

实时库数据网关驱动规范

(V5.0)

编写：

审核：

更新日期： 2018.5.11

重庆唐码软件有限公司

www.kipway.com

内容目录

1 概述.....	1
2.接口函数.....	1
2.1 dac_getversion.....	1
2.2 dac_create.....	1
2.3 dac_destroy.....	2
2.4 dac_getalltags.....	2
2.5 dac_getalltagsex.....	3
2.6 dac_start.....	3
2.7 dac_stop.....	3
2.8 dac_isrun.....	3
2.9 dac_devicewriteval.....	4
2.10 dac_devicewriteobj.....	4
3 常数定义.....	4
3.1 错误码.....	4
3.2 日志类型.....	5
3.3 标签访问属性.....	5

1 概述

实时数据库数据网关驱动规范采用约定导出接口函数的动态连接库封装，所有导出函数采用 C 方式导出。在 windows 下为标准 DLL 动态库文件，在 Linux 下为 SO 动态连接库文件。开发工具推荐使用 C/C++，其他能生成满足此规范的动态库的开发工具也可。驱动有以下几个主要特点：

- 支持多线程，同一个驱动可生成多个驱动实例。
- 数据上行提交采用回调模式，包括实时数据，SOE 事件，日志信息。
- 支持数据的下传(向设备发送控制指令)
- 支持动态加载卸载和启动停止。
- 支持驱动自解析配置文件。

由于规定了导出函数名和导出函数的参数，因此使用驱动程序的数据网关可采用动态加载的方式来使用驱动程序，并动态的创建和销毁驱动实例。

2.接口函数

接口函数分为和实例相关的接口和无关的接口。dac_getversion 和 dac_create 为实例无关的接口，其余为实例相关接口。

所谓实例就是一个具体驱动的设备，比如一个连接到 OPC DA 的客户端或者一个 MODBUS 总线。一个实例根据具体的应用层协议可能对应一个具体设备或者一组具体设备。如果在面向对象的概念中，一个驱动程序就像一个类(class),而驱动实例就是该类产生的对象(object)。

在本规范中，使用一个大于等于 0 的整数来表示一个具体的驱动实例。

具体接口的定义参见 rdbdef.h 和 dacdrv.h 文件

2.1 dac_getversion

```
int dac_getversion();
```

功能：获取支持驱动程序的规范版本。为了编译支持以后的规范升级设计的，目前固定返回 500 表示 Ver5.00

参数：无

返回值：500

2.2 dac_create

```
int dac_create(const char* sdevicename, const char* scfgfile,  
              dac_onmsg onmsg, void* onmsgparam,
```

```

    dac_ontagval ontagval,void* ontagvalparam,

    dac_ontagobj ontagobj,void *ontagobjparam,

    dac_onsoe onsoe ,void* onsoeparam

);

```

功能：创建一个驱动实例。一个驱动实例使用一个大于等于 0 的整数表示，和 Linux 中的文件句柄类似，和面向对象的 this 指针的功能一样。

参数：

sdevicename [in] 设备名

scfgfile [in] 配置文件名，全路径

onmsg [in] 消息信息和日志回调函数

onmsgparam [in] 消息信息和日志回调函数的 param 参数

dac_ontagval [in] 值标签数据到达通知回调函数

ontagvalparam [in] 值标签数据到达通知回调函数的参数

ontagobj [in] 对象标签数据到达通知回调函数

ontagobjparam [in] 对象标签数据到达通知回调函数的参数

onsoe [in] SOE 事件到达通知回调函数

ontagobjparam [in] SOE 事件到达通知回调函数的参数

返回值：

返回驱动实例句柄（大于等于 0 的整数），如果失败，返回-1。

2.3 dac_destroy

```
void dac_destroy(int h);
```

功能：销毁驱动实例，回收资源，相当于面向对象的体系中析构函数。

参数：

h [in] dac_create 创建的句柄

返回值：无，驱动实例删除后就不能再被使用。

2.4 dac_getalltags

```
int dac_getalltags(int h,dac_ontag ontag, void* ontagparam);
```

功能：读取所有标签

参数:

h dac_create 创建的句柄

ontag 标签获取回掉函数

ontagparam [in] ontag 的参数

返回值: 返回 0 表示成功,否则为错误码

2.5 dac_getalltagsex

int dac_getalltagsex(int h, dac_ontagex ontag, void* ontagparam);

功能: 读取所有标签扩展版, 多了工程单位和描述两个字段信息。

参数:

h dac_create 创建的句柄

ontag 标签获取回掉函数

ontagparam [in] ontag 的参数

返回值: 返回 0 表示成功,否则为错误码

2.6 dac_start

int dac_start(int h);

功能: 启动驱动实例, 驱动实例应该自带线程来对具体设备的数据进行轮询采集, 该接口让驱动实例运行起来。

参数:

h dac_create 创建的句柄

返回值: 返回 0 表示成功,否则为错误码

2.7 dac_stop

int dac_stop(int h);

功能: 停止驱动实例运行。

参数:

h dac_create 创建的句柄

返回值: 返回 0 表示成功,否则为错误码。

2.8 dac_isrun

bool dac_isrun(int h);

功能：判断驱动实例是否在运行。

参数：

h dac_create 创建的句柄

返回值：返回 0 表示没有运行，非 0 表示正在运行。

2.9 dac_devicewriteval

```
int dac_devicewriteval(int h, rec_tagval *pval);
```

功能：这是输出控制接口，向设备输出一个数值

参数：

h dac_create 创建的句柄

pval 标签值，可以是 TD_DIGITAL,DT_INT32,DT_FLOAT32,DT_INT64, DT_FLOAT64 类型

返回值：返回 0 表示成功,否则为错误码

2.10 dac_devicewriteobj

```
int dac_devicewriteobj(int h, rec_tagobj *pobj);
```

功能：这是输出控制接口，向设备输出一个对象(或一个字符串或一个二进制数据块)。

参数：

h dac_create 创建的句柄

pobj 标签对象，可以是 DT_STRINGT 或 DT_OBJECT 类型

返回值：返回 0 表示成功,否则为错误码

3 常数定义

3.1 错误码

```
#define DACRET_ERRHANDLE (-1) //!<错误的句柄
```

```
#define DACRET_OK 0 //!<成功
```

```
#define DACRET_FAILED 1 //!<失败
```

```
#define DACRET_ERRARGS 2 //!<参数错误
```

```
#define DACRET_NOTAG 3 //!<无此标签
```

```
#define DACRET_TAGACCESS 4 //!<标签访问错误，比如不可写标签
```

```
#define DACRET_CTRLIOERR 5 //!<设备输出 IO 错误
```

```
#define DACRET_CTRLDEV 6 //!<设备操作失败
```

3.2 日志类型

```
#define LOGLEVEL_MSG 0 //!<日志日志
```

```
#define LOGLEVEL_WRN 1 //!<警告日志
```

```
#define LOGLEVEL_ERR 2 //!<错误日志
```

```
#define LOGLEVEL_DBG 3 //!<调试日志
```

3.3 标签访问属性

```
#define DAC_R 1 //只读
```

```
#define DAC_W 2 //只写
```

```
#define DAC_RW 3 //读写
```