Модульная интегрированная

SCADA КРУГ-2000™

Версия 4.4

ФУНКЦИИ API ДОСТУПА К БД

Руководство Пользователя

Модульная интегрированная SCADA КРУГ-2000™. Функции API доступа к БД.
Руководство Пользователя.

ООО НПФ «КРУГ»

440028, г. Пенза, ул. Титова, 1

Телефоны: (841-2) 49-97-75; 49-94-14

E-mail: **support@krug2000.ru**

http:// **www.krug2000.ru**

Обозначение документа: КР01.20134W-4.4.-И2.3.1

📖 СОДЕРЖАНИЕ

1 НАЗНАЧЕНИЕ

API доступа к БД (**KrugDBClient.dll**) предназначен для гибкого взаимодействия «сторонним» клиентским приложениям с базой данных КРУГ-2000, как с локальной, так и с удаленной (рисунок 1).

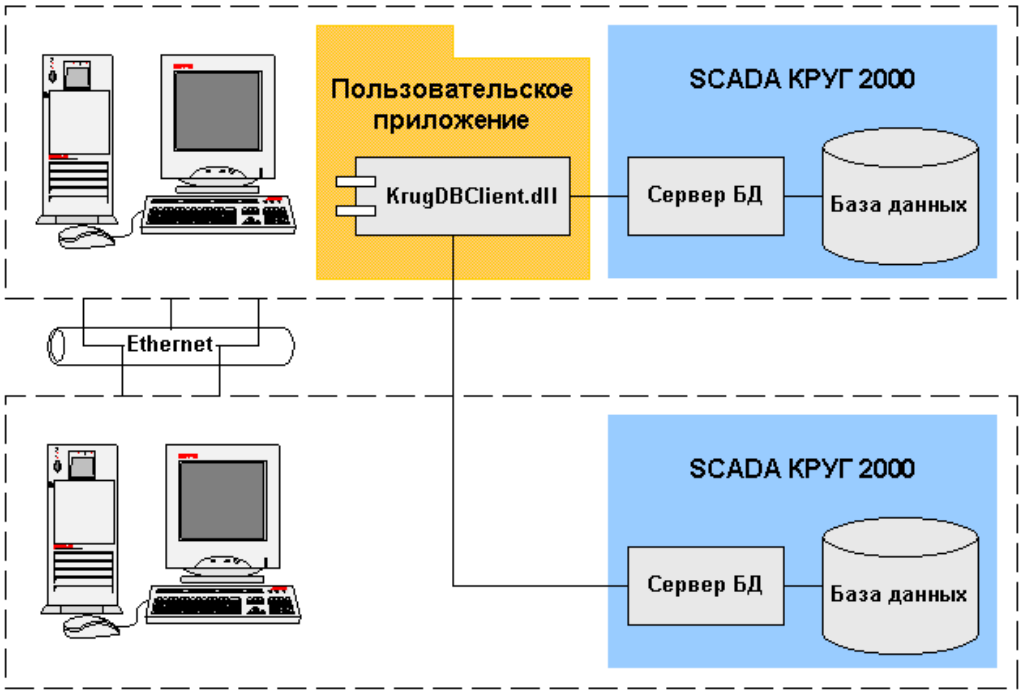


Рисунок 1

Задачей API интерфейса БД является предоставление пользователю функций обмена данными с таблицами переменных Сервера БД КРУГ-2000. Обмен данными производится только с переменными, имеющими ненулевой канал.

API интерфейс БД сопровождается demo-клиентом с открытым кодом. Его назначением является демонстрация использования функций интерфейса.

Основные функции API доступа к БД:

- Взаимодействие с сервером ОБД;
- Работа с переменными и их атрибутами;
- Ведение log-файла.

2 ОПИСАНИЕ ФУНКЦИЙ

HRESULT KrugDBConnect (**LPWSTR** szRemoteMashineName, **DWORD*** pdwConnectID)	
Назначение:	Подключение к серверу
Входные данные:	szRemoteMashineName - имя удаленной машины, либо пустая строка(локальный сервер)
Выходные данные:	pdwConnectID - уникальный идентификатор соединения
Возвращает:	S_OK - успешно E_FAIL – ошибка DB_E_CONNECT_LIMIT - количество соединений ограничено

HRESULT KrugDBDisconnect (**DWORD** dwConnectID)	
Назначение:	Отключение от сервера
Входные данные:	dwConnectID - уникальный идентификатор соединения
Выходные данные:	нет
Возвращает:	S_OK - успешно E_FAIL – ошибка

HRESULT KrugGetServerStatus (**DWORD** dwConnectID, **BOOL*** pbBaseServer)	
Назначение:	Получение статуса сервера (основной или резервный)
Входные данные:	dwConnectID - уникальный идентификатор соединения
Выходные данные:	bBaseServer - статус сервера (TRUE - основной, FALSE - резервый)
Возвращает:	S_OK - успешно E_FAIL - ошибка DB_E_CONNECT - соединение с сервером не найдено

HRESULT KrugDBLog (**BOOL** bEnableLog)	
Назначение:	Ведение log-файла
Входные данные:	bEnableLog - флаг ведения log-файла
Выходные данные:	нет
Возвращает:	S_OK – успешно

HRESULT KrugGetVarIDs (**DWORD** dwConnectID, **DWORD** dwVarType, **DWORD*** pdwVarCount, **DWORD*** pdwVarIDs)	
Назначение:	Получение количества и номеров переменных
Входные данные:	dwConnectID - уникальный идентификатор соединения dwVarType - тип переменной
Выходные данные:	pdwVarCount - количество переменных pdwVarIDs - номера переменных
Возвращает:	S_OK - успешно E_FAIL - ошибка DB_E_CONNECT - соединение с сервером не найдено
Комментарии:	Если указатель pdwVariablesIDs==NULL, то функция возвратит только количество переменных.

HRESULT KrugGetVarValues (**DWORD** dwConnectID, **DWORD** dwVarCount, **VARDATA*** pVarData, **VARIANT*** pvData)	
Назначение:	Получение значений переменных
Входные данные:	dwConnectID - уникальный идентификатор соединения dwVarCount - количество переменных pVarData - данные о переменных
Выходные данные:	pvData - значения переменных

HRESULT KrugGetVarValues (**DWORD** dwConnectID, **DWORD** dwVarCount, **VARDATA*** pVarData, **VARIANT*** pvData)	
Возвращает:	S_OK - успешно E_FAIL - ошибка DB_E_CONNECT - соединение с сервером не найдено

HRESULT KrugSetVarValues (**DWORD** dwConnectID, **DWORD** dwVarCount, **VARDATA*** pVarData, **VARIANT*** pvData)	
Назначение:	Установление значений переменных
Входные данные:	dwConnectID - уникальный идентификатор соединения dwVarCount - количество переменных pVarData - данные о переменных pvData - значения переменных
Выходные данные:	нет
Возвращает:	S_OK - успешно E_FAIL - ошибка DB_E_CONNECT - соединение с сервером не найдено

HRESULT KrugGetAttrInfo (**DWORD** dwConnectID, **DWORD** dwVarType, **DWORD*** pdwAttrCount, **ATTRINFO*** pAttrInfo)	
Назначение:	Получение информации об атрибутах
Входные данные:	dwConnectID - уникальный идентификатор соединения dwVarType - тип переменной
Выходные данные:	dwAttrCount - количество атрибутов pAttrInfo - информация об атрибуте
Возвращает:	S_OK - успешно E_FAIL - ошибка DB_E_CONNECT - соединение с сервером не найдено
Комментарии:	Если указатель pAttrInfo==NULL, то функция возвратит только количество атрибутов.

HRESULT KrugGetAttrValues (**DWORD** dwConnectID, **DWORD** dwAttrCount, **ATTRDATA*** pAttrData, **VARIANT*** pvData, **RESULT*** pResult)	
Назначение:	Получение значений атрибутов переменной
Входные данные:	dwConnectID - уникальный идентификатор соединения dwAttrCount - количество атрибутов pAttrData - данные об атрибутах
Выходные данные:	pvData - значения атрибутов pResult – результаты выполнения по каждому атрибуту
Возвращает:	S_OK - успешно E_FAIL - ошибка S_FALSE - при получении значений возникали ошибки DB_E_CONNECT - соединение с сервером не найдено
Комментарии:	Если элемент pvData имеет тип VT_BSTR, в этом случае необходимо освободить выделенную под строку память.

HRESULT KrugSetAttrValues (**DWORD** dwConnectID, **DWORD** dwAttrCount, **ATTRDATA*** paAttrData, **VARIANT*** pvData, **HRESULT*** pResult)	
Назначение:	Установление значений атрибутов переменной
Входные данные:	dwConnectID - уникальный идентификатор соединения

HRESULT KrugSetAttrValues (**DWORD** dwConnectID, **DWORD** dwAttrCount, **ATTRDATA*** paAttrData, **VARIANT*** pvData, **HRESULT*** pResult)
Выходные данные: dwConnectID - уникальный идентификатор соединения dwAttrcount - количество атрибутов pAttrData - данные о атрибутах pvData - значения атрибутов Выходные данные: pResult – результаты выполнения по каждому атрибуту Возвращает: S_OK - успешно E_FAIL - ошибка S_FALSE - при установлении значений возникали ошибки DB_E_CONNECT - соединение с сервером не найдено

HRESULT KrugRegChangesState (**DWORD** dwConnectID, **BOOL** bActive)
Назначение: Включение/выключение вывода сообщений в роллинг об изменении атрибута переменной через функции данной библиотеки Входные данные: dwConnectID - уникальный идентификатор соединения bActive - флаг регистрации изменений Выходные данные: нет Возвращает: S_OK – успешно DB_E_CONNECT - соединение с сервером не найдено

Описание структур данных, используемых в функциях:

Данные по переменной:

```
typedef struct tagVARDATA
{
        DWORD  m_dwType;        //Тип переменной
        DWORD  m_dwId;          //Номер переменной
}VARDATA;
```

Данные по атрибуту переменной:

```
typedef struct tagATTRDATA
{
        DWORD  m_dwType;        //Тип переменной
        DWORD  m_dwId;          //Номер переменной
        DWORD  m_dwAttrId;      //Номер атрибута
}ATTRDATA;
```

Информация об атрибуте:

```
typedef struct tagATTRINFO
{
        DWORD    m_dwId;          //Номер атрибута
        char     m_szName[64];    //Имя атрибута
        int      m_nType;         //Тип
        int      m_nLen;          //Длина
        bool     m_bReadWrite;    //Доступ (0-чтение,1-чтение/запись)
}ATTRINFO;
```

Определения типов переменных:		
#define **VA**	1	//Входная аналоговая
#define **AV**	2	//Аналоговая выходная
#define **VD**	3	//Входная дискретная
#define **DV**	4	//Дискретная выходная
#define **RV**	5	//РВ составная

На работу с атрибутами переменных с помощью функций API доступа к БД накладывается ряд ограничений. Атрибуты имеют различный уровень доступа. Работа с некоторыми из них запрещена или ограничена (разрешено только чтение).

Доступ к атрибутам переменных типа «Входная аналоговая»:

Номер атрибута	***Имя атрибута***	***Доступ на чтение***	***Доступ на запись***
0	RecordID	+	-
1	Номер канала	+	-
2	Номер переменной в УСО	+	-
70	Коррекция 1(тип/номер)	-	-
71	Коррекция 2(тип/номер	-	-
72	Коррекция 3(Адрес хол Спая)	-	-
n	Все остальные атрибуты	+	+

Доступ к атрибутам переменных типа «Аналоговая выходная»:

Номер атрибута	***Имя атрибута***	***Доступ на чтение***	***Доступ на запись***
0	RecordID	+	-
1	Номер канала	+	-
2	Номер УСО	+	-
8	Позиция (адрес) переменной 1 (ПОЗ)	-	-
9	Позиция (адресс) задания (ПОЗД)	-	-
10	Позиция (адрес) упр Воздействия (ПУВ)	-	-
11	Позиция (адрес) признака ап Упр (ПАУ)	-	-
12	Позиция (адресс) переменной 2 (ВИМ)	-	-
13	Позиция (адресс) переменной 3 (ИМВ)	-	-
n	Все остальные атрибуты	+	+

Доступ к атрибутам переменных типа «Входная дискретная»:

Номер атрибута	*Имя атрибута*	*Доступ на чтение*	*Доступ на запись*
0	RecordID	+	-
1	Номер канала	+	-
2	Номер УСО	+	-
43	Адрес переменной задания	-	-
n	Все остальные атрибуты	+	+

Доступ к атрибутам переменных типа «Дискретная выходная»:

Номер атрибута	*Имя атрибута*	*Доступ на чтение*	*Доступ на запись*
0	RecordID	+	-
1	Номер канала	+	-
2	Номер УСО	+	-
50	Адрес признака аппаратного управления	-	-
51	Адрес переменной задания	-	-
n	Все остальные атрибуты	+	+

Доступ к атрибутам переменных типа «Ручной ввод»:

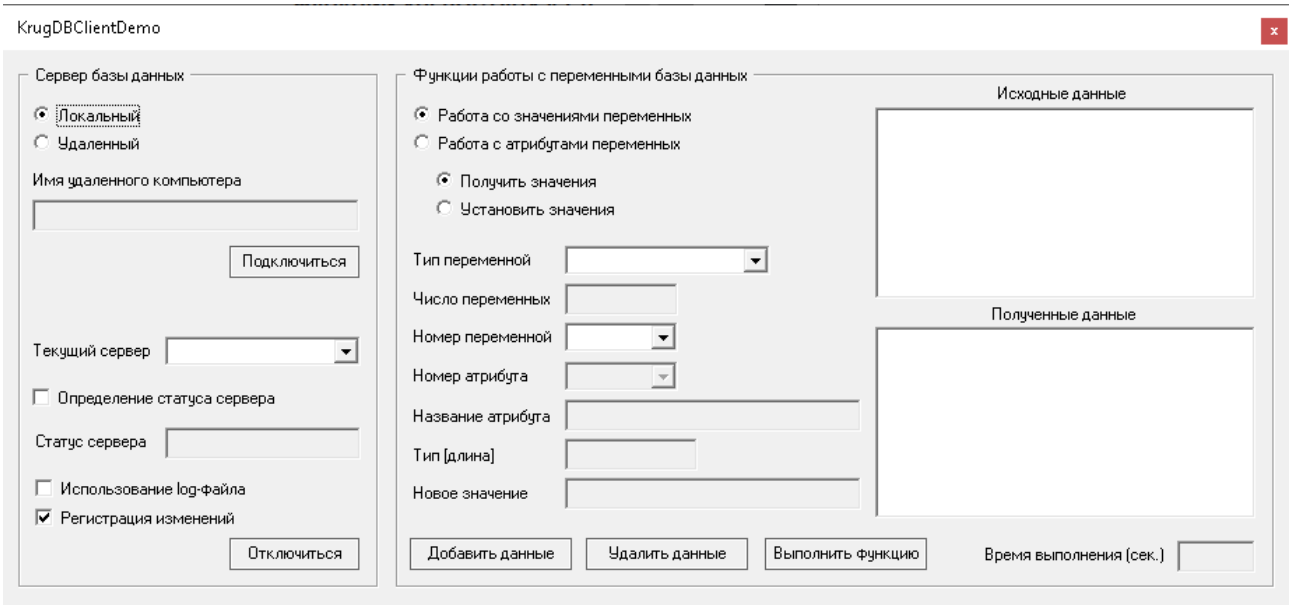
Номер атрибута	*Имя атрибута*	*Доступ на чтение*	*Доступ на запись*
0	RecordID	+	-
1	Номер канала	+	-
2	Номер УСО	+	-
n	Все остальные атрибуты	+	+

3 ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ

3.1 Пример использования API доступа к БД в среде программирования Microsoft Visual C++

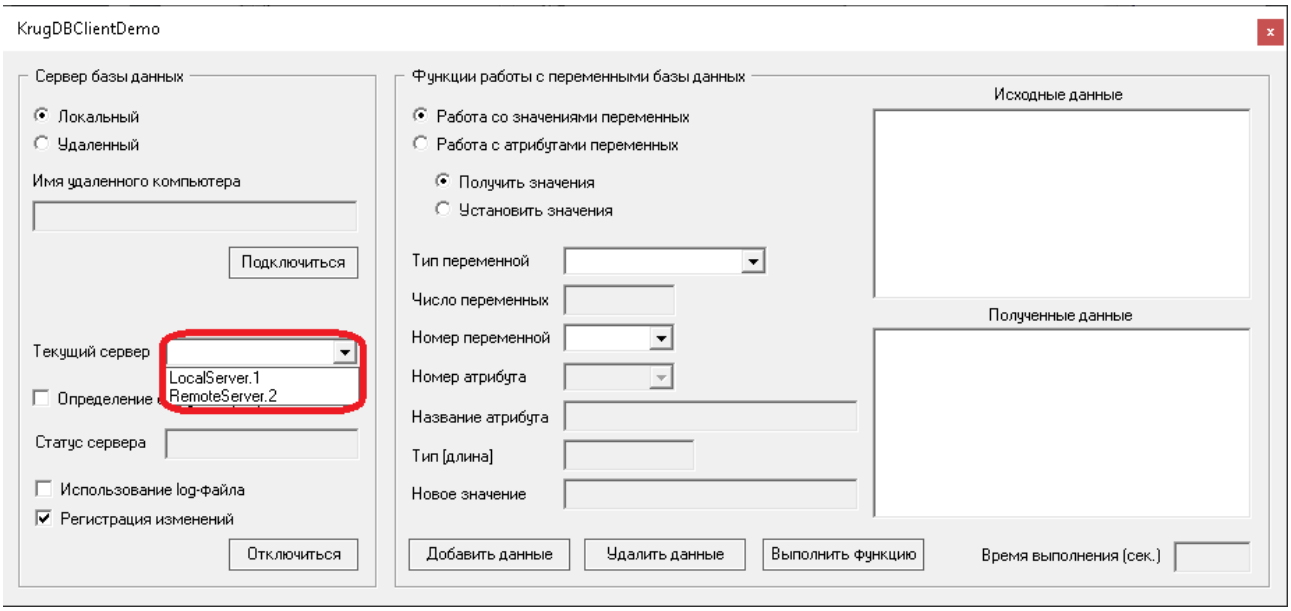
В качестве примера использования функций библиотеки в комплект поставки входит открытый код проекта демонстрационного клиентского приложения KrugDBClientDemo реализованного в среде программирования Microsoft Visual C++ 6.0.

Интерфейс клиентского приложения представляет собой диалоговое окно, на котором размещены элементы управления, каждый из которых связан с вызовом той или иной функции API доступа к БД.



Нажатие кнопки “Подключиться” вызывает функцию **KrugDBConnect**, выполняющую соединение с сервером. По умолчанию, происходит соединение с локальным сервером. Если же выбран удаленный сервер, то активизируется поле ввода для имени удаленного компьютера.

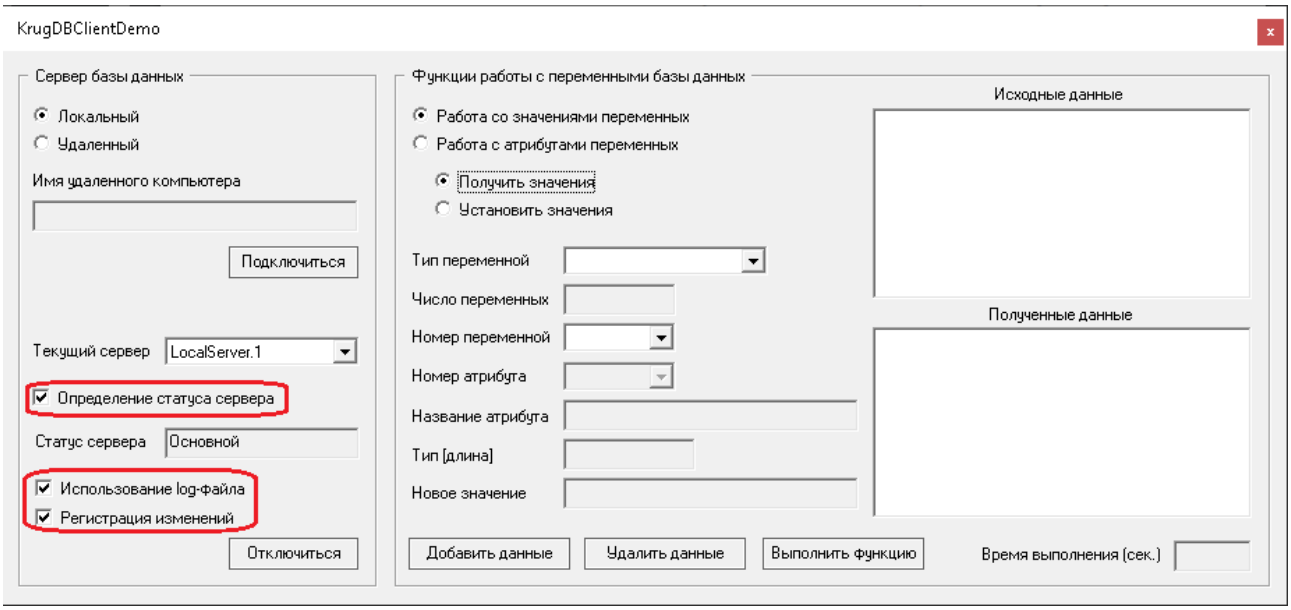
При успешном соединении с сервером данные этого соединения заносятся в список, соответственно, LocalServer или RemoteServer, плюс идентификатор соединения, возвращаемый функцией.
Для работы с необходимо выбрать нужный сервер из выпадающего списка “Текущий сервер”.



Нажатие кнопки “Отключиться” вызывает функцию **KrugDBDisconnect**, которая выполняет отсоединение от сервера указанного в списке “Текущий сервер”.

Выбор или отмена выбора “Использование log-файла” вызывает функцию **KrugDBLog** и передает в нее соответствующий параметр ведения log-файла.
Так же при выборе флажка “Регистрация изменений” вызывается функция **KrugRegChangesState,** в которую передается соединение с текущим сервером и состояние флажка.

Если выбрано “Определение статуса сервера”, то функция **KrugGetServerStatus** будет вызываться через определенный интервал времени. На основе значения, которая она возвращает, в поле “Статус сервера” будет выводиться значение статуса сервера: основной или резервный.



Переключатели группы “Функции работы с переменными базы данных” определяют активность выпадающих списков (Номер переменной, Номер атрибута), и полей ввода (Название атрибута, Новое значение).

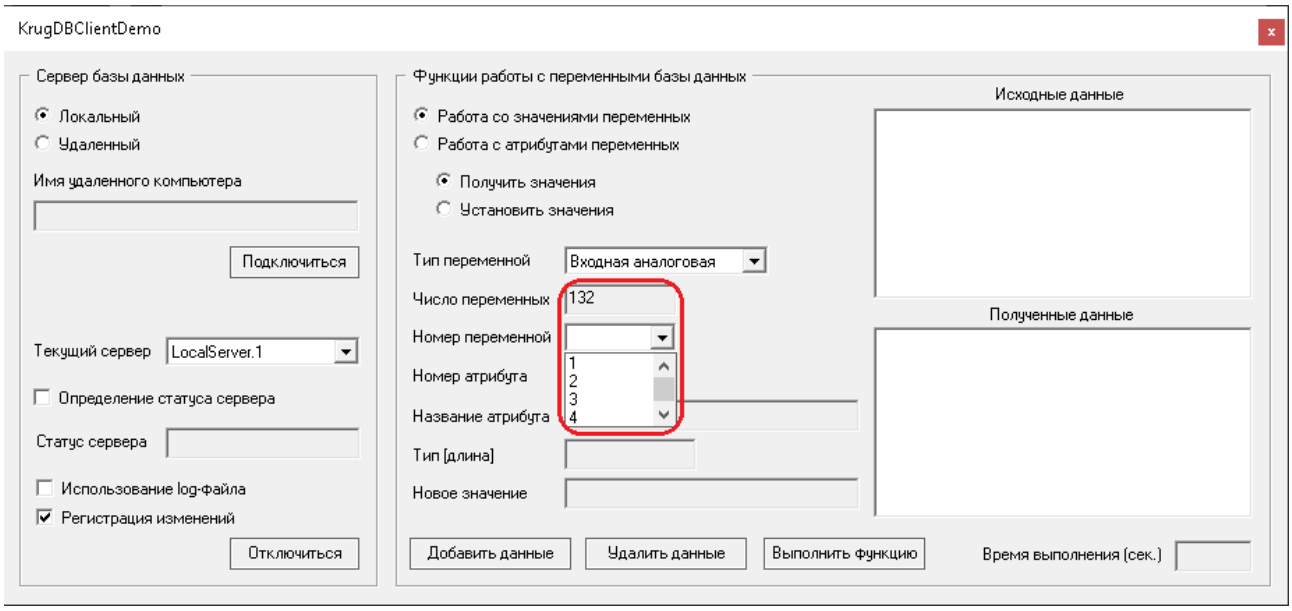
Сочетания переключателей также определяют выбор для выполнения одной из функций.

Функция: **KrugGetVarValues**
Переключатели: Работа со значениями переменных, Получить значения

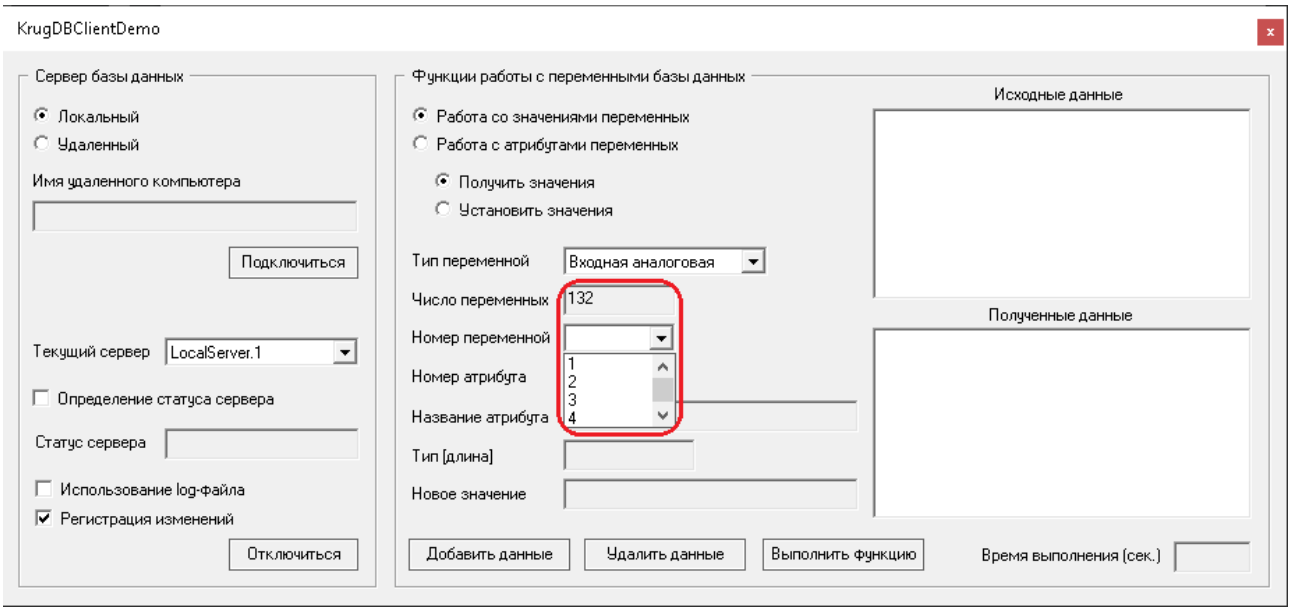
Функция: **KrugSetVarValues**
Переключатели: Работа со значениями переменных, Установить значения

Функция: **KrugGetAttrValues**
Переключатели: Работа с атрибутами переменных, Получить значения

Функция: **KrugSetAttrValues**
Переключатели: Работа с атрибутами переменных, Установить значения

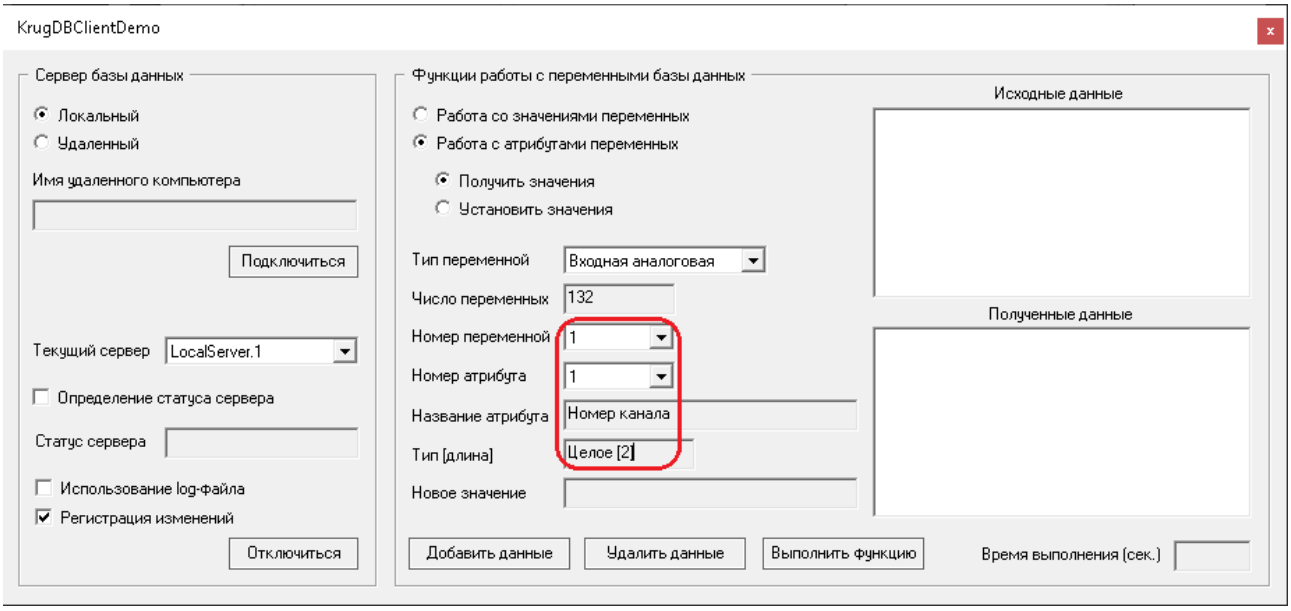


Выпадающий список “Тип переменной” содержит типы переменных: Выбор одного из типов переменных вызывает функцию **KrugGetVarIDs**. Она возвращает количество и номера переменных выбранного типа. Эти значения заносятся соответственно в поле “Число переменных” – число доступных переменных, и в список “Номер переменной” – их номера.



Если выбрана “Работа с атрибутами переменных”, то выпадающий список “Номер атрибута” инициализируется номерами доступных атрибутов. Выбор одного из номеров атрибутов вызывает функцию **KrugGetAttrInfo**. Информация об атрибуте, которую она возвращает, заносится в поля “Название атрибута” и “Тип (длина)”.

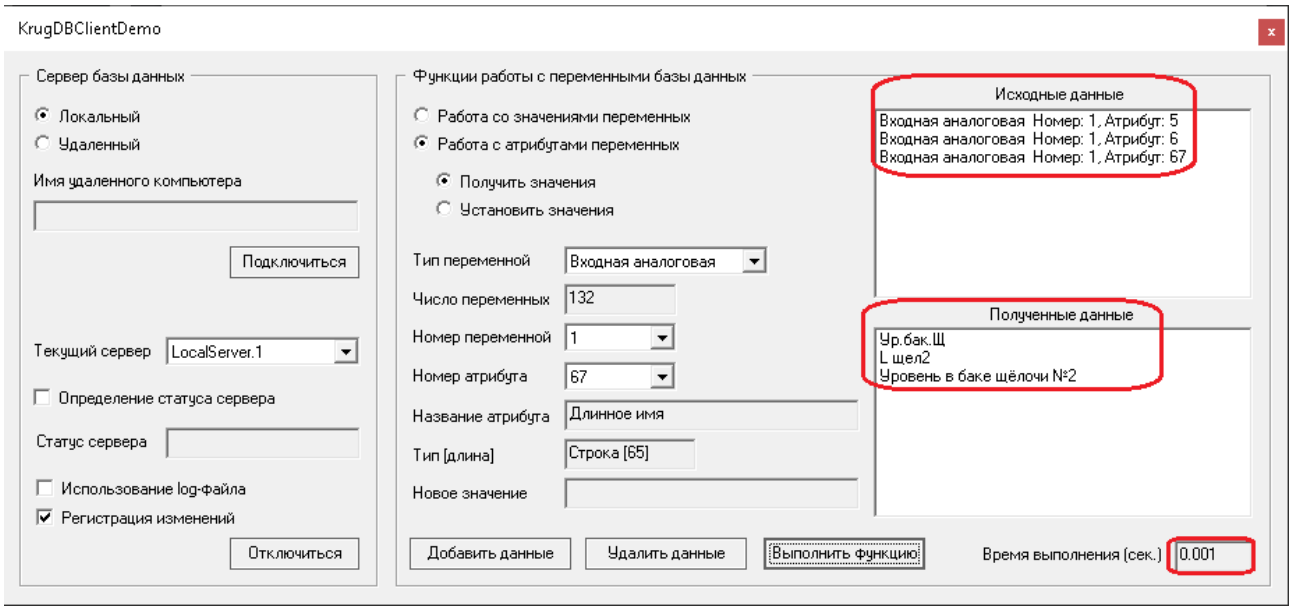
При работе с атрибутами типа “Дата/Время” формат задания значения в поле ввода “Новое значение” должен соответствовать формату, котором оно выводится при получении его значения, т.е. **Час:Минута:Секунда День.Месяц.Год**.



На основе данных в активных полях можно сформировать исходные параметры для функций работы с переменными или их атрибутами.
Нажатием кнопки “Добавить данные” можно занести текущие данные в список “Исходные данные”. Очистить список можно, нажав кнопку “Удалить данные”.

Нажатие кнопки “Выполнить функцию” вызывает выбранную функцию. Параметры функции инициализируются на основе данных списка “Исходные данные”. Данные, возвращаемые функцией, заносятся в список “Полученные данные”.

В поле “Время выполнения” заносится значение времени работы функции в секундах.

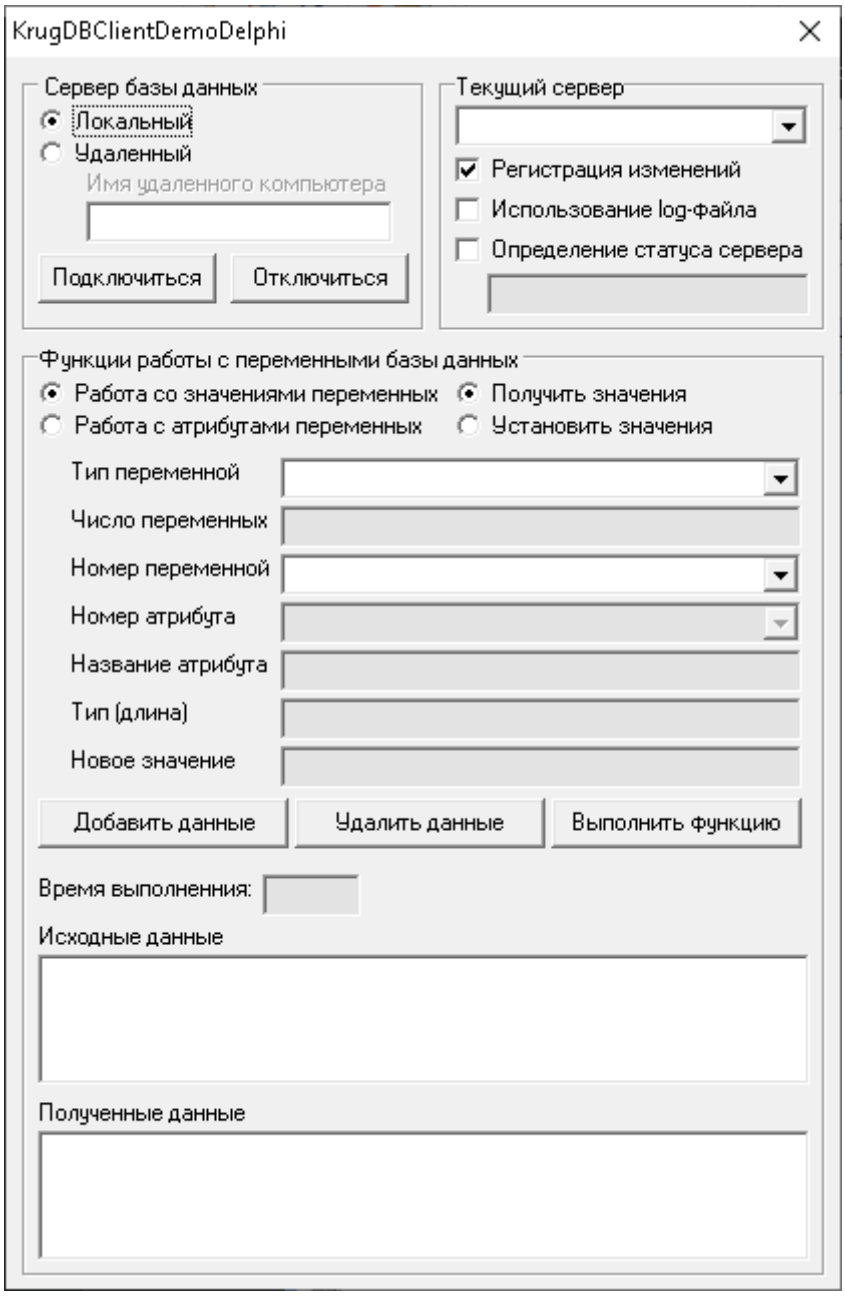


Программа KrugDBClientDemo демонстрирует явный способ подключения библиотеки API доступа к БД (KrugDBClient.dll). Загрузка библиотеки происходит при запуске программы. В директории файла KrugDBClientDemo.exe должны находиться KrugDBClient.dll и KrugDBClient.ini.

3.2 Пример использования API доступа к БД в среде программирования Borland Delphi

В качестве примера использования функций библиотеки в комплект поставки входит открытый код проекта демонстрационного клиентского приложения KrugDBClientDemoDelphi реализованного в среде программирования Borland Delphi 7.0.

Интерфейс клиентского приложения представляет собой диалоговое окно, на котором размещены элементы управления, каждый из которых связан с вызовом той или иной функции API доступа к БД.



Функциональное назначение элементов управления аналогично клиентскому приложению KrugDBClientDemo, рассмотренному в пункте 3.1.

Библиотека **KrugDBClient.dll** подключена к программе неявным способом. Для вызова функций API доступа к БД используется их описание, содержащееся в файле KrugDBClientExports.pas. В директории файла KrugDBClientDemoDelphi.exe должны находиться KrugDBClient.dll и KrugDBClient.ini. Для подробного изучения работы демонстрационного приложения обращайтесь к комментариям в исходном коде программы.

4 РАБОТА С ПРОГРАММОЙ

Есть два основных способа подключить **KrugDBClient.dll** к программе - явный и неявный.

При **неявном** подключении линкеру передается библиотека импорта **KrugDBClient.lib**, содержащая список функций библиотеки, которые могут использовать клиентские приложения. Обнаружив, что программа обращается хотя бы к одной из них, линкер добавляет в целевой exe-файл таблицу импорта.

Таблица импорта содержит список всех DLL, которые использует программа, с указанием конкретных переменных и функций, к которым она обращается. Позже, когда exe-файл будет запущен, загрузчик проецирует все DLL, перечисленные в таблице импорта, на адресное пространство процесса; в случае неудачи весь процесс немедленно завершается.

Во-первых, можно непосредственно добавить файл **KrugDBClient.lib** в проект посредством команды **Project⇒Add to project⇒Files**...

Во-вторых, можно указать имя библиотеки импорта в опциях линкера. Для этого откройте окно настроек проекта (**Project⇒Settings**...) и добавьте в поле **Object/Library modules** на вкладке Link имя **KrugDBClient.lib**.

Наконец, можно встроить ссылку на библиотеку импорта прямо в исходный код программы. Для этого используется директива #pragma с ключем comment. В программу необходимо вставить строчку:

```
#pragma comment(lib,"KrugDBClient.lib")
```

Теперь можно использовать в программе функции, содержащиеся в библиотеке:

```
#include "KrugDBClient.h"


DWORD dwConnectID;
KrugDBConnect(NULL, &dwConnectID);

…

DWORD dwVarCount;
DWORD dwVarIDs[5000];
char chNumber;

if(KrugGetVarIDs(dwConnectID, dwVarType, &dwVarCount, dwVarIDs)==S_OK)
{
        m_VariableNumber.ResetContent();
        for(int i=0;i<(int)dwVarCount;i++)
  {
  itoa(dwVarIDs[i],&chNumber,10);
                m_VariableNumber.AddString(&chNumber);
        }
}
```

При **явном** подключении клиентское приложение вызывает функцию LoadLibrary, которая получает имя библиотеки и возвращает ее дескриптор:

```
HMODULE     m_hDllModule;
m_hDllModule = LoadLibrary("KrugDBClient.dll");
```

После того как библиотека загружена, адрес любой из содержащихся в ней функций можно получить с помощью GetProcAddress, которой необходимо передать дескриптор библиотеки и имя функции. Затем нужную функцию из DLL можно вызывать, используя для нее определение типа, описанное в файле **KrugDBClient.h**.

Например:

```
KRUG_DB_CONNECT fKrugDBConnect;    //тип функции
fKrugDBConnect= (KRUG_DB_CONNECT)GetProcAddress(m_hDllModule,"KrugDBConnect");

KRUG_GET_VAR_IDS fKrugGetVarIDs;
fKrugGetVarIDs = (KRUG_GET_VAR_IDS)GetProcAddress(m_hDllModule,"KrugGetVarIDs");
if(fKrugGetVarIDs)
{
        DWORD dwVarCount;
                DWORD dwVarIDs[5000];
                char chNumber;

                if(fKrugGetVarIDs(m_dwConnectID,dwVarType,&dwVarCount,dwVarIDs)==S_OK)
                {
                        m_VariableNumber.ResetContent();
                        for(int i=0;i<(int)dwVarCount;i++)
                        {

                                itoa(dwVarIDs[i],&chNumber,10);
                                m_VariableNumber.AddString(&chNumber);
                        }
}
}
```

После завершения работы с библиотекой, ее нужно выгрузить, чтобы она не занимала системные ресурсы. Для этого используется функция FreeLibrary, которой следует передать дескриптор библиотеки:

```
if(m_hDllModule)
                FreeLibrary(m_hDllModule);
```

Каждый из способов имеет свои достоинства и недостатки. В случае неявного подключения все библиотеки, используемые приложением, загружаются в момент его запуска и остаются в памяти до его завершения (даже если другие запущенные приложения их не используют). Это может привести к нерациональному расходу памяти, а также заметно увеличить время загрузки приложения, если оно использует очень много различных библиотек. Кроме того, если хотя бы одна из неявно подключаемых библиотек отсутствует, работа приложения будет немедленно завершена. Явный метод лишен этих недостатков, но делает программирование менее удобным, поскольку требуется следить за своевременными вызовами LoadLibrary и соответствующими им вызовами FreeLibrary, а также получать адрес каждой функции через вызов GetProcAddress.