

Using USB Host Mode on the EK-TM4C123GXL LaunchPad

Charles Tsai

ABSTRACT

The EK-TM4C123GXL LaunchPad™ Evaluation Kit is a low-cost evaluation platform for Arm® Cortex®-M4F based microcontrollers from Texas Instruments. The design of the TM4C123G LaunchPad highlights the TM4C123GH6PM microcontroller (MCU) with a USB 2.0 device interface and features including hibernation, user buttons and an RGB LED for custom applications. This application report demonstrates how to turn the LaunchPad into both a USB host and USB device capable evaluation kit for USB application developments. Along with the application report are four USB host examples that interface with a USB mouse, keyboard or USB memory stick.

Project collateral and source code discussed in this application can be downloaded from the following URL: <http://www.ti.com/lit/zip/spna243>.

Contents

| | | |
|---|--|----|
| 1 | EK-TM4C123GXL Board Overview | 2 |
| 2 | Board Modification to Make EK-TM4C123GXL USB Host/OTG Capable..... | 3 |
| 3 | Download and Import the USB Host Examples..... | 6 |
| 4 | Run the usb_host_mouse Example..... | 10 |
| 5 | Run the usb_host_keyboard Example..... | 12 |
| 6 | Run the usb_stick_update and usb_stick_demo Examples | 12 |
| 7 | References | 16 |

List of Figures

| | | |
|----|--|----|
| 1 | Arm Cortex-M4F Based MCU TM4C123G LaunchPad Evaluation Kit | 2 |
| 2 | EK-TM4C123GXL LaunchPad R25 and R29 Resistors Location | 3 |
| 3 | EK-TM4C123GXL LaunchPad Power Selection Schematic | 4 |
| 4 | EK-TM4C123GXL LaunchPad H18 and H19 Locations | 5 |
| 5 | Import CCS Projects Step 1 | 6 |
| 6 | Import CCS Projects Step 2 | 7 |
| 7 | Import CCS Projects Step 3 | 8 |
| 8 | Import CCS Projects Step 4 | 9 |
| 9 | Debug CCS Projects | 10 |
| 10 | Serial Terminal Settings | 11 |
| 11 | Mouse Movement Displayed on Terminal Window | 11 |
| 12 | Keyboard Displayed on Terminal Window | 12 |
| 13 | Adding FIRMWARE.BIN to the Memory Stick..... | 13 |
| 14 | FIRMWARE.BIN Programmed at 0x4800..... | 14 |
| 15 | usb_stick_demo Outputs | 15 |
| 16 | usb_stick_update Waiting for the Memory Stick Insertion | 15 |

Trademarks

LaunchPad, Code Composer Studio are trademarks of Texas Instruments.
 Arm, Cortex are registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere.
 All other trademarks are the property of their respective owners.

1 EK-TM4C123GXL Board Overview

Figure 1 shows a photo of the EK-TM4C123GXL LaunchPad board. As can be seen, there are two USB connectors on the board. On the top side is the USB connector for the in-circuit debug interface (ICDI). On the left side is the second USB Micro-A/B connector. Although this micro-A/B connector is capable for USB device, host and on-the-go (OTG) connectivity, the board as is can only support USB device connectivity. In other words, the EK-TM4C123GXL is only designed to function as a USB device without hardware modification. The lack of USBID and USBVBUS signals at the connector are preventing the board from functioning as a USB host or USB OTG. The USBVBUS signal is required during the session request protocol and it allows the USB PHY to both sense the voltage level of VBUS and pull up VBUS momentarily during the VBUS pulsing. The VBUSID is used by the USB PHY to determine the initial state of the USB controller (pulled down is the A side of the cable and pulled up is the B side).

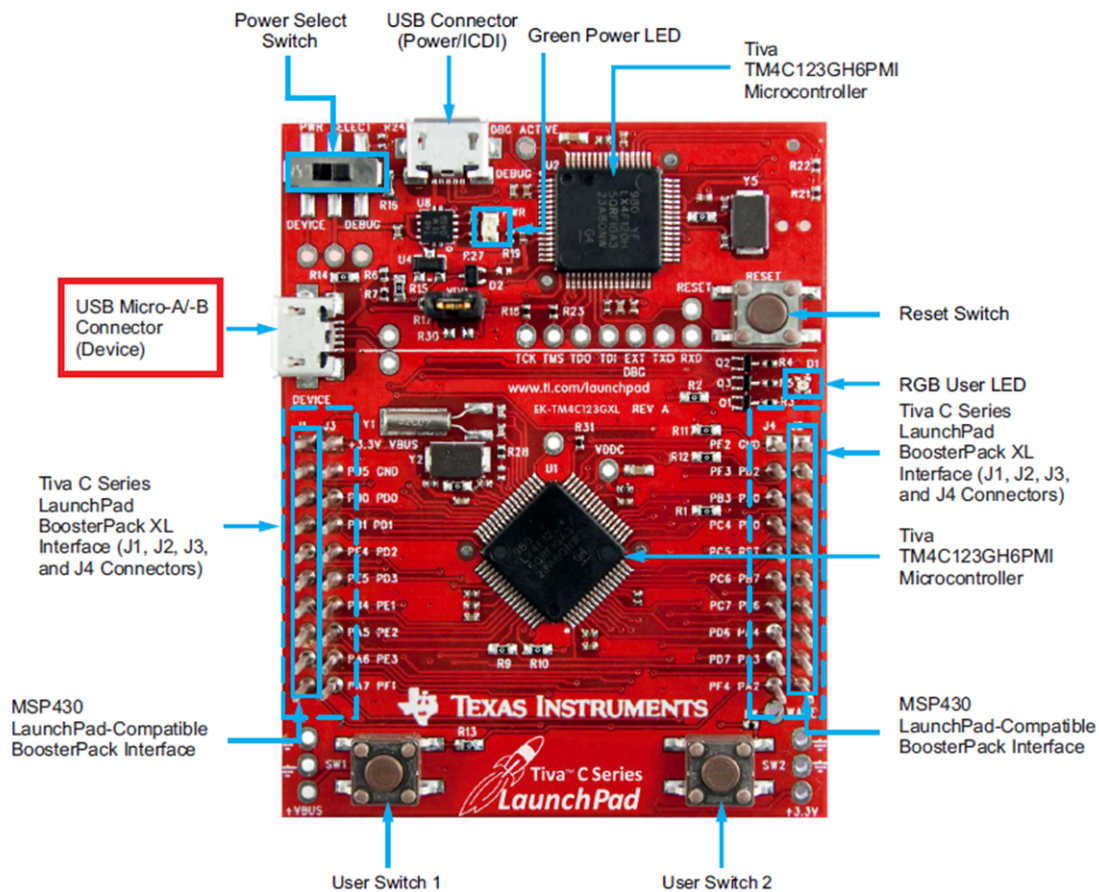


Figure 1. Arm Cortex-M4F Based MCU TM4C123G LaunchPad Evaluation Kit

2 Board Modification to Make EK-TM4C123GXL USB Host/OTG Capable

The TM4C123GH6PM MCU is capable of USB embedded host and OTG functions. The OTG functionality can be enabled by populating R25 and R29 with 0- Ω resistors. These resistors connect the USB ID and USB VBUS signals on the USB device connector to the PB0 and PB1 pins on the MCU. When these resistors are populated, the PB0 and PB1 pins must remain in their respective USB pin mode configurations to prevent device damage. The PB0 and PB1 pins are also present on the J1 BootsterPack header. Therefore, if R25 or R29 are populated, care must be taken not to conflict these signals with BoosterPack signals. For R25 and R29 locations on the board, see [Figure 2](#).

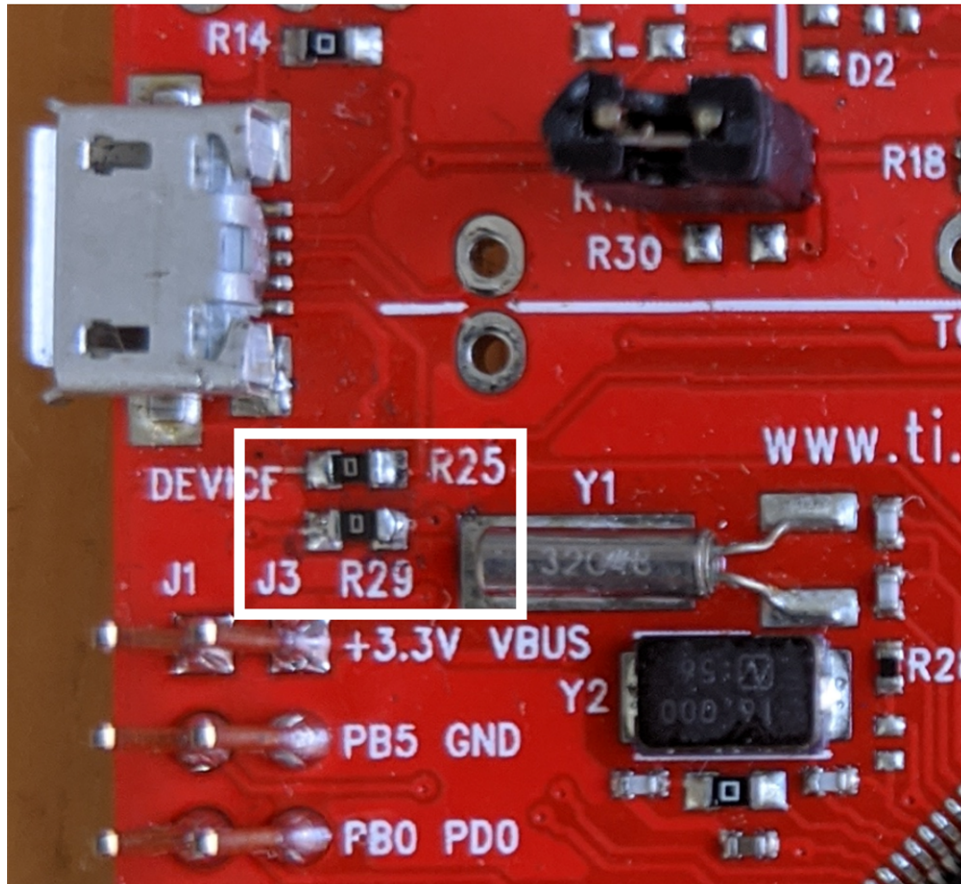


Figure 2. EK-TM4C123GXL LaunchPad R25 and R29 Resistors Location

The Tiva C Series LaunchPad can be powered from one of the two power sources:

- On-board ICDI USB cable (Debug, Default)
- USB device cable (Device)

The POWER SELECT switch (SW3) is used to select one of the two power sources. When the LaunchPad functions as a USB device, it can receive power from the external USB host by toggling the SW3 switch.

For the power selection schematic, see [Figure 3](#).

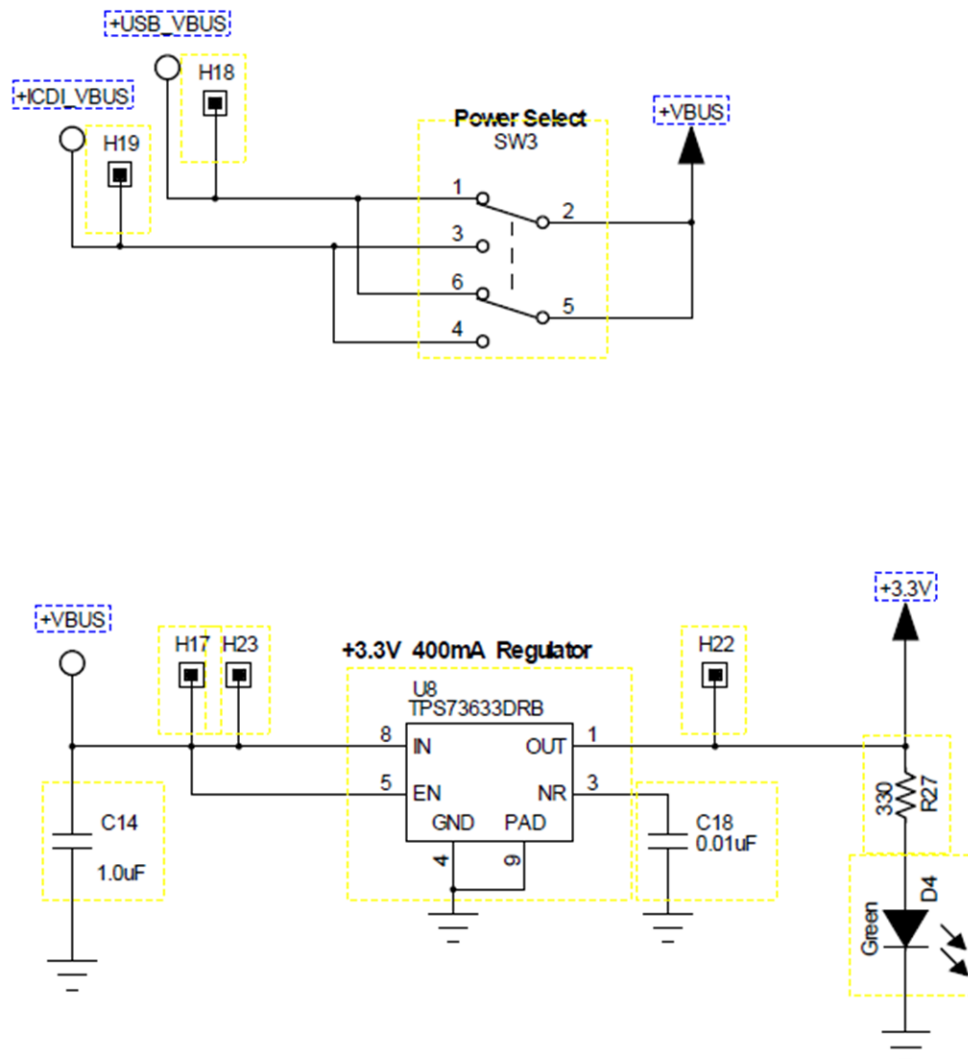


Figure 3. EK-TM4C123GXL LaunchPad Power Selection Schematic

However, in order for the LaunchPad to function as a USB host when the USB device requires bus power, it needs to supply a 5 V power source to the USB device. As a result of the current Power Select switch design, the USB VBUS pin on the Device USB connector (+USB_VBUS) is not connected to the USB VBUS input on the Debug USB connector (+ICDI_VBUS). Therefore, the Device USB connector cannot provide 5 V to any USB device that requires bus power without a hardware modification.

The required hardware modification needs to route the USB VBUS from the Debug USB connector to the Device USB connector. This can be done by connecting the H18 and H19 test points shown in [Figure 3](#) together so the 5 V power from the Debug connector (+ICDI_VBUS) is routed to the Device connector (+USB_VBUS). H18 and H19 are unmarked on the LaunchPad.

Figure 4 shows shows these tied together using a jumper wire.

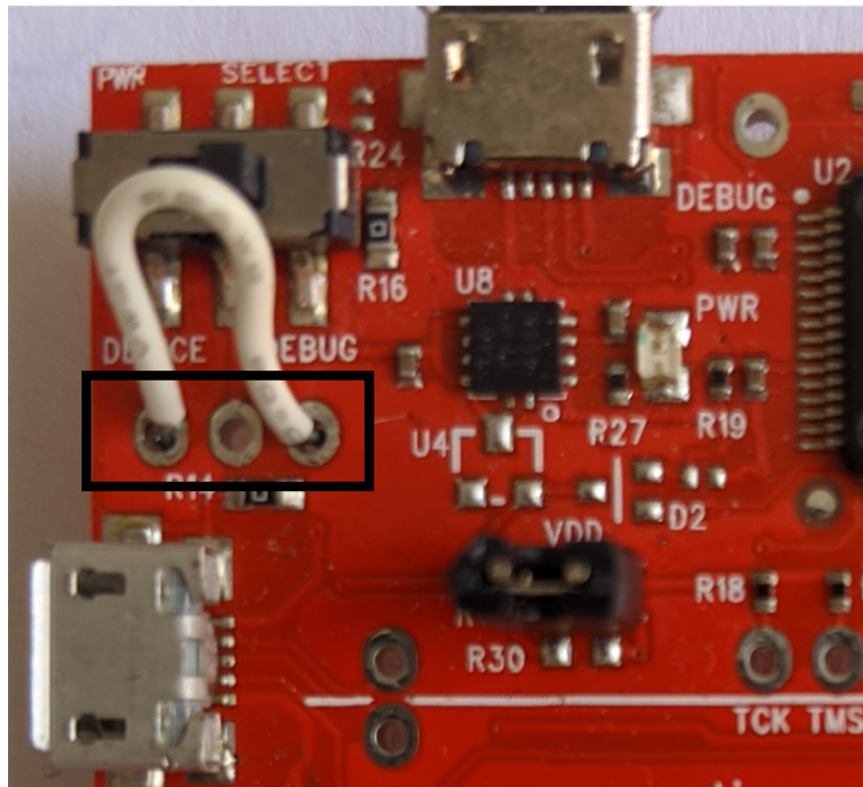


Figure 4. EK-TM4C123GXL LaunchPad H18 and H19 Locations

With the two board modifications, the EK-TM4C123GXL is now able to function as a USB device, USB host or OTG.

3 Download and Import the USB Host Examples

There are four Code Composer Studio™ (CCS) project examples attached as collateral to this application report. The examples can be downloaded from the following URL: <http://www.ti.com/lit/zip/spna243>. You can unzip the project or keep it in zip format. Both formats can be imported into CCS.

1. To import the project into CCS, first select "File" → "Import".

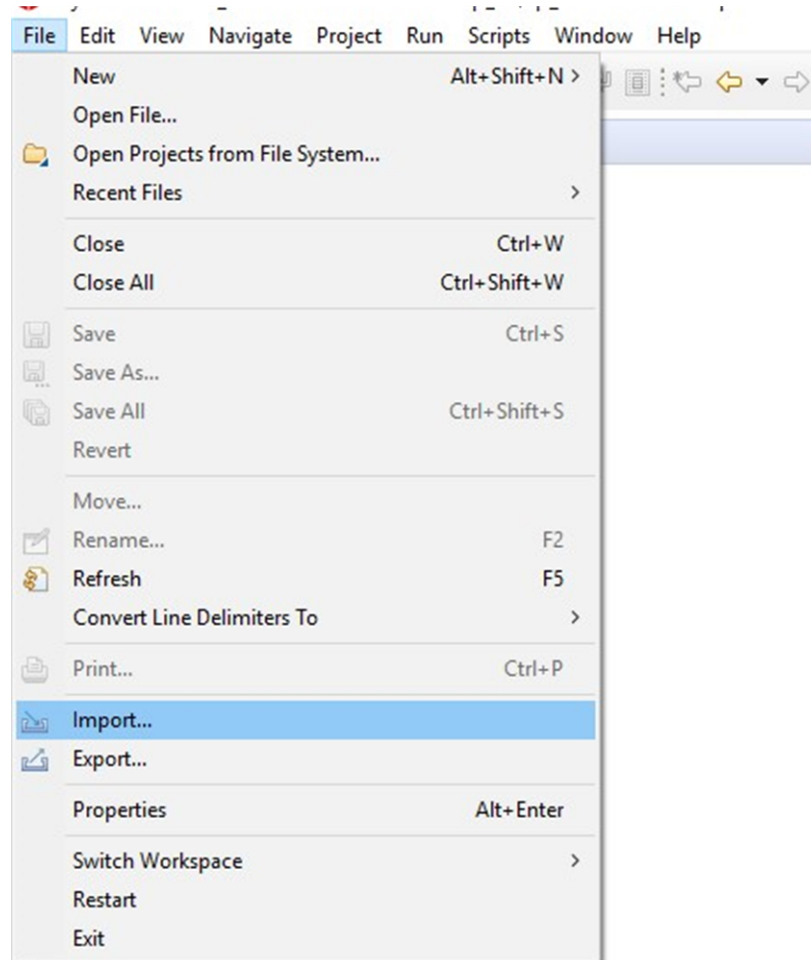


Figure 5. Import CCS Projects Step 1

2. Select "CCS Projects" to import the example and then click "Next".

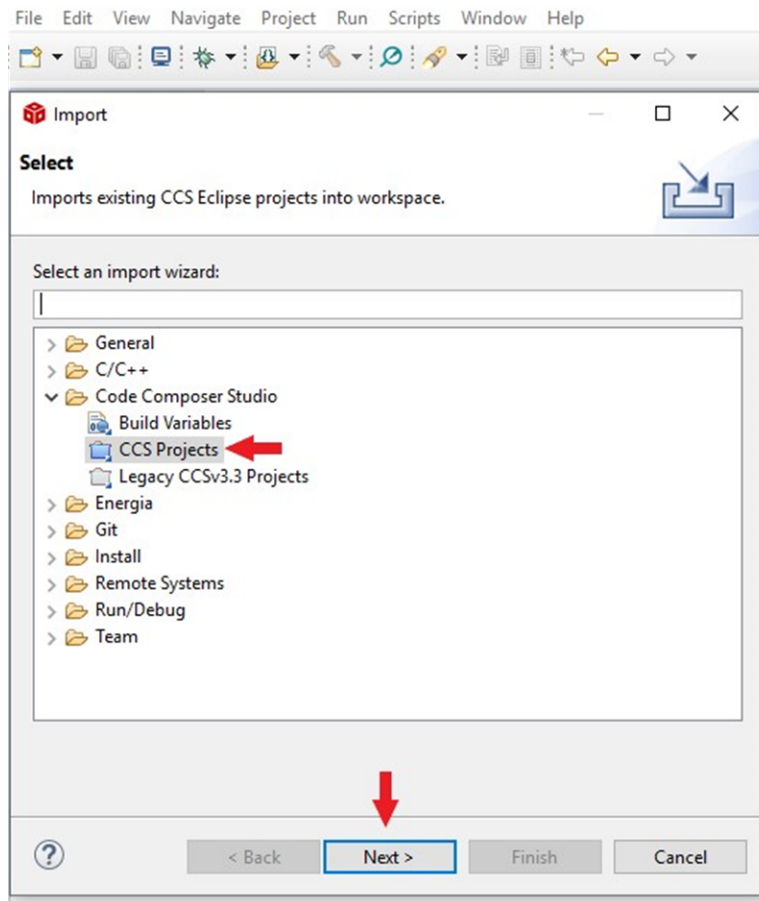


Figure 6. Import CCS Projects Step 2

3. Provide the path to either the unzipped project by selecting the first radio button or import from the zip file directly by selecting the second radio button. Click the "Copy projects into workspace".

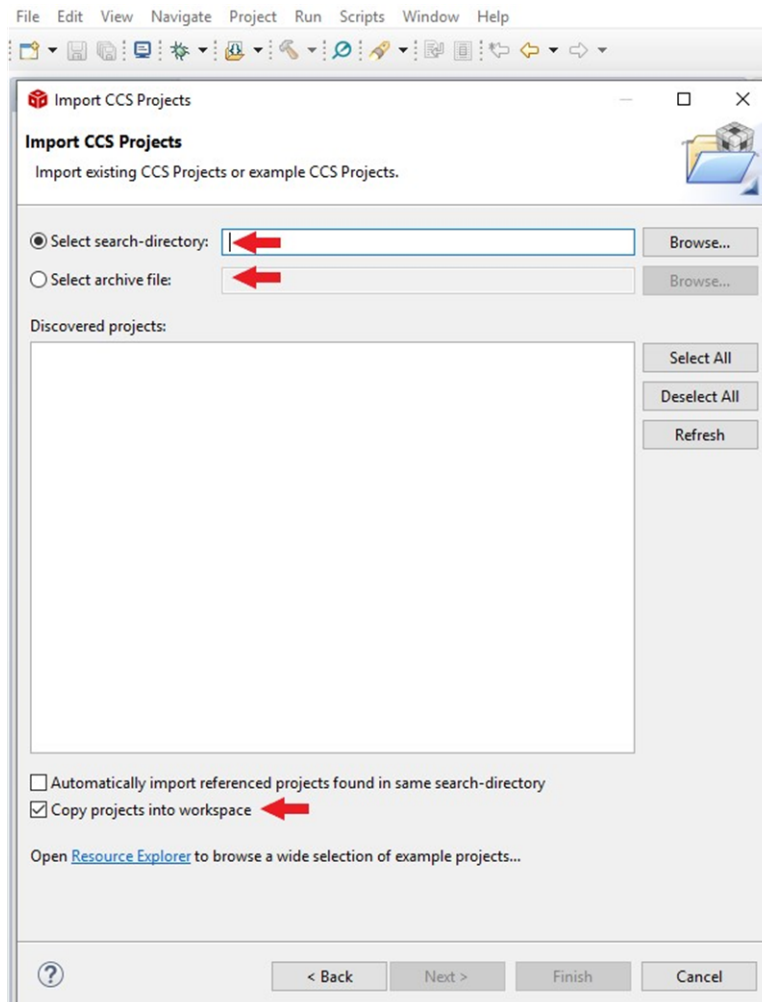


Figure 7. Import CCS Projects Step 3

4. After the project path is provided, four discovered projects will show up. First, click the "Select All" button and then click the "Finish" button to complete the import.

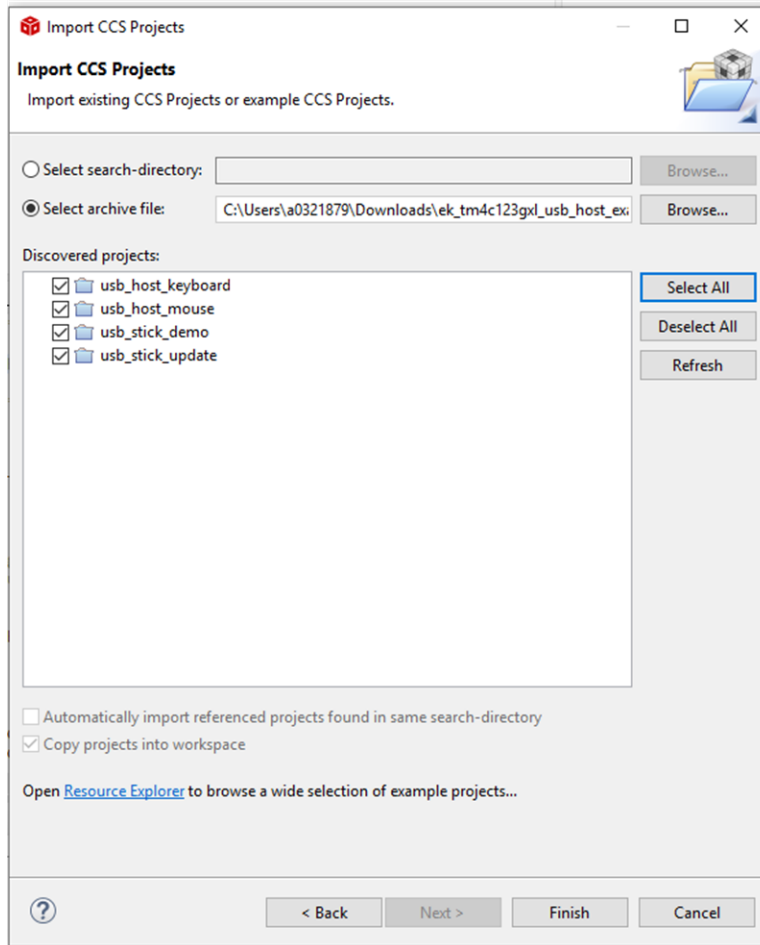


Figure 8. Import CCS Projects Step 4

4 Run the usb_host_mouse Example

This application demonstrates the handling of a USB mouse attached to the evaluation kit. Once attached, the position of the mouse pointer and the state of the mouse buttons are output to the display.

4.1 Build and Flash the Program

Select the usb_host_mouse as the active project. With the LaunchPad's ICD1 USB port connected to the PC via the USB cable, build the project and load the program by clicking the debug icon.

If you are new to CCS, click on the [CCS User's Guide](#).

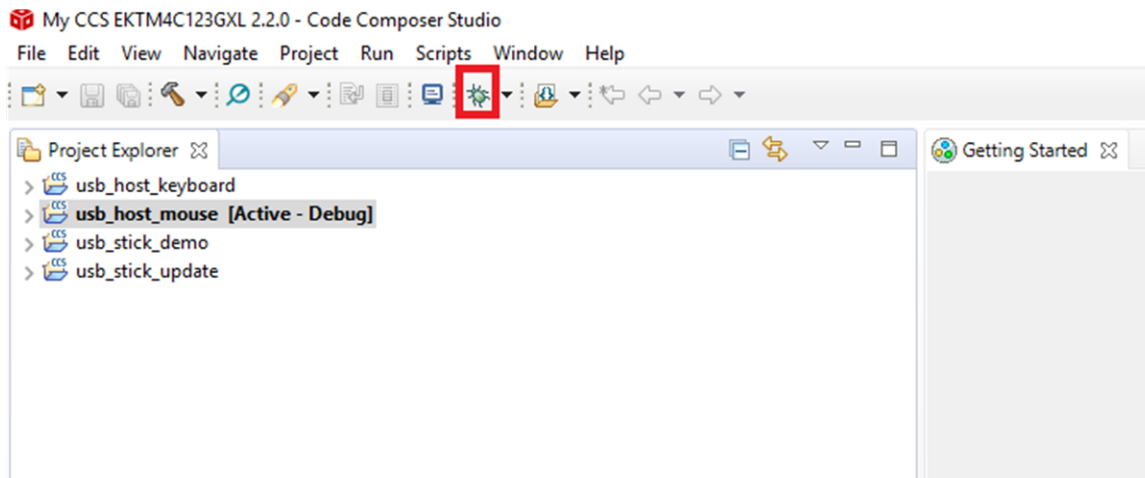


Figure 9. Debug CCS Projects

4.2 Configure the Terminal Window

The example displays the status and coordinates of the mouse's movements on the terminal window. You can use any terminal emulator of your choice. The terminal emulator should be configured for 115200 Baud and 8-N-1 as shown in [Figure 10](#). For this document, the terminal emulator built into CCS will be used. You can launch the CCS terminal emulator by going to "View" → "Terminal".

Then select "Open a Terminal", and choose terminal type "Serial Terminal". Identify the COM Port for the LaunchPad "Stellaris Virtual Serial Port" and input that for the "Serial Port" entry and use the drop down menus to finish the configuration. This process will connect the terminal to the UART output that comes from the LaunchPad debug.

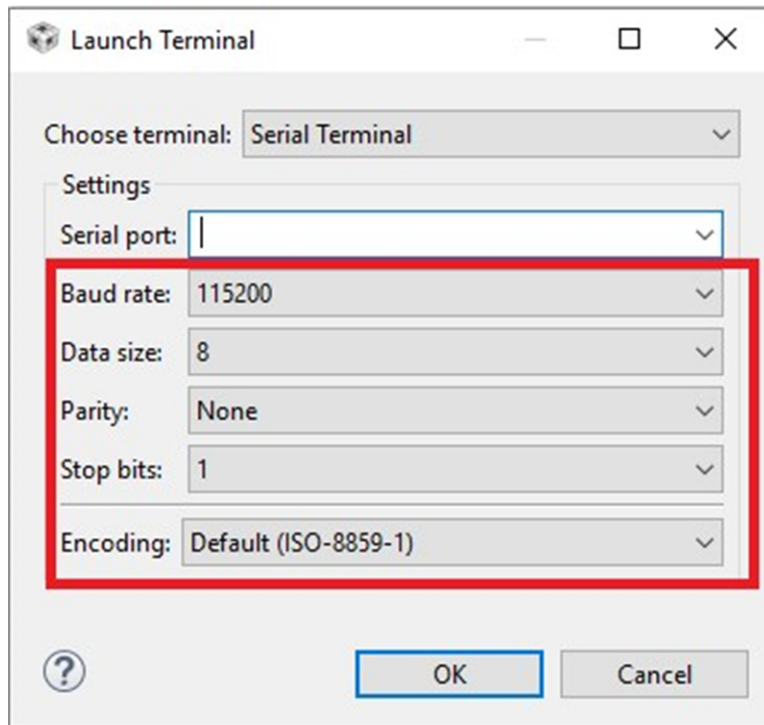


Figure 10. Serial Terminal Settings

4.3 Run the usb_host_mouse Example

Plug in a mouse to the LaunchPad's USB device connector on the left hand side of the board and run the `usb_host_mouse` program. A Micro-USB to USB 2.0 Type A adapter likely will be needed if the mouse comes with the Type A connector. Experiment with the example by moving and clicking the buttons on the mouse. With the terminal window opened, you should see the coordinates of the mouse changing while moving the mouse. In [Figure 11](#), the status of the mouse buttons are shown in the following order: Left, Right and Scroll Wheel Click.

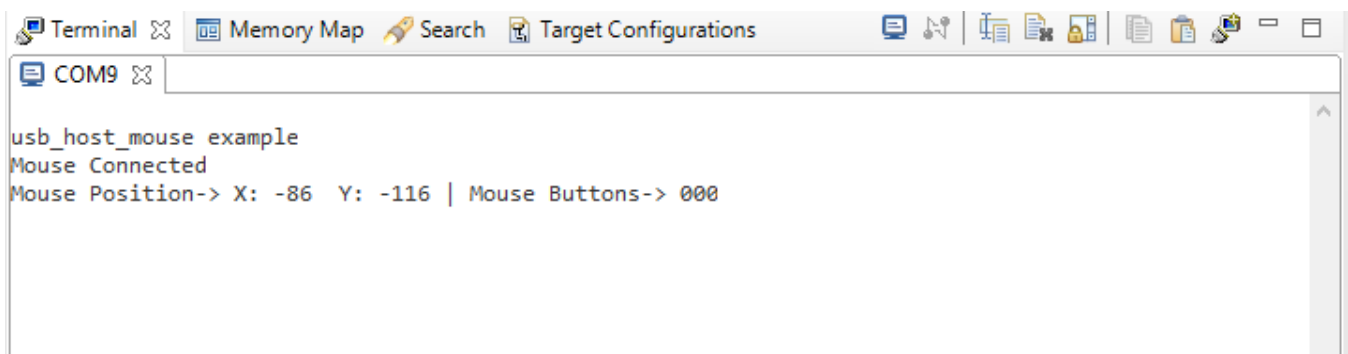


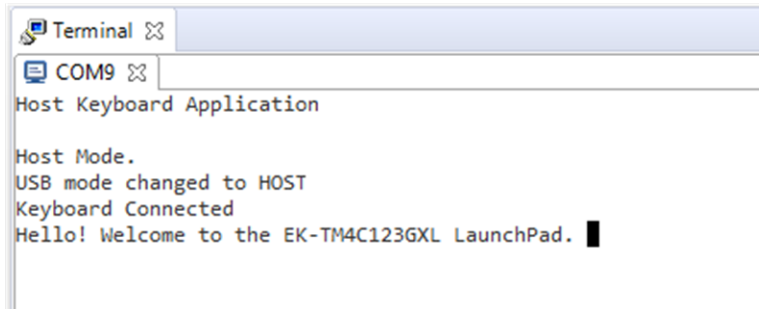
Figure 11. Mouse Movement Displayed on Terminal Window

5 Run the `usb_host_keyboard` Example

This example application demonstrates how to support a USB Keyboard Output attached to the LaunchPad. The display will show if a keyboard is currently connected. Pressing any key on the keyboard will cause it to be printed on the screen and to be sent out the UART at 115200 Baud and 8-N-1. Any keyboard that supports the USB HID BIOS protocol should work with this demo application.

Follow [Section 4.1](#) and [Section 4.2](#) to build and load the `usb_host_keyboard` and connect to a terminal window. Plug in a USB keyboard and run the example. Experiment with the example by typing on the keyboard in upper case and lower cases. With the terminal window opened, you should see the keys typed on the keyboard displayed on the terminal window.

[Figure 12](#) shows the words “Hello! Welcome to the EK-TM4C123GXL LaunchPad.” typed on the keyboard and displayed on the terminal window.



```

Terminal
COM9
Host Keyboard Application
Host Mode.
USB mode changed to HOST
Keyboard Connected
Hello! Welcome to the EK-TM4C123GXL LaunchPad. █
    
```

Figure 12. Keyboard Displayed on Terminal Window

6 Run the `usb_stick_update` and `usb_stick_demo` Examples

6.1 *USB Memory Stick Updater (`usb_stick_update`) Overview*

This example application behaves the same way as a bootloader. It resides at the beginning of flash, and will read a binary file from a USB memory stick and program it into another location in flash. Once the user application has been programmed into flash, this program will always start the user application until requested to load a new application.

When this application (`usb_stick_update`) starts, if there is a user application (`usb_stick_demo`) already in flash (at `APP_START_ADDRESS`), then it will just run the user application. It will attempt to load a new application from a USB memory stick under the following conditions:

- No user application is present at `APP_START_ADDRESS`
- The user application has requested an update by transferring control to the updater

When this application is attempting to perform an update, it will wait indefinitely for a USB memory stick to be plugged in. Once a USB memory stick is found, it will search the root directory for a specific file name, which is `FIRMWARE.BIN` by default. This file must be a binary image of the program you want to load (the `.bin` file), linked to run from the correct address, at `APP_START_ADDRESS`.

NOTE: This example only supports the 8.3 filename naming scheme (also called a short filename or SFN). The name of the file must be 11 characters total, 8 for the base name and 3 for the extension. If the actual file name has fewer characters then it must be padded with spaces. By default, the file name is defined in the directive in the `usb_stick_update.c` as follows:
`#define USB_UPDATE_FILENAME "FIRMWAREBIN"`

The USB memory stick must be formatted as a FAT16 or FAT32 file system (the normal case), and the binary file must be located in the root directory. Other files can exist on the memory stick but they will be ignored.

NOTE: The `APP_START_ADDRESS` for the user application is currently defined as `0x4800` in this example. However, this can be changed to any address that is aligned to the flash page size which is `0x400` for the TM4C123 MCU.

6.2 USB Stick Update Demo (`usb_stick_demo`) Overview

This is an example to demonstrate the use of the flash-based USB stick update program. This example is meant to be loaded into Flash memory from a USB memory stick using the USB stick update program (`usb_stick_update`) running on the microcontroller.

After this program is built, the binary file (`usb_stick_demo.bin`) should be renamed to the filename expected by `usb_stick_update` (“`FIRMWARE.BIN`” by default) and copied to the root directory of a USB memory stick. Then, when the memory stick is plugged into the LaunchPad board that is running the `usb_stick_update` program, this example program will be loaded into flash and run on the microcontroller.

This program simply displays a message on the terminal window and prompts the user to press the SW1 switch. Once the switch is pressed, control is passed back to the `usb_stick_update` program which is still in flash, and it will attempt to load another program from the memory stick. This shows how a user program can force a new firmware update from the memory stick.

6.3 Add the `FIRMWARE.BIN` File in the Memory Stick

The `usb_stick_demo.bin` file is pre-built and can be found in the `$PROJECT_ROOT/Debug` folder. If the `.bin` file is not found, you can follow [Section 4.1](#) and [Section 4.2](#) to rebuild the project.

Copy the `usb_stick_demo.bin` to your USB memory stick and rename the `usb_stick_demo.bin` to `FIRMWARE.BIN`, as shown in [Figure 13](#).

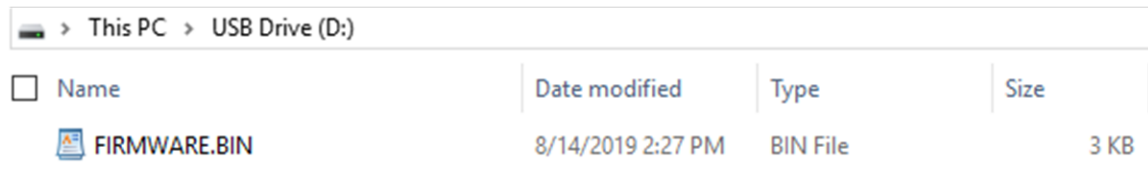


Figure 13. Adding `FIRMWARE.BIN` to the Memory Stick

6.4 Run the `usb_stick_update` Program

Follow [Section 4.1](#) and [Section 4.2](#) to build and flash the `usb_stick_update` program. Carefully view the memory browser window with addresses starting at `0x4800`. This is the starting location at which the user application (`usb_stick_demo.bin`) will be loaded. You should expect the content starting at this location to be in the erased state, which is all `0xFFFFFFFF`.

Plug in the memory stick to the LaunchPad's USB device connector and run the program. The `usb_stick_update` program will detect the presence of the inserted memory stick and find the `FIRMWARE.BIN` file in the root directory. It will program `FIRMWARE.BIN` into the flash starting at address `0x4800`. Once it is done, you can view the memory browser window to see the flash updated with the `usb_stick_demo` program image, as shown in [Figure 14](#).

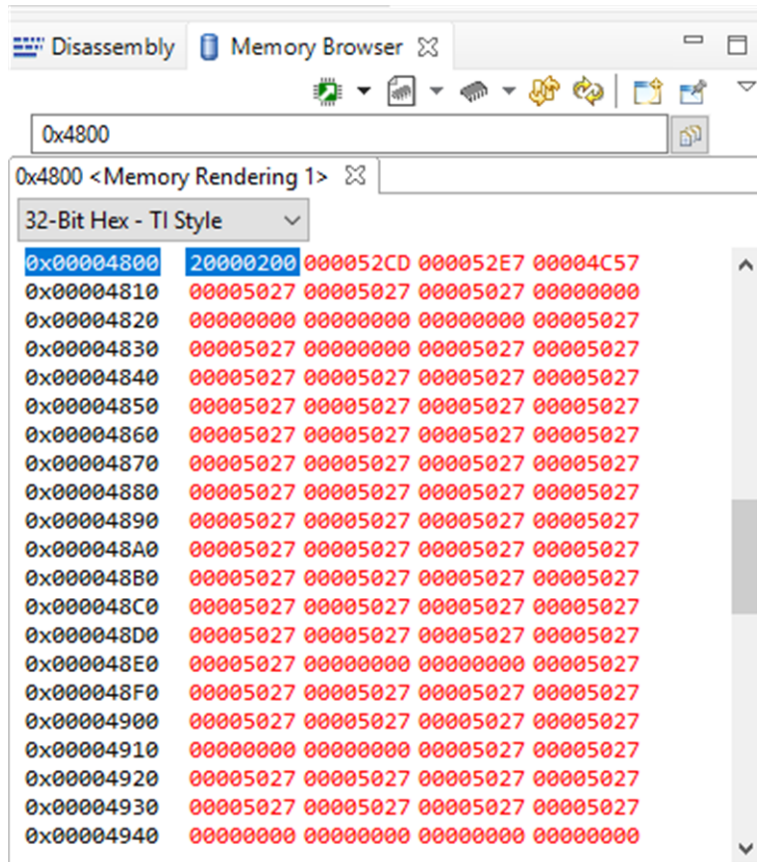


Figure 14. `FIRMWARE.BIN` Programmed at `0x4800`

6.5 Run the `usb_stick_demo` Program

After the `usb_stick_demo` program is loaded into the flash, the processor will skip the `usb_stick_update` (the bootloader application) and will directly jump to the user application (the `usb_stick_demo`).

With the terminal window open, you will see messages as shown in [Figure 15](#). The `usb_stick_demo` is waiting for the user to press the SW1 switch to transfer the control back to the `usb_stick_update` routine to force a new update of the user application.

Before pressing the SW1 switch, carefully observe the LED light on the LaunchPad. The LED will start with one particular color (red) and sequence to blue then green after each update of the user application. This is a confirmation that a new update happened even if the same `usb_stick_demo` is loaded again.

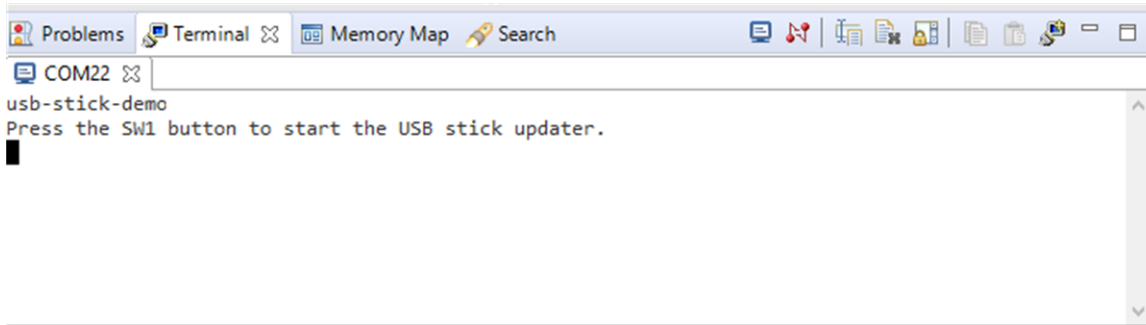


Figure 15. `usb_stick_demo` Outputs

Next, press the SW1 switch. After the SW1 switch is pressed, a message is displayed on the terminal window indicating it is now waiting for the USB Memory Stick to be inserted and control is transferred back to the `usb_stick_update` routine.



Figure 16. `usb_stick_update` Waiting for the Memory Stick Insertion

After the memory stick is inserted, a new image of `FIRMWARE.BIN` (`usb_stick_demo` or your own custom user application) is loaded to the flash at address `0x4800`. Run the updated user application. Carefully observe the LED light again if you are loading the same `usb_stick_demo` program image. You should see a different LED light color compared to the last run. Repeat [Section 6.5](#) and you will see the LED light color sequencing between red, blue and green.

7 References

- [TivaWare™ for C Series \(Complete\)](#)
- [Code Composer Studio User's Guide](#)
- Texas Instruments: [TivaWare™ Peripheral Driver Library for C Series User's Guide](#)
- Texas Instruments: [TivaWare™ USB Library for C Series User's Guide](#)
- Texas Instruments: [Tiva™ C Series TM4C123G LaunchPad Evaluation Board User's Guide](#)
- Texas Instruments: [Tiva™ TM4C123GH6PM Microcontroller Data Sheet](#)

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES "AS IS" AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI's products are provided subject to [TI's Terms of Sale](#) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI's provision of these resources does not expand or otherwise alter TI's applicable warranties or warranty disclaimers for TI products.

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

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2022, Texas Instruments Incorporated