

MCU SDK

Getting Started Guide



Literature Number: SPRUHD3C
July 2012

Preface	3
1 About the MCU SDK	4
1.1 What is the MCU SDK?	4
1.2 What are the MCU SDK Components?	5
1.3 What Drivers Does the MCU SDK Include?	6
1.4 Hardware Resources Consumed by MCU SDK Components	6
1.4.1 TMDXDOCKH52C1 Resources Used	6
1.4.2 DK-LM3S9D96 Resources Used	7
1.4.3 EKS_LM4F232 Resources Used	7
1.5 For More Information	8
2 Installing the MCU SDK	10
2.1 System Requirements	11
2.2 Installing Required Components	11
2.2.1 Installing Code Composer Studio (CCS)	11
2.2.2 Installing System Analyzer as a Software Update	12
2.3 Installing the MCU SDK	13
2.3.1 Installing on Windows	13
2.3.2 Installing on Linux	13
3 Examples for the MCU SDK	14
3.1 Creating an Example Project Using the TI Resource Explorer	15
3.2 Example Overview	16
3.3 Example Settings	17
3.3.1 TMDXDOCKH52C1 Development Board	17
3.3.2 DK_LM3S9D96 Board	18
3.3.3 LM4F120H5QR Board	18
3.4 Installing USB Drivers for the USB Device Examples	19
3.5 Using Examples for Individual Components	20
4 Configuring the MCU SDK	21
4.1 Starting the Configuration Tool	22
4.2 Configuring MCU SDK	23
4.3 Configuring Individual Sub-Components	23
Index	24

Read This First

About This Manual

This manual describes the MCU Software Development Kit (SDK). The version number as of the publication of this manual is v1.0.

Notational Conventions

This document uses the following conventions:

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

Here is a sample program listing:

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

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

Trademarks

Registered trademarks of Texas Instruments include Stellaris and StellarisWare. Trademarks of Texas Instruments include: the Texas Instruments logo, Texas Instruments, TI, TI.COM, C2000, C5000, C6000, Code Composer, Code Composer Studio, Concerto, controlSUITE, DSP/BIOS, SPOX, TMS320, TMS320C5000, TMS320C6000 and TMS320C2000.

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

Windows is a registered trademark of Microsoft Corporation.

Linux is a registered trademark of Linus Torvalds.

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

July 12, 2012

About the MCU SDK

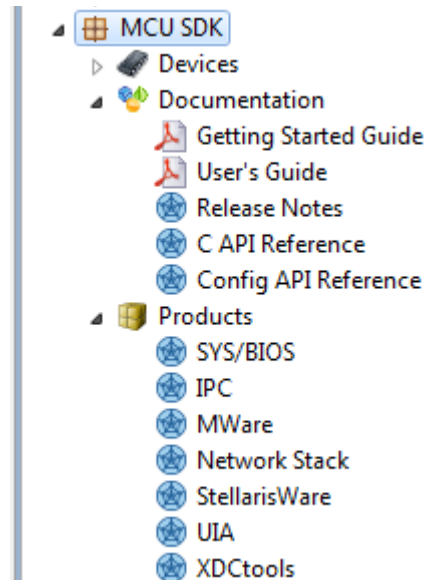
This chapter provides an overview of the MCU SDK (Micro-controller Software Development Kit).

Topic	Page
1.1 What is the MCU SDK?	4
1.2 What are the MCU SDK Components?	5
1.3 What Drivers Does the MCU SDK Include?.....	6
1.4 Hardware Resources Consumed by MCU SDK Components.....	6
1.5 For More Information	8

1.1 What is the MCU SDK?

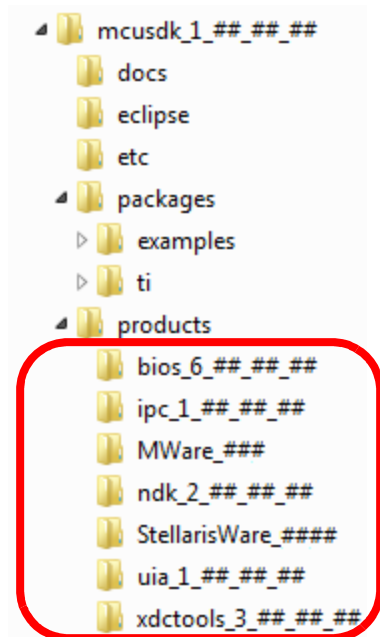
The MCU SDK (Micro-Controller Software Development Kit) makes it easier to develop applications for TI microcontrollers. This SDK contains several software components and examples that use these components together.

The MCU SDK is a one-stop RTOS solution for developing applications for TI embedded microcontrollers. It provides an OS kernel, communications support, drivers, and more. It is tightly integrated with TI's Code Composer Studio (CCS) development environment. In addition, examples demonstrate how to use each supported device and driver. These can be used as a starting point for your own projects.



1.2 What are the MCU SDK Components?

The MCU SDK contains its own source files, pre-compiled libraries (both instrumented and non-instrumented), and examples. Additionally, the MCU SDK contains a number of components within the MCU SDK's "products" subdirectory as shown here.



The components in the "products" subdirectory are:

- **SYS/BIOS.** SYS/BIOS is a scalable real-time kernel. It is designed to be used by applications that require real-time scheduling and synchronization or real-time instrumentation. It provides preemptive multi-threading, hardware abstraction, real-time analysis, and configuration tools. SYS/BIOS is designed to minimize memory and CPU requirements on the target. The FatFs module used by several examples is part of SYS/BIOS.
- **IPC.** This is a component containing packages that are designed to allow communication between processors in a multi-processor environment and communication to peripherals. This communication includes message passing, streams, and linked lists. These work transparently in both uni-processor and multi-processor configurations.
- **MWare.** The M3 portion of ControlSuite. It includes low level drivers and examples.¹
- **NDK.** The Network Developer's Kit (NDK) is a platform for development and demonstration of network enabled applications on TI embedded processors, currently limited to the TMS320C6000 family and ARM processors.
- **StellarisWare.** This software is an extensive suite of software designed to simplify and speed development of Stellaris-based microcontroller applications.
- **UIA.** The Unified Instrumentation Architecture provides target content that aids in the creation and gathering of instrumentation data (for example, Log data).
- **XDCtools.** This component provides the underlying tooling for configuring and building SYS/BIOS, IPC, NDK, and UIA.

To see the release notes for each component, you can select a component in the TI Resource Explorer under **MCU SDK > Products**.

The MCU SDK installs versions of these components that support only the device families supported by the MCU SDK. Currently, the MCU SDK provides support for the following boards:

- **TMDXDOCKH52C1** (both M3 and 28x sides of this Concerto board)
- **DK-LM3S9D96** (Stellaris board)
- **LM4F232H5QD** (Stellaris board)

If you want to use any of these components with other device families, you will need to download and install the complete component separately. For example, if you want to use SYS/BIOS with a C64x+ device, you will need to download and install the full SYS/BIOS product.

Examples are provided specifically for the supported boards, but libraries are provided for each of these device families, so that you can port the examples to similar boards.

1. The driverlib in MWare distributed with MCU SDK is rebuilt with the following compiler option: `--define=RTOS_SUPPLIED_INTERRUPTS`.

1.3 What Drivers Does the MCU SDK Include?

The MCU SDK includes drivers for the following peripherals. These drivers are in the `<install_dir>/packages/ti/drivers` directory. MCU SDK examples show how to use these drivers. Note that all of these drivers are built on top of MWare and StellarisWare.

- **EMAC.** Ethernet driver used by the networking stack (NDK) and not intended to be called directly.
- **SDSPI.** Driver for SD cards using an SPI (SSI) bus. This driver is used by the FatFS and not intended to be interfaced directly.
- **I²C.** API set intended to be used directly by the application or middleware.
- **GPIO.** API set intended to be used directly by the application or middleware to manage the GPIO pins and ports (and therefore the LEDs) on the board.
- **UART.** API set intended to be used directly by the application to communicate with the UART.
- **USBMSCHFatFs.** USB MSC Host under FatFS (for Flash drives). This driver is used by FatFS and is not intended to be called directly.
- **Other USB functionality.** See the USB examples for reference modules that provide support for the Human Interface Device (HID) class (mouse and keyboard) and Communications Device Class (CDC). This code is provided as part of the examples, not as a separate driver.

1.4 Hardware Resources Consumed by MCU SDK Components

This section briefly describes which hardware resources the MCU SDK and its dependent components consume by default. Some of these resources offer flexible options, whereas others are fixed in the current design or implementation.

1.4.1 TMDXDOCKH52C1 Resources Used

The following list shows which TMDXDOCKH52C1 peripheral resources are consumed by MCU SDK applications on that platform. The MCU SDK examples control which peripherals (and which ports) are used.

- **SYS/BIOS.** Uses the first general-purpose timer available and that timer's associated interrupts. Generally, this will be Timer 0. SYS/BIOS manages the Interrupt Vector Table.
- **IPC.** Uses the IPC registers including their associated interrupts.
- **MCU SDK.**
 - **Ethernet.** Uses the EMAC driver and its associated interrupts with the NDK to support networking.
 - **SD Card.** Uses FatFs and the SDSPI driver on SSI0 without interrupts to read and write to files on an SD Card.
 - **EEPROM.** Uses the I²C driver on I2C0 with its associated interrupts to read and write to the onboard EEPROM.
 - **GPIOs.** The GPIO driver is used on 2 onboard LEDs: LD2 (PC6_GPIO70) and LD3 (PC7_GPIO71) as output pins and one pin as an input pin PB4_GPIO12.
 - **Serial.** The UART driver uses UART0 which is attached to the FTDI USB chip to facilitate serial communications.
 - **USB.** The USB reference examples use the USB library and the USB controller with its associated interrupts.

1.4.2 **DK-LM3S9D96 Resources Used**

The following list shows which DK-LM3S9D96 peripheral resources are consumed by MCU SDK applications on that platform. The MCU SDK examples control which peripherals (and which ports) are used.

- **SYS/BIOS.** Uses the first general-purpose timer available and that timer's associated interrupts. Generally, this will be Timer 0. SYS/BIOS manages the Interrupt Vector Table.
- **MCU SDK.**
 - **Ethernet.** Uses the EMAC driver and its associated interrupts with the NDK to support networking.
 - **SD Card.** Uses FatFs and the SDSPI driver on SSI0 without interrupts to read and write to files on an SD Card.
 - **EEPROM.** Uses the I²C driver on I2C0 with its associated interrupts to read and write to the onboard EEPROM.
 - **GPIOs.** The GPIO driver is used on the onboard LED1 (GPIO-PIN_3).
 - **Serial.** The UART driver uses UART0 which is attached to the FTDI USB chip to facilitate serial communications.
 - **USB.** The USB reference examples use the USB library and the USB controller with its associated interrupts.

1.4.3 **EKS_LM4F232 Resources Used**

The following list shows which EKS-LM4F232 peripheral resources are consumed by MCU SDK applications on that platform. The MCU SDK examples control which peripherals (and which ports) are used.

- **SYS/BIOS.** Uses the first general-purpose timer available and that timer's associated interrupts. Generally, this will be Timer 0. SYS/BIOS manages the Interrupt Vector Table.
- **MCU SDK.**
 - **SD Card.** Uses FatFs and the SDSPI driver on SSI0 without interrupts to read and write to files on an SD Card.
 - **GPIOs.** The GPIO driver is used on 2 onboard LEDs: User LED (PG2) and SW1 (PM0).
 - **Serial.** The UART driver uses UART0 which is attached to the FTDI USB chip to facilitate serial communications.
 - **USB.** The USB reference examples use the USB library and the USB controller with its associated interrupts.

1.5 For More Information

To learn more about the MCU SDK and the software components used with it, refer to the following documentation. In addition, you can select a component in the TI Resource Explorer under **MCU SDK > Products** to see the release notes for that component.

- **MCU SDK**
 - [MCU SDK User's Guide \(SPRUHD4\)](#)
 - [SYS/BIOS on TI Embedded Processors Wiki](#)
 - [BIOS forum on TI's E2E Community](#)
- **Code Composer Studio (CCS)**
 - [CCS online help](#)
 - [CCSv5 on TI Embedded Processors Wiki](#)
 - [Code Composer forum on TI's E2E Community](#)
- **SYS/BIOS**
 - [SYS/BIOS 6 Getting Started Guide. <sysbios_install>/docs/Bios_Getting_Started_Guide.pdf](#)
 - [SYS/BIOS User's Guide \(SPRUEX3\)](#)
 - [SYS/BIOS online reference \(also called "CDOC"\).](#)
Open from CCS help or run [<sysbios_install>/docs/cdoc/index.html](#).
 - [SYS/BIOS on TI Embedded Processors Wiki](#)
 - [BIOS forum on TI's E2E Community](#)
 - [SYS/BIOS 6.x Product Folder](#)
 - [Embedded Software Download Page](#)
- **XDCtools**
 - [XDCtools online reference. Open from CCS help or run <xdc_install>/docs/xdctools.chm.](#)
 - [RTSC-Pedia Wiki](#)
 - [BIOS forum on TI's E2E Community](#)
 - [Embedded Software Download Page](#)
- **IPC**
 - [IPC User's Guide \(SPRUGO6\)](#)
 - [IPC online API reference. Run <ipc_install>/docs/doxygen/index.html.](#)
 - [IPC online configuration reference. Open from CCS help or run <ipc_install>/docs/cdoc/index.html.](#)
 - [Embedded Software Download Page](#)
- **NDK**
 - [NDK User's Guide \(SPRU523\)](#)
 - [NDK Programmer's Reference Guide \(SPRU524\)](#)
 - [NDK on TI Embedded Processors Wiki](#)
 - [BIOS forum on TI's E2E Community](#)
 - [Embedded Software Download Page](#)

- **UIA**
 - [System Analyzer User's Guide \(SPRUH43\)](#)
 - UIA online reference. Open from CCS help or run <uia_install>/docs/cdoc/index.html.
 - [System Analyzer on TI Embedded Processors Wiki](#)
 - [Embedded Software Download Page](#)
- **MWare and ControlSuite**
 - Documents in <mcu_sdk_install>/products/MWare_##/docs
 - [ControlSuite on TI Embedded Processors Wiki](#)
 - [ControlSuite Product Folder](#)
- **StellarisWare**
 - Documents in <mcu_sdk_install>/products/StellarisWare_####/docs
 - [StellarisWare Product Folder](#)
 - [Online StellarisWare Workshop](#)
- **FatFS API**
 - [API documentation](#)
- **Microcontroller devices**
 - [Concerto F28M35x Technical Reference Manual](#)
 - [C2000 on TI Embedded Processors Wiki](#)
 - [Concerto on TI Embedded Processors Wiki](#)
 - [Concerto Product Folder](#)
 - [Microcontrollers forum on TI's E2E Community](#)
- **SD Cards**
 - [Specification](#)
- **I²C**
 - [Specification](#)

Installing the MCU SDK

This chapter covers the required steps needed to install and build the MCU SDK.

Topic	Page
2.1 System Requirements	11
2.2 Installing Required Components	11
2.3 Installing the MCU SDK	13

2.1 System Requirements

The Windows version of the MCU SDK can be installed on systems running Windows 7, Windows Vista, or Windows XP (SP2 or SP3).

A Linux version of the MCU SDK is also available. Some restrictions on using SYS/BIOS apply when using the Linux version.

In order to install the MCU SDK, you must have at least 2 GB of free disk space. (If you have not yet installed Code Composer Studio, you will also need at least 2 GB of disk space for that installation.)

2.2 Installing Required Components

The MCU SDK is used in conjunction with Code Composer Studio 5.2 or higher and with System Analyzer 1.1 or higher.

2.2.1 *Installing Code Composer Studio (CCS)*

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

To install CCS 5.2, go to the product page at <http://www.ti.com/tool/ccstudio> and follow a link to download the software for your license type.

Run the executable installer, and answer the prompts as appropriate. We strongly recommend that you install CCS in the default installation directory of `c:\ti`.

When you are prompted for the type of Setup, select "Complete Feature Set". This setup type will automatically include a version of the SYS/BIOS 6 RTOS.

Note: The MCU SDK installs a version of SYS/BIOS that may be newer than the version installed by CCS. For the MCU SDK to work properly, you should use the SYS/BIOS version delivered with the MCU SDK.

2.2.2 Installing System Analyzer as a Software Update

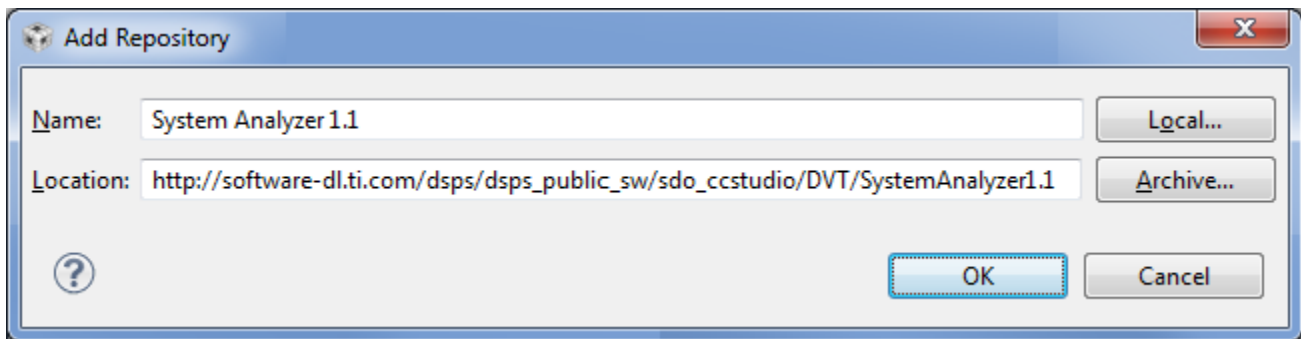
The MCU SDK is used in conjunction with System Analyzer 1.1 or higher. CCS 5.2 installs System Analyzer 1.0

If you have installed CCS 5.3 or higher, you do not need to install System Analyzer separately. System Analyzer 1.1 is included with CCS 5.3. You can check for updates by choosing **Help > Check for Updates** from the menus.

If you have CCS 5.2, you can install System Analyzer 1.1 by following these steps:

1. Choose **Help > Install New Software** from the CCS menus.
2. Click **Add** to the right of the Work with field.
3. In the Add Repository dialog, type "System Analyzer 1.1" as the **Name**.
4. In the **Location** field, type the following URL and then click **OK**:

`http://software-dl.ti.com/dsps/dsps_public_sw/sdo_ccstudio/DVT/SystemAnalyzer1.1`



5. Check the box next to DVT, and click **Next**. (System Analyzer is installed as part of the Data Visualization Technology component of CCS.) Continue clicking **Next** as needed and accept the license agreement as prompted.
6. Click **Finish** to install or update the DVT software component. When the installation is complete, restart CCS as prompted.

2.3 Installing the MCU SDK

The MCU SDK product comes delivered as an installer that needs to be installed in Code Composer Studio's installation directory.

2.3.1 *Installing on Windows*

To install the MCU SDK on a Windows host, follow these steps:

1. Exit from CCS if it is currently open.
2. Download the installer for the MCU SDK. For example, `mcusdk_setupwin32_1_##_##_##.exe`.
3. Run the downloaded file to install the full MCU SDK in the directory where CCS 5.2 is installed. By default, this is `c:\ti`. This is the recommended location for installing the MCU SDK.
4. Start CCS 5.2 or higher.
5. Wait for CCS to scan for newly installed products and display the **Extension Sites** window to notify you of the products that have been discovered. You should see the MCU SDK installation directory listed (for example, `c:\ti\mcusdk_1_##_##_##` by default). Leave the MCU SDK item checked, and click **Finish** to add it to CCS.
6. You will see a window that asks if you want to restart CCS now. Click **Yes**.

2.3.2 *Installing on Linux*

To install the MCU SDK on a Linux host, follow these steps:

1. Exit CCS if it is currently open.
2. Download the installer for the MCU SDK. For example, `mcusdk_setuplinux_1_##_##_##.bin`.
3. You may want to log in as root before performing the installation. It is also possible to run the installation from your user account.
4. Run the downloaded file to install the full MCU SDK. Accept the defaults from the Linux installer. (The installer detects whether you are running it as user or root.)
5. Start CCS 5.2 or higher.
6. Wait for CCS to scan for newly installed products and display the **Extension Sites** window to notify you of the products that have been discovered. You should see the MCU SDK installation directory listed. Leave the MCU SDK item checked, and click **Finish** to add it to CCS.

Examples for the MCU SDK

The MCU SDK comes with a number of examples that illustrate on how to use the individual components. This chapter explains how to create and use these examples.


Topic	Page
3.1 Creating an Example Project Using the TI Resource Explorer	15
3.2 Example Overview	16
3.3 Example Settings	17
3.4 Installing USB Drivers for the USB Device Examples	19
3.5 Using Examples for Individual Components	20

3.1 Creating an Example Project Using the TI Resource Explorer

The MCU SDK uses TI Resource Explorer within CCS to let you quickly create example projects and open documentation. To open this tool, choose **View > TI Resource Explorer** in CCS and select the **MCU SDK** item.

Follow these steps to create a new MCU SDK example in a CCS workspace:

1. Start CCS.
2. If the TI Resource Explorer tab is closed, choose **View > TI Resource Explorer** to open it.
3. Expand the MCU SDK item in the tree to show **MCU SDK > Devices > board > Examples**, where *board* is your platform.
4. Select the example you want to create. See Section 3.2, *Example Overview* and the *MCU SDK User's Guide* (SPRUHD4) for information about individual examples.
5. Click the **Step 1** link in the right pane of the TI Resource Explorer to **Import the example project into CCS**. This adds a new project to your Project Explorer view.

Step 1:  [Import the example project into CCS](#)


6. The project created will have a name with the format `<board>_<example>`. You can expand the project to view or change the source code and configuration file.

The page shown when you select an example in the TI Resource Explorer provides additional links to perform common actions with that example.


- Use the **Step 2** link when you are ready to build the project. If you want to change any build options, right click on the project and select **Properties** from the context menu. For example, you can change compiler, linker, and RTSC (XDCtools) options.

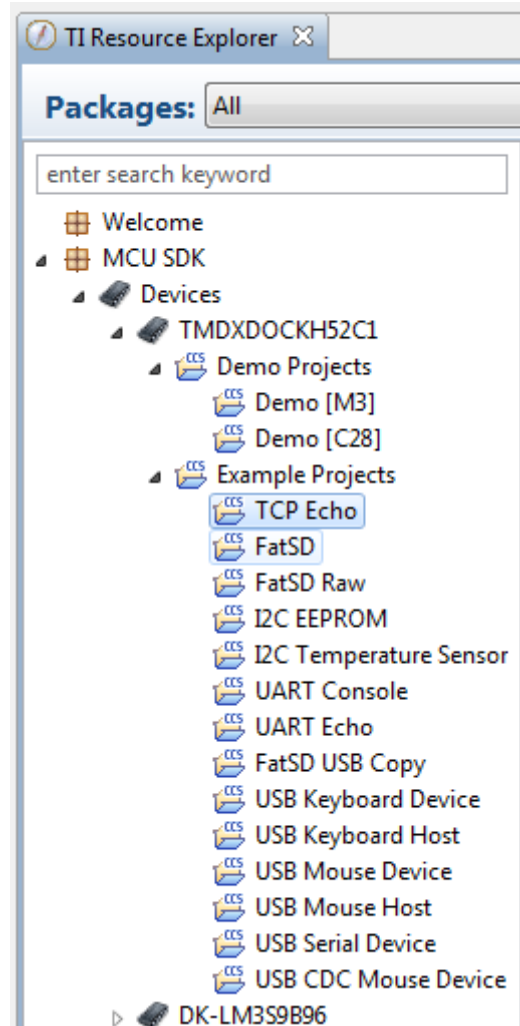
Step 2:  [Build the imported project](#)

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

Step 3:  [Debugger Configuration](#)

- Use the **Step 4** link to launch a debug session for the project and switch to the CCS Debug Perspective. (If you are running a multi-core example, see the "Examples for the MCU SDK" chapter in the *MCU SDK User's Guide* (SPRUHD4) for instructions for running the example.)

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



3.2 Example Overview

The components and hardware used by the MCU SDK examples are shown in the following table. Details about these examples are provided in the *MCU SDK User's Guide* (SPRUHD4).

Table 3-1. Components Used by MCU SDK Examples

Example	SYS/BIOS	FatFS		USB Devices			I ² C	UART	NDK / EMAC	UIA	IPC	GPIO (& LED)
		SD Card / SDSPI	USB Drive	HID	CDC	MSC						
Empty MCU SDK Project	X									X		X
Demo for M3 / Demo for C28 *	X	X			X		X	X	X	X	X	X
TCP Echo	X								X	X		X
FatSD: FatFs File Copy	X	X								X		X
FatSD Raw: FatFs File Copy using FatFs APIs	X	X								X		X
I ² C EEPROM: Communications with the onboard EEPROM *	X						X			X		X
UART Console	X				X			X**		X		X
UART Echo	X							X		X		X
FatSD USB Copy: (SD Card and USB Drive)	X	X	X			X				X		X
USB Keyboard Device	X			X						X		X
USB Keyboard Host	X			X						X		X
USB Mouse Device	X			X						X		X
USB Mouse Host	X			X						X		X
USB Serial Device	X				X					X		X
USB CDC Mouse Device	X			X	X					X		X

* The Demo example is not available for the DK-LM3S9D96 and EKS-LM4F232 boards. The TCP Echo and I²C examples are not available for the EKS-LM4F232 board. I²C communications with the onboard EEPROM for DK-LM3S9D96 requires an additional daughterboard.

** UART is used by SysFlex for sending System_printf() and printf() output to a console. Other examples use SysMin.

3.3 Example Settings

There is a separate readme.txt file for each of the examples. These files are added to your CCS project when you use the TI Resource Explorer to create a project.

The readme.txt files contain the following types of information:

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

The subsections that follow list settings required to run the MCU SDK examples on the supported boards.

3.3.1 TMDXDOCKH52C1 Development Board

Jumper settings:

- J01-J15: B-C position (Ethernet)
- J20-J21: B-C position (I2C EEPROM)
 - J6: 2-3 position (I2C EEPROM Write protection off)
- J22-J25: B-C position (SPI/SSI SD Card slot)
- J30-J31: B-C position (USB Host and Device)
 - A board modification is required for the USB host examples (see Section 3.3.1.2)

Switch settings:

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

3.3.1.1 Setting the MAC Address

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

```

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

```

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

```

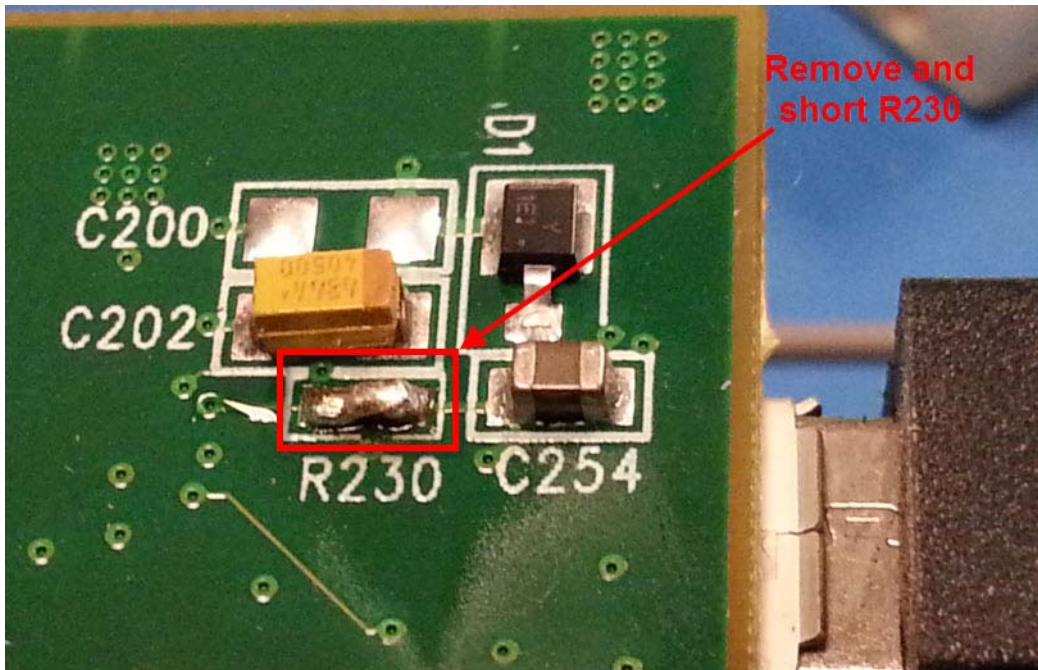
UInt8 macAddress[6] = {0xA8, 0x63, 0xF2, 0x00, 0x05, 0x1A};

```

On the DK-LM3S9D96, the MAC address is automatically read from the user registers in Flash memory. You do not need to modify the board-specific C file.

3.3.1.2 USB Host Mode Board Modification

Using the USB controller in host mode on the TMDXDOCKH52C1 requires a hardware modification to the control card. This modification is not required when using the USB controller in device mode.



Remove and short resistor R230 on the control card.

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

3.3.2 DK_LM3S9D96 Board

Jumper settings:

- JP01-JP02: Closed (Ethernet jack LEDs)
- JP05-JP06: Closed (User LED and User Switch)
- JP09-JP12: Closed (SPI/SSI SD Card slot)
- JP03, JP37-JP38: Closed (VBUS, USB0EPE, and USB0PFLT)

Switch settings:

- SW1: Some examples use PJ7 as an input.

3.3.3 LM4F120H5QR Board

Jumper settings: N/A

Switch settings:

- SW1: Some examples use PM0 as an input.

3.4 Installing USB Drivers for the USB Device Examples

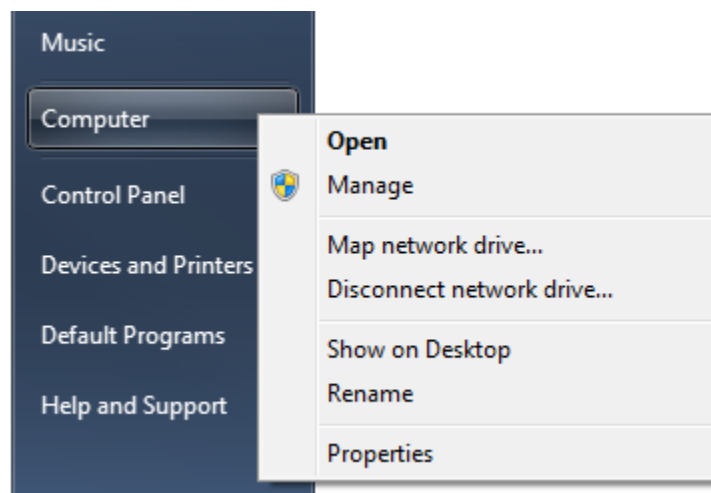
The USB examples build upon the examples provide with StellarisWare and MWare. Because the examples mimic the same functionality, you can use the same drivers delivered with standalone installations of StellarisWare and Control Suite (MWare).

In the MCU SDK, the Windows USB drivers are located in the `<mcusdk_install_dir>/products/StellarisWare_####/windows_drivers` directory.

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

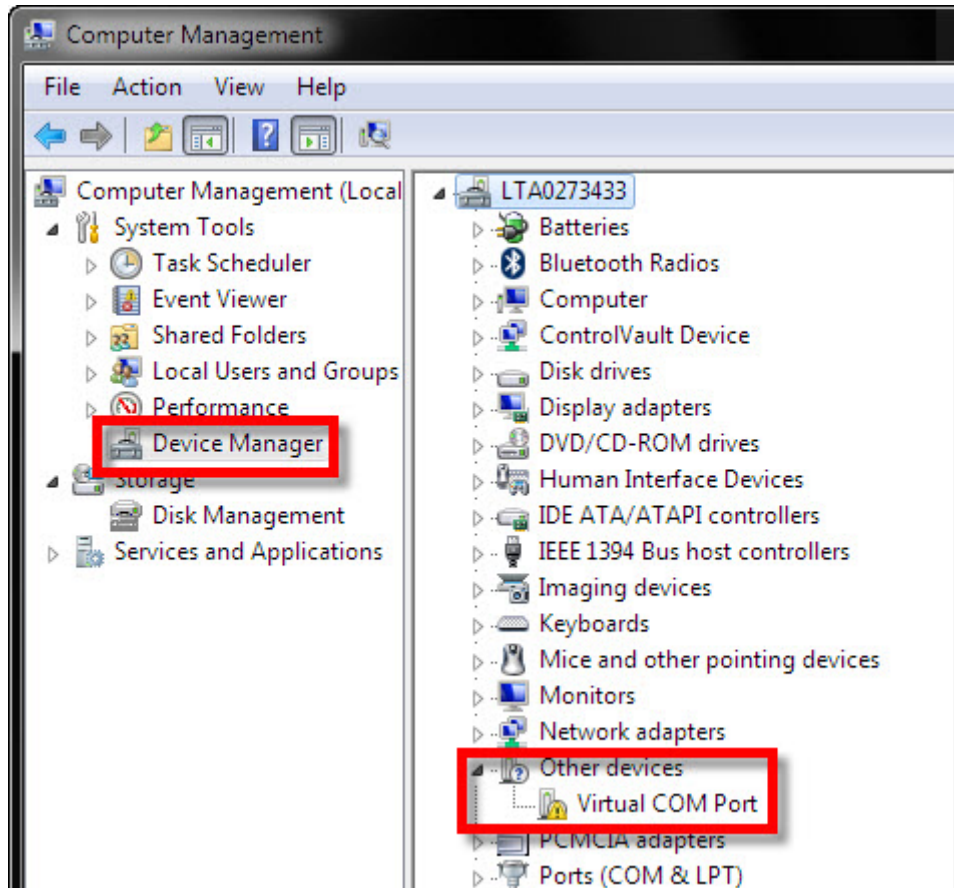
To install the USB driver, follow these steps:

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



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

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



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

3.5 Using Examples for Individual Components

An "empty" MCU SDK project is provided in the CCS new project wizard. It contains some common code excerpts that enable different MCU SDK components. This example is described in the *MCU SDK User's Guide* (SPRUHD4).

Configuring the MCU SDK

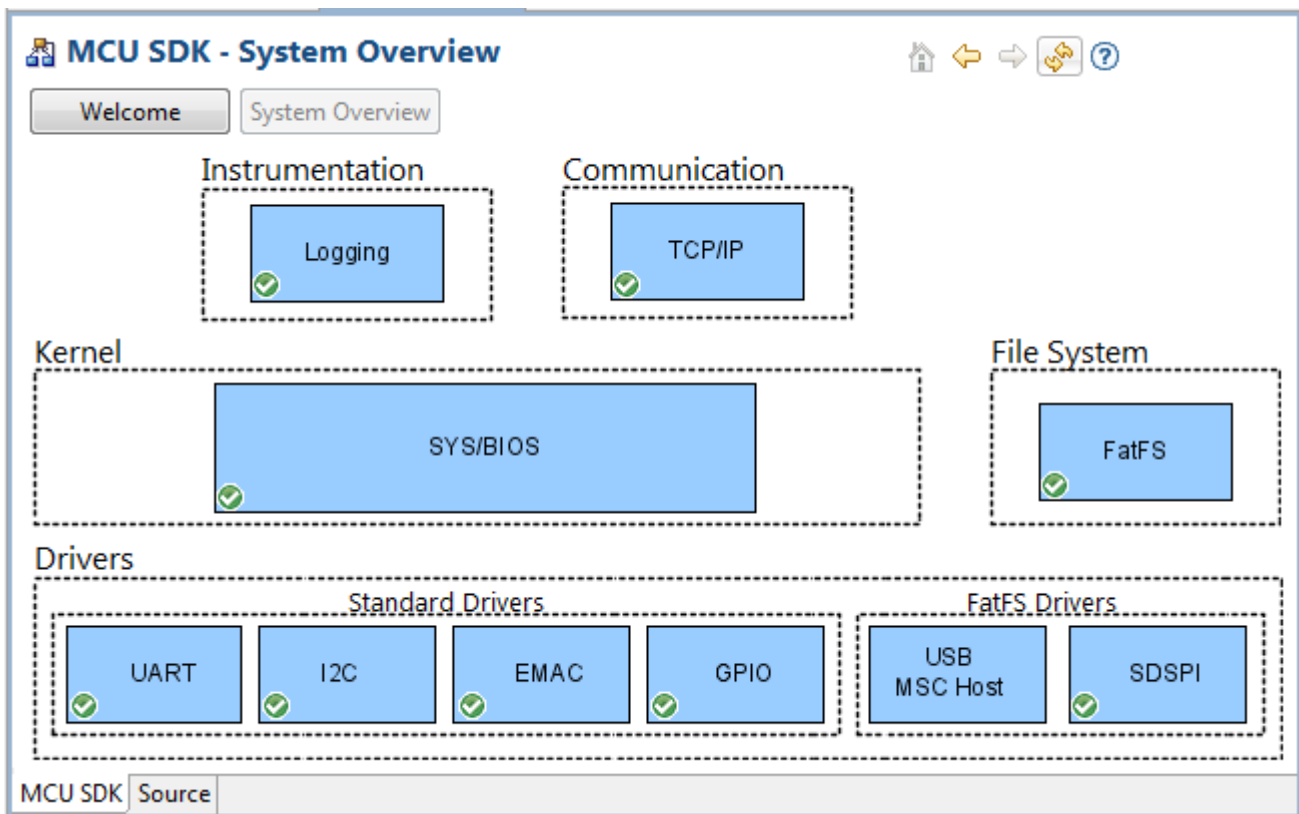
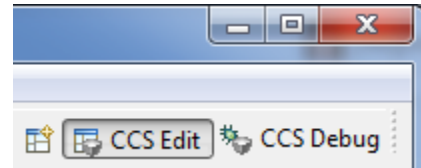
This chapter describes how to configure how MCU SDK and its components will be used by your application.

Topic	Page
4.1 Starting the Configuration Tool	22
4.2 Configuring MCU SDK	23
4.3 Configuring Individual Sub-Components	23

4.1 Starting the Configuration Tool

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

1. Make sure you are in the **C/C++** perspective of CCS. If you are not in that perspective, click the C/C++ icon to switch back.
2. Double-click on the *.cfg configuration file for an MCU SDK example project in the **Project Explorer** tree. (See Section 3.1 if you need to create an example project.) While XGCONF is opening, the CCS status bar shows that the configuration is being processed and validated.
3. When XGCONF opens, you see the **Welcome** sheet for MCU SDK. This sheet provides links to MCU SDK documentation resources.
4. Click the **System Overview** button to see a handy overview of the components available through the MCU SDK. The green check marks indicate which modules are being used by the application.



5. Click a blue box in the System Overview to go to the Welcome sheet for that component or the configuration page for a driver.

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

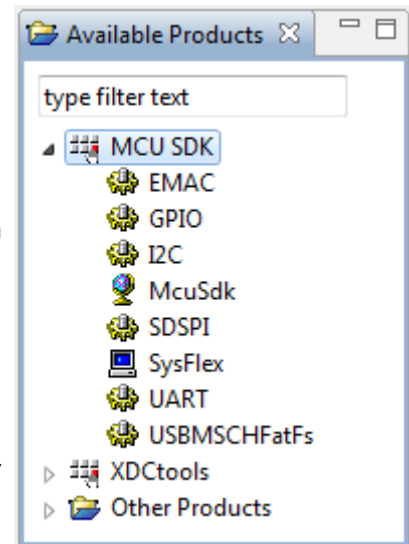
For details about how to use XGCONF, see Chapter 2 of the [SYS/BIOS User's Guide \(SPRUEX3\)](#).

4.2 Configuring MCU SDK

When you open a configuration file in XGCONF, you see a list of Available Products. The components listed here are the same components that were checked in the RTSC Configuration Settings page when you created a new CCS project using one of the MCU SDK project templates.

Under the MCU SDK item, you can select one of the modules listed to configure it. Most of these modules are peripheral drivers. For each driver, you can select to use either the instrumented or non-instrumented libraries when linking. The instrumented libraries process Log events while the non-instrumented libraries do not. See the section on "Using Instrumented or Non-Instrumented Libraries" in the *MCU SDK User's Guide* (SPRUHD4) for more information. The following drivers can be configured:

- **EMAC.** Driver for Ethernet with the NDK. See Section 3.3.1.1 for how to set the MAC address.
- **GPIO.** Driver for the GPIO pins (and therefore the LEDs).
- **I2C.** Driver for the I²C peripheral.
- **SDSPI.** Driver for the SD card using an SPI (SSI) bus and the FatFS.
- **UART.** Driver for the UART peripheral.
- **USBMSCHFatFs.** Driver for the USB MSC Host controller.



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

Other SystemSupport implementations are provided with XDCtools.

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

4.3 Configuring Individual Sub-Components

For information about configuring individual sub-components of the MCU SDK, see the documentation for that component. Chapter 2 of the [SYS/BIOS User's Guide \(SPRUEX3\)](#) provides information about using XGCONF.

Within XGCONF, you can see the full file path to the version of the component being used by hovering your mouse cursor over a component in the "Other Products" list in the **Available Products** area.

The CCS online help lets you access the CDOC reference help for various components. For information about properties you can configure and instances you can create, scroll down in CDOC to see the sections with red outlines. (You can use the XDCscript link at the top of each page to skip to the configuration information.)

Index

A

Available Products list 23

C

C28x
 support 5
CCS
 installation 11
 other documentation 8
CDC device 6
 example 16
components 5
Concerto 5
 other documentation 9
 resources used 6
configuration 21
 graphical editor 22
ControlSuite 5
 other documentation 9

D

disk space 11
DK-LM3S9D96 5
 resources used 7
documentation 8

E

EKS-LM4F232
 resources used 7
EMAC driver 6
 configuration 23
 resources used 6, 7
Ethernet driver 6
examples 14
 creating projects 15
 overview 16

F

FatFs API
 other documentation 9
Flash drives 6
forum 8

G

GPIO driver 6
 configuration 23
 resources used 6, 7

H

HID device 6
 example 16

I

I2C driver 6
 configuration 23
 example 16
 resources used 6, 7
installation
 CCS 11
 directory 11
 MCU SDK 13
instrumented libraries 23
IPC 5
 example 16
 other documentation 8
 resources used 6

L

LEDs
 managed by GPIO driver 6
LM4F232H5QD 5

M

MAC address 17
MCU SDK 4
 other documentation 8
MSC device 6
MWare 5
 other documentation 9

N

NDK 5

example 16
MAC address 17
other documentation 8
non-instrumented libraries 23

P

products directory 5

R

readme.txt file 17

S

SD driver
 example 16
SDSPI driver 6
 configuration 23
 resources used 6, 7
StellarisWare 5
SYS/BIOS 5
 examples 16
 other documentation 8
 resources used 6
SysFlex module
 configuration 23
System Overview configuration 22
system requirements 11

T

TMDXDOCKH52C1 5

U

UART driver 6
 configuration 23
 resources used 6, 7
UIA 5
 example 16
 other documentation 9
USB driver
 example 16
 resources used 6, 7
USBMSCHFatFs driver 6
 configuration 23

W

wiki 8

X

XDCtools 5
 other documentation 8
XGCONF
 configuring MCU SDK modules 23
 configuring other components 23
 starting 22

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

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

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

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

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

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

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products

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

Applications

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

TI E2E Community Home Page e2e.ti.com

Mailing Address: Texas Instruments, Post Office Box 655303 Dallas, Texas 75265

Copyright © 2012, Texas Instruments Incorporated