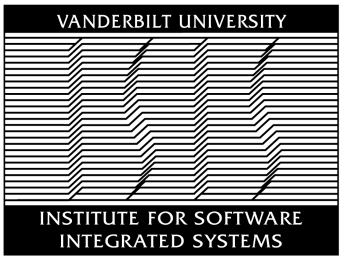
Institute for Software Integrated Systems
Vanderbilt University
Nashville TN 37235



TECHNICAL REPORT

TR #: **ISIS-01-204**

Title: **Analysis of Matlab® Simulink® and Stateflow® Data Model**

Author: **Sandeep Neema**

Abstract

This report presents an analysis of the data models of Matlab Simulink and Stateflow. The data model presented here captures the main classes of objects, their attributes, and their relationships that may occur in a Simulink or Stateflow model. The classes and relationships are captured in a UML class diagram. This study is motivated by the requirements of a framework that will be capable of generating automatic semantic translators, given the data model of the source and destination tools, and the translation/mapping specifications.

KEYWORDS
UML, Simulink, Stateflow, Block Diagrams.

ACKNOWLEDGMENTS
This work was sponsored by the Defense Advanced Research Projects Agency, Information Technology Office, under contract #F30602-00-1-0580.

Introduction

MATLAB is a high-performance language for technical computing from The MathWorks®. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Simulink is a companion program to MATLAB. It is a graphical modeling tool for modeling and simulating nonlinear dynamic systems. It can work with linear, nonlinear, continuous-time, discrete-time, multivariable, and multirate systems. Stateflow is another companion tool to MATLAB. It is a graphical design and development tool for complex control and supervisory logic problems. Stateflow allows visual modeling and simulation of complex reactive systems based on *finite state machine* theory. Stateflow models can be included in a Simulink model as a subsystem. The complete tool suite serves as a powerful design and development environment capable of modeling and simulating complex dynamical systems composed of heterogeneous subsystems.

Simulink, Stateflow and MATLAB together provide excellent modeling and simulation capability; however, often there is a need to subject the models (developed in Simulink) to a more complex, rigorous, and domain-specific analysis. Remodeling the system in the analysis tool's modeling language while possible requires a lot of manual effort. Additionally, maintaining consistency between the Simulink/Stateflow models and the analysis tool's models is error-prone and difficult in the absence of tool support. The popularity of MATLAB, Simulink, and Stateflow implies that significant efforts have already been invested in creating a large model-base in Simulink/Stateflow. It is desirable that application developers take advantage of this effort without foregoing the capabilities of their own analysis and synthesis tools. Owing to these factors a strong need has been expressed for automated semantic translators that can interface with and translate the Simulink/Stateflow models into the models of different analysis and synthesis tools. This study is motivated by the requirements of a framework that will be capable of generating automatic semantic translators, given the data model of the source and destination tools, and the translation/mapping specifications. In this report we describe the data model of Simulink and Stateflow as a UML class diagram. The information provided below is derived from the Simulink documentation as well as by "reverse engineering" a large set of Simulink/Stateflow models.

In order to avoid any ambiguity a complete model of a system in Simulink will be referred to as a Simulink project. A Simulink project is stored in an ASCII text file in a specific format referred to as Model File Format in the Simulink documentation. The Simulink project files are suffixed with ".mdl" and therefore occasionally we may refer to a Simulink project file as an mdl file. There is a clear decoupling between the Simulink and the Stateflow models. When a Simulink projects contain Stateflow models, the Stateflow models are stored in a separate section in the mdl file. We consider the two independently and present their data model in separate sections. It must be remarked that the data models presented here capture only the information that is being exposed by Simulink in the mdl file. A lot of semantic information that is sometimes required for effective understanding of the Simulink models is hidden in the MATLAB simulation engine, or in Simulink's primitive library database.

Simulink Data Model

Simulink models dynamical systems as block diagram, consisting of blocks that represent an elementary dynamic subsystem, and lines that represent connection between block inputs and block outputs. Each block is a black box in the classical sense with a set of inputs, outputs, and a set of states that are internal to the block. Intrinsic to the block are a set of mathematical functions that specify the time-dependent relationships among its inputs, internal states, and outputs. The exact nature of this mathematical relationship is not available in an mdl file, however, a unique block type in the mdl file serves as a reference. Key properties of most standard blocks are parameterized. The parameter values are either constants or MATLAB expressions. The parameters for all the blocks are available as name-value pairs in the mdl file. Simulink allows hierarchical composition of complex systems from subsystems, each of which is composed as a block diagram.

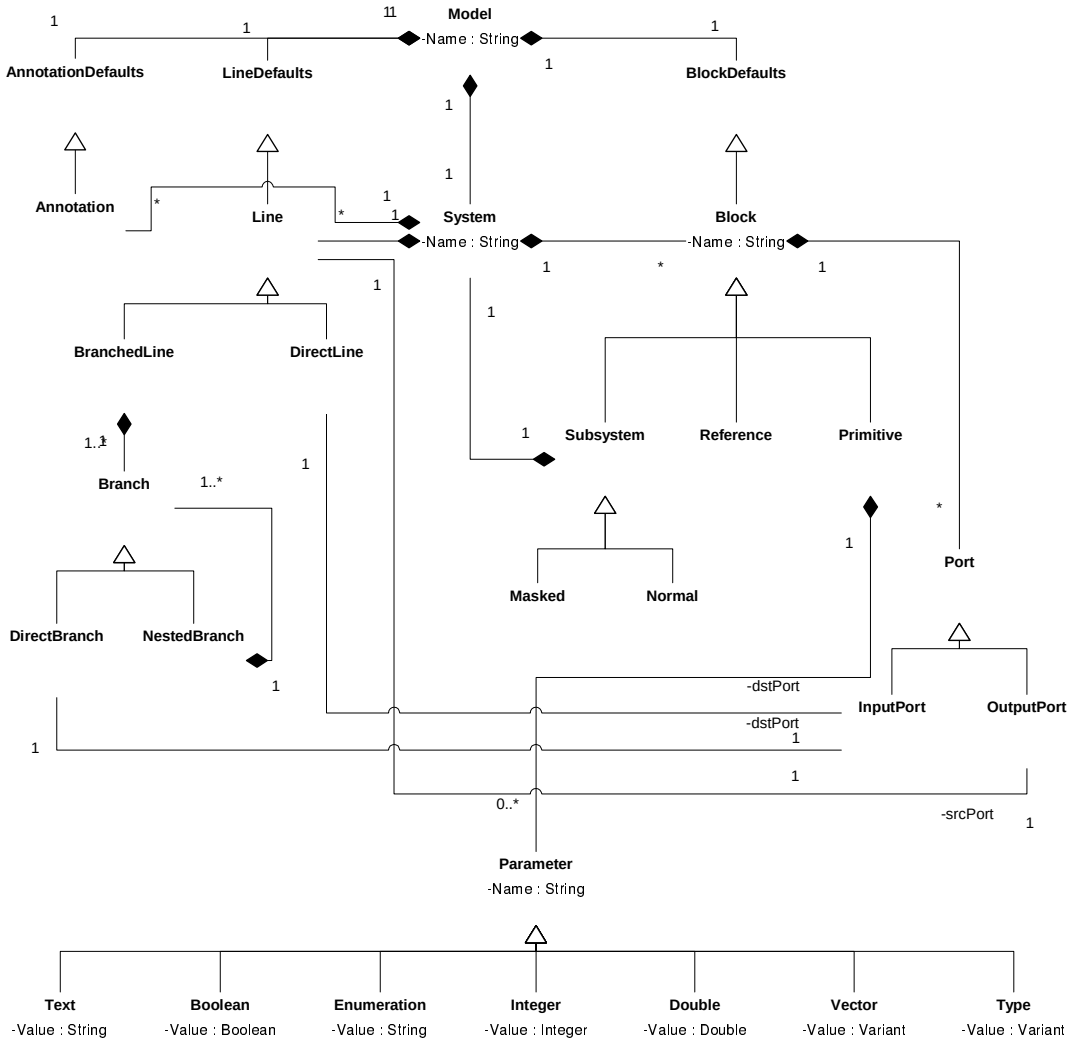


Figure 1: UML class diagram of Simulink data model

The main classes in the Simulink Data Model and their relationship are shown above in Figure 1 as a UML class diagram. In the following paragraphs we briefly describe each of the classes and their attributes. The attributes of each of the class below are described in details in Appendix I.

Model

The *Model* class is the root container in a Simulink project. Every Simulink project contains a single instance of the Model class as the root object. The attributes of the Model class capture global project wide options that include various visual preferences, printing preferences, information about the project, simulation/solver options, code generator options, etc.

System

The *System* class encapsulates a system model in Simulink. It serves as a container for the block diagram that models a dynamical system. It contains *Block* objects and connections (*Line* objects) between the Block objects. Some of these Block objects may be *Subsystem* objects. A Subsystem can be composed as a block diagram. The composition is indirect through a System object i.e. every Subsystem object contains a single System object. This enables hierarchical representation of a complex system. Every Model/Subsystem object contains a single System object. A System object may also contain *Annotation* objects that are used to add documentary information/user comments to the system models. The System class has a small set of attributes that capture various visual preferences, printing preferences, and system information.

Block Class Hierarchy

Blocks are the main functional entities in a Simulink model. A block represents an elementary or compound dynamic subsystem. Primarily the blocks can be categorized into three categories:

1. Primitives – Primitives are basic Simulink library blocks. Simulink provides a rich set of basic building blocks that capture a wide range of mathematical functions. The intrinsic of these blocks are proprietary to Simulink, and are not available in an mdl file. However, documentary information about the intrinsic of the Simulink primitive blocks may be obtained from the Simulink documentation.
2. References – References are link blocks that reference external library blocks. Simulink allows for reuse of modeling efforts by creation of libraries of models, and referencing them in other models. This also provides a way of introducing third-party IP in the models. The library models are essentially subsystems that have been created using Simulink primitives or other libraries. A large variety of advance mathematical processing has been made available to the Simulink users by means of custom libraries or toolboxes. The referencing is implemented as a named association i.e. the referrer stores the library name, and the model name as an attribute.
3. Subsystems – Subsystems are compound blocks that contain a system composed as a block diagram. Subsystems may be further refined into *Masked Subsystems* or *Normal Subsystems*. Masked Subsystems are similar in structure to a Normal Subsystem, however Simulink does not expose the internal structure of a Masked Subsystem. The configurable parameters of the blocks within the masked subsystem are exposed via a user-defined dialog box. Thus, a Masked Subsystem appears like a primitive block to a user.

We recognize this categorization in the UML class diagram by turning Block into an abstract base class and sub-classing it into concrete classes that correspond to the three categories.

Simulink blocks are parameterized. Different blocks have different set of parameters. The *Parameter* class captures these parameters as name-value pairs. A Block can contain an arbitrary number of Parameter objects.

Blocks contain *Ports* that may be *Input Ports* or *Output Ports*. These basically define the interface of the block. The port hierarchy is described below. Simulink primitive library contains two primitive blocks of type InputPort and OutputPort. These should not be confused with the Port objects. When a subsystem contains InputPort block or OutputPort block there is an implicit association between the Ports of the Subsystem object and the InputPort/OutputPort blocks through the port numbers.

Within a system/subsystem blocks are connected to each other through a *Line* object, which represents an association between the Port objects contained in the Block objects.

The primary attributes of the Block class include name, block type, priority, and tag. Each Simulink primitive block has a unique type that serves as an unambiguous reference to the functionality implemented by the block. Simulink orders blocks for execution when simulating the block diagram. This ordering is done based on the data dependency. In the absence of data dependencies, block priorities are used for ordering. The tag attribute is used to capture user annotations. Simulink does not interpret tags. Other attributes of the block capture various visual preferences, graphical information, Matlab interfacing information, etc. The Block class has been derived from *BlockDefaults* (described later) to factor in a set of default attributes.

Parameters

Parameter objects are basically name-value pairs. A block may contain an arbitrary number of Parameters. We define Parameter as a separate class instead of defining these as attributes of block class. Different Primitive blocks have different set of attributes, and while there is a finite set of Primitive Simulink blocks, there are too many of Primitives to be able to subclass and enumerate them all. Additionally, the enumeration minimizes the flexibility of adding new Primitive Simulink blocks.

Parameters can be categorized according to the type of the value they store. We recognize this in the UML class diagram by sub-classing the Parameter class into different classes, each of which holds a different value type.

Port Hierarchy

Ports define the interfaces of blocks. The classes described heretofore Block, Model, System, are first-class concepts in an mdl file and are stored as structured blocks of text enclosed within braces. Ports on the other hand are stored as an attribute of the Block objects in an mdl file. The value of this attribute is a 5-D vector, first two elements of the vector being the number of input ports and the number of output ports respectively. When blocks have just one input port, or one output port, or one input and output port, this attribute is omitted. This presents with some difficulty in disambiguating the number and nature of ports just from the information available in the mdl file. Unambiguously determination of the number and nature of ports is possible only by referring to the documentation of the Simulink primitive block library.

In our data model we recognize ports as first-class concept. The Port class encapsulates a port. The Port class is further specialized into *InputPorts* and *OutputPorts*. The primary attributes of the Port class are name, and number. InputPorts and OutputPorts within a block are uniquely numbered.

Line hierarchy

A Line object connects two Blocks in a System through their Ports. Semantically, a line represents an association between the inputs of one subsystem to the outputs of another subsystem. A Line object is associated with Input Ports and Output Ports. Input Ports play the role of dstPort and Output Ports play the role of srcPort. A Line may be a *BranchedLine* in which case it is associated with an Output Port only and contains one or more *Branch* objects. The Branch objects may be a *DirectBranch* in which case it is directly associated with an Input Port, or it may be a *NestedBranch* in which case it may further contain one or more Branch objects.

The Line class is derived from *LineDefaults* class to factor in the default attributes. The attributes of the Line class capture name, and visual positioning information.

Annotation

An Annotation object is used to add text/comments in a Simulink model. The attributes of the annotation object capture the added text. Additional attributes capture graphical information such as font, font size, color, etc. We derive the Annotation class from *AnnotationDefaults* to factor in the default attributes.

BlockDefaults

A *BlockDefaults* object groups together a set of *Block* attributes and retains project wide defaults for these attributes. These attributes capture various visual preferences. The default value may be overridden by individual Block objects. There is a single instance of BlockDefaults in a Model object.

LineDefaults

A *LineDefaults* object groups together a set of *Line* attributes and retains project wide defaults for these attributes. These attributes capture various visual preferences. The default value may be overridden by individual Line objects. There is a single instance of LineDefaults in a Model object.

AnnotationDefaults

An *AnnotationDefaults* object groups together a set of *Annotation* attributes and retains project wide defaults for these attributes. These attributes capture various visual preferences. The default value may be overridden by individual Annotation objects. There is a single instance of AnnotationDefaults in a Model object.

Stateflow Data Model

Stateflow models behavior of dynamical systems based on finite state machines. The Stateflow modeling formalism is derived from Statecharts developed by Harel [ref], and the Stateflow models follow the same semantics. The key differences from Statecharts are in the action language. Stateflow action language provides has been extended primarily to reference Matlab functions, and Matlab workspace variables. Additionally, a concept of condition action has been added to the transition expression. A condition action is performed every time the condition is evaluated.

The primary entities involved in the creation of a Stateflow model include States, Transitions, Events, Data, and Junctions. Figure 2 shows these entities and their relationship in a UML class diagram. In the following paragraphs each of the classes and their attributes are described briefly. The attributes of each of the classes are described in detail in Appendix II.

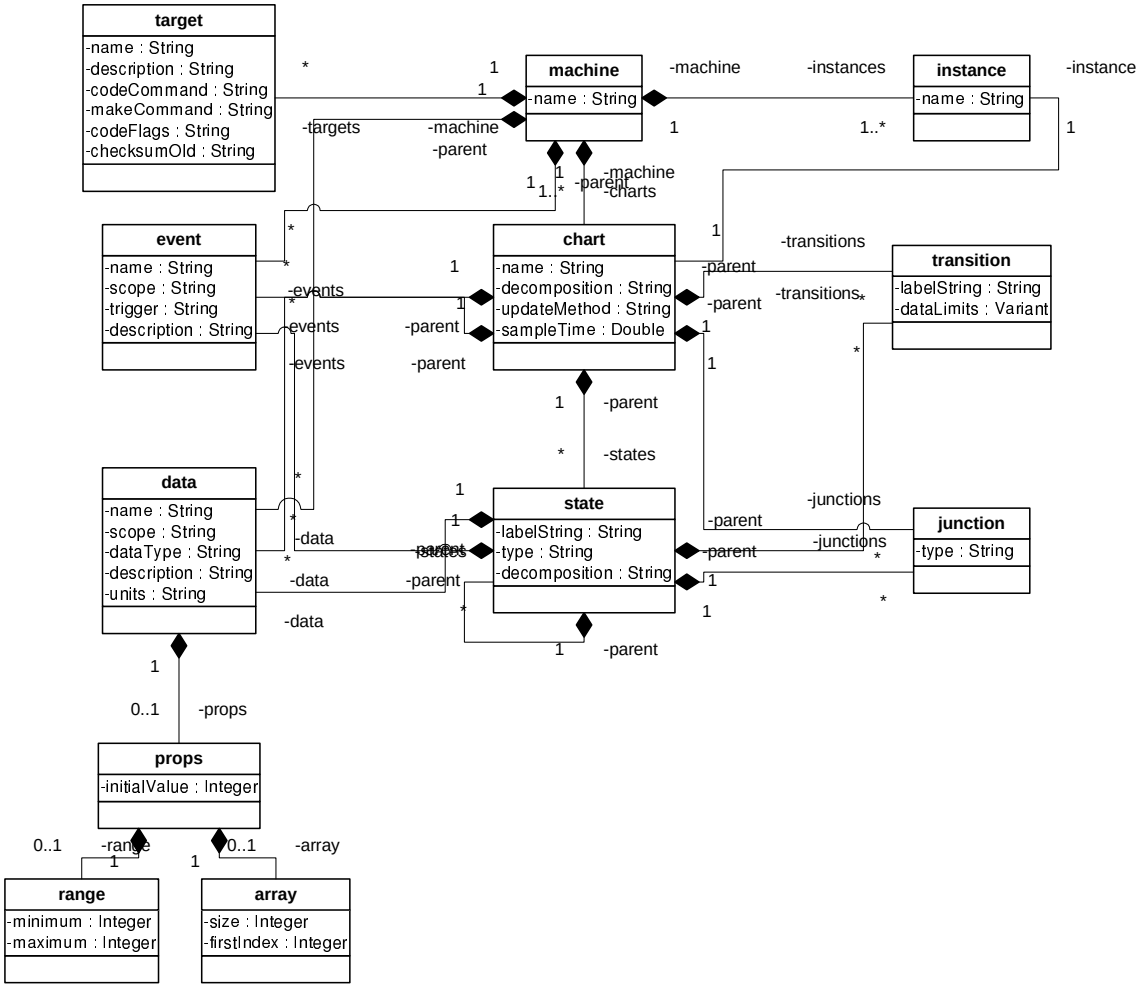


Figure 2: UML diagram of Stateflow

Machine

The *Machine* class is the root container in the Stateflow hierarchy. There can be at most one Machine instance per Simulink project. A Machine object may contain a number of *Chart* instances. A Machine object may also contain *Event*, and *Data* instances, that are available across multiple Charts. The few attributes of the Machine class capture information about the model in terms of author, creation date etc.

Chart

The *Chart* class forms the root state of a hierarchical concurrent state machine. There may be many Chart instances in a Simulink project. Each Chart is uniquely associated with a Simulink block in the Simulink project. This association is made through the *Instance* class. The attributes of Chart class capture visual information about the chart such as screen co-ordinates, various visual preferences such as color, font, etc., and chart properties such as decomposition etc.

State

The *State* class represents a state as defined in the Statechart formalism. Stateflow supports hierarchical composition of state machines i.e. states containing state machines. A state machine can be composed within a State by containing one or more States, and *Transitions*. Two types of composition semantics are possible: parallel and sequential. The decomposition attribute of the State denotes the type of composition semantics to be employed. Parallel decomposition captures concurrency in the system behavior. The State hierarchy is rooted in a Chart. A State may also contain *Events*, and *Data* that are scoped locally within the State. Each state in the Stateflow can be associated with a number of actions. These actions can be entry actions that are performed when the state is entered, exit actions that are performed when the state is exited, during actions that are performed when the system stays in the state, or they can be event actions that are to be performed when the system is in the state and the event occurs. These actions are specified in an action language defined by Stateflow. The details of the action language can be seen in [ref]. The entry, exit, and during actions are captured as attributes of the State class. The event actions are captured as EventAction objects that are event-action pairs. Other attributes of the State capture graphical information, and visual preferences.

Transition

The *Transition* class represents a transition as defined in the Statechart formalism. Every Transition may be associated with a trigger, a condition, a number of condition actions, and a number of actions. A trigger is a Boolean expression over Events. A condition is an expression that evaluates to true or false. Condition actions are actions expressed in action language that must be performed when the condition is evaluated. Other actions are performed when the transition is enabled and taken. Attributes of the Transition class capture the trigger, the guard, condition actions, and actions. Additional attributes capture graphical information.

Junction

The *Junction* class represents a junction as defined in the Statechart formalism. Two types of junctions are possible: 1) Connective junctions: these are used to create multiple transitions with different conditions and actions between a pair of states; and 2) History junctions: these are used to specify the destination sub-state of a transition based on historical information in a hierarchical state machine. History junction keeps track of most recently active state. The attributes of the Junction class capture the type of the junction, and graphical information.

Event

The *Event* class represents an event as defined in the Statechart formalism. Events are scoped and a triggering mechanism is defined. Thus Events could be rising edge, falling edge, or either edge trigger events. The attributes of the Event class capture the scope, and triggering mechanism.

Data

The *Data* class represents data as defined in the Statechart formalism. In the Simulink Stateflow interaction semantics data as well as events are used to exchange information between Simulink blocks and Stateflow blocks. The attributes of the class capture the scope, the data type, and the units.

Instance

The *Instance* class captures the link to the Simulink block that contains the chart.

Target

The *Target* class represents a target for the Stateflow models. The target may be a simulation target (MATLAB S-Function), or a synthesis target that generates code. Attributes are specific to the particular target and capture the various options and preferences.

Appendix I: Attributes of the Classes in Simulink Data Model

In this appendix we tabulate all the attributes of the different classes in the Simulink Data Model. The attributes are tabulated with their names, data types, a short description, and a categorization. The attributes grouped into the following categories:

- Core – These attributes are the core attributes of the class and occasionally have semantic implications. Examples of core attributes include name, block type, state decomposition, etc.
- Information – These attributes store some form of documentary information such as description, author name, version, date of creation/modification, etc.
- Visual – These attributes capture graphical information about the object such as window/screen co-ordinates.
- Visual preferences – These attributes capture various graphical options or preferences such as fonts, colors, show tips, show names, show labels, etc.
- Printing preferences – These attributes capture various options that assume relevance when the models are printed. Attributes such as paper size, orientation (portrait or landscape), printing margins, etc. fall in this category.
- Simulation – These attribute capture various options that are used by Simulink when simulating a model.
- Diagnostics – These attributes capture options that are used when checking a model for well-formedness.
- Matlab – These attributes are Matlab specific. They refer to Matlab Workspace variables and Matlab functions that are to be invoked as callback functions when some model is opened or closed.
- Code generator preferences – These attributes capture options that used by the Real-Time Workshop ® code generator when generating code from Simulink models.

Model Class Attributes

ATTRIBUTE	DESCRIPTION	VALUE	CATEGORY	ACCESSIBLE
Name	Model name	Quoted String	Core	Yes
Version	Simulink version used to create/modify the model	Literal	Information	No
SimParamPage	Simulation Parameters dialog box page to display (page last displayed)	Quoted Enum {Solver | WorkspaceI/O | Diagnostics }	Visual Preference	No
SampleTimeColors	Sample Time Colors menu option	Boolean {on | off}	Visual Preference	No

InvariantConstants	Invariant constant setting	Boolean {on | off}	Matlab	No
WideVectorLines	Wide Vector Lines menu option	Boolean {on | off}	Visual Preference	No
ShowLineWidths	Show Line Widths menu option	Boolean {on | off}	Visual Preference	No
SimulationMode		Quoted Enum {normal | …}	Simulation	No
BlockDataTips	Show tips on block data	Boolean {on | off}	Visual Preference	No
BlockParametersDataTip	Show tips on block parameters	Boolean {on | off}	Visual Preference	No
BlockAttributesDataTip	Show tips on block attributes	Boolean {on | off}	Visual Preference	No
BlockPortWidthsDataTip	Show tips on block port widths	Boolean {on | off}	Visual Preference	No
BlockDescriptionStringDataTip	Show tips on block description string	Boolean {on | off}	Visual Preference	No
BlockMaskParametersDataTip	Show tips on block mask parameter dialog	Boolean {on | off}	Visual Preference	No
ToolBar	Show tool bar	Boolean {on | off}	Visual Preference	No
StatusBar	Show status bar	Boolean {on | off}	Visual Preference	No
BrowserShowLibraryLinks	Show library links on a reference block	Boolean {on | off}	Visual Preference	No
BrowserLookUnderMasks	Show internals of masked blocks	Boolean {on | off}	Visual Preference	No
PaperOrientation	Printing paper orientation	Quoted Enum {portrait | landscape}	Printing Preference	No

PaperPosition	Position of diagram on paper	Vector [left bottom width height]	Printing Preference	No
PaperPositionMode	Paper position mode	Quoted Enum {auto | manual}	Printing Preference	No
PaperSize	Size of `PaperType` in `PaperUnits`	Vector [width height]	Printing Preference	No
PaperType	Printing paper type	Quoted Enum {usletter | uslegal | a0 | a1 | a2 | a3 | a4 | a5 | b0 | b1 | b2 | b3 | b4 | b5 | arch-A | arch-B | arch-C | arch-D | arch-E | A | B | C | D | E | tabloid}	Printing Preference	No
PaperUnits	Printing paper size units	Quoted Enum {normalized | inches | centimeters | points}	Printing Preference	No
StartTime	Simulation start time	Quoted Real	Simulation	No
StopTime	Simulation stop time	Quoted Real	Simulation	No
Solver	The solver to be used	Quoted Enum {ode45 | ode23 | ode113 | ode15s | ode23s | ode5 | ode4 | ode3 | ode2 | ode1 | FixedStepDiscrete | VariableStepDiscrete}	Simulation	No
RelTol	Relative error tolerance	Quoted Real	Simulation	No
AbsTol	Absolute error tolerance	Quoted Real	Simulation	No
Refine	Refine factor	Quoted Integer	Simulation	No
MaxStep	Maximum step size	Quoted Integer | Quoted Enum {auto}	Simulation	No
InitialStep	Initial step size	Quoted Integer | Quoted Enum{auto}	Simulation	No
FixedStep	Fixed step size	Quoted Integer | Quoted Enum{auto}	Simulation	No

MaxOrder	Maximum order for ode15s	Quoted Integer {1 | 2 | 3 | 4 | 5}	Simulation	No
OutputOption	Output option	Quoted Enum {AdditionalOutputTimes | RefineOutputTimes | SpecifiedOutputTimes}	Simulation	No
OutputTimes	Values for chosen OutputOption	Vector	Simulation	No
LoadExternalInput	Load input from workspace	Boolean {on | off}	Matlab	No
ExternalInput	Time and input variable names	Quoted Scalar | Quoted Vector [t, u]	Matlab	No
SaveTime	Save simulation time	Boolean {on | off}	Matlab	No
TimeSaveName	Simulation time name	Quoted String	Matlab	No
SaveState	Save states	Boolean {on | off}	Matlab	No
StateSaveName	State output name	Quoted String	Matlab	No
SaveOutput	Save simulation output	Boolean {on | off}	Matlab	No
OutputSaveName	Simulation output name	Quoted String	Matlab	No
LoadInitialState	Load initial state	Boolean {on | off}	Matlab	No
InitialState	Initial state name or values	Quoted String | Vector	Matlab	No
SaveFinalState	Save final state	Boolean {on | off}	Matlab	No
FinalStateName	Final state name	Quoted String	Matlab	No
LimitMaxRows	Limit output	Boolean {on | off}	Matlab	No
MaxRows	Maximum number of output rows to save	Quoted Integer	Matlab	No

Decimation	Decimation factor	Quoted Integer	Matlab	No
AlgebraicLoopMsg	Algebraic loop diagnostic	Quoted Enum {none | warning | error}	Diagnostics	No
MinStepSizeMsg	Minimum step size diagnostic	Quoted Enum {warning | error}	Diagnostics	No
UnconnectedInputMsg	Unconnected input ports diagnostic	Quoted Enum {none | warning | error}	Diagnostics	No
UnconnectedOutputMsg	Unconnected output ports diagnostic	Quoted Enum {none | warning | error}	Diagnostics	No
UnconnectedLineMsg	Unconnected lines diagnostic	Quoted Enum {none | warning | error}	Diagnostics	No
InheritedInSrcMsg		Quoted Enum {none | warning | error}	Diagnostics	No
IntegerOverflowMsg	Integer Over flow	Quoted Enum {none | warning | error}	Diagnostics	No
UnnecessaryDatatypeConvMsg	Unnecessary Conversion	Quoted Enum {none | warning | error}	Diagnostics	No
Int32ToFloatConvMsg	Int32 to float conversion	Quoted Enum {none | warning | error}	Diagnostics	No
SignalLabelMismatchMsg	Signal label mismatch	Quoted Enum {none | warning | error}	Diagnostics	No
ConsistencyChecking	Consistency checking	Boolean {on | off}	Diagnostics	No
ZeroCross	Intrinsic zero crossing detection	Boolean {on | off}	Simulation	No
BooleanDataType	Enable Boolean mode	Boolean {on | off}	Simulation	No
OptimizeBlockIOStorage		Boolean {on | off}	Code Generation	No
BufferReuse	Enable reuse of block I/O buffers	Boolean {on | off}	Code Generation	No
RTWSystemTargetFile	Target file	Quoted String	Code Generation	No

RTWInlineParameters		Boolean {on | off}	Code Generation	No
RTWRetainRTWFile		Boolean {on | off}	Code Generation	No
RTWTemplateMakefile		Quoted String	Code Generation	No
RTWMakeCommand		Quoted String	Code Generation	No
RTWGenerateCodeOnly		Boolean {on | off}	Code Generation	No
ExtModeMexFile		Quoted String	Simulation	No
ExtModeBatchMode		Boolean {on | off}	Simulation	No
ExtModeTrigType		Quoted Enum {manual | …}	Simulation	No
ExtModeTrigMode		Quoted Enum {oneshot | …}	Simulation	No
ExtModeTrigPort		Quoted Integer	Simulation	No
ExtModeTrigElement		Quoted Enum {any | …}	Simulation	No
ExtModeTrigDuration		Integer	Simulation	No
ExtModeTrigHoldOff		Integer	Simulation	No
ExtModeTrigDelay		Integer	Simulation	No
ExtModeTrigDirection		Quoted Enum { rising | falling | … }	Simulation	No
ExtModeTrigLevel		Integer	Simulation	No
ExtModeArchiveMode		Quoted Enum { off | … }	Simulation	No
ExtModeAutoIncOneShot		Boolean {on | off}	Simulation	No
ExtModeIncDirWhenArm		Boolean {on | off}	Simulation	No
ExtModeAddSuffixToVar		Boolean {on | off}	Simulation	No
ExtModeWriteAllDataToWs		Boolean {on | off}	Simulation	No
ExtModeArmWhenConnect		Boolean {on | off}	Simulation	No
Created	Creation date	Quoted String	Information	No
UpdateHistory	Update history	Quoted String	Information	No
ModifiedByFormat		Quoted String	Information	No

LastModifiedBy		Quoted String	Information	No
ModifiedDateFormat		Quoted String	Information	No
LastModifiedDate		Quoted String	Information	No
ModelVersionFormat		Quoted String	Information	No
ConfigurationManager		Quoted String	Information	No
CloseFcn	Close callback	Quoted String	Matlab	No
PreLoadFcn	Pre-load callback	Quoted String	Matlab	No
PostLoadFcn	Post-load callback	Quoted String	Matlab	No
SaveFcn	Save callback	Quoted String	Matlab	No
StartFcn	Start simulation callback	Quoted String	Matlab	No
StopFcn	Stop simulation callback	Quoted String	Matlab	No

System Class Attributes

ATTRIBUTE	DESCRIPTION	VALUE	CATEGORY	ACCESSIBLE
Name	Name of the system	Quoted String	Core	Yes
Location	Screen position	Vector [left, bottom, width, height]	Visual	No
Open	Whether system/subsystem is open and visible	Boolean {on | off}	Visual Preference	No
ModelBrowserVisibility	Is the model browser visible	Boolean {on | off}	Visual Preference	No
ModelBrowserWidth	Width of the model browser	Integer	Visual Preference	No
ScreenColor	Color of the system screen	Quoted Enum {automatic | … }	Visual Preference	No

PaperOrientation	Orientation when model is printed	Quoted Enum {portrait | landscape}	Printing Preference	No
PaperType	Paper Type	Quoted Enum	Printing Preference	No
PaperUnits	Paper Units	Quoted Enum	Printing Preference	No
ZoomFactor	Degree of zoom	Quoted Integer	Visual Preference	No
AutoZoom	Should zoom automatically	Boolean {on | off}	Visual Preference	No
ReportName	Name of report file	Quoted String	Matlab	No

Block Class Attribute

ATTRIBUTE	DESCRIPTION	VALUES	CATEGORY	ACCESSIBLE
Name	Name of the block	Quoted String	Core	Yes
BlockType	Type of the block	Unquoted Literal	Core	Yes
Description	User-specifiable description	Quoted String	Information	Yes
Priority	Specifies the block's sequencing during execution relative to other blocks in the same priority window	Quoted Integer	Core	Yes
Tag	User-defined string	Quoted String	Information	Yes
AttributesFormat String	Specifies parameters to be displayed below block in a block diagram	Quoted String	Visual Preference	No
Orientation	Where block faces	Quoted Enum {right | left | down | up}	Visual Preference	No
ForegroundColor	Block name, icon, outline, output	Quoted Enum {black | white | red	Visual Preference	No

	signals, and signal label	| green | blue | cyan | magenta | yellow | gray | lightBlue | orange | darkGreen}		
BackgroundColor	Block icon background	Quoted Enum {black | white | red | green | blue | cyan | magenta | yellow | gray | lightBlue | orange | darkGreen}	Visual Preference	No
DropShadow	Display drop shadow	Boolean {off | on}	Visual Preference	No
NamePlacement	Position of block name	Quoted Enum {normal | alternate}	Visual Preference	No
FontName	Font	Quoted Enum {Helvetica | …}	Visual Preference	No
FontSize	Font size	Unquoted Integer	Visual Preference	No
FontWeight	Font weight	Quoted Enum {light | normal | demi | bold}	Visual Preference	No
FontAngle	Font angle	Quoted Enum {normal | italic | oblique}	Visual Preference	No
Position	Position of block in model window	Unquoted Vector [left top right bottom]	Visual	Yes
ShowName	Display block name	Boolean {on | off}	Visual Preference	No
ShowPortLabels	Display port labels	Boolean {on | off}	Visual Preference	No
CloseFcn	Close callback	Quoted String {MATLAB expression}	Matlab	No
CopyFcn	Copy callback	Quoted String {MATLAB expression}	Matlab	No
DeleteFcn	Delete callback	Quoted String	Matlab	No

		{MATLAB expression}		
InitFcn	Initialization callback	Quoted String {MATLAB expression}	Matlab	No
LoadFcn	Load callback	Quoted String {MATLAB expression}	Matlab	No
ModelCloseFcn	Model close callback	Quoted String {MATLAB expression}	Matlab	No
NameChangeFcn	Block name change callback	Quoted String {MATLAB expression}	Matlab	No
OpenFcn	Open callback	Quoted String {MATLAB expression}	Matlab	No
ParentCloseFcn	Parent subsystem close callback	Quoted String {MATLAB expression}	Matlab	No
PreSaveFcn	Pre-save callback	Quoted String {MATLAB expression}	Matlab	No
PostSaveFcn	Post-save callback	Quoted String {MATLAB expression}	Matlab	No
StartFcn	Start simulation callback	Quoted String {MATLAB expression}	Matlab	No
StopFcn	Termination of simulation callback	Quoted String {MATLAB expression}	Matlab	No
UndoDeleteFcn	Undo block delete callback	Quoted String {MATLAB expression}	Matlab	No
LinkStatus	Link status of block for library blocks	Quoted Enum {none | resolved | unresolved | implicit}	Matlab	No

Port Class Attributes

ATTRIBUTE	DESCRIPTION	VALUE	CATEGORY	ACCESSIBLE
PortNumber	A block-wide unique number to identify the port	Integer	Core	Yes
Name	The label of the port	Quoted String	Core	Yes
TestPoint	Whether this port is a test point	Boolean {on | off}	Simulation	No
RTWStorageClass	When the Simulink model is synthesized using the Real-Time Workshop® toolset the storage class of the port	Quoted Enum {Auto | …}	Code Generation	No

Line Class Attributes

ATTRIBUTE	DESCRIPTION	VALUE	CATEGORY	ACCESSIBLE
Name	A name associated with a line	Quoted String	Core	Yes
Labels	The location where the label should appear	Unquoted Vector	Visual	No
Points	Intersection points	Unquoted Vector	Visual	Yes

Annotation Class Attributes

ATTRIBUTE	DESCRIPTION	VALUE	CATEGORY	ACCESSIBLE
Position	Screen position of the annotation	Unquoted Vector	Visual	Yes
Text	The annotation text	Quoted String	Core	Yes

BlockDefaults Class Attributes

ATTRIBUTE	DESCRIPTION	VALUE	CATEGORY	ACCESSIBLE
Orientation	The orientation of the block	Quoted Enum {left | right}	Visual Preference	No
ForegroundColor	Foreground Color	Quoted Enum {black | white | … }	Visual Preference	No

BackgroundColor	Background Color	Quoted Enum {black | white | … }	Visual Preference	No
DropShadow	Drop shadow	Boolean {on | off}	Visual Preference	No
NamePlacement	Location of the name of block	Quoted Enum {normal | …}	Visual Preference	No
FontName	Font to be used	Quoted Enum {Helvetica | …}	Visual Preference	No
FontSize	Size of the font	Integer	Visual Preference	No
FontWeight	Font weight	Quoted Enum {normal | …}	Visual Preference	No
FontAngle	Font angle	Quoted Enum {normal | …}	Visual Preference	No
ShowName	Show the name of the block on screen	Boolean {on | off}	Visual Preference	No

LineDefaults Class Attributes

ATTRIBUTE	DESCRIPTION	VALUE	CATEGORY	ACCESSIBLE
FontName	Font to be used	Quoted Enum {Helvetica | …}	Visual Preference	No
FontSize	Size of the font	Integer	Visual Preference	No
FontWeight	Font weight	Quoted Enum {normal | …}	Visual Preference	No
FontAngle	Font angle	Quoted Enum {normal | …}	Visual Preference	No

AnnotationDefaults Class Attributes

ATTRIBUTE	DESCRIPTION	VALUE	CATEGORY	ACCESSIBLE
HorizontalAlignment	The horizontal alignment of text in an annotation	Quoted Enum {left | center | right | justified}	Visual Preference	No

VerticalAlignment	The vertical alignment of text in an annotation	Quoted Enum {top | middle | bottom}	Visual Preference	No
ForegroundColor	Foreground Color	Quoted Enum {black | white | … }	Visual Preference	No
BackgroundColor	Background Color	Quoted Enum {black | white | … }	Visual Preference	No
DropShadow	Drop shadow	Boolean {on | off}	Visual Preference	No
FontName	Font to be used	Quoted Enum {Helvetica | …}	Visual Preference	No
FontSize	Size of the font	Integer	Visual Preference	No
FontWeight	Font weight	Quoted Enum {normal | …}	Visual Preference	No
FontAngle	Font angle	Quoted Enum {normal | …}	Visual Preference	No

Appendix II: Attributes of the Classes in Stateflow Data Model

Machine Class Attributes

ATTRIBUTE	DESCRIPTION	VALUE	CATEGORY	ACCESSIBLE
name	The name of the machine	Quoted String	Core	Yes
creator	Author	Quoted String	Information	No
created	Model creation date and time	Quoted String	Information	No
version	Version of the model	Quoted Literal	Information	No
document		Quoted String	Information	No
isLibrary	Whether the model is a library model	Boolean {on | off}	Matlab	No
sfVersion	Version of the Stateflow tools with which the Stateflow diagram was generated.	Unquoted Literal	Information	No

Chart Class Attributes

ATTRIBUTE	DESCRIPTION	VALUE	CATEGORY	ACCESSIBLE
name	Name of the chart	Quoted String	Core	Yes
windowPosition	Window co-ordinates	Unquoted Vector	Visual	No
viewLimits	A graphical attribute	Unquoted Vector	Visual	No
zoomFactor	Zoom Factor	Unquoted Double	Visual Preference	No
screen	Screen co-ordinates of the chart object	Unquoted Vector	Visual	No
visible	Whether the chart is open and visible	Boolean {1 | 0}	Visual Preference	No

decomposition	The decomposition of the state chart.	Unquoted Enum {SET_CHART, CLUSTER_CHART, … }	Core	Yes
chartFileNumber		Unquoted Integer	Matlab	No
transitionColor	A 3-d color vector [r g b]	Unquoted Vector	Visual Preference	No
transitionLableColor	A 3-d color vector [r g b]	Unquoted Vector	Visual Preference	No
selectionColor	A 3-d color vector [r g b]	Unquoted Vector	Visual Preference	No
disableImplicitCasting		Boolean	Matlab	No
stateColor	A 3-d color vector [r g b]	Unquoted Vector	Visual Preference	No
stateLabelColor	A 3-d color vector [r g b]	Unquoted Vector	Visual Preference	No
junctionColor	A 3-d color vector [r g b]	Unquoted Vector	Visual Preference	No
chartColor	A 3-d color vector [r g b]	Unquoted Vector	Visual Preference	No
description	Chart description	Quoted String/s	Information	No

State Class Attributes

ATTRIBUTE	DESCRIPTION	VALUE	CATEGORY	ACCESSIBLE
name	The name of the state	Quoted String	Core	Yes
entryAction	The entry action associated with the state	Quoted String	Core	Yes
exitAction	The exit action associated with the state	Quoted String	Core	Yes
duringAction	The during action associated with the state	Quoted String	Core	Yes
position	Window position	Unquoted Vector	Visual	Yes

type	The "decomposition" of the state	Unquoted Enum {OR_STATE | AND_STATE}	Core	Yes
decomposition		Unquoted Enum { CLUSTER_STATE, … }	Matlab	Yes
arrowSize		Unquoted Double	Visual Preference	No
fontSize		Unquoted Double	Visual Preference	No

Junction Class Attributes

ATTRIBUTE	DESCRIPTION	VALUE	CATEGORY	ACCESSIBLE
position	Window co-ordinates	Unquoted Vector	Visual	Yes
type	The type of the junction	Unquoted Enum {CONNECTIVE_JUNCTION | HISTORY_JUNCTION }	Core	Yes
arrowSize		Unquoted Double	Visual Preference	No

Transition Class Attributes

ATTRIBUTE	DESCRIPTION	VALUE	CATEGORY	ACCESSIBLE
trigger	The trigger associated with a transition	Quoted String	Core	???
condition	The condition associated with a transition	Quoted String	Core	???
conditionAction	The condition actions associated with a transition	Quoted String	Core	???
action	The actions associated with a transition	Quoted String	Core	???
fontSize	Font size	Unquoted Double	Visual Preference	No
arrowSize		Unquoted Double	Visual Preference	No

labelPosition	The label graphical co-ordinates	Unquoted Vector	Visual	No
midpoint	Mid point of the line denoting transition	Unquoted Vector	Visual	No
dataLimits		Unquoted Vector	Matlab	Yes

Event Class Attributes

ATTRIBUTE	DESCRIPTION	VALUE	CATEGORY	ACCESSIBLE
name	The name of the event	Quoted String	Core	Yes
scope	Scope of the event	Unquoted Enum { LOCAL_EVENT | INPUT_EVENT | OUTPUT_EVENT | EXPORTED_EVENT? | IMPORTED_EVENT }	Core	Yes
trigger	The triggering mechanism of the event	Unquoted Enum { EITHER_EDGE_EVENT | RISING_EDGE_EVENT | FALLING_EDGE_EVENT }	Core	Yes
description	Some description about the event	Quoted String	Information	Yes

Data Class Attributes

ATTRIBUTE	DESCRIPTION	VALUE	CATEGORY	ACCESSIBLE
name	The name of the data object	Quoted String	Core	Yes
scope	The scope of the data object	Unquoted Enum { LOCAL_DATA | INPUT_DATA | OUTPUT_DATA | TEMPORARY_DATA | WORKSPACE_DATA | CONSTANT_DATA | …}	Core	Yes
dataType	The data type	Quoted String	Core	Yes

units	The data units	Quoted String	Core	Yes
description	Some description	Quoted String	Information	Yes

Instance Class Attributes

ATTRIBUTE	DESCRIPTION	VALUE	CATEGORY	ACCESSIBLE
name	The Simulink name of the chart	Quoted String	Core	Yes

Target Class Attributes

ATTRIBUTE	DESCRIPTION	VALUE	CATEGORY	ACCESSIBLE
name	Name of the target	Quoted String	Core	Yes
description	Description about the target	Quoted String	Information	Yes
codeCommand	Code generation/compilation command	Quoted String	Code Generation	Yes
makeCommand	Code generation/compilation command	Quoted String	Code Generation	Yes
codeFlags	Code generation/compilation flags	Quoted String	Code Generation	Yes
checksumOld		Quoted String	Matlab	Yes