EK-TM4C123GXL-BOOST-DLPTRF7970ABP Firmware Development Package

USER'S GUIDE

SW-EK-TM4C123GXL-BOOST-DLPTRF7970ABP-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™ C Series EK-TM4C123GXL-BOOST-DLPTRF7970ABP evaluation board is a low cost platform that can be used for software development and to prototype a hardware design. It contains a Tiva C Series ARM® Cortex™-M4F-based microcontroller, a USB device port, two push buttons, and a RGB LED that can be used to exercise the peripherals on the microcontroller. Additionally, most of the microcontroller's pins are brought to headers, allowing for easy connection to other hardware for the purposes of prototyping. The outer rows of header pins are compatible with the MSP430™ Launchpad.

This document describes the example applications that are provided for this evaluation board.

2 Example Applications

The example applications show how to use features of the Cortex-M4F microprocessor, the peripherals on the Tiva C Series microcontroller, and the drivers provided by the peripheral driver library. These applications are intended for demonstration and as a starting point for new applications.

There is an IAR workspace file (`ek-tm4c123gxl-boost-dlptrf7970abp.eww`) that contains the peripheral driver library project, USB library project, and all of the board example projects, in a single, easy to use workspace for use with Embedded Workbench version 6.

There is a Keil multi-project workspace file (`ek-tm4c123gxl-boost-dlptrf7970abp.mpw`) that contains the peripheral driver library project, USB library project, and 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-tm4c123gxl-boost-dlptrf7970abp` subdirectory of the firmware development package source distribution.

2.1 NFC P2P Demo (nfc_p2p_demo)

This example application demonstrates the operation of the Tiva C Series evaluation kit with the TRF7970ABP BoosterPack as a NFC P2P device.

The application supports reading and writing Text, URI, and SmartPoster Tags. The application gets a raw message buffer from the TRF79x0 stack, decodes the information to recognized tag types, then re-encodes the data to a buffer to be sent back out. Pressing switch SW1 sends a URI message with a link to the Tiva C series Launchpad website. Pressing switch SW2 echo's back the last tag recieved. If no tag has been recieved then this button does nothing. Full debug information is given across the UART0 channel to aid in NFC P2P development.

For more information on NFC please see the full NFC specification list at `http://www.nfc-forum.org/specs/spec_list/`.

3 Buttons Driver

3.1 Introduction

The buttons driver provides functions to make it easy to use the push buttons on the EK-TM4C123GXL 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-tm4c123gxl-boost-dlptrf7970abp/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 caled 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.

```
//
// 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(SELECT_BUTTON, ucState, ucDelta))
    {
        ...
        // SELECT button action
    }
}
```

4 RGB LED Driver

4.1 Introduction

The RGB LED driver provides a simple interface to control the RGB LED on the EK-TM4C123GXL. The driver provides a function to initialize the timers and GPIO for the RGB. It also provides features for controlling the color and intensity of the LED.

This driver is located in `examples/boards/ek-tm4c123gxl-boost-dlptrf7970abp/drivers`, with `rgb.c` containing the source code and `rgb.h` containing the API declarations for use by applications.

4.2 API Functions

Functions

- void RGBBlinkIntHandler (void)
- void RGBBlinkRateSet (float fRate)
- void RGBColorGet (uint32_t *pui32RGBColor)
- void RGBColorSet (volatile uint32_t *pui32RGBColor)
- void RGBDisable (void)
- void RGBEnable (void)
- void RGBInit (uint32_t ui32Enable)
- void RGBIntensitySet (float fIntensity)
- void RGBSet (volatile uint32_t *pui32RGBColor, float fIntensity)

4.2.1 Function Documentation

4.2.1.1 RGBBlinkIntHandler

Wide Timer interrupt to handle blinking effect of the RGB

Prototype:

```
void
RGBBlinkIntHandler(void)
```

Description:

This function is called by the hardware interrupt controller on a timeout of the wide timer. This function must be in the NVIC table in the startup file. When called will toggle the enable flag to turn on or off the entire RGB unit. This creates a blinking effect. A wide timer is used since the blink is intended to be visible to the human eye and thus is expected to have a frequency between 15 and 0.1 hz. Currently blink duty is fixed at 50%.

Returns:
None.

4.2.1.2 RGBBlinkRateSet

Sets the blink rate of the RGB Led

Prototype:
```
void
RGBBlinkRateSet(float fRate)
```

Parameters:
fRate is the blink rate in hertz.

Description:
This function controls the blink rate of the RGB LED in auto blink mode. to enable blinking pass a non-zero floating pointer number. To disable pass 0.0f as the argument. Calling this function will override the current RGBDisable or RGBEnable status.

Returns:
None.

4.2.1.3 RGBColorGet

Get the output color.

Prototype:
```
void
RGBColorGet(uint32_t *pui32RGBColor)
```

Parameters:
pui32RGBColor points to a three element array representing the relative intensity of each color. Red is element 0, Green is element 1, Blue is element 2. 0x0000 is off. 0xFFFF is fully on. Caller must allocate and pass a pointer to a three element array of uint32_ts.

Description:
This function should be called by the application to get the current color of the RGB LED.

Returns:
None.

4.2.1.4 RGBColorSet

Set the output color.

Prototype:
```
void
RGBColorSet(volatile uint32_t *pui32RGBColor)
```

Parameters:

pui32RGBColor points to a three element array representing the relative intensity of each color. Red is element 0, Green is element 1, Blue is element 2. 0x0000 is off. 0xFFFF is fully on.

Description:

This function should be called by the application to set the color of the RGB LED.

Returns:

None.

4.2.1.5 RGBDisable

Disable the RGB LED by configuring the GPIO's as inputs.

Prototype:

```
void
RGBDisable(void)
```

Description:

This function or RGBEnable should be called during application initialization to configure the GPIO pins to which the LEDs are attached. This function disables the timers and configures the GPIO pins as inputs for minimum current draw.

Returns:

None.

4.2.1.6 RGBEnable

Enable the RGB LED with already configured timer settings

Prototype:

```
void
RGBEnable(void)
```

Description:

This function or RGBDisable should be called during application initialization to configure the GPIO pins to which the LEDs are attached. This function enables the timers and configures the GPIO pins as timer outputs.

Returns:

None.

4.2.1.7 RGBInit

Initializes the Timer and GPIO functionality associated with the RGB LED

Prototype:

```
void
RGBInit(uint32_t ui32Enable)
```

Parameters:
ui32Enable enables RGB immediately if set.

Description:
This function must be called during application initialization to configure the GPIO pins to which the LEDs are attached. It enables the port used by the LEDs and configures each color's Timer. It optionally enables the RGB LED by configuring the GPIO pins and starting the timers.

Returns:
None.

4.2.1.8 RGBIntensitySet

Set the current output intensity.

Prototype:
```
void
RGBIntensitySet(float fIntensity)
```

Parameters:
fIntensity is used to scale the intensity of all three colors by the same amount. fIntensity should be between 0.0 and 1.0. This scale factor is applied individually to all three colors.

Description:
This function should be called by the application to set the intensity of the RGB LED.

Returns:
None.

4.2.1.9 RGBSet

Set the output color and intensity.

Prototype:
```
void
RGBSet(volatile uint32_t *pui32RGBColor,
       float fIntensity)
```

Parameters:
pui32RGBColor points to a three element array representing the relative intensity of each color. Red is element 0, Green is element 1, Blue is element 2. 0x0000 is off. 0xFFFF is fully on.
fIntensity is used to scale the intensity of all three colors by the same amount. fIntensity should be between 0.0 and 1.0. This scale factor is applied to all three colors.

Description:
This function should be called by the application to set the color and intensity of the RGB LED.

Returns:
None.

4.3 Programming Example

The following example shows how to use the rgb driver to initialize the RGB LED.

```
unsigned long ulColors[3];

//
// Initialize the rgb driver.
//
RGBInit(0);

//
// Set the intensity level from 0.0f to 1.0f
//
    RGBIntensitySet(0.3f);

//
// Initialize the three color values.
//
ulColors[BLUE]  = 0x00FF;
ulColors[RED]   = 0xFFFF;
ulColors[GREEN] = 0x0000;
RGBColorSet(ulColors);

//
// Enable the RGB.  This configure GPIOs to the Timer PWM mode needed
// to generate the color spectrum.
//
RGBEnable();


//
// Application may now call RGB API to suit program requirements.
//
...
```

IMPORTANT NOTICE

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