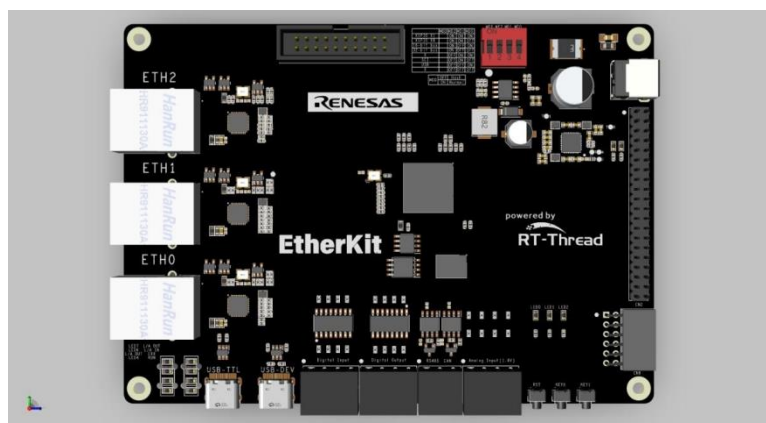


RZN2L CMT 定时器-----基于 Etherkit 开发板

简介

例程功能：添加一 **CMT** 定时器，定时 200ms 产生一次中断，让板载三个 LED 灯，按 200ms 周期闪烁。



开发工具 <ul style="list-style-type: none"> • IDE: IAR EW for Arm 9.50.2 或 E2studio 2024-01.1 • FSP: RZ/N2 FSP V2.0 • 仿真器: Jlink V12 	实验材料 <ul style="list-style-type: none"> • Etherkit 开发板 • Jlink 仿真器，需支持瑞萨 R52 内核
--	--

实验部分

1	硬件设置.....	2
2	IAR 环境开发入门.....	3
3	E2studio 环境开发入门.....	5

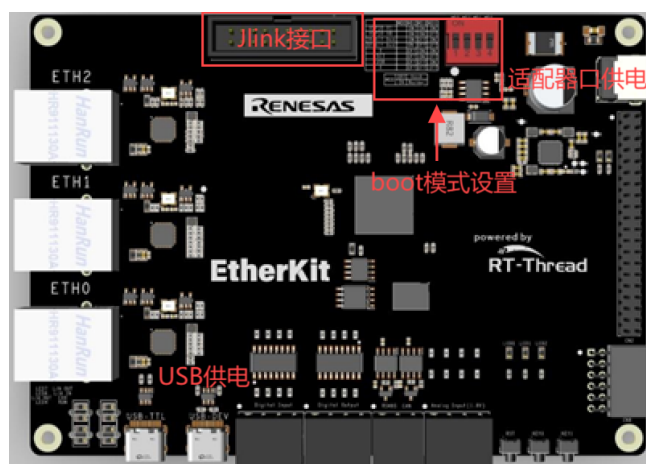
1 硬件设置

本节介绍使用 IAR 环境 和 E2studio 软件安装和环境配置。

1.1

开发板设置：

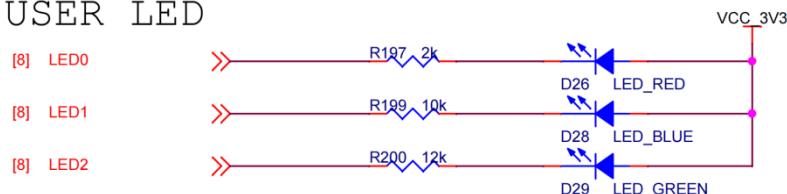
- 供电：可选 USB 供电或适配器供电
- Boot 模式设置：推荐 xSPI0 x1 boot mode
- Jlink v12



1.2

- LED 电路原理图

USER LED



- LED 对应引脚



EtherKit 提供三个用户 LED，分别为 LED0（RED）、LED1（BLUE）、LED2（GREEN），其中 LED_RED 对应引脚 P14_3。单片机引脚输出低电平即可点亮 LED，输出高电平则会熄灭 LED。

本节完

2 IAR 创建 CMT 工程

本节介绍使用 IAR 环境软件安装和环境配置。

2.1

● 新建工程

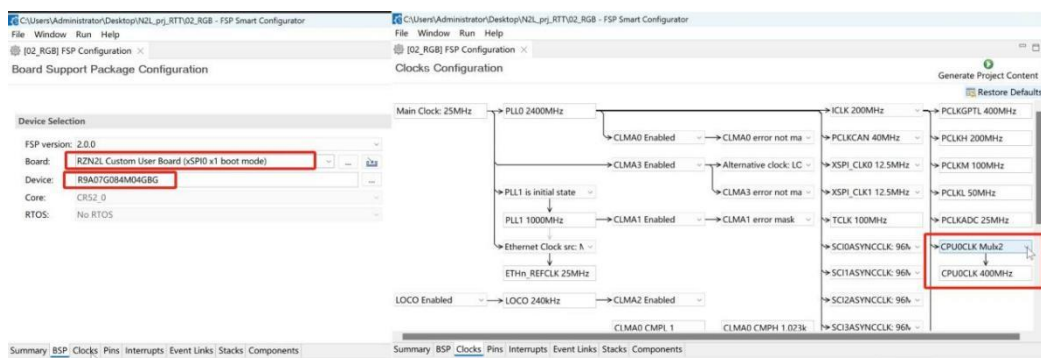
打开 FSP, File--New--FSP Project: 如下图

project name: 02_RGB

Board: RZN2L Custom User Board (xSPI0 x1 boot mode)

Device: R9A07G084M04GBG

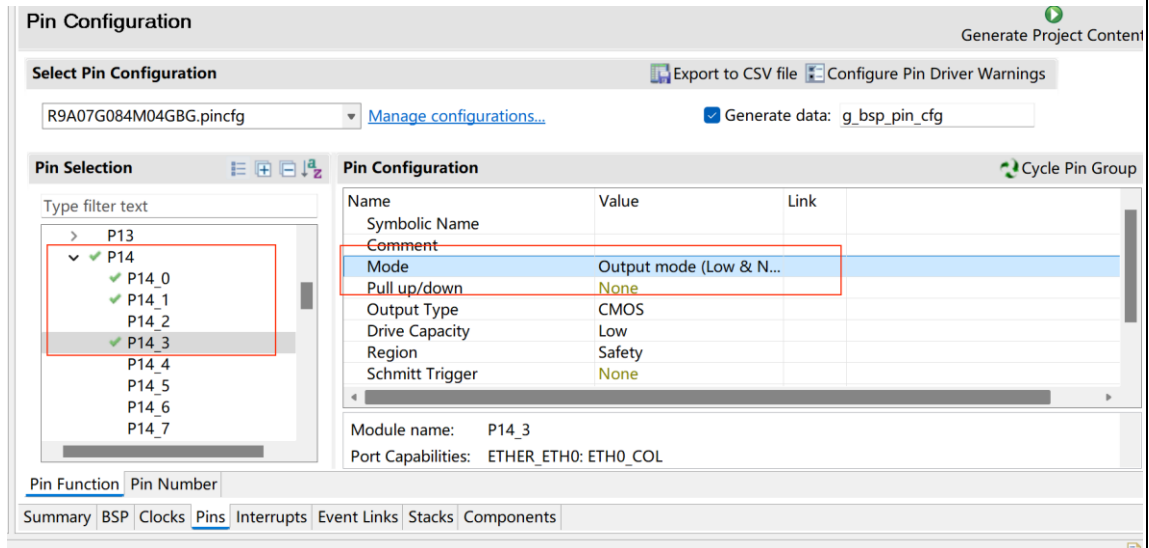
CPU0CLK: 400MHZ



2.2

● LED 引脚配置

1. 配置 P140 P141 P142 为输出模式，初始电平为低。FSP 可以作为 Tools 导入到 IAR 环境，方便后续随时打开 FSP，进行配置修改更新。



2. 添加定时器：New stacks--timers--Timer，Compare Match

● 定时器配置

Mode: Periodic

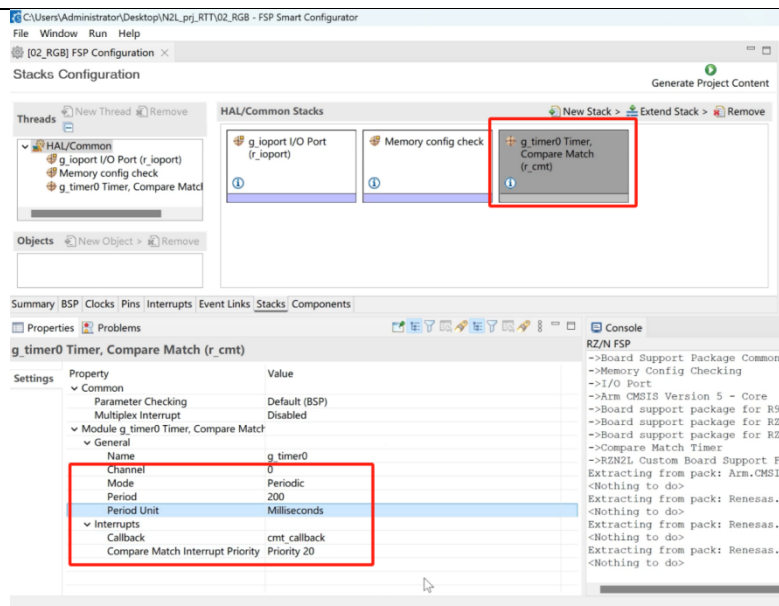
Period: 200

Period Uint: milliseconds

● 定时器中断设置

设置中断优先级，输入 Callback 名称: cmt_callback

说明：callback 函数是用户中断入口函数，需要自己编写，参考 FSP 帮助文档。



设置好后保存，点击右上角 **Generate Project Content**，即可生成工程。

2.3

- 编写用户代码
- 打开生成的工程 02_RGB, 进入 hal_entry.c, 编写用户代码。

```

1 void R_BSP_WarnStart (bsp_warn_start_event_t event) BSP_PLACE_IN_SECTION ("__warn_start");
2 PSP_CFP_FOOTER
3
4 bool current_state = 0;
5 void cnt_callback (timer_callback_args_t * p_args)
6 {
7     if (TIMER_EVENT_CYCLE_END == p_args->event)
8     {
9         /* Add application code to be called periodically here. */
10        current_state = !current_state;
11    }
12}
13
14 /* ===== */
15 /* main() is generated by the FSP Configuration editor and is used to generate threads (if an RTOS is used. This function
16 /* is called by main() when no RTOS is used.
17 /* ===== */
18 void hal_entry (void)
19 {
20     /* ===== Add user code here ===== */
21     R_CMT_Open (&g_timer0_ctrl, &g_timer0_cfg); //open timer
22     __asm volatile ("cpsie i"); // enable interrupt
23     R_CMT_Start (&g_timer0_ctrl); // start timer
24     while (1)
25     {
26         if (current_state == 0)
27         {
28             R_IOPORT_PinsWrite (&g_ioport_ctrl, BSP_IO_PORT_14_PIN_0, BSP_IO_LEVEL_LOW);
29             R_IOPORT_PinsWrite (&g_ioport_ctrl, BSP_IO_PORT_14_PIN_1, BSP_IO_LEVEL_LOW);
30             R_IOPORT_PinsWrite (&g_ioport_ctrl, BSP_IO_PORT_14_PIN_3, BSP_IO_LEVEL_LOW);
31         }
32         else
33         {
34             R_IOPORT_PinsWrite (&g_ioport_ctrl, BSP_IO_PORT_14_PIN_0, BSP_IO_LEVEL_HIGH);
35             R_IOPORT_PinsWrite (&g_ioport_ctrl, BSP_IO_PORT_14_PIN_1, BSP_IO_LEVEL_HIGH);
36             R_IOPORT_PinsWrite (&g_ioport_ctrl, BSP_IO_PORT_14_PIN_3, BSP_IO_LEVEL_HIGH);
37         }
38     }
39 }

```

- startup_core.c 文件添加 boot 模式 debug 需要添加的代码

```

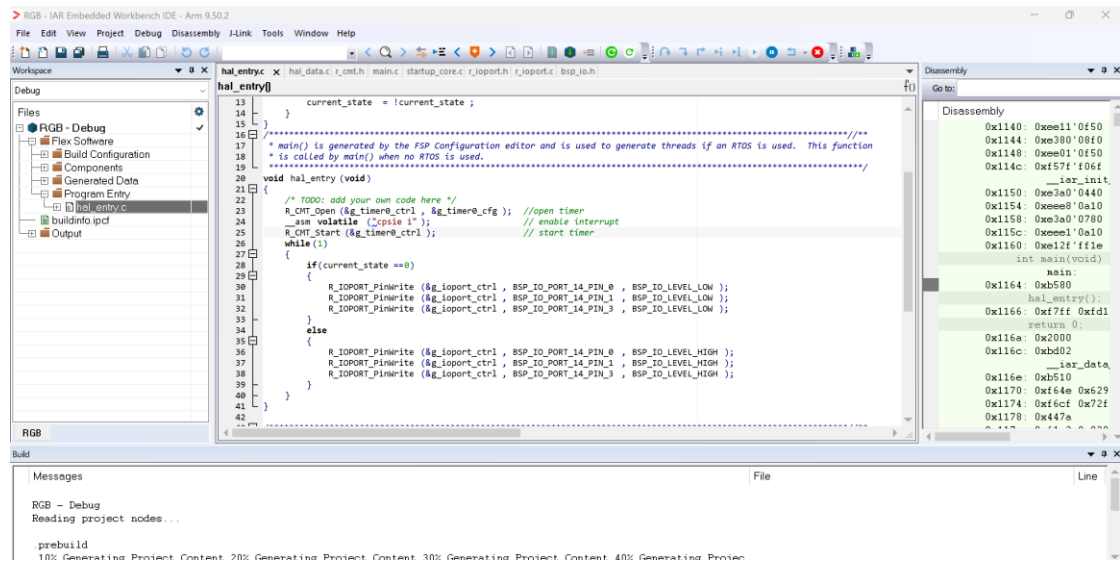
165 }
166
167 /* ===== */
168 /* After boot processing, LSI starts executing here.
169 /* ===== */
170 BSP_TARGET_ARM BSP_ATTRIBUTE_STACKLESS void system_init (void)
171 {
172     #if 1 // Software Loops are only needed when debugging.
173     __asm volatile (
174         "mov r0, #0\n"
175         "movw r1, #0xf07f\n"
176         "movt r1, #0x2fa\n"
177         "software_loop: \n"
178         "adds r0, #1\n"
179         "cmp r0, r1\n"
180         "bne software_loop\n"
181         "::: "memory");
182     #endif
183     __asm volatile (
184         "set_hactlr: \n"
185         "MOVW r0, %[bsp_hactlr_bit_l] \n" /* Set HACTLR bits(L) */
186         "MOVT r0, #0 \n"
187         "MCR p15, #4, r0, c1, c0, #1 \n" /* Write r0 to HACTLR */
188         "::[bsp_hactlr_bit_l] \"i\" (BSP_HACTLR_BIT_L) : \"memory\";
189     )
190     __asm volatile (
191         "set_hcr: \n"
192         "MRC p15, #4, r1, c1, c1, #0 \n" /* Read Hyp Configuration Register */
193         "ORR r1, r1, %[bsp_hcr_hcd_disable] \n" /* HVC instruction disable */
194         "MCR p15, #4, r1, c1, c1, #0 \n" /* Write Hyp Configuration Register */
195         "::[bsp_hcr_hcd_disable] \"i\" (BSP_HCR_HCD_DISABLE) : \"memory\";
196     )
197     __asm volatile (
198         "set_vbar: \n"
199         "IDR r0, = Vectors \n"

```

2.4

● 运行代码

编译代码，进入 debug 界面，全速运行。



2.5

● 运行效果

按观察开发板上 RGB-LED 的实际效果。正常运行后，LED 会周期性变化，如下图所示：



本节完

3 E2studio 创建 CMT 工程

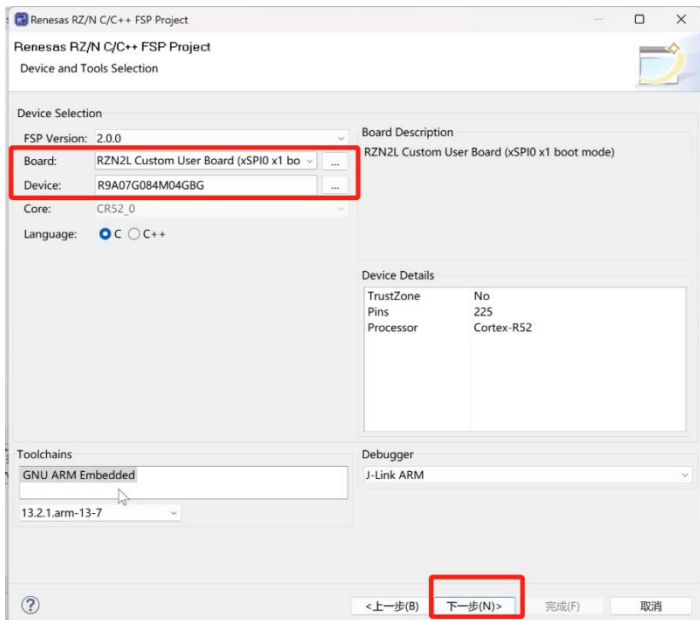
3.1	<p>E2studio 软件安装：</p> <ul style="list-style-type: none"> 安装 E2studio : RZN2L: setup_rznfsp_v2_0_0_e2s_v2024-01.1.exe
3.2	<p>E2studio 新建工程</p> <ul style="list-style-type: none"> 点击：文件 > 新建 > 瑞萨 C/C++ 项目 > Renesas RZ. 

3.3

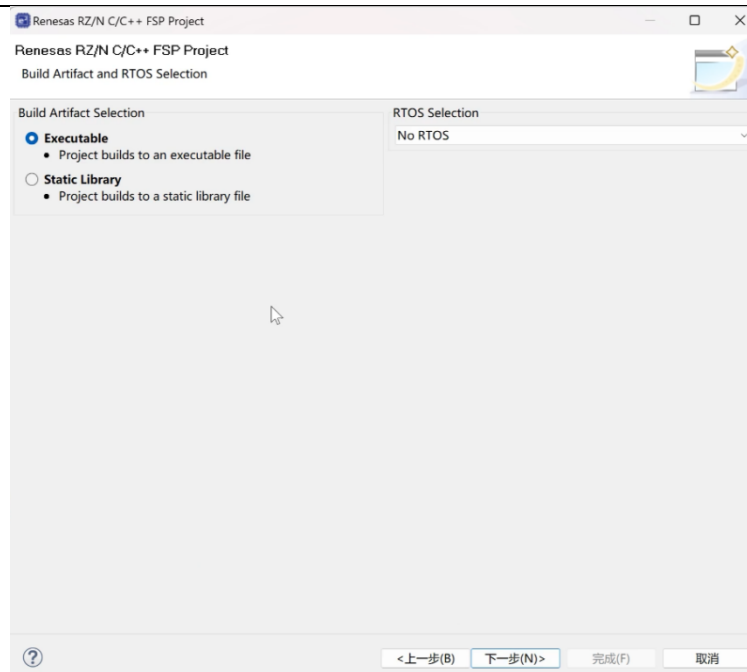
- 选择 Renesas RZ/N C/C++ FSP Project，点击下一步



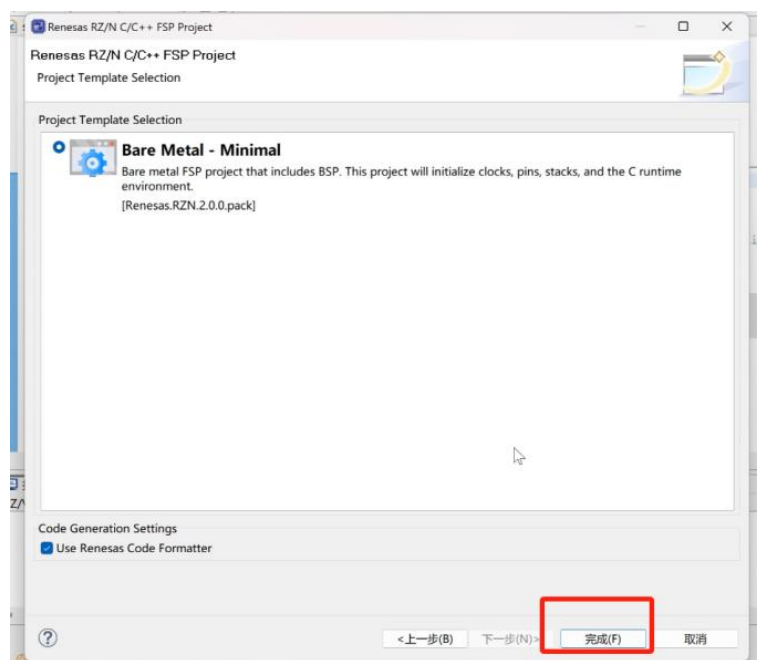
- 新建工程名 RGB，设置文件保存路径，点击下一步
- 选择 RZN2L Custom User Board (xSPI0 x1 boot mode)
- 芯片型号 R9A07G084M04GBG 点击下一步：



- 进入如下界面，点击下一步：



● 点击 完成



● 跳出对话框：选择 打开透视图



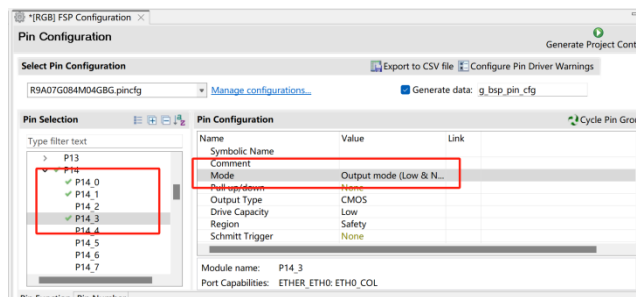
● 进入如下工程界面



3.4

● 引脚配置

配置 P140 P141 P142 为输出模式，初始电平为低。



3.5

● 添加定时器: New stacks--timers--Timer, Compare Match

● 定时器配置

Mode: Periodic

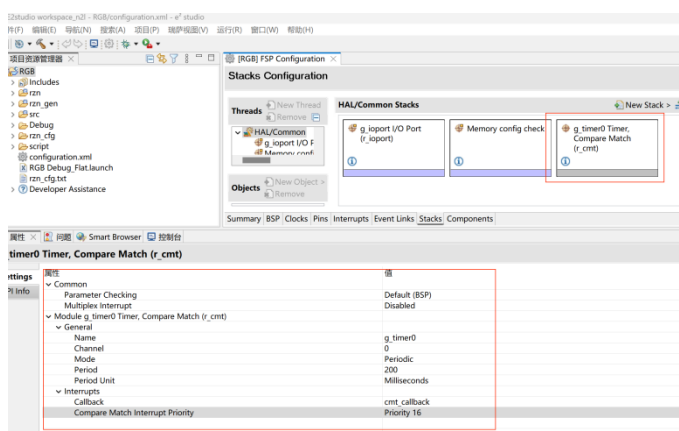
Period: 200

Period Uint: milliseconds

● 定时器中断设置

设置中断优先级，输入 Callback 名称: cmt_callback

说明: callback 函数是用户中断入口函数，需要自己编写，参考 FSP 帮助文档。



设置好后保存，点击右上角 Generate Project Content，即可生成工程。

3.6

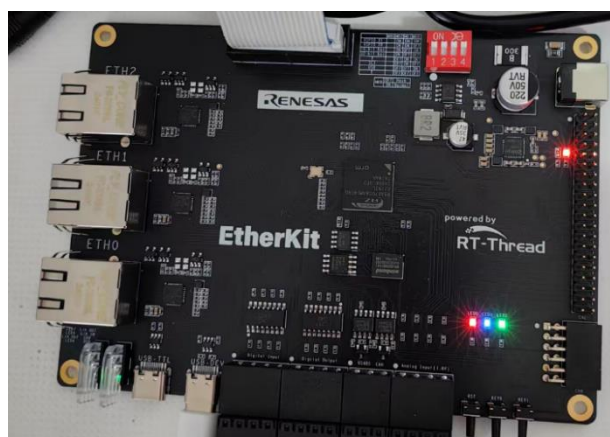
- 进入 hal_entry.c, 编写用户代码。

```

FSP Configuration x hal_entry.c x
bool current_state = 0;
void cmt_callback (timer_callback_args_t * p_args)
{
    if (TIMER_EVENT_CYCLE_END == p_args->event)
    {
        /* Add application code to be called periodically here. */
        current_state = !current_state;
    }
}

/* main() is generated by the FSP Configuration editor and is used to generate threads if an
void hal_entry(void)
{
    /* TODO: add your own code here */
    R_CMT_Open(&g_timer0_ctrl, &g_timer0_cfg); //open timer
    _asm volatile ("cpsie i"); // enable interrupt
    R_CMT_Start(&g_timer0_ctrl); // start timer
    while(1)
    {
        if(current_state==0)
        {
            R_IOPORT_PinWrite(&g_ioport_ctrl, BSP_IO_PORT_14_PIN_0, BSP_IO_LEVEL_LOW);
            R_IOPORT_PinWrite(&g_ioport_ctrl, BSP_IO_PORT_14_PIN_1, BSP_IO_LEVEL_LOW);
            R_IOPORT_PinWrite(&g_ioport_ctrl, BSP_IO_PORT_14_PIN_3, BSP_IO_LEVEL_LOW);
        }
        else
        {
            R_IOPORT_PinWrite(&g_ioport_ctrl, BSP_IO_PORT_14_PIN_0, BSP_IO_LEVEL_HIGH);
            R_IOPORT_PinWrite(&g_ioport_ctrl, BSP_IO_PORT_14_PIN_1, BSP_IO_LEVEL_HIGH);
            R_IOPORT_PinWrite(&g_ioport_ctrl, BSP_IO_PORT_14_PIN_3, BSP_IO_LEVEL_HIGH);
        }
    }
}
    
```

- 编译运行, 观察开发板 LED 灯变化



本节完