锁定节目命令序列

该定制项目需求如下：有多个节目，根据命令来指定播放的节目。
以下命令序列为例，具体分析该命令序列的组成和修改。

命令序列

当需要锁定节目时，需向控制卡发送如下命令序列（十六进制格式）：

1.需要返回值时的锁定节目命令序列：
a5 a5 a5 a5 a5 a5 a5 a5 fe ff 00 80 07 01 fe ff 00 00 01 00 84 00 96 d5 76 00 00 00 20 00 10 00 7f 54 00 2D 01 a2 0e 06 00 00 01 50 30 30 30 00 ff ff ff ff 5a

2.不需要返回值时的锁定节目命令序列：
a5 a5 a5 a5 a5 a5 a5 a5 fe ff 00 80 07 01 fe ff 00 00 01 00 84 00 96 d5 76 00 00 00 20 00 10 00 BE C4 00 7C 00 a2 0e 06 00 00 01 50 30 30 30 00 ff ff ff ff 5a

节目锁定修改

以“需要返回值时的锁定节目命令序列”为例进行说明
第一步：修改节目编号，红色部分(50 30 30 30) 代表第一个节目编号，若要锁定第二个节目，则应修改为(50 30 30 31)，依次类推
第三步：修改 CRC16 较验，从绿色部分(01)开始到蓝色部分(ff)结束，计算的 CRC16 较验值为深蓝色(7f 54)
第四步：修改异或较验，从红色部分(fe) 代开始到红色部分(00)结束，计算的异或较验值为蓝色字符(2D)

通过以上三步即可完成锁定节目命令的修改

返回值验证

检测返回序列以 A5 开头，最后一个字符为 5A，其前面 8 个字符与序列 4 中绿色标识部分一致，就认为该命令正确执行了。若无返回值或与上述标准不一致则认为该命令没有正确执行，重新发送命令即可。

数据转义

前面讲述了通讯内容，为正确识别帧头和帧尾，通讯前需对上述通讯内容（除去帧头的 8

个A5和帧尾的1个5A）进行转义，具体如下：

封包过程：

a) 遇到0xA5，则将之转义为0xA6，0x02，如遇到0xA6，则将之转义为0xA6，0x01

b) 遇到0x5A，则将之转义为0x5B，0x02，如遇到0x5B，则将之转义为0x5B，0x01

解包过程：

a) 如果遇到连续两个字节为 0xA6, 0x02 ,则反转义为0xA5

b) 如果遇到连续两个字节为 0xA6, 0x01 ,则反转义为 0xA6

对0X5A的解包同0XA5

CRC16校验算法

For the calculation of the CRC-16 the following polynomial is used:

X^16 + X ^15 + X^2 + 1 = (x + 1) * (X^15 + x + 1).

For this polynomial efficient calculation via a table is possible. Below the algorithm is given in C:

```
#define CRC(crc,byte) (((crc) > > 8 ) ^ tabel[((crc) ^ (unsigned int) (byte)) & 0XFF])
```

```
unsigned short tabel[256] = {
  0X0000, 0XC0C1, 0XC181, 0X0140, 0XC301, 0X03C0, 0X0280, 0XC241,
  0XC601, 0X06C0, 0X0780, 0XC741, 0X0500, 0XC5C1, 0XC481, 0X0440,
  0XCC01, 0X0CC0, 0X0D80, 0XCD41, 0X0F00, 0XCFC1, 0XCE81, 0X0E40,
  0X0A00, 0XCAC1, 0XCB81, 0X0B40, 0XC901, 0X09C0, 0X0880, 0XC841,
  0XD801, 0X18C0, 0X1980, 0XD941, 0X1B00, 0XDBC1, 0XDA81, 0X1A40,
  0X1E00, 0XDEC1, 0XDF81, 0X1F40, 0XDD01, 0X1DC0, 0X1C80, 0XDC41,
  0X1400, 0XD4C1, 0XD581, 0X1540, 0XD701, 0X17C0, 0X1680, 0XD641,
0XD201, 0X12C0, 0X1380, 0XD341, 0X1100, 0XD1C1, 0XD081, 0X1040,

0XF001, 0X30C0, 0X3180, 0XF141, 0X3300, 0XF3C1, 0XF281, 0X3240,

0X3600, 0XF6C1, 0XF781, 0X3740, 0XF501, 0X35C0, 0X3480, 0XF441,

0X3C00, 0XFCC1, 0XFD81, 0X3D40, 0XFF01, 0X3FC0, 0X3E80, 0XFE41,

0XFA01, 0X3AC0, 0X3B80, 0XFB41, 0X3900, 0XF9C1, 0XF881, 0X3840,

0X2800, 0XE8C1, 0XE981, 0X2940, 0XEB01, 0X2BC0, 0X2A80, 0XEA41,

0XEE01, 0X2EC0, 0X2F80, 0XEF41, 0X2D00, 0XEDC1, 0XEC81, 0X2C40,

0XE401, 0X24C0, 0X2580, 0XE541, 0X2700, 0XE7C1, 0XE681, 0X2640,

0X2200, 0XE2C1, 0XE381, 0X2340, 0XE101, 0X21C0, 0X2080, 0XE041,

0XA001, 0X60C0, 0X6180, 0XA141, 0X6300, 0XA3C1, 0XA281, 0X6240,

0X6600, 0XA6C1, 0XA781, 0X6740, 0XA501, 0X65C0, 0X6480, 0XA441,

0X6C00, 0XACC1, 0XAD81, 0X6D40, 0XAF01, 0X6FC0, 0X6E80, 0XAE41,

0XAA01, 0X6AC0, 0X6B80, 0XAB41, 0X6900, 0XA9C1, 0XA881, 0X6840,

0X7800, 0XB8C1, 0XB981, 0X7940, 0XBB01, 0X7BC0, 0X7A80, 0XBA41,

0XBE01, 0X7EC0, 0X7F80, 0XBF41, 0X7D00, 0XBDC1, 0XBC81, 0X7C40,

0XB401, 0X74C0, 0X7580, 0XB541, 0X7700, 0XB7C1, 0XB681, 0X7640,

0X7200, 0XB2C1, 0XB381, 0X7340, 0XB101, 0X71C0, 0X7080, 0XB041,

0X5000, 0X90C1, 0X9181, 0X5140, 0X9301, 0X53C0, 0X5280, 0X9241,

0X9601, 0X56C0, 0X5780, 0X9741, 0X5500, 0X95C1, 0X9481, 0X5440,

0X9C01, 0X5CC0, 0X5D80, 0X9D41, 0X5F00, 0X9FC1, 0X9E81, 0X5E40,

0X5A00, 0X9AC1, 0X9B81, 0X5B40, 0X9901, 0X59C0, 0X5880, 0X9841,

0X8801, 0X48C0, 0X4980, 0X8941, 0X4B00, 0X8BC1, 0X8A81, 0X4A40,
   0X4E00, 0X8EC1, 0X8F81, 0X4F40, 0X8D01, 0X4DC0, 0X4C80, 0X8C41,

   0X4400, 0X84C1, 0X8581, 0X4540, 0X8701, 0X47C0, 0X4680, 0X8641,

   0X8201, 0X42C0, 0X4380, 0X8341, 0X4100, 0X81C1, 0X8081, 0X4040

};


unsigned short

CalcCRC(data, size)

char       *data;

int   size;

{

int i;

unsigned short crc = 0;


for (i = 0; i < size; i++) {

crc = CRC(crc, data[i]);

  }

return crc;

}
```