1. Record Nr. UNINA9910974333203321

Autore	Karris Steven T
Titolo	Introduction to stateflow with applications / / Steven T. Karris
Pubbl/distr/stampa	Fremont, CA, : Orchard Publications, 2007
ISBN	1-280-95526-0 9786610955268 1-934404-08-X
Edizione	[1st ed.]
Descrizione fisica	iv, (522) p
Disciplina	620.001/13
Soggetti	Computer software - Development
Lingua di pubblicazione	Inglese
Formato	Materiale a stampa
Livello bibliografico	Monografia
Note generali	Title from title screen.
Nota di bibliografia	Includes bibliographical references and index.
Nota di contenuto	Intro -- Preface -- Table of Contents -- Chapter 1 -- 1.1 Finite State Machines -- 1.2 Event-Driven Systems -- 1.3 Construction of Finite-State Machines with Stateflow -- 1.4 Procedure for Creating a Stateflow Chart -- 1.5 Summary -- 1.6 Exercise for the Reader -- 1.7 Solution to the End-of-Chapter Exercise -- Chapter 2 -- 2.1 Truth Tables in Stateflow -- 2.2 Summary -- 2.3 Exercises -- 2.4 Solution to End-of-Chapter Exercises -- Chapter 3 -- 3.1 Introduction to Embedded MATLAB Functions -- 3.2 Building the Model with a Stateflow Embedded MATLAB Function -- 3.3 Programming the Stateflow Chart with an Embedded MATLAB Function -- 3.4 Simulation of the Matrix Operations Stateflow Chart -- 3.5 Summary -- 3.6 Exercises for the Reader -- 3.7 Solution to the End-of-Chapter Exercises -- Chapter 4 -- 4.1 Introduction to Embedded MATLAB Functions -- 4.2 Summary -- 4.3 Exercises for the Reader -- 4.4 Solution to the End-of-Chapter Exercises -- Chapter 5 -- 5.1 Introduction to Graphical Functions -- 5.2 Creating a Graphical Function -- 5.3 Subcharts -- 5.4 Exporting Graphical Functions to Stateflow -- 5.5 Summary -- 5.6 Exercise for the Reader -- 5.7 Solution to the End-of-Chapter Exercise -- Chapter 6 -- 6.1 The Stateflow Connective Junction Tool -- 6.2 Creating a Connective Junction -- 6.3 Changing Connective Junction Size -- 6.4 Changing Connective Junction Properties -- 6.5 Uses of Connective Junctions -- 6.6 Summary -- 6.7 Exercise for the Reader -- 6.8

Solution to the End-of-Chapter Exercise -- Chapter 7 -- 7.1 History Junction Defined -- 7.2 The Stateflow History Junction Tool -- 7.3 Changing the History Junction Size -- 7.4 Changing History Junction Properties -- 7.5 Entering a State -- 7.6 Executing an Active State -- 7.7 Exiting an Active State -- 7.8 Execution Order for Parallel States -- 7.9 Transitions -- 7.10 Transition Connections.
7.11 Inner Transitions -- 7.12 Summary -- 7.13 Exercise for the Reader -- 7.14 Solution to the End-of-Chapter Exercise -- Chapter 8 -- 8.1 Creating a Box -- 8.2 Changing a State to a Box -- 8.3 Using Boxes in Stateflow -- 8.4 Summary -- Chapter 9 -- 9.1 Mealy Machine Defined -- 9.2 Moore Machine Defined -- 9.3 Mealy and Moore Machines in Stateflow -- 9.4 Creating a Mealy Chart -- 9.5 Creating a Moore Chart -- 9.6 Changing Chart Type -- 9.7 Debugging Mealy and Moore Charts -- 9.8 Summary -- 9.9 Exercises for the Reader -- 9.10 Solution to the End-of-Chapter Exercises -- Appendix A -- A.1 MATLAB® and Simulink® -- A.2 Command Window -- A.3 Roots of Polynomials -- A.4 Polynomial Construction from Known Roots -- A.5 Evaluation of a Polynomial at Specified Values -- A.6 Rational Polynomials -- A.7 Using MATLAB to Make Plots -- A.8 Subplots -- A.9 Multiplication, Division, and Exponentiation -- A.10 Script and Function Files -- A.11 Display Formats -- Appendix B -- B.1 Simulink and its Relation to MATLAB -- B.2 Simulink Demos -- Appendix C -- Masked Subsystems -- his appendix presents an overview of masked subsystems, and a step-by-step procedure to create custom user interfaces, i.e., masks for Simulink subsystems. -- C.1 Masks Defined -- A mask is a custom user interface for a subsystem. A masked subsystem conceals the subsystem's contents, and it appear to the user as an atomic block with its own icon and parameter dialog box. However, a masked subsystem provides only graphi... -- C.2 Advantages Using Masked Subsystems -- A masked subsystem allows us to -- 1. Replace the parameter dialogs of a subsystem and its contents with a single parameter dialog with its own block description, parameter prompts, and help text. -- 2. Replace a subsystem's standard icon with a custom icon that shows its purpose.
3. Prevent accidental modification of subsystems by concealing their contents behind a mask. -- 4. Placing a masked subsystem in a library. We can also mask S-Function and Model blocks. -- C.3 Mask Features -- Masks can include any of the following features: -- Mask Icon - The mask icon replaces a subsystem's standard icon, i.e., it appears in a block diagram in place of the standard icon for a subsystem block. Simulink uses MATLAB code that we supply to draw the custom icon. We can use any MATLAB d... -- Mask Parameters - Masked subsystems allow us to define a set of user-specified parameters. Simulink stores the values of these parameters in the mask workspace as the value of a variable whose name you specify. These associated variables allo... -- Mask Parameter Dialog Box - The mask parameter dialog box contains controls that enable a user to set the values of the mask's parameters and hence the values of any internal parameters linked to the mask parameters. The mask parameter dialog... -- Mask Initialization Code - The initialization code is MATLAB code that you specify and that Simulink runs to initialize the masked subsystem at critical times, such as model loading and the start of a simulation run (see Initialization Pane).... -- Mask Workspace - Simulink associates a workspace with each masked subsystem that you create. Simulink stores the current values of the subsystem's parameters in the workspace as well as any variables created by the block's initialization code... -- A block parameter expression can refer only to variables defined in the mask workspaces of the subsystem or nested subsystems that contain the

block or in the model's workspace. -- A valid reference to a variable defined on more than one level in the model hierarchy resolves to the most local definition.
For example, let us suppose that model M contains masked subsystem A, which contains masked subsystem B. Also, let us suppose that B refers to a variable x that exists in both A's and M's workspaces. In this case, the reference resolves to th... -- A masked subsystem's initialization code can refer only to variables in its local workspace. -- The mask workspace of a Model block is not visible to the model that it references. Any variables used by the referenced model must resolve to workspaces defined in the referenced model or to the base (i.e., the MATLAB) workspace. -- C.4 Creating a Masked Subsystem -- It is best to illustrate the creation of a masked subsystem with an example. -- Example C.1 -- The Simulink model in Figure C.1 below implements the quadratic equation . -- Figure C.1. Simulink model for Example C.1 -- To create a subsystem, we encircle all blocks except the Unknown x and Display blocks, and from the Edit drop menu we select Create Subsystem. The model now appears as shown in Figure C.2. -- Figure C.2. The model for Example C.1 shown as a subsystem block -- To see the contents of the Subsystem in Figure C.2, we double-click the Subsystem block and now the model appears as shown in Figure C.3. -- Figure C.3. The contents of the subsystem block -- From the Edit drop menu we click on the Mask Subsystem and the Mask Editor window appears as shown in Figure C.4. With the Icon tab selected as shown in Figure C.4, we position the text cursor in the Drawing commands pane, and we enter the MA... -- Figure C.4. The Mask Editor window for Example c.1 -- Figure C.5. The masked subsystem with an imported image -- We right-click on the Subsystem block in Figure C. 5, and from the drop menu we select Edit Mask. From the Mask Editor window which appears, we select the Parameters tab shown in Fig ure C.6 below.
Figure C.6. The Parameters tab for the Mask Editor window -- We select the Add tool and the Mask Editor window now appears as shown in Figure C.7. -- Figure C.7. The Mask Editor window for specifying the attributes of the masked parameters -- The Mask Editor in Figure C.7 is used to specify the attributes of the masked parameters. The Prompt column under Dialog parameters is used as a text label to describe the parameter. For our example we enter Constant a, Constant b, and Consta... -- Figure C.8. The Masked Editor with the equation constants specified -- We right-click on the masked subsystem block shown in Figure C.5, Page C-5, and in the Func tion Block Parameters dialog box we enter the values 1, -5, and 6 for the variables a, b, and c respectively, as shown in Figure C.9. -- Figure C.9. The Function Block Parameters window with the values of the constants -- With the variables defined as above, the masked subsystem implements the quadratic equation -- and the roots of this equation are and . Our model is tested for the first root as shown in Figure C.10. -- Figure C. 10. -- The Mask Editor also contains the Initialization tab that allows us to enter MATLAB commands that initialize the masked subsystem, and the Documentation tab that lets us define or modify the type description and help text for a masked subsyst... -- Figure C.11. The Initialization tab for the Mask Editor Window -- Figure C.12. The Documentation tab for the Mask Editor window. -- References and Suggestions for Further Study -- Index.

Sommario/riassunto

This text is a sequel to Introduction to Simulink, ISBN 978-0-9344239-8-2. Stateflow is an interactive graphical design tool that works with Simulink to model and simulate event-driven systems.