Spare Register Access

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	Write/Read Spare Register
BSP Version	NUC230/240 Series BSP CMSIS v3.01.002
Hardware	NuTiny–EVB–NUC240 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 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 access spare register.

1.2 Principle

The RTC module is equipped with 80 bytes spare registers to store user's important information, which are also located in VBAT power domain and retained even if system core power off.
User needs to initialize the RTC module and enable spare register function before accessing the spare register.
Figure 1-1 shows the operation flow of example code to access spare register.

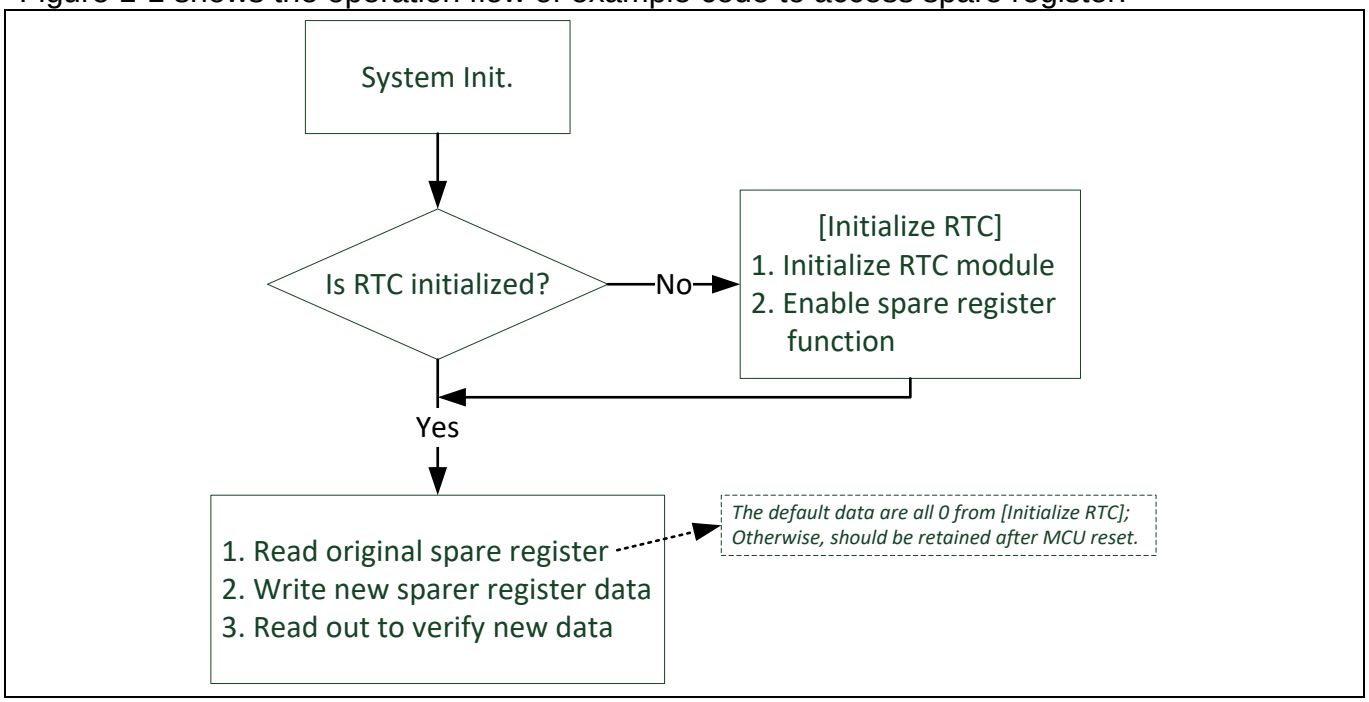

Figure 1-1 Operation Flow

1.3 Demo Result

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

```
CPU @ 72000000 Hz

Initialize RTC module done.

Read original spare register are:
 SPARE_REGISTER[ 0] = 0.
 SPARE_REGISTER[ 1] = 0.
 SPARE_REGISTER[ 2] = 0.
 SPARE_REGISTER[ 3] = 0.
 SPARE_REGISTER[ 4] = 0.
 SPARE_REGISTER[ 5] = 0.
 SPARE_REGISTER[ 6] = 0.
 SPARE_REGISTER[ 7] = 0.
 SPARE_REGISTER[ 8] = 0.
 SPARE_REGISTER[ 9] = 0.
 SPARE_REGISTER[10] = 0.
 SPARE_REGISTER[11] = 0.
 SPARE_REGISTER[12] = 0.
 SPARE_REGISTER[13] = 0.
 SPARE_REGISTER[14] = 0.
 SPARE_REGISTER[15] = 0.
 SPARE_REGISTER[16] = 0.
 SPARE_REGISTER[17] = 0.
 SPARE_REGISTER[18] = 0.
 SPARE_REGISTER[19] = 0.

Write new spare registe are:
 SPARE_REGISTER[ 0] = 100.
 SPARE_REGISTER[ 1] = 101.
 SPARE_REGISTER[ 2] = 102.
 SPARE_REGISTER[ 3] = 103.
 SPARE_REGISTER[ 4] = 104.
 SPARE_REGISTER[ 5] = 105.
 SPARE_REGISTER[ 6] = 106.
 SPARE_REGISTER[ 7] = 107.
 SPARE_REGISTER[ 8] = 108.
 SPARE_REGISTER[ 9] = 109.
 SPARE_REGISTER[10] = 110.
 SPARE_REGISTER[11] = 111.
 SPARE_REGISTER[12] = 112.
 SPARE_REGISTER[13] = 113.
 SPARE_REGISTER[14] = 114.
 SPARE_REGISTER[15] = 115.
 SPARE_REGISTER[16] = 116.
 SPARE_REGISTER[17] = 117.
 SPARE_REGISTER[18] = 118.
 SPARE_REGISTER[19] = 119.

 Compare spare registers data ... Pass!!
```

Figure 1-2 Execution Result at First Initialize RTC

Figure 1-3 shows the second execute result and unnecessary to initialize RTC module again for access spare register after MCU reset.

```
CPU @ 72000000 Hz

RTC module has been initialized.

Read original spare register are:
 SPARE_REGISTER[ 0] = 100.
 SPARE_REGISTER[ 1] = 101.
 SPARE_REGISTER[ 2] = 102.
 SPARE_REGISTER[ 3] = 103.
 SPARE_REGISTER[ 4] = 104.
 SPARE_REGISTER[ 5] = 105.
 SPARE_REGISTER[ 6] = 106.
 SPARE_REGISTER[ 7] = 107.
 SPARE_REGISTER[ 8] = 108.
 SPARE_REGISTER[ 9] = 109.
 SPARE_REGISTER[10] = 110.
 SPARE_REGISTER[11] = 111.
 SPARE_REGISTER[12] = 112.
 SPARE_REGISTER[13] = 113.
 SPARE_REGISTER[14] = 114.
 SPARE_REGISTER[15] = 115.
 SPARE_REGISTER[16] = 116.
 SPARE_REGISTER[17] = 117.
 SPARE_REGISTER[18] = 118.
 SPARE_REGISTER[19] = 119.

Write new spare registe are:
 SPARE_REGISTER[ 0] = 200.
 SPARE_REGISTER[ 1] = 201.
 SPARE_REGISTER[ 2] = 202.
 SPARE_REGISTER[ 3] = 203.
 SPARE_REGISTER[ 4] = 204.
 SPARE_REGISTER[ 5] = 205.
 SPARE_REGISTER[ 6] = 206.
 SPARE_REGISTER[ 7] = 207.
 SPARE_REGISTER[ 8] = 208.
 SPARE_REGISTER[ 9] = 209.
 SPARE_REGISTER[10] = 210.
 SPARE_REGISTER[11] = 211.
 SPARE_REGISTER[12] = 212.
 SPARE_REGISTER[13] = 213.
 SPARE_REGISTER[14] = 214.
 SPARE_REGISTER[15] = 215.
 SPARE_REGISTER[16] = 216.
 SPARE_REGISTER[17] = 217.
 SPARE_REGISTER[18] = 218.
 SPARE_REGISTER[19] = 219.

 Compare spare registers data ... Pass!!
```

Figure 1-3 Execution Result after MCU Reset

2 Code Description

Initialize and enable RTC module.

```
    /* Check if RTC module has initialized or not */
    if(RTC->INIR == RTC_INIR_ACTIVE_Msk)
    {
        printf("RTC module has been initialized.\n\n");
    }
    else
    {
        i = (SystemCoreClock / 10);
        /* Initialize RTC module */
        while(1)
        {
            RTC->INIR = RTC_INIT_KEY;
            if(RTC->INIR == RTC_INIR_ACTIVE_Msk)
                break;
            if(i-- == 0)
            {
                printf("\nInitialize RTC fail.\n\n");
                while(1) {}
            }
        }
        printf("Initialize RTC module done.\n\n");

        /* Enable spare register */
        RTC_EnableSpareRegister();
    }
```

Read original spare register data and assign an initial data for writing new spare register data.

```
    /* Read original spare register data */
    printf("Read original spare register are:\n");
    for(i=0; i<20; i++)
    {
        RTC_WaitAccessEnable();
        u32SPRData = RTC_READ_SPARE_REGISTER(i);
        printf(" SPARE_REGISTER[%2d] = %d.\n", i, u32SPRData);
    }
    RTC_WaitAccessEnable();
```

```
    u32InitData = (((RTC_READ_SPARE_REGISTER(0)/100) + 1) * 100);
    if(u32InitData > 900)
        u32InitData = 0;
```

Write new spare register data and read out to verify.

```
    /* Write new spare register data */
    printf("\n");
    printf("Write new spare registe are:\n");
    for(i=0; i<20; i++)
    {
        RTC_WaitAccessEnable();
        RTC_WRITE_SPARE_REGISTER(i, (i+u32InitData));
    }

    /* Verify spare register data */
    for(i=0; i<20; i++)
    {
        RTC_WaitAccessEnable();
        u32SPRData = RTC_READ_SPARE_REGISTER(i);
        if(u32SPRData != (i+u32InitData))
        {
            printf(" SPARE_REGISTER[%2d] = %d.\n", i, u32SPRData);
            printf(" Get spare register Fail!! \n");
            while(1) {}
        }
        else
        {
            printf(" SPARE_REGISTER[%2d] = %d.\n", i, u32SPRData);
        }
    }
```

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
  - NuTiny–EVB–NUC240 V1.2
- Diagram
  - Connect UART TX (PB.1) pin to PC UART RX for display the execution result of example code on PC.

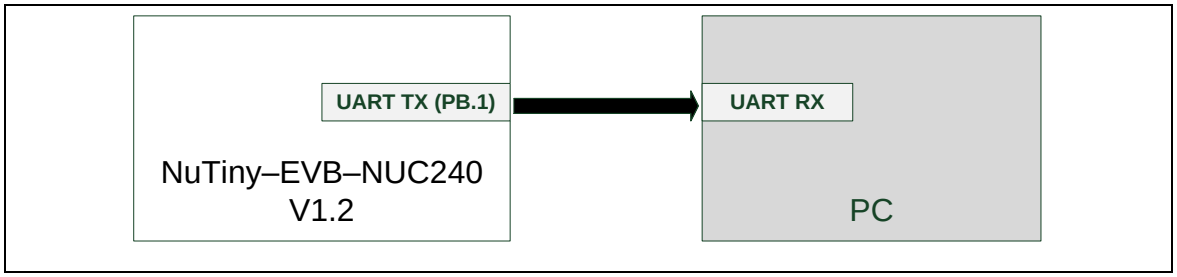

Figure 3-1 NUC240 UART TX and PC UART RX Connection

4 Directory Information

- EC_NUC240_SpareRegister_Access_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
  - 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 SpareRegister_Access.uvproj.
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
Jul 26, 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.