RT-Thread Nano 移植

NuMicro® 32 位元系列微控制器範例代碼介紹

文件資訊

應用簡述	本範例基於 BSP 中 GPIO_OutputInput 參考程式，介紹如何移植 RT-Thread Nano 並創建 2 個執行緒。
BSP 版本	M251_M252_M254_M256_M258_Series_BSP_CMSIS_V3.02.006
開發平台	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

本範例基於 BSP 中 GPIO_OutputInput 參考程式，介紹如何移植 RT-Thread Nano 並創建 2 個執行緒。執行緒 1 通過 GPIO 口，控制 LED 每 500 ms 閃爍一次；執行緒 2 通過 COM 埠，每 1000 ms 交替輸出“Hello”、“World”字樣。

1.1 原理

RT-Thread Nano 是一個精煉的硬即時內核，設計用於記憶體資源有限的系統。功能包括執行執行緒管理、執行緒間同步與通信、時鐘管理、中斷管理和記憶體管理等，這些功能使得 RT-Thread Nano 非常適合需要高性能即時控制的嵌入式系統。支持包括 Arm Cortex M0/M23/M3/M4/M7 以及 RISC-V 等多種 CPU 架構，對記憶體資源的佔用極小，最小只需要 3 K 位元組 Flash 和 1.2 K 位元組 RAM 情況下就可以運行。

本範例使用 M252SD2AE 晶片，包括 64 K 位元組 Flash，12 K 位元組 RAM，介紹如何移植 RT-Thread Nano 並創建管理 2 個執行緒。

1.2 執行結果

本範例代碼執行編譯和燒錄後，LED(PB14) 每 500 ms 閃爍一次，如圖 1-1 所示。

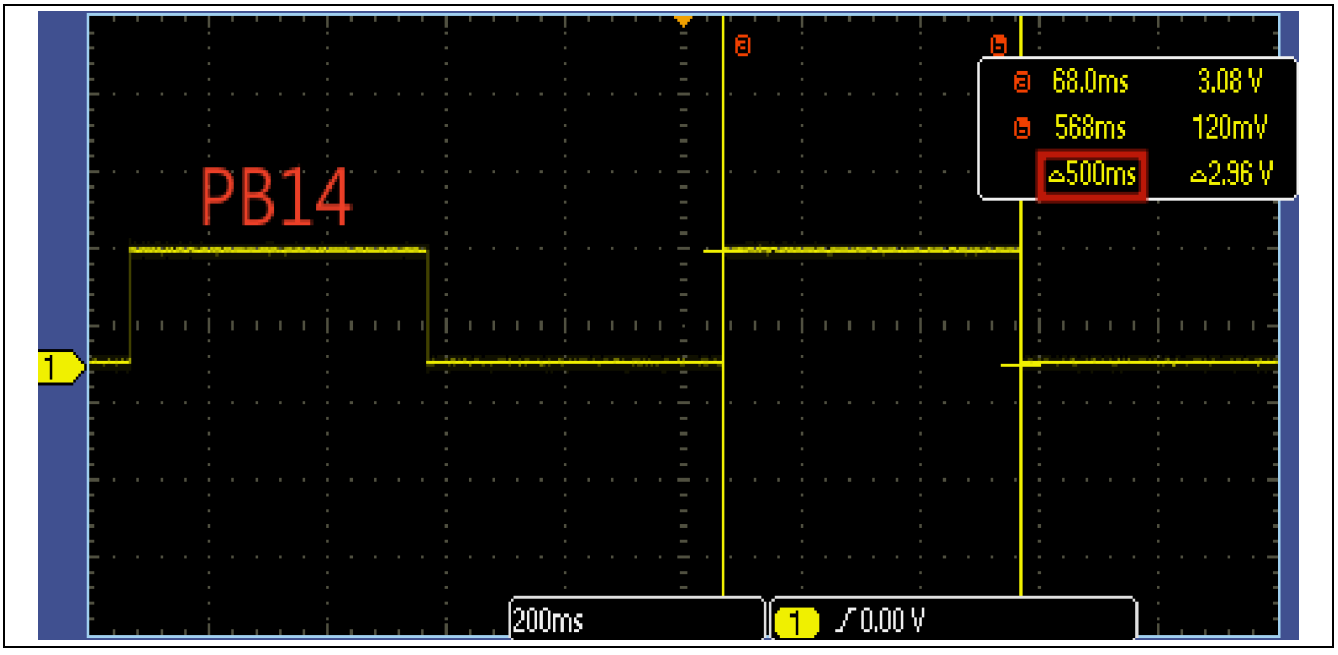


圖 1-1 執行編譯和燒錄結果

COM 埠列印消息如圖 1-2 所示。

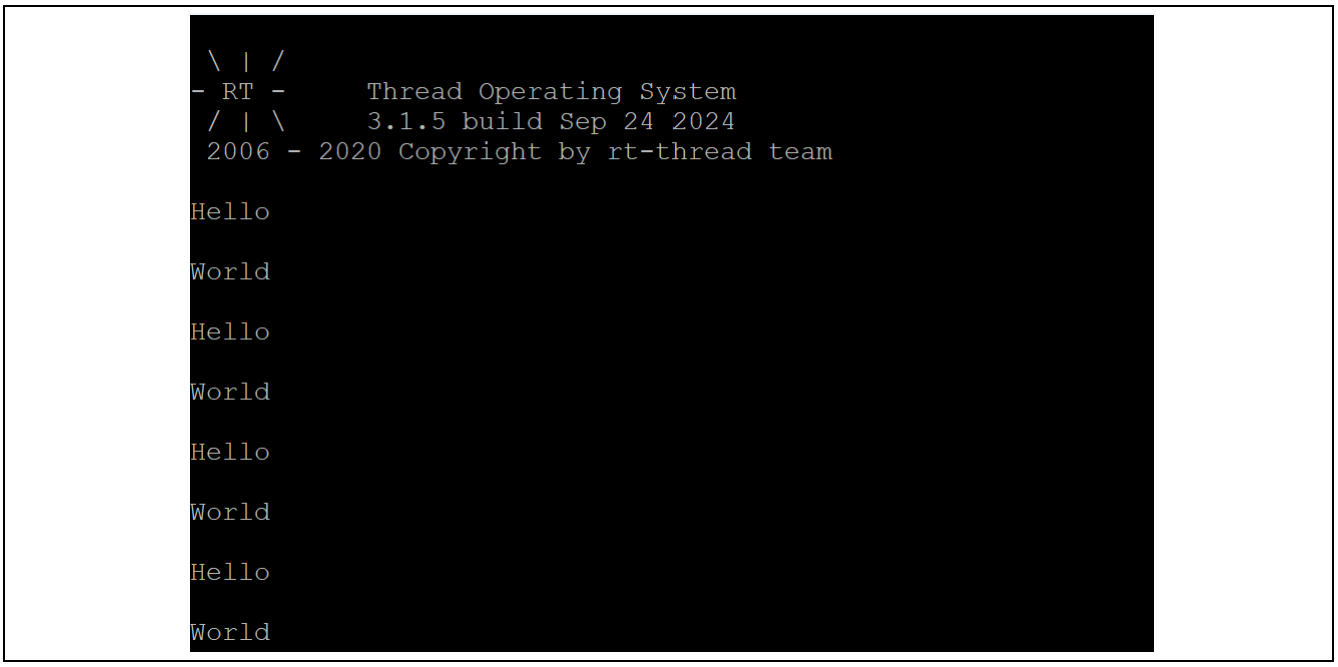


圖 1-2COM 埠列印消息

2. 代碼描述

2.1 下載 RT-Thread Nano 程式碼

單擊下面鏈接，在官網頁面下載 RT-Thread Nano 程式碼。

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

程式碼解壓後如下。

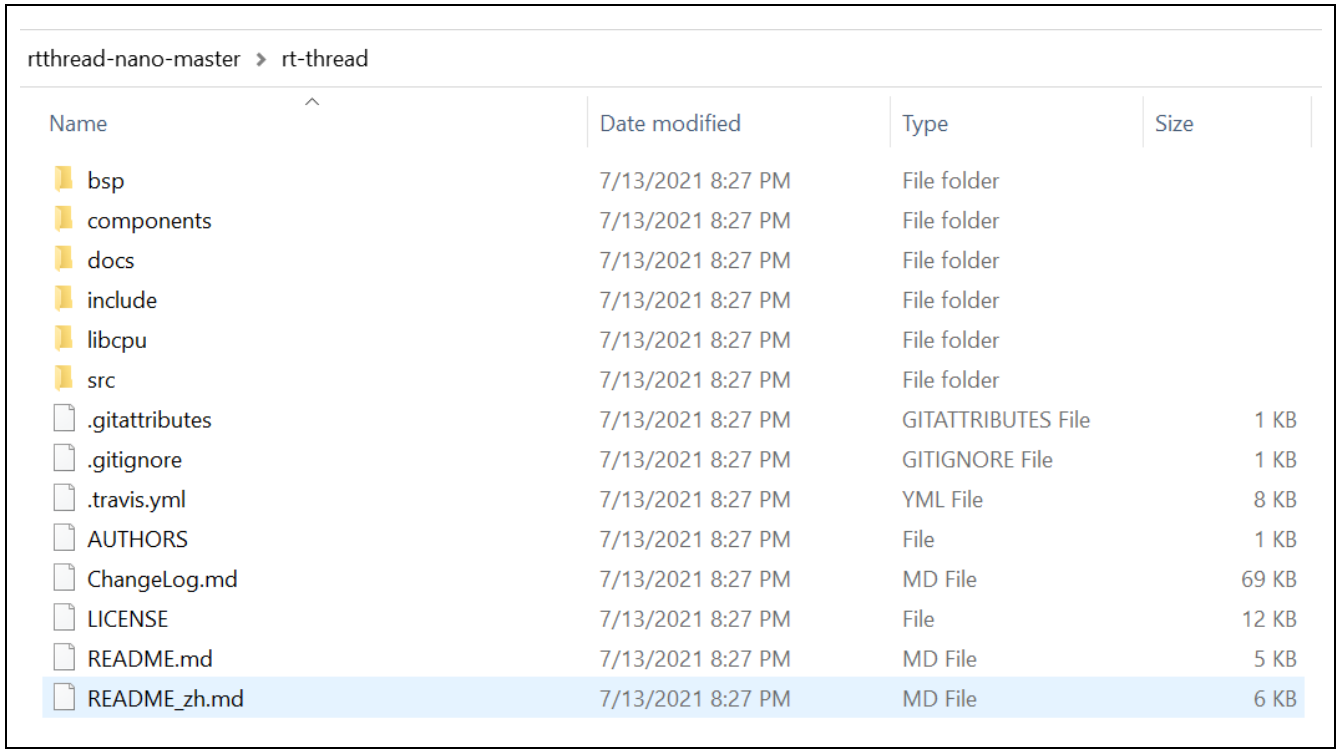


圖 2-1 rt-thread 檔案結構

2.2 拷貝 rt-thread 文件到 BSP 路徑下

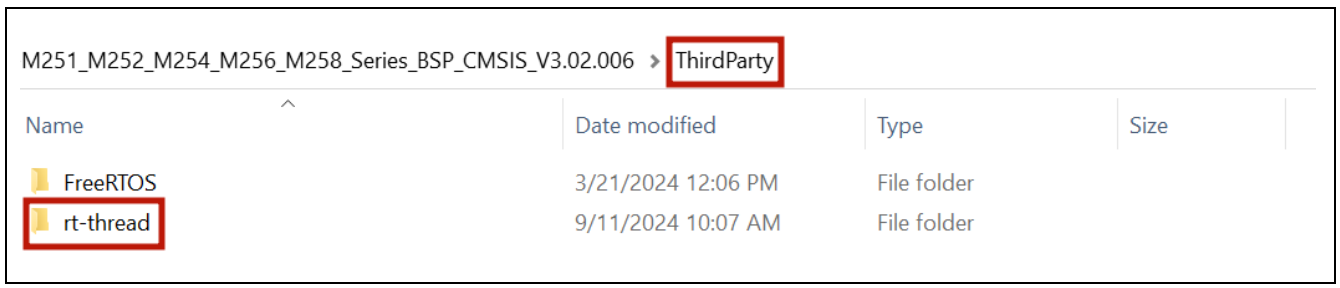


圖 2-2 rt-thread 存放路徑

2.3 新建 rt-thread 組

在專案中新建 rt-thread 組並添加相關源碼檔。

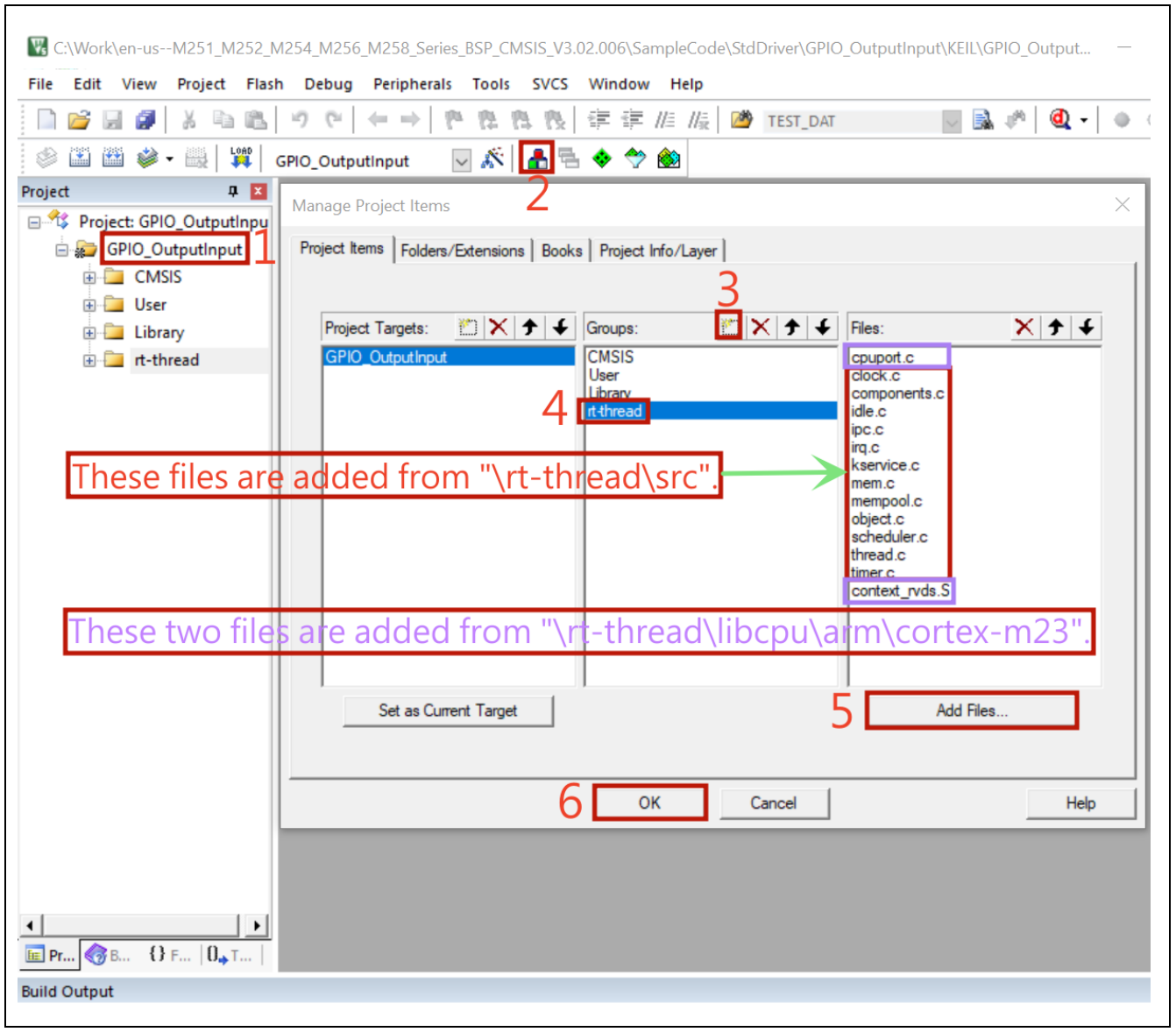


圖 2-3 新建 rt-thread 組並添加源碼檔

2.4 複製 *Board.c* 和 *rtconfig.h* 文件

先將 “\rt-thread\bsp_template\” 路徑下 *Board.c* 和 *rtconfig.h* 文件複製到 “\GPIO_OutputInput\” 路徑，然後將它們添加到 User 組。

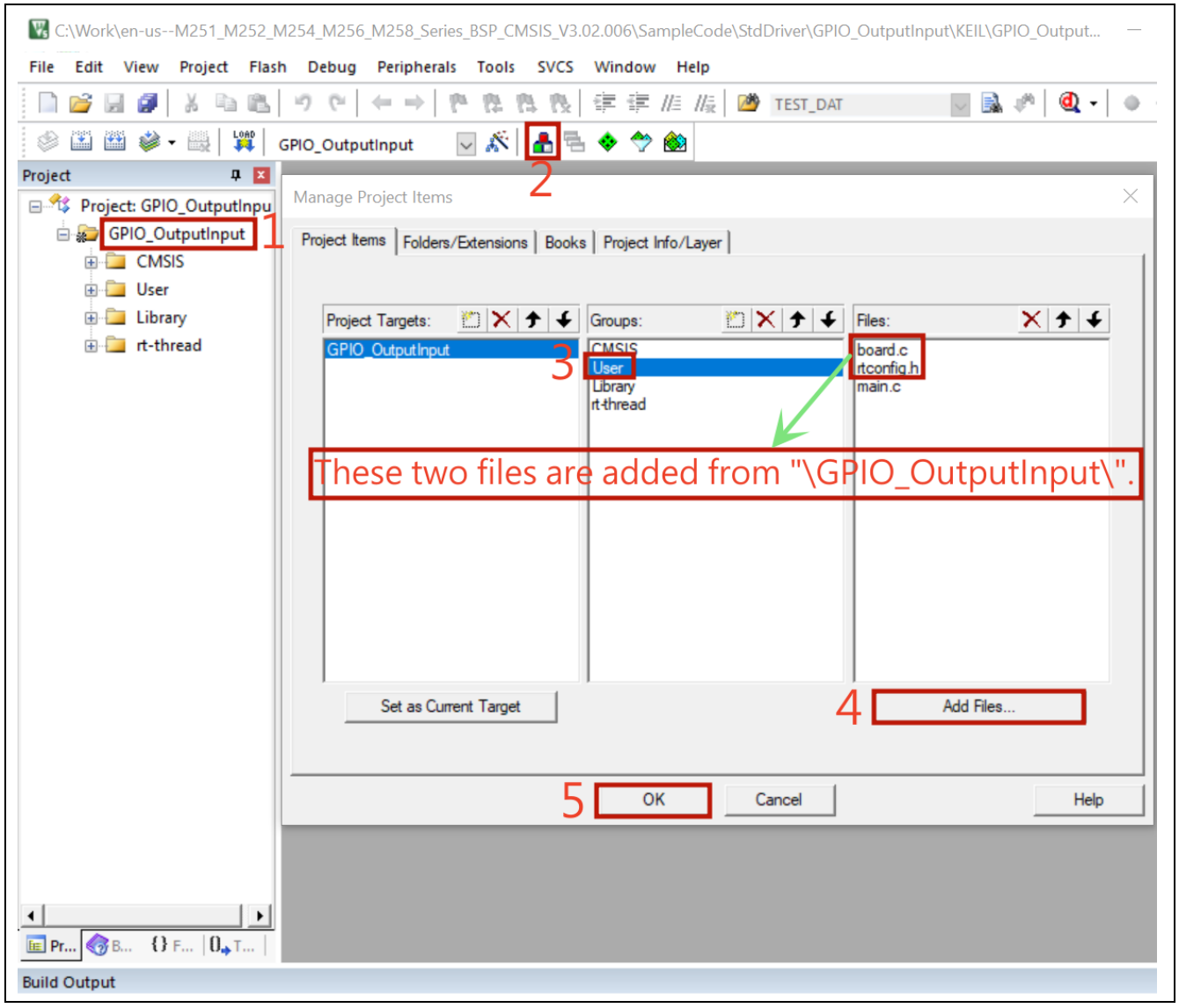


圖 2-4 添加文件到 User 組

2.5 修改專案選項 Include Paths

在專案選項中增加 rt-thread 文件 Include Paths。

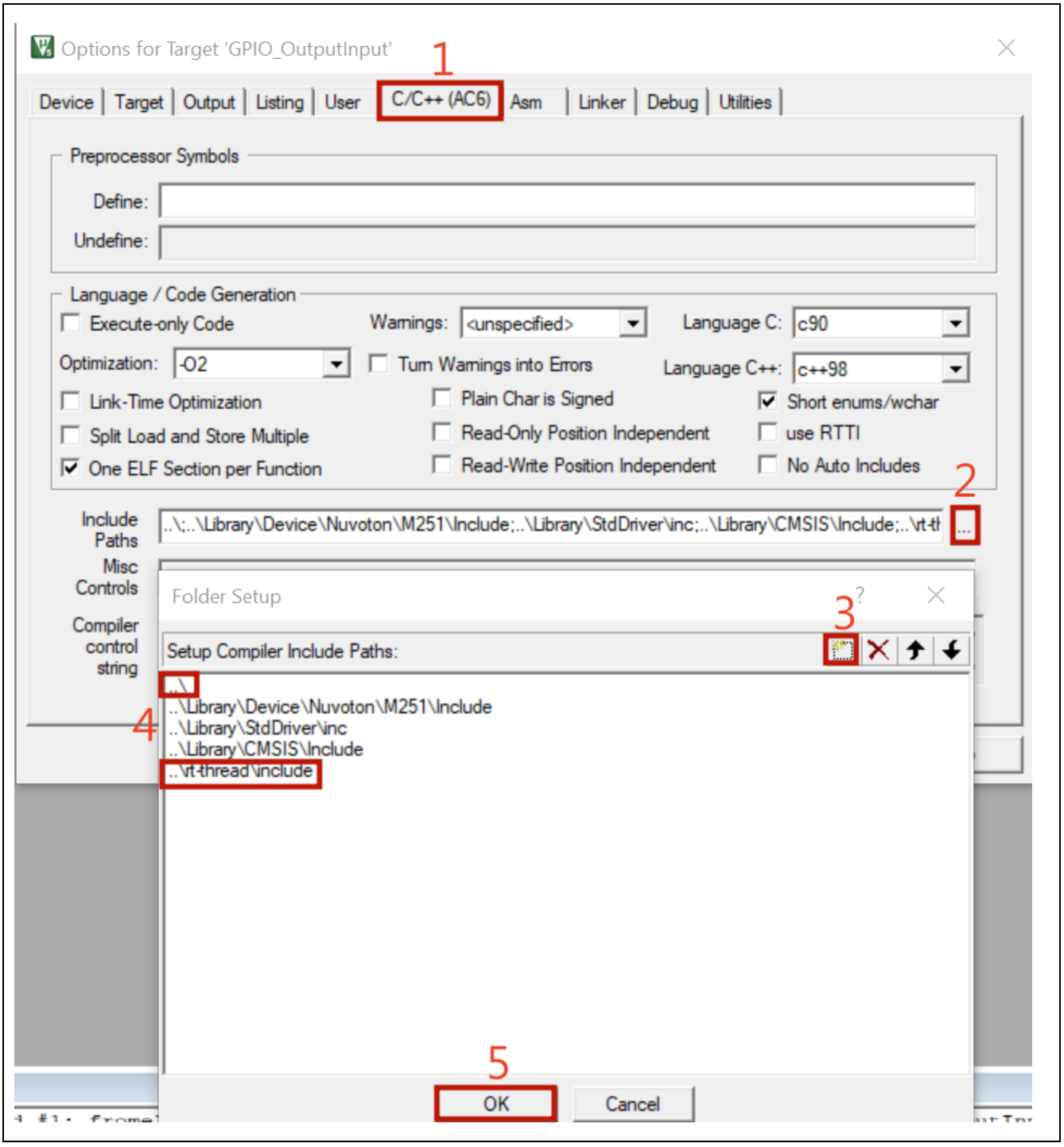


圖 2-5 增加 rt-thread 文件 Include Paths

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"
```

因為目標晶片 SRAM 最大 12 KB，需要將 #define RT_HEAP_SIZE (15*1024) 修改為 #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)
```

將原陣列及函式進行如下重定義，避免 non-word-aligned 風險。

```
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 函式下面新增 SysTick_Handler 函式。

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

在函式 rt_hw_board_init 中，先注釋第一行，然後增加系統初始化、系統內核時鐘更新、配置 SysTick 函式。

```
    //#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);
```

將 *main.c* 檔中 SYS_Init 函式複製到 *board.c* 文件。

在函式 uart_init 中，先注釋第一行，然後增加 UART 復位函式及傳輸速率設置函式。

```
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;
}
```

在函式 rt_hw_console_output 中，先注釋第一行，然後編寫 printf 函式。

```
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>
```

定義 2 個執行緒相關全部變數。

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

執行緒 1 函式設計。

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

執行緒 2 函式設計。

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

刪除 SYS_Init、UART0_Init 函式。

主函式 main 重新設計。

```
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

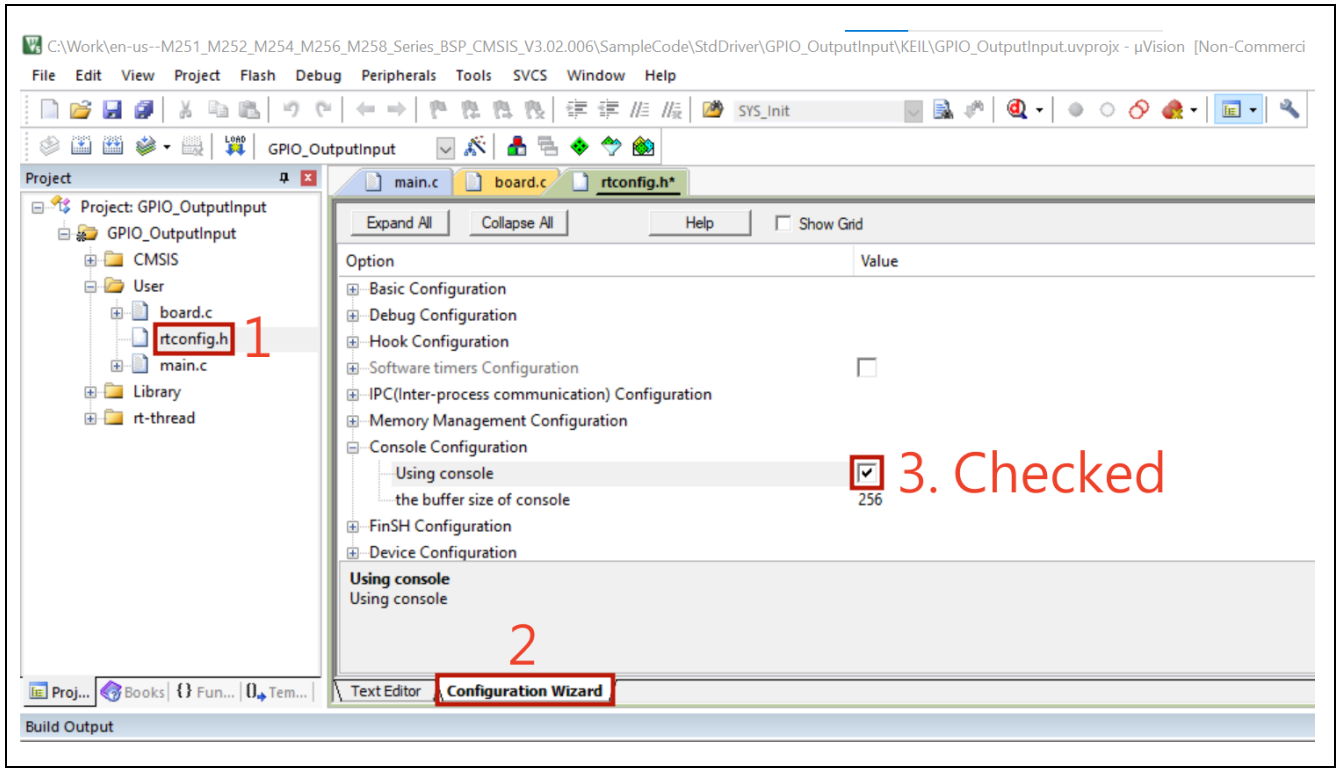
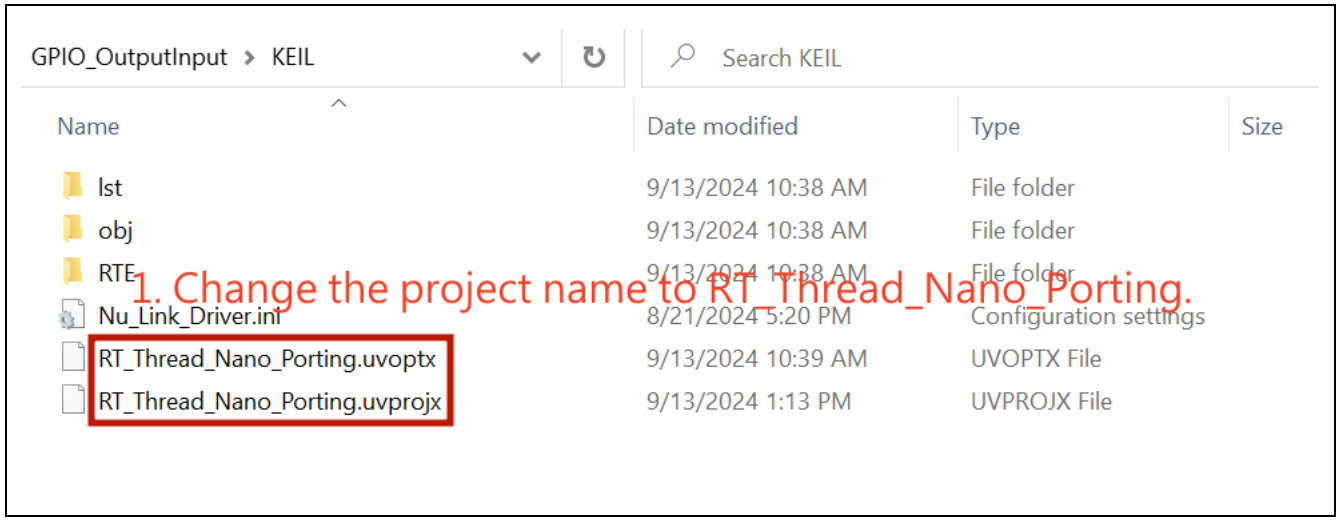


圖 2-6 配置 console

2.9 修改專案名

修改專案名為 RT_Thread_Nano_Porting。



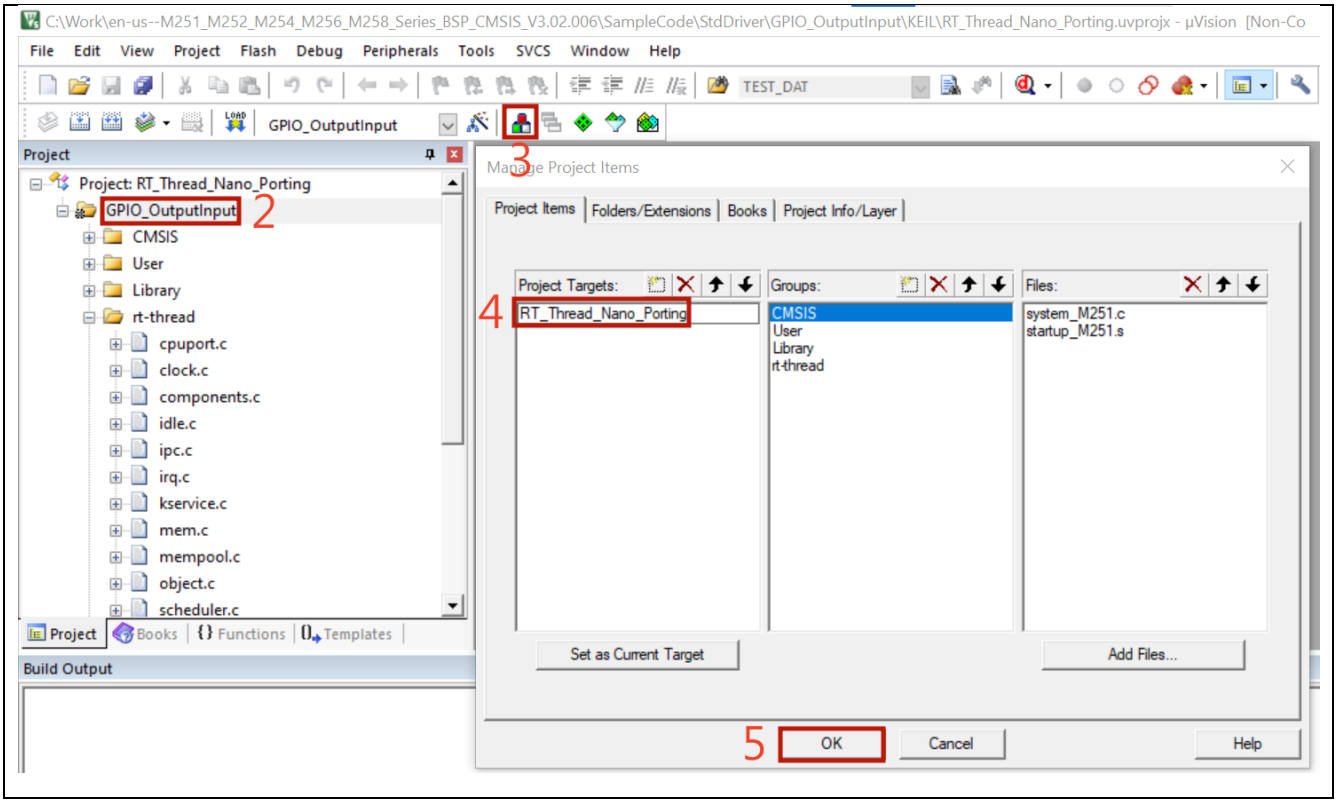


圖 2-7 修改專案名稱

2.10 修改各檔案名稱

對各檔案名稱進行相應修改，並刪除非必要文件。

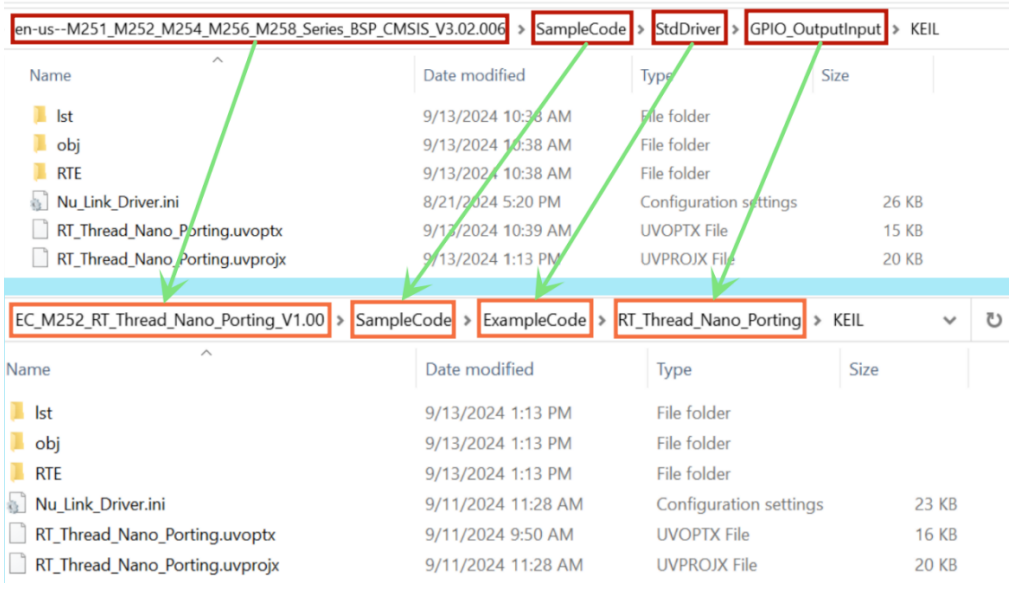


圖 2-8 修改各檔案名稱

3. 軟體與硬體需求

3.1 軟體需求

- BSP 版本
  - M251_M252_M254_M256_M258_Series_BSP_CMSIS_V3.02.00
- IDE 版本
  - Keil uVersion 5.36

3.2 硬體需求

- 電路元件
  - NuMaker-M252SD V1.2
- 線路示意圖
  - 通過設置 Nu-Link2-Me 上的指撥開關，啟用其 USB 轉 COM 埠功能。

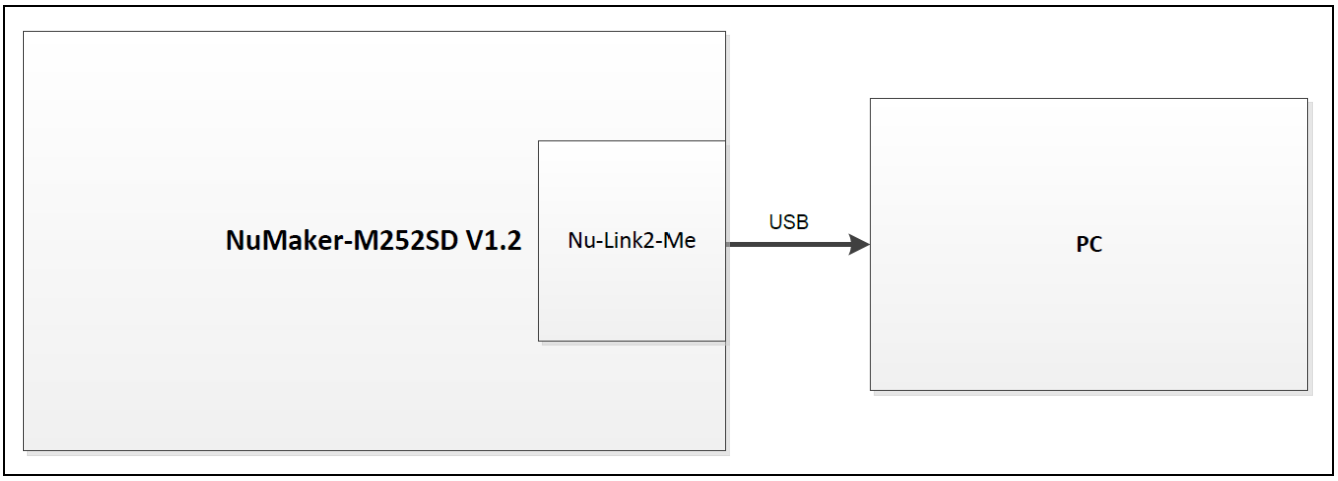


圖 3-1 線路示意圖

4. 目錄資訊

EC_M252_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

圖 4-1 目錄資訊

5. 範例程式執行

1. 根據目錄資訊章節進入 ExampleCode 路徑中的 KEIL 資料夾，雙擊 RT_Thread_Nano_Porting.uvprojx。
2. 進入編譯模式介面
   - 編譯
   - 下載代碼至記憶體
   - 進入 / 離開除錯模式
3. 進入除錯模式介面
   - 執行代碼

6. 修訂紀錄

Date	Revision	Description
2024.10.08	1.00	初始發布。

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.