TMS320F28335 DSP Programming using MATLAB Simulink Embedded Coder: Techniques and Advancements.

Akrem Elrajoubi, Student IEEE member,
Department of Electrical Engineering
University of Arkansas
Fayetteville AR, USA
amelrajo@uark.edu

Simon S. Ang, Fellow, IEEE,
Department of Electrical Engineering
University of Arkansas
Fayetteville AR, USA
siang@uark.edu

Ali Abushaiba, Student IEEE member,
Department of Electrical Engineering
University of Kansas
Lawrence KS, USA
abushaiba@ku.edu

***Abstract*—This paper provides a tutorial on how to program Texas Instruments™ (TI) TMS320F28335 Digital Signal Processor (DSP) through Code Composer Studio (CCS) version 6 and MATLAB Simulink Embedded Coder. First it shows how to setup Simulink and Embedded Coder and produce code to program TMS320F28335 and variant of TI's C2000 DSPs. It describes how to interact between MATLAB 2015b and CCS V6 and provides an explanation of the vital steps and settings needed to program the DSP. Basic functions such as Pulse Width Modulation, Analog Digital Conversion, and Proportional-Integral controllers are explained. Finally, closed loop control model for a micro-inverter topology is developed.**

Keywords— TMS320F28335 DSP, CCS V6, MATLAB Simulink.

I. INTRODUCTION

Designing closed loop feedback multi PWM control schemes is much harder and more time consuming in CCS than in MATLAB Simulink Embedded Coder for Texas Instruments™ TMS320F28335 Digital Signal Processor (DSP). TMS320F28335 DSP is a cheaper controller which proved excellent convergence, and real time control for significant reduction of output ripple [1]. The code is automatically generated using the embedded coder, and so time for programming and control implementation is reduced. TI DSPs present effective and simple control design for DC motors [2]. MATLAB Simulink for DSP controller is highly valuable as model design, simulation, code generation, debugging and running can be accomplished for control algorithm [3]. MATLAB Simulink environment is especially recommended for control algorithm implementation into micro controller [4, 5]. The goal of this paper is to provide a simple and clear procedure to begin learning how to program the TMS320F28335 DSP from Texas Instruments™ (TI) through Code Composer Studio (CCS) and MATLAB Simulink Embedded Coder.

To begin, you need to make sure that you have the Embedded Coder, MATLAB Coder, and Simulink Coder toolboxes installed on your PC. Embedded coder sits on top of MATLAB and Simulink coder; it allows you to add device specific code (ADC's, DAC's, CAN, etc.) to what it produces by the respective coders. An easy way to check the toolboxes installed on your version of MATLAB is by entering the "ver" command into the command window. Using Simulink code generation is more effective than writing line by line any code by CCS which takes a long time for users to program the DSP [6]. Section II presents the required settings for the DSP target configuration, while section III demonstrates the fundamental blocks used in power electronics and motor drives applications from Simulink and their main parameters. Serial Communication Interface (SCI) transmit and receive blocks are shown in section IV, and then closed loop control model is developed for the micro-inverter topology in section V.

II. CCSV6 TARGET CONFIGURATION

First make sure that Code Composer Studio (CCS) version 6 or 5 is installed. If you have an older version of CCS installed, you should upgrade to version 5 or CCS version 6 which is now supported in Simulink for code generation. Embedded Coder (EC) works for previous versions but they are no longer supported by Texas Instruments. For setting up xMakefile in Simulink which tells EC where the CCS 6 compiler is installed among other programs. So MATLAB can call the command lines provided by CCS 6, and where MATLAB can find the compiler needed to create the makefile code. However, "xmakefilesetup" command in MATLAB is no longer needed because of the Embedded Coder Support package for TI C2000 Processors which permits the settings for model configuration parameters easily. Type "supportPackageInstaller" command to launch the Support Package Installer Graphical User Interface in MATLAB.

Then type this command "checkEnvSetup('ccsv5','f28335','check')" in MATLAB, as shown in Fig. 1 the tools must be installed properly. For F28335 we do not need header files but we need to install Flash APIs from TI ControlSUITE webpage. In case you need to check the compiler just type in the command window: mex –setup , or mex.getCompilerConfigurations. For Matlab 2015a you will need to install the Embedded Coder Support package for TI C2000 Processors. But for Matlab 2011b and 2013a versions the library already exists. Then we choose C2833x processor for our DSP control card.

978-1-5090-5326-1/17/$31.00 ©2017 IEEE

```
>> checkEnvSetup('ccsv5','f28335','check')

1. CCSv5 (Code Composer Studio)
   Your version     : 6.1.0
   Required version: 5.0 or later
   Required for     : Code Generation
   TI_DIR="C:\ti\ccsv6"

2. CGT (Texas Instruments C2000 Code Generation Tools)
   Your version     : 5.12.3
   Required version: 5.2.1 to 6.0.2
   Required for     : Code generation
   C2000_CGT_INSTALLDIR="C:\ti\ccsv6\tools\compiler\c2000_15.12.3.LTS"

3. DSP/BIOS (Real Time Operating System)
   Your version     : 5.41.11.38
   Required version: 5.33.05 to 5.41.11.38
   Required for     : Code generation
   CCSV5_DSPBIOS_INSTALLDIR="C:\ti\bios_5_41_11_38"

4. XDC Tools (eXpress DSP Components)
   Your version     :
   Required version: 3.16.02.32 or later
   Required for     : Code generation

5. Flash Tools (TMS320C28335 Flash APIs)
   Your version     : 2.10
   Required version: 2.10
   Required for     : Flash Programming
   FLASH_28335_API_INSTALLDIR="C:\ti\controlSUITE\libs\utilities\flash_api\2833x\28335\v210"
>>
```

Fig.1. MATLAB checkEnvSetup command for F28335 DSP.

A. FLASH memory Programming (stand-alone mode)

If you are going to work on RAM programming mode, you will set the configuration parameters to make the system target file as either “ert.tlc” or “idelink_ert.tlc” on the code generation page. For stand-alone mode, the program will be saved in the flash memory of the DSP, so it will not be erased when the DSP control card is unplugged from the computer. Check boot from flash option in configuration parameters window as shown in Fig. 2. This setting in Model Configuration Parameters window tells EC what sort of DSP is being programmed so that it initializes the right peripherals, uses the correct operation frequency, knows how much memory is available, etc. Open the configuration parameters and ensure that the solver is set to fixed-step and discrete, the fixed-step size should remain auto. The hardware implementation page should show Texas Instruments, C2000, and Little Endian [7, 8].

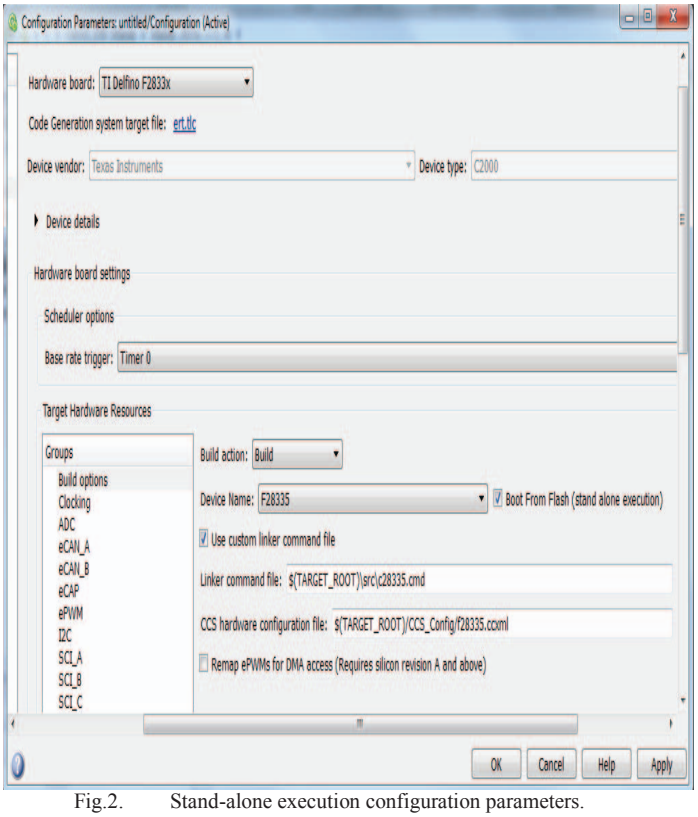

Fig.2. Stand-alone execution configuration parameters.

B. Debug Configurations and Code Running

We can generate the code by click Deploy to Hardware (build model) icon. It will build dot out file which is downloadable program file in CCSV6. Once you open CCS you can select the configuration from the debug dropdown menu and CCS will automatically connect to the DSP and load the .out file for that project as shown in Fig. 3. You can check code generation report after loading the model to see the comments and the highlighted hyperlinks for specific blocks in the Simulink model.

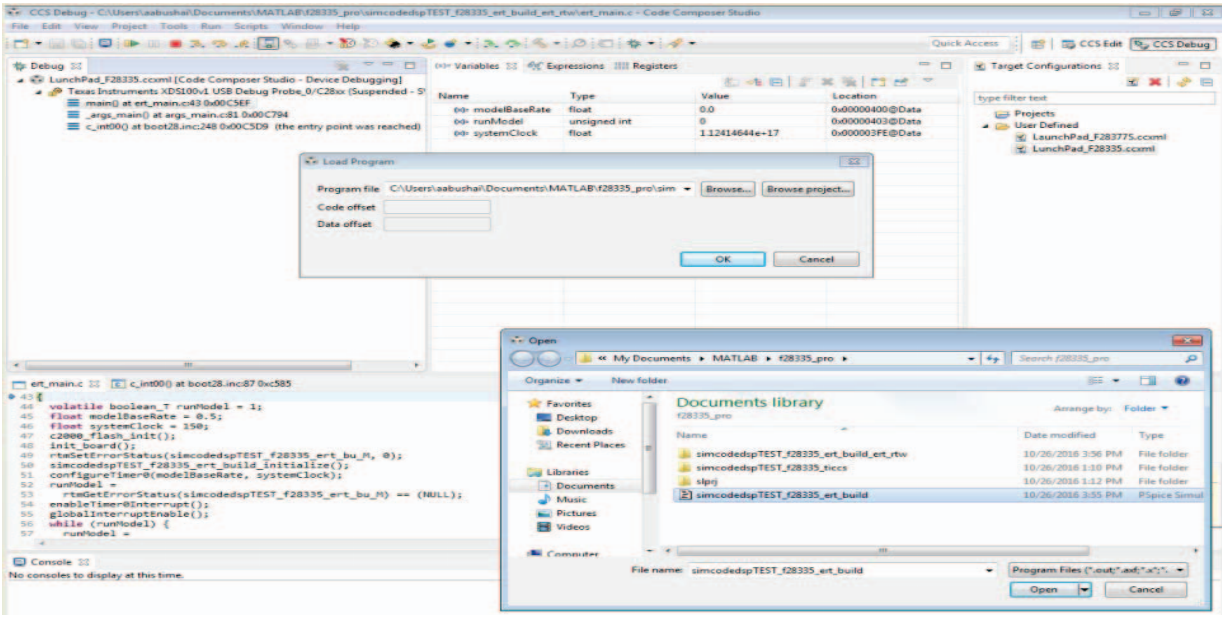

Fig.3. Loading the .out file for that project.

A TMS320F28335DSP board is shown in Fig. 4 with the blinking GPIO34 LED after running the program. Since this programming was done on the flash memory of the DSP, you can turn on and off the USB switch to see that the program is still working and has not been erased from the RAM by turning off the power.

III. PWM, ADC, GPIO, AND PI BLOCKS

TMS320F28335DSP has up to 18 PWM outputs which is adequate to control many three phase power converters. 12 of these outputs are ePWM modules which are shown in Table 1. SYSCLK is set to 150 MHz, which is the maximum clock speed of the TMS320F28335 as specified in the CPU clock. Fig. 5 shows two synchronized enhanced Pulse Width Modulator (ePWM) blocks together to use the frequency and duty cycle as inputs.

The pulses shown in Fig. 6 are for the two ePWM outputs shown in Fig. 5 while they are synchronized with a phase shift of TBPHS = 1500 (number of cycles for the time period). Notice that these two ePWM outputs are 180 degree shifted, the switching frequency is 50 KHz, and each ePWM has two complementary signals. The duty cycles are 0.33 (=1000/(2*1500)) and 0.23 (=700/(2*1500)) for channel 1 (ePWM1A) and channel 3 (ePWM2A), respectively.

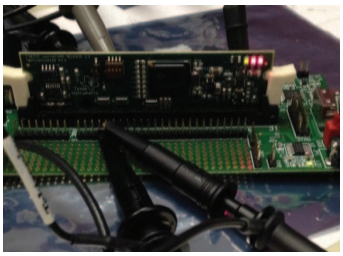

Fig.4. TMS320F28335DSP board.

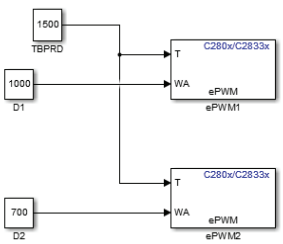

Fig.5. Synchronized two ePWM blocks.

Fig. 7 shows ePWM Block Parameters to generate two complementary PWM signals with a frequency of 20 KHz and duty cycle of 0.5. For frequency of 40 KHz, the Timer period would be 1875. But for up counting mode it would be 3750. The duty cycle is depending on the CMPA value and the counting mode specified in Fig.7. Here it is up counting mode so D= 0.5 since 1875/3750 = 0.5. As can be noticed, the frequency and duty cycle can be specified as input ports or via dialog. Under the General tap, the Timer period can be specified and it is calculated as in (1). Every action can be chosen from the available choices (Do nothing, Clear, Set, and Toggle). As can be seen, Action when counter=CMPA on up-count (CAU) is Set, and Action when counter=CMPA on down-count (CAD) is Clear. Also counting mode can be chosen (Up, Down, or Up-Down).

Timer period = (150 MHz)/(2 * 20 KHz)=3750 (1)

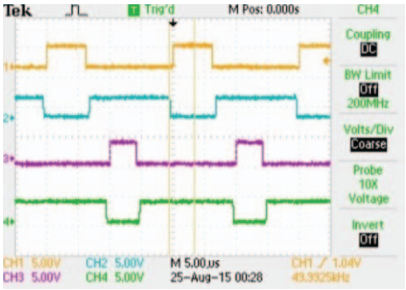

Fig.6. Synchronized ePWM pulses.

TABLE I. EPWM OUTPUT SIGNALS.

ePWM Modeule	Module Outputs	GPIO Pin
ePWM1	ePWM1A	GPIO00
	ePWM1B	GPIO01
ePWM2	ePWM2A	GPIO02
	ePWM2B	GPIO03
ePWM3	ePWM3A	GPIO04
	ePWM3B	GPIO05
ePWM4	ePWM4A	GPIO06
	ePWM4B	GPIO07
ePWM5	ePWM5A	GPIO08
	ePWM5B	GPIO09
ePWM6	ePWM6A	GPIO10
	ePWM6B	GPIO11

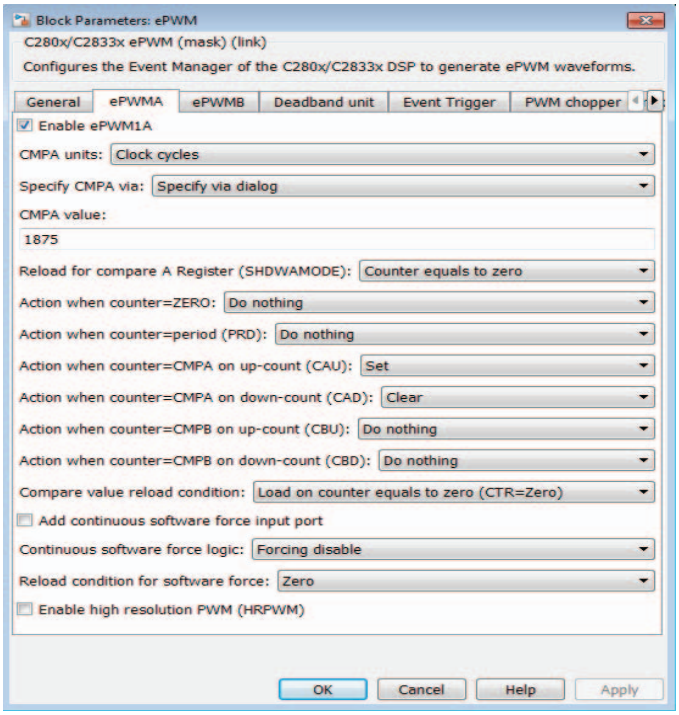

Fig.7. ePWM Block Parameters.

Fig.8 shows a simple example to use the Analog Digital converter (ADC) block to perform analog-to-digital conversion of signals connected to the selected ADC input pins. The output of the ADC is a vector of `uint16` values.

The output values are in the range 0 to 4095 because the ADC is a 12-bit converter, the input channel is ADCINA0. Notice in Fig.9 that the option (Post interrupt at the end of conversion) has been unchecked, and the sample time is equal to the time period of the ePWM output pulse (f=20 KHz). When we build, download and run this simple model to the DSP board, we connect the power supply DC voltage to the pin ADCINA0, and the negative to the ground point. So, we are able to control the duty cycle of this PWM signal by changing the voltage from 0 to 3V as can be seen on the oscilloscope. Notice that on the ePWM block you need to check the option (Enable ADC start of conversion for module A) as shown in the figure below. Likewise we can control the frequency of the ePWM output pulse by the ADC input when we make it an input port.

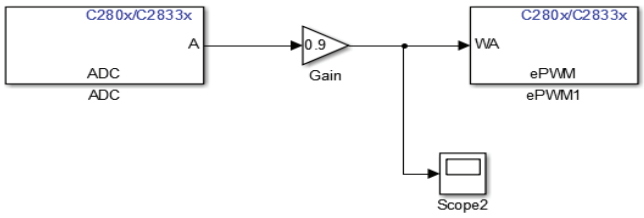

Fig.8. ADC Example.

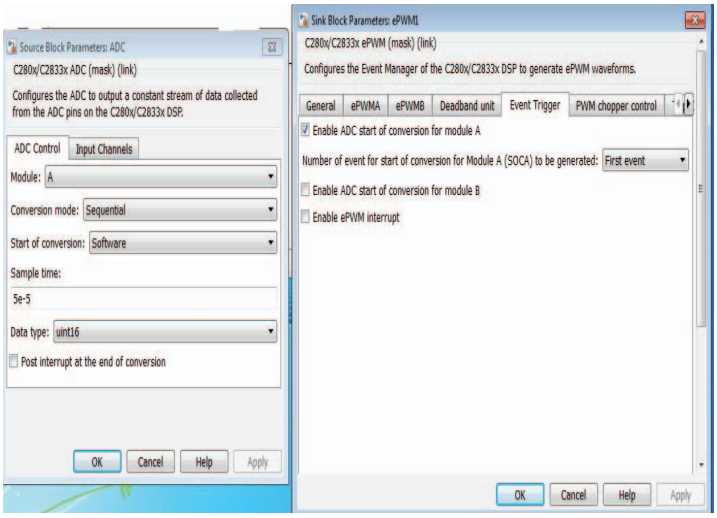

Fig.9. ADC and ePWM Blocks Parameters.

Fig. 10 shows an example model of using the PI controller for changing the duty cycle of the ePWM output signals. Here the frequency was set to 40 KHz by making the time period 3750 and Up counting mode.

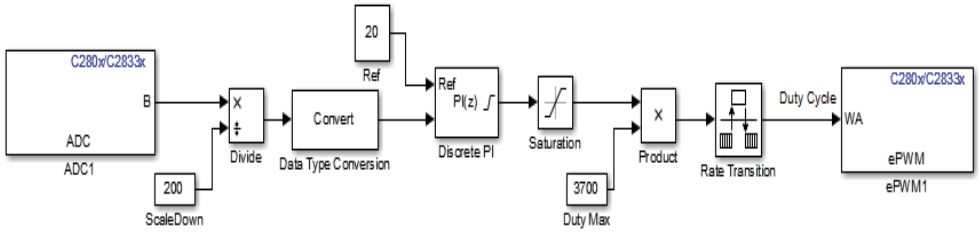

Fig.10. PI controller for duty cycle of ePWM output.

It is important to have the system controller designed conveniently to make sure the response is as desired. Notice that data type conversion block and rate transition block have been included to convert input signal to specified data type, and handle transfer of data between blocks operating at different rates.

These are the basic settings to program the TMS320F28335 DSP; there are a couple other things that can be done to improve the code output. For instance, we can set objectives for the code output in Configuration Parameters window under the code generation advisor, such as execution efficiency, rom efficiency, and ram efficiency. If you click the set objectives button you will see a pop up and you can put the options in a prioritized list. After you have a model built you can click the check model button and Simulink will analyze your model and provide tips to improve the generated code. Now it is good to start building your model. You can use any blocks from math, logic, and discrete block sets. Other blocks can be used as long as they do not have continuous states, the best way to check is by trial and error, add it to your model and build the program. If the block is not compatible Simulink will tell you. In order to get to the DSP specific blocks (ADC, GPIO, CAN, PWM, etc.) scroll down to Embedded Coder > Embedded Targets > Processors > Texas Instruments C2000 > C2833x. There are also a number of IQ math and motor control blocks located in the optimization subgroup. Some of these blocks include Clark transformation, PID controller, Park transformation, speed measurement, and space vector generator.

IV. SERIAL COMMUNICATION

Fig. 11 shows the Serial Communication Interface (SCI) transmit block, and Fig. 12 demonstrates the use of the SCI receive block to establish the serial communication between the host computer and the C2000 target hardware. As such, they work to receive and transmit scalar or vector data using the specified SCI hardware module.

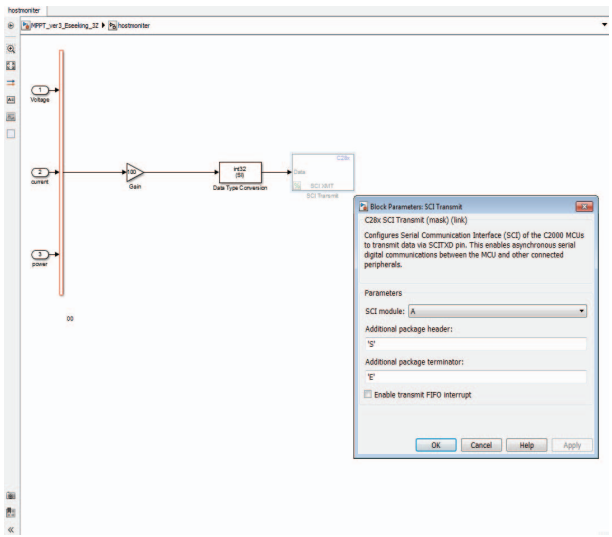

Fig.11. SCI transmit block.

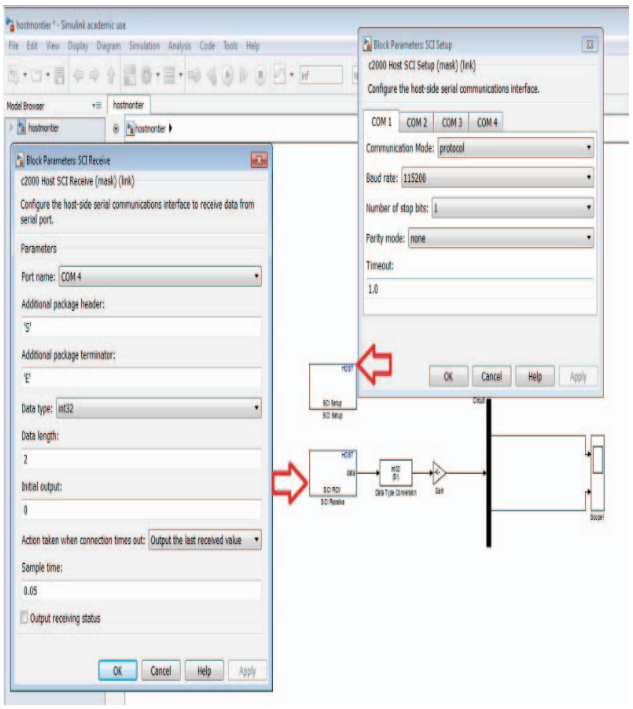

Fig.12. SCI receive block.

V. CLOSED LOOP CONTROL EXAMPLE

This micro inverter has been designed using an interleaved boost series resonant converter (IBSRC). As shown in Fig. 13, the IBSRC topology has two synchronous boost converters having 180° phase shift and connected to a high frequency transformer [10]. Then the secondary side is connected to an H bridge inverter to obtain an AC output voltage.

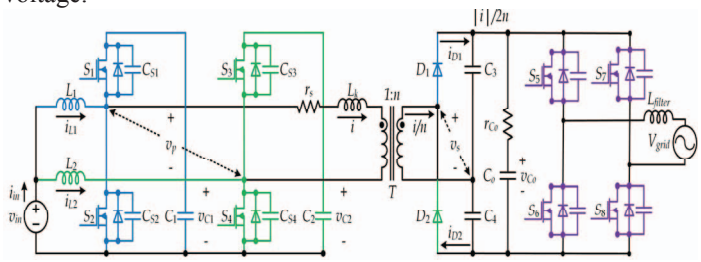

Fig.13. Interleaved-Boost Series-Resonant Micro-inverter Topology [10].

The Model in Fig. 14 has been developed to control the gate drivers of the IBSRC by controlling the frequency and utilizing 4 ADC inputs as feedback from IBSRC sensors. Each ePWM block generates two complementary pulses, so this model produces 8 pulses for the micro inverter system. The DSP will generate the signals for the gate drivers for these 8 switches. The ePWM blocks are connected to these 8 switches as presented in Table 2. This is based on the design of the IBSRC PCB board. In Fig. 14 four sensors were used for the input signals of ADC1 and ADC2, and the PI controllers implemented a discrete-time controller in Simulink® model. The PID Controller block allows you to implement setpoint weighting in your controller to achieve

both smooth setpoint tracking and good disturbance rejection [9]. The reference values for the PI controllers are for the comparison with the ADC output to obtain the targeted time period for ePWM blocks. Fig.15 shows the IBSRC was designed and developed in reference [10] with the F28335 DSP Card on the prototype.

TABLE II. EPWM SIGNALS FOR IBSRC 8 SWITCHES.

ePWM1A	S1	ePWM1B	S2
ePWM2A	S3	ePWM2B	S4
ePWM3A	S5	ePWM3B	S6
ePWM4A	S7	ePWM4B	S8

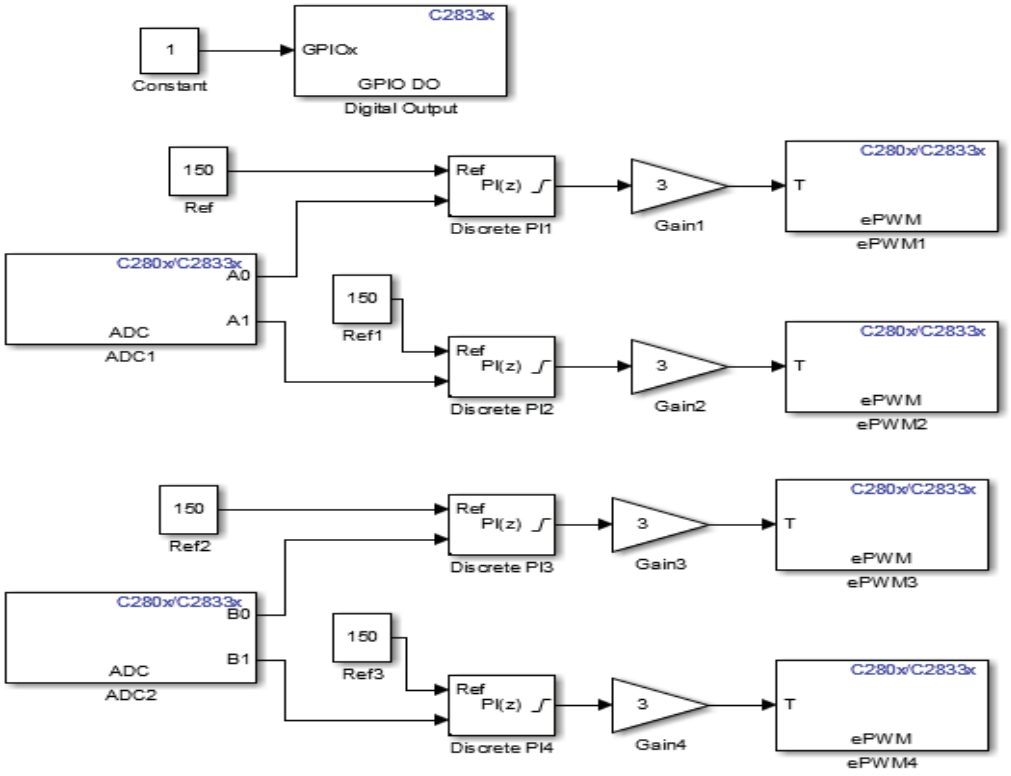

Fig.14. TMS320F28335 DSP Model for IBSRC system.

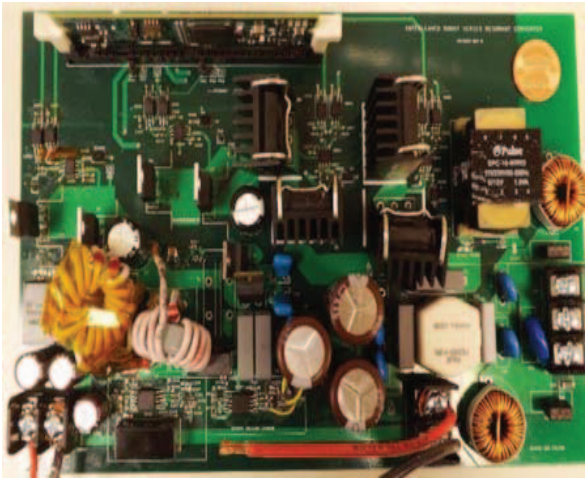

Fig.15. Interleaved Isolated Boost Series Resonant Converter Prototype [10].

VI. CONCLUSION

Using MATLAB Simulink embedded coder tools with no hand writing code is time and money saving, more efficient, faster execution time, and great for research and industrial control design. This tutorial gives the instructions on how to program the TMS320F28335 micro-controller using make file approach with embedded coder in MATLAB Simulink 2015b, and then debugging the program in CCS V6. In this paper, the main steps for the interaction between MATLAB 2015b and CCS V6 are investigated and explained. Different versions of MATLAB have some differences to interact with CCS, F28335 DSP could be programmed using MATLAB 2011b and 2013a, but there are some errors for flash programming mode. IBSRC system was controlled by F28335 DSP through MATLAB Simulink Embedded Coder. The Embedded Coder Support package for TI C2833x Processor provides the fundamental blocks needed for any power electronic, smart grid or motor drives applications.

REFERENCES

[1]. L. Katzir, Y. Loewenstern, B. Bishara and D. Shmilovitz, "Implementation of a high voltage power supply with the MATLAB/Simulink embedded coder," *2014 IEEE 28th Convention of Electrical & Electronics Engineers in Israel (IEEEI)*, Eilat, 2014, pp. 1-4.

[2]. R. Duma, P. Dobra, M. Abrudean and M. Dobra, "Rapid prototyping of control systems using embedded target for TI C2000 DSP," *2007 Mediterranean Conference on Control & Automation*, Athens, 2007, pp. 1-5.

[3]. S. Yuan and Z. Shen, "The Design of Matlab-DSP Development Environment for Control System," *2012 Third International Conference on Digital Manufacturing & Automation*, GuiLin, 2012, pp. 903-906.

[4]. R. Grepl, "Real-Time Control Prototyping in MATLAB/Simulink: Review of tools for research and education in mechatronics," *2011 IEEE International Conference on Mechatronics*, Istanbul, 2011, pp. 881-886.

[5]. B. dos Santos, R. E. Araújo, D. Varajão and C. Pinto, "Rapid Prototyping Framework for real-time control of power electronic converters using simulink," *IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*, Vienna, 2013, pp. 2303-2308.

[6]. J. C. Molina Fraticelli, "Simulink Code Generation Tutorial for generating C code from Simulink Models using Simulink Coder" NASA MARSHALL SPACE FLIGHT CENTER 2012. Available at: https://www.mathworks.com/matlabcentral/answers/uploaded_files/56491/Simulink%20Code%20Generation.pdf

[7].http://www.mathworks.com/help/ecoder/ug/tutorial-using-xmakefile-with-code-composer-studio-4-x-1.html?searchHighlight=code+composer+studio.

[8]. http://www.ti.com/tool/tmdscncd28335

[9]. http://www.mathworks.com/products/embedded-coder/webinars.html#.

[10]. L. A. Garcia-Rodriguez, C. Deng, J. C. Balda and A. Escobar-Mejía, "Analysis, modeling and control of an interleaved isolated boost series resonant converter for microinverter applications," *2016 IEEE Applied Power Electronics Conference and Exposition (APEC)*, Long Beach, CA, 2016, pp. 362-369.