



PLECS

Tutorial

Introduction to the State Machine Block

Implementation of an interleaved triangular current mode (TCM) control scheme for a single-phase PFC rectifier

Tutorial Version 1.0

www.plexim.com

- ▶ Request a PLECS trial license
- ▶ Check the PLECS documentation

1 Introduction

A state machine is a powerful tool which can be used to control, model, and predict system behavior. It is also a convenient way to represent a process which evolves over time. The alternative to using a state machine is distributing specific task-handling functions under the correct loops of the code, which is inefficient and can often be confusing.

The State Machine block provides PLECS users with a graphical way of modeling a system's reaction to input signals. Actions performed during the execution of a state machine - such as enter, during, and exit - can be implemented using the C programming language. PLECS also supports hierarchical state machines, with substates nested within their superstates. Keep in mind that states that intersect each other's edges are not allowed and lead to an error when the simulation is started.

In this exercise you will learn the following:

- How to model a simple state machine in PLECS and tie it into a simulation.
- How to implement a modulator using the State Machine block.

Before you begin Ensure the files `finite_state_machine_start.plecs`, `finite_state_machine_1.plecs` and `finite_state_machine_2.plecs` are located in your working directory to compare with your own model at each stage of the exercise.



Note: This model contains model initialization commands that are accessible from:
PLECS Standalone: The menu **Simulation + Simulation Parameters... + Initializations**
PLECS Blockset: Right click in the **Simulink model window + Model Properties + Callbacks + InitFcn***

2 Modulator State Machine

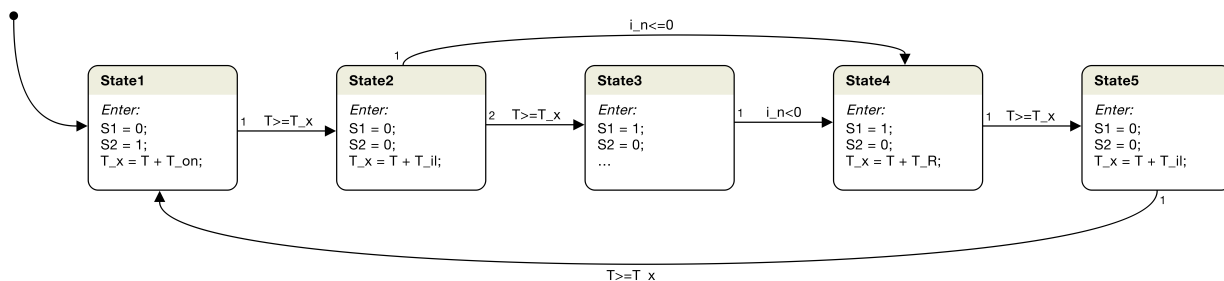


Figure 1: Modulator State Machine model.


We will build a simple modulator model shown in Fig. 1 above for the purposes of this tutorial. This state machine was originally used in an Ultraflat Interleaved Triangular Current Mode (TCM) Single-Phase PFC Rectifier to control the input of its switches, as shown in Fig. 2 below.



Your Task:

- 1 Open the file `finite_state_machine_start.plecs` - this will be your starting point for this exercise. Drag a State Machine component onto the schematic from the library browser. You can do so by navigating to the "Control" section of the library or by using the search box.
- 2 Double-click on the State Machine block to enter the editor window, which allows you to graphically model the states and transitions between them. Everything you model within this window will later be converted into a C code file that is compiled and executed during the simulation, similar to a C-Script.

2.1 Parameter specification

Let's first specify the parameters of this model by clicking on the  button toward the top of the editor window. This is where the inputs, outputs, internal constants and variables, sample time and other settings may be specified. These may then be directly used in the C code that defines the triggers, conditions and actions of the states and transitions in the state machine. The State Machine block will also automatically add and label its input and output terminals accordingly.

Mark the check box next to **Animation** label to enable the animation mode. If the animation mode is active, the state machine displays the execution flow during the simulation by highlighting the active states and transitions. The simulation is paused each time the active state changes which is useful in finding execution errors.

Lastly, this model uses the macro *CurrentTime* in its C code. This value contains the current simulation time, which is useful for the implementation of deferred transitions.




Your Task: Using the + and – buttons, add the variables, constant and C declaration listed in Table 1 below under the appropriate parameter tabs.



At this stage, your model should be the same as the reference model `finite_state_machine_1.plecs`.

2.2 Adding and editing states

To add a new state, click the tool button  and then click on the editor area where you want to place the state. The graphical representation of a state is a rounded rectangle with a header stripe where the name of the state is displayed. Double-click the name to change it. A state rectangle can be re-sized by dragging its corners with the mouse. Double-click on a state to change its parameters. The parameter dialog of a state contains a tab for each kind of action that a state can execute. The “enter action” is executed each time the execution flow of the state machine enters the state. The “during action” is executed in each time step in which the execution flow remains in the state. The “exit action” is executed each time the execution flow leaves the state. All actions can be implemented using C code.

PLECS supports hierarchical state machines. You can drag states into the rectangular border of other states to establish a hierarchy. A state that is contained by (within the border of) another state is called “substate”. The containing state is called “superstate”. States that intersect each others’ edges are not allowed and lead to an error when the simulation is started.



Your Task:

- 1 Add five states, from left to right, as shown in Fig. 1, and name them “State1” through “State5”.
- 2 Enter the parameters shown in Table 2 within the “Enter action” of these five states.

These parameter sets correspond to the transitions illustrated in Fig. 3 below.

2.3 Adding and editing transitions

Now that we have our five states, we need to first specify the starting state for the simulation by creating an initial transition.

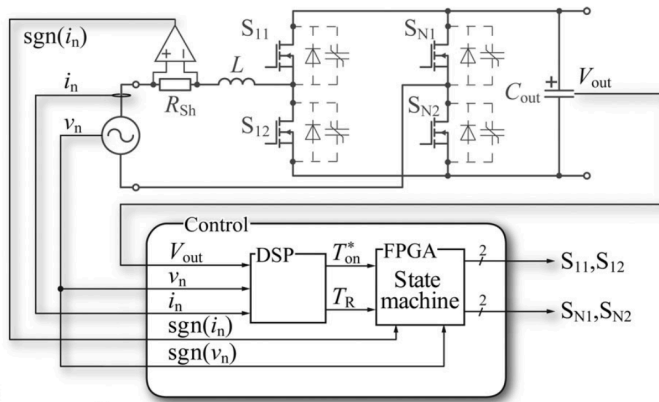


Figure 2: State machine used to control a TCM Single-Phase PFC Rectifier (Figure taken from Fig. 8.(a) in [1]).

Inputs		Explanation
i_n	continuous	Inductor current
T_on	continuous	S2 on-time
T_R	continuous	Reverse conduction time
Outputs		
S1	-	Switch 1 control signal
S2	-	Switch 2 control signal
Constants		
T_il	400e-9	Constant interlock delay
Variables		
T_x	0	Current position in time
C declarations		
#define T CurrentTime	-	Macro containing current time

Table 1: State machine parameter declaration

State1	State2	State3	State4	State5
$S1 = 0;$	$S1 = 0;$	$S1 = 1;$	$S1 = 1;$	$S1 = 0;$
$S2 = 1;$	$S2 = 0;$	$S2 = 0;$	$S2 = 0;$	$S2 = 0;$
$T_x = T + T_{on};$	$T_x = T + T_{il};$	-	$T_x = T + T_R;$	$T_x = T + T_{il};$

Table 2: “Enter action” state parameter declaration.

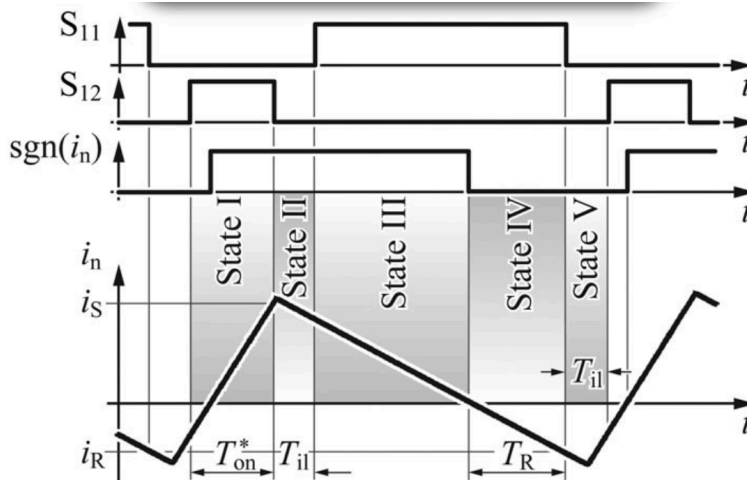




Figure 3: State machine transitions (Figure taken from Fig. 8.(b) in [1]).



Your Task: To add a new initial transition, click the tool button , click on the editor area where you want this transition to start from (within the schematic area just outside of State1), and then click on the edge of the state you want to mark as the initial state – in this case its State1. You may now move this transition arrow in various ways.

Double-clicking on an initial transition opens its parameter dialog where you may specify a set of actions, which would be executed each time the execution flow takes this transition. As with all other state and transition parameters, this can be implemented using C code expressions. In this case, you may leave this action field blank.



Note: This tutorial doesn't make use of the circular Junction component () which serves as a branching point that either joins or forks multiple transitions. Junctions are useful if the beginning state of the simulation isn't always the same one.

To add transitions between states, click on the edge of a state (when the cross cursor appears) and drag the mouse to the edge of the destination state (release the mouse button when the double-cross cursor appears). The end points of a transition can be changed later by mouse-dragging (when the hand cursor appears close to the end that is to be moved). The shape of a transition can be changed by dragging a handle. The available handles are displayed as gray dots when the mouse is hovering over the transition.



Your Task: Add transitions as shown in Fig. 1 above and Table 3 below.

State1 -> State2	State4 -> State5
State2 -> State3	State5 -> State1
State3 -> State4	State2 -> State4

Table 3: State interconnections.

Double-click on a transition to open its parameter dialog. The **Priority** parameter is a number between 1 and the number of outgoing transitions from the source state of the transition. When triggers and conditions (see below) of multiple outgoing transitions evaluate to true, the execution flow will take the transition with the smallest priority parameter.

Triggers and conditions are evaluated in each time step. If both the trigger and the condition of a transition evaluate to true, the transition is taken. PLECS distinguishes between explicit (external input signal), time-based (executed exactly delay seconds after the source state of the transition was entered), and implicit triggers (relational expression which is active when the expression becomes true). Conditions, on the other hand, are boolean expressions that are simply evaluated to check whether the transition should be taken. Notice the fundamental difference between an implicit trigger $x > 0$ and a condition $[x > 0]$. The former will evaluate only at the moment when the value of x becomes greater than zero, whereas the latter will be tried whenever x is greater than zero. Both triggers and conditions are entered as C code expressions (no semicolon at the end).

The “Action” of a transition is executed each time the transition is taken, after any state exit actions and before any state enter actions. The action can be implemented using C code. PLECS supports internal transitions, i.e. transitions that leave their source state edge towards the inside of the state and end at the edge of a substate (see above for more information about hierarchical states). When an internal transition of state A to substate B (or from state B to superstate A) is taken, the exit and enter actions of state A are not executed.



Your Task:

- 1 Specify the expressions shown in Table 4 as triggers of the respective transitions between states. You may leave the action fields as blank for all transitions.
- 2 Go back to the schematic window with the converter model and make sure that all inputs (i_n , T_{on} , and T_R) and outputs (S1 and S2) of the State Machine block are connected to the respective ports of the Ctrl and Gate blocks.
- 3 Double-click on the state machine to go back to the schematic editor and simulate the model. If you checked the **Animation** checkbox within the state machine’s preferences, the current simulation state lights up. Cycle through the states using the space bar and watch the state machine, as well as the output of the scope (you may have to zoom in). You may also then end the simulation by pressing Ctrl+T (or Command+T on Mac), unchecking the **Animation** checkbox and re-running the simulation to see what the end result looks like.




At this stage, your model should be the same as the reference model, `finite_state_machine_2.plecs`.

Transition	Trigger
State1 -> State2	$T \geq T_x$
State2 -> State3	$T \geq T_x$
State3 -> State4	$i_n < 0$
State4 -> State5	$T \geq T_x$
State5 -> State1	$T \geq T_x$
State2 -> State4	$i_n < 0$

Table 4: Triggers of transitions between states.

3 Conclusion

This exercise has demonstrated a step-by-step approach for creating a state machine using PLECS. You have learned how to create states and transitions between them, as well as how to operate their parameters. You have also learned how to tie a State Machine block into a larger PLECS simulation. The graphical editor makes modeling of a complex control algorithm very intuitive, as well as useful for identifying corner cases. For more information, please click on the  button within the State Machine block, or see the appropriate section of the PLECS User Manual found on our website.

References

- [1] Christoph Marxgut, Florian Krismer, Dominik Bortis, Johann W. Kolar, “Ultraflat Interleaved Triangular Current Mode (TCM) Single-Phase PFC Rectifier”, *IEEE Transactions on Power Electronics*, Vol. 29, No. 2, pp. 873-882, Feb. 2014.

Revision History:

Tutorial Version 1.0 First release

How to Contact Plexim:

☎	+41 44 533 51 00	Phone
	+41 44 533 51 01	Fax
✉	Plexim GmbH Technoparkstrasse 1 8005 Zurich Switzerland	Mail
@	info@plexim.com	Email
	http://www.plexim.com	Web

PLECS Tutorial

© 2002–2022 by Plexim GmbH

The software PLECS described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from Plexim GmbH.

PLECS is a registered trademark of Plexim GmbH. MATLAB, Simulink and Simulink Coder are registered trademarks of The MathWorks, Inc. Other product or brand names are trademarks or registered trademarks of their respective holders.