ANYKA
芯 随 你 动

版本: 1.0.0 2015 年 03 月

Sword 37C 中间层接口说明

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	2015 年 03 月

目录

1 概述

中间层封装的接口模块主要包括文件系统、资源管理、字库、频率管理、内存管理、控件模块、媒体库、图片解码库、拍照录像和 App Frame 模块等。

2 文件系统

文件系统的基本概念是指贮存在计算机上的文件和目录。文件系统可以有不同的格式和文件系统类型。文件格式决定信息是如何被贮存为文件和目录。某些文件系统类型贮存重复数据，某些文件系统类型加快硬盘驱动器的存取速度。

文件系统提供的接口如下：

原　　型	T_BOOL　Fwl_InitFs(T_VOID);
功能概述	初始化文件系统
注意事项	开机时调用
返回值说明	T_BOOL AK_TRUE 操作成功 AK_FALSE 操作失败

原　　型	T_VOID　Fwl_DeInitFs(T_VOID);
功能概述	销毁文件系统
注意事项	关机时调用
返回值说明	T_VOID

原　　型	T_pFILE Fwl_FileOpen(T_pCWSTR path, T_FILE_FLAG flag, T_FILE_MODE mode);	
功能概述	打开文件	
参数说明	[in] path	文件完整路径名（unicode 码）
	[in] flag	暂无用
	[in] mode	打开模式（取值：_FMODE_READ、_FMODE_WRITE、_FMODE_CREATE 或 _FMODE_OVERLAY）

原　　型	T_pFILE Fwl_FileOpen(T_pCWSTR path, T_FILE_FLAG flag, T_FILE_MODE mode);
返回值说明	T_pFILE 返回文件句柄 不为－1（_FOPEN_FAIL）操作 成功 －1（_FOPEN_FAIL）操作失败

原　　型	T_U32　Fwl_FileRead(T_pFILE hFile, T_pVOID buffer, T_U32 count);	
功能概述	读文件数据到指定的 buf	
参数说明	[in] hFile	文件句柄（Fwl_FileOpen 返回的）
	[in] buffer	目的 buffer
	[in] count	读数据长度
返回值说明	T_U32 返回实际读的数据长度 >0 操作 成功 0 操作失败	

原　　型	T_U32　Fwl_FileWrite(T_pFILE hFile, T_pCVOID buffer, T_U32 count);	
功能概述	把 buffer 里的数据写入文件	
参数说明	[in] hFile	文件句柄（Fwl_FileOpen 返回的）
	[in] buffer	源 buffer
	[in] count	写数据长度
返回值说明	T_U32 返回实际写的数据长度 >0 操作 成功 0 操作失败	

原　　型	T_BOOL　Fwl_FileClose(T_pFILE hFile);	
功能概述	关闭文件	
参数说明	[in] hFile	文件句柄（Fwl_FileOpen 返回的）

原　　型	T_BOOL Fwl_FileClose(T_pFILE hFile);
返回值说明	T_BOOL AK_TRUE 操作成功 AK_FALSE 操作失败

原　　型	T_S32　Fwl_FileSeek(T_pFILE hFile, T_S32 offset, T_U16 origin);	
功能概述	根据查找原点和偏移量查找文件数据	
参数说明	[in] hFile	文件句柄（Fwl_FileOpen 返回的）
	[in] offset	偏移量
	[in] origin	查找原点（取值：_FSEEK_CUR、_FSEEK_END 或 _FSEEK_SET）
返回值说明	T_S32 返回查找到的数据点 >=0 操作成功 <0　操作失败	

原　　型	T_BOOL Fwl_FileDelete (T_pCWSTR path);	
功能概述	删除文件	
参数说明	[in] path	文件完整路径名（unicode 码）
返回值说明	T_BOOL AK_TRUE 操作成功 AK_FALSE 操作失败	

原　　型	T_U32　Fwl_GetFileLen(T_pFILE hFile);	
功能概述	获取文件长度	
参数说明	[in] hFile	文件句柄（Fwl_FileOpen 返回的）
返回值说明	T_U32 文件长度	

原　　型	T_BOOL Fwl_FileExist(T_pCWSTR path);
功能概述	判断文件是否存在

原　　型	T_BOOL Fwl_FileExist(T_pCWSTR path);	
参数说明	[in] path	文件完整路径名（unicode 码）
返回值说明	T_BOOL AK_TRUE　存在 AK_FALSE 不存在	

以上列出为中间层封装的文件系统常用接口，更多详细情况请参考 platform\firmware\Middle\Include\External_Inc\Fwl_osfs.h。

3 内存管理

内存管理主要协调任务运行时对设备内存资源的分配和使用，以做到内存资源的高效利用和快速分配，并且在适当的时候释放和回收内存资源。

本平台内存管理的主要接口如下：

原　　型	T_VOID Fwl_MallocInit(T_VOID);
功能概述	初始化内存管理系统
返回值说明	T_VOID
注意事项	开机时调用

原　　型	T_pVOID Fwl_MallocAndTrace(T_U32 size, T_pSTR filename, T_U32 line);	
功能概述	申请内存	
参数说明	[in] size	申请的大小
	[in] filename	调用该接口的文件名
	[in] line	调用该接口的文件中的行数
返回值说明	T_pVOID 返回指针	
注意事项	-	
调用示例	-	

原　　型	T_pVOID Fwl_ReMallocAndTrace(T_pVOID var, T_U32 size, T_pSTR filename, T_U32 line);	
功能概述	重新申请内存	
参数说明	[in] var	之前申请内存的指针
	[in] size	重新申请的大小
	[in] filename	调用该接口的文件名
	[in] line	调用该接口的文件中的行数
返回值说明	T_pVOID 返回指针	
注意事项	-	
调用示例	-	

原　　型	T_pVOID Fwl_FreeAndTrace(T_pVOID var, T_pSTR filename, T_U32 line);	
功能概述	释放内存	
参数说明	[in] var	之前申请内存的指针
	[in] filename	调用该接口的文件名
	[in] line	调用该接口的文件中的行数
返回值说明	T_pVOID 返回 AK_NULL	
注意事项	-	
调用示例	-	

注意：以上列出的是中间层封装的内存初始化、申请和释放接口，有关内存调试相关接口应用可参考本文档的“平台内存问题调试参考说明”一节。

4 控件接口说明

4.1 功能概述

控件是用户可与之交互以输入或操作数据的对象，通常出现在对话框中或工具栏上，使用控件可以增强人机交互效果。

下面以 s_ebk_list.c 状态机为例，介绍 displaylist（资源文件列表）控件的用法。

4.2 使用示例

Displaylist 控件的使用示例如下所示：

```
#include "Fwl_public.h"
#include "Ctl_Ebook.h"
……
typedef struct {
   T_DISPLAYLIST          displayList;
   T_MSGBOX               msgbox;
} T_BOOK_LIST_PARM;

static T_BOOK_LIST_PARM *pBook_List_Parm;
/*---------------------- BEGIN OF STATE s_ebk_list ------------------------*/
void initebk_list(void)
{
   T_FILE_TYPE FileType[] = {
      FILE_TYPE_TXT,
      FILE_TYPE_NONE
   };
   pBook_List_Parm = (T_BOOK_LIST_PARM *)Fwl_Malloc(sizeof(T_BOOK_LIST_PARM));
   AK_ASSERT_PTR_VOID(pBook_List_Parm, "initebk_list(): malloc error");
   DisplayList_init(&pBook_List_Parm->displayList, Fwl_GetDefPath(eEBOOK_PATH), \
         GetCustomString(csEBOOK_LIST_TITLE), FileType);       //init 函数里初始化
displaylist
}
void exitebk_list(void)
{
   DisplayList_Free(&pBook_List_Parm->displayList);      //exit 函数里释放 displaylist
   pBook_List_Parm = Fwl_Free(pBook_List_Parm);
}
void paintebk_list(void)
{
```

```
    DisplayList_Show(&pBook_List_Parm->displayList);     //paint 函数里显示 displaylist
    Fwl_RefreshDisplay();
}
unsigned char handleebk_list(T_EVT_CODE event, T_EVT_PARAM *pEventParm)
{
    T_eBACK_STATE DisplayListRet;
    T_FILE_INFO *pFileInfo = AK_NULL;
            if (IsPostProcessEvent(event))
    {
        DisplayList_SetRefresh(&pBook_List_Parm->displayList,
DISPLAYLIST_REFRESH_ALL);//设置刷新
        return 1;
    }
    //handle 函数里调用 displaylist 控件的 handler 接口
    DisplayListRet = DisplayList_Handler(&pBook_List_Parm->displayList, event, pEventParm);
    switch (DisplayListRet)
    {
        case eNext:
            DisplayList_SetRefresh(&pBook_List_Parm->displayList,
DISPLAYLIST_REFRESH_ALL); //设置刷新
            pFileInfo = DisplayList_Operate(&pBook_List_Parm->displayList);//目录操作以及返回
当前项的 fileinfo
            if (pFileInfo != AK_NULL)
            {
                pEventParm = (T_EVT_PARAM *)(&pBook_List_Parm->displayList);
                m_triggerEvent(M_EVT_1, pEventParm);
            }
            break;
        case eMenu:
            pFileInfo = DisplayList_GetItemContentFocus(&pBook_List_Parm->displayList);// 返回
当前项的 fileinfo
            if (pFileInfo != AK_NULL)
```

```
        {
          ……
        }
        break;
    case eReturn:
    case eHome:
        ReturnDefauleProc(DisplayListRet, pEventParm);
        break;
    default:
        break;
  }
  return 0;
}
```

4.3 常用控件接口说明

4.3.1 Topbar控件

4.3.1.1 控件功能

Topbar 是显示在最上面的控件，用于显示界面标题、电池电量、关闭按钮、菜单按钮和后台播放状态等。

4.3.1.2 主要接口

4.3.1.2.1 TopBar_Init

原　　型	T_BOOL TopBar_Init(T_VOID);	
功能概述	Init resource of top bar	
参数说明	T_VOID	-
返回值说明	T_BOOL 类型值	True or not
注意事项	系统共用控件，开机过程调用其初始化。	
调用示例	TopBar_Init();	

4.3.1.2.2 TopBar_EnableShow

原　　型	T_VOID TopBar_EnableShow(T_VOID);	
功能概述	Enable show all the resource of top bar	
参数说明	T_VOID	-
返回值说明	T_VOID	-
注意事项	-	
调用示例	TopBar_EnableShow();	

4.3.1.2.3 TopBar_DisableShow

原　　型	T_VOID TopBar_DisableShow(T_VOID);	
功能概述	Disable show all the resource of top bar	
参数说明	T_VOID	-
返回值说明	T_VOID	-
注意事项	-	
调用示例	TopBar_DisableShow();	

4.3.1.2.4 TopBar_SetTitle

原　　型	T_BOOL TopBar_SetTitle(T_pCWSTR string);	
功能概述	Set title	
参数说明	T_pCWSTR string [in]	Title string
返回值说明	T_BOOL	True or not
注意事项	-	
调用示例	TopBar_SetTitle(GetCustomString(csImg_thumbnail));	

4.3.1.2.5 TopBar_EnableMenuButton

原　　型	T_VOID TopBar_EnableMenuButton(T_VOID);	
功能概述	Enable Menu Button	
参数说明	T_VOID	-
返回值说明	T_VOID	-

原　　型	T_VOID TopBar_EnableMenuButton(T_VOID);
注意事项	-
调用示例	TopBar_EnableMenuButton();

4.3.1.2.6 TopBar_DisableMenuButton

原　　型	T_VOID TopBar_DisableMenuButton(T_VOID);	
功能概述	Disable Memu Button	
参数说明	T_VOID	-
返回值说明	T_VOID	-
注意事项	-	
调用示例	TopBar_DisableMenuButton();	

4.3.1.2.7 TopBar_Show

原　　型	T_BOOL TopBar_Show(T_U16 refresh);	
功能概述	Set show resource	
参数说明	T_U16 refresh [in]	Refresh flag
返回值说明	T_BOOL	True or not
注意事项	-	
调用示例	TopBar_Show(TB_REFRESH_ALL);	

4.3.1.2.8 TopBar_Refresh

原　　型	T_VOID TopBar_Refresh(T_VOID);	
功能概述	Refresh Top Bar resource	
参数说明	T_VOID	-
返回值说明	T_VOID	-
注意事项	-	
调用示例	TopBar_Refresh();	

4.3.1.3 调用流程

```
TopBar_Init(); //初始化
TopBar_EnableShow(); //使能
//设置标题
TopBar_SetTitle(GetCustomString(csImg_thumbnail));
//显示和刷新
TopBar_Show(TB_REFRESH_ALL);
TopBar_Refresh();
```

4.3.2 IconExplorer控件

4.3.2.1 控件功能

IconExplorer 主要用于各种列表式菜单。

4.3.2.2 主要接口

4.3.2.2.1 IconExplorer_Init

原　　型	T_BOOL IconExplorer_Init(T_ICONEXPLORER *pIconExplorer, T_RECT TitleRect, T_RECT ItemRect, T_U32 ItemStyle);	
功能概述	Init IconExplorer	
参数说明	T_ICONEXPLORER *pIconExplorer [in]	IconExplorer handle
	T_RECT TitleRect [in]	Title Rect
	T_RECT ItemRect [in]	Item Rect
	T_U32 ItemStyle [in]	Item Style
返回值说明	T_BOOL	True or not
注意事项	-	
调用示例	IconExplorer_Init(&pDisplayList->IconExplorer, pDisplayList->titleRect, pDisplayList->itemRect, ICONEXPLORER_TITLE_ON | ICONEXPLORER_TITLE_TEXT_HCENTER | ICONEXPLORER_TITLE_TEXT_VCENTER | ICONEXPLORER_ITEM_FRAME);	

4.3.2.2.2 IconExplorer_Free

原　　型	T_BOOL IconExplorer_Free(T_ICONEXPLORER *pIconExplorer);	
功能概述	Free IconExplorer	
参数说明	T_ICONEXPLORER *pIconExplorer [in]	IconExplorer handle
返回值说明	T_BOOL	True or not
注意事项	-	
调用示例	IconExplorer_Free(&pDisplayList->IconExplorer);	

4.3.2.2.3 IconExplorer_Show

原　　型	T_BOOL IconExplorer_Show(T_ICONEXPLORER *pIconExplorer);	
功能概述	show IconExplorer	
参数说明	T_ICONEXPLORER *pIconExplorer [in]	IconExplorer handle
返回值说明	T_BOOL	True or not
注意事项	-	
调用示例	IconExplorer_Show(&pDisplayList->IconExplorer);	

4.3.2.2.4 IconExplorer_Handler

原　　型	T_eBACK_STATE IconExplorer_Handler(T_ICONEXPLORER *pIconExplorer, T_EVT_CODE Event, T_EVT_PARAM *pParam);	
功能概述	Hander func	
参数说明	T_ICONEXPLORER *pIconExplorer [in]	IconExplorer handle
	T_EVT_CODE Event [in]	event
	T_EVT_PARAM *pParam [in]	Event param
返回值说明	T_eBACK_STATE	-
注意事项	-	
调用示例	ret = IconExplorer_Handler(&pDisplayList->IconExplorer, Event, pParam);	

4.3.2.2.5 IconExplorer_SetRefresh

原　　型	T_BOOL IconExplorer_SetRefresh(T_ICONEXPLORER *pIconExplorer, T_U32 RefreshFlag);	
功能概述	Set refresh	
参数说明	T_ICONEXPLORER *pIconExplorer [in]	IconExplorer handle
	T_U32 RefreshFlag [in]	Refresh flag
返回值说明	T_BOOL	True or not
注意事项	-	
调用示例	IconExplorer_SetRefresh(&pDisplayList->IconExplorer, RefreshFlag);	

4.3.2.3 调用流程

```
//初始化
IconExplorer_Init(pIconExplorer, TitleRect, ItemRect, ICONEXPLORER_TITLE_ON | ICONEXPLORER_TITLE_TEXT_HCENTER | ICONEXPLORER_TITLE_TEXT_VCENTER | ICONEXPLORER_ITEM_FRAME);
//设置一些属性，视具体需要而定
IconExplorer_SetTitleRect(pIconExplorer, TitleRect, COLOR_BLUE, pTitleImg);
IconExplorer_SetTitleText(pIconExplorer, AK_NULL, COLOR_BLACK);
IconExplorer_SetItemRect(pIconExplorer, ItemRect, COLOR_WHITE, pItemBackImg);
IconExplorer_SetItemText(pIconExplorer, COLOR_WHITE, COLOR_SKYBLUE, COLOR_BLACK);
IconExplorer_SetItemTransColor(pIconExplorer, g_Graph.TransColor);
IconExplorer_SetItemIconStyle(pIconExplorer, ICONEXPLORER_NONEICON);
IconExplorer_SetNoneIcon(pIconExplorer, ICONEXPLORER_ITEM_NONETEXT_HEIGHT, 1, 4);
IconExplorer_SetScrollBarWidth(pIconExplorer, g_Graph.BScBarWidth);
//显示，通常在状态机的 paint 函数里调用
IconExplorer_Show(pIconExplorer);
//处理事件，通常在状态机的 handle 函数里调用
IconExplorer_Handler(T_ICONEXPLORER *pIconExplorer, T_EVT_CODE Event, T_EVT_PARAM *pParam) ;
//释放
```

```
IconExplorer_Free(T_ICONEXPLORER *pIconExplorer);
```

4.3.3 Displaylist控件

4.3.3.1 控件功能

Displaylist 主要用于各种多媒体资源文件的列表。Displaylist 封装 IconExplorer 的接口。

4.3.3.2 主要接口

4.3.3.2.1 DisplayList_init

原　　型	T_BOOL DisplayList_init(T_DISPLAYLIST *pDisplayList, T_U16 *pCurPath, const T_U16 *pTitleText, T_FILE_TYPE *FileType);	
功能概述	Init Displaylist	
参数说明	T_DISPLAYLIST *pDisplayList [in]	Displaylist handle
	T_U16 *pCurPath [in]	Path str
	const T_U16 *pTitleText [in]	title
	T_FILE_TYPE *FileType [in]	File type
返回值说明	T_BOOL	True or not
注意事项	-	
调用示例	DisplayList_init(&pBook_List_Parm->displayList, Fwl_GetDefPath(eEBOOK_PATH), \ GetCustomString(csEBOOK_LIST_TITLE), FileType);	

4.3.3.2.2 DisplayList_Free

原　　型	T_BOOL DisplayList_Free(T_DISPLAYLIST *pDisplayList);	
功能概述	Free displaylist	
参数说明	T_DISPLAYLIST *pDisplayList [in]	Displaylist handle
返回值说明	T_BOOL	True or not
注意事项	-	

原　　型	T_BOOL DisplayList_Free(T_DISPLAYLIST *pDisplayList);
调用示例	DisplayList_Free(&pBook_List_Parm->displayList);

4.3.3.2.3 DisplayList_Show

原　　型	T_BOOL DisplayList_Show(T_DISPLAYLIST *pDisplayList);	
功能概述	show displaylist	
参数说明	T_DISPLAYLIST *pDisplayList [in]	Displaylist handle
返回值说明	T_BOOL	True or not
注意事项	-	
调用示例	DisplayList_Show(&pBook_List_Parm->displayList);	

4.3.3.2.4 DisplayList_Handler

原　　型	T_eBACK_STATE DisplayList_Handler(T_DISPLAYLIST *pDisplayList, T_EVT_CODE Event, T_EVT_PARAM *pParam);	
功能概述	Hander func	
参数说明	T_DISPLAYLIST *pDisplayList [in]	Displaylist handle
	T_EVT_CODE Event [in]	event
	T_EVT_PARAM *pParam [in]	Event param
返回值说明	T_eBACK_STATE	-
注意事项	-	
调用示例	DisplayListRet = DisplayList_Handler(&pBook_List_Parm->displayList, event, pEventParm);	

4.3.3.2.5 DisplayList_Operate

原　　型	T_FILE_INFO *DisplayList_Operate(T_DISPLAYLIST *pDisplayList);	
功能概述	Operate func	
参数说明	T_DISPLAYLIST *pDisplayList [in]	Displaylist handle
返回值说明	T_FILE_INFO *	Fileinfo
注意事项	-	-
调用示例	pFileInfo = DisplayList_Operate(&pBook_List_Parm->displayList);	

4.3.3.2.6 DisplayList_SetRefresh

原　　型	T_BOOL DisplayList_SetRefresh(T_DISPLAYLIST *pDisplayList, T_U32 RefreshFlag);	
功能概述	Set refresh	
参数说明	T_DISPLAYLIST *pDisplayList [in]	Displaylist handle
	T_U32 RefreshFlag [in]	Refresh flag
返回值说明	T_BOOL	True or not
注意事项	-	
调用示例	DisplayList_SetRefresh(&pBook_List_Parm->displayList, DISPLAYLIST_REFRESH_ALL);	

4.3.3.2.7 DisplayList_GetItemContentFocus

原　　型	T_VOID *DisplayList_GetItemContentFocus(T_DISPLAYLIST *pDisplayList);	
功能概述	Get focus item content	
参数说明	T_DISPLAYLIST *pDisplayList [in]	Displaylist handle
返回值说明	T_VOID *	pFileInfo
注意事项	-	
调用示例	pFileInfo = DisplayList_GetItemContentFocus(&pBook_List_Parm->displayList);	

4.3.3.3 调用流程

Displaylist 调用流程详见 7.4.2 章节。

4.3.4 IconMenu控件

4.3.4.1 控件功能

IconMenu 主要用于待机界面的九宫格式主菜单。

4.3.4.2 主要接口

4.3.4.2.1 IconMenu_Init

原　　型	T_BOOL IconMenu_Init(T_ICONMENU *pIconMenu, T_RECT IconAttachRect, T_RECT IconRect, T_RECT scbarRect);	
功能概述	Init IconMenu	
参数说明	T_ICONMENU *pIconMenu [in]	IconMenu handle
	T_RECT IconAttachRect [in]	item text show rect
	T_RECT IconRect [in]	icon show rect
	T_RECT scbarRect [in]	Scbar rect
返回值说明	T_BOOL	True or not
注意事项	-	
调用示例	IconMenu_Init(&pStdbParm->IconMenu, pStdbParm->stdbRes.IconAttachtRect, pStdbParm->stdbRes.iconRect, pStdbParm->stdbRes.scBarRect);	

4.3.4.2.2 IconMenu_Free

原　　型	T_BOOL IconMenu_Free(T_ICONMENU *pIconMenu);	
功能概述	Free IconExplorer	
参数说明	T_ICONMENU *pIconMenu [in]	IconMenu handle
返回值说明	T_BOOL	True or not
注意事项	-	
调用示例	IconMenu_Free(&pStdbParm->IconMenu);	

4.3.4.2.3 IconMenu_Show

原　　型	T_BOOL IconMenu_Show(T_ICONMENU *pIconMenu);	
功能概述	show IconExplorer	
参数说明	T_ICONMENU *pIconMenu [in]	IconMenu handle
返回值说明	T_BOOL	True or not
注意事项	-	-
调用示例	IconMenu_Show(&pStdbParm->IconMenu);	

4.3.4.2.4 IconMenu_Handler

原　　型	T_eBACK_STATE IconMenu_Handler(T_ICONMENU *pIconMenu, T_EVT_CODE Event, T_EVT_PARAM *pParam);	
功能概述	Hander func	
参数说明	T_ICONMENU *pIconMenu [in]	IconMenu handle
	T_EVT_CODE Event [in]	event
	T_EVT_PARAM *pParam [in]	Event param
返回值说明	T_eBACK_STATE	-
注意事项	-	
调用示例	IconMenuRet = IconMenu_Handler(&pStdbParm->IconMenu, event, pEventParm);	

4.3.4.2.5 IconMenu_AddItem

原　　型	T_BOOL IconMenu_AddItem(T_ICONMENU *pIconMenu, T_U8 Id, T_U8 Place, const T_U16 *pItemText);	
功能概述	Add item	
参数说明	T_ICONMENU *pIconMenu [in]	IconMenu handle
	T_U8 Id [in]	Item id
	T_U8 Place	Place to add
	Const T_U16 *pItemText	Item text
返回值说明	T_BOOL	True or not
注意事项	-	
调用示例	IconMenu_AddItem(pIconMenu, (T_U8)ItemID[i], i, text[i]);	

4.3.4.3 调用流程

```
//初始化
IconMenu_Init(&pStdbParm->IconMenu, pStdbParm->stdbRes.IconAttachtRect, pStdbParm->stdbRes.iconRect, pStdbParm->stdbRes.scBarRect);
//设置一些属性、参数，视具体情况而定
IconMenu_SetIconAttachStyle(&pStdbParm->IconMenu, ICONMENU_ALIGN_HCENTER | ICONMENU_ALIGN_VCENTER, COLOR_BLACK, COLOR_BLUE, AK_NULL, AK_NULL);
IconMenu_SetOtherShowCallBack(&pStdbParm->IconMenu, stdbShwOther);
```

```
IconMenu_AddOtherRect(&pStdbParm->IconMenu, pStdbParm->stdbRes.TopBarRect);
IconMenu_SetItemMatrix(&pStdbParm->IconMenu, 12, 1);
IconMenu_SetItemFocus(&pStdbParm->IconMenu, 2);
//添加 item
for (i=0; i<eMAIN_ICON_MAX_NUM; i++) {
    IconMenu_AddItem(pIconMenu, (T_U8)ItemID[i], i, text[i]);
    for (j=0; j<ICONMENU_ICON_NUM; j++)
    {
      pIconImg = (T_U8 *)Res_GetBinResByID(AK_NULL, AK_TRUE, (iconIDs[i] + j),
&imgLen);
      if (pIconImg != AK_NULL)
        IconMenu_SetItemIcon(pIconMenu, (T_U8)ItemID[i], j, pIconImg);
    }
}
//显示，通常在状态机的 paint 函数里调用
IconMenu_Show(&pStdbParm->IconMenu);
//处理事件，通常在状态机的 handle 函数里调用
IconMenuRet = IconMenu_Handler(&pStdbParm->IconMenu, event, pEventParm);
//释放
IconMenu_Free(&pStdbParm->IconMenu);
```

4.3.5 Msgbox控件

4.3.5.1 控件功能

Msgbox 主要用于各种消息提示框。在用户操作的过程中，当系统需要根据用户的选择，来进行下一步操作的时候，便会弹出相应的界面提示。

4.3.5.2 主要接口

4.3.5.2.1 MsgBox_InitAfx

原　　型	T_VOID MsgBox_InitAfx(T_MSGBOX *mbox, T_U16 retLevel, T_S16 tID, T_S16 sID, T_U16 mode);
功能概述	Init msgbox

原　　型	T_VOID MsgBox_InitAfx(T_MSGBOX *mbox, T_U16 retLevel, T_S16 tID, T_S16 sID, T_U16 mode);	
参数说明	T_MSGBOX *mbox [in]	Msgbox handle
	T_U16 retLevel [in]	Return level
	T_S16 tID [in]	Title str id
	T_S16 sID [in]	Content str id
	T_U16 mode [in]	mode
返回值说明	T_VOID	-
注意事项	-	
调用示例	MsgBox_InitAfx(&pImg_Browser_Parm->msgbox, 2, ctHINT, csFILE_INVALID, MSGBOX_INFORMATION);	

4.3.5.2.2 MsgBox_InitStr

原　　型	T_VOID MsgBox_InitStr(T_MSGBOX *mbox, T_U16 retLevel, T_pCWSTR title, T_pCWSTR content, T_U16 mode);	
功能概述	Init msgbox	
参数说明	T_MSGBOX *mbox [in]	Msgbox handle
	T_U16 retLevel [in]	Return level
	T_pCWSTR title [in]	Title str
	T_pCWSTR content [in]	Content str
	T_U16 mode [in]	mode
返回值说明	T_VOID	
注意事项		
调用示例	MsgBox_InitStr(&pGameParm->msgbox, 1, GetCustomTitle(ctGAME_INFO), GetCustomString(csGAME_HELP_BOAT), MSGBOX_INFORMATION | MSGBOX_EXIT);	

4.3.5.2.3 MsgBox_Show

原　　型	T_VOID MsgBox_Show(T_MSGBOX *mbox);	
功能概述	show msgbox	
参数说明	T_MSGBOX *mbox [in]	Msgbox handle

原　　型	T_VOID MsgBox_Show(T_MSGBOX *mbox);	
返回值说明	T_VOID	-
注意事项	-	
调用示例	MsgBox_Show(&pEbk_Delete_Parm->msgbox);	

4.3.5.2.4 MsgBox_Handler

原　　型	T_eBACK_STATE MsgBox_Handler(T_MSGBOX *mbox, T_EVT_CODE Event, T_EVT_PARAM *pParam);	
功能概述	Hander func	
参数说明	T_MSGBOX *mbox [in]	Msgbox handle
	T_EVT_CODE Event [in]	event
	T_EVT_PARAM *pParam [in]	Event param
返回值说明	T_eBACK_STATE	-
注意事项	-	
调用示例	menuRet = MsgBox_Handler(&pEbk_Delete_Parm->msgbox, event, pEventParm);	

4.3.5.2.5 MsgBox_AddLine

原　　型	T_BOOL MsgBox_AddLine(T_MSGBOX *mbox, T_pCWSTR uText);	
功能概述	Add line	
参数说明	T_MSGBOX *mbox [in]	Msgbox handle
	T_pCWSTR uText [in]	Text str
返回值说明	T_BOOL	True or not
注意事项	-	
调用示例	MsgBox_AddLine(mbox, GetCustomString(sID));	

4.3.5.3 调用流程

1、触发 M_EVT_MESSAGE 事件，到公共消息提示状态机显示。

```
//初始化
```

```
MsgBox_InitAfx(&pThumbnail_Param->msgbox, 2, ctHINT, csFILE_INVALID,
MSGBOX_INFORMATION);
//设置显示时长
MsgBox_SetDelay(&pThumbnail_Param->msgbox, MSGBOX_DELAY_1);
//触发 M_EVT_MESSAGE 事件
m_triggerEvent(M_EVT_MESSAGE, (T_EVT_PARAM *)&pThumbnail_Param->msgbox);
```

2、在 msgbox 所在的状态机显示。

```
//初始化
MsgBox_InitAfx(&pGameLoadParm->msgbox, 0, ctHINT, csGAME_LOAD_NOTE,
MSGBOX_QUESTION | MSGBOX_YESNO);
//显示，在状态机 paint 函数里调用
MsgBox_Show(&pGameLoadParm->msgbox);
//处理事件，在状态机的 handle 函数里调用
msgRet = MsgBox_Handler(&pGameLoadParm->msgbox, event, pEventParm);
```

4.3.6 ToolBar控件

4.3.6.1 控件功能

ToolBar 主要用于工具栏，如照相机模块。

4.3.6.2 主要接口

4.3.6.2.1 ToolBar_Init

原　　型	T_BOOL ToolBar_Init(T_pTOOLBAR pToolBar, T_TB_DIRECTION Direction, T_LEN windowWidth, T_LEN windowHeight, T_U16 ButtonInterval, T_U16 ButtonWidth, T_U16 ButtonHight, T_TB_SHOWN_MODE ShownMode, T_COLOR FontColor, T_COLOR BkGrndColor, T_U8 Trans);	
功能概述	Init toolbar	
参数说明	T_pTOOLBAR pToolBar [in]	toolBar handle
	T_TB_DIRECTION Direction [in]	Direction
	T_LEN windowWidth [in]	windowWidth
	T_LEN windowHeight [in]	windowHeight

原　　型	T_BOOL ToolBar_Init(T_pTOOLBAR pToolBar, T_TB_DIRECTION Direction, T_LEN windowWidth, T_LEN windowHeight, T_U16 ButtonInterval, T_U16 ButtonWidth, T_U16 ButtonHight, T_TB_SHOWN_MODE ShownMode, T_COLOR FontColor, T_COLOR BkGrndColor, T_U8 Trans);	
	T_U16 ButtonInterval [in]	the interval of two buttons
	T_U16 ButtonWidth [in]	ButtonWidth
	T_U16 ButtonHight [in]	ButtonHight
	T_TB_SHOWN_MODE ShownMode [in]	To be shown on YUV data or RGB data
	T_COLOR FontColor [in]	Font color
	T_COLOR BkGrndColor [in]	ToolBar back ground color
	T_U8 Trans [in]	Trans :0-15
返回值说明	T_BOOL	True or not
注意事项	-	
调用示例	ToolBar_Init(pPreview->pToolBar, TB_eBOTTOM, MAIN_LCD_WIDTH, MAIN_LCD_HEIGHT, CAM_BUTTON_INTERVAL, CAM_BUTTON_WIDTH, CAM_BUTTON_HEIGHT, TB_eMODE_SHOWN_NORMAL, CAM_ToolBar_FontColor, CAM_ToolBar_BACKGROUND_COLOR, CAM_TB_TRANS_HALF);	

4.3.6.2.2 ToolBar_Free

原　　型	T_VOID ToolBar_Free(T_pTOOLBAR pToolBar);	
功能概述	Free toolbar	
参数说明	T_pTOOLBAR pToolBar [in]	toolBar handle
返回值说明	T_VOID	-
注意事项	-	
调用示例	ToolBar_Free(pPreview->pToolBar);	

4.3.6.2.3 ToolBar_Show

原　　型	T_VOID ToolBar_Show(T_pTOOLBAR pToolBar);
功能概述	show toolbar

参数说明	T_pTOOLBAR pToolBar [in]	toolBar handle
返回值说明	T_VOID	-
注意事项	-	
调用示例	ToolBar_Show(pPreview->pToolBar);	

4.3.6.2.4 ToolBar_Handler

原 型	T_eBACK_STATE ToolBar_Handler(T_pTOOLBAR pToolBar, T_EVT_CODE event, T_EVT_PARAM *pParam);	
功能概述	Hander func	
参数说明	T_pTOOLBAR pToolBar [in]	toolBar handle
	T_EVT_CODE Event [in]	event
	T_EVT_PARAM *pParam [in]	Event param
返回值说明	T_eBACK_STATE	-
注意事项	-	
调用示例	retState = ToolBar_Handler(pPreview->pToolBar, event, pEventParm);	

4.3.6.2.5 ToolBar_AddButton

原 型	T_pBUTTON ToolBar_AddButton(T_pTOOLBAR pToolBar, T_U32 ButtonId, T_BUTTON_TYPE Type, T_pCWSTR Name, T_pSTATEICON_DATA pButtonIcon);	
功能概述	Add button	
参数说明	T_pTOOLBAR pToolBar [in]	toolBar handle
	T_U32 ButtonId [in]	Button Id
	T_BUTTON_TYPE Type [in]	Button type
	T_pCWSTR Name [in]	Button name
	T_pSTATEICON_DATA pButtonIcon [in]	Button Icon
返回值说明	T_pBUTTON	-
注意事项	-	

原　　型	T_pBUTTON ToolBar_AddButton(T_pTOOLBAR pToolBar, T_U32 ButtonId, T_BUTTON_TYPE Type, T_pCWSTR Name, T_pSTATEICON_DATA pButtonIcon);
调用示例	ToolBar_AddButton(pPreview->pToolBar, pPreview->ButtonIcon.CapRecSwitch.Id, \ BTN_TYPE_SWITCH, pPreview->ButtonIcon.CapRecSwitch.Name, \ pPreview->ButtonIcon.CapRecSwitch.stateIcon);

4.3.6.3　调用流程

```
//初始化
ToolBar_Init(pPreview->pToolBar, TB_eBOTTOM, MAIN_LCD_WIDTH, MAIN_LCD_HEIGHT,
CAM_BUTTON_INTERVAL, CAM_BUTTON_WIDTH, CAM_BUTTON_HEIGHT,
TB_eMODE_SHOWN_NORMAL, CAM_ToolBar_FontColor,
CAM_ToolBar_BACKGROUND_COLOR, CAM_TB_TRANS_HALF);
// 添加 button，设置 button 选项等
ToolBar_AddButton(pPreview->pToolBar, pPreview->ButtonIcon.CapRecSwitch.Id, \
        BTN_TYPE_SWITCH, pPreview->ButtonIcon.CapRecSwitch.Name, \
        pPreview->ButtonIcon.CapRecSwitch.stateIcon);
ToolBar_AddOptionToButton(pPreview->pToolBar, pPreview->ButtonIcon.CapRecSwitch.Id, \
        CAM_DC, Res_GetStringByID(eRES_STR_CMR_SWITCH_TO_DV));
ToolBar_AddOptionToButton(pPreview->pToolBar, pPreview->ButtonIcon.CapRecSwitch.Id, \
        CAM_DV, Res_GetStringByID(eRES_STR_CMR_SWITCH_TO_DC));
ToolBar_SetFocusOption(pPreview->pToolBar, pPreview->ButtonIcon.CapRecSwitch.Id,
gs.CamMode);
//显示，通常在状态机的 paint 函数里调用
ToolBar_Show(pPreview->pToolBar);
//处理事件，通常在状态机的 handle 函数里调用
retState = ToolBar_Handler(pPreview->pToolBar, event, pEventParm);
//释放
ToolBar_Free(pPreview->pToolBar);
```

4.3.7 WaitBox控件

4.3.7.1 控件功能

WaitBox 主要用于各种请稍后之类的等待框。

4.3.7.2 主要接口

4.3.7.2.1 WaitBox_Init

原　　型	T_VOID WaitBox_Init(T_VOID);	
功能概述	Init waitbox	
参数说明	T_VOID	-
返回值说明	T_VOID	-
注意事项	系统共用控件，开机过程调用其初始化。	
调用示例	WaitBox_Init();	

4.3.7.2.2 WaitBox_Free

原　　型	T_VOID WaitBox_Free(T_VOID);	
功能概述	Free waitbox	
参数说明	T_VOID	-
返回值说明	T_VOID	-
注意事项	-	
调用示例	-	

4.3.7.2.3 WaitBox_Start

原　　型	T_VOID WaitBox_Start(T_eWAITBOX_MODE mode, T_pWSTR title);	
功能概述	Start waitbox	
参数说明	T_eWAITBOX_MODE mode [in]	mode
	T_pWSTR title [in]	Title str
返回值说明	T_VOID	-
注意事项	-	

原　　型	T_VOID WaitBox_Start(T_eWAITBOX_MODE mode, T_pWSTR title);
调用示例	WaitBox_Start(WAITBOX_RAINBOW, (T_pWSTR)GetCustomString(csLOADING));

4.3.7.2.4 WaitBox_Stop

原　　型	T_BOOL WaitBox_Stop(T_VOID);	
功能概述	Stop waitbox	
参数说明	T_VOID	-
返回值说明	T_BOOL	-
注意事项	-	
调用示例	WaitBox_Stop();	

4.3.7.2.5 WaitBox_Show

原　　型	T_VOID WaitBox_Show(T_eWAITBOX_MODE mode);	
功能概述	Show waitbox	
参数说明	T_eWAITBOX_MODE mode　[in]	mode
返回值说明	T_VOID	-
注意事项	-	
调用示例	WaitBox_Show(pEvtParam->w.Param1);	

4.3.7.3 调用流程

```
//初始化
WaitBox_Init();
//启动等待框
WaitBox_Start(WAITBOX_RAINBOW, (T_pWSTR)GetCustomString(csLOADING));
//显示等待框，目前是在共用背景线程 CPublicBGApp_ICBThread_Handle 里调用
WaitBox_Show(WAITBOX_RAINBOW);
//停止显示等待框
WaitBox_Stop();
```

5 音视频播放接口说明

5.1 功能概述

音视频播放接口 给应用提供了媒体文件的播放， 封装了播放的流程。相比直接使用多媒体库的接口，简化了许多流程的控制。

5.2 函数接口说明

5.2.1 媒体查询函数接口

原　　型	T_BOOL Media_HasAudio(T_pVOID fname, T_BOOL isFile);	
功能概述	Query Media Whether Has Audio	
参数说明	fname	[in] pointer of File Name Or Buffer Initial Address
	isFile	[in] Media Is A File Or Buffer
返回值说明	T_BOOL	AK_FALSE　　NOT
		AK_TRUE　　YES

原　　型	T_BOOL Media_HasVideo(T_pVOID fname, T_BOOL isFile);	
功能概述	Query Media Whether Has Video	
参数说明	fname	[in] pointer of File Name Or Buffer Initial Address
	isFile	[in] Media Is A File Or Buffer
返回值说明	T_BOOL	AK_FALSE　　NOT
		AK_TRUE　　YES

原　　型	T_AUDIO_TYPE Media_GetAudioType(T_pVOID fname, T_BOOL isFile);	
功能概述	Query Media Audio Encoded Type	
参数说明	fname	[in] pointer of File Name Or Buffer Initial Address
	isFile	[in] Media Is A File Or Buffer
返回值说明	T_AUDIO_TYPE	

原　　型	T_U8 Media_GetVideoType(T_pVOID fname, T_BOOL isFile);
功能概述	Query Media Video Encoded Type

原　　型	T_U8 Media_GetVideoType(T_pVOID fname, T_BOOL isFile);	
参数说明	fname	[in] pointer of File Name Or Buffer Initial Address
	isFile	[in] Media Is A File Or Buffer
返回值说明	T_eVIDEO_DRV_TYPE	

原　　型	T_eMEDIALIB_MEDIA_TYPE Media_GetMediaType(T_pVOID fname, T_BOOL isFile);	
功能概述	Query Media File Header Type	
参数说明	fname	[in] pointer of File Name Or Buffer Initial Address
	isFile	[in] Media Is A File Or Buffer
返回值说明	T_eMEDIALIB_MEDIA_TYPE	

原　　型	T_MEDIALIB_STRUCT Media_GetMetaInfo(T_MEDIALIB_META_INFO **ppMetaInfo, T_pVOID fname, T_BOOL isFile);	
功能概述	Get Media File Meta Infomation	
参数说明	ppMetaInfo	[out] pointer of Meta Infomation
	fname	[in] pointer of File Name Or Buffer Initial Address
	isFile	[in] Media Is A File Or Buffer
注意事项	After Used, Must Call MediaLib_Dmx_Close(T_MEDIALIB_STRUCT) Close Media Lib Handler	
返回值说明	T_MEDIALIB_STRUCT AK_NULL　　Failure Others　　Success	

原　　型	T_pVOID Media_GetPicMetaInfo(T_USTR_FILE filename, T_U8 **picBuf, T_U32 *picLen)	
功能概述	Get Media File Meta Picture Data	
参数说明	fname	[in] pointer of File Name
	picBuf	[out] pointer of Meta Picture Data
	picLen	[out] Length of Meta Picture Data

原　　型	T_pVOID Media_GetPicMetaInfo(T_USTR_FILE filename, T_U8 **picBuf, T_U32 *picLen)
注意事项	After Used, Must Call Media_ReleasePicMetaInfo(T_pVOID pMetapic) to Release Memory
返回值说明	T_pVOID Parameter of Function Media_ReleasePicMetaInfo() AK_NULL Failure Others Success

原　　型	T_VOID Media_ReleasePicMetaInfo(T_pVOID pMetapic);	
功能概述	Release Media File Meta Picture Data	
参数说明	pMetapic	[in] Return Value of Function Media_GetPicMetaInfo()
返回值说明	T_pVOID	

5.2.2 媒体播放器函数接口

原　　型	T_BOOL MPlayer_Init(T_VOID);
功能概述	Initialize MediaLib & Media Player Struct
返回值说明	AK_FALSE Initialized fail AK_TRUE Initialized ok

原　　型	T_BOOL MPlayer_Open(T_pVOID fname, T_BOOL isFile);	
功能概述	Open Media File	
参数说明	pMetapic	[in] pointer of File Name Or Buffer Initial Address
	isFile	[in] The Media Is a File Or Buffer
返回值说明	T_BOOL AK_FALSE Open Failure AK_TRUE Open Success	

原　　型	T_BOOL MPlayer_Play(T_U32 position);	
功能概述	Play Current Opened Media File	
参数说明	position	[in] Played Position of Media

原　　型	T_BOOL MPlayer_Play(T_U32 position);
返回值说明	T_BOOL AK_FALSE　Open Failure AK_TRUE　Open Success

原　　型	T_BOOL MPlayer_Close(T_VOID);
功能概述	Close Current Media Player
返回值说明	T_BOOL AK_FALSE　Close Failure AK_TRUE　Close Success

原　　型	T_VOID MPlayer_SetShowFrameCB(T_fSHOWFRAME_CB pShowFrame);	
功能概述	Set Current Media Player Show Callback Function	
参数说明	pShowFrame	[in] pointer of Show Callback Function
返回值说明	T_VOID	

原　　型	T_VOID MPlayer_SetEndCB(T_fEND_CB endCB);	
功能概述	Set Current Media Player End Callback Function	
参数说明	endCB	[in] pointer of End Callback Function
返回值说明	T_VOID	

原　　型	T_BOOL MPlayer_Start(T_U32 pos);	
功能概述	Start Playing Media After Stoped Playing	
参数说明	pos	[in] Played Position of Media
返回值说明	T_BOOL AK_FALSE　Start Failure AK_TRUE　Start Success	

原　　型	T_BOOL MPlayer_Pause(T_VOID);
功能概述	Pause Current Media Player
返回值说明	T_BOOL AK_FALSE　Pause Failure AK_TRUE　Pause Success

原　　型	T_BOOL MPlayer_Resume(T_VOID);
功能概述	Resume Current Media Player
返回值说明	T_BOOL AK_FALSE　Resume Failure AK_TRUE　Resume Success

原　　型	T_BOOL MPlayer_Stop(T_VOID);
功能概述	Stop Current Media Player
返回值说明	T_BOOL AK_FALSE　Stop Failure AK_TRUE　Stop Success

原　　型	T_BOOL MPlayer_Seek(T_U32 param);
功能概述	From Special Position Play
参数说明	Param　[in] Playing Position
返回值说明	T_BOOL AK_FALSE　Seek Failure AK_TRUE　Seek Success

原　　型	T_eMPLAYER_STATUS MPlayer_GetStatus(T_VOID);
功能概述	Manipulate Current Media Player End Callback Function
参数说明	T_VOID
返回值说明	T_eMPLAYER_STATUS

原　　型	T_U32 MPlayer_GetTotalTime(T_VOID);
功能概述	Get Current Playing Media Total Time
返回值说明	T_U32 0　　Failure Others　　Sucess

原　　型	T_U32 MPlayer_GetCurTime(T_VOID);
功能概述	Get Current Playing Media Current Time
返回值说明	T_U32 0　　Failure Others　　Sucess

原　　型	T_eMEDIALIB_MEDIA_TYPE MPlayer_GetMediaType(T_VOID);
功能概述	Get Current Playing Media Type
返回值说明	T_eMEDIALIB_MEDIA_TYPE

原　　型	T_pVOID MPlayer_GetMediaInfo(T_VOID);
功能概述	Get Current Playing Media Infomation
返回值说明	T_pVOID(T_MEDIALIB_DMX_INFO*) AK_NULL　　Failure Others　　Success

原　　型	T_BOOL MPlayer_HasAudio(T_VOID);
功能概述	Query Current Playing Media Whether Has Audio
返回值说明	T_BOOL AK_FALSE　　NOT AK_TRUE　　YES

原　　型	T_BOOL MPlayer_HasVideo(T_VOID);
功能概述	Query Current Playing Media Whether Has Video
返回值说明	T_BOOL AK_FALSE　　NOT AK_TRUE　　YES

原　　型	T_BOOL MPlayer_AllowSeek(T_VOID);
功能概述	Query Current Playing Media Whether Allow Seek
返回值说明	T_BOOL AK_FALSE　　NOT AK_TRUE　　YES

原　　型	T_LEN MPlayer_GetWidth(T_VOID);
功能概述	Get Current Playing Video Width
返回值说明	T_LEN -1　　Failure Others　Success

原　　型	T_LEN MPlayer_GetHeight(T_VOID);
功能概述	Get Current Playing Video Height
返回值说明	T_LEN -1　　Failure Others　Success

原　　型	T_BOOL MPlayer_SetChannel(T_U8 channel);	
功能概述	Set Current Playing Media Channel	
参数说明	channel	[in] Channel Number
返回值说明	T_BOOL AK_FALSE　　Failure　　AK_TRUE　　Success	

原　　型	T_pVOID MPlayer_GetAudioDecoder(T_VOID);
功能概述	Get Current Player Audio Decoder Pointer
返回值说明	T_pVOID AK_NULL　　Failure Others　　Success

原　　型	T_pVOID MPlayer_GetVideoDecoder(T_VOID);
功能概述	Get Current Player Video Decoder Pointer
返回值说明	T_pVOID AK_NULL　　Failure Others　　Success

原　　型	T_BOOL MPlayer_GetFrameYuv(T_pVOID fname, T_BOOL isFile, T_pDATA pBuf, T_LEN width, T_LEN height, T_S32 pos);	
功能概述	Get Appointed Farme YUV Data From Video File	
参数说明	fname	[in] pointer of File Name Or Buffer Initial Address
	isFile	[in] The Media Is a File Or Buffer
	pBuf	[out] YUV Frame Data
	width	[in] YUV Frame Width
	height	[in] YUV Frame Height
	pos	[in] Appointed Farme (millisecond)
返回值说明	T_BOOL AK_FALSE　　Get YUV Failure AK_TRUE　　Get YUV Success	

5.2.3　媒体播放调用过程样例

媒体播放调用过程样例如下：

```
MPlayer_Init();
if (MPlayer_Open( filename, AK_TRUE))
 {
        MPlayer_SetShowFrameCB(AVIPlayer_ShowFrame);
```

```
        MPlayer_SetEndCB(AVIPlayer_EndOneFile);
        MPlayer_Play(0);
        ... ...
        MPlayer_GetTotalTime();
        MPlayer_GetCurTime();
        MPlayer_HasAudio();
        MPlayer_HasVideo();
        MPlayer_AllowSeek();
        MPlayer_GetWidth();
        MPlayer_GetHeight();
        MPlayer_GetStatus();
        ... ...
        MPlayer_HandleEvt(EVT_PLAY_PAUSE, 0);
        MPlayer_HandleEvt(EVT_PLAY_RESUME, 0);
        MPlayer_HandleEvt(EVT_PLAY_SEEK, 10000);
        ... ...
        MPlayer_Close();
}
```

5.3 系统数据结构定义

5.3.1 BUFFER媒体结构定义

系统中的一些缺省声音可能会数组的形式嵌到代码中，当要播放此类音频数据时，需用此结构封装数据并将结构的首地址传给媒体播放器。

结构体定义如下：

```
/*realize buffer read/seek same as file read/seek*/
typedef struct{
   T_U8        *pBuf;        //!< start addr of buffer
   T_U32       bufLen;       //!< buffer total len
   T_U32       bufPos;       //!< buffer position pointer
}T_MEDIALIB_BUFFER;
```

5.3.2 音频滤波器结构定义

当前系统只支持两种音频滤波，EQ(均衡)和 TEMPO(快慢速度播放)。在选择音效时只能选择一种，不能既做 EQ 又做 TEMPO。结构体定义如下：

```
typedef struct tag_AudioFilter{
    struct{
    T_pVOID     hFilt;              //!< Audio Lib Filter Handler
    T_U8        mode;               //!< EQ Mode / TEMPO Value(0.5 ~ 1.5)
    }filter[AUDIO_FILTER_NUM]; //!< [0]:eq, [1]:tempo, others:reserve
    T_AUDIO_BUF filtBuf;
    T_U8        flag;               //!< bit0: eq, bit1: tempo, others: reserve
}T_AUDIO_FILTER, *T_pAUDIO_FILTER;
```

5.3.3 音频解码器结构定义

当播放视频时，取消音频滤波，将成员 pSdFilt 设为 NULL。成员 decNum 的值主要依据音频采样率估出，表示每一次定时器事件最多的解码次数。结构定义如下：

```
typedef struct tag_AudioDecode{
    T_pVOID                 hSD;    //!< Audio Lib Decoder Handler
    T_MEDIALIB_AUDIO_INFO   audioInfo;
    T_AUDIO_DECODE_OUT      decOut;
    T_pAUDIO_FILTER         pSdFilt;
    T_U8                    decNum; //!< Decoded Times In ONE Timer Event
}T_AUDIO_DEC, *T_pAUDIO_DEC;
```

5.3.4 媒体Demuxer结构定义

当播放 BUFFER 类型的媒体时，成员 hFile 的值是 T_MEDIALIB_BUFFER 的地址。成员 startTime 是记录每次播放时媒体的开始时间位置(包括 SEEK 时)，是媒体的播放位置参考点。zeroTime 则是系统的 T ick 时间, 用于在纯视频媒体时的时间参考。curTime 用于记录播放过程中的当前媒体时间。结构定义如下：

```
typedef struct tag_Demuxer{
    T_pVOID             hMedia;     //!< Media Lib Demuxer Handler
    T_MEDIALIB_DMX_INFO dmxInfo;    //!< Media Information
```

```
    T_pVOID             hFile;          //!< File Handler / Buffer Init Address
    T_U32               startTime;      //!< Start Playing Position Per Times
    T_U32               zeroTime;       //!< System Tick, ms
    T_U32               curTime;        //!< Relative Zero, ms
    volatile T_TIMER    timer;          //!< Demuxer Thread Driver Timer
    T_U8                isFile;   //!< Bit 0: Src File Flag, 1: File, 0: Buffer;
}T_DEMUXER, *T_pDEMUXER;
```

5.3.5 视频解码器结构定义

当用 MPU 屏时，有一个子线程专门负责将 YUV 转换成 RGB 并刷新，需将成员 showCB 的值赋给 MPU 子线程的显示回调函数。结构定义如下：

```
typedef struct tag_VidoeDecode{
    T_pVOID                 hVS;            //!< Video Stream Decoder Handler
    T_VIDEO_DECODE_OUT      videoOut;
    T_fSHOWFRAME_CB         showCB;         //!< Vidoe Show Callback Function
    volatile T_TIMER        timer;
}T_VIDEO_DEC, *T_pVIDEO_DEC;
```

5.3.6 媒体播放器结构定义

在系统初始化时，此结构将会实例化为模块静态成员 s_hPlayer，并调用 VideoStream_Init()， 如果成功，成员 init 赋为真，否则赋为假。当成员 endCB 缺省为 NULL 时，播放器在正常或异常结束时自动释放相关资源并终结本身。结构定义如下：

```
typedef struct tag_MultiThreadMediaPlayer{
    T_pDEMUXER              pDmx;           //!< Media Demuxer Pointer
    T_pAUDIO_DEC            pAudio;         //!< Audio Decoder Pointer
    T_pVIDEO_DEC            pVideo;         //!< Video Decoder Pointer
    T_eMPLAYER_STATUS       status;
    T_U8                    init;
    T_fEND_CB               endCB;          //!< End Callback Function
}T_MT_MPLAYER, *T_pMT_MPLAYER;
```

5.3.7 MPU Asynchronous Refresh结构定义

1、链表结点定义

```
typedef struct tag_VidoeOutList{
  T_VIDEO_DECODE_OUT   videoOut; //!< YUV Data's Buffer Address, Width, Height Etc.
  struct tag_VidoeOutList*      next;
  volatile T_U8                 isValid;
}T_VIDEO_OUT_LIST, *T_pVIDEO_OUT_LIST;
```

2、MPU 异步刷新子线程的结构定义

```
typedef struct tag_MpuRefresh{
  ISubThread              *thread;          //!< Refresh Thread Instance
  T_hEventGroup           evtGroup;         //!< Event Group Handler
  T_pVIDEO_OUT_LIST       head;             //!< The Header of Refresh Queue
  T_pVIDEO_OUT_LIST       tail;             //!< The Tailer of Refresh Queue
  T_fSHOWFRAME_CB         pShowFrameCB;     //!< Show A Frame Callback Function
}T_MPU_REFRESH, *T_pMPU_REFRESH;
```

6 图片解码接口说明

6.1 功能概述

图像编解码模块的主要功能是为上层应用层提供 WBMP、JPEG、PNG、GIF 图片解码的功能，JPEG 编码功能以及 RGB 等转换接口。

6.2 接口调用说明

6.2.1 基本调用流程

打开一张资源列表中的图片，图片模块接口调用的基本流程如下：

1、 先初始化 ImgBrowse，通常在状态机里的 init 函数里

```
ImgBrowse_Init(&pImg_Browser_Parm->ImgBrowse);
```

2、 打开 displaylist 里的焦点项图片，通常在状态机的 handle 函数里

```
ImgBrowse_Open(&pImg_Browser_Parm->ImgBrowse, pImg_Browser_Parm->pDisplayList);
```

3、 显示该图片，通常在状态机的 paint 函数里

```
ImgBrowse_Show(&pImg_Browser_Parm->ImgBrowse);
Fwl_RefreshDisplay();
```

4、 退出应用界面时释放 ImgBrowse，通常在状态机的 exit 函数里

```
ImgBrowse_Free(&pImg_Browser_Parm->ImgBrowse);
```

6.2.2 ImgBrowse_Handle接口说明

通常在状态机的 handle 函数里调用，响应按键、触屏、timer 等事件，用于切换下一张图片、旋转、缩放图片、显示 gif 图片的各个帧、退出等。

```
ret = ImgBrowse_Handle(&pImg_Browser_Parm->ImgBrowse, event, pEventParm);
switch(ret)
{
    case IMG_RETURN:
        m_triggerEvent(M_EVT_EXIT, pEventParm);
        break;
    case IMG_RETURN_HOME:
        m_triggerEvent(M_EVT_Z09COM_SYS_RESET, pEventParm);
        break;
    case IMG_MENU:
        ……
        break;
    case IMG_OPEN_ERROR:
    case IMG_MULTIMETHODSHOW_ERROR:
        ……
        break;
    default:
        break;
}
```

6.2.3 ImgBrowse_Open接口说明

ImgBrowse_Open 接口为平台对图象库的各个格式解码接口经过几层封装而来的，图象库的详细编解码接口可参见《图像库接口说明》文档。

6.3 图片模块主要接口

6.3.1 ImgBrowse_Init

原　　型	T_BOOL ImgBrowse_Init(T_IMGBROWSE * pImgBrowse)	
功能概述	Init ImgBrowse	
参数说明	T_IMGBROWSE * pImgBrowse	[in]ImgBrowse handle
返回值说明	T_BOOL	True or not
注意事项	-	
调用示例	-	

6.3.2 ImgBrowse_Free

原　　型	T_VOID ImgBrowse_Free(T_IMGBROWSE * pImgBrowse)	
功能概述	Free ImgBrowse	
参数说明	T_IMGBROWSE * pImgBrowse [in]	ImgBrowse handle
返回值说明	T_VOID	
注意事项	-	
调用示例	-	

6.3.3 ImgBrowse_Open

原　　型	T_BOOL ImgBrowse_Open(T_IMGBROWSE * pImgBrowse, T_DISPLAYLIST *pDisplayList)	
功能概述	Open a picture from the displaylist	
参数说明	T_IMGBROWSE * pImgBrowse	[in]ImgBrowse handle
	T_DISPLAYLIST *pDisplayList	Displaylist handle
返回值说明	T_BOOL	True or not

注意事项	-
调用示例	-

6.3.4 ImgBrowse_Show

原　　型	T_BOOL ImgBrowse_Show(T_IMGBROWSE * pImgBrowse)	
功能概述	Show picture	
参数说明	T_IMGBROWSE * pImgBrowse	[in]ImgBrowse　handle
返回值说明	T_BOOL	True or not
注意事项	-	
调用示例	-	

6.3.5 ImgBrowse_Handle

原　　型	T_IMG_RET ImgBrowse_Handle(T_IMGBROWSE * pImgBrowse, T_EVT_CODE event, T_EVT_PARAM *pEventParm)	
功能概述	Hander func	
参数说明	T_IMGBROWSE * pImgBrowse　[in]	ImgBrowse　handle
	T_EVT_CODE Event [in]	event
	T_EVT_PARAM *pParam [in]	Event param
返回值说明	T_IMG_RET	-
注意事项	-	
调用示例	-	

6.4 图像库接口说明

有关图像库的详细接口说明请用户参见《图像库接口说明》文档。

6.5 其他转换接口说明

6.5.1 平台封装的2D转换接口

6.5.1.1 Fwl_ScaleConvert

原　　型	T_U8 Fwl_ScaleConvert(T_U32 *iBuf, T_U32 niBuf, T_U16 srcW, T_pRECT srcWin, E_ImageFormat formatIn, T_U32* oBuf, T_U8 noBuf, T_U16 dstW, T_pRECT dstWin, E_ImageFormat formatOut, T_U8 luminanceEn, T_U8* luminanceTab);	
功能概述	image scale and format convert function, support format:yuv420, rgb565	
参数说明	iBuf <in>	src buffer address arrays
	niBuf <in>	num of src buffer address array.
	srcW <in>	src image width
	srcWin <in>	src rect
	formatIn <in >	src image format: FORMAT_YUV420, FORMAT_RGB565
	oBuf <in>	dst buffer address arrays
	noBuf <in>	num of dst buffer address array
	dstW <in>	dst image width
	dstWin <in>	dst rect
	formatOut <in>	dst image format:FORMAT_YUV420, FORMAT_RGB565
	luminanceEn <in>	if input image format is FORMAT_YUV420, this param can set AK_TRUE to transform luminance.
	luminanceTab <in>	a continuous memory buffer width length of 256 byte, used to transform yuv420 luminance.
返回值说明	T_U8	0：成功；非 0：失败
注意事项	支持 yuv420 和 rgb565 格式的。若是 rgb888 格式，要先转换成 rgb565 格式才能用此接口。	
调用示例	-	

6.5.1.2 Fwl_ScaleConvertNoBlock

原　　型	T_U8 Fwl_ScaleConvertNoBlock(T_U32 *iBuf, T_U32 niBuf, T_U16 srcW, T_pRECT srcWin, E_ImageFormat formatIn, T_U32* oBuf, T_U8 noBuf, T_U16 dstW, T_pRECT dstWin, E_ImageFormat formatOut, T_U8 luminanceEn, T_U8* luminanceTab);	
功能概述	image scale and format convert function, support format:yuv420, rgb565 -- no block version	
参数说明	iBuf <in>	src buffer address arrays
	niBuf <in>	num of src buffer address array.
	srcW <in>	src image width
	srcWin <in>	src rect
	formatIn <in >	src image format: FORMAT_YUV420, FORMAT_RGB565
	oBuf <in>	dst buffer address arrays
	noBuf <in>	num of dst buffer address array
	dstW <in>	dst image width
	dstWin <in>	dst rect
	formatOut <in>	dst image format:FORMAT_YUV420, FORMAT_RGB565
	luminanceEn <in>	if input image format is FORMAT_YUV420, this param can set AK_TRUE to transform luminance.
	luminanceTab <in>	a continuous memory buffer width length of 256 byte, used to transform yuv420 luminance.
返回值说明	T_U8	0：成功；非 0：失败
注意事项	支持 yuv420 和 rgb565 格式的。若是 rgb888 格式，要先转换成 rgb565 格式才能用此接口。	
调用示例	-	

6.5.1.3 Fwl_ScaleConvertEx

原　　型	T_U8 Fwl_ScaleConvertEx(T_U32 *iBuf, T_U32 niBuf, T_U16 srcW, T_pRECT srcWin, E_ImageFormat formatIn, T_U32* oBuf, T_U8 noBuf, T_U16 dstW, T_pRECT dstWin, E_ImageFormat formatOut, T_BOOL alphaEn, T_U8 alpha, T_BOOL colorTransEn, T_U32 color);	
功能概述	image scale and format convert function, support format:yuv420, rgb565 -- block version This function includes alpha blending and color transparency function. If alpha blending and color transparency used, output image format must be FORMAT_RGB565!	
参数说明	iBuf <in>	src buffer address arrays
	niBuf <in>	num of src buffer address array.
	srcW <in>	src image width
	srcWin <in>	src rect
	formatIn <in >	src image format: FORMAT_YUV420, FORMAT_RGB565
	oBuf <in>	dst buffer address arrays
	noBuf <in>	num of dst buffer address array
	dstW <in>	dst image width
	dstWin <in>	dst rect
	formatOut <in>	dst image format:FORMAT_YUV420, FORMAT_RGB565
	alphaEn <in>	if dst image is FORMAT_RGB565, then alpha enabled is effective.
	alpha <in>	alpha value for alpha transparence (0 ~ 0xf).
	colorTransEn <in>	if dst image is FORMAT_RGB565, then color transparency is effective.
	color <in>	color value is 24bits, must value & 0xf8fcf8 because of input format FORMAT_RGB565.
返回值说明	T_U8	0：成功；非 0：失败

原　　型	T_U8 Fwl_ScaleConvertEx(T_U32 *iBuf, T_U32 niBuf, T_U16 srcW, T_pRECT srcWin, E_ImageFormat formatIn, T_U32* oBuf, T_U8 noBuf, T_U16 dstW, T_pRECT dstWin, E_ImageFormat formatOut, T_BOOL alphaEn, T_U8 alpha, T_BOOL colorTransEn, T_U32 color);
注意事项	支持 yuv420 和 rgb565 格式的。但若使用 alpha 和 color transparency，formatOut 必须是 FORMAT_RGB565 的。若是 rgb888 格式，要先转换成 rgb565 格式才能用此接口。
调用示例	

6.5.1.4 Fwl_ScaleConvertNoBlockEx

原　　型	T_U8 Fwl_ScaleConvertNoBlockEx(T_U32 *iBuf, T_U32 niBuf, T_U16 srcW, T_pRECT srcWin, E_ImageFormat formatIn, T_U32* oBuf, T_U8 noBuf, T_U16 dstW, T_pRECT dstWin, E_ImageFormat formatOut, T_BOOL alphaEn, T_U8 alpha, T_BOOL colorTransEn, T_U32 color);	
功能概述	image scale and format convert function, support format:yuv420, rgb565 -- no block version this function include alpha blending and color transparency function. if alpha blending and color transparency used, output image format must be FORMAT_RGB565!	
参数说明	iBuf <in>	src buffer address arrays
	niBuf <in>	num of src buffer address array.
	srcW <in>	src image width
	srcWin <in>	src rect
	formatIn <in >	src image format: FORMAT_YUV420, FORMAT_RGB565
	oBuf <in>	dst buffer address arrays
	noBuf <in>	num of dst buffer address array
	dstW <in>	dst image width
	dstWin <in>	dst rect
	formatOut <in>	dst image format:FORMAT_YUV420, FORMAT_RGB565

原　　型	T_U8 Fwl_ScaleConvertNoBlockEx(T_U32 *iBuf, T_U32 niBuf, T_U16 srcW, T_pRECT srcWin, E_ImageFormat formatIn, T_U32* oBuf, T_U8 noBuf, T_U16 dstW, T_pRECT dstWin, E_ImageFormat formatOut, T_BOOL alphaEn, T_U8 alpha, T_BOOL colorTransEn, T_U32 color);	
	alphaEn <in>	if dst image is FORMAT_RGB565, then alpha enabled is effective.
	alpha <in>	alpha value for alpha transparence (0 ~ 0xf).
	colorTransEn <in>	if dst image is FORMAT_RGB565, then color transparency is effective.
	color <in>	color value is 24bits, must value & 0xf8fcf8 because of input format FORMAT_RGB565.
返回值说明	T_U8	0：成功；非 0：失败
注意事项	支持 yuv420 和 rgb565 格式的。但若使用 alpha 和 color transparency，formatOut 必须是 FORMAT_RGB565 的。若是 rgb888 格式，要先转换成 rgb565 格式才能用此接口。	
调用示例	-	

6.5.2 RGB格式转换接口

6.5.2.1 Fwl_RGB565toRGB888

原　　型	T_VOID Fwl_RGB565toRGB888(T_S8 * pDestData888, T_S8 * pSrcData565, T_S32 nWidth, T_S32 nHeight)	
功能概述	convert RGB565 to RGB888	
参数说明	pDestData888 <in>	destation buffer, format is RGB888
	pSrcData565 <in>	source buffer, format is RGB565
	nWidth <in>	width
	nHeight <in>	height
返回值说明	T_VOID	
注意事项	pDestData888 和 pSrcData565 都是事先申请好的。	
调用示例	-	

6.5.2.2 Fwl_RGB888toRGB565

原　　型	T_VOID Fwl_RGB888toRGB565(T_S8 * pDestData565, T_S8 * pScrData888, T_S32 nWidth, T_S32 nHeight)	
功能概述	convert RGB888 to RGB565	
参数说明	pDestData565 <in>	destation buffer, format is RGB565
	pScrData888 <in>	source buffer, format is RGB888
	nWidth <in>	width
	nHeight <in>	height
返回值说明	T_VOID	-
注意事项	pDestData565 和 pSrcData888 都是事先申请好的。	
调用示例	-	

7 拍照录像接口说明

本章主要介绍有关拍照和录像功能的相关接口说明。

7.1 拍照

7.1.1 功能概述

拍照模块提供的对外接口主要负责相关 camera 的一些参数配置说明以及相关的调用示例。更多详细的接口说明用户可以参见《Sword37C 驱动库接口说明》的 Camera 章节。

7.1.2 接口说明

原　　型	T_BOOL cam_open(T_VOID);	
功能概述	open camera, should be done after the reset and initialization of camera //打开 camera，应该在 camera 复位初始化之后执行	
返回值说明	T_BOOL	AK_TRUE Success AK_FALSE Fail

原　　型	T_VOID cam_set_feature(T_CAMERA_FEATURE feature_type, T_U8 feature_setting);	
功能概述	set camera feature//设置 camera 效果参数	
参数说明	feature_type	[in] camera feature type,such as night mode,AWB,contrast etc.//夜间模式、AWB、对比度等等
	feature_setting	[in] feature setting//效果参数设置（包括效果、亮度、饱和度、对比度等设置）
返回值说明	T_VOID	
调用示例	-	

原　　型	T_BOOL cam_set_to_cap(T_U32 srcWidth, T_U32 srcHeight);	
功能概述	switch from preview mode to capture mode//预览模式切换到拍照模式	
参数说明	srcWidth	[in] window width//窗口宽度
	srcHeight	[in] window height//窗口高度
返回值说明	T_BOOL AK_TRUE if successed//操作成功 AK_FALSE if failed//操作失败	

原　　型	T_S32 cam_set_window(T_U32 srcWidth, T_U32 srcHeight);	
功能概述	set camera window//设置 camera 窗口	
参数说明	srcWidth	[in] window width//camera 窗口宽度
	srcHeight	[in] window height//camera 高度高度
返回值说明	T_S32 0 if error mode //错误模式 1 if success//操作成功 -1 if fail//操作失败	

原　　型	T_BOOL cam_capture_YUV(T_U8 *dstY, T_U8 *dstU, T_U8 *dstV, T_U32 dstWidth, T_U32 dstHeight, T_U32 timeout);	
功能概述	capture an image in YUV420 format//拍取 YUV420 格式图像	
参数说明	dstY	[out] Y buffer to save the image data//存储图像数据的 Ybuffer
	dstU	[out] U buffer to save the image data//存储图像数据的 Ubuffer
	dstV	[out] V buffer to save the image data//存储图像的 Vbuffer
	dstWidth	[in] desination width, the actual width of image in buffer//目标图像宽度
	dstHeight	[in] desination height, the actual height of image in buffer//目标图像高度
	timeout	[in] time out value for capture
返回值说明	T_BOOL AK_TRUE if success//操作成功 AK_FALSE if time out//操作失败	

原　　型	IMG_T_BOOL Img_YUV2JPEG_Stamp(const IMG_T_U8 *srcY, const IMG_T_U8 *srcU, const IMG_T_U8 *srcV, IMG_T_U8 *dstJPEG, IMG_T_U32 *size, IMG_T_U32 width, IMG_T_U32 height, IMG_T_U8 quality, J_STAMP_Info *stampinfo);	
功能概述	将 YUV 数据编码成 Jpeg 图片数据，（硬件编码）	
参数说明	srcY	[in] Y 数据指针
	srcU	[in] U 数据指针
	srcV	[in] V 数据指针
	dstJPEG	[out] Jpeg 数据指针
	size	[in,out] 输入：dstJPEG 缓冲大小；输出：Jpeg 长度
	width	[in] 目标图像宽度
	height	[in] 目标图像高度

原　　型	IMG_T_BOOL Img_YUV2JPEG_Stamp(const IMG_T_U8 *srcY, const IMG_T_U8 *srcU, const IMG_T_U8 *srcV, IMG_T_U8 *dstJPEG, IMG_T_U32 *size, IMG_T_U32 width, IMG_T_U32 height, IMG_T_U8 quality, J_STAMP_Info *stampinfo);	
	quality	[in] 编码质量，取值范围 0-200
	stampinfo	[in] 和源数据合成的时间戳等数据, 可以为 AK_NULL
注意事项	输入 YUV 格式 和 stamp 格式均为 YUV420	
返回值说明	AK_TRUE：成功；AK_FALSE：失败	

7.1.3 拍照用例

拍照用例如下所示：

```
typedef struct{
 T_U8       *mj; //jpg picture data
 T_U32      len; //file size per picture
 T_U8       *pY;
 T_U8       *pU;
 T_U8       *pV;
 T_U32   width;
 T_U32   height;
}T_CAPTURE;
T_CAPTURE *pCapture = AK_NULL;
#define JPG_BUF_SIZE    800*600*3
//camera 初始化
if (!cam_open())
{
   return AK_FALSE;
}
//效果参数初始化
cam_set_feature(CAM_FEATURE_EFFECT, effect);
cam_set_feature(CAM_FEATURE_BRIGHTNESS, brightness);
```

```
cam_set_feature(CAM_FEATURE_SATURATION, saturation);
cam_set_feature(CAM_FEATURE_NIGHT_MODE, mode);
cam_set_feature(CAM_FEATURE_CONTRAST, contrast);
//设置拍照尺寸
cam_set_to_cap(pCapture->width, pCapture->height);//拍照尺寸
//申请 YUV 数据空间
pCapture->pY = (T_U8*)Fwl_Malloc(szbuf + 64);
pCapture->pU = (T_U8*)Fwl_Malloc((szbuf >> 2) + 64);
pCapture->pV = (T_U8*)Fwl_Malloc((szbuf >> 2) + 64);
//图片数据空间
pCapture->mj = (T_U8 *)Fwl_Malloc(JPG_BUF_SIZE);
//获取 YUV 数据
cam_capture_YUV(pCapture->pY, pCapture->pU, pCapture->pV, pCapture->width, pCapture->height, timeout);
//保存 YUV 数据到 JPG 图片
pCapture->len = JPG_BUF_SIZE;
IImg_YUV2JPEG_Stamp(pCapture->pY, pCapture->pU, pCapture->pV, pCapture->mj, &pCapture->len,, pCapture->width, pCapture->height, quality,AK_NULL);
fp = Fwl_FileOpen(pFilePath, _FMODE_CREATE, _FMODE_CREATE);
Fwl_FileWrite(fp, pCapture->mj, pCapture->len);
Fwl_FileClose(fp);
//释放申请的空间
pCapture->mj = Fwl_Free(pCapture->mj);
pCapture->pY = Fwl_Free(pCapture->pY);
pCapture->pU = Fwl_Free(pCapture->pU);
pCapture->pV = Fwl_Free(pCapture->pV);
pCapture->mj = AK_NULL;
pCapture->pY = AK_NULL;
pCapture->pU = AK_NULL;
pCapture->pV = AK_NULL;
```

7.2 录像

7.2.1.1 功能概述

录像过程，需要多个线程协作。录像接口主要是各模块录像逻辑的控制。

7.2.1.2 录像各模块接口说明

7.2.1.2.1 视频缩放（Zoom）

视频 Zoom 管理的功能主要是通过缩放或透传摄像头原始的 yuv 数据，给视频编码和显示提供缩放处理后或真实的 yuv 数据。其对外接口如下所示：

原　　型	T_HANDLE VideoZoom_Open(T_U8 focusLvl, T_U32 width, T_U32 height);	
功能概述	创建一个缩放实例, 用于把 camera 的原始输出处理为指定大小的输出。	
参数说明	T_U8 [IN]focusLvl	默认变焦级别,0:不变焦;其它(不大于 10 的整数):初始化变焦级数。
	T_U32 [IN]width	camera 的原始输出大小（宽度/pixel）
	T_U32 [IN]height	camera 的原始输出大小（高度/pixel）
返回值说明	返回: 0:失败； 其它： 控制 Zoom 模块运行的句柄。	

原　　型	T_S32 VideoZoom_Start(T_HANDLE hdl);	
功能概述	启动 Zoom 模块，开始处理 camera 的原始输出	
参数说明	T_HANDLE [IN] hdl	Zoom 模块的控制句柄
返回值说明	AK_SUCCESS: Success Else: Fail	

原　　型	VideoZoom_Restart(T_HANDLE hdl, T_U32 width, T_U32 height)	
功能概述	重新启动 Zoom 模块	
参数说明	T_HANDLE [IN] hdl	Zoom 模块的控制句柄
	T_U32 [IN]width	camera 的原始输出大小（宽度/pixel）
	T_U32 [IN]height	camera 的原始输出大小（高度/pixel）
返回值说明	AK_SUCCESS: Success Else: Fail	

原　　型	T_S32 VideoZoom_Pause(T_HANDLE hdl)	
功能概述	暂停 Zoom 模块	
参数说明	T_HANDLE [IN] hdl	Zoom 模块的控制句柄
返回值说明	AK_SUCCESS 成功；其它：失败	

原　　型	T_S32 VideoZoom_Close(T_HANDLE hdl)	
功能概述	销毁 Zoom 模块的实例	
参数说明	T_HANDLE [IN] hdl	Zoom 模块的控制句柄
返回值说明	AK_SUCCESS: Success Else: Fail	

原　　型	T_S32 VideoZoom_OpenVideoSrc(T_HANDLE hdl, T_U32 width, T_U32 height)	
功能概述	打开 camera 的输出，用于给 Zoom 模块提供原始输入(camera 的原始输出)	
参数说明	T_HANDLE [IN] hdl	Zoom 模块的控制句柄
	T_U32 [IN]width	camera 的原始输出大小（宽度/pixel）
	T_U32 [IN]height	camera 的原始输出大小（高度/pixel）
返回值说明	AK_SUCCESS: Success Else: Fail	
注意事项	此接口是提供给当外部需要直接处理视频源，而不经过 Zoom 模块来处理。建议与 VideoZoom_CloseVideoSrc 配对使用。	

原　　型	T_S32 VideoZoom_RestartVideoSrc(T_HANDLE hdl, T_U32 width, T_U32 height)	
功能概述	重新设置 Zoom 模块的原始输入(camera 的原始输出)大小.	
参数说明	T_HANDLE [IN] hdl	Zoom 模块的控制句柄
	T_U32 [IN]width	camera 的原始输出大小（宽度/pixel）
	T_U32 [IN]height	camera 的原始输出大小（高度/pixel）
返回值说明	AK_SUCCESS: Success Else: Fail	
注意事项	此接口是提供给当外部需要直接处理视频源，而不经过 Zoom 模块来处理。	

原　　型	T_S32 VideoZoom_CloseVideoSrc(T_HANDLE hdl)	
功能概述	关闭 camera 的输出(主要是释放一些存放 camera 输出的 buffer)	
参数说明	T_HANDLE [IN] hdl	Zoom 模块的控制句柄
返回值说明	AK_SUCCESS 成功， 其它：失败	
注意事项	此接口是提供给当外部需要直接处理视频源，而不经过 Zoom 模块来处理。建议与 VideoZoom_ OpenVideoSrc 配对使用。	

原　　型	T_S32 VideoZoom_SetFocusLvl(T_HANDLE hdl, T_U8 focusLvl)	
功能概述	设置 Zoom 模块对 camera 输出的变焦级别	
参数说明	T_HANDLE [IN] hdl	Zoom 模块的控制句柄
	T_U8 [IN]focusLvl	变焦级别 (不大于 10 的整数,如果等于 0，则不变焦)
返回值说明	AK_SUCCESS 成功，其它：失败	

原　　型	T_S32 VideoZoom_DetectEnable(T_HANDLE hdl, T_BOOL isEnalbe)	
功能概述	打开/关闭 Zoom 模块的移动侦测功能	
参数说明	T_HANDLE [IN] hdl	Zoom 模块的控制句柄
	T_BOOL [IN] isEnalbe	是否打开移动侦测,AK_TRUE:打开,AK_FALSE 关闭
返回值说明	AK_SUCESS: Sucess Else: Fail	

原　　型	T_S32 VideoZoom_DetectSetInterval(T_HANDLE hdl, T_U32 intervalMs)	
功能概述	设置移动侦测的灵敏度(两次侦测之间的间隔时间)	
参数说明	T_HANDLE [IN] hdl	Zoom 模块的控制句柄
	T_U32 [IN]intervalMs	检测间隔时间(毫秒)
返回值说明	AK_SUCCESS 成功，其它：失败	

7.2.1.2.2 编码模块（MediaEncode）

视频编码模块主要用于对从 2D 模块获取到的 yuv Stream 进行视频编码，其对外接口如下所示。

原　　型	T_HANDLE MRec_Open(T_REC_AUDIO_INIT_PARAM *pAudioParm)	
功能概述	创建一个录制视频文件的实例，用于把 Zoom 模块的输出编码为视频文件	
参数说明	T_REC_AUDIO_INIT_PARAM [IN]	音频编码参数
返回值说明	返回: 0:失败, 其它: 控制编码模块运行的句柄。	
注意事项	1. 调用此接口之前需要先调用 MEnc_Init 初始化媒体库; 2. 创建编码任务将提前把音频 pcm 打开，目的是为了避免刚启动时候可能带来的噪音问题	

原　　型	T_REC_ERROR_STATUS MRec_Start(T_HANDLE handle,T_HANDLE srcCtlHdl, T_REC_CTRL_INIT_PARAM *pRecCtlParam, T_REC_AUDIO_INIT_PARAM *pRecAudioParam,T_REC_VIDEO_INIT_PARAM *pRecVideoParam)	
功能概述	启动编码模块	
参数说明	T_HANDLE [IN] handle l	编码模块的控制句柄
	T_HANDLE [IN]srcCtlHdl	Zoom 模块的控制句柄
	T_REC_CTRL_INIT_PARAM [IN]	编码控制参数
	T_REC_AUDIO_INIT_PARAM [IN]	音频编码参数
	T_REC_VIDEO_INIT_PARAM [IN]	视频编码参数
返回值说明	T_REC_ERROR_STATUS: REC_ERROR_OK 成功, 其它:失败 typedef enum { REC_ERROR_NULL, REC_ERROR_OK, REC_ERROR_ENCODE_EXP,//媒体库异常 REC_ERROR_NO_SPACE,//无可用存储空间(用于文件保存) REC_ERROR_NO_MEM,//内存不足(系统运行需要) REC_ERROR_DETECT_LIMIT,//移动侦测超时 REC_ERROR_SIZE_LIMIT,//文件大小超出用户配置 REC_ERROR_TIME_LIMIT,//编码时间超出用于配置 } T_REC_ERROR_STATUS;	

原　　型	T_S32 MRec_Stop(T_HANDLE handle, T_BOOL isSave)	
功能概述	停止编码	
参数说明	T_HANDLE [IN] hdl	编码模块的控制句柄
	T_BOOL [IN] isSave	是否保存编码文件，AK_TRUE:保存,AK_FALSE:删除
返回值说明	T_S32: AK_SUCCESS 成功, 其它:失败	

原　　型	T_S32 MRec_Close(T_HANDLE handle, T_BOOL isSave)	
功能概述	销毁编码实例	
参数说明	T_HANDLE [IN] hdl	编码模块的控制句柄
	T_BOOL [IN] isSave	是否保存编码文件，AK_TRUE:保存,AK_FALSE:删除
返回值说明	T_S32: AK_SUCCESS 成功, 其它:失败	

原　　型	T_REC_ERROR_STATUS MRec_Restart(T_HANDLE hdl, T_BOOL isSavePre)	
功能概述	重新启动编码模块	
参数说明	T_HANDLE [IN] hdl	编码模块的控制句柄
	T_BOOL [IN] isSave	在重启编码前，是否保存上一次的编码文件， AK_TRUE:保存,AK_FALSE:删除
返回值说明	T_REC_ERROR_STATUS: REC_ERROR_OK 成功, 其它:失败	

原　　型	T_S32 MRec_Ioctl(T_HANDLE handle, T_eMREC_IOCTL ctlType, T_VOID *arg)	
功能概述	获取编码模块的运行信息	
参数说明	T_HANDLE [IN] hdl	编码模块的控制句柄
	T_eMREC_IOCTL [IN] ctlType,	信息分类标识
	T_VOID [OUT]*arg	用于存储信息的 buffer
返回值说明	T_S32: AK_SUCCESS 成功, 其它:失败	

7.2.1.2.3　视频显示（VideoDisplay）

显示模块主要负责把正在录像的视频回馈到显示设备中,并在视频的基础上叠加上用户的ＵＩ,比如时间戳、进度条等。对外接口如下：

1、VideoDisp_Open

原　　型	T_HANDLE VideoDisp_Open(T_VOID)
功能概述	创建视频显示实例，用于把 Zoom 模块的视频刷新到显示设备上。
返回值说明	T_HANDLE：返回: NULL:失败，其它：控制视频显示模块运行的句柄

2、VideoDisp_SetPaintUiCbf

原　型	T_S32 VideoDisp_SetPaintUiCbf(T_HANDLE hdl,VIDEO_DISP_VIDEO_UI_CBF videoUiCbf,VIDEO_DISP_PAINT_UI_CHECK_CBF isPaintMenuCbf,VIDEO_DISP_MENU_UI_CBF menuPainCbf);	
功能概述	设置显示视频时的用户回调函数，用于在视频层上叠加 Icon 和菜单。	
参数说明	T_HANDLE [IN] hdl	视频显示模块的控制句柄
	VIDEO_DISP_VIDEO_UI_CBF	用于在视频上叠加 Icon 的回调函数
	VIDEO_DISP_PAINT_UI_CHECK_CBF	用于判断是否显示菜单控件的回调函数
	VIDEO_DISP_MENU_UI_CBF	用于在视频上叠加菜单控件的回调函数
返回值说明	T_S32: AK_SUCCESS 成功, 其它:失败	
注意事项	回调函数类型说明: 1 typedef T_S32 (*VIDEO_DISP_VIDEO_UI_CBF)(T_FRM_DATA *pDispFrame); 2. typedef T_S32 (*VIDEO_DISP_MENU_UI_CBF)(T_VOID); 3. typedef T_BOOL (*VIDEO_DISP_PAINT_UI_CHECK_CBF)(T_VOID);	

3、VideoDisp_Start

原　型	T_S32 VideoDisp_Start(T_HANDLE hdl,T_HANDLE srcCtlHdl, DISPLAY_TYPE_DEV tvOutMode)	
功能概述	启动视频显示模块	
参数说明	T_HANDLE [IN] hdl	视频显示模块的控制句柄
	T_HANDLE [IN]srcCtlHdl	Zoom 模块的控制句柄
	DISPLAY_TYPE_DEV[IN]	刷新方式,LCD/TV_PAL/TV_NTSC
返回值说明	T_S32: AK_SUCCESS 成功, 其它:失败	

4、VideoDisp_Pause

原　型	T_S32 VideoDisp_Pause(T_HANDLE hdl)	
功能概述	暂停视频显示模块	
参数说明	T_HANDLE [IN] hdl	视频显示模块的控制句柄
返回值说明	T_S32: AK_SUCCESS 成功, 其它:失败	

5、VideoDisp_Restart

原　型	T_S32 VideoDisp_Restart(T_HANDLE hdl)	
功能概述	重新启动异步视频显示模块	
参数说明	T_HANDLE [IN] hdl	视频显示模块的控制句柄
返回值说明	T_S32: AK_SUCCESS 成功, 其它:失败	

6、VideoDisp_SwitchTvOut

原　型	T_S32 VideoDisp_SwitchTvOut(T_HANDLE hdl, DISPLAY_TYPE_DEV tvOutMode, T_U8 brightness)	
功能概述	切换刷新模式	
参数说明	T_HANDLE [IN] hdl	视频显示模块的控制句柄
	DISPLAY_TYPE_DEV[IN]	刷新方式,LCD/TV_PAL/TV_NTSC
	T_U8 [IN]brightness	切换后显示亮度
返回值说明	T_S32: AK_SUCCESS 成功, 其它:失败	

7.2.1.2.4　异常管理（ExceptionManage）

录像异常管理主要用于检测录像的编码异常、录像时间超时、录像文件大小超出限制、剩余空间不足的检测，通过对不同异常的处理可实现循环录像和录像超时的应用。

1、MEnc_ExceptionStatusCheck

原　型	T_S32 MEnc_ExceptionStatusCheck(T_HANDLE hdl,T_REC_ERROR_STATUS *status)	
功能概述	编码异常检测	
参数说明	T_HANDLE [IN] hdl	编码模块的控制句柄
	T_REC_ERROR_STATUS [OUT]	编码异常类型

原　型	T_S32 MEnc_ExceptionStatusCheck(T_HANDLE hdl,T_REC_ERROR_STATUS *status)
返回值说明	T_S32: AK_SUCCESS 未检测到异常, 其它:检测到异常

2、MEnc_ExceptionIsNeedRestart

原　型	T_BOOL MEnc_ExceptionIsNeedRestart(T_HANDLE hdl, T_REC_ERROR_STATUS exitMode, T_BOOL * isNeedSavePre)	
功能概述	获取异常类型处理方式	
参数说明	T_HANDLE [IN] hdl	编码模块的控制句柄
	T_REC_ERROR_STATUS [IN]status	异常类型
	T_BOOL [OUT]* isNeedSavePre	是否需要保存编码文件
返回值说明	T_BOOL: AK_TRUE 需要重新开始编码,AK_FALSE :停止编码	
注意事项	异常类型说明，详见 MRec_Start 接口说明	

7.2.1.3　录像调用实例

```
// usage demo
T_S32  recorder_demo (T_VOID)
{
   T_HANDLE zoomHdl = 0; //视频 Zoom 模块控制句柄
   T_HANDLE displayHdl = 0;// 视频显示模块控制句柄
   T_HANDLE recorderHdl = 0;// 视频编码模块控制句柄
   T_U32 videoSrcWidth = 0;// camera 原始输出大小(宽度)
   T_U32 videoSrcHeight = 0;
   T_U8 videoFoculLvl = 0;//变焦级别
   T_S32 ret = AK_EFAILED;
   T_REC_AUDIO_INIT_PARAM  recAudioParam;//音频编码参数
   T_REC_VIDEO_INIT_PARAM recVideoParam;//视频编码参数
   T_REC_CTRL_INIT_PARAM   recCtlParam;//录像控制参数
   //参数设置：录像文件存储路径
         Utl_UStrCpy(recCtlParam.recFilePath, Fwl_GetDefPath(eVIDEOREC_PATH));
   //参数设置：录像临时索引文件存储路径
```

```
        Utl_UStrCpy(recCtlParam.indexFilePath, Fwl_GetDefPath(eRECIDX_PATH));
        //参数设置：循环录像时间间隔(毫米), 此处配置为 15 分钟（如果为 0，则为普通录像）
    recCtlParam.cycRecTimeMs = 15*60*1000;
        //参数设置：普通录像时间限制 (秒), 此处配置为 2 小时（如果为 0，则无时间限制）
        recCtlParam.recTimeSecLimit      = 2*60*60;
        //参数设置： 当个文件大小限制, 此处配置为 2 G 字节
        recCtlParam.recSizeLimit         = 2<<30;
        //参数设置：存储空间保留大小, 此处配置为 32M 字节，这些空间将不被使用
        recCtlParam.recResSize           = 32<<20;
    //参数设置：编码临时索引是否用 RamFile, 此处配置使用 DiskFile，不适用内存模拟
        recCtlParam.useMemIndex          = AK_FALSE;
    //参数设置：异步写文件缓冲大小, 此处配置为 2M 字节（如果为 0，则不适用异步写文件）
        recCtlParam.asynWriteSize        = (2<<20);
//参数设置：当配置为循环录像时，此配置有效。用于在循环录像时存储空间不够用时，如何
//控制删除文件。此处配置指删除文件时，只要删到够用，则停止删除。
        recCtlParam.autoDelAllFile       = AK_FALSE;
        //参数设置：是否打开动态变焦功能
        recCtlParam.isEnableFocus        = AK_TRUE;
        //参数设置：录像时候静止画面的侦测间隔时间（毫米）, 此处配置为 2 分钟
    recCtlParam.detectNoMovingTimeMs  = (1000 * 60 * 2);
    //参数设置： 音频编码的格式, 此处配置为 wave 格式
        recAudioParam.audioEncType = eRECORD_MODE_WAV;
    //参数设置： 音频编码的采样率大小，此处配置为 8K 采样率
        recAudioParam.audioEncSamp = 8000;
    //参数设置：视频编码的格式，此处配置为 AVI 格式
        recVideoParam.videoEncType                = MEDIALIB_REC_AVI_NORMAL;
    //参数设置： 视频编码的大小，此处配置为 VGA
        recVideoParam.videoWidth                  = 640;
        recVideoParam.videoHeight                 = 480;
    //参数设置： 视频编码使用的源大小，此处配置为 VGA
    recVideoParam.viedoSrcWinWidth  = 640;
```

```
    recVideoParam.viedoSrcWinHeight = 480;
    //参数设置： 视频编码的帧率，此处配置为 30 帧
            recVideoParam.FPS                                   = 30;
            recVideoParam.keyfFameInterval   = 1;
    //参数设置：视频编码的码流比特率，此处配置为 20M 比特每秒
            recVideoParam.vbps                                  = 20<<20;
    //创建视频放大的实例 (这个需要第一个调用)
    zoomHdl = VideoZoom_Open(videoFoculLvl, videoSrcWidth, videoSrcHeight);
    if (0 == zoomHdl)
    {
        goto Quit;
    }
    //创建视频显示的实例
    displayHdl = VideoDisp_Open();
    if (0 == displayHdl)
    {
        goto Quit;
    }
    // 创建视频编码的实例
    recorderHdl = MRec_Open(&recAudioParam);
    if (0 == recorderHdl)
    {
        goto Quit;
    }
    // 启动视频 Zoom 任务 (This func must call first)
    if (AK_IS_FAILURE(VideoZoom_Start(zoomHdl)))
    {
        goto Quit;
}
//设置变焦级别为 5
VideoZoom_SetFocusLvl(zoomHdl, 5);
// 打开移动侦测功能
```

```
VideoZoom_DetectEnable l(zoomHdl,  AK_TRUE);
//设置移动侦测灵敏度为 1 秒
VideoZoom_DetectSetInterval l(zoomHdl,  1*1000);
   //启动视频显示任务
   if (AK_IS_FAILURE(VideoDisp_Start(displayHdl,zoomHdl, DISPLAY_LCD_0)))
   {
      goto Quit;
   }
   //启动视频编码任务
   ret =  MRec_Start(recorderHdl,zoomHdl,&recCtlParam, &recAudioParam, &recVideoParam);
   if(REC_ERROR_OK != ret)
   {
      goto Quit;
   }
   while (1)
   {
    /*  如果需要暂停编码，请调用:
         MRec_Pause(recorderHdl);
      */
    /*  如果需要停止编码，请调用:
         MRec_Stop(recorderHdl, AK_TRUE);
      */
    /*  如果需要重新下一个编码，请调用:
         MRec_Restart(recorderHdl, AK_TRUE);
      */
    /*  如果 MMI 接受到这个消息，意味着编码任务自动结束:
          if (Get_MMI_Event == USER_EVT_STOP_REC)
          {
               break;
          }
       */
   }
 Quit:
```

```
        // 销毁显示实例
    if (0 != displayHdl)
    {
        VideoDisp_Close(displayHdl);
        displayHdl = 0;
    }
// 销毁编码实例
    if (0 != recorderHdl)
    {
        MRec_Close(recorderHdl, AK_TRUE);
        recorderHdl = 0;
    }
  // 销毁 Zoom 实例
    if (0 != zoomHdl)//(这个需要最后调用)
    {
        // close video zoom handle
        VideoZoom_Close(zoomHdl);
        zoomHdl = 0;
    }
    return ret;
}
```

8 网络

8.1 功能概述

网络接口提供了上次应用所关心的网络链接、收发等过程，与 socket 接口类似。封装了具体的 tcip/ip 协议实现，平台目前采用的是 lwip。

8.2 接口说明

8.2.1 主要封装接口

8.2.1.1 Fwl_Lwip_Init

原　　型	T_BOOL Fwl_Lwip_Init(T_VOID);	
功能概述	Lwip 模块初始化	
参数说明	T_VOID	-
返回值	T_BOOL	
返回值说明	AK_TRUE，成功；AK_FALSE，失败	
注意事项	Lwip 没有提供释放接口，因此，开机调用 lwip 初始化，之后不能释放。	

8.2.1.2 Fwl_Net_Init

原　　型	T_BOOL Fwl_Net_Init(T_U8 *pmac_addr, T_U32 ipaddr, T_U32 netmask, T_U32 gw);	
功能概述	网络模块初始化，包括 mac 初始化、网络环境初始化等，不包括 lwip 初始化。	
参数说明	pmac_addr [in]	Mac address
	ipaddr [in]	Ip address
	netmask [in]	Subnet mask
	gw [in]	Gate way
返回值	T_BOOL	
返回值说明	AK_TRUE，成功；AK_FALSE，失败	
注意事项	不包括 lwip 初始化	

8.2.1.3 Fwl_Net_Free

原　　型	T_BOOL Fwl_Net_Free(T_VOID);	
功能概述	网络模块释放	
参数说明	T_VOID	
返回值	T_BOOL	

原　　型	T_BOOL Fwl_Net_Free(T_VOID);
返回值说明	AK_TRUE，成功；AK_FALSE，失败
注意事项	不包括 lwip 释放

8.2.1.4 Fwl_Net_Conn_New

原　　型	T_NETCONN_STRUCT *Fwl_Net_Conn_New(NETWORKCONN_TYPE type);	
功能概述	新建网络连接句柄	
参数说明	Type [in]	Type
返回值	T_NETCONN_STRUCT *	
返回值说明	net connect struct handle	
注意事项	-	

8.2.1.5 Fwl_Net_Conn_Delete

原　　型	T_BOOL　Fwl_Net_Conn_Delete(T_NETCONN_STRUCT *pConnStruct);	
功能概述	网络连接句柄释放	
参数说明	pConnStruct [in]	net connect struct handle
返回值	T_BOOL	
返回值说明	AK_TRUE，成功；AK_FALSE，失败	
注意事项	-	

8.2.1.6 Fwl_Net_Conn_Bind

原　　型	T_BOOL　Fwl_Net_Conn_Bind(T_NETCONN_STRUCT *pConnStruct, T_U32 ipaddr, T_U16 port);	
功能概述	绑定本地 ip 地址和端口	
参数说明	pConnStruct [in]	net connect struct handle
	ipaddr [in]	0 means local ip
	port [in]	Local port, 0 means appointed by the protocol. if don't care the value of local port, such as tcp client, suggest to use 0 to solve the problem of port reuse .

原　　型	T_BOOL　Fwl_Net_Conn_Bind(T_NETCONN_STRUCT *pConnStruct, T_U32 ipaddr, T_U16 port);
返回值	T_BOOL
返回值说明	AK_TRUE，成功；AK_FALSE，失败
注意事项	-

8.2.1.7　Fwl_Net_Conn_Connect

原　　型	T_BOOL　Fwl_Net_Conn_Connect(T_NETCONN_STRUCT *pConnStruct, T_U32 ipaddr, T_U16 port);	
功能概述	连接	
参数说明	pConnStruct [in]	net connect struct handle
	ipaddr [in]	Remote Ip address
	port [in]	Remote port
返回值	T_BOOL	
返回值说明	AK_TRUE，成功；AK_FALSE，失败	
注意事项	仅 TCP 客户端调用	

8.2.1.8　Fwl_Net_Conn_Disconnect

原　　型	T_BOOL　Fwl_Net_Conn_Disconnect (T_NETCONN_STRUCT *pConnStruct);	
功能概述	UDP 断开连接	
参数说明	pConnStruct [in]	net connect struct handle
返回值	T_BOOL	
返回值说明	AK_TRUE，成功；AK_FALSE，失败	
注意事项	仅 UDP 用	

8.2.1.9　Fwl_Net_Conn_Listen

原　　型	T_BOOL　Fwl_Net_Conn_Listen(T_NETCONN_STRUCT *pConnStruct);	
功能概述	监听	
参数说明	pConnStruct [in]	net connect struct handle

原　　型	T_BOOL　Fwl_Net_Conn_Listen(T_NETCONN_STRUCT *pConnStruct);
返回值	T_BOOL
返回值说明	AK_TRUE，成功；AK_FALSE，失败
注意事项	仅 TCP 服务器端调用

8.2.1.10　Fwl_Net_Conn_Accept

原　　型	T_BOOL　Fwl_Net_Conn_Accept(T_NETCONN_STRUCT *pConnStruct, T_NETCONN_STRUCT ** ppNewConnStruct);	
功能概述	接受客户端连接	
参数说明	pConnStruct [in]	net connect struct handle
	ppNewConnStruct [out]	New net connect struct handle
	pInfo [out]	net connect info, can be null if don't need
返回值	T_BOOL	
返回值说明	AK_TRUE，成功；AK_FALSE，失败	
注意事项	仅 TCP 服务器端调用，阻塞的。	

8.2.1.11　Fwl_Net_Conn_Recv

原　　型	T_BOOL　Fwl_Net_Conn_Recv(T_NETCONN_STRUCT *pConnStruct, T_U8 *pdata, T_U32* size, T_U32 flag, T_U32 timeout);	
功能概述	接收数据	
参数说明	pConnStruct [in]	net connect struct handle
	pdata [out]	Receive data buffer
	size [in/out]	In:buffer size;out:write size
	flag [in]	flag
	timeout [in]	Timeout in ms, 0 means forever
返回值	T_BOOL	
返回值说明	AK_TRUE，成功；AK_FALSE，失败	
注意事项	阻塞的。	

8.2.1.12 Fwl_Net_Conn_Sendto

原　　型	T_BOOL　Fwl_Net_Conn_Sendto(T_NETCONN_STRUCT *pConnStruct, T_U8 *pdata, T_U32 size, T_U32 ipaddr, T_U16 port, T_U32 flag);	
功能概述	UDP 发送数据	
参数说明	pConnStruct [in]	net connect struct handle
	pdata [in]	send data buffer
	size [in]	size
	ipaddr [in]	Remote ip address
	port [in]	Remote port
	flag [in]	flag
返回值	T_BOOL	
返回值说明	AK_TRUE，成功；AK_FALSE，失败	
注意事项	仅 UDP 调用	

8.2.1.13 Fwl_Net_Conn_Send

原　　型	T_BOOL　Fwl_Net_Conn_Send(T_NETCONN_STRUCT *pConnStruct, T_U8 *pdata, T_U32 size, T_U32 flag);	
功能概述	TCP 发送数据	
参数说明	pConnStruct [in]	net connect struct handle
	pdata [in]	send data buffer
	size [in]	size
	flag [in]	flag
返回值	T_BOOL	
返回值说明	AK_TRUE，成功；AK_FALSE，失败	
注意事项	仅 TCP 调用	

8.2.1.14 Fwl_Net_Conn_Close

原　　型	T_BOOL　Fwl_Net_Conn_Close(T_NETCONN_STRUCT *pConnStruct);	
功能概述	TCP 网络连接关闭	
参数说明	pConnStruct [in]	net connect struct handle

原　　型	T_BOOL　Fwl_Net_Conn_Close(T_NETCONN_STRUCT *pConnStruct);
返回值	T_BOOL
返回值说明	AK_TRUE，成功；AK_FALSE，失败
注意事项	仅 TCP 调用

8.2.2 PING应用接口

对于收到外界发来的 ping 请求，lwip 会自动处理并发出 ping 回复，无需做任何处理；

对于主动向外发送 ping 请求，以及收到对方回复后，需要做处理。

8.2.2.1 ping_init

原　　型	T_VOID ping_init(ping_recv_fn func);	
功能概述	Ping 应用初始化	
参数说明	func [in]	注册接收到回复的回调函数
返回值	T_VOID	
返回值说明	-	
注意事项	-	

8.2.2.2 ping_free

原　　型	T_VOID ping_free(T_VOID);	
功能概述	Ping 应用释放	
参数说明	T_VOID	-
返回值	T_VOID	
返回值说明	-	
注意事项	-	

8.2.2.3 ping_send_now

原　　型	T_VOID ping_send_now(T_U32 ipaddr);	
功能概述	Ping 发送请求	
参数说明	ipaddr [in]	Remote ip address

原　　型	T_VOID ping_send_now(T_U32 ipaddr);
返回值	T_VOID
返回值说明	-
注意事项	-

8.2.3 其它相关接口

8.2.3.1 Fwl_Net_GetMacAddr

原　　型	T_BOOL Fwl_Net_GetMacAddr(T_U8 *pbuf, T_U32 *len);	
功能概述	从隐藏区读出烧录的 mac 地址	
参数说明	pbuf [out]	buffer to write mac addr
	len [in/out]	In:buffer length;out:write length
返回值	T_BOOL	
返回值说明	AK_TRUE，成功；AK_FALSE，失败	
注意事项	-	

8.2.3.2 Fwl_Net_Ip2str

原　　型	T_S32 Fwl_Net_Ip2str(T_U32 ipaddr, T_U16 *pbuf);	
功能概述	Ip 地址由数字转换成 unicode 字符串	
参数说明	ipaddr [in]	Ip address
	pbuf [out]	Unicode string buffer
返回值	T_S32	
返回值说明	len	
注意事项	-	

8.2.3.3 Fwl_Lwip_GetVersion

原　　型	T_pSTR Fwl_Lwip_GetVersion(T_VOID);	
功能概述	获取 lwip 库版本号	
参数说明	T_VOID	-
返回值	T_pSTR	
返回值说明	Version string	

原　　型	T_pSTR Fwl_Lwip_GetVersion(T_VOID);
注意事项	-

8.2.4 调用示例

8.2.4.1 模块初始化和释放

1、Lwip 初始化：在开机 VME_Main 函数中调用 **Fwl_Lwip_Init**();

2、网络模块初始化：可在进入网络模块状态机时调用。

```
T_BOOL network_enter(T_VOID)
{
	T_U32 ipaddr = 0;
	T_U32 netmask = 0;
	T_U32 gw = 0;
	T_U8 macaddr[6] = {0};
	T_U32 len = 6;
	IPADDR_CALC(&ipaddr, 172, 16, 5, 249);
	IPADDR_CALC(&netmask, 255, 255, 0, 0);
	IPADDR_CALC(&gw, 172, 16, 10, 10);
	//从隐藏区读出烧录的 mac 地址
	if (!Fwl_Net_GetMacAddr(macaddr, &len))
	{
		return AK_FALSE;
	}
	//网络模块初始化
	return Fwl_Net_Init(macaddr, ipaddr, netmask, gw);
}
```

3、网络模块释放：可在退出网络模块状态机时调用：**Fwl_Net_Free**();

8.2.4.2 TCP服务器端

TCP 服务器端，简单来说，流程为：new->bind->listen->accept->receive/send->close->delete，其中 listen 和 accept 是 TCP 服务器端专用的，send 和 close 是 TCP 专用。由于 accept 和 receive 是阻塞的，可以创建一个新线程出来运行 accept 和 receive 功能，阻塞时，不影响其他的操作响应。

示例是一个连接的，较简单。若想实现多连接，请参考平台应用文件 s_tcp_server.c 的代码。

1、 监听

```
T_BOOL Tcp_Server_Listen(T_VOID)
{
```

```
	T_BOOL ret = AK_FALSE;

	pTcpSvrParm->pNetconn = Fwl_Net_Conn_New(NETWORKCONN_TCP);

	if (AK_NULL == pNetconn)
		return ret;

	ret = Fwl_Net_Conn_Bind(pTcpSvrParm->pNetconn, 0, pTcpSvrParm->ListenPort);

	if (!ret)
		return ret;

	ret = Fwl_Net_Conn_Listen(pTcpSvrParm->pNetconn);

	if (!ret)
		return ret;

	//下面创建了一个线程，来接受客户端的连接，以及接收数据
	pTcpSvrParm->pStackAddr = Fwl_Malloc(TCP_SERVER_RECV_STACK_SIZE);
	AK_ASSERT_PTR(pTcpSvrParm->pStackAddr, "pTcpSvrParm->pStackAddr malloc error", ret);
	memset(pTcpSvrParm->pStackAddr, 0, TCP_SERVER_RECV_STACK_SIZE);

	pTcpSvrParm->task = AK_Create_Task((T_VOID*)Tcp_Server_Accept, "server accept", 1,
		AK_NULL, pTcpSvrParm->pStackAddr, TCP_SERVER_RECV_STACK_SIZE, 100, 5,
		AK_PREEMPT, AK_START);

	return ret;
}
```

2、 接受客户端连接以及接收数据

```
T_VOID Tcp_Server_Accept(T_U32 argc, T_VOID *argv)
{
	T_U8 data[TCP_SERVER_RECV_DATABUF_SIZE] = {0};
	T_U32 size = TCP_SERVER_RECV_DATABUF_SIZE;

	if (!Fwl_Net_Conn_Accept(pTcpSvrParm->pNetconn, &pTcpSvrParm->pNewNetconn, &pTcpSvrParm->ConnInfo))
		return;

	while (1)
```

```
        {
                if (!Fwl_Net_Conn_Recv(pTcpSvrParm->pNewNetconn, data, &size, 0,
0))
                        break;
                ……
        }
}
```

3、 **发送数据：Fwl_Net_Conn_Send**(pNetconn, data, size, NETCONN_COPY);

4、 **关闭服务器**

```
T_BOOL Tcp_Server_Close(T_VOID)
{
        T_BOOL ret = AK_FALSE;

        if (AK_INVALID_TASK != pTcpSvrParm->task)
        {
                AK_Terminate_Task(pTcpSvrParm->task);
                AK_Delete_Task(pTcpSvrParm->task);
                pTcpSvrParm->task = AK_INVALID_TASK;

                pTcpSvrParm->pStackAddr = Fwl_Free(pTcpSvrParm->pStackAddr);

                if (AK_NULL != pTcpSvrParm->pNewNetconn)
                {
                        Fwl_Net_Conn_Close(pTcpSvrParm->pNewNetconn);
                        Fwl_Net_Conn_Delete(pTcpSvrParm->pNewNetconn);
                        pTcpSvrParm->pNewNetconn = AK_NULL;
                }
        }

        Fwl_Net_Conn_Close(pTcpSvrParm->pNetconn);

        ret = Fwl_Net_Conn_Delete(pTcpSvrParm->pNetconn);
        pTcpSvrParm->pNetconn = AK_NULL;

        return ret;
}
```

8.2.4.3 TCP客户端

TCP 客户端，简单来说，流程为：new->bind->connect->receive/send->close->delete，其中 connect 是 TCP 客户端专用的，send 和 close 是 TCP 专用。由于 receive 是阻塞的，可以创建一个新线程出来运行 receive 功能，阻塞时，不影响其它的操作响应。

1、 连接

```
T_BOOL Tcp_Client_Connect(T_VOID)
{
	T_BOOL ret = AK_FALSE;

	pTcpCltParm->pNetconn = Fwl_Net_Conn_New(NETWORKCONN_TCP);

	if (AK_NULL == pTcpCltParm->pNetconn)
		return AK_FALSE;

	ret = Fwl_Net_Conn_Bind(pTcpCltParm->pNetconn, 0, pTcpCltParm->ConnInfo.LocalPort);

	if (!ret)
		return AK_FALSE;

	ret = Fwl_Net_Conn_Connect(pTcpCltParm->pNetconn, pTcpCltParm->ConnInfo.RemoteIp, pTcpCltParm->ConnInfo.RemotePort);

	if (!ret)
		return ret;

	//下面创建了一个线程，来接收数据
	pTcpCltParm->pStackAddr = Fwl_Malloc(TCP_CLIENT_RECV_STACK_SIZE);
	AK_ASSERT_PTR(pTcpCltParm->pStackAddr, "pTcpCltParm->pStackAddr malloc error", ret);
	memset(pTcpCltParm->pStackAddr, 0, TCP_CLIENT_RECV_STACK_SIZE);

	pTcpCltParm->task = AK_Create_Task((T_VOID*)Tcp_Client_Recv, "client recv", 1,
		AK_NULL, pTcpCltParm->pStackAddr, TCP_CLIENT_RECV_STACK_SIZE, 100, 5,
		AK_PREEMPT, AK_START);

	return ret;
}
```

2、 接收数据

```
T_VOID Tcp_Client_Recv(T_U32 argc, T_VOID *argv)
{
	T_U8 data[TCP_CLIENT_RECV_DATABUF_SIZE] = {0};
	T_U32 size = TCP_CLIENT_RECV_DATABUF_SIZE;

	while (1)
```

```
        {
                if (!Fwl_Net_Conn_Recv(pTcpCltParm->pNetconn, data, &size, 0, 0))
                        break;
                ……
        }
}
```

3、 发送数据：Fwl_Net_Conn_Send(pNetconn, data, size, NETCONN_COPY);

4、关闭客户端

```
T_BOOL Tcp_Client_Close(T_VOID)
{
        T_BOOL ret = AK_FALSE;

        if (AK_INVALID_TASK != pTcpCltParm->task)
        {
                AK_Terminate_Task(pTcpCltParm->task);
                AK_Delete_Task(pTcpCltParm->task);
                pTcpCltParm->task = AK_INVALID_TASK;

                pTcpCltParm->pStackAddr = Fwl_Free(pTcpCltParm->pStackAddr);
        }

        Fwl_Net_Conn_Close(pTcpCltParm->pNetconn);

        ret = Fwl_Net_Conn_Delete(pTcpCltParm->pNetconn);
        pTcpCltParm->pNetconn = AK_NULL;
        return ret;
}
```

8.2.4.4 UDP

UDP 协议无连接，也不用区分服务器端和客户端。简单来说，流程为：new->bind-> receive/sendto ->delete，其中 sendto 是 UDP 专用的。同样，receive 是阻塞的，可以创建一个新线程出来运行 receive 功能，阻塞时，不影响其它的操作响应。

1、 建立

```
T_BOOL Udp_Setup(T_VOID)
{
        T_BOOL ret = AK_FALSE;

        pUdpTransParm->pNetconn = Fwl_Net_Conn_New(NETWORKCONN_UDP);

        if (AK_NULL == pUdpTransParm->pNetconn)
```

```
            return ret;
        ret = Fwl_Net_Conn_Bind(pUdpTransParm->pNetconn, 0, pUdpTransParm->ConnInfo.LocalPort);

        if (!ret)
            return ret;
        //下面创建了一个线程，来接收数据
        pUdpTransParm->pStackAddr = Fwl_Malloc(UDP_RECV_STACK_SIZE);
        AK_ASSERT_PTR(pUdpTransParm->pStackAddr, "pUdpTransParm->pStackAddr malloc error", ret);
        memset(pUdpTransParm->pStackAddr, 0, UDP_RECV_STACK_SIZE);

        pUdpTransParm->task = AK_Create_Task((T_VOID*)Udp_Recv, "UDP recv", 1,
            AK_NULL, pUdpTransParm->pStackAddr, UDP_RECV_STACK_SIZE, 100, 5,
            AK_PREEMPT, AK_START);
        return ret;
}
```

2、 接收数据

Udp_Recv 实现同上一节的 Tcp_Client_Recv 类似。

3、 发送数据

Fwl_Net_Conn_Sendto(pNetconn, data, size, RemoteIp, RemotePort, NETCONN_COPY);

4、 关闭

```
T_BOOL Udp_Close(T_VOID)
{
    T_BOOL ret = AK_FALSE;

    if (AK_INVALID_TASK != pUdpTransParm->task)
    {
        AK_Terminate_Task(pUdpTransParm->task);
        AK_Delete_Task(pUdpTransParm->task);
        pUdpTransParm->task = AK_INVALID_TASK;

        pUdpTransParm->pStackAddr = Fwl_Free(pUdpTransParm->pStackAddr);
    }

    ret = Fwl_Net_Conn_Delete(pUdpTransParm->pNetconn);
    pUdpTransParm->pNetconn = AK_NULL;
```

```
        return ret;
}
```

8.2.4.5 Ping应用

1、 ping 初始化：ping_init(Ping_Test_Recv);

注册回调函数 Ping_Test_Recv。

```
T_VOID Ping_Test_Recv(T_U32 addr, T_U32 size, T_U32 time)
{
        //收到对方回复，可以显示对方的 ip 地址、回复数据包大小、耗时。
}
```

2、 发送 ping 请求

可根据需求，启动一个 timer，定时发送请求。

ping_send_now(pPingTestParm->remoteIp);

3、 ping 释放：ping_free();