CAS 机顶盒移植文档

一、 宏和数据结构定义说明 2
1、 模块功能开关 2
2、 变量类型定义 2
3、 CA 任务优先级定义 3
4、 设置 ECM，EMM 数据接口定义 3
5、 滚动字幕显示位置 3
6、 ECM,EMM 数据过滤号定义 3
7、 ECM,EMM 数据过滤超时时间定义 3
8、 CAS 状态和错误信息定义 3
9、 消息队列的定义 4
10、 设置 ECM 信息接口定义 4
11、 OSD 信息显示定义 4
12、 时间结构定义 4
13、 邮件相关定义 4
14、 函数返回值定义 5
二、 机顶盒需要调用的函数接口 5
1、 CAS 初始化接口，机顶盒刚开机初始化好硬件环境后调用 5
2、 获取智能卡卡号，长度 8 位 5
3、 查询智能卡是否授权标志 5
4、 查询智能卡类型 5
5、 查询是机卡否配对标志 6
6、 获取智能卡的网络 ID 号 6
7、 设置智能卡的 NIT 表中解析出来的网络 ID 号 6
8、 设置机顶盒的 ID 号 6
9、 机顶盒接收到 ECM,EMM 数据后反馈给 CA 库 6
10、 设置 CW 密钥接口 6
11、 读取母卡喂养数据接口 7
12、 喂养母卡数据到子卡 7
13、 获取邮件头部信息接口和总个数 7
14、 获取新邮件总个数 7
15、 获取邮件头部信息 7
16、 获取邮件内容 7
17、 删除邮件接口 7
三、 机顶盒需要实现的接口 7
1、 CAS 消息通知接口 7
2、 智能卡初始化接口 8
3、 智能卡复位接口 8
4、 智能卡通讯接口 8

5、 获得当前节目的 ServiceID.... 8
6、 获得当前播放节目的 Emm Pid.... 8
7、 获得当前播放节目的 Ecm Pid.... 9
8、 机顶盒检测到卡状态变化时通知 CA 库.... 9
9、 校验 CA 系统 ID 是否是芯视猫 CAS.... 9
10、 设置 ECM PID 接口.... 9
11、 设置 EMM PID 接口.... 9
12、 过滤 ECM,EMM 数据接口.... 9
13、 获取机顶盒提供给 CA 模块保存信息的起始地址和空间大小.... 10
14、 从机顶盒分配空间的指定起始地址读指定长度的数据.... 10
15、 向机顶盒的存储空间写信息.... 10
16、 创建任务接口.... 11
17、 创建一个消息队列接口.... 11
18、 获取消息接口.... 11
19、 发送消息到消息队列接口.... 12
20、 CA 库打印接口.... 12
21、 内存设置接口.... 12
22、 字符串比较接口.... 12
23、 设置机顶盒时间接口.... 12

一、宏和数据结构定义说明

1、模块功能开关

```
#define SUPPORT_MATHER_SON_CARD \\是否打开子母卡功能
#define SUPPORT_CARD_STB_PAIR \\是否打开机卡配对功能
//#define SUPPORT_EMAIL_OSD \\是否打开邮件字幕功能
#define SUPPORT_REGION_LIMIT \\是否打开区域控制功能
```

2、变量类型定义

```
typedef signed char INT8;
typedef signed short INT16;
typedef signed long    INT32;
typedef unsigned char UINT8;
typedef unsigned short UINT16;
typedef unsigned long    UINT32;
typedef unsigned char    CAS_BOOL;
#define CAS_SUCCESS 0
#define CAS_FAILTURE -1
#define CAS_TRUE 1
```

```
#define CAS_FALSE 0
#define CAS_NULL      (0)
```

3、CA 任务优先级定义

```
#define     CA_TASK_PRIORITY     4
```

4、设置 ECM，EMM 数据接口定义

```
#define CA_INFO_CLEAR 0 //清除上一次  设置的 ECM，EMM PID
#define CA_INFO_ADD       1 //增加设置  ECM，EMM PID
```

5、滚动字幕显示位置

```
#define CA_OSD_ON_TOP     0
#define CA_OSD_ON_BOTTOM 1
```

6、ECM,EMM 数据过滤号定义

```
#define CA_STB_FILTER_1 1 //emm  授权
#define CA_STB_FILTER_2 2 //emm osd email
#define CA_STB_FILTER_3 3// emm osd email
#define CA_STB_FILTER_4 4 //ecm
```

7、ECM,EMM 数据过滤超时时间定义

```
#define ECM_FITLER_TIMEOUT (30)//单位:毫秒
#define EMM_FITLER_TIMEOUT (100)//单位:毫秒
```

8、CAS 状态和错误信息定义

```
#define       SC_NORMAL                       0    /*正常状态*/
#define       SC_NO_CARD                1    /*未插卡*/
#define       SC_NO_PPV_RIGHT           2    /*没有 PPV 授权*/
#define       SC_NO_PPC_RIGHT           3    /*没有 PPC 授权*/
#define       SC_PARENT_CTRL            4 /*家长控制*/
#define       SC_NO_CONDITION           5 /*条件限播*/
#define       SC_INVALID_CARD              6    /*无效卡*/
#define      SC_TYPEERROR                 7    /*子母卡喂养失败,插入智能卡类型错误*/
#define      SC_NEEDFEED            8    /*子卡需要与母卡对应,请喂养子卡*/
#define       SC_ZONEERROR                    9    /*区域错误*/
#define       SC_FEEDTIMEERROR             10 /*喂养时间错误*/
#define       SC_FEEDDATEERROR             11 /*喂养日期错误*/
#define       SC_FEEDSYSTEMTIMEERROR    12 /*系统时间没有正确获取,喂养失败*/
```

```
#define        SC_ACSEVICEIDEERROR    13 /*加扰机 AC  错误*/
#define        SC_CARDSTBNOTPAIR    14 /*加扰机 AC  错误*/
#define        SC_CARDENTITLEREMAINTDAY    15 /*授权剩余天数*/
```

9、消息队列的定义

```
typedef struct {
            UINT32 q1stWordOfMsg;
        UINT32 q2ndWordOfMsg;
        UINT32 q3rdWordOfMsg;
        UINT32 q4thWordOfMsg;
}XinShiMao_Queue_message;
```

10、设置 ECM 信息接口定义

```
#define CA_MAX_SERVICE_PER_ECM 1
typedef struct _CAServiceInfo {
      UINT16 m_wEcmPid;// ecm pid
      UINT8 m_bServiceCount;//默认填 1
      UINT16 m_wServiceId[CA_MAX_SERVICE_PER_ECM];// service id
}XinShiMao_CAServiceInfo;
```

11、OSD 信息显示定义

```
typedef struct _CAOSDInfo{
      UINT8 m_bOSD_Position;//显示位置  0:top,1:bottom
      UINT16 m_wOSD_Show_Circle;//是否循环显示
      UINT8 m_bOSD_Text_length;//文本长度
      UINT8 m_bOSD_Text[150];//文本内容
}XinShiMao_CAOSD_Info;
```

12、时间结构定义

```
typedef struct _TXinShiMao_DATETIME
{
      UINT16 Y;     //年
      UINT16 M;    //月
      UINT8      D;    //日
      UINT8      H;    //时
      UINT8      MI; //分
      UINT8      S;    //秒
}TXinShiMao_DATETIME;
```

13、邮件相关定义

```
#define     MAX_EMAIL_NUM                10\\最大邮件数
#define     EMAIL_TITLE_LEN              36\\邮件标题最大长度
#define     EMAIL_CONTENT_LEN            128\\邮件内容最大长度

typedef struct _TXinShiMao_EmailHead{
    TXinShiMao_DATETIME    m_tCreateTime; /*创建时间*/
    UINT8                  m_bEmail_Level;      /*重要程度*/
    UINT8          m_bNewEmail;               /*0:旧邮件  1:新邮件*/
    UINT16         m_bEmailID;     /*邮件 ID 号*/
    UINT8     m_szEmailTitle[EMAIL_TITLE_LEN] /*邮件标题*/
}TXinShiMao_EmailHead;

typedef struct _TXinShiMao_EmailContent{
    UINT8          m_szEmail[EMAIL_CONTENT_LEN]; /*邮件内容*/
}TXinShiMao_EmailContent;
```

14、函数返回值定义

```
typedef enum{
    XINSHIMAO_ERR = -100,//成功
    XINSHIMAO_OK = 0//失败
}XINSHIMAO_RESULT;
```

二、机顶盒需要调用的函数接口

1、 CAS 初始化接口，机顶盒刚开机初始化好硬件环境后调用

```
extern INT32 XinShiMao_Initialize(UINT8 mailManagerType);
```

2、 获取智能卡卡号，长度 8 位

```
extern INT32 XinShiMao_GetSMCNO(UINT8 * pbCardno);
```

3、 查询智能卡是否授权标志

```
//isEntitle 1：已授权，0：未授权
extern INT32 XinShiMao_GetSMCEntitle(UINT8 * isEntitle);
```

4、 查询智能卡类型

```
//0:母卡，1:子卡
extern INT32 XinShiMao_GetSMCType(UINT8 * isType);
```

5、 查询是机卡否配对标志

//0:自由配对，1:机卡不配对，2:机卡已配对
extern INT32 XinShiMao_GetSMCStbPaired(UINT8 * isCardStbPaired);

6、 获取智能卡的网络 ID 号

extern UINT16 XinShiMao_GetCardRegionID(void);

7、 设置智能卡的 NIT 表中解析出来的网络 ID 号

NIT 表中的第 3,4 字节
extern void XinShiMao_SetNetWorkRegionID(UINT16 chRegionID);

8、 设置机顶盒的 ID 号

2 字节，用于区域控制，开机初始化好库后调用，在广电要求机顶盒，卡都要区域控制的情况下调用，机顶盒将区域号写成跟智能卡的区域号一致。一般情况下可以不调用。
extern void XinShiMao_SetStbRegionID(UINT16 chRegionID);

9、 机顶盒接收到 ECM,EMM 数据后反馈给 CA 库

pbReceiveData[0]:0x80,0x81 ecm;0x82,0x83,0x84 emm */
输入参数：
bOK： 收取数据有没有成功；TRUE：成功，FALSE：失败。
wPid： 接收的流的 PID。
pbReceiveData： 收取私有数据的指针，CA 模块不负责其空间的释放。
wLen： 收取到的私有数据的长度。

extern void XinShiMao_TableReceived(UINT8 bRequestID,UINT8 bOK,UINT16 wPid,const UINT8 * pbReceiveData,UINT16 wLen);

10、 设置 CW 密钥接口

输入参数：
wEcmPid， 控制字的所在的 ecm 包的 PID 值。
szOddKey， 奇控制字数据的指针。
szEvenKey， 偶控制字数据的指针。
bKeyLen， 控制字长度。
bReservedFlag, 保留。
extern void XinShiMao_SetCW(UINT16 wEcmPid,const UINT8 * szOddKey,const UINT8 * szEvenKey,UINT8 bKeyLen,UINT8 bReservedFlag);

11、读取母卡喂养数据接口

//ret: 1:invailed card 2:mother card and son card is not paired
extern UINT8 XinShiMao_ReadFeedDataFromParent(UINT8 * chSpbyFeedData, UINT8 *chLen);

12、 喂养母卡数据到子卡

extern UINT8 XinShiMao_WriteFeedDataToChild(UINT8 * chSpbyFeedData, UINT8 chLen);

13、 获取邮件头部信息接口和总个数

返回邮件总个数
extern UINT16 XinShiMao_GetEmailHeads(TXinShiMao_EmailHead* pEmailHeads, UINT8 nTitleNum);

14、 获取新邮件总个数

extern UINT8 XinShiMao_GetNewMailCount(void);

15、 获取邮件头部信息

CAS_BOOL XinShiMao_GetEmailHead(UINT8 bEmailID,TXinShiMao_EmailHead* pEmailHead);

16、 获取邮件内容

CAS_BOOL XinShiMao_GetEmailContent(UINT8 bEmailID,TXinShiMao_EmailContent* pEmailContent);

17、 删除邮件接口

CAS_BOOL XinShiMao_DelEmail(UINT8 bEmailID);

三、机顶盒需要实现的接口

1、CAS 消息通知接口

输入参数： event 为消息类型 1: OSD 消息；4：邮件消息
其它：大画面的消息提示:param1 为消息提示的内容
extern void XinShiMao_EventHandle(UINT32 event,UINT32 param1,UINT32 param2,UINT32 param3);

2、智能卡初始化接口

extern INT32 XinShiMao_SC_DRV_Initialize(void);

3、智能卡复位接口

机顶盒检测到卡插入后，需要机顶盒调用复位接口，系统在卡通讯出错情况下，会调用复位接口。bCardNumber 默认 0。
extern void XinShiMao_SC_DRV_ResetCard(UINT8 bCardNumber);

4、智能卡通讯接口

输入参数：
bCardNumber： 暂时保留不用；
bLength： pabMessage 的长度；
pabMessage： 发送命令的消息指针；
pabResponse： 接收响应结果的数据块的指针；
bRLength： 响应结果的数据块长度指针。
输出参数：
pabResponse： 响应结果的数据块；
bRLength： 响应结果的数据块长度。
pbSW1 智能卡状态字节 1
pbSW2 智能卡状态字节 2
返 回 值：
CAS_TRUE：成功
CAS_FALSE：失败 */

```
extern INT32 XinShiMao_SC_DRV_SendDataEx(UINT8 bCardNumber,UINT8 bLength,
                                         UINT8 * pabMessage,
                                         UINT8 * pabResponse,
                                         UINT8 * bRLength,
                                         UINT8 * pbSW1,
                                         UINT8 * pbSW2);
```

5、获得当前节目的 ServiceID

返 回 值： 当前节目的 ServiceID
extern UINT16 XinShiMao_GetCurr_ServiceID(void);

6、获得当前播放节目的 Emm Pid

返 回 值： 当前的 Emm Pid
extern UINT16 XinShiMao_GetCurr_EmmID(void);

7、获得当前播放节目的 Ecm Pid

extern UINT16 XinShiMao_GetCurr_EcmID(void);

8、**机顶盒检测到卡状态变化时通知 CA 库**

extern void　XinShiMao_SCStatusChange(UINT8 status);

9、校验 CA 系统 ID 是否是芯视猫 CAS

extern CAS_BOOL　XinShiMao_IsMatchCAID(UINT16 wCASystemID);

10、设置 ECM PID 接口

当有新 ECM PID 变化时(如切换频道)调用该接口把相关信息传递给 CA 模块。
输入参数：
　　bType：　设置的类型，清空、增加、修改当前 ECMPID 列表，　使机顶盒同时可以处理多路节目。为以下几个值之一:
　　CAS_INFO_CLEAR：用户告诉 CA 清空 ECMPID, pEcmInfo　须是空；
　　CAS_INFO_ADD：　用户告诉 CA 增加一个 ECMPID；
extern void　XinShiMao_SetEcmPID(UINT8 bType,XinShiMao_CAServiceInfo * pEcmInfo);

11、设置 EMM PID　接口

extern void　XinShiMao_SetEmmPID(UINT8 bType,UINT16 wEmmPid);

12、过滤 ECM,EMM 数据接口

输入参数：
　　iRequestID　　,　　为如下几个值之一
　　CA_STB_FILTER_1,
　　CA_STB_FILTER_2
　　CA_STB_FILTER_3
　　CA_STB_FILTER_4
　　pbFilterMatch1，　如果是过滤 Ecm 数据，则该参数为 Ecm filter 的数据；
　　如果是过滤 Emm 数据，则该参数为 Emm filter1 的数据；
　　pbFilterMask1，　如果是过滤 Ecm 数据，则该参数为 Ecm filter 的 mask；
　　如果是过滤 Emm 数据，则该参数为 Emm filter1 的 mask；
　　bLen，　filter 的长度。
　　wPid，　通道的 PID 值。
　　bWaitTime，　通道过期时间，单位毫秒，超时时间见宏定义 ECM_FITLER_TIMEOUT, EMM_FITLER_TIMEOUT
　　if (wPid <0 && wPid >0x1fff) 提示机顶盒 DEMMUX 需要释放过滤器空间，

EMM 可以不释放，ECM 必须释放

ECM 过滤数据 TABILE_ID:0X80,0X81;emm 授权过滤 TABLE id:0X82;osd,email table id:0x83

返 回 值：

SUCCESS：成功，

FAILURE： 失败。 */

```
extern INT32 XinShiMao_TableStart(UINT8 iRequestID,
                                  const UINT8 * pbFilterMatch1,
                                                const UINT8 * pbFilterMask1,
                                                UINT8 bLen,
                                                UINT16 wPid,
                                                UINT8 bWaitTime);
```

13、获取机顶盒提供给 CA 模块保存信息的起始地址和空间大小

输出参数：

lStartAddr 机顶盒分配空间的起始地址

lSize 机顶盒分配空间的大小 ，目前只需 4K

返 回 值：

TRUE 成功

FALSE 失败 */

```
extern CAS_BOOL XinShiMao_GetBuffer(UINT32 *lStartAddr,UINT32 * lSize);
```

14、 从机顶盒分配空间的指定起始地址读指定长度的数据

输入参数：

lStartAddr： 要读取数据的存储空间的地址。

pbData： 被读取数据的存放地址指针。

nLen： 要读的数据的长度

输出参数：

pbData： 被读出的数据。

返 回 值：

返回实际读到的字节数

```
extern UINT32 XinShiMao_ReadBuffer(const UINT8 *lStartAddr,UINT8 *pData,INT32 nLen);
```

15、向机顶盒的存储空间写信息

输入参数：

lStartAddr： 要写的存储空间的目标地址。

pData： 要写的数据

nLen： 要写的数据的长度

输出参数： 无。

返 回 值：

TRUE：成功

FALSE ： 失 败 。 extern CAS_BOOL XinShiMao_WriteBuffer(const UINT32

lStartAddr,const UINT8 *pData,INT32 nLen);

16、创建任务接口

输入参数：

name[]， 4 个字节的任务名称。

stackSize， 任务所使用的堆栈的大小。

entryPoint， 任务的入口地址。

priority， 任务的优先级。

arg1， 传递给任务的第一个参数。

arg2， 传递给任务的第二个参数。 taskId， 任务的 ID。

输出参数： 无。

返 回 值：

SUCCESS：成功。

FAILURE： 发生错误。

```
extern INT32 XinShiMao_OSPTaskCreate(char name[],UINT32 stackSize,
                                     void (*entryPoint)(void*),
                                     INT32 priority,
                                     UINT32 arg1,
                                     UINT32 arg2,
                                     UINT32 * taskId);
extern INT32 XinShiMao_OSPTaskTemporarySleep(UINT32 milliSecsToWait);
```

17、创建一个消息队列接口

输入参数：

name[]， 4 个字节的队列名称。

maxQueueLength，消息队列中可以存放的消息的数量。当消息队列中该数量达到该数量时，再往该消息队列发消息将会失败。

taskWaitMode:可以不管

queueId， 消息队列的 ID。

输出参数： 无。

返 回 值：

SUCCESS：成功；

FAILURE： 发生错误

```
extern INT32 XinShiMao_OSPQueueCreate(char name[],UINT32 maxQueueLength,UINT32
taskWaitMode,UINT32 * queueId);
```

18、获取消息接口

输入参数：

queueId， 所要取得的消息的 ID。

messsage， 为消息的格式。参照 XINSHIMAO_QUEUE_MESSAGE。

waitMode:目前只用到，等待直到得到消息

SUCCESS：成功；

FAILURE： 发生错误。*/

extern INT32 XinShiMao_OSPQueueGetMessage(UINT32 queueId,XinShiMao_Queue_message * message,UINT32 waitMode,UINT32 milliSecsToWait);

19、 发送消息到消息队列接口

输入参数：

queueId， 消息队列 ID。

message， 要发送的消息。其格式见 XINSHIMAO_QUEUE_MESSAGE 结构。

输出参数： 无。

返 回 值：

SUCCESS：成功；

FAILURE： 发生错误。*/

extern INT32 XinShiMao_OSPQueueSendMessage(UINT32 queueId,XinShiMao_Queue_message * message);

20、CA 库打印接口

extern UINT32 XinShiMao_Printf(const char * fmt,...);

22、内存拷贝接口

extern void XinShiMao_memcpy(void* pDestBuf,const void* pSrcBuf,UINT32 wSize);

21、内存设置接口

extern void XinShiMao_memset(void* pDestBuf,UINT8 c,UINT32 wSize);

22、字符串比较接口

extern INT32 XinShiMao_memcmp(const void *buf1, const void *buf2, UINT32 count);

23、设置机顶盒时间接口

//设置机顶盒时间，机顶盒无需过滤 TDT 表，由 CA 来提供时间基准

extern void XinShiMao_SetStbTime(TXinShiMao_DATETIME* ca_time);