RT-Thread Nano 移植

NuMicro® 32 位系列微控制器范例代码介绍

文件信息

应用简述	本范例基于 BSP 中 GPIO_OutputInput 参考程序，介绍如何移植 RT-Thread Nano 并创建 2 个线程。
BSP 版本	M251_M252_M254_M256_M258_Series_BSP_CMSIS_V3.02.006
开发平台	NuMaker-M252SD V1.2

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller and microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1. 概述

本范例基于 BSP 中 GPIO_OutputInput 参考程序，介绍如何移植 RT-Thread Nano 并创建 2 个线程。线程 1 通过 GPIO 口，控制 LED 每 500 ms 闪烁一次；线程 2 通过串口，每 1000 ms 交替输出 “Hello”、“World”字样。

1.1 原理

RT-Thread Nano 是一个精炼的硬实时内核，设计用于内存资源有限的系统。功能包括线程管理、线程间同步与通信、时钟管理、中断管理和内存管理等，这些功能使得 RT-Thread Nano 非常适合需要高性能实时控制的嵌入式系统。支持包括 Arm Cortex M0/M23/M3/M4/M7 以及 RISC-V 等多种 CPU 架构，对内存资源的占用极小，最小只需要 3 KB Flash 和 1.2 KB RAM 情况下就可以运行。

本范例使用 M252SD2AE 芯片，包括 64 KB Flash，12 KB RAM，介绍如何移植 RT-Thread Nano 并创建管理 2 个线程。

1.2 执行结果

本范例代码执行编译和烧录后，LED(PB14) 每 500 ms 闪烁一次，如图 1-1 所示。

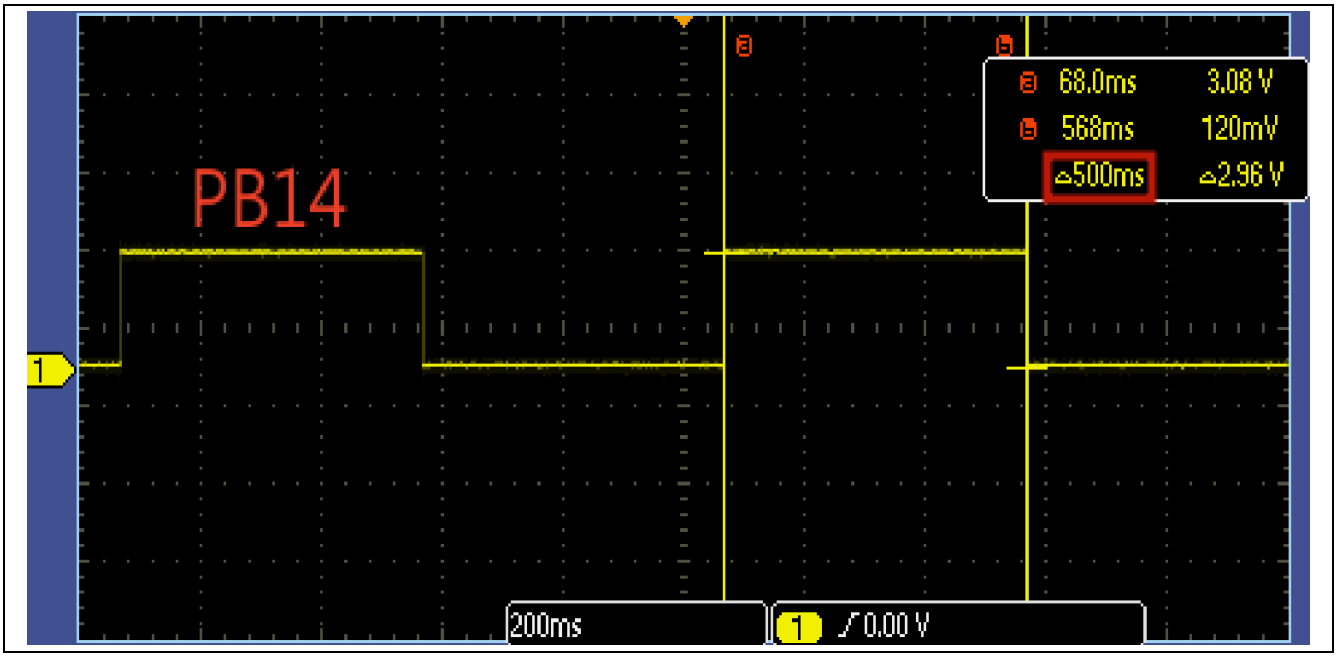


图 1-1 执行编译和烧录结果

串口打印消息如图 1-2 所示。

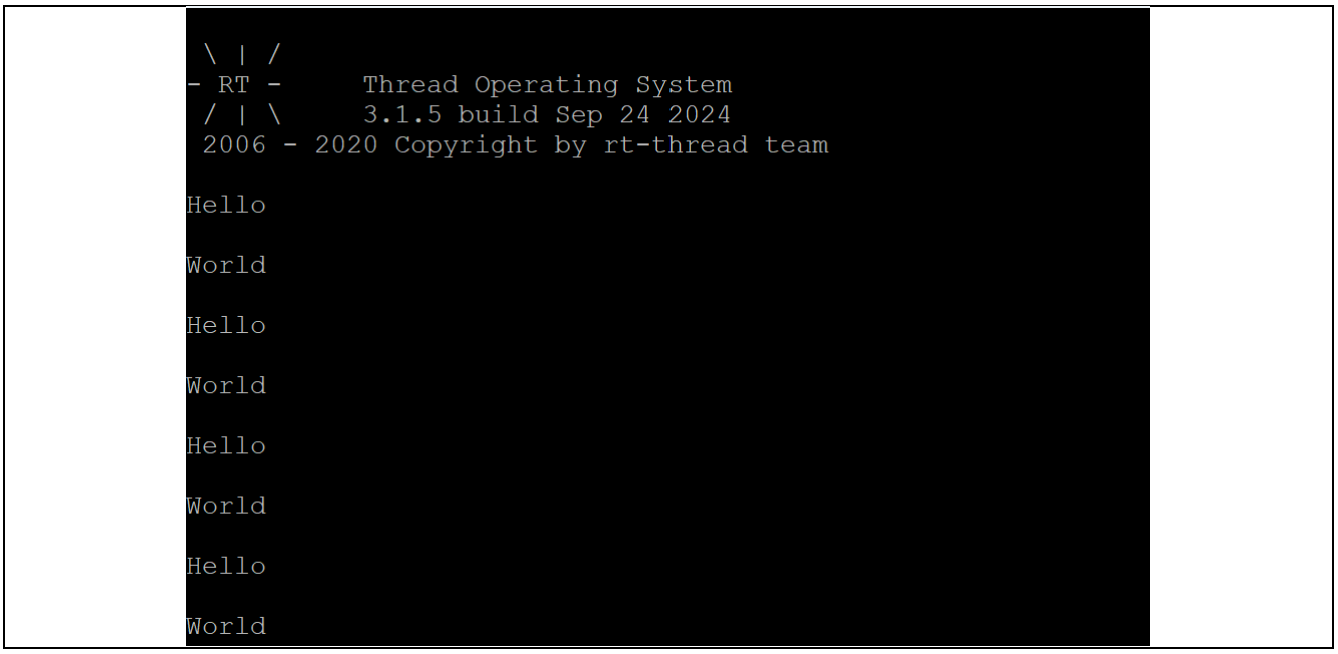


图 1-2 串口打印消息

2. 代码描述

2.1 下载 RT-Thread Nano 源码

单击下面链接，在官网页面下载 RT-Thread Nano 源码。

https://www.rt-thread.org/document/site/#/rt-thread-version/rt-thread-nano/an0038-nano-introduction

源码解压后如下。

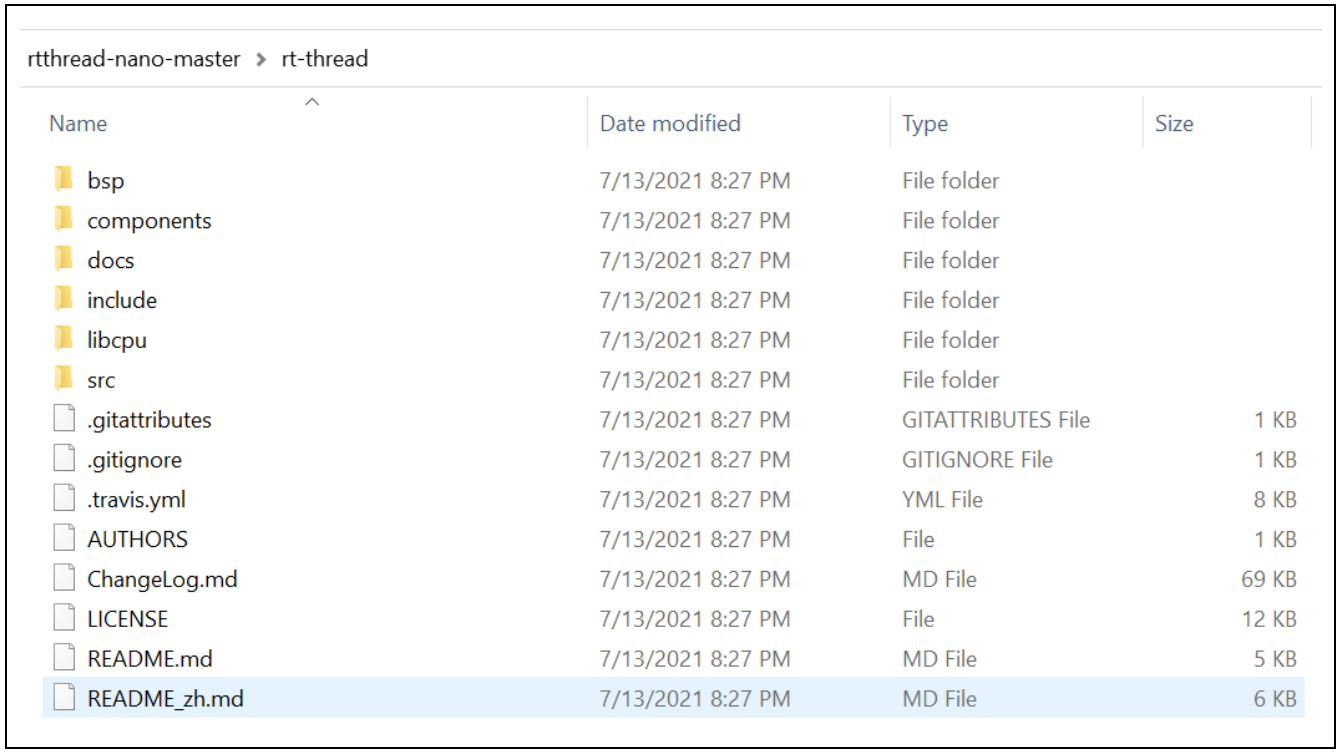


图 2-1 rt-thread 文件结构

2.2 拷贝 rt-thread 文件到 BSP 路径下

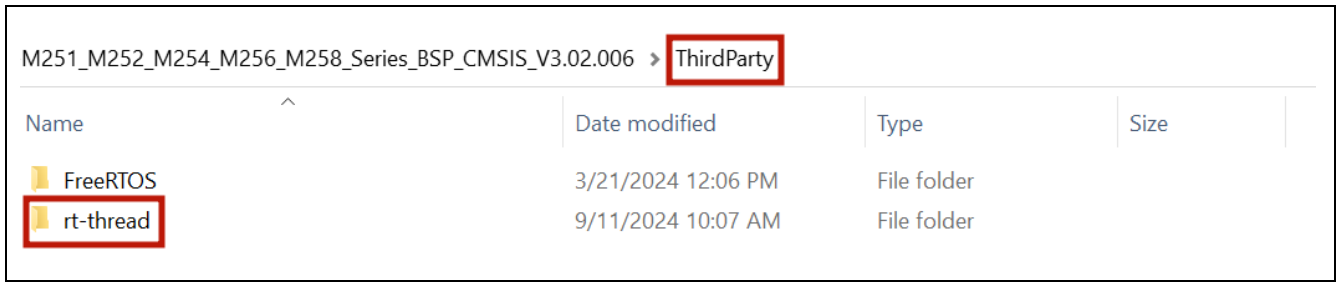


图 2-2 rt-thread 存放路径

2.3 新建 rt-thread 组

在工程中新建 rt-thread 组并添加相关源码文件。

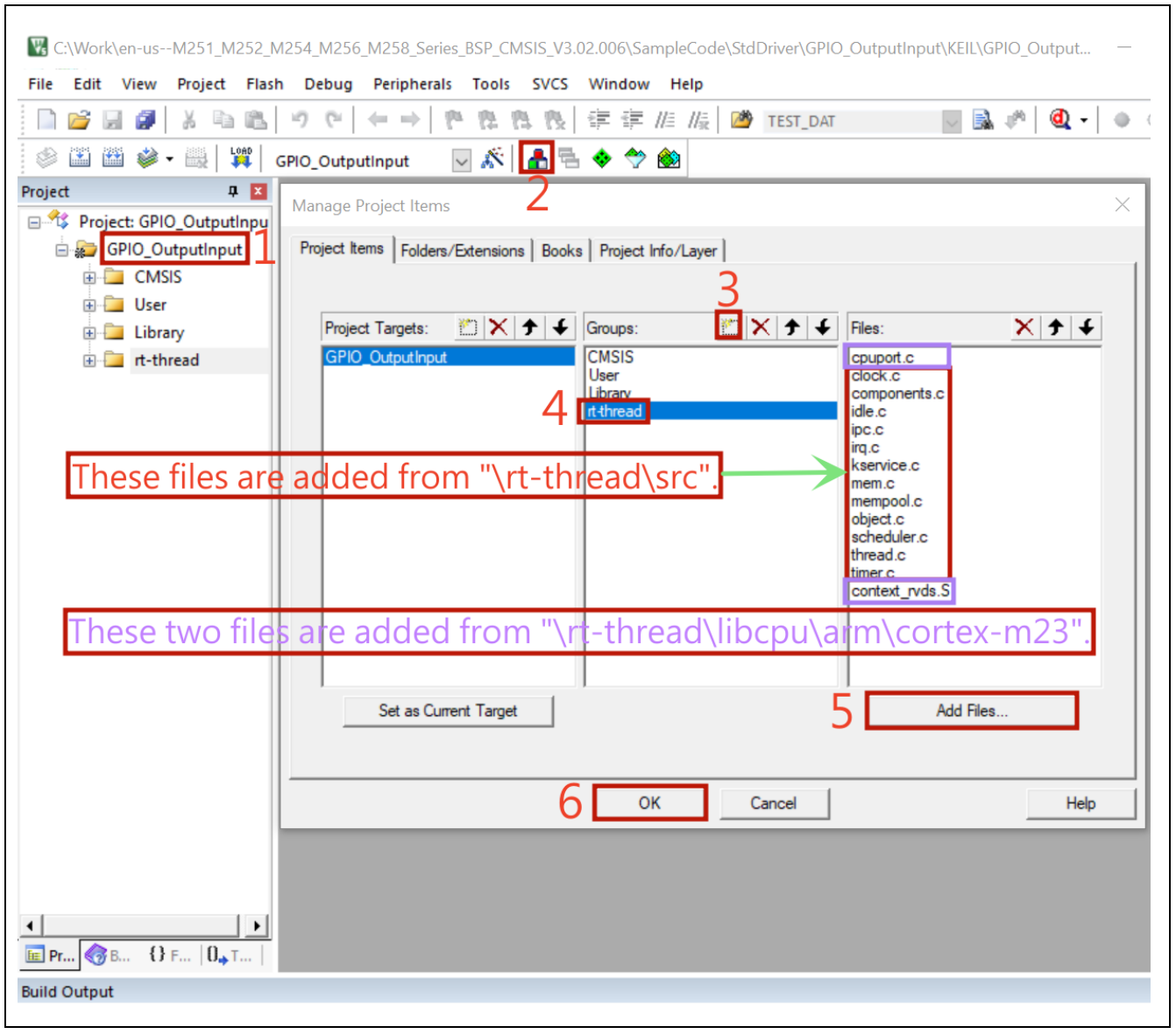


图 2-3 新建 rt-thread 组并添加文件

2.4 复制 Board.c 和 rtconfig.h 文件

将 “\rt-thread\bsp_template\” 路径下 *Board.c* 和 *rtconfig.h* 文件复制到 “\GPIO_OutputInput\ ” 路径，然后将它们添加到 User 组。

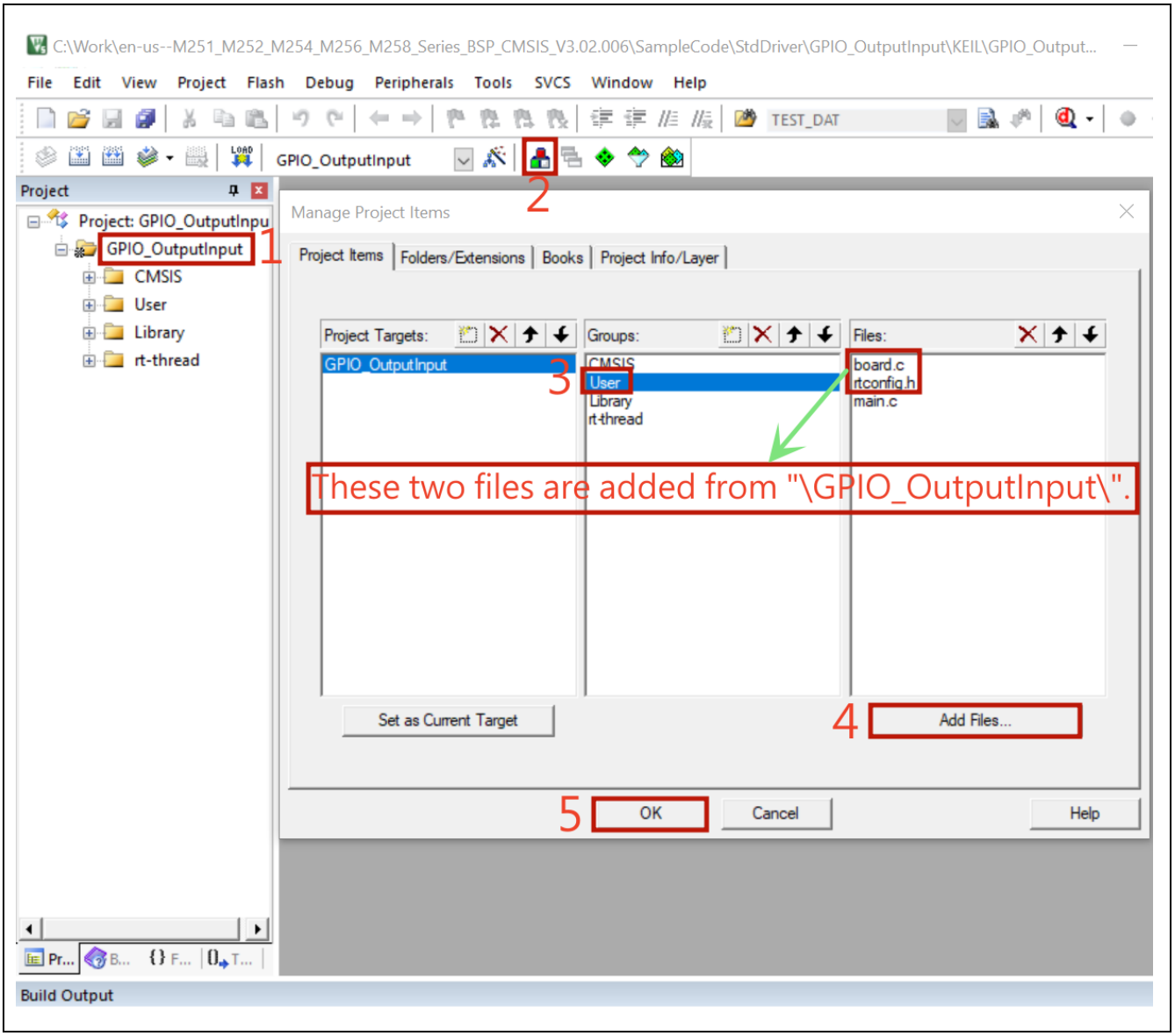


图 2-4 添加文件到 User 组

2.5 修改工程选项 Include Paths

在工程选项中增加 rt-thread 文件 Include Paths。

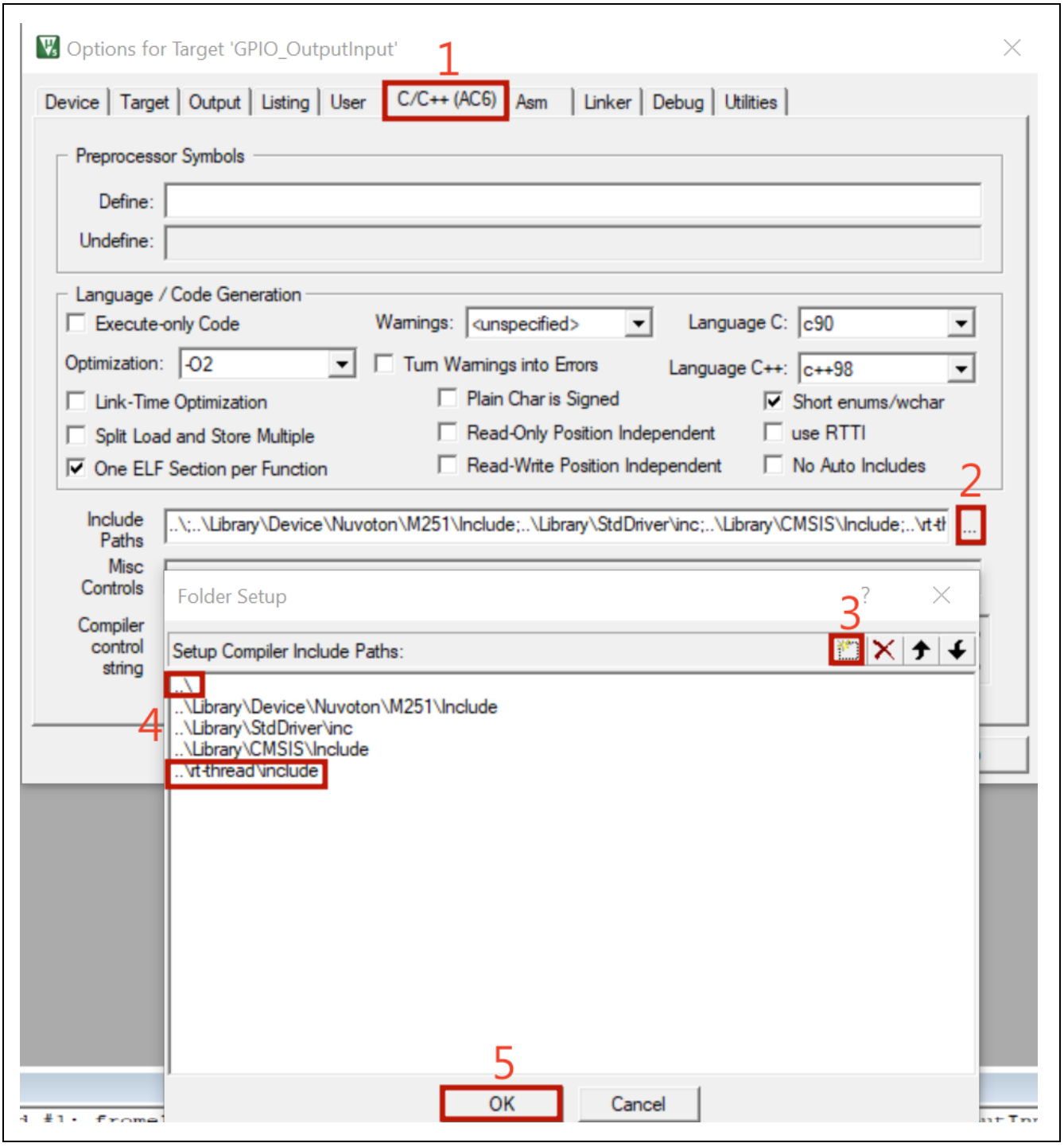


图 2-5 增加 rt-thread 文件 Include Paths

2.6 修改 *board.c* 文件

文件中新增包含 *NuMicro.h*、*stdio.h*。

```
#include <rthw.h>
#include <rtthread.h>
```

```
#include <stdio.h>
#include "NuMicro.h"
```

因为目标芯片 SRAM 最大 12 KB，需要将 #define RT_HEAP_SIZE (15*1024) 修改为 #define RT_HEAP_SIZE (4*1024)。

```
/*
 * Please modify RT_HEAP_SIZE if you enable RT_USING_HEAP
 * the RT_HEAP_SIZE max value = (sram size - ZI size), 1024 means 1024 bytes
 */
#define RT_HEAP_SIZE (4*1024)
```

将原数组及函数进行如下重定义，避免 non-word-aligned 风险。

```
static rt_uint32_t rt_heap[RT_HEAP_SIZE/sizeof(rt_uint32_t)];

RT_WEAK void *rt_heap_begin_get(void)
{
    return (void *)rt_heap;
}

RT_WEAK void *rt_heap_end_get(void)
{
    return (void *)((rt_uint32_t)rt_heap + sizeof(rt_heap));
}
```

在 rt_os_tick_callback 函数下面新增 SysTick_Handler 函数。

```
void SysTick_Handler(void)
{
    rt_os_tick_callback();
}
```

在函数 rt_hw_board_init 中，先注释第一行，然后增加系统初始化、系统内核时钟更新、配置 SysTick 函数。

```
    //#error "TODO 1: OS Tick Configuration."
    /*
     * TODO 1: OS Tick Configuration
     * Enable the hardware timer and call the rt_os_tick_callback function
     * periodically with the frequency RT_TICK_PER_SECOND.
     */
    /* System Init */
    SYS_UnlockReg();
    SYS_Init();
    SYS_LockReg();

    /* System Clock Update */
    SystemCoreClockUpdate();

    /* System Tick Configuration */
    SysTick_Config(SystemCoreClock / RT_TICK_PER_SECOND);
```

将 *main.c* 文件中 SYS_Init 函数复制到 *board.c* 文件。

在函数 uart_init 中，先注释第一行，然后增加 UART 复位函数及波特率设置函数。

```
static int uart_init(void)
{
    //#error "TODO 2: Enable the hardware uart and config baudrate."
    /* Reset UART module */
    SYS_ResetModule(UART0_RST);

    /* Configure UART0 and set UART0 baud rate */
    UART_Open(UART0, 115200);

    return 0;
}
```

在函数 rt_hw_console_output 中，先注释第一行，然后编写 printf 函数。

```
void rt_hw_console_output(const char *str)
{
    //#error "TODO 3: Output the string 'str' through the uart."
    printf("%s", str);
}
```

2.7 修改 main.c 文件

文件中新增加包含 *rtthread.h*。

```
#include "stdio.h"
#include "NuMicro.h"
#include <rtthread.h>
```

定义 2 个线程相关全部变量。

```
static struct rt_thread blink_thread;
static char blink_thread_stack[256];
static struct rt_thread print_thread;
static char print_thread_stack[256];
```

线程 1 函数设计。

```
void blink_thread_entry(void *parameter)
{
    while (1)
    {
        PB14 = 1;
        rt_thread_mdelay(500);
        PB14 = 0;
        rt_thread_mdelay(500);
    }
}
```

线程 2 函数设计。

```
void print_thread_entry(void *parameter)
{
    while (1)
    {
        rt_kprintf("\nHello\n");
        rt_thread_mdelay(1000);
        rt_kprintf("\nWorld\n");
        rt_thread_mdelay(1000);
    }
}
```

删除 SYS_Init、UART0_Init 函数。

主函数 main 重新设计。

```
int main(void)
{
    rt_err_t ret;

    /* Configure PB.14 as Output mode */
    GPIO_SetMode(PB, BIT14, GPIO_MODE_OUTPUT);

    /* Create thread 1 */
    ret = rt_thread_init(&blink_thread,
                         "led blink",
                         blink_thread_entry,
                         RT_NULL,
                         &blink_thread_stack[0],
                         sizeof(blink_thread_stack),
                         RT_THREAD_PRIORITY_MAX-2,
                         20);

    if (ret == RT_EOK)
    {
        /* After the thread 1 is created successfully, start the thread 1 */
        rt_thread_startup(&blink_thread);
    }

    /* Create thread 2*/
    ret = rt_thread_init(&print_thread,
                         "print info",
                         print_thread_entry,
                         RT_NULL,
                         &print_thread_stack[0],
                         sizeof(print_thread_stack),
                         RT_THREAD_PRIORITY_MAX-3,
                         20);

    if (ret == RT_EOK)
    {
        /* After the thread 2 is created successfully, start the thread 2 */
        rt_thread_startup(&print_thread);
```

```
        }

        return 0;
    }
```

2.8 修改 *rtconfig.h* 文件

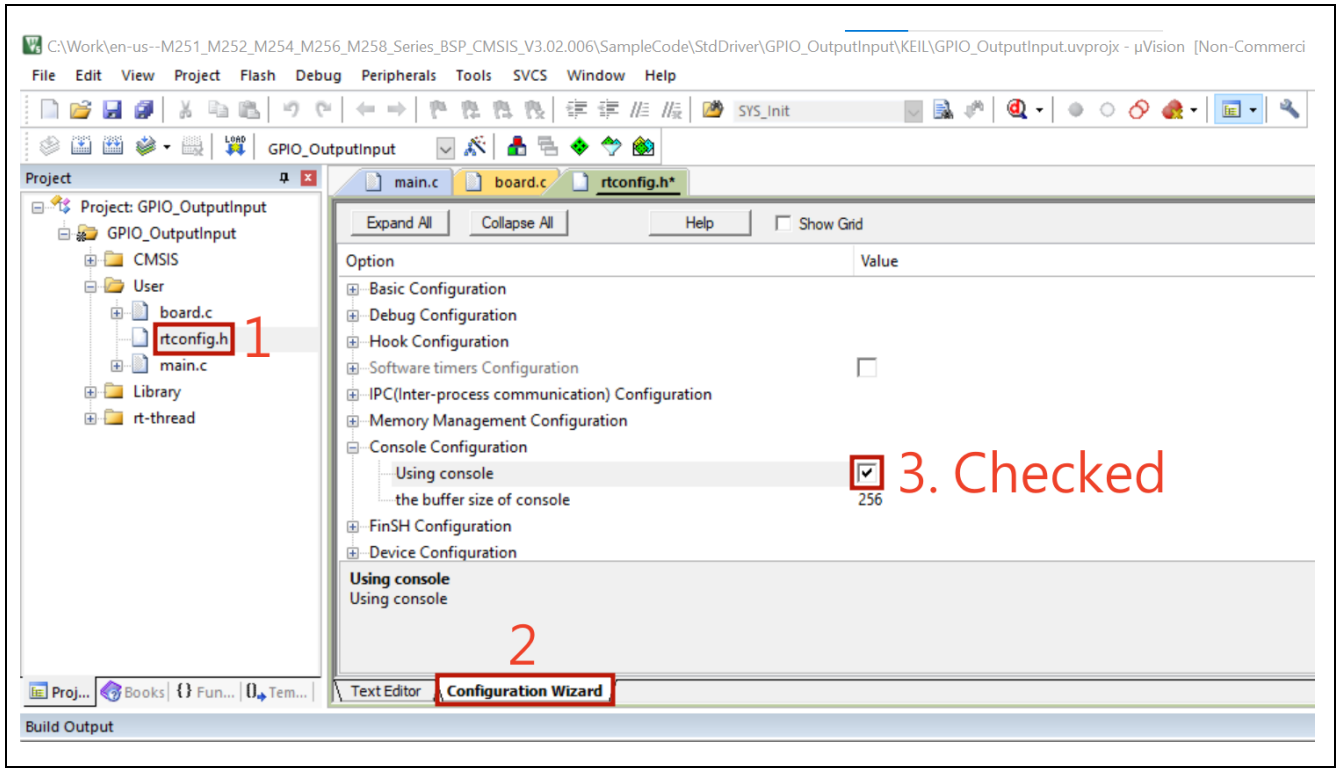
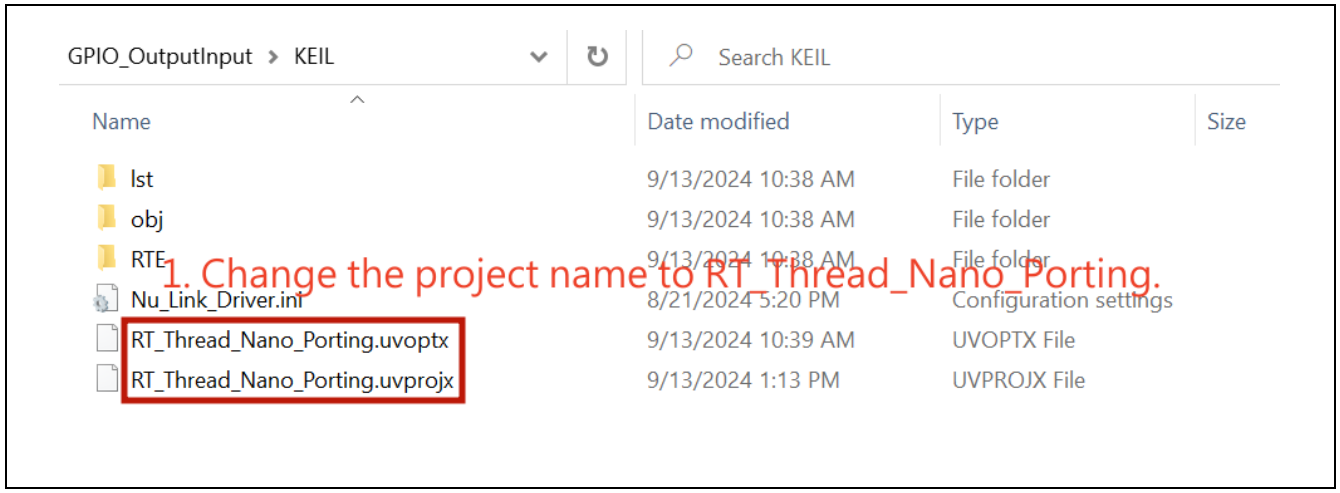


图 2-6 配置 console

2.9 修改工程名

修改工程名为 RT_Thread_Nano_Porting



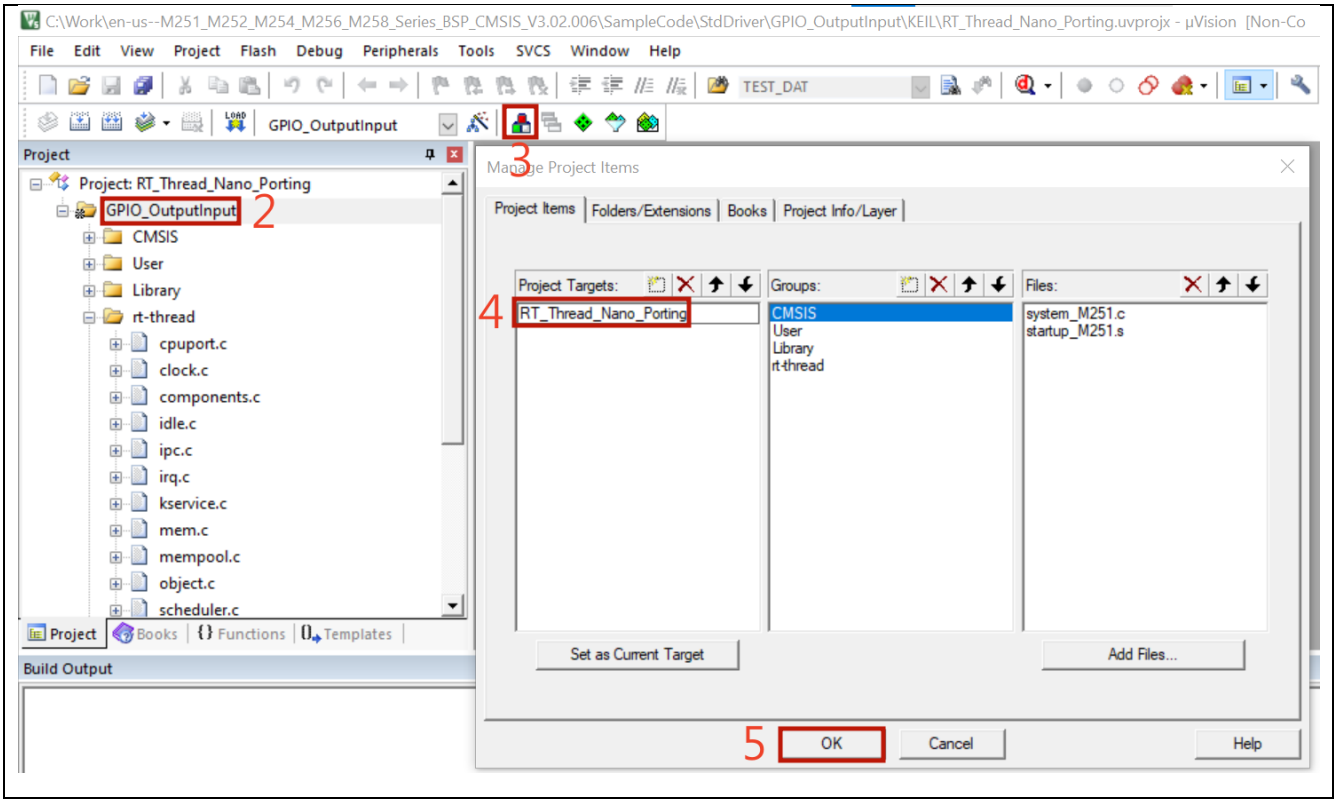


图 2-7 修改工程名称

2.10 修改各文件名称

对各文件名称进行相应修改，并删除非必要文件。

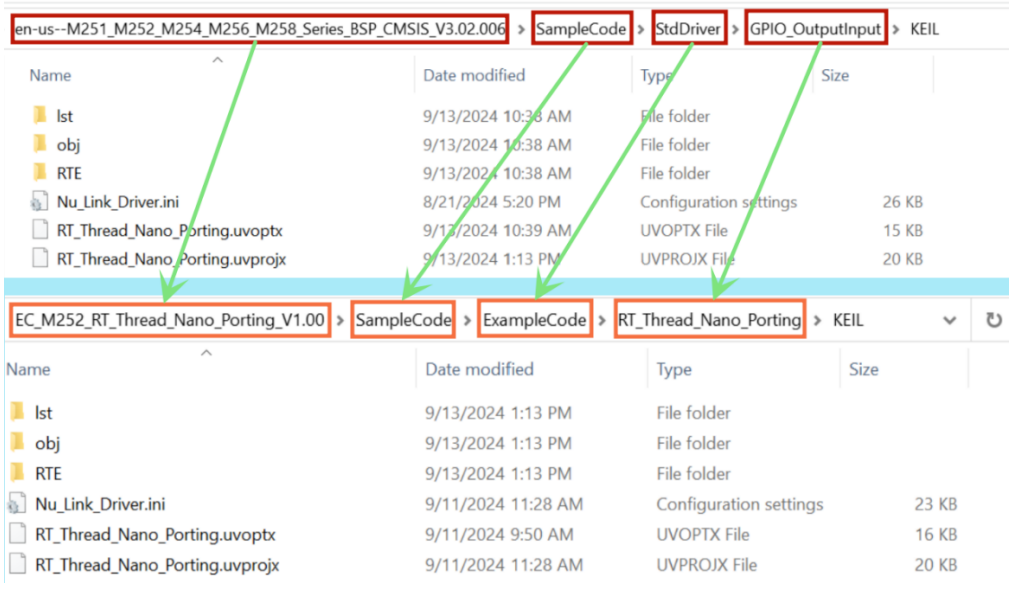


图 2-8 修改各文件名称

3. 软件与硬件需求

3.1 软件需求

- BSP 版本
  - M251_M252_M254_M256_M258_Series_BSP_CMSIS_V3.02.006
- IDE 版本
  - Keil uVersion 5.36

3.2 硬件需求

- 电路组件
  - NuMaker-M252SD V1.2
- 线路示意图
  - 通过设置 Nu-Link2-Me 上的拨码开关，使能其 USB 转串口功能。

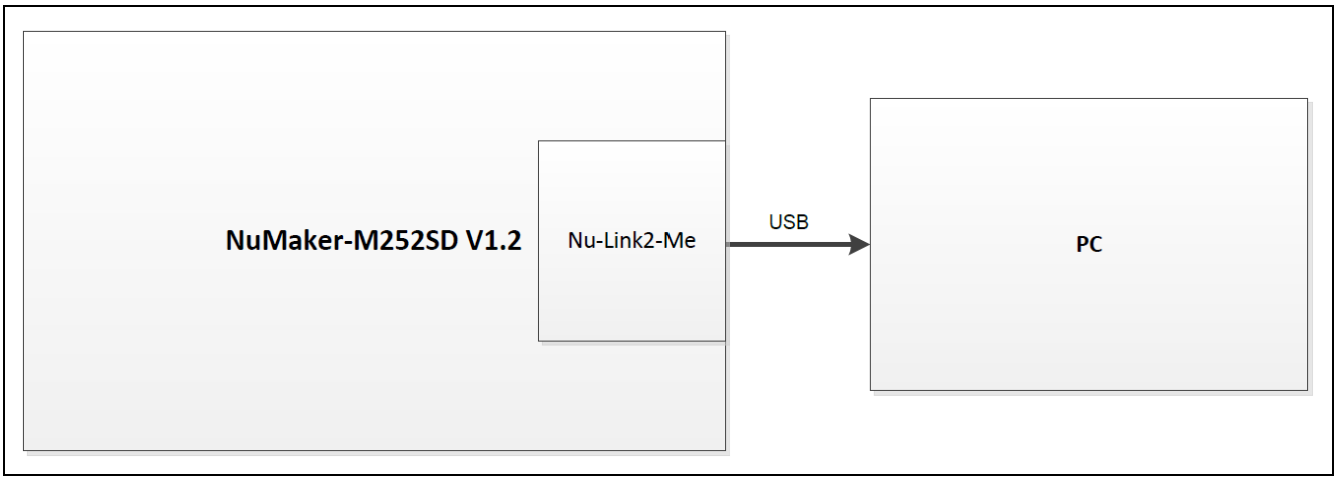


图 3-1 线路示意图

4. 目录信息

EC_252_RT_Thread_Nano_Porting_V1.00	
Library	Sample code header and source files
CMSIS	Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp.
Device	CMSIS compliant device header file
StdDriver	All peripheral driver header and source files
ThirdParty	
rt-thread	Source file of RT-Thread Nano code
SampleCode	
ExampleCode	
Project	Source file of example code

图 4-1 目录信息

5. 范例程序执行

1. 根据目录信息章节进入 ExampleCode 路径中的 KEIL 文件夹，双击 RT_Thread_Nano_Porting.uvprojx。
2. 进入编译模式界面
   - 编译
   - 下载代码至内存
   - 进入 / 离开仿真模式
3. 进入仿真模式界面
   - 执行代码

6. 修订纪录

Date	Revision	Description
2024.10.08	1.00	初始发布。

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.