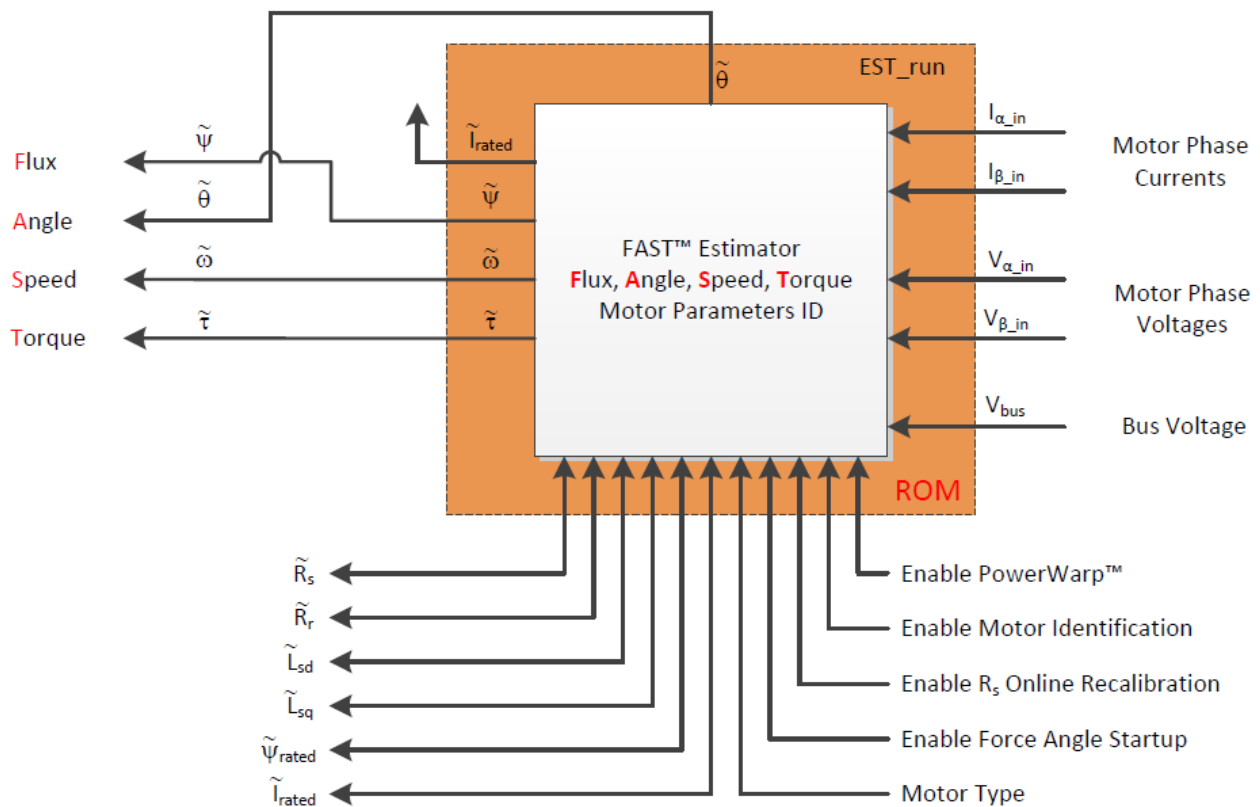


TUTORIAL
Simulation and Code Generation of
TI InstaSPIN Using DRV8312 EVM

January 2017

PSIM supports TI’s InstaSPIN-FOC sensorless motor control algorithm in simulation and SimCoder auto code generation. With this capability, PSIM provides the easiest way for users to evaluate the performance of a motor control algorithm with InstaSPIN.

The core of the InstaSPIN algorithm is a FAST estimator that performs parameter identifications and calculates flux, angle, speed, and torque based on motor phase voltages and currents and dc bus voltage. The FAST block is illustrated below.



In PSIM, a block called “PIL Block (InstaSPIN)” is provided under **Elements >> Control >> PIL Module** is provided to implement the functions of InstaSPIN’s FAST block. This block can be used in general simulation or in a SimCoder circuit for auto code generation for DSP hardware. At the moment, only F2806x DSP is supported. F2802x DSP can be supported upon request.

This tutorial describes how the PIL InstaSPIN block is used for simulation and for auto code generation. The TI DRV8312 EVM board with the F28069M controlCARD is used to run the examples. For further information on InstaSPIN functions, please refer to relevant TI documents.

1. Quick Start

If you wish to run the examples described in this tutorial right away, follow the steps below.

To run the simulation with InstaSPIN control:

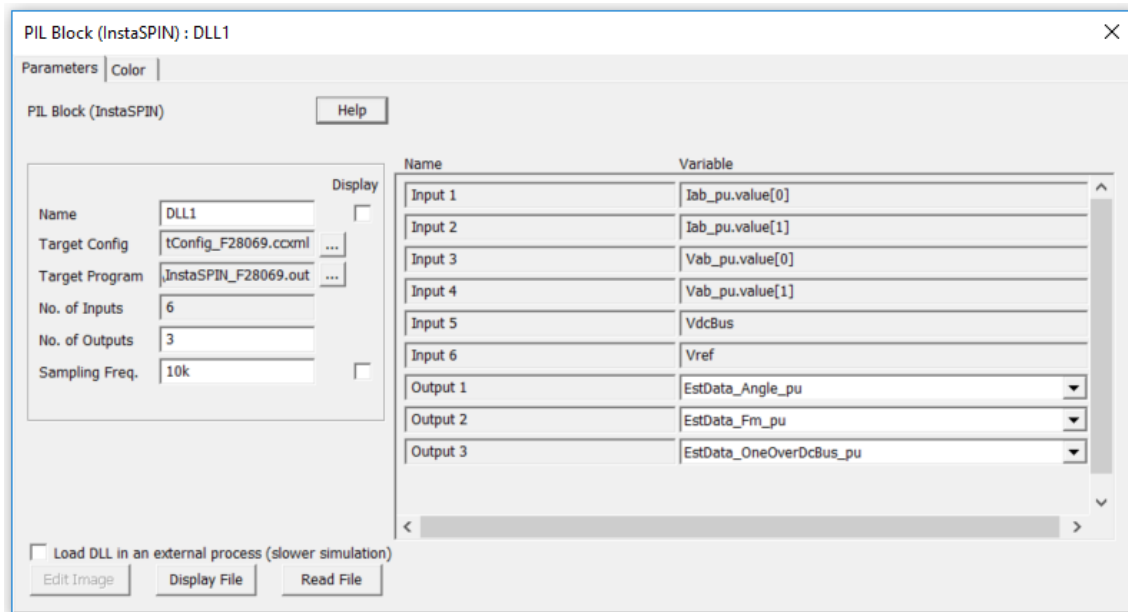
- Insert the F28069M controlCARD to the DIM100 connector of the DRV8312 EVM board. Connect the computer to the F28069M controlCARD. Connect a 24-Vdc power supply to the DRV8312 EVM board.
- Launch PSIM. Load the file “DRV8312 PMSM InstaSPIN Lab11 (F28069).psimsch” from the PSIM folder “examples\PIL\DRV8312 PMSM InstaSPIN Lab11 (F28069)”.
- Select **Simulate >> Run Simulation** to run the simulation. After simulation is completed, display waveforms in SIMVIEW.

To generate code from the schematic with InstaSPIN control, and run code on the hardware to spin the motor:

- Insert the F28069M controlCARD to the DIM100 connector of the DRV8312 EVM board. Connect the computer to the F28069M controlCARD. Connect a 24-Vdc power supply to the DRV8312 EVM board.
- Connect the Anaheim BLDC motor BLY172S-24V-4000 to the DRV8312 EVM board.
- Launch PSIM. Load the file “DRV8312 PMSM InstaSPIN Lab11 (F8069).psimsch” from the PSIM folder “examples\SimCoder\F2806x Target\TI DRV8312 PMSM InstaSPIN Lab11”.
- Select **Simulate >> Generate Code** to generate code. The generated code will be stored in the folder “examples\SimCoder\F2806x Target\TI DRV8312 PMSM InstaSPIN Lab11\DRV8312 PMSM InstaSPIN Lab11 (F8069) (C code)”.
- Launch TI Code Composer Studio (CCS). Select **Project >> Import Legacy CCSv3.3 Projects**, and navigate to the folder of the generated code and import the project.
- Compile the project, and run the code. The motor should start to spin.

2. PIL InstaSPIN Block Definition

The PIL InstaSPIN block has 6 inputs and 3 outputs by default. The dialog window is shown below.



The number of inputs is fixed, but the number of outputs can be changed. All inputs are in per unit, and the data format is IQ24.

The block parameters are explained as below:

Target Config	Target configuration file .ccxml used in CCS for the specific DSP hardware. The configuration file must match the hardware connected to the computer. By default, a F28069 configuration file "TargetConfig_F28069" in the lib subfolder is used.
Target Program	Target hardware executable .out file. By default, the file "InstaSPIN_F28069.out" in the lib subfolder is used.
No. of Inputs	The number of inputs to the block is fixed to 6 at the moment. These inputs are: <ul style="list-style-type: none"> lab_pu.value[0]: alpha component of the 3-phase currents after abc-alpha/beta transformation lab_pu.value[1]: beta component of the 3-phase currents after abc-alpha/beta transformation Vab_pu.value[0]: alpha component of the 3-phase voltages after abc-alpha/beta transformation Vab_pu.value[1]: beta component of the 3-phase voltages after abc-alpha/beta transformation Vdcbus: DC bus voltage Vref: Speed reference to the controller

No. of Outputs	Number of outputs from the block. It can be changed to any number as needed. Also, each output variable can be changed through the drop-down menu. The three default outputs are: EstData_Angle_pu: Rotor angle, in per unit EstData_Fm_pu: Mechanical frequency of the motor, in per unit EstData_OneOverDcBus: Inverse of the dc bus voltage, in per unit
Sampling Freq.	Sampling frequency of the block, in Hz. It is the frequency in which the block runs.

The block can have the following outputs, all in IQ24 format and in per unit unless otherwise stated.

gldq_pu.value[0]:	Id value of the current
gldq_pu.value[1]:	Iq value of the current
EstData_Angle_pu:	Rotor angle
EstData_DcBus_pu:	DC bus value
EstData_ErrorCode:	Error code
EstData_Fe_pu:	Electrical frequency of the motor
EstData_Flux_pu:	Flux value
EstData_Fm_pu:	Mechanical frequency of the motor
EstData_ForceAngleDelta_pu:	Force angle delta value
EstData_FreqB0_lp_pu:	Low-pass filter numerator value in the frequency estimator (in IQ30)
EstData_FreqBeta_lp_pu:	Value used to set the pole location in the low-pass filter of the frequency estimator (in IQ30)
EstData_Fslip_pu:	Slip frequency of the motor
EstData_IdRated_pu:	Id rated current value
EstData_IdRated_indEst_pu:	Id current value used for inductance estimation of induction motors
EstData_IdRated_ratedFlux_pu:	Id current value used for flux estimation of induction motors
EstData_KRPM_to_PU_sf:	krpm to pu scale factor
EstData_Lr_pu:	Rotor inductance value (in IQ30)
EstData_Ls_d_pu:	Direct stator inductance value (in IQ30)
EstData_Ls_q_pu:	Stator inductance value in the quadrature coordinate direction (in IQ30)
EstData_Ls_max_pu:	Maximum stator inductance value from the stator inductance estimator
EstData_Ls_min_pu:	Minimum stator inductance value from the stator inductance estimator
EstData_Ls_coarse_max_pu:	Maximum stator inductance value during coarse estimation in the stator inductance estimator
EstData_MaxAccel_pu:	Maximum acceleration value used in the estimator

EstData_MaxAccel_est_pu:	Maximum estimation acceleration value used in the estimator
EstData_MaxCurrentSlope_pu:	Maximum current slope value used in the estimator
EstData_MaxCurrentSlope_epl_pu	Maximum EPL (Efficient Partial Load) current slope value used in the estimator
EstData_OneOverDcBus_pu:	Inverse of the dc bus voltage
EstData_PU_to_KRPM_sf:	pu to krpm scale factor
EstData_Rr_pu:	Rotor resistance value (in IQ30)
EstData_Rs_pu:	Stator resistance value (in IQ30)
EstData_RsOnLine_pu:	Online stator resistance value (in IQ30)
EstData_RsOnLineId_mag_pu:	Id magnitude value used for online stator resistance estimation
EstData_RsOnLineId_pu:	Online stator resistance value (in IQ30)
EstData_Speed_pu:	Mechanical frequency of the motor
EstData_Speed_krpm:	Speed value in krpm
EstData_Torque_lbin:	Torque value
EstData_Torque_Nm:	Torque value in N*m

The PIL InstaSPIN block requires additional parameters, and these parameters are defined in a parameter file called “InstaSPIN_params.txt”. The file name is hard coded, and a different file name cannot be used. This file must be in the same folder as the schematic file.

There are many parameters that need to be defined and calculated for InstaSPIN, based on motor parameters, inverter operating conditions, and how voltages and currents are sensed. To ease the process of preparing the parameter file, PSIM provides an **InstaSPIN Parameter Editor** to help users quickly generate the required “InstaSPIN_params.txt” file.

To generate the InstaSPIN parameter file, in PSIM, go to **Utilities >> InstaSPIN Parameter Editor**. The dialog window is shown below. For more information on how to use the Parameter Editor, click on the **Help** button.

The entry of the InstaSPIN Parameter Editor can be saved to a text file for later use. Sample files are provided in the “examples\SimCoder\F2806x Target” folder for the following hardware setup:

- “InstaSPIN_param_DRV8305EVM.txt” for TI DRV8305 EVM with Anaheim BLDC motor BLY172S-24V-4000
- “InstaSPIN_param_DRV8312EVM.txt” for TI DRV8312 EVM with Anaheim BLDC motor BLY172S-24V-4000
- “InstaSPIN_param_HV_PFCMT_kit.txt” for TI high-voltage PFC and Motor Control Kit with Estun PMSM motor EMJ-04APB22

To use the sample files, click on the **Load** button and load the file into the InstaSPIN Parameter Editor.

The dialog window below shows the entry for the TI DRV8312 EVM with Anaheim BLDC motor BLY172S-24V-4000. The dialog window is divided into several sections, and they are explained below.

InstaSPIN Parameter Editor

File: _____ Comment: _____

Motor

Motor Type: BLDC/PMSM Induction Motor

Back emf Coefficient	3.35	Rated Frequency	60
Stator Resistance Rs	0.405	Stator Resistance Rs	9.25
d-axis Inductance Ls_d	0.64m	Rotor Resistance Rr	7.15
q-axis Inductance Ls_q	0.64m	Stator Leakage Reactance Xls	9.08
Number of Poles	8	Rotor Leakage Reactance Xlr	4.28
Motor Max. Current	5	Magnetizing Reactance Xm	170
Motor Max. RPM	11000	No-load RMS Current	0.7
		Rated Line-to-line Voltage	230

Motor Identification

Inverter

DC Bus Voltage: 24 Switching Frequency (kHz): 20

DC Voltage Sensing

Sensing Selection: Divider User Defined

Resistance R1	95.3k	DC Sensing Gain	0.00805
Resistance R2	4.99k		

AC Voltage Sensing

Sensing Selection: RC Circuit User Defined

Resistance R1	95.3k	AC Voltage Sensing Filter Cut-off Frequency	375
Resistance R2	4.99k		
Capacitance C1	0.047u		

AC Current Sensing

Sensing Selection: User Defined DRV8305 EVM DRV8312 EVM

Shunt Resistance Rsense	10m	Reference Divider k	2
Current Gain	19.1	Resistance Rgain	50k

DSP Control

DSP Frequency (MHz)	90
No. of PWM Periods Per Interrupt	1
Ratio of InstaSPIN vs. ISR Rate	1
Ratio of InstaSPIN vs. Current Loop Rate	1
Ratio of InstaSPIN vs. Estimator Rate	1

InstaSPIN Parameters

USER_MAX_VS_MAG_PU	0.5
USER_R_OVER_L_EST_FREQ_Hz	100
USER_MOTOR_RES_EST_CURRENT	0.5
USER_MOTOR_IND_EST_CURRENT	-0.5
USER_MOTOR_FLUX_EST_FREQ_Hz	0.5
USER_MAX_ACCEL_EST_Hzps	2

For the *Motor* section:

This section defines the motor type and motor parameters. Two types of motors can be defined: BLDC/PMSM (including IPM and SPM), and induction motor. For the DRV8305 EVM kit with the Anaheim motor BLY172S-24V-4000, the following parameters are defined:

BLDC/PMSM box checked	
Back emf Coefficient Ke:	3.35 Vpk/krpm
Stator Resistance Rs:	0.405 Ohm
d-axis Inductance Ls_d:	0.64 mH
q-axis Inductance Ls_q:	0.64 mH
Number of Poles:	8
Motor Max. Current	5A

Motor Max. RPM: 11000 rpm

For the *Inverter* section:

This section defines the inverter dc bus voltage and switching frequency. For DRV8312 EVM, the following are defined:

DC Bus Voltage: 24 V
Switching Frequency (kHz): 20 kHz

For the *DC Voltage Sensing* section:

This section defines how the dc bus voltage is sensed. There are two ways to sense the voltage: either Voltage Divider or User Defined. With Voltage Divider, divider resistances need to be specified. With User Defined, the dc sensing gain is defined directly. For DRV8312 EVM, it uses a voltage divider, and based on the DRV8305 EVM schematic, the resistances are:

Divider box checked
Resistance R1: 95.3 kOhm
Resistance R2: 4.99 kOhm

For the *AC Voltage Sensing* section:

This section defines how the ac voltage is sensed. There are two ways to sense the voltage: either RC Circuit or User Defined. With RC Circuit, resistances R1 and R2 and capacitance C1 need to be specified. With User Defined, the ac voltage sensing filter cut-off frequency is defined directly. For DRV8312 EVM, it uses a RC circuit, and based on the DRV8312 EVM schematic, the values are:

RC Circuit box checked
Resistance R1: 95.3 kOhm
Resistance R2: 4.99 kOhm
Capacitance C1: 0.047 uF

For the *AC Current Sensing* section:

This section defines how the ac current is sensed. There are three choices: either User Defined, or DRV8305 EVM, or DRV8312 EVM.

For User Defined, the shunt resistance and current gain are defined. Note that when the shunt resistance is not 0, it is assumed that the current is measured through the shunt resistance, and the overall gain is equal to the shunt resistance multiplied by the current gain entered. If the shunt resistance is 0, it is assumed that the current is measured through a hall effect sensor, and the overall gain is the same as the current gain entered.

For DRV8305 EVM, the op. amp. circuit that it uses is fixed, but the reference can be changed through the Reference Divider k, and the gain can be changed through the resistance Rgain. Both can be set through SPI commands.

For DRV8312 EVM, the op. amp. Circuit that it uses is fixed, and the and the current gain is 19.1.

For this example, the DRV8312 EVM checkbox is checked.

For the *DSP Control* section:

This section defines the parameters related to DSP and InstaSPIN settings. The following parameters are defined:

DSP Frequency (MHz):	90 MHz
No. of PWM Periods Per Interrupt:	1
Ratio of InstaSPIN vs. ISR Rate:	1
Ratio of InstaSPIN vs. Current Loop Rate:	1
Ratio of InstaSPIN vs. Estimator Rate:	1

For more information on the parameter definition, refer to the Help page.

For the *InstaSPIN Parameters* section:

This section defines InstaSPIN parameters that can be set directly. Most of the parameters are used for motor identification. For this example, the only parameter that needs to be defined is:

USER_MAX_VS_MAG_PU:	0.5
---------------------	-----

For more information on the parameter definition, refer to the Help page.

Once above data are entered into their respective fields, click on the **Save** button to save the data to a text file for later retrieval using the **Load** button. To generate the InstaSPIN parameter file for PSIM, click on the **Generate** button and save the file “InstaSPIN_params.txt” to a desired location.

The generated InstaSPIN parameter file is shown below. For easy inspection, the parameters are divided into 4 sections: commonly changed parameters, rarely changed parameters, constants, and derived parameters.

```

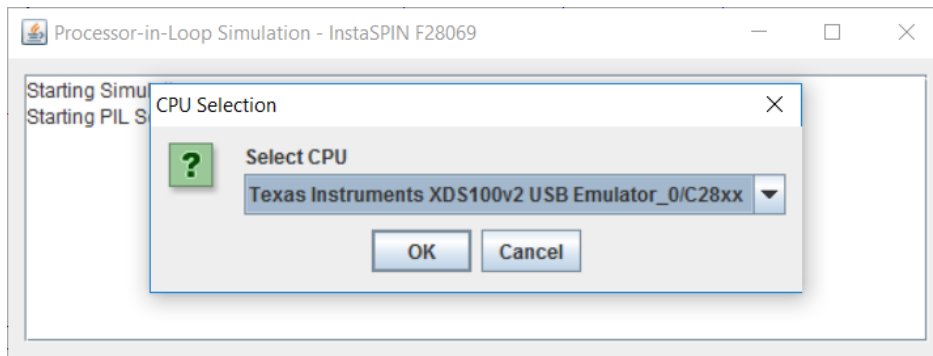
// InstaSPIN Parameters
// Commonly changed
//-----
USER_PWM_FREQ_kHz           = 20      // User defined
USER_NUM_PWM_TICKS_PER_ISR_TICK = 1      // User defined
USER_MOTOR_TYPE             = 1      // Motor dependent
USER_MOTOR_NUM_POLE_PAIRS   = 4      // Motor dependent
USER_MOTOR_RATED_FLUX       = 0.0290119 // Motor dependent
USER_MOTOR_Rr               = 0      // Motor dependent
USER_MOTOR_Rs               = 0.4051206 // Motor dependent
USER_MOTOR_Ls_d             = 0.000639871 // Motor dependent
USER_MOTOR_Ls_q             = 0.000639879 // Motor dependent
USER_MOTOR_MAX_CURRENT      = 5      // Motor dependent
USER_MOTOR_RES_EST_CURRENT  = 1      // Motor dependent
USER_MOTOR_IND_EST_CURRENT  = -1     // Motor dependent
USER_MOTOR_MAGNETIZING_CURRENT = 0    // Motor dependent
USER_MOTOR_FLUX_EST_FREQ_Hz = 20    // Motor dependent
//-----
// Rarely changed
//-----
USER_IQ_FULL_SCALE_CURRENT_A = 9      // Kit dependent
USER_IQ_FULL_SCALE_VOLTAGE_V = 24     // Kit dependent
USER_MAX_ACCEL_EST_Hzps     = 5      // Kit dependent
USER_ADC_FULL_SCALE_VOLTAGE_V = 66.324 // Kit dependent
USER_ADC_FULL_SCALE_CURRENT_A = 17.2775 // Kit dependent
USER_VOLTAGE_FILTER_POLE_Hz = 714.145 // Kit dependent
NUM_ISR_TICKS_PER_CTRL_TICK = 1      // User defined
NUM_CTRL_TICKS_PER_CURRENT_TICK = 1 // User defined
NUM_CTRL_TICKS_PER_EST_TICK = 1      // User defined
NUM_CTRL_TICKS_PER_SPEED_TICK = 15   // User defined
NUM_CTRL_TICKS_PER_TRAJ_TICK = 15    // User defined
USER_SYSTEM_FREQ_MHz        = 90     // DSP dependent
USER_NUM_CURRENT_SENSORS    = 3      // Hardware defined
USER_NUM_VOLTAGE_SENSORS    = 3      // Hardware defined
USER_IQ_FULL_SCALE_FREQ_Hz  = 806.667 // Motor dependent
USER_IDRATED_DELTA         = 0.00002 // Hardware dependent
USER_R_OVER_L_EST_FREQ_Hz  = 100    // Motor dependent
//-----
// Constants
//-----
MATH_PI_VALUE               = 3.1415926535897932384626433832795
USER_OFFSET_POLE_rps        = 20
USER_FLUX_POLE_rps          = 100
USER_MAX_ACCEL_Hzps         = 20
USER_MAX_VS_MAG_PU          = 0.5
USER_DIRECTION_POLE_rps     = 6.0
USER_SPEED_POLE_rps         = 100
USER_DCBUS_POLE_rps         = 100
USER_FLUX_FRACTION          = 1.0
SPEEDMAX_FRACTION_FOR_L_IDENT = 1.0
USER_POWERWARP_GAIN         = 1.0
USER_EST_KAPPAQ             = 1.5
IDRATED_FRACTION_FOR_L_IDENT = 1.0
IDRATED_FRACTION_FOR_RATED_FLUX = 1.0
//-----
// Derived variables
//-----
USER_ZEROSPEEDLIMIT         = 0.5 / USER_IQ_FULL_SCALE_FREQ_Hz
USER_FORCE_ANGLE_FREQ_Hz    = 2.0 * USER_ZEROSPEEDLIMIT * USER_IQ_FULL_SCALE_FREQ_Hz
USER_PWM_PERIOD_usec        = 1000.0 / USER_PWM_FREQ_kHz
USER_VOLTAGE_SF              = USER_ADC_FULL_SCALE_VOLTAGE_V / USER_IQ_FULL_SCALE_VOLTAGE_V
USER_CURRENT_SF              = USER_ADC_FULL_SCALE_CURRENT_A / USER_IQ_FULL_SCALE_CURRENT_A
USER_VOLTAGE_FILTER_POLE_rps = 2.0 * MATH_PI_VALUE * USER_VOLTAGE_FILTER_POLE_Hz
USER_MAX_VS_MAG_PU           = 2.0 / 3.0
USER_ISR_FREQ_Hz             = USER_PWM_FREQ_kHz * 1000.0 / USER_NUM_PWM_TICKS_PER_ISR_TICK
USER_CTRL_FREQ_Hz           = USER_ISR_FREQ_Hz / NUM_ISR_TICKS_PER_CTRL_TICK
USER_TRAJ_FREQ_Hz           = USER_CTRL_FREQ_Hz / NUM_CTRL_TICKS_PER_TRAJ_TICK
USER_EST_FREQ_Hz            = USER_CTRL_FREQ_Hz / NUM_CTRL_TICKS_PER_EST_TICK
USER_MAX_CURRENT_SLOPE      = USER_MOTOR_RES_EST_CURRENT / USER_IQ_FULL_SCALE_CURRENT_A / USER_TRAJ_FREQ_Hz
MAX_CURRENT_SLOPE_POWERWARP = 0.3 * USER_MOTOR_RES_EST_CURRENT / USER_IQ_FULL_SCALE_CURRENT_A / USER_TRAJ_FREQ_Hz
USER_ISR_PERIOD_usec        = USER_PWM_PERIOD_usec * USER_NUM_PWM_TICKS_PER_ISR_TICK
USER_CTRL_PERIOD_usec       = USER_ISR_PERIOD_usec * NUM_ISR_TICKS_PER_CTRL_TICK
USER_CTRL_PERIOD_sec        = USER_CTRL_PERIOD_usec / 1000000.0
MAX_NEGATIVE_ID_REF_CURRENT_A = -0.5 * USER_MOTOR_MAX_CURRENT

```

3. Simulation with PIL InstaSPIN Block

Below are the steps to simulate a circuit with the PIL InstaSPIN block:

- Place the block in the circuit and connect it with the rest of the circuit.
- Define the InstaSPIN parameters and other parameters. Obtain the InstaSPIN parameter file “InstaSPIN_params.txt” from the InstaSPIN Parameter Editor as described in the previous section.
- Connect the computer to the F28069M controlCARD. Note that the F28069 controlCARD must be the M version that supports InstaSPIN.
- Run the simulation. A dialog window as shown below will appear.



Select the correct CPU type, and click on OK to continue. The simulation will run. Note that in the middle of the simulation, do not close the “Processor-in-Loop Simulation” popup window.

An InstaSPIN example is provided in the folder “examples\PIL\DRV8312 PMSM InstaSPIN Lab11 (F28069)”. This example is based on Lab 11 of TI’s Motorware InstaSPIN examples for F28069 for the DRV8312 kit. The PSIM schematic of the example is shown below.

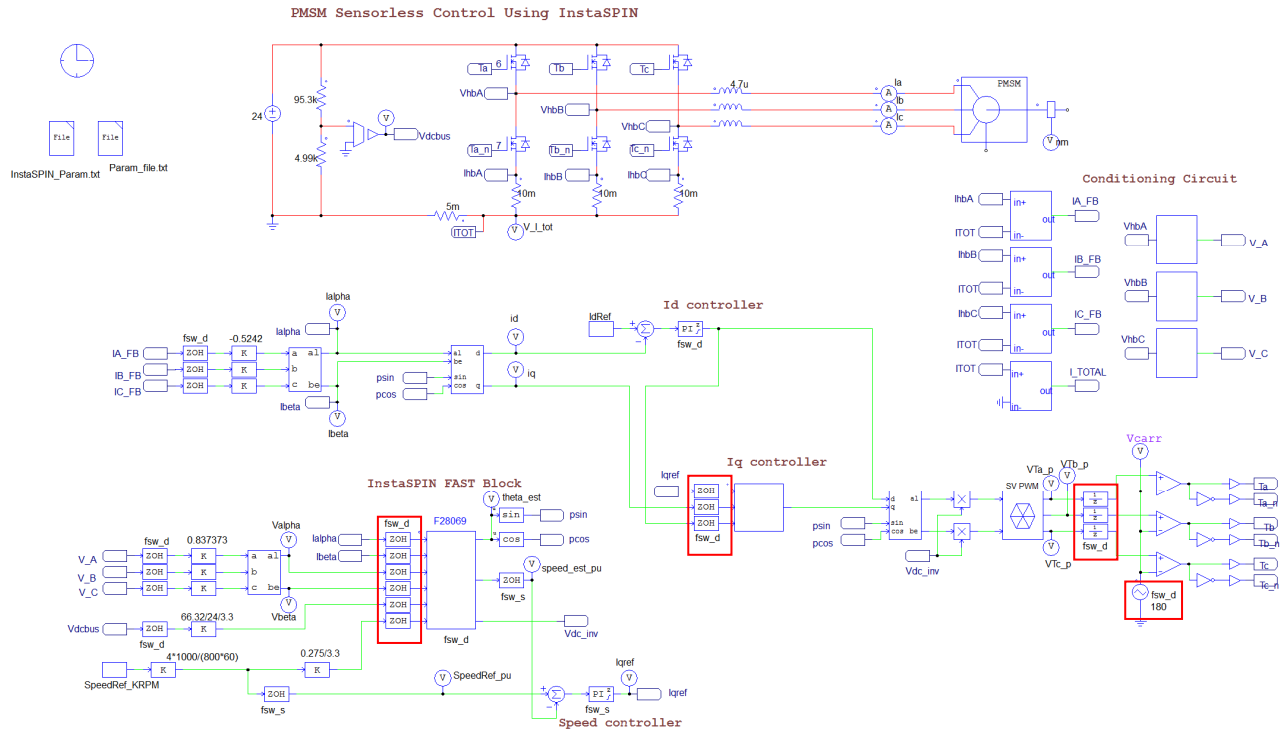
Notice that beside the parameter file “InstaSPIN_params.txt”, another parameter file “Param_file.txt” is used in the schematic to define parameters used in other part of the circuit. The file “InstaSPIN_params.txt” has a higher priority than the file “param_file.txt”, which means that parameters defined in “InstaSPIN_params.txt” can be used in “param_file.txt”. To define the file priority, in the parameter file dialog window, select **Edit >> Priority**.

In this example, 3-phase ac voltages and currents as well as dc bus voltage are measured. These measured voltages and currents are scaled as per unit quantities as shown below:

$$\text{Current scaling factor} = \text{USER_ADC_FULL_SCALE_CURRENT_A} / \text{USER_IQ_FULL_SCALE_CURRENT_A} / 3.3$$

$$\text{Voltage scaling factor} = \text{USER_ADC_FULL_SCALE_VOLTAGE_V} / \text{USER_IQ_FULL_SCALE_VOLTAGE_V} / 3.3$$

The scaled ac voltages and currents are converted to the alpha/beta frame through the Clarke transformation. The alpha/beta quantities, together with the dc bus voltage and speed reference, are sent to the PIL InstaSPIN block, which generates estimated rotor angle theta, estimated speed, and the inverse of the dc bus voltage.



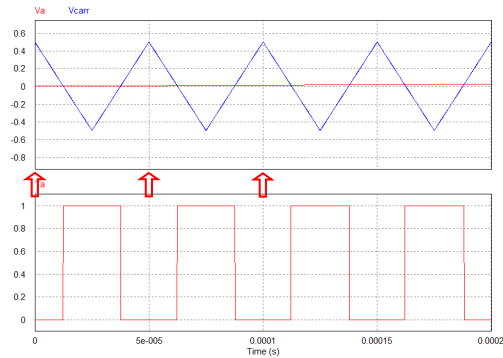
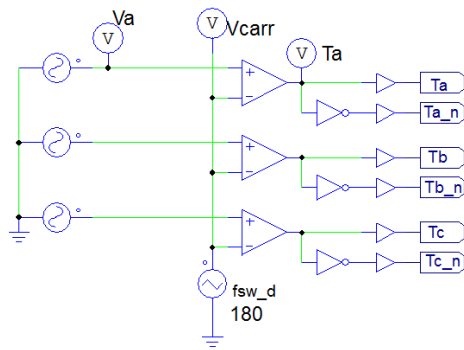
The estimated speed is used in the speed control loop to generate the Iq reference. The outputs of the Id and Iq control loops are sent to the inverse Park transformation block, and the outputs are then used to generate PWM gating signals.

There are two sampling rates in this circuit, fsw_d for the inner current loop (set to 20kHz in “InstaSPIN_params.txt”), and fsw_s for the outer speed loop (fsw_s = fsw_d/15).

Note that in this schematic, zero-order-hold blocks of fsw_d are connected to all inputs of the PIL InstaSPIN block and the Iq controller C block. This is needed to force these two blocks to compute in the sampling rate of fsw_d.

Three unit delay blocks at the input of the PWM generation circuit (highlighted in red) are used to model the one cycle delay inherent in digital control.

Also, the PWM carrier waveform phase delay must be set to 180 deg. This is very important as otherwise the circuit will not work. This is because the bottom switch currents are measured and sampled by the DSP ADC. This means that, at the moment of sampling, the bottom switch must be on and the top switch must be off. In order to achieve this, the carrier waveform must be phase shifted by 180 deg. To understand this, let’s look at the PWM circuit by itself, as shown below on the left:



The carrier waveform is a triangular waveform from -0.5 to 0.5, with the phase delay set to 180 deg. and a switching frequency of 20kHz. The simulation waveforms are shown on the right. In PSIM simulation, the beginning of sampling periods starts naturally at the time of zero and at each integer number of the sampling period. In this case, the beginning of the periods starts at 0, 50us, 100us, etc., as marked by the red arrow.

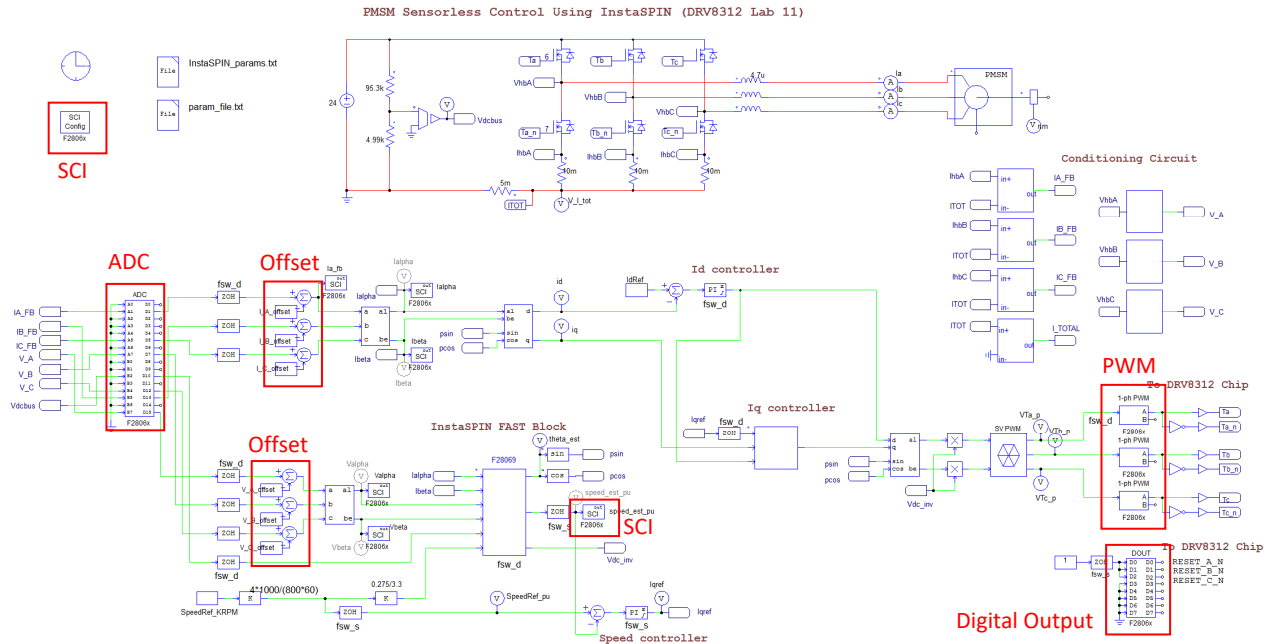
If A/D conversion is performed at the beginning of each period, with the settings above, the top switch gating signal T_a will be low, and the bottom switch gating signal T_{a_n} will be high. This is exactly what is needed for DSP implementation. If the phase delay of the carrier wave is set to 0 as is done typically, at the beginning of each period, T_a will be high and T_{a_n} will be low. The A/D conversion result for the current will be all 0 since the bottom switch is not conducting.

To summarize, the carrier wave phase delay should be set to 180 deg. if the bottom switch currents are sampled, and the phase delay should be set to 0 if the top switch currents are sampled.

4. Auto Code Generation with the PIL InstaSPIN Block

With SimCoder and the F2806x Target library, PSIM can automatically generate code that is ready to run on an InstaSPIN-enabled DSP hardware.

An example is provided in the folder “examples\SimCoder\F2806x Target\TI DRV8312 PMSM InstaSPIN Lab11”. This example is based on Lab 11 of TI’s Motorware InstaSPIN examples for F28069 for the DRV8312 kit. The PSIM schematic is shown below.



In the schematic, the zero-order-hold blocks at the inputs of the PIL InstaSPIN block and the Iq controller block (except the Iqref input) are no longer needed as SimCoder will detect the sampling rates of these blocks automatically.

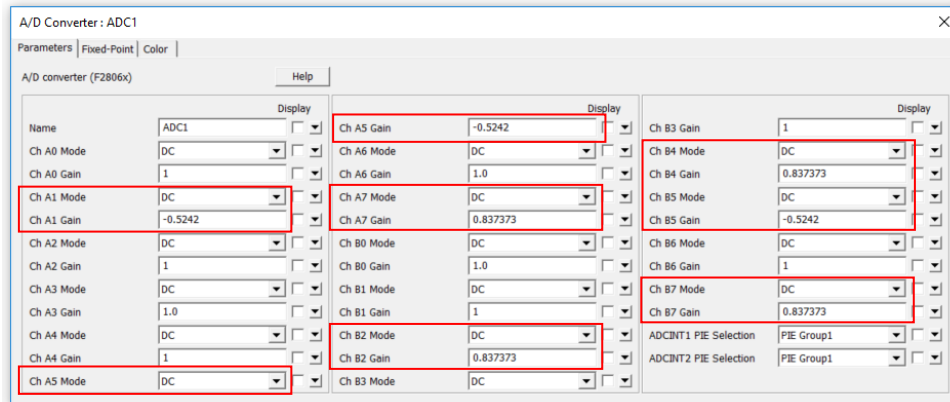
Also, since ac voltages and currents at the ADC inputs have dc offset due to conditioning circuits, to restore the voltages/currents back to ac, the dc offsets are removed after the ADC and ZOH blocks, as highlighted in red.

The offset values from the TI Lab 11 code are used here. Note that the offsets are not exactly equal among the three phases in the hardware circuit, but they are equal in simulation. To address the discrepancy, a flag called “flag_simulation” is defined in the parameter file “param_file.txt”. When the flag is set to 1, identical offset values are used. When the flag is 0, the actual hardware offset values are used.

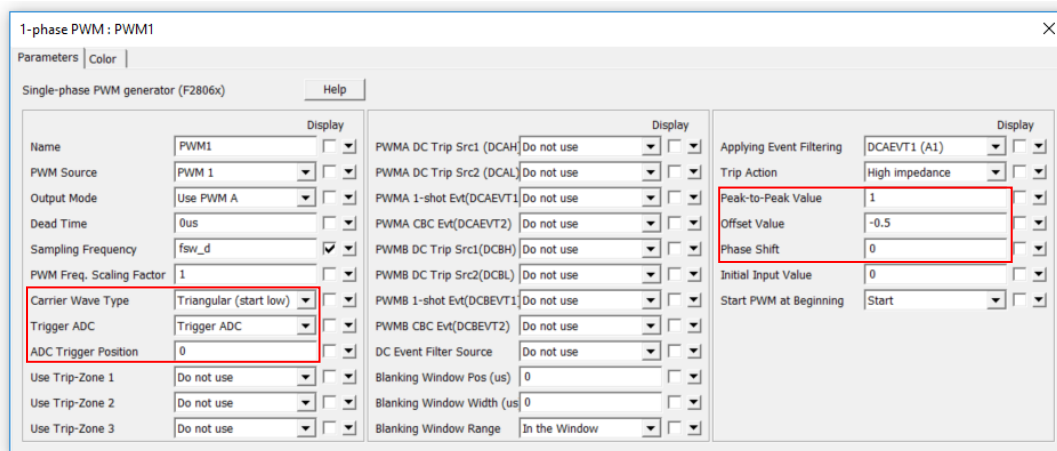
The current and voltage sensing circuits match the DRV8312 EVM hardware.

To perform auto code generation, ADC, PWM, SCI, and digital output blocks, as highlighted in red, are used in the circuit. They simulate the functions of the corresponding F28069 hardware peripheral blocks. The SCI blocks are used to display DSP waveforms in real time.

The figure below shows the ADC block definition. The modes of all the ADC channels are set to DC as all incoming voltage and current signals are dc quantities.



The figure below shows the PWM block definition for the Phase A PWM signal Ta. The definitions of other two phases are the same except that the parameter “Trigger ADC” is set to *Do not trigger ADC*.

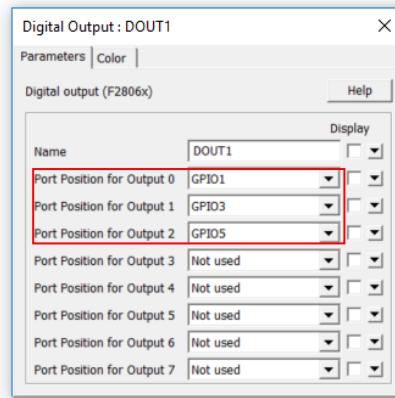


In the definition, the parameters “Carrier Wave Type” is set to *Triangular (start low)*, “Trigger ADC” is set to *Trigger ADC*, and “ADC Trigger Position” is set to 0. This means that the PWM generator will trigger ADC, and ADC will perform A/D conversion at the beginning of the sampling period. The setting *Triangular (start low)* means that when the modulation wave is greater than the carrier wave, the PWM signal will be low. This will give a low signal to the top switch and a high signal to the bottom switch, exactly what is required for bottom switch current sampling.

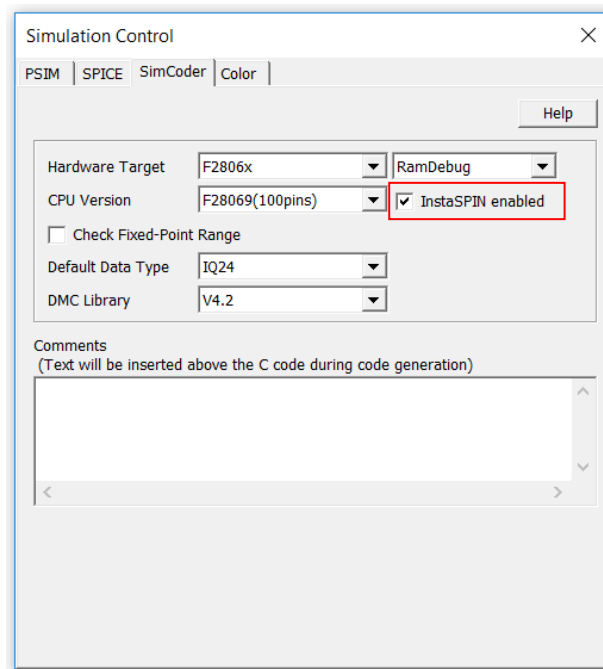
The carrier waveform is defined to be a triangular waveform from -0.5 to +0.5.

For the DRV8312 chip, it needs not only 3 PWM signals PWMA, PWMB, and PWMC for the top switches, but also 3 reset signals RESET_A_N, RESET_B_N, and RESET_C_N which should be set to high all the time. These three reset signals are generated through the digital output block. Since it does not have to run at the frequency of fsw_d, it is set to run at a slower rate of fsw_s.

The definition of the digital output block is shown below. GPIO ports GPIO1, 3, and 5 are used as required by the DRV8312 hardware.



Please note that when generating code for InstaSPIN-enabled DSP hardware, in **Simulation Control** >> **SimCoder**, the checkbox *InstaSPIN enabled* must be checked, as shown below. Otherwise the generated code will be incorrect.



To run simulation, select **Simulate** >> **Run Simulation** in the same way as with other circuits.

To generate code automatically, select **Simulate** >> **Generate Code**. This will generate the code that is ready to run on the DRV8312 hardware kit.

Launch TI CCS, import the generated project by selecting **Project** >> **Import Legacy CCSv3.3 Projects**, and navigate to the subfolder “DRV8312 PMSM InstaSPIM Lab11 (F28069) (C code)”. Note that importing legacy CCSv3.3 project needs to be done only the first time. Once the CCSv3.3 project is converted, use **Project** >> **Import CCS Projects** to load the project the next time.

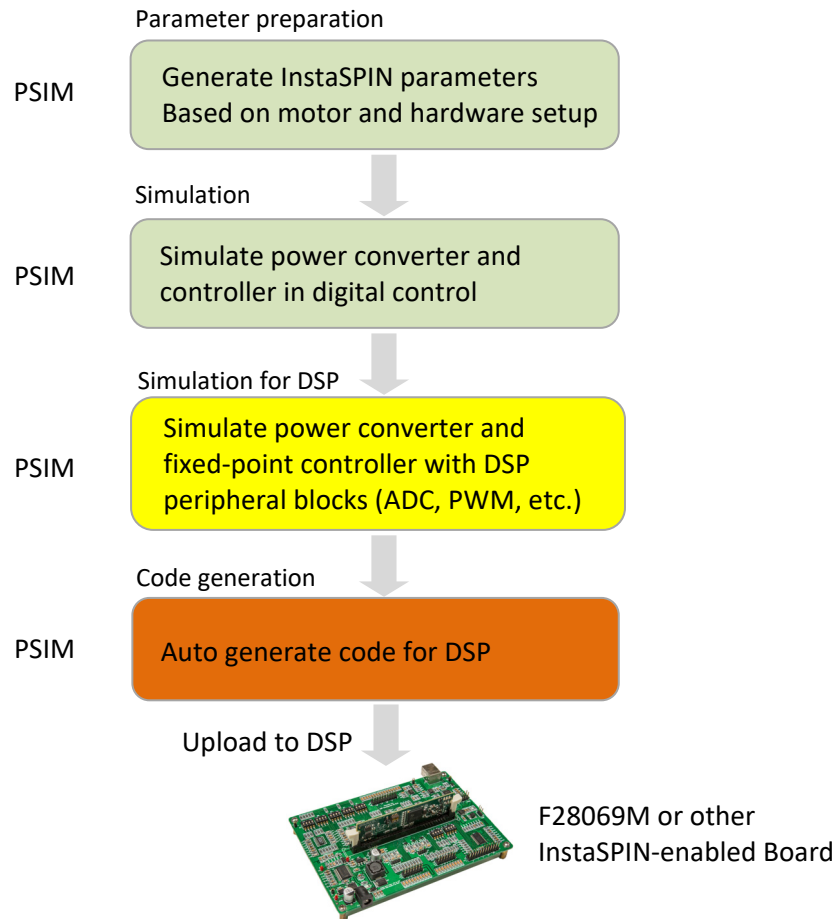
In CCS, build the project. Select **Run** >> **Debug** to upload the code to DSP. Click on the Resume icon on the toolbar to run the code, and the motor should start to rotate.

Once the code is running, in PSIM, select **Utilities >> DSP Oscilloscope**, and display waveforms of *la_fb*, *lalpha*, *lbeta*, *Valpha*, *Vbeta*, and *speed_est_pu* in real time.

For more information on how to generate code and run on F2806x DSP, please refer to the tutorial “Tutorial – Auto code generation for F2806x Target.pdf”.

For more information on how to use SCI for real-time waveform display, refer to the tutorial “Tutorial – Using SCI for waveform monitoring.pdf”.

With PSIM’s automatic code generation capability, one can perform both simulation and auto code generation for rapid control prototyping and hardware implementation in one seamless workflow, as shown below:



The integrated environment greatly speeds up the development process and helps to reduce development cost and time-to-market.