HIGHTEC

# TC234 PXROS-HR

BSP Example Quick Guide

Version 1.1, May 2019

# Table of Contents

# 1. Terms & Abbreviations

Message
: An PXROS-HR object to exchange data among tasks.

Event
: An PXROS-HR object to activate a task.

Periodic Event Object
: A periodic event PXROS-HR object (Pe) is a specialized version of a Delay object. A Task creating and starting the Pe object receives events at regular intervals.

BSP
: Board Suport Package

EVB
: Evaluation Board

CSA
: Context Save Area

PXUser0Privilege
: A task with this privilege has no direct access to peripherals, only via PXROS-HR system calls API.

PXUser1Privilege
: A task with this privilege posses direct access to peripherals without a need to switch to Supervisor mode through a system call.

# 2. Introduction

The PXROS-HR BSP example shows how to configure and build a simple project based on the PXROS-HR operating system. It introduces two main inter-task communication objects, Events and Messages, in their most basic configuration.

## 2.1. Supported Boards

1. APPKIT_TC2X4_V1.0

## 2.2. Development Tools

1. HighTec toolchain version 4.9.3.0 or higher
2. HighTec IDE version 2.2.x or higher
3. HighTec PXROS-HR operating system version 7.3.0
    - kernel
    - utilities

## 2.3. HW resources

Hardware resources used by this example:

| HW unit | Channel/output pin | Function |
|---|---|---|
| EVB | LED[0] | A notification of an Event reception in LedServer task (LED control) |
| EVB | LED[1] | A notification of a Message reception in LedServer task (LED control) |
| μC | STM[0] | PXROS-HR kernel tick base - core0 |

*Tab. 1. HW resources*

To know the physical mapping of LEDs to microcontroller IO pins, please see Board (EVB) configuration chapter.

# 3. Example overview

## 3.1. Application perspective

The example implements three user tasks, where two `LedClient` tasks ask a service from the third LedServer task to toggle a particular LED. Each client task utilizes a different communication mean to ask for it.

To achieve a periodic behavior of a running application each client task creates a periodic object and starts it with a predefined period.

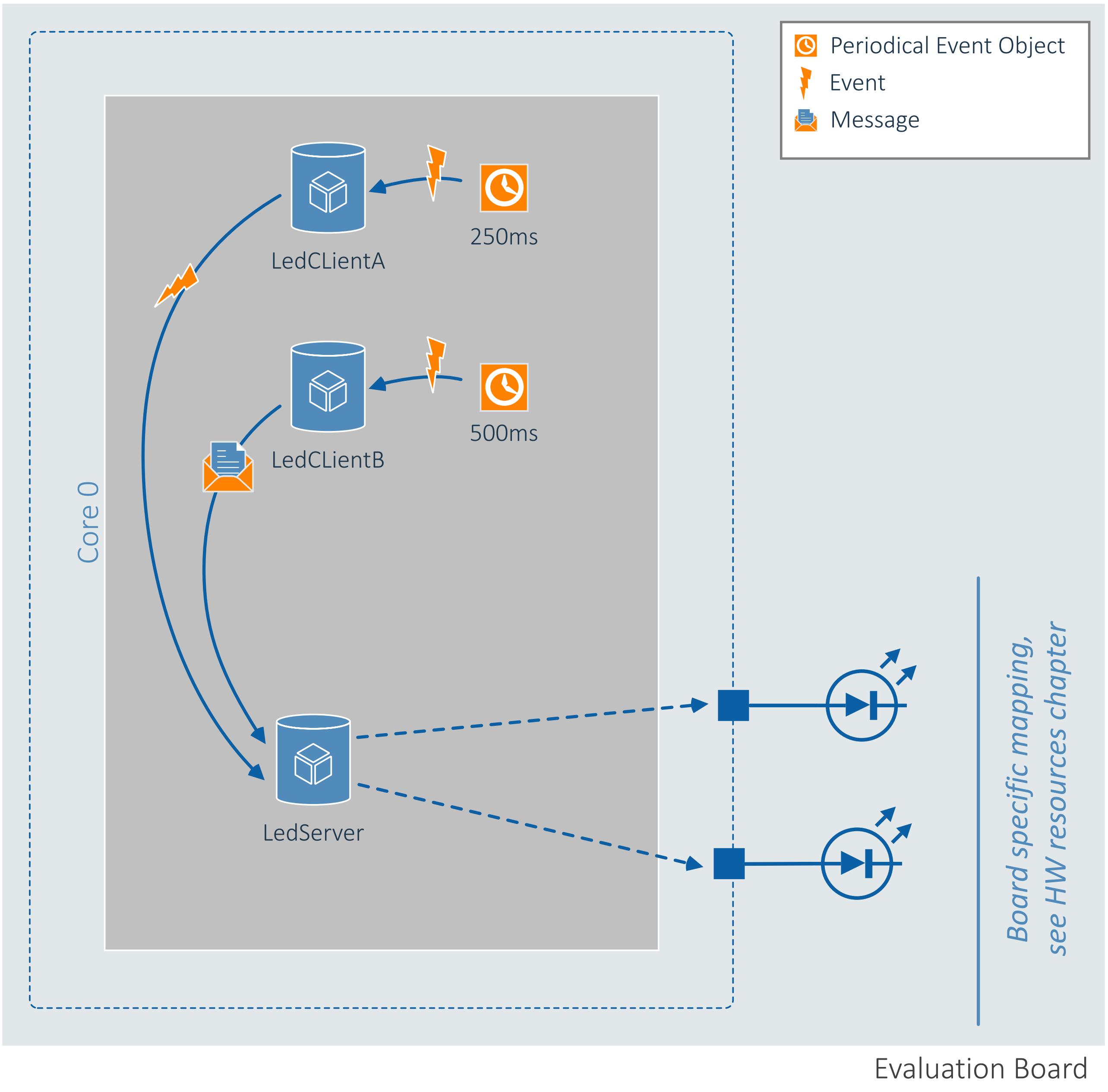


*Fig. 1. System block diagram*

## 3.2. Component view

The example consists of logically separated elements each having its folder within the project for easier navigation, bsp, src, and pxros.

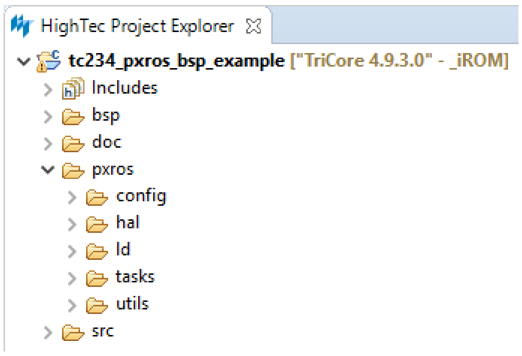


Fig. 2. Component view

bsp

A BSP component represents toolchain, microcontroller and evaluation board dependent SW tailored to a particular microcontroller derivative. For more details see BSP guide document.

doc

An example quick guide.

src

It contains one file `shared_main.c` implemented as a shared code executed on each core. PXROS-HR operating system starts here by execution of `PxInit` API function in the code branch belonging to the *MASTER_CORE*.

pxros

The `pxros` folder contains all application files related to the PXROS-HR operating system from its configuration, connection to underlying target, user task implementation to linker file description with the memory layout.

*config* - PXROS-HR system configuration. Here the user specifies parameters for each of the instantiated kernels, like the number of objects, tasks, and size of the default user memory.

*hal* - An abstraction of the underlying microcontroller for PXROS-HR time tick functionality.

*ld* - The folder contains linker files for the target memory layout and PXROS-HR system placement. Linker files are tightly coupled with the microcontroller architecture.

*tasks* - A folder with example tasks implementation.

*utils* - Common utilities used across user tasks.

## 3.3. User tasks

### 3.3.1. LedServer

The LedServer task acts as an independent GPIO peripheral driver. It is capable of toggling the state of any of the four GPIOs connected to AppKit on-board LEDs. The LedServer selects the LED either according to the Event number or based on the data value in the received message.

The LedServer get active for two reasons:

1. reception of an Event from LedClientA task
2. reception of a message from the LedClientB task

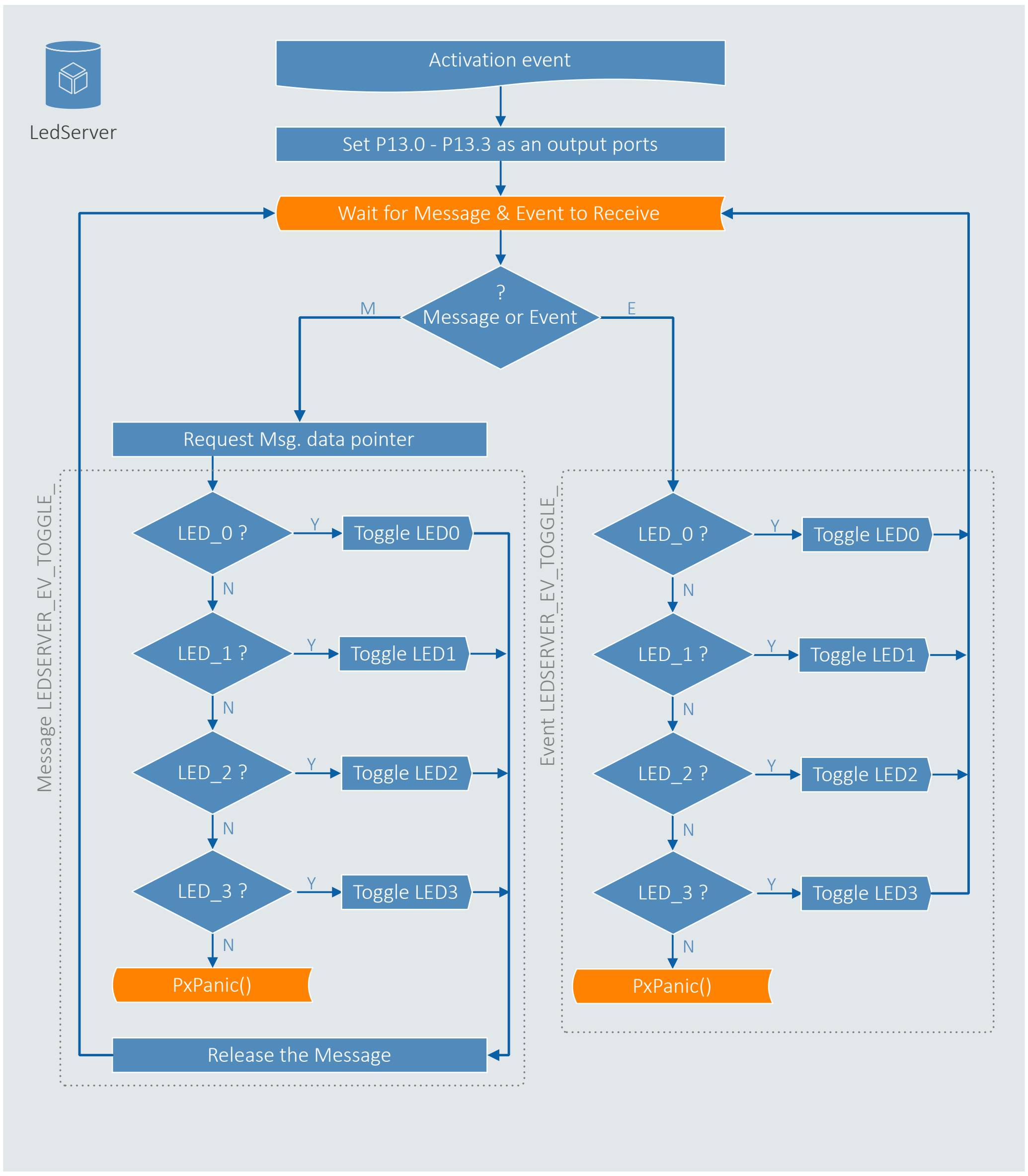

LedServer
Activation event
Set P13.0 - P13.3 as an output ports
Wait for Message & Event to Receive
? Message or Event
M
E
Request Msg. data pointer
Message LEDSERVER_EV_TOGGLE_
Event LEDSERVER_EV_TOGGLE_
LED_0 ?
Toggle LED0
LED_1 ?
Toggle LED1
LED_2 ?
Toggle LED2
LED_3 ?
Toggle LED3
Y
N
PxPanic()
Release the Message

*Fig. 3. Simplified LedServer task execution flow*

## 3.3.2. LedClientA

The task uses Periodic Event object to regularly wake up at 250ms intervals to send an Event to the LedServer task. The Event value encodes the LED.

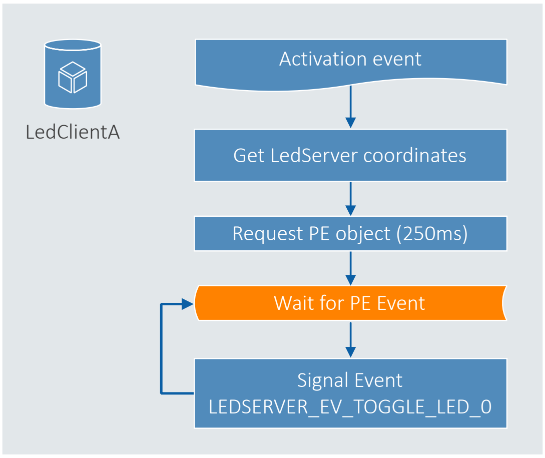

LedClientA
Activation event
Get LedServer coordinates
Request PE object (250ms)
Wait for PE Event
Signal Event LEDSERVER_EV_TOGGLE_LED_0

*Fig. 4. Simplified LedClientA tasks execution flow*

## 3.3.3. LedClientB

The LedClientB task is almost identical to LedClientA task. The task uses Periodic Event object to regularly wake up at 500ms intervals to send a Message to the LedServer task. The message data value encodes the LED.

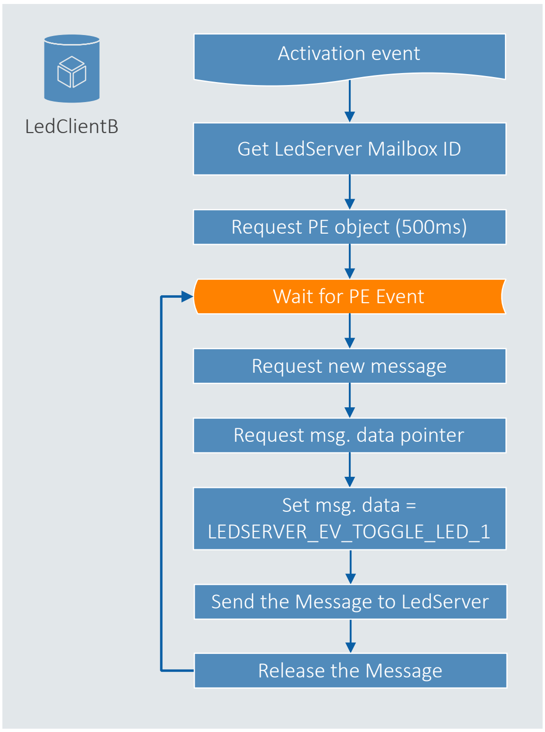

LedClientB
Activation event
Get LedServer Mailbox ID
Request PE object (500ms)
Wait for PE Event
Request new message
Request msg. data pointer
Set msg. data = LEDSERVER_EV_TOGGLE_LED_1
Send the Message to LedServer
Release the Message

*Fig. 5. Simplified LedClientB tasks execution flow*

## 3.4. Tasks to Core distribution

The task's execution core assignment is part of the InitTask function, located in the `initTask.c` file. The InitTask function shared across cores contains a run-time execution branch instantiating user tasks on each core independently.

In the current example, the task distribution in InitTask looks like in the following code snapshot.

*Code 1. Core task distribution*

```
void InitTask_Func(PxTask_t myID, PxMbx_t myMailbox, PxEvents_t myActivationEvents)
{
    ...
    /* ----------  CORE 0 TASKS  ---------- */
    case 0:
    {
        ...
        /* Create LedClientA */
        PxTask_t LedClientA_TaskId = LedClientA_Create (...);
        if (PxTaskIdError (LedClientA_TaskId) != PXERR_NOERROR) PxPanic();

        /* Create LedServer */
        PxTask_t LedServer_TaskId = LedServer_Create (...);
        if (PxTaskIdError (LedServer_TaskId) != PXERR_NOERROR) PxPanic();

        /* Register LedServer task ID in the Name Server (so other tasks may query it) */
        PxError_t regErr = PxNameRegister (...);
        if (regErr != PXERR_NOERROR) PxPanic();

        /* Activate Tasks event - starts after InitTask according to Task priority */
        PxTaskSignalEvents (LedServer_TaskId, LEDSERVER_ACTIVATION_EVENT);

        /* Create LedClientB */
        PxTask_t LedClientB_TaskId = LedClientB_Create (...);
        if (PxTaskIdError (LedClientB_TaskId) != PXERR_NOERROR) PxPanic();

        /* Activate Tasks event - starts after InitTask according to Task priority */
        PxTaskSignalEvents (LedClientB_TaskId, LEDCLIENTB_ACTIVATION_EVENT);

        /* Activate Tasks event - starts after InitTask according to Task priority */
        PxTaskSignalEvents (LedClientA_TaskId, LEDCLIENTA_ACTIVATION_EVENT);
    }
    ...
}
```

# 4. Example Import & Build

## 4.1. Importing the project

Follow these steps to import an example project to the HighTec IDE environment:

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

## 4.2. Building the project

Execute the following steps to build a final image:

1. Activate the project from the menu **Project → Set Active Project**.
2. Update `PXROS_HR_ROOT` variable in menu **Project → Properties → C/C++ Build → Environment** to point to the root of the HighTec PXROS-HR installation folder containing kernel, utility libraries, and interface header files (*`<PXROS_INSTALL_DIR>\pxros-hr\tricore\v7.3.0`*).
3. Select the build configuration matching your board from the menu **Project → Active Build Configuration**.
4. Build the project from the menu **Project → Build Project**.
5. The output binary file is located under the `_iROM_<EVB>` folder.

After each `Path Variable` and `Environment variable` change, refresh the project by pressing `F5` key. It is also recommended to rebuild index from menu **Project → C/C++ Index → Rebuild**

## 4.3. Enable Active Build Configuration

Using `Active Build Configuration` might require additional setting either in the IDE workspace or in the Project Settings to make project Indexer reflect changes in Build configuration target from the user.

The user has two options on how to enable this feature:

1. Workspace wide setting influencing all projects from the menu **Windows → Preferences → C/C++→Indexer → Use active build configuration**.
2. Project-specific setting from the menu **Project → Properties → C/C++ General → Indexer → Enable project specific setting (Use active build configuration)**.

# 5. Example configuration

## 5.1. Board (EVB) configuration

### 5.1.1. APPKIT_TC2X4_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] = {
    {13, 0}, /* LED_0 */
    {13, 1}, /* LED_1 */
    {13, 2}, /* LED_2 */
    {13, 3}, /* LED_3 */
};

/* External WDG Disable */
void board_wdg_Disable(void)
{
    disable TLF35584 external WDG
}
```

## 5.2. Hardware configuration

The hardware configuration relies on underlying BSP functionality for a given evaluation board. Executing a set of BSP API functions during Pre and Port Init hooks sets the hardware platform:

Cpu

- System clock = 200MHz
- SRI clock = 200MHz
- SPB clock = 100MHz

Stm

- STM0 Period = 1ms

Wdg

- Core 0 WDT = DISABLE

## 5.3. PXROS-HR System configuration

The PXROS-HR configuration provides parameters for kernel initialization and is part of `system_cfg.h` file. As each core holds its kernel instance, the configuration defines values for each of them.

PXROS-HR system

- Master core: Core0
- PXROS timebase ticks: 1000Hz

Core0

- State: ACTIVE
- Number of objects: 260
- Number of global objects: 10
- Max number of task: 20
- Number of CSA region: 64
- Task memory size: 16kB
- User system stack: 512B

## 5.4. Task configuration

To successfully create a task in PXROS-HR system the task must provide a set of initialization parameters during `PxTaskCreate` system call.

The main difference between three tasks in the example is the task privilege, where only the LedServer task has the right to directly access hardware peripherals having such peripheral regions in its Protected Regions configuration list.

LedClientA

- Execution core: Core0
- Stack type: PXStackAlloc
- Stack size: 400 Bytes
- Task privileges: PXUser0Privilege

LedClientB

- Execution core: Core0
- Stack type: PXStackAlloc
- Stack size: 400 Bytes
- Task privileges: PXUser0Privilege

LedServer

- Execution core: Core0
- Stack type: PXStackAlloc
- Stack size: 400 Bytes
- Task privileges: PXUser1Privilege
- Protected Regions:
    - range: &MODULE_P13 .. &MODULE_P33
    - protection type: WRProtection

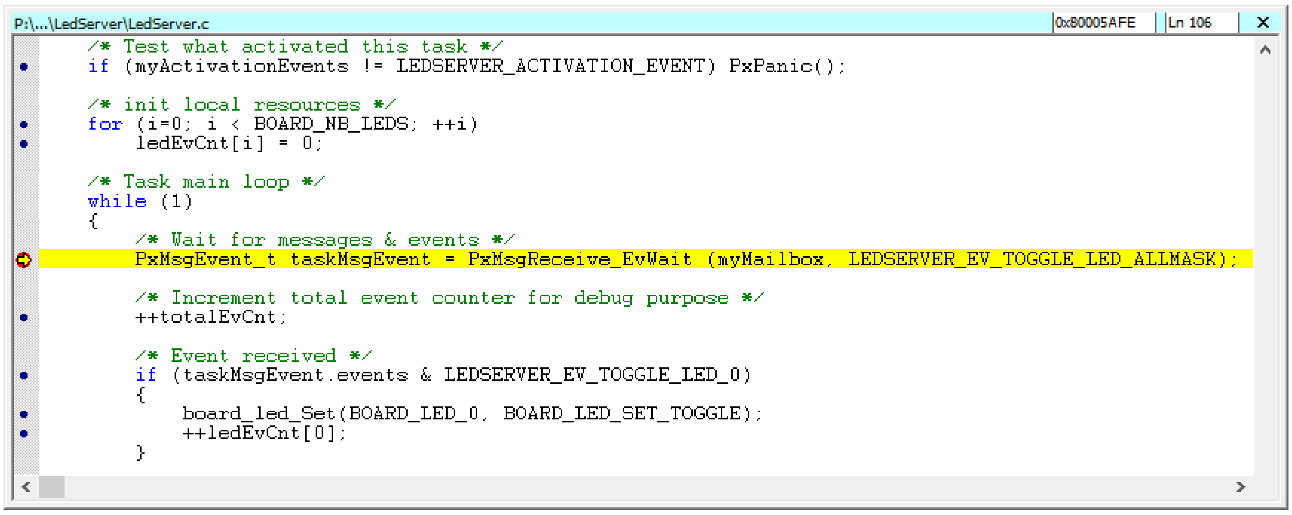
# 6. Debugging and testing the application

To run and verify the application

1. connect to the target in your debugger tool
2. load the `tc234_pxros_bsp_example.ELF` image from the `iROM_<board>` directory of the `tc234_pxros_bsp_example` IDE project
3. run the application

LED0 & LED1 shall toggle periodically if the application runs correctly.

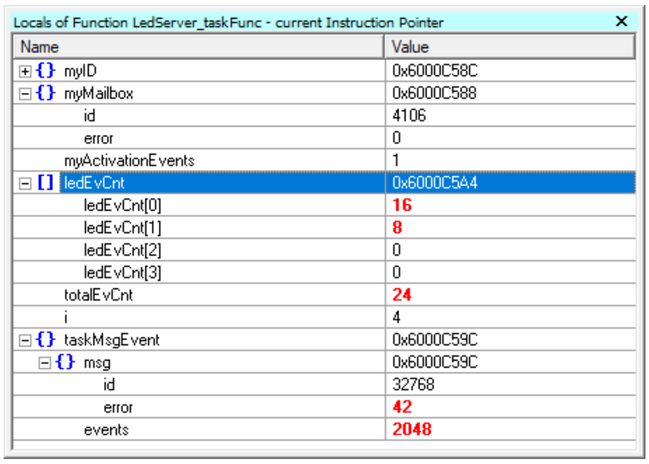
Another view on application execution is an observation of LedServer task local variables `ledEvCnt` and `totalEvCnt` in the debugger environment. To do so, set the breakpoint into the main loop of the `LedServer` task (located in the `LedServer.c` source), and open the *Locals* variable view. When the application triggers the breakpoint and stops, you can see the requested results.



P:\...\LedServer\LedServer.c | 0x80005AFE | Ln 106

```
    /* Test what activated this task */
    if (myActivationEvents != LEDSERVER_ACTIVATION_EVENT) PxPanic();

    /* init local resources */
    for (i=0; i < BOARD_NB_LEDS; ++i)
        ledEvCnt[i] = 0;

    /* Task main loop */
    while (1)
    {
        /* Wait for messages & events */
        PxMsgEvent_t taskMsgEvent = PxMsgReceive_EvWait (myMailbox, LEDSERVER_EV_TOGGLE_LED_ALLMASK);

        /* Increment total event counter for debug purpose */
        ++totalEvCnt;

        /* Event received */
        if (taskMsgEvent.events & LEDSERVER_EV_TOGGLE_LED_0)
        {
            board_led_Set(BOARD_LED_0, BOARD_LED_SET_TOGGLE);
            ++ledEvCnt[0];
        }
```

Fig. 6. LedServer main loop



Locals of Function LedServer_taskFunc - current Instruction Pointer

| Name | Value |
|---|---|
| myID | 0x6000C58C |
| myMailbox | 0x6000C588 |
| id | 4106 |
| error | 0 |
| myActivationEvents | 1 |
| ledEvCnt | 0x6000C5A4 |
| ledEvCnt[0] | 16 |
| ledEvCnt[1] | 8 |
| ledEvCnt[2] | 0 |
| ledEvCnt[3] | 0 |
| totalEvCnt | 24 |
| i | 4 |
| taskMsgEvent | 0x6000C59C |
| msg | 0x6000C59C |
| id | 32768 |
| error | 42 |
| events | 2048 |

Fig. 7. Event counts

# 7. Document References

[1] "HighTec PXROS-HR Project Management", HighTec EDV Systeme GmbH, 2019

[2] "AURIX BSP Guide" , HighTec EDV Systeme GmbH, 2019

[3] "PXROS-HR BSP Example - PXROS Guide" , HighTec EDV Systeme GmbH, 2019

# 8. Document history

| Version | Date | Changes to the previous version |
|---|---|---|
| 1.0 | April 2019 | Initial version<br>Based on TC234 BSP v2.0 |
| 1.1 | May 2019 | Added Active Build Configuration chapter.<br>Document structure updated. |

# 9. 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.

HIGHTEC

HighTec EDV-Systeme GmbH
Europaallee 19, D-66113 Saarbrücken
info@hightec-rt.com
+49-681-92613-16
www.hightec-rt.com