МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра механики, мехатроники и робототехники

КОНЕЧНЫЕ АВТОМАТЫ

Методические указания по выполнению практической и самостоятельной работ по курсу
«Управление мехатронными системами и роботами»
по направлению 15.04.06 - «Мехатроника и робототехника»

Курск 2017

УДК 681.5.01

Составитель: П.А. Безмен

Рецензент
Кандидат технических наук, доцент кафедры механики,
мехатроники и робототехники
Е.Н. Политов

Конечные автоматы: методические указания по выполнению практической и самостоятельной работ по дисциплине «Управление мехатронными системами и роботами» по направлению 15.04.06 - «Мехатроника и робототехника» / Юго-Зап. гос. ун-т; сост.: П.А. Безмен; Курск, 2017. 17 с.: ил. 10, табл. 2.

Содержат сведения по вопросам моделирования работы конечных автоматов в среде Mathworks MATLAB/Stateflow. Рассмотрен процесс создания модели системы управления в среде Stateflow.

Методические указания соответствуют требованиям программы, утверждённой учебно-методическим объединением (УМО).

Предназначены для студентов направления 15.04.06 - «Мехатроника и робототехника» всех форм обучения.

Текст печатается в авторской редакции

Подписано в печать . Формат 60x84 1/16.
Усл. печ. л. . Уч.-изд. л. .Тираж 100 экз. Заказ. Бесплатно.
Юго-Западный государственный университет.
305040, Курск, ул. 50 лет Октября, 94.

Практическая работа № 7
КОНЕЧНЫЕ АВТОМАТЫ

Цель работы

Целью работы является изучение работы конечных автоматов, овладение навыками моделирования конечных автоматов в среде Mathworks MATLAB/Stateflow.

В результате работы у обучающегося формируются компетенции:

ОПК-2 – владение в полной мере основным физико-математическим аппаратом, необходимым для описания и исследования разрабатываемых систем и устройств;

ПК-1 – способность составлять математические модели мехатронных и робототехнических систем, их подсистем, включая исполнительные, информационно-сенсорные и управляющие модули, с применением методов формальной логики, методов конечных автоматов.

Задание

Изучить работу конечных автоматов. Представить конечные автоматы диаграммами MATLAB/Stateflow с двумя входами, двумя состояниями и двумя выходами (рис. 1).

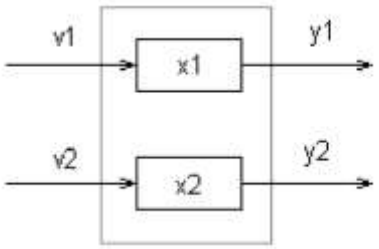


Рис. 1. Конечный автомат с двумя входами, двумя состояниями и двумя выходами

Варианты заданий к практической работе представлены в таблице 1.

Таблица 1 – Варианты алфавитов состояний, входов и выходов для конечного автомата

№ варианта	Входы		Состояния		Выходы	
	1	2	1	2	1	2
1	012345	012345	01	01	ABC	ABC
2	ABCD	ABCD	ABC	ABC	01234	01234
3	0123	0123	01	01	ABCDE	ABCDE
4	ABC	ABC	AB	AB	ABCD	ABCD
5	XYZ	ABC	012	012	ABCD	ABCD
6	ABCDE	ABCDE	01	01	012	012
7	XYZ	XYZ	ABC	ABC	AB	AB
8	0123	0123	01	01	XYZ	XYZ
9	ABCD	ABC	01	01	ABC	AB
10	0123	0123	ABC	ABC	01	01

Содержание отчета

1. Название работы.
2. Цель работы.
3. Содержание работы.
4. Результаты выполнения программы в среде MATLAB/Stateflow на ЭВМ, включающие таблицы переходов.

Методические указания

Подсистема Simulink Stateflow

Stateflow представляет собой графический инструментарий для проектирования сложных систем управления и является самым значительным дополнением к среде Simulink. Stateflow дает возможность моделировать поведение сложных событийно-управляемых систем.

Создание блока управления Stateflow

Процесс создания модели системы управления в Stateflow обычно подразумевает следующие этапы:

- создание новой модели управляемого объекта в Simulink или использование уже существующей модели;
- создание диаграммы в Stateflow;
- добавление к Stateflow-блоку интерфейса событий и данных;
- отладка модели;
- генерация кода.

Для создания Stateflow-диаграммы необходимо сначала создать модель в Simulink или открыть уже существующую. Затем нужно выбрать в библиотеке блок, обозначающий диаграмму Stateflow, и перенести его в свою модель. После двойного щелчка по нему мышью, откроется окно Stateflow (chart) (рис. 2), в котором, пользуясь меню и панелью инструментов, можно описывать сами состояния, связи между ними, определять переменные, условия переключения и другие, необходимые для управления элементы.

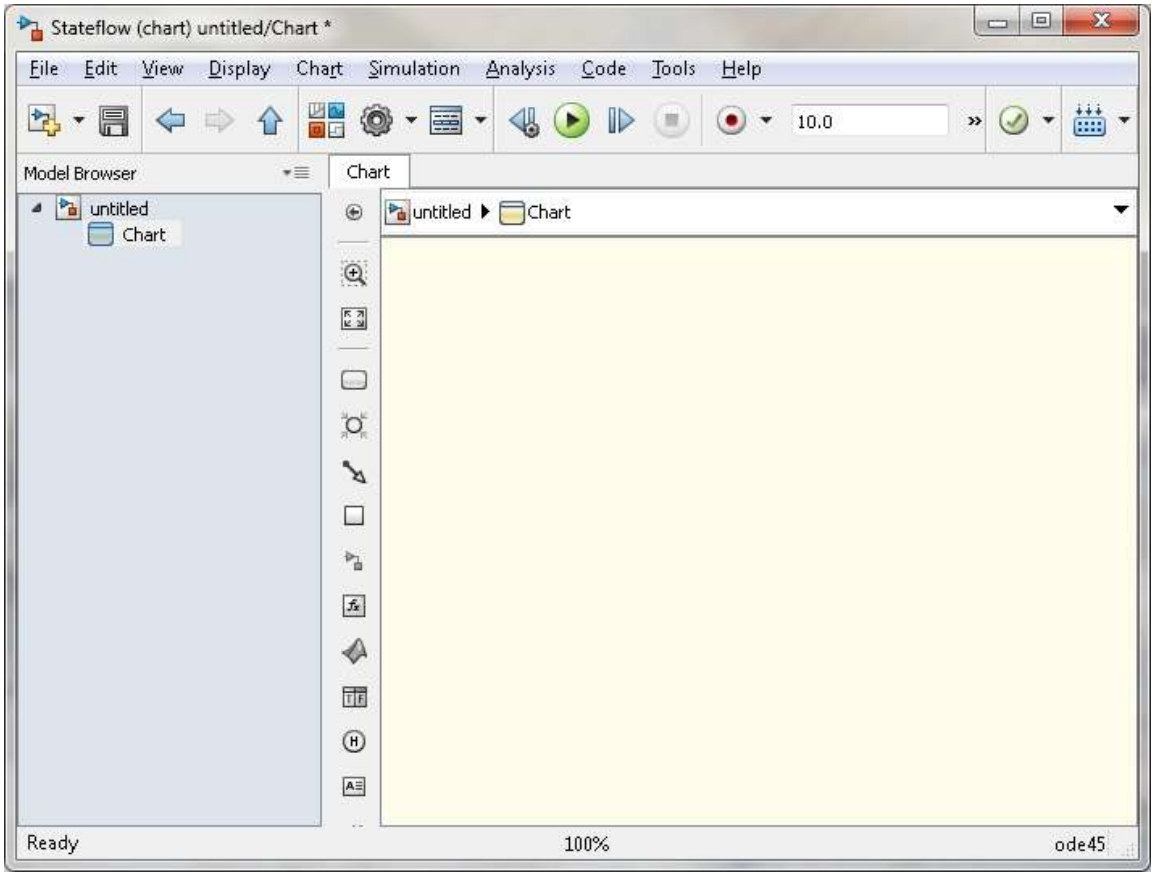


Рис. 2. Основное окно подсистемы Stateflow (chart)

Основные элементы диаграмм Stateflow

На диаграмме Stateflow различают две основные группы элементов: графические и неграфические. Все графические элементы диаграммы представлены на левой панели основного окна - Stateflow (chart) (см. рис. 2), графического редактора карт состояний. Это - **state** (состояние), **default transition** (переход по умолчанию), **history junction** (переход в последнее активное состояние) и **connective junction** (соединительный переход). Неявно на диаграмме задается еще один графический элемент – **transition** (переход). Состояния могут быть простыми и сложными (то есть имеющими иерархическую структуру), они также могут объединяться в цепочки состояний, функционирующих параллельно (и независимо друг от друга) и последовательно. В число графически непредставимых элементов входят **data** (переменные) и **event** (событие).

В основе Stateflow лежит специальная форма представления гибридного поведения – карта состояний. Основные элементы карты состояний - это состояние (state) и переход (transition).

State

Каждое состояние описывает один режим работы событийно-управляемой системы. Состояние становится активным, если получает значение "истина" условие перехода (срабатывает переход), ведущего к нему, или если это состояние является начальным. Каждое состояние на диаграмме Stateflow имеет "родителя", которым, по умолчанию, является сама диаграмма Stateflow.

Для создания графического образа состояния необходимо, выбрав на панели кнопку ▭ щелкнуть кнопкой мыши в том месте, где его необходимо разместить.

Состояние имеет текстовые метки, которые определяют действия, выполняемые во время его активности. Имя состояния

вводится первым. При описании самого состояния могут быть определены следующие действия:

- *entry* – действие, выполняемое на входе в состояние;
- *during* – действие, выполняемое, пока состояние активно;
- *exit* – действие, выполняемое на выходе из состояния;
- *on: <имя события>* – действие, выполняемое в момент появления события (имя которого указано в угловых скобках) при условии, что система будет находиться в данном состоянии.

Создание подсостояний

Подсостояние – это состояние, которое может быть активно, только если активно состояние, называемое его родителем. Для того, чтобы создать подсостояние, необходимо, выбрав кнопку State, щелкнуть кнопкой мыши в поле того состояния, которое должно быть родительским, причем вложенность подсостояний может быть произвольной (рис. 3). Для того чтобы сменить родителя, следует "подхватить" мышью необходимое подсостояние и "перетащить" его из поля предыдущего родителя в поле нового родителя.

Состояние с его подсостояниями можно сгруппировать в единое целое. Это удобно при перемещении такого состояния с одного места на другое без нарушения расположения его подсостояний относительно друг друга. Для этого необходимо, щелкнув правой кнопкой мыши, выбрать в появившемся меню пункт Make Contents/Group, или просто дважды щелкнуть левой кнопкой мыши в поле состояния. Сгруппированные состояния Simulink выделяет, обводя утолщенной линией. Разгруппировать состояния можно опять же, дважды щелкнув в поле состояния кнопкой мыши.

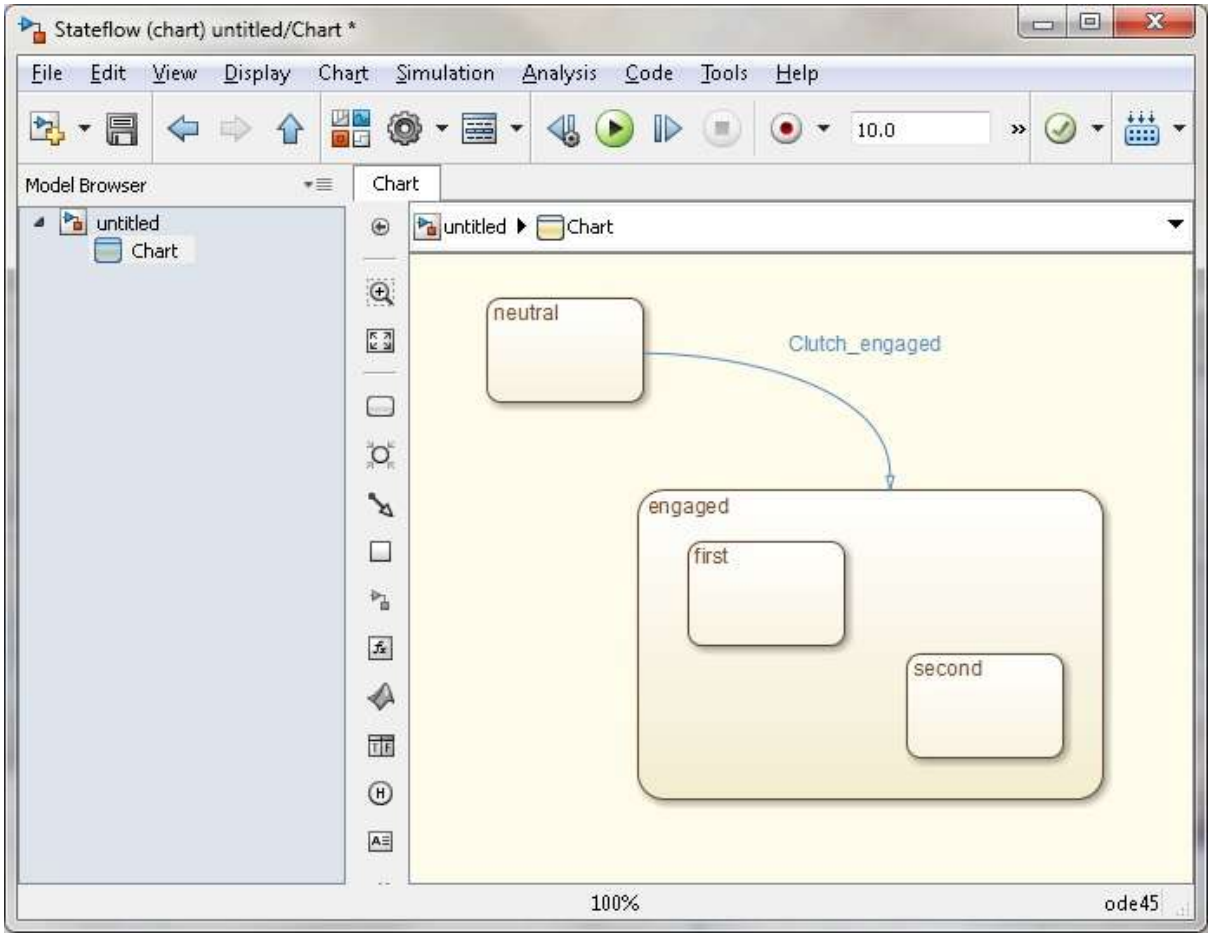


Рис. 3. Иерархическая карта состояний

Параллелизм

Система с параллелизмом допускает существование нескольких активных состояний одновременно. Каждое из них работает независимо от других таких же состояний. Представленные на рис. 4 отдельные параллельные состояния (они обведены пунктирной линией) являются подсостояниями некого сложноорганизованного родительского состояния**.**

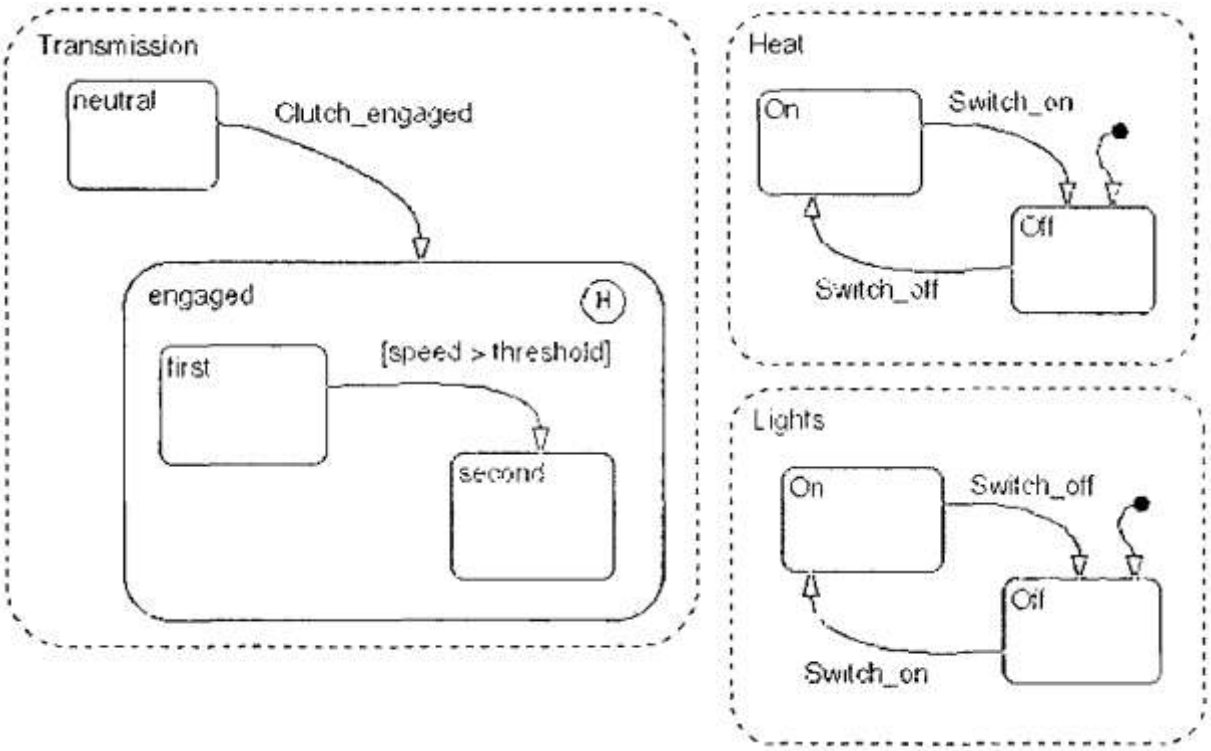


Рис. 4. Фрагмент карты состояний с параллельными процессами

Transition

Переход – это объект, связывающий между собой два состояния. На диаграмме Stateflow он представляется стрелкой, начало которой относится к состоянию-источнику, а конец – к состоянию-адресату. Для того чтобы нарисовать переход, достаточно, установив мышь у состояния-источника, нажать ее левую кнопку и, удерживая ее, протянуть появившуюся стрелку до состояния-адресата.

Переход имеет метку, которая описывает условия срабатывания перехода и выполняемые при этом действия (рис. 5).

Текст метки имеет следующий формат:

event [condition] {condition_action} / transition_action

где:

event – определяет событие, которое инициирует переход. Если событие не указано, то переход инициируется при выполнении булевского выражения. Если событий,

инициирующих данный переход, несколько, то они все указываются, при этом разделяются оператором OR. Например, на рис. 4 метка представляет переход, который срабатывает, если при возникновении события E становится истинным выражение off_count == 0.

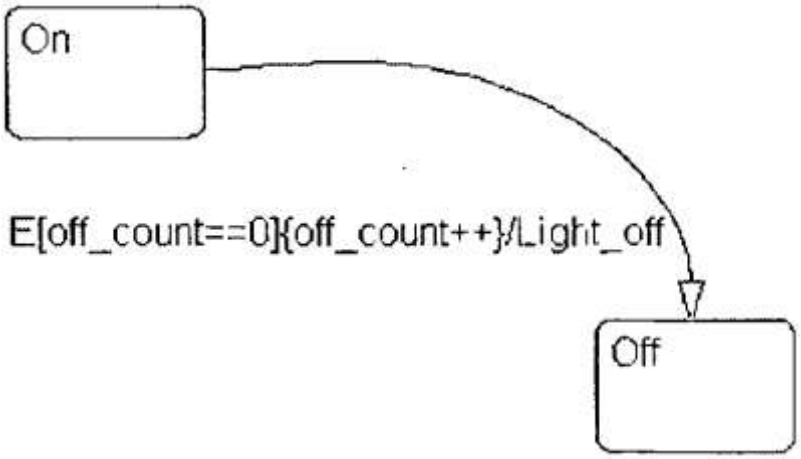


Рис. 5. Переход с приписанным ему условием срабатывания

condition – булевское выражение, инициирующее переход, становясь истинным. В данном примере выражение off_count == 0, как только оно становится истинным, инициирует действие off_count++.

condition_action – действие, выполняемое после того, как стало истинным условие перехода, но до того, как весь переход определился истинным (и определилось состояние-адресат). В данном примере, как только становится истинным условие off_count == 0, мгновенно выполняется действие off_count++.

transit*ion action* – определяет действия, совершаемые при переходе, если все описанные ранее условия истинны и уже определено состояние-адресат. В данном примере действие Light_off выполняется, если условие off_count == 0 истинно и найдено состояние-адресат.

Default transition

Переход по умолчанию – это переход с заданным адресатом, но без заданного источника. Его используют как аналог

начального состояния, а также в сложных состояниях для того, чтобы определить, какое подсостояние становится активным, когда система попадает в данное сложное состояние (рис. 6). Для того, чтобы установить на диаграмме переход по умолчанию, необходимо, выбрав в меню кнопку ↘ установить курсор мыши недалеко от границы того состояния, к которому надо подсоединить переход и нажать её левую кнопку. Переход по умолчанию присоединяют, протянув курсор к границе состояния.

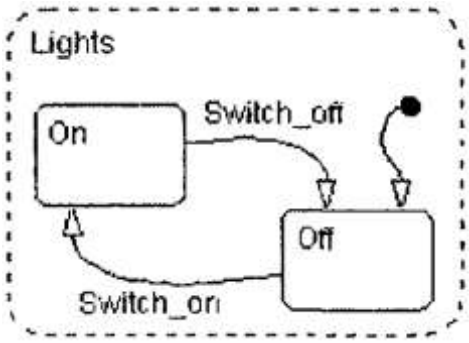


Рис. 6. Установка начального состояния

History junction

Представим себе, что система покидает некоторое сложноорганизованное состояние с множеством подсостояний. Через некоторое время она возвращается в него же, и по определению должна возобновить работу с начального состояния, а нам бы хотелось, чтобы она продолжила работу с предыдущего активного состояния. Для этого нужен переход в последнее активное состояние (history junction). Он срабатывает на том уровне иерархии, где был определен (рис. 7). Переход в последнее активное состояние отменяет все заданные по умолчанию переходы в данном сложном состоянии.

Для того, чтобы создать на диаграмме переход в последнее активное состояние, необходимо, выбрав в меню кнопку Ⓗ, щелчком левой кнопки мыши указать место в поле того сложного состояния, где его требуется разместить. Если дважды щелкнуть левой кнопкой мыши по соответствующей кнопке, то можно устанавливать несколько переходов в последнее активное

состояние, не выбирая соответствующую кнопку каждый раз. Чтобы отменить такой выбор, достаточно щелкнуть правой кнопкой мыши.

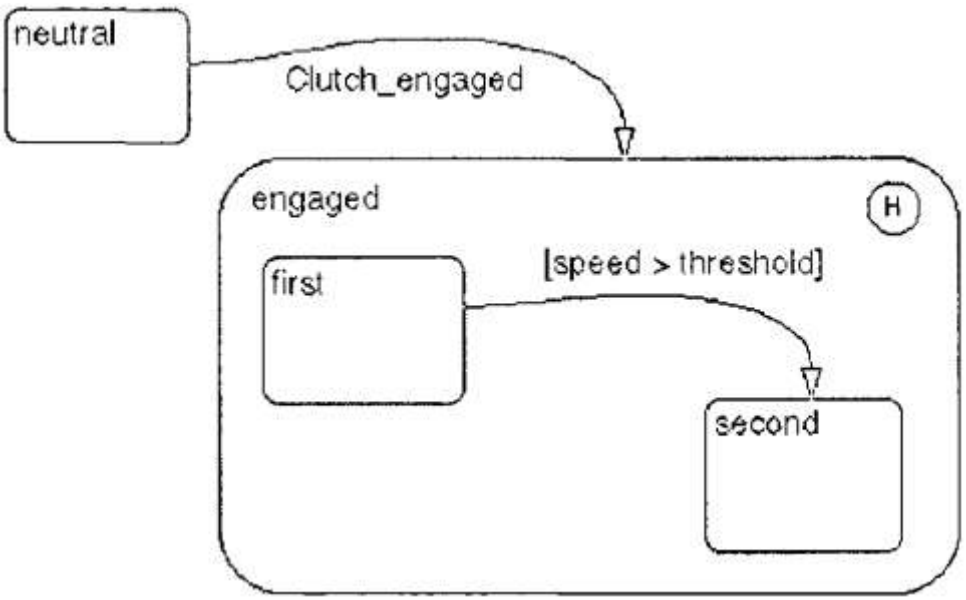
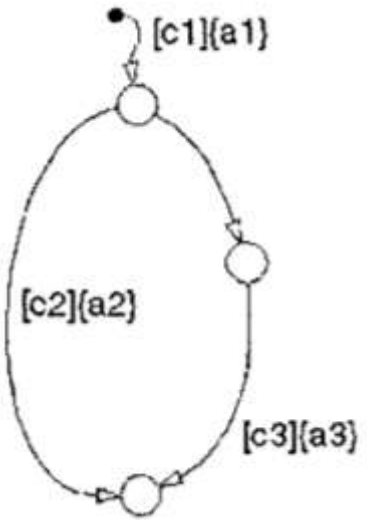


Рис. 7. Карта состояний с переходом в последнее активное состояние

Connective junction

Соединительный переход – это графический объект, который позволяет упрощать диаграммы Stateflow и генерировать более эффективный код. Соединительный переход можно трактовать как состояние, в котором ничего не происходит, а лишь ожидается выполнение условий перехода. Его можно использовать, если у состояния существует два перехода, ожидающих одного и того же события, но срабатывающих при истинном значении различных булевских выражений. Соединительный переход можно помещать на диаграмме теми же способами, что и переход в последнее активное состояние с помощью кнопки меню .

На рис. 8 представлена иллюстрация использования соединительного перехода в случае, когда реакция на произошедшее событие многовариантна и зависит от уточняющей информации.




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

Рис. 8. Использование соединительных переходов

Data

Переменные – это неграфические объекты на диаграмме Stateflow, предназначенные для хранения числовых данных. Переменные можно использовать на любом уровне иерархии. Различают следующие виды переменных:

- входные;
- выходные;
- локальные;
- константы;
- существующие только в течение определенного временного интервала;
- определенные в рабочем пространстве Matlab;
- импортируемые из источника, внешнего относительно диаграмм Stateflow и Simulink;
- экспортируемые адресату, *внешнему* относительно диаграмм Stateflow и Simulink.

Для того чтобы создать входные (выходные и т. д.) переменные в блоке Stateflow, необходимо в окне соответствующей диаграммы Stateflow выбрать соответствующий пункт меню: **Add/Data/Input from Simulink** для создания входной, **Add/Data/Output to Simulink** для создания выходной и **Add/Data/Local для** создания локальной переменной, и в

открывшемся диалоговом окне **Data** (рис. 9) ввести имя переменной и другие её характеристики.

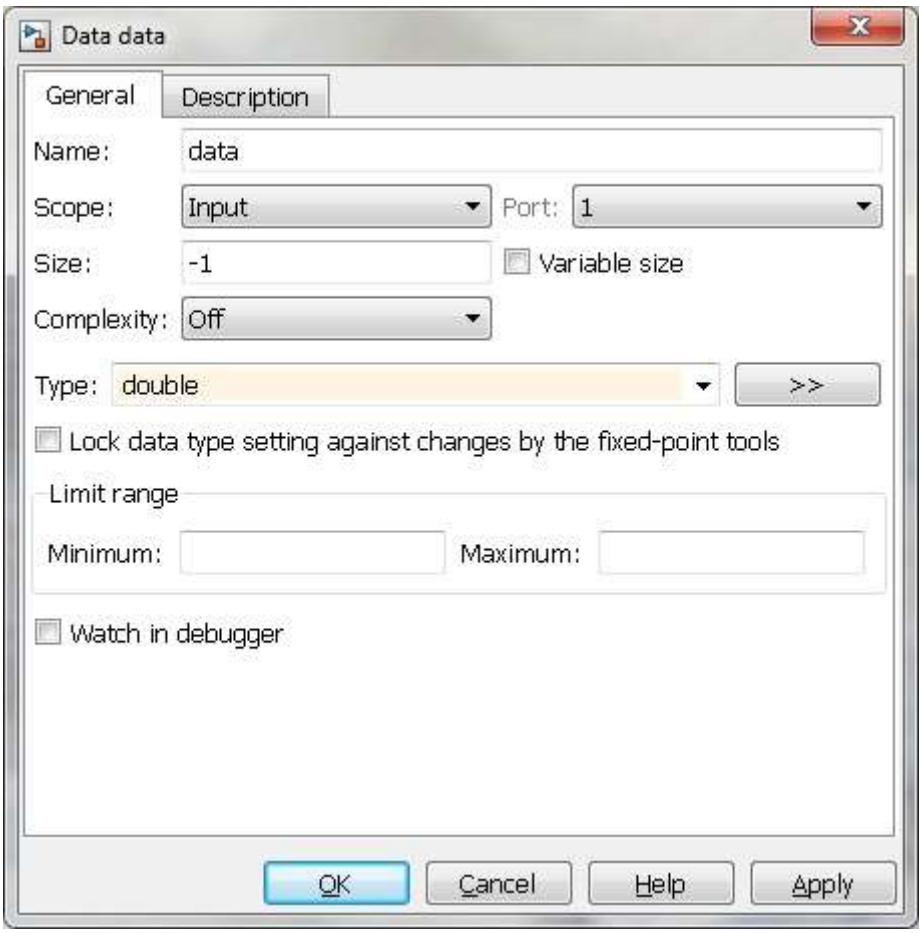


Рис. 9. Диалоговое окно Data

Event

События – это неграфические объекты на диаграмме Stateflow, управляющие диаграммой. Все события, имеющие отношение к диаграмме Stateflow, должны быть определены. Различают следующие виды событий:

- входные;
- выходные;
- локальные;
- импортируемые из источника, внешнего относительно диаграмм Stateflow и Simulink;

- экспортируемые адресату, внешнему относительно диаграмм Stateflow и Simulink.

Для того чтобы создать входные (выходные и т. д.) события в блоке Stateflow, необходимо в окне соответствующей диаграммы Stateflow выбрать соответствующий пункт меню: **Add/Event/Input from Simulink** для создания входного и **Add/Event/Output to Simulink** для создания выходного события, и в открывшемся диалоговом окне **Event** (рис. 10) ввести его имя и другие характеристики.

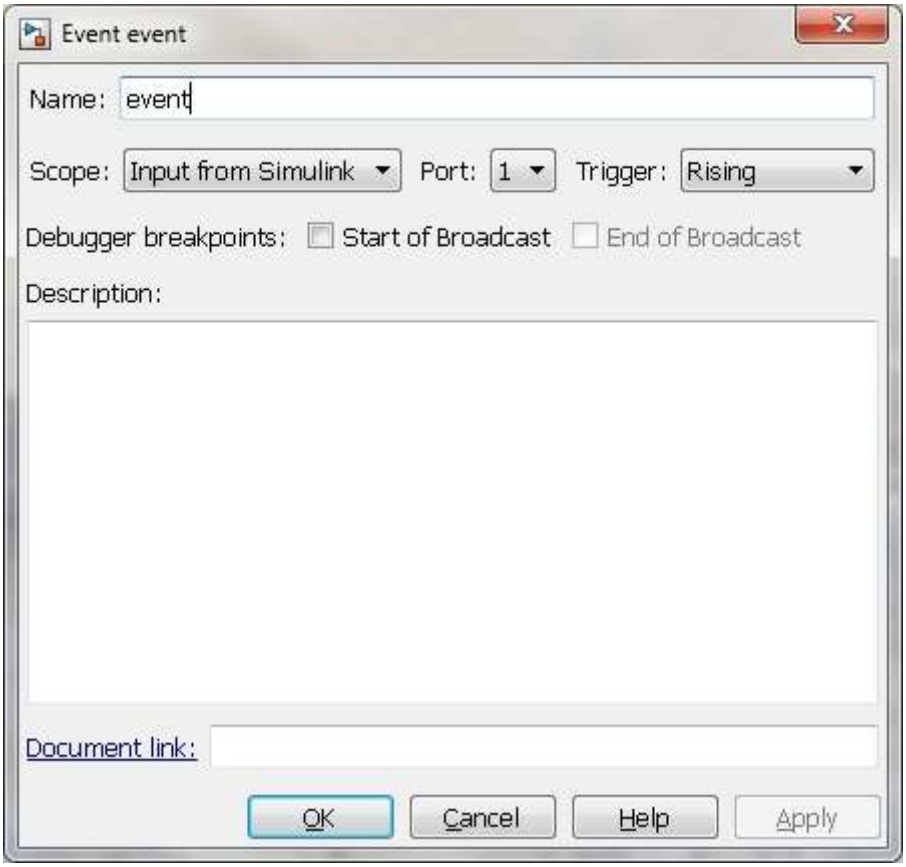


Рис. 10. Диалоговое окно Event

Запуск модели Simulink

После того, как проектирование всех необходимых компонентов модели будет завершено, можно приступать к её исследованию с помощью Simulink.

Прежде всего, надо указать условия проведения вычислительного эксперимента. Для этого следует выбрать в меню окна разработанной модели пункт **Simulation/Parameters** и в открывшемся диалоге указать время, в течение которого будет проводиться эксперимент; выбрать метод интегрирования, если это необходимо, и указать желаемые погрешности, с которыми должно быть получено численное решение.

Прежде чем начинать моделирование (команда **Simulation/Start) необходимо** сохранить информацию о разработанной модели с помощью команды **File/Save.** По команде **Simulation/Start** Simulink прежде всего проверяет правильность разработанной модели, и, если не обнаруживает ошибок, переходит к моделированию.

Задания для самостоятельной работы

Варианты заданий для самостоятельной работы представлены в таблице 2.

Таблица 2 – Варианты алфавитов состояний, входов и выходов для конечного автомата

№ варианта	Входы		Состояния		Выходы	
	1	2	1	2	1	2
1	0123	0123	ABC	ABC	01	01
2	0123	0123	01	01	XYZ	XYZ
3	ABCD	ABC	01	01	ABC	AB
4	ABCDE	ABCDE	01	01	012	012
5	XYZ	XYZ	ABC	ABC	AB	AB
6	XYZ	ABC	012	012	ABCD	ABCD
7	ABCD	ABCD	ABC	ABC	01234	01234
8	ABC	ABC	AB	AB	ABCD	ABCD
9	012345	012345	01	01	ABC	ABC
10	0123	0123	01	01	ABCDE	ABCDE

Контрольные вопросы

1. Назовите отличие конечных автоматов Мура и Мили.
2. Как определяется полный набор сочетаний входов, выходов или состояний?
3. Чем отличаются таблицы с одним и двумя входами?
4. В чем отличие отмеченной таблицы переходов автомата Мура от таблицы переходов Мили?
5. Какие принципы используются при составлении таблицы выходов интерпретирующего автомата?
6. Что служит элементами матрицы соединений?
7. Как строятся графы конечного автомата?

Литература

1. Горбатов, В. А. Теория автоматов / В. А. Горбатов, А. В. Горбатов, М. В. Горбатова. - М.: Астрель, 2008. – 560 с.

2. Блюмин, С.Л. Конечные автоматы Мили и Мура. Методические указания к лабораторное работе №1 для студентов специальности 22.02 «Автоматизированные системы обработки информации и управления» / С.Л.Блюмин, А.М. Корнеев. - Липецк: ЛипПИ, 1993. – 16 с.

3. Брауэр, В. Введение в теорию конечных автоматов / В. Брауэр. -М.: Радио и связь, 1987. – 390 с.

4. Бенькович Е.С., Колесов Ю.Б., Сениченков Ю.Б. Практическое моделирование динамических систем – СПб.: БХВ-Петербург, 2002. – 464 с.