User's Guide for the TIDA-01373

Quick Start Guide

Literature Number: SLVUB30A
April 2017–Revised October 2017

Contents

List of Figures

List of Tables

User's Guide for the TIDA-01373

This guide assists the user in working with the TIDA-01373; a *DRV10983, DRV10975, DRV10983-Q1, and DRV10987 EEPROM Programming Tool Reference Design* (TIDUCX2). Before teaching the user how to operate the TIDA-01373, this guide assists the user in understanding the process for programming the EEPROM registers of the DRV10983, DRV10975, DRV10983-Q1, and DRV10987 motor drivers. The procedures in this document aid in educating the user so that they may use the design files and collateral of the TIDA-01373 system and scale them to their own specific requirements. Standard I²C is used to communicate with the device to read and write values to the registers.

1 Using I²C to Communicate With the DRV10983 and DRV10975 Motor Drivers

The DRV10983 and DRV10975 both use the standard I²C protocol and are the slave device in the communicating pair. Both the DRV10983 and the DRV10975 have the slave address 1010 010. When a master addresses the motor driver, it sends 8 bits, the address and one bit specifying a write (0) or read (1). The master sends 1010 0100 (0xA4) for write and 1010 0101 (0xA5) for read. To program these devices, first load the registers with the desired motor parameter values. When the registers have the desired values in them, setting the eeWrite bit writes the Data in the registers into EEPROM.

1.1 Writing to a Register

Send the following basic command chains (see Figure 1 and Figure 2) from the master to the slave to program the register settings. The first command enables the Sidata bit. If this bit is not enabled, the register values cannot change. Set the Sidata bit to '1', one time before setting all of the register values.

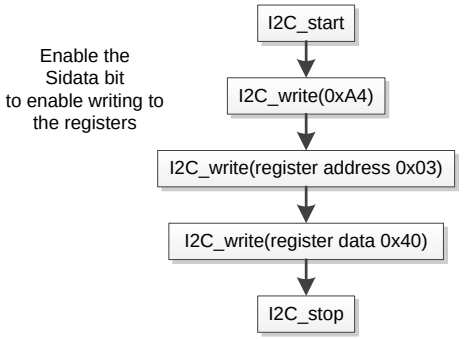


Figure 1. I²C Command Chain for Enabling Sidata Bit

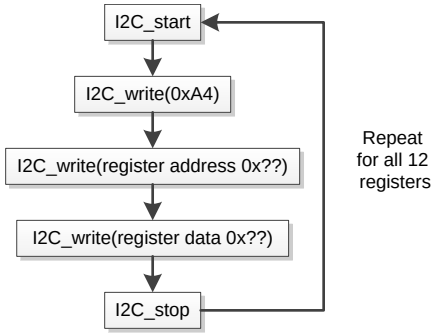


Figure 2. I²C Command Chain for Setting a Register

Figure 3 shows the waveforms for writing the value 0x39 to register 0x20.

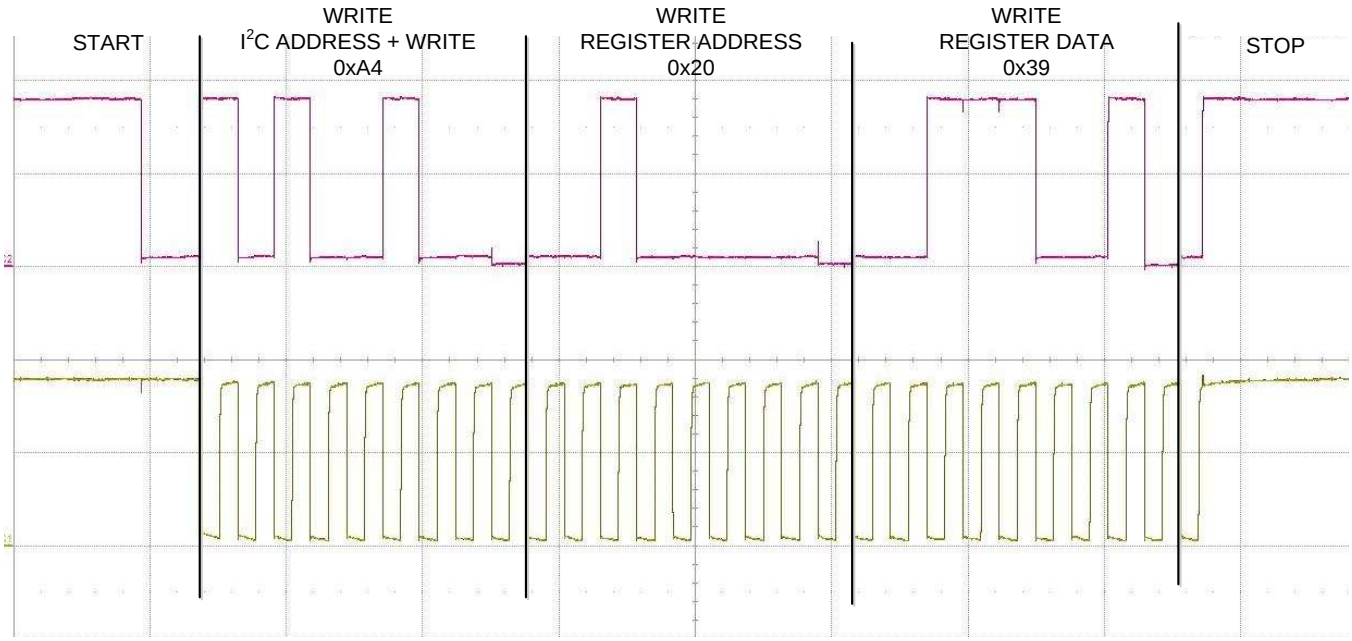


Figure 3. I²C Waveform of Writing to a Register

Repeat those steps as many times as needed until all 12 registers (0x20 through 0x2B) are loaded with the correct values. For specific information on the registers, see the DRV10983 data sheet (SLVSCP6) and the DRV10975 data sheet (SLVSCP2). The logic core runs the motor based on the register values so it is not necessary to write the register values to EEPROM. However, writing the values to EEPROM saves the parameters and auto loads them after a power cycle.

1.2 Writing the Registers to EEPROM

After the registers are loaded, the next step is to write them to EEPROM so the device can maintain those settings. For the device to properly write to EEPROM, the V_{CC} must be at least 22 V and have at least a 24-ms delay before power cycling or refreshing the device. The DRV10983 and DRV10975 calculate V_{CC} differently, adjust each device in the software. For the DRV10983, the power supply voltage is calculated using Equation 1 and recorded in the register SupplyVoltage indicated in the device data sheet. For the DRV10975 the power supply voltage is calculated using Equation 2 and recorded in the register SupplyVoltage indicated in the device data sheet.

$$V_{POWERSUPPLY}(V) = SupplyVoltage \times \frac{30\,V}{256} \quad (1)$$

$$V_{POWERSUPPLY}(V) = SupplyVoltage \times \frac{22.8\,V}{256} \quad (2)$$

Figure 4 describes the command chain for writing to EEPROM. First, enter the program key (0xB6) in the device control register (0x02), and then immediately after, set the eeWrite bit to 1 (0x50) in the EEPROM control register (0x03). If the eeWrite bit is not set directly after the program key is entered, the program key will be reset.

The programming time is about 24 ms, and when finished, the device clears the eeWrite bit. After the data is stored in EEPROM, the device can be powered down, and upon power-up, it auto loads the values into the registers.

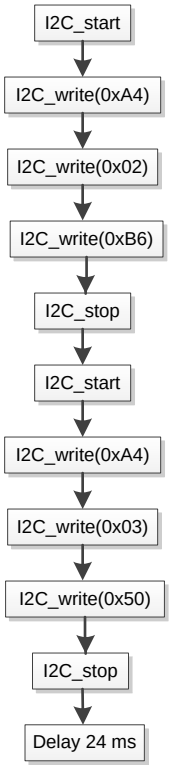


Figure 4. I²C Command Chain for Writing to EEPROM

1.3 Reading the Registers

Reading the registers is a way to verify the values saved in EEPROM or to obtain feedback from both the DRV10983 and DRV10975. Because the EEPROM cannot be read directly in these two devices, the settings saved in EEPROM can be loaded into the registers by two methods. Power cycling the device auto loads the registers from EEPROM upon startup. Alternatively, the registers can be refreshed with the EEPROM values while still powered.

The device has a bit, 'eeRefresh', which loads the registers with the values in EEPROM. Setting 'eeRefresh' has the same effect as power cycling the device. After the bit is set to 1, the registers are loaded with the values stored in EEPROM, then the device clears the bit. Figure 5 shows the commands for setting eeRefresh.

To read data from the registers, the user must make a few changes from the writing process. Most importantly, the master must tell the slave to send information (see the command chain shown in Figure 6). Note that the user does not need to set the eeRefresh bit to read the registers; it is only for setting the registers back to the EEPROM values.

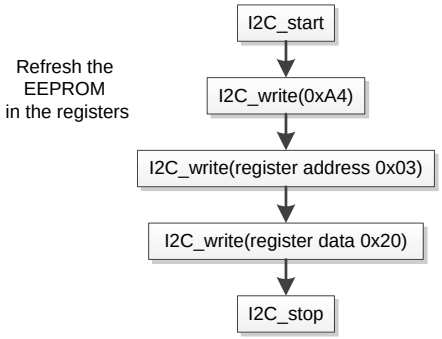


Figure 5. I²C Command Chain for Refreshing the Registers

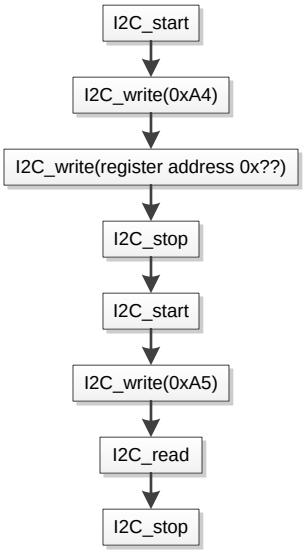


Figure 6. I²C Command Chain for Reading a Register

Figure 7 shows the SDA and SCL lines during a read command. The register 0x20 is read and shows a value of 0x39.

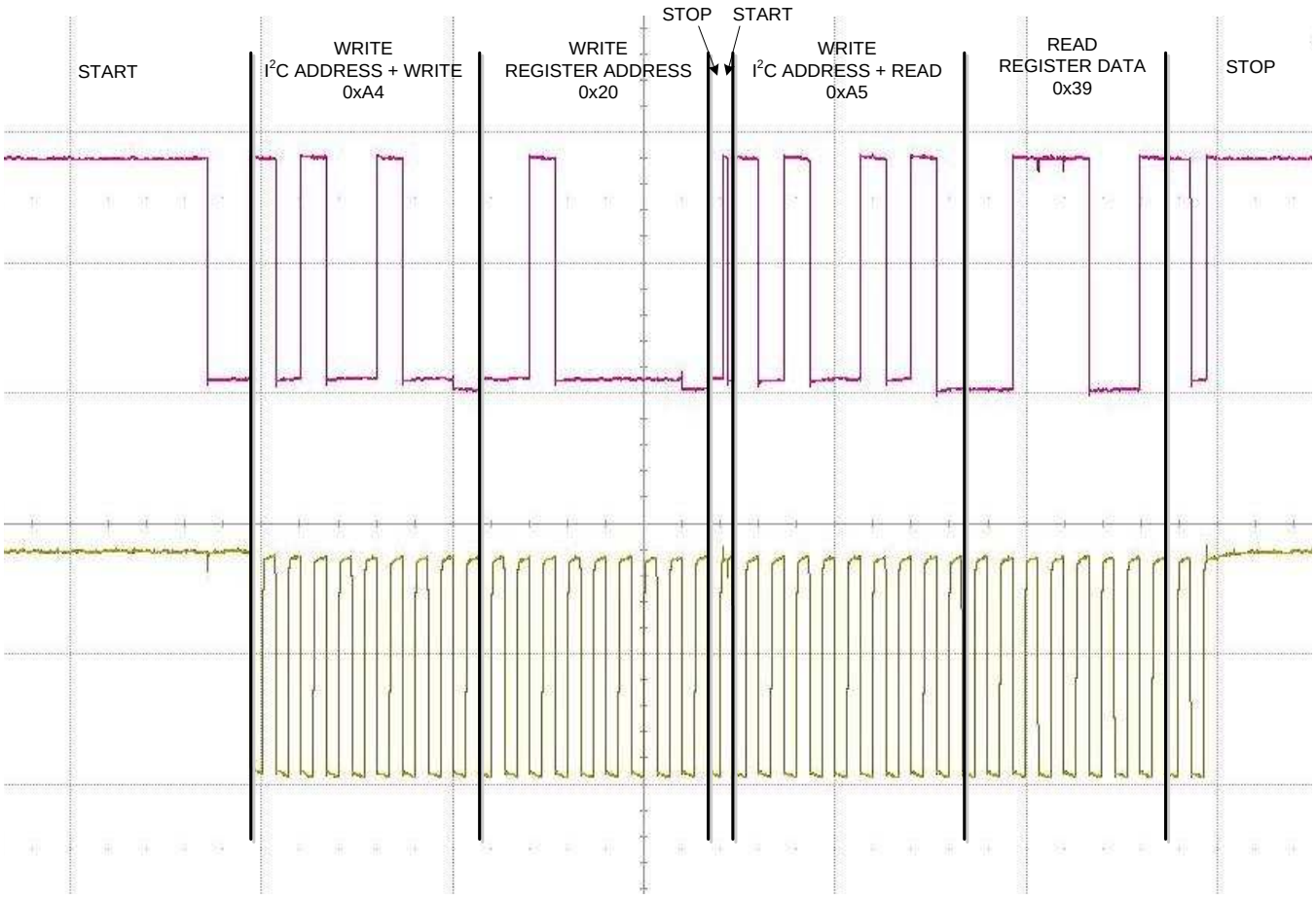


Figure 7. I²C Waveform of Reading a Register

2 Using I²C to Communicate With the DRV10983-Q1 and DRV10987 Motor Driver

The DRV10983-Q1 and DRV10987 uses the standard I²C protocol and is the slave device in the communicating pair. The slave address of the DRV10983-Q1 and DRV10987 is 1010 010. When a master addresses the motor driver, it sends 8 bits, the address and one bit specifying a write (0) or read (1). The master sends 1010 0100 (0xA4) for write and 1010 0101 (0xA5) for read. To program the device, first load the registers with the desired motor parameter values. Once the registers have the desired value in them, follow the next instructions to update the EEPROM with the contents of the registers.

2.1 Mass Write to EEPROM for DRV10983-Q1 and DRV10987

The procedure for programming the EEPROM is as shown in Figure 8. TI recommends performing the EEPROM programming without the motor spinning; power cycle after the EEPROM write, and read back the EEPROM to verify the programming is successful.

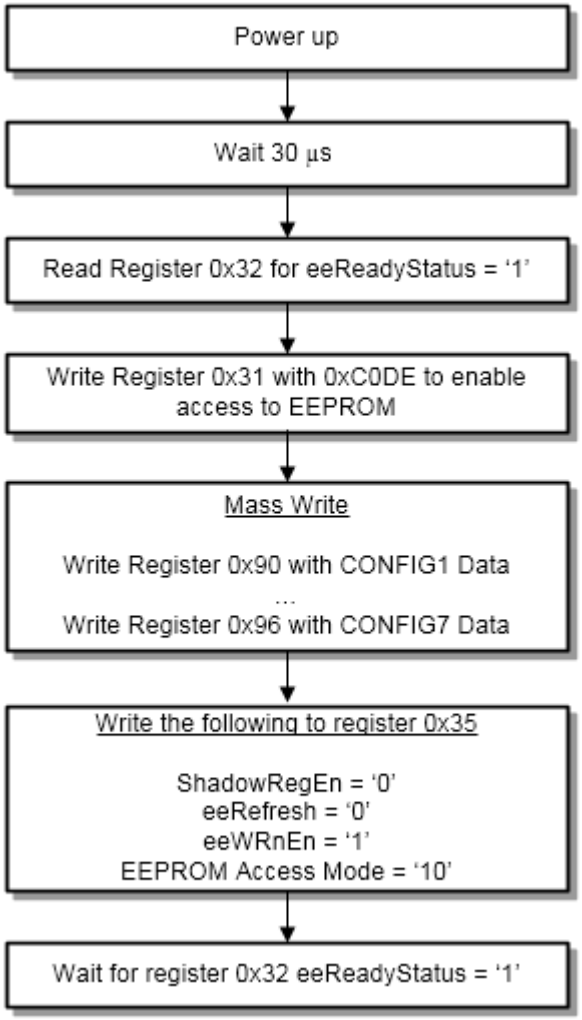


Figure 8. DRV10983-Q1 and DRV10987 Mass Write Protocol Using I²C

2.2 Mass Read From EEPROM for DRV10983-Q1 and DRV10987

Reading the registers is a way to verify the values saved in EEPROM or to obtain feedback from the DRV10983-Q1 and DRV10987. Because the EEPROM cannot be read directly in these two devices, the settings saved in EEPROM can be loaded into the registers by two methods. Power cycling the device auto loads the registers from EEPROM upon startup. Alternatively, the registers can be refreshed with the EEPROM values while still powered. See Figure 9 for the sequence to refresh the registers with the EEPROM values.

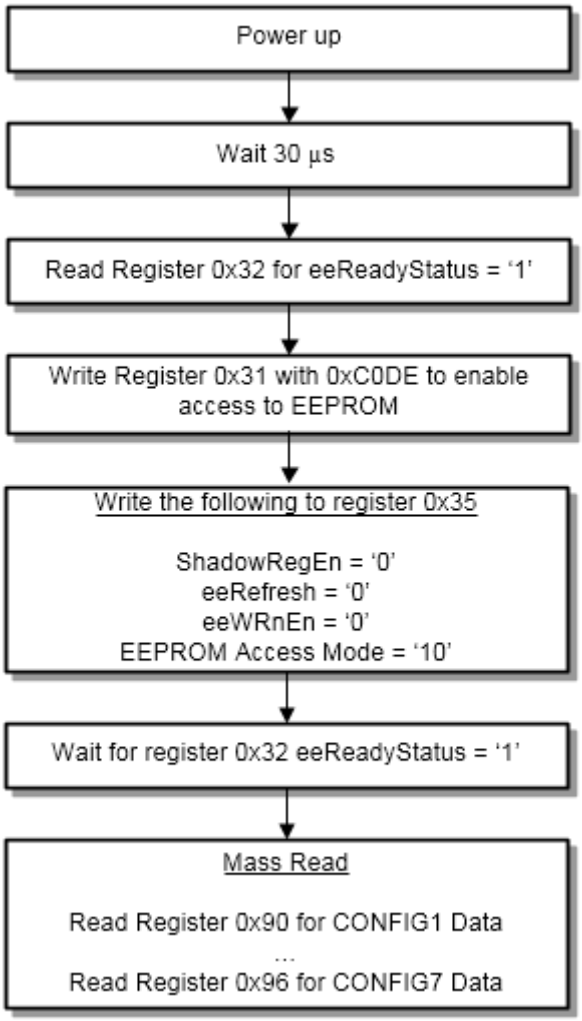


Figure 9. DRV10983-Q1 and DRV10987 Mass Read Protocol Using I²C

3 Introduction to the TIDA-01373

For a detailed description of the design behind the TIDA-01373, see the *DRV10983, DRV10975, DRV10983-Q1, and DRV10987 EEPROM Programming Tool Reference Design* (TIDUCX2).

The DRV10983, DRV10975, DRV10983-Q1, and DRV10987 are integrated brushless DC (BLDC) motor drivers with configurable *electrically erasable programmable read-only memory* (EEPROM) registers that need to be set to motor-specific parameters. The EEPROM registers can be written to and read from through I^2C protocol.

The TIDA-01373 is a hardware and software reference design for a DRV10983, DRV10975,DRV10983-Q1, and DRV10987 programming tool. The design is meant to be used as an example of how a scalable programming tool may be built in order to program the configurable registers of the DRV10983, DRV10975, DRV10983-Q1, or DRV10987 devices through the I^2C protocol described in the *I^2C Serial Interface* section in the data sheet of each device.

The hardware of the TIDA-01373 consists of two separate PCB boards. The first board is referred to as the *Programming Board*. The second is referred to as the *Socket Board*. The *Programming Board* consists of power management, a whole system ON/OFF switch, microcontroller (MCU), I^2C multiplexer (MUX), LCD screen, buttons, and isolation. The *Socket Board* consists of power management, and eight sockets in which the DRV10983, DRV10975, DRV10983-Q1, or DRV10987 motor driver is placed. The two boards are connected via two ribbon wires; one for the I^2C communication lines as well as an ON/OFF signal, and one for a ground connection. Overvoltage protection is built into both boards as well as isolation to protect both the motor drivers and the MCU in the system.

The design allows for a user to select custom motor parameter settings to program the DRV10983, DRV10975, DRV10983-Q1, or DRV10987 EEPROM registers by editing a header file found in the source code. The user manually inputs the parameters with the desired values that were generated when the motor was tuned. The source code is then compiled and flashed to the MSP430G2553 MCU on the *Programming Board*. The design offers a 2-wire JTAG interface to the MCU to reduce pin count. Once the MCU is flashed, the TIDA-01373 is a stand-alone system. The LCD on the *Programming Board* allows for user-friendly operation of the design.

In operation the TIDA-01373 has three general functions. The first function reads back the values that will be programmed into the EEPROM registers for the motor parameters of the DRV10983, DRV10975, DRV10983-Q1, or DRV10987. These values correspond to the values entered by the user into the header file in the source code. The second function reads the DRV10983, DRV10975, DRV10983-Q1, or DRV10987 devices that are in the system and verifies that the EEPROM registers in each of the devices match the desired values in the header file. This function will indicate which devices have EEPROM registers that match and which ones do not. The third function is the general programming routine. This function writes the desired motor parameter values that the user specified in the header file to the EEPROM registers in each of the populated DRV10983, DRV10975, DRV10983-Q1, or DRV10987 motor drivers on the *Socket Board*. This function then verifies that all of the devices were properly programmed and indicates back to the user if any errors occurred.

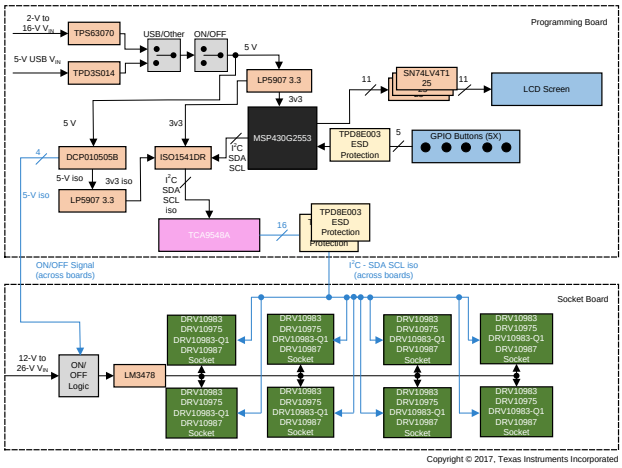


Figure 10. TIDA-0137 Block Diagram

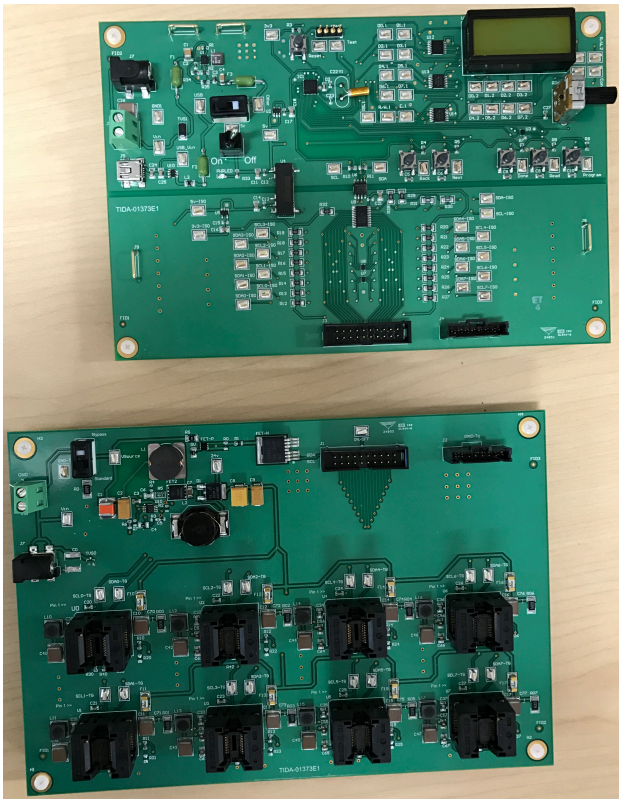


Figure 11. TIDA-01373 Board Image

4 Using the TIDA-01373 Hardware and Software for the DRV10983 and DRV10975 Devices

Using the TIDA-01373 is great way for users to begin understanding how to create a scalable programming tool for the DRV10983 and DRV10975 motor drivers. This section gives the details needed to operate the tool. Note that the TIDA-01373 reference design does not include any DRV10983 or DRV10975 integrated circuits (IC). These ICs must be ordered separately.

4.1 Hardware Modifications

Modify C30,C31,C32, C33, C34, C35, C36, and C37 from 10-nF capacitors to 0.1-µF capacitors.

The capacitor for one socket is depicted in Figure 12.

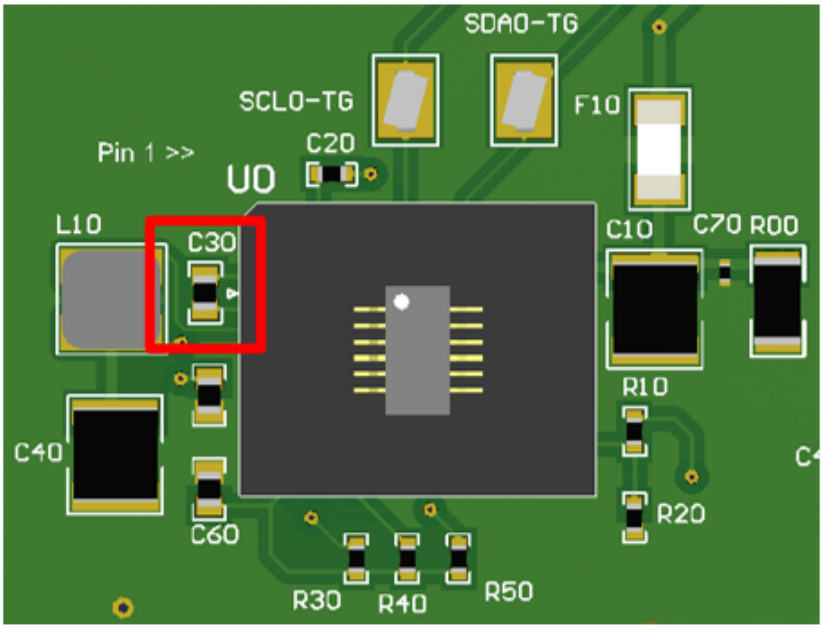


Figure 12. C30 Capacitor

4.2 Initial Setup (Done Once for a Set of Motor Parameters)

The initial setup of the TIDA-01373 system is only done one time each time the motor parameters are set. There are three steps for initial setup:

1. Hook up the hardware, see Section 4.2.1
2. Obtain, modify, and load the software, see Section 4.2.2
3. Power down, see Section 4.2.3

4.2.1 Hooking up the Hardware

When connecting the hardware, use the following steps **in the order** they are listed:

1. Initialize the switches, see Section 4.2.1.1
2. Connect the *Programming Board* to the *Socket Board*, see Section 4.2.1.2
3. Connect the power supply, see Section 4.2.1.3

A detailed explanation of each step is shown in the following three sections.

4.2.1.1 Initializing the Switches

- Programming Board: ON/OFF switch
  - Set the ON/OFF switch to the “OFF” position
- Programming Board: USB vs Other switch
  - If you are going to power the programming board with USB, set the switch to “USB” as shown in Figure 13
  - If you are going to power the programming board with an external power supply or wall socket, set the switch to the “Other” state, as shown in Figure 14
- Socket Board: Bypass and Standard switch
  - Set this switch to “Standard” *(the “Bypass” setting is for designs that did not use TI’s programming board for the solution)* as shown in Figure 15

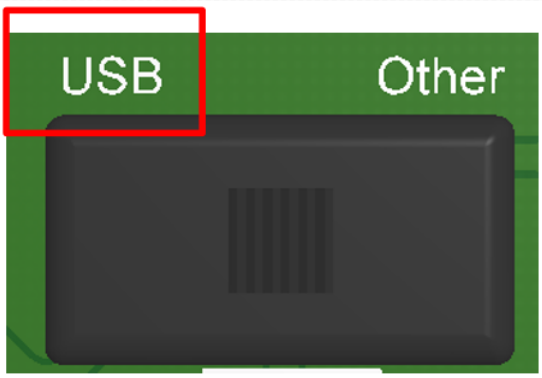


Figure 13. USB vs Other = 'USB'

Figure 14. USB vs Other = 'Other'

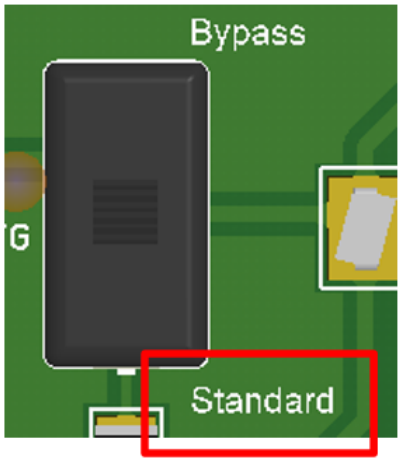


Figure 15. Bypass vs Standard = 'Standard'

4.2.1.2 Connecting the Programming Board to the Socket Board

Connect the two boards with two ribbon wires, the ribbon headers are shown in Figure 16.

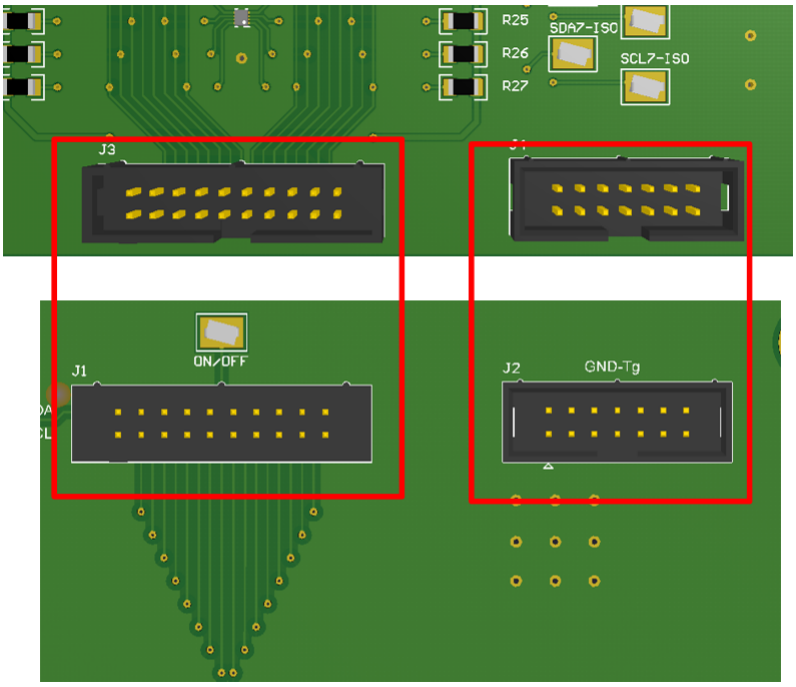


Figure 16. Ribbon Wire Headers on Both Boards (Top = Programming Board, Bottom = Socket Board)

4.2.1.3 Connecting the Power Supply

In the following order:

1. Connect power to the *Programming Board*
   - DC power supply
   - USB
2. Connect power to the *Socket Board*
   - DC power supply

4.2.2 Obtaining, Modifying, and Loading the Software for Custom Register Values

When setting up the software, use the following steps **in the order** they are listed:

1. Import the *Project*, see Section 4.2.2.1
2. Find the proper motor parameter values, see Section 4.2.2.2
3. Modify the software, see Section 4.2.2.3
4. Load the *Project* to the TIDA-01373, see Section 4.2.2.4

A detailed explanation of each step is shown in the following four sections.

4.2.2.1 Importing the Project

To run the provided programming example found at TIDA-01373, the user's PC must have CCS or an IDE that works with the MSP430™. The project developed for download was used in CCS and only tested to work with CCS.

The code used to program the DRV10983 or the DRV10975 with the MSP430G2553 can be downloaded from the TIDA-01373 product page. After the download completes, extract the files.

Import the *Project* downloaded into CCS through the project drop-down menu or the resource explorer welcome page.

4.2.2.2 Finding the Proper Motor Parameter Values

The software was written with register values for a specific motor that may or may not work for the end application. With that in mind, it is best to obtain the proper settings for the application and load those into the programming software.

1. Find the proper settings. The best way to get these settings is to follow the *DRV10983 and DRV10975 Evaluation Module* user's guide (SLOU393) and *DRV10983 and DRV10975 Tuning Guide* (SLOU395).
2. After the system is tuned, save the parameters using the save feature of the GUI. When saving the file, it does not matter what the name is, but remember where it is saved. Locate the parameters file (that was just saved) and open it. The file should contain 12 rows with register data for the device.

	A	B
1	DRV109830x200x39	
2	DRV109830x210x1E	
3	DRV109830x220x3A	
4	DRV109830x230x8	
5	DRV109830x240x50	
6	DRV109830x250xDA	
7	DRV109830x260x8B	
8	DRV109830x270x10	
9	DRV109830x280x27	
10	DRV109830x290x37	
11	DRV109830x2A0x4	
12	DRV109830x2B0xC	
13		

Figure 17. File Saved From DRV10983 GUI

4.2.2.3 Modifying the Software

Once the GUI file is generated, open the CCS Project, then the Register_Values.h file. Now manually enter in the desired register data from the GUI file into program code as shown in Figure 18.

Figure 18. Example Register Settings Entered into the Programming Tools Code for DRV10983

The explanation of an example follows:

When looking at the output file from the GUI, a cell may read like this:

DRV109830x200x39

This corresponds to the following information:

- DRV10983 = Device = DRV10983
- 0x20 = Register # = REG 0x20
- 0x39 = Data = 0x39

Now transfer the data to the Register_Values.h header file, manually.

It is important that we **only** change the **data** values in the header file.

Open the Header file and type the following:

#define REG_20_DATA 0x39. This is shown in Figure 18.

4.2.2.4 Loading the Project to the TIDA-01373 System

NOTE: If the *Programming Board* does not have a 7×2 header for the MSP-FET, then see Section 6 to learn how to load the program with an MSP430G2553 LaunchPad™.

Plug in the MSP-FET into your computer. The MSP-FET is found at the following link: http://www.ti.com/tool/MSP-FET. This page also provides information on how to properly use the MSP-FET.

Plug the MSP-FET into the TIDA-01373 Program Board JTAG header, as shown in Figure 19.

Figure 19. MSP-FET and TIDA-01373 System Diagram

Power on the TIDA-01373 by setting the ON/OFF switch to the 'ON' position.

Once all of the #define statements in the Register_Values.h file correctly correspond to the data produced in the GUI file, re-compile the CCS project and re-Flash the MSP430G2553. Now the software will write specific motor parameter data to the DRV10983 or DRV10975 device. Note that after the downloaded project is imported into the workspace in CCS, the project should reference all of the necessary header files from the path variables. Build, load, and run the code. For help running a project in CCS, see the web resources link in the welcome menu on the resource explorer page. Find this by clicking on the *TI Resource Explorer* under the *View* drop-down menu. If not using CCS, make sure all the necessary files are in the workspace for the project to compile correctly.

4.2.3 Power Down

Power off the TIDA-01373 by setting the ON/OFF switch to the 'OFF' position.

4.3 General Operation

In operation the TIDA-01373 has three general functions. The first function reads back the values that will be programmed into the EEPROM registers for the motor parameters of the DRV10983 or DRV10975 devices. These values correspond to the values entered by the user into the header file in the source code. The second function reads the DRV10983 or DRV10975 devices that are in the system and verifies that the EEPROM registers in each of the devices match the desired values in the header file. This function will indicate which devices have EEPROM registers that match and which ones do not. The third function is the general programming routine. This function writes the desired motor parameter values the user specified in the header file to the EEPROM registers in each of the populated DRV10983 or DRV10975 motor drivers on the *Socket Board*. This function then verifies that all of the devices were properly programmed and indicates back to the user if any errors occurred.

4.3.1 Hooking Up the Hardware

This section is similar to Section 4.2.1.

When hooking up the hardware, use the following steps **in the order** they are listed:

1. Initialize the switches, see Section 4.3.1.1
2. Connect the *Programming Board* to the *Socket Board*, see Section 4.3.1.2
3. Connect the power supply, see Section 4.3.1.3

4.3.1.1 Initializing the Switches

- Programming Board: ON/OFF switch
  - Set the ON/OFF switch to the “OFF” position
- Programming Board: USB vs Other switch
  - If you are going to power the programming board with USB, set the switch to “USB” as shown in Figure 20
  - If you are going to power the programming board with an external power supply or wall socket then set the switch to the “Other” state as shown in Figure 21
- Socket Board: Bypass and Standard switch
  - Set this switch to “Standard” *(the “Bypass” setting is for designs that did not use TI’s programming board for the solution)* as shown in Figure 22

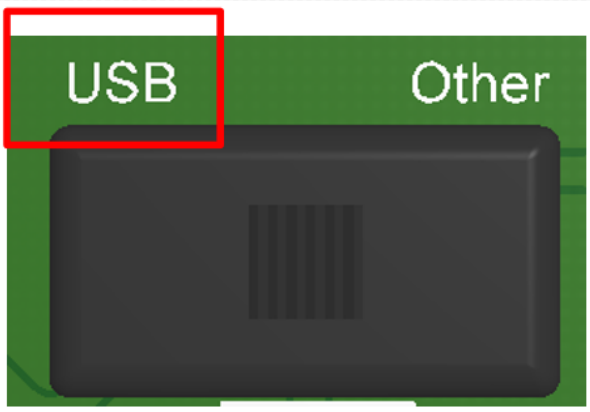


Figure 20. USB vs Other = 'USB'

Figure 21. USB vs Other = 'Other'

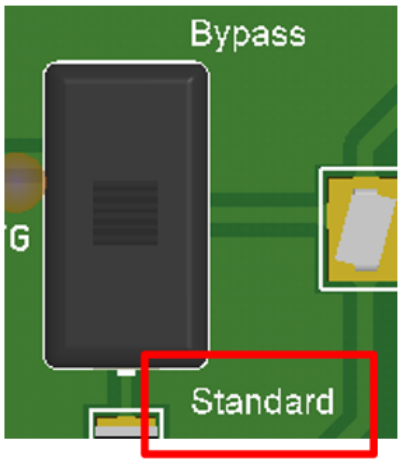


Figure 22. Bypass vs Standard = 'Standard'

4.3.1.2 Connecting the Programming Board to the Socket Board

Connect the two boards with the two ribbon wires shown in Figure 23.

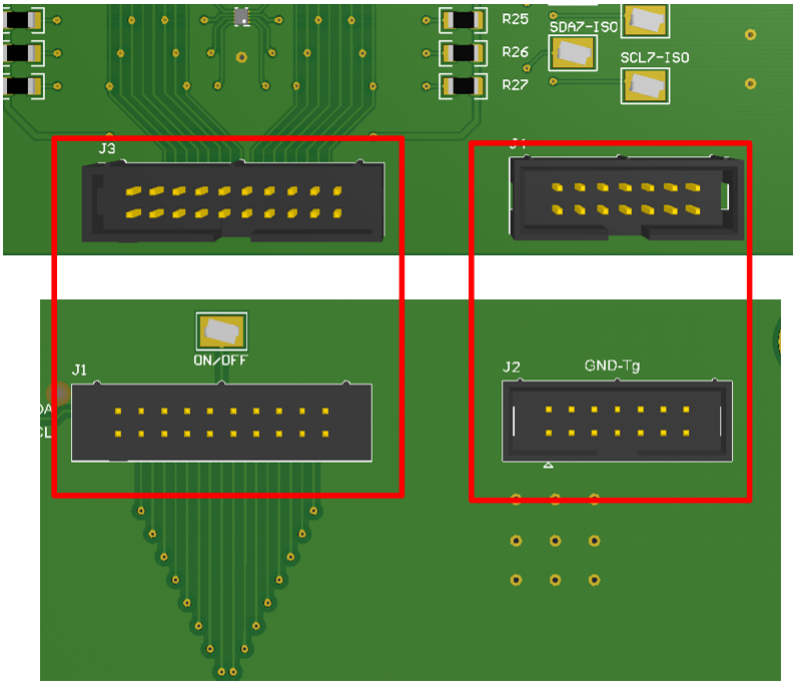


Figure 23. Ribbon Wire Headers on Both Boards

4.3.1.3 Connecting the Power Supply

In the following order:

1. Connect power to the *Programming Board*
   - DC power supply
   - USB
2. Connect power to the *Socket Board*

- DC power supply

4.3.2 Using the TIDA-01373 (for DRV10983 and DRV10975)

4.3.2.1 Populating the Sockets

Make sure the ON/OFF switch is set to 'OFF'.

Carefully place the devices in the sockets with pin 1 in the top left corner as shown in Figure 24.

A second image (Figure 25) shows how the sockets are numbered in the system. This shows in what order the sockets are sequentially programmed and read from.

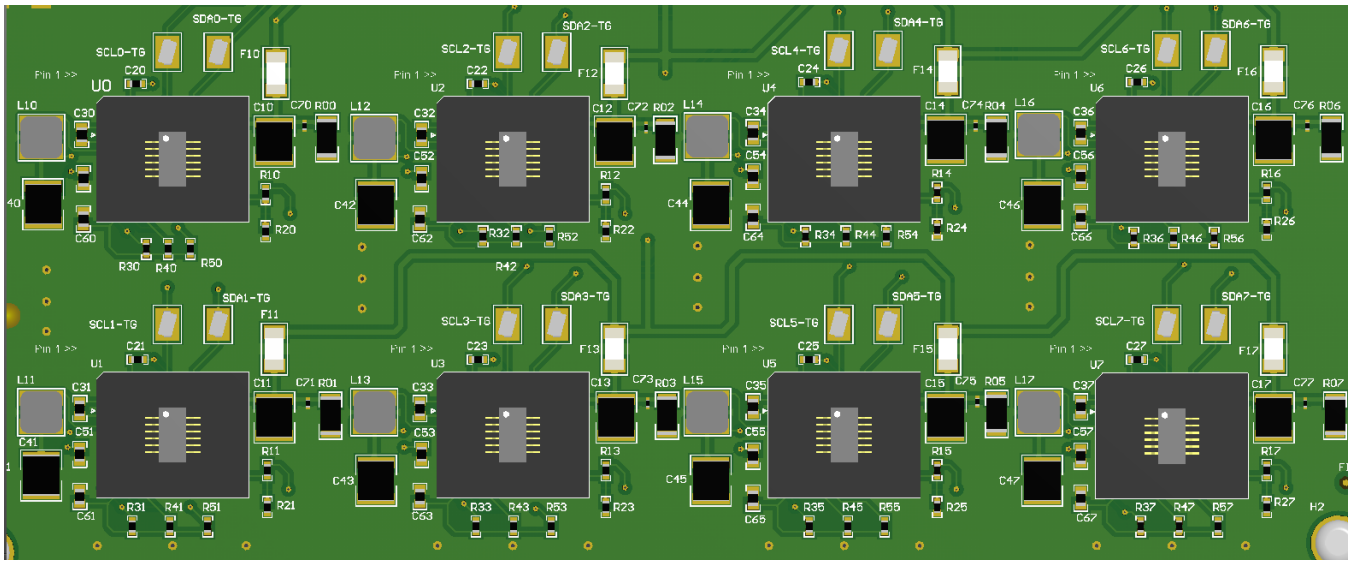


Figure 24. Sockets With Device Populated in Proper Orientation

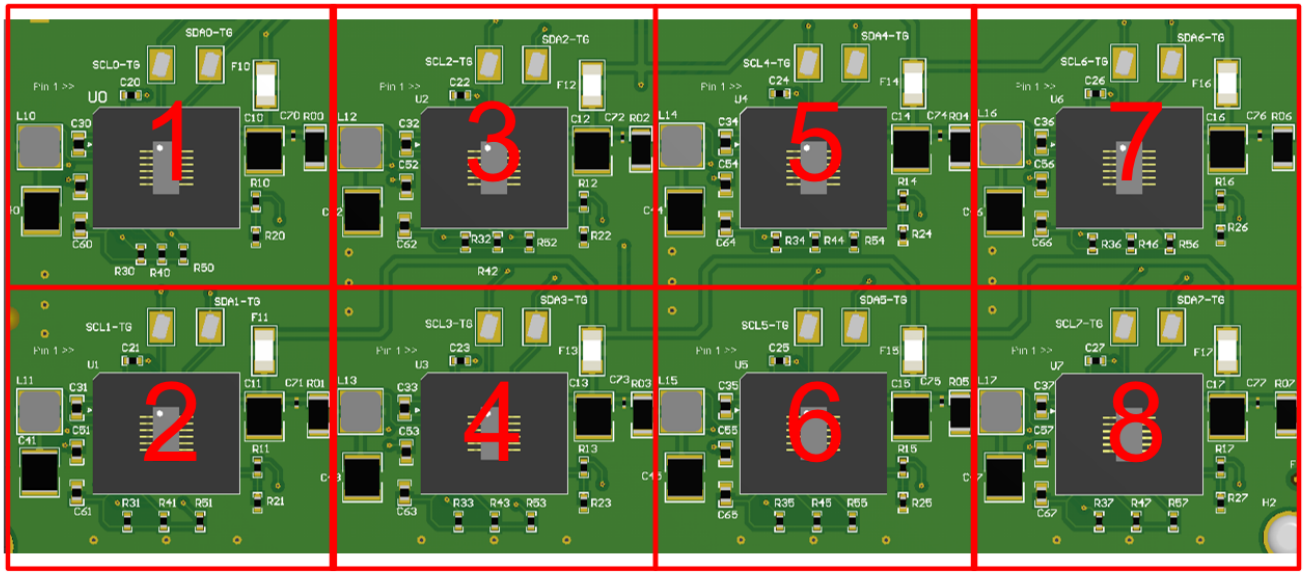


Figure 25. Sockets With Device Numbered

4.3.2.2 Powering Up the TIDA-10373

Set the ON/OFF switch to 'ON'.

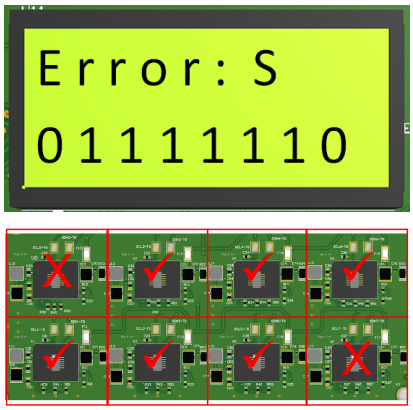
Once on, the software immediately checks to see if all the sockets are populated.

If all the sockets are populated then “Ready” is displayed on the LCD screen.

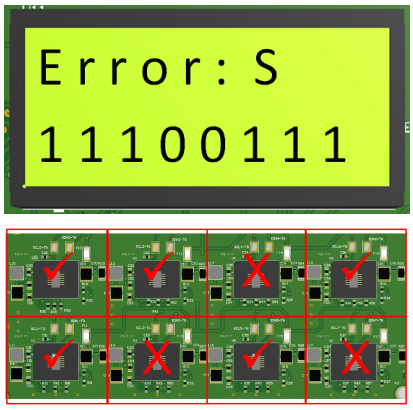
If not all the sockets are populated then the LCD screen displays “Error: S” as well as indicating which sockets are populated or not by writing a '1' if it is populated or a '0' if the socket is not populated. For example:

- 01111110 indicates sockets 1 and 8 are not populated
- 11100111 indicates sockets 4 and 5 are not populated
- 11011011 indicates sockets 3 and 6 are not populated
- ...and so on

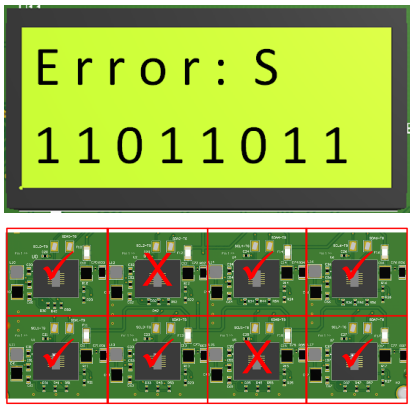
Error: S 01111110



Error: S 11100111



Error: S 11011011



If 8 devices have been placed in the socket and the system still gives this error message, it means that one of the devices was incorrectly placed in the socket; either the leads are not all properly touching or it was place in upside-down. If you wish to replace the IC in the socket, then power down the TIDA-01373 with the ON/OFF switch by switching it to 'OFF' and start back at Section 4.3.2.1.

If the result of this error message is okay and the number of sockets placed on the *Socket Board* is desired, then press the **Next** button and continue. The software will remember what sockets are populated and what socket are not populated and will only write and read to the sockets that have a device in them. Once the **Next** button is pressed “Ready” should be displayed on the screen.

4.3.2.3 Choosing What to Do

- Back button, see Section 4.3.2.3.1
- Next button, see Section 4.3.2.3.2
- Done button, see Section 4.3.2.3.3
- Read button, see Section 4.3.2.3.4
- Program button, see Section 4.3.2.3.5

4.3.2.3.1 Pressing the Back Button

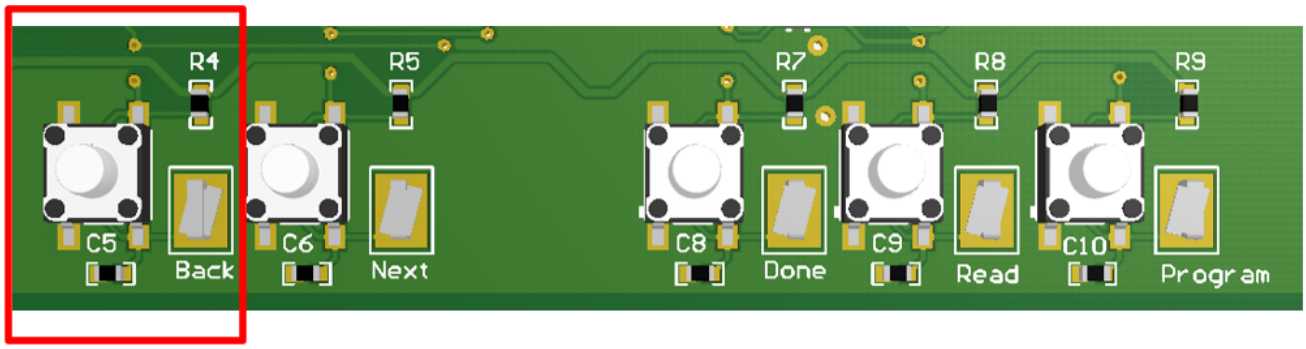


Figure 26. Back Button

In the initial state machine, pressing the **Back** button does nothing, pressing this button will not result in any action taken in the software. Stay in Section 4.3.2.3.

4.3.2.3.2 Pressing the Next Button

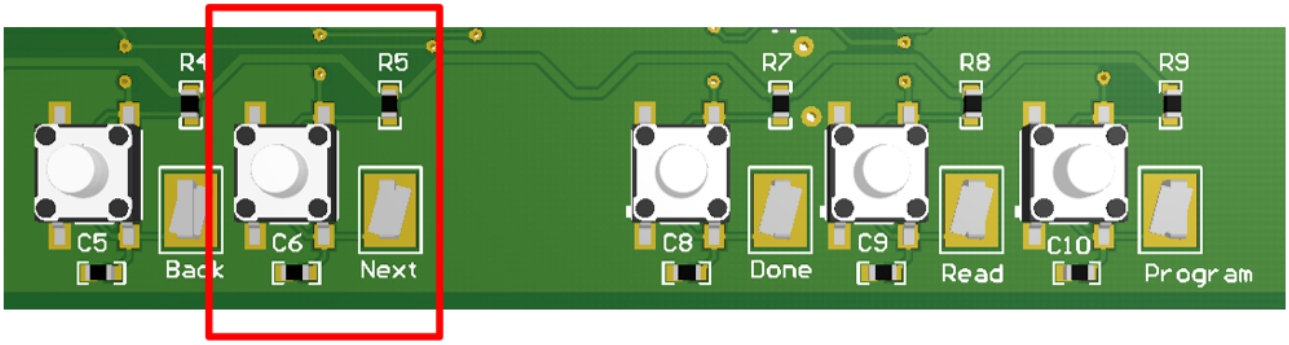


Figure 27. Next Button

Pressing the **Next** button will allow the user to read back the values in the header file. These are the values that the MCU will be programming into the device and the user may wish to verify.

First “Read Header” will display on the LCD as seen in Figure 28, click the **Next** button again to continue.

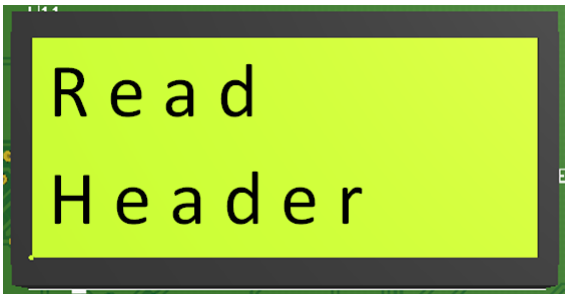


Figure 28. LCD: Read Header

Once the **Next** button is clicked, the device will begin to read back the desired register values that are located in the header file starting with “REG 0x20 0xdata”. The user will continue to click the **Next** button to cycle through all 12 configuration registers. An example is depicted in Figure 29.

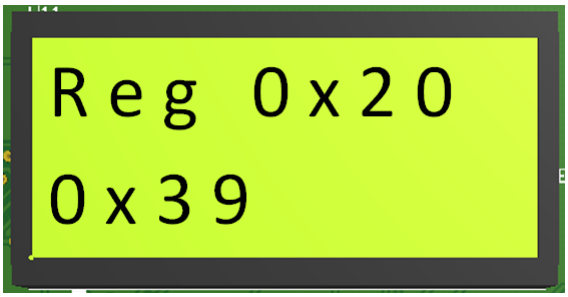


Figure 29. LCD: Reg 0x20 - 0x39

Once all registers have been read, the software will display “Ready” again. Stay in Section 4.3.2.3.

4.3.2.3.3 Pressing the Done Button

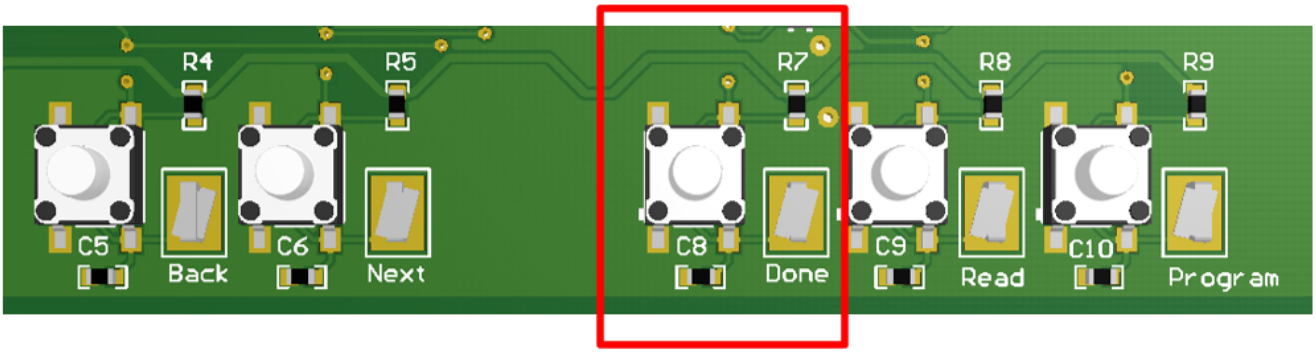


Figure 30. Done Button

Pressing the **Done** button will put the MCU in a low power mode and enter an infinite while loop. Once this button is pressed you can power down the system by setting the ON/OFF switch to the 'OFF' State. Move to Section 4.3.2.4.

4.3.2.3.4 Pressing the Read Button

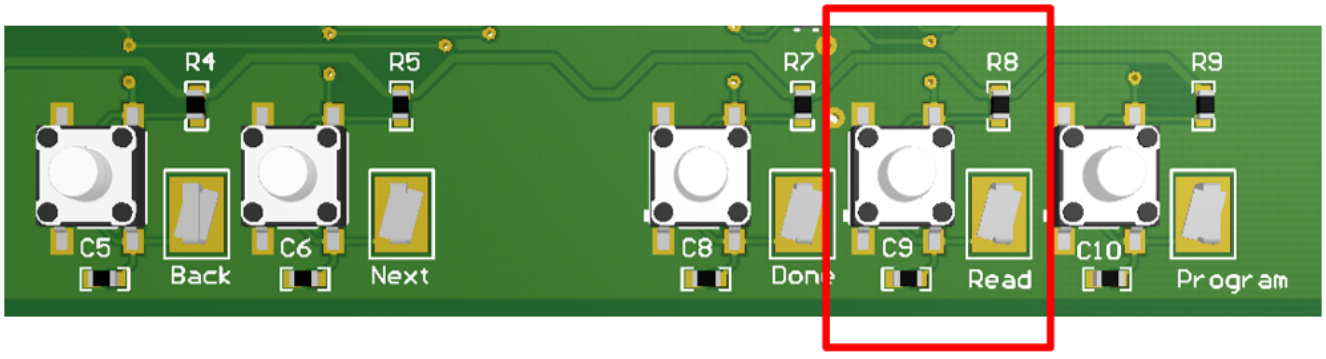


Figure 31. Read Button

Pressing the **Read** button results in the software reading the 12 configuration registers of each device in each of the populated sockets. Once read, the software will compare them to the desired values indicated in the header file.

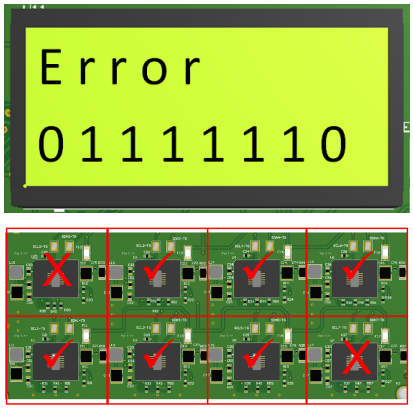
If all configuration registers on each device in the populated sockets match the desired values in the header file, then “All Regs Match!” will display on the LCD, shown in Figure 32. Pressing the **Next** button will return you to Section 4.3.2.3 and “Ready” will display again.

Figure 32. LCD: All Regs Match!

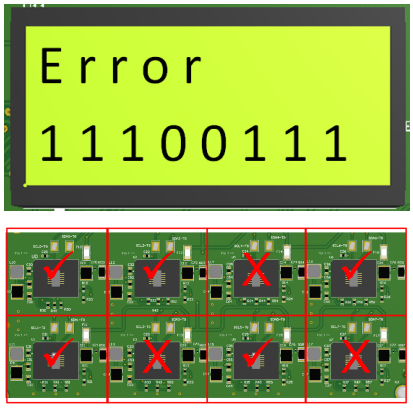
If there is a discrepancy, then the LCD will display “Error” as well as indicate which device has registers that do not match the desired ones in the header. For example:

- 01111110 indicates sockets 1 and 8 are not programmed according to the header file
- 11100111 indicates sockets 4 and 5 are not programmed according to the header file
- 11011011 indicates sockets 3 and 6 are not programmed according to the header file
- ... and so on

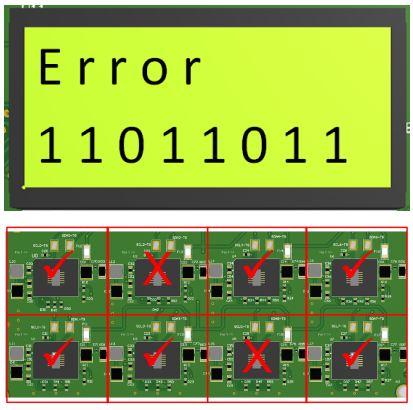
Error: 01111110



Error: 11100111

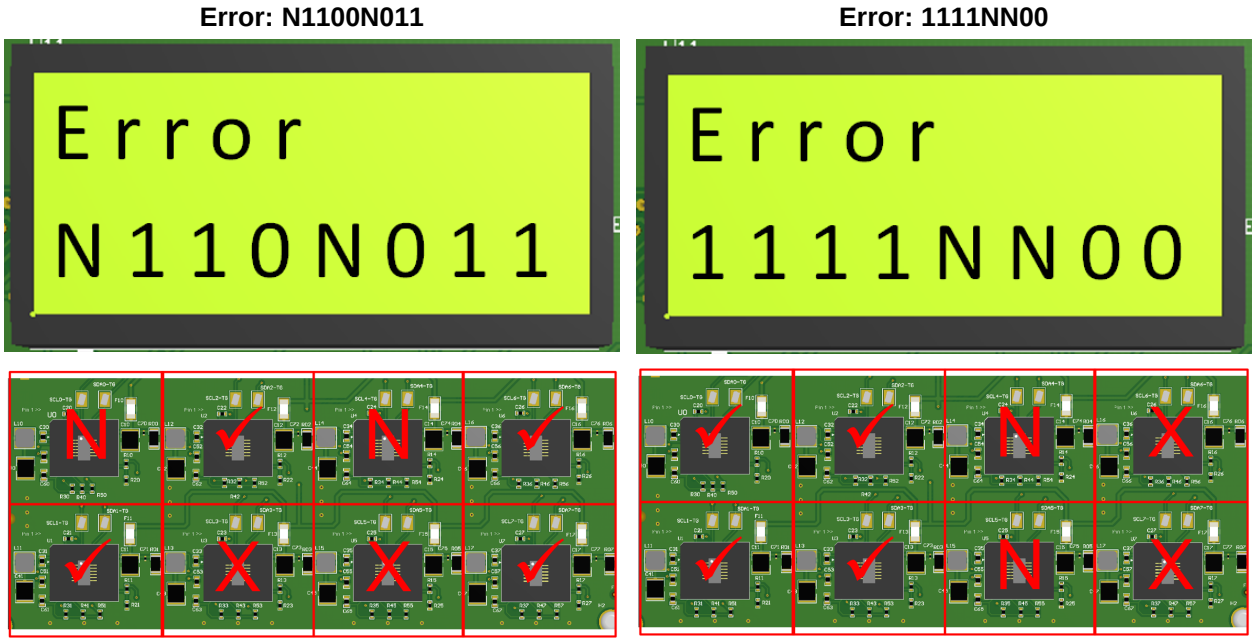


Error: 11011011



This will also show which sockets are not populated by displaying “N” in that location. For example:

- N110N011 indicates sockets 1 and 5 are not populated and sockets 4 and 6 are not programmed according to the header file
- 1111NN00 indicates sockets 5 and 6 are not populated and sockets 7 and 8 are not programmed according to the header file
- ... and so on



After displaying this error message, the user can press the **Next** button again and the screen will display "Try Again Ready" as shown in Figure 33. Pressing the **Next** button will return you to Section 4.3.2.3 and "Ready" will display.

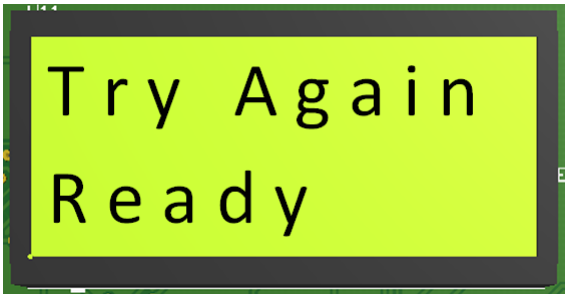


Figure 33. LCD: Try Again Ready

The user may also chose to go to Section 4.3.2.4 and begin troubleshooting.

4.3.2.3.5 Pressing the Program Button

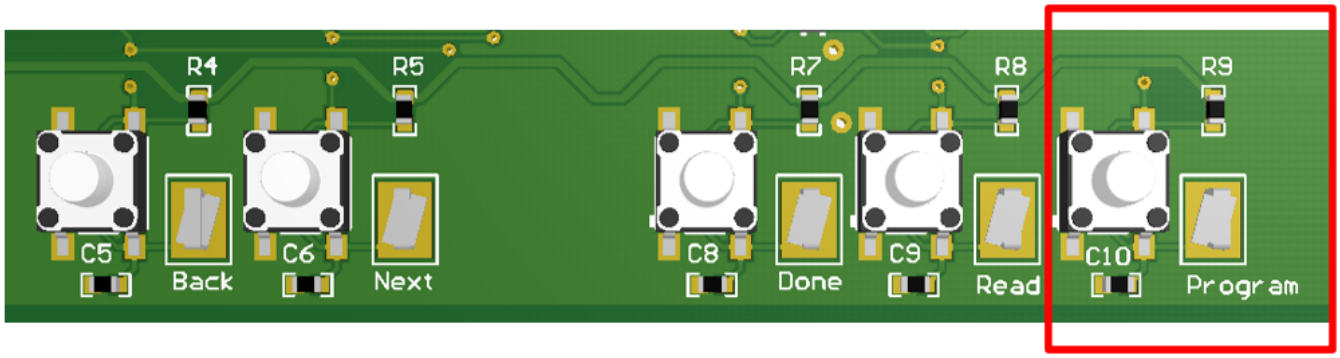


Figure 34. Program Button

Pressing the **Program** button will result in the software writing the desired values in the header into the configuration registers of each populated device. This set up will also run the routine in Section 4.3.2.3.4. Once finished, the LCD screen will display “Complete” as shown in Figure 35. At this point the user may move to Section 4.3.2.4.

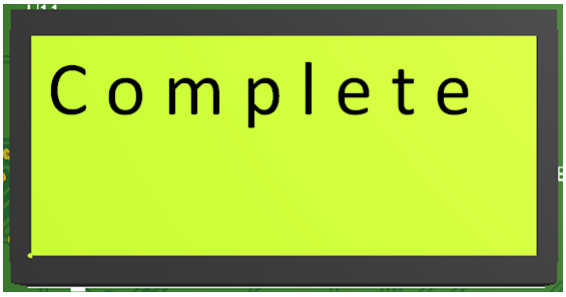


Figure 35. LCD: Complete

4.3.2.4 Powering Down the TIDA-10373

Power off the TIDA-01373 by setting the ON/OFF switch to the 'OFF' position. Move to Section 4.3.2.5.

4.3.2.5 Depopulating the Sockets

Make sure the ON/OFF switch is set to 'OFF'. Carefully remove the devices in the sockets.

For repetitive use, now return to Section 4.3.2.1.

5 Using the TIDA-01373 Hardware and Software for the DRV10983-Q1 and DRV10987

Using the TIDA-01373 is great way for users to begin understanding how to create a scalable programming tool for the DRV10983-Q1 and DRV10987 motor drivers. This section gives the details needed to operate the tool. Note that the TIDA-01373 reference design does not include any DRV10983-Q1 or DRV10987 ICs. These ICs must be ordered separately.

5.1 Hardware Modifications

Modify C30, C31, C32, C33, C34, C35, C36, and C37 from 0.1-µF capacitors to 10-nF capacitors. The capacitor for one socket is depicted in Figure 36.

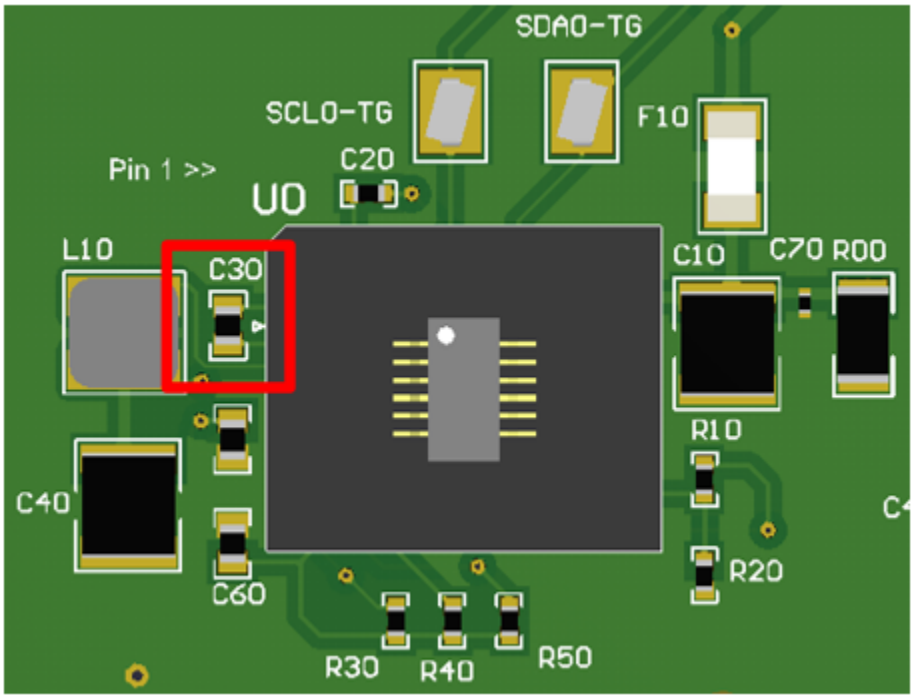


Figure 36. C30 Capacitor

5.2 Initial Setup (Done Once for a Set of Motor Parameters)

The initial setup of the TIDA-01373 system is only done one time, each time the motor parameters are set. There are three steps for initial setup:

1. Hook up the hardware, see Section 5.2.1
2. Obtain, modify, and load the software, see Section 5.2.2
3. Power down, see Section 5.2.3

5.2.1 Hooking Up the Hardware

When connecting the hardware, use the following steps **in the order** they are listed:

1. Initialize the switches, see Section 5.2.1.1
2. Connect the *Programming Board* to the *Socket Board*, see Section 5.2.1.2
3. Connect the power supply, see Section 5.2.1.3

5.2.1.1 Initializing the Switches

- Programming Board: ON/OFF switch
  - Set the ON/OFF switch to the "OFF" position
- Programming Board: USB vs Other switch
  - If you are going to power the programming board with USB, set the switch to “USB” as shown in Figure 37
  - If you are going to power the programming board with an external power supply or wall socket, then set the switch to the “Other” state as shown in Figure 38
- Socket Board: Bypass and Standard switch
  - Set this switch to “Standard” (the “Bypass” setting is for designs that did not use TI’s programming board for the solution) as shown in Figure 39

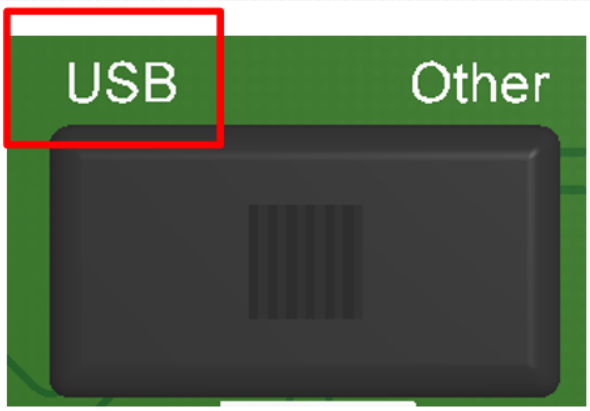


Figure 37. USB vs Other = 'USB'

Figure 38. USB vs Other = 'Other'

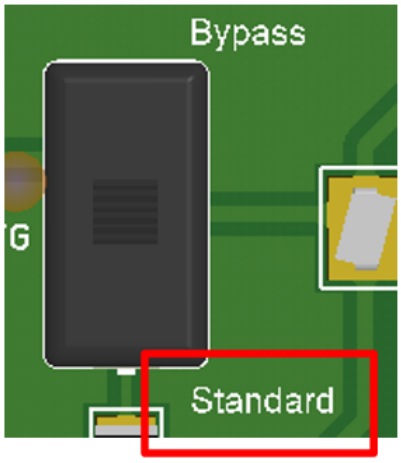


Figure 39. Bypass vs Standard = 'Standard'

5.2.1.2 Connecting the Programming Board to the Socket Board

Connect the two boards with two ribbon wires, the ribbon headers are shown in Figure 40.

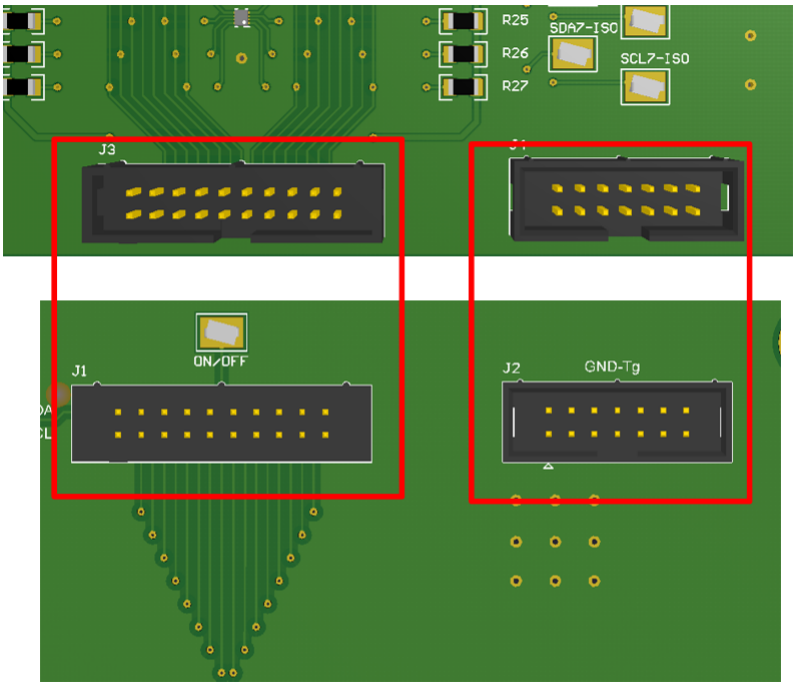


Figure 40. Ribbon Wire Headers on Both Boards (Top = Programming Board, Bottom = Socket Board)

5.2.1.3 Connecting the Power Supply

In the following order:

1. Connect power to the *Programming Board*
   - DC power supply
   - USB
2. Connect power to the *Socket Board*
   - DC power supply

5.2.2 Obtaining, Modifying, and Loading the Software For Custom Register Values

When setting up the software, use the following steps **in the order** they are listed:

1. Import the *Project*, see Section 5.2.2.1
2. Find the proper Motor Parameter Values, see Section 5.2.2.2
3. Modify the software, see Section 5.2.2.3
4. Load the *Project* to the TIDA-01373, see Section 5.2.2.4

5.2.2.1 Importing the Project

To run the provided programming example found at TIDA-01373, the user's PC must have CCS or an IDE that works with the MSP430. The project developed for download was used in CCS and only tested to work with CCS.

The code used to program the DRV10983-Q1 and DRV10987 with the MSP430G2553 can be downloaded from the TIDA-01373 product page. After the download completes, extract the files.

Import the *Project* downloaded into CCS through the project drop-down menu or the resource explorer welcome page.

5.2.2.2 Finding the Proper Motor Parameter Values

The software was written with register values for a specific motor that may or may not work for the end application. With that in mind, it is best to obtain the proper settings for the application and load those into the programming software.

1. Find the proper settings. The best way to get these settings is to follow the DRV10983-Q1 EVM User's Guide (SLVUAV1) or DRV10987 EVM User's Guide (SLOU476), and *DRV10983-Q1 Tuning Guide* (SLVUAV9) or *DRV10987 Tuning Guide* (SLOU447).
2. After the system is tuned, save the parameters using the save feature of the GUI. When saving the file, it does not matter what the name is, but remember where it is saved. Locate the parameters file (that was just saved) and open it. The file should contain 7 rows with register data for the device.

	A	B	C
1	DRV10983Q10x900x0123		
2	DRV10983Q10x910x4567		
3	DRV10983Q10x920x89AB		
4	DRV10983Q10x930xCDE1		
5	DRV10983Q10x940x0246		
6	DRV10983Q10x950x8ACE		
7	DRV10983Q10x960x159D		

Figure 41. Example File Saved From DRV10983-Q1 GUI

5.2.2.3 Modifying the Software

Once the GUI file is generated, open the CCS Project, then the Register_Values.h file. Now manually enter in the desired register data from the GUI file into program code, as shown in Figure 42.

Figure 42. Example Register Settings Entered Into the Programming Tools Code for DRV10983-Q1

5.2.2.4 Loading the Project to the TIDA-01373 System

NOTE: If the *Programming Board* does not have a 7×2 header for the MSP-FET, then see Section 6 to learn how to load the program with an MSP430G2553 LaunchPad.

Plug in the MSP-FET into your computer. The MSP-FET is found at the following link: http://www.ti.com/tool/MSP-FET. This page will also provide information on how to properly use the MSP-FET.

Plug the MSP-FET into the TIDA-01373 Program Board JTAG header, as shown in Figure 43.

Figure 43. MSP-FET and TIDA-01373 System Diagram

Power on the TIDA-01373 by setting the ON/OFF switch to the 'ON' position.

Once all of the #define statements in the Register_Values.h file correctly correspond to the data produced in the GUI file, re-compile the CCS project and re-Flash the MSP430G2553. Now the software will write specific motor parameter data to the DRV10983-Q1 or DRV10987 device. Note that after the downloaded project is imported into the workspace in CCS, the project should reference all of the necessary header files from the path variables. Build, load, and run the code. For help running a project in CCS, see the web resources link in the welcome menu on the resource explorer page. Find this by clicking on the TI Resource Explorer under the View drop-down menu. If not using CCS, make sure all the necessary files are in the workspace for the project to compile correctly.

5.2.3 Power Down

Power off the TIDA-01373 by setting the ON/OFF switch to the 'OFF' position.

5.3 General Operation

In operation the TIDA-01373 has three general functions. The first function reads back to the user the values that will be programmed into the EEPROM registers for the motor parameters of the DRV10983-Q1 or DRV10987 device. These values correspond to the values entered by the user into the header file in the source code. The second function reads the DRV10983-Q1 or DRV10987 devices that are in the system and verifies that the EEPROM registers in each of the devices match the desired values in the header file. This function will indicate which devices have EEPROM registers that match and which ones do not. The third function is the general programming routine. This function writes the desired motor parameter values that the user specified in the header file to the EEPROM registers in each of the populated DRV10983-Q1 or DRV10987 motor drivers on the *Socket Board*. This function then verifies that all of the devices were properly programmed and indicates back to the user if any errors occurred.

5.3.1 Hooking Up the Hardware

This is similar to Section 5.2.1.1.

When hooking up the hardware, use the following steps **in the order** they are listed:

1. Initialize the switches, see Section 5.3.1.1
2. Connect the *Programming Board* to the *Socket Board*, see Section 5.3.1.2
3. Connect the power supply, see Section 5.3.1.3

5.3.1.1 Initializing the Switches

- Programming Board: ON/OFF switch
  - Set the ON/OFF switch to the “OFF” position
- Programming Board: USB vs Other switch
  - If you are going to power the programming board with USB, set the switch to “USB” as shown in Figure 44
  - If you are going to power the programming board with an external power supply or wall socket then set the switch to the “Other” state as shown in Figure 45
- Socket Board: Bypass and Standard switch
  - Set this switch to “Standard” (the “Bypass” setting is for designs that did not use TI’s programming board for the solution) as shown in Figure 46

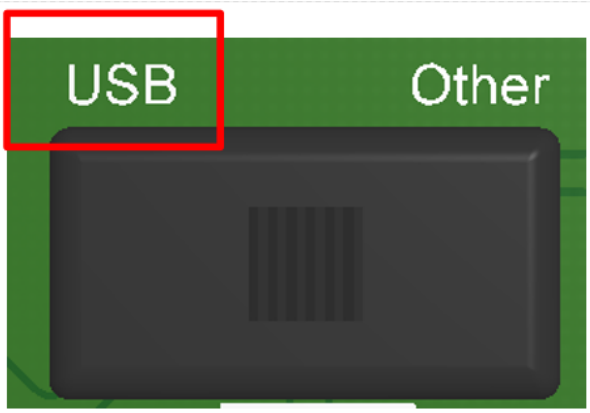


Figure 44. USB vs Other = 'USB'

Figure 45. USB vs Other = 'Other'

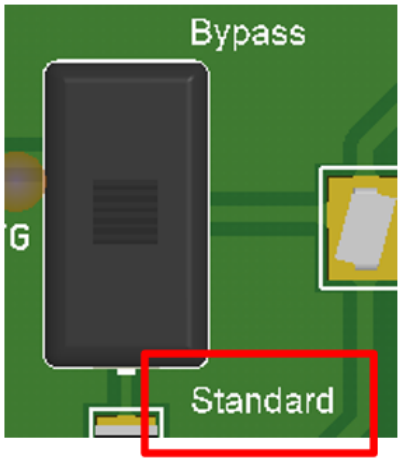


Figure 46. Bypass vs Standard = 'Standard'

5.3.1.2 Connecting the Programming Board to the Socket Board

Connect the two boards with two ribbon wires shown in Figure 47.

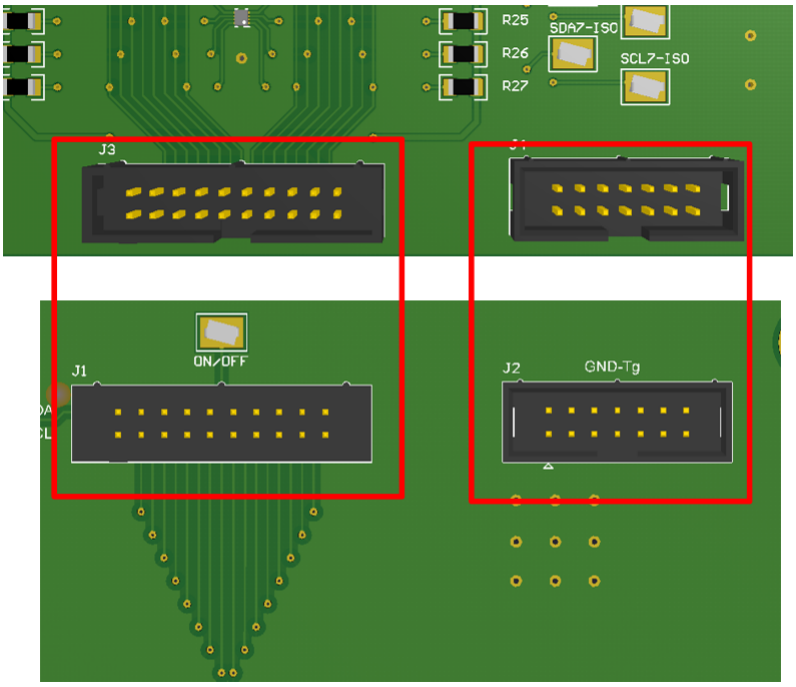


Figure 47. Ribbon Wire Headers on Both Boards

5.3.1.3 Connecting the Power Supply

In the following order:

1. Connect power to the *Programming Board*
   - DC power supply
   - USB
2. Connect power to the *Socket Board*
   - DC power supply

5.3.2 Using the TIDA-01373 (for DRV10983-Q1 and DRV10987)

5.3.2.1 Populating the Sockets

Make sure the ON/OFF switch is set to 'OFF'. Carefully place the devices in the sockets with pin 1 in the top left corner as shown in Figure 48.

A second image (Figure 49) shows how the sockets are numbered in the system. This shows in what order the sockets are sequentially programmed and read from.

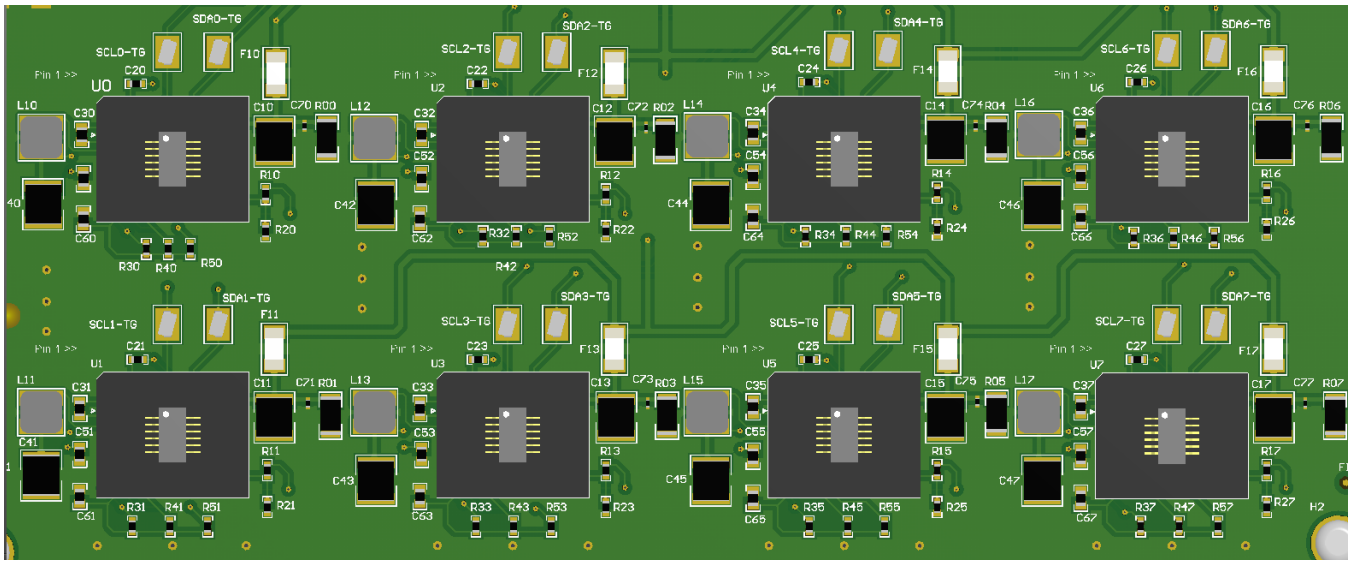


Figure 48. Sockets With Device Populated in Proper Orientation

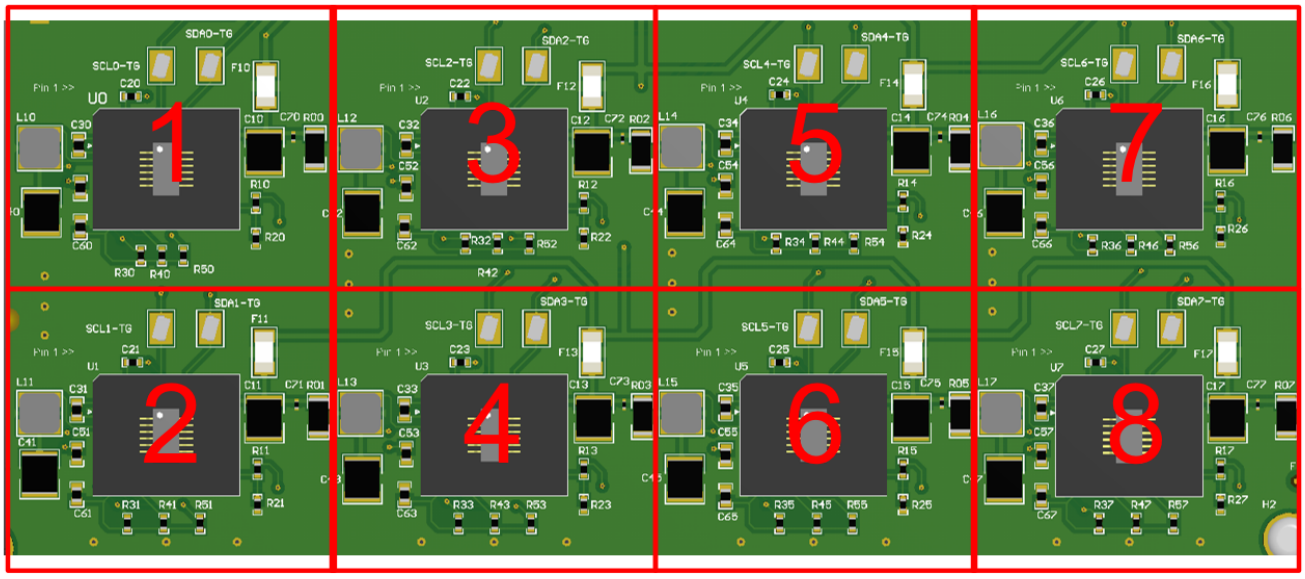


Figure 49. Sockets With Device Numbered

5.3.2.2 Powering Up the TIDA-01373

Set the ON/OFF switch to 'ON'.

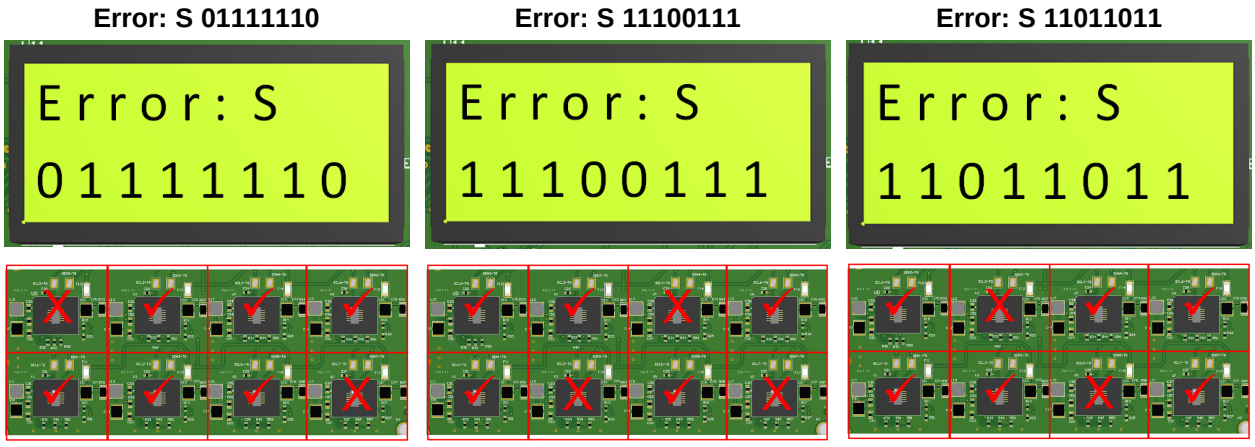
Once on, the software immediately checks to see if all the sockets are populated.

If all the sockets are populated then “Ready” is displayed on the LCD screen.

If not all the sockets are populated, then the LCD screen displays “Error: S” as well as indicating which sockets are populated or not by writing a '1' if it is populated or a '0' if the socket is not populated. For example:

- 01111110 indicates sockets 1 and 8 are not populated
- 11100111 indicates sockets 4 and 5 are not populated

- 11011011 indicates sockets 3 and 6 are not populated
- ...and so on



If 8 devices have been placed in the socket and the system still gives this error message, it means that one of the devices was incorrectly placed in the socket; either the leads are not all properly touching or it was place in upside-down. If you wish to replace the IC in the socket then power down the TIDA-01373 with the ON/OFF switch by switching it to 'OFF' and start back at Section 5.3.2.1.

If the result of this error message is okay and the number of sockets placed on the *Socket Board* is desired, then press the **Next** button and continue. The software remembers what sockets are populated and not populated and will only write and read to the sockets that have a device in them. Once the **Next** button is pressed, “Ready” should be displayed on the screen.

5.3.2.3 Choosing What To Do

- Back button, see Section 5.3.2.3.1
- Next button, see Section 5.3.2.3.2
- Done button, see Section 5.3.2.3.3
- Read button, see Section 5.3.2.3.4
- Program button, see Section 5.3.2.3.5

5.3.2.3.1 Pressing the Back Button

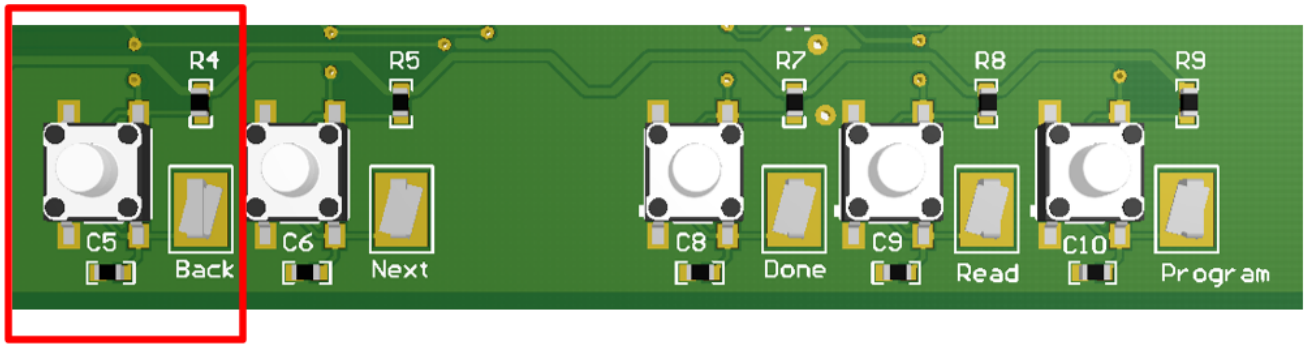


Figure 50. Back Button

In the initial state machine, pressing the **Back** button does nothing, pressing this button will not result in any action taken in the software. Stay in Section 5.3.2.3.

5.3.2.3.2 Pressing the Next Button

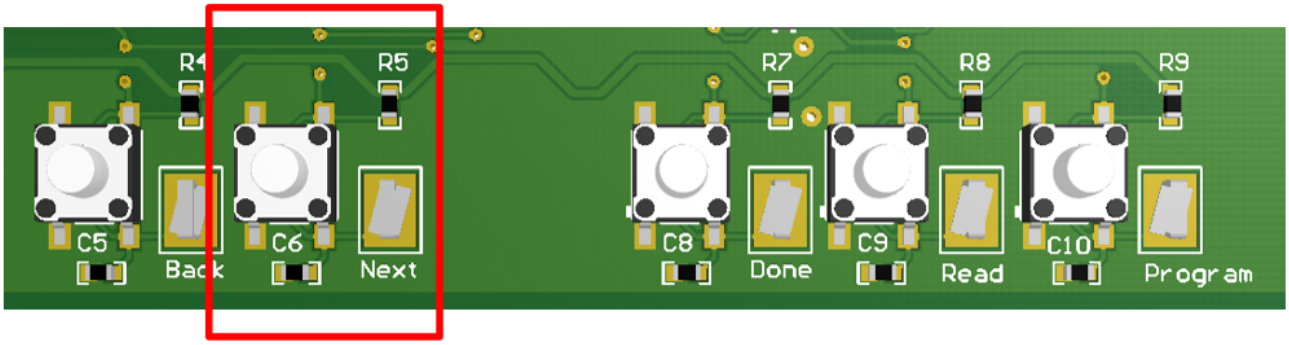


Figure 51. Next Button

Pressing the **Next** button will allow the user to read back the values in the header file. These are the values that the MCU will be programming into the device and the user may wish to verify.

First “Read Header” will display on the LCD as in Figure 52, click the **Next** button again to continue.

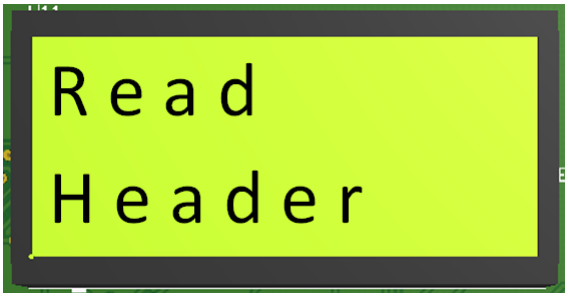


Figure 52. LCD: Read Header

Once the **Next** button is clicked, the device will begin to read back the desired register values that are located in the header file starting with “REG 0x90 0xdata”. The user will continue to click the **Next** button to cycle through all 12 configuration registers. An example is depicted in Figure 53.

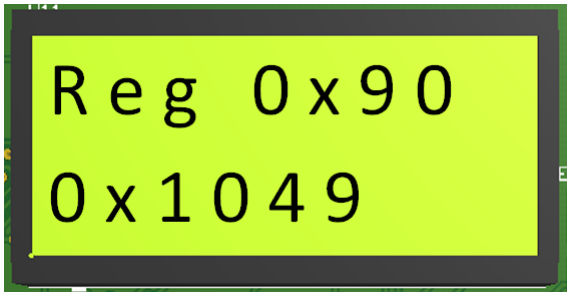


Figure 53. LCD: Reg 0x90 - 0x96

Once all registers have been read, the software will display “Ready” again. Stay in Section 5.3.2.3.

5.3.2.3.3 Pressing the Done Button

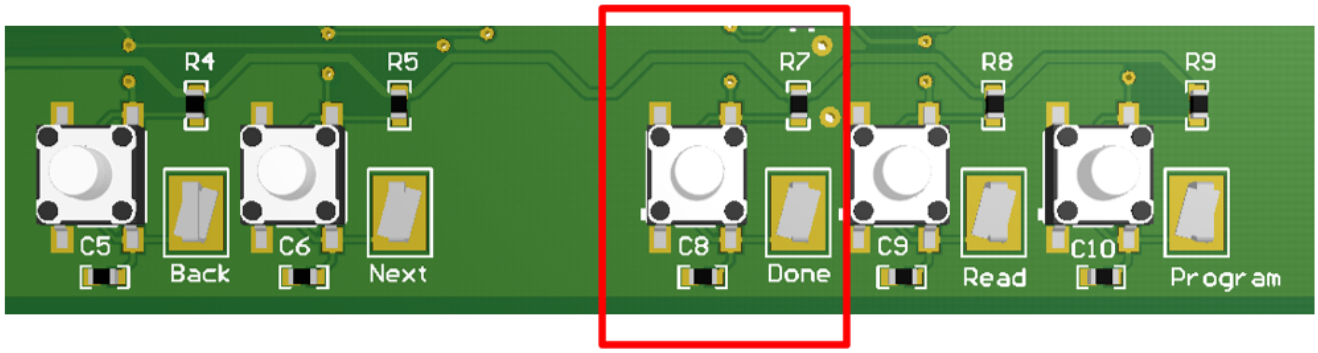


Figure 54. Done Button

Pressing the **Done** button will put the MCU in a low power mode and enter an infinite while loop. Once this button is pressed you can power down the system by setting the ON/OFF switch to the 'OFF' state. Move to Section 5.3.2.4.

5.3.2.3.4 Pressing the Read Button

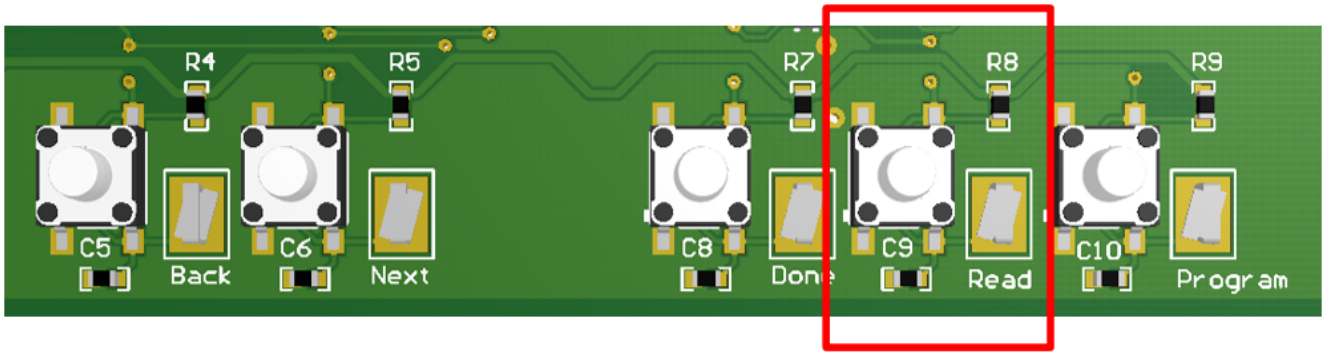


Figure 55. Read Button

Pressing the **Read** button results in the software reading the 12 configuration registers of each device in each of the populated sockets. Once read, the software will compare them to the desired values indicated in the header file.

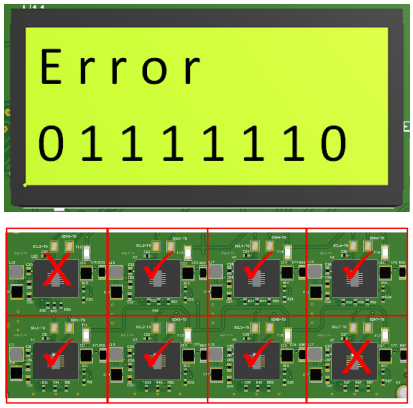
If all configuration registers on each device in the populated sockets match the desired values in the header file then “All Regs Match!” will display on the LCD, as shown in Figure 56. Pressing the **Next** button will return you to Section 5.3.2.3 and “Ready” will display again.

Figure 56. LCD: All Regs Match!

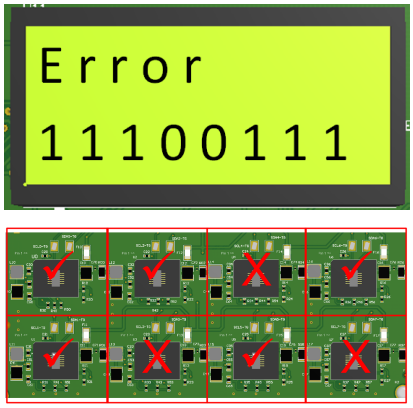
If there is a discrepancy, then the LCD will display “Error” as well as indicate which device has registers that do not match the desired ones in the header. For example:

- 01111110 indicates sockets 1 and 8 are not programmed according to the header file
- 11100111 indicates sockets 4 and 5 are not programmed according to the header file
- 11011011 indicates sockets 3 and 6 are not programmed according to the header file
- .. and so on

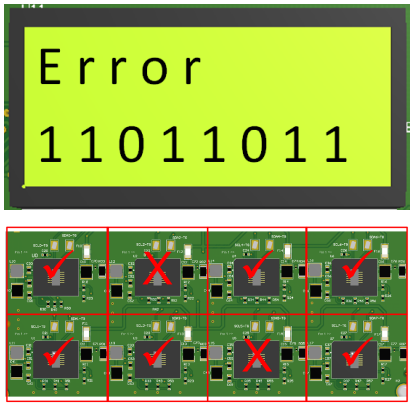
Error: 01111110



Error: 11100111



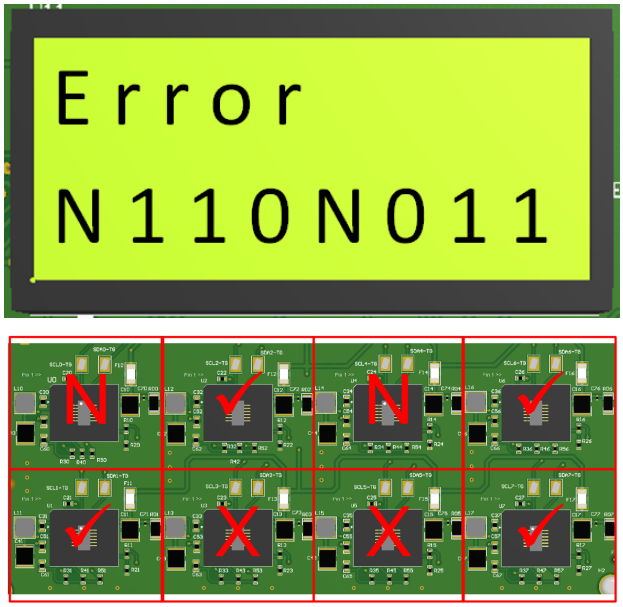
Error: 11011011



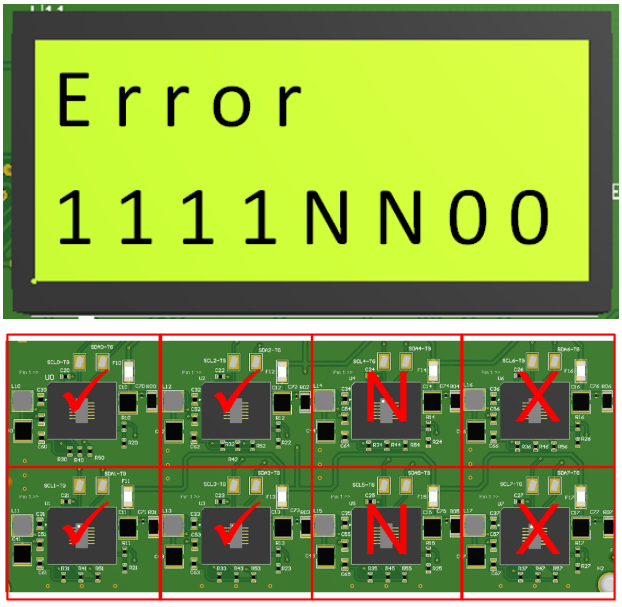
This will also show which sockets are not populated by displaying “N” in that location. For example:

- N110N011 indicates sockets 1 and 5 are not populated and sockets 4 and 6 are not programmed according to the header file
- 1111NN00 indicates sockets 5 and 6 are not populated and sockets 7 and 8 are not programmed according to the header file
- ... and so on

Error: N1100N011



Error: 1111NN00



After displaying this error message, the user can press the **Next** button again and the screen will display "Try Again Ready" as shown in Figure 57. Pressing the **Next** button will return you to Section 5.3.2.3 and "Ready" will display.

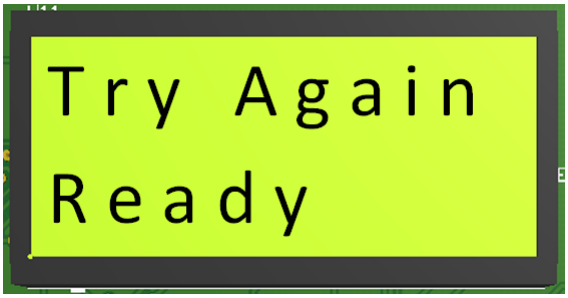


Figure 57. LCD: Try Again Ready

The user may also chose to go to Section 5.3.2.4 and begin troubleshooting.

5.3.2.3.5 Pressing the Program Button

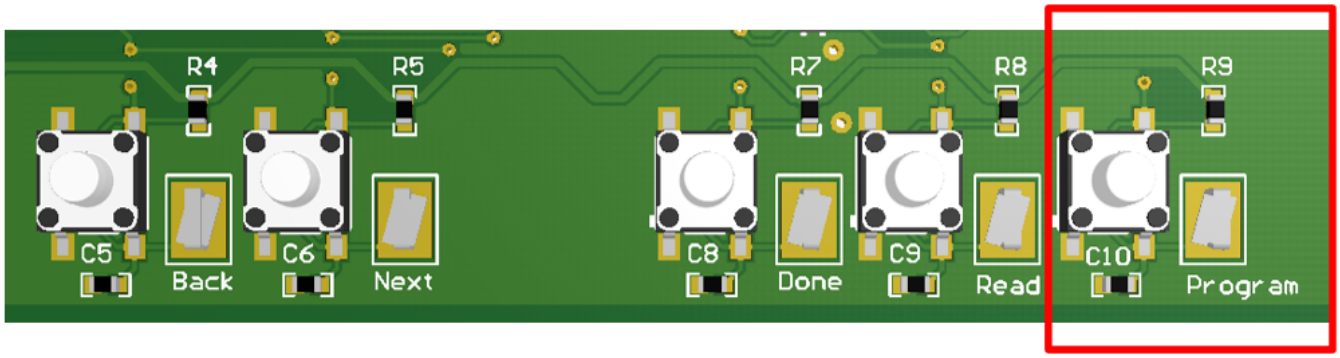


Figure 58. Program Button

Pressing the **Program** button will result in the software writing the desired values in the header into the configuration registers of each of the populated devices. This routine will also run the routine in Section 5.3.2.3.4. Once finished the LCD screen will display "Complete" as shown in Figure 59. At this point the user may move to Section 5.3.2.4.

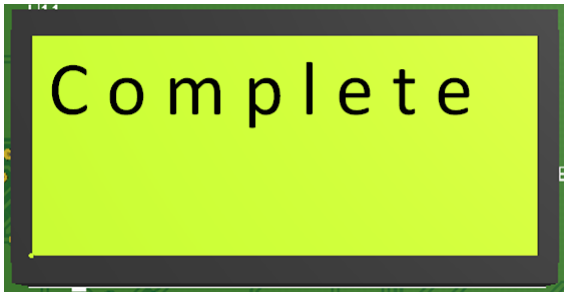


Figure 59. LCD: Complete

5.3.2.4 Powering Down the TIDA-01373

Power off the TIDA-01373 by setting the ON/OFF switch to the 'OFF' position, move to Section 5.3.2.5.

5.3.2.5 Depopulating the Sockets

Make sure the ON/OFF switch is set to 'OFF'. Carefully remove the devices in the sockets.

For repetitive use, return to Section 5.3.2.1.

6 Using a LaunchPad™ to Program the MSP4330G2553 on the Programming Board

Some of the TIDA-01373 Reference Design solutions have a different JTAG header meant to be programed with an MSP430G2553 LaunchPad.

6.1 Programming Tools

To program the MSP430G2553 on the TIDA-01373, use the emulation hardware of the MSP430 LaunchPad with the J4 header populated with short soldered wires and the Code Composer IDE from TI.

The list of devices supported by the on-board emulator of the MSP430 LaunchPad is found in Table 1.

Table 1. Features Supported by On-Board Emulator

Feature	Supported by MSP-EXP430G2 LaunchPad Development Kit
Supports MSP430F20xx, F21x2, F22xx, G2x01, G2x11, G2x21, G2x31, G2x53	✓
Supported by IAR	
Allows fuse blow	
Adjustable target supply voltage	
Fixed 2.8-V target supply voltage	
Fixed 3.6-V target supply voltage	✓
4-wire JTAG	
2-wire JTAG	✓
Application UART	✓
Supported by CCS	✓
Supported by IAR	✓

6.2 Programming Tools Setup

1. Acquire the MSP430 LaunchPad with a mini-USB cable
2. Download version 6.2 of Code Composer Studio, CCS-FREE with support for the MSP430G2553
3. Acquire proper short wires to solder to J4 on the MSP430 LaunchPad, see Figure 60

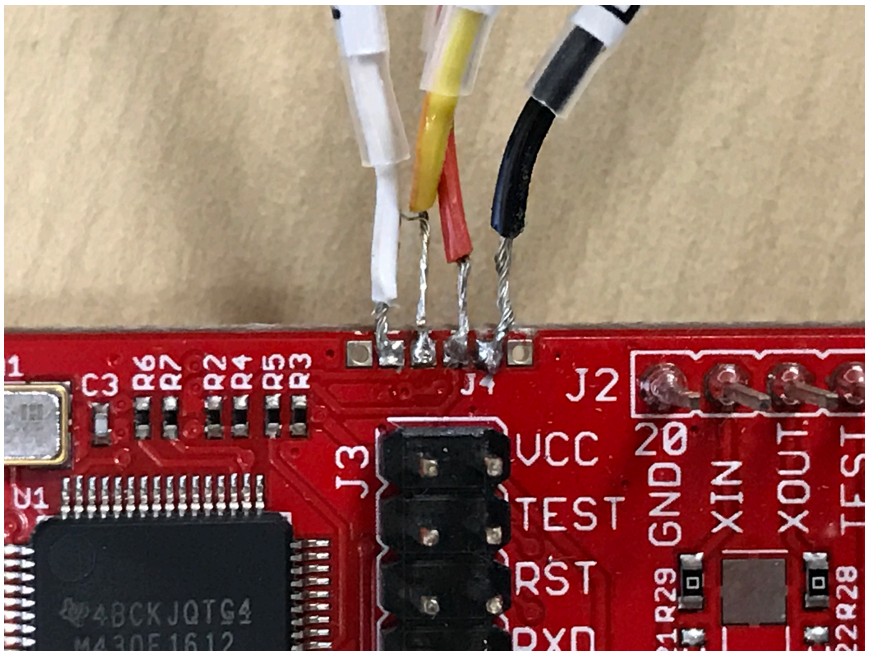


Figure 60. LaunchPad™ 4-Pin Programming Header with Soldered Wires

4. Remove the MSP430 chip from the LaunchPad, if one is present, see Figure 61

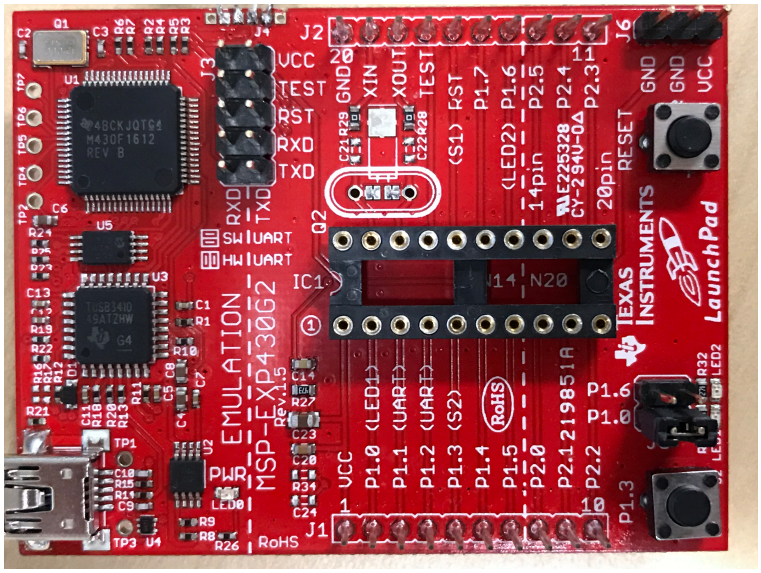


Figure 61. LaunchPad™ MSP430™ Socket (Depopulated)

5. Connect the TIDA-01373 to the MSP430 LaunchPad with the short wires. Both boards should be orientated with the silkscreen up. Note that the two outside wires (GND and 3v3) are opposite on the TIDA-01373 Programming Board than on the MSP430 Launchpad (shown in Figure 62).

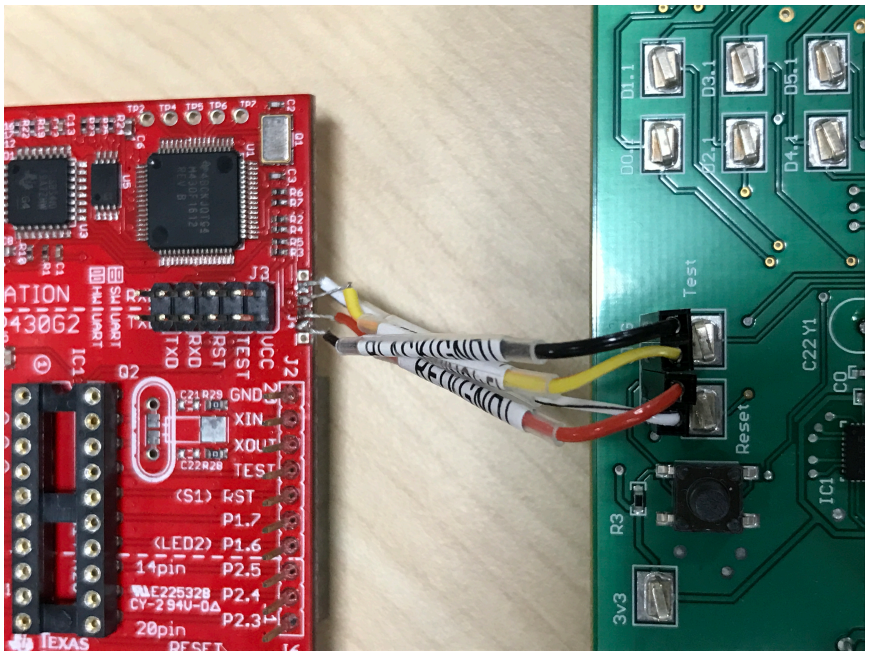


Figure 62. LaunchPad™ Connected to TIDA-01373

6. Connect the mini-USB cable to the LaunchPad from the USB port of your computer (the GREEN PWR LED on the LaunchPad should turn on)
7. Return to Section 4.2.2.4 (DRV10983 and DRV10975) or Section 5.2.2.4 (DRV10983Q1 and DRV10987) to finish flashing the TIDA-01373 solution

7 Using An Alternate Programming Board Solution

The TIDA-01373 two-board solution design permits using the *Socket Board* with a custom-designed programming board, if the user so chooses. The *Socket Board* already has the proper I^2C pullup resistors for the I^2C communication to the DRV devices.

See the design files for the *Programming Board* at TIDA-01373 for an example *Programming Board* solution.

See the software file for the MCU of the *Programming Board* at TIDA-01373 for example software solutions.

8 Using An Alternate Socket Board Solution

The TIDA-01373 two-board solution design permits using the *Programming Board* with a custom designed *Socket Board*, if the user so chooses. The Programming Board already has the proper I^2C pullup resistors for the I^2C communication to the DRV devices.

See the design files for the *Socket Board* at TIDA-01373 for an example *Socket Board* solution.

9 Related Documentation

1. *DRV10983 12- to 24-V, Three-Phase, Sensorless BLDC Motor Driver* data sheet (SLVSCP6)
2. *DRV10975 12-V, Three-Phase, Sensorless BLDC Motor Driver* data sheet (SLVSCP2)
3. *DRV10983-Q1 12- to 24-V, Three-Phase, Sensorless BLDC Motor Driver* data sheet (SLVSD14)
4. *DRV10987 12- to 24-V, Three-Phase, Sensorless BLDC Motor Driver* (SLVSE89)
5. *DRV10983, DRV10975, and DRV10983Q1 EEPROM Programming Tool Reference Design* design guide (TIDUCX2)
6. *MSP430G2553 Mixed Signal Microcontroller* data sheet (SLAS735)

10 Terminology

BLDC— Brushless DC

CCS— Code Composer Studio

EEPROM — Electrically erasable programmable read-only memory

GND— Ground

JTAG— Joint Test Action Group

MCU— Microcontroller

MUX— Multiplexer

USB— Universal serial bus

11 About the Author

Michael Thomas Schneider is a Motor Driver Applications Associate in the Applications Rotation Program at Texas Instruments, where he is responsible for developing reference design solutions and supporting the DRV10x family of devices. Michael brings to this team his knowledge of mixed signal system level designs as well as electric motor drives in order to continue to build collateral material for the DRV10x family. Michael earned his Bachelor of Science in Electrical Engineering (BSEE) at the University of Texas at Austin.

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from Original (April 2017) to A Revision	Page
• Added DRV10987 globally throughout the document.	7
• Changed DRV10983Q1 to DRV10983-Q1 throughout document.	7
• Changed *TIDA-0137 Block Diagram*.	15
• Changed and DRV10983 to DRV10983-Q1 and added DRV10975 in the *Using the TIDA-01373 Hardware and Software for the DRV10983-Q1 and DRV10987 section.*	31

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