



TC37x PXROS-HR BSP example

Quick Guide

Table of Contents

Terms & Abbreviations	1
1. Introduction	2
1.1. Supported microcontrollers	2
1.2. Supported boards	2
1.3. Development tools	2
1.4. HW resources	3
2. Example overview	4
2.1. Application perspective	4
2.2. Component view	5
2.3. User tasks	6
2.4. Tasks to Core distribution	8
3. Example configuration	9
3.1. uC configuration	9
3.2. EVB configuration	9
3.3. PXROS-HR System configuration	10
4. Example Assemble & Build	12
4.1. Importing projects to Eclipse	12
5. Debugging and testing the application	13
Document References	14
Document history	15
Disclaimer	16

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 Support 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.

1. 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.

1.1. Supported microcontrollers

- TC37x

1.2. Supported boards

- APPKIT_TC3X7_V1.0
- APPKIT_TC3X7_V2.0
- TRIBOARD_TC3X5_V1.X
- TRIBOARD_TC377TX_V1.X
- TRIBOARD_TC3X7_V1.0
- TRIBOARD_TC3X7_V2.0

1.3. Development tools

- HighTec Development Suite, version 4.9.3.0 or higher
- HighTec PXROS-HR operating system, version 8.2.0

1.4. 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 from LedClientA task
EVB	LED[1] *	A notification of a Message reception in LedServer task from LedClientB task
µC	STM[0]	PXROS-HR kernel tick base - core0
µC	STM[1]	PXROS-HR kernel tick base - core1
µC	STM[2]	PXROS-HR kernel tick base - core2

Tab. 1. HW resources

* Please look at the [EVB configuration](#) chapter to find the LED connection to uC GPIO port-pin.

2. Example overview

2.1. Application perspective

The example implements three user tasks. Two LedClient tasks ask service from the third LedServer task to toggle a particular LED. Each client task utilizes a different communication means to ask for it.

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

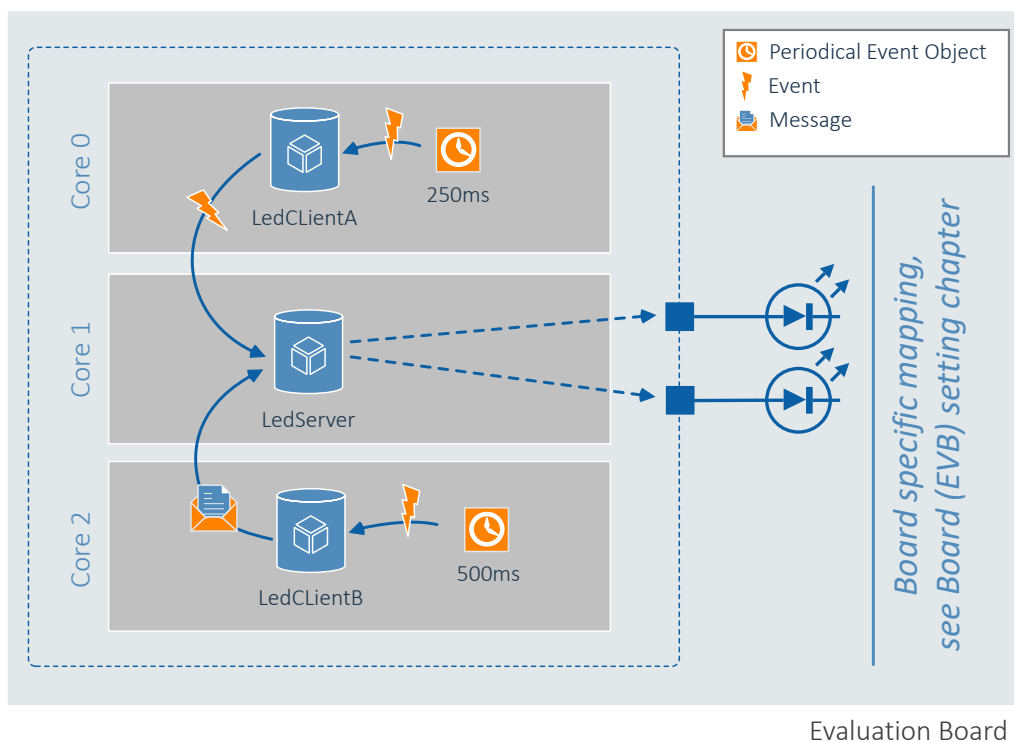


Fig. 1. Application block diagram

2.2. Component view

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

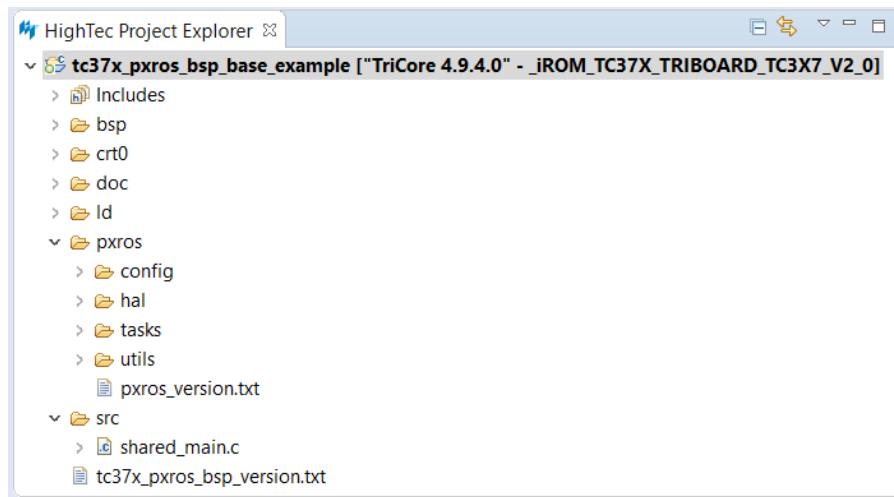


Fig. 2. Component view

bsp

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

crt0

A toolchain and a microcontroller dependent startup code that initializes a 'C' runtime environment.

ld

Linker files prescribing placement of the final application code and data.

src

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

pxros/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.

pxros/hal

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

pxros/tasks

A folder with the implementation of example tasks.

pxros/utils

Common utilities used across user tasks.

2.3. User tasks

2.3.1. 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 the EVB on-board LEDs. The LedServer selects the LED according to the received event or message.

The LedServer gets active for two reasons:

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

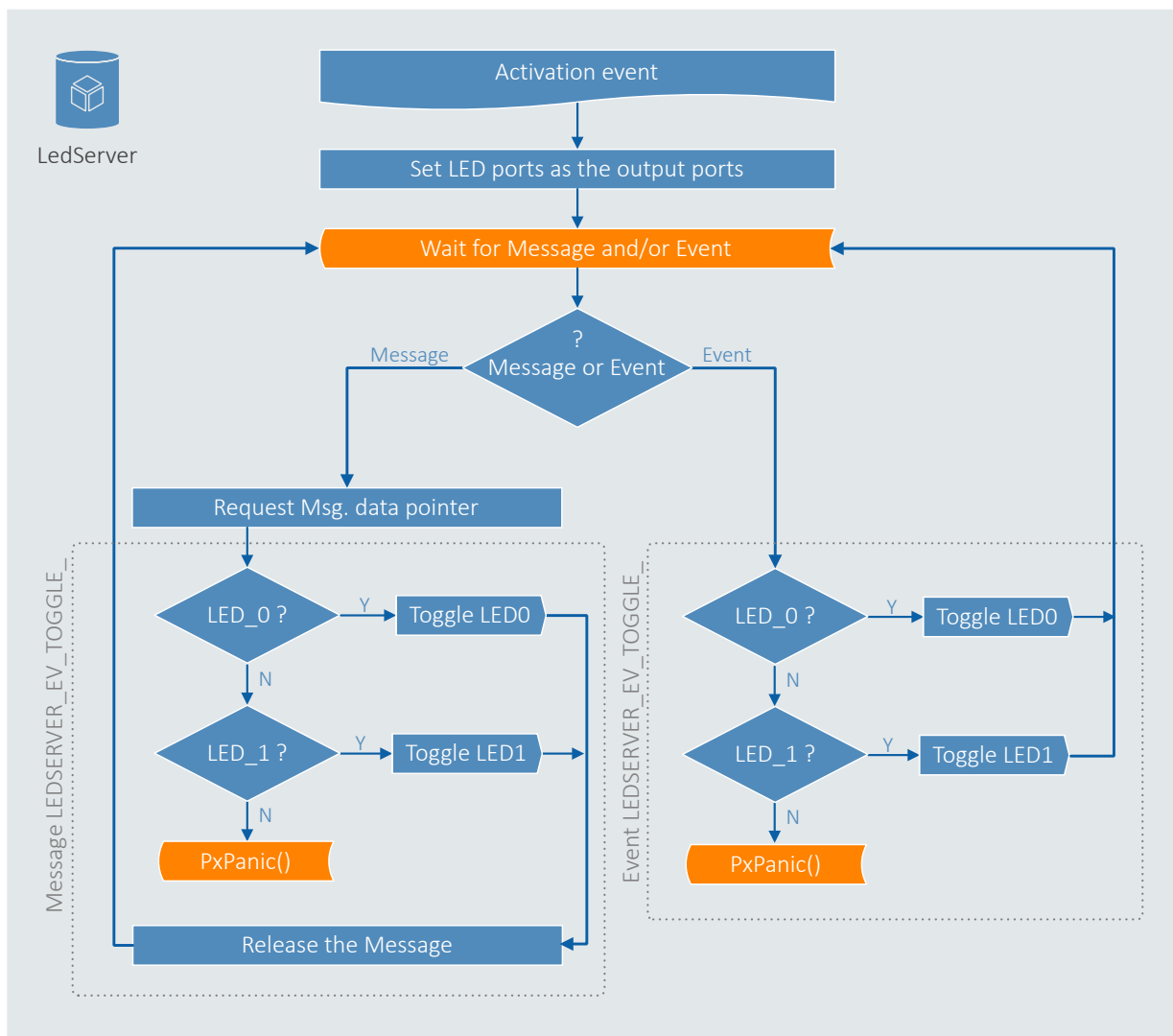


Fig. 3. Simplified LedServer task execution flow

2.3.2. LedClientA

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

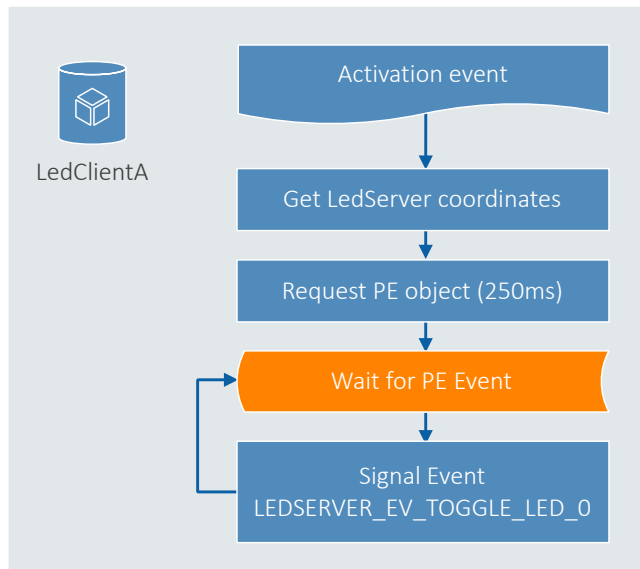


Fig. 4. Simplified LedClientA tasks execution flow

2.3.3. LedClientB

A structure of LedClientB tasks is almost identical to the LedClientA task. The task uses its 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.

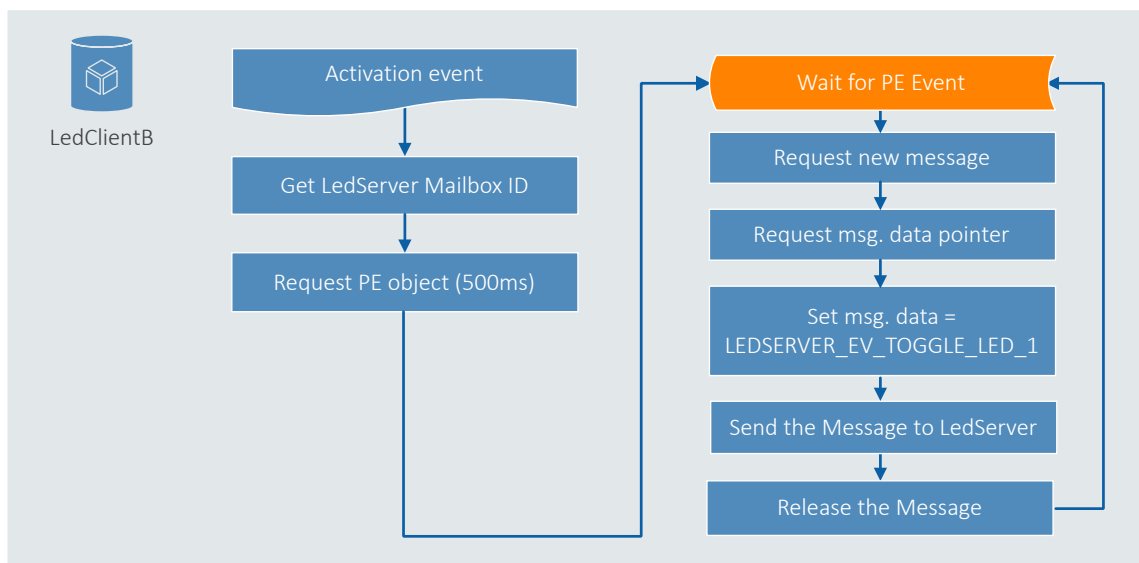


Fig. 5. Simplified LedClientB tasks execution flow

2.4. Tasks to Core distribution

The decision of which task will be running on which core is made inside the `InitTask_Func()` located in the `InitTask.c` file. This function calls the `TaskDeploy()` function which reads the "Task Deployment Table" and, together with the `coreId` information, creates and starts the user task(s). The snapshot of the `InitTask_Func()` is shown below.

Code 1. Task to core distribution via the `TaskDeploy` function

```
void InitTask_Func (PxTask_t myID, PxBbx_t myMailbox, PxEvents_t myActivationEvents)
{
    ...

    /* User Task Deployment */
    TaskDeploy(coreId);

    /* Infinite loop on all cores as background activity */
    while (1)
        ;
}
```

The "Task Deployment Table" is declared in `taskDeployment.h` header file and defined in the `taskDeployment.c` source file. The following two snapshots show the table structure more in detail.

Code 2. Declaration of the Task Deployment Table

```
/* prototype of the Task Create Function */
typedef PxTask_t (*TaskCreateFnc)(PxPrio_t prio, PxEvents_t ev, PxMc_t mc, PxOpool_t op);

typedef struct {
    TaskCreateFnc fnc;      // address of the Task create function
    PxPrio_t prio;         // task priority on assigned core
    PxUInt_t core;         // assigned core to run on
} task_deployment_t;
```

Code 3. Definition of the Task Deployment Table

```
const task_deployment_t taskTable[] =
{
    { LedClientA_Create, LEDCLIENTA_PRIO,          PXCORE_0          },
    { LedServer_Create, LEDSERVER_PRIO, (CORE1_ACTIVE) ? PXCORE_1 : PXCORE_0 },
    { LedClientB_Create, LEDCLIENTB_PRIO, (CORE2_ACTIVE) ? PXCORE_2 : PXCORE_0 },
};
```

Note that the shown implementation is only one of many possible solutions. Here, most of the task's parameters are fixed - only the task priority is configurable. It is, however, easy to make other parameters like default memory class and object pool configurable in the same way.

3. Example configuration

3.1. uC configuration

The hardware configuration relies on underlying BSP functionality for a given evaluation board. Executing a set of BSP API functions during PreInit and PostInit hooks brings the hardware platform to the following state.

Cpu

- System clock = 300MHz
- SRI clock = 300MHz
- SPB clock = 100MHz

Stm

- STM[0] period = 1ms
- STM[1] period = 1ms
- STM[2] period = 1ms

Wdg

- Core[0] WDT = DISABLE
- Core[1] WDT = DISABLE
- Core[2] WDT = DISABLE

3.2. EVB configuration

The bsp/board folder abstracts the physical LED connection and provides a routine to disable eventual external watchdog for each EVB board specifically. The user selects one of the supported boards through the Project Build Configuration.

Common configuration for all EVBs

- TLF WDG = Present (Disabled)

APPKIT_TC3X7 (V1.0, V2.0)

- LED[0] = Port[13], Pin[0]
- LED[1] = Port[13], Pin[1]

All supported TRIBOARDS

- LED[0] = Port[33], Pin[4]
- LED[1] = Port[33], Pin[5]

3.3. PXROS-HR System configuration

3.3.1. Kernel configuration

The PXROS-HR configuration provides parameters for all kernel instances running on all configured cores. The configuration structures can be found in the `system_cfg.h` file.

PXROS-HR system

- Master core = Core 0
- PXROS timebase ticks = 1000Hz (1ms)

All cores

- State = ACTIVE
- Number of objects = 200
- Number of global objects = 0 (i.e. all objects are considered global)
- Max number of task = 20
- Number of CSA region = 128
- Task memory size = 16 kB
- User system stack = 512 B

3.3.2. MPU configuration

The MPU is configured during the `PxInit()` call according to the initialization structure passed as an input argument. This structure holds pointers to another structures that defines the static settings of the MPU. The definition of these MPU-related structures can be found in `system_mpu_cfg.c` file.

Regions executable by System and Kernel

- Whole Program Flash memory

Regions executable by Tasks

- Whole Program Flash memory

Data regions accessible by Kernel and System (i.e. interrupts)

- Whole Program Flash memory (R/O access for Kernel)
- CSA (R/W access for Kernel)
- Peripherals (R/W access for Kernel)
- Kernel's R/W data (R/W access for Kernel)
- System stack (R/W access for System)

3.3.3. User task configuration

All user tasks must provide a set of initialization parameters for the `PxTaskCreate()` system call.

The main differences among the tasks in this example are the task privileges: the `LedServer` task has the permission to access hardware peripherals directly while the `LedClient` tasks haven't.

Common settings

- Stack type = `PXStackAlloc`
- Stack size = 400 Bytes

LedClientA

- Execution core = Core 0
- Task privileges = `PXUser0Privilege`
- Task Priority = 15

LedClientB

- Execution core = Core 2
- Task privileges = `PXUser0Privilege`
- Task Priority = 15

LedServer

- Execution core = Core 1
- Task privileges = `PXUser1Privilege`
- Protected Regions:
 - range = `&MODULE_P00 ... &MODULE_P33`
 - protection type = `WRProtection`
- Task Priority = 16

4. Example Assemble & Build

The build of the application example consists of these steps:

1. Import IDE project (tc37x_pxros_bsp_example)
2. Update PXROS_HR_INSTALL_PATH variable in *Environment* setting according to the installation folder of the PXROS-HR package
3. Set the project Active from menu **Project → Set Active Project**.
4. Build the project from the menu **Project → Build Project**.
5. The output binary file is located under the _iROM_<BuildConfiguration> folder.

Eclipse Environment

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**

4.1. Importing projects to Eclipse

Since the HighTec IDE projects are Eclipse based, the import process is following:

1. Start HighTec IDE platform
2. Choose from the menu **File → Import**
3. Select **General → Existing Project into Workspace**
4. Browse for your project location
5. Leave the option Copy projects into workspace unchecked
6. Finish import

For more details about project structure, project import and build see [\[1\]](#).

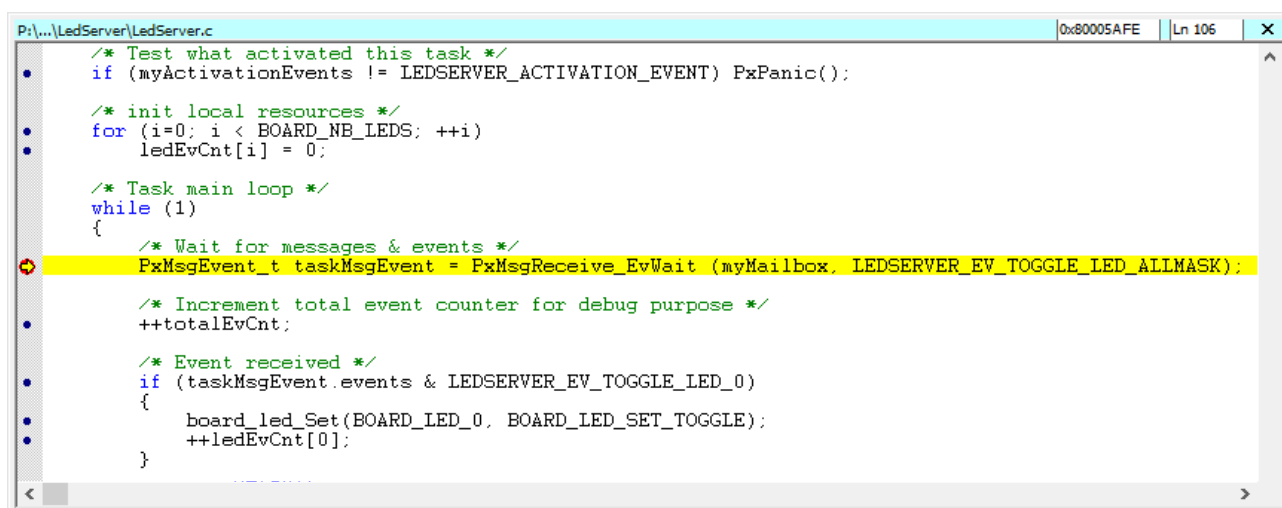
5. Debugging and testing the application

To run and verify the application

1. Connect to the target in your debugger tool . Load the tc37x_pxros_bsp_example.ELF image from the _iROM_<BuildConfiguration> directory
2. Run the application

Evaluation board's LED shall toggle periodically in case 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();

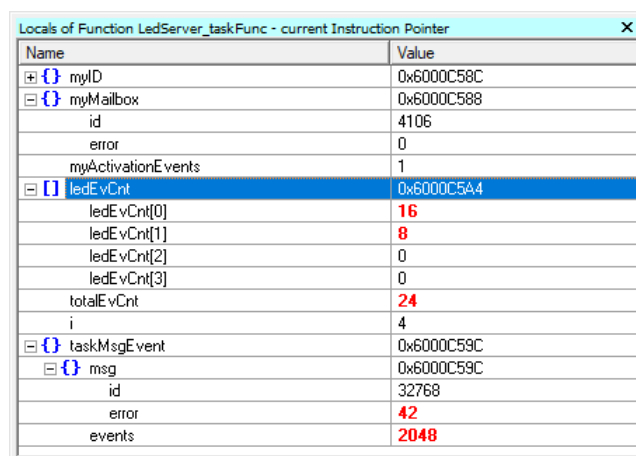
/* init local resources */
for (i=0; i < BOARD_NB_LEDS; ++i)
    ledEvCnt[i] = 0;

/* 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



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

Document References

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

Document history

Version	Date	Changes to the previous version
4.0	August 2020	Document updated to align with new TC3xx PXROS BSP examples that are built upon PXROS-HR version 8.2.0.

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 © 2020 HighTec EDV-Systeme GmbH, D-66113 Saarbrücken.



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