TI-RTOS 2.11 for SimpleLink™ Wireless MCUs

Getting Started Guide

Literature Number: SPRUHU8B
January 2015

Contents

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 v2.11.

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, 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.

January 6, 2015

Chapter 1

SPRUHU8B—January 2015

About TI-RTOS

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

Topic **Page**

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 differentiating your application.

TI-RTOS is *not* installed automatically as part of the Code Composer Studio v6.0 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 2.4). In addition to the Texas Instruments Code Generation Tools, TI-RTOS includes support for the IAR and GNU tool chains (see Section 2.5).

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	PDF Documentation Location
TI-RTOS	TI-RTOS examples	Chapter 3 of this Getting Started Guide
TI-RTOS Kernel	SYS/BIOS	*SYS/BIOS (TI-RTOS Kernel) User's Guide* -- SPRUEX3
TI-RTOS Instrumentation	UIA	*System Analyzer User's Guide* -- SPRUH43
TI-RTOS File System	FatFS	*TI-RTOS User's Guide* -- SPRUHD4
TI-RTOS Drivers and Board Initialization	Drivers and CC26xxWare and CC3200 SDK's driverlib	*TI-RTOS User's Guide* -- SPRUHD4

The components in the "`products`" subdirectory are:

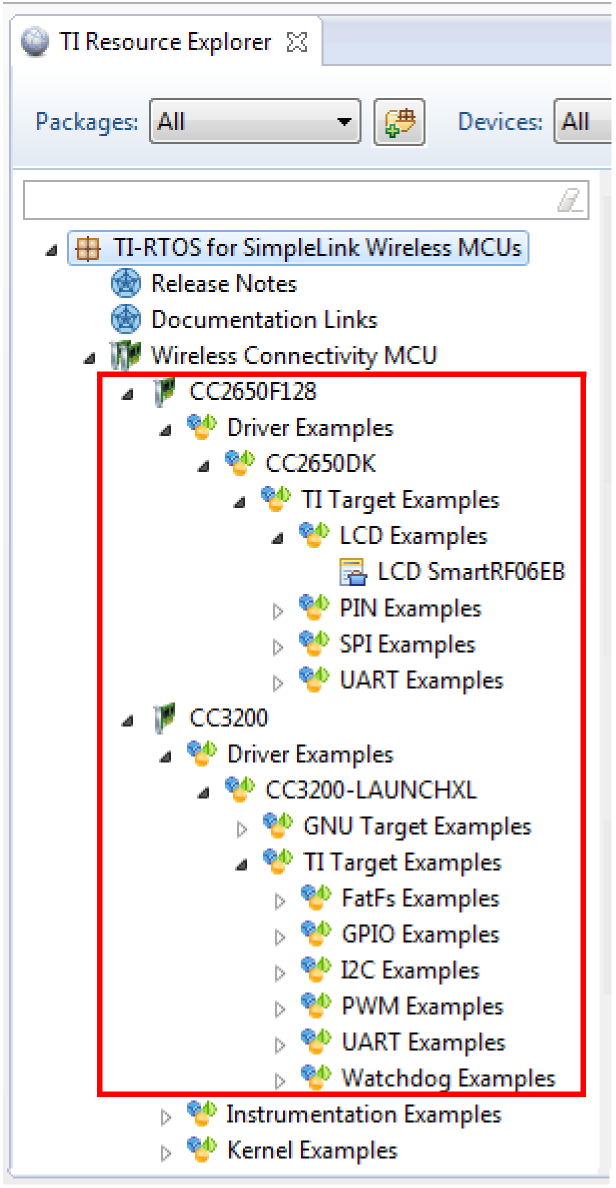
- **TI-RTOS Kernel — SYS/BIOS.** SYS/BIOS is a scalable real-time kernel. It is designed to be used by applications that require real-time scheduling and synchronization or real-time instrumentation. It provides preemptive multi-threading, hardware abstraction, real-time analysis, and configuration tools. SYS/BIOS is designed to minimize memory and CPU requirements on the target.
- **TI-RTOS Instrumentation — UIA.** The Unified Instrumentation Architecture (UIA) provides target content that aids in the creation and gathering of instrumentation data (for example, Log data).
- **CC26xxWare.** This software component contains low-level drivers for use with CC26xx targets.
- **SimpleLink CC3200 Driverlib** is a subset of the SimpleLink Wi-Fi CC3200 Software Development Kit (SDK) needed to use the drivers with CC3200. The Wi-Fi functionality is not included in this driverlib subset; to use Wi-Fi, download and install the **SimpleLink Wi-Fi CC3200 Software Development Kit (SDK)**. You can download this SDK at http://www.ti.com/tool/cc3200sdk. For details about using the Wi-Fi examples in the SDK, see the TI-RTOS CC3200Wireless wiki page.
- **XDCtools.** This core component provides the underlying tooling for configuring and building TI-RTOS and its components. XDCtools is installed as part of CCSv6. 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 **TI 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 TI Resource Explorer to see the examples that are available for your device. TI Resource Explorer provides TI-RTOS examples for both the TI and GNU tool chains.

- **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 v5.1.5
- **GCC:** gcc-arm-none-eabi-4_7-2013q3 (CC3xxx only)
- **IAR:** 7.30.3

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

- CC2xxx
  - ti.targets.arm.elf.M3
  - iar.targets.arm.M3
  - gnu.targets.arm.M3
- CC3xxx
  - ti.targets.arm.elf.M4
  - iar.targets.arm.M4
  - gnu.targets.arm.M4

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

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

Family	Device on Board	Board
ARM	CC3200	CC3200 LaunchPad
ARM	CC2650	CC2650 SimpleLink Development Kit
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, including a topic on Creating TI-RTOS Projects for Other MSP430 Devices.

1.6 What Drivers Does TI-RTOS Include?

TI-RTOS includes drivers for the following peripherals. These drivers are in the `<install_dir>/packages/ti/drivers` directory. TI-RTOS examples show how to use these drivers. Note that all of these drivers are built on top of CC26xxWare or the driverlib from the CC3200 SDK.

- **Crypto.** Advanced Encryption Standard (AES) driver for data encryption and decryption. (CC26xx devices only)

- **I^{2}C.** API set intended to be used directly by the application or middleware.
- **GPIO.** API set intended to be used directly by the application or middleware to manage the GPIO interrupts, pins, and ports.
- **LCD.** Driver for CC26xx LCD display.
- **PIN.** Driver for CC26xx Pin interrupts.
- **PWM.** API set intended to be used directly by the application or middleware to generate Pulse Width Modulated signals. (CC32xx devices only.)
- **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).
- **SDSPI.** Driver for SD cards using a SPI (SSI) bus. This driver is used by the FatFS and not intended to be called directly by the application.
- **UART.** API set intended to be used directly by the application to communicate with the UART.
- **Watchdog.** API set intended to be used directly by the application or middleware to manage the watchdog timer.

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)
  - 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 TI Resource Explorer (in CCS), choose the **Documentation Links** item for your version of TI-RTOS. In the Documentation Links 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 TI Resource Explorer (in CCS), choose the **Documentation Links** item for your version of TI-RTOS. In the Documentation Links page, choose the **TI-RTOS Instrumentation Runtime APIs and Configuration (cdoc)** item.
  - System Analyzer on the Texas Instruments Wiki
- **FatFS API**
  - Open source documentation
  - FatFS for SYS/BIOS wiki page
  - SYS/BIOS API and configuration reference. In TI Resource Explorer (in CCS), choose the **Documentation Links** item for your version of TI-RTOS. In the Documentation Links page, choose the **TI-RTOS Kernel Runtime APIs and Configuration (cdoc)** item and see help under the ti.sysbios.fatfs.FatFS module topic.
- **General microcontroller information**
  - Microcontrollers forum on TI's E2E Community
- **SimpleLink resources**
  - SimpleLink WiFi Overview
  - CC3200 Launchpad
  - CC3100 Boosterpack
  - SimpleLink WiFi Radio Tool
  - CC3200 WiFi SDK
  - CC3100 WiFi SDK
  - CC3200 Programmer's Guide (SWRU369)
  - CC3200 Technical Reference Manual (SWRU367)
  - SimpleLink WiFi Wiki
  - CC32xx & CC31xx E2E Forum
- **I^{2}C**
  - Specification

Chapter 2

SPRUHU8B—January 2015

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

The Microsoft Windows version of TI-RTOS can be installed on systems running 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 2.11 is used in conjunction with Code Composer Studio 6.0 or higher. (TI-RTOS can also be used with the IAR Embedded Workbench IDE. See page 13 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.

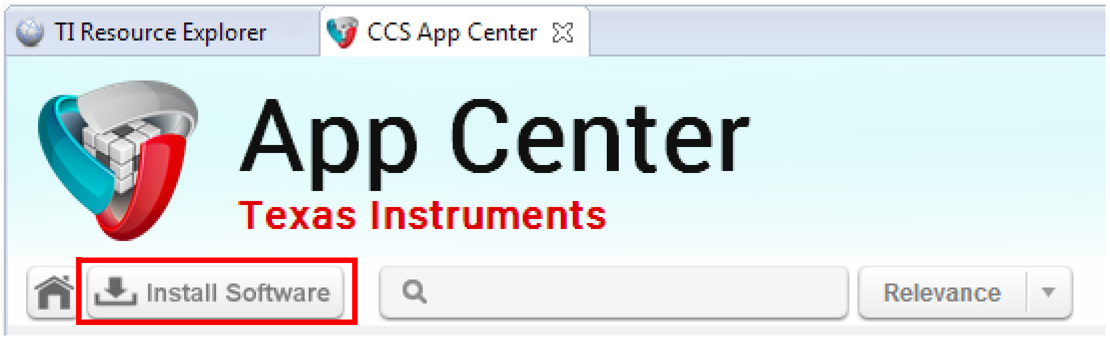
To install CCS 6.0, go to the "Download CCS" page on the Texas Instruments wiki and follow a link to download the software for your license type. For multi-user licenses, see the CCS product page.

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 v6.0 installation. Instead, you install it through the CCS App Center. Follow these steps to install TI-RTOS:

1. Run CCS v6.0 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 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 and GNU tool chains (GNU is supported for CC3xxx only).

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 recommended.

   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_30_##_##_core`.

Follow the instructions in Section 3.3 to complete the installation of the TI-RTOS examples.

2.5 Installing TI-RTOS for Use in IAR Embedded Workbench

If you do not use Code Composer Studio, you can install TI-RTOS for use with IAR Embedded Workbench. In addition to compiling and linking with the Texas Instruments Code Generation Tools, TI-RTOS includes support for the IAR and GNU tool chains (GNU is supported for CC3xxx only).

1. Install IAR Embedded Workbench for Texas Instruments ARM devices.
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 recommended.

   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_30_##_##_core`.

Follow the instructions in Section 3.3 to complete the installation of the TI-RTOS examples. See Section 3.4 to build the examples with the IAR compiler and to load and run the examples with IAR Embedded Workbench.

Chapter 3

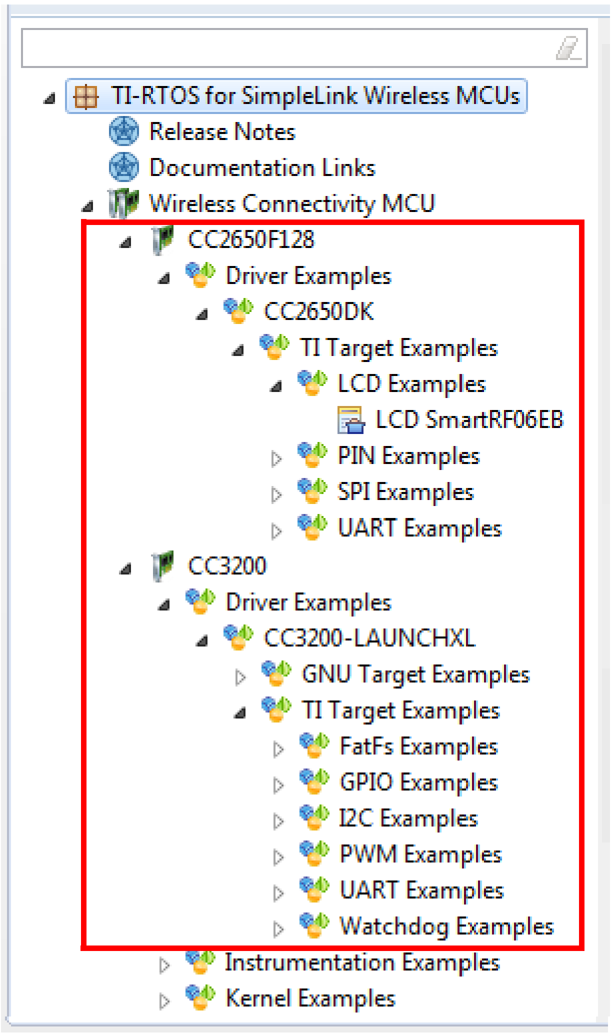
SPRUHU8B—January 2015

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.

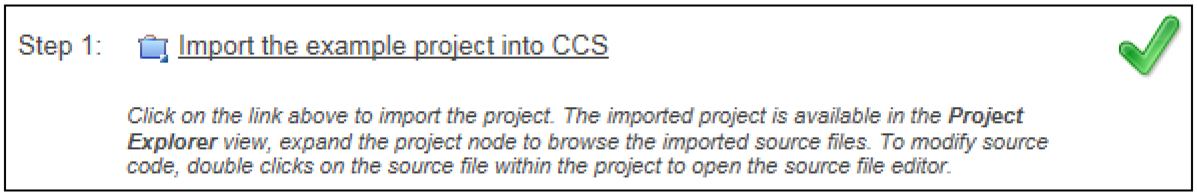
Topic **Page**

3.1 Creating Example Projects Using the TI Resource Explorer



You can use the **TI 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:

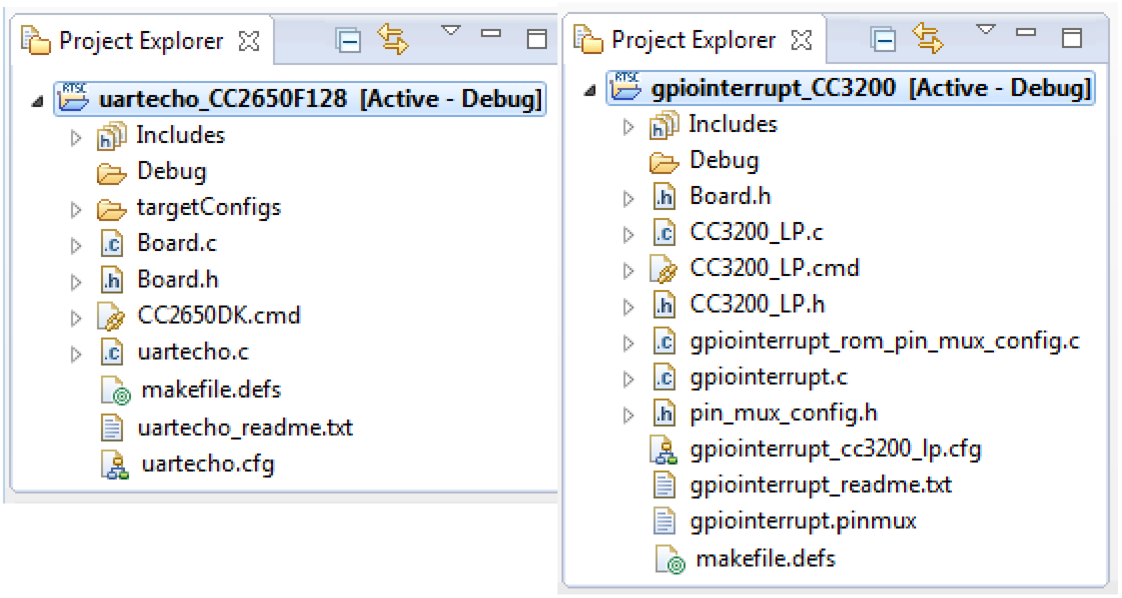
1. Open CCS. If you do not see the TI Resource Explorer, make sure you are in the **CCS Edit** perspective and choose **View > Resource Explorer (Examples)** from the menus.
2. Type the name or part of the name of your device in the **enter search keyword** field to hide all the examples that don't apply to your device. Or, you can type "Driver Examples" to find TI-RTOS 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. TI Resource Explorer provides TI-RTOS examples for both the TI and GNU (CC32xx only) tool chains.
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 TI 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

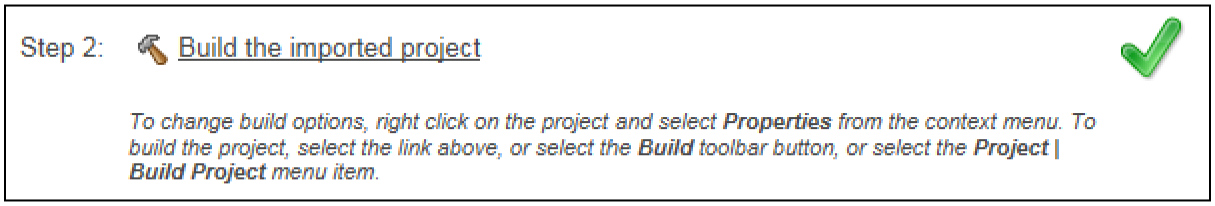
*Click on the link above to import the project. The imported project is available in the **Project Explorer** view, expand the project node to browse the imported source files. To modify source code, double clicks on the source file within the project to open the source file editor.*

Note: CCS will not allow two projects with the same name in a workspace. If you want to import both TI and GNU (CC32xx only) projects for the same board and example in the same workspace, you would need to rename the first project you import before importing the second version.

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.

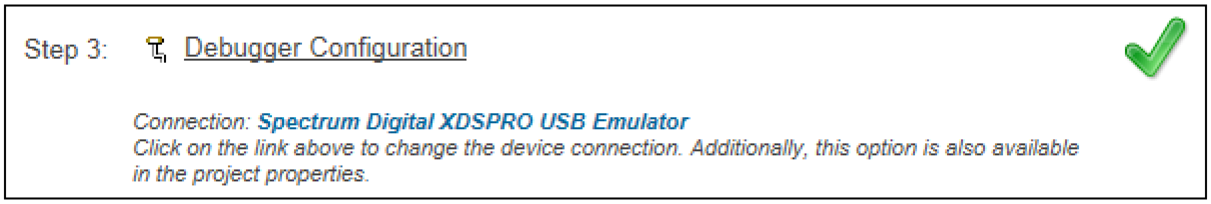


7. The page shown when you select an example in the TI Resource Explorer provides additional links to perform common actions with that example.
8. Use the **Step 2** link when you are ready to build the project. If you want to change any build options, right click on the project and select **Properties**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


Step 2: Build the imported project

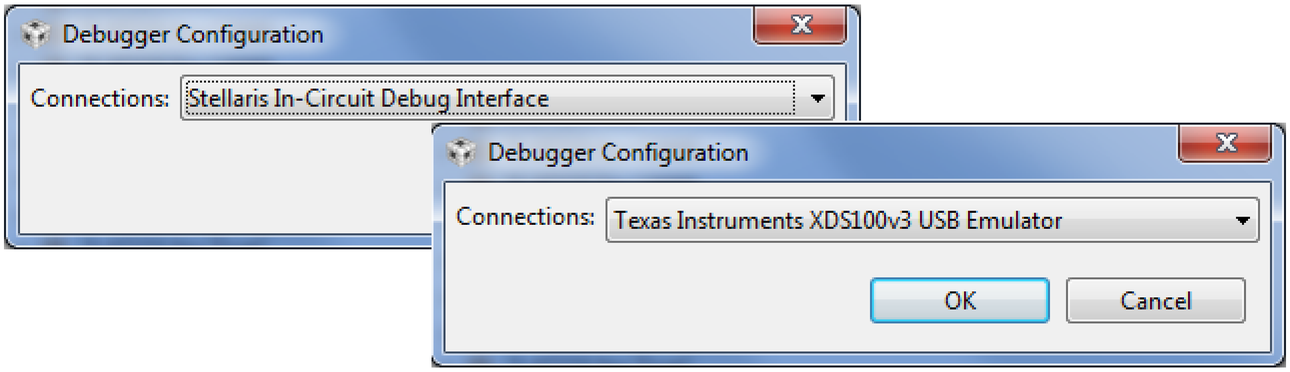
*To change build options, right click on the project and select **Properties** from the context menu. To build the project, select the link above, or select the **Build** toolbar button, or select the **Project | Build Project** menu item.*

9. Use the **Step 3** link to change the connection used to communicate with the board. The current setting is shown in the TI Resource Explorer page for the selected example. (If you want to use a simulator instead of a hardware connection, double-click the *.ccxml file in the targetConfigs folder of the project to open the Target Configuration File editor. Change the **Connection** as needed, and click **Save**.)

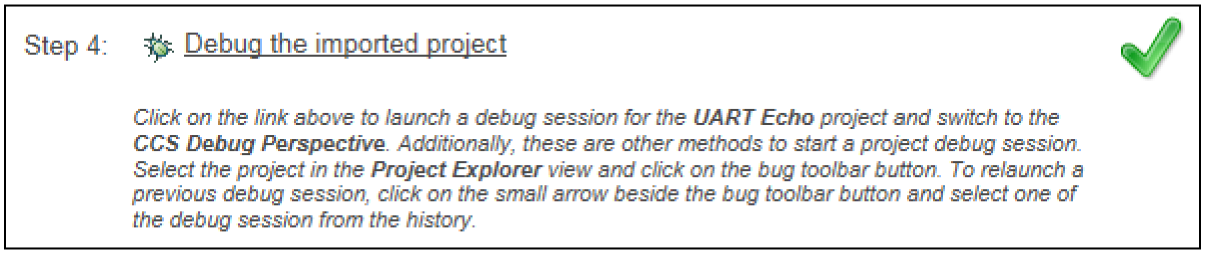

Step 3: Debugger Configuration

*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.

10. You will see the Debugger Configuration dialog. Choose an emulator from the list. For CC26xx SimpleLink devices, choose the XDC100v3. For CC32xx SimpleLink devices, choose the **Stellaris In-Circuit Debug Interface**.


Debugger Configuration
Connections: Stellaris In-Circuit Debug Interface
Debugger Configuration
Connections: Texas Instruments XDS100v3 USB Emulator
OK
Cancel

11. 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.*

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 TI Resource Explorer to create a project. You can open the <*example_name*>_readme file within CCS.

The Driver Examples share the following features:

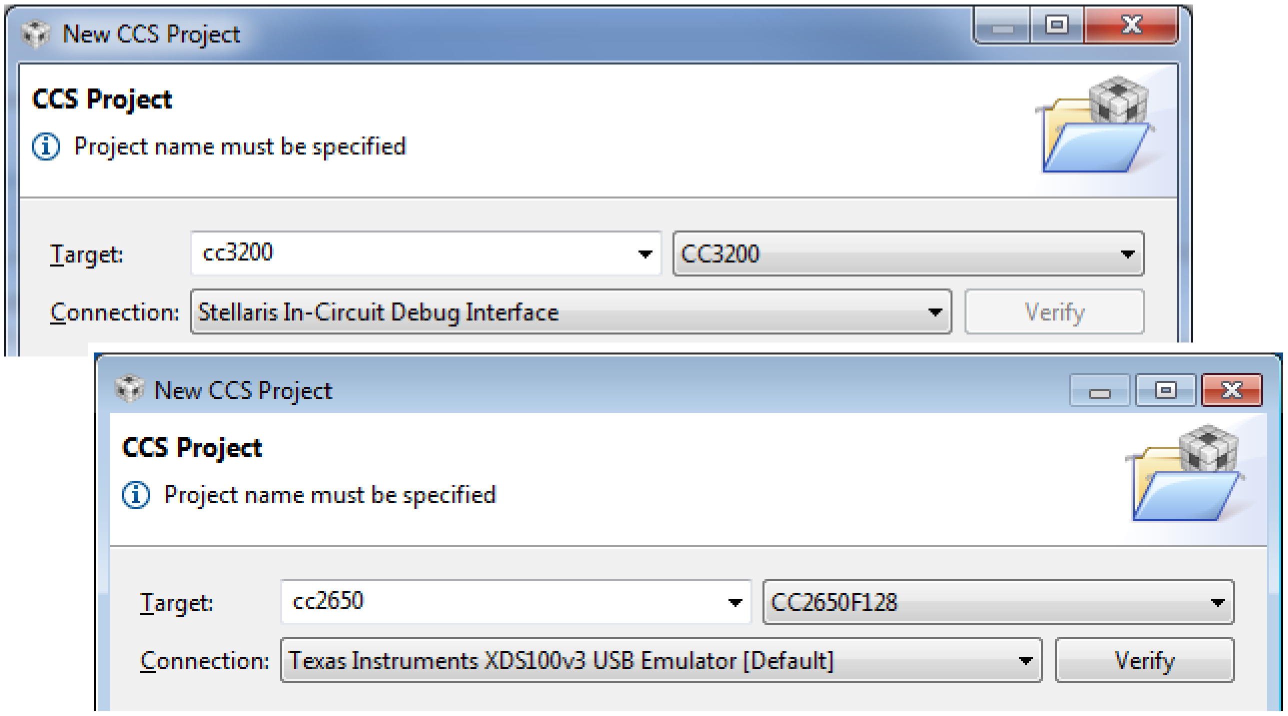
- Most TI-RTOS driver examples use the SysMin System Support module. See the readme files in the individual example projects for details.
- The empty, demo, and most UART examples use the ti.uia.sysbios.LoggingSetup module with stop mode data collection. The UART Console example uses run-time data collection 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).

3.2 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 are not created with the TI Resource Explorer (see Section 3.1). Instead, create empty projects as follows:

1. Choose the **Project > New CCS Project** menu command. (This has the same effect as using the **File > New > CCS Project** menu command.)
2. In the Target field, type part of the name of your device or select the device family from the drop-down list. In the list on the right, select your device.



Project name must be specified

Target: cc3200 CC3200

Connection: Stellaris In-Circuit Debug Interface

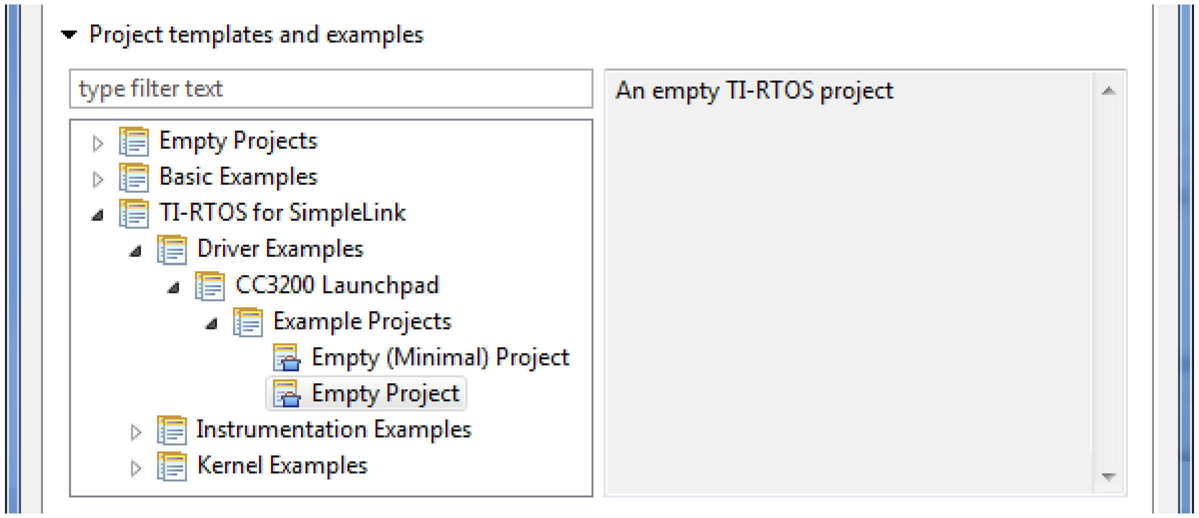
Project name must be specified

Target: cc2650 CC2650F128

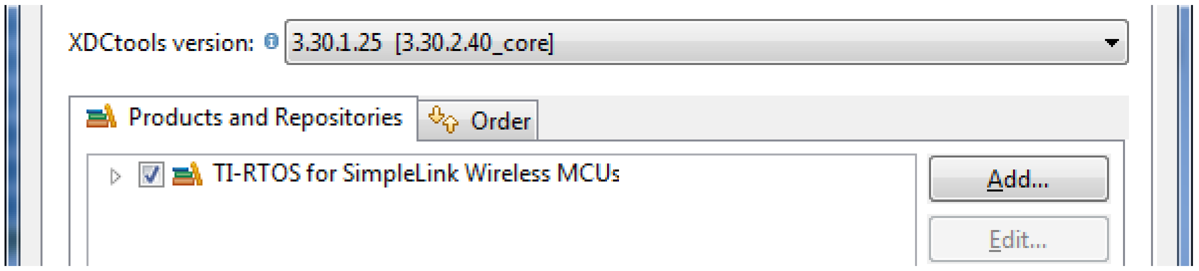
Connection: Texas Instruments XDS100v3 USB Emulator [Default]

3. In the **Connection** field, choose your emulator from the list. For CC26xx SimpleLink devices, choose the XDC100v3. For CC32xx SimpleLink devices, choose the **Stellaris In-Circuit Debug Interface**.
4. If there are multiple cores, select the tab for the core that will run the application.
5. In the **Project name** field, type a name for the project.

6. In the **Project templates and examples** list, expand the TI-RTOS category for your device and select the **Empty Project** item within the list of **Driver Examples**. If no Driver Examples are listed, no TI-RTOS driver examples are provided for the device you selected. Instrumentation and Kernel examples may be available for TI-RTOS, but these examples are the same ones that are available through TI Resource Explorer (Section 3.1).



7. Click **Next**. The next page of the new project wizard shows only the version of TI-RTOS being used. The Target, Platform, and Build-profile to be used by XDCtools to build the application should be filled in and correct.


XDCtools version: 3.30.1.25 [3.30.2.40_core]
Products and Repositories
Order
TI-RTOS for SimpleLink Wireless MCUs
Add...
Edit...

8. Click **Finish**. 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
9. Add to the example as needed to implement your application.

Note: Additional configuration might be needed as you add to the example. For example, if you add networking, you will likely need to increase the heap sizes.

3.3 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 and GNU (CC32xx only). 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 2.4 or Section 2.5), 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\tirtos_simplelink2_##_##_##_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_INSTALLATION_DIR: Full path to the location where TI tools are installed.
   — IAR_COMPILER_INSTALLATION_DIR: Full path to the IAR code generation tools installation.
   — GCC_INSTALLATION_DIR: Full path to the GCC code generation tools installation. (GCC is not supported for CC26xx devices.)
   — TIRTOS_INSTALLATION_DIR: Full path to the TI-RTOS installation.
   — XDCTOOLS_INSTALLATION_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. If you plan to use TI-RTOS with GCC, set the GCC_BUILD variable in the `tirtos.mak` file to true.

```
GCC_BUILD ?= true
```

4. 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_30_##_##_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 a `tirtos_simplelink_2_##_##_##_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 and GNU 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. The `makefile.defs` file in each example is used when cleaning up a previous *.cfg file build within CCS.

3.4 Creating Examples with IAR Embedded Workbench

After you have installed IAR Embedded Workbench and TI-RTOS as described in Section 2.5, you can follow the steps in this section to use the examples that come with TI-RTOS.

3.4.1 Preparing the TI-RTOS Examples for Use in IAR Embedded Workbench

Before using any of the TI-RTOS examples, you must generate the necessary example files and project files needed to build the examples with the IAR Embedded Workbench IDE. After you install TI-RTOS, follow the steps in Section 3.3. You only need to perform those setup steps once.

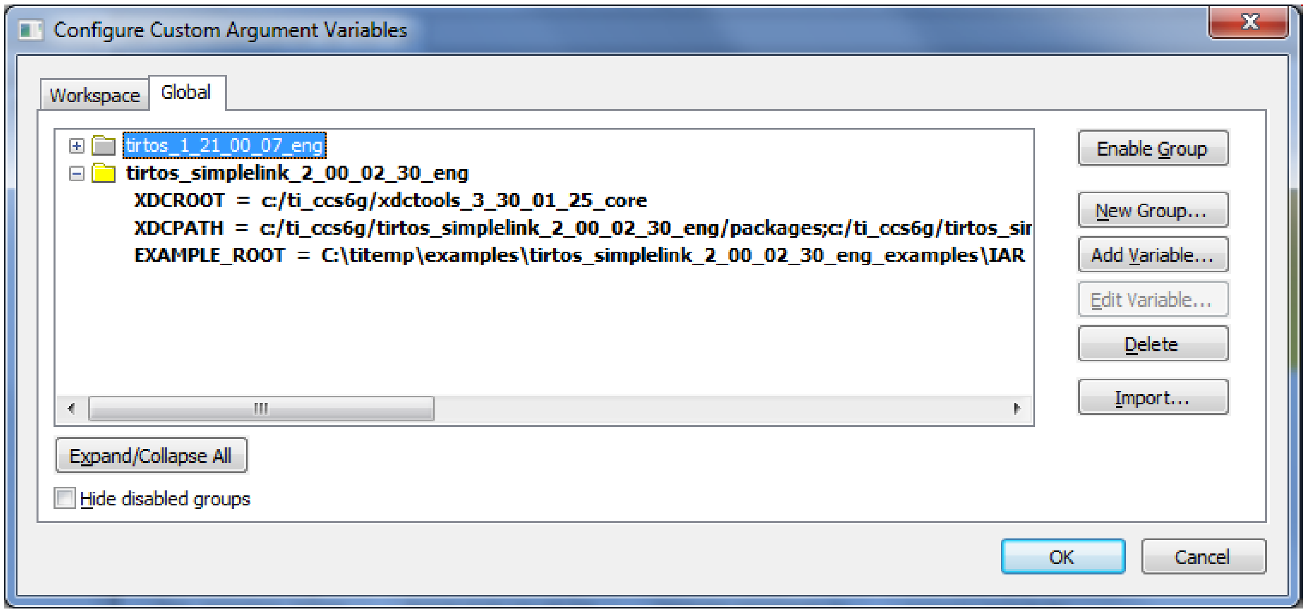
Note: If you installed TI-RTOS using the standalone installer (Section 2.4 or Section 2.5), this step is not necessary because pre-generated examples are included as part of the installation for all supported boards and tool chains. The provided examples are located in the `TIRTOS_INSTALL_DIR\tirtos_simplelink2_##_##_##_examples` directory. Pre-generated examples are not provided if you installed TI-RTOS through the CCS App Center.

3.4.2 Integrating TI-RTOS Examples with IAR IDE

In order for IAR to discover your generated TI-RTOS examples, you must do the following once before you can create, build, and run TI-RTOS examples within IAR:

1. Open IAR Embedded Workbench.
2. Choose **Tools > Options** to open the IDE Options dialog. Select the **Project** category and check the box to **Enable project connections**. Click **OK**.
3. Choose **Tools > Configure Custom Argument Variables**. This opens a dialog that allows you to define paths to integrate TI-RTOS with IAR Embedded Workbench. The IAR folder in the examples you generated with the examplesgen utility, there is a `tirtos_#_##_##_##.custom_argvars` file that sets variables for these paths.
4. Select the **Global** tab, and click **Import**.
5. Browse to the IAR subdirectory of the examples tree you generated with the examplesgen utility. For example, if you specified `C:\myfiles` as the destination, browse to the `C:\myfiles\tirtos_#_##_##_##_examples/IAR` directory.
6. Select the `tirtos_#_##_##_##.custom_argvars` file, and click **Open**.

7. The result should look similar to this. Click **OK** to finish.



```
tirtos_1_21_00_07_eng
tirtos_simplelink_2_00_02_30_eng
  XDCROOT = c:/ti_ccs6g/xdctools_3_30_01_25_core
  XDCPATH = c:/ti_ccs6g/tirtos_simplelink_2_00_02_30_eng/packages;c:/ti_ccs6g/tirtos_sir
  EXAMPLE_ROOT = C:\titemp\examples\tirtos_simplelink_2_00_02_30_eng_examples\IAR
```

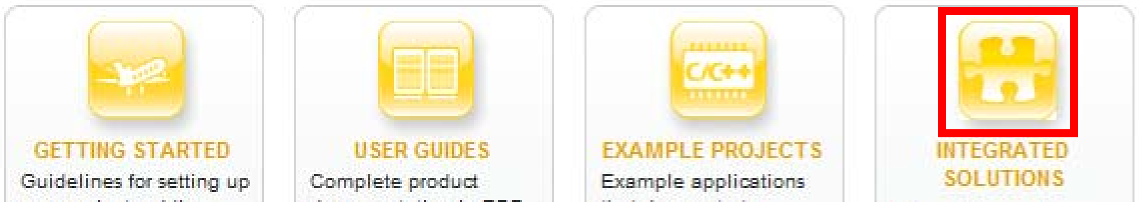
3.4.3 *Creating TI-RTOS Examples with IAR Embedded Workbench*

Follow these steps to import a TI-RTOS example into an IAR project:

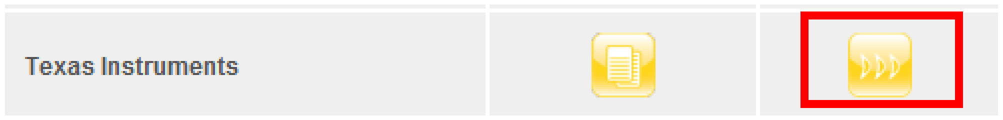
1. First, choose **Project > Create New Project** from the IAR menus.
2. Select the correct **Tool chain**, and choose the **Empty project** template. Click **OK**.
3. Browse to the location where you want to save this project, and type a name for the project file (*.ewp). It is best to use a directory that does not contain other projects or workspaces. Click **Save**.

Note: Do not use names for a project or workspace that contain spaces.

4. If the IAR Information Center is not open, choose **Help > Information Center** from the IAR menus.
5. Click the **Integrated Solutions** link.

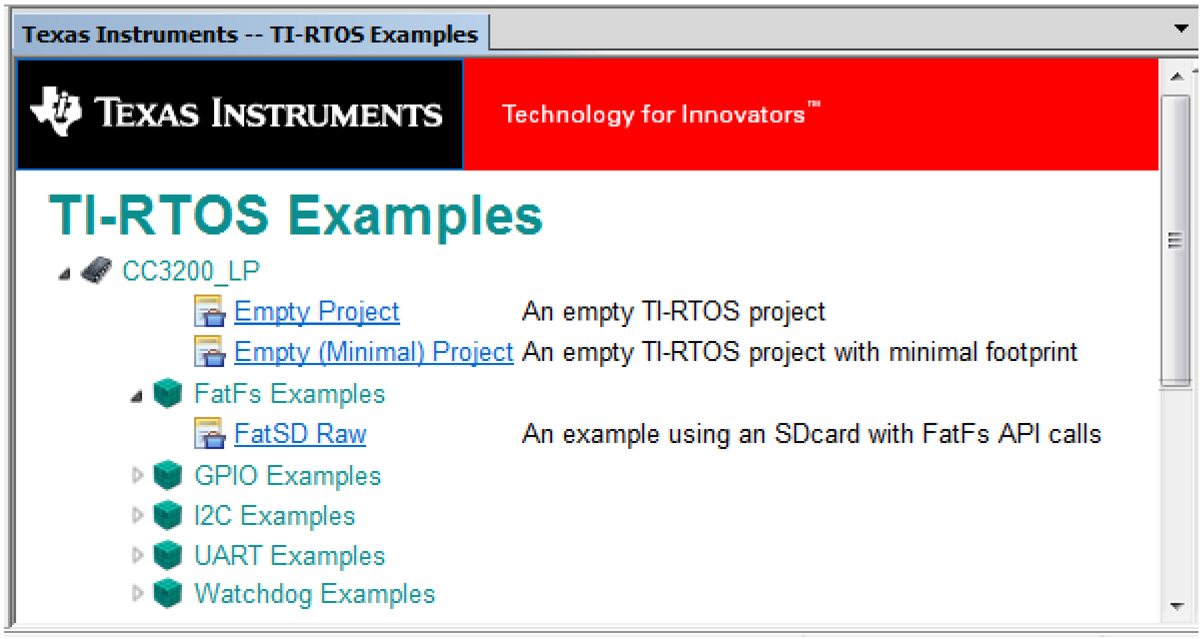


6. In the partners table, click the **Example Projects** button in the Texas Instruments row.



7. Click the "example applications" link to open the **Texas Instruments -- TI-RTOS Examples** page.
8. Click the link to browse example applications.

9. Expand the list of examples. Click on an example to add files for that example to your current project.

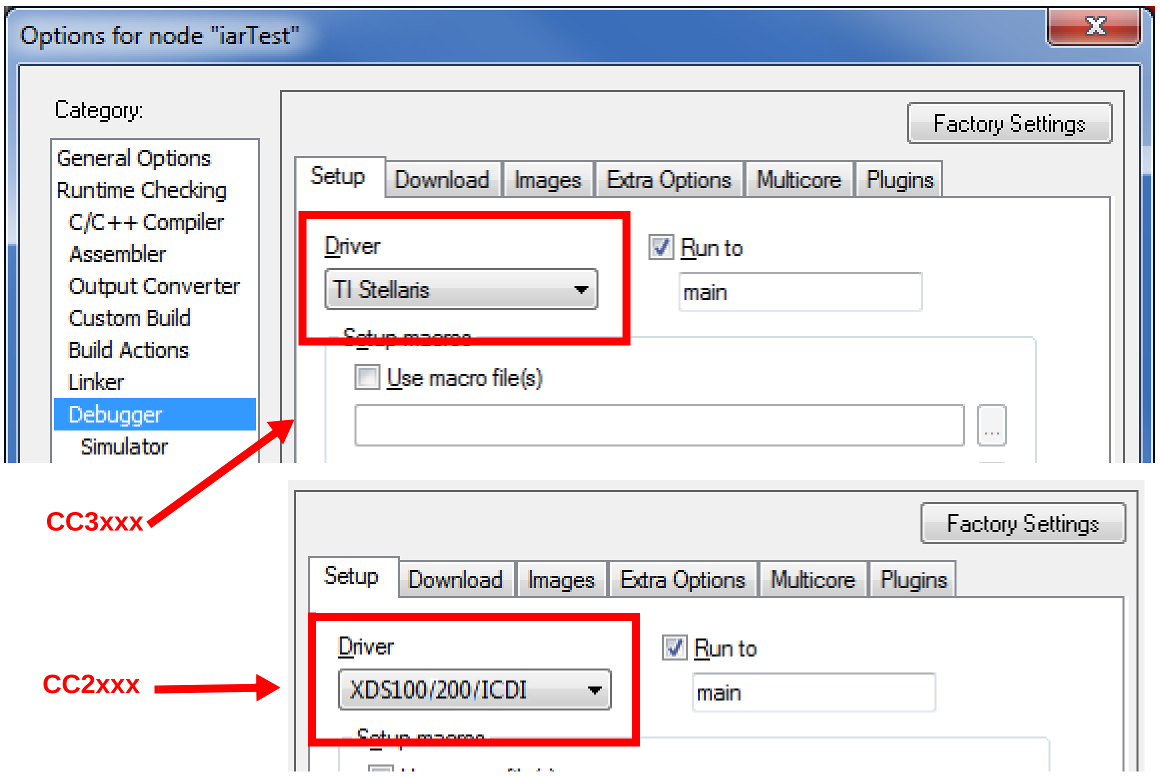


Texas Instruments -- TI-RTOS Examples

TI-RTOS Examples

- CC3200_LP
  - Empty Project — An empty TI-RTOS project
  - Empty (Minimal) Project — An empty TI-RTOS project with minimal footprint
  - FatFs Examples
    - FatSD Raw — An example using an SDcard with FatFs API calls
  - GPIO Examples
  - I2C Examples
  - UART Examples
  - Watchdog Examples

10. Right-click on the project file, and choose **Options**. Select the **Debugger** category. For CC26xx SimpleLink devices, choose the **XDC100/200/ICDI** driver in the **Setup** tab. For CC32xx SimpleLink devices, choose the **TI Stellaris** driver, in the **Setup** tab.



Options for node "iarTest"

Category: General Options, Runtime Checking, C/C++ Compiler, Assembler, Output Converter, Custom Build, Build Actions, Linker, Debugger, Simulator

CC3xxx — Setup tab: Driver: TI Stellaris; Run to: main

CC2xxx — Setup tab: Driver: XDS100/200/ICDI; Run to: main

11. If you are using a CC2xxx target, move to the **Download** tab and put a checkmark in the **Use flash loader(s)** box.

12. Move to the **Plugins** tab and put a checkmark in the box next to **TI-RTOS**. This enables the RTOS Object View (ROV) plugin tools.
13. If you are using a CC2xxx target, select the **XDS100/200/ICDI** category at the end of the Category list. Select the **TI XDS 100v3 USB Emulator, 2-pin CJTAG mode** emulator.
14. Click **OK**. Note that you will need to change these Debugger options separately for each TI-RTOS project you create in IAR.
15. Choose **File > Save Workspace** from the IAR menus.
16. Browse to the location where you want to save this workspace, and type a name for the workspace file (*.eww). It is common to save the project and workspace in the same directory. Click **Save**.

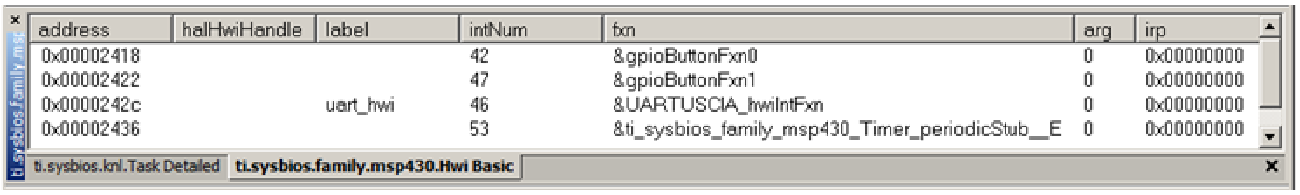
Note: Do not use names for a project or workspace that contain spaces.

3.4.4 Using TI-RTOS Examples for IAR Embedded Workbench

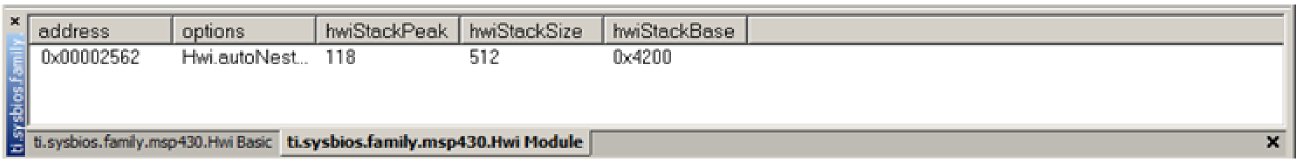
You can use the text editor in IAR to modify *.c and *.cfg files in the example. The graphical configuration tool that is available for editing *.cfg files in CCS is not available within IAR.

To build examples, choose **Project > Make** or press F7. You may be asked to save the workspace. To run examples, choose **Project > Download and Debug** or press Ctrl+D.

You can halt the target and choose any of the tools in the **TI-RTOS** menu to run the RTOS Object View (ROV) plugins. These plugins let you see information such as the peak stack size and addresses. The plugins work similarly to the ROV tool in CCS. However, you open a separate pane for each module that you want to view. See the http://rtsc.eclipse.org/docs-tip/RTSC_Object_Viewer web page for more about using the ROV tool.


ti.sysbios.family.msp
ti.sysbios.knl.Task Detailed
ti.sysbios.family.msp430.Hwi Basic

address	halHwiHandle	label	intNum	fxn	arg	irp
0x00002418			42	&gpioButtonFxn0	0	0x00000000
0x00002422			47	&gpioButtonFxn1	0	0x00000000
0x0000242c		uart_hwi	46	&UARTUSCIA_hwiIntFxn	0	0x00000000
0x00002436			53	&ti_sysbios_family_msp430_Timer_periodicStub__E	0	0x00000000


ti.sysbios.family
ti.sysbios.family.msp430.Hwi Basic
ti.sysbios.family.msp430.Hwi Module

address	options	hwiStackPeak	hwiStackSize	hwiStackBase
0x00002562	Hwi.autoNest...	118	512	0x4200

3.4.5 Using Command Line Builds with TI-RTOS Examples for IAR Embedded Workbench

You can import, build, and run TI-RTOS examples from within IAR Workbench as described in the previous section.

Note: If you installed TI-RTOS for IAR (Section 2.5), pre-built examples are included as part of the installation for all supported boards and tool chains. Pre-built examples are not provided if you installed TI-RTOS through the CCS App Center. The pre-built examples are located in `TIRTOS_INSTALL_DIR\tirtos_simplelink_2_##_##_##_examples`.

Alternately, if you want to build TI-RTOS examples from the command line, you can use the provided makefiles outside the IAR Embedded Workbench IDE. The executable files you build will have a *.out file extension. To build an example project from the command line, follow these steps:

1. Move to the board-level directory of the destination you specified for the `examplesgen` command. For example, if you built the examples tree in C:\myfiles and you are using the CC3200 board, move to the `C:\myfiles\tirtos_simplelink_2_##_##_##_examples\IAR\CC3200_LP` directory.
2. Use a text editor to edit the `makedefs` file in that directory.
3. Modify the CODEGEN_INSTALLATION_DIR variable in the `makedefs` file as needed to make it point to the location of the IAR compiler. For example:

```
CODEGEN_INSTALLATION_DIR = C:/Program Files (x86)/IAR Systems/Embedded Workbench 6.5/430
      or
CODEGEN_INSTALLATION_DIR = C:/Program Files (x86)/IAR Systems/Embedded Workbench 6.5/arm
```

4. To build all the examples for a particular board, navigate to that board's directory and run the following command, where <xdctools_dir> is the location of the XDCtools component that was installed with TI-RTOS. For example, `C:\myfiles\xdctools_3_30_##_##_core`.

```
> <xdctools_dir>\gmake all
```

5. To build only one example, go to that example’s directory and run the same `gmake all` command.

After you build a TI-RTOS example from the command line, follow these steps to use IAR to load the externally-built executable onto the target and debug the application:

1. Choose **Project > Create New Project** from the IAR menus.
2. Select the **Tool chain**, and choose the **Externally built executable** template. Click **OK**.
3. Browse to the location where you want to save this project, and type a name for the project file (*.ewp). Click **Save**.
4. Delete any readme.txt file that is added to the project.
5. Choose **File > Save Workspace** from the IAR menus. Browse to the location where you want to save this workspace, and type a file name for the workspace file (*.eww). Click **Save**.
6. Choose **Project > Options**.
7. In the **General Options** category, go to the **Target** tab and select your specific **Device**.
8. In the **Debugger** category of the Options dialog, replace the default **Simulator** driver with the appropriate TI debugger. For CC26xx SimpleLink devices, choose **XDC100v3**. For CC32xx SimpleLink devices, choose the **TI Stellaris** debugger.
9. If you are using a CC2xxx target, move to the **Download** tab and put a checkmark in the **Use flash loader(s)** box.
10. Move to the **Plugins** tab and put a checkmark in the box next to **TI-RTOS**. This enables the RTOS Object View (ROV) plugin tools.
11. If you are using a CC2xxx target, select the **XDS100/200/ICDI** category at the end of the Category list. Select the **TI XDS 100v3 USB Emulator, 2-pin CJTAG mode** emulator.
12. Click **OK**. Note that you will need to change these Debugger options separately for each project you create in IAR.
13. Choose **Project > Add Files**. Change the file types filter to show **All Files (*.*)**.

14. Browse to the location of the executable file you built. Select the <*examplename*>.out executable file and click **Open**.
15. Choose **Project > Download and Debug** to load the executable onto the target hardware.
16. Use the debugging features in IAR to debug the application.

3.4.6 Managing Program Stack Sizes for TI-RTOS Examples for IAR

For IAR, the program stack size for each device is set in its linker command file. The Program.stack line in an example's configuration file is not recognized by the IAR environment, so the value you specify is not the actual stack size for your example.

For SimpleLink boards, the *.icf linker command file has lines like the following to set the program stack size:

```
/*-Sizes-*/
define symbol __ICFEDIT_size_cstack__  =  0x1000;
define symbol __ICFEDIT_size_heap__     = 0x2000;
```

Note: The default stack sizes provided are sufficient for the TI-RTOS examples.

SYS/BIOS manages its own heap via the BIOS.heapSize configuration parameter. Therefore, the IAR heap can be set to 0.

3.5 Summary of Example Peripheral Use and Target Support

The boards, components, and hardware used by the TI-RTOS examples are shown in the following table. Details about these examples are provided in the readme files in the example projects.

There are several 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 remaining examples show how to use a specific peripheral.

Table 3-1. TI-RTOS for SimpleLink Examples: Boards, Components, and Drivers

Example	CC3200 Launchpad	CC2650 SimpleLink Dev Kit	CC2650 SimpleLink SensorTag	TI-RTOS Kernel (SYS/BIOS)	SD Card / SDSPI	I^{2}C	GPIO	LCD	PIN	Power	PWM	SPI	Watchdog Timer	WiFi [3]	UART	UIA
Empty and Empty (Minimal) TI-RTOS Projects	X	X	X	X			X									X
FatSD: FatFs File Copy	x[1]			X	X		X									
FatSD Raw: FatFs File Copy using FatFs APIs	x[1]			X	X		X									
GPIO Interrupt	X			X			X									
I^{2}C TMP006	x[2]		X	X		X	X									
LCD SmartRF06EB		X		X				X		X						
Pin Interrupt		X	X	X					X	X						
PWM LED	X			X			X				X					
UART Echo	X	X	X	X			X			X					X	X
UART Logging	X			X			X								X	X
Watchdog	X			X			X						X			

[1]This example requires an SD Card reader BoosterPack.

[2]To use this example, close jumpers J2 and J3.

[3]Wireless examples for TI-RTOS are provided with the SimpleLink Wi-Fi CC3200 Software Development Kit (SDK). To access those examples, download the kit at http://www.ti.com/tool/cc3200sdk. For information about using the Wi-Fi examples in the SDK, see the TI-RTOS CC3200Wireless wiki page.

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 TI Resource Explorer to create a project.

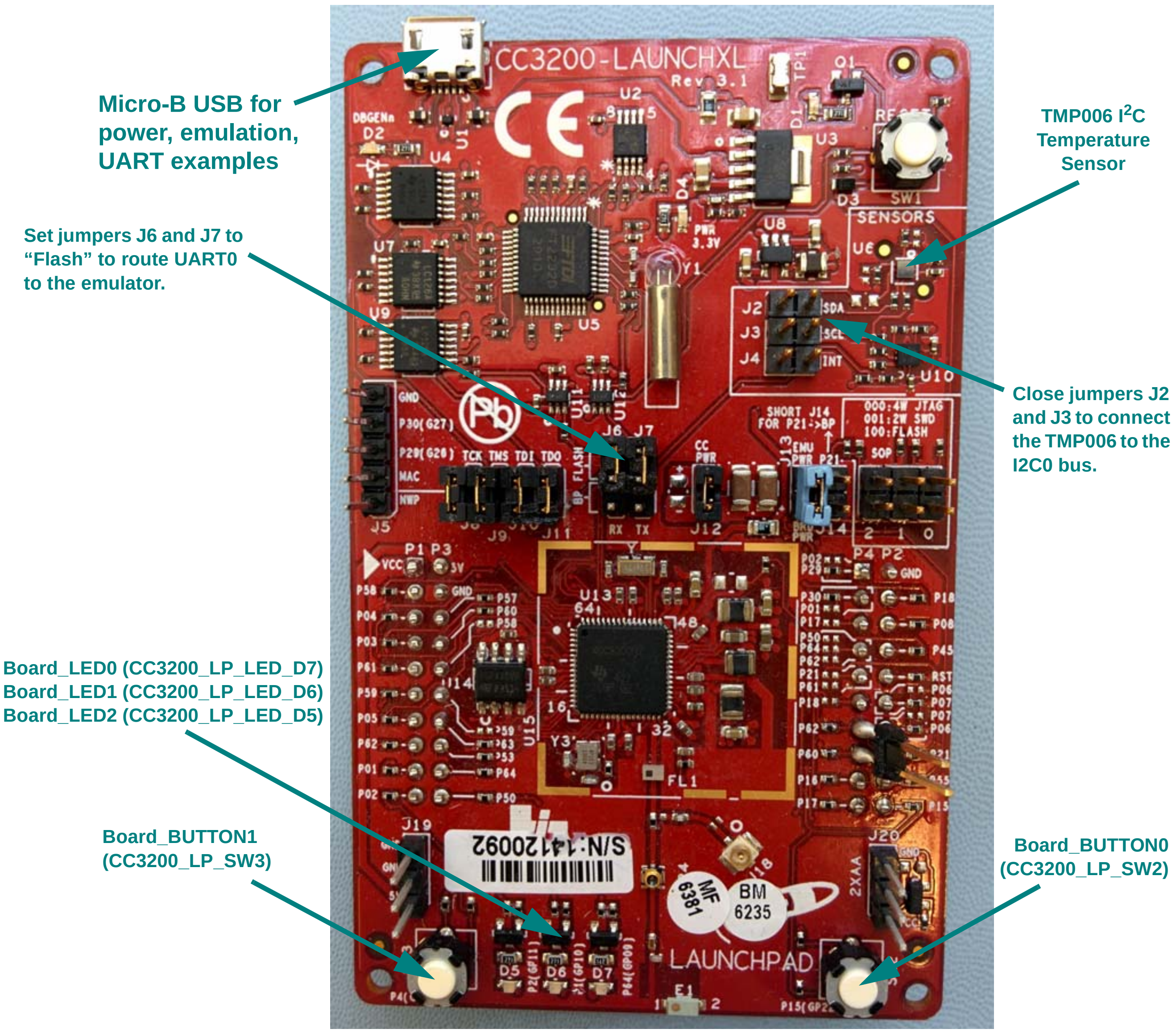
The <*example_name*>_readme files contain the following types of information:

- 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.

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.

3.6 CC3200 SimpleLink LaunchPad Settings and Resources

The CC3200 SimpleLink Launchpad contains a CC3200HZ device.


Micro-B USB for power, emulation, UART examples
TMP006 I²C Temperature Sensor
Set jumpers J6 and J7 to "Flash" to route UART0 to the emulator.
Close jumpers J2 and J3 to connect the TMP006 to the I2C0 bus.
Board_LED0 (CC3200_LP_LED_D7)
Board_LED1 (CC3200_LP_LED_D6)
Board_LED2 (CC3200_LP_LED_D5)
Board_BUTTON1 (CC3200_LP_SW3)
Board_BUTTON0 (CC3200_LP_SW2)

The Micro-B connector on the CC3200 LaunchPad can be used for power, debugger emulation, and UART communications.

Jumper Settings:

- To use the TMP006 I^{2}C temperature sensor, close jumpers J2 and J3.
- To use the UART connected to the emulator, move jumpers J6 and J7 into the "flash" (2-3) position.

Switch Settings:

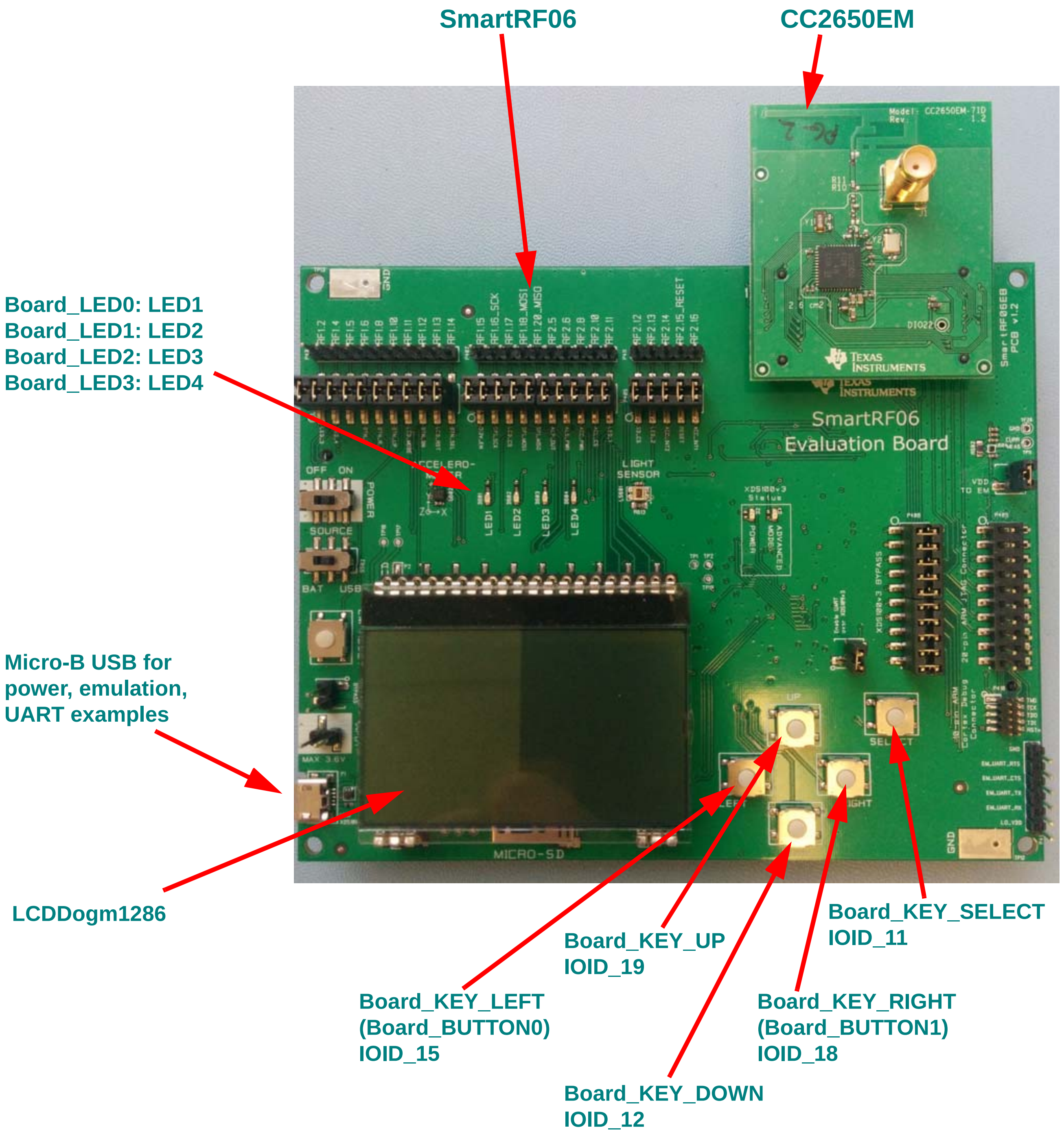
- SW2: Some examples use PIN 15 as a GPIO input.
- SW3: Some examples use PIN 04 as a GPIO input.

Resources Used:

- **TI-RTOS Kernel (SYS/BIOS).** Uses the first general-purpose timer available and that timer's associated interrupts. Generally, this will be Timer 0. The TI-RTOS Kernel manages the Interrupt Vector Table.
- **TI-RTOS.**
  - **GPIOs.** The GPIO driver uses 3 output pins for the onboard LEDs and 2 input pins for switches SW2 (PIN 15) and SW3 (PIN 04).
  - **I^{2}C.** The I^{2}C driver is configured on I2CA0 to support various BoosterPacks or onboard peripherals.
  - **PWM.** The PWM driver uses the onboard LEDs D5 (PIN 02) and D6 (PIN 01). While these pins coincide with GPIO driver pins, they are configured for the PWM driver for the PWM examples. The GPIO driver APIs should not be used.
  - **SD Card.** Uses FatFs and the SDSPI driver on GSPI without interrupts to read and write to files.
  - **Serial.** The UART driver uses UARTA0, which is attached to the FTDI USB chip to facilitate serial communications.
  - **SPI.** The SPI driver uses GSPI for Board_SPI0 and LSPI for Board_SPI1.
  - **Watchdog.** The Watchdog example uses the Watchdog Timer and its associated interrupt.

3.7 CC2650 SimpleLink Development Kit Settings and Resources

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



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 first general-purpose timer available and that timer's associated interrupts. Generally, this will be Timer 0. 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 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 CC2650 specific LCD driver controls the DOGM1286 LCD controller via the SPI driver.
  - — **I^{2}C.** The I^{2}C driver is configured on I2C0.
  - — **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.
  - — **Watchdog.** The Watchdog driver uses the Watchdog timer.

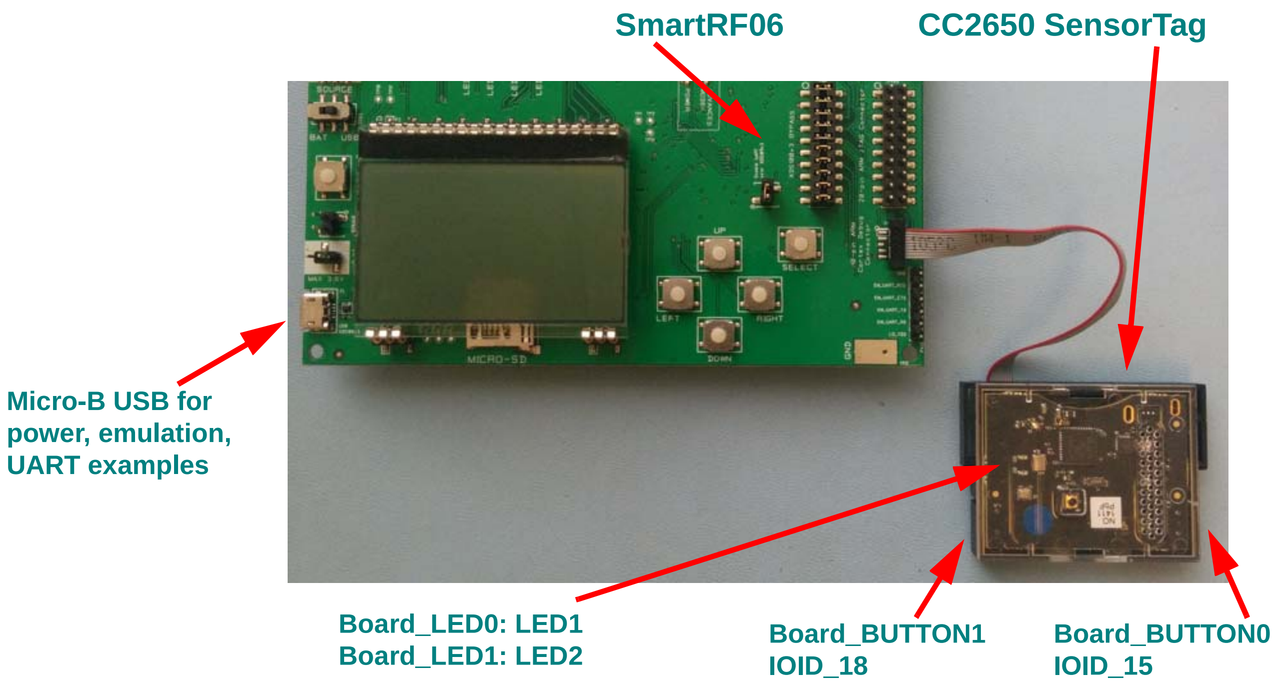
3.8 CC2650 SimpleLink SensorTag Settings and Resources

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

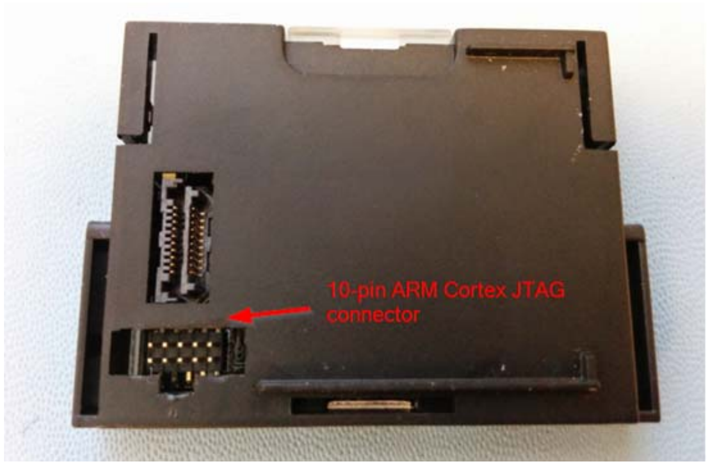
The CC2650 SensorTag can connects to an external debugger in either of two ways:

- via a low cost "DevPack" debugger designed to be interfaced via the DevPack connector
- via the optional standard 10-pin ARM Cortex JTAG connector.

In this section, the SensorTag is connected to an external SmartRF06 development board via the standard 10-pin ARM Cortex JTAG connector. The Micro-B connector of the SmartRF06 is used for power, debugger emulation, and UART communications.

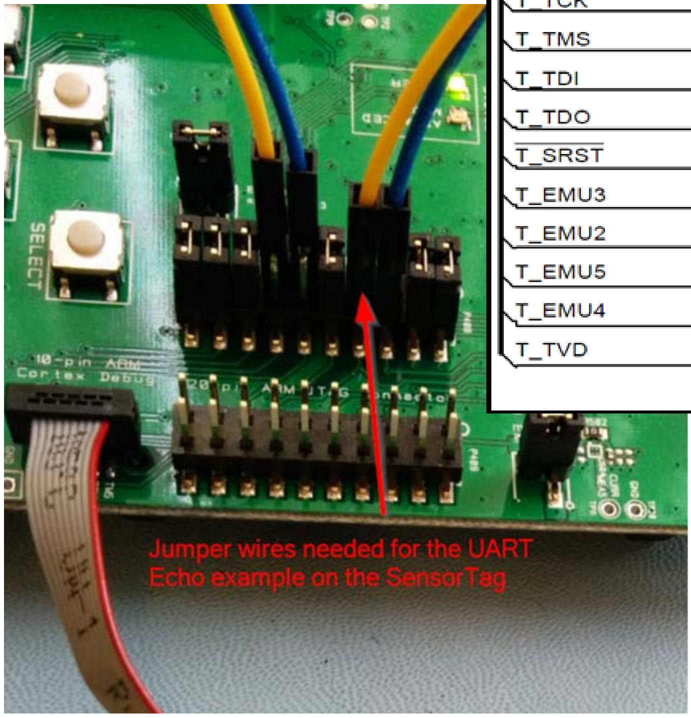


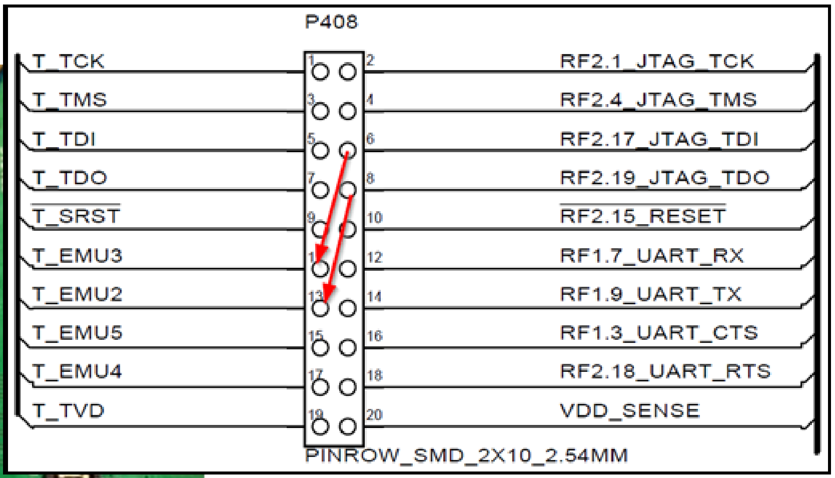
The 10-pin ARM Cortex JTAG connector is located on the back of the CC2650 SensorTag. Note that by default the JTAG connector is concealed by the plastic housing.



Jumper Settings:

For the UART Echo example, the SensorTag's UART data pins are multiplexed with the JTAG_TDO and JTAG_TDI pins. On the SmartRF06 development board, these pins need to be rerouted from the JTAG debugger to the onboard UART - USB converter (XDS100v3) using jumper bypasses as shown in the photo and schematic diagram below.


Jumper wires needed for the UART Echo example on the SensorTag
SELECT
10-pin ARM Cortex Debug



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 first general-purpose timer available and that timer's associated interrupts. Generally, this will be Timer 0. 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.
  - **I^{2}C.** The I^{2}C 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.
- — **Watchdog.** The Watchdog example uses the Watchdog Timer.

3.9 BoosterPacks

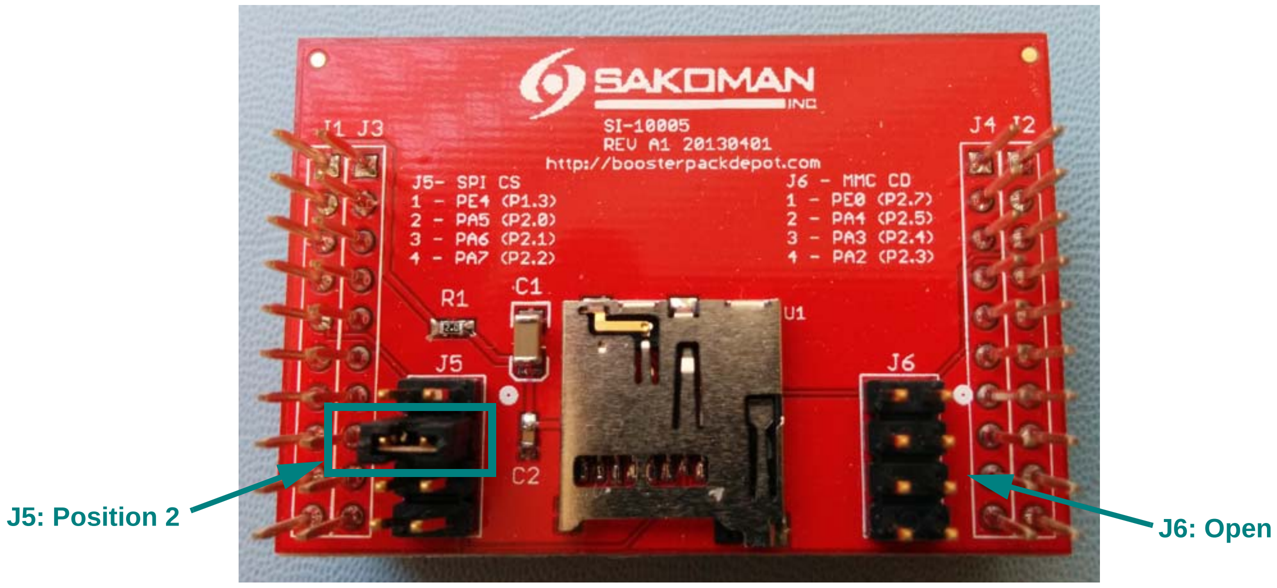
Note: This section applies to CC32xx boards only; it does not apply to CC26xx boards.

Several BoosterPack boards are used with the TI-RTOS examples. This section described those boards and provides any special notes about installing the board.

3.9.1 SD Card BoosterPack

A microSD BoosterPack or SD Card BoosterPack should be used with examples that require an SD Card reader on target boards that do not include an SD Card reader. This board may be used with the CC3200.

The figure below shows the jumper needed on the microSD BoosterPack to make it pin-compatible with the SD Card BoosterPack.

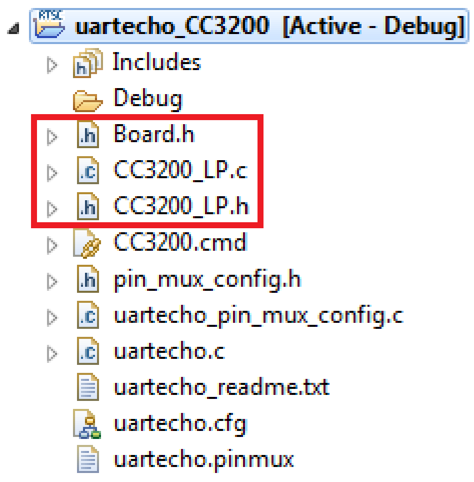


3.10 <Board>.c file and PinMux Tool Integration

Note: This section applies to CC32xx boards only; it does not apply to CC26xx boards.

For every board supported in each TI-RTOS product (for example, TI-RTOS for SimpleLink Wireless MCUs and TI-RTOS for MSP430), 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 the above picture. These files are specific to a board. The same file is used for all driver examples for that board.


uartecho_CC3200 [Active - Debug]
Includes
Debug
Board.h
CC3200_LP.c
CC3200_LP.h
CC3200.cmd
pin_mux_config.h
uartecho_pin_mux_config.c
uartecho.c
uartecho_readme.txt
uartecho.cfg
uartecho.pinmux

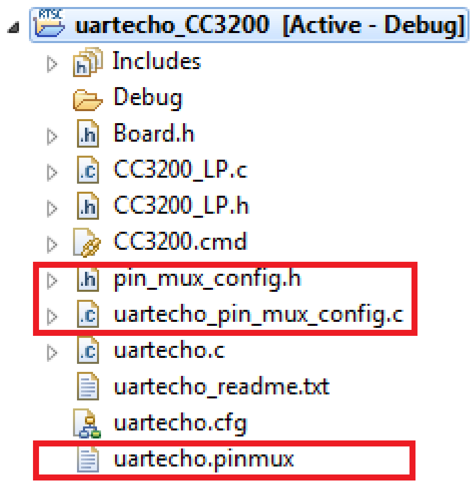
For TI-RTOS products other than TI-RTOS for CC32xx SimpleLink Wireless MCUs, all the necessary pinmuxing is performed in the <Board>.c/.h files. For example, here is the I^{2}C initialization code from the EK_TM4C1294XL.c board file.

```
void EK_TM4C1294XL_initI2C(void)
{
    /* I2C0 Init */
    /* Enable the peripheral */
    SysCtlPeripheralEnable(SYSCTL_PERIPH_I2C0);

    /* Configure the appropriate pins to be I2C instead of GPIO. */
    GPIOPinConfigure(GPIO_PB2_I2C0SCL);
    GPIOPinConfigure(GPIO_PB3_I2C0SDA);
    GPIOPinTypeI2CSCL(GPIO_PORTB_BASE, GPIO_PIN_2);
    GPIOPinTypeI2C(GPIO_PORTB_BASE, GPIO_PIN_3);
    ...
    I2C_init();
}
```

For the TI-RTOS for SimpleLink Wireless MCUs product, the TI PinMux Tool was used to generate the pinmux code for the examples. You can install the TI PinMux Tool from the CCS App Center.

For example, the pin_mux_config.h and uartecho_pin_mux_config.c files were generated by the TI PinMux Tool based on the uartecho.pinmux project. The generated pin_mux_config.c file was renamed to uartecho_pin_mux_config.c so that the pinmux project and generated files for each driver example are uniquely named.


uartecho_CC3200 [Active - Debug]
Includes
Debug
Board.h
CC3200_LP.c
CC3200_LP.h
CC3200.cmd
pin_mux_config.h
uartecho_pin_mux_config.c
uartecho.c
uartecho_readme.txt
uartecho.cfg
uartecho.pinmux

For SimpleLink boards, the <Board>.c file accommodates the use of the PinMux Tool. For example, the Board_initGeneral() function for the CC3200 is as follows, where the PinMuxConfig() function is in the generated uartecho_pin_mux_config.c file).

```
void CC3200_LP_initGeneral(void)
{
    PinMuxConfig();
}
```

The pinxmux code is removed from the peripheral initialization function. For example in the CC3200_LP_initI2C function makes only the driver initialization call:

```
void CC3200_LP_initI2C(void)
{
    I2C_init();
}
```

When peripherals are added or removed, the driver configuration structures in the <Board>.c/.h files must be manually changed. For example if an additional I^2C is added, the following structures in CC3200_LP.c must be manually updated (as well as the enums in the CC3200_LP.h file):

```
I2CCC3200_Object i2cCC3200Objects[ CC3200_LP_I2CCOUNT ];

/* I2C configuration structure */
const I2CCC3200_HWAttrs i2cCC3200HWAttrs[ CC3200_LP_I2CCOUNT ] = {
    { I2CA0_BASE, INT_I2CA0 }
};

const I2C_Config I2C_config[] = {
    { &I2CCC3200_fxnTable, &i2cCC3200Objects[0], &i2cCC3200HWAttrs[0] },
    { NULL, NULL, NULL }
};
```

Please refer to Section 5.2, "Driver Framework" in the *TI-RTOS User's Guide* for details on managing these structures.

Note that a #if directive in the <Board>.c file is used to include or exclude initialization code. For example

```
#if TI_DRIVERS_I2C_INCLUDED
#include <ti/drivers/I2C.h>
#include <ti/drivers/i2c/I2CCC3200.h>
#include <driverlib/i2c.h>

I2CCC3200_Object i2cCC3200Objects[CC3200_LP_I2CCOUNT];

...

void CC3200_LP_initI2C(void)
{
    I2C_init();
}

#endif /* TI_DRIVERS_I2C_INCLUDED */
```

The #defines are generated and set to 1 when a driver is included by the .cfg file. It is set to 0 when the driver is not included in the .cfg file.

3.11 Using Driverlib in ROM

Note: This section applies to CC32xx boards only; it does not apply to CC26xx boards.

The ROM on the CC3200 rev 1.33 includes a version of driverlib. By default, TI-RTOS drivers (such as UART and SDSPI) and examples do not use this ROM version of driverlib. Instead they link with driverlib functions in the `<tirtos_install_dir>/products/CC3200_driverlib_<version>/driverlib` library. However, both the TI-RTOS drivers and examples call MAP_xyz driverlib functions with the same names and calling syntax as those in ROM. They can thus be rebuilt to use the ROM functions (as described in the rest of this section).

To use the driverlib functions in ROM, you will need to define TARGET_IS_CC3200 and rebuild both the TI-RTOS driver libraries and the application. Follow these steps to perform the necessary builds:

1. Edit the `<tirtos_install_dir>\tirtos.bld` file and add TARGET_IS_CC3200 into the ccOpts definition as follows:

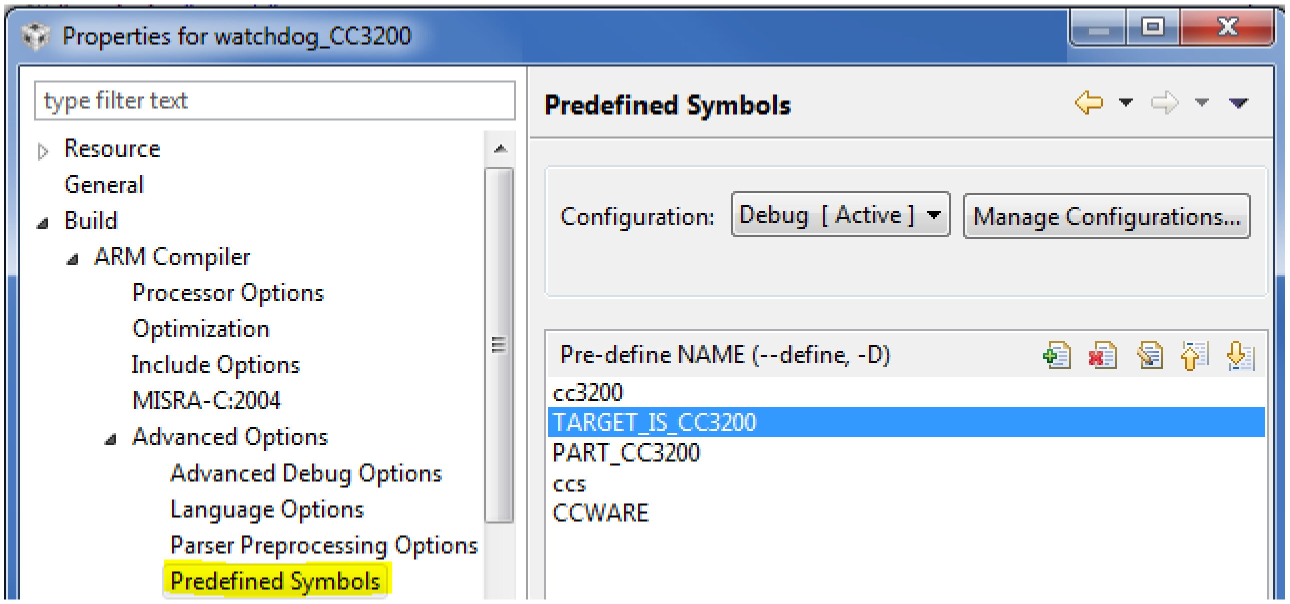
```
var ccOpts = {
    "ti.targets.arm.elf.M4" : " -ms -g --gcc --define=ccs -DTARGET_IS_CC3200 ",
    "iar.targets.arm.M4"    : " -Dewarm -DIAR -DTARGET_IS_CC3200 ",
    "gnu.targets.arm.M4"    : " -g -D gcc -DTARGET_IS_CC3200 ",
};
```

2. Rebuild the driver libraries as explained in the “Rebuilding TI-RTOS” chapter in the TI-RTOS User Guide (SPRUHD4). For example:

```
% ..\<xdctools>\gmake -f tirtos.mak drivers
```

3. Define TARGET_IS_CC3200 for builds of the application project.

In CCS or IAR you can modifying the project properties to add the predefined symbol. For example for project that uses a TI compiler within CCS, you can choose **Project > Properties** from the menus and go to the **Build > ARM Compiler > Advanced Options > Predefined Symbols** category to add TARGET_IS_CC3200 to the list of Pre-defined names.



Predefined Symbols

Configuration: Debug [Active]

Pre-define NAME (--define, -D)

cc3200
TARGET_IS_CC3200
PART_CC3200
ccs
CCWARE

If you are not using an IDE, add TARGET_IS_CC3200 to the makedefs file in the generated command-line examples. For example, this command line for the TI compiler includes the symbol definition (in bold).

```
CFLAGS =  -mv7M4 --code_state=16 --abi=eabi -me -DPART_CC3200 -g
--display_error_number --diag_warning=255 --diag_wrap=off -DTARGET_IS_CC3200
-Dccs -DCCWARE -I$(CCWARE_INSTALLATION_DIR) -I$(CCWARE_INSTALLATION_DIR)/inc
-I$(CCWARE_INSTALLATION_DIR)/driverlib
```

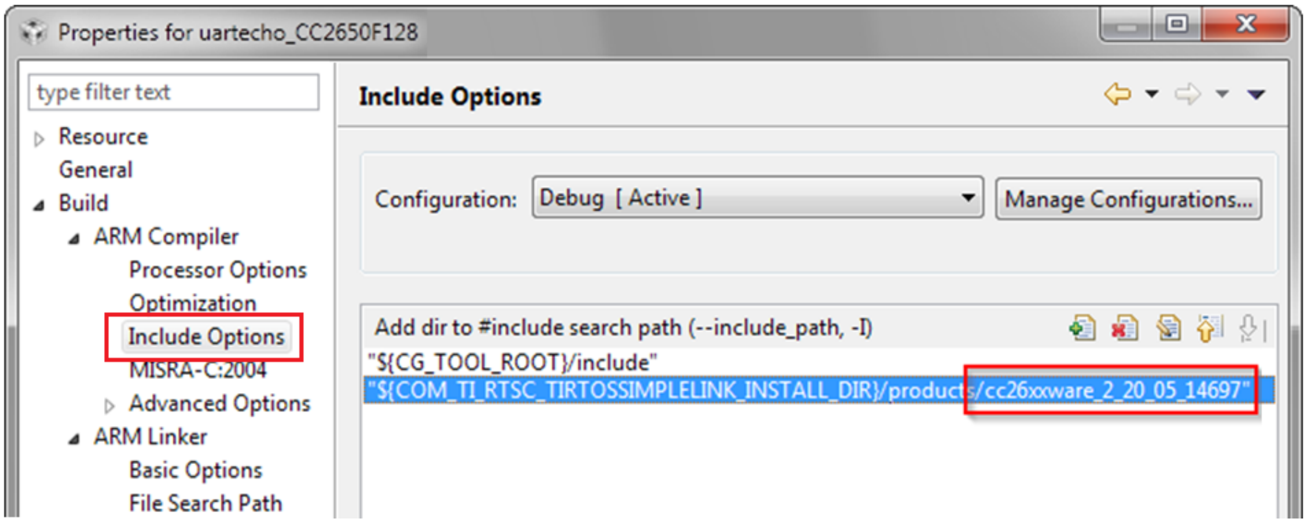
4. Rebuild your application.

3.12 Updating driverlib for CC26xx

To build TI-RTOS examples against different versions of CC26xxWare, follow these instructions:

3.12.1 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 PG2.2 devices, use "cc26xxware_2_20_05_14697" or later. For example:

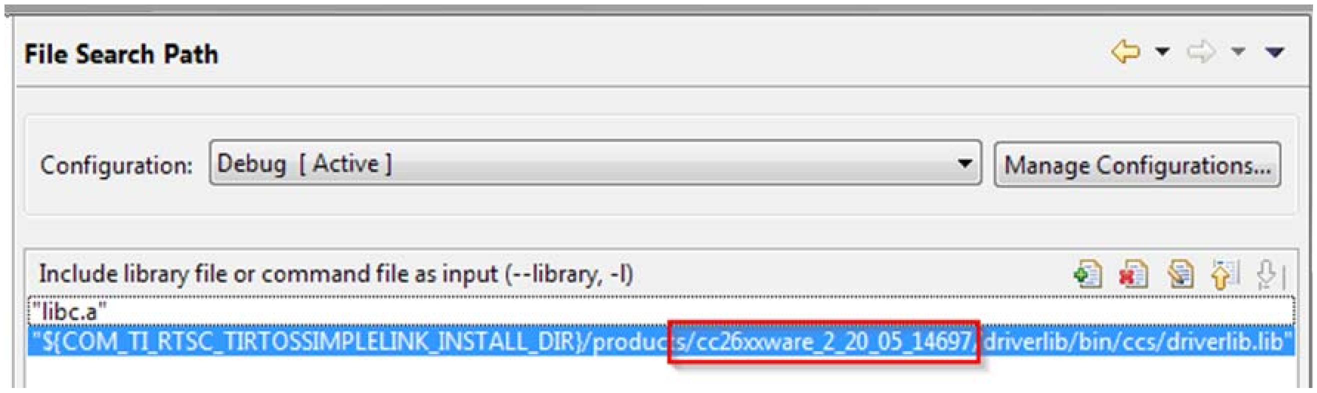


Configuration: Debug [Active]

Add dir to #include search path (--include_path, -I)

"${CG_TOOL_ROOT}/include"
"${COM_TI_RTSC_TIRTOSSIMPLELINK_INSTALL_DIR}/products/cc26xxware_2_20_05_14697"

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 PG2.2 devices, use "cc26xxware_2_20_05_14697" or later. For example:



Configuration: Debug [Active]

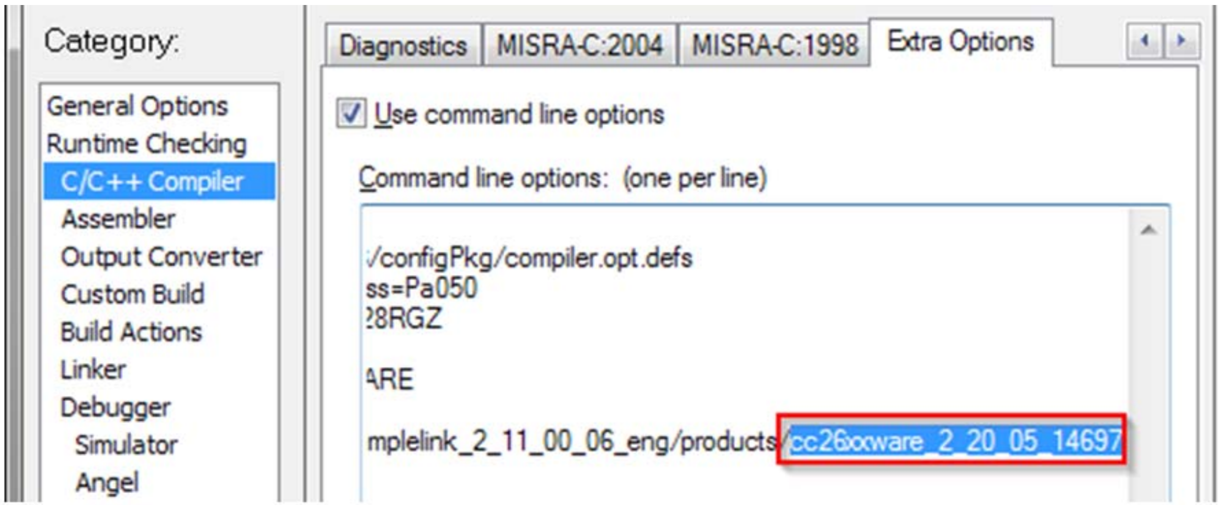
Include library file or command file as input (--library, -l)

"libc.a"
"${COM_TI_RTSC_TIRTOSSIMPLELINK_INSTALL_DIR}/products/cc26xxware_2_20_05_14697/driverlib/bin/ccs/driverlib.lib"

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

3.12.2 Building with IAR

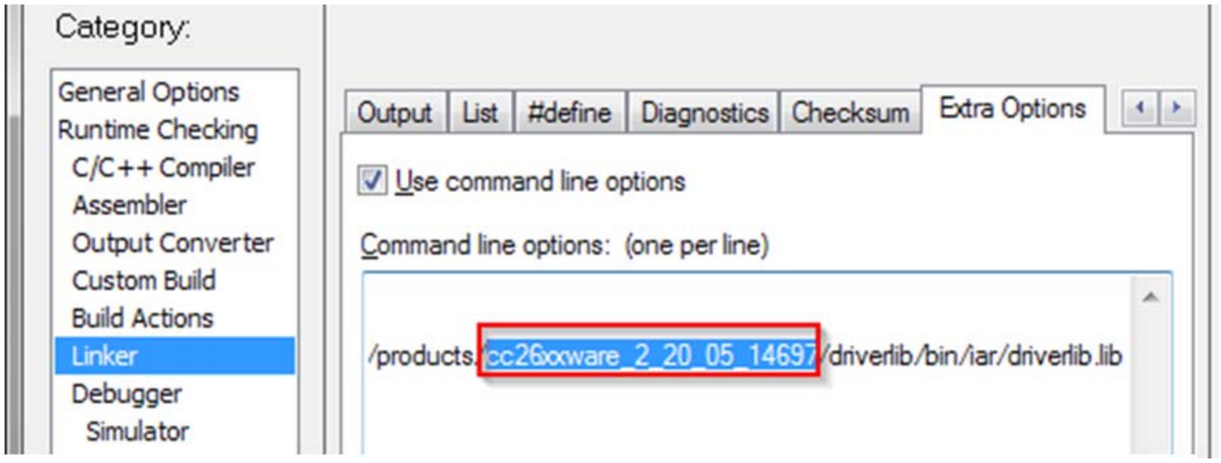
1. Open an existing TI-RTOS project in IAR Embedded Workbench.
2. Choose **Project > Options** from the IAR menus.
3. In the Options dialog, select the **C/C++ Compiler** category and then the **Extra Options** tab.
4. Edit the search path to include (`-I`) the location of the desired version of CC26xxWare. For PG2.2 devices, use "cc26xxware_2_20_05_14697" or later.



```
/configPkg/compiler.opt.defs
ss=Pa050
28RGZ

ARE
mplelink_2_11_00_06_eng/products/cc26xxware_2_20_05_14697
```

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. For PG2.2 devices, use "cc26xxware_2_20_05_14697" or later.

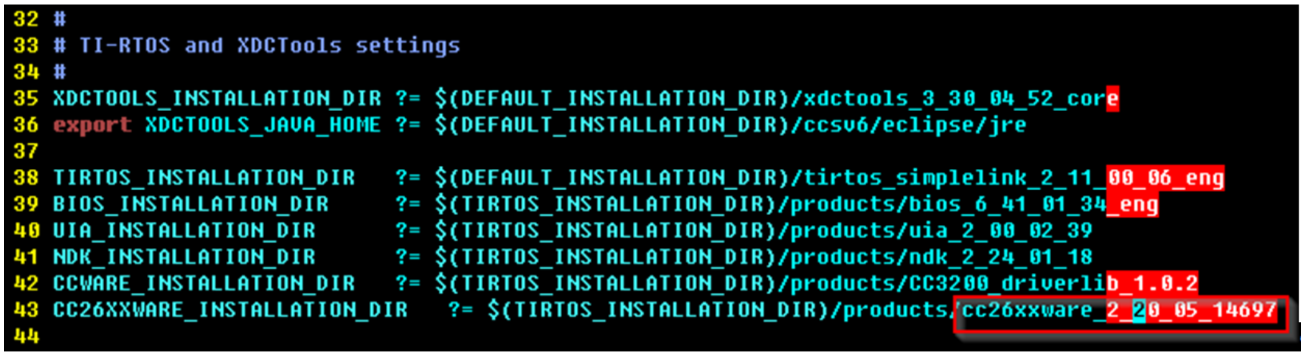


```
/products/cc26xxware_2_20_05_14697/driverlib/bin/iar/driverlib.lib
```

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

3.12.3 *Building with a Makefile*

1. Open the `tirtos.mak` file in the TI-RTOS installation directory.
2. Edit the definition of CC26XXWARE_INSTALLATION_DIR to point to the desired version of CC26xxWare. For PG2.2 devices, use "cc26xxware_2_20_05_14697" or later.



```
#
# TI-RTOS and XDCTools settings
#
XDCTOOLS_INSTALLATION_DIR ?= $(DEFAULT_INSTALLATION_DIR)/xdctools_3_30_04_52_core
export XDCTOOLS_JAVA_HOME ?= $(DEFAULT_INSTALLATION_DIR)/ccsv6/eclipse/jre

TIRTOS_INSTALLATION_DIR   ?= $(DEFAULT_INSTALLATION_DIR)/tirtos_simplelink_2_11_00_06_eng
BIOS_INSTALLATION_DIR     ?= $(TIRTOS_INSTALLATION_DIR)/products/bios_6_41_01_34_eng
UIA_INSTALLATION_DIR      ?= $(TIRTOS_INSTALLATION_DIR)/products/uia_2_00_02_39
NDK_INSTALLATION_DIR      ?= $(TIRTOS_INSTALLATION_DIR)/products/ndk_2_24_01_18
CCWARE_INSTALLATION_DIR   ?= $(TIRTOS_INSTALLATION_DIR)/products/CC3200_driverlib_1.0.2
CC26XXWARE_INSTALLATION_DIR   ?= $(TIRTOS_INSTALLATION_DIR)/products/cc26xxware_2_20_05_14697

```

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\tirtos_simplelink_2_11_00_07_examples\TI\CC2650DK

<xdctools_root_dir>/gmake
```

Chapter 4

SPRUHU8B—January 2015

Configuring TI-RTOS

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

Topic **Page**

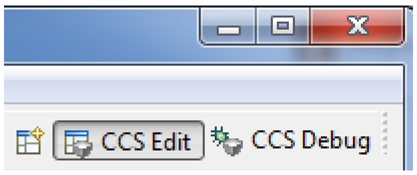
4.1 Starting the Configuration Tool

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.

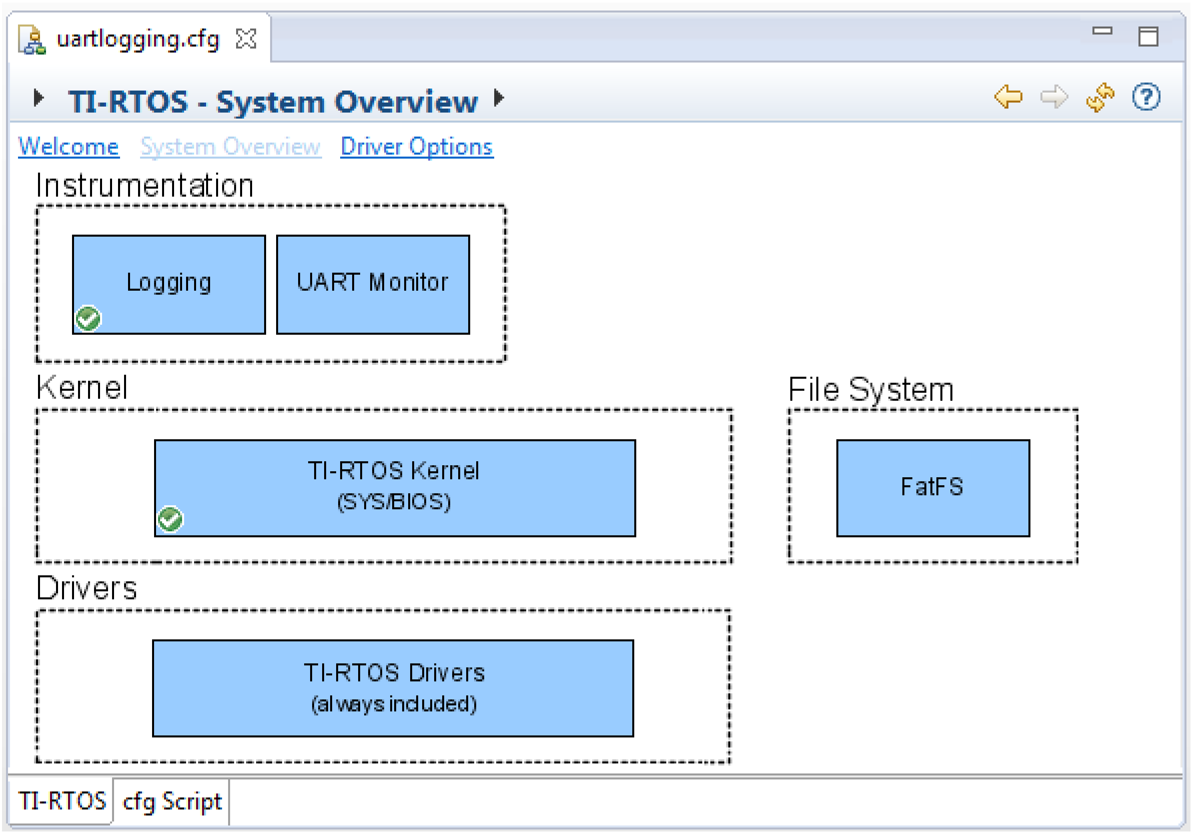
This section shows how to open XGCONF and view the System Overview. For details about how to use XGCONF, see Chapter 2 of the SYS/BIOS User's Guide (SPRUEX3).

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

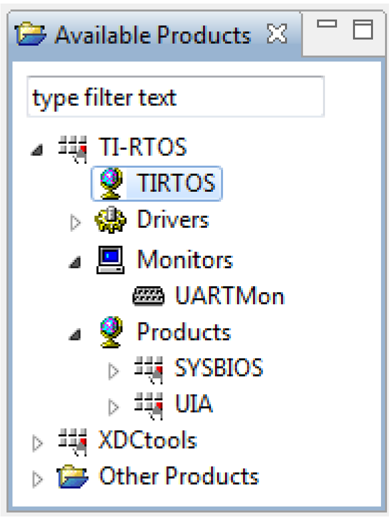
1. Make sure you are in the **C/C++** perspective of CCS. If you are not in that perspective, click the C/C++ icon to switch back.



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 which modules are being used by the application.



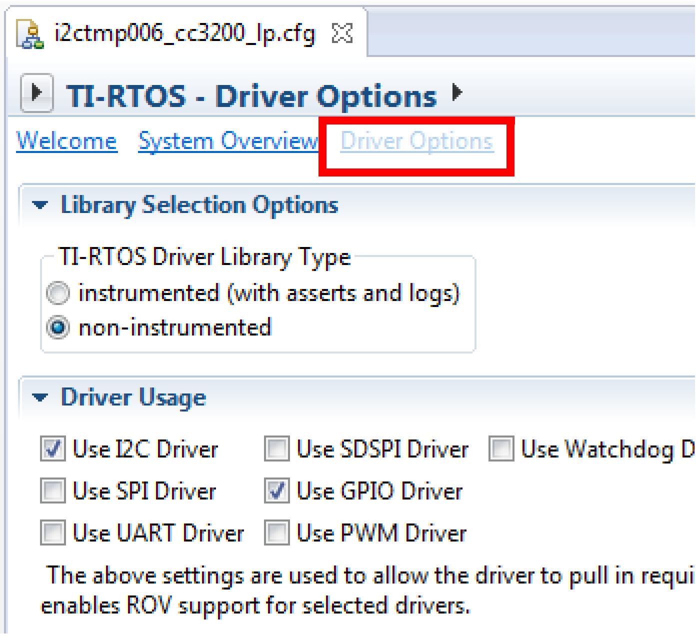
5. You also see a list of Available Products in a pane on the left of the CCS window. This list allows you to select the TIRTOS module and any configurable modules in the products TI-RTOS provides.
6. Click a blue box in the System Overview to go to the configuration page for a module.



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, select the **Driver Options** link.



TI-RTOS Driver Library Type
( ) instrumented (with asserts and logs)
(●) non-instrumented

[✓] Use I2C Driver [ ] Use SDSPI Driver [ ] Use Watchdog D
[ ] Use SPI Driver [✓] Use GPIO Driver
[ ] Use UART Driver [ ] Use PWM Driver

The above settings are used to allow the driver to pull in requi
enables ROV support for selected drivers.

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.

Check the boxes for any drivers your application will use. For most drivers, this is the only configuration necessary. Enabling the drivers here also enables the instrumentation provided to the RTOS Object View (ROV) plugins.

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 CC2650

Many of the common SYS/BIOS functions are provided in the ROM area of the 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 does not have any performance advantages over Flash; the main advantage is saving space in Flash.

To use the SYS/BIOS modules in the ROM, add the following 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;
```

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.

The TI Resource Explorer (in CCS) lets you access documentation for all the components of TI-RTOS. Choose the **Documentation Links** item for your version of TI-RTOS. Use the configuration reference documentation, and click the "Configuration settings" link at the top of each page to skip to the configuration information.

Index

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		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video & Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Mobile Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303 Dallas, Texas 75265
Copyright © 2015, Texas Instruments Incorporated