Sensorless Trapezoidal Control of BLDC Motors using BEMF Integration (InstaSPIN^{TM}-BLDC)

Jon Warriner D3 Engineering

Abstract

This application note presents a solution for sensorless control of Brushless DC motors using the TMS320F2803x microcontrollers. TMS320F280x devices are part of the family of C2000 microcontrollers which enable cost-effective design of intelligent controllers for three phase motors by reducing the system components and increasing efficiency. Using these devices, it is possible to realize precise control algorithms.

This application note covers the following:

- Incremental build levels based on modular software blocks.
- Experimental results

Table of Contents

Version 1.0 – Aug 2011

System Overview

This document describes the "C" real-time control framework used to demonstrate the trapezoidal control of BLDC motors. The "C" framework is designed to run on TMS320C2803x based controllers on Code Composer Studio. The framework uses the following modules[1]:

Macro Names	Explanation
BLDCPWM / PWMDAC	PWM and PWMDAC Drives
InstaSPIN™-BLDC	InstaSPIN-BLDC Library Functions
PID_GRANDO	PID Regulators
RC	Ramp Controller (slew rate limiter)
RC3	Ramp down Module
SPEED_PR	Speed Measurement (based on sensor signal period)
IMPULSE	Impulse Generator
MOD6_CNT_DIR	Mod 6 Counter with direction control
[1] Please refer to pdf documents in motor control folder explaining the details and theoretical background of each macro	

In this system, the sensorless trapezoidal control of BLDC motors will be experimented with and will explore the performance of the speed controller. The BLDC motor is driven by a DRV830x Three Phase PWM Motor Driver. The TMS320F2803x control card is used to generate three pulse width modulation (PWM) signals. The motor is driven by an integrated power module by means of BLDC specific PWM technique. Phase voltages and DC bus return current (Ifb Ret) is measured and sent to the TMS320x2803x via analog-to-digital converters (ADCs).

InstaSPIN_BLDC project has the following properties:

C Framework		
System Name	Program Memory Usage 2803x	Data Memory Usage[1] 2803x
InstaSPIN_BLDC	4597 words[2]	1200 words

[1] Excluding the stack size
[2] Excluding "IQmath" Look-up Tables

CPU Utilization of Trapezoidal BLDC Control (Sensorless)	
Name of Modules *	**Number of Cycles**
BLDCPWM	105
InstaSPIN™-BLDC Library	277
PID	91
RC	29
RC3	26 √
SPEED_PR	42
IMPULSE	17 √
MOD6_CNT_DIR	9
Contxt Save, Virtual Timer etc.	153
Pwm Dac (optional)	
DataLog (optional)	
Total Number of Cycles	**749 ****
CPU Utilization @ 60 Mhz	**25%**
CPU Utilization @ 40 Mhz	**37.4%**

* The modules are defined in the header files as "macros"
** At **20 kHz** ISR freq.
√ Not included in the speed loop

System Features	
Development /Emulation	Code Composer Studio v4.1 (or above) with Real Time debugging
Target Controller	TMS320F2803x
PWM Frequency	20kHz PWM (Default), 60kHz PWMDAC
PWM Mode	Symmetrical with 4 quadrant switching and programmable dead-band.
Interrupts	ADCINT1 EOC
Peripherals Used	PWM 1 / 2 / 3 for motor control PWM 5A, 6A, 6B & 4A for DAC outputs ADC A2 for low side DC bus return current sensing, B7, A7 and B4 for Bemf sensing SPI-B for communication and configuration of the DRV8301 (DRV8302 uses discrete digital and analog I/O for configuration)

The overall system implementing a 3-ph sensorless BLDC control is depicted in Figure 1

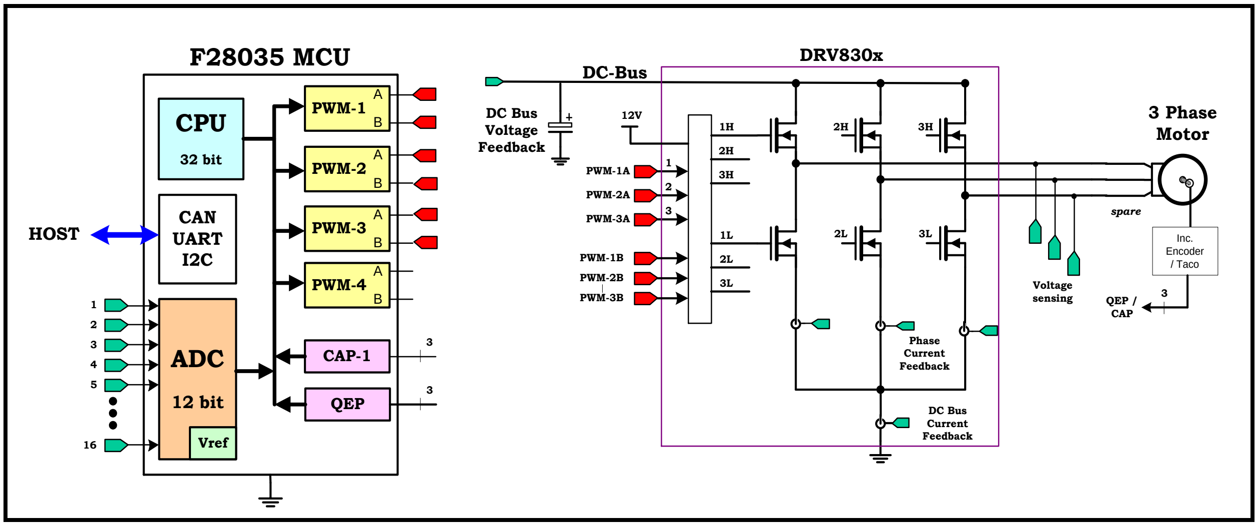


Figure 1 A 3-ph BLDC drive implementation

The software flow is described in the Figure 2 below.

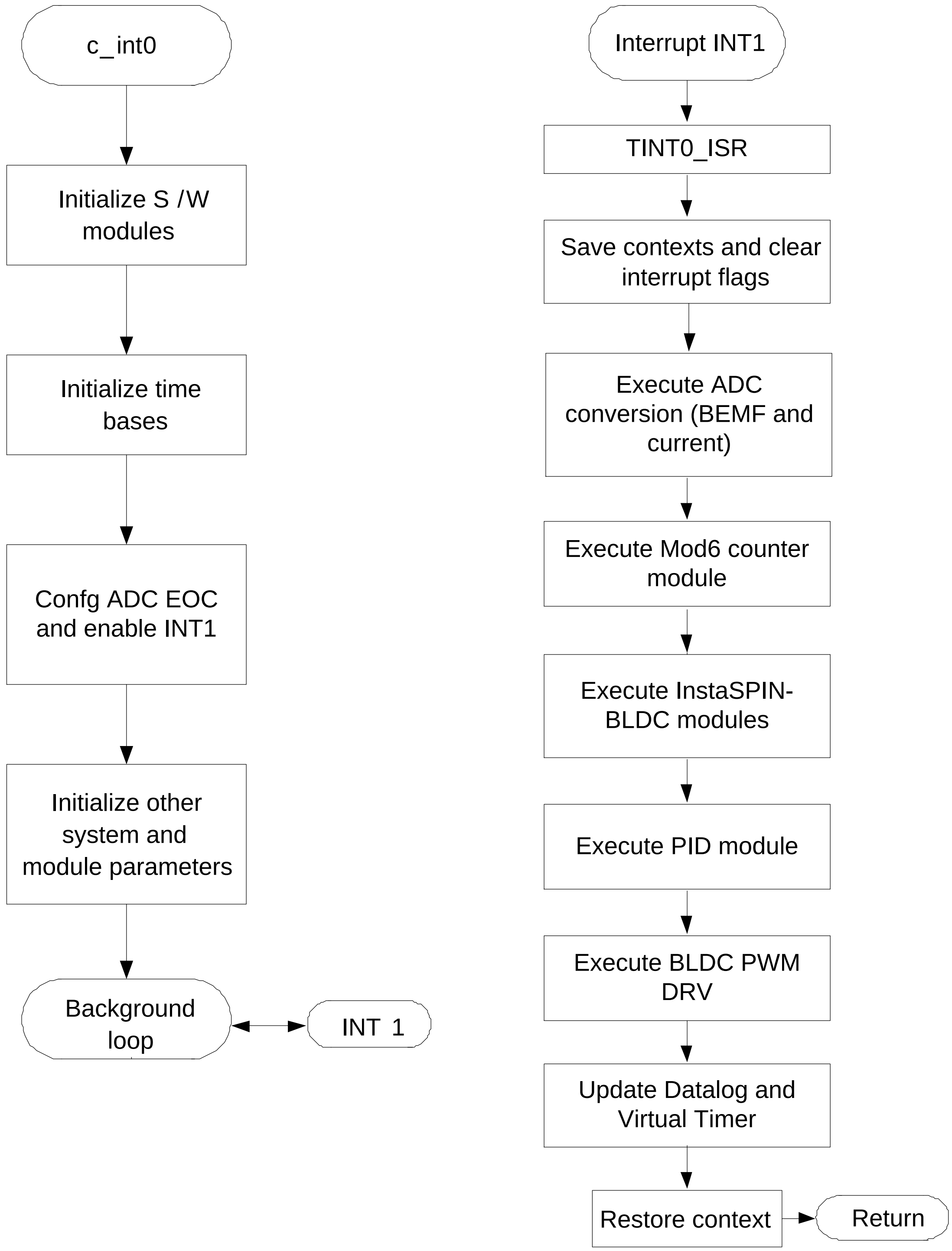


Figure 2 Software Flowchart

For reference the pictures below show the jumper and connectors that need to be connected for this lab.

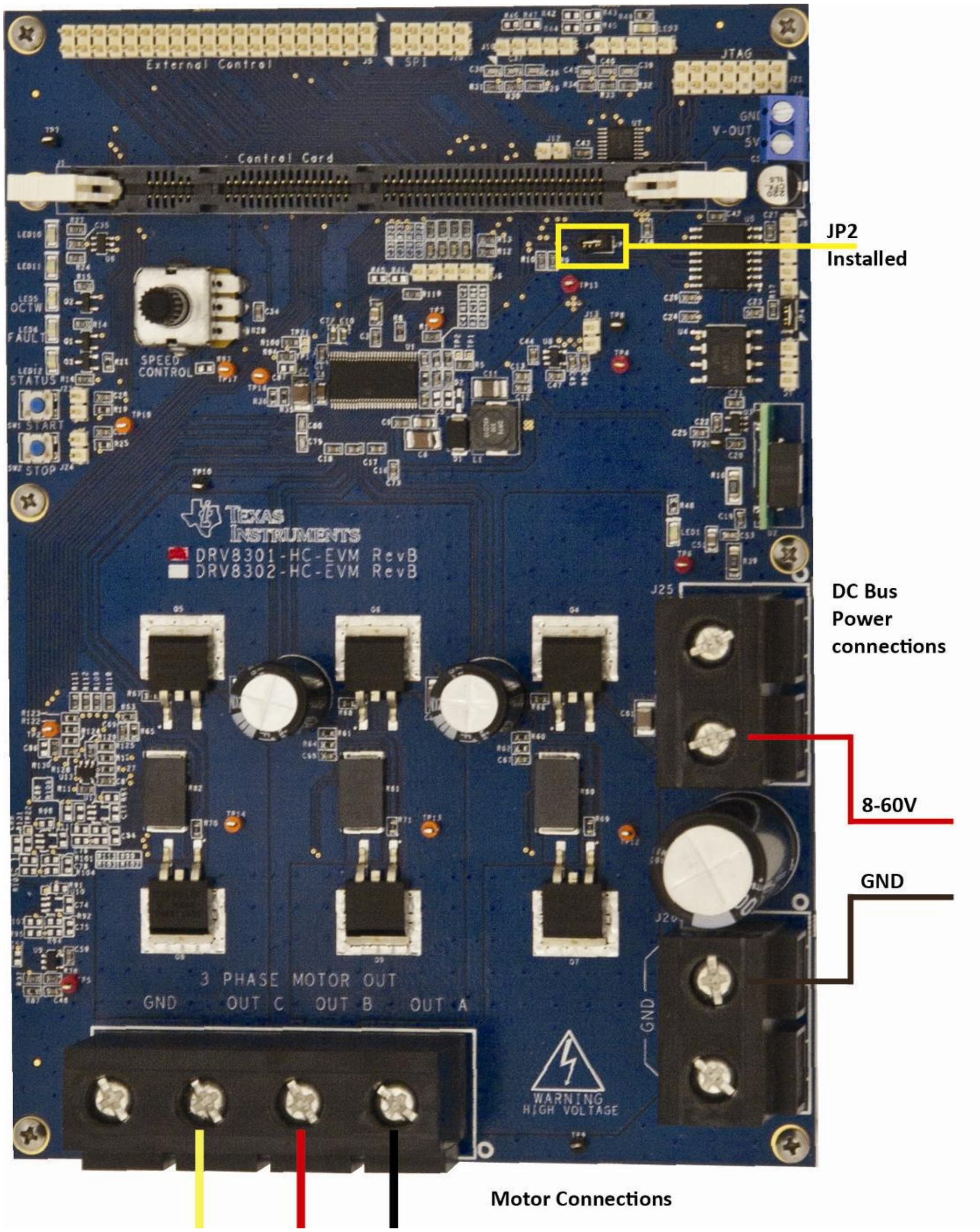


Figure 4 DRV830x-HC-EVM Connections and Settings

CAUTION: The inverter bus capacitors remain charged for a long time after the high power line supply is switched off/disconnected. Proceed with caution!

Software Setup Instructions to Run InstaSPIN_BLDC Project

Please refer to the "Generic Steps for Software Setup for DRV830x-HC-C2-KIT Projects" section in the DRV830x-HC-EVM How To Run Guide

C:\TI\controlSUITE\development_kits\DRV830x-HC-C2-KITv*\~Docs

This section goes over how to install CCS and set it up to run with this project.

The remainder of this application note will specifically discuss a hardware configuration consisting of a DRV8301-HC-EVM with a TMS320F2803x controlCARD installed. The process for other configurations, such as a DRV8302-HC-EVM, would be similar except the corresponding build configuration would need to be chosen in Code Composer Studio.

The default configuration of this project is optimized for running low to medium current motors. The gain of the DRV830x built-in current sense amplifiers is set to 40 giving a measurable current range of +/-20.625A. The gain can be changed by choosing the desired #define for DRV_GAIN in the file BLDC_Int-Settings.h. Note that there are four possible settings for the DRV8301 while the DRV8302 is limited to gains of 10 or 40.

Select the InstaSPIN_BLDC as the active project. Verify that the build level is set to 1, and then right click on the project name and select "Rebuild Project". Once build completes, launch a debug session to load the code into the controller. Now open a watch window and add the variables shown in the table below and select the appropriate Q format for them.

Watch (1)

Name	Value	Address	Type	Format
(x)= EnableFlag	1	0x00009B4F@Data	unsigned int	Natural
(x)= IsrTicker	0	0x00009B5C@Data	unsigned long	Natural
(x)= dlog.prescalar	1	0x00009C0B@Data	int	Natural
(x)= speed1.Speed	0.0	0x00009BF8@Data	long	Q-Value(24)
(x)= speed1.SpeedRpm	0	0x00009BFC@Data	long	Natural
(x)= DfuncTesting	0.3000000119	0x00009B5A@Data	long	Q-Value(24)
(x)= DfuncStartup	0.09999996424	0x00009B56@Data	long	Q-Value(24)
(x)= BemfA_offset	0.006999969482	0x00009B70@Data	long	Q-Value(24)
(x)= BemfB_offset	0.007199943066	0x00009B6A@Data	long	Q-Value(24)
(x)= BemfC_offset	0.006299972534	0x00009B6C@Data	long	Q-Value(24)
(x)= SpeedRef	0.3000000119	0x00009B64@Data	long	Q-Value(24)
(x)= IRef	0.01999998093	0x00009B6E@Data	long	Q-Value(24)
<new>				

Figure 5 Watch Window Setup

Setup time graph windows by importing Graph1.graphProp and Graph2.graphProp from the following location C:\TI\ControlSUITE\developement_kits\DRV830x-HC-C2-KITv*\InstaSPIN_BLDC Click on Continuous Refresh button on the top left corner of the graph tab to enable periodic capture of data from the microcontroller.

Incremental System Build for InstaSPIN™-BLDC project

The system is gradually built up in order so that the final system can be confidently operated. Eight phases of the incremental system build are designed to verify the major software modules used in the system. The table below summarizes the modules testing and using in each incremental system build.

Testing modules in each incremental system build								
Software Module	**Phase 1**	**Phase 2**	**Phase 3**	**Phase 4**	**Phase 5**	**Phase 6**	**Phase 7**	**Phase 8**
PWMDAC_MACRO	√√	√√		√	√	√	√	√
RC3_MACRO	√√	√		√	√	√	√	√
MOD6_CNT_DIR_MACRO	√√	√		√	√	√	√	√
IMPULSE_MACRO	√√	√		√	√	√	√	√
BLDCPWM_MACRO	√√	√		√	√	√	√	√
ADC Offset Calibration			√√	√	√	√	√	√
InstaSPIN™-BLDC Lib				√√	√	√	√	√
SPEED_PR_MACRO				√√	√	√	√	√
PID_MACRO (IDC)					√√	√		√
RC_MACRO					√			
PID_MACRO (SPD)							√√	√
Note: the symbol √ means this module is using and the symbol √√ means this module is testing in this phase.								

Level 1 Incremental Build

Assuming the load and build steps described in the “DRV830x-HC-C2-KIT How To Run Guide” completed successfully, this section describes the steps for a “minimum” system check-out which confirms operation of system interrupts, some peripheral & target independent modules and one peripheral dependent module. Open BLDC_Int-Settings.h and select level 1 incremental build option by setting the BUILDLEVEL to LEVEL1 (#define BUILDLEVEL LEVEL1). Now Right Click on the project name and click Rebuild Project. Once the build is complete click on debug button, reset CPU, restart, enable real time mode and run. Set “EnableFlag” to 1 in the watch window. The variable named “IsrTicker” will be incrementally increased as seen in watch windows to confirm the interrupt working properly.
In the software, the key variables to be adjusted are summarized below.

- RampDelay (Q0 format): for changing the ramping time.
- CmtnPeriodTarget (Q0 format): for changing the targeted commutation interval.
- CmtnPeriodSetpt (Q0 format): for changing the initial startup commutation interval.
- DfuncStartup: for changing the PWM duty cycle in per-unit.

The key explanations and steps are given as follows:

- The start-up and the initial speed up of the BLDC motor is controlled by the RMP3CNTL module. This module generates a ramp down function. This ramp down feature of RMP3CNTL module allows speed up of the BLDC motor from stand still in an open loop configuration (like a stepper motor).
- One of the inputs to RMP3CNTL module, DesiredInput, determines the final speed at the end of the motor speed up phase. This input is provided from the system using the system variable CmtnPeriodTarget. User initializes this system variable with appropriate value depending on the type of the BLDC motor. The second input to RMP3CNTL module is rmp3_dly, which is also user initialized by using the system variable RampDelay. This determines the rate at which the motor speeds up. The output of RMP3CNTL module is Out, which provides a variable time period gradually decreasing in time. The Out terminal is initialized by the system variable CmtnPeriodSetpt which sets the initial startup speed of the motor. CmtnPeriodTarget and CmtnPeriodSetpt are both initialized by the #defnes for RAMP_END_RATE and RAMP_START_RATE respectively. These #defines are located in BLDC_Int-Settings.h and set the initial and final speed of the startup ramp. The #defines allow these quantities to be entered in units of RPM. The second output of RMP3CNTL module is Ramp3DoneFlag, which, when set to 0x7FFF, indicates the end of the ramp down (or motor speed up) phase.
- Out is used to provide the input Period for the IMPULSE module. This module generates periodic impulses with period specified by its input Period.
- The DATALOG module is used to view the output variables of the modules. The initialization required to perform this, is done in the level 1 incremental build initialization routine. During this initialization, one of the inputs of DATALOG module is configured to point to mod1.Counter. Thus Out signal is shown in the graph in CCS.
- The periodic impulse output, Out, is applied to the input TrigInput of the MOD6_CNT module. The output of this module is Counter, which can assume one of the 6 possible values 0, 1, 2, 3, 4 or 5. This output changes from one state to the next when a trigger pulse is applied to the input. This Counter is finally used as the pointer input, CmtnPointer, for the module BLDC_3PWM_DRV. These 6 values of the pointer variable, CmtnPointer , are used to generate the 6 commutation states of the power inverter driving the BLDC motor. The duty cycle of the generated PWM outputs (according to the 6 commutation states) during the motor speed up phase are determined by the input DfuncStartup.

- Now, compile/load/run program with real time mode and set “EnableFlag” to 1 in the watch window. Initially when RMP3CNTL ramps down, Period (the period of Out) will also gradually go down. At the end of ramp period (when Out equals DesiredInput) Period will become constant and Ramp3DoneFlag will set to 0x7FFF.
- Check MOD6_CNT_DIR output variable Counter in the watch window and graph window. This will vary between 0 and 5.
- Use a scope to check the PWM outputs controlled by the peripheral dependent module BLDC_3PWM_DRV.
- The output states of all the 6 PWM outputs will be such that together they generate the 6 commutation states of the power inverter driving the BLDC motor.
- After verifying this take the controller out of real time mode (disable), reset the processor and then terminate the debug session.

While running this level, the Graph windows should look similar to Figure 6

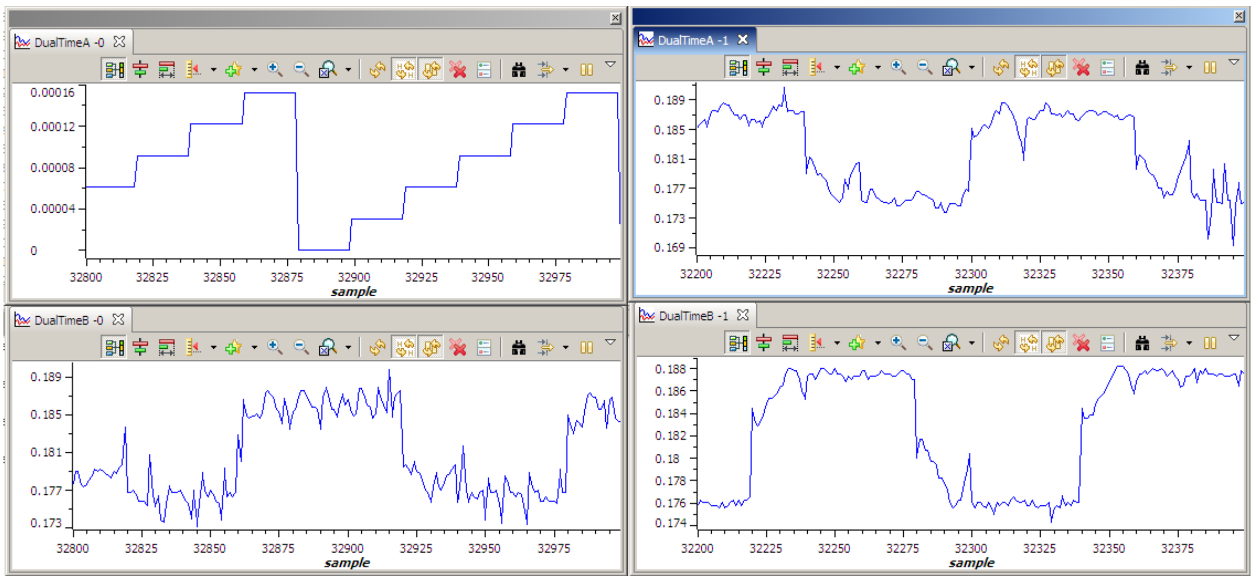


Figure 6 Graph Windows for Build Level 1 (a) mod6 counter, (b) BemfA, (c) BemfB and (d)BemfC

While running this level, the PWM outputs should appear as in Figure 7

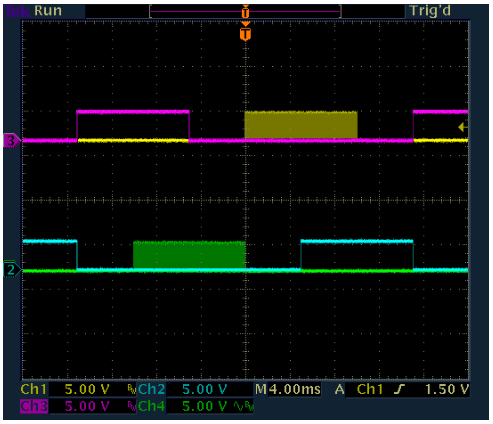


Figure 7 The PWM outputs , PWM 1 (Yellow) , PWM 2 (Pink) and PWM 5 (Green), PWM 6 (Blue)

Level 6 Incremental Build

Assuming the previous section is completed successfully, this section verifies the closed current loop and current PI controller. Open BLDC_Int-Settings.h and select level 6 incremental build option by setting the BUILDLEVEL to LEVEL6 (#define BUILDLEVEL LEVEL6). Now Right Click on the project name and click Rebuild Project. Once the build is complete click on debug button, reset CPU, restart, enable real time mode and run. Set “EnableFlag” to 1 in the watch window. The variable named “IsrTicker” will be incrementally increased as seen in watch windows to confirm the interrupt working properly.

In the software, the key variables to be adjusted are summarized below.

- RampDelay (Q0 format): for changing the ramping time.
- CmtnPeriodTarget (Q0 format): for changing the targeted commutation interval.
- CmtnPeriodSetpt (Q0 format): for changing the initial startup commutation interval.
- CurrentStartup: for changing the startup current in per-unit.
- IRef: changing the running current in per-unit.
- InstaSPIN_BLDC1.Int_Threshold: for changing the BEMF integration threshold in per-unit

The steps are explained as follows:

- Compile/load/run program with real time mode.

- The motor will gradually speed up and finally switch to closed loop commutation mode.

- Now use the variable IRef to specify the reference current for the PI controller. The PI controller will start to regulate the DC bus current and hence the motor current. Gradually increase/decrease the command current (IRef value) to change the torque command and adjust PI gains. Note that the speed is not controlled in this step and a non-zero torque reference will keep increasing the motor speed. Therefore, the motor should be loaded using a brake/generator (or manually if the motor is small enough) after closing the loop. Initially apply relatively light load and then gradually increase the amount of the load. If the applied load is higher than the torque reference, the motor cannot handle the load and stops immediately after closing the current loop.

- Verify the motor speed (both pu and rpm) calculated by SPEED_PR. View the following variables in the Watch Window.

    - speed1.Speed (pu)

    - speed1.SpeedRpm (rpm)

- Bring the system to a safe stop as described below by setting EnableFlag to 0, taking the controller out of realtime mode and reset.

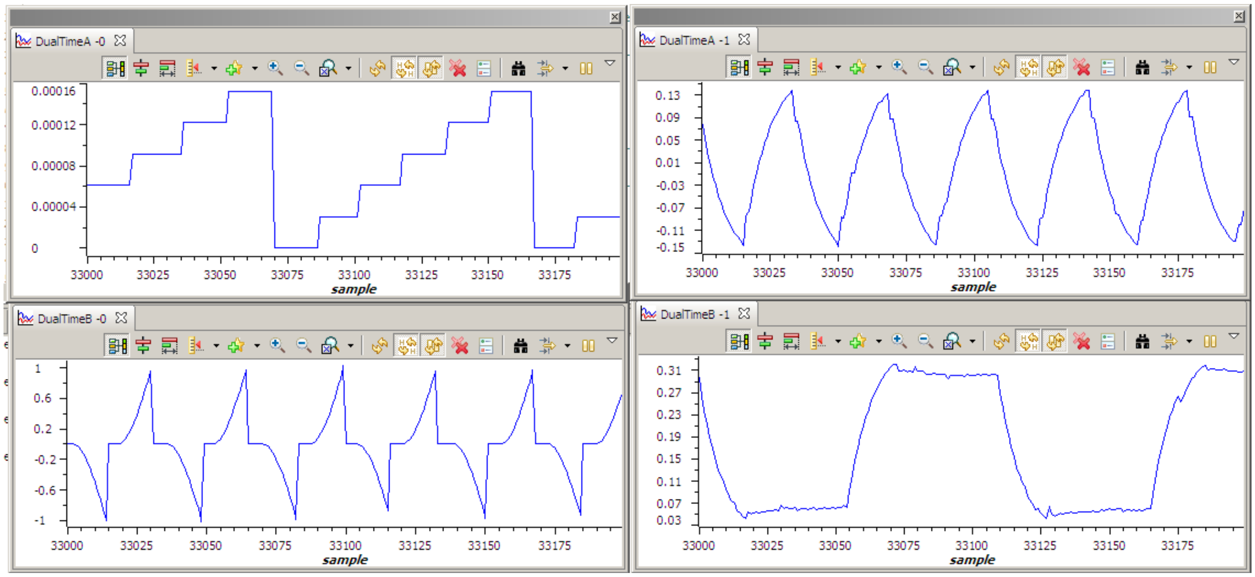


Figure 11 Graph Windows for Build Level 6 (a) mod6 counter, (b) V_int, (c) Vphase and (d)Vag

Level 8 Incremental Build

Assuming the previous section is completed successfully, this section verifies the cascaded closed speed and current loops. Open BLDC_Int-Settings.h and select level 8 incremental build option by setting the BUILDLEVEL to LEVEL8 (#define BUILDLEVEL LEVEL8). Now Right Click on the project name and click Rebuild Project. Once the build is complete click on debug button, reset CPU, restart, enable real time mode and run. Set "EnableFlag" to 1 in the watch window. The variable named "IsrTicker" will be incrementally increased as seen in watch windows to confirm the interrupt working properly.

In the software, the key variables to be adjusted are summarized below.

- SpeedRef (GLOBAL_Q format): for changing the reference Speed in per-unit.

The steps are explained as follows:

- Compile/load/run program with real time mode.
- The motor will gradually speed up and finally switch to closed loop commutation mode.
- Now use the variable SpeedRef to specify the reference speed for the PI controller PID_REG3. The SpeedLoopFlag is automatically activated when the PI reference is ramped up from zero speed to SpeedRef. Once this is done, the PI controller will start to regulate the motor speed. Gradually increase the command speed (SpeedRef value) to increase the motor speed.
- Adjust speed PI gains to obtain the satisfied speed responses, if needed.
- Bring the system to a safe stop as described below by setting EnableFlag to 0, taking the controller out of realtime mode and reset.

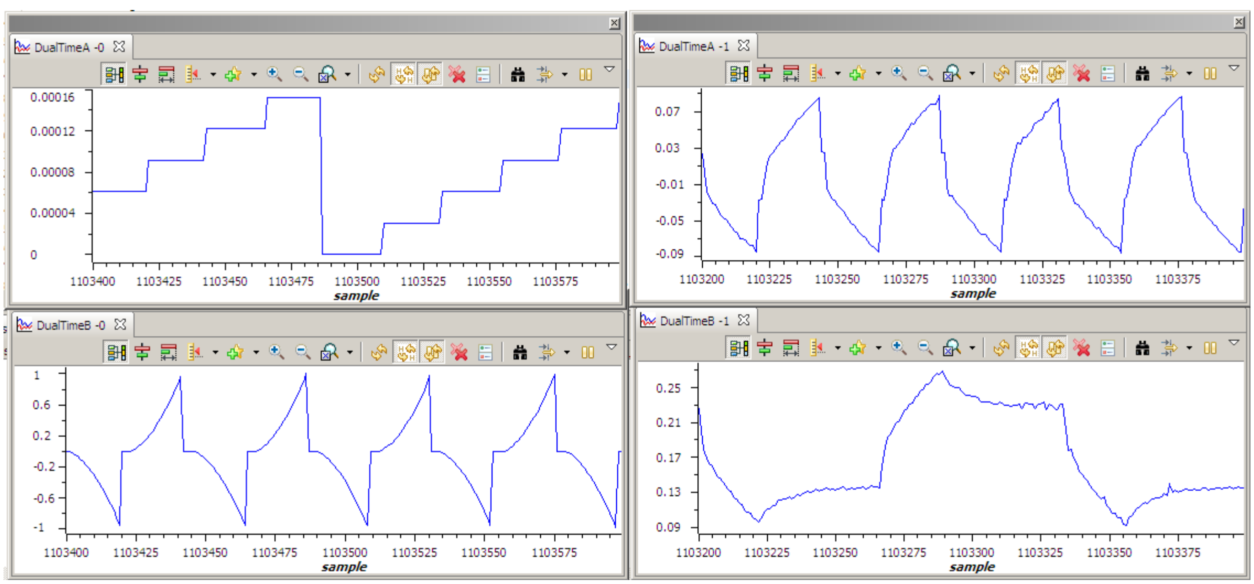
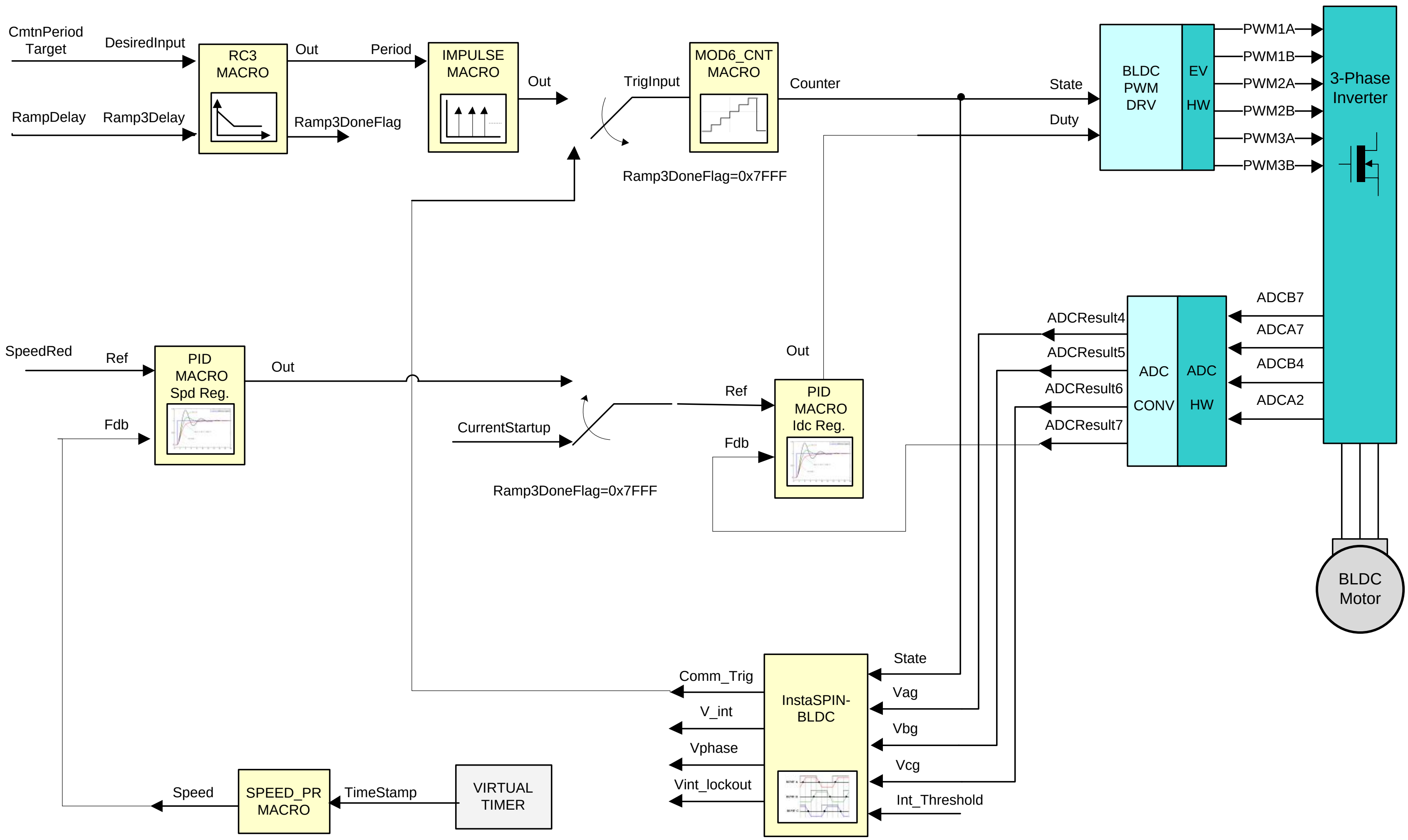


Figure 13 Graph Windows for Build Level 8 (a) mod6 counter, (b) V_int, (c) Vphase and (d)Vag

Level 8 Incremental System Build Block Diagram



Level 8 verifies the cascaded closed speed and current loops.