BCH编码，解码，纠错

标签： string class 360

2012-09-21 16:11 1946人阅读 评论(0) 收藏 举报

分类： BCH C#（2）

[csharp]

```
class BchClass
{
    //BCH编码
    //把数据从16位变成26位
    public static string  BuildEfficacyCode(uint mx)
    {
        ushort g = 0x5B9;//11位
        ushort register = 0x0000;//定义16位的寄存器,最后数据为10位

        //获取mx的长度
        int mxLength = Convert.ToString(mx, 2).Length;
        //定义i的总数
        int iLength = mxLength + 9;

        /*将mx后面补10个0,从16位变成26位*/
        mx <<= 10;

        for (int i = iLength; i >= 0; i--)
        {
            if (((register >> 10) & 0x0001) == 0x1)
            {
                register = (ushort)(register ^ g);
            }

            /*寄存器慢慢获取值*/

            //寄存器左移动一位
            register <<= 1;
            //信息位从最高位慢慢取数据，给寄存器最低位。
            ushort tmp = (ushort)((mx >> i) & 0x0001);
            register |= tmp;

            /*寄存器慢慢获取值*/

        }

        if (((register >> 10) & 0x0001) == 0x1) register = (ushort)(register ^ g);
        string registerString;
        registerString = Convert.ToString(register, 16).ToUpper();

        return registerString;

    }

    //基础需要校验的数组
```

```
46.     public static uint CorrectSignalBchcode(string receiveData)
47.     {
48.         ushort sn;
49.         sn = CheckBchCode(receiveData);//sn=rx/g
50.
51.
52.         //查询里面的数字，如果有，则把这个数组下标对应的位置和接收数据进行取反，就纠错完成。
53.         ushort[,] snTable;
54.         snTable = new ushort[26, 26]
55.         {
56.             //26行，26列
57.             {119,656,984,892,814,775,463,171,25,64,688,968,884,810,773,462,631,375,247,0,87,103,12
58.             {0,743,328,492,446,407,863,571,649,720,32,344,484,442,405,862,231,999,615,679,0,759,75
59.             {0,0,943,164,246,223,535,883,961,920,360,16,172,242,221,534,431,687,815,1007,911,0,935
60.             {0,0,0,779,82,123,691,983,869,828,460,180,8,86,121,690,267,523,907,843,811,795,0,783,7
61.             {0,0,0,0,857,41,737,901,823,878,414,230,90,4,43,736,345,601,985,793,889,841,849,0,859,
62.             {0,0,0,0,0,880,712,940,798,839,439,207,115,45,2,713,368,624,1008,816,848,864,888,0,0,8
63.             {0,0,0,0,0,0,440,356,470,399,895,519,699,741,714,1,952,184,312,504,408,424,432,444,0,6
64.             {0,0,0,0,0,0,0,220,178,235,539,867,991,897,942,357,732,476,92,156,252,204,212,216,222,
65.             {0,0,0,0,0,0,0,0,110,89,681,977,877,819,796,471,622,366,238,46,78,126,102,106,108,111}
66.             {0,0,0,0,0,0,0,0,0,55,752,904,820,874,837,398,567,311,183,0,23,39,63,51,53,54},//10
67.             {0,0,0,0,0,0,0,0,0,0,711,376,452,410,437,894,199,967,583,647,0,727,719,707,709,710},
68.             {0,0,0,0,0,0,0,0,0,0,0,959,188,226,205,518,447,703,831,1023,927,0,951,955,957,958},
69.             {0,0,0,0,0,0,0,0,0,0,0,0,771,94,113,698,259,515,899,835,803,787,0,0,769,770},
70.             {0,0,0,0,0,0,0,0,0,0,0,0,0,861,47,740,349,605,989,797,893,845,853,0,0,860},
71.             {0,0,0,0,0,0,0,0,0,0,0,0,0,0,882,715,370,626,1010,818,850,866,890,886,0,0},
72.             {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,441,953,185,313,505,409,425,433,445,443,0},
73.             {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,512,768,640,576,544,528,520,516,514,513},
74.             {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,256,384,320,288,272,264,260,258,257},
75.             {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,128,192,160,144,136,132,130,129},
76.             {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,96,80,72,68,66,65},
77.             {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,48,40,36,34,33},
78.             {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,24,20,18,17},
79.             {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,12,10,9},
80.             {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,5},
81.             {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3},
82.             {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}
83.
84.         };
85.
86.         //定位位置
87.         int p = 26;
88.         int q = 26;
89.
90.         //在数组中寻找数据
91.         for (int i = 0; i < 26; i++)
92.         {
93.             for (int j = 25; j >= i; j--)
94.             {
95.                 //如果找到，就进行
96.                 if (snTable[i, j] == sn)
97.                 {
98.                     Console.WriteLine(i + "==" + j);
99.                     p = i + 1;
100.                    q = j + 1;
101.                    break;
102.                }
103.
104.            }
```

```
        }

        //如果找到了数据，进行修改，没有找到数据，则舍弃数据

        //把接收数据由字符串转换为ushort类型
        uint ushortReceiveData = new uint();
        ushortReceiveData = Convert.ToUInt32(receiveData, 2);


        //定义纠正后数据
        uint correctData = new uint();
        if (p == 26 && q == 26)
        {
            //舍弃数据
            sn = 0;
        }
        else
        {
            //把sn中的第P和第Q位数据进行修改。
            if (p == q)
            {
                correctData = (uint)(ushortReceiveData ^ (1 << (26 - p)));
            }
            else
            {
                ushortReceiveData = (uint)(ushortReceiveData ^ (1 << (26 - p)));
                correctData = (uint)(ushortReceiveData ^ (1 << (26 - q)));
            }

        }

        return correctData;
    }

    //效验26位数据是否正确
    public static ushort CheckBchCode(string receiveData)
    {
        ushort g = 0x5B9;//11位
        ushort register = 0x0000;//定义16位的寄存器,最后数据为10位

        //获取mx的长度
        int mxLength = receiveData.Length;
        //定义i的总数
        int iLength = mxLength - 1;//如果总数为26位，则为iLength为25

        //将receiveData转换为mx uint型。
        uint mx;
        mx = Convert.ToUInt32(receiveData, 2);

        for (int i = iLength; i >= 0; i--)
        {
            if (((register >> 10) & 0x0001) == 0x1)
            {
                register = (ushort)(register ^ g);
            }

            /*寄存器慢慢获取值*/

            //寄存器左移动一位
```

```
164.                register <<= 1;
165.                //信息位从最高位慢慢取数据，给寄存器最低位。
166.                ushort tmp = (ushort)((mx >> i) & 0x0001);
167.                register |= tmp;
168.
169.                /*寄存器慢慢获取值*/
170.
171.            }
172.
173.            if (((register >> 10) & 0x0001) == 0x1) register = (ushort)(register ^ g);
174.
175.            return register;
176.        }
177.    }
```