TI Designs

Three-Phase BLDC and PMSM Motor Drive With High-Performance Microcontrollers Design Guide

TI Designs

TI Designs provide the foundation that you need including methodology, testing and design files to quickly evaluate and customize the system. TI Designs help *you* accelerate your time to market.

Design Resources

TIDM-RM46xDRV8301KIT	Design Folder
RM46L852	Product Folder
DRV8301-Q1	Product Folder
TPS65381-Q1	Product Folder
TPS73633-EP	Product Folder
TXB0106	Product Folder
OPA365	Product Folder
ISO7241C	Product Folder
ISO1050	Product Folder
SN74LVC2G17	Product Folder
DCH010505S	Product Folder
TLV2781	Product Folder
DRV8301-RM46-KIT	Tools Folder

TI E2E™ Community

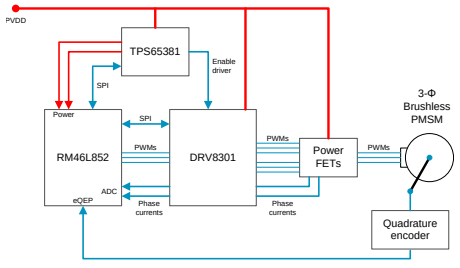
ASK Our E2E Experts
WEBENCH® Calculator Tools

Design Features

- Describes the DRV8301-RM46-KIT
- Includes a RM46x Control Card mounted on a DRV8301 Evaluation Board (EVM)
- Offers a RM46x Control Card That Includes a Power-Management IC, TPS65381-Q1
- Offers a Kit That Includes a 6000-rpm, 7.1-A Teknic Motor With a Built-in Quadrature Encoder
- Includes the Following Software Examples in the Design Kit:
  - Encoder-Based FOC With Redundant-Sensorless Speed Estimation
  - Speed and Torque Control Loops
- Leverages an ARM® CMSIS Math Library

Featured Applications

- Industrial Automation
- Robotics
- Electronically Controlled Pumps
- Medical Electronics



An IMPORTANT NOTICE at the end of this TI reference design addresses authorized use, intellectual property matters and other important disclaimers and information.

1 System Description

The TIDM-RM46xDRV8301KIT describes a system for evaluating the capabilities of Hercules™ microcontroller units (MCUs) for controlling three-phase brushless DC (BLDC) and brushless AC (BLAC) motors (often referred to as permanent-magnet synchronous motors [PMSM]). The kit features an RM46L852 Hercules MCU. The design files in this TI Design include the board schematics, bill of materials (BOM), board layout files, and reference code projects to evaluate the performance of the MCU in this application.

1.1 RM46L852 Hercules Microcontroller

The RM46L852 device is part of a high-performance microcontroller family for industrial applications. The architecture includes dual ARM® Cortex™-R4F processors working in lock-step, a CPU self-test, a memory self-test, an ECC for both flash and data SRAM, parity on peripheral memories, and a loopback capability on peripheral I/Os. The CPU can run up to 220 MHz, providing up to 365 DMIPS of processing power. The RM46L852 MCU has 1.25 MB of on-chip flash memory and 192 KB of on-chip SRAM. Figure 1 presents a block diagram of the key features of the RM46L852 microcontroller.

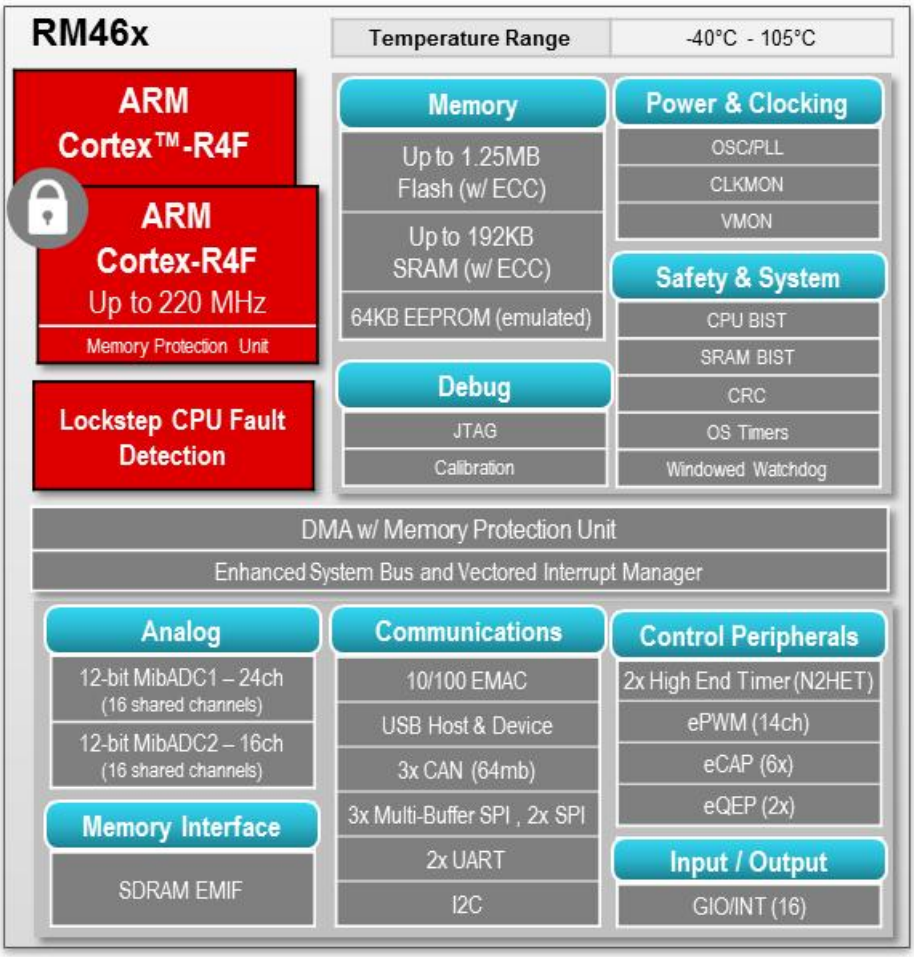


Figure 1. RM46L852 Features

In the example application that accompanies this TI Design, the RM46L852 MCU executes the speed- and torque-control loops. The MCU uses a serial peripheral interface (SPI) port to configure the power-management IC (PMIC) and manage the features inside this PMIC. The MCU also uses a separate SPI port to configure the DRV8301 gate driver IC. An on-chip 12-bit ADC senses the motor phase currents and the DC-bus voltage. The enhanced quadrature pulse (eQEP) module decodes quadrature-encoded speed and position information from the encoder built into the Teknic motor. The on-chip enhanced PWM generator (ePWM) modules generate pulse-width-modulated (PWM) signals to the DRV8301.

1.2 TPS65381-Q1 Multirail Power Supply

The TPS65381-Q1 is a PMIC designed for Hercules MCUs. This PMIC is qualified for automotive applications that require support for ambient temperatures from –40°C to 125°C. For an overview of the TPS65381 functional block diagram, see Figure 2.

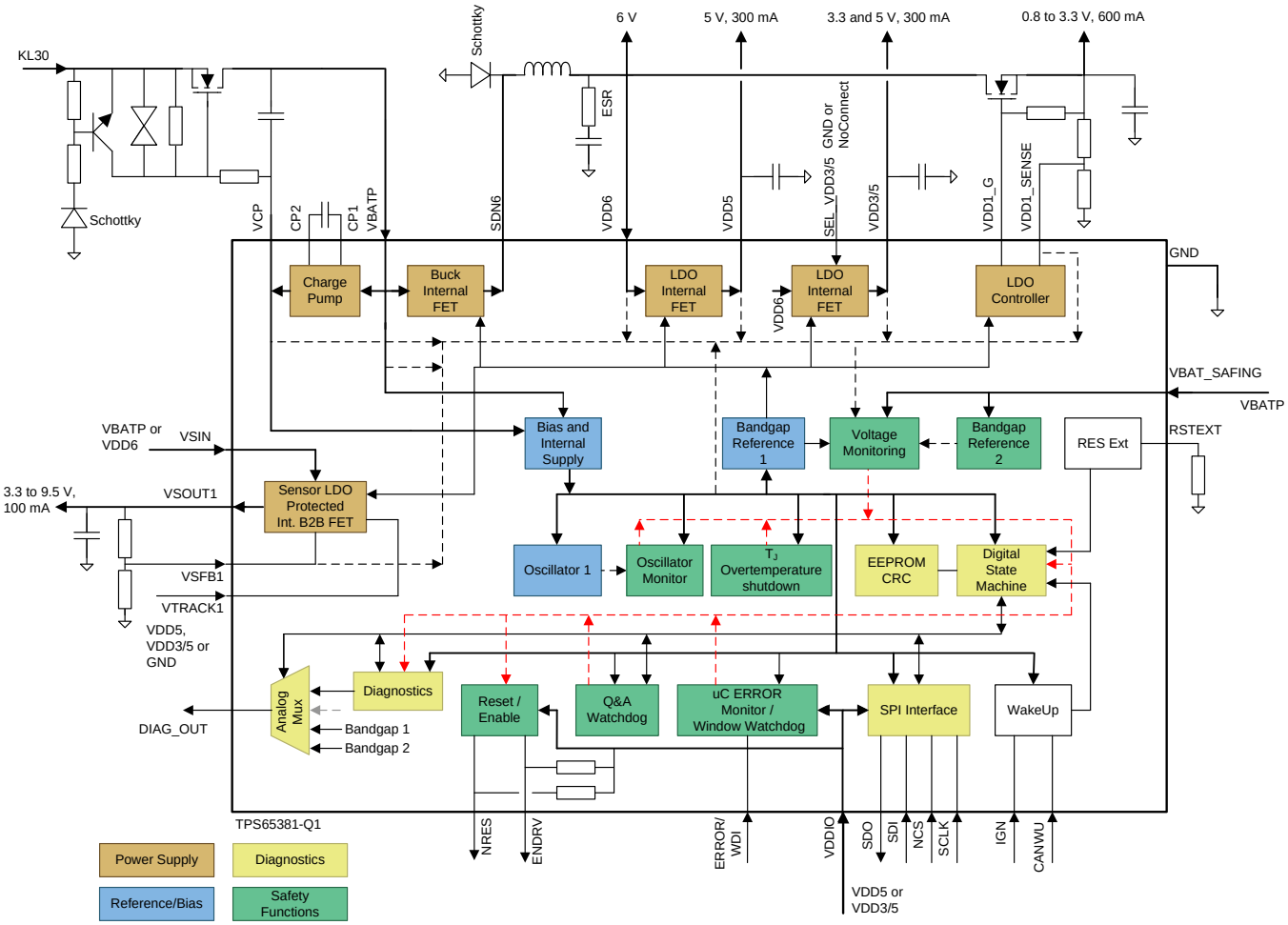


Figure 2. TPS65381 Functional Block Diagram

The PMIC supports input voltages from 5.8 V to 36 V and generates the 3.3-V and 1.2-V supplies required by the Hercules MCU. The PMIC incorporates independent undervoltage and overvoltage monitoring on all outputs, battery-voltage input, and all internal supplies. The TPS65381-Q1 communicates with the RM46x Hercules MCU through an SPI. The SPI configures the entire PMIC. In this TI Design, the TPS65381-Q1 is a companion IC. The PMIC features a question and answer watchdog function that is serviced through the SPI port by the RM46x MCU. The PMIC also monitors the nERROR signal output from the RM46x MCU, which indicates a severe error inside the MCU. The TPS65381-Q1 is targeted for industrial applications like PLCs, I/O control modules, elevator and escalator control modules, and wind turbine control modules.

1.3 DRV8301 Three-Phase Gate Driver

The DRV8301 is a gate driver IC for three-phase motor drive applications. This device provides three half-bridge drivers that can drive two N-channel MOSFETs. The DRV8301 supports up to a 1.7-A source and a 2.3-A peak-current capability and can operate from a power supply with a range from 6 V to 60 V.

The DRV8301 gate driver IC also includes the following protection features:

- Programmable Dead Time Control (DTC)
- Programmable Overcurrent Protection (OCP)
- Power Supply Undervoltage Lockout (UVLO)
- Overtemperature Warning/Shutdown (OTW/OTS)

The DRV8301 uses a dedicated SPI port to communicate with the RM46x Hercules MCU. The RM46x MCU can query the DRV8301 registers through this SPI port to identify the cause of any fault indication from the DRV8301. This fault indication from the DRV8301 is connected to a trip-zone input of the RM46x MCU. This connection lets the system designer automatically disable the PWM driven from the RM46x MCU to the DRV8301. This connection also lets an interrupt generate on the RM46x MCU whenever the DRV8301 indicates a fault condition. For an overview of the DRV8301 three-phase gate driver functional block diagram, see Figure 3.

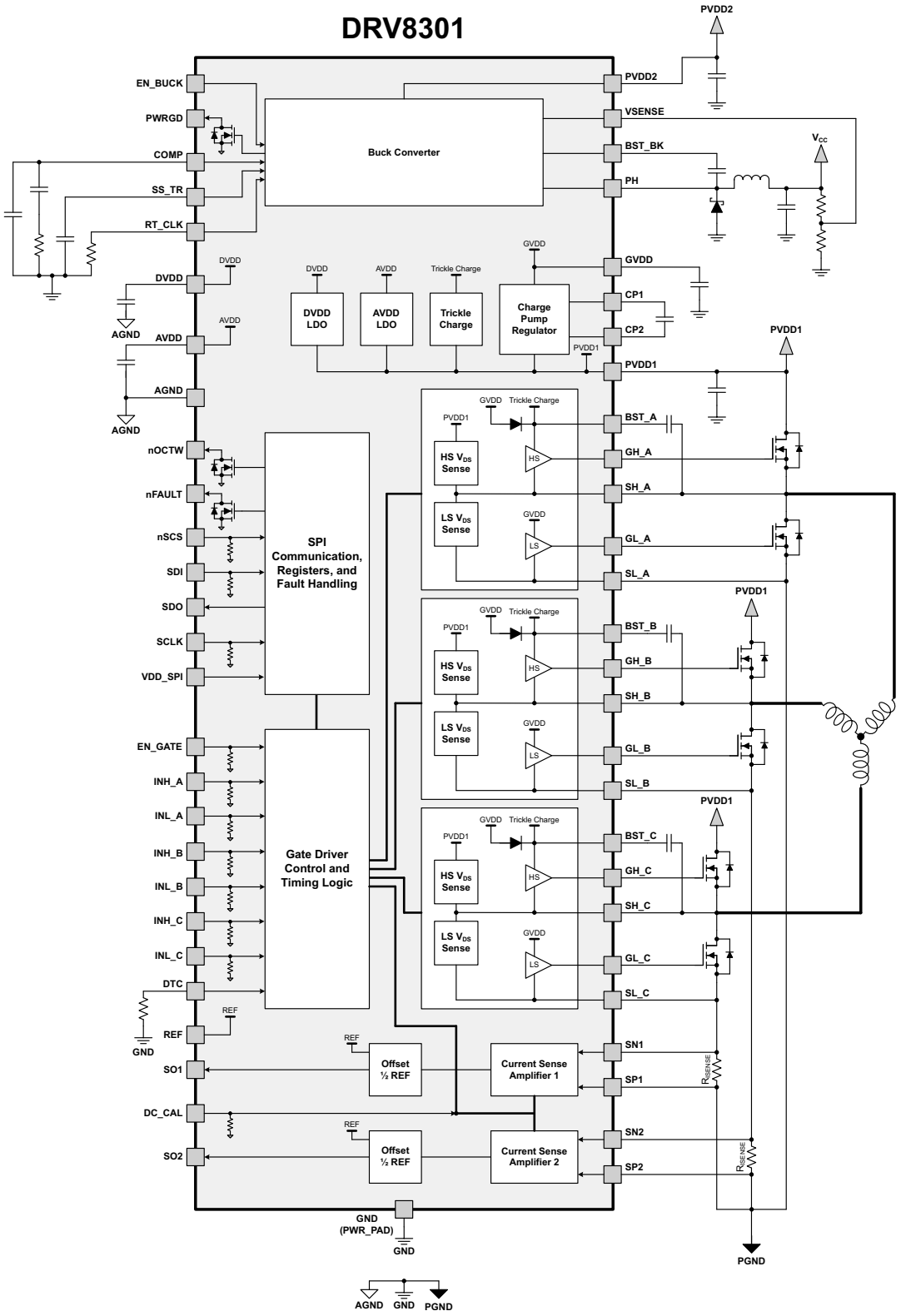


Figure 3. DRV8301 Three-Phase Gate Driver Functional Block Diagram

2 System Block Diagram

For a view of the interconnectivity between the various components in the three -phase brushless PMSM drive, see Figure 4. The features for these components, described in Section 1, make these components uniquely capable of implementing a motor drive application.

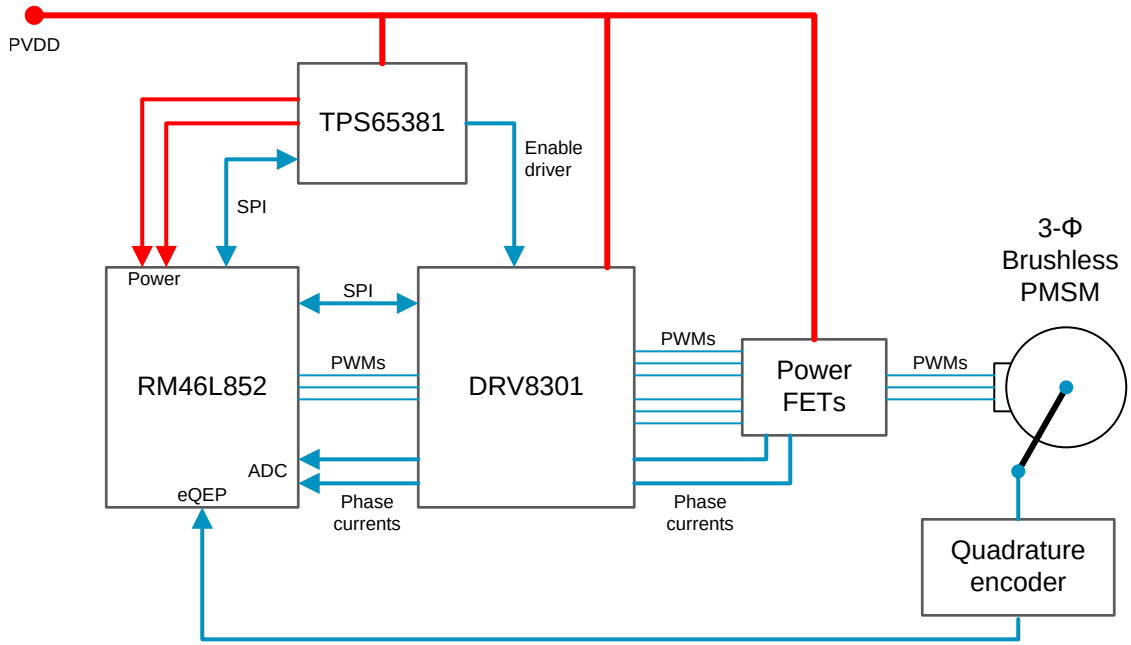


Figure 4. 3-Phase Brushless Permanent Magnet Synchronous Motor Drive Implementation

3 System Design Theory

In this section, sensored and sensorless field-oriented control (FOC) of brushless PMSMs are demonstrated and the performance of the speed controller is examined. An DRV8301-EVM board with an DRV8301 gate driver IC through external H-bridge stages drive the motor. The RM46x Hercules MCU control card generates three PWM signals for the high-side gate drive. The motor-phase currents and DC-bus voltage are sensed by the analog-to-digital converters (ADCs) on the RM46x MCU.

3.1 RM46x Features

Application Characteristic	Description
Development/Emulation	Code Composer Studio v5.2.1 (or above)
Target Controller	Hercules RM46L852
PWM Frequency	20-kHz PWM (Default)
PWM Mode	PWM control of high-side gates only (that is, three independent signals). Low side is complementary to high-side gates with programmable dead-band.
Interrupts	adc1Group1Interrupt() adc1Group2Interrupt()
Peripherals	NHET – PWM Generation, QEP reading, and SPI configuration of the gate driver GIO – Fault diagnostics, gate driver enable and disable RTI – Time base for delays and profiling ADC – Motor phase currents and DC bus voltage

3.2 CPU Memory Usage

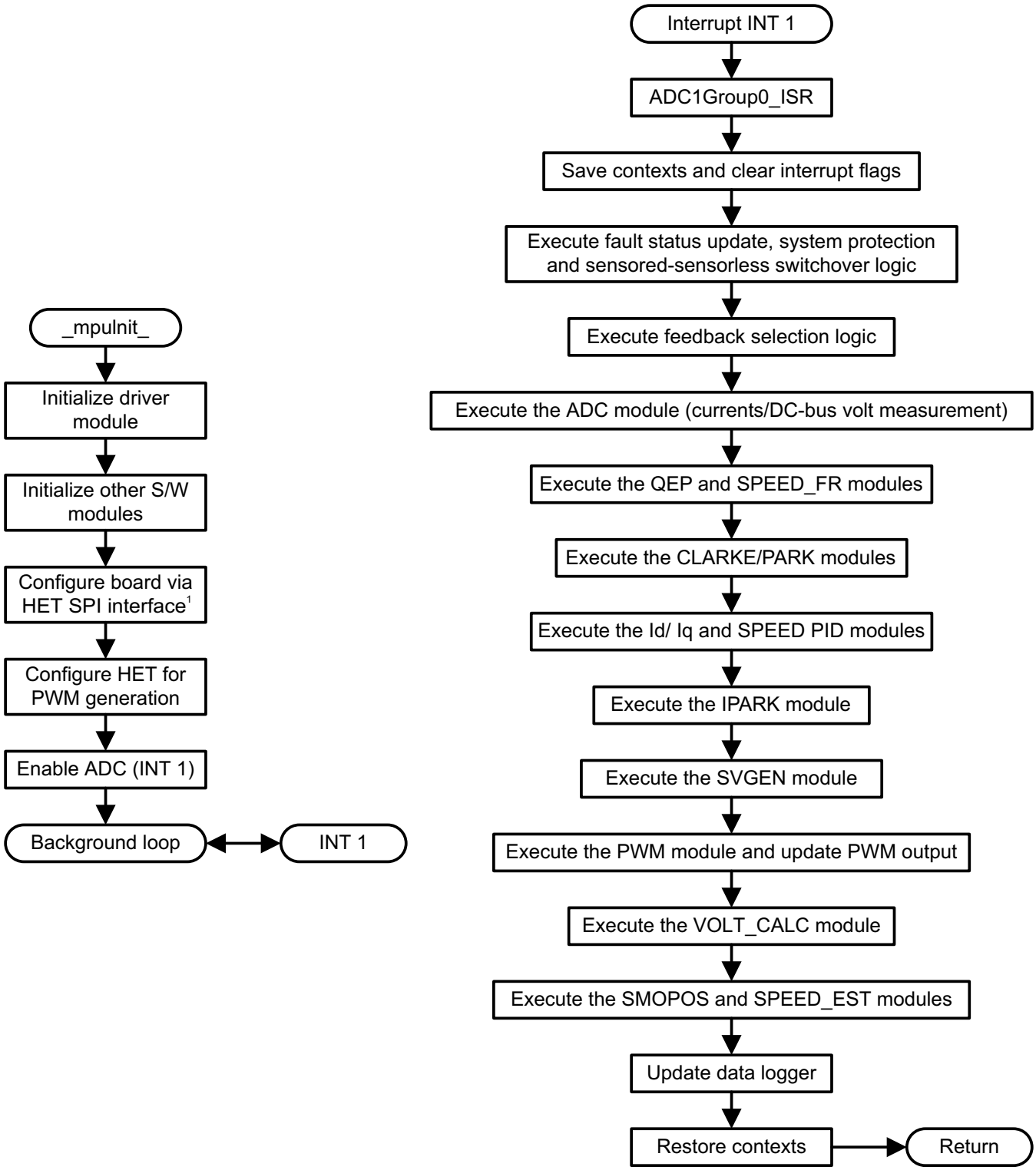
System Name	Flash Memory Usage RM46x	RAM Memory Usage RM46x
RM46L852_sensored_speed_smo	25,984 bytes	5836 bytes

3.3 CPU Bandwidth

For CPU bandwidth data, see Section 7.

3.4 Software Flowchart

For an overview of the software flow in this demonstration code project, see Figure 5.



(1) The DRV8301 gate driver IC settings, like current amplifier gains, PWM mode, and diagnostic reporting, are configured through the SPI interface. When the gate-driver IC configuration completes through the SPI, the PMIC asserts the gate driver enable signal (EN_GATE) of the DRV8301 IC. The RTI timer module counts down a 50-ms delay for the gate driver power up (that is, charge pumps, internal regulator, and current amplifiers). After initialization of the remaining driver and software modules, the SPI port issues commands to configure the DRV8301 IC.

Figure 5. Software Flowchart

3.5 Software Modules Block Diagram

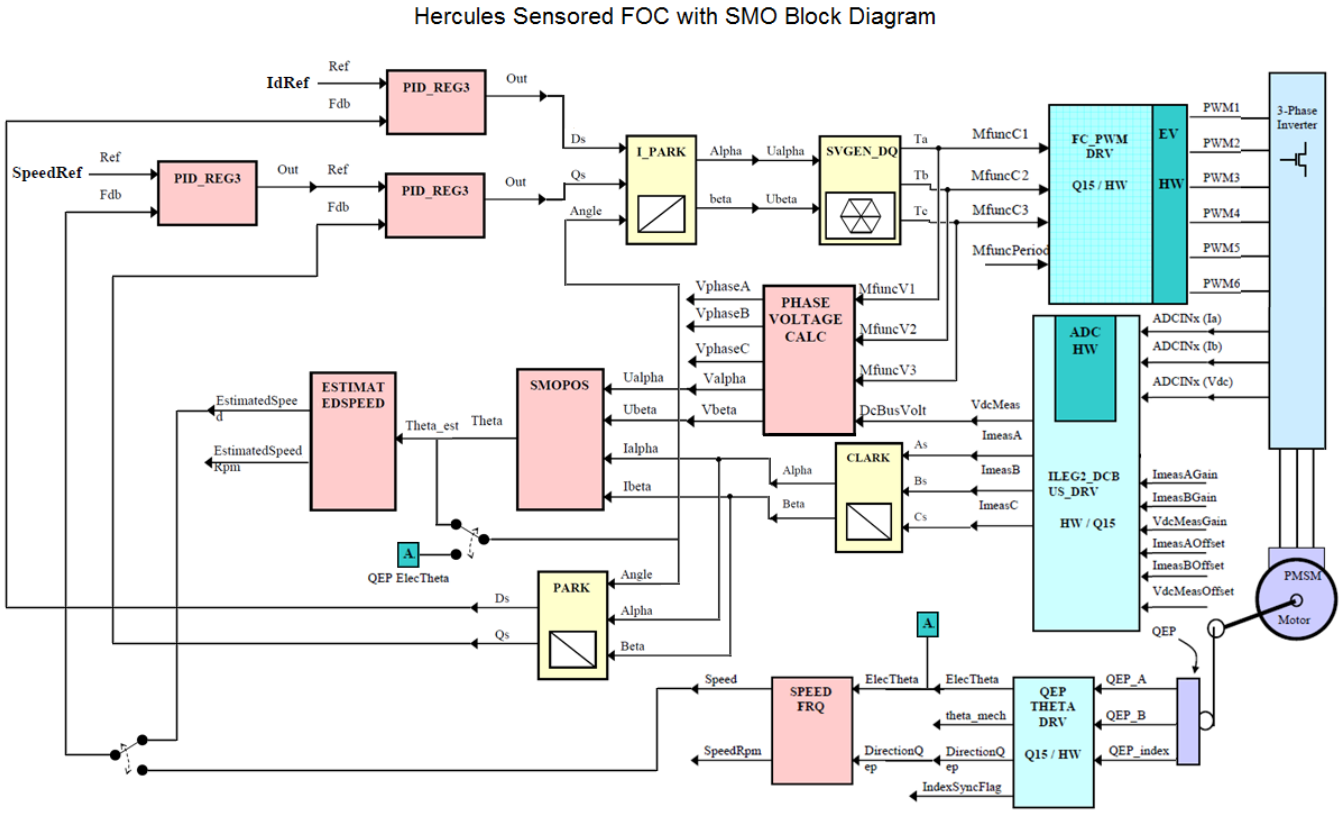


Figure 6. Hercules-Sensored FOC With SMO Block Diagram

4 Getting Started Hardware

This TI Design kit consists of the following macro blocks:

- RM46x control card
- DC bus connection
  - PVDD/GND terminals to connect the external 8-V to 60-V power supply
- DRV8301 gate driver IC
- Current-sense circuitry – low-side shunt current sensing on each half-bridge
- Quadrature encoder connections
- Hall-effect sensor connections

4.1 DRV8301-EVM Board

To view the position of these macro blocks on the DRV8301-EVM board, see Figure 7. All the PWMs and ADC signals that are actuation and sense signals have designated test points on the board. This makes it easy to try out new control algorithms and strategies.

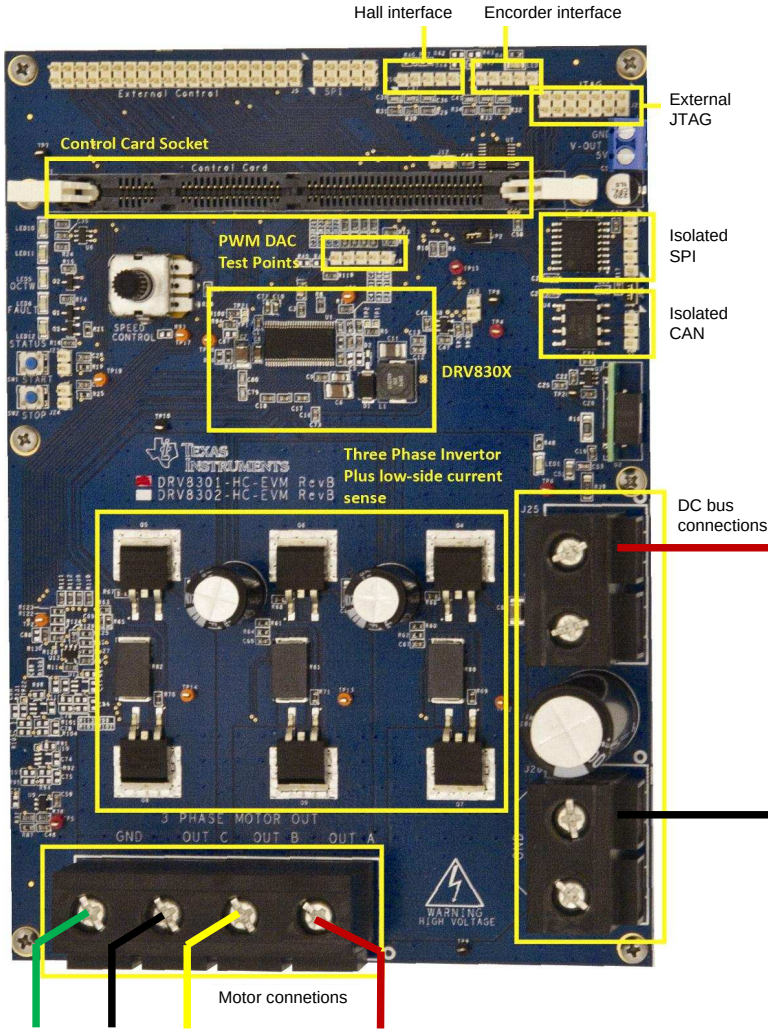


Figure 7. DRV8301-EVM Connections and Settings

The DRV8301-EVM board has two power domains: the low-voltage controller power domain that powers the controller and the logic circuits on the board, and the medium-voltage power delivery line that carries the medium-voltage and current (like the DC power for the inverter or DC bus).

4.2 RM46L852 Control Card

The TMDXRM46CNCD should be part of a motor control kit. The control card offers additional control, connectivity, and evaluation features.

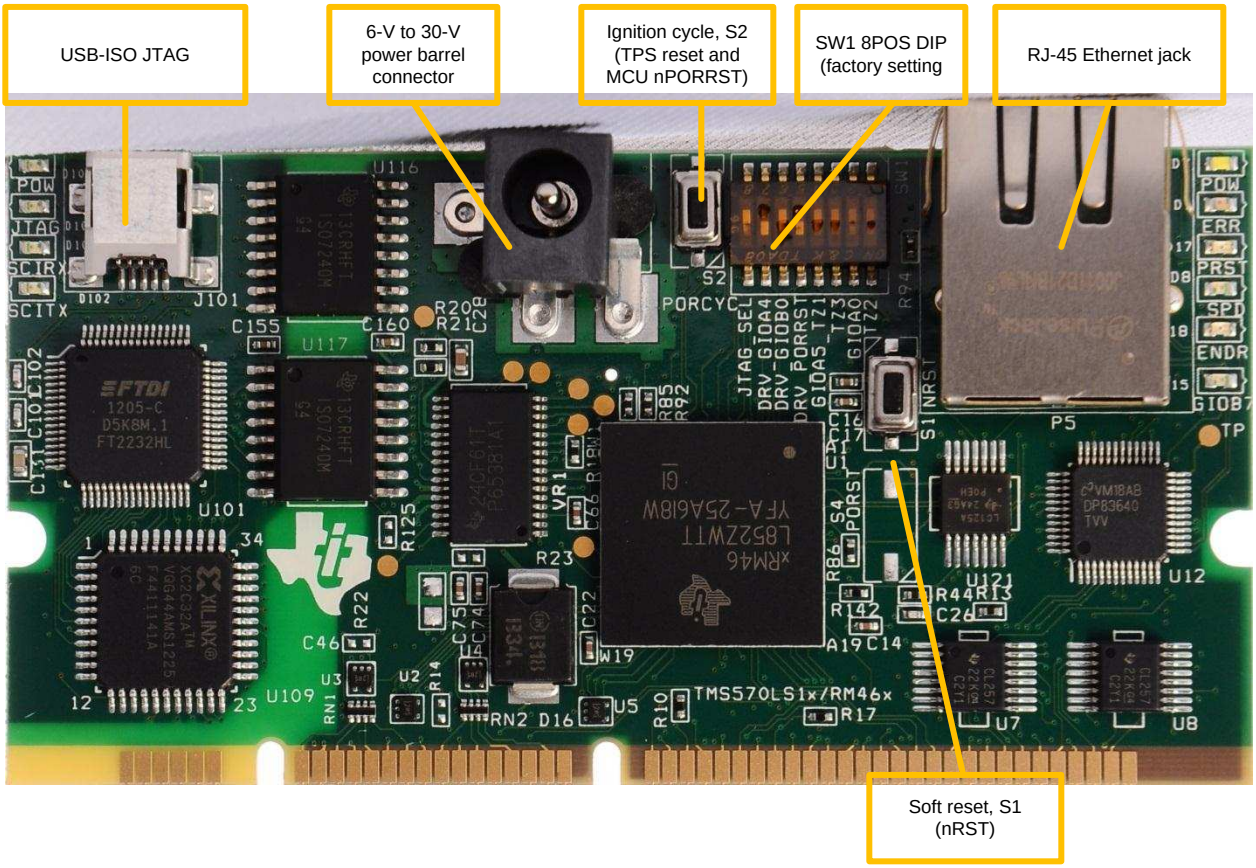


Figure 8. RM46L852 Control Card Features

Key features include:

- A TI RM46L852 337-ball BGA microcontroller
- A TI TPS65381 integrated-power companion
- Onboard, isolated USB XDS100v2 JTAG emulator
- Optional path to EVM JTAG connector (through DIMM)
- Hardware option for routing N2HET timers or ePeripherals to the DIMM interface (includes HET monitoring of ePeripheral outputs)
- 10/100 Mbps Ethernet interface through RJ-45 with same PHY as on the TMDXRM46HDK
- Isolated UART/SCI accessible through a USB virtual port (VCP)
- Ignition cycle simulator switch S2 (serves as system reset of TPS and MCU)
- LPO_TEST (SW3) push-button switch (causes CLKDET hardware fault on MCU)
- LED indicators for xds100 power, activity, target/MCU power, GIOB7 pin activity, Ethernet link and activity, Ethernet speed, and nERROR.
- Soft-reset push button (nRST)
- Onboard power supply supporting a 6-V to 30-V DC input and producing 6 V, 3.3 V, and 1.2 V for the MCU

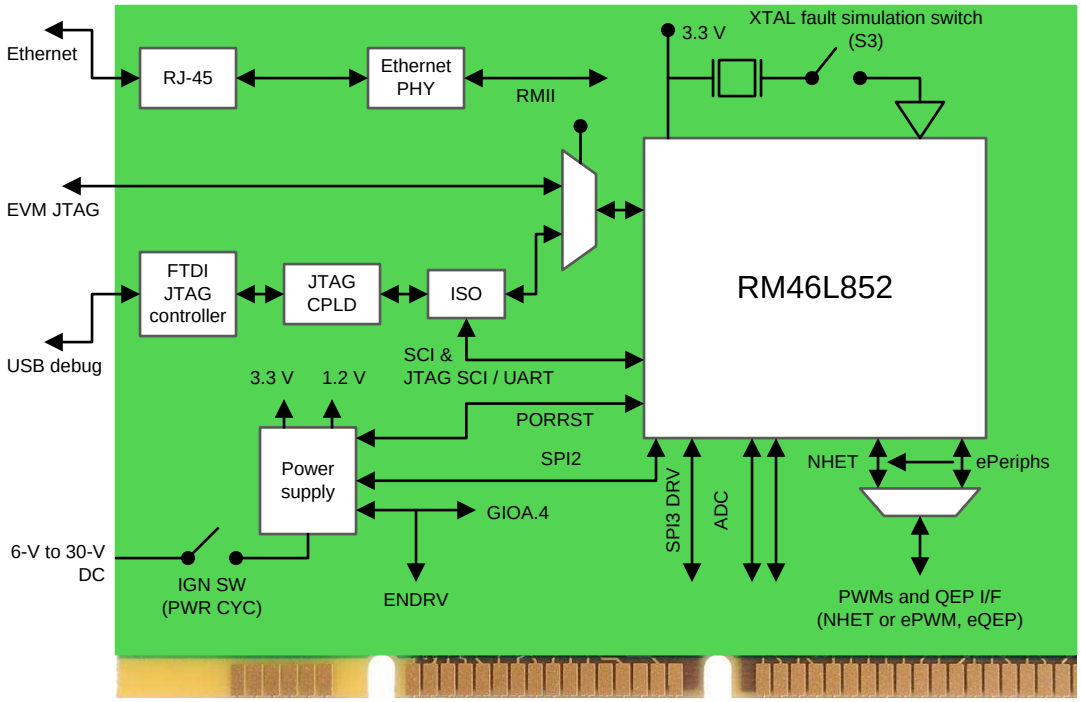


Figure 9. RM46x MCU Interfaces on the Control Card

4.3 Powering the Board

WARNING

TI intends this EVM is meant to be operated in a lab environment only. TI does not consider this EVM to be a finished end-product fit for general consumer use.

TI intends only by qualified engineers and technicians familiar with risks associated with handling high-voltage electrical and mechanical components, systems, and subsystems to use this EVM.

If not properly handled or applied, this equipment operates at voltages and currents that can shock, burn, and/or injure you or your property. Use the equipment with caution and employ safeguards to avoid being injured or damaging property.

Confirm that the voltages and isolation requirements are identified and understood before energizing the board and/or simulation. When energized, do not touch the EVM or components connected to it.

Before power is applied to the board, do the following:

1. Ensure the board is free of connections and unpowered.
2. Insert the RM46L852 control card in the J1 slot on the DRV8301-EVM.
3. Ensure the jumper JP2 is installed to supply 5 V to the RM46x control card.
4. Connect a USB cable to the J101 connecter on the RM46x control card.
5. Connect the hall-effect sensor feedback cable to the J10 connector.
6. Connect the quadrature-encoder feedback cable to the J4 connector.
7. Connect the R, S, and T motor phases to the OUTA, OUTB, and OUTC terminals, respectively.
8. Connect the shield (drain) wire to the ground terminal.
9. Connect a 24-V power supply to J25 (+ terminal) and J26 (– terminal) of the DRV8301-EVM (see Figure 7). (LED1 and LED3 should light up. The control-card LEDs should also light up, indicating the control card is receiving power from the board.)

NOTE: The power supplies included with this TI Design kit can supply a maximum of 2.5 A and may not provide a current spike when rapidly changing speeds or loads, especially with motors that have a current rating higher than 2.5 A. Consider this when evaluating the full performance of the system and use a power supply that meets the demands of the motor.

5 Getting Started Firmware

This section describes the folder structure in the demonstration code project and includes setup instructions required to run the demonstration.

5.1 Directory Structure

All the source and header files for the software project are located in the src sub-folder under the RM46L852_sensored_smo_new project folder.

5.2 Getting Started With Code Composer Studio™

This demonstration code project was built using Code Composer Studio version 6.1.0.00104 (CCSv6).

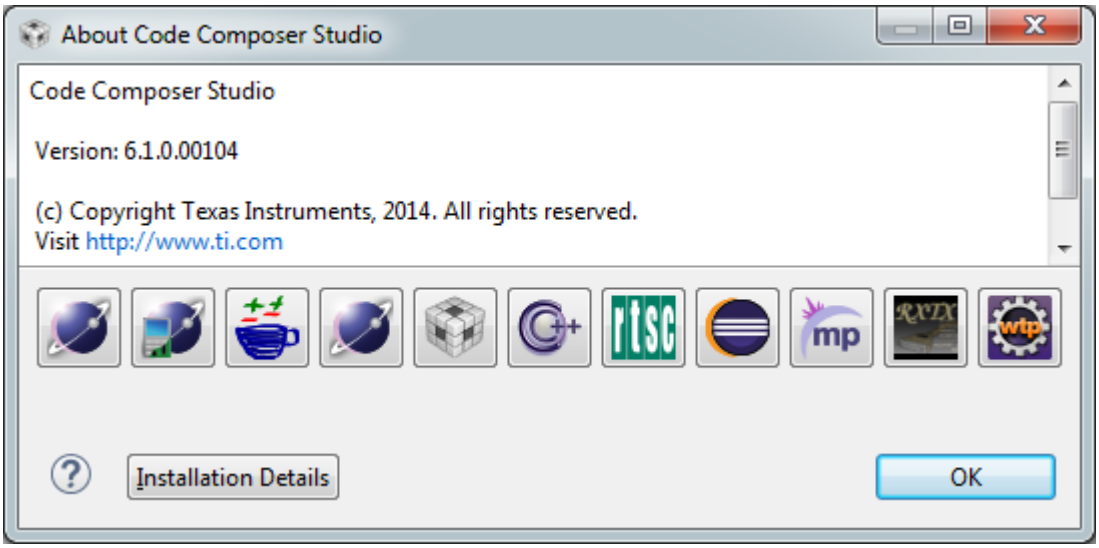


Figure 10. Code Composer Studio Version Information

For detailed information on getting started with CCSv6, see the CCSv6 Getting Started Guide.

This guide discusses all start-up steps, from downloading to running CCSv6 for the first time.

5.3 Importing Demonstration Code Project into CCSv6

When CCSv6 has been downloaded, installed, and is running, import the demonstration code project into the debug workspace.

To import the project into the debug workspace, do the following:

1. Click Project.
2. Click Import Existing CCS/CCS Eclipse Project.
3. Browse to the folder where the software examples zipped file is unzipped
4. Find the project named RM46L852_sensored_smo_new.
5. Click the checkbox next to this project name.
6. Click Finish to import the project into your workspace.

5.4 Building the Demonstration Code Project and Downloading to RM46x MCU

To build the demo code project and download it to the MCU, do the following:

1. Ensure the USB cable is plugged into your PC and into the connector on the RM46x control card.
2. In the project explorer window, right click on the project name.
3. Select Clean Project.
4. Click Build Project.
5. When the build completes, click Run.
6. Click Debug to launch a debug session to load the code into the MCU. (Alternatively, download the code into the MCU by clicking Run, then Load, then Load Program.)

5.5 Setting Up the Debug Environment for the Demonstration Code Project

To set up the debug environment for the demo code project, do the following:

1. Press Run to start code execution on the target.
2. Right click inside the Debug window.
3. Select Show All Cores. (This shows ICEPick and Dap as Nondebuggable Devices in the Debug window.)
4. Click on the line containing the Dap.
5. Click Tools.
6. Click GEL Files.
7. Right click on the GEL Files window.
8. Select Load GEL…
9. Choose the GEL file: RM46L852_sensored_smo_new\
10. Click Scripts.
11. Click DAP_Access.
12. Click LoadSymbolsForDAP.
13. Click View.
14. Click Expressions. (The Expressions tab of the CCS Watch window pops up [if not already open].)
15. Right-click inside the Expressions window.
16. Select Import.
17. Choose the file: RM46L852_sensored_smo_new\
18. Click  on the top-right corner of the Expressions window to enable Continuous Refresh.
19. Click Tools.
20. Click Graph.
21. Click Single Time to open a Graph Properties window.
22. Select Import at the bottom of this window.
23. Select the Graph1.graphProp file from the same location as the watch window variables file
24. Follow the same procedure to select the Graph2.graphProp file.
25. Click  on the top-right corner of the graph windows to enable Continuous Refresh.

5.6 Enabling the TPS65381 PMIC With Watchdog

The PMIC powers the RM46 control card and includes a watchdog that monitors the operation of the RM46x. Disable the watchdog within 600 ms of the power-up or it will require servicing. If the PMIC is enabled and the RM46x control card fails to disable the watchdog within 600 ms, you will be unable to connect to the control card. The demonstration software project requires the PMIC to be enabled to run correctly.

When the PMIC is enabled, the demonstration software detects it and disables the watchdog within the first 600 ms of power-up. The software will then wait for user input. Because the PMIC controls the enable signal of the DRV8301 gate driver, the system waits for the you to enable the gGUIObj.TPSFlag in the watch window. Enabling this flag lets the demonstration software use the SPI port to enable the TPS65381 watchdog. The PMIC activates the DRV8301 that powers up the motor inverter. You must enable to watchdog to power the motor inverter. When the watchdog is enabled, it must be serviced through the SPI regularly or power to the processor will be disabled. Though the demonstration software handles the servicing if you halt the code execution in CCS or set a break point, the TPS65381 will power cycle and you will lose your connection to the processor.

5.7 System Calibration Sequence

After enabling the TPS65381, click Run. After clicking Run, the system accepts commands. During run time, the motor can be enabled or disabled using the gGUIObj.EnableFlg variable (0 = Disabled, 1 = Enabled). The default state of the variable is 0 (that is, disabled).

The system calibration routine calculates the offset (in terms of encoder ticks) between the encoder index position and the motor zero pole pair position. The routine is calculated between these positions to align the direct and quadrature current vectors applied to the motor. In the demonstration software, the key variables to be adjusted and/or observed for the system calibration routine are the following:

- gGUIObj.TPSFlag — enables the drive to the motor inverter
- gGUIObj.EnableFlg— enables or disables the motor
- gGuiObj.calibrateEnable — enables or disables the motor encoder index calibration routine
- gGUIObj.CommAngleOffset — observes the calculated encoder index angle offset
- drv.commutationMode — selects the motor commutation mode

To run the calibration routine, do the following:

1. Set gGUIObj.TPSFlag to 1. (This enables the output from the DRV8301 gate driver IC to the motor inverter.)
2. Set gGUIObj.EnableFlg to 1. (This starts the main interrupt service routine for the field-oriented-control algorithm, enables the ADC sampling of the motor-phase currents and DC-bus voltage, and enables the PWM outputs. For expected PWM outputs, see Figure 11. If no PWM output is observed, check that Continuous Refresh is enabled on the Expression window, and that the CPU is not halted.)

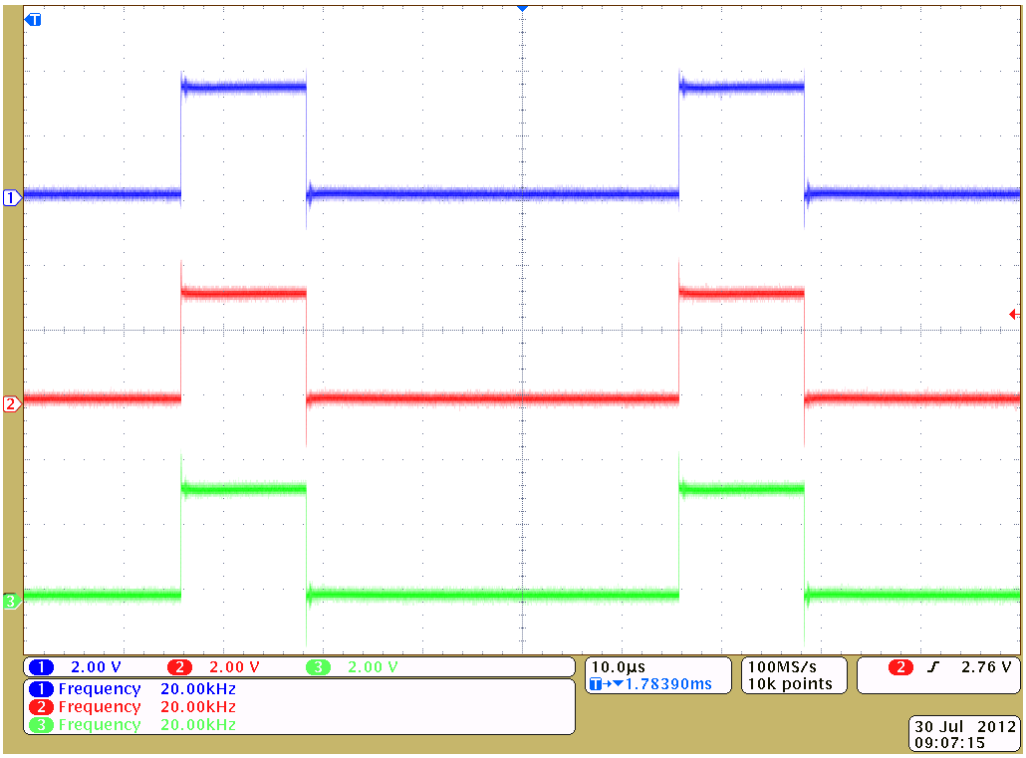


Figure 11. PWM Output on Pins 25, 26, and 28 of Connector J5

3. Set gGuiObj.calibrateEnable to 1 (This causes the motor to start ramp-commutating to determine the index position of the encoder. When the index is discovered, the rotor is locked in position to read in the offset between the encoder index and zero-pole-pair-position. At this point, the variable drv.commutationMode is set to be LOCK_ROTOR_MODE. The motor rotates once and then stops.)
4. Set gGUIObj.calibrateEnable to 0. (The variable gGUIObj.CommAngleOffset will now have a value in the range from 160 to 180. This is the calculated encoder index angle offset.)

The system is now ready for motor control commands. This calibration routine must be executed once during motor power up. If the commutation angle offset is known from a prior calibration sequence run, it can be used to directly update the drv.calibrateAngle variable.

5.8 Configuring Motor Control Methods

The gGUIObj.CtrlType variable configures the mode to control the motor. For the possible values for enabling motor control mode, see Table 1.

Table 1. Selecting Motor Control Modes

gGUIObj.CtrlType	Motor Control Mode
0 (default)	Disabled
1	Closed loop Torque Control
2	Closed loop Speed Control

NOTE: The motor control must be disabled (gGUIObj.CtrlType = 0) before switching between any other mode. When the motor control mode changes from disabled to any other mode, the motor ramp commutates to find the encoder index position. When the index is discovered, the system accepts torque or speed setting commands from the Expression window.

5.8.1 Torque Control Mode

In the demonstration software, the key variables used to configure the torque-control mode are summarized below:

- gGUIObj.CtrlType — selects the motor control mode
- gGUIObj.IqCmd — selects the torque output point of the motor
- drv.commutationMode — selects the motor commutation mode
- gGUIObj.CommAngleOffset —holds the calculated/applied encoder index angle offset
- gGUIObj.GraphInput — modifies the graphing variables fed to the data logger module

To run the motor in torque-control mode, do the following:

1. Ensure that PWM output from RM46x is disabled (that is, gGUIObj.EnableFlg = 0).
2. Ensure that motor control is disabled (that is, gGUIObj.CtrlType = 0).
3. Set gGUIObj.EnableFlg to 1 to enable PWM outputs to gate driver IC.
4. Ensure that gGUIObj.CommAngleOffset is set to the correct value after system calibration.
5. Ensure that gGUIObj.IqCmd is set to 0.0. (This requests zero torque from the motor).
6. Ensure that drv.commutationMode is set to ENCODER_COMMUTATION_MODE.
7. Set gGUIObj.CtrlType to 1 for torque-control mode. (The motor ramp-commutates to find the encoder index. The duration of the ramp-commutation depends on the position of the rotor.)
8. Set gGUIObj.IqCmd to 0.06. (This specifies the reference current command for the quadrature component [that is, the torque component] PI controller. The motor will start spinning.)
9. Adjust gGUIObj.IqCmd for the desired torque output set point.

NOTE: The motor speed is not controlled in this mode and a nonzero torque reference will continue to increase the speed of the motor. Load the motor using a brake or generator after closing the loop. For smaller motors, apply the brake manually.

10. Set gGUIObj.GraphInput to 2 to enable graphing of current control mode variables. (Verify the DC bus voltage [real world value, unit Volt] and PWM output [per unit] waveforms. Refer to Figure 12.)
11. Set gGUIObj.IqCmd to 0 to stop driving the motor. (If a fault condition is detected, gGUIObj.CtrlType and gGUIObj.IqCmd are set to 0. A fault signaled by the DRV8301 gate driver IC is indicated by LED6 on the DRV8301-EVM board. The variable gGUIObj.RstFault can be set to 1 to reset the fault indication.)

The Teknic motor in the kit has a maximum operating voltage range greater than 50 V and a back-EMF rating of 4.6 V/krpm. When using the 24-V brick power supply in the kit, the maximum speed of the motor is physically restricted. Operating the motor without any load in the torque-control mode may cause a fault condition.

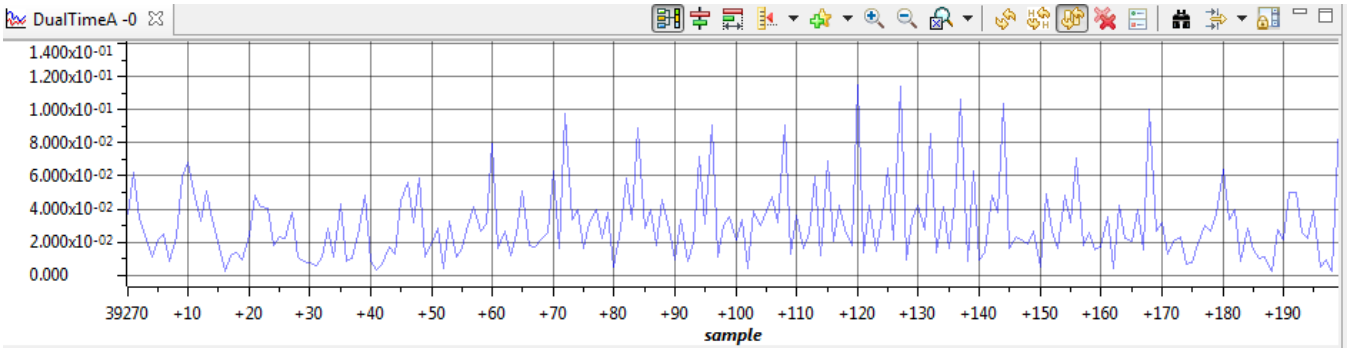


Figure 12. Motor Torque Output

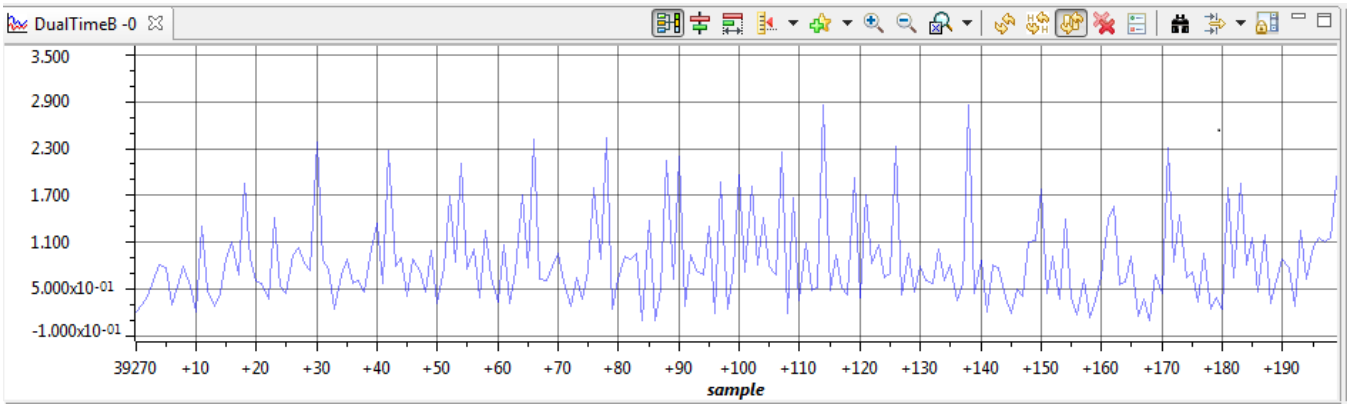


Figure 13. Total Motor Current

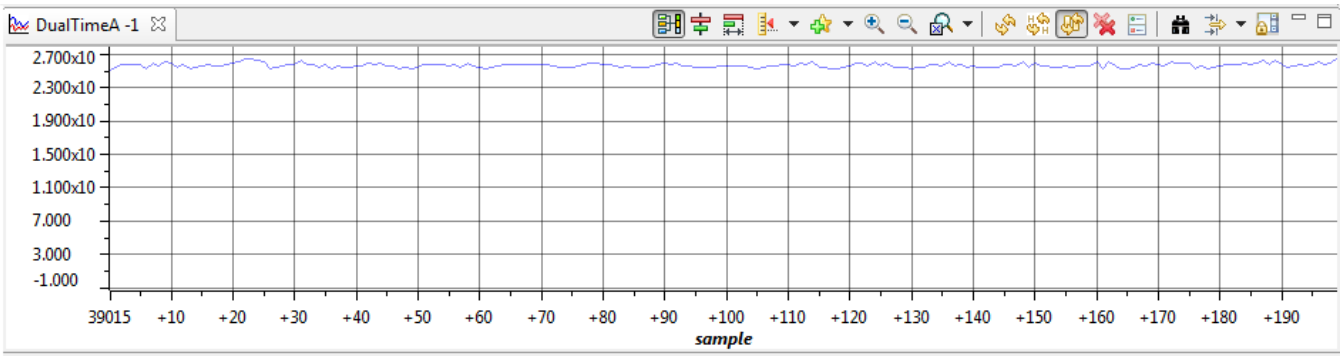


Figure 14. DC Bus Voltage

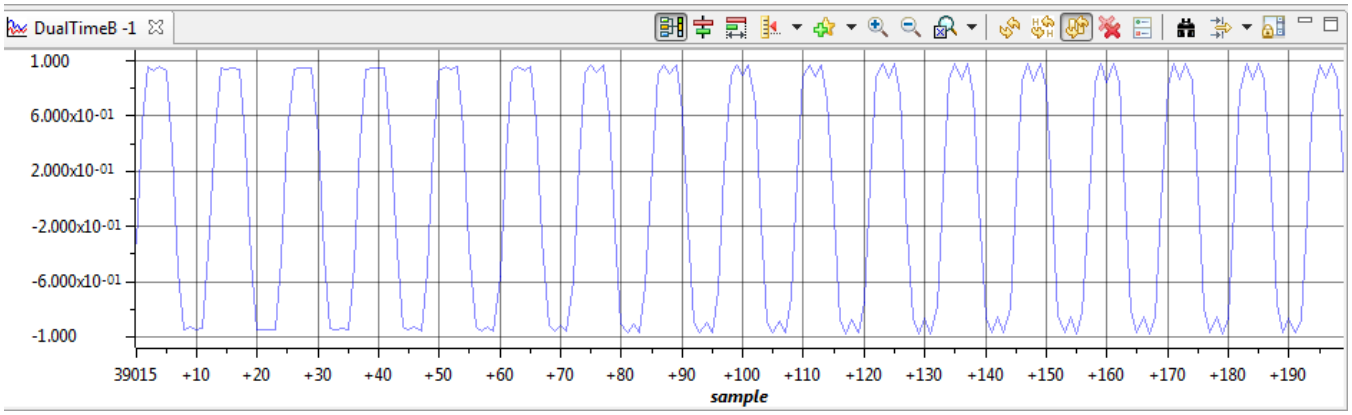


Figure 15. PWM1 Output

5.8.2 Speed Control Mode

In the demonstration software, the key variables used to configure the speed-control mode are summarized as follows:

- gGUIObj.CtrlType — selects the motor control mode
- gGUIObj.SpdCmd — sets the speed output point of the motor
- drv.commutationMode — chooses between commutation using feedback from an encoder or sensorless commutation
- gGUIObj.CommAngleOffset —provides information about the offset (in encoder ticks) between the encoder index position and the motor's zero pole-pair position
- gGUIObj.GraphInput — modifies graphing variables fed to data logger module
- drv.currentSpd — indicates the current speed that the motor
- gGUIObj.SwitchOverSpdFwd — is a speed threshold while the motor speed increases (If speed exceeds this threshold, the commutation mode switches from an encoder-based commutation to a sensorless commutation.)
- gGUIObj.SwitchOverSpdRev — is a speed threshold while the motor speed is decreasing (If speed drops below this threshold, the commutation mode switches from a sensorless commutation to an encoder-based commutation.)
- gGUIObj.SpeedEncoder — is the actual speed reported by the encoder inside the motor
- gGUIObj.SMO —is the motor speed estimated by the sliding mode observer algorithm running on the RM46x MCU
- gGUIObj.FaultEncoder — encoder fault status

To run the motor in speed-control mode, do the following:

1. Ensure that the program is running and that gGUIObj.EnableFlg is 0.
2. Verify that gGUIObj.CtrlType is 0, for disabled mode.
3. Set gGUIObj.EnableFlg to 1 to enable PWM outputs to gate driver IC.
4. Ensure that gGUIObj.CommAngleOffset is set to the correct value. (If going through start-up, complete the system calibration routine described in Section 5.7. Alternatively, set drv.calibrateAngle to a known value.)
5. Verify that gGUIObj.SpdCmd is set to 0. (This sets the output speed point for the motor.)
6. Verify that drv.commutationMode is set to ENCODER_COMMUTATION_MODE.
7. Set gGUIObj.CtrlType to 2 for enabling speed-control mode.
8. Set gGUIObj.SpdCmd to 0.5. (The motor starts spinning to try and reach the set speed point. Adjust gGUIObj.SpdCmd to the target speed. If the measured speed of the motor (drv.currentSpd) is greater than the forward speed threshold (gGUIObj.SwitchOverSpdFwd), the commutation mode switches to the sensorless commutation mode. If drv.currentSpd is less than the reverse speed threshold (gGUIObj.SwitchOverSpdRev), the commutation mode switches to the encoder-based commutation mode.)
9. Set gGUIObj.GraphInput to 1 to enable graphing of the speed-control mode variables.
10. Verify the motor speed feedback (pu) and Sliding-Mode-Observer-based electrical angle waveforms.
11. When the motor is running in the sensorless commutation mode, unplug the encoder feedback cable from the J4 connector on the DRV8301-EVM board.

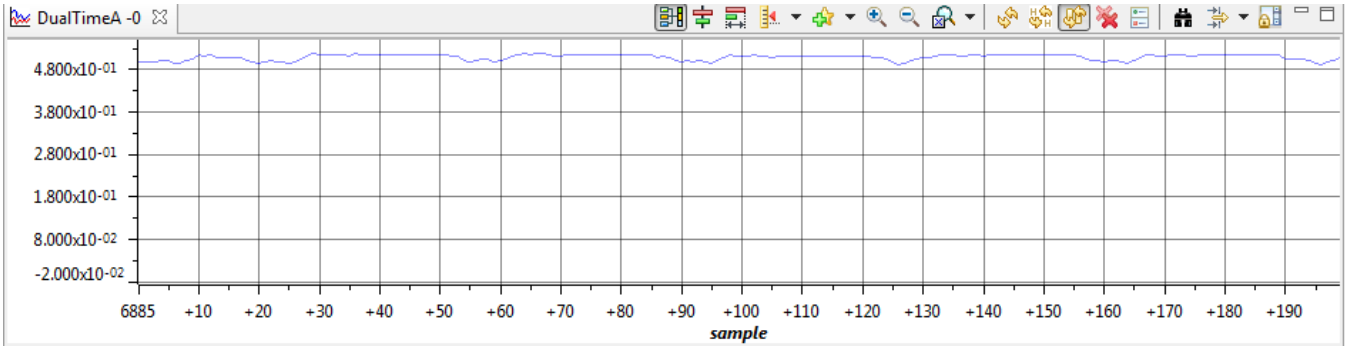


Figure 16. Motor Speed Feedback

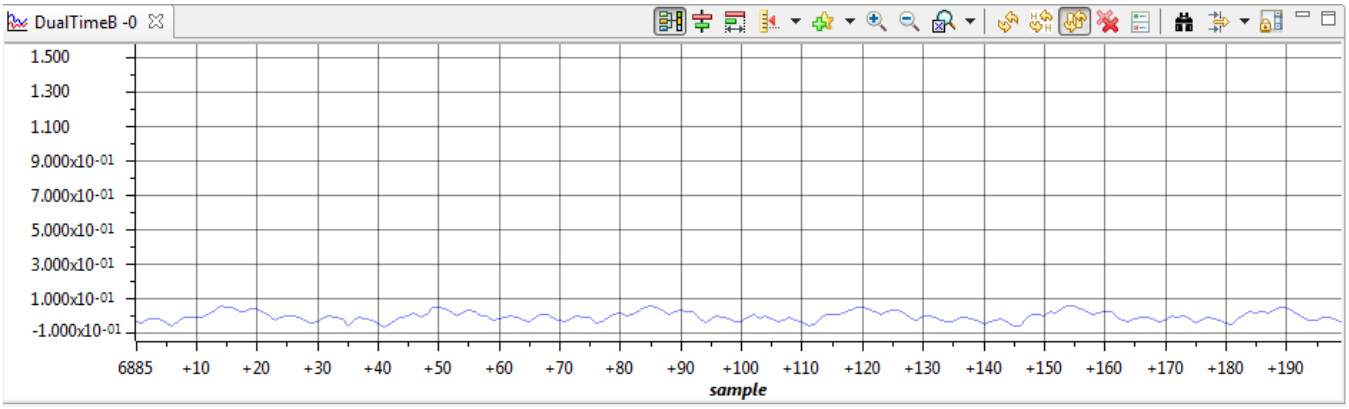


Figure 17. Phase A Current

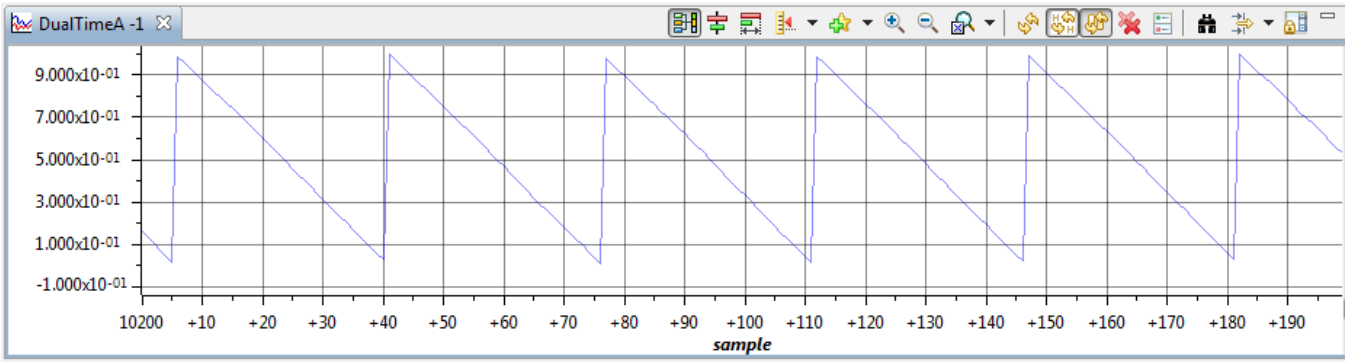


Figure 18. SMO-based Electrical Angle

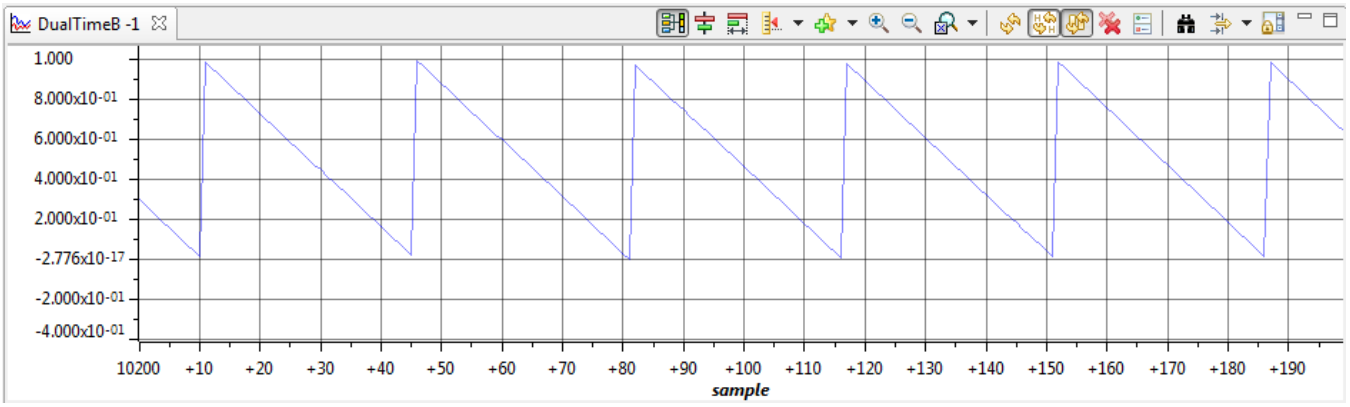


Figure 19. Encoder-based Electrical Angle

6 Test Setup

The system described in the preceding sections was used to collect the benchmark data being presented. For the RM46x MCU configuration, see Table 2.

Table 2. RM46L852 Test Software Configuration

Parameter	Value
Runtime library used	rtsv7R4_T_le_V3D16_eabi.lib
Floating-Point / Fixed-Point	Float
Control Algorithm	Speed + Torque, Encoder-based / Sensorless
CPU Frequency	GCLK = HCLK = 80 MHz
Peripheral clock domain frequency	VCLK = HCLK = 80 MHz
PWM Output Frequency	20 kHz
Program Flash Random Read Wait States	1
Debugger	Code Composer Studio, v6.1.0.00104
Compiler	TI CodeGen Tools for ARM CPUs, v5.2.3
CPU Instruction Set	ARM 32-bit Instruction Set
Hardware Kit	DRV8301-RM46-KIT

7 Test Data

Table 3 presents the results of the benchmark test of the RM46L852 MCU driving the PMSM using an encoder-based or sensorless control algorithm. The left-hand column presents the key software module used as part of the control algorithm. The right-hand column presents the average number of CPU cycles as well as the maximum number of CPU cycles used for that software module.

Table 3. CPU Cycles for Executing Software Modules

Software Module	Number of CPU Cycles (Average / Maximum)
RAMPGEN√	58 / 79
QEP	93 / 110
SPEED_FR	69 / 79
CLARKE	27 / 27
PARK	87 / 95
RAMP_CTL	47 / 100
PI(D) x3	46 / 46 (x3)
IPARK	31 / 31
SVGEN	71 / 71
PWM Update	144 / 144
VOLT_CALC	72 / 79
SMOPOS	213 / 229
SPEED_EST	65 / 75
DLOG(1)	56 / 110
Total	1152 / 1265
CPU Utilization(2) @ 80 MHz	28.8% / 31.6%

(1) Not included in the count of CPU cycles because DLOG can be removed from the loop.
(2) At 20-kHz ISR frequency

8 Design Files

The design files related to this TI Design are available at: http://www.ti.com/tool/TIDM-RM46xDRV8301KIT.

This database includes the design schematics, the bill of materials, the PCB design files, and the Gerber files for each board in this TI Design.

8.1 Software Files

To download the software files, see the design files at TIDM-RM46xDRV8301KIT.

9 References

- *RM46L852 16- and 32-BIT RISC Flash Microcontroller* (SPNS185)
- *RM46x 16/32-Bit RISC Flash Microcontroller Technical Reference Manual* (SPNU514)

10 About the Author

SUNIL OAK is a systems engineer at TI, where he develops reference design solutions using the Hercules family of microcontrollers. Sunil brings his extensive experience in high-speed digital and microcontroller system-level design expertise to this role. Sunil earned his Master of Science in Electrical Engineering (MSEE) from the University of Houston in Houston, TX. Sunil is a member of the Institute of Electrical and Electronics Engineers (IEEE) and the Society of Automotive Engineers.

IMPORTANT NOTICE FOR TI REFERENCE DESIGNS

Texas Instruments Incorporated ("TI") reference designs are solely intended to assist designers (“Buyers”) who are developing systems that incorporate TI semiconductor products (also referred to herein as “components”). Buyer understands and agrees that Buyer remains responsible for using its independent analysis, evaluation and judgment in designing Buyer’s systems and products.

TI reference designs have been created using standard laboratory conditions and engineering practices. **TI has not conducted any testing other than that specifically described in the published documentation for a particular reference design.** TI may make corrections, enhancements, improvements and other changes to its reference designs.

Buyers are authorized to use TI reference designs with the TI component(s) identified in each particular reference design and to modify the reference design in the development of their end products. HOWEVER, NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY THIRD PARTY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT, IS GRANTED HEREIN, including but not limited to 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.

TI REFERENCE DESIGNS ARE PROVIDED "AS IS". TI MAKES NO WARRANTIES OR REPRESENTATIONS WITH REGARD TO THE REFERENCE DESIGNS OR USE OF THE REFERENCE DESIGNS, EXPRESS, IMPLIED OR STATUTORY, INCLUDING ACCURACY OR COMPLETENESS. TI DISCLAIMS ANY WARRANTY OF TITLE AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, QUIET ENJOYMENT, QUIET POSSESSION, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS WITH REGARD TO TI REFERENCE DESIGNS OR USE THEREOF. TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY BUYERS AGAINST ANY THIRD PARTY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON A COMBINATION OF COMPONENTS PROVIDED IN A TI REFERENCE DESIGN. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, SPECIAL, INCIDENTAL, CONSEQUENTIAL OR INDIRECT DAMAGES, HOWEVER CAUSED, ON ANY THEORY OF LIABILITY AND WHETHER OR NOT TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, ARISING IN ANY WAY OUT OF TI REFERENCE DESIGNS OR BUYER’S USE OF TI REFERENCE DESIGNS.

TI reserves 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 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 for TI components 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.

Reproduction of significant portions of TI information in TI data books, data sheets or reference designs 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.

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 that anticipate dangerous failures, monitor failures and their consequences, lessen the likelihood of dangerous failures 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 Buyer’s 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 an agreement specifically governing such use.

Only those TI components that 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 that have ***not*** been so designated is solely at Buyer's risk, and 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.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265