LK9000

编程接口说明书

高立开元公司
Gicom Enterprises, Inc.

本接口说明书详细介绍了 LK9000 编程接口函数的功能，用法、声明所在的头文件。用户在开发 LK9000 软件时必须阅读该说明书。

一、数据库编程接口说明

DB_Initial

【函数原型】void DB_Initial()
【功能】在使用数据库功能时，对数据库管理模块进行初始化。
【声明头文件】dbms.h
【参数说明】无
【返回值说明】无
【使用方法】在要调用其他操作数据库的函数时，首先要进行数据库管理模块初始化。并且仅被调用一次。
【提示】

DB_Uninitial

【函数原型】void DB_Uninitial()
【功能】释放数据库管理模块所占用的内存、卸载数据库管理模块。
【声明头文件】dbms.h
【参数说明】无
【返回值说明】无
【使用方法】在结束对数据库的操作，最后必须调用该函数卸载数据库管理模块。

DB_Get_Last_Error

【函数原型】char * DB_Get_Last_Error()
【功能】返回数据库操作函数最近一次出错的错误信息。
【声明头文件】dbms.h
【参数说明】无
【返回值说明】返回错误信息的字符串的指针。
【使用方法】
在调用了任何一个的数据库操作函数之后都可以调用该函数捕捉函数运行过程中产生的错误。

```
if (DB_Open("usr/db/demo.dbf", &hdb) != DB_NO_ERROR) {
    MessageBox(DB_Get_Last_Error(), ONLY_OK );
}
```

【提示】

DB_Open

【函数原型】int DB_Open(char *File_Name, DB_HANDLE *DB_Handle)
【功能】打开 DBF 文件
【声明头文件】dbms.h
【参数说明】
1、File_Name DBF 文件名
2、DB_Handle 正确 返回数据库句柄 错误 0
【返回值说明】
DB_ERROR_OPEN_FILE 数据库打开错误
DB_ERROR_FILE_INVALID 数据库文件无效
DB_ERROR_NOT_ENOUGH_MEM 申请内存失败
DB_ERROR_OPEN_INDEX_FILE 打开索引文件失败
DB_NO_ERROR 执行成功
【使用方法】

```
DB_HANDLE hdb;
DB_Open("/usr/db/demo.dbf", &hdb);
```

【提示】
1、“/usr/db/”为存放数据库文件的路径，打开数据库是务必要加入。
2、打开数据库文件时将索引文件一并打开。

DB_Is_Deleted

【函数原型】int DB_Is_Deleted(DB_HANDLE DB_Handle)
【功能】判断当前记录是否已被删除。
【声明头文件】dbms.h
【参数说明】
1、DB_Handle 数据库句柄
【返回值说明】
DB_ERROR_POINT_INVALID 参数 DB_Handle 所指向的数据库无效
DB_ERROR_RECORD_POSITION 记录位置无效
DB_RECORD_DELETED 当前记录已被删除
DB_RECORD_NOT_DELETE 当前记录未被删除
【使用方法】
在数据库记录指针移动到当前记录时，调用该函数判断该记录是否被逻辑删除。

DB_Set_Delete_Filter_On

【函数原型】void DB_Set_Delete_Filter_On (void)
【功能】对已删除的记录进行过滤。
【声明头文件】dbms.h
【参数说明】

【返回值说明】
【使用方法】
对已删除的记录进行过滤后，使用 DB_Go_Next、DB_Go_Prev、DB_Go_Top、DB_Go_Bottom、DB_Scan、DB_Seek_First、DB_Seek_Next 将不会定位到已删除的记录上。DB_Go_RecNo 除外。

DB_Set_Delete_Filter_Off

【函数原型】void DB_Set_Delete_Filter_Off(void)
【功能】解除对已删除的记录的过滤。
【声明头文件】dbms.h
【参数说明】
【返回值说明】
【使用方法】

DB_Go_Next

【函数原型】int DB_Go_Next(DB_HANDLE DB_Handle)
【功能】移动数据库记录指针到下一条记录
【声明头文件】dbms.h
【参数说明】
1、DB_Handle 数据库句柄
【返回值说明】

DB_ERROR_POINT_INVALID	参数 DB_Handle 所指向的数据库无效
DB_ERROR_RECORD_POSITION	记录位置无效
DB_ERROR_FILE_INVALID	数据库文件无效（ 已被破坏 ）
DB_ERROR_MUST_POST_RECORD	记录正处于编辑状态
DB_NO_ERROR	执行成功

【使用方法】
移动数据库记录指针时使用，当前数据库文件一定要在打开状态。
【提示】
如果当前记录处于编辑状态（执行了追加，更改，删除，恢复操作），记录指针是不允许移动的。
DB_post_Rec(DB_HANDLE DB_Handle)函数则将当前记录的修改写入文件，并恢复当前记录的浏览状态
DB_Unpost_Rec(DB_HANDLE DB_Handle)函数则可以放弃了对当前记录的修改，解除当前记录的编辑状态

DB_Go_Prev

【函数原型】int DB_Go_Prev(DB_HANDLE DB_Handle)
【功能】移动数据库记录指针到上一条记录

【声明头文件】dbms.h
【参数说明】
1、DB_Handle 数据库句柄
【返回值说明】

DB_ERROR_POINT_INVALID	参数 DB_Handle 所指向的数据库无效
DB_ERROR_RECORD_POSITION	记录位置无效
DB_ERROR_MUST_POST_RECORD	记录正处于编辑状态
DB_ERROR_FILE_INVALID	数据库文件无效（ 已被破坏 ）
DB_NO_ERROR	执行成功

【使用方法】
移动数据库记录指针时使用，当前数据库文件一定要在打开状态。
【提示】
如果当前记录处于编辑状态（执行了追加，更改，删除，恢复操作），记录指针是不允许移动的。
DB_post_Rec(DB_HANDLE DB_Handle)函数则将当前记录的修改写入文件，并恢复当前记录的浏览状态
DB_Unpost_Rec(DB_HANDLE DB_Handle)函数则可以放弃了对当前记录的修改，解除当前记录的编辑状态

DB_Go_Top

【函数原型】int DB_Go_Top(DB_HANDLE DB_Handle)
【功能】移动数据库记录指针到第一条记录
【声明头文件】dbms.h
【参数说明】
1、DB_Handle 数据库句柄
【返回值说明】

DB_ERROR_POINT_INVALID	参数 DB_Handle 所指向的数据库无效
DB_ERROR_RECORD_POSITION	记录位置无效
DB_ERROR_MUST_POST_RECORD	记录正处于编辑状态
DB_ERROR_FILE_INVALID	数据库文件无效（ 已被破坏 ）
DB_NO_ERROR	执行成功

【使用方法】
移动数据库记录指针时使用，当前数据库文件一定要在打开状态。
【提示】
如果当前记录处于编辑状态（执行了追加，更改，删除，恢复操作），记录指针是不允许移动的。
DB_post_Rec(DB_HANDLE DB_Handle)函数则将当前记录的修改写入文件，并恢复当前记录的浏览状态
DB_Unpost_Rec(DB_HANDLE DB_Handle)函数则可以放弃了对当前记录的修改，解除当前记录的编辑状态

DB_Go_Bottom

【函数原型】int DB_Go_Bottom(DB_HANDLE DB_Handle)
【功能】移动数据库记录指针到最后一条记录
【声明头文件】dbms.h
【参数说明】
1、DB_Handle 数据库句柄
【返回值说明】

DB_ERROR_POINT_INVALID	参数 DB_Handle 所指向的数据库无效
DB_ERROR_RECORD_POSITION	记录位置无效
DB_ERROR_MUST_POST_RECORD	记录正处于编辑状态
DB_ERROR_FILE_INVALID	数据库文件无效（ 已被破坏 ）
DB_NO_ERROR	执行成功

【使用方法】
移动数据库记录指针时使用，当前数据库文件一定要在打开状态。
【提示】
如果当前记录处于编辑状态（执行了追加，更改，删除，恢复操作），记录指针是不允许移动的。
DB_post_Rec(DB_HANDLE DB_Handle)函数则将当前记录的修改写入文件，并恢复当前记录的浏览状态
DB_Unpost_Rec(DB_HANDLE DB_Handle)函数则可以放弃了对当前记录的修改，解除当前记录的编辑状态

DB_Go_RecNo

【函数原型】int DB_Go_RecNo(DB_HANDLE DB_Handle, unsigned long Rec_No)
【功能】移动数据库记录指针到指定记录号的记录上。
【声明头文件】dbms.h
【参数说明】
1、 DB_Handle 数据库句柄
2、 Rec_No 记录号
【返回值说明】

DB_ERROR_POINT_INVALID	参数 DB_Handle 所指向的数据库无效
DB_ERROR_RECORD_POSITION	记录位置无效
DB_ERROR_MUST_POST_RECORD	记录正处于编辑状态
DB_ERROR_FILE_INVALID	数据库文件无效（ 已被破坏 ）
DB_NO_ERROR	执行成功

【使用方法】
移动数据库记录指针时使用，当前数据库文件一定要在打开状态。
【提示】
如果当前记录处于编辑状态（执行了追加，更改，删除，恢复操作），记录指针是不允许移动的。
DB_post_Rec(DB_HANDLE DB_Handle)函数则将当前记录的修改写入文件，并恢

复当前记录的浏览状态

DB_Unpost_Rec(DB_HANDLE DB_Handle)函数则可以放弃了对当前记录的修改，解除当前记录的编辑状态

DB_Append_Rec

【函数原型】int DB_Append_Rec(DB_HANDLE DB_Handle)
【功能】在数据库末尾追加一条空记录
【声明头文件】dbms.h
【参数说明】

1、 DB_Handle 数据库句柄

【返回值说明】

DB_ERROR_POINT_INVALID	参数 DB_Handle 所指向的数据库无效
DB_ERROR_MUST_POST_RECORD	记录正处于编辑状态
DB_NO_ERROR	执行成功

【使用方法】

追加新记录时调用该函数。

【提示】

记录指针会自动指向新添加的记录，并进入编辑状态，此时记录指针不可移动。

DB_post_Rec(DB_HANDLE DB_Handle)函数则将当前记录的修改写入文件，并恢复当前记录的浏览状态.

DB_Unpost_Rec(DB_HANDLE DB_Handle)函数则可以放弃了对追加记录的修改，并定位至最后一条记录.

默认值：

C 字符型字段为空格
N 数值型字段为 0
D 日期型字段为 1970-01-01
L 逻辑型字段为 F

DB_Delete_Rec

【函数原型】int DB_Delete_Rec(DB_HANDLE DB_Handle)
【功能】给当前记录添加删除标记
【声明头文件】dbms.h
【参数说明】

1、DB_Handle 数据库句柄

【返回值说明】

DB_ERROR_POINT_INVALID	参数 DB_Handle 所指向的数据库无效
DB_ERROR_RECORD_POSITION	记录位置无效
DB_NO_ERROR	执行成功

【使用方法】

调用该函数对当前记录进行逻辑删除

【提示】

加入删除标记后，记录内容及索引文件都不会发生变化
进入编辑状态，此时记录指针不可移动
使用 DB_Post_Rec(DB_HANDLE DB_Handle)更新数据库后生效
使用 DB_Unpost_Rec(DB_HANDLE DB_Handle)取消编辑

DB_Recall_Rec

【函数原型】int DB_Recall_Rec(DB_HANDLE DB_Handle)
【功能】恢复已删除的当前记录
【声明头文件】dbms.h
【参数说明】
1、DB_Handle 数据库句柄
【返回值说明】

DB_ERROR_POINT_INVALID	参数 DB_Handle 所指向的数据库无效
DB_ERROR_RECORD_POSITION	记录位置无效
DB_NO_ERROR	执行成功

【使用方法】
若当前记录为已被删除的记录，可以调用该函数对该记录进行恢复。
【提示】
进入编辑状态，此时记录指针不可移动
使用 DB_Post_Rec(DB_HANDLE DB_Handle)更新数据库后生效
使用 DB_Unpost_Rec(DB_HANDLE DB_Handle)取消编辑

DB_Get_Field_Val

【函数原型】
int DB_Get_Field_Val(DB_HANDLE DB_Handle , char *Field_Name, char *Data, char Size)
【功能】通过字段名读取当前记录的字段
【声明头文件】dbms.h
【参数说明】

1、DB_Handle	数据库句柄
2、Field_Name	字段名
3、Data	存放返回字段值的缓冲区，由用户提供的长度为参数 Size 的数组
4、Size	存放返回字段值的缓冲区的长度（大于 0 小于 255）

【返回值说明】

DB_ERROR_POINT_INVALID	参数 DB_Handle 所指向的数据库无效
DB_ERROR_RECORD_POSITION	记录位置无效
DB_ERROR_FIELD_NAME	参数 Field_Name 所指定的字段名错误
DB_ERROR_FIELD_SIZE	缓冲区的长度错误，缓冲区的长度应大于字段长度。
B_ERROR_FILE_INVALID	数据库文件无效（ 已被破坏 ）

DB_NO_ERROR 执行成功

【使用方法】

读当前记录中的某一个字段的值时，调用该函数。

【提示】

当处于编辑状态中该函数返回修改后的字段值

DB_Get_Field_Val_With_No

【函数原型】

```
int DB_Get_Field_Val_With_No(DB_HANDLE DB_Handle, char No,
char *Data, char Size)
```

【功能】通过字段序号读取当前记录的字段值。

【声明头文件】dbms.h

【参数说明】

DB_Handle 数据库句柄

No 字段序号

Data 存放返回字段值的缓冲区，由用户提供的长度为参数 Size 的数组

Size 存放返回字段值的缓冲区的长度（大于 0 小于 255）

【返回值说明】

DB_ERROR_POINT_INVALID 参数 DB_Handle 所指向的数据库无效

DB_ERROR_RECORD_POSITION 记录位置无效

DB_ERROR_FIELD_NAME 参数 No 所指定的字段名错误

DB_ERROR_FIELD_SIZE 缓冲区的长度错误，缓冲区的长度应大于字段长度。

实际长度

DB_ERROR_FILE_INVALID 数据库文件无效（ 已被破坏 ）

DB_NO_ERROR 执行成功

【使用方法】

在知道字段顺序号的情况下调用该函数取得字段的值

【提示】

当处于编辑状态中该函数返回修改后的字段值

DB_Set_Field_Value

【函数原型】

```
int DB_Set_Field_Value(DB_HANDLE DB_Handle, char *Field_Name,
char *Data)
```

【功能】通过字段名更新当前记录的字段值

【声明头文件】dbms.h

【参数说明】

1、DB_Handle 数据库句柄

2、Field_Name 字段名

3、Data 要更新字段值的缓冲区

【返回值说明】

DB_ERROR_POINT_INVALID 参数 DB_Handle 所指向的数据库无效
DB_ERROR_RECORD_POSITION 当前记录位置无效，数据库为空
DB_ERROR_FIELD_NAME 参数 Field_Name 所指定的字段名错误
DB_ERROR_FILE_INVALID 数据库文件无效（ 已被破坏 ）
DB_ERROR_FIELD_TYPE Data 中的内容与字段类型不符
DB_ERROR_FIELD_SIZE Data 的长度大于字段规定的长度
DB_NO_ERROR 执行成功

【使用方法】

【提示】

进入编辑状态，此时记录指针不可移动
使用 DB_Post_Rec(DB_HANDLE DB_Handle)更新数据库后生效
使用 DB_Unpost_Rec(DB_HANDLE DB_Handle)取消编辑

C 字符型 字符串
N 数值型 十进制数
D 日期型 8 位 例如 20050112
L 逻辑型 1 位 T 为真 F 为假

DB_Set_Field_Val_With_No

【函数原型】int DB_Set_Field_Val_With_No(DB_HANDLE DB_Handle, int No, char *Data)

【功能】通用字段序号更新当前记录的字段。

【声明头文件】dbms.h

【参数说明】

1、DB_Handle 数据库句柄
2、No 字段名
3、Data 要更新字段值的缓冲区

【返回值说明】

DB_ERROR_POINT_INVALID 参数 DB_Handle 所指向的数据库无效
DB_ERROR_RECORD_POSITION 记录位置
DB_ERROR_FIELD_NAME 参数 No 所指定的字段名错误
DB_ERROR_FILE_INVALID 数据库文件无效（ 已被破坏 ）
DB_ERROR_FIELD_TYPE Data 中的内容与字段类型不符
DB_ERROR_FIELD_SIZE Data 的长度大于字段规定的长度
DB_ERROR_UPDATE_INDEX 更新索引文件错误
DB_NO_ERROR 执行成功

【使用方法】

【提示】

进入编辑状态，此时记录指针不可移动
使用 DB_Post_Rec(DB_HANDLE DB_Handle)更新数据库后生效
使用 DB_Unpost_Rec(DB_HANDLE DB_Handle)取消编辑

C 字符型

N 数值型 十进制数
D 日期型 8 位 例如 20050112
L 逻辑型 1 位 T 为真 F 为假

DB_Set_Rec_Content

【函数原型】int DB_Set_Rec_Content(DB_HANDLE DB_Handle , const char * fmt , ...)
【功能】以变参的方式修改当前记录值。
【声明头文件】dbms.h
【参数说明】
DB_Handle 数据库句柄
const char * fmt,... 变参
【返回值说明】

DB_ERROR_POINT_INVALID	参数 DB_Handle 所指向的数据库无效
DB_ERROR_RECORD_POSITION	当前记录位置无效，数据库为空
DB_ERROR_FIELD_NAME	参数 Field_Name 所指定的字段名错误
DB_ERROR_FILE_INVALID	数据库文件无效（ 已被破坏 ）
DB_ERROR_FIELD_TYPE	Data 中的内容与字段类型不符
DB_ERROR_FIELD_SIZE	Data 的长度大于字段规定的长度
DB_NO_ERROR	执行成功

【使用方法】
调用格式 DB_Set_Rec_Content(hdb,“字段名=字段值,字段名=字段值”,参量表) 字段值可以使用如%d、%s 等参量，具体使用方式可以参考 C 语言中的 sprintf 函数。
【提示】
参考 examples\new_modi\main.c

DB_Append_Rec_Content

【函数原型】int DB_Append_Rec_Content(DB_HANDLE DB_Handle , const char * fmt , ...)
【功能】以变参的方式追加一条记录。
【声明头文件】dbms.h
【参数说明】
DB_Handle 数据库句柄
const char * fmt,... 变参
【返回值说明】

DB_ERROR_POINT_INVALID	参数 DB_Handle 所指向的数据库无效
DB_ERROR_RECORD_POSITION	当前记录位置无效，数据库为空
DB_ERROR_FIELD_NAME	参数 Field_Name 所指定的字段名错误
DB_ERROR_FILE_INVALID	数据库文件无效（ 已被破坏 ）
DB_ERROR_FIELD_TYPE	Data 中的内容与字段类型不符
DB_ERROR_FIELD_SIZE	Data 的长度大于字段规定的长度

DB_NO_ERROR 执行成功

【使用方法】

调用格式 DB_Append_Rec_Content(hdb,“字段名=字段值,字段名=字段值”,参量表),字段值可以使用如%d、%s 等参量，具体使用方式可以参考 C 语言中的 sprintf 函数。

【提示】

参考 examples\new_modi\main.c

DB_Close

【函数原型】int DB_Close(DB_HANDLE DB_Handle)

【功能】关闭数据库文件

【声明头文件】dbms.h

【参数说明】DB_Handle 数据库句柄

【返回值说明】

DB_ERROR_POINT_INVALID 参数 DB_Handle 所指向的数据库无效

DB_NO_ERROR 执行成功

【使用方法】

【提示】

连打开的索引一并关闭

DB_Delete_All

【函数原型】int DB_Delete_All(DB_HANDLE DB_Handle)

【功能】

清空数据库文件

【声明头文件】dbms.h

【参数说明】

1、DB_Handle 数据库句柄

【返回值说明】

DB_ERROR_POINT_INVALID 参数 DB_Handle 所指向的数据库无效

DB_NO_ERROR 执行成功

【使用方法】

【提示】

连同索引文件一同清空

DB_Scan

【函数原型】

int DB_Scan(DB_HANDLE DB_Handle, char *Field, char *Value)

【功能】搜索数据库

【声明头文件】dbms.h

【参数说明】

1、DB_Handle 数据库句柄
2、Field 要查找数据的字段
3、Value 要查找的数据

【返回值说明】

DB_ERROR_POINT_INVALID 参数 DB_Handle 所指向的数据库无效
DB_ERROR_MUST_POST_RECORD 记录正处于编辑状态
DB_ERROR_RECORD_POSITION 记录位置无效
DB_ERROR_FIELD_NAME 错误的字段名
DB_FIND_RECORD 发现记录
DB_RECORD_NOT_EXIST 记录不存在

【使用方法】

【提示】

从当前记录开始，向下顺序搜索记录

DB_Get_RecCnt

【函数原型】int DB_Get_RecCnt(DB_HANDLE DB_Handle , unsigned long *RetV)

【功能】得到数据库中的记录总数

【声明头文件】dbms.h

【参数说明】

1、DB_Handle 数据库句柄
2、RetV 返回记录总数

【返回值说明】

DB_ERROR_POINT_INVALID 参数 DB_Handle 所指向的数据库无效
DB_NO_ERROR 执行成功

【使用方法】

【提示】返回所有记录数,包括已逻辑删除的

DB_Get_RecNo

【函数原型】

int DB_Get_RecNo(DB_HANDLE DB_Handle, unsigned long *RetV)

【功能】得到当前记录号

【声明头文件】dbms.h

【参数说明】

1、DB_Handle 数据库句柄
2、RetV 返回当前记录号

【返回值说明】

DB_ERROR_POINT_INVALID 参数 DB_Handle 所指向的数据库无效
DB_NO_ERROR 执行成功

【使用方法】

【提示】

DB_Get_Field_Name

【函数原型】int DB_Get_Field_Name(DB_HANDLE DB_Handle, int No,
char *Retval)
【功能】根据字段序号得到字段名称
【声明头文件】dbms.h
【参数说明】
1、DB_Handle 数据库句柄
2、No 字段序号
3、Retval 返回字段名称
【返回值说明】
DB_ERROR_POINT_INVALID 参数 DB_Handle 所指向的数据库无效
DB_ERROR_FIELD_NO 错误的字段号
DB_NO_ERROR 执行成功
【使用方法】
【提示】

DB_Get_Field_Cnt

【函数原型】int DB_Get_Field_Cnt(DB_HANDLE DB_Handle, int *Retval)
【功能】得到字段的总数
【声明头文件】dbms.h
【参数说明】
1、DB_Handle 数据库句柄
2、Retval 返回字段总数
【返回值说明】
DB_ERROR_POINT_INVALID 参数 DB_Handle 所指向的数据库无效
DB_NO_ERROR 执行成功
【使用方法】
【提示】

DB_Get_Field_Type

【函数原型】int DB_Get_Field_Type(DB_HANDLE DB_Handle,
char *Field_Name, char *Retval)
【功能】通过字段名得到该字段的类型
【声明头文件】dbms.h
【参数说明】
1、DB_Handle 数据库句柄
2、Field_Name 字段名称
2、Retval 返回字段类型
【返回值说明】

DB_ERROR_POINT_INVALID 参数 DB_Handle 所指向的数据库无效
DB_ERROR_FIELD_NAME 错误字段名
DB_NO_ERROR 执行成功

【使用方法】

【提示】

C 字符型
N 数值型 十进制数
D 日期型 8 位 例如 20050112
L 逻辑型 1 位 T 为真 F 为假

DB_Get_Field_Len

【函数原型】int DB_Get_Field_Len(DB_HANDLE DB_Handle,
char *Field_Name, int *Retval)

【功能】通过字段名得到该字段的长度

【声明头文件】dbms.h

【参数说明】

1、DB_Handle 数据库句柄
2、Field_Name 字段名称
2、Retval 返回字段长度

【返回值说明】

DB_ERROR_POINT_INVALID 参数 DB_Handle 所指向的数据库无效
DB_ERROR_FIELD_NAME 错误字段名
DB_NO_ERROR 执行成功

【使用方法】

【提示】

DB_Post_Rec

【函数原型】

int DB_Post_Rec(DB_HANDLE DB_Handle) ;

【功能】

提交已修改的记录

【声明头文件】dbms.h

【参数说明】

1、DB_Handle 数据库句柄

【返回值说明】

DB_ERROR_POINT_INVALID 参数 DB_Handle 所指向的数据库无效
DB_ERROR_UPDATE_FAIL 更新文件失败

DB_ERROR_UPDATE_INDEX_FAIL 更新索引文件失败

【使用方法】

【提示】

DB_Unpost_Rec

【函数原型】

```
int DB_Unpost_Rec( DB_HANDLE DB_Handle ) ;
```

【功能】

恢复已修改的记录

【声明头文件】dbms.h

【参数说明】

1、DB_Handle 数据库句柄

【返回值说明】

DB_ERROR_POINT_INVALID 参数 DB_Handle 所指向的数据库无效

DB_ERROR_UPDATE_FAIL 更新文件失败

DB_ERROR_UPDATE_INDEX_FAIL 更新索引文件失败

【使用方法】

【提示】

在没有调用 DB_Post_Rec(DB_HANDLE DB_Handle)前，如果只是对当前记录进行了修改 DB_Unpost_Rec 会将当前记录恢复为原先的内容，如果是追加的记录，则放弃添加新的记录，并将记录指针定位于最后一条记录

DB_Create_DBF

【函数原型】

```
int DB_Create_DBF( char *name,TAG_FIELD_DESC Fields[] , unsigned int Field_Cnt )
```

【功能】

建立数据库文件

【声明头文件】dbms.h

【参数说明】

1、DB_Handle 数据库句柄

2、TAG_FIELD_DESC Fields[] 字段信息

```
typedef struct { // TAG_FIELD_DESC 结构体成员
    char name[11];          //字段名    只允许十个字符
    char type;              //字段类型
    unsigned char len;      //字段长度
    unsigned char bit;      //小数位数
}TAG_FIELD_DESC;
```

3、unsigned int Field_Cnt 字段个数

【返回值说明】

DB_ERROR_OVER_FIELD_NUMBER 超过字段最大数（30）

DB_ERROR_FIELD_NAME 错误的字段名
DB_ERROR_FIELD_TYPE 错误的字段类型
DB_ERROR_FIELD_SIZE 错误的字段长度
DB_ERROR_OPEN_FILE 不能打开文件
DB_NO_ERROR 成功

【使用方法】

使用方法详见 examples\ dbms_create \ dbms_create_DB.c

【提示】

1、此函数只是生成数据库文件，并不将其打开。
2、字段名只能由大写字母、数字、和'_'组成。
3、字段名第一个字符必须为大写字母。
4、字段名不能为关键字。
关键字有 "CHAR" , "DATE" , "DATETIME" ,"FLOAT" ,"INTEGER" , "LONGCHAR" , "NO"。

DB_Create_Index

【函数原型】int DB_Create_Index(DB_HANDLE DB_Handle,char *Key_Field, unsigned int Bucket_Num);

【功能】 建立一个空的索引文件

【声明头文件】dbms.h

【参数说明】

1、DB_Handle 数据库句柄
2、char *Key_Field 关键字
3、Bucket_Num 桶数，对文件中记录数的估计值

【返回值说明】

DB_ERROR_POINT_INVALID 参数 DB_Handle 所指向的数据库无效
DB_ERROR_KEY_FIELD Key_Field 非法
DB_ERROR_CREATE_INDEX 建立索引文件失败

【使用方法】

【提示】

可以递增的方式建多个索引文件，如已有 a.hix 、a.h01 两个索引文件，再执行此函数时会建立第三个索引文件 a.h02。

最多可建立 8 个索引文件。

关键字可由多个字段组成 例如 SN+PN+CN

Key_Field 中不可以有空格

DB_Rebuild_Index

【函数原型】int DB_Rebuild_Index(DB_HANDLE DB_Handle, unsigned int Bucket_Num)

【功能】 对数据库中已有记录进行重新索引

【声明头文件】dbms.h

【参数说明】

1、DB_Handle 数据库句柄

2、Bucket_Num 桶数 对文件中记录数的估计值

【返回值说明】

DB_ERROR_POINT_INVALID 参数 DB_Handle 所指向的数据库无效

DB_ERROR_KEY_FIELD Key_Field 非法

DB_ERROR_CREATE_INDEX 建立索引文件失败

【使用方法】

【提示】

对数据库进行重建索引之前该数据库必须已有索引文件。

如果 Bucket_Num = 0 则使用 DB_Create_Index 函数时所用到的 Bucket_Num

DB_Seek_First

【函数原型】int DB_Seek_First(DB_HANDLE DB_Handle, char * Value)

【功能】在当前索引文件中按关键字内容搜索记录

【声明头文件】dbms.h

【参数说明】

1、DB_Handle 数据库句柄

2、Value 要查找的关键的值

【返回值说明】

DB_ERROR_POINT_INVALID 参数 DB_Handle 所指向的数据库无效

DB_ERROR_INDEX_FILE_NOT_OPEN 索引文件没有打开

DB_ERROR_INDEX 索引发生错误

DB_RECORD_NOT_EXIST 没有满足条件的记录

DB_FIND_RECORD 已找到记录

【使用方法】

【提示】

关键字可由多个字段组成

例如： SN+PN+CN SN 长度为 15 PN 长度为 10 CN 长度为 5，要查找 SN="001" PN="002" CN="003" 的记录。

```
char Tmp[SN_LEN+PN_LEN+CNLEN+1] = {0};
sprintf(Tmp, "%-15s%-10s%-5s", sn, pn, cn);
DB_Seek_First(hdb, Tmp);
```

关键字的顺序按数据库的字段的顺序为准。

例如：数据库中字段为 SN、PN、CN，以 PN+CN+SN 为关键字建立索引。

如果是按关键字段的顺序组合要查找的字段，是无法查找到记录的。即：

```
sprintf(Tmp, "%-10s%-5s%-15s ", pn, cn, sn);
```

只有按照数据库中的字段顺序，才能查找到相应的记录：

```
sprintf(Tmp, "%-15s%-10s%-5s", sn, pn, cn);
```

DB_Seek_Next

【函数原型】int DB_Seek_Next(DB_HANDLE DB_Handle)
【功能】在当前索引文件中继续按关键字内容搜索记录
【声明头文件】dbms.h
【参数说明】
1、DB_Handle 数据库句柄
【返回值说明】

DB_ERROR_POINT_INVALID	参数 DB_Handle 所指向的数据库无效
DB_ERROR_INDEX_FILE_NOT_OPEN	索引文件没有打开
DB_ERROR_INDEX	索引发生错误
DB_RECORD_NOT_EXIST	没有满足条件的记录
DB_FIND_RECORD	已找到记录

【使用方法】
【提示】
当使用 DB_Seek_First 函数后，如有相同的关键字，可用 DB_Seek_Next 继续搜索。

DB_Set_Index_With_No

【函数原型】int DB_Set_Index_With_No(DB_HANDLE DB_Handle , int index_no)
【功能】根据索引文件号，选择当前索引文件。
【声明头文件】dbms.h
【参数说明】
1、DB_Handle 数据库句柄
2、index_no 索引文件号
0 对应 .hix
1 对应 .h01
.
.
7 对应 .h07
【返回值说明】

DB_ERROR_POINT_INVALID	参数 DB_Handle 所指向的数据库无效
DB_ERROR_INDEX_FILE_NOT_OPEN	索引文件没有打开
DB_ERROR_INDEX	索引发生错误
DB_NO_ERROR	成功

【使用方法】
使用方法详见 examples\db_multi_find\dbms_find.c
【提示】

DB_Set_Index_With_Key

【函数原型】int DB_Set_Index_With_No(DB_HANDLE DB_Handle , char* key)
【功能】根据索引关键字，选择当前索引文件。
【声明头文件】dbms.h
【参数说明】
1、DB_Handle 数据库句柄
2、key 关键字
【返回值说明】

DB_ERROR_POINT_INVALID	参数 DB_Handle 所指向的数据库无效
DB_ERROR_INDEX_FILE_NOT_OPEN	索引文件没有打开
DB_ERROR_KEY_FIELD	关键字错误
DB_ERROR_INDEX	索引发生错误
DB_NO_ERROR	成功

【使用方法】
使用方法详见 examples\db_multi_find\dbms_find.c
【提示】
最多可建 8 个索引文件

DB_Get_Key_Val

【函数原型】int DB_Get_Key_Val(DB_HANDLE DB_Handle, char *Key_Value);
【功能】得到关键字的值
【声明头文件】dbms.h
【参数说明】
1、DB_Handle 数据库句柄
【返回值说明】

DB_ERROR_POINT_INVALID	参数 DB_Handle 所指向的数据库无效
DB_ERROR_INDEX_FILE_NOT_OPEN	索引文件没有打开
DB_ERROR_INDEX	索引发生错误
DB_RECORD_NOT_EXIST	没有满足条件的记录
DB_FIND_RECORD	已找到记录

【使用方法】
【提示】
返回关键字的值的长度与该字段的长度一致,相差部分由空格补齐。
例如 关键字 CN 长度为 4 值为"05" 则返回"05 "
在关键字为多个字段时，则返回字段拼在一起的字符串
例如 关键字为 SN+PN+CN 其中 SN = "01" PN = "02" CN = "03" SN,PN,CN 字段长度都为 4
返回值为"01 02 03"

二、系统接口函数说明

sys_select_ime

【函数原型】int sys_select_ime(int state)
【功能】选择输入法，并且屏蔽 ALP 键。
【声明头文件】sys_fun.h
【参数说明】

int state 输入法类型

取值范围如下：

```
#define IME_NUM     0x01   // 数字
#define IME_LOWER   0x02   // 小写字符
#define IME_UPPER   0x04   // 大写字符
#define IME_CN      0x08   // 中文
#define IME_ALL     0x0f   // 输入法全开
```

【返回值说明】

0 类型错误

1 成功

【使用方法】

只输入数字：

```
sys_select_ime( IME_NUM ) ;
```

允许输入数字及小写字符：

```
sys_select_ime( IME_NUM|IME_LOWER ) ;
```

【提示】

sys_open_laser

【函数原型】int sys_open_laser()
【功能】打开激光模块
【声明头文件】sys_fun.h
【参数说明】无
【返回值说明】

0 失败

1 成功

【提示】

1、打开激光模块后，扫描键有效。
2、扫描数据时,系统会自动在扫描出的条码信息的头和尾部加入开始和结束标志，格式为:

_KEY _LASER _BEGIN + 条码信息 + _KEY _LASER _END

3、系统默认激光模块为关状态。

sys_close_laser

【函数原型】int sys_close_laser();
【功能】关闭激光模块
【声明头文件】sys_fun.h
【参数说明】无
【返回值说明】
0 失败
1 成功

sys_get_barcode_type

【函数原型】void sys_get_barcode_type(unsigned char *value , unsigned char *type , unsigned char *barcode);
【功能】获取条码类型。
【声明头文件】sys_fun.h
【参数说明】
unsigned char *value 扫描器得到的全部数据
unsigned char *type 返回条码类型
unsigned char *barcode 返回条码内容
【提示】
此函数中有在打开“系统管理程序/系统设置/激光头扫描头设置/报告条码类型”选项后才可以正常执行。

sys_get_serial_no

【函数原型】unsigned char * sys_get_serial_no();
【功能】获取设备序列号
【声明头文件】sys_fun.h
【参数说明】无
【返回值说明】
0 失败 否则 序列号字串

sys_set_font_16

【函数原型】void sys_set_font_16(void)
【功能】将字体设置为 16 点阵，可显示 8 行*16 列 ASCII 字符。
【声明头文件】sys_fun.h

【参数说明】无

sys_set_font_12

【函数原型】void sys_set_font_12(void)
【功能】将字体设置为 12 点阵，可显示 10 行*21 列 ASCII 字符。
【声明头文件】sys_fun.h
【参数说明】无

sys_beep

【函数原型】void sys_beep(int time)
【功能】让蜂鸣器发出一段时间的提示音。
【声明头文件】sys_fun.h
【参数说明】
Int time 提示音持续的时间（毫秒）。

sys_ shake

【函数原型】void sys_shake(int time)
【功能】让手持机震动一段时间。
【声明头文件】sys_fun.h
【参数说明】
Int time 震动持续的时间（毫秒）。

sys_ shake_beep

【函数原型】void sys_shake_beep(int time)
【功能】同时震动与发音。
【声明头文件】sys_fun.h
【参数说明】
Int time 持续的时间（毫秒）。

三、界面接口函数说明

MessageBox

【函数原型】int MessageBox(char *Message, int Type)

【功能】消息框--用户按相应键后消失
【声明头文件】MessageBox.h
【参数说明】
char * Message 消息框中要显示的信息，可用'\n'换行
int Type 消息框类型
取值范围如下：
ONLY_OK 1 按任意键返回 (空出标题栏) （可显示 4 行信息）
OK_CANCEL 2 ESC 键取消 'G'或'确认'键确认 (空出标题栏)
【返回值说明】
返回值的取值范围如下：
TYPE_ERROR 0 输入的类型错误
SYSTEM_ERROR -1 系统错误(内存不足)

如果消息框类型为 ONLY_OK 返回用户所按的键值
如果消息框类型为 OK_CANCEL :
用户按 ESC 值 返回 KEY_F(12)
用户按'G'或'_KEY_ENTER'键返回_KEY_ENTER

OpenPopWin

【函数原型】POPWIN OpenPopWin(char *Msg)
【功能】由用户编程控制消息框的打开与关闭
【声明头文件】MessageBox.h
【参数说明】
char * Message 消息框中要显示的字符串(只可以有一行信息)
【返回值说明】
弹出式消息框返回值：
0 创建消息框失败
否则返回消息框句柄

ClosePopWin

【函数原型】void ClosePopWin(POPWIN PopWin)
【功能】关闭由 OpenPopWin(char *Msg)函数所创建的消息框
【声明头文件】MessageBox.h
【参数说明】
POPWIN PopWin 消息框句柄

PopWinMsg

【函数原型】void PopWinMsg(POPWIN PopWin, char *Msg)
【功能】更改由 OpenPopWin(char *Msg)函数所创建的消息框的显示内容

【声明头文件】MessageBox.h
【参数说明】
POPWIN PopWin 消息框句柄
char * Msg 新信息

Creat_Window_Menu

【函数原型】MENU *Creat_Window_Menu(char(*name)[ITEM_LEN], char(*desc)[ITEM_LEN], int row_cnt, int display_rows, int row, int top_row)
【功能】
建立菜单
【声明头文件】Menu_extend.h
【参数说明】
char (*name)[ITEM_LEN] 菜单项名称数组
char (*desc)[ITEM_LEN] 菜单项描述数组
int row_cnt 总行数
int display_rows 可显示的总行数
int row 在屏幕第几行开始显示(0 - 9)
int top_row 从第几项菜单开始显示
【返回值说明】
成功 返回菜单句柄
失败 返回 0

Free_Menu

【函数原型】void Free_Menu(MENU *Menu)
【功能】释放菜单
【声明头文件】Menu_extend.h
【参数说明】
MENU *Menu 用 Creat_Window_Menu 生成好的菜单句柄

draw_rectangle

【函数原型】int draw_rectangle(int lup_row , int lup_col , int rdown_row, int rdown_col)
【功能】画一个矩形
【声明头文件】MessageBox.h
【参数说明】
lup_row 矩形左上角行坐标
lup_col 矩形左上角列坐标

rdown_row 矩形右下角行坐标
rdown_col 矩形右下角列坐标

draw_line

【函数原型】int draw_line(int row , int start_col , int end_col)
【功能】画一条水平直线
【声明头文件】MessageBox.h
【参数说明】
row 水平直线所在行坐标
start_col 水平直线开始的列坐标
end_col 水平直线结束的列坐标

Get_Selected_Menu_Item

【函数原型】int Get_Selected_Menu_Item(MENU *Menu)
【功能】根据用户选择返回相应的菜单项的索引
【声明头文件】Menu_extend.h
【参数说明】
MENU *Menu 用 Creat_Window_Menu 生成好的菜单句柄
【返回值说明】
-1 系统异常
0 用户按 ESC 键
> 0 用户按"G"或"确认"键所返回的菜单项的索引值

四、通讯接口函数说明

open_infrared

【函数原型】int open_infrared(char *reason)
【功能】打开红外端口
【声明头文件】infrared.h
【参数说明】
char * reason 如果打开端口失败 返回错误原因
【返回值说明】
0 失败
1 成功

infrared_print

【函数原型】int infrared_print(char *buf)
【功能】向红外端口输出字符串
【声明头文件】infrared.h
【参数说明】
char * buf 向红外端口输出的字符串
【返回值说明】
0 失败
1 成功

close_infrared

【函数原型】void close_infrared()
【功能】关闭红外端口
【声明头文件】infrared.h
【参数说明】无

download_file

【函数原型】int download_file(char *DB_Name, char *Reason)
【功能】通过串口下载文件
【声明头文件】com2pc.h
【参数说明】
char *DB_Name 如使用 NORMAL_MODE 模式通讯返回已下载文件名（不带路径）。
如使用 BACKCOM_MODE 模式通讯要下载文件名称（不带路径）。
char *Reason 当下载失败时返回错误的原因 长度为 20 字节的字符串
【返回值说明】
0 下载失败
1 下载成功
【使用方法】
【提示】
该接口默认在波特率 115200 下以 YMODEM 协议传输文件。如需更改波特率请使用 set_baud_rate 函数。
可在文件传输过程中，按 ESC 中断通讯。

upload_file

【函数原型】int upload_file(char *DB_Name, char *Reason)
【功能】通过串口上传文件
【声明头文件】com2pc.h

【参数说明】

char *DB_Name 要上传的文件名 （不带路径）

char *Reason 当上传失败时返回错误的原因 长度为 20 字节的字符串

【返回值说明】

0 上传失败

1 上传成功

【使用方法】

【提示】

该接口默认在波特率 115200 下以 YMODEM 协议传输文件。如需更改波特率请使用 set_baud_rate 函数。

可在文件传输过程中，按 ESC 中断通讯。

irda_download_file

【函数原型】int irda_download_file(char *DB_Name, char *Reason)

【功能】通过红外下载文件

【声明头文件】com2pc.h

【参数说明】

char *DB_Name 返回 已下载文件名 （不带路径）

char *Reason 当下载失败时返回错误的原因 长度为 20 字节的字符串

【返回值说明】

0 下载失败

1 下载成功

【使用方法】

【提示】

该接口默认在波特率 115200 下以 YMODEM 协议传输文件。如需更改波特率请使用 set_baud_rate 函数。

可在文件传输过程中，按 ESC 中断通讯。

Irda_upload_file

【函数原型】int irda_upload_file(char *DB_Name, char *Reason)

【功能】通过红外上传文件

【声明头文件】com2pc.h

【参数说明】

char *DB_Name 要上传的文件名 （不带路径）

char *Reason 当上传失败时返回错误的原因 长度为 20 字节的字符串

【返回值说明】

0 上传失败

1 上传成功

【使用方法】

【提示】

该接口默认在波特率 115200 下以 YMODEM 协议传输文件。如需更改波特率请使用 set_baud_rate 函数。

可在文件传输过程中，按 ESC 中断通讯。

modem_download_file

【函数原型】int modem_download_file(char *DB_Name, char *phone , char *Reason)

【功能】通过外接 modem 下载文件

【声明头文件】com2pc.h

【参数说明】

char *DB_Name 返回 已下载文件名 （不带路径）

char *phone 电话号码

char *Reason 当下载失败时返回错误的原因 长度为 20 字节的字符串

【返回值说明】

0 下载失败

1 下载成功

【使用方法】

【提示】

可在文件传输过程中，按 ESC 中断通讯。

modem_upload_file

【函数原型】int modem_upload_file(char *DB_Name, char *phone ,char *Reason)

【功能】通过外接 modem 上传文件

【声明头文件】com2pc.h

【参数说明】

char *DB_Name 要上传的文件名 （不带路径）

char *phone 电话号码

char *Reason 当上传失败时返回错误的原因 长度为 20 字节的字符串

【返回值说明】

0 上传失败

1 上传成功

【使用方法】

【提示】

可在文件传输过程中，按 ESC 中断通讯。

set_baud_rate

【函数原型】void set_baud_rate(int baut_rate)
【功能】设置串口通讯的波特率
【声明头文件】com2pc.h
【参数说明】
baut_rate 串口波特率 如 9600、57600 、115200

set_com_mode

【函数原型】int set_com_mode(int mode)
【功能】设置通讯模式（普通串口通讯模式与 BACKCOM 通讯模式）
【声明头文件】com2pc.h
【参数说明】

Mode	NORMAL_MODE	普通串口通讯模式（默认）
	BACKCOM_MODE	BACKCOM通讯模式

【返回值说明】
0 设置失败
1 设置成功
【使用方法】
BACKCOM_MODE BACKCOM 通讯模式是用于同 BACKCOM 控件通讯。
NORMAL_MODE 使用标准 YMODEM 协议进行通讯。
【提示】
使用方法详见 examples\com\com.c

五、GPRS 相关函数

gsm_open

【函数原型】int gsm_open(void)
【功能】打开通讯模块。
【声明头文件】gprs.h
【参数说明】无
【返回值说明】
0 成功 -1 失败

gprs_init()

【函数原型】int gprs_init(void)
【功能】初始化 GPRS 模块。
【声明头文件】gprs.h
【参数说明】无
【返回值说明】
0 成功 -1 失败

gprs_dial()

【函数原型】int gprs_dial(void)
【功能】拨号并将终端连接至公网。
【声明头文件】gprs.h
【参数说明】无
【返回值说明】
0 成功 -1 失败

ibs_connect ()

【函数原型】int ibs_connect(char *UserName ,
char *PWD ,
char *IP_Adder ,
unsigned short int Port ,
unsigned int TimeOut)
【功能】连接 IBS 服务器。
【声明头文件】gprs.h
【参数说明】 char *UserName IBS 用户名
char *PWD IBS 密码
char *IP_Adder IBS 服务器密码
unsigned short int Port IBS 服务器端口号
unsigned int TimeOut 超时时间
【返回值说明】
TCPIP_ERR_SOCKET_COMM_FAIL 通讯失败
TCPIP_ERR_SOCKET_CREATE_FAIL 无法创建 SOCKET
TCPIP_ERR_SOCKET_CONNECT_FAIL 与服务器建立连接失败
IBC_ERR_PROTOCOL 与 IBS 服务器发生通讯协议错误
IBC_ERR_RECEIVE_ZERO 没有收到数据
IBC_NOERROR 连接成功

ibs_lookup ()

【函数原型】short int ibs_lookup(unsigned char * pBuf,
unsigned int cbSend,
unsigned int dwBufLen,
unsigned int * pcbRecv)
【功能】 向 IBS 服务器发送数据，并等待和接收由 IBS 服务器反馈的数据。
【声明头文件】gprs.h
【参数说明】

unsigned char * pBuf	数据缓冲区用于发送和接收。
unsigned int cbSend	发送数据的长度。
unsigned int dwBufLen	缓冲区实际长度。
unsigned int * pcbRecv	返回接收到的数据长度。

【返回值说明】

TCPIP_ERR_SOCKET_COMM_FAIL	通讯失败
IBC_ERR_OVER_BUFFER_LENGTH	接收到的数据长度大于缓冲的长度。
IBC_NOERROR	成功

ibs_disconnect()

【函数原型】int ibs_disconnect (void)
【功能】断开与 IBS 服务器的连接。
【声明头文件】gprs.h
【参数说明】无
【返回值说明】

0	成功
1	TCPIP_ERR_SOCKET_CLOSE_FAIL 失败

gprs_hangup ()

【函数原型】int gprs_hangup (void)
【功能】挂断 GPRS。
【声明头文件】gprs.h
【参数说明】无
【返回值说明】

0	成功	-1	失败

gsm_close ()

【函数原型】int gsm_close (void)
【功能】关闭通讯模块。
【声明头文件】gprs.h
【参数说明】无
【返回值说明】
0 成功 -1 失败