

ЛЕКЦИЯ 11 РАБОТА С ПАКЕТОМ ДИНАМИЧЕСКОГО МОДЕЛИРОВАНИЯ VISSIM 6.0

1. Назначение программы
2. Графический интерфейс VisSim 6.0
3. Принципы построения и редактирования диаграмм
4. Принципы управления моделью и получения результатов моделирования. Пример модели

1.1 Назначение программы

VisSim является аббревиатурой выражения Visual Simulator, т. е. визуальная, воспринимаемая зрением, среда и средство для моделирования. Программа VisSim предназначена для построения, исследования и оптимизации виртуальных моделей физических и технических объектов, в том числе систем управления и их элементов. Используя графические возможности программы, можно создавать модель из виртуальных элементов привычным образом аналогично тому, как если бы строили реальную систему из настоящих элементов. При описании и последующем построении модели в среде VisSim нет необходимости записывать и решать дифференциальные уравнения, программа это сделает сама по заданной структуре системы и параметрам ее элементов. Результаты решения выводятся в наглядной графической форме.

1.2 Графический интерфейс VisSim

Программа VisSim предоставляет исследователю графический интерфейс, позволяющий основную часть работы по созданию модели выполнить с помощью мыши, а параметры элементов ввести с клавиатуры. Другими словами, интерфейс представляет собой интерактивный виртуальный лабораторный стенд, обеспечивающий построение моделей из отдельных блоков, запуск процесса моделирования, управление им и контроль результатов.

После загрузки на экране появляется окно программы, которое включает в себя основные элементы (рисунок 1):

- **Заголовок окна**, в котором указано название программы «VisSim» и название текущей диаграммы. Первоначально никакой диаграммы в VisSim не загружено, поэтому можно видеть чистый лист, название которого по умолчанию – «Diagram1».

- **Строка меню**, состоящего из пунктов «File», «Edit», «Simulate», «Analyze», «Blocks», «Diagrams», «Tutor», «Tools», «Window» и «Help».

Для работы с шрифтом следует выбрать пункт меню **Fonts** (*Шрифты*) для чего необходимо перейти в меню по пути **View** (*Вид*) → **Fonts...** (*Шрифты*). Здесь можно выбрать тип шрифта, его начертание, размер, цвет и кодировку (рисунок 2).

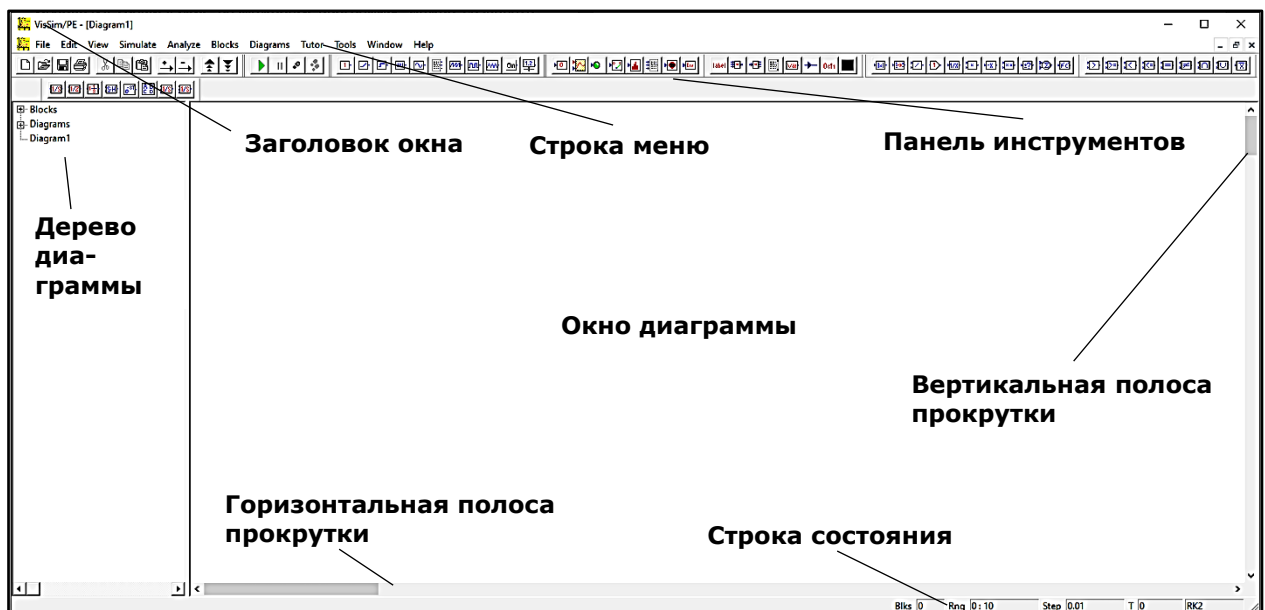


Рисунок 1 – Окно основной программы VisSim 6.0

- **Панель инструментов.** На этой панели расположены кнопки, каждая из которых соответствует какому-либо действию. Например, первая кнопка создает новый пустой лист диаграммы, вторая - позволяет открыть уже имеющуюся на диске диаграмму и т. д. Все эти кнопки дублируют некоторые из команд основного меню и предназначены для обеспечения большего удобства работы с пакетом. На панели инструментов также располагаются некоторые палитры типовых блоков.

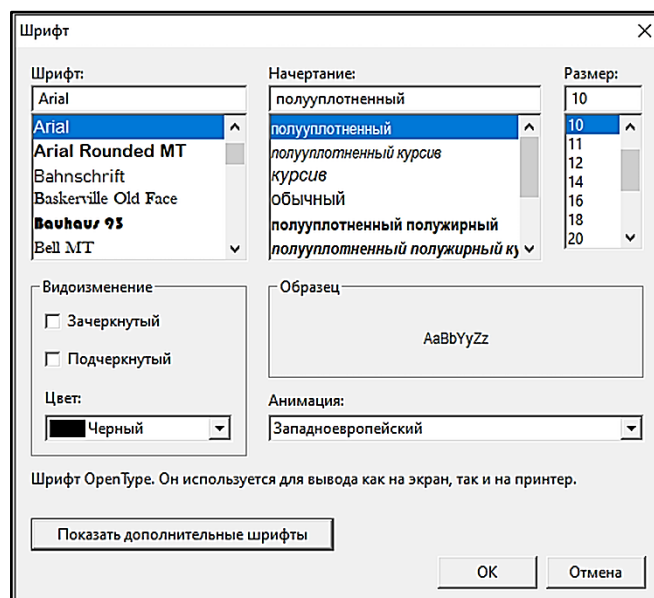


Рисунок 2 – Выбор шрифта

- **Дерево диаграммы** показывает иерархическую структуру блоков диаграммы.

- **Окно диаграммы** – так называемое, рабочее пространство. В этом окне находится текущий лист диаграммы (первоначально он

пуст), в котором строится и корректируется модель, наблюдаются результаты ее работы.

- **Линии прокрутки** справа и внизу окна диаграммы позволяют перемещать видимую область листа для просмотра больших диаграмм.

- **Строка состояния**, на которой отображается информация о текущем состоянии системы:

 - Blks* – указывает число функциональных блоков на диаграмме;

 - Rng* – начальный и конечный моменты времени расчета;

 - Step* – шаг по времени;

 - T(Time)* – текущее время расчета (если диаграмма не запущена на выполнение, то время равно нулю).

 - Также в строке указывается метод интегрирования:

 - Euler* – метод Эйлера;

 - Trapezoidal* – метод трапеций;

 - RK2* – Рунге-Кутта 2-го порядка;

 - RK4* – Рунге-Кутта 4-го порядка;

 - Adaptive RK5* – адаптивный метод Рунге-Кутта 5-го порядка;

 - Adaptive Bulirsh-Stoer*;

 - Backward Euler*.

1.3 Принципы построения и редактирования диаграмм

Под диаграммой в VisSim понимается совокупность связанных между собой, а также автономных функциональных блоков и надписей, помещенных на рабочее пространство и способных функционировать при запуске процесса моделирования. Палитры функциональных блоков находятся в меню **Blocks** (**Блоки**) – рисунок 3 и представляют собой достаточно большой набор основных элементов автоматики. Каждый из блоков моделирует динамические свойства некоторого соответствующего ему простого объекта. Диаграмма может быть сохранена в виде отдельного файла с расширением (.vsm) и, при необходимости, открыта вновь.

Модель VisSim – это часть диаграммы, содержащая виртуальный аналог реальной или проектируемой системы. Диаграмма может содержать несколько моделей. Важным компонентом модели является соединительная линия – виртуальный аналог физического соединения элементов (блоков), передающего воздействия от одного элемента к другому.

Процесс создания диаграммы заключается в выборе мышью соответствующих блоков из палитр и размещении их на листе диаграммы с последующим их соединением связями.

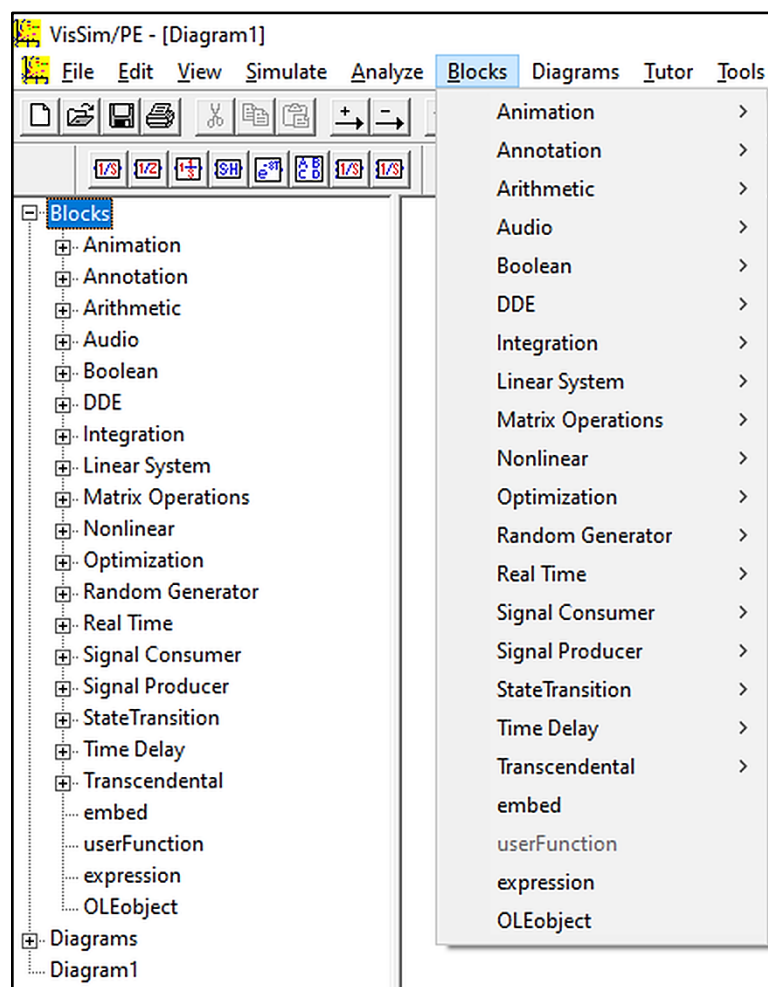


Рисунок 3 – Выбор необходимого блока автоматики

Для **размещения** какого-либо блока на диаграмме необходимо выполнить следующие действия мышью:

- 1) кликнуть на меню **Blocks** (*Блоки*);
- 2) выбрать соответствующую палитру;
- 3) выбрать соответствующий блок и кликнуть на нем мышью.

Если блок продублирован в виде соответствующей кнопки на панели инструментов, достаточно кликнуть по ней мышью. После этого остается кликнуть на том месте диаграммы, куда необходимо поместить блок и он сразу же появится на экране.

Все основные операции в VisSim производятся левой кнопкой мыши. Правая кнопка предназначена только для изменения свойств отдельных блоков и если кликнуть на каком-либо блоке правой кнопкой, то появится окно с его настройками.

Каждый блок изображается на диаграмме, как правило, в виде прямоугольника, внутри которого схематично изображено его функциональное назначение. С левой стороны блока находятся его *входные сигналы* (изображены в виде стрелок), с правой – *выходные*.

Для добавления и удаления входов/выходов у определенных блоков необходимо выбрать соответствующие пункты меню: **Edit** (*Редактирование*) → **Add Connector...** (*Добавить соединитель*)

для добавления коннектора (точки ввода вывода) – рисунок 4а или **Edit (Редактирование) → Remove Connector...** (Убрать соединитель) – рисунок 4б для удаления коннектора, после чего подвести указатель ко входу/выходу блока и кликнуть левой кнопкой мыши.

Edit	View	Simulate	Analyze	Blocks	Diagram
Undo					Ctrl+Z
Redo					Ctrl+A
Cut					Ctrl+X
Copy					Ctrl+C
Paste					Ctrl+V
Paste Link					
Delete					Del
Clear Errors					Ctrl+E
Flip Horizontal					Ctrl+Left
Create Compound Block...					
Dissolve Compound Block					
Auto Connect					
Auto Disconnect					
Align					>
Find...					Alt+F3
Replace...					
Block Properties...					Ctrl+RtBtn
Add Connector...					
Remove Connector...					
Toggle Tag					Ctrl+F2
Goto Tag					F2
Goto Next Change					
Reset Bitmap Scaling					
Repaint Screen					
Preferences...					
Объект					

А

Edit	View	Simulate	Analyze	Blocks	Diagram
Undo					Ctrl+Z
Redo					Ctrl+A
Cut					Ctrl+X
Copy					Ctrl+C
Paste					Ctrl+V
Paste Link					
Delete					Del
Clear Errors					Ctrl+E
Flip Horizontal					Ctrl+Left
Create Compound Block...					
Dissolve Compound Block					
Auto Connect					
Auto Disconnect					
Align					>
Find...					Alt+F3
Replace...					
Block Properties...					Ctrl+RtBtn
Add Connector...					
Remove Connector...					
Toggle Tag					Ctrl+F2
Goto Tag					F2
Goto Next Change					
Reset Bitmap Scaling					
Repaint Screen					
Preferences...					
Объект					

Б

Рисунок 4 – Добавление (А) или удаление (Б) коннектора

Блоки можно **перемещать** на диаграмме для их наиболее удобного расположения. Для этого необходимо нажать левую кнопку мыши на блоке, который следует переместить и не отрывая пальца от кнопки, переместить мышь в нужную точку и отпустить кнопку мыши. После отпускания кнопки мыши блок расположится в том месте, где был курсор. Для одновременного перемещения *группы блоков* необходимо сначала эту группу **выделить**, т.е. обвести мышью область окна диаграммы, в который входят необходимые для выделения блоки. Как только кнопка мыши будет отпущена, блоки

окажутся выделенными. Далее с группой блоков можно оперировать как с единым блоком. Например, ее можно перемещать, удалять, копировать и объединять.

Для **удаления** блоков следует сначала выделить блок (или блоки) и нажать на клавишу **Delete** (*Удалить*) на клавиатуре.

Если на строящейся диаграмме должны иметь место несколько групп блоков, похожих по структуре и (или) составу (повторяющиеся элементы схемы), то значительно упростить процесс ее создания поможет **копирование** блоков. Для копирования необходимо выделить копируемую группу блоков и нажать на клавиатуре комбинацию клавиш **Ctrl+C** (для этого нажать клавишу **Ctrl** и, удерживая ее в нажатом положении, нажать **C**). Группа блоков копируется в буфер обмена Windows. Далее необходимо нажать комбинацию клавиш **Ctrl+V** и мышью указать место, куда будет помещена скопированная группа. Клик мыши располагает группу на диаграмме. Если необходимо разместить третью, четвертую и т. д. группы, то необходимо повторить вышеописанные действия необходимое число раз.

Объединение блоков применяется для визуального упрощения диаграммы. Например, если диаграмму можно разбить на несколько структурно отдельных объектов (например, модель технологического объекта, модель регулятора и т. д.) и при этом структура диаграммы достаточно сложна, то каждую группу блоков можно объединить в единый блок. Для этого необходимо выделить группу блоков, войти в меню **Edit** (*Редактирование*) и выбрать пункт меню **Create Compound Block** (*Создать составной блок*). В появившемся диалоговом окне следует ввести название создаваемого объекта и нажать кнопку «ОК». Сразу же группа блоков на диаграмме будет заменена одним блоком зеленого цвета. Если войти в этот блок (кликом правой кнопки мыши или двойным кликом левой кнопки), то можно увидеть первоначально располагавшиеся на диаграмме блоки. Количество вложений блоков не ограничено.

Создание линий связи производится левой кнопкой мыши. Чтобы соединить два расположенных на диаграмме блока необходимо подвести мышь к выходу (расположен с правой стороны) первого соединяемого блока. Как только курсор превратится в стрелку, необходимо нажать левую кнопку мыши и не отпуская ее, подвести мышь ко входу второго соединяемого блока (находится с левой стороны). В процессе движения мыши за ней будет тянуться линия связи. Как только курсор достигнет входа, необходимо отпустить кнопку. Практика показывает, что линии удобнее проводить в обратном порядке.

Для **разрыва линий связи** необходимо навести мышь на начало (или конец) линии, нажать левую кнопку, немного отодвинуть мышь от блока и отпустить кнопку. Линия связи исчезнет.

Undo	Ctrl+Z
Redo	Ctrl+A
Cut	Ctrl+X
Copy	Ctrl+C
Paste	Ctrl+V
Paste Link	
Delete	Del
Clear Errors	Ctrl+E
Flip Horizontal	Ctrl+Left
Create Compound Block...	
Dissolve Compound Block	
Auto Connect	

Рисунок 5 – Разворот блока по горизонтали

Иногда (например, при изображении линий обратной связи) требуется **развернуть блок** (блоки), то есть расположить его так, чтобы его входы находились с правой стороны, а выходы – с левой. Для этого необходимо выделить блок (блоки) и нажать комбинацию клавиш на клавиатуре **Ctrl+Left** (левая стрелка клавиш управления курсором на клавиатуре) или **Ctrl+R** (первая буква слова *rotation* – разворот), или перейти в меню по пути **Edit (Редактирование) → Flip Horizontal (Перевернуть по горизонтали)** – рисунок 5.

1.4 Принципы управления моделью и получения результатов моделирования. Пример модели

Для того, чтобы промоделировать построенную блок-схему следует запустить процесс на исполнение для чего нужно кликнуть на панели инструментов (рисунок 1) по кнопке с зеленым треугольником , которая означает команду **Пуск** или нажать на клавиатуре клавишу **F5**. В результате работы модели выходные сигналы блоков начнут изменяться, их величины будут отображаться на виртуальном осциллографе и других индикаторах. Параметры некоторых сигналов и блоков можно изменять в процессе симуляции, другие параметры можно изменить, остановив процесс работы модели, нажав кнопку  (**Пауза**). Настройки расчета модели (продолжительность работы, выбор метода интегрирования и т. д.) можно задавать до ее запуска в окне по пути **Simulate (Имитация) → Simulation Properties... (Свойства имитации)**.

Продемонстрируем работу программы на примере создания модели для измерения и индикации постоянного сигнала (рисунок 6).

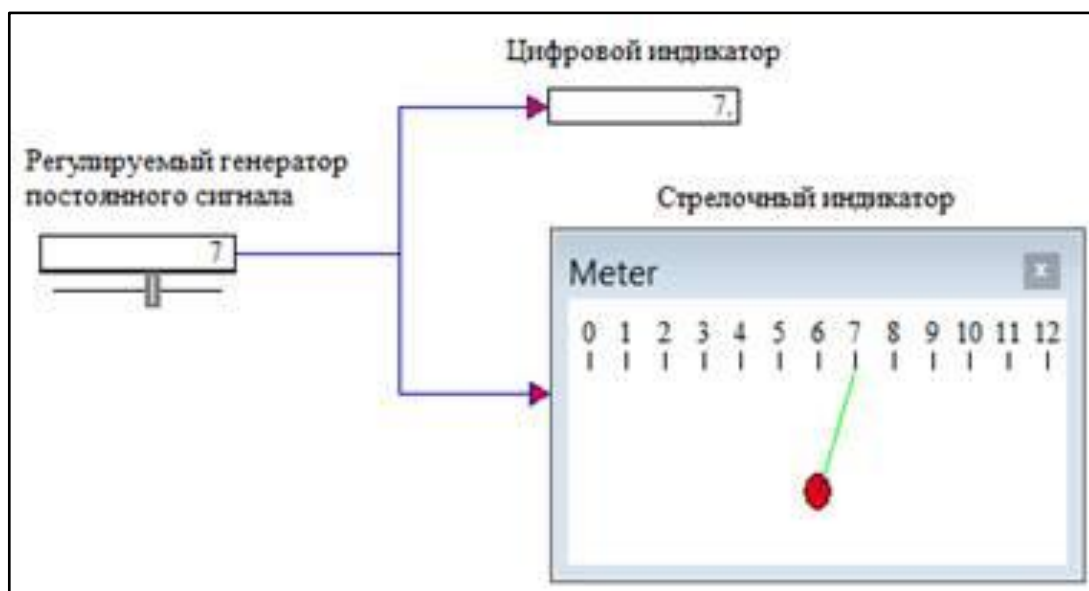


Рисунок 6 – Модель для измерения и индикации постоянного сигнала

Перед началом работы необходимо установить кириллический шрифт, например, для шрифта Arial. Первоначально необходимо перейти к пункту меню шрифтов по пути **View (Вид) → Fonts...** (Шрифты). В диалоговом окне выбрать шрифт Arial и в свойствах его начертания выбрать из раскрывающегося списка кириллицу (рисунок 7). Для лучшей читаемости в этом же разделе меню (**View**) нужно установить режим презентации (опция **Presentation Mode**).

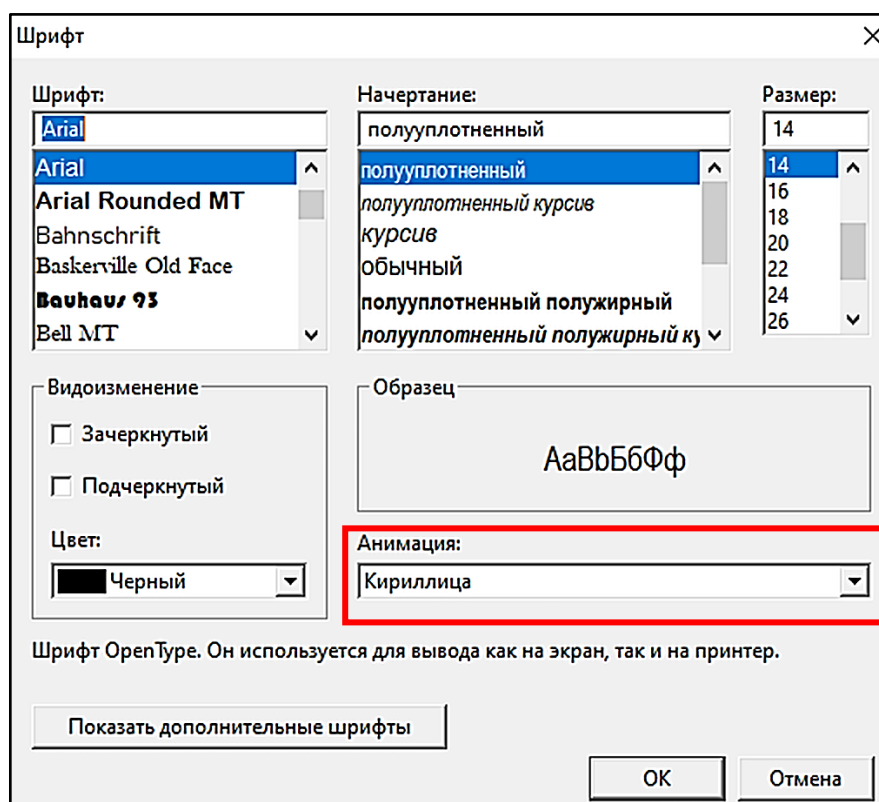


Рисунок 7 – Выбор кириллического начертания шрифта

Затем необходимо убедиться, что в параметрах симуляции (пункт меню **Simulate** (Имитация) → **Simulation Properties** (Свойства имитации)) – рисунок 8 правильно выставлены временной интервал (от 0 до 10 секунд с шагом моделирования 0.01 с) – рисунок 9.

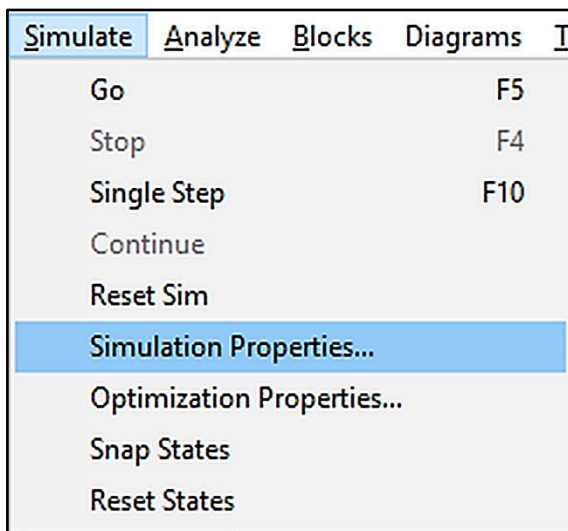


Рисунок 8 – Выбор пункта меню Simulation Properties

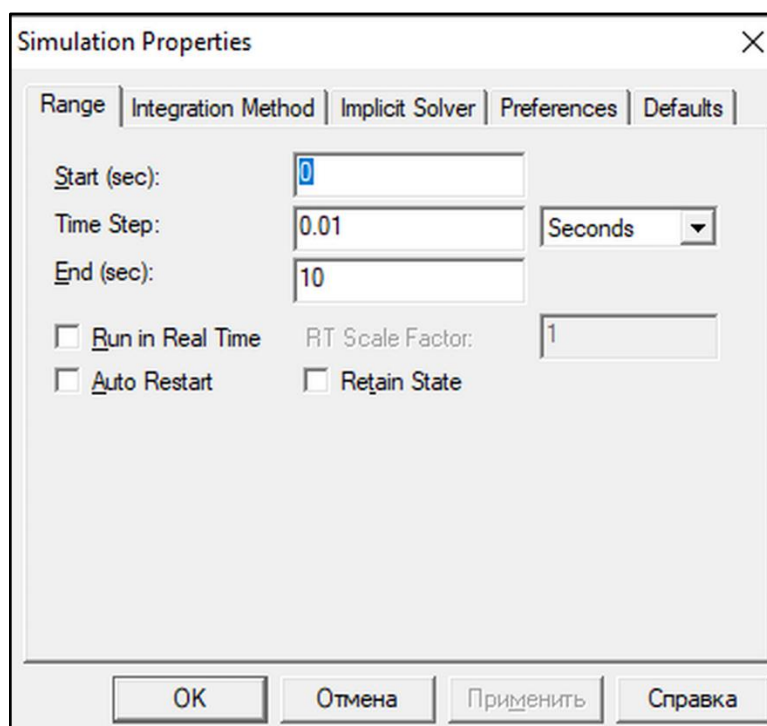


Рисунок 9 – Диалоговое окно Simulation Properties

После этого необходимо озаглавить диаграмму. Для этого необходимо поместить в верхней левой части окна диаграммы этикетку с названием, например «Модель для измерения и индикации постоянного сигнала». Размещение этикетки с названием осуществляется с использованием блока **label** (Метка). Он

добавляется в рабочее окно программы с использованием следующего пути в меню: **Blocks** (Блоки) → **Annotation** (Аннотация) → **label** (Метка). Можно в настройках поэкспериментировать со шрифтами и цветовым оформлением на вкус пользователя.

После размещения названия диаграммы можно разместить комментарий к ней используя команду **comment** и перейдя по пути меню: **Blocks** (Блоки) → **Annotation** (Аннотация) → **comment** (Комментарий), в котором можно указать цель составления диаграммы, например «Цель – демонстрация возможностей программы».

После этого можно перейти непосредственно к созданию самой диаграммы в виде следующей схемы:

1) вставить **генератор постоянного сигнала** (меню **Blocks** (Блоки) → **Signal Producer** (Генератор сигнала) → **slider** (Ползунок));

2) добавить цифровой индикатор из меню: **Blocks** (Блоки) → **Signal Consumer** (Потребитель сигналов) → **display** (Цифровой индикатор));

3) добавить аналоговый (стрелочный) индикатор из меню: **Blocks** (Блоки) → **Signal Consumer** (Потребитель сигнала) → **meter** (Измерительный прибор)), в свойствах которого очистите поле **Axis Label** (Подпись оси) и установите значение **Upper Bound** (Верхняя граница значений) = 12 (рисунок 10);

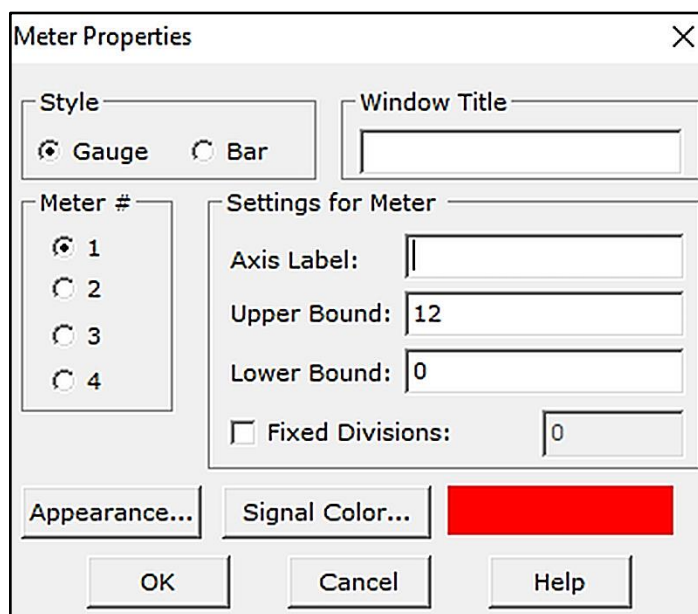
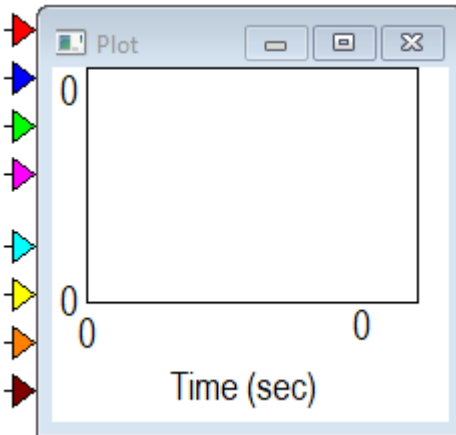
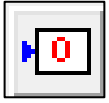
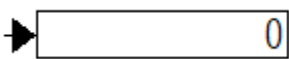
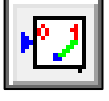
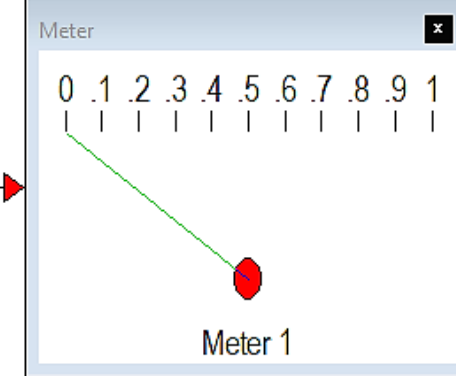
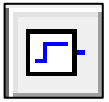
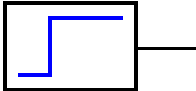
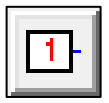
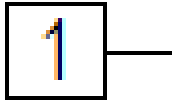
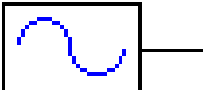
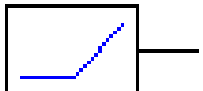
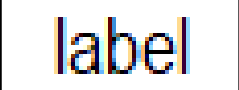
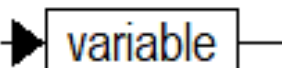


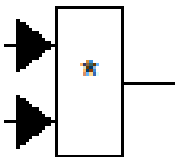
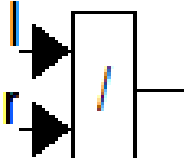
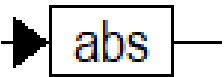
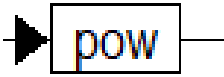
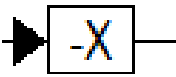
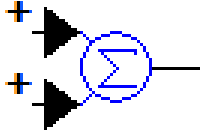
Рисунок 10 – Диалоговое окно свойств блока meter

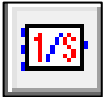
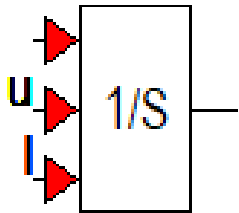
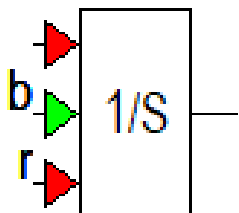
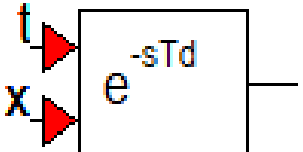
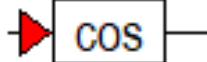
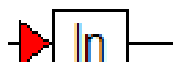
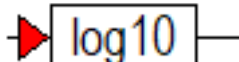
4) Переместите указателем мыши ползунок в блоке **slider** (Ползунок), например, на значение 2 и запустите программу на расчет, нажав в панели инструментов кнопку .

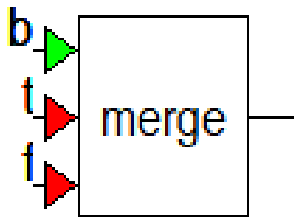
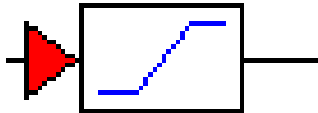
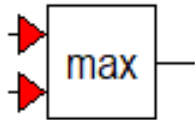
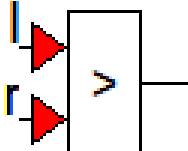
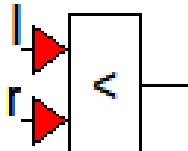
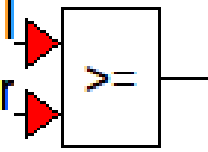
ОСНОВНЫЕ И ЧАСТО ИСПОЛЬЗУЕМЫЕ ФУНКЦИОНАЛЬНЫЕ БЛОКИ VISSIM

Изображение кнопки	Изображение блока	Назначение
Приборы и датчики (Blocks → Signal Consumer)		
		Блок plot (<i>осциллограф</i>) является универсальным прибором визуализации входных сигналов. Виртуальный запоминающий осциллограф в пакете VisSim является прибором двумерной визуализации информации
		Блок display (<i>цифровой индикатор</i>) отображает текущие значения входного сигнала
		Блок meter (<i>измерительный прибор</i>) визуализирует величину входного сигнала, отображая ее с помощью стрелочного или шкального указателя. Первоначально при установке в блок-схему он отображается в виде стрелочного указателя с одним входом. Однако, добавляя входы, можно получить приборную панель, содержащую до четырех указателей
Генераторы (Blocks → Signal Producer)		
		Блок step генерирует единичную ступенчатую функцию $1(t)$, которая часто используется в качестве возмущающего воздействия для получения переходной функции системы – $h(t)$
		Блок const (<i>константа</i>) генерирует либо постоянный сигнал, или матрицу элементов-констант, или алфавитно-цифровую текстовую строку

		Блок slider (<i>регулятор</i>) генерирует постоянный сигнал, величину которого в процессе симуляции можно менять либо с произвольным приращением путем перемещения мышью движка регулятора, либо с фиксированным с помощью щелчка мышью по оси движка. Для установки прецизионного значения нужно открыть диалоговое окно свойств
		Блок синусоида генерирует сигнал синусоидальной формы. Изменяя параметр Time Delay (<i>временное смещение</i>), можно произвольно менять начальную фазу сигнала
		Блок ramp (<i>const t</i>) генерирует сигнал, изменяющийся с постоянной скоростью
Аннотационные блоки (Blocks → Annotation»)		
		Блок comment (<i>комментарий</i>) позволяет качественно документировать модели
		Блок label (<i>метка</i>) весьма удобен при оформлении блок-схемы небольшими комментариями. Например, при маркировке сигналов. Для метки можно определить атрибуты текста и цвета фона
		Блок variable (<i>переменная</i>) существенно облегчает разработку больших блок-схем и повышает их наглядность. Он позволяет передавать сигналы без использования проводников. Имеется механизм локализации переменной в пределах ветви иерархии или уровня диаграммы
		Блок wirePositioner (<i>фиксатор Провода</i>) позволяет позиционировать направление линий связи между блоками. Он не преобразовывает сигналы и не требует временного ресурса в процессе симуляции. Может фиксировать как одиночные проводники, так и шинные. Бывает удобен в процессе отладки, если необходимо переподключить совокупность входов к одной из альтернативных точек блок-схемы (типовой случай - передача сигнала единичной обратной связи на вход сумматора)
Арифметические (Blocks → Arithmetic)		
		Блок 1/X возвращает обратную входному сигналу величину

		Блок * (<i>умножение</i>) перемножает входные сигналы. Если на часть входов сигналы не поданы, то VisSim подает на них единичное значение (это исключение – значения на неподключенных входах других блоков обнуляются)
		Блок / (<i>деление</i>) делит сигнал на верхнем входе с литерой 'l' на сигнал на нижнем входе с литерой 'r'. Если сигнал на входе с литерой 'r' не подан или равен нулю, то VisSim выводит информационное сообщение о факте деления на ноль и устанавливает флаг ошибки, раскрашивая блок красным цветом
		Блок abs возвращает абсолютное значение входного сигнала
		Блок power масштабирует входной сигнал, возводя его в степень, значение которой определяет внутренний параметр (указывается в свойствах блока). Для динамического задания показателя степени блоку power необходимо добавить второй вход
		Блок -X (<i>отрицание</i>) инвертирует входной сигнал
		Блок summingJunction (<i>сумматор</i>) суммирует совокупность входных сигналов. При этом, для любого входного сигнала можно изменить знак, и он будет вычитаться. Для этого, удерживая клавишу CTRL, нужно щелкнуть правой кнопкой мышки по входу сумматора, когда вид указателя сменился на стрелку ↑
		Блок gain (<i>коэффУсиления</i>) масштабирует входную величину пропорционально задаваемому коэффициенту. Не следует путать блок коэффУсиления с усилителем. Блок коэффУсиления – это безинерционное звено, поэтому его нельзя охватывать цепью отрицательной обратной связи, т. к. получается алгебраическая петля
Интеграторы (Blocks → Integration)		
		Блок 1/S (<i>интегратор</i>) численно интегрирует входной сигнал, используя назначенный метод интегрирования. Интегрирующий элемент (блок) является фундаментальным для любого пакета, назначение которого симуляция движения (моделирование) систем

		Блок limitedIntegrator (<i>интегратор с насыщением</i>) численно интегрирует сигнал, который подается на верхний вход. Если значение интеграла входного сигнала достигает пределов, которые определяются сигналами на входах с литерой 'u' (верхний предел насыщения) и 'l' (нижний предел насыщения) соответственно, то интегратор теряет свои интегрирующие свойства до момента смены знака входного сигнала, или до момента расширения пределов насыщения
		Блок resetIntegrator (<i>интегратор со сбросом</i>) численно интегрирует входной сигнал, если модуль сигнала на входе логического управления 'b' меньше 1. Если модуль сигнала на входе с литерой 'b' больше или равен 1, то внутреннее состояние этого интегратора (выходное значение) будет динамически сброшено к задаваемому на входе 'r' начальному значению (будет повторять сигнал, который подается на вход 'r')
Запаздывания (Blocks → Time Delay)		
		Блок timeDelay (<i>временная Задержка смещение</i>) является звеном чистого запаздывания для входного сигнала. Величину запаздывания можно менять в процессе симуляции. Для этого блок имеет вход, который маркирован литерой 't' (верхний). Сигнальный вход маркирован литерой 'x' (нижний)
Трансцендентные (Blocks → Transcendental)		
		Блок cos (<i>косинус</i>) возвращает значение косинуса входного сигнала, который должен быть представлен в радианах
		Блок sin возвращает значение синуса входного сигнала, который должен быть представлен в радианах
		Блок ln возвращает натуральный логарифм входного сигнала
		Блок log10 возвращает десятичный логарифм входного сигнала
		Блок sqrt возвращает корень квадратный из входного сигнала
Нелинейные (Blocks → Nonlinear)		

		Блок merge (<i>переключатель</i>) осуществляет условную коммутацию двух сигнальных проводников или шин (подаются на входы с литерой 't' и 'f'), в зависимости от значения логического управляющего сигнала на входе, маркированном литерой 'b'. Литеры b, t, и f, которыми помечены входы блока – это первые буквы слов <i>boolean</i> , <i>true</i> и <i>false</i> , т. к. фактически блок является конструкцией условного исполнения <i>if.. then.. else..</i>
		Блок limit (<i>ограничитель</i>) ограничивает выходной сигнал в соответствии с заданными пределами – верхним <i>Upper Bound</i> и нижним <i>Lower Bound</i> . Если входной сигнал находится в заданных пределах, то блок передает его на выход без преобразований. Если входной сигнал выходит за пределы, то до тех пор, пока ситуация не измениться, выходному сигналу будет присваиваться значение соответствующего предела. Верхний и нижний пределы задаются в диалоговом окне свойств блока, и нет возможности динамически менять их значения
		Блок max сравнивает значения двух входных сигналов на каждом шаге симуляции и больший из них передает на выход
		Блок min сравнивает значения двух входных сигналов на каждом шаге симуляции и меньший из них передает на выход
Логические (Blocks → Boolean)		
		Блок > (<i>больше чем</i>) сравнивает два входных сигнала: - если сигнал на входе с литерой 'l' больше чем сигнал на входе с литерой 'r', то на выходе будет 1; - если сигнал на входе с литерой 'l' меньше чем или равен сигналу на входе с литерой 'r', то на выходе будет 0.
		Блок < (<i>меньше чем</i>) сравнивает два входных сигнала: - если сигнал на входе с литерой 'l' меньше чем сигнал на входе с литерой 'r', то на выходе будет 1; - если сигнал на входе с литерой 'l' больше чем или равен сигналу на входе с литерой 'r', то на выходе будет 0.
		Блок >= (<i>больше чем или равно</i>) сравнивает два входных сигнала: - если сигнал на входе с литерой 'l' больше чем или равен сигналу на входе с литерой 'r', то на выходе будет 1;

		- если сигнал на входе с литерой 'l' меньше чем сигнал на входе с литерой 'r', то на выходе будет 0.
		Блок <= (<i>меньше чем или равно</i>) сравнивает два входных сигнала: - если сигнал на входе с литерой 'l' меньше чем или равен сигналу на входе с литерой 'r', то на выходе будет 1; - если сигнал на входе с литерой 'l' больше чем сигнал на входе с литерой 'r', то на выходе будет 0.
		Блок == (<i>равно</i>) сравнивает два входных сигнала: - если сигнал на входе с литерой 'l' равен сигналу на входе с литерой 'r', то на выходе будет 1; - если сигнал на входе с литерой 'l' не равен сигналу на входе с литерой 'r', то на выходе будет 0.
		Блок != (<i>не равно</i>) сравнивает два входных сигнала: - если сигнал на входе с литерой 'l' не равен сигналу на входе с литерой 'r', то на выходе будет 1; - если сигнал на входе с литерой 'l' равен сигналу на входе с литерой 'r', то на выходе будет 0.
		Блок and (<i>и</i>) выполняет поразрядное логическое умножение входных сигналов (логическая операция И)
		Блок not (<i>не</i>) выполняет логическую инверсию входного сигнала (логическая операция НЕ): - если сигнал на входе равен нулю (ложь), то на выходе будет 1 (истина); - если сигнал на входе не равен нулю (истина), то на выходе будет 0 (ложь)
		Блок or (<i>или</i>) выполняет поразрядное логическое сложение входных сигналов (логическая операция ИЛИ)