

ГЛАВА

4

Глава 4 Программа 'Разработчик'

4.1 Новые возможности

4.1.1 май 2007г.

Конфигурационные переменные разных типов.

Глобальные переменные следующих типов:

- простых (целые числа, числа с плавающей точкой, строки)
- структуры (группы переменных простых типов)
- массивы, одномерные простых типов или структур простых типов, двумерные простых типов

то есть типов, представимых в виде таблицы могут быть объявлены конфигурационными.

Установка признака "Конфигурационная" происходит в диалоге [свойств переменной](#).

Каждый раз при построении компонента происходит проверка конфигурации на изменение. Если формат или количество конфигурационных таблиц изменилось, то выдается предупреждение об увеличении версии компонента.

Полную информацию о конфигурации компонента можно посмотреть и поправить в диалоге ["Конфигурация компонента"](#) из меню Проект/Конфигурация.

4.2 Введение

ПО "SyTrack-TOOL" DEVELOP (инструмент **"Разработчик"**), обладая интуитивно понятным графическим интерфейсом, предоставляет возможность создавать прикладные и пользовательские алгоритмы и вычисления для контроллеров как профессиональным программистам, так и пользователям, не имеющим высокого уровня подготовки в области программирования.

Программа поддерживает язык FDB стандарта МЭК 61131 - язык функциональных блоковых диаграмм.



В **ПО "SyTrack-TOOL" DEVELOP** предусмотрено лицензирование по количеству вновь создаваемых пользователем проектов:

- до 10 проектов
 - от 10 проектов
-

4.3 Назначение, состав и возможности

Программа **"Разработчик"** является средой разработки прикладных алгоритмов для контроллеров Decont-182, контроллеров Decont-A9 и виртуальных контроллеров [WinDecont](#), работающих в ОС Windows.

"Разработчик" дает возможность создавать и редактировать собственные прикладные алгоритмы. Созданный алгоритм можно загрузить в контроллер с помощью программы "Конфигуратор".

ПО "SyTrack-PLC" DEV (программа **"Исполнение прикладных алгоритмов: средства исполнения алгоритмов, созданных в программе «Разработчик»"**) позволяет выполнять эти алгоритмы в контроллере.

Принципиальная схема работы алгоритма в контроллере заключается в тактовом выполнении некоторого блока, называемого главным. В процессе работы он считывает значения одних элементов глобальных баз (дискретов, аналогов, счетчиков) и в соответствии с заложеной логикой формирует и записывает новые значения в другие элементы.

Такт работы алгоритма задается в его конфигурации (из программы "Конфигуратор").

Среда программирования программы "Разработчик" поддерживает графический язык программирования и включает:

- графический редактор;
- встроенные библиотеки функциональных блоков (функций) и глобальных переменных (дискретов, аналогов, счетчиков);
- встроенный компилятор и средства для подготовки конфигурационных данных и библиотек для последующей загрузки в контроллер.

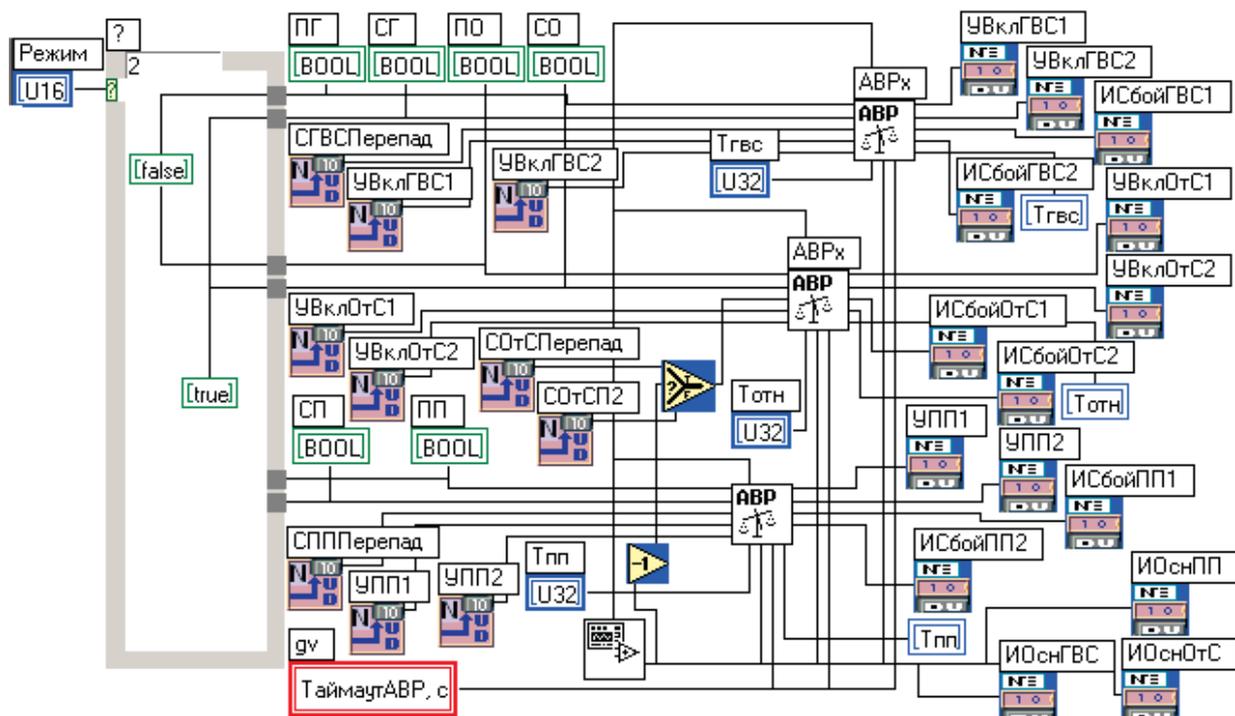
Функциональный блок (функция) – это совокупность переменных и правила преобразования входных переменных в выходные. Блоки, от элементарных блоков операций над данными до блоков, определенных пользователем, образуют алгоритм. Пользователь сам может определять новые необходимые ему функциональные блоки. Один блок может вызывать другие блоки. При написании объемных алгоритмов обычно применяется разбиение на блоки. Блок, который вызывает другие блоки, но сам не является вызываемым, называется **главным** блоком. Блоки сохраняются на диске в виде отдельных файлов с расширением "blk".

Имеется набор готовых блоков (см. главу "[Описание дополнительных блоков](#)"), например, PID- регулятор и другие.

Среда программирования гарантирует создание надежного кода программы, который не будет вызывать критических сбоев в работе контроллера Деконт. Ошибки в логике влияют только на правильность работы алгоритма, реализуемого контроллером, но не вызывают "подвисания" или иного сбоя контроллера.

Пользователь может написать в среде программы "Разработчик" алгоритм любой сложности. Следует учитывать системные возможности контроллера при написании очень объемного алгоритма.

Пример алгоритма в графическом виде:

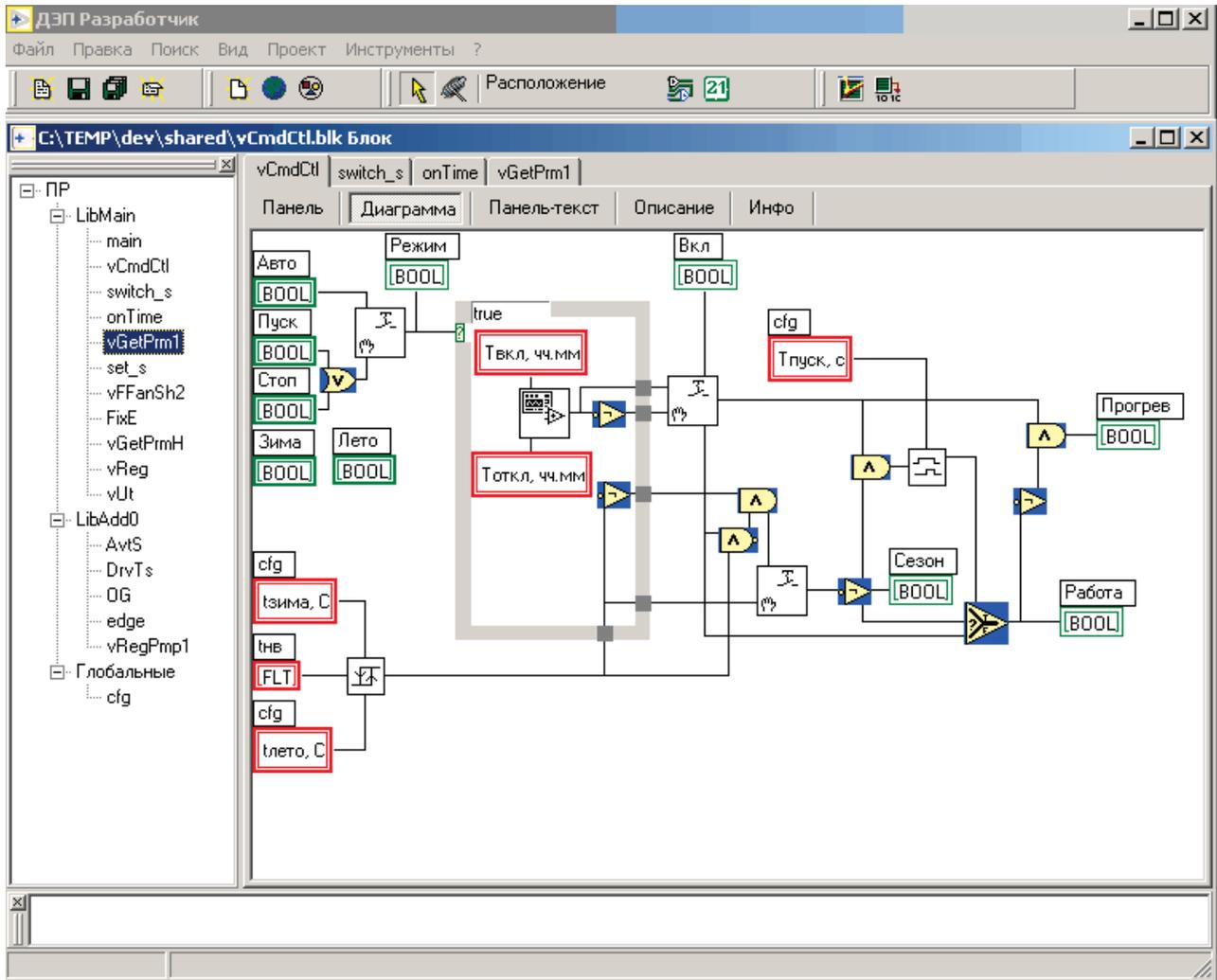


В ПО "SyTrack-PLC" DEV предусмотрено лицензирование по количеству выполняемых алгоритмов:

- до 5 алгоритмов
- от 5 алгоритмов

4.4 Описание пользовательских интерфейсов

Окна, с которыми работает пользователь, это: [основная панель](#) (сверху), [окно графического редактора](#) (состоящее из вкладок), ["Проводник"](#) - окно структуры проекта и [окно сообщений](#) (внизу). При открытии программы все окна открываются вместе:



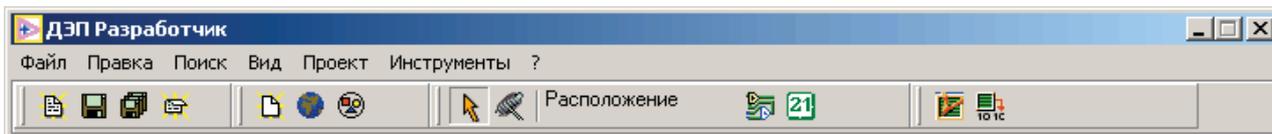
Меню "Вид" позволяет назначать вам те окна, которые вы хотите видеть на экране при работе.

4.4.1 Основная панель программы

На основной панели программы "Разработчик" находится главное меню программы.

В нижнем ряду отдельно вынесены кнопки часто выполняемых операций, а также кнопки смены режимов, кнопка выбора функций и кнопка выбора элементов управления .

Все подменю и кнопки панели расписаны подробно в главах "[Кнопки основной панели](#)" и "[Меню основной панели](#)". Меню <?> содержит справочную информацию.



4.4.1.1 Кнопки основной панели

-  **Расположение**

Переход в режим расположения, в котором блоки можно добавлять, перемещать и удалять, провода – перемещать и удалять.

-  **Соединение**

Переход в режим соединения, который используется для связи выходов одних блоков со входами других. При перемещении мыши над блоками их части начинают мерцать и появляются подсказки. Это **входные и выходные терминалы**, или просто **входы и выходы**. Для соединения терминалов нужно сделать щелчок левой кнопкой мыши на одном и, отпустив кнопку мыши (при этом появится провод), щелкнуть мышью на втором терминале. Для отмены проводки провода нужно щелкнуть мышью на исходном терминале.

-  **Проверить**

Происходит проверка блока на правильность соединений проводов. Проверяется возможность формирования последовательности выполнения блоков, соединение всех обязательных проводов, соответствие типов данных всех пар терминалов, соединенных проводами.

При нажатии кнопки **<Проверить>** в меню **<Проект \Проверить>** происходят те же действия, только применительно к проекту.

Перечень всех кнопок:



 -открыть...

 - сохранить

 -сохранить как...

 - закрыть

 - новый блок

 - новая глобальная переменная

 - новый тип

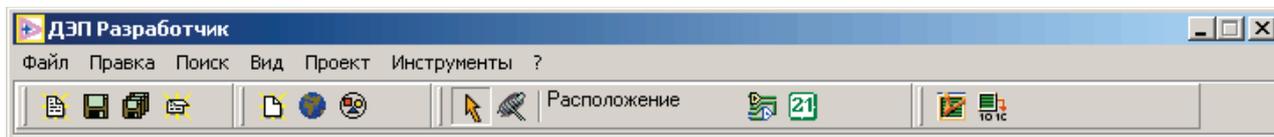
 - функции

 - элементы управления

 - построение.

4.4.1.2 Меню основной панели

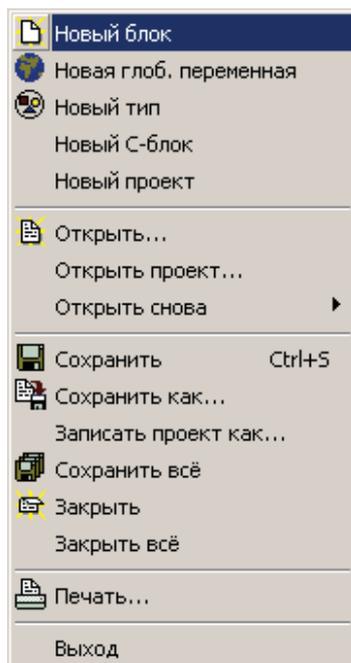
На основной панели имеются следующие меню:



- [Меню Файл](#)
- [Меню Вид](#)
- [Меню Правка](#)
- [Меню Поиск](#)
- [Меню Проект](#)
- [Меню Инструменты](#)

4.4.1.2.1 Меню Файл

Меню Файл



<Новый блок> – создается [файл нового блока](#).

- <Новая глоб. переменная> – создается [файл глобальных переменных](#).

<Новый тип> – создается файл для объявления [новых типов](#).

- <Новый С-блок> – создается файл нового блока, который реализуется на языке Си.

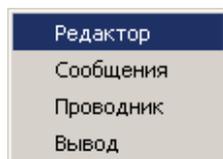
- <Новый проект> – создается новый [проект](#).

- <Открыть> – открывается файл (может быть одного из трех типов – файл блока, файл глобальных переменных, файл объявлений типов).

- **<Открыть проект>** – открывается проект.
- **<Открыть снова>** – открыть один из последних используемых проектов.
- **<Сохранить>** – сохранить файл (может быть одного из трех типов – файл блока, файл глобальных переменных, файл объявлений типов).
- **<Сохранить как>** - сохраняет блок с новым именем (может быть одного из трех типов – файл блока, файл глобальных переменных, файл объявлений типов).
- **<Записать проект как>** – сохраняется проект с новым именем.
- **<Сохранить все>** – сохраняется как проект, так и все открытые файлы.
- **<Закрыть>** – закрывается файл одного из трех типов.
- **<Закрыть всё>** - закрываются все открытые файлы.
- **<Печать>** – вызывается диалог для печати активного файла (диаграммы и панели). В диалоговом окне можно выбрать расположение листа (вертикально или горизонтально), масштаб, назначить печать заголовка (представляет собой полный путь к файлу, выдаваемому на печать), а также выбрать, что вы хотите печатать – панель, диаграмму или обе на одном листе (поделенном пополам).
- **<Выход>** – выход из программы.

4.4.1.2.2 Меню Вид и меню Правка

Меню Вид

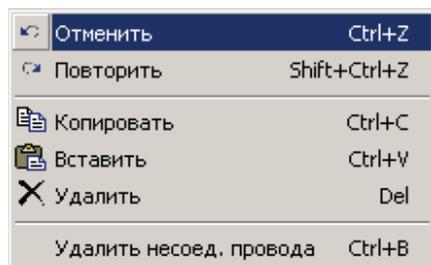


Позволяет выбирать вам те окна, которые вы хотите видеть на экране при работе.

При открытии программы по умолчанию открывается три окна - окно "Редактор", окно "Сообщения" и окно "Проводник" (см. главу "[Окна графического редактора](#)"). Но вы имеете возможность открыть их отдельно друг от друга.

- **<Редактор>** – открыть окно редактора
- **<Сообщения>** – открыть окно сообщений
- **<Проводник>** – открыть окно проводника
- **<Вывод>** – открывается окно вывода компилятора.

Меню Правка



- **<Отменить>** - производится отмена перемещения блоков или проводов
- **<Повторить>**- производится повтор перемещения блоков или проводов
- **<Копировать>** - производится копирование в буфер выделенных элементов диаграммы или панели
- **<Вставить>**- вставка выделенных элементов диаграммы или панели из буфера
- **<Удалить>** – удаление выделенных элементов диаграммы или панели
- **<Удалить несоединенные провода>** – производится анализ проводов в диаграмме; те несоединенные провода, которые вы забыли удалить, удаляются.

Для каждой диаграммы заведен буфер на 10 позиций. В каждой позиции сохраняются координаты всех визуальных элементов диаграммы. С помощью этого буфера реализуются операции "Отмена" и "Повтор".

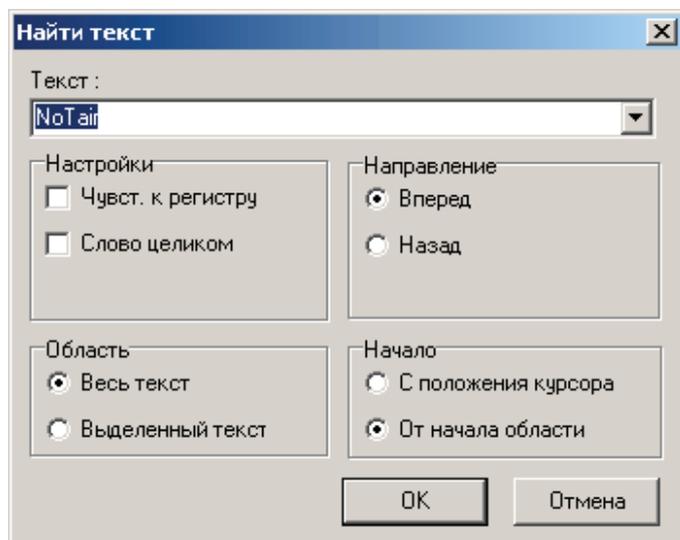
4.4.1.2.3 Меню Поиск

Меню Поиск

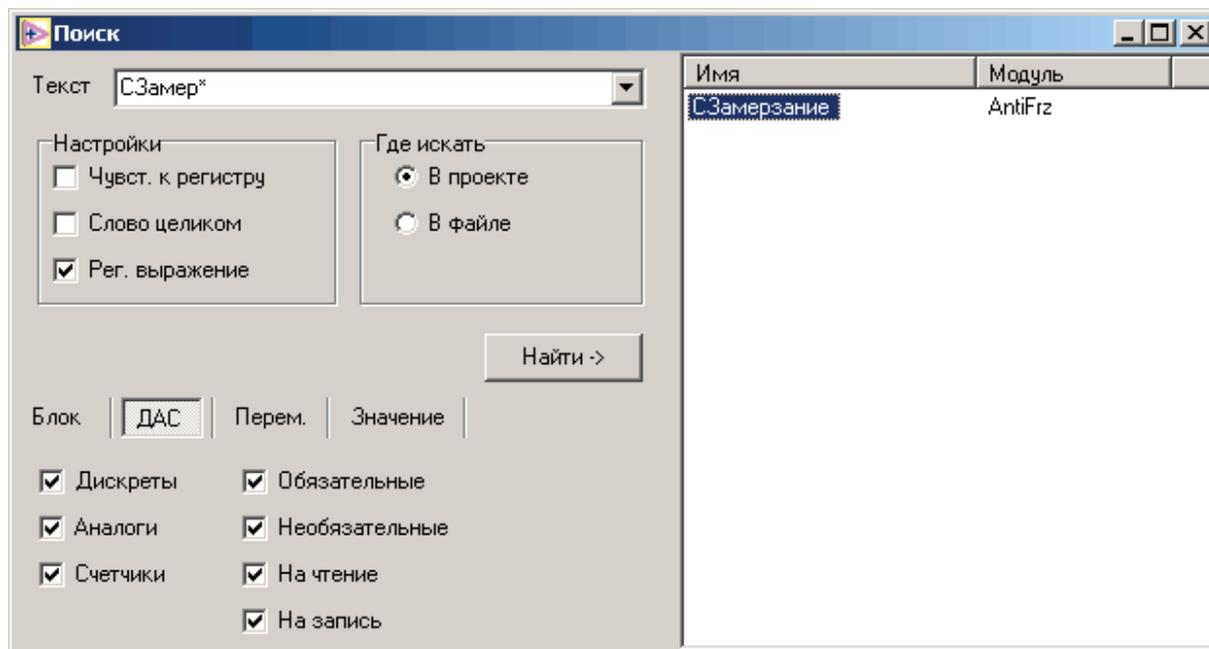
Найти...	Ctrl+F
Искать дальше	F3
Заменить	Ctrl+R

- **<Найти...>**

При выборе данного пункта меню если открыто и активно окно, содержащее С-текст, то появляется стандартный диалог поиска текста.



Если активно любое другое окно (не С-текст), то открывается специальный диалог поиска.

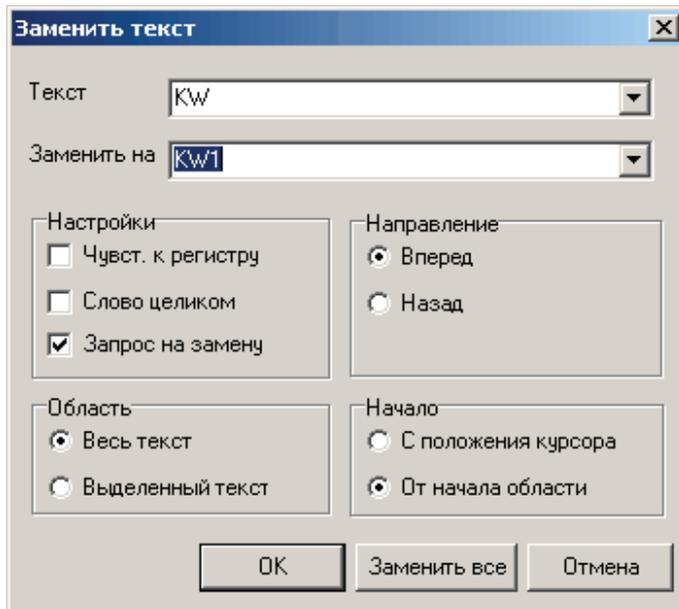


Искать строку можно в активном файле или в проекте. Поиск разделен на следующие варианты:

- 1 **Блок.** Поиск мест использования блока, имя которого совпадает со строкой поиска.
- 2 **ДАС** (дискретные, аналоги, счетчики). Поиск блоков работы с элементами баз, номера которых задаются по именам через конфигурацию.
- 3 **Перем.** Поиск переменных, метки которых совпадают со строкой поиска.
- 4 **Значение** Поиск значений переменных и констант.

При двойном щелчке по элементу в окне результата поиска в редакторе открывается соответствующий модуль, и в нем выделяется найденный элемент.

- **<Искать дальше>** - продолжение поиска в С-тексте.
- **<Заменить>** - замена в С-тексте. При выборе этого пункта меню появляется следующий диалог:



Заменить текст

Текст: KW

Заменить на: KW1

Настройки

Чувств. к регистру

Слово целиком

Запрос на замену

Направление

Вперед

Назад

Область

Весь текст

Выделенный текст

Начало

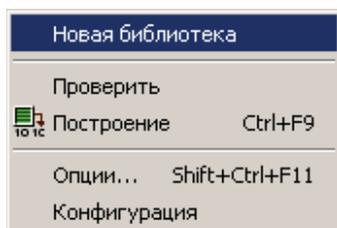
С положения курсора

От начала области

OK Заменить все Отмена

4.4.1.2.4 Меню Проект

Меню Проект



- **<Новая библиотека>** – добавить к проекту новую [библиотеку](#).

-  **<Проверить>** – происходит проверка проекта на правильность соединений проводов. Проверяется возможность формирования последовательности выполнения блоков, соединение всех обязательных кодов, соответствие типов данных всех пар терминалов, соединенных проводами. При нажатии кнопки **<Проверить>** на главной панели происходят те же действия, только применительно к блоку. Но в отличие от "Проверить" применительно к блоку проверка проекта включает в себя еще и проверку на то, указана ли [главная функция](#) (если не указана - указать).

-  **<Построение>** – происходит выполнение следующей последовательности действий:
 1. проверка проекта;
 2. регистрация компонента в базах программы "Конфигуратор";
 3. сохранение файлов;
 4. построение исходного кода на языке Си;
 5. построение файлов библиотек.

Можно отключить выполнение всех пунктов, кроме первого.

- **<Опции>** - вызвать диалог настроек проекта. Состоит из нескольких вкладок:
 - **Общие** - необходимо указать имя главной библиотеки (той, что содержит главную функцию) и имя главной функции и указывается каталог вывода проекта (каталог, где будут расположены промежуточные файлы построения).

Общие | Параметры | Описание | Дополн

Главные

Библиотека LibMain ...

Функция bmain ...

Каталог вывода

...

- **Параметры** - в этом окне указывается размер стека.

Общие | Параметры | Описание | Дополн

Размеры

Стек 2048 По умолчанию

Куча 0 По умолчанию

Переменные: глобальные 0 байт
 сохраняемые 0 байт

Номер компонента, под которым он будет зарегистрирован в "Конфигураторе" 34502

Приоритет Низкий (75)

Дополнительная библиотека инициализации

Поле "**Куча**" в настоящий момент не используется.

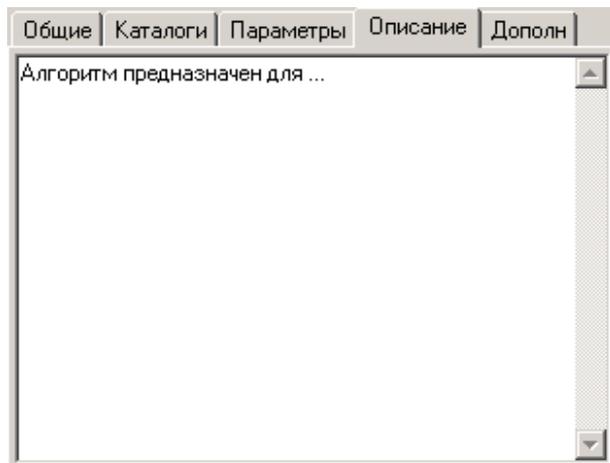
Кнопкой **<По умолчанию>** можно установить размеры по умолчанию.

Номер компонента возвращается из программы "Конфигуратор" во время регистрации компонента. Можно видеть также размер (в байтах), занимаемый глобальными и сохраняемыми переменными.

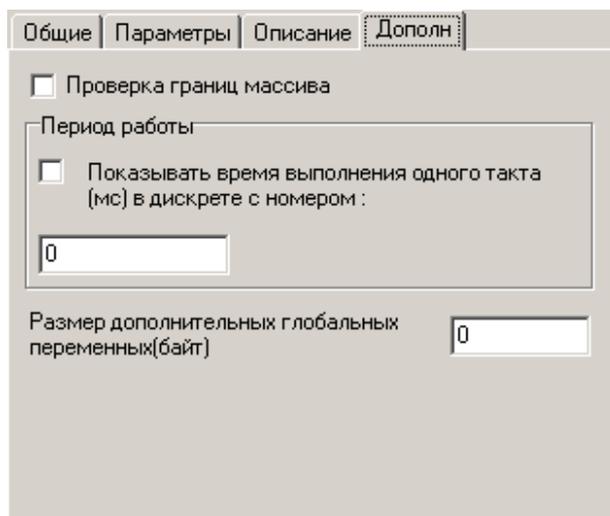
Приоритет определяет приоритет работы компонента. В поле можно выбрать один из predetermined priorities or set a number.

Флаг "**Дополнительная библиотека инициализации**" используется, когда библиотека инициализации LibIni превышает допустимый размер (только для платформы Decont 182). В этом случае установкой этого флага библиотека инициализации разбивается на две.

- **Описание** - текстовая информация, связанная с алгоритмом:



- **Дополнение** - дополнительные параметры:

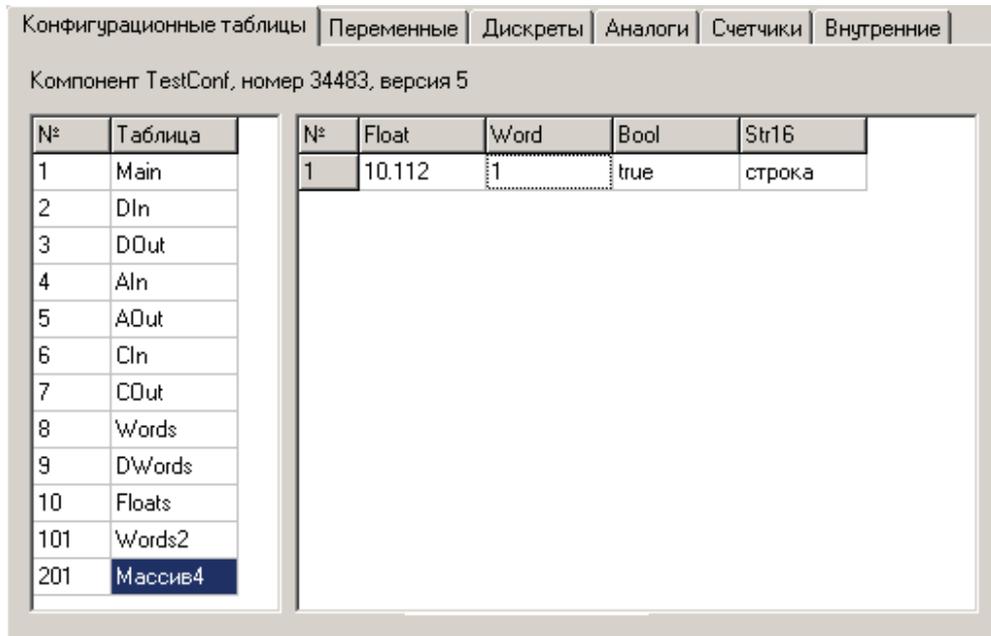


Проверка границ массива - при установке галочки происходит проверка границ массива. Если в процессе работы компонента происходит выход за границы массива то контроллер рестартует в минимальный режим и в журнале ошибок появляется сообщение "Выход за границы массива". Проверка выполняется только для "графических" блоков и не выполняется для блоков, реализация которых написана на языке С.

Период работы - если установить галочку и указать номер дискрета, тогда в дискрете с этим номером будет показано время выполнения одного такта алгоритма в миллисекундах.

Размер дополнительных глобальных переменных - указывается размер глобальных переменных, определенных на языке С во вкладках "Панель-С" файлов глобальных переменных.

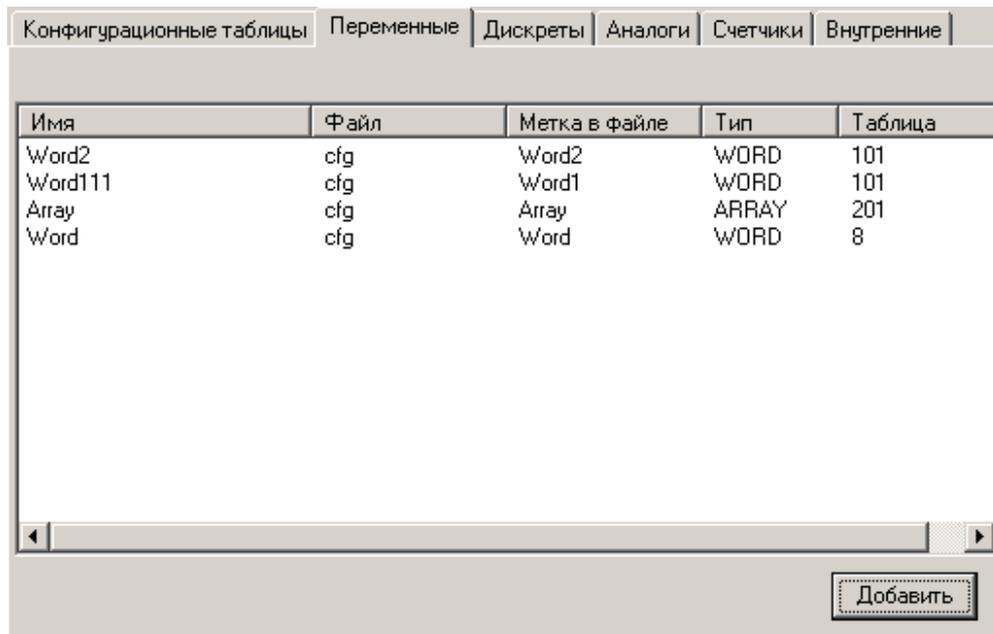
- **<Конфигурация>** компонента - содержит вкладки - конфигурационные таблицы, переменные, дискретные, аналоговые, счетчики и внутренние.
- **Вкладка "Конфигурационные таблицы"** - отображает название, номер и версию компонента.



В этой вкладке отображается конфигурация компонента, как она выглядит в программе "Конфигуратор". Здесь можно поменять названия таблиц, данные таблиц, порядок строк и столбцов, где это допустимо. Если данные введены неверно, то ячейка отображается красным цветом.

- Вкладка "Переменные" отображает конфигурационные переменные компонента (не забудьте, что проект считается компонентом, когда у него определена [главная функция](#)):

Можно добавлять переменные, удалять, изменять имена, которые будут представлять эти переменные в конфигурационных таблицах.



Если две переменные имеют одинаковое имя, то они выделяются красным цветом.

Дискретные, аналогии и счетчики перечислены в соответствующих закладках. Можно менять значения конфигурационных параметров. Можно (правой кнопкой мыши) менять параметры местами. Столбец "0" указывает на то, является ли элемент обязательным (стоит "1").

- **Дискретные** - в данной вкладке перечисляются используемые в алгоритме входные и выходные дискретные, номера которых можно менять из программы "Конфигуратор".

Входные			Выходные		
Имя	Номер	0	Имя	Номер	0
КМест	1	1	ИРежим	7	
КДист	2		УКлапОтС	8	
КАвто	3		УКлапПП	9	
СРБО	4		ИЗЛ	10	
КОРБО	5		УВклГВС1	11	
КЗРБО	6		УВклГВС2	12	

рис. Конфигурация компонента, вкладка "Дискретные"

- **Аналоги** - в данной вкладке перечисляются используемые в алгоритме входные и выходные аналогии, номера которых можно менять из программы "Конфигуратор".

Входные			Выходные		
Имя	Номер	0	Имя	Номер	0
СТгвс	1		ИГВСУст	11	
КГВСУст	2		ИУстОВ	12	
СТпв	3		ИУстПВ	13	
СТов	4		ИППУст	14	1
СТнв	5				
КПВУст	6				

рис. Конфигурация компонента, вкладка "Аналоги"

- **Счетчики** - в данной вкладке перечисляются используемые в алгоритме входные и выходные счетчики, номера которых можно менять из программы "Конфигуратор".

Входные			Выходные		
Имя	Номер	0	Имя	Номер	0
C1	1		1-й тариф	0	1
C2	2		2-й тариф	0	1

рис. Конфигурация компонента, вкладка "Счетчики"

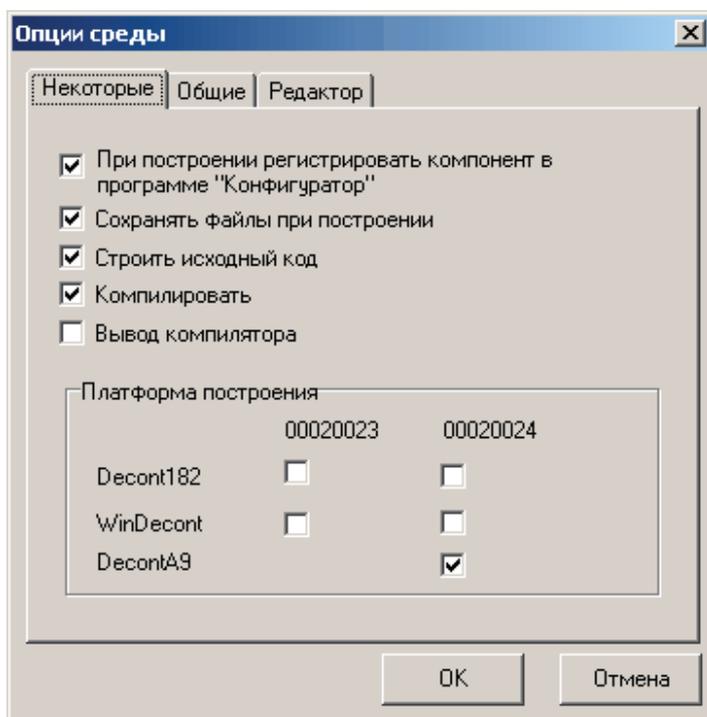
- **Внутренние** - отображаются все внутренние (не сохраняемые) и сохраняемые переменные проекта.

Модуль	Метка	Тип
mainx	Время	ARRAY
ABPx	Сбой1	WORD
ABPx	Сбой2	WORD
base	OldStamp	FLOAT
base	OldMain	BYTEBOOL

4.4.1.2.5 Меню Инструменты

Меню Инструменты

- **<Опции сред ы...>** - содержит в себе три вкладки:
 - Вкладка "**Некоторые**" позволяет управлять процессом построения. В зависимости от того, где вы поставите галочку, будут выполняться (не выполняться) следующие команды:

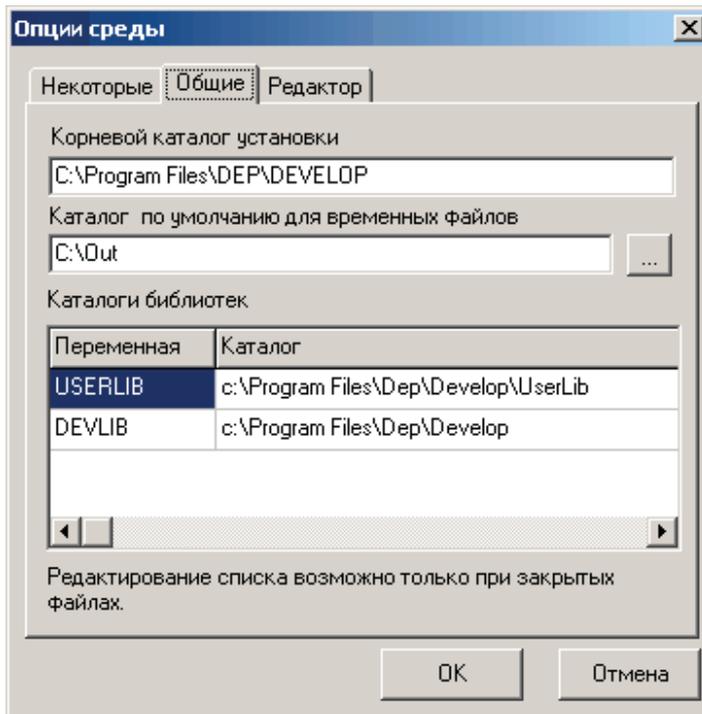


- Регистрировать ли компонент в программе "Конфигуратор" при построении;
- Сохранять ли файлы при построении;
- Строить исходный код - генерировать исходный текст на языке "Си" из диаграмм блок;
- Компилировать - компилировать и создавать программный код;
- Вывод компилятора - Заполнять окно вывода сообщениями от компилятора;

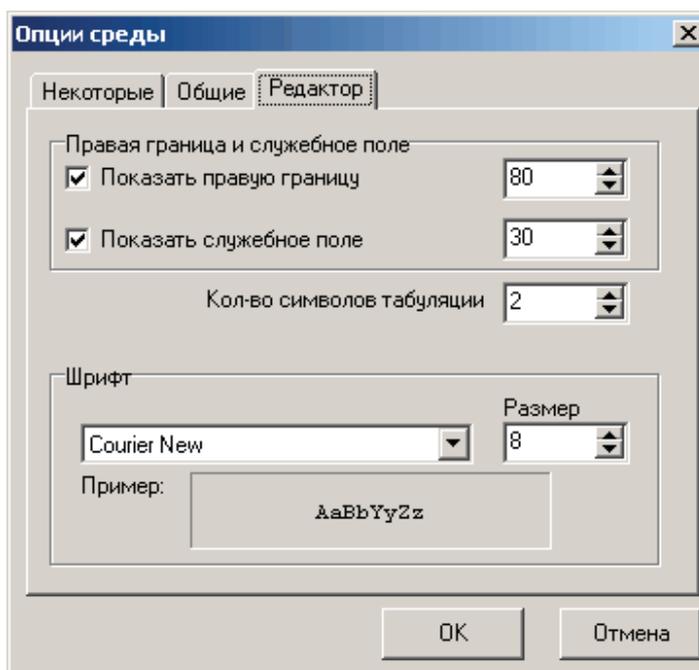
- Decont182 – построение компонента для работы в контроллере Decont-182 (указать версию);
- WinDecont - построение компонента для работы в контроллере "WinDecont" (указать версию).
- DecontA9 - построение компонента для работы в контроллере Decont-A9(доступна только версия 00020024)

Если выбраны оба построения, то полное построение проходит в два этапа.

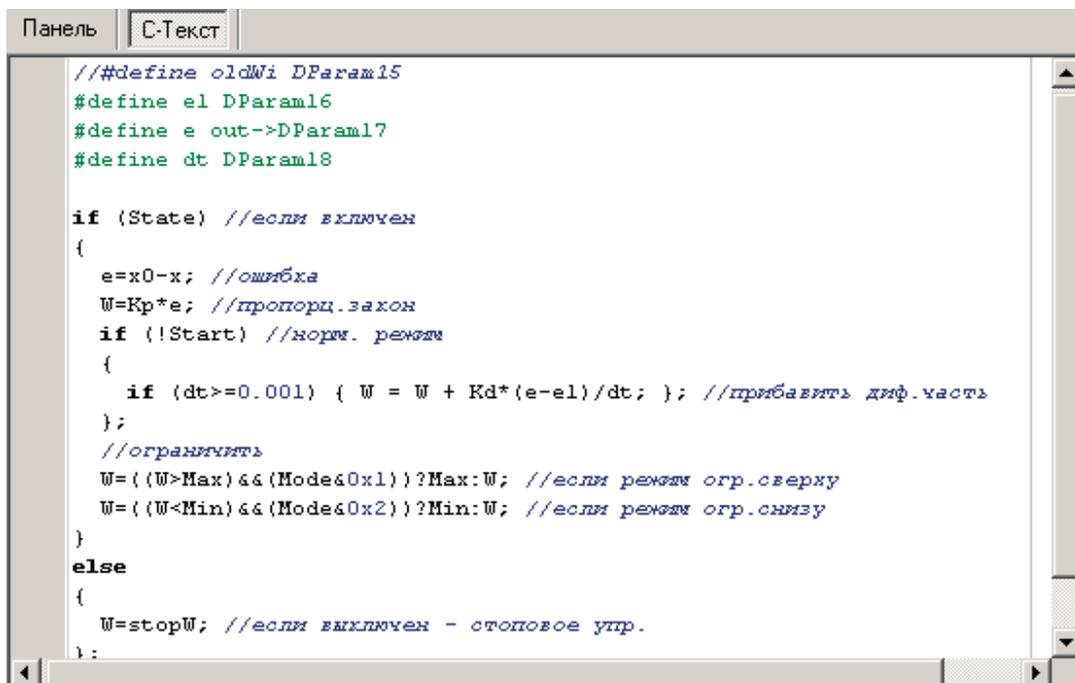
- Вкладка "**Общие**" содержит общие сведения о программе:



- Корневой каталог – корневой каталог установки программы;
 - Каталог по умолчанию для временных файлов ставится C:\Out, но можно его изменить.
 - Каталоги библиотек используются при сохранении блоков, так что если путь к файлу блока содержит каталог библиотек, то вместо каталога библиотек в пути к блоку сохраняется соответствующая переменная.
- Вкладка "Редактор" работает с редактором C-текста:



Можно редактировать правую границу и служебное поле, а также менять шрифт текста:

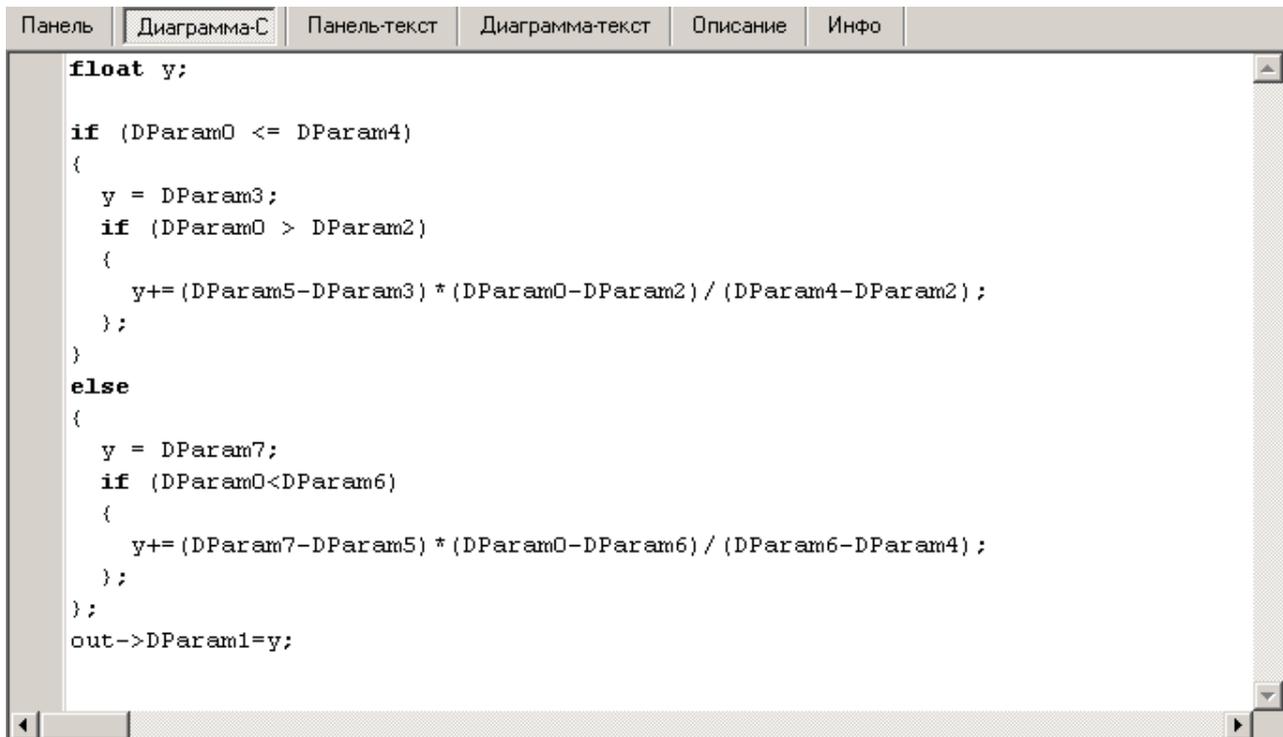


```
Панель C-Текст
//#define oldWi DParam15
#define e1 DParam16
#define e out->DParam17
#define dt DParam18

if (State) //если включен
{
    e=x0-x; //ошибка
    W=Kp*e; //пропорц.закон
    if (!Start) //норм. режим
    {
        if (dt>=0.001) { W = W + Kd*(e-e1)/dt; }; //прибавить диф.часть
    };
    //ограничить
    W=(W>Max)&&(Mode&0x1)?Max:W; //если режим огр.сверху
    W=(W<Min)&&(Mode&0x2)?Min:W; //если режим огр.снизу
}
else
{
    W=stopW; //если выключен - стоповое упр.
};
```

4.4.2 Окна редактора

Редактор работает с несколькими типами окон-вкладок. Рассмотрим например окна С-Блока.



```
float y;

if (DParam0 <= DParam4)
{
    y = DParam3;
    if (DParam0 > DParam2)
    {
        y+= (DParam5-DParam3) * (DParam0-DParam2) / (DParam4-DParam2) ;
    };
}
else
{
    y = DParam7;
    if (DParam0<DParam6)
    {
        y+= (DParam7-DParam5) * (DParam0-DParam6) / (DParam6-DParam4) ;
    };
};
out->DParam1=y;
```

- **"Панель"** - окно, где размещаются переменные блока (входные, выходные, внутренние и тд.).
- **"Диаграмма-С"** - реализация блока на языке "С".
- **"Панель-текст"** - текст на языке "С", добавляемый в ".h" файл
- **"Диаграмма-текст"** - текст на языке "С", добавляемый в ".c" файл
- **"Описание"** - текстовое описание блока
- **"Инфо"** - дополнительная информация о блоке.

"Панель", "Панель-С", "Описание", "Инфо" используются для всех типов файлов, а **диаграммы** - только для функциональных блоков. У обычного блока будет отсутствовать окно "Диаграмма-текст", а вместо "Диаграмма-С" будет "Диаграмма" - графическая реализация блока. У файла глобальных переменных будет отсутствовать "Диаграмма".

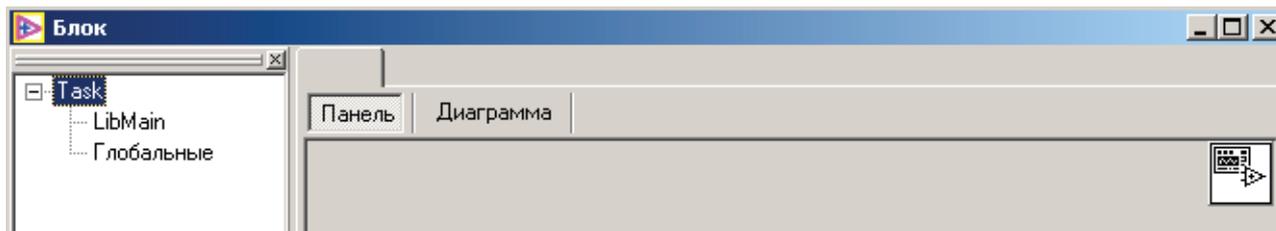
Программа "Разработчик" использует четыре типа файлов – файлы **проекта**, файлы **блоков** (блоки описаны в главе ['Назначение, состав и возможности программы'](#)), файлы **глобальных переменных** (расширение "glb") и файлы **типов переменных** (расширение "tdf").

Когда создается блок, создаются сразу пустое окно панели и пустое окно диаграммы. На панели блока задаются переменные этого блока, диаграмма отражает логику выполнения этого блока, используя переменные, задаваемые на панели блока и глобальные переменные, задаваемые на панели глобальных переменных.

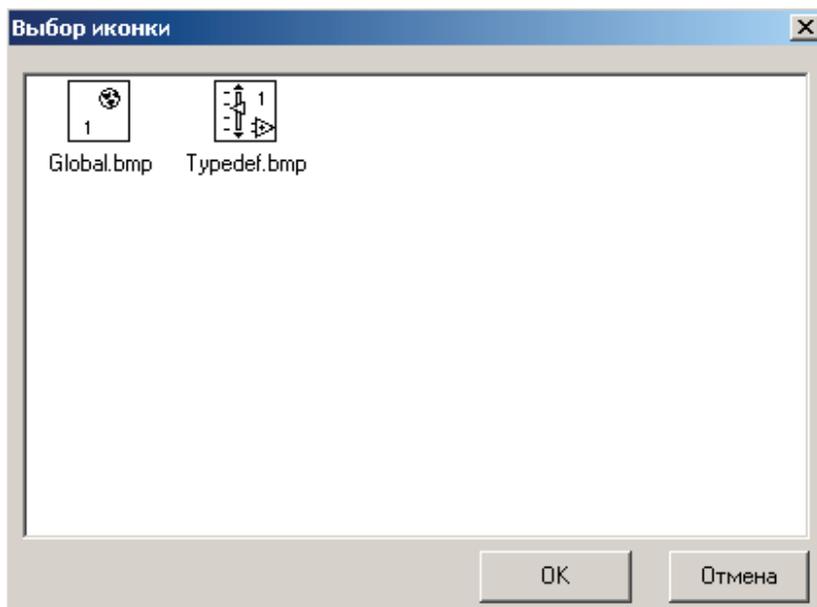
4.4.2.1 Панели

Панели, в зависимости от типа файла, для которого они используются, бывают 3-х видов:

Панель блока.  Создается нажатием кнопки <Новый блок> на основной панели. Содержит набор входных, выходных и др. переменных для блока. Имеет **серый** фон:



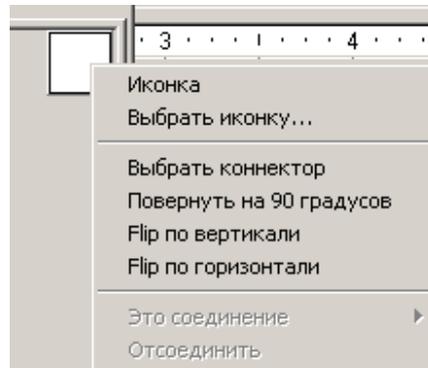
В правом верхнем углу находится **значок блока**. Его можно представить в виде абстрактной иконки (нажать правую кнопку мыши на значке, **<Выбрать иконку>**) или же как коннектор (значок, отображающий количество входных и выходных переменных и их расположение относительно друг друга):



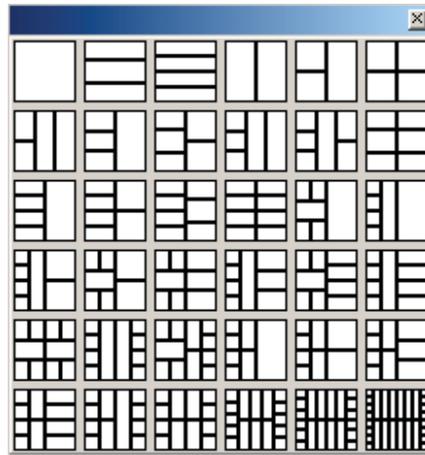
 Все переменные, принадлежащие этому блоку, создаются на этой панели. Для того, чтобы **создать переменную**, нужно выбрать ее, нажав **кнопку <Элементы управления>**.

Левой кнопкой мыши выбирается элемент, затем кнопка отпускается и также нажатием левой кнопки мыши закрепляется на панели. Появляется терминал переменной, над ним – метка. Можно, нажав правую кнопку мыши, задать свойства этой переменной. Подробнее о переменных можно прочитать в соответствующей главе. Кнопкой **<Показать метку>** можно показать или, наоборот, убрать метку.

Для связывания переменных с терминалами блока необходимо **выбрать коннектор**, который показывает, как расположены терминалы данного блока. Для этого необходимо нажать правой кнопкой мыши на иконку блока в правом верхнем углу и выбрать меню <Выбрать коннектор>:



Появится набор предлагаемых коннекторов:



Исходя из количества переменных и задачи, необходимо выбрать нужный коннектор. Для связи элемента панели с одним из входов/ выходов необходимо в режиме соединения нажать левой кнопкой мыши на выбранный элемент (он выделится более жирными границами), отпустить кнопку, затем нажать на выбранный терминал (он выделится черным цветом). Таким образом, устанавливается соответствие, которое соблюдается при написании алгоритма.

По умолчанию переменная является входной. Изменить ее на выходную можно из контекстного меню (правой кнопкой мыши нажать на переменную, выбрать **<Изменить на выход >**).

В свойствах переменной нажатой кнопкой указывается, входная она или выходная, только после того, как вы свяжете ее с терминалом.

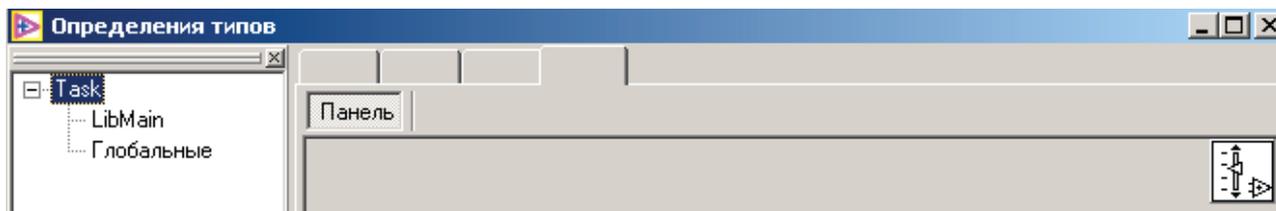
Перейдя на диаграмму блока, видно, что все созданные переменные блока отразились на диаграмме с теми же метками и даже на тех же местах. В дальнейшем, когда вы будете перемещать переменные на диаграмме, идентификация осуществляется по метке.

Панель – Глобальные переменные. Создается нажатием кнопки  ("новая глобальная переменная") на основной панели:



Имеет серый фон. Содержит **набор переменных для файла глобальных переменных**. Переменные создаются так же, как и на панели блоков, но они являются глобальными, доступными из любого блока проекта. Значок в правом верхнем углу такой панели играет только декоративную роль, его можно сменить на тот, что больше нравится.

Панель определения типов. Создается нажатием кнопки  ("новый тип") на основной панели. Содержит набор определений типов переменных для файла определения типов. Имеет серый фон.



Если вы используете локальную или глобальную переменную на диаграмме не один раз, воспользуйтесь кнопкой **<Функции\Инструкции\Глобальная(Локальная) переменная>**.

Тип переменной можно задать на панели, на которой она создается. Но для сложных типов переменных применяется панель типов. Панель позволяет использовать созданный вами тип переменных как шаблон для разных переменных в разных диаграммах.

Чтобы вставить эту переменную в диаграмму, необходимо нажать кнопку **<Элементы управления \Выбор из файла>**, указав в появившемся диалоге имя панели и имя нужной вам переменной.

Если изменить тип какой-то переменной на этой панели, то на диаграммах, которые используют эту переменную, появится сигнализация о необходимости обновления диаграммы (кнопкой **<Обновить>** в контекстном меню этой структуры или массива). Сигнализация выражается в изменении границы структуры или массива – белая граница станет серой.

4.4.2.2 Диаграммы

Диаграмма содержит реализацию выполнения блока.

Диаграммы имеют белый фон.

Основными элементами диаграммы являются: переменные, константы, блоки (содержащие терминалы) и провода, соединяющие их. Провода имеют различный цвет в зависимости от типа данных, передающихся по ним.

Пример диаграммы:

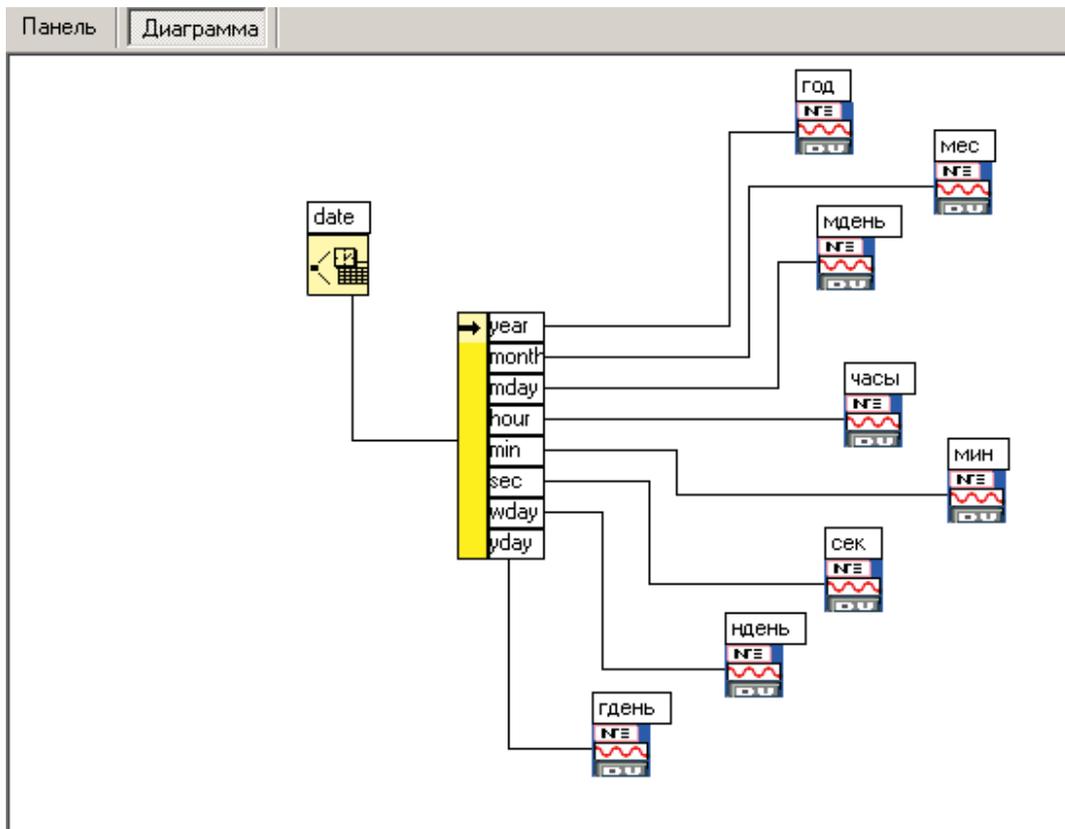


Рис. Диаграмма

Порядок выполнения диаграммы.

Выполнение диаграммы - это выполнение блоков в составе диаграммы. Выполнение блока - это выполнение диаграммы, описывающей реализацию блока. Порядок выполнения блоков диаграммы определяется связями между выходными терминалами одних блоков и входными терминалами других. Для выполнения некоторого блока диаграммы необходимо, чтобы блоки, выходные терминалы которых связаны с его входными терминалами, были выполнены.

Порядок выполнения блоков, готовых к выполнению, но не имеющих связей между их терминалами, неопределен.

Для **добавления новых элементов** в диаграмму используется меню **<Функции>**.

Для добавления блока в диаграмму нажмите на кнопку **<Функции>**, выберите нужную функцию, нажмите на нее левой кнопкой, отпустите, затем нажмите левой кнопкой на то место в диаграмме, в которое вы хотите поместить элемент:

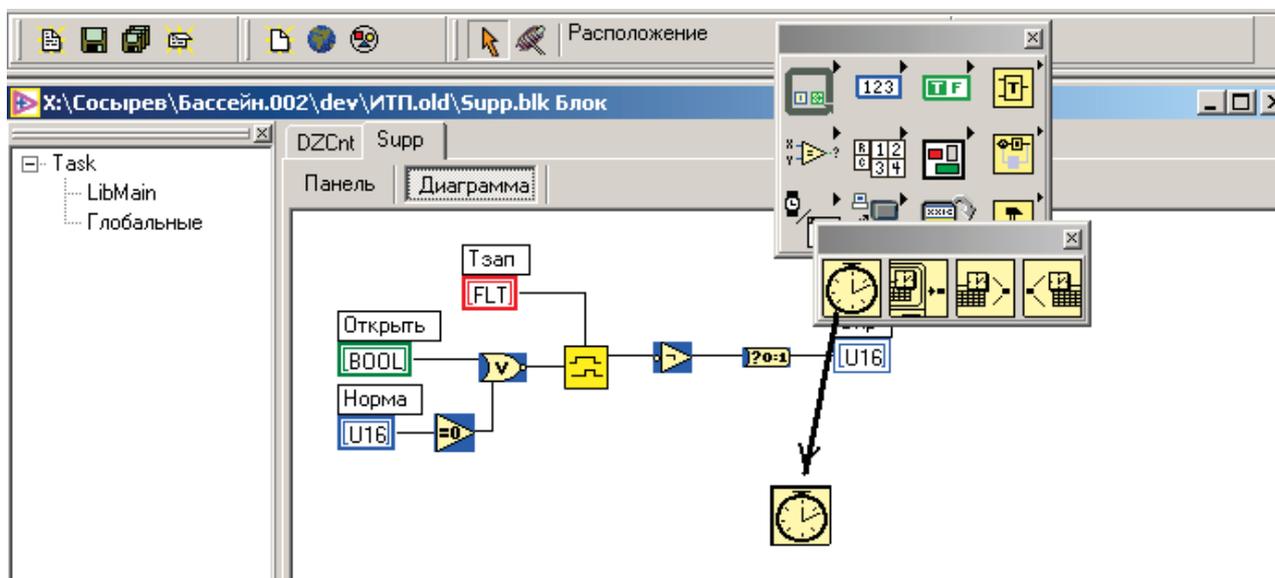


Рис. Добавление блока в диаграмму.

При работе с редактором существуют **три режима работы**:

- 
Расположение. В режим можно войти, нажав кнопку **<Расположение>** на основной панели. Выбранный режим отражается в этой же строке панели. В режиме "Расположение" блоки можно добавлять, перемещать и удалять. Провода можно перемещать и удалять. Провода удаляются аналогично блокам: выделением мышью и нажатием на кнопку ****.
- 
Соединение. В режим можно войти, нажав кнопку **<Соединение>** на основной панели. Данный режим используется для связи выходов одних блоков со входами других. При перемещении мыши над блоками их части начинают мерцать, и появляются подсказки. Это **входные и выходные терминалы**, или просто **входы** и **выходы**:

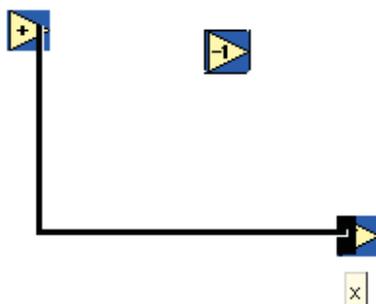


рис. Соединение терминалов

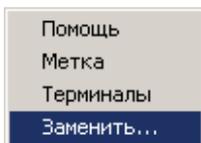
Для соединения терминалов нужно сделать щелчок левой кнопкой мыши на одном и, отпустив кнопку мыши (при этом появится провод), щелкнуть мышью на втором терминале (см. рис. Соединение терминалов.). Для отмены проводки

провода нужно щелкнуть мышью на исходном терминале. Менять численные значения и метки элементов можно, щелкнув на соответствующей метке в любом режиме.

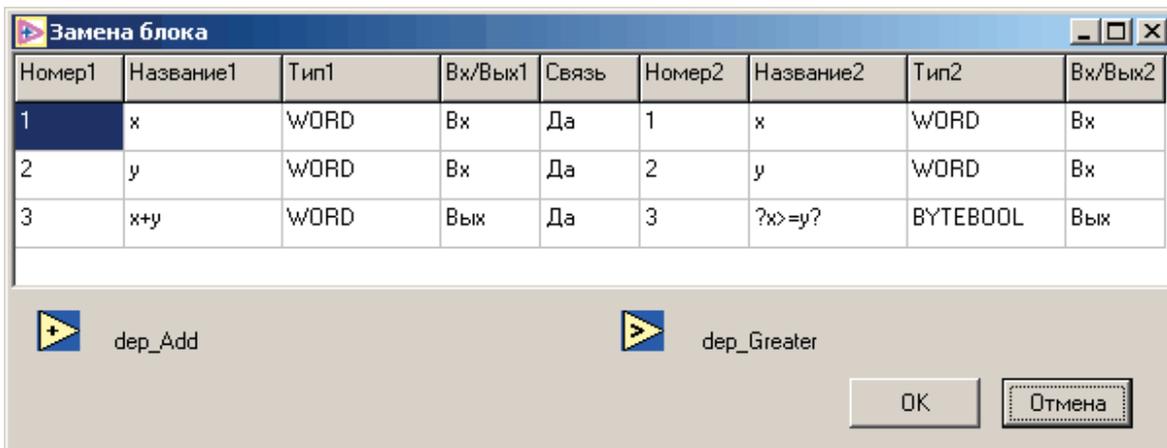
- **Редактирование.** Вход в этот режим осуществляется нажатием левой кнопкой мыши на названии метки блока. В этом режиме можно менять численные значения и метки элементов. При нажатии мышью на константе появится окно редактирования, в котором нужно ввести нужное значение.

4.4.2.1 Замена блока

Контекстное меню блока содержит пункт "Заменить...":



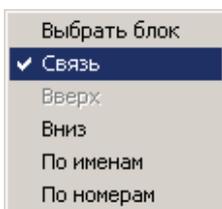
При выборе этого пункта вызывается диалог замены:



В этом диалоге предлагается заменить существующий на диаграмме блок сложения (dep_Add) на блок сравнения (dep_Greater), так чтобы при этом сохранились связи проводов в терминалах блока таким образом, как это указано в таблице диалога.

Названия с индексом 1 соответствуют старому блоку, названия с индексом 2 - новому.

С помощью контекстного



меню над ячейкой таблицы можно

- выбрать блок, на который заменить существующий,
- указать, сохранить ли связь с проводом
- установить нужный порядок связей, то есть, какой терминал нового блока соответствует какому терминалу старого.

При повторном открытии диалога, ранее выбранный блок и расположение его терминалов в таблице сохраняется.

4.4.2.3 Инфо

Вкладка "Инфо" показывает дополнительную информацию при разработке блока.

Имя переменной	Название	Тип данных	Вх/Вых	Значение	Сохранение	Псевдоним
DPParam2	Закреть	BYTEBOOL	Вх	false		
DPParam3	Пуск	BYTEBOOL	Вх	false		
DPParam4	Инверсия	BYTEBOOL	Вх	false		
(*DPParam5)	N№N№	STRUCT	Вх			
internal->DPParam6	вРежим	BYTE	Лок	5	Внутренняя	
internal->DPParam7	су	FLOAT	Лок	0	Внутренняя	
DPParam8	у	BYTEBOOL	Лок	false		
DPParam9	+	WORD	Лок	0		

Проверка стека

Используемые блоки					Файлы глобальных переменных	
Имя	Функция	Внутрен...	Сохран...	РПЗУ	Имя	Кол-во
DBlock0	dep_BundleByName	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	pidv	7
DBlock1	dep_UnbundleByNa	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
DBlock3	dep_PIDinc	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
DBlock22	dep_Select	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
DBlock23	dep_Select	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
DBlock4	dep_PIDpos	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
DBlock18	dep_UnbundleByNa	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
DBlock19	dep_Add	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
DBlock20	den_Subtract	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

В верхней ее части показан список переменных на панели с их именами, типами и значениями. Внизу показаны используемые блоки и файлы глобальных переменных.

При работе с С-Блоком используемые блоки и файлы глобальных переменных должны добавляться пользователем через диалог, который вызывается из контекстного меню. Например, если в Диаграмме-С пользователь хочет вызвать некоторую функцию fff(), то он должен добавить ее в список используемых функций. Для вызова доступны также блоки из библиотеки программы "Разработчик". Файлы этих блоков располагаются в папке "Корневая папка установки программы"\Lib\Blocks. Для выбора таких блоков в диалоге выбора тип файла должен быть установлен в "Стандартный блок".

При установке галочки "Проверка стека" в блок добавляется проверка выхода за границы стека, так что программный код становится больше, но более надежный.

Двойной щелчок левой кнопки мыши на списке используемых блоков приводит к открытию соответствующего блока (пользовательского или стандартного).

Вкладка стандартного блока содержит только окно "Инфо", что бывает полезно при вызове стандартного блока из С-блока.

4.5 Понятия

Функциональный блок.

Может быть системным или пользовательским. Системный входит в библиотеку блоков программы "Разработчик".

Пользовательский создается пользователем.

Пользовательский блок состоит из класса блока, панели и диаграммы.

Класс включает информацию о входных, выходных и сохраняемых переменных, их типах и т.д.

Панель - это окно, в котором визуально располагаются переменные блока.

Диаграмма - это окно, в котором графически изображается логика выполнения блока.

Системный блок состоит только из класса блока - описания его переменных.

Число входных и выходных переменных блока, как правило, фиксировано. Но некоторые системные блоки имеют переменное количество входных и выходных переменных.

Пользовательский блок сохраняется в один файл. Имя файла совпадает с именем блока. Имя блока может быть только английским и состоять не более чем из 8 символов. имеет расширение blk.

Файл глобальных переменных.

Состоит из класса - описания переменных и панели - их графического изображения. Глобальные переменные доступны из любого блока. Файл имеет расширение glb.

Файл определения типа.

Состоит из класса - описания типа и панели - его графического изображения. На панели можно определить тип - структуру и использовать ее в качестве типа переменной на панели блока и на панели файла глобальных переменных. При изменении типа (добавлении или удалении элементов структуры), переменные ссылающиеся на данный тип, обновляются в соответствии со этим типом. Файл имеет расширение tdf.

Панель.

Графическое представление класса - описания переменных. Панель может принадлежать функциональному блоку, файлу глобальных переменных или файлу определения типа. Для размещения элементов на панели используется меню, вызываемое по кнопке на главном окне.

Панель функционального блока содержит пиктограмму и разметку блока. С помощью контекстного меню можно поменять пиктограмму и разметку.

В режиме соединения переменные панели связываются с терминалами разметки.

Разметка блока.

Определяет количество и взаимное расположение терминалов (входов и выходов) блока. Каждый терминал связывается с определенной переменной класса. Если терминал не связан с переменной, то он не используется.

Соединение с входным терминалом может быть обязательным, рекомендуемым и необязательным.

Если соединение обязательное, то если на терминал не подается значение (с помощью провода), то при проверке выдается сообщение об ошибке.

Если соединение рекомендуемое, то если на терминал не подается значение, то при проверке выдается предупреждение.

Если соединение необязательное, то на терминал можно не подавать значение и сообщений выдаваться не будет.

Соединение с выходным терминалом всегда необязательное.

Пиктограмма блока.

При создании блока ему присваивается пиктограмма по умолчанию. Чтобы ее поменять нужно создать файл bmp с изображением размера 32x32 пикселя и поместить его в каталог DEVELOP\Lib\Images. После этого его можно выбрать и установить для блока.

Класс блока или файла глобальных переменных.

Содержит информацию о переменных блока - входных, выходных, сохраняемых.

Переменная (класса).

Переменная имеет следующие атрибуты:

- Метка - русское название переменной
- Тип данных
- Значение - начальное значение переменной. Зависит от типа данных.
- Сохранение - признак сохранения переменной.

В смысле сохранения переменная может быть внутренней, сохраняемой и РПЗУ-переменной.

Переменные верхнего уровня (не являющиеся элементами структур) имеют дополнительные атрибуты:

- Псевдоним - английское название, используется при программировании на языке С.
- Входная\Выходная - признак направления данных. Есть только у переменных блока.

Если переменная входная и связана с терминалом разметки, то свое значение она получает по проводу, связанному с этим терминалом.

Если переменная выходная и связана с терминалом разметки, то эта переменная выставляет свое значение на выход блока.

Если переменная не связана с терминалом разметки, то признак "Входная\Выходная" особого смысла не имеет.

Сохранение переменной.

Переменные класса имеют признак сохранения. Переменная может быть локальной, внутренней, сохраняемой и сохраняемой в РПЗУ.

Локальная переменная блока не сохраняет своего значения при следующем вызове этого экземпляра блока (при следующем такте).

Внутренняя переменная блока сохраняет свое значение при следующем вызове этого экземпляра блока.

Сохраняемая переменная сохраняет свое значение как внутренняя, но сохраняет свое значение также при рестарте компонента(контроллера).

При перестроении компонента и его загрузке в контроллер, сохраняемая переменная теряет свое сохраненное значение.

Переменная сохраняемая в РПЗУ сохраняет значение так же, как и просто сохраняемая переменная, но в отличие от просто сохраняемой переменной, сохраняется и при смене компонента в контроллере, если ее тип данных не изменился.

Тип данных.

Делятся на простые и составные.

К простым относятся:

- **Целочисленные** типы (провода синего цвета в графическом редакторе):

I8	- 1-байтовое целое	CHAR
I16	- 2-байтовое целое	SHORT
I32	- 4-байтовое целое	LONG
U8	- 1-байтовое беззнаковое целое	BYTE
U16	- 2-байтовое беззнаковое целое	WORD
U32	- 4-байтовое беззнаковое целое	DWORD

- **Логический** тип (зеленого цвета) - BYTEBOOL.
- **Тип с плавающей точкой** (красного цвета) – FLOAT
- **Тип Строка** (желтого цвета) STR16 - Строка длиной 16 символов

К составным типам относятся массивы и структуры.

- **Массив** – набор переменных одного типа, число которых может меняться.
- **Структура** – набор переменных разных типов, число переменных фиксировано.

Диаграмма

Диаграмма - это окно, в котором графически представлен алгоритм выполнения блока. Диаграмма может принадлежать только блоку. На диаграмме могут располагаться следующие элементы: экземпляры блоков (или просто блоки), терминалы, провода и инструкции.

Поддиаграмма.

Содержится в инструкции. Может содержать в себе те же элементы, что и диаграмма. Имеет все свойства диаграммы.

Экземпляр блока.

Содержится на диаграмме или поддиаграмме. Имеет свою копию сохраняемых переменных, если они есть в классе блока. Экземпляр может отображаться как пиктограмма и как разметка.

При наведении курсора мыши на терминал, появляется подсказка с меткой и типом данных соответствующей терминалу входной или выходной переменной.

При двойном щелчке мышью на экземпляре блока, открывается диаграмма этого блока.

Терминал.

Терминал обозначает место возможного соединения провода, то есть место источника или потребителя данных.

Существуют следующие типы терминалов:

терминал блока, терминал переменной, константа, глобальная переменная, локальная переменная, туннель.

Терминал блока принадлежит блоку и располагается на разметке блока. Использование его на чтение или запись записит от признака "Входная\Выходная", соответствующей терминалу переменной.

Терминал переменной появляется на диаграмме при добавлении переменной на панель. Его нельзя удалить. Он удаляется только вместе с переменной панели. Этот терминал можно использовать или на чтение, или на запись в зависимости от признака "Входная\Выходная" переменной панели.

Константа - это терминал с заданным значением. Используется только на чтение.

Глобальная переменная - терминал, связанный с глобальной переменной. Имеет признак "Чтение\Запись".

Локальная переменная аналогична глобальной, но связана с переменной на панели этого же блока. Имеет признак

"Чтение\Запись".

Туннель - терминал на границе инструкции. Проводит данные в и из инструкции. Направление движения данных через туннель зависит от терминалов связанных с ним проводом.

Инструкция.

Блок переменного размера. Имеет границу и одну или несколько поддиаграмм. Есть инструкции следующих типов: цикл для, цикл пока, выбор и последовательность. Тип инструкции задает правило выполнения поддиаграмм.

Провод.

Провод соединяет терминалы и представлен последовательностью точек и соединяющих их прямых. Прямые могут быть горизонтальными или вертикальными. После процедуры проверки диаграммы провода окрашиваются в цвет проходящих по ним данных.

При удалении сегмента провода или рассоединении одного из его концов нельзя заново задать соединение этому проводу, поэтому его нужно удалить.

Проект.

Соответствует компоненту Конфигуратора. Содержит библиотеки. Имеет следующие параметры:

- **Главный блок.**
- **Номер.** Под этим номером компонент регистрируется программе Конфигуратор.
- **Каталог вывода** - Каталог вывода временных файлов построения.
- **Размер стека.** Число байт выделяемых под стек. Имеет смысл только при построении для Decont182. В некоторых случаях(для больших проектов) этот параметр нужно увеличивать.

Конфигурация компонента в Конфигураторе имеет параметр "Такт" - число секунд. Работа компонента заключается в выполнении главного блока через число секунд, заданных в параметре "Такт".

Проект сохраняется в файл с расширением .dfm. Имя проекта совпадает с именем файла, может быть русским и не должно превышать 20 символов.

Библиотека.

Содержит информацию о входящих в нее блоках и об используемых ею других библиотек. Входит в состав проекта. Соответствует исполняемой библиотеке, загружаемой в контроллер.

4.6 Данные (переменные)

4.6.1 Свойства данных

Среда "Разработчик" поддерживает **следующие свойства данных** (переменных):

- **Глобальные** – переменные, которые доступны из любого блока, как на чтение, так и на запись. Глобальные переменные создаются на панели глобальных переменных (подробно описано в главе "[Панель](#)") и делятся на:
 - **Конфигурационные** - при старте работы алгоритма берут свое начальное значение из конфигурации контроллера в программе "Конфигуратор".
 - **Не конфигурационные** – значения этих переменных задаются в окошке объявления переменных на панели глобальных переменных.
- **Входные** – переменные, которые принимают свое значение перед выполнением блока. Входные переменные заполняются извне и входят в блок.
- **Выходные** - переменные, заполняемые самим блоком, которому они принадлежат.
- **Локальные** - переменные внутри блока, используемые для удобства его реализации. Сохраняются в течение одного вызова блока, используются для промежуточных вычислений.
 - **Внутренние и сохраняемые** - переменные, принадлежащие блоку, но, в отличие от локальных, сохраняющиеся от вызова к вызову блока.

- **Сохраняемые** переменные алгоритма - переменные, значения которых сохраняются после рестарта контроллера.
- **Внутренние** - переменные, значения которых не сохраняются после рестарта контроллера.
- **Сохраняемые в РПЗУ** - переменные, значения которых сохраняются в РПЗУ.

Все переменные, кроме глобальных, видны только из блоков, которым они принадлежат.

Просто сохраняемые переменные сохраняются в ОЗУ. При смене компонента (заливке новых библиотек) сохраненные данные теряются.

Переменные сохраняемые в РПЗУ сохраняются в РПЗУ. При смене компонента (заливке новых библиотек) сохраненные данные сохраняются для переменных, у которых не изменился тип данных.

Сохраняемыми переменными могут быть не только переменные панели, но и элементы структур переменных панели.

Выходные переменные могут быть сохраняемыми или иметь сохраняемые подэлементы (для структур).

Сохраняемыми могут быть глобальные переменные или подэлементы глобальных переменных(для структур).

Таблица типов:

тип переменной	область видимости	источник начального значения	время жизни
входная	блок, которому принадлежит переменная	вызывающий блок	время выполнения вызывающего блока
выходная	блок, которому принадлежит переменная	нет источника, начальное значение случайно	время выполнения вызывающего блока
глобальная не конфигурационная	любой блок	окно объявления переменной	время работы программы
глобальная конфигурационная	любой блок	конфигурационная таблица	время работы программы
локальная	блок, которому принадлежит переменная	окно объявления переменной	время выполнения блока
внутренняя не сохраняемая	блок, которому принадлежит переменная	окно объявления переменной	время работы программы
внутренняя сохраняемая	блок, которому принадлежит переменная	окно объявления переменной	до тех пор, пока не изменятся библиотеки
сохраняемая в РПЗУ	блок, которому принадлежит переменная	окно объявления переменной	пока не изменится тип переменной
глобальная сохраняемая	любой блок	окно объявления переменной	до тех пор, пока не изменятся библиотеки
глобальная сохраняемая в РПЗУ	любой блок	окно объявления переменной	пока не изменится тип переменной

Все переменные алгоритма представлены элементами на панелях блока или панели глобальных переменных. По старту алгоритма переменные некоторых типов (см. таблицу типов) принимают значения, заданные при их редактировании.

Чаще всего пользователю необходимо видеть **внутренние** (сохраняемые и не сохраняемые) и **конфигурационные** переменные. Эти три вида условно объединены названием "**специальные переменные**". Информацию о специальных переменных алгоритма можно получить, вызвав диалог через меню [Проект\Конфигурация](#):

Переменные | Дискретные | Аналоги | Счетчики | Внутренние

Такт работы: 1 Задержка старта: 1

Таблицы по типам: Полный список

Имя	Файл	Метка в файле	Тип	Примечание
gvПервый дискрет	gv	Первый дискрет	WORD	
gvКол-во каналов	gv	Кол-во каналов	WORD	
gvТаймаутABP, с	gv	ТаймаутABP, с	FLOAT	
gvТцикла, д.чч	gv	Тцикла, д.чч	FLOAT	
gvРежим	gv	Режим	WORD	

Добавить

Окно Конфигурация, закладка Переменные

Окно "**Конфигурация компонента**" содержит все конфигурационные таблицы компонента.

Данные первой таблицы (такт работы, задержка такта) представлены в виде окон в закладке "**Переменные**". В этой же закладке перечислены переменные типов WORD, DWORD и FLOAT.

Можно добавлять переменные, удалять, изменять имена, которые будут представлять эти переменные в конфигурационных таблицах.

Дискретные, аналогии и счетчики перечислены в соответствующих таблицах. Можно менять значения конфигурационных параметров. Можно (правой кнопкой мыши) менять параметры местами.

Закладка "**Внутренние**" отображает таблицы внутренних переменных.

При двойном щелчке на переменной панели открывается диалог свойств переменной:

В котором отображаются метка переменной, тип данных, начальное значение, имя переменной в с-тексте, псевдоним в с-тексте, признак входа\выхода, признак сохранения,

признак "Конфигурационная". В некоторых случаях некоторые параметры становятся недоступными, если они не имеют смысла для выбранной переменной.

Переменные, расположенные на панели глобальных переменных, могут быть конфигурационными.

Если переменная имеет тип WORD, DWORD или Float, то она попадает в одну из предопределенных таблиц с номером

- для WORD : 8, 101, 104, ..., 191
- для DWORD : 9, 102, 105, ..., 192
- для Float : 10, 103, 106, ..., 193

Если в диалоге установлен номер 0, то это равносильно номеру таблицы 8,9 или 10 в зависимости от типа.

Переменные остальных типов (не WORD, DWORD или Float), при установленном признак "конфигурационная", имеют номер от 201 до 299.

Такие переменные целых типов, в том числе внутри структур, могут быть связаны со справочником дискретов, аналогов или счетчиков.

4.6.2 Типы данных

Типы данных включают в себя **простые** и **составные**.

К простым типам относятся:

Целочисленные типы (провода синего цвета в графическом редакторе):

I8	- 1-байтовое целое	CHAR
I16	- 2-байтовое целое	SHORT
I32	- 4-байтовое целое	LONG
U8	- 1-байтовое беззнаковое целое	BYTE
U16	- 2-байтовое беззнаковое целое	WORD
U32	- 4-байтовое беззнаковое целое	DWORD

- **Логический** тип (зеленого цвета) - BYTEBOOL.
- **Тип с плавающей точкой** (красного цвета) – FLOAT

- **Тип Строка** (желтого цвета):

STR16 - Строка длиной 16 символов

В памяти строка занимает 17 байт.

Объявление:

```
typedef struct
{
    char p[17];
} STR16;
```

Длина строки определяется символом '\0'.

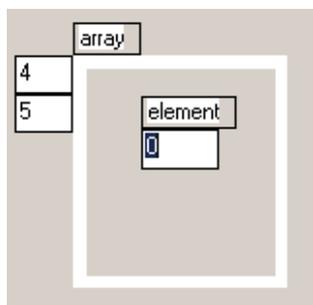
К составным типам относятся массивы и структуры.

- **Массив** – набор переменных одного типа, число которых может меняться.
- **Структура** – набор переменных разных типов, число переменных фиксировано.

Переменные типа массив и структура устанавливаются на панели из меню **<Элементы управления\Массивы (Структуры)>** соответственно. Элементы отображаются областями переменного размера.

4.6.2.1 Массив

Для массива необходимо указать **количество размерностей**, **максимальное количество элементов** по каждой из размерностей и **тип элементов**:



На рисунке изображен 2-х размерный массив с максимальным числом элементов (4 и 5 соответственно) и простым типом элементов.

Размерности добавляются и удаляются из контекстного меню, вызываемого нажатием правой кнопки мыши на границе массива.

Максимальное число элементов в каждой размерности вписывается в прямоугольники, обозначающие размерности слева вверху. Типом элементов массива будет тип элемента, расположенного внутри массива. Элемент, определяющий тип массива, может быть простого типа или структурой, но не может быть массивом.

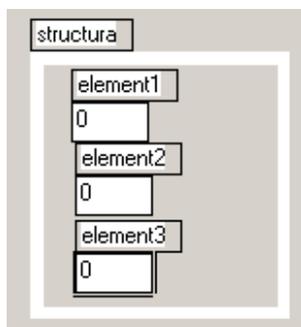
Число элементов массива может меняться во время выполнения программы, но не может превышать максимальное число. Число элементов увеличивается по следующим правилам: в начале выполнения программы число элементов в массиве равно нулю. При записи в элемент под номером n (номера элементов принимают значения от 0 до $n_{\max}-1$, где n_{\max} – максимальное число элементов) число элементов в массиве становится равным $n+1$. При дальнейшей записи в элемент с номером m , если $n \geq m$, то число элементов не меняется, если $n < m$, то число элементов становится равным $m+1$ и так далее. Для массивов размерности 1 и 2 можно задавать начальные значения элементов по правому клику на границе массива из диалога "данные...".

Работа с массивами производится через меню [<Функции/Массивы строки Пульт>](#):

4.6.2.2 Структура

Структура является упорядоченным набором элементов простых типов и других структур. Считается, что типы структур равны, если порядок и типы их элементов совпадают.

Структуры показываются как области переменного размера, внутри которых можно расположить переменное число элементов – простых типов, других структур или массивов. Элементы имеют имена, которые задаются как метки и порядок в структуре:



[Функции для работы со структурами.](#)

4.7 Работа с проектом

Любой алгоритм (логика работы), написанный в программе "Разработчик", представляет собой набор функциональных блоков ([функций](#)) и файлов [глобальных переменных](#). В блоках используются другие блоки и глобальные переменные.

Таким образом, можно вообразить себе дерево блоков и файлов глобальных переменных, на вершине которого, находится "**главный блок**", то есть такой блок, который использует прямо или косвенно все остальные блоки и файлы глобальных переменных. Этот блок не используется никаким другим.

Необходимо из этого набора блоков получить прикладной компонент, который можно затем записать в контроллер. В программе "Разработчик" для этого используется понятие **проекта**.

Проект создается для того, чтобы созданные вами блоки алгоритма, переменные и константы были уложены в библиотеки таким образом, чтобы на следующем этапе построить из проекта компонент программы "Конфигуратор". Компонент программы "Конфигуратор" состоит из библиотек. Проект содержит нужную для этого структуру, кроме того, в проекте вы указываете главную функцию и библиотеку, содержащую эту функцию. В проекте также есть диалоги различных настроек.

Процедура создания проекта, построения компонента и сопутствующих действий подробно описаны в данной главе.

4.7.1 Состав проекта

Чтобы из алгоритма, написанного в программе "Разработчик", получить работающий компонент программы "Конфигуратор", вводится понятие [проекта](#).

Каждая [функция](#) (блок) хранится в отдельном файле с расширением "blk". Файл [глобальных переменных](#) имеет расширение .glb.

Библиотека - набор блоков. Проект вместе с библиотеками сохраняется в одном файле "*.ddp". Библиотеки содержат ссылки на блоки, которые могут одновременно использоваться несколькими проектами.

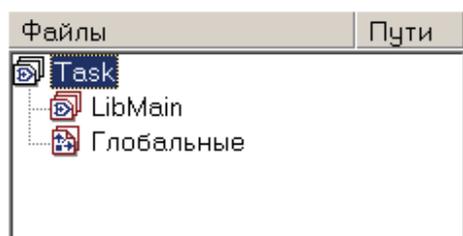
Структура проекта представлена в левой части редактора в виде дерева:

```
Проект\  
  Используемые им библиотеки\  
    Содержащиеся в них функции  
  Глобальные\  
    Список файлов глобальных переменных
```

Пример:

Файлы	Пути
ДвухтарифныйСчетчик	C:\Program Files\DEF
LibMain	
CntMain2	C:\Program Files\DEF
DZCntF	C:\Program Files\DEF
DZCnt	C:\Program Files\DEF
DeltaCnt	C:\Program Files\DEF
CurrZone	C:\Program Files\DEF
TimeConv	C:\Program Files\DEF
CntInc	C:\Program Files\DEF
Глобальные	
Zones	C:\Program Files\DEF

В пустом проекте (начальное состояние) есть только имя проекта "**Task**" и главная библиотека "**LibMain**" (пока пустая).



Библиотеки бывают **системные** (содержат встроенные, [базовые функции](#)) и **пользовательские** (содержат функции, определяемые пользователем). Системные библиотеки всегда добавляются в проект и всегда в нем находятся (пример: LibUtil), но скрыты от пользователя. Пользователь не может изменить их состав.

Блоки нужно распределить по библиотекам, исходя из того, что в контроллер не может быть загружена библиотека с размером, превышающим 8 Кбайт. Превышение сигнализируется при попытке загрузки такой библиотеки в "Менеджере файлов" программы "Конфигуратор". В случае возникновения такой библиотеки к проекту должна быть добавлена новая библиотека, в которую должна быть перенесена часть функций из "слишком большой" библиотеки.

[Главный блок](#) (функцию) необходимо добавить в **главную библиотеку** (LibMain).

Пункт **Глобальные** содержит список [глобальных переменных](#) проекта, заполняется автоматически. Если автоматического заполнения не произошло, нажмите "Обновить" (правый клик, контекстное меню).

4.7.2 Порядок работы с проектом

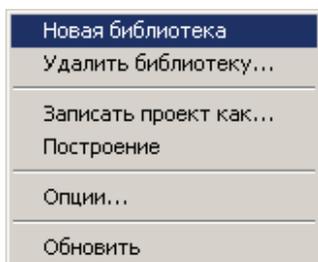
В этой главе описана вся последовательность действий, производимых после того, как вы начали писать алгоритм. Конечным продуктом будет готовый к загрузке в контроллер исполняемый программный код - **компонент** контроллера.

В процессе работы можно добавлять дополнительные блоки, удалять ненужные, при этом соответствующая информация обновляется в программе "Конфигуратор" при каждом построении проекта.

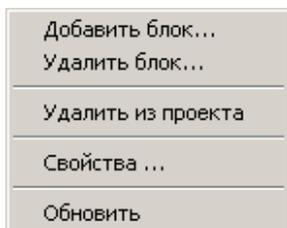
Последовательность работы с проектом:

- **Создайте необходимые вам блоки** (один из них, верхний по иерархии, является главным), которые реализуют задуманный вами алгоритм (см. "[Примеры работы в программе](#)"), проверьте их на правильность.
- **Создайте проект** (при запуске программы он уже существует). Выберите меню **<Файл\Новый проект>**. В левом окне будет отражена структура проекта: на самом верху – Task – имя проекта, в котором находится библиотека LibMain. Команды над проектом и библиотеками выполняются из их контекстного меню, которое вызывается правым нажатием мыши на соответствующий элемент.
- **Сохраните проект** (рекомендуется) Меню **<Файл\Записать проект как...>**, далее указываете место на жестком диске компьютера.
- **Выберите главный блок (главную функцию):**
 - Добавьте главную функцию в главную библиотеку LibMain:

Добавление (удаление) новой библиотеки в проект производится с использованием контекстного меню (выделить проект мышью, далее правый клик):



Добавление (удаление) блока в библиотеку производится использованием контекстного меню (выделить библиотеку мышью, далее правый клик):

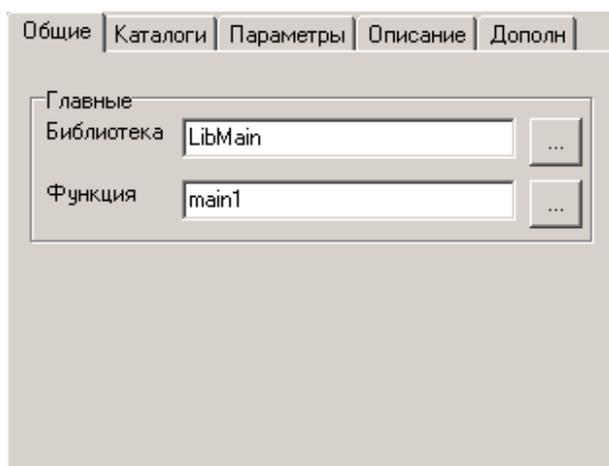


После выполнения этой команды кроме добавляемого блока в библиотеку будут добавлены все блоки, им используемые, кроме входящих в базовую (встроенную) библиотеку программы "Разработчик" или другие библиотеки проекта. Все это отразится в окне проекта:

Файлы	Пути
Пример	C:\Program Files\DEP\DE\
LibMain	
main1	C:\Program Files\DEP\DE\
Oldvalue	C:\Program Files\DEP\DE\
pid	C:\Program Files\DEP\DE\
indic	C:\Program Files\DEP\DE\
error	C:\Program Files\DEP\DE\
mode	C:\Program Files\DEP\DE\
Глобальные	

- Укажите в свойствах проекта, какая функция будет главной:

Из контекстного меню проекта выберите команду **<Опции>**. В появившемся диалоге во вкладке "Общие" укажите главный блок ([главную функцию](#)):



Во вкладке "Каталоги" укажите какой-нибудь каталог вывода промежуточных файлов компиляции. (Например, стандартное расположение: C:\Out). Закройте диалог.

- **Проверьте проект.** Следует подготовить проект до состояния, когда по кнопке "Проверить проект" появится сообщение "Ошибок нет".
- **Постройте (скомпилируйте) проект.** Запустите процедуру построения проекта нажатием кнопки "Построение" на панели инструментов главного окна (при этом будет скомпилирован программный код). Появится окно статуса компиляции. Удачное завершение сигнализируется надписью "ОК" в этом окне. Если построение не завершено успешно, повторите построение.

Если вы хотите **зарегистрировать** этот компонент в программе "Конфигуратор", то при построении в меню Инструмент\Опции среды должна быть установлена галочка «при построении регистрировать компонент в программе "Конфигуратор"». Происходит регистрация компонента и его библиотек в программе "Конфигуратор". О пользовательских библиотеках изначально информации в программе "Конфигуратор" нет. При регистрации компонента регистрируются (кроме системных) также и библиотеки, которые он использует. В программе "Конфигуратор" создается каталог **Components**, в котором содержатся каталоги всех зарегистрированных сейчас и далее компонентов. Содержимое папки Components помещается также в каталог вывода "C:\Out".

В программу "Разработчик" возвращаются номера библиотек и сохраняются в файле проекта (по аналогии с компонентом).

В нынешней реализации программы "Разработчик" компонент может иметь только **одну версию**.

Компонент имеет конфигурацию, в которой указываются: такт работы алгоритма, номера дискретов, аналогов и счетчиков, используемых в алгоритме, значения конфигурационных переменных алгоритма типа WORD, DWORD, FLOAT. Все эти данные сведены в программе "Разработчик" в одном диалоге - меню Проект\Конфигурация.

Когда "Разработчик" впервые регистрирует компонент в программе "Конфигуратор", добавляется запись в таблице компонентов с новым свободным номером. Этот номер возвращается в "Разработчик" и сохраняется в файле проекта.

Если вы строите компонент без регистрации в программе "Конфигуратор", необходимо дать ему **номер**. (Меню Проект\Опции\Параметры).

4.7.3 Работа с С-текстом

В программе "Разработчик" можно создавать блоки используя язык программирования С. В данном разделе будет показано как это сделать.

4.7.3.1 Описание блоков

"Разработчик" имеет 2 типа блоков: блоки и С-блоки. В предыдущих разделах было показано, как закладывать функциональность в "простые" блоки, а теперь рассмотрим С-блоки.

С - блоки состоят из:

"Панель" - здесь можно определить входные - выходные параметры блока, локальные переменные, выбрать их сохранение (нет, внутреннее, сохр. в ОЗУ, сохр. в РПЗУ), проинициализировать начальными значениями.

"Диаграмма - С" - здесь пишется код самого блока С, тело функции. Можно использовать переменные, определенные в "Панель-текст", функции с переменными из "Диаграмма-текст", а также "Глобальные переменные" и блоки, описанные во вкладке "Инфо".

"Панель - текст" - здесь необходимо объявлять переменные и функции со спецификатором extern.

"Диаграмма - текст" - здесь определяются переменные и функции, которые можно вызывать в "Диаграмма - С".

"Описание" - вкладка для описания и комментариев данного блока.

"Инфо" - на этой вкладке можно просматривать и редактировать переменные, определенные на вкладке "Панель".
Подробнее смотрите [Инфо](#).

При построении проекта для любого блока генерируется пара файлов *.c и *.h:

```
h-файл блока состоит из:  
// Объявление функции  
// Содержимое Панели-текст
```

```
С-файл блока состоит из:  
// Включения используемых файлов блоков и глоб. переменных  
// Содержимого Диаграмма-текст  
// Объявление функции { Содержимое Диаграмма-С }
```

Для файлов глобальных переменных генерируется только файл *.h. Результат генерации h-файл выглядит так:

```
// Содержимое Панели (объявления переменных и т.д.)
// Содержимое Панели-текст
```

Сгенерированные файлы можно посмотреть в "Каталоге вывода".

4.7.3.2 Вызов одно блока из другого

После того, как мы определились со структурой блоков, покажем как можно вызывать один блок из другого.

Предположим, что в блоке VMain необходимо вызвать блок Func1, причем Func1 может быть как "простым" блоком, так и блоком - C.

Для начала необходимо в блоке VMain на вкладке "Инфо->Используемые блоки" добавить блок Func1.

Используемые блоки				
Имя	Функция	Внутрен...	Сохран. ОЗУ	Сохран. РПЗУ
DBlock0	Func1	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

В диалоге добавления есть возможность выбрать кроме пользовательских блоков (файлов с расширением blk) стандартные блоки библиотеки программы (файлы с расширением dfm), открыв предварительно папку "Папка установки программы Разработчик\Lib\Blocks и установив тип файлов "Стандартный блок".

Подробнее об "Используемых блоках" [Инфо](#).

Теперь для нас очень важно то имя, которое находится на вкладке "Используемые блоки". В нашем случае - это DBlock0. Именно в этом параметре будет выделена вся память под переменные.

В общем случае прототип вызова блока func выглядит так:

```
void func(_tI_func *internal, _tS_func *save, _tR_func *rpzu, _tO_func *out, type1 var1, ... , typeN varN);
```

где

```
internal - указатель на внутренние переменные
save - указатель на сохраняемые переменные
rpzu - указатель на сохраняемые в РПЗУ переменные
out - указатель на выходные переменные
var1,...,varN - входные переменные.
```

Имена полей структуры _tO_user определяется по именам выходных переменных на вкладке "Инфо" блока func.

Если в вызываемом блоке, например, нет внутренних переменных, то прототип функции будет выглядеть так

```
void func( tS_func *save, _tR_func *rpzu, _tO_func *out, type1 var1, ... , typeN varN );
```

То есть этого параметра просто не будет в прототипе. И аналогично с другими параметрами.

В нашем случае блок Func1 имеет внутренние переменные, переменные, хранимые в РПЗУ, и одну входную переменную типа WORD, значит в "Диаграмма - C" блока VMain пишем:

```
Func1(&internal->DBlock0, &rpzu->DBlock0, 5); //в качестве входной переменной, например, передадим 5
```

Если мы добавим еще раз блок Func1 на вкладку "Используемые блоки", то добавится еще одно имя. Если использовать эти два имени в коде, то они будут отвечать за разные места в памяти, и, естественно, там будут храниться разные значения переменных.

Бывают случаи, когда несколько блоков должны использовать одни и те же переменные. Для этого необходимо при разработке блока использовать не локальные, а глобальные переменные.

4.7.3.3 Работа с переменными

Переменные в "Разработчике" могут быть глобальными и локальными. Глобальные переменные хранятся в специальном блоке - блоке глобальных переменных. Локальные переменные хранятся в самом блоке.

Пусть glob - имя файла глобальных переменных. Тогда сгенерированный файл глобальных переменных представляет собой структуру:

```
struct _t_glob
{
    type1 var1;
    ...
    type1 varN;
}
```

где;

var1,..., varN - переменные панели.

Для того, чтобы в блоке использовать файл глобальных переменных, его необходимо добавить во вкладку "Инфо->Файлы глобальных переменных".

Для обращения к системным глобальным переменным используйте структуру dep_SysVars:

```
"Первый старт"    dep_SysVars.FirstStart
"Первый старт"    dep_SysVars.ReStart
```

Обращение к элементам, имеющим простой тип данных.

Для обращения к элементам на языке C, имеющим простой тип, необходимо задать псевдоним, который задается в окне "Свойства переменной".

Например, добавим на "Панель" переменную типа WORD. Для того, чтобы к ней обратиться из функции на C необходимо ей задать "Псевдоним (#define)", например

Псевдоним (#define)

Temp

Тогда можно писать:

```
Temp = 3;
```

Обращение к элементам, имеющим сложный тип данных.

Рассмотрим работу со сложными типами на примере двумерного массива типа DWORD размерности 10x5. Вот его сгенерированная структура:

```
typedef struct
{
    DWORD ArrData[10][5];
}
```

```
WORD Counts[2];
}_t_ArrName;
```

В ArrData находятся данные массива, в Counts находятся количество элементов по каждой из размерностей. Рассмотрим методы обращения к массиву.

1) Если

Псевдоним (#define)	<input type="text" value="ArrayTemp"/>
Псевдоним массива	<input type="text"/>

Тогда

```
//1ая размерность = 10
int size1 = ArrayTemp.Counts[0];
//2ая размерность = 5
int size2 = ArrayTemp.Counts[1];
//обращение к элементу
DWORD element = ArrayTemp.ArrData[0][0];
```

2) Если

Псевдоним (#define)	<input type="text"/>
Псевдоним массива	<input type="text" value="ArrayTemp"/>

Тогда

```
//обращение к элементу
DWORD element = ArrayTemp[0][0];
```

Создание глобальных переменных на языке C.

Данная возможность работает только для платформы Decomp-182!!!

Глобальные переменные можно создавать на языке C. Покажем это на примере переменной Var типа WORD.

Для этого создадим файл глобальных переменных и в его вкладке "Панель-текст" напишем следующую строку:

```
extern WORD Var;
```

Во вкладке "Панель-C":

```
WORD Var;
```

При старте компонента переменная не проинициализирована, поэтому начальное значение нужно задать программно. Писать во вкладке

```
"Панель-C" WORD Var = начальное_значение;
```

нельзя, иначе переменная не будет глобальной (из-за особенностей компилятора). Теперь переменную Var можно использовать в любом блоке, использующем созданный файл глобальных переменных.

Перед сборкой компонента, нужно выяснить полный размер всех созданных таким образом переменных и занести его в свойства проекта в параметр "Размер дополнительных глобальных переменных". Полный размер равен сумме размеров переменных.

Размеры структур рассчитываются сложением размеров отдельных их элементов.

4.7.3.4 Отладка

При отладке С-текста используется окно вывода, которое вызывается из меню Вид\Вывод. Вывод работает только при включенной "галочке" в диалоге "Инструменты\Опции среды\Некоторые\Вывод компилятора". В случае ошибки в окне вывода будет указан номер строки в которой произошла ошибка. Чтобы по этому номеру определить место ошибки, нужно использовать файл из каталога вывода. Также можно два раза щелкнуть на ошибке и она подсветится.

4.7.3.5 Справочник специальных функций

В данном разделе описаны функции, которые можно использовать в С-блоках. Прототипы функций и дополнительную информацию можно найти в h-файлах, расположенные в папке [каталог Разработчика]\INCLUDE. Для каждой платформы построения (WinDecont, Decont-A9 и Decont-182) есть свои особенности, которые надо учитывать при программировании.

Работа с базой текущих значений

Базы текущих значений описаны здесь. Прототипы функции для работы с базой в файле base.h.

Для работы с текущими значениями определены следующие типы:

GLOBAL – номер в базе параметров

STATE – слово состояния параметра.

Биты слова состояния:

flagU - признак неопределенности - значение неизвестно

flagD - бит Динамики

flagExt - прикладной флаг

Маска кода ошибки в слове состояния параметра:

MAXBASEERR

Удобно использовать макросы:

ERRCOD (xx) – возвращает код ошибки из STATE

SETERR (err) – возвращает сформированный STATE с признаком неопределенности и кодом ошибки err

STATECOD (xx) – возвращает значение дискрета.

Работа с битами:

SET (f, xx) – возвращает xx с установленным битом f

RESET (f, xx) - возвращает xx со сброшенным битом f

IFSET (f, xx) - возвращает, установлен ли бит f в параметре xx

Формат параметра типа дискрет:

```
typedef STATE DISCRET;
```

Формат параметра аналог:

```
typedef struct {
    STATE    state;
    ANVAL    value;
}ANALOG;
```

Формат счетчика:

```
typedef struct {
    STATE    state;
    CNTVAL   value;
}COUNTER;
```

Простая Запись значений: если в записываемом значении установлен бит Динамики, то в базе параметров бит Динамики будет так же установлен. Если в записываемом значении сброшен бит Динамики, тогда в базе параметров бит динамики останется без изменений.

```
void DiscretWrite (GLOBAL glb, DISCRET st);
void AnalogWrite (GLOBAL glb, ANALOG *aa);
BOOL CounterWrite (GLOBAL glb, COUNTER *cc);
```

Запись с автоматическим формированием динамики: для дискретов бит динамики устанавливается при изменении состояния дискрета с «0» на «1», для аналогов и счетчиков – при любом изменении значения.

```
void DiscretSet (GLOBAL glb, DISCRET st);
void AnalogSet (GLOBAL glb, ANALOG *aa);
BOOL CounterSet (GLOBAL glb, COUNTER *cc);
```

Простое Чтение значений: прочитать значение параметра из базы текущих значений.

```
DISCRET DiscretRead (GLOBAL glb);
void AnalogRead (GLOBAL glb, ANALOG *aa);
void CounterRead (GLOBAL glb, COUNTER *cc);
```

Чтение со сбросом динамики в базе

```
DISCRET DiscretGet (GLOBAL glb);
void AnalogGet (GLOBAL glb, ANALOG *aa);
void CounterGet (GLOBAL glb, COUNTER *aa);
```

Примеры:

Чтение аналога:

```
ANALOG analog;

AnalogRead(10, &analaog); // прочитали значение 10-ого аналога
```

```
if( !IFSET(flagU, analog.state) ){
    // значение определено можно использовать analog.value
}else{
    // значение аналога неопределенно, надо обрабатывать ошибку
}
```

Запись аналога:

```
ANALOG analog;
analog.state = 0;      // значение достоверно
analog.value = 10.5;  // и равно 10,5
```

Чтение дискрета:

```
DISCRET dsc;

dsc = Read(10001);    // прочитали значение дискрета 10001
if( !IFSET(flagU,dsc) ){
    // значение определено можно использовать STATECOD(dsc)
}else{
    // значение неопределенно, надо обрабатывать ошибку
    // код ошибки ERRCOD(dsc)
}
```

Запись дискрета:

```
DiscretWrite(10002,1); // записать значение 1 в дискрет с номером 10002
```

Системные параметры

В контроллере есть большая группа параметров, называемых системные параметры. Часть из них доступна только на чтение (например, серийный номер контроллера), другая часть доступна для чтения-записи (например, режим работы – при чтении возвращает текущий режим работы, при записи рестартует контроле в указанном режиме работы). Коды системных параметров и функции чтения-записи описаны в файле sm.h.

Чтение системного параметра:

```
WORD smGetValue( HSMP ParamID, void* Value, void* Target );
```

ParamID – номер системного параметра

Value – указатель на переменную, в которую поместить значение прочитанного параметра

Target – NULL

Возвращаемое значение:

RES_OKAY – чтение успешно выполнено,
иначе код ошибки (коды ошибок в results.h)

Пример: чтение серийного номера контроллера

```
DWORD dwSerNo;
```

```
dwSerNo = 0;
```

```
if ( smGetValue(SMP_NUM_SERIAL_NUMBER, & dwSerNo, NULL) != RES_OKAY )
```

```
dwSerNo = 0;
```

Запись системного параметра:

```
WORD smSetValue( HSMP ParamID, void* Value, void* Target );
```

ParamID – номер системного параметра

Value – указатель на переменную, в которой лежит записываемое значение

Target – NULL

Возвращаемое значение:

RES_OKAY – чтение успешно выполнено,
иначе код ошибки (коды ошибок в results.h)

Пример: запись параметра режим работы – приводит к рестарту контроллера.

```
WORD aValue;  
aValue=0; // рестарт в том же режиме работы  
smSetValue(SMP_SYS_SYSTEM_MODE, &aValue, NULL);
```

Прикладные параметры

Конфигурационные параметры компонентов называют прикладными параметрами. Возможно, с ними Вы сталкивались при настройке компонента «Дисплей». Функции работы с прикладными параметрами описаны в arman.h.

```
// прочитать/записать значение параметра (CompID, ParmID) в *Value:  
// Arg1 - номер строки в таблице (нумерация с 0),  
// Arg2 - смещение номера таблицы в диапазоне (нумерация с 0)  
WORD amGet(WORD CompID, WORD ParmID, void *Value, xWORD Arg1, WORD Arg2);  
WORD amSet(WORD CompID, WORD ParmID, void *Value, xWORD Arg1, WORD Arg2);  
// прочитать/записать значение параметра (CompID, ParmID) в *Value из той же строки  
// таблицы,  
// в которой значение параметра (CompID, IndexID) равно *IndexValue:  
// Arg2 - смещение номера таблицы в диапазоне  
WORD amGetByParm(WORD CompID, WORD ParmID, void *Value, WORD IndexID, void *IndexValue, WORD  
Arg2);
```

Возвращаемое значение:

RES_OKAY – чтение успешно выполнено,
иначе код ошибки (коды ошибок в results.h)

CompID – ID компонента, к параметру которого идет обращение. Номер компонента проще смотреть в таблице "Системная задача\Компоненты" если отжать ключик ("Скрыть ключевое поле") то в поле "Компонент ID" виден ID всех экземпляров.

Есть макрос для расчета ID компонента:

```
MAKE_COMPID(c, i)  
c-компонент ID (из справочника Конфигуратора или clist.h)  
i-номер экземпляра
```

ParmID – номер прикладного параметра. Этот номер можно посмотреть в Конфигураторе: меню "Администратор\Администратор задач". Вкладка Компоненты: поле ID Компонента - номер компонента. Вкладка Задачи: в

списке "Список компонентов" активизируйте нужный компонент - справа будет описание таблиц. Активизируйте нужную конфигурационную таблицу, будет список полей таблицы (внизу справа). Номер параметра - это поле ParmID.

Arg1 - номер строки в конфигурационной таблице, нумерация начинается с 0.

Arg2 – смещение номера таблицы в диапазоне необязательных таблиц компонента (нумерация с 0). Например, «Список номеров_2» в компоненте «База-клиент» будет иметь смещение «1».

Пример чтения серийного номера первого модуля в компоненте BUS-драйвер (из первой строки в таблице "Модули ввода/вывода")

```
DWORD dw;
if( amGet(33288,14,&dw,0,0) != RES_OKAY) dw=0xFFFF;
```

CompID - номер компонента «BUS-драйвер_0»=33288

ParmID - номер параметра. Серийный номер =14

Arg1 - номер строки в таблице=0 - адрес находится в первой строке.

Arg2 =0 для данного параметра не используется. Данный аргумент нужен, если параметр лежит в необязательной таблице, которых может быть много.

Чтение серийного номера устройства номер "3":

```
DWORD dw,dwIndex;
dwIndex=3;
if( amGetByParm(33288,14,&dw,13,&dwIndex,0) != RES_OKAY) dw=0xFFFF;
```

Отладочные функции

Запись в журнал ошибок (ctrl.h)

```
void WatchPoint ( xWORD DebugCode, xWORD Argument );
```

Запись в журнал сообщений

```
void TracePrintf( char* Message, ... );
```

Аргументы и формат сообщения как у обычной функции printf

```
void TracePrintDump(unsigned char *Buffer, unsigned int Size, unsigned int Addr);
```

Выводит дамп буфера Buffer длиной Size. Перед дампом печатается адрес Addr в шестнадцатеричном виде.

4.8 Функции

Функции (функциональные блоки) подразделяются на **базовые** (встроенные) и **определенные пользователем** и являются основными образующими элементами алгоритма.

Существует понятие главной функции.

Главная функция – это функциональный блок, вызывающий остальные блоки. Он находится на верхней ступени иерархии блоков, его не вызывает ни один блок. В проекте обязательно должна быть **задана главная функция**. Она

задается в меню Проект \ Опции \ Общие:

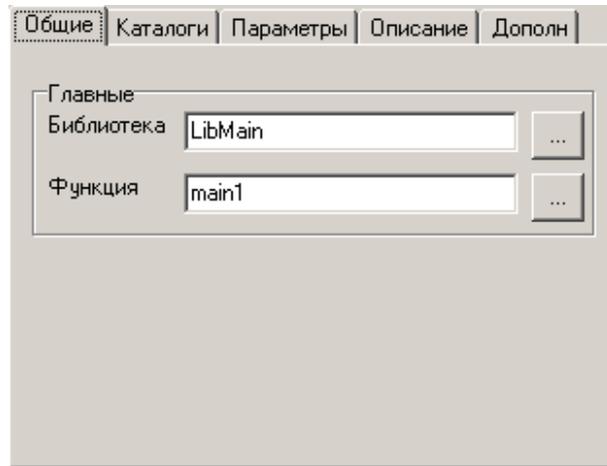
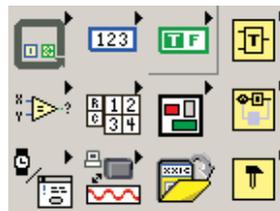


рис. Свойства проекта

Эта процедура должна быть обязательно выполнена **перед любым действием с проектом** - проверкой проекта, построением проекта и так далее.

4.8.1 Базовые функции

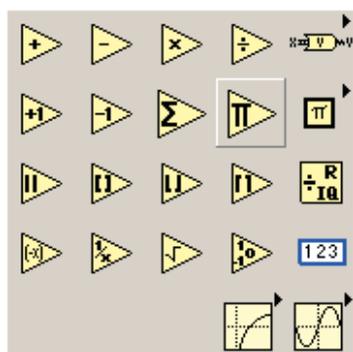
Функции **базовой библиотеки** предоставляют основные операции манипулирования данными. Выбор функций осуществляется через панель - меню **<Функции>** ([меню основной панели](#)).



Некоторые функции поддерживают полиморфизм – возможность функции принимать на вход данные разного типа. В описании каждой полиморфной функции указывается, входные данные каких типов она допускает.

4.8.1.1 Численные функции

 Численные функции вызываются из подменю <Функции\Численные> [меню основной панели](#) и принимают на вход переменные численного типа. Тип выхода является, как правило, "наибольшим" типом входных переменных. В других случаях тип выхода оговаривается особо.



	Сложение Результат – сумма двух чисел.		Вычитание Результат – разность двух чисел.
	Умножение Результат – умножение двух чисел.		Деление Результат – деление двух чисел.
	Инкремент Увеличивает число на входе на единицу.		Декремент Уменьшает число на входе на единицу.
	Сумма элементов массива Возвращает сумму всех элементов массива. Тип элемента массива должен быть численным. Выход имеет тип элемента массива.		Произведение элементов массива Возвращает произведение всех элементов массива. Тип элемента массива должен быть численным. Выход имеет тип элемента массива.
	Модуль числа - abs(x) Возвращает абсолютное значение числа		Обратное число Делит 1 на входную величину. Тип выходной переменной – FLOAT.
	Округлить до большего целого - ceil(x) Возвращает ближайшее большее целое. Например, если на входе 3,1, то на выходе 4.		Округлить до меньшего целого - floor(x) Возвращает ближайшее меньшее целое. Например, если на входе 3,1 - на выходе 3.
	Округлить до ближайшего целого Возвращает ближайшее целое число. Если значение входа точно посередине между целыми (например 1,5) - возвращается меньшее целое.		Целая часть и остаток от деления Подсчитывает целочисленные частное и остаток деления.

	Знак числа На выходе – 1, если на входе число больше нуля, 0 – равное нулю и –1 – меньше нуля.		Квадратный корень Подсчитывает квадратный корень входной величины. Тип выходной переменной – FLOAT.
	Противоположное число Возвращает противоположное по знаку число. Тип входа должен быть знаковым.		

4.8.1.1.1 Универсальные константы (подменю числ. функций)



Универсальные константы доступны из подменю <Константы> меню <Численные> [основной панели](#).



	π (число Пи)
	2π (число Пи, умноженное на 2)
	$\pi/2$ (число Пи, деленное на 2)
	$1/\pi$ (единица, деленная на Пи)
	$\ln \pi$ (натуральный логарифм Пи)
	e (число e)
	$1/e$ (единица, деленная на e)
	$\ln 10$ (натуральный логарифм 10)
	$\ln 2$ (натуральный логарифм 2)
	$\log_{10} e$ (десятичный логарифм e)

4.8.1.2 Логарифмические функции



Логарифмические функции вызываются из подменю <Функции\Численные\Логарифмические> [меню основной панели](#):

Имеются следующие логарифмические функции:

	e^x (экспонента)
	10^x (степень числа 10)
	x^y (X в степени Y)
	\ln (натуральный логарифм)



$\log x$ (логарифм по основанию)

Все они возвращают данные **типа FLOAT**. У всех этих функций, кроме X в степени Y , один вход, (у этой функции – два) и один выход.

4.8.1.3 Тригонометрические функции



Тригонометрические функции вызываются из подменю <Функции\Численные\Тригонометрические> [МЕНЮ ОСНОВНОЙ панели](#). Имеются следующие тригонометрические функции:

	Sin (синус)
	Cos (косинус)
	Tg (тангенс)
	Arcsin (арксинус)
	Arccos (арккосинус)
	Arctg (арктангенс)

Все они возвращают данные типа **FLOAT**. У всех этих функций один вход, один выход.

4.8.1.4 Преобразование типов (подменю численных функций)

Функции преобразования типов доступны в подменю <Преобразования> меню <Численные>:



	Преобразование входной переменной в BYTE – беззнаковое 8-битное целое
	Преобразование входной переменной в CHAR – знаковое 8-битное целое
	Преобразование входной переменной в WORD – беззнаковое 16-битное целое
	Преобразование входной переменной в SHORT – знаковое 16-битное целое
	Преобразование входной переменной в DWORD – беззнаковое 32-битное целое
	Преобразование входной переменной в LONG – знаковое 32-битное целое
	Преобразование входной переменной в FLOAT – число с плавающей точкой
	Преобразование входной логической переменной в число WORD, которое равно 0, если вход FALSE и 1, если вход TRUE
	Преобразование логического массива в число
	Преобразование числа в логический массив

4.8.1.5 Логические функции

В разделе описаны имеющиеся логические функции.

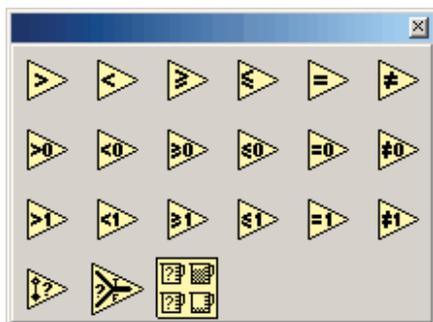
В пояснениях x - верхний вход, а y – нижний вход для блоков с двумя входами:



	Логическое И Оба входа блока должны быть логического типа или числа. Если входы численные, то выполняется побитовое И.		Логическое ИЛИ Оба входа блока должны быть логического типа или числа. Если входы численные, то выполняется побитовое ИЛИ
	Исключающее ИЛИ Входы должны быть логического или численного типа. Если входы численные, то выполняется побитовое Исключающее ИЛИ.		Отрицание Подсчитывает логическое НЕТ входа. Тип входа численный или логический. Если вход численный, то выполняется побитовое НЕТ.
	Логическое И элементов массива Возвращает TRUE, если все элементы массива равны TRUE. Массив может быть произвольной размерности.		Логическое ИЛИ элементов массива Возвращает FALSE, если все элементы массива равны FALSE, в противном случае возвращает TRUE
	НЕ И Подсчитывает логическое отрицание от логического И между x и y. Оба входа должны быть численного или логического типа. Для чисел выполняется побитовая операция.		НЕ ИЛИ Подсчитывает логическое отрицание от логического ИЛИ между x и y. Оба входа должны быть численного или логического типа. Для чисел выполняется побитовая операция.
	НЕ Исключающее ИЛИ Подсчитывает логическое отрицание от Исключающего ИЛИ между x и y. Оба входа должны быть численного или логического типа. Для чисел выполняется побитовая операция.		Следование Подсчитывает логическое ИЛИ между y и отрицанием x. Оба входа должны быть логического или численного типа. Для чисел выполняются побитовые операции
	Логическая константа – константа типа BOOLEAN		

4.8.1.6 Функции сравнения

В пояснениях x указывает на верхний вход, а y - на нижний для блоков с двумя входами.



	Больше Возвращает TRUE, если x больше y. Иначе возвращает FALSE.		Меньше Возвращает TRUE, если x меньше y. Иначе возвращает FALSE.
	Больше или равно Возвращает TRUE, если x больше или равно y. Иначе возвращает FALSE.		Меньше или равно Возвращает TRUE, если x меньше или равно y. Иначе возвращает FALSE.
	Равно Возвращает TRUE, если x равно y. Иначе возвращает FALSE.		НЕ равно Возвращает TRUE, если x не равно y. Иначе возвращает FALSE.
	Больше нуля Возвращает TRUE, если вход больше нуля.		Меньше нуля Возвращает TRUE, если вход меньше нуля
	Больше или равно нулю Возвращает TRUE, если вход больше или равен нулю.		Меньше или равно нулю Возвращает TRUE, если вход меньше или равен нулю.
	Больше единицы Возвращает TRUE, если вход больше единицы.		Меньше единицы Возвращает TRUE, если вход меньше единицы
	Больше или равно единицы Возвращает TRUE, если вход больше или равен единице.		Меньше или равно единицы Возвращает TRUE, если вход меньше или равен единице.
	Равно единице Возвращает TRUE, если вход равен единице.		Не равно единице Возвращает TRUE, если вход не равен единице.
	Равно нулю Возвращает TRUE, если вход равен нулю.		Не равно нулю Возвращает TRUE, если вход не равен нулю.
	Принадлежность диапазону hi – верхний вход, x- средний вход, lo – нижний вход. Возвращает TRUE, если hi>=x и x>=lo.		Выбор из двух переменных t – верхний вход, s- средний вход, f – нижний вход. Типы t и f должны совпадать, s имеет тип BOOLEAN. Возвращает t, если s равно TRUE, и f в противном случае.

	Определение максимума и минимума. Выдает из верхнего вывода наибольшее из x и y , и из нижнего наименьшее		
---	--	--	--

4.8.1.7 Функции для работы с массивами

Работа с массивами производится через меню <Функции/Массивы строки Пульт>:



Для работы с массивами определены следующие функции:

	<p>Количество элементов в массиве Имеет один вход и один выход. На вход подается массив. Если массив одномерный, то функция возвращает число типа WORD – число элементов. Если массив многомерный, то результат функции – массив с числом элементов, равным количеству размерностей поданного на вход массива, в элементе с индексом n – число элементов исходного массива в размерности $n+1$.</p>
	<p>Получить элемент массива На входы функции подаются массив и индекс элемента по каждой из размерностей. Число входов блока переменное и должно быть установлено в соответствии с числом размерностей массива. Нумерация массива начинается с 0. На выходе блока – переменная, значение и тип которой равны соответствующему элементу массива.</p>
	<p>Заменить элемент массива На входы функции подаются массив M, элемент Ei и индекс элемента по всем размерностям. Число входов переменное и должно быть установлено в соответствии с числом размерностей входного массива. На выходе получаем массив, равный M, с замененным соответствующим элементом на Ei.</p>

Одномерные и двумерные массивы простых типов данных и одномерные массивы структур из простых типов данных можно заполнить начальными значениями с помощью следующей формы:

	0	1	2	3
0	1	2	3	4
1	1	2	3	4
2	1	2	3	4
3	1	2	3	4
4	1	2	3	4

Строки: Кол-во 5, Макс. кол-во 5

Столбцы: 4, 6

Тип данных: WORD

Без данных

Имя переменной: .DGlobItem0.ArrData[3][1]

OK Отмена

В этой форме можно задать размерности массива, количество элементов по каждой размерности и для простых типов данных - тип. Если массив не имеет данных, то нужно ставить галочку - "Без данных". Если данные введены неверно, то соответствующее поле окрашивается в красный цвет.

4.8.1.8 Функции для работы со структурами



Работа со структурами производится из меню <Функции/Структуры>:



	<p>Создание структуры Блок имеет входной терминал 1, на который может быть подана переменная типа структура, выходной терминал 2, выдающий структуру, и переменное число входных терминалов 3 – элементы структуры. Если на терминал 1 подана структура, то на выходе блока будет структура того же типа. В этом случае связывание (подача переменных) терминалов 3 обязательно. Связанные терминалы изменят значения элементов структуры в соответствии с их порядком. Если терминал 1 не связан, то связывание терминалов 3 обязательно и на выходе блока будет структура, порядок и типы элементов которой соответствуют порядку и типам переменных, поданных на терминалы 3-го блока.</p>
	<p>Элементы структуры На вход блока подается структура. Число выходов переменное, и на них выдаются элементы структуры в соответствии с их порядком.</p>
	<p>Создание структуры по имени Так же, как и блок "создание структуры", блок "создание структуры по имени" имеет входной терминал 1, на который может быть подана переменная типа структура, выходной терминал 2, выдающий структуру, и переменное число входных терминалов 3 – элементы структуры. На вход 1 должна быть подана структура. На выходе будет структура, равная поданной. На терминалы 3 подаются элементы, которые заменят значения элементов входной структуры в соответствии с именами элементов. Имена элементов задаются из всплывающего меню при нажатии правой кнопкой мыши на области входного терминала.</p>
	<p>Элементы структуры по имени Блок аналогичен блоку "Элементы структуры", в отличие от которого выходные элементы выдаются не в соответствии с порядком элементов в структуре, а в соответствии с заданными именами. Имена задаются через всплывающее меню при нажатии правой кнопкой мыши на области входного терминала.</p>

Замечание: Для появления списка элементов структуры в контекстном меню для блоков "Создание структуры по имени" и "Элементы структуры по имени" необходимо выполнить операцию проверки диаграммы при соединенном терминале входной структуры.

4.8.1.9 Функции для работы со строками

Меню открывается :Функции\ МАССИВЫ СТРОКИ ПУЛЬТ. 



Для работы со строками используются следующие функции:

	<p>Функция "Длина строки" Вход: STR16 str; строка Выход: BYTE len; длина строки str.</p>
	<p>Функция "Строку в целое" Вход: STR16 str; строка Выходы: LONG value; целое значение BYTEBOOL valid; признак достоверности значения value. Функция преобразует строку в целое число. Признак valid указывает на то, удалось ли преобразование.</p>
	<p>Функция "Строку в вещественное" Вход: STR16 str; строка Выходы: FLOAT value; вещественное значение BYTEBOOL valid; признак достоверности значения value. Функция преобразует строку в целое число. Признак valid отображает, удалось ли преобразование.</p>
	<p>Функция "Целое в строку" Входы: LONG value; целое число BYTEBOOL sign; указывать ли знак, если число положительное. По умолчанию равно FALSE BYTE digits; количество символов, которое должна занимать строка – по умолчанию 0 Выход: STR16 str; результат преобразования.</p>
	<p>Функция "Вещественное в строку" Входы: LONG value; вещественное число BYTEBOOL sign; указывать ли знак, если число положительное. По умолчанию равно FALSE BYTE digits; количество знаков после запятой в результирующей строке – по умолчанию 0 Выход: STR16 str; результат преобразования</p>
	<p>Функция «Поместить строку в строку» Входы: STR16 str1; первая строка STR16 str2; вторая строка BYTE pos; позиция в первой строке – по умолчанию равно 0 BYTE Count; количество элементов второй строки – по умолчанию 16 Выход: STR16 str; результат</p>

	<p>При выполнении функции Count символов второй строки помещаются в первую строку, начиная с позиции pos. Символы, выходящие за пределы длины первой строки, будут отброшены.</p>
	<p>Функция «Под строка» Входы: STR16 str; исходная строка BYTE pos; позиция в исходной строке – по умолчанию 0 BYTE count; количество символов – по умолчанию 16 Выход: STR16 out; результирующая строка В результате выполнения функции получаем строку, заполненную символами строки str, начиная с позиции pos, количеством не больше, чем count или не больше, чем длина строки. Из count и длины строки выбирается меньшее.</p>
	<p>Функция «Добавить строки» Входы: STR16 left; левая строка STR16 delim; разделитель STR16 right; правая строка Выход: STR16 out; результирующая строка На выходе функции получаем совмещение трех строк: левой, разделителя и правой.</p>
	<p>Функция «Добавить строку в конец» Входы: STR16 left; левая строка STR16 right; правая строка Выход: STR16 out; результат На выходе функции получаем строку длиной 16 символов, составленную из строк left и right. Left выровнена по левому краю, right – по правому. Если общая длина строк left и right меньше 16 символов, то между ними будут пробелы, иначе строка right перекроет строку left.</p>

4.8.1.9.1 Функции для работы с пультом

Функции для работы с минипультотом или шкафным пультом (BoxPult) встроены в меню [Функции\Массивы строки](#) (на рисунке обведены):



	<p>"Использовать пульт" Входы: BYTEBOOL todo; не обязателен, по умолчанию равен TRUE. Выходы: BYTEBOOL ok; Функция вызывается алгоритмом, в котором выполняется работа с пультом. В работающем контроллере только один компонент может быть зарегистрирован для работы с пультом. Если несколько компонентов вызывают эту функцию, то зарегистрирован будет компонент, вызвавший функцию первым. Регистрация компонента выполняется, если todo равно TRUE. Ok равно TRUE, если компонент успешно зарегистрирован для работы с пультом, или если todo был равен FALSE.</p>
---	---

	<p>"Код нажатой клавиши". Выходы: BYTEBOOL mode; режим работы BYTE code; код нажатой клавиши.</p> <p>Функция возвращает код нажатой клавиши и режим работы с пультом. Mode равно TRUE, если в данный момент пульт находится в пользовательском режиме работы. Если нет нажатой клавиши, то code равно 0.</p>
	<p>"Выйти из консоли" Вход: BYTEBOOL todo; выполнять или не выполнять операцию.</p> <p>Если пульт находится в пользовательском режиме, то при выполнении этой функции пульт переходит в системный режим.</p>
	<p>"Вывести строку на консоль" Входы: STR16 str; выводимая на пульт строка BYTE line; позиция вывода: 0 – вывод в верхнюю строчку пульта, 1 – в нижнюю.</p>

Дополнительная информация по работе с Прикладным экраном описана в компоненте ["Дисплей"](#)

Таблица кодов клавиш:

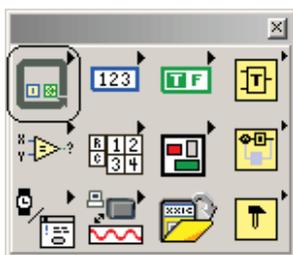
Клавиша	Код нажатия	Код отжатия
UP	27	155
DOWN	30	158
LEFT	29	157
RIGHT	31	159
ESC	33	
ENTER	28	156
OVER	255	
Клавиша	Код нажатия	Код отжатия
F1	8	136
F2	9	137
F3	10	138
F4	11	139
F5	12	140
F6	13	141
F7	14	142
F8	15	143

Клавиша	Код нажатия	Код отжатия
0	16	144
1	17	145
2	18	146
3	19	147
4	20	148
5	21	149

6	22	150
7	23	151
8	24	152
9	25	153
Клавиша	Код нажатия	Код отжатия
Point	26	154
Passw	1	129
Backsp	32	160

4.8.1.10 Инструкции

Работа с инструкциями производится через меню Функции/Инструкции:



Инструкции являются специальными элементами диаграммы функциональных блоков и аналогичны инструкциям в обычных процедурных языках программирования. В библиотеке программы "Разработчик" реализованы следующие инструкции:

<ПОСЛЕДОВАТЕЛЬНОСТЬ>, **<ВЫБОР>**, **<ЦИКЛ ДЛЯ>**, **<ЦИКЛ ПОКА>**, которые можно вызвать из меню <Инструкции>:



Графически инструкции представляют собой блоки переменного размера с выделенной границей (см.рис).

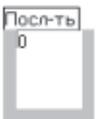
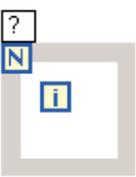
Инструкции ведут себя аналогично блокам диаграммы, то есть они выполняются, как только заполнены данными их входы, и выдают данные на выходы после завершения выполнения. Однако каждая инструкция выполняет свои поддиаграммы по правилам, описанным далее. Каждая поддиаграмма, также, как и верхняя – это набор блоков, проводов и терминалов.

Инструкции **<ЦИКЛ ДЛЯ>** и **<ЦИКЛ ПОКА>** содержат одну поддиаграмму и используются для циклических операций.

<ПОСЛЕДОВАТЕЛЬНОСТЬ> и **<ВЫБОР>** имеют переменное количество поддиаграмм, которые строятся аналогично диаграмме верхнего уровня.

Редактор автоматически создает терминалы для прохода данных в инструкцию и из нее при соединении узлов внутри и вне инструкции на ее границе. Такие терминалы называются тоннелями. Тоннели всегда расположены на границе инструкции и

могут свободно перемещаться по ней.

	<p><ВЫБОР></p> <p>Такая инструкция может иметь несколько поддиаграмм, из которых видна только одна. На верхней границе инструкции находится окно, которое содержит идентификатор поддиаграммы, видимой в данный момент. Видимая поддиаграмма может быть изменена выбором соответствующего идентификатора из выпадающего меню, которое вызывается при нажатии на окно с идентификатором.</p> <p>При выполнении инструкции <ВЫБОР> выполняется одна из диаграмм. Выбор выполняемой поддиаграммы зависит от значения численного или логического типа, поданного на терминал выбора с внешней стороны инструкции. Для добавления новой поддиаграммы используется команда <добавить> из контекстного меню, которое вызывается при нажатии правой кнопкой мыши на границе инструкции. Для удаления поддиаграммы следует сделать ее видимой и воспользоваться командой <удалить> из того же меню.</p>
	<p><ПОСЛЕДОВАТЕЛЬНОСТЬ></p> <p>применяется в случаях, когда необходимо явно задать последовательность выполнения блоков или заполнения локальных переменных. Также, как и <ВЫБОР>, инструкция имеет несколько поддиаграмм, к каждой из которых приписан номер. Поддиаграммы выполняются, начиная с нулевой.</p>
	<p><ЦИКЛ ДЛЯ></p> <p>N - терминал числа операций I - терминал номера итерации</p> <p>Терминал числа итераций с внешней стороны инструкции. Терминал номера операций содержит число полностью выполненных итераций, 0 – при первой итерации, 1 – при второй и тд до N-1.</p>
	<p><ЦИКЛ ПОКА></p> <p> - условный терминал I - терминал номера итераций</p> <p>Инструкция выполняет поддиаграмму до тех пор, пока значение, поданное на условный терминал, не равно FALSE. Проверка значения в условном терминале производится в конце каждой итерации, поэтому поддиаграмма выполняется по крайней мере один раз. Терминал номера итерации ведет себя так же, как и в <ЦИКЛ ДЛЯ>.</p>

Пункт меню **"ГЛОБ"** предоставляет возможность добавления на диаграмму терминала связанного с глобальной переменной. Терминал может быть связан с пользовательской глобальной переменной или системной. Для выбора пользовательской переменной нужно выбрать из контекстного меню файл глобальных переменных и выбрать в нем переменную.

Для выбора системной переменной нужно выбрать пункт "Системные" в контекстном меню и далее выбрать одну из переменных:

"Первый старт" или "Рестарт". Переменная "Первый старт" равна TRUE на первом такте работы компонента, если не удалось прочитать сохраняемые переменные. На следующих тактах она равна FALSE. Переменная "Рестарт" равна TRUE всегда на первом такте работы компонента, затем она равна FALSE.

Пункт меню **"ЛОКАЛ"** используется для создания ссылок на переменные панели. Если на диаграмму блока поместить такую ссылку, то в ее контекстном меню можно выбрать любую переменную панели. В контекстном меню можно установить направление использования ссылки: на чтение или на запись. Если это ссылка на входную или выходную переменную блока, то направление использования переменной задается в самой переменной.

4.8.1.11 Функции для работы со временем

В этом разделе рассматриваются функции для работы со временем, которые доступны из меню <Функции\Дата\Время>.



Некоторые функции работы со временем используют структуру времени, содержащую следующие элементы:

- SHORT sec, // секунды (0-59)
- SHORT min, // минуты (0-59)
- SHORT hour, // часы (0-23)
- SHORT mday, // день месяца (1-31)
- SHORT month; // месяц (0-11)
- WORD year; // год
- BYTE wday; // день недели (0-6, Суббота - 0)
- WORD yday; // день года (0-365)
- BYTEBOOL isdst; // признак летнего времени

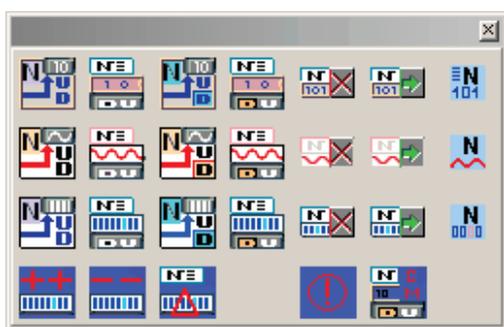
	<p>Счетчик миллисекунд Возвращает значение счетчика миллисекунд. Выход – 4 байтовое беззнаковое целое (DWORD). Время начала отсчета счетчика неопределено, то есть вы не можете преобразовать значение счетчика в реальные дату/время. Если из меньшего числа вычитается большее, то результат – сумма меньшего числа и $2^{32}-1$ – большее число. Если в результате сложения получается число, большее, чем $2^{32}-1$, то результатом сложения будет это число - 2^{32}.</p>
	<p>Дата/время в секундах Возвращает текущее административное время в секундах, начиная с 1 января 1980 года. Тип возвращаемого значения DWORD. Число секунд может отличаться от реального числа прошедших секунд в течение летнего времени.</p>
	<p>Дату/время в секунды Переводит структуру времени в количество секунд прошедших с 1980 года. В структуре используются следующие поля: sec(секунды), min(минуты), hour(часы), mday(день месяца), month(месяц), year(год). .Остальные поля игнорируются.</p>
	<p>Секунды в дату/время Функция переводит административное число секунд, прошедших с 1 января 1980 года в структуру времени. Если на вход ничего не подано, то будет взято текущее административное время. Для получения отдельных элементов структуры времени удобно пользоваться блоком <Элементы структуры по имени>. Если на вход подано число секунд из промежутка перевода времени с летнего на зимнее, то в этом случае невозможно определить значение флага isdst, поэтому он искусственно устанавливается в FALSE. Если на вход ничего не подано, то флаг isdst устанавливается правильно.</p>
	<p>Дата/время в секундах(зима) Возвращает текущее время в секундах, начиная с 1 января 1980 года. Тип возвращаемого значения DWORD.</p>
	<p>Секунды в дату/время(зима) Функция переводит число секунд, прошедших с 1 января 1980 года в структуру времени. Если на вход ничего не подано, то будет взято текущее время. Для получения отдельных элементов структуры времени удобно пользоваться блоком <Элементы структуры по имени>.</p>

4.8.1.12 Функции для работы с элементами баз

Операции чтения, записи, взятия и установки аналогичны для всех элементов баз и различаются только типами значений элементов: WORD для дискретов, FLOAT для аналогов и DWORD для счетчиков.

Для всех рассматриваемых блоков действует следующее правило: номер элемента базы равен значению, поданному на вход "Номер", если терминал "Номер" соединен и значению, заданному в конфигурационной таблице, если терминал "Номер" не соединен. В случае несоединенного терминала блок должен иметь метку, которая появится в конфигурационной таблице.

Работа с функциями для работы с элементами баз осуществляется через меню **Функции/Глобальные базы**:



Перечень функций:

 дискрет  аналог  счетчик	<p>ЧТЕНИЕ ЭЛЕМЕНТА ОДНОЙ ИЗ ТРЕХ БАЗ (дискрета, аналога, счетчика).</p> <p>Эти блоки имеют два входа "Номер" (DWORD) и "Значение неопр." (тип значения элемента). Четыре выхода: "Значение", "Код ошибки" (WORD), "Динамика" (BYTEBOOL) и "Неопред." (BYTEBOOL). Терминалы "Значение неопр." и "Значение" имеют тип значения элемента базы. Выход "Код ошибки" возвращает код ошибки. Значение этого выхода, равное 0, соответствует достоверному значению элемента базы. Выход "Неопред." равен TRUE, если значение не достоверно, что равносильно значению выхода "Код ошибки", не равному 0. Выход "Динамика" равен TRUE, если установлен бит динамики. Если значение не определено ("Код ошибки" не равен 1), то на выход "Значение" будет подано значение со входа "Значение неопр." Если в этом случае вход "Значение неопр." не связан, то на выходе "Значение" будет 0.</p>
 дискрет  аналог  счетчик	<p>ЗАПИСЬ ЭЛЕМЕНТА</p> <p>Блок имеет только входы: "Номер", "Значение", "Ошибка", "Динамика". Обязателен для соединения только вход "Значение".</p> <p>Если вход "Динамика" не связан, то он равен FALSE.</p> <p>Если вход "Ошибка" не связан, то он равен 0, что соответствует достоверному значению.</p> <p>Если на вход "Ошибка" подается 1, то будет записано недостоверное значение с кодом ошибки 0x0D80 - "Ошибка выставлена алгоритмом".</p> <p>Если на вход "Ошибка" подается значение больше 1, то будет записано недостоверное значение с кодом ошибки равным значению на входе "Ошибка".</p> <p>После записи элемента бит Динамики в базе не изменится, если вход "Динамика" равен FALSE, и установится в TRUE, если вход "Динамика" равен TRUE.</p>
	<p>ВЗЯТИЕ ЭЛЕМЕНТА</p>

 дискрет  аналог  счетчик	<p>Блок аналогичен блоку "Чтение элемента" за исключением того, что при чтении происходит сброс бита динамики, то есть если "Динамика" равна TRUE, то при последующем чтении из этого же элемента "Динамика" равна FALSE (конечно если между этими чтениями кто-то не установил бит Динамики).</p>
 дискрет  аналог  счетчик	<p>УСТАНОВКА ЭЛЕМЕНТА</p> <p>Блок аналогичен блоку "Запись элемента", за исключением того, что если вход "Динамика" равен FALSE, то при записи элемента производится автоматическое формирование бита динамики. Бит динамики будет установлен: для дискрета - если значение в базе равно 0, а записываемое значение равно 1; для аналогов и счетчиков - если значение в базе и записываемое определены и не равны.</p>

Максимальное значение дискрета равно 0x3FFF или 16383.

Имеются также операции по блокированию и разблокированию элементов баз:

<p>БЛОКИРОВАНИЕ</p>  дискрет  аналог  счетчик	<p>РАЗБЛОКИРОВАНИЕ</p>  дискрет  аналог  счетчик
---	--

Задание номера или имени по тем же правилам, что и для остальных блоков. Блокирование означает, что другие компоненты не могут писать в заданный элемент, разблокирование снимает это ограничение. Если при блокировании элемента другой компонент пытается записать в заблокированный элемент, то в журнал ошибок контроллера помещается соответствующая запись.

	<p>Инкремент значения счетчика</p> <p>- на вход подается значение счетчика и величина, на которую его нужно увеличить. Сложение происходит с учетом десятичной коррекции.</p>
	<p>Декремент значения счетчика</p> <p>- на вход подается значение счетчика и величина, на которую его нужно уменьшить. Вычитание происходит с учетом десятичной коррекции.</p>
	<p>Инкремент счетчика</p> <p>- блок имеет два входа: номер счетчика и значение, на которое его нужно увеличить. Номер счетчика в базе определяется по тем же правилам, что и у блоков "чтение счетчика" и "запись счетчика".</p>

	<p>Фатальная ошибка</p> <p>Блок имеет два входа – код ошибки и аргумент. Оба имеют тип WORD. При выполнении этой функции контроллер перестартовывает. После рестарта в журнале ошибок появляется запись с данными кодом ошибки и аргументом.</p>
	<p>Установка дискрета с временем</p> <p>Работает аналогично блоку "Установка дискрета". Добавлено еще два входа – секунды и миллисекунды. Тип обоих входов – DWORD. Вход "миллисекунды" необязателен. По умолчанию на него подается "0". На вход "секунды" подается время в секундах, начиная с 1 января 1980 года.</p>
 "Номер дискрета"  "Номер аналога"  "Номер счетчика"	<p>Номер элемента</p> <p>Возвращает номер элемента, имя которого задано в метке блока. При использовании в проекте блока "Номер элемента" с заданным именем в конфигурации компонента появляется выходной элемент с таким именем. Блок необходим, когда требуется программно узнать номер именованного элемента, например, для дальнейшего пересчета номера.</p> <p>Выход: WORD по; номер элемента в базе.</p>

Контекстное меню блоков, имеющих вход "номер", содержит пункт "необязательный". Этот пункт работает только для блоков, номер которых задается в конфигурационной таблице. Этот пункт относится к номеру элемента. Если блок необязательный, (то есть в контекстном меню напротив пункта "необязательный" стоит галочка), то в конфигурационной таблице для этого элемента можно задать 0. Если же блок обязательный (галочки нет), то номер этого элемента не может быть равен 0, так как в этом случае при старте алгоритма в журнал ошибок будет помещено соответствующее сообщение об ошибке, и контроллер перейдет в [минимальный режим](#) работы.

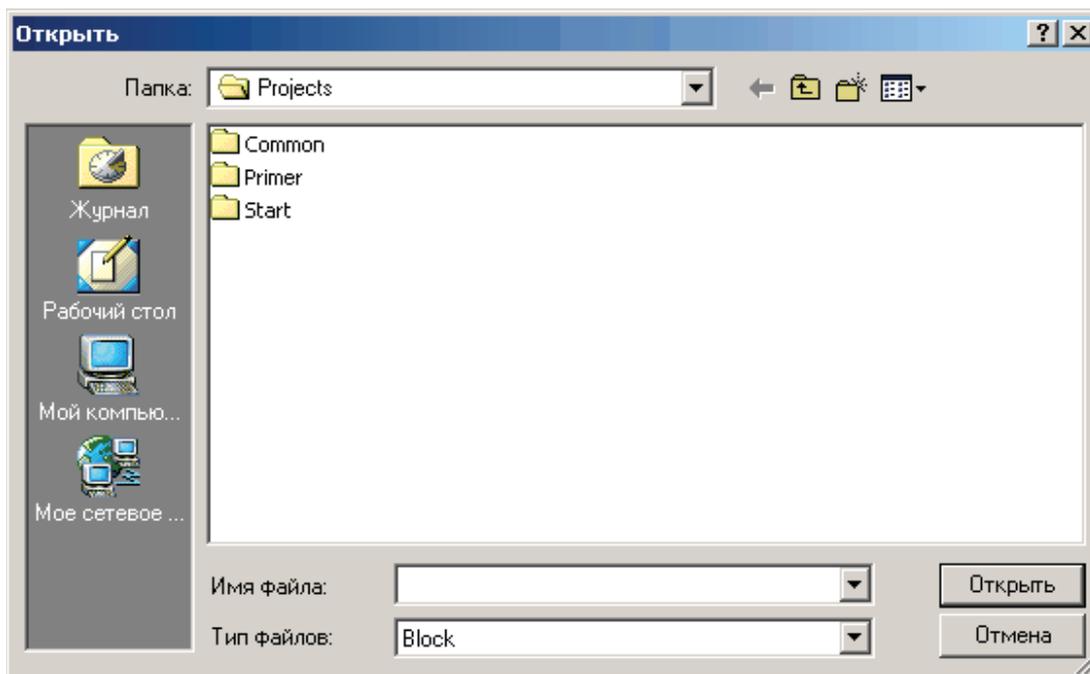
Для всех элементов баз в проекте существует [сводная таблица](#), в которой можно посмотреть, является ли элемент обязательным. Если в проекте есть несколько блоков с одинаковым именем элемента (то есть соответствующих одному элементу базы), то такой элемент будет считаться обязательным, если хотя бы один из блоков обязательный.

4.8.1.13 Выбор функции из файла

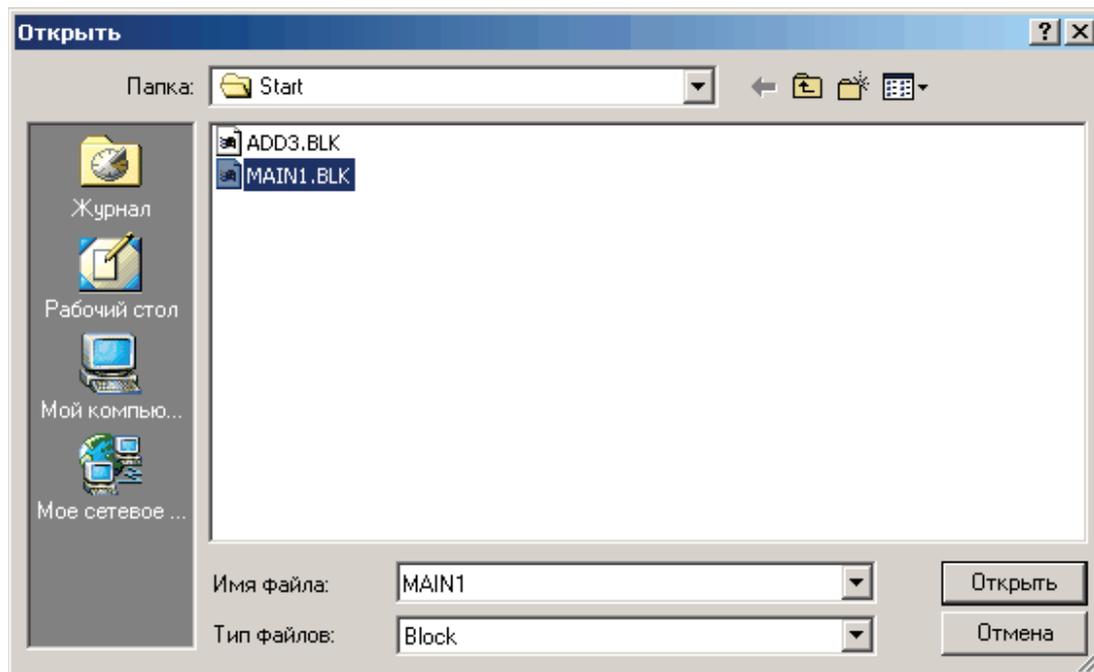


Выбор функции из файла осуществляется через меню <Функции/ Выбор из файла...>.

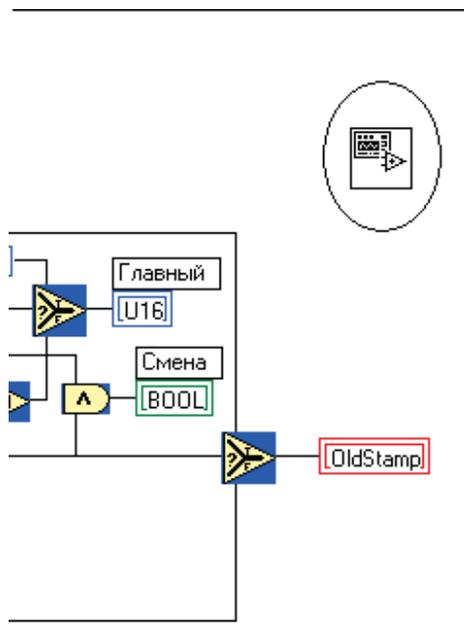
Можно выбрать готовый блок из имеющихся. Открывается окно выбора:



Укажите на проект, а в проекте на блок, который вы хотите выбрать:



Далее выбираете блок и указываете место (мышью) на диаграмме, куда выхотите поместить этот блок:



Его иконка по умолчанию стандартная- , но имеется возможность выбирать иконки для блоков при создании блока (см. "Панель блока").

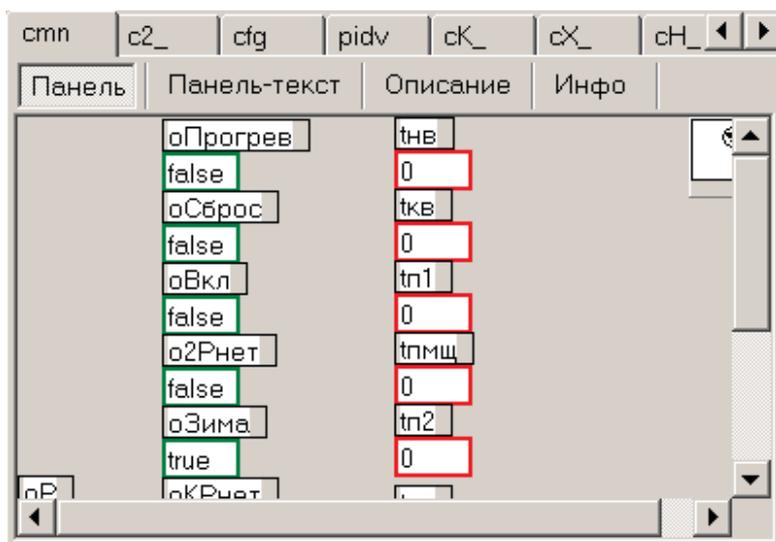
4.9 Глобальные переменные

Глобальные переменные задаются в отдельном файле.

Для создания файла глобальных переменных нужно вызвать пункт меню Файл\Новая глоб. переменная.

При этом будет создан новый файл, содержащий панель, но не содержащий диаграммы.

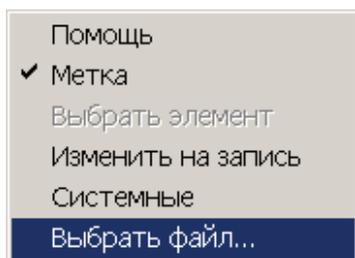
На панели можно разместить переменные и задать их характеристики. Выбор переменных производится из меню кнопки "Переменные".



Глобальные переменные доступны из любого блока как на чтение так и на запись.

Для использования глобальной переменной на диаграмме нужно добавить терминал "Глобальная переменная" из меню "Функции\Инструкции".

Затем из контекстного меню терминала выбрать файл глобальных переменных:



После этого будет доступен пункт меню "Выбрать элемент", из которого можно выбрать конкретную переменную из файла.

Кроме пользовательских, есть еще **системные глобальные переменные**.

Они задаются выбором пункта "Системные". После этого в подменю "Выбрать элемент" будут две переменные:

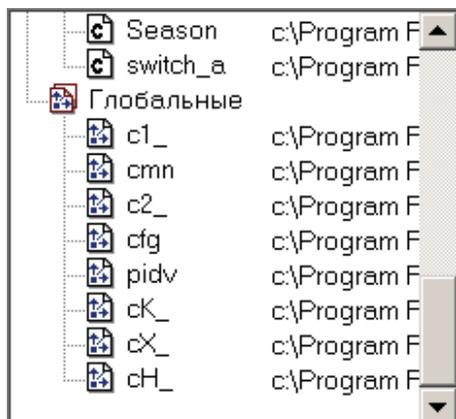
"Первый старт" и "Рестарт". Обе переменные типа BYTEBOOL.

На первом такте работы алгоритма переменная "Первый старт" равна TRUE, если не удалось прочесть сохраненные переменные.

Переменная "Рестарт" на первом такте равна TRUE.

На следующих тактах обе переменные равны FALSE.

В окне проводника файлы глобальных переменных, использующихся в проекте, отображаются отдельным списком:



Глобальные переменные можно создавать не только графически, но и на языке С. Как это делать описано в разделе "[Работа с С-Текстом](#)".

4.10 Описание дополнительных блоков

Имеется набор готовых блоков, созданных пользователями программы "Разработчик". Набор блоков постоянно пополняется.

4.10.1 Панель Триггеры

Содержит 5 блоков: SR, JK, T, D и ждущий триггеры. Все блоки после перезапуска контроллера устанавливаются в начальное состояние. 

SR-триггер



Простейший элемент памяти с двумя состояниями. Выполняет операцию <Значение = NOT(Сбросить) AND (Установить OR Значение)>, где Значение' обозначает предыдущее состояние выхода Значение.

Терминал	Тип	Назначение	Тип данных
установить	вход	устанавливает триггер в true, если нет сигнала "сбросить"	BYTEBOOL
сбросить	вход	сбрасывает триггер в false	BYTEBOOL
значение	выход	возвращает состояние триггера	BYTEBOOL

JK-триггер

Универсальный триггер. Выполняет операцию <Значение = (Значение') ? NOT (Сбросить) : (Установить)>, где Значение' обозначает предыдущее состояние выхода Значение. Таким образом, при поступлении сигнала только на один из входов JK-триггер работает как SR-триггер, а при поступлении сигнала сразу на оба входа – инвертирует свое состояние, т.е. работает как T-триггер.

Терминал	Тип	Назначение	Тип данных
установить	вход	устанавливает триггер в true, одновременно со входом сбросить – инвертирует состояние триггера.	BYTEBOOL
сбросить	вход	сбрасывает триггер в false, одновременно со входом установить – инвертирует состояние триггера.	BYTEBOOL
значение	выход	возвращает состояние триггера	BYTEBOOL

T-триггер

Переключающий триггер. Выполняет операцию <Значение = (Переключить != Значение')>, где Значение' обозначает предыдущее состояние выхода Значение.

Терминал	Тип	Назначение	Тип данных
переключить	вход	инвертирует состояние триггера	BYTEBOOL
значение	выход	возвращает состояние триггера	BYTEBOOL

D-триггер

"Защелка" для значения входа Данные. Всегда возвращает предыдущее запомненное значение входа Данные, которое обновляется при поступлении импульса Выбор. Таким образом, новое значение отстает на 1 такт от импульса Выбор и

удерживается до поступления нового импульса Выбор плюс еще 1 такт.

Терминал	Тип	Назначение	Тип данных
данные	вход	запоминаемое логическое значение	BYTEBOOL
выбор	вход	команда запомнить значение входа "данные". Необязательный, по умолчанию всегда активен и в этом случае блок работает как линия задержки на 1 такт	BYTEBOOL
значение	выход	возвращает предыдущее значение	BYTEBOOL

Ждущий триггер



Ждущий триггер при поступлении сигнала Взвод переходит в состояние ожидания – взводится. В таком состоянии импульс на входе Пуск вызывает появление импульса на выходе Значение и сброс состояния ожидания. Для сброса состояния ожидания без формирования выходного импульса служит вход Сброс.

Терминал	Тип	Назначение	Тип данных
взвод	вход	взводит триггер в состояние ожидания сигнала пуск	BYTEBOOL
пуск	вход	спуск триггера (импульс на выходе и сброс триггера).	BYTEBOOL
сброс	вход	снимает взвод триггера (без импульса на выходе)	BYTEBOOL
значение	выход	активизируется при поступлении сигнала пуск на взведенный триггер	BYTEBOOL

4.10.2 Панель Регуляторы

Содержит блоки релейного и ПИД (в позиционной форме и форме приращений) регулирования; фильтры скользящего среднего и экспоненциальные (1-го и 2-го порядков); зоны нечувствительности; насыщения; широтно-импульсной модуляции (ШИМ):



Релейный регулятор

Релейный регулятор служит для 2-х позиционного управления с гистерезисом с автоматическим выбором условий сравнения по значениям уставок включения и выключения. Если вход "Уставка ВКЛ" больше входа "Уставка ВЫКЛ", выход Включить устанавливается в TRUE, как только вход Значение становится больше или равен входу "Уставка ВКЛ" и удерживается таким до тех пор, пока вход Значение не станет меньше или равен входу "Уставка ВЫКЛ". Если же, наоборот, "Уставка ВЫКЛ" больше, чем "Уставка ВКЛ", то выход Включить устанавливается, как только вход Значение становится меньше или равен входу "Уставка ВКЛ" и сбрасывается, как только вход Значение станет больше или равен входу "Уставка ВЫКЛ".

Терминал	Тип	Назначение	Тип данных
значение	вход	регулируемая величина	FLOAT
уставка вкл	вход	уставка включения	FLOAT
уставка выкл	вход	уставка выключения	FLOAT
включить	выход	устанавливается, если параметр перешел через уставку включения и снимается при переходе уставки выключения. Условия сравнения определяются автоматически по соотношению уставок.	BYTEBOOL



ПИД-регулятор в позиционной форме

Блок реализует алгоритм ПИД-регулятора в позиционной форме с защитой от насыщения и возможностью принудительной инициализации и безударного включения.

При активном входе Пуск величина управляющего воздействия (выход Значение) устанавливается равной величине параметра Y_s , которая для безударного включения должна соответствовать текущему положению исполнительного механизма. Также при пуске регулятора производится инициализация интегральной составляющей, чтобы она соответствовала пусковой величине управления. Величина Y_s и, следовательно, пусковое управление могут выходить из диапазона ограничения управления, заданного параметрами Y_{min} и Y_{max} , – ограничение величины управляющего воздействия не работает во время пуска. В остальное время величина управления ограничена, а величина интегральной составляющей скорректирована для предотвращения интегрального насыщения. Вычисление величины управления производится периодически с тактом, заданным параметром T_s , начиная с момента пуска. Вычисление дифференциальной составляющей производится с использованием аппроксимации первого порядка с постоянной времени слежения, нормированной относительно постоянной времени дифференцирования $T_f = T_d / N_f$. При задании $N_f = 0$ используется простая разностная аппроксимация.

Терминал	Тип	Назначение	Тип данных
значение	вход	регулируемая величина	FLOAT

Вход, при | Вход - Смещение | > -Точность.

Терминал	Тип	Назначение	Тип данных
вход	вход	обрабатываемая величина	FLOAT
точность	вход	полуширина зоны нечувствительности	FLOAT
смещение	вход	центр зоны нечувствительности, необязательный, по умолчанию 0	FLOAT
выход	выход	преобразованное значение	FLOAT



Фильтр скользящего среднего

Блок вычисления скользящего среднего по заданному (от 1 до 5) количеству точек усреднения. Всегда сохраняет 5 предыдущих значений фильтруемой величины, что позволяет динамически изменять количество точек усреднения во время работы. После первоначального запуска, пока не набрано требуемое количество значений, усреднение производится по актуальным значениям.

Терминал	Тип	Назначение	Тип данных
вход	вход	фильтруемая величина	FLOAT
кол-во точек	вход	количество точек усреднения	WORD
выход	выход	отфильтрованное значение	FLOAT



Экспоненциальные фильтры 1-го и 2-го порядков

Экспоненциальный фильтр – авторегрессивный фильтр скользящего среднего. Можно интерпретировать как астатическое звено соответствующего порядка.

Терминал	Тип	Назначение	Тип данных
вход	вход	фильтруемая величина	FLOAT
период	вход	постоянная времени фильтра	FLOAT
выход	выход	отфильтрованное значение	FLOAT



Линейная зависимость с насыщением

Вычисление кусочно-линейной зависимости $Y(X)$, заданной двумя краевыми точками. Вне диапазона $[X1; X2]$ выход $Y(X)$ равен значению ординаты соответствующей краевой точки. Если $X1=X2$, то $Y(X=X1) = (Y1+Y2)/2$.

Терминал	Тип	Назначение	Тип данных
x	вход	абсцисса	FLOAT
x1	вход	абсцисса 1-ой точки	FLOAT
y1	вход	ордината 1-ой точки	FLOAT
x2	вход	абсцисса 2-ой точки	FLOAT
y2	вход	ордината 2-ой точки	FLOAT
y(x)	выход	значение кусочно-линейной функции заданной краевыми	FLOAT

точками (x1, y1) и (x2, y2).



Широтно-импульсная модуляция (ШИМ)

Блок широтно-импульсной модуляции трактует модуль входного значения как задание длительности импульса в секундах, а знак – как направление управления. Положительные величины формируют импульсы на выходе Увеличить, отрицательные – на выходе Уменьшить. Модуль входа "Значение" преобразуется в значение дискрета управления (если больше 8 секунд – на выходе устанавливается в 1, иначе длительность импульса в миллисекундах) и подается на выбранный выход. Выход "Управлять" постоянно активен (TRUE).

Терминал	Тип	Назначение	Тип данных
значение	вход	величина импульса (модуль – длительность, знак – направление)	FLOAT
управлять	выход	признак необходимости записи управляющих значений и установки бита динамики.	BYTEBOOL
увеличить	выход	значение управляющего дискрета увеличения регулируемой величины	WORD
уменьшить	выход	значение управляющего дискрета уменьшения регулируемой величины	WORD

4.10.3 Панель Вспомогательные



Содержит блоки вычисления кванта (промежутка) времени, таймаута, генератора импульсов и таймера.



Квант времени.

Блок возвращает промежуток времени между моментами выполнения данного блока на текущем такте компонента и предыдущим:

Следует учитывать, что при условном выполнении этого блока возвращаемое значение не совпадает с тактом компонента, а равно промежутку времени с предыдущего исполнения этого же блока.

Терминал	Тип	Назначение	Тип данных
квант	выход	длительность промежутка времени между предыдущим и текущим вызовом данного блока.	FLOAT



Таймаут

Блок отсчитывает момент наступления таймаута. Отсчитываемое время задается по входу Период и может динамически изменяться. Отсчет начинается с такта, в котором вход Считать становится активным. На такте, когда остаток времени (выход Осталось) станет меньше или равен нулю, выход Таймаут активизируется и удерживается таким до тех пор, пока вход Считать активен. При входе Считать = FALSE, выход Таймаут = FALSE и выход Осталось = 0.

Терминал	Тип	Назначение	Тип данных
считать	вход	признак контроля таймаута. таймаут отсчитывается от момента перехода этого сигнала из 0 в 1.	BYTEBOOL
период	вход	длительность таймаута	FLOAT
таймаут	выход	признак истечения таймаута	BYTEBOOL
осталось	выход	остаток времени до наступления таймаута	FLOAT



Метроном (генератор импульсов)

Блок периодически (с тактом, заданным по входу Период) выдает импульсы на выход Тик с момента перехода входа Считать в активное состояние. Остаток времени до следующего тика подается на выход Осталось. При входе Считать = FALSE, выход Таймаут = FALSE и выход Осталось = 0.

Терминал	Тип	Назначение	Тип данных
считать	вход	сигнал включения метронома	BYTEBOOL
период	вход	период следования импульсов	FLOAT
тик	выход	импульсы (единичной длительности) метронома	BYTEBOOL
осталось	выход	остаток времени до очередного импульса	FLOAT



Таймер

Блок реализует таймер обратного отсчета с возможностью приостановки и произвольного запуска. В начальном состоянии таймер выключен, на выходе Осталось – 0. По поступлению импульса на вход Пуск или С_начала (в выключенном состоянии они равнозначны) таймер инициализируется (выход Осталось становится равным входу Период) и переходит в состояние "идет отсчет". В этом состоянии выход Осталось уменьшается на время, истекшее с предыдущего такта. При идущем отсчете по поступлению импульса на вход Пауза таймер переходит в состояние "отсчет остановлен" и выход Осталось не изменяется. По поступлению импульса Пуск таймер возвращается в режим отсчета. По истечении времени таймер переходит в состояние "время истекло" и на выходе Осталось – 0. В любом состоянии импульс на входе С_начала вызывает инициализацию таймера и переход в режим отсчета времени, а импульс на входе "Стоп" переводит таймер в режим "выключен". Значение на входе Период может изменяться динамически, что может привести к переходу из состояния остановленного отсчета в состояние "время истекло", если до паузы прошло больше времени, чем задано по входу Период.

Терминал	Тип	Назначение	Тип данных
пуск	вход	пуск выключенного/ остановленного таймера	BYTEBOOL
пауза	вход	остановка таймера	BYTEBOOL
стоп	вход	выключение таймера	BYTEBOOL

с начала	вход	перезапуск таймера (отсчет начинается сначала)	BYTEBOOL
период	вход	длительность отсчитываемого промежутка времени	FLOAT
этап	выход	состояние таймера: dts_halt = 0 – выключен dts_wait = 1 – идет отсчет dts_pause = 2 – отсчет остановлен dts_done = 3 – время истекло	WORD
осталось	выход	обратный отсчет времени	FLOAT

4.11 Общие замечания к написанию алгоритмов

Точность вычисления интервалов времени

Точность вычисления интервалов времени ограничена длительностью такта исполнения пользовательского компонента – она не может быть лучше.

По этой же причине, когда говорится о такте вычислений (например, для ПИД-регулятора) или о периоде (например, для таймаута) следует понимать, что этот промежуток времени, скорее всего, будет несколько больше заказанного. Поэтому практически везде вместо заданного периода используется счетчик реально прошедшего времени.

Точное формирование импульсов управления

Так как модули дискретного вывода могут формировать импульсы длительностью не более 8191 мсек., для выдачи более длительных импульсов следует записать в дискрет 1, а когда до конца импульса останется менее 8 сек. – записать этот остаток (и установить бит динамики). Естественно, это работает только тогда, когда такт компонента и логика программы (при условном выполнении блоков формирования длительности импульса или записи управляющих дискретов) позволяют измерять промежутки времени менее 8 сек.

Использование блоков внутри цикла

Сейчас простое использование блоков с внутренними (и сохраняемыми) переменными, а таковы практически все описанные выше блоки, в циклах невозможно, так как отдельные копии таких переменных не создаются для каждой итерации. Для выхода из такой ситуации есть два способа. Первый – это написать аналог блока, в котором внутренние (и сохраняемые) переменные будут описаны массивами требуемой размерности, и на вход блока будет подаваться номер итерации для выбора нужных экземпляров переменных. Второй – написать аналог блока, который будет принимать на входе текущие значения внутренних/сохраняемых переменных для данной итерации и будет возвращать их измененные значения для сохранения их к следующему такту (скорее всего опять же в массиве). В обоих случаях эффективно использовать С-блок и описание переменных и функции блока для организации требуемого метода доступа к структуре внутренних (сохраняемых) переменных без повторного программирования логики.

Быстрые клавиши

При рисовании диаграммы удобно пользоваться быстрыми клавишами Ctrl+1(режим расположения) и Ctrl+2(режим соединения)

Для удаления несоединенных проводов используется комбинация Ctrl+B.

Для копирования, вставки и удаления используются стандартные комбинации Ctrl+C, Ctrl+V, Del. lmk I

4.12 Связь программ 'Конфигуратор' и 'Разработчик'

Для написания алгоритмов, работающих в контроллере Деконт-182, используется программа "Разработчик".

Для того, чтобы написанный в "Разработчике" алгоритм стал компонентом программы "Конфигуратор", из программы

"Разработчик" производят его (будущего компонента) построение. При построении первого компонента программы "Разработчик" в папке C:\Program Files\DEP\DB создается папка "**Components**", в которой с этого момента хранятся папки с описанием всех компонентов программы "Разработчик", и происходит регистрация этого компонента в программе "Конфигуратор". После регистрации компонент вносится во все справочники программы, и его уже можно, как и другие компоненты, добавлять в любые контроллеры любого проекта.

Каталоги компонентов в папке "**Components**" называются по имени соответствующего проекта программы "Разработчик". В составе папки – также одноименный с папкой файл с расширением .xml и библиотеки компонента.

При построении компонентов, зарегистрированных в программе "Конфигуратор" ранее, но измененных, происходит **обновление** файлов в папке "Components", с учетом изменений. Для того, чтобы отразить эти изменения в конфигурации контроллера, в окне контроллера есть кнопка "Обновить":



При **переустановке ПО** набор зарегистрированных прикладных компонентов в программе "Конфигуратор" сохраняется.

Для переноса компонентов на другой компьютер без использования программы "Разработчик" необходимо вручную скопировать папку "Components" вместе с содержимым в то же место на другом компьютере. Далее воспользоваться пунктом меню главного окна программы "Конфигуратор": "Пользователь\Прикладные компоненты\Установить", указав на файл с расширением xml, входящий в состав папки компонента.

4.13 Платформы построения

Разработанный компонент может быть построен для разных платформ: Decont182, Windecont и DecontA9.

Но при разработке алгоритма необходимо учитывать особенности платформ:

• Decont182

1. размер библиотек не должен превышать 8192 байта. Если размер превышен, то после построения будет выдано предупреждающее сообщение. В этом случае необходимо разбить "большую" библиотеку - добавить новую библиотеку и перенести в нее часть блоков из данной;
2. не рекомендуется создавать массивы, размер которых достигает нескольких килобайт из-за ограничений работы с памятью;
3. при разработке больших компонентов может оказаться недостаточно стека. Эта ситуация может быть обнаружена при включенной галочке "Контроль стека" (Устанавливается в программе Конфигуратор, в окне контроллера, во вкладке Системные параметры), тогда контроллер рестартует в Минимальный режим и в журнале ошибок появляется сообщение "Превышена красная граница стека" с указанием компонента, в котором произошла ошибка. В этом случае нужно увеличить размер стека в свойствах проекта и пересобрать компонент.

• WinDecont

1. никакой блок не может иметь имя "main";
2. при загрузке компонента в контроллер необходимо записать только его конфигурацию. Библиотеки (исполняемый код) загружать не нужно. Они загружаются в WinDecont при обновлении;
3. нет ограничений на размер библиотек и объем данных.

• DecontA9

1. нет ограничений на размер библиотек и объем данных;
2. при сборке компонента каталог вывода должен быть установлен на одном диске с каталогом установки программы Разработчик.

4.14 Примеры работы в программе

Лучший способ научиться программировать в среде программы "Разработчик" – написать алгоритм. В разделе "[Порядок работы с проектом](#)" показаны дальнейшие действия для построения простейшего алгоритма и последующего запуска его в контроллере.

Написание алгоритма рассмотрим на следующем примере:

Из дискрета DIN читается значение. Если оно равно 0, то в дискрет DOUT записывается 2, в противном случае в DOUT записывается 3.

1. Загрузите программу "Разработчик"
2. Создайте новый блок из меню **<Файл\Новый блок>**. В редакторе появятся две вкладки: "Панель" и "Диаграмма".
3. Активируйте "Диаграмму". Диаграмма является тем окном, где располагаются блоки, составляющие логику алгоритма.
4. Из панели инструментов главного окна нажмите на кнопку **<Функции>**. Появится панель-меню выбора функций. При наведении на элемент в заголовке меню появляется его имя. Выберите подменю **<Глобальные базы>**. В панели "Глобальные базы" выберите функцию **<Чтение дискрета>**
5. Переместите курсор мыши на Диаграмму (курсор будет иметь вид, как при операции DragAndDrop). После щелчка мышью на поверхности Диаграммы появится блок с иконкой, такой же, как и на панели выбора. Из той же панели выбора выберите блок запись дискрета и аналогичным образом поместите его на диаграмму. Для алгоритма нам понадобятся еще несколько элементов. Из панели "Функции" выберите подменю **<Сравнение>**, из которого поместите на диаграмму функцию **<Выбор из двух переменных>** и **<Равно нулю>**. А из подменю **<Численные>** поместите на диаграмму два элемента **<Целые константы>**.
6. Расположите (в режиме "Расположение") блоки на диаграмме так, как показано на рисунке:

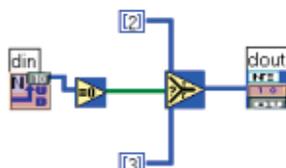


7. Перейдите в режим "Связывание".

При перемещении мыши над блоками их части начинают мерцать и появляются подсказки. Это входные и выходные терминалы, или просто входы и выходы. Будем именовать терминалы по появляющимся над ними подсказкам. Соедините выход "значение" блока "чтение дискрета" с входом "х" блока "Равно нулю", выход блока "Равно нулю" с входом "s" блока "Выбор из двух переменных", вход "t" которого соедините с верхней константой, а вход "f" с нижней. Выход "s? t:f" этого же блока соедините с входом "значение" блока запись дискрета. Для соединения терминалов нужно сделать щелчок мышью на одном и отпустив кнопку мыши (при этом появится провод), щелкнуть мышью на втором терминале. Для отмены проводки провода нужно щелкнуть мышью на исходном терминале.

8. Перейдите в режим "Редактирование" (кликнуть левой кнопкой мыши на названии метки блока: например, dep_DiscretRead).

В этом режиме можно менять численные значения и метки элементов. При нажатии мышью на константе появится окно редактирования, в котором нужно ввести нужное значение. Вызовите всплывающее меню блока "Чтение дискрета" нажатием правой кнопкой мыши на блоке и выполните команду **<Показать\Метку>**. Щелкнув на метке, измените ее на **din**. Аналогично поступите с блоком "Запись дискрета". Должно получиться, как на рисунке:



9. Проверьте правильность алгоритма, нажав на кнопку **<Проверить>** на панели инструментов главного окна.

Если все верно, то окно сообщений будет пустым, и провода будут окрашены в цвета, соответствующие типам данных, проходящих по ним при выполнении. В противном случае проверьте правильность соединений.

10. Создайте свой каталог (например, Test): Корневой каталог программы "Разработчик" \Projects\Test.

11. Запишите в него созданный блок выбором меню **<Файл\Записать как...>**, например, под именем simple.blk.

Таким образом, мы реализовали алгоритм и создали проект. В главе "[Порядок работы с проектом](#)" подробно описано, какие действия нужно произвести далее, чтобы алгоритм (состоящий на данный момент из блока simple.blk) стал сначала [проектом](#) программы "Разработчик", а затем компонентом программы "Конфигуратор".

4.15 Возможные ситуации

После старта контроллера он работает некоторое время, но затем падает в минимальный режим с сообщением прикладного компонента **"##VTABLE: Превышен номер элемента"**. Это значит, что была запись или чтение из элемента базы (дискрета, аналога или счетчика) с номером, превышающим максимальный для данной конфигурации. Нужно проверить значение номеров в таблицах конфигурации прикладного компонента и количество элементов баз в конфигурации Системной задачи.

4.16 Сообщения и ошибки

"Программа Конфигуратор не установлена должным образом." - для устранения ошибки нужно запустить программу "Конфигуратор" с параметром /regserver.

4.17 Ошибки диаграммы

Ошибки :

"Провод не соединен" - провода на диаграмме должны быть соединены.

"Провод имеет несколько источников" - ошибка возникает, когда с входным терминалом связаны несколько проводов.

"Несовпадение типов" - типы переменных связанных проводом, таковы, что невозможно выполнить преобразование переменной - источника в переменную - приемник.

"Терминал блока не имеет источника" - входной терминал объявлен как обязательный для подсоединения и поэтому нужно соединить его с другим терминалом.

"Обнаружен цикл" - ошибка не указывает на элемент диаграммы, а объявляет что существует такая последовательность проводов и терминалов, для которой нельзя определить начало и конец.

"Неверны типы у терминалов блока" - процедура проверки

```
case WIRE_CONNECTEDUNDEF :
    return "Провод соединен с неопределенным терминалом";
case BLOCKTERM_MULTISOURCE :
    return "Терминал блока имеет более 1 источника";
case TERM_MULTISOURCE :
    return "Терминал имеет более 1 источника";
case WIRE_INCYCLE :
    return "Провод принадлежит циклу";
case WIRE_ERROR :
    return "Ошибка в соединении провода";
case STRUCTTERM_MULTISOURCE :
    return "Терминал структуры имеет более 1 источника";
case TERM_HAVENOSOURCE :
```

```
    return "Терминал не имеет источника";
case BLOCK_BADLINKAGE :
    return "Нарушена связь блока с его определением";
case CONTROL_BADLINKAGE :
    return "Нарушена связь элемента управления с определением типа";
case WIRE_HASNOSOURCE :
    return "Провод не имеет источника";
case TERM_HASNODATA :
    return "Терминал не имеет данных";
case CONTROL_UNDEFINED :
    return "Тип элемента управления не определен";
case ARRAY_DIMENTIONS :
    return "Неверно задан размер массива";
case STR16_IN_REQUIRED :
    return "Входная переменная типа STR16, массив или структура должна быть обязательной";
case TERM_HAVENOSOURCE1 :
    return "Терминал может быть неопределен";
case WIRE_NOTEXACTTYPE :
    return "Не точное совпадение типов";
case TERM_OUT_NOTDIAGRAM :
    return "Использование выходной переменной не на главной диаграмме.";
```