

TI MCU Day - Internet of Things 2013

STUDENT GUIDE



Important Notice

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgment, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Customers are responsible for their applications using TI components.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. 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 of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty or endorsement thereof.

Copyright © 2013 Texas Instruments Incorporated

Revision History

August 2013 – Revision 1.0

September 2013 – Revision 1.1 (edited lab instructions)

September 2013 – Revision 1.2 (rearranged the order, new CC3000 lab, TI-RTOS demo)

Mailing Address

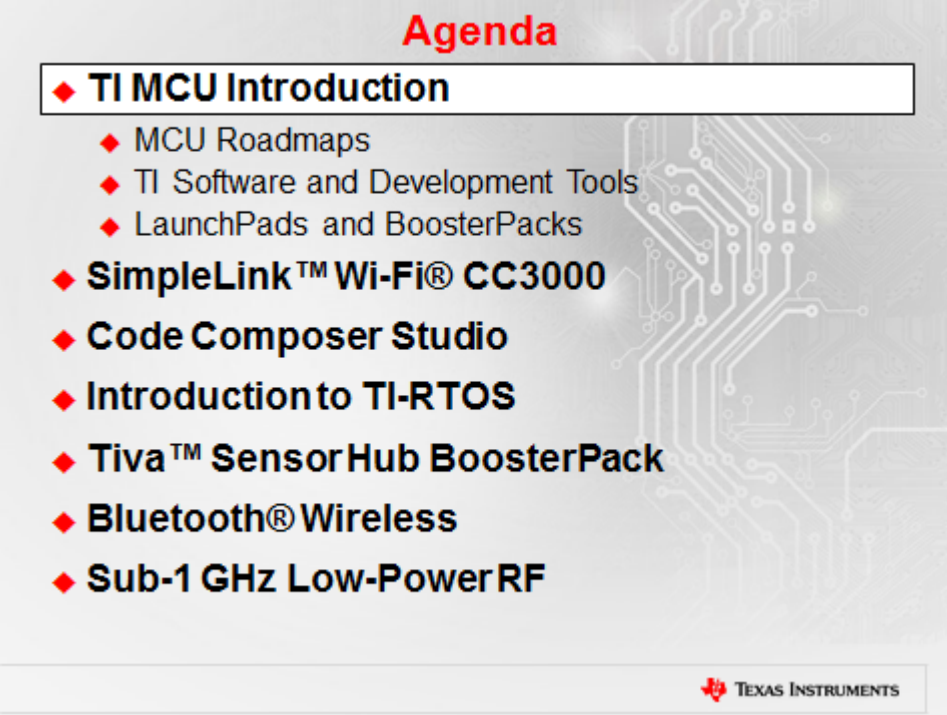
Texas Instruments
Training Technical Organization
6500 Chase Oaks Blvd Building 2
M/S 8437
Plano, Texas 75023

Workshop Objectives

Workshop Objectives


- ◆ Welcome to TI MCU Day – Internet of Things 2013
- ◆ The objectives of this training are to:
 - ◆ Become familiar with current TI MCU roadmaps
 - ◆ Install an MSP430 or Tiva LaunchPad board, and run the out-of-box application
 - ◆ Run a WiFi application based on CC3000 BoosterPack and Tiva LaunchPad
 - ◆ Build, run, and debug a Blink LED project using Code Composer Studio
 - ◆ Understand the benefits of TI-RTOS
 - ◆ Demo the SensorHub BoosterPack, use the SensorHub library
 - ◆ Learn about Bluetooth/BLE, demo SensorTag
 - ◆ Run a Sub 1-GHz Low-Power Wireless application

Workshop Agenda



Agenda

- ◆ **TI MCU Introduction**
 - ◆ MCU Roadmaps
 - ◆ TI Software and Development Tools
 - ◆ LaunchPads and BoosterPacks
- ◆ **SimpleLink™ Wi-Fi® CC3000**
- ◆ **Code Composer Studio**
- ◆ **Introduction to TI-RTOS**
- ◆ **Tiva™ SensorHub BoosterPack**
- ◆ **Bluetooth® Wireless**
- ◆ **Sub-1 GHz Low-Power RF**

 **TEXAS INSTRUMENTS**

TI MCU Introduction

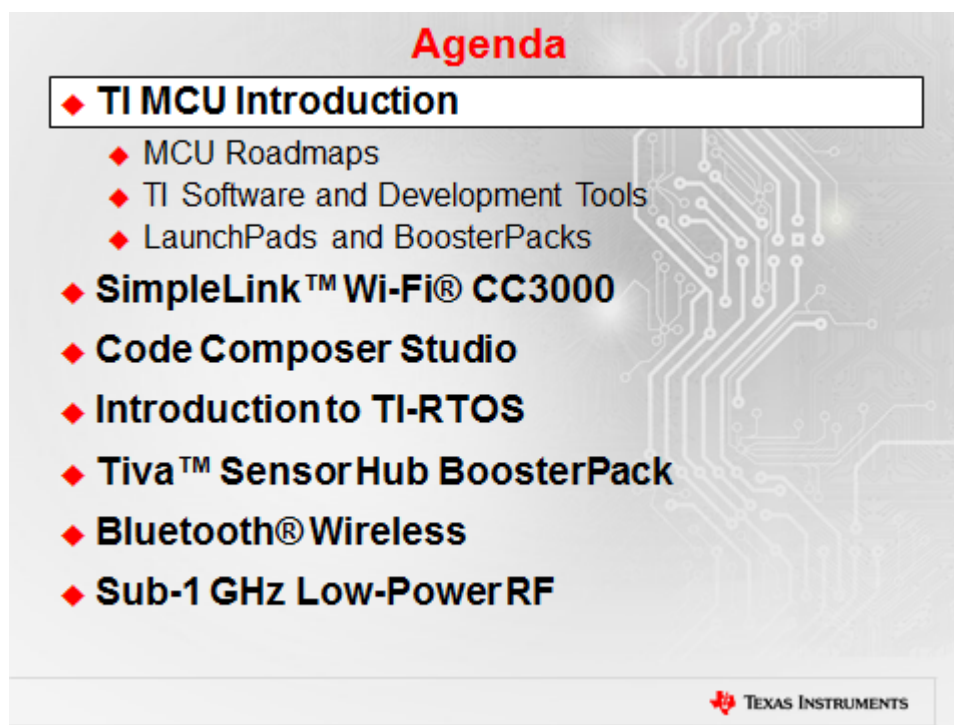
Introduction

This chapter is an introduction to the broad range of TI MCUs which include four families: MSP430™ Ultra-Low Power 16-bit MCUS, C2000™ 32-bit Real-Time Control MCUs, Tiva™ C Series ARM® Cortex™-M4 MCUs, and Hercules™ Safety ARM MCUs.

The TI LaunchPad MCU Evaluation Kits are also introduced, along with some of the companion BoosterPack modules.

The first lab is to run the user experience application on the new MSP430F5529 LaunchPad. The second lab is to run the quickstart application on the new Tiva C Series LaunchPad.

Learning Objectives



Agenda

- ◆ **TI MCU Introduction**
 - ◆ MCU Roadmaps
 - ◆ TI Software and Development Tools
 - ◆ LaunchPads and BoosterPacks
- ◆ **SimpleLink™ Wi-Fi® CC3000**
- ◆ **Code Composer Studio**
- ◆ **Introduction to TI-RTOS**
- ◆ **Tiva™ SensorHub BoosterPack**
- ◆ **Bluetooth® Wireless**
- ◆ **Sub-1 GHz Low-Power RF**

TEXAS INSTRUMENTS

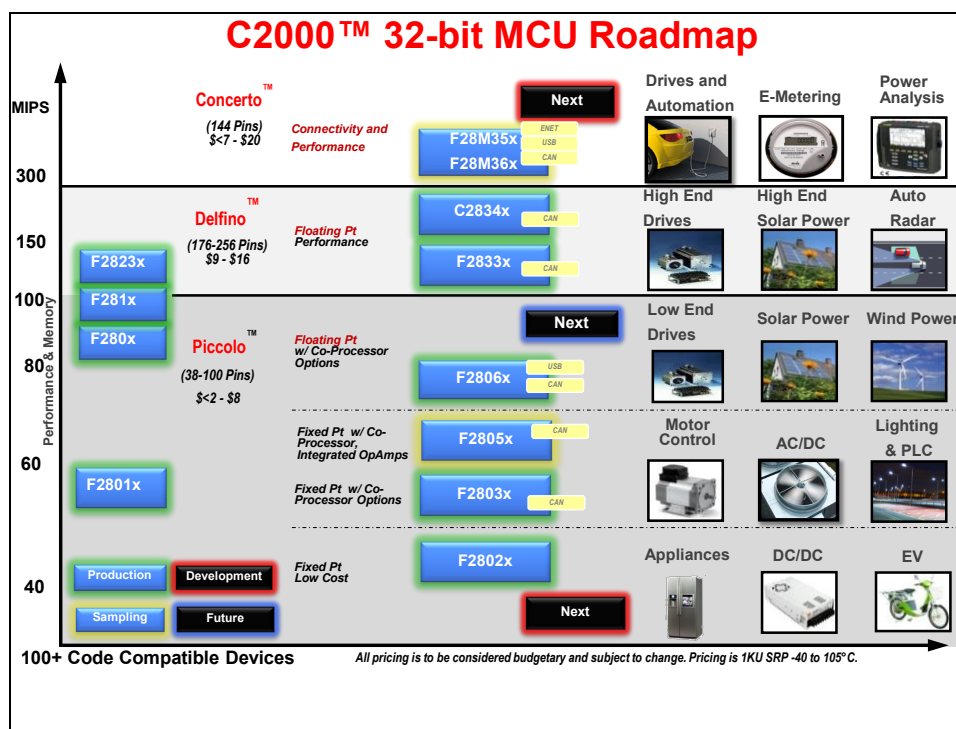
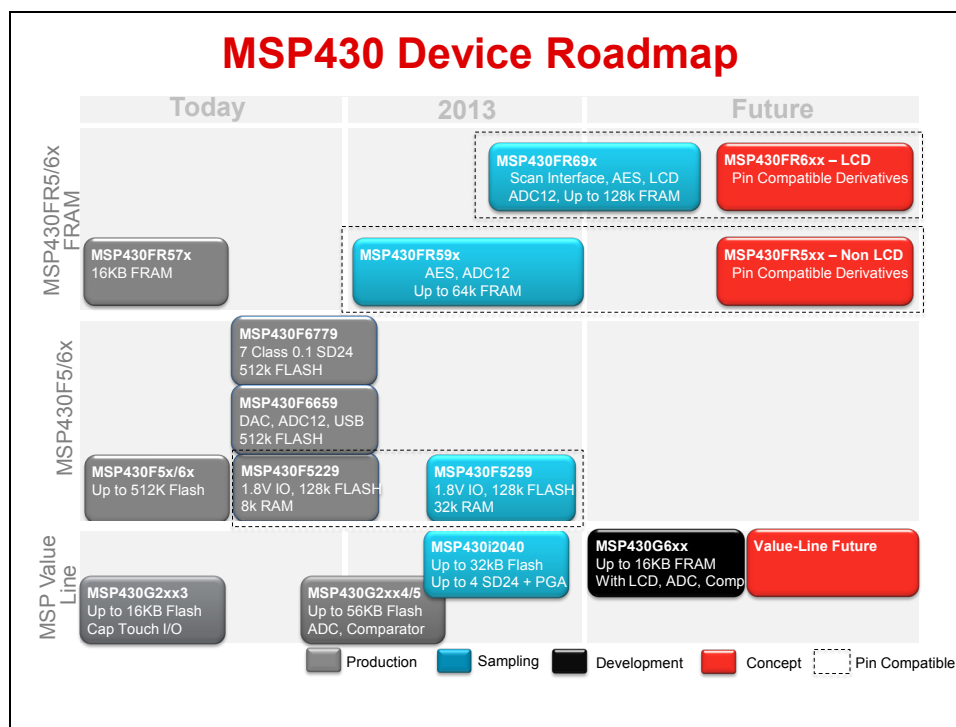
Chapter Topics

TI MCU Introduction.....	1-1
<i>Texas Instruments Embedded Processor Portfolio</i>	<i>1-3</i>
<i>MCU Roadmaps.....</i>	<i>1-4</i>
<i>TI Software and Development Tools.....</i>	<i>1-8</i>
<i>LaunchPads and BoosterPacks.....</i>	<i>1-13</i>
<i>Lab 1(a) – MSP4305529 LaunchPad User Experience.....</i>	<i>1-15</i>
<i>Lab 1(b) – Tiva C Series LaunchPad QuickStart.....</i>	<i>1-17</i>

Texas Instruments Embedded Processor Portfolio

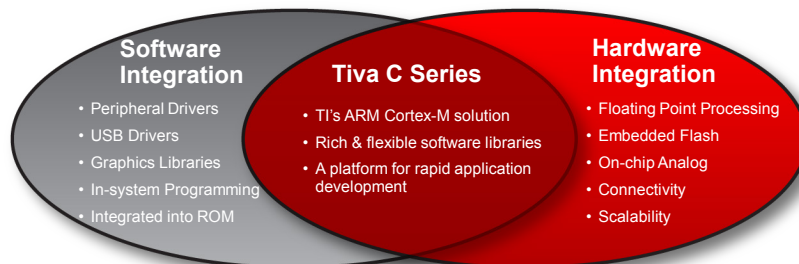
Microcontrollers (MCU)				Application (MPU)		
MSP430	C2000	Tiva	Hercules	Sitara	DSP	Multicore
16-bit Ultra Low Power & Cost	32-bit Real-time	32-bit All-around MCU	32-bit Safety	32-bit Linux Android	16/32-bit All-around DSP	32-bit Massive Performance
MSP430 ULP RISC MCU	Real-time C28x MCU • ARM M3+C28	ARM Cortex-M3 Cortex-M4F	ARM Cortex-M3 Cortex-R4	ARM Cortex-A8 Cortex-A9	DSP C5000 C6000	C66 + C66 A15 + C66 A8 + C64 ARM9 + C674
<ul style="list-style-type: none"> Low Pwr Mode = 0.1 μA = 0.5 μA (RTC) Analog I/F RF430 	<ul style="list-style-type: none"> Motor Control Digital Power Precision Timers/PWM 	<ul style="list-style-type: none"> 32-bit Float Nested Vector IntCtrl (NVIC) Ethernet (MAC+PHY) 	<ul style="list-style-type: none"> Lock step Dual-core R4 ECC Memory SIL3 Certified 	<ul style="list-style-type: none"> \$5 Linux CPU 3D Graphics PRU-ICSS industrial subsys 	<ul style="list-style-type: none"> C5000 Low Power DSP 32-bit fix/float C6000 DSP 	<ul style="list-style-type: none"> Fix or Float Up to 12 cores 4 A15 + 8 C66x DSP MMAC's: 352,000
TI-RTOS	TI-RTOS	TI-RTOS	Several RTOSs	Linux, Android, SYS/BIOS	C5x: DSP/BIOS C6x: SYS/BIOS	Linux SYS/BIOS
Flash: 512K FRAM: 64K	512K Flash	512K Flash	256K to 3M Flash	L1: 32K x 2 L2: 256K	L1: 32K x 2 L2: 256K	L1: 32K x 2 L2: 1M + 4M
25 MHz	300 MHz	80 MHz	220 MHz	1.35 GHz	800 MHz	1.4 GHz
\$0.25 to \$9.00	\$1.85 to \$20.00	\$1.00 to \$8.00	\$5.00 to \$30.00	\$5.00 to \$25.00	\$2.00 to \$25.00	\$30.00 to \$225.00

MCU Roadmaps



What is Tiva C Series®?

TI's 32-bit ARM® MCU family for home, building, and industrial applications



• Industry-leading software

- Free license & royalty free
- Software libraries provided in ROM
- Supported on 5 separate IDEs

• Advanced Integration

- 1 MSPS 12-bit ADCs
- USB Host / Device / On-The-G, CAN
- The most SPI, UART and I2C in the market

• Broad Portfolio

- 50+ Cortex-M4F devices
- Up to 256KB Flash, 32KB SRAM
- 64-LQFP, 100-LQFP, 144-LQFP, 157-BGA
- Roadmap to 1MB Flash, 256KB SRAM, Ethernet

• Time to Market

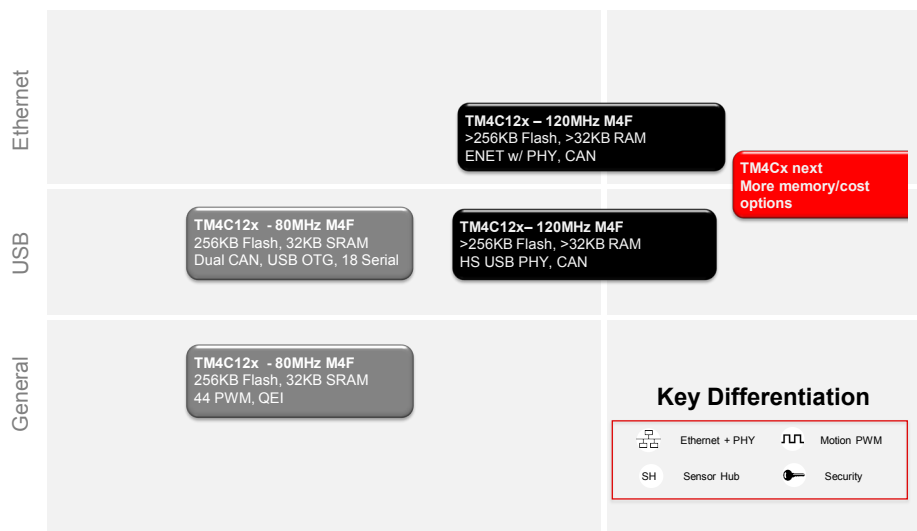
- Get started in 10 min or less
- Tiva LaunchPad is only \$12.99!

Tiva-C Roadmap

Production Sampling Development Concept

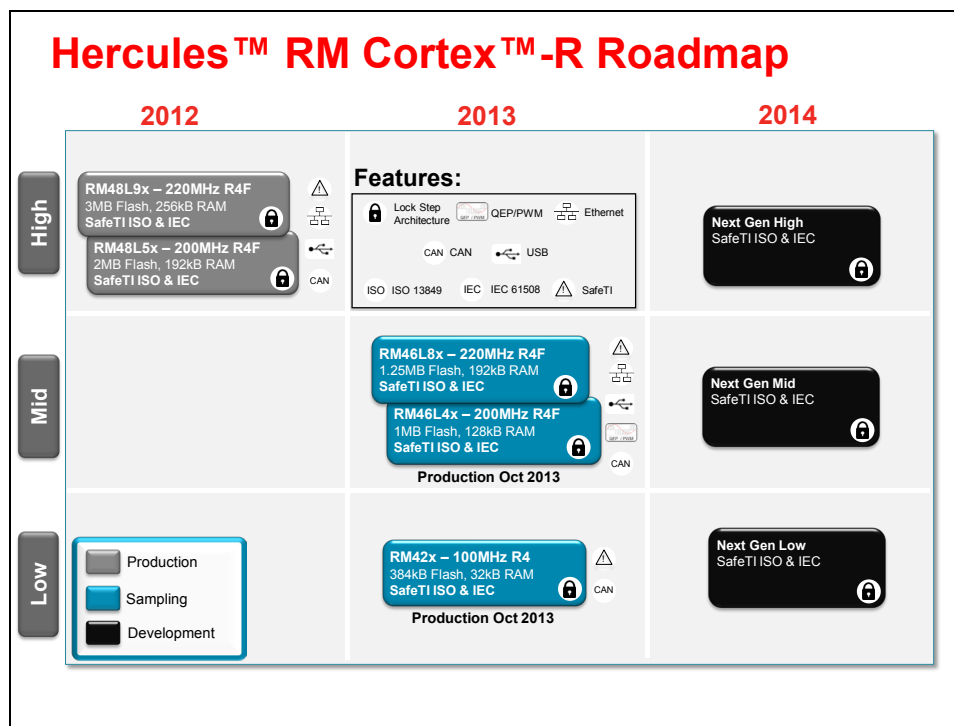
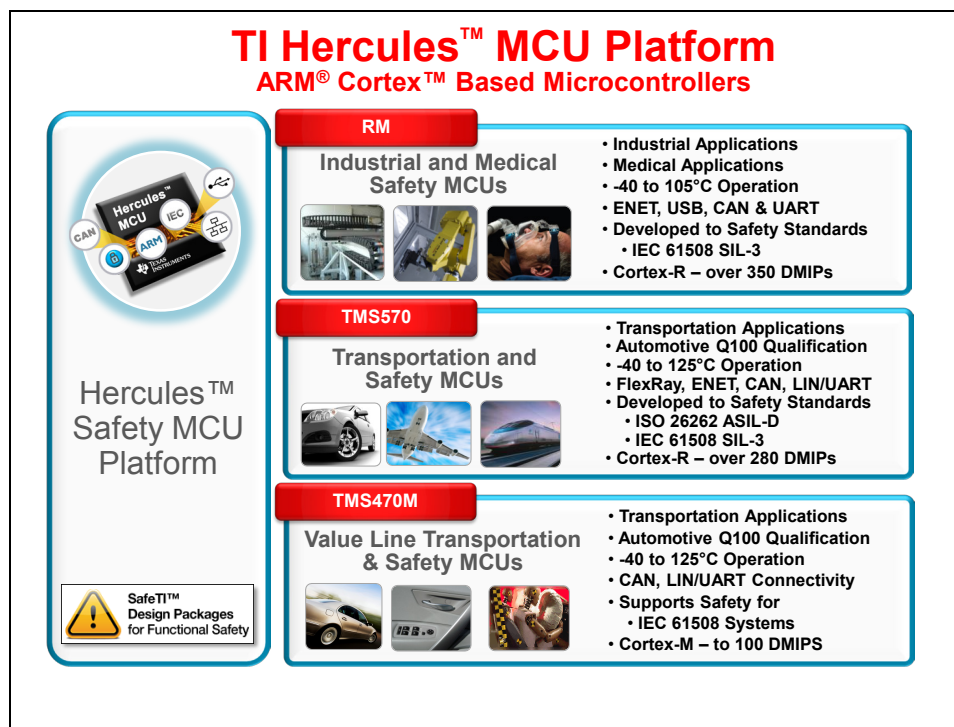
2013

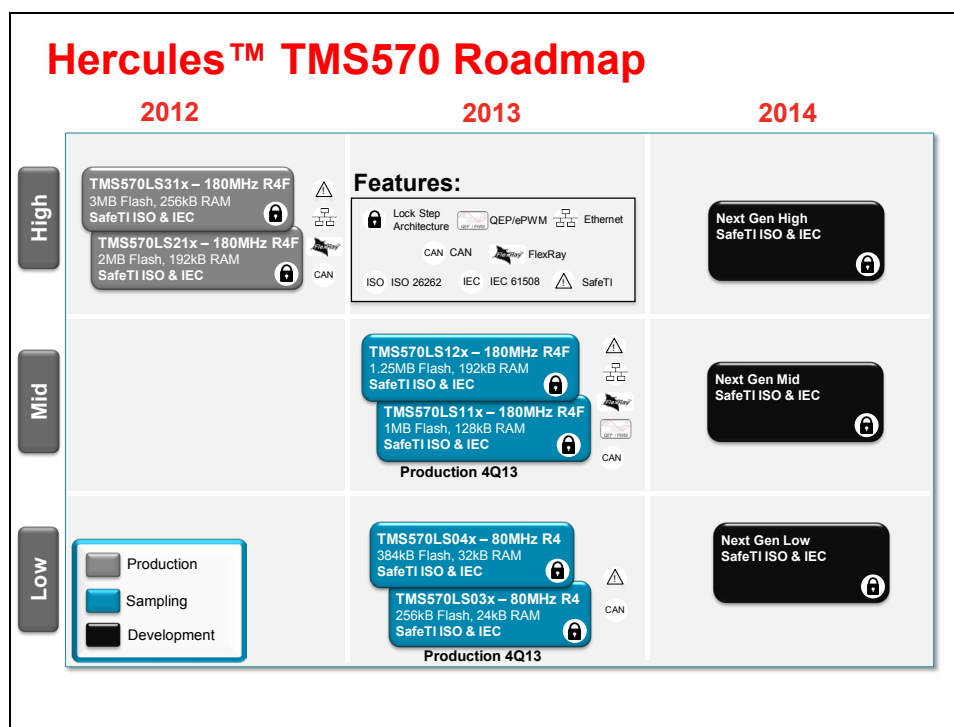
2014+



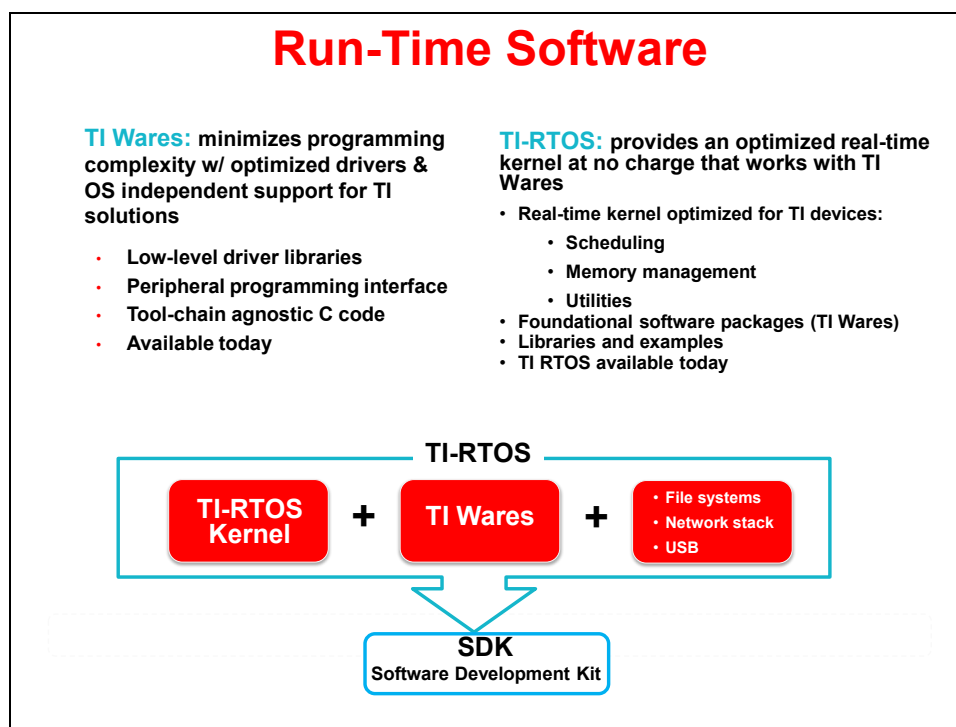
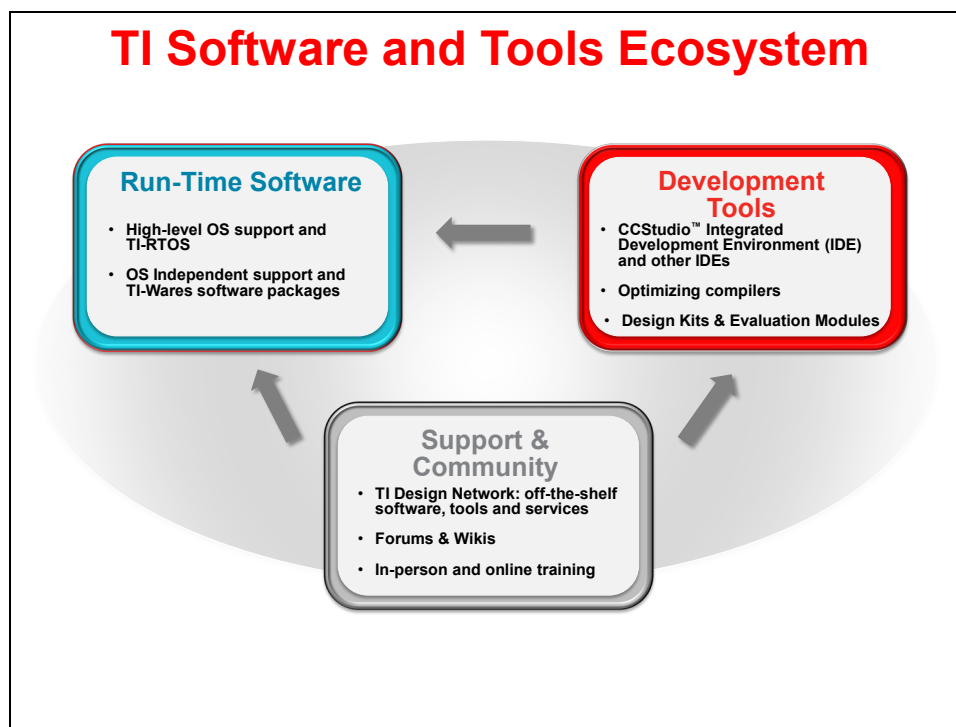
TEXAS INSTRUMENTS

9





TI Software and Development Tools



MSP430 Development Tools

MSP430 IDEs



Unparalleled integration into CCS

- Featuring MSP430Ware
- Grace
- ULP Advisor
- More...



Continued collaboration with IAR

- Integrating MSP430Ware content into IAR
- Up-to-date device support



Increased support for MSPGCC

- Free, Open Source & community-supported
- Up-to-date device support
- Coming soon – code examples, projects & other resources

MSP430Ware – A collection of MSP430 resources

Everything you need to become an MSP430 expert delivered in an intuitive & sleek GUI.

Find documentation & software resources quickly & easily:

- Featuring **Brand New Driver Library**
- C Code Examples
- Grace templates
- Datasheets
- User Guides
- HW Design Files



Software Tools for Ultra-Low Power (ULP)

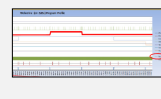
ULP Advisor™

- Checks your code against a "ULP Checklist"
- Currently 15 ULP Rules, with a growing backlog of new rules
- Highlights areas of improvement and points to helpful resources such as code examples, documentation & more



EnergyTrace™

- During debug, EnergyTrace reports back current status of the CPU, peripherals & timers.
- Allows developers to correlate power-relevant information to the MCU program code
- Coming Soon...



Other MSP430 Software Tools



Grace™ Code Generation Tool for enabling & configuring GPIO and peripherals



Other software packages include:

- Capacitive touch library
- Energy Library (AFE2xx)
- USB Developer's Package
- RF software resources

See the complete portfolio of MSP430 Software Tools

@ ti.com/msp430software

Development Tools for MSP430

Evaluation License	<input type="checkbox"/> 32KB code-size or 30-day limit <input type="checkbox"/> Upgradeable	<input type="checkbox"/> Full function <input type="checkbox"/> JTAG limited after 90-days	N/A	N/A
Compiler	IAR C/C++	TI C/C++	MSPGCC*	MSPGCC*
Debugger and IDE	<input type="checkbox"/> C-SPY <input type="checkbox"/> Embedded Workbench	<input type="checkbox"/> TI or GDB <input type="checkbox"/> CCStudio (Eclipse-based)	Energia IDE (Arduino port)	MSPDEBUG (gdb proxy)
Full Upgrade	\$2700	\$445	Free	Free
JTAG Debugger	J-Link \$299	MSP-FET430UIF \$99	No JTAG <input type="checkbox"/> serial.printf() <input type="checkbox"/> LED or scope	MSP-FET430UIF \$99

MSPGCC*: RedHat GCC compiler in development

ULP...

TivaWare™ for C Series Features

Peripheral Driver Library

- ◆ High-level API interface to complete peripheral set
- ◆ License & royalty free use for TI Cortex-M parts
- ◆ Available as object library and as source code
- ◆ **Programmed into the on-chip ROM**



USB Stacks and Examples

- ◆ USB Device and Embedded Host compliant
- ◆ Device, Host, OTG and Windows-side examples
- ◆ Free VID/PID sharing program



Extras

- ◆ Wireless protocols
- ◆ IQ math examples
- ◆ Bootloaders
- ◆ Windows side applications

Ethernet

- ◆ lwip and uip stacks with 1588 PTP modifications
- ◆ Extensive examples



Graphics Library

- ◆ Graphics primitive and widgets
- ◆ 153 fonts plus Asian and Cyrillic
- ◆ Graphics utility tools







Sensor Library

- ◆ An interrupt driven I²C master driver for handling I²C transfers
- ◆ A set of drivers for I²C connected sensors
- ◆ A set of routines for common sensor operations
- ◆ Three layers: Transport, Sensor and Processing



Development Tools for Tiva C Series

				
Evaluation License	<input type="checkbox"/> 30-day full function <input type="checkbox"/> Upgradeable	<input type="checkbox"/> 32KB code-size or 30-day limit <input type="checkbox"/> Upgradeable	<input type="checkbox"/> 32KB code-size limited <input type="checkbox"/> Upgradeable	<input type="checkbox"/> Full function <input type="checkbox"/> JTAG limited after 90-days
Compiler	GNU C/C++	IAR C/C++	RealView C/C++	TI C/C++
Debugger and IDE	<input type="checkbox"/> GDB <input type="checkbox"/> Eclipse	<input type="checkbox"/> C-SPY <input type="checkbox"/> Embedded Workbench	µVision	<input type="checkbox"/> TI or GDB <input type="checkbox"/> CCStudio (Eclipse-based)
Full Upgrade	<input type="checkbox"/> \$99 (Personal) <input type="checkbox"/> \$2800 (Full)	\$2700	MDK-Basic (256 KB) €2000 / \$2895	\$445
JTAG Debugger		J-Link \$299	U-Link \$199	XDS100v2 \$79

TI Wiki: <http://processors.wiki.ti.com>

TEXAS INSTRUMENTS Products Applications Tools & Software Support & Community Sample & Buy About TI

Texas Instruments Wiki

Welcome to the Texas Instruments Wiki

Searching

Google™ Custom Search

- [RSS feed for Wiki changes](#)
- Check out the [FAQ](#) section, [GSG](#) category for Getting Started Guides or [Training](#) homepage for online training material.

Embedded Processors

Microcontrollers		ARM Based Processors			Digital Signal Processors	
16-bit ultra low power MCU	32-bit Real-time MCUs	32-bit ARM MCU	32-bit ARM Safety MCU	32-bit ARM MPU Performance	DSP & DSP + ARM	Multicore DSP
MSP430	C2000	Stellaris Cortex-M	Hercules Cortex-R4	Sitara Cortex-A8 and ARM9	C6000 Single Core	C6000 Multicore

Software & Development Tools

Development Tools

- Code Composer Studio™ IDE integrated development environment for TI embedded processors.
- Code Generation Tools Compiler, assembler, linker and associated tools.
- Emulation XDS JTAG emulators
- [C6000 Ease of Development Tools](#)

Technical Training Organization (TTO)

Welcome to the Texas Instruments Wiki

Searching and RSS Feed

• [G](#) Search for an article here:

Google™ Custom Search

- [RSS feed for Wiki changes](#)
- Check out the [FAQ](#) section, [GSG](#) category for Getting Started Guides or [Training](#) homepage for online training material.

Embedded Processors

Microcontrollers		ARM Based Processors		
16-bit ultra low power MCU	32-bit Real-time MCUs	32-bit ARM MCU	32-bit ARM Safety MCU	32-bit ARM MPU Performance
MSP430	C2000	Stellaris Cortex-M	Hercules Cortex-R4	Sitara Cortex-A8 and ARM9

Engineer-2-Engineer Forums

The screenshot shows the TI E2E Community website. At the top is the Texas Instruments logo and a navigation bar with links: Products, Applications, Tools & Software, Support & Community, Sample & Buy, and About TI. Below this is the 'TI E2E™ Community' header with the tagline 'engineer to engineer, solving problems' and links for 'Join' and 'Sign In with my.TI Login'. A secondary navigation bar includes 'Support Forums', 'Blogs', 'Groups', 'Videos', and '简体中文', along with a 'Search Community' button. The main content area features a search bar with the text 'Find out if your question has already been answered' and 'Search through 1,055,110 questions and answers in TI E2E Community'. Below the search bar is a section titled 'Choose a support forum to post a new question' with a grid of buttons for various TI product categories: ARM®-based Processors, Amplifiers, DLP® & MEMS, Applications, Digital Signal Processors, Broadband RF/IF & Digital Radio, Interface, Tools & Software, Microcontrollers, Clocks & Timers, Logic, Wireless Connectivity, OMAP™ Applications Processors, Data Converters, and Power Management. A link 'See all support forums here >' is also present. At the bottom, there are two sections: 'Recent Forum Activity' showing a post by 'swati arora' and 'TI E2E Top Contributors' with a list of top contributors.

<http://e2e.ti.com>

LaunchPads and BoosterPacks

TI LaunchPad MCU Evaluation Kits

- ◆ Everything you need to get started
- ◆ Embedded emulation via USB cable
- ◆ BoosterPack connector enables plug-in modules to add functionality
- ◆ Low-cost (\$10-20)
- ◆ We will use MSP-EXP430F5529LP and EK-TM4C123GXL LaunchPad in this workshop

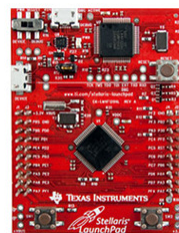
MSP430



C2000



Tiva C Series

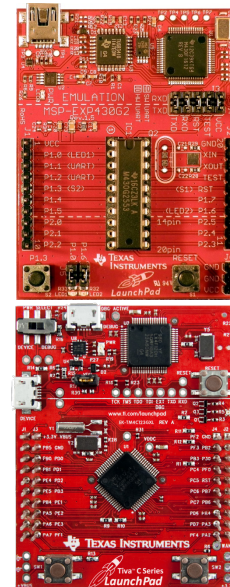


Hercules



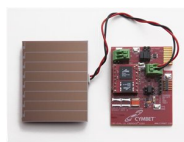
BoosterPack Connectors

- ◆ **Original Format (MSP430)**
 - VCC and Ground
 - 14 GPIO
 - Emulator Reset and Test
 - Crystal inputs or 2 more GPIO
- ◆ **XL Format is a superset of the original, adding two rows of pins with:**
 - USB V_{BUS} and Ground
 - 18 additional GPIO



Available Boosterpacks...

Some of the Available BoosterPacks



Solar Energy Harvesting



RF Module w/ LCD



Olimex 8x8 LED Matrix



TMP006 IR Temperature Sensor



Universal Energy Harvesting



Inductive Charging



Sub-1GHz RF Wireless



C5000 Audio Capacitive Touch



Capacitive Touch

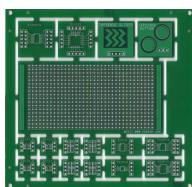


Proto Board

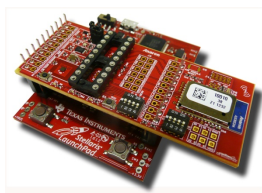
TPL0401 SPI Digital Pot.

TPL0501 SPI Digital Pot.

Some of the Available BoosterPacks



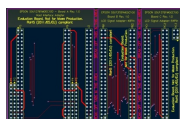
Proto board



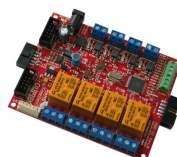
ZigBee Networking



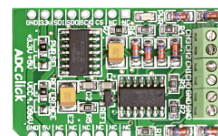
OLED Display



LCD Controller Development Package

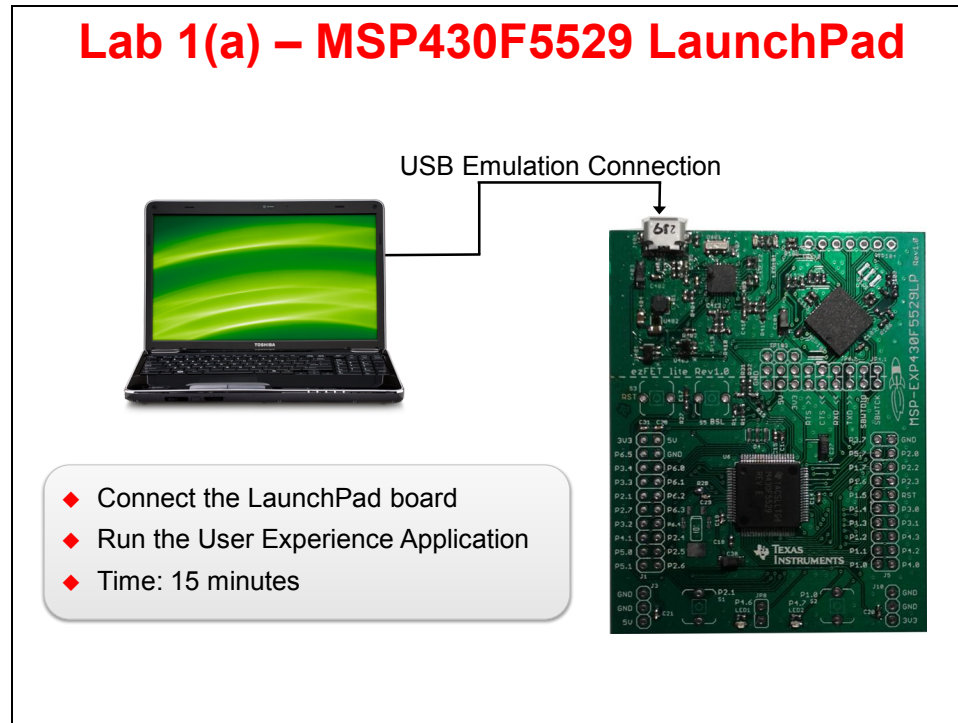


MOD Board Adapter



Click Board Adapter

Lab 1(a) – MSP4305529 LaunchPad User Experience



1. Examine the MSP-EXP430F5529LP LaunchPad kit contents

- ▶ Open up your MSP430F5529 LaunchPad box. You should find the following:
 - The MSP-EXP430F5529LP LaunchPad Board
 - USB cable (A-male to micro-B-male)
 - “Meet the MSP430F5529 Launchpad Evaluation Kit” card

2. Initial Board Set-Up

- ▶ Using the included USB cable, connect the USB emulation connector on your evaluation board to a free USB port on your PC.

A PC's USB port is capable of sourcing up to 500 mA for each attached device, which is sufficient for the evaluation board. If connecting the board through a USB hub, it must be a powered hub. The drivers should install automatically.

3. Run the User Experience Application

Your LaunchPad Board came preprogrammed with a User Experience application. This software enumerates as a composite USB device (defined in \USB_config)

HID (Human Interface device): an emulated keyboard

MSC (Mass Storage class): an emulated hard drive with FAT volume

The contents of the hard drive can be viewed with a file



browser such as Windows Explorer.

4. View the contents of the emulated hard drive

► Open Windows Explorer and browse to the emulated hard drive.

You should see four files there:

`Button1.txt` – the contents of this file are "typed out" to the PC, using the emulated keyboard when you press button S1

`Button2.txt` – the contents of this file are "typed out" to the PC, using the emulated keyboard when you press button S2

`MSP430 USB LaunchPad.url` – when you double-click, your browser launches the MSP-EXP430F5529LP home page

`README.txt` – a text file that describes this example

5. Use S1 and S2 buttons to send ASCII strings to the PC

The LaunchPad's buttons S1 and S2 can be used to send ASCII strings to the PC as if they came from a keyboard. These strings that are sent are stored in the files `Button1.txt` and `Button2.txt`, respectively; and these files can be modified to change the strings. The text string is limited to 2048 characters, so even though you can make the file contents longer, be aware that the string will be truncated to 2048.

► Open Notepad. In the start menu, type "Run", then type "Notepad".

► To send the strings to Notepad, press S1. What do you see? Now press S2. What happens now?

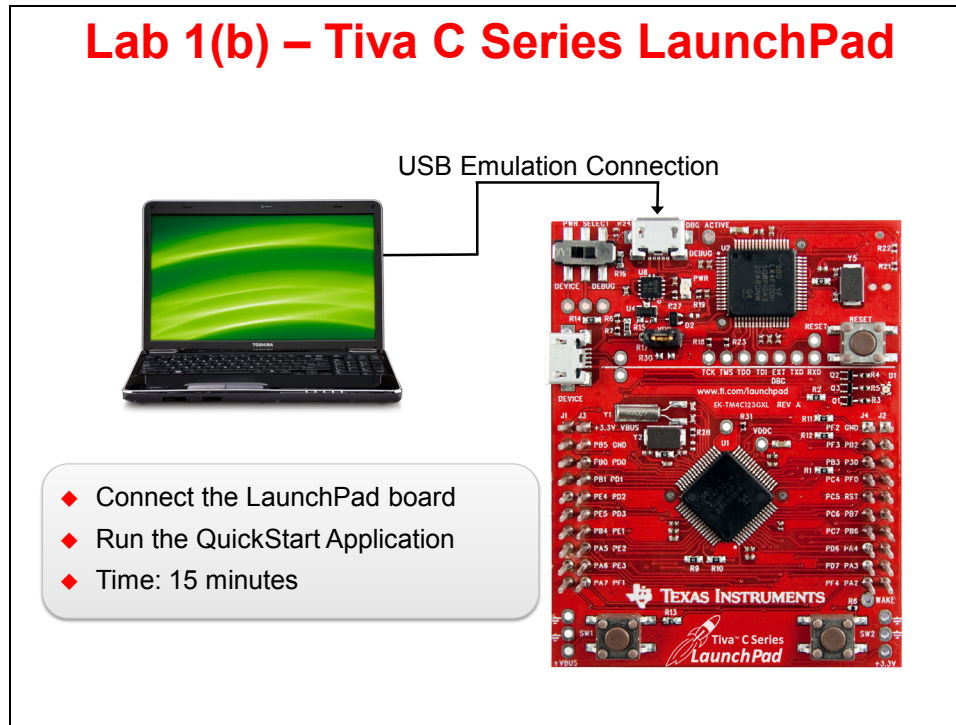
The default ASCII strings stored in the two text files are are:

`Button1.txt`: "Hello world"

`Button2.txt`: an ASCII-art picture of the LaunchPad rocket

For the rocket picture, please note that the display can be affected by settings of the application receiving the typed characters. On Windows, the basic Notepad.exe is recommended.

Lab 1(b) – Tiva C Series LaunchPad QuickStart



6. Examine the TM4C123GXL LaunchPad kit contents

- ▶ Open up your TM4C123GXL LaunchPad box. You should find the following:
 - The TM4C123GXL LaunchPad Board
 - USB cable (A-male to micro-B-male)
 - README First card

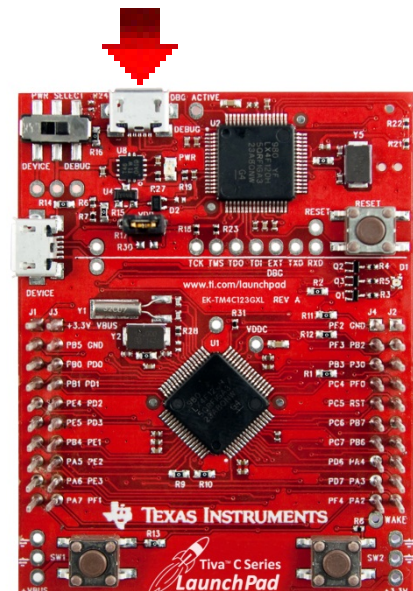
7. Initial Board Set-Up

- ▶ Using the included USB cable, connect the USB emulation connector on your evaluation board (marked DEBUG) to a free USB port on your PC.

A PC's USB port is capable of sourcing up to 500 mA for each attached device, which is sufficient for the evaluation board. If connecting the board through a USB hub, it must be a powered hub. The drivers should install automatically.

The TM4C123GXL LaunchPad Board ICDI USB port (marked DEBUG and shown in the picture below) is a composite USB port and consists of three connections:

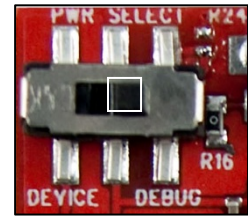
- Stellaris ICDI JTAG/SWD Interface – debugger connection
- Stellaris ICDI DFU Device – firmware update connection
- Stellaris Virtual Serial Port – a serial data connection



8. Run the QuickStart Application

Your LaunchPad Board came preprogrammed with a quickstart application. Once you have powered the board, this application runs automatically. You probably already noticed it running as you installed the drivers.

► Make sure that the power switch in the upper left hand corner of your board is in the right-hand DEBUG position as shown:



The software on the TM4C123GH6PM uses the timers as pulse-width modulators (PWMs) to vary the intensity of all three colors on the RGB LED (red, green, and blue) individually. By doing so, your eye will perceive many different colors created by combining those primary colors.

The two pushbuttons at the bottom of your board are marked SW1 (the left one) and SW2 (the right one). ► Press or press and hold SW1 to move towards the red-end of the color spectrum. ► Press or press and hold SW2 to move towards the violet-end of the color spectrum.

If no button is pressed for 5 seconds, the software returns to automatically changing the color display.

9. Enter Hibernate Mode

► Press and hold both **SW1** and **SW2** for 3 seconds to enter hibernate mode.

In this mode the last color will blink on the LEDs for ½ second every 3 seconds. Between the blinks, the device is in the VDD3ON hibernate mode with the real-time-clock (RTC) running.

► Press **SW2** at any time to wake the device and return to automatically changing the color display.

10. Communicate with the board through the UART

We can communicate with the board through the UART. The UART is connected as a virtual serial port through the emulator USB connection. The following steps will show how to open a connection to the board using HyperTerminal (in WinXP) and PuTTY (in Windows 7 or 8).

We need to find the COM port number of the Stellaris Virtual Serial Port in the Device Manager. **Skip to the next page if you are using Windows 7 or 8.**

For Windows XP Users Only:

► Click on the Windows Start button. ► Right-click on My Computer and select Properties from the drop-down menu.

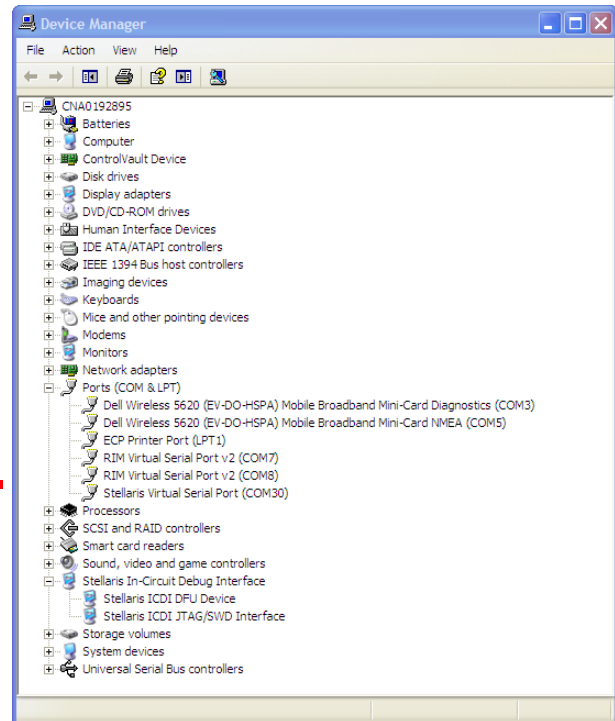
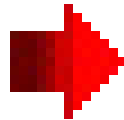
In the System Properties window, ► click the Hardware tab.

► Click the Device Manager button.

The Device Manager window displays a list of hardware devices installed on your computer and allows you to set the properties for each device.

Expand the Ports heading and write the number for the Stellaris Virtual Serial Port here:

COM_____

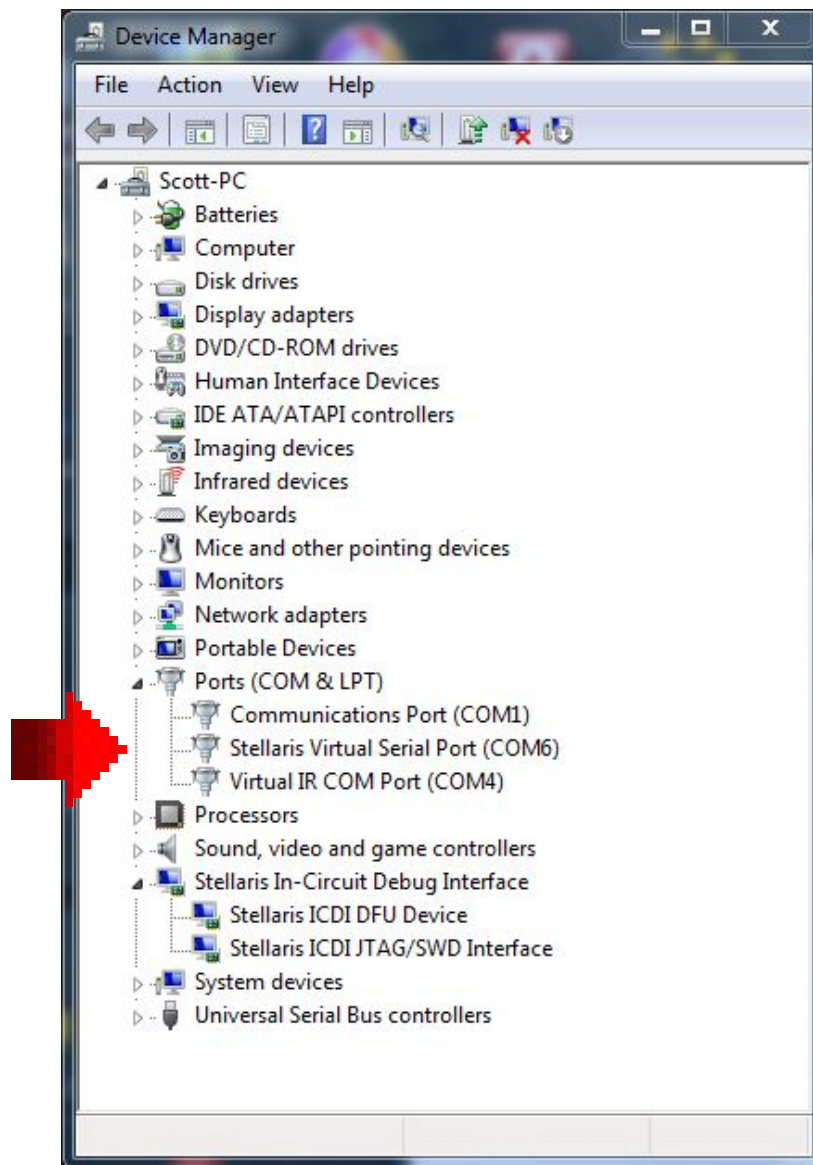


For Windows 7 or Windows 8 Users Only:

► Click on the Windows Start button. ► Right-click on Computer and select Properties from the drop-down menu.

► Click on Device Manager on the left of the dialog.

The Device Manager window displays a list of hardware devices installed on your computer and allows you to set the properties for each device.

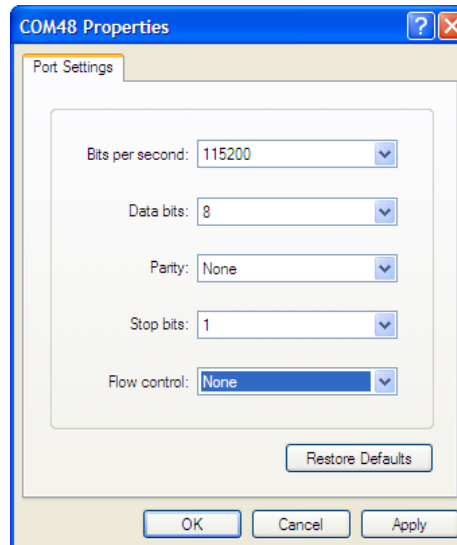


► Expand the Ports heading and write number for the Stellaris Virtual Serial Port here:

COM_____

11. Windows XP Users Only – Open HyperTerminal

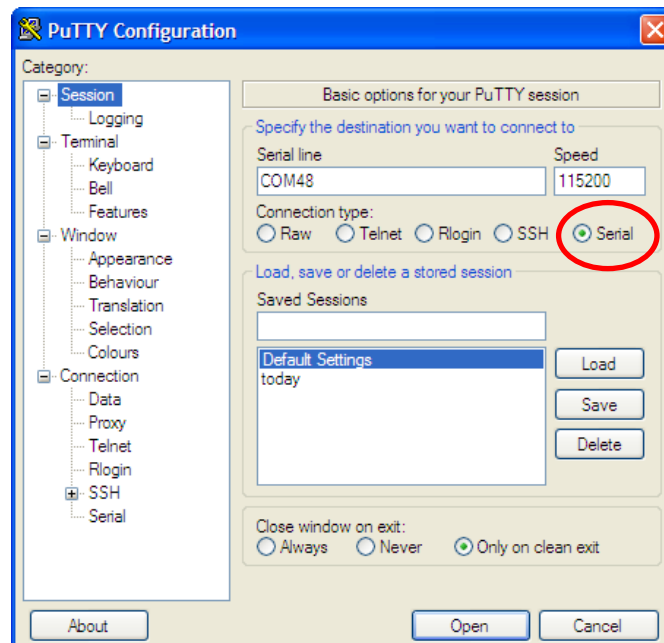
► Clicking Start → Run..., then type `hypertrm` in the Open: box and click OK. Pick any name you like for your connection and click OK. In the next dialog box, change the Connect using: selection to COM##, where ## is the COM port number you noted earlier. Click OK. Make the selections shown below and click OK.



When the terminal window opens, press Enter once and the LaunchPad board will respond with a > indicating that communication is open. Skip the next step.

12. Windows 7 or Windows 8 Users Only – Open Putty

► Double click on `putty.exe`. Make the settings shown below and then click Open. Your COM port number will be the one you noted earlier



When the terminal window opens, press Enter once and the LaunchPad board will respond with a > indicating that communication is open.

13. You can communicate by ► typing the following commands and pressing enter:

help: will generate a list of commands and information

hib: will place the device into hibernation mode. Pressing SW2 will wake the device.

rand: will start a pseudo-random sequence of colors

intensity: adjust the LED brightness between 0 to 100 percent. For instance intensity 100 will change the LED to maximum brightness.

rgb: follow with a 6 hex character value to set the intensity of all three LEDs. For instance: rgb FF0000 lights the red LED, rgb 00FF00 lights the blue LED and rgb 0000FF lights the green LED.

14. ► Close your terminal program.



You're done.

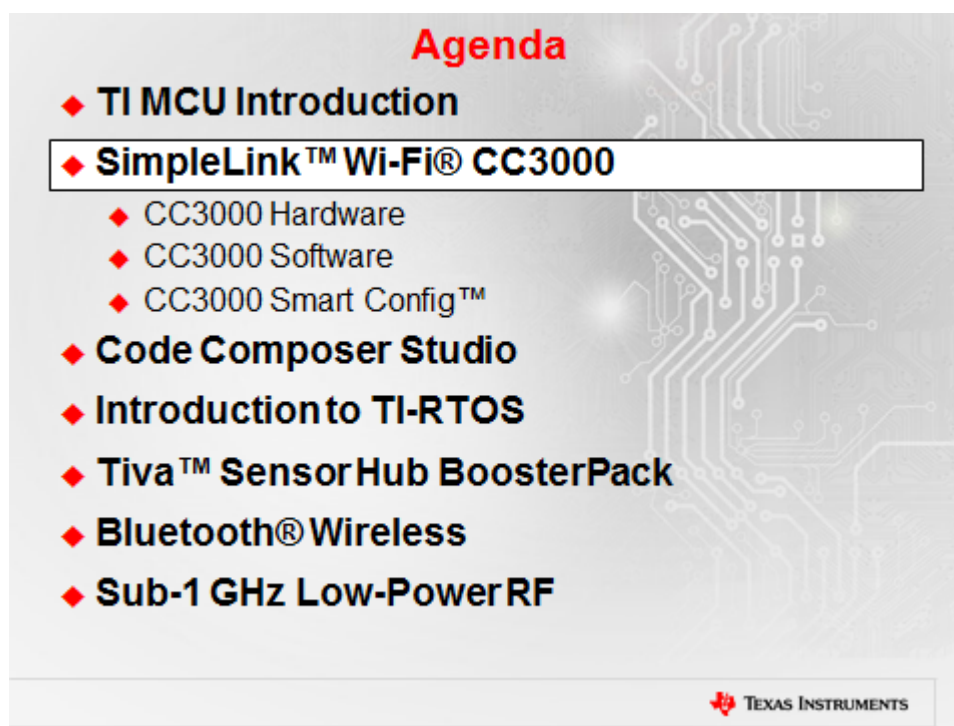
SimpleLink™ Wi-Fi® CC3000

Introduction

The SimpleLink™ CC3000 WiFi BoosterPack is a great way to add Wi-Fi network processor to a TI MCU. This module minimizes host MCU software requirements making it the ideal solution for embedded applications using any low-cost and low-power MCU.

This BoosterPack aids in the evaluation and development of CC3000 solutions and contains the CC3000 module, power supply, and standard BoosterPack headers to connect to MSP430™, Tiva™ C Series, and future TI MCU LaunchPad evaluation kits. Additionally, this complete platform solution includes software drivers, sample applications, API guide, user documentation and a world-class support community.

Agenda



Agenda

- ◆ TI MCU Introduction
- ◆ **SimpleLink™ Wi-Fi® CC3000**
 - ◆ CC3000 Hardware
 - ◆ CC3000 Software
 - ◆ CC3000 Smart Config™
- ◆ Code Composer Studio
- ◆ Introduction to TI-RTOS
- ◆ Tiva™ SensorHub BoosterPack
- ◆ Bluetooth® Wireless
- ◆ Sub-1 GHz Low-Power RF




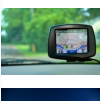










TEXAS INSTRUMENTS

Chapter Topics

SimpleLink™ Wi-Fi® CC3000.....	2-1
<i>CC3000 Overview</i>	<i>2-3</i>
<i>CC3000 Hardware</i>	<i>2-5</i>
<i>CC3000 Software.....</i>	<i>2-7</i>
<i>CC3000 Smart Config™</i>	<i>2-12</i>
<i>Lab 2 – TCP Echo for CC3000 Example</i>	<i>2-15</i>
Objectives.....	2-15
Lab Procedure.....	2-16

CC3000 Overview

The industry's broadest wireless connectivity portfolio

Supported standards						
13.4KHz /13.56MHz	Sub 1GHz	2.4GHz to 5GHz			Wi-Fi	Satellite
RFID NFC ISO14443A/B ISO15693	SimpliciTI 6LoWPAN W-MBus	SimpliciTI PurePath Wireless	ZigBee® 6LoWPAN RF4CE	Bluetooth® technology Bluetooth® low energy ANT	Wi-Fi	GPS
Example applications						
						
						
						
Product line up						
TMS37157 TRF796x TRF7970	CC1110 CC1190 CC11xL CC430 CC112X CC120X CC1180	CC2500 CC2543/44/45 CC2590/91 CC8520/21 CC2530/31	CC2530 CC2530ZNP CC2531 CC2533 CC2520	CC2560/4 CC2540/1 CC2570/1	WL1271/3 WL128xx CC3000	CC4000

Red = SimpleLink family

Enabling the IoT

Monitor and control
CC3000-enabled products
from a mobile device,
in your home local network





SimpleLink™ Wi-Fi® CC3000

Same great SimpleLink features for Internet Connectivity

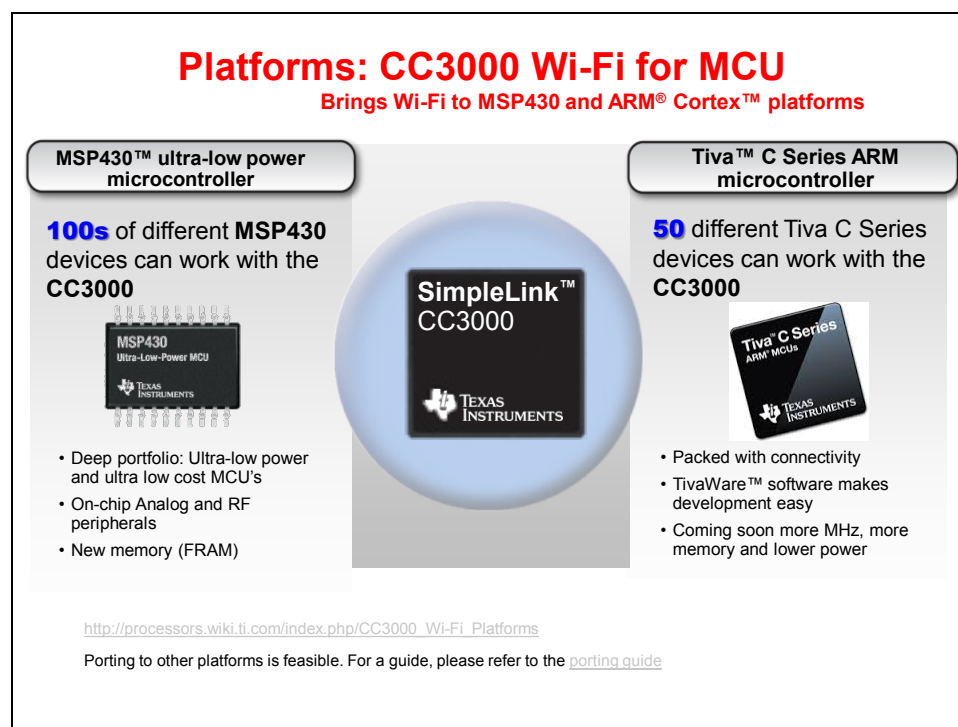
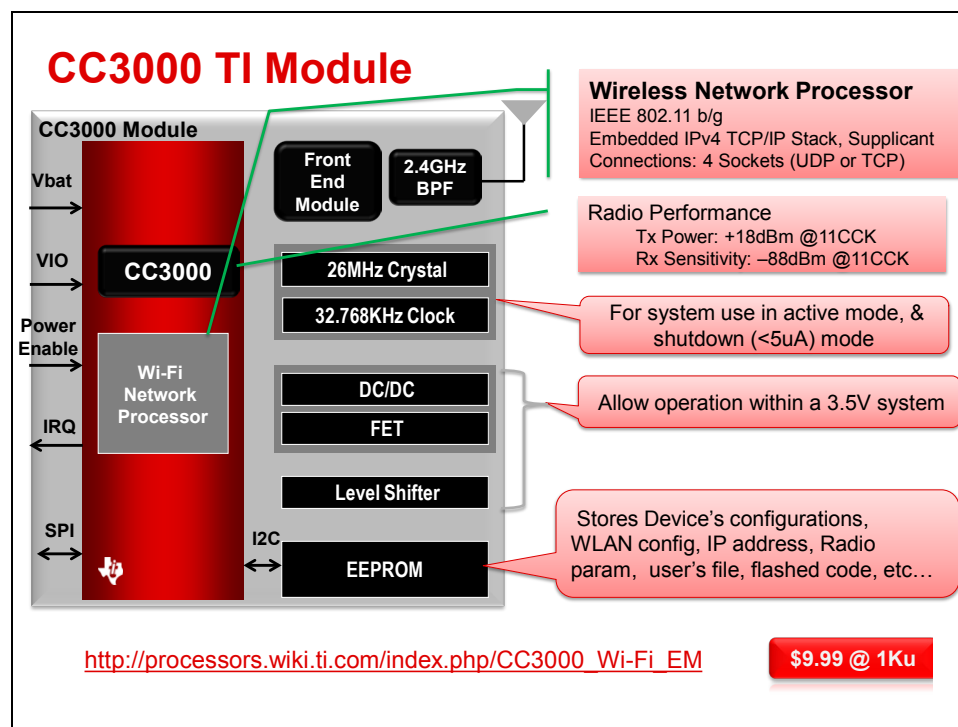
Self-Contained Solution	Low MIPS / CPU Overhead
Very Simple API	Low Memory Footprint

Coming soon with software enhancements

SmartConfig™ Technology Simplest Wi-Fi configuration scheme	Flexible Memory Size Pair with low cost, low MIPS, small size MCUs
Service Discovery Enhance your mobile app experience	Cloud Enabled Client / server cloud connectivity

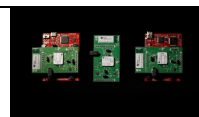
Offered as a TI module – CC3000MOD






CC3000 Hardware



CC3000 kits and bundles







- CC3000 now offered as a BoosterPack
- Working with both the [MSP430™](#) and [Tiva™ C Series](#) microcontroller (MCU) LaunchPad evaluation kits



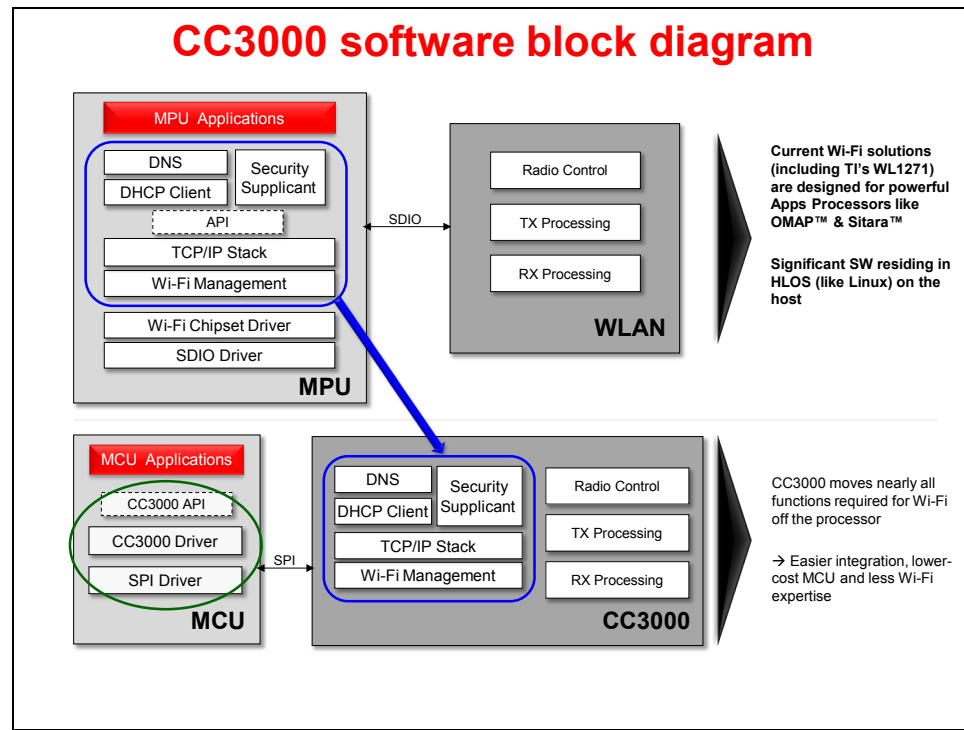
	Kit Name & Price	Kit Contents	Focus Market Areas
	CC3000EM \$35	<ul style="list-style-type: none"> • 1 TI CC3000 Evaluation Module • Compatible Experimenter Boards (sold separately) - MSP-EXP430F5529, MSP-EXPF430FR5739, TM4C123G (EK-LM4F232) 	MSP430 Applications: Ultra-low-power and portable applications Tiva C Applications: Connectivity applications within the home, building, commercial, and industrial markets
	CC3000BOOST \$35 See Board Tour Video	<ul style="list-style-type: none"> • 1 TI CC3000 BoosterPack • Compatible LaunchPads (sold separately) - MSP-EXP430G2, EK-TM4C123GX 	
	MSP-EXP430G2-CC3000BOOST \$40.99	<ul style="list-style-type: none"> • Soft bundle <ul style="list-style-type: none"> • 1 CC3000 BoosterPack • 1 MSP-EXP430G2 Launchpad 	
	EK-TM4C123GXL-CC3000BOOST \$43.99 (coming soon)	<ul style="list-style-type: none"> • Soft bundle <ul style="list-style-type: none"> • 1 CC3000 BoosterPack • 1 EK-TM4C123GXL Launchpad 	
	MSP-EXP430FR5739-CC3000EM \$57	<ul style="list-style-type: none"> • Soft Bundle <ul style="list-style-type: none"> • 1 CC3000 EM board • 1 MSP-EXP430FR5739 experimenter board 	

TI SimpleLink™ Certified Wi-Fi® Module

- Reduces time to market
- Guarantees performance and test reports complies with participating countries
- Reduces certification costs
- FCC/IC Certified
- ETSI Tested & CE Radio compliant

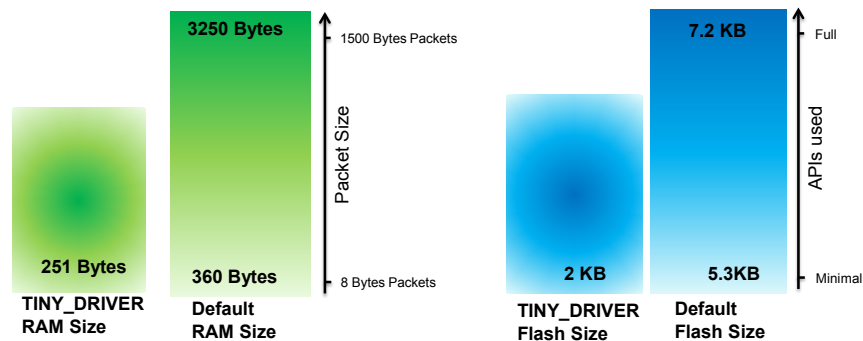
		<ul style="list-style-type: none"> • Certified for United States • Tests reports recognized by other countries for filing
		<ul style="list-style-type: none"> • Certified for Canada • Accepts FCC test reports for filing certifications
		<ul style="list-style-type: none"> • CE Radio Approval • ETSI testing is recognized by 40+ countries for filing

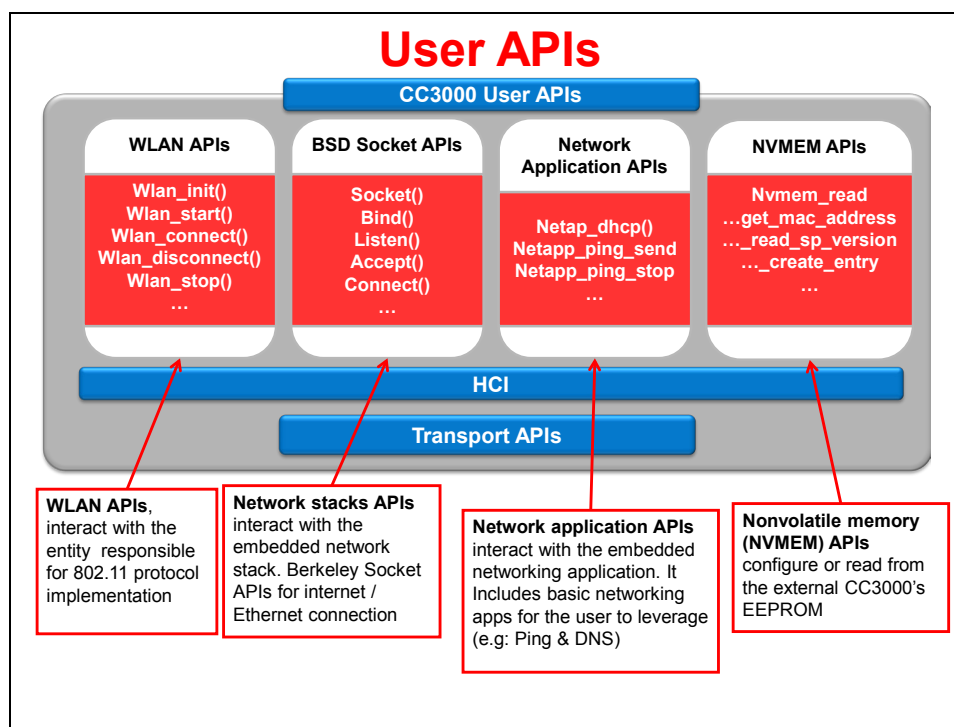
CC3000 Software



Flexible Memory Size for CC3000

- ◆ CC3000's has flexible memory compile options
 - ◆ **Default:** Full Options
 - ◆ **Tiny_driver:** Lowest footprint, limited capabilities
e.g: Non secured connection
- ◆ Final code size depends on actual APIs used
- ◆ RAM size depends on the largest packet sent and received





Doxygen API

- ◆ Go to the [CC3000 wiki web page](#).
- ◆ Download the [API Documentation](#)
- ◆ Open the file `...\swrc266\html`
- ◆ Open `index.html`.
- ◆ Find/Search the API you need.

SimpleLink API Reference Manual 1.11

The screenshot shows the SimpleLink API Reference Manual 1.11. On the left, a sidebar lists the contents: Main Page, Modules, Data Structures, Files, and a list of modules including Netapp_api, Nvmem_api, and Socket_api. The "Wlan_api" module is selected and expanded, showing a list of functions: wlan_init, wlan_start, wlan_stop, wlan_connect, wlan_disconnect, wlan_add_profile, wlan_ioctl_del_profile, wlan_ioctl_event_mask, wlan_ioctl_statusget, wlan_ioctl_set_connection_policy, wlan_ioctl_get_scan_results, wlan_ioctl_set_scan_params, wlan_smart_config_start, wlan_smart_config_stop, wlan_smart_config_set_prefix, and wlan_smart_config_process.

The main content area displays the C code for the `wlan_init` function:

```
void wlan_init(tWlanCB sWlanCB, tFWPatches sFWPatches, tDriverPatches sDriverPatches,
tBootLoaderPatches sBootLoaderPatches, tWlanReadInterruptPin sReadWlanInterruptPin,
tWlanInterruptEnable sWlanInterruptEnable, tWlanInterruptDisable sWlanInterruptDisable,
tWriteWlanPin sWriteWlanPin)
{
    Initialize wlan driver.

    void wlan_start (unsigned short usPatchesAvailableAtHost)
    {
        Start WLAN device. This function asserts the enable pin of the device (WLAN_EN), starting the HW
        initialization process. The function blocked until device initialization is completed. Function also
        configure patches (FW, driver or bootloader) and calls appropriate device callbacks.
    }

    void wlan_stop (void)
    {
        Stop WLAN device by putting it into reset state.
    }

    long wlan_connect (unsigned long ulSecType, char *ssid, long ssid_len, unsigned char *bssid, unsigned
    char *key, long key_len)
    {
        Connect to AP.
    }

    long wlan_disconnect (void)
    {
        Disconnect connection from AP.
    }

    long wlan_add_profile (unsigned long ulSecType, unsigned char *ucSsid, unsigned long ulSsidLen,
    unsigned char *ucBssid, unsigned long ulPriority, unsigned long ulPairwiseCipher_Or_Key, unsigned
    long ulGroupCipher_TxKeyLen, unsigned long ulKeyMgmt, unsigned char *ucPT_OrKey, unsigned
    long ulPassPhraseLen)
    {
        When auto start is enabled, the device connects to station from the profiles table. Up to 7 profiles
        are supported. If several profiles configured the device choose the highest priority profile, within
    }
}
```

SimpleLink™ Wi-Fi CC3000				
Getting Started	Hardware	Software	Test & Certification	Support & Community
Starting Out [pdf] CC3000 Overview Key Software Features First Time Getting Started Get the HW Documentation CC3000 Boot Loader User Guides	Information [pdf] Evaluation Modules Platforms Datasheet Errata Other Technical Documents Reference Design Schematic Checklist PCB Design Guidelines	Releases [pdf] Release Notes Patch Programmer Sample Apps Downloads Platform Dependent Downloads Smart Config Apps Host Guides Host Driver Porting Host Programming Serial Configuration Software API [pdf] API Documentation	System Test [pdf] RF System Test tool Production Line Test TX Rate socket Certification [pdf] Product Certification	Forums [pdf] SimpleLink Wi-Fi Forum E2E Community Debug Tools [pdf] Logger for remote debugging
Development Tools		Information		

Typical Commands/Events Flow

1. Initialization

- Initialize the SPI driver & CC3000 driver `init_spi();`
`wlan_init(...);`
- Start WLAN `wlan_start();`

2. Setting IP address configuration

- Use DHCP for dynamic IP or configure static IP manually `netapp_dhcp(...);`

3. Association to an AP & IP address configuration

- Use an existing profile stored in CC3000 `done automatically!`
- MCU code to configure manually, or program a new profile `wlan_connect(...);`
- Use FirstTimeConfig procedure to add a new profile `wlan_smart_config_set_prefix();`
`wlan_smart_config_start();`
`wlan_smart_config_stop();`

Typical Commands/Events Flow

4. Create connection with another networked device

- Open socket `socket(); bind();`
- Connect or accept socket `connect(); accept();`
`listen();`

5. Start MCU application


- Send/receive data `send(); read(socket..);`
- Handle connections / sockets `close();`

6. Close WLAN connection

- Disconnect connection from AP `wlan_disconnect();`
- Stop WLAN device by putting it into reset state `wlan_stop();`

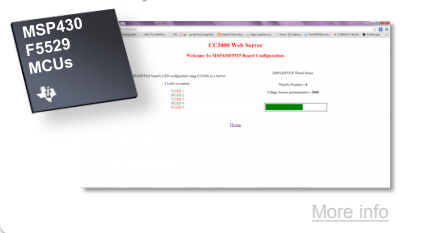
**Now supported by SimpleLink™
Wi-Fi® CC3000**

Home Automation



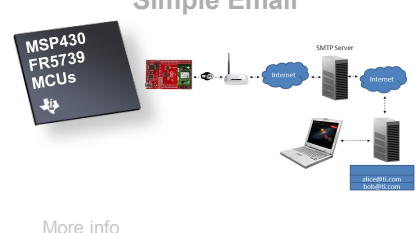
[More info](#)

Simple HTTP Server




[More info](#)

Simple Email



[More info](#)

Platforms Supported



[More info](#)

CC3000 sample applications

Download all examples in Single SDK – [Click Here](#)
Now available for Windows & Linux environments

Application / Demo	Platform Supported	Description	Version	Guide/Documentation
Basic Wi-Fi Application	<ul style="list-style-type: none"> •FRAM - MSP430FR5739 •MSP430F5529 •MSP430 Launchpad - MSP-EXP430G2 •Tiva C Series - TM4C123GH6PGE •Tiva C LP EK-TM4C123GXL 	This application gives the user a sample application to evaluate basic commands and build upon or change as their custom application needs	1.11	<ul style="list-style-type: none"> •MSP430FRAM (MSP430FR5739) •MSP430 Launchpad (MSP430G2553) •MSP430F5529 •Tiva C Series (TM4C123GH6PGE) •Tiva C LP (EK-TM4C123GXL)
Simple Email	MSP430 FRAM (MSP430FR5739)	This demo shows how to use the CC3000 to send an email via the Simple Mail Transfer Protocol (SMTP)	1.11	CC3000 Email Demo Application
Simple HTTP Server	MSP430F5529	This demo shows how CC3000 module can be used as an HTTP server. It allows users to easily communicate with the device	1.11	CC3000 HTTP Web Server Guide
SNTP Simple Network Time Protocol	Tiva C Series (TM4C123GH6PGE)	This protocol implementation is added on top of the basic Wi-Fi application demo; SNTP is used to retrieve the accurate time from multiple servers	1.11	SNTP Guide
Home Automation (Android only)	MSP430 FRAM (MSP430FR5739)	This demo shows how CC3000 module can be used to secure the house and control appliances using an android application	1.11	Home Automation Application
Multithread - TI	Tiva C LP (EK-TM4C123GXL)	This demos shows multithread application with three concurrent TCP sessions (1 Tx, 2 Rx). Can work with any RTOS, sample supports TI RTOS	1.11	Multithread Guide

Patch Programmer

CC3000 Updates

- The service pack is used to distribute new features and fixes to CC3000 SimpleLink device
- Patch Programmer utility burns TI's released Service Pack into the CC3000 EEPROM using CC3000 Host Driver API's.

Methods

- Without IDEs: Using the **Patch Programmer Utility / Installer**
 - Most common way to flash a new service pack, it uses the host driver's APIs
- With IDEs: IAR Embedded Workbench v5.51.4 (MSP430) / v6.50.1 (Tiva C Series), or CCSv5.3

Steps

- An update of the Service Pack is divided into two parts
 - Step 1: Driver Patch Programmer
 - Step 2: Firmware Patch Programmer
- Flashing the network stack patches and the WLAN low level activities patches is separated into two utilities due to limited memory on most MSP430 platforms. On the LM4F232H5QD platform these two flashing steps are united into one.

For detailed instructions on how to use the patch programmer utility for the different platforms
Refer to the following wiki page: http://processors.wiki.ti.com/index.php/CC3000_Patch_Programmer

CC3000 Smart Config™

Simple provisioning with SmartConfig™ apps

Overview

Create a great user experience for IoT applications while setting up a headless device onto the Wi-Fi network




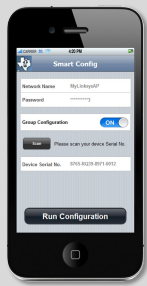
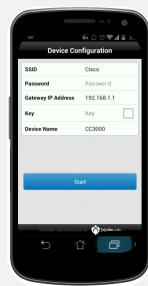
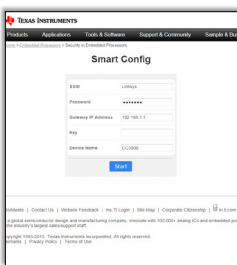
TI has a revolutionary provisioning method based on a proprietary algorithm - SmartConfig technology

Key Points

- The market is confirming this feature as a true differentiator
- ODMs/OEMs can customize their application using the TI SmartConfig library for iOS, Android or Java
- Only one step is needed compared to multiple steps required by other methods
- Multiple devices can be provisioned at the same time
- The configuration protocol is secured with AES-128 encryption



Three ways to use SmartConfig™ technology

 iOS	 Android	 Java
 <ul style="list-style-type: none"> • Using iPhone, iPod and iPad • Sample App available on iTunes • Library and sample code available on ti.com <p>Click here for more information</p>	 <ul style="list-style-type: none"> • Using Android phones and tablets • Sample App, library and sample code available on ti.com <p>Click here for more information</p>	 <ul style="list-style-type: none"> • Using a web browser in PCs and laptops • Locally Installed or embedded in web site • Sample code and library available on ti.com <p>Click here for more information</p>

Secured Provisioning with SmartConfig™

- ◆ SmartConfig encrypts the SSID and the Password before they are sent over the air with an AES-128 cipher
- ◆ The key used in the encryption is a 128 bit long private secret between the CC3000 end point and the configuration application



Service Discovery with CC3000

- ◆ CC3000 supports service discovery using industry standard protocols: Bonjour .
- ◆ Integrated with SmartConfig™ to give a great first time provisioning experience
- ◆ Once provisioned on the network,
 - ◆ The CC3000-enabled device can advertise itself within a local link network such as a home network
 - ◆ Shows up as a new service on a Phone, a Tablet or a PC allowing the application to connect quickly and easily to the device
 - ◆ Allow easy access to any registered Phone or PC within the home network
 - ◆ Access to the device can be protected by password

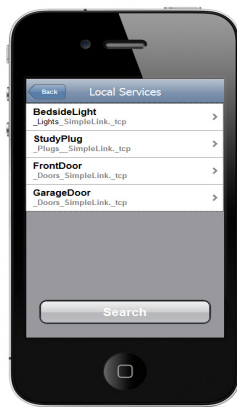
mDNS: Self - Advertisement

- ◆ Client are connected to an access point
- ◆ Clients advertise their service, via multicast packets
- Providing its own information
 - ◆ Device name
 - ◆ IP address
 - ◆ Vendor
 - ◆ Port number
- ◆ Any listening device can connect to these clients



mDNS: Service Discovery

- Enhance your app experience with easy access to SimpleLink devices
- Easily discover embedded Wi-Fi devices, available on your network



- CC3000 supports service discovery using industry standard protocols: Bonjour
mDNS + Smart Config → Great first time provisioning experience

mDNS multicast Service discovery

Lab 2 – TCP Echo for CC3000 Example

Objectives

Lab 2 – TCP Echo for CC3000 Example



- ◆ Run the TCP Echo for CC3000 TI-RTOS Example on a Tiva LaunchPad board and CC3000 BoosterPack
- ◆ Connect the CC3000 to an access point using SmartConfig
- ◆ Send/Receive echoed TCP packets between CC3000/LaunchPad and your computer
- ◆ Time: 30 minutes

Lab Procedure

1. Connect the CC3000 BoosterPack to the Tiva LaunchPad board.

Make sure the LaunchPad board is disconnected from the USB cable and/or unpowered.

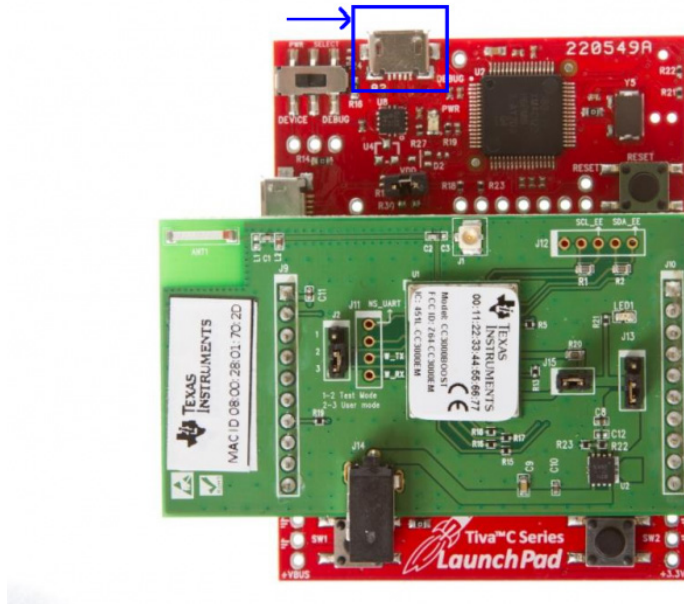
► Gently connect the CC3000 BoosterPack to the J1 and J2 connector pins on the Tiva LaunchPad.

Note the proper orientation, with the CC3000 power supply jack at the “bottom” of the LaunchPad where the two user pushbuttons are located (CC3000 J9 goes to Tiva J1, J10 goes to J2).

2. Check the power supply jumper J13 on CC3000 BoosterPack to power from the LaunchPad board

► Place the jumper at J13 (on the right side of the CC3000 board) at position 1-2 (the top two pins). This will power the board from the USB power supplied by the LaunchPad board. If you want to power the CC3000 with an external supply, this jumper should be at position 2-3 (the lower two pins).

Note: In the picture shown here, the J13 jumper is at positions 2-3. If you see this, make sure to move the jumper to the top two pins 1-2.



► Connect the USB debug port on the LaunchPad board to your laptop.

3. ► Connect your laptop WiFi to SSID “TTOWiFi”.

If your laptop is already connected to a WiFi router, you may need to disconnect, and then connect to “TTOWiFi”. If you’re using Windows 7, you will likely select “Do not connect automatically”. When prompted for “Setup Network?” skip this step. Then choose “Home Network” which is the least restrictive settings for this connection. Once connected, you’ll see the connection in the task tray, but it will show “No Internet Access” since this router is not connected to anything, it’s just being used to establish connections between your laptop and your CC3000.

Install SmartConfig on your phone or tablet

4. Installing SmartConfig on an iOS device (skip this if using Android device).

The iOS application can be installed from the Apple App Store. Search for SmartConfig.

5. Installing on an Android Device (skip this if using iOS device).

- ▶ Download and install the SmartConfig tool for Android on your PC from <http://www.ti.com/tool/smartconfig>.
- ▶ Connect your device to your laptop as a USB Mass Storage Device.
- ▶ Create a folder on your SD card named SmartConfig.
- ▶ Copy SmartConfigCC3X.apk located in C:\TI\CC3000AndroidApp\SmartConfigCC3X\bin on your PC to the new SmartConfig folder on your SD card.
- ▶ Disconnect your Android device from your PC.
- ▶ On your Android device, open the Play Store and find and install the APK Installer of your choice. The following steps are for Easy Installer.
- ▶ Run your APK Installer. It should scan your SD card and find SmartConfigCC3C.
- ▶ Search manually if necessary.
- ▶ Select and install SmartConfig to your phone. You may need to change your security settings to allow installation of non-Market apps. You will now be able to find the SmartConfig application on your device.

Open CCS and Select a Workspace and License

6. Launch CCS.

When the “Select a workspace” dialog appears, ▶select Browse and pick the workspace located at c:\MCUday\workspace.

Do not check the box “Use this as the default and do not ask again”. If at some point you accidentally check this box, it can be changed in CCS.

- ▶ Click OK.

7. Select a CCS License

If you haven't already licensed Code Composer, you may be asked to do so in the next few installation steps. You can do this step manually from the CCS Help menu.

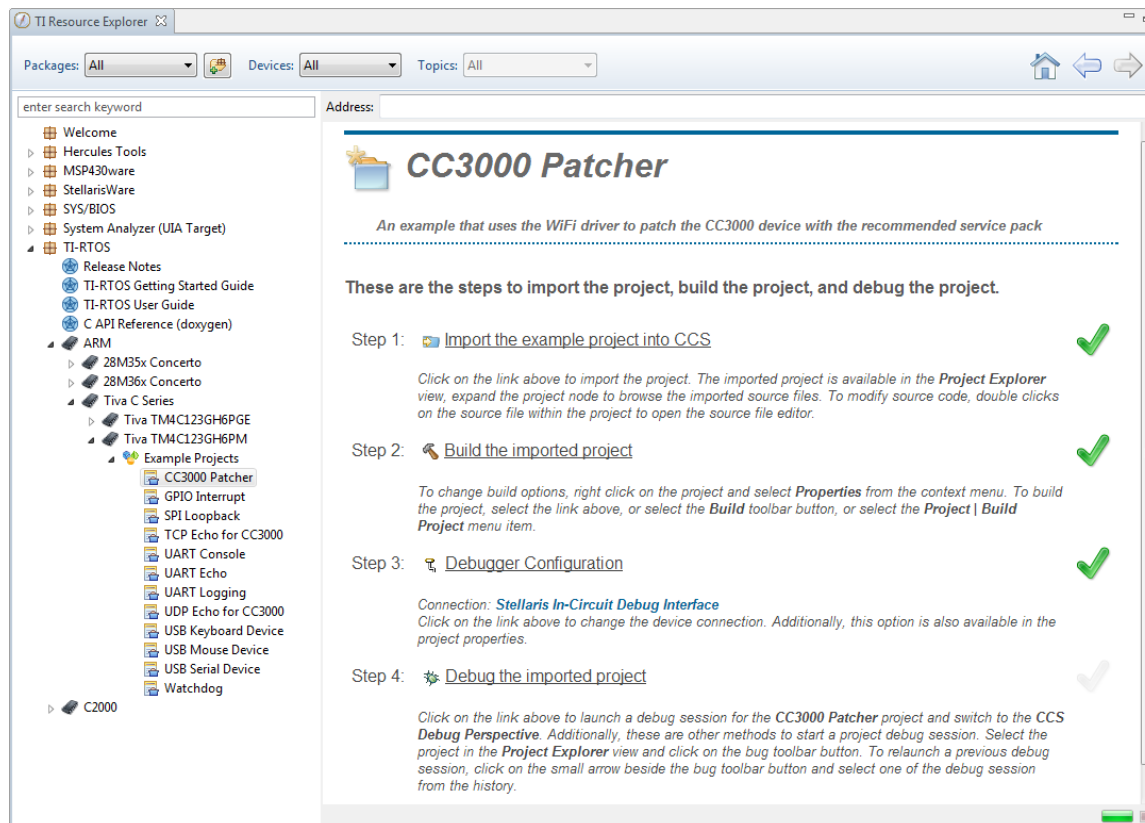
- ▶ Click on *Help* → *Code Composer Studio Licensing Information*.
- ▶ Select the “Upgrade” tab.
- ▶ Select the “Free” license. As long as your PC is connected to the Tiva LaunchPad board, CCS will have full functionality, free of charge.

Run the TI-RTOS CC3000 Patcher Project

8. Use TI Resource Explorer to run the TI-RTOS CC3000 Patcher

If you don't see the TI Resource Explorer window:

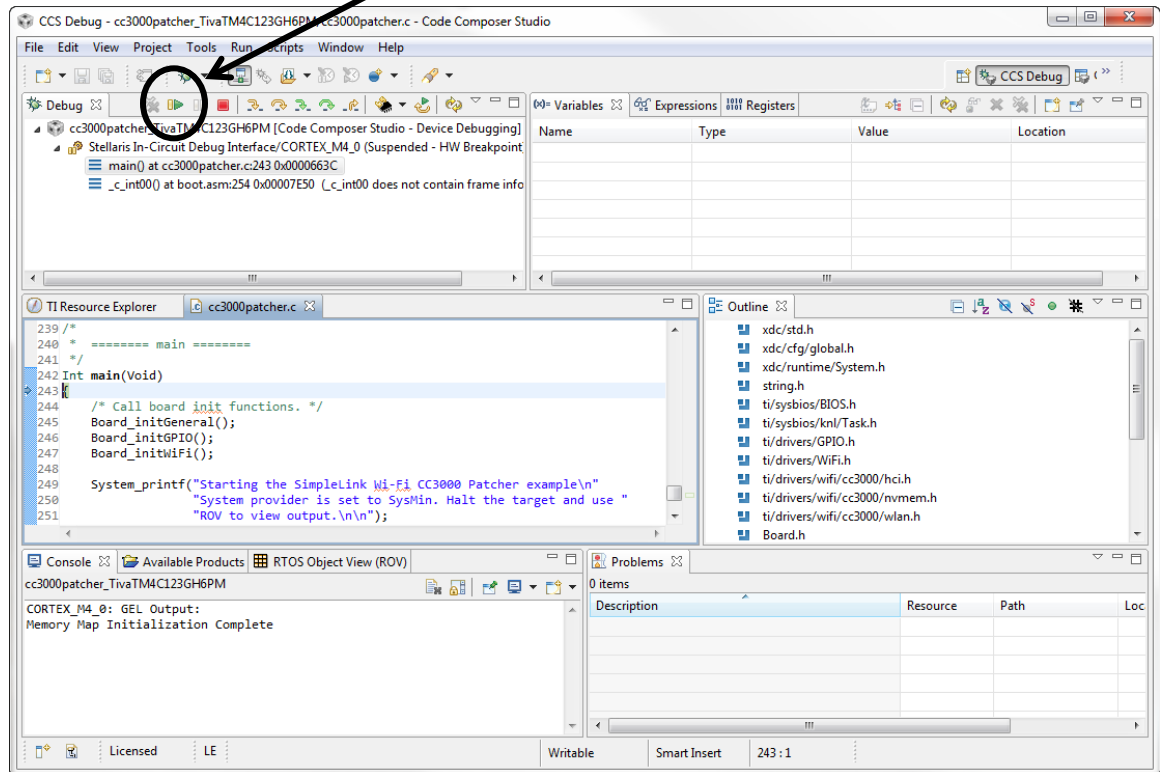
- ▶ Click on the CCS View menu, then click on the top item, *TI Resource Explorer*.
- ▶ In the left-hand pane, expand *TI-RTOS*, then *ARM*, *Tiva C Series*, *Tiva TM4C123GH6PM*, *Example Projects*. Then click on *CC3000 Patcher*. You should see this:



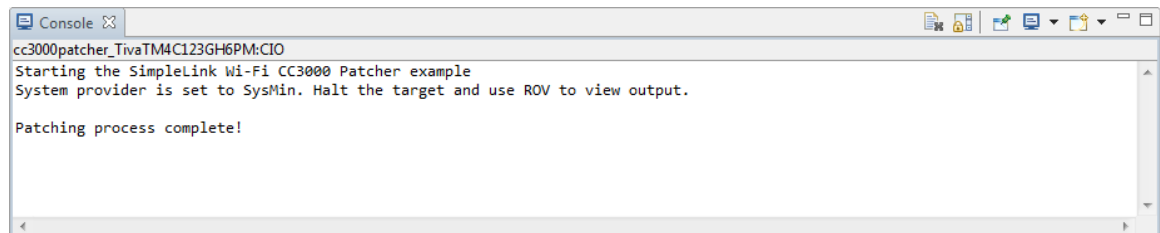
In the right-hand pane, there are four steps to running this project.

- ▶ Click on the link in Step 1 to import the project into CCS. When this completes,
- ▶ Click on the link in Step 2 to build the project. When this completes,
- ▶ Click on the link in Step 3 to select the debugger configuration. From the drop-down menu, select *Stellaris In-Circuit Debug Interface*. This is the JTAG debug interface used on the Tiva LaunchPad board.
- ▶ Click on the link in Step 4 to launch the debugger.

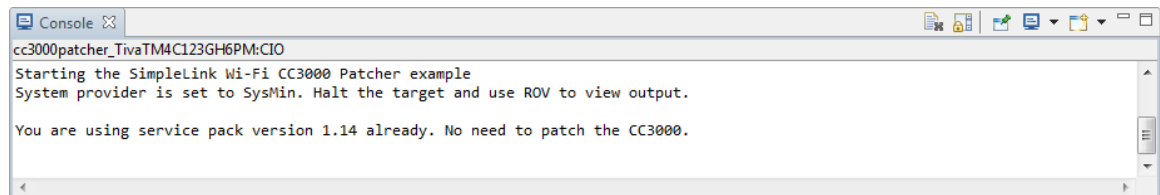
You should see CCS switch from Edit Perspective to Debug Perspective. To run this example, just click on the “Resume” button,



The blue LED should flash while the project runs and updates the EEPROM on the CC3000 device. When complete, the Console window should show this:



If you restart, and run the program again, you should see this:



which indicates you've already patched the CC3000.

9. Terminate the debug session.

- ▶ Click the red Terminate button:



This closes the debug session (and Debug Perspective). CCS will switch back to the *Edit* perspective. You are now completely disconnected from the target.

Run the TI-RTOS TCP Echo for CC3000 Example

10. Use TI Resource Explorer to run the TI-RTOS TCP Echo for CC3000 Example.

Click on the TI Resource Explorer tab to open it again. For this project, navigate to the same Example Projects folder you used in the previous steps.

- ▶ Select TCP Echo for CC3000
- ▶ Go through steps 1, 2, 3, and 4 in the left-pane to import, build, set the debugger connection, and go to the Debugger.

Note: If the Resume button is greyed-out, that means something happened in code execution before main() was reached. You may see some errors in the Console like the following:

```
ti.sysbios.family.arm.m3.Hwi: line 938: E_hardFault: FORCED
ti.sysbios.family.arm.m3.Hwi: line 1050: E_usageFault: UNDEFINSTR: Undefined instruction
Exception occurred in background thread at PC = 0x000084d8.
Core 0: Exception occurred in ThreadType_Task.
```

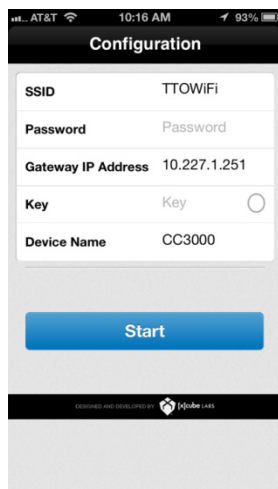
If this happens, click the Pause button. Then in the CCS Menu, click *Run-Reset-System Reset*. You should then be able to use the Resume button

- ▶ Click the Resume button to run your program
- ▶ To enter “SmartConfig” mode, it’s now time to press the SW1 button on the bottom right side on the LaunchPad board. You should see the blue LED start blinking. The CC3000 is waiting for you to connect it to the access point.

11. Use SmartConfig to connect the CC3000 to the wireless router with SSID “TTOWiFi”.

If you don’t have an iOS device or Android device, don’t worry. It is highly likely that at least one person attending the workshop will be able to run this from their phone/tablet. Actually, multiple CC3000s can be connected using only one SmartConfig application:

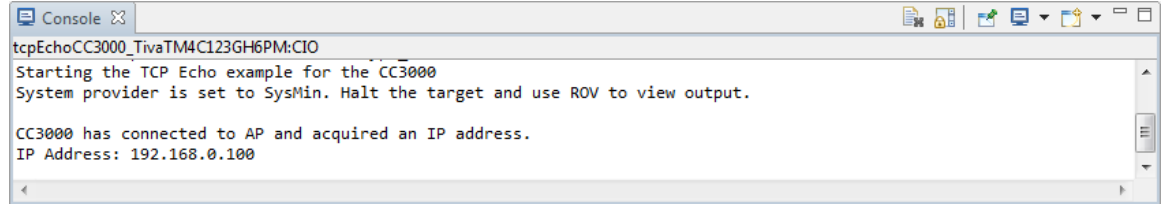
- Each CC3000 in the workshop will be assigned a different address by the router,
 - The IP address will be stored in the EEPROM of the CC3000 for subsequent connections
- ▶ Run the SmartConfig application. You’ll see a screen like the following:



Note that the SSID “TTOWiFi” is entered automatically, and the device name is CC3000. The Gateway IP Address is also “discovered” as the last gateway used by the device. The TTOWiFi router does not have a password or key so you can leave these blank.

► On your iPhone or Android device, click the *Start* button on the SmartConfig application.

When the CC3000 finds the access point, the console window will display the IP address:



```

Console
tcpEchoCC3000_TivaTM4C123GH6PM:CIO
Starting the TCP Echo example for the CC3000
System provider is set to SysMin. Halt the target and use ROV to view output.

CC3000 has connected to AP and acquired an IP address.
IP Address: 192.168.0.100
  
```

After connecting, the blue LED on the LaunchPad should be ON (stops blinking)..

At this point, your laptop and CC3000 are now both connected to TTOWiFi.

► Make a note of the CC3000 IP address (it is displayed in the console window).

_____ (it should be of the form 192.168.0.xxx)

12. Send a TCP packets to the CC3000 and have them echoed back to your computer.

The TI-RTOS example has a Windows (and Linux) command line utility that allows you to Send and Receive packets to/from the CC3000 device. This utility, `tcpSendReceive.exe`, sends a 1024-byte packet to the CC3000 which is echoed back. The utility verifies the first and last bytes in the echoed packet for the correct value. The utility also prints a status line every 1000 packets. This must be run from a CMD window.

► Click on the Windows Start menu, and enter CMD into the text box to open a command window.

► Change the directory using the following CD command (you may copy and paste this by selecting the following line in this .pdf file, do a Ctrl-V (Copy), and in the command window, right-click and select *Paste*:

```
cd c:\ti\tirtos_1_10_00_23\packages\examples\tools
```

If you have a newer version of TI-RTOS, the pathname may be slightly different.

You should now be in the directory where the utility is located.

► Type the command line as follows, but use the IP address that is assigned to your CC3000 from the step above:

```
tcpSendReceive 192.168.0.100 1000 1
```

The arguments are: IPv4 or IPv6 address of CC3000, TCP port number, id

We're using port 1000, and id=1. You could send/receive packets from another id number if you have multiple instances of the utility, so it's used to identify which executable the packets came from).

You should then see the utility start printing a line for every 1000 packets:

```

[id 1] count = 1000, time = 11
[id 1] count = 2000, time = 22
[id 1] count = 3000, time = 33
[id 1] count = 4000, time = 44 ...
  
```

You're done. ► Click the red Terminate button:



(blank page)

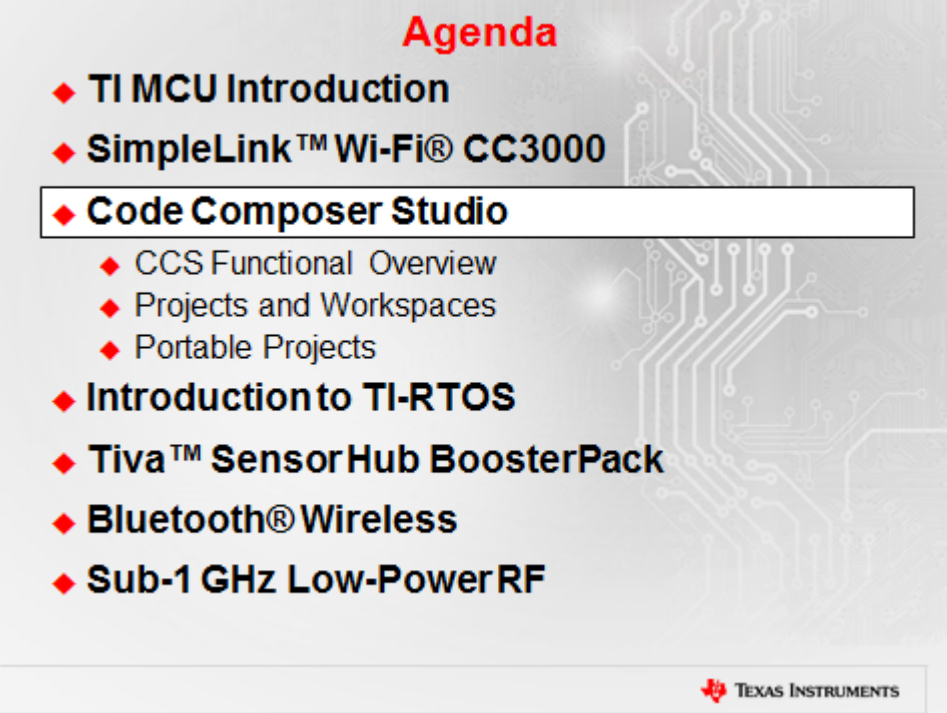
Code Composer Studio

Introduction

This chapter will introduce you to Code Composer Studio (CCS).


In the lab, we will build our first project using CCS and then experiment with some useful debugging features. Even if you have some experience with CCS, this lab is a good review and you will likely learn some new things you don't know.

Learning Objectives



Agenda

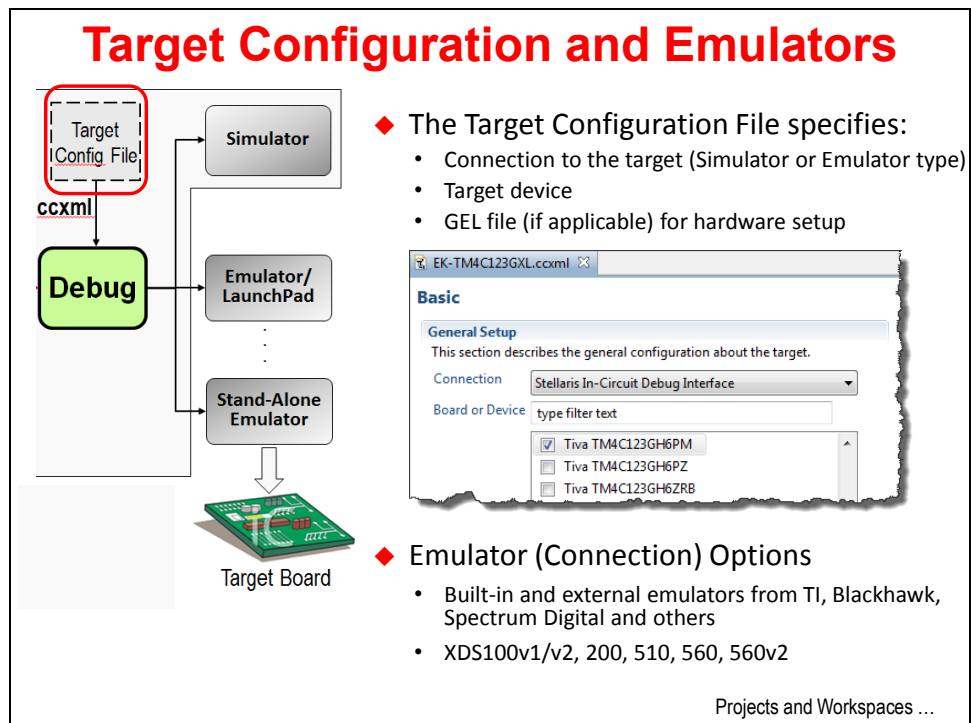
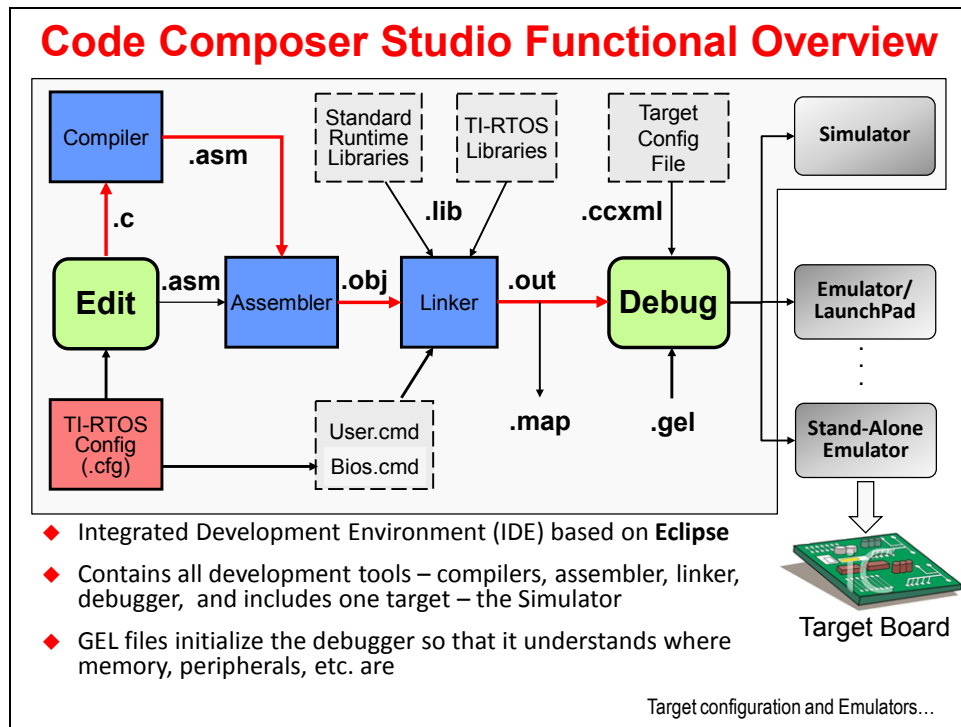
- ◆ TI MCU Introduction
- ◆ SimpleLink™ Wi-Fi® CC3000
- ◆ **Code Composer Studio**
 - ◆ CCS Functional Overview
 - ◆ Projects and Workspaces
 - ◆ Portable Projects
- ◆ Introduction to TI-RTOS
- ◆ Tiva™ SensorHub BoosterPack
- ◆ Bluetooth® Wireless
- ◆ Sub-1 GHz Low-Power RF

 TEXAS INSTRUMENTS

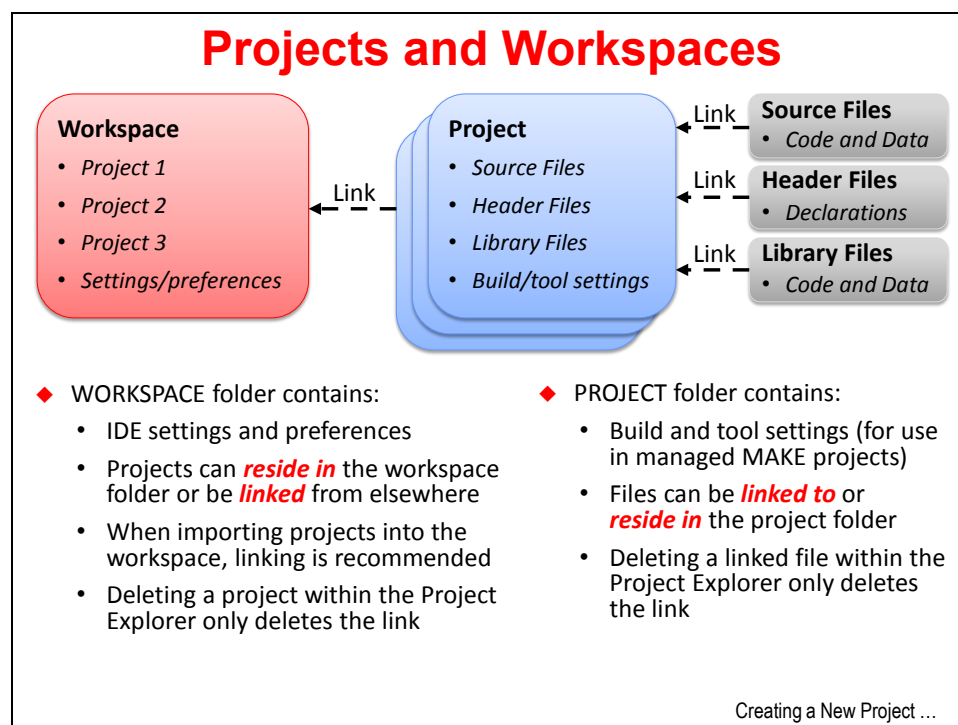
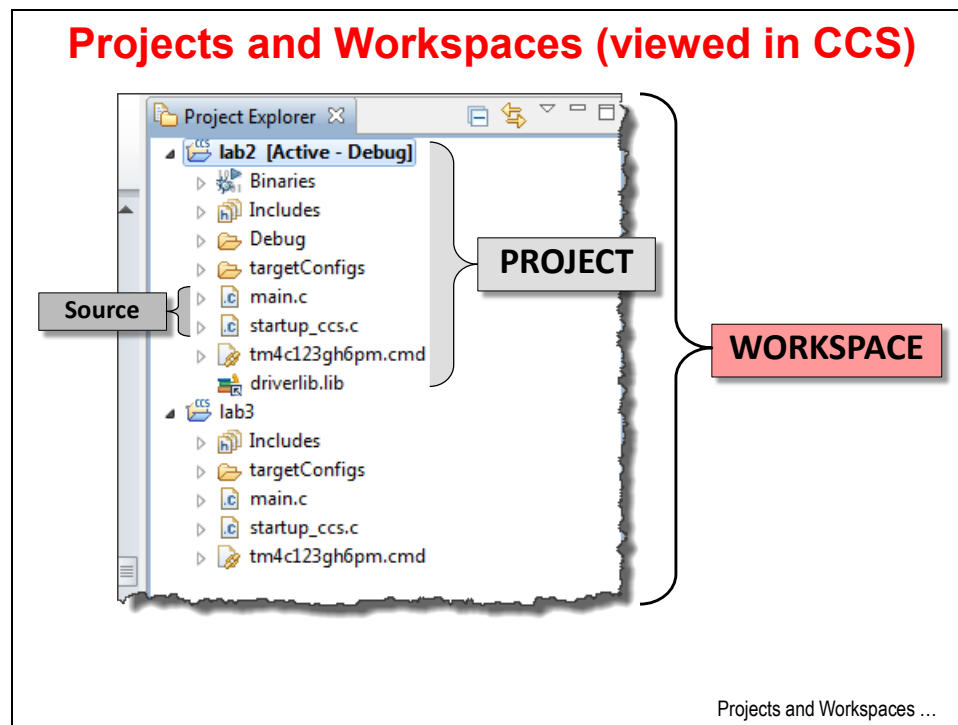
Chapter Topics

Code Composer Studio	3-1
<i>CCS Functional Overview</i>	<i>3-3</i>
<i>Projects and Workspaces</i>	<i>3-4</i>
<i>Portable Projects.....</i>	<i>3-6</i>
<i>Lab 2 - Code Composer Studio</i>	<i>3-9</i>
Lab Procedure.....	3-10
Useful Debugging Tools in CCS	3-18
[Optional] LM Flash Programmer	3-21
[Optional] Creating a bin File for the Flash Programmer	3-23
[Optional] Build, Load, Run Step-by-Step Method	3-24
[Optional] Add Path and Build Variables Manually to a Project	3-27

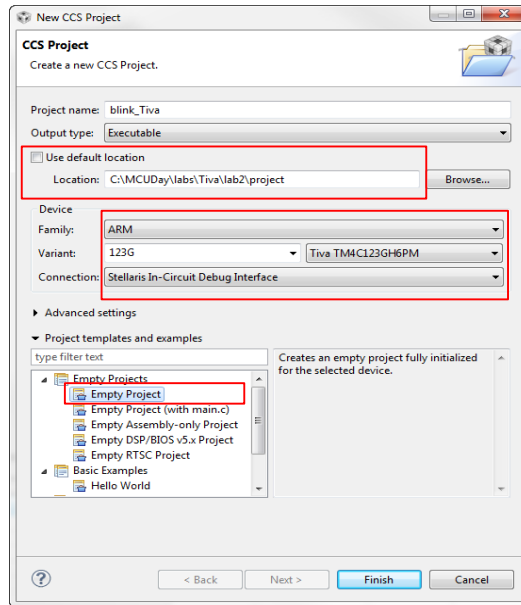
CCS Functional Overview



Projects and Workspaces



Creating a New Project



File→New→CCS Project

(in Edit perspective...)

◆ Project Location

- Default = workspace
- Manual = anywhere you like

◆ Connection

- If target is specified, user can choose "connection" (i.e. the target configuration file)

◆ Project templates

- Empty
- Empty but with a main.c
- Assembly only
- others

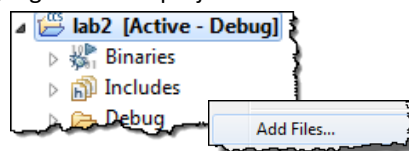
Adding Files to a Project ...

Adding Files to a Project

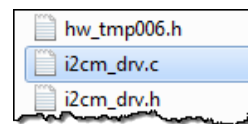
◆ Users can ADD (copy or link) files into their project

- SOURCE files are typically COPIED
- LIBRARY files are typically LINKED (referenced)

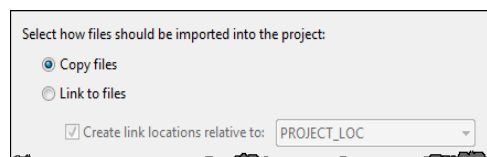
1 Right-click on project and select:



2 Select file(s) to add to the project:



3 Select "Copy" or "Link"



◆ COPY

- Copies file from original location to *project folder* (two copies)

◆ LINK

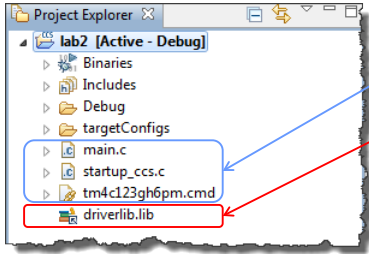
- References (points to) source file in the *original folder*
- Can select a "reference" point – typically PROJECT_LOC

Making a Project Portable ...

Portable Projects

Portable Projects

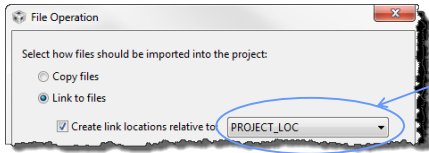
- ◆ **Why make your projects “portable”?**
 - Simplifies project sharing
 - You can easily re-locate your projects
 - Allow simple changes to link to new releases of software libraries



Copied files are not a problem (they move with the project folder)

Linked files may be an issue. They are located outside the project folder via a:

- absolute path, or
- relative path





This is the Path Variable for a relative path. This can be specified for every linked file.

Path and Build Variables ...

Path Variables and Build Variables

- ◆ **Path Variables**
 - Used by CCS (Eclipse) to store the base path for relative linked files
 - Example: **PROJECT_LOC** is set to the path of your project, say
`c:/Tiva_LaunchPad_Workshop/lab2/project`
 - Used as a reference point for relative paths, e.g.
`${PROJECT_LOC}/../files/main.c`
- ◆ **Build Variables**
 - Used by CCS (Eclipse) to store base path for build libraries or files
 - Example: **CG_TOOL_ROOT** is set to the path for the code generation tools (compiler/linker)
 - Used to find #include .h files, or object libraries, e.g.
`${CG_TOOL_ROOT}/include` or `${CG_TOOL_ROOT}/lib`
- ◆ **How are they defined?**
 - **PROJECT_LOC** is created when you create the project
 - **CG_TOOL_ROOT** is created when you install CCS





Why not use PROJECT_LOC?

- ◆ Relative pathnames work fine using **PROJECT_LOC** at first, BUT...
What if you move the project to a different location in the file system?
What if you zip the project and send it to a peer?
The project won't be able to locate the linked file, so...

◆ SOLUTION:

Use a library specific “anchor point” for relative paths
Define your own Path and Build variables that will be relative to:
TivaWare or MSP430Ware installation locations
Now you can move your project anywhere in the file system

Adding Variables ...

Creating Path and Build Variables

- ◆ Create Path and Build Variables by importing from ini files
- ◆ Name of ini file determines its scope

vars.ini or **macros.ini**

```
MSP430WARE_INSTALL = c:\ti\ccsv5\ccsbase\msp430\MSP430ware_1_40_01_19
TIVAWARE_INSTALL = c:\ti\TivaWare_C_Series-1.0
```

Variable's Scope	Import Filename
Project	macros.ini
Workspace	vars.ini

CCSv5 – For More Information

Category: CCS Training

Category: CCS Training

This page provides a collection of training materials.

Contents [hide]

- 1 Getting Started Guides
- 2 Workshops
 - 2.1 CCS Specific Workshops
 - 2.1.1 Fundamentals Workshops
 - 2.1.2 Advanced Workshops
 - 2.2 Device Specific Workshops
 - 2.2.1 MSP430
 - 2.2.2 C2000
 - 2.2.3 Stellaris (ARM Cortex-Mx)
 - 2.2.4 Sitara (ARM Cortex-A8)
 - 2.2.5 DaVinci / ARM Cortex-A8
 - 2.2.6 C6000
- 3 Video Training
- 4 Miscellaneous Presentations
- 5 Modules Library

Modules Library

The goal of the modules library is to provide training materials to facilitate customization and translation of the materials to a particular device but the training focuses on the features.

Module	Video
Overview	Materials
Portable Projects	MSP430
Target Configuration	Materials
Compiler Tips & Tricks	Materials
GRACE	MSP430

Video Training

- CCSv5 Getting Started (Video): This demo goes through a basic project.
- CCSv5 Video Tutorials: Collection of short video tutorials (with audio) on various topics.
- CCS Quick Tips: Collection of short quick video captures (no audio) to demonstrate various features.
- Introduction to CCSv5: An in-depth video (with audio) introducing the CCSv5 in a (of course) informative way. The version of CCS shown is v4 but many of the features are the same.
- C2000 Piccolo Control Law Accelerator Debug with CCS: This video will demonstrate how to use the CCSv5 to debug the C2000 Piccolo.
- C2000 Real-Time Features: This video tutorial covers two very useful features of the C2000 Real-Time.

http://processors.wiki.ti.com/index.php/Category:CCS_Training

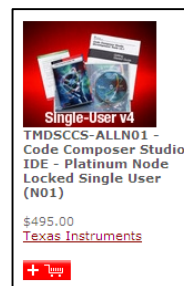
CCSv5 Licensing and Pricing

◆ Licensing

- Wide variety of options (node locked, floating, time based)
- All versions (full, DSK, free tools) use the same image
- Updates readily available online

◆ Pricing

- Includes FREE options noted below
- Annual subscription - \$99 (\$159 for floating license)



Item	Description	Price	Annual
Platinum Eval Tools	Full tools with 90 day limit (all EMU)	FREE	
Platinum Bundle	XDS100 use (EVM or simulator)	FREE *	
Platinum Node Lock	Full tools tied to a machine	\$495/\$445 **	\$99
Platinum Floating	Full tools shared across machines	\$795	\$159
MSP430 Code-Limited	MSP430 (16KB code limit)	FREE	


* recommended option: purchase Development Kit, use XDS100v1-2, & Free CCSv5

** \$495 includes DVD, \$445 is download only

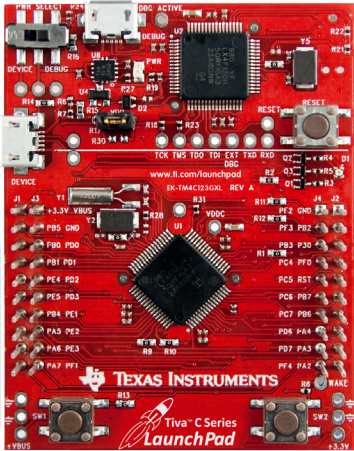
Lab 3 - Code Composer Studio

In this lab, we will create a project that contains one simple source file, `main.c`, which has the necessary code to blink an LED on Tiva LaunchPad board. It simply makes a few calls to a few library functions to set up the pins and then toggle them.

Lab 3 – Code Composer Studio



USB Emulation Connection
↓



- ◆ Create a new CCS project for Tiva LaunchPad
- ◆ Experiment with some CCS features
- ◆ Optional: Re-program the flash to run the Quickstart Application
- ◆ Time: 45 minutes

Lab Procedure

Folder Structure for the Labs

1. Browse the directory structure for the workshop labs

- Using Windows Explorer, locate the following folder:

`c:\MCUday\labs\Tiva`

In this folder, you will find all the lab folders for the workshop. If you don't see this folder on your `c:\` drive, check to make sure you have installed the workshop lab files. Expand the `c:\MCUday\labs\Tiva` folder and you'll notice that there are sub-folders for each of the labs. The `\files` folder contains the "starter files" you may need to create each project. The `\project` folder will contain your project files.

Note: When you create a project, you have a choice to use the "default location" which is the CCS workspace or to select another location. In this workshop, we will not be using the workspace for the project files, rather, we'll use the folder where you installed the lab files, `c:\MCUday\labs`. The workspace will only contain CCS settings, and links to the projects we create or import.

Open CCS

2. Launch CCS

- Launch CCS. When the "Select a workspace" dialog appears, select Browse and pick the workspace located at:

`c:\MCUday\workspace.`

Do not check the box "*Use this as the default and do not ask again*". If at some point you accidentally check this box, it can be changed in CCS.

- Click OK.

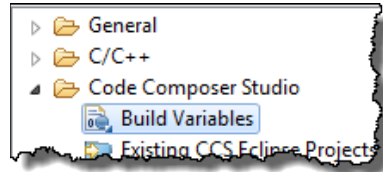
3. Close TI Resource Explorer and/or Grace

When the "TI Resource Explorer" and/or "Grace" windows appear, close these windows using the "X" (Close) icon. We won't need these windows during this lab.

Add Workspace Variables

4. Import the file `vars.ini` to set Path Variable and Build Variable

- Select *File* → *Import*, then expand the *Code Composer Studio* category, click on *Build Variables* (as shown):



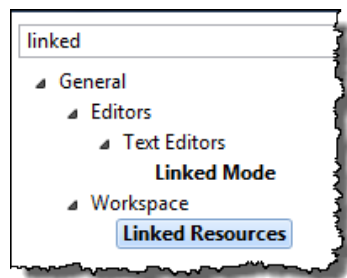
- Click *Next* and browse to the location of `vars.ini`:

`c:\MCUday\labs\Tiva\vars.ini`

- Click *Open*, then click *Finish*.

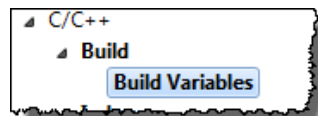
5. Verify that the path and build variables were set.

- Select *Window* → *Preferences*. When the dialogue appears, ► type “*linked*” into the filter field as shown – then click on *Linked Resources*:



This displays all of your WORKSPACE level path variables. You should see the Path variable for `TIVWARE_INSTALL`.

- Type “*build*” into the filter area and click on *Build Variables* as shown:



Here is where you can find your WORKSPACE level build variables. You should see the Build variable for `TIVWARE_INSTALL`.

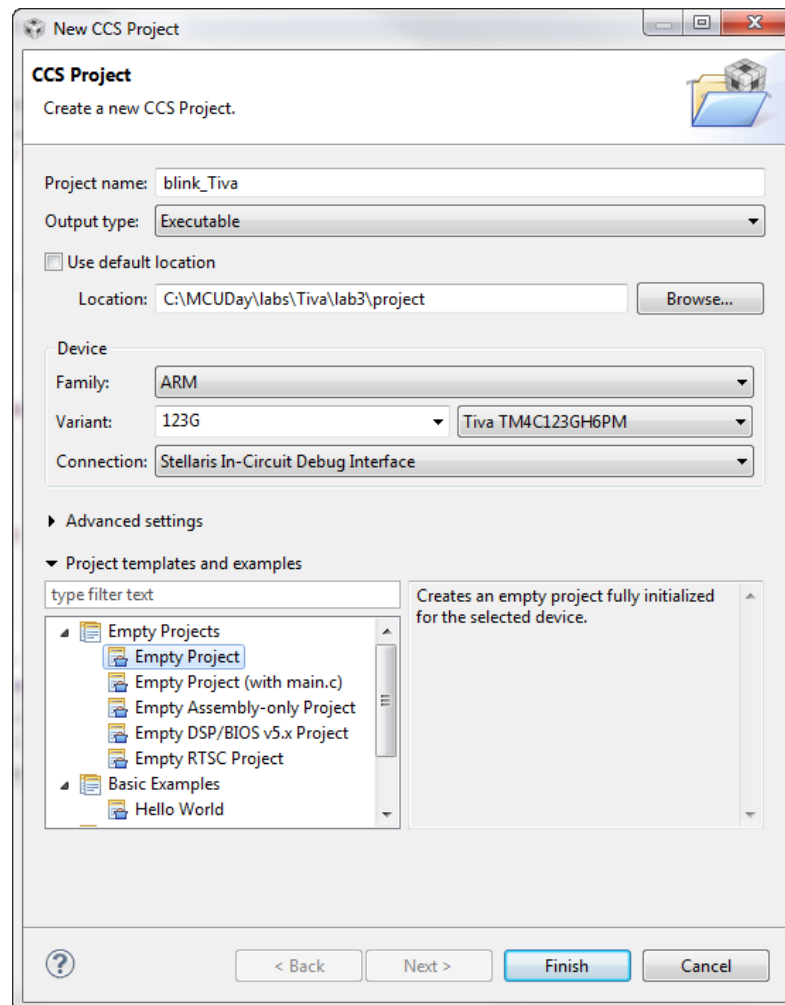
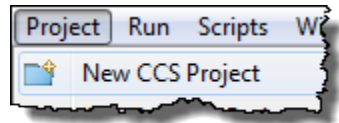
- Click *Cancel*.

Note: Any project in this workspace can now use these variables. If you change workspaces, you will have to re-import `vars.ini` to set these variables again. If your tools installation changes, you'll have to edit `vars.ini` and re-import. So be careful.

Create a New Project

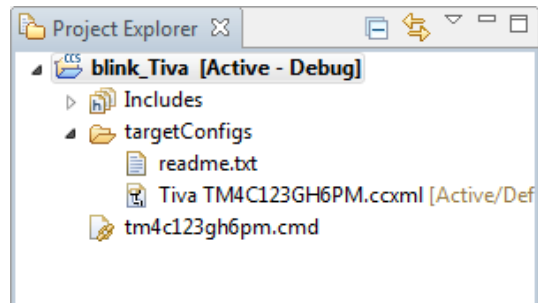
6. To create a new project,

- select *Project* → *New CCS Project*:



- For the project name, type *blink_Tiva*
- Uncheck the box “*Use default location*” click *Browse...* button to select the folder:
`c:\MCUDay\labs\Tiva\lab3\project`
- Device family: *ARM*
- Variant: type *123G* in the filter text field, then select *Tiva* ► For Connection: choose *Stellaris In-Circuit Debug Interface*
- Choose *Empty Project* and then click *Finish*.

Your project should look something like this.



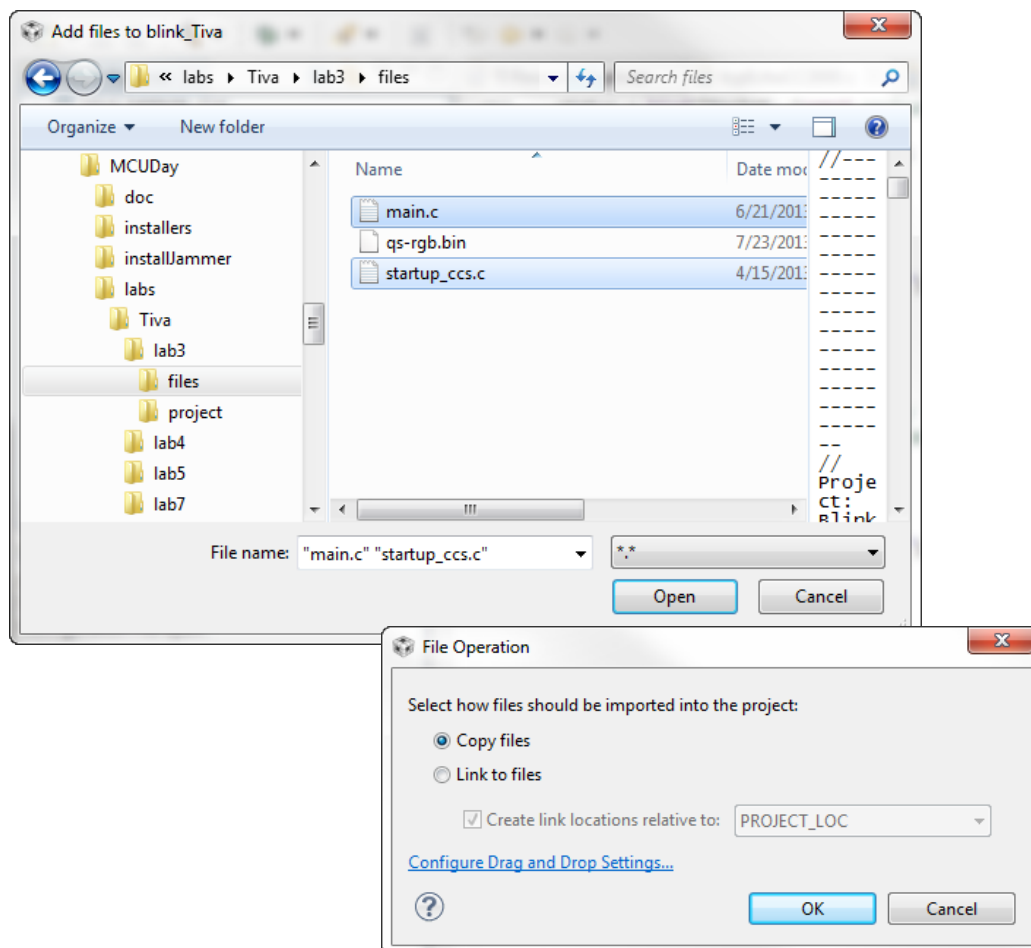
Add files to your project

7. Add (copy) the C file(s)

- Select *Project* → *Add Files...* ► Navigate to the folder:

`c:\MCUday\labs\Tiva\lab3\files`

Select the `main.c` and `startup_ccs.c` files and click *Open*. Then select *Copy Files* and click *OK*.

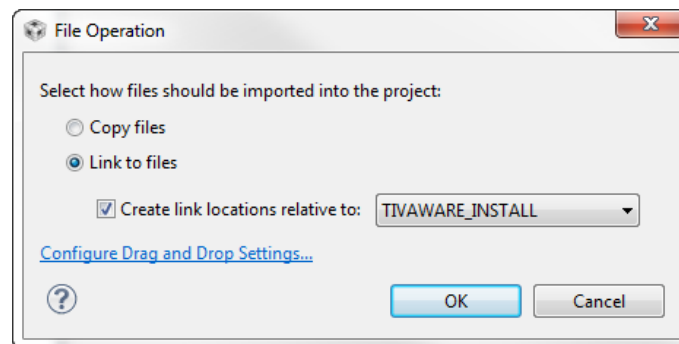


8. Link the TivaWare driverlib.lib file to your project

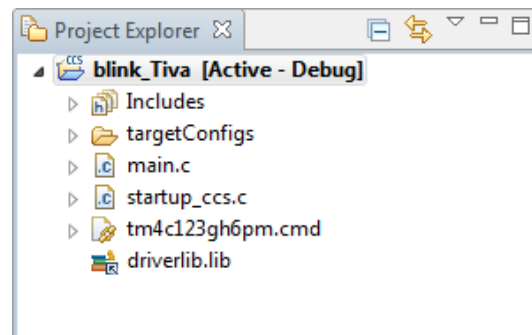
- Select *Project-Add Files...* ► Navigate to select:

C:\TI\TivaWare_C_Series-1.0\driverlib\ccs\Debug\driverlib.lib

Use the *TIVAWARE_INSTALL* path variable you created earlier. This means that the LINK (or reference to the library) file will be RELATIVE to the location of the TivaWare installation. If you hand this project to someone else, they can install the project anywhere in the file system and this link will still work. If you choose *PROJECT_LOC*, you would get a path that is relative to the location of your project and it would require the project to be installed at the same “level” in the directory structure. Another advantage of this approach is that if you wanted to link to a new version, say TivaWare_C_Series-1.1, all you have to do is modify the variable to the new folder name.

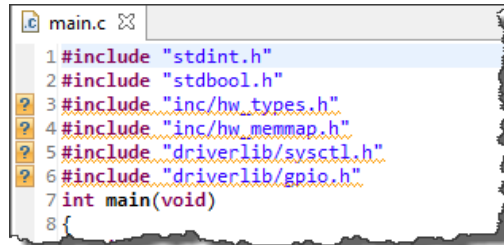


Your project should now look something like this:



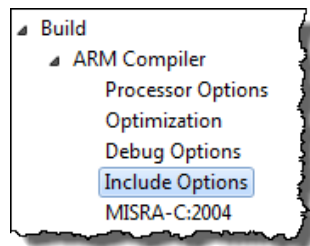
9. Add INCLUDE search paths for the header files

► Open your `main.c` file by double-clicking on the filename. You may see “?” warnings in the left margin which indicate “unresolved inclusion”. Until now, you haven’t told the project where to find these `.h` header files.



► Right-click on your project and select *Properties*.

► Click on *Build* → *Compiler* → *Include Options* (as shown):

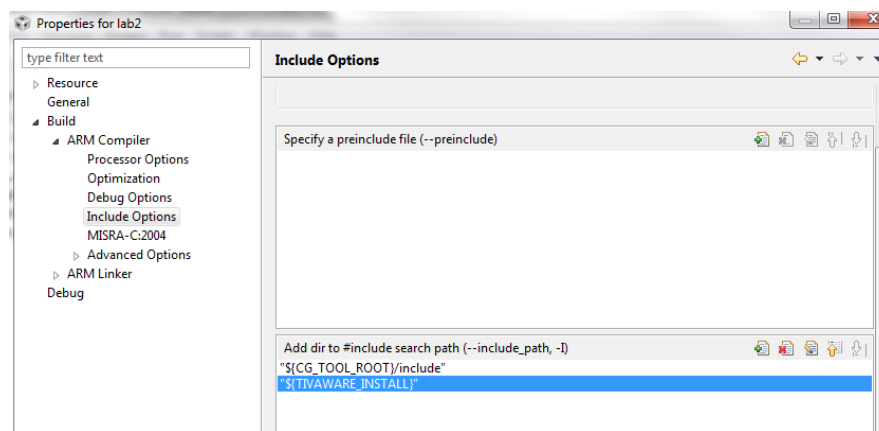


So, which variable knows where your include files are? The build variable `TIVWARE_INSTALL`

► In the lower-right panel, click the “+” sign next to *Add dir #include search path* and add the following path using the build variable you created earlier. Place the variable name inside braces, after the \$ as shown. To prevent errors, you may simply copy the following line (Ctrl-C), and paste (Ctrl-V) into the include search path:

```
{TIVWARE_INSTALL}
```

► Click Ok.



► Click Ok, and now you should see those “?” in `main.c` disappear. You’ve solved the problem.

Explore the Blink LED Code

10. Explore code in main.c.

In this lab, we are using a simple blink LED program – the famous “hello world” for MCUs. The goal in this workshop is to keep the code very simple and focus on concepts where you will be able to learn valuable skills without huge/complex code getting in the way.

► Open `main.c` for editing and peruse the whole file. You will see the header files, prototypes and global variables used.

```
//-----  
// main()  
//-----  
void main(void)  
{  
  
    hardware_init();           // init hardware via Xware  
  
    while(1)                   // forever loop  
    {  
        ledToggle();           // toggle LED  
  
        delay();               // create a delay of ~1/2sec  
  
        i16ToggleCount += 1;    // keep track of #toggles  
    }  
  
}
```

First we perform some setup for the hardware to blink the LED by calling `hardware_init()`. In the `while(1)` loop, we have three steps:

- Toggle the LED
- Delay function
- Increment `i16ToggleCount` global variable (we'll use this in a few ways later)

Build and Run Your Project

11. Build your project using the Build button and check for errors.

At this point, it is a good time to build your code to check for any errors before moving on.

► Click the *Build* button:



If you have any errors, try to fix them. Ask your instructor for help if you cannot get your project to build successfully.

12. Start the Debugger, connect to your target, program the flash, and run to main().

► Click the *Debug* button:



You will see the CCS perspective switch to the Debug Perspective and the flash memory on the device will be programmed with the .out file you created. To run your program,

13. Run your program.

Now, it's finally time to run your code. ► Click the *Resume* button:



The LED on your target board should blink. If not, attempt to solve the problem yourself for a few minutes ... then, ask your instructor for help.

To stop your program running, ► click the *Suspend* button:

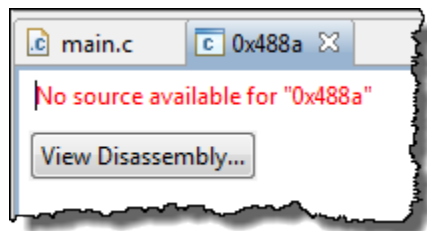


Hint: *Suspend* button is different than *Terminate* button !!!



If you click the Terminate button, the debugger – and your connection to the target – will be closed. If you're debugging and just want to view a variable or memory, you will have to start all over again. Yes, this is very irritating. Remember to **pause** and think, before you halting your program.

Do you end up with a file that cannot be displayed?



This error occurs because the source file for the library or object has not been added to our project. This is entirely acceptable since we really don't want to see inside the delay routine.

Useful Debugging Tools in CCS

Breakpoints and Single-Stepping

14. Add a breakpoint in your code.

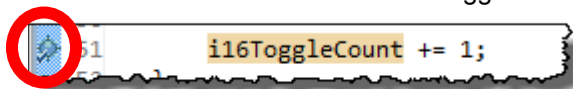
Breakpoints are very useful debug tools. Besides helping us to halt execution within a specific source file (to solve our previous problem), they also allow us to halt in a location where we may want to view a variable's value.

Note: Breakpoints can only be added while the target is suspended. You can set them while the target processor is running.

Let's add breakpoint and then run to it.

► Click into the `main.c` file, if you're not already halted there.

In the column next to the increment of `toggleCount`, ► double-click to add a breakpoint:

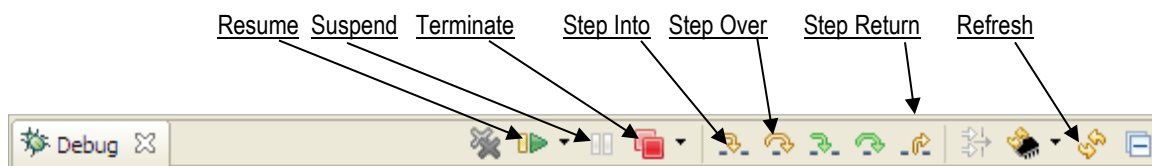


► Click the *Resume* button. The PC should stop at this line. This should happen each time you hit Resume button.

15. Single-step your program.

Breakpoints are handy, but sometimes you want to view code execution after every line of code – doing this with breakpoints would be very tedious. This is where single-stepping a program comes in handy.

With the program halted, click the **Step Over (F6)** toolbar button (or tap the F6 key):



Notice how one line of code is executed each time you click *Step Over*; in fact, this action treats function calls as a single point of execution – that is, it steps *over* them. On the other hand *Step Into* will execute a function call step-by-step – go *into* it. *Step Return* helps to jump back out of any function call you're executing.

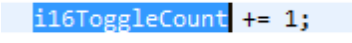
Watch Expressions, Memory Browser, and Registers

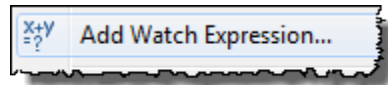
16. Hover over a variable to view it's information & value.

- Hover over the variable *i16ToggleCount* in *main()*. After a few seconds, you should see an information box pop up and show the value.

What is the value? _____

17. View/Watch variables.

- Double-click on *i16ToggleCount* in *main()* to select the variable. 
- Right-click on the selected variable and choose:



- Click Ok. Do you see *i16ToggleCount* in the list? What is the value? _____
Is it the same as the previous step?

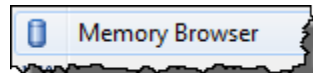
Hint: If the variable is not selected when you right-click and choose “Add Watch Expression...”, you will have to type the name into the dialog – which is not as easy as selecting the variable first.

Note that you can add any expression to a Watch entry. For example, this means we could have the watch window show the value of: `i16ToggleCount * 3`

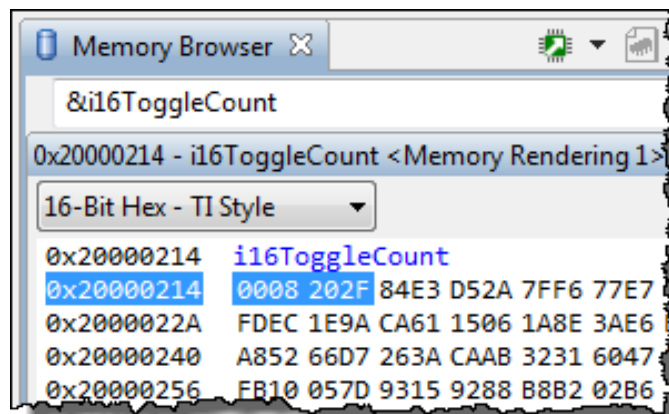
18. Viewing memory...

Does *i16ToggleCount* live somewhere in memory? Yes. You can see the actual address in the expressions view. But let's go see it in a Memory Browser window.

- Select *View → Memory Browser*:



- Type “&*i16ToggleCount*” into the memory window to display *i16ToggleCount* in memory:



What does the “&” mean?

What happens if you forget to use it? (Yes, you see it's address, rather than it's value.)

- Try changing the memory windows format from:

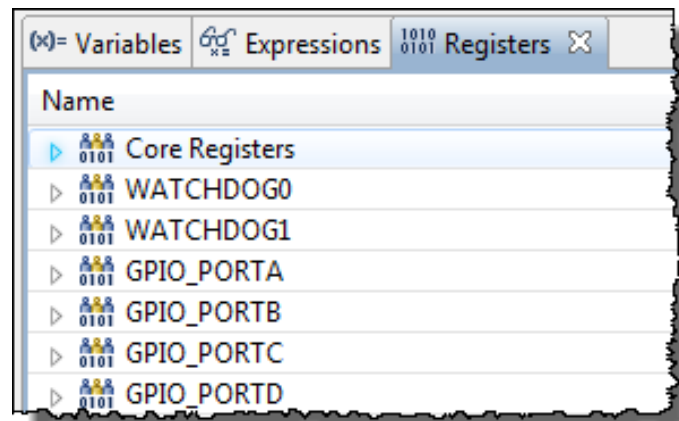
“16-bit Hex – TI Style”

What changes when you do this?

19. Viewing CPU registers...

- Select *View* → *Registers* and notice you can see the contents of all of the registers in your target's architecture. Sometimes quite handy when debugging.

Note: You can only view Tiva registers for peripherals that have been enabled.



Terminate

20. Terminate the debug session.

OK, this time we really want to terminate our debug session. (This way, we can start up the debugger again ... the easy way.)

- Click the red *Terminate* button:



This closes the debug session (and Debug Perspective). CCS will switch back to the *Edit* perspective. You are now completely disconnected from the target.

21. Close the Project.



You're done. Please let your instructor know that you have completed the main part of this lab.

[Optional] LM Flash Programmer

LM Flash Programmer is a standalone programming GUI that allows you to program the flash of a Tiva device through multiple ports. Creating the files required for this is a separate build step in Code Composer Studio that is shown on the next page. If you have not done so already, install the LM Flash Programmer onto your PC.

Make sure that Code Composer Studio is not actively running code in the CCS Debug perspective... otherwise CCS and the Flash Programmer may conflict for control of the USB port.

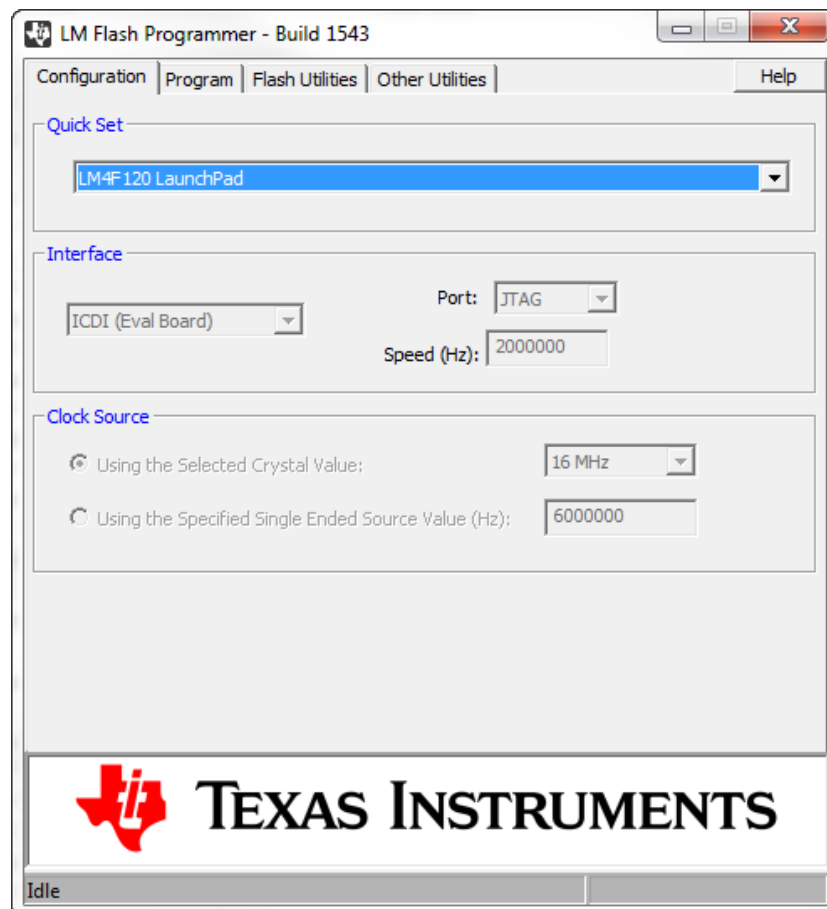
22. Open LM Flash Programmer



There should be a shortcut to the LM Flash Programmer on your desktop, double-click it to open the tool. If the shortcut does not appear, go to *Start → All Programs → Texas Instruments → Stellaris → LM Flash Programmer* and click on *LM Flash Programmer*.

Your evaluation board should currently be running the lab3 application. If the User LED isn't blinking, press the RESET button on the board. We're going to program the original application back into the TM4C123GH6PM.

► Click the Configuration tab. Select the LM4F120 LaunchPad from the Quick Set pull-down menu under the Configuration tab. See the user's guide for information on how to manually configure the tool for targets that are not evaluation boards.

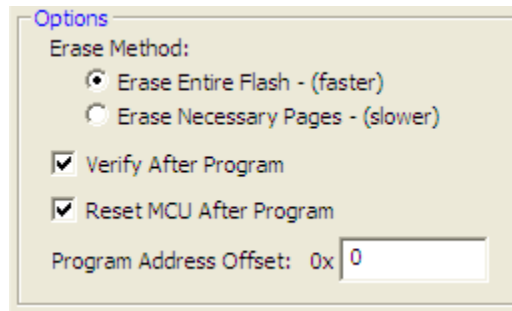


23. Click the Program Tab, then click the Browse button and navigate to:

```
c:\TI\TivaWare_C_Series-1.0\examples\boards\ek-tm4c123gx1\
qs-rgb\ccs\Debug\qs-rgb.bin
```

This is the application that was programmed into the flash memory of the TM4C123GH6PM during the evaluation board assembly process.

Note that there are applications here which have been built with each supported IDE. Make sure that the following checkboxes are selected:



24. Click the Program button

You should see the programming and verification status at the bottom of the window. After these steps are complete, the quickstart application should be running on your evaluation kit.

25. Close the LM Flash Programmer

[Optional] Creating a bin File for the Flash Programmer

If you want to create a `.bin` file for use by the stand-alone programmer in any of the Tiva labs in this workshop or in your own project, use these steps below. Remember that the project will have to be open before you can change its properties.

26. Set Post-Build step to call “tiobj2bin” utility

► In CCS Project Explorer, right-click on your project and select *Properties*. On the left, click Build and then the *Steps* tab. Paste the following commands into the *Post-build steps Command* box.

Note: The following four “lines” should be entered as a single line in the *Command* box. To make it easier, we included a `.txt` file you can copy-paste. Navigate to `c:\MCUday\labs\Tiva\postbuild.txt` to find the complete command line.

```
"${CCS_INSTALL_ROOT}/utils/tiobj2bin/tiobj2bin"  
"${BuildArtifactFileName}" "${BuildArtifactFileName}.bin"  
"${CG_TOOL_ROOT}/bin/armofd" "${CG_TOOL_ROOT}/bin/armhex"  
"${CCS_INSTALL_ROOT}/utils/tiobj2bin/mkhex4bin"
```

27. Rebuild your project

This post-build step will run after your project builds and the `.bin` file will be in the `c:\MCUday\labs\Tiva\labx\project\debug` folder. You can access this `.bin` in the CCS Project Explorer in your project by expanding the Debug folder.

If you try to re-build and you receive a message “`gmake: Nothing to be done for 'all'.`” this indicates that no files have changed in your project since the last time you built it. You can force the project to build by first right-clicking the project and then select *Clean Project*. Now you should be able to re-build your project which will run the post-build step to create the `.bin` file.

[Optional] Build, Load, Run Step-by-Step Method

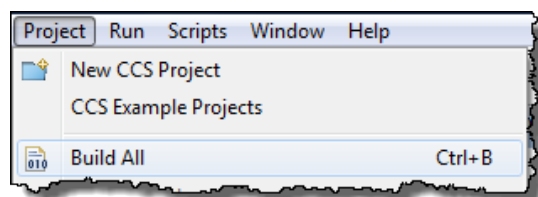
There are five steps required to run code within CCS:

- Build (Compile, Assemble, Link) your code.
- Launch the debugger.
- Connect to your target board.
- Load your program.
- Click the *Resume* button.

Launching the Debugger step-by-step

28. Build your project and fix any errors.

- Build your project by using *Project* → *Build All* or Ctrl-B:



- Or, by clicking *Build*:

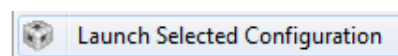


- Fix any errors that occur.

29. LAUNCH a debug session.

- Select *View* → *Target Configurations*. Expand the *Projects* folder, expand the project name, e.g. *blink_Tiva*, expand *targetConfigs* and you'll see the target configuration file (.ccxml).

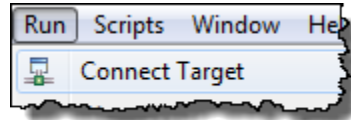
- Right-click on this target config file and select:



Your perspective will change to the *Debug* perspective and a few lines may be sent to the *Console* window.

30. CONNECT to your target board.

- Connect to the Target via *Run – Connect Target*:



- Or via the *Connect Target* button:



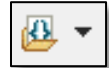
If your TargetConfig specifies a GEL file, this is when it runs – so you may see a few more comment lines in the *Console* window. If the error “cannot connect to target” appears, the problem is most likely due to:

- wrong board/target config file or both – i.e. board does not match the target config file
- wrong target bad/wrong GEL file (rare, but it can happen)
- bad USB cable
- Windows USB driver is incorrect – or just didn't get enumerated correctly

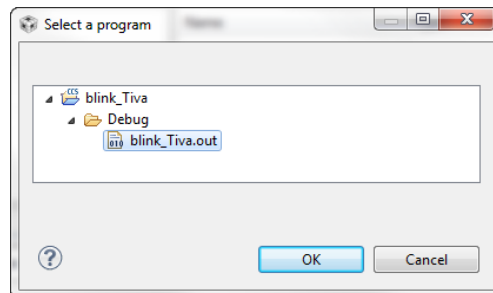
Hint: If you're using the “easy one button” approach to loading your program, if see an error, we recommend going back and launching the debugger using these three discrete steps. It can often help you deduce when/where the problem occurred.

31. Load your program.

- Load your program via *Run → Load → Load Program* or via the *Download* button:



When the dialog appears, ► select *Browse Project* and navigate to the `\project\Debug\blink_Tiva.out` file.



Hint: The reason to use *Browse Project* is that the default `.out` file that appears is often NOT the `.out` file you want.

If you get into the habit of using *Browse Project*, it will default to the active project which is usually what you want.

- Select your `.out` file and click *Ok* twice. Your program will now download to the target board and the PC will auto-run to `main()` and stop.

32. Run your program.

► Click the *Resume* button:



The LED on your target board should blink. If not, attempt to solve the problem yourself for a few minutes ... then, ask your instructor for help.

To stop your program running, ► click the *Suspend* button:



Hint: **Suspend** is different than **Terminate** !!!

If you click the Terminate button, the debugger – and your connection to the target – will be closed. If you're debugging and just want to view a variable or memory, you will have to start all over again. Yes, this is very irritating. Remember to **pause** and think, before you halting your program.

Terminate

33. Terminate the debug session.

OK, this time we really want to terminate our debug session. (This way, we can start up the debugger again ... the easy way.)

► Click the red *Terminate* button:



This closes the debug session (and Debug Perspective). CCS will switch back to the *Edit* perspective. You are now completely disconnected from the target.

[Optional] Add Path and Build Variables Manually to a Project

If you recall the lecture, these variables are used for:

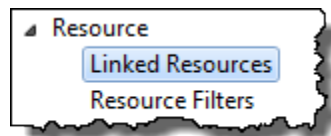
- Path variable – when you ADD (link) a file to your project, you can specify a “relative to” path. The default is `PROJECT_LOC` which means that your linked resource (like a `.lib` file) will be linked relative to your project directory.
- Build variable – used for items such as the search path for include files associated with a library – i.e. it is used when you build your project.

Variables can either have a `PROJECT` scope or a `WORKSPACE` scope.

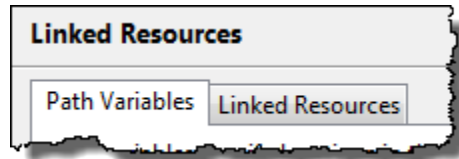
In the earlier part of this lab, we defined these variables by importing the `vars.ini` file. This is the easiest way to define `WORKSPACE` level variables that apply to every project in the workspace. Using the steps below, we can define `PROJECT` level variables.

34. Adding a Path Variable

To add a path variable, ► Right-click on your project and select *Properties*. ► Expand the *Resource* list in the upper left-hand corner as shown and click on *Linked Resources*:



You will see two tabs on the right side – *Path Variables* and *Linked Resources*:

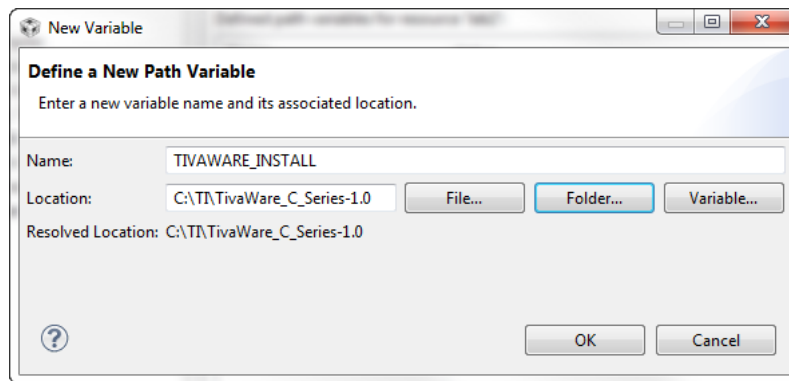


In the *Path Variables* tab, notice that `PROJECT_LOC` is one of them and will display as the default path variable for linked resources in your project.

We want to add a *New* variable to specify exactly where you installed TivaWare.

- Click *New*
- For the *Name*, type `TIVAWARE_INSTALL`
- For the *Location*, click the *Folder...* button and navigate to either the TivaWare folder.

```
c:\ti\TivaWare_C_Series-1.0
```

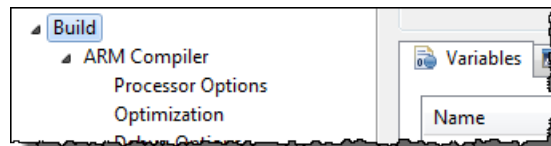


- Click Ok. When finished, you should see your new path variable listed.

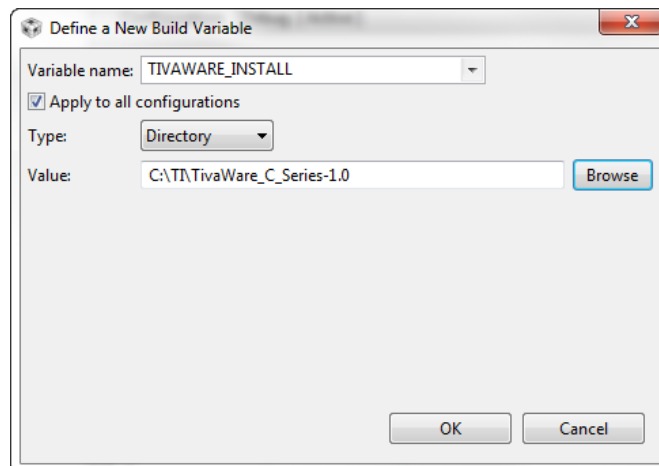
35. Adding a Build Variable

Now let's add a build variable that we will use in the include search path for the INCLUDE files associated with the TivaWare driver libraries.

- Right-click on your project, select *Properties*. ► Click on *Build* and then the *Variables* tab:



- Click the *Add* button. When the dialogue appears, specify the SAME variable name as before, TIVWARE_INSTALL
- Select Type: *Directory* so the Browse button pops up and then browse to your correct installation directory:



- Make sure *Apply to all configurations* is checked (that way, if you change the build configuration from *Debug* to *Release*, you can use this variable). ► Click Ok.

You should now see your new build variable in the list:

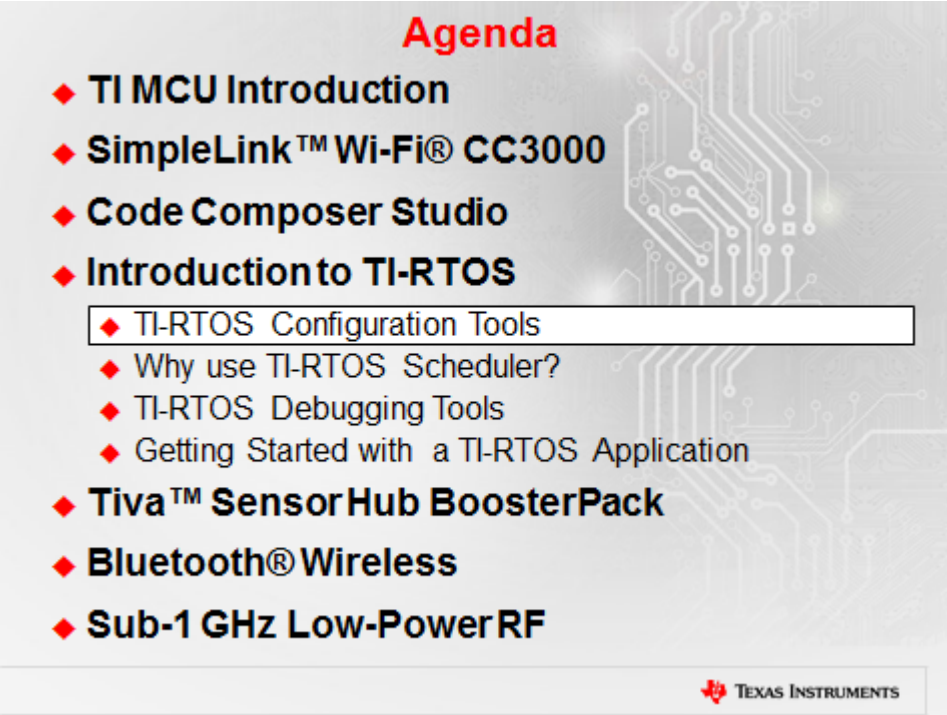
- Click Ok.

Introduction to TI-RTOS

Introduction


This chapter provides an introduction to the basic concepts of TI-RTOS, a real-time operating system for TI MCUs. The focus is on the scheduling kernel, known as TI-RTOS Kernel. A case is made for WHY an RTOS is needed and how it can help users develop system code and perform common tasks using services provided.

Learning Objectives



Agenda

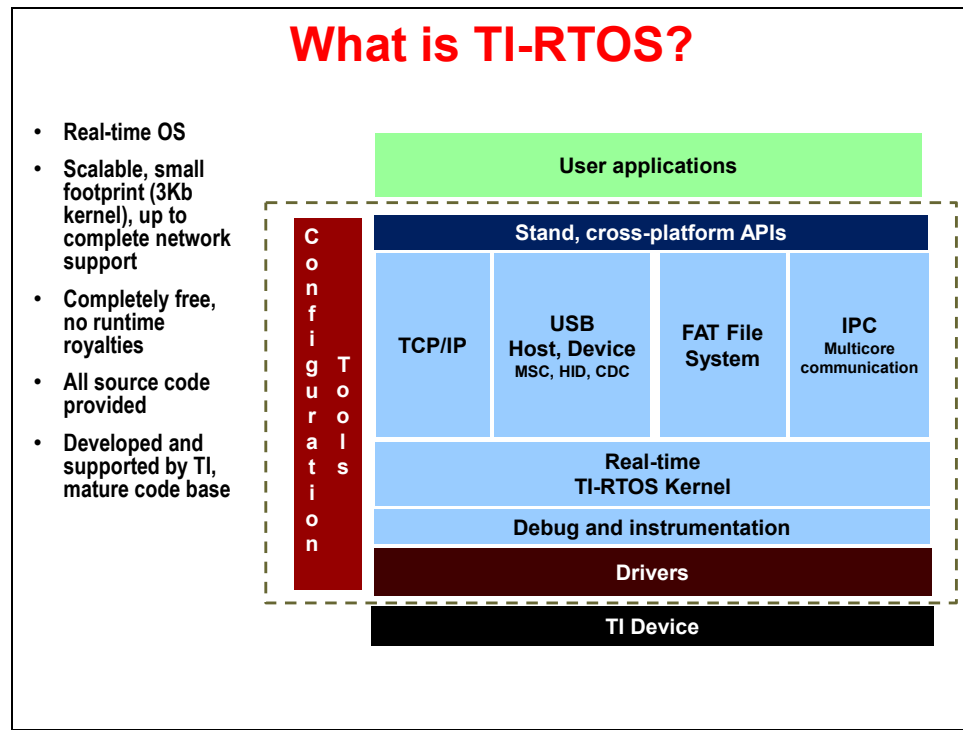
- ◆ **TI MCU Introduction**
- ◆ **SimpleLink™ Wi-Fi® CC3000**
- ◆ **Code Composer Studio**
- ◆ **Introduction to TI-RTOS**
 - ◆ TI-RTOS Configuration Tools
 - ◆ Why use TI-RTOS Scheduler?
 - ◆ TI-RTOS Debugging Tools
 - ◆ Getting Started with a TI-RTOS Application
- ◆ **Tiva™ SensorHub BoosterPack**
- ◆ **Bluetooth® Wireless**
- ◆ **Sub-1 GHz Low-Power RF**

 **TEXAS INSTRUMENTS**

Chapter Topics

Introduction to TI-RTOS	4-1
<i>Introduction to TI-RTOS.....</i>	<i>4-3</i>
<i>TI-RTOS Configuration Tools</i>	<i>4-5</i>
<i>Why use TI-RTOS Scheduler?</i>	<i>4-7</i>
<i>TI-RTOS Debugging Tools</i>	<i>4-11</i>
<i>Getting Started with a TI-RTOS Application</i>	<i>4-13</i>

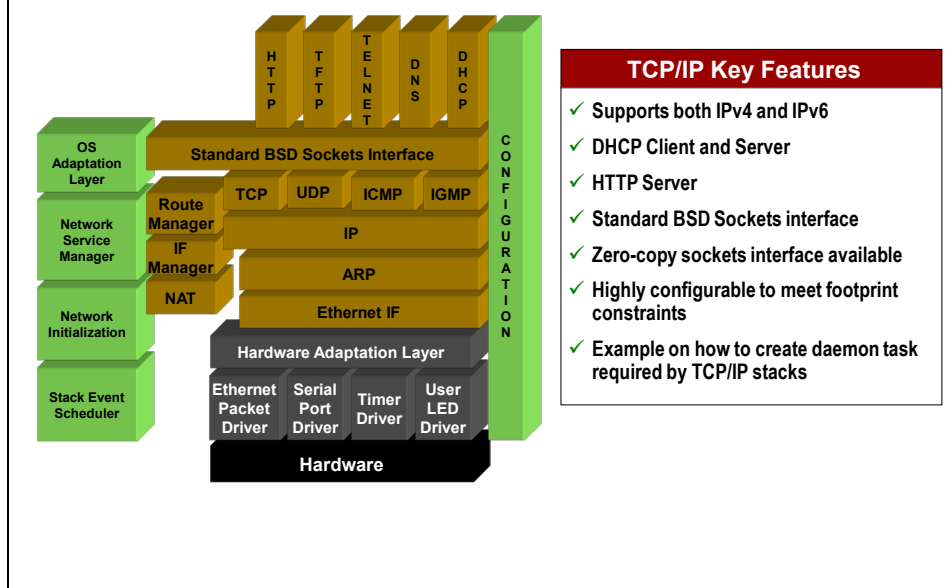
Introduction to TI-RTOS



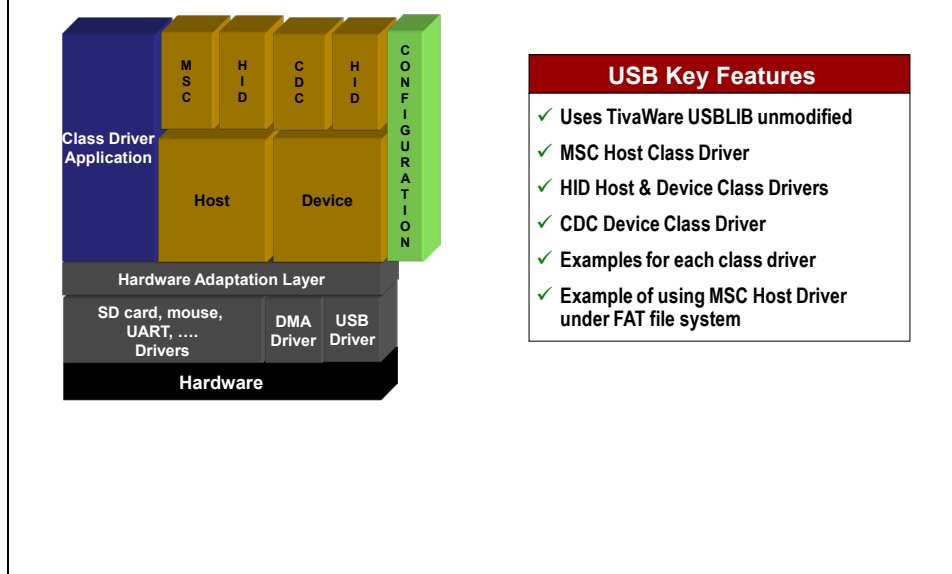
TI-RTOS Device Drivers

Driver	Description
Ethernet	Used by TCP/IP stack
WiFi	Used by socket interface
SD Card	Used by FAT File System (uses SPI interface)
USB Host MSC to USB Flash Card	Used by combination of FAT file system, USB Host stack, and MSC class driver
USB HID	Working example code that will likely be modified by user
USB CDC	Working example code that will likely be modified by user
UART	Intended for direct use by application
SPI	Intended for direct use by application
Watchdog	Intended for direct use by application
I ² C	Intended for direct use by application
GPIO	Intended for direct use by application

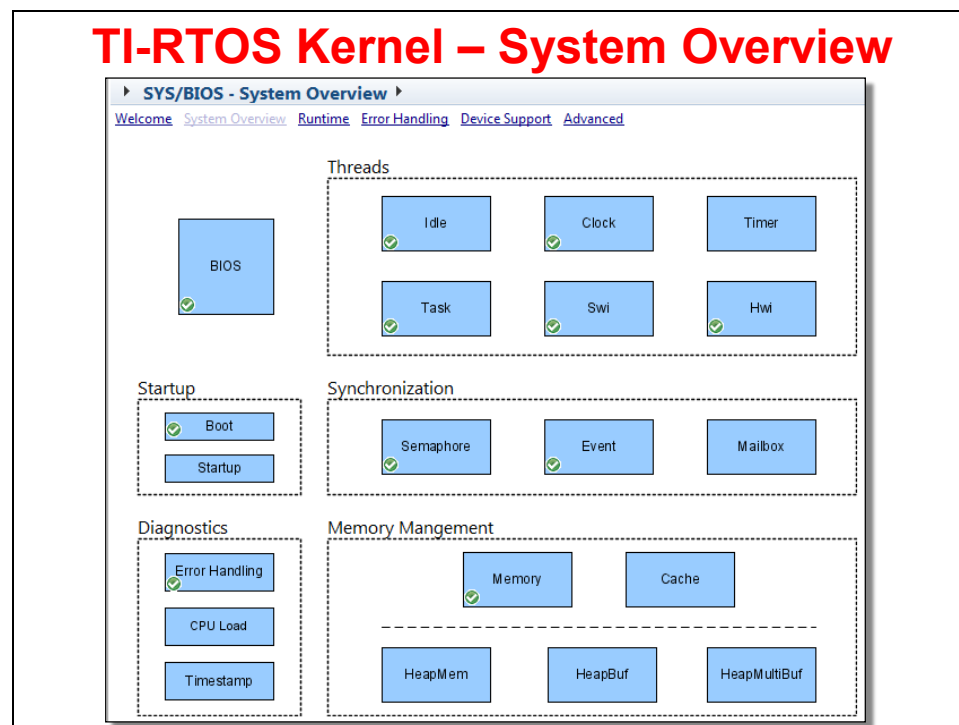
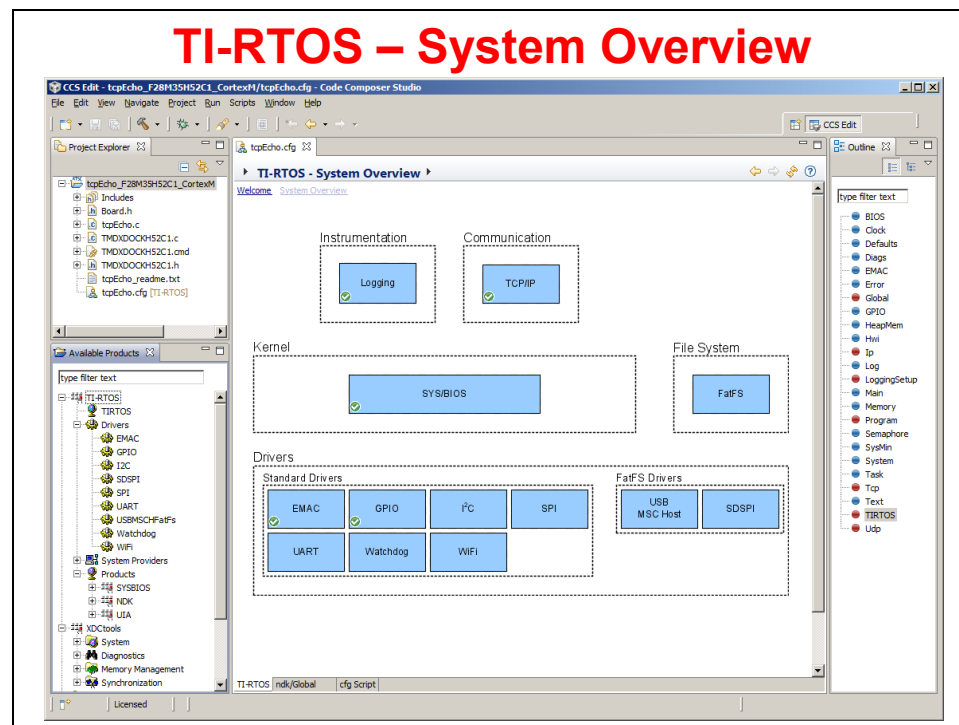
TCP/IP Stack



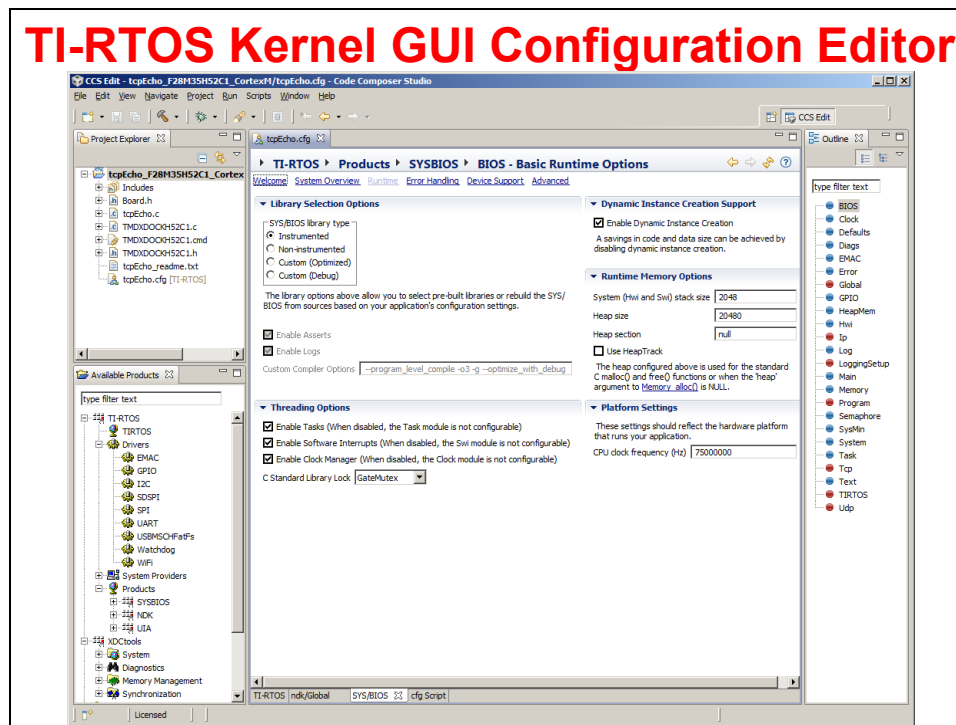
USB Stack



TI-RTOS Configuration Tools

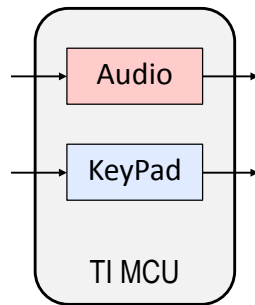


TI-RTOS Kernel GUI Configuration Editor



Why use TI-RTOS Scheduler?

Scheduling Problem – Two Threads



Problem Definition: you have two different threads that need to be serviced independently

- ◆ Will one routine conflict with the other?
- ◆ How do you SCHEDULE each thread?
- ◆ Is one “thread” higher PRIORITY than the other?

Let's explore a few options we can use to SCHEDULE these two threads...

Solution #1 – Super Loop

```
main()
{
    while(1)
    {
        Audio
        KeyPad
    }
}
```

Solution #1 – put each algo into an endless loop in main()

- ◆ What if algos run at different rates?
 - Audio – 100KHz (10µs)
 - KeyPad – 10 Hz (100ms)
- ◆ What if one starves the other or delays the response causing jitter/noise?

Could you use a TIMER to trigger each “thread” (or process) via an interrupt?

Solution #2 – Timer-based INTs

```
TimerA_ISR()
{
    read sample;
    Audio
}
```

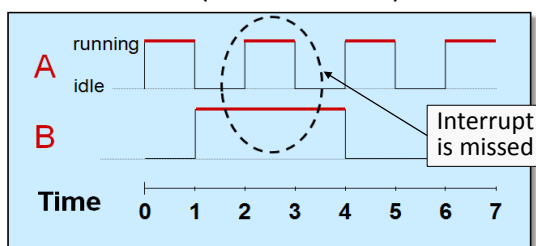
```
TimerB_ISR()
{
    read keypad;
    KeyPad
}
```

```
main()
{
    while(1);
}
```

Solution #2 – an *interrupt driven system* places each function in its own ISR

	Period	Compute	Usage
Audio	10 μ S	5 μ S	50%
Keypad	100ms	1ms	1%
			51%

- While CPU usage is fine, one interrupt may block the other (instantaneous):



How could we prevent this?

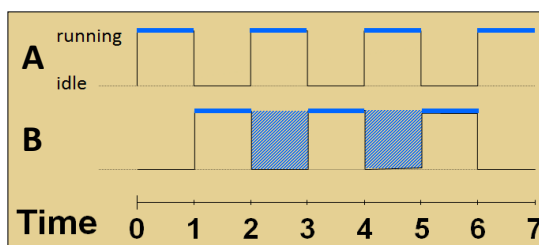
Solution #3 – Nested INTs (1)

```
TimerA_ISR()
{
    read sample;
    Audio
}
```

```
TimerB_ISR()
{
    read keypad;
    KeyPad
}
```

```
main()
{
    while(1);
}
```

Solution #3 – *nested interrupts* allow hardware interrupts to preempt each other



- Number of priorities are tied to the number of interrupts (one fxn/ISR), h/w priorities inflexible
- Lower priority ISRs must enable higher priorities via manual code (touch one, touch all) – very messy and hard to validate

Why is nesting required in this system?

Solution #4 ? – Separate Process & ISR

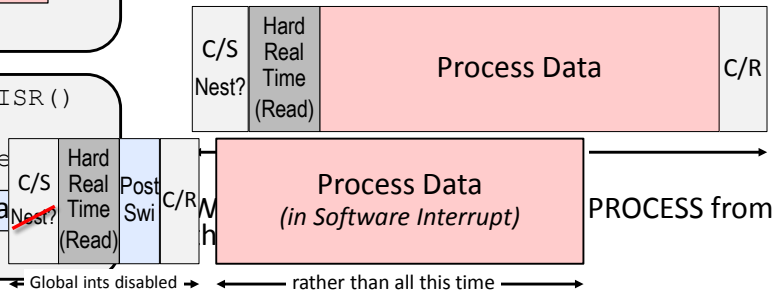
```
TimerA_ISR()
{
  read sample;
  Audio
}
```

```
TimerB_ISR()
{
  read keypad;
  KeyPa
}
```

```
main()
{
  while(1);
}
```

Problem – *nested interrupts* are used because “Process” (algo) is done IN the ISR !

- ◆ When HI PRI is running, you could STILL miss interrupts:



This is what the TI-RTOS Scheduler is all about...

Solution #4 – TI-RTOS Scheduler

```
Audio_ISR()
{
  read sample;
  post_Audio;
}
```

```
Keypad_ISR()
{
  read keypad;
  post_KeyPad;
}
```

```
main()
{
  init_stuff;
  while(1);
  BIOS_start();
}
```

Hwi – ISR

- Context save/restore (done by the scheduler)
- Hard real-time “read”
- Post Swi for follow-up

Swi – Software INT

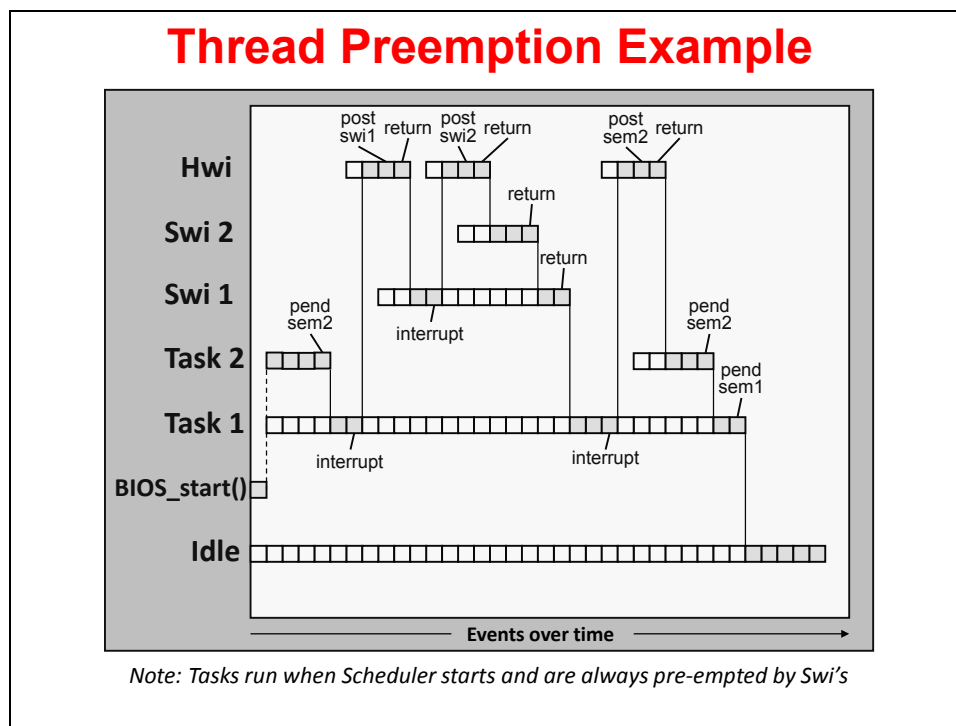
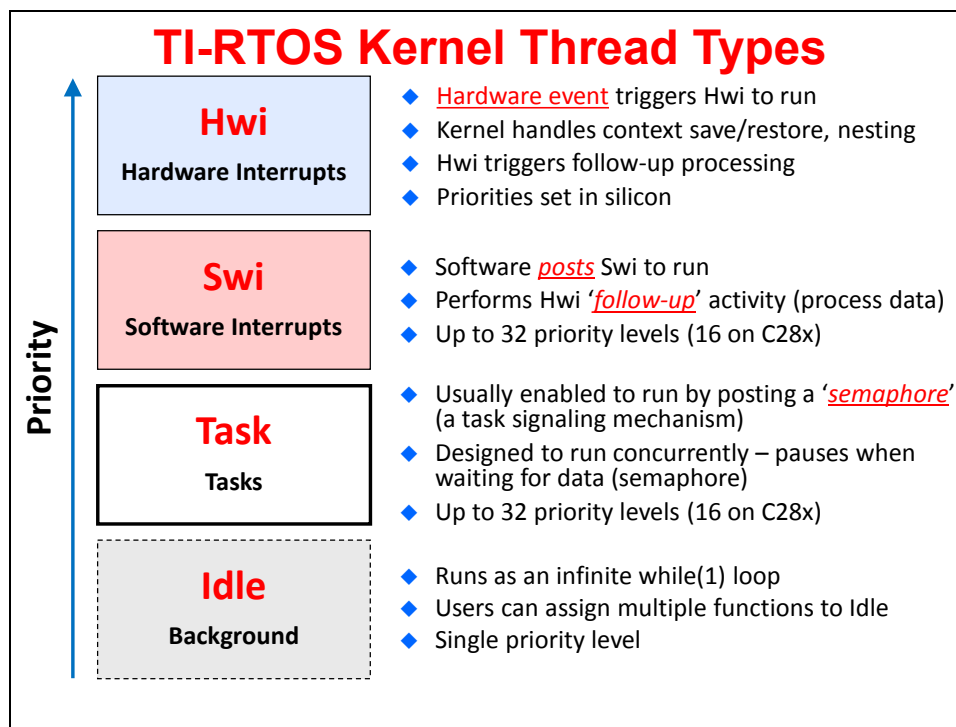
- Posted by software
- PROCESS data
- User can select priority

Idle – Background

- Runs multiple fxns inside of a while(1) loop

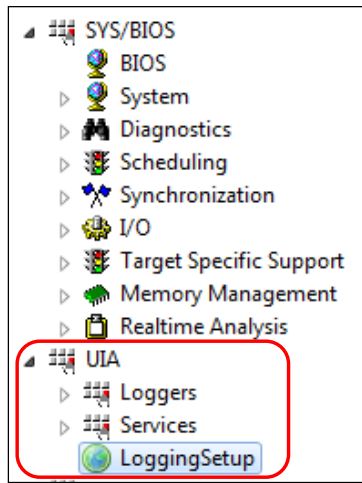
If we can DECOUPLE the processing from the TRIGGER (Hwi):

- ◆ ISRs become VERY short (no need for nesting)
- ◆ Configure PRIs of threads via software
- ◆ Add as many threads as we need (no limit)
- ◆ Touch ONE, no changes to the others !!



TI-RTOS Debugging Tools

Built-in Debug Tools (UIA & ROV) – Intro



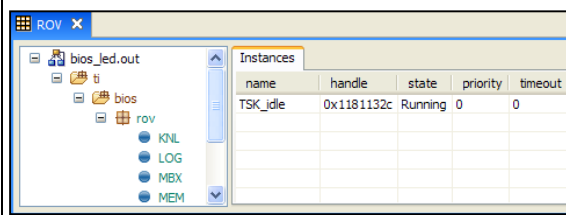
- ◆ Universal Instrumentation Architecture (**UIA**) tools provide *visibility* into what is going on in your system:
 - Logging – “printf()-lite”
 - Execution Graph – software “logic analyzer”
 - Load – CPU/Thread loading
 - UIA replaces the older RTA tools – requires “LoggingSetup”
- ◆ **ROV** – RTOS Object Viewer – see status of TI-RTOS objects in your system (when halted)

Let's look at what these tools look like in CCS...

ROV and UIA – Visibility/Debug Tools

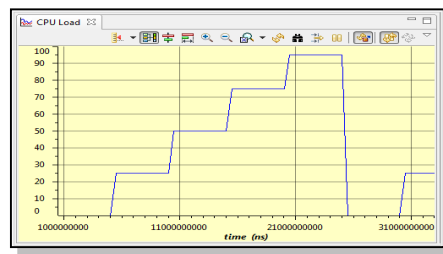
Real-time is...

- ◆ Gather data on target (30-40 CPU cycles)
- ◆ Format data on host (1000s of host PC cycles)
- ◆ Data gathering does NOT stop target CPU
- ◆ Halt CPU to see results (stop-time debug)



RTOS Obj Viewer (ROV)

- ◆ Halt to see results
- ◆ Displays stats about all threads in system



CPU/Thread Load Graph

- ◆ Analyze time NOT spent in Idle

ROV and UIA – Visibility/Debug Tools

Logs

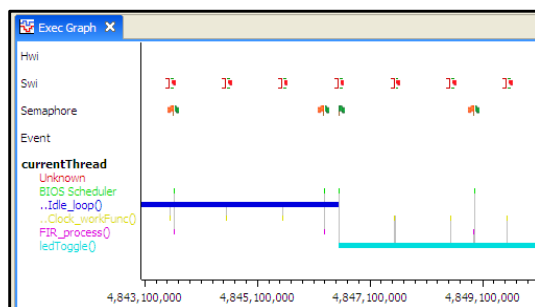
- ◆ Send DBG Msgs to PC
- ◆ Data displayed during stop-time
- ◆ Deterministic, low CPU cycle count
- ◆ WAY more efficient than traditional `printf()`

time	seqID	module	formattedMsg
4,257,279,253	125	Main	"../led.c", line 47: CPU LOAD = [38]
4,257,280,226	126	Main	"../led.c", line 49: TOGGLED LED [42] times
4,357,270,273	127	Main	"../led.c", line 43: BENCHMARK = [3221757] cycles
4,357,271,406	128	Main	"../led.c", line 47: CPU LOAD = [38]
4,357,275,486	129	Main	"../led.c", line 49: TOGGLED LED [43] times
4,457,286,080	130	Main	"../led.c", line 43: BENCHMARK = [3224677] cycles

```
Log_info1("TOGGLED LED [%u] times", count);
```

Execution Graph

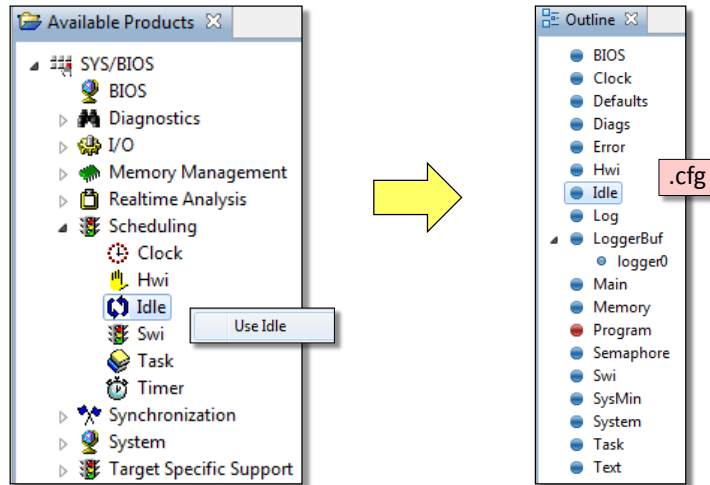
- ◆ View system events down to the CPU cycle...
- ◆ Calculate benchmarks



Getting Started with a TI-RTOS Application

Static Kernel Configuration (.cfg)

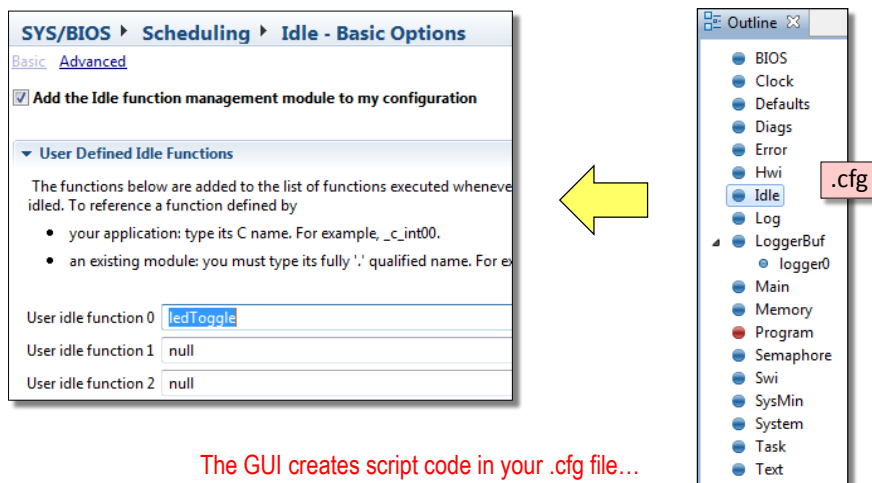
- ◆ The .cfg file contains all STATIC services used in the project
- ◆ To ADD a service, right-click on service in *Available Products* and select "Use xyz". It will then appear in your .cfg file (see Outline view)



When you click on Idle...

Static Kernel Configuration – Adding Idle Fxn

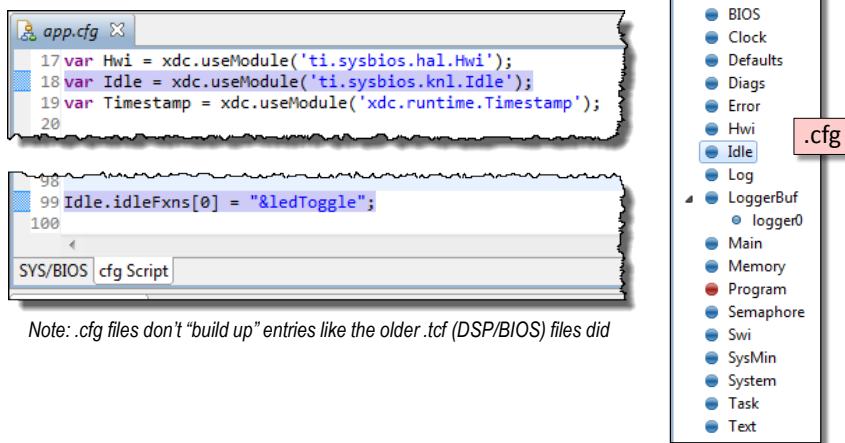
- ◆ When you click on a service (or kernel module) in the Outline view, you can then configure its settings (e.g. adding an Idle function):



The GUI creates script code in your .cfg file...

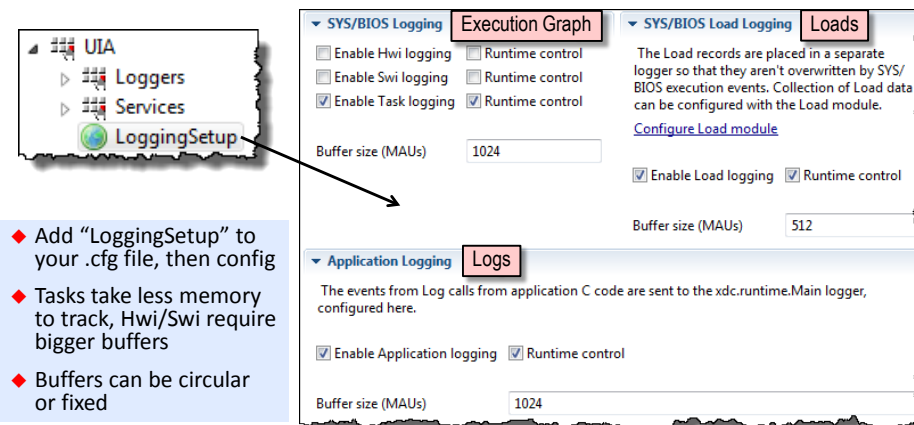
Static Kernel Configuration – Adding Idle Fxn

- ◆ All changes made to the GUI are reflected in the script (.cfg file) and vice versa
- ◆ Click on a module on the right and you can see the corresponding script in *app.cfg*:



Configuring UIA & RTOS Analyzer

- ◆ UIA (Unified Instrumentation Architecture) provides instrumentation APIs that run on the **TARGET** – Why? Visibility into what your program is doing (or not doing)
- ◆ The RTOS Analyzer displays the results of these commands in **CCS**
- ◆ Multiple transport protocols are supported – we will use **STOP-MODE JTAG**



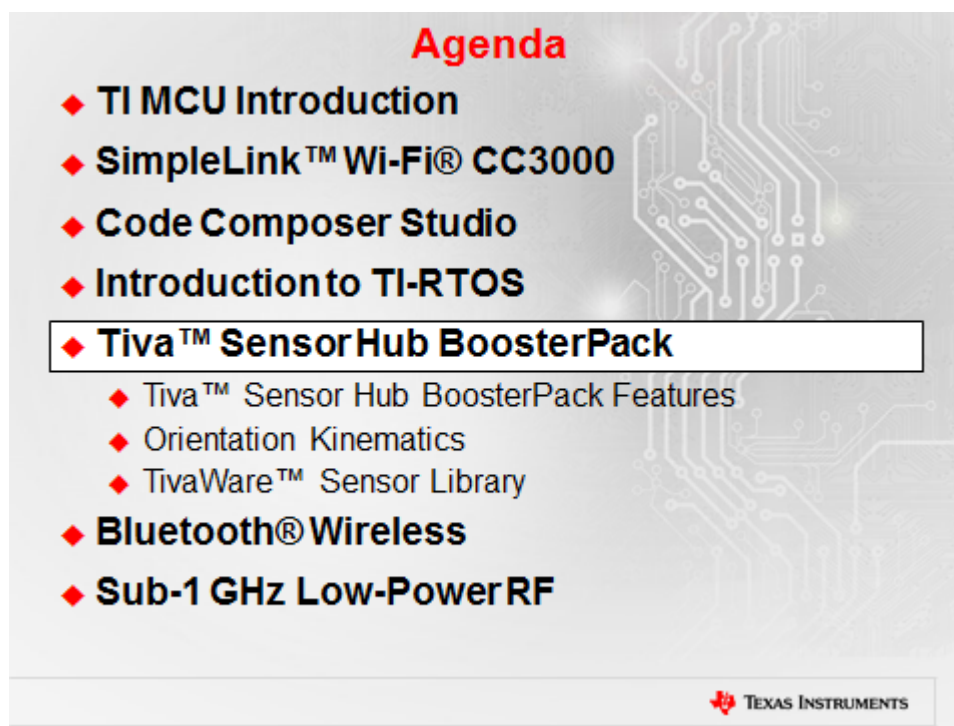
Tiva™ Sensor Hub BoosterPack

Introduction

The Tiva™ Sensor Hub BoosterPack is an exciting new addition to TI's MCU LaunchPad ecosystem. It is a plug-in daughter board that allows developers to create products with up to nine axes of motion tracking and multiple environmental sensing capabilities.

This BoosterPack is designed for TI's new Tiva C Series TM4C123G LaunchPad, but it will also work equally well with its predecessor, the Stellaris LM4F120XL LaunchPad. The BoosterPack is also hardware compatible with the existing MSP430, C2000, and Hercules LaunchPads.

Learning Objectives



Agenda

- ◆ TI MCU Introduction
- ◆ SimpleLink™ Wi-Fi® CC3000
- ◆ Code Composer Studio
- ◆ Introduction to TI-RTOS
- ◆ **Tiva™ SensorHub BoosterPack**
 - ◆ Tiva™ Sensor Hub BoosterPack Features
 - ◆ Orientation Kinematics
 - ◆ TivaWare™ Sensor Library
- ◆ Bluetooth® Wireless
- ◆ Sub-1 GHz Low-Power RF

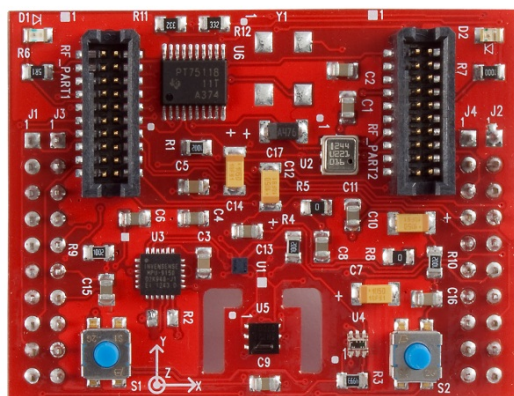
TEXAS INSTRUMENTS

Chapter Topics

Tiva™ Sensor Hub BoosterPack	5-1
<i>Tiva™ Sensor Hub BoosterPack Features</i>	<i>5-3</i>
<i>Orientation Kinematics</i>	<i>5-6</i>
<i>TivaWare™ Sensor Library.....</i>	<i>5-8</i>
<i>Lab 5(a) – Air Mouse Example</i>	<i>5-10</i>
Objective	5-10
<i>Lab 5(b) – Sensor Library Usage.....</i>	<i>5-15</i>
Procedure.....	5-15

Tiva™ Sensor Hub BoosterPack Features

Tiva™ Sensor Hub BoosterPack Features

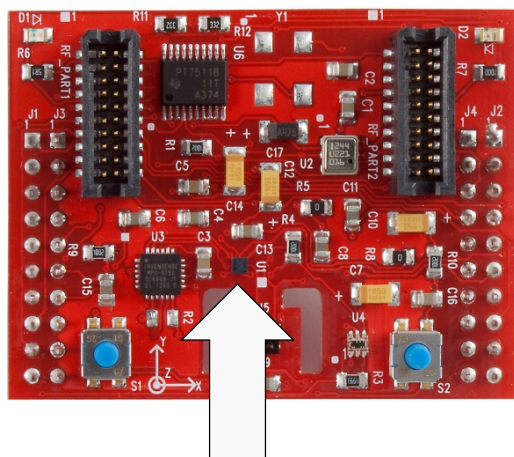


BOOSTXL-SENSHUB

- ◆ Motion and environmental sensing
- ◆ BoosterPack XL connectors (compatible with earlier BoosterPack connectors)
- ◆ EM board connectors (for TI's wireless RF evaluation kits)
- ◆ 2 buttons & 2 LEDs
- ◆ Example applications for each unique sensor
- ◆ "Air" Mouse (PC HID) example demonstrates sensor fusion
- ◆ CCS, Keil, IAR, & Sourcery CodeBench IDEs supported
- ◆ TivaWare DriverLib under TI BSD-style license
- ◆ Runs on Tiva TM4C123G and LM4F120 LaunchPads. HW compatible with MSP430 & C2000 LaunchPads
- ◆ MSRP \$49.99 USD

TMP006...

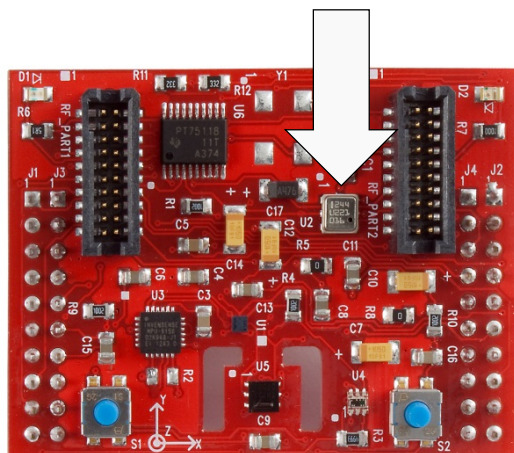
TI TMP006 Infrared Temperature Sensor



- ◆ No contact temperature measurement
- ◆ -40C to 125C measurement range
- ◆ 240uA supply current
- ◆ 2.2 to 7V supply
- ◆ I2C interface (address 0x41)
- ◆ Host calculates observed temperature

BMP180...

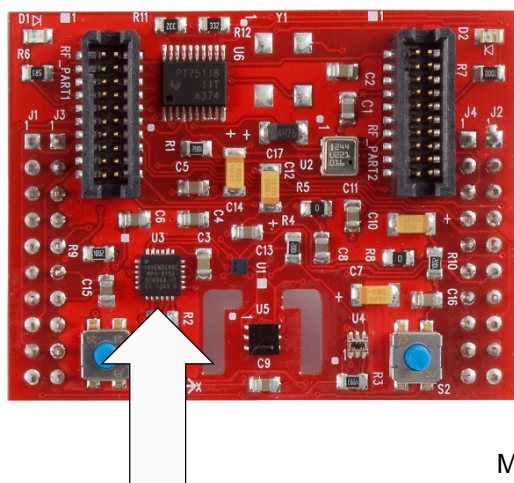
Bosch BMP180 Digital Pressure Sensor



- ◆ -500 to 9000m Mean Sea Level (1100 to 300hPa)
- ◆ Temperature sensing for altitude compensation
- ◆ 1.8 – 3.6V supply
- ◆ 5uA supply current at 1 sample/sec
- ◆ Very low noise
- ◆ Multiple modes for power/accuracy tradeoff
- ◆ I2C interface (address 0x77)
- ◆ Host calculates altitude

MPU9150...

Invensense MPU-9150 9-axis Motion Sensor

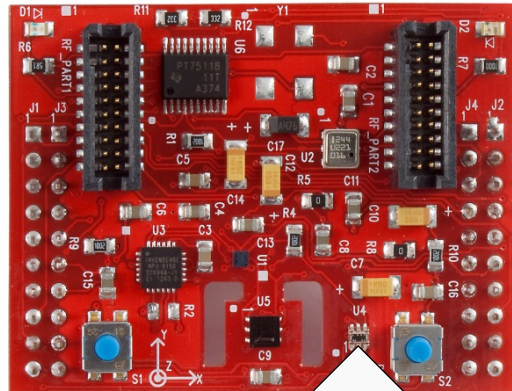


- ◆ 3-axis MEMS accelerometer
- ◆ 3-axis MEMS gyroscope
- ◆ 3-axis MEMS magnetometer
- ◆ 16-bit gyroscope and accelerometer resolution
- ◆ 13-bit magnetometer resolution
- ◆ I2C interface (address 0x68)
- ◆ 2.375 to 3.465V supply

MEMS = Micromechanical system

ISL29023...

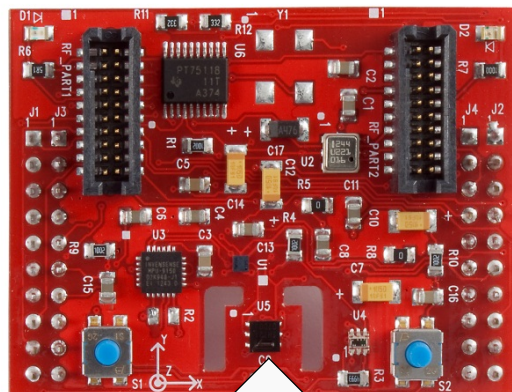
Intersil ISL29023 Ambient & Infrared Light Sensor



- ◆ 16-bit resolution
- ◆ 50 & 60Hz flicker rejection
- ◆ 1.7 to 3.63V supply
- ◆ I2C interface (address 0x44)
- ◆ HW (BoosterPack XL pin) and SW Interrupts on light levels

SHT21...

Sensirion SHT21 Humidity & Ambient Temperature Sensor



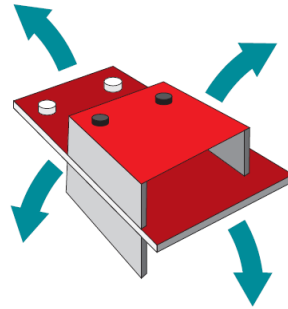
- ◆ 8/12-bit humidity resolution
- ◆ 12/14-bit temperature resolution
- ◆ 2.1 to 3.6V supply
- ◆ I2C interface (address 0x40)
- ◆ Slots in board for air circulation

Orientation Kinematics

Orientation Kinematics

Orientation Kinematics

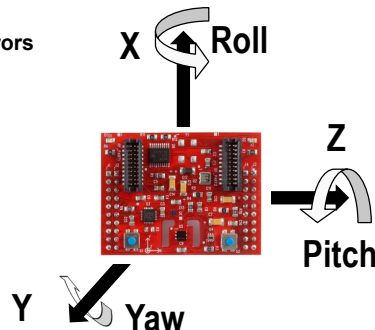
- ◆ The Direct Cosine Matrix (DCM) algorithm combines multiple axes of motion data into a single set of Euler angles for roll, pitch and yaw. The final calculated position is :
 - ◆ Less prone to drop-out
 - ◆ Of higher accuracy than the best individual sensor
- ◆ The DCM algorithm calculates the orientation of a rigid body, in respect to the rotation of the earth by using rotation matrices. The rotation matrices are related to the Euler angles, which describe the three consecutive rotations needed to describe the orientation
- ◆ The three sensors used in the algorithm are:
 - ◆ 3 axis accelerometer (measures earth's gravity field minus acceleration)
 - ◆ 3 axis magnetometer (measures earth's magnetic field)
 - ◆ 3 axis gyroscope (measures angular velocity)



DCM Algorithm...

DCM Algorithm

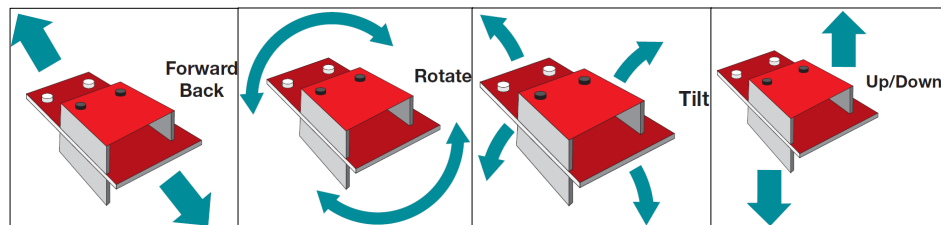
- ◆ The gyroscope is the primary sensor
 - ◆ Unaffected by the gravitational or magnetic field
 - ◆ Prone to drift
- ◆ The accelerometer is used as an orientation reference in the X and Z axes
 - ◆ Compensates for roll and pitch errors
- ◆ The magnetometer is used to calculate reference vector in the Y axis
 - ◆ Compensates for yaw errors
- ◆ Proportional feedback removes the gyro's drift



Air Mouse Example...

Air Mouse Example

- ◆ The Invensense MPU-9150 provides raw acceleration, angular velocity and magnetic field measurements.
- ◆ All 9 axes are fused and filtered using a complimentary direct cosine matrix or DCM algorithm into Euler angles for roll, pitch and yaw.
- ◆ Roll and pitch are used to perform the mouse movements.
- ◆ Raw angular velocities and accelerations are used to interpret gestures
- ◆ Angles are calculated 100 times per second

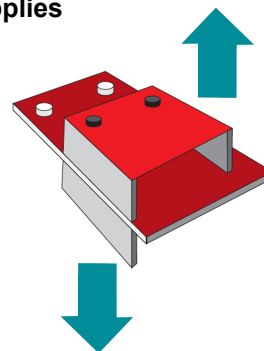


Lab ...

TivaWare™ Sensor Library

TivaWare™ Sensor Library Contents

- ◆ Drivers for the I²C port and each sensor
- ◆ Functions for manipulating the magnetometer readings
- ◆ DCM Algorithm
 - ◆ comp_dcm.c/h reads the sensors and applies the DCM algorithm to the data
- ◆ Vector operations
 - ◆ VectorAdd()
 - ◆ VectorCrossProduct()
 - ◆ VectorDotProduct()
 - ◆ VectorScale()



Sensor Library Usage...

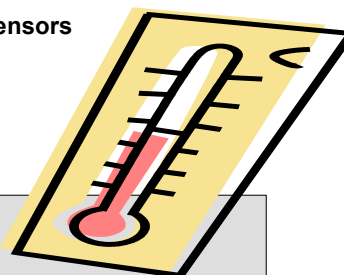
TivaWare™ Sensor Library Usage

The Sensor library is a consistent API with this general flow for all sensors

It's easy to leverage the library for custom I²C sensors

For instance, to interface with the TMP006:

- ◆ Initialize I²C pins and I²C peripheral normally
- ◆ Initialize the I²C driver `I2CMiniInit()`
- ◆ Initialize the TMP006 `TMP006Init()`
- ◆ Configure the TMP006 `TMP006ReadModifyWrite()`
- ◆ Read data from the TMP006 `TMP006DataRead()`
- ◆ Convert data into temperature `TMP006DataTemperatureGetFloat()`



Examples...

TivaWare™ Sensor Hub Examples

airmouse

- ◆ fuses motion data into mouse and keyboard events

compcdm_mpu9150

- ◆ basic data gathering from the MPU-9150

drivers

- ◆ for buttons and LEDs

humidity_sht21

- ◆ periodic measurements of humidity

light_isl29023

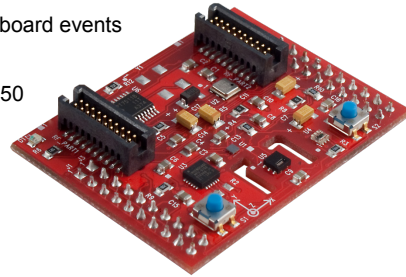
- ◆ uses measurements of ambient visible and IR light to control the “white” LED

pressure_bmp180

- ◆ periodic measurements of air pressure and temperature

temperature_tmp006

- ◆ periodic measurements of ambient and IR temperatures to calculate actual object temperature



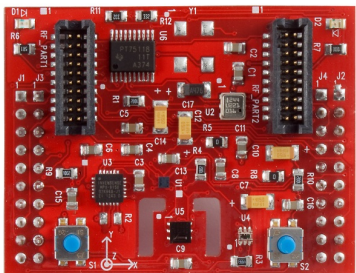
Lab...

Lab 5(a) – Air Mouse Example

Objective

In this lab you will experiment with the Air Mouse example, programming the code into the TM4C123G's flash memory using the LM Flash Programmer. To complete labs 5(a) and 5(b) you will need a BOOSTXL-SENSHUB Sensor Hub Boosterpack.

Lab 5(a) – Air Mouse

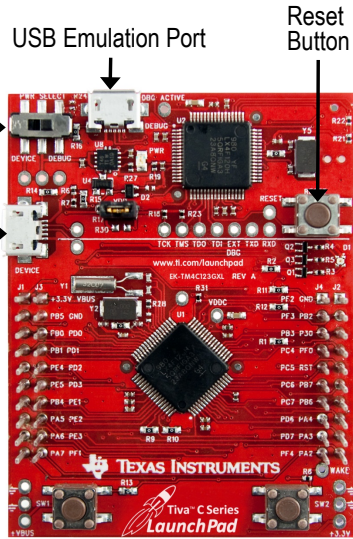


Reset Button

USB Emulation Port

Power Switch

USB H/D/OTG Port



- ◆ Program the airmouse.bin binary into the MCU's flash memory using LM Flash Programmer
- ◆ Install the Sensor Hub BoosterPack
- ◆ Experiment with the Air Mouse example

Sensor Library ...

LM Flash Programmer

1. Install the Sensor Hub BoosterPack

- ▶ Remove the USB cable from the LaunchPad's emulator port.
- ▶ Connect the Sensor Hub BoosterPack onto the XL connectors of the LaunchPad board.

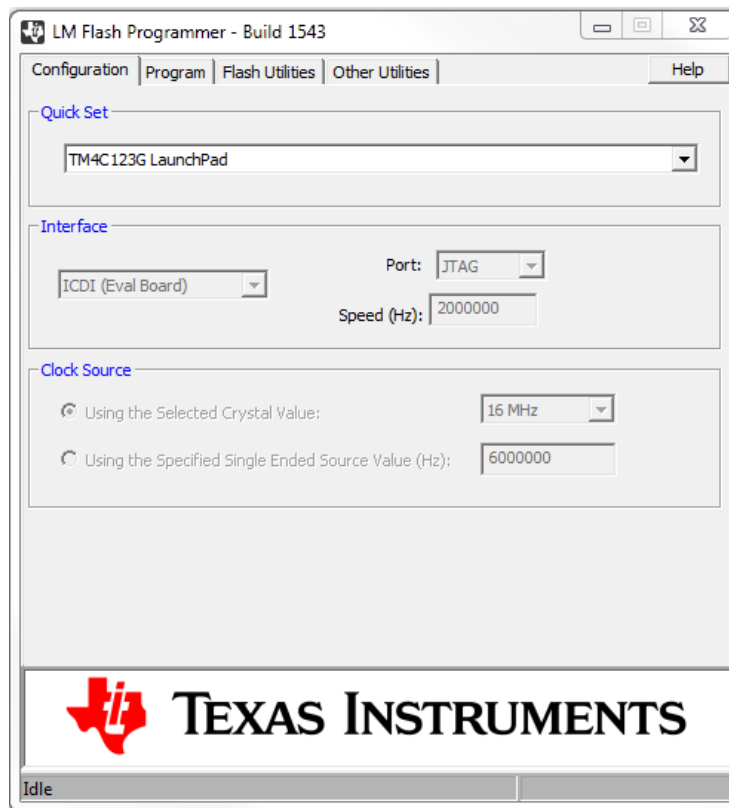
The buttons on the BoosterPack should be at the same end as the ones on the LaunchPad. You may need to carefully bend the power measurement jumper out of the way slightly.

- ▶ Connect the USB cable from the emulation port to an open USB port on your computer.

2. Program the Air Mouse application into the flash using LM Flash Programmer

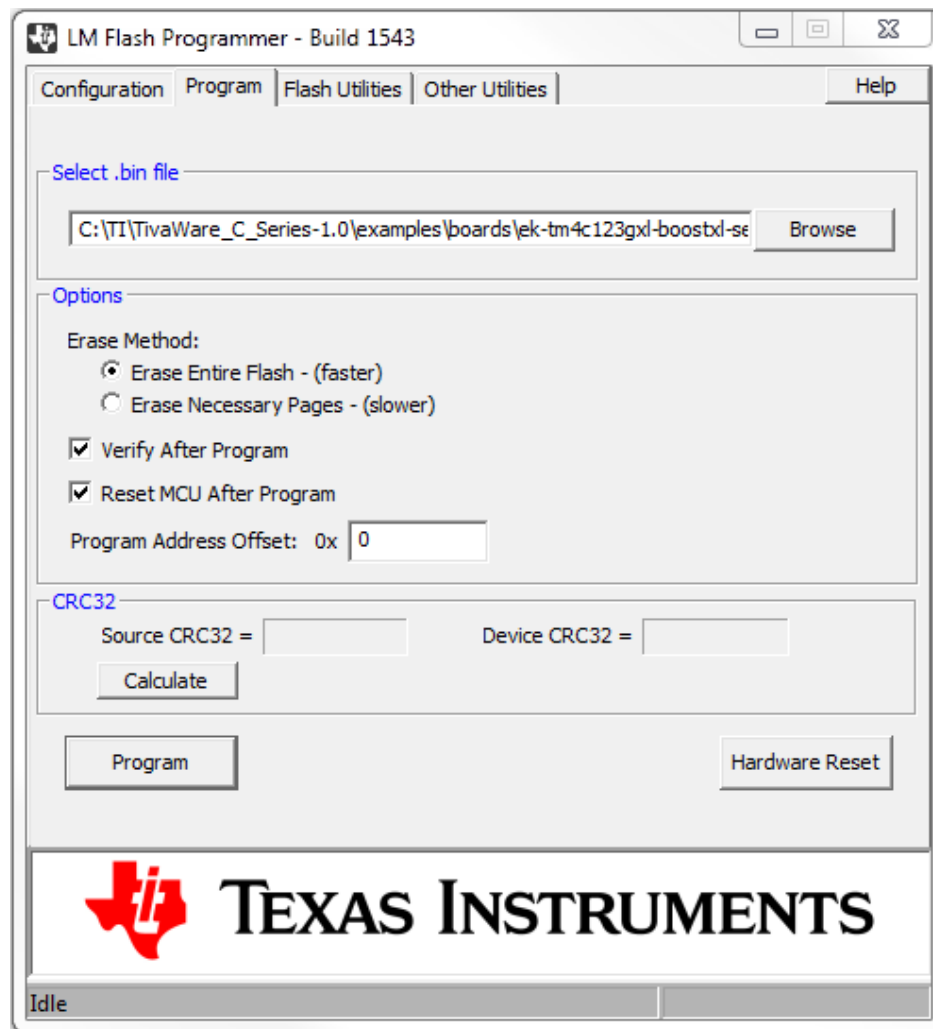
- ▶ Run the LM Flash Programmer, make the selection in the Quick Set window shown below.

If you have an older version of this tool, use the LM4F120XL selection instead.



- ▶ Click on the *Program* tab at the top, then click the *Browse* button and browse to:
C:\TI\TivaWare_C_Series-1.0\examples\boards\ek-tm4c123gx1-boostx1-senshub\airmouse\ccs\Debug and select **airmouse.bin**.
- ▶ Check the “Verify After Program” and “Reset MCU After Program” checkboxes.
- ▶ Click the Program button.

When the process completes, close the LM Flash Programmer.



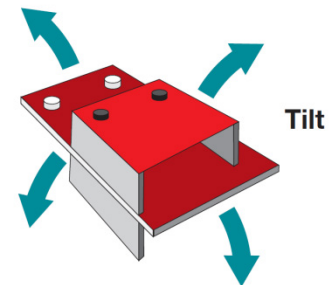
- ▶ Open a browser window or a longer pdf or Word document on your desktop.
- ▶ Unplug your USB cable from the LaunchPad's emulation port.
- ▶ Switch the power switch to the DEVICE (left-most) position and connect the USB cable to the H/D/OTG port on the side of the LaunchPad (see the earlier diagram).
- ▶ Press the LaunchPad's reset button to make sure that the code starts up properly.

Air Mouse Example

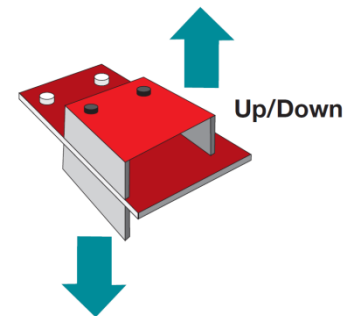
Your computer will detect the new USB device and install the standard mouse drivers. If everything is working properly, the LEDs on the LaunchPad will be blinking quickly.

The proper way to hold the air mouse is with the USB cable to the right and the buttons under your fingers. Although both sets of buttons will work, it's easier to use the LaunchPad buttons.

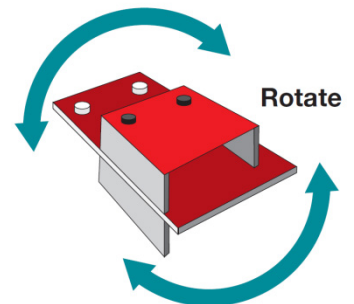
3. ► Gently pitch the LaunchPad forward and back to mouse down and up. Roll left and right to mouse left and right. The left and right buttons should work normally.



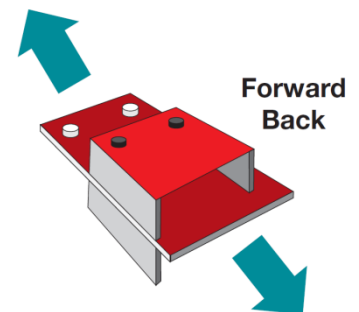
4. ► From a resting position flat and level, a quick jerk up will simulate ALT+TAB on your keyboard to show your open programs. Once "lifted", a quick twist left or right will select between the available windows. A quick jerk down will make the selection stick and release the ALT key. If you find yourself "stuck", press the ALT key on your keyboard to exit the mode.



5. ► From flat and level, a spin about the Z (vertical) axis will PAGE UP or PAGE DOWN.



6. ► From flat and level, a quick forward or back motion while keeping the air mouse flat will zoom in and out.



Explanation: For the air mouse example, the Invensense MPU-9150 provides raw acceleration, angular velocity and magnetic field measurements. All 9 axes are fused and filtered using a complimentary direct cosine matrix, or DCM, algorithm into Euler angles for roll, pitch and yaw. Roll and pitch are used to perform the mouse movements. Raw angular velocities and accelerations are used to interpret gestures.

7. When you're done experimenting,

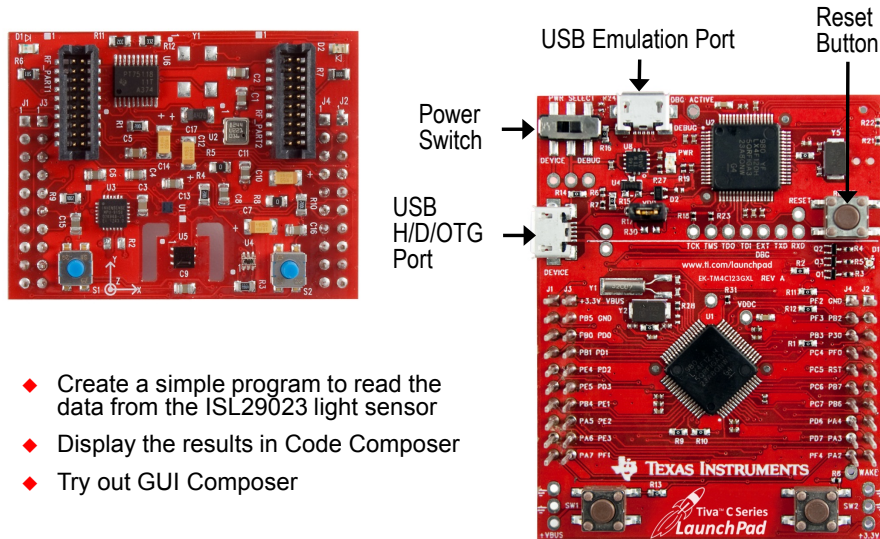
- ▶ remove the USB cable from the LaunchPad's device port,
- ▶ move the power switch back to the DEBUG (right-most) position and
- ▶ connect the USB cable to the LaunchPad's emulator port.



You're done with Lab 5(a)

Lab 5(b) – Sensor Library Usage

Lab 5(b) – Sensor Library Usage



- ◆ Create a simple program to read the data from the ISL29023 light sensor
- ◆ Display the results in Code Composer
- ◆ Try out GUI Composer

Procedure

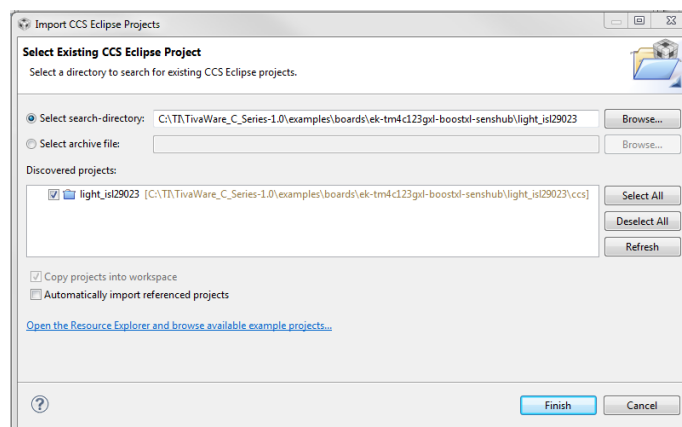
1. Import the Project

- Open CCS.

On the CCS Menu bar, ► click Project → Import Existing CCS Eclipse Project.

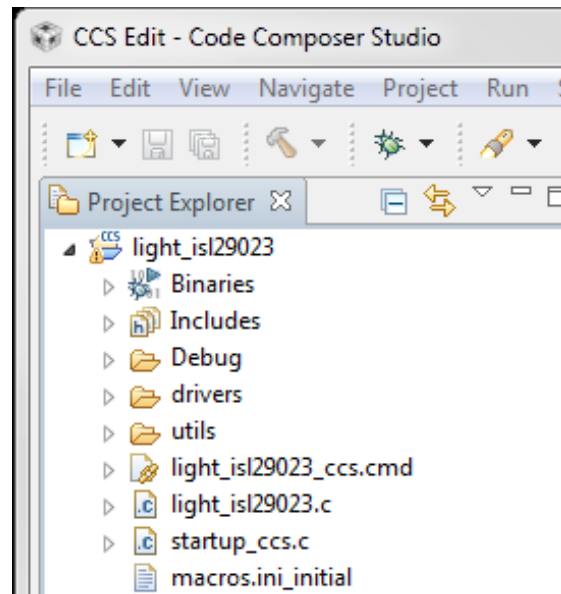
► Click Browse, then navigate to C:\TI\TivaWare_C_Series-1.0\examples\boards\ek-tm4c123gx1-boostx1-senshub\light_isl29023, ► click OK.

► Click the Finish button. The project files will be copied into your workspace folder.



In the Project Explorer pane, ► click the ► beside the light_isl29023 project name to expand the project. The main components of the project are light_isl29023.c, which

contains all of the c code needed to run the program and `startup_ccs.c`, which contains the reset vector, ISR vectors and system fault handlers.

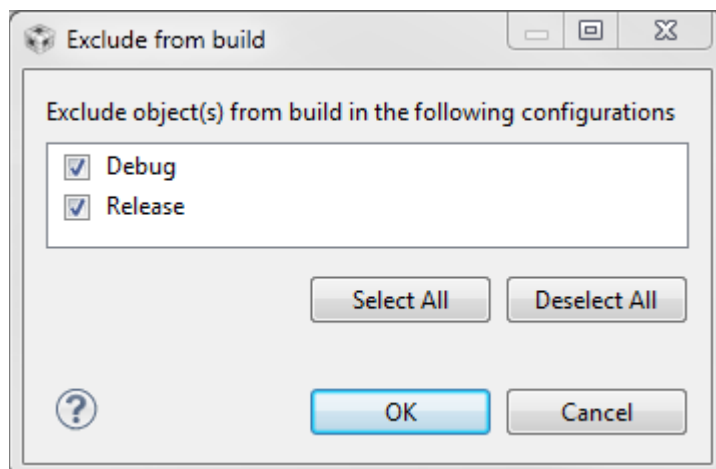


2. Make a copy of `light_isl29023.c` to save the original file.

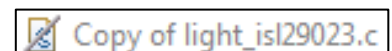
- ▶ Right-click on `light_isl29023.c` and select **Copy**.
- ▶ Right-click again in the open space of the Project Explorer pane and select **Paste**.

When the **Name Conflict** dialog appears, accept the “Copy of `light_isl29023.c`” name by clicking **OK**. This will preserve the existing file for later use, but you’ve created a problem ... both files contain a `main()` and both are part of the project, so this needs to be fixed.

- ▶ Right-click on the “Copy of `light_isl29023.c`” file, select **Resource Configurations** and then **Exclude from Build ...**. When the Exclude from build dialog appears, click the **Select All** button and then click **OK**.



Note that the symbol for the file will have a “strike” through it.



3. Make a copy of `startup_ccs.c` to save the original file.

- ▶ Right-click on `startup_ccs.c` and select **Copy**.

- ▶ Right-click again in the open space of the Project Explorer pane and select **Paste**.

When the **Name Conflict** dialog appears, accept the “Copy of startup_ccs.c.c” name by clicking **OK**.

- ▶ Right-click on the “Copy of startup_ccs.c” file, select **Resource Configurations** and then **Exclude from Build ...**. When the Exclude from build dialog appears, click the **Select All** button and then click **OK**.

Write the Code

4. If we're going to write a sensor application from a blank sheet of paper, we first need that blank sheet.

- Double-click on `light_isl29023.c` in the Project Explorer pane to open the file for editing in the Editor pane.
- Click anywhere in the code, press Ctrl-A on your keyboard to select all the code and then press your delete button. Viola, a blank sheet to start from.
- **Copy** the following lines from this pdf file and **insert** them into the blank sheet you just created. In most pdf readers you can select either a screen capture (arrow pointer) or text (cursor or I pointer). Use the text selector for the best results.

```
#include "stdint.h"
#include "stdbool.h"
#include "inc/hw_memmap.h"
#include "inc/hw_ints.h"
#include "driverlib/debug.h"
#include "driverlib/gpio.h"
#include "driverlib/interrupt.h"
#include "driverlib/pin_map.h"
#include "driverlib/rom.h"
#include "driverlib/sysctl.h"
#include "sensorlib/hw_isl29023.h"
#include "sensorlib/i2cm_drv.h"
#include "sensorlib/isl29023.h"
#define DEBUG
```

In order, the purpose of each of these is:

`stdint.h`: assure that integer types are compatible with the 1999 C standard
`stdbool.h`: assure that Boolean types are compatible with the 1999 C standard
`hw_memmap.h`: define the memory map of the device
`hw_ints.h`: macros defining the interrupt assignments
`hw_ints.h`: macros for assisting debug of the driver library
`gpio.h`: definitions and macros for the general purpose I/O APIs
`interrupt.h`: prototypes for the interrupt controller driver
`pin_map.h`: the mapping of the peripherals to the pins
`rom.h`: macros to facilitate calling the functions in ROM
`sysctl.h`: prototypes for the system control driver
`hw_isl29023.h`: macros for accessing the Intersil light sensor
`i2cm_drv.h`: prototypes for the I²C master driver
`isl29023.h`: prototypes for the light sensor driver
`DEBUG`: See later lab step

- Leave a blank line for spacing and add the following five lines below the includes:

```
#define ISL29023_I2C_ADDRESS    0x44      // ISL29023 I2C address
tI2CInstance g_sI2CInst;              // I2C master driver structure
tISL29023 g_sISL29023Inst;            // ISL29023 sensor driver structure
volatile unsigned long g_vui8DataFlag; // Data ready flag
volatile unsigned long g_vui8ErrorFlag; // Error flag
```


5. Add Handlers and Functions

- Leave a blank line for spacing and add the following code below the last.

This error routine will be called if the driver library experiences an error. Run-time error checking is fairly simple so that the impact to speed during runtime will be minimal, but accounting for potential errors is good programming practice. This code will save the location of the error, but only when the project is built with the DEBUG definition.

```

//*****
#ifdef DEBUG
void
__error__(char *pcFilename, uint32_t ui32Line)
{
}
#endif
1. //*****
***

```

- Leave a blank line for spacing and add the following code below the last. This is the ISL29023 sensor callback function, which will be called at the end of the ISL29023 sensor driver transaction. It is called from the I²C interrupt context that we'll add shortly. It assures that the I²C communication was successfully completed and sets the appropriate flags.

```

//*****
void
ISL29023AppCallback(void *pvCallbackData, uint_fast8_t ui8Status)
{
    if(ui8Status == I2CM_STATUS_SUCCESS)
    {
        g_vui8DataFlag = 1;
    }
    g_vui8ErrorFlag = ui8Status;
}
2. //*****
***

```

- Leave a blank line for spacing and add the following code below the last. This handler code will be called by the device's interrupt controller when an I2C3 interrupt occurs. I²C port 3 on the TM4C123G is the connection to the ISL29023.

```

//*****
void
ISL29023I2CIntHandler(void)
{
    I2CMIntHandler(&g_sI2CInst);
}
3. //*****
***

```

- Leave a blank line for spacing and add the following code below the last. This is the ISL29023 application error handler. If an error occurs, execution will trap here. Maybe that isn't what you'd like to happen in a production system.

```
/*******  
void  
ISL29023AppErrorHandler(char *pcFilename, uint_fast32_t ui32Line)  
{  
    while(1)  
    {  
    }  
}  
4.  /*******  
***
```

- Leave a blank line for spacing and add the following code below the last. This function waits for the ISL29023 I²C transactions to complete. If an error occurs the error handler will be called immediately.

```
/*******  
void  
ISL29023AppI2CWait(char *pcFilename, uint_fast32_t ui32Line)  
{  
    while((g_vui8DataFlag == 0) && (g_vui8ErrorFlag == 0))  
    {  
    }  
    if(g_vui8ErrorFlag)  
    {  
        ISL29023AppErrorHandler(pcFilename, ui32Line);  
    }  
    g_vui8DataFlag = 0;  
}  
5.  /*******  
***
```

6. Now let's add main() to the file

In this code `fAmbient` is a variable that holds the light reading from the sensor. `ui8Mask` holds a series of parameters with which to program the ISL29023.

► Leave a blank line for spacing and add the following code below the last.

```

//*****
int
main(void)
{
    float fAmbient;
    uint8_t ui8Mask;
}

```

The first thing we want the processor to do after reset is to properly configure the clock. The following API will set up the system clock at 40MHz using the PLL with the 16MHz external crystal as a reference.

► After the line containing `uint8_t ui8Mask;` add a line for spacing and then add the API below.

```

ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);

```

Note the `ROM_` preceding the API. The ROM on all Tiva C Series devices contains the entire TivaWare™ peripheral driver library. Calling functions from ROM saves precious Flash memory for the users' functions.

Next we need to enable I²C port 3 on the device. The pins are multiplexed with 4 functions per pin, so this programming is critical. TI has created a pin mux GUI to ease this programming; you can find it at: http://www.ti.com/tool/Im4f_pinmux. The last line turns on the “master interrupt switch”, enabling interrupts on the processor.

- Add a line for spacing under the last and add these seven lines to **main()**.

```
ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_I2C3);
ROM_GPIOPinConfigure(GPIO_PD0_I2C3SCL);
ROM_GPIOPinConfigure(GPIO_PD1_I2C3SDA);
GPIOPinTypeI2CSCL(GPIO_PORTD_BASE, GPIO_PIN_0);
ROM_GPIOPinTypeI2C(GPIO_PORTD_BASE, GPIO_PIN_1);

ROM_IntMasterEnable();
```

Now we're ready to initialize I²C port 3.

- Add a line for spacing under the last and add the next two lines to **main()**.

```
I2CInit(&g_sI2CInst, I2C3_BASE, INT_I2C3, 0xFF, 0xFF, ROM_SysCtlClockGet());

SysCtlDelay(SysCtlClockGet() / 3);
```

The parameters in the first line specify the I²C instance, the base address of the I²C module, the μ DMA Tx and Rx channels used (none) and the clock frequency used as the I²C module input clock. The second line provides a 1000mS delay to allow for any possible conflicts on the I²C bus to resolve.

Now it's time to initialize the ISL29023.

- Add a line for spacing under the last and add these lines to **main()**.

```
ISL29023Init(&g_sISL29023Inst, &g_sI2CInst,
            ISL29023_I2C_ADDRESS, ISL29023AppCallback, &g_sISL29023Inst);

6.    ISL29023AppI2CWait(__FILE__, __LINE__);
```

The first API initializes the ISL29023 driver, preparing it for operation. It also asserts a reset signal to the ISL29023 itself, to clear any previous configuration data.

The first parameter is a pointer to the ISL29023 instance data. The second is a pointer to the I²C driver instance data. The third is the I²C address of the ISL29023 device. The fourth is the function to be called when the initialization has completed (can be NULL if a callback is not required). The last is a pointer that is passed to the callback function.

The second API simply waits for the I²C communication to complete. If an error occurs, its location will be preserved.

The next lines configure the ISL29023. The R-M-W API modifies the operation register on the ISL29023. These parameters are defined in **isl29023.h**.

- Skip a line for spacing and add this code under the rest inside **main()**.

```
ui8Mask = (ISL29023_CMD_I_OP_MODE_M );

ISL29023ReadModifyWrite(&g_sISL29023Inst, ISL29023_O_CMD_I, ~ui8Mask,
    (ISL29023_CMD_I_OP_MODE_ALS_CONT),
    ISL29023AppCallback, &g_sISL29023Inst);

ISL29023AppI2CWait(__FILE__, __LINE__);
```

- Double-click on any of the parameters and press F3 to quickly see its definition.

The parameters passed in this configuration assure that the ISL29023 is in operation mode and continuous sampling mode.

Again, the wait is needed to ensure the completion of the last communication before starting the next.

7. Add the code that continuously reads the light sensor – a while(1) loop

- Skip a line for spacing and add the following after the last code.

```
while(1)
{
}
```

- Insert the following lines into the **while(1)** loop.

```
ISL29023DataRead(&g_sISL29023Inst, ISL29023AppCallback, &g_sISL29023Inst);
ISL29023AppI2CWait(__FILE__, __LINE__);

7. ISL29023DataLightVisibleGetFloat(&g_sISL29023Inst, &fAmbient);
```

The first line reads the data from the sensor and the third converts it into a floating point number stored in **fAmbient**. This will occur as quickly as the I2C communication transactions will allow, based on the wait APIs.

Note: We will be using breakpoints in this lab to slow the interaction. Without those breakpoints we would likely be sampling the sensor far too quickly for it to perform a proper conversion.

- Correct the indentation of your code if necessary.
- Click the Save button on the CCS menu bar to save your work. Note that the asterisk on the tab will disappear when the saved version is current.
- Compare your code with the code on the next two pages. If you are having problems, you can copy/paste this into Code Composer Studio.

```

#include "stdint.h"
#include "stdbool.h"
#include "inc/hw_memmap.h"
#include "inc/hw_ints.h"
#include "driverlib/debug.h"
#include "driverlib/gpio.h"
#include "driverlib/interrupt.h"
#include "driverlib/pin_map.h"
#include "driverlib/rom.h"
#include "driverlib/sysctl.h"
#include "sensorlib/hw_isl29023.h"
#include "sensorlib/i2cm_drv.h"
#include "sensorlib/isl29023.h"
#define DEBUG

#define ISL29023_I2C_ADDRESS    0x44           // ISL29023 I2C address
tI2CInstance g_sI2CInst;                 // I2C master driver structure
tISL29023 g_sISL29023Inst;              // ISL29023 sensor driver structure
volatile unsigned long g_vui8DataFlag;    // Data ready flag
volatile unsigned long g_vui8ErrorFlag;    // Error flag

//*****
void
ISL29023AppCallback(void *pvCallbackData, uint_fast8_t ui8Status)
{
    if(ui8Status == I2CM_STATUS_SUCCESS)
    {
        g_vui8DataFlag = 1;
    }
    g_vui8ErrorFlag = ui8Status;
}
//*****

//*****
void
ISL29023I2CIntHandler(void)
{
    I2CIntHandler(&g_sI2CInst);
}
//*****

//*****
void
ISL29023AppErrorHandler(char *pcFilename, uint_fast32_t ui32Line)
{
    while(1)
    {
    }
}
//*****

//*****
void
ISL29023AppI2CWait(char *pcFilename, uint_fast32_t ui32Line)
{
    while((g_vui8DataFlag == 0) && (g_vui8ErrorFlag == 0))
    {
    }
    if(g_vui8ErrorFlag)
    {
        ISL29023AppErrorHandler(pcFilename, ui32Line);
    }
    g_vui8DataFlag = 0;
}
//*****

```

```

//*****
int
main(void)
{
    float fAmbient;
    uint8_t ui8Mask;

    ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);

    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_I2C3);
    ROM_GPIOPinConfigure(GPIO_PD0_I2C3SCL);
    ROM_GPIOPinConfigure(GPIO_PD1_I2C3SDA);
    GPIOPinTypeI2CSCL(GPIO_PORTD_BASE, GPIO_PIN_0);
    ROM_GPIOPinTypeI2C(GPIO_PORTD_BASE, GPIO_PIN_1);

    ROM_IntMasterEnable();

    I2CMinInit(&g_sI2CInst, I2C3_BASE, INT_I2C3, 0xFF, 0xFF, ROM_SysCtlClockGet());

    SysCtlDelay(SysCtlClockGet() / 3);

    ISL29023Init(&g_sISL29023Inst, &g_sI2CInst,
                ISL29023_I2C_ADDRESS, ISL29023AppCallback, &g_sISL29023Inst);
    ISL29023AppI2CWait(__FILE__, __LINE__);

    ui8Mask = (ISL29023_CMD_I_OP_MODE_M );

    ISL29023ReadModifyWrite(&g_sISL29023Inst, ISL29023_O_CMD_I, ~ui8Mask,
                            (ISL29023_CMD_I_OP_MODE_ALS_CONT),
                            ISL29023AppCallback, &g_sISL29023Inst);

    ISL29023AppI2CWait(__FILE__, __LINE__);

    while(1)
    {
        ISL29023DataRead(&g_sISL29023Inst, ISL29023AppCallback, &g_sISL29023Inst);
        ISL29023AppI2CWait(__FILE__, __LINE__);

        ISL29023DataLightVisibleGetFloat(&g_sISL29023Inst, &fAmbient);
    }
}

```

8. Eliminate unused interrupts from startup_ccs.c

The original light sensor project used several interrupts that we will not be using. We need to eliminate them from the `startup_ccs.c` file.

- ▶ Double-click on `startup_ccs.c` in the Project Explorer pane to open it for editing in the Editor pane
- ▶ Find the external declarations around line 59. **Comment out** the first, third, fourth and fifth as shown below.

```
//extern void IntGPIOe(void);  
extern void ISL29023I2CIntHandler(void);  
//extern void SysTickIntHandler(void);  
//extern void UARTStdioIntHandler(void);  
//extern void RGBBlinkIntHandler(void);
```

- ▶ Page down to around line 77.

The system exception and peripheral interrupt vectors start here.

- ▶ Find `IntDefaultHandler` and double-click on it to select it. Then press Ctrl-C to copy it to the clipboard.

This handler is the one that is called when an unexpected interrupt occurs. In a production environment, you might want to change the “trap” behavior of this code.

- ▶ Around line 91, find `SysTickIntHandler`. Double-click on it and press Ctrl-V to replace it with `IntDefaultHandler`.

- ▶ Do the same to:

```
IntGPIOe at about line 96,  
UARTStdioIntHandler at about line 97, and  
RGBBlinkIntHandler at about line 197.
```

- ▶ Save your work.

9. Build and Download your Project

Make sure that your LaunchPad/SensorHub combination is connected from a free USB port on your PC to the emulation port on the LaunchPad.

► Cycle the power on the board by moving the power switch from the DEBUG (right-most) position to the DEVICE (left-most) position and back to the DEBUG (right-most) position.

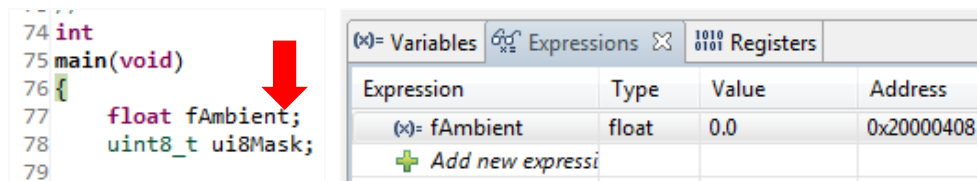
► Build and download the program to the flash memory of the TM4C123GH6PM by clicking on the Debug button on the CCS menu bar.



10. Set Watch Expressions and Breakpoints

► Click on the **Expressions** tab in the Watch and Expressions pane. If there are any Expressions in the window, right click in the window and select **Remove All**.

► Find **fAmbient** in the **light_isl29023.c** code pane (right after **main**) and double-click on it to select it. Right-click on it and select **Add Watch Expression ...** Click **OK** to leave the name as-is.



► Page down to the end of **light_isl29023.c** and find the **while(1)** loop. Identify the line of code that contains **ISL29023DataLightVisibleGetFloat()**. Double-click in the blue area just left of the line number to set a breakpoint on this line.

You'll see a blue dot with a check mark appear. When code execution reaches this point, control will be returned to CCS (before the line runs).

Remember that the current drivers do not support setting breakpoints while the code is executing.


```

107 while(1)
108 {
109     ISL29023DataRead(&g_sISL29023Inst, ISL29023AppCallback, &g_sISL29023Inst);
110     ISL29023AppI2CWait(__FILE__, __LINE__);
111
112     ISL29023DataLightVisibleGetFloat(&g_sISL29023Inst, &fAmbient);
113 }
  
```

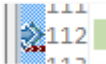
11. Run the Code

- ▶ Click the **Resume** button  or press F8 on your keyboard to run your code.

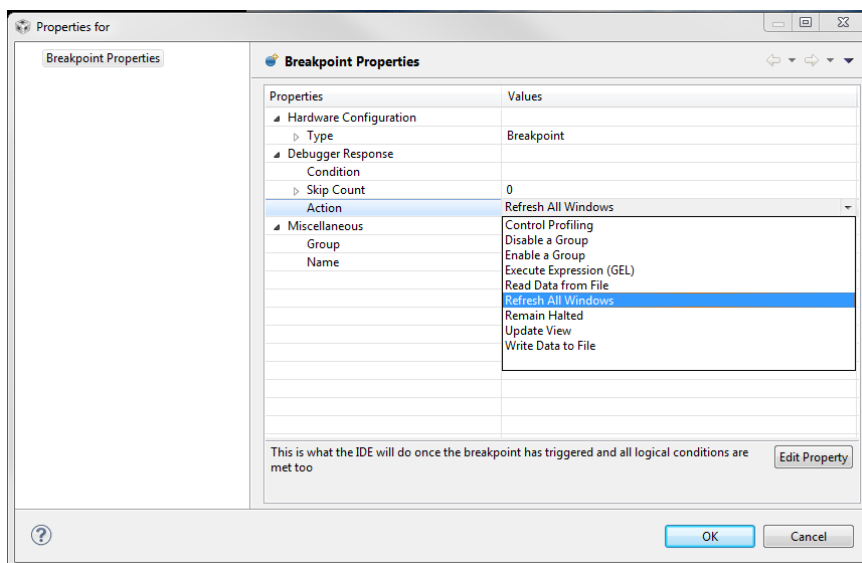
Since the breakpoint is set before the `ISL29023DataLightVisibleGetFloat()` API was run, `fAmbient` was not updated.

- ▶ Click the Resume button  or press F8 on your keyboard repeatedly.

Continuously clicking the Resume button can get pretty tedious. We can change the behavior of the breakpoint we set so that it doesn't stay halted.

- ▶ Right-click on the breakpoint symbol (it will now have a blue arrow on  it) and select Breakpoint Properties ...

- ▶ On the row containing Action, click on the Remain Halted value. When the down-arrow appears on the right, click on it. Select **Refresh All Windows** from the list and click **OK**.



- ▶ Click the **Resume** button  or press F8 on your keyboard to run your code.

Now the `while(1)` loop will run to the breakpoint, stop, update the `fAmbient` value in CCS and re-start code execution. Based on how the code is written in the `while(1)` loop, this will happen as quickly as possible (as soon as the I²C communication is finished, another will begin). Note that every time the value changes, CCS will highlight it in yellow.

- ▶ Pass your hand over the SensorHub or shine a bright light on it and watch the value of `fAmbient` change.

- ▶ Note the maximum value of `fAmbient` here: _____

Set up GUI Composer to Visualize our Data

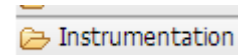
Earlier in the workshop we used the CCS graphing feature to visualize our data. TI introduced a new feature in CCS version 5.3 called “GUI Composer”. Let’s use it to visualize the data from the light sensor.

- ▶ Click the Suspend button to halt your program.

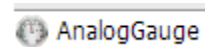
From the CCS menu bar, ▶ click View – GUI Composer.

- ▶ When you see the New Project button, click it. Enter a name of your choice in the dialog and click OK

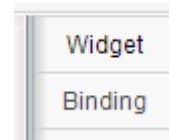
- ▶ When the GUI Composer tab and workspace appears, click Instrumentation on the left



- ▶ Find the AnalogGauge and drag it to the open design area. . Resize the gauge to make it as large as possible.



- ▶ Make sure the Widget is selected (click on it) and click the Widget tab on the far right. .



- ▶ Find the Title box and enter “Light Level” into it. Find the Maximum Value box and enter a value somewhat greater than the maximum value of `fAmbient` you noted in step 32.

- ▶ Click the Binding tab on the far right. In the Value box, enter `fAmbient`. Be careful with the spelling and case.

- ▶ Click the Resume button on the CCS menu bar.



- ▶ Click the Run button in the GUI Composer pane. Pretty cool, huh?



- ▶ Suspend the code and delete the breakpoint.
- ▶ Resume the code.



GUI Composer is capable of reading memory locations in the background through the emulator hardware. Since `fAmbient` is a global variable, we are assured that it has a memory location and has not been optimized into a register.

GUI Composer can be used inside of CCS or can be exported to run “stand-alone” without starting CCS. The steps to do this can be found in the GUI Composer documentation.

- ▶ Click the **X** in the upper-right corner of Code Composer Studio to close the program.
- ▶ Disconnect the USB cable from the LaunchPad and your computer.



You’re done with Lab 5(b)

(blank page)

Bluetooth® Wireless

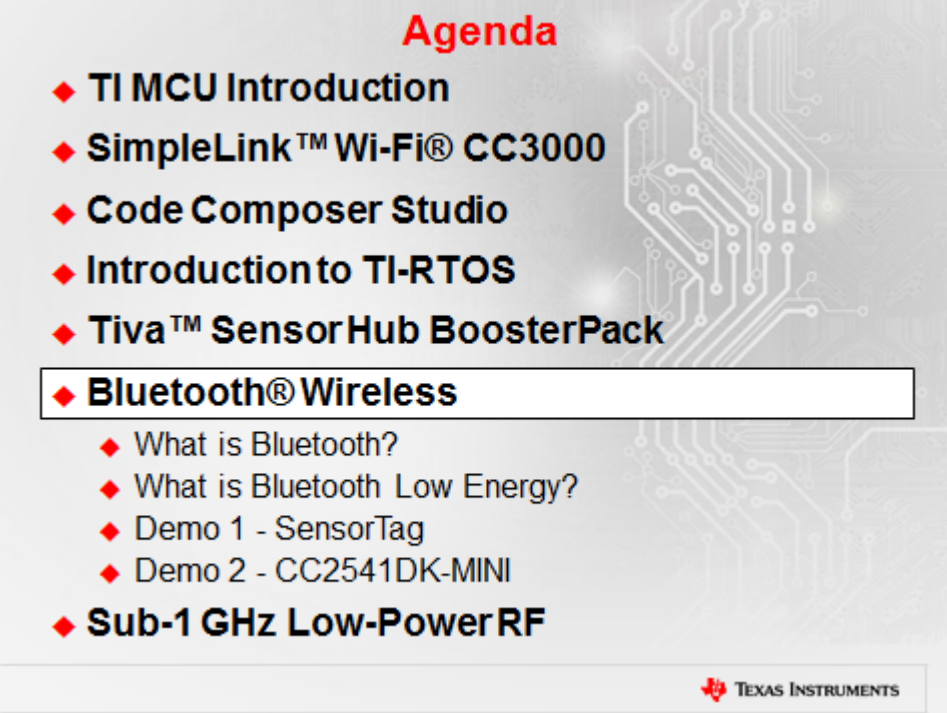
Introduction

TI has a full portfolio of Bluetooth (BT) offerings including Bluetooth Low Energy (Ble), Bluetooth Classic and Bluetooth dual mode with Bluetooth + Bluetooth Low Energy.

Bluetooth Low energy solutions are typically focused on sensor type applications or low data rate apps with short range and have longer battery life ... longer than one year on a coin cell

TI has two Bluetooth Classic offerings. The dual mode solution allows you to connect to newer devices that have BLE and to older phones/tablets that only have Bluetooth in them with a single solution. The Bluetooth classic solution is designed for audio applications or those that do not require Ble.

Learning Objectives



Agenda

- ◆ TI MCU Introduction
- ◆ SimpleLink™ Wi-Fi® CC3000
- ◆ Code Composer Studio
- ◆ Introduction to TI-RTOS
- ◆ Tiva™ SensorHub BoosterPack
- ◆ **Bluetooth® Wireless**
 - ◆ What is Bluetooth?
 - ◆ What is Bluetooth Low Energy?
 - ◆ Demo 1 - SensorTag
 - ◆ Demo 2 - CC2541DK-MINI
- ◆ Sub-1 GHz Low-Power RF











TEXAS INSTRUMENTS

Chapter Topics

Bluetooth® Wireless Technology.....	6-1
<i>Choosing the Right BT Device and MCU.....</i>	<i>6-3</i>
<i>What is Bluetooth?</i>	<i>6-4</i>
<i>What is Bluetooth Low Energy?</i>	<i>6-7</i>
<i>Demo 1 – Ble SensorTag.....</i>	<i>6-14</i>
Procedure.....	6-15
<i>Demo 2 - CC2541DK-MINI</i>	<i>6-18</i>
Objective	6-18
Procedure.....	6-19

Choosing the Right BT Device and MCU

Choosing the Right BT Device and MCU

Low Power Sensors	Data	Music
  <div style="display: flex; justify-content: center; align-items: center; gap: 10px;">  +  </div>	 <div style="display: flex; justify-content: center; align-items: center; gap: 10px;">  +  </div>	 <div style="display: flex; justify-content: center; align-items: center; gap: 10px;">  +  </div>
<ul style="list-style-type: none"> 1 year+ on Coin Cell Battery No Apple MFi Royalties Less than 100Kbps data rate 	<ul style="list-style-type: none"> Supports new and old phones (BT and BLE) No Apple MFi Royalties Up to 3Mbps data rate 	<ul style="list-style-type: none"> Use A2DP Profile to stream music from phones Use Tiva C Series or any other ARM Cortex M3/M4

What is Bluetooth?

What is Bluetooth?

Bluetooth characteristics

- ♦ Used to transfer data or audio
- ♦ Using 2.4Ghz - unlicensed frequency - used globally
- ♦ 10-100m range personal 'bubble' (Personal Area Network)
- ♦ Commonly used in computers, phones, and cars
- ♦ Good data rates (~2Mbps throughput)
- ♦ Installed base of ~6 billion Bluetooth devices



Why Bluetooth from TI?

Same great Best in Class Bluetooth solution

Based on TI's 7th Generation of Bluetooth

Available with new offerings

TI is now offering CC2560/64 Bluetooth QFN devices

Dual mode support for *Bluetooth + BLE* and *Bluetooth + ANT*

TI Bluetooth module offering has expanded to LSR, Murata, BlueRadios & Panasonic

Flexible *Bluetooth* stack supports more Profiles and MCUs

Offered in new package from TI



TI device Samples & Production Units Today

CC256x Bluetooth Devices



- TI sold and supported CC2560 & CC2564 devices with QFN Package - No minimum volumes

- Your one-stop shop

– Hardware

- TI QFN devices ([CC2560/CC2564](#))
- Datasheets and design guides
- Samples – [CC2560](#) & [CC2564](#)

– Software

- [Royalty Free Stonestreet One Bluetooth/BLE Software stack](#)
- Sample applications in source

– Evaluation Tools & Support

- [Bluetooth RF evaluation tool](#)
- [Comprehensive Bluetooth Wiki](#)
- [E2E Support Network](#)
- Evaluation kits, EM board

Part Numbers (orderable through TI and distribution)	CC2560: BT 4.0 CC2564: BT 4.0 Dual mode, or BT + ANT
Size	Device 7.83 x 8.10 mm QFN Design 16.5x16.5 mm
Temp Range	-40°C to +85°C
Tx Power	+ 12dBm ("Class 1.5")
Rx Sensitivity	- 95dBm
Host I/F	UART 4 wires H4
Audio I/F	PCM-I2S
IC Certification	BT SIG
Reference layout	EM Board, 4 layer
Ref Antenna	Printed PCB antenna
EM Certification	FCC, IC, CE

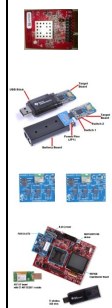
CC2560/4 Bluetooth® software

Stonestreet One Bluetooth software stack

Protocol communication Single Mode – Bluetooth Dual Mode – Bluetooth + Bluetooth Low Energy	<ul style="list-style-type: none"> • Bluetooth 4.0 • Bluetooth – SPP, HID profile & sample app • BLE – GATT profile/sample app; ANS, ANP, DIS, HTP, HTS, HRP, HRS, PASP
Hardware supported	• Works with TI devices and 3 rd Party Module solutions
Tool Chains supported	• IAR and CCS
OS	• No OS, scheduler
MCU supported	• Various MSP430 devices – 96k flash and up; ARM Cortex M4*
Download / Wiki	<ul style="list-style-type: none"> • Bluetooth stack, profile, sample app download – MSP430 & M4 • Bluetooth Wiki • MSP430 Bluetooth stack wiki • Stellaris LM4 Bluetooth Stack wiki

*For customers who are using Stellaris LM3 for Bluetooth audio, the new stack maintains the support for A2DP and AVRCP profiles. Please note that current Stellaris M4 devices do not support Bluetooth audio.

Getting Started with TI Bluetooth

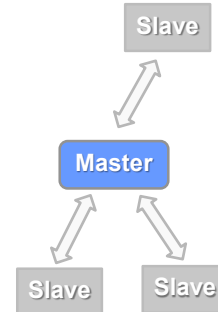


Kit Name Price	Technology	Kit Contents	Additional HW Needed
CC256xQFNEM Coming May 2013	Bluetooth, Bluetooth + BLE or Bluetooth + ANT	• 1 CC256x EM board with TI QFN device	CC256xBTBLE_EMK MCU board such as MSP-EXP430F5438 or MSP320F5529
EZ430-RF256x \$99	Bluetooth	• 2 Bluetooth target Boards • 1 USB emulator • 1 Battery board and batteries	None
PAN1323EMK \$99	Bluetooth, Bluetooth/BLE or Bluetooth/ANT	• 2 Bluetooth EM boards • 2 Jumpers for MSP-EXP430F5438 board	2x MSP-EXP430F5438 & Debugger or 2x MSP430F5529
CC2567-PAN1327ANT-BTKIT \$399	Bluetooth + ANT	• Sensor simulator (USB and module based on CC2571) • Bluetooth EM board • MSP430 Experimenter Board/USB debugger • TI wireless USB stick	None

What is Bluetooth Low Energy?

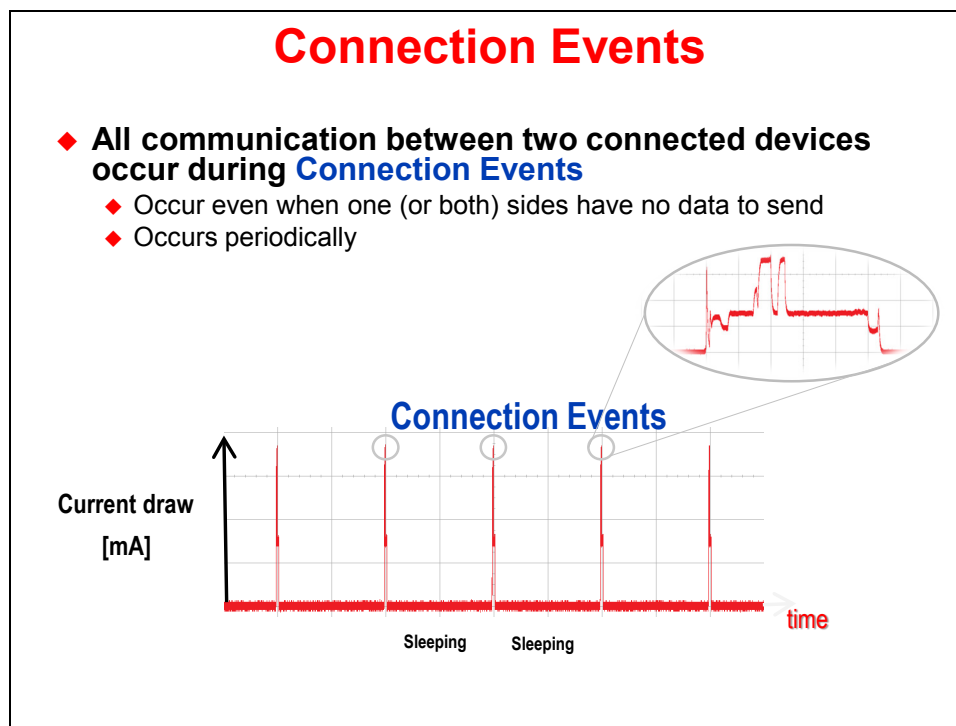
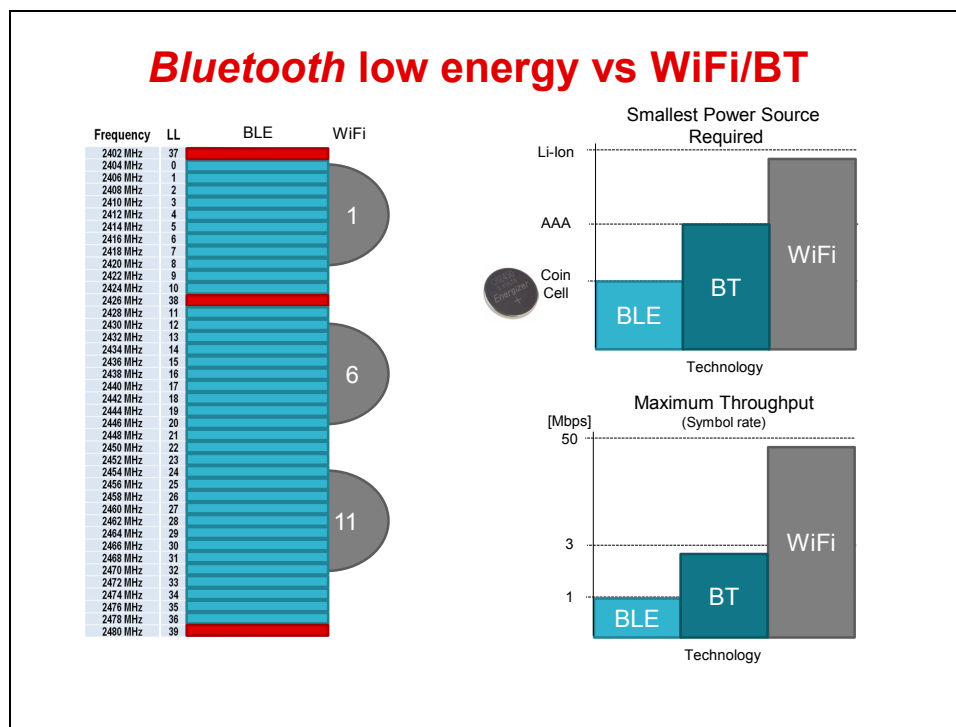
What is Bluetooth Low Energy?

- ◆ Ble is part of the *Bluetooth 4.0* specification
- ◆ Wireless Personal Area Network technology
- ◆ Target Applications:
 - ◆ Low Power
 - ◆ Low Latency
 - ◆ Low Throughput
- ◆ Spread Market and Application Areas
 - ◆ Entertainment, Sports & Fitness
 - ◆ Home Automation, Security & Proximity
 - ◆ Medical, Industrial & Automotive



Bluetooth Branding





Ble Device Roles

♦ A Bluetooth low energy device can operate in four profile roles:

♦ **Peripheral**

- ♦ An advertiser that is connectable
- ♦ Operates as a slave in a connection
- ♦ Example: *Heart Rate Monitor*

♦ **Central**

- ♦ Scans for advertisements and initiates connections
- ♦ Operates as a master in connections.
- ♦ Example: *Smartphone*

♦ **Broadcaster**

- ♦ An advertiser that is non-connectable
- ♦ Example: *Temperature Sensor*

♦ **Observer**

- ♦ Scans for advertisements, but cannot initiate connections
- ♦ Example: *Temperature Display*



TI Ble Hardware



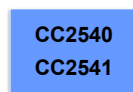
CC2540
Flash SoC
USB
Higher Output Power



CC2541
Flash SoC
I2C
Power optimized
Radio
Proprietary Mode

Two ways of using TI Bluetooth low energy

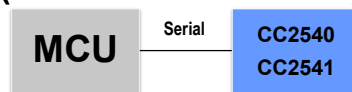
One-Chip Solution



BLE Stack + Application + Profiles

Two-Chip Solution


(Network Processor)




Application + Profiles

BLE Stack

TI Ble Hardware




CC2540
Flash SoC
USB
Higher Output Power




CC2541
Flash SoC
I2C
Power optimized Radio
Proprietary Mode

Development Kits




CC2540DK-MINI
CC2541DK-MINI

The easiest way of evaluating *Bluetooth* low energy




CC2540DK
CC2541EMK

Advanced kit with several peripheral interfaces and features



CC2541DK-SENSOR

Targeting smartphone App developers. The SensorTag provides sensor data from six sensors



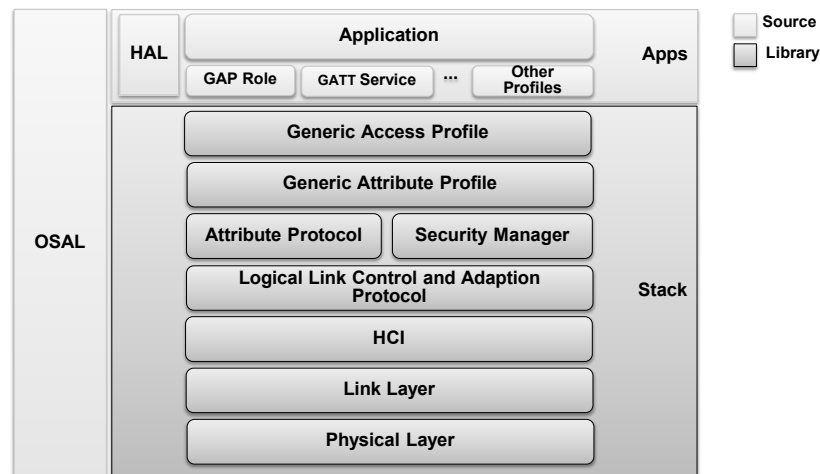
CC2541DK-RC

The Remote operates as a mouse, keyboard and consumer remote using HOGP

TI Ble Software Solution

- ◆ **Full Protocol stack for single mode *Bluetooth* low energy solution**
available at www.ti.com/ble-stack
- ◆ **Complete power optimized stack including controller and host**
 - ◆ Device Roles: Central, Peripheral, Observer or Broadcaster
 - ◆ GATT – Client and Server
 - ◆ Security Manager – 128 bit AES Encryption and Decryption
- ◆ **Sample applications and profiles**
 - ◆ Generic applications for all GAP roles and supported profiles.
 - ◆ The latest adopted SIG Profiles and example Proprietary Profiles
 - ◆ Over the Air Download (OAD)
- ◆ **Fully certified solution**
 - ◆ TI BLE Controller Subsystem
 - ◆ TI BLE Host Subsystem
 - ◆ TI BLE Profile Subsystem

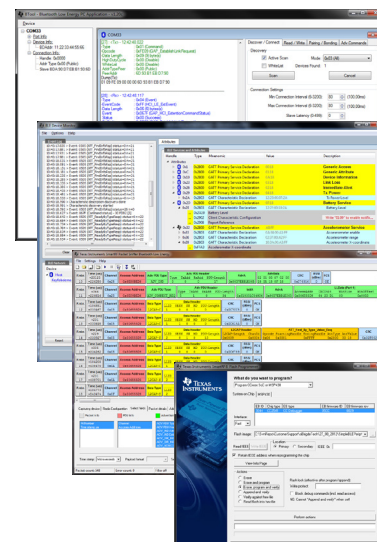
TI Ble Software Solution



OSAL = Operating System Abstraction Layer (Prioritized task handling loop)
 HAL = Hardware Abstraction Layer (Drivers and API for LEDs, Buttons etc)
 Full API to access all stack functionality in the stack (Library) from the Application and Profiles

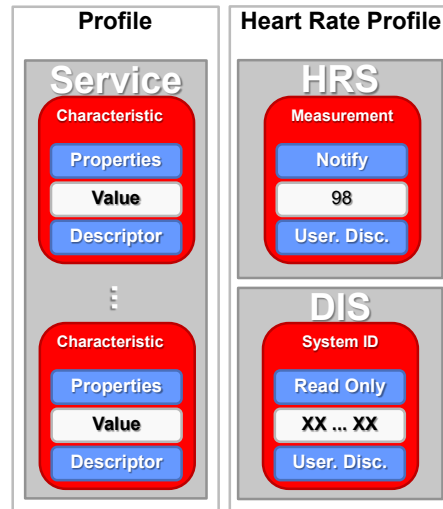
TI Ble Tools

- ◆ **BTool**
 - ◆ Run and test all possible *Bluetooth* low energy functionality controlled from the PC tool.
- ◆ **BLE Device Monitor**
 - ◆ Provides an intuitive and graphical way to explore *Bluetooth* low energy Services and Characteristics.
- ◆ **SmartRF™ Protocol Packet Sniffer**
 - ◆ Capture Bluetooth low energy communication live with full overview.
- ◆ **SmartRF™ Flash Programmer**
 - ◆ Program CC254x devices
 - ◆ Read and write IEEE addresses



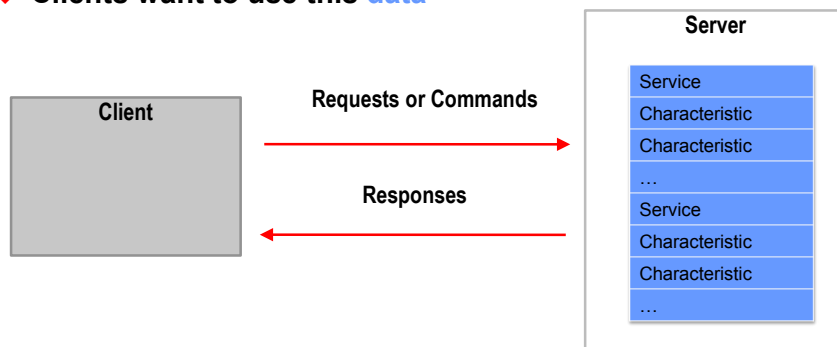
Profiles and Services

- ◆ A **Profile** defines and describes the use of **Services** necessary to implement a given Application
- ◆ **Bluetooth SIG adopted examples:**
 - ◆ **Heart Rate Profile**
 - ◆ Heart Rate Service (HRS)
 - ◆ Device Information Service (DIS)
 - ◆ **Proximity Profile**
 - ◆ Link Loss Service
 - ◆ Immediate Alert Service
 - ◆ **Find Me Profile**
 - ◆ Immediate Alert Service



GATT Architecture

- ◆ The **Generic Attribute Profile (GATT)** specifies the structure in which **data** is stored and exchanged.
- ◆ **Servers have data**, which are exposed using characteristics.
- ◆ **Clients want to use this data**



TI Ble Software Examples

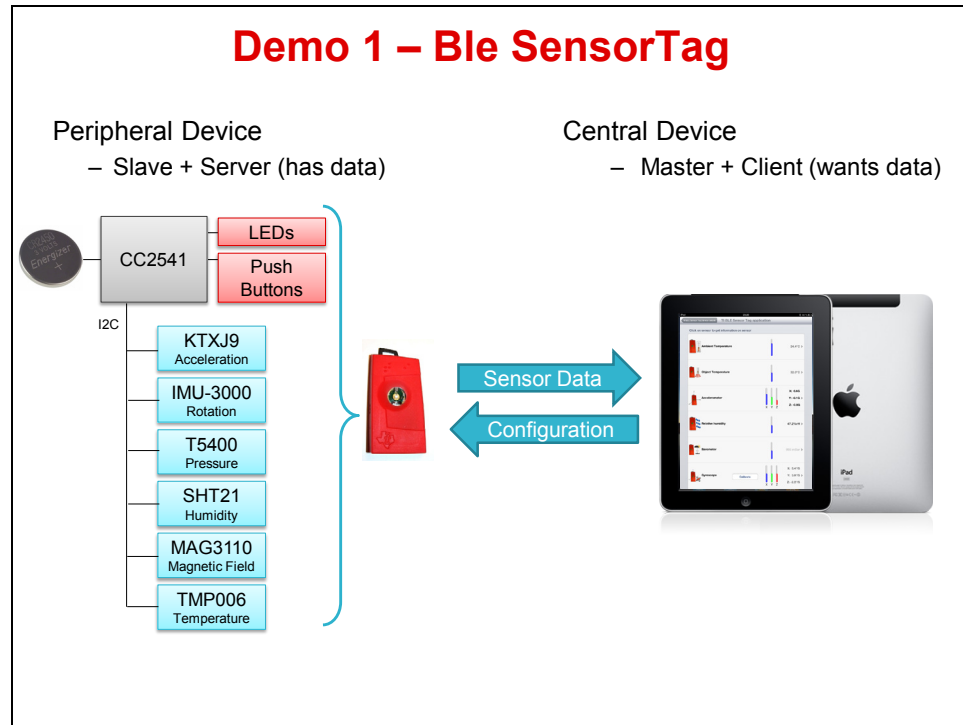
- ◆ Profile Specific
 - ◆ Heart Rate
 - ◆ Glucose
 - ◆ HID Keyboard
 - ◆ Etc.
- ◆ Generic
- ◆ SimpleBLEPeripheral
- ◆ SimpleBLECentral
- ◆ SimpleBLEBroadcaster
- ◆ SimpleBLEObserver



Bluetooth SIG adopted Profiles

Cycling Speed and Cadence		(BLEv1.4)
Running Speed and Cadence		(BLEv1.4)
Blood Glucose	- Certified (pending)	(BLEv1.2, February '12)
HID over GATT	- Certified	(BLEv1.2, February '12)
Proximity	- Certified	(BLEv1.1, July '11)
Find me	- Certified	(BLEv1.1, July '11)
Health thermometer	- Certified	(BLEv1.1, July '11)
Heart rate sensor	- Certified	(BLEv1.1, July '11)
Time	- Certified	(BLEv1.1, July '11)
Alert Notification	- Certified	(BLEv1.1, July '11)
Battery Status	- Certified	(BLEv1.1, July '11)

Demo 1 – Ble SensorTag



The SensorTag simplifies development of Bluetooth low energy sensor applications. It allows application developers to quickly and easily write smart phone apps for Ble accessories without any embedded hardware or software development.

To run this demo you will need a Ble equipped device. Currently Ble is supported on Bluetooth 4.0 enabled iOS devices like the:

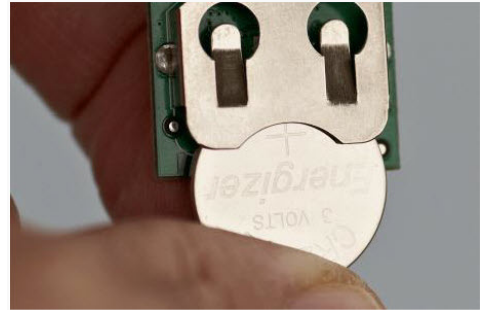
- iPhone 4S
- iPhone5
- iPod Touch (5th generation)
- iPad3 or newer
- MacBook Air (2011 model and newer)
- MacBook Pro (2012 model and newer)

There is currently no common Ble Application Program Interface (API) for Android devices.

Procedure

SensorTag Assembly

1. Remove the SensorTag hardware components from the kit and assemble them according to the included instructions. Insert the battery in the clip with the negative side facing the circuit board.
2. Press the side button and observe the on-board LED blinking to verify correct insertion. The battery clip may have a thin layer of solder residue that prevents contact. Remove and insert the battery a few times to power the SensorTag if necessary. Note that the LED flashing is fairly dim and fast ... you may need to cup your hand over the board to see the LED.
3. The included screw secures the board to the plastic case through the hole located on the circuit board. This is not absolutely required.
4. Close the transparent face of the case and insert the assembly into the silicone holder. Once done, the LED will not be visible.



Sensors

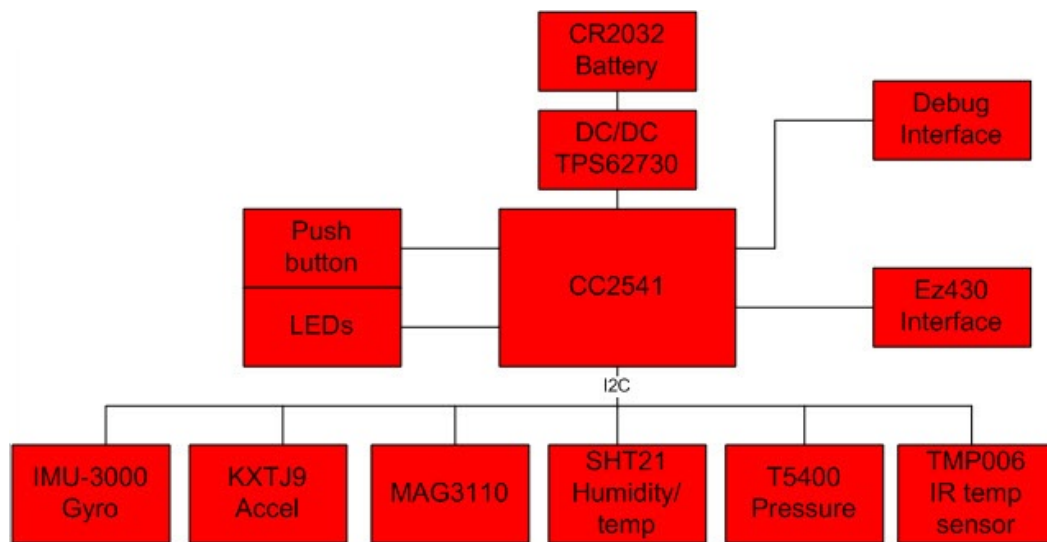
The SensorTag is fitted with six sensors connected to TI's CC2541. All of the sensors have been chosen to be small, energy efficient, low cost and surface-mount devices. They all use the same I²C interface with separate enable signals. To minimize current consumption, all of the sensors are disabled by default and are in sleep mode between measurements. In this way the sensors can be enabled and read individually or collectively.

The SensorTag includes the following sensors:

- The TMP006 Infrared Temperature sensor from Texas Instruments
- The SHT21 Humidity Sensor from Sensirion.
- The T5400 Pressure Sensor from Epcos.
- The KXTJ9 Accelerometer from Kionix.
- The IMU-3000 Gyroscope from Invensense.
- The MAG3110 Magnetometer from Freescale.

Block Diagram

The block diagram below shows the overall layout of the SensorTag design.



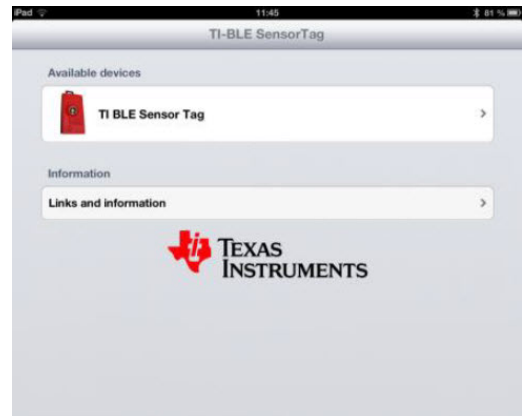
Software

- On your compatible iOS device, download and install the **TI Ble Multitool** application from Apple's App Store. Simply search for SensorTag on the App Store.

Connecting

In order for a Bluetooth 4.0 device to communicate with the SensorTag, Bluetooth must be turned on and the SensorTag must be advertising. In the settings on your device, assure that Bluetooth is enabled.

- Start the TI Ble Multitool application. When the app is launched it will search for all Ble devices in the area.
- On the SensorTag, enable advertising by clicking the side button. The SensorTag will advertise its presence for 30 seconds. If no Bluetooth 4.0 device is detected by the SensorTag within 30 seconds, advertising will stop and the SensorTag will go back to sleep. To re-activate advertising, press the side button once more.
- Back on your device, the SensorTag icon should appear indicating a successful connection.



Running the App

- The TI Ble Multitool allows you to customize the display by selecting the different sensors that you want to use. When connected to the SensorTag, click on the arrow next to the SensorTag symbol to view the sensor selection screen. Select the sensors you like by turning them on. Adjust the data collection interval by moving the sliding bars. You can also select which axis to display for the multi-axis sensors. When you click **Show application**, the app will display the selected sensor data with the selected data interval.
- To view and store the sample code of the selected sensor configuration select **Store sensor configuration**. Go back to **SensorTag device details** to select another combination of sensor data to display.

Source code for the SensorTag can be found by searching TI's wiki for **SensorTag**.

In addition, you can use the TI Bluetooth SensorTag app to read the services and attributed of any connected Ble device. When connected, select the arrow next to the device name and then select **Service discovered** and/or **Characteristics discovered**.

- Close the TI Ble Multitool app. The SensorTag will cease transmitting after it is no longer connected and return to a low power state.

Ble Device Monitor

If you have a Windows 7/8/XP PC and a CC2540 USB dongle you can download and install TI's Ble Device Monitor from www.ti.com/SensorTag. This application allows you to discover, read and alter attributes on any Ble device. More detailed information can be found on the website.

Demo 2 - CC2541DK-MINI



Objective

The CC2541 mini development-kit provides a great out-of-box experience and in the next few steps we will show how to get a Bluetooth low energy solution up and running in matter of minutes.

In this demonstration, you can use either the BTool Windows application or you can use the TI Ble MultiTool on selected iOS devices like|

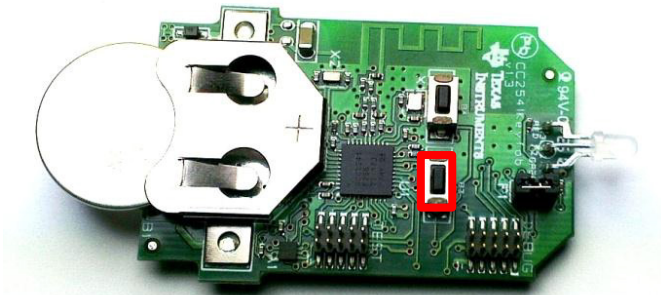
- iPhone 4S
- iPhone5
- iPod Touch (5th generation)
- iPad3 or newer
- MacBook Air (2011 model and newer)
- MacBook Pro (2012 model and newer)

Procedure

Hardware

12. Power up the CC2541 keyfob by inserting CR2032 battery into the holder. Note that the positive side should be oriented **away** from the circuit board. The LED will light green for one second.

13.



14. The CR2032 battery is a non-rechargeable lithium battery. Make sure to remove the battery from the keyfob when it is connected to an external power source. Failure to do so can result in injury and/or fire.

Using the BTool

15. Download, unzip and install the latest BLE-STACK from www.ti.com/ble-stack. You will need a my.TI account to complete this. The default installation location is:

C:\Texas Instruments\BLE-CC254x-1.3.2.

The installation will place a shortcut on your desktop.

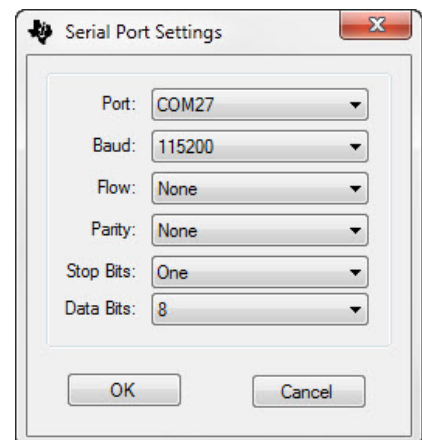


16. The IAR project and source code for this demo (Keyfobdemo) are included in this download. A debugger interface and cable are included in the kit.

17. Connect the dongle in the kit to a free USB port or USB hub port. If the driver does not install automatically, update the driver using the Device Manager. You can find the driver in the default installation folder in the *Accessories/Driver* folder. The device will install in the Ports area as **TI CC2540 USB CDC Serial Port**.

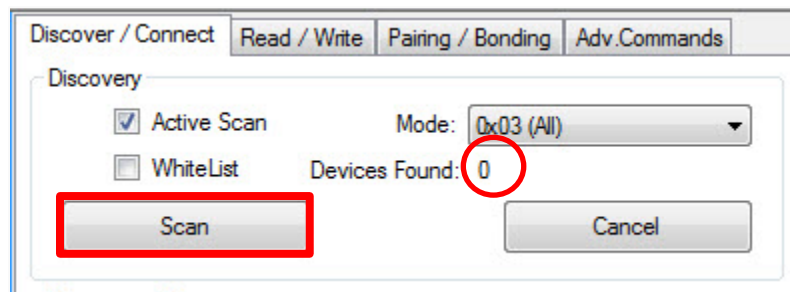


18. If you don't already have the Device Manager open, open it now and note the COM port number of the TI CC2540 USB CDC Serial Port here: _____
19. Start the **TI BTool** by double-clicking on its shortcut on your desktop. Make the selections shown and click OK. Your COM number will likely be different.

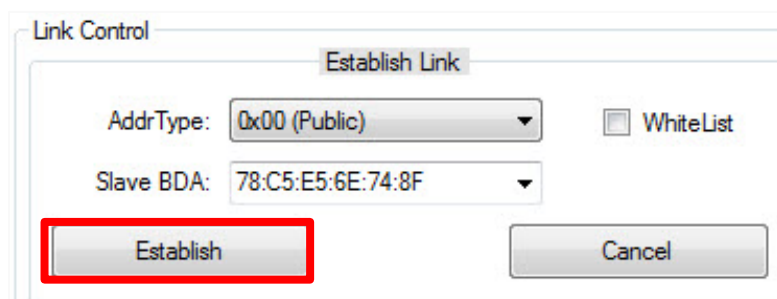


20. Press button **S3** on the keyfob to start Bluetooth advertising. Bluetooth advertisements can be toggled on and off by pressing this button. During advertisement, the LED will blink red. The keyfob will remain discoverable for 30 seconds and will return to the standby mode afterwards.

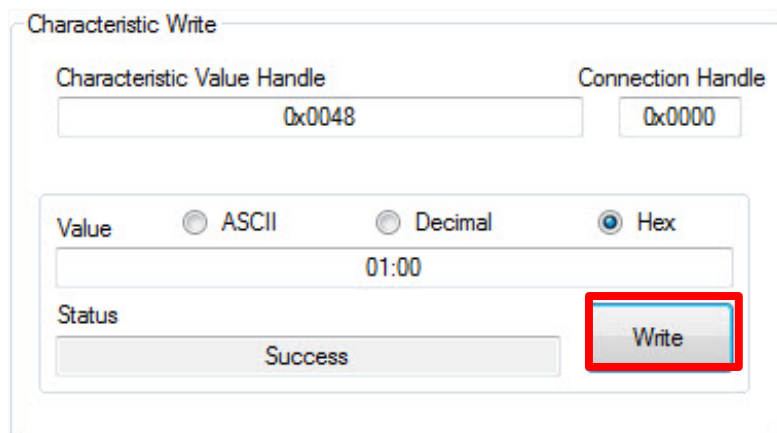
21. While the keyfob is advertising, click the Scan button in the **TI BTool** as shown:



22. After scanning is complete, note that the Device Found count will indicate 1. The **Slave BDA** address will appear as shown below (yours may be different). Click the **Establish** button to connect the keyfob and the dongle. The red LED will stop blinking.



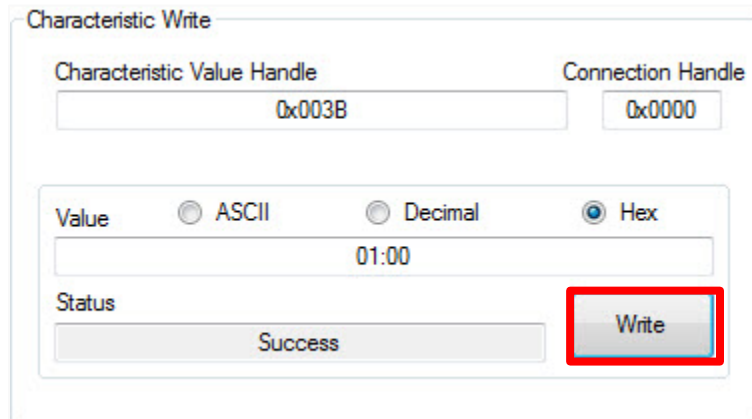
23. To observe when button are pressed on the keyfob, notifications must be enabled. Click on the Read/Write tab in the TI BTool. Write **Hex 01:00** to character handle **0x0048** as shown below in the Characteristic Write area. Enter the values and click the Write button.



24. Press buttons **S3** and **S4** on the keyfob and observe the notifications in the left-hand log window. Both the press and release events will be logged for each button.

Similar to the button notifications, notifications can be enabled for the keyfob's on-board accelerometer. For this example we will only enable notifications for the x-axis.

25. First, the accelerometer must be enabled. Write **Hex 01:00** to characteristic handle **0x003B** as shown below. Enter the values and click the Write button.



Characteristic Write

Characteristic Value Handle: 0x003B Connection Handle: 0x0000

Value: ☐ ASCII ☐ Decimal ☒ Hex

01:00

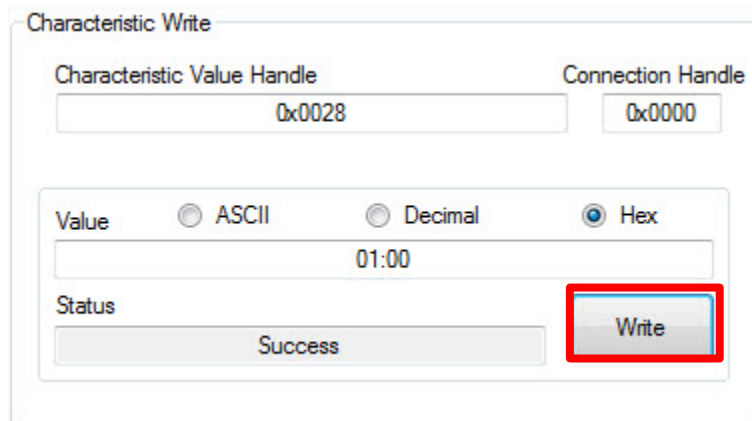
Status: Success

Write

26. Rotate the keyfob back and forth in the same plane as the CR2032 battery. Observe the notifications in the left-hand log window of TI BTool.

27. The buzzer on the keyfob can be triggered by writing the following Hex values to characteristic handle **0x0028**:

- **01:00** for low alert
- **02:00** for high alert
- **00:00** to turn off alert



Characteristic Write

Characteristic Value Handle: 0x0028 Connection Handle: 0x0000

Value: ☐ ASCII ☐ Decimal ☒ Hex

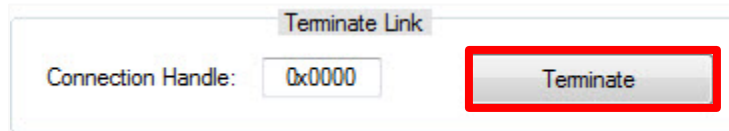
01:00

Status: Success

Write

28. Terminate the connection. There are three options:

- Press the Terminate button under the Discover/Connect tab in TI BTool as shown below.
- Remove the battery from the CC2541 keyfob, which will trigger a supervision timeout.
- Move the keyfob out of range (typically >10m), which will trigger a supervision timeout.



29. Close the **TI BTool** and remove the CR2032 battery from the CC2541 keyfob.

Evaluate using an iOS Device

30. If you haven't done so already, download the **TI BLE Multitool** from the Apple App store. Note the supported devices at the beginning of this demo ... your iOS device must support Bluetooth low energy to run this demo.
31. Insert the battery into the keyfob and press S3 to enable advertising. Run the TI BLE Multitool. It will automatically scan for devices.
32. When your TI BLE Keyfob is found, click the arrow to the right. This will run the keyfob application and give you access to the notifications and sensors.
33. Remember to remove the battery from the keyfob when you're done experimenting.

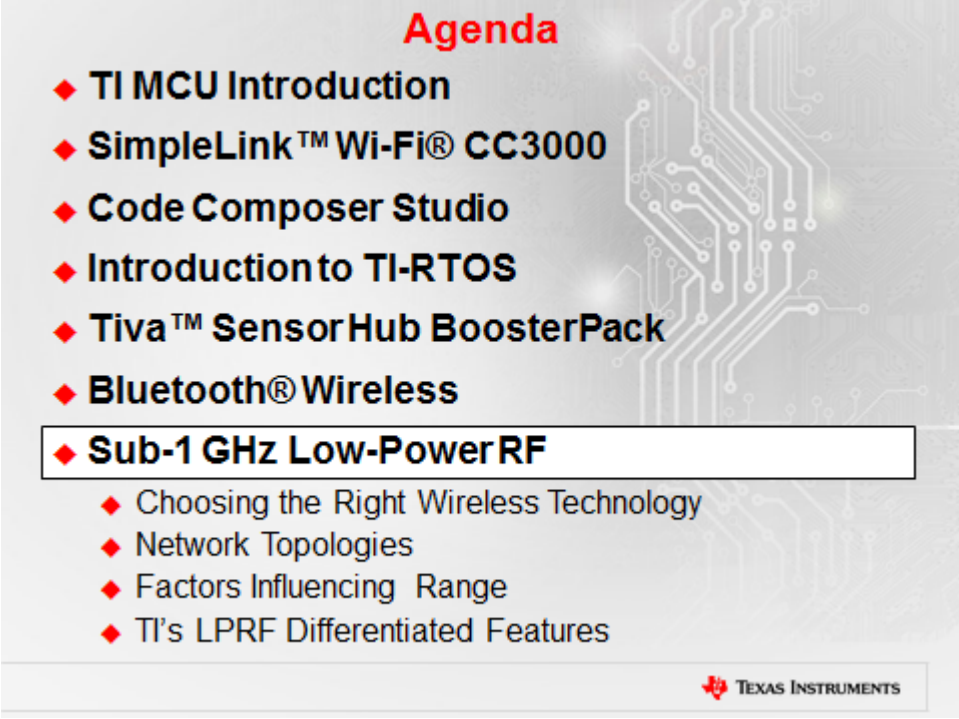
Sub-1 GHz Low-Power RF

Introduction

The purpose of this chapter is to provide an overall introduction to the Low-Power RF products available from TI's Wireless Connectivity Solutions portfolio. The chapter material covers the devices and some of the innovative features available on TI's sub-1 GHz lineup of products.

The lab that follows provides a simple demonstration of the CC110L device that uses Anaren's CC110L Air Module Booster Pack. The current lab uses Energia as the IDE which provides a simple build/load interface and the code is quite easy to understand and modify.

Agenda



Agenda

- ◆ TI MCU Introduction
- ◆ SimpleLink™ Wi-Fi® CC3000
- ◆ Code Composer Studio
- ◆ Introduction to TI-RTOS
- ◆ Tiva™ SensorHub BoosterPack
- ◆ Bluetooth® Wireless
- ◆ **Sub-1 GHz Low-Power RF**
 - ◆ Choosing the Right Wireless Technology
 - ◆ Network Topologies
 - ◆ Factors Influencing Range
 - ◆ TI's LPRF Differentiated Features

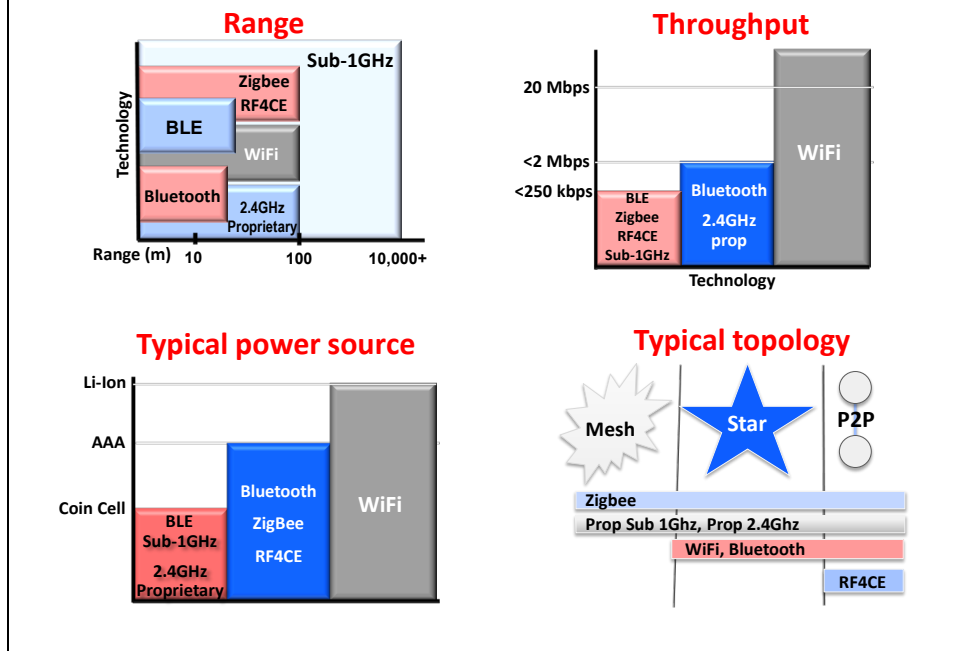
TEXAS INSTRUMENTS

Chapter Topics

Sub-1 GHz Low-Power RF	7-1
<i>Chapter Topics</i>	<i>7-2</i>
<i>Choosing the Right Wireless Technology</i>	<i>7-3</i>
<i>Network Topologies</i>	<i>7-6</i>
<i>Factors Influencing Range</i>	<i>7-7</i>
<i>TI's LPRF Differentiated Features</i>	<i>7-8</i>
<i>Lab 7 – System Setup.....</i>	<i>7-12</i>
<i>Lab 7 Procedure.....</i>	<i>7-13</i>
Hardware Setup	7-13
Example 1 – Wireless Test	7-14
Example 3 – Change LED colors.....	7-20
That's it, You're Done.....	7-20

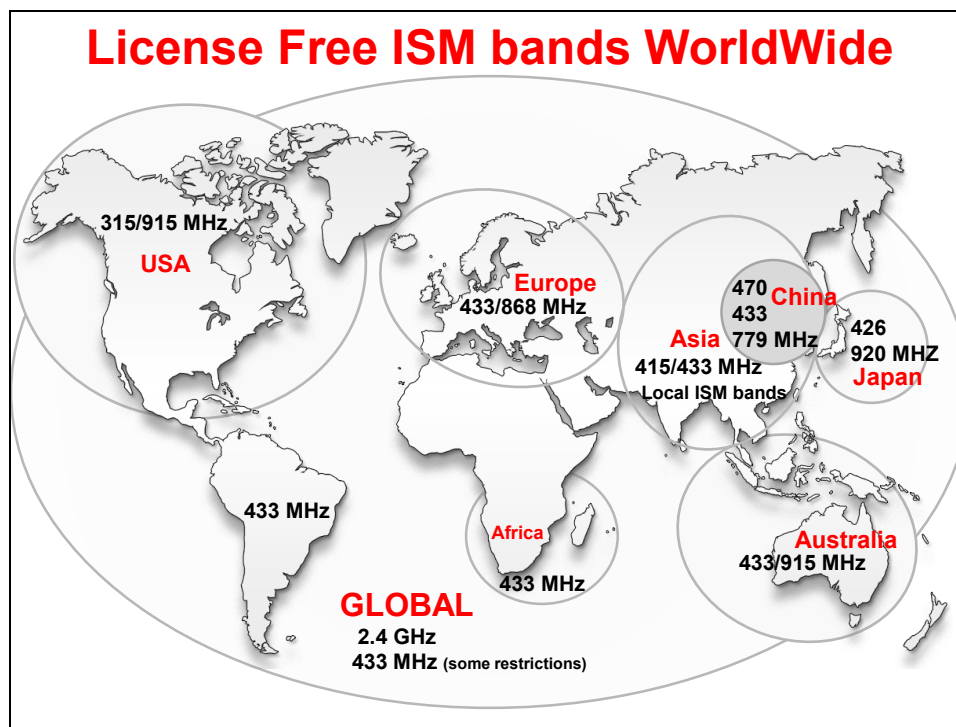
Choosing the Right Wireless Technology

Choosing the Right Wireless Technology



Frequency Band

Band	Advantages	Disadvantages
5 GHz	Highest Data Rate Least crowded	Less Range
2.4 GHz	Works worldwide High data rate Full duty cycle allowed	Most crowded
< 1GHz	Best range Less crowded	Lowest data rate Restrictions on duty cycle at some freq's



Proprietary vs. Industry Standard

Technology	Advantages	Disadvantages	Examples
Proprietary	<ul style="list-style-type: none"> Tailored to the application Specialized fxns Smaller s/w stack Simpler deployment Choice of freq bands 	<ul style="list-style-type: none"> Each OEM's topology is different Fewer options among suppliers 	Sub 1GHz 2.4 GHz PurePath
Industry Standard	<ul style="list-style-type: none"> Inter-operability among different suppliers Standardization – customer choice of suppliers Ease of network expansion 	<ul style="list-style-type: none"> Larger software stack in most cases Potentially higher power consumption 	Zigbee Bluetooth BLE ANT 6LoWPAN Wi-Fi RF4CE

How to Choose the Right Wireless Technology

Question	Answer	Technology
What's your application?	Data	All
	Audio	WiFi, Bluetooth, Proprietary 2.4G
	Video	WiFi
What are you connecting to?	Phone	WiFi, Bluetooth, BLE
	Internet	WiFi
	Other	All
What's the transmit range?	Personal	WiFi, Bluetooth, BLE
	Home	WiFi, ZigBee, Sub 1Ghz
	Outdoor	Sub 1Ghz
How many units in network?	≤10 Units	All
	>10 Units	ZigBee, Sub 1Ghz, Proprietary

Why Sub-1GHz RF Technology ?

Long Range

- Radio range, up to several kilometers
- New technology from TI makes it more reliable

Co-existence

Rapidly growing number of RF devices requires good co-existence properties

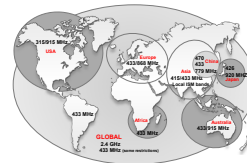
License Free Radio Communication

ISM (Industrial, Scientific and Medical)

- License free RF bands
- Available worldwide
- No need for subscription or fees

Battery life

On small Lithium cells, designed to last up to 15 years

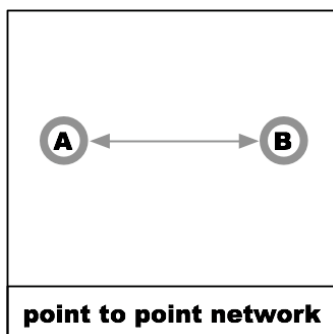


Network Topologies

Network Topology Defined (1)

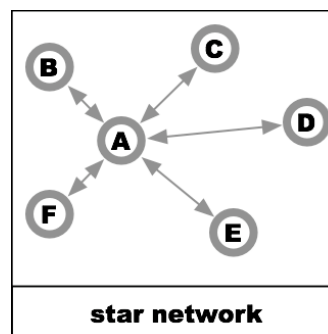
Point to Point:

- one way or two way communication
- simple protocol using **SimpliciTI** or **TIMAC**



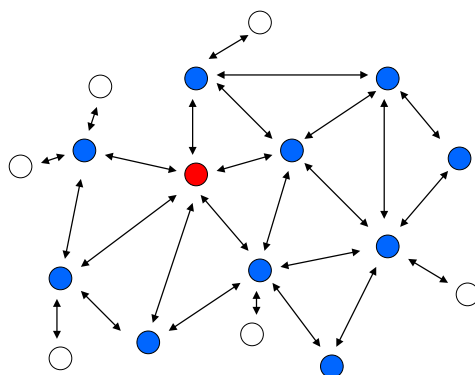
Star Network w/Multiple Nodes:

- Host device with hub function
- Simple end devices



- A = Access Point
- B-D = End devices

Network Topology Defined (2)



- ◆ Devices are pre-programmed for their network function
- ◆ Coordinator can be removed

Coordinator

- Starts the Network
- Routes packets
- Manages security
- Associates Routers & End Devices
- Example: Heating Central

Router

- Routes packets
- Associates Routers & End Devices
- Example: Light

End Device

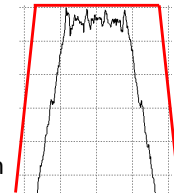
- Sleeps most of the time
- Can be battery powered
- Does not route
- Example: Light switch

Factors Influencing Range

Factors that Influence Range (1)

Sensitivity

- Sensitivity is the minimum received signal power required to demodulate the transmitted information
 - CC1120 has excellent sensitivity of -129 dBm
- Transmitted signal has a certain signal bandwidth
 - Higher data rate typically means wider signal bandwidth
- The receiver's filter BW must be wide enough to fit the signal bandwidth and allow some margin for crystal inaccuracies
- Receiver Noise Floor = $-174 + 10\log_{10}(BW)$ [dBm]



Noise Floor

1 MHz channel
-114 dBm

100 kHz channel
-124 dBm

10 kHz channel
-134 dBm

Factors that Influence Range (2)

Output Power

- Higher output power gives longer range, but maximum output power is limited by regional regulations
- For increased output power and range use CC1190 (up to +27 dBm)

Environment

- Theoretical range is only possible for line-of-sight. Several factors such as obstructions, reflections, multipath fading, etc, will limit range

RF Frequency

- RF frequency typically gives better range and penetration for the same sensitivity, output power, and antenna gain (i.e. for same link budget)

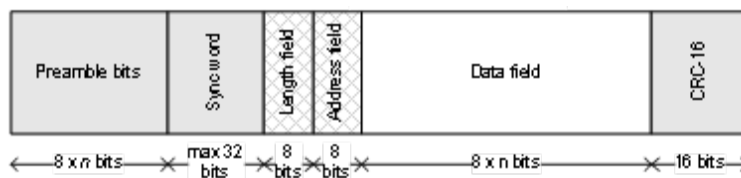
Antenna Performance

- Antenna gain is typically much lower at lower frequencies (e.g. 169 MHz compared to 868/915 MHz). Ground plane size affects antenna gain

TI's LPRF Differentiated Features

On-chip Packet Handling Features (Tx)

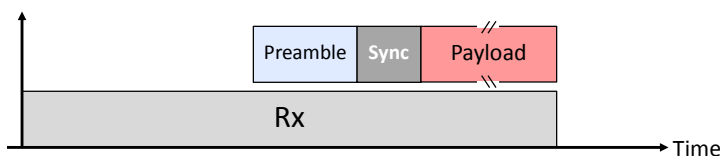
- ◆ Packet-handling capabilities that are built INTO the hardware:
 - Puts less burden on the MCU
 - MCU can be in low-power mode until packet interrupt
- ◆ In Tx mode, the packet handler can add the following elements:
 - Programmable number of preamble bytes
 - Synchronization word
 - Compute and add CRC checksum



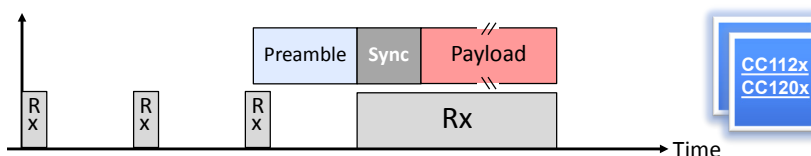
- ◆ User only needs to provide data payload
- ◆ MCU can sleep or perform other tasks during Tx state (significant reduction in power consumption)

WaveMatch Feature Reduces Power

- ◆ **Traditional Receiver**
 - ◆ Radio must stay in RX continuously to ensure transmitted packet is received
 - ◆ This draws power during the entire Rx sequence



- ◆ **Performance Line – WaveMatch Receiver**
 - ◆ Fast settling receiver automatically duty-cycles Rx to greatly reduce power
 - ◆ Receiver only “wakes up” at Sync signal – does not sacrifice RF performance



Noise Immunity & Settling Time

◆ Traditional Receiver

- ◆ Bit-based sync word detector – needs longer preamble of 4 bytes
- ◆ Slow sync detection – can detect false sync signal in noise



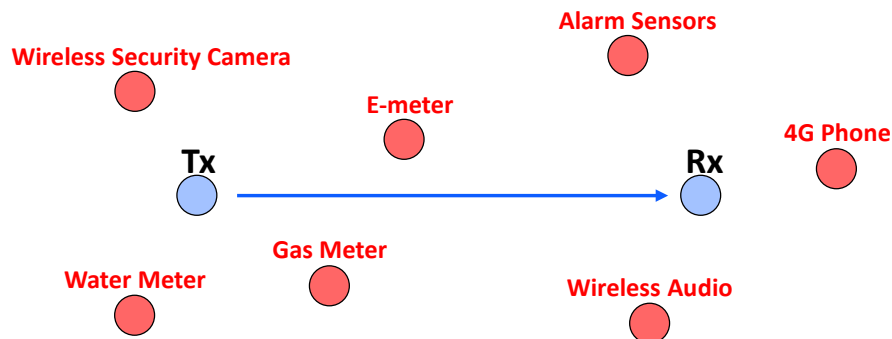
◆ TI Sub-1GHz Performance Line – WaveMatch Receiver

- ◆ Short preamble (0.5 bytes) – very quick sync detection
- ◆ No false sync detection in noise and lower power operation



Range and Co-Existence

- ◆ Interference can severely limit communication range
- ◆ Two key factors influence RANGE:
 - Sensitivity: the receiver's ability to decode a "weaker" signal – the higher the sensitivity, the higher the range
 - Output Power: more power, more range – but limited by RF regulations
- ◆ Good Co-existence = first pass installation success

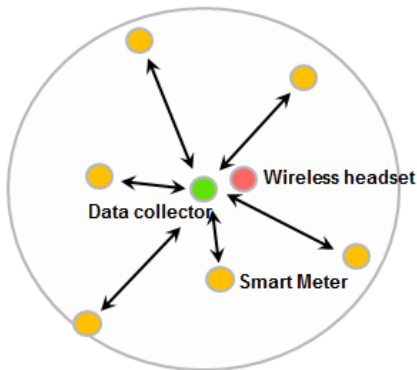


Co-Existence – How it Works

◆ Selectivity and Blocking translate to...

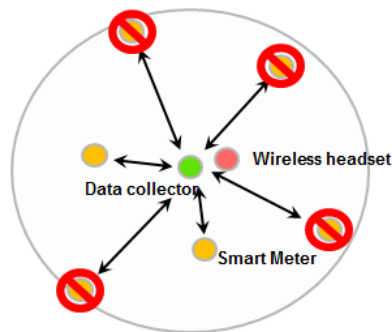
◆ TI Performance Line

- ◆ No range reduction for data collector
- ◆ Fewer data collectors needed
- ◆ Longer links possible
- ◆ Fewer re-transmissions



◆ Traditional Receiver

- ◆ Data collector range is reduced
- ◆ More data collectors needed
- ◆ Interference results in higher packet loss
- ◆ More re-transmissions needed



Why TI Sub-1GHz Technology

Best Range and Co-Existence

- Fewer nodes needed for network coverage → Lower system cost
- Reliable RF link → Robust solution
- Fewer re-transmissions → Longer Battery Life

Longest Battery Life

- Sub-3mA average RX current → operate for years on single battery
- Short start-up times → longer battery life
- Highly noise immune → No false system wakeups

15 years of experience: Tools and Support

- Development Kits (HW) and PC Tools (SW) → Jump-start development
- Reference designs → Faster time to market
- RF and Antenna design support → System optimization

For More Information...

◆ Performance Line: www.ti.com/rfperformanceline

◆ Value Line: www.ti.com/rfvalueline

◆ E2E forum: www.ti.com/e2e

◆ "Getting Started on Sub-1GHz" training:
Available on TI training portal

◆ Wireless Connectivity Selection Guide:
www.ti.com/wirelessconnectivityguide



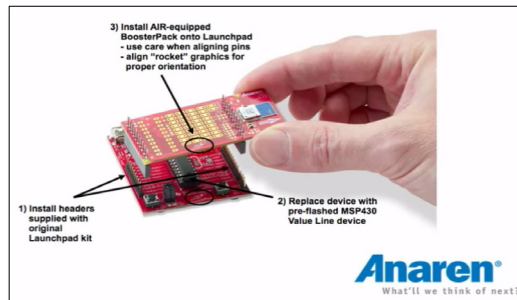
Lab 7 – System Setup

This lab will provide a quick demonstration of the CC110L Air Booster Pack available from Anaren that currently only runs on the Tiva-C LaunchPad. In the near future, the MSP430F5529 LaunchPad will be supported as well.

Students will have the opportunity to pair up and send/receive messages back and forth between their boards and in a later part of the lab you can actually blink the other person's LED.

LPRF Lab – Using Anaren AIR Module

- ◆ Team up with another Tiva-C user
- ◆ Instructor will determine “Channel #'s” to avoid conflict
- ◆ Teams will install the CC110 AIR Module from Anaren and then communicate between the two boards



- ◆ Example 1 – Load the wireless test example and run
- ◆ Example 2 – Blink each other's LED
- ◆ Example 3 – Modify receiver's LED color

Time: 45 min


Sub-1GHz
Robustness, long range

Lab 7 Procedure

In this lab, you will use Energia to build and run example code from Anaren. We have provided all the files you need for this lab in the c:\MCU Day\labs\Tiva\lab7 folder.

All of these files are readily available online and relatively easy to use. You can obtain them from <http://jmr2012.de/ener/AIR430BoostUSAFCC.zip>. This zip file from Anaren contains four examples and this lab only uses one of them with some slight modifications.

If you are interested in exploring more details about the Air Module, Anaren Products and TI support for these products, click on the links below:

<http://www.ti.com/tool/430boost-cc110l>

<http://www.anaren.com/air/air-boosterpack>

<http://www.anaren.com/air/software-demos-examples>

For more information about how to get started using Energia:

<https://github.com/energia/Energia/wiki/Getting-Started>

Hardware Setup

1. Find another student in the class who would like to do this lab with you.

This lab must be done in pairs.

► Ask a neighbor if they would like to join you on this adventure. You will both be using the Tiva LaunchPad board for this lab. Future versions of this lab will support MSP430F5529 LaunchPad as well.

This lab requires the following hardware:

► Please double check you have each item:

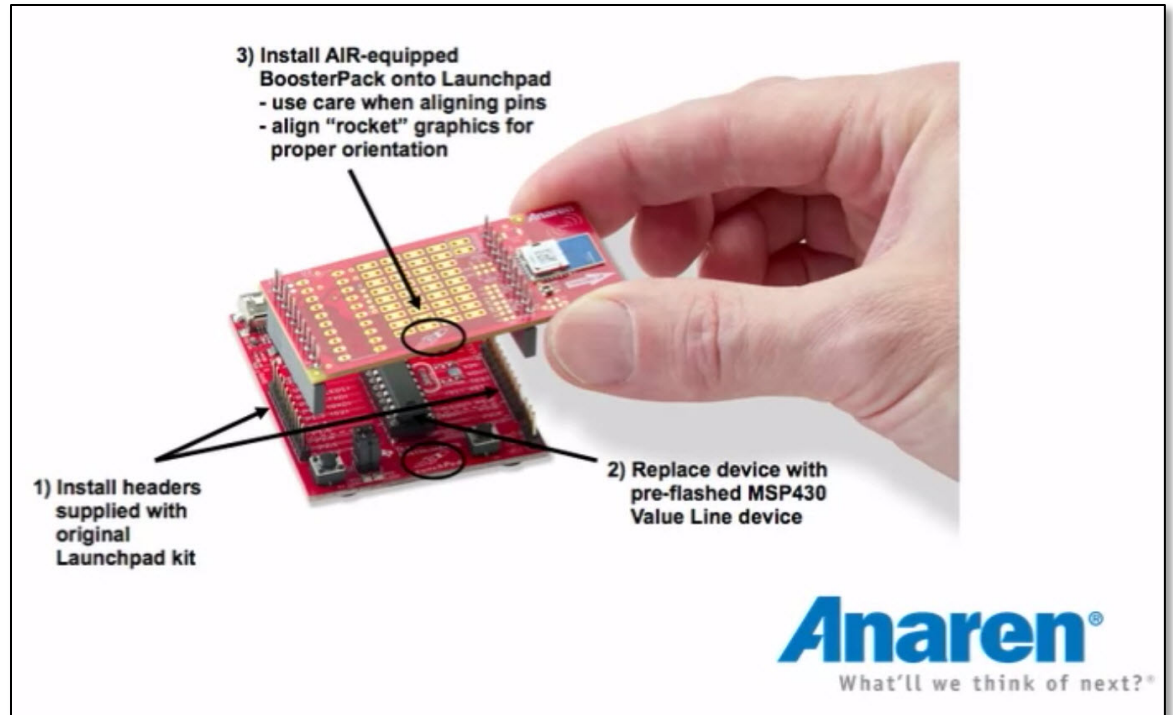
- Laptop loaded with Energia
- Tiva-C LaunchPad plus USB cable (one per user)
- CC110L Air Module Booster Pack from Anaren (one per user)

2. Connect the Anaren Air Module Booster Pack to the LaunchPad.

The following picture shows how to install the Anaren Air Module Booster Pack to the MSP430 LaunchPad (shown is the value line MSP430LP). Installing this on the Tiva LaunchPad is similar to this.

Note: Make sure the ROCKETS at the bottom of each board LINE UP as shown !

► Each student needs to connect the two boards together as shown:



3. Hook up the USB cable from the LaunchPad to your laptop.

- Each student should now hook up their USB cable to the PC if not already done.

Example 1 – Wireless Test

In this part, you are simply going to import the sketch of the Wireless Test and build/run it. First, you'll need to set a few settings in Energia, then we'll move on to the actual use of the example.

Note: Each student in the pairing should follow EVERY step as shown in the lab.

Configure Energia

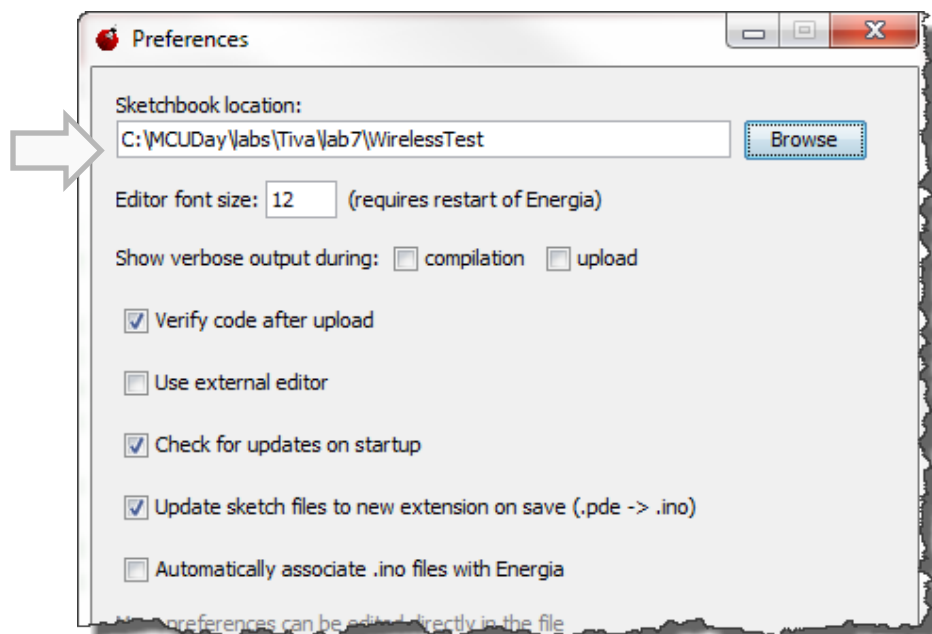
4. Configure Energia Settings.

- Launch the Energia software:

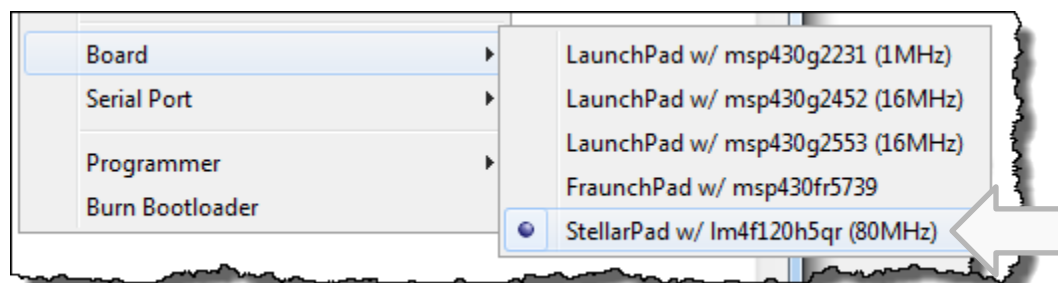


- Select *File* → *Preferences* and set your *Sketchbook* location to:

c:\MCUDay\labs\Tiva\lab7\examples\WirelessTest (as shown)

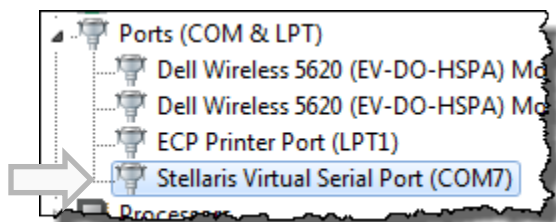


- Select *Tools* → *Board* → *StellarPad* (as shown):



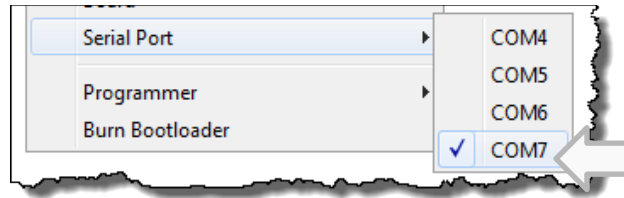
- Determine your COM port number by whatever means you have.

For Windows users, simply select the *Device Manager* and look for:



- Write down your COM # here: COM ____

- Select *Tools* → *Serial Port* → *Com__?__*:



- Close Energia and re-open it. This step is needed for Energia to recognize the new workspace path and the new `\Libraries` folder.

Import Wireless Test Example & Libraries

5. Import Wireless Test example.

- Select *File* → *Open* and navigate to:

`c:\MCUDay\labs\Tiva\lab7\examples\WirelessTest\WirelessTest.ino`

Analyze the Wireless Test Example

6. Let's take a little time to investigate what is going on in this example.

You now have, in front of you, the wiring language “code” that performs a wireless test between two LaunchPads for transmit and receive. It is important to understand the basics of what is going on so that in the next example, when you start blinking LEDs, you have a sense of how the code works.

All Energia (or Arduino) examples are required to have two functions – `setup()` and `loop()`. Users are able to create their own functions in addition to these, but expect to see both `setup()` and `loop()` in the code. This is analogous to having an `init()` routine in `main()` to set up everything you need and then a `while(1)` loop that is waiting for real-time events to occur.

So let's start at the top of the sketch you just opened:

- Near the top of the file, you will see two arrays – one for Transmit (TX) and one for Receive (RX):

```
// Data to write to radio TX FIFO (60 bytes MAX.)
unsigned char txData[6] = { 0x30, 'A', 'i', 'r', '!', '\0' };

// Data to read from radio RX FIFO (60 bytes MAX.)
unsigned char rxData[6] = { '\0', '\0', '\0', '\0', '\0', '\0' };
```

So the transmitter will be sending a set of characters like “Air!0” and the receiver will receive this data as well – initially, the receiver data is set to all zeroes.

- The following two functions – `printTxData()` and `printRxData()` will be called during the loop to print (to a terminal) what is going on in the code.
- The `setup()` routine first begins the radio transmission and sets the channel to talk on:

```
// *** IoT STUDENTS - PICK A CHANNEL BETWEEN 1 and 12 to REPLACE "X" below ***
Radio.begin(0x01, CHANNEL_X, POWER_MAX);
```

With multiple users in the workshop room, Channels 1-12 have been enumerated, so you MUST use a CHANNEL # that no one else picked. At this point, hopefully your instructor has assigned Channel numbers to everyone doing this lab in order to avoid collision with other users.

- ▶ WRITE DOWN YOUR CHANNEL NUMBER HERE: _____
- ▶ Edit the code to reflect your channel number (e.g. CHANNEL_1 ro CHANNEL_12).
- ▶ Save your sketch (*File* → *Save*).
- Then, the red LED on the board is set “off” (LOW) even though the comment says “ON”. Ah, the joys of comments.
- In the loop() code the following occurs:
 - Transmit `Txdata()` and then print to the terminal what was transmitted
 - Cycle through numbers 0-9 ASCII
 - When receiver receives data, turn on the red LED, else keep it off. When you run the code, the red LED will flash at a very fast rate (assuming your neighbor’s board is transmitting to you on the same channel). If you are NOT receiving any data, you will not see the LED flash at all.

Build Load and Run

7. Build the code.

Now, you need to build the code and check for errors.

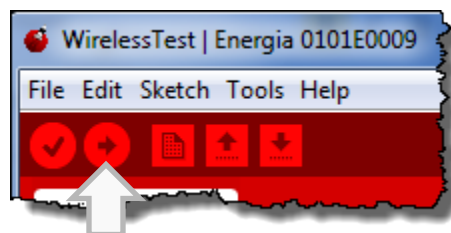
- ▶ Click on the checkmark (✓) as shown below:



8. Upload the executable to your LaunchPad.

With a clean build, you can now upload your code to the LaunchPad.

- ▶ Select the *Upload* button (right-hand arrow):



This will compile AND upload the executable to your LaunchPad. Each user in the “pair” of LaunchPads should now have the same code loaded onto their target. It is now time to test the receive/transmit of the LPRF device.

9. Start the LPRF receive/transmit sequence.

- ▶ Unplug your board from your laptop and plug it back in.

Each user should now unplug your board and plug it back into the USB connection on your laptop. This ensures that the program you uploaded is truly programmed into the flash memory of the device.

If your red LED is flashing, you are receiving data. If the LED is NOT flashing, there is a problem. You can try to debug it or ask your instructor for help.

Ok, so the LED is flashing – so what? In the next step, you'll open the built-in terminal to see the Rx/Tx messages being sent.

Energia includes a built-in terminal program. Yes, you can use any terminal emulator you think is great, but this lab just uses the built-in terminal to Energia.

First, HOW is this data being transmitted and received to a terminal program?

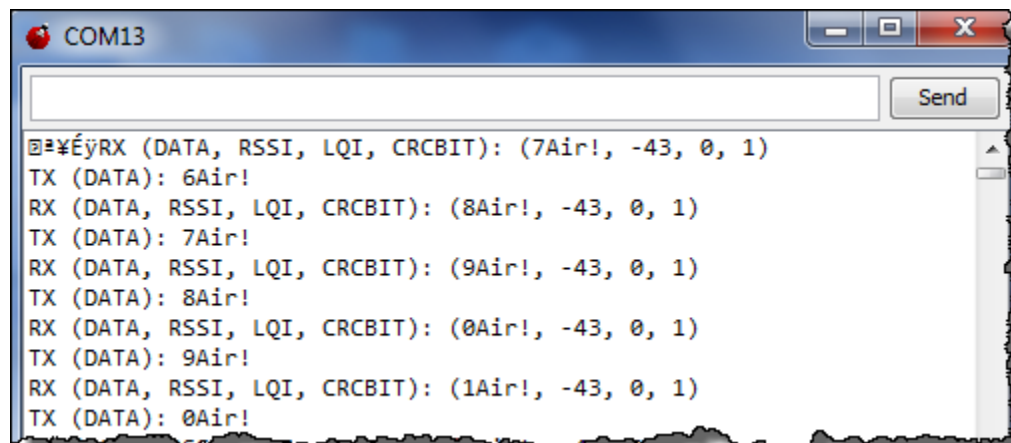
► Peruse the code to find the following code listing.

The answer is – it is via the RSSI data as part of the Tx and Rx data structure:

```
void printRxData()
{
    /**
     * The following illustrates various information that can be obtained when
     * receiving a message. This includes: the received data and associated
     * status information (RSSI, LQI, and CRC_OK bit).
     */
    Serial.print("RX (DATA, RSSI, LQI, CRCBIT): ");
    Serial.print("(");
    Serial.print((char*)rxData);
    Serial.print(", ");
    Serial.print(Radio.getRssi());
    Serial.print(", ");
    Serial.print(Radio.getLqi());
    Serial.print(", ");
    Serial.print(Radio.getCrcBit());
    Serial.println(")");
}
```

► Open the terminal by selecting Tools → Serial Monitor.

Each user should see a display that looks similar to this:



If both users see a similar display and the red LED is blinking, then communication is happening. Pretty cool. Now let's move on to the next example....

Example 2 – Blink the Other User's LED !

Using this same code base, you can actually blink the other person's LED. We took the base code for Wireless Test and added the ability to blink the LEDs of the other user's LaunchPad. Instead of sending the entire array (Tx/Rx) of data, we'll send a variable that says "LED on" or "LED off".

We also added some code to use the pushbuttons on the LaunchPad. When user1 pushes down pushbutton 2, the "sender's" blue LED lights up (indicating that the sender's button has been depressed) and then the receiver's RED LED lights up indicating that the "message" has been received.

10. Open a new sketch.

- Select *File* → *Open* and browse to the sketch for LED_GO_LIGHTLY:

```
c:\MCUDay\labs\Tiva\lab7\LED_GO_LIGHTLY\LED_GO_LIGHTLY.ino
```

11. Analyze the code changes.

In this sketch, we simply edited the data packets to only include a binary value that represents whether an LED should be turned on or off.

- Scroll down to the `printTxData()` and `printRxData()` functions. Notice that these functions only include a two-element array – one of which is the status of pushbutton #2.

- Change `CHANNEL_X` in the code below to use the same channel number as before:

```
// *** IoT STUDENTS - PICK A CHANNEL BETWEEN 1 and 12 to REPLACE "X" below ***
Radio.begin(0x01, CHANNEL_X, POWER_MAX);
```

- Save your sketch.

- Further down the code, you can see that both LEDs (red and blue) and pushbutton #2 are set up for use. If the "sender" (transmitter) depresses pushbutton #2 (bottom-right of the board), then the blue LED will light up on the sender's board. If this occurs, and the receiver is ready to receive, the receiver's RED LED will light up.

12. Upload the new sketch to your LaunchPad.

- Click the *Upload* button in Energia to compile and upload the new program to your LaunchPad. If there are any errors, try to fix them quickly. If the problems continue, ask your instructor.
- Unplug and plug back in your board – again, making sure the code is resident in flash and the program resets itself.

13. Run the new program and blink your partner's LED.

No LEDs should be flashing at this point.

- One user should depress pushbutton #2 (bottom right-hand button on the Tiva-C LP). The SENDER's blue LED should light up. At the same time, the RECEIVER's RED LED should light up.

- Then, switch the order. Have the other person push their button and see if the communication is two-way.

Then, if all is well, ► both users should push their buttons to see if both the blue and red LEDs light up (you should see two of the three colors of the tri-color LED light up).

If this is all working properly, move on to the next part...

Example 3 – Change LED colors

In this example, all you are going to do is change the color of the LED that is set on the receiver's board. Only ONE of the users in the pair needs to make this code change – that way, it is a bit more interesting.

14. Choose a user to make the code change.

Between the two people in the pair, ► pick one user to make the following changes to the code.

15. Modify the code to use the GREEN LED.

One of the users needs to modify the RED LED setup and use code to use the GREEN LED.

► In `setup()`, find the code that sets up the RED LEDs:

```
pinMode(RED_LED, OUTPUT);  
digitalWrite(RED_LED, LOW);
```

16. Change the `pinMode()` and `digitalWrite()` calls to use "GREEN_LED".

► In the `loop()` code further down, change the following code to use "GREEN_LED":

```
if (rxData[0] == 1) {  
    digitalWrite(RED_LED, LOW);  
}  
else {  
    digitalWrite(RED_LED, HIGH);  
};
```

17. Upload the new code to the chosen user's board.

► Click on the *Upload* button in Energia to load the program into flash.

► Push the buttons on both boards to make sure the user's board who change to "green" receives the communication and their green LED lights up.

If all is well, you're done with this lab.

That's it, You're Done.

- Close Energia.
- Remove your board from the USB port.
- Gently unhook the Anaren AIR module from the LaunchPad.



You're finished with this lab. Raise your hand and let your instructor know you are finished. If you have time, help a neighbor get through their lab – giving is a good thing... ☺