Dual-Axis Current Control of Brushed DC Motors Application Report

Application Report

Literature Number: TIDA013
May 2014

Contents

Submit Documentation Feedback

List of Figures

List of Tables

Dual-Axis Current Control of Brushed DC Motors Application Report

ABSTRACT

This application note presents a solution to microstep a bi-polar stepper motor, using TMS320F2803x microcontrollers and the DRV84x2 Dual Full-Bridge PWM Motor Driver. The TMS320F2803x devices are part of the C2000 family of microcontrollers. These devices enable cost-effective design of intelligent motor controllers by reducing the number of system components and increasing the system efficiency. With these devices it is possible to achieve more precise digital control algorithms. The DRV84x2 devices are high-performance, integrated dual full-bridge motor drivers with an advanced protection system.

This application note covers the following topics:

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

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

1 Benefits of Using C2000 Controllers for Digital Motor Control (DMC)

Devices that belong to the C2000 family posses the 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 ADC, ePWM, QEP, eCAP and others). These peripherals have all the hooks needed to implement systems that meet safety requirements, like the trip zones for PWMs and comparators.

Along with these hooks, the C2000 ecosystem of software (libraries and application software) and hardware (application kits) help reduce the time and effort needed to develop a solution for digital motor control. The DMC Library provides configurable blocks that can be reused to implement new control strategies. The IQMath Library enables easy migration from floating-point algorithms to fixed-point algorithms, which accelerates the development cycle. Using devices from the C2000 family, developers can quickly and easily implement complex control algorithms for motor control.

Using the C2000 devices and the advanced control schemes provides the following system improvements:

- System cost reduction by using an efficient control in all speed ranges, and by applying the right dimensioning of power to device circuits.
- Decreases the number of look-up tables, which reduces the amount of required memory
- Improved performance by using real-time generation of smooth, near-optimal, reference profiles and move trajectories
- Allows generation of high-resolution PWMs by using the ePWM peripheral for controlling the power-switching inverters
- Provides a single-chip control system

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

- Control multi-variable and complex systems by using modern intelligent methods, such as neural networks and fuzzy logic
- Perform adaptive control. C2000 controllers have the speed capabilities to concurrently monitor and control the system. A dynamic control algorithm adapts itself in real-time to variations in system behavior.
- Provide diagnostic monitoring with spectrum analysis. By observing the frequency spectrum of mechanical vibrations, failure modes can be predicted in early stages.
- Produce sharp-cut-off notch filters that eliminate narrow-band mechanical resonance. Notch filters remove energy that would otherwise excite resonant modes and could make the system unstable.

2 Texas Instruments Documentation and the DMC Library

The Digital Motor Control (DMC) library is composed of functions, represented as blocks. These blocks are categorized as one of three categories: Transforms & Estimators (Clarke, Park, Sliding Mode Observer, Phase Voltage Calculation, 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 that is separately documented with source code, use, and technical theory. Check the folder directories below for the source codes and explanations of the macro blocks:

- C:\TI\controlSUITE\libs\app_libs\motor_control\math_blocks\fixed
- C:\TI\controlSUITE\libs\app_libs\motor_control\drivers\f2803x

The library supports the five motor types: AC induction (ACI), Brushless DC (BLDC), Permanent Magnet Synchronous (PMSM), Brushed DC, and the Stepper Motor and comprises both peripheral-dependent (software drivers) and target-dependent modules. The DMC Library components have been used by Texas Instruments (TI) to provide system examples.

At initialization, all DMC Library variables are defined and interconnected, and the macro functions are called in order at run time. Each system is built using an incremental build approach, which allows a few sections of the code to be built at a time. By using incremental builds, the developer can verify each section of their application one step at a time. This verification is critical in real-time control applications, where 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 (refer to SPRAAK2 for more details). The macros are defined in the header files, and the macro definitions can be changed in the respective header file, if needed.

In the macro definitions, there should be a backslash (\) at the end of each line, as shown in the code snippet below, to show that the code continues on the next line. Ensure the backslash is the last character in the line. Any character after the backslash, including "invisible" characters like a space, causes a compilation error. In terms of code development, the macros are almost identical to C functions, and the user can easily convert the macro definition to a C function.

The following code example is 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);
```

For more information, review the C2000 Motor Control Primer (link: SPRUGI6)

3 System Overview

This document describes the “C” real-time control framework that is used to demonstrate current-controlled microstepping of bi-polar stepper motors. The “C” framework is designed to run on TMS320C2803x-based controllers in Code Composer Studio.

The framework uses the modules listed in Table 1:

Table 1. Framework Modules[1]

Macro Names	Explanation
PID	PID Regulators
RC	Ramp Controller (slew rate limiter)
RG	Rampe/Sawtooth Generator
SINCOSTBL	Sin/Cos Lookup Table
PWM/PWMDAC	PWM and PWMDAC Drivers

[1] For details and theoretical background of each macro, see C:\ti\controlSUITE\libs\app_libs\motor_control

This system demonstrates current-controlled microstepping of a bi-polar stepper motor. The stepper motor is driven by a H-bridge, which is provided by the DRV8412 Dual Full-Bridge PWM Motor Driver IC.

The Piccolo F28035 MCU controlCARD generates four pulse-width modulation (PWM) signals, two for each motor phase. Two phase currents of each motor phase are measured from the H-bridge and sent to the F28035 device through four analog-to-digital converters (ADCs). In addition, the DC-bus voltage is measured and sent to the F28035 through an ADC.

The Stepper project has the properties shown in Table 2, Table 3, and Table 4

Table 2. C Framework

System Name	Program Memory Use 2803x	Data Memory Use 2803x [1]
Stepper	2847 words [2]	1060 words

[1] Excluding the stack size

[2] Excluding "IQmath" Look-Up Tables

Table 3. CPU Use of Microstepping Control

Name of Modules [1]	Number of Cycles
Ramp Controller	24
Ramp Generator	60
Sin/Cos Table	33
2 x Pid	96
2 x Pwm Drv	150
A/D Feedback scaling	54
Contxt Save etc.	53
Pwm Dac (optional)	
DataLog (optional)	
Total Number of Cycles	**470** [2]
CPU Use @ 60 Mhz	7.8% [3]
CPU Use @ 40 Mhz	11.8% [3]

[1] The modules are defined in the header files as "macros"

[2] 556 including the optional modules

[3] At 10 kHz ISR freq.

Table 4. System Features

Development /Emulation	Code Composer Studio v4.0 (or above) with Real Time debugging
Target Controller	TMS320F2803x
PWM Frequency	10kHz PWM (Default), 60kHz PWMDAC
PWM Mode	Symmetrical with a programmable dead band
Interrupts	EPWM1 Time Base CNT_Zero – Implements 10 kHz ISR execution rate
Peripherals Used	PWM 1 / 2 for motor control
	PWM 6A, 6B for DAC Outputs
	ADC A4 for DC Bus Voltage Sensing, A0, A1, A2 & A3 for Phase Current Sensing

Figure 1 depicts the overall system that accomplishes current-controlled microstepping.

Thestepper motor isdriven by the conventional H-bridge configuration. The F28035 device generates the four pulse-width modulation (PWM) signals needed to drive the DRV8412 Dual Full-Bridge PWM Motor Driver. Two input currents of each motor are measured from the H-bridge and 4 ADCs send these currents to the F28035 device .

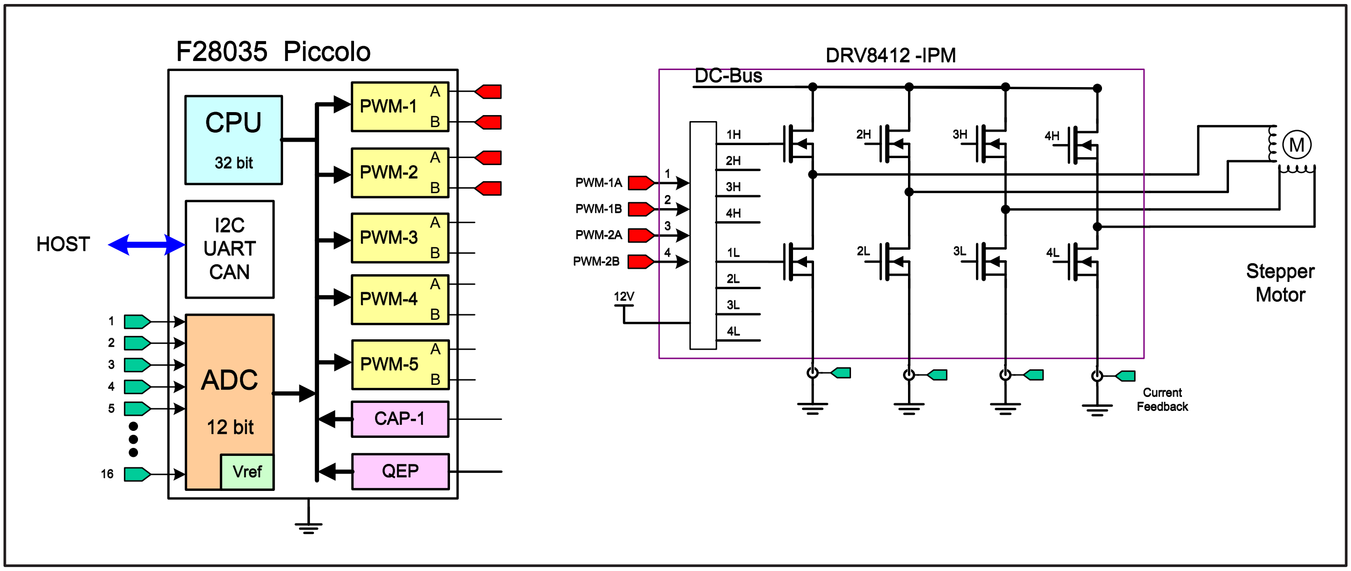


Figure 1. Stepper Motor Drive

Figure 2 shows the software flow.

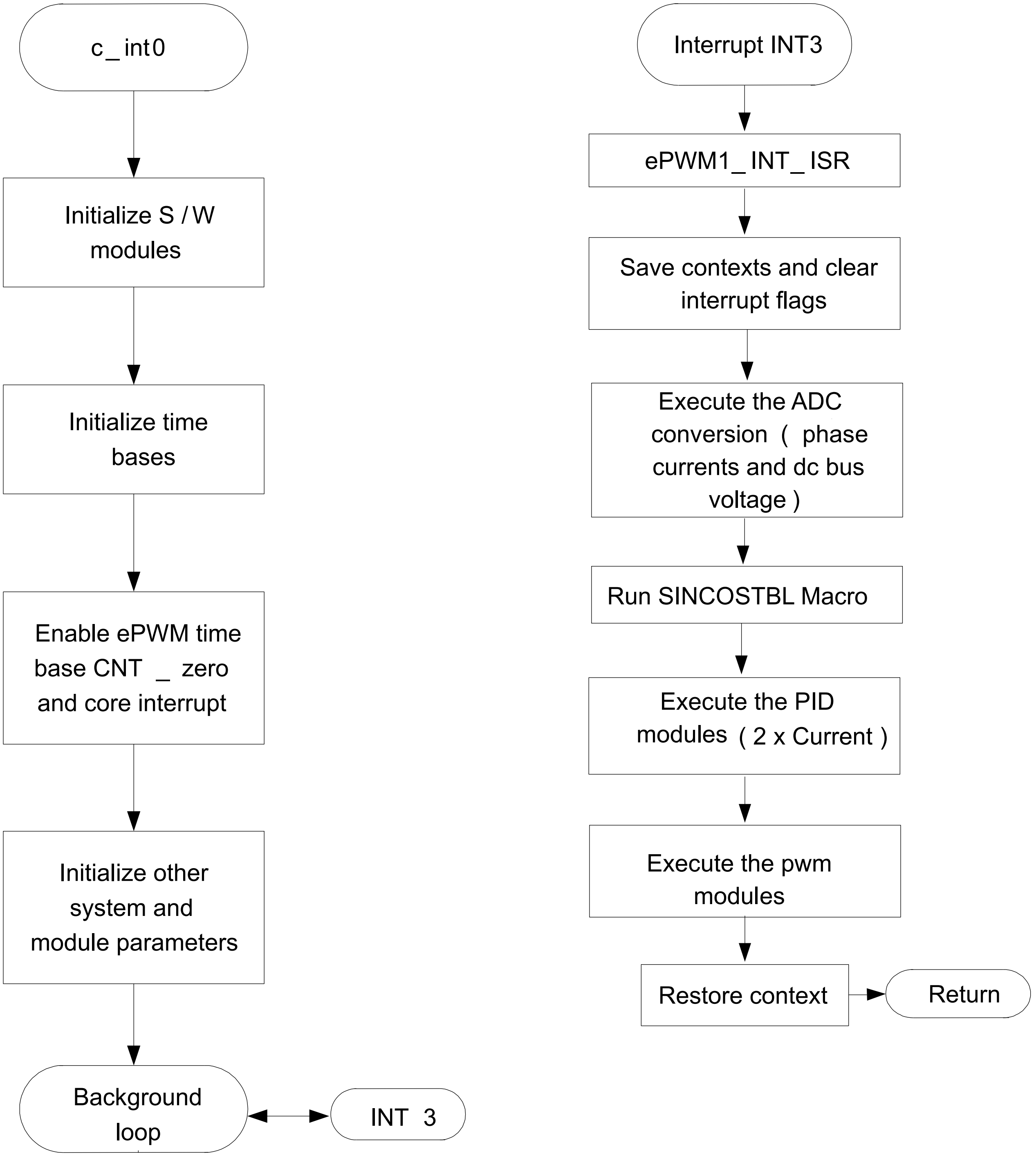

c_int0
Initialize S / W modules
Initialize time bases
Enable ePWM time base CNT _ zero and core interrupt
Initialize other system and module parameters
Background loop
INT 3
Interrupt INT3
ePWM1_INT_ISR
Save contexts and clear interrupt flags
Execute the ADC conversion (phase currents and dc bus voltage)
Run SINCOSTBL Macro
Execute the PID modules (2 x Current)
Execute the pwm modules
Restore context
Return

Figure 2. System Software Flowchart

4 Hardware Configuration (DRV8412-EVM)

For an overview of the kit hardware and the steps required to set up the kit, refer to the *DRV8412-EVM How to Run Guide* and the *DRV8412-EVM Hardware Reference Guide*, located in the following directory, :

```
C:\TI\controlSUITE\development_kits\DRV8412-EVM\~Docs
```

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

4.1 Hardware Setup Instructions

1. Unpack the DIMM-style controlCARD and verify that the DIP switch settings match those shown in Figure 3

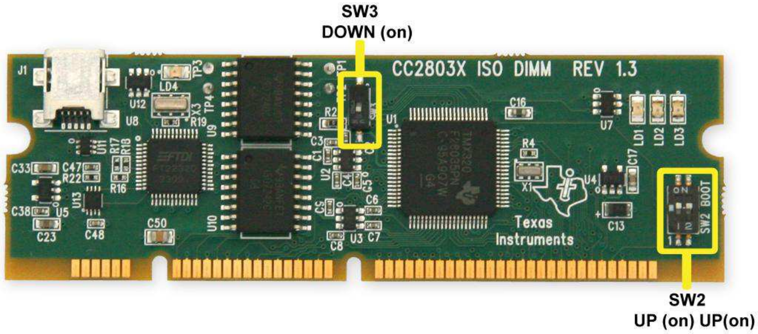


Figure 3. controlCARD DIP Switch Settings

2. Place the controlCARD in the connector slot of J1. Push vertically down, 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 thumbs
3. Ensure the DRV8412 mode jumpers and the +12V source jumper are set according to Figure 4.
4. Connect a USB cable to connector J1 on the controlCARD. This connection will enable isolated JTAG emulation to the C2000 device. LD4 should turn on.

   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 device.
5. Connect a 24-V power supply to the PVDD and GND terminals of the DRV8412-EVM. Now, LED2 and LED3 should turn on. Notice the control card LED would also turn on, indicating the control card is receiving power from the board.
6. After completing the first incremental build step, motor 1 should be connected to the OUTA and OUTB terminals and motor 2 should be connected to the OUTC and OUTD terminals

For reference, Figure 4 shows the jumper and connectors that need to be made for this lab. The motor connections shown are for an Anaheim Automation 23Y106S-LW8 stepper motor wired in a parallel configuration. Note that there are two motor wires per terminal.

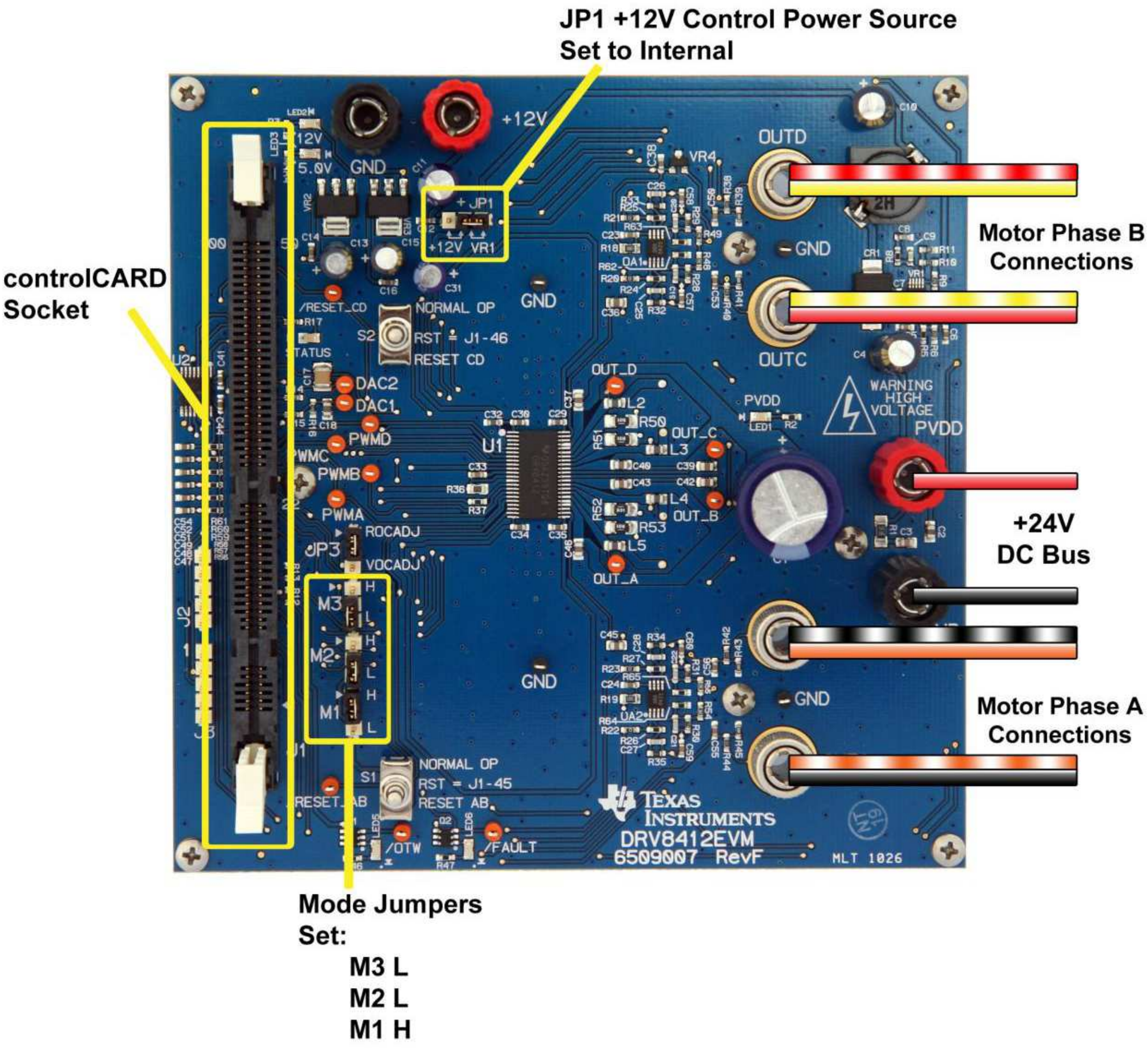


Figure 4. EVM Connections

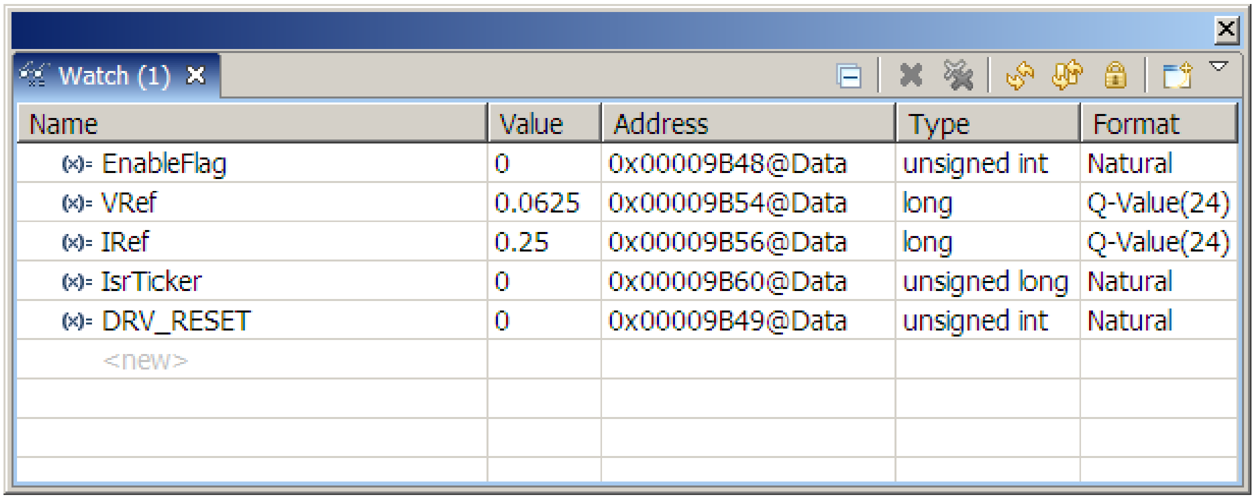
4.2 Software Setup Instructions to Run Stepper Motor Project

Please refer to the *Generic Steps for Software Setup for DRV8412-EVM Projects* section in the *DRV8412-EVM How To Run Guide*.

`C:\TI\controlSUITE\development_kits\DRV8412-EVM\~Docs`

This section instructs how to install Code Composer Studio (CCS) and how to set up CCS to run with this project.

1. Select the Stepper as the active project.
2. Verify that the build level is set to 1
3. Right-click on the project name and select “Rebuild Project”.
4. Once the build completes, launch a debug session to load the code into the controller.
5. Open a watch window and add the critical variables as shown in Figure 5and select the appropriate Q format for them.



Name	Value	Address	Type	Format
(x)= EnableFlag	0	0x00009B48@Data	unsigned int	Natural
(x)= VRef	0.0625	0x00009B54@Data	long	Q-Value(24)
(x)= IRef	0.25	0x00009B56@Data	long	Q-Value(24)
(x)= IsrTicker	0	0x00009B60@Data	unsigned long	Natural
(x)= DRV_RESET	0	0x00009B49@Data	unsigned int	Natural
<new>				

Figure 5. Watch Window Variables

6. Setup time graph windows by importing *graph12.graphProp* and *graph34.graphProp* from the following location `C:\TI\ControlSUITE\development_kits\DRV8412-EVM\Stepper\`.
7. Click on the Continuous Refresh button in the top-left corner of the graph tab to enable periodic capture of the data from the microcontroller.

5 Incremental System Build for Stepper Project

For the final system to be confidently operated, we gradually build the system over three phases. The three phases (builds) of the incremental system build are designed to test and verify the correct operation of the major software modules used in the system. Table 5 summarizes the modules tested and used in each incremental system build.

Table 5. Modules Used in the Incremental System Build

Software Module	Phase 1	Phase 2	Phase 3
PWMDAC_MACRO		[1][1]	[1]
RC_MACRO	[1]	[1]	[1]
RG_MACRO	[1]	[1]	[1]
SINCOSTBL_MACRO		[2][1]	[1]
PWM_MACRO	[2]	[1]	[1]
A/D Conversion		[2]	[1]
PID_MACRO			[2]

[1] This module is used in this phase

[2] this module is tested in this phase.

5.1 Level 1 Incremental Build

In this build, keep the motor disconnected. Assuming the load and build steps described in the *DRV8412-EVM How To Run Guide* are completed successfully, this section describes the steps for a “minimum” system check-out, which confirms operation of the system interrupt, the peripheral-dependent PWM_MACRO (PWM initializations and update), and the SINCOSTBL_MACRO modules.

1. Open the *Stepper-Settings.h* file
2. Select the level 1 incremental build option by setting the BUILDLEVEL to LEVEL1 (#define BUILDLEVEL LEVEL1).
3. Right-click on the project name and click Rebuild Project.
4. Once the build is complete, click on debug button
5. Reset the CPU and restart
6. Enable real-time mode
7. Run.
8. Set “EnableFlag” to 1 in the watch window. The variable named “IsrTicker” will now continue to increase, 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.

- **VRef(Q24):** for changing the motor voltage in per-unit.
- **SpeedRef:** for changing the motor electrical speed in per unit
- **DRV_RESET:** for holding the DRV8412 chip in reset.

5.1.1 Level 1A (PWM_MACRO Test)

In this level we will test the sin/cos table module (SINCOSTBL_MACRO). This module takes a discrete angle as input and outputs the sin and cos of that angle from a look-up table. The range of the angle input is dependant on the desired number of microsteps per step. This can be configured in *Stepper-Settings.h* and can range from whole stepping to 128 microsteps/step.

The SpeedRef value is specified to the RG_MACRO module through RC_MACRO. The output of the RG_MACRO is scaled and used as the Angle input to the SINCOSTBL_MACRO. The outputs of the SINCOSTBL_MACRO are then scaled by VRef to obtain the desired amplitude. The outputs of the RC_MACRO, RG_MACRO and SINCOSTBL_MACRO can be monitored using the graph windows in CCS. Figure 6 shows the resulting graph windows when SpeedRef is 0.25pu and VRef is 0.0625pu.

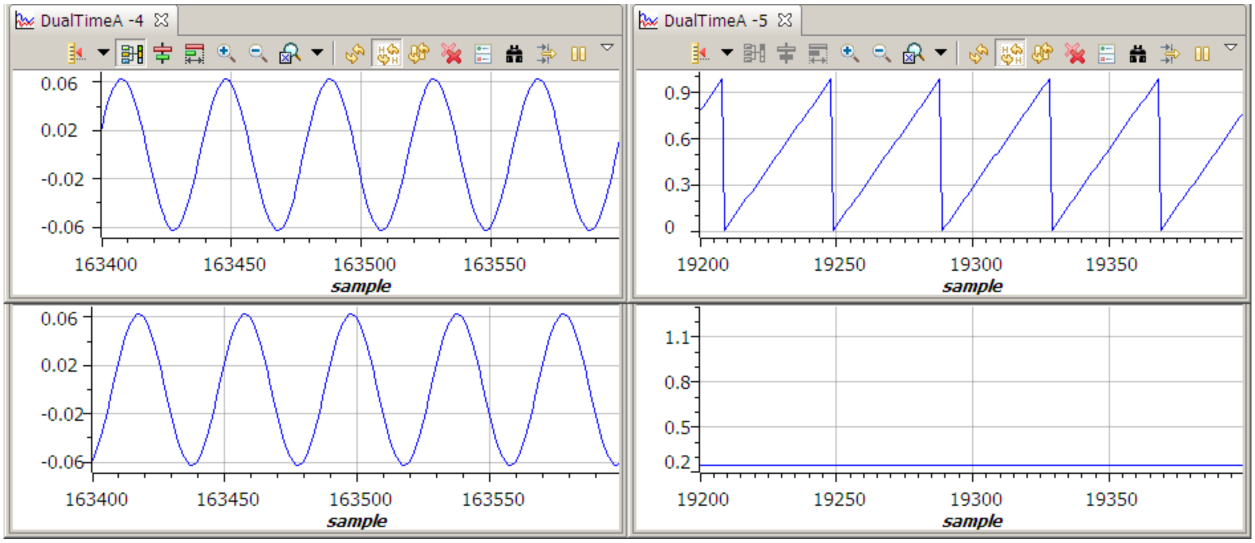


Figure 6. CCS Graph Windows for BuildLevel = 1

5.1.2 Level 1B: Testing the PWM_MACRO Macro

In this level we will test the PWM Generator Macro (PWM_MACRO). The outputs of the SINCOSTBL_MACRO are scaled by VRef and then specified to the PWM_MACRO. The duty cycle of the PWM outputs should vary sinusoidally at the frequency specified by SpeedRef. Check the PWM test points on the board to observe PWM pulses (PWMA and PWMB for Phase A and PWMC and PWMD for Phase B) and make sure that the PWM module is running properly.

When the duty cycle is positive PWMA (PWMC for Phase B) should show a 10KHz square wave with a duty cycle corresponding to the cosine (or sine for phase B) and PWMB (PWMD for Phase B) should be held low. When the duty cycle is negative PWMB (PWMD for Phase B) will be switching while PWMA (PWMC for Phase B) will be forced low.

5.1.3 Level 1C: Testing the PWMDAC Macro

PWM DACs are useful tools to monitor internal signal values in real time. PWM DACs on the DRV8412-EVM board use an external, low-pass filter to generate the waveforms (Test point is labeled DAC2). A simple, 1st–order, low-pass filter RC circuit filters out the high-frequency components.

Select the value of R and C (or the time constant, t) according to the value of the cut-off frequency (f_c). For this type of filter; the relation is as follows:

$$t=RC=\frac{1}{2\pi f_c} \tag{1}$$

For example, if R=470Ω and C=0.47uF, the result is f_c = 720.5 Hz. This cut-off frequency has to be below the PWM frequency. Using the formula above, one can customize the low-pass filters used for a signal being monitored. Refer to application note SPRAA88 for more details.

The PWM DACs should represent an IQ range of ±1.0 with an analog range of 0-3.3V.

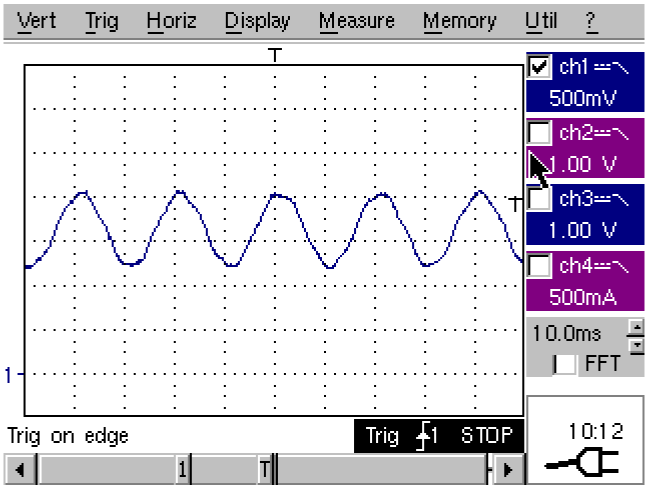


Figure 7. PWM DAC2 Output for SpeedRef = 0.25 and VRef = 0.25

5.1.4 Level 1D: Testing the DRV8412 Device

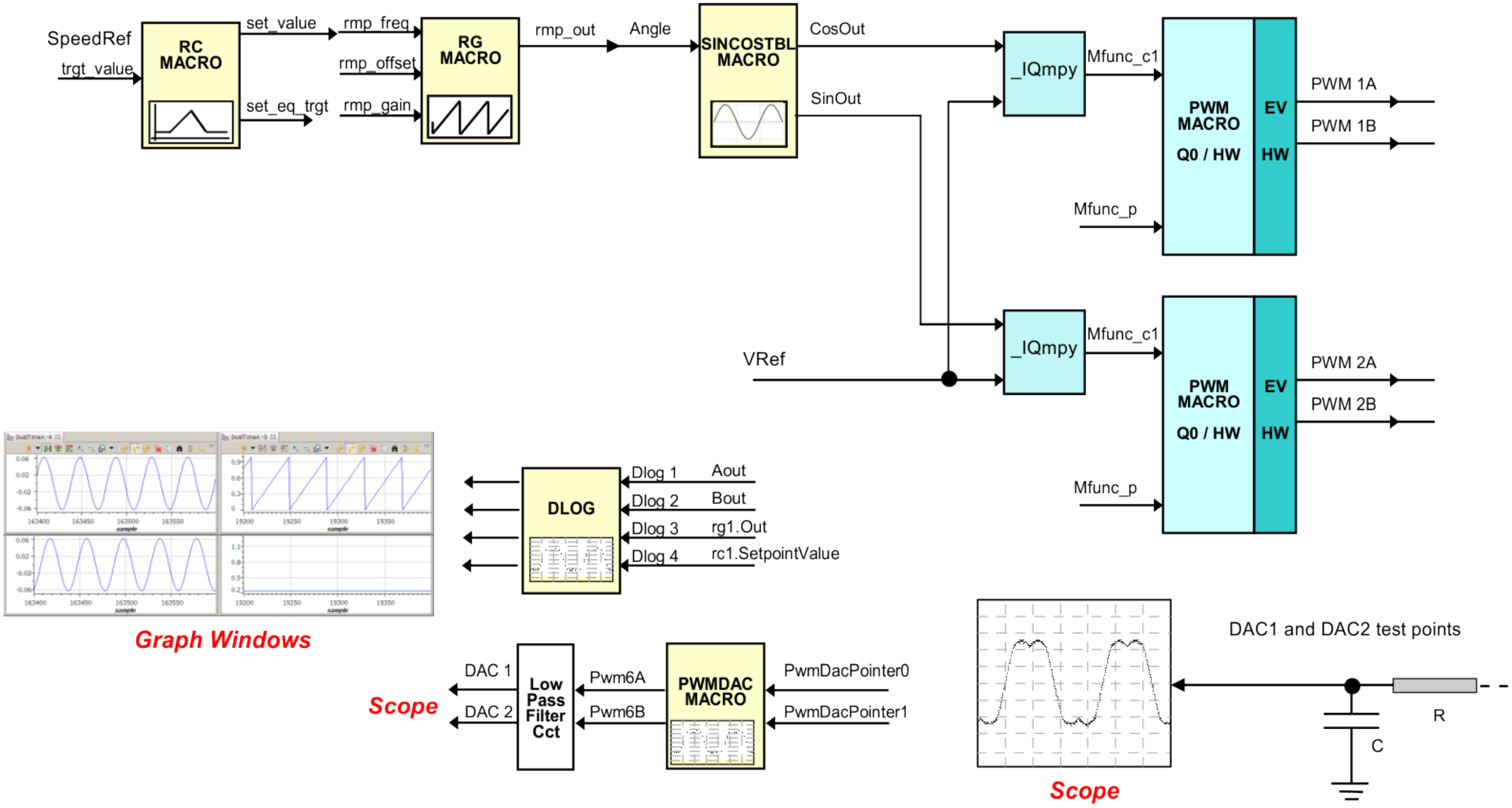
After verifying PWM_MACRO module in level 1a, the Dual Full-Bridge hardware in the DRV8412 device are tested by looking at the low-pass filter outputs..

1. Set the variable DRV_RESET to 0 through the watch window
2. Check the ADC_Vhb1 and ADC_Vhb2 schematic nodes with an oscilloscope.
3. Observe that the phase-voltage dividers and the waveform-monitoring filters (R42 to 45 & C55 to 56) enable the generation of the waveform and ensure that the DRV8412 device is working appropriately

CAUTION

After verifying the DRV8412 device is working, set the variable DRV_RESET to 1, take the controller out of real time mode (disable), and reset the processor (see *DRV8412-EVM How To Run Guide* for details).

Note, repeat this step after each test for safety purposes. Also, note that improper shutdown might halt the PWMs at certain states where high currents can be drawn, hence caution needs to be taken when performing a shutdown.



(1) Level 1 verifies the target independent modules, duty cycles andPWM update. The motor is disconnected at this level.

Figure 8. Level 1 – Incremental System Build Block Diagram

5.2 Level 2 Incremental Build

Assuming the Level 1 incremental build is completed successfully, this phase verifies the analog-to-digital conversion. For this build, connect the motor to the board. In the software, the key variables to be adjusted are summarized below.

- **VRef (Q24)**: for changing the motor voltage in per-unit.
- **SpeedRef (Q24):** for changing the motor electrical speed in per-unit

How to adjust the key variables:

1. Open *Stepper-Settings.h*
2. Select the level 2 incremental build option by setting the BUILDLEVEL to LEVEL2 (#define BUILDLEVEL LEVEL2)
3. Save the file.
4. Right-click on the project name and click Rebuild Project.
5. Once the build is complete click on the debug button.
6. Reset the CPU and restart
7. Enable real-time mode and run.
8. Set “EnableFlag” to 1 in the watch window. The variable named “IsrTicker” will be incrementally increased as seen in watch windows to confirm the interrupt working properly.

5.2.1 Testing the A/D Conversion

In this step, the A/D conversion will be tested.

1. Using the Watch Window, set VRef1 to 0.2 pu. The motor should start turning slowly.
2. Enable Continuous Refresh for the graph window and a graph similar to Figure 9 should appear. The top two graphs show the cos and sin voltage waveforms that are being generated. The bottom two graphs show the corresponding ADC sampled current waveforms.

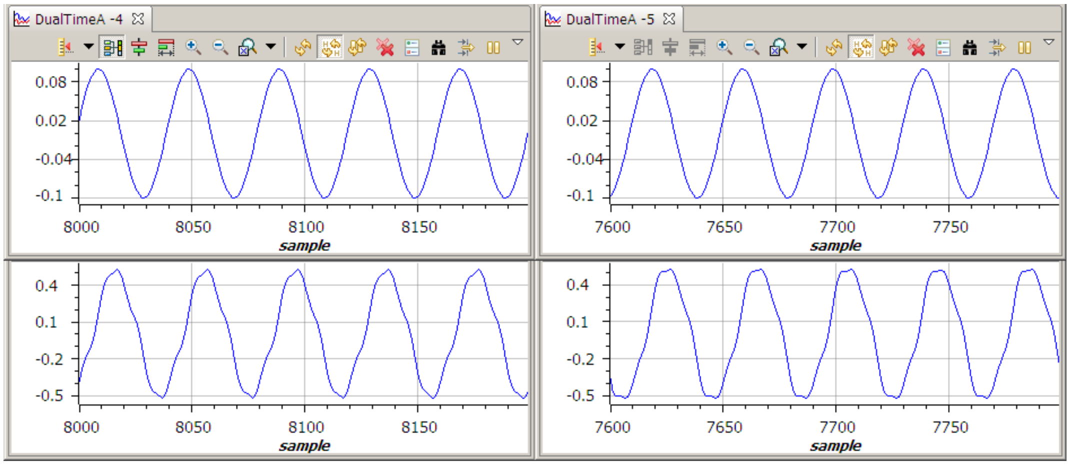
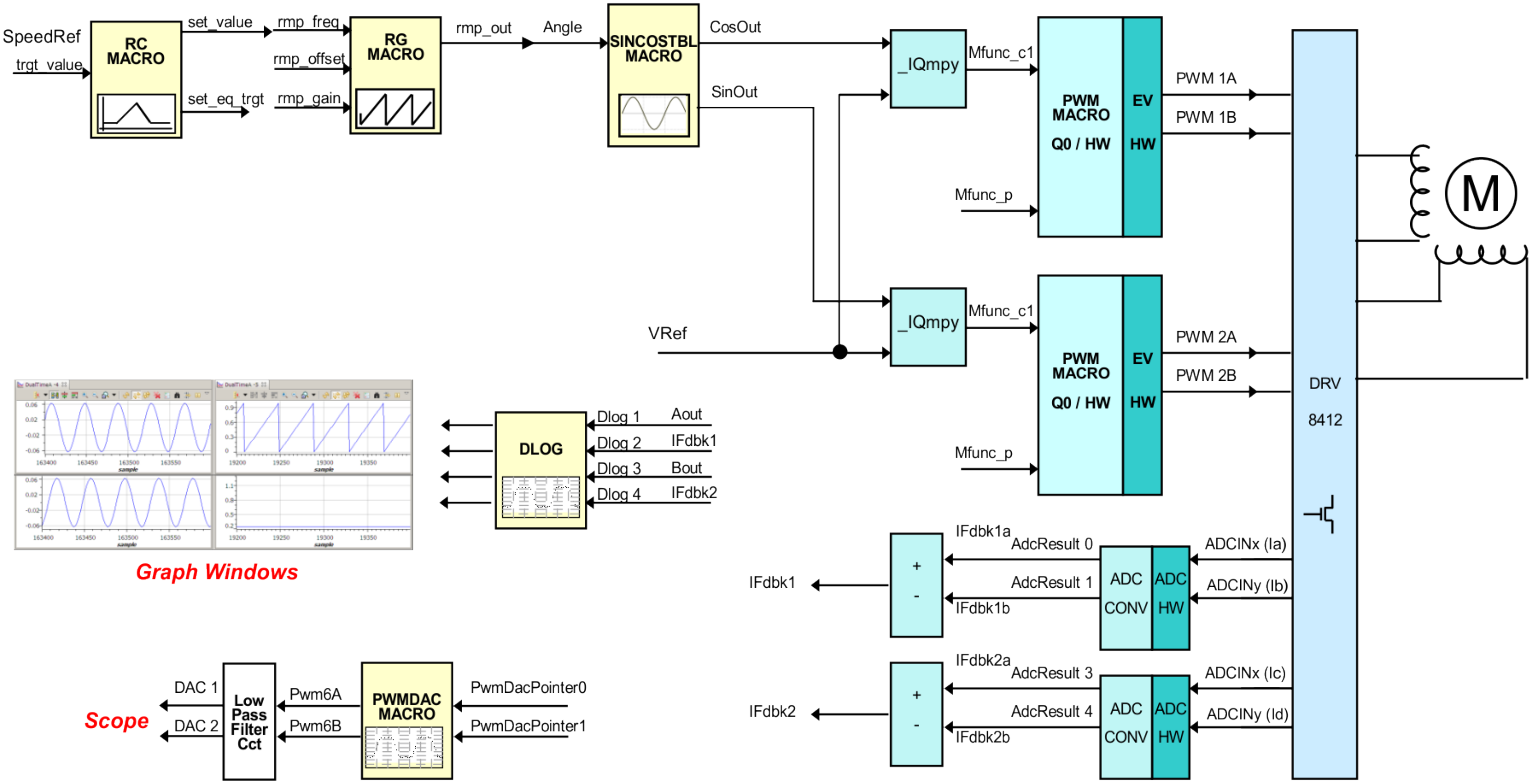


Figure 9. Build Level 2 Graph Showing Generated Sinusoidal Voltages And Current Feedback

3. Note that the open loop experiments are meant to test the ADCs, DRV8412, software modules etc, so running the motor under load or at various operating points is not recommended.
4. Bring the system to a safe stop as described at the end of build 1 by setting DRV_RESET to 1, taking the controller out of real-time mode and reset.



(1) Level 2 verifies the analog-to-digital conversion

Figure 10. Level 2 – Incremental System Build Block Diagram

5.2.2 Level 3 Incremental Build

Assuming the previous build is completed successfully, this build verifies the current regulation performed by the PID_REG3 module. To confirm the operation of the current regulation, the gains of these two PID controllers are necessarily tuned for proper operation.

1. Open *Stepper-Settings.h*
2. Select the level 3 incremental build option by setting the BUILDLEVEL to LEVEL3 (#define BUILDLEVEL LEVEL3).
3. Right-Click on the project name and click Rebuild Project.
4. Once the build is complete click on debug button
5. Reset the CPU and restart
6. Enable real-time mode and run.
7. Set “EnableFlag” to 1 in the watch window. The variable named “IsrTicker” will be incrementally increased as seen in watch windows to confirm the interrupt working properly.

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

- **IRef1(Q24):** for changing the motor current in per-unit.
- **SpeedRef (Q24:** for changing the motor electrical speed in per-unit

In this build, two quadrature, sinusoidal motor currents are dynamically regulated by using PID_REG3 module.

The steps are explained as follows:

1. Launch a debug session
2. Enable real-time mode and run the project.
3. Set IRef to 0.25 pu (or another suitable value if the base current is different).
4. Set SpeedRef to 0.25 pu
5. Check the graph windows with the continuous refresh feature. Figure 11 shows the graphs for this build level. The top two graphs show the commanded cos and sin current waveforms. The bottom two graphs show the corresponding current feedback waveforms. The feedback currents should track the commanded currents. If not, adjust its PID gains properly.

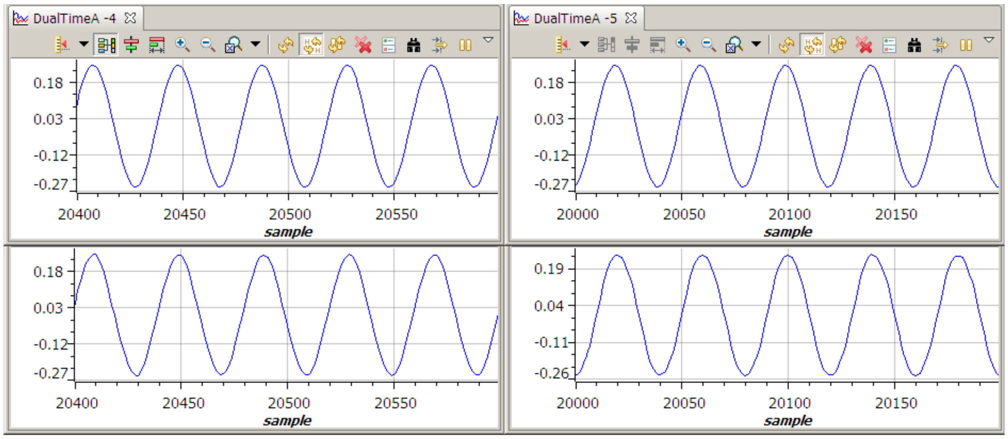


Figure 11. Build Level 3 Graphs Of Commanded And Regulated Motor Currents

- To confirm proper function of the PID module, try different values of IRef.
- For the PID controller, the proportional, integral, derivative, and integral correction gains may be re-tuned to have the satisfied response.
- Bring the system to a safe stop, as described at the end of build 1, by setting DRV_RESET to 1, taking the controller out of real-time mode, and reset.

5.2.2.1 Level 3 Additional Testing

Additional testing may be performed by modifying the value of MICROSTEPS in *Stepper-Settings.h*. Rebuild the project for this change to take effect. Currently whole stepping through 128 microsteps/step are supported. The lower the number of microsteps, the more discrete the sine wave will appear, which gives it a stair-step appearance. A higher number of microsteps will provide more continuous-looking sinusoidal currents. Figure 12 and Figure 13 illustrate the effect of changing the microstep resolution.

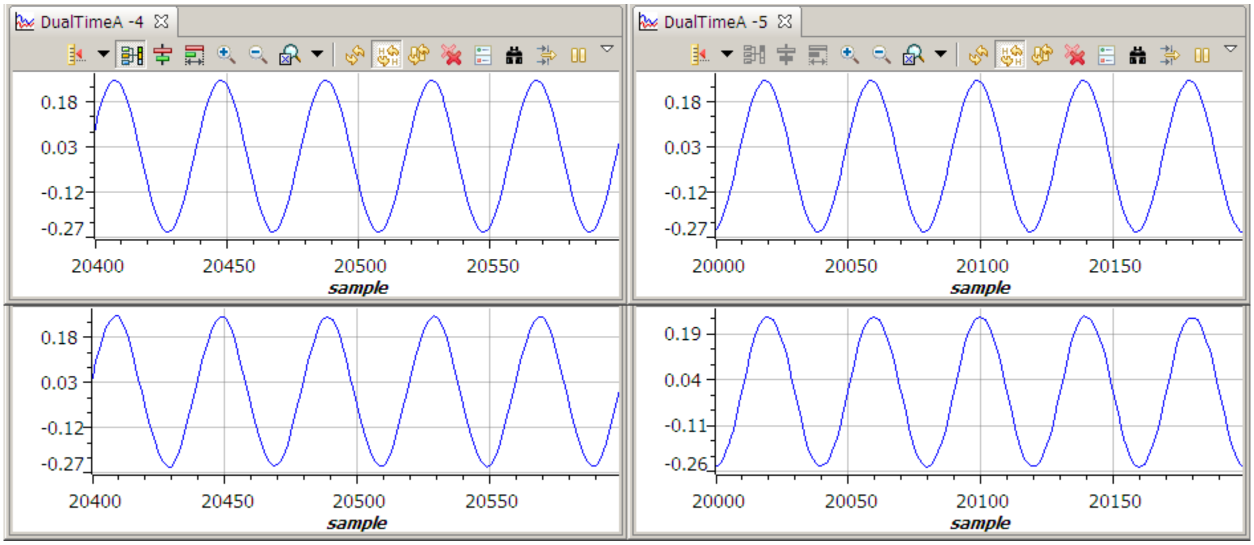


Figure 12. Command And Feedback Currents With 32 Microsteps/Steps

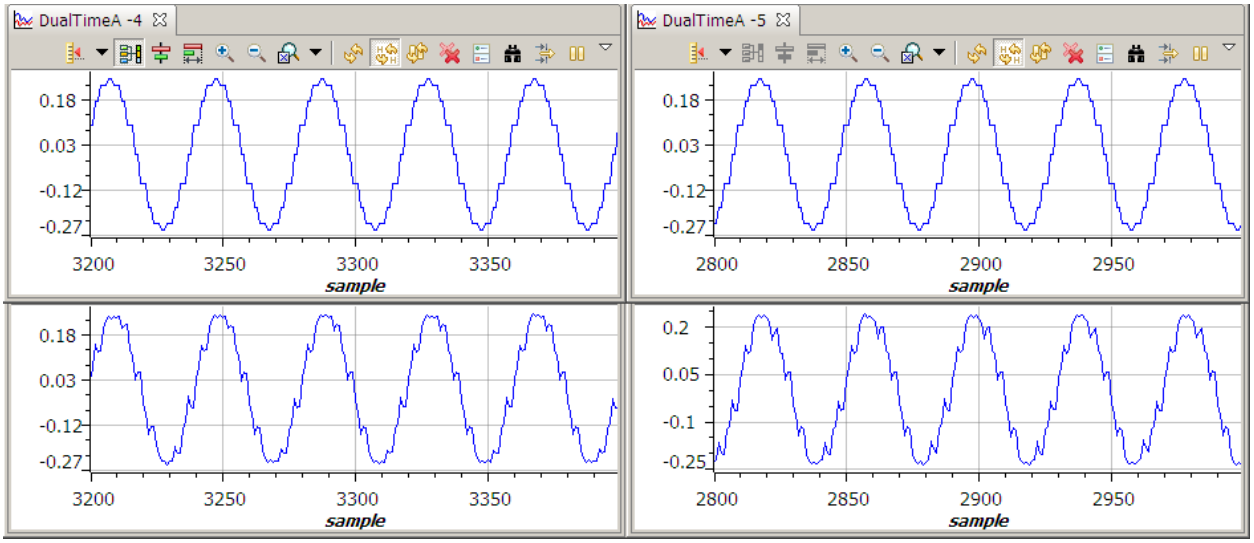
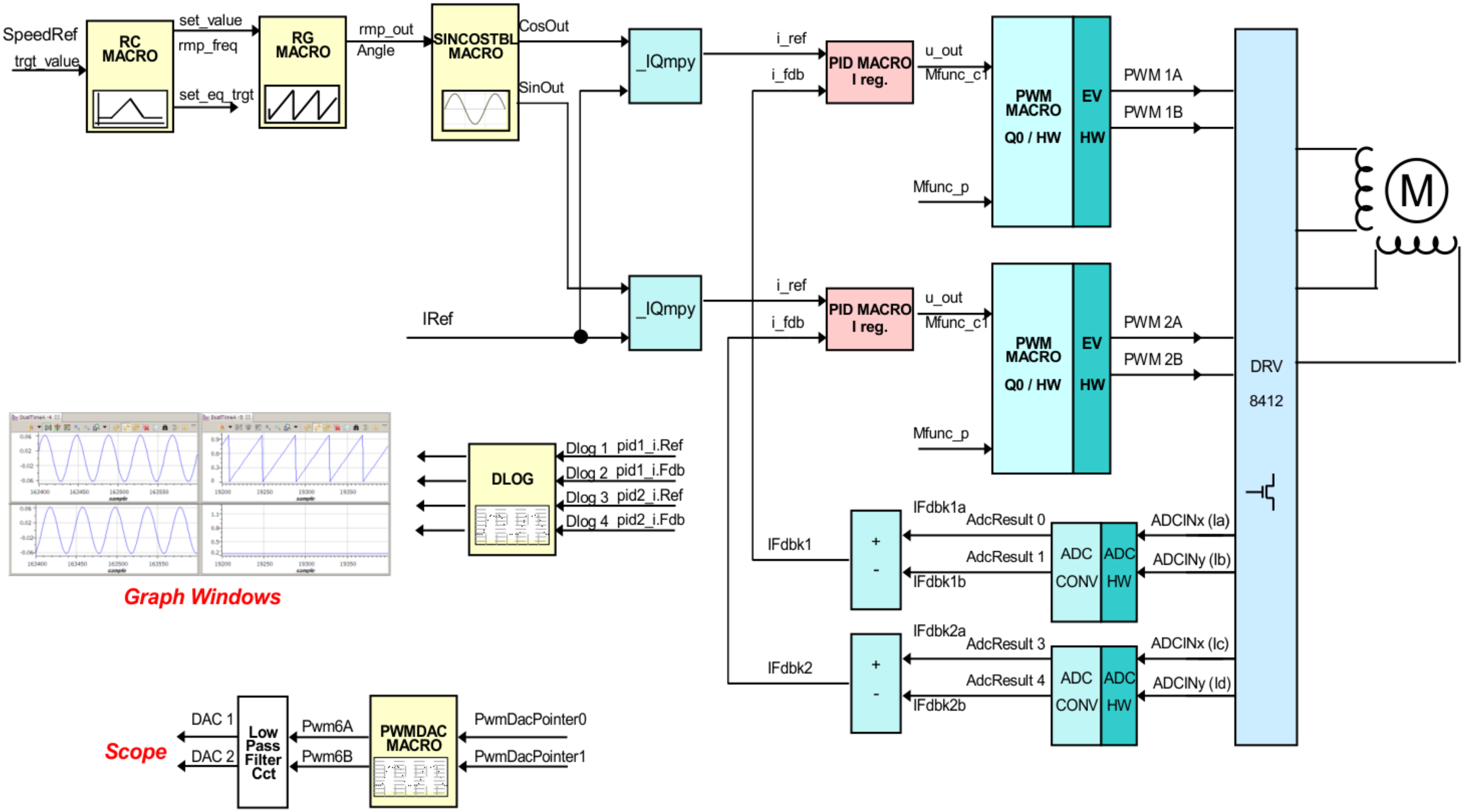


Figure 13. Command and Feedback Currents with Quarter Stepping



(1) Level 3 verifies the close-loop current regulation performed by pid_i module

Figure 14. Level 3 – Incremental System Build Block Diagram

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