FreeRTOS Porting

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	The document introduces how to port **FreeRTOS** on NUC100 and demonstration sample codes base on **FreeRTOS**.
BSP Version	NUC100 Series BSP CMSIS v3.00.003
Hardware	NuTiny_EVB_NUC140_LQFP100 V1.0

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 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 Function Description

1.1 Introduction

This document will introduce how to port **FreeRTOS** on NuMicro® NUC100 Series MCU.

And demonstrate example code to the porting the **FreeRTOS** on NuMicro® NUC100 Series MCU.

The main system configuration of the NUC100 MCU is as follows:

- System clock frequency is 50 MHz
- System clock source is from PLL.
- SysTick interrupt period is 1 ms

1.2 Porting FreeRTOS

Porting within the same processor family is a much more straightforward task – for example, from one M0 based device to another. The documentation page detailing how to modify an existing demo application would be a good point to start reading if this is your aim.

The **FreeRTOS** kernel source code is generally contained within 3 source files (4 if co-routines are used) that are common to all ports, and one or two 'port' files that tailor the RTOS kernel to a particular architecture.

Suggested steps:

- Download the latest version of the **FreeRTOS** source code from following website.

  https://www.freertos.org/a00104.html

- Unzip the files into a convenient location, taking care to maintain the directory structure.
- Familiarize yourself with the source code organization and directory structure.
- Create a directory that will contain the 'port' files for the [architecture] port. Following the convention outlined in the link, the directory should be of the form: FreeRTOS/Source/portable/[compiler name]/[processor name]. For example, if you are using the Keil MDK compiler you can get a ported done code from \\FreeRTOS\Source\portable\RVDS\ARM_CM0\port.c

On the implement of example, we set the project to links the port.c file directly.

- If the stack on the microcontroller being ported to grows downward from high memory to low memory then set portSTACK_GROWTH in portmacro.h to -1, otherwise set portSTACK_GROWTH to 1.
- Create a directory that will contain the demo project demo files for the port. On the implement of example, we set the example code path as \\bsp\SampleCode\FreeRTOS
- Copy existing FreeRTOSConfig.h and main.c files into the directory just created.
- Take a look at the FreeRTOSConfig.h file. It contains some macro's that will need setting for your chosen hardware. And the file will link the handlers for **FreeRTOS** necessities.

1.2.1 Demo Code Project Architecture

Demo code project architecture shown in Figure 1-1 Demo code project architecture:

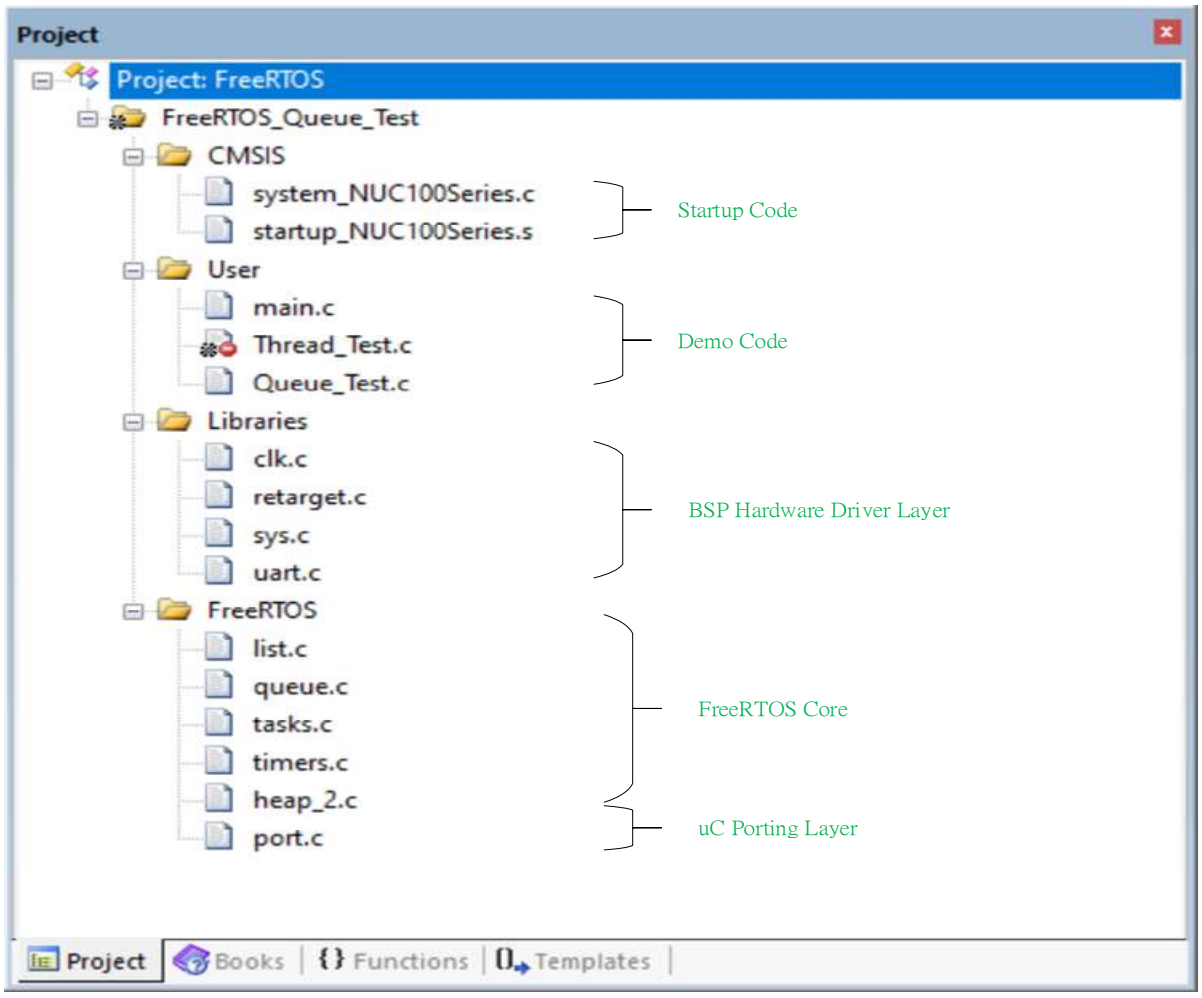

Figure 1-1 Demo code project architecture

1.2.2 Demo Code Project Include Path

Demo code project include path shown in Figure 1-2 Demo project include path:

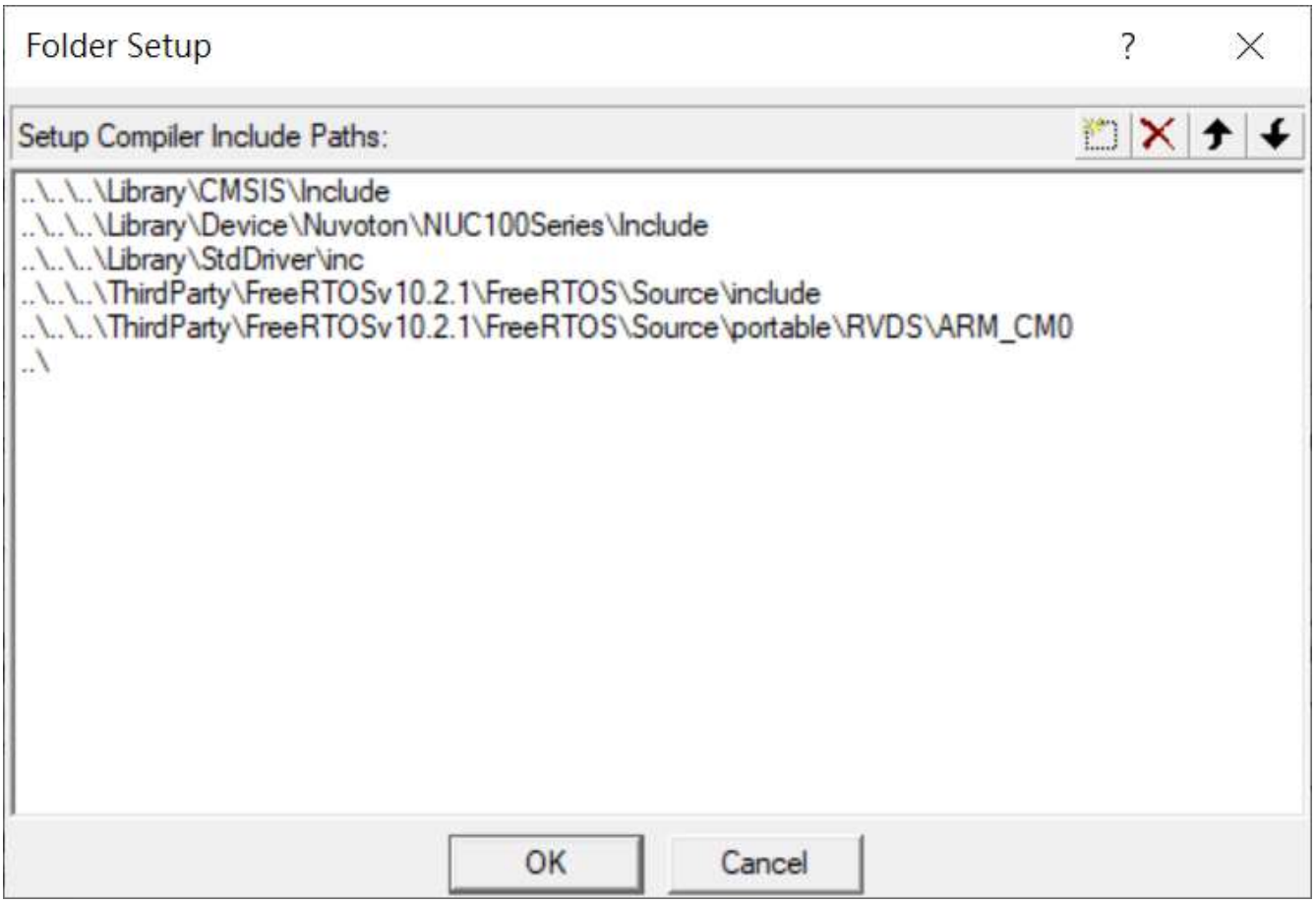

Figure 1-2 Demo project include path

1.3 Thread Test Demo Result

The application execution result shows on the Hyper Terminal - SecureCRT as shown in Figure 1-3 Thread test demo result:

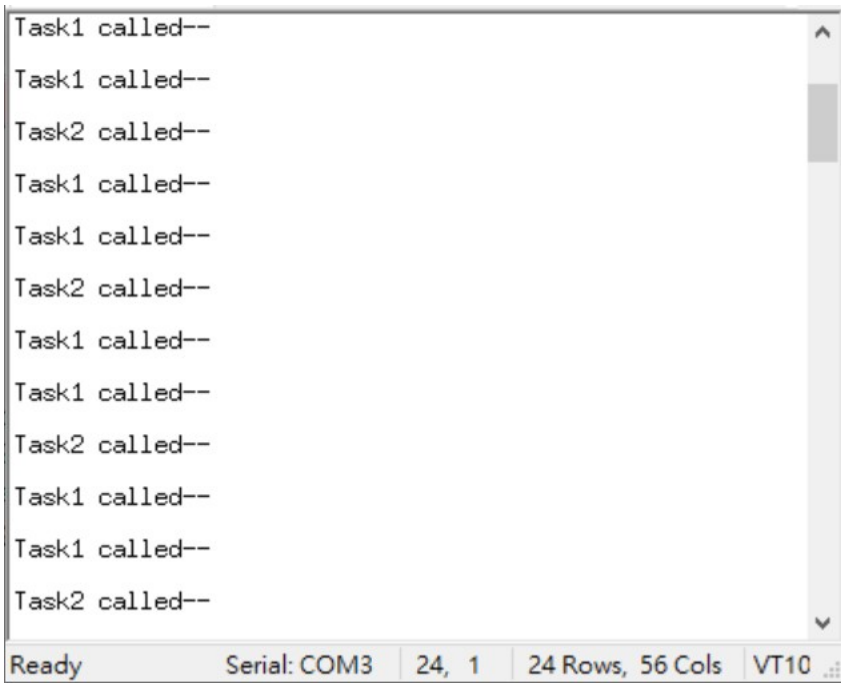

Figure 1-3 Thread test demo result

1.4 Queue Test Demo Result

The application execution result shows on the Hyper Terminal - SecureCRT as shown in Figure 1-4 Queue test demo result:

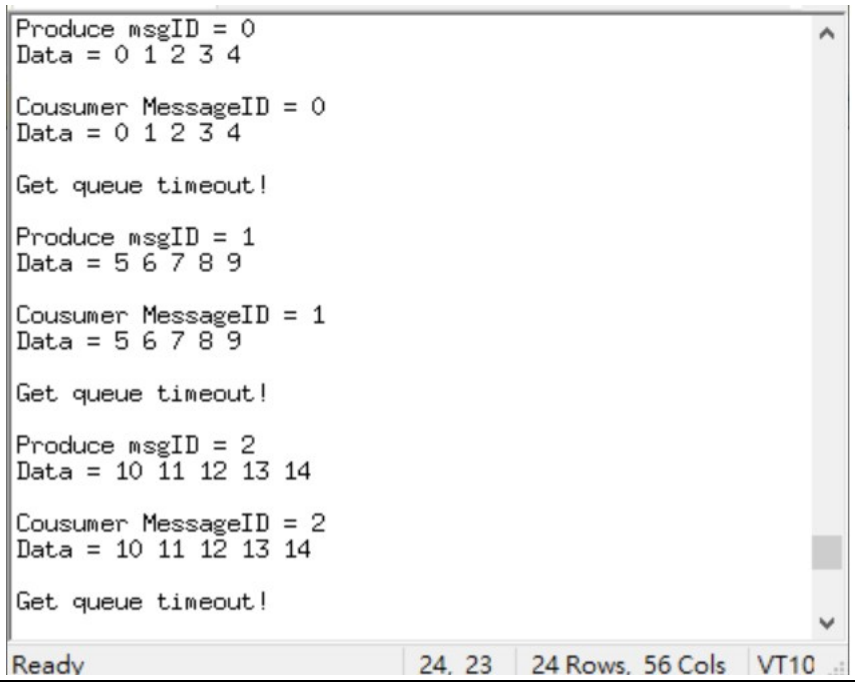

Figure 1-4 Queue test demo result

2 Demo Code Description

2.1 Introduction

Demo code project file located under folder \\NUC100\bsp\SampleCode\FreeRTOS\KEIL

There are two target projects, FreeRTOS_Thread_Test and FreeRTOS_Queue_Test.

2.2 FreeRTOS_Thread_Test

Following code initializes system and UART.

```
void prvSetupHardware(void)
{
    /* Initialize clock and pins function */
    SYS_Init();

    /* Initialize UART0 */
    UART0_Init();
}
```

Following code creates two tasks.

```
void vStartThreadTasks( void )
{
    /* Create task1 and add it to the list of task that ready to run */
    xTaskCreate( Thread_Task1, ( const char * ) "Th_Task1", THREAD_TEST_STACK_SIZE, NULL,
THREAD_TEST_PRIORITY, ( xTaskHandle * ) NULL );
    /* Create task2 and add it to the list of task that ready to run */
    xTaskCreate( Thread_Task2, ( const char * ) "Th_Task2", THREAD_TEST_STACK_SIZE, NULL,
THREAD_TEST_PRIORITY, ( xTaskHandle * ) NULL );
}


void Thread_Task1( void *pvParameters )
{
    portTickType xLastTime1;

    /* Get current system TickCount. */
    xLastTime1 = xTaskGetTickCount();

    for( ;; )
    {
```

```
        /* Run a time every 10 milliseconds. */
        vTaskDelayUntil( &xLastTime1, 10 );
        printf("\nTask1 called--\n");
    }
}

void Thread_Task2( void *pvParameters )
{
    portTickType xLastTime2;

    /* Get current system TickCount. */
    xLastTime2 = xTaskGetTickCount();

    for( ;; )
    {
        /* Run a time every 20 milliseconds. */
        vTaskDelayUntil( &xLastTime2, 20 );
        printf("\nTask2 called--\n");
    }
}
```

2.3 FreeRTOS_Queue_Test

Following code initializes system and UART.

```
void prvSetupHardware(void)
{
    /* Initialize clock and pins function */
    SYS_Init();

    /* Initialize UART0 */
    UART0_Init();
}
```

Following code creates two tasks. One is producer task. The other is consumer task.

```
void vStartQueueTasks( void )
{
    int i = 0;

    /* Create the queue used by consumer and producer tasks. */
```

```
    xQueue = xQueueCreate( MSG_COUNT, sizeof( msg_t ) );

    for( index = 0; index < MSG_COUNT; index++ )
    {
        msg[index].ucMessageID = index;
        for( i = 0; i < 5; i++ )
        {
            msg[index].ucData[i] = index*5 + i;
            printf( "input data[%d] = %d\n", index*5+i, msg[index].ucData[i] );
        }
    }
    /* Create the task for queue consumer. */
    xTaskCreate( Queue_Task1, ( const char * ) "Q_Task1",\
                 QUEUE_TEST_STACK_SIZE, ( void * ) NULL, QUEUE_CONSUMER_PRIORITY,\
                  ( xTaskHandle * ) NULL );
    /* Create the task for queue producer. */
    xTaskCreate( Queue_Task2, ( const char * ) "Q_Task2",\
                 QUEUE_TEST_STACK_SIZE, ( void * ) NULL, QUEUE_PRODUCER_PRIORITY,\
                  ( xTaskHandle * ) NULL );
}

/* Consumer task will check if any message produced by Producer task */
void Queue_Task1( void *pvParameters )
{

    msg_t message;

    int i = 0;

    for( ;; )
    {   /* Receive a message. Block for 10 ticks if message is not available */
        if( xQueueReceive( xQueue, &message, 10 ) == pdPASS )
        {
            printf("\nCousumer MessageID = %d\n", message.ucMessageID);
            printf("Data = ");
            for( i = 0; i < 5; i++ )
            {
                printf( "%d ", message.ucData[i]);
            }
            printf("\n");
        }
```

```
        else
        {
            printf( "\nGet queue timeout!\n" );
        }
    }
}

/* Producer task will produce 1 message every 20ms*/
void Queue_Task2( void *pvParameters )
{
    portTickType xLastTime1;

    int32_t i;
    index = 0;

    /* Get current system TickCount. */
    xLastTime1 = xTaskGetTickCount();

    for( ;; )
    {
        /* Run a time every 20 milliseconds. */
        vTaskDelayUntil( &xLastTime1, 20);

        printf( "\nProduce msgID = %d\n", msg[index].ucMessageID);
        printf( "Data = ");
        for(i=0; i<5; i= i+1)
            printf( "%d ", msg[index].ucData[i]);
        printf("\n");
        /* Send a message. Wait for 10 ticks for space available if necessary */
        if( xQueueSend( xQueue, ( void * ) &msg[index], 10 ) != pdPASS )
        {
            printf( "Fill queue error\n" );
        }
        else
        {
            index++;
            if( index > 4 )
            {
                index = 0;
            }
        }
```

```
    }
}
```

3 Software and Hardware Environment

Software Environment

- BSP version
  - NUC100 Series BSP CMSIS v3.01.002
- IDE version
  - Keil uVersion 5.24

Hardware Environment

- Circuit component
  - Nu-Tiny-EVB-NUC140-LQFP100 V1.0
- Diagram
  - Connect Nu-Tiny-NUC140-LQFP10 to PC for downloading code and debugging

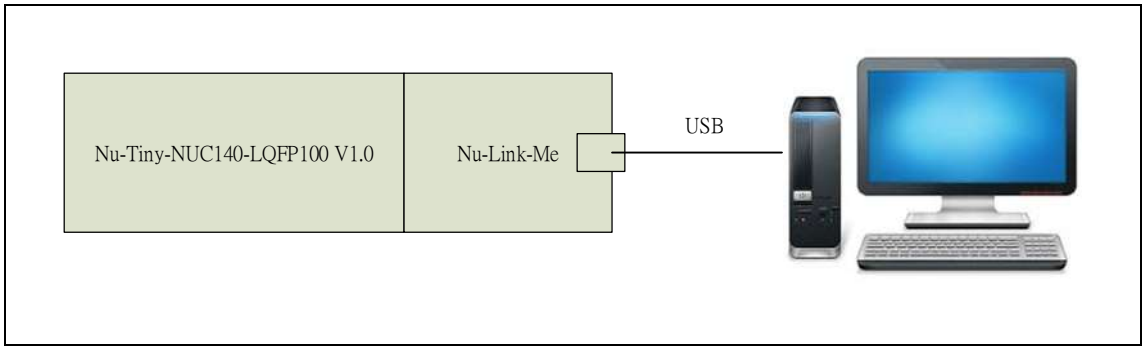

Figure 3-1 Hardware environment

4 Directory Information

EC_NUC100_FreeRTOS_Porting_V1.00	
Library	Libraries files
CMSIS	Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp.
Device	CMSIS compliant device header file
StdDriver	All peripheral driver header and source file
SampleCode	
FreeRTOS	**FreeRTOS** sample code
ThirdParty	
FreeRTOS_V10.2.1	**FreeRTOS** V10.2.1 source code

5 How to Execute Example Code

1. Opening the path NUC100\bsp\SampleCode\FreeRTOS\KEIL folder by Directory Information and double click FreeRTOS.uvproj.
2. Enter Keil compile mode
   a. Build the target project - FreeRTOS_Thread_Test or FreeRTOS_Queue_Test
   b. Download
   c. Start/Stop debug session
   d. Open Hyper-Terminal, choose correct UART port and set baud rate as 115200 bps, none parity bit, data bits 8 and 1 stop bit.
3. Enter debug mode
   a. Run

6 Revision History

Date	Revision	Description
Oct. 7, 2019	1.00	1. Initially issued.

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.