Version: Thursday, April 03, 2014

Applies to: SAT0042 E4 brushless DC motor drive board

# **1** Initial Evaluation

- **1.1 Visual inspection**
- 1.1.1 Verify the components are correctly installed
- 1.1.2 Verify the DNP components are not installed



- 1.1.3 Verify the blue wires (if any) are correctly installed not applicable
- 1.1.4 Verify pin 1 orientation and diode orientation
- 1.2 Apply Vbatt (12V) power to board
- 1.2.1 GND to J1-1 and J1-4, +12V to J1-2 and J1-3
- **1.2.2 Green LED D6 should be illuminated.**



Figure 1 SAT0042 E4 board with 12V power applied and LED D6 illuminated (center)

#### **1.2.3** Measure 12V current (no-load, idle conditions)

Measures approximately 20 mA at 12V before loading microcontroller code.

- 1.2.4 Verify DC+ (TP1) is 12V with respect to GND (TP2)
- 1.2.5 Verify +3.3V net is 3.3V; measure +3.3V at J5-4

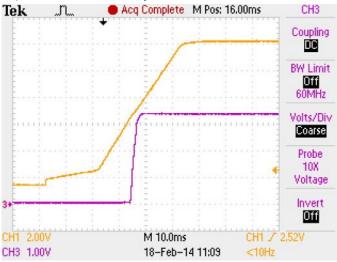
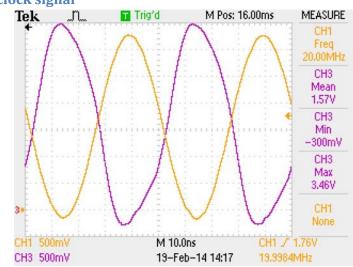


Figure 2 Top (orange) trace is 12V from bench supply, bottom (pink) trace is 3.3V on board

**1.2.6** Measure ADC\_REF at TP5, should be 3.3V if R36 is installed, should be 3V if R38 is installed. R38 is installed (R36 is not installed) – measurement is 3.08V.



1.2.7 Verify the 20 MHz clock signal

Figure 3 Scope traces of 20 MHz clock signal

### 1.3 Vary the Vbatt applied

1.3.1 Reduce the Vbatt (12V nominal) voltage until the Green LED is no longer illuminated. Record the voltage at which the Green LED transitions.

LED first illuminates at 6V, and turns off when supply is turned down to 4V.

### **1.3.2** Measure the input current (no-load, idle conditions) at 8V, 12V, 16V 19 mA @ 8V, 15 mA @ 12V, 13 mA @16V (approximately 180 mW)

### **1.4 Test point voltages**

	Test Point Color	Signal Name	Typical measurement value
TP1		DC+	12V (set by external supply)
TP2	Black	GND	< 40 mV
TP3	Black	GND	< 40 mV
TP4	Black	GND	< 40 mV
TP5		ADC_REF	3.08 V
TP6		DC_Volt	1.04V
TP7		Phase B Voltage	Depends on motor conditions
TP8		Phase B Current	Depends on motor conditions
TP9		Phase A Voltage	Depends on motor conditions
TP10		Phase A Current	Depends on motor conditions
TP11		Phase C Voltage	Depends on motor conditions
TP12		Phase C Current	Depends on motor conditions

# **1.5 Motor voltage test point frequency response**

Table 1 Frequency measurements of motor phase test points (TP7, TP9, TP11)

Input from signal generator, 2Vpp				
Measured at phase voltage test points				
No power on boar	No power on board			
Frequency (Hz)	Α	В	С	
	TP9	TP7	TP11	
100	250			
200	250			
500	212			
1000	134	139	129	
2000	78	84	78	
5000	37	42	37	
10000	24	31	21	
20000	18	25	18	
50000	17	23	13	

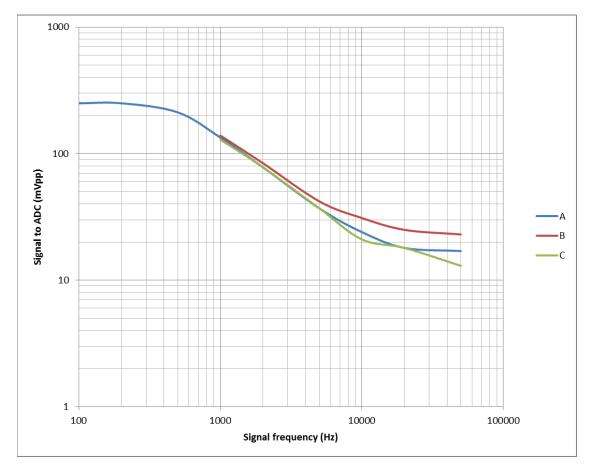


Figure 4 Frequency response of motor phase voltage filters

### **1.6 Reverse polarity protection**

The board has components that should prevent damage during reverse polarity conditions on VBATT with respect to BATT\_GND. When VBATT < VGS(th), then Q1 disconnects BATT\_GND from the GND node.

No observable current from supply with reversed leads to J1.

# 2 Load C2000 software

### 2.1 connect to JTAG, load code with CCS, run in CCS debug environment,

Code Composer Studio<sup>™</sup> (CCStudio) is an integrated development environment (IDE) for Texas Instruments (TI) embedded processor families. CCStudio comprises a suite of tools used to develop and debug embedded applications. It includes compilers for each of TI's device families, source code editor, project build environment, debugger, profiler, simulators, real-time operating system and many other features. The intuitive IDE provides a single user interface taking you through each step of the application development flow. Familiar tools and interfaces allow users to get started faster than ever before and add functionality to their application thanks to sophisticated productivity tools.

See the Code Composer Studio web page at <u>http://www.ti.com/tool/ccstudio</u> for information on downloading the integrated development environment for the C2000 code.

Step 1: Import the existing project, for example proj\_lab02b, from the motorware directory. In this instance, there are two projects in the directory.

Import CCS Eclipse Projec	ts	
Select Existing CCS Eclipse Project Select a directory to search for existing CCS Eclipse projects.		
Select search-directory:	C:\ti\motorware_1_01_00_11	Browse
Select archive file:		Browse
Discovered projects:		
	i\motorware_1_01_00_11\motorware_1_01_00_11\sw\solutions\instaspin_foc\boards\di \ti\motorware_1_01_00_11\motorware_1_01_00_11\sw\solutions\instaspin_foc\boards\	
	(n/notorware_1_or_oo_11/notorware_1_or_oo_11/sw/solutions/instashin_loc/boards/i	Deselect All
		Refresh
•	4	
Copy projects into work	space	
Automatically import re		
Open the Percurse Explorer	and browse available example projects	
open the resource explorer	and browse available example projects	
(?)	Finish	Cancel

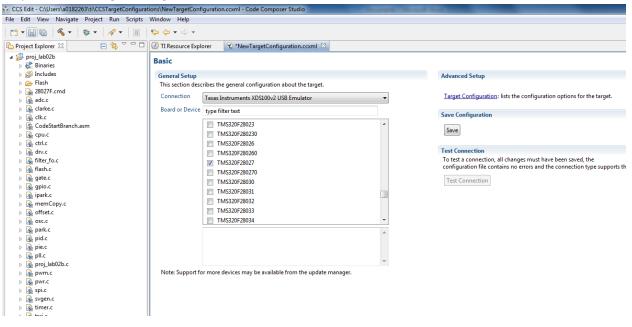
Figure 5 Import Existing CCS Eclipse Project screen

Step 2: Import the project 2b



Figure 6 Imported project file and sub-files

Step 3: Set the target configuration:



#### Figure 7 New target configuration screen

The connection will depend on the JTAG emulator you use. The target device on this board is the TMS320F28027 picollo microcontroller. After selecting the connection and target device, save the configuration set-up by clicking the "Save" button.

Step 4: Test the connection to the target.

#### Figure 8 Test connection window after completion of connection test

# Step 5: Build the project.

🖳 Console 🛛 🕴 🖓 👘 🔛 🖛 📑 🖛 🗮 🖛 👘 🖛 👘		🛃 Problems 🖾
CDT Build Console [proj_lab02b]		0 items
ursapre_anro_ursuraB_anbuses=rooosuraB_anbuses=rooos-n		<b>A</b>
xml link info="proj lab02b linkInfo.xml"entry point=code startrom model -o		Description
"proj_lab02b.out" -1"rts2800_ml.lib"		
"C:/ti/motorware_1_01_00_11/motorware_1_01_00_11/sw/modules/fast/lib/32b/f28x/f2802x/fast_pub		
lic.lib"		
"C:/ti/motorware_1_01_00_11/motorware_1_01_00_11/sw/modules/fast/lib/32b/f28x/f2802x/2802xRevB		
_Fast_ONLY_ROMSymbols.lib"		
"C:/ti/motorware_1_01_00_11/motorware_1_01_00_11/sw/modules/iqmath/lib/f28x/32b/IQmath.lib"		
"./wdog.obj" "./user.obj" "./usDelay.obj" "./traj.obj" "./timer.obj" "./svgen.obj"		
"./spi.obj" "./pwr.obj" "./pwm.obj" "./proj lab02b.obj" "./pll.obj" "./pie.obj" "./pid.obj"		
"./park.obj" "./osc.obj" "./offset.obj" "./memCopy.obj" "./ipark.obj" "./gpio.obj"		
"./gate.obj" "./flash.obj" "./filter fo.obj" "./drv.obj" "./ctrl.obj" "./cpu.obj" "./clk.obj"		
"./clarke.obj" "./adc.obj" "./CodeStartBranch.obj"		
"C:/ti/motorware 1 01 00 11/motorware 1 01 00 11/sw/ide/ccs/cmd/f2802x/28027F.cmd"		
<pre><linking></linking></pre>		
'Finished building target: proj lab02b.out'		
**** Build Finished ****		
	-	4

#### Figure 9 CCS console window after successful build of the project

Step 6: Start a Debug session with the project.

e Edit View Project Tools Run Scripts Window Help			
🗈 🕶 🔚 👘 🕶 🔛 🎭 🕶 🔛 🍬 🛹 🕶			
۶ Debug 🛿 👘 🦓 🕪 🕕 🔳 🔍 👁 کې 👁 دی 🌸 🛪 🕹 🏟	▽ □ □ 🖾 Variables 🕸 Expressions 🔀 👭 Re	gisters	
🕡 😳 proj_lab02b [Code Composer Studio - Device Debugging]	Expression	Туре	Value
Texas Instruments XDS100v2 USB Emulator_0/C28xx (Running)	iproj_lab02b.c'::gMotorVars	struct_MOTOR_Vars_t_	{}
	(x)= Flag_enableSys	unsigned int	1
	(x)= Flag_Run_Identify	unsigned int	0
	(x)= Flag_MotorIdentified	unsigned int	0
	(x)= Flag_enableForceAngle	unsigned int	1
	(x)= Flag_enableFieldWeakening	unsigned int	0
	(x)= Flag_enableRsRecalc	unsigned int	0
	(x)= Flag_enableUserParams	unsigned int	1
	(x)= Flag_enableOffsetcalc	unsigned int	1
	(x)= Flag_enableEpl	unsigned int	0
	(x)= CtrlState	enum unknown	CTRL_State_Idle
	(x)= EstState	enum unknown	EST State Idle
	(×)= IdRef_A	long	0
	(x)= IqRef_A	long	0
	(×)= SpeedRef_krpm	long	0.09999996424 (Q-Value(24))
	(x)= SpeedTraj_krpm	long	0
	(x)= MaxAccel_krpmps	long	3355443
	(x)= Speed_krpm	long	0
	(x)= Torque_lbin	long	0
	(x)= OverModulation	long	16777216
	(x)= RsOnLineCurrent_A	long	8388608
	(x)= SvgenMaxModulation_ticks	long	400
	(x)= MagnCurr_A	float	0.0
	(×)= Rr_Ohm	float	0.0
	(x)= Rs_Ohm	float	0.0
	(x)= RsOnLine_Ohm	float	0.0
	(x)= Lsd_H	float	0.0
	(x)= Lsq_H	float	0.0
	(x)= Flux_VpHz	float	0.001616162
	(×)= Kp_spd	long	0
	(×)= Ki_spd	long	0
	(x)= Kp_Idq	long	0
	(×)= Ki_Idq	long	0
	(x)= Vd	long	0
	(×)= Vq	long	0
	(x)= Vs	long	0
	(×)= VsRef	long	15099494
	(x)= VdcBus_kV	long	0.007869780064 (Q-Value(24))
	(×)= Id_A	long	0
	(×)= Iq_A	long	0
	(x)= Is_A	long	0
	I_bias	struct _MATH_vec3_	{}
	V_bias	struct _MATH_vec3_	{}
	🚽 Add new expression		

Figure 10 Expressions window in the debug view of CCS, program running

Step 7: Run the project

Change the expression VdcBus\_kV to Q-Value(24) by right-clicking in the Value field, then selecting Q-values and "24". Verify that the value for VdcBus\_kV corresponds to the DC supply voltage. In the first case (below), the supply voltage is 8V.

10	5 A 1 -		-	
	(×)= VsRef	long	15099494	0x0000007C@Data
	(×)= VdcBus_kV	long	0.007869780064 (Q-Value(24))	0x0000007E@Data
	(×)= Id_A	long	0	0x0000080@Data

In the second case (below), the supply voltage is 7V.

(×)= Vs	long	0	0x0000007A@Data
(x)= VsRef	long	15099494	0x0000007C@Data
(×)= VdcBus_kV	long	0.006873965263 (Q-Value(24))	0x0000007E@Data
(×)= Id_A	long	0	0x00000080@Data
A ST A	1	A	0.000000000000

In the third case (below), the supply voltage is 12V.

(×)= Vs	long	0	0x0000007A@Data
(×)= VsRef	long	15099494	0x0000007C@Data
(×)= VdcBus_kV	long	0.01185280085 (Q-Value(24))	0x0000007E@Data
(×)= Id_A	long	0	0x00000080@Data

Step 8a: Set the Flag\_enableSys to 1 by clicking in the value field and entering a 1.

Step 8b: Set the Flag\_Run\_Identify to 1 by clicking in the value field and entering a 1.

Motor will be driven with small and large motions, drawing up to 5+ Amps. After about a minute, the Flag\_MotorIdentified is set to 1 by the controller. This indicates the motor has been successfully identified for sensorless operation.

xpression	Туре	Value	Address
🛭 🥭 'proj_lab02b.c'::gMotorVars	struct _MOTOR_Vars_t_	{}	0x00000040@Data
(x)= Flag_enableSys	unsigned int	1	0x00000040@Data
(x)= Flag_Run_Identify	unsigned int	0	0x00000041@Data
(x)= Flag_MotorIdentified	unsigned int	1	0x00000042@Data
(x)= Flag_enableForceAngle	unsigned int	1	0x00000043@Data
(x)= Flag_enableFieldWeakening	unsigned int	0	0x00000044@Data
(x)= Flag_enableRsRecalc	unsigned int	0	0x00000045@Data
(x)= Flag_enableUserParams	unsigned int	1	0x00000046@Data
(x)= Flag_enableOffsetcalc	unsigned int	1	0x00000047@Data
(x)= Flag_enableEpl	unsigned int	0	0x00000048@Data
(×)= CtrlState	enum unknown	CTRL_State_Idle	0x00000049@Data
(x)= EstState	enum unknown	EST_State_Idle	0x0000004A@Data

pression	Туре	Value	Address
'proj_lab02b.c'::gMotorVars	struct _MOTOR_Vars_t_	{}	0x00000040@Dat
(x)= Flag_enableSys	unsigned int	1	0x00000040@Dat
(x)= Flag_Run_Identify	unsigned int	1	0x00000041@Dat
(×)= Flag_MotorIdentified	unsigned int	1	0x00000042@Dat
(x)= Flag_enableForceAngle	unsigned int	1	0x00000043@Dat
(x)= Flag_enableFieldWeakening	unsigned int	0	0x00000044@Dat
(x)= Flag_enableRsRecalc	unsigned int	0	0x0000045@Dat
(x)= Flag_enableUserParams	unsigned int	1	0x00000046@Dat
(x)= Flag_enableOffsetcalc	unsigned int	1	0x00000047@Dat
(x)= Flag_enableEpl	unsigned int	0	0x00000048@Dat
(x)= CtrlState	enum unknown	CTRL_State_OnLine	0x00000049@Dat
(x)= EstState	enum unknown	EST_State_OnLine	0x0000004A@Da
(x)= IdRef_A	long	0	0x0000004C@Da
(x)= IqRef_A	long	0	0x0000004E@Da
(x)= SpeedRef_krpm	long	0.3000000119 (Q-Value(24))	0x00000050@Da
(×)= SpeedTraj_krpm	long	5033158	0x00000052@Da
(x)= MaxAccel_krpmps	long	3355443	0x00000054@Da
(x)= Speed_krpm	long	0.3000246882 (Q-Value(24))	0x00000056@Da
(×)= Torque_lbin	long	0.008493959904 (Q-Value(24))	0x00000058@Da
(x)= OverModulation	long	16777216	0x0000005A@Da
(x)= RsOnLineCurrent_A	long	0.5 (Q-Value(24))	0x0000005C@Da
(x)= SvgenMaxModulation_ticks	long	400	0x0000005E@Da
(x)= MagnCurr_A	float	0.0	0x00000060@Da
(×)⊧ Rr_Ohm	float	0.0	0x00000062@Da
(×)= Rs_Ohm	float	1.101674	0x00000064@Da
(x)= RsOnLine_Ohm	float	0.0	0x00000066@Da
(×)⊧ Lsd_H	float	1.182403e-09	0x00000068@Da
(×)⊧ Lsq_H	float	1.182403e-09	0x0000006A@Da
(x)= Flux_VpHz	float	0.03356038	0x0000006C@Da
(×)⊧ Kp_spd	long	0	0x0000006E@Da
(x)= Ki_spd	long	0	0x00000070@Da
(×)⊧ Kp_Idq	long	0	0x00000072@Da
(×)⊧ Ki_Idq	long	0	0x00000074@Da
(×)= Vd	long	0	0x00000076@Da
(×)= Vq	long	0	0x00000078@Da
(×)= Vs	long	0	0x0000007A@Da
(×)= VsRef	long	15099494	0x0000007C@Da
(×)= VdcBus_kV	long	0.01183354855 (Q-Value(24))	0x0000007E@Da
(×)= Id_A	long	0	0x0000080@Da
(×)= Iq_A	long	0	0x00000082@Da
(×)= Is_A	long	0	0x0000084@Da
⊳ 📁 I_bias	struct _MATH_vec3_	{}	0x0000086@Da
b 🥮 V_bias	struct _MATH_vec3_	{}	0x000008C@Da

Figure 11 Exressions window in CCS debug view during motor operation

		From drv.c			
//c	onfigure	the SOCs for drv8301_027_ref			
//	sample t	he first sample twice due to <u>errata</u> sprz342f			
//d	//drv8301_027_ref				
//	ADC-A0	ADC_REF			
//	ADC-A1	IA-FB x			
//	ADC-A2	AIO2 mode (LED)			
//	ADC-A3	IC-FB x			
//	ADC-A4	AIO4 mode (U5 OUT)			
//	ADC-A5	internal <u>temp</u> sensor			
//	ADC-A6	IC-FB			
//	ADC-A7	ADC-Vhb2 (phase B) x			
//	ADC-B0	not available on 027			
//	ADC-B1	IB-FB x			
//	ADC-B2	VDCBUS x			
//	ADC-B3	IA-FB			
//	ADC-B4	ADC-Vhb3 (phase C) x			
//	ADC-B5	not available on 027			
//	ADC-B6	IB-FB			
//	ADC-B7	ADC-Vhb1 (phase A) x			

#### Table 2 Analog-to-digital converter assignments on SAT0042 E4 board

# 2.2 read ADC measurements of, DC voltage at idle, currents should be zero

Supply voltage	VdcBus_kV (Q-Value(24))	Voltage
8V	0.00786	7.86 V
12V	0.01183	11.83 V
15V	0.01481	14.81 V

# 3 Spin a motor ("kit" motor)

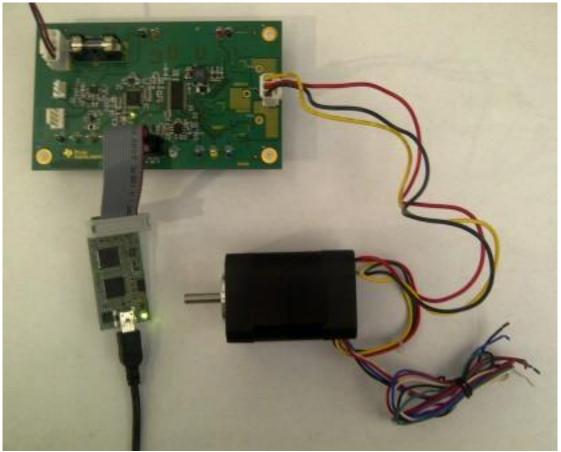


Figure 12 SAT0042 E4 board and Telco motor

- 3.1 Control the motor drive functions through CCS/JTAG
- 3.2 Use InstaSpin to identify motor parameters
- 3.3 Run at speed with nominal supply, no load, record currents, voltages

Kit motor	12VDC
Speed	Current
0 before loading	0.041
3000	0.35

# 4 Spin a pump motor (Cooper Standard 50W water pump)

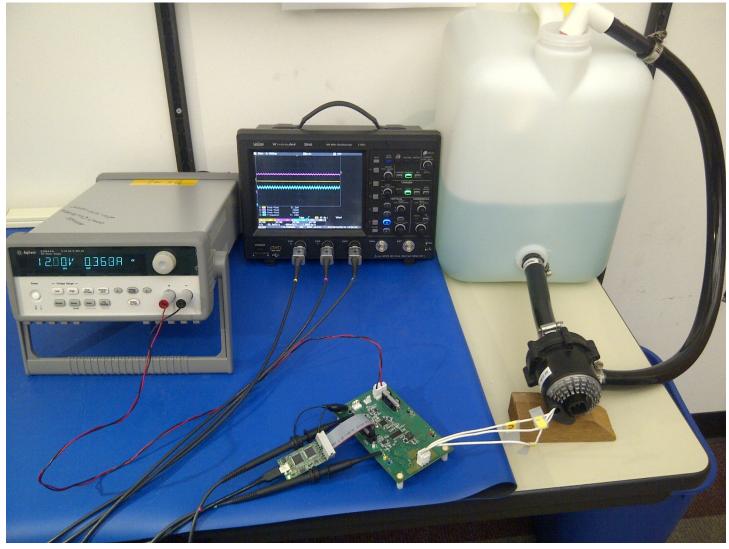


Figure 13 Test set-up with Cooper Standard 50W water pump

# 4.1 Instaspin to identify motor parameters

Cooper Standard Pump electrical parameters

Stator Inductance 0.69 mH

### 4.2 Run at 1000 rpm with nominal supply, no load, record currents, voltages

Note: due to the construction of typical water pump motors, it is not recommended to run the motor for long periods of time without water or another fluid flowing through the pump.

# 4.3 Vary speed (positive only) and measure current

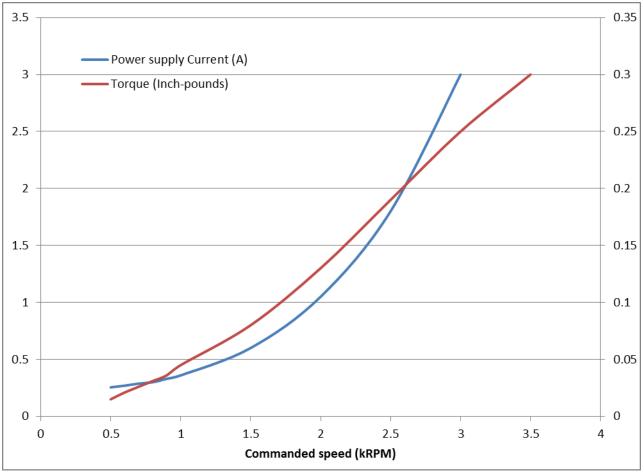


Figure 14 Power and torque as a function of motor speed



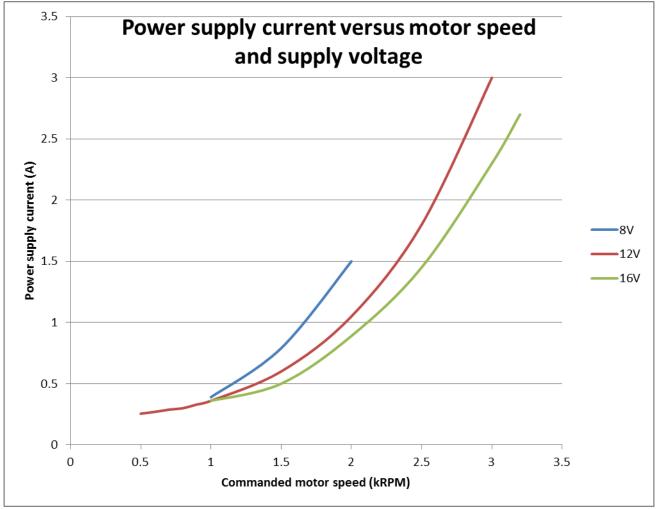


Figure 15 Power supply current versus supply voltage and motor speed

# 4.5 Measure flow rate (flow meter or bucket method)

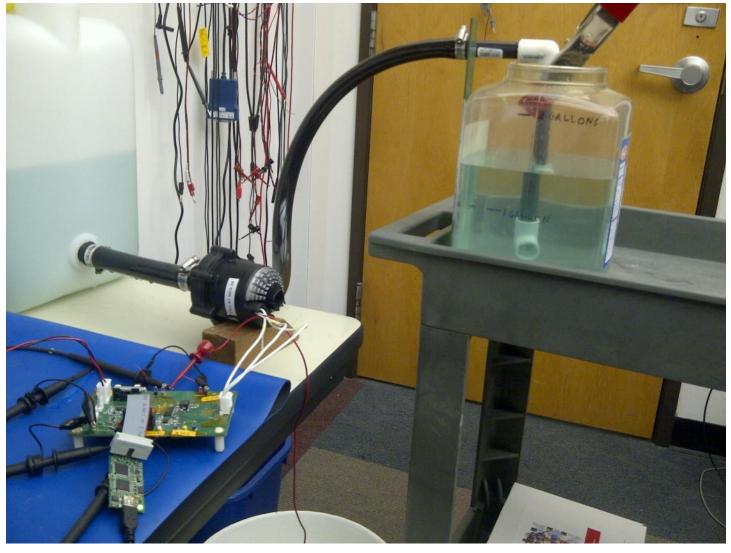


Figure 16 Flow rate measurement set-up

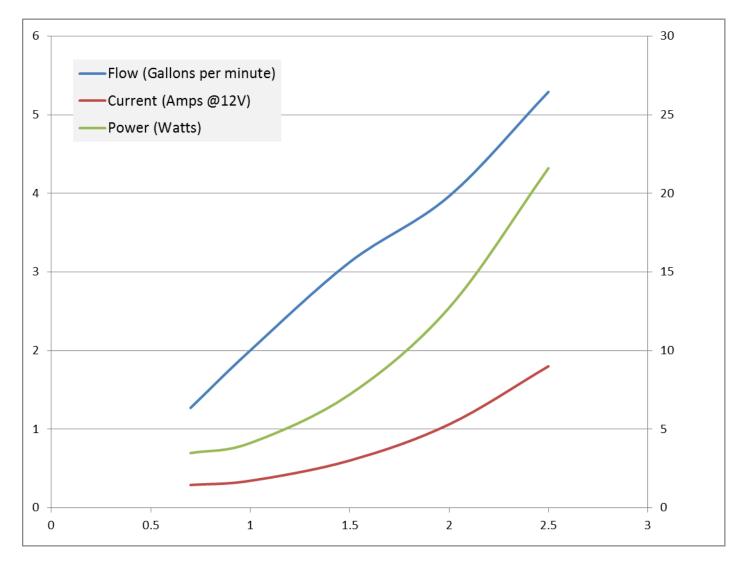


Figure 17 Flow rate measurements with Cooper Standard 50W water pump

# 5 Additional test data

# 5.1 Idle current with no dynamic motor load – table of current versus supply voltage

Conditions	Supply current with 12Vdc supply
Motor disabled, microcontroller	26 mA
not running	201111
Motor disabled, microcontroller program	41 mA
started, Flag_enableSys = 0	
Motor disabled, microcontroller program	42 mA
started, Flag_enableSys = 1	
Motor @ 0 RPM, microcontroller	235 mA
running, Flag_enableSys = 1	

## 5.2 Operational supply range

The DRV8301 buck converter correctly indicates POWER\_GOOD when the input power on J1 is above 5.9V. When the input power on J1 is less than 4.7V, the DRV8301 will discontinue generating a 3.3V supply.

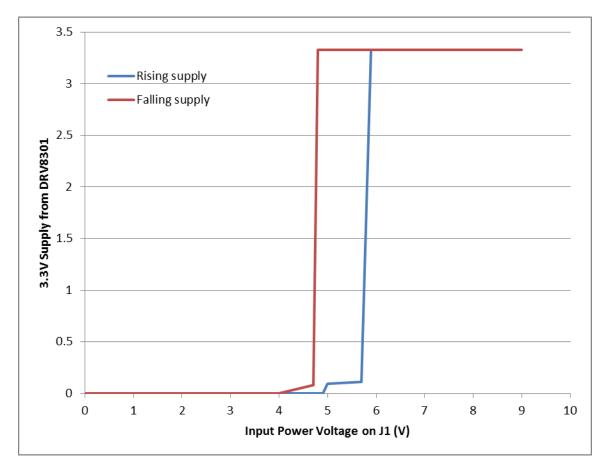
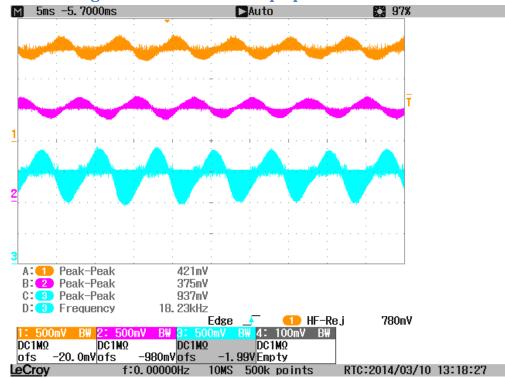


Figure 18 Operational range is indicated by 3.3V power supply versus input supply



### 5.3 Current sense voltages at load – oscilloscope plots

Figure 19 Motor current sense signals showing sinusoidal envelope

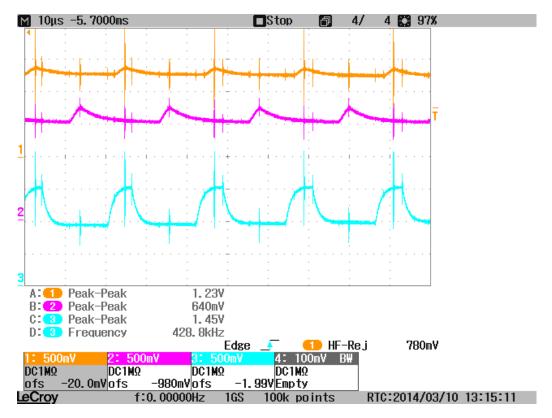
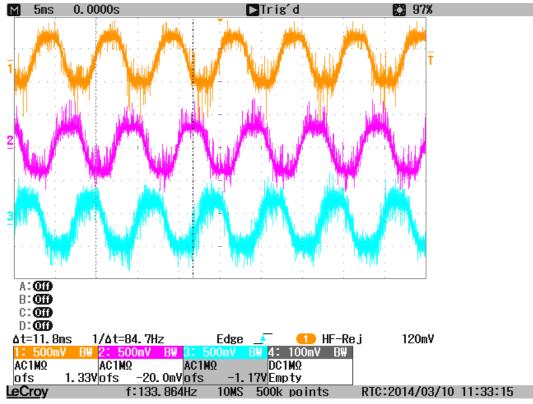


Figure 20 Detail of motor current sense signals



# 5.4 Phase voltages at load – oscilloscope plots

Figure 21 Three-phase motor voltages after filtering and scaling

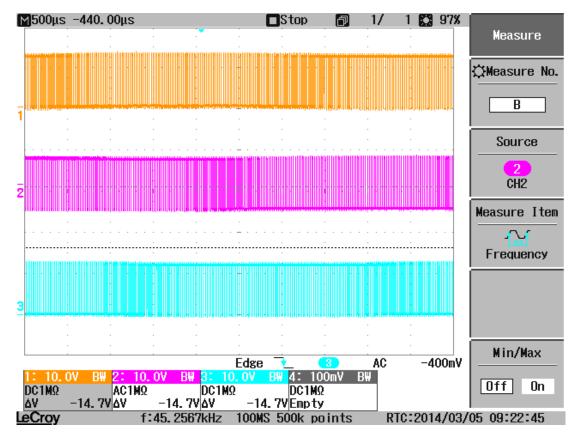


Figure 22 Three motor phase voltage signals - direct to motor windings

# 5.5 Temperature profile with load (top view) – infrared camera

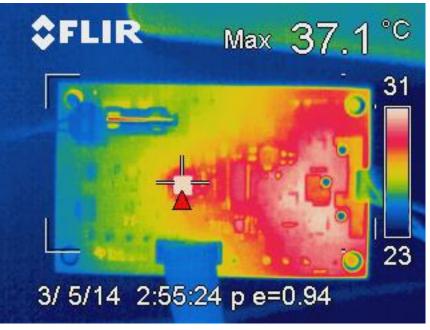


Figure 23 Infrared photo of SAT0042 E4 board in operation (top side)

### IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES "AS IS" AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI's products are provided subject to TI's Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI's provision of these resources does not expand or otherwise alter TI's applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265 Copyright © 2021, Texas Instruments Incorporated