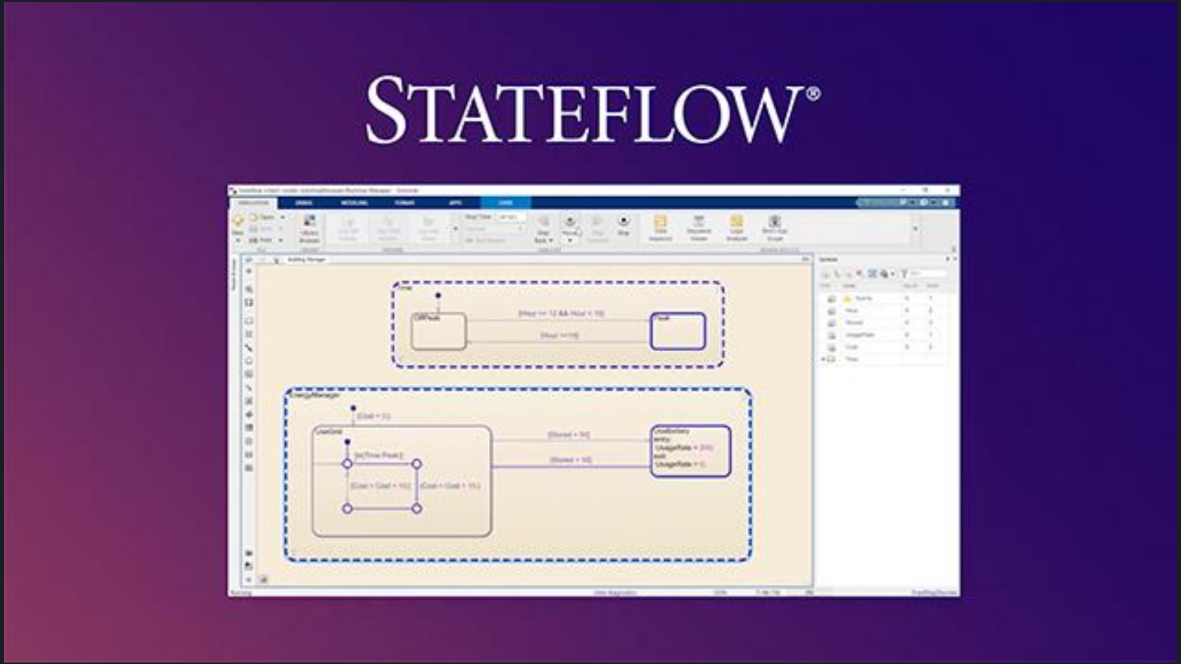
Stateflow Design

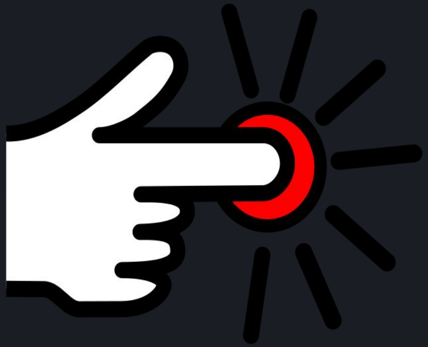


Model-Based Development Program

Stateflow Design

What is a Stateflow?

- Stateflow is a graphical modeling Environment that allow you to design and simulate logic based on **events**, **time-based conditions** or **external signals.**



Stateflow Design

What is a Stateflow?

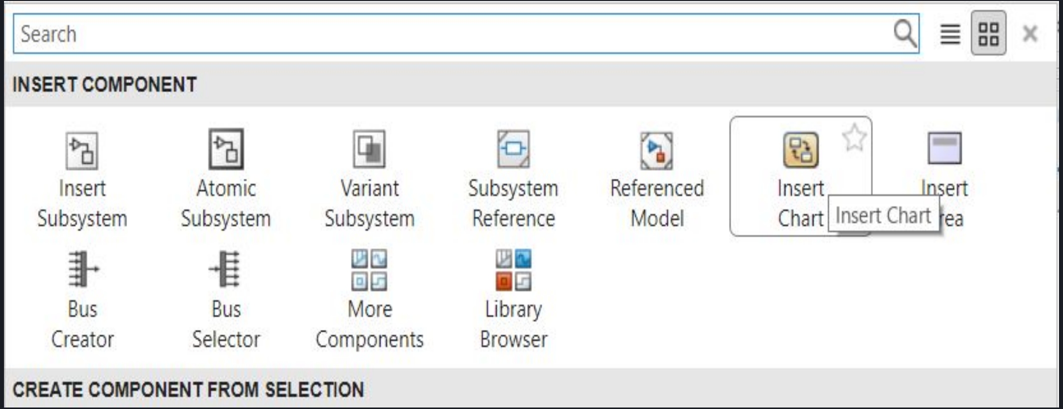
- In stateflow, logical systems are modeled as state machines, which are systems with specific modes

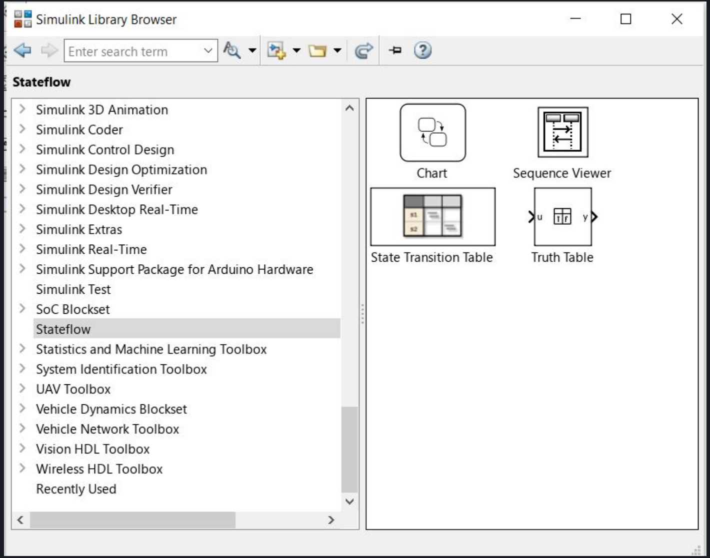
Heat	Cool	Off
Hungry	Tired	Sleeping

Stateflow Design

What is a Stateflow?

- This state flow chart exists as a block on a simulink model.
- Stateflow uses a component called a "chart" to create state diagrams.
- A "chart" contains states and transitions.
- To add a new "chart," you can insert it from the Stateflow library.

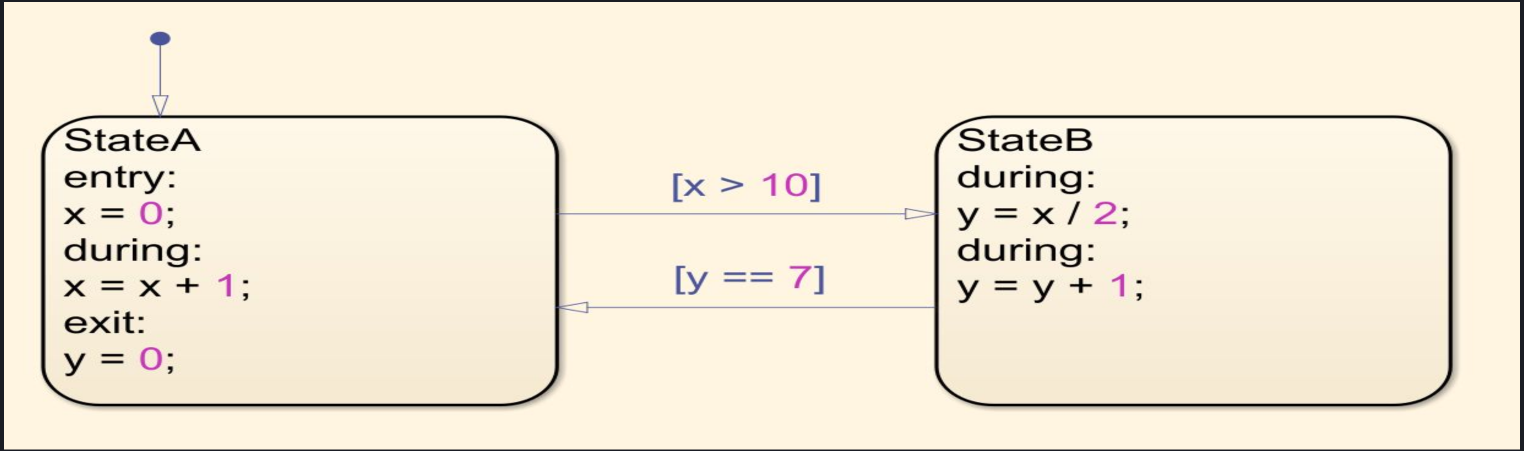




Stateflow Design

What is a Stateflow?

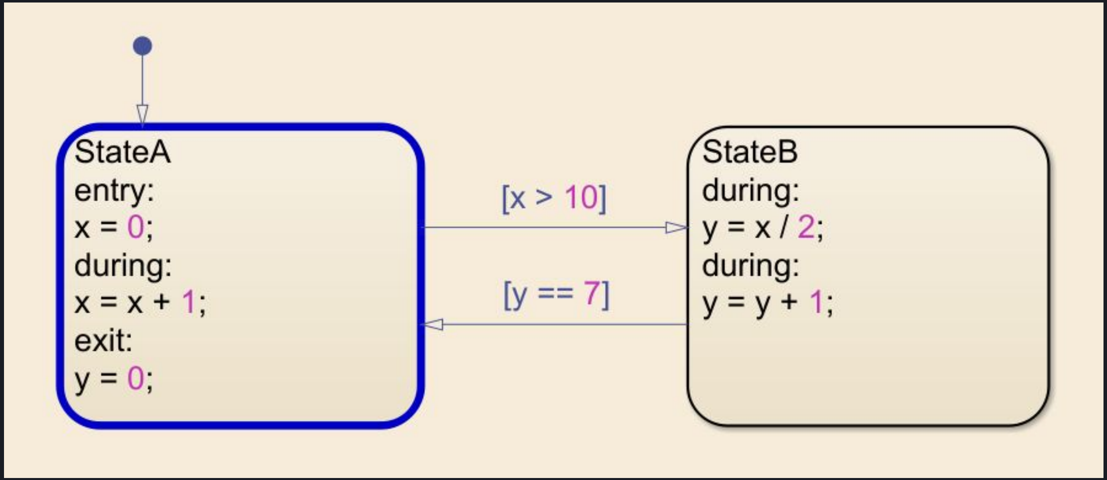
- Stateflow is a MATLAB module for modeling sequential behavior in state machines.
- State machines consist of "states" representing different modes of a system.
- "Transitions" define conditions for moving from one state to another.
- When a state is activated, the system can perform actions associated with that state.



Stateflow Design

What is a state machines?

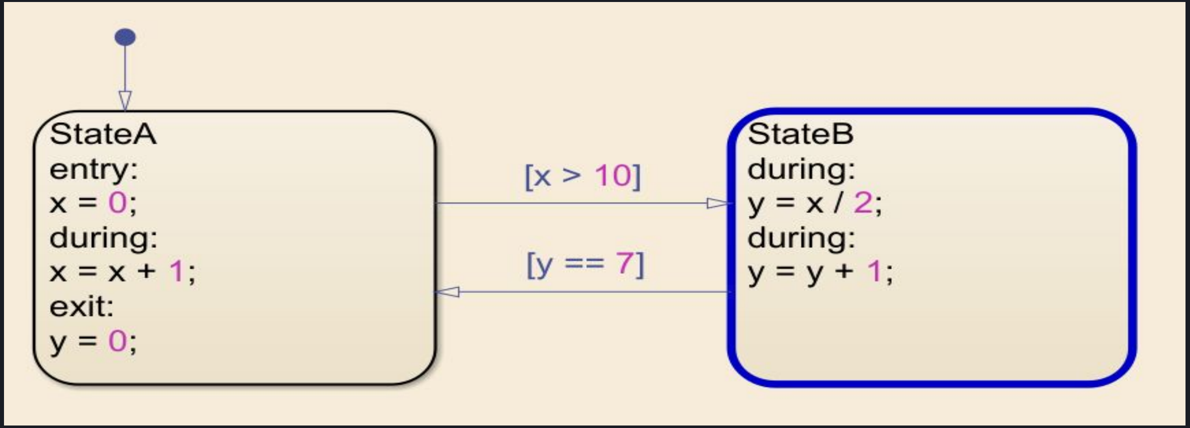
- state is an operating mode of system.
- The active state is the current operating mode.
- a system that has a set of predefined states and can excite in only one state at a time.



Stateflow Design

What is a state machines?

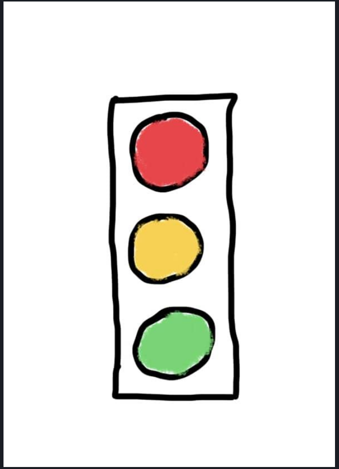
- State machines have a set of rules for how to transition between states.
- we digram these rules using a state transition diagram.
- In this diagram a shape such as a box represents the state. and arrows indicates the transitions.
- The current state of a state machine depend on variables and the previous state.

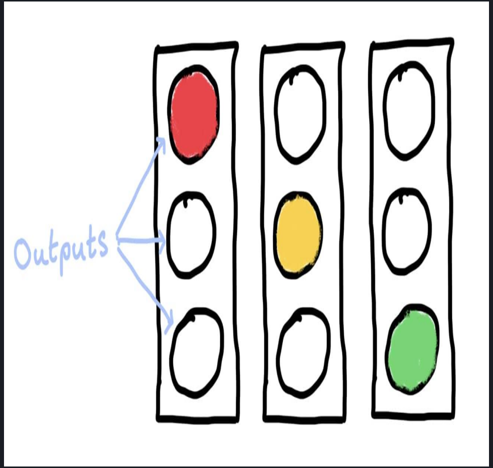


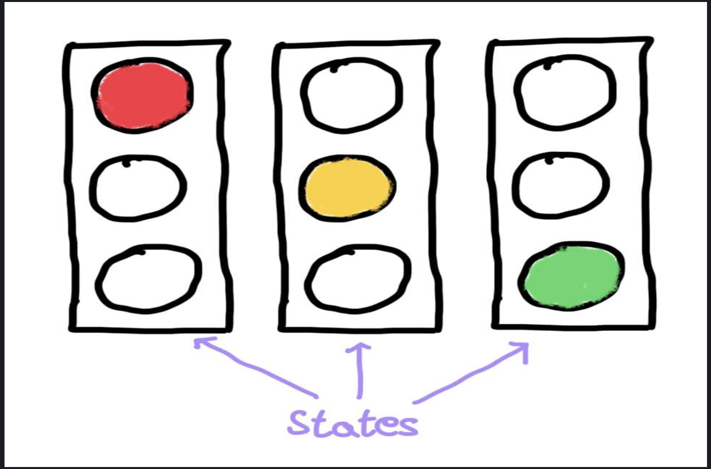
Stateflow Design

What is a state machines?

- There are many examples of state machines.







OS-Academy

Stateflow Design

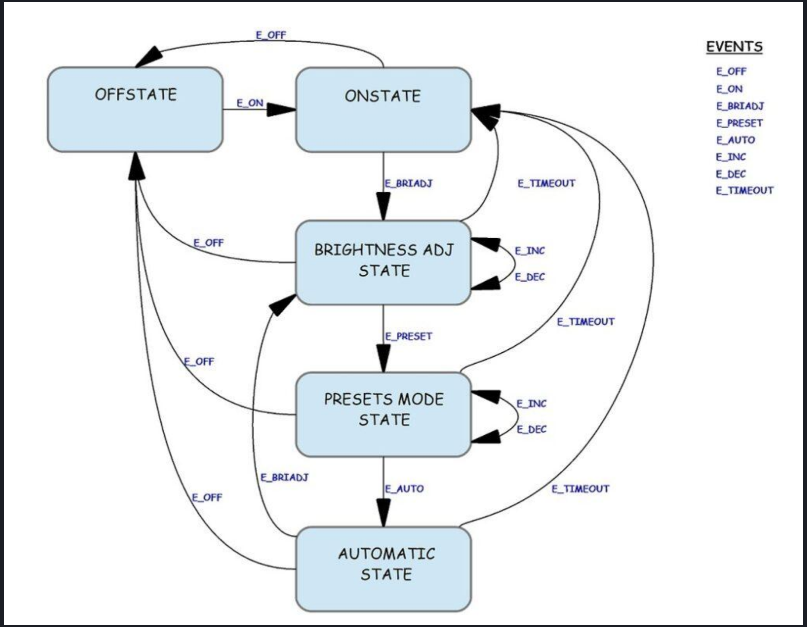
What is a state machines?

- There are many examples of state machines.

Stateflow Design

How to implement state machines?

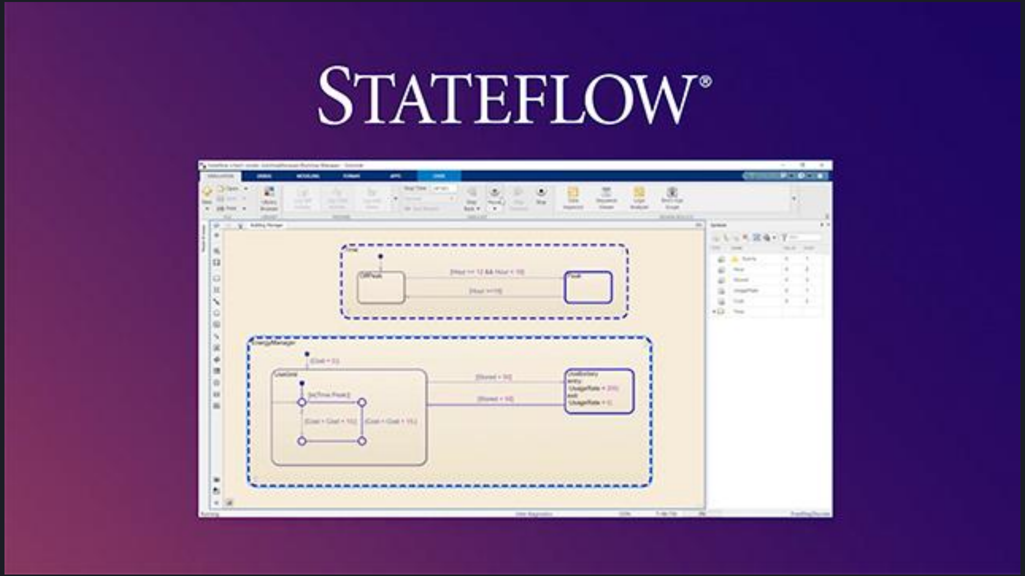
- you could translate, your state machine and transition diagram into code by hand.



Stateflow Design

How to implement state machines?

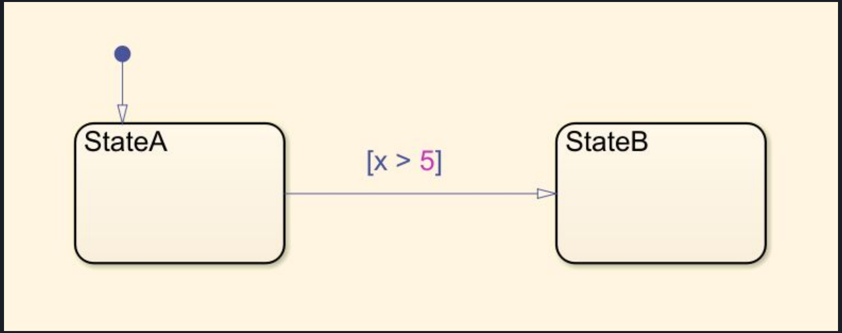
- Stateflow helps you model state machines the way, you would design them on a whiteboard.

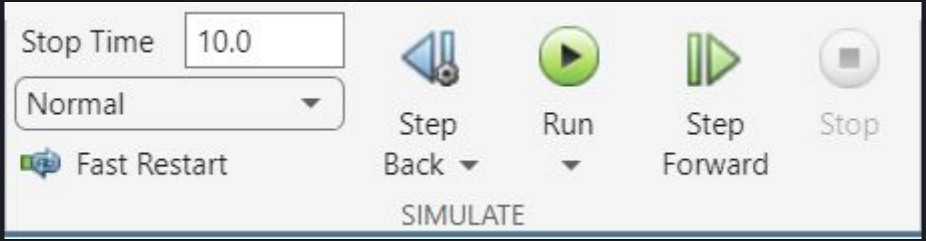


Stateflow Design

How to implement state machines?

- then when you complete, you simply press RUN to simulate the model.

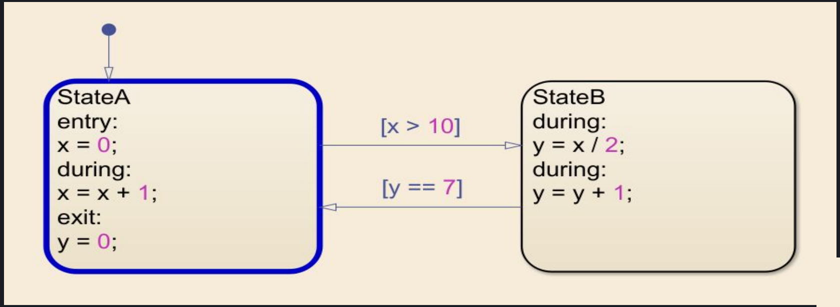


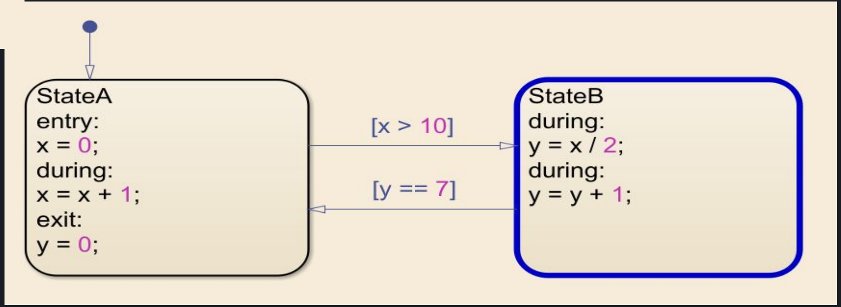


Stateflow Design

How to implement state machines?

- then when you complete, you simply press RUN to simulate the model.

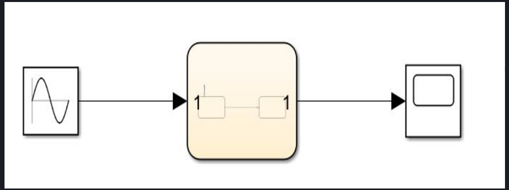


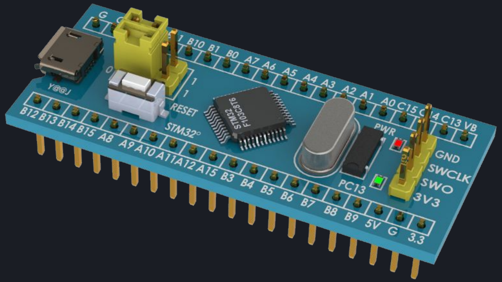


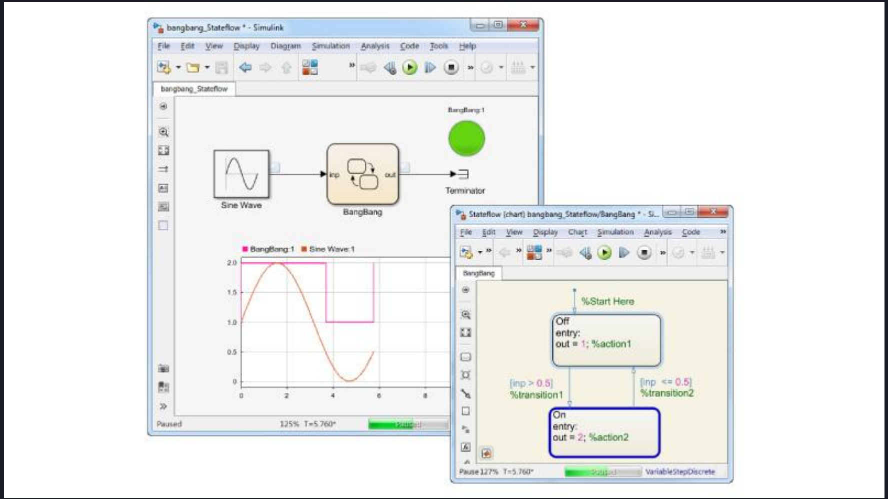
Stateflow Design

How to implement state machines?

- Stateflow also allows you to easily exchange information with simulink in MATLAB or to generate code to run your model on target hardware.



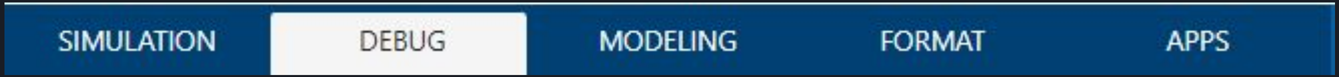


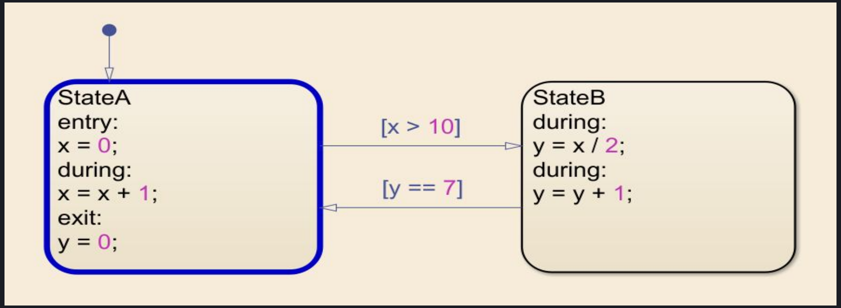


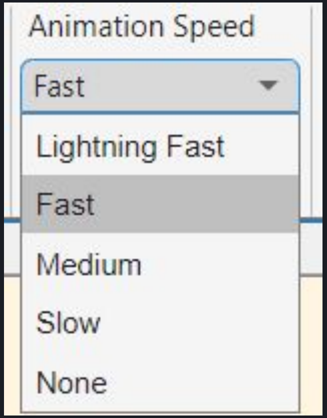
Stateflow Design

How to implement state machines?

- To see the chart behavior more clearly you can change the animation speed of the chart.





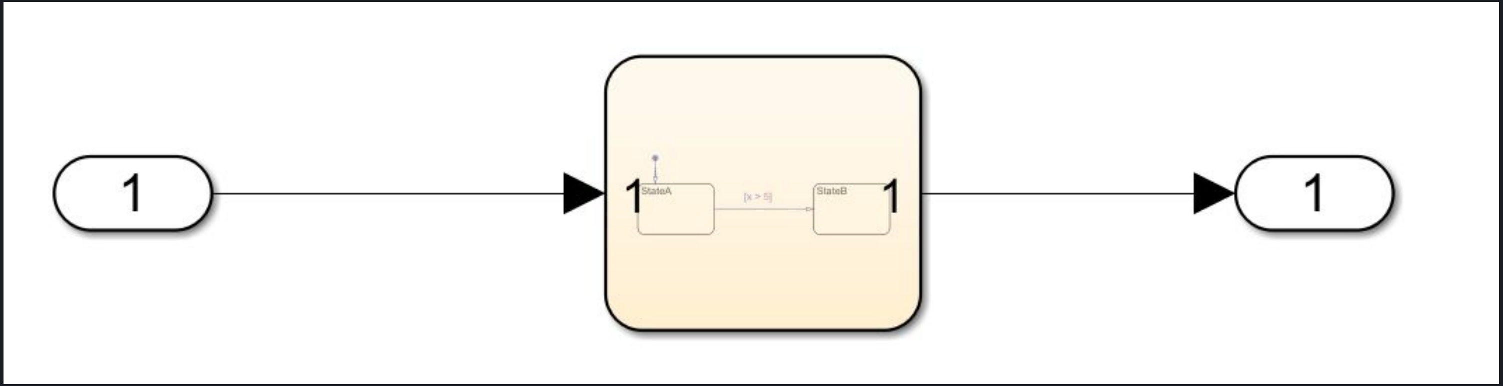

Animation Speed

Fast

Stateflow Design

How to implement state machines?

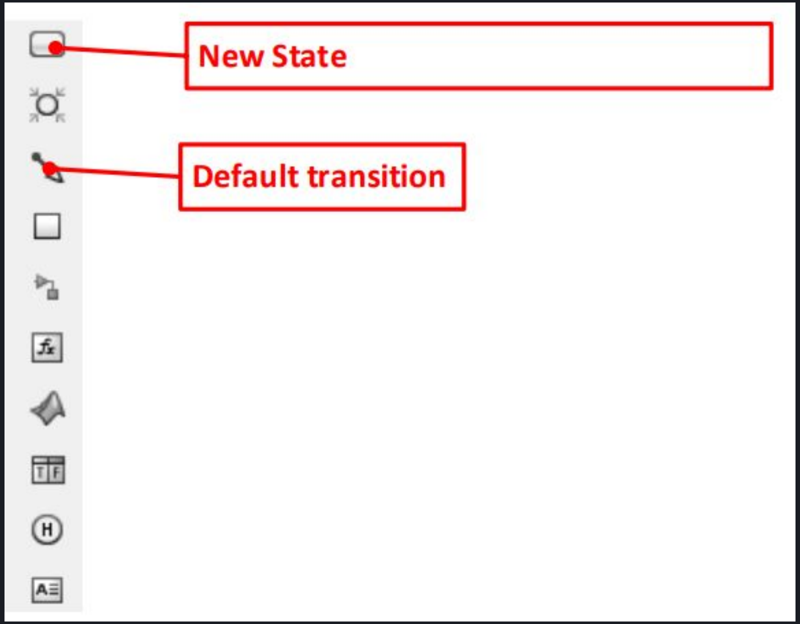
- it receives and sends data via ports, just like other simulink blocks



Stateflow Design

Stateflow Editor

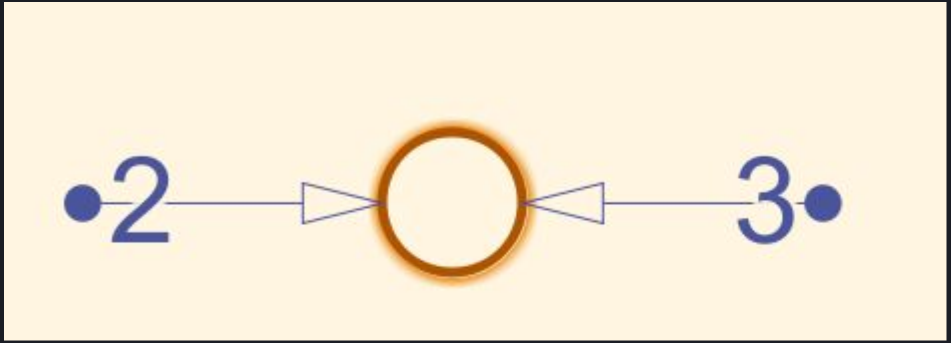
- The Stateflow toolbar appears on the left side of the window.
- It provides controls for creating states and a "default transition.
- Zoom: Mouse scroll wheel
- Fit the chart to screen: Spacebar
- Automatically arrow: ctrl+shift+A

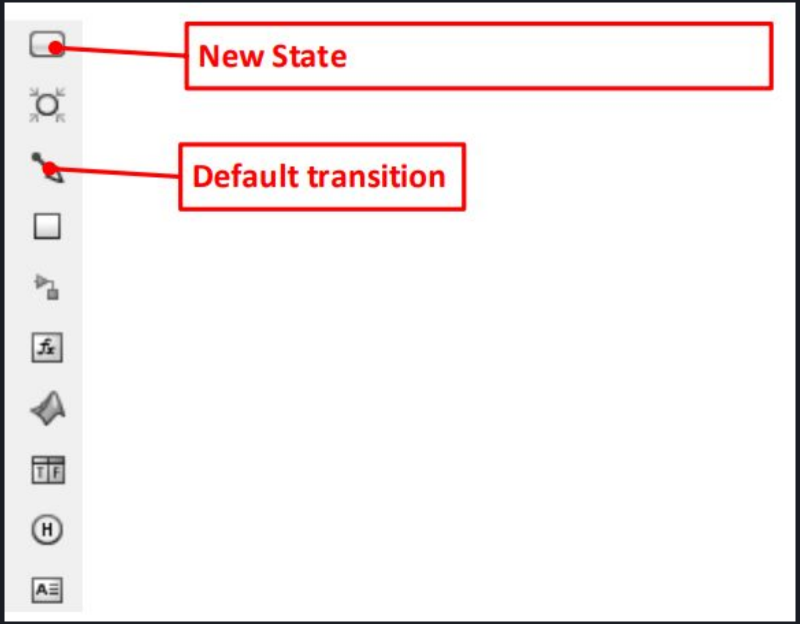


Stateflow Design

Stateflow Editor

- The transition execution order specifics the sequence in which transition are tested.
- for readability, it can be helpful to introduce right angles into the transition arrows using junctions

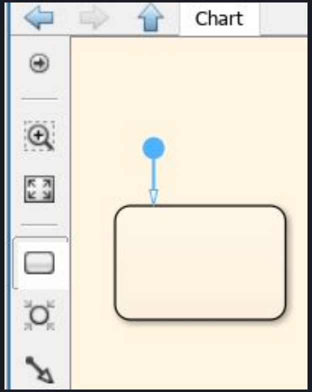


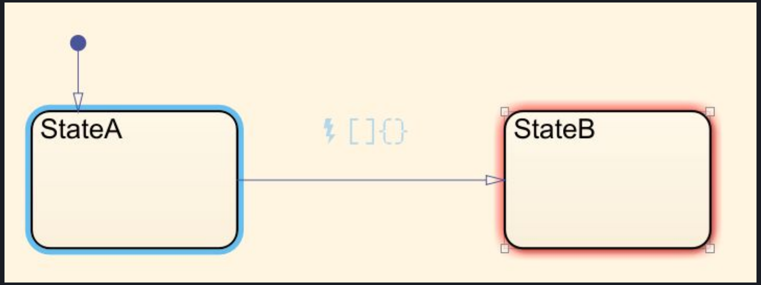


Stateflow Design

Stateflow Editor

- How to add new state?
  - *Drag and drop the state icon from the object palette to add a new state to your chart. States must have valid names, following the same rules as MATLAB variable names.*
  - *Add transitions between existing states by using the left mouse button to click and drag from the edge of one state to the other.*

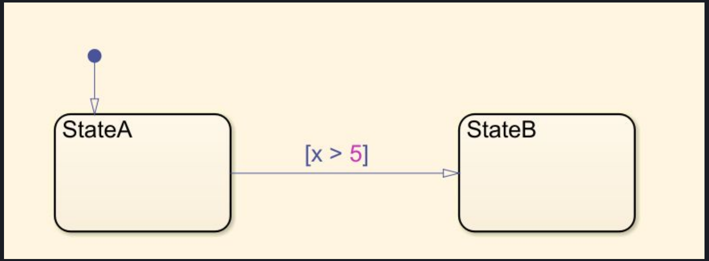


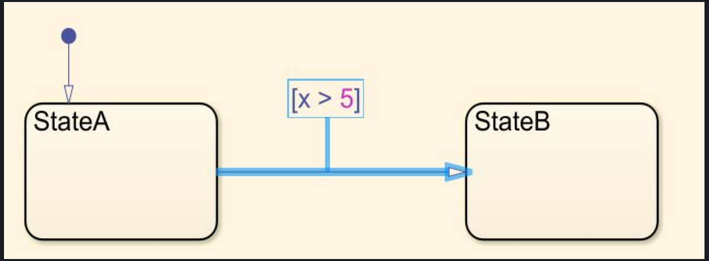


Stateflow Design

Stateflow Editor

- How to add new state?
  - *Double-click on a transition to add a transition condition. You can now enter a MATLAB expression for the condition.*
  - *Move a condition by clicking and dragging it. The associated transition is indicated by a blue line.*

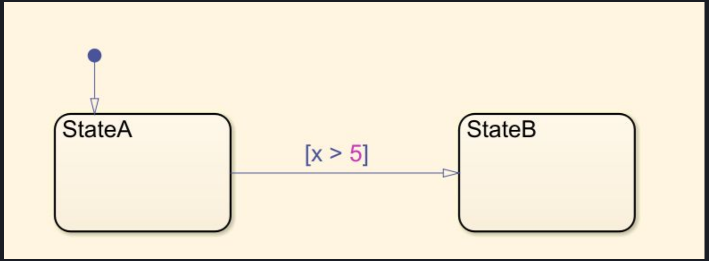


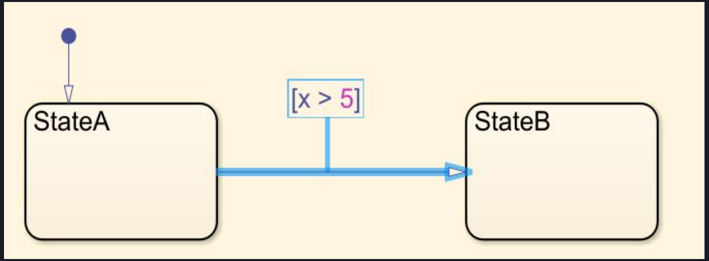


Stateflow Design

Stateflow Editor

- How to add new state?
  - *Double-click on a transition to add a transition condition. You can now enter a MATLAB expression for the condition.*
  - *Move a condition by clicking and dragging it. The associated transition is indicated by a blue line.*

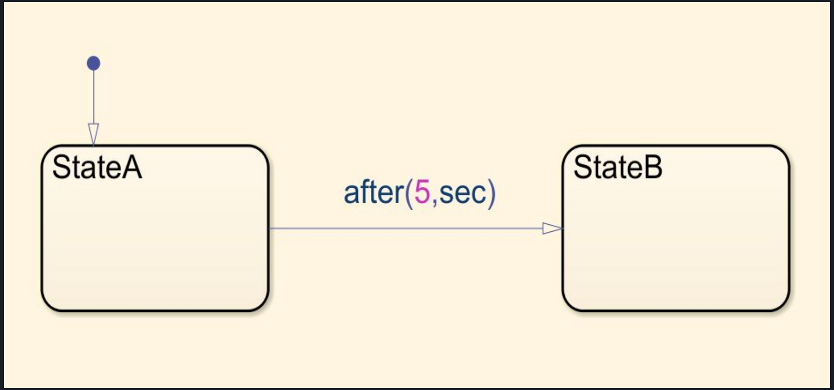




Stateflow Design

Stateflow Editor

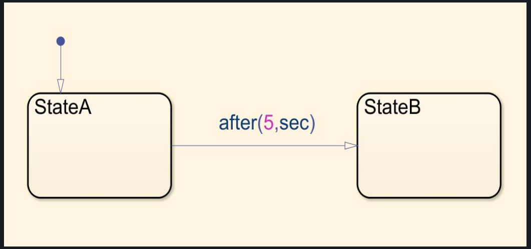
- Temporal Logic
  - Condition that are based on elapsed time
  - a common temporal logic operator is the after(n,sec).
    - seconds : sec
    - milliseconds msec
    - microsecondes: usec
    - simulation time step: tick

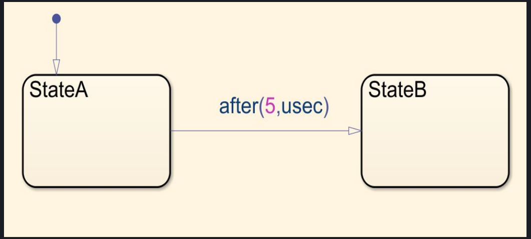


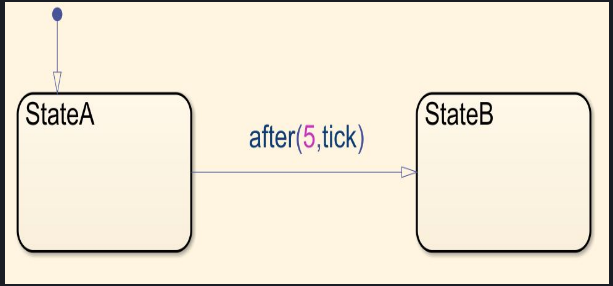
Stateflow Design

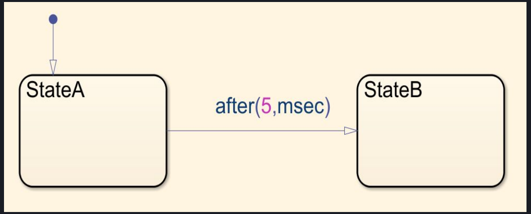
Stateflow Editor

- Temporal Logic
  - Condition that are based on elapsed time
  - a common temporal logic operator is the after(n,sec).
    - seconds : sec
    - milliseconds msec
    - microsecondes: usec
    - simulation time step: tick





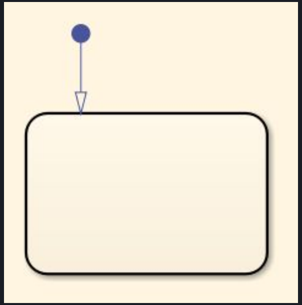


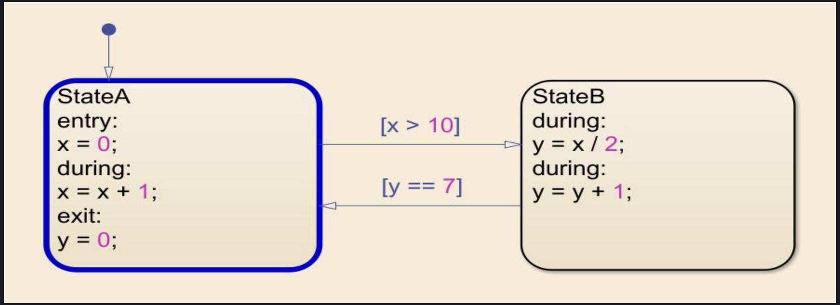


Stateflow Design

Stateflow Editor

- Default Transition
  - This is a default transition it specifics the initial state of the chart. when you add a ste to a blank chart, stateflow automatically add a default transition.
  - when you run this chart, it enter on by default. you may need to change the animation speed to slow to view the animation.

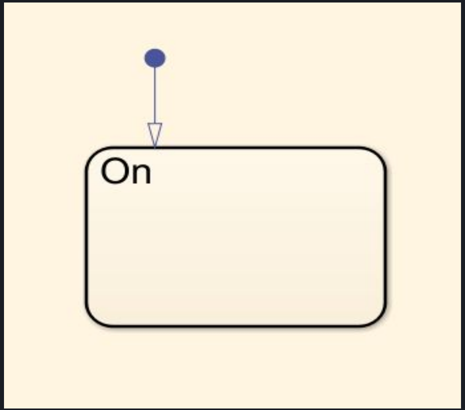


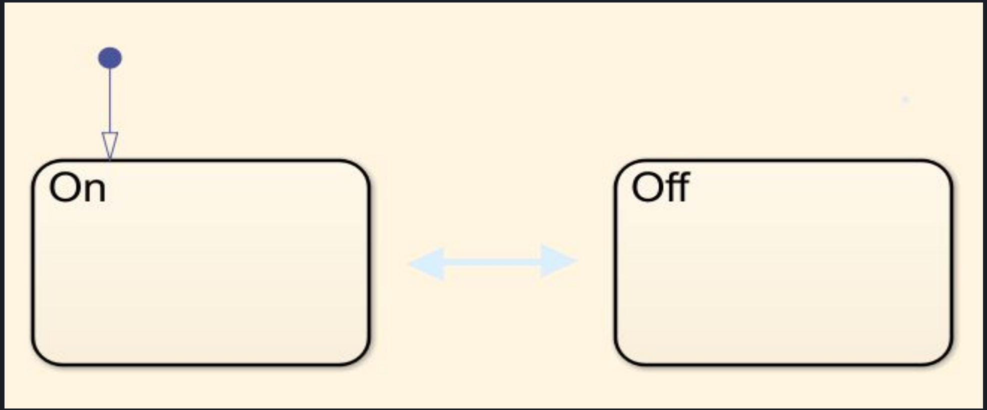


Stateflow Design

Stateflow Editor

- Default Transition
  - Stateflow can recognize certain keywords and intelligence name new states
  - this off state is highlighted orange because there is no way to reach this state.

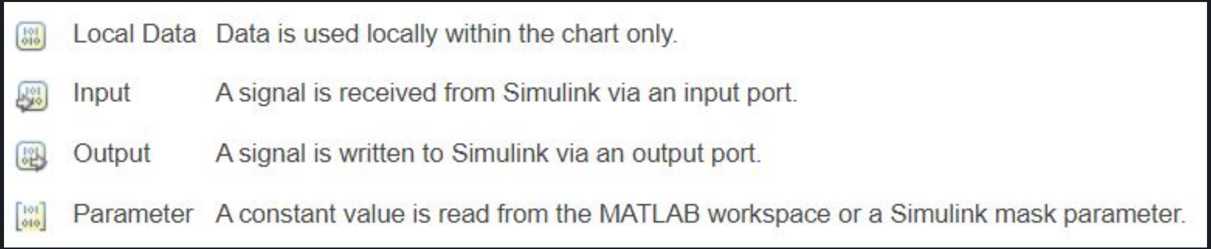




Stateflow Design

Stateflow Editor

- Data Scope In Stateflow
  - In Stateflow, the inputs, outputs and local variables used in a chart are referred to as chart data.
  - all chart data must have a specific scope.
  - common data scopes are listed below
    - Local Data: data is used locally with chart
    - Input: A signal is received from simulink
    - Output: A signal is writing the simulations
    - Parameter: A constant value is read from the matlab workspace or simulink mask parameter .

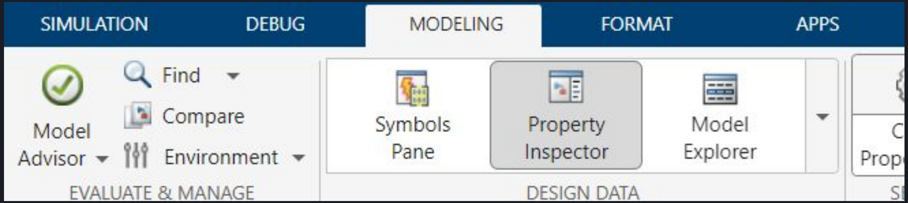


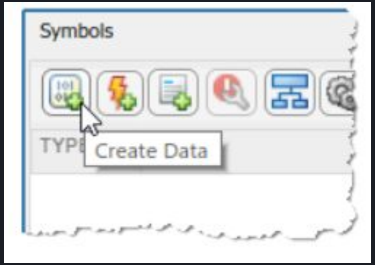
Local Data	Data is used locally within the chart only.
Input	A signal is received from Simulink via an input port.
Output	A signal is written to Simulink via an output port.
Parameter	A constant value is read from the MATLAB workspace or a Simulink mask parameter.

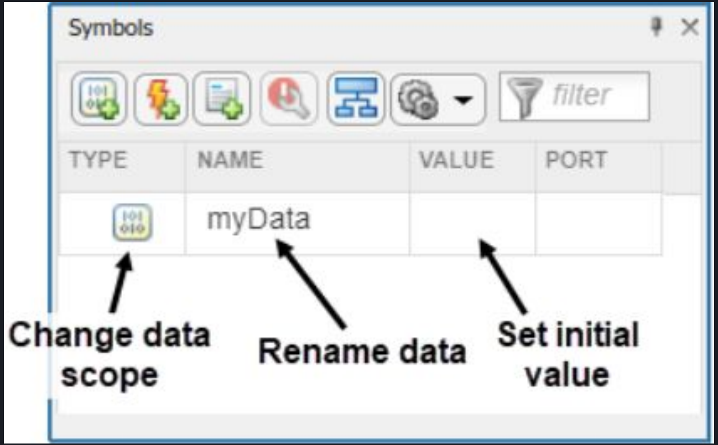
Stateflow Design

Stateflow Editor

- Symbole pane
  - *The Symbols Pane allows you to quickly add and remove Stateflow Data and assign the Data scope using the graphical interface.*
  - *In the Symbols Pane, you can also rename data throughout a chart, change scope, or set a default value.*



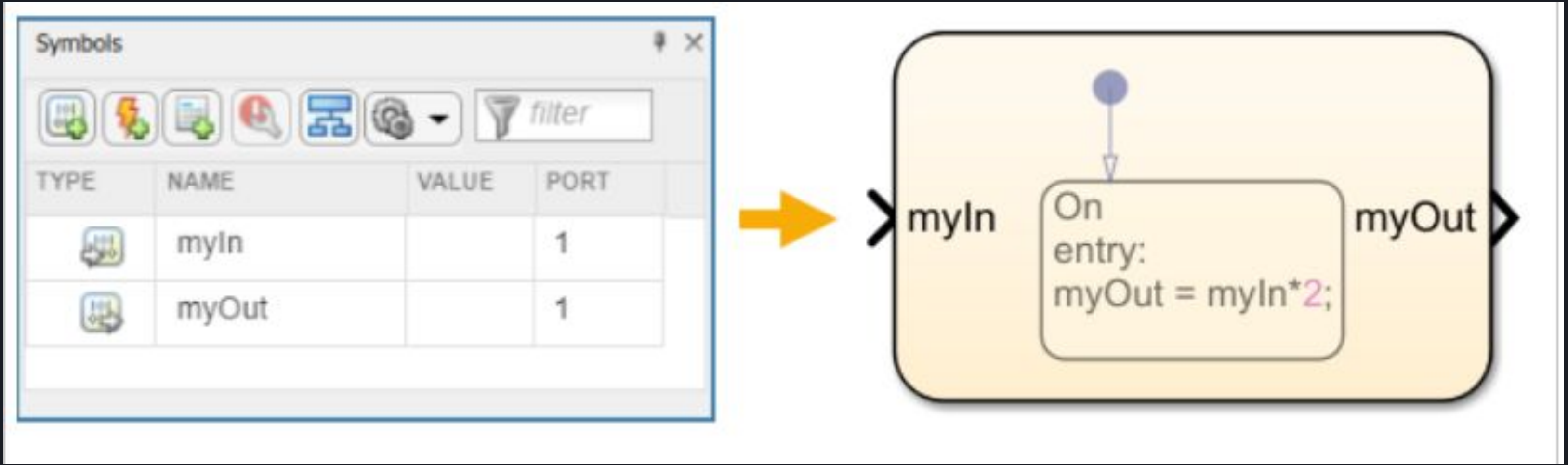




Stateflow Design

Stateflow Editor

- Symbole pane
  - *Data having either input or output scope will create ports on the Stateflow block.*



Symbols

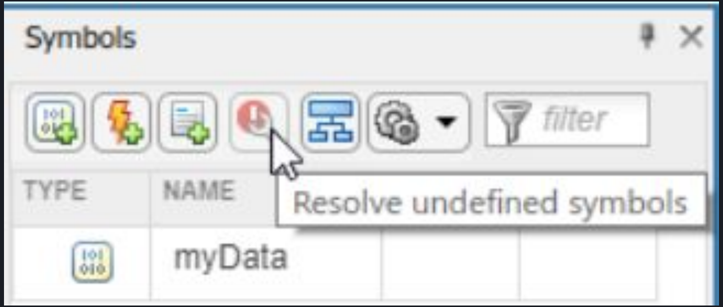
TYPE	NAME	VALUE	PORT
	myIn		1
	myOut		1

```
On
entry:
myOut = myIn*2;
```

Stateflow Design

Stateflow Editor

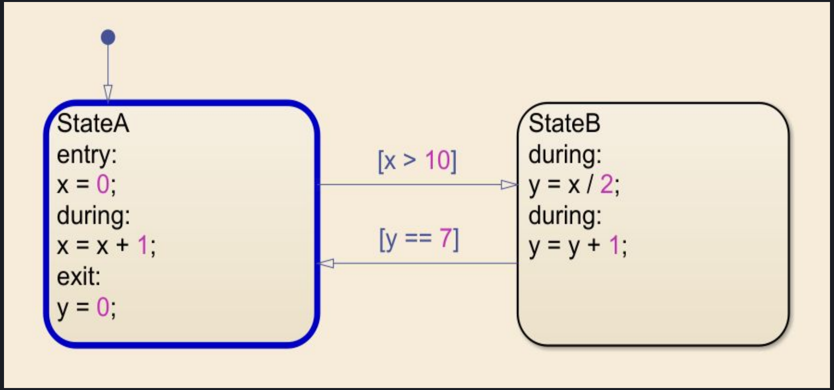
- Symbole pane
  - *If you create a chart and add data before defining it, Stateflow will predict what the scope should be. To use the default suggestion, click the* ***Resolve undefined symbols*** *button .*



Stateflow Design

Stateflow Editor

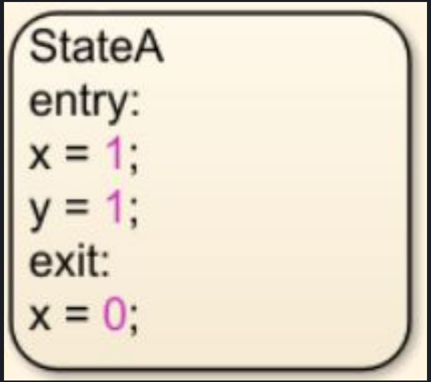
- State Actions
  - entry: allows you include code that is executed when a state is entered. this is one example of a state action. state actions are always specified in this format



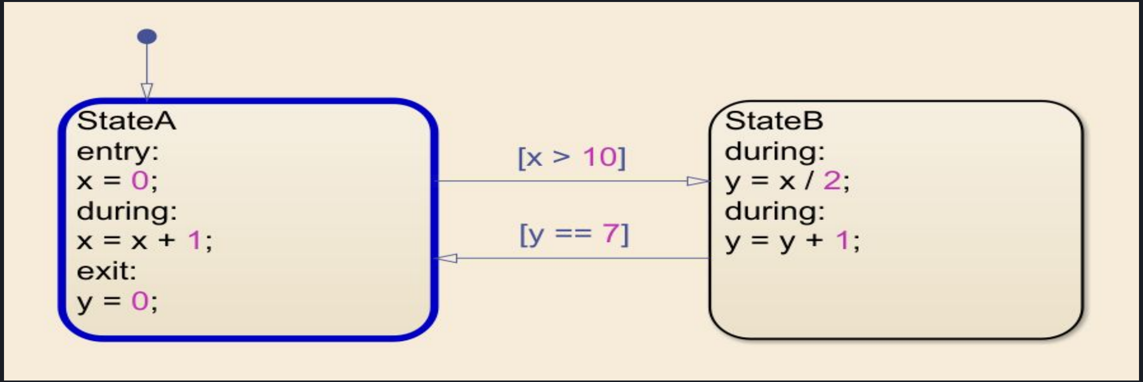
Stateflow Design

Stateflow Editor

- State Actions
  - the three most common states actions are entry, during and exit. these occur when they sound like they would, when entering a state, when remaining in a state , and when leaving a state.
  - All lines of code below one action keyword before another action keyword will be executed.
  - in this example x and y are set to 1 when entering the state when exiting the state x is set to 0.



```
StateA
entry:
x = 1;
y = 1;
exit:
x = 0;
```



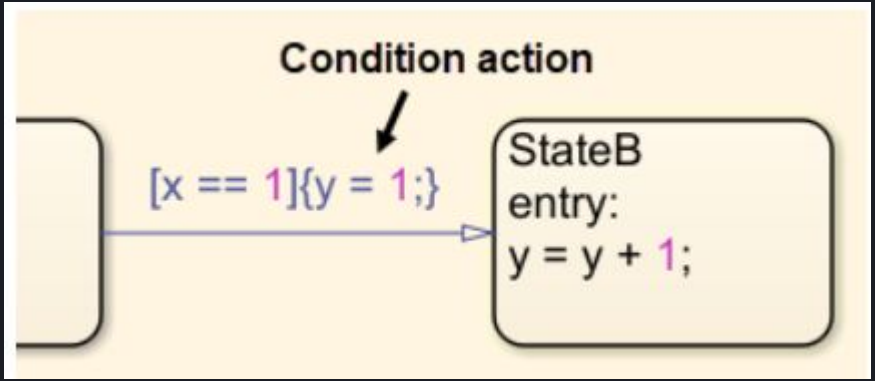
```
StateA
entry:
x = 0;
during:
x = x + 1;
exit:
y = 0;
```

```
StateB
during:
y = x / 2;
during:
y = y + 1;
```

Stateflow Design

Stateflow Editor

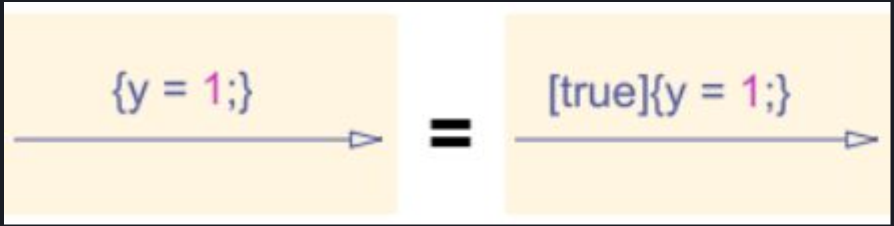
- Conditions Actions
  - stateflow can also perform actions associated with transitions conditions.
  - the condition action is executed when the preceding transition condition is true.
  - condition actions are denoted by curly brackets { } to distinguish them from the condition in square brackets [ ]
  - in this example, y is assigned a value 1 as soon as x == 1 return true. this occurs before the state entry action for state B



Stateflow Design

Stateflow Editor

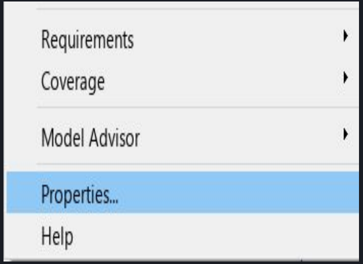
- Conditions Actions
  - if you do not have a transition condition, the transition is considered to always be valid, and the condition action will execute whenever tested.



Stateflow Design

Stateflow Editor

- Action Language [MATLAB,C]



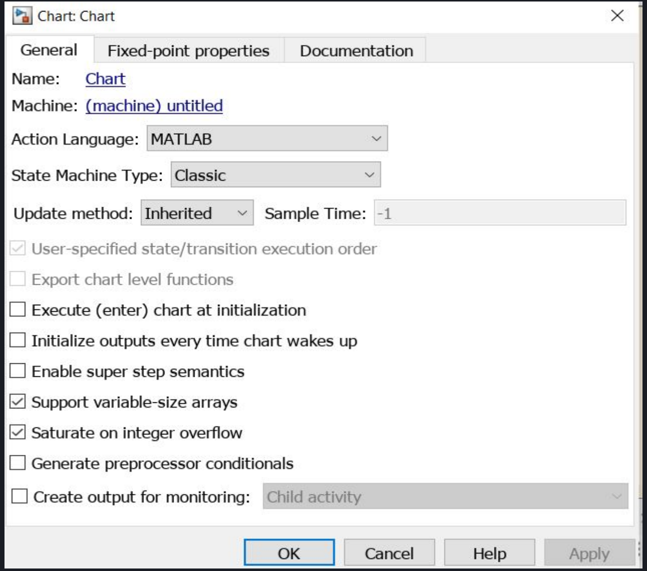


Chart: Chart

General | Fixed-point properties | Documentation

Name: Chart
Machine: (machine) untitled
Action Language: MATLAB
State Machine Type: Classic
Update method: Inherited Sample Time: -1

- [✓] User-specified state/transition execution order
- [ ] Export chart level functions
- [ ] Execute (enter) chart at initialization
- [ ] Initialize outputs every time chart wakes up
- [ ] Enable super step semantics
- [✓] Support variable-size arrays
- [✓] Saturate on integer overflow
- [ ] Generate preprocessor conditionals
- [ ] Create output for monitoring: Child activity

OK | Cancel | Help | Apply

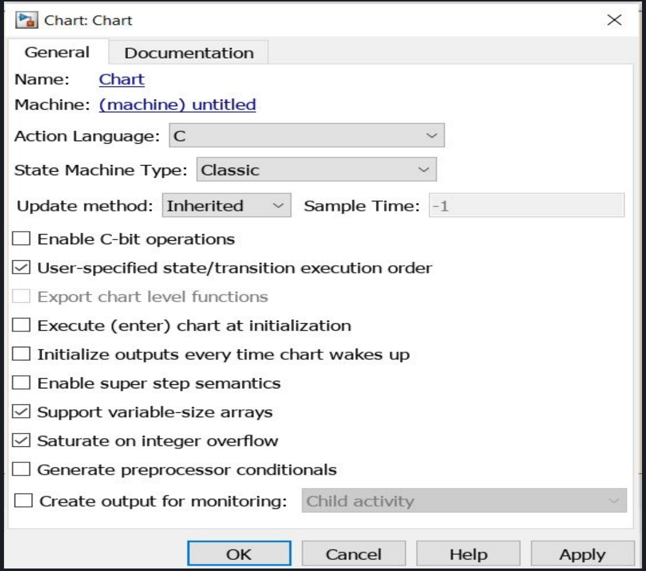


Chart: Chart

General | Documentation

Name: Chart
Machine: (machine) untitled
Action Language: C
State Machine Type: Classic
Update method: Inherited Sample Time: -1

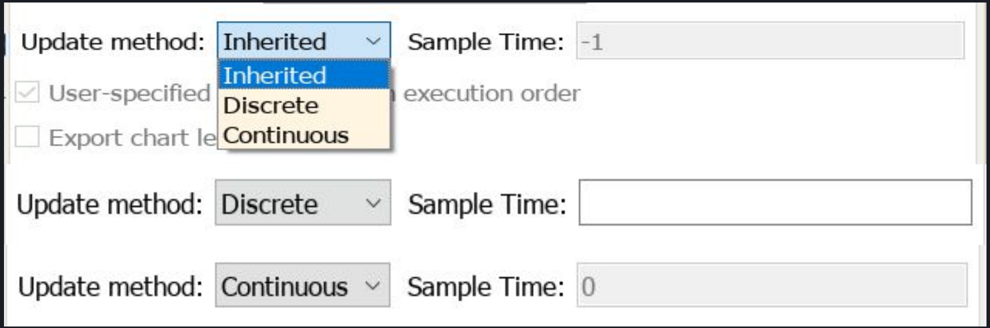
- [ ] Enable C-bit operations
- [✓] User-specified state/transition execution order
- [ ] Export chart level functions
- [ ] Execute (enter) chart at initialization
- [ ] Initialize outputs every time chart wakes up
- [ ] Enable super step semantics
- [✓] Support variable-size arrays
- [✓] Saturate on integer overflow
- [ ] Generate preprocessor conditionals
- [ ] Create output for monitoring: Child activity

OK | Cancel | Help | Apply

Stateflow Design

Stateflow Editor

- Chart execution in Simulink
  - Sample Time of SImulink
  - Sample Time of chart
  - Stateflow chart in simulink execute a given through a simulation accruing to a given sample time.
  - This sample time can be set specifically for the chart or can be inherited from simulink, the stateflow chart excures once per time step



Update method: Inherited Sample Time: -1

User-specified execution order

Export chart le

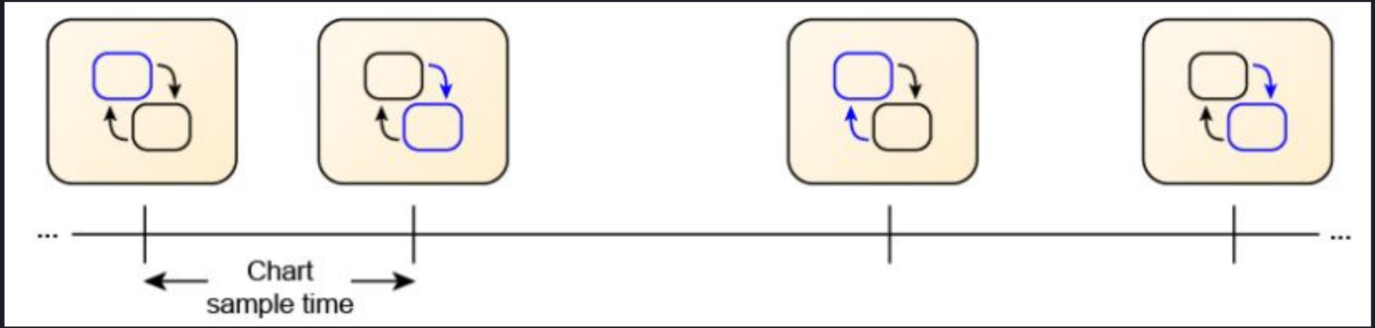
Update method: Discrete Sample Time:

Update method: Continuous Sample Time: 0

Stateflow Design

Stateflow Editor

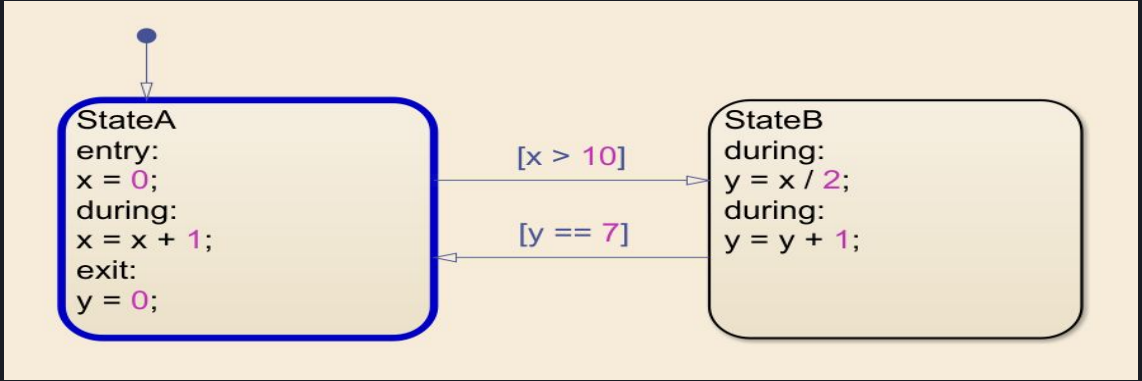
- Chart execution in Simulink
  - Sample Time of SImulink
  - Sample Time of chart
  - Stateflow chart in simulink execute a given through a simulation accruing to a given sample time.
  - This sample time can be set specifically for the chart or can be inherited from simulink, the stateflow chart excures once per time step



Stateflow Design

Stateflow Editor

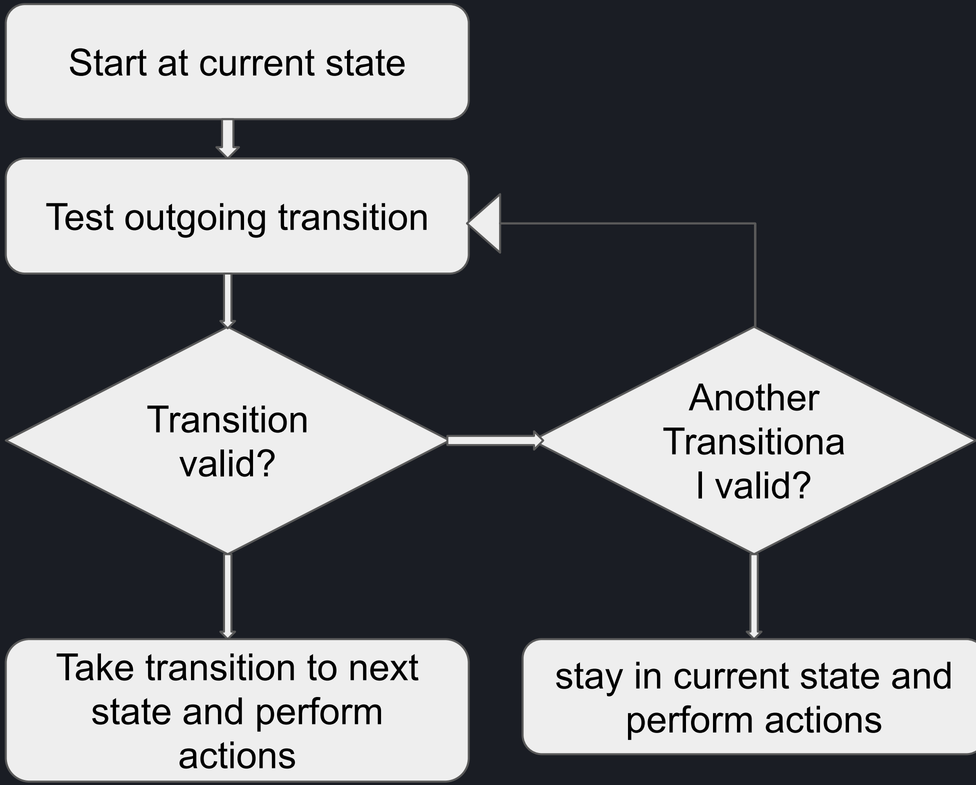
- Chart execution in Simulink
  - chart execution results in one of two outcomes
    - A valid transition is taken and a new state becomes active.
    - no valid transition excites, and the current state remains active



Stateflow Design

Stateflow Editor

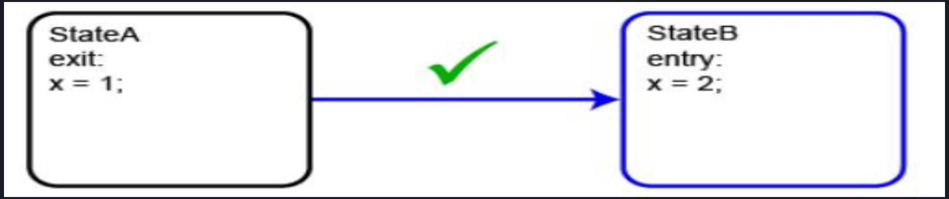
- Chart execution in Simulink
  - When testing transition all transitions leaving the active state are tested according to their execution order . if a valid transition is found no, more transitions are tested.
  - When testing transition all transitions leaving the active state are tested according to their execution order . if a valid transition is found no, more transitions are tested.

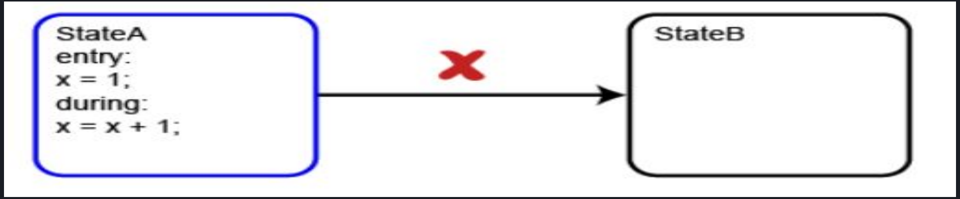

Start at current state
Test outgoing transition
Transition valid?
Another Transitiona l valid?
Take transition to next state and perform actions
stay in current state and perform actions

Stateflow Design

Stateflow Editor

- Chart execution in Simulink
  - state entry and exit actions occur sequentially when leaving and entering a state. the during action occurs for each consecutive time step that a state remains active.
  - in the first case, when there is avlied transitions x is first set to 1 when exiting state A

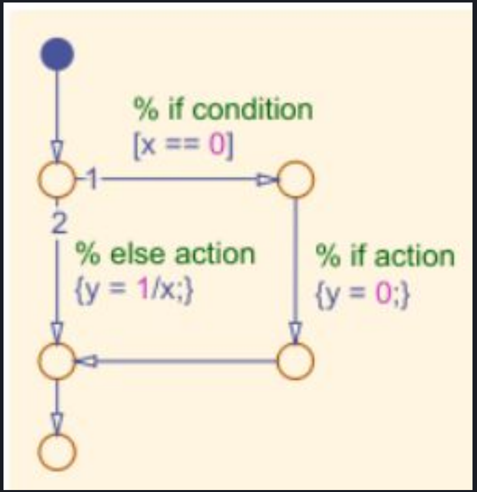




Stateflow Design

Stateflow Editor

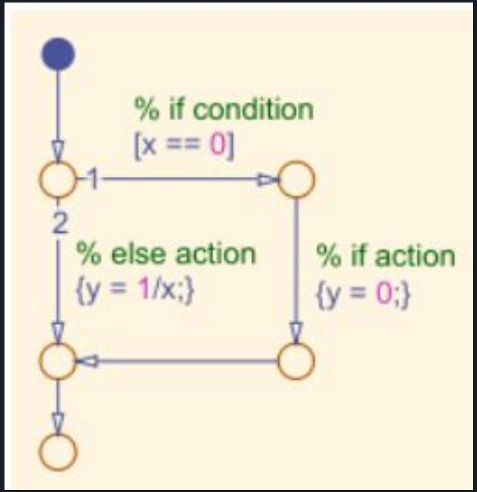
- Flow Charts
  - Stateflow support the construction of flow chart stateless models of logic patterns such as decision trees and iterative loops.
  - Flow charts can visually represent complex algorithms in a format that's easy to read and state flow charts can be used to describe advanced rules for transiting between states.



Stateflow Design

Stateflow Editor

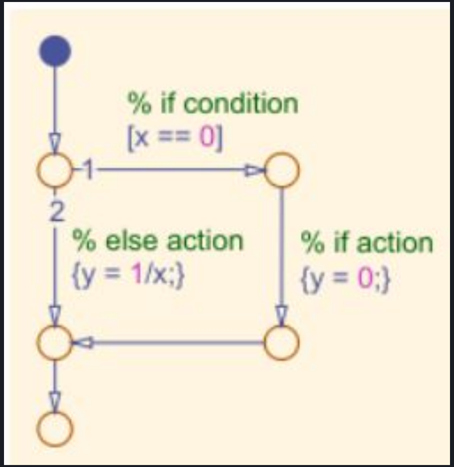
- Flow Charts
    - flow charts are build using two graphical elements available in state flow : transition and junctions .
    - Transition: conditional pathway represented by arrows.
    - junction: are nodes that divide the pathway into segment represents by circuiral
    - They can provide addition branches along a decision path way. for example, transitions and junctions can be used to model as if-else statement



Stateflow Design

Stateflow Editor

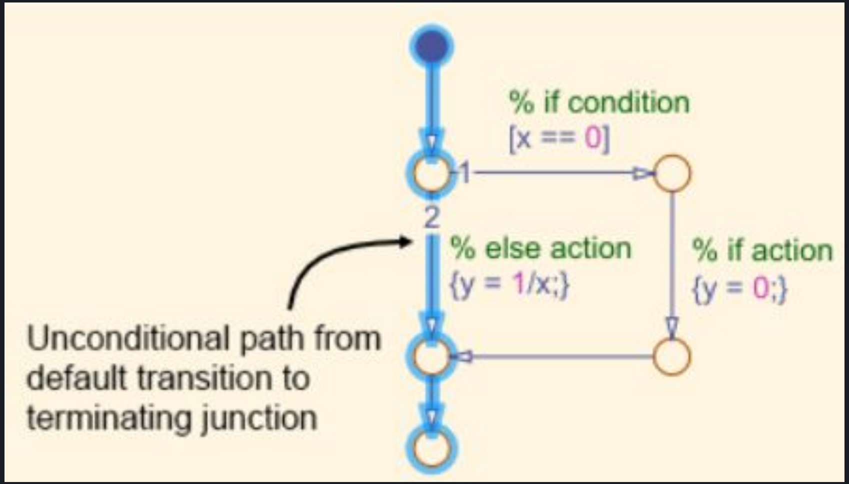
- Flow Charts
    - when a stateflow chart consists of only a flow chart, the flow chart needs to
        - Have a single, default transition
        - coverage at a single, terminating junction



Stateflow Design

Stateflow Editor

- Flow Charts
  - there should also be an unconditional transition from every junction that represents a decision point. in other words, provide an else statement for any if or if-else if decision tree, or a default case for switch statement


% if condition
[x == 0]
1
2
% else action
{y = 1/x;}
% if action
{y = 0;}
Unconditional path from default transition to terminating junction

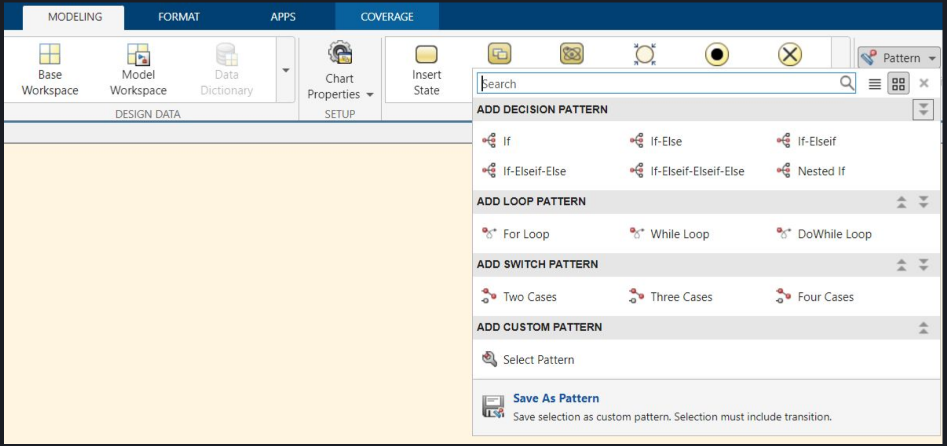
Stateflow Design

Stateflow Editor

- Flow Charts

Stateflow provides a Pattern Wizard to simplify creating common flow charts. These patterns include

1. Decision: `if`, `if-else`, and nested `if` decision patterns.
2. Loop: `for`, `while`, and `do-while` loop patterns.
3. Switch: `switch` patterns with up to four cases.
4. Custom: custom patterns that you saved for later reuse.



Save selection as custom pattern. Selection must include transition.

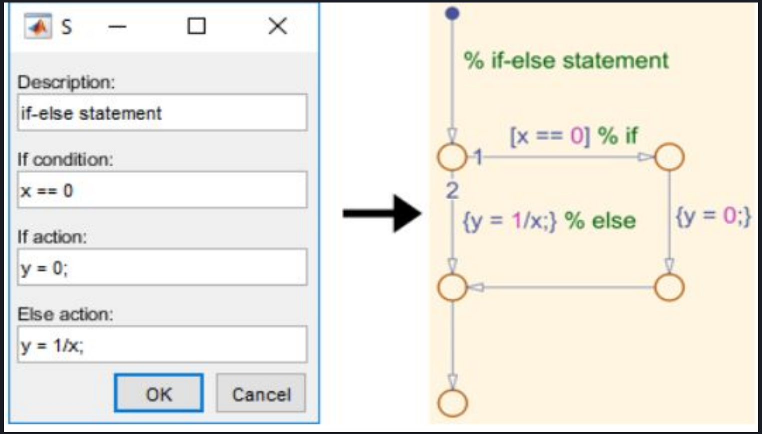
Stateflow Design

Stateflow Editor

- Flow Charts

Stateflow provides a Pattern Wizard to simplify creating common flow charts. These patterns include

1. Decision: `if`, `if-else`, and nested `if` decision patterns.
2. Loop: `for`, `while`, and `do-while` loop patterns.
3. Switch: `switch` patterns with up to four cases.
4. Custom: custom patterns that you saved for later reuse.



Description: if-else statement

If condition: x == 0

If action: y = 0;

Else action: y = 1/x;

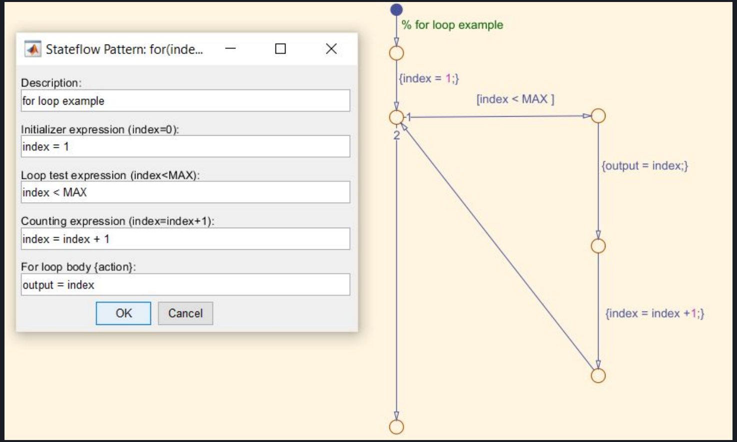
Stateflow Design

Stateflow Editor

- Flow Charts

Stateflow provides a Pattern Wizard to simplify creating common flow charts. These patterns include

1. Decision: `if`, `if-else`, and nested `if` decision patterns.
2. Loop: `for`, `while`, and `do-while` loop patterns.
3. Switch: `switch` patterns with up to four cases.
4. Custom: custom patterns that you saved for later reuse.


Stateflow Pattern: for(inde...

Description:
for loop example

Initializer expression (index=0):
index = 1

Loop test expression (index<MAX):
index < MAX

Counting expression (index=index+1):
index = index + 1

For loop body {action}:
output = index

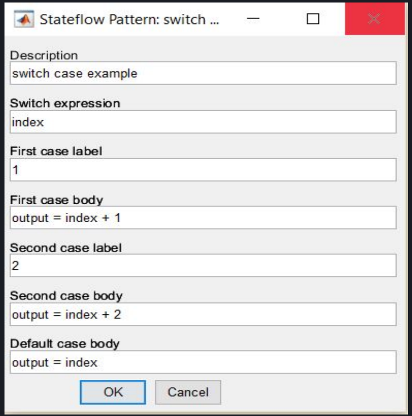
Stateflow Design

Stateflow Editor

- Flow Charts

Stateflow provides a Pattern Wizard to simplify creating common flow charts. These patterns include

1. Decision: `if`, `if-else`, and nested `if` decision patterns.
2. Loop: `for`, `while`, and `do-while` loop patterns.
3. Switch: `switch` patterns with up to four cases.
4. Custom: custom patterns that you saved for later reuse.



Description: switch case example

Switch expression: index

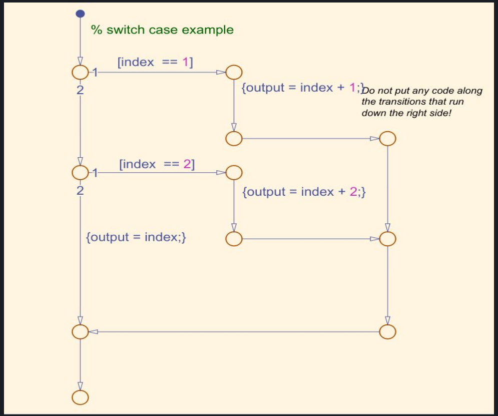
First case label: 1

First case body: output = index + 1

Second case label: 2

Second case body: output = index + 2

Default case body: output = index

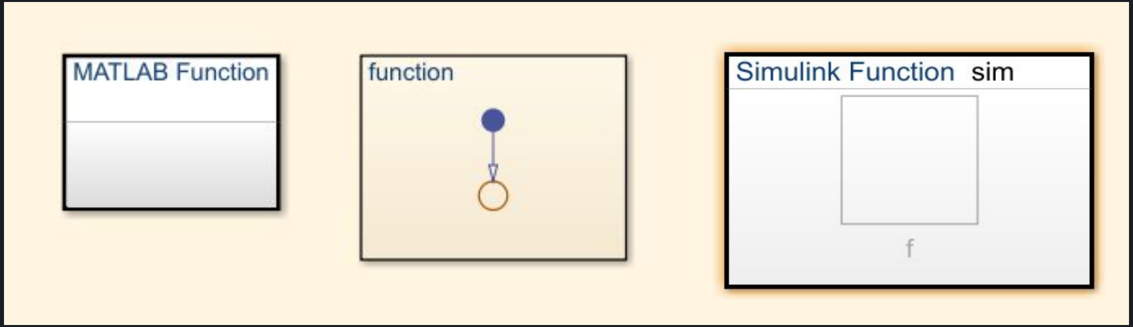


Model-Based Development Program

Stateflow Design

Stateflow Editor

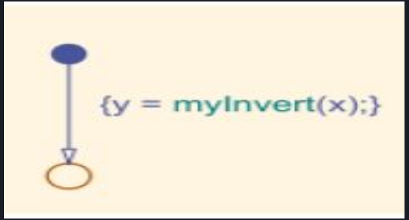
- Functions
  - Stateflow facilitates the development of diverse functions that can be invoked repeatedly within a chart. Similar to traditional text-based programming, utilizing functions minimizes repetition and enhances the clarity of the code. Throughout this module, you'll gain insights into two specific types of Stateflow functions: graphical functions and MATLAB functions. Additional information on various function types can be explored in the documentation page titled "Reusable Components in Charts." These functions can be invoked as either state actions or condition actions. If a function yields a Boolean value, it can also serve as a transition condition.

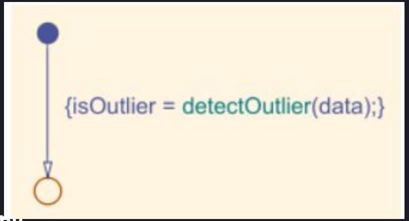


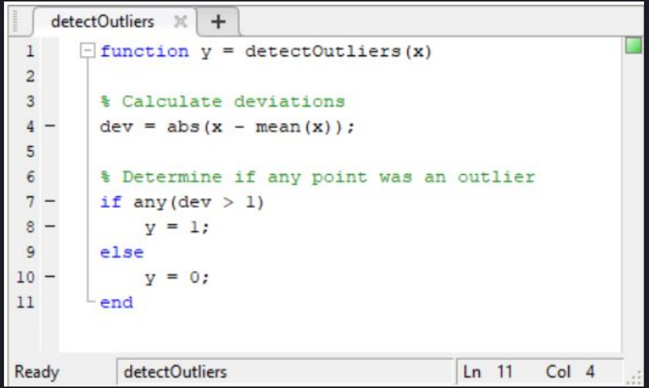
Stateflow Design

Stateflow Editor

- Functions
  - *When first added to a model, the graphical function box will contain a blank function signature, a default transition, and a terminating junction.*
  - *The function signature for graphical functions is the same as the textual MATLAB function signature. After defining your function, add the graphical elements that define the function's behavior.*
  - *To use a function, call it via a state or condition action.*



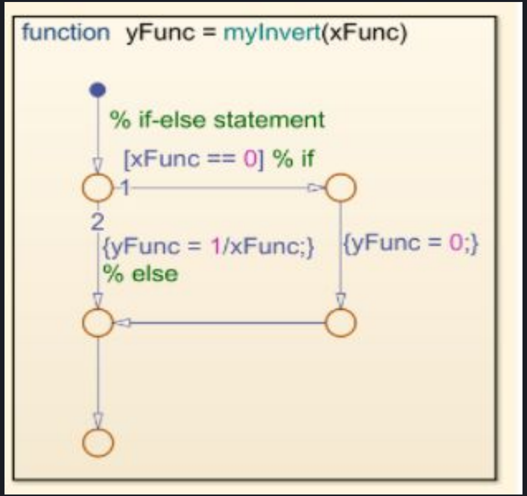




```
function y = detectOutliers(x)

% Calculate deviations
dev = abs(x - mean(x));

% Determine if any point was an outlier
if any(dev > 1)
    y = 1;
else
    y = 0;
end
```

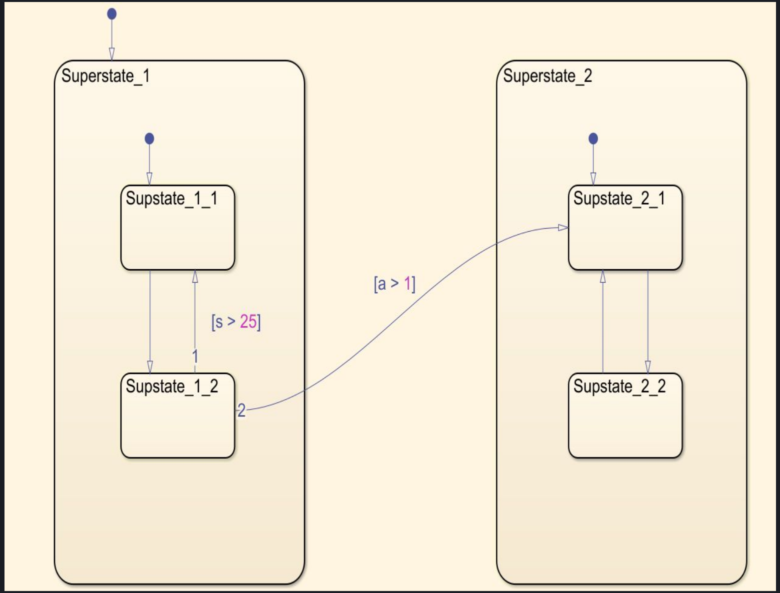


Model-Based Development Program

Stateflow Design

Stateflow Editor

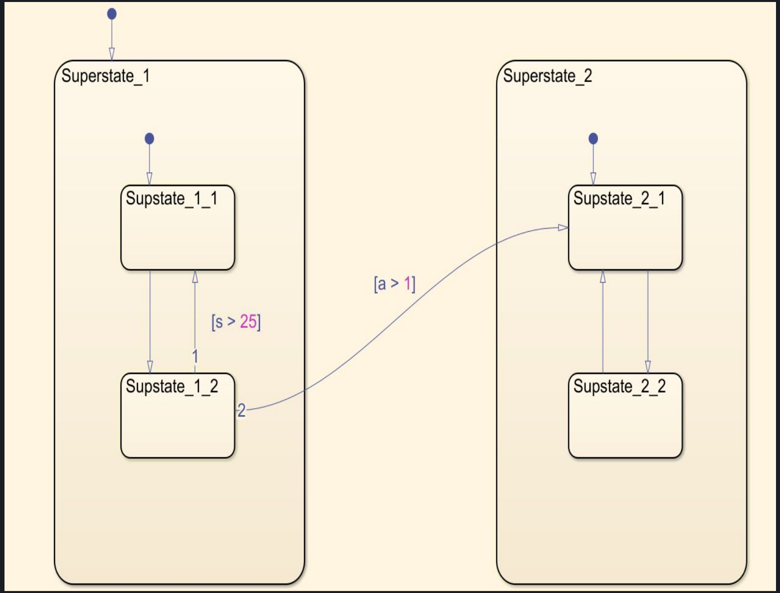
- Stats Hierarchy
  - States Hierarchy: The previous state diagram had 3 states of the same level without hierarchy.
  - Stateflow uses the "superstate" concept for modeling complex logic behavior.
  - Superstate (Outer State): A parent state containing other states.
  - Substates (Child State): Child states contained within an outer state.
  - Substates can only be active if the outer state is active.
  - Using superstates and substates improves model readability.
  - There is no limit to the number of hierarchy levels that can be constructed in a diagram.



Stateflow Design

Stateflow Editor

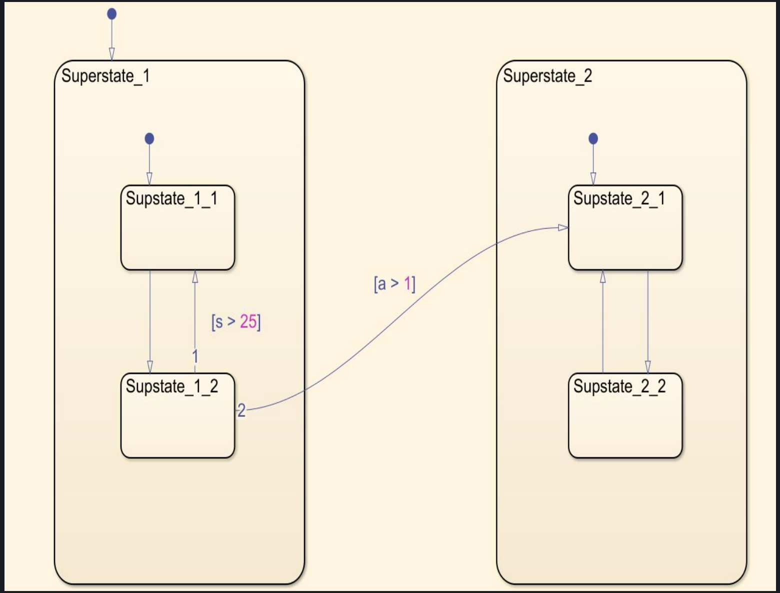
- Stats Hierarchy
  - More Complex Architectures: Complex architectures require specific change rules and test priorities for transitions.
  - Types of Transitions:
    - Internal Transitions: Transitions that do not cross the edge of the state and are contained within the state.
    - External Transitions: Transitions that cross the border of the state.
  - Test Rules for Transitions:
    - Superstate Transitions are tested before substate transitions.
    - External Transitions are tested before internal transitions.



Stateflow Design

Stateflow Editor

- States Hierarchy
  - More Complex Architectures: Complex architectures require specific change rules and test priorities for transitions.
  - Types of Transitions:
    - Internal Transitions: Transitions that do not cross the edge of the state and are contained within the state.
    - External Transitions: Transitions that cross the border of the state.
  - Test Rules for Transitions:
    - Superstate Transitions are tested before substate transitions.
    - External Transitions are tested before internal transitions.



Stateflow Design

What is a finite state machine?

- A finite state machine (FSM) or finite state automaton is a method of modeling a system comprising a limited number of modes; depending on which mode it is in, the machine will behave in one way or another.

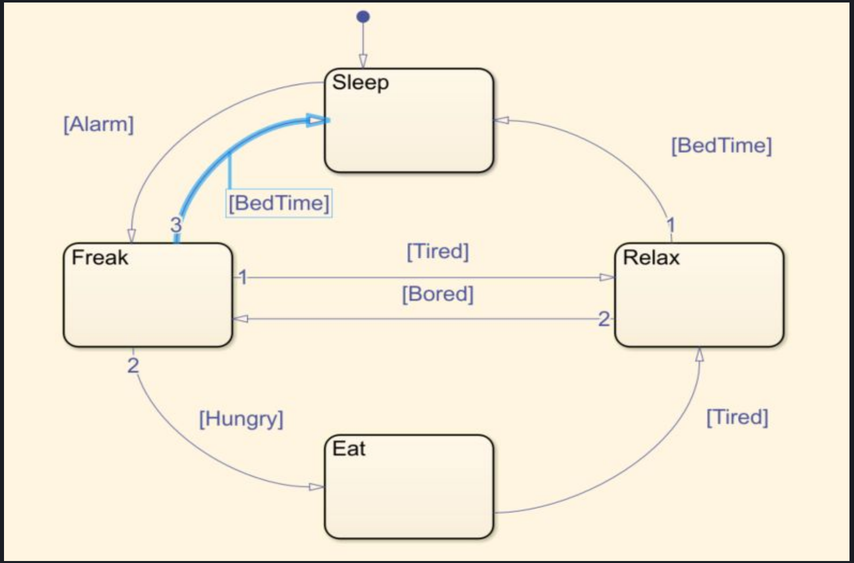
- FSM can be expressed graphically by means of the “state transition diagram”

- The basic building blocks of a FSM are states, transitions and events.

Stateflow Design

What is a finite state machine?

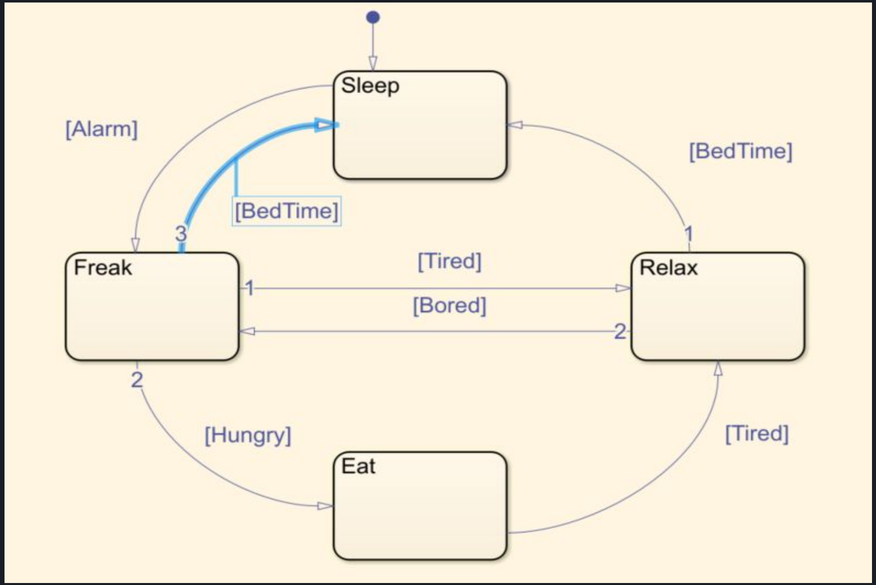
- As an example, the behavior of a house cat is modeled as a FSM using following state transition diagram .



Stateflow Design

What is a finite state machine?

- What if the cat sees a mouse? then we need to adapt our design to let the cat chase the mouse

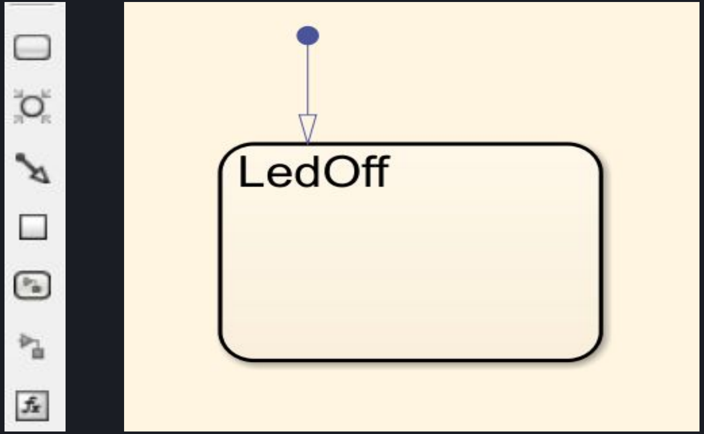


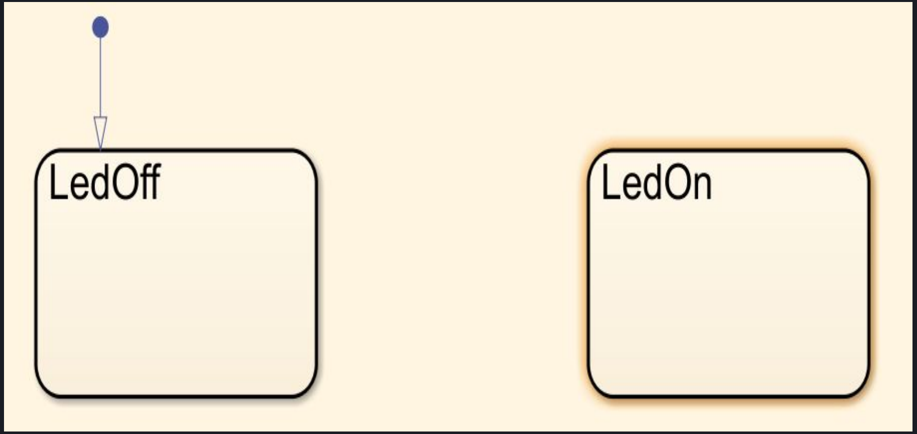
Model-Based Development Program

Stateflow Design

What is a finite state machine?

- Exercise: Blink Led
  - it is required to build state machine to model the blink led every 1 sec application

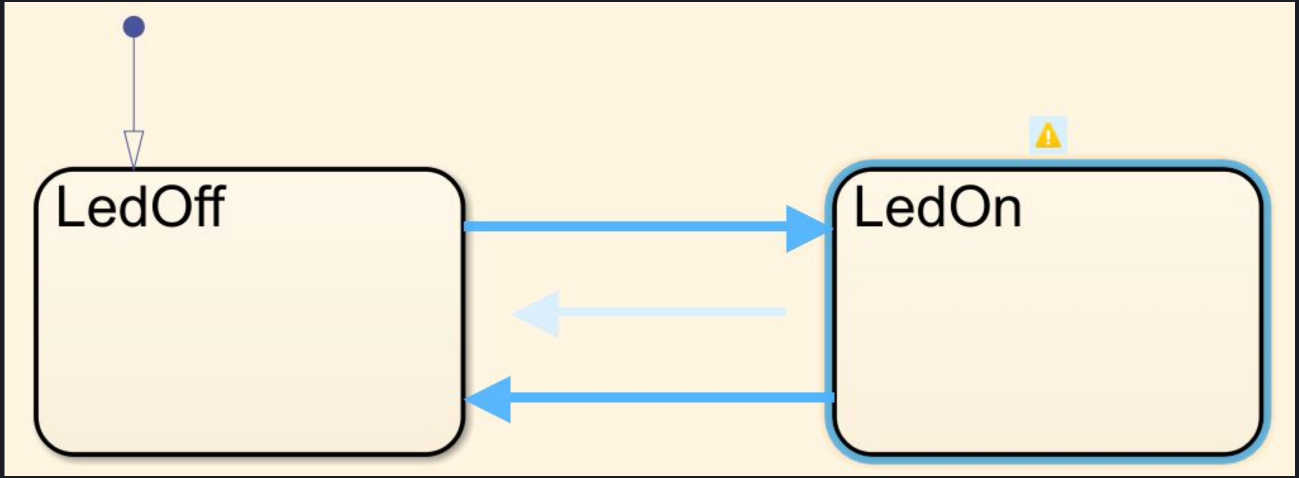




Stateflow Design

What is a finite state machine?

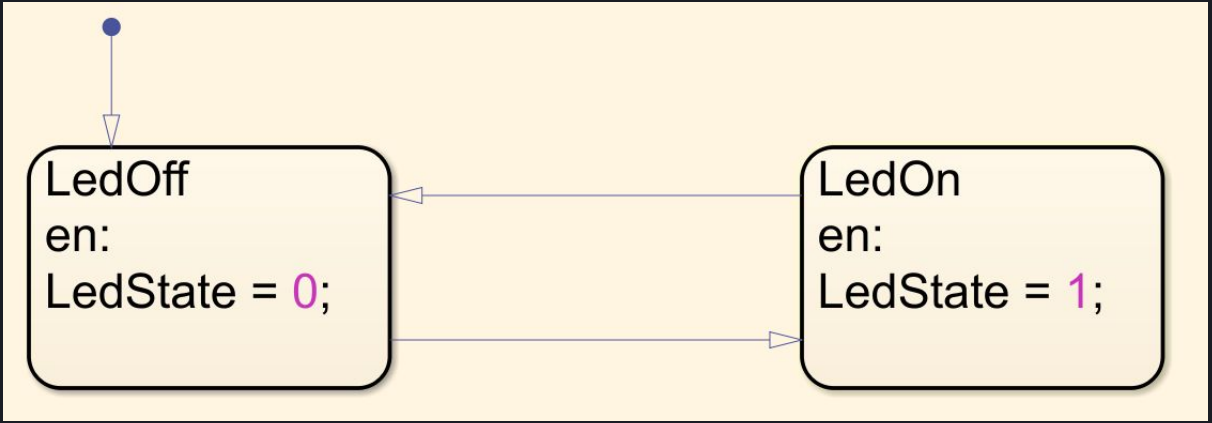
- Exercise: Blink Led
  - it is required to build state machine to model the blink led every 1 sec application



Stateflow Design

What is a finite state machine?

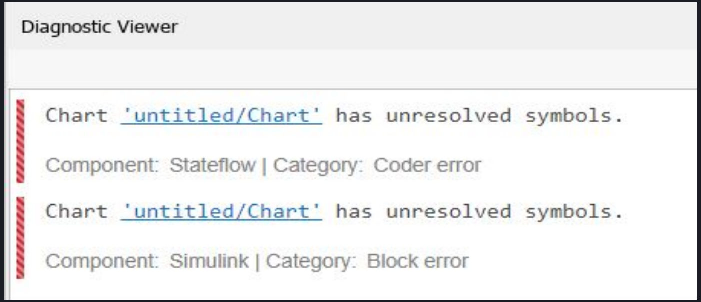
- Exercise: Blink Led
  - it is required to build state machine to model the blink led every 1 sec application



Stateflow Design

What is a finite state machine?

- Exercise: Blink Led
  - it is required to build state machine to model the blink led every 1 sec application



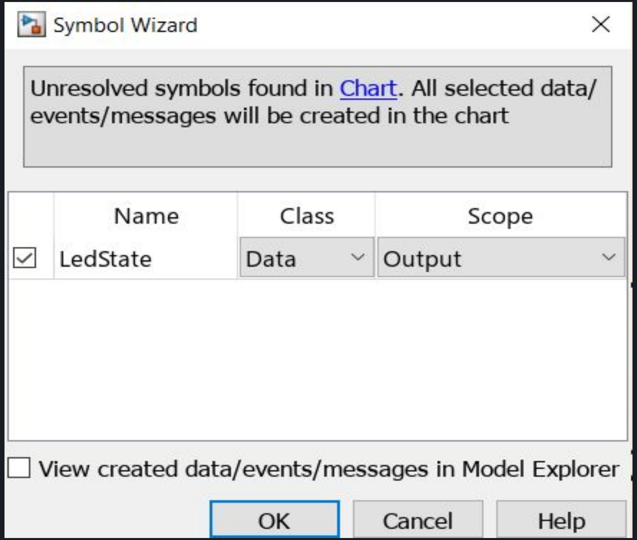
Diagnostic Viewer

```
Chart 'untitled/Chart' has unresolved symbols.
```

Component: Stateflow | Category: Coder error

```
Chart 'untitled/Chart' has unresolved symbols.
```

Component: Simulink | Category: Block error



Symbol Wizard

Unresolved symbols found in Chart. All selected data/events/messages will be created in the chart

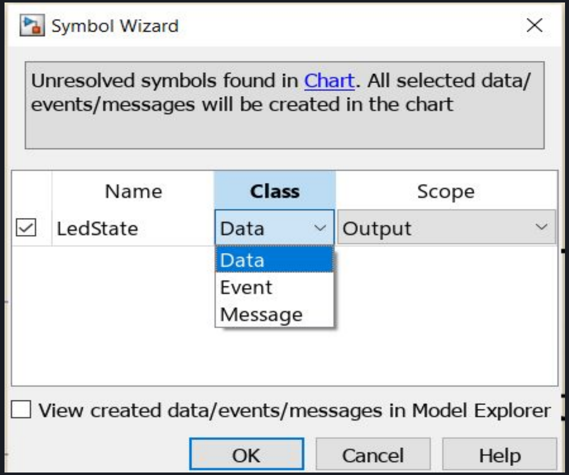
	Name	Class	Scope
☑	LedState	Data	Output

☐ View created data/events/messages in Model Explorer

Stateflow Design

What is a finite state machine?

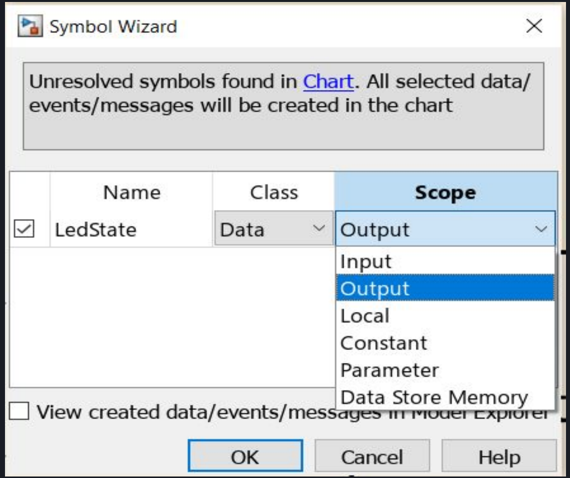
- Exercise: Blink Led
  - it is required to build state machine to model the blink led every 1 sec application



Symbol Wizard

Unresolved symbols found in Chart. All selected data/events/messages will be created in the chart

	Name	Class	Scope
✓	LedState	Data	Output



Symbol Wizard

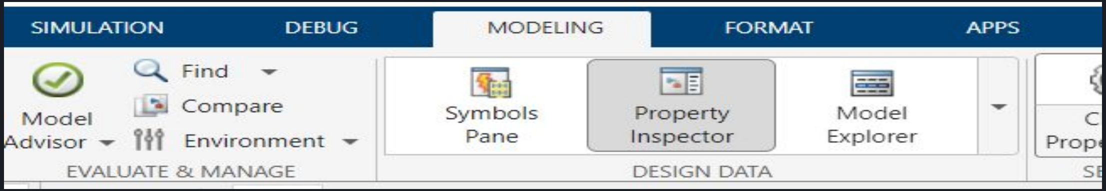
Unresolved symbols found in Chart. All selected data/events/messages will be created in the chart

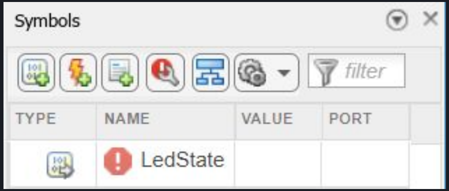
	Name	Class	Scope
✓	LedState	Data	Output

Stateflow Design

What is a finite state machine?

- Exercise: Blink Led
    - it is required to build state machine to model the blink led every 1 sec application



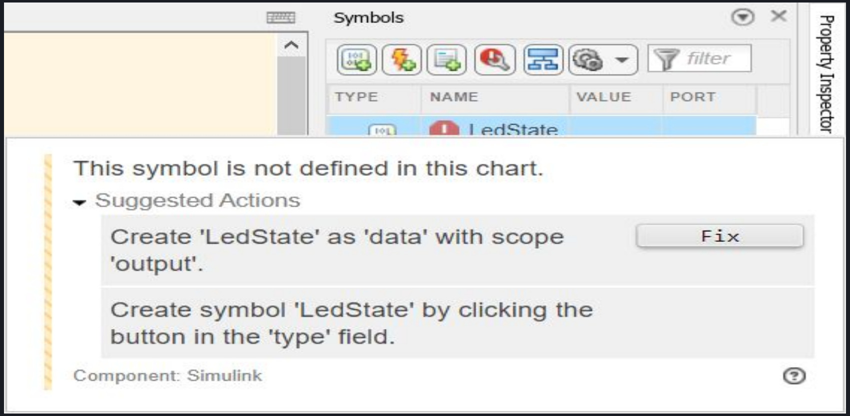


TYPE	NAME	VALUE	PORT
	LedState		

Stateflow Design

What is a finite state machine?

- Exercise: Blink Led
  - it is required to build state machine to model the blink led every 1 sec application



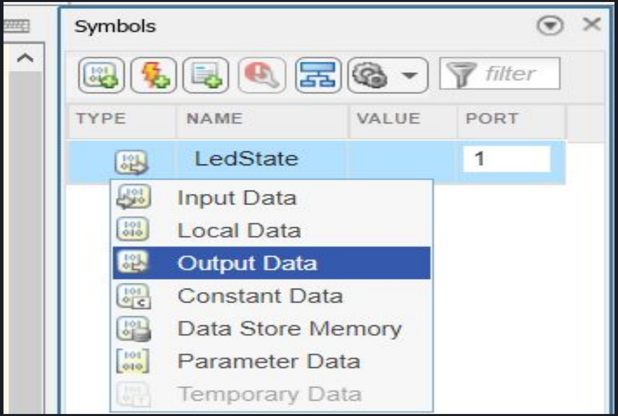
This symbol is not defined in this chart.

Suggested Actions

Create 'LedState' as 'data' with scope 'output'. Fix

Create symbol 'LedState' by clicking the button in the 'type' field.

Component: Simulink



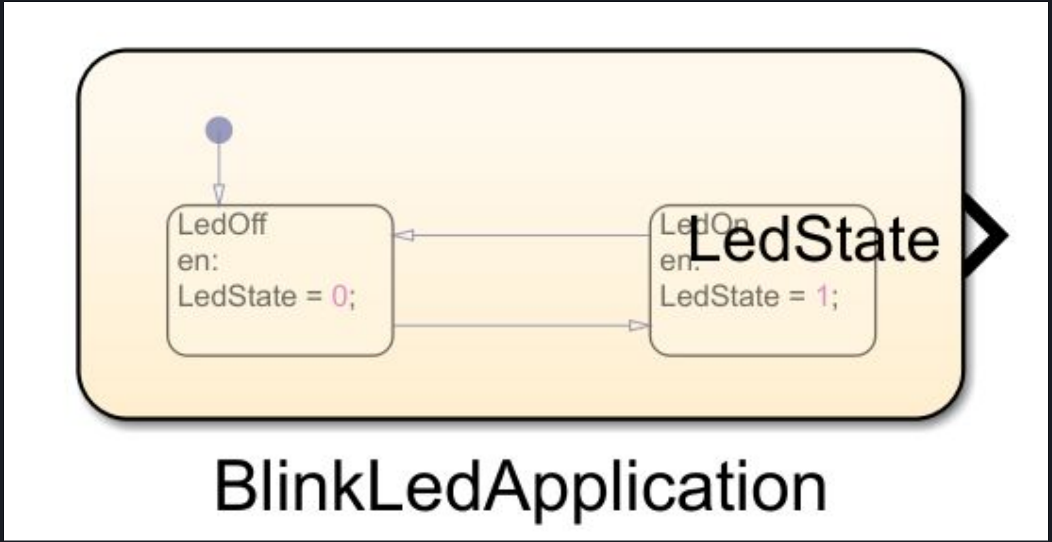
TYPE	NAME	VALUE	PORT
	LedState		1

Model-Based Development Program

Stateflow Design

What is a finite state machine?

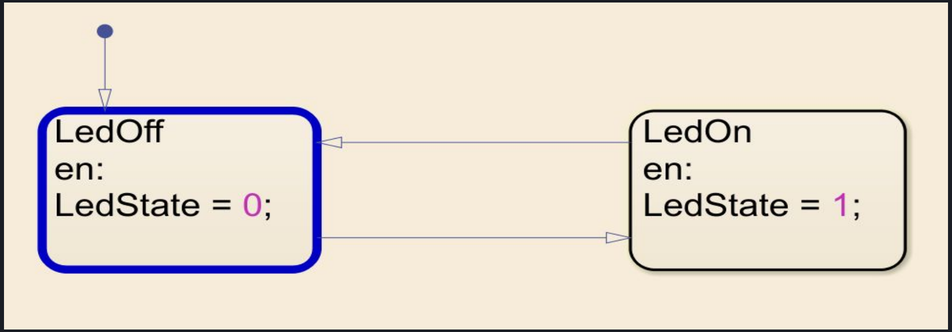
- Exercise: Blink Led
  - it is required to build state machine to model the blink led every 1 sec application

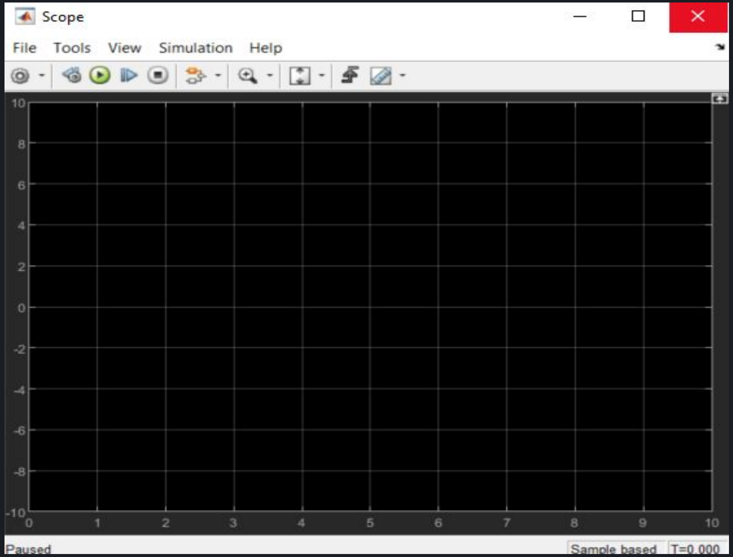


Stateflow Design

What is a finite state machine?

- Exercise: Blink Led
  - it is required to build state machine to model the blink led every 1 sec application

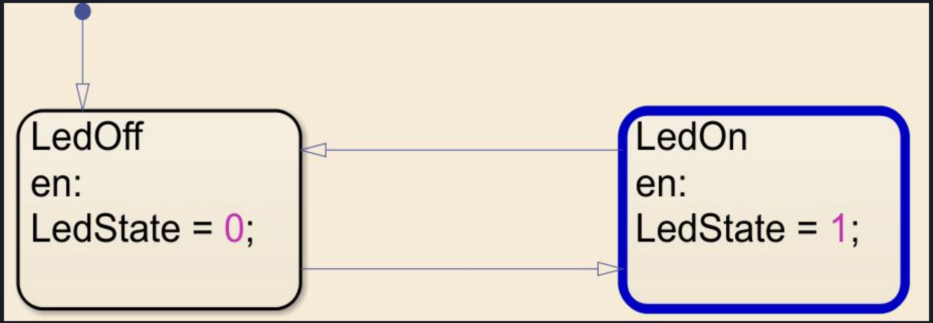


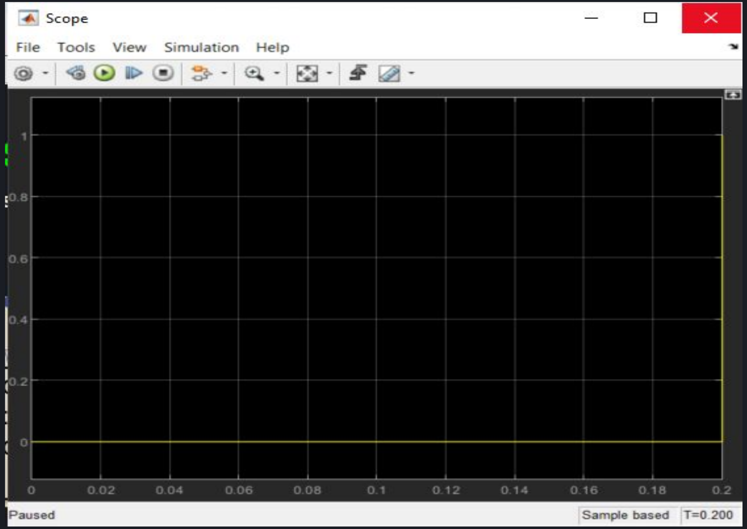


Stateflow Design

What is a finite state machine?

- Exercise: Blink Led
    - it is required to build state machine to model the blink led every 1 sec application

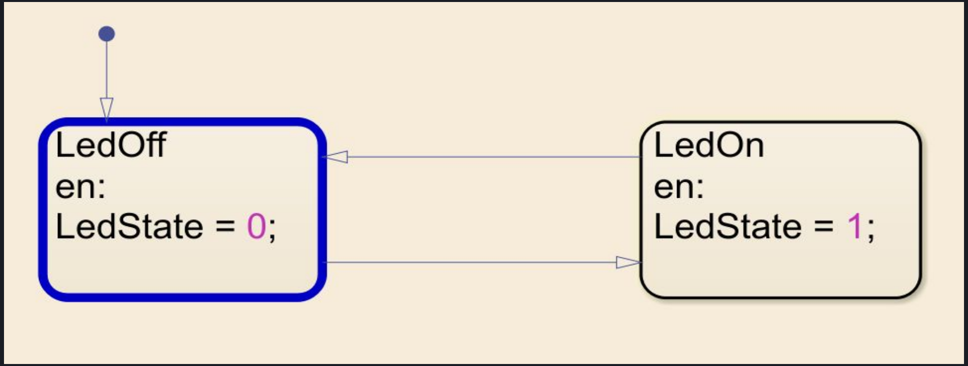


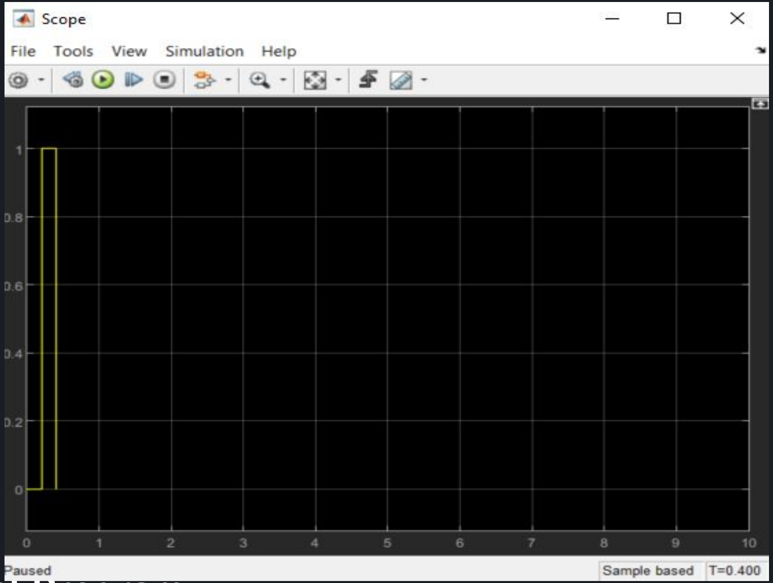


Stateflow Design

What is a finite state machine?

- Exercise: Blink Led
    - it is required to build state machine to model the blink led every 1 sec application

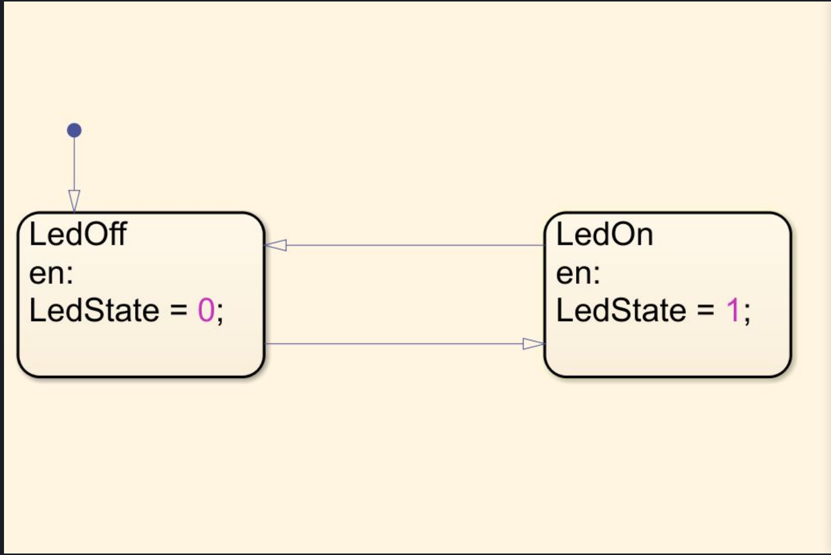


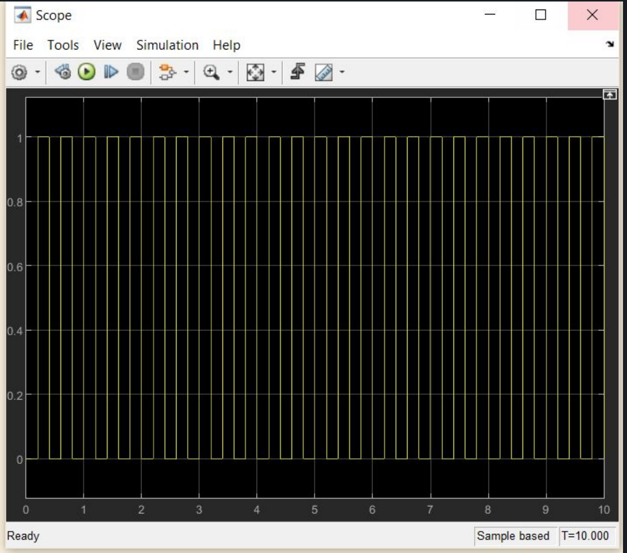


Stateflow Design

What is a finite state machine?

- Exercise: Blink Led
  - it is required to build state machine to model the blink led every 1 sec application

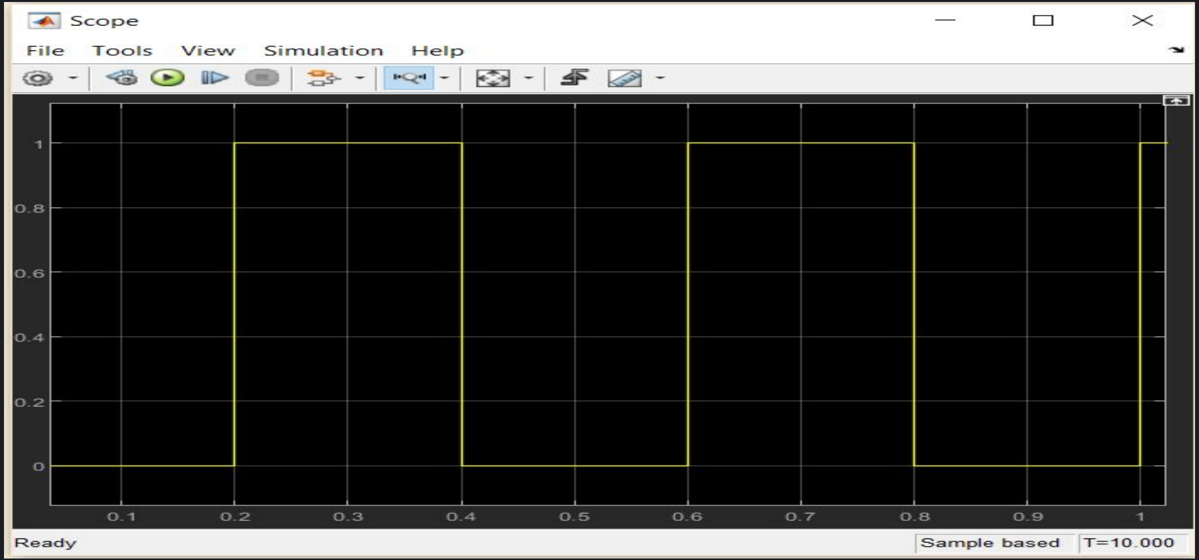




Stateflow Design

What is a finite state machine?

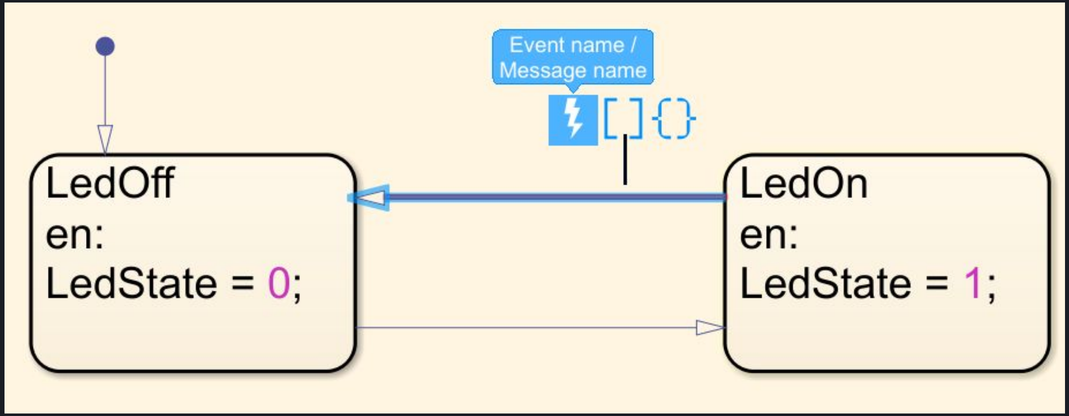
- Exercise: Blink Led
  - it is required to build state machine to model the blink led every 1 sec application

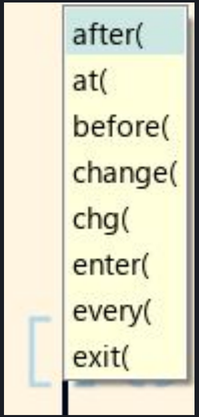


Stateflow Design

What is a finite state machine?

- Exercise: Blink Led
  - it is required to build state machine to model the blink led every 1 sec application

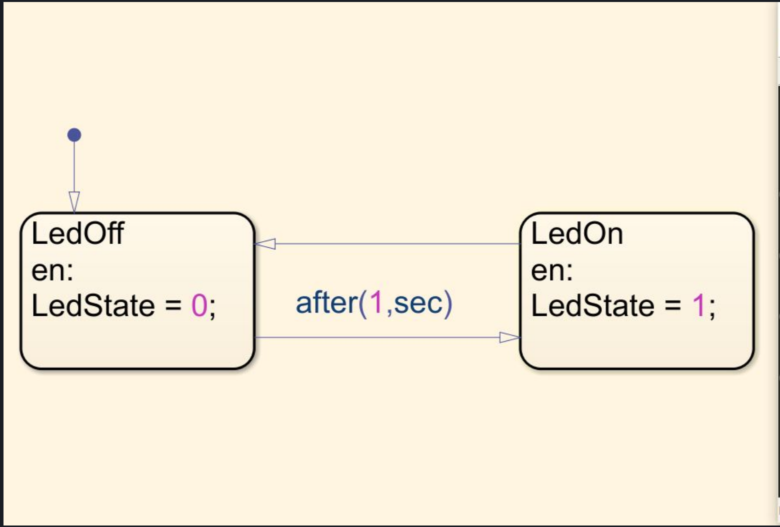


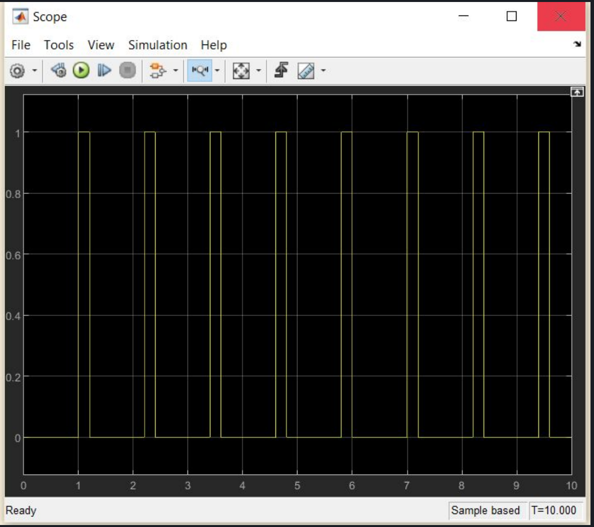


Stateflow Design

What is a finite state machine?

- Exercise: Blink Led
    - it is required to build state machine to model the blink led every 1 sec application

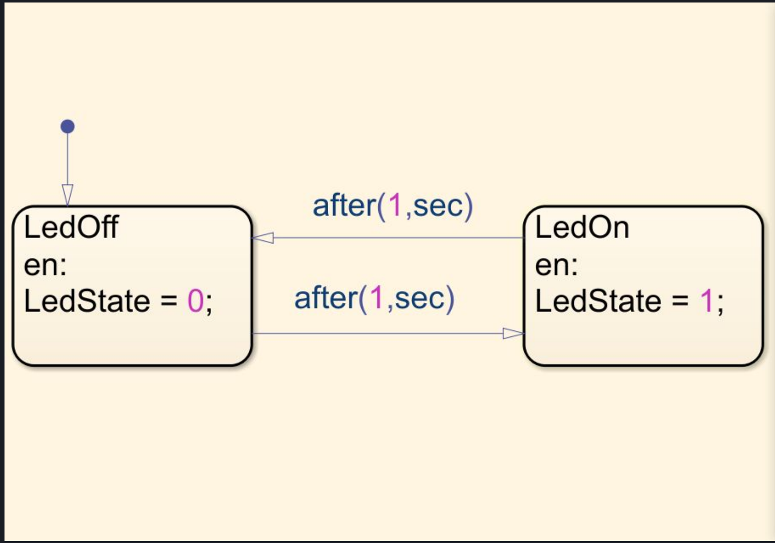


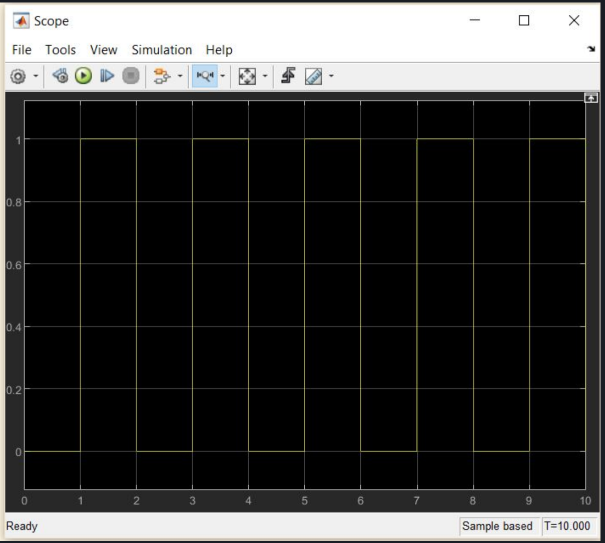


Stateflow Design

What is a finite state machine?

- Exercise: Blink Led
    - it is required to build state machine to model the blink led every 1 sec application

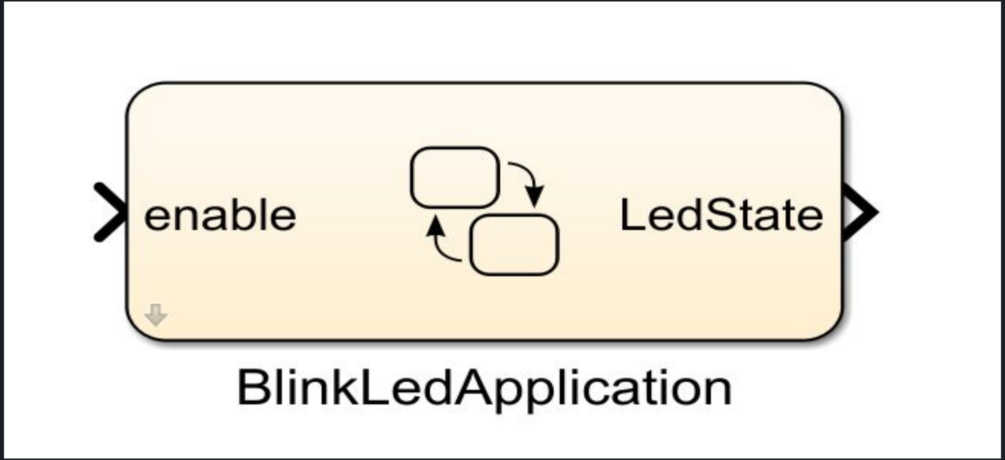




Stateflow Design

What is a finite state machine?

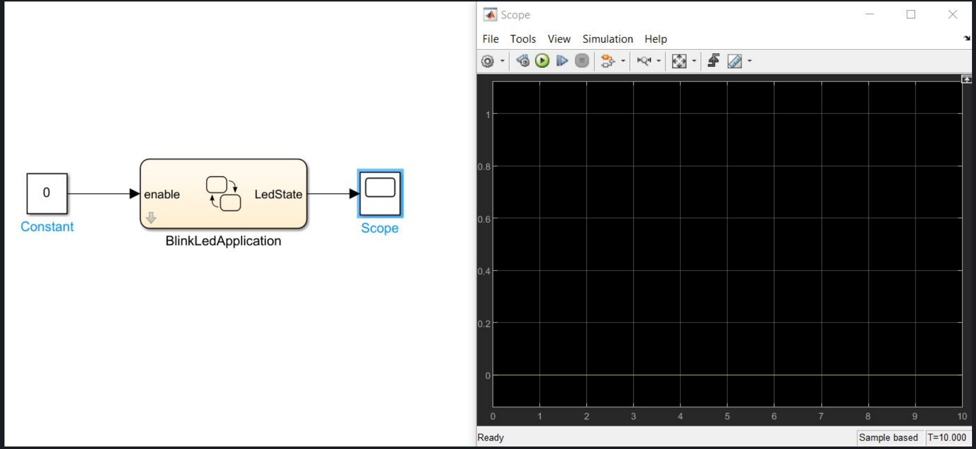
- Exercise: Blink Led with enable signal
  - Now, implement the enable signal that control of system of blink led
    - if the enable signal equal 1 the blink led is on
    - if the enable signal equal 0 the blink led is off



Stateflow Design

What is a finite state machine?

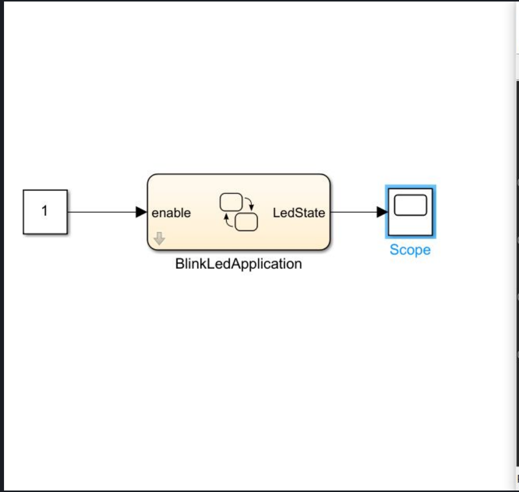
- Exercise: Blink Led with enable signal
  - Now, implement the enable signal that control of system of blink led
    - if the enable signal equal 1 the blink led is on
    - if the enable signal equal 0 the blink led is off

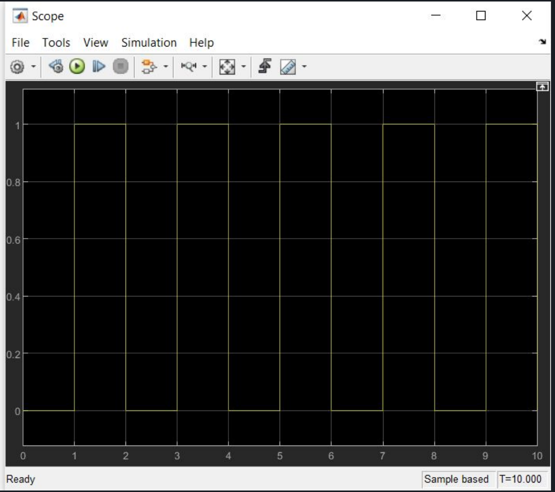


Stateflow Design

What is a finite state machine?

- Exercise: Blink Led with enable signal
  - Now, implement the enable signal that control of system of blink led
    - if the enable signal equal 1 the blink led is on
    - if the enable signal equal 0 the blink led is off





Stateflow Design

What is a finite state machine?

- Exercise: Traffic Light
  - Now, implement the traffic light
    - Normal State
      - Stop State
      - Go State
      - PrepareToStop State
    - Fault State
      - On state
      - Off State

Thank You!