ANYKA
芯 随 你 动

版本: 1.0.2 2014 年 03 月

FHA 和 FSA 库接口说明

声　　明

联　系　方　式

安凯（广州）微电子技术有限公司

地址：广州科学城科学大道 182 号创新大厦 C1 区 3 楼
电话: (86)-20-3221 9000

传真: (86)-20-3221 9258

邮编: 510663

销售热线:

(86)-20-3221 9499

电子邮箱:

sales@anyka.com

主页:

http: //www. anyka.com

版 本 变 更 说 明

以下表格对于本文档的版本变更做一个简要的说明。版本变更仅限于技术内容的变更，不包括版式、格式、句法等的变更。

版本	说明	完成日期
V1.0.0	正式发布	2012-1-9
V1.0.1	更新版本的对应关系	2013-4-26
V1.0.2	针对 LINUX 或 ROTS 增加 6 个接口： ● 二个接口是写整个 SPI 的数据，用于芯片 BBT 测试； ● 二个接口是读整个 SPI 数据，用于 SPI 镜像制作； ● 二个接口是针对 LINUX 下载镜像文件的数据。	2014-3-25

FHA 和 FSA 库接口说明与库对应的关系

Version	The Corresponding Lib
V1.0.0	fhalib V1.0.6 & fsalib V1.0.0
V1.0.1	fhalib V1.0.13 & fsalib V1.0.10
V1.0.2	fhalib V1.0.15 & fsalib V1.0.11

目录

1 引言

本文档介绍本公司 FHA 和 FSA 的参数配置和使用方法，目的是为用户、研发人员和测试人员等相关人员的开发和应用提供指导。

1.1 术语

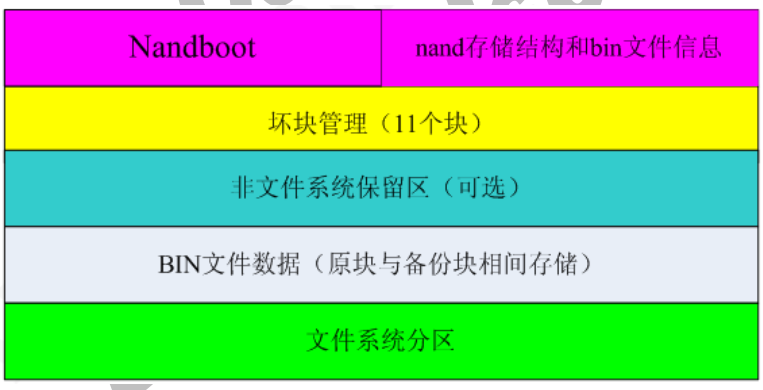
FHA：Flash hide area, flash 设备隐藏区。

FSA：File System area, 文件系统区域。

1.2 数据存储结构

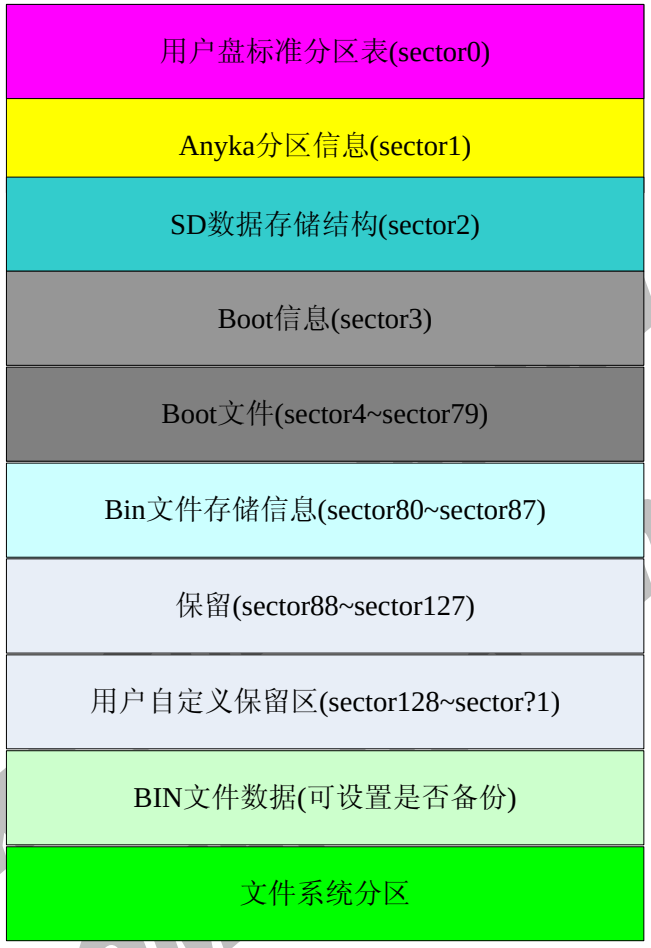
存储设备：nandflash，spiflash，sd/mmc card，虽然存储介质不相同，但是在存储设备上的逻辑划分都可以整体上划分为隐藏区和文件系统区。下面各图将描述各种存储介质上的数据存储结构。

1.2.1 Nandflash数据存储结构

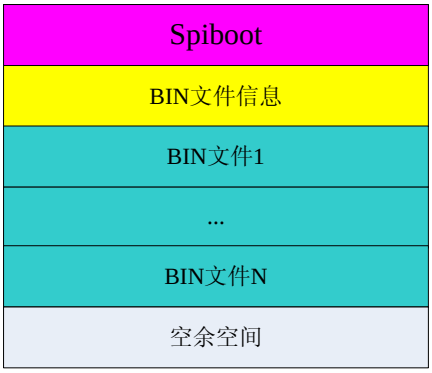


1.2.2 SD/EMMC数据存储结构

由于采用 SD 卡作为存储介质时，SD 卡可以被其他读卡器等单独访问，因此 sector0 会存储标准分区表信息，指引 PC 系统（windows）访问到后面的用户盘，而隐藏区以及不希望访问的文件系统分区将不被访问。Sector3 是芯片指定的 boot 信息区。



1.2.3 SPIflash数据存储结构



Spiflash 存储容量相对比较小（如 256KB~4MB），因此采用 spiflash 作为启动方案的系统一般以 SD 卡作为文件系统区。

2 接口说明

2.1 FHA接口设计

返回值：

#define FHA_SUCCESS 1

#define FHA_FAIL 0

2.1.1 隐藏区管理烧录初始化--FHA_burn_init

原　型	T_U32 FHA_burn_init(T_PFHA_INIT_INFO pInit, T_PFHA_LIB_CALLBACK pCB, T_pVOID pPhyInfo)	
功能概述	烧录时传入隐藏区管理必要的使用环境参数和回调函数	
参数说明	pInit	库使用环境参数信息
	pCB	库使用回调函数
	pPhyInfo	各种硬件参数：Nand，sd card，spiflash
返回值说明	FHA_SUCCESS	成功
	FHA_FAIL	失败
注意事项	-	
调用示例	-	

说明：该接口主要提供给烧录时传入必要的使用环境参数和回调函数。

参数结构体说明：

typedef T_U32 (*FHA_Erase)(T_U32 nChip, T_U32 nPage)

功能：擦除，默认都是擦除一个块（一个 nand 块或者一个 spiflash 块）

返回：FHA_SUCCESS 或 FHA_FAIL

参数：片选和起始页地址

typedef T_U32 (*FHA_Write)(T_U32 nChip, T_U32 nPage, const T_U8 *pData, T_U32 nDataLen, T_U32 Oob, T_FHA_DATA_TYPE eDataType)

功能：写数据

返回：FHA_SUCCESS 或 FHA_FAIL

参数：片选、起始页地址、数据和数据长度（nand 以字节为单位， SD 为 sector 个数(1sec = 512byte)，SPI 为页数目)，Oob 固定为四个字节（仅 nand 使用）

eDataType 主要是为了给调用者不同数据类型时用不同的方式写

typedef T_U32 (*FHA_Read)(T_U32 nChip, T_U32 nPage, T_U8 *pData, T_U32 nDataLen, T_U32 *Oob , T_FHA_DATA_TYPE eDataType)

功能：读数据

返回：FHA_SUCCESS 或 FHA_FAIL

参数：片选、起始页地址、数据和数据长度（nand 以字节为单位， SD 为 sector 个数(1sec = 512byte)，SPI 为页数目)，Oob 固定为四个字节（仅 nand 使用）

eDataType 主要是为了给调用者不同数据类型时用不同的方式读

typedef T_U32 (*FHA_ReadNandBytes)(T_U32 nChip, T_U32 rowAddr, T_U32 columnAddr, T_U8 *pData, T_U32 nDataLen);

功能：不带 ECC 的读取 nandflash 的数据

返回：FHA_SUCCESS 或 FHA_FAIL

参数：片选、地址、数据和数据长度

typedef T_pVOID (*FHA_RamAlloc)(T_U32 size);

typedef T_pVOID (*FHA_RamFree)(T_pVOID var);

typedef T_pVOID (*FHA_MemSet)(T_pVOID pBuf, T_S32 value, T_U32 count);

typedef T_pVOID (*FHA_MemCpy)(T_pVOID dst, T_pVOID src, T_U32 count);

typedef T_S32 (*FHA_MemCmp)(T_pVOID pbuf1, T_pVOID pbuf2, T_U32 count);

typedef T_S32 (*FHA_Printf)(T_pCSTR s, ...);

typedef tag_FHA_LibCallback

{

FHA_Erase Erase;

FHA_Write Write;

```
        FHA_Read Read;
        FHA_ReadBytes ReadBytes;
        FHA_RamAlloc RamAlloc;
        FHA_RamFree RamFree;
        FHA_MemSet MemSet;
        FHA_MemCpy MemCpy;
        FHA_MemCmp MemCmp;
        FHA_Printf     Printf
    }T_FHA_LIB_CALLBACK, T_PFHA_LIB_ CALLBACK;

typedef tag_FHA_Init_Info
    {
        T_U32 nChipCnt;                    //片选数
        T_U32 nBlockStep;                  //nand block step 值
        T_CHIP_TYPE  eAKChip;     //AK 芯片类型
      T_PLATFORM_TYPE ePlatform;//系统类型
        T_MEDIUM_TYPE eMedium;  //存储介质类型
        T_BURN_MODE eMode;                 //烧录模式
    }T_FHA_INIT_INFO, T_PFHA_INIT_INFO;
```

2.1.2 开始写隐藏区bin文件--FHA_write_bin_begin

原　　型	**T_U32 FHA_write_bin_begin(T_PFHA_BIN_PARAM bin_param)**	
功能概述	传入写隐藏区一个 bin 文件的必要参数	
参数说明	bin_param	Bin 文件信息
返回值说明	FHA_ SUCCESS	表示操作成功
	FHA_ FAIL	表示操作失败
注意事项	-	
调用示例	-	

参数结构体说明：

```
typedef struct tag_FHABinParam
    {
      T_U32   data_length;   //数据长度
      T_U32   ld_addr;                    //运行地址
      T_U8    file_name[16]; //文件名称
        T_BOOL  bBackup;              //是否备份
       T_BOOL  bCheck     //是否校验
        T_BOOL  bUpdateSelf  //spotlight 自升级用，会给每个 BIN 预留同样多的空间
    }T_FHA_BIN_PARAM, T_PFHA_BIN_PARAM;
```

2.1.3 写隐藏区Bin文件数据-- FHA_write_bin

原　　型	**T_U32 FHA_write_bin(const T_U8 * pData, T_U32 data_len)**	
功能概述	写一段数据 buffer	
参数说明	pData	Bin 文件数据 buffer
	data_len	Bin 文件数据 buffer 长度
返回值说明	FHA_ SUCCESS	表示操作成功
	FHA_ FAIL	表示操作失败
注意事项	-	
调用示例	-	

2.1.4 开始写隐藏区boot文件-- FHA_write_boot_begin

原　　型	T_U32 FHA_write_boot_begin(T_U32 bin_len, T_BOOL bCheck)	
功能概述	告诉模块开始写 boot 文件，并传入 boot 文件数据长度	
参数说明	bin_len	Boot Bin 文件长度
	bCheck	是否校验
返回值说明	FHA_ SUCCESS	表示操作成功
	FHA_ FAIL	表示操作失败
注意事项	-	
调用示例	-	

2.1.5 写隐藏区Boot文件数据-- FHA_write_boot

原　　型	T_U32 FHA_write_boot(const T_U8 * pData, T_U32 data_len)	
功能概述	写一段数据 buffer	
参数说明	pData	Boot Bin 文件数据 buffer
	data_len	Boot Bin 文件数据 buffer 长度
返回值说明	FHA_ SUCCESS	表示操作成功
	FHA_ FAIL	表示操作失败
注意事项	-	
调用示例	-	

2.1.6 获取隐藏区目前最后的block位置-- FHA_get_last_pos

原　　型	T_U32 FHA_get_last_pos()
功能概述	得到隐藏区最后的位置（单位：block），随着写数据将实时变化
参数说明	无
返回值说明	返回位置：block 值
注意事项	-
调用示例	-

2.1.7 设置文件系统分区-- FHA_set_fs_part

原　　型	T_U32 FHA_set_fs_part(T_U8 *pInfoBuf , T_U32 buf_len);	
功能概述	传入分区要求，记录到隐藏区	
参数说明	pInfoBuf	文件系统分区信息的数据 buffer
	buf_len	pInfoBuf 的数据长度
返回值说明	FHA_ SUCCESS	表示操作成功
	FHA_ FAIL	表示操作失败
注意事项	-	
调用示例	-	

2.1.8 建立安全区-- FHA_asa_format

原　　型	T_U32 FHA_asa_format(T_U32 type)	
功能概述	当安全区不存在时建立坏块管理的安全区	
参数说明	type	安全区坏块管理建立的方式，比如正常读取出厂坏块、全盘读写擦扫描等人为定义方式，默认是正常方式“0”
返回值说明	FHA_ SUCCESS	表示操作成功
	FHA_ FAIL	表示操作失败
注意事项	-	
调用示例	-	

2.1.9 加载安全区-- FHA_asa_scan

原　　型	T_U32 FHA_asa_scan(T_BOOL bMount)	
功能概述	扫描隐藏区，加载已经建立好的安全区	
参数说明	bMount	是否把所有坏块信息缓存到一个 buffer 中
返回值说明	FHA_ SUCCESS	表示操作成功
	FHA_ FAIL	表示操作失败
注意事项	-	
调用示例	-	

2.1.10 置一个块的坏块信息--FHA_set_bad_bloc

原　　型	T_U32 FHA_set_bad_block(T_U32 block)	
功能概述	把一个块的坏块信息加入安全区	
参数说明	block	该块为坏块
返回值说明	FHA_ SUCCESS	表示操作成功
	FHA_ FAIL	表示操作失败
注意事项	-	
调用示例	-	

2.1.11 判断一个块是否坏块-- FHA_check_bad_block

原　　型	T_BOOL FHA_check_bad_block(T_U32 block)	
功能概述	判断一个块是否是坏块	
参数说明	block	被判断的块
返回值说明	AK_TRUE	该块为坏块
	AK_FALSE	该块不是坏块
注意事项	-	
调用示例	-	

2.1.12 获取一片连续区域的坏块信息--FHA_get_bad_bloc

原　　型	T_U32 FHA_get_bad_block(T_U32 start_block, T_U8 pData[], T_U32 blk_cnt)	
功能概述	得到一个连续区域块的坏块信息位表	
参数说明	start_block	起始块
	pData	（out）坏块信息位表的 buffer
	blk_cnt	获取多少个块信息
返回值说明	FHA_ SUCCESS	表示操作成功
	FHA_ FAIL	表示操作失败
注意事项	-	
调用示例	-	

2.1.13 加载隐藏区-- FHA_mount

原　　型	T_U32 FHA_mount(T_PFHA_INIT_INFO pInit, T_PFHA_LIB_CALLBACK pCB)	
功能概述	在需要读取安全区的信息时，初始化安全区	
参数说明	pInit	库使用环境参数信息
	pCB	库使用回调函数
返回值说明	FHA_ SUCCESS	表示操作成功
	FHA_ FAIL	表示操作失败
注意事项	-	
调用示例	-	

2.1.14 开始读取一个bin文件-- FHA_read_bin_begin

原　　型	T_U32 FHA_read_bin_begin(T_PFHA_BIN_PARAM bin_param)	
功能概述	开始读取隐藏区的文件	
参数说明	bin_param	(in/out)传入 BIN 文件名称，获取 Bin 文件信息
返回值说明	FHA_ SUCCESS	表示操作成功
	FHA_ FAIL	表示操作失败
注意事项	注意：增加 if (bin_param->file_name[0] < file_count) 判断 也就是如果输入的参数中的 bin_param->file_name[0]小于文件个数，就认为是按照序号 bin_param->file_name[0]来查找文件，0~file_count	
调用示例	-	

2.1.15 读取bin文件的一段buffer-- FHA_read_bin

原　　型	T_U32 FHA_read_bin(T_U8 *pData, T_U32 data_len);	
功能概述	读取隐藏区的文件	
参数说明	pData	（out）Bin 文件数据 buffer
	data_len	Bin 文件数据 buffer 长度
返回值说明	FHA_ SUCCESS	表示操作成功
	FHA_ FAIL	表示操作失败
注意事项	-	
调用示例	-	

2.1.16 获取Nandflash的参数-- FHA_get_nand_para

原　　型	T_U32FHA_get_nand_para(T_NAND_PHY_INFO *pNandPhyInfo);	
功能概述	获取 nand 参数	
参数说明	pNandPhyInfo	（out）nand 参数
返回值说明	FHA_ SUCCESS	表示操作成功
	FHA_ FAIL	表示操作失败
注意事项	-	
调用示例	-	

NandFlash 物理特性参数

```
typedef struct
{
  T_U32        chip_id;        //芯片 ID 号
  T_U16        page_size;          //Page 大小
  T_U16        page_per_blk;       //一个 block 的总 Page 数
  T_U16        blk_num;            //总 block 数目
  T_U16        group_blk_num; //
   T_U16       plane_blk_num; //
  T_U8   spare_size;           //spare size
  T_U8         col_cycle;          //column address cycle
  T_U8         lst_col_mask;       //last column address cycle mask bit
  T_U8         row_cycle;          //row address cycle
  T_U8         delay_cnt;          // Rb delay, unit is 1024 asic clock, default value corresponds to 84MHz
  T_U8         custom_nd;          //initial bad block type
  T_U32   flag;                    //
   T_U32 cmd_len;                  //nandflash command length
  T_U32data_len;              //nandflash data length
   T_U8   des_str[32];             //描述符
}T_NAND_PHY_INFO;
```

2.1.17 取文件系统分区信息-- FHA_get_fs_part

原　　型	T_U32 FHA_get_fs_part(T_U8 *pInfoBuf , T_U32 buf_len);	
功能概述	获取文件系统分区信息	
参数说明	pInfoBuf	pInfoBuf
	buf_len	buf_len
返回值说明	FHA_ SUCCESS	表示操作成功
	FHA_ FAIL	表示操作失败
注意事项	-	
调用示例	-	

2.1.18 销毁函数-- FHA_close

原　　型	T_U32 FHA_close();	
功能概述	烧录时把需要的 bin 文件信息，nand 参数，分区信息等写上，并释放隐藏区占用的资源，系统读取时释放资源	
参数说明	无	
返回值说明	FHA_ SUCCESS	表示操作成功
	FHA_ FAIL	表示操作失败
注意事项	-	
调用示例	-	

2.1.19 往安全区写文件--FHA_asa_write_file

原　　型	T_U32　FHA_asa_write_file(T_U8 file_name[], const T_U8 *pData, T_U32 data_len, T_U8 mode)	
功能概述	往安全区写一段自己命名的数据 buffer	
参数说明	file_name	安全区文件命名，最长 7 个字节
	pData	文件数据
	data_len	数据长度
	mode	写模式，ASA_MODE_OPEN(若原来存在则直接返回)，ASA_MODE_CREATE（若原来存在则改写原来数据）
返回值说明	FHA_ SUCCESS	表示操作成功
	FHA_ FAIL	表示操作失败
注意事项	#define　ASA_MODE_OPEN　　0 #define　ASA_MODE_CREATE　　1	
调用示例	注意：读写 mac 地址或序列号的数据的结构是：前 4 个字节是数据的长度+数据 烧录时，会把由前 4 个字节的数据长度+数据这样的结构写下去；所以在平台重启时要进行读写时，也必需要以这样的结构进行读写操作。 FHA_burn_init(T_PFHA_INIT_INFO pInit, T_PFHA_LIB_CALLBACK pCB, T_pVOID pPhyInfo) const T_U8 pData []; pData[0] = **buf_len**; Memcpy（pData + 4, buf, **buf_len**); FHA_asa_write_file(T_U8 file_name[], const T_U8 *pData, T_U32 data_len, T_U8 mode) FHA_close();	

2.1.20 读取安全区文件-- FHA_asa_read_file

原　　型	T_U32 FHA_asa_read_file(T_U8 file_name[], T_U8 *pData, T_U32 data_len)	
功能概述	从安全区读取一段自己命名的曾经写入的数据	
参数说明	file_name	安全区文件命名，最长 7 个字节
	pData	（out）文件数据
	data_len	数据长度
返回值说明	FHA_ SUCCESS	表示操作成功
	FHA_ FAIL	表示操作失败
注意事项	-	

调用示例	注意：读写 mac 地址或序列号的数据的结构是：前 4 个字节是数据的长度+数据
	烧录时，会把由前 4 个字节的数据长度+数据这样的结构写下去；所以在平台重启时要进行读写时，也必需要以这样的结构进行读写操作。
	
	FHA_burn_init(T_PFHA_INIT_INFO pInit, T_PFHA_LIB_CALLBACK pCB, T_pVOID pPhyInfo)
	//先读 4 个字节的数据长度
	FHA_asa_read_file(T_U8 file_name[], T_U8 *pData, 4)
	memcpy(&len, data, 4);
	//再读数据
	FHA_asa_read_file(T_U8 file_name[], T_U8 *pData, len)
	FHA_close();	

2.1.21 设置隐藏区保留区大小-- FHA_set_resv_zone_info

原　　型	T_U32 FHA_set_resv_zone_info(T_U32 nSize, T_BOOL bErase)	
功能概述	设置隐藏区保留区的数据大小	
参数说明	start_block	保留区大小（单位：M）
	bErase	是否擦除保留区数据
返回值说明	FHA_ SUCCESS	表示操作成功
	FHA_ FAIL	表示操作失败
注意事项	-	
调用示例	-	

2.1.22 读取隐藏区保留区信息-- FHA_get_resv_zone_info

原　　型	T_U32 FHA_get_resv_zone_info(T_U16 *start_block， T_U16 *block_cnt)	
功能概述	从隐藏区读取保留区的其实地址和占用 block 个数	
参数说明	start_block	(out) 保留区起始块
	block_cnt	（out）保留区块个数
返回值说明	FHA_ SUCCESS	表示操作成功
	FHA_ FAIL	表示操作失败
注意事项	-	
调用示例	-	

2.1.23 获取BIN文件的个数-- FHA_get_bin_num

原　　型	T_U32 FHA_get_bin_num(T_U32 *cnt)	
功能概述	获取 BIN 文件的个数	
参数说明	cnt	（out） cnt—输出 BIN 文件的个数
返回值说明	FHA_ SUCCESS	表示操作成功
	FHA_ FAIL	表示操作失败
注意事项	-	
调用示例	-	

2.1.24 写整个spi数据开始-- FHA_write_AllDatat_begin

原　　型	T_U32 FHA_write_AllDatat_begin(T_PFHA_BIN_PARAM bin_param);	
功能概述	写整个 spi 数据开始-	
参数说明	T_PFHA_BIN_PARAM bin_param	（int ）文件信息
返回值说明	FHA_ SUCCESS	表示操作成功
	FHA_ FAIL	表示操作失败
注意事项	-	
调用示例	-	

2.1.25 写整个spi数据--FHA_write_AllDatat

原　　型	T_U32 FHA_write_AllDatat(const T_U8 * pData, T_U32 data_len);	
功能概述	写整个 spi 数据	
参数说明	const T_U8 * pData	数据
	T_U32 data_len	数据的长度
返回值说明	FHA_ SUCCESS	表示操作成功
	FHA_ FAIL	表示操作失败
注意事项	-	
调用示例	-	

2.1.26 下载镜像文件的数据开始-- FHA_write_MTD_begin

原　　型	T_U32 FHA_write_MTD_begin(T_PFHA_BIN_PARAM bin_param);	
功能概述	下载镜像文件的数据开始	
参数说明	T_PFHA_BIN_PARAM bin_param	镜像文件信息
返回值说明	FHA_ SUCCESS	表示操作成功
	FHA_ FAIL	表示操作失败
注意事项	-	
调用示例	-	

2.1.27 下载镜像文件的数据-- FHA_get_bin_num

原　　型	T_U32 FHA_write_MTD(const T_U8 * pData, T_U32 data_len);	
功能概述	下载镜像文件的数据	
参数说明	const T_U8 * pData,	镜像数据
	T_U32 data_len	镜像数据的长度
返回值说明	FHA_ SUCCESS	表示操作成功
	FHA_ FAIL	表示操作失败
注意事项	-	
调用示例	-	

2.1.28 读取整个spi数据开始-- FHA_read_AllDatat_begin

原　　型	T_U32 FHA_read_AllDatat_begin(T_PFHA_BIN_PARAM bin_param);	
功能概述	获取 BIN 文件的个数	
参数说明	T_PFHA_BIN_PARAM bin_param	要读取的文件信息，只需要传文 spi 的容量大小就可以了
返回值说明	FHA_ SUCCESS	表示操作成功
	FHA_ FAIL	表示操作失败
注意事项	-	
调用示例	-	

2.1.29 读取整个spi数据-- FHA_read_AllDatat

原　　型	T_U32 FHA_read_AllDatat(const T_U8 * pData, T_U32 data_len);	
功能概述	读取整个 spi 数据	
参数说明	const T_U8 * pData,	回读出来的数据
	T_U32 data_len	数据长度
返回值说明	FHA_ SUCCESS	表示操作成功
	FHA_ FAIL	表示操作失败
注意事项	-	
调用示例	-	

以下是 spotlight 的接口需求：

2.1.30 读取隐藏区bin文件的block映射表-- FHA_get_maplist

原　　型	T_U32 FHA_get_maplist(T_U8 file_name[]， T_U16 *map_data, T_U32 *file_len， T_BOOL bBackup)	
功能概述	从隐藏区读取一个自己命名的 bin 文件所在 block 的映射表	
参数说明	file_name	隐藏区文件命名，最长 15 个字节
	map_data	（out）bin 文件 block 映射信息表
	file_len	（out）bin 文件长度
	bBackup	读取原块或者备份块的数据
返回值说明	FHA_ SUCCESS	表示操作成功
	FHA_ FAIL	表示操作失败
注意事项	-	
调用示例	-	

2.2 FSA接口设计

返回值：

#define FSA_SUCCESS 1

#define FSA_FAIL 0

初始化回调接口：

```
typedef  T_hFILE (*FS_FILE_Open)(T_pCSTR path, T_FILE_MODE mode);
typedef  T_BOOL  (*FS_FILE_Close)(T_hFILE hFile);
typedef  T_U32   (*FS_FILE_Read)(T_hFILE hFile, T_pVOID buffer, T_U32 count);
typedef  T_U32   (*FS_FILE_Write)(T_hFILE hFile, T_pVOID buffer, T_U32 count);
typedef  T_S32   (*FS_FILE_Seek)(T_hFILE hFile, T_S32 offset, T_U16 origin);
typedef  T_BOOL  (*FS_FILE_MkDirs)(T_pCSTR path);
typedef struct tag_FsaFileFunc
{
   FSA_FILE_Open       FileOpen;
   FSA_FILE_Close      FileClose;
   FSA_FILE_Read       FileRead;
   FSA_FILE_Write      FileWrite;
   FSA_FILE_Seek       FileSeek;
      FSA_FILE_MkDirs     FsMkDir;
 }T_FSA_FILE_FUNC, *T_PFSA_FILE_FUNC;

typedef T_pVOID (*FSA_RamAlloc)(T_U32 size);
typedef T_pVOID (*FSA_RamFree)(T_pVOID var);
typedef T_pVOID (*FSA_MemSet)(T_pVOID pBuf, T_S32 value, T_U32 count);
typedef T_pVOID (*FSA_MemCpy)(T_pVOID dst, T_pVOID src, T_U32 count);
typedef T_S32   (*FSA_MemCmp)(T_pVOID pbuf1, T_pVOID pbuf2, T_U32 count);
typedef  T_S32 (*FSA_Printf)(T_pCSTR s, ...);
typedef  T_pVOID (*FSA_GetImgMedium)(T_U8 driverID);

typedef tag_FSA_LibCallback
```

```
{
    FSA_RamAlloc RamAlloc;
    FSA_RamFree RamFree;
    FSA_MemSet MemSet;
    FSA_MemCpy MemCpy;
    FSA_MemCmp MemCmp;
    FSA_Printf          Printf;
    FSA_GetImgMedium GetImgMedium;
    T_PFSA_FILE_FUNC pFileFunc;
}T_FSA_LIB_CALLBACK, *T_PFSA_LIB_ CALLBACK;
```

2.2.1 FSA库初始化-- FSA_init

原　　型	T_U32 FSA_init(T_PFSA_LIB_ CALLBACK *pFSA)	
功能概述	烧录时传入隐藏区管理必要的使用环境参数和回调函数	
参数说明	pFSA	库使用回调函数
返回值说明	FSA_SUCCESS	成功
	FSA_FAIL	失败
注意事项	-	
调用示例	-	

说明：该接口主要提供给烧录时传入必要的使用环境参数和回调函数

2.2.2 开始写文件系统区文件-- FSA_write_file_begin

原　　型	T_U32 FSA_write_file_begin(T_UDISK_FILE_INFO *file_info)	
功能概述	传入写隐藏区一个 bin 文件的必要参数	
参数说明	bin_param	Bin 文件信息
返回值说明	FSA_ SUCCESS	表示操作成功
	FSA_ FAIL	表示操作失败
注意事项	-	
调用示例	-	

```
typedef struct
    {
      T_U32 file_length;
      T_U32 file_mode;
      T_U32 file_time;
      T_U8 bCheck;
      T_U8 resv[3];
      T_U8 apath[MAX_PATH+1];
    }T_UDISK_FILE_INFO;
```

2.2.3 写文件系统区文件数据-- FSA_write_file

原　　型	T_U32 FSA_write_file(const T_U8 * pData, T_U32 data_len)	
功能概述	写一段数据 buffer	
参数说明	pData	Bin 文件数据 buffer
	data_len	Bin 文件数据 buffer 长度
返回值说明	FSA_ SUCCESS	表示操作成功
	FSA_ FAIL	表示操作失败
注意事项	-	
调用示例	-	

2.2.4 开始写镜像文件-- FSA_write_image_begin

原　　型	T_U32 FSA_write_image_begin(T_IMG_INFO *img_info)	
功能概述	开始写镜像文件，传入镜像参数，包括数据长度，写入盘符，是否比较	
参数说明	img_info	烧录镜像文件信息
返回值说明	FSA_ SUCCESS	表示操作成功
	FSA_ FAIL	表示操作失败
注意事项	-	
调用示例	-	

```
typedef struct tag_ImgInfo
    {
      T_U32   data_length;
      T_U8    DriverName;
      T_U8    bCheck;          //0 、会找原 meidum 比较再写入，1、 直接拷贝镜像
      T_U8    resv[2];
    }T_IMG_INFO;
```

2.2.5 写镜像文件数据--FSA_write_image

原　　型	T_U32 FSA_write_image(const T_U8 * pData, T_U32 data_len)	
功能概述	写一段数据 buffer	
参数说明	pData	镜像文件数据 buffer
	data_len	镜像文件数据 buffer r 长度
返回值说明	FSA_ SUCCESS	表示操作成功
	FSA_ FAIL	表示操作失败
注意事项	-	
调用示例	-	

3 系统数据结构

3.1.1 AK芯片类型

```
typedef enum
{
   FHA_CHIP_880X,     //AK880X
   FHA_CHIP_10XX,     //AK10XX
   FHA_CHIP_980X,     //AK980X
   FHA_CHIP_37XX,     //AK37XX
   FHA_CHIP_11XX,     //AK11XX
}E_FHA_CHIP_TYPE;
```

3.1.2 烧录模式

```
typedef enum
{
   MODE_NEWBURN = 1,
      MODE_UPDATE,
      MODE_UPDATE_SELF,          //update mode self
}E_BURN_MODE;
```

3.1.3 平台系统类型

```
typedef enum
{
   PLAT_SPOT,         //spot system
       PLAT_SPR,          //spring system
       PLAT_SWORD,    //sword system
       PLAT_LINUX       //linux system
} E_FHA_PLATFORM_TYPE;
```

3.1.4 介质类型

```
typedef enum
{
    MEDIUM_NAND,
    MEDIUM_SPIFLASH,
    MEDIUM_EMMC,
    MEDIUM_SPI_EMMC,
    MEDIUM_EMMC_SPIBOOT,
} E_FHA_MEDIUM_TYPE;
```

3.1.5 回调函数数据类型

```
typedef enum
{
        FHA_DATA_BOOT,
        FHA_DATA_ASA,
       FHA_DATA_BIN
}E_FHA_DATA_TYPE
```

4 版本烧录

现在系统使用的库主要包括：

- 文件系统：fslib 和 mount ；
- MTD
- 驱动库 DRVLIB
- FHA：烧录 BIN 文件,安全区管理
- FSA：烧录文件和镜像

库信息结构体定义为：

```
typedef struct
{
      T_U8 lib_name[10];
```

T_U8 lib_version[40];

}T_LIB_VER_INFO;

lib_name 包括 fslib，mount，mtd，drvlib，fha，fsa。

针对上述库我们烧录的时候会把相应的库版本烧录进去, 然后系统可以读取相应库版本号。目前的应用范围是：nandflash、SD。

4.1.1 往安全区写入库的版本号-- FHA_set_lib_version

原　　型	T_U32 FHA_set_lib_version(T_U8 lib_info[], T_U32 lib_cnt)	
功能概述	往安全区写入库的版本号。	
参数说明	lib_version	库的版本号及名称
	lib_cnt	库版本个数
返回值说明	FHA_ SUCCESS	表示操作成功
	FHA_ FAIL	表示操作失败
注意事项	-	
实现方法	-	

4.1.2 从安全区获取库的版本号-- FHA_get_lib_verison

原　　型	T_U32 FHA_get_lib_verison(T_LIB_VER_INFO *lib_info);	
功能概述	从安全区获取库的版本号	
参数说明	lib_info	结构体成员 lib_name 为传入参数，传入要获取的库名称， lib_version 为获取到的库版本号
返回值说明	FHA_ SUCCESS	表示操作成功
	FHA_ FAIL	表示操作失败
注意事项	-	
实现方法	-	

4.1.3 检测版本号是否兼容-- FHA_check_lib_verison

原　　型	T_U32 FHA_check_lib_verison(T_LIB_VER_INFO *lib_info);	
功能概述	版本定义方式：库名字符串 VmainVersion.sub1Version.sub2Version, 烧录库（fha.a）仅检测倒数第一个版本分隔符之前的字符(.)，需要完全相同，否则版本是不匹配的。因此如果库版本的一般变更可以修改“sub2Version”，但如果重大变更，如涉及版本不兼容现象，需要修改“库名字符串 VmainVersion.sub1Version”，该函数的功能就是从安全区读取版本号，检测安全区版本号与传入的版本号是否兼容、匹配。	
参数说明	lib_info	传入需要检测的版本名称以及库的版本号。
返回值说明	FHA_ SUCCESS	表示匹配
	FHA_ FAIL	表示不匹配
注意事项	-	
实现方法	-	

5 安全区简介

安全区主要是管理坏块，从 1-50 个 block 中找到 10 写有标志位的 block 来作为安全区的块，每次只命中一个 block 来存储坏块表，安全区文件，所以的信息都在一个 block 中，安全区文件主要管理一些客户认为比较重要，而已内容很少，只读类型的小文件，比如：库的版本号就是安全区管理的，还有可以写入密码、序列号等。

坏块表是记载整块 nand 的坏块状况，应用中主要写入某个 block 的时候需要调用来查询是否坏块，也可以在写入、擦除失败的时候设置坏块。