

## example\_freertosBlinky.c

This is an example which describes the steps to create an example application which toggles the High End Timer (HET) pin 17 (LED in USB & HDK) based on the FreeRTOS timer tick of one second.

### Step 1:

Create a new project.

Navigate: -> File -> New -> Project

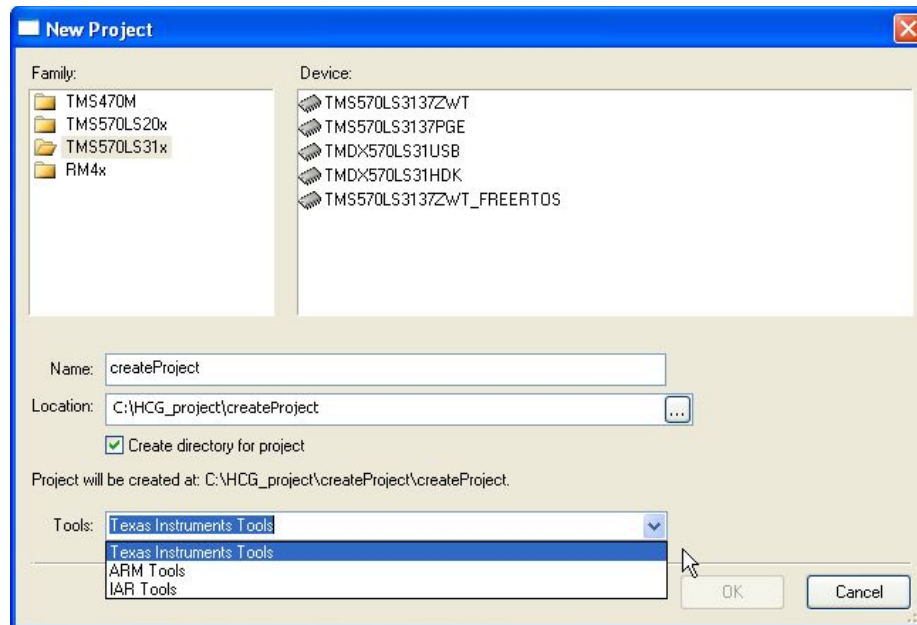


Figure: Create a new Project

### Step 2:

Configure driver code generation:

- Enable GIO driver
- Disable others

Navigate: -> TMS570LSxx /RM4 -> Driver Enable

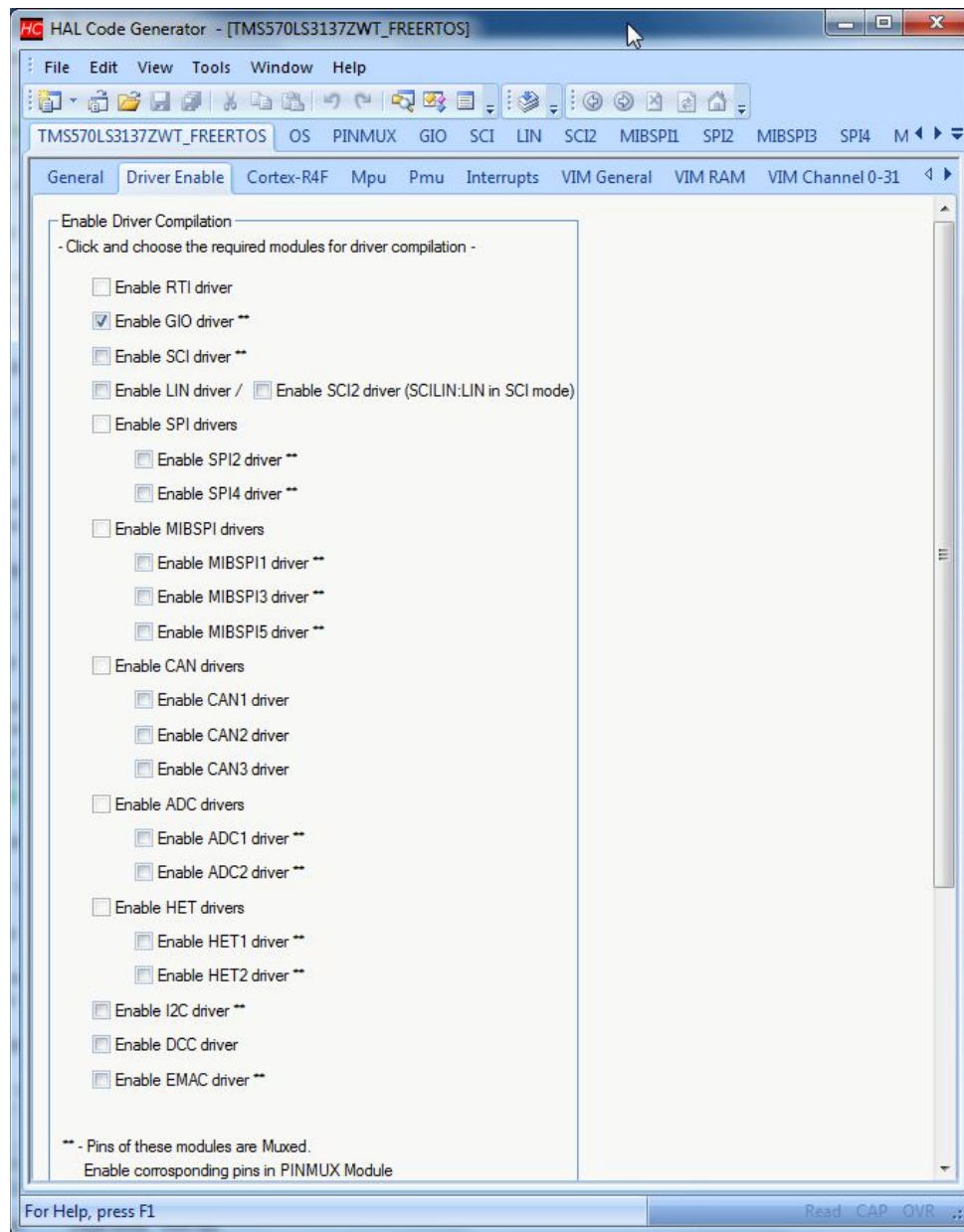


Figure: Driver Configuration

**Step 3:**

Configure Interrupt handling:

- Enable SVC
- Enter FreeRTOS SVC handler name 'vPortSWI'

Navigate: -> TMS570LSxx /RM4 -> Interrupts

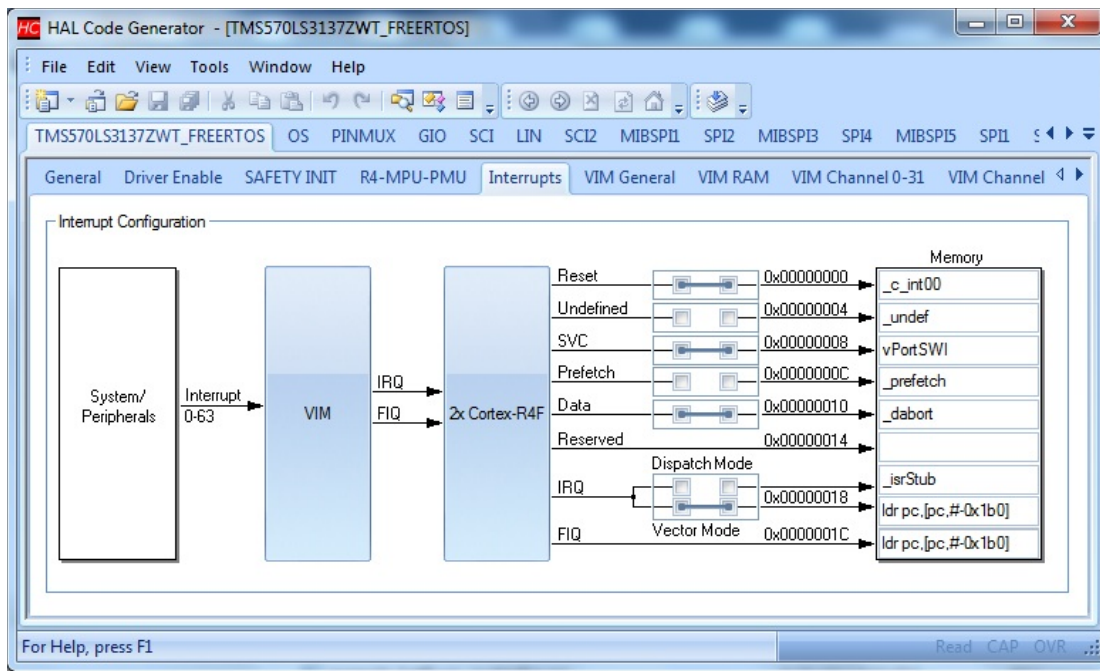


Figure: Interrupt Configuration

**Step 4:**

Configure VIM RAM:

- Enter FreeRTOS Timer Tick handler name 'vPortPreemptiveTick' at offset 0x0000000C
- Enter SSI handler name 'vPortYieldWithinAPI' at offset 0x00000058

Navigate: -> TMS570LSxx /RM4 -> VIM RAM

The screenshot shows the 'VIM RAM : ISR Assignments' window. The 'Base Address' is set to 0xFF82000. The table below lists the assignments:

Offset	Handler Name	Offset	Handler Name	Offset	Handler Name	Offset	Handler Name
0x00000000:PH	phantomInterrupt	0x00000040:15	adc1Group1Interrupt	0x00000080:31	phantomInterrupt	0x000000C0:47	phantomInterrupt
0x00000004:00	esmHighInterrupt	0x00000044:16	can1HighLevelInterrupt	0x00000084:32	phantomInterrupt	0x000000C4:48	phantomInterrupt
0x00000008:01	phantomInterrupt	0x00000048:17	spi2HighLevelInterrupt	0x00000088:33	phantomInterrupt	0x000000C8:49	spi4HighLevelInterrupt
0x0000000C:02	vPortPreemptiveTick	0x0000004C:18	phantomInterrupt	0x0000008C:34	phantomInterrupt	0x000000CC:50	adc2Group0Interrupt
0x00000010:03	phantomInterrupt	0x00000050:19	crclInterrupt	0x00000090:35	can2HighLevelInterrupt	0x000000D0:51	adc2Group1Interrupt
0x00000014:04	phantomInterrupt	0x00000054:20	esmLowLevelInterrupt	0x00000094:36	phantomInterrupt	0x000000D4:52	phantomInterrupt
0x00000018:05	phantomInterrupt	0x00000058:21	vPortYieldWithinAPI	0x00000098:37	mibspi3HighLevelInterruptLeve	0x000000D8:53	mibspi5HighLevelInterrupt
0x0000001C:06	phantomInterrupt	0x0000005C:22	phantomInterrupt	0x0000009C:38	mibspi3LowLevelInterrupt	0x000000DC:54	spi4LowLevelInterrupt
0x00000020:07	phantomInterrupt	0x00000060:23	gioLowLevelInterrupt	0x000000A0:39	phantomInterrupt	0x000000E0:55	can3LowLevelInterrupt
0x00000024:08	phantomInterrupt	0x00000064:24	het1LowLevelInterrupt	0x000000A4:40	phantomInterrupt	0x000000E4:56	mibspi5LowLevelInterrupt
0x00000028:09	gioHighLevelInterrupt	0x00000068:25	phantomInterrupt	0x000000A8:41	phantomInterrupt	0x000000E8:57	adc2Group2Interrupt
0x0000002C:10	het1HighLevelInterrupt	0x0000006C:26	mibspi1LowLevelInterrupt	0x000000AC:42	can2LowLevelInterrupt	0x000000EC:58	phantomInterrupt
0x00000030:11	phantomInterrupt	0x00000070:27	linLowLevelInterrupt	0x000000B0:43	phantomInterrupt	0x000000F0:59	phantomInterrupt
0x00000034:12	mibspi1HighLevelInterrupt	0x00000074:28	adc1Group2Interrupt	0x000000B4:44	phantomInterrupt	0x000000F4:60	phantomInterrupt
0x00000038:13	linHighLevelInterrupt	0x00000078:29	can1LowLevelInterrupt	0x000000B8:45	can3HighLevelInterrupt	0x000000F8:61	phantomInterrupt
0x0000003C:14	adc1Group0Interrupt	0x0000007C:30	spi2LowLevelInterrupt	0x000000BC:46	phantomInterrupt	0x000000FC:62	phantomInterrupt

Figure: VIM RAM Configuration

**Step 5:**

Configure Vectored Interrupt Module Channels:

- Enable VIM Channel 2
- Map VIM Channel 2 to IRQ
- Enable VIM Channel 21
- Map VIM Channel 21 to IRQ

Navigate: -> TMS570LSxx /RM4 -> VIM Channel 0-31

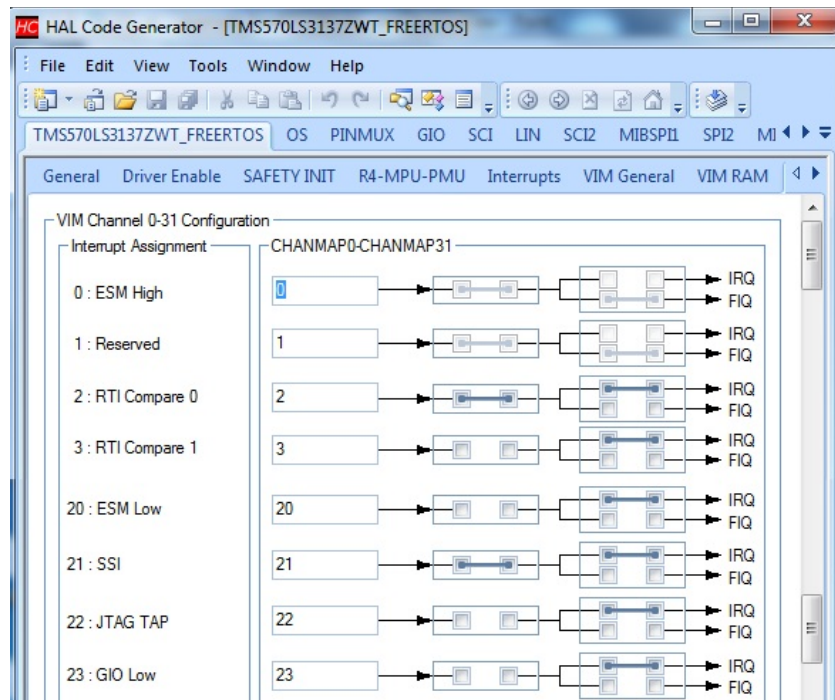


Figure: VIM Channel Configuration

**Step 6:**

Configure OS timer tick to 1 ms:

- Enter Tick Rate of 1000

Navigate: -> OS -> General

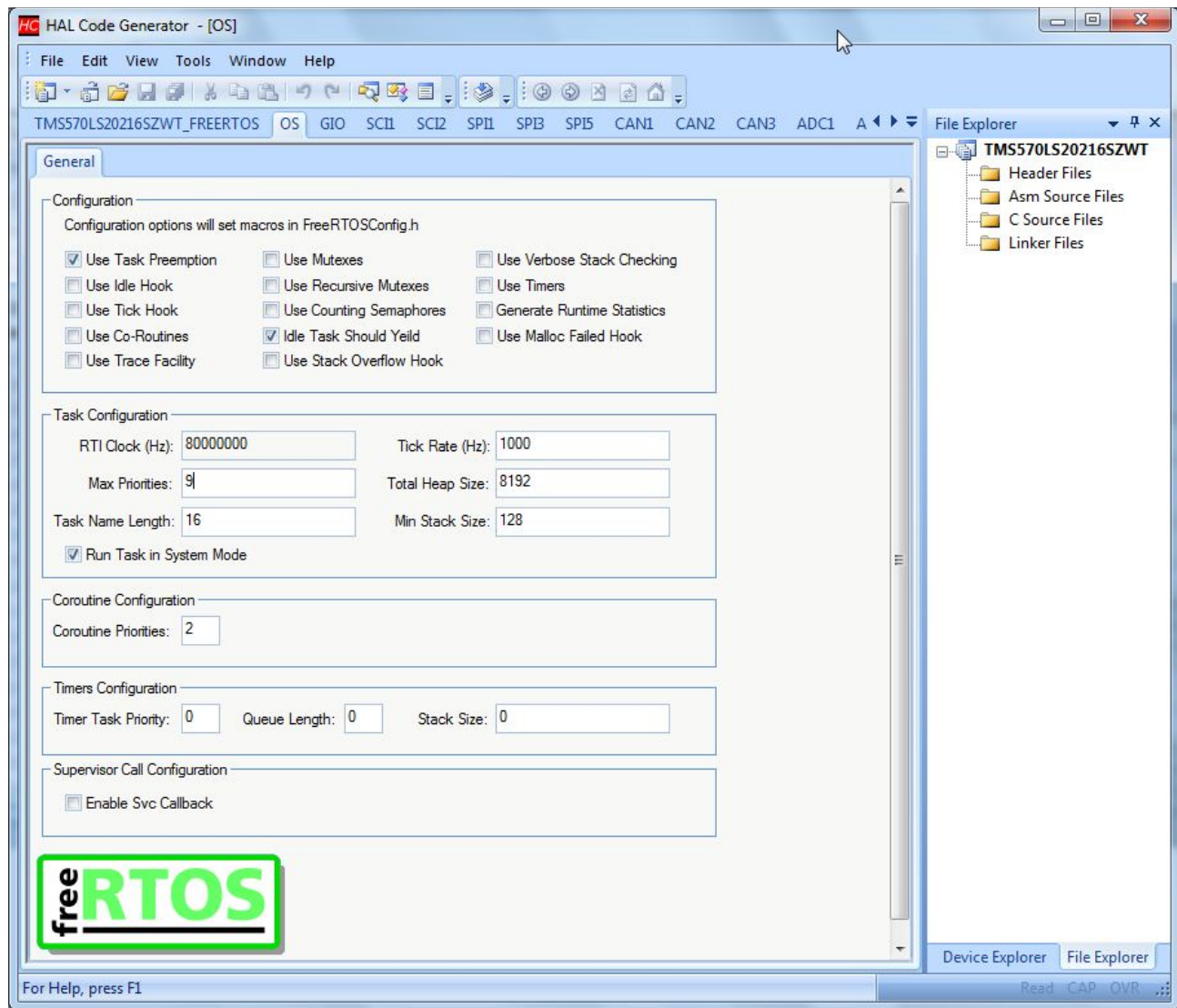


Figure: OS General Configuration

**Step 7:**

Generate code

Navigate: -&gt; File -&gt; Generate Code

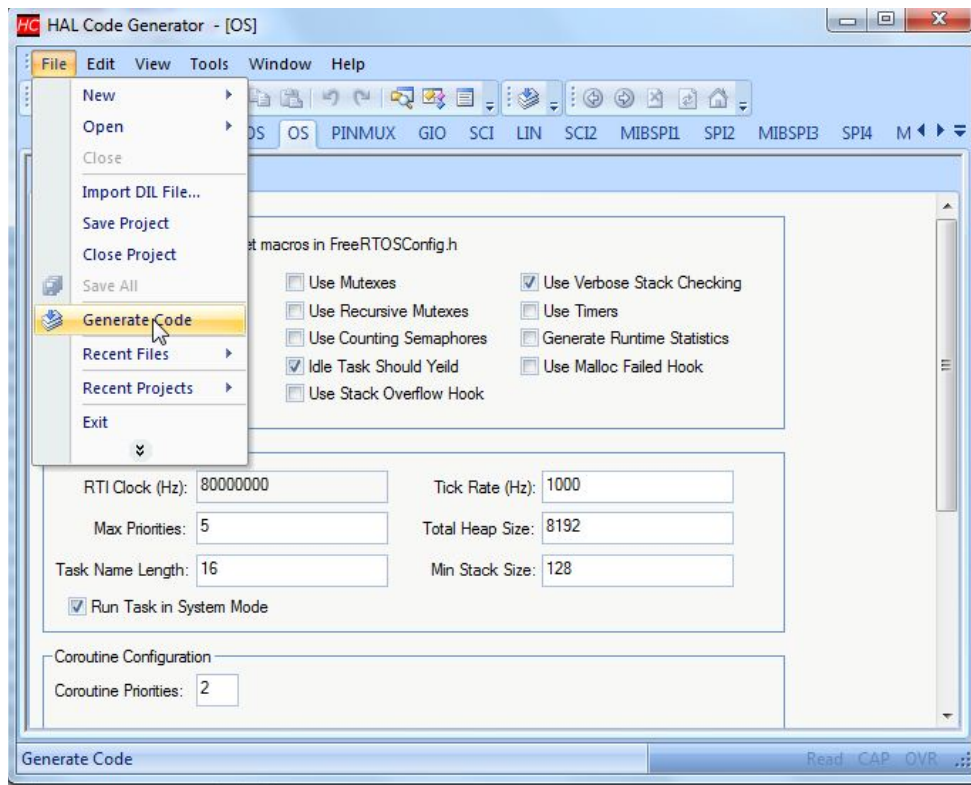


Figure: Generate Code

**Step 8:**

Copy source code below into your application.

The example file [example\\_freRTOSBlinky.c](#) can also be found in the examples folder: `../HALCoGen/examples`

**Note**

HALCoGen generates an empty main function in `sys_main.c`, please make sure that you link in the right main function or copy the source into the user code sections of this file.

Enable GCC extension in the CCS project (Project properties -> Build -> ARM Compiler -> Advanced options -> Language options -> Enable support for GCC extensions)

```

/*
 * Copyright (C) 2009-2015 Texas Instruments Incorporated - www.ti.com
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 *   Redistributions of source code must retain the above copyright
 *   notice, this list of conditions and the following disclaimer.
 *
 *   Redistributions in binary form must reproduce the above copyright
 *   notice, this list of conditions and the following disclaimer in the
 *   documentation and/or other materials provided with the
 *   distribution.
 *
 *   Neither the name of Texas Instruments Incorporated nor the names of
 *   its contributors may be used to endorse or promote products derived
 *   from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES INCLUDING, BUT NOT
 * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
 * DATA, OR PROFITS; OR BUSINESS INTERRUPTION HOWEVER CAUSED AND ON ANY
 * THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * INCLUDING NEGLIGENCE OR OTHERWISE ARISING IN ANY WAY OUT OF THE USE
 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */

```

```

/* USER CODE BEGIN (0) */
/* USER CODE END */

/* Include Files */
#include "sys_common.h"
#include "system.h"

/* USER CODE BEGIN (1) */
/* Include FreeRTOS scheduler files */
#include "FreeRTOS.h"
#include "os_task.h"

/* Include HET header file - types, definitions and function declarations for system driver */
#include "het.h"
#include "gio.h"

/* Define Task Handles */
xTaskHandle xTask1Handle;

/* Task1 */
void vTask1(void *pvParameters)
{
    for(;;)
    {
        /* Taggle HET[1] with timer tick */
        gioSetBit(hetPORT1, 17, gioGetBit(hetPORT1, 17) ^ 1);
        vTaskDelay(100);
    }
}
/* USER CODE END */

/* USER CODE BEGIN (2) */
/* USER CODE END */

void main(void)
{
/* USER CODE BEGIN (3) */

    /* Set high end timer GIO port hetPort pin direction to all output */
    gioSetDirection(hetPORT1, 0xFFFFFFFF);

    /* Create Task 1 */
    if (xTaskCreate(vTask1,"Task1", configMINIMAL_STACK_SIZE, NULL, 1, &xTask1Handle) != pdTRUE)
    {
        /* Task could not be created */
        while(1);
    }

    /* Start Scheduler */
    vTaskStartScheduler();

    /* Run forever */
    while(1);
/* USER CODE END */
}

/* USER CODE BEGIN (4) */
/* USER CODE END */

```