TI-RTOS 2.16 for C2000

Getting Started Guide

Literature Number: SPRUHU3D
February 2016

Contents

Preface

SPRUHU3D—February 2016

Read This First

About This Manual

This manual describes TI-RTOS for C2000. The version number as of the publication of this manual is v2.16.

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.

February 9, 2016

About TI-RTOS

This chapter provides an overview of TI-RTOS for C2000.

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 2.4).

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 C2000 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 Drivers and Board Support	Drivers and Mware	*TI-RTOS User's Guide* -- SPRUHD4
TI-RTOS Instrumentation	UIA	*System Analyzer User's Guide* -- SPRUH43
TI-RTOS Networking	NDK	*TI Network Developer's Kit (NDK) Guide* -- SPRU523 *TI Network Developer's Kit (NDK) API Reference* -- SPRU524
TI-RTOS File System	FatFS	*TI-RTOS User's Guide* -- SPRUHD4
TI-RTOS USB	USB stack	*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 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).
- **TI-RTOS Networking — NDK.** The Network Developer's Kit (NDK) is a platform for development and demonstration of network enabled applications on TI embedded processors.
- **MWare.** The M3 portion of controlSUITE. It includes low level drivers and examples.[1]
- **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.

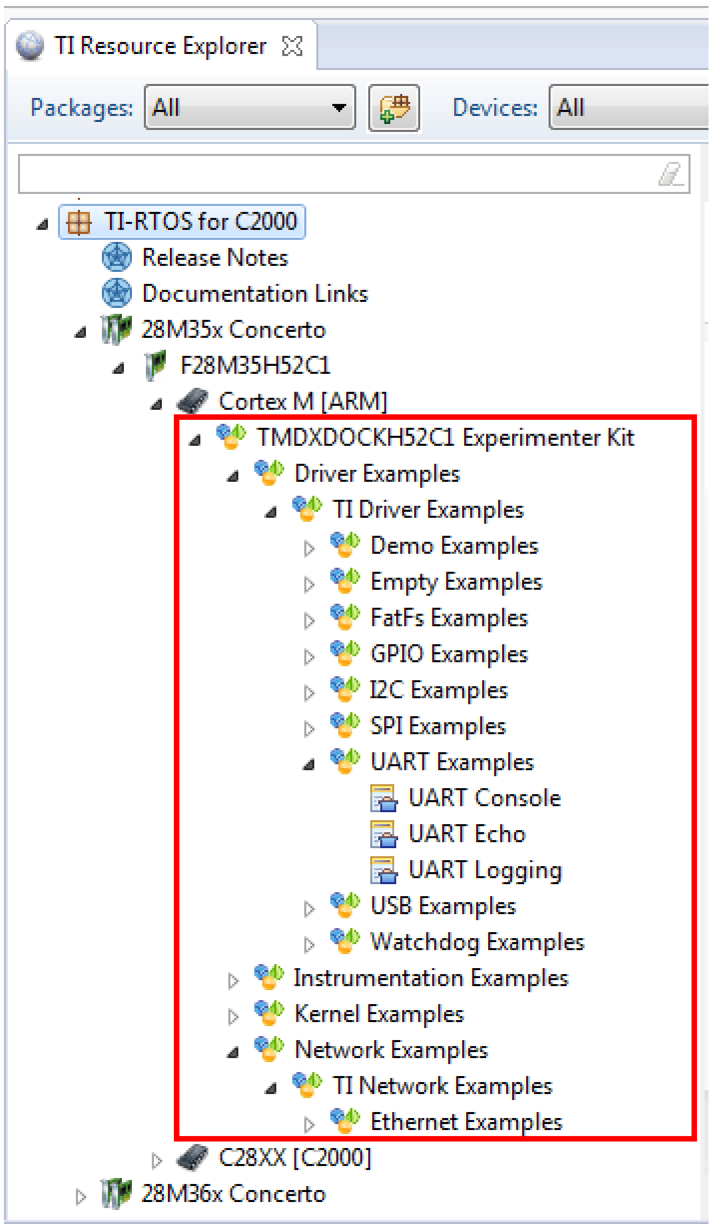
1. The MWare libraries distributed with TI-RTOS have been rebuilt with the following compiler option: --define=USE_RTOS. See the TI-RTOS.README file in the top-level folders of the MWare component of TI-RTOS for details.

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.
- **Network Examples** show how to use capabilities such as WiFi, PPP, Ethernet, and HTTP/HTTPS.

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) version is supported. The version listed is recommended because it was used to build the TI-RTOS libraries and to perform testing. More recent versions are expected to be compatible.

- **Texas Instruments:** ARM CodeGen Tools v5.2.4 (for M3)
- **Texas Instruments:** TMS320C28x CodeGen Tools v6.2.5

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

- ti.targets.arm.elf.M3 (TI-RTOS drivers are provided only on the M3)
- ti.targets.C28_float
- ti.targets.C28_large

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
Concerto (ARM M3 + DSP 28x)	F28M35H52C1	TMDXDOCKH52C1 Experimenter Kit
Concerto (ARM M3 + DSP 28x)	F28M36P63C2	TMDXDOCK28M36 Experimenter Kit

Both M3 and 28x sides of Concerto boards are supported.

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 MWare.

The TI-RTOS installation installed drivers for multiple device families. The following list indicates which drivers can be use with C2000 targets:

- **EMAC.** Ethernet driver used by the networking stack (NDK) and not intended to be called directly.
- **GPIO.** API set intended to be used directly by the application or middleware to manage the GPIO interrupts, pins, and ports.
- **I^{2}C.** API set intended to be used directly by the application or middleware.
- **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.
- **USBMSCHFatFs.** USB MSC Host under FatFS (for Flash drives). This driver is used by FatFS and is not intended to be called directly by the application.

- **Other USB functionality.** See the USB examples for reference modules that provide support for the Human Interface Device (HID) class (mouse and keyboard) and Communications Device Class (CDC). This code is provided as part of the examples, not as a separate driver.
- **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_c2000_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

- **NDK**
  - — NDK User's Guide (SPRU523)
  - — NDK Programmer's Reference Guide (SPRU524)
  - — NDK API reference. Run <*tirtos_install*>/products/ndk_#_##_##_##/docs/doxygen/html/index.html.
  - — NDK configuration reference. In the TI-RTOS Release Notes, follow the **Documentation Overview** link. In the Documentation Overview page, choose the **TI-RTOS Networking Configuration (cdoc)** item.
  - — NDK on the Texas Instruments 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
- **MWare and controlSUITE**
  - — Documents in <*tirtos_install*>/products/MWare_##_##_##_##/doc
  - — controlSUITE on the Texas Instruments Wiki
  - — controlSUITE Product Folder
- **FatFS API**
  - — Open source documentation
  - — TI-RTOS User's Guide (SPRUHD4)
- **General microcontroller information**
  - — Microcontrollers forum on TI's E2E Community
- **Concerto boards and devices**
  - — Concerto F28M35x Technical Reference Manual
  - — Concerto F28M36x Technical Reference Manual
  - — C2000 on the Texas Instruments Wiki
  - — Concerto on the Texas Instruments Wiki
  - — Concerto Product Folder
  - — H52C1 Concerto Experimenter Kit
  - — F28M35H52C Concerto Microcontroller datasheets
  - — H63C2 Concerto Experimenter Kit
  - — F28M36P63C2 Concerto Microcontroller datasheets
- **I^2C**
  - — Specification

Installing TI-RTOS

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

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. 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.

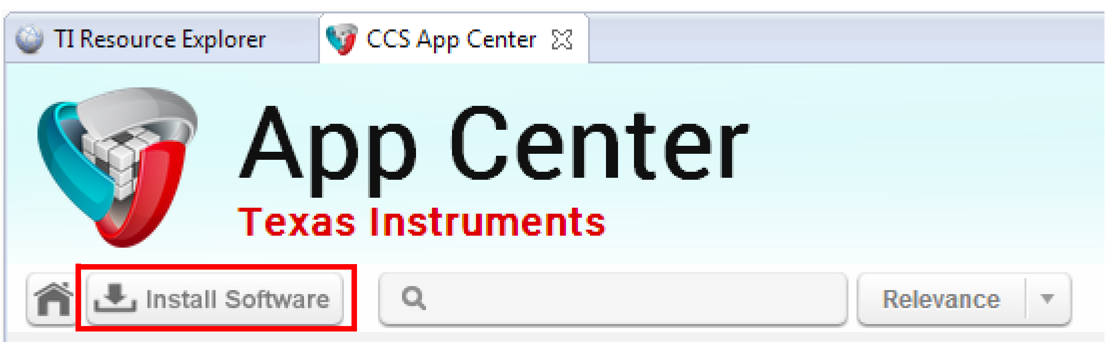
Run the installer, and answer the prompts as appropriate.

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 or as a standalone product as described in Section 2.4.

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 as a Standalone Product

If you do not use Code Composer Studio, you can install TI-RTOS as a standalone product.

1. Download the Windows or Linux installer for TI-RTOS for the device family you use. For example, `tirtos_setupwin32_c2000_2_##_##_##.exe` or `tirtos_setuplinux_c2000_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_c2000_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.

Chapter 3

SPRUHU3C—February 2016

Examples for TI-RTOS

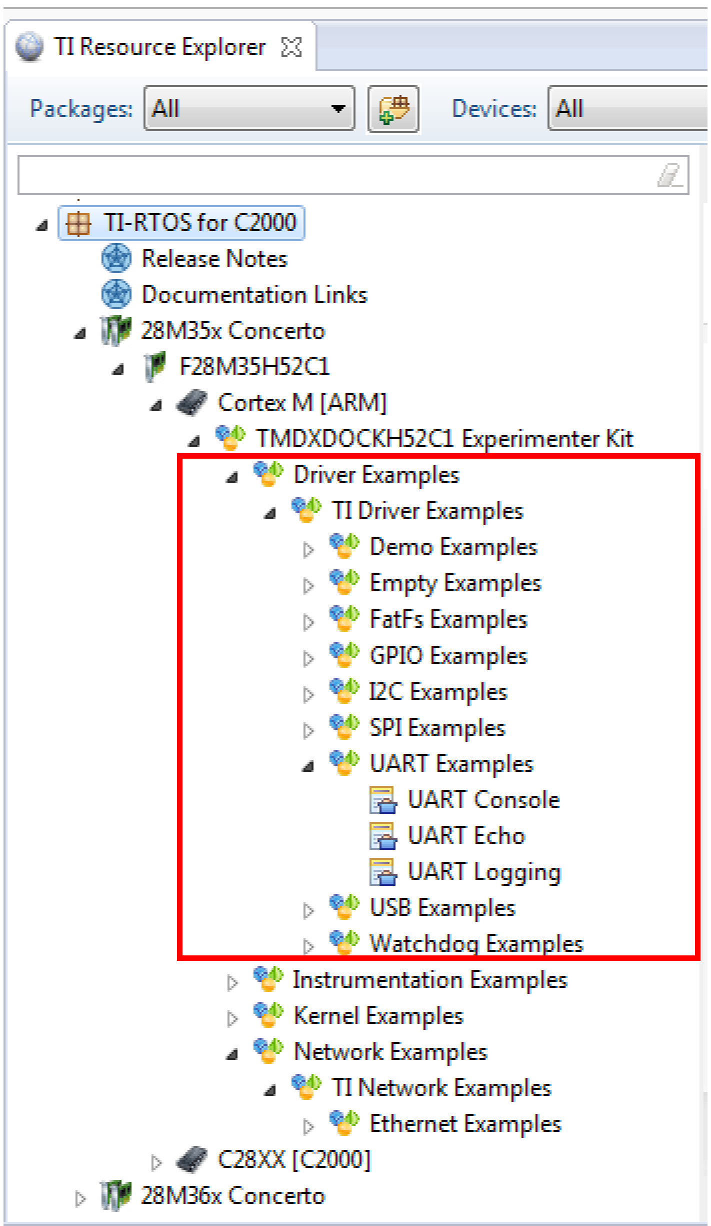
TI-RTOS comes with a number of examples that illustrate on how to use the individual components. This chapter explains how to create and use these examples.

Topic **Page**

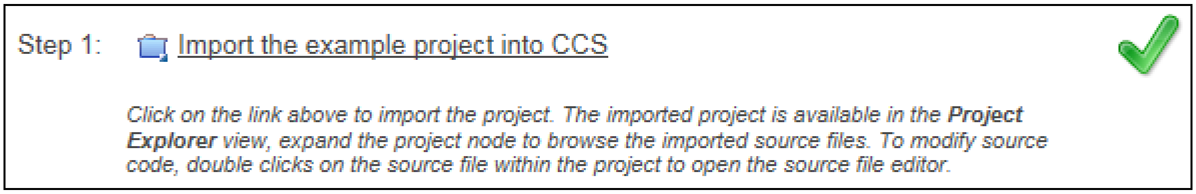
3.1 Creating Example Projects Using the Resource Explorer in CCS

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:

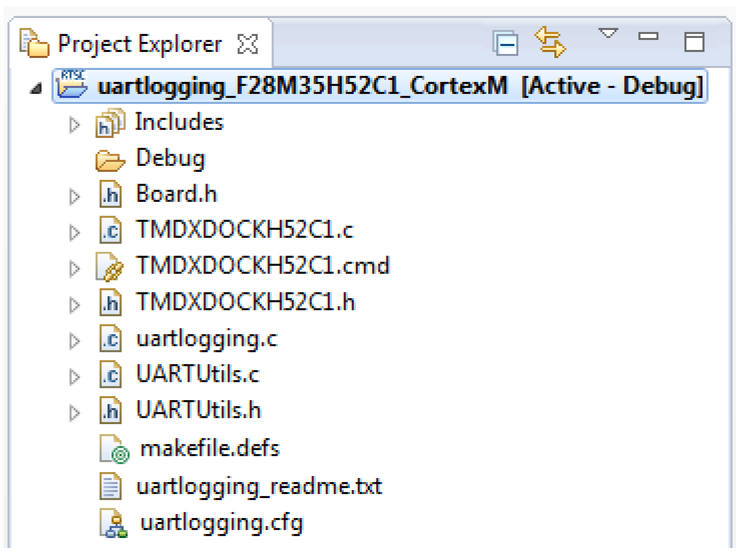
1. Open CCS. If you do not see the Resource Explorer, make sure you are in the **CCS Edit** perspective and choose **View > Resource Explorer (Examples)** from the menus.
2. Type the name or part of the name of your device in the **enter search keyword** field to hide all the examples that don't apply 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. **Network Examples** use capabilities such as WiFi, PPP, Ethernet, and HTTP/HTTPS.



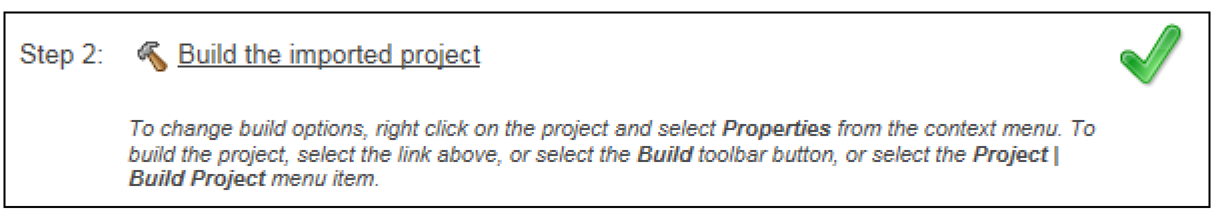
4. Select the example you want to create. A description of the selected example is shown to the right of the example list.
5. Click the **Step 1** link in the right pane of the Resource Explorer to **Import the example project into CCS**. This adds a new project to your Project Explorer view. Once you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



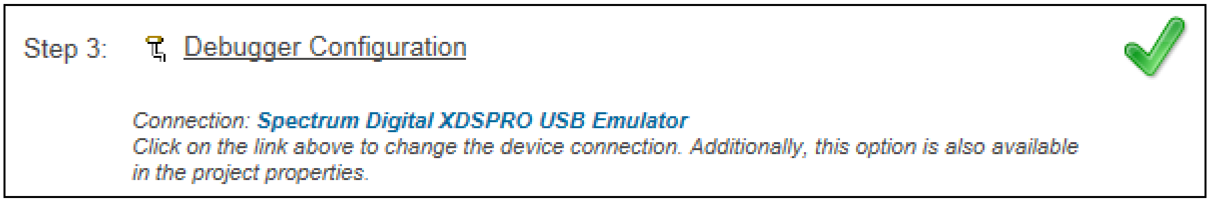
6. The project created will have a name with the format <*example_name*>_<*board*>. You can expand the project to see the source code, configuration, and other files in the project.



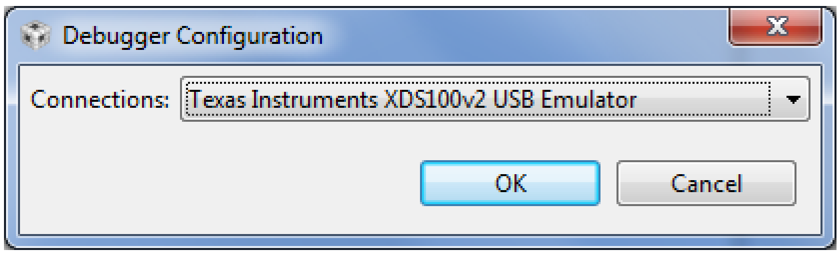
7. The page shown when you select an example in the Resource Explorer provides additional links to perform common actions with that example. Use the **Step 2** link when you are ready to build the project. If you want to change any build options, right click on the project and select **Properties** from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



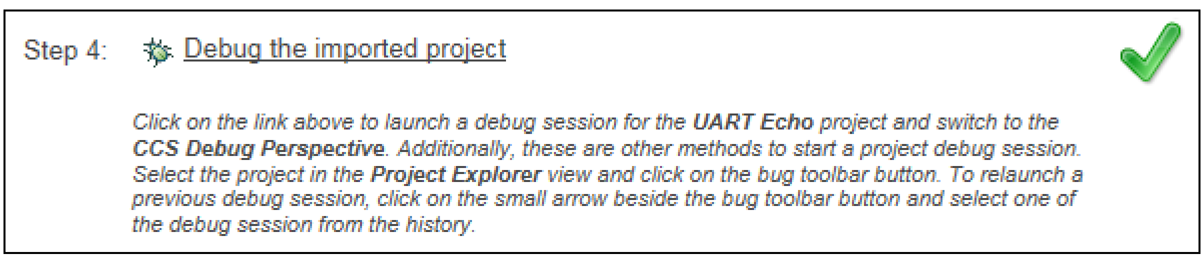
8. Use the **Step 3** link to change the connection used to communicate with the board. The current setting is shown in the Resource Explorer page for the selected example. (If you want to use a simulator instead of a hardware connection, double-click the *.ccxml file in the targetConfigs folder of the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



9. You will see the Debugger Configuration dialog. Choose an emulator. For the F28M3x devices, choose the **Texas Instruments XDS 100v2 USB Emulator**.



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



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 TI. 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), this step is not necessary because pre-generated examples are included as part of the installation for all supported boards. The provided examples are located in the `TIRTOS_INSTALL_DIR\tirtos_c2000_2_##_##_##_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. (IAR is not supported for this device family.)
   - GCC_INSTALLATION_DIR: Full path to the GCC code generation tools installation. (GCC is not supported for this device family.)
   - TIRTOS_INSTALL_DIR: Full path to the TI-RTOS installation.
   - XDCTOOLS_INSTALL_DIR: Full path to the XDCtools installation.
2. 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 a `tirtos_c2000_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 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 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:

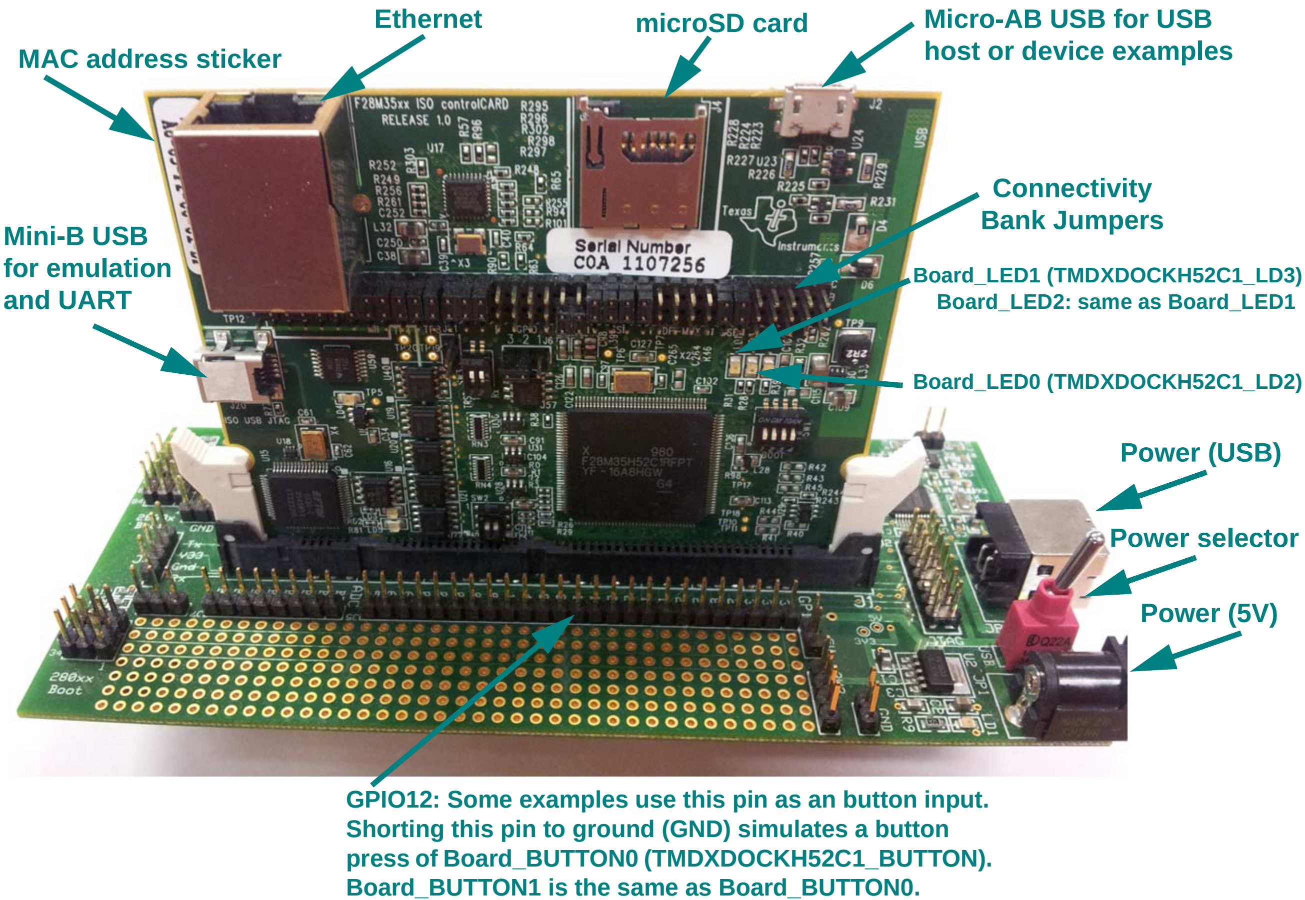
- Most TI-RTOS driver examples use the SysMin System Support module. See the readme files in the individual example projects for details.
- The empty, demo, and most UART examples use the ti.uia.sysbios.LoggingSetup module with stop mode data collection. The UART Console example uses run-time data collection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

3.3 Concerto TMDXDOCKH52C1 Settings and Resources

The TMDXDOCKH52C1 Experimenter Kit has a Concerto F28M35H52C1 device.

Concerto TMDXDOCKH52C1 connections:



The USB connector for devices can accept a Micro-B cable for USB Device examples and a Micro-A dongle for USB Host examples.

Jumper settings:

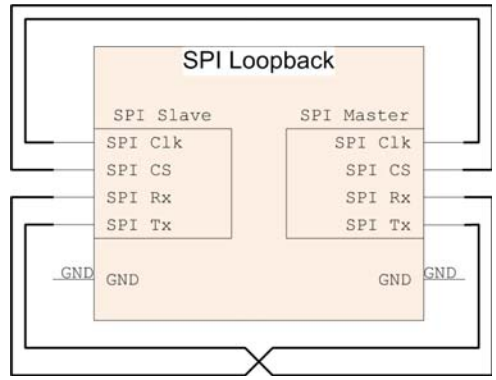
- Connectivity Bank 01-15: B-C position (Ethernet)
- Connectivity Bank 20-21: B-C position (I2C EEPROM)
  — J6: 2-3 position (I2C EEPROM Write protection off)
- Connectivity Bank 22-25: B-C position (SPI/SSI SD Card slot)
- Connectivity Bank 30-31: B-C position (USB Host and Device)
  — A board modification is required for the USB host examples (see Section 3.3.2)

Switch settings:

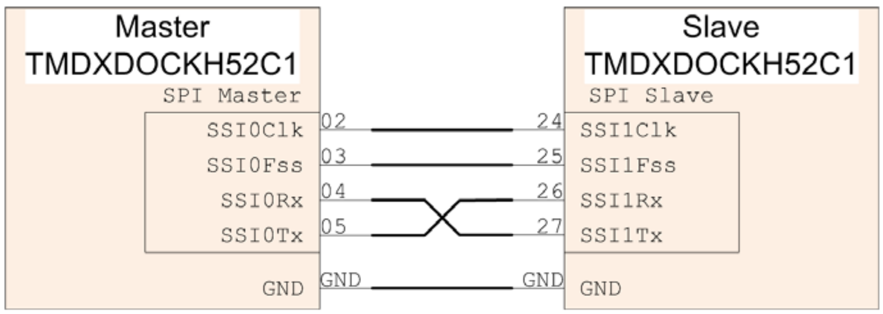
- SW1: Open all 4 switches by bringing them into the "down" position. This allows the M3 (master subsystem) to boot out of Flash memory.
- GPIO12: Some examples use this pin as an input. When shorting this pin to ground (GND), it will simulate a button press.

SPI Loopback example pin connections: When wiring for SPI loopback, GPIO pins on a single board are wired to other GPIO pins on the same board.

Master SPI Pin (Function)	Slave SPI Pin (Function)
GPIO02 (SPI0CLK) →	GPIO24 (SPI1CLK)
GPIO03 (SPI0FSS) →	GPIO25 (SPI1FSS)
GPIO04 (SPI0RX) →	GPIO27 (SPI1TX)
GPIO05 (SPI0TX) →	GPIO26 (SPI1RX)



SPI IPC Master/Slave example pin connections: When wiring for the IPC SPI Master/Slave example, two boards are used:



Resources Used:

The following list shows which TMDXDOCKH52C1 resources are used by TI-RTOS examples that make use of a particular component or peripheral.

- **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.
- **IPC.** Uses the IPC registers including their associated interrupts.
- **TI-RTOS.**
  - **Ethernet.** Uses the EMAC driver and its associated interrupts with NDK to support networking.
  - **SD Card.** Uses FatFs and the SDSPI driver on SSI0 without interrupts to read and write to files on an SD Card.
  - **EEPROM.** Uses the I^{2}C driver on I2C0 with its associated interrupts to read and write to the onboard EEPROM.
  - **GPIOs.** The GPIO driver is used on 2 onboard LEDs: LD2 (PC6_GPIO70) and LD3 (PC7_GPIO71) as output pins and one pin as an input pin PB4_GPIO12.
  - **Serial.** The UART driver uses UART0, which is attached to the FTDI USB chip for serial communications.
  - **SPI.** The SPI driver uses SPI0 for Board SPI0 and SPI1 for Board SPI1.
  - **USB.** The USB reference examples use the USB library and the USB controller with its associated interrupts.
  - **Watchdog.** The Watchdog driver example uses Watchdog Timer 0 and its associated interrupt.

3.3.1 Setting the MAC Address

If you are using the NDK and the EMAC peripheral, you will need to edit the MAC address in the board-specific C file (for example, TMDXDOCKH52C1.c) in the examples. Modify the following definition in the file to match the MAC address printed on your board.

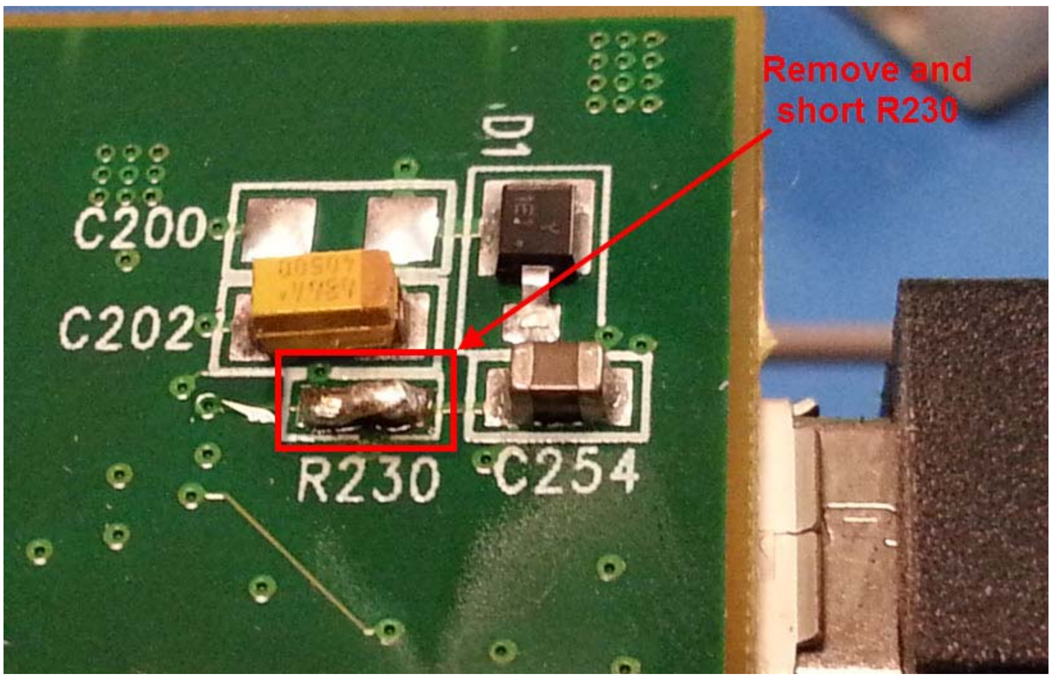
```
/*
 *  EMAC configuration structure
 *  Set user/company specific MAC octates. The following sets the address
 *  to ff-ff-ff-ff-ff-ff. Users need to change this to make the label on
 *  their boards.
 */
UInt8 macAddress[6] = {0xff, 0xff, 0xff, 0xff, 0xff, 0xff};
```

For example, the following would set the MAC address to A8-63-F2-00-05-1A:

```
UInt8 macAddress[6] = {0xA8, 0x63, 0xF2, 0x00, 0x05, 0x1A};
```

3.3.2 USB Host Mode Board Modification

Using the USB controller in host mode on the TMDXDOCKH52C1 requires a hardware modification to the control card. This modification is not required, but can be performed without causing problems, when using the USB controller in device mode.



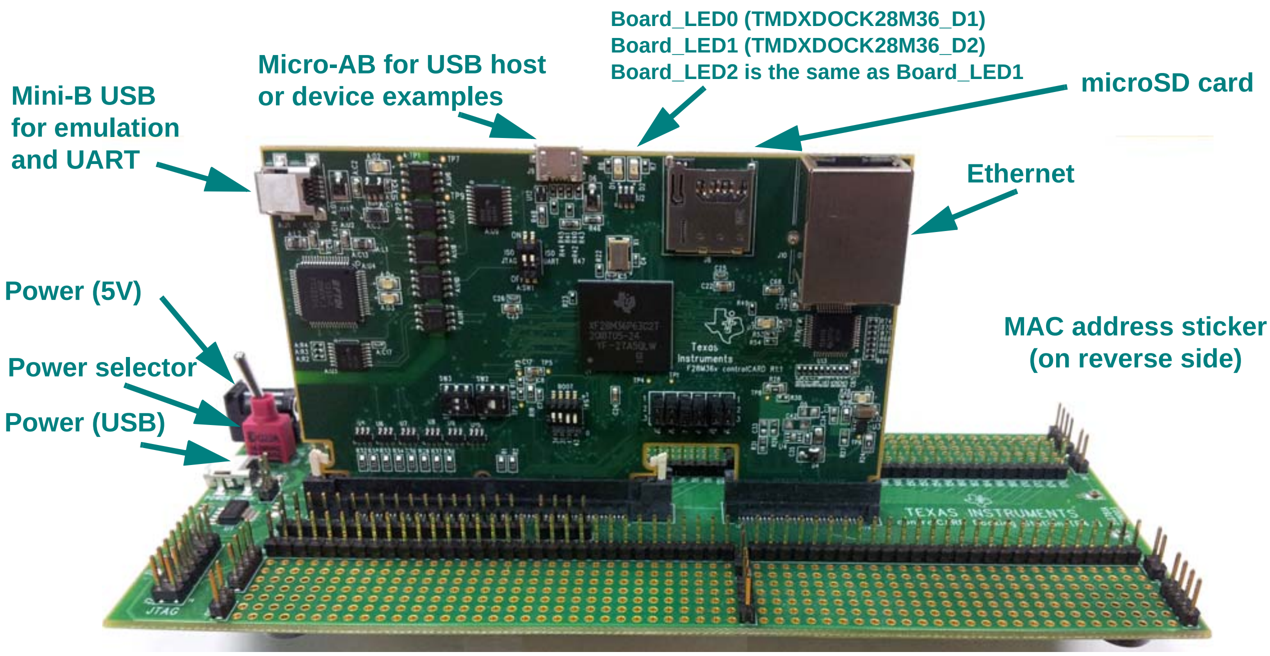
Remove and short resistor R230 on the control card.

This modification allows the USB_VBUS pin to correctly detect the VBUS voltage level; preventing a false VBUS_ERR from being generated by the USB controller.

3.4 Concerto TMDXDOCK28M36 Settings and Resources

The TMDXDOCK28M36 Experimenter Kit has a Concerto F28M36P63C2 device.

Concerto TMDXDOCK28M36 connections:



The USB connector for devices can accept a Micro-B cable for USB Device examples and a Micro-A dongle for USB Host examples.

Jumper settings:

- J2-J7: 1-2 position (USB Host and Device)

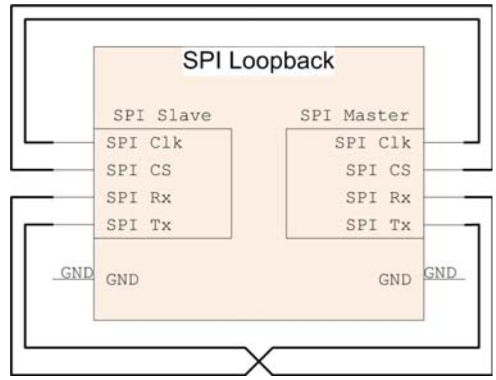
Switch settings:

- A:SWI: Both switches should be in the ON position
- SW1: Place all switches in the 1 (up) position to allow the M3 to boot out of Flash memory.
- GPIO 58: Some examples use pin 58 as it is labeled on the docking station as an input. Shorting this pin to ground (GND) will simulate a button press.

SPI Loopback example pin connections:

When wiring for SPI loopback, pins on a single board are wired to other pins on the same board.

Master SPI Pin (Function)	Slave SPI Pin (Function)
71 (SPI0CLK) →	75 (SPI1CLK)
73 (SPI0FSS) →	77 (SPI1FSS)
69 (SPI0RX) →	81 (SPI1TX)
67 (SPI0TX) →	79 (SPI1RX)



Resources Used:

The following list shows which TMDXDOCK28M36 resources are used by TI-RTOS examples that make use of a particular component or peripheral.

- **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.
- **IPC.** Uses the IPC registers including their associated interrupts.
- **TI-RTOS.**
  - **Ethernet.** Uses the EMAC driver and its associated interrupts with the NDK for networking.
  - **SD Card.** Uses FatFs and the SDSPI driver on SSI3 without interrupts to read and write to files.
  - **GPIOs.** The GPIO driver is used on 2 onboard LEDs: D1 (PE7_GPIO31) and D2 (PF2_GPIO34) as output pins and one pin as an input pin PB4_GPIO12, pin 58 on the docking station.
  - **Serial.** The UART driver uses UART0, which is attached to the FTDI USB chip.
  - **SPI.** The SPI driver uses SPI0 for Board SPI0 and SPI1 for Board SPI1.
  - **USB.** The USB reference examples use the USB library and the USB controller with its associated interrupts.
  - **Watchdog.** The Watchdog driver example uses Watchdog Timer 0 and its associated interrupt.

3.4.1 Setting the MAC Address

If you are using the NDK and the EMAC peripheral, you will need to edit the MAC address in the board-specific C file (for example, TMDXDOCK28M36.c) in the examples. See Section 3.3.1 for details.

3.5 Installing USB Drivers for the USB Device Examples

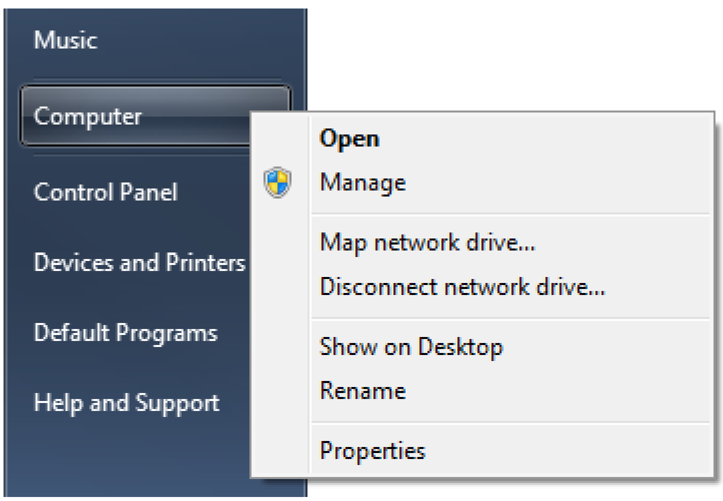
The USB examples build upon the examples provided with MWare. Because the examples mimic the same functionality, you can use the same drivers delivered with standalone installations of controlSUITE (MWare).

Windows USB drivers for MWare are located in the `<tirtos_install>/products/MWare_v20##/MWare/windows_drivers` directory.

The Windows menus and dialogs you see may be slightly different from those shown here, depending on your version of Windows.

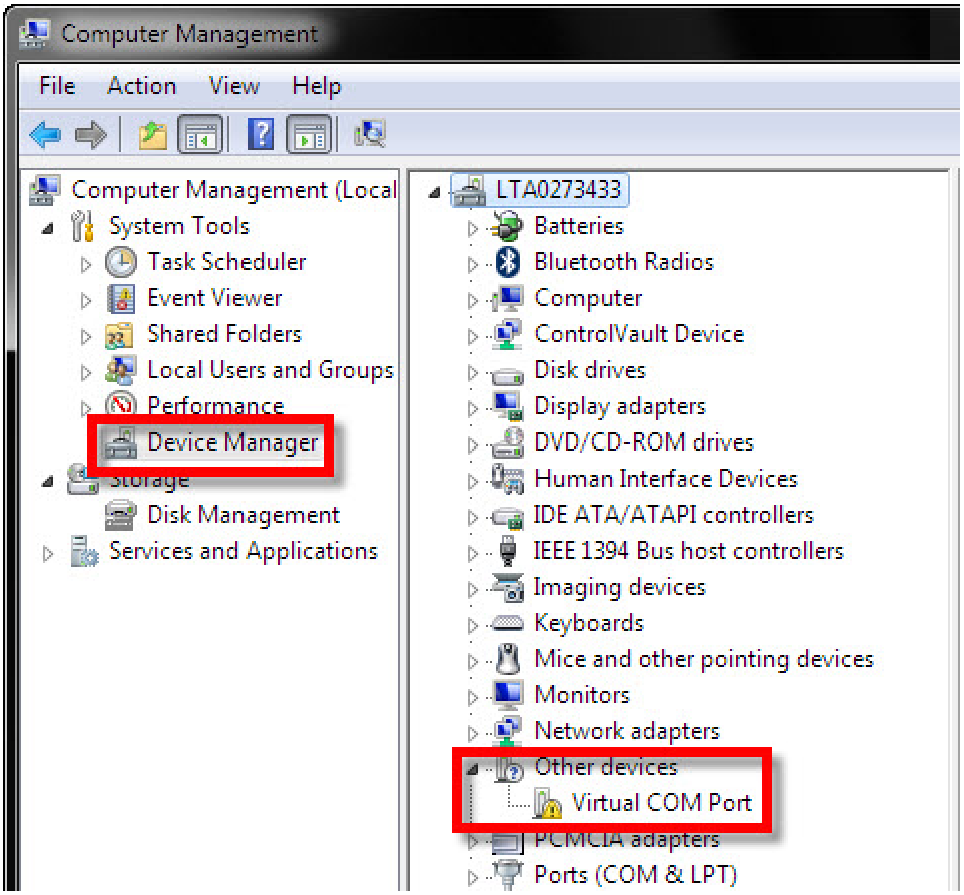
To install the USB driver, follow these steps:

1. Load and run a USB device reference example—USB Keyboard Device, USB Mouse Device, or USB CDC Mouse Device.
2. While the example is running, connect the device to the Windows PC via a USB cable. At this point, Windows will detect the device and attempt to enumerate it.
3. Open the Windows Device Manager by right-clicking on the **My Computer** desktop icon or **Computer** in the Start Menu and selecting **Manage**.



4. If you are prompted by a security warning in Windows 7, click **Yes**.

5. Select the **Device Manager** category in the left pane.
6. In the center pane, select the unknown driver that you are trying to install. For example, the device shown here is for the USB CDC driver.



7. Right-click on the device, and select **Update Driver Software**.
8. Select **Browse my computer for driver software** and browse to the location of the Windows USB drivers <tirtos_install>/products/MWare_v20##/MWare/windows_drivers. Make sure the box to **Include subfolders** is checked.
9. Click **Next** to run the installation wizard. If you see a Window Security prompt, click **Install**.
10. After the driver is installed, you can determine the COM port number for the CDC (Virtual COM Port) device.

Chapter 4

SPRUHU3D—February 2016

Configuring TI-RTOS

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

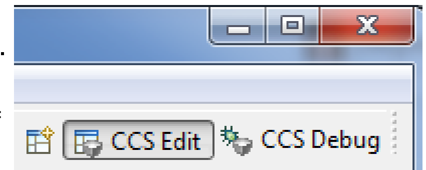
Topic **Page**

4.1 Starting the Configuration Tool

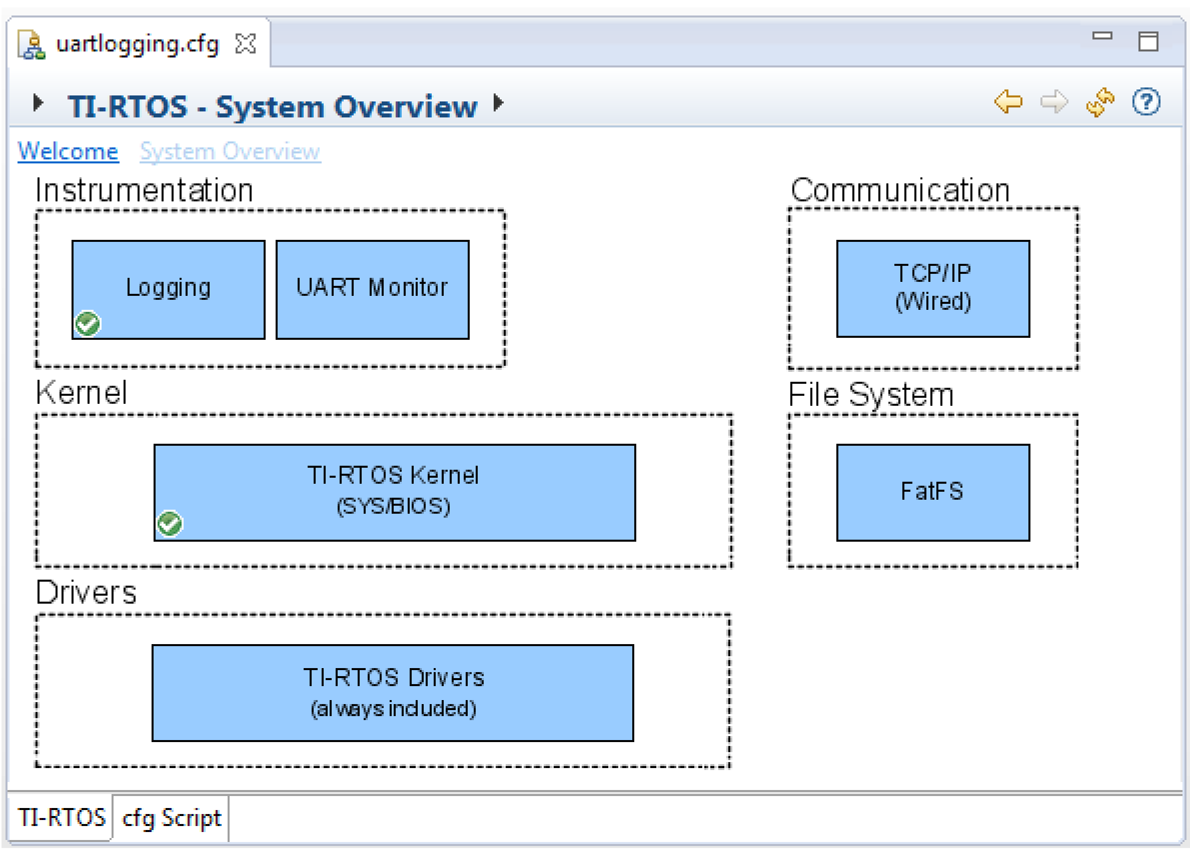
This section shows how to open the Graphical Configuration Tool (XGCONF) to view the System Overview. For details on using XGCONF, see Chapter 2 of the SYS/BIOS User's Guide (SPRUEX3).

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

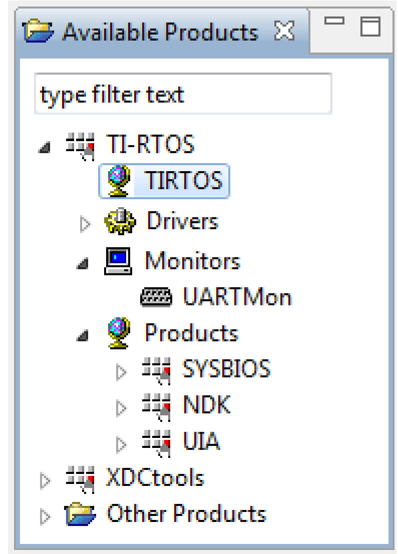
1. Make sure you are in the **CCS Edit** perspective of CCS. If you are not in that perspective, click the CCS Edit icon to switch back.



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.



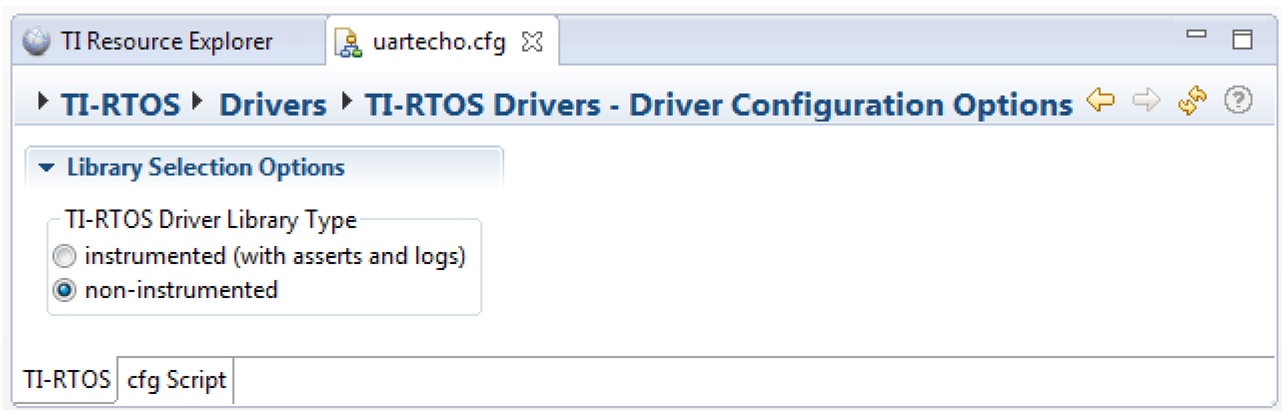
5. You also see a list of Available Products in a pane on the left of the CCS window. This list allows you to select the TIRTOS module and any configurable modules in the products TI-RTOS provides.
6. Click a blue box in the System Overview to go to the configuration page for a module.



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, 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.

4.3 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

P

R

S

T

U

W

X