



**Embedded
Code Generation**
Tutorial

Hands on Introduction to PLECS TI C2000 Code Generation

Tutorial Version 1.0

www.plexim.com

- ▶ Request a PLECS and PLECS Coder trial license
- ▶ Get the latest TI C2000 and RT Box Target Support Package
- ▶ Check the PLECS, RT Box and TI C2000 TSP documentation

1 Introduction

In this tutorial you will learn about the major functional blocks of the PLECS embedded code generation tool for the Texas Instruments (TI) C2000 processor family. TI C2000 microcontrollers (MCUs) are purpose built for real-time power electronics control.

Before you begin Prior to starting this tutorial, please have available or install the following:

- One PLECS Coder license
- The TI C2000 Target Support Library: Follow the step-by-step instructions on configuring TI C2000 Code Generation in the Quick Start guide of the TI C2000 Target Support User Manual [1].
- One TI C2000 Piccolo MCU F28069 LaunchPad [2] development kit, referred to as “LaunchPad” hereafter.
- The RT Box 1 Target Support Library: Follow the step-by-step instructions on configuring PLECS and the RT Box in the Quick Start guide of the RT Box User Manual [3].

Optionally, the following items are used in are Sections 4, 5 and 6 to validate the controller in real-time. Note that these items aren’t necessary when running closed-loop simulations offline (in PLECS).

- One PLECS RT Box 1
- One RT Box LaunchPad Interface Board [4], referred to as “Interface Board” hereafter.

The hardware setup of this tutorial, with the RT Box (optional), is shown in Fig. 1. Please also check the following configurations:

- The Jumper **RST** on the Interface Board (green) is left open.
- The Jumper **JP6** on the LaunchPad (red) is left open.



Figure 1: Hardware setup of this tutorial with the RT Box (optional)

2 LED Blinker

In the first exercise you will create a simple program to blink the red LED “D9” on the LaunchPad, which is connected to GPIO 34 of the MCU. Open PLECS and create a new model file. Open the **Library Browser** from the **Window** drop-down menu. Place a Subsystem block on the main schematic and label it “Controller”.

Within the “Controller” subsystem create a model to blink the LED at 0.5 Hz with a 50% duty cycle. The LED will be switched on for 1 second and then off for 1 second. You may refer to the schematic shown in Fig. 2 for a simple approach. Please note that the Digital Out block should be taken from the **TI C2000 Target** library. Configure the **Digital output GPIO number(s)** parameter of the Digital Out block to 34.

After finishing the schematic, enter the following line of code in the **Model initialization commands** window found in the **Initialization** tab of the **Simulation + Simulation parameters...** window to define the execution step size: `Ts.Controller = 100e-6;`

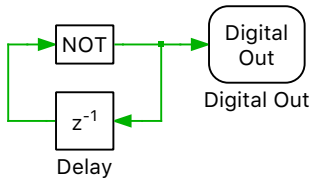



Figure 2: Controller schematic of the LED blinker

The “Controller” subsystem can be directly converted into target specific code for the TI 28069 LaunchPad, as configured. Follow the instructions below to upload the “Controller” subsystem to the LaunchPad.

- 1** Right-click on the “Controller” subsystem from the top-level schematic and select **Subsystem + Execution settings...**. In the configuration window select the check box **Enable code generation**.
- 2** From the **Coder + Coder options...** window, select “Controller” in the **System** list.
- 3** From the **General** tab enter `Ts.Controller` into the “Discretization step size” field.
- 4** Next, from the **Target** tab, select the TI2806x device from the drop-down menu.
- 5** Select **Build** and **program** as the **Build type** and select **LaunchPad** as the **Board**.
- 6** Ensure the LaunchPad is connected to the host computer through a USB cable and then click **Build**.


Please wait until the build process is finished. If programmed correctly, the red LED “D9” on the LaunchPad should blink at 0.5 Hz with a 50% on-time.

 At this stage, your model should be the same as the reference model: `c2000_tutorial_1.plecs`.

3 PWM Output

The PWM Output component of the **TI C2000 Target** library is used to generate PWM signals to control the switching devices of a power electronic converter.

- 1** Add a PWM Output block from the **TI C2000 Target** library in the “Controller” subsystem and make sure that the **PWM generator(s)** parameter is configured to 1 and the **Carrier frequency** parameter is set to `10e3`.
- 2** Use a Constant block to provide the duty cycle to the PWM block. To generate a PWM signal with a 50% duty cycle, the constant value should be set to 0.5. You may refer to the schematic shown in Fig. 3 for this step.

- 3 In order to change the duty cycle of the PWM signal during the MCU's operation, drag the Constant block into the **Exceptions** list of the **Parameter Inlining** tab from the **Coder + Coder options...** window.
- 4 Then upload the “Controller” subsystem onto the MCU by following the same instructions from the end of the last exercise.
- 5 When the upload is finished, you can connect the MCU to the host PC from the **External Mode** tab of the **Coder Options** window. First, click the  button next to the **Target device** field. Select the appropriate **Device type**, then click the **Scan** button to see the available devices. Lastly, select the appropriate device from the list of available devices using the **Device name** drop-down menu. Click **OK** to confirm the chosen device, then click the **Connect** button to activate the External Mode.

At this point, you should be able to change the duty cycle of the PWM through the parameter window of the Constant block.

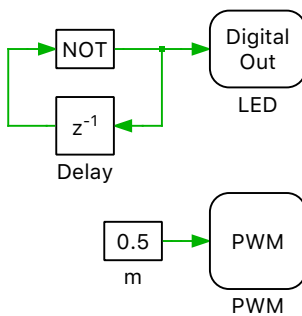


Figure 3: Controller schematic with PWM block added

Optional: You may take an analog oscilloscope and connect the two probes to Pins **39** and **40** from the **J4** connector on the LaunchPad. You will measure two PWM signals with opposite polarity. When you change the value of the Constant block (when connected to the MCU via External Mode), the change in duty cycle can be observed with the oscilloscope.

4 Hardware-in-the-Loop Simulation of a Buck Converter

Hardware-in-the-loop (HIL) simulation is a popular approach to validate embedded controller behavior. Now we will create a simple HIL model to test the LaunchPad that you have programmed. Deactivate the External Mode by clicking on the **Disconnect** button from the **External Mode** tab before making changes to the model.

Next create a new subsystem on the top-level schematic and name it as “Plant”. Please refer to Fig. 4 to build the simulation model of a buck converter in the “Plant” subsystem. Enter the following line of code in the **Model initialization commands** window found in the **Initialization** tab of the **Simulation + Simulation parameters...** window to define the plant model execution step size: `Ts.Plant = 1e-6;`

- 1 Use the IGBT Half Bridge block from the **Electrical/Power Modules** library for the switching cell.
- 2 Configure the inductor, capacitor and resistor using the values shown in the schematic.
- 3 Connect a PWM Capture block to the two IGBTs via a Signal Demultiplexer and configure the **Digital Input Channel(s)** to be the vector [0 1]. Set the **Averaging interval (offline only)** to `Ts.Plant`.

- From the top-level schematic, connect the PWM output of the “Controller” subsystem to the PWM Capture input of the “Plant” subsystem.

At this stage, you can now simulate the power converter and open-loop controller in PLECS. Run the model and observe the inductor current and output voltage in the “Plant” subsystem scope.

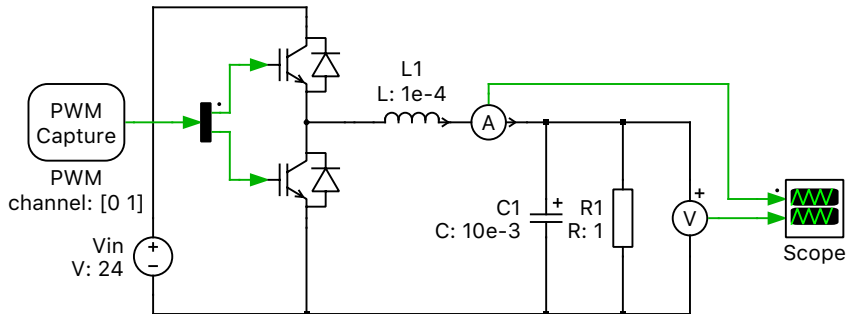



Figure 4: HIL simulation model of a buck converter

Upload the “Plant” subsystem to the RT Box by following the instructions below:

- Right-click on the “Plant” subsystem from the top-level schematic and select **Subsystem + Execution settings**. In the configuration window select the check box **Enable code generation**.
- From the **System** list of the **Coder Options** window, select “Plant”.
- From the **General** tab enter `Ts.Plant` into the **Discretization step size** field.
- From the **Target** tab, select the PLECS RT Box 1 from the drop-down menu.
- Click the binoculars button beside the **Target device** field to select an available RT Box.
- Click **Build**.

After a successful upload, keep “Plant” selected from the **System** list of the **Coder Options** window. Then activate the External Mode for the “Plant” subsystem by pushing the **Connect** button from the **External Mode** tab and click the **Activate autotriggering** button below. Open the PLECS Scope inside the “Plant” subsystem, where the inductor current with switching frequency ripple should be centered at approximately 12 A and the capacitor voltage at 12 V. You may also activate the External Mode of the “Controller” subsystem as described before and change the duty cycle of the PWM output. By doing so you will observe a change in the average inductor current and capacitor voltage of the buck converter.

 At this stage, your model should be the same as the reference model: `c2000_tutorial_2.plecs`.

5 ADC Sampling

In closed-loop control the Analog-to-Digital Converters (ADC) of the MCU are used to convert the analog signal measurements into a variable representing that signal that we can use in our embedded control code. In this exercise we will practice different ways of ADC sampling in combination with the PWM Output block introduced in the last exercise.

- Place an ADC block from the **TI C2000 Target** library in the “Controller” subsystem and configure the **Analog input channel(s)** parameter to 7.

- 2 This configuration indicates that the input of ADC channel 7 is read into the control environment, which comes from Pin **23** of the Jumper **J1** on the LaunchPad.
- 3 Then configure the “Scale” parameter to 10. Connect a Scope component to the output port of the ADC block, as shown in Fig. 5.

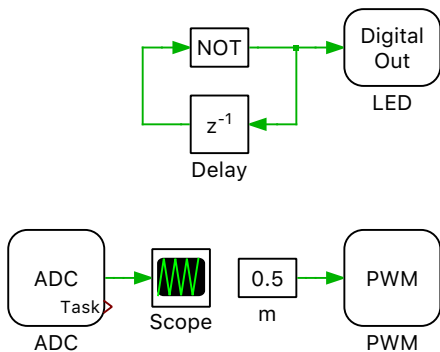


Figure 5: Controller model with ADC block added

Now we modify the “Plant” subsystem to create a current sensing signal for the LaunchPad. In low voltage applications the circuit current is usually taken from the low side switch, where a low cost shunt-based current sensor can be deployed.

- 1 Place an Ammeter in series with the low side switch of the buck converter. Note the direction of the Ammeter shown in Fig. 6.
- 2 Drag an Analog Out block from the **PLECS RT Box** library into the “Plant” subsystem, rename it as “Isw” and connect its input port to the Ammeter.
- 3 Specify the **Scale** parameter of the Analog Out block as 0.1. Then set the **Minimum output voltage** to 0 and **Maximum output voltage** to 3.3.

With these configurations, the analog output channel 0 of the RT Box will generate an analog voltage signal equal to $V = I_{sw} \cdot 0.1$ and limited between 0V and 3.3V, where I_{sw} is the current measurement from the Ammeter. On the Interface Board, this analog output channel 0 of the RT Box is connected to the ADC input 7 of the MCU on the LaunchPad.

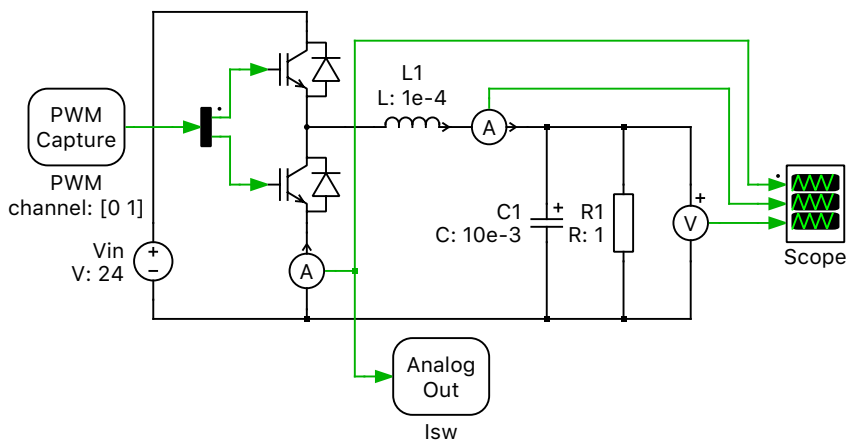





Figure 6: Buck converter model with Analog Out block added

Upload the modified “Controller” subsystem to the LaunchPad and the “Plant” subsystem to the RT Box. Activate the External Mode for the “Controller” subsystem, and observe the output in the PLECS Scope.

 At this stage, your model should be the same as the reference model: `c2000_tutorial_3_1.plecs`.

 Why is the ADC signal in the scope constantly zero (except for some irregular noise of small amplitude)?

 While the actual current has a pulsed nature, the controller happens to only be measuring it during periods when it is zero.



Note: In an MCU, the PWM signal is typically generated by comparing the duty cycle (modulation index) to a carrier wave. When the modulation index is greater than the carrier, the MCU turns on the high side switch and turns off the low side switch, which makes the inductor current increase in a buck converter. Otherwise current flows through the low side switch and the inductor current decreases. By default, the sensing signal on the ADC input channel is read when the carrier reaches “0” and the inductor current begins to flow through the high side switch. Therefore zero current is measured, as demonstrated in Fig. 7.

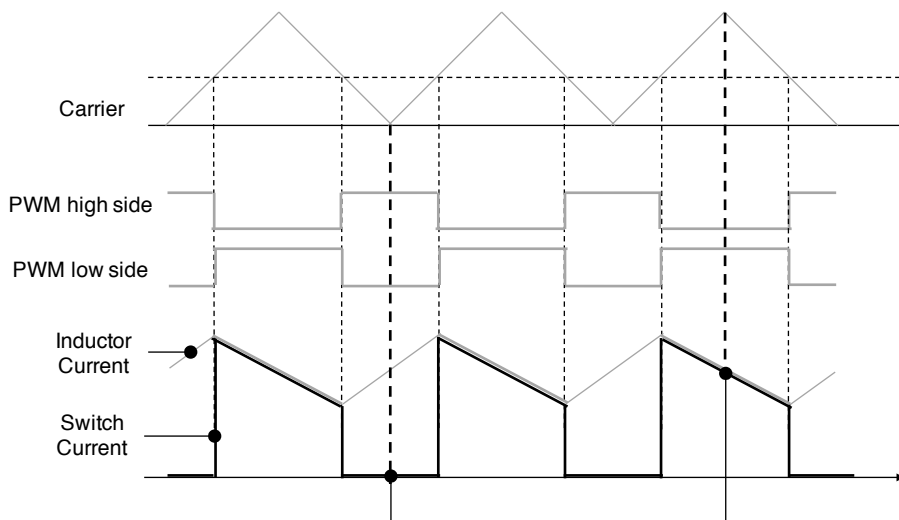


Figure 7: Options of the current sampling

When controlling the average inductor current, it is desirable to sample the current midway between the minimum and maximum current ripple. For this purpose, with low side switch current sensing, we must reconfigure the ADC so that it reads the current sensing signal when the carrier reaches its maximum. Please refer to Fig. 8 and make the following modifications:

- 1 Open the parameter window of the PWM block and switch to the **Events** tab. Change the **ADC trigger** parameter to **Overflow**.
- 2 Then in the parameter window of the ADC block change the **Trigger source** parameter to **Show trigger port**.
- 3 After completing the previous steps, an extra trigger output port should appear on the PWM block and a trigger input port should appear on the ADC block. Connect these two ports together.
- 4 Lastly, take a Control Task Trigger component from the **TI C2000 Target** and connect it with the “Task” port of the ADC block.

In summary, these configurations make the PWM block trigger the ADC sampling at carrier maximum, and once the ADC finishes the sampling, the control task (e.g., blinking the LED) is executed.

Upload the modified “Controller” subsystem and activate the External Mode. Now you should be able to see non-zero current in the PLECS Scope.

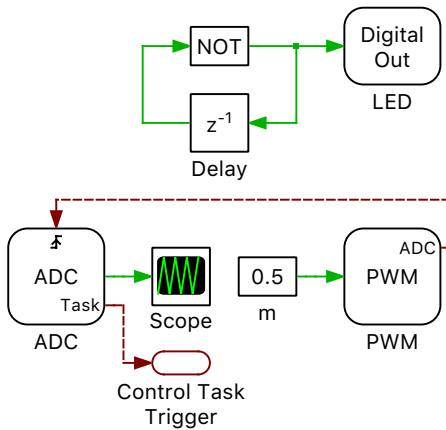


Figure 8: Configure the ADC sampling and control task execution

At this stage, your model should be the same as the reference model: `c2000_tutorial_3_2.plecs`.

6 Closed-Loop Control

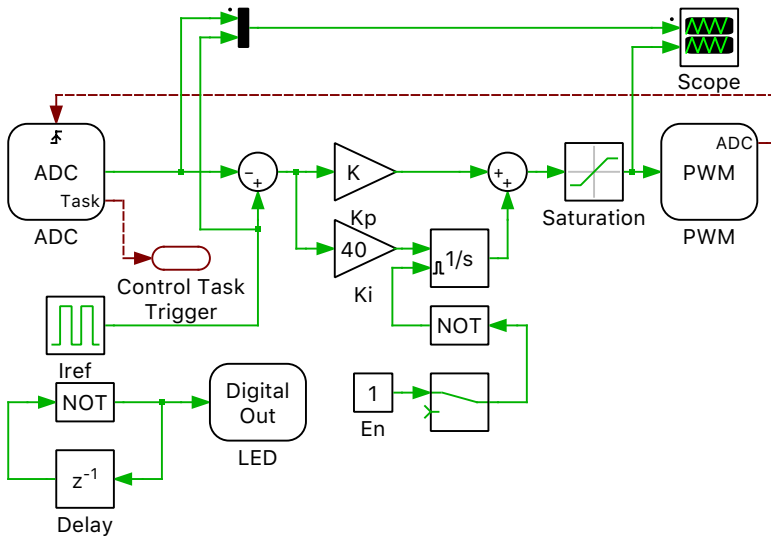


Figure 9: Controller subsystem with closed-loop PI regulator for current control

In this exercise we will configure a closed-loop current controller for the buck converter. Please refer to Fig. 9 to modify the “Controller” subsystem and do the following:

- 1 Include a Pulse Generator block “Iref” to serve as the reference current input. Configure its **Frequency** to 20 Hz, **High-state output** to 7 and **Low-state output** to 5.
- 2 Configure the “Kp” gain to 0.01 and the “Ki” gain to 40.
- 3 Configure the **Upper limit** parameter of the Saturation block to 0.95 and the **Lower limit** parameter to 0.05.
- 4 In order to avoid integrator overflow when the PWMs are deactivated, set the Integrator’s **Upper saturation limit** to 1 and the **Lower saturation limit** to 0. Then, enable the reset port of the In-

tegrator block by selecting level1 for the **External reset** parameter. Connect a Manual Switch, Constant block, and Logical Operator block with the **NOT** operator selected as shown in Fig. 9. Connect the Logical Operator output to the Integrator reset port.

- 5 Drag the Gains “Kp” and “Ki”, the Manual Switch block, and the Constant block “Iref” into the **Parameter Inlining** tab from the **Coder Options** window.

Before we upload the modified controller model onto the MCU, an offline simulation could help to pre-verify the performance of the controller. Start a simulation and observe the circuit response, where the waveforms in the scope in the “Controller” subsystem should look like those in Fig. 10. Please note that the current waveform in the “Plant” subsystem includes the switching frequency ripple, while in the “Controller” subsystem it shows only the fundamental component.

After finishing these modifications, upload the new “Controller” subsystem to the LaunchPad, then activate the **External Mode** and **Autotriggering**. You may want to synchronize the trigger with the change in current reference from the “Iref” block. Now we are going to tune the parameters of the PI regulator online.

- 1 Activate the PWM output of the MCU by toggling the Manual Signal Switch. Note that when the integrator is reset only the very small Kp gain is in effect, resulting in a low duty cycle. The Saturation block clamps the duty cycle of the PWM to 0.05. Thus there should be a small amount of current measured on the PLECS Scope.
- 2 Configure the “Kp” and “Ki” gains to some other values and observe the resulting measurements in the scope. Please be sure that the **External Mode** of the “Controller” subsystem is activated, so that the results of all changes to the PI parameters are observable during run time.
- 3 Continue to tune the PI parameters with the goal of reducing the oscillation and overshoot.

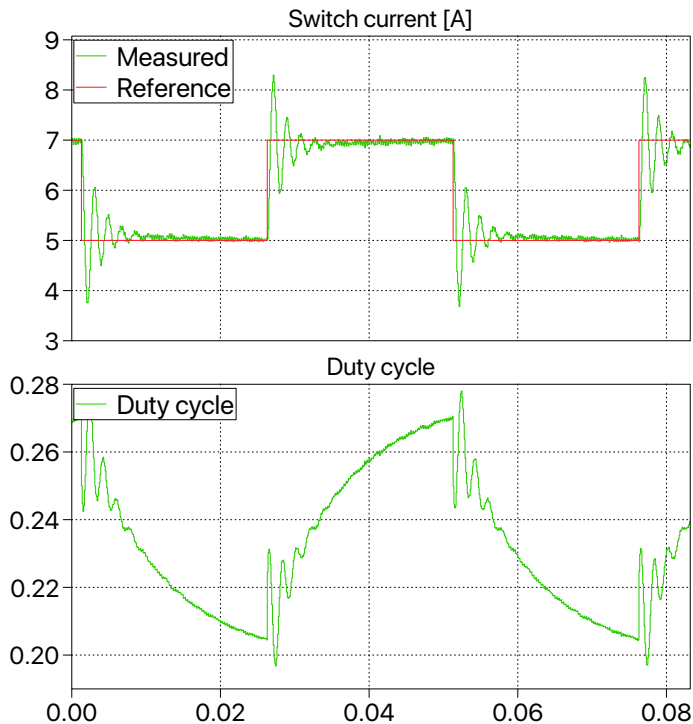


Figure 10: Circuit response with “Kp” gain equal to 0.01 and the “Ki” gain equal to 40



At this stage, your model should be the same as the reference model: `c2000_tutorial_4.plecs`.

7 Conclusion

You have now built a controller model in PLECS and deployed the control logic to a real embedded processor. During control design for a power electronic converter prototype in the future, consider utilizing the PLECS toolchain environment to take advantage of its flexibility and observability.

References

- [1] *TI C2000 Target Support User Manual*, Plexim GmbH, Online: <https://www.plexim.com/sites/default/files/c2000manual.pdf>
- [2] Texas Instruments, “*LAUNCHXL-F28069M Overview*” *User’s Guide*, SPRUI11B, 2019.
- [3] *RT Box User Manual*, Plexim GmbH, Online: <https://www.plexim.com/sites/default/files/rtboxmanual.pdf>
- [4] *RT Box LaunchPad Interface Board*, Plexim GmbH, Online: <https://www.plexim.com/sites/default/files/launchpadinterfacemanual.pdf>

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

Embedded Code Generation Tutorial

© 2002–2021 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.