EK-TM4C1294XL-BOOSTXL-SENSHUB Firmware Development Package

USER'S GUIDE

SW-EK-TM4C1294XL-BOOSTXL-SENSHUB-UG-2.1.4.178

Copyright

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

Revision Information

This is version 2.1.4.178 of this document, last updated on February 22, 2017.

Table of Contents

1 Introduction

The Texas Instruments® Tiva™ EK-TM4C1294XL-BOOSTXL-SENSHUB evaluation board (Tiva C Series TM4C1294 Connected LaunchPad) is a low cost platform that can be used for software development and prototyping a hardware design. A variety of BoosterPacks are available to quickly extend the LaunchPad's features.

The EK-TM4C1294XL includes a Tiva ARM® Cortex™-M4-based microcontroller and the following features:

- Tiva™ TM4C1294NCPDT microcontroller
- Ethernet connector
- USB OTG connector
- 2 user buttons
- 4 User LEDs
- 2 BoosterPack XL sites
- On-board In-Circuit Debug Interface (ICDI)
- Power supply option from USB ICDI connection, USB OTG connection or external power connection
- Shunt jumper for microcontroller current consumption measurement

This document describes the example applications that are provided for the EK-TM4C1294XL when paired with the BOOSTXL-SENSHUB BoosterPack. This BoosterPack provides a variety of motion and environmental sensors. It also provides an EM expansion option for attachement of additional peripherals such as the CC2533EMK or CC4000EMK. These examples utilize the TivaWare™ for C Series Sensor Library to extract and process information from the BOOSTXL-SENSHUB.

2 Example Applications

The example applications show how to utilize features of this evaluation 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 evaluation board.

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

There is an IAR workspace file (`ek-tm4c1294xl-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

There is a Keil multi-project workspace file (`ek-tm4c1294xl-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/ek-tm4c1294xl-boostxl-senshub` subdirectory of the firmware development package source distribution.

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

This example demonstrates the basic use of the Sensor Library, TM4C1294 LaunchPad 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 a complimentary-filtered direct cosine matrix (DCM) algorithm. The algorithm is provided as part of the Sensor Library.

This example requires that the BOOSTXL-SENSHUB be installed on BoosterPack 1 interface headers. See code comments for instructions on how to use BoosterPack 2 interface.

Connect a serial terminal program to the LaunchPad's ICDI virtual serial port at 115,200 baud. Use eight bits per byte, no parity and one stop bit. The raw sensor measurements, Euler angles and quaternions are printed to the terminal. An LED begins to blink at 1Hz after initialization is completed and the example application is running.

2.2 Humidity Measurement with the SHT21 (humidity_sht21)

This example demonstrates the basic use of the Sensoror Library, TM4C1294XL LaunchPad and SensHub BoosterPack to obtain temperature and relative humidity of the environment using the Sensirion SHT21 sensor.

This example requires that the SensHub BoosterPack is installed on BoosterPack 1 interface headers on the LaunchPad. See the code comments for information on porting this to use BoosterPack 2.

Connect a serial terminal program to the LaunchPad's ICDI virtual serial port at 115,200 baud. Use eight bits per byte, no parity and one stop bit. The humidity and temperature as measured by the SHT21 is printed to the terminal. An LED will blink to indicate the application is running.

2.3 Light Measurement with the ISL29023 (light_isl29023)

This example demonstrates the basic use of the Sensor Library, TM4C1294 Connected LaunchPad and the SensHub BoosterPack to obtain ambient and infrared light measurements with the ISL29023 sensor.

The SensHub BoosterPack must be installed on BoosterPack 1 interface. See code comments for changes needed to use BoosterPack 2 interface.

Connect a serial terminal program to the LaunchPad's ICDI virtual serial port at 115,200 baud. Use eight bits per byte, no parity and one stop bit. The raw sensor measurements are printed to the terminal. An LED blinks at 1Hz once the initialization is complete and the example is running.

The code automatically adjusts the dynamic range of the sensor when the intensity reaches a min or max threshold within the current range setting.

2.4 Pressure Measurement with the BMP180 (pressure_bmp180)

This example demonstrates the basic use of the Sensor Library, the EK-TM4C1294XL LaunchPad, and the SensHub BoosterPack to obtain air pressure and temperature measurements with the BMP180 sensor.

SensHub BoosterPack (BOOSTXL-SENSHUB) must be installed on BoosterPack 1 interface headers.

Instructions for use of SensorHub on BoosterPack 2 headers are in the code comments.

Connect a serial terminal program to the LaunchPad's ICDI virtual serial port at 115,200 baud. Use eight bits per byte, no parity and one stop bit. The raw sensor measurements are printed to the terminal. The LED blinks at 1 Hz once the initialization is complete and the example is running.

2.5 SensHub Internet of Things Example (senshub_iot)

This application uses FreeRTOS to manage multiple sensor tasks and aggregate sensor data to be published to a cloud server. The senshub_iot.c file contains the main function and perform task init before handing control over to the FreeRTOS scheduler.

The tasks and their responsibilities are as follows:

- cloud_task.c is a manager of the cloud interface. It gathers the sensor data and builds it into a packet for transmission to the cloud.

- command_task.c is a manager of the UART virtual com port connection to a local PC. This interface allows advanced commands and data.

- isl29023_task.c is a task to manage the interface to the isl29023 light sensor. It collects data from the sensor and makes it available to other tasks.
- tmp006_task.c is a task that manages the tmp006 temperature sensor. It gathers data from the temperature sensor and makes it available to other tasks.
- bmp180_task.c is a task that manages the bmp180 pressure sensor. It gathers data from the sensor and makes it available to other tasks.
- compdcm_task.c is a task that manages data from the MPU9150. It performs complimentary direct cosine matrix filter on the data to determine roll, pitch and yaw as well as quaternions. This data is made available to other tasks.
- sht21_task.c is a task that manages the SHT21 humidity and temperature sensor. It collects data from the sensor and makes it available to other tasks.

In addition to the tasks, this application also uses the following FreeRTOS resources:

- Queues enable information transfer between tasks.
- Mutex Semaphores guard resources such as the UART from access by multiple tasks at the same time.
- Binary Semaphores synchronize events between interrupts and task contexts.
- A FreeRTOS Delay to put the tasks in blocked state when they have nothing to do.
- A Software timer to regulate the timing of cloud sync events.
- The FreeRTOS run time stats feature to show CPU usage of each task at run time.

For additional details on FreeRTOS, refer to the FreeRTOS web page at: `http://www.freertos.org/`

2.6 Temperature Measurement with the TMP006 (temperature_tmp006)

This example demonstrates the basic use of the Sensor Library, TM4C1294 Connected LaunchPad and the SensHub BoosterPack to obtain ambient and object temperature measurements with the Texas Instruments TMP006 sensor.

SensHub BoosterPack (BOOSTXL-SENSHUB) Must be installed on BoosterPack 1 interface headers.

Connect a serial terminal program to the LaunchPad's ICDI virtual serial port at 115,200 baud. Use eight bits per byte, no parity and one stop bit. The raw sensor measurements are printed to the terminal. An LED blinks at 1Hz once the initialization is complete and the example is running.

3 Buttons Driver

3.1 Introduction

The buttons driver provides functions to make it easy to use the push buttons on this evaluation board. The driver provides a function to initialize all the hardware required for the buttons, and features for debouncing and querying the button state.

This driver is located in `examples/boards/ek-tm4c1294xl-boostxl-senshub/drivers`, with `buttons.c` containing the source code and `buttons.h` containing the API declarations for use by applications.

3.2 API Functions

Functions

- void ButtonsInit (void)
- uint8_t ButtonsPoll (uint8_t *pui8Delta, uint8_t *pui8RawState)

3.2.1 Function Documentation

3.2.1.1 ButtonsInit

Initializes the GPIO pins used by the board pushbuttons.

Prototype:

```
void
ButtonsInit(void)
```

Description:

This function must be called during application initialization to configure the GPIO pins to which the pushbuttons are attached. It enables the port used by the buttons and configures each button GPIO as an input with a weak pull-up.

Returns:

None.

3.2.1.2 ButtonsPoll

Polls the current state of the buttons and determines which have changed.

Prototype:

```
uint8_t
ButtonsPoll(uint8_t *pui8Delta,
            uint8_t *pui8RawState)
```

Parameters:

pui8Delta points to a character that will be written to indicate which button states changed since the last time this function was called. This value is derived from the debounced state of the buttons.

pui8RawState points to a location where the raw button state will be stored.

Description:

This function should be called periodically by the application to poll the pushbuttons. It determines both the current debounced state of the buttons and also which buttons have changed state since the last time the function was called.

In order for button debouncing to work properly, this function should be called at a regular interval, even if the state of the buttons is not needed that often.

If button debouncing is not required, the the caller can pass a pointer for the *pui8RawState* parameter in order to get the raw state of the buttons. The value returned in *pui8RawState* will be a bit mask where a 1 indicates the buttons is pressed.

Returns:

Returns the current debounced state of the buttons where a 1 in the button ID's position indicates that the button is pressed and a 0 indicates that it is released.

3.3 Programming Example

The following example shows how to use the buttons driver to initialize the buttons, debounce and read the buttons state.

```
//
// Map Left button to the GPIO Pin 0 of the button port.
//
#define LEFT_BUTTON             GPIO_PIN_0

//
// The button example
//
void
ButtonExample(void)
{
    unsigned char ucDelta, ucState;

    //
    // Initialize the buttons.
    //
    ButtonsInit();

    //
    // From timed processing loop (for example every 10 ms)
    //
    {
        //
        // Poll the buttons.  When called periodically this function will
        // run the button debouncing algorithm.
```

```
        //
        ucState = ButtonsPoll(&ucDelta, 0);

        //
        // Test to see if the SELECT button was pressed and do something
        //
        if(BUTTON_PRESSED(LEFT_BUTTON, ucState, ucDelta))
        {
            //
            // TODO: SELECT button action code
            //
        }
    }
}
```

4 Pinout Module

4.1 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/ek-tm4c1294xl-boostxl-senshub/drivers`, with `pinout.c` containing the source code and `pinout.h` containing the API declarations for use by applications.

4.2 API Functions

Functions

- void LEDRead (uint32_t *pui32LEDValue)
- void LEDWrite (uint32_t ui32LEDMask, uint32_t ui32LEDValue)
- void PinoutSet (bool bEthernet, bool bUSB)

4.2.1 Function Documentation

4.2.1.1 LEDRead

This function reads the state to the LED bank.

Prototype:

```
void
LEDRead(uint32_t *pui32LEDValue)
```

Parameters:

pui32LEDValue is a pointer to where the LED value will be stored.

Description:

This function reads the state of the CLP LEDs and stores that state information into the variable pointed to by pui32LEDValue.

Returns:

None.

4.2.1.2 LEDWrite

This function writes a state to the LED bank.

Prototype:

```
void
LEDWrite(uint32_t ui32LEDMask,
         uint32_t ui32LEDValue)
```

Parameters:

ui32LEDMask is a bit mask for which GPIO should be changed by this call.

ui32LEDValue is the new value to be applied to the LEDs after the ui32LEDMask is applied.

Description:

The first parameter acts as a mask. Only bits in the mask that are set will correspond to LEDs that may change. LEDs with a mask that is not set will not change. This works the same as GPIOPinWrite. After applying the mask the setting for each unmasked LED is written to the corresponding LED port pin via GPIOPinWrite.

Returns:

None.

4.2.1.3 PinoutSet

Configures the device pins for the standard usages on the EK-TM4C1294XL.

Prototype:

```
void
PinoutSet(bool bEthernet,
          bool bUSB)
```

Parameters:

bEthernet is a boolean used to determine function of Ethernet pins. If true Ethernet pins are configured as Ethernet LEDs. If false GPIO are available for application use.

bUSB is a boolean used to determine function of USB pins. If true USB pins are configured for USB use. If false then USB pins are available for application use as GPIO.

Description:

This function enables the GPIO modules and configures the device pins for the default, standard usages on the EK-TM4C1294XL. 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.

4.3 Programming Example

The following example shows how to configure the device pins.

```
//
// The pinout example.
//
void
PinoutExample(void)
{
    //
    // Configure the device pins.
    // First argument determines whether the Ethernet pins will be configured
    // in networking mode for this application.
    // Second argument determines whether the USB pins will be configured for
    // USB mode for this application.
    //
    PinoutSet(true, false);
}
```

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