DK-TM4C129X-BOOSTXL-SENSHUB
Firmware Development Package

USER'S GUIDE

SW-DK-TM4C129X-BOOSTXL-SENSHUB-UG-2.1.2.111

Copyright

Texas Instruments
108 Wild Basin, Suite 350
Austin, TX 78746
www.ti.com/tiva-c

Revision Information

This is version 2.1.2.111 of this document, last updated on December 16, 2015.

Table of Contents

Copyright . . . 2

Revision Information . . . 2

1 Introduction . . . 5

2 Example Applications . . . 7
2.1 Motion Air Mouse (airmouse) . . . 7
2.2 Nine Axis Sensor Fusion with the MPU9150 and Complimentary-Filtered DCM (compdcm_mpu9150) 8
2.3 Humidity Measurement with the SHT21 (humidity_sht21) . . . 8
2.4 Light Measurement with the ISL29023 (light_isl29023) . . . 8
2.5 Pressure Measurement with the BMP180 (pressure_bmp180) . . . 9
2.6 Temperature Measurement with the TMP006 (temperature_tmp006) . . . 9

3 Frame Module . . . 11
3.1 Introduction . . . 11
3.2 API Functions . . . 11
3.3 Programming Example . . . 12

4 Kentec 320x240x16 Display Driver . . . 13
4.1 Introduction . . . 13
4.2 API Functions . . . 13
4.3 Programming Example . . . 14

5 MX66L51235F Driver . . . 17
5.1 Introduction . . . 17
5.2 API Functions . . . 17
5.3 Programming Example . . . 17

6 Pinout Module . . . 19
6.1 Introduction . . . 19
6.2 API Functions . . . 19
6.3 Programming Example . . . 19

7 Sound Driver . . . 21
7.1 Introduction . . . 21
7.2 API Functions . . . 21
7.3 Programming Example . . . 21

8 Touch Screen Driver . . . 23
8.1 Introduction . . . 23
8.2 API Functions . . . 24
8.3 Programming Example . . . 24

IMPORTANT NOTICE . . . 26

1 Introduction

The Texas Instruments® Tiva™ DK-TM4C129X-BOOSTXL-SENSHUB development board is a platform that can be used for software development and prototyping a hardware design. It can also be used as a guide for custom board design using a Tiva microcontroller.

The DK-TM4C129X-BOOSTXL-SENSHUB includes a Tiva ARM® Cortex™-M4-based microcontroller and the following features:

- Tiva™ TM4C129XNCZAD microcontroller
- TFT display (320x240 16 bpp) with capacitive touch screen overlay
- Ethernet connector
- USB OTG connector
- 64 MB SPI flash
- MicroSD card connector
- Temperature sensor
- Speaker with class A/B amplifier
- 3 user buttons
- User LED
- 2 booster pack connectors
- EM connector
- On-board In-Circuit Debug Interface (ICDI)
- Power supply option from USB ICDI connection or external power connection
- Shunt for microcontroller current consumption measurement

2 Example Applications

The example applications show how to utilize features of the DK-TM4C129X development board. Examples are included to show how to use many of the general features of the Tiva microcontroller, as well as the feature that are unique to this development board.

A number of drivers are provided to make it easier to use the features of the DK-TM4C129X. These drivers also contain low-level code that make use of the TivaWare peripheral driver library and utilities.

There is an IAR workspace file (`dk-tm4c129x-boostxl-senshub.eww`) that contains the peripheral driver library project, along with all of the board example projects, in a single, easy-to-use workspace for use with Embedded Workbench version 5.

There is a Keil multi-project workspace file (`dk-tm4c129x-boostxl-senshub.mpw`) that contains the peripheral driver library project, along with all of the board example projects, in a single, easy-to-use workspace for use with uVision.

All of these examples reside in the `examples/boards/dk-tm4c129x-boostxl-senshub` subdirectory of the firmware development package source distribution.

2.1 Motion Air Mouse (airmouse)

This example demonstrates the use of the Sensor Library, DK-TM4C129X and the SensHub BoosterPack to fuse nine axis sensor measurements into motion and gesture events. These events are then transformed into mouse and keyboard events to perform standard HID tasks.

Connect the USB OTG port, between the EM connectors of the DK-TM4C129X, to a standard computer USB port. The DK-TM4C129X with SensHub BoosterPack enumerates on the USB bus as a composite HID keyboard and mouse.

Hold the DK-TM4C129X with the buttons and LCD away from the user and toward the computer with USB Device cable exiting from the left side of the board.

- Roll or tilt the DK-TM4C129X to move the mouse cursor of the computer up, down, left and right.
- The buttons on the DK-TM4C129X, SEL and DOWN, perform the left and right mouse click actions respectively. The buttons on the SensHub BoosterPack are not currently used by this example.
- A quick spin of the DK-TM4C129X generates a PAGE_UP or PAGE_DOWN keyboard press and release depending on the direction of the spin. This motion simulates scrolling.
- A quick horizontal jerk to the left or right generates a CTRL+ or CTRL- keyboard event, which creates the zoom effect used in many applications, especially web browsers.
- A quick vertical lift generates an ALT+TAB keyboard event, which allows the computer user to select between currently open windows.
- A quick twist to the left or right moves the window selector.

- A quick jerk in the down direction selects the desired window and closes the window selection dialog.

This example also supports the RemoTI low power RF Zigbee® human interface device profile. The wireless features of this example require the CC2533EMK expansion card and the CC2531EMK USB Dongle. For details and instructions for wireless operations see the Wiki at `http://processors.wiki.ti.com/index.php/Wireless_Air_Mouse_Guide.`

2.2 Nine Axis Sensor Fusion with the MPU9150 and Complimentary-Filtered DCM (compdcm_mpu9150)

This example demonstrates the basic use of the Sensor Library, DK-TM4C129X and SensHub BoosterPack to obtain nine axis motion measurements from the MPU9150. The example fuses the nine axis measurements into a set of Euler angles: roll, pitch and yaw. It also produces the rotation quaternions. The fusion mechanism demonstrated is complimentary-filtered direct cosine matrix (DCM) algorithm that is provided as part of the Sensor Library.

The raw sensor measurements, Euler angles and quaternions are printed to LCD and terminal. Connect a serial terminal program to the DK-TM4C129X's ICDI virtual serial port at 115,200 baud. Use eight bits per byte, no parity and one stop bit. The blue LED blinks to indicate the code is running.

2.3 Humidity Measurement with the SHT21 (humidity_sht21)

This example demonstrates the basic use of the Sensor Library, DK-TM4C129X and SensHub BoosterPack to obtain temperature and relative humidity of the environment using the Sensirion SHT21 sensor.

The humidity and temperature as measured by the SHT21 is printed to LCD and terminal. Connect a serial terminal program to the DK-TM4C129X's ICDI virtual serial port at 115,200 baud. Use eight bits per byte, no parity and one stop bit. The blue LED blinks to indicate the code is running.

2.4 Light Measurement with the ISL29023 (light_isl29023)

This example demonstrates the basic use of the Sensor Library, DK-TM4C129X and the SensHub BoosterPack to obtain ambient and infrared light measurements with the ISL29023 sensor.

Note that the jumper on J36 for PQ7 must be disconnect to GREEN led as PQ7 is also used as INT signal by ISL29023.

The raw sensor measurements are printed to LCD and terminal. Connect a serial terminal program to the DK-TM4C129X's ICDI virtual serial port at 115,200 baud. Use eight bits per byte, no parity and one stop bit. The blue LED blinks to indicate the code is running.

2.5 Pressure Measurement with the BMP180 (pressure_bmp180)

This example demonstrates the basic use of the Sensor Library, DK-TM4C129X and the SensHub BoosterPack to obtain air pressure and temperature measurements with the BMP180 sensor.

The raw sensor measurements are printed to LCD and terminal. Connect a serial terminal program to the DK-TM4C129X's ICDI virtual serial port at 115,200 baud. Use eight bits per byte, no parity and one stop bit. The blue LED blinks every time data is read from the BMP180 sensor.

2.6 Temperature Measurement with the TMP006 (temperature_tmp006)

This example demonstrates the basic use of the Sensor Library, DK-TM4C129X and the SensHub BoosterPack to obtain ambient and object temperature measurements with the Texas Instruments TMP006 sensor.

Note that the jumper on J36 for PQ7 must be disconnect to GREEN led as PQ7 is also used as DRDY signal by TMP006.

The raw sensor measurements are printed to LCD and terminal. Connect a serial terminal program to the DK-TM4C129X's ICDI virtual serial port at 115,200 baud. Use eight bits per byte, no parity and one stop bit. The blue LED blinks to indicate the code is running.

3 Frame Module

Introduction 11
API Functions 11
Programming Example 12

3.1 Introduction

The frame module is a common function for drawing an application frame on the display. This is used by the example applications to provide a uniform appearance.

This driver is located in `examples/boards/dk-tm4c129x-boostxl-senshub/drivers`, with `frame.c` containing the source code and `frame.h` containing the API declarations for use by applications.

3.2 API Functions

Functions

- void FrameDraw (tContext *psContext, const char *pcAppName)

3.2.1 Function Documentation

3.2.1.1 FrameDraw

Draws a frame on the LCD with the application name in the title bar.

Prototype:

```
void
FrameDraw(tContext *psContext,
          const char *pcAppName)
```

Parameters:

psContext is a pointer to the graphics library context used to draw the application frame.

pcAppName is a pointer to a string that contains the name of the application.

Description:

This function draws an application frame onto the LCD, using the supplied graphics library context to access the LCD and the given name in the title bar of the application frame.

Returns:

None.

3.3 Programming Example

The following example shows how to draw the application frame.

```
//
// The frame example.
//
void
FrameExample(void)
{
    tContext sContext;

    //
    // Draw the application frame.  This code assumes the the graphics library
    // context has already been initialized.
    //
    FrameDraw(&sContext, "example");
}
```

4 Kentec 320x240x16 Display Driver

Introduction 13
API Functions 13
Programming Example 14

4.1 Introduction

The display driver offers a standard interface to access display functions on the Kentec K350QVG-V2-F 320x240 16-bit color TFT display and is used by the TivaWare Graphics Library and widget manager. The display is controlled by the embedded SSD2119 display controller, which provides the frame buffer for the display. In addition to providing the `tDisplay` structure required by the graphics library, the display driver also provides an API for initializing the display.

The display driver can be built to operate in one of four orientations:

- **LANDSCAPE** - In this orientation, the screen is wider than it is tall; this is the normal orientation for a television or a computer monitor, and is the normal orientation for photographs of the outdoors (hence the name). For the K350QVG-V2-F, the flex connector is on the bottom side of the screen when viewed in **LANDSCAPE** orientation.

- **PORTRAIT** - In this orientation, the screen is taller than it is wide; this is the normal orientation of photographs of people (hence the name). For the K350QVG-V2-F, the flex connector is on the left side of the screen when viewed in **PORTRAIT** orientation.

- **LANDSCAPE_FLIP** - **LANDSCAPE** mode rotated 180 degrees (in other words, the flex connector is on the top side of the screen).

- **PORTRAIT_FLIP** - **PORTRAIT** mode rotated 180 degrees (in other words, the flex connector is on the right side of the screen).

One of the above highlighed defines selects the orientation that the display driver will use. If none is defined, the default orientation is **LANDSCAPE_FLIP** (which corresponds to how the display is mounted to the DK-TM4C129X development board).

This driver is located in `examples/boards/dk-tm4c129x-boostxl-senshub/drivers`, with `kentec320x240x16_ssd2119.c` containing the source code and `kentec320x240x16_ssd2119.h` containing the API declarations for use by applications.

4.2 API Functions

Functions

- void Kentec320x240x16_SSD2119Init (uint32_t ui32SysClock)

Variables

- const tDisplay g_sKentec320x240x16_SSD2119

4.2.1 Function Documentation

4.2.1.1 Kentec320x240x16_SSD2119Init

Initializes the display driver.

Prototype:

```
void
Kentec320x240x16_SSD2119Init(uint32_t ui32SysClock)
```

Parameters:

ui32SysClock is the frequency of the system clock.

Description:

This function initializes the LCD controller and the SSD2119 display controller on the panel, preparing it to display data.

Returns:

None.

4.2.2 Variable Documentation

4.2.2.1 g_sKentec320x240x16_SSD2119

Definition:

```
const tDisplay g_sKentec320x240x16_SSD2119
```

Description:

The display structure that describes the driver for the Kentec K350QVG-V2-F TFT panel with an SSD2119 controller.

4.3 Programming Example

The following example shows how to initialize the display and prepare to draw on it using the graphics library.

```
//
// The Kentec 320x240x16 SSD2119 example.
//
void
Kentec320x240x16_SSD2119Example(void)
{
    uint32_t ui32SysClock;
    tContext sContext;
```

```
    //
    // Initialize the display.  This code assumes that ui32SysClock has been
    // set to the clock frequency of the device (for example, the value
    // returned by SysCtlClockFreqSet).
    //
    Kentec320x240x16_SSD2119Init(ui32SysClock);

    //
    // Initialize a graphics library drawing context.
    //
    GrContextInit(&sContext, &g_sKentec320x240x16_SSD2119);
}
```

5 MX66L51235F Driver

Introduction 17
API Functions 17
Programming Example 17

5.1 Introduction

The MX66l51235F driver provides functions to make it easy to use the MX66L51235F SPI flash on the DK-TM4C129X development board. The driver provides a function to read, erase, and program the SPI flash.

On the DK-TM4C129X development board, the SPI flash shares a SPI port with the SD card socket. If not properly initialized into SPI mode, the SD card will occasionally drive data onto the SPI bus despite the fact that it is “not selected” (which is in fact valid since there is not chip select for an SD card in SD card mode). Therefore, the SD card must be properly initialized (via a call to the disk_initialize() function in the SD card driver) prior to using this driver to access the SPI flash.

This driver is located in `examples/boards/dk-tm4c129x-boostxl-senshub/drivers`, with `mx66l51235f.c` containing the source code and `mx66l51235f.h` containing the API declarations for use by applications.

5.2 API Functions

5.3 Programming Example

The following example shows how to use the MX66L51235F driver to read and write data in the SPI flash.

```
//
// A buffer to hold the data read from and written to the SPI flash.
//
uint8_t g_pui8MX66L51235FData[32];

//
// The MX66L51235F example.
//
void
MX66L51235FExample(void)
{
    uint32_t ui32SysClock;

    //
    // Initialize the SPI flash driver.  This code assumes that ui32SysClock
    // has been set to the clock frequency of the device (for example, the
    // value returned by SysCtlClockFreqSet).
    //
    MX66L51235FInit(ui32SysClock);

    //
    // Erase the first sector (4 KB) of the SPI flash.
```

```
    //
    MX66L51235FSectorErase(0);

    //
    // Program some data into the first page of the SPI flash.  This assumes
    // that the data buffer has been filled with the data to be programmed.
    //
    MX66L51235FPageProgram(0, g_pui8MX66L51235FData,
                           sizeof(g_pui8MX66L51235FData));

    //
    // Read some data from the second page of the SPI flash.
    //
    MX66L51235FRead(0x100, g_pui8MX66L51235FData,
                    sizeof(g_pui8MX66L51235FData));
}
```

6 Pinout Module

Introduction 19
API Functions 19
Programming Example 19

6.1 Introduction

The pinout module is a common function for configuring the device pins for use by example applications. The pins are configured into the most common usage; it is possible that some of the pins might need to be reconfigured in order to support more specialized usage.

This driver is located in `examples/boards/dk-tm4c129x-boostxl-senshub/drivers`, with `pinout.c` containing the source code and `pinout.h` containing the API declarations for use by applications.

6.2 API Functions

Functions

- void PinoutSet (void)

6.2.1 Function Documentation

6.2.1.1 PinoutSet

Configures the device pins for the standard usages on the DK-TM4C129X.

Prototype:

```
void
PinoutSet(void)
```

Description:

This function enables the GPIO modules and configures the device pins for the default, standard usages on the DK-TM4C129X. Applications that require alternate configurations of the device pins can either not call this function and take full responsibility for configuring all the device pins, or can reconfigure the required device pins after calling this function.

Returns:

None.

6.3 Programming Example

The following example shows how to configure the device pins.

```
//
// The pinout example.
//
void
PinoutExample(void)
{
    //
    // Configure the device pins.
    //
    PinoutSet();
}
```

7 Sound Driver

Introduction 21
API Functions 21
Programming Example 21

7.1 Introduction

The sound driver provides a set of functions to stream 16-bit PCM audio data to the speaker on the DK-TM4C129X development board. The audio can be played at 8 kHz, 16 kHz, 32 kHz, or 64 kHz; in each case the data is output to the speaker at 64 kHz, and at lower playback rates the intervening samples are computed via linear interpolation (which is fast but introduces high-frequency artifacts).

The audio data is supplied via a ping-pong buffer. This is a buffer that is logically split into two halves; the "ping" half and the "pong" half. While the sound driver is playing data from one half, the application is responsible for filling the other half with new audio data. A callback from the sound driver indicates when it transitions from one half to the other, which provides the indication that one of the halves has been consumed and must be filled with new data.

The sound driver utilizes timer 5 subtimer A. The interrupt from the timer 5 subtimer A is used to process the audio stream; the SoundIntHandler() function should be called when this interrupt occurs (which is typically accomplished by placing it in the vector table in the startup code for the application).

This driver is located in `examples/boards/dk-tm4c129x-boostxl-senshub/drivers`, with `sound.c` containing the source code and `sound.h` containing the API declarations for use by applications.

7.2 API Functions

7.3 Programming Example

The following example shows how to use the sound driver to playback a stream of 8 kHz audio data.

```
//
// The buffer used as the audio ping-pong buffer.
//
#define SOUND_NUM_SAMPLES       256
int16_t g_pi16SoundExampleBuffer[SOUND_NUM_SAMPLES];

//
// The callback function for when half of the buffer has been consumed.
//
void
SoundExampleCallback(uint32_t ui32Half)
{
    //
    // Generate new audio in the half the has just been consumed.  As this is
    // in the context of the timer interrupt, it is possible/likely that this
```

```
    // should just set a flag to trigger something else (outside of interrupt
    // context) to do the actual work.  In either case, this needs to return
    // prior to the next timer interrupt (in other words, within ~15 us).
    //
}

//
// The sound example.
//
void
SoundExample(void)
{
    uint32_t ui32SysClock;

    //
    // Initialize the sound driver.  This code assumes that ui32SysClock has
    // been set to the clock frequency of the device (for example, the value
    // returned by SysCtlClockFreqSet).
    //
    SoundInit(ui32SysClock);

    //
    // Prefill the audio buffer with the first segment of the audio to be
    // played.
    //

    //
    // Start the playback of audio.
    //
    SoundStart(g_pi16SoundExampleBuffer, SOUND_NUM_SAMPLES, 8000,
               SoundExampleCallback);
}
```

8 Touch Screen Driver

Introduction ... 23
API Functions ... 24
Programming Example ... 24

8.1 Introduction

The touch screen is a pair of resistive layers on the surface of the display. One layer has connection points at the top and bottom of the screen, and the other layer has connection points at the left and right of the screen. When the screen is touched, the two layers make contact and electricity can flow between them.

The horizontal position of a touch can be found by applying positive voltage to the right side of the horizontal layer and negative voltage to to the left side. When not driving the top and bottom of the vertical layer, the voltage potential on that layer will be proportional to the horizontal distance across the screen of the press, which can be measured with an ADC channel. By reversing these connections, the vertical position can also be measured. When the screen is not being touched, there will be no voltage on the non-powered layer.

By monitoring the voltage on each layer when the other layer is appropriately driven, touches and releases on the screen, as well as movements of the touch, can be detected and reported.

In order to read the current voltage on the two layers and also drive the appropriate voltages onto the layers, each side of each layer is connected to both a GPIO and an ADC channel. The GPIO is used to drive the node to a particular voltage, and when the GPIO is configured as an input, the corresponding ADC channel can be used to read the layer's voltage.

The touch screen is sampled every 2.5 ms, with four samples required to properly read both the X and Y position. Therefore, 100 X/Y sample pairs are captured every second.

Like the display driver, the touch screen driver operates in the same four orientations (selected in the same manner). Default calibrations are provided for using the touch screen in each orientation; the calibrate application can be used to determine new calibration values if necessary.

The touch screen driver utilizes sample sequence 3 of ADC0 and timer 5 subtimer B. The interrupt from the ADC0 sample sequence 3 is used to process the touch screen readings; the TouchScreenIntHandler() function should be called when this interrupt occurs (which is typically accomplished by placing it in the vector table in the startup code for the application).

The touch screen driver makes use of calibration parameters determined using the "calibrate" example application. The theory behind these parameters is explained by Carlos E. Videles in the June 2002 issue of Embedded Systems Design. It can be found online at `http://www.embedded.com/story/OEG20020529S0046`.

This driver is located in `examples/boards/dk-tm4c129x-boostxl-senshub/drivers`, with `touch.c` containing the source code and `touch.h` containing the API declarations for use by applications.

8.2 API Functions

8.3 Programming Example

The following example shows how to initialize the touchscreen driver and the callback function which receives notifications of touch and release events in cases where the StellarisWare Graphics Library widget manager is not being used by the application.

```
//
// The touch screen driver calls this function to report all state changes.
//
static long
TouchTestCallback(uint32_t ui32Message, int32_t i32X, int32_t i32Y)
{
    //
    // Check the message to determine what to do.
    //
    switch(ui32Message)
    {
        //
        // The screen is no longer being touched (in other words, pen/pointer
        // up).
        //
        case WIDGET_MSG_PTR_UP:
        {
            //
            // Handle the pointer up message if required.
            //
            break;
        }

        //
        // The screen has just been touched (in other words, pen/pointer down).
        //
        case WIDGET_MSG_PTR_DOWN:
        {
            //
            // Handle the pointer down message if required.
            //
            break;
        }

        //
        // The location of the touch on the screen has moved (in other words,
        // the pen/pointer has moved).
        //
        case WIDGET_MSG_PTR_MOVE:
        {
            //
            // Handle the pointer move message if required.
            //
            break;
        }

        //
        // An unknown message was received.
        //
        default:
        {
            //
            // Ignore all unknown messages.
            //
```

```
            break;
        }
    }

    //
    // Success.
    //
    return(0);
}

//
// The first touch screen example.
//
void
TouchScreenExample1(void)
{
    uint32_t ui32SysClock;

    //
    // Initialize the touch screen driver.  This code assumes that ui32SysClock
    // has been set to the clock frequency of the device (for example, the
    // value returned by SysCtlClockFreqSet).
    //
    TouchScreenInit(ui32SysClock);

    //
    // Register the application callback function that is to receive touch
    // screen messages.
    //
    TouchScreenCallbackSet(TouchTestCallback);
}
```

If using the StellarisWare Graphics Library widget manager, touchscreen initialization code is as follows. In this case, the touchscreen callback is provided within the widget manager so no additional function is required in the application code.

```
//
// The second touch screen example.
//
void
TouchScreenExample2(void)
{
    uint32_t ui32SysClock;

    //
    // Initialize the touch screen driver.  This code assumes that ui32SysClock
    // has been set to the clock frequency of the device (for example, the
    // value returned by SysCtlClockFreqSet).
    //
    TouchScreenInit(ui32SysClock);

    //
    // Register the graphics library pointer message callback function so that
    // it receives touch screen messages.
    //
    TouchScreenCallbackSet(WidgetPointerMessage);
}
```

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as "components") are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

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

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

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

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

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

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI's goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or "enhanced plastic" are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

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

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