接触 CPU 卡 COS 开发包用户文档

PBOC/石化用户卡 COS

用

户

详

细

手

册

2010.10.7

一、简介

该文档是本人在开发石化标准协议 COS 的详细资料，它是在 PBOC1.0 的基础上修改而来，主要分两个文档（一，COS 开发包用户卡文档，二，COS 开发包 PSAM 卡文档），而本文档分上下两篇，上篇主要讲述该 COS 的开发环境、文件结构、整体流程和详细的每条 COS 指令的详细开发过程，下篇主要讲述每个交易流程（圈存交易、圈提交易、普通消费交易、灰锁、解灰、联机解灰、修改透支限额）、各种密钥和 MAC、TAC 的产生流程

二、开发环境

1.硬件环境

Flash 卡片

CPU:8 位 CPU 内核，指令集兼容标准 51，

FLASH:64

SRAM:2k

真随机数产生器

内置 DES 协处理器，能支持 DES,3DES 运算，

2.软件环境

KEILC2.0

三、文件结构

(1).目录结构

```
struct DF_HEADER
{
  unsigned short FID; //目录文件标识
  unsigned int BODY_ADDR;   //目录体存储地址
  unsigned int EF_KEY_ADDR;   //该目录下密钥地址
  unsigned int EF_ADDR; //该目录下第一个 EF 地址
  unsigned int SON_DF_ADDR; //子 DF 地址
  unsigned int BROTH_DF_ADDR; //兄弟 DF 地址
```

```
  unsigned char read_allow; //该目录的权限
  unsigned char write_allow;//该目录的写权限
  unsigned char SFI;//目录短文件标识
  unsigned char flag; //生命周期 0:正常  1:永久锁定  2：临时锁定 3:被撤消  FF:表示建立未完成
  unsigned char reser_ch[3];//备注
  unsigned char length; //目录体的长度
  unsigned char password_block; //0:正常  1:密码锁定
  unsigned char re_char; //备注
  unsigned short EDC;     //校验
 }; //24              SFI  DIR 文件的短文件标识 01
```

(2).文件结构

```
struct EF_HEADER
 {
  unsigned short FID; //文件标识
  unsigned int BODY_ADDR;//文件体的存放位置
  unsigned char read_allow;//文件的读权限
  unsigned char write_allow;//文件的写权限
  unsigned char length1;   //如果是记录型文件，则表示记录的条数,否则为长度的高字节
  unsigned char length2;   //如果是记录型文件，则表示是每条记录的长度，否则为长度的低字节
  unsigned int EF_ADDR; //下一个文件头的位置
  unsigned char  curr_record;//当前记录号
  unsigned char reser_char[2];//备注
  unsigned char file_type;//文件类型
  unsigned short EDC;//CRC 校验
 }; //16
```

(3).文件类型

B7	B6	B5	B4	B3	B2	B1	B0	说明
X	X	X	X	0	0	0	0	透明二进制文件

X	X	X	X	0	0	0	1	定长记录文件
X	X	X	X	0	0	1	0	变长记录文件
X	X	X	X	0	0	1	1	循环记录文件
X	X	X	X	1	0	0	0	ATR 文件
X	X	X	X	1	0	0	1	密钥文件
X	X	X	X	1	1	0	0	钱包文件
X	X	X	X	1	1	0	1	存折文件
X	X	X	X	1	1	1	0	电子油票文件
X	X	X	0	X	X	X	X	普通 EF 文件
X	X	X	1	X	X	X	X	系统 EF 文件
0	0	X	X	X	X	X	X	明文方式或密文+mac 方式写入数据
0	1	X	X	X	X	X	X	明文+MAC 方式写入数据
1	0	X	X	X	X	X	X	密文方式写入数据
1	1	X	X	X	X	X	X	密文+MAC 方式写入数据

四、系统空间分布

(1).程序区（0-0X9FFF)

程序区大小为：40K BYTES

(2).外部传输密钥区（0xA000-0xA03F）

大小为：64BYTES

(3).系统重要数据区(0xA040-0XA0FF)

大小为：192BYTES

(4).系统重要数据备份区(0xA100-0xA7FF)

大小为：1792BYTES

(5).用户区(0xA800-0xEFFF)

大小为:18K

(6).其他区域（0XF000 以上）

预留为 BOOTLOADER，方便还原成下载状态

五、系统流程

(1) 主守护流程

<1>. 初始化硬件

<2>. 初始化软件（从 FLASH 中读取相关的信息）

<3>. ATR 发送

<4>. 轮询命令的输入

<5>. 命令数据的接收(命令头的接收和命令数据的接收)

<6>. 启动命令处理流程

<7>. 命令响应数据的返回(响应数据的总长度不超过 256Byte 的响应数据和 2Byte 的响应状态码),通常，响应数据缓冲区可以和命令的缓冲区复用同一块 RAM

<8>. 返回 4

(2) 数据 I0 流程（该流程现在采用芯片厂家提供例程操作）

<1>. 命令报文的输入流程(P3 可能对应 Lc,也可能对应 Le,如果 P3 对应的是 LC 的话，命令报文中还包涵数据域)

<2>. 字符的输入流程(接受成功后将顺序保存在命令缓冲区中形成一个完整的命令报文)

<3>. 命令响应报文的输出流程(响应数据__可能不存在和响应状态码)

<4>. ATR 的输出流程

<5>. 字符的输出流程(如果每次发送不成功的话，则连续的发送 3 次，如果还不成功的话，则中断此次发送)

(3) 数据安全写流程

数据安全写的实现基本方法是备份机制。具体过程为:真正想 EEPROM/FLASH 写入数据前将所有数据以及需更新的地址备份起来，然后逐渐开始更新，如果在更新过程中发生意外断点的话，下次卡片上电时可以利用备份区的数据进行恢复操作。

〈1〉. 旧数据备份方式

将要更新的 EEPROM/FLASH 区原有数据进行备份，如果意外中断的话，EEPROM/FLASH 中的数据可以恢复到操作前的状态，相当于相应的交易没有发生。

〈2〉. 新数据备份方式

将要更新的数据和地址先集中写入备份区，如果意外中断的话，EEPROM/FLASH 中的数据可以在下次上电过程中得到更新。

具体操作步骤:

<1>. 数据备份阶段

<2>. 数据的实际更新阶段

<3>. 安全写的恢复阶段

设计例子:

<1>. 环境初始化(将备份区清空，数据长度计算变量置为初始值，并且将备份标志复位)

<2>. 旧数据备份

<3>. 备份标志置位

<4>. 实际数据更新

<5>. 备份标志复位

六、详细阐述

1. 主流程

2. 初始化 COS 命令

<1>建立 MF(80e00100+LEN)

命令：

create 命令报文

代码	值
CLA	‘80’
INS	‘E0’
P1	01:建立 MF　02:建立 DF　03:建立 EF
P2	'00'
Lc	如果是 MF,DF，长度为 0x0a-0x15,如果是 EF，则长度为 7
Data	数据项
Le	‘00’

80e00100Length(1)+FILE_ID(2)+短文件标识(1)+建立文件权限（1）+删除 MF 权限(1)+MF 名称(05-0x10)

详细：

1.判断文件头及文件体是否在争取，不是则提示 6581

2.判断 P1,P2 是否正确，不是则提示 6A86

3.判断 CLA 是否为 80，不是则提示 6E00，回送 INS

4.判断长度是否在 0A-15 之间，不是则提示 6700

5.判断目录标识是否是 3F00，不是则提示 6581

6.搜索是否存在目录标识为 3F00，如果存在，则提示 6A82

7.判断文件体的空间是否能够保存下文件体，如不能保存，则提示 6A84

8.充实目录文件结构，保存目录文件，返回 9000

详细代码：

```
 if(PA2!=00)
    return 0x6A86;
 if((LEN<0x0a)||(LEN>0x15))
   return 0x6700;
 if((DAT[0]!=0x3F)&&(DAT[1]!=0x00))
    return 0x6581;
 i=Search(0x3F00,0,NULL);
 if((i>=User_Area_Base)&&(i<=(User_Area_Base+User_Area_Size)))
    return 0x6A82;
 length=LEN-5;
 if((length>(Body_offset-User_Area_Base))
||((Body_offset-Header_offset-length)<sizeof(struct DF_HEADER)))
    return 0x6A84;
 memset(&MF,0,sizeof(struct DF_HEADER));
 MF.FID=0x3F00;
 MF.length=length;
 MF.BODY_ADDR=Body_offset-length;
 MF.EF_KEY_ADDR=0xFFFF;   //EF_KEY
 MF.SON_DF_ADDR=0xFFFF;   //子 DF
 MF.BROTH_DF_ADDR=0xFFFF; //兄弟 DF
 MF.EF_ADDR=0xFFFF;
 MF.read_allow=DAT[3]; //建立文件权限
 MF.write_allow=DAT[4];//删除文件权限
 MF.SFI=DAT[2];
 MF.flag=0x00;
 Body_offset=Body_offset-length;
```

```
   if(writeFlash_source(6,8,length,DAT+5,(Body_offset),0)!=0) //不需要保存
     return 0x6581;
   if(writeFlash_source(6,8,sizeof(struct DF_HEADER),(unsigned char
*)&MF,(Header_offset),0)!=0) //不需要保存
     return 0x6581;
   Header_offset+=sizeof(struct DF_HEADER);

   Ptr_MF=User_Area_Base;
   Parent_DF=User_Area_Base;
   Current_DF=User_Area_Base;
   Current_Allow_Read=DAT[3];  //建立
   Current_Allow_Write=DAT[4]; //删除



   gCommBuf[0]=(Header_offset>>8)&0x00ff;
   gCommBuf[1]=(Header_offset&0x00ff);
   gCommBuf[2]=(Body_offset>>8)&0x00ff;
   gCommBuf[3]=(Body_offset&0x00ff);

 if(writeFlash_source(6,8,4,gCommBuf,(unsigned char xdata *)(System_Data_Base+2),0)!=0)
return 0x6581;
return 0x9000;
```

<2>建立目录(80e00200+LEN)

命令：

80e00200+数据长度(1)+文件标识符(2)+文件类型(1,38)+RFU+建立文件权限(1)+擦除权限(1)+DF 名称(05--0X10)

详细：

1.判断文件头及文件体是否在争取，不是则提示 6581

2.判断 P1,P2 是否正确，不是则提示 6A86

3.判断 CLA 是否为 80，不是则提示 6E00，回送 INS

4.判断长度是否在 0A-15 之间，不是则提示 6700

5.判断目录标识是否小于 3F00，如果小于 3F 则提示 6581（建立的目录必须大于 3F00）

6.搜索是否存在此目录标识，如果存在，则提示 6A82

7.判断文件体的空间是否能够保存下文件体，如不能保存，则提示 6A84

8.充实目录文件结构，保存目录文件

9.预留 320BYTES 的空间，为目录下的交易作备用空间，返回 9000

详细代码：

```
    if(PA2!=00)
      return 0x6A86;
    if((LEN<0x0a)||(LEN>0x15))
      return 0x6700;
    if(DAT[0]<0x3F)
      return 6581;
    file_id=0;
    file_id=DAT[0];
    file_id<<=8;
    file_id+=DAT[1];
   i=Search(file_id,0,NULL);
   if((i>User_Area_Base)&&(i<(User_Area_Base+User_Area_Size)))
     return 0x6A82;
    length=Current_DF;
    while(1)
     {
      memcpy((char *)&MF,(unsigned char xdata *)(length),sizeof(struct DF_HEADER));
       if(length==Current_DF)
        {
       if((MF.SON_DF_ADDR>User_Area_Base)&&(MF.SON_DF_ADDR<=(User_Area_Base+User_Area_Size)))
          {
          length=MF.SON_DF_ADDR;
           continue;
          }
        }
       else

{      if((MF.BROTH_DF_ADDR>User_Area_Base)&&(MF.BROTH_DF_ADDR<=(User_Area_Base+User_Area_Siz
e)))
         {
          length=MF.BROTH_DF_ADDR;
          continue;
         }
        }
      break;
     }
    flag=0;
    flag=(LEN-5);
    if((flag>(Body_offset-User_Area_Base))  ||((Body_offset-Header_offset-flag)<sizeof(struct
DF_HEADER)))
      return 0x6A84;
   memset((char *)&DF,0,sizeof(struct DF_HEADER));
    DF.FID=DAT[0];
    DF.FID<<=8;
```

```
   DF.FID+=DAT[1];
   DF.length=flag;
   DF.BODY_ADDR=(Body_offset-flag);
   DF.read_allow=DAT[3]; //建立文件权限
   DF.write_allow=DAT[4];//删除文件权限
   DF.SFI=DAT[2];
   DF.flag=0x00;
   if(length==Current_DF)  //当前 DF 就是目前的最后的 DF
    {
     Parent_DF=Current_DF;
     MF.SON_DF_ADDR=Header_offset;
    }
     else
    {
     MF.BROTH_DF_ADDR=Header_offset;
    }
   if(writeFlash_source(6,8,sizeof(struct DF_HEADER),(unsigned char *)&MF,(length),0)!=0) //
不需要保存
     return 0x6581;
   Body_offset=(Body_offset-LEN+5);
   if(writeFlash_source(6,8,LEN-5,DAT+5,(Body_offset),0)!=0) //不需要保存
     return 0x6581;
   if(writeFlash_source(6,8,sizeof(struct              DF_HEADER),(unsigned              char
*)&DF,(Header_offset),0)!=0) //不需要保存
     return 0x6581;

   Current_Allow_Read=DAT[3];  //建立
   Current_Allow_Write=DAT[4]; //删除
   Current_DF=Header_offset;
   Header_offset+=sizeof(struct DF_HEADER);
  //对 header_offset 进行判断,如果

  kk=Header_offset%BPS;
  if(kk!=0)
   kk=BPS-kk;
  Header_offset+=kk;

  gCommBuf[0]=0x00;
  if(writeFlash_source(0x84,8,Application_Size,gCommBuf,(Header_offset),0)!=0) //不需要保存
     return 0x6581;

   Header_offset+=Application_Size;  //需要预留 320 多个空间

   gCommBuf[0]=(Header_offset>>8)&0x00ff;
```

```
gCommBuf[1]=(Header_offset&0x00ff);
gCommBuf[2]=(Body_offset>>8)&0x00ff;
gCommBuf[3]=(Body_offset&0x00ff);
if(writeFlash_source(6,8,4,gCommBuf,(System_Data_Base+2),0)!=0) //不需要保存
  return 0x6581;
return 0x9000;
```

<3>建立文件(80e00300+LEN)

80e00300+数据长度(1,08)+文件标识符(2)+文件类型(1)+权限 1(1)+权限 2(1)+长度 1(1)+长度 2(2)

1.判断文件头及文件体是否在争取，不是则提示 6581

2.判断 P1,P2 是否正确，不是则提示 6A86

3.判断 CLA 是否为 80，不是则提示 6E00

4.判断长度是 7，不是则提示 6700，回送 INS

5.搜索当前目录下是否存在此文件标识，如果存在，则提示 6A82

6.判断需要建立的文件类型：

如果为 9（密钥文件，则把读的权限设置为禁用，长度 1 为记录数，长度 2 为每条记录的大小，文件体的大小为：长度 1X 长度 2）

如果为 1，3（定长记录或循环记录，长度 1 为记录数，长度 2 为每条记录的大小，文件体的大小为：长度 1X（长度 2+4），每条记录增加一个 4BYTES 的序号）

如果是电子油票，电子钱包，电子存在，则 COS 内部处理成循环文件结构，这样每次进行交易处理时不同时写一块区域，避免 FLASH 写坏）

如果是其他类型（二进制文件，则文件体的大小为:长度 1*256+长度 2）

7.判断文件体的空间是否能够保存下文件体，如不能保存，则提示 6A84

8.充实文件结构，保存文件，返回 9000

<3>.建立完成(0044000000)

运行完毕此命令后，表示卡片建立完成

流程：

1.判断 CLA，如果不是 00，则返回 6E00

2.判断 P1,P2,如果不正确则返回 6A86

3.清除建立标志，返回 9000

<4>.删除卡片(00E4000000)

运行完毕此命令后，卡片的数据区被清空

流程：

1.判断 CLA，如果不是 00，则返回 6E00

2.判断 P1,P2,如果不正确则返回 6A86

3.判断权限，如果不符合清除卡片权限，则返回 6985

4.初始化各种变量

5.清 FLASH 的数据区，返回 9000

3. 基本操作 COS 命令

<1>.选择文件(00a404/0000+LEN)

路程：

1.判断 CLA 是否为 00，不为，则返回 6E00

2.判断 P1,P2,不符合则返回 6A86

3.判断长度，不符合则返回 6700

4.回送 INS

5.如果 P2 是 04，则表示通过名称来选择目录

如果是 00，则表示通过文件标识的方式来选择

6.搜索当前的文件系统，如果没有找到合适的，则返回 6A82

7.如果是选择的目录

（1）把当前的读写权限更新成选择目录的读写权限

（2）把当前的密钥文件改变成选择目录的密钥文件

（3）搜索当前目录下的电子钱包，电子存折，电子油票

SELECT 命令报文

代码	值
CLA	‘00’
INS	‘A4’
P1	引用控制参数（下表）
P2	‘00’第一个或仅有一个 ‘02’下一个
Lc	‘05’－ ‘10’

Data	文件名
Le	‘00’

SELECT 命令引用控制参数

b8	b7	b6	b5	b4	b3	b2	b1	含　义
0	0	0	0	0				
					1			通过文件名选择
						0	0	

8.返回数据

响应报文中数据域应包括所选择的 PSE、DDF 或 ADF 的 FCI。表 35 到表 37 规定了此定义所用的标志。不规定 FCI 中回送的附加标志。

SELECT PSE 的响应报文（FCI）

标志	值	存在方式
‘6F’	FCI 模板	M
‘84’	DF 名	M
‘A5’	FCI 专用数据	M
‘88’	目录基本文件的 SFI	M

表 36　　SELECT DDF 的响应报文（FCI）

标志	值	存在方式
‘6F’	FCI 模板	M
‘84’	DF 名	M
‘A5’	FCI 专用数据	M
‘88’	目录基本文件的 SFI	M

SELECT ADF 的响应报文（FCI）

标志	值	存在方式
‘6F’	FCI 模板	M
‘84’	DF 名	M
‘A5’	FCI 专用数据	M
‘9F0C’	发卡方自定数据的 FCI	O

<2>读二进制文件

READ BINARY 命令报文

代码	值
CLA	‘00’或‘04’
INS	‘B0’
P1	见表 26
P2	从文件中读取的第一个字节的偏移地址
Lc	不存在；（CLA＝‘04’时除外）
Data	不存在；（CLA＝‘04’时，应包括 MAC）
Le	‘00’

READ BINARY 命令引用控制参数

B8	b7	b6	b5	b4	b3	b2	b1	含　义
X								读取模式：
1								－用 SFI 方式

	0	0						RFU（如果 b8＝1）
			X	X	X	X	X	SFI（取值范围 21－30）

流程：

1.判断 CLA，不为 00 或 04，则返回 6E00

2.PA1 最高位不为 1，则返回 6A86

3.搜索此文件标识，如果不存在则返回 6A82

4.发现此文件不为二进制文件，则返回 6A82

5.如果文件长度小于读取的长度，则返回 6CXX，XX 表示文件的实际长度

6.如果读取的长度+P2（偏移地址）大于文件长度，则返回 6700

7.查询文件的读取权限，如果当前权限小于文件的读取权限，则返回 6985

8.如果读取的长度为 0，则把文件的内容复制到响应缓冲区中，并返回 61XX，XX 表示文件的实际长度

9.回送 INS，发送文件内容，返回 9000

<3>写二进制文件

UPDATE BINARY 命令报文

代码	值
CLA	‘00/04’
INS	‘D6’
P1	见表 41
P2	要修改的第一个字节的偏移地址
Lc	后续数据域的长度
Data	修改用的数据
Le	不存在

UPDATE BINARY 命令引用控制参数

b8	b7	b6	b5	b4	b3	b2	b1	含　义

X 1								读取模式： －用 SFI 方式
	0	0						RFU（如果 b8＝1）
			X	X	X	X	X	SFI（取值范围 21－30）

流程：

1.判断 CLA 是否不为 00 或 04，不是的话则返回 6E00

2.如果 PA1 最高位不为 1 的话，则返回 6A86

3.搜索此文件标识的文件，如果没搜索到，则返回 6A82

4.返回 INS

5.判断该文件是否为二进制文件，不为，则返回 6A82

6.如果该目录为没锁定目录，则返回 6902

7.查询文件的读取权限，如果当前权限小于文件的读取权限，则返回 6985

8.如果 CLA 为 04，

搜索密钥类型为 1,密钥标识为 0 的密钥，如果没搜索到，则返回 9403

如果发现密钥已经被锁死，则返回 6983

如果发现没有合适的随机数，则返回 6984

文件类型为更新时需要进行 MAC 校验的，则用查找到的密钥对随机数，及数据进行 MAC 校验

MAC 校验失败时，更新密钥的 MAC 校验失败次数（次数加 1），返回 6988

MAC 校验成功，则更新密钥的 MAC 交易成功次数（失败次数清零）

文件类型为更新时需要进行解密的，则需要用查找到的密钥对输入的数据进行解密

解密失败，则返回 6A82

9.如果 CLA 为 0，如果文件类型需要进行 MAC 校验，则进行以上操作，如果文件类型需要对输入数据进行解密，则返回 6A80

9.更新文件，返回 9000

<4>读记录文件

READ RECORD 命令报文

代码	值

CLA	‘00’或‘04’
INS	‘B2’
P1	记录的序列号
P2	引用控制参数（见表 30）
Lc	不存在；（CLA＝‘04’时除外）
Data	不存在；（CLA＝‘04’时除外）
Le	‘00’

READ RECORD 命令引用控制参数

b8	b7	b6	b5	B4	b3	b2	b1	含　义
X	X	X	X	X				SFI
					1	0	0	P1 为记录的序列号

流程：

1.判断 CLA 是否不为 00 或 04，不是的话则返回 6E00

2.如果 PA2 第 3 位不为 1 的话，则返回 6A86

3.搜索此文件标识的文件，如果没搜索到，则返回 6A82

4.返回 INS

5.判断该文件是否为循环记录文件、定长文件、不定长文件，不为，则返回 6A82

6.判断当前的读权限是否大于该文件的读权限，小于的话则返回 6985

7.判断当前目录的状态，如果为锁定，则返回 6901

8.如果是循环文件或定长文件

如果数据长度不是循环文件或定长文件每条记录的长度，则返回 6CXX，XX 为该文件的每条记录长度

如果该文件的记录数小于 PA1,则返回 6B00

如果是定长文件，则直接返回该笔记录的数据，然后返回 9000

搜索循环文件的记录，找到倒数第 PA1 笔数据，返回该笔数据后再返回 9000

9.如果是不定长文件

查找 PA1 号记录，如果查找不到，则返回 6B00

返回该笔记录，再返回 9000

<5>写记录文件

UPDATE RECORD 命令报文

代码	值
CLA	‘00’
INS	‘DC’
P1	P1＝‘00’表示当前记录 P1≠‘00’指定的记录号
P2	见表 45
Lc	后续数据域的长度
Data	更新原有记录的新记录
Le	不存在

UPDATE RECORD 命令引用控制参数

b8	b7	b6	b5	b4	b3	b2	b1	含　义
X	X	X	X	X				SFI
					0	0	0	第一个记录
					0	0	1	最后一个记录
					0	1	0	下一个记录
					0	1	1	上一个记录
					1	0	0	记录号在 P1 中给出
其余值								RFU

流程：

1.判断 CLA 是否不为 00 或 04，不是的话则返回 6E00

2.如果 PA2 第 3 位不为 1 的话，则返回 6A86

3.搜索此文件标识的文件，如果没搜索到，则返回 6A82

4.返回 INS

5.如果该目录被锁定，则返回 6901

6.判断当前的读权限是否大于该文件的读权限，小于的话则返回 6985

7.判断该文件是否为循环记录文件、定长文件、不定长文件，不为，则返回 6A82

8.如果 CLA 为 04，则查找密钥类型为 01,密钥标识为 00 的密钥，没查找到，则返回 9403

如果发现密钥已经被锁死，则返回 6983

如果发现没有合适的随机数，则返回 6984

文件类型为更新时需要进行 MAC 校验的，则用查找到的密钥对随机数，及数据进行 MAC 校验

MAC 校验失败时，更新密钥的 MAC 校验失败次数（次数加 1），返回 6988

MAC 校验成功，则更新密钥的 MAC 交易成功次数（失败次数清零）

文件类型为更新时需要进行解密的，则需要用查找到的密钥对输入的数据进行解密

解密失败，则返回 6A82

9.如果 CLA 为 0，如果文件类型需要进行 MAC 校验，则进行以上操作，如果文件类型需要对输入数据进行解密，则返回 6A80

10.如果是循环文件或定长文件

如果数据长度不是循环文件或定长文件每条记录的长度，则返回 6700

如果该文件的记录数小于 PA1,则返回 6B00

如果是定长文件，则直接写记录，然后返回 9000

搜索循环文件的记录，找到倒数第 PA1 笔记录，写记录再返回 9000

11.如果是不定长文件

查找 PA1 号记录，如果查找不到，则返回 6B00

更新 PA1 号记录，返回 9000

<6>校验 PIN

VERIFY 命令报文

代码	值
CLA	‘00’
INS	‘20’
P1	‘00’
P2	‘00’
Lc	可变

Data	外部输入的个人密码
Le	不存在

P2＝‘00’表示无特殊限定符被使用。在 IC 卡上，VERIFY 命令在处理过程中应明确知道如何去寻找个人密码。

流程：

1.判断 CLA 是否不为 00，不是的话则返回 6E00

2.如果 PA1，PA2，不正确则返回 6A86

3.判断当前目录是否被锁定，被锁定则返回 6983

4.返回 INS

5.搜索密码，没找到，则返回 9403

6.匹配密码，成功，则更改当前权限，恢复密码校验失败次数，返回 9000

7.增加密码校验失败次数，

没超过指定密码校验次数，则返回 63CX，X 表示还剩余的校验次数

如果超过指定密码校验次数，则被密码锁定，不能进行密码校验，需重新进行密码解锁，返回 6983

<7>密码解锁/更改密码

PIN CHANGE / UNBLOCK 命令报文

代码	值
CLA	‘84’；根据本规范“安全机制”部分的规定进行编码
INS	‘24’
P1	‘00’
P2	‘00’或’01’
Lc	数据字节数
Data	加密的个人密码数据元和报文鉴别代码（MAC）数据元,根据本规范“安全机制”部分的规定进行编码
Le	不存在

P2＝‘00’，表示解锁个人密码。此时应重置尝试计数器，但不更改个人密码。

P2＝‘01’，表示更改个人密码。此时应重置尝试计数器，并以一个新的个人密码取代原有个人密码。

当 P2＝‘00’时，Lc 应包括 MAC 数据元的长度。

当 P2＝‘01’时，Lc 应同时包括个人密码数据元和 MAC 数据元的长度。

流程：

1.判断 CLA 是否不为 84，不是的话则返回 6E00

2.如果 PA1，PA2，不正确则返回 6A86

3.判断 LEN 是否正确，不正确则返回 6700

4.判断是否取随机数，没取，则返回 6984

5.返回 INS

6.搜索密钥类型为 03，密钥标识为 00 的密钥，没搜索到则返回 9403

7.判断该密钥是否被锁死，如果被锁死，则返回 6A81

8.用该密钥对随机数进行计算 MAC

9.判断计算的 MAC 和输入的 MAC 是否一致

不一致，则增加 MAC 校验失败次数，返回 6988

一致，则清零 MAC 交易失败次数，恢复密码状态（如果 P2 是 00,则个人密码为原密码，如果 P2 为 01，则密码为输入的密码），返回 9000

<8> 内部认证

INTERNAL AUTHENTICATION 命令报文

代码	值
CLA	‘00’
INS	‘88’
P1	‘00’
P2	‘00’
Lc	认证数据的长度
Data	认证数据

Le	‘00’

INTERNAL AUTHENTICATION 命令的参数 P1 为‘00’时的含义是无信息。P1 的值可事先得到，也可以在数据域中提供。

流程：

1.判断 CLA 是否不为 00，不是的话则返回 6E00

2.如果 PA1，PA2，不正确则返回 6A86

3.判断输入数据长度，不为 8 或 16，则返回 6700

4.输出 INS

5.搜索密钥类型为 09，密钥标识为 00 的密钥，没找到则返回 9403

6.用该密钥对输入数据进行 3DES 加密

7.把加密后的密文拷贝到输出数据缓冲区，返回 61XX，XX 加密后的数据长度

<9>.装载/更新密钥

流程：

1.判断 CLA，不符合则返回 6E00

2.判断 PA1,PA2,不符合则返回 6A86

3.搜索密钥文件，如果不存在则返回 6583

4.输出 INS

5.判断当前目录是否被锁死，如果锁死，则返回 6901

6.如果 CLA 为 84，

搜索密钥文件中的主控密钥（密钥类型为 0，密钥标识为 0）的密钥，没找到则返回 9403

如果需要校验 MAC，则用搜索到的密钥进行 MAC 计算，判断是否一致，不一致返回 6988

用该密钥对输入数据进行解密

7.搜索密钥文件中此密钥类型和密钥标识的密钥，如果没找到且密钥已经存满，则返回 6986

如果该密钥存在，则判断当前权限是否能更新，不能更新则返回 6985

如果不存在，则判断当前权限是否满足追加密钥，不能则返回 6985

8.更新密钥，返回 9000

<10>.应用锁定

APPLICATION BLOCK 命令报文

代码	值
CLA	’84’
INS	‘1E’
P1	‘00’，其他值保留为将来使用
P2	‘00’或‘01’
Lc	数据字节数
Data	报文鉴别代码（MAC）数据元；根据本规范“安全机制”部分的规定进行编码
Le	不存在

流程：

1.判断 CLA，不符合则返回 6E00

2.判断 PA1,PA2,不符合则返回 6A86

3.判断输入数据长度，不符合则返回 6700

4.判断是否取随机数，没取则返回 6984

5.输出 INS

6.搜索密钥类型为 01，密钥标识为 00 的密钥，没搜索到则返回 9403

7.判断该密钥是否锁死，锁死则返回 6A81

8.用该密钥对随机数进行 MAC 计算

9.判断计算的 MAC 和输入的 MAC，不一致则增加失败次数，返回 6988

10.清零失败次数，该应用锁定，返回 9000

<11>.应用解锁

APPLICATION UNBLOCK 命令报文

代码	值
CLA	‘84’
INS	‘18’
P1	‘00’，其他值保留为将来使用

P2	‘00’，其他值保留为将来使用
Lc	数据字节数
Data	报文鉴别代码（MAC）数据元；根据本规范“安全机制”部分的规定进行编码
Le	不存在

流程：

1.判断 CLA，不符合则返回 6E00

2.判断 PA1,PA2,不符合则返回 6A86

3.判断输入数据长度，不符合则返回 6700

4.判断当前目录是否锁死，如果是永久锁定，则返回 9303

5.判断是否取随机数，没取则返回 6984

6.输出 INS

7.搜索密钥类型为 01，密钥标识为 00 的密钥，没搜索到则返回 9403

8.判断该密钥是否锁死，锁死则返回 6A81

9.用该密钥对随机数进行 MAC 计算

10.判断计算的 MAC 和输入的 MAC，不一致则增加失败次数，返回 6988

11.清零失败次数，该应用解锁，返回 9000

<12>.锁卡（永久锁定）

CARD BLOCK 命令报文

代码	值
CLA	‘84’
INS	‘16’
P1	‘00’，其他值保留为将来使用
P2	‘00’，其他值保留为将来使用
Lc	数据字节数
Data	报文鉴别代码（MAC）数据元；根据本规范“安全机制”部分的规定进行编码
Le	不存在

流程：

1.判断 CLA，不符合则返回 6E00

2.判断 PA1,PA2,不符合则返回 6A86

3.判断输入数据长度，不符合则返回 6700

4.判断是否取随机数，没取则返回 6984

5.输出 INS

6.搜索密钥类型为 01，密钥标识为 00 的密钥，没搜索到则返回 9403

7.判断该密钥是否锁死，锁死则返回 6A81

8.用该密钥对随机数进行 MAC 计算

9.判断计算的 MAC 和输入的 MAC，不一致则增加失败次数，返回 6988

10.清零失败次数，该卡片永久锁定，返回 9000

<13>.外部认证

EXTERNAL AUTHENTICATION 命令报文

代码	值
CLA	‘00’
INS	‘82’
P1	‘00’
P2	‘00’
Lc	8－16
Data	发卡方认证数据
Le	不存在

流程：

1.判断 CLA，不符合则返回 6E00

2.判断 PA1,PA2,不符合则返回 6A86

3.判断输入数据长度，不符合则返回 6700

4.判断是否取随机数，没取则返回 6984

5.输出 INS

6.搜索密钥类型为 00，密钥标识为 00 的密钥，没搜索到则返回 9403

7.判断该密钥是否锁死，锁死则返回 6A81

8.用该密钥对随机数进行加密计算

9.判断计算的数据和输入的数据，不一致则增加失败次数，返回 9302

10.清零失败次数，更新当前权限，返回 9000

<14>.取随机数

GET CHALLENGE 命令报文

代码	值
CLA	‘00’
INS	‘84’
P1	‘00’
P2	‘00’
Lc	不存在
Data	不存在
Le	‘04’

详细代码：

```
if((LEN!=4)&&(LEN!=8)&&(LEN!=16))
  return 0x6700;
writeINS_backup();
length=LEN;
gCommBuf[0]=LEN;
getRand(length,gCommBuf+1);
memcpy(rand_buf,gCommBuf,length+1);
writeUART(gCommBuf+1,length);
return 0x9000;
```

<15>.取响应数据

GET RESPONSE 命令报文

代码	值
CLA	‘00’
INS	‘C0’
P1	‘00’
P2	‘00’
Lc	不存在

Data	不存在
Le	响应的期望数据最大长度

<16>.修改个人密码

CHANGE PIN 命令报文

代码	值
CLA	‘80’
INS	‘5E’
P1	‘01’
P2	‘00’
L_c	‘05’－ ‘0D’
Data	当前 PIN||‘FF’||新的 PIN
L_e	不用

流程：

1.判断 CLA，不符合则返回 6E00

2.判断 PA1,PA2,不符合则返回 6A86

3.判断输入数据长度，不符合则返回 6700

4.输出 INS

5.查找个人密码，如果没找到则返回 9403

6.如果输入旧密码同实际的密码不一致，则增加失败次数，如果还没超过规定失败次数，则返回 63CX，X 表示还剩余的校验次数，如果超过则返回 6983，密码锁定

7.更新密码，更新当前权限，返回 9000

<17>.圈存

CREDIT FOR LOAD 命令报文

代码	值
CLA	‘80’

INS	‘52’
P1	‘00’
P2	‘00’
Lc	‘0B’
Data	见表 60
Le	‘04’

命令数据：

CREDIT FOR LOAD 命令报文数据域

说明	长度（字节）
交易日期（主机）	4
交易时间（主机）	3
MAC2	4

返回数据：

CREDIT FOR LOAD 命响应报文数据域

说明	长度（字节）
TAC	4

流程：

1.判断 CLA，不符合则返回 6E00

2.判断 PA1,PA2,不符合则返回 6A86

3.判断状态机，没有执行圈存初始化，则返回 6985

4.判断输入数据长度，不符合则返回 6700

5.输出 INS

6.搜索电子油票/电子钱包/电子存折，如果不存在则返回 6A82

7.如果类型不对，则返回 6A80

8.搜索密钥类型为 06,密钥标识为指定标识的密钥(圈存密钥),如果不存在,则返回 9403

9.用圈存初始化产生的临时密钥对相关数据计算 MAC2

交易金额+交易类型标识+终端机编号+交易日期+交易时间

10.比较 MAC2，不一致则返回 9302

11.计算 TAC

12.查找 IC 卡交易明细文件，没找到则返回 6581

13.更新 IC 卡交易明细文件，电子油票/电子钱包/电子存折记录

14.更新状态机为空闲状态，把 TAC 写回响应缓冲区中，返回 6104

<18>.消费

DEBIT FOR PURCHASE 命令报文

代码	值
CLA	’80’
INS	’54’
P1	’01’
P2	’00’
Lc	‘0F’
Data	见表 64
Le	‘08’

DEBIT FOR PURCHASE 命令报文数据域

说明	长度
终端交易序号	4
交易日期（终端）	4
交易时间（终端）	3
MAC1	4

DEBIT FOR PURCHASE 响应报文数据域

说明	长度(字节)
TAC	4
MAC2	4

流程：

1.判断 CLA，不符合则返回 6E00

2.判断 PA1,PA2,不符合则返回 6A86

3.判断状态机，没有执行消费初始化，则返回 6985

4.判断输入数据长度，不符合则返回 6700

5.输出 INS

6.查找密钥类型为 02,密钥标识为指定标识的密钥，没找到则返回 9403

7.用该密钥对以下数据进行计算得到临时密钥

伪随机数+电子油票脱机交易序号+"8000"

8.用临时密钥对以下数据进行计算得到 MAC：

交易金额+交易类型标识+终端机编号+交易日期+交易时间

9.比较 MAC，不一致则返回 9302

10.用临时密钥对交易金额计算得到 MAC2

11.搜索密钥类型为 06,密钥标识为指定标识的密钥，没找到则返回 9403

12.该密钥异或后对以下数据计算后得到 TAC

交易金额+交易类型标识+终端机编号+终端交易序号+日期时间

13.查找 IC 卡交易明细文件，如果没找到，则返回 6581

14.写 IC 卡交易明细记录，操作电子油票/电子钱包/电子存折，写 MAC2 和 TAC 到响应缓冲区，返回 6108

<19>.圈提

DEBIT FOR UNLOAD 命令报文

代码	值
CLA	’80’
INS	’54’
P1	’03’
P2	’00’
Lc	‘0B’
Data	见表 68
Le	‘04’

DEBIT FOR UNLOAD 命令报文数据域

说明	长度(字节)
交易日期(主机)	4
交易时间(主机)	3
MAC2	4

DEBIT FOR UNLOAD 响应报文数据域

说明	长度(字节)
MAC3	4

流程：

1.判断 CLA，不符合则返回 6E00

2.判断 PA1,PA2,不符合则返回 6A86

3.判断状态机，没有执行圈提初始化，则返回 6985

4.判断输入数据长度，不符合则返回 6700

5.输出 INS

6.用圈提初始化产生的密钥对以下数据进行计算得到 MAC2

交易金额+交易类型标识+终端机编号+交易日期+交易时间

7.比较 MAC2 是否一致，不一致则返回 9302

8.判断余额与圈提额的大小，如果余额小于圈提额，则返回 9401

9.用圈提初始化产生的密钥对以下数据进行计算得到 MAC3

余额+联机交易序号+交易金额+交易类型标识+终端机编号+日期时间

10.查找 IC 卡交易明细文件，没找到则返回 6581

11.写 IC 卡交易明细记录，操作电子油票/电子钱包/电子存折，写 MAC3 到响应缓冲区，返回 6104

<20>.解扣

DEBIT FOR UNLOCK 命令报文

代码	值
CLA	‘E0’
INS	‘7E’
P1	‘08’
P2	‘01’ 其他值保留
L_c	‘1B’
Data	见表 72
L_e	‘04’

DEBIT FOR UNLOCK 命令报文数据域

说明	长度（字节）
交易金额	4
ET 脱机交易序号	2
终端机编号	6
终端交易序号	4
交易日期（终端）	4
交易时间（终端）	3
GMAC	4

DEBIT FOR UNLOCK 响应报文数据域

说明	长度（字节）
TAC	4

流程：

1.判断 CLA，不符合则返回 6E00

2.判断 PA1,PA2,不符合则返回 6A86

3.判断状态机，没有执行灰锁，则返回 6985

4.判断输入数据长度，不符合则返回 6700

5.输出 INS

6.判断当前状态，如果没有灰锁，则返回 6985

7.判断脱机交易序号是否一致，不一致，则返回 9406

8.判断电子油票余额是否小于交易金额，如果小于，则返回 9401

9.用灰锁产生的临时密钥对以下数据计算得到 GMAC

交易金额

10.比较 GMAC，如果不一致，则增加失败次数，返回 9302

11.搜索密钥类型为 06,密钥标识为指定标识的密钥，如没找到，则返回 9403

12.用此密钥进行异或操作后对以下数据进行计算得到 TAC

交易金额+交易类型标识+终端机编号+终端机交易序号+日期时间

13.查找内部交易明细表，没找到则返回 6A82

14.查找 IC 卡交易明细表，没找到则返回 6A82

15.写 IC 卡交易明细记录，内部交易明细表，操作电子油票/电子钱包/电子存折，清状态机，写 TAC 到响应缓冲区，返回 6104

<21>.读余额

GET BALANCE 命令报文

代码	值
CLA	‘80’
INS	‘5C’
P1	‘00’
P2	‘01’ ,其它 RFU
L_c	不存在
Data	不存在
L_e	‘04’

GET BALANCE 响应报文数据域

说明	长度（字节）
ET 余额	4

流程：

1.判断 CLA，不符合则返回 6E00

2.判断 PA1,PA2,不符合则返回 6A86

3.判断状态机，没有执行灰锁，则返回 6985

4.判断输入数据长度，不符合则返回 6700

5.判断是否通过密码校验，没有则返回 6985

6.搜索电子油票/电子存折/电子钱包，没找到则返回 6A82

7.直接写余额，返回 9000

<22>.取灰状态

GET LOCK PROOF 命令报文

代码	值
CLA	‘E0’
INS	‘CA’
P1	‘00’：普通读取； ‘01’：清除 TACUF(交易验证码待读标志)；
P2	‘00’
L_c	不存在
Data	不存在
L_e	‘1E’或不存在

参数、状态与 GET LOCK PROOF 响应报文数据域的关系

P1 参数	TACUF (交易验证码待读标志)	ET 灰锁状态	响应报文数据域	
			数据域列表	报文中的状态字
P1=‘00’	标志复位 （TAC 已读）	无灰锁	表 80	‘00’
		有灰锁	表 81	‘01’
	标志置位（TAC 未读）	不影响	表 82	‘10’
P1=‘01’	将 TACUF 标志复位	不影响	不存在	

当 P1 为 0 时，表示普通读取，根据灰锁状态和 TACUF(交易验证码待读标志)，形成不同的响应报文。

当前电子油票应用未灰锁，而且 TACUF(交易验证码待读标志)复位时的响应报文数据域。

正常状态 GET LOCK PROOF 响应报文数据域

说明	长度（字节）
状态字（=‘00’表示当前应用无灰锁）	1
上次发生的解扣、联机解扣的交易类型标识	1
上次解扣、联机解扣的 ET　(‘01’,其它 RFU)	1
上次解扣、联机解扣的 ET 的 余额	4
上次解扣的 ET 的脱机交易序号； 或联机解扣的 ET 的联机交易序号	2
上次执行解扣、联机解扣的终端机编号	6
上次执行解扣、联机解扣的日期	4
上次执行解扣、联机解扣的时间	3
上次解扣、联机解扣的交易金额	4
上次解扣的 TAC， 或联机解扣的 MAC3	4

定义了当前电子油票应用被灰锁时的响应报文数据域。

灰锁状态 GET LOCK PROOF 响应报文数据域

说明	长度（字节）
状态字（=‘01’表示当前应用已灰锁）	1
灰锁的交易类型标识	1
被灰锁的 ET(‘01’,其它 RFU)	1
被灰锁的 ET 余额	4
被灰锁的 ET 脱机交易序号	2
执行 GREY LOCK 时的终端机编号	6
执行 GREY LOCK 时的日期	4
执行 GREY LOCK 时的时间	3
灰锁时的 MAC2	4
灰锁时的 GTAC	4

TACUF（交易验证码待读标志）置位时的响应报文数据域。

TAC 未读时 GET LOCK PROOF 响应报文数据域

说明	长度（字节）
状态字（=‘10’表示当前应用 TAC 未读）	1
上次解扣的交易类型标识	1
上次解扣的 ET(‘01’,其它 RFU)	1
上次解扣的 ET 余额	4
上次解扣的 ET 脱机交易序号	2
上次执行解扣的终端机编号	6
上次执行解扣的日期	4
上次执行解扣的时间	3
解扣交易金额	4
上次解扣的 TAC	4

当 P1=‘01’时，表示清除 TACUF（交易验证码待读标志），响应报文的数据域不存在。

流程：

1.判断 CLA，不符合则返回 6E00

2.判断 PA1,PA2,不符合则返回 6A86

3.判断状态机，没有执行灰锁，则返回 6985

4.判断输入数据长度，不符合则返回 6700

5.判断是否通过密码校验，没有则返回 6985

6.查找内部交易明细表，没找到则返回 6A82

7.如果 P1 为 00，读取灰锁状态，写相关数据后返回 9000

8.清楚灰锁标志，返回 9000

<22>.取交易认证码

GET TRANSACTION PROOF 命令报文

代码	值
CLA	‘80’
INS	‘5A’
P1	‘00’
P2	要取的 MAC 或 / 和 TAC 所对应的交易类型标识。

L_c	‘02’
Data	见表 85
L_e	‘08’

GET TRANSACTION PROOF 命令报文数据域

说明	长度（字节）
要取的 MAC 或 / 和 TAC 所对应的当前的 ET 联机或脱机交易序号	2

GET TRANSACTION PROOF 响应报文数据域

说明	长度（字节）
MAC	4
TAC	4

流程：

1.判断 CLA，不符合则返回 6E00

2.判断 PA1,PA2,不符合则返回 6A86

3.判断输入数据长度，不符合则返回 6700

4.判断 ET 联机或脱机交易序号是否一致，不一致则返回 9406

5.把 MAC 和 TAC 写入响应缓冲区中，返回 6108

<23>.灰锁

GREY LOCK 命令报文

代码	值
CLA	‘E0’
INS	‘7C’
P1	‘08’
P2	‘00’
L_c	‘13’
Data	见表 89
L_e	‘08’

GREY LOCK 命令报文数据域

说明	长度（字节）
终端交易序号	4
终端随机数（TRAN）	4
交易日期（终端）	4
交易时间（终端）	3
MAC1	4

GREY LOCK 响应报文数据域

说明	长度（字节）
GTAC	4
MAC2	4

流程：

1.判断 CLA，不符合则返回 6E00

2.判断 PA1,PA2,不符合则返回 6A86

3.判断状态机，没有执行灰锁初始化，则返回 6985

4.判断输入数据长度，不符合则返回 6700

5.输出 INS

6.搜索密钥类型为 02，密钥标识为指定标识的密钥，没找到则返回 9403

7.用此密钥对以下数据进行计算得到临时密钥 A

伪随机数+电子油票联机交易序号+"8000"

8.用临时密钥对终端随机数进行计算得到临时密钥 SESLK

9.用 SESLK 对以下数据进行计算得到 MAC

交易类型标识+终端机编号+日期时间

10.比较 MAC，不一致则返回 9302

11.用 SELSK 对数据：卡余额+脱机交易序号进行计算得到 MAC2

12.搜索密钥类型为 06，密钥标识为指定标识的密钥，没找到则返回 9403

13.用该密钥异或后对以下数据进行计算得到 TAC

交易类型标识+终端机编号+终端交易序号+日期时间

14.查找内部交易明细文件，没找到则返回 6A82

15.写内部交易明细表，改变状态机为灰锁，更新灰锁状态，写 MAC2,TAC 到响应缓冲区，返回 6108

<24>.联机解扣

GREY UNLOCK 命令报文

代码	值
CLA	‘E0’
INS	‘7E’
P1	‘09’
P2	‘00’
L_c	‘0F’
Data	见表 93
L_e	‘04’

GREY UNLOCK 命令报文数据域

说明	长度（字节）
交易金额	4
交易日期（主机）	4
交易时间（主机）	3
MAC2	4

GREY UNLOCK 响应报文数据域

说明	长度（字节）
MAC3	4

流程：

1.判断 CLA，不符合则返回 6E00

2.判断 PA1,PA2,不符合则返回 6A86

3.判断状态机，没有执行联机解扣初始化，则返回 6985

4.判断输入数据长度，不符合则返回 6700

5.输出 INS

6.判断联机交易序号是否匹配，不一致，则返回 9406

7.判断余额与解扣金额大小，如果余额小于解扣金额，则返回 9401

8.搜索密钥类型为 09,密钥标识为指定标识的密钥（联机解扣密钥），如果没找到，则返回 9403

9.用该密钥对以下数据进行计算得到临时密钥

伪随机数+联机交易序号

10.用临时密钥对以下数据进行计算得到 MAC2

解扣金额+交易类型标识+终端机编号+日期时间

11.判断 MAC2 是否一致，不一致则增加校验 MAC2 失败次数，如果没超过指定次数，返回 9302

如果超过指定次数，则应用被临时锁定，返回 9302

12.用临时密钥对以下数据计算得到 MAC3

余额+联机交易序号+交易金额+交易类型标识+终端机编号+日期时间

13.查找内部交易明细文件，如没找到，则返回 6A82

14.查找 IC 卡交易明细文件，如没找到，则返回 6A82

15.操作内部交易明细文件，IC 卡交易明细文件，操作电子油票/电子钱包/电子存折，灰锁标志清零，状态机清除，把 MAC3 写入响应数据缓冲区，返回 6104

<25>.灰锁初始化

INITIALIZE FOR GREY LOCK 命令报文

代码	值
CLA	‘E0’
INS	‘7A’
P1	‘08’
P2	‘01’ 其他值保留
L_c	‘07’
Data	见表 97
L_e	‘0F’

INITIALIZE FOR GREY LOCK 命令报文数据域

说明	长度（字节）
密钥索引号	1
终端机编号	6

INITIALIZE FOR GREY LOCK 响应报文数据域

说明	长度（字节）
ET 余额	4
ET 脱机交易序号	2
透支限额	3
密钥版本号(DPK)	1
算法标识 (DPK)	1
伪随机数 (ICC)	4

流程：

1.判断 CLA，不符合则返回 6E00

2.判断 PA1,PA2,不符合则返回 6A86

3.判断输入数据长度，不符合则返回 6700

4.判断是否进行了密码校验，如果没有则返回 6985

5.输出 INS

6.判断当前的灰锁状态，如果已经灰锁，则返回 6985

7.搜索密钥类型为 02,密钥标识为指定标识的密钥，如没找到，则返回 9403

8.状态机发生改变，把以下数据写入响应缓冲区中，返回 610F

余额+脱机交易序号+透支限额+密钥版本号+算法标识+伪随机数

<26>.联机解扣初始化

INITIALIZE FOR GREY UNLOCK 命令报文

代码	值
CLA	‘E0’
INS	‘7A’
P1	‘09’
P2	‘01’ 其他值保留
L_c	‘07’
Data	见表 101
L_e	‘12’

INITIALIZE FOR GREY UNLOCK 命令报文数据域

说明	长度（字节）
密钥索引号	1
终端机编号	6

INITIALIZE FOR GREY UNLOCK 响应报文数据域

说明	长度（字节）
ET 余额	4
ET 脱机交易序号	2
ET 联机交易序号	2
密钥版本号(DUKK)	1
算法标识 (DUKK)	1
伪随机数(ICC)	4
MAC1	4

流程：

1.判断 CLA，不符合则返回 6E00

2.判断 PA1,PA2,不符合则返回 6A86

3.判断输入数据长度，不符合则返回 6700

4.判断是否进行了密码校验，如果没有则返回 6985

5.输出 INS

6.判断当前的灰锁状态，如果没有灰锁，则返回 6985

7.搜索密钥类型为 09,密钥标识为指定标识的密钥（联机解扣密钥），如没找到，则返回 9403

8.用该密钥对以下数据进行计算，得到临时密钥

伪随机数+联机交易序号

9.用临时密钥对以下数据进行计算，得到 MAC1

余额+脱机交易序号+交易类型标识+终端机编号

10.把以下数据写入响应缓冲区中，改变状态机，返回 6112

余额+脱机交易序号+联机交易序号+密钥版本号+算法标识+伪随机数+MAC1

<27>.圈存初始化

INITIALIZE FOR LOAD 命令报文

代码	值
CLA	‘80’
INS	‘50’

P1	‘00’
P2	‘01’；其他值保留
L_c	‘0B’
Data	见表 105
L_e	‘10’

INITIALIZE FOR LOAD 命令报文数据域

说明	长度（字节）
密钥索引号	1
交易金额	4
终端机编号	6

INITIALIZE FOR LOAD 响应报文

说明	长度（字节）
ET 余额	4
ET 联机交易序号	2
密钥版本号（DLK）	1
算法标识（DLK）	1
伪随机数（IC 卡）	4
MAC1	4

流程：

1.判断 CLA，不符合则返回 6E00

2.判断 PA1,PA2,不符合则返回 6A86

3.判断输入数据长度，不符合则返回 6700

4.判断是否进行了密码校验，如果没有则返回 6985

5.判断是否已经被灰锁，如果灰锁，则返回 6985

6.输出 INS

7.搜索密钥类型为 05,密钥标识为指定标识的密钥（圈存密钥），如没找到则分那会 9403

8.用该密钥对以下数据进行运算，得到临时密钥

伪随机数+联机交易序号

9.用临时密钥对以下数据运算，得到 MAC1

余额+交易金额+交易类型标识+终端机编号

10.把以下数据写入响应缓冲区，改变状态机，返回 6110

余额+联机交易序号+密钥版本+算法标识+伪随机数+MAC1

<28>.消费初始化

INITIALIZE FOR PURCHASE 命令报文

代码	值
CLA	‘80’
INS	‘50’
P1	‘02’
P2	‘01’用于加油普通消费交易； 其他值保留
L_c	‘0B’
Data	见表 109
L_e	‘0F’或’00’

INITIALIZE FOR PURCHASE 命令报文数据域

说明	长度（字节）
密钥索引号	1
交易金额	4
终端机编号	6

表 110　　INITIALIZE FOR PURCHASE 响应报文数据域

说明	长度（字节）
ET 余额	4
ET 脱机交易序号	2
透支限额	3
密钥版本号（DPK）	1
算法标识（DPK）	1
伪随机数（IC 卡）	4

流程：

1.判断 CLA，不符合则返回 6E00

2.判断 PA1,PA2,不符合则返回 6A86

3.判断输入数据长度，不符合则返回 6700

4.判断是否进行了密码校验，如果没有则返回 6985

5.判断是否已经被灰锁，如果灰锁，则返回 6985

6.输出 INS

7.搜索密钥类型为 02,密钥标识为指定标识的密钥（消费密钥），如没找到则分那会 9403

8.搜索电子油票/电子钱包/电子存折，如没找到则返回 6A82

9.把以下数据写入响应缓冲区，改变状态机，返回 610F

余额+脱机交易序号+透支限额+密钥版本号+算法标识+伪随机数

<29>.圈提初始化

INITIALIZE FOR UNLOAD 命令报文

代码	值
CLA	’80’
INS	’50’
P1	’05’
P2	‘01’用于 ET 圈提交易；其他值保留
Lc	‘0B’
Data	‘见表 113’

Le	‘10’

INITIALIZE FOR UNLOAD 命令报文数据域

说明	长度(字节)
密钥索引号	1
交易金额	4
终端机编号	6

INITIALIZE FOR UNLOAD 响应报文数据域

说明	长度(字节)
ET 余额	4
ET 联机交易序号	2
密钥版本号(DULK)	1
算法标识(DULK)	1
伪随机数(IC 卡)	4
MAC1	4

流程：

1.判断 CLA，不符合则返回 6E00

2.判断 PA1,PA2,不符合则返回 6A86

3.判断输入数据长度，不符合则返回 6700

4.判断是否进行了密码校验，如果没有则返回 6985

5.判断是否已经被灰锁，如果灰锁，则返回 6985

6.输出 INS

7.搜索密钥类型为 07,密钥标识为指定标识的密钥（圈提密钥），如没找到则分那会 9403

8.搜索电子油票/电子钱包/电子存折，如没找到则返回 6A82

9.用该密钥对以下数据进行运算，得到临时密钥

伪随机数+联机交易序号

10.用临时密钥对以下数据进行计算，得到 MAC1

余额+交易金额+交易类型标识+终端机编号

11.把以下数据写入响应缓冲区，改变状态机，返回 6110

余额+联机交易序号+密钥版本号+算法标识+伪随机数+MAC1

<30>.修改透支限额初始化

INITIALIZE FOR UPDATE 命令报文

代码	值
CLA	‘80’
INS	‘50’
P1	‘04’
P2	‘01’
L_c	‘07’
Data	见表 117
L_e	‘13’

INITIALIZE FOR UPDATE 命令报文数据域

说明	长度（字节）
密钥索引号	1
终端机编号	6

INITIALIZE FOR UPDATE 响应报文数据域

说明	长度（字节）
ET 余额	4
ET 联机交易序号	2
旧透支限额	3
密钥版本号（DUK）	1
算法标识（DUK）	1
伪随机数（IC 卡）	4
MAC1	4

<31>.重装个人密码

RELOAD PIN 命令报文

代码	值
CLA	‘80’

INS	‘5E’
P1	‘00’
P2	‘00’
L_c	‘06’~’OA’
Data	见表 121
L_e	不存在

RELOAD PIN 命令报文数据域

说明	长度（字节）
重装的 PIN 值	2－6
MAC	4

<32>.修改透支限额

UPDATE OVERDRAW LIMIT 命令报文

代码	值
CLA	‘80’
INS	‘58’
P1	‘00’
P2	‘00’
L_c	‘0E’
Data	见表 124
L_e	‘04’

UPDATE OVERDRAW LIMIT 命令报文数据域

说明	长度（字节）
新透支限额	3
交易日期（发卡方）	4
交易时间（发卡方）	3
MAC2	4

UPDATE OVERDRAW LIMIT 响应报文数据域

说明	长度（字节）
TAC	4