EK-TM4C1294XL-BOOSTXL-KENTEC-S1 Firmware Development Package

USER'S GUIDE

SW-EK-TM4C1294XL-BOOSTXL-KENTEC-S1-UG-2.1.3.156

Copyright

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

Revision Information

This is version 2.1.3.156 of this document, last updated on July 25, 2016.

Table of Contents

1 Introduction

The Texas Instruments® Tiva™ EK-TM4C1294XL-BOOSTXL-KENTEC-S1 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-K350QVG-S1 BoosterPack. This BoosterPack provides a QVGA (320 x 240) display. These examples utilize the TivaWare™ for C Series Graphics Library to create a rich graphical user interface with widgets and multi-language support.

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-kentec-s1.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-kentec-s1.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-kentec-s1` subdirectory of the firmware development package source distribution.

2.1 Font Viewer (fontview)

This example displays the contents of a TivaWare Graphics Library font on the board's LCD touch-screen. By default, the application shows a test font containing ASCII, the Japanese Hiragana and Katakana alphabets, and a group of Korean Hangul characters.

The LCD BoosterPack (BOOSTXL-K350QVG-S1) should be installed on BoosterPack 2 interface headers.

2.2 Graphics Library Demonstration (grlib_demo)

This application provides a demonstration of the capabilities of the TivaWare Graphics Library. A series of panels show different features of the library. For each panel, the bottom provides a forward and back button (when appropriate), along with a brief description of the contents of the panel.

The first panel provides some introductory text and basic instructions for operation of the application.

The second panel shows the available drawing primitives: lines, circles, rectangles, strings, and images.

The third panel shows the canvas widget, which provides a general drawing surface within the widget hierarchy. A text, image, and application-drawn canvas are displayed.

The fourth panel shows the check box widget, which provides a means of toggling the state of an item. Four check boxes are provided, with each having a red "LED" to the right. The state of the LED tracks the state of the check box via an application callback.

The fifth panel shows the container widget, which provides a grouping construct typically used for radio buttons. Containers with a title, a centered title, and no title are displayed.

The sixth panel shows the push button widget. Two columns of push buttons are provided; the appearance of each column is the same but the left column does not utilize auto-repeat while the

right column does. Each push button has a red “LED” to its left, which is toggled via an application callback each time the push button is pressed.

The seventh panel shows the radio button widget. Two groups of radio buttons are displayed, the first using text and the second using images for the selection value. Each radio button has a red “LED” to its right, which tracks the selection state of the radio buttons via an application callback. Only one radio button from each group can be selected at a time, though the radio buttons in each group operate independently.

The eighth and final panel shows the slider widget. Six sliders constructed using the various supported style options are shown. The slider value callback is used to update two widgets to reflect the values reported by sliders. A canvas widget near the top right of the display tracks the value of the red and green image-based slider to its left and the text of the grey slider on the left side of the panel is update to show its own value. The slider on the right is configured as an indicator which tracks the state of the upper slider and ignores user input.

The LCD BoosterPack (BOOSTXL-K350QVG-S1) should be installed on BoosterPack 2 interface headers.

2.3 Graphics Library String Table Demonstration (lang_demo)

This application provides a demonstration of the capabilities of the TivaWare Graphics Library’s string table functions. Two panels show different implementations of features of the string table functions. For each panel, the bottom provides a forward and back button (when appropriate).

The first panel provides a large string with introductory text and basic instructions for operation of the application.

The second panel shows the available languages and allows them to be switched between English, German, Spanish and Italian.

The string table and custom fonts used by this application can be found under /third_party/fonts/lang_demo. The original strings that the application intends displaying are found in the language.csv file (encoded in UTF8 format to allow accented characters and Asian language ideographs to be included. The mkstringtable tool is used to generate two versions of the string table, one which remains encoded in UTF8 format and the other which has been remapped to a custom codepage allowing the table to be reduced in size compared to the original UTF8 text. The tool also produces character map files listing each character used in the string table. These are then provided as input to the ftrasterize tool which generates two custom fonts for the application, one indexed using Unicode and a smaller one indexed using the custom codepage generated for this string table.

The command line parameters required for mkstringtable and ftrasterize can be found in the makefile in third_party/fonts/lang_demo.

By default, the application builds to use the custom codepage version of the string table and its matching custom font. To build using the UTF8 string table and Unicode-indexed custom font, ensure that the definition of **USE_REMAPPED_STRINGS** at the top of the lang_demo.c source file is commented out.

The LCD BoosterPack (BOOSTXL-K350QVG-S1) should be installed on BoosterPack 2 interface headers.

2.4 Scribble Pad (scribble)

The scribble pad provides a drawing area on the screen. Touching the screen will draw onto the drawing area using a selection of fundamental colors (in other words, the seven colors produced by the three color channels being either fully on or fully off). Each time the screen is touched to start a new drawing, the drawing area is erased and the next color is selected.

The LCD BoosterPack (BOOSTXL-K350QVG-S1) should be installed on BoosterPack 2 interface headers.

3 Kentec 320x240x16 Display Driver

3.1 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** (which corresponds to how the display is mounted to the BOOSTXL-K350QVG-S1 BoosterPack).

This driver is located in `examples/boards/ek-tm4c1294xl-boostxl-kentec-s1/drivers`, with `kentec320x240x16_ssd2119_spi.c` containing the source code and `kentec320x240x16_ssd2119_spi.h` containing the API declarations for use by applications.

3.2 API Functions

3.3 Programming Example

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

```
extern const tDisplay g_sKentec320x240x16_SSD2119;

//
// 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);
}
```

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-kentec-s1/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);
}
```

5 Touch Screen Driver

5.1 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 1 subtimer A. 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/ek-tm4c1294xl-boostxl-kentec-s1-kentec/drivers`, with `touch.c` containing the source code and `touch.h` containing the API declarations for use by applications.

5.2 API Functions

Functions

- void TouchScreenCallbackSet (int32_t (*pfnCallback)(uint32_t ui32Message, int32_t i32X, int32_t i32Y))
- void TouchScreenInit (uint32_t ui32SysClock)
- void TouchScreenIntHandler (void)

5.2.1 Function Documentation

5.2.1.1 TouchScreenCallbackSet

Sets the callback function for touch screen events.

Prototype:

```
void
TouchScreenCallbackSet(int32_t (*ui32Message,)(uint32_t int32_t i32X,
int32_t i32Y) pfnCallback)
```

Parameters:

pfnCallback is a pointer to the function to be called when touch screen events occur.

Description:

This function sets the address of the function to be called when touch screen events occur. The events that are recognized are the screen being touched (“pen down”), the touch position moving while the screen is touched (“pen move”), and the screen no longer being touched (“pen up”).

Returns:

None.

5.2.1.2 TouchScreenInit

Initializes the touch screen driver.

Prototype:

```
void
TouchScreenInit(uint32_t ui32SysClock)
```

Parameters:

ui32SysClock is the frequency of the system clock.

Description:

This function initializes the touch screen driver, beginning the process of reading from the touch screen. This driver uses the following hardware resources:

- ADC sample sequence 3
- Timer 1 subtimer A

Returns:
None.

5.2.1.3 TouchScreenIntHandler

Handles the ADC interrupt for the touch screen.

Prototype:

```
void
TouchScreenIntHandler(void)
```

Description:
This function is called when the ADC sequence that samples the touch screen has completed its acquisition. The touch screen state machine is advanced and the acquired ADC sample is processed appropriately.

It is the responsibility of the application using the touch screen driver to ensure that this function is installed in the interrupt vector table for the ADC3 interrupt.

Returns:
None.

5.3 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 TivaWare 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 TivaWare 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