



PLECS

Tutorial

Introduction to PLECS Spice

Start learning how to incorporate SPICE netlists into your PLECS models

Tutorial Version 1.1

www.plexim.com

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

1 Introduction

PLECS can achieve fast and efficient simulation of complex power electronic systems by using ideal switches. These models enable short simulation times and robust numerical behavior, but they do generally capture less details of physical effects such as component parasitics.

When a higher level of detail is required, for example to analyze the impact of component parasitics, SPICE models can be used. These models aim to represent the physical behavior of components more accurately, but at the cost of significantly increased simulation time and computational effort.

In this tutorial, you will learn the fundamental capabilities and workflow of PLECS Spice. You will extend a typical idealized PLECS model of a flyback converter with SPICE-based components. Step-by-step you will integrate PLECS Spice features into the idealized design.

Learning objectives:

- How to import an existing SPICE netlist into a PLECS schematic
- How to write a simple SPICE netlists based on datasheet parameters
- How to take advantage of the Model Reference block for SPICE components
- How to make your schematic PLECS Spice compliant
- How to define component probes
- How to create configurable subsystems to switch seamlessly between PLECS and SPICE models

Before you begin

- 1 Download the .zip-file of tutorial “PLECS: Introduction to PLECS Spice” from the [Plexim tutorial web page](#) and extract it to your local drive.
- 2 The folder `input_data` contains SPICE netlists (.lib-files) and component datasheets for your reference.
- 3 The files `flyback_converter_1...5.plecs` represent intermediate tutorial states and serve as reference models.

This tutorial starts with the file `flyback_converter_0.plecs` which contains a schematic of a flyback converter. The schematic is reported in Fig. 1 for reference.

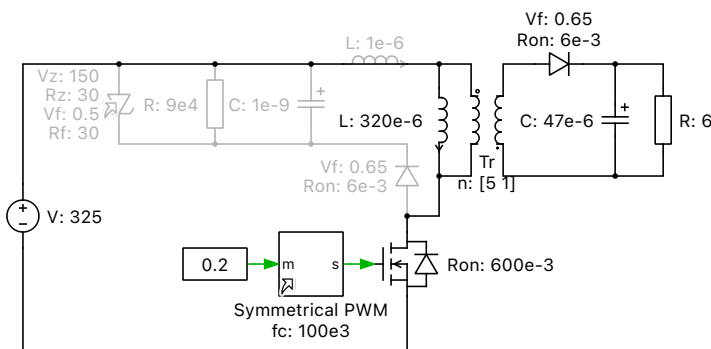


Figure 1: Flyback converter modelled in PLECS (tutorial starting point)



Note: Some components are deliberately commented out and will be enabled in a later stage of the tutorial.



Your Task: Open the PLECS model `flyback_converter_0.plecs`, run the simulation by clicking on **Simulation + Start**, and verify that the simulation runs as expected. Inspect the simulation results in the Scopes.

2 Add a SPICE Netlist for the MOSFET

In this stage of the tutorial, you will create an alternative description of the MOSFET using the SPICE netlist of a specific component.



Your Task:

- 1 **Right-click** on the **MOSFET with Diode** component and select **Create subsystem** or press **Ctrl+G**. See Fig. 2.

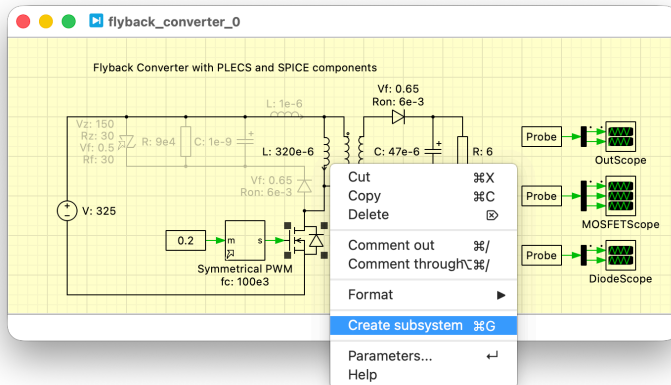


Figure 2: Create subsystem

- 2 **Right-click** on the newly created subsystem and select **Subsystem + Convert to configurable subsystem**. *Optional:* Rename the configurable subsystem by **Double-clicking** the *Sub* label and entering a new name, e.g., *ConfigMOSFET*. To hide the component name, press **Ctrl+Shift+N** or **Right-click** the component and select **Format + Show name**.
- 3 Open the configurable subsystem (press **Ctrl+U** or **Right-click + Subsystem + Open subsystem**).
- 4 Rename the existing configuration by **Double-clicking** on *Configuration 1*. Enter a meaningful name, e.g., *PLECS*.
- 5 Add a new configuration after the existing one (**Right-click** on *PLECS* configuration, **Insert configuration after**) and rename the new configuration, for example, to *SPICE* (see Fig. 3).

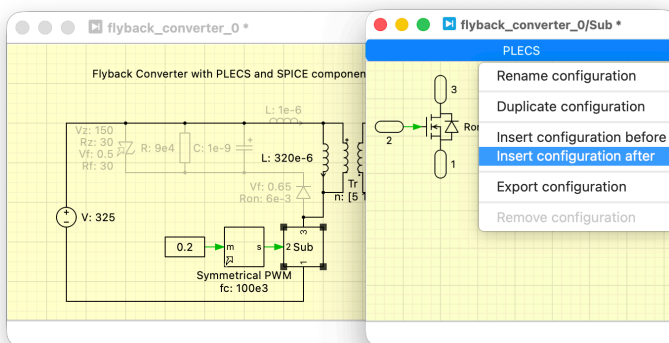


Figure 3: Insert configuration after

- 6 Now this second configuration is still empty. Place a **Netlist** component from the SPICE section of the component library.
- 7 **Double-click** on the **Netlist** component to open the netlist window as shown in Fig. 4. **Click on Import Netlist**, select the desired netlist library file, e.g., Infineon_CoolMOS_C3_600V_Spice.lib.

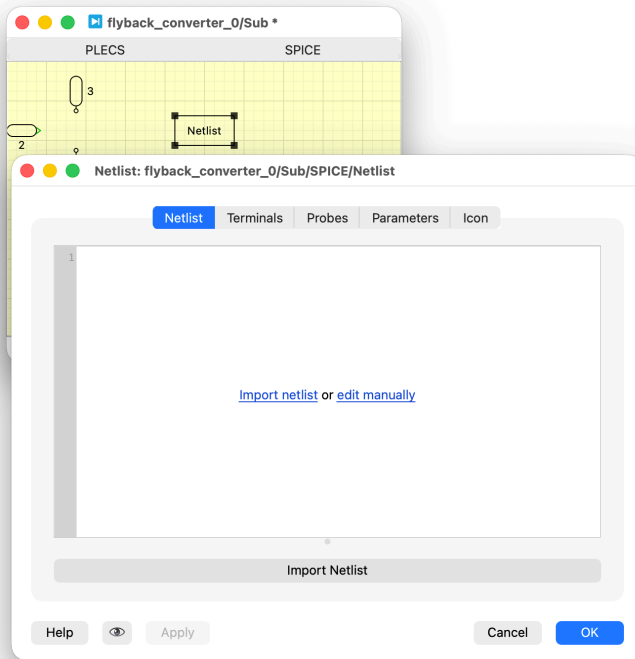


Figure 4: Netlist window

- 8 You should see a window similar to the one shown in Fig. 5. In the lower section, a drop-down menu allows you to select the desired subcircuit from the ones available in the netlist library. From the drop-down menu select the **IPD60R600CMS_L0** component. The selected subcircuit and all the dependencies are now imported and embedded into the Netlist component.

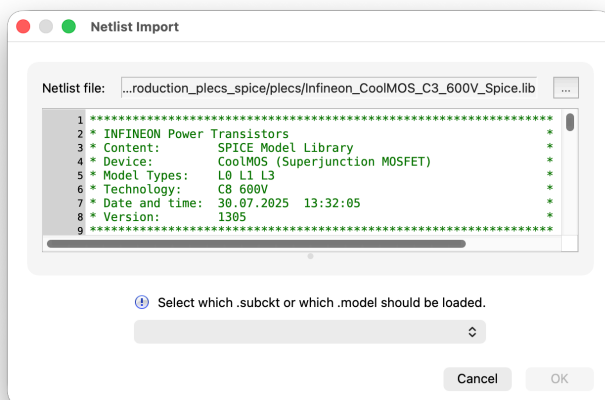


Figure 5: Netlist import wizard

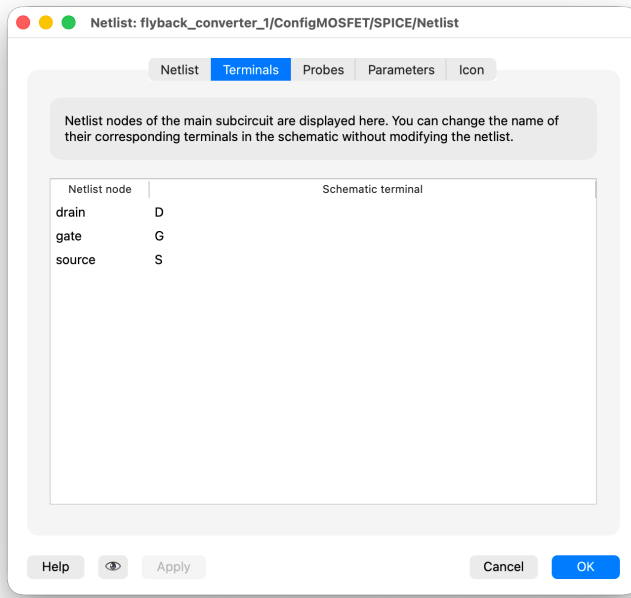


Figure 6: Netlist terminals tab

- 9 From the **Terminals** tab of the netlist window, it is possible to assign custom names to each schematic terminal. You can assign the custom terminal names *D*, *G*, and *S* to the netlist nodes *drain*, *gate*, and *source*, respectively, as shown in Fig. 6.

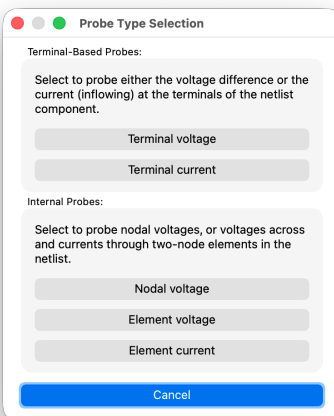


Figure 7: Netlist probes selection window

- 10 From the **Probe** tab of the netlist window, it is possible to add probe signals to the netlist component. Clicking on the “+” icon opens the menu shown in Fig. 7.
- Add a **Terminal voltage** probe between terminal *D* and *S* to probe the MOSFET channel voltage. Enter a signal name, e.g.: *Vds*.
 - Add a **Terminal current** probe on terminal *D* to probe the current entering in the drain terminal of the MOSFET. Enter a signal name, e.g.: *Id*.
 - Add a **Terminal voltage** probe between terminal *G* and *S* to probe the MOSFET gate voltage.

Enter a signal name, e.g.: V_{gs} .

11 To complete the modelling of the MOSFET and make it PLECS Spice capable, you should include a gate driving circuit. The simplest gate driver can be realized with just four components:

- A **Rate Limiter** to smooth the square pulses from the control signal and model a finite rise time. In this example, set the rising and falling slew rate to $1\text{V}/1\text{ps}$ (i.e., $1\text{e}12\text{V/s}$). This value approximates an almost ideal transition.
- A **Gain** to scale the logical control signal to the appropriate gate voltage level. By setting the gain to 10, the input signal ranging from 0 to 1 is converted into a gate voltage ranging from 0 to 10 V.
- A **Controlled voltage source** to convert the signal into a voltage.
- A **Resistor** to model the gate driving resistance. This resistance controls the charging and discharging speed of the MOSFET gate capacitances and therefore influences the switching speed, switching losses, and ringing behavior. In this example, set the resistance to $6\ \Omega$.

12 Optional: Rename the subsystem **Signal Input** and **Electrical ports D, G, and S** according with the schematic in Fig. 8.

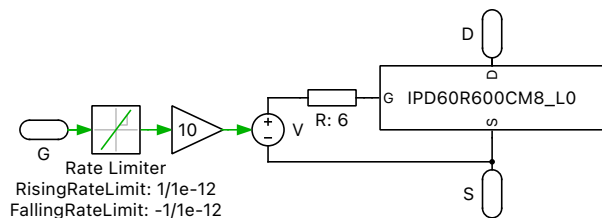


Figure 8: Schematic of the SPICE MOSFET configuration



At this stage the schematic of the SPICE configuration of the MOSFET should be similar to the one shown in Fig. 8.

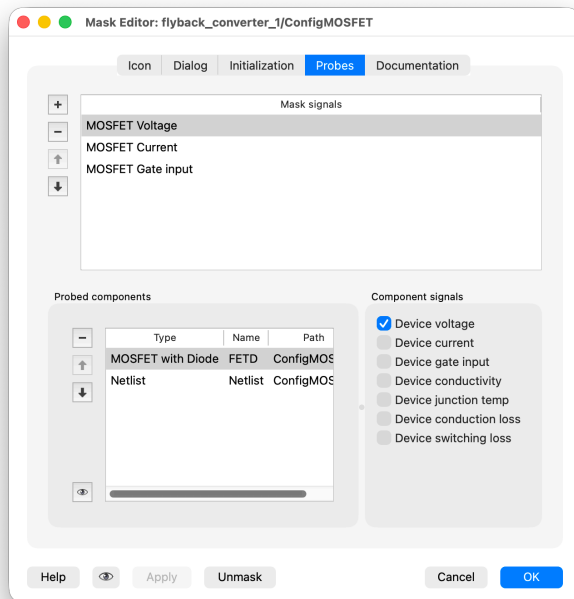


Figure 9: Configurable MOSFET subsystem: probes tab

13 The probe signals defined in step 10 are used to access the SPICE circuit signals from the outer PLECS schematic. Therefore, these signals must be passed to the configurable subsystem. To achieve this, create a mask for the subsystem and define the probe signals as follows:

- Select the configurable subsystem and, from the contextual menu, select **Subsystem + Create Mask...** or press **Ctrl+M**.
- In the **Probes** tab of the mask window add a new **Mask signal** and name it with a meaningful name, e.g., *MOSFET Voltage*.
- In the **Probed components** section, **drag and drop** both the MOSFET with Diode component from the PLECS configuration and the Netlist component from the SPICE configuration of the MOSFET component and assign the probing signals accordingly.
- Repeat this procedure for each signal that needs to be probed. As a result, the **Probes** tab of the mask should appear similar to Fig. 9.
- *Attention:* To enable seamless switching between the PLECS and SPICE configuration of the MOSFET, assign the same number of probe signals to both configurations.
- *Optional:* In the **Icon** tab of the mask window, an icon can be drawn using Lua syntax and the `PlecsIconLib` library. The documentation is available [online](#). To draw the MOSFET as shown in Fig. 11, use the following lines of code:

```
local IconLib = require('PlecsIconLib')
IconLib.mosfetDiode(0, 0)
```

14 In case the icon does not align with the terminal of the masked subsystem:

- If you define a mask icon for a Subsystem block, PLECS automatically protects the block and the ports of the underlying schematic. You can no longer resize the Subsystem block or change the positions of the terminals on the subsystem frame.
- If you want to change a masked Subsystem block, you can unprotect it by **right-clicking** on the subsystem and select **Subsystem + Unprotect**.
- Once the model is in unprotected state you can hold **Shift + Right Mouse Button** to drag each terminal in the correct position marked by the mask icon.
- With the terminals in the right position with respect to the mask, it is a good time to resize the subsystem to fit with the mask itself.
- You can later protect it again by choosing **Protect** from the same menu.
- You can hide the component frame by deselecting the `Show frame` and hide the terminal labels by selecting the `Hide terminal labels` options in the **Icon** tab of the mask window.

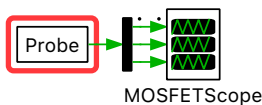


Figure 10: Probe component connected to the “MOSFETScope” scope.

15 The probes defined in the masked subsystem in step 13 are used to access signals from either the PLECS or the SPICE configuration, for example to display them in a scope. In the Probe component connected to the “MOSFETScope” (See Fig. 10), remove the MOSFET with Diode component by clicking on the “-” icon, then drag and drop the newly created masked subsystem into the **Probed components** window and select all the signals in **Component signals**.

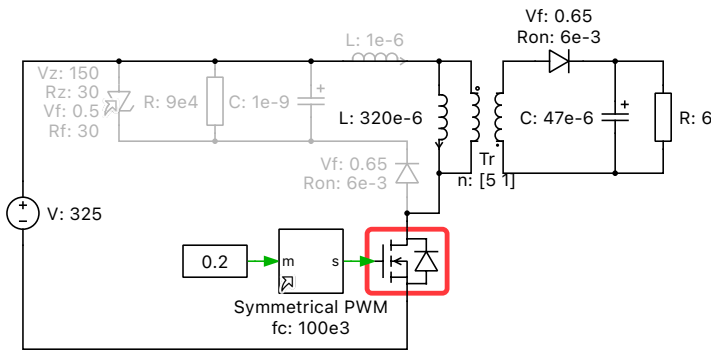




Figure 11: Flyback converter with a configurable MOSFET (tutorial stage 1)

 The original MOSFET with Diode component has been replaced by a configurable subsystem. This subsystem enables switching between the PLECS-based and SPICE-based MOSFET models within the same schematic. Your model should match the one shown in Fig. 11, which is available in the file `flyback_converter_1.plecs`.

 **Note:** The model is not yet ready for SPICE simulation because other semiconductors in the same electrical circuit are still implemented using idealized PLECS models only. Electrical connections between PLECS and SPICE components within the same circuit are not allowed and will cause the simulation to fail.

3 Create a Custom SPICE Netlist for a Diode

In this stage of the tutorial, you will learn how to create a SPICE-based description for a diode by writing your own netlist based on datasheet parameters. You will create a SPICE netlist for the diode using the datasheet provided in the tutorial materials (`STPS20120C.pdf`).

Your Task:

- 1 Repeat steps 1 - 6 of Section 2 for the secondary side diode of the flyback converter, highlighted in Fig. 12.

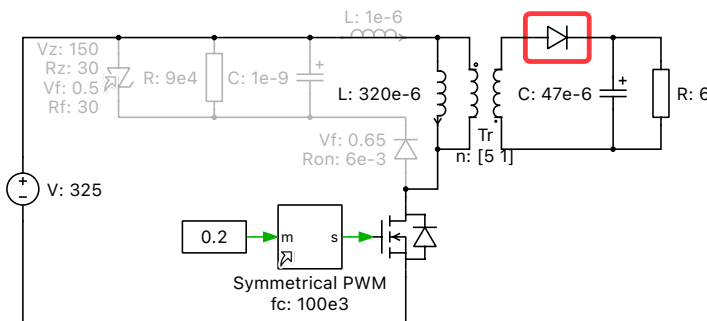


Figure 12: Flyback converter with a configurable diode (tutorial stage 2)

- 2 On the **Netlist window** click on **edit manually** to write your own netlist.
- 3 The minimal syntax to define a SPICE subcircuit appears.

- 4** The first line defines the name of the subcircuit and its terminals, which are also nodes of the circuit.

```
.subckt CUSTOM_DIODE 1 2
```

defines a subcircuit called *CUSTOM_DIODE* with netlist terminal *1* and *2*.

- 5** In the second line, define the type of component:

```
d1 1 2 diode
```

a diode *d1*, connected between nodes *1* and *2* with a custom diode model called *diode*.

- 6** The third line specifies the diode model with the SPICE statement *D*:

```
.model diode D(N=1.4 VJ=.6 RS=7.5E-3 IS=350E-9 CJO=550E-12)
```

where the basic guidelines to extract those parameters from the datasheet are reported in Section 8.

- 7** From the **Terminals** tab of the netlist window, it is possible to assign custom names to each schematic terminal, in this case the nodes *1* and *2* can be renamed as *A* and *C*, representing the anode and cathode, respectively.

- 8** From the **Probe** tab of the netlist window, it is possible to add probe signals to the netlist component. **Clicking** on the “+” icon opens the menu shown in Fig. 7.

- Add a **Terminal voltage** probe between terminals *A* and *C* to probe the diode voltage. Enter a signal name, e.g.: *Vf*.
- Add a **Terminal current** probe on terminal *A* to probe the current at the diode anode terminal. Enter a signal name, e.g.: *If*.

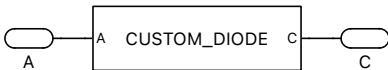


Figure 13: Schematic of the SPICE diode configuration

 At this stage, the schematic of the SPICE configuration should match the one in Fig. 13.

- 9** The probe signals defined in step 8 are used to access the SPICE circuit signals from the outer PLECS schematic. Refer to step 13 in Section 2 for instructions on configuring the mask of the diode configurable subsystem. The **Probes** tab of the mask should appear similar to Fig. 14.

- *Optional*: To draw the Diode as shown in Fig. 12, use the following lines of code in the **Icon** tab:

```
local IconLib = require('PlecsIconLib')
IconLib.diode(0, 0)
```

- 10** *Optional*: Rename the subsystem **Electrical ports** *A* and *C* according with the schematic in Fig. 13. In case the icon does not align with the connections to the masked subsystem, follow the **Note** in step 13 of Section 2.

- 11** In the Probe component connected to the “DiodeScope”, remove the Diode component by clicking on the “-” icon, then drag and drop the newly created masked subsystem into the **Probed components** window and select all the signals in **Component signals**.

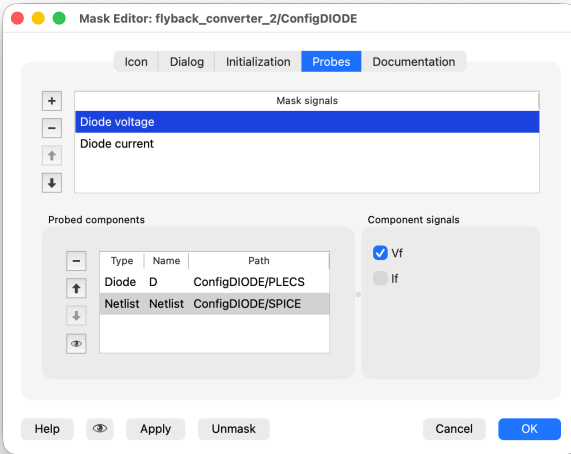



Figure 14: Configurable diode subsystem: probes tab


 The original diode component has been replaced by a configurable subsystem. This subsystem enables switching between the PLECS-based and SPICE-based diode models within the same schematic. Your model should match the one shown in Fig. 12, which is available in the file `flyback_converter_2.plecs`.

The model is not yet ready for SPICE simulation. Refer to the **Note** on page 7 at the end of Section 2 for additional details.

4 Run your First PLECS Spice Simulation

The model is almost ready to be simulated with PLECS Spice. However, few last touches are required.

4.1 Add Electrical Ground Reference Ports

 **Your Task:** Add an **Electrical Ground** component to both the primary and the secondary side of the flyback converter.

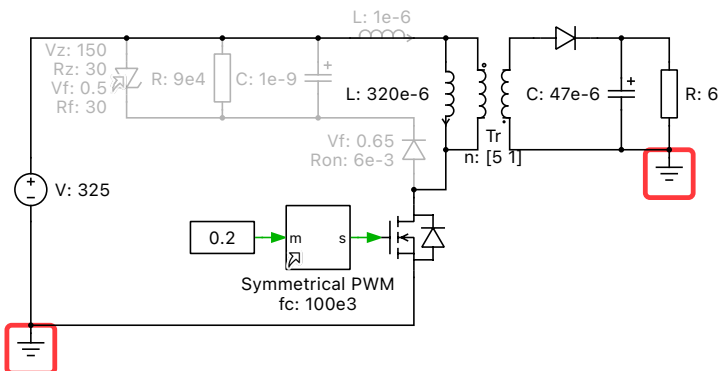


Figure 15: Flyback converter PLECS and PLECS Spice compatible (tutorial stage 3)



Your model should now match the one shown in Fig. 15, which is available in the file `flyback_converter_3.plecs`. It is now possible to simulate the converter with PLECS Spice by selecting the SPICE configuration for all subsystems.



Note: PLECS and SPICE models cannot be mixed within a single connected electrical network. Therefore, the simulation will succeed only if all electrically connected semiconductor components use the same implementation, i.e., either all PLECS or all SPICE. In addition, the transformer does not isolate the PLECS and SPICE electrical domains, since both sides remain part of the same electrical network from the solver perspective.

4.2 Use a Variable to Control the Configurations

Manually opening each configurable subsystem to switch between the PLECS and SPICE configurations is cumbersome. Instead, you can initialize a variable at the simulation startup. By setting this single variable, all configurable subsystems will switch configuration accordingly.



Your Task:

- 1 Open the **Simulation + Simulation Parameters** menu. In the **Initialization commands** tab, define a variable used to switch between configurations. For example, insert the following code:

```
PLECS = 1;
SPICE = 2;
simMode = PLECS;
```

This variable, e.g., `simMode`, can assume the value `1` or `2` and is used to select the first configuration, i.e., *PLECS*, or the second configuration, i.e., *SPICE*, of the configurable subsystems.

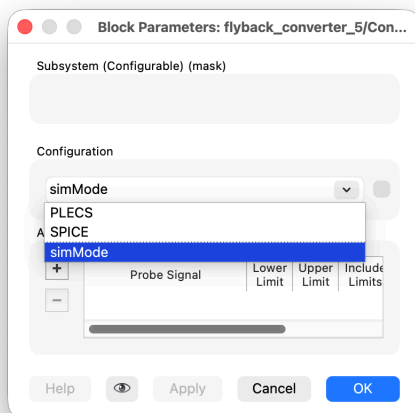


Figure 16: Variable used to control the configuration of configurable subsystems

- 2 Open each configurable subsystem created in this tutorial, by **double-clicking** on it. In the **Configuration** drop-down menu, replace the `<reference>` placeholder with the name of the variable defined in the **Initialization commands**, e.g., `simMode`, as in Fig. 16.



Note: It is important to keep a consistent order of the different configurations across all configurable subsystems. This ensures that the configuration selection variable, e.g., `simMode`, activates the same implementation (PLECS or SPICE) for every subsystem. Otherwise, PLECS and SPICE configurations would be connected within the same electrical circuit, which is not allowed and will cause the simulation to fail.



Note: Another option would be to define the variable `simMode` as a string, e.g. “PLECS” or “SPICE”, to select the corresponding subsystem configuration. In this case the order of the configurations is not important. However, it is crucial to keep consistency in the wording between the configurations of all the subsystems to select the correct component modelling.



With this final adjustment, the model can switch between PLECS and PLECS Spice simulation by simply changing the value of a variable in **Simulation + Simulation Parameters + Initialization commands**. You completed successfully the conversion of a PLECS model into a PLECS Spice compliant circuit. The simulation can now be run and used to compare PLECS and PLECS Spice results in the following tutorial stage.

4.3 Results

In this part of the tutorial, you will compare the simulation results obtained with the idealized PLECS configurations to those obtained with the SPICE netlist configurations.



Your Task:

- 1 Set the variable, e.g., `simMode`, to configure all the configurable devices with PLECS models.
- 2 Run the simulation, which will be executed with PLECS models.
- 3 Hold the traces in the scopes by clicking the hold button in the scope toolbar, shown in Fig. 17. If the toolbar is not visible in the scope window, enable it from **View + Toolbar**.

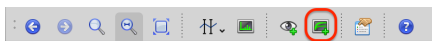


Figure 17: Hold trace symbol in scope toolbar

- 4 Change the value of the variable, `simMode`, in the initialization commands, to configure all the configurable device with SPICE netlist-based models.
- 5 Run the simulation, which now will be executed with SPICE models.

The results should resemble those shown in Fig. 18 and Fig. 19, which report the output voltage, magnetizing inductance current, and MOSFET voltages and current.



Note: The bottom plot of Fig. 19 shows the gate command of the MOSFET. In the PLECS model, this signal is a logical control input assuming values of 0 or 1. In contrast, in the SPICE model it corresponds to the physical gate-to-source voltage, which reflects the actual device V_{GS} .

The additional details provided by the SPICE models are essential to capture the effects introduced by the devices parasitics. One of these effects is the low-frequency oscillation of V_{DS} which is a consequence of the energy exchange between the magnetizing inductance and the MOSFET output capacitance.

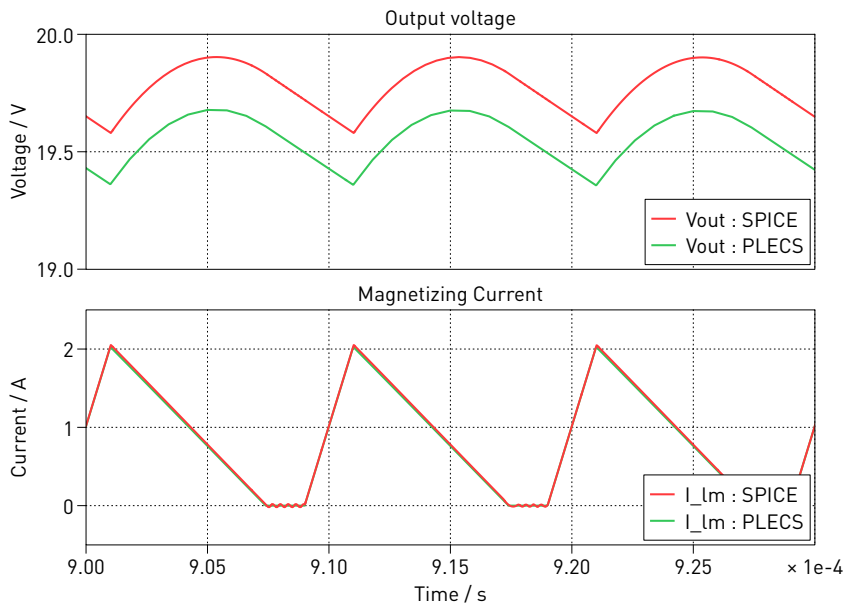


Figure 18: Output voltage and magnetizing inductor current of PLECS (green) and PLECS Spice (red) simulations

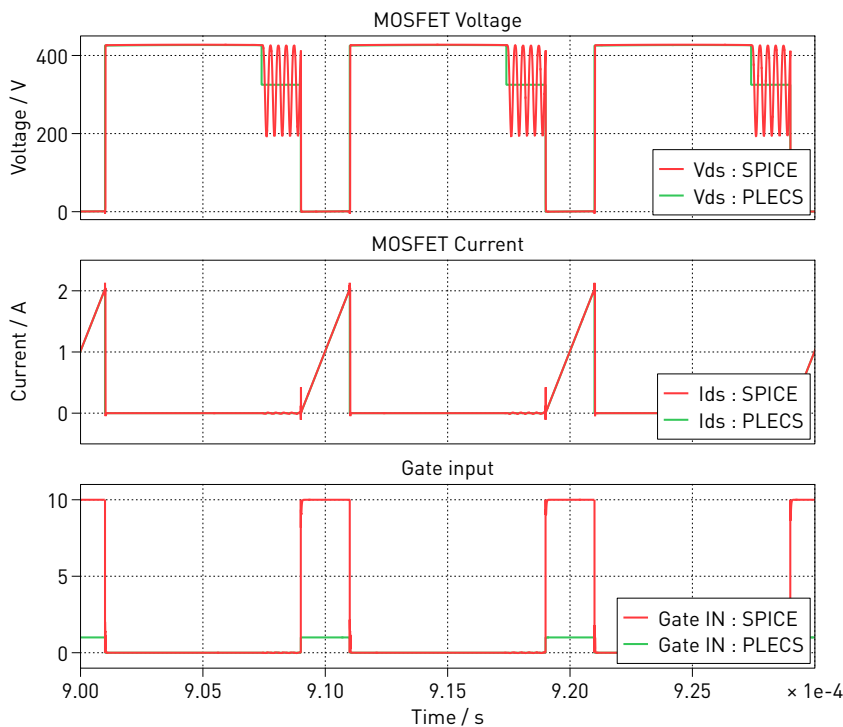


Figure 19: MOSFET voltages and current of PLECS (green) and PLECS Spice (red) simulations

5 Transformer Leakage and Snubber Circuit

In reality the primary and secondary side of a flyback transformer will not be perfectly coupled. Thus, to refine the analysis of your circuit you can include the leakage inductance in the circuit model.

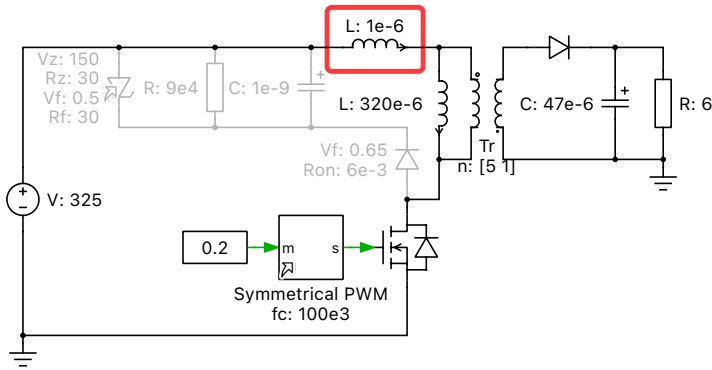


Figure 20: Flyback converter with transformer leakage inductance



Your Task:

- 1 Uncomment the leakage inductor as shown in Fig. 20.
- 2 Run a PLECS Spice simulation and verify that the results are as expected.
- 3 Set the model back to ideal components and run a PLECS simulation.



Why does PLECS abort with an error message? (See the conduction state of the MOSFET.)



When the MOSFET opens, there is no longer a path for the magnetizing inductance current to flow, thus forcing it to instantaneously jump to zero. This isn't possible in PLECS. Hence, the “*State discontinuity after switching.*” error is thrown.



Note: If PLECS aborts with such an error message, there is usually something wrong with the circuit design.

In reality, the MOSFET in off-state represents a very large resistance rather than an open circuit. This effect is modeled within the SPICE netlist of the MOSFET. Thus, the simulation with PLECS Spice can run without issues. In PLECS, this effect can be modelled with an additional resistor, e.g. $R_{\text{off}} = 10 \text{ k}\Omega$, connected in parallel to the ideal MOSFET. If you look at the voltage across the MOSFET, you will observe a very high voltage when the MOSFET is turned off. This overvoltage is caused by the energy stored in the leakage inductor that is drained through the MOSFET channel and could potentially destroy the component itself. A simple snubber design is proposed here in order to mitigate the overvoltage across the MOSFET caused by the energy in the transformer leakage as represented in Fig. 21.

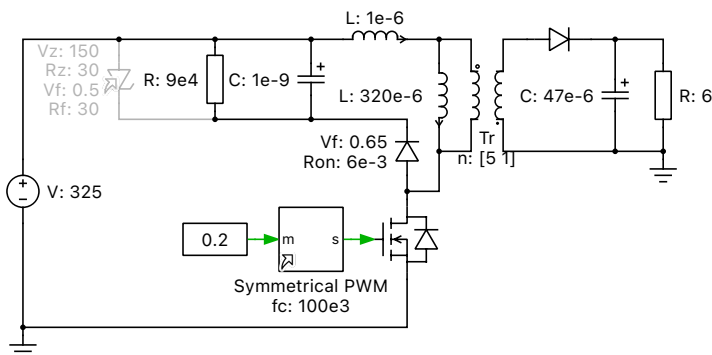


Figure 21: Flyback converter with simple snubber circuit



Your Task: Uncomment the **Resistor**, **Capacitor**, and **Diode** component at the primary side of the flyback to include the snubber circuit.



Note: Now this circuit can be executed with PLECS revealing the overvoltage spikes on the MOSFET channel voltage due to the energy in the transformer leakage. However, the newly included diode has only the ideal modelling which is only compatible with PLECS. Therefore, a SPICE simulation of the circuit as it is, will throw an error.

5.1 Replace the Snubber Diode with a Model Reference Component

Repetitive parts of a circuit can be implemented using the **Model Reference** component. In this stage of the tutorial you will use the **Model Reference** component to replace the diode on the primary side of the flyback converter and reference to the configurable subsystem implemented in Section 3.



Your Task:

- 1 Place a **Model Reference** block from the PLECS component library.
- 2 **Double-click** on the block to open the dialog window.
- 3 **Drag and drop** the configurable subsystem implemented in Section 3 for the secondary-side diode.
- 4 A **model reference** has now been created, indicated by the small black arrow in the bottom-left corner of the component (see Fig. 15). Any edits made to the original component are automatically applied to the referenced subsystem.
- 5 The original PLECS diode can now be replaced with **Model Reference** component.

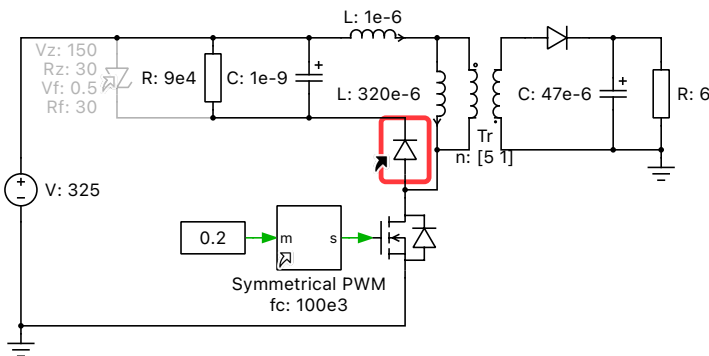


Figure 22: Flyback converter with a model referenced diode (tutorial stage 4)



The original snubber diode has been replaced by a reference to a configurable subsystem. Your model should match the one shown in Fig. 22, which is available in `flyback_converter_4.plecs`.

6 Snubber circuit with Zener Diode

A Zener diode can be included to clamp the overvoltage across the MOSFET. This is especially helpful during the converter startup.



Your Task: Uncomment the Zener diode as shown in Fig. 23.

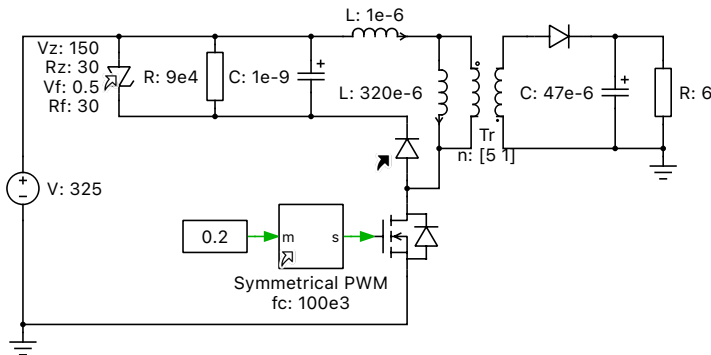


Figure 23: Flyback converter with a model referenced diode



Note: As experienced in Section 5 this circuit can only be executed with PLECS because of the ideal modelling of the newly introduced zener diode. Therefore, a SPICE simulation of the circuit as it is, will throw an error.

6.1 Add a SPICE Netlist for the Zener Diode

At this stage, the workflow for converting a PLECS component into a PLECS Spice-capable subsystem should be clear.



Your Task: The next stage is to convert the PLECS Zener diode. Repeat the procedure described in the previous sections, with the following differences:

- 1 The netlist of the Zener diode, i.e., *SMAJ150A*, is available in the file *SMAJ1_series_Spice.lib*.
- 2 *Optional:* Define suitable probing signals. However, this is not necessary to complete the tutorial.
- 3 *Optional:* Draw the Zener diode as shown in Fig. 23 using the following lines of code in the **Icon** tab:

```
Icon:line({-6, -6, 6, -6}, {-7, 7, 0, -7})
Icon:line({3, 6, 6, 9}, {-9, -7, 7, 9})
Icon:line({-10, -6}, {0, 0})
Icon:line({10, 6}, {0, 0})
Icon:line({10, 6}, {0, 0})
```

In case the icon does not align with the connections to the masked subsystem, follow the **Note** in step **13** of Section 2.



At this stage, all semiconductor components have been replaced by configurable subsystems. These subsystems enable switching between the PLECS-based and SPICE-based semiconductor models within the same schematic. Your model should match the one shown in Fig. 24, which is available in the file *flyback_converter_5.plecs*.

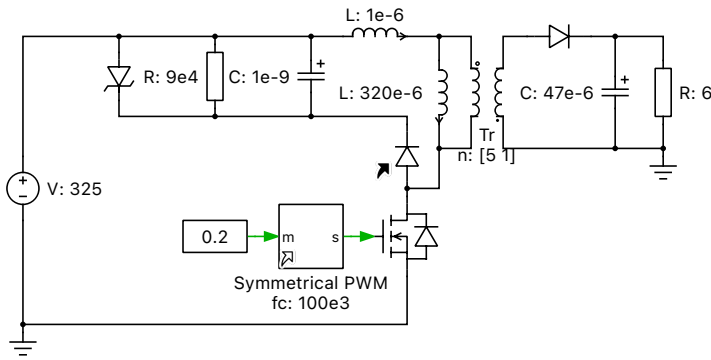


Figure 24: Flyback converter with a configurable zener diode (final tutorial stage)



Note: Although all semiconductor components have been replaced by configurable subsystems that support both PLECS and SPICE implementations, the model is not yet ready for SPICE simulation or seamless switching between the two. Further adjustments to the model configuration are required.

7 Final Results



Your Task: In the final part of the tutorial, you will compare the simulation results of the flyback converter including the transformer leakage inductance and the snubber circuit. To do so repeat the steps explained in Section 4.3.

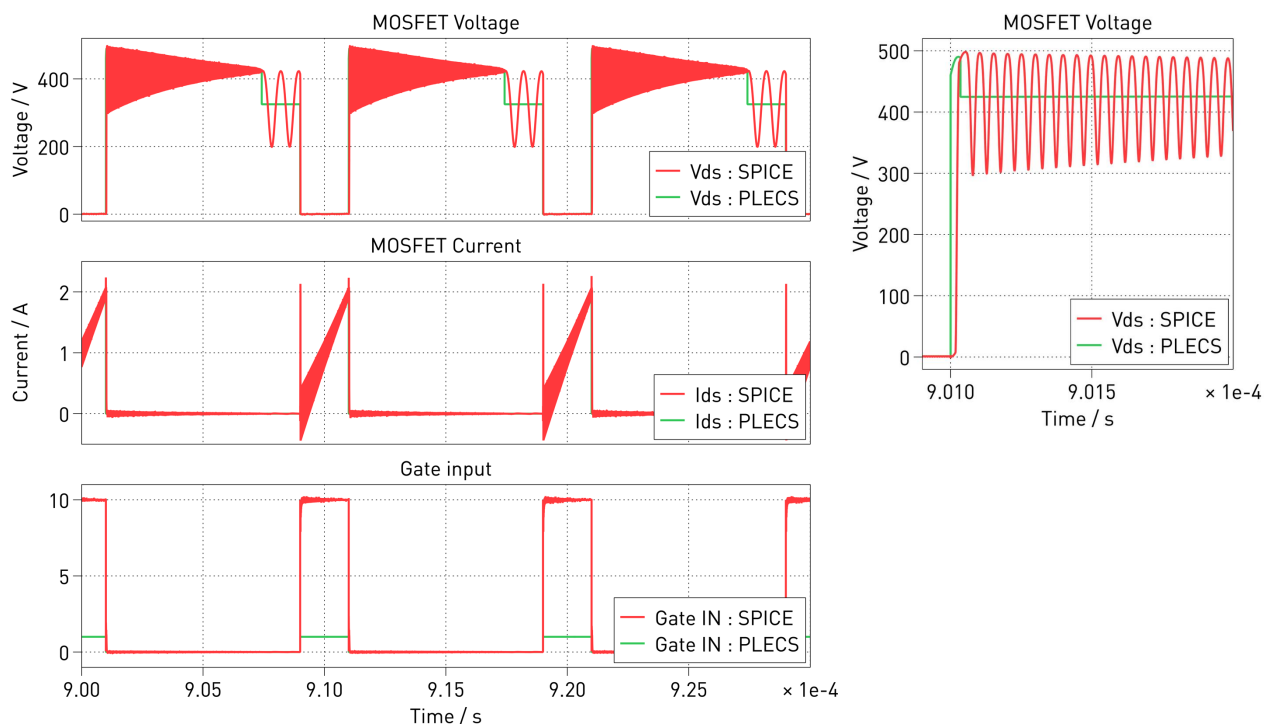


Figure 25: MOSFET voltages and current of PLECS (green) and PLECS Spice (red) simulations

The results should resemble those shown in Fig. 25, which reports the MOSFET voltages and currents. Because the converter is modeled in all its parts using configurable components which combine both ideal PLECS models and a netlist-based SPICE approach, it is possible to analyze the system at multiple levels of abstraction, enabling a step-by-step understanding of the behavior and interactions among its components.

With the inclusion of transformer leakage inductance and the snubber circuit, the V_{DS} waveform exhibits additional dynamic features that are only captured by SPICE-based models. Compared to the previous case, this behavior is now influenced by the presence of the transformer leakage inductance and the snubber network.

The leakage inductance introduces an additional energy storage element, which modifies the effective resonant network seen at turn-off. At the same time, the RC and Zener snubber provide a dissipative and clamping path, altering the oscillation frequency.

Superimposed on this behavior, high-frequency ringing caused by leakage inductance and parasitic capacitances is also visible in the V_{DS} waveform. A detailed analysis of this high-frequency behavior is beyond the scope of this tutorial.

8 Appendix: Basic Diode Netlist Parameter Extraction from Datasheet



Note: This explanation is not intended to be an exhaustive guide on extracting diode netlist parameters from datasheet values. However, it is sufficient for the purpose of this tutorial and provides a starting point for simulations with PLECS Spice. A full specification of all the parameters of the diode statement D is available in the official [PLECS documentation](#).

In Section 3 a diode model has been specified with the following SPICE statement D :

```
.model diode D(N=1.4 VJ=.6 RS=7.5E-3 IS=350E-9 CJ0=550E-12)
```

where:

N is the emission coefficient, which controls the curvature of the exponential. Generally, 1.4 is a good approximation for Schottky diodes.

VJ is the junction potential, which shifts the IV curve. A safe assumption for Schottky diodes is 0.6 V.

IS is the diode saturation current, which sets the forward voltage. The Shockley diode equation states:

$$I(V_j) = I_S \left(e^{\frac{V_j}{nV_T}} - 1 \right) \quad (1)$$

where n has been defined above and $V_T = 26$ mV is the thermal voltage at ambient temperature. It is possible to isolate the saturation current under the deep conduction approximation:

$$I_S = I(V_j) e^{-\frac{V_j}{nV_T}} \quad (2)$$

From the datasheet, it is possible to extract a point (V_F, I_F) from the IV characteristic. Use the extracted values in the equation to derive I_S :

$$I_S = I_F \cdot e^{-\frac{V_F - R_S \cdot I_F}{nV_T}} \quad (3)$$

where R_S is defined below in Eq. (5). Note that the datasheet reports the forward voltage as $V_F = V_j + R_S \cdot I(V_j) = V_j + R_S \cdot I_F$.

RS is the diode series resistance, which dominates the diode behavior at high-voltage/high-current conditions. It is sometimes specified explicitly in the datasheet. If not, two points can be selected from the linear region of the IV characteristic and used in the following equation:

$$I_S = I_1 e^{-\frac{V_1 - R_S \cdot I_1}{nV_T}} = I_2 e^{-\frac{V_2 - R_S \cdot I_2}{nV_T}} \quad (4)$$

which allows deriving an expression for the series resistance:

$$R_S = \frac{V_1 - V_2 + nV_T \ln\left(\frac{I_2}{I_1}\right)}{I_1 - I_2} \quad (5)$$

To improve accuracy, iterating between Eq. (3) and Eq. (5) is required.

CJO is the zero-bias junction capacitance, which is sometimes specified in the datasheet. Alternatively, a reasonable value can be estimated from the capacitance–reverse voltage characteristic of the diode. In this simulation, the junction capacitance is not critical, since the focus is on the large-signal behavior of the component.

Revision History:

Tutorial Version 1.0	First release
Tutorial Version 1.1	Introduced intermediate checkpoint

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

© 2002–2026 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.