Application Report

How to Migrate CCS 3.x Projects to the Latest Code Composer Studio™ (CCS)

DSP Processors

ABSTRACT

This application report covers the key points on how to migrate an legacy Code Composer Studio™ project developed probably years back to the latest CCS and tools. For this application report, the DSP processors is the main focus although the methods may be applicable to other processors as well.

Table of Contents

List of Figures

Trademarks

1 Introduction

There is not a general set of steps to migrate CCS 3.x projects to the latest version. The steps required will depend on which device, along with which software packages are involved. At a high level, it is recommended to pay attention to the following:

- CCS migration
- DSP/BIOS to SYS/BIOS
- Toolchain: compiler, linker
- RTSC and XDC
- COFF vs ELF
- Processor SDK
- NDK Migration

2 CCS Migration

Code Composer Studio Downloads

Licensing: CCS V7 and later versions are Technology Software Publicly Available (TSPA) compliant. CCS V4, V5, V6, V7, V8, V9 are all free of charge.

2.1 Deprecation Notices

- All 32-bit versions of Code Composer Studio are deprecated.
  - Windows since CCSv9.0.0
  - Linux since CCSv6.2.0
  - macOS never had a 32-bit version
- Spectrum Digital XDS510USB JTAG debuggers and any other 32-bit emulators are not supported with 64-bit CCS installations.
- Ubuntu 14.04 support was deprecated with the release of CCSv8.
- Ubuntu 12.04LTS support is deprecated with the release of CCSv8.
- Windows XP support is deprecated with the release of CCSv7.

Eclipse is an integrated development environment (IDE) used in computer programming. It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in Java.

CCS v3 to CCS v5 migration

2.2 CCS Training Within CCS

Training material for Code Composer Studio is now integrated into Resource Explorer inside CCS or dev.ti.com. Under the Development Tools → Integrated Development Environments → Code Composer Studio section, you can explore all of the training material available for CCS including workshops, training modules and videos.

Click “View”, then “Resource Explorer”. Figure 2-1 shows the screenshots from CCS v9.3.

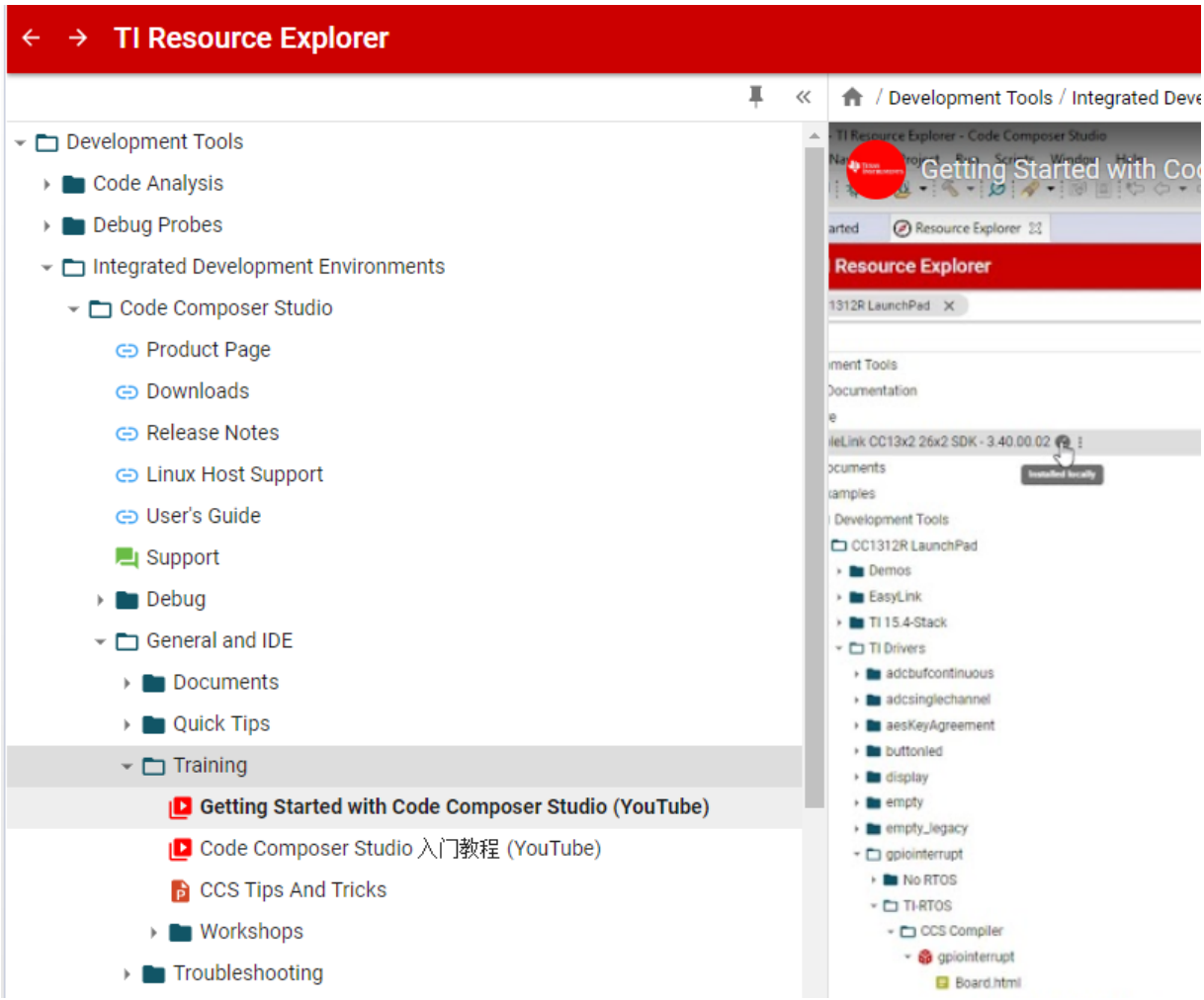

Figure 2-1. CCS Training With TI Resource Explorer

Under debugging:

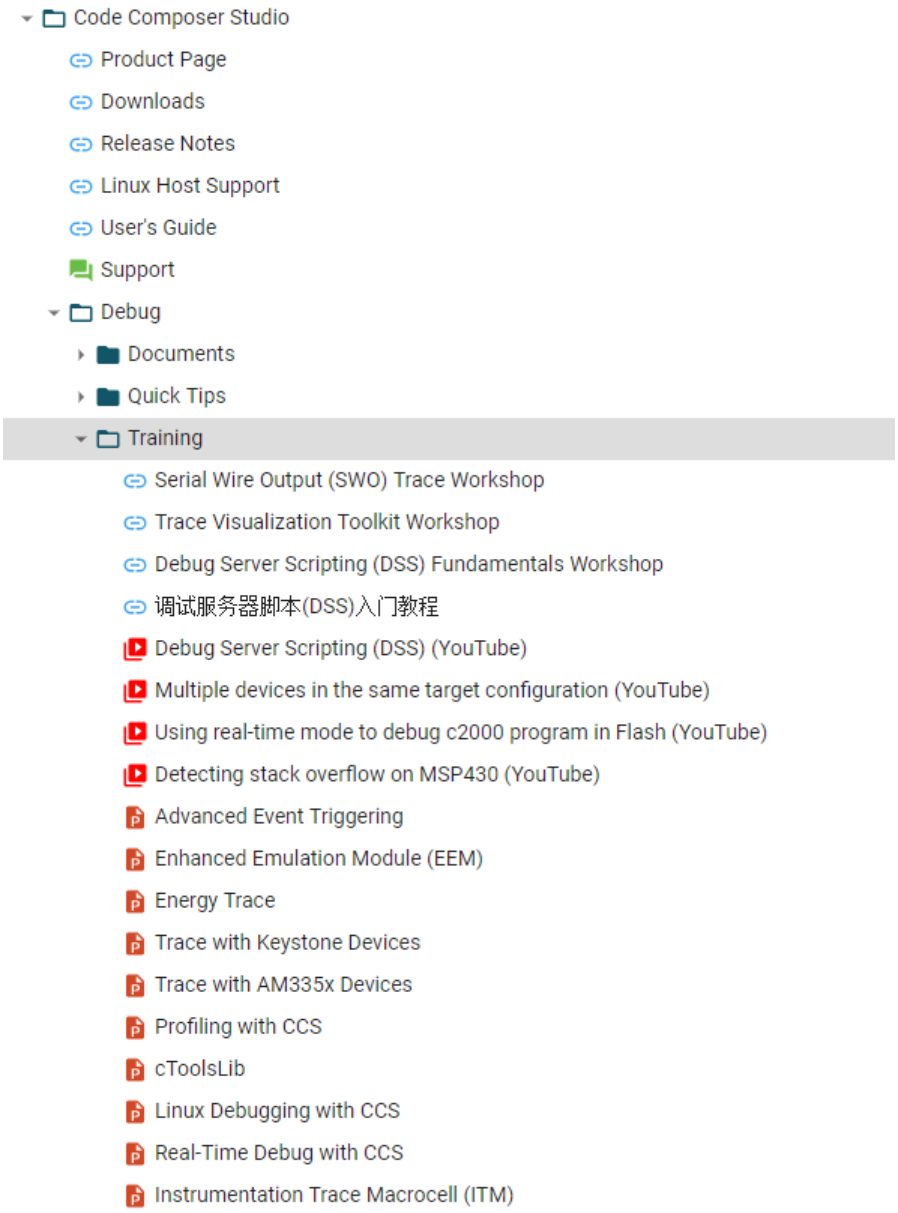

Figure 2-2. CCS Training on Debug Topics

Eclipse-based framework was introduced in CCS v4.

2.3 CCS Legacy Project Import Wizard

For more details on the CCS Legacy Project Import Wizard, see https://software-dl.ti.com/ccs/esd/documents/ccs_legacy-project-import.html.

3 DSP/BIOS vs SYS/BIOS

- TI-RTOS (SYS/BIOS) release
- *SYS/BIOS (TI-RTOS Kernel) User's Guide*

For TI-RTOS download information, see https://software-dl.ti.com/dsps/dsps_registered_sw/sdo_sb/targetcontent/bios/index.html.

3.1 Migrating Old Projects to the Latest Tools

There is a migration guide that shows how to migrate from DSP/BIOS to SYS/BIOS. You may find the guide useful from a “what changed” perspective. For more information, see *Migrating a DSP/BIOS 5 Application to SYS/BIOS 6*.

Getting started with TI-RTOS training series

4 Toolchain: CGT, Compiler, Linker

Code Generation Tools for Texas Instruments Processors: Downloads

CCS comes with TI compilers already but each will have a fixed version.

The C6000 CGT v8.3 is a NEW compiler:

- v8.3 supports C6400+, C6740, and C6600 in ELF EABI mode only
- v8.3 supports the C++14 Standard ISO/IEC 14882:2014, while C++03 is no longer supported
- C++ obj code generated by older compilers is not compatible with v8.0+ RTS obj libraries
- v8.3 offers comparable performance to v7.4. Performance may vary based on the application
- v7.4.x will continue to support (long-term) all processor variants in ELF EABI or COFF ABI mode

Customers should use CGT v8.3 if you are:

- Developing new applications using OpenCL, OpenMP, or HPC-MCSDK
- Developing new applications that utilize new compiler features only available in v8.0 and above (Native Vector Types, for example)

Customers should use CGT v7.4.x if you are:

- Maintaining an existing code base that you do not want or need to transition to v8.3 in the near-term
- Developing new applications or maintaining existing applications that use the COFF ABI
- Developing new applications or maintaining existing applications on C6200, C6400, C6700, C6700+, or Tesla

You can use older version of the build tools with later versions of CCS.

The old version of compilers does not come with CCS; you have to install it separately. Either individually directly or you can install old versions directly from the CCS IDE: Installing New Software.

5 RTSC and XDC

The Real-Time Software Components (RTSC) project provides foundational tools and low-level runtime content to enable component-based development using the C language targeting all embedded platforms. With RTSC we can enjoy higher-level programming and higher-levels of performance with RTSC.

The starting point for the RTSC project will include the XDC (eXpress DSP Components) tools currently available from Texas Instruments as a free download. XDC tools is a product that contains all of the tools necessary to create, test, deploy, install, and use RTSC components.

The main benefit of XDC is that it standardizes the delivery of target content and makes it easier for target content to be included in applications.

The XDC users are divided into developers we call "consumers' and "producers". Consumer integrate content packages: DSP algorithms, device drivers, TCP/IP stacks, real-time OSes, and so forth into their applications. Producers create the packages used by consumers.

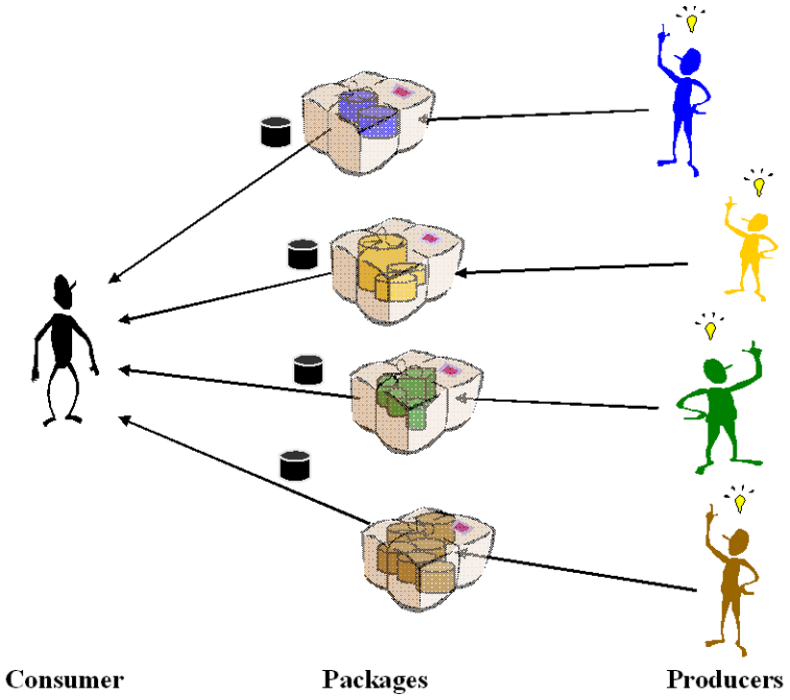

Figure 5-1. Users of XDC

An XDC "package" is a named collection of files that form a unit of versioning, upate, and delivery from a producer to a consumer. Each package is embodied as a specially-named directory (and its contents) within a file system. Packages are the focal point for managing content throughout its life-cycle. All packages are built, tested released, and deployed as a unit.

Here's a link: https://www.eclipse.org/rtsc/ Click "User's Guide".

Overview: http://rtsc.eclipseprojects.io/docs-tip/Overview_of_RTSC.

XDC releases: http://software-dl.ti.com/dsps/dsps_public_sw/sdo_sb/targetcontent/rtsc/.

Typically all RTSC projects are also CCS projects. The only difference is that they use SYSBIOS, TI-RTOS or a SDK that depends on the RTSC component.

6 COFF vs ELF

The term ABI stands for Application Binary Interface. The ABI specifies how a compiler and linker should handle various things like register assignment, calling convention, type sizes, and object file format. The conventions specified by an ABI make it possible for separately compiled object files and libraries to be linked together into a cohesive executable. An ABI named EABI, Embedded Application Binary Interface was introduced in 2010.

The main (but not only) difference between COFF ABI and EABI is the object file format. COFF ABI uses COFF, Common Object File Format, and EABI uses ELF, Executable and Linking Format.

The first version of the C6000 compiler to support EABI is version 7.2.0. The last version to support COFF ABI is version 7.4.24.

Executable and Linkable Format (ELF) has more debugging capabilities than Common Object File Format (COFF).

Regarding how to change from COFF ABI to EABI:

- First make sure all of your libraries, and any other software your project depends on, have EABI variants available. If not, you have to either not use those libraries, or stay on COFF ABI.

TI compiler switch: --abi=eabi or –abi=coffabi.

Linking COFF and ELF object files together is not possible.

With regard to migrating your own code from COFF ABI to EABI, see the C6000 EABI Migration wiki.

7 Processor SDK

For release, see the E2E thread: Sitara & DSP Announcement on Software.

Note

To stay up to date with the latest bug fixes and features, it is recommended to click "Alert Me" on the SDK download page to get an e-mail notification when a new SDK is released.

The recommendation is, on the particular product page for the SDK download, click "Alert Me" to get automatic email alert when a new release comes out.

Useful SDK training material is located at: Processor SDK Training Series.

In particular, this one is useful for RTOS: 1.5 Application Development Using Processor SDK RTOS Description

Figure 7-1 shows the approach that protects software investments made by embedded developers on existing TI parts, as all software releases for future TI SoC platforms will use this software methodology. From application perspective even if the underlying SoC is different, the API interfaces to functional CSL and LLD remains the same, which allows you to reuse the application software even when the underlying software may be different. Due to this approach to software, application developer who have developed their software on one of TI`s Processors will not need to re-learn working with the TI software.

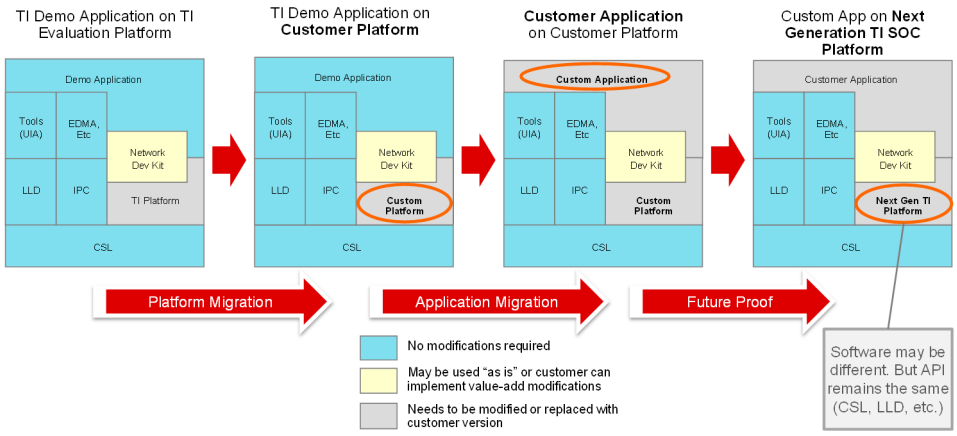

Figure 7-1. Maximize Software Reuse

Figure 7-2 shows the typical application development flow that a software developer would go through while using Processor SDK RTOS. As you go through the presentation, this development flow provides a little more detail on the flow of the training at each of these stages.

You typically start off by purchasing an Evaluation platform on which the processor functionality has been validated and then downloading the software environment required on the host to start their development.

Once you have the evaluation module in hand, you need to follow some common hardware Setup steps in order to run the software on your evaluation platforms. Such as, setting up the development environment by hooking up an emulator, which may optionally involve hooking up cables for connectivity through universal asynchronous receiver/transmitter (UART), USB interfaces. Typically, the EVM kit comes with the Quick start guide instructions that provide steps to setup the EVM and run an out of box demonstration on the platform.

Once the hardware is setup to run the software, go though the process of setting up the host development environment and then run some simple example code to verify the EVM is functional. For instance, running hello world examples on cores, learn to run RTOS applications and check some basic functionality on the EVM like blinking LEDs using general-purpose input/output (GPIOs) and checking for UART, USB and Network connectivity on the EVM.

After checking out the basic functionality on the EVM, it is recommended to run demo applications provided in the Processor SDK. These demos integrate multiple components of SDK and highlight the device features by creating a real world system use case.

More details in application development using drivers is included in the SDK. Some key elements in the SDK are discussed that will aid you to create your own application.

The Portability aspect of your application while migrating from the TI evaluation platform to your custom application board is discussed and components of the Processor SDK are looked at that makes the software easier to port.

Finally, look at customizing your software after migrating to the custom platform and provide some pointers that will help with your system integration.

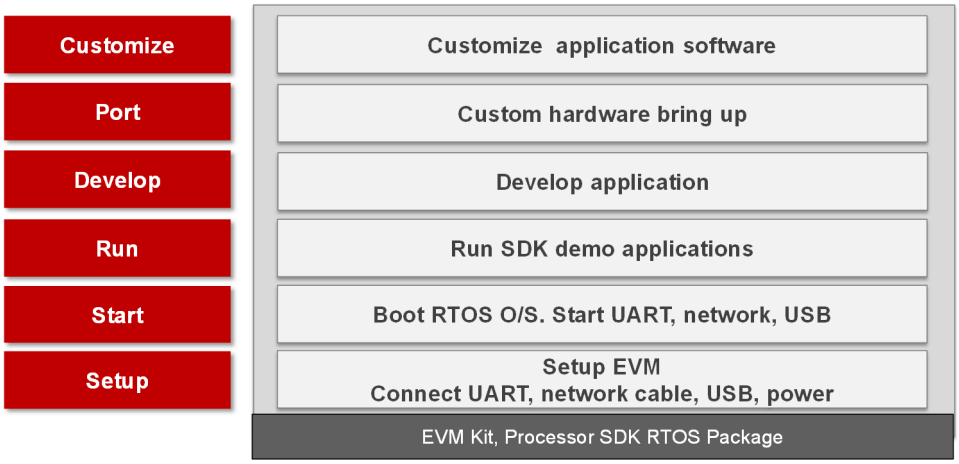

Figure 7-2. Typical Development Flow

8 NDK Migration

The NDK migration guide can be found in the installed Processor SDK package. For example: C:/ti/ndk_3_61_01_01/docs/ndk/NDK_2_to_3_Migration_Guide.html.

On BIOS configuration, there are some important configuration that needs to be kept. For example, on C6657 platform:

```
/* Load the CSL package */
var devType                             = "c6657";
var Csl                                 = xdc.useModule('ti.csl.Settings');
Csl.deviceType                          = devType;
Csl.useCSLIntcLib                       = true;
/* Load the OSAL package */
var osType = "tirtos"
var Osal = xdc.useModule('ti.osal.Settings');
Osal.osType = osType;
/* Load the QMSS package */
var Qmss                        =   xdc.loadPackage('ti.drv.qmss');
/* Load the EMAC packages */
var Emac = xdc.loadPackage('ti.drv.emac');
Emac.Settings.socType = devType;
var socType           = "c6657";
var Nimu            = xdc.loadPackage('ti.transport.ndk.nimu');
Nimu.Settings.socType  = socType;
/*
** Use this load to configure NDK 2.2 and above using RTSC. In previous versions of
** the NDK RTSC configuration was not supported and you should comment this out.
*/
var Ndk       = xdc.loadPackage('ti.ndk.config');
var Global    = xdc.useModule('ti.ndk.config.Global');
/*
** This allows the heart beat (poll function) to be created but does not generate the stack threads
**
** Look in the cdoc (help files) to see what CfgAddEntry items can be configured. We tell it NOT
** to create any stack threads (services) as we configure those ourselves in our Main Task
** thread hpdspuaStart.
*/
Global.enableCodeGeneration = false;
```

9 References

- Training
- Processors E2E Forum
- Processor SDK for OMAPL138 Processors for Linux and TI-RTOS Support
- Processor SDK for 66AK2Ex Processors - Linux and TI-RTOS Support
- Processor SDK for 66AK2Gx Processors - Linux and TI-RTOS Support
- Processor SDK for 66AK2HX Processors - Linux and TI-RTOS Support
- TMS320C6657 product folder
- TMS320C6678 product folder
- Texas Instruments: *SYS/BIOS (TI-RTOS Kernel) User's Guide*
- Texas Instruments: *Migrating a DSP/BIOS 5 Application to SYS/BIOS 6*

10 Revision History

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

Changes from Revision * (May 2020) to Revision A (February 2021)	Page
• Updated the numbering format for tables, figures, and cross-references throughout the document	1

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), 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, regulatory 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 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.

TI objects to and rejects any additional or different terms you may have proposed.

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