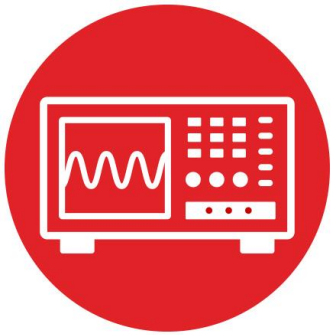

Module 12

Lab 12: DC motors

Lab: DC motors

12.0 Objectives

The purpose of this lab is to build the electronics needed to spin the motors. The hardware interface includes an H-bridge motor driver using the TI DRV8838 driver that allows the software to spin each motor forward or backward. The software can vary the **electrical power** delivered to each motor using **pulse width modulation** (PWM). In this module,

1. You will learn the electromagnetic aspects of the motor.
2. You will attach the motors and wheels to the robot.
3. You will use the driver board to interface the motors to the microcontroller.
4. You will measure the voltage and current to the motors.
5. You will perform an analysis of the behavior of the motor, plotting motor speed versus duty cycle.

Good to Know: Even though you will measure motor speed as a function of duty cycle, this relationship depends on many factors that can change over time, such as motor efficiency, battery voltage, voltage drop in the H-bridge, mechanical forces, and friction. For all practical purposes, without sensors, the software can only choose to go faster or to go slower, but it cannot set the motor speed. On this robot, there are two motors in differential drive configuration. This means even the simplest operation such as moving in a straight line will require sensor feedback. There are three such sensors available in this course: the line sensor (Module 6), the IR distance sensors (Module 15), and the tachometer (Module 16).

12.1 Getting Started

12.1.1 Software Starter Projects

Look at these two projects:
Lab09_SysTick (your solution to Lab 9)
Lab12_Motors (starter project for this lab)

Note: Please do not use the voltmeter, oscilloscope or logic analyzer created by TExaS for this lab. Voltages applied to inputs of the MSP432 must remain between 0 and 3.3V. Voltages outside this range will damage the MSP432.

12.1.2 Student Resources (in datasheet directory)

MotorDriverPowerDistribution.pdf	Data sheet for power board
Pololu Romi Chassis User's Guide.pdf	How to build the robot
drv8838.pdf	Data sheet for the H-bridge driver

12.1.3 Reading Materials

Volume 1 Sections 8.1, 8.6, and 8.7
Embedded Systems: Introduction to the MSP432 Microcontroller",
or
Volume 2 Sections 1.4 and 6.5
Embedded Systems: Real-Time Interfacing to the MSP432 Microcontroller"

Read the specifications on the Motor Driver and Power Distribution board
website https://www.pololu.com/product/3543
https://www.pololu.com/docs/0J68

12.1.4 Components needed for this lab

Quantity	Description	Manufacturer	Mfg P/N
1	MSP-EXP432P401R LaunchPad	TI	MSP-EXP432P401R
1	Romi Chassis Kit - Red	Pololu	3502
1	Motor Driver and Power Distribution Board for Romi	Pololu	3543
1	Romi Encoder Pair Kit, 12 CPR* (optional)	Pololu	3542
2	Rechargeable Battery, Pack of 4, Metal Hydride 1300 mAh, 1.2V, AA	Energizer	626831
4	1.375in 4-40 Nylon standoff	Keystone	4809
2	0.187in 4-40 metal nut	Keystone	4694
6	0.5in 4-40 Nylon machine screw	Pololu	1962

12.1.5 Lab equipment needed

Oscilloscope (one or two channels at least 10 kHz sampling)

Lab: DC motors

Voltmeter, ohmmeter, and current meter

12.2 System Design Requirements

The goal of this lab is to place the motors and wheels on the robot and configure the motor control board so the software can control the two motors. The Motor Driver and Power Distribution Board (**MDPDB**) used in Module 5 lab also includes two H-bridge drivers (TI DRV8838) that provide the voltage and current needed to spin the motors.

First, you will mechanically build, and then electrically connect the two motors, two wheels, the caster, and the **MDPDB**. Six control signals will be connected from the microcontroller to the **MDPDB** so the software can control both motors {forward, stop, reverse}. Furthermore, you will use the PWM software from Lab 9 to adjust the delivered power to the two wheels.

The second part of this lab is to study the behavior of the motor. You will measure voltage (volts), current (amps), average power (watts), and rotational speed (rpm) of the DC motor as a function of duty cycle.

The outcome of this lab is to build a system that drives in more or less a straight line until one of the bump sensors detects a collision.

12.3 Experiment set-up

The first step is to read the data sheet for the Romi chassis, and follow the directions on **https://www.pololu.com/docs/0J68/all** to connect the two wheels, caster, two motors, and motor board per instructions to the Romi chassis. Figure 1 shows some of the parts needed for the robot.

Note A: If you do not intend to buy and build the tachometer, labeled as Encoder in Figure 1 (used in Lab 16 with the Romi Encoder Pair Kit, 12 CPR https://www.pololu.com/product/3542), then you will solder four wires from the two motors to the motor board (MR+, MR-, ML+, ML-).

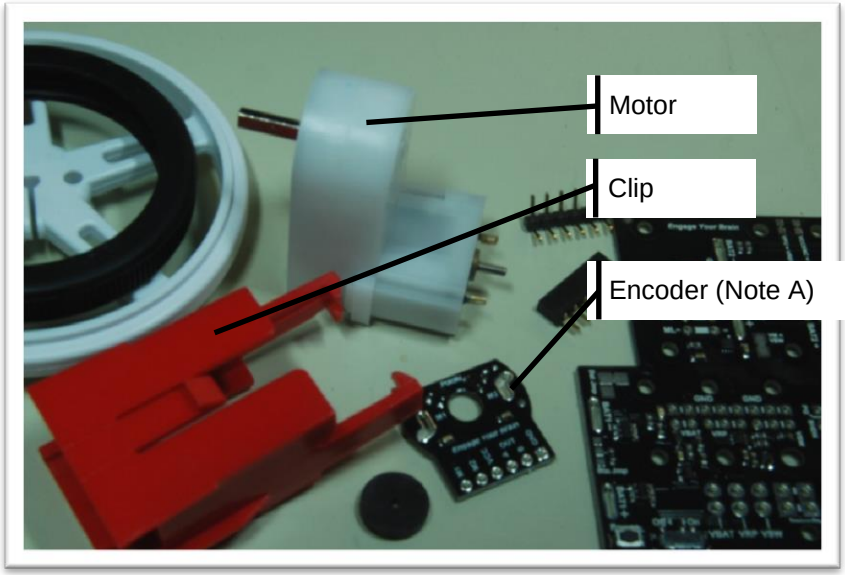

Figure 1. Parts needed to build the motor system.

Next, you will connect six wires from the **MDPDB** to the LaunchPad. Since these signals are on the regular LaunchPad connectors, you can use either male or female wires on the LaunchPad side (the robots in the figures use female connectors). Figure 2 shows a possible interface circuit. On the **MDPDB** side you can solder wires directly, or solder a male header into the **MDPDB** and use female-female cables, see Figure 3. Refer to the data sheet of the DRV8838 to see how the software output values to these six signals affect motor behavior.

Lab: DC motors

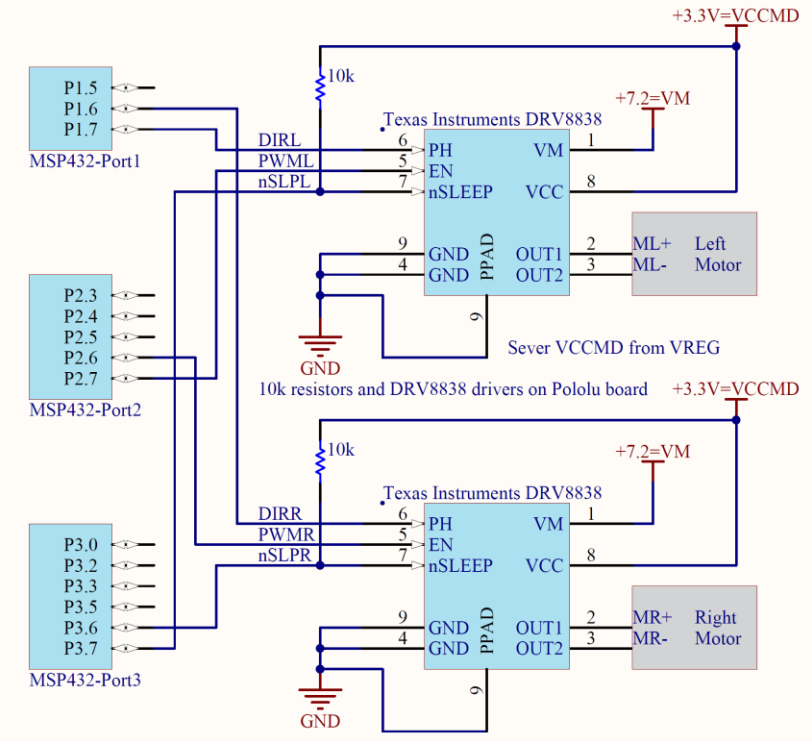

Figure 2. Interface circuit.

LaunchPad	MDPDB	DRV8838	Description
P1.6	DIRR	PH	Right Motor Direction
P3.6	nSLPR	nSLEEP	Right Motor Sleep
P2.6	PWMR	EN	Right Motor PWM
P1.7	DIRL	PH	Left Motor Direction
P3.7	nSLPL	nSLEEP	Left Motor Sleep
P2.7	PWML	EN	Left Motor PWM

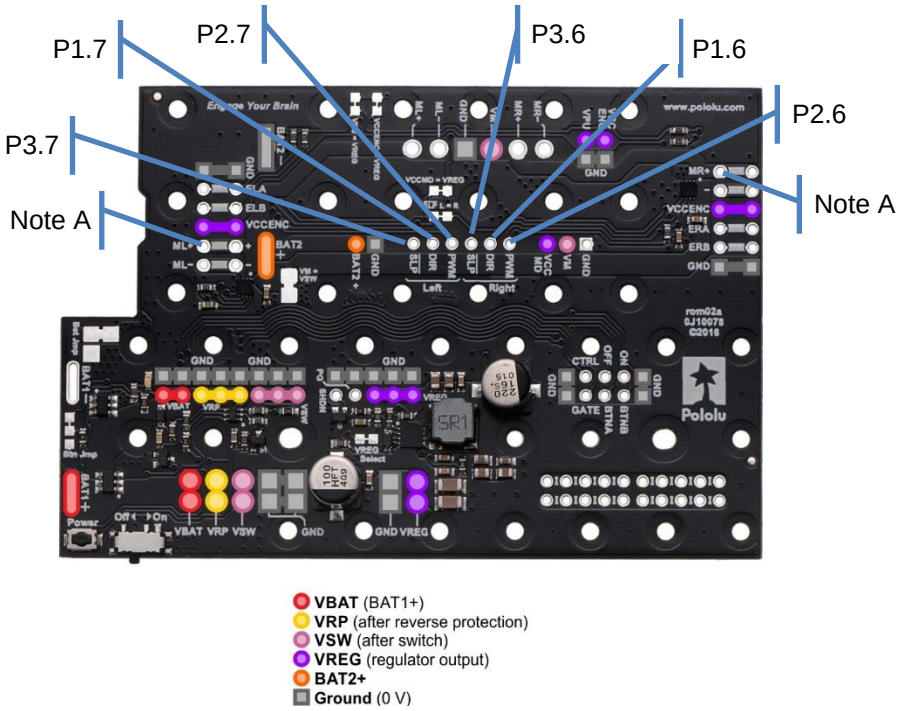

Figure 3. Motor Driver and Power Distribution Board for Romi Chassis. Refer back to Module 5 for power and ground connections. See instructions for Romi chassis for how to connect motors and encoders to the board.

Figure 4 shows a partially completed wheel assembly, and Figure 5 shows one completed wheel assembly.

Warning: Disconnect the VREG↔+5V wire when the LaunchPad USB cable is connected to the PC. Connect the VREG↔+5V wire when the robot is running on battery power. This way the motors always get power from the batteries, and never get power from the USB.

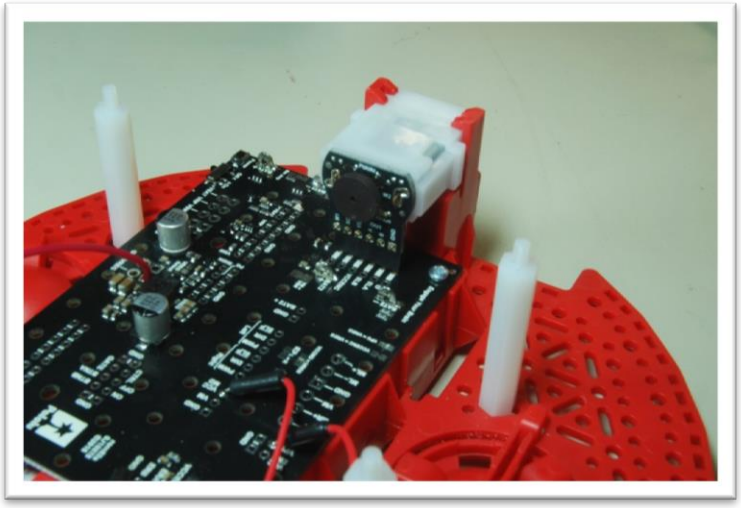

Figure 4. Partially completed wheel assembly.

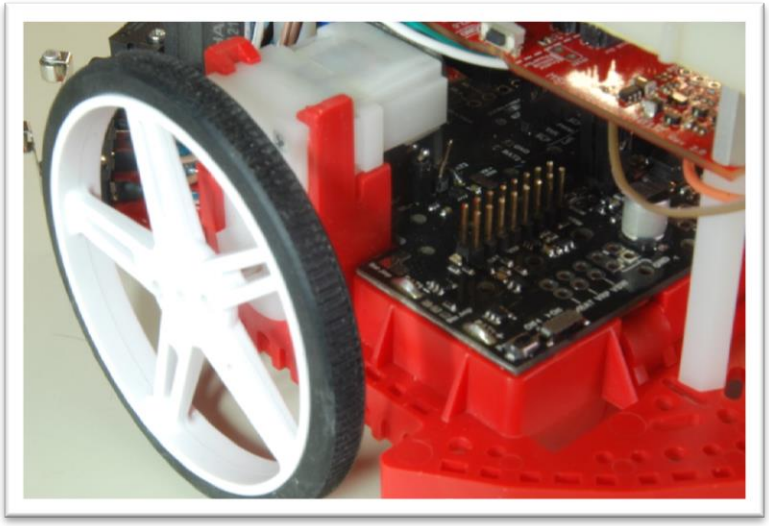

Figure 5. Completed wheel assembly.

12.4 System Development Plan

12.4.1 Low-level software driver

You will start with creating a suite of software functions that control the two wheels on the robot. The frequency of the PWM signal sent to both motors should be 100 Hz (10ms). In this lab, we will keep the duty cycle the same for both motors as well. In the next module, we will use the hardware timer so each motor will have its own duty cycle. To stop the motors you will stop the PWM and set the **nSleep** signal to 0. Use the simple approach of Lab 9 to create the PWM signals. The prototypes for the driver are:

void Motor_InitSimple(void);
Initializes the 6 GPIO lines and puts driver to sleep
Returns right away

void Motor_StopSimple(void);
Stops both motors, puts driver to sleep
Returns right away

void Motor_ForwardSimple(uint16_t duty, uint32_t time)
Drives both motors forward at **duty** (100 to 9900)
Runs for **time** duration (units=10ms), and then stops
Stop the motors and return if any bumper switch is active
Returns after **time***10ms or if a bumper switch is hit

void Motor_BackwardSimple(uint16_t duty, uint32_t time)
Drives both motors backward at **duty** (100 to 9900)
Runs for **time** duration (units=10ms), and then stops
Stop the motors and return if any bumper switch is active
Returns after **time***10ms or if a bumper switch is hit

void Motor_LeftSimple(uint16_t duty, uint32_t time)
Drives just the left motor forward at **duty** (100 to 9900)
Right motor is stopped (sleeping)
Runs for **time** duration (units=10ms), and then stops
Stop the motor and return if any bumper switch is active
Returns after **time***10ms or if a bumper switch is hit

void Motor_RightSimple(uint16_t duty, uint32_t time)
Drives just the right motor forward at **duty** (100 to 9900)
Left motor is stopped (sleeping)
Runs for **time** duration (units=10ms), and then stops
Stop the motor and return if any bumper switch is active
Returns after **time***10ms or if a bumper switch is hit

Lab: DC motors

12.4.2 Control of the motor

In this part of the lab you will implement the functions to test the motors. Place voltmeters on the VM line (+7.2) and on VREG line (+5V) the first time you power up the wheeled robot. Place the robot on blocks, so the wheels do not touch the ground, and test the low-level motor functions, using a program like **Program12_1**. This allows the motors to spin without the robot moving. With the wheels off the ground, there will be minimal friction, the fastest rotation, and the smallest current.

```
// Driver test
void Pause(void){
  while(LaunchPad_Input()==0);  // wait for touch
  while(LaunchPad_Input());     // wait for release
}
int Program12_1(void){
  Clock_Init48MHz();
  LaunchPad_Init(); // built-in switches and LEDs
  Bump_Init();      // bump switches
  Motor_InitSimple();     // your function
  while(1){
    Pause();
    Motor_ForwardSimple(5000,2000);  // your function
    Pause();
    Motor_BackwardSimple(5000,2000); // your function
    Pause();
    Motor_LeftSimple(5000,2000);     // your function
    Pause();
    Motor_RightSimple(5000,2000);    // your function
  }
}
```

Use an oscilloscope to observe the motor signals motor board (MR+, MR-, ML+, ML-) during operation. *You should see voltage versus time. The voltage difference between MR+ and MR- is the applied voltage to the motor.*

Note: As mentioned in Lab 9, using software delays to create PWM consumes all of the processor time. In the next module, we will use the hardware timers on the microcontroller to create the two PWM outputs. In this way, software needs to execute only when it wishes to change the duty cycle or change direction.

12.4.3 Behavior

From an electrical standpoint the motor has three components, resistance (caused by the long wires), inductance (caused by the coiled wires) and electro motive force (emf -voltage caused by the coupling between mechanical and electrical forces). Begin by measuring the resistance of the motor when all power is turned off and the motor is not spinning. Let **R** be this static resistance. Assuming a voltage of 7V, use Ohm's Law to calculate the expected current.

In this section, you will measure actual voltage (**V** in volts), current (**I** in amps), and speed (**s** in rpm) as a function of the **duty** parameter (2000 to 8000). If you place the robot on blocks and attach string/yard/tape to a wheel you can both see and hear the wheel turn. First you will use a stopwatch to count the number of rotations in a fixed time (e.g., 60 seconds).

There are two approaches to measuring motor **current** (**I**). One approach is to remove the batteries and connect a bench supply (which allows you to set the voltage to 7.2V and measure the current) to power the robot. The second approach is to place a current meter in the loop between the batteries and the robot. For example, you can make a 3-layer stack of wire-insulator-wire, and place this stack between the contacts in the battery compartment. You then can place the current meter on the two wires. You can measure motor **voltage** (**V**) with the oscilloscope and verify which duty cycle is active. You will first measure current to the robot with the motors stopped, and then you will measure voltage, current, speed required to spin one motor. The difference in these two current measurements is the current to the motor. You can use a program like **Program12_2** to collect data.

```
// Voltage current and speed as a function of duty cycle
int Program12_2(void){ uint16_t duty;
  Clock_Init48MHz();
  LaunchPad_Init();   // built-in switches and LEDs
  Bump_Init();        // bump switches
  Motor_InitSimple(); // initialization
  while(1){
    for(duty=2000; duty<=8000; duty=duty+2000){
      Motor_StopSimple();   // measure current
      Pause();
      Motor_LeftSimple(duty,6000);  // measure current
    }
  }
}
```

Lab: DC motors

Make a table and graphs of voltage, current, power, emf, and speed as a function of duty cycle. Calculate **emf** as

emf = **V** – **I***R**

where **V** is the measured motor voltage, **I** is the measured motor current, and **R** is the static resistance of the motor. Under normal operating conditions, emf will be negative, meaning it draws more current than predicted using the static resistance. Calculate power as

P = **V** * **I** * **duty**/10000

Describe the general behavior of the motor.

Perform a maximum speed test using **Program12_3**. First measure the rotational speed of the motors when the robot is on blocks, and then repeat the measurement when the robot is on the ground.

```
int Program12_3(void){
  Clock_Init48MHz();
  LaunchPad_Init();   // built-in switches and LEDs
  Bump_Init();        // bump switches
  Motor_InitSimple(); // initialization
  while(1){
    Pause();
    Motor_ForwardSimple (9900,1500); // max speed 15 s
  }
}
```

12.5 Troubleshooting

Motors not do spin or gets hot:

- Remove power and double check the connections.
- Review steps in Lab 5.
- Recharge the batteries.
- Verify the six signals from the LaunchPad to the motor board using a voltmeter, an oscilloscope or a logic analyzer.

One motor spins faster than the other:

- It is normal for the motor speeds to be ±20% of each other
- Check for friction on the slower motor

12.6 Things to think about

In this section, we list thought questions to consider after completing this lab. These questions are meant to test your understanding of the concepts in this lab. The goal of this module is for you to experience voltage, current, and power as they relate to DC motors.

- How does friction affect motor current?
- In this lab, we do not set the speed or the current. Rather, we set just the voltage and duty cycle. Why is it difficult in this lab for the robot to go straight?
- How does the two H-bridges allow the robot to turn, to back up?
- How does the software adjust power delivered to the motors?
- In what two ways could software cause the robot to turn?

12.7 Additional challenges

In this section, we list additional activities you could do to further explore the concepts of this module. For example,

- If you do not have the Pololu motor board, you could build your own H-bridge circuits to control the motors on the robot. In particular, you could build two H-bridges described in lecture using the L293. If you build your own H-bridge please test it before attaching the motors and before attaching the microcontroller.
- An impossible challenge would be to try to write software that makes the robot travel in a square pattern. Basically, repeat this two-step process: 1) go straight for a fixed amount of time; 2) turn left 90 degrees. It will not be possible. However, it will be instructive to determine why the effort fails.

12.8 Which modules are next?

There are two major limitations to the robot conceived in this lab. 1) the software consumes all the processor time, and 2) the speed of the motors depends on many factors most of which cannot be predicted in advance. Over the remaining labs we will solve these limitations.

Module 13) Use timers to create PWM signals, and use interrupts to manage multiple software tasks
Module 15) Use the ADC to interface distance sensors. Two distance sensors can be used to drive the robot at a fixed distance and fixed angle to the wall.
Module 16) Interface tachometers (Romi Encoder Pair Kit) and use timer capture to measure the speeds of each wheel directly.
Module 17) Combine modules 12, 13, and 16 to create a control system that does spin the motors at a desired speed.

12.9 Things you should have learned

In this section, we review the important concepts you should have learned in this module:

- Understand voltage, current, and power to a motor.
- Be able to use PWM output to adjust power to the motors.
- Understand basic operation and purpose of an H-bridge.
- Know how to write and test a low-level software driver.

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ('TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

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