

TI C2000 Toolbox Documentation

This document shows basic information on TI C2000 Toolbox package for Typhoon HIL Control Center.

Introduction

TI C2000 Toolbox aims to enable the user to quickly and easily deploy the control algorithm on [C2000 family](#) of real-time microcontrollers developed by [Texas Instruments](#). User configure device peripherals and design the control algorithm using the Typhoon [Schematic Editor](#) and run this algorithm on target platform in several clicks, without any manual coding.

Signal processing

Signal processing is the part of the model built using components from [Signal processing components library](#). It is designed to enable the user to design control algorithm in a graphical manner. Additionally, it enables the user to perform analysis of electrical measurements, model the mechanical and thermal elements of the system, etc. Figure 1 shows the example on how signal processing can be used to control output voltage of a boost converter. Control uses two analog measurements as inputs (voltage and current) and provides gating signals as inputs of the converter. Graphical representation of the control algorithm enables the designer to rapidly makes any modifications of control algorithm and test the control performance in a simulation. This concept of testing is known as Model in the Loop (MIL).

Purpose of TI C2000 Toolbox is to make it possible to design the control algorithm in graphical environment, through signal processing, and deploy that control algorithm on a target MCU.

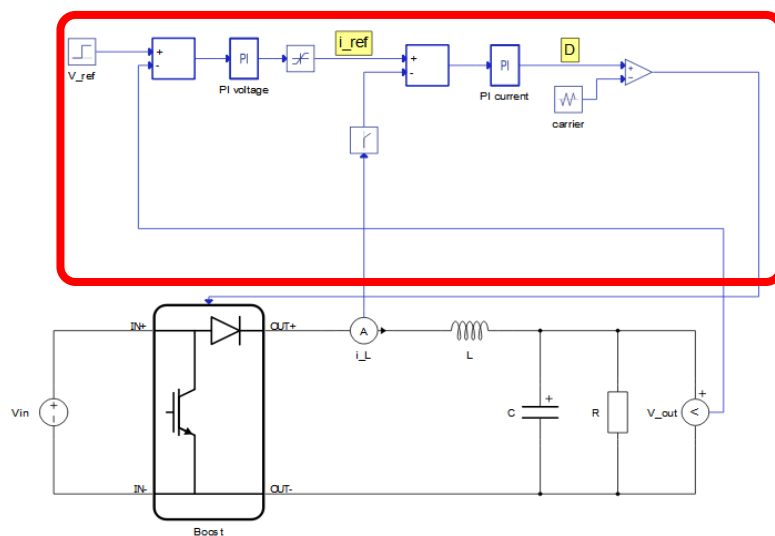


Figure 1 Signal processing control of boost converter

C code export

All of the signal processing in the model can be translated into C code, which can be exported using [C code export](#) functionality. Exported code contains two function definitions for every [execution rate](#) present in the exported part of the model:

1. Init function – Function which initializes all the variables in the code.
2. Step function – Function which is called when the control code is executed.

Toolbox functionalities

When using TI C2000 Toolbox, user can rely on several functionalities that are implemented on top of [C code export](#), as shown in Figure 2:

1. [Code Composer Studio](#) project can be generated directly from the Schematic Editor. This project includes a task scheduler which configures the step function(s) to be executed on the execution rate defined in the schematic. User can configure the project properties manually, add additional files, such as peripheral configuration files, debug the project and quickly export the new C code into the same project.
2. Configure the peripherals of the MCU, using the custom-built platform-specific components which generate the C code according to the user's input.
3. Multiplatform peripheral components can be used for several different C2000 devices.
4. Schedule code execution on given execution rates using several different interrupt sources.
5. Executable can be built directly from the Schematic Editor, once user configures all the required peripheral interfaces.
6. Flashing the controller can be done in the Schematic Editor as well.

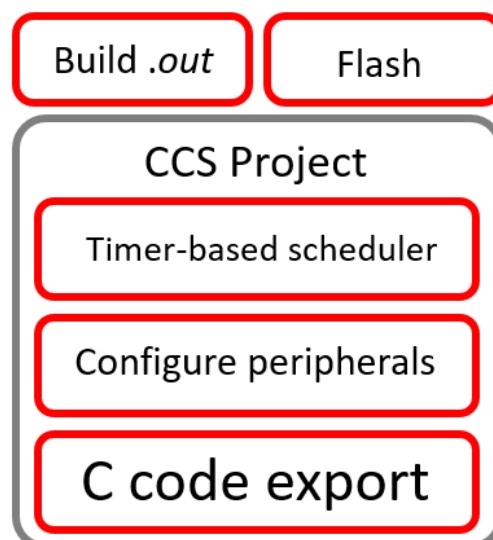


Figure 2. Functionalities of TI C2000 Toolbox

Documentation resources

First-time setup - [link](#)

This document describes how to install the package and configure the TI C2000 settings. **Recommended to follow before first deployment.**

How to configure development board - [link](#)

This document describes how to configure jumpers and DIP switches on development board to make it compatible with the toolbox. **Recommended to follow before first deployment.**

How to blink an LED? - [link](#)

Demonstrates the workflow, using the example of LED blinking.

Components overview - [link](#)

Describes what platform-specific components are available in the library, as well as some of the features that are specific to certain categories of components.

Examples - [link](#)

Demonstrates how to open an example model and run a test.

Known issues - [link](#)

Lists the known issues that will be fixed in upcoming versions.

Task scheduling principle – [link](#)

Describes architecture of the task scheduler.

How to scale analog signals - [link](#)

Explains how to scale simulated signals to adapt to ADC input voltage range.

How to reset states - [link](#)

Explains how to reset states in generated code.