

TI-RTOS 2.20 for CC13xx/CC26xx SimpleLink™ Wireless MCUs

Getting Started Guide



Literature Number: SPRUHU7D
June 2016

Preface	3
1 About TI-RTOS	4
1.1 What is TI-RTOS?	4
1.2 What are the TI-RTOS Components?	5
1.3 How Can I Find Example Projects?	6
1.4 What Compilers and Targets are Supported?	6
1.5 What Boards and Devices Have TI-RTOS Driver Examples?	7
1.6 What Drivers Does TI-RTOS Include?	7
1.7 For More Information	8
2 Installing TI-RTOS	10
2.1 System Requirements	10
2.2 Installing Code Composer Studio	11
2.3 Installing TI-RTOS in Code Composer Studio	11
2.4 Installing TI-RTOS for Use in IAR Embedded Workbench	12
2.5 Installing TI-RTOS as a Standalone Product	12
3 Examples for TI-RTOS	13
3.1 Creating Example Projects Using the Resource Explorer in CCS	14
3.1.1 Creating an Empty TI-RTOS Project	16
3.1.2 Creating Examples to Build via a Command Line	16
3.2 Creating Examples with IAR Embedded Workbench	17
3.3 Driver Examples: Readme Files and Common Features	17
3.4 CC1310/CC2650 SimpleLink Development Kit Settings & Resources	19
3.5 CC1310 SimpleLink LaunchPad Settings & Resources	21
3.6 CC1350 SimpleLink LaunchPad Settings & Resources	23
3.7 CC1350 SimpleLink SensorTag Settings & Resources	25
3.8 CC2650 SimpleLink LaunchPad Settings & Resources	27
3.9 CC2650 SimpleLink SensorTag Settings and Resources	28
3.10 <Board>.c File	30
3.11 Updating driverlib for CC13xx/CC26xx	30
3.11.1 Building with CCS	30
3.11.2 Building with IAR	31
3.11.3 Building with a Makefile	32
4 Configuring TI-RTOS	34
4.1 Starting the Configuration Tool	35
4.2 Configuring TI-RTOS Drivers	36
4.2.1 Configuring System Support	36
4.2.2 Configuring ROM for CC13xx/CC2650	37
4.3 Configuring Components of TI-RTOS	37
Index	38

Read This First

About This Manual

This manual describes TI-RTOS for SimpleLink™ Wireless MCUs. The version number as of the publication of this manual is v2.20.

Notational Conventions

This document uses the following conventions:

- Program listings, program examples, and interactive displays are shown in a special typeface. Examples use a bold version of the special typeface for emphasis.

Here is a sample program listing:

```
#include <xdc/runtime/System.h>
int main(void)
{
    System_printf("Hello World!\n");
    return (0);
}
```

- Square brackets ([and]) identify an optional parameter. If you use an optional parameter, you specify the information within the brackets. Unless the square brackets are in a **bold** typeface, do not enter the brackets themselves.

Trademarks

Registered trademarks of Texas Instruments include Stellaris and StellarisWare.

Trademarks of Texas Instruments include: the Texas Instruments logo, Texas Instruments, TI, TI.COM, C2000, C5000, C6000, Code Composer, Code Composer Studio, Concerto, controlSUITE, DSP/BIOS, MSP430, MSP430Ware, MSP432, SimpleLink, Sitara, SPOX, TI-RTOS, Tiva, TivaWare, TMS320, TMS320C5000, TMS320C6000, and TMS320C2000.

ARM is a registered trademark, and Cortex is a trademark of ARM Limited.

Windows is a registered trademark of Microsoft Corporation.

Linux is a registered trademark of Linus Torvalds.

IAR Systems and IAR Embedded Workbench are registered trademarks of IAR Systems AB.

All other brand or product names are trademarks or registered trademarks of their respective companies or organizations.

June 6, 2016

About TI-RTOS

This chapter provides an overview of TI-RTOS for SimpleLink™ Wireless MCUs.

Topic	Page
1.1 What is TI-RTOS?	4
1.2 What are the TI-RTOS Components?	5
1.3 How Can I Find Example Projects?	6
1.4 What Compilers and Targets are Supported?	6
1.5 What Boards and Devices Have TI-RTOS Driver Examples?	7
1.6 What Drivers Does TI-RTOS Include?	7
1.7 For More Information	8

1.1 What is TI-RTOS?

TI-RTOS is a scalable, one-stop embedded tools ecosystem for TI devices. It scales from a real-time multitasking kernel (SYS/BIOS) to a complete RTOS solution including additional middleware components and device drivers. By providing essential system software components that are pre-tested and pre-integrated, TI-RTOS enables you to focus on creating your application.

TI-RTOS is *not* installed automatically as part of the Code Composer Studio v6.x installation. You can install TI-RTOS from the CCS App Center (choose **View > CCS App Center** in CCS). Choose the version of TI-RTOS for your device family. If you use devices in multiple families, you can install multiple TI-RTOS versions. See [Section 2.3](#) for details.



If you do not use CCS, you can [download and install TI-RTOS as a standalone product](#) (see [Section 2.5](#)). In addition to the Texas Instruments Code Generation Tools, TI-RTOS includes support for the IAR tool chain (see [Section 2.4](#)).

TI-RTOS is provided with full source code and requires no up-front or runtime license fees.

1.2 What are the TI-RTOS Components?

TI-RTOS contains its own source files, pre-compiled libraries (both instrumented and non-instrumented), and examples. Additionally, TI-RTOS contains a number of components within its "products" subdirectory. The components of TI-RTOS for SimpleLink Wireless MCUs are as follows.

Table 1–1. TI-RTOS Components

TI-RTOS Component	Name	Documentation Location
TI-RTOS	TI-RTOS examples	Chapter 3 of this Getting Started Guide
TI-RTOS Kernel	SYS/BIOS	<i>SYS/BIOS (TI-RTOS Kernel) User's Guide</i> -- SPRUEX3
TI-RTOS Drivers & Board Support	Drivers and CC13xxWare and CC26xxWare	<i>TI-RTOS User's Guide</i> -- SPRUHD4
TI-RTOS Instrumentation	UIA	<i>System Analyzer User's Guide</i> -- SPRUH43
TI-RTOS File System	FatFS	<i>TI-RTOS User's Guide</i> -- SPRUHD4

The components in the "products" subdirectory are:

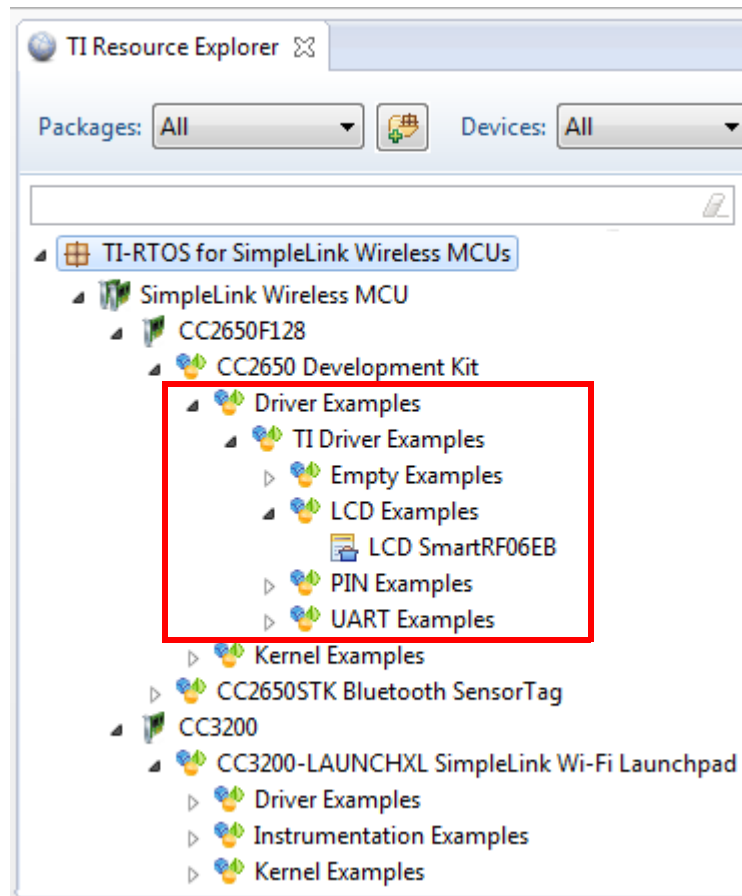
- **TI-RTOS Kernel — SYS/BIOS.** SYS/BIOS is a scalable real-time kernel. It is designed to be used by applications that require real-time scheduling and synchronization or real-time instrumentation. It provides preemptive multi-threading, hardware abstraction, real-time analysis, and configuration tools. SYS/BIOS is designed to minimize memory and CPU requirements on the target.
- **TI-RTOS Drivers and Board Support.** TI-RTOS includes drivers for a number of peripherals. These drivers are thread-safe for use with the TI-RTOS Kernel. The drivers have a common framework, so using multiple drivers in your application is easy. Both instrumented and non-instrumented versions of the drivers are provided. Board support files to configure the drivers is provided for several targets.
- **TI-RTOS Instrumentation — UIA.** The Unified Instrumentation Architecture (UIA) provides target content that aids in the creation and gathering of instrumentation data (for example, Log data).
- **CC13xxWare.** This software component contains low-level drivers for use with CC13xx targets.
- **CC26xxWare.** This software component contains low-level drivers for use with CC26xx targets.
- **XDCtools.** This core component provides the underlying tooling for configuring and building TI-RTOS and its components. XDCtools is installed as part of CCS v6.x. If you install TI-RTOS outside CCS, a compatible version of XDCtools is installed automatically. XDCtools is installed in a directory at the same level as TI-RTOS, not in the "products" directory of the TI-RTOS installation.

1.3 How Can I Find Example Projects?

TI-RTOS and its components provide numerous examples that you can import using the **Resource Explorer** in Code Composer Studio (CCS). These examples use TI-RTOS and its components and have all the settings needed for your device. Expand the tree in the Resource Explorer to see the examples that are available for your device.

- **Driver Examples** are TI-RTOS driver examples.
- **Instrumentation Examples** are UIA examples.
- **Kernel Examples** are the SYS/BIOS examples.

Follow the steps in [Section 3.1](#) to import, build, and run these examples.



1.4 What Compilers and Targets are Supported?

The following code generation tool (compilers and linkers) versions are supported. The versions listed are recommended because they were used to build the TI-RTOS libraries and to perform testing. More recent versions are expected to be compatible.

- **Texas Instruments:** ARM CodeGen Tools v5.2.2
- **IAR Workbench for ARM:** 7.40.3

The configuration uses a "target" specification during the build. This specification is sometimes called the "RTSC target." The targets supported are:

- CC13xx
 - ti.targets.arm.elf.M3
 - iar.targets.arm.M3
- CC26xx
 - ti.targets.arm.elf.M3
 - iar.targets.arm.M3

1.5 What Boards and Devices Have TI-RTOS Driver Examples?

Currently, TI-RTOS provides driver examples for the following boards:

Family	Device on Board	Board
ARM	CC1310	CC1310 SimpleLink Development Kit + SmartRF06 development base board
ARM	CC1310	CC1310 LaunchPad
ARM	CC1350	CC1350 LaunchPad
ARM	CC1350	CC1350 SimpleLink SensorTag
ARM	CC2650	CC2650 SimpleLink Development Kit + SmartRF06 development base board
ARM	CC2650	CC2650 LaunchPad
ARM	CC2650	CC2650 SimpleLink SensorTag

Examples are provided specifically for the supported boards, but libraries are provided for each of these device families, so that you can port the examples to similar boards. Porting information for TI-RTOS is provided on the [Texas Instruments Wiki](#).

1.6 What Drivers Does TI-RTOS Include?

TI-RTOS includes drivers for the following peripherals. These drivers are in the `<install_dir>/products/tidrivers_<version>/packages/ti/drivers` directory. TI-RTOS examples show how to use these drivers. Note that all of these drivers are built on top of CC13xxWare and CC26xxWare.

The TI-RTOS installation installed drivers for multiple device families. The following list indicates which drivers can be use with CC13xx and CC26xx targets:

- **Crypto.** Advanced Encryption Standard (AES) driver for data encryption and decryption.
- **GPIO.** API set intended to be used directly by the application or middleware to manage the GPIO interrupts, pins, and ports.
- **I²C.** API set intended to be used directly by the application or middleware.
- **LCD.** Driver for the LCDDogm1286 LCD controller.
- **PIN.** Driver for Pin interrupts.
- **Power.** Power management framework. See the *TI-RTOS Power Management User's Guide* (SPRU118).
- **PWM.** API set intended to be used directly by the application or middleware to generate Pulse Width Modulated signals.

- **RF.** Radio driver for CC13xx devices.
- **SPI.** API set intended to be used directly by the application or middleware to communicate with the Serial Peripheral Interface (SPI) bus. SPI is sometimes called SSI (Synchronous Serial Interface).
- **UART.** API set intended to be used directly by the application to communicate with the UART.

1.7 For More Information

To see release notes for a component, go to the subdirectory for that component within the TI-RTOS products directory. For example,

C:\ti\tirtos_simplelink_2_##_##_##\products\bios_6_40_##_## contains release notes for SYS/BIOS.

To see user guide PDFs and other documentation for a component, go to the "docs" subdirectory within the directory that contains the release notes.

To learn more about TI-RTOS and its components, refer to the following documentation:

- **TI-RTOS**
 - [TI-RTOS User's Guide \(SPRUHD4\)](#)
 - In the TI-RTOS Release Notes, follow the **Documentation Overview** link. In the Documentation Overview page, choose the **TI-RTOS Drivers Runtime APIs (doxygen)** item.
 - [TI-RTOS on the Texas Instruments Wiki](#)
 - [TI-RTOS forum on TI's E2E Community](#)
 - [TI-RTOS Porting Guide](#)
 - [Embedded Software Download Page](#)
- **Code Composer Studio (CCS)**
 - CCS online help
 - [CCSv6 on the Texas Instruments Wiki](#)
 - [Code Composer forum on TI's E2E Community](#)
- **SYS/BIOS**
 - [SYS/BIOS User's Guide \(SPRUEX3\)](#)
 - SYS/BIOS API and configuration reference. In the TI-RTOS Release Notes, follow the **Documentation Overview** link. In the Documentation Overview page, choose the **TI-RTOS Kernel Runtime APIs and Configuration (cdoc)** item.
 - [SYS/BIOS on the Texas Instruments Wiki](#)
 - [TI-RTOS forum on TI's E2E Community](#)
 - [SYS/BIOS 6.x Product Folder](#)
- **XDCtools**
 - [SYS/BIOS User's Guide \(SPRUEX3\)](#)
 - XDCtools online reference. Open from CCS help or run <xdc_install>/docs/xdctools.chm.
 - [RTSC-Pedia Wiki](#)
 - [TI-RTOS forum on TI's E2E Community](#)

- **UIA**
 - [System Analyzer User's Guide \(SPRUH43\)](#)
 - UIA API and configuration reference. In the TI-RTOS Release Notes, follow the **Documentation Overview** link. In the Documentation Overview page, choose the **TI-RTOS Instrumentation Runtime APIs and Configuration (cdoc)** item.
 - [System Analyzer on the Texas Instruments Wiki](#)
- **FatFS API**
 - [Open source documentation](#)
 - [TI-RTOS User's Guide \(SPRUHD4\)](#)
- **General microcontroller information**
 - [Microcontrollers forum on TI's E2E Community](#)
- **SimpleLink resources**
 - [SimpleLink WiFi Overview](#)
 - [CC13xx & CC26xx Tools Overview](#)
 - [CC2650 Development Kit](#)
- **I²C**
 - [Specification](#)

Installing TI-RTOS

This chapter covers the steps to install TI-RTOS within Code Composer Studio or as a standalone software product.

Topic	Page
2.1 System Requirements	10
2.2 Installing Code Composer Studio	11
2.3 Installing TI-RTOS in Code Composer Studio	11
2.4 Installing TI-RTOS for Use in IAR Embedded Workbench.	12
2.5 Installing TI-RTOS as a Standalone Product	12

2.1 System Requirements

The Microsoft Windows version of TI-RTOS can be installed on systems running Windows 8, Windows 7, Windows Vista, or Windows XP (SP2 or SP3).

The Linux version of TI-RTOS can be installed on systems that are running Linux RedHat v4 and higher or Ubuntu v10.04 and higher.

Separate versions of TI-RTOS are available for various Texas Instruments device families.

In order to install TI-RTOS, you must have at least 1 GB of free disk space. (If you have not yet installed Code Composer Studio, you will also need at least 4 GB of disk space for that installation.)

2.2 Installing Code Composer Studio

TI-RTOS is used in conjunction with Code Composer Studio 6.1 or higher. (TI-RTOS can also be used with the IAR Embedded Workbench IDE. See [page 12](#) for more information.) CCS is available for Microsoft Windows and Linux.

For Windows installations, we recommend that you install CCS in the default installation directory of `c:\ti`. If you install in `c:\Program Files` (or `c:\Program Files (x86)` with Windows 7), you are likely to run into problems related to Windows security permissions.

Note: Do not install CCS in a location that contains any spaces in the full path. For example, CCS should not be installed in `c:\Program Files`. Makefiles may not function correctly with directory paths that include spaces.

To install CCS 6.x, go to the ["Download CCS" page on the Texas Instruments wiki](#) and follow a link to download the software for your license type. For multi-user licenses, see the [CCS product page](#).

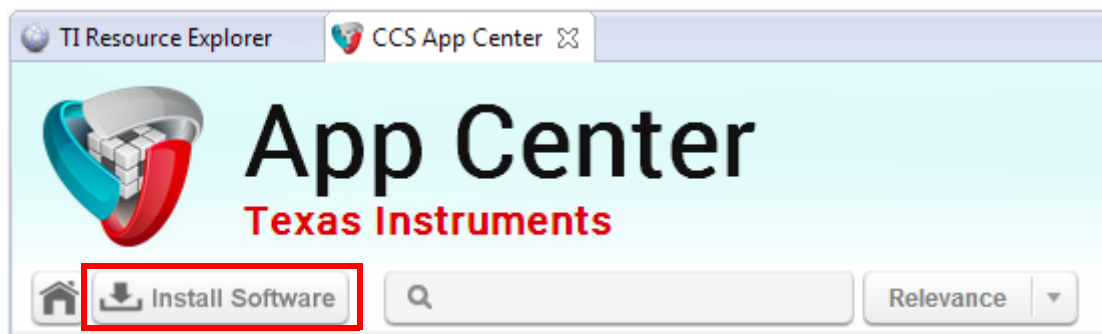
Run the installer, and answer the prompts as appropriate.

2.3 Installing TI-RTOS in Code Composer Studio

TI-RTOS is *not* installed automatically as part of the Code Composer Studio v6.x installation. Instead, you can install it through the CCS App Center as described in this section, for use in IAR Workbench as described in [Section 2.4](#), or as a standalone product as described in [Section 2.5](#).

Follow these steps to install TI-RTOS in CCS:

1. Run CCS v6.1 or higher.
2. Choose **View > CCS App Center** in CCS.
3. Select the version of TI-RTOS for your device family. If you use devices from multiple families, you can select multiple TI-RTOS versions.
4. Click the **Install Software** button near the top of the App Center view.



5. Answer the prompts as necessary to complete the TI-RTOS installation.
6. Restart CCS in order for TI-RTOS and its components to be available.

2.4 Installing TI-RTOS for Use in IAR Embedded Workbench

You can install TI-RTOS for use with IAR Embedded Workbench as follows:

1. Install IAR Embedded Workbench for Texas Instruments ARM devices. See [Section 1.4](#) for the supported versions of IAR Workbench.
2. [Download the Windows installer](#) for TI-RTOS for SimpleLink. For example, `tirtos_setupwin32_simplelink_2_##_##_##.exe`.
3. Run the downloaded file to install the full TI-RTOS product. You can install TI-RTOS in a standalone directory. Installing in a directory path that contains spaces, such as `C:\Program Files (x86)`, is not supported.

Note: TI-RTOS installs the core functionality of the XDCtools component if you have not already installed the necessary version as part of a CCS installation. TI-RTOS places XDCtools in a separate directory at the same level where you install TI-RTOS. For example, if the TI-RTOS installation directory is `C:\ti\tirtos_simplelink_2_##_##_##`, the XDCtools directory will be `C:\ti\xdctools_3_31_##_##_core`.

Follow the instructions in [Section 3.1.2](#) to complete the installation of the TI-RTOS examples. See [Section 3.2](#) to build the examples with the IAR compiler and to load and run the examples with IAR Embedded Workbench.

2.5 Installing TI-RTOS as a Standalone Product

If you do not use Code Composer Studio, you can install TI-RTOS as a standalone product. In addition to compiling and linking with the Texas Instruments Code Generation Tools, TI-RTOS includes support for the IAR tool chain.

1. [Download the Windows or Linux installer](#) for TI-RTOS for the device family you use. For example, `tirtos_setupwin32_simplelink_2_##_##_##.exe` or `tirtos_setuplinux_simplelink_2_##_##_##.bin`.
2. Run the downloaded file to install TI-RTOS. You can install TI-RTOS in a standalone directory. Installing in a directory path that contains spaces, such as `C:\Program Files (x86)`, is not supported.

Note: TI-RTOS installs the core functionality of the XDCtools component if you have not already installed the necessary version as part of a CCS installation. TI-RTOS places XDCtools in a separate directory at the same level where you install TI-RTOS. For example, if the TI-RTOS installation directory is located in `C:\ti\tirtos_simplelink_2_##_##_##`, the XDCtools directory will be in `C:\ti\xdctools_3_31_##_##_core`.

Follow the instructions in [Section 3.1.2](#) to complete the installation of the TI-RTOS examples.

Examples for TI-RTOS

TI-RTOS comes with a number of examples that illustrate on how to use the individual components. This chapter explains how to create and use these examples.

Topic	Page
3.1 Creating Example Projects Using the Resource Explorer in CCS ..	14
3.2 Creating Examples with IAR Embedded Workbench	17
3.3 Driver Examples: Readme Files and Common Features	17
3.4 CC1310/CC2650 SimpleLink Development Kit Settings & Resources	19
3.5 CC1310 SimpleLink LaunchPad Settings & Resources	21
3.6 CC1350 SimpleLink LaunchPad Settings & Resources	23
3.7 CC1350 SimpleLink SensorTag Settings & Resources	25
3.8 CC2650 SimpleLink LaunchPad Settings & Resources	27
3.9 CC2650 SimpleLink SensorTag Settings and Resources	28
3.10 <Board>.c File	30
3.11 Updating driverlib for CC13xx/CC26xx	30

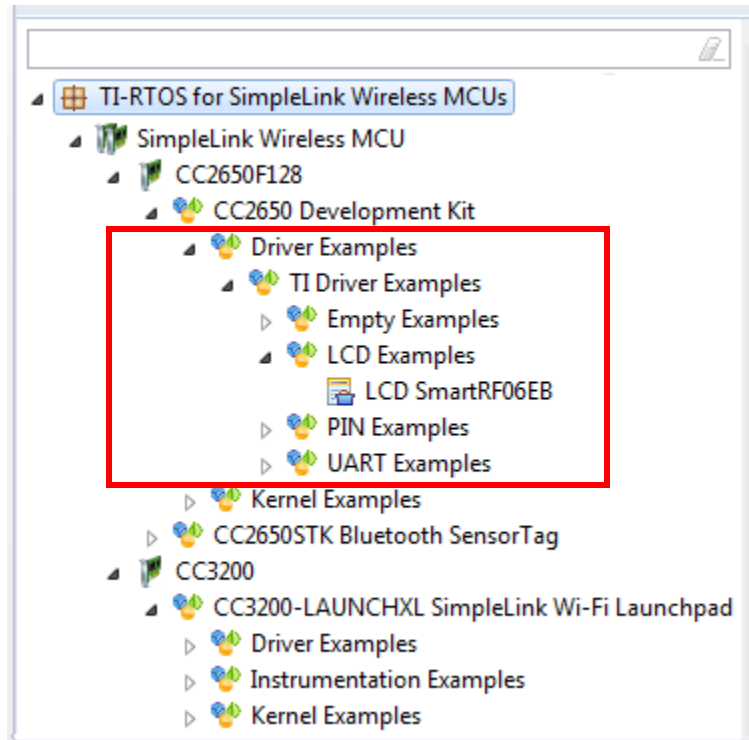
3.1 Creating Example Projects Using the Resource Explorer in CCS

You can use the **Resource Explorer** in Code Composer Studio (CCS) to create example projects that use TI-RTOS and its components and have all the settings needed for your device. Follow these steps:

1. Open CCS. If you do not see the Resource Explorer, make sure you are in the **CCS Edit** perspective and choose **View > Resource Explorer (Examples)** from the menus.

2. Type the name or part of the name of your device in the **enter search keyword** field to hide all the examples that don't apply to your device. Or, type "Driver Examples" to find driver examples.

3. Expand the tree until you see the examples for your device. Any **Driver Examples** listed are TI-RTOS driver examples. Any **Instrumentation Examples** listed are UIA examples. The **Kernel Examples** are the TI-RTOS Kernel (SYS/BIOS) examples.



4. Select the example you want to create. A description of the selected example is shown to the right of the example list.
5. Click the **Step 1** link in the right pane of the Resource Explorer to **Import the example project into CCS**. This adds a new project to your Project Explorer view. Once you have completed a step for a particular example and device, a green checkmark will be shown next to that step.

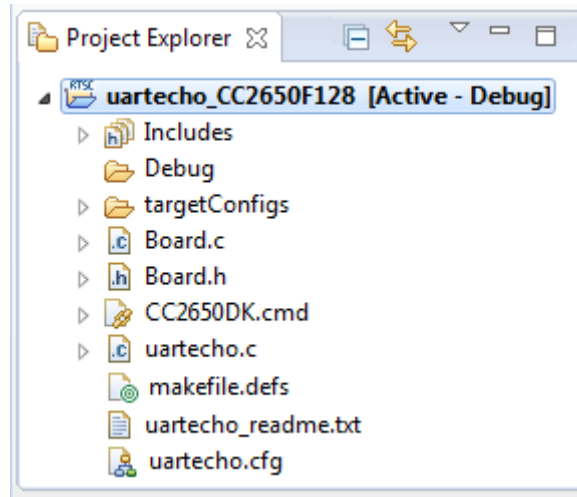
Step 1: Import the example project into CCS



*Click on the link above to import the project. The imported project is available in the **Project Explorer** view, expand the project node to browse the imported source files. To modify source code, double clicks on the source file within the project to open the source file editor.*

Note: CCS will not allow two projects with the same name in a workspace.

6. The project created will have a name with the format <example_name>_<board>. You can expand the project to see the source code, configuration, and other files in the project.



7. The page shown when you select an example in the Resource Explorer provides additional links to perform common actions with that example. Use the **Step 2** link when you are ready to build the project. If you want to change any build options, right click on the project and select **Properties** from the context menu. For example, you can change compiler, linker, and RTSC (XDCtools) options.

Step 2: [Build the imported project](#)



To change build options, right click on the project and select **Properties** from the context menu. To build the project, select the link above, or select the **Build** toolbar button, or select the **Project | Build Project** menu item.

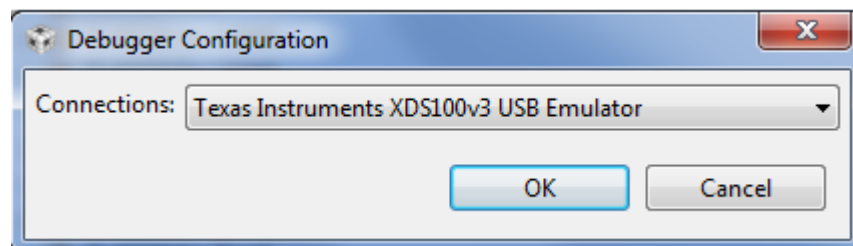
8. Use the **Step 3** link to change the connection used to communicate with the board. The current setting is shown in the Resource Explorer page for the selected example.

Step 3: [Debugger Configuration](#)



Connection: [Spectrum Digital XDSPRO USB Emulator](#)
Click on the link above to change the device connection. Additionally, this option is also available in the project properties.

9. You will see the Debugger Configuration dialog. Choose an emulator. When using the SmartRF06 development board, choose the XDS100v3. However, for the CC2650-BLE (SensorTag) with the DevPack debugger, choose the XDS110 USB Debug Probe.



10. Use the **Step 4** link to launch a debug session for the project and switch to the **CCS Debug Perspective**.

Step 4: [Debug the imported project](#)



*Click on the link above to launch a debug session for the **UART Echo** project and switch to the **CCS Debug Perspective**. Additionally, these are other methods to start a project debug session. Select the project in the **Project Explorer** view and click on the bug toolbar button. To relaunch a previous debug session, click on the small arrow beside the bug toolbar button and select one of the debug session from the history.*

3.1.1 Creating an Empty TI-RTOS Project

TI-RTOS provides blank projects you can use as a starting point for creating your own projects that utilize TI-RTOS. Both "Empty" and "Empty (Minimal)" versions are provided. The "Empty" version enables more kernel features and debug capabilities at the cost of large footprint. The "Empty (Minimal)" version disables various kernel features and debug capabilities to minimize the footprint. See the "Memory Usage with TI-RTOS" chapter in the [TI-RTOS User's Guide](#) (SPRUHD4) for details about techniques used to minimize the footprint.

Empty TI-RTOS driver projects can be created with the Resource Explorer (see [Section 3.1](#)).

After you create the example, the files in the empty project example include:

- Key C files: empty.c, <board>.c/.h
- Key configuration files: empty.cfg
- Linker command file: <board>.cmd

Add to the example as needed to implement your application.

3.1.2 Creating Examples to Build via a Command Line

TI-RTOS has a command-line utility called `examplesgen` that generates example projects along with the makefiles needed to build the examples for all supported tool chains—TI and IAR. The files are created in a location you specify on the command line. The `tirtos.mak` file in the top level directory of your TI-RTOS installation can be used to run `examplesgen`.

Note: If you installed TI-RTOS using the standalone installer ([Section 2.5](#) or [Section 2.4](#)), this step is not necessary because pre-generated examples are included as part of the installation for all supported boards and tool chains. The provided examples are located in the `TIRTOS_INSTALL_DIR\examples` directory. Pre-generated examples are not provided if you installed TI-RTOS through the CCS App Center.

You only need to perform these steps once:

1. If you installed TI-RTOS in a location other than the default location of `C:\ti`, edit the `tirtos.mak` file in the TI-RTOS installation directory. Modify the following variables as needed to make them point to the correct locations.
 - `DEFAULT_INSTALL_DIR`: Full path to the location where TI tools are installed.
 - `IAR_COMPILER_INSTALL_DIR`: Full path to the IAR code generation tools installation.

- TIRTOS_INSTALL_DIR: Full path to the TI-RTOS installation.
 - XDCTOOLS_INSTALL_DIR: Full path to the XDCtools installation.
2. If you plan to use TI-RTOS with IAR, set the IAR_BUILD variable in the `tirtos.mak` file to true.

```
IAR_BUILD ?= true
```

3. Open a command line window, and use the following commands to run the `examplesgen` utility. (If you installed TI-RTOS in a protected directory, you should run the command window as the administrator.)

```
> cd <tirtos_install>
> ..\xdctools_3_31_##_##_core\gmake -f tirtos.mak examplesgen DEST="YOURPATH"
```

For the destination path, use a UNIX-style path. That is, use forward slashes (/) instead of backslashes (\). For example, `DEST="C:/myfiles"`.

The output from this command is an `examples` directory tree containing folders for the supported boards. Each board folder contains folders for all the examples available for that board.

Examples for TI and IAR are generated for boards supported by TI-RTOS. Each board directory contains a `makedefs` file that can be modified to specify other installation paths or compiler/linker options and a `makefile` that can be used to build all the examples for that board. Each example directory has its own `makefile` that can be used to build that example specifically.

3.2 Creating Examples with IAR Embedded Workbench

For information about using TI-RTOS examples with IAR Embedded Workbench, see the wiki page on [Creating TI-RTOS Applications in IAR Embedded Workbench](#) on the Texas Instruments wiki.

3.3 Driver Examples: Readme Files and Common Features

Details about the driver examples are provided in the readme files in the example projects. There is a separate `<example_name>_readme` file for each of the examples. These files are added to your CCS project when you use the Resource Explorer to create a project. You can open the `<example_name>_readme` file within CCS. The `<example_name>_readme` files contain the following types of information:

- Actions performed by functions in the example.
- Hardware-specific descriptions of buttons, LEDs, etc...
- Which external components are (or may be) needed to run with particular examples.

There are several TI-RTOS example categories. The Empty and Empty (Minimal) projects are configured to make TI-RTOS available but do not contain specific code that uses TI-RTOS. The Demo examples use several peripherals working together. The remaining examples show how to use a specific peripheral.

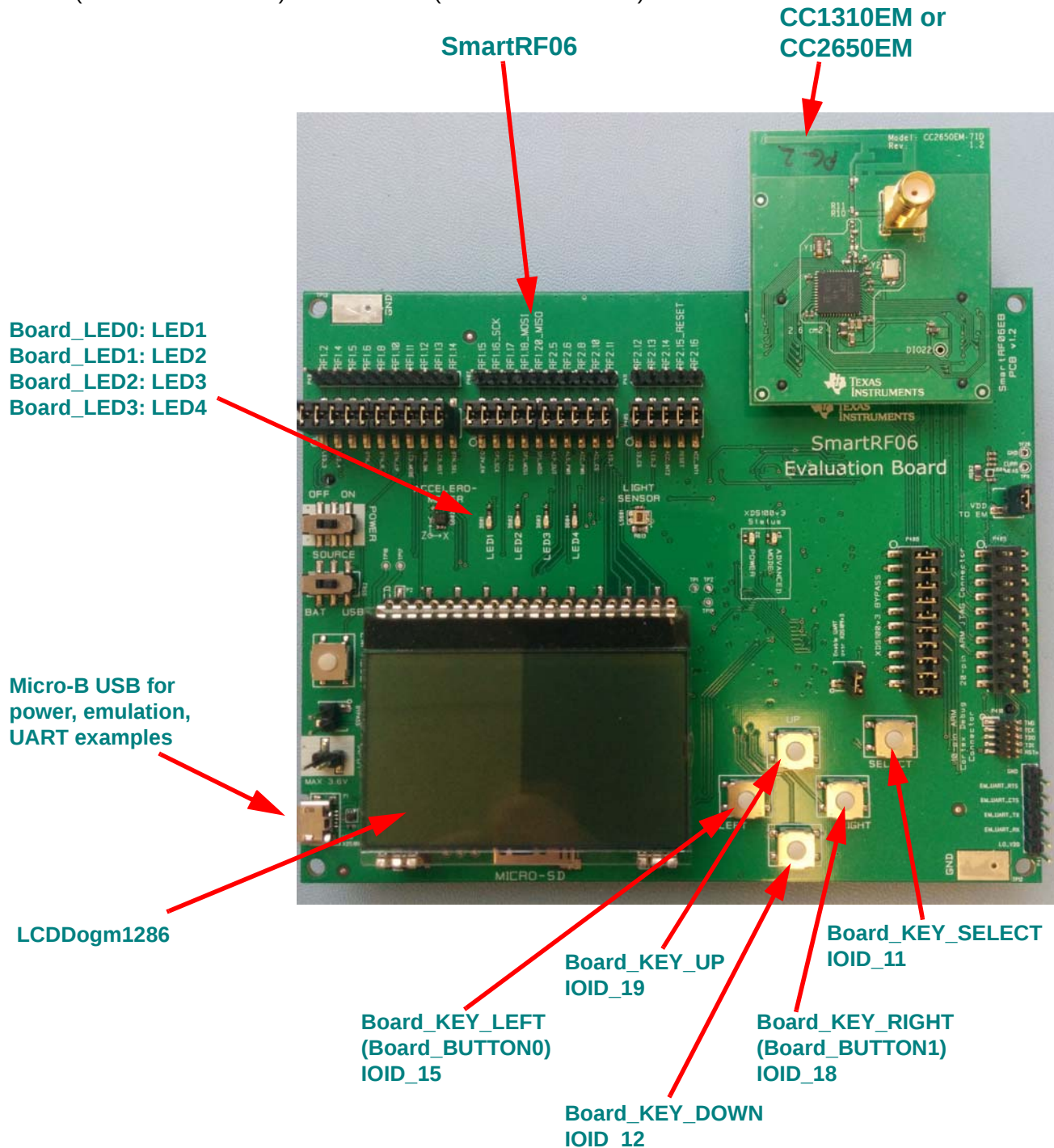
The Driver Examples share the following features:

- Most TI-RTOS driver examples use the SysMin System Support module. See the readme files in the individual example projects for details.
- The empty, demo, and most UART examples use the ti.uia.sysbios.LoggingSetup module with stop mode data collection. The UART Console example uses run-time data collection during Idle thread processing. For more details on data collection, see Chapter 2 of the [TI-RTOS User's Guide](#) (SPRUHD4).
- Driver Examples for a particular target all have the same <board>.c and <board>.h files. These files perform board-specific configuration of the drivers provided by TI-RTOS. For more details, see Chapter 4 of the [TI-RTOS User's Guide](#) (SPRUHD4).

The sections that follow list settings required to run the TI-RTOS examples on the supported boards. They also list the hardware resources that TI-RTOS and its dependent components use by default. Some of these resources offer flexible options, whereas others are fixed in the current design or implementation.

3.4 CC1310/CC2650 SimpleLink Development Kit Settings & Resources

The CC2650DK (Development Kit) consists of a a SmartRF06 base board and either a CC1310EM (CC1310F128 device) or CC2650EM (CC2650F128 device).



The Micro-B connector on the SmartRF06 can be used for power, debugger emulation, and UART communications.

Jumper Settings:

For the LCD example, use the default jumper settings, in which jumper blocks P403, P404, P405 are closed.

Switch Settings:

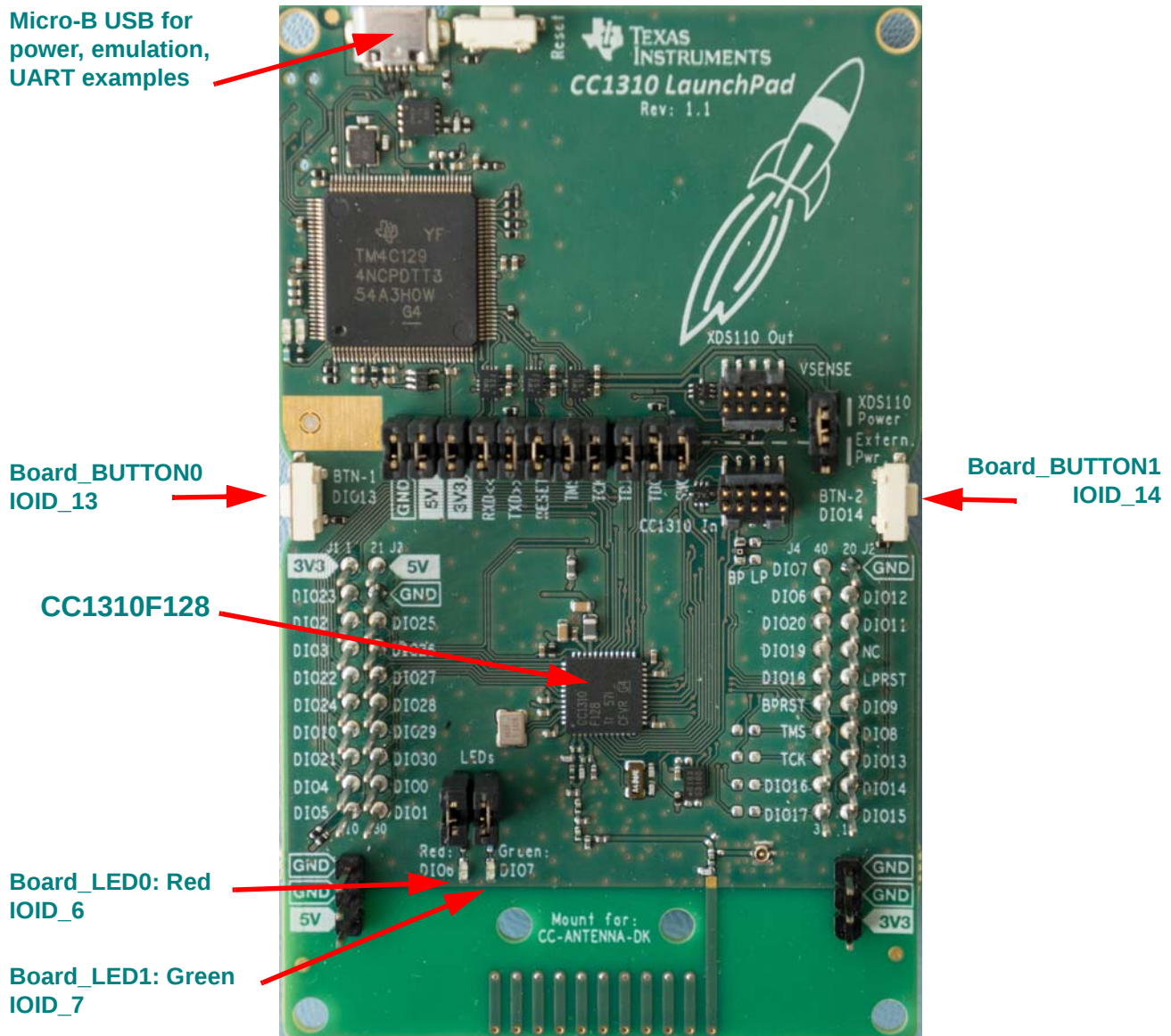
- LEFT: Some examples use IOID_15 as PIN input.
- RIGHT: Some examples use IOID_18 as PIN input.
- UP: Some examples use IOID_19 as PIN input.
- DOWN: Some examples use IOID_12 as PIN input.
- SELECT: Some examples use IOID_11 as PIN input.

Resources Used:

- **TI-RTOS Kernel (SYS/BIOS).** Uses the CC26xx's RTC timer and that timer's associated interrupts. The TI-RTOS Kernel manages the Interrupt Vector Table.
- **TI-RTOS.**
 - **Crypto:** A CC1310/CC2650 specific driver that uses the onboard AES Cryptoprocessor.
 - **PINs.** A CC1310/CC2650 specific driver PIN driver that uses 4 output pins for the onboard LEDs and 5 input pins for switches Board_KEY_LEFT, Board_KEY_RIGHT, Board_KEY_UP, Board_KEY_DOWN, and Board_KEY_SELECT.
 - **LCDDogm1286.** A CC1310/CC2650 specific LCD driver controls the DOGM1286 LCD controller via the SPI driver.
 - **I²C.** The I²C driver is configured on I2C0.
 - **RF** The RF driver uses the on-chip radio module on CC13xx devices.
 - **Serial.** The UART driver uses UART0, which is attached to the XDS100v3 emulator to facilitate serial communications.
 - **SPI.** The SPI driver uses SPI0 for Board_SPI0 and SPI1 for Board_SPI1.

3.5 CC1310 SimpleLink LaunchPad Settings & Resources

The CC1310 LaunchPad consists of a LaunchPad board with a CC1310EM (CC1310F128 device).



Jumper Settings:

For any examples that use LEDs, make sure the LED's jumper settings are closed.

Switch Settings:

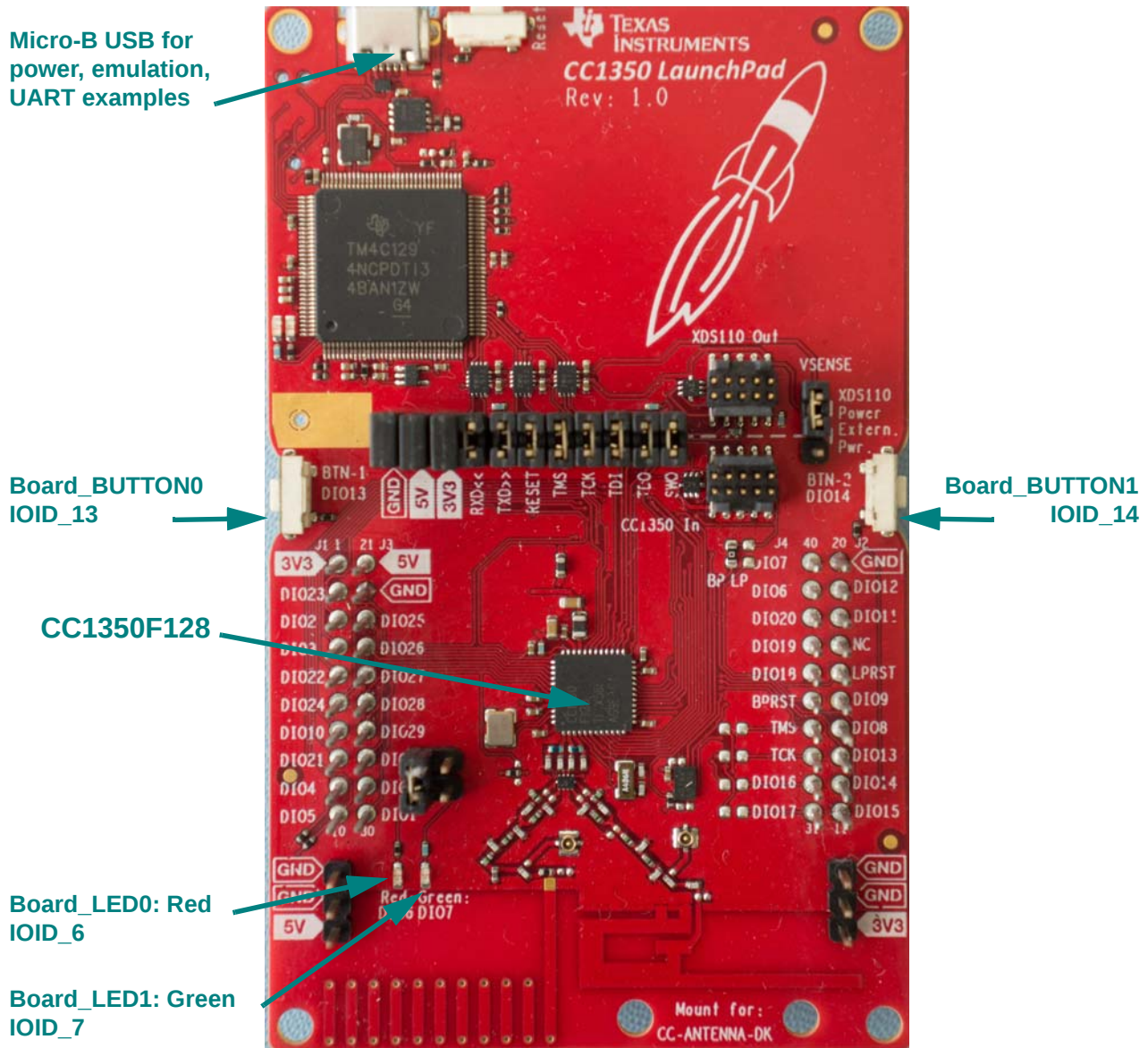
- LEFT: Some examples use IOID_13 as PIN input.
- RIGHT: Some examples use IOID_14 as PIN input.

Resources Used:

- **TI-RTOS Kernel (SYS/BIOS).** Uses the CC26xx's RTC timer and that timer's associated interrupts. The TI-RTOS Kernel manages the Interrupt Vector Table.
- **TI-RTOS.**
 - **Crypto:** A CC13xx/CC26xx specific driver that uses the onboard AES Cryptoprocessor.
 - **PINs.** A CC13xx/CC26xx specific driver PIN driver that uses 2 output pins for the onboard LEDs and 2 input pins for buttons Board_BUTTON0 and Board_BUTTON1.
 - **I²C.** The I²C driver is configured on I2C0.
 - **RF** The RF driver uses the on-chip radio module on CC13xx devices.
 - **Serial.** The UART driver uses UART0, which is attached to the XDS100v3 emulator to facilitate serial communications.
 - **SPI.** The SPI driver uses SPI0 for Board_SPI0.

3.6 CC1350 SimpleLink LaunchPad Settings & Resources

The CC1350 LaunchPad consists of a LaunchPad board with a CC1350EM (CC1350F128 device).



Jumper Settings:

For any examples that use LEDs, make sure the LED's jumper settings are closed.

Switch Settings:

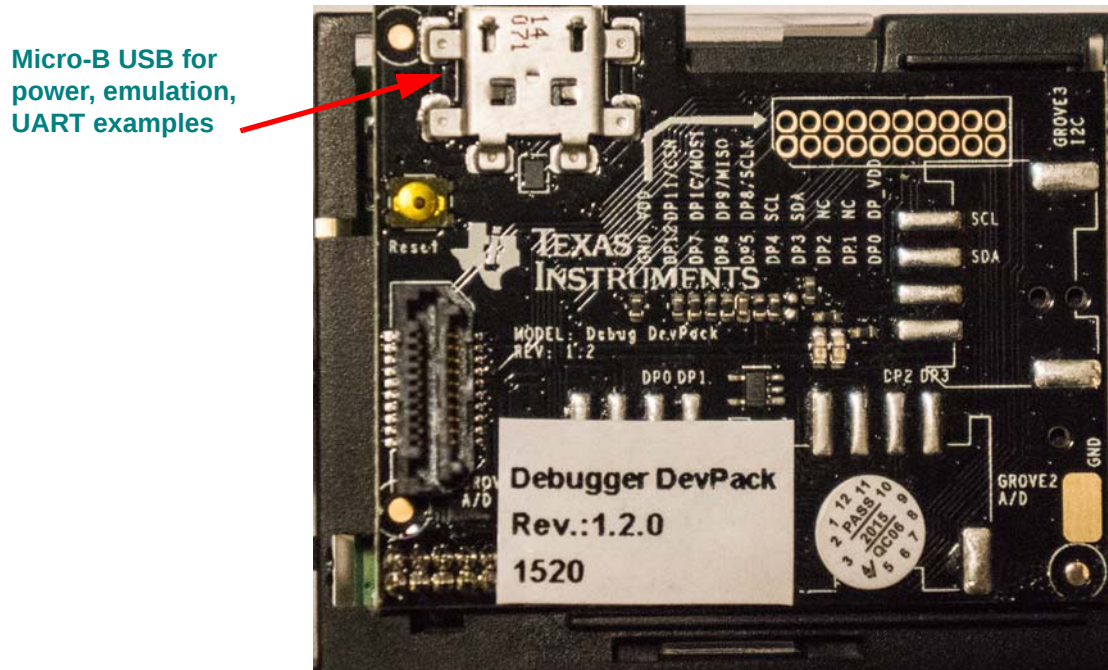
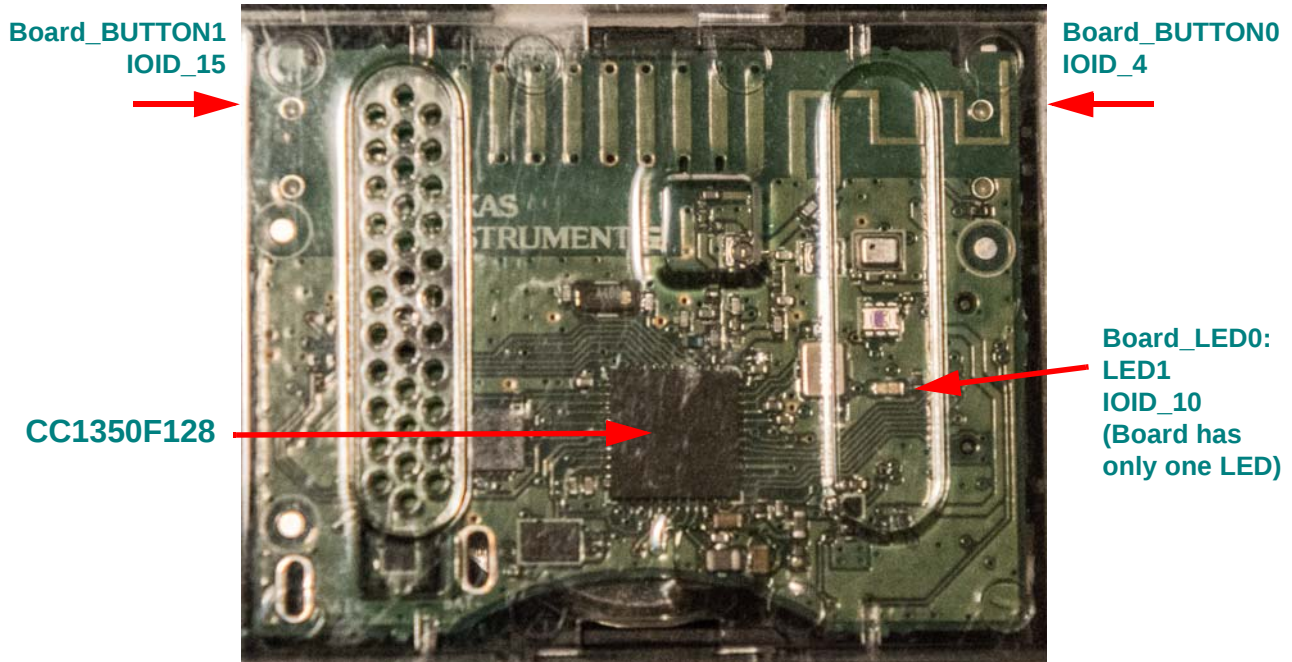
- LEFT: Some examples use IOID_13 as PIN input.
- RIGHT: Some examples use IOID_14 as PIN input.

Resources Used:

- **TI-RTOS Kernel (SYS/BIOS).** Uses the CC26xx's RTC timer and that timer's associated interrupts. The TI-RTOS Kernel manages the Interrupt Vector Table.
- **TI-RTOS.**
 - **Crypto:** A CC13xx/CC26xx specific driver that uses the onboard AES Cryptoprocessor.
 - **PINs.** A CC13xx/CC26xx specific driver PIN driver that uses 2 output pins for the onboard LEDs and 2 input pins for buttons Board_BUTTON0 and Board_BUTTON1.
 - **I²C.** The I²C driver is configured on I2C0.
 - **RF** The RF driver uses the on-chip radio module on CC13xx devices.
 - **Serial.** The UART driver uses UART0, which is attached to the XDS100v3 emulator to facilitate serial communications.
 - **SPI.** The SPI driver uses SPI0 for Board_SPI0.

3.7 CC1350 SimpleLink SensorTag Settings & Resources

The CC1350 SensorTag consists of a SensorTag board with a CC1350EM (CC1350F128 device).



Switch Settings:

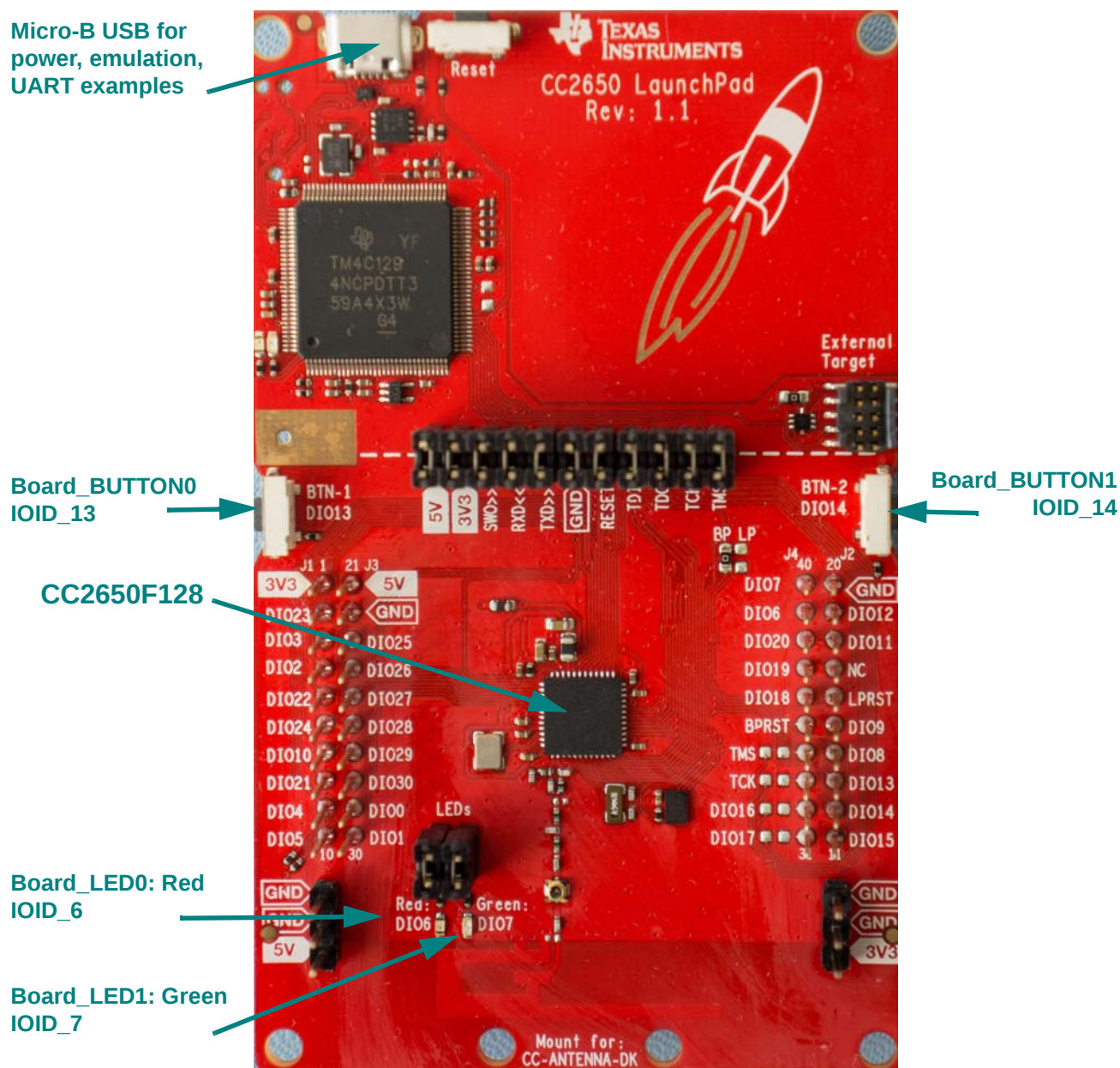
- LEFT: Some examples use IOD_15 as PIN input.
- RIGHT: Some examples use IOD_4 as PIN input.

Resources Used:

- **TI-RTOS Kernel (SYS/BIOS).** Uses the CC26xx's RTC timer and that timer's associated interrupts. The TI-RTOS Kernel manages the Interrupt Vector Table.
- **TI-RTOS.**
 - **Crypto:** A CC13xx/CC26xx specific driver that uses the onboard AES Cryptoprocessor.
 - **PINs.** A CC13xx/CC26xx specific driver PIN driver that uses 1 output pin for the onboard LED and 2 input pins for buttons Board_BUTTON0 and Board_BUTTON1.
 - **I²C.** The I²C driver is configured on I2C0.
 - **Serial.** The UART driver uses UART0, which is attached via the 10-pin ARM cortex JTAG header. Emulation and UART communication are possible when the debugger is configured for Serial Wire Debugging.
 - **SPI.** The SPI driver uses SPI0 for Board_SPI0.

3.8 CC2650 SimpleLink LaunchPad Settings & Resources

The CC2650 LaunchPad consists of a LaunchPad board with a CC2650EM (CC2650F128 device).



Jumper Settings:

For any examples that use LEDs, make sure the LED's jumper settings are closed.

Switch Settings:

- LEFT: Some examples use `IOID_13` as PIN input.
- RIGHT: Some examples use `IOID_14` as PIN input.

Resources Used:

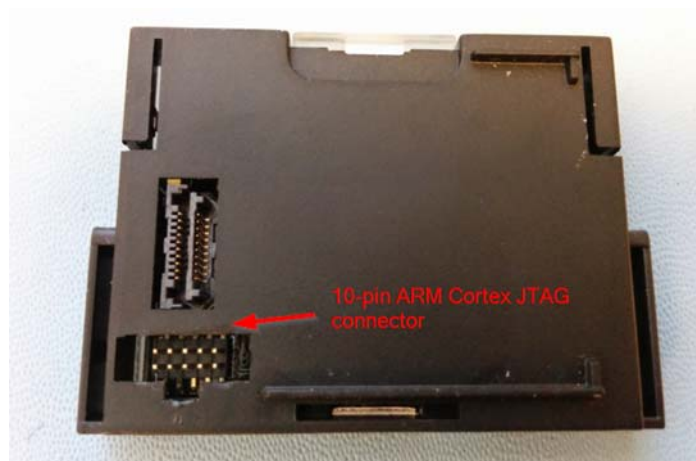
- **TI-RTOS Kernel (SYS/BIOS).** Uses the CC26xx's RTC timer and that timer's associated interrupts. The TI-RTOS Kernel manages the Interrupt Vector Table.
- **TI-RTOS.**
 - **Crypto:** A CC13xx/CC26xx specific driver that uses the onboard AES Cryptoprocessor.
 - **PINs.** A CC13xx/CC26xx specific driver PIN driver that uses 2 output pins for the onboard LEDs and 2 input pins for buttons Board_BUTTON0 and Board_BUTTON1.
 - **I²C.** The I²C driver is configured on I2C0.
 - **RF** The RF driver uses the on-chip radio module on CC13xx devices.
 - **Serial.** The UART driver uses UART0, which is attached to the XDS100v3 emulator to facilitate serial communications.
 - **SPI.** The SPI driver uses SPI0 for Board_SPI0.

3.9 CC2650 SimpleLink SensorTag Settings and Resources

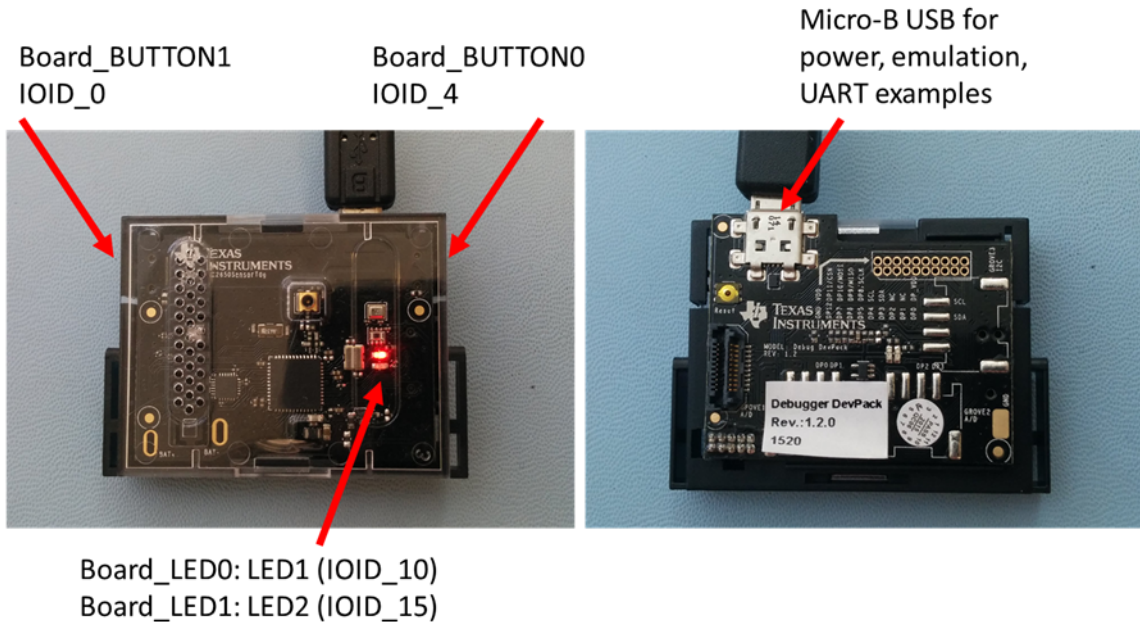
The CC2650STK-BLE (SensorTag) consists of a CC2650F128 device.

The CC2650 SensorTag connects to an external low cost DevPack debugger. The DevPack itself hosts an on-board XDS110 debugger, which also supports UART communications. See the [CC13xx/CC26xx Tools Overview page](#) regarding the XDS110.

The 10-pin ARM Cortex JTAG connector is located on the back of the CC2650 SensorTag. Note that by default the JTAG connector is concealed by the plastic housing.



The following figure shows the external low cost DevPack debugger used with the CC2650 SensorTag.



Switch Settings:

- LEFT: Some examples use IOID_0 as PIN input.
- RIGHT: Some examples use IOID_4 as PIN input.

These switches are located on the sides of the SensorTag. Push the plastic housing so that it makes contact with the push button.

Resources Used:

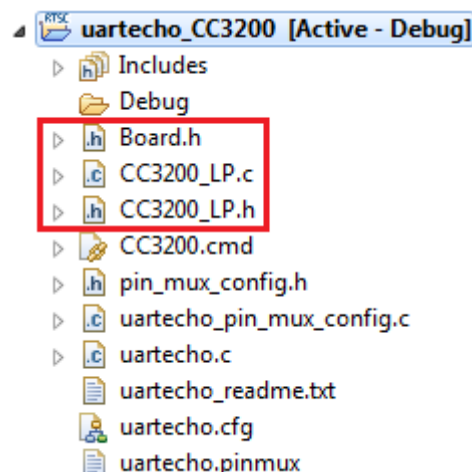
- **TI-RTOS Kernel (SYS/BIOS).** Uses the CC26xx's RTC timer and that timer's associated interrupts. The TI-RTOS Kernel manages the Interrupt Vector Table.
- **TI-RTOS.**
 - **Crypto.** A CC2650 specific driver that uses the onboard AES Cryptoprocessor.
 - **PINs.** A CC2650 specific driver that uses 2 output pins for the onboard LEDs and 2 input pins for switches Board_KEY_LEFT and Board_KEY_RIGHT.
 - **I²C.** The I²C driver is configured on I2C0.
 - **Serial.** The UART driver uses UART0, which is attached via the 10-pin ARM cortex JTAG header. Emulation and UART communication are possible when the debugger is configured for Serial Wire Debugging.
 - **SPI.** The SPI driver uses SPI0 for Board_SPI0.

3.10 <Board>.c File

For every board supported in each TI-RTOS product, there are three common files that are used in all the driver examples.

- **Board.h.** This is a small shim file that allows example code (for example, uartecho.c) to be used on different boards in different TI-RTOS products.
- **<Board>.c/.h files.** For example, CC3200_LP.c and CC3200_LP.h in this picture. These files are specific to a board. The same files are used for all driver examples for that board.

For CC13xx and CC26xx examples, a table of PIN_Config structures is defined in the example's .c files. No pin setup is done in the board-specific files (<board>.c/.h)



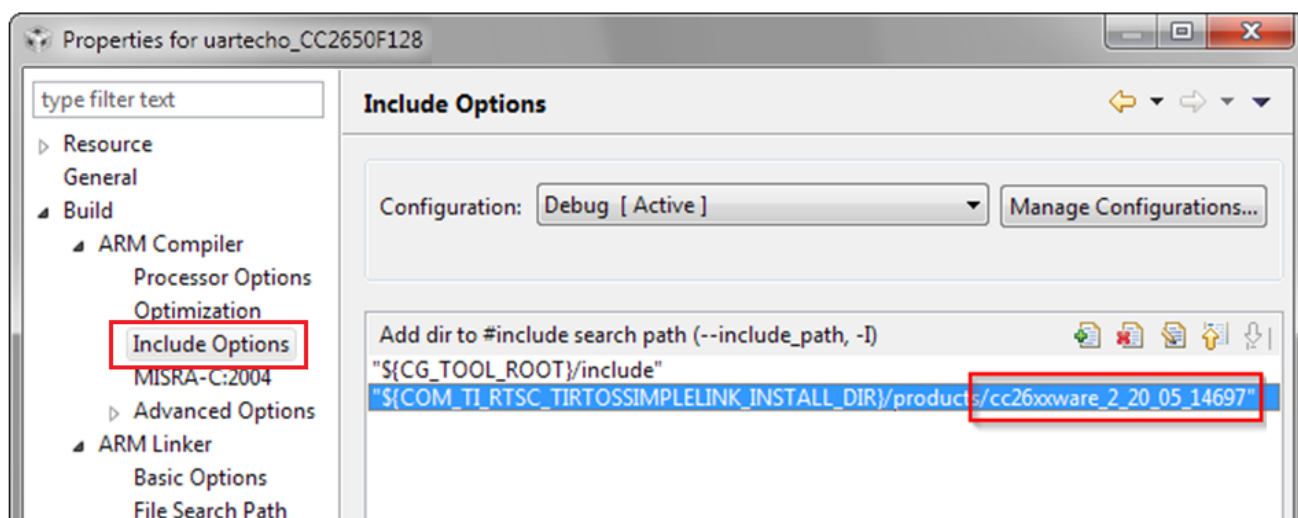
3.11 Updating driverlib for CC13xx/CC26xx

To build TI-RTOS examples against different versions of CC13xxWare and CC26xxWare, follow the instructions in Section 3.11.1 for building with CCS, Section 3.11.2 for IAR, and Section 3.11.3 for makefiles.

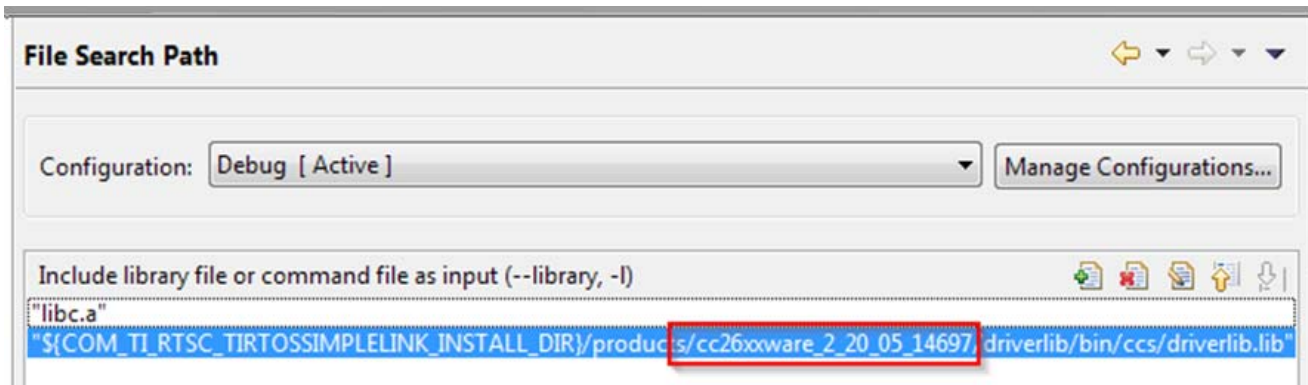
The following sections show CC26xx as an example, but the instructions can also be followed for CC13xx.

3.11.1 Building with CCS

1. Open an existing TI-RTOS project in CCS.
2. Choose **Project > Properties** from the CCS menus.
3. In the Properties dialog, select the **Build > ARM Compiler > Include Options** category.
4. Edit the search path to include the location of the desired version of CC26xxWare. For example:



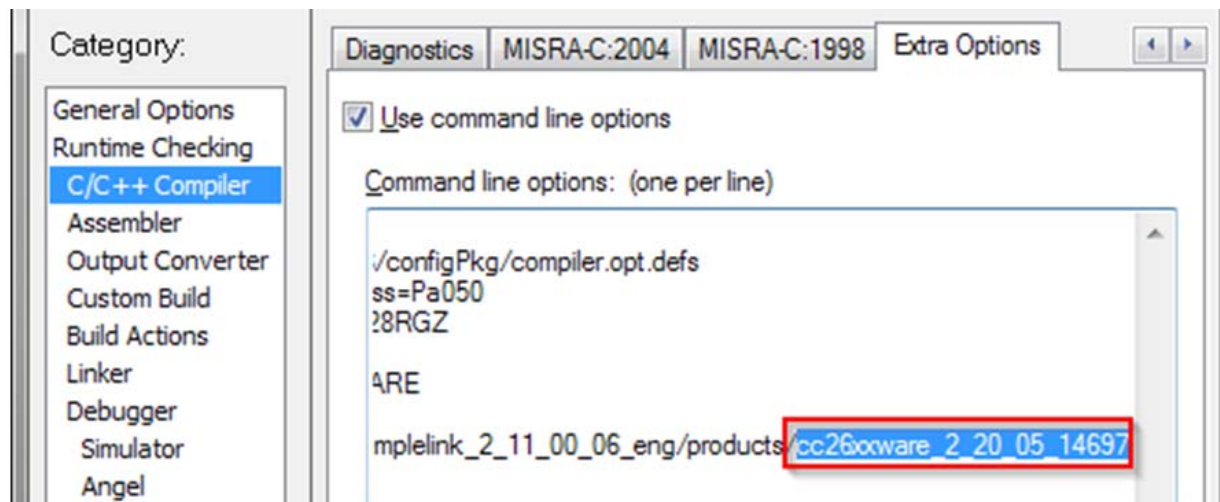
5. In the Properties dialog, select the **Build > ARM Linker > File Search Path** category.
6. Edit the included library path to point to the location of the desired CC26xxWare driverlib.lib library file. For example:



7. Click **OK** to save the project settings. Rebuild the project to use the new settings.

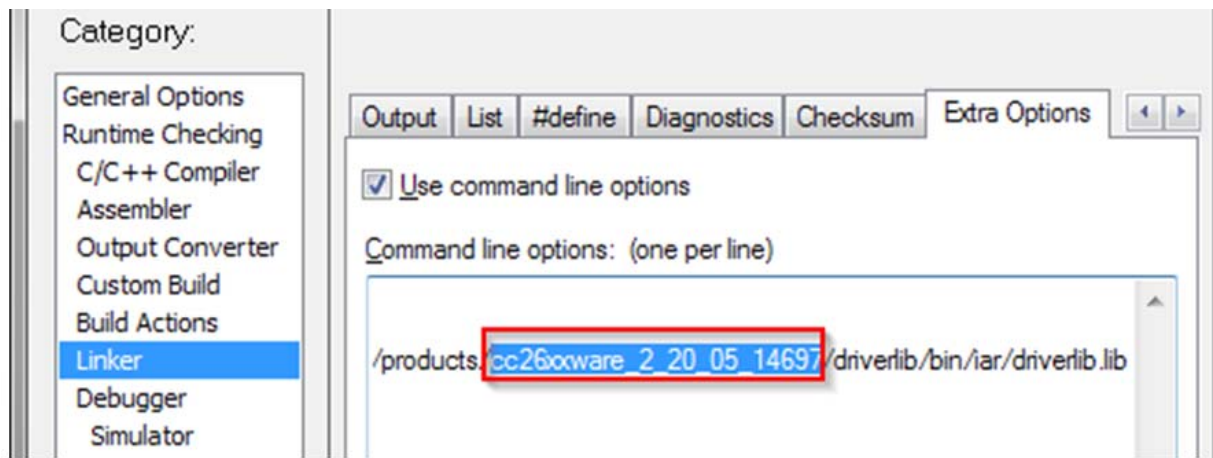
3.11.2 Building with IAR

1. Open an existing TI-RTOS project in IAR Embedded Workbench.
2. Choose **Project > Options** from the IAR menus.
3. In the Options dialog, select the **C/C++ Compiler** category and then the **Extra Options** tab.
4. Edit the search path to include (-I) the location of the desired version of CC26xxWare.



5. In the Options dialog, select the **Linker** category and then the **Extra Options** tab.

6. Edit the included library path to point to the location of the desired CC26xxWare driverlib.lib library file.



7. Click **OK** to save the project settings. Rebuild the project to use the new settings.

3.11.3 Building with a Makefile

1. Open the `tirtos.mak` file in the TI-RTOS installation directory.
2. Edit the definition of `CC26XXWARE_INSTALL_DIR` to point to the desired version of CC26xxWare and the definition of `CC13XXWARE_INSTALL_DIR` to point to the desired version of CC13xxWare.

```
#
# TI-RTOS and XDCTools settings
#
XDCTOOLS_INSTALL_DIR      ?= $(DEFAULT_INSTALL_DIR)/xdctools_3_31_01_33_core
export XDCTOOLS_JAVA_HOME ?= $(DEFAULT_INSTALL_DIR)/ccsv6/eclipse/jre

TIRTOS_INSTALLATION_VER   ?= tirtos_simplelink_2_20_00_04_eng
TIRTOS_INSTALL_DIR        ?= $(DEFAULT_INSTALL_DIR)/$(TIRTOS_INSTALLATION_VER)
BIOS_INSTALL_DIR          ?= $(TIRTOS_INSTALL_DIR)/products/bios_6_42_01_15_eng
UIA_INSTALL_DIR           ?= $(TIRTOS_INSTALL_DIR)/products/uia_2_00_02_39
CCWARE_INSTALL_DIR        ?= $(TIRTOS_INSTALL_DIR)/products/CC3200_driverlib_1.1.0
CC26XXWARE_INSTALL_DIR    ?= $(TIRTOS_INSTALL_DIR)/products/cc26xxware_2_21_02_15830
CC13XXWARE_INSTALL_DIR    ?= $(TIRTOS_INSTALL_DIR)/products/cc13xxware_2_00_02_15830
```

3. Generate the new Make files using the following commands.

```
<xdctools_root_dir>/gmake -f tirtos.mak examplesgen DEST="c:\tirtos_examples"
<tirtos_install_dir>/gmake -f tirtos.mak examplesgen DEST="c:\tirtos_examples"
```

4. You should see messages like the following:

```
generating examples in c:\tirtos_examples ...
*****
Please refer to "Examples for TI-RTOS" section in the TI-RTOS
"Getting Started Guide" for details on how to build and load the examples
into IAR WorkBench and CCS.
```

5. Move to the directory that contains the generated examples and build them as follows:

```
cd c:\tirtos_examples\examples\TI\CC2650DK  
<xdctools_root_dir>/gmake
```

Configuring TI-RTOS

This chapter describes how to configure TI-RTOS and its components for use by your application.

Topic		Page
4.1	Starting the Configuration Tool	35
4.2	Configuring TI-RTOS Drivers	36
4.3	Configuring Components of TI-RTOS	37

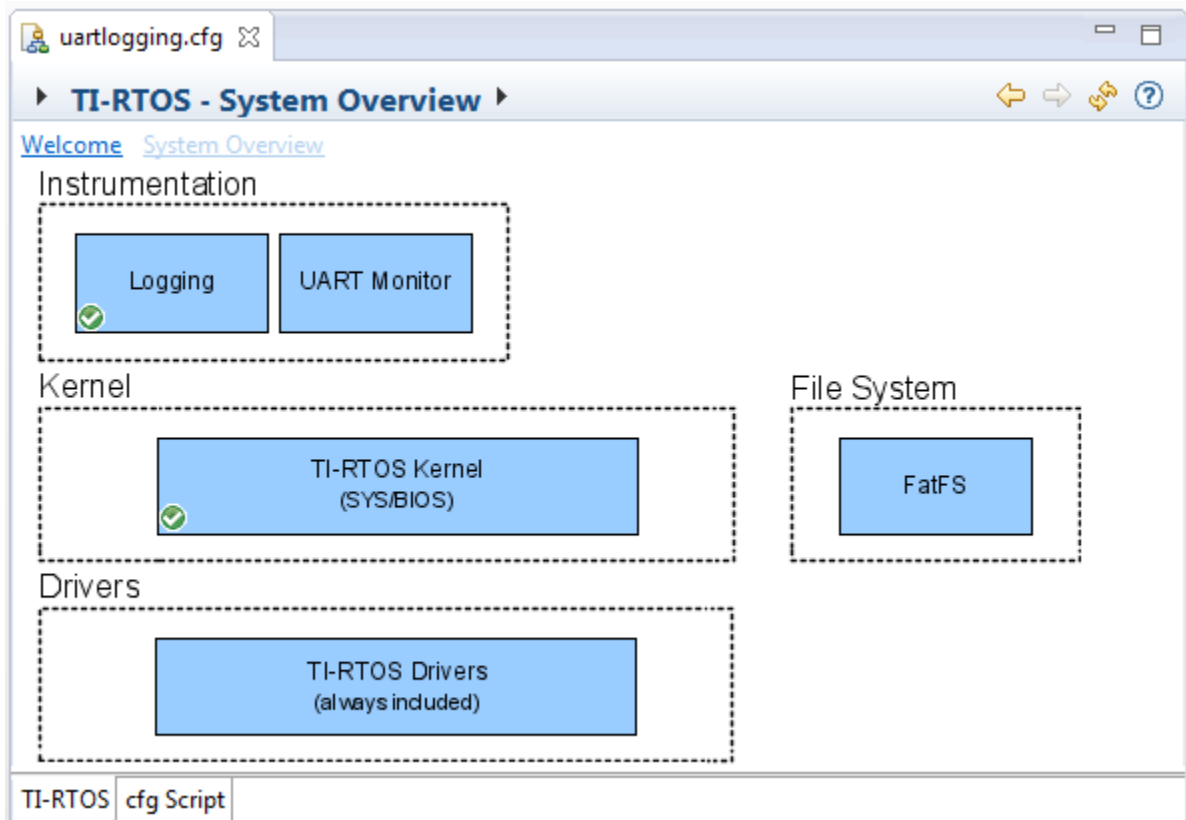
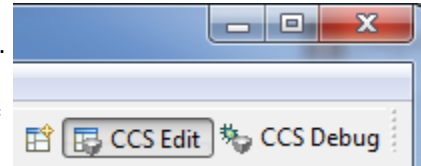
4.1 Starting the Configuration Tool

Note: The graphical configuration tool is not available within IAR Embedded Workbench. If you are using IAR, edit the project's *.cfg file within IAR as a text-based source file. See the [Texas Instruments Wiki](#) for more about using IAR with TI-RTOS.

This section shows how to open the Graphical Configuration Tool (XGCONF) to view the System Overview. For details on using XGCONF, see Chapter 2 of the [SYS/BIOS User's Guide \(SPRUEX3\)](#).

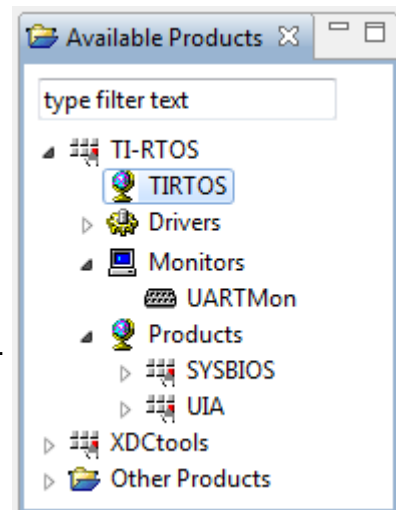
To use CCS to open the graphical tool for editing configuration files (XGCONF), follow these steps:

1. Make sure you are in the **CCS Edit** perspective of CCS. If you are not in that perspective, click the CCS Edit icon to switch back.
2. Double-click on the *.cfg configuration file for a TI-RTOS example project in the **Project Explorer** tree. (See [Section 3.1](#) if you need to create an example project.) While XGCONF is opening, the CCS status bar shows that the configuration is being processed and validated.
3. When XGCONF opens, you see the **Welcome** sheet for TI-RTOS if you are using a Driver example. (If this is the configuration file for a Kernel example or an Instrumentation example, the Welcome sheet for SYS/BIOS opens first, instead.) The Welcome sheet provides links to documentation resources.
4. Click the **System Overview** link to see a diagram of the components available through TI-RTOS. (SYS/BIOS modules are shown if you are using a Kernel example or an Instrumentation example.) A green check mark indicates the modules that are being used by the application.



5. You also see a list of Available Products in a pane on the left of the CCS window. This list allows you to select the TI-RTOS module and any configurable modules in the products TI-RTOS provides.
6. Click a blue box in the System Overview to go to the configuration page for a module.

Note: If the configuration is shown in a text editor instead of XGCONF, close the text editor window. Then, right-click on the *.cfg file and choose **Open With > XGCONF**. If you are comfortable editing configuration scripts with a text editor, you can do that. However, you should not have the file open in both types of editor at the same time.



4.2 Configuring TI-RTOS Drivers

In the System Overview display for the TIRTOS module, click on the **TI-RTOS Drivers** block.



You can choose to use either the instrumented or non-instrumented driver libraries when linking with TI-RTOS. The instrumented libraries process Log events while the non-instrumented libraries do not. See the section on "Using Instrumented or Non-Instrumented Libraries" in the [TI-RTOS User's Guide](#) (SPRUHD4) for more information. This setting affects all the TI-RTOS drivers listed in [Section 1.6](#) together.

All of the TI-RTOS drivers are available to your application without being separately enabled. To reduce code size, only the driver code that your application needs to use will be compiled into your application.

4.2.1 Configuring System Support

The SysCallback module lets you configure the functions that handle System output—for example, `System_printf()` and `System_abort()`. This module handles transmissions to System output only; it does not handle responses received. See the chapter on "TI-RTOS Utilities" in the [TI-RTOS User's Guide](#) (SPRUHD4) for more about the SysCallback module.

Other SystemSupport implementations are provided with XDCtools.

- **SysMin** stores System_printf() strings in an internal buffer in RAM. SysMin requires RAM, so it not ideal for devices with minimal RAM.
- **SysStd** writes System_printf() strings to STDOUT (the CCS Console window). By default, SysStd allows System_printf() to be called from Tasks only (not Swis or hardware interrupts); it can be modified to allow calls from Swis and Hwis, but this impacts real-time performance.

4.2.2 Configuring ROM for CC13xx/CC2650

Many of the common SYS/BIOS functions are provided in the ROM area of the CC13xx/CC26xx. About 12 KB of the ROM is reserved for these modules. Applications can be built to use this code in ROM, which frees up Flash memory for use by the application. The ROM version of driver library functions execute faster and reduce power consumption; Flash is both slower and more power consuming than the ROM.

To use SYS/BIOS modules in the ROM of the CC2650, add these lines to your application's *.cfg file:

```
/* use the kernel in the ROM */  
var ROM = xdc.useModule('ti.sysbios.rom.ROM');  
ROM.romName = ROM.CC2650;
```

To use SYS/BIOS modules in the ROM of the CC13xx, add these lines to your application's *.cfg file:

```
/* use the kernel in ROM */  
var ROM = xdc.useModule('ti.sysbios.rom.ROM');  
ROM.romName = ROM.CC1350;
```

4.3 Configuring Components of TI-RTOS

For information about configuring individual sub-components of TI-RTOS, see the documentation for that component. Chapter 2 of the [SYS/BIOS User's Guide \(SPRUEX3\)](#) provides details about XGCONF. Within XGCONF, you can see the full file path to the version of the component being used by hovering your mouse cursor over a component in the "Other Products" list in the **Available Products** area.

Index

A

App Center 11
Available Products list 36

C

CC1310 LaunchPad 7
CC1350 LaunchPad 7
CC1350 SensorTag 7
CC13xxWare 5
CC2650 LaunchPad 7
CC2650 Sensor Tag 7
CC2650 SimpleLink Development Kit 7
CC26xxWare 5
CCS
 creating a project 6, 14
 installation 11
 other documentation 8
CCS App Center 4, 11
components 5
Concerto
 other documentation 9
configuration 34
 graphical editor 35
Crypto driver 7

D

disk space 10
documentation 8

E

Empty Project example 16
empty.c file 16
examples 13

F

FatFs API, documentation 9
forum 8

G

GPIO driver 7

I

I2C driver 7
installation
 CCS 11
 directory 11
instrumentation 5
instrumented libraries 36

K

kernel examples 6

L

LCD driver 7
LEDs
 managed by GPIO driver 7
LoggingSetup module 18

N

non-instrumented libraries 36

P

PIN driver 7
Power driver 7
products directory 5
PWM driver 7

R

readme.txt file 17
Resource Explorer 6, 14
RF driver 8

S

SPI driver 8
SYS/BIOS 5
 other documentation 8
SysCallback module, configuration 36
SysMin module 18
System Overview configuration 35
system requirements 10

U

UART driver 8
UIA 5

other documentation 9

W

wiki 8

X

XDCtools 5
 other documentation 8
XGCONF
 configuring other components 37
 starting 35

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI's terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers' products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers' products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information 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.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI's goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have not been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products

Audio	www.ti.com/audio
Amplifiers	amplifier.ti.com
Data Converters	dataconverter.ti.com
DLP® Products	www.dlp.com
DSP	dsp.ti.com
Clocks and Timers	www.ti.com/clocks
Interface	interface.ti.com
Logic	logic.ti.com
Power Mgmt	power.ti.com
Microcontrollers	microcontroller.ti.com
RFID	www.ti-rfid.com
OMAP Mobile Processors	www.ti.com/omap
Wireless Connectivity	www.ti.com/wirelessconnectivity

Applications

Automotive and Transportation	www.ti.com/automotive
Communications and Telecom	www.ti.com/communications
Computers and Peripherals	www.ti.com/computers
Consumer Electronics	www.ti.com/consumer-apps
Energy and Lighting	www.ti.com/energy
Industrial	www.ti.com/industrial
Medical	www.ti.com/medical
Security	www.ti.com/security
Space, Avionics and Defense	www.ti.com/space-avionics-defense
Video & Imaging	www.ti.com/video

TI E2E Community

e2e.ti.com