ALTAIR
ONLY FORWARD

How to Use PSIM-Embed Wrapper Link

Altair PSIM Tutorial

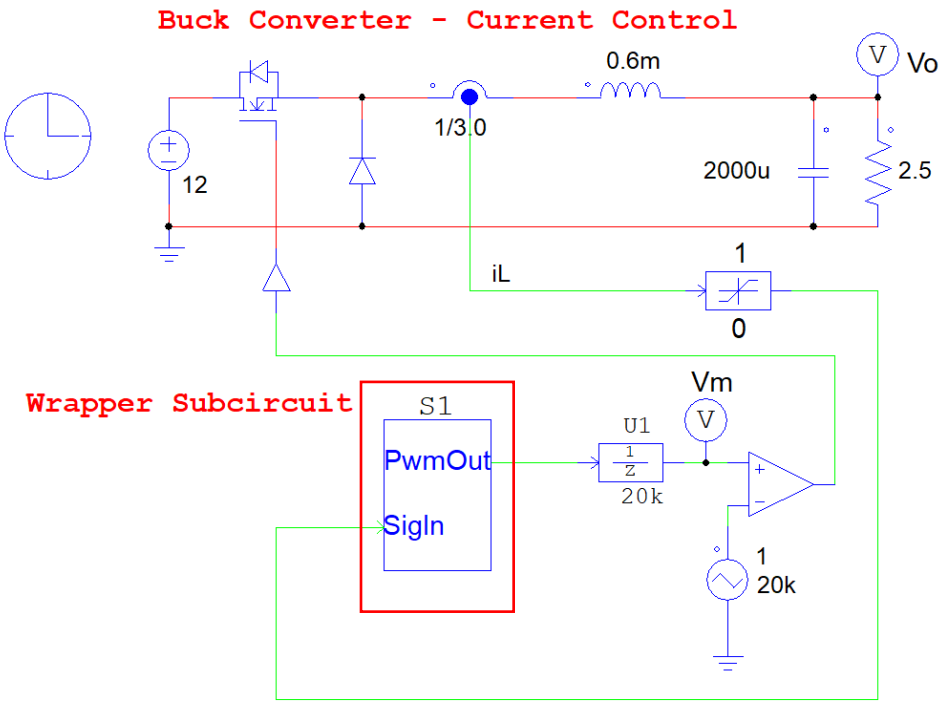
Introduction

PSIM-Embed Wrapper generates a DLL block for controller implementations inside PSIM and exports it to Embed so that the user can generate target code for any DSP that Embed supports. The generated wrapper blocks using the PSIM subcircuit can be used inside Embed's schematic environment. The DLL file will be automatically registered inside Embed under its main menu named "Wrapper Blocks". Each DLL file has one or multiple blocks under a sub-menu named "My Blocks for ***", where *** is the name of the PSIM subcircuit.

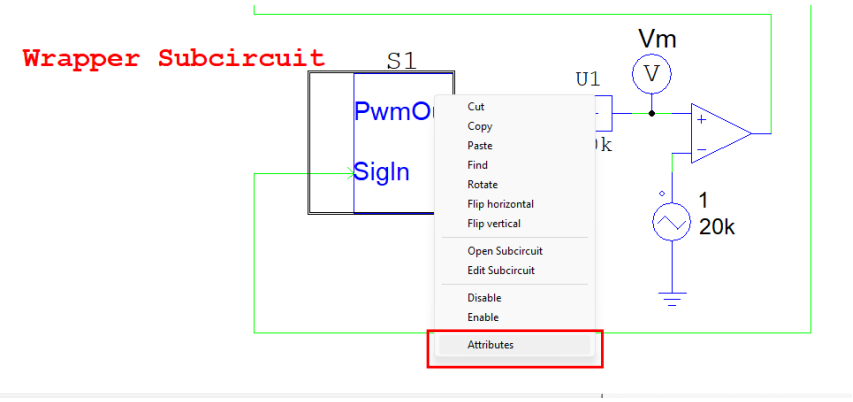
Step-by-step wrapper implementation

This tutorial will guide users on how to generate a wrapper DLL and use the generated wrapper blocks inside Embed.

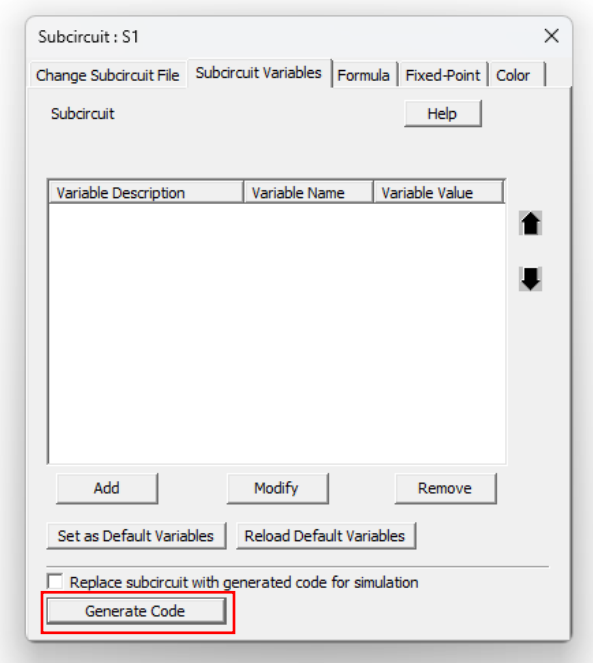
Step 1: Open a PSIM schematic example having a wrapper subcircuit.



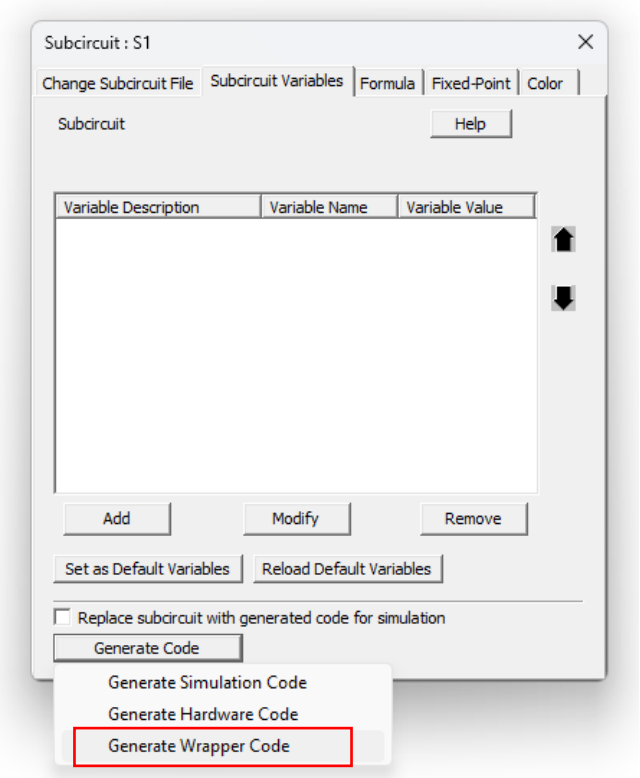
Step 2: Right mouse click on the "Wrapper Subcircuit". Select "Attributes" from the pop-up menu.



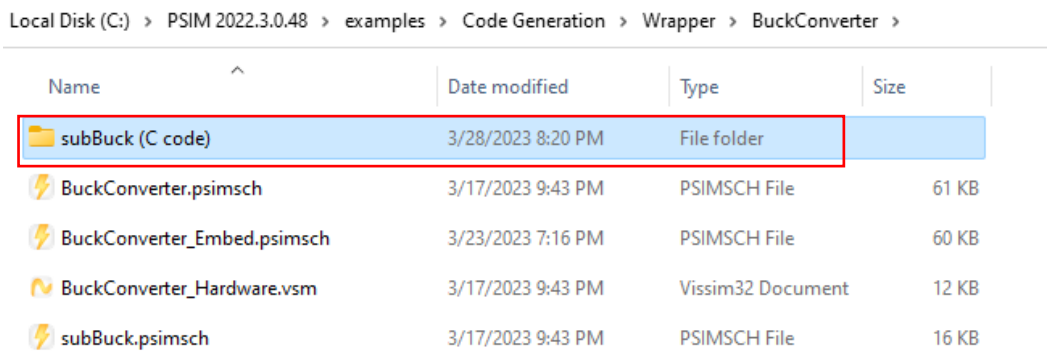
Next, a dialog will be displayed as shown below to “Generate Code”.



Step 3: Click on the “Generate Code” button and select “Generate Wrapper Code” (as shown below).



This will generate a wrapper folder at this example path destination. The wrapper folder's name will be the subcircuit's file name + (C code), as shown below.



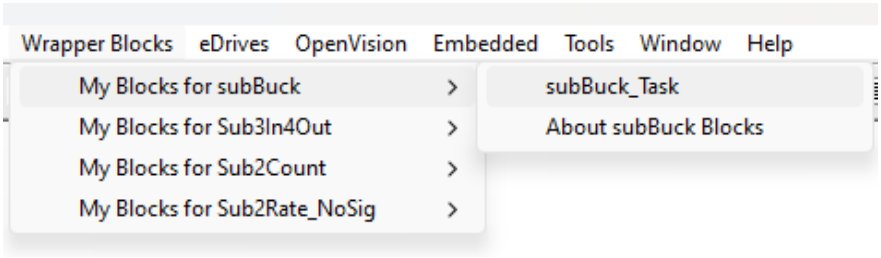
For example, suppose that one wants to generate wrapper code for subcircuit S1 and the subcircuit's file name is Sub2Count.psimsch. Then, the code is generated under the same folder as the current example. The folder name will be "Sub2Count (C code)".

Please Note: A wrapper code is compiled by Visual Studio. Wrapper searches from "C:\Program Files (x86)" to find all Visual Studio (VS) versions and choose the latest one to compile Wrapper DLL, and this Visual Studio installation folder is registered to the system registry. If Visual Studio is not installed by the default location, Wrapper may not find it; Wrapper pops up a dialog to let the user to specify the folder of VC under the visual studio folder. The wrapper can use any version from Visual Studio 2012, Visual Studio version can be any one of the followings: Industrial, Professional, Community, or Build Tools. Community and Build Tools versions are free downloads from the Microsoft website.

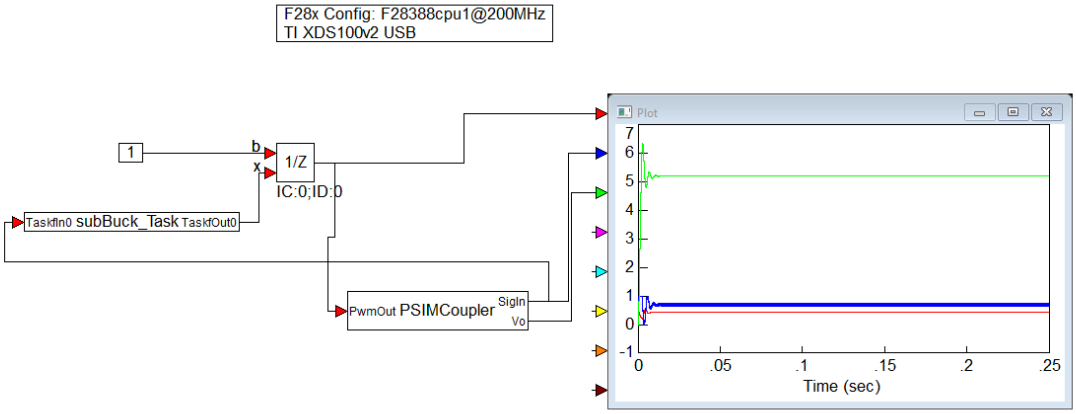
Step 4: Use Wrapper blocks inside the Embed

Wrapper DLL will be automatically registered under the main menu item “Wrapper Blocks” with the name “My Blocks for ***”, here *** is the name of Wrapper DLL.

Let’s say for this example, we have the subcircuit’s name as subBuck.psimsch. Then, the wrapper generates a block for this subcircuit and this can be reached from the main menu inside Embed, as below:

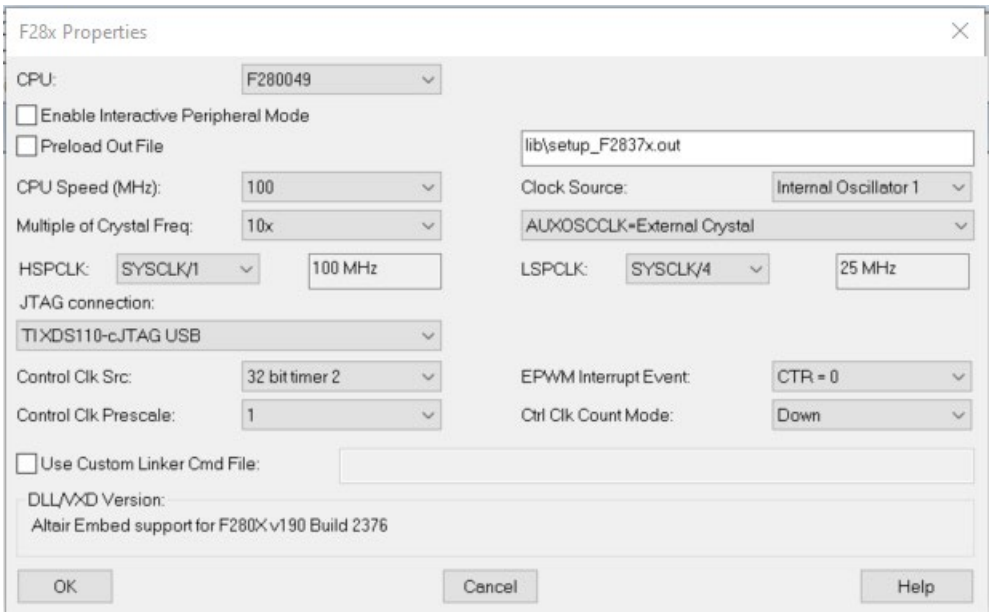


The co-simulation schematic inside the Embed, which utilizes the wrapper block generated from PSIM’s subcircuit, is shown below. There is also the PSIMCoupler block which is for co-simulation between PSIM and Embed.

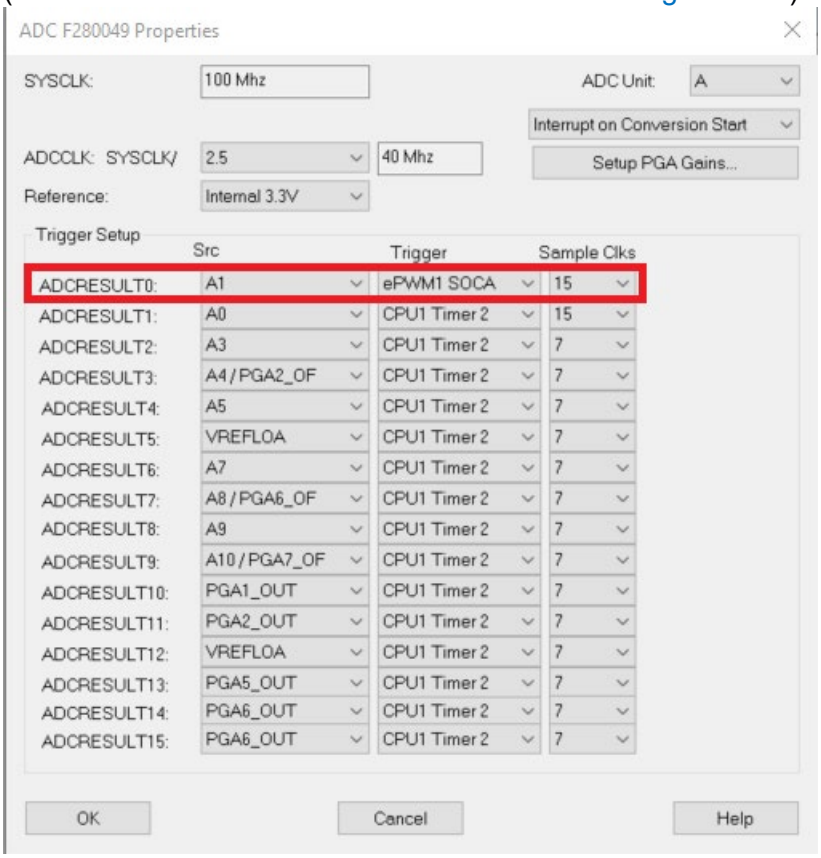


For Target code generation: To make a hardware diagram, one should put hardware device blocks before or after Wrapper blocks, Note that all Wrapper blocks must be used in the hardware diagram. Here we show the sequence to make a hardware diagram for BuckConverter:

- Assume that the hardware target is a F28049 MCU, add a F28x Config block (under Embed //Embedded/F280x/F280x Config… menu) into a new diagram, choose a JTAG type that suites your target board. For example, if the target board is a F280049C LaunchPad, select “TI XDS110-cJTAG USB”. You can also change other parameters to suite your target board.

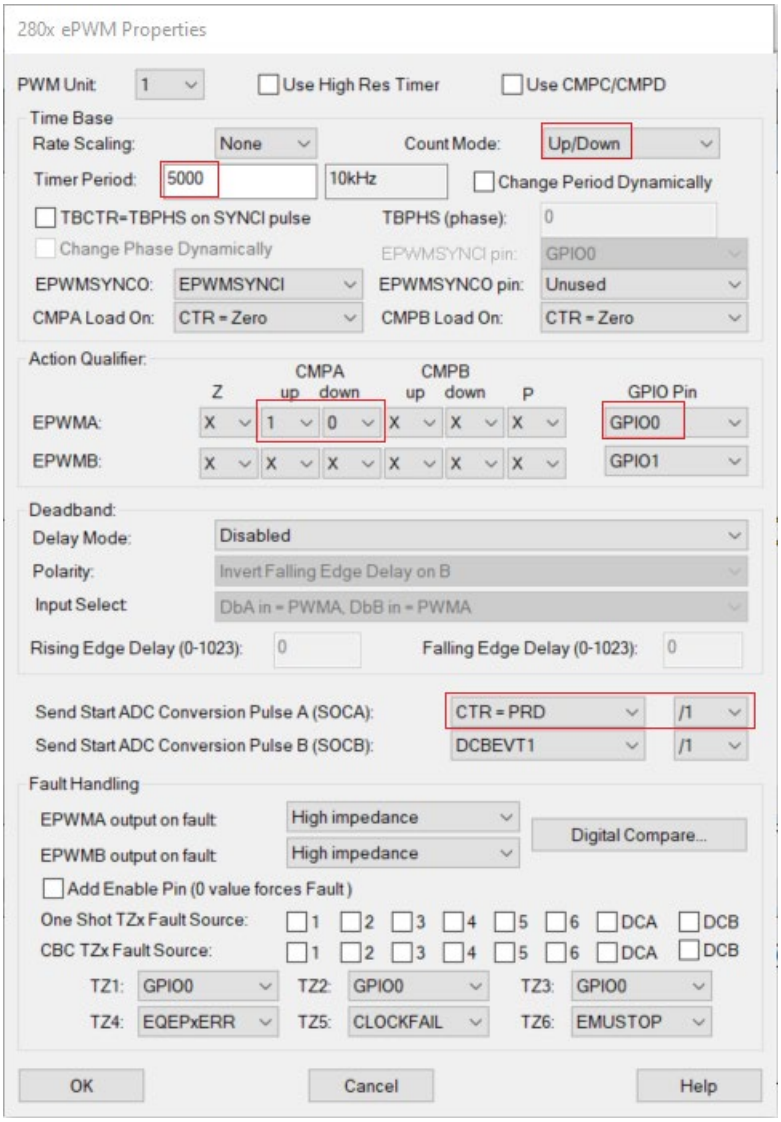


- Buck converter uses an ADC channel (say ADCA1), one should add an Adc Config block (under Embed //Embedded/F280x/ADC/ADC Config… menu) for ADC settings:

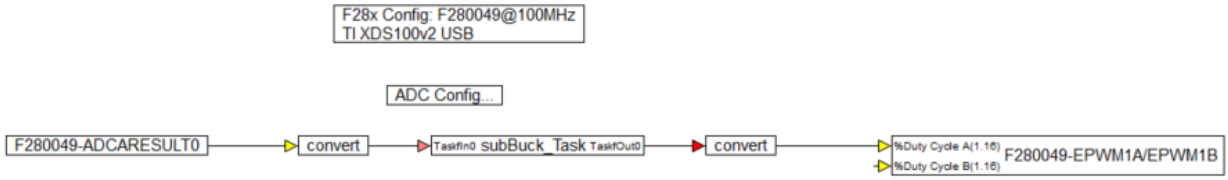


- Add all SubBuck blocks (only one here) into a new diagram, then add ADC result block (under Embed //Embedded/F280x/ADC/Analog Input menu) before SubBuck block, Since ADC result is a fixed-point number, one should add a convert block to convert the ADC result into float number.

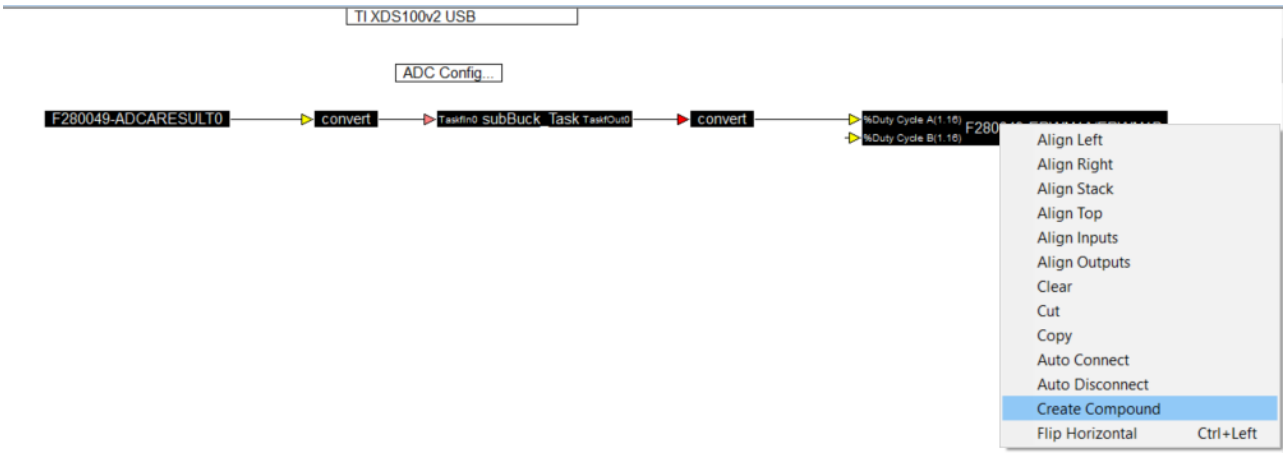
- Add PWM1 block (under Embed //Embedded/F280x/PWM/ePWM menu) then set PWM properties as below, then add a convert block set the float number to a fx1.16 fixed-point number:



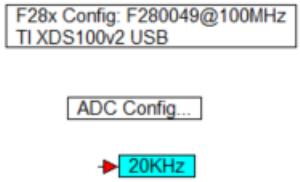
- The final diagram is as below:



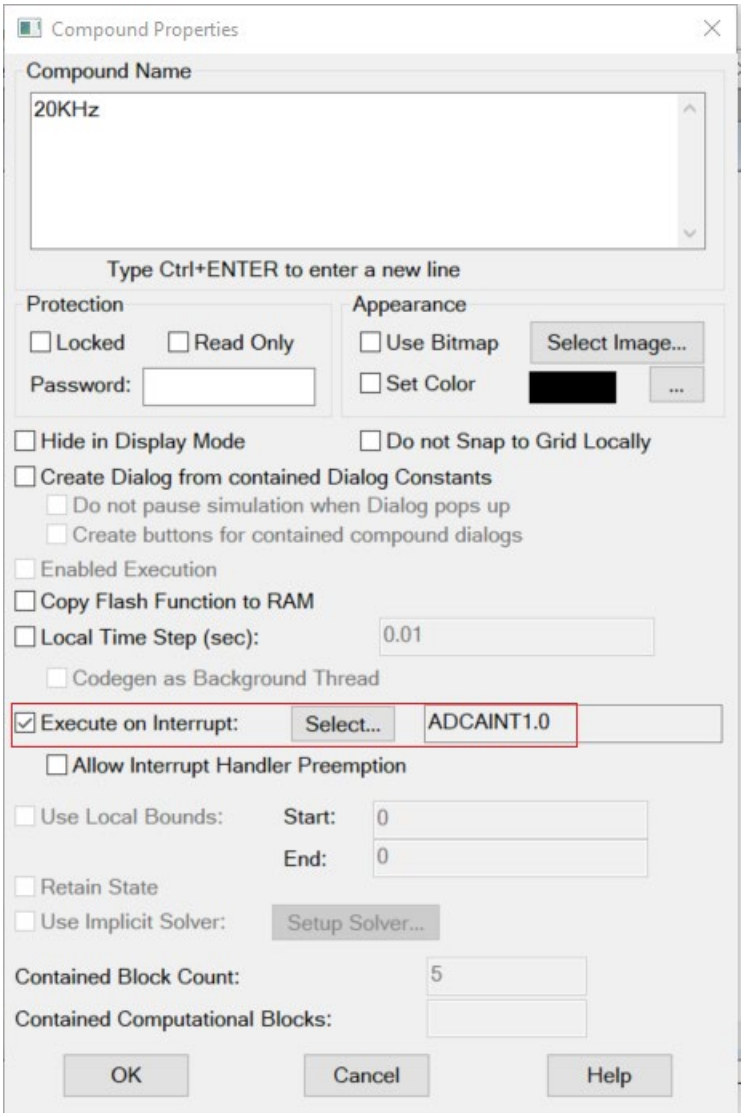
- The hardware system is designed that PWM1 triggers ADC at the end of a PWM period, ADC starts to convert then causes interrupt at the end of conversion. To implement this, one needs to select all blocks from input device block to output device block as below and right mouse click on a selected block to create a compound block.



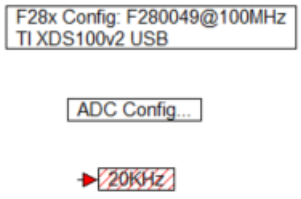
- Input the compound name at the properties page. The diagram is changed as below:



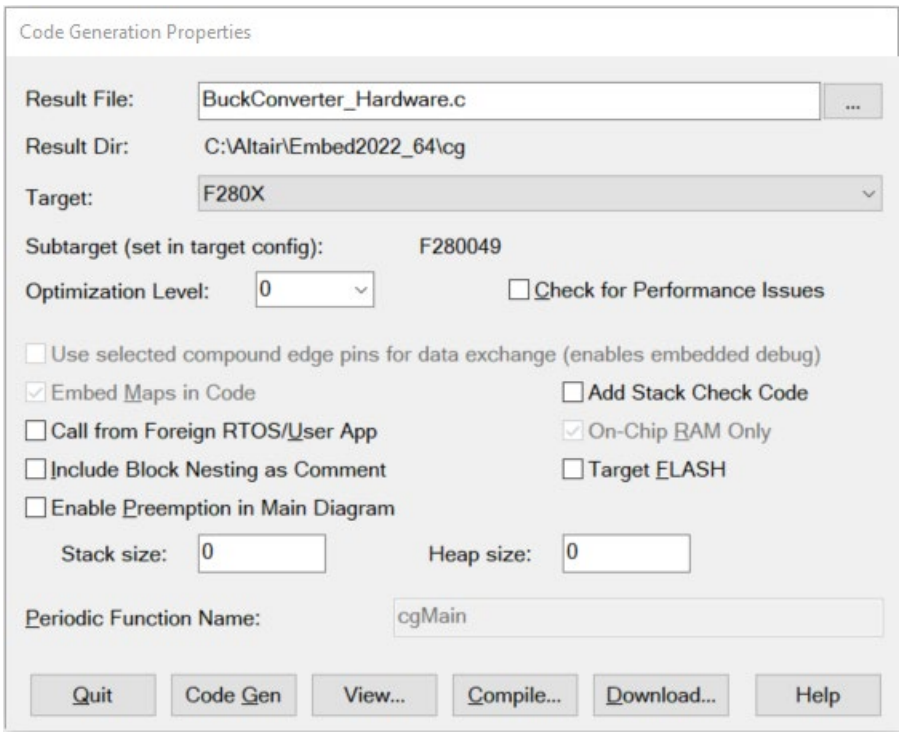
- Ctrl + right mouse click on the compound block, the following properties page shows up:



- The final diagram is as below:

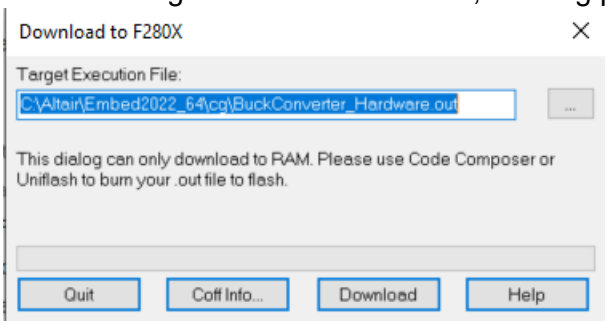


- One can generate hardware code in Embed main menu //Tools/CodeGen…, a dialog show up as below:



Click “Code Gen” button to generate hardware code, Click “Compile…” button to compile the hardware code, Click “View…” button to show the generated code. Click “Download” button to send the compiled binary code to MCU.

- When clicking on “Download” button, a dialog pops up as below:



Connect your target board with the specified JTAG, Power on the board, then click “Download” button, the binary code will be sent to the MCU and executed.

This includes the steps on PSIM-Embed wrapper implementation.

Wrapper’s limitation

The PSIM-Embed wrapper is a new enhancement from PSIM v2022.3 release. There are currently some limitations while using Wrapper. When generating a Wrapper code for a PSIM subcircuit, this subcircuit must follow the following rules (including the child subcircuits):

- No power blocks.
- No hardware blocks.
- Not support fixed point type (only floating point type now).
- No bi-directional ports in the current subcircuit and its child subcircuits.
- No PSIM motor control blocks (They uses F28x .obj files, we may compile them to ST or other DSP object file if needed).

- No SimCoder C block.
- No TIDMC blocks.
- Not support IQmath and any blocks use IQmath functions.