

Модульная интегрированная

SCADA КРУГ-2000[™]

**КРУГОЛ[™] . ИНТЕГРИРОВАННАЯ
СРЕДА РАЗРАБОТКИ**

Версия 2.6

Руководство Пользователя

Редакция 1

Модульная интегрированная SCADA КРУГ-2000™. КРУГОЛ™. Интегрированная среда разработки. Руководство Пользователя/1-е изд.

© 1992-2013. ООО НПФ «КРУГ», ООО «КРУГ-Софт». Все права защищены.

Никакая часть настоящего издания ни в каких целях не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами, будь то электронные или механические, включая фотографирование, магнитную запись или иные средства копирования или сохранения информации, без письменного разрешения владельцев авторских прав.

Все упомянутые в данном издании товарные знаки и зарегистрированные товарные знаки принадлежат своим законным владельцам.

ООО НПФ «КРУГ», ООО «КРУГ-Софт»

440028, г. Пенза, ул. Титова 1

Тел. +7 (8412) 49-97-75, 55-64-97, 49-94-14, 48-34-80

Факс: +7 (8412) 55-64-96

E-mail: support@krug2000.ru

http:// www.krug2000.ru

http:// www.krugsoft.ru

СОДЕРЖАНИЕ

Стр.

СЛОВАРЬ ТЕРМИНОВ И СОКРАЩЕНИЙ	1
ВВЕДЕНИЕ	3
В.1 Указания по инсталляции ИСР КРУГОЛ	4
В.2 Порядок инсталляции	4
1 ОСНОВНЫЕ ПРИНЦИПЫ ПРОГРАММИРОВАНИЯ В СРЕДЕ РАЗРАБОТКИ	1-1
1.1 Задачи, решаемые на уровне станции оператора	1-1
1.2 Задачи, решаемые на уровне контролеров	1-2
1.3 Версии СРВК и среды разработки КРУГОЛ	1-2
1.4 Режимы работы ИСР КРУГОЛ	1-2
2 ГЛАВНОЕ ОКНО И КОМПОНЕНТЫ СРЕДЫ РАЗРАБОТКИ	2-1
2.1 Настройка панелей инструментов и групп команд	2-2
2.2 Настройка "горячих" клавиш ИСР КРУГОЛ	2-3
3 ЯЗЫК СТРУКТУРИРОВАННОГО ТЕКСТА	3-1
3.1 Операнды	3-1
3.1.1 Типы данных	3-1
3.1.2 Константные значения	3-1
3.1.3 Предопределенные операнды	3-2
3.1.4 Операнды, определяемые пользователем	3-4
3.2 Операции и операторы	3-7
3.2.1 Операции	3-7
3.2.2 Оператор присваивания	3-9
3.2.3 Операторы включения/выключения таймеров	3-9
3.2.4 Условный оператор	3-10
3.2.5 Оператор цикла	3-12
3.2.6 Оператор досрочного завершения	3-13
3.3 Комментарии	3-13
3.4 Программа	3-13
3.5 Процедура	3-14
3.6 Функции	3-15
3.7 «Перегрузка» вызова функций	3-18
3.8 Алгоблок	3-19
3.9 Автоматическая нумерация алгоблоков	3-20
3.10 Примеры программ на языке СТ	3-21
3.11 Формальное описание грамматики языка СТ	3-25
4 ЯЗЫК ФУНКЦИОНАЛЬНЫХ БЛОЧНЫХ ДИАГРАММ	4-1
4.1 Общие сведения	4-1
4.2 Элементы языка	4-2
4.2.1 Константа	4-2
4.2.2 Выражение СТ	4-2
4.2.3 Переменная	4-2
4.2.4 Массив данных	4-3

**СОДЕРЖАНИЕ**

	Стр.
4.2.5 Переменная массива данных _____	4-3
4.2.6 ЕСЛИ _____	4-3
4.2.7 ДЛЯ _____	4-4
4.2.8 Прервать _____	4-4
4.2.9 Функция _____	4-4
4.2.10 Процедура _____	4-5
4.2.11 Выход _____	4-5
4.2.12 Порядок _____	4-5
4.2.13 Сообщение _____	4-6
4.2.14 Комментарий _____	4-6
4.2.15 Линия связи _____	4-7
5 ПРОЕКТ _____	5-1
5.1 Архитектура проекта _____	5-1
5.2 Окно проекта _____	5-1
5.3 Работа с проектом _____	5-4
5.3.1 Создание, открытие и сохранение проекта _____	5-4
5.3.2 Создание, открытие и сохранение файлов проекта _____	5-8
5.3.3 Создание пользовательских данных _____	5-10
5.3.4 Печать исходных текстов программ _____	5-11
5.3.5 Настройка параметров печати _____	5-12
5.3.6 Формирование списка файлов для печати _____	5-12
5.3.7 Настройка нумерации страниц при печати _____	5-13
5.4 Редактирование файлов проекта _____	5-16
5.4.1 Редактирование файлов СТ _____	5-16
5.4.2 Редактирование файлов ФБД _____	5-20
5.5 Трансляция проекта _____	5-40
5.5.1 Транслятор языка КРУГОЛ _____	5-40
5.5.2 Сообщения транслятора об ошибках _____	5-41
5.6 Отладка проекта _____	5-42
5.6.1 Общие сведения _____	5-42
5.6.2 Окно проекта в режиме отладки _____	5-42
5.6.3 Режимы отладки _____	5-44
5.6.4 Изменение значений переменных _____	5-45
5.6.5 Связь с БД _____	5-45
5.6.6 Удаленная отладка _____	5-46
5.6.7 Вывод сообщений _____	5-47
5.6.8 Окно переменных _____	5-48
5.6.9 Автоматическое окно переменных _____	5-49
5.7 Программирование контроллера _____	5-49
5.8 Поиск задействованных объектов в проекте _____	5-50
5.8.1 Настройка критериев поиска _____	5-51

СОДЕРЖАНИЕ

	Стр.
5.8.2 Выполнение поиска задействованных объектов _____	5-53
5.9 Редактирование базы данных контроллера _____	5-54
5.9.1 Редактор базы данных контроллера _____	5-54
5.9.2 Создание БД контроллера _____	5-55
5.9.3 Открытие БД контроллера _____	5-55
5.9.4 Сохранение БД контроллера. _____	5-56
5.9.5 Редактирование БД контроллера. _____	5-56
5.9.6 Атрибуты переменных БД _____	5-57
5.10 Конфигурирование трендов _____	5-81
5.10.1 Модуль XML-описания тренда _____	5-81
5.10.2 Общие настройки _____	5-83
5.10.3 Настройка самописцев _____	5-83
6 БИБЛИТЕКАРЬ ЯЗЫКА КРУГОЛ _____	6-1
6.1 Назначение _____	6-1
6.2 Интерфейс пользователя _____	6-1
6.2.1 Главное окно _____	6-1
6.2.2 Меню _____	6-2
6.3 Панель инструментов _____	6-9
6.3.1 "Библиотеки" _____	6-9
6.3.2 "Файлы" _____	6-13
6.3.3 "Привязки" _____	6-14
6.4 Пример создания библиотеки _____	6-16
6.4.1 Порядок создания библиотеки _____	6-16
6.4.2 Реализация функции на языке C/C++ _____	6-16
6.4.3 Создание описания библиотеки _____	6-16
6.4.4 Создание описания функции _____	6-18
6.4.5 Использование функции в технологической программе _____	6-18
7 ВКЛЮЧЕНИЕ ПРОГРАММ ПОЛЬЗОВАТЕЛЯ В СИСТЕМУ РЕАЛЬНОГО ВРЕМЕНИ _____	7-1
7.1 Система реального времени контроллера _____	7-1
7.2 Среда исполнения станции оператора _____	7-2

СЛОВАРЬ ТЕРМИНОВ И СОКРАЩЕНИЙ

АВ	–	Аналоговая выходная
ВА	–	Входная аналоговая
ВД	–	Входная дискретная
ГБД	–	Генератор базы данных
ГД	–	Генератор динамики
ДВ	–	Дискретная выходная
ИСП	–	Интегрированная среда разработки
ПВ	–	Промежуточная вещественная
ПЛ	–	Промежуточная логическая
ПрП	–	Программа Пользователя
ПТК	–	Программно – технический комплекс
ПЦ	–	Промежуточная целая
РВ	–	Ручной ввод
СО	–	Станция оператора
СРВ		Система реального времени
СРВК	–	Система реального времени контроллера
СТ	–	Структурированный текст
ТМ	–	Таймер минутный
ТС	–	Таймер секундный
ТЧ	–	Таймер часовой
УСО	–	Устройство связи с объектом (контроллер)
ФБД	–	Функционально – блочная диаграмма

ВВЕДЕНИЕ

Энергетика, нефтепереработка, химическое производство, а также другие отрасли предъявляют свои требования к программной реализации задач, регламентируемых технологией процесса производства.

Прикладные задачи в рамках конкретной автоматизированной системы управления также могут иметь весьма специфичный характер.

Интегрированная среда разработки (ИСР) КРУГОЛ™ – это набор инструментальных средств автоматизации программирования, позволяющих в полном объеме реализовать задачи практически любого уровня сложности и специфики конкретного технологического процесса.

Таковыми задачами могут быть:

- программно-логическое управление технологическим оборудованием
- алгоритмы рационального управления
- расчет косвенных переменных по формулам
- визуализация значений в цифровом виде (трендов целевой обработки) – текущие, средние или суммарные значения параметров по часам, сменам и суткам
- формирование трендов целевой обработки из программы Пользователя (ПрП) постфактум
- архивирование дат и времени событий
- создание сценария динамики видеокадра
- интегрирование мгновенных расходов для задач дозирования
- создание альтернативных фильтров входных параметров
- и другие.

ИСР КРУГОЛ отвечает требованиям IEC 61131-3 к разработке технологических программ и объединяет в своем составе компоненты разработки на языках структурированного текста (СТ) и функциональных блочных диаграмм (ФБД).

Язык СТ – процедурно-ориентированный язык программирования с несложным русифицированным синтаксисом. Язык СТ позволяет Вам быстро овладеть правилами программирования и освобождает от задач распределения памяти под переменные, используемые в программе.

Язык структурированного текста реализует основные управляющие структуры (Последовательность, Ветвление, Цикл, Составные структуры, Структуры прерывания выполнения блока программы), а также обеспечивает разработку и выполнения программ с вложенными процедурами и функциями.

Язык функциональных блочных диаграмм – графический язык. Элементами языка ФБД являются графические символы, которые используются для создания схемы ФБД.

Язык ФБД позволяет разработчику строить сложные процедуры, используя существующие функции из поставляемой библиотеки, и связывать их с другими элементами ФБД.

Созданные в Интегрированной среде программы Пользователя позволяют осуществить простой доступ к базе данных реального времени SCADA КРУГ-2000 (чтение, запись текущих значений переменных и их целевых трендов), создавать архивы данных на жестком диске, выполнять алгоритмы автоматического регулирования и коммерческого учета, а также множество других функций.

Библиотека функций КРУГОЛ насчитывает более 200 функций, используемых для решения разнообразных задач. Подробное описание функций приведено в книге «Модульная интегрированная SCADA КРУГ-2000. КРУГОЛ™. Библиотека функций».

В.1 Указания по инсталляции ИСР КРУГОЛ

- Интегрированная среда разработки КРУГОЛ (**ИСР КРУГОЛ**) поставляется как вместе со SCADA КРУГ-2000, так и в виде отдельного программного продукта
- ИСР КРУГОЛ для полнофункциональной работы требует **электронный ключ**
- Для инсталляции ИСР Пользователь должен иметь права администратора
- Перед установкой ИСР закройте все работающие программы
- Если Вы устанавливали предыдущую версию ИСР (для версии 2.1 и ниже), то удалите ее
- Если в процессе установки Вы хотите прервать инсталляцию, нажмите кнопку «**Отмена**». В появившемся окне инсталлятор спросит Вас, действительно ли Вы хотите прервать установку. Вы можете нажать кнопку «**Да**» и выйти из инсталлятора или нажать кнопку «**Нет**» и вернуться к процессу инсталляции
- В окнах инсталлятора может присутствовать кнопка «<<**Назад**», при нажатии на которую Вы можете вернуться в предыдущее окно. Данная возможность может пригодиться, если Вы захотите изменить настройки, выбранные в предыдущих окнах.

В.2 Порядок инсталляции

Шаг 1. Начало инсталляции



ВНИМАНИЕ!!!

Для корректной работы мастера установки необходимо, чтобы на компьютере был установлен пакет Microsoft Windows installer 3.0 и выше. Для работы под Windows 7 необходимо чтобы на компьютере было установлено следующее обновление операционной системы: Windows6.0-KB917607-x86 или Windows6.0-KB917607-x64 для 32-х и 64-х разрядной редакции системы.

При установке продукта на SCADA КРУГ-2000 4.1 и ниже под управлением Windows 7 необходимо отключить контроль учётных записей (User Account Control)

Для установки интегрированной среды разработки КРУГОЛ запустите на выполнение инсталлятор (рисунок В.1) и следуйте его указаниям.

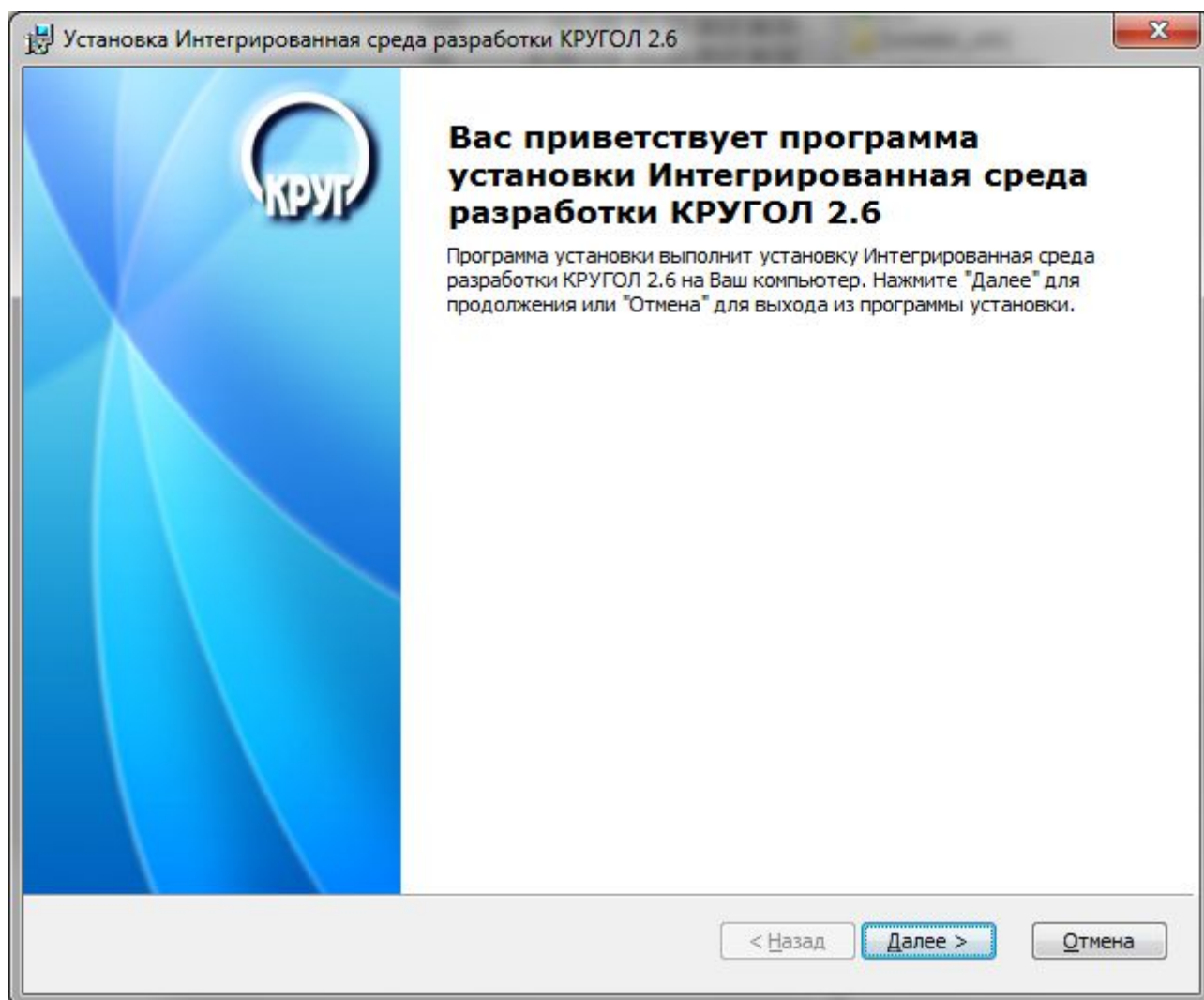


Рисунок В.1 - Окно инсталлятора ИСР КРУГОЛ

Шаг 2. Лицензионное соглашение

Для продолжения установки внимательно прочитайте лицензионное соглашение. Если Вы согласны с условиями соглашения, и нажмите кнопку «**Я согласен**» (рисунок В.2).

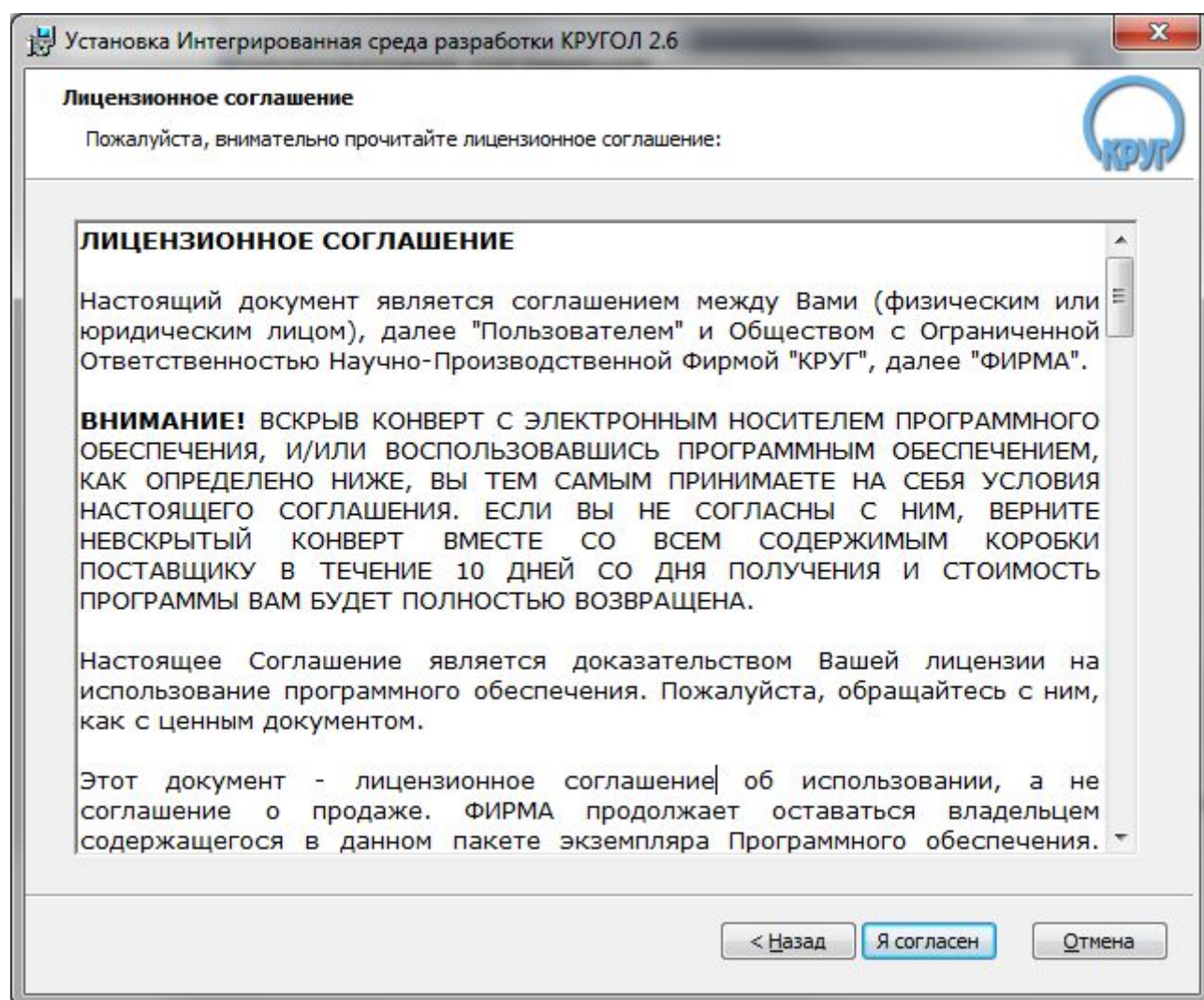


Рисунок В.2 – Окно «Лицензионное соглашение»

Шаг 3. Выбор пути установки и устанавливаемых компонентов

В окне «**Выбор пути установки**» указаны путь установки по умолчанию (рисунок В.3).

 **ВНИМАНИЕ!!!**

Если установка ИСР производится на компьютер, где имеется SCADA КРУГ-2000, то путь, куда будет устанавливаться ИСР, изменить нельзя.

Если же SCADA КРУГ-2000 не установлена, то Вы можете изменить путь установки ИСР.

Для продолжения нажмите на кнопку «**Далее>>**».

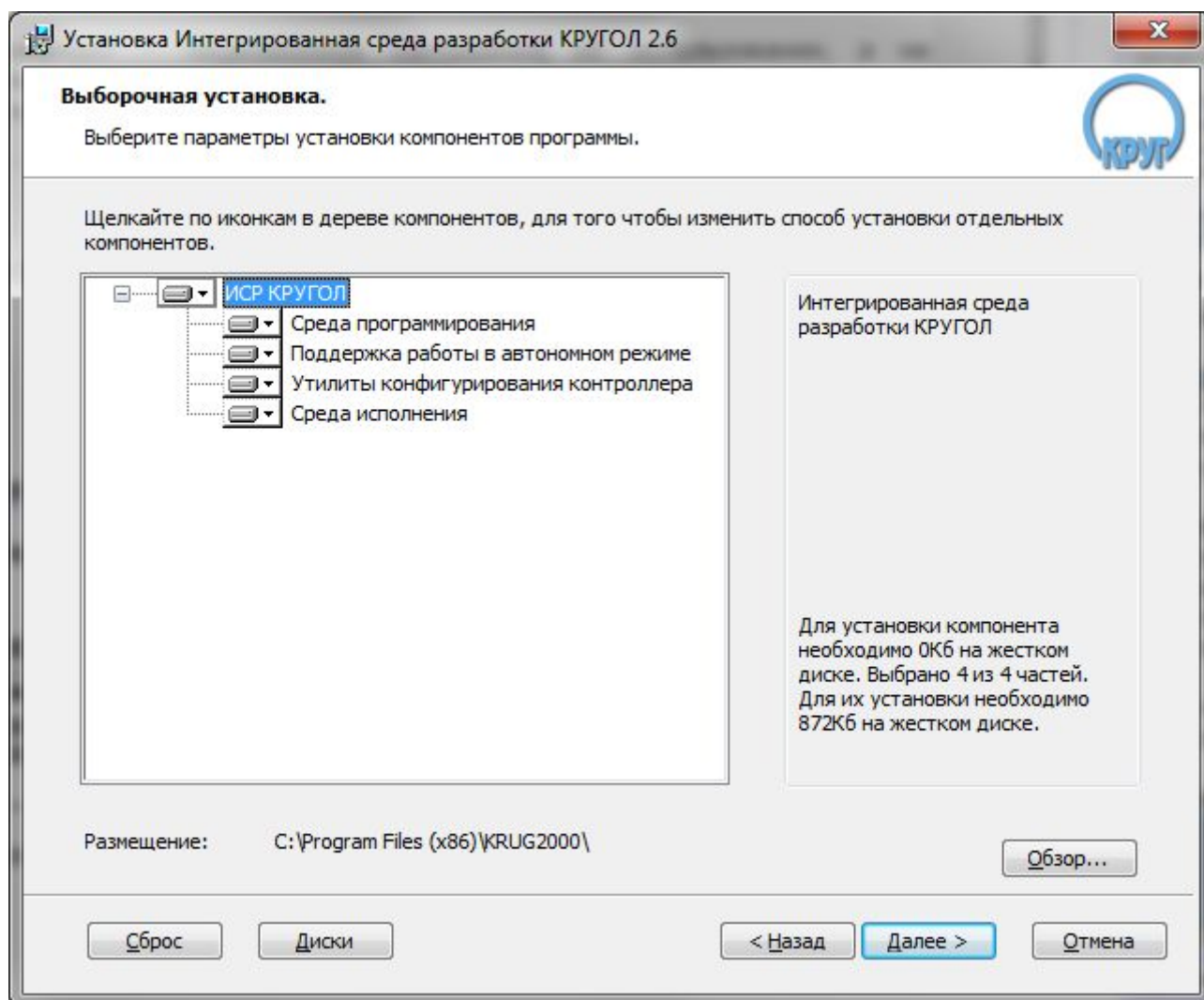


Рисунок В.3 - Окно выбора пути установки

При выборе одного из компонентов в правой части окна отображается его описание.

Описание и назначение устанавливаемых компонентов ИСП приведено в таблице В.1.

Наименование компонента	Описание
Среда программирования	Набор компонентов, обеспечивающих создание редактирование и компиляцию отладку программ на языках ФБД и СТ
Поддержка работы в автономном режиме	Набор компонентов, обеспечивающий корректную работу ИСП в автономном режиме (когда на компьютере не установлена SCADA КРУГ-2000)
Среда исполнения	Среда исполнения программ КРУГОЛ
Утилиты конфигурирования контроллера	Набор компонентов, обеспечивающих возможность редактирования БД контроллера и трендов

 ВНИМАНИЕ!!!

Набор компонент необходимых, для работы ИСП в зависимости от наличия или отсутствия SCADA может изменяться. В случае, если пользователь выберет недопустимое сочетание устанавливаемых компонент, инсталлятор сообщит о некорректности действий пользователя.

Шаг 4. Подтверждение готовности к установке

После выбора компонент для продолжения нажмите на кнопку «Далее>>» и следуйте указаниям инсталлятора (рисунок В.4).

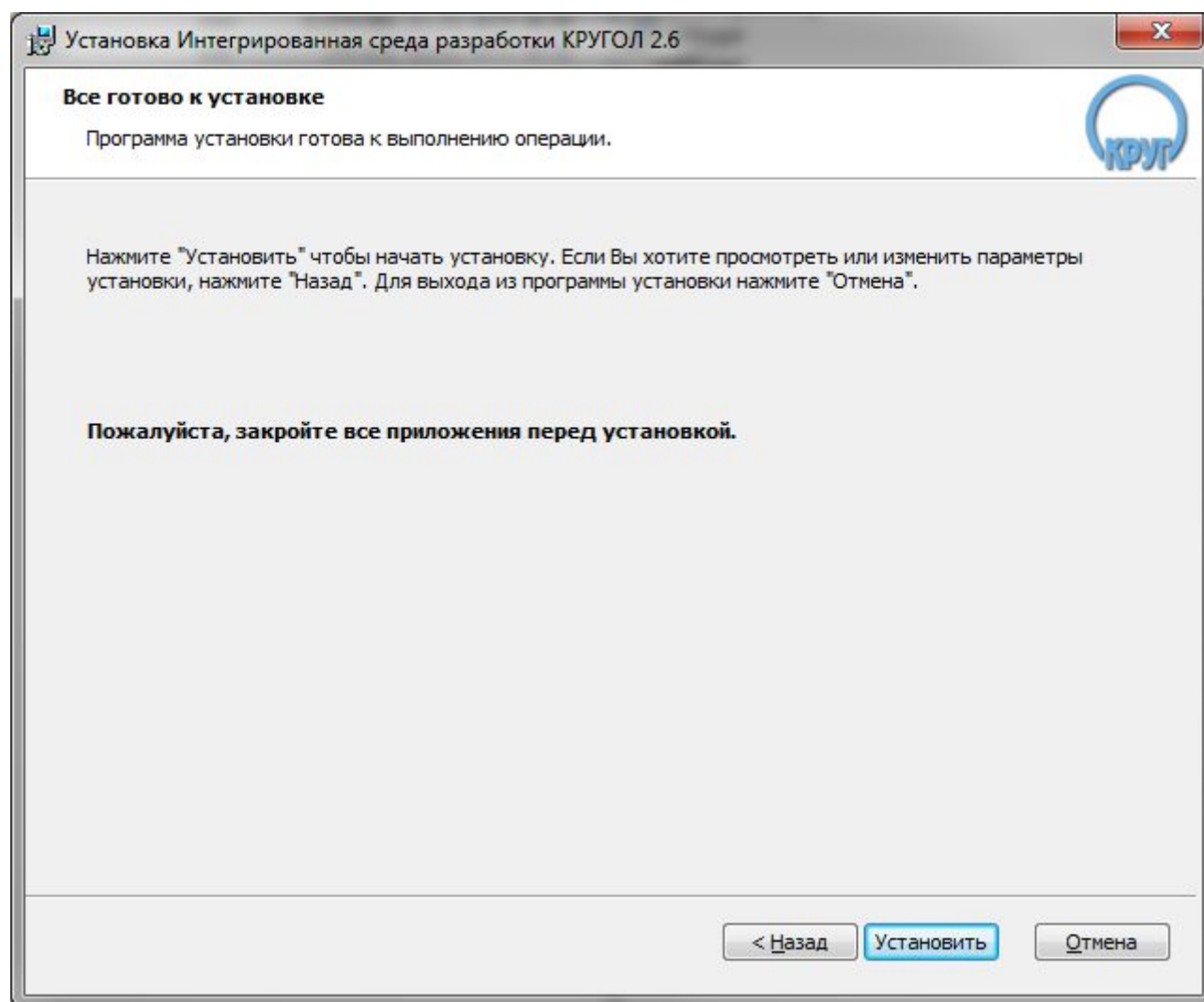


Рисунок В.4 – Всё готово для установки приложения

Шаг 7. Завершение установки

Нажмите на кнопку «Готово» (рисунок В.5).

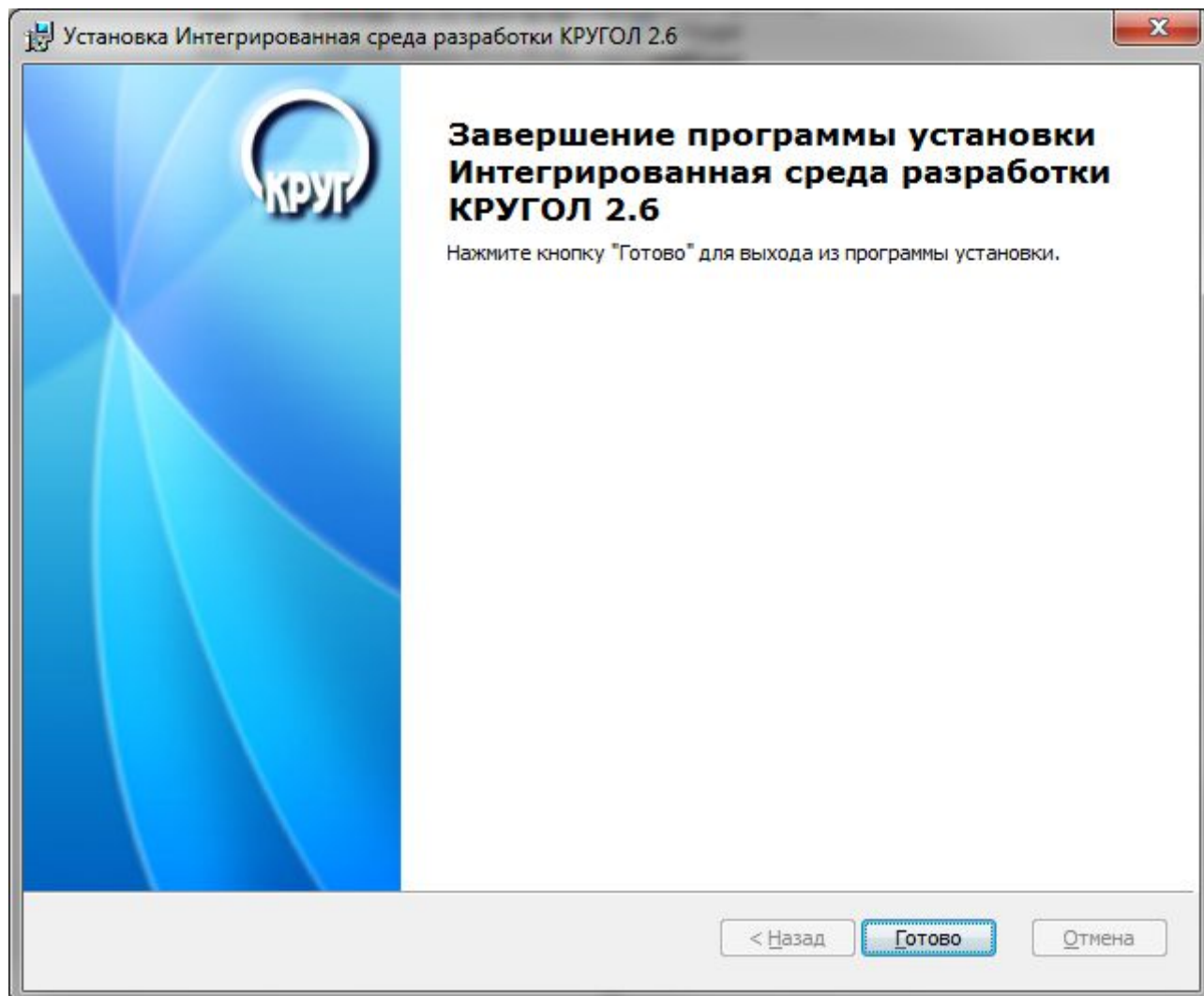


Рисунок В.5 – «Установка ИСР КРУГОЛ версии 2.6 завершена»

Установка ИСР КРУГОЛ завершена.

1 ОСНОВНЫЕ ПРИНЦИПЫ ПРОГРАММИРОВАНИЯ В СРЕДЕ РАЗРАБОТКИ

Главные принципы программирования в ИСР КРУГОЛ™ заключаются в следующем:

- В одну среду разработки объединены средства для программирования на языках ФБД и СТ
- Программы КРУГОЛ могут использовать общие исходные тексты СТ и схемы ФБД как в одном проекте, так и в одной программе
- Отладка программ, как для станций верхнего уровня, так и для контроллеров в одной среде
- Расширенные функциональные возможности ФБД, в частности добавлены удобные блоки условия и цикла
- Возможность легко включать функции Пользователя (языки C/C++/Delphi/...) в библиотеку функций КРУГОЛ для их выполнения на различных платформах (Windows, Linux, QNX) – компонент «Библиотекарь КРУГОЛ»
- Возможность разработки проектов как для станций оператора, так и для контроллеров
- Возможность разработки проектов для различных платформ (Windows, Linux, QNX).

Разработчику программ КРУГОЛ рекомендуется:

- Использовать модульный подход при реализации различных алгоритмов
- Структурировать исходные тексты программ СТ и схем ФБД для удобства их понимания и сопровождения
- Широко использовать в комментариях к программам соответствующую терминологию технологических процессов и оборудования.

Основными платформами, на базе которых функционируют программы КРУГОЛ, являются на нижнем уровне (контроллеры) – **Linux, QNX**; на верхнем уровне (станции оператора) – **Windows**.

1.1 Задачи, решаемые на уровне станции оператора

На станции оператора, как правило, выполняются следующие функции, реализованные на языке ИСР КРУГОЛ:

- Формирование и печать печатных документов, протоколов до- и послеаварийных ситуаций, режимных листов
- Формирование точек трендов параметров постфактум из программы Пользователя
- Статистическая обработка показателей и параметров процесса, путем чтения значений из оперативных и исторических трендов
- Имитация параметров процесса
- Управление обменом данными между абонентами ПТК.

1.2 Задачи, решаемые на уровне контролеров

Технологический контроллер является "исполнителем" самых ответственных задач в системах автоматизированного управления технологическими процессами.

На базе контроллера (СРБК), как правило, выполняются следующие алгоритмы, реализованные на языках ИСР КРУГОЛ:

- Противоаварийные защиты и блокировки
- Управление задвижками с электроприводом
- Управление отсечными кранами
- Управление насосами
- Управление электрооборудованием
- Управление агрегатами в обвязке
- Реализация расчетных задач
- Учет наработки оборудования
- Технический и коммерческий учет энергоносителей
- Коммерческий и технический учет тепловой энергии.

1.3 Версии СРБК и среды разработки КРУГОЛ

Новая функциональность, расширение базовых типов данных и другие изменения, как правило, отражаются в новых версиях языка КРУГОЛ и соответственно включаются в новые версии систем реального времени контроллеров.

Поэтому в текст руководства добавлены примечания, которые конкретизируют использование элементов языка, зависящих от версий среды исполнения КРУГОЛ и СРБК. Например:

«...Использование глобальных переменных не допускается для платформ среды исполнения КРУГОЛ версии 1.0 и СРБК версии 6.5 ...»

1.4 Режимы работы ИСР КРУГОЛ

ИСР КРУГОЛ может функционировать в 2-х режимах:

- **Режим взаимодействия со SCADA КРУГ-2000**
- **Автономный режим.**

Указанные режимы активируются автоматически на этапе инсталляции ИСР: если на целевом компьютере установлена SCADA КРУГ-2000 версии 4.0 или выше, то ИСР инсталлируется для работы в режиме взаимодействия со SCADA, в противном случае ИСР КРУГОЛ инсталлируется для работы в автономном режиме.

При работе в режиме взаимодействия со SCADA в ИСР доступна трансляция программ как для платформ верхнего уровня – Среда исполнения КРУГОЛ версий 1.0,2.0,2.1,2.2 и выше, так и для контроллерных платформ.



ВНИМАНИЕ!!!

В автономном режиме ИСР не поддерживает работу с проектами, созданными для платформ верхнего уровня.

Для модификации базы данных контроллера в этом режиме используется функция редактирования базы данных ИСР КРУГОЛ.

2 ГЛАВНОЕ ОКНО И КОМПОНЕНТЫ СРЕДЫ РАЗРАБОТКИ

Главное окно интегрированной среды разработки КРУГОЛ показано на рисунке 2.1.

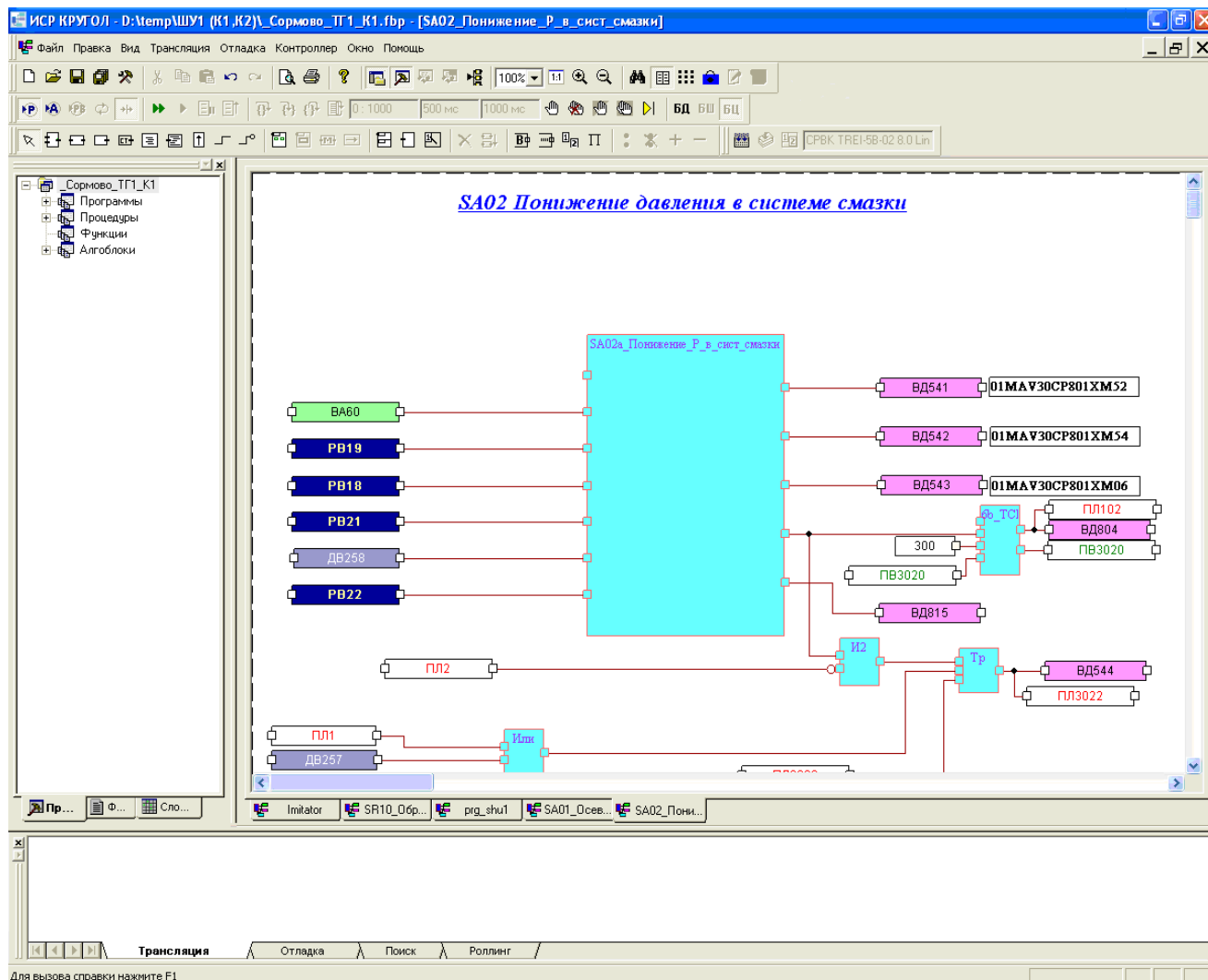


Рисунок 2.1 – Главное окно интегрированной среды КРУГОЛ

В верхней части окна ИСР расположены меню, панели инструментов, под которыми располагаются окна программ пользователя.

Слева располагается окно проекта со списком программ пользователя. Окно проекта используется для удобства работы пользователя с создаваемыми проектами КРУГОЛ.

В нижней части располагается окно вывода информационных сообщений. Окно содержит четыре вкладки: «Трансляция» – сообщения среды разработки о результатах трансляции проекта; «Отладка» – сообщения отладчика; «Поиск» – результаты поиска Пользователем информации в проекте в процессе его редактирования; «Роллинг» - отображение протокола событий (только для контроллера с CPBK версии 8.0 в режиме удаленной отладки), а также всех сообщений функций message, message2, message3 и элемента «Сообщение», выдаваемых в процессе локальной отладки проекта для контроллера или станции оператора. Управлять показом/скрытием окна вывода можно при помощи команды меню «Вид\Окно вывода».

Интегрированная среда разработки КРУГОЛ состоит из нескольких компонент:

- Редактор структурированного текста
- Редактор языка функционально-блочных диаграмм
- Транслятор программ Пользователя
- Отладчик программ Пользователя.

Редактор СТ предназначен для разработки программ пользователя на языке СТ. Редактор предоставляет пользователю набор различных сервисных функций: подсветка синтаксических конструкций СТ, поиск заданного фрагмента текста и другие.

Редактор ФБД предназначен для создания и редактирования схем ФБД. В процессе редактирования схемы редактор ФБД создает базу данных программы пользователя, используемую в дальнейшем транслятором. Редактор также предоставляет Пользователю набор различных сервисных функций: добавление, выделение, удаление элементов, задание свойств элементам и другие.

Транслятор осуществляет лексический и синтаксический анализ исходных программ Пользователя (СТ и ФБД). Результатом работы транслятора являются исполняемые модули – файлы с расширением **.out** – предназначенные для запуска в контроллере или станции оператора.

Отладчик предназначен для отладки программ Пользователя. Отладка может осуществляться в циклическом и пошаговом режимах. По результатам отладки программ формируется отчет.

Внешний вид среды разработки может быть настроен в соответствии с потребностями Пользователя.

2.1 Настройка панелей инструментов и групп команд

Пользовательская настройка производится при помощи команды **«Настройка»** меню **«Вид»**. При выборе команды на экран выводится диалоговое окно **«Настроить»**: (рисунок 2.2).

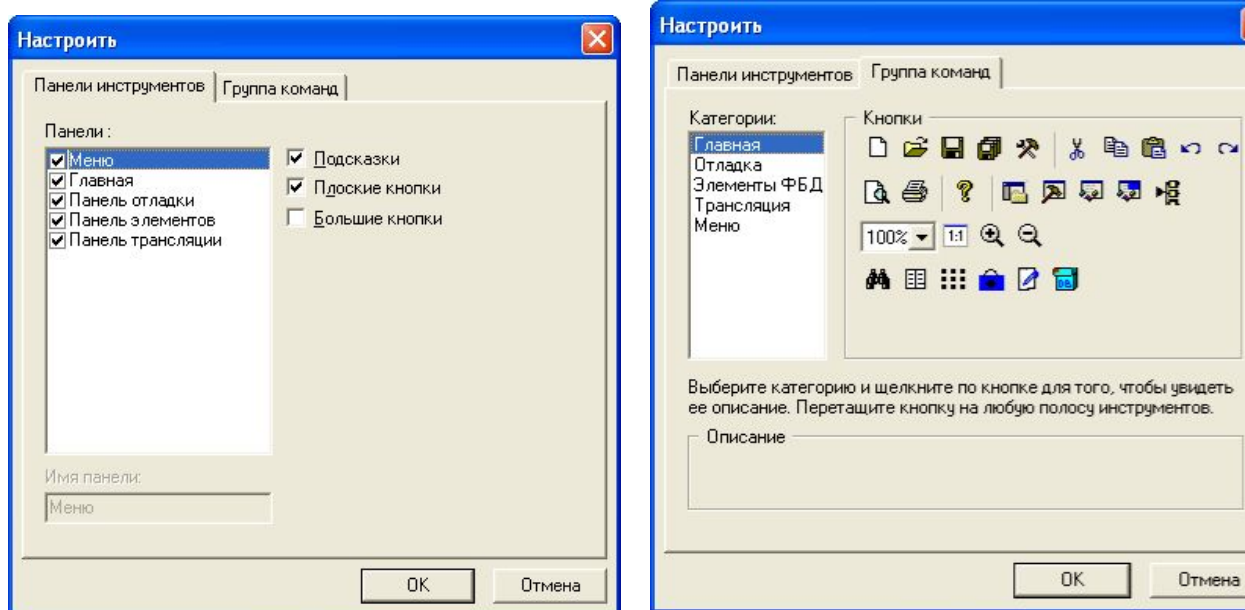


Рисунок 2.2 – Окно «Настроить». Закладки «Панели инструментов» и «Группа команд»

2.2 Настройка "горячих" клавиш ИСР КРУГОЛ

Наиболее часто выполняемые действия в процессе разработки программ КРУГОЛ можно инициировать при помощи сочетаний клавиш. Назначить сочетания клавиш можно командой «Клавиатура» меню «Вид». Диалоговое окно "Настройка клавиатуры" приведено на рисунке 2.3. Допускается назначение для каждой команды нескольких различных сочетаний клавиш.

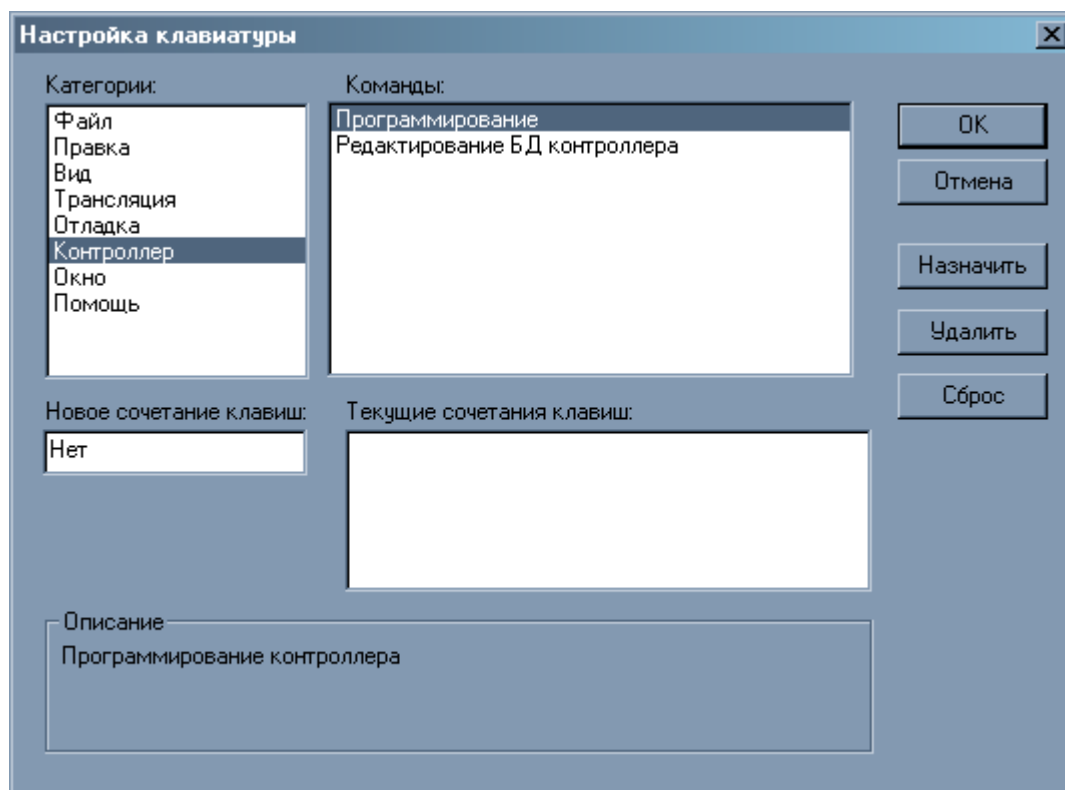


Рисунок 2.3 – Окно настройки сочетаний клавиш

3 ЯЗЫК СТРУКТУРИРОВАННОГО ТЕКСТА

3.1 Операнды

Операнды являются основными объектами, над которыми выполняются действия при программировании на технологическом языке КРУГОЛ. К операндам относятся константы, переменные, массивы, а также вызовы функций, возвращающих одно значение.

3.1.1 Типы данных

Тип данных – основная характеристика операнда. Тип данных определяет, какие значения может принимать операнд и какие действия могут быть выполнены с его использованием. В технологическом языке КРУГОЛ допускается использование операндов следующих типов:

- Логический. Принимает значения 0 (ложь) и 1 (истина)
- Целочисленный 8-битовый. Принимает значения в диапазоне от -128 до 127*.
- Целочисленный 16-битовый. Принимает значения в диапазоне от -32768 до 32767.
- Целочисленный 32-битовый. Принимает значения в диапазоне от -2147483647 до 2147483647*.
- Вещественный 32-битовый. Принимает положительные и отрицательные значения в диапазоне от 1.175494351e-38 до 3.402823466e+38.
- Вещественный 64-битовый. Принимает положительные и отрицательные значения в диапазоне от 1.7976931348623158e+308 до 2.2250738585072014e-308*.
- Строка. Последовательность символов (длина строки ≤ 79)
- Неопределенный тип данных. Тип определяется на этапе трансляции. Может использоваться только как выходной тип СТ-выражения. Указанный тип данных может применяться для платформ СРВК, начиная с версии 8.0, и среды исполнения КРУГОЛ, начиная с версии 2.2.

Если значение константы, записанное в целочисленном формате, выходит за допустимые пределы, то оно автоматически преобразуется транслятором к вещественному типу.

Действия над операндами выполняются с помощью операций, соответствующих типу операндов.

3.1.2 Константные значения

Для непосредственного задания константных значений в программе пользователя используются следующие правила.

- Истинное значение логического типа представляется единицей, ложное - нулем.
- Целочисленное значение представляется числом в десятичной системе счисления
- Значение вещественного типа представлено числом с плавающей точкой
- Строковая константа – последовательность символов в тексте программы – записывается в виде "**строка символов**"

Ниже приведены несколько примеров записи констант

Логическая константа	1	0	
Целочисленная константа	0	18	-4
Вещественная константа	0.05	-84.39	1.2e3
Строковая константа	"строка символов"		

3.1.3 Предопределенные операнды

Для непосредственного задания константных значений в программе пользователя в языке КРУГОЛ используются два подмножества предопределенных операндов – зарезервированных переменных:

- **системные переменные** – значениями этих переменных являются значения данных базы данных реального времени КРУГ-2000
- **внутренние переменные** – значениями этих переменных являются результаты промежуточных вычислений и операций с таймерами программы Пользователя.

3.1.3.1 Системные переменные

Системные переменные предназначены для обмена информацией программы Пользователя с базой данных (БД) КРУГ-2000.

База данных КРУГ-2000 содержит следующие типы переменных:

- **ва** – входные аналоговые переменные
- **ав** – выходные аналоговые переменные
- **вд** – входные дискретные переменные
- **дв** – выходные дискретные переменные
- **рв** – переменные ручного ввода.

Тип значений переменных «ва», «ав» и «рв» соответствует 32-битовому вещественному типу, тип значений переменных «вд» и «дв» – логическому типу.

Для обращения к значению системной переменной в тексте программы на языке СТ используется синтаксис:

<тип переменной БД><номер переменной>

или

<тип переменной БД>[<переменная оператора цикла>]

Последний вариант соответствует косвенному обращению к переменной, номер которой задается переменной оператора последовательности.

Кроме текущего значения системные переменные характеризуются значениями **паспорта** – специального набора атрибутов. Для обращения к значениям атрибутов системных переменных используется следующий синтаксис

<системная переменная>.а<номер атрибута>

Примеры записи системных переменных и их атрибутов: **вд1 дв4 рв24 ва3.а12 ав2.а22**

В качестве примера косвенного обращения к системным переменным можно привести следующий фрагмент программы:

```
Для i[1...10]
{
    дв[i] = 0
}
```


3.1.3.2 Внутренние переменные

Внутренние переменные можно разделить на две группы.

Первая группа – **промежуточные переменные** – предназначена для сохранения промежуточных результатов вычислений. Переменные данной группы являются глобальными для всех программ, выполняемых средой исполнения. Существуют следующие типы промежуточных переменных:

- **пл** – переменные логического типа
- **пц** – переменные целочисленного 16-битового типа
- **пв** – переменные вещественного 32-битового типа.

Вторая группа – **таймерные переменные** – объединяет переменные, предназначенные для выполнения операций над таймерами и хранения их текущих значений. Существуют следующие типы таймерных переменных:

- **тс** – переменные секундных таймеров
- **тм** – переменные минутных таймеров
- **тч** – переменные часовых таймеров.

Все значения таймерных переменных соответствуют вещественному 32-битовому типу.

Обращение к значениям внутренних переменных происходит аналогично обращению к значениям системных переменных. При программировании на языке СТ используется синтаксис:

<тип переменной><номер переменной>

или

<тип переменной>[<переменная оператора цикла>]

Последний вариант соответствует косвенному обращению к переменной, номер которой задается переменной оператора последовательности.

Примеры записи внутренних переменных: **пл1 пв4 пц24 тс12**

Примеры форматов представления времени:

- **тс1** = 15.50 – значение таймера № 1 равно 15.5 сек.
- **тм3** = 15.50 – значение таймера № 3 (15 с половиной минут) равно 15 мин.30 сек.
- **тч4** = 15.75 – значение таймера № 4 (15 и $\frac{3}{4}$ часа) равно 15 час.45 мин.

Количество внутренних переменных в программе Пользователя:

- **пв, пц, пл** – с 1 по 9999
- **тс, тм, тч** – с 1 по 1024.

В качестве примера косвенного обращения к внутренним переменным можно привести следующий фрагмент программы

```
Для i[1...10]
{
    пв[i] = 0
}
```

3.1.4 Операнды, определяемые пользователем

При программировании на технологическом языке КРУГОЛ пользователю предоставляется возможность объявлять собственные константы, переменные и массивы, используя для них удобные имена в контексте решаемой задачи.

3.1.4.1 Константы

Константы предназначены для удобства использования константных значений и повышения читаемости текста программы.

Обращение к значению константы в тексте программы происходит по ее имени.

Константы являются операндами доступными только для чтения.

Константы могут быть как локальными, так и глобальными.

Возможность создания и редактирования списка глобальных констант обеспечивается редактором словаря интегрированной среды разработки.

Локальная константа объявляется в секции объявлений внутри программ, процедур или функций, написанных на языке СТ, или в словаре локальных констант программ, процедур или функций, выполненных на языке ФБД, и доступна только программе, процедуре или функции, в которой она объявлена.

При программировании на языке СТ используется синтаксис вида:

Константы

Начало

<список объявлений>

Конец

Список объявлений содержит конструкции вида:

<тип данных> <имя константы> = <значение>

Допускается объявление констант логического, целочисленного и вещественного типов данных. Для указания типа данных используются соответствующие ключевые слова "Лог", "Цел", "Цел8", "Цел16", "Цел32", "Вещ", "Вещ32", "Вещ64", "Стр". Число в конце ключевого слова обозначает разрядность соответствующего типа данных. Типы данных "Цел" и "Цел16", "Вещ" и "Вещ32" эквивалентны.



ВНИМАНИЕ!!!

Типы данных Вещ64, Цел8 и Цел32 могут быть использованы для платформ СРВК, начиная с версии 8.0, и среды исполнения КРУГОЛ, начиная с версии 2.2.

В качестве примера объявления и использования константы можно привести следующий фрагмент текста программы

Константы

Начало

Вещ PI = 3.14159

Конец

пв1 = sin(пв2 * 180 / PI)

3.1.4.2 Переменные

Пользователю интегрированной среды разработки предоставляется возможность объявления собственных переменных и дальнейшего их использования в качестве операндов.

Переменные могут быть как локальными, так и глобальными.



ВНИМАНИЕ!!!

Использование глобальных переменных НЕ допускается для платформ СРВК версии 6.5 и среды исполнения КРУГОЛ версии 1.0.

Возможность создания и редактирования списка глобальных переменных обеспечивается редактором словаря интегрированной среды разработки.

Локальная переменная объявляется в секции объявлений внутри программ, процедур или функций, написанных на языке СТ, или в словаре локальных переменных программ, процедур или функций, выполненных на языке ФБД, и доступна только программе, процедуре или функции, в которой она объявлена.

При программировании на языке СТ используется синтаксис вида:

Переменные

Начало

<список объявлений>

Конец

Список объявлений содержит следующие конструкции

<тип данных> <имя переменной>

или

<тип данных> <имя переменной> = <начальное значение>

Допускается объявление переменных логического, целочисленного и вещественного типов данных. Для указания типа данных используются соответствующие ключевые слова "Лог", "Цел", "Цел8", "Цел16", "Цел32", "Вещ", "Вещ32", "Вещ64". Число в конце ключевого слова обозначает разрядность соответствующего типа данных. Типы данных "Цел" и "Цел16", "Вещ" и "Вещ32" эквивалентны.

Обращение к значению переменной в тексте программы происходит по ее имени.

В качестве примера объявления и использования переменной можно привести следующий фрагмент текста программы:

Переменные

Начало

Вещ sum = 0

Конец

Для i[1...5]

```
{
    sum = sum + пв[i]
}
```

**ВНИМАНИЕ!!!**

Типы данных **Вещ64**, **Цел8** и **Цел32** могут быть использованы для платформ СРВК, начиная с версии 8.0, и среды исполнения КРУГОЛ, начиная с версии 2.2.

3.1.4.3 Массивы

Массивы являются элементами технологического языка, предназначенными для одновременного хранения нескольких значений.

Массивы могут быть как локальными, так и глобальными.

Возможность создания и редактирования описаний глобальных массивов обеспечивается редактором словаря интегрированной среды разработки.

Локальный массив объявляется в секции объявлений переменных внутри программ, процедур или функций, написанных на языке СТ, или в словаре локальных переменных программ, процедур или функций, выполненных на языке ФБД, и доступен только программе, процедуре или функции, в которой он объявлен.

Допускается объявление массивов логического, целочисленного и вещественного типов данных. Для указания типа данных используются соответствующие ключевые слова "Лог", "Цел", "Цел8", "Цел16", "Цел32", "Вещ" "Вещ32", "Вещ64". Число в конце ключевого слова обозначает разрядность соответствующего типа данных. Типы данных "Цел" и "Цел16", "Вещ" и "Вещ32" эквивалентны. В квадратных скобках указывается количество элементов массива.

**ВНИМАНИЕ!!!**

Типы данных **Вещ64**, **Цел8** и **Цел32** могут быть использованы для платформ СРВК, начиная с версии 8.0, и среды исполнения КРУГОЛ, начиная с версии 2.2.

Использование глобальных массивов НЕ допускается для платформ СРВК версии 6.5 и среды исполнения КРУГОЛ версии 1.0

Синтаксис объявления массива в языке СТ следующий:

<тип данных> <имя массива>[<размер>]

Для обращения к элементам массива при задании номера элемента константным значением используется синтаксис:

<имя массива>[<номер элемента>]

или в случае косвенного обращения к элементам, номера которых задаются переменной оператора цикла:

<имя массива>[<переменная оператора цикла>]

Нумерация элементов массива начинается с единицы.

В качестве примера объявления и использования массива можно привести следующий фрагмент текста программы:

```
Переменные
Начало
Вещ массив[5]
Конец
Для i[1...5]
{
    массив[i] = пв[i]
}
```

3.1.4.4 Переменные оператора цикла

Переменные оператора цикла определяются в описании оператора цикла (описание в разделе 3.2.5) и используются при косвенном обращении к системным и внутренним переменным, элементам массива, а также в качестве операндов доступных только для чтения.

3.2 Операции и операторы

Операции определяют действия над данными и используются в выражениях.

Операторы представляют собой конструкции языка, изменяющие значения операндов или управляющие последовательностью обработки данных.

К *операторам, изменяющим значения операндов*, относятся оператор присваивания и операторы включения/выключения таймеров.

К *управляющим операторам* относятся условный оператор, оператор цикла и оператор досрочного завершения.

3.2.1 Операции

Операции, используемые в выражениях, выполняют действие над одним или двумя операндами и формируют результирующее значение, которое может использоваться в других выражениях или операторах.

Унарными являются операции, выполняющие действие над одним операндом. В языке СТ унарной операции соответствует синтаксис:

<символ операции> <операнд>

Бинарными являются операции, выполняющие действие над двумя операндами. Бинарным операциям в языке СТ соответствует синтаксис:

<операнд> <символ операции> <операнд>



ВНИМАНИЕ!!!

Для платформ СРВК до версии 8.0 и среды исполнения КРУГОЛ до версии 2.2 оба бинарных операнда должны иметь один тип .

Операции можно разделить на три группы:

- Арифметические операции
- Операции сравнения
- Логические операции.

Арифметические операции приведены в следующей таблице:

Операция	Символ	Тип операции	Тип операндов
Изменение знака	-	Унарная	Целочисленный, вещественный. Логический* – для СРВК, начиная с версии 8.0, ИСП КРУГОЛ, начиная с версии 2.2
Умножение	*	Бинарная	Целочисленный, вещественный.

Деление	/	Бинарная	Логический* – для СРВК, начиная с версии 8.0, ИСР КРУГОЛ, начиная с версии 2.2
			Целочисленный, вещественный.
Сложение	+	Бинарная	Логический* – для СРВК, начиная с версии 8.0, ИСР КРУГОЛ, начиная с версии 2.2
			Целочисленный, вещественный.
Вычитание	-	Бинарная	Логический* – для СРВК, начиная с версии 8.0, ИСР КРУГОЛ, начиная с версии 2.2
			Целочисленный, вещественный.

* – В указанных операциях значение логического типа интерпретируется как целое 8-битовое значение.



ВНИМАНИЕ!!!

Тип результата выполнения арифметической операции для платформ СРВК до версии 8.0 и среды исполнения КРУГОЛ до версии 2.2 соответствует типу операндов, для платформ СРВК, начиная с версии 8.0, и среды исполнения КРУГОЛ, начиная с версии 2.2 – соответствует типу операнда с наибольшей размерностью или должен иметь тип с большей размерностью, чем типы операндов выражения.

К операциям сравнения относятся:

Операция	Символ	Тип операции	Тип операндов
Больше	>	Бинарная	Целочисленный, вещественный.
			Логический* – для СРВК, начиная с версии 8.0, ИСР КРУГОЛ, начиная с версии 2.2
Меньше	<	Бинарная	Целочисленный, вещественный.
			Логический* – для СРВК, начиная с версии 8.0, ИСР КРУГОЛ, начиная с версии 2.2
Больше или равно	>=	Бинарная	Целочисленный, вещественный.
			Логический* – для СРВК, начиная с версии 8.0, ИСР КРУГОЛ, начиная с версии 2.2
Меньше или равно	<=	Бинарная	Целочисленный, вещественный.
			Логический* – для СРВК, начиная с версии 8.0, ИСР КРУГОЛ, начиная с версии 2.2
Равно	=	Бинарная	Логический, целочисленный, вещественный
Не равно	#	Бинарная	Логический, целочисленный, вещественный

* – В указанных операциях значение логического типа интерпретируется как целое 8-битовое значение.

Операции сравнения возвращают результат логического типа.

К логическим операциям относятся

Операция	Символ	Тип операции	Тип операндов
Логическое НЕ	!	Унарная	Логический
Логическое И	&	Бинарная	Логический
Исключающее ИЛИ	^	Бинарная	Логический
Логическое ИЛИ		Бинарная	Логический

Логические операции возвращают результат логического типа.

Вычисление сложных выражений производится слева направо с учетом приоритета операций.

Операция	Приоритет
Изменение знака, логическое НЕ	Максимальный
Умножение, деление	
Сложение, вычитание	
Операции сравнения	
Логическое И	
Исключающее ИЛИ	
Логическое ИЛИ	Минимальный

Для явного указания приоритета возможно использование круглых скобок.

Ниже приведен пример выражения:

пв1 > 10 & пв1 < 20

Эквивалентное выражение с явным указанием приоритета с помощью круглых скобок имеет вид:

(пв1 > 10) & (пв1 < 20)

3.2.2 Оператор присваивания

Оператор присваивания обеспечивает формирование нового значения операнда. Синтаксис присваивания на языке СТ имеет вид:

<операнд-приемник> = <операнд-источник или выражение>

В качестве операнда-приемника могут использоваться доступные для записи атрибуты системных переменных, переменные и элементы массивов всех видов. Исключением являются переменные оператора цикла.

В качестве источника возможно использование любых операндов.

Пример операторов присваивания:

дв1 = 1

пв1 = пв2 + пв3

3.2.3 Операторы включения/выключения таймеров

Операция включения таймера обеспечивает начало отсчета времени соответствующей таймерной переменной:

Вкл <переменная таймера>

Операция выключения таймера используется для прекращения счета:

Выкл <переменная таймера>



ВНИМАНИЕ!!!

Операция выключения таймера прекращает счет, но не выполняет сброс таймера. Для сброса таймера необходимо присвоить нулевое значение соответствующей таймерной переменной.

В качестве примера можно привести фрагмент программы:

```
Вкл тс1
Если (тс1 > 20)
{
    тс1 = 0
}
Если тс1 = 0
{
    Для i[2...10]
    {
        Вкл тс[i]
    }
}
Если тс1 >= 10
{
    Для i[2...10]
    {
        Выкл тс[i]
    }
}
```

3.2.4 Условный оператор

Условный оператор обеспечивает выбор выполняемых операторов в зависимости от результата вычисления условного выражения.

Условное выражение должно возвращать результат логического типа.

Синтаксис условного оператора на языке СТ имеет вид:

```
Если <логический операнд или условное выражение>
{
    <список операторов>
}
Иначе_Если <логический операнд или условное выражение>
{
    <список операторов>
}
Иначе
{
    <список операторов>
}
```


Список операторов секции "Если" выполняется при истинности соответствующего логического операнда или условного выражения (результат вычисления равен единице), при этом операторы других секций (Иначе_Если, Иначе) не выполняются.

В случае ложности (равенства нулю) условия секции "Если" проверяется альтернативное условие в секции "Иначе_Если". Допускается использование нескольких секций "Иначе_Если". Если условие секции "Иначе_Если" истинно, выполняется список операторов соответствующей секции, в противном случае аналогичным образом проверяется альтернативное условие в следующей секции "Иначе_Если".

Список операторов секции "Иначе" выполняется в случае ложности условий всех предыдущих секций.

Секция "Если" является обязательной. Остальные секции могут отсутствовать.

Фрагмент программы, демонстрирующий использование оператора условия:

```
Если пв1 < 10.0
{
    пц1 = 1
}
Иначе_Если пв1 < 20.0
{
    пц1 = 2
}
Иначе_Если пв1 < 30.0
{
    пц1 = 3
}
Иначе
{
    пц1 = 0
}
```



ВНИМАНИЕ!!!

Имеется отличие при вычислении логических выражений оператора **Если** для трансляторов языка КРУГОЛ версии 2.0 (ИСР КРУГОЛ 2.0) и более ранних версий (ИСР КРУГОЛ 1.0). Проиллюстрируем данное отличие на следующем примере:

```
Если пл1 = пл2 & пл3
{
    <список операторов>
}
```

ИСР КРУГОЛ 2.0: Транслятор воспринимает условие $пл1 = пл2 \& пл3$ как выражение $(пл1 = пл2) \& пл3$: сначала проверяются на равенство переменные $пл1$ и $пл2$ – приоритет операции сравнения выше чем приоритет операции логического умножения – после чего выполняется логическое умножение полученного результата и переменной $пл3$

ИСР КРУГОЛ 1.0: Транслятор воспринимает условие $пл1 = пл2 \& пл3$ как выражение $пл1 = (пл2 \& пл3)$: в этом случае $пл1$ сравнивается с результатом логического умножения $пл2$ на $пл3$.

РЕКОМЕНДУЕМ : Для корректного вычисления логических выражений определяйте порядок действий явно – с помощью скобок!!!

3.2.5 Оператор цикла

Оператор цикла обеспечивает циклическое выполнение списка операторов.

Переменные оператора цикла принимают на каждом шаге цикла одно значение из заданного множества значений (на первом шаге цикла – первое значение переменной, на втором – второе и т.д.).

Переменные оператора цикла могут принимать только целочисленные значения в диапазоне от 0 до 32767.

Для оператора цикла в языке СТ предусмотрен следующий синтаксис:

```
Для <имя переменной 1>[<список значений>], ... , <имя переменной N>[<список значений>]  
{  
  <список операторов>  
}
```

Список значений содержит значения, перечисленные через запятую. В списке также допускается задание диапазонов значений, при этом указываются начальное и конечное значения, разделенные многоточием "...".



ВНИМАНИЕ!!!

Количество шагов цикла определяется количеством значений переменных оператора цикла.

Количество значений, принимаемых переменными оператора цикла, должно быть одинаковым.

Переменные оператора последовательности могут использоваться внутри него для косвенного обращения (т.е. для использования в качестве индекса) к системным и внутренним переменным, элементам массива, а также в качестве операндов доступных только для чтения. Выполнение оператора последовательности может быть завершено досрочно с помощью оператора «Прервать».

Ниже в качестве примера приведен фрагмент программы с использованием оператора цикла.

```
Для i[1, 2, 3...10, 20, 30], j[1...12]  
{  
  пв[i] = i * i  
  Если j = 5  
  {  
    Прервать  
  }  
}
```

3.2.6 Оператор досрочного завершения

Выполнение оператора досрочного завершения приводит к немедленному выходу из текущей программы, процедуры или функции. В языке СТ данному оператору соответствует ключевое слово **Выход**.

Пример использования оператора **Выход** приведен в описании процедуры (подраздел 3.5).

3.3 Комментарии

В тексте программы на языке СТ имеется возможность использования как однострочных, так и многострочных комментариев.

Однострочные комментарии записываются в форме

: <текст однострочного комментария>

Многострочные комментарии записываются в форме

**(*
 <текст многострочного
 комментария>
*)**

3.4 Программа

Программа является базовой конструкцией технологического языка программирования КРУГОЛ.

Выполнение программы производится на каждом цикле обработки в среде исполнения контроллера или станции оператора.

Программы выполняются безусловно. Исключением является использование системной функции «**ВЫЗОВ**» в программах, исполняемых в контроллере.

Для определения программы на языке СТ используется следующий синтаксис:

Программа **<имя программы>**
Начало
 <локальные объявления>
 <список операторов>
Конец

Секция локальных объявлений содержит объявления пользовательских констант, переменных или массивов. Данная секция является необязательной.

Поведение программы определяется последовательным выполнением операторов следующих после секции локальных объявлений.

Пример простой программы:

Программа Main
Начало
 message("Hello world!")
Конец

3.5 Процедура

Процедура является логически завершенной частью программы.

Процедура не принимает и не возвращает значений.

Для определения процедуры на языке СТ используется следующий синтаксис:

Процедура *<имя процедуры>*

Начало

<локальные объявления>

<список операторов>

Конец

Секция локальных объявлений содержит объявления пользовательских констант, переменных или массивов. Данная секция является необязательной.

Поведение процедуры определяется последовательным выполнением операторов.

В отличие от программ, процедура никогда не выполняется безусловно. Для выполнения процедуры ее необходимо вызвать из какой-либо программы, функции или другой процедуры. Вызов процедуры осуществляется по ее наименованию отдельным оператором. Синтаксис оператора вызова процедуры следующий: *<имя процедуры>*

В качестве примера объявления и вызова процедуры можно привести следующий текст на языке СТ:

Программа Prog

Начало

(* Вызываем процедуру *)

Proc

Конец

Процедура Proc

Начало

пв1 = пв1 + 1

Конец

Пример использования оператора "Выход" в процедуре:

Процедура Test_Proc

Начало

Если !пл1

{

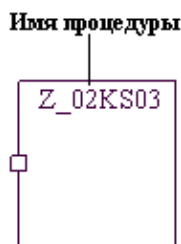
Выход

}

пв1 = пв1 + 1

Конец

Пример вызова процедуры на языке ФБД:



3.6 Функции

Функция, как и процедура, является логически завершенной частью программы.

В отличие от процедуры функция может принимать и возвращать значения (иметь входные и выходные параметры).

Для определения функции на языке СТ используется следующий синтаксис:

```

Функция <имя функции>
  Входные_Переменные
  Начало
    <список объявлений>
  Конец
  Выходные_Переменные
  Начало
    <список объявлений>
  Конец
Начало
  <локальные объявления>
  <список операторов>
Конец
    
```

Для передачи в функцию параметров используются переменные, объявленные в секции "Входные_Переменные". Если функция не принимает параметров, допускается отсутствие данной секции. Для объявления входных переменных используется синтаксис вида:

<тип данных> <имя переменной>

или

<тип данных> <имя переменной> = <значение по умолчанию>

В качестве входных параметров допускается использовать переменные логического, целочисленного и вещественного типов данных. Для указания типа данных переменной используются соответствующие ключевые слова "Лог", "Цел", "Цел8", "Цел16", "Цел32", "Вещ" "Вещ32", "Вещ64".



ВНИМАНИЕ!!!

Типы данных Вещ64, Цел8 и Цел32 могут быть использованы для платформ СРВК, начиная с версии 8.0, и среды исполнения КРУГОЛ, начиная с версии 2.2.



ВНИМАНИЕ!!!

Если значение входного параметра при вызове функции не было сформировано в вызывающей программе или не задано Пользователем явно, транслятор языка КРУГОЛ использует в качестве значения такого параметра значение по умолчанию, заданное в секции "Входные_Переменные"

Возврат значений из функции обеспечивается с помощью переменных, объявленных в секции "Выходные_Переменные". Если функция не возвращает значений, допускается отсутствие данной секции. Выходные переменные объявляются аналогично входным:

<тип данных> <имя переменной>

или

<тип данных> <имя переменной> = <значение по умолчанию>

**ВНИМАНИЕ!!!**

Значение по умолчанию в секции "Выходные_Переменные" обеспечивает инициализацию соответствующей выходной переменной.

Секция локальных объявлений содержит объявления пользовательских констант, переменных или массивов. Данная секция является необязательной.

Поведение функции определяется выполнением операторов.

Аналогично процедуре, функция никогда не выполняется безусловно и может только вызываться из программ, процедур или других функций. Синтаксис вызова функции имеет следующий вид:

<имя функции>(<параметр 1>, <параметр 2>, ... , <параметр N>)

Порядок перечисления входных параметров соответствует порядку объявления входных переменных при определении функции. Если какие либо из входных переменных функции принимают значение по умолчанию, допускается оставлять соответствующие позиции в списке пустыми. В этом случае значение по умолчанию используется в качестве параметра.

Возврат значений из функции имеет несколько особенностей:

- Если вызов функции осуществляется отдельным оператором, расположенным в списке операторов, то все возвращаемые функцией значения игнорируются – функция вызывается как процедура!
- Если функция возвращает одно значение, то вызов такой функции может использоваться в качестве операнда с типом данных, соответствующим типу возвращаемого значения
- Если функция возвращает несколько значений, то для их получения необходимо использование особой формы оператора присваивания. В этом операторе переменным-приемникам присваиваются значения результатов вызова функции. Синтаксис такого оператора присваивания имеет вид

(<переменная 1>, ... , <переменная N>) = <вызов функции>

Допускается оставлять пустыми позиции в списке переменных. Эти позиции соответствуют переменным, возвращаемые значения которых не интересуют Пользователя.

Кроме функций, определяемых пользователем на языке КРУГОЛ, существуют функции, встроенные в Среду исполнения, которые могут вызываться из программ Пользователя аналогичным образом.

Пример использования функции на языке СТ:

Программа Test

Начало

(пл1, пл2, пл3) = Сравнение(пв1, пв2)

пв3 = пв4 + MyAdd(пв5, пв6)

Конец

Функция Сравнение

Входные_Переменные

Начало

Вещ значение1

Вещ значение2

Конец

Выходные_Переменные

Начало

Лог больше = 0

Лог равно = 0

Лог меньше = 0

Конец

Начало

Если значение1 > значение2

{

больше = 1

Выход

}

Если значение1 < значение2

{

меньше = 1

Выход

}

равно = 1

Конец

Функция MyAdd

Входные_Переменные

Начало

Вещ val1

Вещ val2

Конец

Выходные_Переменные

Начало

Вещ res

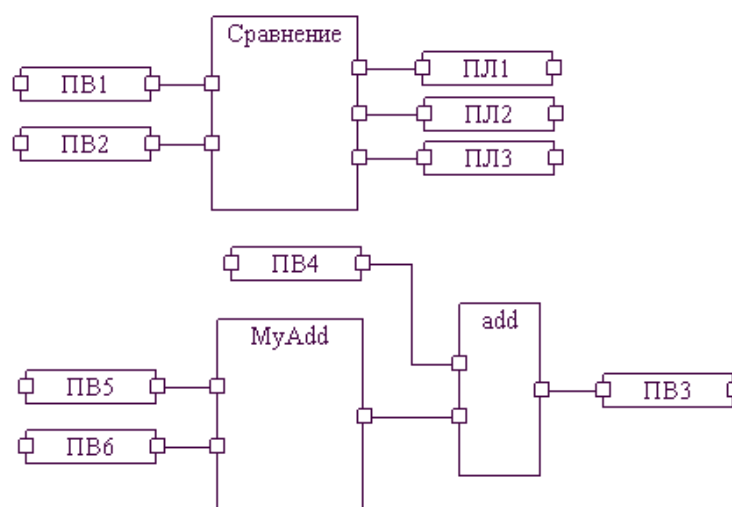
Конец

Начало

res = val1 + val2

Конец

Пример вызова функции в языке ФБД:



3.7 «Перегрузка» вызова функций

Начиная с библиотеки версии 2.0, в языке «КРУГОЛ» появилось множество функций, которые реализуют одну и ту же операцию вычисления, но с различными типами данных. Например: **Add_i8** – сложение целых 8 битовых чисел, **Add_i16** – сложение целых 16 битовых чисел, **Add_r32** – сложение вещественных 32 битовых чисел и т. д.

Функцию, которая реализует операцию для конкретных типов данных, будем называть *типизированной* (например, Add_i8).

Для соблюдения уникальности имён функций в языке «КРУГОЛ» принято соглашение, что имена типизированных функции будут иметь суффикс, соответствующий типу данных, с которым работает функция. Суффиксу соответствует синтаксис:

_<тип данных> <размер данных>

Таблица соответствия типов данных и суффиксов приведена ниже.

Тип данных	Суффикс1	Суффикс2
Вещ32	_r32	_в32
Вещ64	_r64	_в64
Цел8	_i8	_ц8
Цел16	_i16	_ц16
Цел32	_i32	_ц32

Для удобства пользователя в интегрированной среде языка «КРУГОЛ» разработан механизм «перезгрузки» вызова *типизированных функций*. Суть этого механизма заключается в том, что Пользователь может для вызова оператора вычисления выражений указать *обобщенное имя* (без суффикса) функции и операнды, требуемого типа. Например, для сложения целых чисел – Add (5, 2), для сложения вещественных – Add (2.5, 1.2).

В процессе трансляции программы на основании имени функции, типов входных параметров и их количества транслятор решает какую функцию конкретно вызвать и подставляет в исполняемый код вызов соответствующей типизированной функции.

В случае если транслятор не сможет найти соответствующую типизированную функцию, то будет выдано сообщение об ошибке и исполняемый код не будет сформирован.



ВНИМАНИЕ!!!

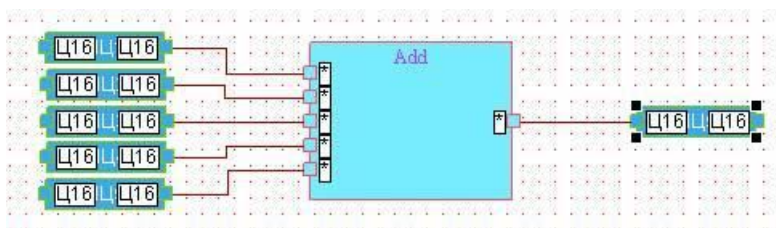
Механизм перегрузки вызова работает только для типизированных функций, которые имеют:

- Одинаковые обобщенные имена
- Одинаковое количество входных параметров
- Одинаковый тип входных параметров.

Примеры вызова функции на языке СТ:

Пц1 = Add(1,1) : с использованием механизма перегрузки вызова
Пв1 = Слож(1.1, 2.2) : с использованием механизма перегрузки вызова
Пц1 = Add_ц16(1,1) : без использованием механизма перегрузки вызова
Пв1 = Слож_в32(1.1, 2.2) : без использованием механизма перегрузки вызова

В языке ФБД входные контакты «перегруженной» функции будут отображаться, как контакты неопределённого типа (отмечены звездочкой):



3.8 Алгоблок

Алгоблок, аналогично процедуре и функции, является логически завершенной частью программы.

Алгоблок, также как и функция, принимает входные и возвращает выходные параметры. Несмотря на сходство с функцией, алгоблок существенно отличается от нее. В отличие от функции, алгоблок обладает способностью запоминать свое внутреннее состояние, сформированное при его вызове, а также использовать это состояние при повторном вызове. Внутреннее состояние алгоблока определяется значениями его локальных и выходных переменных, а также внутренним состоянием вызываемых из него других алгоблоков. Значения входных переменных алгоблока не определяют его внутреннего состояния (и следовательно не запоминаются), т.к. они формируются заново при каждом вызове алгоблока.



ВНИМАНИЕ!!!

Автоматическая нумерация алгоблоков НЕ поддерживается для платформ СРВК версий 6.5, 7.0 и среды исполнения КРУГОЛ версий 1.0, 2.0.

В связи с наличием внутреннего состояния алгоблока, для идентификации этого состояния при вызове должен указываться дополнительный первый параметр – номер алгоблока.

Алгоблоки могут быть системными, т.е. встроенными в библиотеку среды исполнения, или пользовательскими, разрабатываемыми Пользователем ИСР КРУГОЛ на технологических языках СТ или ФБД.

Для определения алгоблока на языке СТ используется следующий синтаксис:

```

Алгоблок <имя функции>
  Входные_Переменные
  Начало
    <список объявлений>
  Конец
  Выходные_Переменные
  Начало
    <список объявлений>
  Конец
Начало
  <локальные объявления>
  <список операторов>
Конец
    
```

Синтаксис определения алгоблока отличается от синтаксиса определения функции только заменой ключевого слова «Функция» на ключевое слово «Алгоблок».

Синтаксис вызова алгоблока имеет следующий вид:

<имя алгоблока>(<номер алгоблока 1>, <параметр 1>, <параметр 2>, ... , <параметр N>)

Возврат значений из алгоблока ничем не отличается от возврата значений из функции.

3.9 Автоматическая нумерация алгоблоков

Явное задание номера алгоблока при его вызове требует от пользователя тщательного контроля отсутствия нежелательного повторного вызова алгоблока с одним и тем же номером. Для устранения возможности возникновения подобных нежелательных эффектов в ИСР КРУГОЛ реализована автоматическая нумерация алгоблоков. Для автоматического формирования номера он просто не указывается при вызове алгоблока (при программировании на языке ФБД соответствующий вход остается неподключенным).

Если пользовательский алгоблок содержит вызовы других алгоблоков с автонумерацией, то при вызове данного алгоблока с разными номерами обеспечивается отсутствие совпадения номеров вложенных алгоблоков.



ВНИМАНИЕ!!!

Автоматическая нумерация алгоблоков НЕ поддерживается для платформ СРВК версий 6.5, 7.0 и среды исполнения КРУГОЛ версий 1.0, 2.0.

Пример использования алгоблока на языке СТ:

Программа Test

Начало

: Явное задание номера.

пл1 = ДелительЧастоты(1, пл2)

: Автоматическая нумерация.

пл3 = ДелительЧастоты(, пл4)

Конец

Алгоблок ДелительЧастоты

Входные_Переменные

Начало

Лог вх

Конец

Выходные_Переменные

Начало

Лог вых

Конец

Начало

Переменные

Начало

Лог фр

Конец

фр = ПФр(, вх)

вых = (!вых & фр) | (вых & !фр)

Конец

3.10 Примеры программ на языке СТ

Пример 1. Дозирование по значению мгновенного расхода

Необходимо, управляя отсечным клапаном, подавать через каждые 3 минуты дозу раствора (V), заданную технологом – оператором.

Часовой расход раствора подсчитывается по формуле:

$$\text{часовой расход (л/ч)} = 60 \text{ мин./3 мин.} * V$$

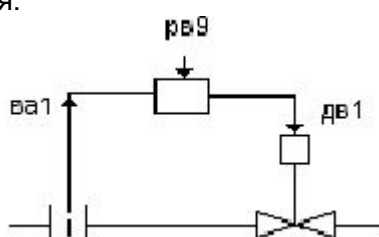
Системные переменные:

ва1 – измеряемый текущий расход, л/ч;

рв9 – задаваемая величина разовой дозы, л;

дв1 – управляющий выход на клапан дозирования, 0- закрыть, 1- открыть;

Структура системы дозирования:



ПРОГРАММА Doza

НАЧАЛО

```

Если пл1= 0                : Включение счетчиков времени
{
    Вкл тм1                  : в первом проходе процедуры (ПЛ1=0 флаг первого прохода)
    ва1= 60                   : ТМ1 - счетчик времени пауза 3 минуты
    Вкл тс1                  : ТС1- счетчик времени такта интеграции
    пл1= 1
}
Если тм1> 3.0               : Проверка окончания паузы 3 минуты
{
    тм1= тм1- 3.0             : Сброс таймера паузы с учетом задержки цикла ПрП
    дв1= 1                    : Формирование команды "Клапан открыть"
}
Если тс1> 3.0               : Проверка окончания эталонного такта 3 сек
                             : для интеграции текущего расхода
{
    тс1=тс1- 3.0              : Сброс таймера с учетом задержки цикла ПрП
    пв3= пв3+ ва1/1200.0      : Интеграции текущего расхода
    Если пв3> рв9             : Проверка выполнения величины дозы
    {
        Если дв1= 1          : Проверка выполнения дозы
        {
            пв3= пв3- рв9      : Определение ошибки дозы для учета в следующем такте
            дв1= 0             : Формирование команды "Клапан закрыть"
        }
    }
}
КОНЕЦ

```

Пример 2. Калькулятор для значений часовых трендов

Необходимо определить суммарный расход раствора за указываемый технологическим оператором интервал времени, если известны расходы за каждый час в сутках.

Операнды:

пв101- пв124 – расход раствора за каждый час суток

рв11 – астрономическое время начала определения суммы (ручной ввод)

рв12 – астрономическое время окончания определения суммы (ручной ввод)

пв9 – результат (суммарный расход за указанный период)

ПРОГРАММА **summa****НАЧАЛО**

```

Если рв12> 0.0           : Вход в определение суммы, по вводу времени
{
    пв1= 0
    пв9= 0.0
    Для hF[101...124]       : Список переменных ПВ101-ПВ124
    {
        пв1= пв1+ 1         : Формирование времени суток с 1 по 24 час для
                               : каждого прохода оператора "Для"

        Если пв1> рв12       : Поиск часа окончания определения суммы
        {                     : для выхода из процедуры "summa"
            рв12= 0.0
            ВЫХОД
        }

        Если пв1> рв11       : Поиск заданного интервала времени
        {
            пв9= пв9+ пв[hF] : Определение суммы за заданный интервал
        }
    }
    рв12= 0.0
}

```

КОНЕЦ**Пример3.** Поиск отклонений от общей тенденции изменений значений

Необходимо определить параметры, несоответствующие общей тенденции изменения значений по группе технологически связанных параметров (нагрев котла).

Функцию выполнять до обнаружения 5-ти «отклонившихся» параметров.

Для проверки программы запрограммировать имитатор значений параметров группы (процедура "отладка").

Операнды:

ва21 - ва32 – группа параметров с общей тенденцией изменения

рв10 – количество отклонившихся параметров от общей тенденции

рв20 – процент допустимого отклонения от среднего значения

ПРОГРАММА poisk1	: Определение среднего значения по группе
НАЧАЛО	
Отладка	: Вызов отладочной процедуры
Если рв10 < 5.0	: Проверка на допустимость выполнения функции
{	
Для i1[1...12]	: Организация проходов процедуры "poisk1"
	: При досрочном выходе из процедуры "poisk2"
{	
пв2 = 0	
пв3 = 0	
Для i2[21...32]	
{	
Если пл[i2] = 0	: Проверка флага нарушения тенденции
{	
пв2 = пв2 + ва[i2]	: Сумма достоверных значений
пв3 = пв3 + 1	: Число слагаемых
}	
}	
пв4 = пв2 / пв3	: Среднее значение по группе параметров
пв5 = пв4 * (1.0 - рв20 / 100.0)	: Нижняя граница допустимых значений в группе
пв6 = пв4 * (1.0 + рв20 / 100.0)	: Верхняя граница допустимых значений в группе
poisk2	: Вызов процедуры "poisk2"
Если пл33 = 0	: Выход из процедуры "poisk1" по отсутствию флага пл33
{	
ВЫХОД	: прерывания процедуры "poisk2"
}	
}	
}	
КОНЕЦ	
ПРОЦЕДУРА poisk2	: Анализ текущих значений на достоверность
НАЧАЛО	
пл33 = 0	
рв10 = 0	
Для i3[21...32]	: Поиск нарушения тенденции новым параметром
{	
Если ва[i3] > пв6	: Проверка нарушения верхней границы тенденции
{	
пл33 = 1	: Флаг прерывания процедуры
}	
Если ва[i3] < пв5	: Проверка нарушения нижней границы тенденции
{	
пл33 = 1	: Флаг прерывания процедуры
}	
}	

```

Если пл33= 0
{
    пл[i3]= 0           : Сброс флага нарушения тенденции
}
Иначе
{
    Если пл[i3]= 0       : Вход для нового нарушения тенденции
    {
        пл[i3]= 1       : Флаг нарушения тенденции
        ВЫХОД          : Возврат на перерасчет среднего значения
    }
    Иначе              : Обработка ранее обнаруженного нарушения
    {
        рв10=рв10+1     : Подсчет флагов нарушений
        пл33= 0         : Сброс флага прерывания
    }
}
}
КОНЕЦ

```

ПРОЦЕДУРА Отладка **НАЧАЛО**

```

Если пл39= 0           : Стартовая инициализация значений
{
    пл39= 1
    Вкл тс1
    рв20= 1.0           : Процент разрешенного отклонения = 1 %
}
Для i4[21...32]       : Формирование текущих значений ва21-ва32
{
    ва[i4]= ва[i4]+ 0.05
}
Если тс1> 60          : Внесение возмущения для двух переменных
{
    тс1= 0
    Для i5[22,30]
    {
        ва[i5]= ва[i5]*1.015
    }
}
КОНЕЦ

```

3.11 Формальное описание грамматики языка СТ

Данный раздел содержит формальное описание грамматики технологического языка КРУГОЛ в форме Бэкуса-Наура.

Транслятор программ на языке КРУГОЛ является нечувствительным к регистру символов.

Проект состоит из произвольного количества определений программ, процедур и функций.
`project ::= {program | procedure | function | algoblock}`

Определение программы начинается ключевым словом ПРОГРАММА, за которым следуют ее идентификатор (или имя) и блок операторов.

`program ::= 'ПРОГРАММА' identifier statement_block`

Определение процедуры начинается ключевым словом ПРОЦЕДУРА, за которым следуют ее идентификатор (или имя) и блок операторов.

`procedure ::= 'ПРОЦЕДУРА' identifier statement_block`

Определение функции начинается ключевым словом ФУНКЦИЯ, за которым следуют ее идентификатор (или имя), секция объявлений входных переменных, секция объявлений выходных переменных и блок операторов.

`function ::= 'ФУНКЦИЯ' identifier
 [inputs_declaration] [outputs_declaration] statement_block`

Определение алгоблока начинается ключевым словом АЛГОБЛОК, за которым следуют его идентификатор (или имя), секция объявлений входных переменных, секция объявлений выходных переменных и блок операторов.

`algoblock ::= 'АЛГОБЛОК' identifier
 [inputs_declaration] [outputs_declaration] statement_block`

Секция объявлений входных переменных начинается ключевым словом ВХОДНЫЕ_ПЕРЕМЕННЫЕ, за которым следует список объявлений переменных, ограниченный ключевыми словами НАЧАЛО и КОНЕЦ.

`inputs_declaration ::=
 'ВХОДНЫЕ_ПЕРЕМЕННЫЕ' 'НАЧАЛО' var_declaration_list 'КОНЕЦ'`

Секция объявлений выходных переменных начинается ключевым словом ВЫХОДНЫЕ_ПЕРЕМЕННЫЕ, за которым следует список объявлений переменных, ограниченный ключевыми словами НАЧАЛО и КОНЕЦ.

`outputs_declaration ::=
 'ВЫХОДНЫЕ_ПЕРЕМЕННЫЕ' 'НАЧАЛО' var_declaration_list 'КОНЕЦ'`

Блок операторов начинается ключевым словом НАЧАЛО, за которым следуют список локальных объявлений и далее список операторов. Блок операторов завершается ключевым словом КОНЕЦ.

`statement_block ::=
 'НАЧАЛО' local_declaration_list statement_list 'КОНЕЦ'`

Список локальных объявлений содержит произвольное количество секций объявлений локальных переменных и констант.

`locals_declaration ::=
 {local_vars_declaration | local_constants_declaration}`

Секция объявлений локальных переменных начинается ключевым словом ПЕРЕМЕННЫЕ, за которым следует список объявлений переменных, ограниченный ключевыми словами НАЧАЛО и КОНЕЦ.

```
local_vars_declaration ::=  
  'ПЕРЕМЕННЫЕ' 'НАЧАЛО' local_var_declaration_list 'КОНЕЦ'
```

Секция объявлений локальных констант начинается ключевым словом КОНСТАНТЫ, за которым следует список объявлений констант, ограниченный ключевыми словами НАЧАЛО и КОНЕЦ.

```
local_constants_declaration ::=  
  'КОНСТАНТЫ' 'НАЧАЛО' constant_declaration_list 'КОНЕЦ'
```

Список объявлений переменных может содержать произвольное количество объявлений.

```
var_declaration_list ::= {var_declaration}
```

Объявление переменной состоит из спецификатора типа переменной, за которым следуют идентификатор имени переменной, необязательное начальное значение и символ перевода строки.

```
var_declaration ::= type_name identifier [init_value] end_of_line
```

Список объявлений локальных переменных может содержать произвольное количество объявлений переменных и массивов.

```
local_var_declaration_list ::=  
  {var_declaration | array_declaration}
```

Объявление массива состоит из спецификатора типа данных массива, за которым следует идентификатор его имени. После идентификатора следует константное выражение в квадратных скобках, которое определяет размер массива.

```
array_declaration ::=  
  type_name identifier '[' expression ']' end_of_line
```

Список объявлений констант может содержать произвольное количество объявлений.

```
constant_declaration_list ::= {constant_declaration}
```

Объявление константы состоит из идентификатора типа константы, за которым следуют идентификатор имени константы начальное значение и символ перевода строки.

```
constant_declaration ::=  
  type_name identifier init_value end_of_line
```

Спецификаторами типов данных являются ключевые слова ЛОГ, ЦЕЛ, ЦЕЛ8, ЦЕЛ16, ЦЕЛ32, ВЕЩ, ВЕЩ32, ВЕЩ64 и СТР, которые обозначают различные типы данных.

```
type_name ::= 'ЛОГ' | 'ЦЕЛ' | 'ЦЕЛ8' | 'ЦЕЛ16' | 'ЦЕЛ32' | 'ВЕЩ' | 'ВЕЩ32' | 'ВЕЩ64' | 'СТР'
```

Начальное значение определяется знаком равенства, за которым следует числовое значение константы.

```
init_value ::= '=' constant_literal
```

Список операторов состоит из произвольного количества операторов.

```
statement_list ::= {statement}
```

В качестве оператора могут использоваться следующие элементы языка СТ: вызов процедуры, вызов функции, оператор выхода; оператор досрочного завершения; оператор включения

таймера, оператор выключения таймера; оператор присваивания; условный оператор и оператор последовательности.

Одиночный перевод строки является пустым оператором, который игнорируется транслятором.

```
statement ::= end_of_line |  
  procedure_call end_of_line | function_call end_of_line |  
  return_statement end_of_line | break_statement end_of_line |  
  timer_statement end_of_line | assignment_statement end_of_line |  
  if_statement end_of_line | for_statement end_of_line
```

Вызов процедуры обеспечивается с помощью записи ее идентификатора.

```
procedure_call ::= identifier
```

Оператору выхода соответствует ключевое слово выход.

```
return_statement ::= 'ВЫХОД'
```

Оператор включения/выключения таймера определяется описанием действия над таймером и последующим указанием таймерной переменной.

```
timer_statement ::= timer_action krug_timer_variable
```

Действие над таймером предполагает две операции – его включение и выключение. Для этого используются ключевые слова ВКЛ и ВЫКЛ.

```
timer_action ::= 'ВКЛ' | 'ВЫКЛ'
```

В качестве оператора присваивания может использоваться конструкция, состоящая из элемента языка, доступного для записи, за которым следует знак равенства и арифметическое выражение. Также в качестве оператора присваивания может использоваться конструкция, используемая при вызове функции, возвращающей несколько значений.

```
assignment_statement ::= lvalue '=' expression |  
  '(' function_return_parameter_list ')' '=' function_call
```

Для вызова функции используется запись идентификатора имени функции и списка входных параметров в круглых скобках.

```
function_call ::= identifier '(' function_input_parameter_list ')'
```

Список входных параметров функции содержит параметры, перечисленные через запятую.

```
function_input_parameter_list ::=  
  function_input_parameter {' function_input_parameter}
```

В качестве входного параметра может использоваться выражение или строковая константа. Если выражение отсутствует – используется значение параметра по умолчанию.

```
function_input_parameter ::= [expression | string_literal]
```

Список выходных параметров функции содержит параметры, перечисленные через запятую.

```
function_return_parameter_list ::=  
  function_return_parameter {' function_return_parameter}
```

В качестве выходного параметра может использоваться элемент языка, доступный для записи. Если такой элемент отсутствует – значение выходного параметра функции игнорируется.

```
function_return_parameter ::= [lvalue]
```

Условный оператор начинается ключевым словом ЕСЛИ, за которым следуют условное выражение, список операторов в фигурных скобках и альтернативное условие.

```
if_statement ::=
```

'ЕСЛИ' expression '{' statement_list '}' else_statement

Альтернативное условие может отсутствовать. В секциях ИНАЧЕ_ЕСЛИ используется правая рекурсия для проверки нескольких альтернативных условий и обеспечения требования завершения всей конструкции опциональной секцией ИНАЧЕ.

else_statement ::=

```
[  
'ИНАЧЕ_ЕСЛИ' expression '{' statement_list '}' else_statement |  
'ИНАЧЕ' '{' statement_list '}'  
]
```

Оператор цикла начинается ключевым словом ДЛЯ, за которым следуют список объявлений переменных оператора последовательности и список операторов в фигурных скобках.

for_statement ::= 'ДЛЯ' for_var_declaration_list '{' statement_list '}'

Объявления переменных оператора цикла перечисляются через запятую.

for_var_declaration_list ::= for_var_declaration {' , ' for_var_declaration }

Объявление переменной оператора цикла состоит из идентификатора, за которым в квадратных скобках находится список значений.

for_var_declaration ::= identifier '[' for_var_value_list ']'

Список значений переменной оператора цикла содержит значения, перечисленные через запятую.

for_var_value_list ::=

for_var_value_element {' , ' for_var_value_element }

В качестве значения переменной оператора цикла могут использоваться одно значение или диапазон значений.

for_var_value_element ::= for_var_value |

for_var_value '...' for_var_value

Значение переменной оператора цикла может быть представлено числовым значением или идентификатором константы.

for_var_value ::= index | identifier

Оператор досрочного завершения выполнения оператора цикла.

break_statement ::= 'ПЕРВАТЬ'

Арифметическое выражение, в котором логическое ИЛИ обладает минимальным приоритетом.

expression ::= or_expression

Логическое ИЛИ с приоритетом, меньшим, чем у логического И.

or_expression ::=

and_expression | or_expression '|' and_expression

Логическое И с приоритетом, меньшим, чем у операции сравнения.

and_expression ::= comparison | and_expression '&' comparison

Операция сравнения с приоритетом, меньшим, чем у операций сложения и вычитания.

comparison ::= add_sub_expression | comparison

('<' | '>' | '<=' | '>=' | '=' | '#') add_sub_expression

Операция сложения или вычитания с приоритетом меньшим, чем у операций умножения и деления.

`add_sub_expression ::= mult_div_expression |
add_sub_expression ('+' | '-') mult_div_expression`

Операция умножения или деления с приоритетом, меньшим, по сравнению с унарной операцией.

`mult_div_expression ::= unary_expression |
mult_div_expression ('*' | '/') unary_expression`

Унарная операция изменения знака или инверсии логического значения.

`unary_expression ::= ([ '-' ] | [ '! ' ]) primary_expression`

Первичное выражение, в качестве которого может использоваться другое выражение в круглых скобках или операнд.

`primary_expression ::= '(' expression ')' | operand`

В качестве операнда могут использоваться элементы языка, доступные для чтения. Идентификатор может соответствовать переменной, объявленной пользователем; константе или параметру оператора последовательности.

`operand ::= identifier | array_element | krug_variable |
krug_variable_attribute | function_call | constant_literal`

В качестве элемента, доступного для записи могут использоваться переменные, объявленные пользователем; элементы массивов; внутренние или системные переменные и их атрибуты.

`lvalue ::= identifier | array_element |
krug_variable | krug_variable_attribute`

Внутренняя или системная переменная с явным указанием ее номера или с косвенным указанием номера через параметр оператора последовательности в квадратных скобках.

`krug_variable ::= krug_variable_type index |
krug_variable_type '[' identifier '']`

Атрибут системной переменной.

`krug_variable_attribute ::= krug_variable '.' A' index`

Тип внутренней или системной переменной.

`krug_variable_type ::=
krug_system_variable_type | krug_internal_variable_type`

Тип системной переменной.

`krug_system_variable_type ::= 'BA' | 'AB' | 'BD' | 'DB' | 'PB'`

Тип внутренней переменной.

`krug_internal_variable_type ::=
'PB' | 'ПЦ' | 'ПЛ' | timer_variable`

Тип таймерной переменной.

`krug_timer_variable_type ::= 'ТЧ' | 'ТМ' | 'ТС'`

Элемент массива. Выражение в квадратных скобках должно возвращать константное значение или являться параметром оператора последовательности.

`array_element ::= identifier '[' expression '']`

Индекс, используемый для задания номера внутренней или системной переменной.

index ::= integer

Числовая константа.

constant_literal ::= signed_integer | real

Строковая константа.

string_literal ::= "" {any_character} ""

Произвольный символ.

any_character ::= <любой печатаемый символ, кроме "">

Вещественное число.

real ::= ((integer '.' integer [exponent]) | (integer exponent))

Экспонента вещественного числа.

exponent ::= 'e' signed_integer

Целое число со знаком.

signed_integer ::= ['- | '+] integer

Целое число без знака, состоящее из одной или более цифр.

integer ::= digit {digit}

Цифра.

digit ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

Идентификатор, состоящий из букв, цифр и символов подчеркивания. Идентификатор не должен начинаться с цифры.

identifier ::=

(letter | ('_' (letter | digit))) {['_'] (letter | digit)}

Буква английского или русского алфавита.

letter ::= en_letter | ru_letter

Буква английского алфавита.

en_letter ::= 'A' | 'B' | <...> | 'Z' | 'a' | 'b' | <...> | 'z'

Буква русского алфавита.

ru_letter ::= 'А' | 'Б' | <...> | 'Я' | 'а' | 'б' | <...> | 'я'

Символ конца строки.

end_of_line ::= '\r\n' | '\n'

Лексический анализатор транслятора должен игнорировать регистр символов текста транслируемой программы.

4 ЯЗЫК ФУНКЦИОНАЛЬНЫХ БЛОЧНЫХ ДИАГРАММ

4.1 Общие сведения

Язык функциональных блочных диаграмм (ФБД) – графический язык. Элементами языка ФБД являются графические символы, которые используются для создания схемы ФБД.

Язык ФБД позволяет программисту строить программу и сложные процедуры, используя существующие функции из поставляемой библиотеки, и связывать их с другими элементами ФБД при помощи специального элемента – связь.

Общий вид блока языка ФБД приведен на рисунке 4.1.1.



Рисунок 4.1.1 – Общий вид блока

На рисунке 4.1.1:

- В верхней части прямоугольника отображается имя блока
- Небольшими квадратами на сторонах прямоугольника обозначены входы (слева) и выходы (справа) блока
- Слева/справа от каждого квадрата отображается название входа/выхода
- С внутренней стороны квадрата указывается тип входа/выхода. Например, <В32> - вещественный 32-битовый, <Ц16> - целый 16-битовый и другие.

Связь между блоками обозначается как линия связи, идущая от какого-либо выхода одного блока к какому-либо входу другого блока (рисунок 4.1.2).

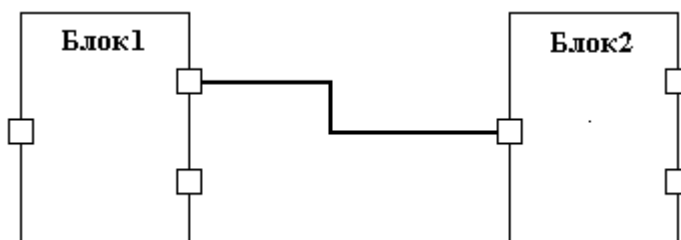


Рисунок 4.1.2 – Общий вид связи между блоками

С одного выхода может исходить несколько линий связи, а на каждый вход может подаваться только одна линия связи.

4.2 Элементы языка

4.2.1 Константа

Константа – блок (рисунок 4.2.1), которому присваивается какое-либо значение пользователя. Это значение подается на вход какого-либо другого блока. Константа всегда имеет только один выход, и может иметь значение следующих типов: вещественный 32 или 64-битовый, целый 18-,8- или 32-битовый, логический, строковый.

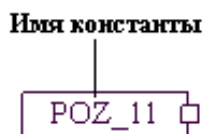


Рисунок 4.2.1 – Блок «Константа»

4.2.2 Выражение СТ

Выражение СТ – блок (рисунок 4.2.2), которому присваивается какое-либо выражение пользователя, написанное на языке СТ.



Рисунок 4.2.2 – Блок «Выражение СТ»

Значение этого выражения подаётся на вход какого-либо другого блока. Блок «Выражение СТ» всегда имеет только один выход, и может иметь значение следующих типов: вещественный 32 или 64-битовый, целый 18-,8- или 32-битовый, логический, неопределенный (определяется на этапе трансляции). При использовании СТ-выражений с выходным типом "целый 32-битовый", для корректной работы с нетипизированными математическими функциями, рекомендуется использовать явное преобразование типов.

4.2.3 Переменная

Переменная – блок (рисунок 4.2.3), который используется для хранения значений вещественного 32- или 64-битового, целого 16-, 8- или 32-битового целого или логического типов в переменных следующего назначения:

- системная переменная – переменная базы данных : ва, ав, вд, дв, рв
- внутренняя переменная (переменная Среды исполнения): пв, пц, пл
- пользовательская переменная: глобальная или локальная.

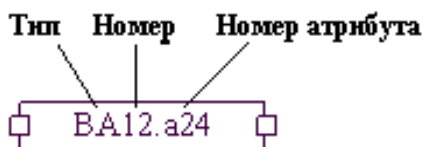


Рисунок 4.2.3 – Блок «Переменная»

Блок «Переменная» всегда имеет один вход и один выход.

Вход блока «Переменная» используется для получения значения переменной с выхода другого блока.

Выход блока «Переменная» используется для передачи значения переменной на вход другого блока.

4.2.4 Массив данных

Массив данных – блок (рисунок 4.2.4), который используется для хранения номеров переменных массива данных.



Рисунок 4.2.4 – Блок «Массив данных»

Размерность массива определяет число шагов цикла, которое будет выполнено соответствующим оператором "Для".

Массив данных не имеет входов и выходов.

4.2.5 Переменная массива данных

Переменная массива данных – блок (рисунок 4.2.5), который используется для хранения значений переменной, связанной с массивом данных.

Блок «Переменная массива данных» в целом аналогичен блоку «Переменная», за исключением следующих особенностей:

- Переменная массива данных может использоваться только внутри какого-либо блока "Для"
- В качестве номера переменной указывается имя массива
- Номер переменной определяется в момент выполнения очередного цикла блока "ДЛЯ" и равен соответствующему значению из массива, который сопоставлен данной переменной.



Рисунок 4.2.5 – Блок «Переменная массива данных»

4.2.6 ЕСЛИ

ЕСЛИ – блок (рисунок 4.2.6), который служит для организации ветвления в программе по условию. Он изображается в виде прямоугольника, разделенного на две части – секции <Если> и <Иначе>. Блок имеет один вход логического типа. Если на вход поступает значение 1, то происходит выполнение элементов схемы ФБД в секции <Если>, если 0 – выполняются элементы схемы ФБД в секции <Иначе>.

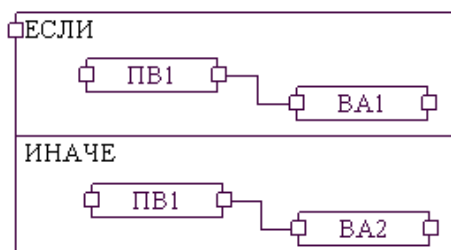


Рисунок 4.2.6 – Блок «Если»

4.2.7 ДЛЯ

ДЛЯ – блок (рисунок 4.2.7), обеспечивающий циклическое выполнение всех элементов схемы ФБД, помещенных в прямоугольник, ограничивающий данный блок.

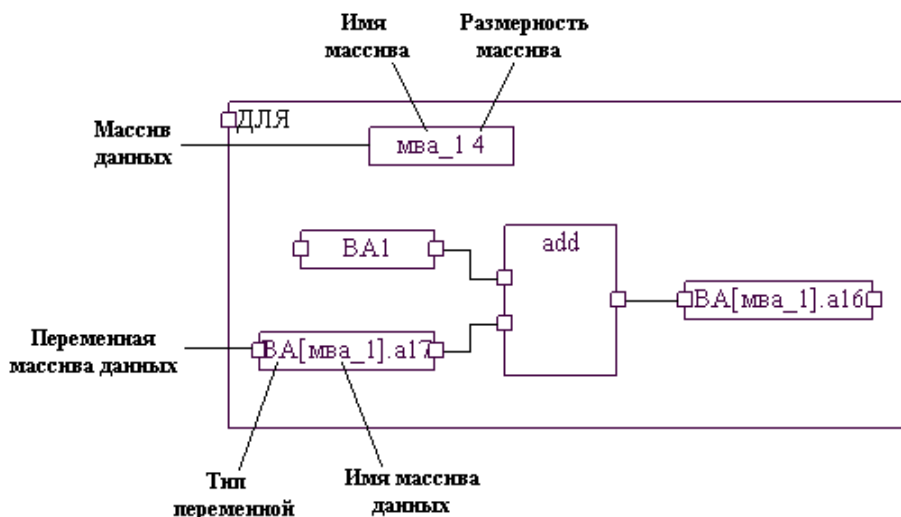


Рисунок 4.2.7 – Блок «Для»

Блок имеет один вход логического типа. Если на вход поступает значение 1, блок выполняется, если 0 - не выполняется.

Каждый блок «ДЛЯ» должен содержать один или более элементов «Массив данных».

Размерность массива определяет число циклов, которое будет выполнено соответствующим оператором «ДЛЯ». Размерность всех массивов данных в пределах данного блока «ДЛЯ» должна быть одинаковой.

4.2.8 Прервать

Прервать – блок (рисунок 4.2.8), который используется для прерывания выполнения цикла блока "Для".

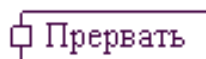


Рисунок 4.2.8 – Блок «Прервать»

Блок имеет один вход логического типа. Если на вход поступает значение 1, выполнение блока «ДЛЯ» прерывается, иначе – не прерывается.

4.2.9 Функция

Функция – это блок (рисунок 4.2.9), реализующий какую-либо функцию.

Данный блок может иметь входы и выходы вещественного 32- или 64 битового, целого 16-, 8- или 32 битового или логического типа. Если тип входа не определен, то это означает, что на данный вход могут поступать значения любого из вышеназванных типов или значение строкового типа.

В случае, если функция не получает значения или не возвращает результаты, то входы/выходы функции могут отсутствовать.

Некоторые функции могут иметь признак наличия номера алгоблока. В этом случае внизу блока «Функция» отображается заданный Пользователем номер алгоблока.

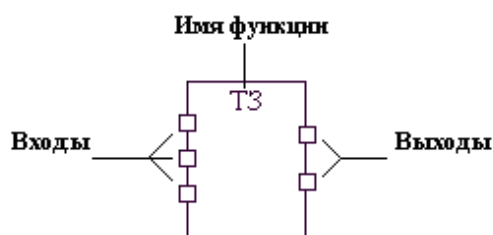


Рисунок 4.2.9 – Блок «Функция»

4.2.10 Процедура

Процедура – это блок (рисунок 4.2.10), который используется для вызова процедуры ФБД или СТ, находящейся во внешнем файле *.dfb или *.lg.

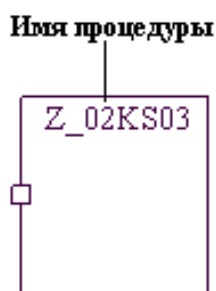


Рисунок 4.2.10 – Блок «Процедура»

Блок имеет один вход логического типа. Если на вход поступает значение 1, происходит вызов процедуры, если 0 - вызова процедура не происходит.

4.2.11 Выход

Выход – это блок (рисунок 4.2.11), который используется для досрочного завершения выполнения программы/процедуры/функции.

Блок имеет один вход логического типа. Выход из программы/процедуры/функции происходит, если на вход поступает значение 1, иначе – выход не происходит.

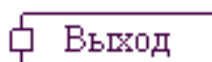


Рисунок 4.2.11 – Выход

4.2.12 Порядок

Порядок – этот блок (рисунок 4.2.12) используется для группировки элементов схемы, порядок выполнения которых не зависит от порядка выполнения других элементов схемы. Блок не имеет входов и выходов.

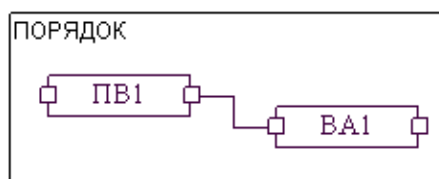


Рисунок 4.2.12 – Порядок

4.2.13 Сообщение

Сообщение – это блок (рисунки 4.2.13 и 4.2.14), который используется для вывода сообщений в протокол событий.

Сообщение имеет один вход логического типа. Если на вход поступает значение 1, то сообщение выводится в протокол событий, если же 0 - то не выводится.

Текст сообщения можно задать двумя способами:

- Путем выбора события из Словаря событий
- Задать в формате (X1 [, X2, ... , Xn]).

В случае формирования текста сообщения из Словаря событий элемент «Сообщение» будет иметь следующий вид (рисунок 4.2.13):

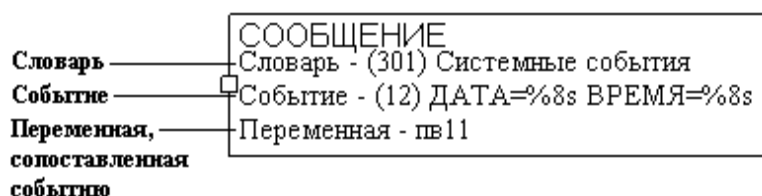


Рисунок 4.2.13 – Сообщение (выбор из Словаря событий)

Вид блока «Сообщение» в случае задания текста сообщения в формате (X1 [,X2, ... ,Xn]) приведен на рисунке 4.2.14.

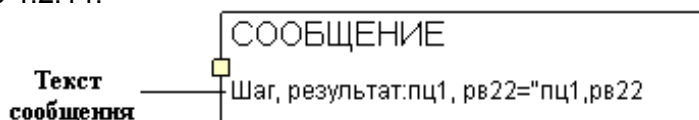


Рисунок 4.2.14 – Сообщение (текст задается Пользователем)

В формате (X1 [,X2, ... ,Xn]) входной параметр X_i может быть:

- Строковой константой, записанной в кавычках (например, "текст сообщения")
- Именем переменной, записанным в виде <Тип><Номер> (например, ва11)
- Обозначением атрибута переменной (например, ва1.а1)
- Числовой константой.

Независимо от вида параметра X_i его значение преобразуется в строку.

4.2.14 Комментарий

Комментарий – это блок (рисунок 4.2.15), содержащий текст пользовательского комментария. Комментарий не привязывается к какому-либо конкретному элементу, а свободно располагается где-либо на схеме ФБД.



Рисунок 4.2.15 – Комментарий

4.2.15 Линия связи

Линия связи – элемент (рисунок 4.2.16), который используется для установки зависимости между входами/выходами элементов ФБД. Изображается в виде линии, идущей от выхода одного элемента к входу другого.

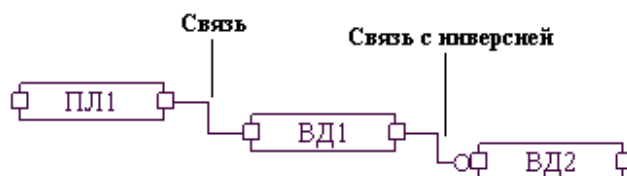


Рисунок 4.2.16 – Линия связи

ВНИМАНИЕ!!!

Для всех платформ, кроме СРВК версии 8.0 и среды исполнения КРУГОЛ версии 2.2, линия связи может соединять выходы-входы только одинакового типа.
Для платформ СРВК версии 8.0 и среды исполнения КРУГОЛ версии 2.2 линия связи может соединять выходы типа меньшей разрядности с входами типов большей или равной разрядности

Если линия связи соединяет контакты логического типа, то она может быть установлена как инверсная – в конце линии ставится небольшой кружок. Инверсия линии связи означает, что в процессе выполнения значение входа элемента, которое передается по данной связи, будет меняться на противоположное, т.е. с 1 на 0 и наоборот.

5 ПРОЕКТ

5.1 Архитектура проекта

Проект языка КРУГОЛ представляет собой набор программ, процедур, функций и алгоблоков Пользователя. Для проекта характерно наличие логической и физической структур.

Логическая структура проекта определяет взаимосвязи (или зависимости) между программами, процедурами, функциями и алгоблоками.

Физическая структура проекта определяет размещение программ, процедур, функций и алгоблоков в файлах проекта.

Программы, процедуры, функции и алгоблоки проекта независимо от их физического размещения находятся в его глобальной области и доступны в любой области видимости проекта.

Интегрированная среда разработки (ИСР) предоставляет пользователю возможность разрабатывать программы, процедуры, функции и алгоблоки с помощью текстового языка структурированного текста (СТ) или графического языка функциональных блочных диаграмм (ФБД).

Каждый файл языка структурированного текста может содержать одновременно несколько программ, процедур, функций или алгоблоков на языке СТ.

Каждый файл ФБД содержит в точности одну программу, процедуру, функцию или алгоблок на языке ФБД.

Основными элементами проекта являются программы. Программы запускаются на выполнение модулем «Ядро КРУГОЛ» Среды исполнения (run-time) контроллера или Среды исполнения (run-time) Станции оператора.

Процедуры, функции и алгоблоки вызываются программами или другими процедурами, функциями и алгоблоками.



ВНИМАНИЕ!!!

Рекурсивные вызовы не допускаются, их отсутствие контролируется транслятором языка КРУГОЛ.

5.2 Окно проекта

Окно проекта предназначено для удобного управления проектом, созданным пользователем.

Управлять показом/скрытием окна проекта на экране можно при помощи команды меню «Вид\Окно проекта».

Окно проекта содержит три вкладки:

- **Проект** – логическая структура проекта (рисунок 5.2.1)
- **Файлы** – физическая структура проекта (рисунок 5.2.2)
- **Словари** – структура данных проекта; (рисунок 5.2.3)

Вкладка «Проект»

Показывает логическую структуру проекта – группы программ, процедур, функций и алгоблоков пользователя, отображаемых в виде дерева проекта.

Открыть элемент проекта для редактирования можно двойным щелчком мыши по имени элемента (рисунок 5.2.1).

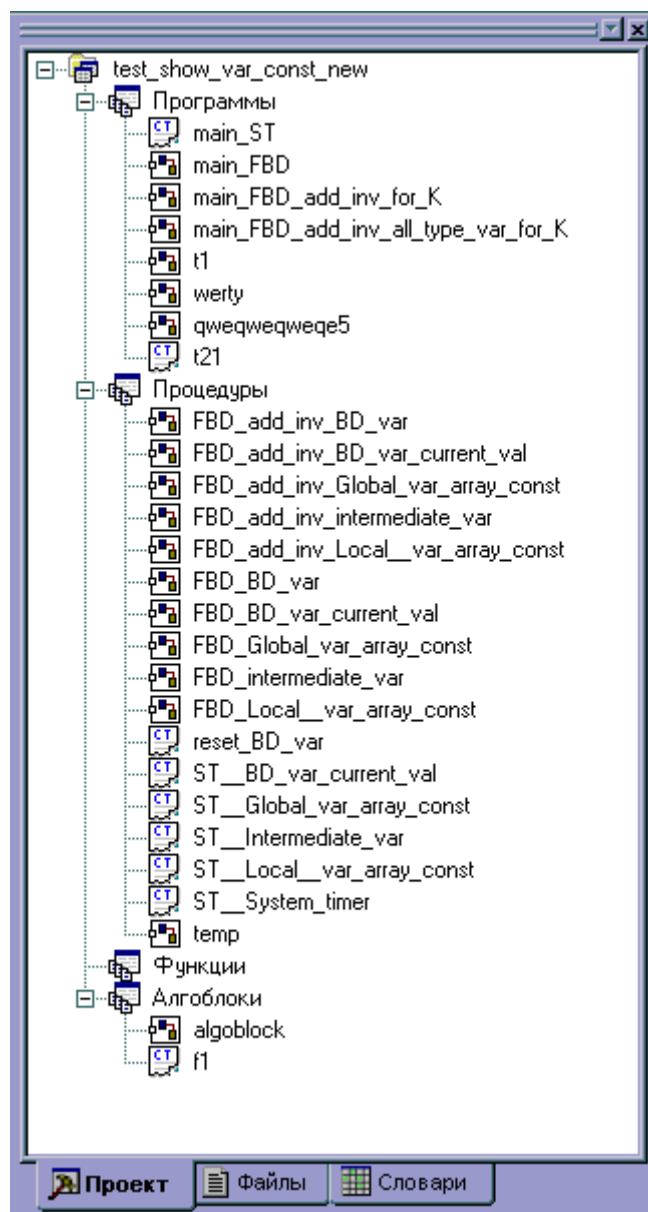


Рисунок 5.2.1 – Окно проекта. Вкладка «Проект»

 ВНИМАНИЕ!!!

Программы в дереве проекта (рисунок 5.2.1, ветвь «Программы») расположены в том порядке, в котором они будут выполняться. Для того, чтобы изменить расположение программ, а, следовательно, и порядок их запуска на выполнение, достаточно нажать левую кнопку мыши на имени требуемой программы и, не отпуская кнопку, перетащить программу на нужную позицию, после чего отпустить кнопку мыши.

Вкладка «Файлы»

Показывает физическую структуру проекта – группы файлов ФБД и СТ, отображаемых в виде дерева.

Двойным щелчком по имени какого-либо файла (рисунок 5.2.2) открывается окно программы, процедуры, алгоблока или функции, которая содержится в данном файле.

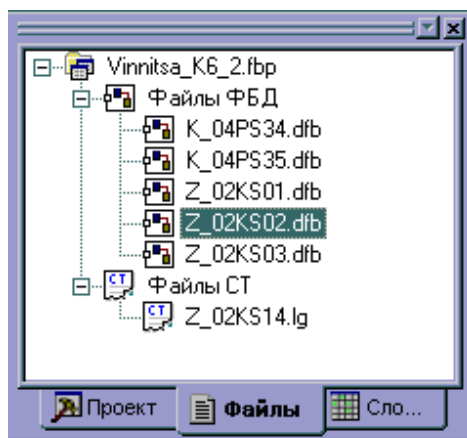


Рисунок 5.2.2 – Окно проекта. Вкладка «Файлы»

Вкладка «Словари»

Отображает структуру данных проекта (рисунок 5.2.3):

- списки глобальных переменных и констант
- для каждой программы и процедуры ФБД – списки локальных переменных и констант
- для каждой функции ФБД – списки локальных переменных и констант, а также входных и выходных параметров
- для каждого алгоблока ФБД – списки локальных переменных и констант, а также входных и выходных параметров

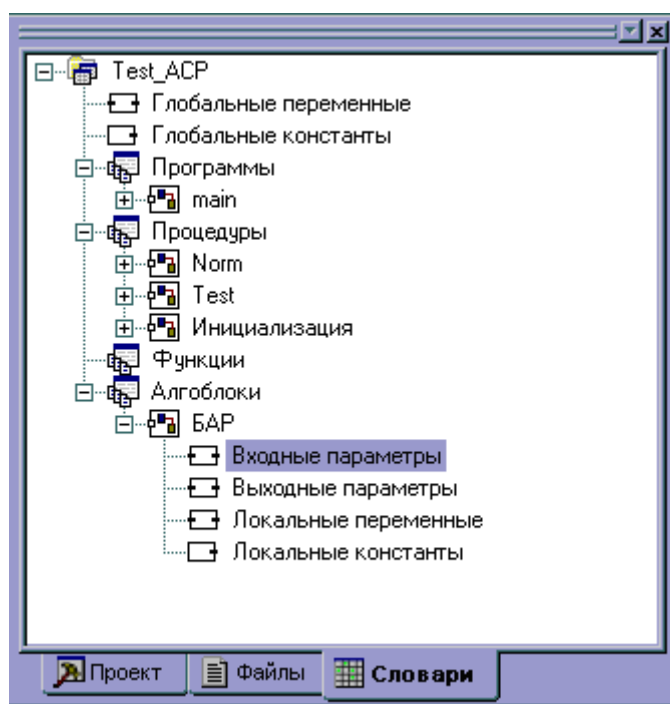


Рисунок 5.2.3 – Окно проекта. Вкладка «Словари»

Двойным щелчком по обозначению какому-либо типа пользовательских данных открывается окно словаря для редактирования соответствующего типа данных.



ВНИМАНИЕ!!!

Пользовательские данные для программ, процедур, функций и алгоблоков на языке СТ на вкладке «Словари» не отображаются, т.к. задаются непосредственно в тексте соответствующего файла СТ.

5.3 Работа с проектом

5.3.1 Создание, открытие и сохранение проекта

Создание нового проекта осуществляется командой «**Новый проект...**» в меню «**Файл**».

Окно «**Создание проекта**» содержит несколько вкладок, приведенных на рисунках 5.3.1 – 5.3.4.

Вкладка «Общие»

На вкладке следует ввести имя создаваемого проекта, путь к каталогу, в котором он будет расположен, и целевую платформу – выбирается из списка доступных платформ.

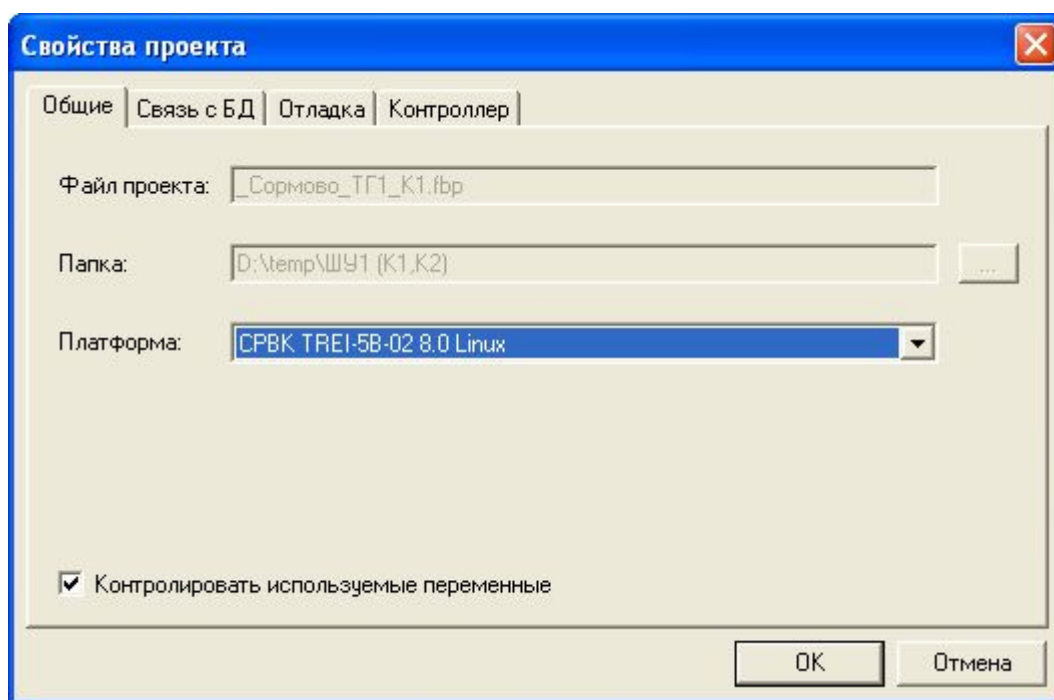


Рисунок 5.3.1 – Окно «Создание проекта ». Вкладка «Общие»

Параметр «**Контролировать используемые переменные**» влияет на режим трансляции проекта. Если данный параметр установлен, то транслятор проверяет наличие в базе данных всех используемых в проекте переменных. В случае отсутствия таковых транслятор выдает сообщение об ошибке и процесс трансляции на этом прекращается.

Вкладка «Связь с БД»

Если проект создается для целевой платформы контроллера, то на вкладке «Связь с БД» (рисунок 5.3.2) следует задать дополнительные параметры:

- **Путь к БД канала** – путь к каталогу, в котором расположена БД контроллера
- **IP-адрес** – задает адрес компьютера, на котором осуществляется отладка программ.

IP-адрес выбирается из выпадающего списка.

Параметр используется при локальной отладке программ, если разрешён обмен данными с БД станции оператора (раздел 5.6.5 «Связь с БД»).

- **Порт** – номер порта, через который сервер БД будет связываться с отладчиком. Если порт указать равным нулю, то сервер БД не сможет связаться с отладчиком. Параметр используется при локальной отладке программ, если разрешён обмен с БД станции оператора.

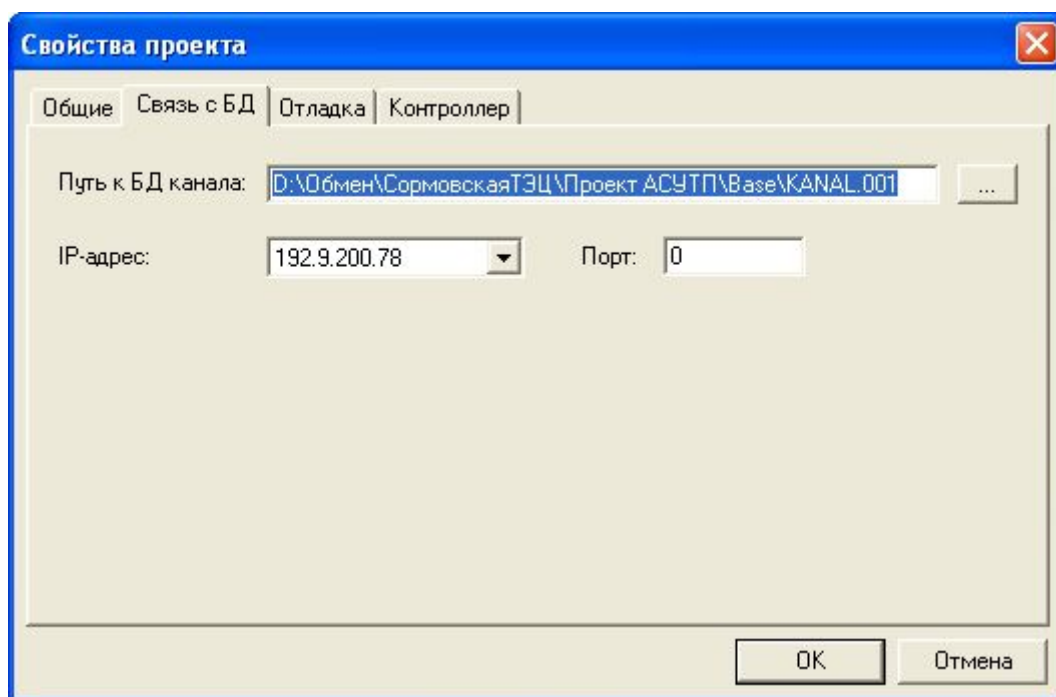


Рисунок 5.3.2 – Окно «Создание проекта». Вкладка «Связь с БД»

Используя значения параметров «IP-адрес» и «Порт», отладчик имитирует контроллер, с которым связывается сервер БД станции оператора.



ВНИМАНИЕ!!!

При работе ИСР в автономном режиме поля ввода IP-адреса и порта недоступны.!

Вкладка «Отладка»

Вкладка «Отладка» (рисунок 5.3.3) служит для задания параметров, влияющих на процесс отладки проекта.

Выпадающий список «**Режим**» позволяет выбрать один из трех режимов выполнения отлаживаемой программы: «**Ручной**», «**Автоматический**» и «**Выполнение в реальном времени**».

При выборе автоматического режима выполнения становятся доступными переключатели «**Выполнение по шагам**» и «**Выполнение по циклам**». В этом случае становятся активны соответствующие им поля ввода «**Время между шагами**» и «**Время между циклами**».

Параметр «**Время между шагами**» определяет временной интервал между окончанием предыдущего шага и началом нового шага. Параметр влияет на автоматическую отладку в пошаговом режиме.

Параметр «**Время между циклами**» определяет временной интервал между окончанием предыдущего цикла и началом нового цикла. Параметр влияет на автоматическую отладку в циклическом режиме.

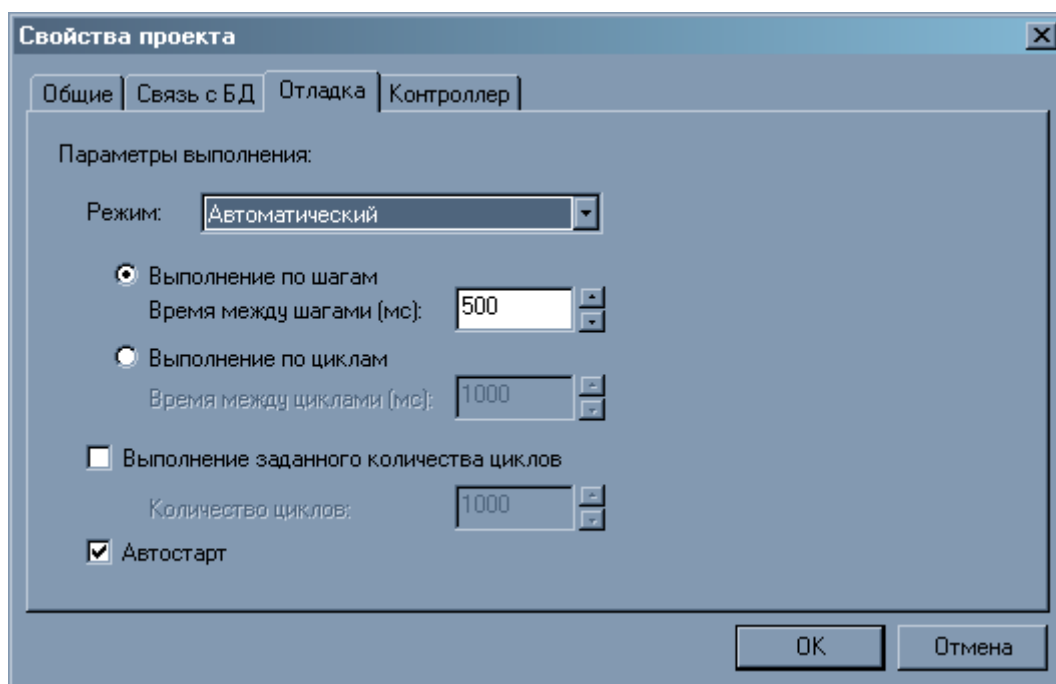


Рисунок 5.3.3 – Окно «Создание проекта». Вкладка «Отладка»

Переключатель **«Выполнение заданного количества циклов»** позволяет выбрать режим выполнения заданного количества циклов, при этом должно становиться доступным соответствующее поле ввода **«Количество циклов»**, которое определяет, сколько будет выполнено циклов в автоматическом режиме до выполнения команды **«Пауза»**. Этот параметр выполняет функцию точки останова, активизирующейся после выполнения заданного количества циклов.

Переключатель **«Автостарт»** позволяет задать режим запуска программы после перехода в отладку. При включенном переключателе после перехода в режим отладки происходит запуск программы, при выключенном переключателе происходит переход отладчика в режим "паузы"

Вкладка «Контроллер»

Вкладка «Контроллер» (рисунок 5.3.4) обеспечивает задание параметров связи с контроллером, таких как IP-адрес и номер порта, используемых для связи с контроллером в режиме удаленной отладки и при программировании контроллера из ИСР КРУГОЛ.

Включением переключателя **«Удаленная отладка»** обеспечивается выбор режима, в котором отладка программ пользователя будет производиться не в локальном отладчике, эмулирующем работу контроллера, а в реальном контроллере. При этом становится доступным выпадающий список **«Режим отладки»** и переключатели **«Инициализация среды исполнения при запуске»** и **«Автоматическое обновление программ»**.

Выпадающий список **«Режим отладки»** позволяет выбрать один из режимов: отладка **«С остановкой контроллера»** и отладка **«Без остановки контроллера»**.

В режиме отладки без остановки на контроллере должна запускаться копия среды исполнения, в которой собственно и будет происходить отладка. При этом входные переменные системной базы данных принимают реальные значения, но выдача управляющих воздействий запрещена, т.е. запись в выходные переменные не влияет на выходные сигналы контроллера, за счет чего не нарушается его нормальная работа.

В режиме отладки с остановкой контроллера копия среды исполнения не запускается, и отладка производится в основной среде исполнения, которая переводится в режим отладки с возможностью выполнения программы пользователя по шагам и по циклам. При этом выдача управляющих воздействий разрешена и при записи значений в выходные переменные будет

происходить формирование соответствующих выходных сигналов контроллера после завершения цикла выполнения программы.

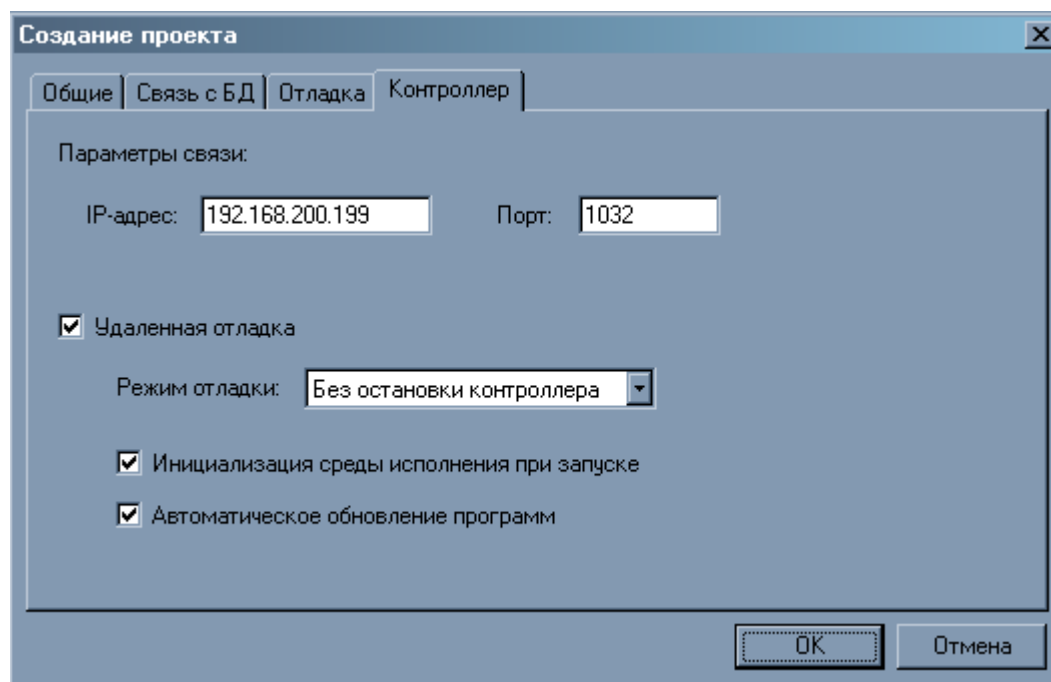


Рисунок 5.3.4 – Окно «Создание проекта». Закладка «Контроллер»

Включение переключателя **«Инициализация среды исполнения при запуске»** обеспечивает сброс состояния среды исполнения контроллера (в которой производится отладка) в исходное состояние при входе в режим отладки (сброс происходит только для внутренних переменных). Выключение данного переключателя оставляет текущее состояние среды исполнения без изменения при входе в режим отладки.

Включение переключателя **«Автоматическое обновление программ»** отключает запрос у пользователя подтверждения о необходимости загрузки программ.



ВНИМАНИЕ!!!

Для режима отладки «Без остановки контроллера» переключатели «Инициализация среды исполнения при запуске» и «Автоматическое обновление программ» всегда включены и их состояние изменить нельзя.

После закрытия окна **«Создание проекта»** в каталоге проекта будет создан файл проекта – файл с именем проекта и расширением ***.fbp**.

В меню «Файл» расположены и другие команды для работы с проектом.

- **«Открыть проект...»** – данная команда предназначена для открытия уже существующих проектов языка КРУГОЛ. При этом на экран выводится стандартный диалог открытия файла, в котором следует указать требуемый файл проекта
- **«Сохранить проект»** – по данной команде производится сохранение файла проекта с учетом всех сделанных изменений
- **«Закрыть проект»** – данная команда закрывает текущий открытый проект
- **«Свойства проекта»** – данная команда позволяет просмотреть свойства текущего открытого проекта и, при необходимости, изменить их. При этом на экран будет выведен окно **«Свойства проекта»** (рисунок 5.3.5). Это то же самое окно, которое используется для создания проекта.

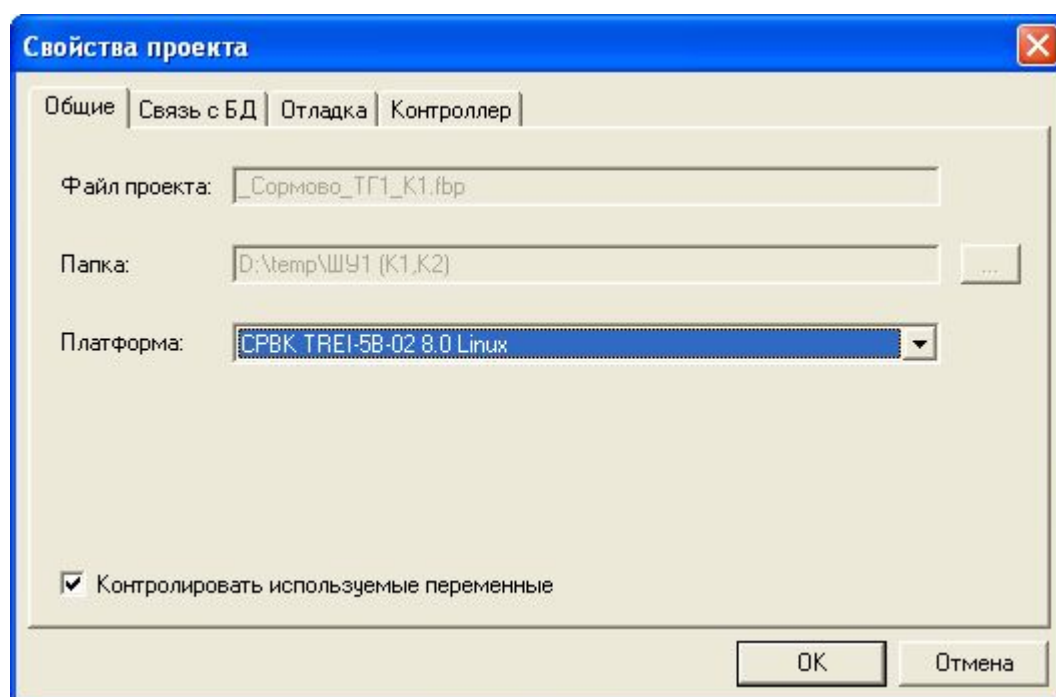


Рисунок 5.3.5 – Окно «Свойства проекта»

ВНИМАНИЕ!!!

Если ИСР функционирует в автономном режиме, то при попытке открыть проект, созданный для платформ "верхнего" уровня, будет выдано предупреждение о невозможности работы с проектом для данной платформы и процесс открытия проекта будет прерван

5.3.2 Создание, открытие и сохранение файлов проекта

Создание нового файла в проекте производится при помощи команды **«Новый...»** из меню **«Файл»**. При этом на экран будет выведено окно **«Создать»** (рисунок 5.3.6).

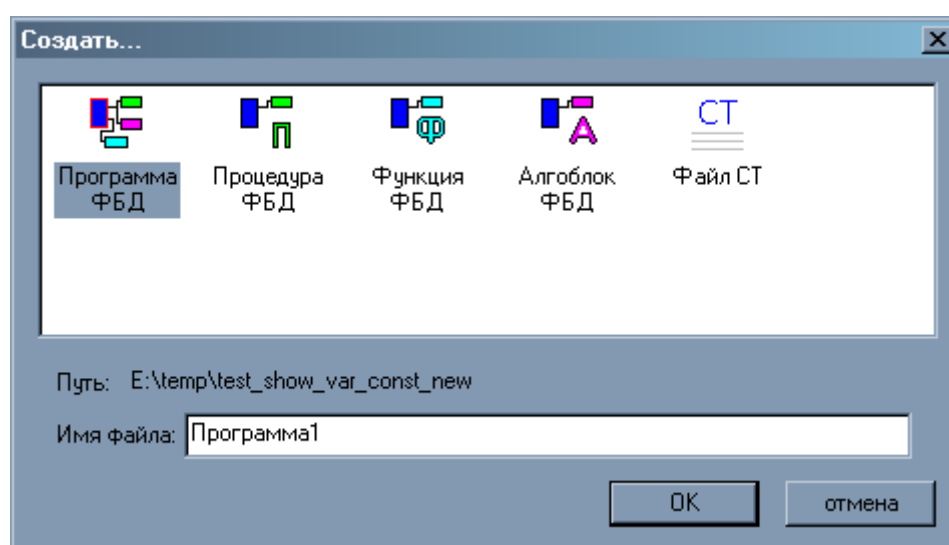


Рисунок 5.3.6 – Окно «Создать»

В данном окне следует выбрать требуемый тип файла и в поле **«Имя файла»** ввести необходимое имя.

ВНИМАНИЕ!!!

ПРАВИЛА НАИМЕНОВАНИЯ ПРОГРАММ, ПРОЦЕДУР И ФУНКЦИЙ:

- a) Имя может содержать буквы латинского и русского алфавитов, цифры и символ подчеркивания
- b) Имя всегда начинается с буквы латинского или русского алфавита
- c) Длина имени – не более 79 символов.

В меню «Файл» расположены также и другие команды для работы с файлами:

- «Открыть...» – данная команда предназначена для открытия уже существующих файлов проекта. При этом на экран выводится окно «Открыть» (рисунок 5.3.7), в котором следует выбрать требуемые файлы.

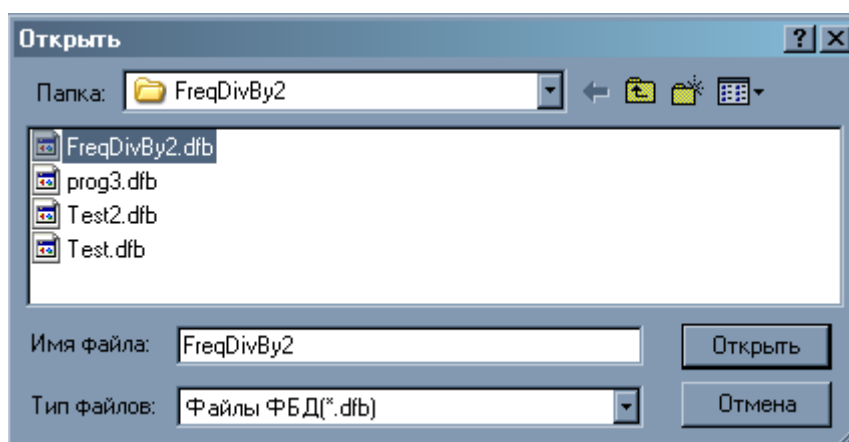


Рисунок 5.3.7 – Окно «Открыть»

Список выбора «Тип файла» позволяет задать тип файлов проекта – файлы ФБД или файлы СТ.

- «Сохранить» – данная команда сохраняет текущий открытый файл проекта.
- «Сохранить как...» – данная команда позволяет сохранить текущий открытый файл ФБД или СТ под другим именем. При выборе команды выводится диалоговое окно «Сохранить как» (рисунок 5.3.8)

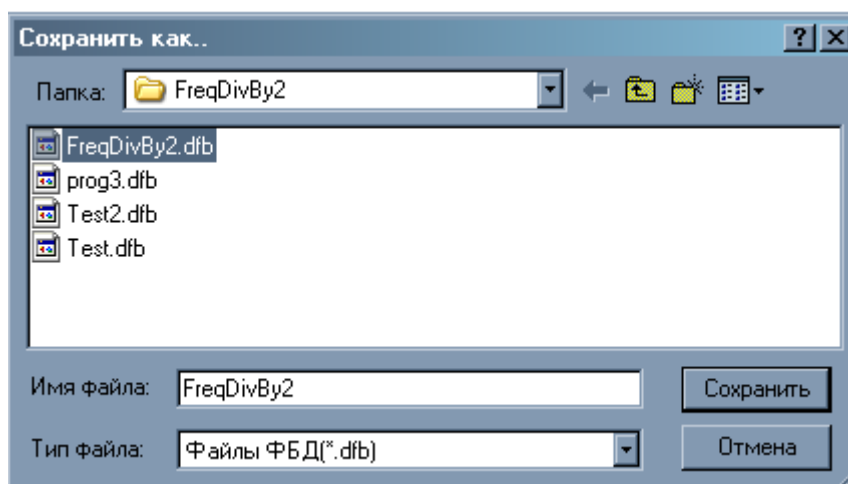


Рисунок 5.3.8 – Окно «Сохранить как»

В поле «Имя файла» следует задать новое имя текущего файла. При этом программа, процедура, функция или алгоблок, сохраненные в файле с новым именем, не будут

автоматически включены в состав проекта. Для включения их в проект необходимо воспользоваться командой **«Открыть»**.

- **«Сохранить все»** – команда сохраняет все открытые файлы проекта.
- **«Закрыть»** – команда закрывает текущий открытый файл.

5.3.3 Создание пользовательских данных

Среда разработки поддерживает возможность создания констант, переменных, параметров функций для использования их в дальнейшем в качестве обычных элементов ФБД.

Локальные пользовательские данные – это данные, область видимости которых ограничена схемой ФБД, для которой они созданы. В этом случае:

- Переменные могут использоваться в схеме ФБД в качестве обычного элемента схемы типа «Переменная»
- Константы могут использоваться в схеме ФБД в качестве обычного элемента схемы типа «Константа»
- Параметры функции могут использоваться в схеме ФБД в качестве элемента схемы типа «Параметр функции». При этом параметры функции доступны только для схем ФБД, являющихся пользовательскими функциями.

Глобальные пользовательские данные – это данные, область видимости которых весь проект, т.е. их можно использовать в любой программе, процедуре, функции или алгоблоке ФБД или СТ проекта. В этом случае:

- В схеме ФБД они используются как обычные элементы типа «Переменная» или «Константа»
- В программе, процедуре, функции или алгоблоке СТ - в качестве обычных операндов типов «Переменная» или «Константа».

Создание и редактирование глобальных и локальных пользовательских данных в целом идентично и осуществляется с помощью словаря соответствующего типа данных. Ниже приводится пример создания списка глобальных переменных.

Для того, чтобы открыть словарь глобальных переменных, следует сделать двойной щелчок мышью по обозначению **«Глобальные переменные»** на вкладке **«Словари»** окна проекта, после чего откроется окно словаря (рисунок 5.3.9).



№	Имя	Тип	Значение	Размер	Комментарий
1	ARG_TEMP	Вещ	96.8	0	
2	ARRAY_TEMP	Вещ		100	
3	POZ_11	Вещ	11.0	0	
4	POZ_12	Вещ	12.0	0	
5	STOP	Лог	0	0	Остановка подачи

Рисунок 5.3.9 – Окно «Глобальные переменные»

Списки глобальных переменных отображаются в табличном формате. Таблица состоит из шести полей:

- **«№»** – порядковый номер записи
- **«Имя»** – имя переменной

- «**Тип**» – тип значения переменной
- «**Значение**» – значение переменной
- «**Размер**» – если значение этого параметра больше нуля, то создаваемая переменная на самом деле будет являться массивом с размерностью, заданной в данном поле, а поле «Значение» будет игнорироваться
- «**Комментарий**» – используется для задания краткого комментария к переменной.

Для работы со словарем используется контекстное меню, появляющееся при нажатии в окне словаря правой кнопки мыши:

- «**Добавить строки**» – создание новой переменной. В таблицу будет добавлена новая пустая строка, в которой можно задать параметры создаваемой переменной
- «**Удалить строки**» – удаление какой-либо переменной. Для удаления следует выделить строку, соответствующую удаляемой переменной, и выбрать данный пункт контекстного меню

Для ввода данных следует выделить требуемую ячейку и нажать на клавишу «**Ввод/Enter**» после чего ячейка переходит в режим редактирования; повторное нажатие «**Ввод/Enter**» приводит к выходу из режима редактирования с сохранением сделанных изменений. Клавиша «**Esc**» – выход из режима редактирования с отменой сделанных изменений.

При работе в окне «Глобальные переменные» поддерживаются стандартные функции копирования в буфер обмена Windows и вставки данных из буфера, отмена последних сделанных изменений (при помощи команды «Правка\Отменить»), а также функция контроля за корректностью вводимых данных (например, не допускается ввод букв или отрицательных чисел в поле «Размер»).

5.3.4 Печать исходных текстов программ

Созданные файлы ФБД и СТ можно распечатать на принтере. Для этого используется команда «**Печать...**» меню «**Файл**».

Для выбора принтера и управления печатью используется окно «**Печать**» (рисунок 5.3.10).

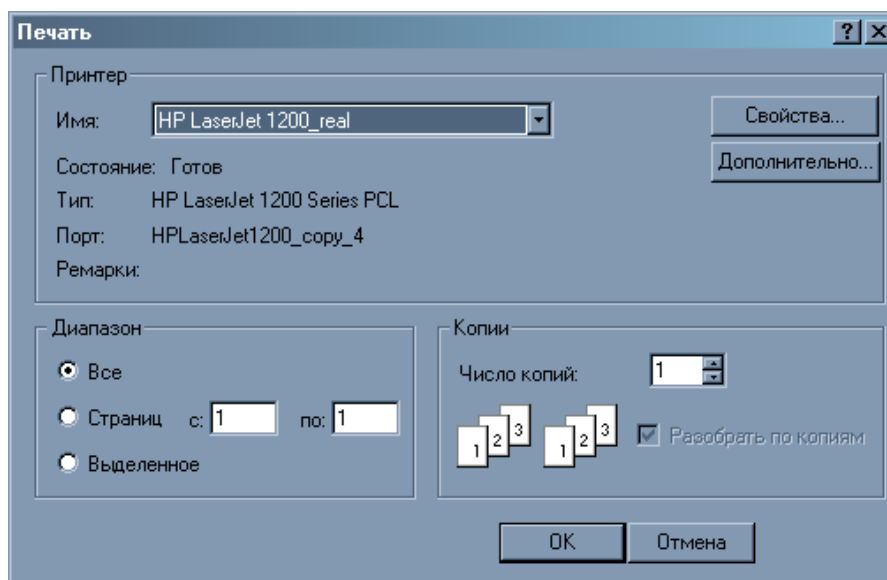


Рисунок 5.3.10 – Окно «Печать»

5.3.5 Настройка параметров печати

Окно **«Дополнительно»** позволяет задать дополнительные параметры нумерации страниц при печати и организовать печать нескольких файлов в порядке, определяемом пользователем.

Для доступа к дополнительным параметрам печати необходимо в окне **«Печать»** (рисунок 5.3.10) нажать на кнопку **«Дополнительно»**.

5.3.6 Формирование списка файлов для печати

Для формирования списка файлов для печати необходимо воспользоваться вкладкой **«Файлы»** (рисунок 5.3.11) в окне **«Дополнительно»**.

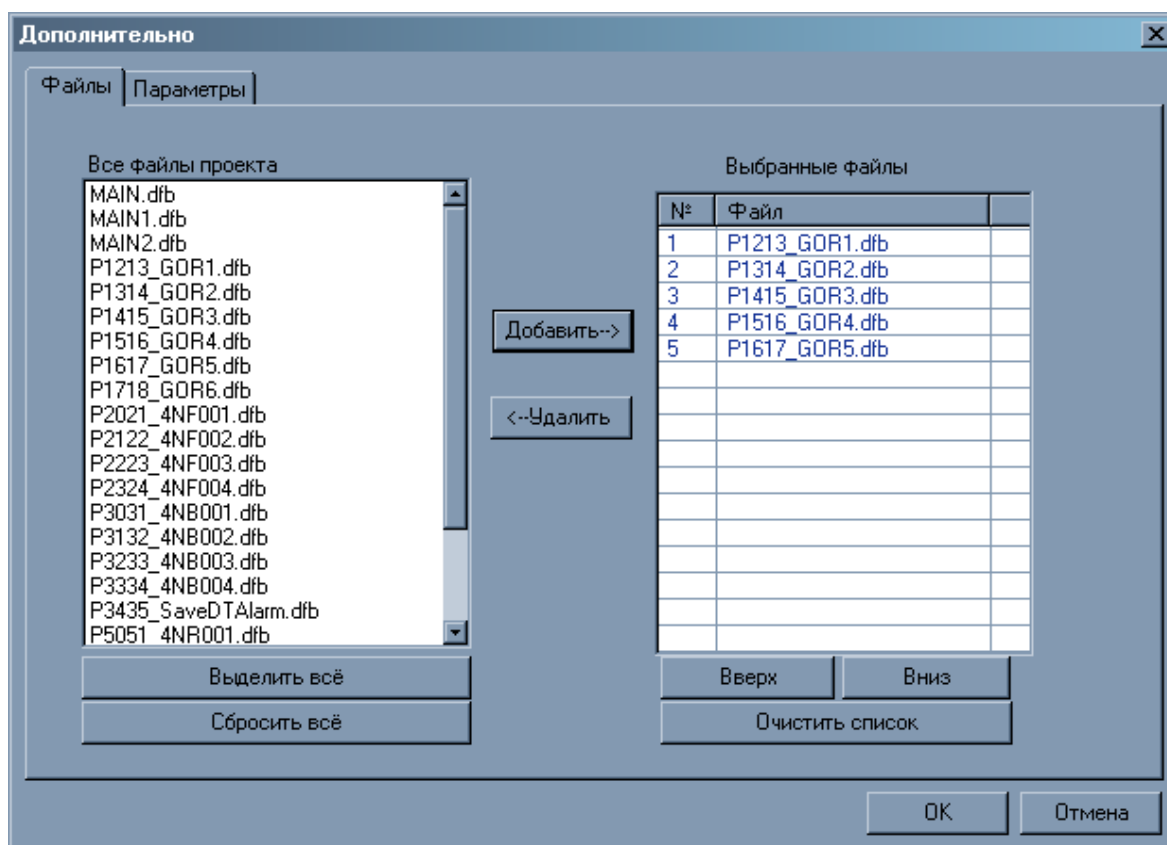


Рисунок 5.3.11 – Окно «Дополнительно». Формирование списка файлов для печати

В списке **«Все файлы проекта»** содержатся имена файлов, содержащиеся в проекте

В списке **«Выбранные файлы»** содержатся имена файлов, который пользователь предполагает вывести на печать.

Управление содержимым списков осуществляется при помощи следующих кнопок:

- **«Добавить»** – добавляет выделенные имена файлов в список файлов проекта в конец списка выбранных файлов
- **«Удалить»** – удаляет выделенный файл из списка «Выбранные файлы»
- **«Выделить всё»** – выделяет все файлы в списке файлов проекта
- **«Сбросить всё»** – снимает выделение файлов в списке «Все файлы проекта»
- **«Вверх»** – перемещает выделенный файл в списке «Выбранные файлы» на одну позицию вверх
- **«Вниз»** – перемещает выделенный файл в списке «Выбранные файлы» на одну позицию вниз
- **«Очистить список»** – удаляет все файлы из списка «Выбранные файлы»

ВНИМАНИЕ!!!

Если список «Выбранные файлы» пуст, то производится печать текущего активного документа

5.3.7 Настройка нумерации страниц при печати

Для выполнения настройки параметров нумерации страниц необходимо перейти на вкладку «**Параметры**» (рисунок 5.3.12) в окне дополнительных параметров печати.

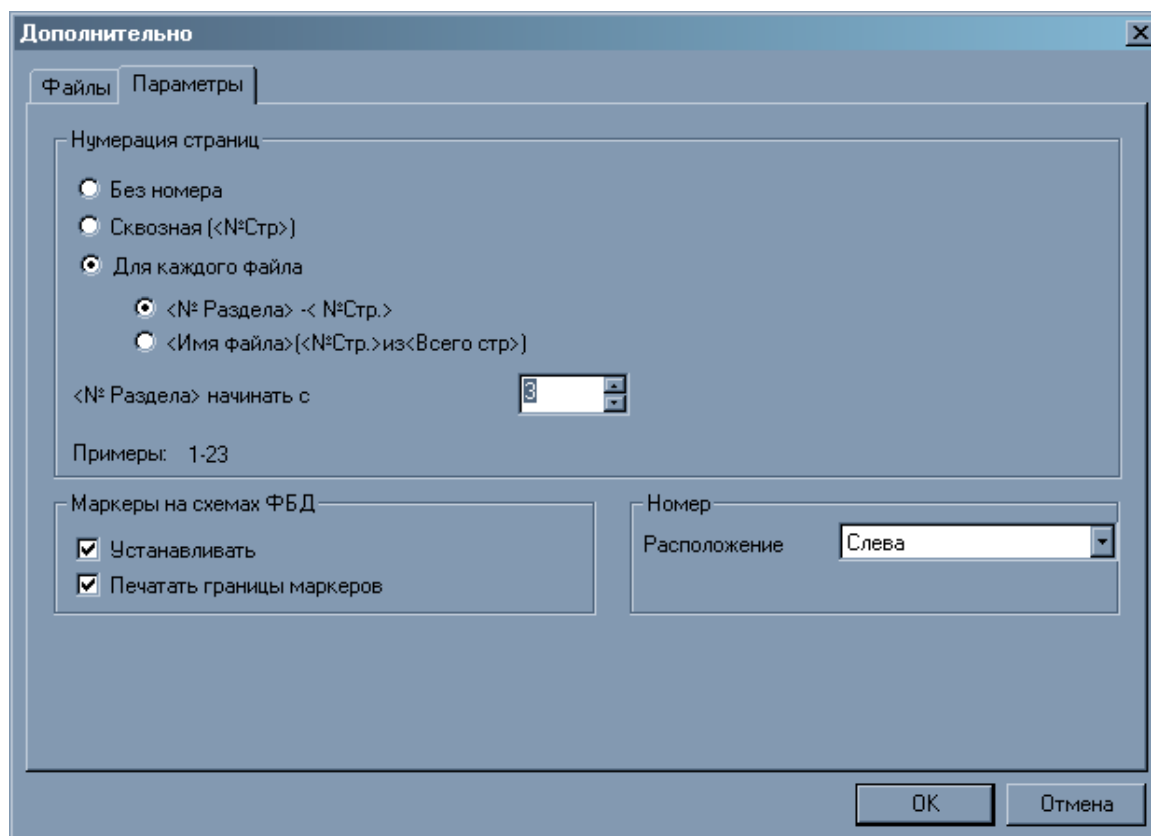


Рисунок 5.3.12 – Настройка параметров нумерации страниц

Данная вкладка позволяет задать следующие режимы нумерации страниц:

- **Без номера**
- **Сквозная**
- **Нумерация страниц для каждого файла**

Если выбрана нумерация страниц для каждого файла, то необходимо задать формат номера страницы. Возможен выбор одного из следующих форматов номера страниц:

- <№ Раздела>-<№ Стр>.
- <Имя файла> (<№ Стр> из <Всего страниц>)

Возможна также настройка начального значения номера страницы при сквозной нумерации или номера раздела в режиме нумерации для каждого файла.

Для наглядности в поле «**Примеры**» выводятся примеры нумерации страниц.

ВНИМАНИЕ!!!

В случае, когда пользователь выбрал только один файл для печати или когда список выбранных файлов пуст, режим нумерации страниц «Для каждого файла не доступен»

На вкладке «Параметры» возможно осуществить настройку положения номера страницы на листе. Для этого необходимо воспользоваться раскрывающимся списком «**Расположение**», который предоставляет следующие варианты выбора расположения номера страницы:

- **Слева** – номер располагается по левому полю страницы
- **Справа** – номер располагается по правому полю страницы
- **Сверху** – номер располагается по верхнему полю страницы
- **Снизу** – номер располагается по нижнему полю страницы
- **По короткой стороне** – номер располагается либо снизу при книжной ориентации листа, либо справа при альбомной ориентации
- **По длинной стороне** – номер располагается либо справа при книжной ориентации листа, либо снизу при альбомной ориентации листа.

Вкладка «Параметры» позволяет также настроить отображение маркеров перехода.

Маркер перехода – это указатель, показывающий с какого листа и на какой лист идёт линия связи на схеме ФБД. Изображение маркера перехода приведено на рисунке 5.3.13

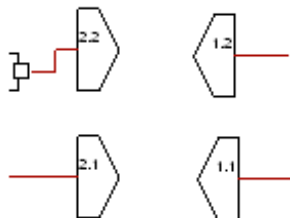


Рисунок 5.3.13 – Маркеры перехода

Формат номера маркера перехода зависит от параметров нумерации страниц.

В случае, когда выбран режим нумерации «**Без номера**», маркеры нумеруются числами, начиная с единицы.

Когда выбран режим нумерации «**Сквозная**» или «**Для каждого файла**» номер маркера имеет следующий формат:

<№ страницы>. <№ маркера>

На вкладке «Параметры» с помощью переключателей в группе «**Маркеры на схемах ФБД**» следует задать режим отображения маркеров перехода.

- Показывать/не показывать маркеры
- Показывать/не показывать границы маркеров

ВНИМАНИЕ!!!

Для корректной работы функции печати необходимо использовать драйвер с поддержкой PCL (Printer Common Language).

При использовании других драйверов правильность работы функции не гарантируется.

Параметры страницы (ориентация, размер и отступы по краям листа) устанавливаются в окне «**Параметры страницы**» (рисунок 5.3.14), которое вызывается по команде «**Параметры страницы...**» меню «**Файл**»).

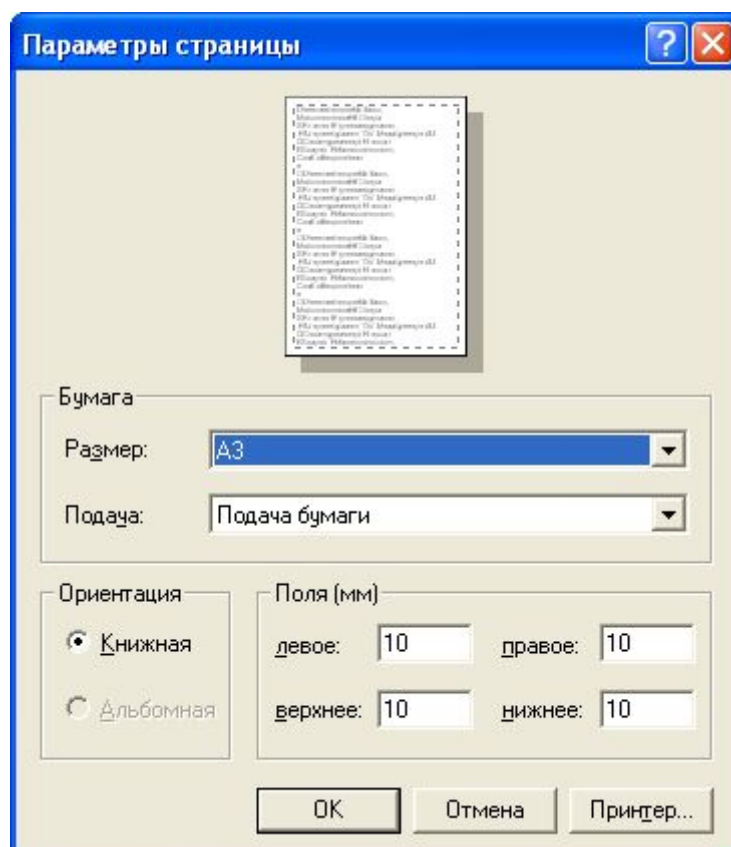


Рисунок 5.3.14 – Окно «Параметры страницы»

Оценить, как будет выглядеть схема ФБД или текст СТ в "бумажном" варианте можно по команде «Предварительный просмотр» меню «Файл» (пример на рисунке 5.3.15).

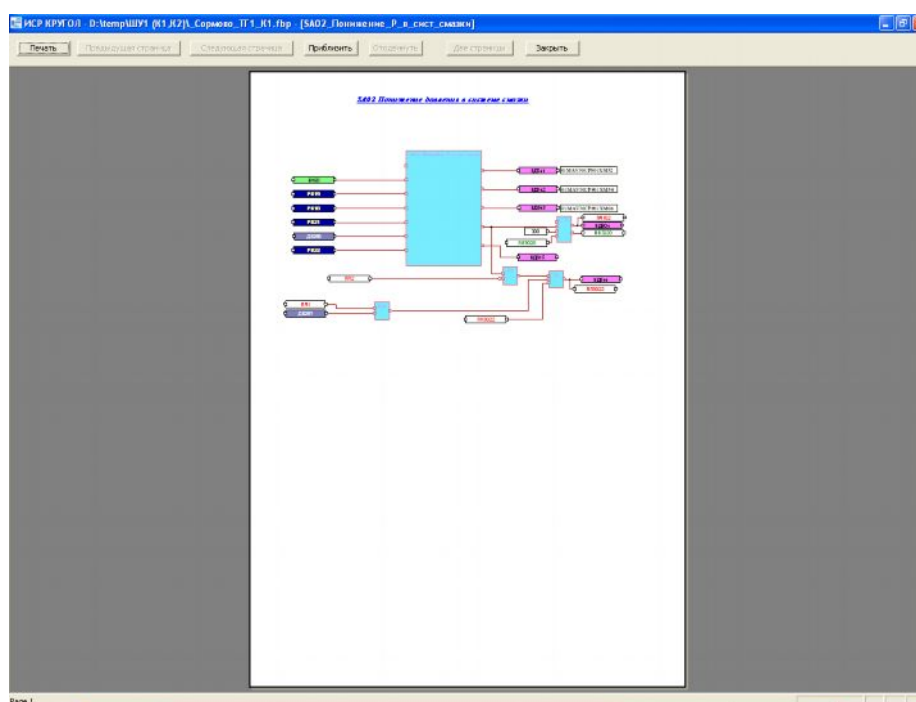


Рисунок 5.3.15 – Окно предварительного просмотра исходной программы КРУГОЛ

5.4 Редактирование файлов проекта

5.4.1 Редактирование файлов СТ

5.4.1.1 О редакторе СТ

Редактор файлов СТ (рисунок 5.4.1) представляет собой текстовый редактор с стандартными для Windows-приложения возможностями по созданию и редактированию текста. В частности, поддерживаются функции копирования (главное меню «**Правка\Копировать**») в буфер обмена и вставка («**Правка\Вставить**») из буфера обмена фрагментов текста; отмена сделанных изменений («**Правка\Отменить**») и возврат отмененных действий («**Правка\Вернуть**»).

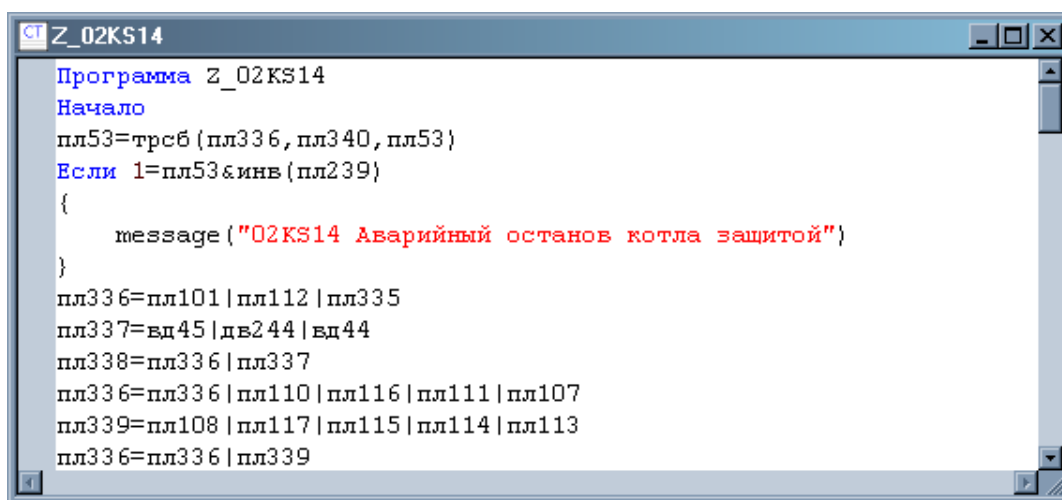


Рисунок 5.4.1 – Редактор СТ

5.4.1.2 Поиск и замена

В редакторе СТ реализованы функции поиска и замены заданных фрагментов текста.

Поиск текста

Поиск фрагмента текста производится при помощи команды «**Найти**» в меню «**Правка**» с выводом на экран соответствующего диалога (рисунок 5.4.2).

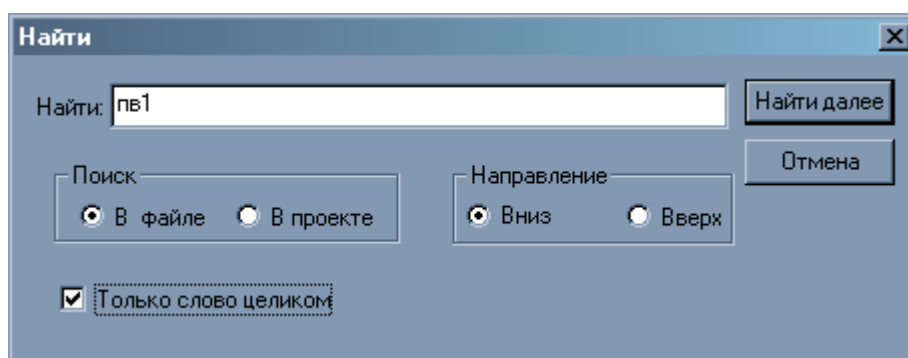


Рисунок 5.4.2 – Окно поиска

Для поиска Пользователь должен в поле «**Найти**» ввести текст, выбрать местоположение («**В файле**»/ «**В проекте**») и направление («**Вниз**»/ «**Вверх**»), а затем нажать клавишу «**Найти далее**». Поиск введенной строки начинается с позиции, в которой находится курсор в окне программы.

В случае установки уточняющего параметра «**Только слово целиком**» (рисунок 5.4.2) подразумевается, что искомое слово отделено от всех остальных пробелами. По умолчанию

этот параметр не установлен и, в общем случае, поиск производится по всему тексту, без учета разделения слов пробелами и регистра букв.

Если в тексте найдется хотя бы одно слово, соответствующее введенному образцу, поиск прервется, а найденный фрагмент будет выделен.

В меню «Правка» есть команда «Найти далее», позволяющая продолжать поиск заданного фрагмента текста без вывода окна диалога.

Замена текста

Замена искомого фрагмента текста на заданный производится при помощи команды «Изменить...» меню «Правка» с выводом на экран соответствующего диалога (рисунок 5.4.3).

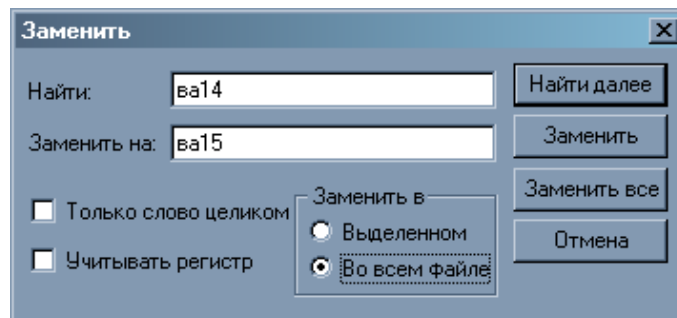


Рисунок 5.4.3 - Окно замены фрагмента текста

Команда «Заменить» идентична команде «Найти» и может работать также в режиме простого поиска, для этого служит кнопка «Найти далее».

Кнопку «Заменить» следует применять в случае, когда необходимо произвести множество одинаковых операций по редактированию программы, например, в случае изменения номера какой-либо переменной. Для этого Пользователь должен ввести в поле «Заменить на» новую строку, которая заменит строку в поле «Найти», а после этого нажать кнопку «Заменить». В случае если Пользователь полностью уверен в своих действиях, следует нажать клавишу «Заменить все». В этом случае процедура замены не будет останавливаться на каждом новом слове, и ждать подтверждения Пользователя, все замены произойдут автоматически.

Замену фрагмента можно производить как во всем файле, так и в пределах выделенного фрагмента текста.

5.4.1.3 Работа с точками останова

Редактор СТ поддерживает возможность работы с точками останова.

Точка останова – эта «метка» на строке программы (рисунок 5.4.4), которая означает, что выполнение программы в режиме отладки будет приостановлено на этой строке и продолжено по команде Пользователя.

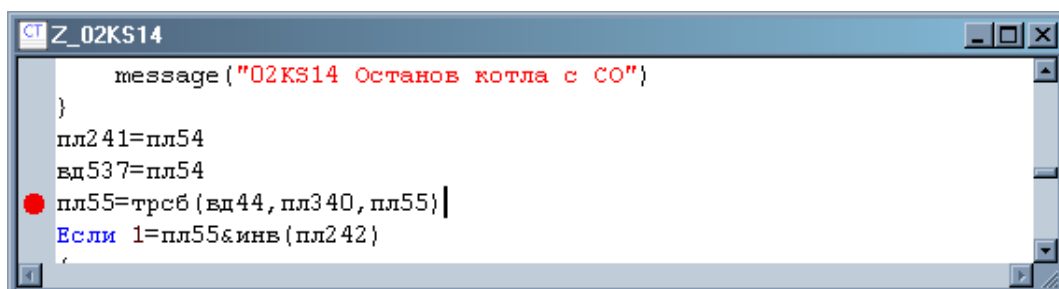


Рисунок 5.4.4 – Точка останова

Работа с точками останова осуществляется по командам **«Точки останова»** меню **«Отладка»**.

Если требуется установить точку останова на какую-либо строку текста программы, следует, предварительно поставив курсор на эту строку и выбрать в меню команду **«Установить точку останова»**. Если же на данной строке уже установлена точка останова, то она, наоборот, удалится. То есть одна и та же команда позволяет устанавливать и снимать точки останова.

Команда **«Снять все точки останова»** удаляет все точки останова во всех программах проекта.

Точку останова можно временно отключить – выполнение программы на этой строке приостанавливаться не будет, хотя сама точка останова будет видна – просто ее цвет поменяется на другой (рисунок 5.4.5).

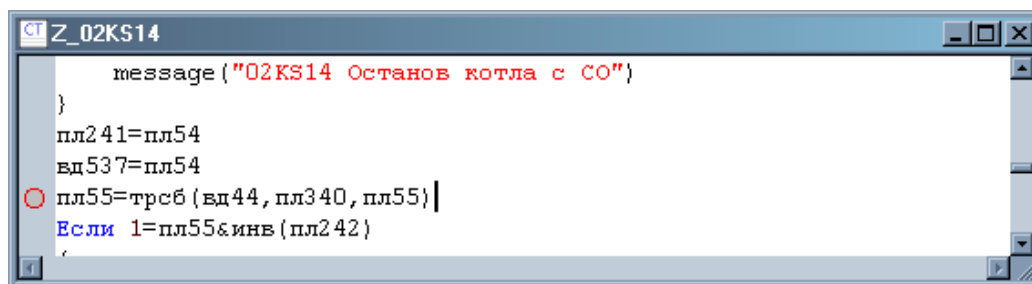


Рисунок 5.4.5 – Точка останова временно отключена

Отключение производится командами **«Отключить точки останова»** (курсор должен находиться на соответствующей строке) и **«Отключить все точки останова»** – в этом случае будут отключены все точки останова во всех программах проекта.

В любой момент отключенные точки останова можно активизировать, выбрав команду **«Установить точку останова»** (курсор должен находиться на соответствующей строке).

Работа с точками останова возможна как в режиме редактирования, так и в режиме отладки.

5.4.1.4 Работа с закладками

Редактор СТ поддерживает возможность работы с закладками.

Закладка – эта «метка» на строке открытого файла СТ, предназначенная для использования в качестве ссылки. Закладки используются для быстрого перехода в определенное место файла. В одном файле можно установить сразу несколько закладок.

Работа с закладками осуществляется с помощью команд, расположенных в меню **«Правка\Закладки\...»**.

Если требуется установить закладку на какую-либо строку текста программы, процедуры, функции или алгоблока, следует, предварительно поставив курсор на эту строку, выбрать команду меню **«Установить закладку»** – слева от начала строки появится символ закладки (небольшой кружок, рисунок 5.4.6).

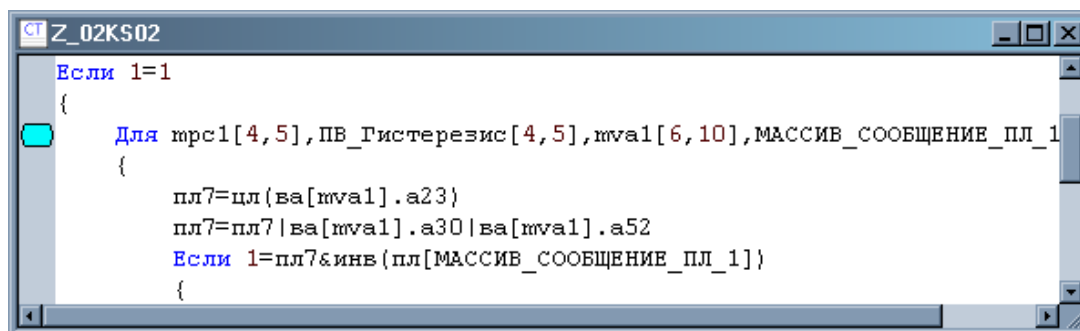


Рисунок 5.4.6 – Закладка

Если же на данной строке уже установлена закладка, то она, наоборот, удалится. То есть одна и та же команда позволяет устанавливать и снимать закладки.

Команда **«Снять все закладки»** удаляет все закладки в данном файле.

Для перехода к следующей закладке (вниз от строки, на которой находится курсор) используется команда **«Перейти на следующую»**, а для перехода к предыдущей закладке (вверх от строки с курсором) – команда **«Перейти на предыдущую»**. При этом окно файла СТ будет перемещаться на строку с искомой закладкой.

5.4.1.5 Сервисные функции

Для более удобного редактирования текста программ, процедур, функций или алгоблоков в среде СТ реализованы следующие сервисные функции:

- **Подсветка синтаксических элементов языка СТ.** Некоторые конструкции языка СТ выделяются (подсвечиваются) различными цветами для улучшения восприятия текста программы. К ним, в частности, относятся ключевые слова языка СТ (**Процедура**, **Начало**, **Для** и другие), числовые конструкции (**110**), строковые константы (**"строка"**) и другие элементы (рисунок 5.4.7).

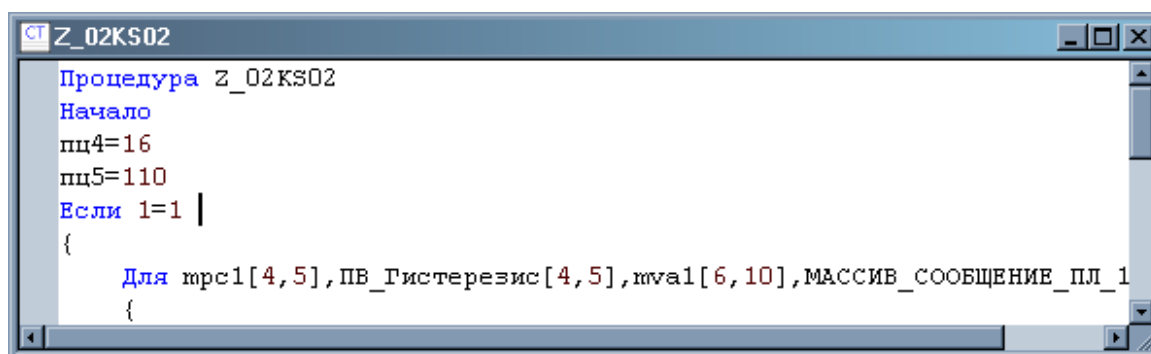


Рисунок 5.4.7 – Окно редактора СТ

- **Поиск парных скобок.** Синтаксис языка СТ включает в себя понятие парных скобок «{» и «}», ограничивающих составные конструкции операторов. Иногда скобки могут отстоять друг от друга на довольно значительные расстояния и, к тому же, быть вложенными. Для того чтобы легче было определять области действия операторов, можно использовать механизм поиска парных скобок. Достаточно поместить курсор непосредственно рядом со скобкой «{» и нажать клавиши <Ctrl>+<>> – курсор автоматически переместится на скобку, парную данной.
- **Вставка параметров функции «message2».** Функция «message2» требует в качестве параметров «код словаря событий» и «код события». С целью упрощения ввода данных параметров предусмотрена команда **«Словарь событий»** меню **«Правка»**... Команда выводит окно **«Выбор события»** (описание в разделе «Редактирование файлов ФБД\Вставка элементов»). После выбора требуемых словаря и события соответствующие им код словаря и код события будут вставлены в текст программы, процедуры, функции или алгоблока СТ с позиции курсора, который следует поместить в место описания параметров.

5.4.2 Редактирование файлов ФБД

5.4.2.1 Вставка элементов

Вставка новых элементов в схему ФБД производится при помощи команд меню «**Правка\Вставка элементов\...**».

Возможны следующие варианты вставки:

- Вставка с настройкой свойств элемента. На экране появляется диалог для задания свойств данного элемента. В этом случае следует выбрать требуемые свойства, после чего закрыть диалог с подтверждением сделанных изменений. В этом случае курсор мыши примет форму, соответствующую типу вставляемого элемента.
- Вставка без задания свойств элемента. Курсор мыши сразу примет форму, соответствующую типу вставляемого элемента. Это означает, что для элемента данного типа не требуется предварительного задания свойств перед вставкой в схему.

После того, как курсор примет форму, соответствующую типу вставляемого элемента, следует щелкнуть мышью в свободной области схемы (т.е. не занятой каким-либо другим элементом схемы) и выбранный элемент вставится в схему. Если же вставляемый элемент "попадает" на существующий другой элемент схемы, то вставки выбранного элемента не произойдет, а курсор примет форму редактирования.

В дальнейшем свойства созданного элемента можно изменить. Это можно сделать с помощью двойного щелчка мышью по элементу – на экран будет выведен тот же диалог, что и при создании данного элемента, в котором можно поменять свойства элемента – либо выделить элемент и выбрать команду «**Изменить...**» из меню "**Правка**".

Вставка функции

Вставка функции производится командой «**Список функций...**». На экран выводится диалоговое окно «Список функций» (рисунок 5.4.8).

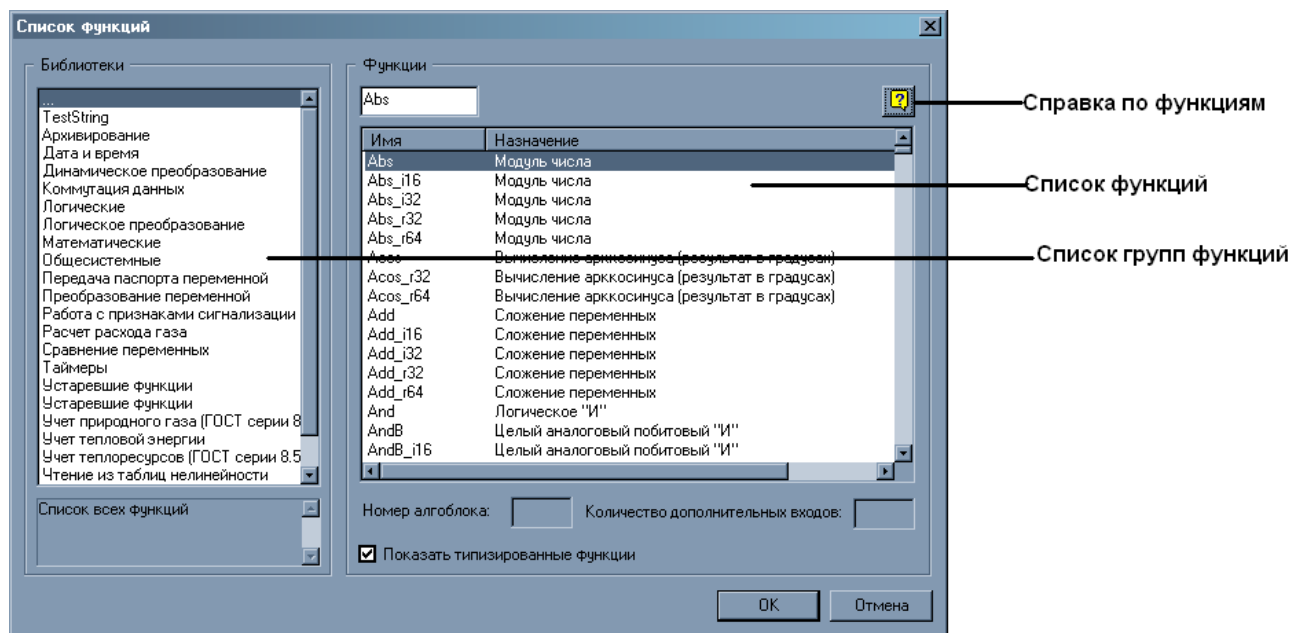


Рисунок 5.4.8 – Окно «Список функций»

В левой части диалога отображается список библиотек, доступных пользователю. При выборе той или иной библиотеки в правой части диалога будет отображаться список функций из

данной библиотеки. Для каждой выделенной функции можно посмотреть ее описание, нажав на кнопку с изображением знака вопроса в верхнем правом углу диалога.

При выборе некоторых функций могут активизироваться поля «**Номер алгоблока**» и «**Количество дополнительных входов**» (функция принимает неограниченное число входных параметров), значения которых необходимо задать Пользователю. Поле «**Количество дополнительных входов**» можно оставить пустым

Для платформ CPVK версии 8.0 и ИСР КРУГОЛ версии 2.2 можно задать режим отображения имен типизированных функций – включить переключатель «**Показать типизированные функции**» (информация о типизированных функциях приведена в разделе 3.7 «Перегрузка вызова функций» данного руководства).

Вставка переменной

Вставка переменной производится командой «**Список переменных...**». На экран выводится диалоговое окно «**Выберите переменные**» (рисунок 5.4.9), в котором следует выбрать переменную для вставки в схему.

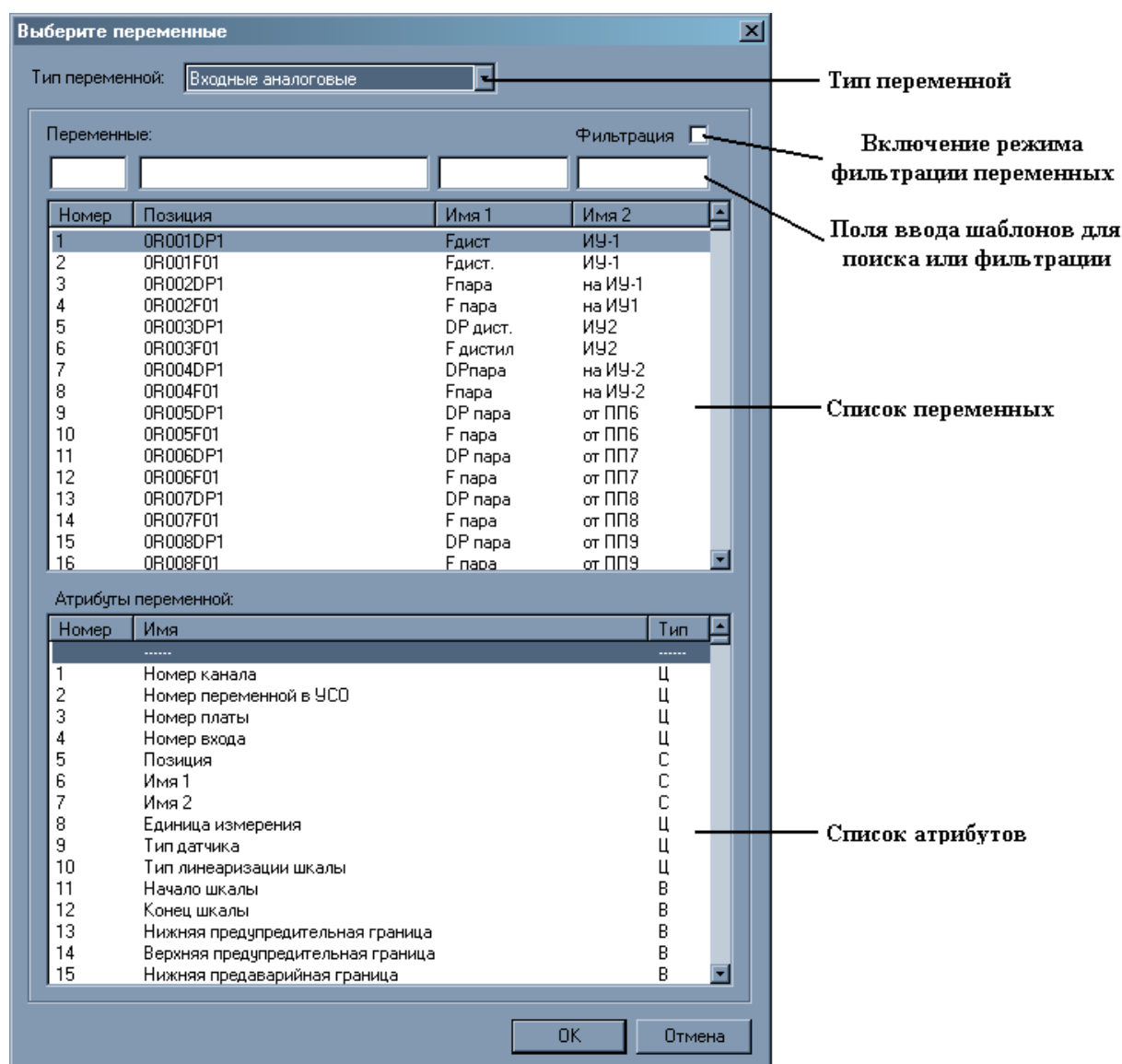


Рисунок 5.4.9 – Окно «Выберите переменные»

Раскрывающийся список «**Тип переменной**» позволяет выбрать тип переменной для вставки:

- **Системные переменные** – входные/выходные аналоговые, входные/выходные дискретные, переменные ручного ввода
- **Внутренние переменные** – промежуточные вещественные, целые, логические
- **Глобальные переменные** – пользовательские переменные, массивы пользовательских переменных
- **Локальные переменные** – пользовательские переменные, массивы пользовательских переменных
- **Параметры функции** – входные и выходные параметры пользовательской функции или алгоблока, могут быть вставлены только в схему ФБД для функции, определенного пользователем;

Пример окна «**Выберите переменные**» для переменных типа «**Входные аналоговые**» (в случае, когда ИСП установлена для работы со SCADA КРУГ-2000) приведен на рисунке 5.4.9.а.

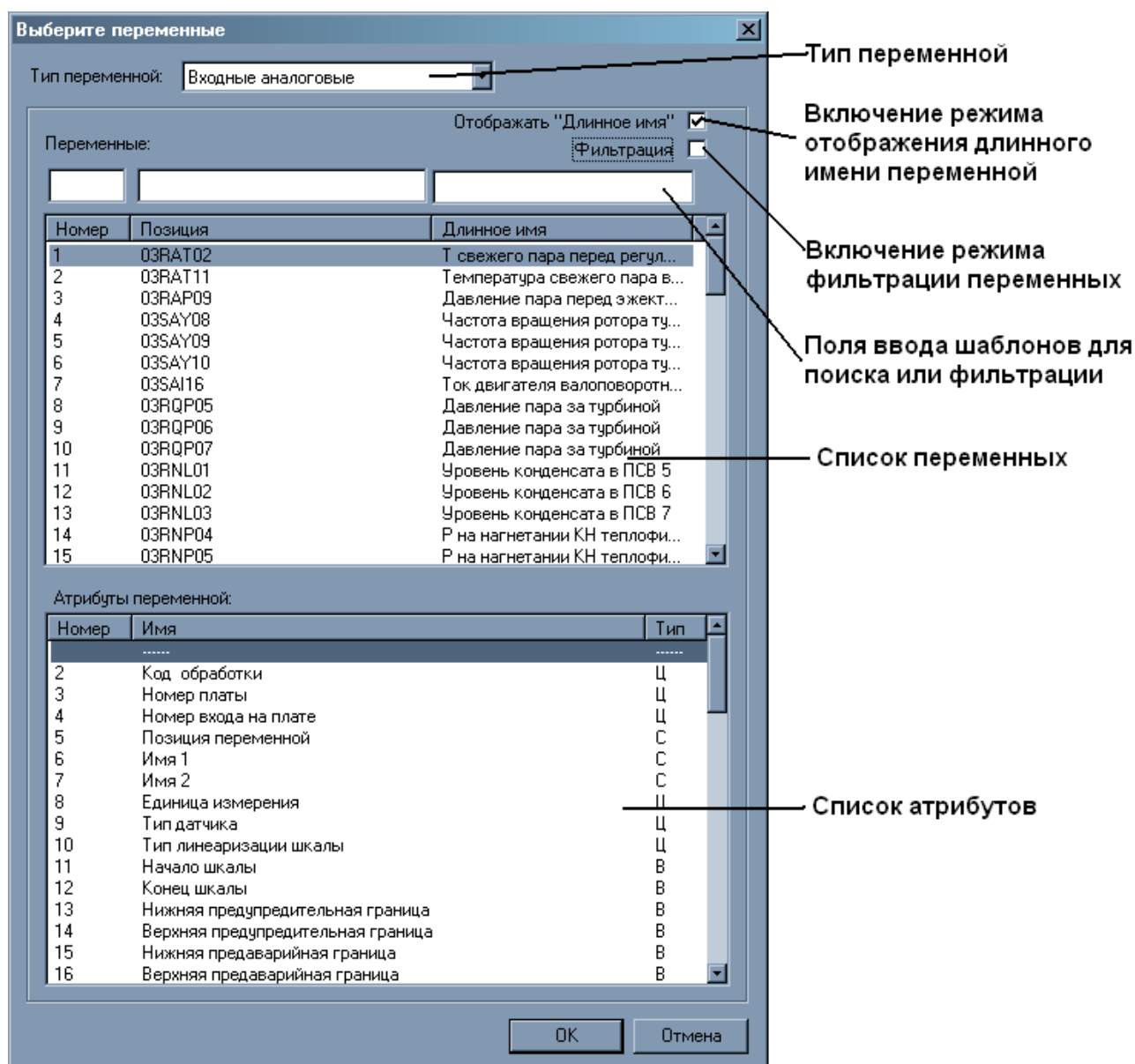


Рисунок 5.4.9 а – «Окно вставки переменной»

В верхней части диалога отображается список доступных переменных, в нижней – список атрибутов переменной (на рисунке 5.4.9 список для входной аналоговой переменной).

Для удобства поиска требуемой переменной над списком переменных расположены поля, в которых задается шаблон поиска. Если установить признак фильтрации, то вместо поиска будет осуществляться фильтрация переменных.

Поиск и фильтрация осуществляется по первым символам содержимого полей:

- «Номер», «Позиция», «Имя1» или «Имя2» («Длинное имя») – для системных переменных;
- «Номер» - для внутренних переменных;
- «Имя», «Тип», «Значение» - для глобальных и локальных переменных;
- «Имя», «Тип», «Размерность», «Индекс» - для массивов глобальных и локальных переменных;
- «Имя», «Вход/выход», «Тип», «Значение» - для параметров функций

Для системных переменных можно установить признак **«Отображать «Длинное имя»**. Если этот признак установлен, и данный проект создан для станции оператора или контроллера без поддержки 24-хсимвольной позиции, то в окне «Выберите переменные» список полей будет сформирован в составе: «Номер», «Позиция», «Имя 1», «Имя 2».

Если признак не установлен, и проект создан для контроллера с поддержкой 24-хсимвольной позиции, то список полей будет сформирован в составе: «Номер», «Позиция».

В случае, если признак «Отображать «Длинное имя»» установлен пользователем, то список полей будет сформирован в составе: Номер, Позиция, Длинное имя.

Информация о значениях атрибутов **«Длинное имя»** переменных определяется:

- для платформ проекта, созданного для контроллера, из файла **db_common.dat** в соответствии с номером канала связи
- для платформ проекта, созданного для станции оператора, данные читаются из атрибута **«Длинное имя»** соответствующего типа переменных.

Если SCADA не установлена, в диалоге выбора переменной отсутствует возможность установки режима отображения «Длинного имени».

Вставка константы

Вставка константы производится командой **«Константа»**. На экран выводится диалоговое окно **«Выберите константу»** (рисунок 5.4.10), в котором следует выбрать константу для вставки в схему.

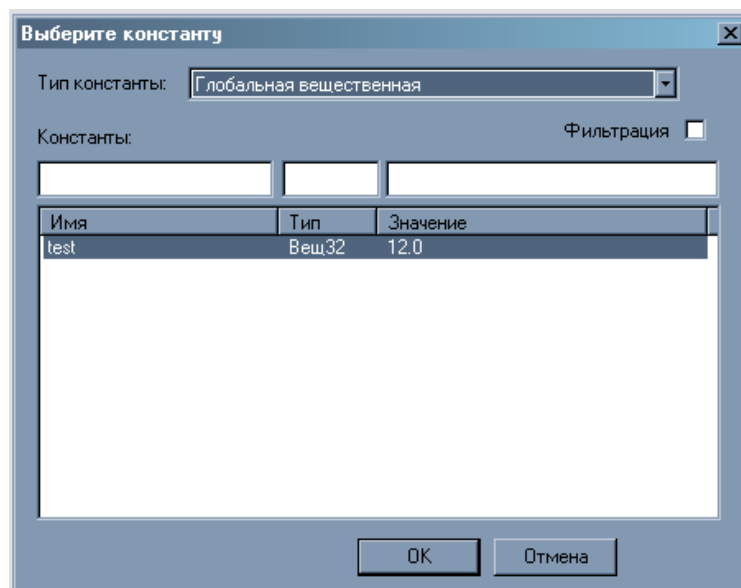


Рисунок 5.4.10 – Окно «Выберите константу»

Раскрывающийся список «**Тип константы**» позволяет выбрать тип константы для вставки:

- Глобальные – вещественная, целая, логическая, строковая (пример на рисунке 5.4.10)
- Локальные – вещественная, целая, логическая, строковая
- Массивы данных.

В общем случае поиск и фильтрация констант осуществляется по первым символам содержимого полей «Имя», «Тип» или «Значение».

Для удобства поиска и выборки констант можно задать шаблон в полях, которые расположены над списком констант.

Если установить признак фильтрации, то будет осуществляться выборка всех констант в соответствии с первыми символами полей «Имя», «Тип», «Значение» или с заданным пользователем шаблоном.

Вставка комментария

Вставка комментария производится командой «**Комментарий**». На экран выводится диалоговое окно «**Комментарий**» (рисунок 5.4.11), в котором следует ввести текст комментария.

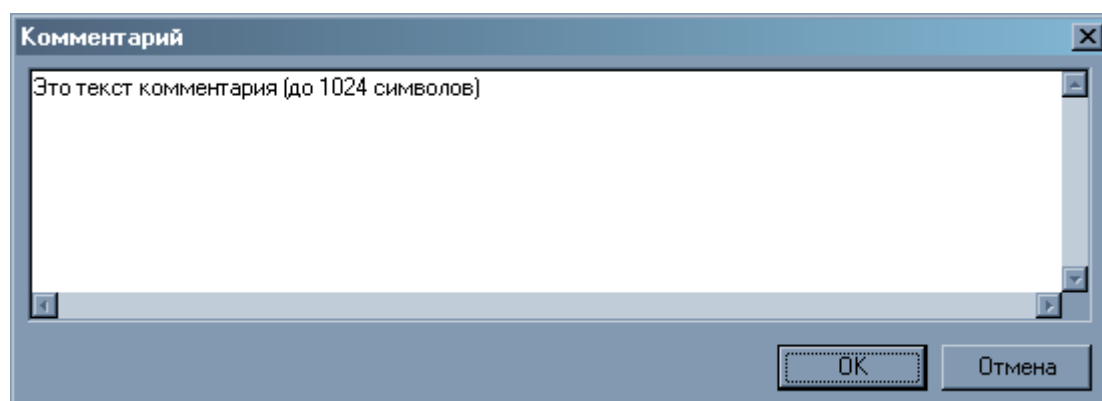


Рисунок 5.4.11 – Окно «Комментарий»

Вставка сообщения

Вставка сообщения производится командой «**Элемент сообщения**». На экран выводится диалоговое окно «**Свойства сообщения**» (рисунок 5.4.12).

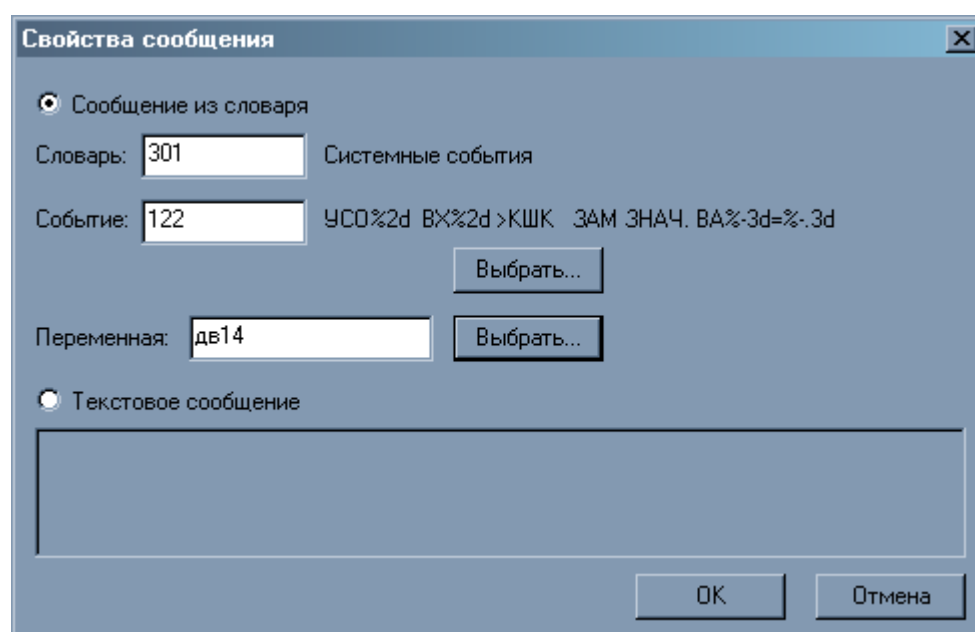


Рисунок 5.4.12 – Окно «Свойства сообщения»

В окне «Свойства сообщения» следует либо задать сообщение из словаря (установить признак **«Сообщение из словаря»**), либо ввести собственный текст сообщения .

В случае задания сообщения из словаря код словаря и код события можно вручную ввести в поля ввода **«Словарь»** и **«Событие»** соответственно или выбрать их в окне **«Выбор события»** (рисунок 5.4.13), которое открывается после нажатия на верхнюю из кнопок **«Выбрать»** в окне .«Свойства сообщения».

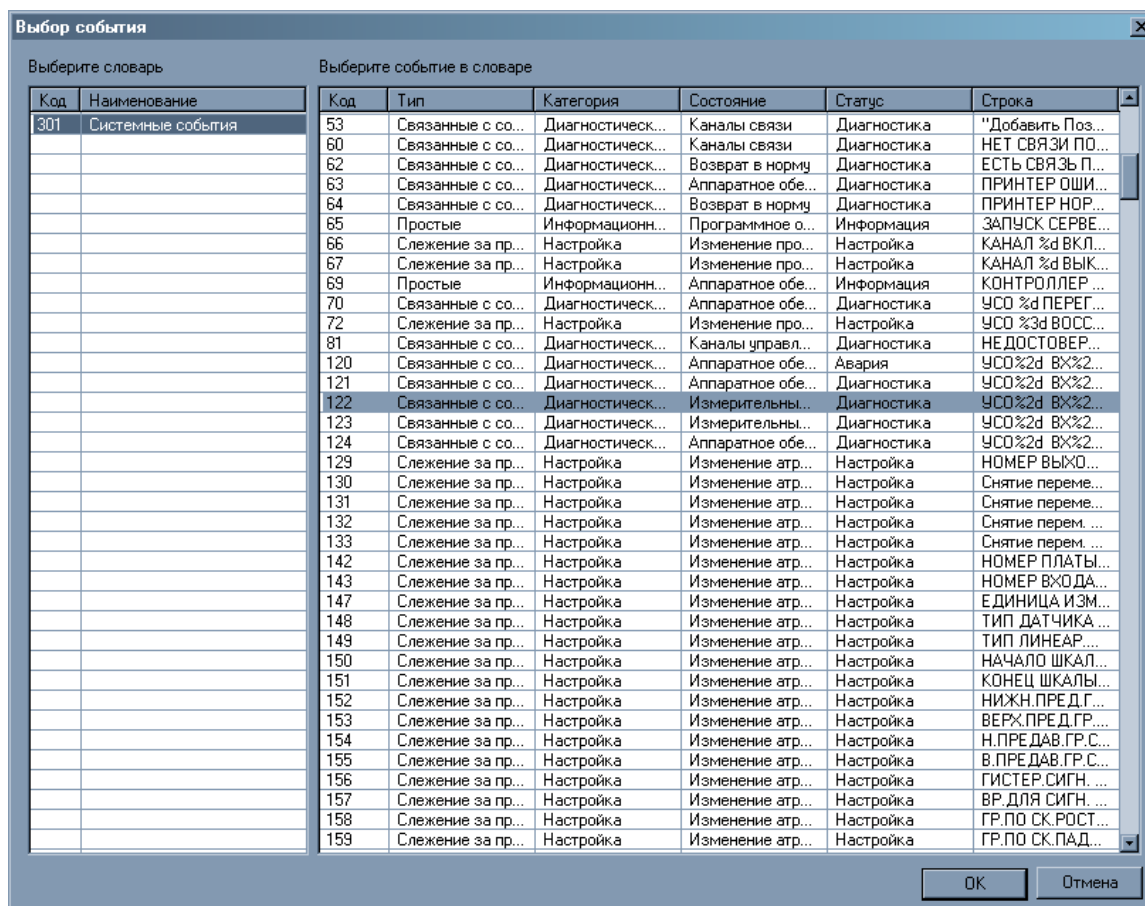


Рисунок 5.4.13 - Окно «Выбор события»

В случае создания собственного текста сообщения следует поставить переключатель в положение **«Текстовое сообщение»** и ввести требуемый текст сообщения в поле ввода.



ВНИМАНИЕ!!!

Текст сообщения должен удовлетворять следующим правилам:

- Сообщение может состоять из нескольких элементов, разделенных запятыми
- Элементами сообщения могут быть: текстовая строка, заключенная в кавычки; значения переменных и их атрибутов; числовые значения

Пример текста сообщения: "Это вывод значения переменной ПВ1 = ", пв1

В окне "Свойства сообщения" следует указать переменную, с которой будет связано сообщение. Переменную можно ввести вручную (в поле **«Переменная»**) или выбрать в окне **«Выбор переменной»**, которое открывается после нажатия на нижнюю из кнопок **«Выбрать»**.

Вставка блока «ЕСЛИ»

Вставка блока «ЕСЛИ» производится командой **«Блок «Если»**.

Вставка процедуры

Вставка процедуры производится командой **«Список процедур...»**. На экран выводится диалоговое окно **«Выберите процедуру»** (рисунок 5.4.14).

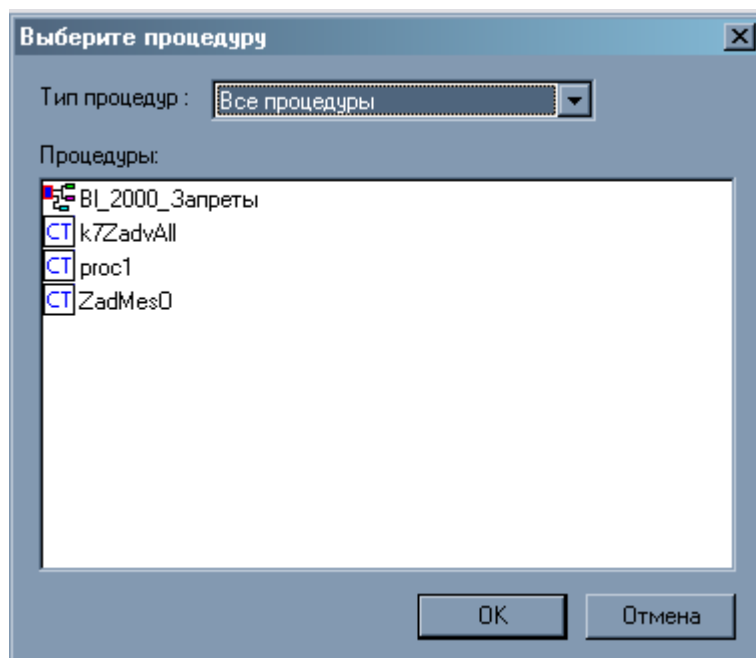


Рисунок 5.4.14 – Окно «Выберите процедуру»

В диалоговом окне необходимо выбрать вызываемую процедуру. С помощью раскрывающегося списка "Тип процедур" можно вывести для отображения только процедуры ФБД, только процедуры СТ или все процедуры.

Вставка элемента "Выход"

Вставка элемента «Выход» производится командой **«Выход»**.

Вставка блока ДЛЯ

Вставка блока «ДЛЯ» производится командой **«Блок «Для»**.

Вставка массива данных

Вставка массива данных производится командой **«Массив данных...»**. Команда доступна только при выделении какого-либо блока «Для» на схеме. На экран выводится диалоговое окно **«Создать массив данных»** (рисунок 5.4.15).

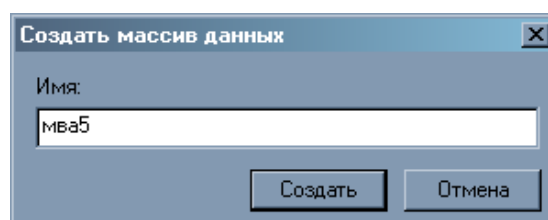


Рисунок 5.4.15 – Окно «Создать массив данных»

В окне следует задать имя нового массива данных.

Для формирования списка номеров массива данных необходимо дважды щёлкнуть левой кнопкой мыши по элементу и в появившемся окне (рисунок 5.4.16) задать необходимые номера.

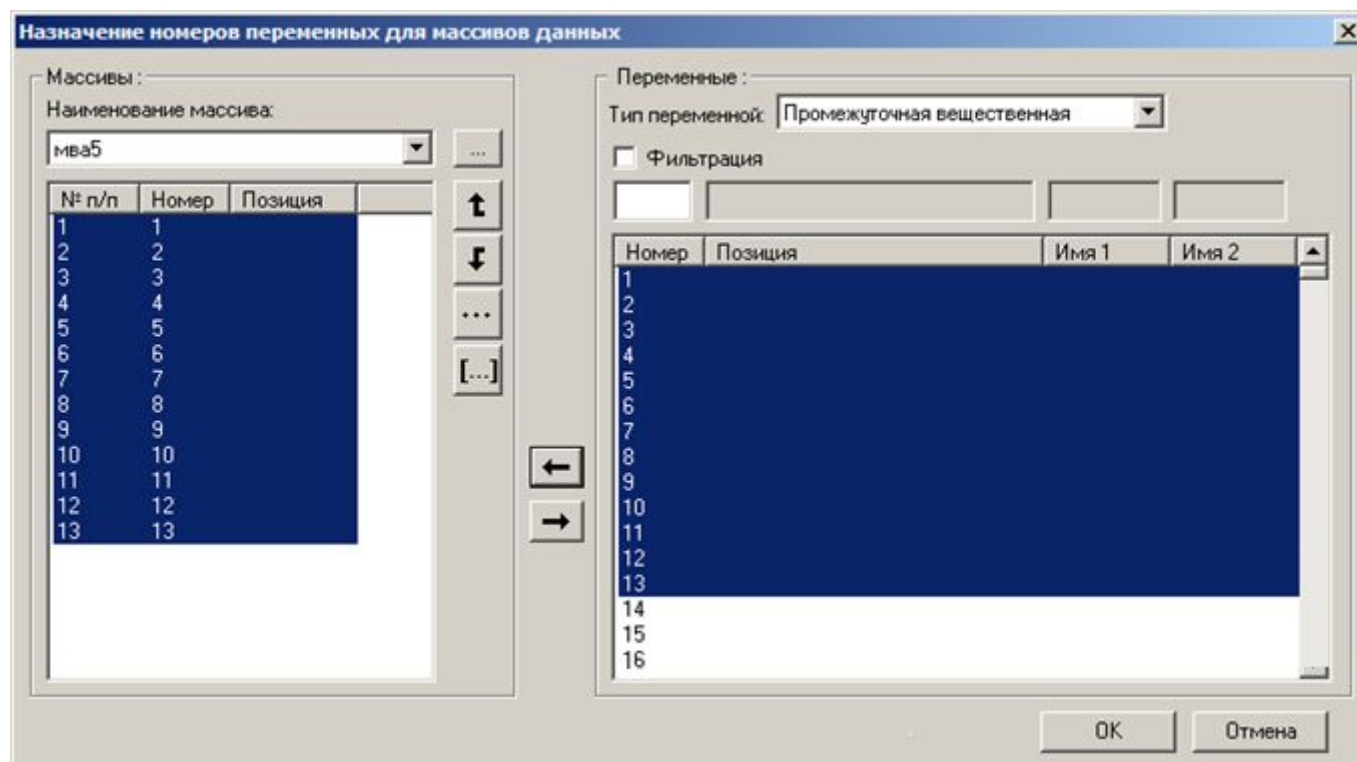


Рисунок 5.4.16 – Формирование списка номеров массива данных

Процедура добавления номеров массива данных происходит в несколько шагов:

- 1 Выбор типа переменной в поле «**Тип переменной**»
- 2 Выбор в панели «Переменные» номеров переменных, которые будут добавляться в массив, с помощью мыши и удерживаемых нажатыми кнопками **shift** или **ctrl**.
- 3 Перенос списка выбранных номеров переменных в список номеров переменных массива осуществляется путём нажатия кнопки с изображением стрелки соответствующего направления.

Вставка переменной массива

Вставка переменной массива производится командой «**Переменные МД...**». Команда вставляет на схему переменную, адресуемую через массив данных. Команда доступна только при выделении элемента «Для» на схеме. На экран выводится диалоговое окно «Переменная массива данных» (рисунок 5.4.17).

Раскрывающийся список «**Список массивов**» позволяет выбрать массив данных, который будет использоваться для адресации данной переменной.

Раскрывающийся список «**Тип переменной**» позволяет выбрать тип переменной, для которой будет отображаться список атрибутов в поле «**Атрибуты переменной**».

Доступны следующие типы переменных:

- **Системные** – входные/выходные аналоговые, Входные/выходные дискретные, переменные ручного ввода
- **Внутренние** – промежуточные вещественные, целые, логические
- **Массивы глобальных переменных**
- **Массивы локальных переменных.**

Если выбран любой из типов системной переменной, то в нижней части окна будет отображаться список атрибутов этого типа переменной, с помощью которого можно задать атрибут переменной массива данных.

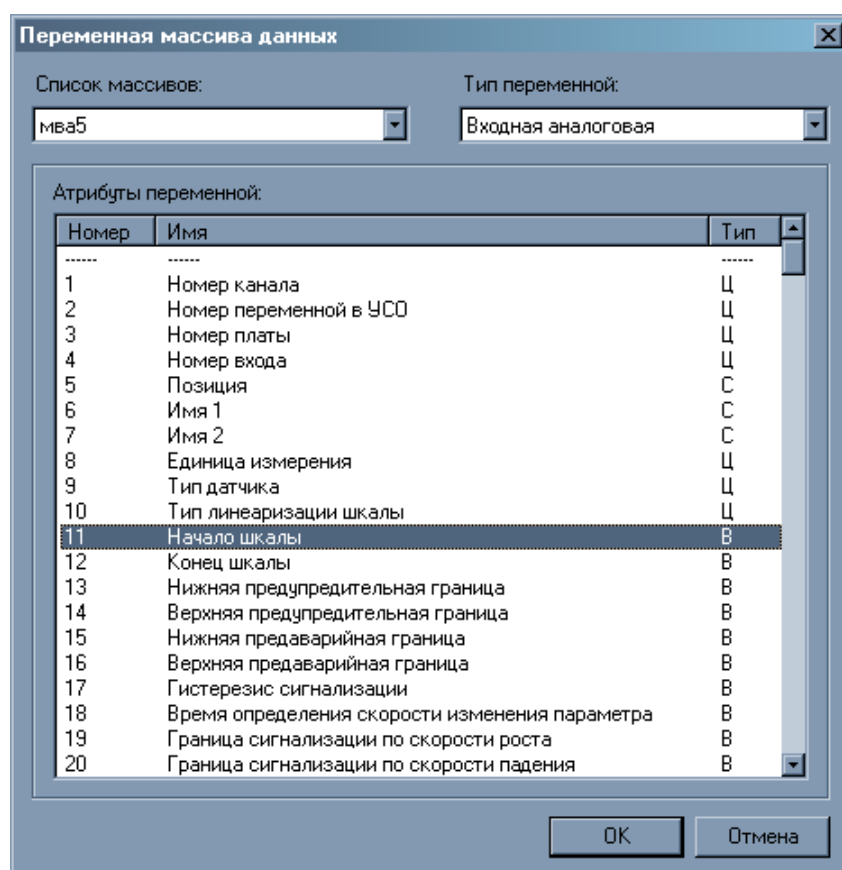


Рисунок 5.4.17 – Окно команды «Переменная массива данных»

Если выбраны массивы глобальных или локальных переменных, то в нижней части окна отобразится список с массивами глобальных или локальных переменных (рисунок 5.4.17а), в котором выбирается конкретный массив.

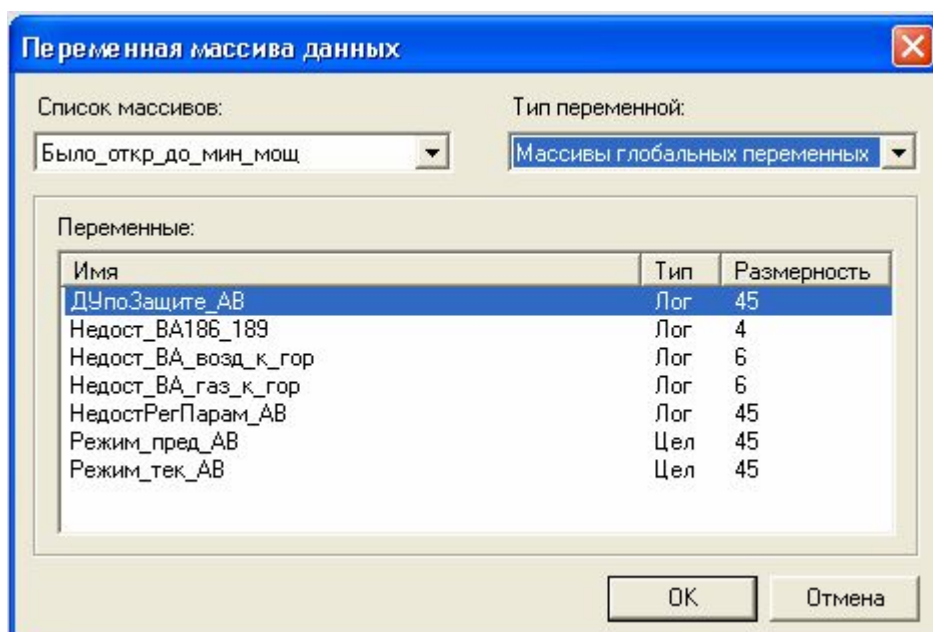


Рисунок 5.4.17а – Окно команды «Переменная массива данных»

Если выбран любой из внутренних типов, то в нижней части окна ничего не будет отображаться.

Вставка элемента «Прервать»

Вставка элемента «Прервать» производится командой «Прервать».

Вставка блока порядка выполнения

Вставка блока порядка выполнения производится командой «Блок порядка выполнения».

Вставка линии связи

Для соединения каких-либо двух элементов схемы ФБД линией связи следует нажать левую клавишу мыши в области выходного/входного контакта первого элемента, затем, не отпуская кнопки мыши, перетащить ее в область соответственно входного/выходного контакта второго элемента и отжать кнопку, после чего линия связи установится.

Для всех платформ СРВК ниже версии 8.0 и среды исполнения КРУГОЛ ниже версии 2.2 линия связи может соединять выходы и входы только одинакового типа.

Для платформ СРВК версии 8.0 и среды исполнения КРУГОЛ версии 2.2 можно соединять линией связи входы и выходы разных типов. Такое соединение возможно при выполнении следующих условий:

- Возможно преобразование данных из одного типа в другой без потери точности
- Тип выходного контакта явно задан.

В случае задания линии связи, при которой будет происходить потеря точности, пользователю выдаётся сообщение о необходимости использования функций преобразования данных.

При задании линий связи существует ряд ограничений:

- Задание линии связи с выходного контакта с неявным типом на входной контакт с неявным типом невозможно.
- Задать несколько линий связи с контакта с неявным типом данных возможно только в том случае, если тип входов явно задан и все входы имеют одинаковый тип. В противном случае можно задать только одну линию связи. При попытке задания большего количества линий связи будет выдано сообщение об ошибочных действиях, и линия связи не будет установлена

Таким же образом можно установить и линию связи с инверсией. Для этого достаточно перед вставкой линии связи выбрать элемент «Линия связи с инверсией» (команда меню «Правка\Вставка элементов\Линия связи с инверсией»), а затем с помощью мыши соединить требуемые контакты. Линией связи с инверсией можно соединять контакты только логического типа, иначе она будет автоматически преобразована в обычную линию связи.

Вставка элемента «Выражение СТ»

Вставка элемента «Выражение СТ» производится командой «Выражение СТ». На экран выводится диалоговое окно (рисунок 5.4.18), в котором необходимо ввести текст выражения на языке СТ.

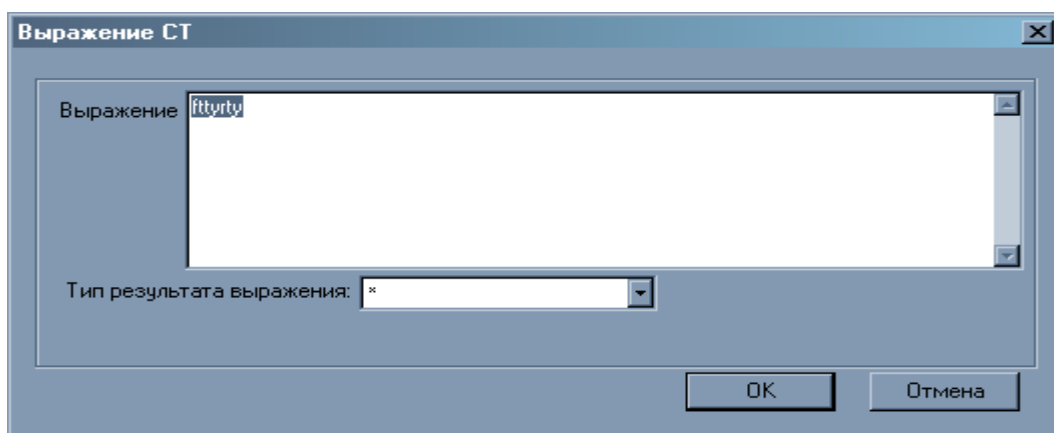


Рисунок 5.4.18 – Окно команды «Выражение СТ»

Для выражения, введённого пользователем, тип результата можно выбрать в списке «**Тип результата выражения**». По умолчанию тип выражения не определён – «*».

Контроль синтаксиса выражения производится на этапе трансляции программы.

5.4.2.2 Выделение элементов

Все действия редактирования производятся, как правило, над выделенными элементами. Выделить элемент/группу элементов можно следующими способами:

- **Выделение «щелчком» мыши.** Для выделения следует установить курсор мыши на требуемом элементе и нажать левую клавишу (рисунок 5.4.19).

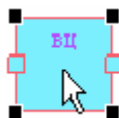


Рисунок 5.4.19 – Выделение элемента щелчком мыши

Если требуется выделить несколько элементов сразу (выделить группу элементов), следует нажать клавишу **Shift**, после чего, не отпуская ее, последовательно произвести «щелчки» мышью на каждом из требуемых элементов.

- **Выделение областью.** В этом случае следует нажать левую клавишу мыши в свободной области схемы, после чего, не отпуская ее, переместить ее в другую область схемы так, чтобы элементы, которые необходимо выделить полностью оказались внутри «прямоугольника выделения» и отжать клавишу мыши (рисунок 5.4.20).

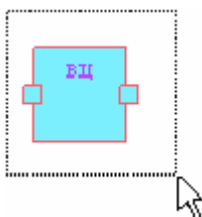


Рисунок 5.4.20 – Выделение элементов областью

При этом, если требуется выделить группу элементов, содержащуюся в каком-либо блоке (например, «ДЛЯ»), то операцию выделения областью следует производить при нажатой клавише **Shift**.

- **Выделение всех элементов.** Если требуется выделить сразу все элементы схемы, следует выбрать пункт меню «**Правка/Выделить все**».

Подобным образом осуществляется и выделение линий связи.

5.4.2.3 Перемещение элементов на схеме

Перемещение выделенного элемента/группы элементов с одной области схемы на другую осуществляется путем захвата их мышью и перетаскиванием в требуемую область с отпусканием мыши в точке назначения. При этом если перетаскиваемые элементы в точке назначения пересекутся с другими элементами схемы, то перемещение не осуществится, так как никакие два элемента схемы ФБД не могут иметь общих областей пересечения.

В то же время, возможно перемещение элемента из области непосредственно схемы в область какого-либо блока, например, "ДЛЯ" или "ЕСЛИ", а также из блока в блок. При этом, если перемещаемый в один блок элемент имеет связь с элементом, оставшимся в другом блоке, то данная связь разрывается (линия связи удаляется).

Также выделенные элементы можно перемещать при помощи клавиш управления курсором.

5.4.2.4 Изменение размеров элементов

Изменение размеров элементов применяется для придания схеме ФБД большей наглядности. Для этого, прежде всего, следует выделить элемент, размеры которого требуется изменить. Далее следует подвести курсор мыши к одному из 4-х маркеров выделения – при этом курсор примет форму в виде двунаправленной стрелки. Далее следует нажать левую кнопку мыши, и, не отпуская кнопки, переместить курсор на другое место (тем самым, задавая новые размеры элемента) в свободной области схемы, после чего отжать кнопку – размеры элемента изменятся (рисунок 5.4.21).

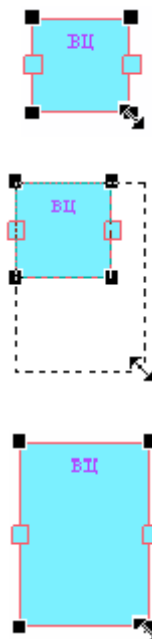


Рисунок 5.4.21 – Изменение размера элемента

5.4.2.5 Изменение линий связи

Изменение формы линий связи используется для составления более наглядных схем ФБД. Для этого следует «щелкнуть» мышью по требуемому сегменту линии связи, и, не отпуская кнопку, перетащить на другое место, после чего отжать кнопку мыши. Выбранный сегмент переместится на новое место, а соседние сегменты соответственно станут длиннее или короче (рисунок 5.4.22).

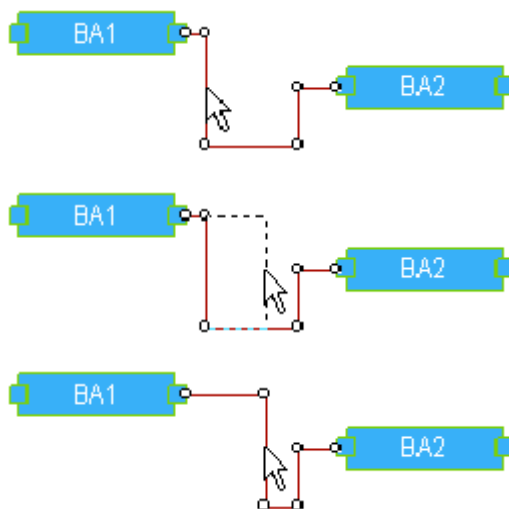


Рисунок 5.4.22 – Изменение линий связи

При этом, если перемещаемый сегмент является первым/последним для данной линии связи, то перед ним/после него будет создан новый сегмент, что позволяет добавлять новые сегменты к линии связи (рисунок 5.4.23).

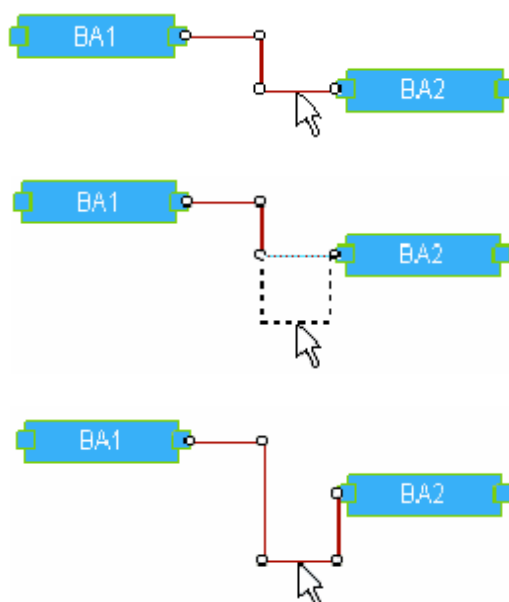


Рисунок 5.4.23 – Добавление новых сегментов линии связи

Если же, наоборот, требуется уменьшить число сегментов линии связи, то для этого достаточно соединить (сблизить) любые два соседних сегмента (горизонтальных или вертикальных) (рисунок 5.4.24).

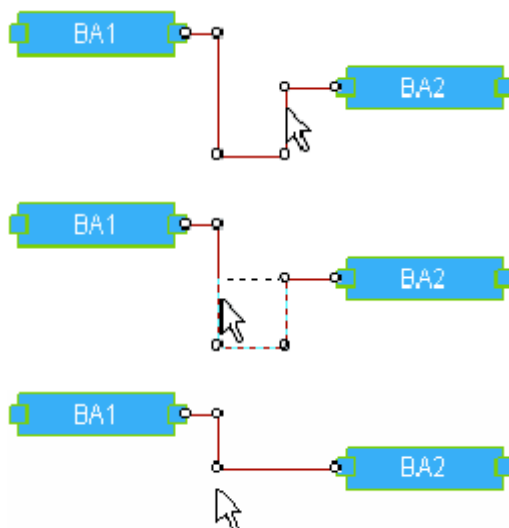


Рисунок 5.4.24 – Удаление сегментов линии связи

5.4.2.6 Удаление элементов из схемы

Для удаления требуемых элементов следует их выделить и выбрать пункт меню «Правка/Удалить» или нажать клавишу **Delete**. При этом будут удалены все линии связи, соединенные с контактами данных элементов.

Аналогичным образом осуществляется удаление линий связи.

5.4.2.7 Копирование и вставка элементов в/из буфера обмена

Редактор ФБД реализует функции копирования и вставки элементов, для чего поддерживает собственный буфер обмена. Для копирования требуемых элементов в буфер обмена следует выделить их, а затем выбрать пункт меню **«Правка/Копировать»**. Для вставки сохраненных элементов из буфера обмена сначала выбирается пункт меню **«Правка/Вставить»**, после чего форма курсора поменяется на соответствующий данному типу операции. Затем следует щелкнуть мышью в требуемом месте в свободной области схемы. Элементы из буфера вставятся в схему. Кроме того, осуществить операцию копирования/вставки можно путем перемещения выделенной группы элементов с одной области схемы на другую (свободную от других элементов) при нажатой клавише **Ctrl**.

Команда меню **«Правка\Вырезать»** копирует выделенные элементы в буфер, а затем удаляет их из схемы.

Командой меню **«Правка\Копировать как метафайл»** копирует схему ФБД в виде рисунка в буфер обмена Windows. В дальнейшем этот рисунок можно вставить в другое приложение (например, документ MS Word). Если же нужно скопировать только часть схемы, достаточно выделить требуемые элементы и выбрать в меню данную команду.

5.4.2.8 Работа с точками останова

Редактор ФБД поддерживает возможность работы с точками останова.

Точка останова – это «метка» на элементе схемы, которая означает, что выполнение программы в режиме отладки будет приостановлено на этом элементе и продолжено по команды Пользователя.

Работа с точками останова осуществляется посредством команд меню, расположенных в меню **«Отладка\Точки останова\...»** (рисунок 5.4.25).

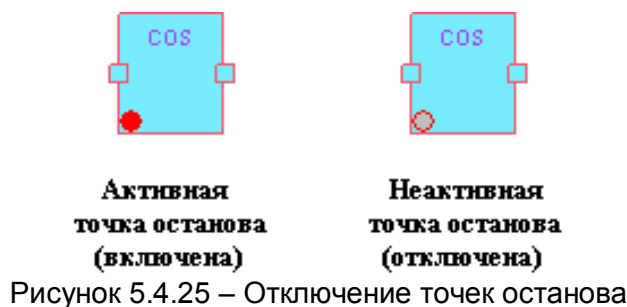


Рисунок 5.4.25 – Отключение точек останова

Если требуется установить точку останова на один или более элемент схемы, следует, предварительно выделив эти элементы, выбрать команду меню **«Установить точку останова»**. Если же на каком-либо из выделенных элементов уже установлена точка останова, то она, наоборот, удалится. То есть одна и та же команда позволяет устанавливать и снимать точки останова.

Команда **«Снять все точки останова»** приводит к удалению точек останова со всех элементов во всех схемах проекта.

Команда **«Отключить точки останова»** (соответствующие элементы должны быть выделены) временно отключает точку останова, и выполнение программы на этом элементе приостанавливаться не будет, хотя сама точка останова будет видна (ее цвет поменяется на другой, рисунок 5.4.25).

Отключение производится

Команда **«Отключить все точки останова»** приводит к отключению всех точек останова со всех элементов во всех схемах проекта.

В любой момент отключенные точки останова можно активизировать, выбрав команду **«Установить точку останова»** (соответствующие элементы должны быть выделены).

Работа с точками останова возможна как в режиме редактирования, так и в режиме отладки.

5.4.2.9 Временное исключение элементов – комментирование

Иногда требуется временно исключить некоторые части схемы из процесса трансляции и отладки. Эта возможность реализуется при помощи выделения нужной части схемы и выбора команды **«Правка\Вставка элементов\Комментировать»**. При этом цвет выделенных элементов поменяется на серый, что будет означать, что они закомментированы (рисунок 5.4.26).

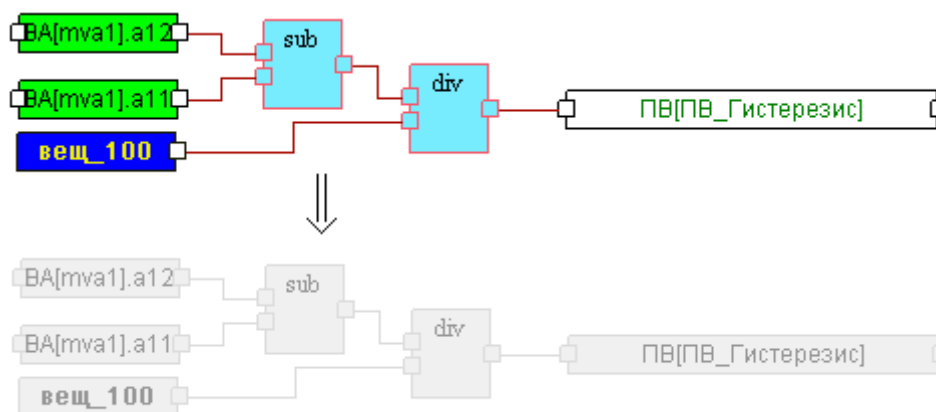


Рисунок 5.4.26 – Комментирование элементов схемы

Включение закомментированных элементов в схему – «раскомментирование» выделенных элементов – выполняется командой **«Правка\Вставка элементов\Снять комментарий»**.

5.4.2.10 Порядок выполнения элементов

Порядок выполнения элементов – это очередность, в которой они будут транслироваться в код и выполняться при выполнении всей схемы.

Порядок выполнения рассчитывается автоматически. Для отображения порядка выполнения элементов следует выбрать команду меню **«Вид\Порядок выполнения»**. В этом случае над каждым элементом отобразится номер порядка его выполнения (рисунок 5.4.27):

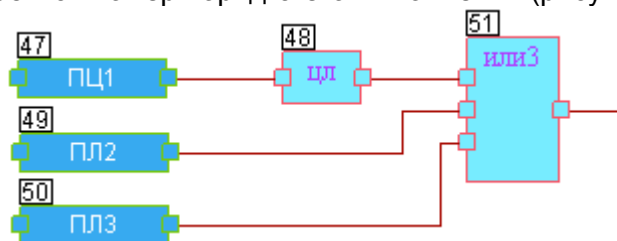


Рисунок 5.4.27 – Порядок выполнения элементов

5.4.2.11 Изменение свойств элементов

Редактор ФБД поддерживает возможность изменять графические свойства элементов схемы (стиль схемы) – цвета элементов, их размеры, шрифт и другие. Для того чтобы задать собственный стиль схемы, следует выбрать команду меню **«Вид\Стиль схемы...»**. При этом на экран будет выведено окно **«Стиль схемы»** с двумя вкладками: «Цвета и линии» и «Шрифт».

Вкладка «Цвета и линии»

Вкладка «Цвета и линии» (рисунок 5.4.28) предназначена для изменения цвета и толщины линий, ограничивающей прямоугольную область элемента, и линий связи, а также для изменения цвета заливки области элемента и размеров элемента. Здесь же можно изменить свойства отображения контактов элементов. Сделанные изменения будут применяться ко всем существующим элементам схемы, а также элементам, вставляемым в схему.

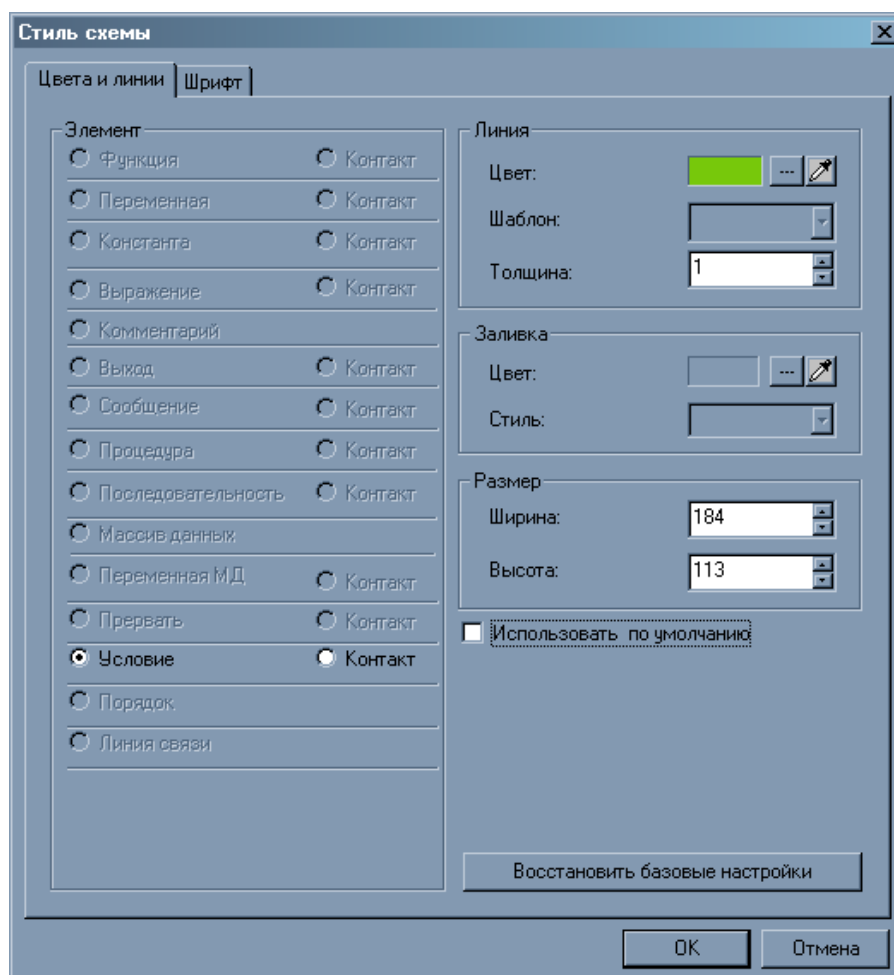


Рисунок 5.4.28 – Окно «Стиль схемы». Закладка «Цвета и линии»

Вкладка «Шрифт»

Вкладка «Шрифт» (рисунок 5.4.29) предназначена для изменения параметра шрифта названия элемента, а также шрифта текста, который может отображать элемент (например, элемент «Условие»).

В случае, когда на схеме выделены один или несколько элементов, можно изменить стили соответствующих выделенных элементов или выбрать стили по умолчанию. В случае, когда на схеме нет выделенных элементов, производится редактирование параметров стилей по умолчанию.

Установка "галочки" «**По умолчанию**» означает, что все заданные в данном диалоге свойства будут применяться ко всем вновь создаваемым элементам. Иначе новые свойства будут применены только к текущим выделенным элементам.

Кнопка «**Восстановить базовые настройки**» позволяет вернуть predetermined settings of highlighted elements.

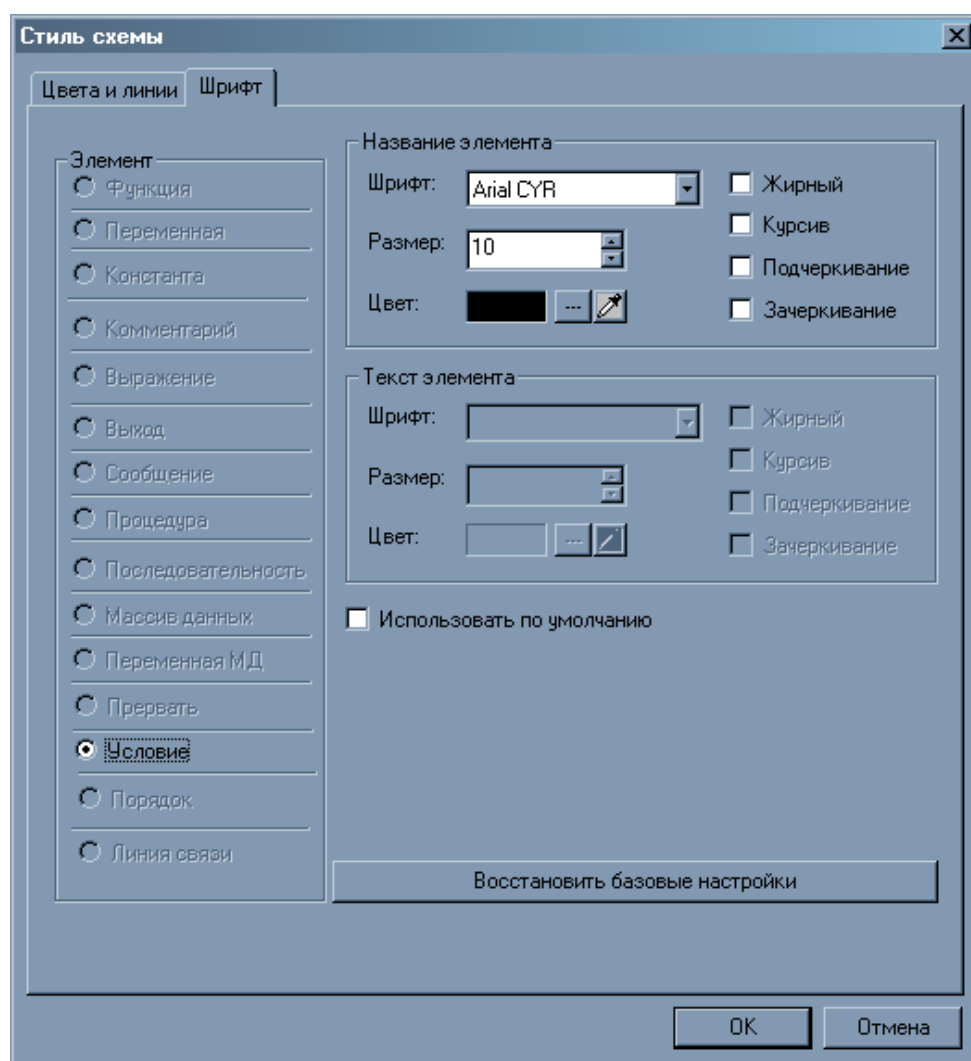


Рисунок 5.4.29 – Окно «Стиль схемы». Закладка «Шрифт»

5.4.2.12 Свойства отображения

В процессе редактирования схемы можно включать отображение различной дополнительной информации об элементах. Для этого предназначены несколько команд меню «Вид»:

- **Типы контактов** – команда предназначена для отображения или скрытия на схеме ФБД информации о типах контактов элементов (рисунок 5.4.30).

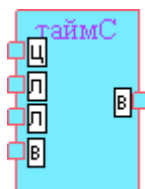


Рисунок 5.4.30 – Типы контактов

Возможны следующие типы контактов:

- **B32** – контакт 32-х битового вещественного типа
- **B64** – контакт 64-х битового вещественного типа
- **Ц16** – контакт 16-ти битового целого типа
- **Ц32** – контакт 32-х битового целого типа
- **Ц8** – контакт 8-ми битового целого типа

- **Л** – контакт логического типа
- **С** – контакт строкового типа
- ***** – контакт неопределенного типа, может принимать значения любого типа.
- **Названия контактов функций** – команда предназначена для отображения названий контактов функций. В режиме отображения около каждого контакта функции выводится соответствующее название (рисунок 5.4.31).

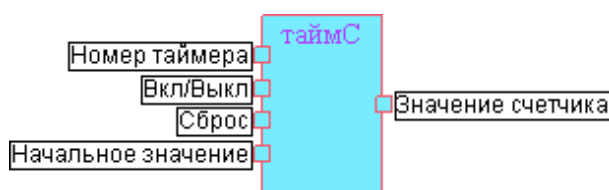


Рисунок 5.4.31 – Названия контактов

- **Позиция** – команда управляет отображением названий переменных. Если команда активна, то в названии переменной отображается позиция переменной. Если команда не активна, то в названии переменной отображается тип переменной, номер переменной и номер атрибута (рисунок 5.4.32).

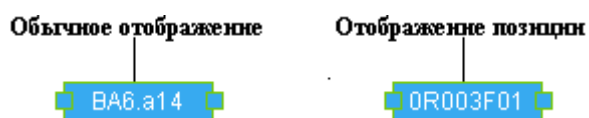


Рисунок 5.4.32 – Название переменных

5.4.2.13 Масштабирование отображения схемы

Пользователь может изменить масштаб отображения схемы ФБД. Это можно сделать при помощи команд меню «Вид\Увеличить» и «Вид\Уменьшить». Уменьшить масштаб отображения можно до 25% от исходного, а увеличить – до 200% от исходного.

На главной панели инструментов присутствует список выбора «Масштаб», позволяющий выбрать любой из предусмотренных масштабов отображения, а также кнопка «Масштаб 100%», предназначенная для быстрого сброса ранее выбранного масштаба изображения в исходные 100%.

Масштаб отображения свой для каждой схемы ФБД.

5.4.2.14 Отмена сделанных изменений

Для удобства редактирования схем Редактор ФБД поддерживает функцию отката сделанных изменений. Любые действия по редактированию схемы ФБД – вставка, удаление, перемещение элементов, изменение размеров и другие изменения – могут быть отменены. Поддерживается только отмена назад, то есть возврат отмененных действий невозможен.

При этом "глубина" (т.е. количество действий, которые могут быть отменены) зависит от объема используемой оперативной памяти.

Отмена сделанных изменений осуществляется по команде меню «Правка\Отменить». За один раз происходит отмена только одного действия по редактированию схемы, которое может быть комплексным – например, удаление группы выделенных элементов.

5.4.2.15 Разметка страницы

Для удобства организации предпечатной подготовки в редакторе имеется режим разметки страницы. При включении этого режима на схеме ФБД появляются горизонтальные и вертикальные пунктирные линии чёрного цвета, которые образуют сетку. Ячейка этой сетки представляет собой видимую область печати листа с учётом всех отступов, заданных в диалоге настройки параметров страницы.

Вид разметки страницы приведён на рисунке 5.4.33

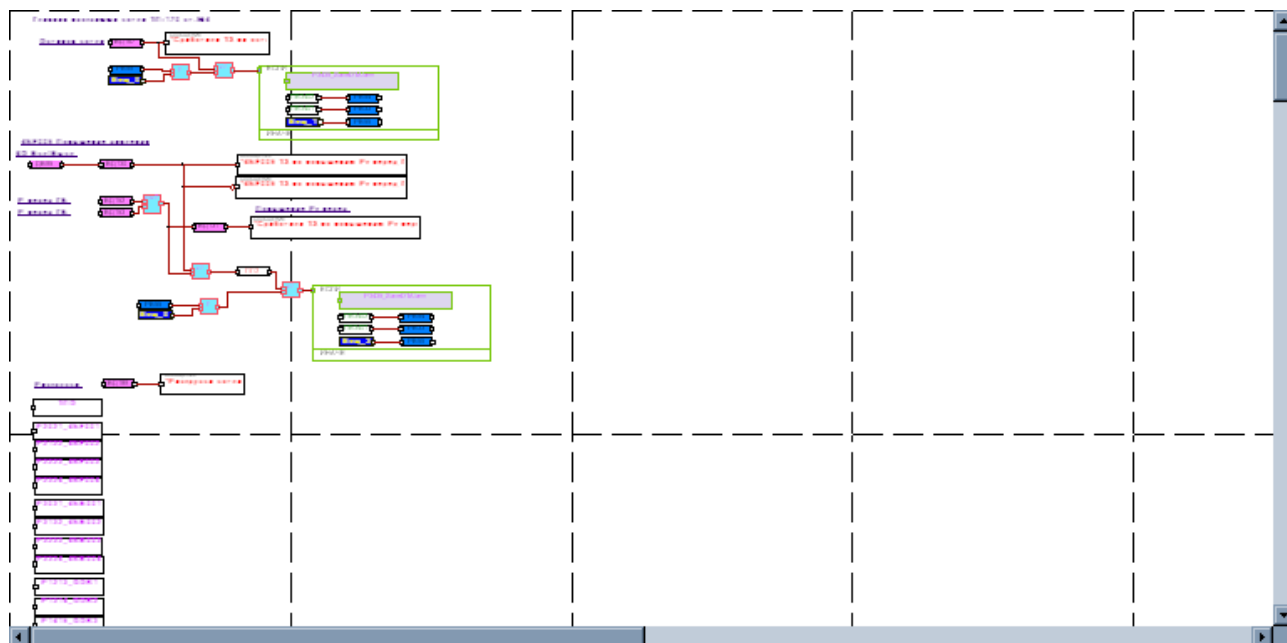


Рисунок 5.4.33 – «Разметка страницы»

Включение/выключение режима разметки страницы осуществляется из меню «Вид» командой «**Разметка страницы**».

ВНИМАНИЕ!

Для корректной работы режима разметки страницы необходимо, чтобы в системе был установлен хотя бы один драйвер принтера. Наличие самого принтера не обязательно.

5.4.2.16 Отображение сетки

Режим отображения сетки используется для выравнивания элементов на схеме ФБД.

Вид схемы ФБД с включенным режимом отображения сетки представлен на рисунке 5.4.34

Режим отображения можно включить или выключить с помощью команды «**Отображение сетки**» меню «Вид».

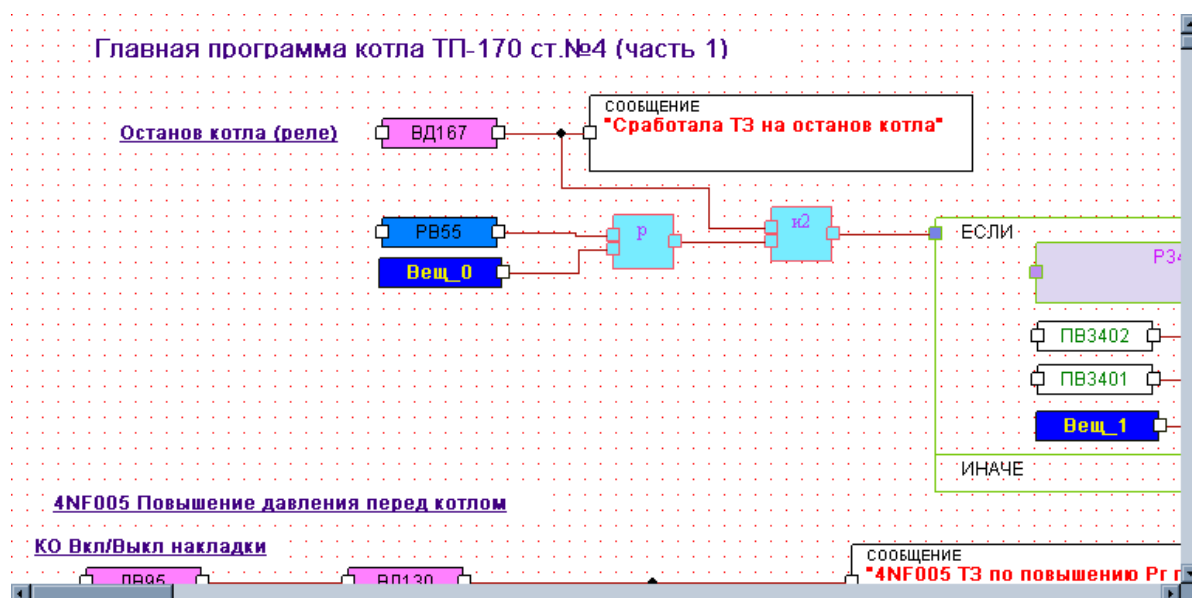


Рисунок 5.4.34 – «Схема ФБД при включенном режиме отображения сетки»

5.4.2.17 Настройка параметров отображения сетки

В редакторе предусмотрена возможность настройки параметров сетки.

Настройка параметров отображения сетки осуществляется в окне «Сетка» (рисунок 5.4.35).

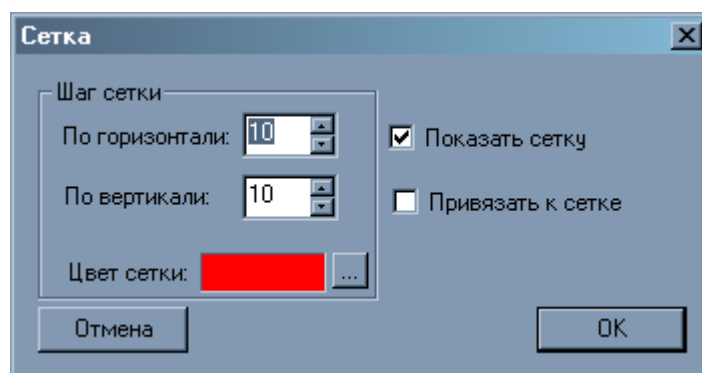


Рисунок 5.4.35 – «Настройка параметров сетки»

Настроить можно следующие параметры:

- **Вертикальный шаг сетки.** Значение шага может принимать значения от 4 до 40 пикселей
- **Горизонтальный шаг сетки.** Значение шага может принимать значения от 4 до 40 пикселей
- **Цвет сетки.** Может быть любым из доступных цветов в операционной системе
- **Режим отображения сетки**
- **Режим привязки к сетке** При включенном режиме привязки к сетке перемещение элементов схемы ФБД осуществляется по узлам этой сетки. Для точного перемещения элементов (с шагом в 1 пиксель) при включенном режиме привязки необходимо использовать клавишу Alt+клавиши управления курсором.

Все настройки запоминаются на уровне проекта.

5.5 Трансляция проекта

5.5.1 Транслятор языка КРУГОЛ

После завершения разработки текста программ на языках СТ и ФБД Пользователь может осуществить их трансляцию.

Результатом трансляции являются исполняемые модули в виде файлов с расширением .out, предназначенные для запуска в контроллере или станции оператора. Каждой программе проекта соответствует отдельный out-файл.

Исполняемые файлы, получаемые в результате трансляции, сохраняются в текущем каталоге проекта. Имя исполняемого файла состоит из имени программы, для которой генерируется файл, и расширения .out.

Например, для программы:

```
Программа Test
Начало
<список операторов>
Конец
```

будет создан файл **Test.out** независимо от названия файла, в котором эта программа определена.

Транслятором производится анализ зависимостей (вызовов процедур, функций и алгоблоков) транслируемой программы. В результате анализа в out-файл попадают только те процедуры, функции и алгоблоки, которые реально используются программой, хотя проект может содержать произвольное количество процедур, функций или алгоблоков, совместно используемых разными программами.

В процессе трансляции проверяется правильность составления программ. В случае обнаружения ошибок в окно вывода интегрированной среды разработки выдаются соответствующие сообщения об ошибках и исполняемые модули не создаются. После исправления ошибок необходимо повторить трансляцию.

При отсутствии ошибок транслятор формирует сообщение об успешной генерации out-файла.

Транслировать можно как весь проект в целом, так и отдельную программу СТ или ФБД:

Трансляция проекта

Трансляция проекта осуществляется по команде меню «**Трансляция\Транслировать проект**»

Трансляции отдельной программы

Трансляция отдельной программы осуществляется выбором пункта «**Транслировать**» в контекстном меню, которое появляется при нажатии правой кнопкой мыши на имени программы во вкладке «**Проект**» окна проекта или выбором пункта меню «**Трансляция\Транслировать программу**».

Проверка пользовательских словарей

Проверка пользовательских словарей для выявления некорректных данных осуществляется при помощи команды меню «**Трансляция\Проверить словари**». В случае наличия таковых в окно вывода выдается соответствующее сообщение с указанием словаря и строки таблицы, содержащей ошибку. Данная функция также автоматически вызывается средой разработки перед каждой трансляцией проекта или программы.

Генерацию исполняемого кода

Транслятор обеспечивает генерацию исполняемого кода для следующих платформ:

Система реального времени (СРВ) контроллера:

- СРВ версии 8.0 для контроллеров TREI-5B-02, МФК-3000, P06, операционная система Linux.
- СРВ версии 7.11 для контроллеров P06, операционная система Linux
- СРВ версии 7.1x для контроллеров TREI-5B-02, МФК-3000, операционная система Linux
- СРВ версии 7.0 для контроллеров TREI-5B-02, МФК, МФК-3000, операционная система Linux
- СРВ версии 6.5 для контроллеров TREI-5B-02, МФК, операционная система Linux. При этом кроме файла с расширением .out дополнительно генерируется файл с расширением .lab.
- СРВ версии 6.5 для контроллеров TREI-5B-02, операционная система QNX,. При этом кроме файла с расширением .out дополнительно генерируется файл с расширением .lab
- СРВ версии 7.1x для контроллеров TREI-5B-02, МФК-3000, операционная система Linux
- ПО контроллеров DevLink-D500, DevLink-C1000
- СРВК версии версии 7.1x, операционная система Linux, для контроллеров MDS100CPU.

СРВ станции оператора:

- Среда исполнения КРУГОЛ версии 2.2, операционная система Windows
- Среда исполнения КРУГОЛ версии 2.1, операционная система Windows
- Среда исполнения КРУГОЛ версии 2.0, операционная система Windows
- Среда исполнения КРУГОЛ версии 1.0, операционная система Windows (со SCADA КРУГ-2000 версий 2.3 – 2.5). При этом кроме файла с расширением .out дополнительно генерируется файл с расширением .lab.



ВНИМАНИЕ!!!

В версиях СРВК ниже 7.0 или ИСР КРУГОЛ ниже 2.0 не поддерживаются следующие возможности технологического языка КРУГОЛ:

- Глобальные массивы
- Глобальные переменные
- Использование переменной оператора последовательности для индексации локальных массивов
- Использование переменной оператора последовательности в качестве операнда.

Алгоблоки пользователя и автоматическая нумерация алгоблоков доступны, начиная с ИСР КРУГОЛ версии 2.1 и СРВК версии 7.1.

Расширенные типы данных доступны начиная с ИСР КРУГОЛ версии 2.2 и СРВК версии 8.0.

5.5.2 Сообщения транслятора об ошибках

Если в процессе трансляции обнаруживаются несоответствия правилам технологического языка (СТ или ФБД), тогда транслятор выдает сообщения об ошибках в «**Окно вывода**» ИСР КРУГОЛ.

Для того чтобы перейти к элементу программы, вызвавшему сообщение об ошибке, достаточно поставить курсор мыши на текст сообщения в окне вывода и дважды щелкнуть левой клавишей мыши.

Сводная таблица сообщений транслятора об обнаруженных в программе ошибках приведена в **Приложение D** данного документа.

5.6 Отладка проекта

5.6.1 Общие сведения

Отладчик ИСР КРУГОЛ предоставляет Пользователю следующие возможности:

- Отладку как для платформы станции оператора, так и для платформы контроллера
- Для платформы контроллера как локальная, так и удаленная отладка
- Отладка проекта или отдельной программы
- Отладка в ручном или автоматическом режиме
- Пошаговая или циклическая отладка
- Приостановить/продолжить отладку
- Запись значений переменных в БД.



ВНИМАНИЕ!

Перед началом процесса отладки НЕОБХОДИМО:

- Для платформы Станции оператора – запустить Сервер базы данных совместно с той базой данных, к которой привязаны переменные проекта
- Для платформы контроллера – убедиться в правильности настроек связи проекта с БД контроллера (связь с БД задается в свойствах проекта).

Для работы в режиме отладки предназначены меню «**Отладка**», и соответствующая панель инструментов, а также окно переменных и автоматическое окно переменных.

Запуск проекта на выполнение в режиме отладки

Запуск проекта на выполнение в режиме отладки осуществляется командой «**Выполнить проект**» меню «**Отладка**». В этом случае программы будут выполняться последовательно в том порядке, в котором они расположены в секции «**Программы**» проекта (смотрите описание вкладки в разделе 5.2 «Окно проекта»).

Запуск отдельной программы в режиме отладки

Запуск отдельной программы в режиме отладки производится путем нажатия правой кнопки мыши на ее имени и выбора в контекстном меню команды «**Отладка**».

Временная приостановка процесса отладки

В **автоматическом режиме** процесс отладки можно временно приостановить по команде «**Отладка\Пауза**». Возобновление отладки осуществляется по команде Пользователя.

Выход из режима отладки

Выход из режима отладки производится по команде «**Выход из отладки**» меню «**Отладка**».

5.6.2 Окно проекта в режиме отладки

Общий вид интегрированной среды разработки КРУГОЛ в режиме отладки показан на рисунке 5.6.1:

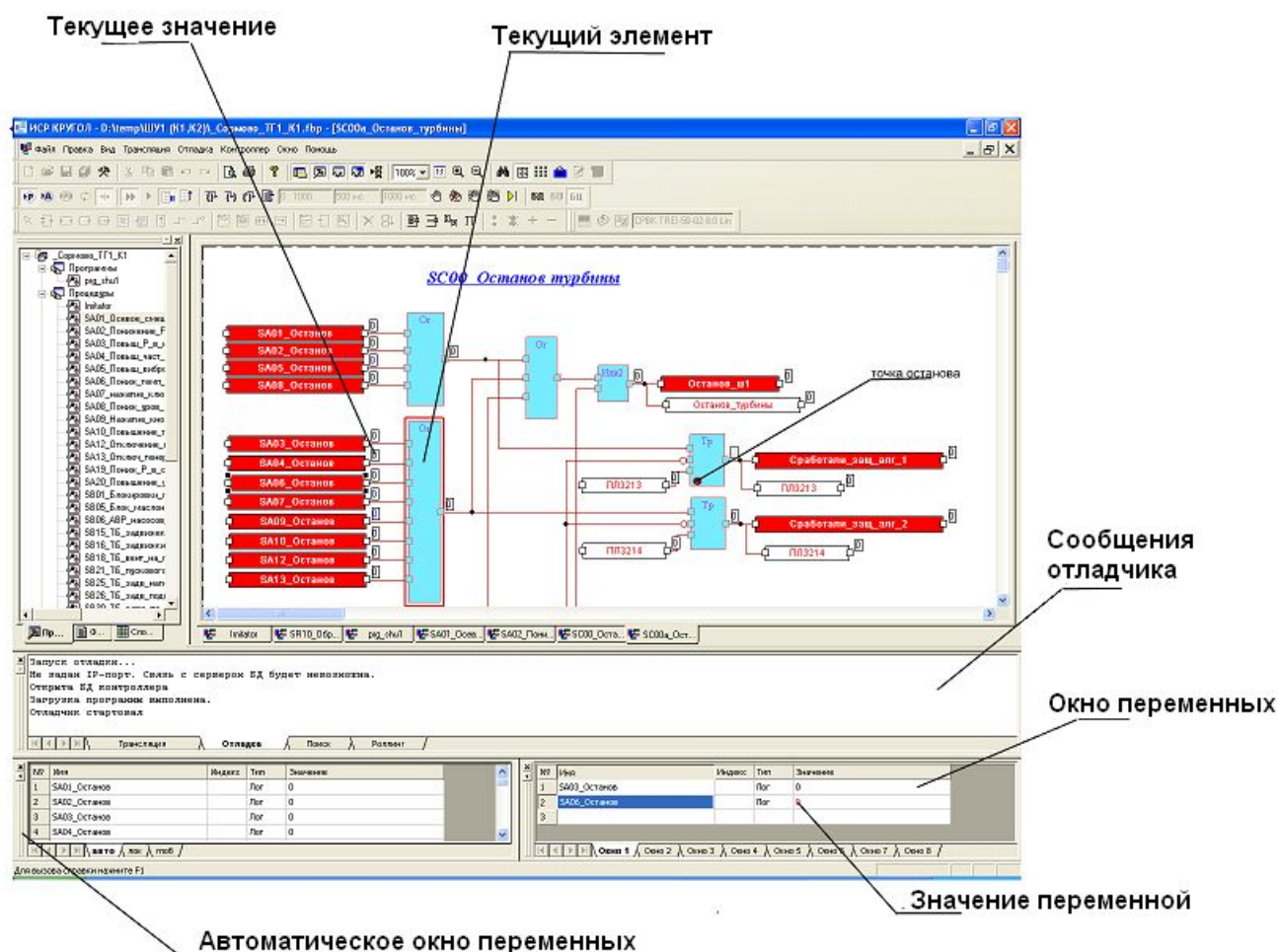


Рисунок 5.6.1 – Среда разработки КРУГОЛ в режиме отладки

На рисунке 5.6.1:

- Текущий выполняемый элемент помечается «маркером выполнения», для ФБД – это рамка красного цвета вокруг элемента, для СТ – желтый треугольник слева от строки
- Точки останова – небольшие красные круги
- Рядом с некоторыми элементами схемы ФБД выводятся их текущие значения
- В автоматическом окне переменных отображается информация о тех переменных, информация о которых доступна в данный момент (на рисунке «Авто» в нижней части окна).
- В окне переменных отображается информация о тех переменных, значения которых желает посмотреть пользователь (на рисунке «Окно1» в правой части окна).
- В окне вывода сообщений (в левой части окна) выводится текст сообщений самого отладчика.

ВНИМАНИЕ!

ИСР позволяет выключить режим указания текущей выполняемой строки или текущего выполняемого элемента. Для этого можно воспользоваться командой «С остановкой курсора» из меню «Отладка».

5.6.3 Режимы отладки

Отладчик ИСР КРУГОЛ поддерживает три режима отладки:

- Ручной
- Автоматический
- В реальном времени.

Ручной режим отладки

Установка ручного режима отладки осуществляется по команде **«Отладка\Режим отладки\Ручной»**.

В данном режиме Пользователю доступны следующие действия (команды меню «Отладка»):

- **Следующий шаг.** Выполняется следующий элемент схемы ФБД или текущая строка программы СТ
- **Следующий шаг с заходом.** Если текущий выполняемый элемент – вызов процедуры или пользовательской функции, то будет «произведен заход» в данную процедуру или функцию (с открытием ее схемы для ФБД или текста для СТ) и выполнение ее первого элемента. Далее отладка будет продолжаться до конца выполнения данной процедуры и возврата в вызывающую программу.
В остальных случаях данная команда работает так же, как и команда «Следующий шаг»
- **Следующий шаг с выходом.** Если текущий выполняемый элемент находится в теле процедуры или пользовательской функции, то данная процедура или функция будет выполнена до конца, после чего произойдет выход из нее в вызывающую программу.
В остальных случаях данная команда работает так же, как и команда «Следующий шаг».
- **Следующий цикл.** Программа выполняется до конца. Далее ее выполнение продолжится с самого начала – происходит переход на новый цикл выполнения.

Автоматический режим отладки

Установка автоматического режима отладки осуществляется по команде **«Отладка\Режим отладки\Автоматический»**.

Для удобства пользователя в ИСР предусмотрена функция включения/отключения слежения за текущим выполняемым элементом программы. Для задания этого режима необходимо воспользоваться командой **«С остановкой курсора»** из меню **«Отладка»**. При выборе этого режима курсор выполнения программы останавливается на первом элементе программы.

Для начала отладки необходимо выполнить команду **«Отладка\Выполнить»**.

Отладка в автоматическом режиме возможна двумя способами:

- **По шагам.** Выполнение одного элемента схемы ФБД потом другого или одной строки СТ затем другой. При этом переход от одного элемента к другому происходит с некоторой задержкой во времени – интервалом времени между шагами.
- **По циклам.** Выполнение программы происходит по циклам: в каждом цикле программа выполняется от начала до конца. При этом переход от одного цикла к другому происходит с некоторой задержкой во времени – интервалом времени между циклами.

Прервать отладку можно следующими способами:

- при помощи команды **«Отладка\Пауза»**. В этом случае выполнение будет приостановлено, и отладчик будет ожидать следующей команды Пользователя
- при помощи команды **«Отладка\Выход из отладки»**. В этом случае выполнение программы прервется и будет произведен выход из режима отладки.

Режим реального времени отладки

В данном режиме программа непрерывно выполняется на контроллере, а отладчик КРУГОЛ периодически опрашивает контроллер на предмет измененных значений переменных.

Установка режима реального времени отладки осуществляется по команде «**Отладка\Режим отладки\В реальном времени**».

Данный режим отладки доступен только в случае удаленной отладки на контроллере, параметры которого заданы на вкладке «Контроллер» в окне свойств проекта.

Прервать отладку можно следующими способами:

- при помощи команды «**Отладка\Пауза**». В этом случае выполнение будет приостановлено, и отладчик будет ожидать следующей команды Пользователя
- при помощи команды «**Отладка\Выход из отладки**». В этом случае выполнение программы прервется и будет произведен выход из режима отладки.

5.6.4 Изменение значений переменных

В режиме отладки имеется возможность изменять значения элементов типа «Переменная» схемы ФБД.

Для изменения значения переменной следует двойным щелчком мыши по нужному элементу схемы вызвать окно ввода нового значения (рисунок 5.6.2), в котором задать необходимое значение.

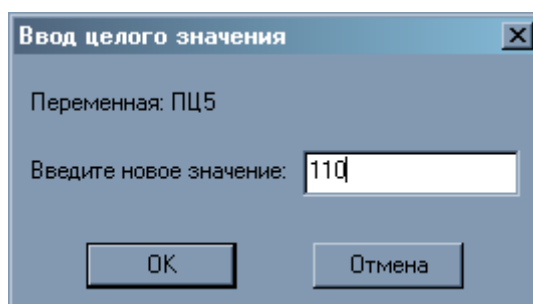


Рисунок 5.6.2 – Диалог для ввода нового значения

5.6.5 Связь с БД



ВНИМАНИЕ!

Работа в данном режиме возможна, только если ИСР установлена для работы со SCADA КРУГ-2000 и запущен Сервер БД!

Для обмена данными должен активизироваться пункт меню «**Отладка\Записывать в БД**». После установки этого параметра отладки текущие значения переменных и сообщения в роллинг передаются в БД.

Обмен с БД может осуществляться как после каждого шага, так и после каждого цикла отладки. Для этого предназначены две следующие команды:

- «**После каждого шага**» – значения переменных передаются в БД после каждого выполненного шага программы (очередного элемента схемы ФБД или строки программы СТ)
- «**После каждого цикла**» – значения переменных передаются в БД после каждого выполненного цикла программы.

5.6.6 Удаленная отладка

Удаленная отладка программы на контроллере может осуществляться в двух режимах: с остановкой контроллера и без остановки контроллера.

 **ВНИМАНИЕ!!!**

Режим удалённой отладки с остановкой контроллера доступен в СРВК версии не ниже 7.0, а режим удалённой отладки без остановки контроллера доступен, начиная с версии 8.0.

5.6.6.1 Отладка с остановкой контроллера

Для отладки программы в данном режиме следует:

- 1 В окне «**Свойства проекта**» на вкладке «**Контроллер**» задать параметры связи ИСР с контроллером: «**IP-адрес**» и «**Номер порта**»
- 2 Включить переключатель «**Удаленная отладка**»
- 3 Выбрать режим отладки «**С остановкой контроллера**»
- 4 Включить переключатель «**Инициализация среды исполнения при запуске**» при необходимости сброса в начальное состояние системных и внутренних переменных, а также алгоблоков СРВ контроллера
- 5 Включить переключатель «**Автоматическое обновление программ**», если необходимо отключить выдачу запроса пользователю о необходимости обновления программ пользователя в контроллере при несовпадении программ проекта и контроллера (рисунки 5.6.3, 5.6.4). В этом случае программы будут обновляться автоматически
- 6 Разрешить в контроллере удалённую отладку средствами модуля визуализации СРВК (смотрите документацию на соответствующую СРВК).
- 7 Запустить проект или программу на отладку. Если программы проекта совпадают с программами контроллера, программы контроллера продолжают выполняться, но уже в режиме отладки в соответствии с параметрами выполнения, заданными на вкладке «**Отладка**» окна «**Свойства проекта**». Если программы отличаются, то будет выведен запрос на обновление программ контроллера (рисунок 5.6.3). В случае подтверждения необходимо выбрать из списка (рисунок 5.6.4) программы, которые будут загружены в контроллер для последующей отладки. Порядок перечисления программ соответствует порядку их перечисления в окне проекта ИСР КРУГОЛ. После загрузки программ производится их запуск на выполнение в реальном времени.

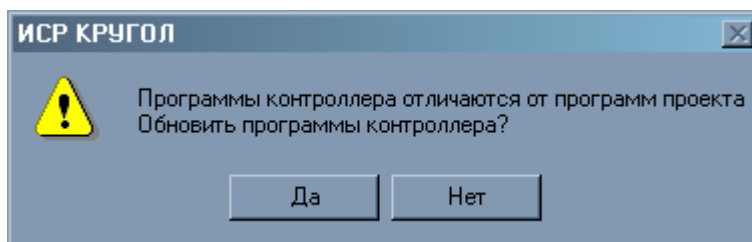


Рисунок 5.6.3 – Запрос на обновление программ

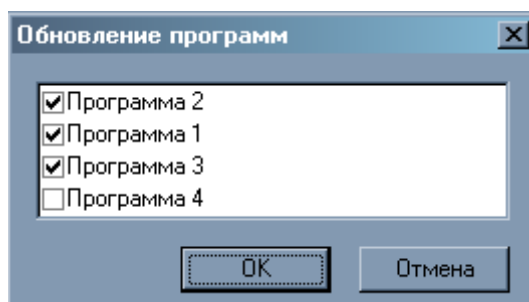


Рисунок 5.6.4 – Окно выбора программ для загрузки в контроллер

5.6.6.2 Отладка без остановки контроллера

Для отладки программы в режиме без остановки контроллера следует выполнить следующие действия:

- 1 В окне «Свойства проекта» на вкладке «Контроллер» задать параметры связи ИСР с контроллером («IP-адрес» и «Номер порта»)
- 2 Включить переключатель «Удаленная отладка»
- 3 Выбрать режим отладки «Без остановки контроллера»
- 4 Разрешить в контроллере удалённую отладку средствами модуля визуализации СРВК (смотрите документацию на соответствующую СРВК)
- 5 Запустить проект или программу на отладку. Затем пользователю будет предложено выбрать программы для загрузки в отладочную часть контроллера (рисунок 5.6.4). Если на контроллере уже была запущена отладочная часть СРВК и ее программы отличаются от программ проекта, то последовательно выводятся диалоговые окна обновления программ (рисунки 5.6.3, 5.6.4). После этого программы отладочной части СРВ контроллера запускаются на выполнение в реальном времени.
- 6 При завершении отладки будет выдан запрос (рисунок 5.6.5)

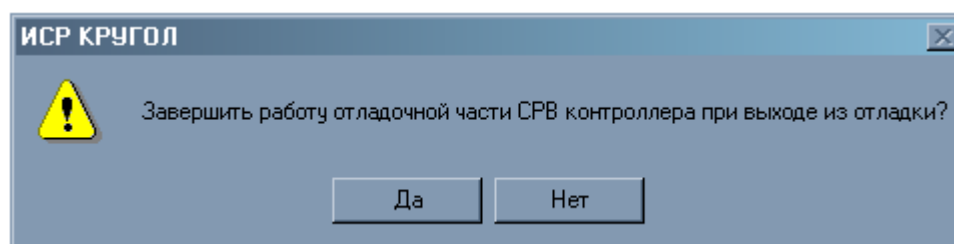


Рисунок 5.6.5 – Запрос на подтверждение завершения работы отладочной части СРВ контроллера

5.6.7 Вывод сообщений

В окно вывода сообщений на закладку «Отладка» выводятся сообщения о текущем состоянии отладчика (сообщения о старте отладчика, сообщения об успешном/неудачном запуске программ проекта, сообщения о прекращении процесса отладки и другие). На закладку «Роллинг» (рисунок 5.6.6) выводятся все сообщения функций **message**, **message2**, **message3** и элемента «Сообщение».



Рисунок 5.6.6 – Окно вывода сообщений

Для платформ СРВК версии 8.0 при удаленной отладке на закладку «Роллинг» так же выводятся сообщения из роллинга контроллера.

5.6.8 Окно переменных

Окно переменных предназначено для просмотра текущих значений переменных, используемых в программах проекта (в примере на рисунке 5.6.7 закладка «Окно 1»).

Имя	Тип	Значение
BA1.a23	Цел	41
ПЦ4	Цел	16
ПВ37	Вещ	0.450000
ПЛ105	Лог	0
ПЛ243	Лог	1

Рисунок 5.6.7 – Окно переменных

Каждая закладка окна переменных представляет собой таблицу, состоящую из трех полей:

- **«Имя»** – Пользователь вводит имена переменных, текущие значения которых он хотел бы отслеживать.
Если имя переменной введено неправильно или данная переменная не используется в проекте, то в полях **«Тип»** и **«Значение»** будет отображаться слово **«Ошибка»**
- **«Индекс»** – отображает текущее значение индекса переменной массива данных (не редактируется Пользователем)
- **«Тип»** – отображает тип переменной (не редактируется Пользователем)
- **«Значение»** – отображается текущее значение переменной.
В этом поле Пользователь может задать новые текущие значения переменных, которые будут учитываться в дальнейшем процессе отладки.
Значения, которые были изменены на последнем, предыдущем шаге выполнения программы, выделяются красным цветом (на рисунке 5.6.7 значение ПЛ243 – **1**).
Изменение значений переменных возможно только после приостановки отладки с помощью команды **«Отладка/Пауза»**. В противном случае изменение значения переменной невозможно.

Работа с таблицами осуществляется следующим образом:

- для ввода данных следует выделить нужную ячейку таблицы и ввести с клавиатуры требуемое значение, после чего нажать клавишу **«Enter»**.
Для более удобного ввода переменных данное окно поддерживает механизм Drag&Drop. То есть, для того, чтобы создать новый элемент для просмотра отслеживания его значения в таблице, достаточно с помощью мыши перетащить его с исходного текста СТ (имя переменной), схемы ФБД (элемент ФБД) или словаря пользовательских данных (строка таблицы словаря) в окно переменных. После этого для данного элемента в таблице будет создана строка с именем данного элемента и его текущими значениями.
Клавиша **«Esc»** отменяет введенные данные.
- для удаления переменной, значения которых не требуется просматривать, следует выделить ячейку с именем данной переменной и нажать клавишу **«Delete»** – строка с информацией о данной переменной удалится.

Окно переменных поддерживает возможность создания новых закладок, удаление или переименование существующих. Для этого используется контекстное меню, которое открывается при щелчке по активной закладке правой кнопкой мыши. Контекстное меню содержит команды:

- **Добавить закладку** – в окне переменных будет создана новая закладка с пустой таблицей
- **Удалить закладку** – текущая активная закладка будет удалена

- **Переименовать закладку** – вызов окна «Переименовать закладку» с предложением ввести новое имя закладки (рисунок 5.6.8).

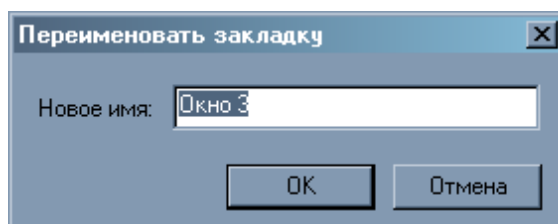


Рисунок 5.6.8 – Окно переименования закладки

5.6.9 Автоматическое окно переменных

Автоматическое окно переменных (рисунок 5.6.9) предназначено для просмотра текущих значений переменных, доступных на текущем шаге отладки.

№	Имя	Индекс	Тип	Значение
1	вд130		Лог	0
2	вд131		Лог	0
3	вд152		Лог	0
4	вд153		Лог	0

Рисунок 5.6.9 – Автоматическое окно переменных

Для отображения автоматического окна переменных необходимо перейти в режим отладки и воспользоваться командой «**Автоматическое окно переменных**» в меню «**Вид**»

В отличие от наборного окна переменных, в автоматическое окно переменных нельзя добавлять, удалять или переименовывать закладки и изменять список переменных в закладках. Список переменных в данном окне формируется автоматически.

Автоматическое окно переменных имеет 3 закладки:

- «**Авто**» – отображается список переменных, значения которых доступны на данном шаге отладки
- «**Лок**» – отображаются локальные переменные
- «**Глоб**» – отображаются глобальные переменные.

5.7 Программирование контроллера

Программирование контроллера заключается в загрузке программ пользователя в контроллер и обновлении базы данных системных переменных.

Для программирования контроллера следует:

- 1 В окне «**Свойства проекта**» на вкладке «**Контроллер**» задать параметры связи ИСР с контроллером («**IP-адрес**» и «**Номер порта**»).
- 2 Выбрать команду «**Программирование**» меню «**Контроллер**»
- 3 В открывшемся окне «**Программирование контроллера**» (рисунок 5.7.1) установить переключатель «**Обновление программ**» и отметить в списке программы, загружаемые в контроллер. Порядком перечисления отмеченных программ будет определяться их порядковый номер при выполнении на контроллере (аналогично перечислению программ в файле *programs.lst* контроллера)

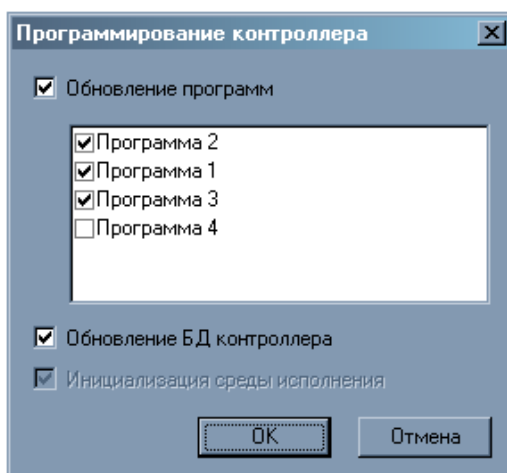


Рисунок 5.7.1 - Окно « Программирование контроллера»

Если переключатель **«Обновление программ»** выключен (при этом список программ становится недоступным), то загрузка программ в контроллер не производится

- 4 Для загрузки системной базы данных в контроллер «включить» переключатель **«Обновление БД контроллера»**
- 5 Для выполнения сброса в начальное состояние системных и внутренних переменных, а также алгоблоков СРВ контроллера установить переключатель **«Инициализация среды исполнения»**. Если было выбрано обновление БД контроллера, то данный переключатель будет принудительно включен и станет недоступным для редактирования
- 6 Нажать на кнопку **«ОК»**. В результате старые программы контроллера будут выгружены, новые загружены и запущены на выполнение. При этом если выбран режим «Обновление БД контроллера», то будет автоматически произведен перезапуск контроллера.

**ВНИМАНИЕ!!!**

Программирование контроллера из ИСР КРУГОЛ доступно для платформ СРВК, начиная с версии 7.0. Программирование контроллера возможно только при включенном режиме удалённой отладки в СРВК (подробнее в документации на СРВК).

5.8 Поиск задействованных объектов в проекте

С помощью функции поиска задействованных объектов в проекте можно получить информацию об объектах, используемых в программе. В данном случае под понятием «объект» подразумевается любая переменная, константа, процедура, функция, алгоблок, программа.

Для доступа к данной функции необходимо воспользоваться командой **«Перекрёстные ссылки»** меню **«Вид»**. Вид окна поиска задействованных объектов приведён на рисунке 5.8.1.

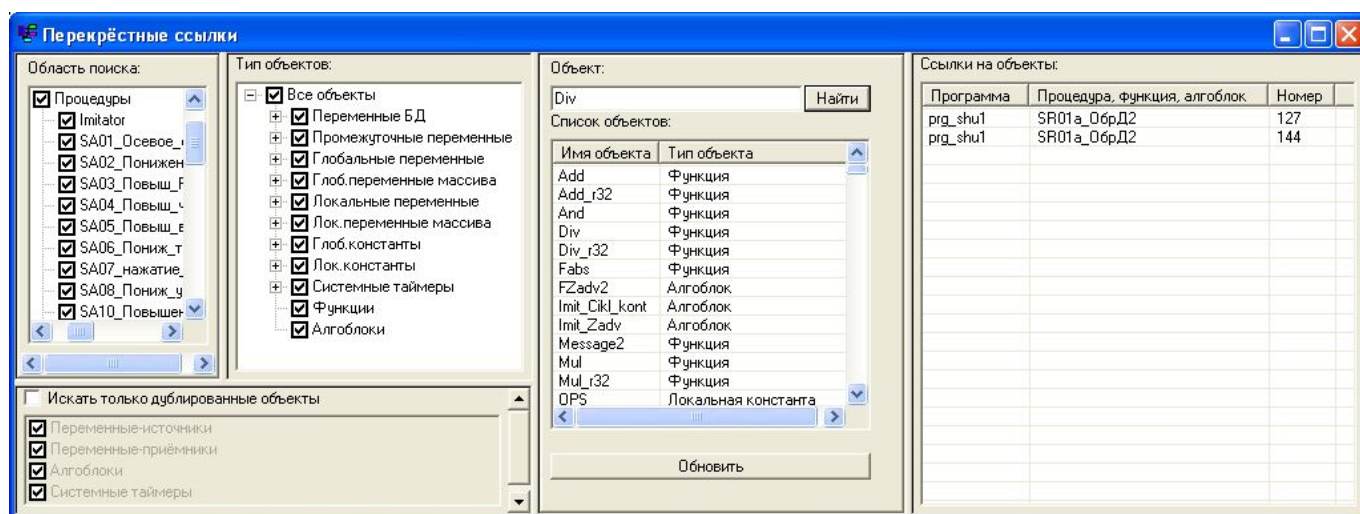


Рисунок 5.8.1 – «Окно поиска задействованных объектов»

5.8.1 Настройка критериев поиска

Диалог поиска задействованных объектов позволяет пользователю произвести тонкую настройку критериев поиска объектов в проекте. Возможна настройка следующих критериев:

- Область поиска объектов
- Типы объектов
- Типы дублированных объектов.

5.8.1.1 Настройка области поиска объектов

Для задания области поиска необходимо установить флажки в дереве «**Область поиска**» (рисунок 5.8.2) напротив соответствующего элемента дерева, в котором отображены все программы проекта, процедуры, функции, алгоблоки, используемые в программе

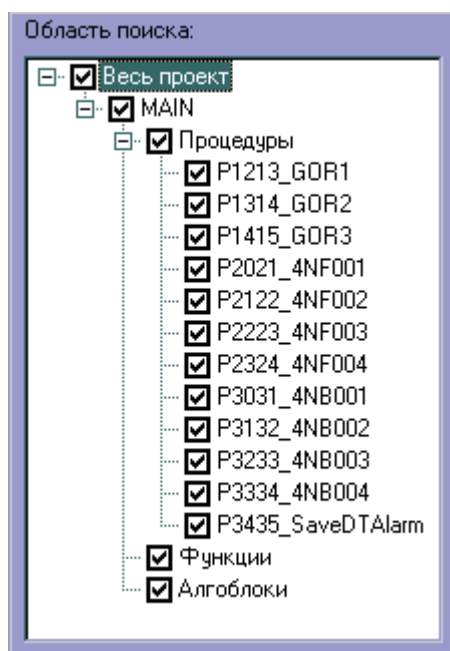


Рисунок 5.8.2 – «Область поиска»

5.8.1.2 Задание типов объектов для поиска

Для задания типов объектов для поиска необходимо расставить флажки в дереве «**Типы объектов**» (рисунок 5.8.3) напротив элементов дерева, соответствующих типам объектов, выбранных для поиска.

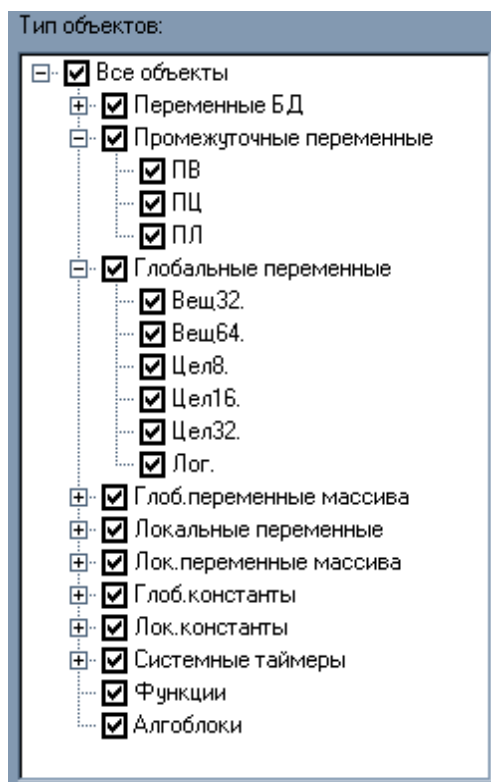


Рисунок 5.8.3 – «Типы объектов»

5.8.1.3 Настройка поиска только дублированных объектов

Под понятием «дублированный объект» (рисунок 5.8.4) подразумевается объект проекта, к которому происходит обращение более одного раза. Возможно, осуществить поиск следующих дублированных объектов:

- Переменные-приёмники
- Переменные-источники
- Алгоблоки
- Системные таймеры

Для выбора того или иного класса дублированных объектов необходимо установить флажок напротив соответствующего элемента в списке «**Искать только дублированные объекты**» (рисунок 5.8.1), а затем напротив названия соответствующего класса объектов в ставшем активном списке.

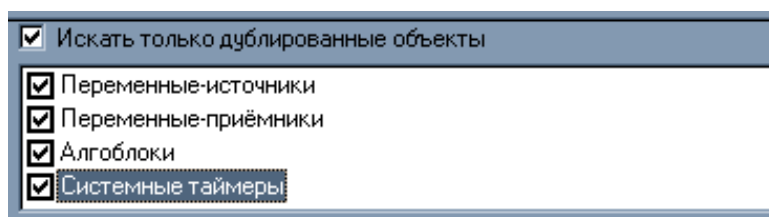


Рисунок 5.8.4 – «Дублированные объекты»

5.8.2 Выполнение поиска задействованных объектов

Для поиска задействованных объектов следует задать критерии поиска (подраздел 5.8.1) и затем нажать на кнопку **«Обновить»**. Результаты поиска (имя объекта и его тип) отображаются в списке объектов (рис. 5.8.5).

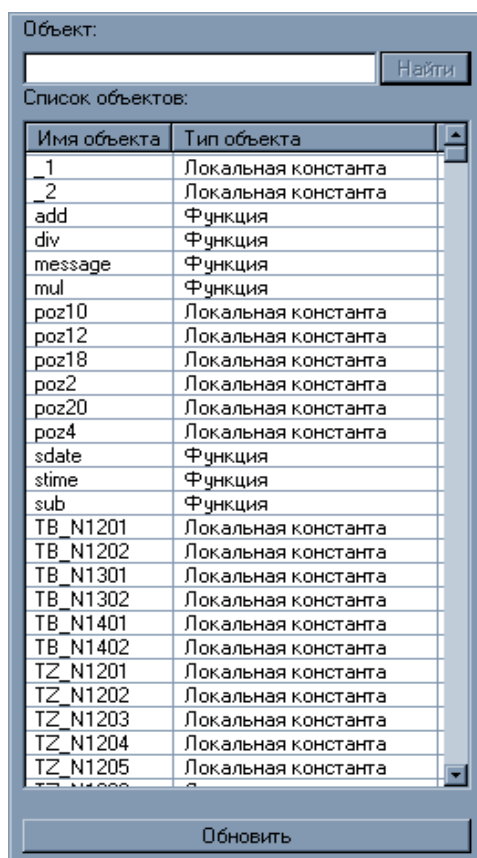


Рисунок 5.8.5 – «Найденные объекты»

В списке найденных объектов может оказаться большое число объектов. Поэтому для поиска объекта по имени среди найденных объектов необходимо ввести текст в поле **«Объект»** и нажать на кнопку **«Найти»**.

В случае, если пользователь изменил параметры поиска или изменилась структура проекта, в окне высвечивается индикатор **«Данные изменены»**.



ВНИМАНИЕ!!!

Алгоблоки с ручной нумерацией отображаются в формате: <Имя алгоблока>(<Номер алгоблока>). В алгоблоках с автоматической нумерацией номер не отображается.

Список ссылок на найденный объект отображается в таблице **«Ссылки на объекты»** (рисунок 5.8.6).

Ссылки на объекты:		
Программа	Процедура, функция, алгоблок	Номер
MAIN	P2122_4NF002	35
MAIN	P2324_4NF004	31
MAIN	P3233_4NB003	13
MAIN	P3233_4NB003	56
MAIN	P3435_SaveDTAlarm	18

Рисунок 5.8.6 – «Ссылки на объекты»

В этой таблице содержится следующая информация:

- Имя программы, в которой имеется обращение к объекту
- Имя процедуры, функции или алгоблока, в которых имеется обращение к объекту
- Номер элемента схемы ФБД или строка в файле на СТ.

Для того чтобы вывести список ссылок на соответствующий объект, необходимо выбрать его в списке найденных объектов с помощью мыши или клавиатуры. В окне поиска задействованных объектов предусмотрена функция быстрого перехода к месту обращения к объекту. Чтобы осуществить подобный переход надо выполнить двойной щелчок мышью по соответствующей ссылке в списке ссылок на объект.

ВНИМАНИЕ!!!

Если проект содержит синтаксические ошибки в исходном коде, выдаётся сообщение о том, что поиск задействованных объектов может быть не достоверным. В этом случае поиск выполняется только в той части проекта, в которой не содержится ошибок.

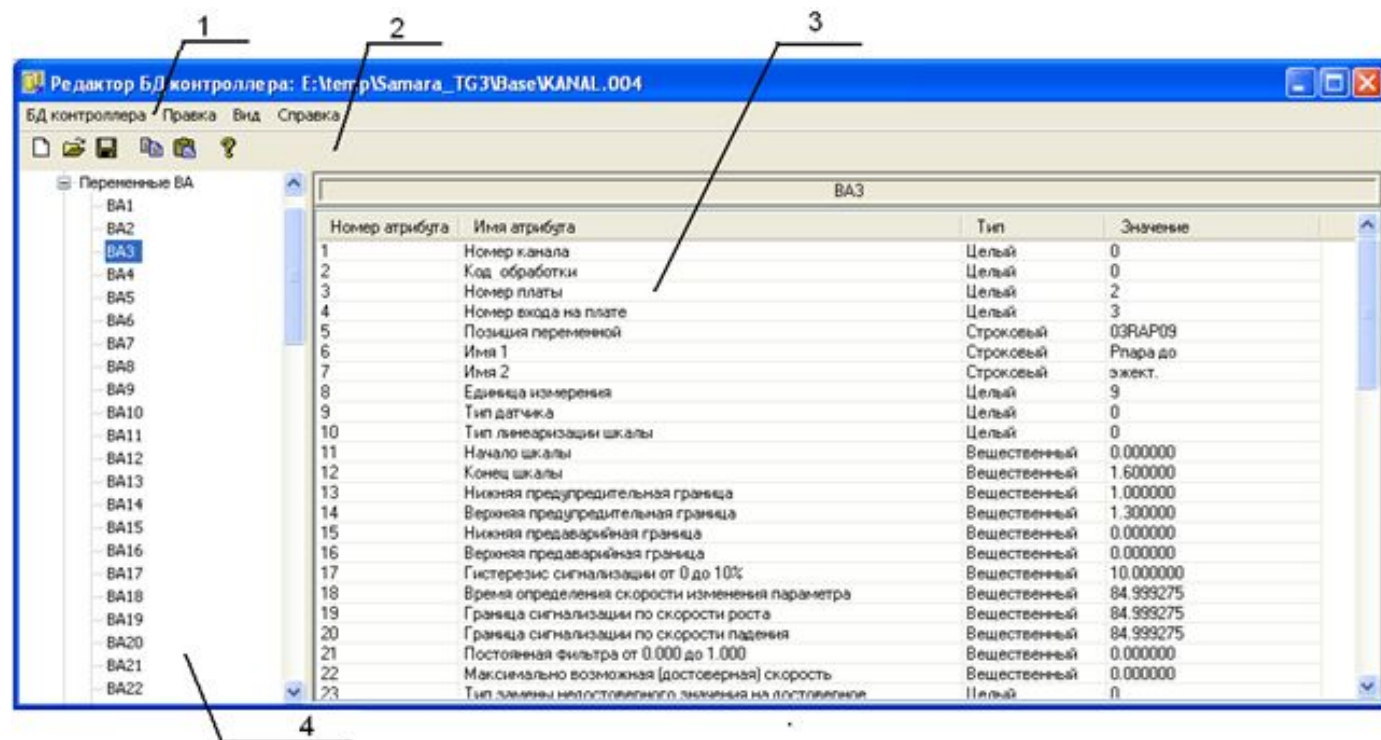
5.9 Редактирование базы данных контроллера

5.9.1 Редактор базы данных контроллера

Для редактирования БД контроллера следует вызвать Редактор базы данных.

Вызов редактора осуществляется из меню ИСР КРУГОЛ «Контроллер/Редактирование БД контроллера...».

Главное окно редактора БД контроллера показано на рисунке 5.9.1.



1 - Главное меню

2 - Панель инструментов

3 - Панель отображения контекстнозависимой информации

4 - Список переменных

Рисунок 5.9.1 – Редактор БД контроллера. Главное окно

5.9.2 Создание БД контроллера

Для создания БД контроллера следует:

- 1 В меню «**БД контроллера**» Редактора БД выполнить команду «**Создать**»
- 2 В появившемся окне «**Создать БД...**» (рисунок 5.9.2) задать следующие параметры:
 - Имя БД контроллера
 - Путь к БД контроллера
 - Количество переменных каждого типа в БД
 - Признак поддержки 24-символьной позиции
 - Признак поддержки протокола событий версии 2.0.

Создать БД...

Имя базы данных контроллера: test

Путь: c:\

Переменные ВА
Количество переменных 5

Переменные АВ
Количество переменных 5

Переменные ДВ
Количество переменных 5

Переменные ВД
Количество переменных 11

Переменные РВ
Количество переменных 11

Дополнительно:
☒ Поддержка 24-х символьной позиции
☒ Поддержка протокола событий версии 2.0 (для SCADA КРУГ-2000 3.0 и выше)

OK Отмена

Рисунок 5.9.2 – Параметры БД контроллера

При попытке создать БД с некорректными параметрами выдаётся сообщение о том, что введены некорректные данные.

5.9.3 Открытие БД контроллера

Открыть базу данных контроллера возможно двумя способами:

- Вызвать Редактор БД из главного меню ИСР. При этом откроется БД контроллера, путь к которой указан в настройках проекта
- Выполнить команду «**Открыть**» меню «**БД контроллера**» Редактора БД. При этом пользователю будет предложено указать путь к БД. Диалог открытия БД в этом случае представляет собой стандартный диалог выбора папки в Windows.

5.9.4 Сохранение БД контроллера.

Для сохранения БД контроллера следует выполнить команду «Сохранить» меню «Файл» Редактора БД.

5.9.5 Редактирование БД контроллера.

Редактор БД позволяет осуществлять следующие операции:

- **Изменение глобальных настроек БД.** Производится путём выбора элемента «БД контроллера» в списке переменных. После чего на панели отображения контекстнозависимой информации появится диалог, в котором возможно установить признаки поддержки 24-х символьной позиции и поддержки протокола событий версии 2.0. Внешний вид диалога приведён на рисунке 5.9.3.

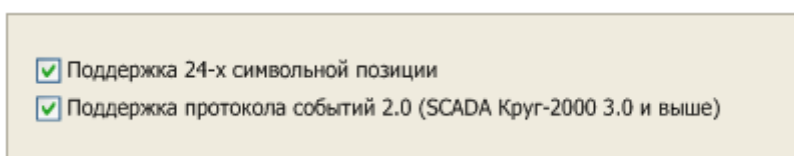


Рисунок 5.9.3 – Изменение настроек базы данных

- **Изменение количества переменных БД.** Производится путём выбора элемента с именем «Переменные <Тип переменной>» в списке переменных. После выбора соответствующего элемента в правой части главного окна приложения отображается диалог, в котором можно изменить количество переменных определённого типа. В случае увеличения количества переменных происходит добавление новых переменных в конец списка переменных данного типа. Если количество переменных изменилось в сторону уменьшения, то происходит удаление переменных из конца списка. Количество переменных, которое необходимо добавить или удалить рассчитывается как абсолютная величина разности между "новым" и "старым" значением количества переменных. Диалог изменения количества переменных приведен на рисунке 5.9.4:



Рисунок 5.9.4 – Изменение количества переменных базы данных



ВНИМАНИЕ!!!

Суммарное количество переменных не должно превышать значение, указанное в электронном ключе защиты.

- **Изменение данных в паспорте переменной.** Для просмотра и редактирования паспорта переменной БД следует:
 - Выбрать переменную списке переменных. При этом в правой части окна редактора БД отображается диалог редактирования паспорта переменной (рисунок 5.9.5)
 - Изменить значение атрибута выбранной переменной в столбце «Значение». Изменённое значение атрибута применяется только после нажатия клавиши ввода или перевода фокуса ввода на другой атрибут.
 Некоторые атрибуты переменных могут иметь фиксированный набор значений (например, атрибуты «Код логического состояния», «Единицы измерения»).

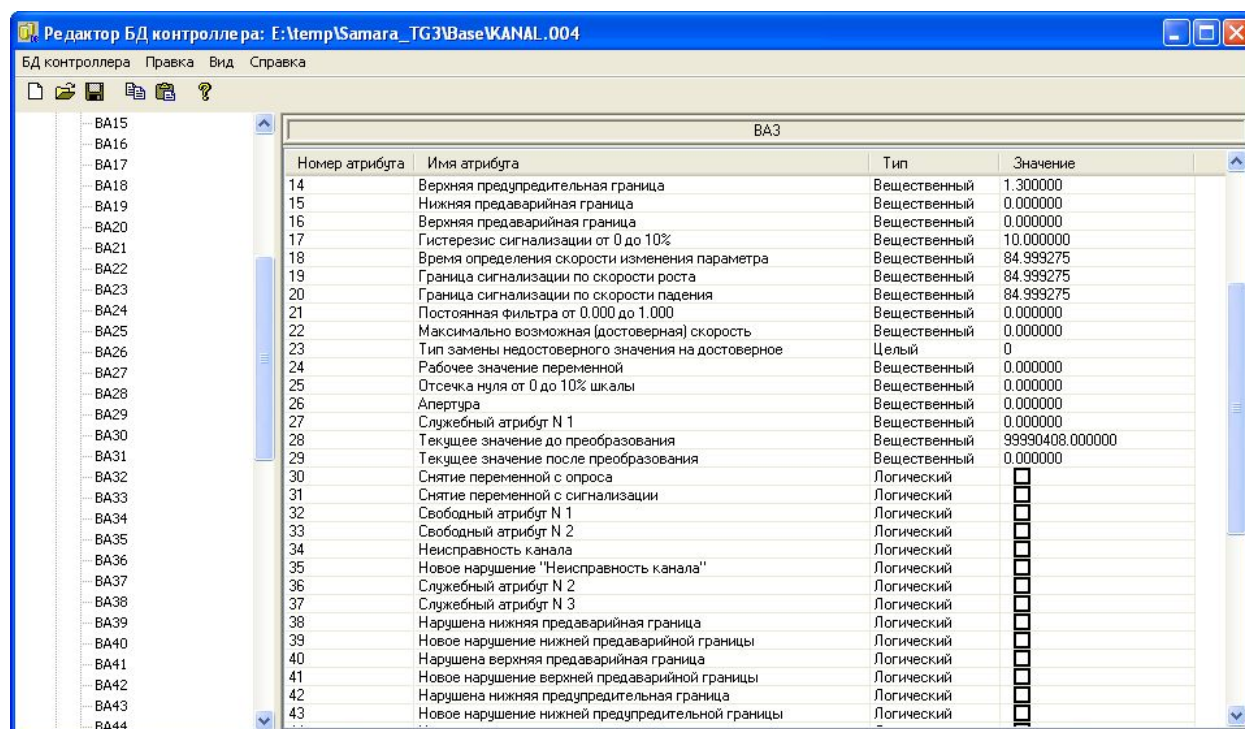


Рисунок 5.9.5 – Изменение паспорта переменной

Изменить значение данных паспорта переменной можно с помощью копирования паспорта из другой переменной. Для этого следует:

- 1 Выбрать переменную в списке
- 2 Выполнить команду «Копировать» меню «Правка»
- 3 Выбрать в списке переменную, паспорт которой изменяется
- 4 Выполнить команду «Вставить» меню «Правка»

5.9.6 Атрибуты переменных БД

5.9.6.1 Аналоговая выходная переменная

Атрибут №1

№ КАНАЛА – по умолчанию значение атрибута для контроллера равно 0

Атрибут №2

КОД ОБРАБОТКИ – определяет возможность управления алгоритмами обработок переменной в СРВК. При создании базы данных по каналу значение данного атрибута записывается в атрибут №2 – "Код обработки" переменной СРВК. При значении 0 – в СРВК выполняются стандартные обработки переменной в полном объеме. Подробное описание использования данного атрибута смотрите в приложении Б.1 руководства Пользователя для СРВК версии 7.0 и выше.

Атрибут №3

№ ПЛАТЫ – номер платы в УСО, к которой подключена данная переменная, 0 – если переменная «виртуальная» в контроллере (не имеет физического выхода).

Атрибут №4

№ ВЫХОДА НА ПЛАТЕ – номер выхода на плате УСО для физического подключения переменной, 0 – если переменная «виртуальная» в контроллере (не имеет физического выхода, используется для построения каскадных и других, более сложных схем регулирования). Правила привязки к физическим выходам контроллеров при использовании импульсных типов регуляторов представлены в приложении к документации соответствующего УСО.

Атрибут №5

ПОЗИЦИЯ ПЕРЕМЕННОЙ – код параметра, в качестве которого обычно используются первые 8 символов технологической (или КИП и А) позиции данного параметра (например: 302PICAE).

Атрибут №6

ИМЯ1 – это первые 8 символов в названии параметра, если для БД не установлен признак поддержки 24-хсимвольной позиции, или вторые 8 символов технологической позиции параметра, если признак поддержки 24-хсимвольной позиции установлен.

Атрибут №7

ИМЯ2 – это вторые 8 символов в названии параметра, если для БД не установлен признак поддержки 24-хсимвольной позиции, или третьи 8 символов технологической позиции параметра, если признак поддержки 24-хсимвольной позиции установлен которые в дальнейшем можно использовать для отображения на станции оператора.

Атрибут №8

ТИП ПЕРЕМЕННОЙ 1 – тип переменной, являющейся для данного регулятора регулируемым параметром. Выбирается из выпадающего списка типов переменных. В качестве типа переменной может использоваться только входная аналоговая переменная, причем эта переменная может быть как измеряемая, так и рассчитанная на языке КРУГОЛ.

Атрибут №9

№ ПЕРЕМЕННОЙ 1 – номер переменной, являющейся для данного регулятора регулируемым параметром. Тип этой переменной задается атрибутом №8.

Атрибут №10

ТИП ПЕРЕМЕННОЙ ЗАДАНИЯ – тип переменной, являющейся для данного регулятора сигналом внешнего задания. Т.е. текущее значение данной переменной используется в качестве задания для ведомого регулятора при его работе в схеме каскадного управления. Выбирается из выпадающего списка типов переменных. В качестве типа переменной может использоваться входная (значение данного атрибута равно 1) или выходная аналоговая (значение данного атрибута равно 8) переменная, причем эта переменная может быть как измеряемая, так и рассчитанная на языке КРУГОЛ. Диапазон изменения данной переменной от 0 до 100 %.

Атрибут №11

НОМЕР ПЕРЕМЕННОЙ ЗАДАНИЯ – номер переменной, являющейся для данного регулятора сигналом внешнего задания. Тип этой переменной задается атрибутом №10.

**ВНИМАНИЕ !!!**

В контроллерах, для того, чтобы внешнее задание учитывалось регулятором при расчете рассогласования в автоматическом режиме управления, необходимо перевести регулятор в режим работы с внешним заданием, т.е. установить значение атрибута «РЕЖИМ ВВОДА ЗАДАНИЯ <ВНЕШНИЙ>(<КАСКАД>)» = 1.

Если Вам необходимо создать стандартный одноконтурный регулятор, то этот атрибут описывать не надо. При этом значение задания будет локальным и первоначально определяется атрибутом «Величина задания». Все дальнейшие изменения задания записываются в этот же атрибут, поэтому даже при перезапуске контроллера значение задания будет соответствовать последнему заданному значению на момент перезапуска.

Атрибут №12

ТИП ПЕРЕМЕННОЙ РУЧНОГО ЗАДАТЧИКА/ПОЛОЖЕНИЕ МЭО - тип переменной, являющейся для регулятора типа 0 сигналом управления ИМ от внешнего задатчика (т.е. текущее значение данной переменной используется для формирования выходного сигнала на ИМ в режиме ручного аппаратного управления). При использовании импульсных типов регулятора - на данный вход может заводиться сигнал положения МЭО. В качестве типа переменной может использоваться только входная аналоговая переменная, причём эта переменная может быть как измеряемая, так и рассчитанная на языке КРУГОЛ. Диапазон изменения данной переменной от 0% до 100%.

Атрибут №13

НОМЕР ПЕРЕМЕННОЙ РУЧНОГО ЗАДАТЧИКА/ПОЛОЖЕНИЕ МЭО - номер переменной, являющейся для регулятора типа 0 сигналом управления ИМ от внешнего физического задатчика в режиме ручного аппаратного управления.

Атрибут №14

ТИП ПЕРЕМЕННОЙ ПРИЗНАКА АППАРАТНОГО УПРАВЛЕНИЯ – тип переменной, являющейся для данного регулятора сигналом переключения режимов работы регулятора от внешнего дискретного сигнала. Выбирается из выпадающего списка типов переменных. В качестве типа переменной может использоваться только входная дискретная переменная (при этом значение данного атрибута равно 2), причем эта переменная может быть как измеряемая, так и рассчитанная на языке КРУГОЛ. При значении данной переменной =1 – устанавливается ручной аппаратный режим управления, при значении переменной =0 – автоматический аппаратный режим управления.

Атрибут №15

НОМЕР ПЕРЕМЕННОЙ ПРИЗНАКА АППАРАТНОГО УПРАВЛЕНИЯ – номер переменной, являющейся для данного регулятора сигналом переключения режимов работы регулятора от внешнего дискретного сигнала. Тип этой переменной задается атрибутом №14. Выбирается из выпадающего списка переменных заданного типа. Данная переменная должна находиться в том же УСО, что и регулируемый параметр.

Атрибут №16

ТИП ПЕРЕМЕННОЙ 2 – тип переменной, являющейся дополнительной переменной, которая может быть использована в алгоритме регулирования. При этом сам алгоритм регулирования с участием этой переменной описывается с помощью языка КРУГОЛ. Выбирается из выпадающего списка типов переменных. В качестве типа переменной может использоваться только входная аналоговая переменная (при этом значение данного атрибута равно 1), причем эта переменная может быть как измеряемая, так и рассчитанная на языке КРУГОЛ.

Атрибут №17

№ ПЕРЕМЕННОЙ 2 – номер переменной, являющейся дополнительной переменной. Тип этой переменной задается атрибутом №16.

Атрибут №18

ТИП ПЕРЕМЕННОЙ 3 – тип переменной, являющейся дополнительной переменной, которая может быть использована в алгоритме регулирования. При этом сам алгоритм регулирования с участием этой переменной описывается с помощью языка КРУГОЛ. Выбирается из выпадающего списка типов переменных. В качестве типа переменной может использоваться только входная аналоговая переменная (при этом значение данного атрибута равно 1), причем эта переменная может быть как измеряемая, так и рассчитанная на языке КРУГОЛ.

Атрибут №19

№ ПЕРЕМЕННОЙ 3 – номер переменной, являющейся дополнительной переменной. Тип этой переменной задается атрибутом №18.

Атрибут №20

ТИП РЕГУЛЯТОРА (АНАЛОГОВЫЙ, ИМПУЛЬСНЫЙ) – тип алгоритма регулятора. Алгоритмы регуляторов реализованы в ПО контроллера. Подробное описание работы регуляторов см.: Система реального времени контроллера TREI-05B-02. Алгоритм регулятора может быть:

- «Базовым» - алгоритм регулятора определяется жестко его типом, а именно:
 - 0 – аналоговый
 - 1 – импульсный
 - 2 – импульсный
 - 3 – Ремиконт
 - 4 – импульсный (должен использоваться только с модулями ввода/вывода, поддерживающими аппаратный ШИМ или для «виртуальных» регуляторов)
 - 5 – импульсный (должен использоваться только с модулями ввода/вывода, поддерживающими аппаратный ШИМ или для «виртуальных» регуляторов)
 - 6 – импульсный (должен использоваться только с модулями ввода/вывода, поддерживающими аппаратный ШИМ или для «виртуальных» регуляторов)
 - 7 – импульсный (должен использоваться только с модулями ввода/вывода, поддерживающими аппаратный ШИМ или для «виртуальных» регуляторов)
 - 8 – импульсный
 - 9 – импульсный
 - 10 – импульсный (должен использоваться только с модулями ввода/вывода, поддерживающими аппаратный ШИМ или для «виртуальных» регуляторов)
 - 11 – импульсный (должен использоваться только с модулями ввода/вывода, поддерживающими аппаратный ШИМ или для «виртуальных» регуляторов)
 - 12 – импульсный
- «Пользовательским» - алгоритм регулятора определяется программой Пользователя, реализованной на языке КРУГОЛ, а именно:
 - 100 – аналоговый
 - 101 – импульсный

Атрибут №21

ВЕЛИЧИНА ЗАДАНИЯ - это начальное значение задания регулятора, если он находится в режиме локального автоматического управления. Все дальнейшие изменения задания записываются в это же поле, поэтому, даже при перезапуске контроллера, значение задания будет соответствовать последнему на момент перезапуска.

Атрибут №23

ВЕЛИЧИНА КОЭФФИЦ. ПРОПОРЦИОН. - параметр настройки регулятора. Коэффициент пропорциональности регулятора определяется по формуле:

$$КП = 100 / K_y, (\%)$$

где K_y - коэффициент усиления,
 $КП = 0$ - отключение пропорциональной части регулятора

Атрибут №24

ПОСТОЯННАЯ ВРЕМЕНИ ИНТЕГРИРОВ - параметр настройки регулятора. Постоянная времени интегрирования в секундах.

Т_и = 0 - отключение интегральной части регулятора.

Атрибут №25

ВЕЛИЧИНА КОЭФФИЦ. ДИФФЕРЕНЦ. - параметр настройки регулятора. Коэффициент предварения (дифференцирования) может быть положительным (прямое предварение) и отрицательным (обратное предварение).

К_д = 0 - отключение дифференциальной части регулятора.

Атрибут №26

ЗОНА НЕЧУВСТВИТЕЛЬНОСТИ - параметр настройки регулятора. Величина отклонения регулируемого параметра от задания в любую из сторон (задается в процентах от диапазона измерения регулируемого параметра). В пределах зоны нечувствительности рассогласование рассчитывается с учётом коэффициента ослабления для зоны нечувствительности.

Атрибут №27

КОЭФФИЦ. ОСЛАБЛ. ДЛЯ ЗОНЫ НЕЧУВСТВ. - параметр настройки регулятора. Перерасчет величины рассогласования в рамках зоны нечувствительности осуществляется по формуле:

РАССОГЛАСОВАНИЕ расч. = К * РАССОГЛАСОВАНИЕ действ.,

где **К** - коэффициент ослабления (от 0 до 1)

Атрибут №28

ВЕРХНЯЯ ГРАНИЦА ОТКЛОНЕНИЯ ОТ ЗАДАНИЯ - это величина допустимого положительного отклонения регулируемого параметра от задания, при достижении которой включится сигнализация по отклонению от задания (задается в единицах измерения регулируемого параметра).

Атрибут №29

НИЖНЯЯ ГРАНИЦА ОТКЛОНЕНИЯ ОТ ЗАДАНИЯ - это величина допустимого отрицательного отклонения от задания, при достижении которой включится сигнализация по отклонению от задания (задается в единицах измерения регулируемого параметра).

Атрибут №30

ВЕРХНЕЕ ОГРАНИЧ. ХОДА ИМ - параметр настройки. Для аналогового регулятора - это верхнее ограничение значения выходного сигнала регулятора (в процентах) при автоматическом режиме управления. Для импульсных регуляторов 1, 2, 4, 5, 6, 7, 8, 9, 10, 11 заполнение данного атрибута не требуется.

Атрибут №31

НИЖНЕЕ ОГРАНИЧ. ХОДА ИМ/ПР УПРАВЛЕНИЕ ИМ - параметр настройки. Для аналогового регулятора - это нижнее ограничение значения выходного сигнала регулятора (в процентах) при автоматическом режиме управления. Для импульсных регуляторов 1 и 2 необходимо записать константу 0,0. Для импульсных регуляторов 4 и 5 значение данного поля – задание для алгоритма псевдоручного управления ИМ (алгоритм вывода ИМ в заданное положение в режиме регулятора «Ручной дистанционный», подробнее о работе данного алгоритма смотри приложение Е). Для импульсных регуляторов 6, 7, 8 и 9 заполнение данного атрибута не требуется.

Атрибут №32

ВЕРХНЯЯ ГРАНИЦА СИГНАЛИЗ. ХОДА ИМ/ ЛЮФТ 'БОЛЬШЕ' - это величина выходного сигнала (в процентах) регулятора типа 0, выше которого сработает предупредительная сигнализация по положению ИМ. Для импульсных типов регуляторов 1, 2, 6, 7, 8, 9 заполнение данного атрибута не требуется. Для импульсных типов регуляторов 4 и 5 данный атрибут

используется в качестве значения длительности команды выборки люфта ИМ в сторону «Больше» (дополнительный импульс в секундах, который будет выдан на ИМ при подаче очередной команды «Больше», если предыдущая команда была «Меньше».

Атрибут №33

НИЖНЯЯ ГРАНИЦА СИГНАЛИЗ. ХОДА ИМ/ ЛЮФТ 'МЕНЬШЕ' - это величина выходного сигнала (в процентах) регулятора типа 0, ниже которого сработает предупредительная сигнализация по положению ИМ. Для импульсных регуляторов 4, 5, 10, 11 данный атрибут используется в качестве значения длительности команды выборки люфта ИМ в сторону «Меньше» (дополнительный импульс в секундах, который будет выдан на ИМ при подаче очередной команды «Меньше», если предыдущая команда была «Больше». Для импульсных типов регуляторов 1, 2, 6, 7, 8, 9 заполнение данного атрибута не требуется.

Атрибут №34

ТАКТОВАЯ ЧАСТОТА / ДЛИТЕЛЬНОСТЬ ИМПУЛЬСА - параметр настройки (задается в секундах). Для регуляторов типа 0, 1 и 2 - определяет период выполнения алгоритма регулятора (он должен задаваться кратным времени цикла опроса контроллера). Для типов регуляторов 4, 5, 6, 7, 10, 11 данный атрибут является минимальной длительностью импульса, выдаваемой на ИМ (алгоритм регулятора в данном случае выполняется в каждом такте контроллера). Для типов регуляторов 8 и 9 данный атрибут является периодом выполнения алгоритма, заданным количеством циклов контроллера (при задании не целого числа в данном случае оно округляется в большую сторону).

Атрибут №35

ПЕРЕХОД К НОВОМУ ЗАДАНИЮ - режимы перехода к новому заданию:

- 0 - обычный скачкообразный переход к новому заданию,
- 1 - «плавный» переход к новому заданию (смотри атрибут **ПОСТОЯННАЯ ВРЕМЕНИ ПО ЗАДАНИЮ**),
- 2 - «форсированный» переход к новому заданию (смотри атрибут **КОЭФФИЦИЕНТ ДЛЯ ФОРСИРОВАННОГО ПЕРЕХОДА**),
- 3 - режим «подхватывания» (безударного перехода) заданием последнего значения регулируемого параметра при переходе на **«АВТОМАТ»**, а в ручном дистанционном режиме управления сигнал задания всегда отслеживает (равен) значение регулируемого параметра,
- 13 - одновременно выполняются режимы «1» и «3»,
- 23 - одновременно выполняются режимы «2» и «3».

Атрибут №36

ПОСТОЯННАЯ ВРЕМЕНИ ПО ЗАДАНИЮ - параметр настройки регулятора. Определяет скорость перехода к новому заданию в секундах при значениях атрибута **ПЕРЕХОД К НОВОМУ ЗАДАНИЮ**, равных 1 или 13.

$$Y_{\text{вых.}} = Y_{\text{расч.}} + (-) \text{РАССОГЛАСОВАНИЕ} * K_{\phi},$$

где K_{ϕ} - коэффициент для форсированного перехода.

Атрибут №37

КОЭФФИЦИЕНТ ДЛЯ ФОРСИР. ПЕРЕХОДА - параметр настройки регулятора. Определяет степень дополнительного воздействия на ИМ при значениях атрибута **ПЕРЕХОД К НОВОМУ ЗАДАНИЮ**, равных 2 или 23. В момент изменения задания регулятору расчет осуществляется по следующей формуле:

Атрибут №38

СКОРОСТЬ ХОДА ИМ / ВРЕМЯ ПОЛНОГО ХОДА ИМ - параметр настройки регулятора. Для регулятора типа 0 - это максимальная величина, на которую может измениться выходной сигнал регулятора за один цикл его работы (по умолчанию – 100%). Для импульсных типов регулятора 1, 2, 4 и 5 - это время полного хода ИМ в секундах.

Атрибут №49

ЗНАЧЕНИЕ ПРИ РУЧНОМ ДИСТАНЦ. УПРАВЛЕНИИ. Для регулятора типа 0 -это начальное значение выходного сигнала регулятора, если он находится в режиме ручного дистанционного управления. Все дальнейшие изменения выходного сигнала записываются в это же поле, поэтому, даже при перезапуске контроллера, значение выходного сигнала будет соответствовать последнему значению на момент перезапуска. Для импульсных типов регуляторов 1, 2, 4, 5, 6, 7, 8, 9, 10, 11 данный атрибут используется для хранения результатов промежуточных расчётов и является расчетной длительностью импульса, который в данный момент необходимо выдать на ИМ.

Атрибут №67

ЗАПРЕТ ПЕРЕХОДА В ДУ ПО НЕДОСТОВЕРНОСТИ - логический атрибут,

- переход в режим «**РУЧНОЕ ДИСТАНЦИОННОЕ УПРАВЛЕНИЕ**» при недостоверности регулируемой переменной («0»),
- запрет перехода в режим «**РУЧНОЕ ДИСТАНЦИОННОЕ УПРАВЛЕНИЕ**» при недостоверности регулируемой переменной («1»).

Атрибут №68

ИНВЕРСИЯ ВЫХОДНОГО СИГНАЛА - логический атрибут,

- 0% (значение атрибута №48) соответствует ток начала диапазона выходного узла, а 100% - ток конца диапазона выходного узла, например, 0% - 4 мА, 100% - 20 мА для узла 4...20 мА.
- значение физического выходного сигнала инвертируется по отношению к значению выходного сигнала регулятора (атрибут №48), например, 0% - 20 мА, 100% - 4 мА для узла 4...20 мА.

Атрибут №77

ВИД ДЕЙСТВИЯ (1 – ПРЯМОЙ / 0 – ОБРАТНЫЙ) - вид действия регулятора.

Для прямого регулятора рассогласование считается по формуле:

Рассогласование = Текущее значение регулируемого параметра - Задание,

Для обратного регулятора рассогласование считается по формуле:

Рассогласование = Задание - Текущее значение регулируемого параметра.

- **ОБРАТНЫЙ** («0»),
- **ПРЯМОЙ** («1»).

Атрибут №78

ВИД ДЕЙСТВИЯ ИМ (НЗ–1 / НО–0) - вид действия исполнительного механизма (ИМ) для аналогового регулятора (используется только для визуализации).

- **НО** («0»),
- **НЗ** («1»).

Атрибут №83

СНЯТИЕ С СИГНАЛИЗАЦИИ ЗАДАНИЮ – логический признак отключения аварийной сигнализации по отклонению регулируемого параметра от задания.

- сигнализация по отклонению от задания включена («0»),
- сигнализация по отклонению от задания отключена («1»).

Атрибут №84

СНЯТИЕ С СИГНАЛИЗАЦИИ ПО ПОЛОЖЕНИЮ ИМ – логический признак отключения предупредительной сигнализации по выходному сигналу на исполнительный механизм (для регулятора типа 0).

- сигнализация по ходу ИМ включена («0»),
- сигнализация по ходу ИМ отключена («1»).

Атрибут №91

РЕЖИМ РЕГУЛЯТОРА <РУЧ. АППАРАТНЫЙ> – логический признак для режима управления исполнительным механизмом с помощью ручной байпасной панели или с БРУ. Если указана переменная в атрибутах «тип» и «номер признака ручного аппаратного управления», то данный режим будет зависеть от состояния данной переменной, если переменная не указана – данный атрибут можно изменять из программ пользователя на языке КРУГОЛ или с помощью кнопок соответствующего виртуального прибора управления через графический интерфейс системы.

- режим ручного аппаратного управления («1»).

Атрибут №92

РЕЖИМ РЕГУЛЯТОРА <РУЧ. ДИСТАНЦИОННЫЙ> – логический признак для включения / выключения режима ручного дистанционного управления исполнительным механизмом со станции оператора или из программ пользователя на языке КРУГОЛ.

- режим ручного дистанционного управления («1»).



ВНИМАНИЕ !!!

Режим «РУЧНОЙ АППАРАТНЫЙ» имеет высший приоритет.

Далее идет режим «РУЧНОЙ ДИСТАНЦИОННЫЙ».

Отсутствие этих двух режимов переводит регулятор в режим «АВТОМАТИЧЕСКОЕ УПРАВЛЕНИЕ».

Атрибут №96

РЕЖИМ ВВОДА ЗАДАНИЯ <ВНЕШНИЙ> (<КАСКАД>) – логический признак.

- режим регулятора – локальный ввод задания (0). В этом режиме возможно непосредственное изменение задания регулятору через программу пользователя на языке КРУГОЛ или с помощью ввода задания в соответствующее поле прибора управления.
- режим регулятора - «**КАСКАДНОЕ УПРАВЛЕНИЕ**» (1). В этом режиме задание регулятору поступает из переменной, указанной в атрибутах «тип» и «номер задания», при отсутствии в данных атрибутах ссылки на переменную возможно непосредственное изменение задания регулятору через программу пользователя на языке КРУГОЛ.



ВНИМАНИЕ !!!

В разделе не рассматриваются атрибуты, значения которых формируются на основе внутренних алгоритмов СРВК

5.9.6.2 Входная аналоговая переменная

Атрибут №1

№ КАНАЛА – по умолчанию значение атрибута для контроллера равно 0

Атрибут №2

КОД ОБРАБОТКИ определяет возможность управления алгоритмами обработок переменной в СРВК. При создании базы данных по каналу значение данного атрибута записывается в атрибут №2 – "Код обработки" переменной СРВК. При значении 0 – в СРВК выполняются стандартные обработки переменной в полном объеме. Подробное описание использования данного атрибута смотрите в приложении Б.1 руководства Пользователя для СРВК версии 7.0 и выше.

Атрибут №3

№ ПЛАТЫ – номер платы в УСО, к которой подключена данная переменная, 0 – если переменная «виртуальная» в контроллере (не имеет физического входа).

Атрибут №4

№ ВХОДА НА ПЛАТЕ – номер входа на плате УСО для физического подключения переменной, 0 – если переменная «виртуальная» в контроллере (не имеет физического входа).

Атрибут №5

ПОЗИЦИЯ – код параметра, в качестве которого обычно используются первые 8 символов технологической (или КИП и А) позиции данного параметра (например, 302PICAЕ), которая в дальнейшем может использоваться при отображении (24 символа).

Атрибут №6

ИМЯ1 – это первые 8 символов в названии параметра, если для БД не установлен признак поддержки 24-хсимвольной позиции, или вторые 8 символов технологической позиции параметра, если признак поддержки 24-хсимвольной позиции установлен.

Атрибут №7

ИМЯ2 – это вторые 8 символов в названии параметра, если для БД не установлен признак поддержки 24-хсимвольной позиции, или третьи 8 символов технологической позиции параметра, если признак поддержки 24-хсимвольной позиции установлен.

Атрибут №8

ЕДИНИЦА ИЗМЕРЕНИЯ – единица измерения параметра. Атрибут может принимать целые значения от 1 до 63, выбирается из выпадающего списка единиц измерения при его активизации. Чтобы узнать, какому значению атрибута соответствует какая единица измерения -В качестве списка используется словарь единиц измерения (смотрите раздел «[Единицы измерения](#)»).

Атрибут №9

ТИП ДАТЧИКА – этот атрибут необходимо указывать для УСО с программируемыми характеристиками преобразования сигналов первичных преобразователей (например: термопар, термометров сопротивления и т.п.). Тип датчика выбирается из выпадающего списка наименований типов датчиков. Каждому типу датчика соответствует определенное цифровое значение, которое записывается в данный атрибут (смотрите раздел «Словарь типов датчиков»).

Атрибут №10

ТИП ЛИНЕАРИЗАЦИИ ШКАЛЫ это тип зависимости, по которой производится линеаризация шкалы.

В системе используются следующие типы линеаризации шкалы:

- Линейная шкала (0),
- Квадратичная (1),

Линеаризация шкалы производится по формуле:

$$Y = НШК + (КШК - НШК) \times \sqrt{1 + \frac{КШК - X}{НШК - КШК}},$$

где: Y - значение переменной после линеаризации;
X - значение переменной до линеаризации;
НШК - начало шкалы;
КШК - конец шкалы.

Обратная шкала (2),

- минимальному значению входного сигнала соответствует «КОНЕЦ ШКАЛЫ»,
- максимальному - «НАЧАЛО ШКАЛЫ»,

Линеаризация по 2-х мерным таблицам нелинейности, создаваемым самим пользователем (номер таблицы нелинейности).

При описании каналов термопар и термометров сопротивления, подключаемых к контроллерам TREI, в графе «тип линеаризации шкалы» указывается линейная шкала.

Атрибут №11

НАЧАЛО ШКАЛЫ (НШК) - используется для масштабирования переменной или для определения нижней границы достоверности. Диапазон изменения атрибутов от минус 99999.9999 до 99999.9999 (но общее количество значащих цифр с учетом запятой не должно превышать восьми).

Атрибут №12

КОНЕЦ ШКАЛЫ (КШК) - используется для масштабирования переменной или для определения верхней границы достоверности. Диапазон изменения атрибутов от минус 99999.9999 до 99999.9999 (но общее количество значащих цифр с учетом запятой не должно превышать восьми).

Атрибут №13

НИЖНЯЯ ПРЕДУПРЕДИТЕЛЬНАЯ ГРАНИЦА – нижняя предупредительная граница (НПГ) сигнализации по параметру, цвет сигнализации - желтый.

Атрибут №14

ВЕРХНЯЯ ПРЕДУПРЕДИТЕЛЬНАЯ ГРАНИЦА – верхняя предупредительная граница (ВПГ) сигнализации по параметру, цвет сигнализации - желтый.

Атрибут №15

НИЖНЯЯ ПРЕДАВАРИЙНАЯ ГРАНИЦА – нижняя предаварийная граница (НАГ) сигнализации по параметру, цвет сигнализации - красный.

Атрибут №16

ВЕРХНЯЯ ПРЕДАВАРИЙНАЯ ГРАНИЦА – верхняя предаварийная граница (ВАГ) сигнализации по параметру, цвет сигнализации - красный.

Границы сигнализации задаются в физических единицах. Диапазон изменения атрибутов от минус 99999.9999 до 99999.9999 (но общее количество значащих цифр с учетом запятой не должно превышать восьми). При значении атрибута границы сигнализации равном 0, сигнализация по данной границе считается не назначенной. Порядок срабатывания сигнализации описан в «Инструкции по эксплуатации Системы КРУГ-2000».

Атрибут №17

ГИСТЕРЕЗИС СИГНАЛИЗАЦИИ. Диапазон изменения от 0 до 9,999% от шкалы. При сигнализации об отклонениях от заданных границ, необходимо избегать появления избыточной сигнализации при небольших колебаниях переменной вверх и вниз от границы. Поэтому в системе предусмотрен следующий алгоритм сигнализации:

Для верхней предупредительной (предаварийной) границы алгоритм реализуется следующим образом:

- параметр считается отклонившимся (выдается сигнализация), если переменная X больше, например, верхней предупредительной (предаварийной) границы, т.е.

$$X > \text{ВПГ}$$

- параметр считается вошедшим в норму, если выполняется условие

$$X < (\text{ВПГ} - \text{ГСС} * (\text{КШК} - \text{НШК})),$$

где ГСС - величина гистерезиса сигнализации
КШК – конец шкалы диапазона измерения переменной
НШК – начало шкалы диапазона измерения переменной.

Для нижней предупредительной (предаварийной) границы алгоритм реализуется следующим образом:

- параметр считается отклонившимся (выдается сигнализация), если выполняется условие:

$$X < \text{НПГ}$$

- параметр считается вошедшим в норму, если выполняется условие:

$$X > (\text{НПГ} + \text{ГСС} * (\text{КШК} - \text{НШК}))$$

где: ГСС - величина гистерезиса сигнализации
КШК – конец шкалы диапазона измерения переменной
НШК – начало шкалы диапазона измерения переменной.

Атрибут №18

ВРЕМЯ ОПРЕДЕЛЕНИЯ СКОРОСТИ ИЗМЕНЕНИЯ ПАРАМЕТРА – интервал времени, на основании значения которого выполняется расчет скорости роста или падения переменной.

Атрибут №19

ГРАНИЦА СИГНАЛИЗАЦИИ ПО СКОРОСТИ РОСТА – максимальное значение скорости переменной за время, заданное в атрибуте «Время границы сигнализации».

Атрибут №20

ГРАНИЦА СИГНАЛИЗАЦИИ ПО СКОРОСТИ ПАДЕНИЯ – максимальное значение скорости переменной за время, заданное в атрибуте «Время границы сигнализации».

Атрибут №21

ПОСТОЯННАЯ ФИЛЬТРА ОТ 0,000 ДО 1,000 - в системе предусмотрена цифровая фильтрация сигналов, при этом используется алгоритм экспоненциального фильтра первого порядка:

$$Y(i) = \text{ПФ} \times Y(i - 1) + (1 - \text{ПФ}) \times X(i) ,$$

где:

- i – номер цикла расчета;
- $Y(i)$ – сглаженное значение сигнала измерительной информации, в i -ом цикле расчета, получаемое в результате фильтрации;
- ПФ – постоянная фильтра (от 0 до 1);
- $Y(i-1)$ – значение сигнала, обновляемое в каждом цикле расчета;
- $X(i)$ – значение сигнала до фильтрации.

Атрибут №22

Максимально возможная (ДОСТОВЕРНАЯ) скорость – максимально возможная скорость изменения переменной за один цикл опроса, которая используется для определения достоверности параметра. В поле задается допустимая разница между текущим и предыдущим значениями параметра в процентах. Если реальная разница между текущим и предыдущим значениями превысит указанное значение, то останется предыдущее значение параметра, но сработает функция диагностики по данному параметру. Используется в случае необходимости защиты от отдельных выбросов сигнала. Если значение равно «0», то эта диагностика будет отключена.

Атрибут №23

ТИП ЗАМЕНЫ НЕДОСТОВЕРНОГО ЗНАЧЕНИЯ. Возможны ситуации, когда в процессе работы системы значение той или иной переменной могут стать недостоверными (например: обрыв датчика, неисправность датчика, неисправность УСО и т.д.) или эта переменная участвует в расчетах, в результате чего все расчеты могут стать неверными. Поэтому имеется возможность заменить недостоверное значение одним из следующих вариантов:

- «0 – по шкале прибора» - текущему значению переменной присваивается значение начала шкалы датчика, если текущее значение параметра меньше НШК, и конец шкалы, если текущее значение параметра больше КШК,
- «1 – последнее достоверное» - текущему значению переменной присваивается последнее достоверное значение,
- «2 – без диагностики» - текущему значению переменной присваивается текущее значение параметра, измеряемого в УСО, при этом отключается режим диагностики переменных по началу и концу шкалы,
- «3 – рабочее значение» - текущему значению переменной присваивается рабочее значение (смотри ниже),

Атрибут №24

РАБОЧЕЕ ЗНАЧЕНИЕ ПЕРЕМЕННОЙ – это величина, которая присваивается текущему значению переменной, если выставлен признак недостоверности по данной переменной и тип замены недостоверного значения на достоверное значение выбран «3».

Атрибут №25

ОТСЕЧКА НУЛЯ. Параметр настройки, используемый для отсечки колебаний сигнала датчика возле нулевого значения, задается в процентах от диапазона измерения. При использовании данного атрибута, значения границ сигнализации должны устанавливаться вне диапазона отсечки нуля.

В некоторых случаях (например, при расчете технико-экономических показателей) с целью повышения достоверности при первичной обработке переменной целесообразно использовать алгоритм коррекции нулевого значения сигнала от датчика. Сущность алгоритма заключается в следующем: переменной X присваивается нулевое значение, если значение переменной становится ниже какого-то заданного порогового значения ОТС, которое называется «отсечка нуля», т.е.:

$$X=0, \text{ если } X < \text{ОТС} \cdot (\text{КШК} - \text{НШК})$$

Атрибут №29

ТЕКУЩЕЕ ЗНАЧЕНИЕ ПОСЛЕ ПРЕОБРАЗОВАНИЯ (КОНТРОЛЛЕР) – атрибут вещественного формата, используется для возможности задания значения по умолчанию для не физических переменных контроллера.

Атрибут №30

СНЯТИЕ ПЕРЕМЕННОЙ С ОПРОСА – логический признак снятия переменной с опроса в УСО. Снятая с опроса переменная имеет цвет состояния «белый». При этом опрос переменной в УСО прекращается.

- переменная поставлена на опрос в УСО («0»),
- переменная снята с опроса в УСО («1»).

Атрибут №31

СНЯТИЕ ПЕРЕМЕННОЙ С СИГНАЛИЗАЦИИ – логический признак снятия переменной с сигнализации в УСО. Снятая с сигнализации переменная имеет цвет состояния «циановый». При этом прекращается обработка по границам сигнализации для данной переменной в УСО.

- переменная поставлена на сигнализацию в УСО («0»),
- переменная снята с сигнализации в УСО («1»).



ВНИМАНИЕ !!!

В разделе не рассматриваются атрибуты, значения которых формируются на основе внутренних алгоритмов СРВК.

5.9.6.3 Входная дискретная переменная

Атрибут №1

№ КАНАЛА – по умолчанию значение атрибута для контроллера равно 0

Атрибут №2

КОД ОБРАБОТКИ – определяет возможность управления алгоритмами обработок переменной в СРВК. При создании базы данных по каналу значение данного атрибута записывается в атрибут №2 – "Код обработки" переменной СРВК. При значении 0 – в СРВК выполняются стандартные обработки переменной в полном объеме. Подробное описание использования данного атрибута смотрите в приложении Б.1 руководства Пользователя для СРВК версии 7.0 и выше.

Атрибут №3

№ ПЛАТЫ – номер платы в УСО, к которой подключена данная переменная, 0 – если переменная «виртуальная» в контроллере (не имеет физического входа).

Атрибут №4

№ ВХОДА НА ПЛАТЕ - номер входа на плате УСО для физического подключения переменной, 0 – если переменная «виртуальная» в контроллере (не имеет физического входа).

Атрибут №5

ПОЗИЦИЯ – код параметра, в качестве которого обычно используются первые 8 символов технологической (или КИП и А) позиции данного параметра (например: 302PAL01).

Атрибут №6

ИМЯ1 – это первые 8 символов в названии параметра, если для БД не установлен признак поддержки 24-хсимвольной позиции, или вторые 8 символов технологической позиции параметра, если признак поддержки 24-хсимвольной позиции установлен.

Атрибут №7

ИМЯ2 – это вторые 8 символов в названии параметра, если для БД не установлен признак поддержки 24-хсимвольной позиции, или третьи 8 символов технологической позиции параметра, если признак поддержки 24-хсимвольной позиции установлен.

Атрибут №8

КОД ЦВЕТА СОСТОЯНИЯ <0> означает цвет, которым будет отображаться описатель логического состояния для данной переменной на видеокадрах, если текущее значение переменной равно "0".

Атрибут №9

КОД ЛОГИЧЕСКОГО СОСТОЯНИЯ <0> означает, какое название будет иметь описатель состояния переменной, если текущее значение переменной будет равно "0". Коды логических состояний описываются в разделе «Код логического состояния».

Атрибут №10

КОД ЦВЕТА СОСТОЯНИЯ <1> означает цвет, которым будет отображаться описатель логического состояния для данной переменной на видеокадрах, если текущее значение переменной равно "1".

Атрибут №11

КОД ЛОГИЧЕСКОГО СОСТОЯНИЯ <1> означает, какое название будет иметь описатель состояния переменной, если текущее значение переменной будет равно "1". Коды логических состояний описываются в разделе «Код логического состояния».

ПРИМЕЧАНИЕ:

Коды цветов для логических состояний переменной "0" и "1" могут принимать значения:

0 - черный;	8 - темно-серый;
1 - синий;	9 - ярко-синий;
2 - зеленый;	10 - ярко-зеленый;
3 - голубой;	11 - ярко-голубой;
4 - красный;	12 - ярко-красный;
5 - фиолетовый;	13 - ярко-фиолетовый;
6 - коричневый;	14 - желтый;
7 - светло-серый;	15 - белый.

Атрибут №13

ПРИЗНАК НЕОБХОДИМОСТИ ИНВЕРСИИ - логический признак,

- Рисунок 5.9.4 – Изменение количества переменных базы данных инверсия текущего значения переменной не осуществляется ("0"),
- текущее значение переменной в базе данных инвертируется по отношению к дискретному сигналу, подаваемому на вход модуля дискретного ввода УСО ("1").

Атрибут №14

РЕГИСТРАЦИЯ ПЕРЕХОДА ИЗ 0 В 1 РГ 0 > 1 – логический признак регистрации перехода переменной из "0" в "1":

- при переходе текущего значения переменной из "0" в "1" регистрации в «Протоколе событий» не будет ("0"),
- при переходе текущего значения переменной из "0" в "1" в «Протоколе событий» появится соответствующее сообщение зеленого цвета ("1").

Атрибут №15

РЕГИСТРАЦИЯ ПЕРЕХОДА ИЗ 1 В 0 РГ 1 > 0 – логический признак регистрации перехода из "1" в "0":

- при переходе текущего значения переменной из "1" в "0" регистрации в «Протоколе событий» не будет ("0"),
- при переходе текущего значения переменной из "1" в "0" в «Протоколе событий» появится соответствующее сообщение зеленого цвета ("1").

Атрибут №16

ЗВУКОВАЯ СИГНАЛИЗАЦИЯ ПЕРЕХОДА ИЗ 1 В 0 логический признак включения звуковой сигнализации при переходе из "1" в "0":

- при переходе текущего значения переменной из "1" в "0" звуковой сигнализации не будет ("0"),
- при переходе текущего значения переменной из "1" в "0" включается звуковая и световая сигнализация по данному параметру и в «Протоколе событий» появится соответствующее сообщение о нарушении с цветом, соответствующим типу звуковой сигнализации (см. атрибут №18) ("1").

ЗВУКОВАЯ СИГНАЛИЗАЦИЯ ПЕРЕХОДА ИЗ 0 В 1 $CG\ 1 > 0$ – логический признак включения звуковой сигнализации при переходе из "01" в "10":

- при переходе текущего значения переменной из "01" в "10" звуковой сигнализации не будет ("0"),
- при переходе текущего значения переменной из "01" в "10" включается звуковая и световая сигнализация по данному параметру и в «Протоколе событий» появится соответствующее сообщение о нарушении с цветом, соответствующим типу звуковой сигнализации (см. атрибут №18) ("1").

ПРИМЕЧАНИЕ:

Для исключения дублирования сообщений при переходах текущего значения переменной из "0" в "1" или наоборот не рекомендуется назначать одновременно регистрацию и сигнализацию по переходу переменной в одну и ту же сторону.

Атрибут №18

ТИП ЗВУКОВОЙ СИГНАЛИЗАЦИИ – может принимать следующие значения:

- 0** - Звуковая сигнализация - низкого тона (предупредительная), световая – мигание атрибутов переменной желтым цветом. После квитирования сигнализации по переменной - цвет атрибутов переменной желтый
- 1** - Звуковая сигнализация - высокого тона (аварийная), световая – мигание атрибутов переменной красным цветом. После квитирования сигнализации по переменной - цвет атрибутов переменной красный
- 2** - Звуковая сигнализация - низкого тона (предупредительная), световая – мигание атрибутов переменной желтым цветом. После квитирования сигнализации по переменной, переменная возвращается в норму - цвет атрибутов переменной зеленый
- 3** - Звуковая сигнализация - высокого тона (аварийная), световая – мигание атрибутов переменной красным цветом. После квитирования сигнализации по переменной, переменная возвращается в норму - цвет атрибутов переменной зеленый.

Атрибут №23

ДОПУСТИМОЕ ВРЕМЯ ПЕРЕХОДА ИЗ СОСТОЯНИЯ 0 В 1 - время, в течение которого подтверждается стабильность входного дискретного сигнала для перехода из '0' в '1'. Задается пользователем в циклах СРВК.

Атрибут №24

ДОПУСТИМОЕ ВРЕМЯ ПЕРЕХОДА ИЗ СОСТОЯНИЯ 1 В 0 - время, в течение которого подтверждается стабильность входного дискретного сигнала для перехода из '1' в '0'. Задается пользователем в циклах СРВК.

Атрибут №25

СНЯТИЕ ПЕРЕМЕННОЙ С ОПРОСА – логический признак. Снятая с опроса переменная имеет цвет состояния «белый». При этом прекращается опрос данной переменной в УСО.

- переменная поставлена на опрос («0»),
- переменная снята с опроса («1»).

Атрибут №26

СНЯТИЕ ПЕРЕМЕННОЙ С СИГНАЛИЗАЦИИ – логический признак. Снятая с сигнализации переменная имеет цвет состояния «циановый». При этом прекращается обработка по сигнализации переходов из "0" в "1", а также из "1" в "0", для данной переменной в УСО.

- переменная поставлена на сигнализацию ("0"),
- переменная снята с сигнализации ("1").

Атрибут №27

ТЕКУЩЕЕ ЗНАЧЕНИЕ ПЕРЕМЕННОЙ – логический признак, используется для возможности задания через Генератор базы данных значения по умолчанию для не физических переменных контроллера

- текущее значение переменной равно ("0"),
- текущее значение переменной равно ("1").

**ВНИМАНИЕ !!!**

В разделе не рассматриваются атрибуты, значения которых формируются на основе внутренних алгоритмов СРВК .

5.9.6.4 Дискретная выходная переменная

Атрибут № 1

№ ПЛАТЫ – по умолчанию значение атрибута для контроллера равно 0

Атрибут № 2

КОД ОБРАБОТКИ – определяет возможность управления алгоритмами обработок переменной в СРВК. При создании базы данных по каналу значение данного атрибута записывается в атрибут №2 – "Код обработки" переменной СРВК. При значении 0 – в СРВК выполняются стандартные обработки переменной в полном объеме. Подробное описание использования данного атрибута смотрите в приложении Б.1 руководства Пользователя для СРВК версии 7.0 и выше.

Атрибут № 4

№ ПЛАТЫ – номер платы в УСО, к которой подключена данная переменная, 0 – если переменная «виртуальная» в контроллере (не имеет физического выхода).

Атрибут № 5

№ ВЫХОДА НА ПЛАТЕ – номер выхода на плате УСО для физического подключения переменной, 0 – если переменная «виртуальная» в контроллере (не имеет физического входа).

Атрибут № 6

ПОЗИЦИЯ – код параметра, в качестве которого обычно используются первые 8 символов технологической (или КИП и А) позиции данного параметра (например: 303МС-10).

Атрибут № 7

ИМЯ1 – это первые 8 символов в названии параметра , если для БД не установлен признак поддержки 24-хсимвольной позиции, или вторые 8 символов технологической позиции параметра, если признак поддержки 24-хсимвольной позиции установлен.

Атрибут № 8

ИМЯ2 – это вторые 8 символов в названии параметра, если для БД не установлен признак поддержки 24-хсимвольной позиции, или третьи 8 символов технологической позиции параметра, если признак поддержки 24-хсимвольной позиции установлен.

Атрибут № 9

КОД ЦВЕТА СОСТОЯНИЯ <0> означает цвет, которым будет отображаться описатель логического состояния для данной переменной на видеокадрах, если текущее значение переменной равно "0".

Атрибут № 10

КОД ЛОГИЧЕСКОГО СОСТОЯНИЯ <0> означает, какое название будет иметь описатель состояния переменной, если текущее значение переменной будет равно "0". Коды логических состояний описываются в разделе «Код логического состояния».

Атрибут № 11

КОД ЦВЕТА СОСТОЯНИЯ <1> означает цвет, которым будет отображаться описатель логического состояния для данной переменной на видеокадрах, если текущее значение переменной равно "1".

Атрибут № 12

КОД ЛОГИЧЕСКОГО СОСТОЯНИЯ <1> означает, какое название будет иметь описатель состояния переменной, если текущее значение переменной будет равно "1". Коды логических состояний описываются в разделе «Код логического состояния».

ПРИМЕЧАНИЕ:

Коды цветов для логических состояний переменной "0" и "1" могут принимать значения:

0 - черный;	8 - темно-серый;
1 - синий;	9 - ярко-синий;
2 - зеленый;	10 - ярко-зеленый;
3 - голубой;	11 - ярко-голубой;
4 - красный;	12 - ярко-красный;
5 - фиолетовый;	13 - ярко-фиолетовый;
6 - коричневый;	14 - желтый;
7 - светло-серый;	15 - белый.

Атрибут № 14

ПРИЗНАК НЕОБХОДИМОСТИ ИНВЕРСИИ - логический признак,

- инверсия выходного сигнала на узел дискретного вывода в контроллере по отношению к текущему значению переменной в базе данных не осуществляется («0»),
- выходной сигнал на узел дискретного вывода в контроллере TREI инвертируется по отношению к текущему значению переменной в базе данных («1»).

Атрибут №15

РЕГИСТРАЦИЯ ПЕРЕХОДА ИЗ 0 В 1 РГ 0 > 1 – логический признак регистрации перехода переменной из "0" в "1":

- при переходе текущего значения переменной из "0" в "1" регистрации в «Протоколе событий» не будет («0»),
- при переходе текущего значения переменной из "0" в "1" в «Протоколе событий» появится соответствующее сообщение зеленого цвета («1»).

Атрибут №16

РЕГИСТРАЦИЯ ПЕРЕХОДА ИЗ 1 В 0 РГ 1 > 0 – логический признак регистрации перехода из "1" в "0":

- при переходе текущего значения переменной из "1" в "0" регистрации в «Протоколе событий» не будет («0»),
- при переходе текущего значения переменной из "1" в "0" в «Протоколе событий» появится соответствующее сообщение зеленого цвета («1»).

Атрибут № 17

РЕЖИМ ВВОДА СИМВОЛА – признак выдачи выходного сигнала в виде импульса заданной длительности (от 0 до 255),

- 0 - изменение текущего значения переменной осуществляется непосредственно вводом необходимого значения переменной из программы пользователя, с помощью прибора управления или через таблицу настройки переменной,
- от 1 до 255 - при передаче в контроллер TREI текущего значения в виде логической 1 через заданное время (от 1 до 255 сек) текущее значение переменной сбрасывается в 0.

Атрибут № 33

СНЯТИЕ С ОПРОСА – логический признак. Снятая с опроса переменная имеет цвет состояния «белый».

- переменная поставлена на опрос («0»),
- переменная снята с опроса («1»).



ВНИМАНИЕ !!!

В разделе не рассматриваются атрибуты, значения которых формируются на основе внутренних алгоритмов СРВК .

5.9.6.5 Ручной ввод

Атрибут №1

ПОЗИЦИЯ – код параметра, в качестве которого обычно используются первые 8 символов технологической (или КИП и А) позиции данного параметра.

Атрибут №2

ИМЯ1 – это первые 8 символов в названии параметра, если для БД не установлен признак поддержки 24-хсимвольной позиции, или вторые 8 символов технологической позиции параметра, если признак поддержки 24-хсимвольной позиции установлен.

Атрибут №3

ИМЯ2 – это вторые 8 символов в названии параметра, если для БД не установлен признак поддержки 24-хсимвольной позиции, или третьи 8 символов технологической позиции параметра, если признак поддержки 24-хсимвольной позиции установлен

Атрибут №4

ТИП ПЕРЕМЕННОЙ - определяет тип текущего значения переменной. Может быть вещественным, строковым, целым или логическим.

Атрибут №5

ЕДИНИЦА ИЗМЕРЕНИЯ – единица измерения параметра. Атрибут может принимать целые значения от 1 до 63, выбирается из выпадающего списка единиц измерения при его активизации. Чтобы узнать, какому значению атрибута соответствует какая единица измерения -В качестве списка используется словарь единиц измерения (смотрите раздел «[Единицы измерения](#)»).

Атрибут №6

Максимум.

Попытка ввода в действующей системе числа большего, чем заданное в этом атрибуте значение, игнорируется и сохраняется предыдущее значение.

Атрибут №7

Минимум

Попытка ввода в действующей системе числа меньшего, чем заданное в этом атрибуте значение, игнорируется и сохраняется предыдущее значение.

Атрибут №8

ТЕКУЩЕЕ ЗНАЧЕНИЕ это вещественное значение, которое будет присвоено данному атрибуту переменной при первоначальном запуске системы.

Атрибут №9

ТЕКУЩЕЕ ЗНАЧЕНИЕ 2 - это вещественное значение, которое будет присвоено данному атрибуту переменной при первоначальном запуске системы. Передается в УСО только в случае изменения атрибута при ручном вводе или по команде языка программирования КРУГОЛ (ПосДВ(N), где N – номер переменной в БД). Передается из УСО только в случае изменения атрибута при ручном вводе, по команде языка программирования КРУГОЛ (ПосДВ(N), где N – номер переменной в УСО), а также при обновлении базы данных после восстановления связи с УСО.

Атрибут №12

ТЕКУЩЕЕ ЗНАЧЕНИЕ (СТР) - это значение, которое будет присвоено переменной при первоначальном запуске системы (16 символов).

Атрибут №13

ЦВЕТ ОТОБРАЖЕНИЯ СИМВОЛОВ - код цвета, которым будет отображаться строка на видеокадрах.

ПРИМЕЧАНИЕ:

Код цвета отображения строки может принимать значения:

0 - черный;	8 - темно-серый;
1 - синий;	9 - ярко-синий;
2 - зеленый;	10 - ярко-зеленый;
3 - голубой;	11 - ярко-голубой;
4 - красный;	12 - ярко-красный;
5 - фиолетовый;	13 - ярко-фиолетовый;
6 - коричневый;	14 - желтый;
7 - светло-серый;	15 - белый.

Атрибут №14

ТЕКУЩЕЕ ЗНАЧЕНИЕ (ЛОГ) - это значение, которое будет присвоено данному атрибуту переменной при первоначальном запуске системы.

Атрибут №15

КОД ЛОГИЧЕСКОГО СОСТОЯНИЯ <0> означает, какое название будет иметь описатель состояния переменной, если текущее значение переменной будет равно "0". Коды логических состояний описываются в разделе «Код логического состояния».

Атрибут №16

КОД ЦВЕТА СОСТОЯНИЯ <0> означает цвет, которым будет отображаться описатель логического состояния для данной переменной на видеокадрах, если текущее значение переменной равно "0".

Атрибут №17

КОД ЛОГИЧЕСКОГО СОСТОЯНИЯ <1> означает, какое название будет иметь описатель состояния переменной, если текущее значение переменной будет равно "1". Коды логических состояний описываются в разделе «Код логического состояния».

Атрибут №18

КОД ЦВЕТА СОСТОЯНИЯ <1> означает цвет, которым будет отображаться описатель логического состояния для данной переменной на видеокадрах, если текущее значение переменной равно "1".

ПРИМЕЧАНИЕ:

Коды цветов для логических состояний переменной "0" и "1" могут принимать значения:

0 – черный;	8 – темно-серый;
1 – синий;	9 – ярко-синий;
2 – зеленый;	10 – ярко-зеленый;
3 – голубой;	11 – ярко-голубой;
4 – красный;	12 – ярко-красный;
5 – фиолетовый;	13 – ярко-фиолетовый;
6 – коричневый;	14 – желтый;
7 – светло-серый;	15 – белый.

**ВНИМАНИЕ !!!**

В разделе не рассматриваются атрибуты, значения которых формируются на основе внутренних алгоритмов СРВК .

5.9.6.6 Код логического состояния

Коды и расшифровка логических состояний приведены в таблице ниже

Код	Значение
1	-----
2	"ОТКР"
3	"ЗАКР"
4	ОТКРЫТИЕ
5	ЗАКРЫТИЕ
6	ОТКРЫТА
7	ЗАКРЫТА
8	СРЕДНЕЕ
9	ОТКРЫТ
10	ЗАКРЫТ
11	СОБРАНА
12	РАЗОБРАН
<u>Код</u>	<u>Значение</u>
13	НЕИСПРАВ
14	РАЗРЕШЕН
15	"ВКЛ"
16	"ОТКЛ"
17	ВКЛЮЧЕН
18	ОТКЛЮЧЕН
19	ДЕБЛОКИР
20	Р пониж.
21	ЗАПРЕТ
22	НОРМА
23	Р низкое

Код	Значение
24	dP низк.
25	Вибрация
26	P повыш.
27	F низкий
28	ВЫСОКАЯ
29	L высок.
30	P высок.
31	L низкий
32	ЗАГАЗОВ
33	P пониж.
34	P норма
35	L норма
36	dP норма
37	F норма
38	T норма
39	Ос.сдвиг
40	Кл.на бл
41	Кл.дебл.
42	Бл.по #P
43	Бл.включ
44	Бл.выкл.
45	ВКЛЮЧЕНА
46	ЖДУ
47	СРАБОТАЛ
48	#P тг ог
49	#P тг пг
50	T пониж.
51	T высок.
52	СЪЕМ
53	АВАРИЯ
54	СИГНАЛ
55	РУЧНОЙ
56	АВТОМАТ
57	~220В
58	Эл.защит
59	~U норма
60	~U низк.
61	НОРМА
62	Помпаж
63	ОКОНЧЕНА
64	ПОВЫШЕНО
65	ПОНИЖЕНО
66	НИЗКИЙ
67	ВЫСОКИЙ
68	АВТ.ПУСК
69	U сх.упр
70	СТАНЦИЯ1
71	СТАНЦИЯ2
72	СТАНЦИЯ3
73	СТАНЦИЯ4

Код	Значение
74	ОБЩЕЕ
75	L повыш.
76	L пониж.
77	ПОЖАР!!!
78	"ПУСК"
79	СХ.ОТКЛ.
80	ЗАКЛИНИЛ
81	УпитН201
82	Уупр.хол
83	НОРМА
84	АВАРИЯ
85	Уупр.нас
86	АВРвключ
87	6кVЦК201
88	Авар.ТР1
89	Авар.ТР2
90	ВКЛЮЧЕНО
91	Включ.АВ
92	dP пониж
93	ШКАЛА 1
94	ШКАЛА 2
95	ВКЛЮЧЕНЫ
96	СБРОШЕНА
97	ОТКЛ.АВР
98	dP повыш
99	ОБНУЛЕН
100	ОСНОВНОЙ
101	РЕЗЕРВ
102	РЕГЕНЕР
103	ВЫЖ.УВГ
104	ВЫЖ.КОКС
105	ПРОКАЛКА
106	БАТАРЕЯ
107	НИЗКАЯ
108	L > 60%
109	L > 80%
110	L > 90%
111	"СТОП"

5.9.6.7 Единицы измерения

Атрибут «Единица измерения» может принимать следующие значения:

Значение	Описание
1	мм/сек
2	°С
3	об./мин
4	мг/м3

Значение	Описание
5	мм.вд.ст
6	мм.рт.ст
7	Па
8	кПа
9	Мпа
10	Гпа
11	кг/см2
12	м3/час
13	м3/мин
14	м3/сек
15	Нм3/час
16	Нм3/мин
17	Нм3/сек
18	кг
19	м.куб.
20	Нм.куб.
21	кг/час
22	кг/мин
23	кг/сек
24	т/час
25	т/мин
26	т/сек
27	м
28	см
29	мм
30	кг/м3
31	т/м3
32	г/см3
33	г/м3
34	г/л
35	мг/л
36	мг %
37	г
38	кг
39	т
40	тыс.т
41	рН
42	мА
43	А
44	кА
45	В
46	кВ
47	МВт
48	Вт
49	кВт
50	мДж
51	Дж
52	кДж
53	МДж
54	ГДж

Значение	Описание
55	кал
56	ккал
57	сек
58	мин
59	час
60	смена
61	сутки
62	%
63	кг/м2

5.9.6.8 Словарь типов датчиков

Значение	Описание
0	0-5mA, 4-20mA, 0-10V, 0-1000m и т.д.
2	Платинородий/Платина (S) (с 1999 г)
3	Платинородий/Платина (B) (с 1999 г)
4	Железо/Константан (J) (с 1999 г)
5	Медь/Константан (T) (с 1999 г)
6	Хромель/Константан (E) (с 1999 г)
7	Хромель/Алюмель (K) (с 1999 г)
8	НикельХром/Никель (N) (с 1999 г)
9	ВольфрамРений (A-1) (с 1999 г)
10	ВольфрамРений (A-2) (с 1999 г)
11	ВольфрамРений (A-3) (с 1999 г)
12	Хромель/Копель (L) (с 1999 г)
13	100П (w=1.3910) (с 1999 г)
14	50П (w=1.3910) (с 1999 г)
15	100М (w=1.4280) (с 1999 г)
16	50М (w=1.4280) (с 1999 г)
17	гр.21
18	гр.23
19	100П (w=1.3850) (с 1999 г)
20	Платинородий/Платина (R) (с 1999 г)
21	Платинородий/Платина (S) (до 1999 г)
22	Платинородий/Платина (B) (до 1999 г)
23	Железо/Константан (J) (до 1999 г)
24	Медь/Константан (T) (до 1999 г)
25	Хромель/Константан (E) (до 1999 г)
26	Хромель/Алюмель (K) (до 1999 г)
27	НикельХром/Никель (N) (до 1999 г)
28	ВольфрамРений (A-1) (до 1999 г)
29	ВольфрамРений (A-2) (до 1999 г)
30	ВольфрамРений (A-3) (до 1999 г)
31	Хромель/Копель (L) (до 1999 г)
32	100П (w=1.3910) (до 1999 г)
33	50П (w=1.3910) (до 1999 г)
34	100М (w=1.4280) (до 1999 г)
35	50М (w=1.4280) (до 1999 г)
38	100П (w=1.3850) (до 1999 г)

Значение	Описание
39	50П (w=1.3850) (с 1999 г)
40	100М (w=1.4260) (с 1999 г)
41	50М (w=1.4260) (с 1999 г)
50	Счетчик импульсов
51	Измерение периода для ICNT
52	Измерение об/мин для ICNT
53	Измерение частоты для ICNT
54	Счетчик импульсов с меткой перепол
101	(4-20мА) Платинородий/Платина (S)
102	(4-20мА) Платинородий/Платина (B)
103	(4-20мА) Железо/Константан (J)
104	(4-20мА) Медь/Константан (T)
105	(4-20мА) Хромель/Константан (E)
106	(4-20мА) Хромель/Алюмель (K)
107	(4-20мА) НикельХром/Никель (N)
108	(4-20мА) ВольфрамРений/ Вольфрам
109	(4-20мА) ВольфрамРений/ Вольфрам
110	(4-20мА) ВольфрамРений/ Вольфрам
111	(4-20мА) Хромель/Копель (L)
112	(4-20мА) 100П (w=1.3910) (до 1999 г)
113	(4-20мА) 50П (w=1.3910) (до 1999 г)
114	(4-20мА) 100М (w=1.4280) (до 1999 г)
115	(4-20мА) 50М (w=1.4280) (до 1999 г)
116	(4-20мА) гр.21
117	(4-20мА) гр23
118	(4-20мА) 100П (w=1.3850) (до 1999 г)
212	100П/5мА (w=1.3910) (с 1999 г)
213	50П/5мА (w=1.3910) (с 1999 г)
218	100П/5мА (w=1.3850) (с 1999 г)
232	100П/5мА (w=1.3910) (до 1999 г)
233	50П/5мА (w=1.3910) (до 1999 г)
238	100П/5мА (w=1.3850) (до 1999 г)

5.10 Конфигурирование трендов

5.10.1 Модуль XML-описания тренда

XML-описание трендов используется «ОПС-сервером СРВК» (версии 1.0 или более поздней версией) для определения текущей конфигурации HDA, а также «Модулем ведения трендов на контроллере» (СРВК версии 7.1x Linux).

Для формирования XML-описания трендов следует вызвать «Модуль XML-описания тренда». Вызов «Модуля XML-описания тренда» осуществляется командой «Конфигурирование трендов...» меню «Контроллер» или с помощью одноименной кнопки Панели инструментов, а также с помощью «горячей клавиши», которая может быть назначена Пользователем с помощью команды «Клавиатура» меню «Вид»

Главное окно Модуля XML-описания тренда показано на рисунке 10.1.

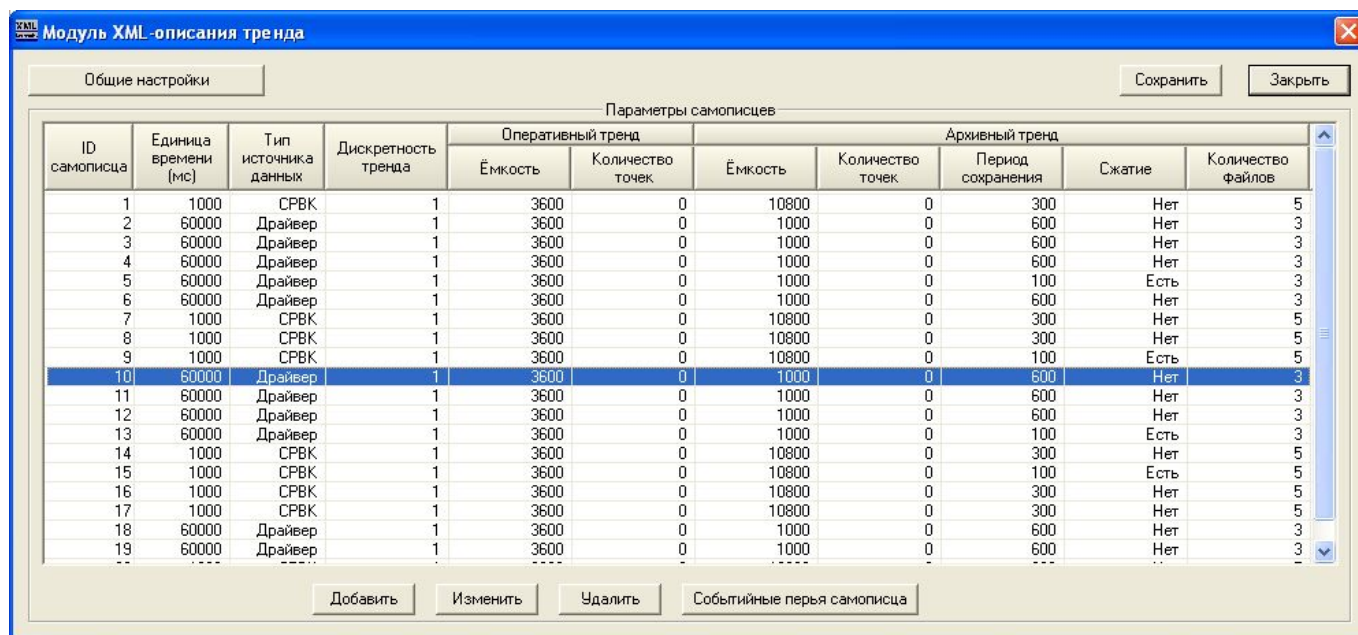


Рисунок 10.1– Главное окно Модуля XML-описания тренда

Главное окно Модуля XML-описания тренда содержит следующие элементы управления:

- Кнопка **«Общие настройки»** – для вызова диалогового окна для настройки общих параметров (смотрите раздел «Общие настройки»)
- Кнопка **«Сохранить»** – для сохранения XML-описания тренда в соответствующий файл ***trendcfg.xml*** (путь к файлу определяется из настроек проекта ИСР «Путь к БД канала»).
- Кнопка **«Заккрыть»** – для закрытия Модуля без сохранения изменений
- Табличный список **«Параметры самописцев»** – для отображения текущих настроек самописцев
- Кнопка **«Добавить»** – при нажатии производит добавление самописца с настройками по умолчанию
- Кнопка **«Изменить»** – для вызова диалогового окна изменения параметров текущих выделенных самописцев (смотрите «Редактирование самописцев» в разделе «Настройка самописцев»)
- Кнопка **«Удалить»** – для удаления текущих выделенных самописцев
- Кнопка **«Событийные перья самописца»** – для вызова диалогового окна настройки перьев самописцев (смотрите «Настройка событийных перьев самописца» в разделе «Настройка самописцев»).

Для операций над данными табличного списка **«Параметры самописцев»** можно воспользоваться контекстным меню, которое содержит пункты **«Изменить»**, **«Удалить»** и **«Событийные перья самописца»** (рисунок 10.2). Действия при активации пунктов контекстного меню аналогичны соответствующим кнопкам.

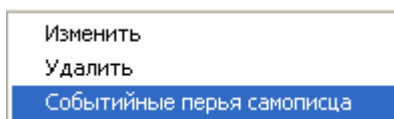
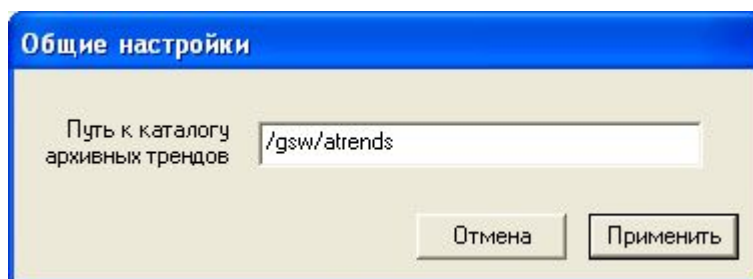


Рисунок 10.2 – Контекстное меню для Табличного списка самописцев

5.10.2 Общие настройки

При нажатии кнопки «**Общие настройки**» открывается окно «**Общие настройки**» (рисунок 10.3).



10.3 - Диалоговое окно «Общие настройки»

Диалоговое окно «Общие настройки» содержит следующие элементы управления:

- Поле «**Путь к каталогу архивных трендов**» – содержит путь к каталогу, в котором будут храниться архивные тренды. Если поле отсутствует, либо пустое используется путь по умолчанию. Данная настройка необходима для модуля ведения трендов на контроллере. Допустимые значения: 200 символов. Значение по умолчанию: **/gsw/atrends**.
- Кнопка «**Применить**» – для применения изменений общих настроек (без сохранения).
- Кнопка «**Отмена**» – для закрытия данного диалогового окна без применения настроек.

5.10.3 Настройка самописцев

Для настройки самописцев предусмотрены следующие возможные действия:

- Добавление
- Удаление
- Редактирование
- Настройка событийных перьев самописца.

Добавление самописцев

Добавление самописца осуществляется с помощью кнопки главного окна «**Добавить**». При однократном нажатии кнопки «**Добавить**» производится добавление одного самописца с настройками по умолчанию.

Примечание:

По ТМ каналу не передаются запросы самописцев с ID больше 255.

Удаление самописцев

Удаление выделенных самописцев осуществляется с помощью кнопки «**Удалить**» или при активации соответствующего пункта контекстного меню. Для выделения нескольких самописцев в списке необходимо, удерживая кнопку «**Ctrl**» или «**Shift**», выделить строки описания самописцев с помощью мыши. Кнопка «**Удалить**» доступна только при выделении одного или нескольких самописцев.

Редактирование самописцев

Редактирование самописцев осуществляется в окне «**Настройка самописца**» (рисунок 10.4), которое вызывается при нажатии на кнопку «**Изменить**» или при активации соответствующего пункта контекстного меню.

Рисунок 10.4 – Диалоговое окно «Настройка самописца»

Кнопка «**Изменить**» доступна только при выделении одного или нескольких самописцев.

Диалоговое окно «**Настройка самописца**» содержит следующие элементы управления:

- «**ID**» – информативное поле, в случае выделения одного самописца должно содержать его идентификатор. При редактировании нескольких самописцев поле ID будет содержать список ID редактируемых самописцев через запятую с использованием диапазонов (например, при редактировании самописцев с ID 1,2,3,4,5,15,20 поле ID будет содержать "1-5,15,20")
- Поле ввода «**Единица времени (мс)**» – используется для ввода единицы времени, являющейся единицей измерения всех остальных "временных" параметров самописца. Измеряется в мс (Например, 60000 мс = 1 минута).
Допустимые значения: целое положительное число от 1 до 2592000000 (30 дней).
Значение по умолчанию: 1000 (1 секунда)
- Выпадающий список «**Тип источника данных**» – определяет источник заполнения тренда: «CPVK» или «Драйвер». Значение по умолчанию – «CPVK».
- Поле ввода «**Дискретность тренда**» – используется для ввода времени между занесением в тренд двух последовательных значений переменной. С этим периодом ведется анализ изменений в БД и генерация новых точек. Измеряется в единицах времени.
Допустимые значения: целое положительное число от 1 до 4294967295 (максимальное положительное число, 4 байта)
Значение по умолчанию – 1
- Группа элементов «**Оперативный тренд**»:
 - Поле ввода «**Ёмкость**» – используется для ввода глубины тренда, хранимого в памяти. Измеряется в единицах времени.
Допустимые значения: целое положительное число от 10 до 4294967295 (максимальное положительное число, 4 байта).
Значение по умолчанию: 3600 (1 час при единице времени 1000 мс)
 - Поле ввода «**Количество точек**» – используется для ввода максимального количества точек, допустимого в оперативном тренде. Если равно 0 или поле отсутствует – ограничения нет.
Допустимые значения: 0 или целое положительное число от 3 до 100000.
Значение по умолчанию: 0 (нет ограничения)

- Группа элементов «**Архивный тренд**»:
 - Поле ввода «**Ёмкость**» – используется для ввода глубины тренда, хранимого в памяти. Измеряется в единицах времени. Если 0, то архивный тренд не ведётся.
Допустимые значения: 0 или целое положительное число от 10 до 4294967295 (максимальное положительное число, 4 байта)
 - Поле ввода «**Количество точек**» – используется для ввода максимального количества точек, допустимого в архивном тренде. Если равно 0 или поле отсутствует – ограничения нет.
Допустимые значения: 0 или целое положительное число от 3 до 100000.
Значение по умолчанию: 0 (нет ограничения)
 - Поле ввода «**Период сохранения**» – используется для ввода периода сохранения. Он же – максимальное время, для которого потеряются данные после рестарта. Если 0 – сохранение архивного тренда отключено. Измеряется в единицах времени.
Допустимые значения: 0 или целое положительное число от 10 до 4294967295 (максимальное положительное число, 4 байта).
Значение по умолчанию: 300 (5 минут при единице времени 1000 мс)
 - Признак «**Сжатие**» – используется для задания сжатия архивного тренда. Если галочка установлена, то полностью завершённые файлы архивного тренда будут сжиматься. Если галочка не установлена – сжатие отсутствует.
Значение по умолчанию: «Нет сжатия».
 - Поле ввода «**Количество файлов**» – используется для ввода количества файлов архивного тренда.
Значение по умолчанию: 5.
Для экономии места на файловой системе рекомендуется использовать значение – 2.

В случае настройки нескольких самописцев сразу, элементы управления будут содержать общие для них настройки. Каждый элемент управления при наличии разных настроек у редактируемых самописцев либо не содержит значения (пустое поле ввода), либо имеет неопределённое значение (например, в выпадающем списке не будет выбранного значения). При изменении и сохранении настроек изменения будут применяться для всех редактируемых самописцев.

Настройка событийных перьев самописца

Редактирование событийных перьев самописца осуществляется в окне «**Событийные перья**» (рисунок 10.5), которое вызывается при нажатии на кнопку «**Событийные перья самописца**» или при активации соответствующего пункта контекстного меню. Кнопка «**Событийные перья самописца**» доступна только при наличии выделения одного самописца.

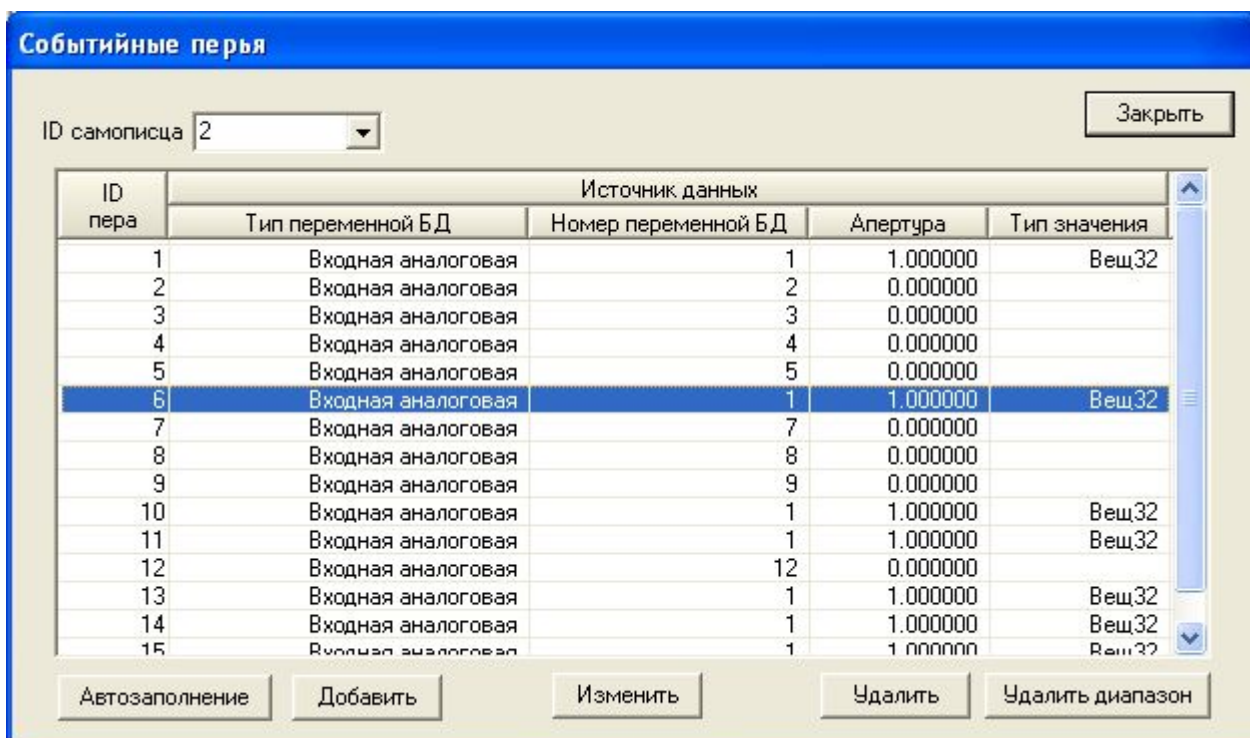


Рисунок 10.5 – Диалоговое окно «Событийные перья»

Диалоговое окно «Событийные перья» содержит следующие элементы управления:

- Выпадающий список «ID самописца» – для выбора самописца, перья которого необходимо настроить
- Табличный список перьев – для отображения текущих настроек перьев
- Кнопка «Заккрыть» – для закрытия диалогового с применением изменений
- Кнопка «Автозаполнение» – при нажатии производит вызов диалогового окна «Перья – Автозаполнение»
- Кнопка «Добавить» – при нажатии производит добавление пера с настройками по умолчанию
- Кнопка «Изменить» – для вызова диалогового окна настройки параметров текущих выделенных перьев
- Кнопка «Удалить» – для удаления текущих выделенных перьев
- Кнопка «Удалить диапазон» – при нажатии производит вызов диалогового окна «Перья - Удаление».

Для операций над данными табличного списка перьев можно воспользоваться контекстным меню, которое содержит пункты «Изменить», «Удалить» (рисунок 10.6). Действия при активации пунктов контекстного меню аналогичны соответствующим кнопкам.

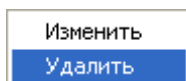


Рисунок 10.6 – Контекстное меню для табличного списка перьев

Для настройки событийных перьев самописца предусмотрены следующие действия:

- Добавление.
- Автозаполнение.
- Удаление, удаление диапазона.
- Редактирование.

Добавление событийных перьев

Добавление событийных перьев осуществляется к выбранному самописцу с помощью кнопки «Добавить». При однократном нажатии кнопки «Добавить» производится добавление одного пера с настройками по умолчанию.

Автозаполнение событийных перьев

Автозаполнение событийных перьев предназначено для настройки автоматического создания перьев с аналогичными параметрами. Настройка осуществляется для выбранного самописца в окне «Перья - Автозаполнение» (рисунок 10.7), которое открывается при нажатии на кнопку «Автозаполнение».

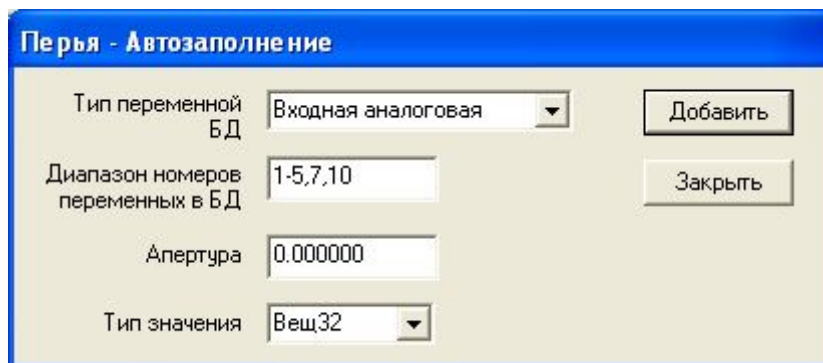


Рисунок 10.7 – Диалоговое окно «Перья – Автозаполнение»

Окно «Перья – Автозаполнение» содержит следующие элементы управления:

- Выпадающий список «**Тип переменной БД**» – для выбора типа переменной БД. Доступны следующие значения: «Входная аналоговая», «Аналоговая выходная», «РВ составная», «Входная дискретная», «Дискретная выходная». Значение по умолчанию – «Входная аналоговая»
- Поле ввода «**Диапазон номеров переменных в БД**» – текстовое поле для ввода диапазона записей для выбранного типа переменных. Указание диапазона записей выполняется с помощью символа «-», отдельные номера записей перечисляются через запятую (например: «1-3,4,7,22-20»).
Допустимые значения номера переменной: целое положительное число от 1 до 32767.
- Поле ввода «**Апертура**» – для задания апертуры для определения изменения значений, которые необходимо сохранять (задается в абсолютной величине).
Допустимые значения: число с плавающей точкой 0.0 или от 0.0001 до максимальное значение типа **float**.
Значение по умолчанию – 0.0.
- Выпадающий список «**Тип значения**» – для выбора типа значения, сохраняемого в тренде, соответствующем данному перу.
Данное поле доступно только при типе источника данных самописца «**Драйвер**» (свойство самописца "Тип источника данных").
Список допустимых значений:
 - пустое поле – используется значение, соответствующее типу переменной БД
 - Вещ32
 - Лог
 - Цел8; Цел16; Цел32; Цел8 (беззнак); Цел16 (беззнак); Цел32 (беззнак).

При нажатии кнопки «Добавить» производится добавление перьев с указанными настройками.

Удаление событийных перьев

Удаление выделенных событийных перьев осуществляется с помощью кнопки «**Удалить**» или при активации соответствующего пункта контекстного меню. Для выделения нескольких перьев следует удерживая нажатой кнопку «**Ctrl**» или «**Shift**», выделить мышью в списке необходимые строки описания перьев. Кнопка «Удалить» доступна только при выделении одного или нескольких перьев.

Для удаления списка перьев можно воспользоваться диалоговым окном «**Перья – Удаление**» (рисунок 10.8), которое вызывается при нажатии на кнопку «**Удалить диапазон**».

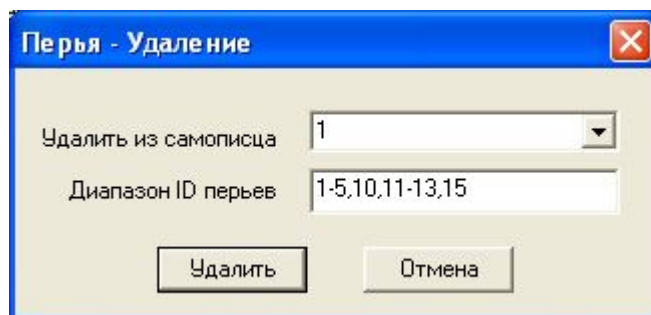


Рисунок 10.8 – Диалоговое окно «Перья – Удаление»

Элементы управления диалогового окна «Перья – Удаление»:

- Поле ввода «**Удалить из самописца**» – для задания ID самописца.
- Поле ввода «**Диапазон ID перьев**» – для указания диапазона перьев по ID для удаления. Указание диапазона записей выполняется с помощью символа «-», отдельные номера записей перечисляются через запятую (например: «1-3,4,7,22-20»).
Допустимые значения номера переменной: положительное число 1 ч 32767.
Кнопка «**Удалить**» – производит удаление перьев в соответствии с настройками в других элементах управления. При нажатии кнопки «Удалить» производится удаление существующих перьев, входящих в указанный диапазон ID перьев.
- Кнопка «**Отмена**» – для закрытия диалогового окна без сохранения изменений.

Редактирование событийных перьев

Редактирование событийных перьев осуществляется с помощью диалогового окна «**Перья – Настройка**» (рисунок 10.9), которое вызывается при нажатии на кнопку «**Изменить**» или при активации соответствующего пункта контекстного меню. Кнопка «Изменить» доступна только при выделении одного или нескольких перьев.

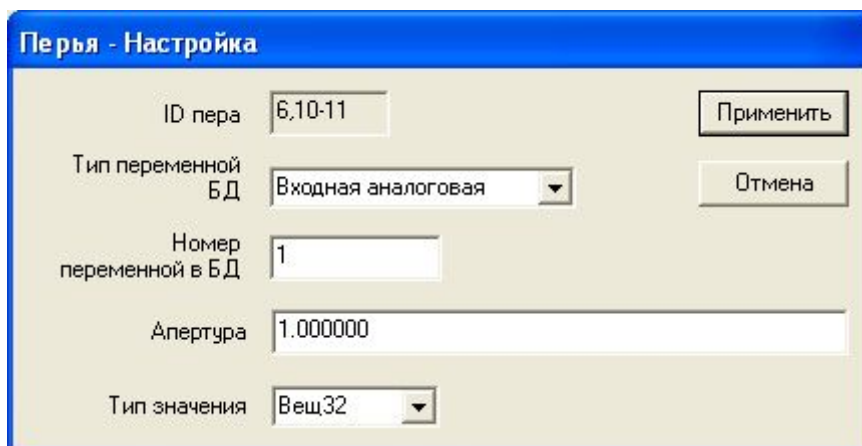


Рисунок 10.9 – Диалоговое окно «Перья – Настройка»

Окно «**Перья – Настройка**» содержит элементы управления, необходимые для настройки одного текущего или нескольких выделенных перьев:

- **«ID пера»** – информативное поле, в случае выделения одного пера содержит его идентификатор. В случае настройки нескольких перьев сразу элементы управления содержат общие для них настройки. В этом случае поле ID содержит список ID редактируемых перьев через запятую с использованием диапазонов (например, при редактировании перьев с ID 1,2,3,4,5,15,20 поле ID будет содержать "1-5,15,20").
- Выпадающий список **«Тип переменной БД»** – для выбора типа переменной БД. Доступны следующие значения: «Входная аналоговая», «Аналоговая выходная», «РВ составная», «Входная дискретная», «Дискретная выходная». Значение по умолчанию – «Входная аналоговая».
- Поле ввода **«Номер переменной в БД»** – для задания номера переменной в БД. Допустимые значения номера переменной: целое положительное число от 1 до 32767. Значение по умолчанию: 1.
- Поле ввода **«Апертура»** – для задания апертуры для определения изменения значений, которые необходимо сохранять (задается в абсолютной величине). Допустимые значения: число с плавающей точкой 0.0 или от 0,0001 до максимальное значение типа *float*. Значение по умолчанию: 0.0
- Выпадающий список **«Тип значения»** – для выбора типа значения, сохраняемого в тренде, соответствующем данному перу. Данное поле доступно только при типе источника данных самописца "Драйвер" (свойство самописца "Тип источника данных"). Список допустимых значений:
 - пустое поле – используется значение, соответствующее типу переменной БД
 - Вещ32
 - Лог
 - Цел8; Цел16; Цел32; Цел8 (беззнак); Цел16 (беззнак); Цел32 (беззнак).

В случае настройки нескольких перьев сразу, элементы управления будут содержать общие для них настройки. При изменении и сохранении настроек изменения будут применяться для всех выделенных перьев.

При вводе недопустимых значений в поля ввода (кроме полей ввода для задания диапазонов) во всех вышеперечисленных диалоговых окнах недопустимое значение заменяется на ближайшее допустимое без вывода предупреждений.

6 БИБЛИОТЕКАРЬ ЯЗЫКА КРУГОЛ

6.1 Назначение

«Библиотекарь языка КРУГОЛ», далее по тексту «Библиотекарь», предназначен для создания и редактирования описаний библиотек функций.

Библиотекарь обеспечивает:

- Создание описания библиотеки функций
- Редактирование описания библиотеки функций
- Добавление описания функции в библиотеку функций
- Редактирование описания функции
- Удаление описания функции из библиотеки.

6.2 Интерфейс пользователя

6.2.1 Главное окно

Главное окно Библиотекаря приведено на рисунке 6.2.1.

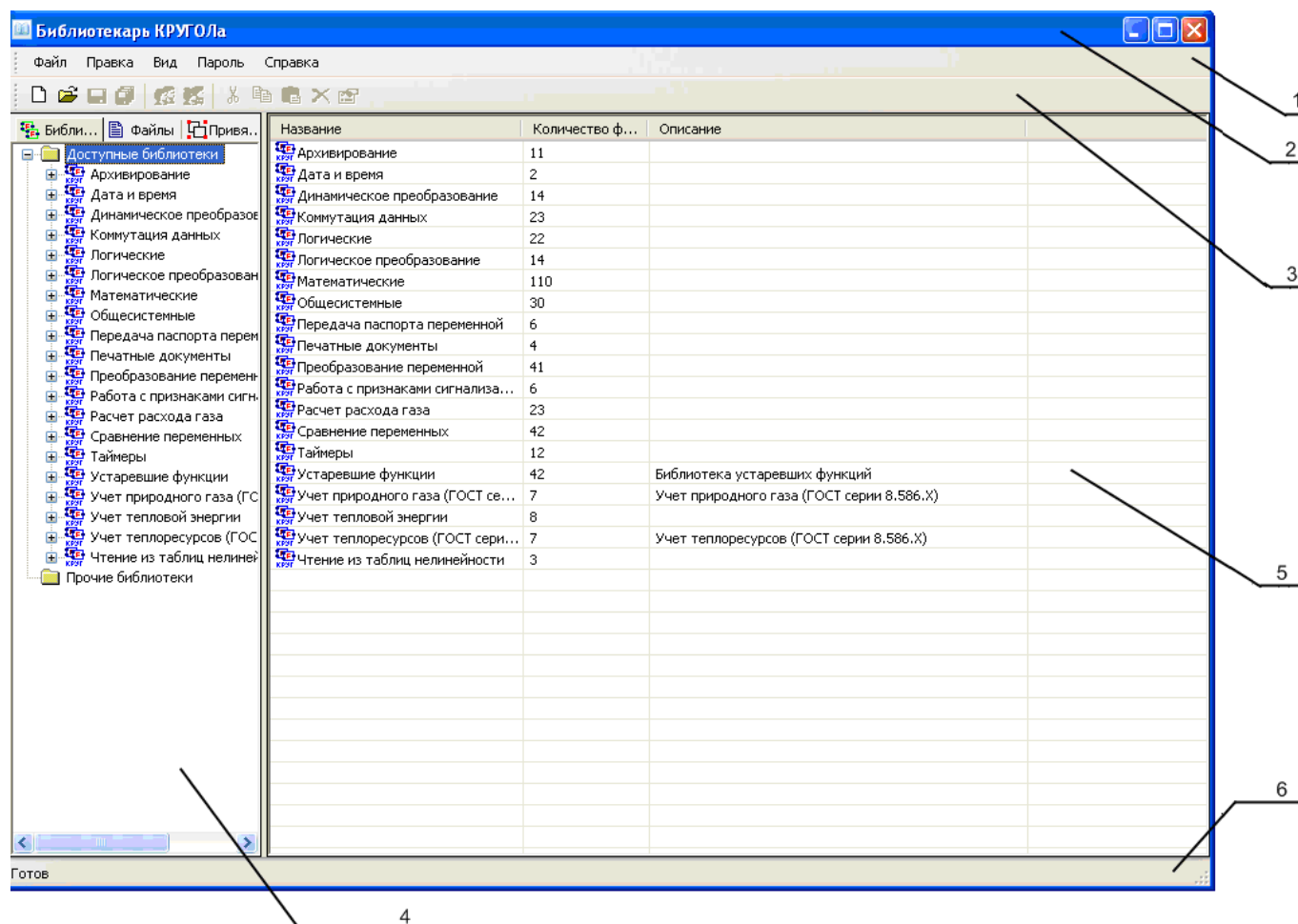


Рисунок 6.2.1 – Главное окно Библиотекаря

В верхней части окна находится заголовок (1), меню (2) и панель инструментов(3). Центральная часть окна разделена на две части. В левой части окна (4) располагается дерево описания библиотек. В правой части окна (5) располагается таблица, в которой отображаются дополнительные сведения о выбранной библиотеке (выбранный узел дерева). Внизу окна располагается панель состояния (6).

6.2.2 Меню

Меню содержит команды, доступные Пользователю (рисунок 6.2.2).

Рисунок 6.2.2 – Меню Библиотекаря

- **Файл** – команды работы с файлами
- **Правка** – команды редактирования
- **Вид** – команды настройки внешнего вида окна Библиотекаря
- **Пароль** – команды ввода и блокирования пароля
- **Справка** – команды вызова справки.

6.2.2.1 Меню "Файл"

Меню «**Файл**» содержит следующие команды (рисунок 6.2.3):

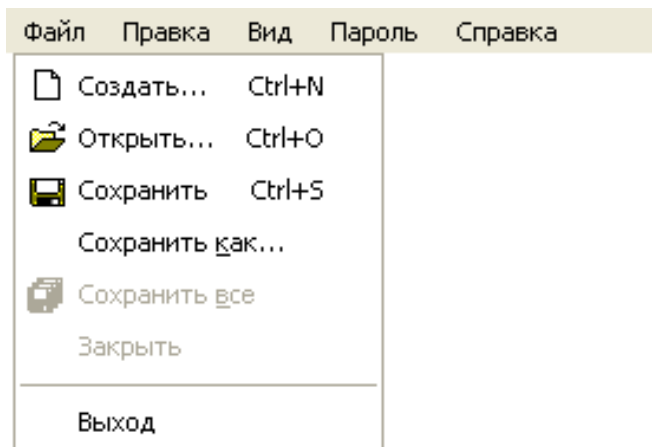


Рисунок 6.2.3 – Меню «Файл»

- «**Создать**» – создание описания новой библиотеки. При выборе команды выводится окно «Новая библиотека» для задания параметров библиотеки (рисунок 6.2.4). На рисунке 6.2.4:
 - «**Название**» – название библиотеки
 - «**Файл**» – путь к файлу, в котором будет сохраняться описание библиотеки и функций, входящих в нее
 - «**Описание**» – краткое текстовое описание библиотеки.
 - «**Защитить паролем**» – признак использования пароля.
Если его задать, то изменение описания библиотеки и функций, входящих в нее, возможно только после ввода пароля
 - «Пароль» и «Подтверждение» – поля для ввода пароля

- «**Системная библиотека**» – данный флажок означает, что функции этой библиотеки разработаны в фирме "КРУГ" или «КРУГ-Софт»
- «**Файлы с реализациями функций**» – необходимо указать названия файлов, которые содержат реализацию функций библиотеки.

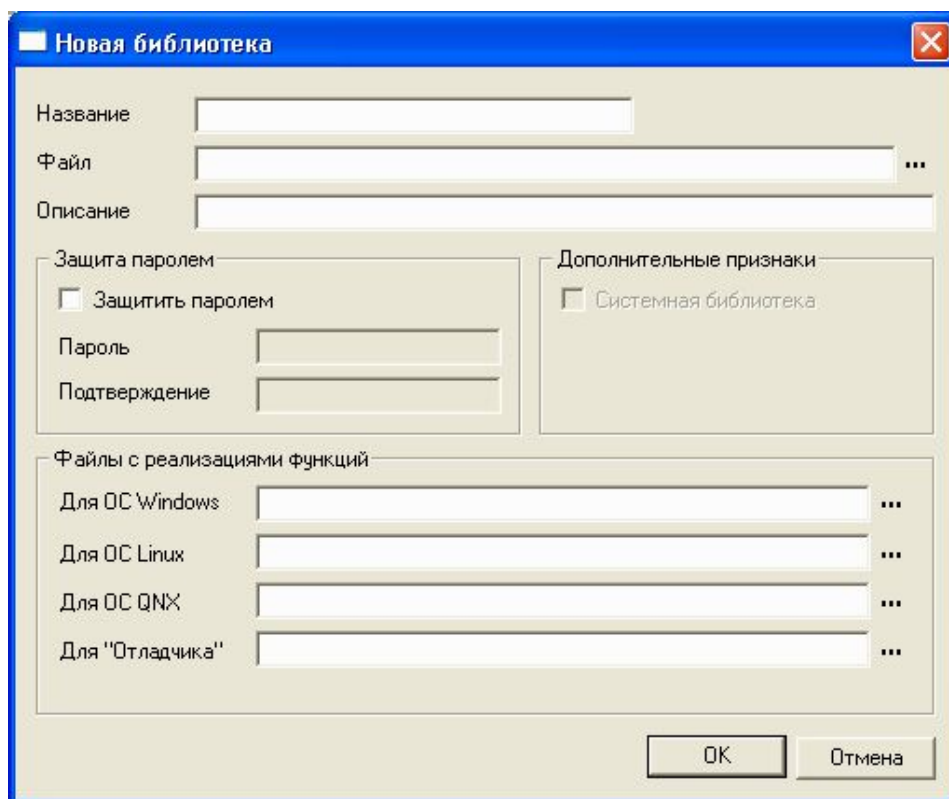


Рисунок 6.2.4 – Параметры библиотеки

- «**Открыть**» – открытие описания библиотеки. При выборе команды выводится диалоговое окно выбора файла библиотеки (рисунок 6.2.5)

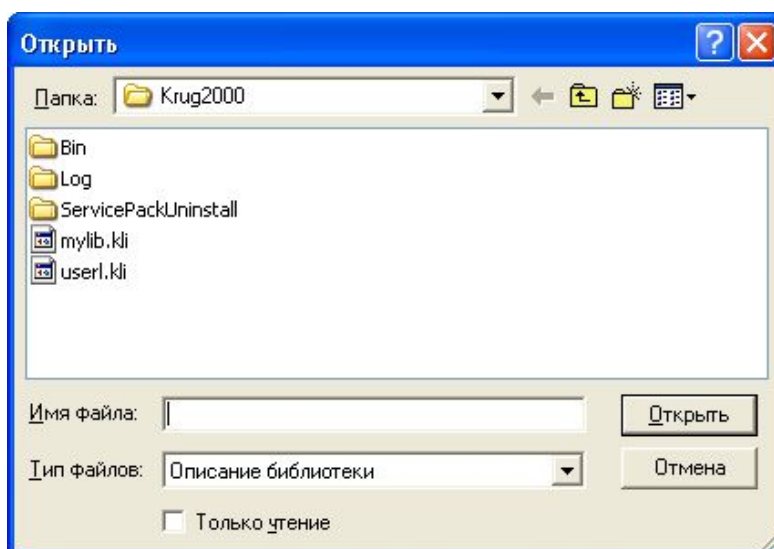


Рисунок 6.2.5 – Выбор файла библиотеки

- «**Сохранить**» – сохранение описания библиотеки

- **«Сохранить как...»** – сохранение описания библиотеки под новым именем (диалоговое окно на рисунке 6.2.6)

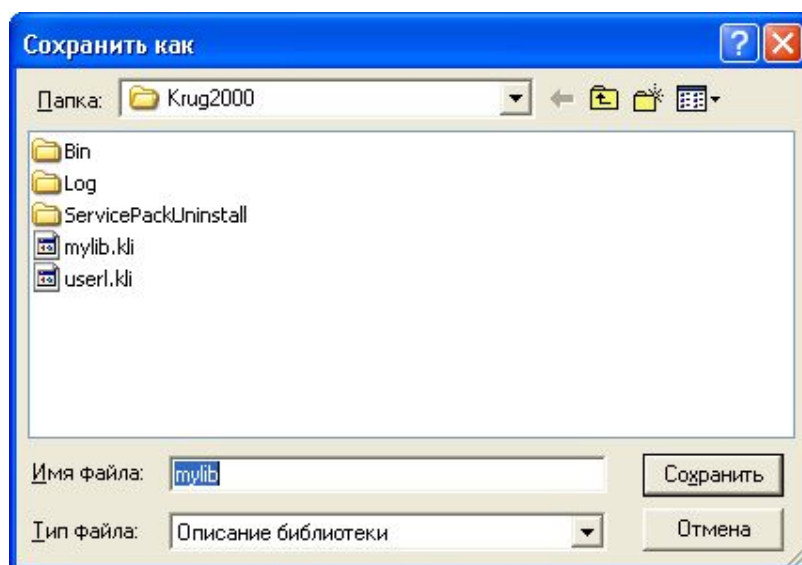


Рисунок 6.2.6 – Окно сохранения библиотеки

- **«Сохранить все»** – сохранение изменений во всех библиотеках.
- **«Заккрыть»** – закрытие описания библиотеки. Команда доступна только для библиотек, расположенных вне папки "KRUG2000\Bin".
- **«Выход»** – завершение работы.

Если есть несохраненные изменения, то выводится запрос на сохранение изменений.

6.2.2.2 Меню "Правка"

Меню **«Правка»** содержит следующие команды (рисунок 6.2.7):

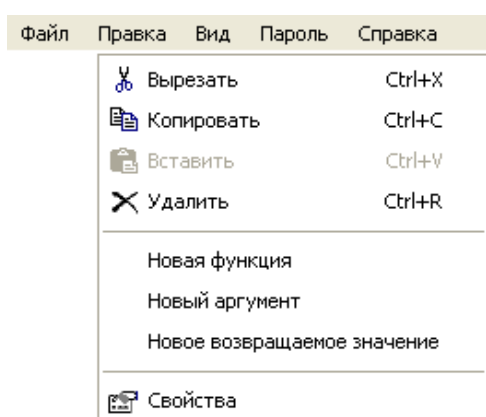


Рисунок 6.2.7 – Меню «Правка»

- **«Вырезать»** – копирование выделенного элемента в буфер обмена и удаление из списка элементов
- **«Копировать»** – копирование выделенного элемента в буфер обмена
- **«Вставить»** – вставка элемента из буфера обмена

- «Удалить» – удаление выделенного элемент
- «Новая функция» – создание описания новой функции. При выборе команды выводится диалоговое окно описания функции (рисунок 6.2.8)

Рисунок 6.2.8 – Описание новой функции

На рисунке 6.2.8:

- «Библиотека» – имя библиотеки, в которую добавляется функция
- «Номер функции» – номер функции (только для функций системных библиотек)
- «Название функции 1», «Название функции 2» – два названия функции
- «Название экспортируемой функции» – имя функции, как оно задано в исходном тексте функции (на языке C/C++)
- «Описание» – текстовое описание функции
- «Номер алгоблока» – признак использования номера алгоблока.
Если задан, то считается, что самый первый аргумент функции используется для передачи номера алгоблока. Первый аргумент функции должен иметь целый тип
- «Тип привязки» – список, в котором необходимо выбрать тип привязки функции. Привязка позволяет транслятору контролировать использование функций
- «Переменное количество входов» – признак переменного количества входов. Если признак установлен, то считается, что функция кроме описанных входов может иметь еще несколько входов, количество и тип которых может меняться и определяется во время исполнения программы. Например, переменное количество входов имеет функция «message»
- «Переменное количество выходов» – признак переменного количества выходов. Если признак установлен, то считается, что функция может иметь кроме описанных выходов еще несколько выходов, количество и тип которых определяются во время исполнения программы
- «Входы» – в этой таблице перечисляются все входы функции
- «Выходы» – в этой таблице перечисляются все выходы функции

- **«Новый аргумент»** – создание описание аргумента функции. При выборе команды открывается диалоговое окно описания аргумента (рисунок 6.2.9).

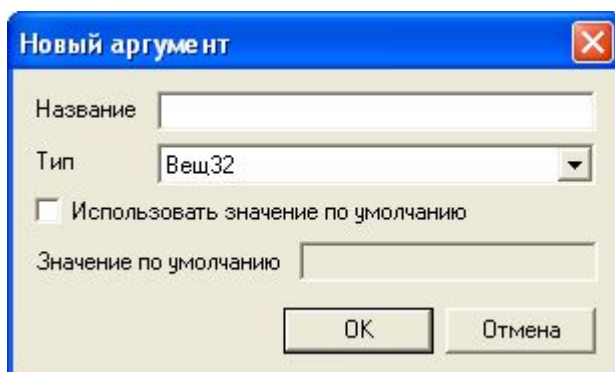


Рисунок 6.2.9 – Описание аргумента функции

На рисунке 6.2.9:

- **«Название»** – название аргумента
 - **«Тип»** – список доступных типов для выбора типа значения аргумента
 - **«Использовать значение по умолчанию»** – признак использования значения по умолчанию. Значение по умолчанию позволяет не подключать вход функции на схемах ФБД и пропускать аргумент функции в программах СТ
 - **«Значение по умолчанию»** – поле ввода значения по умолчанию.
- **«Новое возвращаемое значение»** – описание возвращаемого значения функции (рисунок 6.2.10). Включает задание названия и типа (выбирается из списка) возвращаемого значения.

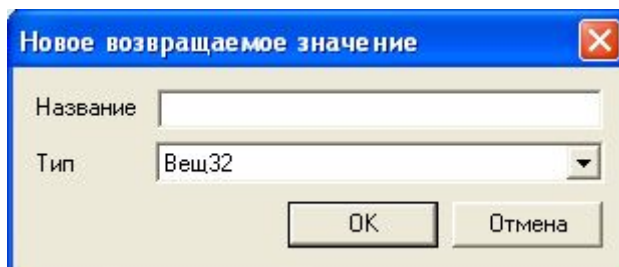


Рисунок 6.2.10 – Задание возвращаемого значения функции

- **«Свойства»** – описание свойств выделенного элемента. При вызове команды выводится диалоговое окно свойств, вид которого зависит от выбранного элемента:
 - **Для библиотеки** – окно **«Свойства библиотеки»** (идентично окну **«Новая библиотека»**, рисунок 6.2.4).
 - **Для функции** – окно **«Свойства функции»** (рисунок 6.2.11)
 - **Для входа функции** – окно **«Свойства аргумента»** (рисунок 6.12а)
 - **Для выхода функции** – окно **«Свойства возвращаемого значения»** (рисунок 6.12.б).

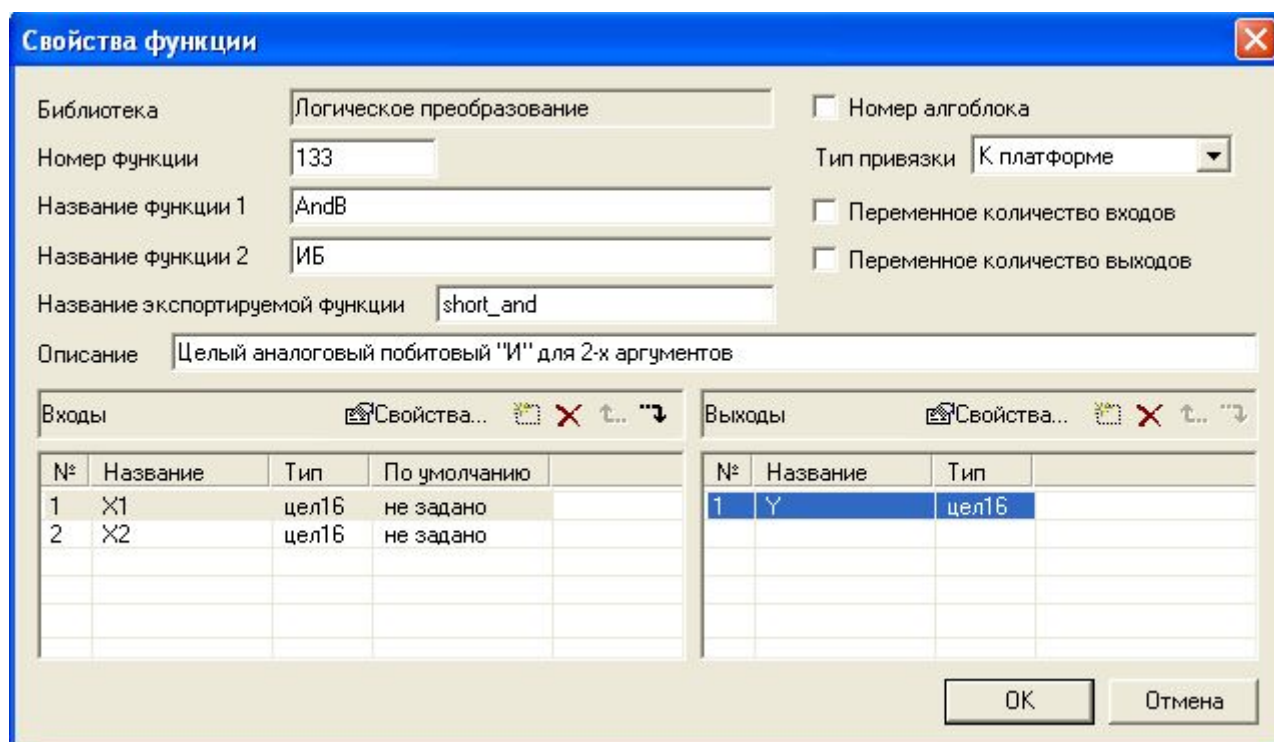


Рисунок 6.2.11 – Свойства функции

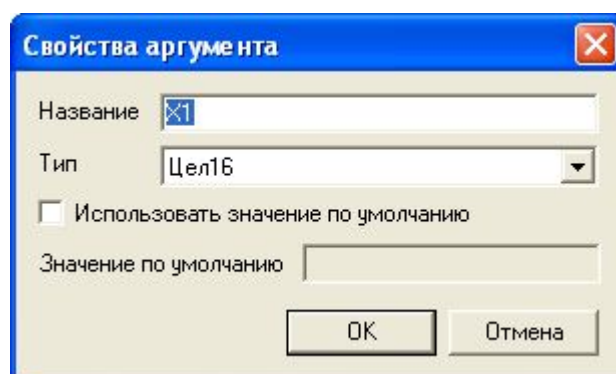


Рисунок 6.2.12а – Свойства входа функции

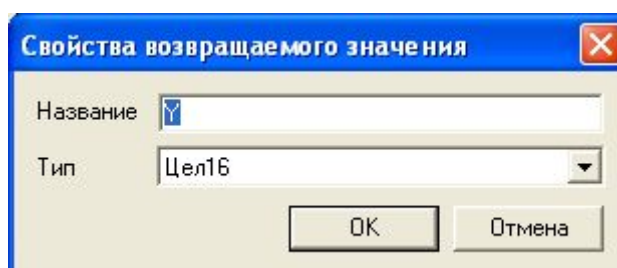


Рисунок 6.2.12б – Свойства выхода функции

6.2.2.3 Меню «Вид»

Меню «**Вид**» содержит следующие команды (рисунок 6.2.13):

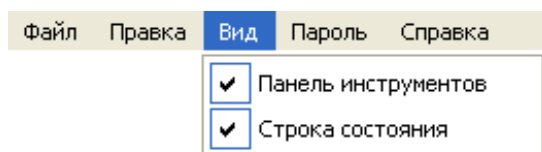


Рисунок 6.2.13 – Меню «Вид»

- **«Панель инструментов»** – показывает/прячет панель инструментов
- **«Строка состояния»** – показывает/прячет строку состояния.

6.2.2.4 Меню «Пароль»

Меню «**Пароль**» содержит следующие команды (рисунок 6.2.14):

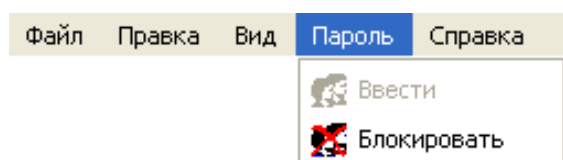


Рисунок 6.2.14 – Меню «Пароль»

- **«Ввести»** – позволяет ввести пароль для библиотеки, защищенной паролем
- **«Блокировать»** – позволяет заблокировать пароль.

6.2.2.5 Меню «Справка»

Меню "**Справка**" содержит следующие команды (рисунок 6.2.15):

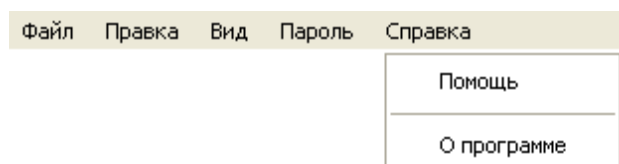


Рисунок 6.2.15 – Меню «Справка»

- **«Помощь»** – вызов помощи
- **«О программе»** – выводит краткую информацию о программе(рисунок 6.2.16).



Рисунок 6.2.16 – Окно «О программе»

6.3 Панель инструментов

Панель инструментов используется для быстрого вызова часто используемых команд (рисунок 6.3.1):



Рисунок 6.3.1 – Панель инструментов

- – дублирует команду меню "Файл\Создать".
- - дублирует команду меню "Файл\Открыть".
- - дублирует команду меню "Файл\Сохранить".
- - дублирует команду меню "Файл\Сохранить все".
- - дублирует команду меню "Пароль\Ввести".
- - дублирует команду меню "Пароль\Блокировать".
- - дублирует команду меню "Правка\Вырезать".
- - дублирует команду меню "Правка\Копировать".
- - дублирует команду меню "Правка\Вставить".
- - дублирует команду меню "Правка\Удалить".
- - дублирует команду меню "Правка\Свойства".

6.3.1 "Библиотеки"

На вкладке «**Библиотеки**» в виде дерева отображаются библиотеки, доступные для использования, а также загруженные пользователем (рисунок 6.3.2).

Дерево библиотек содержит два узла «**Доступные библиотеки**» и «**Прочие библиотеки**» (рисунок 6.3.3).

При запуске Библиотекарь КРУГОЛ сканирует папку «**Bin**» и загружает описания всех библиотек, которые встретит. Все эти библиотеки попадают в узел «Доступные библиотеки», что означает, что функции из этих библиотек могут быть использованы в программах на КРУГОЛ.

Если открыть библиотеку, расположенную не в папке «**Bin**», то она попадет в узел «Прочие библиотеки». Такие библиотеки не доступны для использования в КРУГОЛ.

Если в дереве выбран список библиотек, то в таблице справа отображается список библиотек. Для каждой библиотеки в таблице показывается название библиотеки, количество функций в библиотеке и описание библиотеки.

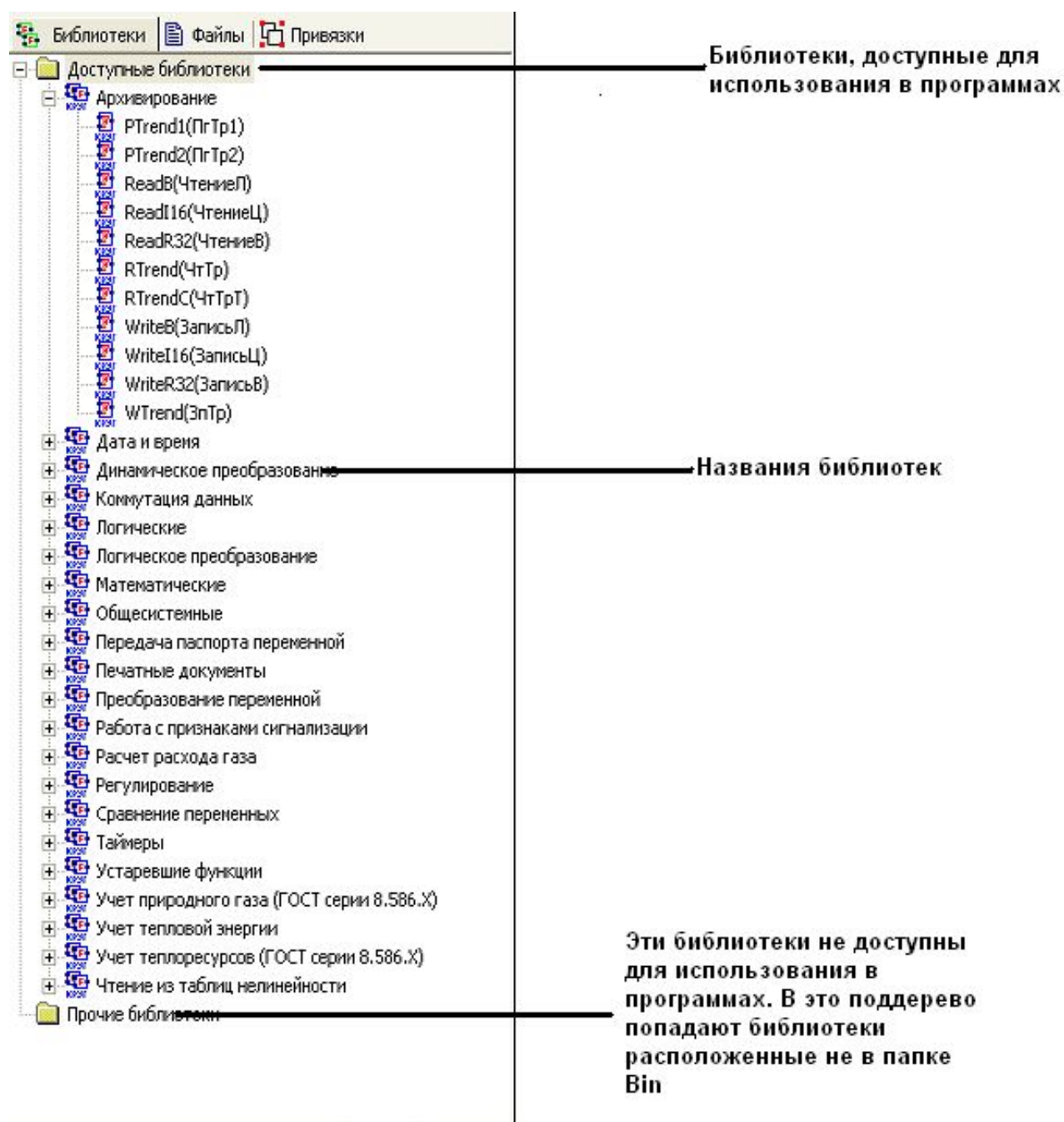


Рисунок 6.3.2 – Вкладка «Библиотеки»

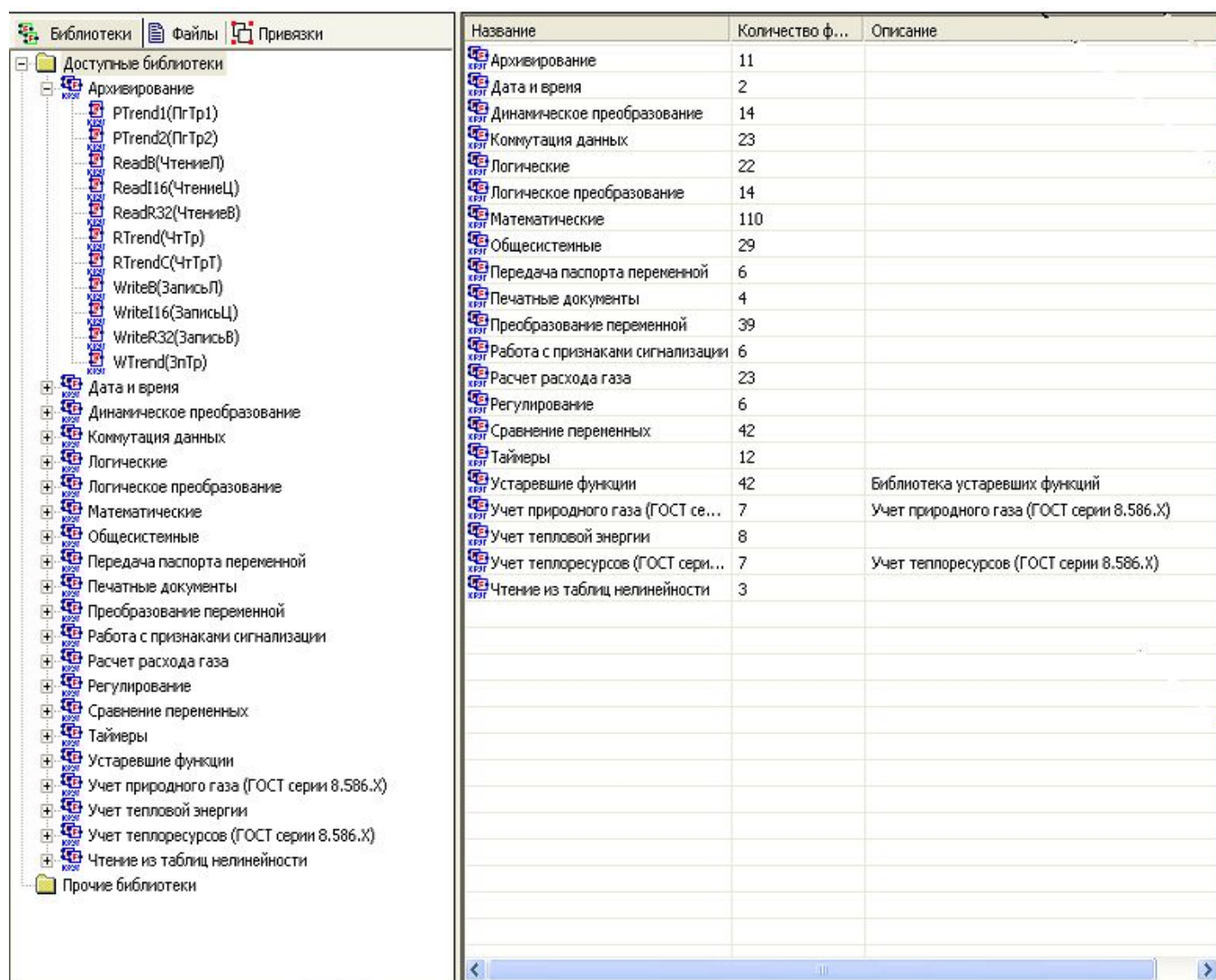


Рисунок 6.3.3 – Информация о доступных библиотеках

Если в дереве выбрана библиотека, то в таблице справа отображается список функций, входящих в библиотеку.

Для каждой функции показываются оба имени функции и описание функции (рисунок 6.3.4).

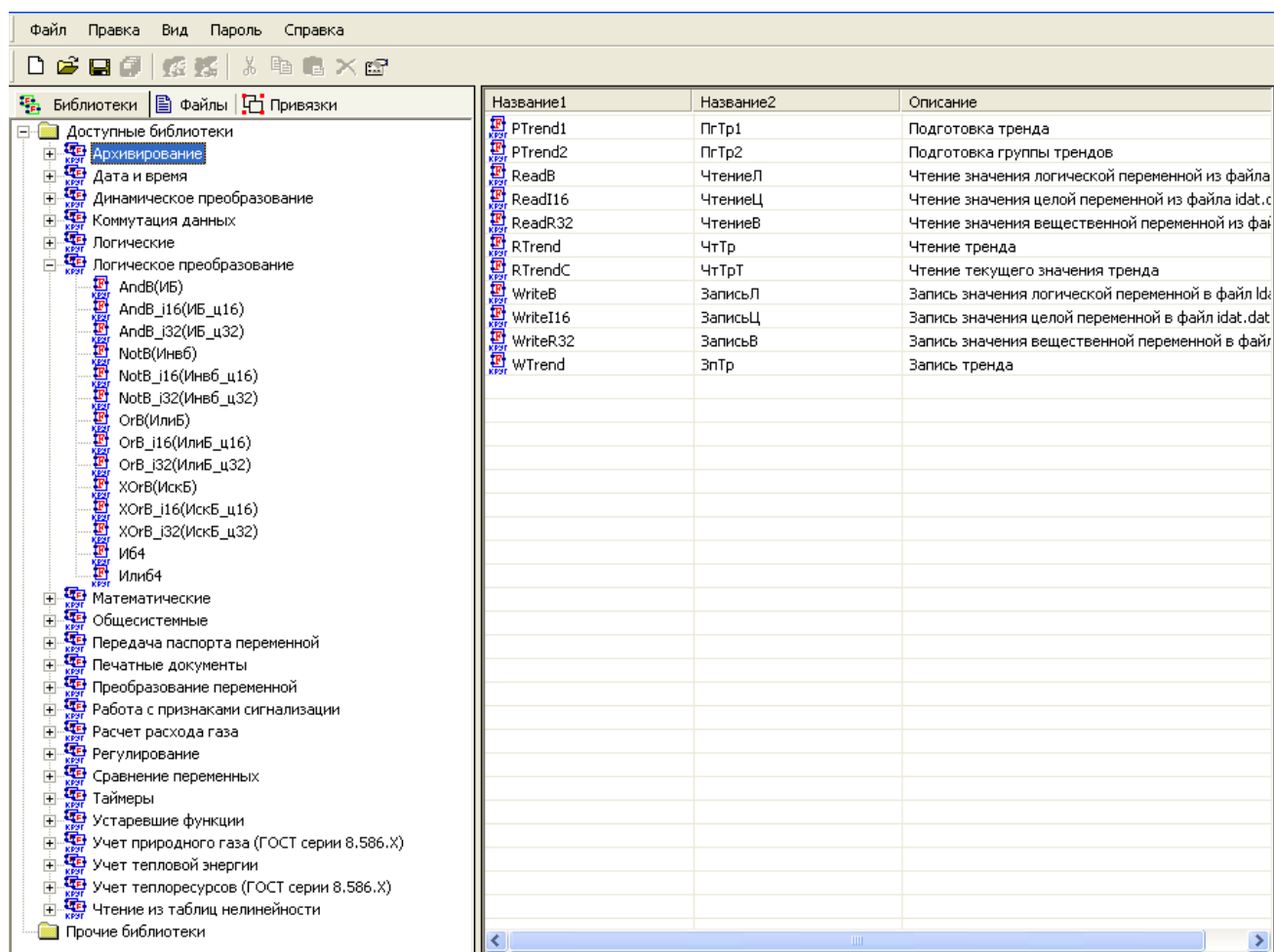


Рисунок 6.3.4 – Информация о библиотеке «Архивирование»

Если в дереве выбрана функция, то в таблице справа отображается список аргументов и возвращаемых значений функции.

Для каждого аргумента указывается порядковый номер, вход или выход, тип значения и значение по умолчанию (рисунок 6.3.5).

Библиотеки	Файлы	Привязки	№	Название	Вход...	Тип	По умолчанию
Доступные библиотеки			1	Номер самописца	Вход	Цел16	
Архивирование			2	Номер пера	Вход	Цел16	
PTrend1(ПтTp1)			3	Отступ в циклах	Вход	Цел16	
PTrend2(ПтTp2)			4	Отступ в циклах	Вход	Цел16	
ReadB(ЧтениеБ)							
ReadI16(ЧтениеЦ)							
ReadR32(ЧтениеВ)							
RTrend(ЧтTp)							
RTrendC(ЧтTpT)							
WriteB(ЗаписьБ)							
WriteI16(ЗаписьЦ)							
WriteR32(ЗаписьВ)							
WTrend(ЗнTp)							
Дата и время							
Динамическое преобразование							
Коммутация данных							
Логические							
Логическое преобразование							
AndB(ИБ)							
AndB_j16(ИБ_ц16)							
AndB_j32(ИБ_ц32)							
NotB(ИнеБ)							
NotB_j16(ИнеБ_ц16)							
NotB_j32(ИнеБ_ц32)							
OrB(ИлиБ)							
OrB_j16(ИлиБ_ц16)							
OrB_j32(ИлиБ_ц32)							
XOrB(ИскБ)							
XOrB_j16(ИскБ_ц16)							
XOrB_j32(ИскБ_ц32)							
ИБ4							
ИлиБ4							
Математические							
Общесистемные							
Передача паспорта переменной							
Печатные документы							
Преобразование переменной							

Рисунок 6.3.5 – Информация об аргументах функции

6.3.2 "Файлы"

Вкладка «**Файлы**» позволяет просмотреть все файлы, с которыми связана библиотека. Файлы отображаются справа в таблице (рисунок 6.3.6).

Библиотека может иметь четыре реализации функции:

- Для платформы "WINDOWS"
- Для платформы "LINUX"
- Для платформы "QNX"
- Для отладчика.

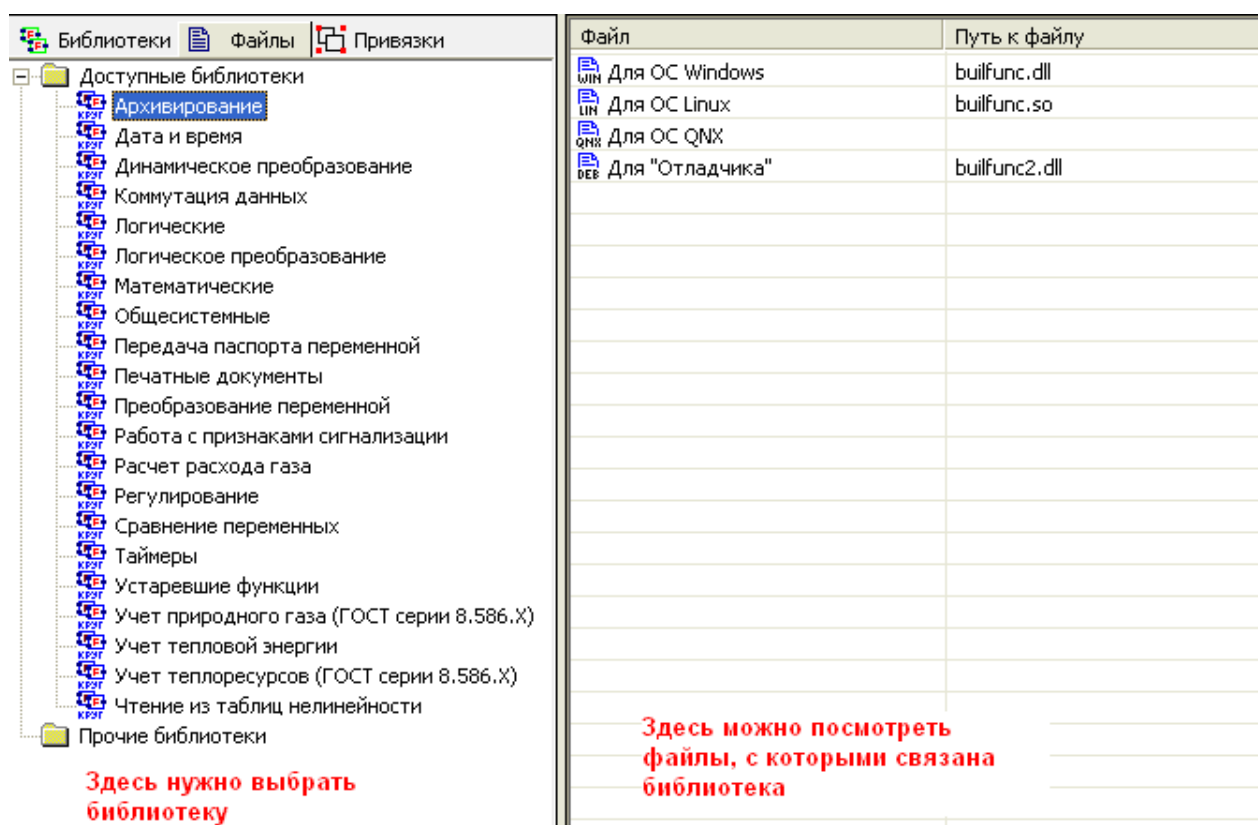


Рисунок 6.3.6 – Вкладка «Файлы»

6.3.3 "Привязки"

Эта вкладка позволяет посмотреть привязки функции.

Привязки используются компилятором для проверки корректности использования функций.

Что дает использование привязок?

Допустим, что в новой среде разработки создается технологическая программа для достаточно старой версии CPB. В этой ситуации вполне возможно допустить ошибку и использовать функцию, которая в этой CPB не реализована. Эта ошибка проявится только при выполнении программы.

Для исключения подобных ошибок введены привязки. Используя их, компилятор может контролировать использование функций на этапе компиляции программ (рисунки 6.3.7).

	Среда исполнения КРУГОЛ 1.0 Windows	Среда исполнения КРУГОЛ 2.0 Windows	Среда исполнения КРУГОЛ 2.1 Windows	Среда исполнения КРУГОЛ 2.2 Windows	СРВК TREI-5B-02 6.5 QNX	СРВК TREI-5B-02 6.5 Linux	СРВК МФК 6.5 Linux	СРВК TREI-5B-02 7.0 Linux	СРВК МФК 7.0 Linux	СРВК МФК3000 7.0 Linux	СРВК TREI-5B-02 7.1 Linux	СРВК МФК3000 7.1 Linux	СРВК P06 7.1x Linux	СРВК МФК3000 8.0 Linux	СРВК TREI-5B-02 8.0 Linux	СРВК P06 8.0 Linux
PTrend1(ПрТр1)	✓	✓	✓	✓												
PTrend2(ПрТр2)	✓	✓	✓	✓												
ReadB(ЧтениеЛ)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
ReadI16(ЧтениеЦ)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
ReadR32(ЧтениеВ)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
RTrend(ЧтТр)	✓	✓	✓	✓												
RTrendC(ЧтТрТ)	✓	✓	✓	✓												
WriteB(ЗаписьЛ)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
WriteI16(ЗаписьЦ)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
WriteR32(ЗаписьВ)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
WTrend(ЗнТр)	✓	✓	✓	✓												

Рисунок 6.3.7 – Привязки функции

Привязки существуют двух типов:

- **«Привязка к типу платформы»** – позволяет привязать функцию к семейству СРВ.
- **«Привязка к платформе»** – позволяет привязать функцию к определенной платформе.

Примечание:

Возможно описание пользовательских функций, которые имеют одинаковые имена, но разные привязки. Это связано с особенностями реализации функций под разные версии одной и той же платформы.

6.4 Пример создания библиотеки

6.4.1 Порядок создания библиотеки

В этом разделе рассмотрим порядок создания описания библиотеки, описания функции и использование этой функции в технологической программе.

Для этого необходимо выполнить следующие шаги:

- 1 Реализация функции на языке C/C++.
- 2 Создание описания библиотеки.
- 3 Создание описания функции.
- 4 Использование функции в технологической программе.

6.4.2 Реализация функции на языке C/C++

Первым шагом необходимо реализовать функцию на языке C/C++.

В качестве примера рассмотрим функцию, которая выполняет сложение вещественных аргументов.

Функция имеет два входа: **x1** и **x2**, за которыми может еще следовать несколько входов **args**, и один выход **y**.

```
BOOL __stdcall MyAdd(float x1, float x2, KLSafeArray* args, float* y)
{
    *y = x1 + x2;
    for (DWORD i = 0; i < args->GetCount(); i++)
    {
        if ((*args)[i].vt == KLSafeArray::KL_REAL)
        {
            *y += ((*args)[i].fltVal);
        }
    }
    return TRUE;
}
```

6.4.3 Создание описания библиотеки

Этот шаг описывает создание описания библиотеки. Если описание библиотеки уже существует, то переходите к следующему шагу.

Для создания библиотеки перейдите на вкладку «**Библиотеки**», в дереве выберите узел «**Прочие библиотеки**» и выберите команду «**Файл\Создать...**» (рисунок 6.4.1).

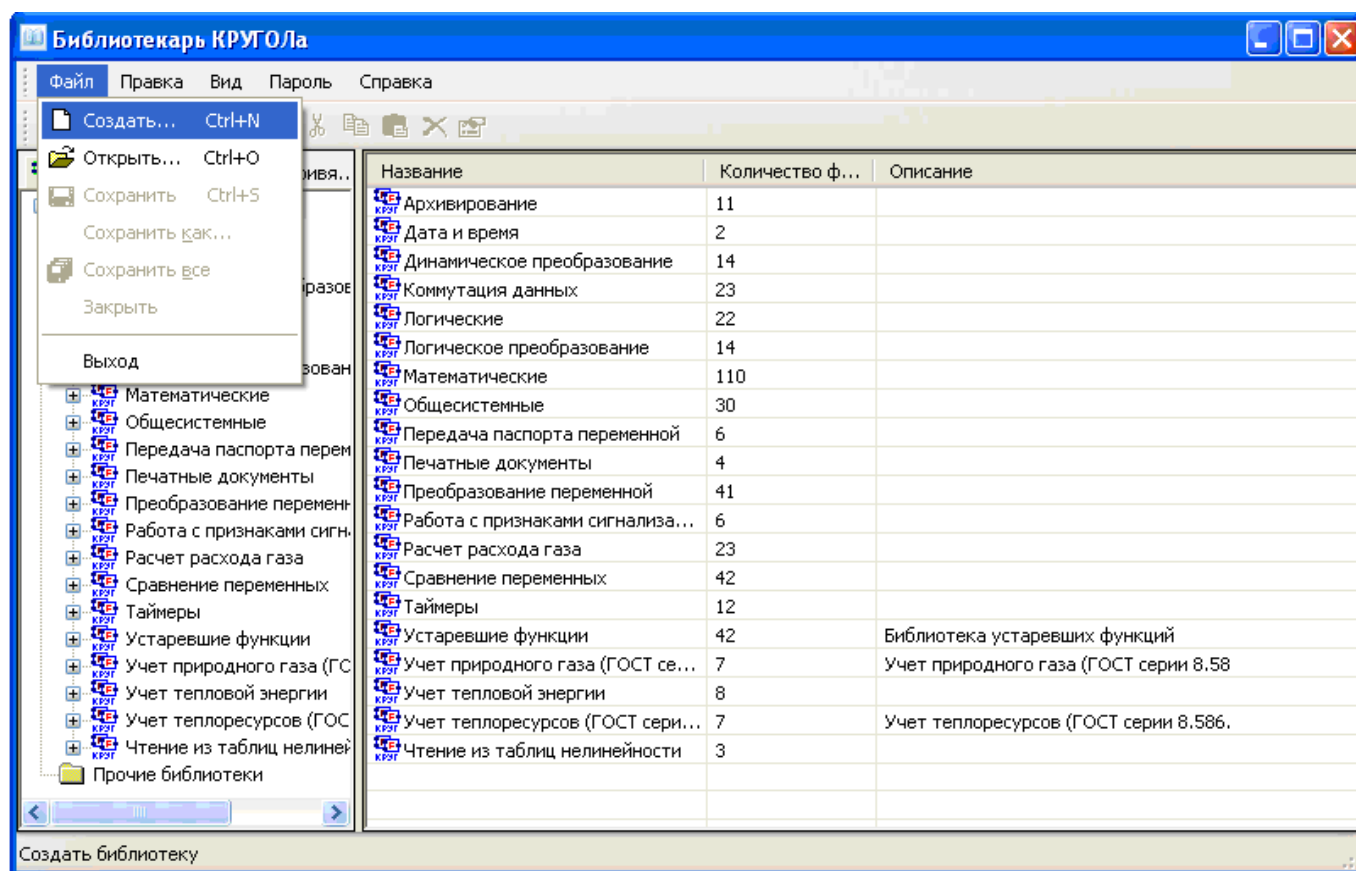


Рисунок 6.4.1 – Создание описания библиотеки

В появившемся диалоговом окне нужно настроить свойства описания библиотеки (рисунок 6.4.2).

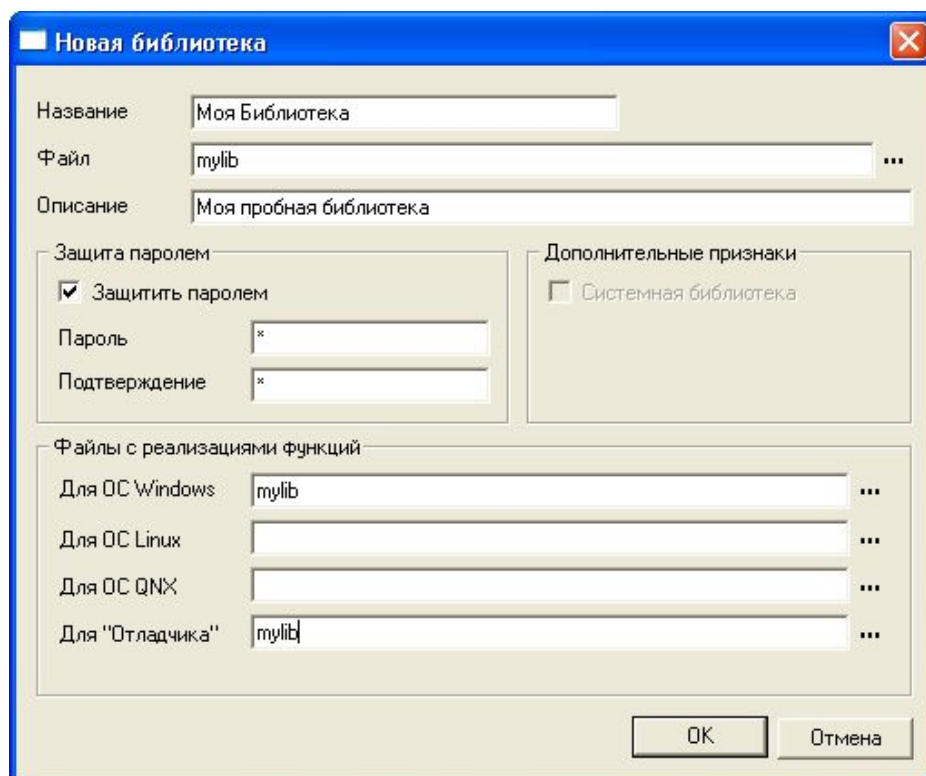


Рисунок 6.4.2 – Новая библиотека

Оставляем значения свойств такими, как на рисунке 6.4.2, и «ажимаем на кнопку **«ОК»**».

6.4.4 Создание описания функции

На этом шаге создадим описание функции. Переходим на вкладку **«Библиотеки»**. В дереве выбираем библиотеку, в которую нужно добавить описание функции. Выбираем команду **«Правка\Новая функция...»**.

В появившемся окне **«Свойства функции»** заполняем свойства функции, как показано на рисунке 6.4.3.

Свойства функции

Библиотека: Моя библиотека

Номер функции: 0

Название функции 1: MyAdd

Название функции 2:

Название экспортируемой функции: MyAdd

Описание: Сложение вещественных чисел

☐ Номер алгоблока

Тип привязки: К платформе

☐ Переменное количество входов

☐ Переменное количество выходов

№	Название	Тип	По умолчанию
1	X1	вещ32	не задано
2	X2	вещ32	не задано

№	Название	Тип
1	Y	вещ32

ОК Отмена

Рисунок 6.4.3 – Свойства функции

На этом описание функции создано.

Сохраняем описание библиотеки, для этого выбираем команду **«Файл\Сохранить все»**.

6.4.5 Использование функции в технологической программе

Чтобы использовать созданную функцию в технологической программе следует:

- 1 Скопировать описание созданной библиотеки – файл **mylib.kli** – в папку **«KRUG2000\bin»**..
- 2 Запустить Интегрированную среду разработки (ИСР) КРУГОЛ

- 3 Создать программу ФБД и разместить в программе только что созданную функцию (рисунок 6.4.4).

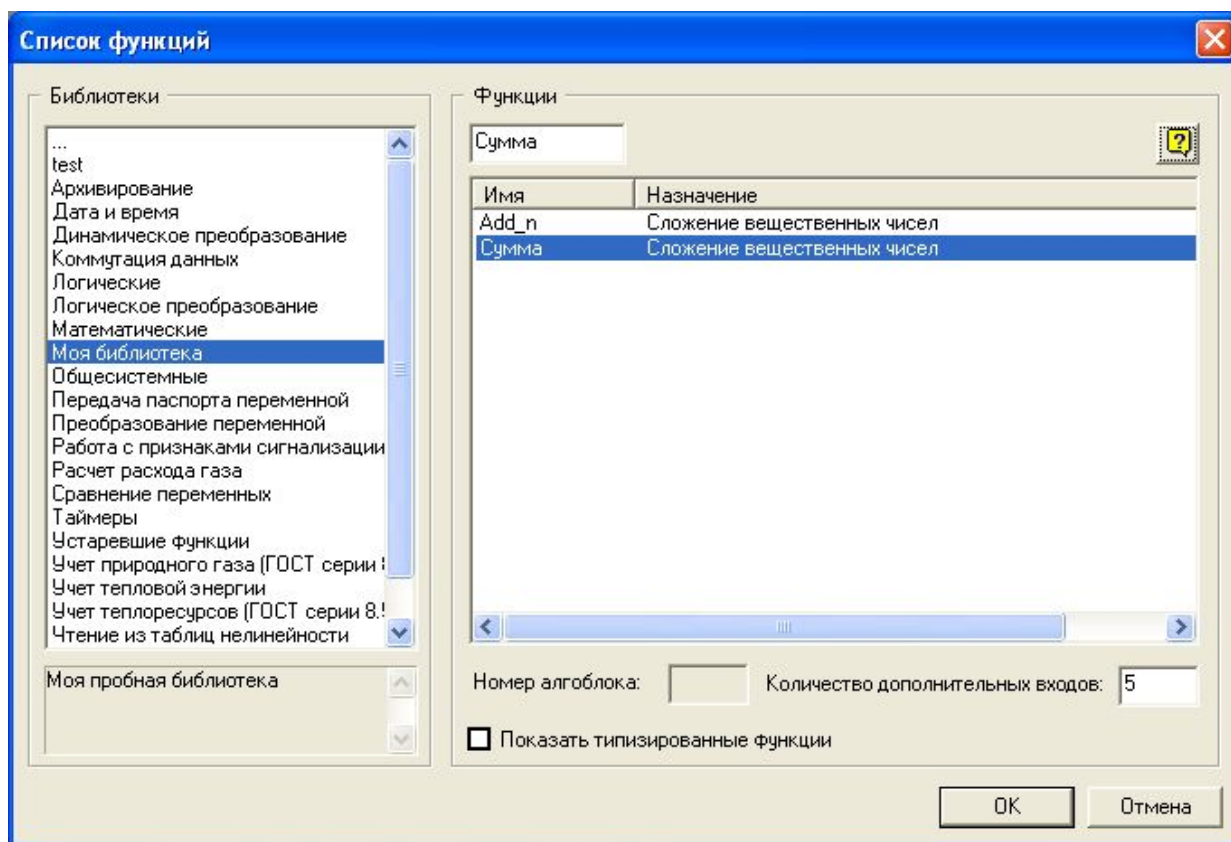


Рисунок 6.4.4 – Список функций

Поставить функцию на схему (рисунок 6.4.5).

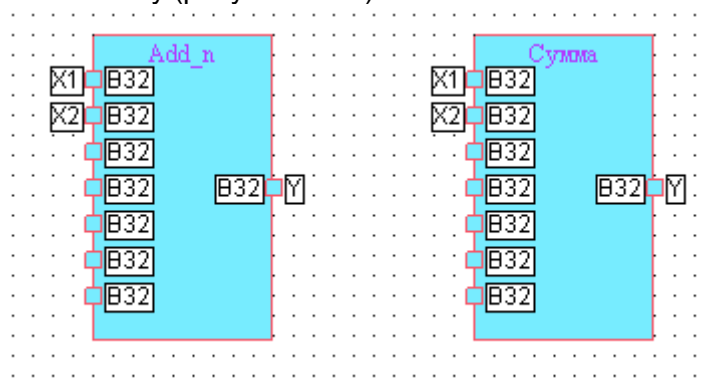


Рисунок 6.4.5 - Элемент ФБД – функция «add_n»

Чтобы использовать функцию в «Ядре КРУГОЛ», скопируйте файл **mylib_wn.kli** и библиотеку **mylib.dll** в папку **bin**. После этих действий "Ядро КРУГОЛ" сможет выполнять программу, содержащую созданную функцию.

7 ВКЛЮЧЕНИЕ ПРОГРАММ ПОЛЬЗОВАТЕЛЯ В СИСТЕМУ РЕАЛЬНОГО ВРЕМЕНИ

В Системе реального времени контроллера и в Среде исполнения станции оператора могут выполняться несколько исполняемых программ Пользователя (файлы с расширением .out).

Важным является размещение исполняемых программ и их порядок.

7.1 Система реального времени контроллера

Для выполнения программ Пользователя (ПрП) в Системе реального времени контроллера (СРВК) необходимо:

- Записать программы в контроллер
- Сформировать список исполняемых в СРВК программ – файл ***programs.lst***.

Запись программ Пользователя в контроллер выполняется с помощью специальных программ. Например, для сетевой версии контроллера – с помощью программы «Станция инжиниринга».

Список программ Пользователя, выполняемых в системе реального времени контроллера – файл ***programs.lst*** – должен находиться в поддиректории ***/GSW/PRG*** контроллера.

Файлы ПрП с расширением ***.out*** должны располагаться в поддиректории ***/GSW/PRG*** файловой системы контроллера.

Рассмотрим формирование файла ***programs.lst*** для следующей структуры проекта (проект СРВК_1, рисунок 7.1.1).

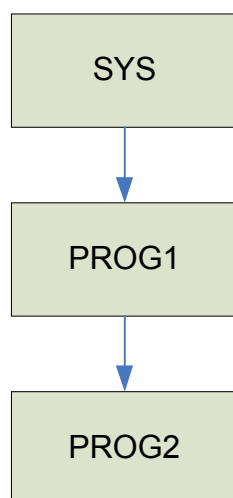


Рисунок 7.1.1 – Структура проекта СРВК_1

Проект включает три программы: **SYS**, **PROG1** и **PROG2**. Программа **SYS** – главная. Данной структуре проекта в ИСП КРУГОЛ может соответствовать, например, «Проект_СРВК_1» (рисунок 7.1.2).

В результате трансляции проекта «Проект_СРВК_1» будут сформированы файлы исполняемых в контроллере программ: ***SYS.out***, ***PROG1.out***, ***PROG2.out***.

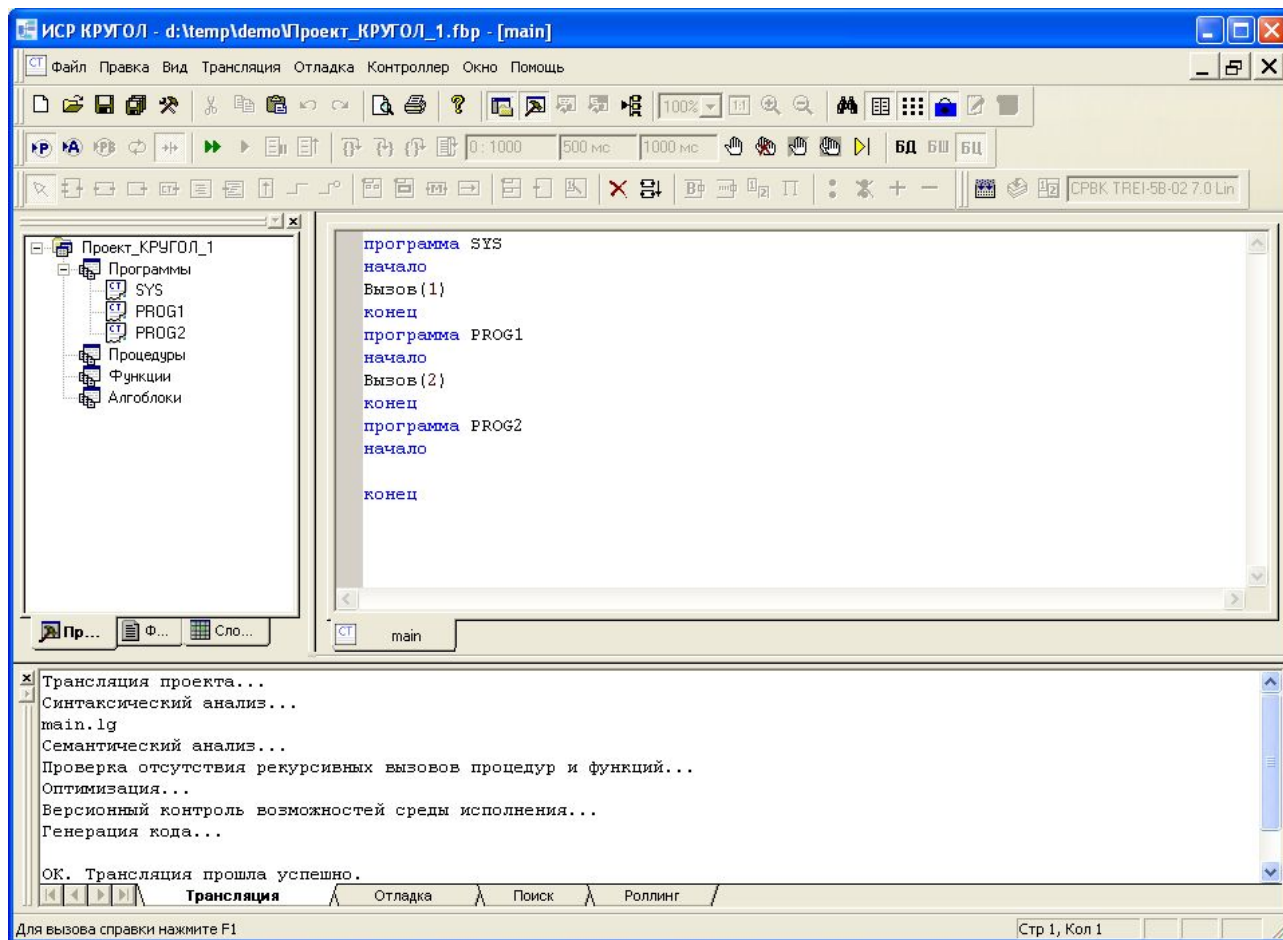


Рисунок 7.1.2 – Проект CPBK_1 в ИСР КРУГОЛ

Список в файле **programs.lst** должен содержать наименования программ, расположенные в следующем порядке:

SYS
PROG1
PROG2

Программа SYS – **главная, обязательно первая в списке – выполняется всегда в каждом цикле контроллера.**

Из главной программы могут вызываться программы, следующие за ней в списке (в данном примере – PROG1 или PROG2).

Вызов программ осуществляется с помощью функции «**вызов**». В качестве параметра эта функция получает порядковый номер программы из списка (файл **programs.lst**):

- вызов(1) – вызов программы PROG1
- вызов(2) – вызов программы PROG2.

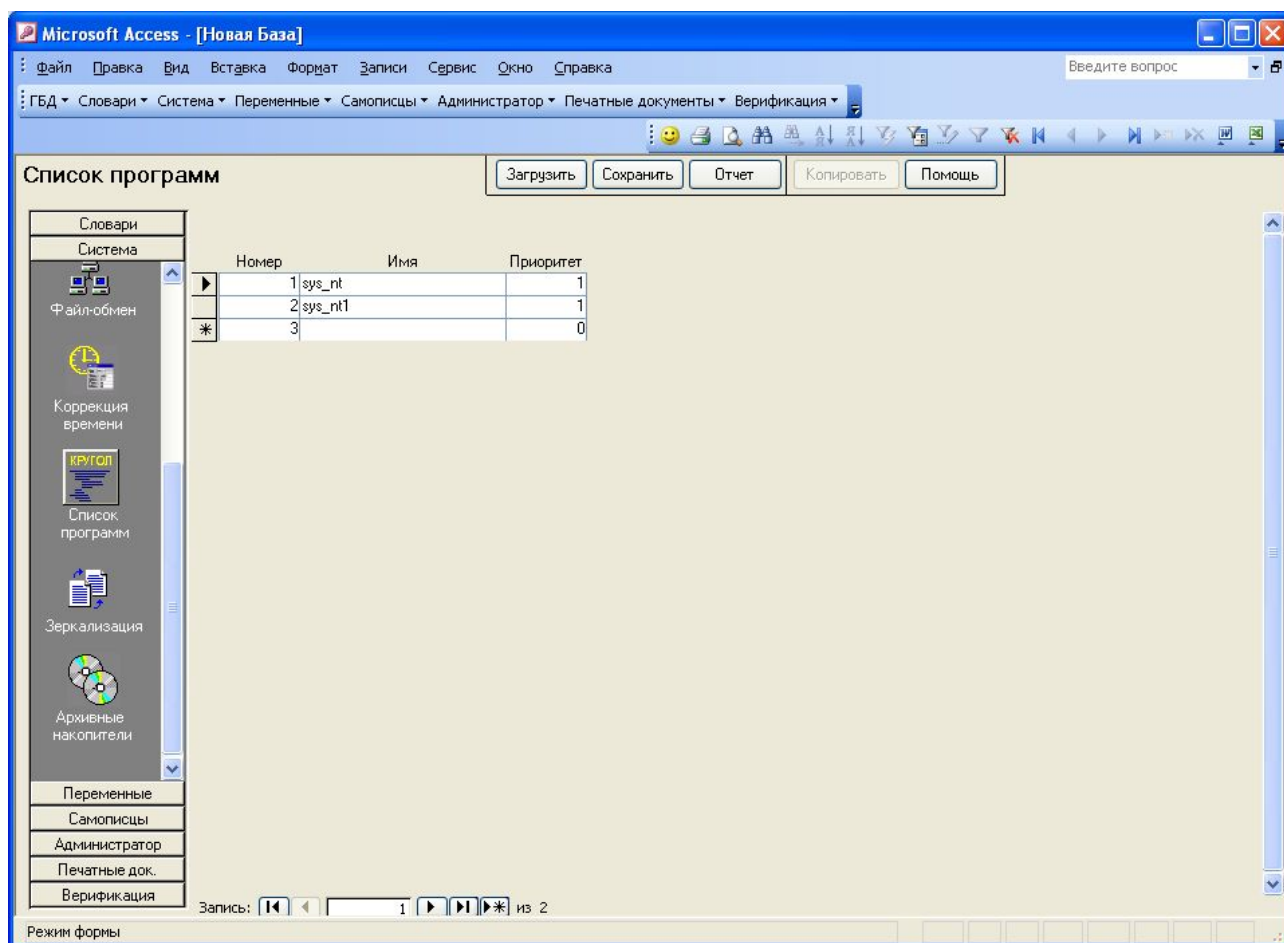
7.2 Среда исполнения станции оператора

Для выполнения программ Пользователя (ПрП) в Среде исполнения станции оператора необходимо:

- Разместить исполняемые программы Пользователя (файлы с расширением ***.out**) в поддиректории, из которой происходит загрузка файлов базы данных станции оператора
- Сформировать список исполняемых программ в базе данных станции оператора.

Формирование списка осуществляется в Генераторе базы данных (описание приведено в книге «Модульная интегрированная SCADA КРУГ-2000. Среда разработки. Генератор базы данных» в разделе «Генератор базы данных\Система\Список программ на языке КРУГОЛ»).

Окно конфигурирования списка программ показано на рисунке 7.2.1



7.2.1 – Формирование списка программ Пользователя

ВНИМАНИЕ!!!

В Среде исполнения программы Пользователя выполняются циклически. Цикл включает выполнение программ в той последовательности, в которой они описаны в Списке программ Генератора базы данных (например, список на рисунке 7.2.1 определяет следующую последовательность выполнения: цикл 1 – sys_nt, sys_nt1; цикл 2 – sys_nt, sys_nt1 и так далее).

