TI 设计：TIDA-01496
具有速度调节功能的单层 12V 可编程三相 BLDC 电机驱动器参考设计

说明

此参考设计是使用 DRV10983-Q1 电机驱动器和 MSP430G2553 微控制器 (MCU) 的无传感器 BLDC 电机正弦驱动器。MCU 仅用于速度控制，而 DRV 器件是具有集成式 FETS 的主电机驱动器，用于驱动电机。此设计专门用于小型电机模块，尤其是风扇。此设计可实现专有无传感器控制，并可对电机参数进行调优，从而优化最终应用的性能。

资源

TIDA-01496	设计文件夹
DRV10983-Q1	产品文件夹
MSP430G2553-Q1	产品文件夹
DRV10xx 软件	软件下载

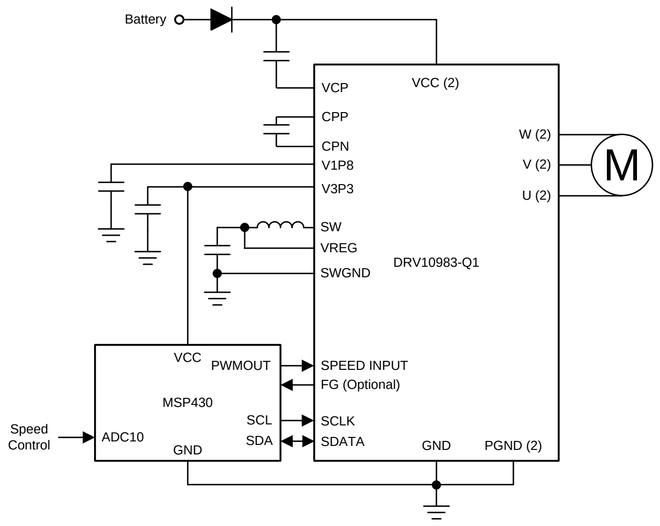
TI E2E™ Community 咨询我们的 E2E™ 专家

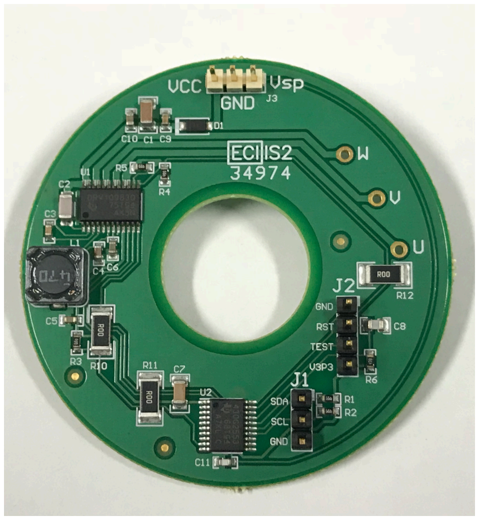
特性

- DRV10983-Q1 支持的具有高达 45V 负载突降功能、三相无刷直流 (BLDC) 电机驱动器的 12V 运算 VBAT 解决方案
- 单层设计降低制造成本
- DRV10983-Q1 采用专有的无传感器控制方案提供连续正弦驱动，显著降低了纯音效果
- MSP430G2553 MCU 可处理速度输入、控制环路并对电机参数进行编程
- 使用具有集成功率 MOSFET 的 DRV10983-Q1 三相无传感器电机驱动器来提供高达 2A （3A 峰值）的连续驱动电流
- DRV10983-Q1 提供的输出转换率和频率配置可增加 EMC 性能调优灵活性
- 集成降压稳压器可高效地将电源电压降至 5.0V 或 3.3V，从而为内外部电路供电

应用

- 全功能座椅
- 加热/冷却座椅





1 System Description

This reference design is a cost-effective, small-form-factor (SFF), single-layer, three-phase sinusoidal motor drive for brushless DC (BLDC) motors. The board accepts 12 V at input and provides the three motor phase outputs to drive the BLDC motor sinusoidally. The board consists of the DRV10983-Q1 motor driver and MSP430G2553 MCU. The DRV10983-Q1 delivers 3.3 V to the MCU through its V3P3 LDO regulator.

The board also has a speed pin input, which is connected to the ADC10 of the MSP430™ MCU. The ADC10 takes an input voltage from 0 to 3.3 V and converts it to a digital 10-bit value. This value controls the percentage duty cycle for the output PWM signal. The MSP430 MCU creates a pulse width modulation (PWM) signal output, which connects to the DRV10983-Q1 SPEED input pin. The DRV10983-Q1 then uses this input to control the speed of the motor.

The SPEED pin in the DRV10983-Q1 accepts either an analog or PWM input. The DRV10983-Q1 device provides motor speed measurement information on the frequency generator (FG) output or I^2C, which can be used to implement the speed control loop. The I^2C interface is also available as a header where an external graphical user interface (GUI) can be used for programming the DRV10983-Q1 device. For this reference design, the GUI can be used to configure and tune the motor parameters.

The MSP430G2553 MCU uses the speed measurement feedback to implement a speed control loop. The MSP430G2553 also programs the motor parameter registers into the DRV10983-Q1 without the need of a GUI. The DRV10983-Q1 device is specifically designed for cost-sensitive, low-noise, low-external-component count, small-motor applications. This reference design shows the application of an automotive seat blower fan. The DRV10983-Q1 device uses a proprietary sensorless control scheme to provide continuous sinusoidal drive, which significantly reduces the pure tone acoustics that typically occur as a result of commutation.

1.1 Key System Specifications

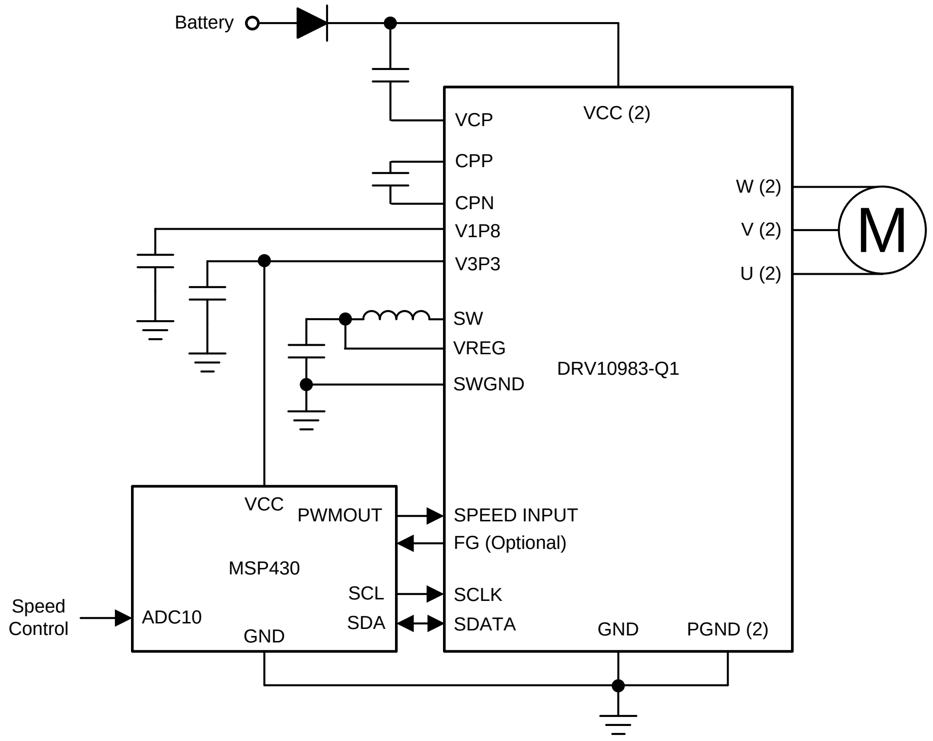
表 1. Key System Specifications

PARAMETER	SPECIFICATIONS
DC input voltage	8 to 16 V
Current	1.0 A continuous
Power level	Thermal design for 8- to 16-W operation
Control method	Integrated 180° sinusoidal control
Analog speed input	0 to 3.3 V
Protection circuits	Overcurrent, lock detection, voltage surge protection, UVLO protection, thermal shutdown
Operating ambient	–40°C to +125°C

2 System Overview

2.1 Block Diagram

图 1 shows the system block diagram for this reference design. The system is supplied with a motor supply voltage of 12 V, which goes to the DRV10983-Q1 device. The DRV10983-Q1 driver's P3V3 LDO regulator supplies 3.3 V to power the MSP430 MCU. Speed control voltage ranging from 0 to 3.3 V is input to the MSP430 MCU, which drives the PWM duty cycle into the DRV10983-Q1, which drives the speed of the motor. I^{2}C connections are used to program the registers onto the DRV10983-Q1 as well as provide speed feedback data for the speed control loop.



Copyright © 2017, Texas Instruments Incorporated

图 1. System Block Diagram

The DRV10983 driver can be configured through I^{2}C using the external GUI with the header available onboard. The user can also implement the MSP430 MCU using a JTAG interface if additional features such as speed loop are required.

2.2 Highlighted Products

2.2.1 DRV10983-Q1

The DRV10983-Q1 device is a three-phase sensorless motor driver with integrated power MOSFETs, which can provide continuous drive current up to 2 A. The device is specifically designed for cost-sensitive, low-noise, low-external-component-count fan and pump applications.

The DRV10983-Q1 device preserves register setting down to 4.5 V and delivers current to the motor with supply voltage as low as 6.2 V. If the power supply voltage is higher than 28 V, the device stops driving the motor and protects the DRV10983-Q1 circuitry. This function is able to handle a load dump condition up to 45 V.

表 2 highlights the different versions of the DRV10983-Q1: sleep version and standby version. The DRV10983-Q1 enters either sleep or standby to conserve energy. For more information, see *DRV10983-Q1 Automotive, Three-Phase, Sensorless BLDC Motor Driver*.

表 2. Device Options for DRV10983-Q1

DEVICE OPTION	DESCRIPTION	PACKAGE	BODY SIZE (NOM)
DRV10983Q	Sleep version	HTSSOP (24)	7.80 × 6.40 mm
DRV10983SQ	Standby version	HTSSOP (24)	7.80 × 6.40 mm

The DRV10983-Q1 device uses a proprietary sensorless control scheme to provide continuous sinusoidal drive, which significantly reduces the pure tone acoustics that typically occur as a result of commutation. The interface to the device is designed to be simple and flexible. The motor can be controlled directly through PWM, analog, or I^2C inputs. Motor speed feedback is available through both the FG pin and the I^2C interface simultaneously.

The DRV10983-Q1 device features an integrated buck regulator to step down the supply voltage efficiently to 5 V for powering both internal and external circuits. The 3.3-V LDO also may be used to provide power for external circuits. The device is available in either a sleep mode or a standby mode version to conserve power when the motor is not running. The standby mode (8.5 mA) version (DRV10983SQ) leaves the regulator running and the sleep mode (48 µA) version (DRV10983Q) shuts off the regulator. Use the standby mode version in applications where the regulator is used to power an external microcontroller. Throughout this datasheet, the DRV10983-Q1 is used for both devices [DRV10983Q (sleep version) and DRV10983SQ (standby version)], except for specific discussions of sleep versus standby functionality.

An I^2C interface allows the user to reprogram specific motor parameters in registers and to program the EEPROM to help optimize the performance for a given application. The DRV10983-Q1 device is available in a thermally-efficient HTSSOP, 24-pin package with an exposed thermal pad. The operating ambient temperature is specified from –40°C to +125°C.

2.2.2 MSP430G2553

The Texas Instruments MSP430 family of ultra-low-power microcontrollers consists of several devices featuring different sets of peripherals targeted for various applications. The architecture, combined with five low-power modes, is optimized to achieve extended battery life in portable measurement applications. The device features a powerful 16-bit RISC CPU, 16-bit registers, and constant generators that contribute to maximum code efficiency. The digitally controlled oscillator (DCO) allows wake-up from low-power modes to active mode in less than 1 µs.

The MSP430G2x13 and MSP430G2x53 series are ultra-low-power mixed signal microcontrollers with built-in 16-bit timers, up to 24 I/O capacitive-touch enabled pins, a versatile analog comparator, and built-in communication capability using the universal serial communication interface. In addition, the MSP430G2x53 family members have a 10-bit analog-to-digital (A/D) converter.

Typical applications include low-cost sensor systems that capture analog signals, convert them to digital values, and then process the data for display or for transmission to a host system.

2.3 Design Considerations

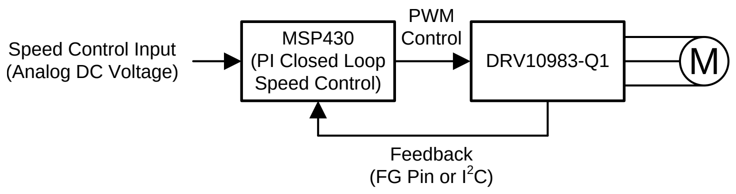
表 3 shows the recommended components for the DRV10983-Q1 device to function. In addition to these components, a Schottky diode is added at VCC for reverse voltage protection as well as an addition capacitor in parallel with C_{VCC}. JTAG pin headers are used for MSP430 programming and I^2C headers are used for GUI connection and EEPROM register programming. For compact board layouts, it is sufficient to use a smaller inductor for the buck regulator to minimize board footprint. Additional layout guidelines are found in *DRV10983-Q1 Automotive, Three-Phase, Sensorless BLDC Motor Driver*.

表 3. External Components

COMPONENT	PIN 1	PIN 2	RECOMMENDED
C1	VCC	GND	10-µF ceramic capacitor rated for VCC
C2	VCP	VCC	0.1-µF ceramic capacitor rated for 10 V
C3	CPP	CPN	10-nF ceramic capacitor rated for VCC × 2
C4	VREG	GND	10-µF ceramic capacitor rated for 10 V
C5	V3P3	GND	1-µF ceramic capacitor rated for 5 V
C6	V1P8	GND	1-µF ceramic capacitor rated for 5 V
C7	DVCC	GND	0.1-µF ceramic capacitor
C8	RST	GND	2200-pF ceramic capacitor
C10	VCC	GND	(DNP) 10-µF ceramic capacitor rated for VCC
C11	P1.1	GND	470-pF ceramic capacitor
C9	VCC	GND	0.1-µF ceramic capacitor rated for VCC
D1	VSource	VCC	Schottky diode rated for 30 V, 2 A
L1	SW	VREG	47-µH ferrite rated for 1.15 A (inductive mode)
R1	FG	V3P3	4.75-kΩ pullup to V3P3
R2	SDA	V3P3	4.75-kΩ pullup to V3P3
R3	SCL	V3P3	4.75-kΩ pullup to V3P3
R4	SPEED	GND	4.75-kΩ pulldown
R5	DIR	GND	4.75-kΩ pulldown
R6	DVCC	RST	47 kΩ
R10	FG	FG	Jumper

2.4 System Design Theory

2.4.1 PI Speed Loop Implementation



Copyright © 2017, Texas Instruments Incorporated

图 2. High-Level System Block Diagram With Speed Loop

The system design includes two main device components, the MSP430G2553 and DRV10983-Q1. The MSP430 MCU takes in a voltage input through its ADC10 and converts it to a 10-bit number. This number determines the percentage duty cycle of the PWM output to the DRV10983-Q1. The MSP430G2553 also programs the motor parameters into the DRV10983-Q1 EEPROM registers through I^{2}C. The DRV10983-Q1 drives the motor and at the same time send motor speed feedback information through I^{2}C or FG pin. This feedback goes back to the MSP430 MCU, which processes the proportional integrator (PI) speed control loop.

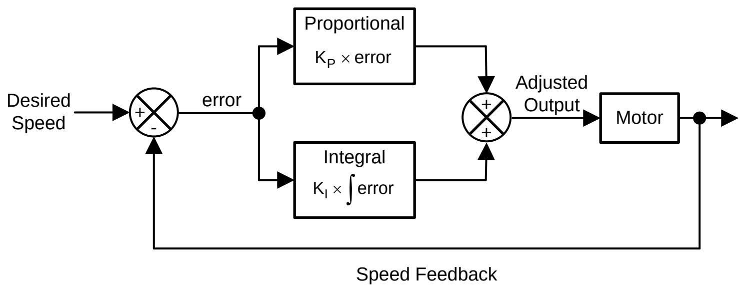


图 3. PI Loop Implementation

A PI controller is made up of two separate controller terms, the proportional term and the integral term. The system takes in the desired input and compares it to the current feedback from the output system. The difference between these two values is the error term. This calculated error term is the input to the controller. The proportional term is calculated by taking the proportion gain, K_P, and multiplying it by the magnitude of the error term. The integral of the error is also calculated and multiplied by the integral gain, K_I. These two values are then added to each other and output the new speed command to the motor. The motor takes this command and turn while at the same time providing feedback to the PI controller. The gains of the proportional and integral terms determine the overshoot and settling time.

3 Hardware, Software, Testing Requirements, and Test Results

3.1 Required Hardware and Software

3.1.1 Hardware

This reference design is powered through the V_{BAT} input and controlled using the V_{SP} input. Specifically, V_{SP} refers to the analog voltage used by the MCU that generates the PWM signals necessary to control the speed of the motor. The vias marked U, V, and W refer to the phase winding inputs of a three-phase BLDC motor. J1 is used for I^2C communication with the motor driver and MCU while J2 is used for the JTAG interface with the MCU. While the DRV10983-Q1 can drive the motor without the use of an MCU, the design uses the MCU for implementing the speed loop functionality and providing the PWM speed signal for motor drive.

To set up the system, connect a power supply capable of providing 12 to 28 V and 3 A using V_{BAT}. Note to turn on the power supply only when all connections are finalized. Then, connect another voltage source capable of providing 3.3 V using V_{SP}. It is recommended to use a USB2ANY and the DRV10x Software GUI when communicating with the DRV10983-Q1 using I^2C. For pins 1 through 3 of the J1, see the SDA, SCL, and GND connections needed for I^2C communication. See 图 4 for more information when connecting to the system through I^2C.

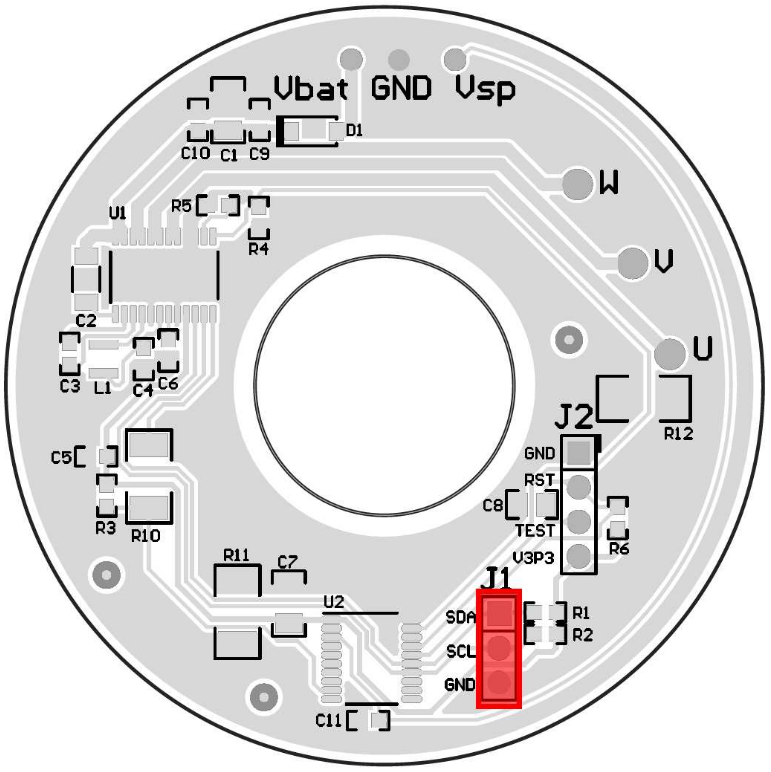


图 4. Hardware Setup for I^2C Communication

It is also recommended to use a JTAG emulator like the one found through the MSP430 LaunchPad™ Value Line Development Kit (MSP-EXP430G2) to interact with the MSP430G2553. See 图 5 for more information when connecting to the system using the JTAG interface. For pins 1 through 4 of J2, see the GND, RESET, TEST, and V3P3 required for the JTAG interface. For more information about I^{2}C and the JTAG interface, see 节 3.1.2.

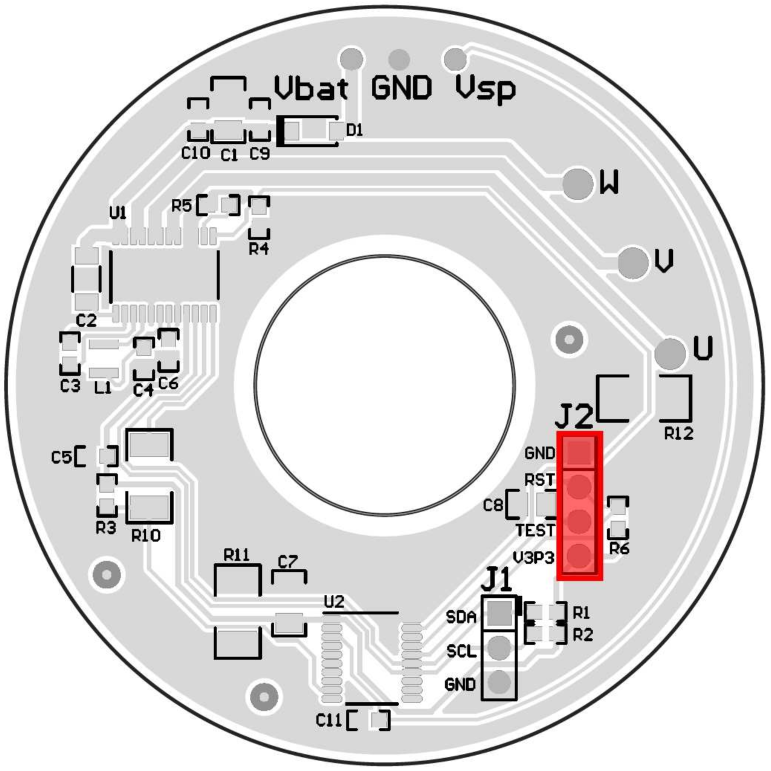


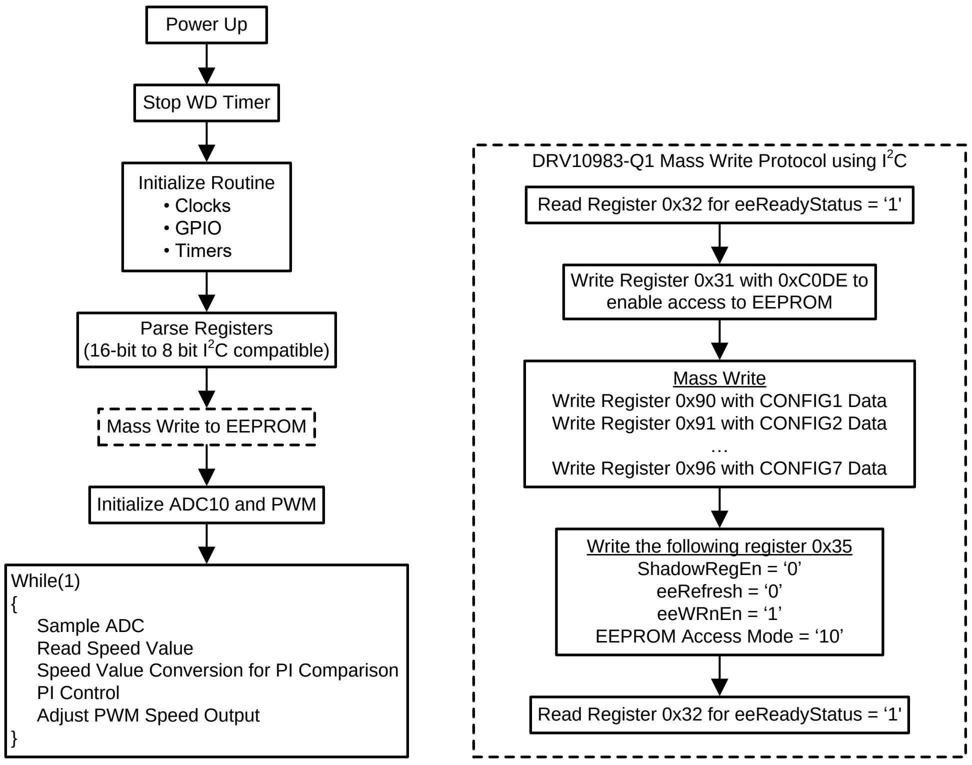
图 5. Hardware Setup for JTAG Interface and Communication With MSP430 MCU

Once the MCU has been programmed and the DRV10983-Q1's EEPROM settings have been tuned, the user only needs to power on the system and control the speed voltage to operate the design.

3.1.2 Software

The firmware is written for the onboard MSP430 controller. The existing firmware serves two major functionalities:

- The firmware can configure the DRV10983-Q1 device. This firmware is usually a one-time configuration unless the user wants to modify parameters runtime of the configuration.
- The firmware processes the input speed using the ADC10 of the MSP430 MCU and implements a PI speed control system using feedback from the DRV10983-Q1 from either I^{2}C or FG.



Power Up
Stop WD Timer
Initialize Routine
• Clocks
• GPIO
• Timers
Parse Registers (16-bit to 8 bit I2C compatible)
Mass Write to EEPROM
Initialize ADC10 and PWM
While(1)
{
Sample ADC
Read Speed Value
Speed Value Conversion for PI Comparison
PI Control
Adjust PWM Speed Output
}
DRV10983-Q1 Mass Write Protocol using I2C
Read Register 0x32 for eeReadyStatus = '1'
Write Register 0x31 with 0xC0DE to enable access to EEPROM
Mass Write
Write Register 0x90 with CONFIG1 Data
Write Register 0x91 with CONFIG2 Data
…
Write Register 0x96 with CONFIG7 Data
Write the following register 0x35
ShadowRegEn = '0'
eeRefresh = '0'
eeWRnEn = '1'
EEPROM Access Mode = '10'
Read Register 0x32 for eeReadyStatus = '1'

Copyright © 2017, Texas Instruments Incorporated

图 6. Software System Block Diagram

3.1.2.1 Using Software to Configure DRV10983-Q1

Configuring the DRV10983-Q1 refers to programming the tuning parameters, which are written to the EEPROM. These settings contain intrinsic motor parameters unique to the motor, such as Kt and Phase Resistance, and tuning parameters just as the acceleration coefficients and open to closed loop threshold. Consult the *DRV10983-Q1 Tuning Guide* for more information.

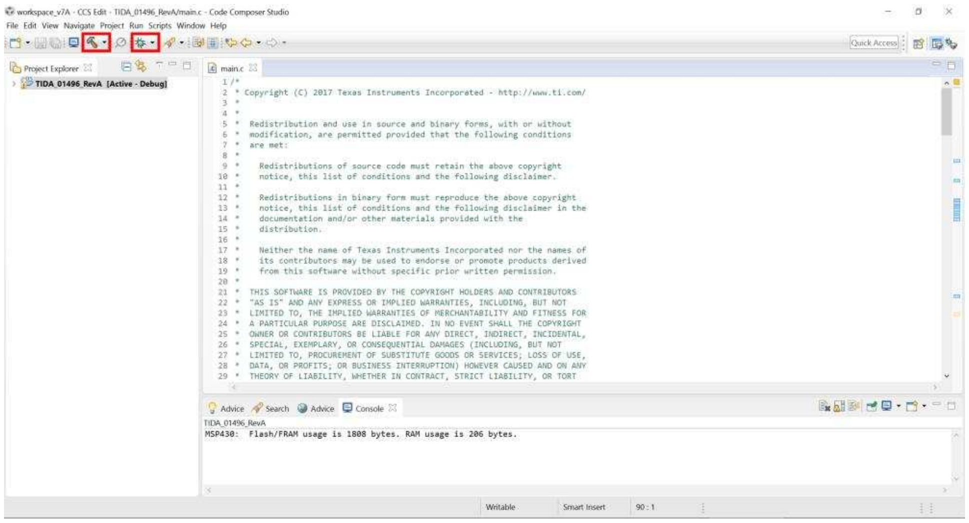
Ideally, the motor parameters can be obtained by using I^{2}C to communicate with the motor driver. Download the *DRV10xx Software GUI*. [(1)] to easily implement EEPROM settings and monitor motor faults during operation. A guide for installing the GUI software is found in "Appendix A GUI Installation and Overview" of the *DRV10983-Q1 Evaluation Module User's Guide*.

When interfacing with the MSP430G2553, download the latest version of *Code Composer Studio™ (CCS)* to easily implement the provided code. By simply importing the project or copying and pasting the code, the MCU can be programmed to implement EEPROM settings, the speed loop, and the PWM signals.

It is highly recommended to make all the connections first before turning on the voltage supply. Once the connections are made, the voltage supply can be turned on. A common error can occur during programming: "MSP430: Error initializing emulator: Could not find MSP-FET430UIF on specified COM port". Usually this error can be remedied by turning off the power supply, disconnecting the LaunchPad from the computer, reconnecting the LaunchPad first, and then turning on the power supply again.

Follow these instructions to program the MSP430 MCU:

1. Import the "TIDA-01496-ExampleSoftware" project using TI's CCS software.
2. Connect the LaunchPad programmer to the TIDA-01496 board, as shown in 图 5.
3. Build the project by clicking the Build button (hammer icon), which if run successfully, appears as shown in 图 7.
4. Click the Debug button (bug icon) as 图 7 shows.


workspace_v7A - CCS Edit - TIDA_01496_RevA/main.c - Code Composer Studio
File Edit View Navigate Project Run Scripts Window Help
Quick Access
Project Explorer
TIDA_01496_RevA [Active - Debug]
main.c
Advice
Search
Advice
Console
Writable
Smart Insert
90 : 1

```
/*
 * Copyright (C) 2017 Texas Instruments Incorporated - http://www.ti.com/
 *
 *
 *  Redistribution and use in source and binary forms, with or without
 *  modification, are permitted provided that the following conditions
 *  are met:
 *
 *    Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 *    Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the
 *    distribution.
 *
 *    Neither the name of Texas Instruments Incorporated nor the names of
 *    its contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 *  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 *  "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 *  LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
 *  A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 *  OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 *  SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
 *  LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
 *  DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
 *  THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
```

```
TIDA_01496_RevA
MSP430:  Flash/FRAM usage is 1808 bytes. RAM usage is 206 bytes.
```

图 7. Building and Debugging Project in CCS

(1) This requires a TI.com login

Follow these instructions to program the motor parameter registers into the MSP430 software:

1. Find the optimum motor parameter settings using the GUI. For help tuning the motor, see the *DRV10983-Q1 Tuning Guide*.
2. Save the motor configurations using the GUI. This action saves the seven CONFIG registers as a .csv file as shown in 图 8.

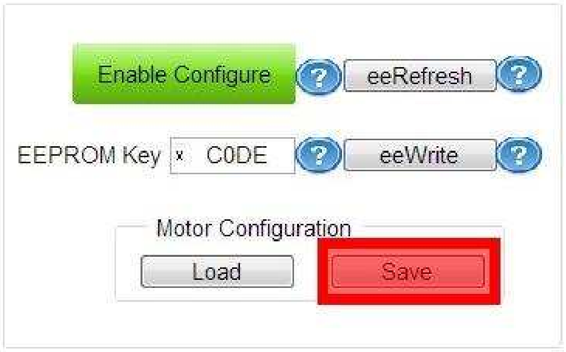
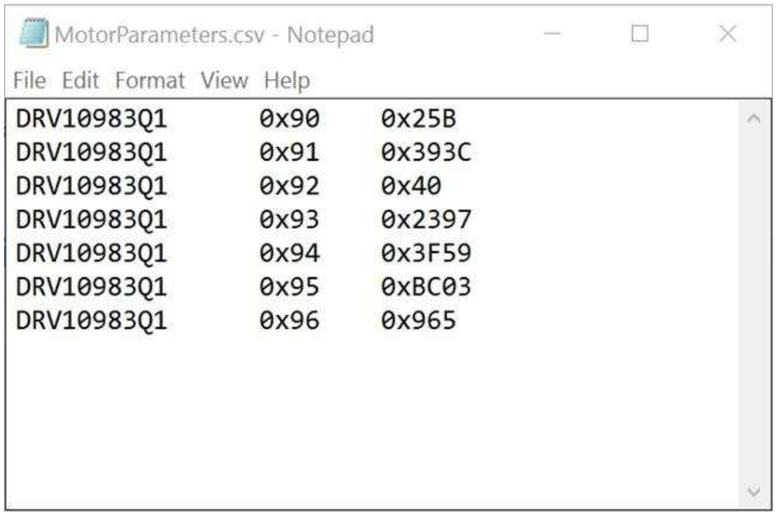


图 8. How to Save a Motor Configuration File in DRV10x GUI

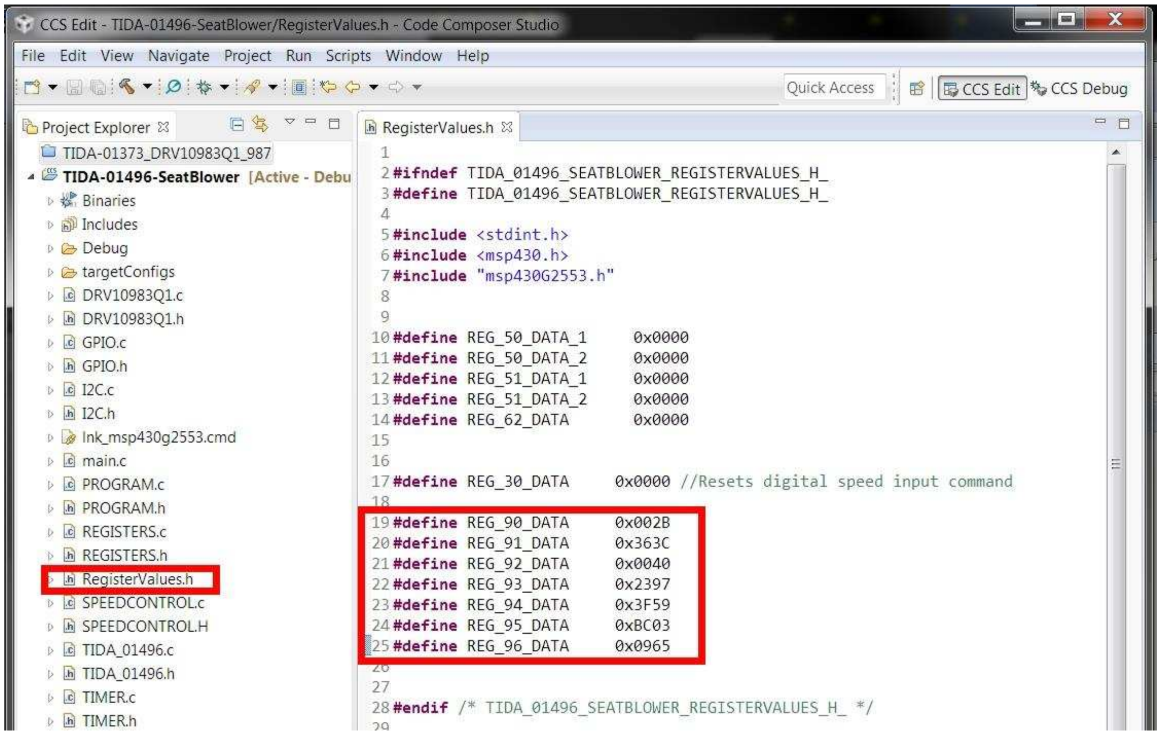
3. Open the .csv file with Notepad. These registers are now implemented in the reference design CCS as shown in 图 9.



```
DRV10983Q1      0x90    0x25B
DRV10983Q1      0x91    0x393C
DRV10983Q1      0x92    0x40
DRV10983Q1      0x93    0x2397
DRV10983Q1      0x94    0x3F59
DRV10983Q1      0x95    0xBC03
DRV10983Q1      0x96    0x965
```

图 9. Example Motor Configuration File in .csv Format

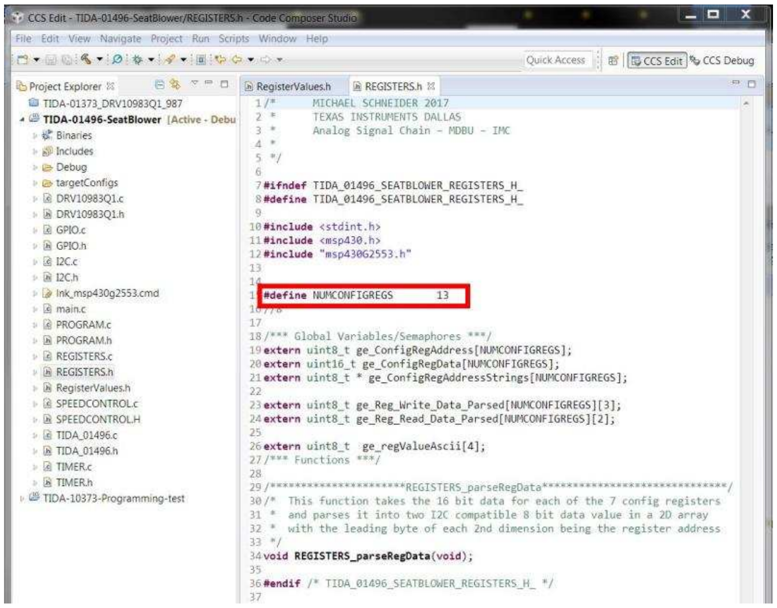
4. Open the RegisterValues.h header file and place the CONFIG values in their corresponding registers as shown in 图 10.



```

#ifndef TIDA_01496_SEATBLOWER_REGISTERVALUES_H_
#define TIDA_01496_SEATBLOWER_REGISTERVALUES_H_

#include <stdint.h>
#include <msp430.h>
#include "msp430G2553.h"


#define REG_50_DATA_1     0x0000
#define REG_50_DATA_2     0x0000
#define REG_51_DATA_1     0x0000
#define REG_51_DATA_2     0x0000
#define REG_62_DATA       0x0000


#define REG_30_DATA     0x0000 //Resets digital speed input command

#define REG_90_DATA     0x002B
#define REG_91_DATA     0x363C
#define REG_92_DATA     0x0040
#define REG_93_DATA     0x2397
#define REG_94_DATA     0x3F59
#define REG_95_DATA     0xBC03
#define REG_96_DATA     0x0965


#endif /* TIDA_01496_SEATBLOWER_REGISTERVALUES_H_ */
```

图 10. Adding Motor Configuration Settings into RegisterValues.h

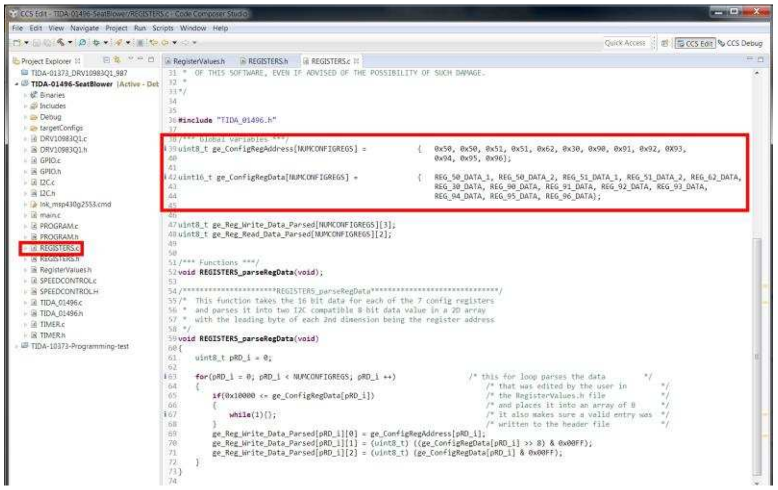
5. Save the project and rebuild.
6. Optional: Program other registers the seven CONFIG registers if needed:
   a. Define the new register value in the RegisterValues.h header file.
   b. Define the number of total registers in the REGISTERS.h header file as shown in 图 11.



```
/*       MICHAEL SCHNEIDER 2017
 *       TEXAS INSTRUMENTS DALLAS
 *       Analog Signal Chain - MDBU - IMC
 *
 */

#ifndef TIDA_01496_SEATBLOWER_REGISTERS_H_
#define TIDA_01496_SEATBLOWER_REGISTERS_H_

#include <stdint.h>
#include <msp430.h>
#include "msp430G2553.h"


#define NUMCONFIGREGS       13
//8

/*** Global Variables/Semaphores ***/
extern uint8_t ge_ConfigRegAddress[NUMCONFIGREGS];
extern uint16_t ge_ConfigRegData[NUMCONFIGREGS];
extern uint8_t * ge_ConfigRegAddressStrings[NUMCONFIGREGS];

extern uint8_t ge_Reg_Write_Data_Parsed[NUMCONFIGREGS][3];
extern uint8_t ge_Reg_Read_Data_Parsed[NUMCONFIGREGS][2];

extern uint8_t  ge_regValueAscii[4];
/*** Functions ***/

/*********************REGISTERS_parseRegData****************************/
/*  This function takes the 16 bit data for each of the 7 config registers
 *  and parses it into two I2C compatible 8 bit data value in a 2D array
 *  with the leading byte of each 2nd dimension being the register address
 */
void REGISTERS_parseRegData(void);

#endif /* TIDA_01496_SEATBLOWER_REGISTERS_H_ */
```

图 11. How to Change Total Number of Registers Used in Project

c. Define the register address in the REGISTERS.c source file in both corresponding arrays as shown in 图 12.



```
/*  OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 */


#include "TIDA_01496.h"

/*** Global Variables ***/
uint8_t ge_ConfigRegAddress[NUMCONFIGREGS] =            {   0x50, 0x50, 0x51, 0x51, 0x62, 0x30, 0x90, 0x91, 0x92, 0X93,
                                                            0x94, 0x95, 0x96};

uint16_t ge_ConfigRegData[NUMCONFIGREGS] =              {   REG_50_DATA_1, REG_50_DATA_2, REG_51_DATA_1, REG_51_DATA_2, REG_62_DATA,
                                                            REG_30_DATA, REG_90_DATA, REG_91_DATA, REG_92_DATA, REG_93_DATA,
                                                            REG_94_DATA, REG_95_DATA, REG_96_DATA};


uint8_t ge_Reg_Write_Data_Parsed[NUMCONFIGREGS][3];
uint8_t ge_Reg_Read_Data_Parsed[NUMCONFIGREGS][2];


/*** Functions ***/
void REGISTERS_parseRegData(void);

/**********************REGISTERS_parseRegData*****************************/
/*  This function takes the 16 bit data for each of the 7 config registers
 *  and parses it into two I2C compatible 8 bit data value in a 2D array
 *  with the leading byte of each 2nd dimension being the register address
 */
void REGISTERS_parseRegData(void)
{
    uint8_t pRD_i = 0;

    for(pRD_i = 0; pRD_i < NUMCONFIGREGS; pRD_i ++)            /* this for loop parses the data        */
    {                                                              /* that was edited by the user in       */
        if(0x10000 <= ge_ConfigRegData[pRD_i])                     /* the RegisterValues.h file            */
        {                                                          /* and places it into an array of 8     */
            while(1){};                                            /* it also makes sure a valid entry was */
        }                                                          /* written to the header file           */
        ge_Reg_Write_Data_Parsed[pRD_i][0] = ge_ConfigRegAddress[pRD_i];
        ge_Reg_Write_Data_Parsed[pRD_i][1] = (uint8_t) ((ge_ConfigRegData[pRD_i] >> 8) & 0x00FF);
        ge_Reg_Write_Data_Parsed[pRD_i][2] = (uint8_t) (ge_ConfigRegData[pRD_i] & 0x00FF);
    }
}
```

图 12. How to Change Array of Registers Used in Project

3.1.2.2 *Flow for Using ADC10 of MSP430 to Implement PI Speed Control System*

As shown in 图 6, after initialization, the program stays in the while loop continually running the SPEEDCONTROL function. The SPEEDCONTROL function flows by the following process:

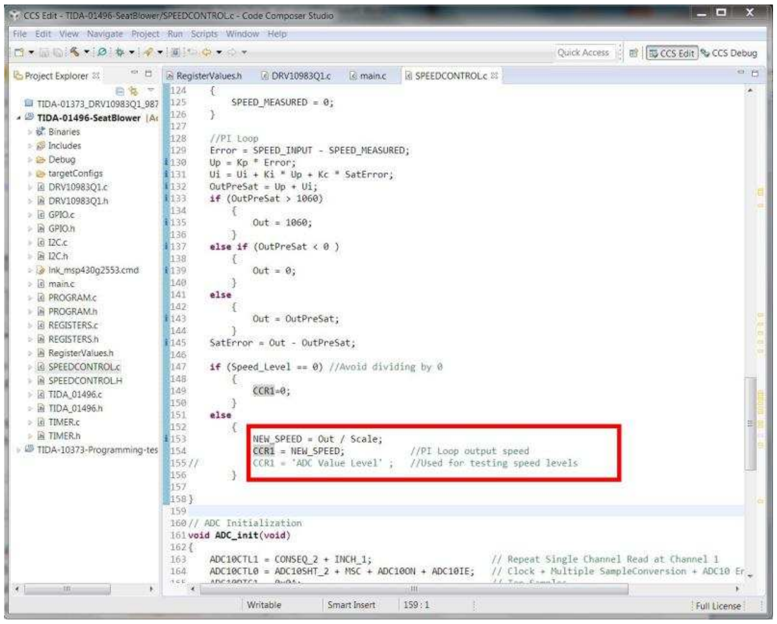
1. ADC samples voltage of MCU and translates into a bit value defined by *ADC_Value*.
2. Using this bit value, a speed level is assigned (that is, 0 to 5) defined by *Speed_Level*.
3. The speed level determines the target speed of the electrical cycle (Hz), defined by *SPEED_INPUT*, and scale value, defined by *Scale*.
4. The current speed of the electrical cycle speed is read through I^{2}C from the DRV10983-Q1 defined by *SPEED_MEASURED*.
5. The difference between the target speed and current speed is defined by *Error*.
6. The PI loop operation calculates the error and adjustments needed to reach the target speed defined by *OutPreSat*.
7. An if statement is used to bound the error and adjustments if the error is too large defined by *Out* (Hz).
8. The error for the next cycle used by the PI loop is determined by *SatError*.
9. The PWM signal uses the error and adjustments. This is defined by *CCR1* (Hz).

3.1.2.3 Tuning Speed Levels

Each motor has its own specific speed curve and parameters. This requires tuning for the speed levels and PI loop. Any speed levels can be set for the device depending on the application of the system. The easiest method is to run the firmware with one simple modification. In the SPEEDCONTROL.c file, change:

`CCR1 = NEW_SPEED;` to `CCR1 = ADC_Value;`

This change is shown in 图 13.



```
    {
        SPEED_MEASURED = 0;
    }

    //PI Loop
    Error = SPEED_INPUT - SPEED_MEASURED;
    Up = Kp * Error;
    Ui = Ui + Ki * Up + Kc * SatError;
    OutPreSat = Up + Ui;
    if (OutPreSat > 1060)
        {
            Out = 1060;
        }
    else if (OutPreSat < 0 )
        {
            Out = 0;
        }
    else
        {
            Out = OutPreSat;
        }
    SatError = Out - OutPreSat;

    if (Speed_Level == 0) //Avoid dividing by 0
        {
            CCR1=0;
        }
    else
        {
            NEW_SPEED = Out / Scale;
            CCR1 = NEW_SPEED;           //PI Loop output speed
//          CCR1 = 'ADC Value Level' ;  //Used for testing speed levels
        }

}

// ADC Initialization
void ADC_init(void)
{
    ADC10CTL1 = CONSEQ_2 + INCH_1;                    // Repeat Single Channel Read at Channel 1
    ADC10CTL0 = ADC10SHT_2 + MSC + ADC10ON + ADC10IE;  // Clock + Multiple SampleConversion + ADC10 Er
```

图 13. Example of Changing SPEEDCONTROL.c File for Testing

The ADC_Value is the value read by the ADC10 ranging from 0 to 1023. There needs to be a one-to-one relationship between the ADC_Value and the PWM output, meaning an ADC_Value of 1023 is 100% PWM and an ADC_Value of 511 is 50% PWM. Under ideal conditions, choose the required speed levels and convert them to an ADC_Value. See 表 4 for an example:

表 4. Translating Speed Levels, PWM Duty Cycles, and ADC_Value

SPEED LEVEL	PWM DUTY	ADC_Value (1023×DUTY)
0	0%	0
1	20%	205
2	40%	409
3	60%	614
4	80%	818
5	100%	1023

Now set the ADC_Value ranges in the SPEEDCOMMAND.c file to correspond with these specified levels:

```
if (ADC_Value <= 50) {Speed_Level = 0;}
else if (ADC_Value > 50 && ADC_Value <=205)  {Speed_Level = 1;}
else if (ADC_Value >205 && ADC_Value <=409)  {Speed_Level = 2;}
else if (ADC_Value >409 && ADC_Value <=614)  {Speed_Level = 3;}
else if (ADC_Value >614 && ADC_Value <=818)  {Speed_Level = 4;}
```

```
    else if (ADC_Value >818 && ADC_Value <=1023) {Speed_Level = 5;}
    else {Speed_Level = 5;}
```

3.1.2.4 Finding Appropriate Scale for Each Speed Level

Usually, the ADC_Value input, and by extension the SPEED_INPUT, will be compared to the SPEED_MEASURED and put through the PI loop. Setting the CCR1 value will manually determine the PWM duty cycle and ignore the effects of the PI loop. This is important because unique motor parameters will cause different speeds at 100% duty cycle. Determining the max speed will determine the scale between the PWM duty cycle and the SPEED_INPUT.

To determine the scale, observe the PWM duty cycle and SPEED_MEASURED for each CCR1 value and determine the scale value by dividing the CCR1 value by the SPEED_MEASURED. This is implemented in 图 14 and listed in 表 5.

To manually set CCR1, implement the following code:

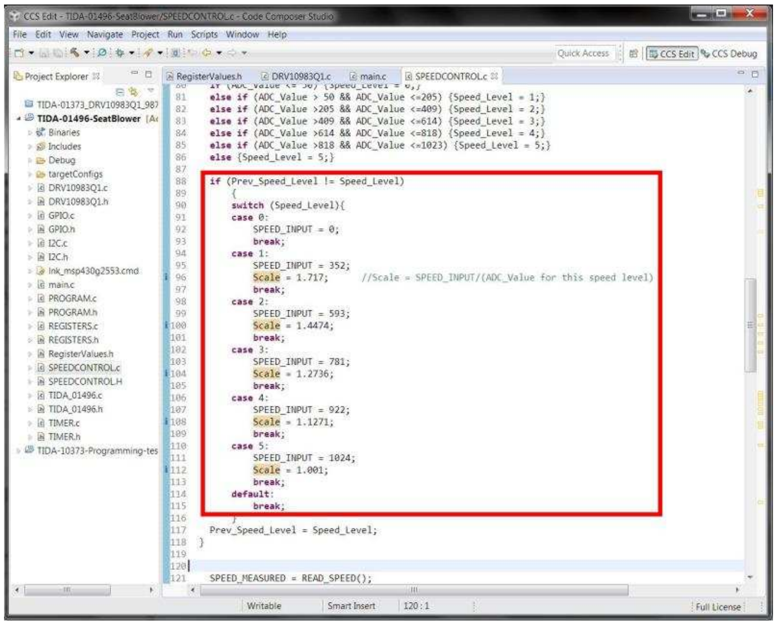
```
CCR1 = 1060;
```

Each time a speed is set, run the motor and check the SPEED_MEASURED variable. To do this, run the debugger and add SPEED_MEASURED to the expressions table by right clicking the table and selecting "add global variable". Pause the debugger while the motor is running in closed loop to observe the speed value measured by the DRV10983-Q1 using I^{2}C.

表 5. Example of Tuning SPEED_INPUT(1) Using Scale Functionality

SPEED LEVEL	EXAMPLE PWM DUTY	EXAMPLE CCR1 Value	EXAMPLE MEASURED (SPEED_MEASURED)	EXAMPLE SCALE (CCR1/SPEED_MEASURED)
0	0%	0	0	—
1	20%	205	31	6.61
2	40%	409	63	6.49
3	60%	614	94	6.53
4	80%	818	126	6.49
5	100%	1060	163	6.5

(1) SPEED_INPUT is motor specific



```
        else if (ADC_Value > 50 && ADC_Value <=205) {Speed_Level = 1;}
        else if (ADC_Value >205 && ADC_Value <=409) {Speed_Level = 2;}
        else if (ADC_Value >409 && ADC_Value <=614) {Speed_Level = 3;}
        else if (ADC_Value >614 && ADC_Value <=818) {Speed_Level = 4;}
        else if (ADC_Value >818 && ADC_Value <=1023) {Speed_Level = 5;}
        else {Speed_Level = 5;}

        if (Prev_Speed_Level != Speed_Level)
            {
            switch (Speed_Level){
            case 0:
                SPEED_INPUT = 0;
                break;
            case 1:
                SPEED_INPUT = 352;
                Scale = 1.717;       //Scale = SPEED_INPUT/(ADC_Value for this speed level)
                break;
            case 2:
                SPEED_INPUT = 593;
                Scale = 1.4474;
                break;
            case 3:
                SPEED_INPUT = 781;
                Scale = 1.2736;
                break;
            case 4:
                SPEED_INPUT = 922;
                Scale = 1.1271;
                break;
            case 5:
                SPEED_INPUT = 1024;
                Scale = 1.001;
                break;
            default:
                break;
            }
        Prev_Speed_Level = Speed_Level;
    }

        SPEED_MEASURED = READ_SPEED();
```

图 14. Example Implementation of Scaled SPEED_INPUT in CCS Project

These speed levels may or may not be linear. This case is for a non-linear relationship between the input command and output speed due to the motor’s inherent load properties. For a more linear motor, a constant scale can be set for any input without using speed levels and these calculations can be simplified.

3.1.2.5 Tuning the PI Loop

Tuning the PI loop consists of simply modifying the proportional and integral constants in the SPEEDCONTROL.c file.

The PI controller is implemented within a while loop, which runs continuously. First, the error is calculated between the desired speed and the feedback speed. The error is multiplied by the proportional gain and the output of the proportional term is stored in the $U_p$term. The integral output is calculated by adding together the previous integral output, the product of the integral gain and proportional output, and the product of the integral gain factor and the saturated error. The pre-saturated value is the sum of the integral and proportional outputs, U_i and U_p. The saturated error is calculated between the output after saturation and the pre-saturation output and applied to the integral output. A PI loop is an essential controller method when closing the speed loop within a motor. The loop allows motors to work across different voltages and different loads while maintaining the same desired speed input. To learn more about the PI loop theory, see 节 2.4.1.

Tuning a PI loop is required to make the system stable and responsive while minimizing overshoot. The last two objectives often conflict with each other, and a compromise must be met to achieve the best results. Tuning a PI loop for a system with a fast response such as for this motor application must have a low P-gain. Set the P-gain low and the I-gain to near zero and run the motor to see if oscillations occur. Once oscillations occur, halve the P-gain value. Next, set a very small integral value. Slowly increase and run the motor until oscillations occur more than half the integral value. 图 15 demonstrates the effects of increasing the proportional and integral terms on overshoot and setting time.

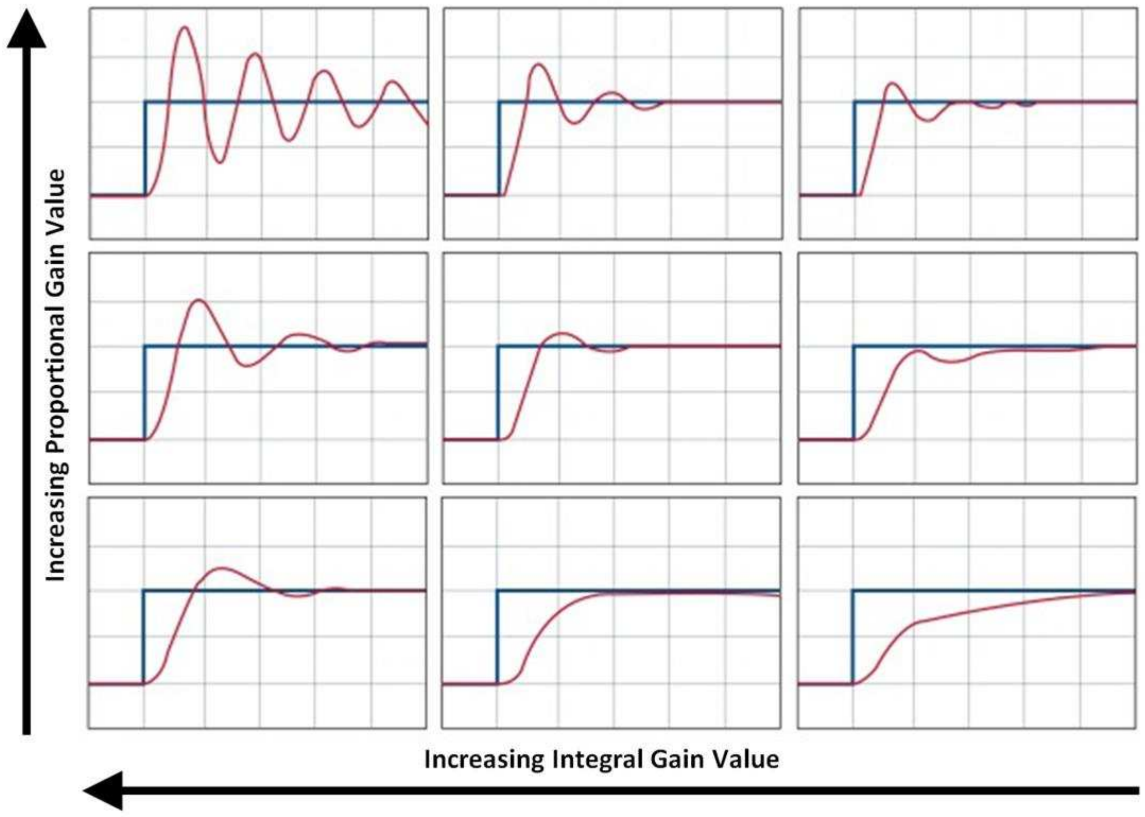


图 15. Effects of Tuning PI Controller Gains

3.2 Testing and Results

3.2.1 Test Setup

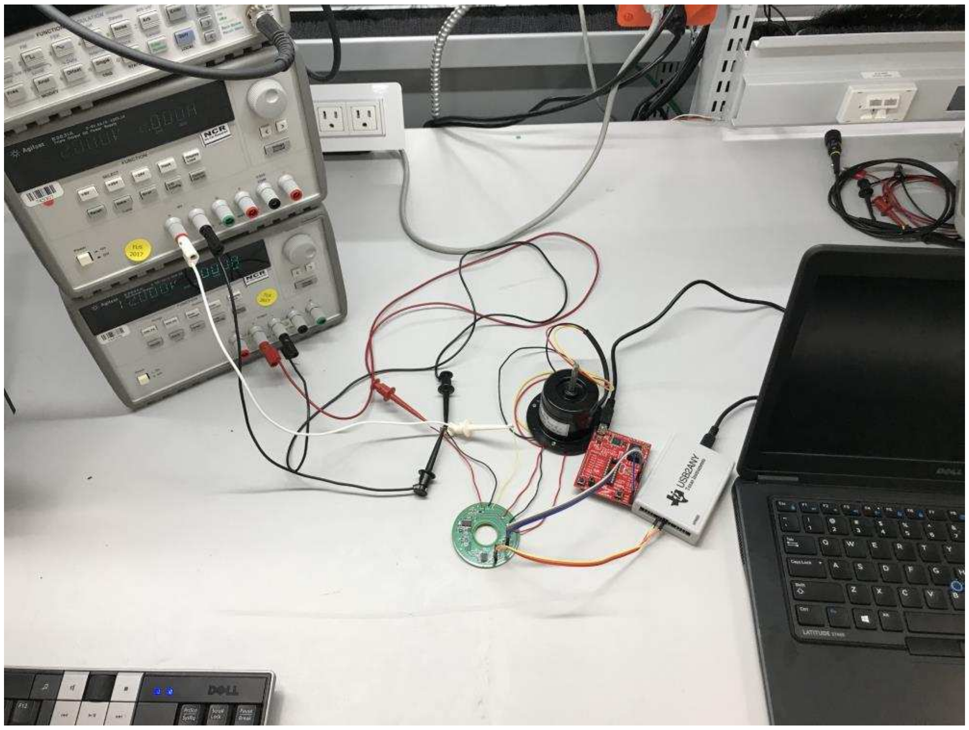


图 16. Example Bench Setup

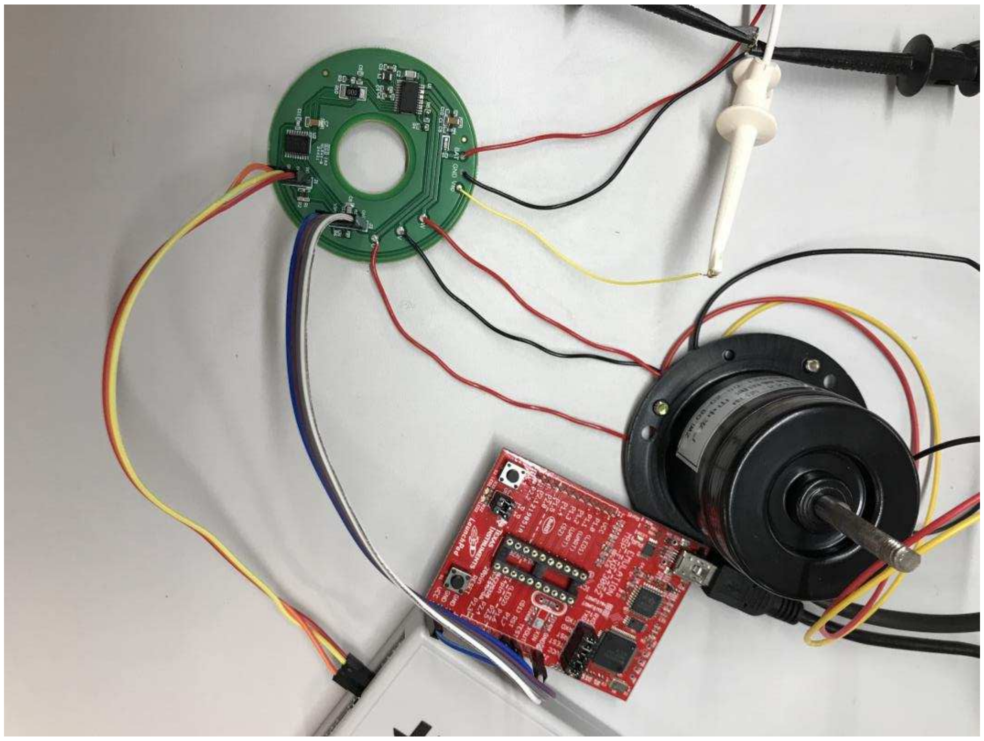
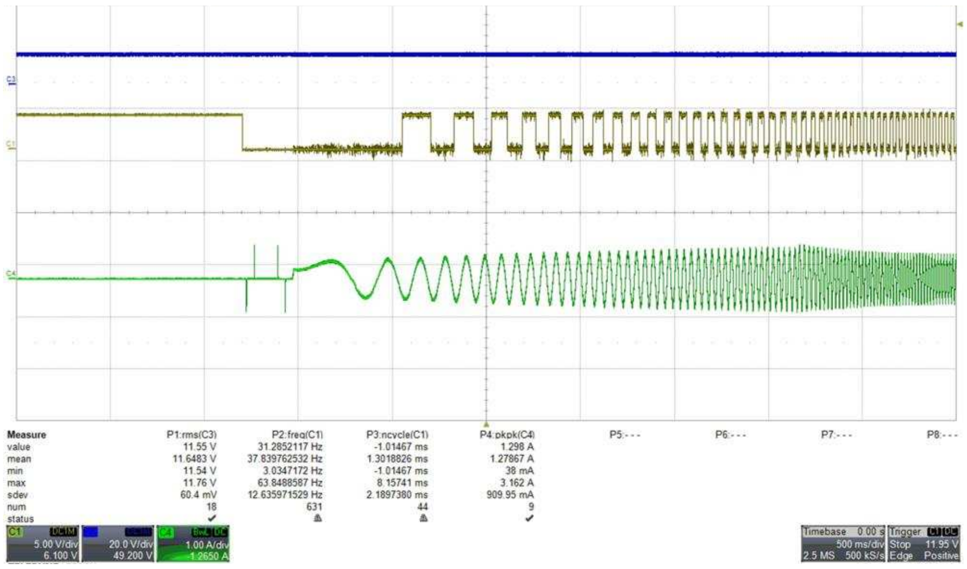


图 17. Closeup of Connections

3.2.2 Functional Test

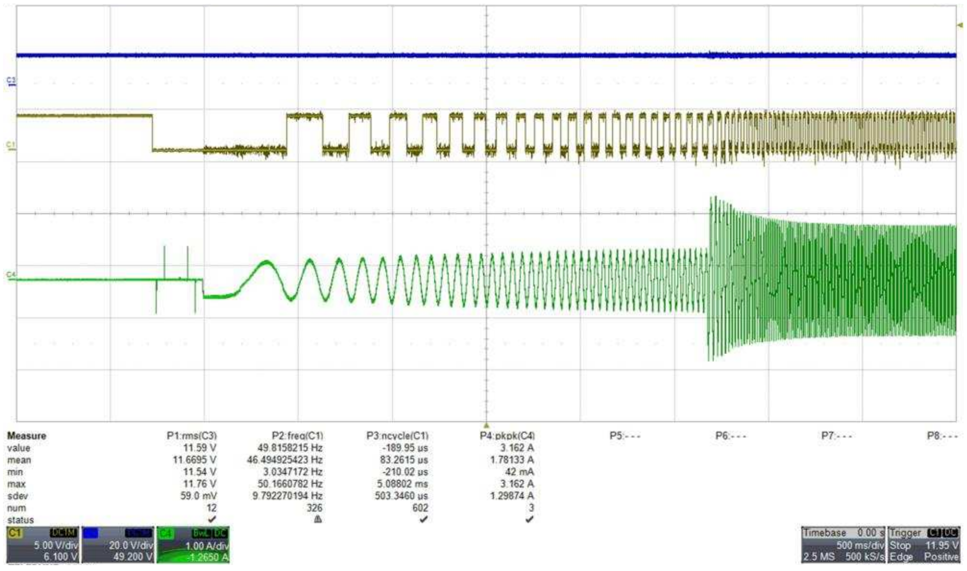
The functional test refers to basic programming and start up using both the DRV10xx Software GUI through I^2C and the MSP430 MCU using CCS and JTAG. The MSP430 MCU specifically is tested with a low- and high-voltage start-up of 8 V and 16 V, respectively.

C1 shows V_{BAT}, C2 shows the Frequency Generate (FG) pin on the DRV10983-Q1, and C3 shows the current of the U phase of the motor.



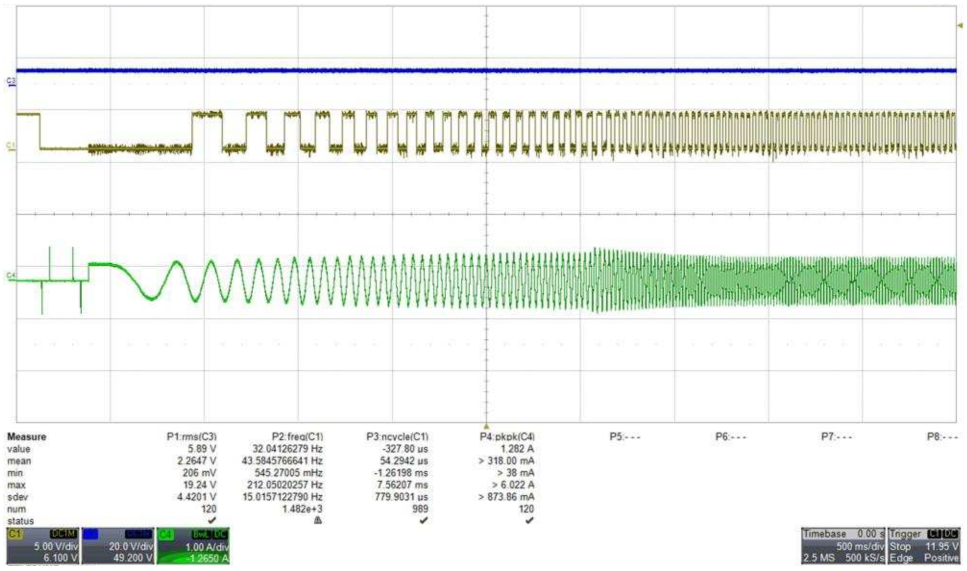
Measure	P1:rms(C3)	P2:freq(C1)	P3:ncycle(C1)	P4:pkpk(C4)	P5:- - -	P6:- - -	P7:- - -	P8:- - -
value	11.55 V	31.2852117 Hz	-1.01467 ms	1.298 A				
mean	11.6483 V	37.839762532 Hz	1.3018826 ms	1.27867 A				
min	11.54 V	3.0347172 Hz	-1.01467 ms	38 mA				
max	11.76 V	63.8488587 Hz	8.15741 ms	3.162 A				
sdev	60.4 mV	12.635971529 Hz	2.1897380 ms	909.95 mA				
num	18	631	44	9				
status	✓	⚠	⚠	✓				

图 18. Startup Profile, GUI Control, 12 V, 50% Speed



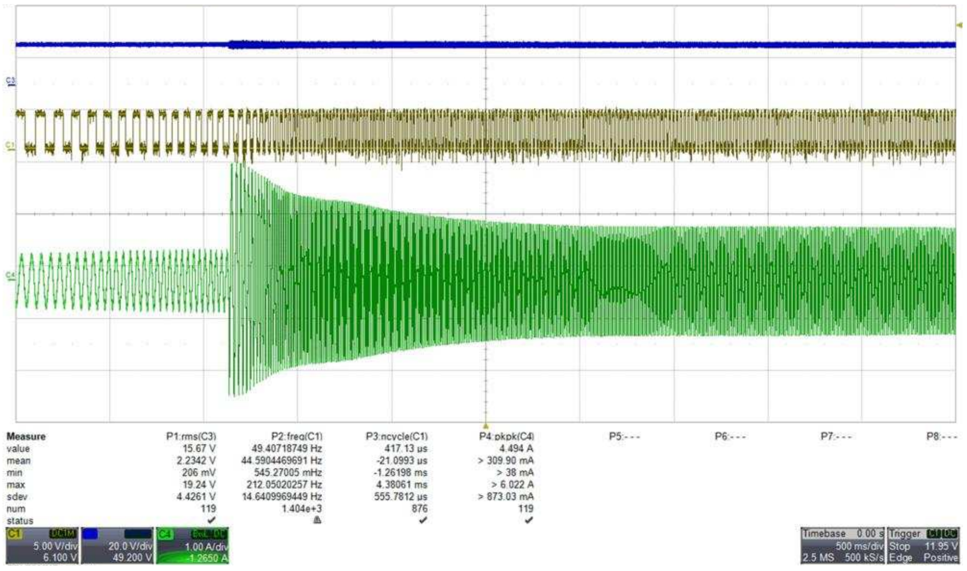
Measure	P1:rms(C3)	P2:freq(C1)	P3:ncycle(C1)	P4:pkpk(C4)	P5:- - -	P6:- - -	P7:- - -	P8:- - -
value	11.59 V	49.8158215 Hz	-189.95 µs	3.162 A				
mean	11.6695 V	46.494925423 Hz	83.2615 µs	1.78133 A				
min	11.54 V	3.0347172 Hz	-210.02 µs	42 mA				
max	11.76 V	50.1660782 Hz	5.08802 ms	3.162 A				
sdev	59.0 mV	9.792270194 Hz	503.3460 µs	1.29874 A				
num	12	326	602	3				
status	✓	⚠	✓	✓				

图 19. Startup Profile, GUI Control, 12 V, 100% Speed



```
Measure      P1:rms(C3)   P2:freq(C1)         P3:ncycle(C1)   P4:pkpk(C4)   P5:---   P6:---   P7:---   P8:---
value        5.89 V       32.04126279 Hz      -327.80 µs      1.282 A
mean         2.2647 V     43.5845766641 Hz    54.2942 µs      > 318.00 mA
min          206 mV       545.27005 mHz       -1.26198 ms     > 38 mA
max          19.24 V      212.05020257 Hz     7.56207 ms      > 6.022 A
sdev         4.4201 V     15.0157122790 Hz    779.9031 µs     > 873.86 mA
num          120          1.482e+3            989             120
status       ✓            ⚠                   ✓               ✓
```

图 20. Startup Profile, MSP430 Speed Control, 8 V, 100% Speed



```
Measure      P1:rms(C3)   P2:freq(C1)         P3:ncycle(C1)   P4:pkpk(C4)   P5:---   P6:---   P7:---   P8:---
value        15.67 V      49.40718749 Hz      417.13 µs       4.494 A
mean         2.2342 V     44.5904469691 Hz    -21.0993 µs     > 309.90 mA
min          206 mV       545.27005 mHz       -1.26198 ms     > 38 mA
max          19.24 V      212.05020257 Hz     4.38061 ms      > 6.022 A
sdev         4.4261 V     14.6409969449 Hz    555.7812 µs     > 873.03 mA
num          119          1.404e+3            876             119
status       ✓            ⚠                   ✓               ✓
```

图 21. Startup Profile, MSP430 Control, 16 V, 100% Speed

3.2.3 Speed Loop Regulation Test

Five motor specific speed levels are programmed. Each speed level is tested for voltage ranging from 8 to 16 V in 2-V increments. Measurements are taken in Hz but converted to RPM. For more information, see 表 6 and 图 22.

表 6. Output Speeds versus Input Speed Commands and Input Voltage Ranges

VOLTAGE INPUT	SPEED INPUT = 1000 RPM	SPEED INPUT = 1692 RPM	SPEED INPUT = 2236 RPM	SPEED INPUT = 2636 RPM	SPEED INPUT = 2956 RPM
8 V	1001	1691	2041	2210	2314
10 V	1000	1691	2236	2559	2686
12 V	1000	1691	2236	2636	2956
14 V	1000	1691	2236	2640	2956
16 V	1001	1691	2236	2641	2956

图 22 compares the output speeds versus input speed commands and input voltage ranges.

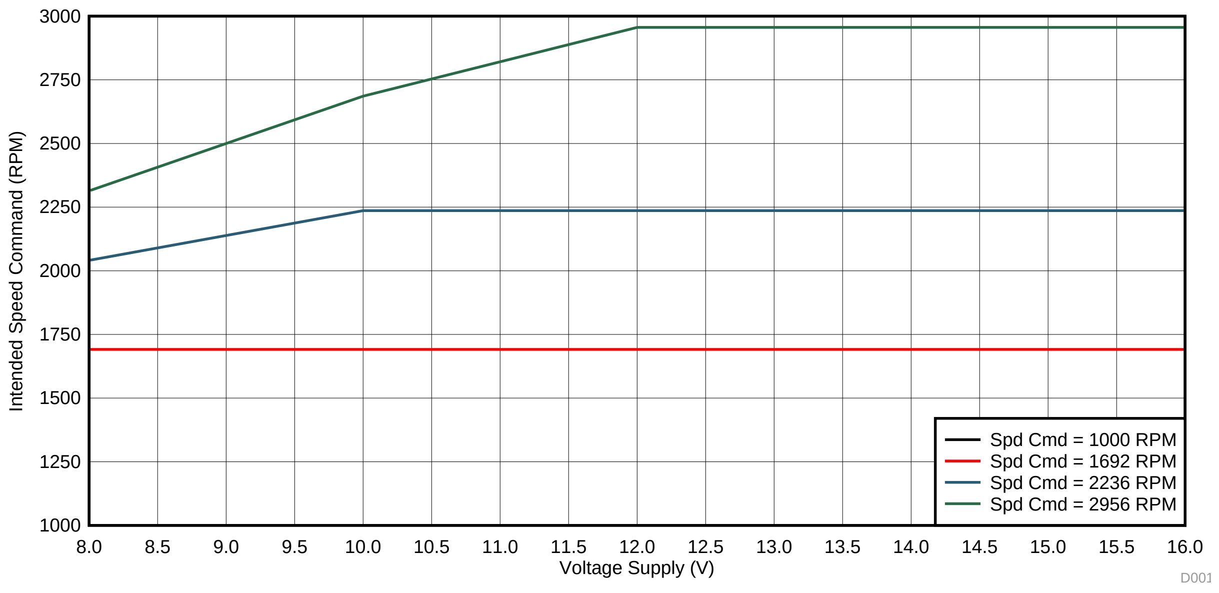


图 22. Speed Regulation: Speed (RPM) versus Voltage Input

3.2.4 Load Dump Test

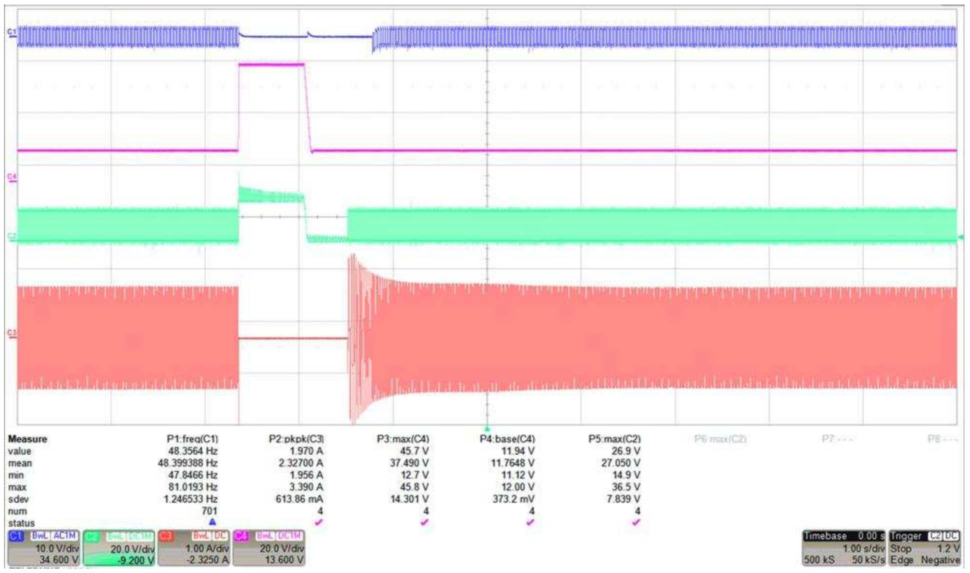
The recommended operation voltage of the DRV10983-Q1 device is from 6.2 V to 28 V. The device is able to drive the motor within this V_{CC} range. In the load dump condition, V_{CC} can rise up to 45 V. Once the DRV10983-Q1 device detects that V_{CC} is higher than VOV_R3 , it stops driving the motor and protects its own circuitry. Find more information in 表 7.

Load dump protection is effectively enabled by the device when the 45-V pulse is injected from the supply. The device shuts down to protect its circuitry and then turns back on again.

C1 shows the voltage on the Frequency Generator (FG) pin of the DRV10983-Q1, C2 shows the voltage on the U phase of the motor, C3 shows the current on the U phase of the motor, and C4 shows the voltage of V_{BAT}.

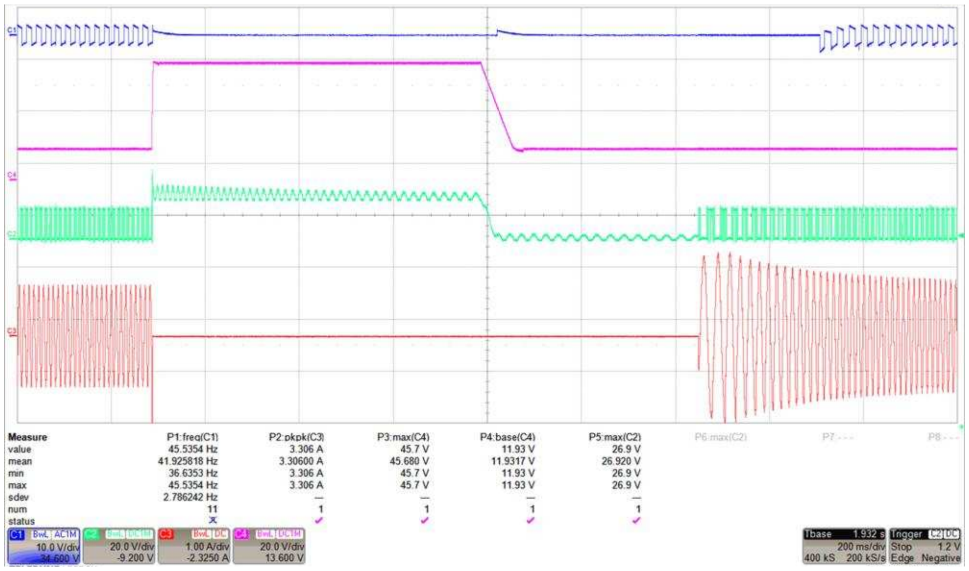
表 7. Load Dump Protection Specifications

LOAD DUMP PROTECTION	DESCRIPTION	MIN	TYP	MAX	UNIT
V_{OV_R}	Load dump protection mode entry on rising V_{CC} threshold	28.5	29.2	30	V
V_{OV_F}	Load dump protection mode entry on falling V_{CC} threshold	27.7	28.2	28.8	V
V_{OV_HYS}	Load dump protection mode hysteresis	0.73	1	1.1	V



```
Measure   P1:freq(C1)     P2:pkpk(C3)   P3:max(C4)   P4:base(C4)   P5:max(C2)   P6:max(C2)   P7:- - -   P8:- - -
value     48.3564 Hz      1.970 A       45.7 V       11.94 V       26.9 V
mean      48.399388 Hz    2.32700 A     37.490 V     11.7648 V     27.050 V
min       47.8466 Hz      1.956 A       12.7 V       11.12 V       14.9 V
max       81.0193 Hz      3.390 A       45.8 V       12.00 V       36.5 V
sdev      1.246533 Hz     613.86 mA     14.301 V     373.2 mV      7.839 V
num       701             4             4            4             4
status

C1 BwL AC1M    C2 BwL DC1M    C3 BwL DC    C4 BwL DC1M        Timebase   0.00 s    Trigger  C2 DC
10.0 V/div     20.0 V/div     1.00 A/div   20.0 V/div                    1.00 s/div  Stop      1.2 V
34.600 V       -9.200 V       -2.3250 A    13.600 V           500 kS     50 kS/s    Edge   Negative
```

图 23. Load Dump Protection Waveforms After Injecting 45-V Pulses



Measure	P1:freq(C1)	P2:pkpk(C3)	P3:max(C4)	P4:base(C4)	P5:max(C2)	P6:max(C2)	P7:---	P8:---
value	45.5354 Hz	3.306 A	45.7 V	11.93 V	26.9 V			
mean	41.925818 Hz	3.30600 A	45.680 V	11.9317 V	26.920 V			
min	36.6353 Hz	3.306 A	45.7 V	11.93 V	26.9 V			
max	45.5354 Hz	3.306 A	45.7 V	11.93 V	26.9 V			
sdev	2.786242 Hz	—	—	—	—			
num	11	1	1	1	1			
status	✗	✓	✓	✓	✓			

图 24. Load Dump Protection Zoomed Waveforms After Injecting 45-V Pulses

The load dump voltage signal is created using the sequence outlined in 表 8.

表 8. Sequence for Creating Load Dump Signal

SEQUENCE	DURATION (s)	SLEW RATE (V/ms)	VOLTAGE (V)
1	5.5	0.001	12
2	0.3	10	45
3	0.4	0.001	45
4	0.3	10	12
5	5.5	0.001	12

3.2.5 Thermal Test

Thermal testing is conducted for a range of temperatures. The device operated and spun the motor at all temperatures. The device operated on average 20°C higher than the ambient temperature, which remained consistent in all the results.

注: The numbers shown in the following images are in degrees Celsius.

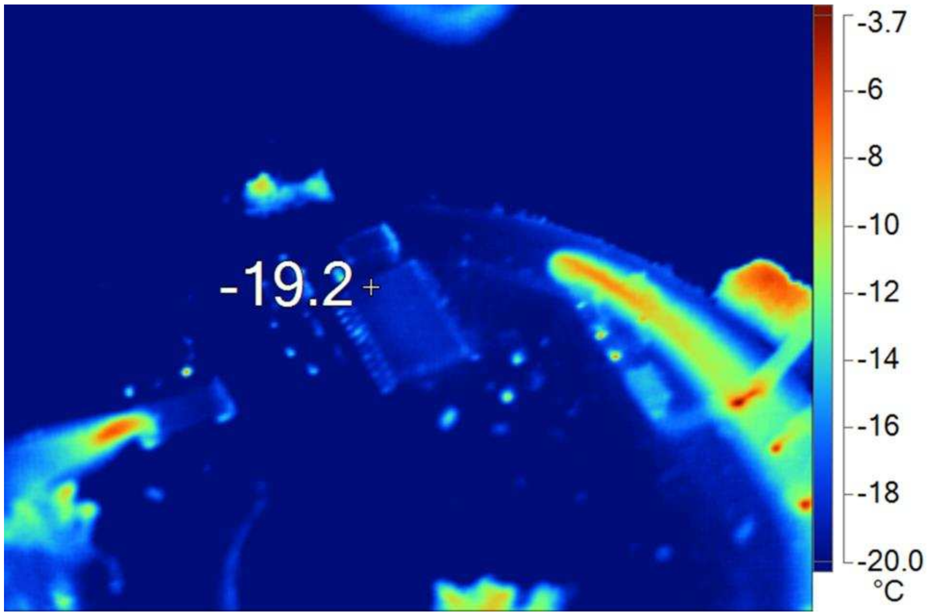


图 25. Thermal Image After 3 Hours at 12 W, –40°C Ambient

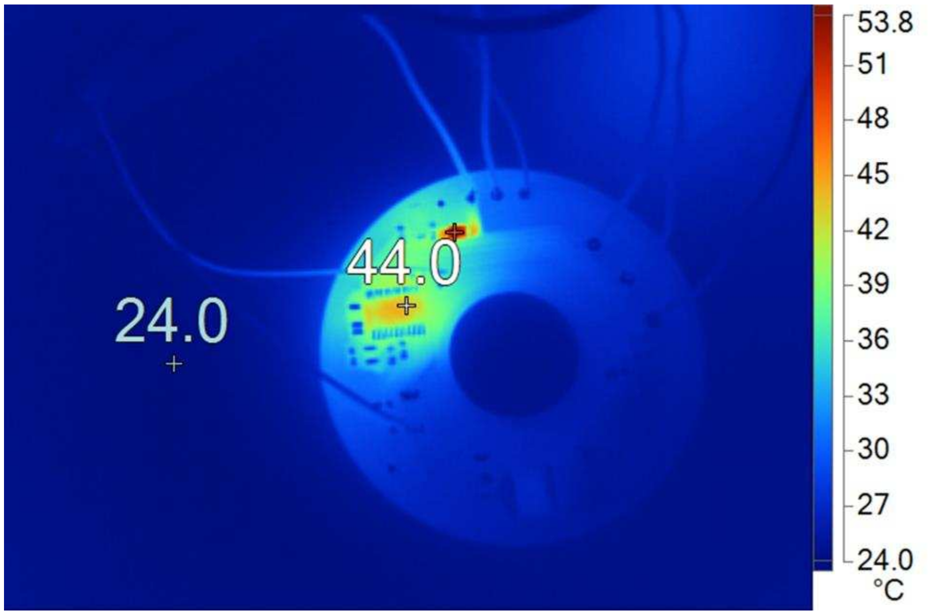


图 26. Thermal Image After 15 Hours at 12 W, –40°C Ambient

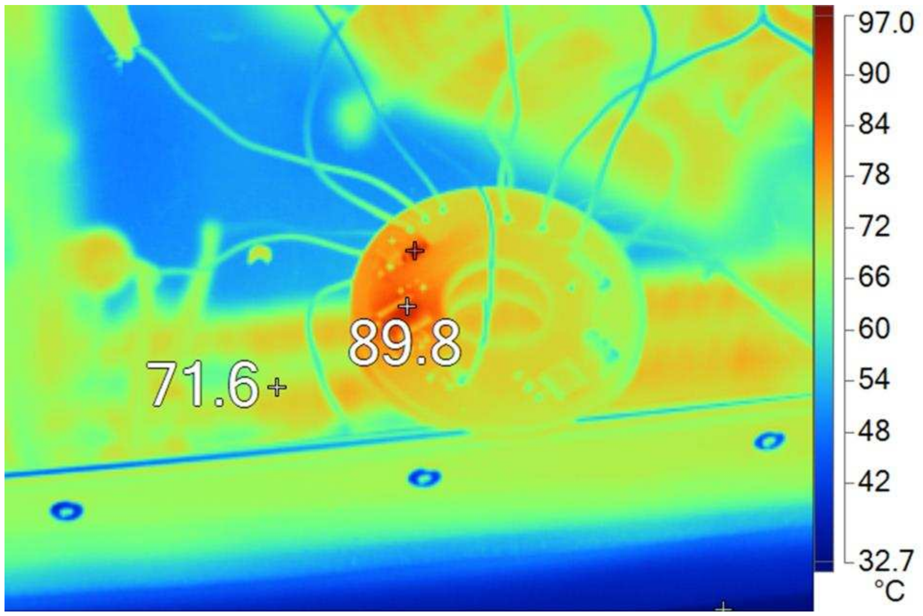


图 27. Thermal Image After 15 Hours at 12 W, 70°C Ambient

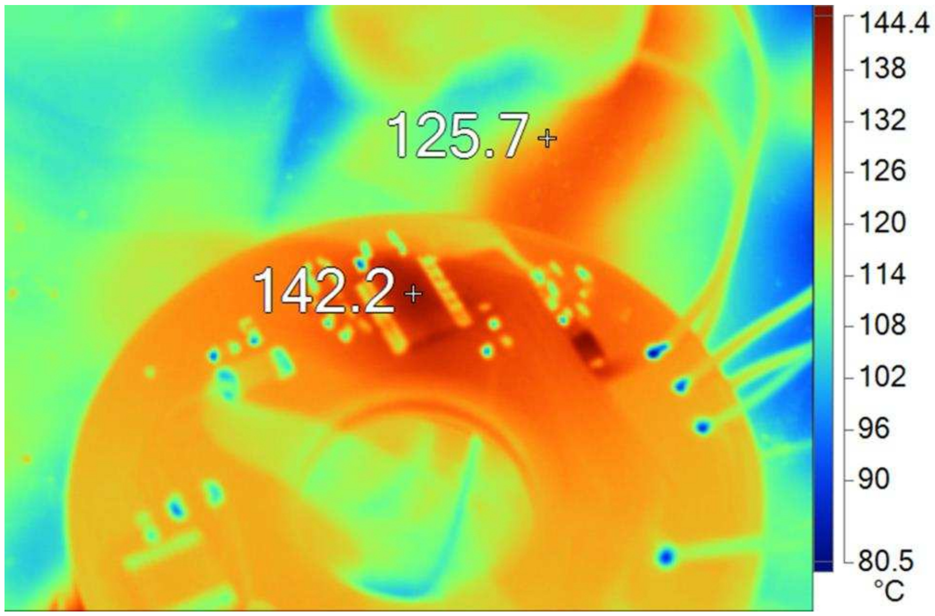


图 28. Thermal Image After 15 Hours at 12 W, 125°C Ambient

4 Design Files

4.1 *Schematics*

To download the schematics, see the design files at TIDA-01496.

4.2 *Bill of Materials*

To download the bill of materials (BOM), see the design files at TIDA-01496.

4.3 *PCB Layout Recommendations*

- Place the VCC, GND, U, V, and W pins with thick traces because high current passes through these traces.
- Place the 10-µF capacitor between VCC and GND, and as close to the VCC and GND pins as possible.
- Place the capacitor between CPP and CPN, and as close to the CPP and CPN pins as possible.
- Connect the GND, PGND, and SWGND under the thermal pad.
- Keep the thermal pad connection as large as possible, on both the bottom side and top sides. The connection must be one piece of copper without any gaps.
- Single-layer boards often suffer in optimal thermal and GND loop designs. Layout can be supplemented using 0-Ω resistors to connect isolated GND planes.

4.3.1 Layout Prints

To download the layer plots, see the design files at TIDA-01496.

4.4 *Altium Project*

To download the Altium project files, see the design files at TIDA-01496.

4.5 *Gerber Files*

To download the Gerber files, see the design files at TIDA-01496.

4.6 *Assembly Drawings*

To download the assembly drawings, see the design files at TIDA-01496.

5 Software Files

To download the software files, see the design files at TIDA-01496.

注: The software files of this reference design require a TI login to access.

6 Related Documentation

1. Texas Instruments, *DRV10983-Q1 Tuning Guide*
2. Texas Instruments, *DRV10983-Q1 Evaluation Module User's Guide*

6.1 商标

E2E, MSP430, LaunchPad, Code Composer Studio are trademarks of Texas Instruments.
All other trademarks are the property of their respective owners.

7 Terminology

BLDC —	Brushless DC motor
ESD—	Electrostatic discharge
FETs, MOSETs—	Metal-oxide-semiconductor field-effect transistor
IGBT—	Insulated gate bipolar transistor
IPD—	Initial position detection
OCP—	Overcurrent protection
OTP—	Overtemperature protection
PWM—	Pulse width modulation
RPM—	Revolutions per minute
RMS—	Root mean square
UVLO—	Undervoltage lockout

8 About the Authors

COLE MACIAS is an applications engineer at Texas Instruments, where he is responsible for supporting customers and customer issues for the DRV10x family of products. These include 3-phase BLDC motor drive solutions in a wide range of applications. Cole obtained his BSEE at Arizona State University.

MOSTAFA SHUBBAR is a product marketing engineer at Texas Instruments currently working in the Integrated Motor Controller group. He completed his BSEE at the University of Texas at Dallas and is currently pursuing an MSEE at UTD.