Application Report
SPRABQ6–July 2013

Trapezoidal Control of BLDC Motors Using Hall Effect Sensors

Bilal Akin and Manish Bhardwaj

ABSTRACT

This application report presents a solution for the control of brushless DC motors using the TMS320F2803x microcontrollers. TMS320F280x devices are part of the C2000™ family of microcontrollers that enable the 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 far more precise control algorithms. A complete solution proposal is presented below: control structures, power hardware topology, control hardware and remarks on energy conversion efficiency can be found in this document.

This application report covers the following:

- A theoretical background on field oriented motor control principle
- Incremental build levels based on modular software blocks
- Experimental results

Contents

1 Introduction 2
2 BLDC Motors 3
3 BLDC Motor Control 4
4 System Topology 6
5 Benefits of 32-Bit C2000 Controllers for Digital Motor Control (DMC) 9
6 TI Literature and Digital Motor Control (DMC) Library 9
7 Hardware Configuration (HVDMC R1.1 Kit) 14
8 Incremental System Build for Sensored BLDC Project 17
9 References 34

List of Figures

1 A Three-Phase Synchronous Motor With a One Permanent Magnet Pair Pole Rotor 3
2 Speed and Current Control Loop Configurations for a BLDC Motor 5
3 Electrical Waveforms in the Two Phase ON Operation and Torque Ripple 6
4 Torque Ripple in a Sinusoidal Motor Controlled as a BLDC 6
5 Three Phase Inverter 6
6 Shunt Resistor Voltage Drop According to PWM Duty Cycles (Soft Chopping) 8
7 A 3-ph BLDC Drive Implementation 11
8 Overall Block Diagram of Hall-Sensor Control of BLDC Motor 12
9 Software Flow 13
10 Using AC Power to Generate DC Bus Power 15
11 Using External DC Power Supply to Generate DC-Bus for the Inverter 16
12 The PWM Outputs: PWM 1 (Yellow), PWM 2 (Pink) and PWM 5 (Green), PWM 6 (Blue) 19
13 Level 1 – Incremental System Build Block Diagram 20

C2000, Code Composer Studio are trademarks of Texas Instruments.
All other trademarks are the property of their respective owners.

Copyright © 2013, Texas Instruments Incorporated

14 The Outputs of Hall Effect Sensors, Hall A, B and C 22
15 PWMDAC Outputs BemfA, BemfB and BemfC (Vdcbus = 160 V) 22
16 (a) mod6 Counter (b) Impulse Output, dlog.prescalar = 3 23
17 (a) mod6 Counter, (b) BemfA, (c) BemfB and (d)BemfC (dlog.prescalar = 25 and Vdcbus = 160 V) 23
18 Level 2 – Incremental System Build Block Diagram 24
19 (a) mod6 Counter, (b) HallGpioAccepted (dlog.prescalar = 25 and Vdcbus = 160 V) 26
20 (a) mod6 Counter, (b) HallGpioAccepted, (c) mod1.trigInput (Vdcbus = 160 V) 26
21 Level 3 – Incremental System Build Block Diagram 27
22 (a) mod6 Counter, (b) HallGpioAccepted, (c) speed, (under 0.5 pu load, Vdcbus = 160 V) 29
23 Level 4 - Incremental System Build Block Diagram 30
24 (a) mod6 Counter, (b)BemfA, (c) BemfB (c)BemfC (under no-load at 0.5pu speed, Vdcbus = 160 V) 31
25 (a) mod6 counter, (b)BemfA, (c) BemfB (c)BemfC (under 0.5 pu load at 0.3pu speed, Vdcbus = 160 V) 32
26 Level 5 - Incremental System Build Block Diagram 33

List of Tables

1 Comparison of BLDC and PMSM Motors 3
2 Watch Window Variables 17
3 Testing Modules in Each Incremental System Build 17

1 Introduction

The economic constraints and new standards legislated by governments place increasingly stringent requirements on electrical systems. New generations of equipment must have higher performance parameters such as better efficiency and reduced electromagnetic interference. System flexibility must be high to facilitate market modifications and to reduce development time. All these improvements must be achieved while, at the same time, decreasing system cost.

Brushless motor technology makes it possible to achieve these specifications. Such motors combine high reliability with high efficiency, and for a lower cost in comparison with brush motors. This document describes the use of a brushless DC (BLDC) motor. Although the brushless characteristic can be applied to several kinds of motors (the AC synchronous motors, stepper motors, switched reluctance motors, AC induction motors), the BLDC motor is conventionally defined as a permanent magnet synchronous motor with a trapezoidal back EMF waveform shape. Permanent magnet synchronous machines with trapezoidal back EMF and (120 electrical degrees wide) rectangular stator currents are widely used as they offer the following advantages first, assuming the motor has pure trapezoidal back EMF and that the stator phases commutation process is accurate, the mechanical torque developed by the motor is constant. Secondly, the brushless DC drives show a very high mechanical power density. This application report covers the 280x controllers and some system considerations to get out high performances from a BLDC motor drive.

Copyright © 2013, Texas Instruments Incorporated

2 BLDC Motors

The BLDC motor is an AC synchronous motor with permanent magnets on the rotor (moving part) and windings on the stator (fixed part). Permanent magnets create the rotor flux and the energized stator windings create electromagnet poles. The rotor (equivalent to a bar magnet) is attracted by the energized stator phase. By using the appropriate sequence to supply the stator phases, a rotating field on the stator is created and maintained. This action of the rotor, chasing after the electromagnet poles on the stator, is the fundamental action used in synchronous permanent magnet motors. The lead between the rotor and the rotating field must be controlled to produce torque and this synchronization implies knowledge of the rotor position.

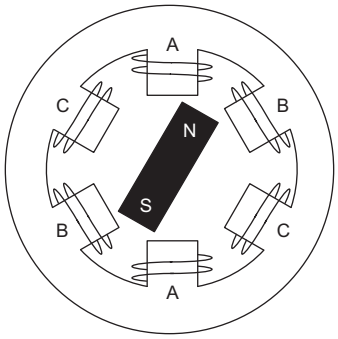


Figure 1. A Three-Phase Synchronous Motor With a One Permanent Magnet Pair Pole Rotor

On the stator side, three phase motors are the most common. These offer a good compromise between precise control and the number of power electronic devices required to control the stator currents. For the rotor, a greater number of poles usually create a greater torque for the same level of current. On the other hand, by adding more magnets, a point is reached where, because of the space needed between magnets, the torque no longer increases. The manufacturing cost also increases with the number of poles. As a consequence, the number of poles is a compromise between cost, torque and volume.

Permanent magnet synchronous motors can be classified in many ways, but a couple are of interest because they depend on back-EMF profiles: the brushless direct current (BLDC) motor and the permanent magnet synchronous motor (PMSM). This terminology defines the shape of the back EMF of the synchronous motor. Both BLDC and PMSM motors have permanent magnets on the rotor, but differ in the flux distributions and back-EMF profiles. To get the best performance out of the synchronous motor, it is important to identify the type of motor in order to apply the most appropriate type of control, as described in the next sections

Table 1. Comparison of BLDC and PMSM Motors

Comparison of BLDC and PMSM Motors	
BLDC	**PMSM**
Synchronous machine	Synchronous machine
Fed with direct currents	Fed with sinusoidal currents
Trapezoidal Bemf	Sinusoidal Bemf
Stator Flux position commutation each 60°	Continuous stator flux position variation
Only two phases ON at the same time	Possible to have three phases ON at the same time
Torque ripple at commutations	No torque ripple at commutations
Low order current harmonics in the audible range	Less harmonics due to sinusoidal excitation
Higher core losses due to harmonic content	Lower core loss
Less switching losses	Higher switching losses at the same switching freq.
Control algorithms are relatively simple	Control algorithms are mathematically intensive

Copyright © 2013, Texas Instruments Incorporated

- Both motor types are synchronous machines. The only difference between them is the shape of the induced voltage, resulting from two different manners of wiring the stator coils. The back EMF is trapezoidal in the BLDC motor case and sinusoidal in the PMSM motor case.
- BLDC machines could be driven with sinusoidal currents and PMSM with direct currents, but for better performance, PMSM motors should be excited by sinusoidal currents and BLDC machines by direct currents.
- The can structure (hardware and software) of a sinusoidal motor required several current sensors and sinusoidal phase currents were hard to achieve with analog techniques. Therefore, many motors (sinusoidal like trapezoidal) were driven with direct current for cost and simplicity reasons (low resolution position sensors and single low cost current sensor), compromising efficiency and dynamic behavior.
- Digital techniques addressed by the C2000 DSP controller make it possible to choose the right control technique for each motor type: processing power is used to extract the best performance from the machine and reduce system costs. Possible options are using sensorless techniques to reduce the sensor cost, or even eliminate it, and also complex algorithms can help simplify the mechanical drive train design, lowering the system cost.

3 BLDC Motor Control

The key to effective torque and speed control of a BLDC motor is based on relatively simple torque and back EMF equations, which are similar to those of the DC motor. The back EMF magnitude can be written as:

$$E = 2NlrBw$$

and the torque term as:

$$T = \left(\frac{1}{2} i^2 \frac{dL}{d\theta}\right) - \left(\frac{1}{2} B^2 \frac{dR}{d\theta}\right) + \left(\frac{4N}{\pi} Brl\pi i\right)$$

where N is the number of winding turns per phase, l is the length of the rotor, r is the internal radius of the rotor, B is the rotor magnet flux density, w is the motor's angular velocity, i is the phase current, L is the phase inductance, θ is the rotor position, R is the phase resistance.

The first two terms in the torque expression are parasitic reluctance torque components. The third term produces mutual torque, which is the torque production mechanism used in the case of BLDC motors. To sum up, the back EMF is directly proportional to the motor speed and the torque production is almost directly proportional to the phase current. These factors lead to the BLDC motor speed control schemes as shown in Figure 2:

Copyright © 2013, Texas Instruments Incorporated

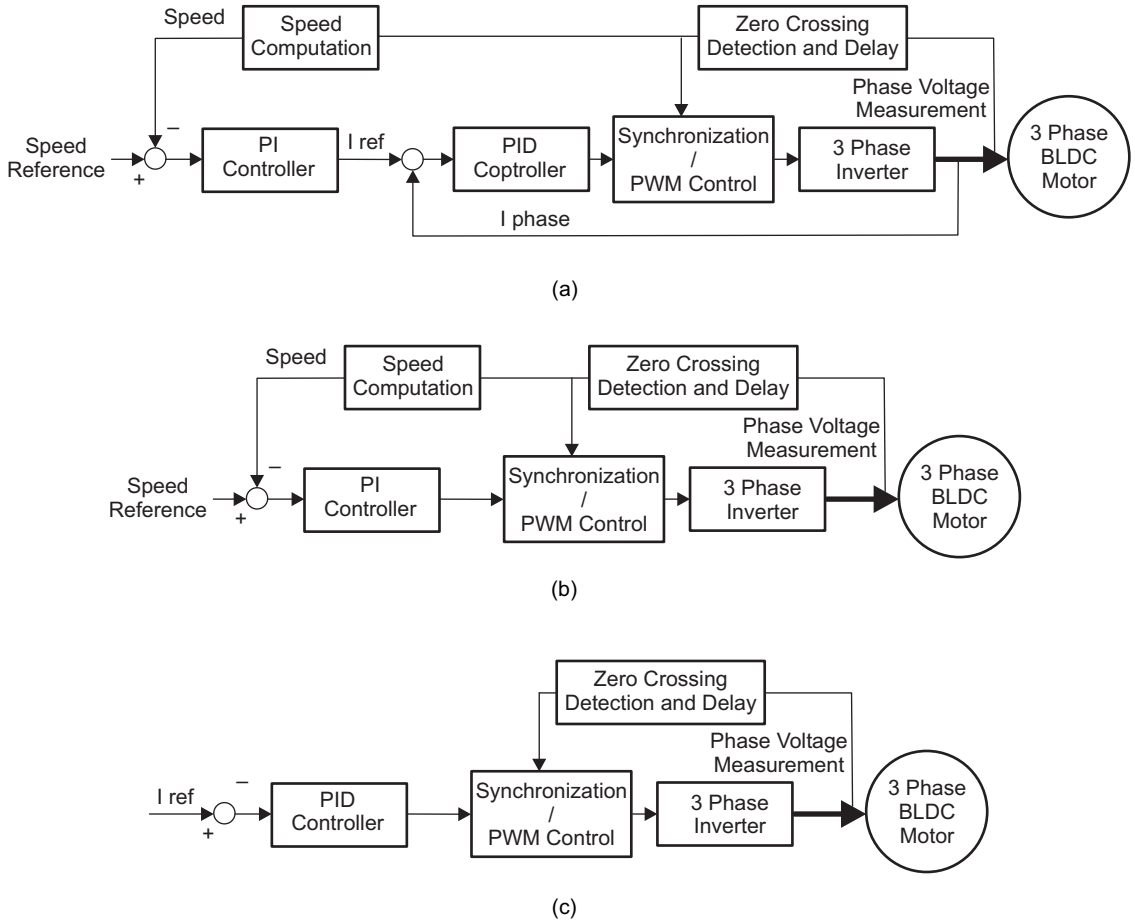


Figure 2. Speed and Current Control Loop Configurations for a BLDC Motor

The BLDC motor is characterized by a two phase ON operation to control the inverter. In this control scheme, torque production follows the principle that current should flow in only two of the three phases at a time and that there should be no torque production in the region of the back EMF zero crossings. Figure 3 describes the electrical wave forms in the BLDC motor in the two phases ON operation.

This control structure has several advantages:

- Only one current at a time needs to be controlled
- Only one current sensor is necessary (or none for speed loop only, as detailed in the next sections)
- The positioning of the current sensor allows the use of low cost sensors as a shunt

The principle of the BLDC motor is, at all times, to energize the phase pair, which can produce the highest torque. To optimize this effect the back EMF shape is trapezoidal. The combination of a DC current with a trapezoidal back EMF makes it theoretically possible to produce a constant torque. In practice, the current cannot be established instantaneously in a motor phase; as a consequence the torque ripple is present at each 60° phase commutation.

Copyright © 2013, Texas Instruments Incorporated

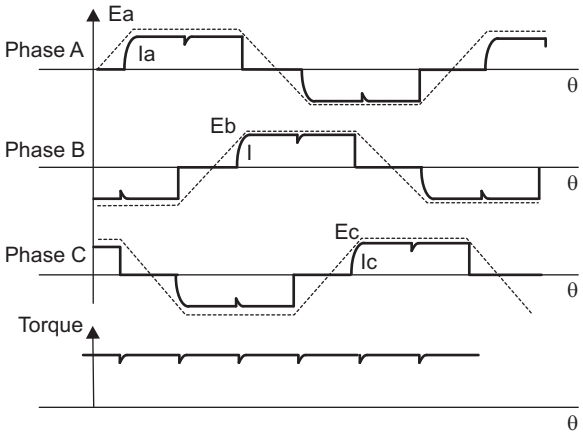


Figure 3. Electrical Waveforms in the Two Phase ON Operation and Torque Ripple

If the motor used has a sinusoidal back EMF shape, this control can be applied but the produced torque is:

- Not constant but made up from portions of a sine wave. This is due to its being the combination of a trapezoidal current control strategy and of a sinusoidal back EMF. Bear in mind that a sinusoidal back EMF shape motor controlled with a sine wave strategy (three phase ON) produces a constant torque.
- The torque value produced is weaker.

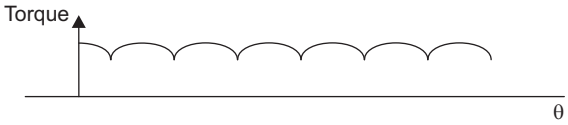


Figure 4. Torque Ripple in a Sinusoidal Motor Controlled as a BLDC

4 System Topology

4.1 Three Phase Inverter

The BLDC motor control consists of generating DC currents in the motor phases. This control is subdivided into two independent operations: stator and rotor flux synchronization and control of the current value. Both operations are realized through the three phase inverter depicted in Figure 5.

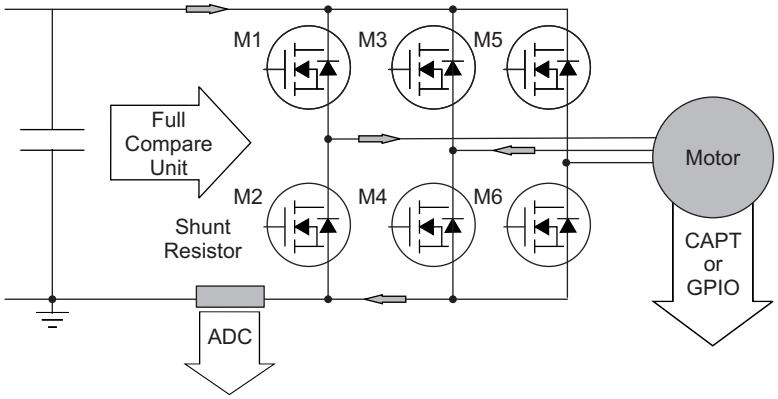


Figure 5. Three Phase Inverter

Copyright © 2013, Texas Instruments Incorporated

The flux synchronization is derived from the position information coming from sensors, or from sensorless techniques. From the position, the controller determines the appropriate pair of transistors (Q1 to Q6) that must be driven. The regulation of the current to a fixed 60° reference can be realized in either of the two different modes:

- The Pulse Width Modulation (PWM) Mode:

  The supply voltage is chopped at a fixed frequency with a duty cycle depending on the current error. Therefore, both the current and the rate of change of current can be controlled. The two phase supply duration is limited by the two phase commutation angles. The main advantage of the PWM strategy is that the chopping frequency is a fixed parameter; hence, acoustic and electromagnetic noises are relatively easy to filter.

  There are also two ways of handling the drive current switching: hard chopping and soft chopping. In the hard chopping technique, both phase transistors are driven by the same pulsed signal: the two transistors are switched-on and switched-off at the same time. The power electronics board is then easier to design and is also cheaper as it handles only three pulsed signals. A disadvantage of the hard chopping operation is that it increases the current ripple by a large factor in comparison with the soft chopping approach.

  The soft chopping approach allows not only a control of the current and of the rate of change of the current but a minimization of the current ripple as well. In this soft chopping mode, the low side transistor is left ON during the phase supply and the high side transistor switches according to the pulsed signal. In this case, the power electronics board has to handle six PWM signals.

- The Hysteresis Mode:

  In the hysteresis-type current regulator, the power transistors are switched off and on according to whether the current is greater or less than a reference current. The error is used directly to control the states of the power transistors. The hysteresis controller is used to limit the phase current within a preset hysteresis band. As the supply voltage is fixed, the result is that the switching frequency varies as the current error varies. Therefore, the current chopping operation is not a fixed chopping frequency PWM technique. This method is more commonly implemented in drives where motor speed and load do not vary too much, so that the variation in switching frequency is small. Here again, both hard and soft chopping schemes are possible. Since the width of the tolerance band is a design parameter, this mode allows current control to be as precise as desired, but acoustic and electromagnetic noise are difficult to filter because of the varying switching frequency.

4.2 Shaft Position Sensors

The position information is used to generate precise firing commands for the power converter, ensuring drive stability and fast dynamic response. In servo applications position feedback is also used in the position feedback loop. Velocity feedback can be derived from the position data, eliminating a separate velocity transducer for the speed control loop.

Three common types of position sensors are used: the incremental sensors, the three Hall Effect sensor, and the resolver.

- The incremental sensors use optically coded disks with either single track or quadrature resolution to produce a series of square wave pulses. The position is determined by counting the number of pulses from a known reference position. Quadrature encoders are direction sensitive and do not produce false data due to any vibration when the shaft begins rotation. The Quadrature encoder pulse unit of the F280x handles encoders' output lines and can provide 1, 2 or 4 times the encoder resolution. Speed information is available by counting the number of pulses within a fix time period.
- The three Hall Effect sensors provide three overlapping signals giving a 60° wide position range. The three signals can be wired to the F280x input capture/GPIO pins, therefore, speed information is available by measuring the time interval between two input captures. The time interval is automatically stored by the 280x into a specific register at each input capture. From speed information, it is numerically possible to get the precise position information needed for sharp firing commands.
- The resolver is made up of three windings (different from the motor's windings): one linked to the rotor and supplied with a sinusoidal source and two other orthogonal coils linked to the stator. A back EMF is induced by the rotating coil in each of the two stator resolver windings. By decoding these two signals, it is possible to get cos(q) and sin(q) where q is the rotor position. The resolver resolution depends only on the AD conversion.

Copyright © 2013, Texas Instruments Incorporated

4.3 Current Sensing

A characteristic of the BLDC control is to have only one current at a time in the motor (two phases ON). Consequently, it is not necessary to put a current sensor on each phase of the motor; one sensor placed in the line inverter input makes it possible to control the current of each phase. Moreover, using this sensor on the ground line, insulated systems are not necessary, and a low cost resistor can be used. Its value is set such that it activates the integrated over-current protection when the maximum current permitted by the power board has been reached.

Each current measurement leads to a new PWM duty cycle loaded at the beginning of a PWM cycle. Note that, during turn OFF, the shunt resistor does not have this current to sense, regardless of whether the inverter is driven in hard chopping or in soft chopping mode. Figure 6 depicts the shunt current in soft chopping mode and shows that in the turn OFF operation the decreasing current flows through the M2 free wheeling diode and through the maintained closed M4 (so there is no current 9 observable in the shunt in this chopping mode during turn OFF). This implies that it is necessary to start a current conversion in the middle of the PWM duty cycle.

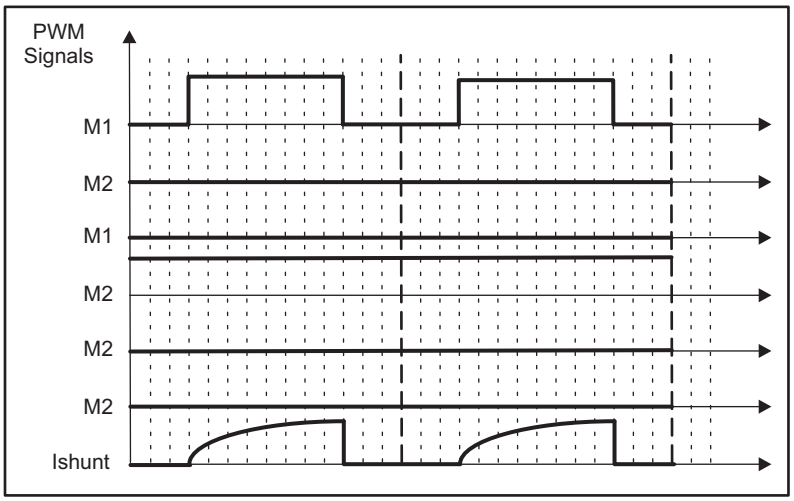


Figure 6. Shunt Resistor Voltage Drop According to PWM Duty Cycles (Soft Chopping)

In the hard chopping mode during the turn OFF, neither M1 nor M4 drive the current so that the decreasing phase current flows from ground through the shunt resistor via M2 and M3 free wheeling diodes and back to ground via the capacitor. In this chopping mode, it is possible to see the exponentially decreasing phase current across the shunt as a negative shunt voltage drop appears. Assuming that neither the power board nor the control board support negative voltages, this necessitates that the current be sensed in the middle of the turn ON.

4.4 Position and Speed Sensing

The motor in this application is equipped with three Hall Effect sensors. These sensors are fed by the power electronics board. The sensor outputs are directly wired to the GPIO pins. The Hall Effect sensors give three 180° overlapping signals, thus providing the six mandatory commutation points: The rising and falling edges of the sensor output are detected, the corresponding flags are generated. The system first determines which edge has been detected, then computes the time elapsed since the last detected edge and commutates the supplied phases.

The speed feedback is derived from the position sensor output signals. As mentioned in the previous paragraph, there are six commutation signals per mechanical revolution. In other words, between two commutation signals there are 60 mechanical degrees. The speed can be written as:

$$\frac{\Delta\theta}{\Delta T}$$

where θ is the mechanical angle, it is possible to get the speed from the computed elapsed time between two captures. Between two commutation signals, the angle variation is constant as the Hall Effect sensors are fixed relative to the motor, so speed sensing is reduced to a simple division.

Copyright © 2013, Texas Instruments Incorporated

5 Benefits of 32-Bit C2000 Controllers for Digital Motor Control (DMC)

The C2000 family of devices posses the desired computation power to execute complex control algorithms along with the right mix of peripherals to interface with the various components of the DMC hardware like the analog-to-digital converter (ADC), enhanced pulse width modulator (ePWM), Quadrature Encoder Pulse (QEP), enhanced Capture (ECAP), and so forth. These peripherals have all the necessary hooks for implementing systems that meet safety requirements, like the trip zones for PWMs and comparators. Along with this the C2000 ecosystem of software (libraries and application software) and hardware (application kits) help in reducing the time and effort needed to develop a Digital Motor Control solution. The DMC Library provides configurable blocks that can be reused to implement new control strategies. IQMath Library enables easy migration from floating point algorithms to fixed point thus accelerating the development cycle.

Therefore, with C2000 family of devices it is easy and quick to implement complex control algorithms (sensored and sensorless) for motor control. The use of C2000 devices and advanced control schemes provides the following system improvements:

- Favors system cost reduction by an efficient control in all speed range implying right dimensioning of power device circuits
- Use of advanced control algorithms it is possible to reduce torque ripple, thus resulting in lower vibration and longer life time of the motor
- Advanced control algorithms reduce harmonics generated by the inverter, reducing filter cost.
- Use of sensorless algorithms eliminates the need for speed or position sensor.
- Decreases the number of look-up tables that reduces the amount of memory required
- The real-time generation of smooth near-optimal reference profiles and move trajectories, results in better-performance
- Generation of high resolution PWM's is possible with the use of ePWM peripheral for controlling the power switching inverters
- Provides single chip control system

For advanced controls, C2000 controllers can also perform the following:

- Enables control of multi-variable and complex systems using modern intelligent methods such as neural networks and fuzzy logic.
- Performs adaptive control. C2000 controllers have the speed capabilities to concurrently monitor the system and control it. A dynamic control algorithm adapts itself in real time to variations in system behavior.
- Performs parameter identification for sensorless control algorithms, self commissioning, online parameter estimation update.
- Performs advanced torque ripple and acoustic noise reduction
- Provides diagnostic monitoring with spectrum analysis. By observing the frequency spectrum of mechanical vibrations, failure modes can be predicted in early stages.
- Produces sharp-cut-off notch filters that eliminate narrow-band mechanical resonance. Notch filters remove energy that would otherwise excite resonant modes and possibly make the system unstable.

6 TI Literature and Digital Motor Control (DMC) Library

The Digital Motor Control (DMC) library is composed of functions represented as blocks. These blocks are categorized as Transforms & Estimators (Clarke, Park, Sliding Mode Observer, Phase Voltage Calculation, and Resolver, Flux, and Speed Calculators and Estimators), Control (Signal Generation, PID, BEMF Commutation, Space Vector Generation), and Peripheral Drivers (PWM abstraction for multiple topologies and techniques, ADC drivers, and motor sensor interfaces). Each block is a modular software macro is separately documented with source code, use, and technical theory. For the source codes and explanations of macro blocks, install controlSUITE from www.ti.com/controlsuite and choose the HVMotorKit installation.

- C:\TI\controlSUITE\libs\app_libs\motor_control\math_blocks\v4.0
- C:\TI\controlSUITE\libs\app_libs\motor_control\drivers\f2803x_v2.0

Copyright © 2013, Texas Instruments Incorporated

These modules allow you to quickly build or customize your own systems. The library supports the three motor types: ACI, BLDC, PMSM, and comprises both peripheral dependent (software drivers) and target dependent modules.

The DMC Library components have been used by TI to provide system examples. All DMC Library variables are defined and inter-connected at initialization. At run-time, the macro functions are called in order. Each system is built using an incremental build approach, which allows sections of the code to be built at different times so that the developer can verify each section of their application one step at a time. This is critical in real-time control applications where so many different variables can affect the system and many different motor parameters need to be tuned.

NOTE: TI DMC modules are written in the form of macros for optimization purposes. For more details, see *Optimizing Digital Motor Control (DMC) Libraries* (SPRAAK2). The macros are defined in the header files. You can open the respective header file and change the macro definition, if needed. In the macro definitions, there should be a backslash "\" at the end of each line as shown in Example 1, which means that the code continues in the next line. Any character including invisible ones like a "space" or "tab" after the backslash will cause compilation error. Therefore, make sure that the backslash is the last character in the line. In terms of code development, the macros are almost identical to C function, and that you can easily convert the macro definition to a C functions.

Example 1. A Typical DMC Macro Definition

```
#define PARK_MACRO(v)                                          \
                                                               \
   v.Ds = _IQmpy(v.Alpha,v.Cosine) + _IQmpy(v.Beta,v.Sine); \
   v.Qs = _IQmpy(v.Beta,v.Cosine)  - _IQmpy(v.Alpha,v.Sine);
```

6.1 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 the TMS320C2803x-based controllers on Code Composer Studio™ software. The framework uses the following modules: (1):

(1) Please refer to pdf documents in the motor control folder explaining the details and theoretical background of each macro.

Macro Names	Explanation
BLDCPWM / PWMDAC	PWM and PWMDAC Drives
HALL_GPIO DRV	Hall Drive
PI	PI Regulators
RC	Ramp Controller (slew rate limiter)
RC2	Ramp up and Ramp down Module
RC3	Ramp down Module
QEP and CAP	QEP and CAP Drives (optional for speed loop tuning with a speed sensor)
SPEED_FR	Speed Measurement (based on sensor signal frequency)
IMPULSE	Impulse Generator
MOD6_CNT	Mod 6 Counter

In this system, the trapezoidal control of BLDC motors using Hall Effect sensors is experimented with and will explore the performance of the speed controller. The BLDC motor is driven by a conventional voltage-source inverter. The TMS320F2803x control card is used to generate three PWM signals. The motor is driven by an integrated power module by means of BLDC-specific PWM technique. The DC bus return current (I fb_Sum) is measured and sent to the TMS320x2803x via analog-to-digital converters (ADCs). Hall Effect signals are level shifted on the board and sent to GPIO pins for commutation.

Copyright © 2013, Texas Instruments Incorporated

The HVBLDC_Sensored project has the following properties:

C Framework		
System Name	**Program Memory Usage 2803x**	**Data Memory Usage 2803x [1]**
HVBLDC_Sensored	3576 words [2]	1980 words

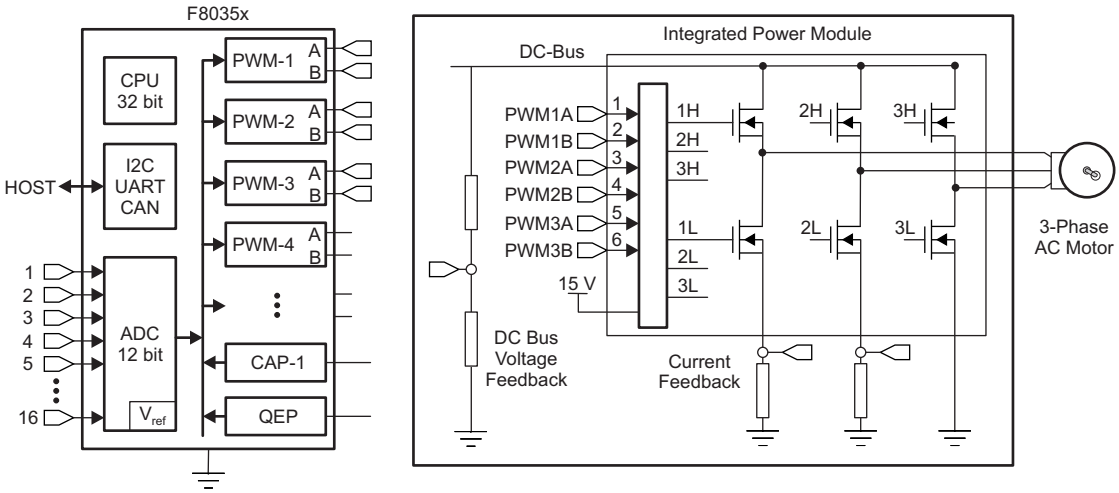
(1) Excluding the stack size
(2) Excluding "IQmath" Look-up Tables

CPU Utilization– PMSM Sensorless	
Total Number of Cycles	943 [1]
CPU Utilization @ 60 Mhz	14.9%
CPU Utilization @ 40 Mhz	22.3%

(1) At 20 kHz ISR frequency. Debug macros excluded.

System Features	
Development and Emulation	Code Composer Studio V4.1 (or above) with real-time debugging
Target Controller	TMS320F2803x
PWM Frequency	20 kHz PWM (Default), 60 kHz PWMDAC
PWM Mode	Asymmetrical with no dead band
Interrupts	CPU Timer 0 – Implements 40 kHz ISR execution rate
Peripherals Used	PWM 1, 2, 3 for motor control
	PWM 6A, 6B, 7A and 7B for DAC outputs (x2803x only)
	ADC A2 for low side DC bus return current sensing

The overall system implementing a 3-ph sensored BLDC control is depicted in Figure 7 and Figure 8.



A Note that the dcbus return current is obtained through the summation of three phase currents in R1.1.

Figure 7. A 3-ph BLDC Drive Implementation

Copyright © 2013, Texas Instruments Incorporated

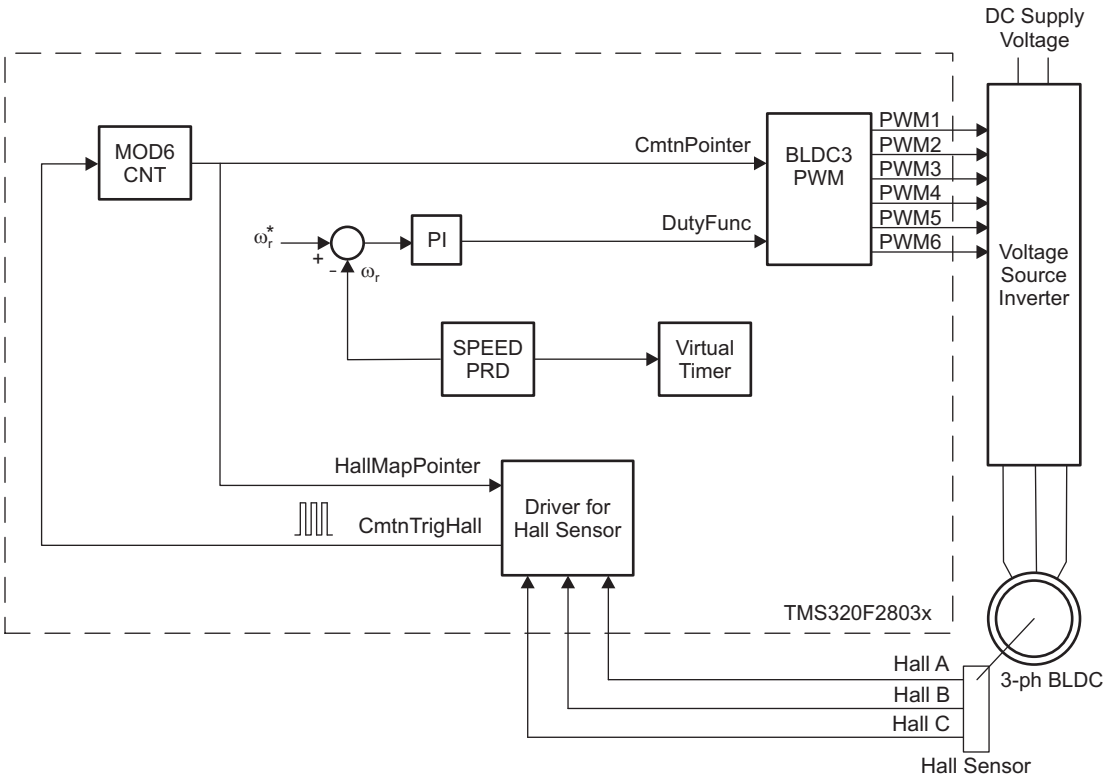


Figure 8. Overall Block Diagram of Hall-Sensor Control of BLDC Motor

Copyright © 2013, Texas Instruments Incorporated

The software flow is described in the Figure 9.

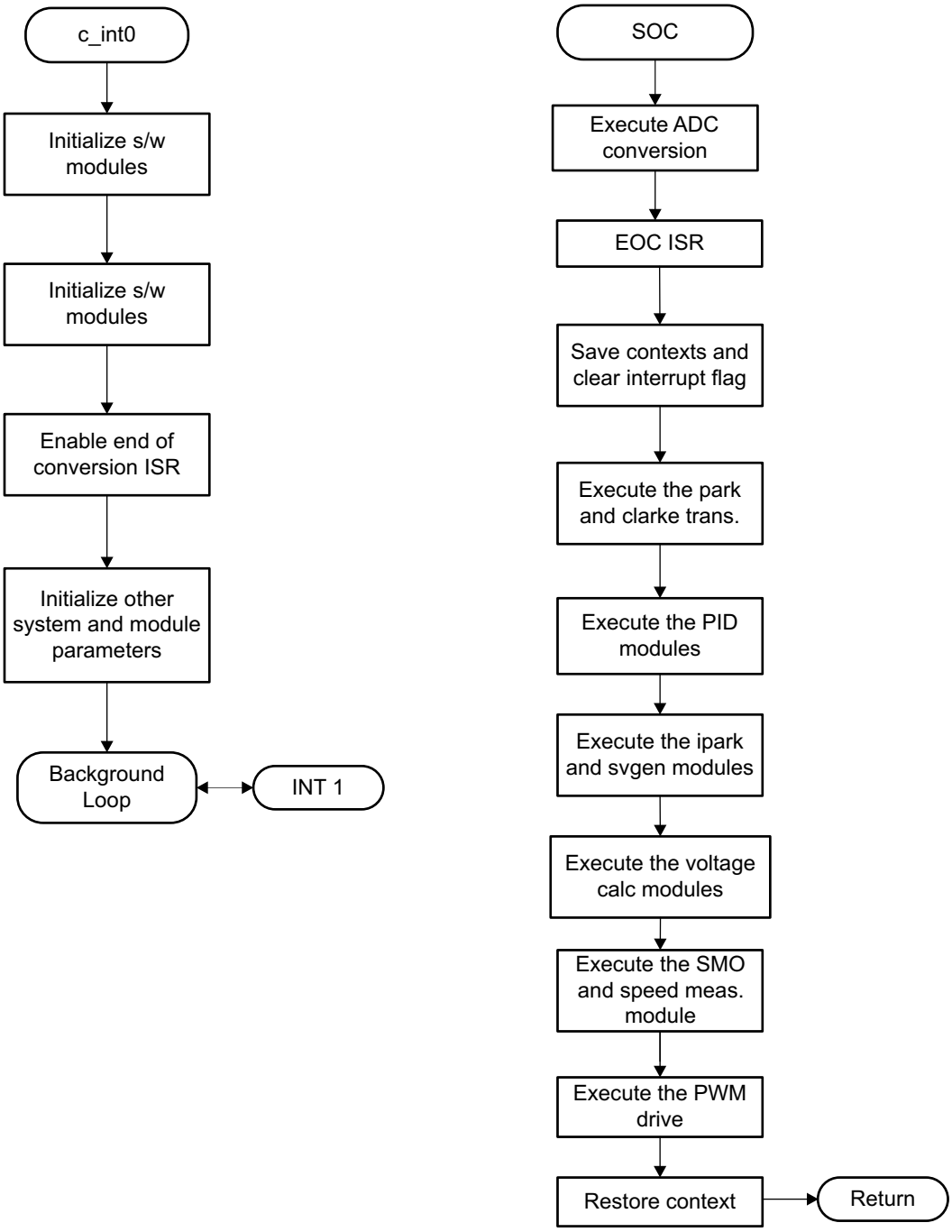


Figure 9. Software Flow

Copyright © 2013, Texas Instruments Incorporated

7 Hardware Configuration (HVDMC R1.1 Kit)

For an overview of the kit's hardware and steps on how to setup this kit, see the *HVMotorCtrl+PFC How to Run Guide* located at: www.ti.com/controlsuite and choose the HVMotorKit installation.

Some of the hardware setup instructions are listed below for quick reference.

1. Open the lid of the HV kit.
2. Install the Jumpers [Main]-J3, J4 and J5, J9 for 3.3 V, 5 V and 15 V power rails and JTAG reset line.
3. Unpack the DIMM style controlCARD and place it in the connector slot of [Main]-J1. Push down vertically using even pressure from both ends of the card until the clips snap and lock. To remove the card, simply spread open the retaining clip with your thumbs.
4. Connect a USB cable to the connector [M3]-JP1. This enables an isolated JTAG emulation to the C2000 device. [M3]-LD1 should turn on. Make sure [M3]-J5 is not populated. If the included Code Composer Studio is installed, the drivers for the onboard JTAG emulation will automatically be installed. If a windows installation window appears, try to automatically install drivers from those already on your computer. The emulation drivers are found at http://www.ftdichip.com/Drivers/D2XX.htm. The correct driver is the one listed to support the FT2232.
5. If a third party JTAG emulator is used, connect the JTAG header to [M3]-J2 and additionally the [M3]-J5 needs to be populated to put the onboard JTAG chip in reset.
6. Ensure that [M6]-SW1 is in the "Off" position. Connect the 15 V DC power supply to [M6]-JP1.
7. Turn on [M6]-SW1. Now [M6]-LD1 should turn on. Notice that the control card LED lights up as well indicating that the control card is receiving power from the board.
8. Note that the motor should be connected to the [M5]-TB3 terminals after you finish with the first incremental build step.
9. Note the DC Bus power should only be applied during incremental build levels when instructed to do so. The two options to get DC Bus power are discussed below:
   - Set the power supply output to zero and connect [Main]-BS5 and BS6 to the DC power supply and ground, respectively, to use DC power supply.
   - Connect [Main]-BS1 and BS5 to each other using the banana plug cord to use AC Mains power. Now, connect one end of the AC power cord to [Main]-P1. The other end needs to be connected to the output of a variac. Make sure that the variac output is set to zero and it is connected to the wall supply through an isolator.

NOTE: Phase voltage sensing caps (C21, 22, 23) are optimized for AC motor control. Use the lower value capacitors (≤ 2.2 nF) to reach peak torque. Also note that, dc bus current feedback is obtained as the sum of all three phases instead of a shunt resistor on the dc bus return path.

Copyright © 2013, Texas Instruments Incorporated

For reference, Figure 10 and Figure 11 show the jumper and connectors that need to be connected for this lab. Note that the order of motor cable colors connected to inverter output should be as shown below. For more details, see the device-specific motor data sheet.

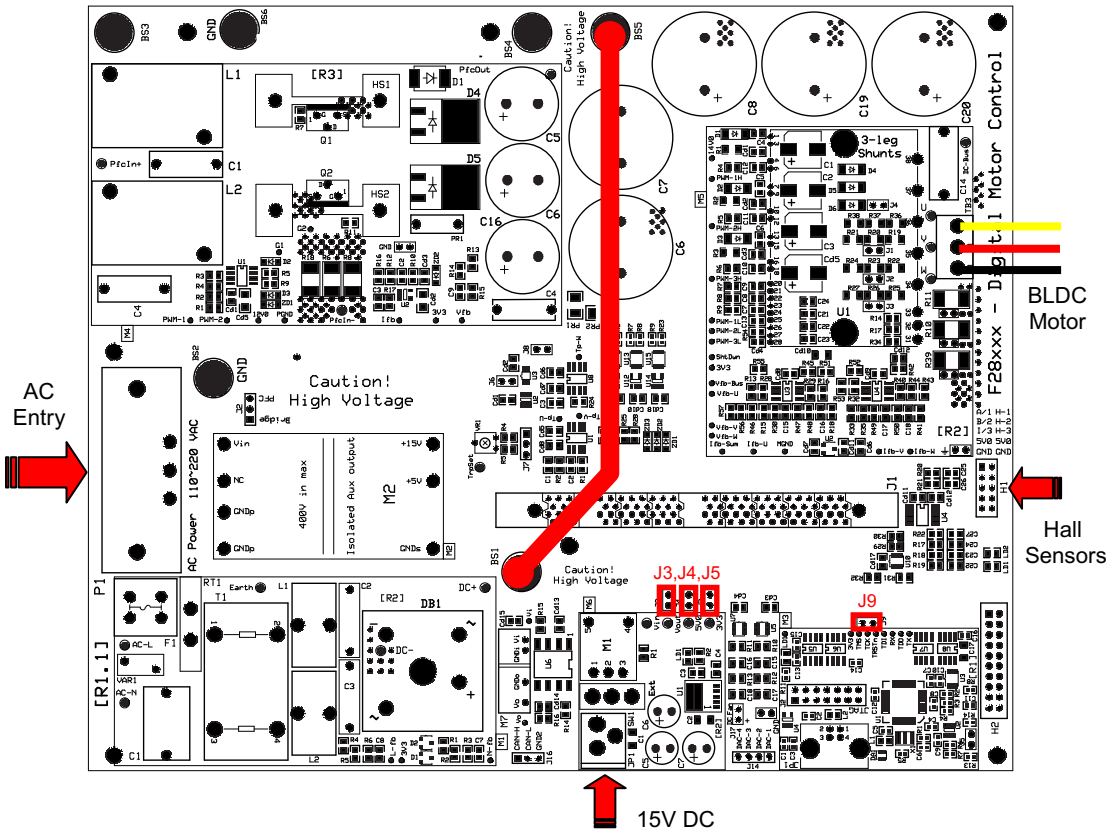


Figure 10. Using AC Power to Generate DC Bus Power

CAUTION

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

Copyright © 2013, Texas Instruments Incorporated

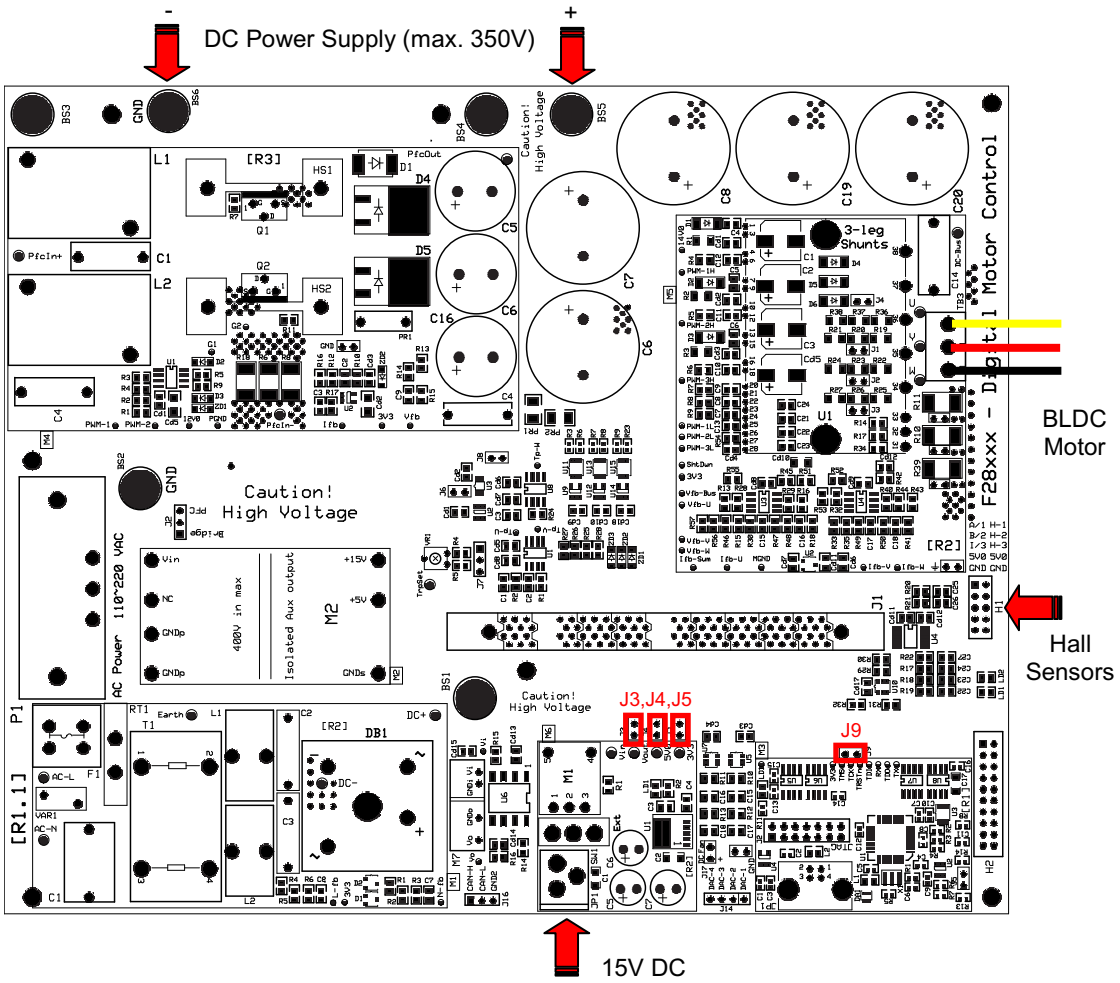


Figure 11. Using External DC Power Supply to Generate DC-Bus for the Inverter

CAUTION

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

7.1 Software Setup Instructions to Run the HVBLDC_Sensored Project

For more information, see the *Software Setup for HVMotorCtrl+PFC Kit Projects* section in the *HVMotorCtrl+PFC Kit How to Run Guide* that can be found at www.ti.com/controlsuite, then choose the HVMotorKit installation.

How to install Code Composer Studio and set it up to run with this project is discussed in this section.

1. Select the HVBLDC_Sensored as the active project.
2. Verify that the build level is set to 1, and then right click on the project name and select “Rebuild Project”. Once the build completes, launch a debug session to load the code into the controller.
3. Open a watch window and add the critical variables as shown in Table 2 and select the appropriate Q format for them.

Copyright © 2013, Texas Instruments Incorporated

Table 2. Watch Window Variables

Variable Name	Viewed as
EnableFlag	Unsigned Integer
IsrTicker	Unsigned Integer
SpeedRef	Q24
Dlog.prescalar	Integer
SpeedLoopFlag	Unsigned Integer
ILoopFlag	Unsigned Integer
CmtnPeriodTarget	Unsigned Integer
DFuncDesired	Unsigned Integer
ClosedFlag	Unsigned Integer
pi_spd.Out	Q24
pi_idc.Out	Q24
Pi_spd.Kp	Q24
speed1.Speed	Q24

4. Setup the time graph windows by importing Graph1.graphProp and Graph2.graphProp from the following location: www.ti.com/controlsuite - `(developement_kits\HVMotorCtrl+PfcKit_v2.0\HVBLDC_Sensored).`
5. Click on the Continuous Refresh button on the top left corner of the graph tab to enable periodic capture of data from the microcontroller.

8 Incremental System Build for Sensored BLDC Project

The system is gradually built up so the final system can be confidently operated. Six phases of the incremental system build are designed to verify the major software modules used in the system. Table 3 summarizes the modules testing and using in each incremental system build.

Table 3. Testing Modules in Each Incremental System Build (1)

Software Module	Phase 1	Phase 2	Phase 3	Phase 4	Phase 5
PWMDAC_MACRO		√√	√	√	√
RC3_MACRO	√√	√√	√	√	√
MOD6_CNT_MACRO	√√	√√	√	√	√
IMPULSE_MACRO	√√	√√	√	√	√
BLDCPWM_MACRO	√√	√√	√	√	√
RC2_MACRO			√√	√	√
HALL3 _READ_MACRO			√√	√	√
SPEED_FR_MACRO			√√	√	√
PI_MACRO (IDC)				√√	√
RC_MACRO					√√
PI_MACRO (SPD)					√√

(1) The symbol √ means this module is using and the symbol √√ means this module is testing in this phase.

8.1 Level 1 Incremental Build

Assuming the load and build steps described in the *HVMotorCtrl+PFC Kit How To Run Guide* completed successfully, this section describes the steps for a “minimum” system check-out, which confirms the operation of the system interrupt, the peripheral and target independent modules, and one peripheral dependent module.

1. Open HVBLDC_Sensored-Settings.h and select the level 1 incremental build option by setting the BUILDLEVEL to LEVEL1 (#define BUILDLEVEL LEVEL1).
2. Right click on the project name and click Rebuild Project.

Copyright © 2013, Texas Instruments Incorporated

3. Click on the debug button, reset the CPU, restart, enable real-time mode and run, once the build is complete.
4. Set the “EnableFlag” to 1 in the watch window. The variable named “IsrTicker” will now keep on increasing.
5. Confirm this by watching the variable in the watch window. This confirms that the system interrupt is 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

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 the RMP3CNTL module allows speed up of the BLDC motor from standstill in an open loop configuration (like a stepper motor).
- One of the inputs to the 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. You initialize this system variable with the appropriate value depending on the type of the BLDC motor. The second input to the 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 the RMP3CNTL module is Out, which provides a variable time period gradually decreasing in time. The second output of the 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 the DATALOG module is configured to point to the mod1.Counter. Therefore, the Out signal is shown in the graph in Code Composer Studio.
- 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 six 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 six values of the pointer variable, CmtnPointer, are used to generate the six commutation states of the power inverter driving the BLDC motor. The duty cycle of the generated PWM outputs (according to the six commutation states) during the motor speed up phase are determined by the input DfuncTesting.
- Compile, load, and run the program with real-time mode.
- Set the “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. Enter a new lower value for CmtnperiodTarget (DesiredInput). Then period will gradually reduce to the new value.
- Check the MOD6_CNT output variable counter in the watch window and graph window. This varies between 0 and 5.
- Use a scope to check the PWM outputs controlled by the peripheral dependent module - BLDC_3PWM_DRV. The odd numbered PWM outputs (PWM1, PWM3, and PWM5) will either generate PWM pulses or remain OFF. The even numbered PWM outputs (PWM2, PWM4, and PWM6) will either remain ON or OFF.
- The output states of all the six PWM outputs will be such that together they generate the six 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.

Copyright © 2013, Texas Instruments Incorporated

When running this build, the PWM outputs should be appeared as shown in Figure 12.

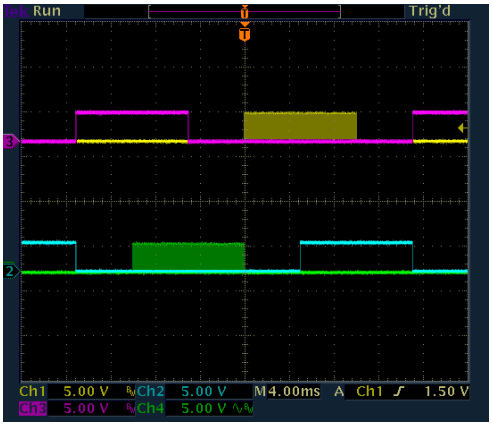


Figure 12. The PWM Outputs: PWM 1 (Yellow), PWM 2 (Pink) and PWM 5 (Green), PWM 6 (Blue)

Copyright © 2013, Texas Instruments Incorporated

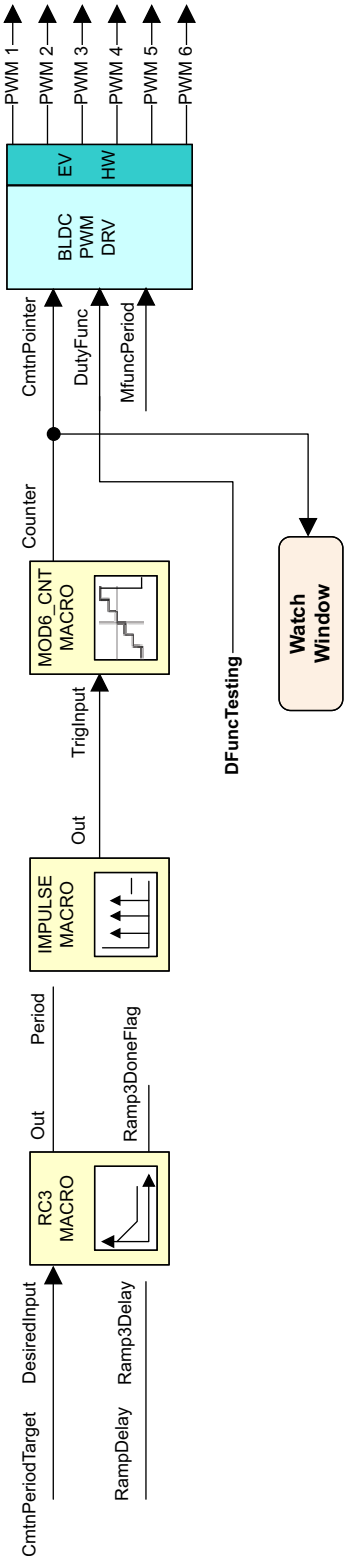


Figure 13. Level 1 – Incremental System Build Block Diagram

Level 1 describes the steps for a “minimum” system check-out, which confirms the operation of system interrupts, some peripheral and target independent modules, and one peripheral dependent module.

Copyright © 2013, Texas Instruments Incorporated

8.2 Level 2 - Incremental Build

Assuming section BUILD 1 is completed successfully, this section verifies the open loop motor operation and current measurement.

1. Open HVBLDC_Sensored-Settings.h and select level 1 incremental build option by setting the BUILDLEVEL to LEVEL2 (#define BUILDLEVEL LEVEL2) and save the file.
2. Right Click on the project name and click Rebuild Project.
3. Click on debug button, reset the CPU, restart, enable real-time mode and run, once the build is complete.
4. Set the “EnableFlag” to 1 in the watch window. The variable named “IsrTicker” is incrementally increased as seen in the 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

The key steps can be explained as follows:

8.3 Level 2A — Open Loop Test

In this part, the phase voltage calculation module, PHASEVOLT_MACRO, is tested. Now, gradually increase the DC bus voltage. The outputs of this module can be checked via the graph window as follows:

- Compile, load, and run program with real-time mode and then increase the voltage at the variac and the dc power supply to get the appropriate DC-bus voltage. Now the motor is running with default DFuncTesting value.
- If the open loop commutation parameters are chosen properly, then the motor will gradually speed up and finally run at a constant speed in open loop commutation mode.
- The final speed of the motor depends on the parameter CmtnPeriodTarget. The lower the value for this variable the higher will be the motor final speed. Since the motor Bemf depends on it’s speed, the value chosen for the CmtnPeriodTarget also determines the generated Bemf.
- The average applied voltage to the motor during startup depends on the parameter DfuncTesting. The parameters DfuncTesting and CmtnPeriodTarget should be such that, at the end of motor speed up phase, the generated Bemf is lower than the average voltage applied to motor winding. This prevents the motor from stalling or vibrating. The default DfuncTesting and CmtnPeriodTarget values in the initialization section are selected for the motor in the HVDMC kit. When a different motor is tested, these values need to be tuned to prevent possible vibration and startup the motor properly. Both DfuncTesting and CmtnPeriodTarget should be adjusted accordingly in the watch window to increase the motor speed. The motor speed up time depends on RampDelay, the time period of the main sampling loop and the difference between CmtnPeriodTarget and CmtnPeriodSetpt.

NOTE: This step is not meant for wide speed and torque range operation; instead the overall system is tested and calibrated before closing the loops at a certain speed under no-load.

Bring the system to a safe stop as described below by reducing the bus voltage, taking the controller out of real-time mode and reset.

CAUTION

After verifying this, reduce the DC Bus voltage, take the controller out of real-time mode (disable), reset the processor (for details, see *HVMotorCtrl+PFC Kit How To Run Guide*). Note that after each test, this step needs to be repeated for safety purposes. Also note that improper shutdown might halt the PWMs at some certain states where high currents can be drawn, therefore, caution needs to be taken while doing these experiments.

Copyright © 2013, Texas Instruments Incorporated

8.4 Phase 2B — ADC Verification and Offset Calibration

- Verify the ADC operation by monitoring the dc bus return current and all three back EMFs (optional).
- Turn off the power supply and compile, load, and run program with real time mode. When the dc bus voltage is zero, the displayed current on the watch window (DCbus_current) should be zero. If not, adjust the offset value in the code by going to:

```
DCbus_current =
                    _IQ12toIQ(AdcResult.ADCRESULT4)-_IQ(0.5);
```

and change IQ15(0.50) offset value (IQ15(0.5087) or IQ15(0.4988) depending on the sign and amount of the offset. Once this step is completed, turn on the power supply and set the output value to zero.

When running level 2, the BLDC Hall Effect sensors' output and PWMDAC outputs should appear as shown in Figure 14 and Figure 15:

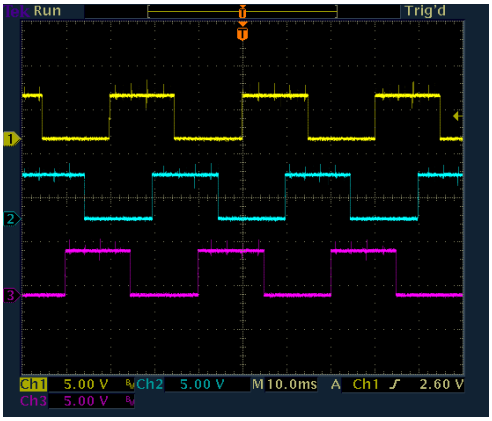


Figure 14. The Outputs of Hall Effect Sensors, Hall A, B and C

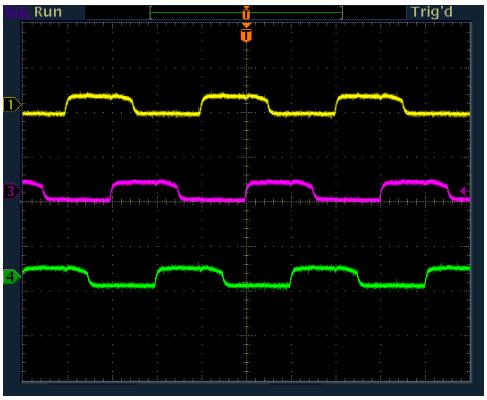


Figure 15. PWMDAC Outputs BemfA, BemfB and BemfC (Vdcbus = 160 V)

Copyright © 2013, Texas Instruments Incorporated

When running this level, the waveforms in the Code Composer Studio graphs should appear as shown in Figure 16 and Figure 17.

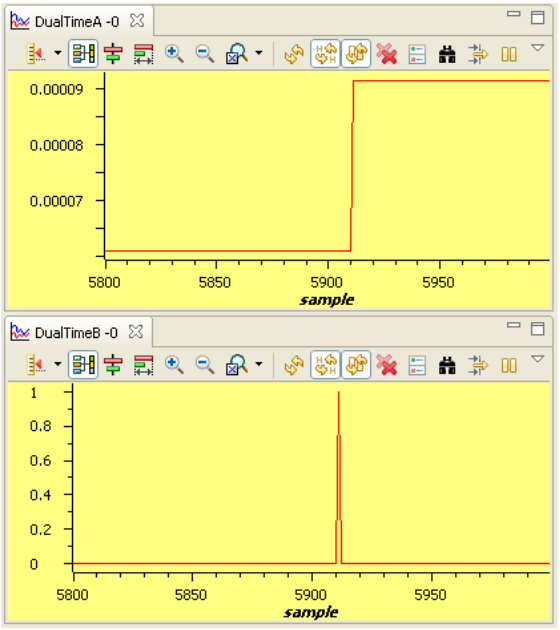


Figure 16. (a) mod6 Counter (b) Impulse Output, dlog.prescalar = 3

Figure 17. (a) mod6 Counter, (b) BemfA, (c) BemfB and (d)BemfC (dlog.prescalar = 25 and Vdcbus = 160 V)

Copyright © 2013, Texas Instruments Incorporated

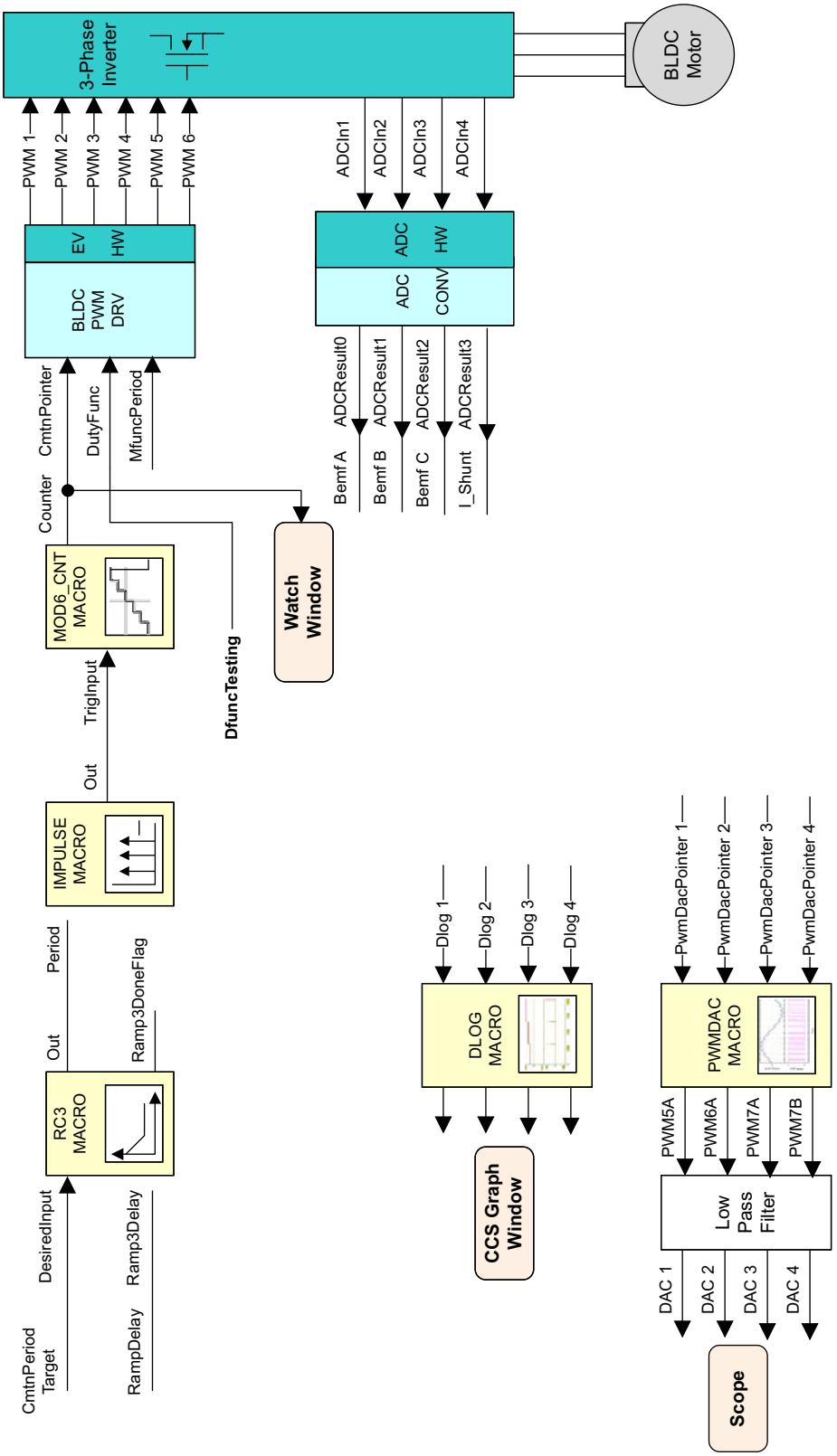


Figure 18. Level 2 – Incremental System Build Block Diagram

Level 2 verifies the open loop motor operation and the current measurement.

Copyright © 2013, Texas Instruments Incorporated

8.5 Level 3 Incremental Build

Assuming the previous section is completed successfully, this section describes the closed-loop operation of the sensored trapezoidal drive of the BLDC motor using the Hall sensor.

1. Open HVBLDC_Sensorless-Settings.h and select the level 1 incremental build option by setting the BUILDLEVEL to LEVEL3 (#define BUILDLEVEL LEVEL3).
2. Right click on the project name and click Rebuild Project.
3. Click on the debug button, reset the CPU, restart, enable real-time mode and run, once the build is complete.
4. Set the “EnableFlag” to 1 in the watch window. The variable named “IsrTicker” is incrementally increased as seen in the watch windows to confirm the interrupt working properly.

In the software, the key variable to be adjusted is mentioned below:

- DFuncDesired (Q15 format): for changing the PWM duty function in per-unit

The key steps are explained as follows:

- Compile, load, and run the program with real-time mode.
- Increase the voltage at the variac and the dc power supply to get the appropriate DC-bus voltage. Now the motor is running with the default DFuncDesired value.
- Then, the motor will be running using the newly created map for every commutation. Vary the motor speed by changing the PWM duty ratio represented by DFuncDesired. Double-click on DFuncDesired in the Watch Window, and enter the new value. This is a Q15 parameter and the max value is 0x7FFF.
- Check the calculated speed based on the Hall signals with the six times frequency of commutation trigger signals in the graph windows or the oscilloscope screen.
- Check the measured DC-bus current if it is nearly zero when the motor is operating at no-load.
- Verify the motor speed (both pu and rpm) calculated by SPEED_PR.
- Bring the system to a safe stop (as described below) by reducing the bus voltage, taking the controller out of real-time mode and reset.

Copyright © 2013, Texas Instruments Incorporated

When running this build, the current waveforms in the Code Composer Studio graphs should appear as shown in Figure 19.

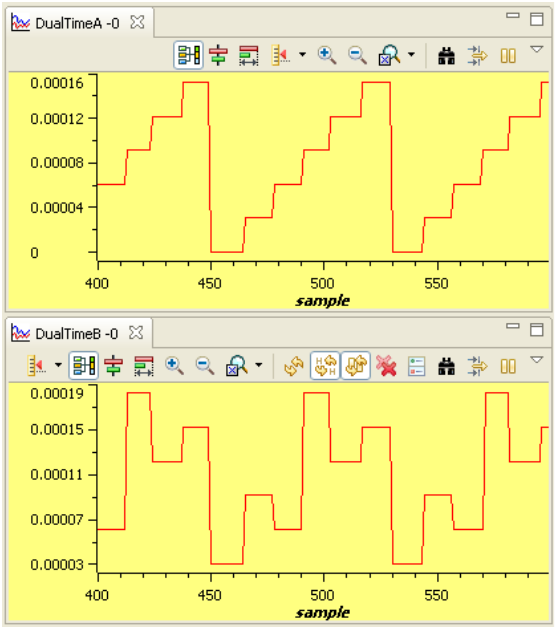


Figure 19. (a) mod6 Counter, (b) HallGpioAccepted (dlog.prescalar = 25 and Vdcbus = 160 V)

PWMDAC outputs should appear as shown in Figure 20.

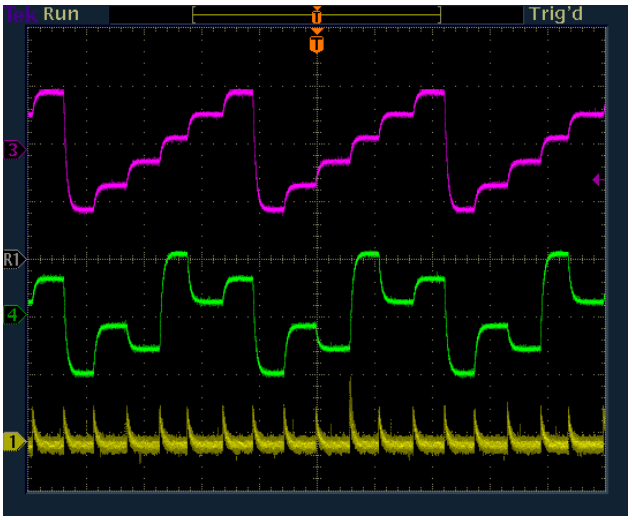


Figure 20. (a) mod6 Counter, (b) HallGpioAccepted, (c) mod1.trigInput (Vdcbus = 160 V)

Copyright © 2013, Texas Instruments Incorporated

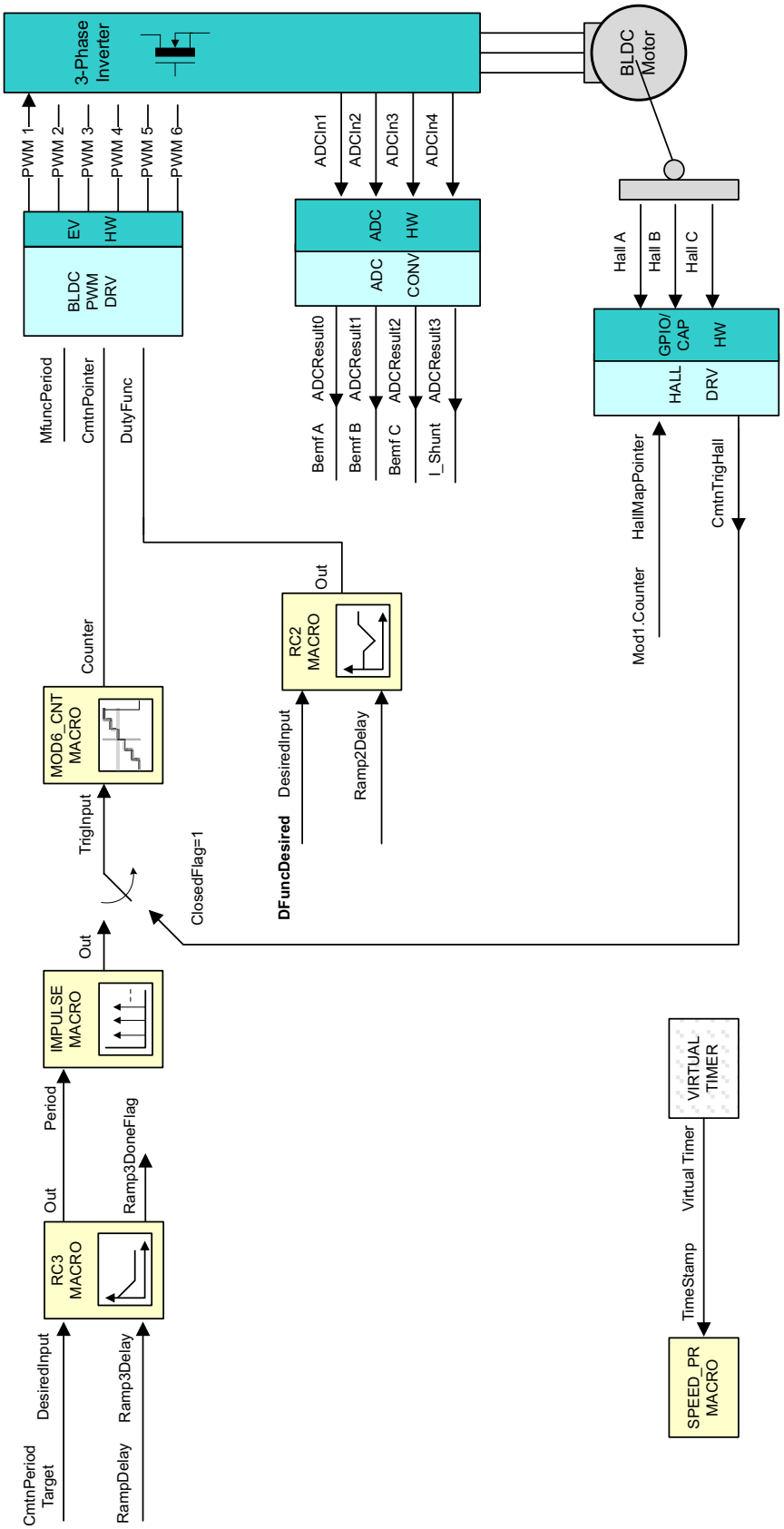


Figure 21. Level 3 – Incremental System Build Block Diagram

Copyright © 2013, Texas Instruments Incorporated

Level 3 describes the closed-loop operation of sensored trapezoidal drive of BLDC motor using Hall sensor.

8.6 Level 4 Incremental Build

Assuming the previous section is completed successfully, this section verifies the closed current loop and current PI controller.

1. Open HVBLDC_Sensored-Settings.h and select level 2 incremental build option by setting the BUILDLEVEL to LEVEL4 (#define BUILDLEVEL LEVEL4).
2. Right Click on the project name and click Rebuild Project.
3. Click on debug button, reset the CPU, restart, enable real-time mode and run, once the build is complete.
4. Set the “EnableFlag” to 1 in the watch window. The variable named “IsrTicker” is incrementally increased as seen in the watch windows to confirm the interrupt working properly.

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

- DFuncDesired (Q15 format): for changing the PWM duty cycle in per-unit
- CurrentSet (GLOBAL_Q format): for changing the reference DC-bus current in per-unit
- ILoopFlag (Q0 format): for switching between fixed duty-cycle and controlled Idc duty-cycle

The key steps can be explained as follows:

- Compile, load, and run the program with real-time mode and then increase voltage at the variac and dc power supply to get the appropriate DC-bus voltage.
- The motor will gradually speed up and finally switch to closed loop commutation mode.
- Increase the motor speed by changing DFuncDesired.
- Set SpeedRef to 0.3 pu (or another suitable value if the base speed is different).
- Use the variable CurrentSet to specify the reference current for the PI controller PI.
- Change ILoopflag to 1 to activate the current loop PI controller once the ClosedFlag is set to 1 in the code.
- Once this is done, the PI controller starts to regulate the DC bus current and the motor current.
- Gradually increase or decrease the command current (CurrentSet value) to change the torque command and adjust the PI gains. Note that the speed is not controlled in this step and a non-zero torque reference keeps increasing the motor speed. Therefore, the motor should be loaded using a brake or generator (or manually if the motor is small enough) after closing the loop. Initially, apply a 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.
- Bring the system to a safe stop (as described below) by reducing the bus voltage, taking the controller out of real-time mode and reset.

Copyright © 2013, Texas Instruments Incorporated

PWMDAC outputs should appear as shown in Figure 22.

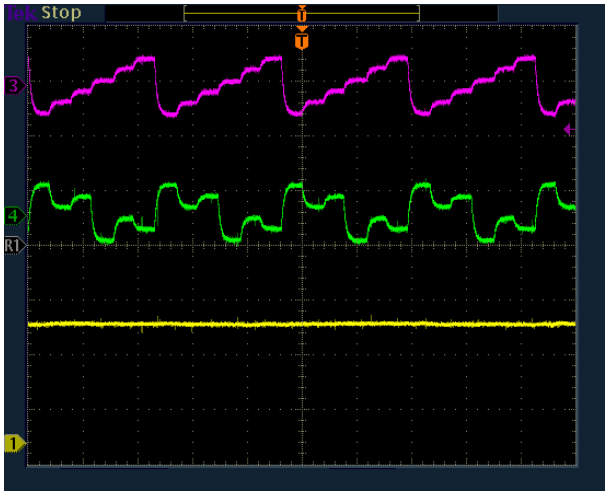


Figure 22. (a) mod6 Counter, (b) HallGpioAccepted, (c) speed, (under 0.5 pu load, Vdcbus = 160 V)

Copyright © 2013, Texas Instruments Incorporated

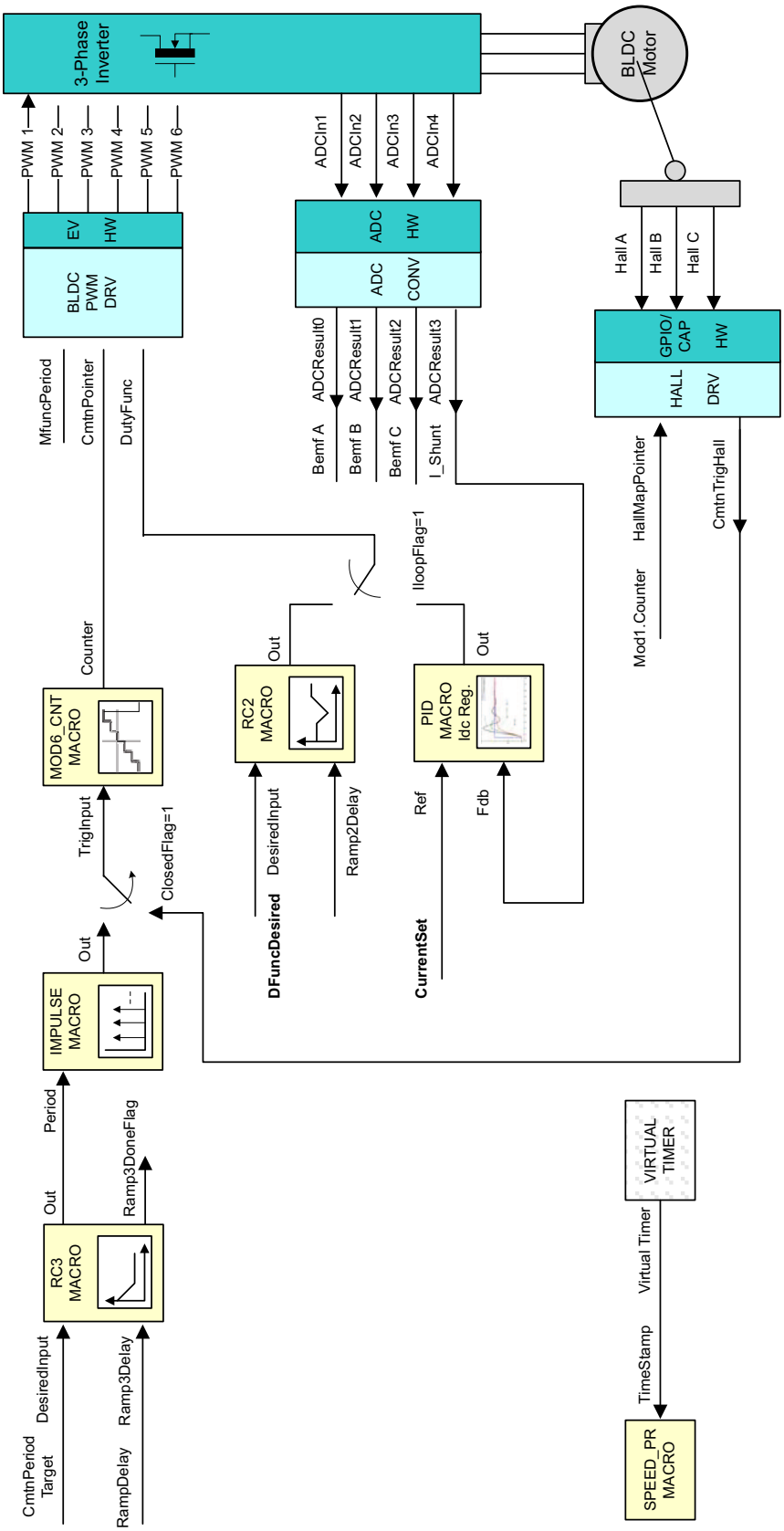


Figure 23. Level 4 - Incremental System Build Block Diagram

Copyright © 2013, Texas Instruments Incorporated

Level 4 verifies the closed current loop and current PI controller.

8.7 Level 5 Incremental Build

Assuming the previous section is completed successfully, this section verifies the closed loop speed PI controller.

1. Open HVBLDC_Sensored-Settings.h and select the level 5 incremental build option by setting the BUILDLEVEL to LEVEL5 (#define BUILDLEVEL LEVEL5).
2. Right click on the project name and click Rebuild Project.
3. Click on the debug button, reset the CPU, restart, enable real-time mode and run, once the build is complete.
4. Set the “EnableFlag” to 1 in the watch window. The variable named “IsrTicker” will now keep on increasing.
5. Confirm this by watching the variable in the watch window. This confirms that the system interrupt is working properly.

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

- SpeedRef (Q24): for changing the reference speed in per-unit

The key steps can be explained as follows:

- Compile, load, and run the program with real-time mode.
- Increase the voltage at the variac and the dc power supply to get the appropriate DC-bus voltage.
- The motor gradually speeds up and finally switches to closed loop commutation mode.
- Use the variable SpeedRef to specify the reference speed for the PI controller PI. The SpeedLoopFlag is automatically activated when the PI reference is ramped up from zero speed to SpeedRef. Once this is done, the PI controller starts 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 at the end of build 1 by reducing the bus voltage, taking the controller out of realtime mode and reset.

When running this level, the current waveforms in the Code Composer Studio graphs should appear as shown in Figure 24.

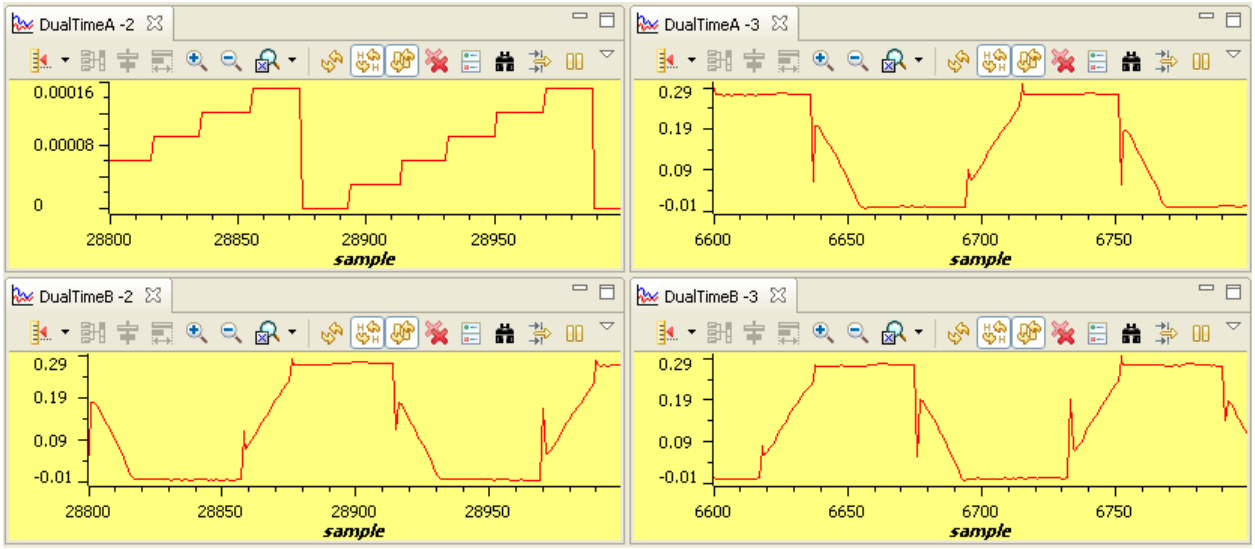


Figure 24. (a) mod6 Counter, (b)BemfA, (c) BemfB (c)BemfC (under no-load at 0.5pu speed, Vdcbus = 160 V)

Copyright © 2013, Texas Instruments Incorporated

PWMDAC outputs should appear as shown in Figure 25.

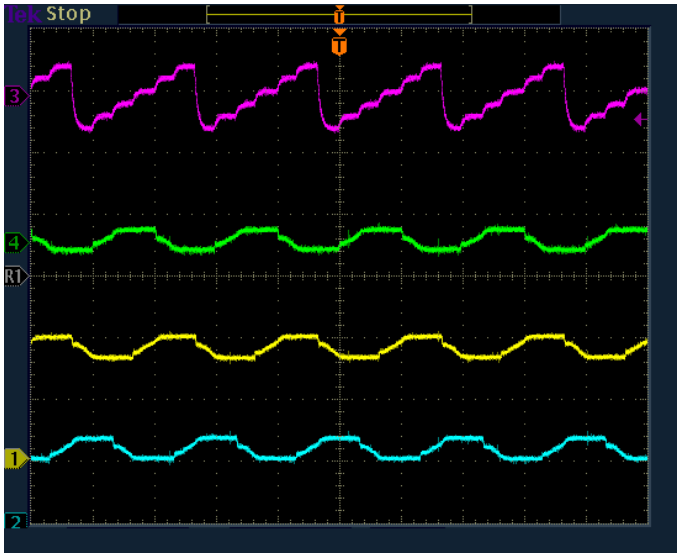


Figure 25. (a) mod6 counter, (b)BemfA, (c) BemfB (c)BemfC (under 0.5 pu load at 0.3pu speed, Vdcbus = 160 V)

Copyright © 2013, Texas Instruments Incorporated

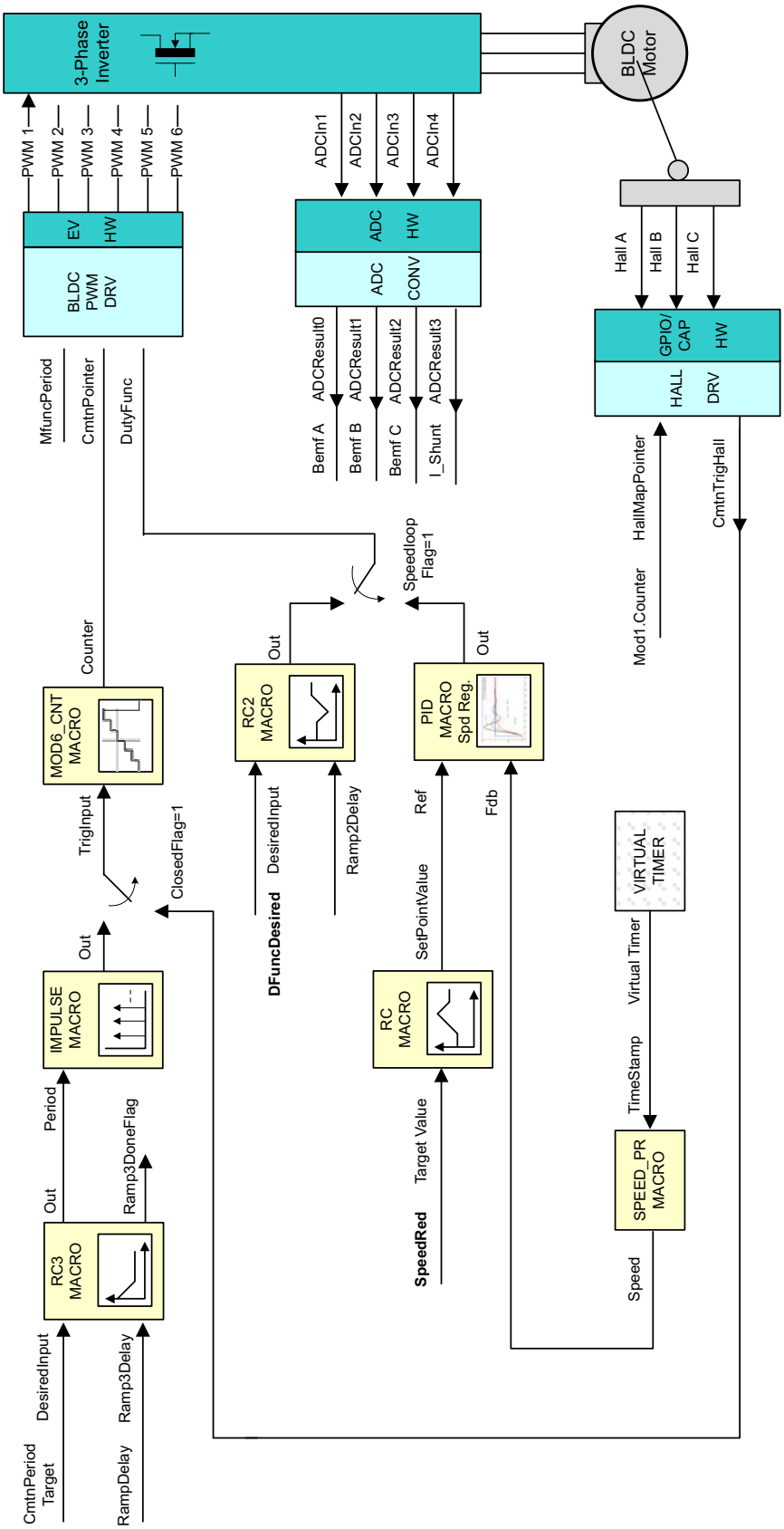


Figure 26. Level 5 - Incremental System Build Block Diagram

Copyright © 2013, Texas Instruments Incorporated

Level 5 verifies the closed speed loop and speed PI controller.

9 References

- *Optimizing Digital Motor Control (DMC) Libraries* (SPRAAK2)

Copyright © 2013, Texas Instruments Incorporated

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2013, Texas Instruments Incorporated