Use RTX to access SPI flash with PDMA

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	CMSIS-RTOS RTX demonstration for accessing SPI flash with PDMA
BSP Version	NUC230/240 Series BSP CMSIS v3.01.002
Hardware	NuEdu-EVB-NUC240 V2.1 NuEdu-Basic01 V2.1

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 example code demonstrates how to run a RTX system on NuMicro® NUC240 Series MCU and access SPI flash with PDMA.

1.2 Principle

The Keil RTX is a royalty-free, deterministic Real-Time Operating System designed for ARM and Cortex-M devices. It allows you to create programs that simultaneously perform multiple functions and helps to create applications which are better structured and more easily maintained. The detail features and benefits of Keil RTX can refer to http://www.keil.com/arm/rl-arm/kernel.asp.

This example code achieves porting the Keil RTX on NuEdu-EVB-NUC240 V2.1 board with NuMicro® NUC240 Series MCU, and creating a RTX task to read external SPI flash with PDMA on NuEdu-Basic01 V2.1 board.
The relative drivers of NuEdu-Basic01 V2.1 board are located in BSP\Library\NuEdu folder.

Figure 1-1 shows the operation flow of example code.

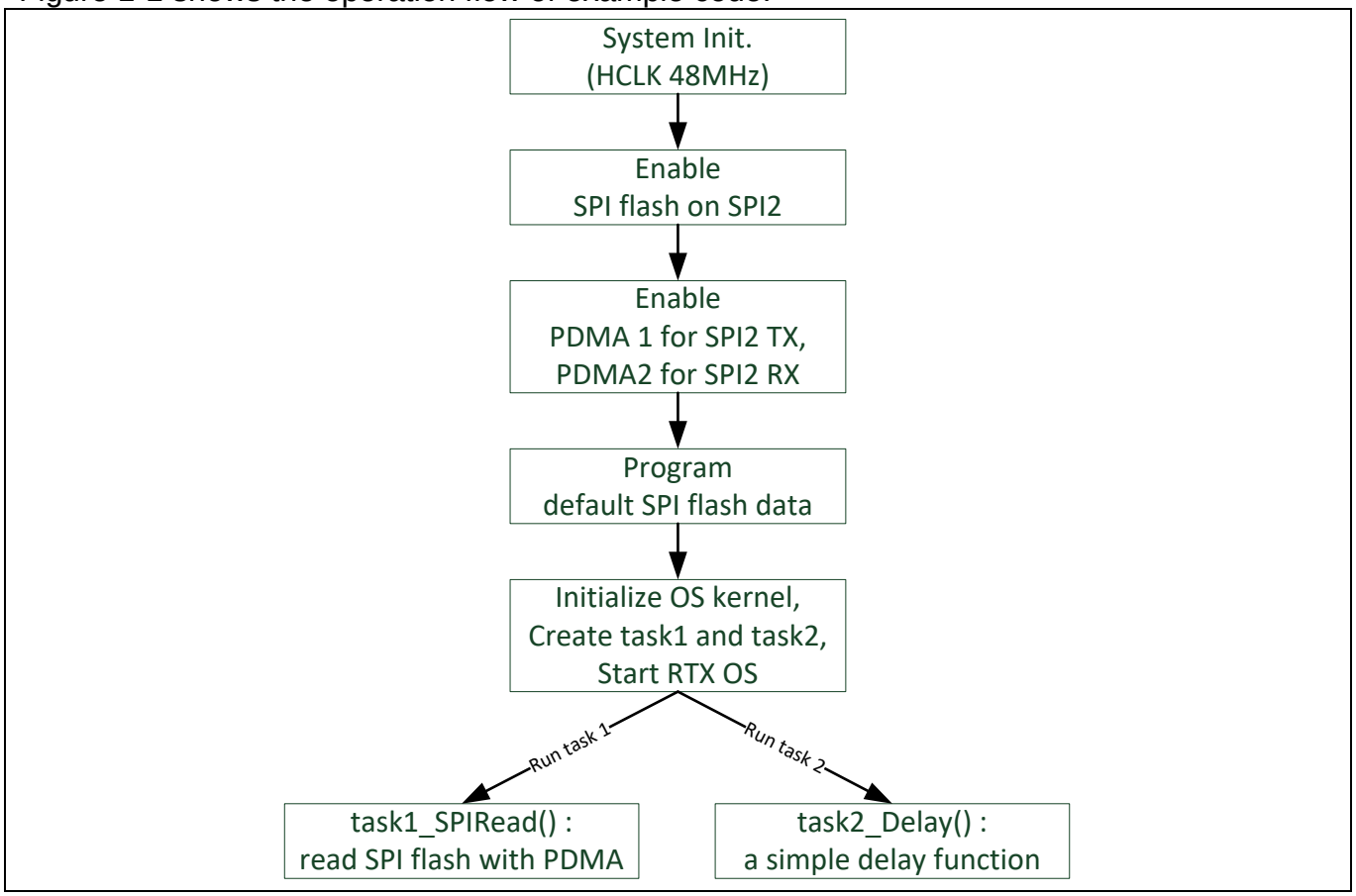

Figure 1-1 Operation Flow

1.3 Demo Result

The execution result can be output on UART, baud rate 115200, as shown in Figure 1-2 on PC.

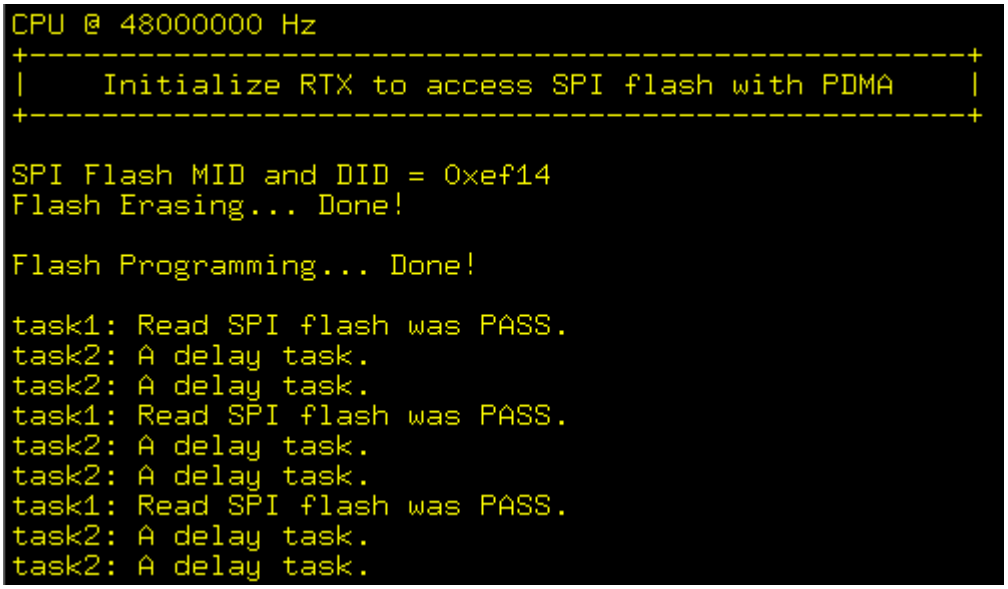

Figure 1-2 Execution Result on UART

2 Code Description

Variables and functions related to RTX usage in main.c.

```
osThreadId  t_task1; /* thread ID of task 1 */
osThreadId  t_task2; /* thread ID of task 2 */

void task1_SPIRead(void const *argument); /* thread function of task 1 */
void task2_Delay(void const *argument);   /* thread function of task 2 */

/* Thread Definition with function, priority, and stack requirements */
osThreadDef(task1_SPIRead, osPriorityNormal, 1, 0);
osThreadDef(task2_Delay, osPriorityNormal, 1, 0);

/*----------------------------------------------------------------------------
-----------------*/
/* task 1 function: read SPI flash with PDMA
*/
/*----------------------------------------------------------------------------
-----------------*/
void task1_SPIRead(void const *argument)
{
    ……
}

/*----------------------------------------------------------------------------
-----------------*/
/* task 2 function: a simple delay function
*/
/*----------------------------------------------------------------------------
-----------------*/
void task2_Delay(void const *argument)
{
    ……
}
```

Initialize PDMA for SPI TX/RX operations and program SPI flash default data in main.c. Most called APIs in InitSPIFlashDataAndPDMA() can refer to NuEdu driver, NuEdu-Basic01_SPI_Flash_w_PDMA.c.

```
void InitSPIFlashDataAndPDMA(PDMA_T *pdma)
{
    unsigned int u32ByteCount;
```

```
    unsigned int u32MidDid;

    /* Initial PDMA Channels */
    Init_PDMA_CH1_for_SPI2_TX((uint32_t)gau32SrcArray);
    Init_PDMA_CH2_for_SPI2_RX((uint32_t)gau32DestArray);

    /* Enable PDMA IRQ */
    NVIC_EnableIRQ(PDMA_IRQn);

    /* Read MID & DID */
    u32MidDid = SpiFlash_w_PDMA_ReadMidDid();
    printf("\nSPI Flash MID and DID = 0x%x", u32MidDid);

    /* Erase SPI Flash */
    SpiFlash_w_PDMA_ChipErase();
    printf("\nFlash Erasing... ");

    /* Wait ready */
    SpiFlash_w_PDMA_WaitReady();
    printf("Done!");

    /* Fill the Source Data and clear Destination Data Buffer */
    for (u32ByteCount = 0; u32ByteCount < 256; u32ByteCount++)
        gau32SrcArray[u32ByteCount] = ((u32ByteCount+0x10)&0xFF);

    /*=== Program SPI Flash ===*/
    printf("\n\nFlash Programming... ");

    /* Trigger PDMA specified Channel */
    pdma->CSR |= (PDMA_CSR_TRIG_EN_Msk | PDMA_CSR_PDMACEN_Msk);

    /* Page Program */
    SpiFlash_w_PDMA_PageProgram(0x0, 256);
    SpiFlash_w_PDMA_WaitReady();
    printf("Done!\n\n");
}
```

Initialize RTOS, create tasks and start the RTOS to run all tasks in main.c.

```
    /* Initialize RTOS */
```

```
    osKernelInitialize();


    /* Create task1 to read SPI flash with PDMA channel 2, and task 2 as a simple delay
loop */
    t_task1 = osThreadCreate(osThread(task1_SPIRead), PDMA_CH2);
    t_task2 = osThreadCreate(osThread(task2_Delay), NULL);


    /* Start the RTOS */
    osKernelStart();
```

3 Software and Hardware Environment

Software Environment

- BSP version
  - NUC230/240 Series BSP CMSIS v3.01.002
- IDE version
  - Keil uVersion 5.28

Hardware Environment

- Circuit component
  - NuEdu-EVB-NUC240 V2.1
  - NuEdu-Basic01 V2.1
- Diagram
  - Connect NUC240 UART TX (PB.1) pin on NuEdu-Basic01 V2.1 board to PC UART RX for display the execution result of example code on PC.

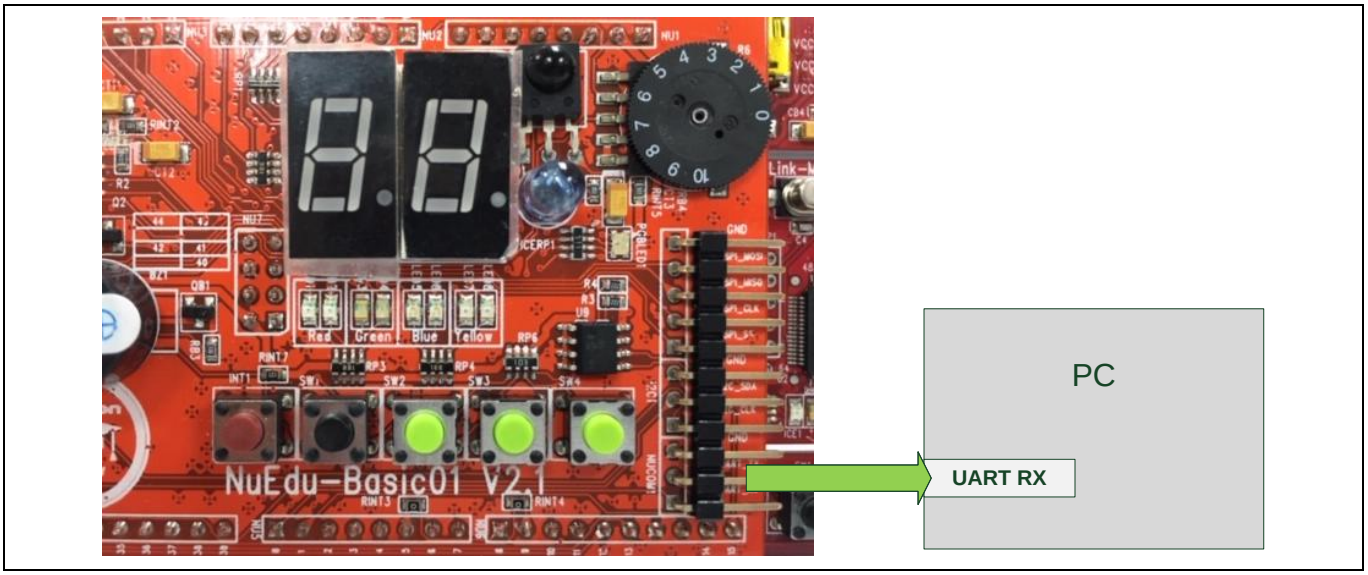

Figure 3-1 NUC240 UART TX and PC UART RX Connection

  - Connect NUC240 SPI2 on NuEdu-EVB-NUC240 V2.1 board to external SPI flash on NuEdu-Basic01 V2.1 board.

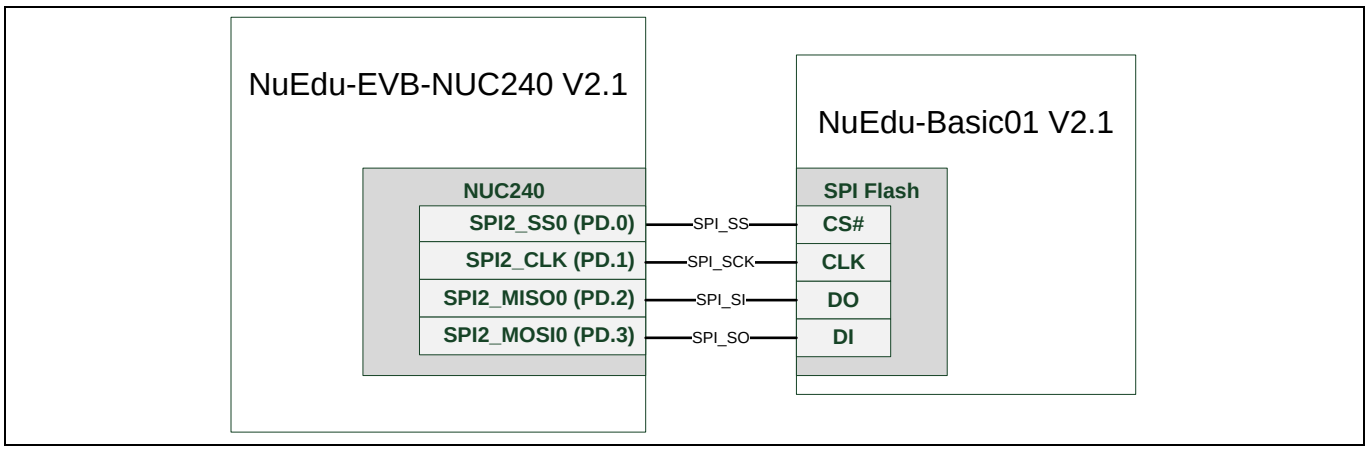

Figure 3-2 NUC240 SPI2 and external SPI flash Connection

4 Directory Information

- EC_NUC240_RTX_SPI_Flash_wPDMA_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
    - NuEdu — Library for NuEdu-SDK-xxxx board
    - StdDriver — All peripheral driver header and source files
  - SampleCode
    - ExampleCode — Source file of example code

5 How to Execute Example Code

1. Browsing into sample code folder by Directory Information (section 4) and double click RTX_SPI_Flash_wPDMA.uvprojx.
2. Enter Keil compile mode
   a. Build
   b. Download
   c. Start/Stop debug session
3. Enter debug mode
   a. Run

6 Revision History

Date	Revision	Description
Aug 21, 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.