

НИУ МИЭТ

Лабораторная работа №2

Разработка моделей с использованием StateFlow

Москва, 2024

Цель работы

1. Изучить основные приемы разработки моделей с использованием интерактивного инструмента Stateflow.
2. Приобрести практические навыки разработки моделей с использованием Stateflow.

Теоретическая часть

Stateflow - инструмент для численного моделирования систем, характеризующихся сложным поведением. К числу таких систем относятся гибридные системы. Примерами гибридных систем могут служить системы управления, используемые в промышленности (автоматизированные технологические процессы), в быту (сложные бытовые приборы), в военной области (высокотехнологичные виды вооружений), в сфере космонавтики, транспорта и связи. Все эти системы состоят из аналоговых и дискретных компонентов. Поэтому гибридные системы - это системы со сложным взаимодействием дискретной и непрерывной динамики. Они характеризуются не только непрерывным изменением состояния системы, но и скачкообразными вариациями в соответствии с логикой работы управляющей подсистемы, роль которой, как правило, выполняет то или иное вычислительное устройство (конечный автомат).

Не удивительно, что в соответствии с изменением окружающего нас мира меняются и подходы к его анализу. Моделирование физики технологических процессов (непрерывная составляющая поведения системы) дополняется моделированием логики работы управляющих ими устройств (дискретная компонента). Математический аппарат описания в данном случае - это система уравнений, но не дифференциальных, а дифференциально-алгебраическо-логических, для которых отсутствует стройная теория и единый подход. Так же обстоит дело и с наглядностью. Визуализация протекания физических процессов обеспечивалась графиками изменения во времени тех или иных величин. Попытка такого графического представления процессов в реактивных системах может закончиться неудачно. Основными причинами этого являются многократное возрастание количества отображаемых величин и отсутствие на графиках

информации о причинно-следственных связях между изменяющимися переменными состояниями.

В настоящее время для моделирования дискретной динамики реактивных систем широко используется предложенный Д. Харелом визуальный формализм - Statechart (диаграммы состояний и переходов). Основные неграфические компоненты таких диаграмм - это событие и действие, основные графические компоненты - состояние и переход.

Событие - нечто, происходящее вне рассматриваемой системы, возможно требуя некоторых ответных действий. События могут быть вызваны поступлением некоторых данных или некоторых задающих сигналов со стороны человека или некоторой другой части системы. События считаются мгновенными (для выбранного уровня абстрагирования).

Действия - это реакции моделируемой системы на события. Подобно событиям, действия принято считать мгновенными.

Состояние - условия, в которых моделируемая система пребывает некоторое время, в течение которого она ведет себя одинаковым образом. В диаграмме переходов состояния представлены прямоугольными полями со скругленными углами.

Переход - изменение состояния, обычно вызываемое некоторым значительным событием. Как правило, состояние соответствует промежутку времени между двумя такими событиями. Переходы показываются в диаграммах переходов линиями со стрелками, указывающими направление перехода.

Каждому переходу могут быть сопоставлены условия, при выполнении которых переход осуществляется.

С каждым переходом и каждым состоянием могут быть соотнесены некоторые действия. Действия могут дополнительно обозначаться как действия, выполняемые однократно при входе в состояние; действия, выполняемые многократно внутри некоторого состояния; действия, выполняемые однократно при выходе из состояния.

Для предотвращения эффекта возрастания сложности при моделировании больших систем были предложены дальнейшие усовершенствования. Наряду с состояниями теперь могут использоваться гиперсостояния (суперсостояния),

объединяющие несколько состояний, имеющих идентичную реакцию на одно и то же событие. При этом вместо изображения таких переходов в некоторое состояние из всех состояний, охватываемых гиперсостоянием, изображается только один переход из гиперсостояния в указанное состояние (обобщение переходов). Гиперсостояния теоретически могут иметь произвольную глубину вложения. Переходы из гиперсостояния связаны со всеми уровнями вложения. Гиперсостояния могут объединять ИЛИ-состояния (последовательные состояния) и И-состояния (параллельные состояния). В первом случае, перейдя в гиперсостояние, система может находиться только в одном из состояний. Во втором случае система, перейдя в гиперсостояние, будет пребывать одновременно в нескольких состояниях.

Диаграммы состояний и переходов в настоящее время широко используются для моделирования сложных систем. Достаточно упомянуть унифицированный язык моделирования (Unified Modeling Language (UML)), одним из элементов которого являются диаграммы состояний и язык "Графсет", который используется при программировании логических контроллеров систем автоматизации.

Существенно повышает степень наглядности модели использование имитации, отображающей изменения в системе, сопровождающиеся переходами от одного состояния к другому. Построение таких имитационных моделей возможно с использованием программ Stateflow и Simulink, входящих в состав пакета MATLAB. MATLAB обеспечивает доступ к различным типам данных, высокоуровневому программированию и инструментальным средствам визуализации. Simulink поддерживает проектирование непрерывных и дискретных динамических систем в графической среде (в виде блок-схем). Stateflow - диаграммы, использующие визуальный формализм Д. Харела, включаются в Simulink -модели, чтобы обеспечить возможность моделирования процессов, управляемых событиями. Stateflow обеспечивает ясное описание поведения сложных систем, используя диаграммы состояний и переходов.

Комбинация MATLAB-Simulink-Stateflow является мощным универсальным инструментом моделирования реактивных систем. Дополнительная возможность следить в режиме реального времени за процессом выполнения диаграммы путем

включения режима анимации делает процесс моделирования реактивных систем по-настоящему наглядным.

Stateflow - мощный графический инструмент проектирования и моделирования комплексных систем локального управления и супервизорного логического контроля. Используя Stateflow, можно:

- Визуально моделировать комплексные реактивные системы.
- Проектировать детерминированные системы супервизорного управления.
- Легко изменять проект, оценивать результаты изменений и исследовать поведение системы на любой стадии проекта.
- Автоматически генерировать программный код (целочисленный или с плавающей точкой) непосредственно по проекту (для этого требуется Stateflow Coder).
- Пользоваться преимуществами интегрирования со средами MATLAB и Simulink в процессе моделирования и анализа систем.

Stateflow позволяет использовать диаграммы потоков (flow diagram) и диаграммы состояний и переходов (state transition) в одной диаграмме Stateflow. Система обозначений диаграммы потоков - логика, представленная без использования состояний. В некоторых случаях диаграммы потоков ближе логике системы, что позволяет избежать использования ненужных состояний. Система обозначений диаграммы потоков - эффективный способ представить общую структуру программного кода как конструкцию в виде условных операторов и циклов.

Stateflow также обеспечивает ясное, краткое описание поведения комплексных систем, используя теорию конечных автоматов, диаграммы потоков и диаграммы переходов состояний. Stateflow делает описание системы (спецификацию) и проект ближе друг другу. Создавать проекты, рассматривая различные сценарии и выполняя итерации, намного проще, если при моделировании поведения системы используется Stateflow.

Stateflow использует вариант системы обозначений конечного автомата, предложенный Дэвидом Харелом [D. Harel Statecharts: A Visual Formalism for Complex Systems," (Диаграммы состояний: визуальный формализм для комплексных систем) *Science of Computer Programming* 8, 1987, pages 231-274].

Используя Stateflow (state - состояние, flow - поток), Вы создаете Stateflow диаграммы. Диаграмма Stateflow - графическое представление конечного автомата, где состояния и переходы формируют базовые конструктивные блоки системы. Вы можете также представлять потоковые (не имеющие состояний) диаграммы с использованием Stateflow. Stateflow образует блоки, которые Вы включаете в модель Simulink. Совокупность Stateflow блоков в модели Simulink - Stateflow машина.

Дополнительно Stateflow допускает в представлении иерархию, параллелизм и хронологию (history). Иерархия дает возможность Вам организовать комплексные системы, определяя структуру объекта в виде "родители / потомки". Т.е. вы можете организовать состояния внутри других состояний высшего уровня. Система с параллелизмом может иметь два или больше активных состояний в одно и то же время. Хронология обеспечивает средства, чтобы определить состояние - адресата для некоторого перехода, основываясь на предшествующей информации. Эти свойства расширяют полезность данного подхода и отсутствуют в STDs и пузырьковых диаграммах.

Система обозначений

Система обозначений определяет набор объектов и правил, которые управляют отношениями между этими объектами. Stateflow система обозначений обеспечивает позволяет разработать проект в полном соответствии с диаграммой Stateflow.

Stateflow система обозначений состоит из:

- Набора графических объектов
- Набора неграфических (текстовых) объектов
- Определенных отношений между этими объектами

Семантика

Семантика описывает, как система обозначений интерпретируется и осуществляется. Законченная диаграмма Stateflow иллюстрирует, как система будет вести себя. Диаграмма Stateflow содержит действия, связанные с переходами и состояниями. Семантика описывает, в какой последовательности эти действия имеют место в ходе выполнения диаграммы Stateflow.

Знание семантики особенно важно для обеспечения правильных проектных решений Stateflow диаграмм при генерации объектного кода. Различное использование системы обозначений приводит к различному порядку выполнения сгенерированного кода.

Определение Stateflow интерфейса

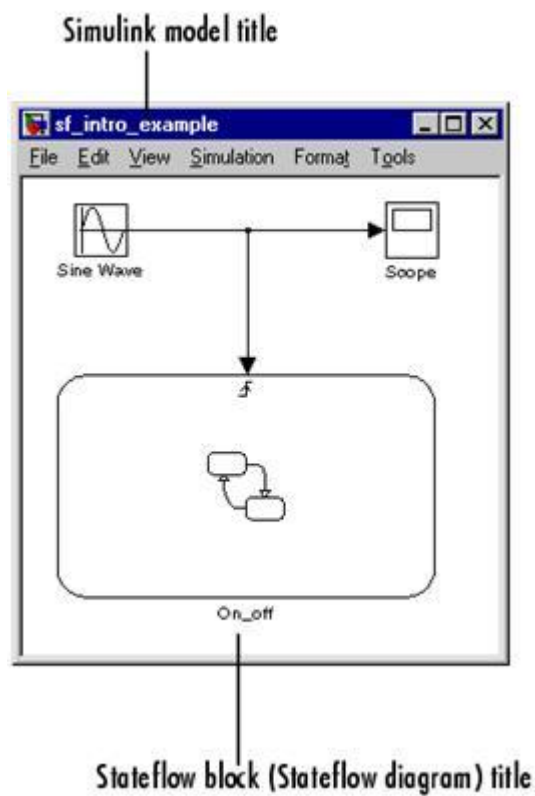
Каждый блок Stateflow соответствует единственной диаграмме Stateflow. Блок Stateflow связывает с моделью Simulink с помощью интерфейса. С помощью интерфейса блок Stateflow подключается к источникам, поступающим от модели Simulink (данные, события, код пользователя).

Stateflow диаграммы управляются событиями. События могут быть локальными для блока Stateflow или могут поступать к и от модели Simulink и источников кода, внешних к Simulink. Данные также могут быть локальными для блока Stateflow или могут поступать к и от модели Simulink и источников кода, внешних к Simulink.

Вы должны определить интерфейс для каждого блока Stateflow. Определение интерфейса для блока Stateflow может включать некоторые или все из этих задач:

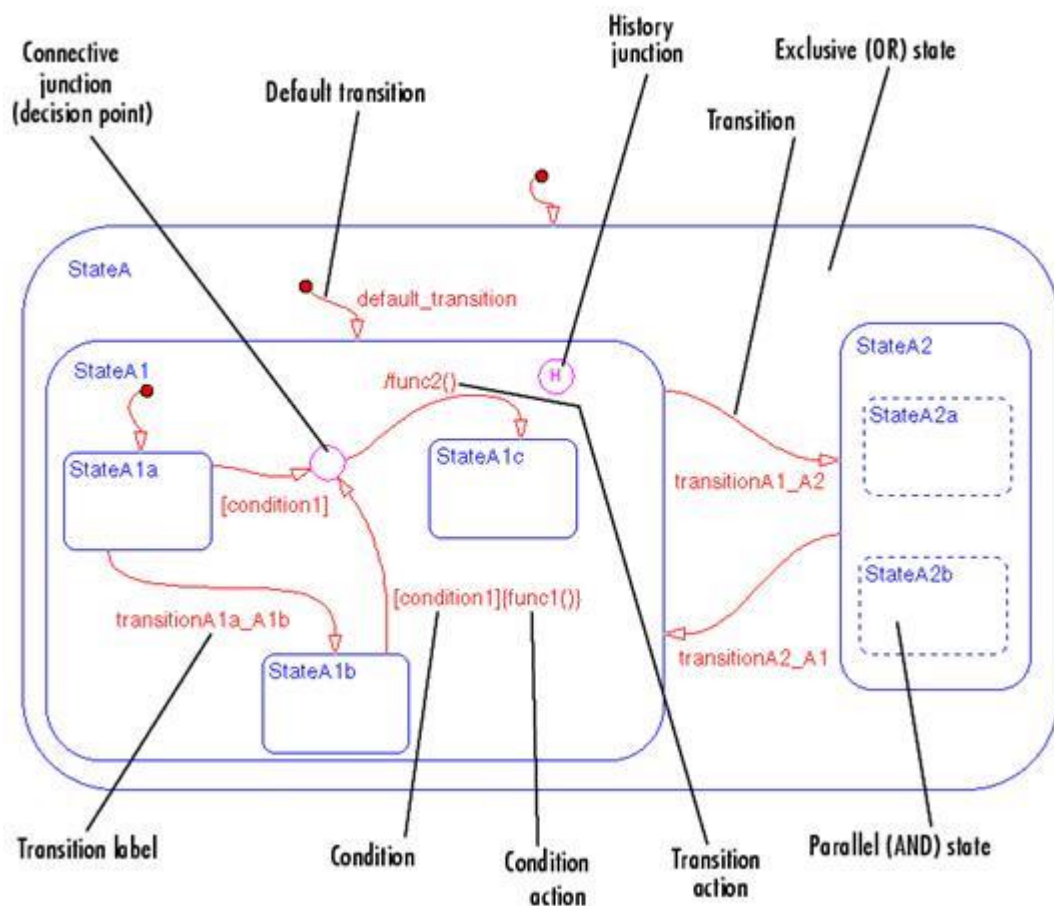
- Определение метода модификации блока Stateflow
- Определение **Output to Simulink** (Выходных к Simulink) событий
- Добавление и определение нелокальных событий и нелокальных данных в пределах диаграммы Stateflow
- Определение отношений с любыми внешними источниками

В рассмотренном ранее примере Simulink модель состоит из Simulink блока - источника Sine Wave (Синусоида), Simulink блока - приемника Scope (Осциллограф) и единственного блока Stateflow с названием On_off.



Объекты Stateflow диаграммы

Далее приводится пример Stateflow диаграммы, в которой используются основные графические компоненты. Кроме того, детально описываются эти графические компоненты, а также некоторые неграфические объекты и связи между ними.



Состояние (State)

Состояние описывает режим управляемой событиями системы. Динамические переходы состояний от активности к неактивности базируются на событиях и условиях. Каждое состояние имеет родителя. В диаграмме Stateflow, состоящей из единственного состояния, родитель состояния - непосредственно диаграмма Stateflow (также называемая корнем диаграммы Stateflow). Вы можете размещать состояния в пределах других состояний высшего уровня. На рисунке StateA1 - потомок в иерархии по отношению к StateA.

Состояние также имеет хронологию. Хронология обеспечивает эффективные средства базирования будущего действия на прошлом действии.

Состояния имеют метки, которые могут определить действия, выполненные в последовательности, основанной на типе действия. Типы действия - `entry` (на входе), `during` (в течение), `exit` (на выходе) и `on event_name` (в случае события с именем `_`).

Переходы (Transitions)

Переход - графический объект, который в большинстве случаев связывает один объект с другим. Один конец перехода приложен к объекту-источнику, а другой конец - к объекту-адресату. Источник - то место, где переход начинается, а адресат - то место, где переход заканчивается. Метки переходов описывают обстоятельства, под действием которых система переходит из одного состояния к другому. Эти обстоятельства - наступление некоторых событий, которые заставляют переход совершаться. На рисунке переход от StateA1 к StateA2 помечен событием transitionA1_A2, которое заставляет переход произойти.

События (Events)

События управляют выполнением диаграммы Stateflow, но являются неграфическими объектами и таким образом не представлены непосредственно в диаграмме Stateflow. Все события, которые имеют отношение к диаграмме Stateflow, должны быть определены. Наступление события заставляет статус состояния (активно - неактивно) в диаграмме Stateflow изменяться. Наступление события может запускать переход, и тогда он происходит, или может запускать действие, и тогда оно выполняется. События наступают по-нисходящей, начиная от родителя события в иерархии.

События создаются и изменяются при помощи Stateflow Explorer (Stateflow проводника). События могут быть созданы на любом уровне иерархии. Событие имеет такое свойство, как видимость. Видимость определяет, является ли событие

- Локальным для диаграммы Stateflow
- Входит в Stateflow диаграмму от модели Simulink
- Выходит из Stateflow диаграммы в модель Simulink
- Экспортируется в код, внешний к Stateflow диаграмме и модели Simulink
- Импортируется из источника кода, внешнего к Stateflow и Simulink

Данные (Data)

Объекты-данные используются, чтобы хранить числовые значения для применения в диаграмме Stateflow. Они являются неграфическими объектами и таким образом не представлены непосредственно в диаграмме Stateflow.

Данные создаются и изменяются в Stateflow Explorer. Они могут быть созданы на любом уровне иерархии. Данные имеют такое свойство, как видимость. Видимость определяет для объектов-данных одну из следующих возможностей:

- Быть локальными для диаграммы Stateflow
- Поступать в Stateflow диаграмму от модели Simulink
- Выходить из Stateflow диаграммы в модель Simulink
- Быть временными данными
- Быть определенными в рабочем пространстве MATLAB
- Быть Константами
- Экспортироваться в код, внешний к Stateflow диаграмме и модели Simulink
- Импортироваться из источника кода, внешнего к Stateflow и Simulink

Иерархия

Иерархия дает возможность организовать сложные системы, определяя предка и структуру объектов-потомков. Иерархически построенный проект обычно сокращает число переходов и приводит к четким, понятным диаграммам. Stateflow поддерживает иерархическую организацию как для диаграмм, так и для состояний. Диаграммы могут существовать внутри других диаграмм. Диаграмма, которая существует в другой диаграмме, называется поддиаграммой.

Точно так же состояния могут существовать внутри других состояний. Stateflow представляет иерархию состояний с суперсостояниями и подсостояниями. Например, эта диаграмма Stateflow имеет суперсостояние, которое содержит два подсостояния.

Выход из состояния высшего уровня или суперсостояния также подразумевает выход из любых активных подсостояний этого суперсостояния. Переходы могут пересекать границы суперсостояния, чтобы достичь подсостояния-адресата. Если подсостояние активно, его родительское состояние (суперсостояние) также активно.

Условия (Conditions)

Условие - булево выражение, определяющее, что переход происходит, если указанное выражение является истинным. На рисунке компонента Stateflow

диаграммы [condition1] представляет булево выражение, которое должно быть истинным, чтобы переход произошел.

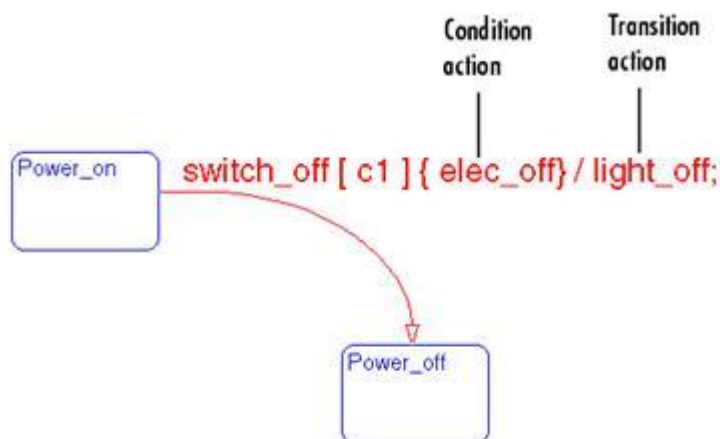
Хронологические соединения (History Junction)

Хронология - средство для определения подсостояния-адресата перехода по предыстории. Если суперсостояние с исключительной (ИЛИ) декомпозицией имеет хронологическое соединение, подсостоянием-адресатом при переходе будет подсостояние, посещенное до этого последним. Хронологическое соединение применяется к тому уровню иерархии, в котором присутствует. Хронологическое соединение отменяет любые переходы по умолчанию. На рисунке хронологическое соединение в StateA1 указывает, что, когда переход к StateA1 происходит, становится активным то из подсостояний (StateA1a, StateA1b или StateA1c), которое было активным в последнюю очередь.

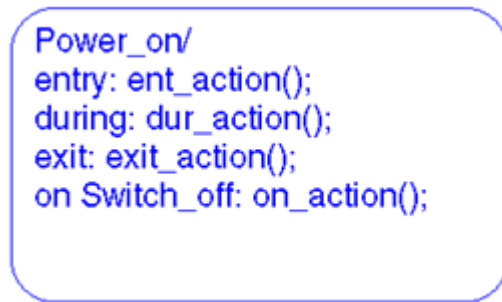
Действия (Actions)

Действия - это результат выполнения какой-либо части диаграммы Stateflow. Действие может быть выполнено в результате перехода от одного состояния к другому. Действие может быть также реакцией на состояние. На рисунке сегмент перехода от StateA1b до соединения помечен действием func1() условия condition 1, а сегмент перехода от соединения до StateA1c помечен действием func2() перехода. Семантика действий будет рассмотрена позднее.

Переход, заканчивающийся в состоянии, может иметь действие условия (condition action) и действие перехода (transition action), как рассмотрено ниже. Однако переходы, которые заканчиваются в соединениях, могут иметь только действия условий (не допускаются действия переходов).



Состояния могут иметь действия *have entry* (на входе), *during* (в течение), *exit* (на выходе) и *on event_name* (в случае события с именем *_*). Например,



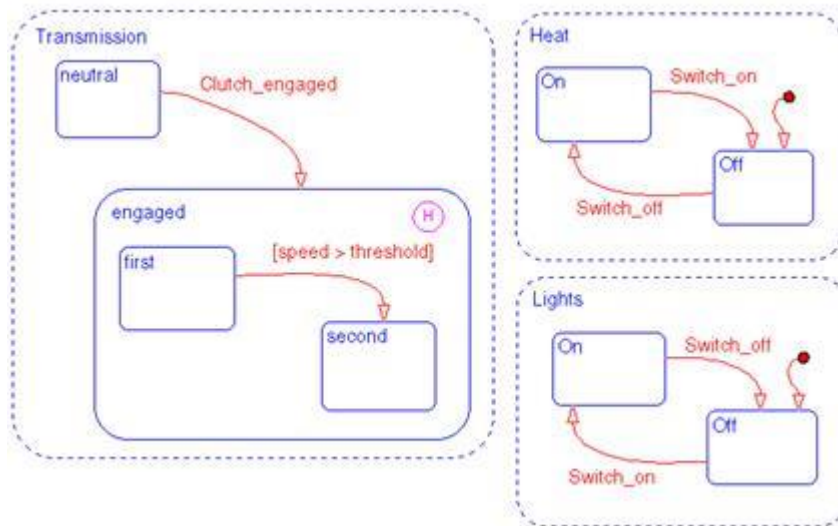
Язык действий определяет типы действий, которые Вы можете использовать и связанные с ними системы обозначений. Действием может быть обращение к функции, наступление события, присвоение некоторого значения переменной и т.д.

Stateflow поддерживает парадигмы моделирования конечного автомата Мура и Мили. В Мили модели действия связаны с переходами, в то время как в Мура модели они связаны с состояниями. Stateflow поддерживает действия состояний, действия переходов и действия условий.

Параллелизм

Система с параллелизмом имеет два или больше состояний, которые могут быть активны в одно и то же время. Действия каждого параллельного состояния по существу независимы от других состояний. На рисунке 2-1 StateA2a и StateA2b - параллельные (И) состояния. StateA2 имеет параллельную (И) декомпозицию состояния.

Например, эта диаграмма Stateflow имеет параллельную декомпозицию суперсостояния.



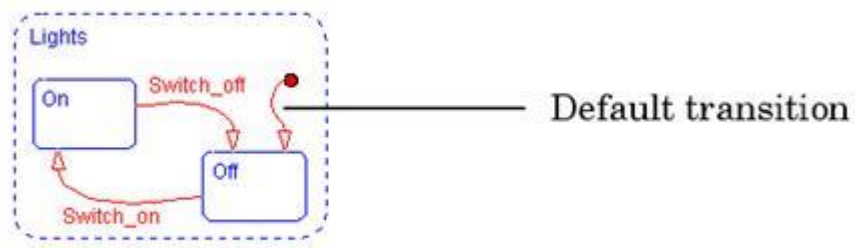
Вы представляете параллелизм в Stateflow, задавая параллельную (И) декомпозицию. Параллельные (И) состояния отображены обведенными штриховой линией областями.

Переходы по умолчанию (Default Transitions)

Переходы по умолчанию определяют, какое из нескольких исключительных (ИЛИ) состояний должно быть активным, когда имеется неопределенность между двумя или более исключительными (ИЛИ) состояниями на одном уровне в иерархии.

Например на рисунке 2-1 переход по умолчанию к StateA1 разрешает неоднозначность, которая существует в отношении того, какое из подсостояний, StateA1 или StateA2, должно быть активным, когда суперсостояние StateA становится активным. В этом случае, когда StateA активно, по умолчанию StateA1 также активно.

В следующей подсистеме Lights (Осветительные приборы) переход по умолчанию к подсостоянию Lights.Off (Осветительные приборы.выключены) указывает, что, когда суперсостояние Lights становится активным, Lights.Off подсостояние становится активным по умолчанию.



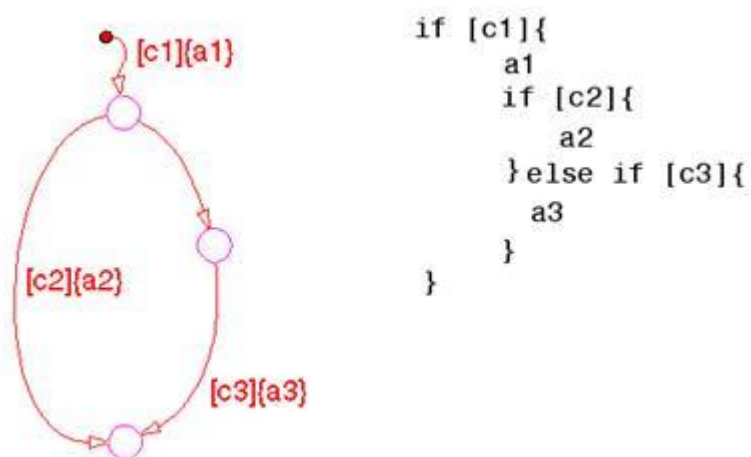
Обратите внимание! Хронологические соединения отменяют переходы по умолчанию в суперсостояниях с исключительными (ИЛИ) декомпозициями.

Обратите внимание! В параллельных (И) состояниях переходы по умолчанию всегда должны присутствовать, чтобы указать, какие из его исключительных (ИЛИ) состояний активны, когда параллельное состояние становится активным.

Соединения (Connective Junctions)

Соединения - точки принятия решений в системе. Соединение - графический объект, который упрощает Stateflow схематические представления и облегчает порождение эффективного кода. Соединения обеспечивают альтернативные способы представить желаемое поведение системы. На представленной выше Stateflow диаграмме соединение используется как точка принятия решения для двух сегментов перехода, завершающихся в состоянии StateA1c.

Следующий пример показывает, как соединения (отображаемые в виде окружностей) используются для конструкции if.



```

if [c1]{
    a1
    if [c2]{
        a2
    } else if [c3]{
        a3
    }
}
  
```

Этот фрагмент выполняется следующим образом:

Если условие [c1] истинно, условное действие a1 выполняется и происходит безусловный переход к первому (верхнему) соединению.

Stateflow определяет, какой сегмент перехода верхнего соединения выбрать (можно выбрать только один). Соединения с условиями имеют приоритет над соединениями без условий, т. о. переход с условием [c2] рассматривается первым.

Если условие [c2] истинно, действие a2 выполняется и происходит переход к нижнему соединению. Так как нет перехода, выходящего из этого соединения, выполнение диаграммы завершено.

Если условие [c2] ложно, происходит безусловный переход по правому из сегментов (он не имеет условия).

Если условие [c3] истинно, условное действие a3 выполняется и происходит переход к нижнему соединению. Выполнение диаграммы завершено.

Если условие [c3] ложно, выполнение заканчивается на среднем соединении.

1.3. Нотации Stateflow.

В этой главе приводится описание отдельных объектов Stateflow, из которых строятся Stateflow-диаграммы. Диаграммы Stateflow состояются из символических объектов языка Stateflow. Изучение этих объектов - первый шаг к построению эффективных диаграмм Stateflow. Познакомившись с нотациями Stateflow, можно будет перейти к рассмотрению того, как объекты диаграмм взаимодействуют друг с другом (Этому вопросу будет посвящена глава "Семантика Stateflow").

Виды объектов Stateflow



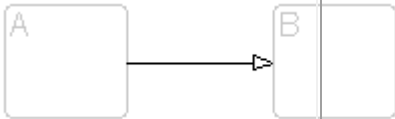
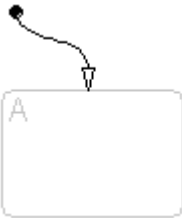







Перечислим различные типы доступных в Stateflow объектов.

графические объекты - диаграммы, состояния, блоки, функции, переходы и соединения, с которыми вы работаете в редакторе Stateflow.

- неграфические объекты - данные и события, которые представлены в текстовом виде в редакторе диаграммы Stateflow или инструментах Проводника Stateflow.
- словарь данных - объединяет все объекты Stateflow (графические и неграфические) в иерархическую базу данных.
- представление иерархии - отображает иерархию объектов Stateflow в словаре данных для графических и неграфических объектов.

Графические Объекты

В таблице приводится название (Name) и условное обозначение (Notation) каждого графического объекта Stateflow. Объект появляется, когда вы рисуете диаграмму в редакторе диаграмм, для этого используется соответствующий значок (Toolbar icon) на панели инструментов¹.

Name	Notation	Toolbar Icon
State (состояние)		
Transition (переход)		NA
Default transition (переход по умолчанию)		
Connective junction (подключаемое соединение)		
History junction (соединение с памятью)		
Box (ящик)		 (state)

¹ Обозначения в таблице представлены для Stateflow 5.0. В версии 6.x добавлены новые элементы и несколько модифицирован вид старых.

Graphical function (графическая функция)	<div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 0 auto;">function ()</div>	 (state)
--	---	--

Неграфические Объекты

Stateflow определяет неграфические объекты, описанные в следующих разделах:

События

Данные

Коды

События, данные и коды не имеют графического представления в редакторе диаграмм Stateflow. Но их можно увидеть в проводнике Stateflow.

События

События - это объекты Stateflow, которые могут переключать всю диаграмму Stateflow или конкретные действия в диаграмме. Так как диаграммы Stateflow выполняются посредством реакции на события, вы определяете и программируете события в ваших диаграммах, чтобы контролировать их выполнение. Вы можете распространять событие на все объекты, либо посылать событие на определенный объект. Вы можете подробно определить события, которые вы сами назначаете. Так же вы можете определить события по умолчанию, имеющие место, например, при входе в состояние.

Данные

Диаграмма Stateflow хранит и восстанавливает данные, которые используются для управления действием диаграммы. Данные Stateflow помещены в свое собственное рабочее пространство, но вы можете иметь доступ к данным, которые располагаются в модели Simulink или приложениях, которые встроены в машину Stateflow. Когда вы создаете диаграмму Stateflow, вы должны определить внутренние и внешние данные, которые вы используете в языке действий диаграммы Stateflow.

Коды

Вы строите коды в Stateflow для выполнения приложений, которые вы программируете в диаграммах Stateflow и моделях Simulink. Код - это программа, которая выполняет диаграммы Stateflow и модели Simulink, включенные в машину

Stateflow. Вы строите коды моделирования (называемые sfun) для выполнения вашей модели. Вы строите коды Мастерской Реального Времени (называемые rtw) для выполнения модели Simulink или поддержки окружения процессора. Вы строите обычные коды (они могут быть названы как угодно, за исключением sfun или rtw), чтобы поместить ваше приложение в определенное окружение.

Словарь данных

Словарь данных - это база данных, включающая в себя всю информацию о графических и неграфических объектах. Словарь данных для графических объектов создается автоматически. Для неграфических объектов необходимо определять данные, используя Проводник. Грамматический отладчик анализирует входы и связи между входами в словаре данных, чтобы проверить правильность нотаций.

Как представлена иерархия

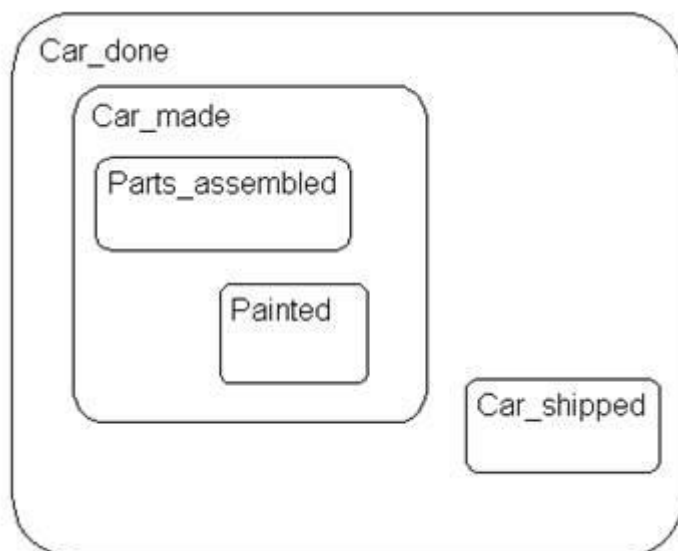
Нотации Stateflow поддерживают представление графических объектов в виде иерархии в диаграммах Stateflow. Рассмотрим следующие примеры представления иерархий в диаграммах Stateflow:

Пример представления иерархии состояний

Иерархия данных и событий, как неграфических объектов, представлена в проводнике. Иерархия данных и событий определяются при их создании назначением им родительского объекта.

Пример представления иерархии состояний

В представленном примере изображено одно состояние, в рамках которого другое состояние показывает, что внутреннее состояние - это подсостояние или потомок внешнего состояния или надсостояния, и внешнее состояние - родительское по отношению к внутреннему:



В этом примере диаграмма Stateflow - родитель по отношению к состоянию Car_done. Состояние Car_done - родитель состояний Car_made и Car_shipped. Состояние Car_made - также родитель состояний Parts_assembled и Painted. Вы также можете сказать, что состояния Parts_assembled и Painted - потомки состояния Car_made. Иерархия Stateflow также может быть представлена в текстовом виде, тогда символ слэш (/) представляет диаграмму Stateflow, и каждый уровень иерархии состояний разделяется символом (.). Далее показано текстовое представление иерархии объектов предыдущего примера:

Состояния

Этот раздел описывает основные объекты Stateflow - состояния. Состояния представляют режимы реактивной системы. Рассмотрим следующие вопросы для получения информации о состояниях и их свойствах:

Что такое состояние? - описывает состояния как режимы, которые могут быть активны или неактивны.

Декомпозиция состояний - показывает, как состояния могут быть последовательны или параллельны по отношению друг к другу в процессе выполнения системы.

Нотации меток состояний - показывает, как определяется имя состояния и его действия посредством меток.

Что такое состояние?

Состояния описывают режимы реактивных структур Stateflow. Состояния в структурах Stateflow представлены этими режимами.

Иерархия состояний

Состояния могут быть надсостояниями, подсостояниями и просто состояниями. Состояние называется надсостоянием, если включает в себя другие состояния, называемые подсостояниями. Состояние называется подсостоянием, если оно находится внутри другого состояния. Если состояние не является ни надсостоянием, ни подсостоянием, то это просто состояние. Каждое состояние - это часть иерархии. В диаграмме Stateflow, состоящей из одного состояния, родителем этого состояния является сама диаграмма Stateflow. Состояние также имеет память, которая применима к его уровню иерархии диаграммы Stateflow. Состояния могут иметь действия, которые выполняются в последовательности, которая определяется их типом. Типы действий: действия при входе (entry), действия во время активности (during), действия при выходе (exit) и действия при наступлении события с именем *event_name* (on *event_name*).

Активные и неактивные состояния

Когда состояние активно, диаграмма переходит в этот режим. Когда состояние неактивно, диаграмма не в этом режиме. Активность и неактивность состояний диаграммы динамически изменяется, базируясь на событиях и условиях. Появление событий управляет выполнение диаграммы Stateflow посредством активности и неактивности состояний. На любой стадии выполнения диаграмма Stateflow есть комбинация активных и неактивных состояний.

Декомпозиция состояний

Каждое состояние и диаграмма имеет декомпозицию, которая определяет, какие виды подсостояний оно может включать в себя. Все подсостояния должны быть такого же типа, как декомпозиция надсостояния. Декомпозиция может быть последовательной (ИЛИ) или параллельной (И). Эти типы декомпозиций описаны в следующих подразделах:

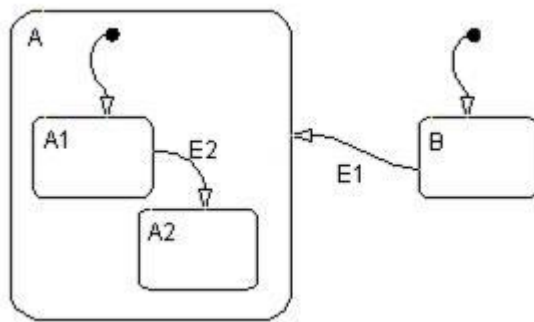
Последовательная декомпозиция состояний (ИЛИ)

Параллельная декомпозиция состояний (И)

Последовательная декомпозиция состояний (ИЛИ)

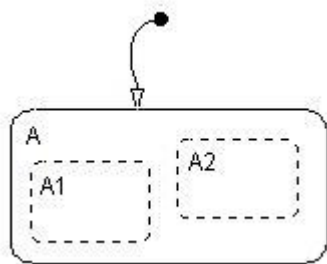
Если подсостояния имеют сплошные границы, декомпозиция состояний является последовательной (ИЛИ). Последовательная (ИЛИ) декомпозиция используется для описания режимов системы, которые взаимно последовательны.

Когда состояние имеет последовательную (ИЛИ) декомпозицию, в любой момент времени может быть активно только одно состояние. Потомками родителей с последовательной (ИЛИ) декомпозицией являются состояния ИЛИ. В следующем примере активным может быть или состояние А, или состояние В. Если состояние А активно, то в любой момент времени может быть активным одно из состояний А1 и А2.

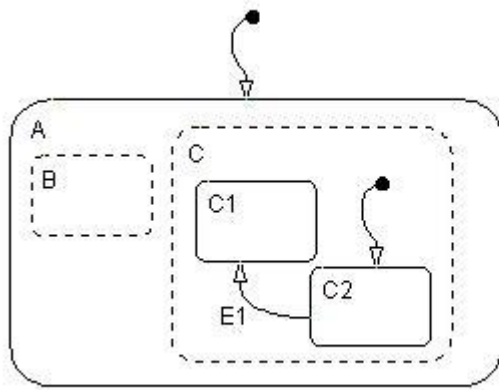


Параллельная декомпозиция состояний (И)

Потомки родителя с параллельной (И) декомпозицией - это параллельные (И) состояния. Если подсостояния имеют пунктирную границу, декомпозиция состояний является последовательной (ИЛИ). Это представление применяется, если все состояния на этом уровне иерархии всегда активны в одно и тоже время. В следующем примере, когда состояние А активно, и А1 и А2 активны одновременно:



Активность в параллельных состояниях практически независима, как это показано в следующем примере. Когда состояние А становится активным, то В и С становятся активными одновременно. Когда С становится активным, активным может стать или С1 или С2.



Нотации меток состояний

Метки состояний отображаются в верхнем левом углу рамки состояния в следующем основном формате:

name/

entry:entry action

during:during action

exit:exit action

on event_name:on event_name action

Метки состояний начинаются с имени состояния. Кроме того, метка состояния может иметь символ / и одно или несколько ключевых слов. Ключевые слова определяют различные типы действий, связанных с состоянием. Эти ключевые слова приведены в таблице (сокращенная форма представлена полужирным шрифтом):

Приставка	Тип действия
ка	
Entry	Действия при входе в состояние.
During	Действия во время активности состояния.
Exit	Действия при выходе из состояния.
On event_name	Действия при наступлении события event_name.

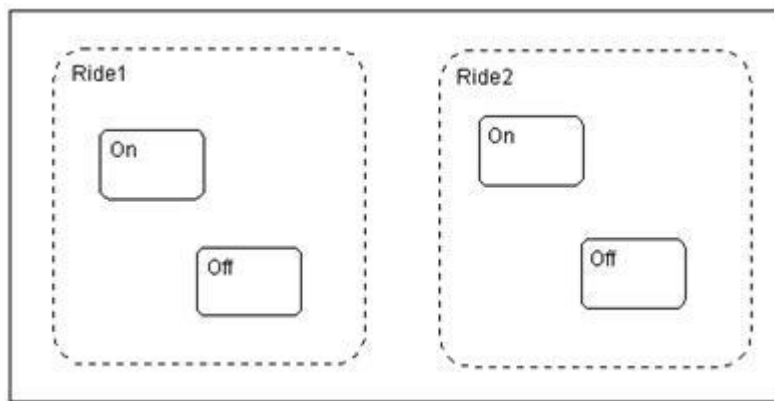
После ввода метки, которая определяет тип действия, Вы должны указать, какие именно действия будут выполняться. Составные действия для каждого типа

разделяются одним из символов: возврат на начало строки; точка с запятой; запятая.

Вы можете определить составные действия типа `on event_name` для различных событий.

Каждое ключевое слово опционально и независимо. Вы можете не определять их вовсе либо определить одно или несколько. После каждого ключевого слова ставится двоеточие. За именем состояния может идти / (слэш). Если вы вводите имя и слэш / непосредственно перед действиями, действия интерпретируются как действия при входе в состояние. Сокращенная форма нужна, когда вы определяете только действия при входе в состояние. Действия определяются в метках состояний, составляющих часть языка действий Stateflow.

Правильное имя состояния может состоять из букв, цифр и может включать знак подчеркивания (`_`), например, `Transmission` или `Green_on`. Использование иерархии придает гибкость наименованию состояний. Имя, которое вы вводите в качестве части метки, должно быть уникальным по отношению к именам предков в его иерархии. Имя сохраняется в словаре данных в виде полного имени. Несколько состояний могут иметь одинаковое имя, если их полные имена в словаре данных уникальны. Иначе при грамматическом разборе выявится ошибка. Следующий пример показывает, как иерархия поддерживает уникальность имен состояний.



Каждое из этих состояний имеет уникальное имя в иерархии диаграммы Stateflow. Хотя имена, являющиеся частью метки, не уникальны, когда иерархия будет добавлена в имя в словаре данных, результат будет уникальным. Полные имена этих состояний, как они выглядят в словаре данных, показаны ниже:

Ride1.On

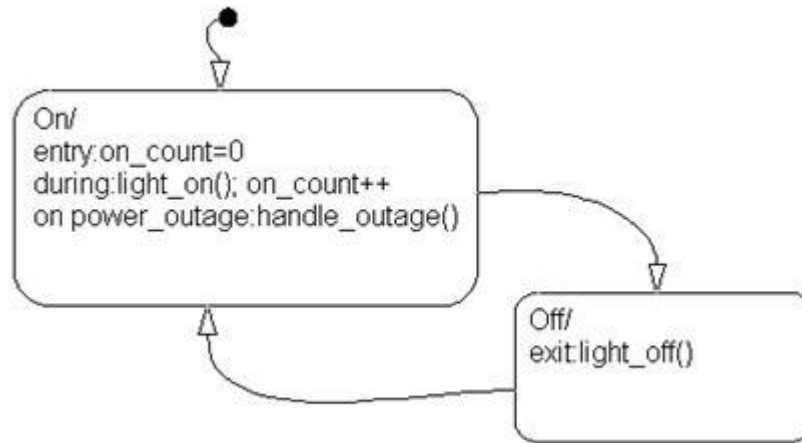
Ride1.Off

Ride2.On

Ride2.Off

Пример метки состояния

Следующий пример демонстрирует компоненты метки состояния.



В этом примере имена состояний - On и Off. Имя состояния формирует первую часть метки состояния. Далее указаны следующие действия.

Entry Action (Действие при входе в состояние). Состояние On имеет действие при входе `on_count=0`. Это значит, что значение `on_count` устанавливается равным 0, когда выполняется вход в состояние On.

During Action (Действие во время активности состояния). Состояние On имеет два During действия: `light_on()` и `on_count++`. Эти действия выполняются, когда выполняются текущие действия состояния On.

Exit Action (Действие при выходе из состояния). Состояние Off имеет действие при выходе `light_off()`. Это действие выполняется, когда выполняется выход из состояния Off.

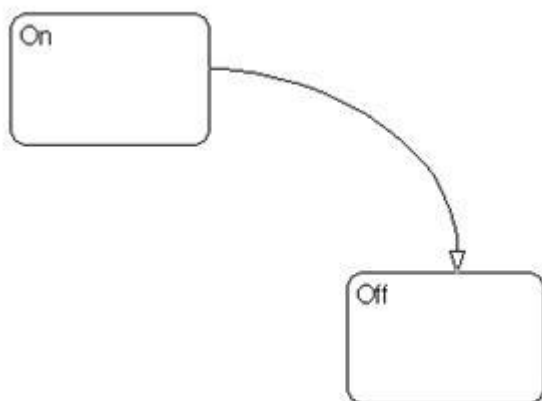
On Event_Name Action (Действие при наступлении события Event_Name). Состояние On имеет такое действие. Когда событие `power_outage` произойдет, действие `handle_outage()` выполнится.

Переходы

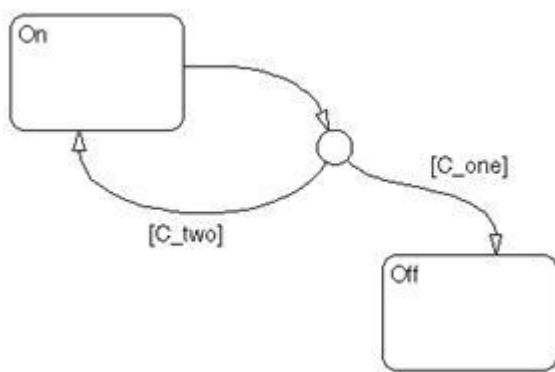
Реактивная система меняет одно состояние на другое при помощи объекта, называемого переходом.

Что такое переход?

Переход - это линия со стрелкой, соединяющая один графический объект с другим. В большинстве случаев переход представляет скачок системы из одного режима (состояния) в другой. Переход соединяет объект-источник с объектом-адресатом. Объект-источник - это место, где переход начинается, объект-адресат - это место, где переход заканчивается. Вот пример перехода из состояния-источника On к состоянию-адресату Off.



Соединения делят переходы на сегменты. В этом случае переход состоит из сегментов перехода и в процессе определения действительности всего перехода каждый сегмент анализируется по отдельности. В следующем примере имеется два сегментированных перехода: один от состояния On к состоянию Off, а другой - от состояния On обратно к состоянию On.



Имеется еще один тип перехода - безусловный переход. Безусловный переход - это специальный тип перехода, у которого нет объекта источника.

Нотации меток переходов

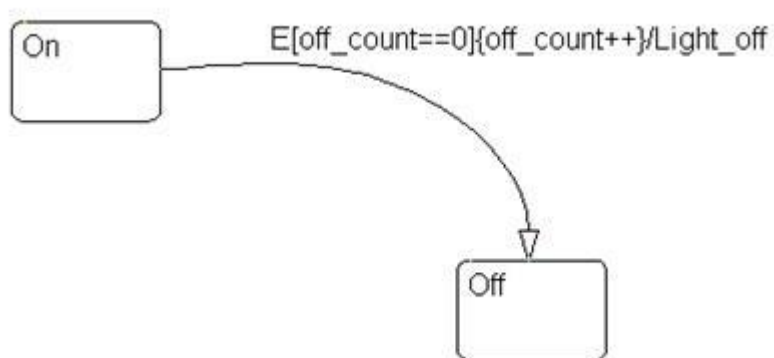
Переходы характеризуются метками. Метка может включать в себя имя события, условие, действие условия и/или действие перехода. Первоначально

переходы помечаются символом (?). Метки перехода имеют следующий основной формат: `event[condition]{condition_action}/transition_action`.

Вы заполняете части `event`, `condition`, `condition_action` и `transition_action`, как показано в примере. Любая часть метки может отсутствовать. Действия, которые вы определяете в метках, составляют часть языка действий Stateflow.

Пример метки перехода

Использование метки перехода иллюстрируется следующим примером.



Переход происходит при наступлении события, но с учетом истинности условия, если оно определено. Определение имени события необязательно. Если имя события не указано, то переход произойдет при наступлении любого события. Составные события определяются посредством использования логического оператора ИЛИ (`()`). В этом примере при наступлении события `E` происходит переход из состояния `On` в состояние `Off`, если при этом будет истинно условие `[off_count==0]`.

Условия - это булевы выражения, которые должны быть истинны для осуществления перехода. Условия заключаются в квадратные скобки (`[]`). В этом примере условие `[off_count==0]` должно быть истинным для того, чтобы произошло действие условия и переход стал возможен.

Действия условий следуют за условиями и заключаются в фигурные скобки (`{}`). Они выполняются тогда, когда условие становится истинным, но перед тем, как переход осуществится. Если ни одно условие не определено, подразумеваемое условие принимается за истинное и действие выполняется. В этом примере если условие `[off_count==0]` истинно, то действие `off_count++` немедленно выполняется.

Действия перехода выполняются после того, как переход стал возможен и при истинности условия, если оно определено. Если переход состоит из сегментов,

действие перехода выполняется только когда становится возможен полный путь перехода. Действия перехода обозначаются символом (\). В данном примере, если условие [off_count==0] истинно и состояния Off достигнуто, осуществляется действие перехода Light_off.

Когда переход возможен

В большинстве случаев переход возможен, когда состояние-источник активно и метка перехода действительна. Переходы по умолчанию отличаются тем, что не имеют состояния-источника. Переход по умолчанию к подсостоянию возможен, когда есть переход к надсостоянию или оно активно. Следующие критерии применим и к переходам, и к безусловным переходам.

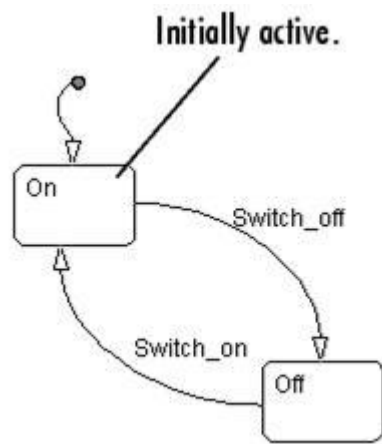
Метка перехода	Переход стал возможен, если...
Событие	Событие произошло
Событие и условие	Событие произошло и условие истинно
Условие	Любое событие произошло и условие истинно
Действие	Любое событие произошло
Не определена	Любое событие произошло

Типы переходов

Нотации Stateflow поддерживают самые различные типы переходов, которые рассмотрены в следующих разделах.

Переходы к последовательным (ИЛИ) состояниям

Рассмотрим простой пример.



Переход от On к Off действителен, когда состояние On активно и событие Switch_off произошло. Переход от Off к On действителен, когда состояние Off активно и событие Switch_on произошло.

Соединение с памятью

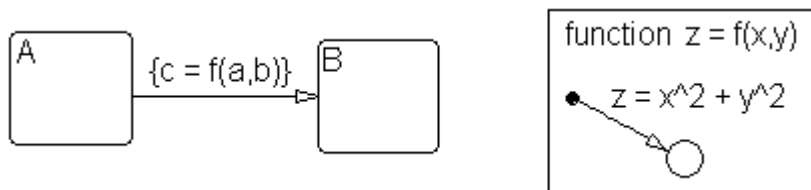
Соединение с памятью запоминает предыдущее активное состояние.

Что такое соединения с памятью?

Соединения с памятью используются, чтобы отметить в Stateflow-диаграмме объекты, где решения принимаются не на основе текущей информации, а опираясь на предысторию. Эта предыстория связана с активностью состояния. Размещение соединения с памятью в надсостоянии показывает, что информация об активности состояния в прошлом используется при определении состояния, которое станет активным в настоящий момент. Соединения с памятью применяются только к тому уровню иерархии, на котором находятся.

Графические функции

Графические функции – это функции, которые определены графом переходов. Наличие графических функций обеспечивает удобство и повышает выразительность языка действий Stateflow. На рисунке приведен пример графической функции и Stateflow-диаграммы с переходом, который ее вызывает.



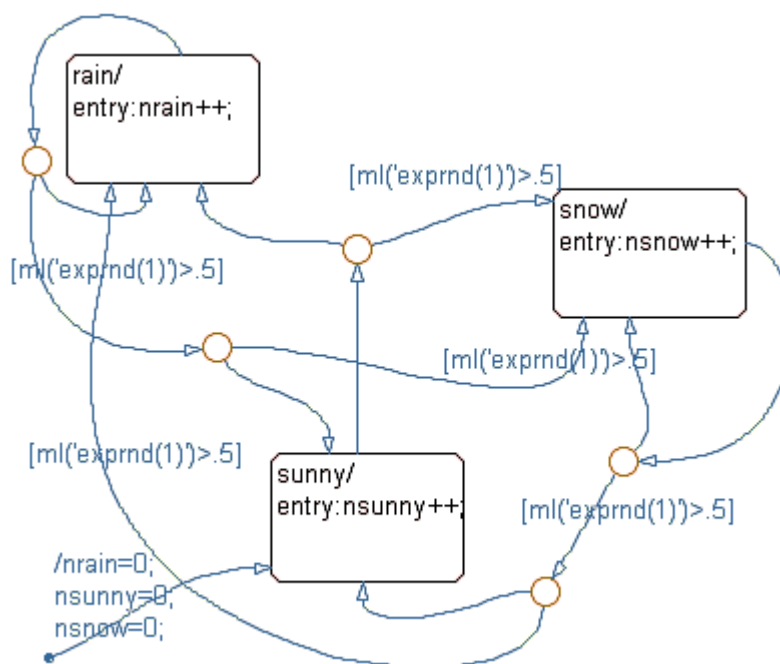
Примеры создания моделей

Модель эргодической цепи Маркова

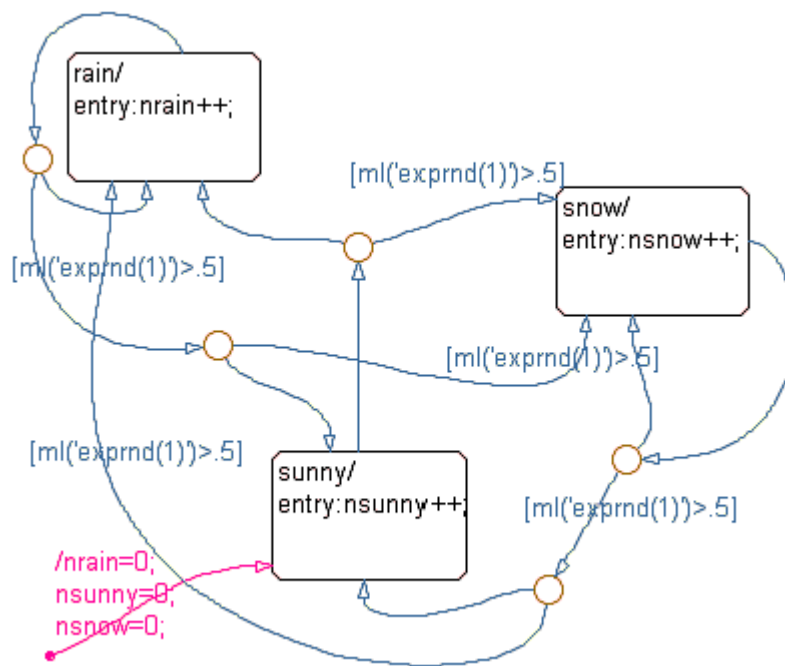
Пример из книги Кемени Дж., Снелл Дж. Кибернетическое моделирование. Некоторые приложения. М.Советское радио, 1972,192 с.

На Земле Оз погода бывает всего трех типов: дождь, солнечно или снег. Если сегодня солнечный день, то завтра будет дождь или снег с одинаковой вероятностью. Если сегодня-дождь (или снег), то половина шансов за то, что такая же погода будет завтра. Если же происходит изменение, то только в половине случаев оно приводит к солнечному дню.

Решение. Образует эргодическую цепь Маркова с тремя состояниями rain, sunny и snow для дождя, солнца и снега соответственно. Ее моделью будет Stateflow-диаграмма.

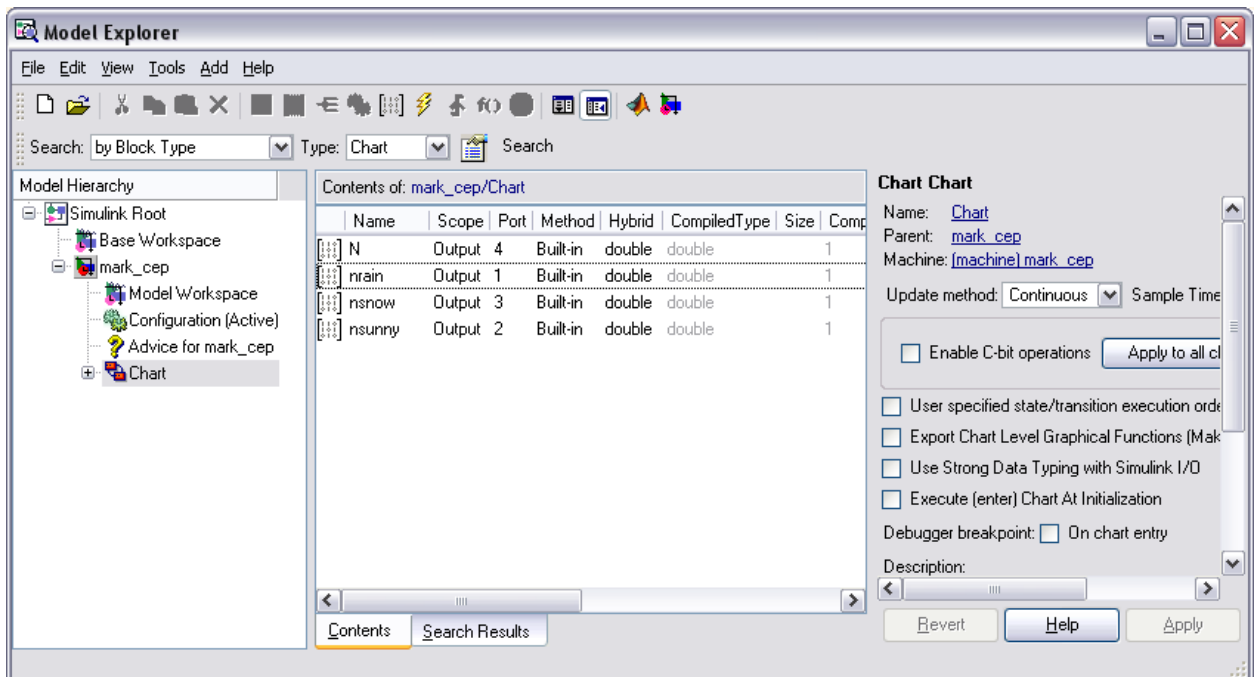
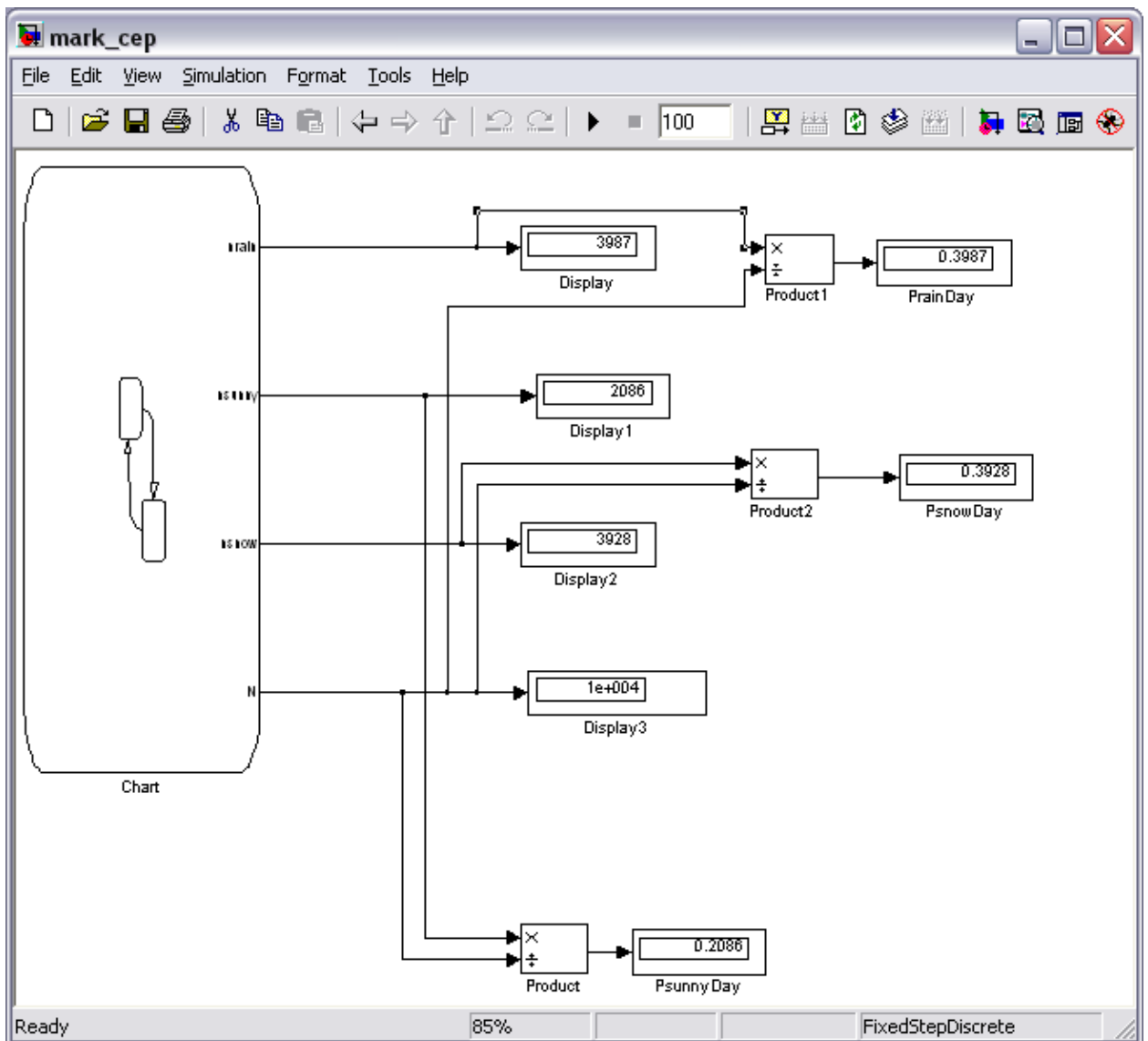


Начальное состояние - sunny, о чем свидетельствует наличие графического объекта переход по умолчанию (Default transition) к состоянию sunny. Этот переход сопровождается действием перехода (Transition action) `/nrain=0;nsunny=0;nsnow=0`. Это действие устанавливает в ноль счетчики количества дождливых, солнечных дней и дней, когда идет снег.

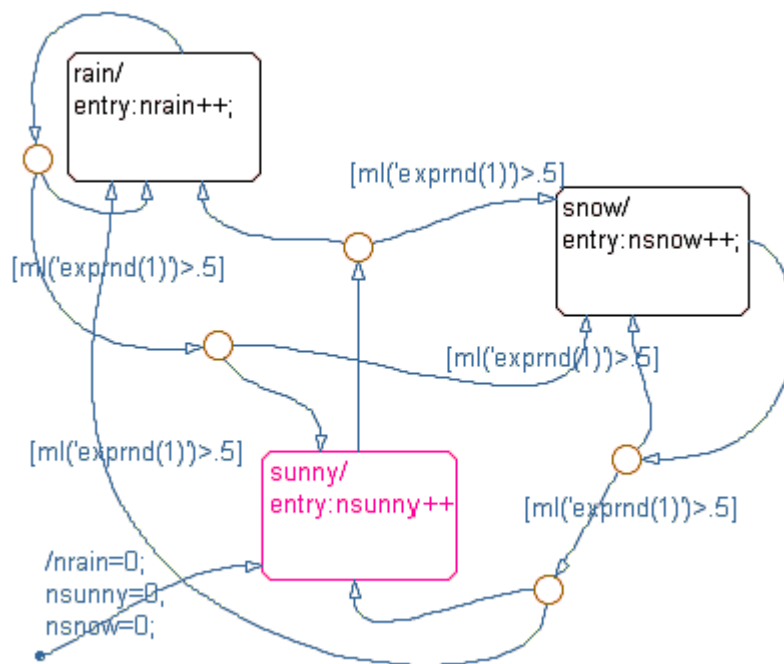


Заметим, что данном случае это действие является избыточным, так как начальные значения этих переменных равны нулю по умолчанию. В этом нетрудно убедиться, открыв проводник Stateflow Explorer и просмотрев графу InitVal.

Итоговый вид модели:



При входе в это состояние выполняется действие `nsunny++`, т.е. количество солнечных дней увеличивается на единицу.

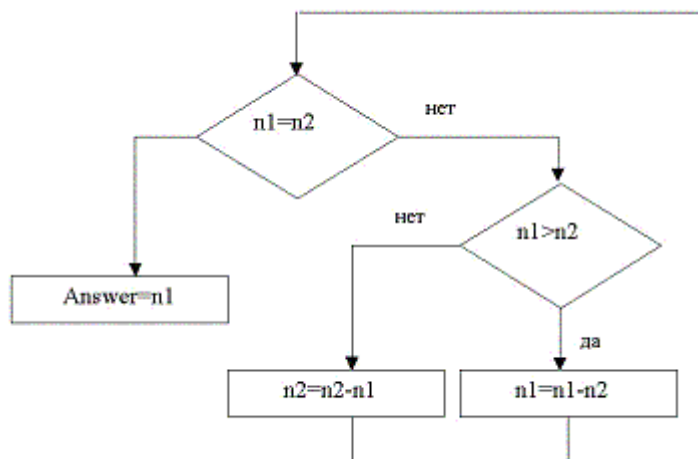


Следующее событие event (смена суток организована в модели при помощи генератора прямоугольных импульсов Pulse Generator) переводит диаграмму в соединяемое подключение Connective Junction, откуда с вероятностью 0.5 диаграмма переходит в состояние snow и с вероятностью 0.5 - в состояние rain. Вероятностный переход основан на использовании условия `ml('exprnd(1)')>.5` (вызов MATLAB-функции `exprnd(1)`, т.е. генерация случайного числа из диапазона (0,1) и сравнение этого числа с числом 0.5). Остальные переходы организованы аналогичным образом в соответствии с логикой, описанной в задаче. Результат работы модели на протяжении 10 лет модельного времени, как это следует из рисунка, дал 728 солнечных, 1514 снежных и 1408 дождливых дней. Аналитическое решение дает вероятность для солнечной погоды $1/5$, а для снега и дождя - $2/5$.

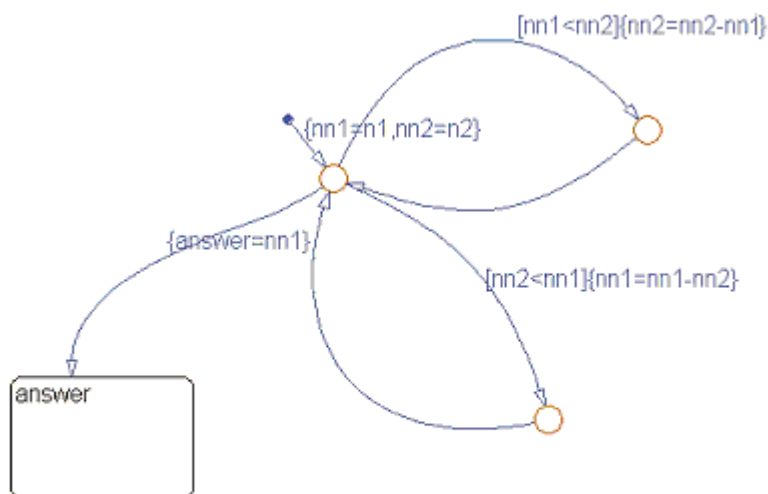
Модель работы алгоритма Евклида (нахождение наибольшего общего делителя)

Моделирование работы компьютера, реализующего известного алгоритма Евклида по нахождению наибольшего общего делителя двух натуральных чисел.

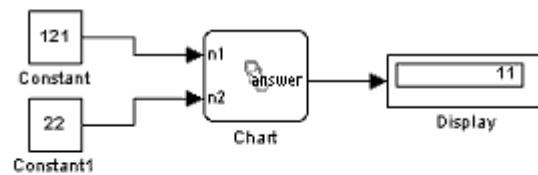
Этот ставший классическим пример разветвляющегося алгоритма можно представить в виде блок-схемы, представленной на рисунке.



Stateflow-моделью работы этого алгоритма может служить показанная на следующем рисунке диаграмма. При моделировании свободно программируемых устройств STATEFLOW - эффективный способ представить общую структуру программного кода как конструкцию в виде условных операторов и циклов



Далее на основе этой диаграммы создаем Simulink-модель, обеспечивающую ввод и вывод информации. Конечный результат работы модели на заданном ей наборе данных показан на рисунке. Наибольшим общим делителем заданных на входе в диаграмму чисел 121 и 22 является число 11.



Практическая часть

Задание 1.

Собрать модель подсчета количества дождливых, снежных и солнечных дней на планете Оз. Для этого:

1. Запустить MatLab.
2. Открыть Simulink и создать новую модель.
3. Объект Chart находится в браузере Simulink, раздел Stateflow.
4. Собрать модель согласно приведенной выше схеме.

Промоделировать систему с заданными начальными параметрами.
Зафиксировать результаты.

Изменить вероятности перехода, зафиксировать полученные результаты.

Задание 2.

Собрать модель нахождения наибольшего общего делителя. Привести примеры работы модели.

Задание 3.

Разработать модель усилителя, удовлетворяющую следующим требованиям.

Входные сигналы:

1. Любой входной сигнал (например синусоида).
2. Время задержки. Задается константной.
3. Коэффициент усиления.

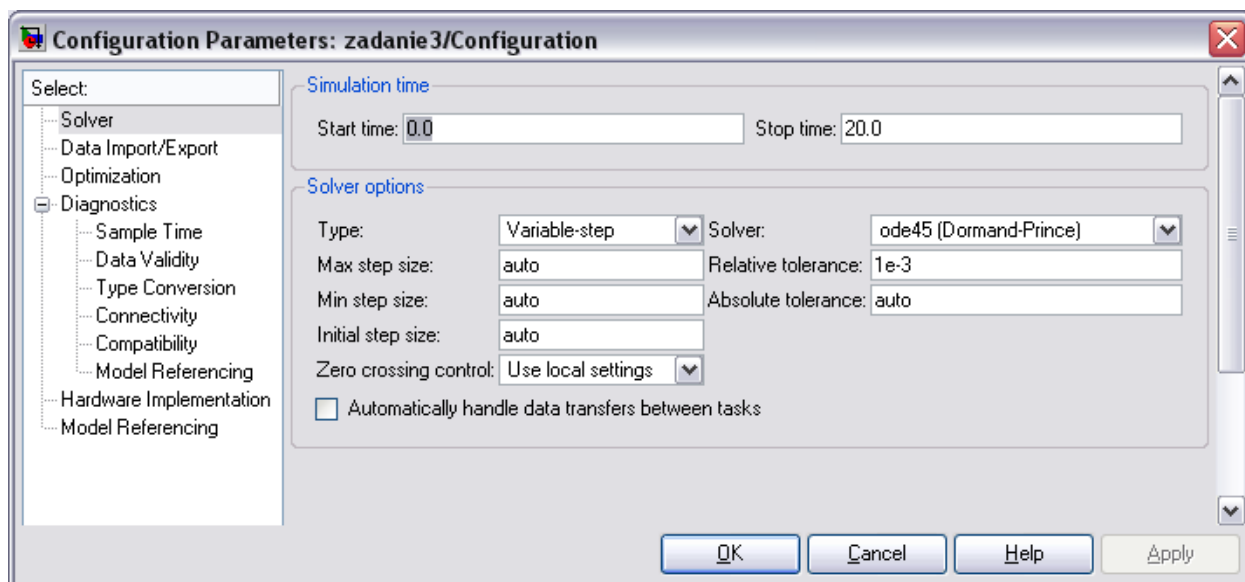
Выходные сигналы:

- 1.

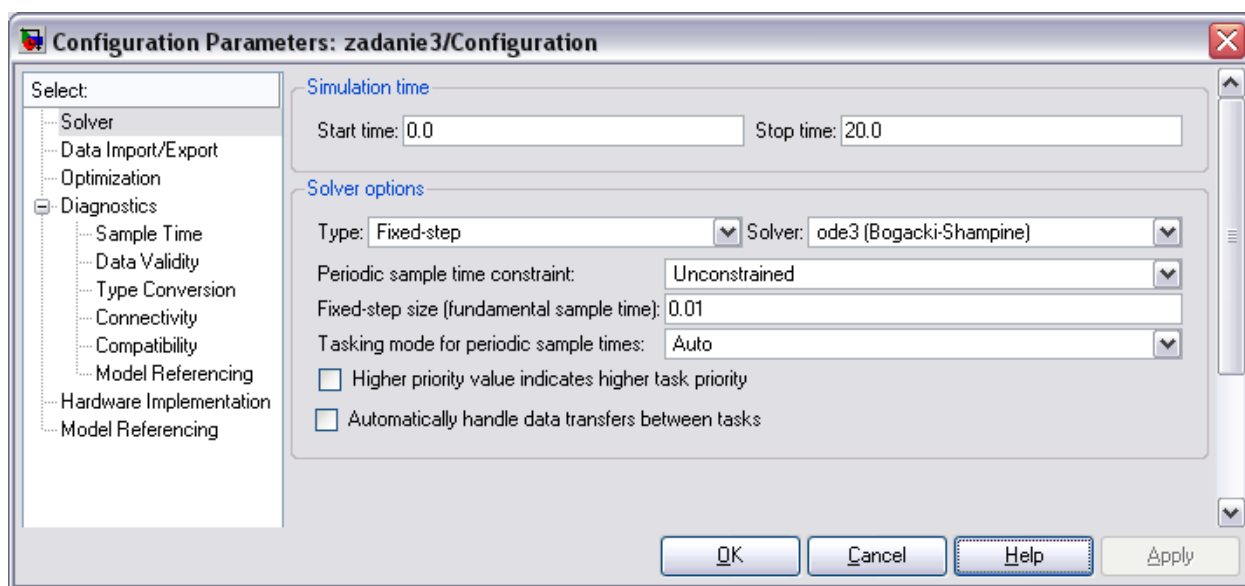
$$U_{вв\text{л}} = \begin{cases} 0, t < t_{\text{зад}} \\ U_{вв} \cdot K_{\text{yc}}, t \geq t_{\text{зад}} \end{cases}$$

Промоделировать. Зафиксировать результаты моделирования.

В параметрах моделирования Simulink изменить «Solver options». По умолчанию стоит:



Изменить параметры на:



Получить новые результаты моделирования. Зафиксировать, промоделировать.