Модульная интегрированная

SCADA КРУГ-2000™

Версия 4.4

SDK ДРАЙВЕРОВ

Руководство Пользователя

Модульная интегрированная SCADA КРУГ-2000™. SDK драйверов. Руководство пользователя

ООО НПФ «КРУГ»

440028, г. Пенза, ул. Титова 1

Тел. +7 (8412) 49-97-75, 49-94-14

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

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

Обозначение документа: КР01.20125W-04.40-И2.1.1

СОДЕРЖАНИЕ

1 НАЗНАЧЕНИЕ

SDK предоставляет программный интерфейс для написания пользователем драйверов УСО, взаимодействующих со SCADA «КРУГ-2000».

2 СОСТАВ SDK ДЛЯ РАЗРАБОТКИ ДРАЙВЕРОВ

SDK для написания драйверов SCADA «КРУГ-2000» состоит из двух частей (рисунок 2.1):

1. Компоненты, необходимые для сборки драйвера в среде программирования C/C++.

   Данные компоненты содержат:

   - **useint.h** – заголовочный файл с описанием прототипов функций
   - **mgmirfc.lib** – файл библиотеки.

2. SDK Run-time – модуль **mgmirfc.dll,** который обеспечивает взаимодействие драйвера и Сервера ввода-вывода.

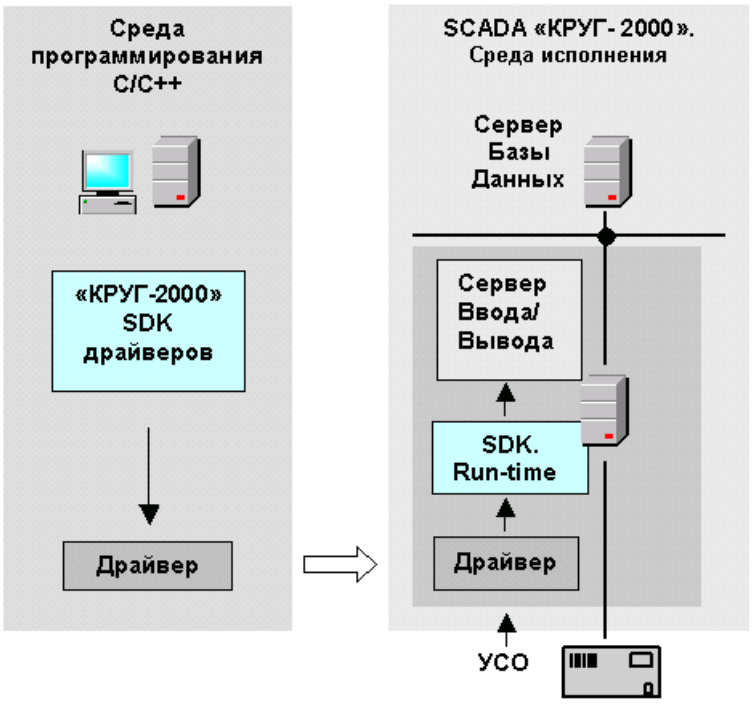

Рисунок 2.1 - SDK для разработки драйверов

3 ФУНКЦИИ ИНТЕРФЕЙСА С СЕРВЕРОМ ВВОДА-ВЫВОДА

3.1 Функции инициализации канала

3.1.1 Функция OpenChanel

Функция используется для предоставления доступа к каналу функциям чтения-записи

HANDLE OpenChanel	
Тип параметра	**Имя параметра**
USHORT	**NumberChanel**
BOOL	**Fsyn**
DWORD	**Time**

- **NumberChanel** – номер канала;
- **FSyn –** флаг, указывающий на необходимость синхронизации при работе с каналом. Если флаг установлен в TRUE, канал открывается с синхронным доступом (создается синхронизирующий объект);
- **Time** –интервал времени в мс, определяющий максимальное время ожидания освобождения синхронизирующего объекта (случай использования синхронного обмена с каналом).
- Если канал создан Сервером ввода-вывода, то функция возвратит описатель канала, иначе будет возвращено значение 0. **Максимальное число открытых каналов в одном процессе равно 255.** Только после того, как пользователь убедится, что канал открыт, он может использовать функции чтения-записи переменных.

3.1.2 Функция CloseChanel

Функция позволяет закрыть канал после окончания работы с ним.

BOOL CloseChanel	
Тип параметра	**Имя параметра**
HANDLE	**Hchanel**

- **HChanel** – описатель канала.

Если функция вызвана корректно, то она возвращает значение типа BOOL TRUE, иначе будет возвращено FALSE.

3.2 Функции чтения записи в канал

Все описанные ниже функции используют для своей работы информацию, получаемую при вызове функции OpenChanel, поэтому перед использованием этих функций необходимо открыть канал с помощью OpenChanel. Если канал был открыт с синхронным доступом, то функции чтения-записи значений будут использовать при своей работе синхронизирующий объект для синхронной записи-чтения в канал.

3.2.1 Функция WriteValueVA

BOOL WriteValueVA	
Тип переменной	**Имя переменной**
HANDLE	**Hchanel**
PVALUE_TYPE	**Value**
USHORT	**NumberValue**

Функция производит запись значения входной аналоговой переменной.

- **HChanel** – описатель канала (канал предварительно должен быть открыт функцией **OpenChanel**);
- **Value** – преобразованное или не преобразованное значение измеренной переменной;
- **NumberValue** – номер переменной;

Структура VALUE_TYPE имеет следующее описание на языке С:

```
typedef  struct _TYPE
{
  float Value;
  BOOL Flag;
  BOOL      Relativ;
  float     BeginVal;
  float     EndVal;
} VALUE_TYPE, *PVALUE_TYPE;
```

где:

- **Value** – оцифрованное значение величины или преобразованное к диапазону шкалы устройства значение;
- **Flag** – указывает на необходимость выполнения преобразования по формуле (1). Если флаг будет установлен в **TRUE** – преобразование будет выполнено, иначе преобразование выполняться не будет;
- **Relativ** – указывает имела ли место недостоверность (если это значение равно 0, то исключительной ситуации не обнаружено, иначе будет присутствовать код исключительной ситуации: 1 – обрыв в линии; 2 – значение меньше допустимого; 3 – значение больше допустимого). По данному параметру функция определяет присутствие исключительной ситуации в обработке полученных значений. Указанные исключительные ситуации должны отслеживаться драйвером устройства (переходником);
- **BeginVal** – начало шкалы измерения устройства в единицах измеряемой величины;
- **EndVal** – конец шкалы измерения устройства в единицах измеряемой величины;

BeginVal и **EndVal** можно не указывать, если **Flag** имеет значение **FALSE**.
Запись в **Relativ** должен производить драйвер устройства.

Примечание:
Функция выполняет преобразование значения **Value** к шкалам базы данных по следующей формуле и только после этого производит запись полученного результата в базу данных.

$$VALdat = НШКbd + \frac{VAL*(КШКbd - НШКbd)}{КШК - НШК} \quad (1),$$

Где:
КШК, НШК – значения **EndVal** и **BeginVal** соответственно;
КШКbd – конец шкалы (берется из паспорта переменной);
НШКbd – начало шкалы (берется из паспорта переменной);
VAL – значение **Value**.

Если по какой либо причине запись сделать не удалось, функция возвратит значение **FALSE**, иначе будет возвращено **TRUE**.

3.2.2 Функция ReadValueVA

BOOL ReadValueVA	
Тип переменной	**Имя переменной**
HANDLE	**Hchanel**
USHORT	**NumberValue**
PVALUE_TYPE	**Value**

Функция позволяет считать значение входной аналоговой переменной по номеру канала и номеру переменной. Функция возвращает значение типа BOOL, указывающее на правильность выполнения операции (в данной версии эта функция при обращении к ней будет всегда возвращать **FALSE**). Результат будет помещен по адресу, указанному в **Value**.

3.2.3 Функция WriteValueAV

BOOL WriteValueAV	
Тип переменной	**Имя переменной**
HANDLE	**Hchanel**
USHORT	**NumberValue**
PVALUE_TYPE	**Value**

Функция предназначена для чтения значения выходной аналоговой переменной из базы данных.

3.2.4 Функция ReadValueAV

BOOL ReadValueAV	
Тип переменной	**Имя переменной**
HANDLE	**Hchanel**
USHORT	**NumberValue**
PVALUE_TYPE	**Value**

Функция предназначена для чтения значения выходной аналоговой переменной из базы данных.

3.2.5 Функция WriteValueVD

BOOL WriteValueVD	
Тип переменной	**Имя переменной**
HANDLE	**Hchanel**
USHORT	**NumberValue**
PDVAL	**Value**

Функция позволяет записать в канал по номеру канала и номеру переменной значение входной дискретной переменной как синхронно, так и асинхронно. Само значение переменной передается через параметр **Value**. Функция возвратит (при успешном выполнении) значение **TRUE**, иначе будет возвращено значение **FALSE**.
Структура VDVAL имеет следующий вид:

```
typedef struct _VD
{
  BOOL Value;
  BOOL Nedost;
}DVAL, *PDVAL;
```

- **Value** – значение входной дискретной переменной;
- **Nedost** – флаг недостоверности (1 – недостоверность имеет место; 0 – достоверное значение);

3.2.6 Функция ReadValueVD

BOOL ReadValueVD	
Тип переменной	**Имя переменной**
HANDLE	**Hchanel**
USHORT	**NumberValue**
PDVAL	**Value**

Функция позволяет считать из канала по его номеру и номеру переменной значение входной дискретной переменной. Значение будет помещено по адресу **Value**. Функция возвратит (при успешном выполнении) значение **TRUE**, иначе будет возвращено значение **FALSE**.
Аналогичные функции чтения- записи предусмотрены и для выходной дискретной переменной.

3.2.7 Функция WriteValueDV

BOOL WriteValueVD	
Тип переменной	**Имя переменной**
HANDLE	**Hchanel**
USHORT	**NumberValue**
PDVAL	**Value**

Функция позволяет записать в канал по номеру канала и номеру переменной значение входной дискретной переменной как синхронно, так и асинхронно. Само значение переменной передается через параметр **Value**.

Функция возвратит (при успешном выполнении) значение **TRUE**, иначе будет возвращено значение **FALSE**.

3.2.8 Функция ReadValueDV

BOOL ReadValueVD	
Тип переменной	**Имя переменной**
HANDLE	**Hchanel**
USHORT	**NumberValue**
PDVAL	**Value**

Функция позволяет считать из канала по его номеру и номеру переменной значение входной дискретной переменной. Значение будет помещено по адресу **Value**. Функция возвратит (при успешном выполнении) значение **TRUE**, иначе будет возвращено значение **FALSE**.

3.2.9 Функция WriteValueRV

BOOL WriteValueRV	
Тип переменной	**Имя переменной**
HANDLE	**Hchanel**
USHORT	**NumberValue**
LPVOID	**PValBuf**
PBYTE	**PTypeValue**

Функция позволяет записать по номеру переменной и номеру канала значение переменной типа «ручной ввод» как синхронно, так и асинхронно.
pValBuf – указатель на буфер с данными, интерпретация которых определяется значением по адресу **pTypeValue**: 0 – тип FLOAT, 1 – тип STRING, 2 – тип BOOL.
При успешном выполнении функция возвратит значение TRUE.

3.2.10 Функция ReadValueRV

BOOL ReadValueRV	
Тип переменной	**Имя переменной**
HANDLE	**Hchanel**
USHORT	**NumberValue**
LPVOID	**PValBuf**
PBYTE	**PTypeValue**

Функция позволяет считать значение переменной типа «ручной ввод» по номеру канала и номеру переменной. Тип значения в буфере **pValBuf** будет помещен по адресу **pTypeValue**. При успешном выполнении функция возвратит значение TRUE.

3.2.11 Функция WriteRol

BOOL WriteRol	
Тип параметра	**Имя параметра**
USHORT	**Color**
LPSTR	**Message**
Int	**Sect**

Функция позволяет записать в роллинг некоторое сообщение, использующееся для диагностики.

- **Color** – код цвета, которым будет выведено сообщение;
- **Message** – указатель на строку, содержащую текст сообщения (строка должна содержать не более 80 символов);
- **Sect –** не используется и должно быть всегда 0 (зарезервировано для будущего использования).

3.2.12 Функция WriteRol (разделенный роллинг)

BOOL WriteRol	
Тип параметра	**Имя параметра**
HANDLE	**HChan**
USHORT	**Color**
LPSTR	**Message**
Int	**Sect**

Функция позволяет записать в роллинг некоторое сообщение, использующееся для диагностики. Запись сообщений производится в «разделенный роллинг», т. е. в роллинг канала.

- **hChan** – описатель канала;
- **Color** – код цвета, которым будет выведено сообщение;
- **Message** – указатель на строку, содержащую текст сообщения (строка должна содержать не более 80 символов);
- **Sect –** не используется и должно быть всегда 0 (зарезервировано для будущего использования).

3.3 Дополнительные функции

3.3.1 Функция GetConfig

VOID GetConfig	
Тип параметра	Имя параметра
HANDLE	HChan
PCHAN_CONF	PCon
UCHAR	ValType

Функция предназначена для чтения двух атрибутов переменных – номер платы, номер входа на плате. Данные атрибуты использовались для конфигурирования устройств через базу данных системы КРУГ.

Также функция полезна, если необходимо получить информацию о количестве переменных определенного типа в канале (для переменной типа PB возможно получение информации только о количестве переменных в канале).

hChan – описатель канала;
pCon – указатель на структуру, следующего вида:

```
 typedef struct _CONF
{
 PUSHORT  pNumBoard;//Указатель на массив с номерами плат для переменных
```

PUSHORT pNumEn; //Указатель на массив с номерами входов на плате для переменных
USHORT AmVal; //Количество переменных
}CHAN_CONF, *PCHAN_CONF;

- **ValType** – тип запрашиваемой переменной. Значения этого параметра: 0-для переменной типа ВА; 1-для ВД; 2-для ДВ; 3-для АВ; 4 – РВ.

Если при вызове функции в AmVal записан 0, то она вернет в этом параметре количество переменных указанного типа для данного канала.

3.3.2 Функция GetCommand

DWORD GetCommand	
Тип параметра	**Имя параметра**
HANDLE	**HChan**
LPCOMMAND	**PCmd**

Функция позволяет драйверу считывать команды управления, пришедшие в Сервер ввода-вывода. При вызове функции читается только одна команда управления.
hChan – описатель канала;
pCmd – указатель на структуру, следующего вида:

```
typedef struct _COMMAND
{
  DWORD dwTypeVar;
  DWORD dwNumVar;
} COMMAND, *LPCOMMAND;
```

dwTypeVar – тип переменной, по которой пришла команда: 2 – ДВ, 3 – АВ, 4 – РВ;
dwNumVar – номер переменной, по которой пришла команда.
При вызове функции с NULL указателем **pCmd** функция возвращает количество пришедших команд управления на данный момент, иначе возвращается количество оставшихся команд.

3.3.3 Функция CommandEnable

BOOL CommandEnable	
Тип параметра	**Имя параметра**
HANDLE	**Hchan**
BOOL	**Fflag**

Функция разрешает(запрещает) получение команд управления драйвером.
hChan – описатель канала;
fFlaf – флаг разрешения или запрещения команд – TRUE разрешает получение команд, FALSE запрещает получение команд.
Если в драйвере необходимо обрабатывать команды управления, то нужно вызвать данную функцию с флагом TRUE.

3.3.4 Функция ChanelDisconnect

BOOL ChanelDisconnect	
Тип параметра	**Имя параметра**
HANDLE	**Hchan**
BOOL	**Connect**

Если по каким-либо причинам необходимо оборвать связь Сервера ввода-вывода с сервером БД, то необходимо использовать данную функцию. Эта же функция позволяет восстановить связь Сервера ввода-вывода с сервером БД.

- **hChan –** описатель канала;
- **connect** – флаг разрешения или запрещения команд – FALSE обрывает связь Сервера ввода-вывода с сервером базы данных, TRUE - восстанавливает связь Сервера ввода-вывода с сервером базы данных.

Функция возвращает флаг состояния связи после ее вызова.

4 ПРИМЕР ИСПОЛЬЗОВАНИЯ ФУНКЦИЙ

```
HANDLE      hChan;                          //Дескриптор канала
VALUE_TYPE  ValVA;
DVAL        ValVD;
CHAN_CONF   sConf;
COMMAND     sCmd;
                                                    …
hChan = OpenChanel(NumChan, TRUE, 500); //Пытаемся открыть канал
if (!hChan)                                         //Проверяем на NULL
{
  MessageBeep(MB_OK);
  MessageBox(NULL, "Ошибка открытия канала!",
             "Ошибка", MB_ICONERROR);
  return 1;
}
                                                    …
sConf.AmVal = 0;            // Для получения количества переменных определенного типа в
                            // канале Сервера ввода-вывода необходимо вызвать функцию
                            // GetConfig с нулевым значением AmVal в
                            // структуре CHAN_CONF
GetConfig(hChan, &sChan, 0);
sConf.pNumBoard = new USHORT[sConf.AmVal];  //Выделяем память под массивы
sConf.pNumEn    = new USHORT[sConf.AmVal];  //
GetConfig(hChan, &sChan, 0);                //Повторно вызываем функцию для заполнения
                                            //массивов
DWORD NumCmd;                               //Счетчик команд
CommandEnable(hChan, TRUE);                 //Разрешаем получение команд от Сервера
                                             //Ввода/Вывода
NumCmd = GetCommand(hChan, NULL);   //Получаем количество команд на данный момент
While (NumCmd)                              //Начало цикла выборки команд
{
 GetCommand(hChan, &sCmd);                  //Считываем 1 команду и выполняем необходимые
                                            //действия и т. д.
  NumCmd--;
}

ValVD.Nedost = TRUE;
WriteValueDV(hChan, 1, &ValVD);             //Пишем значение выходной дискретной
                                            // переменной в канал Сервера ввода-вывода
sprintf(AnBuf, "Есть связь канал %d, контроллер %d", NumChan, NumContr);
CharToOem(AnBuf, RollBuf);                  //Необходимо конвертировать сообщение в
                                            //формат DOS
WriteRol(32, RollBuf, 0);                   //Запись сообщения в роллинг

ValVA.Flag = TRUE;
ValVA.BeginVal = 0.1;
ValVA.EndVal = 1;
ValVA.Value = 0.5;
if (ValVA.Value < ValVA.BeginVal) ValVA.Relativ = 2;
else
{
```

```
    if (ValVA.Value > ValVA.EndVal) ValVA.Relativ = 3;
    else ValVA.Relativ = 0;
}
WriteValueVA(hChan, &ValVA, 1);      //Пишем в канал Сервера ввода-вывода значение
                                     //входной аналоговой переменной
CloseChanel(hChan);                  //Закрываем канал Сервера ввода-вывода
                                     //при завершении работы с ним
```