$x^4 + x + 1$	4	03	00	00	true	true
CRC-5/EPC	$x^4 + x^3 + 1$	5	09	09	00	false	false
CRC-5/ITU	$x^5 + x^4 + x^2 + 1$	5	15	00	00	true	true
CRC-5/USB	$x^5 + x^2 + 1$	5	05	1F	1F	true	true
CRC-6/ITU	$x^6 + x + 1$	6	03	00	00	true	true
CRC-7/MMC	$x^7 + x^3 + 1$	7	09	00	00	false	false
CRC-8	$x^8 + x^2 + x + 1$	8	07	00	00	false	false
CRC-8/ITU	$x^8 + x^2 + x + 1$	8	07	00	55	false	false
CRC-8/ROHC	$x^8 + x^2 + x + 1$	8	07	FF	00	true	true
CRC-8/MAXIM	$x^8 + x^5 + x^4 + 1$	8	31	00	00	true	true

CRC参数模型

- **国际常用的模型表**

CRC算法名称	多项式公式	宽度	多项式	初始值	结果异或值	输入值反转	输出值反转
CRC-16/IBM	$x^{16}+x^{15}+x^{2}+1$	16	8005	0000	0000	true	true
CRC-16/MAXIM	$x^{16}+x^{15}+x^{2}+1$	16	8005	0000	FFFF	true	true
CRC-16/USB	$x^{16}+x^{15}+x^{2}+1$	16	8005	FFFF	FFFF	true	true
CRC-16/MODBUS	$x^{16}+x^{15}+x^{2}+1$	16	8005	FFFF	0000	true	true
CRC-16/CCITT	$x^{16}+x^{12}+x^{5}+1$	16	1021	0000	0000	true	true
CRC-16/CCITT-FALSE	$x^{16}+x^{12}+x^{5}+1$	16	1021	FFFF	0000	false	false
CRC-16/x^{5}	$x^{16}+x^{12}+x^{5}+1$	16	1021	FFFF	FFFF	true	true
CRC-16/XMODEM	$x^{16}+x^{12}+x^{5}+1$	16	1021	0000	0000	false	false
CRC-16/DNP	$x^{16}+x^{13}+x^{12}+x^{11}+x^{10}+x^{8}+x^{6}+x^{5}+x^{2}+1$	16	3D65	0000	FFFF	true	true

CRC参数模型

- 国际常用的模型表

CRC算法名称	多项式公式	宽度	多项式	初始值	结果异或值	输入值反转	输出值反转
CRC-32	$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+1$	32	04C11DB7	FFFFFFFF	FFFFFFFF	true	true
CRC-32/BZIP2	$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+1$	32	04C11DB7	FFFFFFFF	FFFFFFFF	false	false
CRC-32/MPEG-2	$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+1$	32	04C11DB7	FFFFFFFF	00000000	false	false

CRC参数模型

- **国际常用的模型表**

- 一个完整的CRC参数模型应该包含以下信息：WIDTH，POLY，INIT，REFIN，REFOUT，XOROUT。
- NAME：参数模型名称。
- WIDTH：宽度，即生成的CRC数据位宽，如CRC-8，生成的CRC为8位
- POLY：十六进制多项式，省略最高位1，如 $x^8 + x^2 + x + 1$，二进制为1 0000 0111，省略最高位1，转换为十六进制为0x07。
- INIT：CRC初始值，和WIDTH位宽一致。
- REFIN：true或false，在进行计算之前，原始数据是否翻转，如原始数据：0x34 = 0011 0100，如果REFIN为true，进行翻转之后为0010 1100 = 0x2c
- REFOUT：true或false，运算完成之后，得到的CRC值是否进行翻转，如计算得到的CRC值：0x97 = 1001 0111，如果REFOUT为true，进行翻转之后为11101001 = 0xE9。
- XOROUT：计算结果与此参数进行异或运算后得到最终的CRC值，和WIDTH位宽一致。

CRC算法名称	多项式公式	宽度	多项式	初始值	结果异或值	输入值反转	输出值反转
CRC-8	$x^8 + x^2 + x + 1$	8	07	00	00	false	false

- **通常如果只给了一个多项式，其他的没有说明则：INIT=0x00，REFIN=false，REFOUT=false，XOROUT=0x00。**

CRC参数模型

- 以CRC-8为例：
  - 根据CRC参数模型表，得到CRC-8的参数如下：

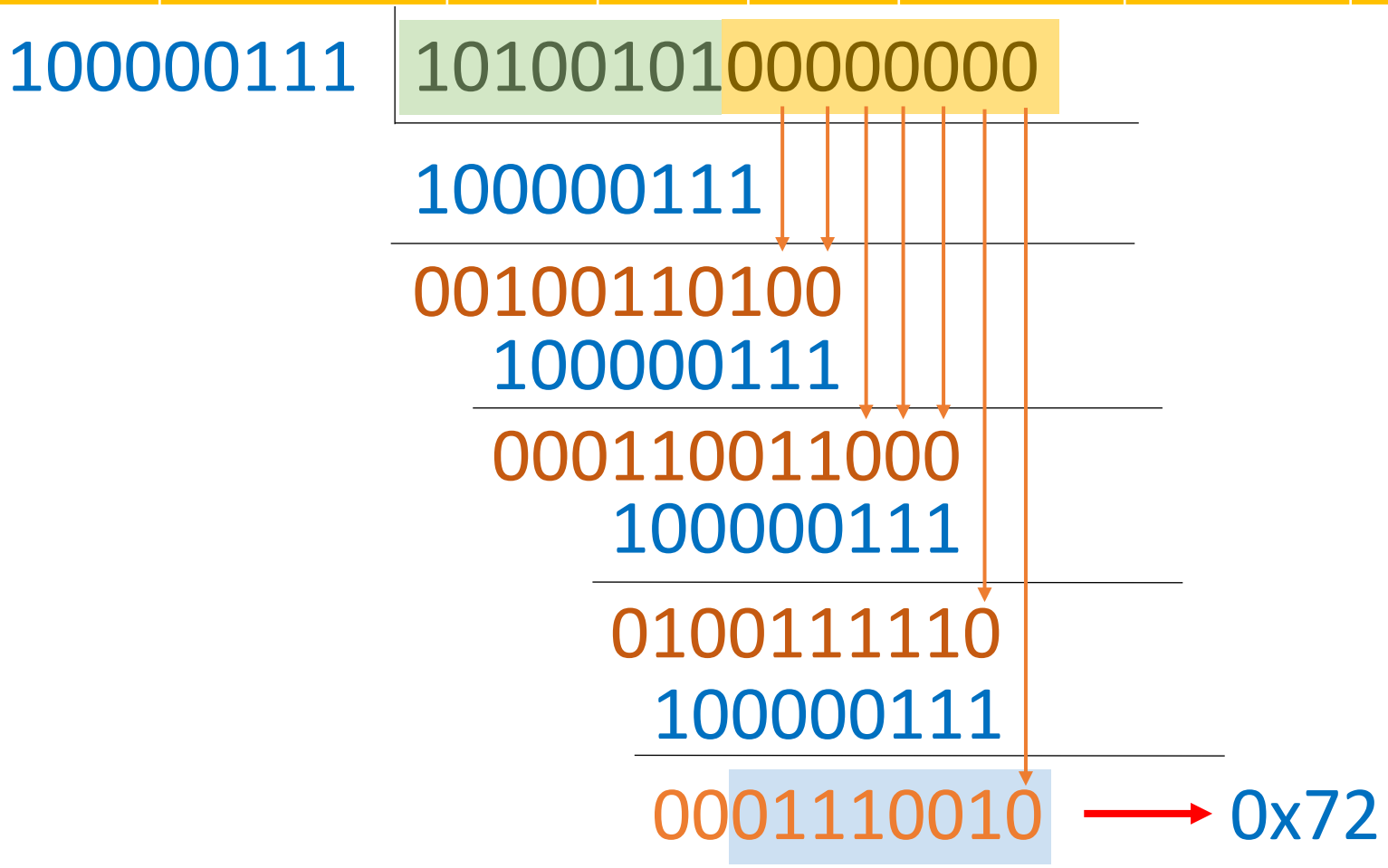
CRC算法名称	多项式公式	宽度	多项式	初始值	结果异或值	输入值反转	输出值反转
CRC-8	$x^8 + x^2 + x + 1$	8	07	00	00	false	false

  - 以x^8+x^2+x+1（0x07）多项式，则除数为100000111（二进制）
  - 假设要传送的原始数据为0xA5（二进制为：1010 0101）
  - 将原始数据左移8位（即后面添加8个0）：1010 0101 0000 0000
  - 先进行高9位与除数异或， 1010 0101 0000 0000。
  - 当多项式最高位为1，才进行异或计算，异或后最高位为0，下次也不需要异或，这样需要采用代码计算的方式，就可以把最高位去掉不需要异或，最后结果是一样的。

CRC参数模型

- 以CRC-8为例：

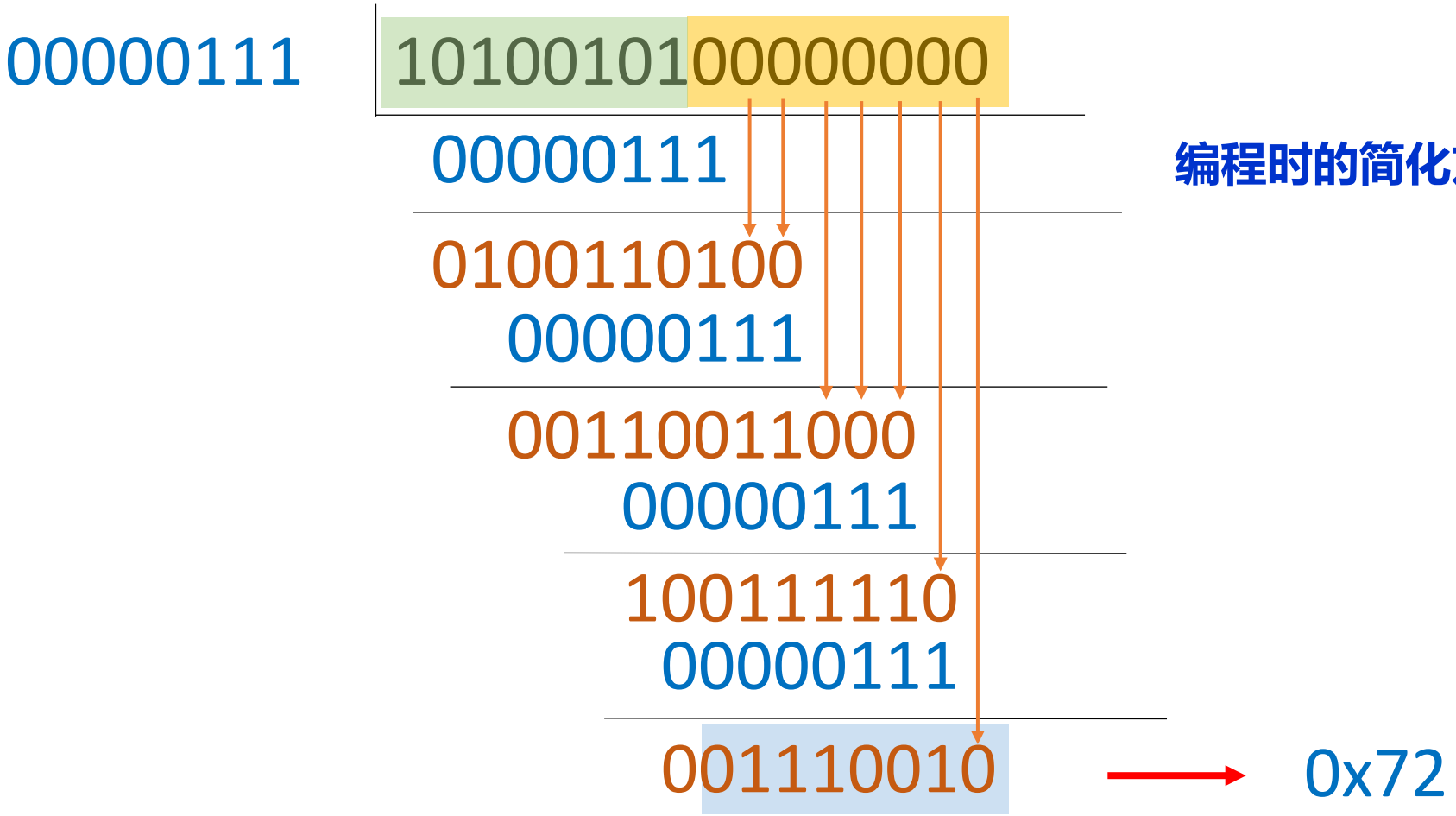
CRC算法名称	多项式公式	宽度	多项式	初始值	结果异或值	输入值反转	输出值反转
CRC-8	$x^8 + x^2 + x + 1$	8	07	00	00	false	false



CRC参数模型

- 以CRC-8为例：

CRC算法名称	多项式公式	宽度	多项式	初始值	结果异或值	输入值反转	输出值反转
CRC-8	$x^8 + x^2 + x + 1$	8	07	00	00	false	false



编程时的简化方法

CRC参数模型

- CRC算法实践思路：
  - 假设生成多项式为：100000111（简记为0x07），也就是CRC-8，先考虑输入数据为一个字节的简单情况，则计算步骤如下：

① 将CRC寄存器（8-bits，比生成多项式少1bit）赋初值0

② 将原始数据与CRC寄存器异或，结果保存在CRC寄存器中

③ for循环（ 8-bits ）

④ If (CRC寄存器首位是1)

⑤ 左移1位再异或

⑥ else(CRC寄存器首位是0)

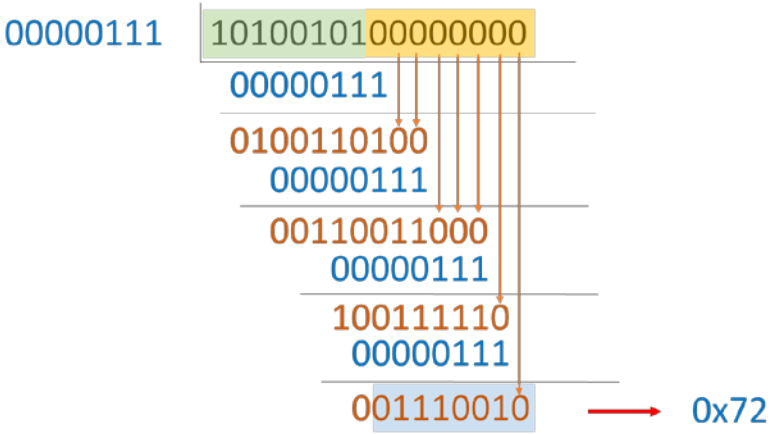
⑦ 左移1位

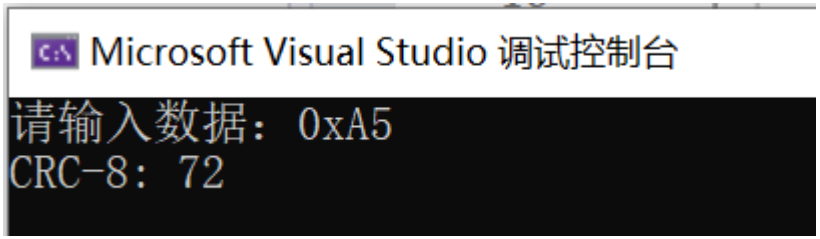
⑧ end

⑨ CRC寄存器就是我们所要求的余数。

CRC参数模型

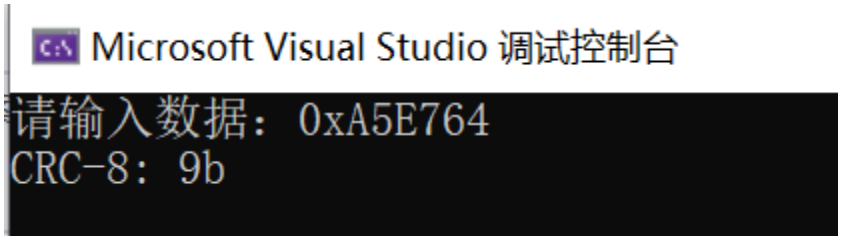
- CRC-8算法C语言实践（单字节输入）：简单
  - 代码：略
  - 示例





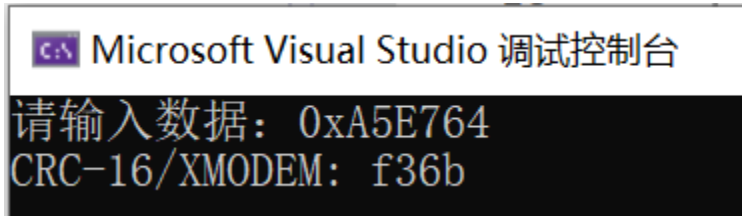
CRC参数模型

- CRC-8算法C语言实践（多字节输入）：中等
  - 代码：略
  - 示例



CRC参数模型

- **CRC-16/XMODEM算法C语言实践（多字节输入）：进阶**
  - **代码：略**
  - **示例**



CRC参数模型

- CRC-16/CCITT算法C语言实践（多字节输入）：进阶
  - 代码：略
  - 示例