福州瑞芯微电子有限公司

统一工具 DLL 接口

用户使用说明

2010-10-10

文档修改记录

版本号	制定日期	编制人/修改人	修改说明	生效日期	备 注
1.0.0	2010-01-22	CW	初版		
1.0.1	2010-10-10	LY	审核		
1.1.0	2010-11-11	LY	重新修正动态库接口		
1.3.5	2011-02-24	LY	增加 Rockusb 设备重启函数		
1.3.15	2011-05-26	LY	增加 IMEI 地址读写和获取 MSC 设备盘符		
1.3.15	2011-05-30	LY	增加格式化用户盘和拷贝数据到用户盘功能		
1.3.15	2011-05-31	LY	修改判断设备是否有插入函数增加设备 ID 输出		
1.3.17	2011-06-09	LY	增加获取 UID 功能		
1.3.20	2011-07-01	LY	封装升级参数		
1.3.21	2011-07-07	LY	升级参数增加 misc 修改标志		
1.3.22	2011-07-25	LY	增加 ReadUID 函数		
1.3.23	2011-07-30	LY	增加读写自定义数据函数		
1.3.24	2011-08-23	LY	单设备操作增加指定设备功能		
1.3.25	2011-11-03	LY	增加获取 parameter 和 sec3 功能		
1.3.26	2011-12-15	LY	增加从固件获取 parameter		
1.3.27	2012-04-01	LY	写自定义数据函数增加偏移参数		
1.6.1	2012-11-08	LY	增加读写 Vendor 信息函数		

目录

概述

本公司提供给客户进行集成的 DLL 接口程序，对固件升级进行了封装,并提供了丰富详实的例程。客户在集成时只要参考我们的例子，直接调用相应的函数就可以实现固件升级等功能。

1. 使用对象

本文档的主要读者是使用瑞芯公司芯片方案的客户公司的开发主管、开发人员。

2. 动态库简介

本套函数名称以RK_ 开头的函数，表示仅支持单台设备操作；以MD_开头的函数，代表支持多台设备操作。在使用最开始必须先调用RK_Initialize进行初始化。在所有操作完毕后，请务必调用RK_Uninitialize进行反初始化，否则将引起内存泄露。

3. 动态库使用

RKUpgrade.dll动态库使用VC6.0环境开发,所以开发环境是VC6.0则可以直接使用静态加载方式使用动态库,其他开发环境则需要使用动态加载方式。

4. 接口说明

4.1. 数据结构

4.1.1. 升级步骤状态

提示用户某一步骤是否成功

```
typedef enum
{
    DOWNLOADBOOT_START=1,          开始下载 Boot
    DOWNLOADBOOT_FAIL=2,           下载 Boot 失败
    DOWNLOADBOOT_PASS=3,           下载 Boot 成功
```

DOWNLOADIDBLOCK_START=4,	开始下载 ID Block 数据
DOWNLOADIDBLOCK_FAIL=5,	下载 ID Block 数据失败
DOWNLOADIDBLOCK_PASS=6,	下载 ID Block 数据成功
DOWNLOADIMAGE_START=7,	开始下载 Image 文件
DOWNLOADIMAGE_FAIL=8,	下载 Image 文件失败
DOWNLOADIMAGE_PASS=9,	下载 Image 文件成功
TESTDEVICE_START=10,	开始测试设备是否就绪
TESTDEVICE_FAIL=11,	测试设备失败
TESTDEVICE_PASS=12,	测试设备成功
RESETDEVICE_START=13,	开始重启设备
RESETDEVICE_FAIL=14,	重启设备失败
RESETDEVICE_PASS=15,	重启设备成功
FORMATDISK_START=16,	开始格式化磁盘
FORMATDISK_FAIL=17,	格式化磁盘失败
FORMATDISK_PASS=18,	格式化磁盘成功
COPYDATA_START=19,	开始拷贝数据
COPYDATA_FAIL=20,	拷贝数据失败
COPYDATA_PASS=21,	拷贝数据成功
WAITMSC_START=22,	开始等待 U 盘重新连上
WAITMSC_FAIL=23,	等待 U 盘重新连上失败
WAITMSC_PASS=24,	等待 U 盘重新连上成功
WAITLOADER_START=25,	开始等待 RockUsb Loader 设备重新连上
WAITLOADER_FAIL=26,	等待 RockUsb Loader 设备重新连上失败
WAITLOADER_PASS=27,	等待 RockUsb Loader 设备重新连上成功
WAITMASKROM_START=28,	开始等待 RockUsb Maskrom 设备重新连上
WAITMASKROM_FAIL=29,	等待 RockUsb Maskrom 设备重新连上失败
WAITMASKROM_PASS=30,	等待 RockUsb Maskrom 设备重新连上成功
ERASEIDB_START=31,	开始擦除 ID Block 数据
ERASEIDB_FAIL=32,	擦除 ID Block 数据失败
ERASEIDB_PASS=33,	擦除 ID Block 数据成功
SWITCHMSC_START=34,	开始切换 U 盘至 RockUsb 设备
SWITCHMSC_FAIL=35,	切换 U 盘至 RockUsb 设备失败
SWITCHMSC_PASS=36,	切换 U 盘至 RockUsb 设备成功
CHECKCHIP_START=37,	开始检测芯片是否支持
CHECKCHIP_FAIL=38,	检测芯片失败
CHECKCHIP_PASS=39,	检测芯片成功
PREPAREIDB_START=40,	开始构建 ID Block 数据
PREPAREIDB_FAIL=41,	构建 ID Block 数据失败
PREPAREIDB_PASS=42,	构建 ID Block 数据成功
MUTEXRESETDEVICE_START=43,	开始互斥重启设备
MUTEXRESETDEVICE_FAIL=44,	互斥重启设备失败
MUTEXRESETDEVICE_PASS=45,	互斥重启设备成功
GETOLDDISKSIZE_START=46,	开始从设备获取旧磁盘大小
GETOLDDISKSIZE_FAIL=47,	从设备获取旧磁盘大小失败

GETOLDDISKSIZE_PASS=48,	从设备获取旧磁盘大小成功
READSN_START=49,	开始读取 SN
READSN_FAIL=50,	读取 SN 失败
READSN_PASS=51,	读取 SN 成功
WRITESN_START=52,	开始写入 SN
WRITESN_FAIL=53,	写入 SN 失败
WRITESN_PASS=54,	写入 SN 成功
ERASEALLBLOCKS_START=55,	开始擦除整片 Flash
ERASEALLBLOCKS_FAIL=56,	擦除整片 Flash 失败
ERASEALLBLOCKS_PASS=57,	擦除整片 Flash 成功
GETBLOCKSTATE_START=58,	开始获取 Flash 块状态
GETBLOCKSTATE_FAIL=59,	获取 Flash 块状态失败
GETBLOCKSTATE_PASS=60,	获取 Flash 块状态成功
GETFLASHINFO_START=61,	开始获取 Flash 信息
GETFLASHINFO_FAIL=62,	获取 Flash 信息失败
GETFLASHINFO_PASS=63	获取 Flash 信息成功
WRITEBACK_START=64,	开始回写备份数据
WRITEBACK_FAIL=65,	回写备份数据失败
WRITEBACK_PASS=66,	回写备份数据成功
FINDUSERDISK_START=67,	开始搜寻用户磁盘
FINDUSERDISK_FAIL=68,	搜寻用户磁盘失败
FINDUSERDISK_PASS=69,	搜寻用户磁盘成功
SHOWUSERDISK_START=70,	开始启用用户磁盘(为了保证不拷贝进SD卡)
SHOWUSERDISK_FAIL=71,	启用用户磁盘失败
SHOWUSERDISK_PASS=72，	启用用户磁盘成功
READMAC_START=73,	开始读网卡地址
READMAC_FAIL=74,	读网卡地址失败
READMAC_PASS=75,	读网卡地址成功
WRITEMAC_START=76,	开始写网卡地址
WRITEMAC_FAIL=77,	写网卡地址失败
WRITEMAC_PASS=78,	写网卡地址成功
READBT_START=79,	开始读蓝牙地址
READBT_FAIL=80,	读蓝牙地址失败
READBT_PASS=81,	读蓝牙地址成功
WRITEBT_START=82,	开始写蓝牙地址
WRITEBT_FAIL=83,	写蓝牙地址失败
WRITEBT_PASS=84	写蓝牙地址成功
LOWERFORMAT_START=85,	设备低格开始
LOWERFORMAT_FAIL=86,	设备低格失败
LOWERFORMAT_PASS=87,	设备低格成功
READIMEI_START=88,	读 IMEI 开始
READIMEI_FAIL=89,	读 IMEI 失败
READIMEI_PASS=90,	读 IMEI 成功
WRITEIMEI_START=91,	写 IMEI 开始

```
    WRITEIMEI_FAIL=92,                  写 IMEI 失败
    WRITEIMEI_PASS=93,                  写 IMEI 成功
    SHOWDATADISK_START=94,              显示数据盘开始
    SHOWDATADISK_FAIL=95,               显示数据盘失败
    SHOWDATADISK_PASS=96,               显示数据盘成功
    FINDDATADISK_START=97,              查找数据盘开始
    FINDDATADISK_FAIL=98,               查找数据盘失败
    FINDDATADISK_PASS=99,               查找数据盘成功
    FORMATDATADISK_START=100,           格式化数据盘开始
    FORMATDATADISK_FAIL=101,            格式化数据盘失败
    FORMATDATADISK_PASS=102,            格式化数据盘成功
    COPYDATADISK_START=103,             数据盘拷贝开始
    COPYDATADISK_FAIL=104,              数据盘拷贝失败
    COPYDATADISK_PASS=105,              数据盘拷贝成功
    READUID_START=106,                  读 UID 开始
    READUID_FAIL=107,                   读 UID 失败
    READUID_PASS=108                    读 UID 成功
}ENUM_UPGRADE_PROMPT;
```

4.1.2. 进度提示

用于时间较长操作的进度提示，例如下载 Image，擦除 Flash 等

```
typedef enum
{
    TESTDEVICE_PROGRESS,                设备测试进度
    DOWNLOADIMAGE_PROGRESS,             Image 文件下载进度
    CHECKIMAGE_PROGRESS,                Image 有效性检测进度
    TAGBADBLOCK_PROGRESS,               Flash 坏块标记进度
    TESTBLOCK_PROGRESS,                 Flash 块检测进度
    ERASEFLASH_PROGRESS                 Flash 擦除进度
}ENUM_PROGRESS_PROMPT;
```

4.1.3. 调用次序

用于进度信息提示的回调函数中，通知用户当前回调状态。

```
typedef enum
{
    CALL_FIRST,                         第一次调用
    CALL_MIDDLE,                        中途调用
    CALL_LAST                           最后一次调用
}ENUM_CALL_STEP;
```

4.1.4. 回调函数

```
----------------------------------------------------------------------*/
typedef VOID (*UpgradeStepPromptCB)(DWORD deviceLayer,
                                    ENUM_UPGRADE_PROMPT promptID,
                                    DWORD oldDeviceLayer=0);
/*----------------------------------------------------------------------
Name      :    UpgradeStepPromptCB
Desc      :    升级步骤执行状态回调函数,是用户改变界面升级信息提示的接口
Params    :    (IN)deviceLayer:       设备层次 ID，用来区分连接在不同 USB 口的设备
               (IN)promptID:          升级步骤执行状态的枚举类型
               (IN)oldDeviceLayer     旧设备层次 ID
Return    :
Notes     :
----------------------------------------------------------------------*/
typedef VOID (*ProgressPromptCB)(DWORD deviceLayer,
                                 ENUM_PROGRESS_PROMPT promptID,
                                 DWORD totalValue,
                                 DWORD currentValue,
                                 ENUM_CALL_STEP emCall=CALL_MIDDLE);
/*----------------------------------------------------------------------
Name      :    ProgressPromptCB
Desc      :    时间较长操作进度状态回调函数，是用户显示此类操作进度的接口
Params    :    (IN)deviceLayer:       设备层次 ID，用来区分连接在不同 USB 口的设备
               (IN)promptID:          进度的操作类型
               (IN)totalValue         总数  例如 Flash 容量
               (IN)currentValue       当前值  例如已经擦除的 Flash 块数
               (IN)emCall             回调状态
Return    :
Notes     :
----------------------------------------------------------------------*/
```

4.1.5. 初始化信息

```
typedef struct _INIT_DEV_INFO                    初始化设备信息
{
    BOOL     bScan4FsUsb;                        是否扫描 USB 全速设备
    USHORT   usRockusbVid;                       Rockusb 设备 VID
    USHORT   usRockusbPid;                       Rockusb 设备 PID
    USHORT   usRockMscVid;                       U 盘设备 VID
    USHORT   usRockMscPid;                       U 盘设备PID
    UINT     uiRockusbTimeout;          等待 Rockusb 设备重新连接超时时长(单位秒)
```

```
    UINT      uiRockMscTimeout;          等待 U 盘设备重新连接超时时长(单位秒)
    UINT      emSupportDevice;           支持的芯片类型
}INIT_DEV_INFO,PINIT_DEV_INFO;
```

Note:emSupportDevice 的值是一个 ENUM_RKDEVICE_TYPE 类型

```
typedef enum
{
        RKNONE_DEVICE=0,
        RK27_DEVICE=0x10,
        RKCAYMAN_DEVICE,
        RK28_DEVICE=0x20,
        RK281X_DEVICE,
        RKNANO_DEVICE=0x30,
        RKCROWN_DEVICE=0x40,
        RK29_DEVICE=0x50
}ENUM_RKDEVICE_TYPE;

typedef struct _INIT_LOG_INFO_W                 初始化日志信息
{
    BOOL bLogEnable;                            是否开启日志
    LPWSTR lpszLogPathName;                     日志保存位置
}INIT_LOG_INFO_W,PINIT_LOG_INFO_W;
typedef struct _INIT_LOG_INFO_A                 初始化日志信息
{
    BOOL bLogEnable;                            是否开启日志
    LPSTR lpszLogPathName;                      日志保存位置
}INIT_LOG_INFO_A,PINIT_LOG_INFO_A;
#ifdef _UNICODE
    #define INIT_LOG_INFO INIT_LOG_INFO_W
    #define PINIT_LOG_INFO PINIT_LOG_INFO_W
#else
    #define INIT_LOG_INFO INIT_LOG_INFO_A
    #define PINIT_LOG_INFO PINIT_LOG_INFO_A
#endif

typedef struct _INIT_CALLBACK_INFO              初始化回调函数
{
    LPVOID pUpgradeStepPromptProc;              提示用户升级步骤执行结果的回调
    LPVOID pProgressPromptProc;                 提示用时较长操作进度的回调
}INIT_CALLBACK_INFO,PINIT_CALLBACK_INFO;
```

Note: pUpgradeStepPromptProc 指向的是 UpgradeStepPromptCB 类型的函数指针，
pProgressPromptProc 指向的是 ProgressPromptCB 类型的函数指针

4.1.6. 设备描述

```
typedef struct _STRUCT_DEVICE_DESC_W
{
    USHORTusVid;                                    //设备 VID
    USHORTusPid;                                    //设备 PID
    WCHAR szDrive;                                  //U 盘设备盘符
    USHORTusbcdUsb;                                 //设备 BCD，区分 Maskrom 和 Loader
    DWORD dwDeviceInstance;                         //设备实例，用来对 U 盘设备操作
    WCHAR szLinkName[MAX_PATH];                     //设备 ID
    WCHAR szLayer[MAX_PATH];                        //设备层次
    UINT    emUsbType;                              //USB 类型
    UINT    emDeviceType;                           //芯片类型
}STRUCT_DEVICE_DESC_W,*PSTRUCT_DEVICE_DESC_W;
typedef struct _STRUCT_DEVICE_DESC_A
{
    USHORTusVid;
    USHORTusPid;
    CHAR    szDrive;
    USHORTusbcdUsb;
    DWORD dwDeviceInstance;
    CHAR      szLinkName[MAX_PATH];
    CHAR    szLayer[MAX_PATH];
    UINT    emUsbType;
    UINT   emDeviceType;
}STRUCT_DEVICE_DESC_A,*PSTRUCT_DEVICE_DESC_A;
#ifdef _UNICODE
    #define STRUCT_DEVICE_DESC STRUCT_DEVICE_DESC_W
    #define PSTRUCT_DEVICE_DESC PSTRUCT_DEVICE_DESC_W
#else
    #define STRUCT_DEVICE_DESC STRUCT_DEVICE_DESC_A
    #define PSTRUCT_DEVICE_DESC PSTRUCT_DEVICE_DESC_A
#endif
Note: emUsbType 实际上是 ENUM_RKUSB_TYPE 枚举类型，
      emDeviceType 实际上是 ENUM_RKDEVICE_TYPE 枚举类型
typedef enum
{
        RKUSB_NONE=0x0,
        RKUSB_MASKROM=0x01,
        RKUSB_LOADER=0x02,
        RKUSB_MSC=0x04
}ENUM_RKUSB_TYPE;
```

4.1.7. 设备集信息

```
typedef struct    _STRUCT_DEVICESET_W
{
    BYTE nDevice;                                          //设备数
    STRUCT_DEVICE_DESC_W deviceSet[MAX_DEVICE];    //设备描述
    BOOL bRun[MAX_DEVICE];                                 //操作就绪表格
    BOOL bConnected[MAX_DEVICE];                           //设备连接记录表
    CHAR nNextPos;                                         //指示下一个设备位置
}STRUCT_DEVICESET_W,*PSTRUCT_DEVICESET_W;
typedef struct    _STRUCT_DEVICESET_A
{
    BYTE nDevice;
    STRUCT_DEVICE_DESC_A deviceSet[MAX_DEVICE];
    BOOL bRun[MAX_DEVICE];
    BOOL bConnected[MAX_DEVICE];
    CHAR nNextPos;
}STRUCT_DEVICESET_A,*PSTRUCT_DEVICESET_A;
#ifdef _UNICODE
    #define STRUCT_DEVICESET STRUCT_DEVICESET_W
    #define PSTRUCT_DEVICESET PSTRUCT_DEVICESET_W
#else
    #define STRUCT_DEVICESET STRUCT_DEVICESET_A
    #define PSTRUCT_DEVICESET PSTRUCT_DEVICESET_A
#endif
```

4.1.8. 升级参数信息

```
typedef struct _STRUCT_UPGRADE_PARAM_A
{
    BOOL bEnableFormat; //是否格式化用户盘
    BOOL bEnableCopyData;//是否拷贝目录或文件到用户盘
    BOOL bEnableFormatDataDisk;//是否格式化数据盘
    BOOL bEnableCopyDataDiskData;//是否拷贝目录或文件到数据盘
    LPSTR lpszDiskVolume;//用户盘格式化使用的卷标
    LPSTR lpszDataPath;//用户盘拷贝指定的目录或文件路径
    LPSTR lpszDataDiskVolume;//数据盘格式化使用的卷标
    LPSTR lpszDataDiskPath;//数据盘拷贝指定的目录或文件路径
    UINT    uiMiscModifyFlag;//0 表示不修改 misc 文件,1 表示 WIPE_ALL,2 表示
WIPE_DATA.
}STRUCT_UPGRADE_PARAM_A,*PSTRUCT_UPGRADE_PARAM_A;
```

```
typedef struct _STRUCT_UPGRADE_PARAM_W
{
    BOOL bEnableFormat;
    BOOL bEnableCopyData;
    BOOL bEnableFormatDataDisk;
    BOOL bEnableCopyDataDiskData;
    LPWSTR lpszDiskVolume;
    LPWSTR lpszDataPath;
    LPWSTR lpszDataDiskVolume;
    LPWSTR lpszDataDiskPath;
    UINT   uiMiscModifyFlag;
}STRUCT_UPGRADE_PARAM_W,*PSTRUCT_UPGRADE_PARAM_W;

#ifdef _UNICODE
#define STRUCT_UPGRADE_PARAM STRUCT_UPGRADE_PARAM_W
#define PSTRUCT_UPGRADE_PARAM PSTRUCT_UPGRADE_PARAM_W
#else
#define STRUCT_UPGRADE_PARAM STRUCT_UPGRADE_PARAM_A
#define PSTRUCT_UPGRADE_PARAM PSTRUCT_UPGRADE_PARAM_A
#endif
```

4.1.9. 批量结果

```
typedef struct _MD_RESULT
{
    UCHAR nErrResult[MAX_DEVICE];                    // 1:成功  0:失败，-1:无效不判断
}MD_RESULT,*PMD_RESULT;
```

4.2. 初始化与反初始化

```
/*----------------------------------------------------------------------
Name    :    RK_Initialize
Desc    :    初始化全局变量
Params  :    (IN)InitDevInfo:          设备信息的初始值
             (IN)InitLogInfo:          日志信息的初始值
             (IN)InitCallbackInfo      回调函数初始值
Return  :    TRUE:                     成功
             FALSE:                    失败
Notes   :
Author  :    Steven Chen
----------------------------------------------------------------------*/
BOOL   RK_InitializeW(INIT_DEV_INFO InitDevInfo,
```

```
                        INIT_LOG_INFO_W InitLogInfo,
                        INIT_CALLBACK_INFO InitCallbackInfo);
BOOL    RK_InitializeA(INIT_DEV_INFO InitDevInfo,
                        INIT_LOG_INFO_A InitLogInfo,
                        INIT_CALLBACK_INFO InitCallbackInfo);
#ifdef _UNICODE
#define    RK_Initialize RK_InitializeW
#else
#define    RK_Initialize RK_InitializeA
#endif

/*----------------------------------------------------------------------
Name    :    RK_Uninitialize
Desc    :    反初始化全局变量
Params  :    无
Return  :    TRUE:                  成功
             FALSE:                 失败
Notes   :
Author  :    Steven Chen
----------------------------------------------------------------------*/
BOOL    RK_Uninitialize();

/*----------------------------------------------------------------------
Name    :    RK_SetFirmware
Desc    :    设置并分析固件
Params  :    (IN)lpszFirmwarePathName:      固件文件路径
Return  :    TRUE:                  成功
             FALSE:                 失败
Notes   :    只有当该操作完成之后，才能实现后面的升级、修复、擦除 Flash 和获取固件版本和磁盘大小等各种操作
Author  :    Steven Chen
----------------------------------------------------------------------*/
BOOL    RK_SetFirmwareW(LPCWSTR lpszFirmwarePathName);
BOOL    RK_SetFirmwareA(LPCSTR lpszFirmwarePathName);
#ifdef _UNICODE
#define    RK_SetFirmware RK_SetFirmwareW
#else
#define    RK_SetFirmware RK_SetFirmwareA
#endif
```

4.3. 扫描设备

```
/*----------------------------------------------------------------------
```

Name : RK_ScanDevice
Desc : 扫描设备
Params : (OUT) nDeviceCounts: 返回扫描到的设备个数
(OUT) bExistMsc: 表明扫描到的设备中是否存在 msc 设备
Return : TRUE: 成功
FALSE: 失败
Notes : 只有在扫描设备后才能进行对设备的后续操作
Author : Steven Chen
----------*/
BOOL RK_ScanDevice(UINT &nDeviceCounts,BOOL &bExistMsc);

4.3.1. 获取设备信息

Name : RK_GetDeviceInfo
Desc : 获取设备的 Layer 值和 usb 类型
Params : (OUT) pLayerArray: 指向保存所有连接设备 layer 值的数组(由调用者分配)
(OUT) pUsbTypeArray: 指向保存所有连接设备 usb 类型的数组(由调用者分配)
(IN) nArrayLen 前面两个参数数组长度
Return : 实际拷贝到数组中的个数
Notes : 只有在扫描设备后才能进行此操作调用，数组空间根据扫描到的设备数进行分配
Author : Seth
----------*/
int RK_GetDeviceInfo(PDWORD pLayerArray,PDWORD pUsbTypeArray,int nArrayLen);

4.4. 单设备操作

4.4.1. Flash 操作

/*----------
Name : RK_EraseFlash
Desc : 擦除 Flash
Params : (IN) dwLayer:值为 0 表示对扫描到第一个连接设备进行操作,其他值则对特定设备进行操作
Return : TRUE: 成功
FALSE: 失败
Notes : 通过进度回调，可以获得当前的擦除进度
Author : Steven Chen
----------*/
BOOL RK_EraseFlash(DWORD dwLayer=0);

4.4.2. 升级与修复

```
/*--------------------------------------------------------------------
Name      :    RK_Upgrade
Desc      :    设备升级
Params    :    (IN)upgradeParam 具体内容请查看升级参数信息说明
               (IN)   dwLayer:值为 0 表示对扫描到第一个连接设备进行操作,其他值则对特定设备进行操作
Return    :    TRUE:                成功
               FALSE:               失败
Notes     :
Author    :    Steven Chen
--------------------------------------------------------------------*/
BOOL   RK_UpgradeW(STRUCT_UPGRADE_PARAM_W &upgradeParam
                   ,DWORD dwLayer=0);
BOOL    RK_UpgradeA (STRUCT_UPGRADE_PARAM_A &upgradeParam
                   ,DWORD dwLayer=0);
#ifdef _UNICODE
#define   RK_Upgrade RK_UpgradeW
#else
#define   RK_Upgrade RK_UpgradeA
#endif

/*--------------------------------------------------------------------
Name      :    RK_Restore
Desc      :    修复设备
Params    :    (IN)upgradeParam 具体内容请查看升级参数信息说明
               (IN)   dwLayer:值为 0 表示对扫描到第一个连接设备进行操作,其他值则对特定设备进行操作
Return    :    TRUE:                成功
               FALSE:               失败
Notes     :    修复过程会先擦除 Flash 再进行升级
Author    :    Steven Chen
--------------------------------------------------------------------*/
BOOL    RK_RestoreW(STRUCT_UPGRADE_PARAM_W &upgradeParam
                   ,DWORD dwLayer=0);
BOOL   RK_RestoreA(STRUCT_UPGRADE_PARAM_A &upgradeParam
                   ,DWORD dwLayer=0);
#ifdef _UNICODE
#define   RK_Restore RK_RestoreW
#else
#define   RK_Restore RK_RestoreA
#endif
```

4.4.3. 序列号操作

```
/*---------------------------------------------------------------------
Name    :   RK_ReadSN
Desc    :   从设备读取序列号（暂不支持 RKnano）
Params  :   (IN)pSN:            指向序列号的缓冲(调用者分配)
            (OUT)nSNLen:        序列号长度（字节为单位）
            (IN)  dwLayer:值为 0 表示对扫描到第一个连接设备进行操作,其他值则对特
定设备进行操作
Return  :   TRUE:               成功
            FALSE:              失败
Notes   :   pSN 由调用者分配，nSNLen 在调用前为 pSN 指向的空间大小(字节为单位)
            调用后为序列号长度
Author  :   Steven Chen
---------------------------------------------------------------------*/
BOOL  RK_ReadSN(PBYTE pSN, INT& nSNLen,DWORD dwLayer=0);
/*---------------------------------------------------------------------
Name    :   RK_WriteSN
Desc    :   向设备写入序列号（暂不支持 RKnano）
Params  :   (IN)pSN:            指向序列号的缓冲
            (IN)nSNLen:         序列号长度（字节为单位）
            (IN)  dwLayer:值为 0 表示对扫描到第一个连接设备进行操作,其他值则对特
定设备进行操作
Return  :   TRUE:               成功
            FALSE:              失败
Notes   :
Author  :   Steven Chen
---------------------------------------------------------------------*/
BOOL   RK_WriteSN(PBYTE pSN, INT nSNLen,DWORD dwLayer=0);
```

4.4.4. 网卡地址操作

```
/*---------------------------------------------------------------------
Name    :   RK_ReadMAC
Desc    :   从设备读取网卡地址（暂不支持 RKnano）
Params  :   (IN)pMac:           指向网卡地址的缓冲(调用者分配)
            (OUT)nMacLen:       网卡地址长度（字节为单位）
            (IN)  dwLayer:值为 0 表示对扫描到第一个连接设备进行操作,其他值则对特
定设备进行操作
Return  :   TRUE:               成功
            FALSE:              失败
Notes   :   pMac 由调用用者分配，nMacLen 在调用前为 pMac 指向的空间大小(字节为单
```

位)调用后为网卡地址长度
Author : Seth
----------*/
BOOL RK_ReadMAC(PBYTE pMac, INT& nMacLen,DWORD dwLayer=0);
/*----------
Name : RK_ WriteMAC
Desc : 向设备写入网卡地址（暂不支持 RKnano）
Params : (IN)pMac: 指向网卡地址的缓冲
(IN)nMacLen: 网卡地址长度（字节为单位）
(IN) dwLayer:值为 0 表示对扫描到第一个连接设备进行操作,其他值则对特定设备进行操作
Return : TRUE: 成功
FALSE: 失败
Notes :
Author : Seth
----------*/
BOOL RK_WriteMAC(PBYTE pMac, INT nMacLen=6,DWORD dwLayer=0);

4.4.5. 蓝牙地址操作

/*----------
Name : RK_ReadBT
Desc : 从设备读取蓝牙地址（暂不支持 RKnano）
Params : (IN)pBT: 指向蓝牙地址的缓冲(调用者分配)
(OUT)nBTLen: 蓝牙地址长度（字节为单位）
(IN) dwLayer:值为 0 表示对扫描到第一个连接设备进行操作,其他值则对特定设备进行操作
Return : TRUE: 成功
FALSE: 失败
Notes : pBT 由调用用者分配，nBTLen 在调用前为 pBT 指向的空间大小(字节为单位)
调用后为蓝牙地址长度
Author : Seth
----------*/
BOOL RK_ReadBT(PBYTE pBT, INT& nBTLen,DWORD dwLayer=0);
/*----------
Name : RK_ WriteBT
Desc : 向设备写入蓝牙地址（暂不支持 RKnano）
Params : (IN)pBT: 指向蓝牙地址的缓冲
(IN)nBTLen: 蓝牙地址长度（字节为单位）
(IN) dwLayer:值为 0 表示对扫描到第一个连接设备进行操作,其他值则对特定设备进行操作
Return : TRUE: 成功
FALSE: 失败

Notes :
Author : Seth
----------*/
BOOL _stdcall RK_WriteBT(PBYTE pBT, INT nBTLen=6,DWORD dwLayer=0);

4.4.6. IMEI 地址操作

/*----------
Name : RK_ReadIMEI
Desc : 从设备读取 IMEI 地址（暂不支持 RKnano）
Params : (IN)pImei: 指向 IMEI 地址的缓冲(调用者分配)
(OUT)nImeiLen: IMEI 地址长度（字节为单位）
(IN) dwLayer:值为 0 表示对扫描到第一个连接设备进行操作,其他值则对特定设备进行操作
Return : TRUE: 成功
FALSE: 失败
Notes : pImei 由调用用者分配，nImeiLen 在调用前为 pImei 指向的空间大小(字节为单位)调用后为 Imei 地址长度
Author : Seth
----------*/
BOOL RK_ReadIMEI(PBYTE pImei, INT& nImeiLen,DWORD dwLayer=0);
/*----------
Name : RK_ WriteIMEI
Desc : 向设备写入 IMEI 地址（暂不支持 RKnano）
Params : (IN)pImei: 指向 IMEI 地址的缓冲
(IN)nImeiLen: IMEI 地址长度（字节为单位）
(IN) dwLayer:值为 0 表示对扫描到第一个连接设备进行操作,其他值则对特定设备进行操作
Return : TRUE: 成功
FALSE: 失败
Notes :
Author : Seth
----------*/
BOOL _stdcall RK_WriteIMEI(PBYTE pIMEI, INT nIMEILen=15,DWORD dwLayer=0);

4.4.7. UID 操作

/*----------
Name : RK_ReadUID
Desc : 从设备读取 UID
Params : (IN)pUID: 指向 UID 的缓冲(调用者分配)

(OUT)nUIDLen: UID 长度（字节为单位）

(IN) dwLayer:值为 0 表示对扫描到第一个连接设备进行操作,其他值则对特定设备进行操作

Return : TRUE: 成功

FALSE: 失败

Notes : pUID 由调用用者分配，nUIDLen 在调用前为 pUID 指向的空间大小(字节为单位)调用后为 UID 长度

Author : Seth

----------*/

```
BOOL RK_ReadUID(PBYTE pUID, INT& nUIDLen,DWORD dwLayer=0);
```

4.4.8. Vendor 信息操作

```
/*----------
```

Name : RK_ReadVendorInfo

Desc : 从设备读取 Vendor 信息

Params : (IN)sectorOffset: 要读取信息相对于 vendor 信息开始位置，偏移多少个扇区，以 0 基准

(IN)sectorCount: 要读取的扇区个数

(OUT)pVendorBuffer: 用于保存读取到的 Vendor 信息，由调用者分配此空间，大小至少为 504*sectorCount 字节

(IN) dwLayer:值为 0 表示对扫描到第一个连接设备进行操作,其他值则对特定设备进行操作

Return : TRUE: 成功

FALSE: 失败

Notes : 1 个 Vendor 扇区等于 504 字节

Author : Seth

----------*/

```
BOOL RK_ReadVendorInfo(int sectorOffset,int sectorCount,PBYTE pVendorBuffer, DWORD dwLayer=0)
/*----------
```

Name : RK_WriteVendorInfo

Desc : 写 Vendor 信息到设备

Params : (IN)sectorOffset: 要写入信息相对于 vendor 信息开始位置，偏移多少个扇区，以 0 基准

(IN)sectorCount: 要写入的扇区个数

(OUT)pVendorBuffer: 保存要写入的 Vendor 信息，由调用者分配此空间，大小至少为 504*sectorCount 字节

(IN) dwLayer:值为 0 表示对扫描到第一个连接设备进行操作,其他值则对特定设备进行操作

Return : TRUE: 成功

FALSE: 失败

Notes : 1 个 Vendor 扇区等于 504 字节
Author : Seth
----------*/

BOOL RK_WriteVendorInfo(int sectorOffset,int sectorCount,PBYTE pVendorBuffer, DWORD dwLayer=0)

4.4.9. 获取 Parameter

/*----------

Name : RK_ReadParameterFile
Desc : Loader 或 msc 状态下,从设备读取 Parameter 文件
Params : (IN) pParameter: 指向 parameter 的缓冲(调用者分配)
(OUT) nParamSize: parameter 大小（字节为单位）
(IN) dwLayer:值为 0 表示对扫描到第一个连接设备进行操作,其他值则对特定设备进行操作
Return : TRUE: 成功
FALSE: 失败
Notes : 当 pParameter=NULL 且 nParamSize=-1 时函数用于获取 parameter 大小，值保存在 nParamSize 中。然后正确分配空间，获取 parameter
Author : Seth
----------*/

BOOL _stdcall RK_ReadParameterFile(PBYTE pParameter, INT& nParamSize,
DWORD dwLayer=0);

4.4.10. 获取固件 Parameter

/*----------

Name : RK_ReadFirmwareParameterFile
Desc : ,从固件读取 Parameter 文件
Params : (IN) pParameter: 指向 parameter 的缓冲(调用者分配)
(OUT) nParamSize: parameter 大小（字节为单位）

Return : TRUE: 成功
FALSE: 失败
Notes : 当 pParameter=NULL 且 nParamSize=-1 时函数用于获取 parameter 大小，值保存在 nParamSize 中。然后正确分配空间，获取 parameter
Author : Seth
----------*/

BOOL _stdcall RK_ReadFirmwareParameterFile(PBYTE pParameter, INT& nParamSize);

4.4.11. 自定义数据操作

```
/*--------------------------------------------------------------------
Name    :  RK_ReadCustomData
Desc    :  从设备读取自定义数据（暂不支持 RKnano）
Params  :  (IN)pCustomData:        指向自定义数据的缓冲(调用者分配)
           (OUT)nCusstomDataLen:自定义数据长度（字节为单位）
           (IN)  dwLayer:值为 0 表示对扫描到第一个连接设备进行操作,其他值则对特定设备进行操作
Return  :  TRUE:                   成功
           FALSE:                  失败
Notes   :  pCustomData 由调用用者分配，nCusstomDataLen 在调用前为 pCustomData 指向的空间大小(字节为单位)调用后为实际读取的长度
Author  :  Seth
--------------------------------------------------------------------*/
BOOL  RK_ReadCustomData(PBYTE pCustomData, INT& nCusstomDataLen
,DWORD dwLayer=0);
/*--------------------------------------------------------------------
Name    :  RK_ WriteCustomData
Desc    :  向设备写入自定义数据（暂不支持 RKnano）
Params  :  (IN) pCustomData:           指向自定义数据的缓冲
           (IN) nCusstomDataOffset:    自定义数据写入的偏移位置（字节为单位）
           (IN) nCusstomDataLen:       自定义数据缓冲长度（字节为单位）
           (IN)  dwLayer:值为 0 表示对扫描到第一个连接设备进行操作,其他值则对特定设备进行操作
Return  :  TRUE:                  成功
           FALSE:                 失败
Notes   :  自定义数据最大支持到 512 个字节
Author  :  Seth
--------------------------------------------------------------------*/
BOOL _stdcall RK_WriteCustomData(PBYTE pCustomData, INT nCustomDataOffset,
INT nCusstomDataLen,DWORD dwLayer=0);
```

4.4.12. 获取磁盘大小

```
/*--------------------------------------------------------------------
Name    :  RK_GetFirmwareDiskSize
Desc    :  获取新固件磁盘容量（暂不支持 RKnano）
Params  :  (OUT) uiSysDiskSize:    系统盘容量(MB)
           (OUT) uiDataDiskSize:   数据盘容量(MB)
Return  :  TRUE:                   成功
           FALSE:                  失败
```

Notes :
Author : Steven Chen
----------*/
BOOL RK_GetFirmwareDiskSize(UINT& uiSysDiskSize,
UINT& uiDataDiskSize);
/*----------
Name : RK_GetDeviceDiskSize
Desc : 获取旧固件磁盘容量（暂不支持 RKnano）
Params : (OUT) uiSysDiskSize: 系统盘容量(MB)
(OUT) uiDataDiskSize: 数据盘容量(MB)
(IN) dwLayer:值为 0 表示对扫描到第一个连接设备进行操作,其他值则对特定设备进行操作
Return : TRUE: 成功
FALSE: 失败
Notes :
Author : Steven Chen
----------*/
BOOL RK_GetDeviceDiskSize(UINT& uiSysDiskSize,
UINT& uiDataDiskSize,
DWORD dwLayer=0);

4.4.13. 获取版本信息

固件版本	主版本	辅版本	微版本
dwFwVer	Bit(12-15)	Bit(8-11)	Bit(0-7)
0x0102000	1	2	0

Boot 版本	预留	主版本	辅版本
dwBootVer	Bit(8-15)	Bit(4-7)	Bit(0-3)
0x00000103		1	3

/*----------
Name : RK_GetFirmwareVersion
Desc : 获取新固件版本信息
Params : (OUT) dwFwVer: 固件版本
(OUT) dwBootVer Boot 版本
Return : TRUE: 成功
FALSE: 失败
Notes :
Author : Seth
----------*/
BOOL RK_GetFirmwareVersion(DWORD &dwFwVer,DWORD &dwBootVer);

```
/*--------------------------------------------------------------------
Name      :    RK_GetDeviceVersion
Desc      :    获取设备上固件版本信息
Params    :    (OUT) dwFwVer:     固件版本
               (OUT) dwBootVer    Boot 版本
               (IN)   dwLayer:值为 0 表示对扫描到第一个连接设备进行操作,其他值则对特定设备进行操作
Return    :    TRUE:              成功
               FALSE:             失败
Notes     :
Author    :    Seth
--------------------------------------------------------------------*/
BOOL   RK_GetDeviceVersion(DWORD &dwFwVer,DWORD &dwBootVer
 ,DWORD dwLayer=0);
```

4.4.14. 获取设备 UID

```
/*--------------------------------------------------------------------
Name      :    RK_GetDeviceUID
Desc      :    获取设备上的 UID 信息
Params    :    (OUT) pUid:     用于存放 UID 信息,由调用者分配空间,至少 30 个字节
               (IN)   dwLayer:值为 0 表示对扫描到第一个连接设备进行操作,其他值则对特定设备进行操作
Return    :    TRUE:              成功
               FALSE:             失败
Notes     :
Author    :    Seth
--------------------------------------------------------------------*/
BOOL   RK_GetDeviceUID(PBYTE pUid,DWORD dwLayer=0);
```

4.4.15. 获取设备产品类型

```
/*--------------------------------------------------------------------
Name      :    RK_GetDeviceProductModel
Desc      :    获取设备上的产品类型信息
Params    :    (OUT) pProductModel:     用于存产品信息,由调用者分配空间,32 个字节
               (IN)   dwLayer:值为 0 表示对扫描到第一个连接设备进行操作,其他值则对特定设备进行操作
Return    :    TRUE:              成功
```

FALSE: 失败
Notes :
Author : Seth
----------*/
BOOL RK_GetDeviceProductModel(PBYTE pProductModel,DWORD dwLayer=0);

4.4.16. 获取设备产品序列号

/*----------
Name : RK_GetDeviceSN
Desc : 获取设备上的产品序列号
Params : (OUT) pProductSN: 用于存产品序列号,由调用者分配空间,32 个字节
(IN) dwLayer:值为 0 表示对扫描到第一个连接设备进行操作,其他值则对特定设备进行操作
Return : TRUE: 成功
FALSE: 失败
Notes :
Author : Seth
----------*/
BOOL RK_GetDeviceSN(PBYTE pProductSN,DWORD dwLayer=0);

4.4.17. 获取 IDBLOCK 扇区 3 数据

/*----------
Name : RK_GetDeviceIDBSector
Desc : msc 状态下读取 IDBlock 中扇区 3 数据
Params : (OUT) pSec: 用于扇区数据,由调用者分配空间,512 个字节
(IN) dwLayer:值为 0 表示对扫描到第一个连接设备进行操作,其他值则对特定设备进行操作
Return : TRUE: 成功
FALSE: 失败
Notes :
Author : Seth
----------*/
BOOL RK_GetDeviceIDBSector(PBYTE pSec,DWORD dwLayer=0);

4.4.18. USB 设备切换

/*----------
Name : RK_SwitchToRockusb
Desc : 从 U 盘设备切换到 Rockusb 设备

Params : (IN) dwLayer:值为 0 表示对扫描到第一个连接设备进行操作,其他值则对特定设备进行操作
Return : TRUE: 成功
FALSE: 失败
Notes : 切换后会等待设备重新连接
Author : Seth
----------*/
BOOL RK_SwitchToRockusb(DWORD dwLayer=0);

4.4.19. 判断设备是否插入

/*----------
Name : RK_IsRockUsbPlugged
Desc : 在没有安装驱动情况下,检测设备是否连接 PC
Params : (OUT) lpszDeviceID 输出设备硬件 ID
Return : TRUE: Found
FALSE: Not found
Notes :
Author : Seth
----------*/
BOOL RK_IsRockUsbPluggedW(LPWSTR lpszDeviceID=NULL);
BOOL RK_IsRockUsbPluggedA(LPSTR lpszDeviceID=NULL);
#ifdef _UNICODE
#define RK_IsRockUsbPlugged RK_IsRockUsbPluggedW
#else
#define RK_IsRockUsbPlugged RK_IsRockUsbPluggedA
#endif

4.4.20. 安装 USB 驱动

/*----------
Name : RK_InstallDriver
Desc : 安装 Rockusb 设备驱动
Params : (IN)lpszDriverInfPathName: 驱动 inf 文件路径
Return : TRUE: 成功
FALSE: 失败
Notes :
Author : Steven Chen
----------*/
BOOL RK_InstallDriverW(LPCWSTR lpszDriverInfPathName);
BOOL RK_InstallDriverA(LPCSTR lpszDriverInfPathName);
#ifdef _UNICODE
#define RK_InstallDriver RK_InstallDriverW

```
#else
#define RK_InstallDriver RK_InstallDriverA
#endif
```

4.4.21. 移除 U 盘

```
/*--------------------------------------------------------------------
Name    :   RK_RemoveRockMsc
Desc    :   移除 U 盘设备
Params  :   (IN)  dwLayer:值为 0 表示对扫描到第一个连接设备进行操作,其他值则对特定
设备进行操作
Return  :   TRUE:                   成功
            FALSE:                  失败
Notes   :
Author  :   Steven Chen
--------------------------------------------------------------------*/
BOOL  RK_RemoveRockMsc(DWORD dwLayer=0);
```

4.4.22. 重启 Rockusb 设备

```
/*--------------------------------------------------------------------
Name    :   RK_ResetRockusb
Desc    :   重启 Rockusb 设备
Params  :   (IN)  dwLayer:值为 0 表示对扫描到第一个连接设备进行操作,其他值则对特定
设备进行操作
Return  :   TRUE:                   成功
            FALSE:                  失败
Notes   :
Author  :   Seth
--------------------------------------------------------------------*/
BOOL  RK_ResetRockusb(DWORD dwLayer=0);
```

4.4.23. 获取 MSC 设备盘符

```
/*--------------------------------------------------------------------
Name    :   RK_GetDeviceDrive
Desc    :   获取 MSC 设备盘符
Params  :   (OUT)driver         存放盘符信息
            (IN)  dwLayer:值为 0 表示对扫描到第一个连接设备进行操作,其他值则对特
```

定设备进行操作
Return : TRUE: 成功
FALSE: 失败
Notes :
Author : Seth
----------*/
BOOL RK_GetDeviceDrive(CHAR &drive, DWORD dwLayer=0);

4.4.24. 格式化用户盘

/*----------
Name : RK_FormatDisk
Desc : 格式化用户盘
Params : (IN)lpszVolume: 用户盘卷标
Return : TRUE: 成功
FALSE: 失败
Notes :
Author : Seth
----------*/
BOOL RK_FormatDiskW(LPCWSTR lpszVolume);
BOOL RK_FormatDiskA(LPCSTR lpszVolume);
#ifdef _UNICODE
#define RK_FormatDisk RK_FormatDiskW
#else
#define RK_FormatDisk RK_FormatDiskA
#endif

4.4.25. 拷贝数据到用户盘

/*----------
Name : RK_CopyData
Desc : 拷贝数据到用户盘
Params : (IN)lpszDataPath: 文件或目录的完整路径
Return : TRUE: 成功
FALSE: 失败
Notes :
Author : Seth
----------*/
BOOL RK_CopyDataW(LPCWSTR lpszDataPath);
BOOL RK_CopyDataA(LPCSTR lpszDataPath);
#ifdef _UNICODE

```
#define RK_CopyData RK_CopyDataW
#else
#define RK_CopyData RK_CopyDataA
#endif
```

4.5.批量操作

4.5.1. 批量切换 USB 设备

```
/*--------------------------------------------------------------------
Name      :    MD_SwitchToRockusb
Desc      :    批量从 U 盘设备切换到 Rockusb 设备
Params    :    (OUT)pResult:          设备集执行结果(结果为-1 时不需要判断)
Return    :     TRUE:                 成功
               FALSE:                 失败
Notes     :    切换后不进行等待设备重新连接，请调用后睡眠几秒以进行等待
Author    :    Steven Chen
--------------------------------------------------------------------*/
BOOL   MD_SwitchToRockusb(PMD_RESULT& pResult);
```

4.5.2. 批量升级

```
/*--------------------------------------------------------------------
Name      :    MD_Upgrade
Desc      :    批量升级固件
Params    :    (OUT)Result:           设备集执行结果(结果为-1 时不需要判断)
               (IN)upgradeParam  具体内容请查看升级参数信息说明
Return    :     TRUE:                 成功
               FALSE:                 失败
Notes     :
Author    :    Steven Chen
--------------------------------------------------------------------*/
BOOL   MD_UpgradeW(PMD_RESULT& pResult,
                   ,STRUCT_UPGRADE_PARAM_W &upgradeParam);
BOOL   MD_UpgradeA(PMD_RESULT& pResult,
                   ,STRUCT_UPGRADE_PARAM_A &upgradeParam);
#ifdef _UNICODE
#define   MD_Upgrade MD_UpgradeW
#else
#define   MD_Upgrade MD_UpgradeA
#endif
```

4.5.3. 批量修复

```
/*--------------------------------------------------------------------
Name      :    MD_Restore
Desc      :    批量修复设备
Params    :    (OUT)Result:              设备集执行结果(结果为-1 时不需要判断)
               (IN)upgradeParam  具体内容请查看升级参数信息说明
Return    :    TRUE:                     成功
               FALSE:                    失败
Notes     :
Author    :    Steven Chen
--------------------------------------------------------------------*/
BOOL   MD_RestoreW(PMD_RESULT& pResult,
                   ,STRUCT_UPGRADE_PARAM_W &upgradeParam);
BOOL   MD_RestoreA(PMD_RESULT& pResult,
                   ,STRUCT_UPGRADE_PARAM_A &upgradeParam);
#ifdef _UNICODE
#define   MD_Restore MD_RestoreW
#else
#define   MD_Restore MD_RestoreA
#endif
```