密级：二级

深圳宇川智能系统
二次开发资料之节能通讯动态库

通讯函数说明

修订历史				
版本	说明	作者	批准	生效日期
V1.2	上海金仕达（1.2.0.0）	胥 申 林		2009 年 08 月 12 日
V1.3	上海金仕达（1.3.0.0）	胥 申 林		2009 年 10 月 19 日

使用须知:

1. 本协议仅适用于金仕达定制机型 V1.2.0.0 版本及以上。
2. 本协议为公司开发部机密资料，未经同意，不得以任何形式提供给第三方。
3. 使用本协议前请确认与本公司签定保密协议，使用者请自觉遵守保密内容。
4. 版权所有，公司对非法使用本协议的用户保留法律起诉权利。
5. 免费技术咨询热线: 400 716 3316 。

一. 函数说明

1: HANDLE OpenCommAuto（int CommPort, int BaudVal）

功能：选择不同波特率打开串口

参数：int CommPort ：串口标志(取值如下：0 — Com1，1 — Com2 最大支持 255)

int BaudVal ：通讯波特率（对应拨特率为 9600—115200）。

返回：设备句柄，类似于 4 字节整数，大于 0 为串口设备句柄，小于 0 表示打开串口错误。

注意：目前支持的波特率：9600，14400，19200，38400，57600，115200

2: int CloseComm（HANDLE icdev）

功能：关闭串口

参数：HANDLE icdev：OpenComm() 返回的设备句柄

返回：成功则返回 0，小于 0 见错误代码

3: int JS_ShakeHand(HANDLE icdev, short int Node_Addr, unsigned char *Password, LPSTR IAP_VER, LPSTR MCU_TYPE, unsigned long *Device_UID, int COM_Time)

功能：通讯握手

参数：HANDLE icdev ：OpenComm() 返回的设备句柄

short int Node_Addr ： 为设备地址号（机器号）

unsigned char *Password ： 授权密码

LPSTR IAP_VER ： 硬件当前版本（字符串）

LPSTR MCU_TYPE ： 当前 MCU 类型（字符串）

unsigned long Device_UID： 返回设备物理 ID

int COM_Time ： 通讯超时时间（根据网络状况调节，标准情况为 200MS）

返回：成功则返回 0，小于 0 见错误代码

4: int JS_SEND_IAP(HANDLE icdev, short int Node_Addr, int COM_Time)

功能：进入在线升级模式

参数：HANDLE icdev ：OpenComm() 返回的设备句柄

short int Node_Addr ： 为设备地址号（机器号）

int COM_Time ： 通讯超时时间（根据网络状况调节，标准情况为 200MS）

返回：成功则返回 0，小于 0 见错误代码

5：int JS_SetNodeTime(HANDLE icdev, int Node_Addr, unsigned char *DateTimes, int COM_Time)

功能：设置机器时间

参数：HANDLE icdev ：OpenComm() 返回的设备句柄

int Node_Addr ：为设备地址号

unsigned char *DateTimes ： 返回节点日期时间，格式为“YYMMDDHHMNSS”，如：1999 年 1 月 28 日 10 点 30 分 50 秒送入为 99 01 28 10 30 50 的 6 字节数组

int COM_Time ： 通讯超时时间（根据网络状况调节，标准情况为 200MS）

返回：成功则返回 0，小于 0 见错误代码

6：int JS_GetNodeTime(HANDLE icdev, int Node_Addr, unsigned char *DateTimes, int COM_Time)

功能：读取机器时间

参数：HANDLE icdev ：OpenComm() 返回的设备句柄

int Node_Addr ：为设备地址号

unsigned char *DateTimes ： 返回节点日期时间，格式为“YYMMDDHHMNSS”，如：1999 年 1 月 28 日 10 点 30 分 50 秒为 99 01 28 10 30 50 的 6 字节数组结构

int COM_Time ： 通讯超时时间（根据网络状况调节，标准情况为 200MS）

返回：成功则返回 0，小于 0 见错误代码

7：int JS_GET_SYSTEM_INFO(HANDLE icdev, int Node_Addr, JS_FEE_RATE& fee_rate, int COM_Time)

功能：读取节能系统参数

参数：HANDLE icdev ：OpenComm() 返回的设备句柄

int Node_Addr ：为设备地址号

JS_FEE_RATE& fee_rate： 节能系统参数，详细见 JS_FEE_RATE 结构体说明

int COM_Time ： 通讯超时时间（根据网络状况调节，标准情况为 200MS）

返回：成功则返回 0，小于 0 见错误代码

8：int JS_SET_SYSTEM_INFO(HANDLE icdev, int Node_Addr, JS_FEE_RATE& fee_rate, int COM_Time)

功能：设置节能系统参数

参数：HANDLE icdev ：OpenComm() 返回的设备句柄

int Node_Addr ：为设备地址号

JS_FEE_RATE& fee_rate： 节能系统参数，详细见 JS_FEE_RATE 结构体说明

int COM_Time ： 通讯超时时间（根据网络状况调节，标准情况为 200MS）

返回：成功则返回 0，小于 0 见错误代码

9：int JS_GET_RECORD_INFO(HANDLE icdev, int Node_Addr, JS_REC_INFO& Record_Info, int COM_Time)

功能：读取节能设备状态信息

参数：HANDLE icdev ：OpenComm() 返回的设备句柄

int Node_Addr ：为设备地址号

JS_REC_INFO& Record_Info： 状态信息，详细见 JS_REC_INFO 结构体说明

int COM_Time ： 通讯超时时间（根据网络状况调节，标准情况为 200MS）

返回：成功则返回 0，小于 0 见错误代码

10：int JS_GET_RECORD(HANDLE icdev, int Node_Addr, unsigned long REC_Number, unsigned long Device_UID, unsigned char *Woke_Flag, JS_RECORD& Record, int COM_Time)

功能：采集节能设备数据（自动偏移记录指针）

参数：HANDLE icdev ：OpenComm() 返回的设备句柄

int Node_Addr ：为设备地址号

unsigned long REC_Number： 上次最后采集记录指针（效正指针，如果为 0xffffffff 则表示按系统当前默认指针，采集当前最后一笔交易记录）

unsigned long Device_UID ： 设备物理 ID

unsigned char *Woke_Flag ：

JS_RECORD& Record： 消费数据结构体，详细见 JS_RECORD 结构体说明

int COM_Time ： 通讯超时时间（根据网络状况调节，标准情况为 200MS）

返回：成功则返回 0，小于 0 见错误代码

11: int JS_GET_APPOINTED_RECORD(HANDLE icdev, int Node_Addr, unsigned long REC_Number, unsigned long Device_UID, JS_RECORD& Record, int COM_Time)

功能：指定采集节能设备某一笔数据

参数：HANDLE icdev ：OpenComm() 返回的设备句柄

int Node_Addr ：为设备地址号

unsigned long REC_Number： 指定记录指针数

unsigned long Device_UID： 设备物理 ID

JS_RECORD& Record： 消费数据结构体，详细见 JS_RECORD 结构体说明

int COM_Time ： 通讯超时时间（根据网络状况调节，标准情况为 200MS）

返回：成功则返回 0，小于 0 见错误代码

12：int JS_SET_Black_Version(HANDLE icdev, int Node_Addr, unsigned char *Ver_Data, int COM_Time)

功能：设置黑名单版本

参数：HANDLE icdev ：OpenComm() 返回的设备句柄

int Node_Addr ：节能设备地址

unsigned char *Ver_Data ：版本信息（从初始 000000000000 开始）.

int COM_Time ： 通讯超时时间（根据网络状况调节，标准情况为 200MS）

返回：成功则返回 0，小于 0 见错误代码

13：int JS_GET_Black_Version(HANDLE icdev, int Node_Addr, unsigned char *Ver_Data, int COM_Time)

功能：读取黑名单版本号

参数：HANDLE icdev ：OpenComm() 返回的设备句柄

int Node_Addr ：节能设备地址

unsigned char *Ver_Data ：返回的版本信息

int COM_Time ： 通讯超时时间（根据网络状况调节，标准情况为 200MS）

返回：成功则返回 0，小于 0 见错误代码

14：int JS_SET_Black(HANDLE icdev, int Node_Addr, int Start_Addr, unsigned long Device_UID, unsigned char *Black_Data, int COM_Time)

功能：批量下载黑名单

参数：HANDLE icdev ：OpenComm() 返回的设备句柄
int Node_Addr ：为设备地址号
int Start_Addr ：下载的块地址
unsigned long Device_UID ：设备物理 ID
unsigned char *Black_Data：待下载的黑名单数据包.
返回：成功则返回 0，小于 0 见错误代码

15：int JS_SET_ONE_Black(HANDLE icdev, int Node_Addr, unsigned char Black_Card_Number, unsigned long Device_UID, unsigned char *Black_Data, int COM_Time)
功能：多条黑名单下载
参数：HANDLE icdev ：OpenComm() 返回的设备句柄
int Node_Addr ：为设备地址号
unsigned char Black_Card_Number ：设置黑名单卡数量
unsigned long Device_UID ：设备物理 ID
unsigned char *Black_Data：待下载的黑名单数据包（每个黑名单占 4 个字节，sizeof(Black_Data) = Black_Card_Number * 4 （第一字节为标识 0x03 为解挂、0x55 为挂失、其他无效，后三个字节为卡号， 高字节在前低字节在后））.
返回：成功则返回 0，小于 0 见错误代码

16：int JS_SET_Address(HANDLE icdev, int Node_Addr, int Address_Data, int COM_Time)
功能：设置黑名单版本
参数：HANDLE icdev ： OpenComm() 返回的设备句柄
int Node_Addr ： 节能设备地址
int Address_Data: 新设备地址（允许范围 1-65534）.
int COM_Time ： 通讯超时时间（根据网络状况调节，标准情况为 200MS）
返回：成功则返回 0，小于 0 见错误代码

17：int JS_UID_SET_Address(HANDLE icdev, int Node_Addr, unsigned int Address_Data, unsigned long Device_UID, int COM_Time)
功能：设置黑名单版本
参数：HANDLE icdev ： OpenComm() 返回的设备句柄
int Node_Addr ： 节能设备地址
int Address_Data: 新设备地址（允许范围 1-65534）
unsigned long Device_UID：设备 UID
int COM_Time ： 通讯超时时间（根据网络状况调节，标准情况为 200MS）
返回：成功则返回 0，小于 0 见错误代码

二、错误代码信息

正确	0
串口初始化错	-1
通讯错	-2
通讯校验错	-3

密码错	-4
超时错	-5
参数错	-6
节点处理命令失败	-7
没有授权信息量	-8
没有纪录	-9
没有此命令	-10

三、结构体信息

```
typedef struct
{
    unsigned char  System_Flag;                 //系统工作模式（实时 0x11/常开 0x22/维护 0x33）
    unsigned char  Use_Mode;                    //扣费工作模式（计量 0x33/计时 0x44）
    unsigned char  PassWordRead[8];             //用户卡密码 1
    unsigned char  PassWordWrite[8];            //用户卡密码 2
    unsigned char  Use_Sector;                  //用户卡使用的钱包
    unsigned char  En_Card_Type[4];             //允许使用的卡类
    struct Multilevel_Rate Rate[20];            //单次多阶费率模式（目前支持 20 类）
}JS_FEE_RATE;

struct One_Rate
{
    unsigned char Start_Time;                   // 记费总时间
    unsigned char Use_Uint;                     // 计费单位
    unsigned char Time_Money;                   // 收费金额
};

struct Multilevel_Rate
{
    unsigned char fee_count;                    // 费率个数
    struct  One_Rate  N_Rate[3];                // 多个费率组 ,目前有效 3 级
};

typedef struct                                  //记录指针结构体
{
    unsigned  char  Record_Flag ;               //存储状态（正常 0x00/满 0x33）
    unsigned  long  Record_All_Number ;         //设备总存储条数
    unsigned  long  Record_Get_Number ;         //已采集总条数
    unsigned  long  Record_Save_Number ;        //已记录总条数
    unsigned  char  Reserved;                   //保留（0x00）
}JS_REC_INFO;

typedef struct                                  //消费记录结构体
{
```

```
    unsigned long Record_Number;          //记录号
    unsigned long User_NO;                //用户卡流水号
    unsigned long User_Card_Value;        //用户卡卡余额
    unsigned long User_Used_Value;        //用户卡消费额
    unsigned long Used_Address;           //用户卡交易设备号
    unsigned long Used_Times;             //用户卡交易次数
    unsigned long User_Card_Bag;          //用户卡钱包号
    unsigned char Used_Time[6];           //消费时间
    unsigned char DEV_UID[4];             //设备的物理 ID（全球唯一）
    unsigned char Used_Flag;              //记录状态字（正常记录 0x00）
}JS_RECORD;
```