Dual-Axis Current Control of Brushed DC Motors

Jon Warriner   D3 Engineering

Abstract

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

This application note covers the following:

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

Table of Contents

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 ADC, ePWM, QEP, eCAP etc. These peripherals have all the necessary hooks for implementing systems which 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.

Thus, with C2000 family of devices it is easy and quick to implement complex control algorithms 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

- Decreases the number of look-up tables which 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.

- 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.

TI Literature and 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. Check the folders below for the source codes and explanations of macro blocks:

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

These modules allow users to quickly build, or customize, their own systems. The Library supports the five motor types: ACI, BLDC, PMSM, Brushed DC, Stepper Motor and comprises both peripheral dependent (software drivers) and target dependent modules.

The DMC Library components have been used by TI to provide system examples. At initialization all DMC Library variables are defined and inter-connected. At run-time the macro functions are called in order. Each system is built using an incremental build approach, which allows some sections of the code to be built at a time, 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 form of macros for optimization purposes (refer to application note *SPRAAK2* for more details at TI website). The macros are defined in the header files. The user 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 below which means that the code continue in the next line. Any character including invisible ones like "space" 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 the user can easily convert the macro definition to a C functions.

```
#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);
```

A typical DMC macro definition

For further information review the C2000 Motor Control Primer
http://focus.ti.com/lit/ug/sprugi6/sprugi6.pdf

System Overview

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

Macro Names	Explanation
PID	PID Regulators
RC	Ramp Controller (slew rate limiter)
PWM / PWMDAC	PWM and PWMDAC Drives
[1] Please refer to pdf documents in C:\ti\controlSUITE\libs\app_libs\motor_control explaining the details and theoretical background of each macro	

Table 1

In this system, current control of Brushed DC Motor will be demonstrated. The examples will start out with a single axis and will end with a multi-axis demonstration. The motor is driven by a H-bridge provided by the DRV8412 Dual Full Bridge PWM Motor Driver IC. The Piccolo F28035 MCU controlCARD is used to generate four pulse width modulation (PWM) signals, two for each motor. Two phase currents of each motor are measured from the H-bridge and sent to the F28035 via four analog-to-digital converters (ADCs). In addition, the DC-bus voltage is measured and sent to the F28035 via an ADC.

2xDC_Motor project has the following properties:

C Framework		
System Name	Program Memory Usage 2803x	Data Memory Usage[1] 2803x
2xDC_Motor	2753 words[2]	1060 words

Table 2

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

| CPU Utilization of 2-axis Brushed DC Current Control	
Name of Modules *	Number of Cycles
2 x Ramp Controller	49
2 x Pid	96
2 x Pwm Drv	133
A/D Feedback scaling	40
Contxt Save etc.	53
Pwm Dac (optional)	
DataLog (optional)	
Total Number of Cycles	**371 ****
CPU Utilization @ 60 Mhz	**6.2% ****
CPU Utilization @ 40 Mhz	**9.3% ****

Table 3

* The modules are defined in the header files as "macros"
** 457 including the optional modules
*** At 10 kHz ISR freq.

| 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

Table 4

The overall system implementing 2-axis brushed DC motor current control is depicted in Figure 1. The brushed DC motors are driven by the conventional H-bridge configuration. The F28035 is being used to generate 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 they are sent to the F28035 via four analog-to-digital converters (ADCs).
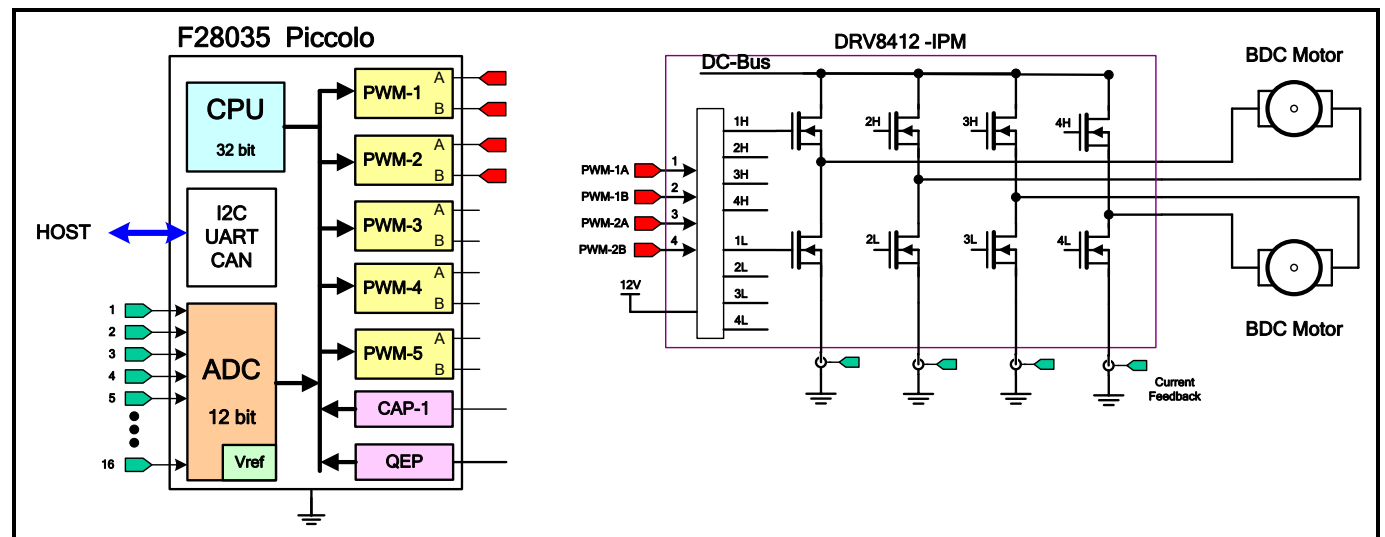
Figure 1 Dual-axis brushed DC motor drive implementation

The software flow is described in the Figure 2 below.
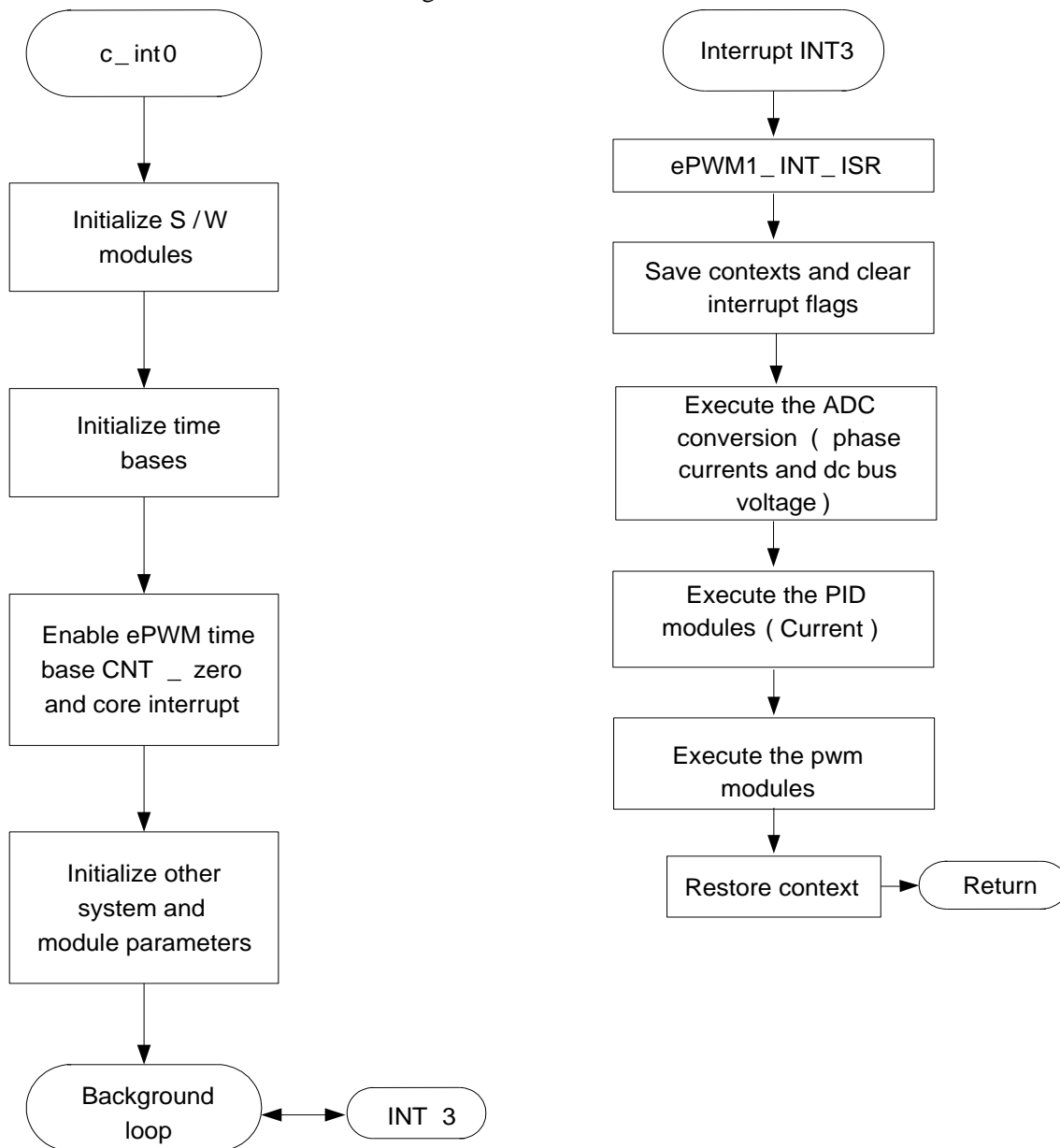
Figure 2 System software flowchart

Hardware Configuration (DRV8412-EVM)

Please refer to the DRV8412-EVM How to Run Guide and HW Reference Guide found:

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

for an overview of the kit's hardware and steps on how to setup this kit. Some of the hardware setup instructions are captured below for quick reference

HW Setup Instructions

1. Unpack the DIMM style controlCARD and verify that the DIP switch settings match Figure 3
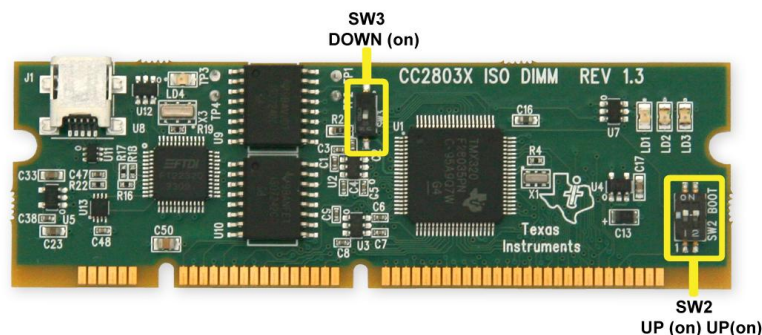
Figure 3 controlCARD DIP Switch Settings

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. Make sure DRV8412 mode jumpers and +12V source jumper are set according to Figure 4.

4. Connect a USB cable to connector J1 on the controlCARD. This 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.

5. Connect a 24V 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 light up as well indicating the control card is receiving power from the board.

6. Note that motor 1 should be connected to the OUTA and OUTB terminals and motor 2 should be connected to the OUTC and OUTD terminals after you finish with the first incremental build step.

For reference Figure 4 below shows the jumper and connectors that need to be connected for this lab.
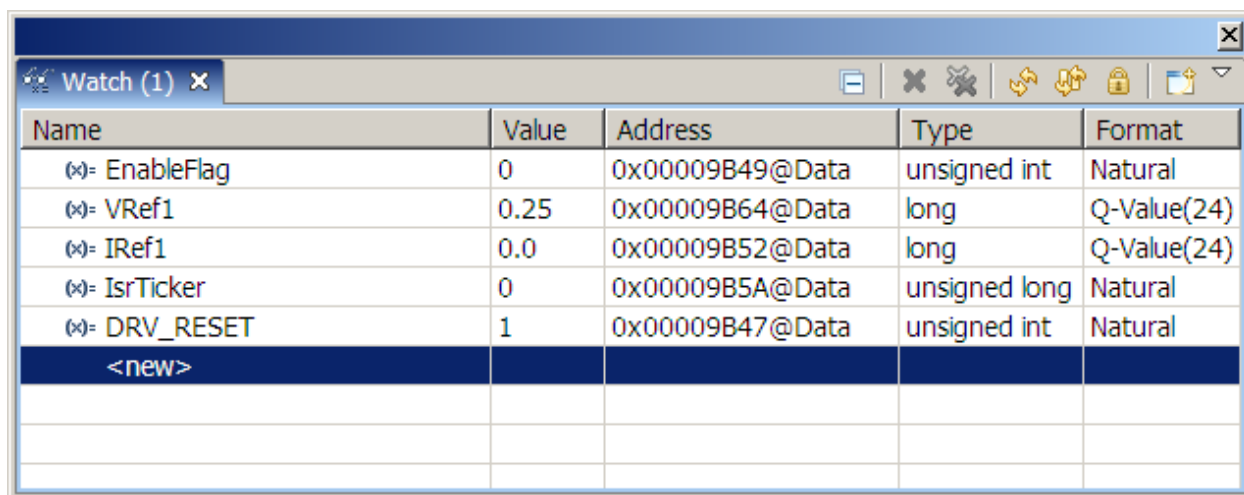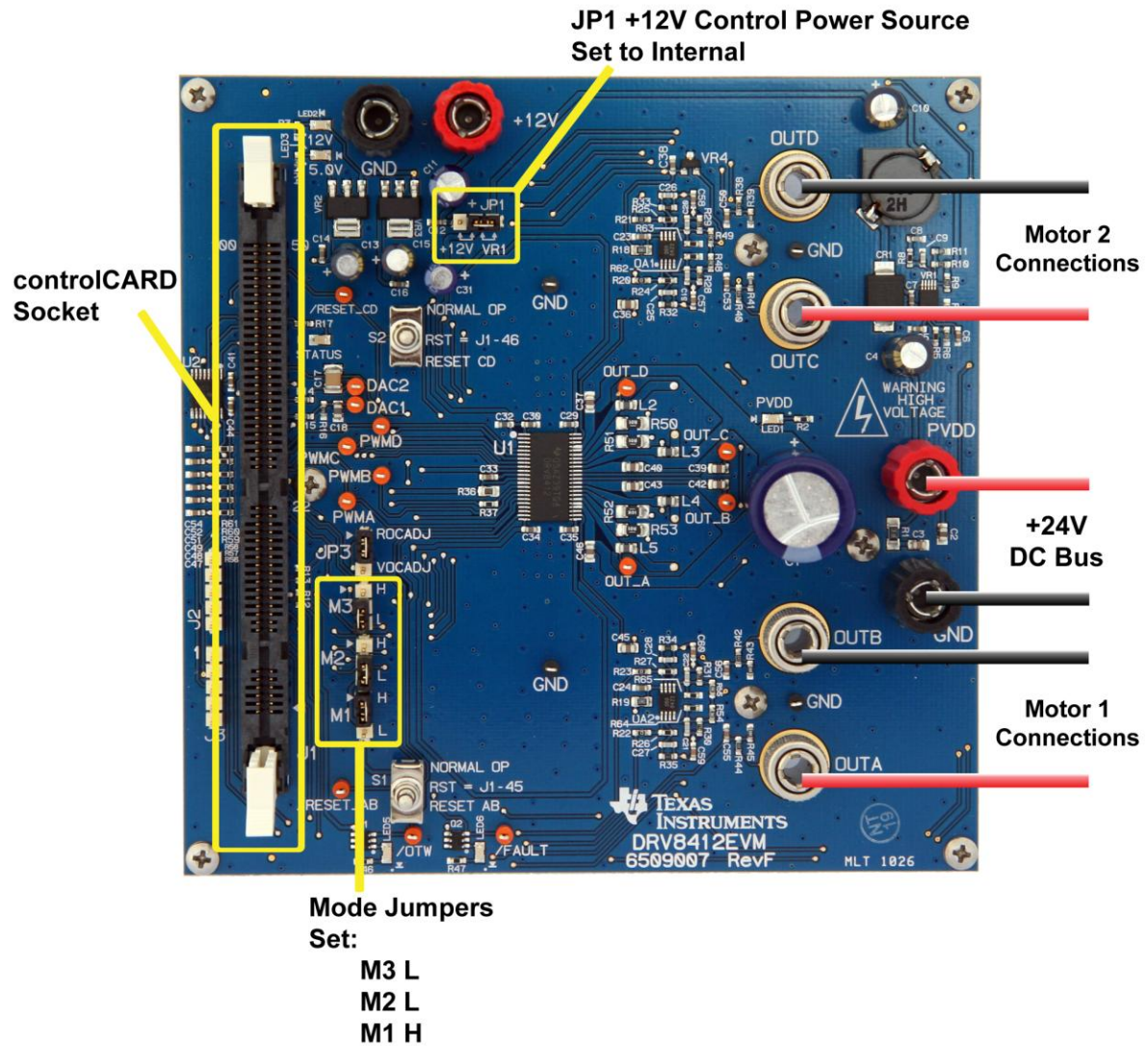

Figure 4 EVM Connections

Software Setup Instructions to Run Dual-Axis Brushed DC 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 goes over how to install CCS and set it up to run with this project.

Select the 2xDC_Motor as the active project. Verify that the build level is set to 1, and then right click on the project name and select "Rebuild Project". Once build completes, launch a debug session to load the code into the controller. Now open a watch window and add the critical variables as shown in Figure 5 below and select the appropriate Q format for them.

Name	Value	Address	Type	Format
(×)= EnableFlag	0	0x00009B49@Data	unsigned int	Natural
(×)= VRef1	0.25	0x00009B64@Data	long	Q-Value(24)
(×)= IRef1	0.0	0x00009B52@Data	long	Q-Value(24)
(×)= IsrTicker	0	0x00009B5A@Data	unsigned long	Natural
(×)= DRV_RESET	1	0x00009B47@Data	unsigned int	Natural
<new>				

Figure 5 Watch Window Variables

Setup time graph windows by importing graph1.graphProp and graph2.graphProp from the following location C:\TI\ControlSUITE\developement_kits\DRV8412-EVM\2xDC_Motor\. These graphs will operate similar to a strip chart with the update rate being configurable in Code Composer. To check the Continuous refresh rate click Window->Preferences. In the Preferences setup window expand CCS and then click on Debug. The window should look like the following:
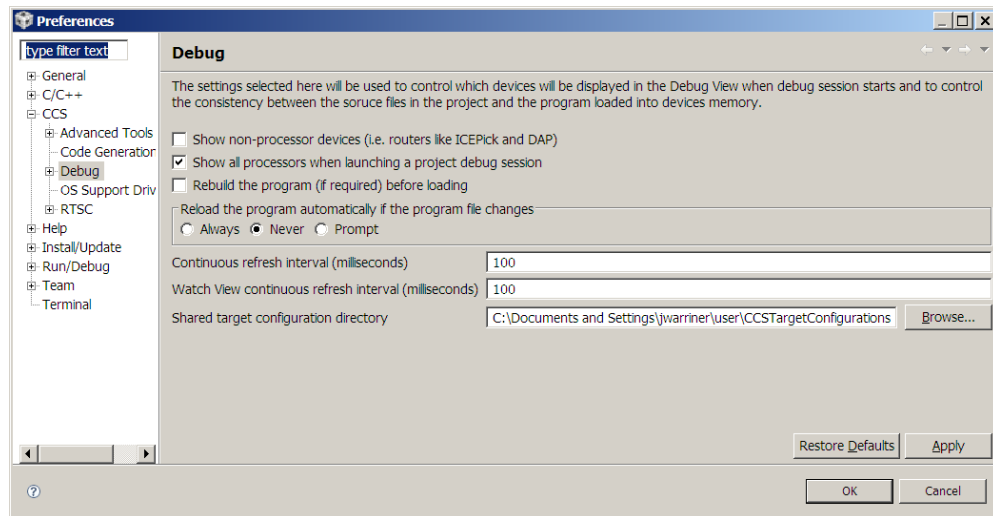
Figure 6 Continuous Refresh Interval configuration

Set the Continuous refresh interval to a suitable value, typically 100ms is a good choice. Click on Continuous Refresh button on the top left corner of the graph tab to enable periodic capture of data from the microcontroller.

Incremental System Build for Dual Axis Brushed DC Motor project

The system is gradually built up in order for the final system to be confidently operated. Four phases of the incremental system build are designed to verify the major software modules used in the system. Table 5 summarizes the modules testing and using in each incremental system build.

Software Module	Phase 1	Phase 2	Phase 3	Phase 4
PWMDAC_MACRO	√	√	√	√
RC_MACRO	√	√	√	√
PWM_MACRO	√√	√	√	√
A/D Conversion		√√	√	√
PID_MACRO			√√	√
Note: the symbol √ means this module is using and the symbol √√ means this module is testing in this phase.				

Table 5

Level 1 Incremental Build

At this step 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 system interrupt and the peripheral dependent PWM_MACRO (PWM initializations and update) modules. Open 2xDC_Motor-Settings.h and select level 1 incremental build option by setting the BUILDLEVEL to LEVEL1 (#define   BUILDLEVEL LEVEL1). Now right click on the project name and click Rebuild Project. Once the build is complete click on debug button, reset CPU, restart, enable real time mode and run. Set "EnableFlag" to 1 in the watch window. The variable named "IsrTicker" will now keep on increasing, 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.
▪ VRef1 (Q24): for changing the motor voltage in per-unit.
▪ DRV_RESET: for holding the DRV8412 chip in reset.

Level 1A (PWM_MACRO Test)

In this level we will test the PWM Generator Macro (PWM_MACRO). The VRef1 value is specified to the PWM_MACRO via RC_MACRO module. The rate of change of the PWM duty cycle is therefore slew rate limited and will not change instantaneously with VRef1. Check the PWM test points on the board to observe PWM pulses (PWMA and PWMB for Motor = 1 and PWMC and PWMD for Motor = 2) and make sure that the PWM module is running properly. When VRef1 is positive PWMA (PWMC for Motor = 2) should show a 10KHz square wave with a duty cycle corresponding to the magnitude of VRef1 and PWMB (PWMD for Motor = 2) should be held low.
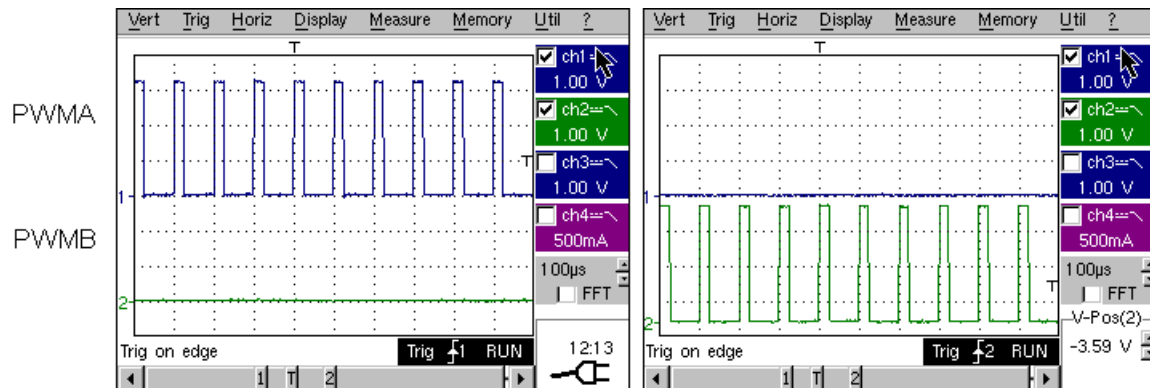

**Figure 7 PWMA and PWMB for VRef1 = 0.25 and -0.25**

Level 1B (testing The PWMDAC Macro)

To monitor internal signal values in real time PWM DACs are very useful tools. Present on the DRV8412-EVM board are PWM DAC's which use an external low pass filter to generate the waveforms (Test point labeled DAC2). A simple 1st–order low-pass filter RC circuit is used to filter out the high frequency components. The selection of R and C value (or the time constant, τ) is based on the cut-off frequency (f_c), for this type of filter; the relation is as follows:

$$\tau = RC = \frac{1}{2\pi f_c}$$

For example, R=470Ω and C=0.47uF, it gives f_c = 720.5 Hz. This cut-off frequency has to be below the PWM frequency. Using the formula above, one can customize low pass filters used for signal being monitored. Refer to application note *SPRAA88A* for more details at TI website.

The PWM DACs should represent an IQ range of +/-1.0 with an analog range of 0-3.3V.
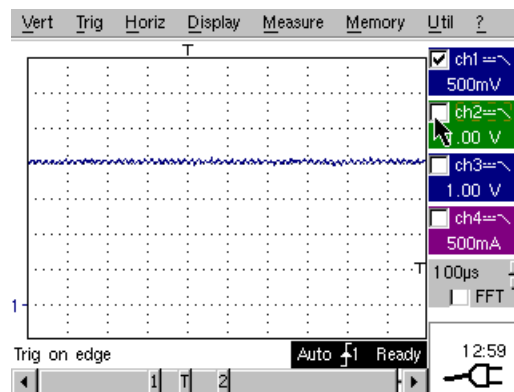
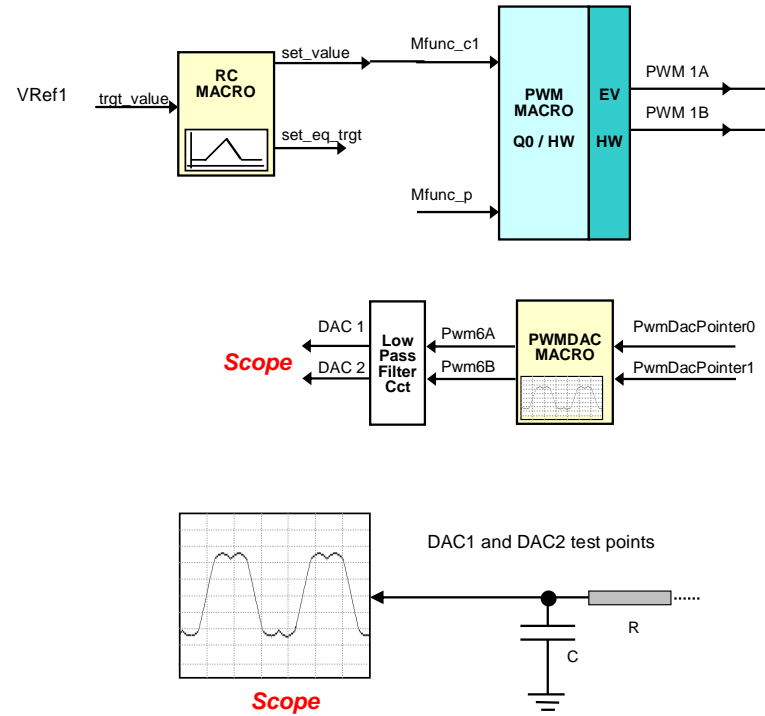**Figure 8 PWM DAC2 output for VRef1 = 0.25**

Level 1C (DRV8412 Test)

After verifying PWM_MACRO module in level 1a, the dual full-bridge hardware in the DRV8412 are tested by looking at the low pass filter outputs. For this purpose, set the variable DRV_RESET to 0 through the watch window and check the ADC_Vhb1 and ADC_Vhb2 schematic nodes using an oscilloscope. You will observe that the phase voltage dividers and waveform monitoring filters (R42 to 45 & C55 to 56) enable the generation of the waveform and ensure that the DRV8412 is working appropriately.

> ⚠️ After verifying this, set the variable DRV_RESET to 1, take the controller out of real time mode (disable), reset the processor 🐛 (see "DRV8412-EVM How To Run Guide" for details). 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, hence caution needs to be taken while doing

After verifying the first axis, repeat the same tests for the second axis. Open 2xDC_Motor-Settings.h, set the variable "Motor" to 2, and repeat the same steps.

Level 1 - Incremental System Build Block Diagram

VRef1 —trgt_value→ **RC MACRO** —set_value→ Mfunc_c1 → **PWM MACRO Q0 / HW** | **EV** / **HW** → PWM 1A, PWM 1B

set_eq_trgt

Mfunc_p

DAC 1 ← **Low Pass Filter Cct** ←Pwm6A— **PWMDAC MACRO** ← PwmDacPointer0
Scope DAC 2 ←Pwm6B— ← PwmDacPointer1

Scope

DAC1 and DAC2 test points

R

C

Level 1 verifies the target independent modules, duty cycles and PWM update. The motors are disconnected at this level.

16

Level 2 Incremental Build

Assuming the Level1 incremental build is completed successfully, this section verifies the analog-to-digital conversion. Connect the motor to the board. In the software, the key variables to be adjusted are summarized below.

- VRef1 (Q24): for changing the motor voltage in per-unit.

Open 2xDC_Motor-Settings.h and select level 2 incremental build option by setting the BUILDLEVEL to LEVEL2 (#define   BUILDLEVEL LEVEL2) and save the file. Now Right Click on the project name and click Rebuild Project. Once the build is complete click on debug button, reset CPU, restart, enable real time mode and run. Set "EnableFlag" to 1 in the watch window. The variable named "IsrTicker" will be incrementally increased as seen in watch windows to confirm the interrupt working properly.

Testing the A/D Conversion

In this part, the A/D conversion will be tested. Using the Watch Window set VRef1 to 0.2 pu. The motor should start turning slowly. Enable Continuous Refresh 🔄 for the graph window and you should see a relatively small value for IFdbk1. Put a load on the motor (i.e. grab the motor shaft) and you should see the value of IFdbk1 increase.
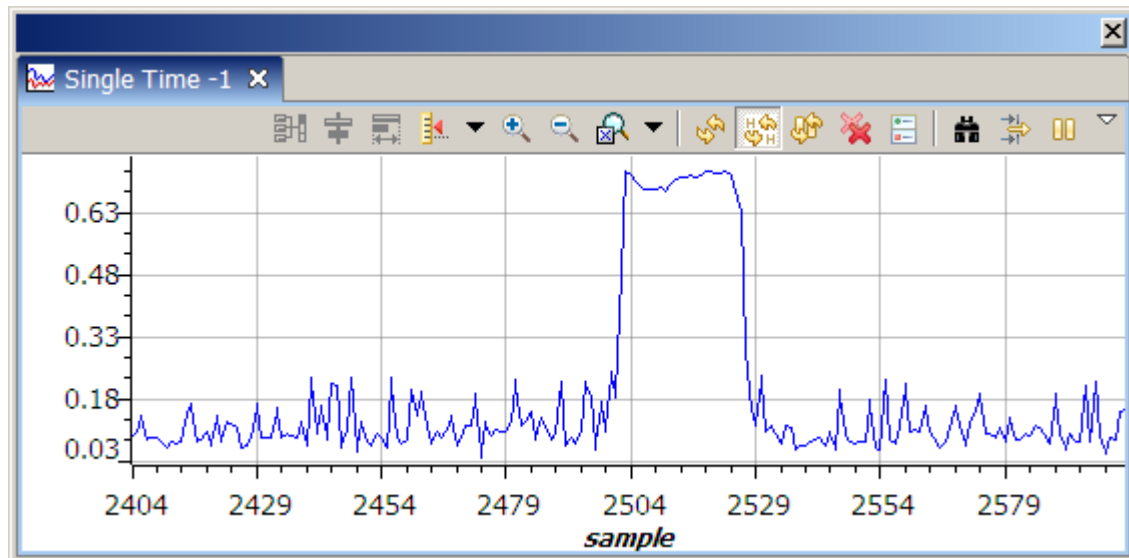

Figure 9 Build Level 2 graph showing current feedback under free running and stalled conditions

Note that the open loop experiments are meant to test the ADCs, DRV8412, sw modules etc. Therefore running motor under load or at various operating points is not recommended.

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 realtime mode and reset.

After verifying the first axis, repeat the same tests for the second axis. Open 2xDC_Motor-Settings.h, set the variable "Motor" to 2, and repeat the same steps

Level 2 - Incremental System Build Block Diagram

Graph Window
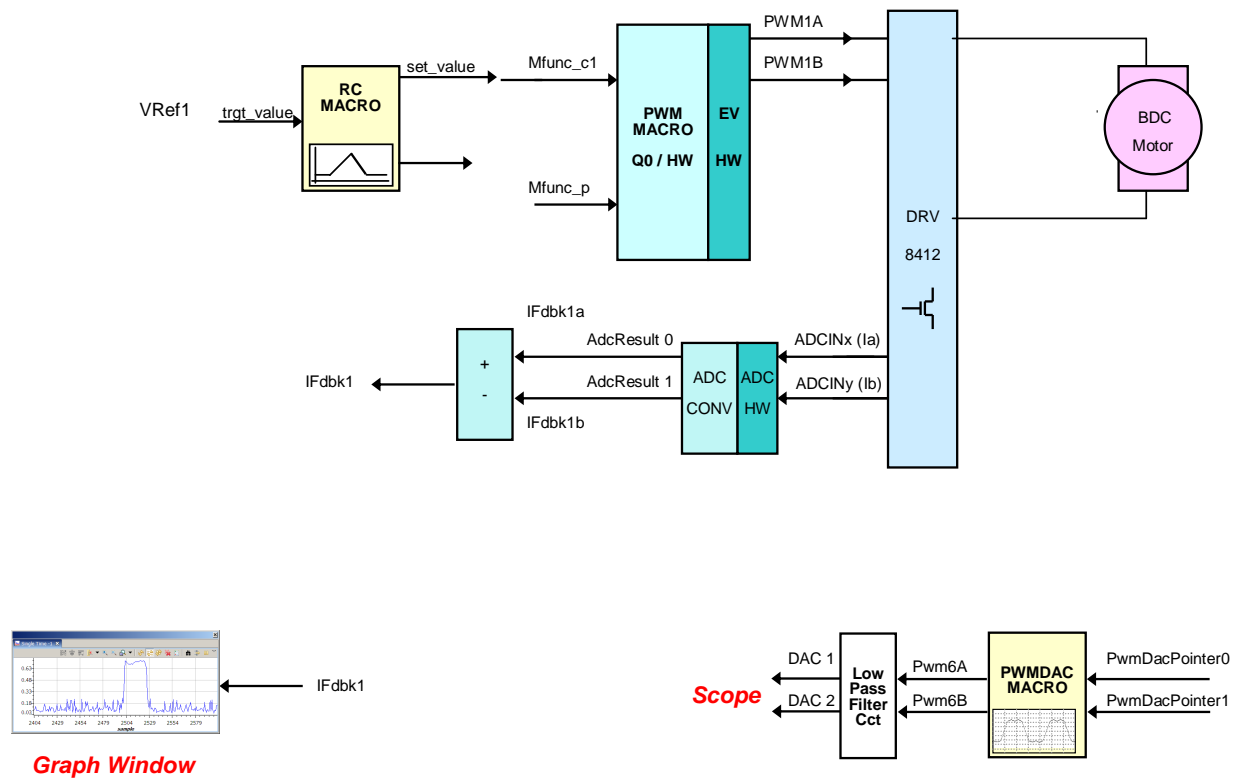
Level 2 verifies the analog-to-digital conversion

Level 3 Incremental Build

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

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

▪ IRef1(Q24): for changing the motor current in per-unit.

In this build, the motor is supplied by DC input voltage and the (DC) motor current is dynamically regulated by using PID_REG3 module.

The steps are explained as follows:

▪ Launch a debug session, enable real-time mode and run the project.

▪ Set IRef1 to 0.25 pu (or another suitable value if the base current is different).

▪ Put some sort of load on the motor (i.e. Grab the motor shaft). An unloaded motor will rapidly accelerate to the maximum speed allowed by the DC bus voltage and will not regulate current.

▪ Check IFdbk1 in the graph window with the continuous refresh feature. It should be keeping track of IRef1 for PID_REG3 module. If not, adjust its PID gains properly.
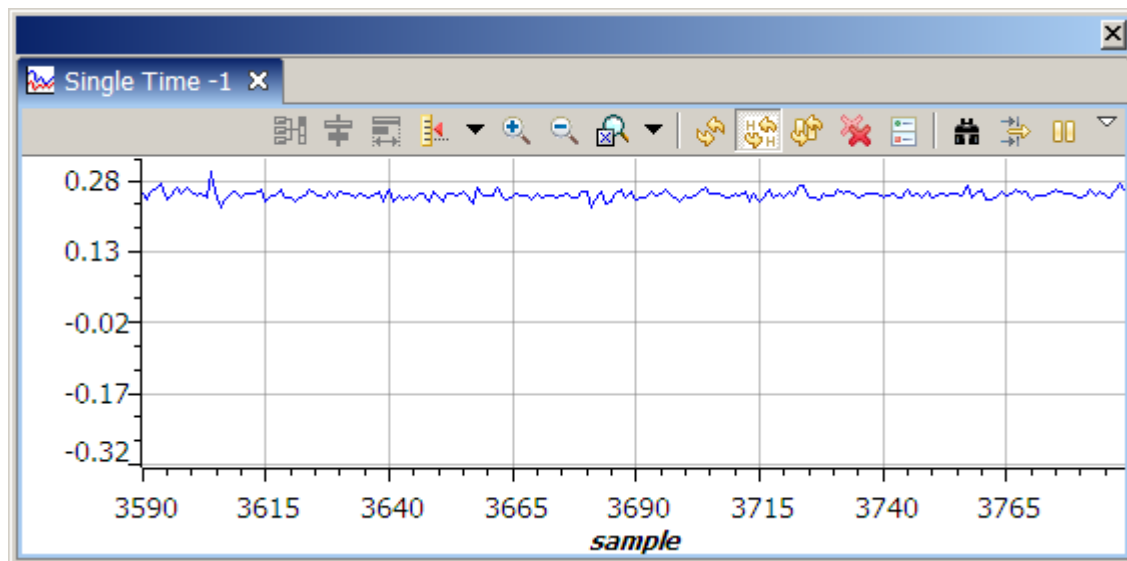

Figure 10 Build Level 3 graph of regulated motor current

▪ To confirm the PID module, try different values of IRef1.

- 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 realtime mode and reset.

After verifying the first axis, repeat the same tests for the second axis. Open 2xDC_Motor-Settings.h, set the variable "Motor" to 2, and repeat the same steps.

Level 3 - Incremental System Build Block Diagram
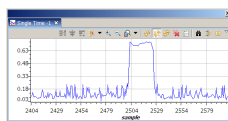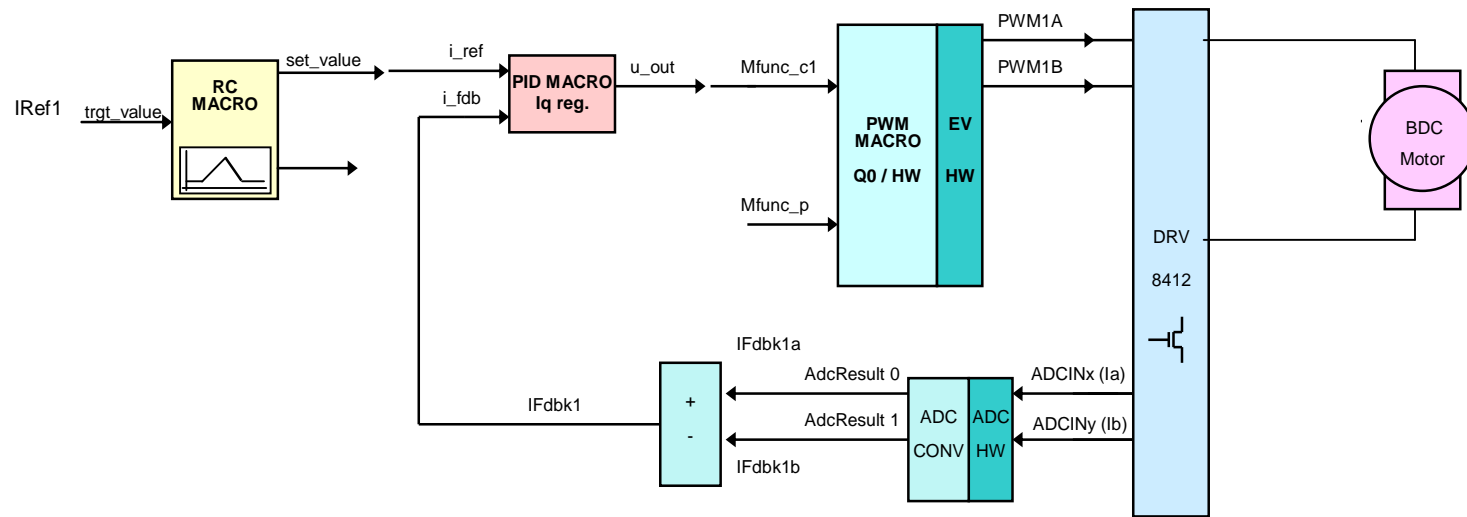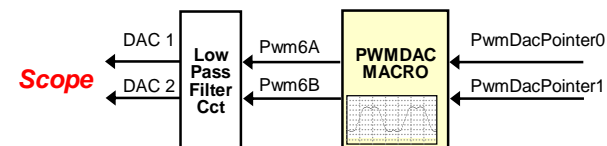
Level 3 verifies the cloose loop current regulation performed by pid_i module

Level 4 Incremental Build

Assuming two motors are connected and the previous section was completed successfully; this section verifies the multi axis concept; which means two DC motors are run simultaneously with current control.

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

- IRef1 (Q24): for changing the motor 1 current in per-unit.
- IRef2 (Q24): for changing the motor 2 current in per-unit.
- IFdbk1 (Q24): for viewing motor 1 current feedback in per-unit.
- IFdbk2 (Q24): for viewing motor 2 current feedback in per-unit.

The key steps can be explained as follows:

- Set IRef1 to 0.25 pu (or another suitable value if the base speed is different).

- Put some sort of load on the motor (i.e. Grab the motor shaft). An unloaded motor will rapidly accelerate to the maximum speed allowed by the DC bus voltage and will not regulate current.

- Check IFdbk1 in the graph window with continuous refresh feature. It should be keeping track of IRef1 for PID_REG3 module.

- Repeat the same tests for the second axis by modifying the value of IRef2 and viewing IFdbk2 in the second graph window.

Level 4 - Incremental System Build Block Diagram
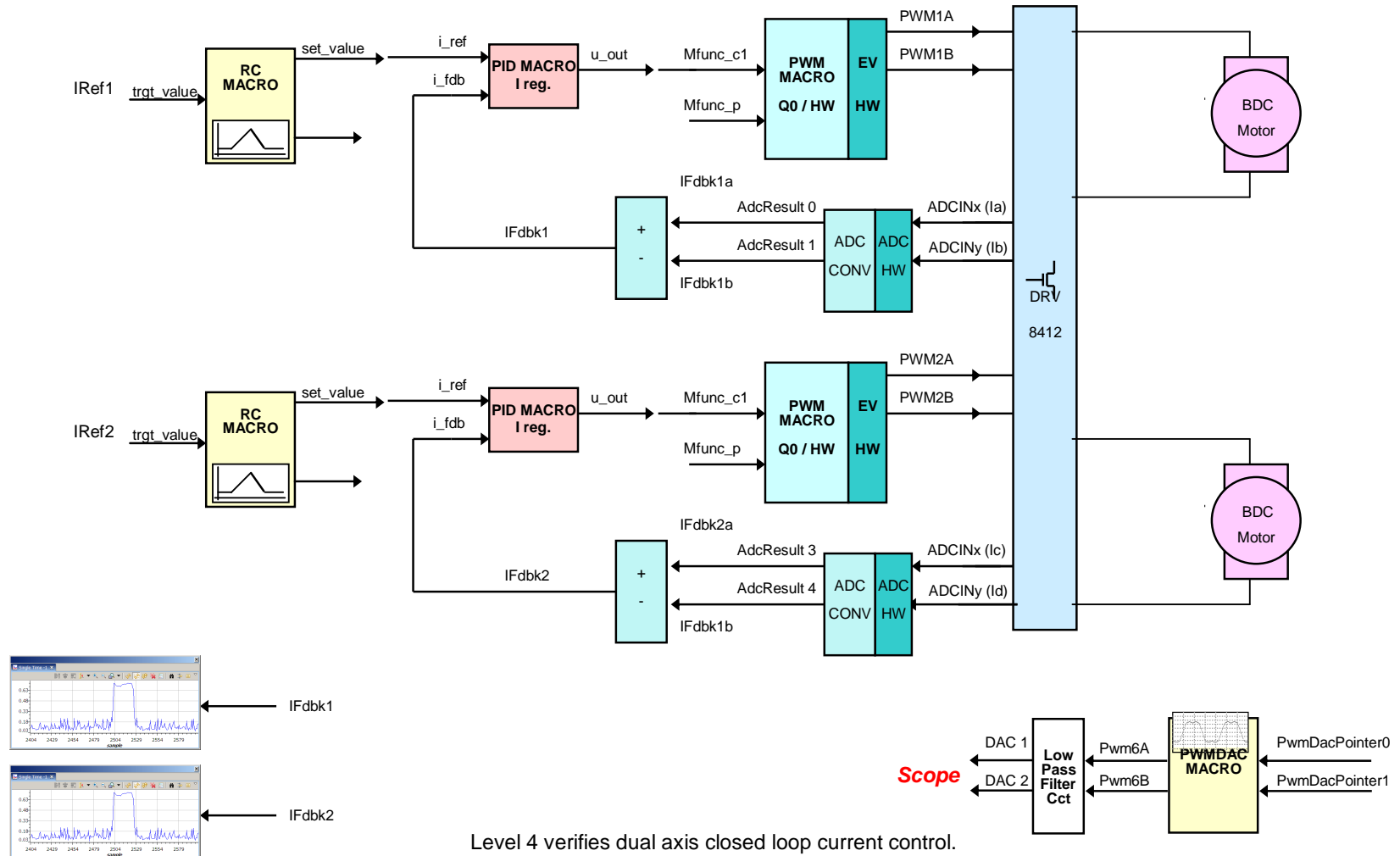
Level 4 verifies dual axis closed loop current control.