Migrating your proprietary solution to the SimpleLink™ WMCU

Evan Wakefield
SimpleLink Embedded Software
Applications Engineer
Texas Instruments

Internet of Things (IoT) market growth has brought an increased focus on the integration, security, performance and capabilities of wireless devices and networks.

Higher-speed interfaces and multiple connectivity options require more advanced microcontrollers (MCUs) that can handle both wireless communication and application needs. Companies are also striving to serve multiple markets with different adaptations of their base product. Developers need a robust software foundation with intuitive abstraction levels and an operating system that supports the faster creation of applications, especially in the low-power wireless markets.

For example, in the Sub-1 GHz space, new single-chip wireless MCUs (WMCUs) are becoming more common, replacing two-chip solutions. These two-chip solutions use a base MCU for both running the application and for controlling the wireless transceiver through some kind of serial interface such as Universal Asynchronous Receiver Transmitter (UART) or the Serial Peripheral Interface (SPI). If you've spent years tuning and perfecting your two-chip design, you may be hesitant when presented with the benefits of moving to a single-chip solution, with concerns about the feasibility of moving your current solution to a WMCU.

Some of these concerns may include:

- Software migration and reuse: You likely want to continue leveraging the software investment you've made on two chips and bring as much of it over to a WMCU.
- Concerns around real-time operating systems (RTOSs). Moving to a newer WMCU will often require an RTOS. Some engineers do not have experience with RTOSs; others want to use an RTOS that they've leveraged in previous solutions.
- Radio-frequency (RF) performance capabilities – can WMCUs match the performance capabilities achieved in transceivers?

Especially in the Sub-1 GHz space, proprietary wireless solution developers have invested time in their wireless stack and end application and spent years perfecting performance, timing and power. When approached with the idea of moving to a new solution, it makes sense to be cautious about migrating, but today's new WMCUs offer improved radio performance, new modulation schemes, lower-power operation, and a suite of pre-built and pre-tested software.

In this paper, I will present why moving a WMCU can be beneficial, try to dispel concerns about migrating to WMCUs and discuss some possible paths available today for migrating your proprietary solution with minimal effort allowing you to reap the benefits of a single-chip WMCU and the SimpleLink SDK.

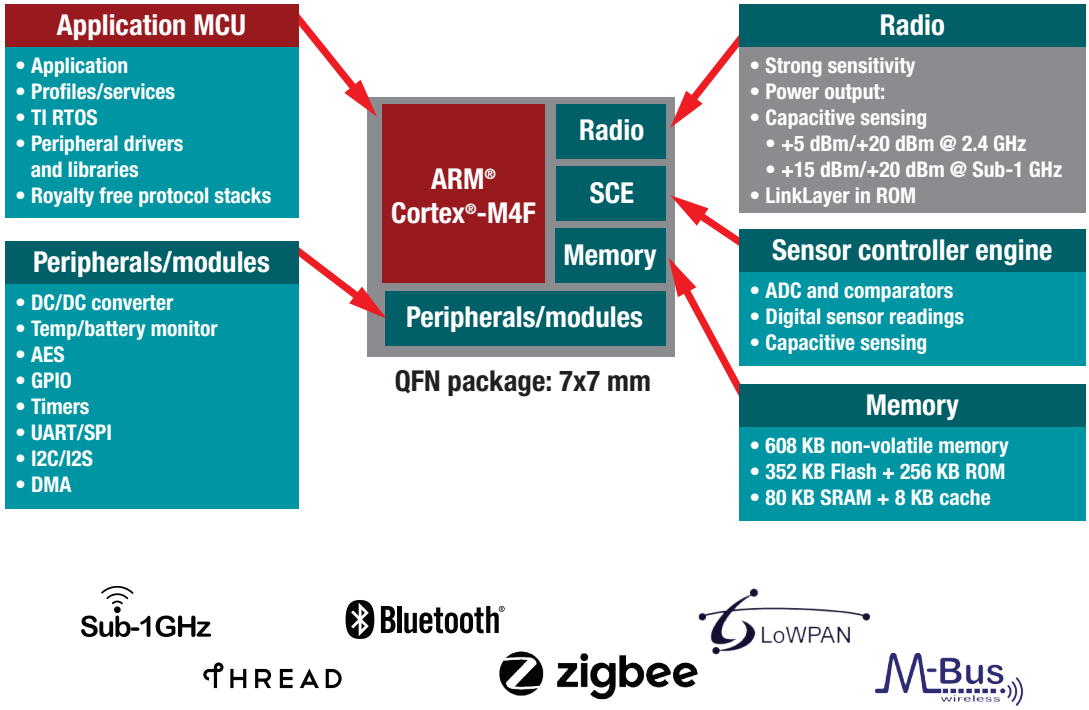

***Figure 1**. The SimpleLink CC1352R.*

Breaking down proprietary wireless applications

Most proprietary applications can be simplified to a few basic blocks where developers will spend a lot of time developing and tuning their solution. This is true for either a two chip solution using a transceiver and a base MCU or of a one chip, wireless MCU. These blocks are:

- Power management
- Overall application code
- RF drivers
- Proprietary stack

Later on, we will address each of these components specifically and help discuss possible paths that can be leveraged, but first, let's look at some of the advantages of the SimpleLink WMCUs and the SimpleLink SDK.

Advantages of moving to a TI WMCU

The CC1352R and the CC1310 WMCU families from Texas Instruments (TI) are multi-protocol Sub-1 GHz and 2.4-GHz WMCUs targeting a variety of different standards and proprietary wireless solutions.

The CC1352R and CC1310 devices combine a flexible, low-power RF transceiver with an Arm® Cortex® host central processing unit (CPU) in a platform that supports multiple physical layers and RF standards. Each device has a dedicated radio controller that handles low-level RF protocol commands stored in read-only memory (ROM) or random access memory (RAM) accessed via RF core drivers, thus ensuring ultra-low power and great flexibility. The low-power consumption of CC13x2R and CC13x0 devices does not come at the expense of RF performance; the WMCUs have excellent sensitivity and robustness.

Alongside this, the CC13x2R and CC13x0 devices, you can leverage the benefits of new power-management drivers, pre-built peripheral and security drivers, newer and more advanced modulation schemes, and higher performance radios available today in the SimpleLink Wireless MCUs.

TI Transceivers	Data Rate/Freq	TX Power Max (dBm)	Sensitivity (dBm)	TX Power Consumption at + dBm (mA)	RX Power Consumption (mA)
CC110L	38.4 kbps/868 MHz	12	-104	30	14.6
CC1101	38.4 kbps/868 MHz	12	-104	30	14.6
Wireless MCUs					
CC1310	50 kbps/868 MHz	14	-110	13.4	5.4
CC1312	50 kbps/868 MHz	14	-110	14.3	5.8
CC1352P	50 kbps/868 MHz	20	-110	65 mA @ +20 dBm	5.8

***Table 1**. RF performance using GFSK modulation between transceivers and wireless MCUs.*

New modulation schemes and higher throughput

Within the Sub-1 GHz space, both the CC13x0 and CC13x2R WMCUs have the capability to perform new and more advanced types of modulation. These new modulation schemes include long-range modes, Wide-band Direct-Sequence Spread Spectrum (DSSS) (Federal Communications Commission 15.247 Compliant), antenna diversity, 802.15.4G and many others, as well as most modulations schemes that were previously available on TI's CC11xx transceivers.

RF performance comparison between transceivers and WMCUs

One of the biggest challenges when integrating radios into an MCU is the introduction of noise; isolation prevents noise from switching characteristics in either the MCU or device radio from influencing one another. TI continues to find new and improved ways to package and isolate silicon to ensure that radio performance is not hindered when using a WMCU. **Table 1** shows a comparison of the radio performance of TI's transceivers compared with the capabilities of today's wireless MCUs.

As noted, there have been significant advances in power consumption for both transmitting and receiving as well as sensitivity. In addition to those improvements, TI has also taken WMCU's a step further and integrated a power amplifier into the CC1352P device that allows developers to transmit at higher TX powers than ever before. The device contains a +20-dBm integrated high-power amplifier with high efficiency for long-range applications. More specific performance details related to the information above can be found in the device specific datasheet on the product page. Explore the SimpleLink Sub-1 GHz products.

The SimpleLink SDK ecosystem

Before migrating, you must first understand the

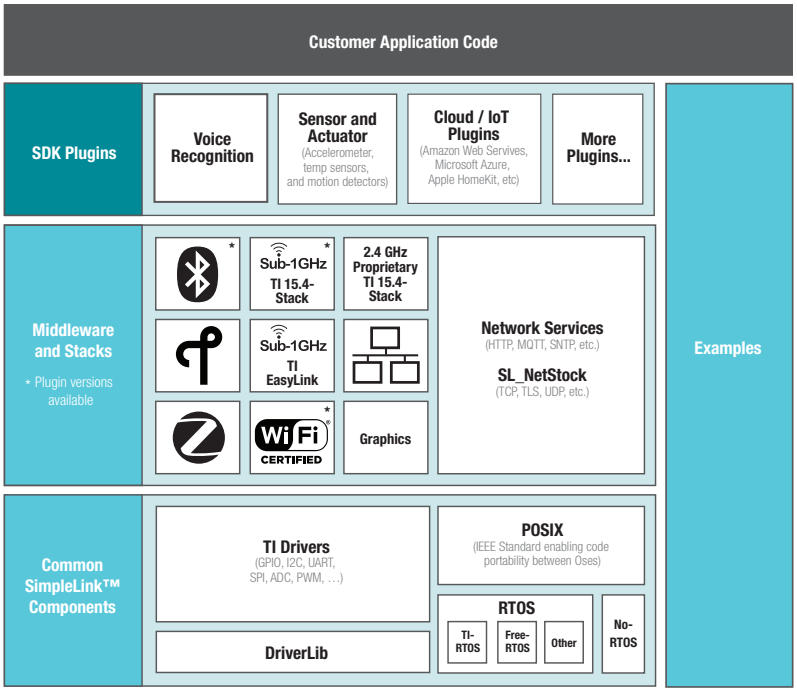

***Figure 2**. Key components of the SimpleLink SDK.*

ecosystem and advantages of where you are migrating to. To leverage the new TI WMCUs, developers can look to the SimpleLink MCU SDK. The SimpleLink SDK is a complete set of validated, fully documented drivers, stacks and code examples that enable engineers to develop innovative and differentiated applications with the SimpleLink connected-MCU family from Texas Instruments. Everything a customer needs to quickly and efficiently develop new applications using a SimpleLink ARM® Cortex®-M-based MCU from TI is included in the SDK. The SDK also integrates seamlessly into development environments making developing and debugging easily accessible.

SimpleLink Academy

In conjunction with the SimpleLink SDK, SimpleLink Academy offers online training modules on topics ranging from "getting started" labs on TI-RTOS and various wireless stacks all the way to more advanced topics such as learning how to do Over-The-Air download. No matter the experience level, SimpleLink Academy is great for anyone wanting to learn about the possibilities of the SimpleLink MCUs and can be leveraged in the process of migrating to acquire understanding of TI's proprietary solutions.

Power management

The SimpleLink MCU SDK includes power-management drivers that enable you to quickly and aggressively control power consumption in your application without developing it yourself.

The power-management framework supports the CC13xx, CC26xx, CC32xx, MSP432E4 and MSP432™ devices. All three MCU families use the same top-level application programming interfaces (APIs), concepts and conventions. The CC13xx and CC26xx share the same device-level implementation and use the "CC26xx" prefix for file names, function names and constants. The framework works in both RTOS and non-RTOS environments.

Application code

Unless you are porting from an MCU like the SimpleLink MSP432 device using TI drivers, most application code will need to be ported. When migrating, you will need to consider the available integrated peripherals, flash and RAM. In addition to migrating the functional application code, you can take advantage of SimpleLink TI drivers.

The TI driver API exposes the functionality of hardware-specific drivers across all TI SimpleLink devices, giving you portable, feature-rich access to a variety of peripherals. TI drivers are open-source (Berkeley Software Distribution license) and built on the hardware abstraction layer, offering full access to the device's complete capability. For example, although the hardware implementation of the UART may be different across each device, the TI driver API used to access its functionality is the same.

TI drivers include serial communication drivers; RF drivers; and security drivers such as elliptic curve Diffie-Hellman (ECDH), elliptic curve digital signature algorithm (ECDSA) and true-random number generator (TRNG).

TI Transceivers	Flash	RAM	Sleep Current
CC110L	N/A	N/A	200 nA + application MCU power consumption
CC1101	N/A	N/A	200 nA + application MCU power consumption
Wireless MCUs			
CC1310	128 KB	20 KB	Standby: 0.7 µA (real-time clock running and RAM and retention)
CC1312	352 KB programmable flash, 256 KB of ROM	80 KB	Standby: <1 µA at 80 KB of RAM retention
CC1352P	352 KB programmable flash, 256 KB of ROM	80 KB	Standby: <1 µA at 80 KB of RAM retention

***Table 2**. Device comparison for memory and sleep current.*

After migrating your existing application code into the TI driver framework, you can leverage that application code across SimpleLink CC26x2, CC3220 and MSP432 MCUs which feature Bluetooth Low Energy, Wi-Fi and other capabilities.

RF core drivers

Another benefit of moving to the SimpleLink WMCUs is that you do not have to write or test your own RF drivers; you can leverage the extensive RF core drivers in the SimpleLink SDK. The RF driver offers very low-level APIs to run radio operation commands on the RF core and send and receive raw packets. They are extensively tested; include a variety of modulation schemes; and are the only RF drivers used across TI wireless network offerings from Zigbee, *Bluetooth*® low energy, Thread, TI 15.4 Stack, and our proprietary network solutions in CC13x0 and CC13x2 devices.

Migrating a proprietary stack

The most common question for developers of an application that leverages a proprietary network stack relates to how easy is it to migrate the existing proprietary network stack to work on the new WMCUs. Developers want to be able to leverage as much of their proprietary network stack as possible in the new design to maintain interoperability and save cost on software development. The good news is that there is a clear path forward, whether you are familiar with RTOSs or have been using lower-level C code.

What does that path forward look like? While not an exhaustive list, your current proprietary solution might sound like one of these three types:

- **Type 1**: This solution uses register-level C code and/or leverages a driver library set of APIs, but there is also a custom-built scheduler for the proprietary stack and application code. This scheduler manages resources within the device but also can invoke the RF driver. The vast majority of the network's protocol exists within this scheduler.
- **Type 2**: This solution also uses register-level C code and/or leverages a driver library set of APIs, but there is no built-in scheduler. Sending or receiving wireless packets is achieved through polling or using interrupts in a traditional bare-metal C program. There is no scheduler to handle multiple tasks.
- **Type 3**: This solution leverages the TI-RTOS, Free-RTOS or any available or even homegrown RTOS. These solutions may feature familiar RTOS software features such as semaphores, events, mutual exclusion objects, threads/tasks, and hardware and software interrupts. The RTOS can schedule and invoke RF drivers. The network scheduler/timing is within the context that the RTOS provides and is tuned to the timing that the RTOS provides.

Type 1: the custom-built scheduler

For those with the first type of existing solution, it is possible to reuse the network stack by placing the entirety of the existing stack/scheduler into an RTOS task or thread that executes at the system's highest priority. You will have to write software to provide a mechanism for the application to interface with the stack services as well as certain primitive services provided by the underlying RTOS. In this architecture, this software would act as a dispatcher, allowing the application and protocol stack to communicate, share resources and operate efficiently in a unified RTOS environment.

TI took a similar approach when moving the Bluetooth low energy stack available on the CC2541 to the CC2640R2F.

When the CC2541 MCU was introduced, TI shipped the device with a Bluetooth low energy stack developed around a concept called an operating system abstraction layer (OSAL). The OSAL is not

an actual operating system in the traditional sense but rather a control loop that allows software to set up the execution of events. The OSAL is considered a basic scheduler and is a non-pre-emptive single-threaded architecture and therefore not a full-blown RTOS.

When the CC2640R2F MCU was introduced a decision was made to support the Bluetooth low energy stack with TI-RTOS. To preserve the investment in the Bluetooth low energy stack, TI developers took the stack and OSAL and placed it within the context of an RTOS task/thread that would execute at the system's highest priority. A layer of software called the indirect call framework (ICall) shown in **Figure 3**, interfaces the stack protocol services with the application. TI is still using this architecture in devices that support Bluetooth low energy, as well as TI's 15.4-Stack supporting Sub-1 GHz, and the Zigbee solution.

Type 2: improving on the basics

If you have the second type of existing solution, there will be a small learning curve. New high-performance capabilities require a scheduler to control resources and manage tasks. Some developers choose to develop their own schedulers, but a number of RTOSs feature multi-threading capabilities inherently.

Working with an RTOS can be challenging at first, but today's more complex, more powerful and more integrated devices spotlight the development advantages of an RTOS, which include:

- **Small latency,** hence real-time.
- **Determinism.** You will need to have knowledge of how long things will take to process in order to meet deadlines.
- **Structured software.** With an RTOS, you can develop in a structured manner. Adding components as needed into the application is straightforward.
- **Scalability.** RTOS's scale with your needs. An RTOS can be used in a simple application to a more complex one with stacks, drivers, file systems, etc.

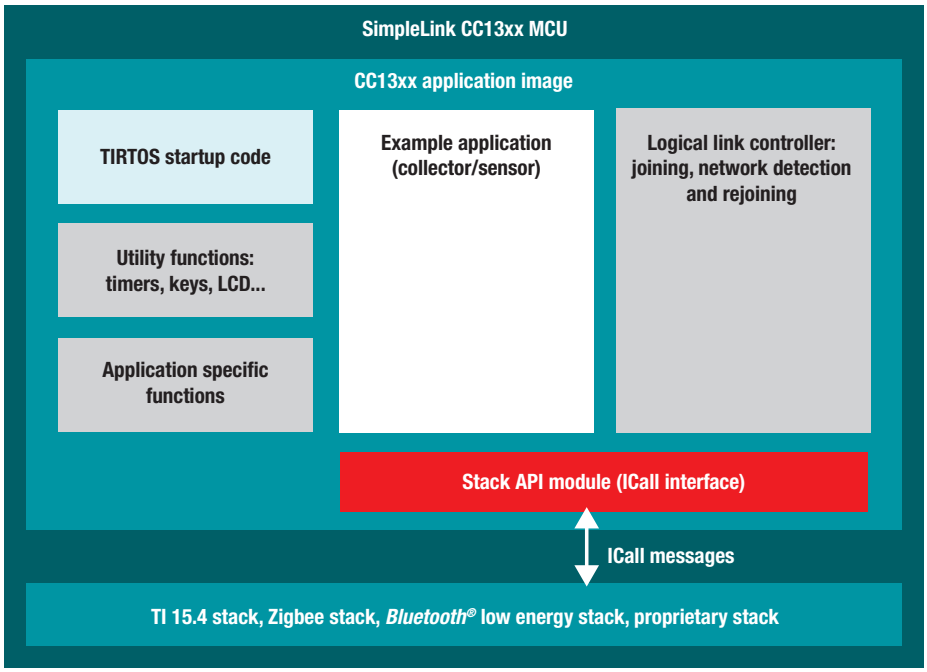

***Figure 3**. Diagram of indirect call architecture.*

- **Offloaded development.** An RTOS already has many aspects of the system integrated into itself including power management, interrupt table management, memory management and exception handling and more.

TI-RTOS

The TI-RTOS (formerly known as SYS/BIOS) is a scalable, embedded tool ecosystem for TI devices that accelerates development schedules by eliminating the need to create basic system software functions from scratch. TI-RTOS scales from a minimal footprint real-time multitasking kernel to a complete RTOS solution including protocol stacks, multicore communications, device drivers and power management. By providing pre-tested and pre-integrated system software components, TI-RTOS enables you to focus on differentiating your application.

In either type 1 or type 2, when starting to implement a proprietary network solution in TI RTOS, we strongly recommend starting with one of the simple examples in the SDK and then implement the needed functionality there, rather than take the original code and try to change that into TI-RTOS code. While doing this, developers can leverage SimpleLink Academy to learn more about each example, how to use it and the capabilities of TI-RTOS. If the project involves migrating from a solution that previously used a CC1101 based transceiver, developers and look at the “Migrating from the CC1101” section of the proprietary RF users guide to understand what specifics can translate from the CC11xx transceivers to the new WMCUs.

Type 3: already using an RTOS

If you have the third type of solution and wish to continue using an RTOS that is not TI-RTOS, you can leverage the drivers porting layer (DPL) built into the SimpleLink SDK. The DPL is an abstraction layer that simplifies the porting of drivers to other RTOSs.

TI drivers are RTOS-independent and will use resources available in any RTOS solution, such as semaphores, mutual exclusion objects, interrupt management and timing services. TI drivers shipped in the SimpleLink SDK function equally across TI-RTOS, Amazon Free RTOS and no-OS implementations.

To migrate a v/ solution from an RTOS not supported in the SimpleLink SDK, you can use the DPL layer to hook up your preferred RTOS by implementing the DPL functions necessary to use the TI drivers.

The TI drivers use these DPL modules to maintain RTOS independence:

- ClockP: clocking services (one-shot timer, timestamps, etc.)
- DebugP: debugging APIs, generally easy to implement on a new RTOS
- HwiP: interrupt management (plugging vector table, disabling interrupts, etc.)
- MutexP: mutual exclusion, generally easy to implement on a new RTOS
- SemaphoreP: counting/binary semaphores, generally easy to implement on a new RTOS
- SwiP: software interrupt management (to defer work from an interrupt service routine to a lower-priority thread)

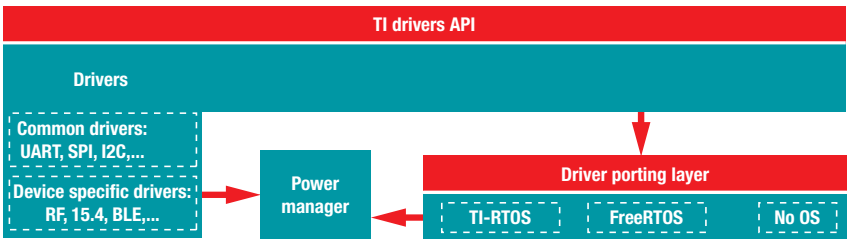

***Figure 4**. Block diagram of drivers porting layer.*

Step 1	Identify where you are coming from	Review the types discussed in this paper
Step 2	Break the software down into blocks (stack, application, power management, RF drivers)	Understand what is offered on the WMCU for each component of your application by studying the topics discussed in this paper closely and think through the best path forward for your application.
Step 3	Learn about and leverage the training and documentation provided in the SimpleLink SDK	See • SimpleLink Academy for SimpleLink CC13x2 SDK • SimpleLink CC13x2 SDK Documentation Overview • Migrating from CC1101
Step 4	Prototype and implement	Buy a launchPad™ development kit, download the SDK and get started!

***Table 3**. High-level steps to consider when migrating to a WMCU.*

- SystemP: optimized snprintf and vsnprintf that can use existing FreeRTOS or no-OS implementations

The SimpleLink SDK provides the header files that define the functions, constants and data types for each DPL module. You can use the header files where the DPL functions are specified as a guide and develop your own header files to integrate your own RTOS. Utilizing the DPL requires a good knowledge and understanding of RTOS software concepts and is best tackled by those with have a strong software background.

After porting over the RTOS, you can leverage most, if not all, of your network protocol stack and third party RTOS. Enabling power management and porting the higher-level application code will require some work, but overall, the DPL significantly decreases the effort and time that it takes to port and application related to Type 3 over to TI WMCUs.

What's next?

You can leverage TI's SimpleLink ecosystem to get a head start, whether you're interested in building your own proprietary network solution or migrating your existing solution to TI's WMCUs. You can look forward to improved radio performance, new modulation schemes and lower-power operation, and leverage a suite of pre-built and pre-tested software while maximizing software reuse from previous proprietary solutions.

The SimpleLink SDK offers a robust set of examples and documentation that works as a foundation and guide for building or migrating a proprietary network.

Table 3 lists a four-step plan if you wish to migrate an existing proprietary network to a TI WMCU. If you're considering moving to a TI WMCU, examine these steps and start developing with TI's WMCUs today! If you have more questions along the way, feel free to post any questions on TI's Engineers to Engineers (E2E) forum.

References

- Check out the SimpleLink™ Sub-1 GHz WMCU page.
- For more information on the SimpleLink SDK, see the white paper, "Simplifying software development to maximize return on investment."
- See the CC1310 and CC1352R data sheets.
- For more information about ICall, see the SimpleLink SDK documentation.
- If you're new to RTOSs or looking to sharpen your knowledge about RTOSs, see the SimpleLink Academy training that covers RTOS Concepts as well as TI-RTOS Basics.
- TI Engineer To Engineer Support Forum: e2e.ti.com

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