Application Report
SCAA137–July 2019

I2C Dynamic Addressing

Bobby Nguyen, Nick Scandy

ABSTRACT

Many complex systems rely upon I2C switches in order to expand the ability of a single I2C master to communicate with higher numbers of I2C slaves. Despite the fact that both the I2C switch and slave devices often offer the ability to change addresses via strappable pins or different orderable versions, it is theoretically possible to have address conflicts between the I2C switch and another slave. This application note explores the different options that a system designer has for avoiding I2C address conflicts via the use of dynamic addressing.

Contents

List of Figures

List of Tables

Trademarks

1 Introduction

TI has a wide selection of I2C slave devices in its portfolio and an I2C master that can communicate with these slaves by using an address that is usually made up of four or more fixed bits and hardware-selectable bits (as seen in Figure 1 for the TCA9555). While this configuration provides eight distinct address options, it is possible that a crowded I2C bus may see address conflicts between multiple slaves. This is a unique situation, but it does not have to lead to big changes in the I2C communication architecture of the system, as there are a few different techniques available for avoiding this address conflict.

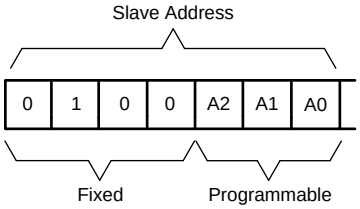

Figure 1. Example of a Three Bit Hardware Addressable I2C Slave (TCA9555)

In this application note, multiple ways of solving this problem are presented, each approach having advantages and disadvantages. The three methods of solving address conflicts that are described are:

- The conventional method of using an I2C switch
- Dynamic addressing by controlling address pins directly via GPOs
- Dynamic addressing by using a combination of a MUX and GPOs

2 Checking for Latched Addresses

Before attempting to dynamically address I2C slaves, you must check if the I2C slave of interest is capable of being dynamically addressed. Unfortunately, this kind of information is typically not stated in a datasheet but rather each I2C slave needs to be tested to see if this is supported by the slave. Typically the determining factor behind whether an I2C slave is dynamically addressable is if the hardware-selectable addresses of the slave are latched during power up. This means any slaves that do not have hardware addressable pins do not qualify as slaves that are dynamically addressable.

Follow these steps to check if an address is latched:

1. Set the hardware-selectable addresses of the slave to a known value then powering up the device.
2. Perform a read or write transaction to the slave. The slave should ACK.
3. After the ACK has been confirmed, flip the hardware selectable address pins to the other logic level (1 to 0 or 0 to 1) while the device is still on (do not reset or power cycle the slave device).
4. Perform another read or write to the old slave address.
5. If the slave NACKs, then perform a read or write transaction to the new slave address. The new slave address transaction should result in an ACK, which confirms the I2C slave can be dynamically addressed.

Example:

TCA6408A has the address: 010 $000A_0$

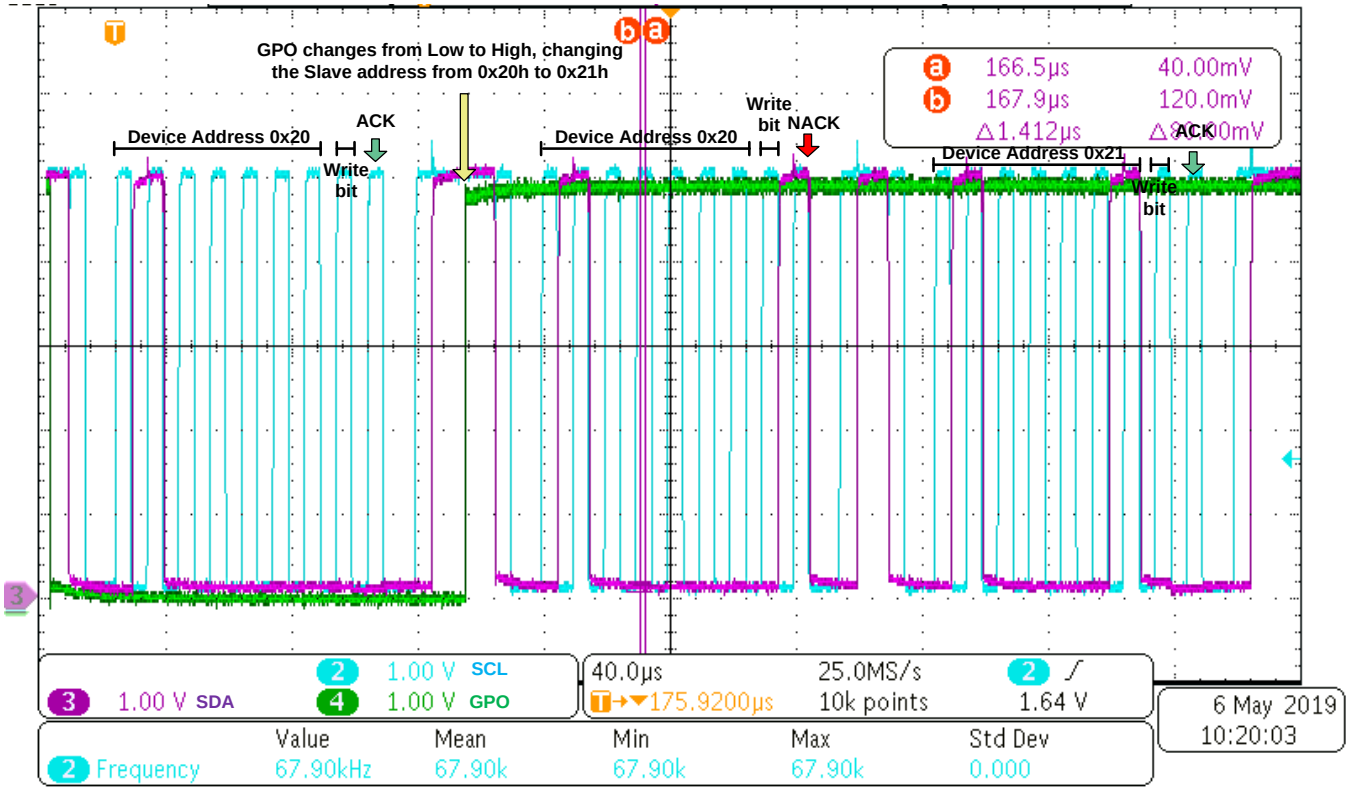

Figure 2. Example Testing to See if TCA6408A Latches Its Address After Power Up

1. Set A0 = GND.
2. Power on device. Address = 0x20h
3. Send a write transaction to slave address 0x20h.
4. Confirm the device ACKs to address 0x20h.
   a. This can be seen in the first nine clock pulses of Figure 2.
5. Without powering down or resetting the device, set A0 = Vcc. The new slave address is 0x21h.
6. Send a write transaction to slave address 0x20h.
7. Confirm the device NACKs to address 0x20h.
   a. This is seen in Figure 2 where after the second start condition, the ninth clock pulse NACKs.
8. Send a write transaction to slave address 0x21h.
9. Confirm the device ACKs to address 0x21h.
   a. This is seen in Figure 2 where after the third start condition, the ninth clock pulse ACKs.

3 Conventional Method (I2C Switch)

In most situations, an I2C switch can be used to solve the problem of address conflicts. Figure 3 shows an I2C switch being used to resolve an address conflict. The advantage to this approach is that the I2C switch is addressable, meaning it can be controlled via the SDA/SCL lines whereas the alternative would be to use a 2:x MUX which would require GPOs on the select pin and does not allow for multiple channels to be enabled. The other advantage the I2C switch has is its capability to support level translation.

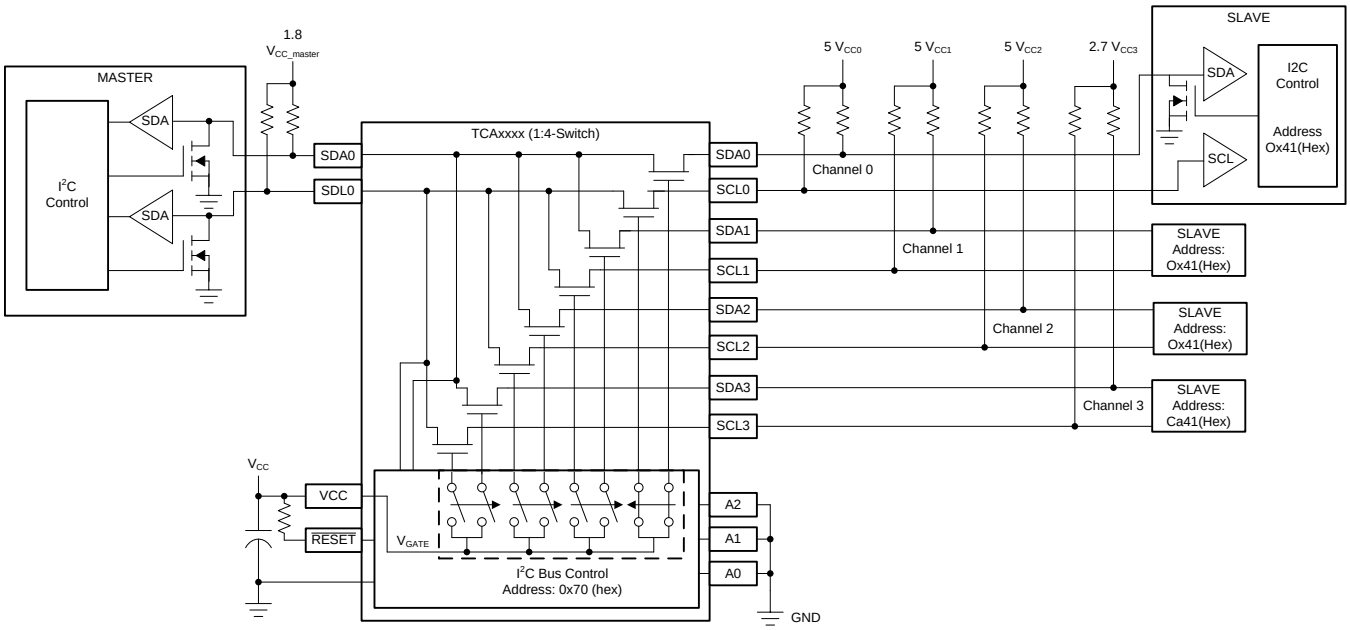

Figure 3. Example of an I2C Switch and How It Switches Internally

The I2C switch should be the recommended solution in most scenarios, but sometimes an I2C switch cannot be used or may not be the ideal situation. I2C switches introduce an Rdson due to their pass FET architecture which in turn generates a larger V_{oL} seen by an I2C device (V_{oL} must be lower than V_{iL} for a logic low to be properly recognized). I2C trees and buses use buffers that provide a current source on one side (devices like the TCA9509 and TCA9800), the side which provides a current source does not work well when connected to a switch because the current source cannot sufficiently provide the logic high through the switch reliably. Placing a pullup resistor past the switch may interfere with the low detection algorithm of the buffer, making I2C switches and current source buffers incompatible and unreliable. In some cases, an I2C bus can be so complex that it requires many I2C switches at which point the number of address pins could run out. Switches may require being used in series and parallel and would require careful software programing to avoid address conflicts. Figure 4 shows an example of a complex I2C tree.

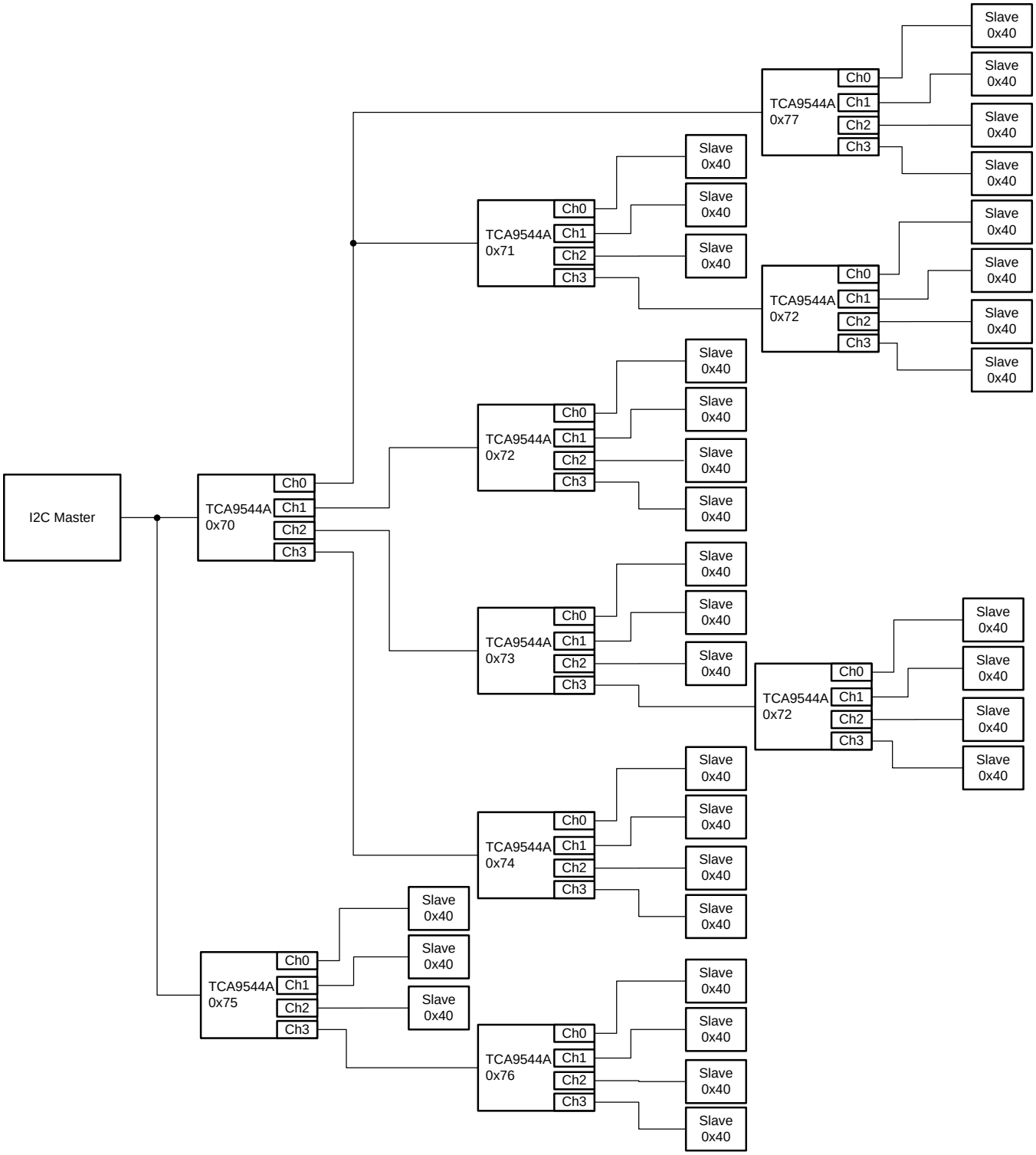

Figure 4. Example of a Complex I2C Tree with Multiple Switches

4 GPO Directly Into Address Pins

Assuming that the slave being addressed can be dynamically addressed (see Section 2) then using a GPO directly onto the addressable hardware pins of the I2C slave can be done as an alternative to an I2C switch. Only one GPO is required per I2C slave. Any additional addressable address pins can be grounded (or pulled to Vcc). When trying to communicate to one of the 'dynamic' slaves, the GPO for the specific slave needs to be different from the other slaves. For example, if there are four slaves, the slave attempting to be written or read needs its address pin to be a logic '1' while the other three are a logic '0' or a logic '0' while the other three slave address pin are a logic '1' like in Figure 5. The processor needs to be programmed to make the slave of interest unique in order to use this method.

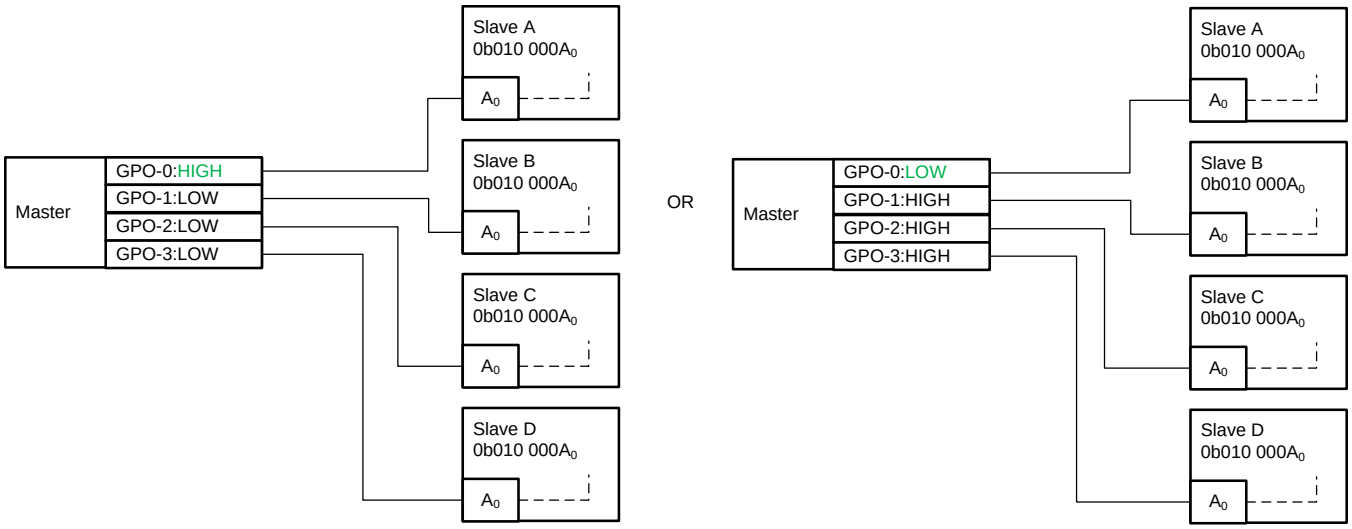

Figure 5. Example of GPOs Controlling Addresses to Communicate with Slave A

The advantage to this approach is that it saves board space when compared to the I2C switch approach and reduces the BOM count by one. Because there is not an I2C switch which needs to be addressed and then enabled, the time to directly address the slave of interest is also shortened.

The down side to this approach is it is limited by the amount of GPOs available, if any. This adds a little bit more complexity to the PCB design (additional traces from GPO to address pin). The additional traces on the PCB are similar to SPI because each address pin needs a GPO to effectively disable/enable the device, which is one of the main advantages to I2C versus SPI.

5 GPO Controls DEMUX Into Address Pins

An alternative to using GPOs tied directly to the I2C address pin approach is using a GPO with a MUX to increase number of address control lines available. By using this method, the number conflicting slaves which can be supported is increased by 2^n where n is the number of GPOs available.

Figure 6 shows the GPO to MUX approach where all slaves are biased to a pullup resistor (effectively sharing the same address) except the channel enabled by the select line. The enabled channel below has its address pins all pulled high except for the selected channel which has the common channel tied to ground. This allows for one slave to have a unique address at a time.

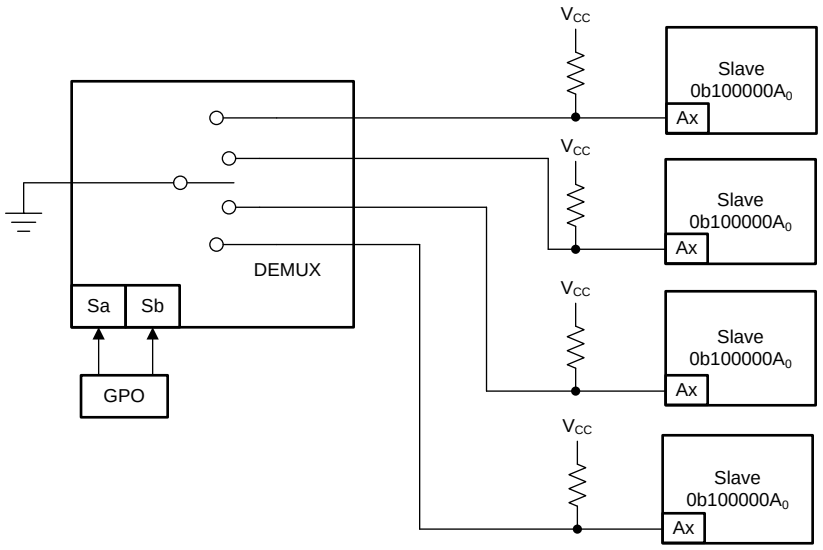

Figure 6. Example of GPO Controlling a MUX

TI has a wide array of analog multiplexers that would be capable of this function. The system designer is invited to choose the optimal device based on their requirements, though the TI recommended device for this application would be the TMUX1204. It is a strong fit for this application based on the following highlighted functionality and specifications:

- 1:4 channel multiplexing configuration which would allow the system designer to use two GPOs to control four slave address pins
- A fixed 1.8 V logic thresholds for the control pins. This allows a possible configuration where the supply voltage of the TMUX1204 may be higher than the I/O voltage used for the A0 and A1 control pins.
- A low supply current of 0.01 µA (typical), 1.3 µA (maximum across -40°C to 125°C), and low leakage current (less than +/-750 nA maximum, across -40°C to 125°C) which provides negligible system impact in terms of total supply current.
- A small uSON package with a footprint of 2.5-mm x 1.00-mm

In the case where there may not be any GPOs free to use on the MUX select pins, an I2C I/O expander could be used to select bias the select pins (this of course assumes the I/O expander has its own unique I2C address).

6 Conclusion

Multiple methods for addressing I2C bus address conflicts have been presented and their trade-offs have been evaluated. An I2C switch can be used in most scenarios, provided there are enough addresses available to give the I2C switch a unique address and the impact of the added series resistance of the switch to the I2C lines can be tolerated. If there are no other options in terms of address conflicts, a slave device can be evaluated for the possibility of dynamic addressing and its address pins can be directly driven by a GPO. In the event that there are address conflicts, a slave device can be dynamically addressed, but there are limited GPOs available, an analog multiplexer can be used to increase the number of dynamically addressed slaves per available GPO.

IMPORTANT NOTICE AND DISCLAIMER

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

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

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

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