



# TC38x BSP example

## Quick Guide

# Table of Contents

<b>Terms &amp; Abbreviations</b>	1
<b>1. Introduction</b>	2
<b>2. Application perspective</b>	3
<b>3. Component view</b>	4
<b>4. Execution flow</b>	5
<b>5. Example Import &amp; Build</b>	6
<b>6. EVB Boards Configuration</b>	7
TRIBOARD_TC3X9_V2_0	8
TRIBOARD_TC3X9_V1_0	9
TRIBOARD_TC3X7_V2_0	10
TRIBOARD_TC3X7_V1_0	11
APPKIT_TC3X7_V2_0	12
<b>7. uC Derivative Configuration</b>	13
TC38x BSP configuration	14
<b>Document history</b>	15
<b>Disclaimer</b>	16

# Terms & Abbreviations

BSP

Board Startup Package

crt0

'C' run-time environment initialization code

EVB

Evaluation Board

uC

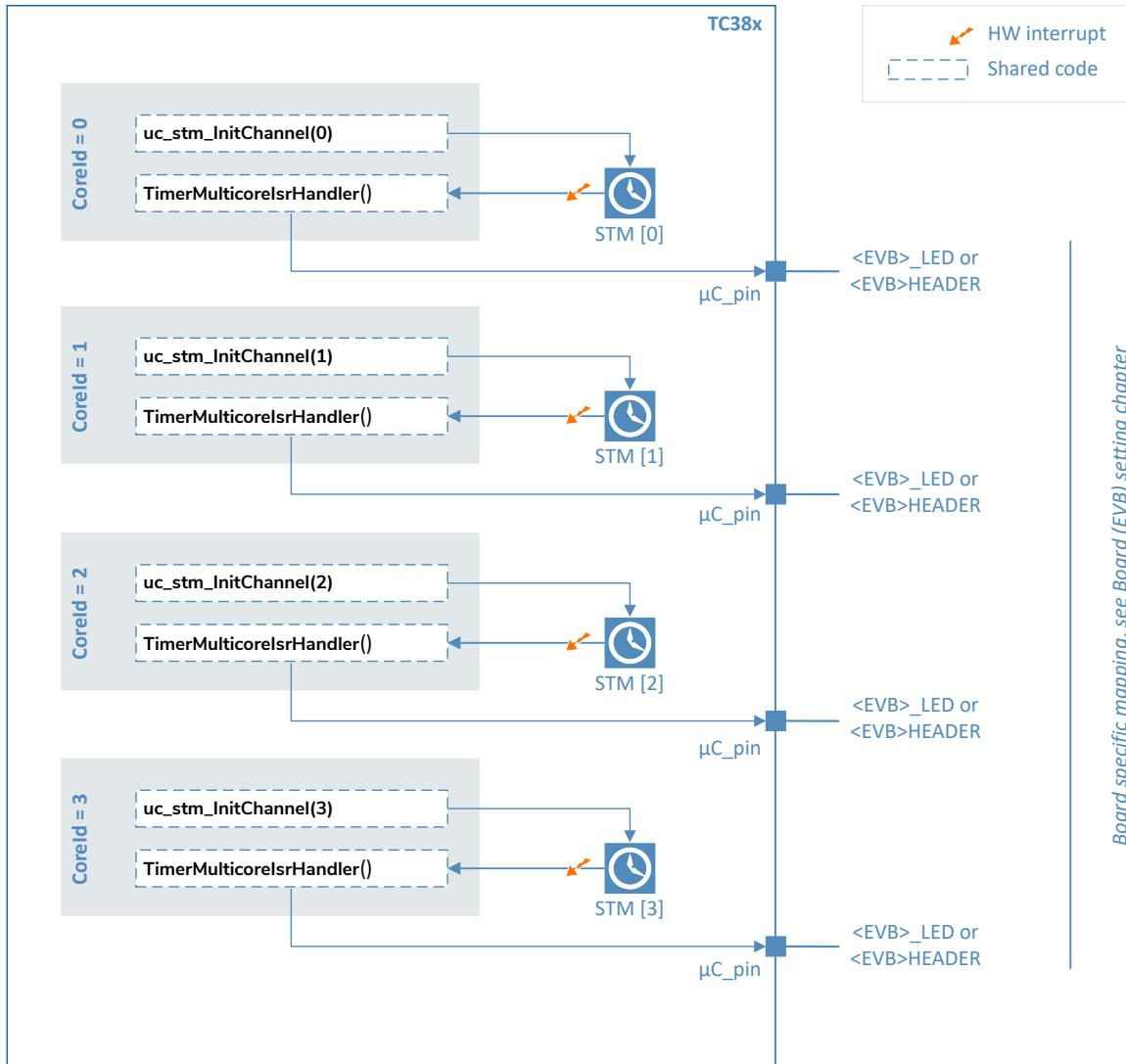
Microcontroller

# 1. Introduction

A tc38x\_bsp\_example is a small functional project designed to simplify an evaluation phase of the TC38x microcontroller architecture. It comes with necessary low-level functions like a startup code, minimalistic hardware abstraction, and predefined memory partitioning. On top of this low-level implementation, it provides a reference application code running on all uC cores.

## 2. Application perspective

From the user perspective, the application functionality consists of a simple LED blinking, where each active core toggles its LED through a dedicated uC pin. The initialization code sets the toggling period in multiples of 250ms according to the current Core Id.



Board specific mapping, see Board (EVB) setting chapter

Fig. 1. Application view of TC38x

### 3. Component view

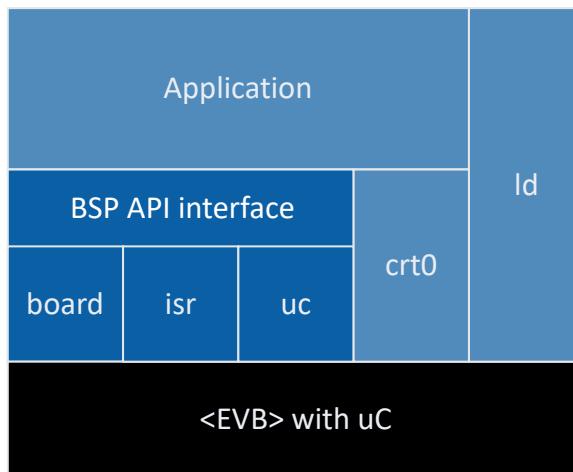


Fig. 2. Component view

#### Application component

A reference application implementing the functionality of the example utilizing provided low-level BSP routines consists of LED blinking by each core. The execution flow is the same on each core. The application folder (src) represents a playground where the user can extend the example functionality. It contains one file `shared_main.c` implemented as a shared code executed on each core.

#### BSP component

A BSP package represents toolchain, microcontroller, and evaluation board dependent SW tailored to a particular microcontroller derivative. It consists of three interconnected modules.

- `board`, an abstraction of an evaluation board functionality supported by BSP.
- `isr`, an implementation of an interrupt vector table to support a hardware interrupt functionality.
- `uc`, microcontroller dependent routines to simplify setup and control of uC hardware modules.

#### Startup component (crt0)

A reset startup and 'C' runtime initialization for AURIX architecture running from the reset till the user's main entry point, here `shared_main()`.

#### Linker component (ld)

Target uC linker file that prescribes placement of the BSP example code and data. It defines a subset of memory regions available on the uC.

## 4. Execution flow

The RESET core is responsible for the initialization of shared resources before enabling inactive cores. Once other core starts running, they can rely on a stable hardware platform. Each core handles its local resources individually.

The platform and core related setup is part of the Crt0PreInit and Crt0PostInit hooks, allowing to enter the user application with a defined state of the hardware platform.

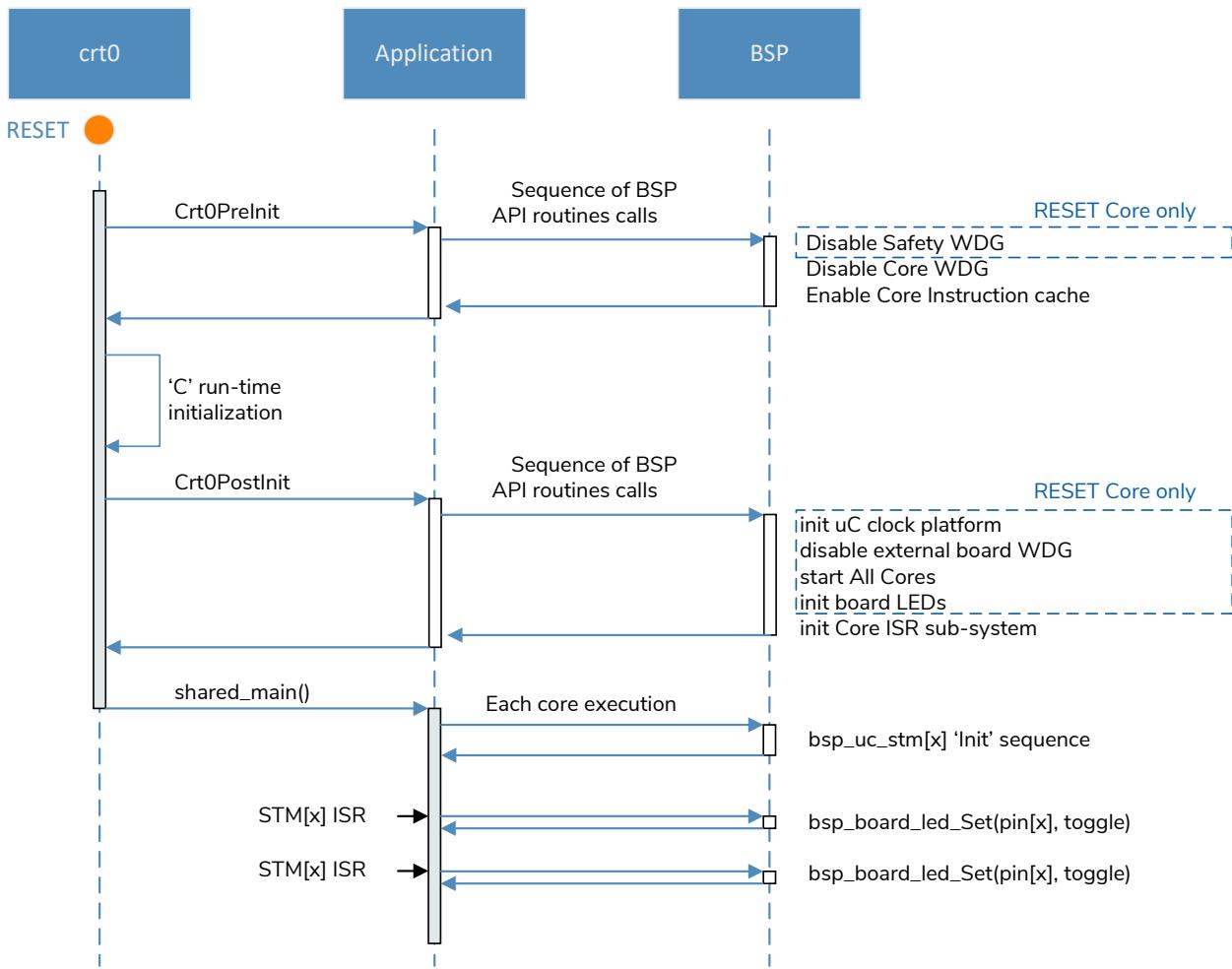


Fig. 3. Simplified individual core execution flow

Software parts of the example select the right execution path according to CoreID during the runtime. The runtime decision allows the user to execute core-specific behavior and still share the same code. Such a shared approach simplifies the microcontroller learning phase and first trials in the case of a multi-core platform.

## 5. Example Import & Build

The example project Import and Build is a straightforward process consisting of four steps described below.

Import the project to the IDE workspace

1. From the menu **File→Import→General** choose an option **Existing Projects into Workspace**
2. Browse for your project location
3. Select project
4. Click Finish

Activate the imported project

1. Select the imported project by Left-Click on it in the Project Explorer
2. From the menu **Project→Set Active Project** activate the project
3. The project name shall turn to bold

Select the right Build Configuration

1. From the menu **Project→Active Build Configuration→Build Configurations** choose the one matching the evaluation board in use
2. The project name shall correspond to the selected Build Configuration name

Build the project

1. Build the project from the menu **Project→Build Project**.
2. The output binary file (.elf) is located under the `_iROM_<BuildConfigurationName>` folder.

## 6. EVB Boards Configuration

The bsp/board folder provides several elements required by the BSP application

### BOARD\_XTAL\_CLOCK

A crystal oscillator frequency provided on the board. This clock is the input for the uC PLL setting in the bsp/uc section. + The user can set board specific clock frequency, but might happen that PLL parameter computation implemented in uc\_spec.h file does not find the correct values for such input. In such case, the user has to provide the hardware clock initialization.

### Board LED pinout

The BSP configures uC port-pin pairs connected to the on-board LED and provides corresponding enum type for the selection.

### External WDG disable routine

The EVB very often assembles Safe Power supply chip having watchdog capability. The user must handle the watchdog expiration in case it is active after the reset. + The BSP board support implements a routine to disable the external watchdog if it exists.

# TRIBOARD\_TC3X9\_V2\_0

```
/* XTAL Clock */
#ifndef BOARD_XTAL_CLOCK
#define BOARD_XTAL_CLOCK          20
#endif

/* pin configuration for LEDs */
const BOARD_LED_S board_led[BOARD_NB_LEDS] = {
    {33, 4}, /* LED_0 */
    {33, 5}, /* LED_1 */
    {33, 6}, /* LED_2 */
    {33, 7}, /* LED_3 */
    {20, 11}, /* LED_4 */
    {20, 12}, /* LED_5 */
    {20, 13}, /* LED_6 */
    {20, 14}, /* LED_7 */
};

/* External WDG = Present */
void bsp_board_wdg_Disable(void)
{
    TLF35584 chip External WDG disable sequence
}
```

# TRIBOARD\_TC3X9\_V1\_0

```
/* XTAL Clock */
#ifndef BOARD_XTAL_CLOCK
#define BOARD_XTAL_CLOCK          20
#endif

/* pin configuration for LEDs */
const BOARD_LED_S board_led[BOARD_NB_LEDS] = {
    {33, 4}, /* LED_0 */
    {33, 5}, /* LED_1 */
    {33, 6}, /* LED_2 */
    {33, 7}, /* LED_3 */
    {20, 11}, /* LED_4 */
    {20, 12}, /* LED_5 */
    {20, 13}, /* LED_6 */
    {20, 14}, /* LED_7 */
};

/* External WDG = Present */
void bsp_board_wdg_Disable(void)
{
    TLF35584 chip External WDG disable sequence
}
```

# TRIBOARD\_TC3X7\_V2\_0

```
/* XTAL Clock */
#ifndef BOARD_XTAL_CLOCK
#define BOARD_XTAL_CLOCK          20
#endif

/* pin configuration for LEDs */
const BOARD_LED_S board_led[BOARD_NB_LEDS] = {
    {33, 4}, /* LED_0 */
    {33, 5}, /* LED_1 */
    {33, 6}, /* LED_2 */
    {33, 7}, /* LED_3 */
    {20, 11}, /* LED_4 */
    {20, 12}, /* LED_5 */
    {20, 13}, /* LED_6 */
    {20, 14}, /* LED_7 */
};

/* External WDG = Present */
void bsp_board_wdg_Disable(void)
{
    TLF35584 chip External WDG disable sequence
}
```

# TRIBOARD\_TC3X7\_V1\_0

```
/* XTAL Clock */
#ifndef BOARD_XTAL_CLOCK
#define BOARD_XTAL_CLOCK          20
#endif

/* pin configuration for LEDs */
const BOARD_LED_S board_led[BOARD_NB_LEDS] = {
    {33, 4}, /* LED_0 */
    {33, 5}, /* LED_1 */
    {33, 6}, /* LED_2 */
    {33, 7}, /* LED_3 */
    {20, 11}, /* LED_4 */
    {20, 12}, /* LED_5 */
    {20, 13}, /* LED_6 */
    {20, 14}, /* LED_7 */
};

/* External WDG = Present */
void bsp_board_wdg_Disable(void)
{
    TLF35584 chip External WDG disable sequence
}
```

## APPKIT\_TC3X7\_V2\_0

```
/* XTAL Clock */
#ifndef BOARD_XTAL_CLOCK
#define BOARD_XTAL_CLOCK          20
#endif

/* pin configuration for LEDs */
const BOARD_LED_S board_led[BOARD_NB_LEDS] = {
    {13, 0}, /* LED_0 */
    {13, 1}, /* LED_1 */
    {13, 2}, /* LED_2 */
    {13, 3}, /* LED_3 */
    {33, 1}, /* GPIO P33.1 */
    {33, 2}, /* GPIO P33.2 */
    {33, 3}, /* GPIO P33.3 */
    {33, 4}, /* GPIO P33.4 */
};

/* External WDG = Present */
void bsp_board_wdg_Disable(void)
{
    TLF35584 chip External WDG disable sequence
}
```

## 7. uC Derivative Configuration

The bsp/uc folder contains files describing the targeted [uC](#) derivative in details necessary for a successful board startup.

They target following needs:

- microcontroller low-level HW access routines that are common across different derivatives, part of `bsp_uc.c`, `bsp_uc.h` files.
- an include of corresponding Infineon HW module peripheral header files allowing to access uC registers in 'C' friendly format.
- a definition of BSP API types, HW platform configuration values, and eventual implementation uC derivative specific version of low-level BSP routines to provide basic HW platform setup.

# TC38x BSP configuration

```

/* =====
 * uC step selection supported by BSP example
 *   TC38XA : step A
 * Default selection is the Step A
 * =====*/
#ifndef !defined(TC38XA)
#define TC38XA
#endif

/* =====
 * CORES SPECIFICATION
 * =====*/
/* Number of Cores instances */
#define UC_NB_CORES          4

/* RESET CORE: core that starts after Power On */
#define UC_RESET_CORE         0

/* =====
 * DEFAULT uC CLOCK SETTING
 * Target is MAX system clock for Core frequency following recommended rules.
 * User ** should not ** change it unless he is certain what he is doing
 * =====*/
/* System PLL */
#define UC_PLL0_CLOCK          300 /* MAX=300; SYS PLL output frequency */

/* Peripheral PLL */
#define UC_PLL1_CLOCK          320 /* MAX=320; PER PLL1 output frequency */
#define UC_PLL2_CLOCK          200 /* MAX=200; PER PLL2 output frequency */

/* Internal oscillator value [MHz] */
#define UC_BACKUP_CLOCK        100

/* Platform clocks */
#define UC_SRI_CLOCK            300 /* MAX=300; source = fPLL0 | fBACK */
#define UC_SPB_CLOCK            100 /* MAX=100; source = fPLL0 | fBACK */
#define UC_FSI_CLOCK             100 /* MAX=100; source = fSRI; */
#define UC_FSI2_CLOCK            300 /* MAX=300; source = fSRI; */

/* Peripheral clocks needed in BSP example */
#define UC_QSPI_CLOCK           200 /* MAX=200; source = fPLL1 | fPLL2 | fBACK */
#define UC_STM_CLOCK              100 /* MAX=100; source = fPLL0 | fBACK */

/* BSP supportive time macros */
#define UC_NB_TICKS_1US          (UC_STM_CLOCK)
#define UC_NB_TICKS_1MS          (UC_NB_TICKS_1US * 1000)

```

## Document history

Version	Date	Changes to the previous version
1.0	June 2020	Initial version.

# Disclaimer

## Please Read Carefully:

This document contains descriptions for copyrighted products that are not explicitly indicated as such. The absence of the TM symbol does not infer that a product is not protected. Additionally, registered patents and trademarks are similarly not expressly indicated in this document.

The information in this document has been carefully checked and is believed to be entirely reliable. However, HighTec EDV-Systeme GmbH assumes no responsibility for any inaccuracies. HighTec EDV-Systeme GmbH neither gives any guarantee nor accepts any liability whatsoever for consequential damages resulting from the use of this document or its associated product. HighTec EDV-Systeme GmbH reserves the right to alter the information contained herein without prior notification and accepts no responsibility for any damages that might result.

HighTec EDV-Systeme hereby disclaims any and all warranties and liabilities of any kind, including without limitation, warranties of non-infringement of intellectual property rights of any third party.

Rights - including those of translation, reprint, broadcast, photomechanical or similar reproduction and storage or processing in computer systems, in whole or in part - are reserved. No reproduction may occur without the express written consent from HighTec EDV-Systeme GmbH.

Copyright © 2022 HighTec EDV-Systeme GmbH, D-66113 Saarbrucken.



HighTec EDV-Systeme GmbH  
Europaallee 19, D-66113 Saarbrücken  
[info@hightec-rt.com](mailto:info@hightec-rt.com)  
+49-681-92613-16  
[www.hightec-rt.com](http://www.hightec-rt.com)