存取备用寄存器

NuMicro® 32 位系列微控制器范例代码介绍

文件信息

代码简述	写/读备用寄存器
BSP 版本	NUC230/240 Series BSP CMSIS v3.01.002
开发平台	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 功能介绍

1.1 简介

此范例程序将示范如何存取备用寄存器。

1.2 原理

RTC 模块配有 80 字节备用寄存器，可用于存储重要的数据或信息数据，这些数据也将同时处于 VBAT 电源区域中，即使系统核心断电，这些数据也会保留。

在访问备用寄存器之前，用户需要先初始化 RTC 模块后并启用备用寄存器功能。
下图 1-1 显示在范例程序内存取备用寄存器的流程。

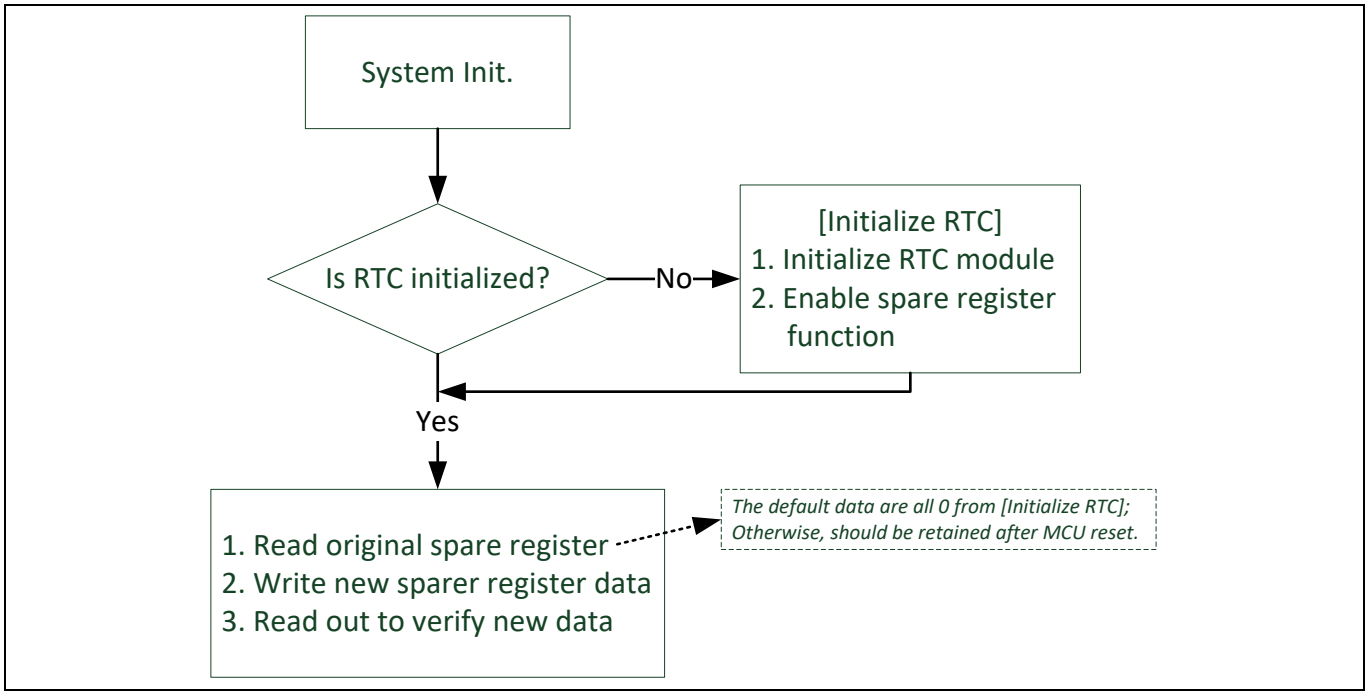

图 1-1 存取备用寄存器的流程

1.3 执行结果

系统初次上电后的执行结果可经由串口鲍率 115200 的输出到 PC 做显示，如下图 1-2 所示。

```
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!!
```

图 1-2 初始化 RTC 模块的执行结果

在图 1-3 显示出程序执行的结果为在 MCU 复位后不需要为存取备用寄存器而再一次初始化

RTC 模块。

```
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!!
```

图 1-3 MCU 复位后的执行结果

2 代码介绍

初始化并启用 RTC 模块。

```
    /* 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();
    }
```

读出备用寄存器内的数据，并分配一个新的初始值以产生后续要写入备用寄存器内的数据。

```
    /* 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;
```

将新的数据写入备用寄存器内，并读出来做验证写入是否正确。

```
    /* 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 软件与硬件环境

- **软件环境**
  - BSP 版本
    - NUC230/240 Series BSP CMSIS v3.01.002
  - IDE 版本
    - Keil uVersion 5.28

- **硬件环境**
  - 电路组件
    - NuTiny–EVB–NUC240 V1.2
  - 示意图
    - 将 NUC240 UART TX (PB.1) 引脚连接到 PC UART RX，以在 PC 上显示示例代码的执行结果。

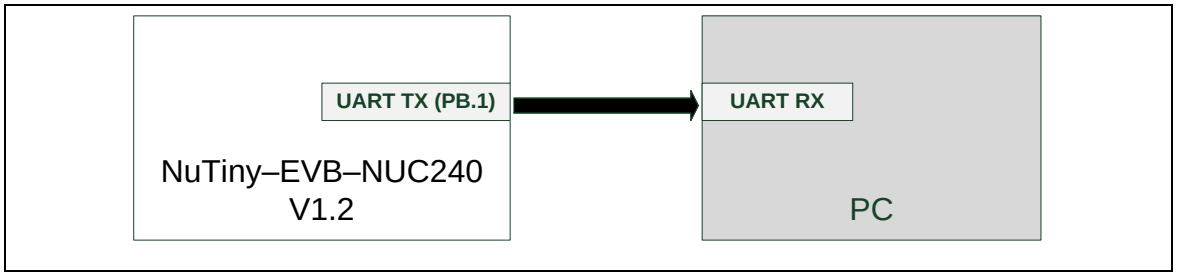

图 3-1 NUC240 UART TX 和 PC UART RX 连接

4 目录信息

- 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 如何执行范例程序

1. 根据目录信息章节进入 ExampleCode 路径中的 KEIL 文件夹，双击 SpareRegister_Access.uvproj。
2. 进入编译模式接口
   a. 编译
   b. 下载代码至内存
   c. 进入 / 离开除错模式
3. 进入除错模式接口
   a. 执行代码

6 修订纪录

Date	Revision	Description
Jul 26, 2019	1.00	1. 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.