Application Note

How to Run Matlab© Simulink© Motor Control Example on C2000 LaunchPad™ Development Kit

Michael Seidl

Abstract

MathWorks provides numerous Simulink examples specifically designed for TI's C2000™ microcontrollers. However, transitioning these examples from a downloaded file to a running prototype requires specific software configurations and hardware setup. This application note provides comprehensive, step-by-step guidance to streamline this process. It covers everything from identifying necessary MathWorks product requirements and selecting the compatible TI Code Composer Studio (CCS) version to configuring COM port settings for seamless PC-to-MCU communication. Finally, it demonstrates how to successfully compile, download, and control the example from a host-side GUI.

Table of Contents

Trademarks

1 Introduction

MathWorks offers a collection of MATLAB Simulink example projects that target Texas Instruments (TI)'s C2000™ microcontrollers, providing a valuable starting point for engineers who wish to implement sophisticated control algorithms on these real-time processors. However, moving from the example files to a fully functional prototype involves a series of configuration steps that are not always obvious to first-time users.

This application note walks you through the entire workflow required to take a MathWorks Simulink example and run it on a TI C2000 launch-pad. Beginning with the download of the example package Sensorless Field-Oriented Control of PMSM Using C2000 Processors - MATLAB & Simulink, the note identifies the additional MathWorks toolboxes and product licenses needed, and points out the specific version of TI Code Composer Studio (CCS) that is known to be compatible. You'll then learn how to configure the serial (COM) port settings both in the Simulink model and in the host GUI so that communication with the MCU is correctly established.

Step-by-step instructions are provided for compiling the generated C code, loading the binary onto the target board, and launching the host-side graphical user interface that interacts with the running application. By following this guide, designers can eliminate common pitfalls, reduce setup time, and gain confidence in deploying their Simulink-based control solutions on TI's C2000 microcontroller platforms.

2 Detailed Description

Use the following steps to obtain the Simulink model-based implementation of sensorless FOC running on the commercially available development kit based on LAUNCHXL-F28379D , BOOSTXL-3PHGANINV,and LVSERVOMTR (Figure 2-1) from Texas Instruments (TI).

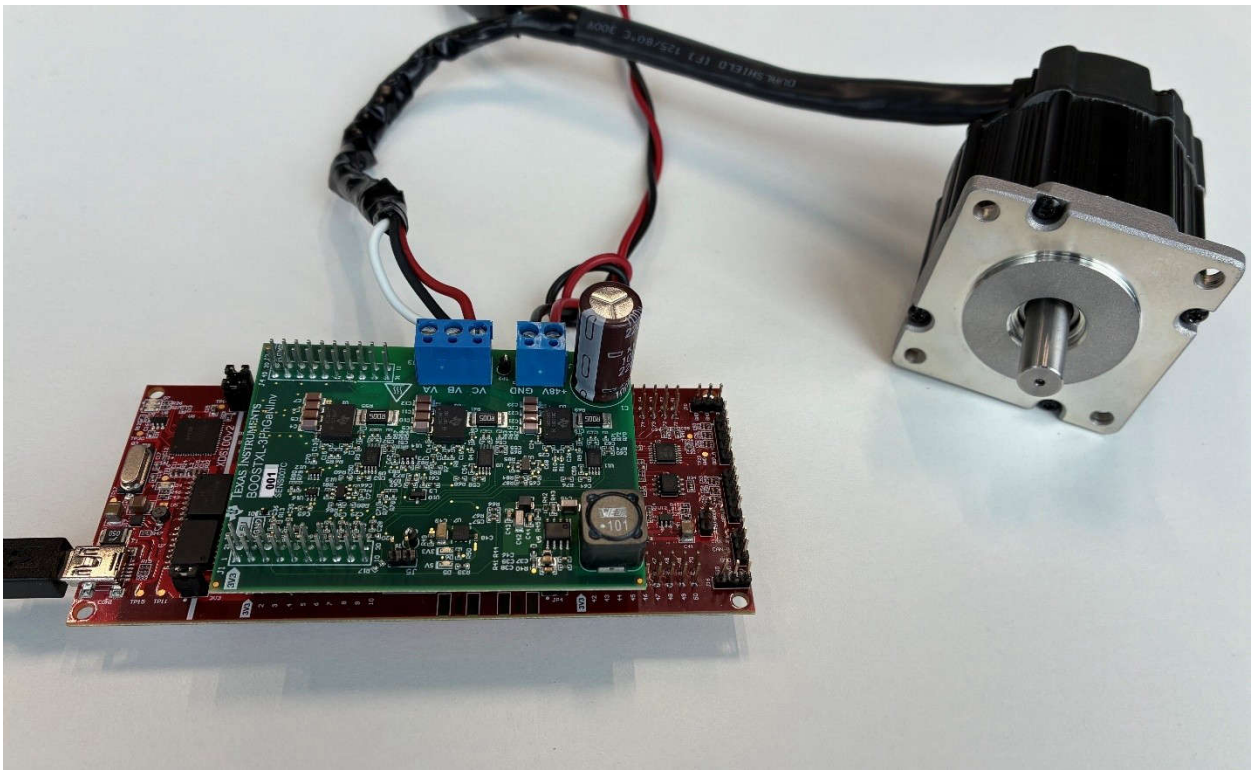

Figure 2-1. LAUNCHXL-F28379D , BOOSTXL-3PHGANINV,and LVSERVOMTR, Supported by Mathworks' Simulink Model-Based Implementation of Sensorless FOC.

Step 1:

Download the example Sensorless Field-Oriented Control of PMSM Using C2000 Processors - MATLAB & Simulink from Mathworks' web page.

Assure to have the following Products from MathWorks© installed, as well:

- Motor Control Blockset
- Embedded Coder
- Simulink Coder
- C2000 Microcontroller Blockset

Step 2: CCStudio™ IDE from Texas Instruments: The example Sensorless Field-Oriented Control of PMSM Using C2000 Processors - MATLAB & Simulink expects Code Composer Studio version 20.0.1 to compile properly.

Step 3: Find the XDS100 port number in PC Management: When the USB cable for the launchpad gets connected to the PC it assigns it to a COM Port. Figure 2-2 shows *COM9* assigned to the launchpad debugging interface in the Device Manager of Windows 11. COM9 is only used as an example for the following steps. Note down the COM port number of the setup.

Figure 2-2. Identify the COM Port Number

Step 4: Start the Matlab application, open the target model *mcb_pmsm_foc_sensorless_f28379d_datascript.m* and update *line 30* to right port (Figure 2-3). Then, click *Run* to extract the motor and board parameters.

```
29    target = mcb.getProcessorParameters('F28379D',PWM_frequency);
30    target.comport = 'COM9';       % update the appropriate serial port
```

Figure 2-3. Update the COM Port Number in the Datascript File

Step 5: Open the target model *mcb_pmsm_foc_sensorless_f28379d.slx*

Step 6: Use the *Position Estimator* button to select one of the following sensorless position estimation techniques:

- Sliding mode observer
- Flux observer
- Extended EMF observer

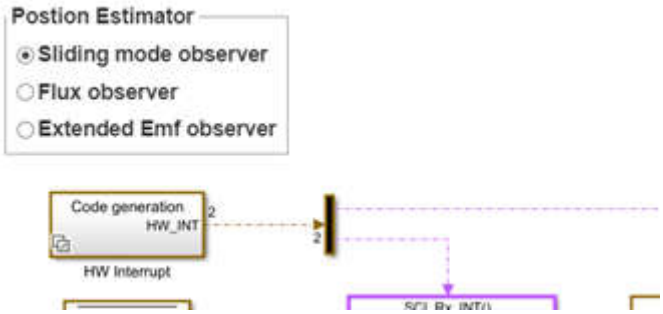

Figure 2-4. Select the Position Estimator Option in the Model

Step 7: Press *CTRL + E* to open the *Configuration Parameters* and select the right Serial PORT in “Hardware Implementation >> Target hardware resources >> External mode for Serial port in MATLAB preference (Figure 2-5)

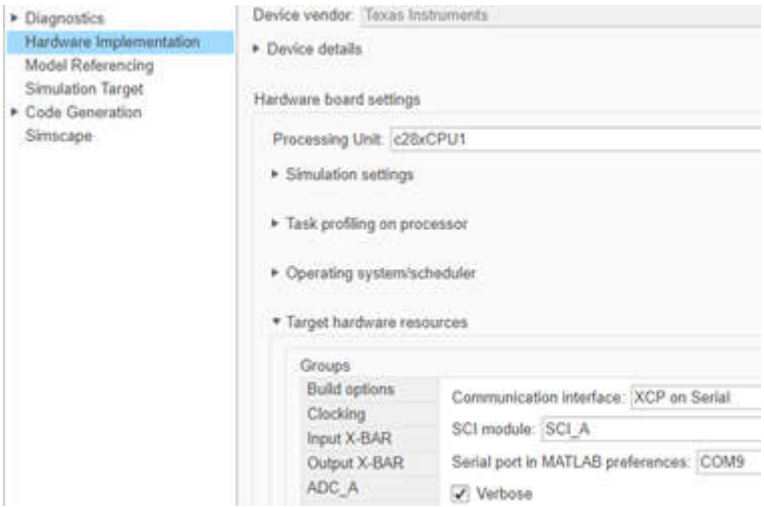

Figure 2-5. Update the COM Port Number in the Configuration Parameters

Step 8: Simulation (optional): To simulate the model, click *Run* on the *Simulation* tab

Step 9: The model automatically computes the Analog-to-Digital Converter (ADC) or current offset values. To disable this functionality (enabled by default), update the value 0 to the variable inverter.ADCOffsetCalibEnable in the model initialization script.

Generate Code and Run Model on Target Hardware

Step 10: Click *Build, Deploy & Start* on the *Hardware* tab to deploy the target model to the hardware.

Step 11: Open the *mcb_pmsm_foc_host_model_f28379d.slx*

Step 12: In the model initialization script associated with the target model, specify the communication port using the variable target.comport. The example uses this variable to update the Port parameter of the Host Serial Setup, Host Serial Receive, and Host Serial Transmit blocks available in the host model.

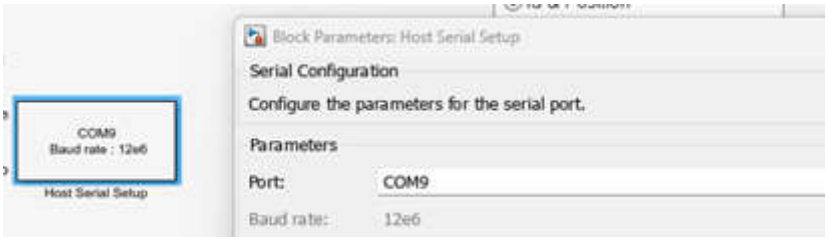

Figure 2-6. Update the COM Port Number for the Host Model

Step 13: Verify or update the Reference Speed value for the motor control in the host model.

Step 14: Click *Run* on the *Simulation* tab to run the host model.

Step 15: Change the position of the Start / Stop Motor switch to On, to start running the motor in the open-loop condition (reference speed < 300rpm).

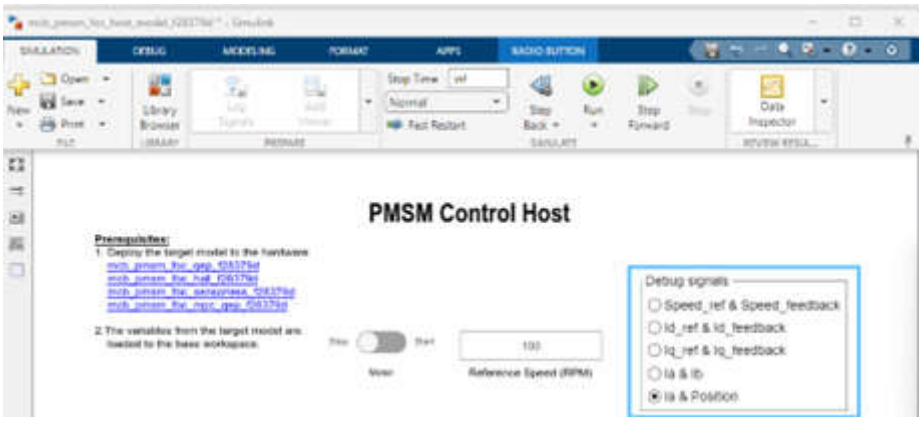

Figure 2-7. Control Host Model GUI

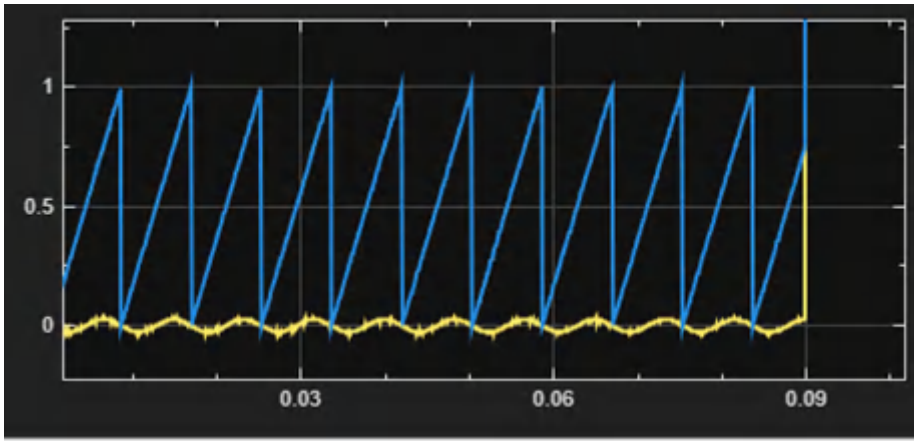

Figure 2-8. Phase A Current (Ia, Yellow Line) and Rotor Position Estimate (Blue Line) Displayed by MATLAB's *Scope* Application Window

Step 16: Select the debug signals to monitor.

3 Summary

This application note has taken you from the initial download of a MathWorks Simulink example to a fully operational prototype running on a TI C2000™ launch-pad. By following the step-by-step guidance you have:

1. Identified the required MathWorks and TI software: the necessary MATLAB/Simulink toolboxes and the compatible version of Code Composer Studio.
2. Prepared the development environment: set up the project workspace, installed required add-ons, and configured the COM-port settings in both the Simulink model and the host GUI.
3. Generated and compiled the code: used Simulink Coder to produce C code, built it in CCS, and resolved any compiler or linker issues.
4. Programmed the target board: downloaded the binary to the C2000 MCU, verified the connection, and confirmed that the firmware runs correctly.
5. Operated the system from the host: launched the GUI, communicated with the MCU, and observed the expected control behavior.

Completing these steps provides a reproducible workflow that can be reused for any of the MathWorks C2000 examples, streamlines future development, and shortens time-to-market for embedded-control projects. With the knowledge and tools presented here, the user is now equipped to leverage MATLAB/Simulink and TI's C2000 ecosystem to build robust, real-time control solutions efficiently.

4 References

1. Texas Instruments, *MATLAB® and Simulink® Model-Based Design Using C2000™ Microcontrollers*, application note.
2. Texas Instruments, *TMS320F287379D LaunchPad Development Kit*, application note.
3. Texas Instruments, *LaunchXL-F28379D Overview*, user's guide.
4. Texas Instruments, *BOOSTXL-3PhGaNInv Evaluation Module*, user's guide.

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you fully indemnify TI and its representatives against any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale, TI’s General Quality Guidelines, or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products. Unless TI explicitly designates a product as custom or customer-specified, TI products are standard, catalog, general purpose devices.

TI objects to and rejects any additional or different terms you may propose.

Last updated 10/2025