Simulate EEPROM Using Data Flash

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	Data Flash Simulate EEPROM and Write/Read EEPROM
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 use a Data Flash region to simulate an EEPROM region.

1.2 Principle

The NuMicro® NUC240 Series is equipped with on-chip embedded Flash for application.
The limitation of write/read flash operation is the access address must be a 32-bit aligned address and the data value is a 32-bit data. Each bit data could be written from 1 to 0 and becomes 1 after erase flash operation completed. The NuMicro® NUC240 Series only supports the page erase operation, and one page size is 512 bytes.

In accessing EEPROM, write/read operation supports byte address alignment, data length is one byte, and each bit data can be written 1 or 0 directly.

In this example code, a specified Data Flash region is allocated to simulate as an EEPROM storage. A SIM_EEPROM_READ function is used to read a 8-bit data from specified EEPROM address, and a SIM_EEPROM_WRITE function is used to write a 8-bit data on specified EEPROM address directly.
Figure 1-1 shows the operation flow of example code to perform write EEPROM.

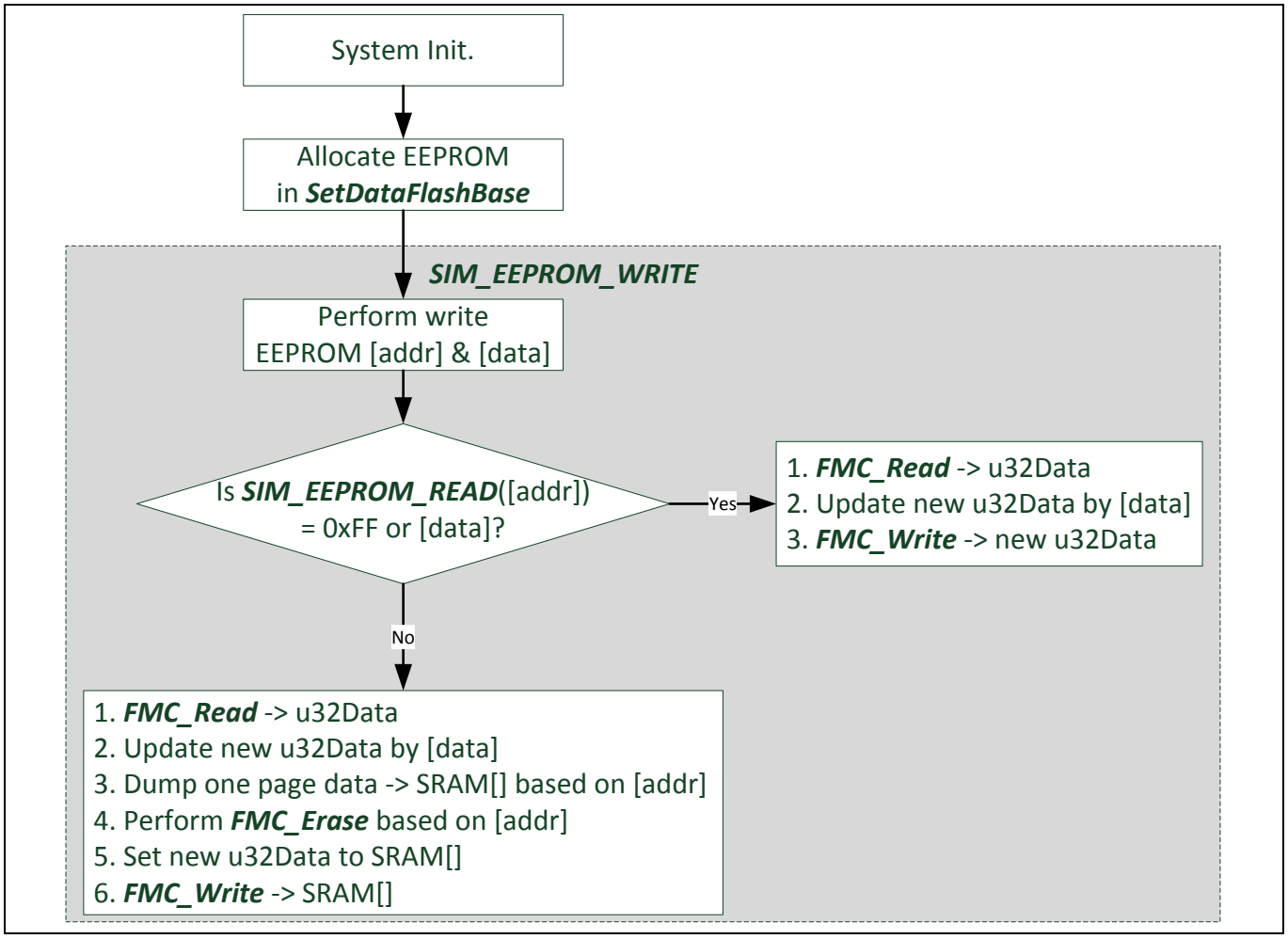

Figure 1-1 Write EEPROM Operation Flow

1.3 Demo Result

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

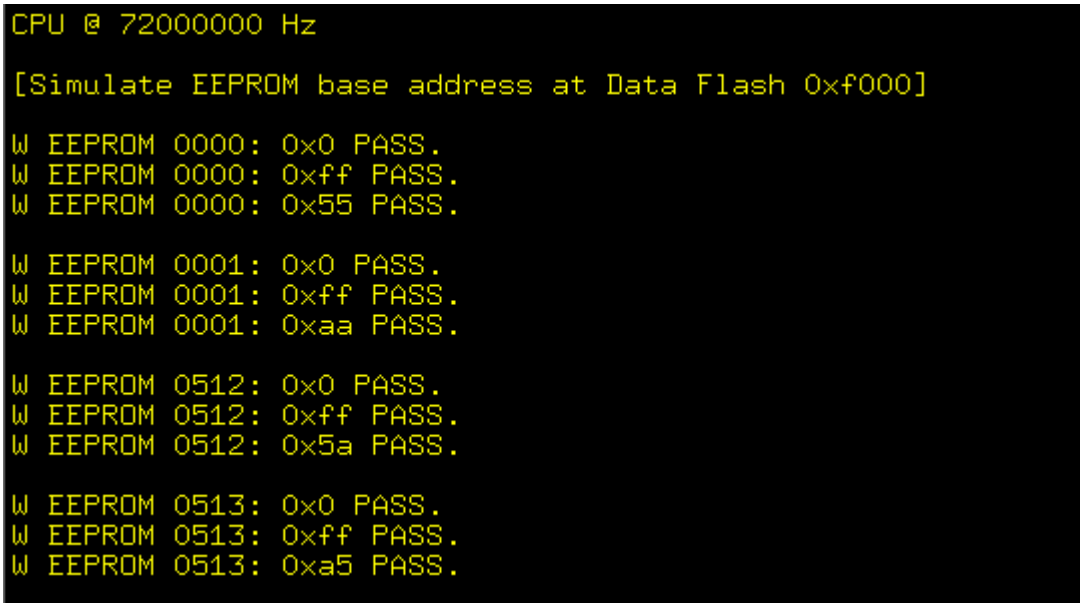

Figure 1-2 Execution Result on UART

2 Code Description

Allocate a Data Flash region to simulate EEPROM storage.

```
int32_t SetDataFlashBase(uint32_t u32DFBA)
{
    uint32_t au32Config[2];

    /* Read current User Configuration */
    FMC_ReadConfig(au32Config, 1);

    /* Just return when Data Flash has been enabled */
    if(!(au32Config[0] & 0x1))
        return 0;

    /* Enable User Configuration Update */
    FMC_EnableConfigUpdate();

    /* Erase User Configuration */
    FMC_Erase(FMC_CONFIG_BASE);

    /* Write User Configuration to Enable Data Flash */
    au32Config[0] &= ~0x1;
    au32Config[1] = u32DFBA;
    if(FMC_WriteConfig(au32Config, 2))
        return -1;

    /* Perform chip reset to make new User Config take effect */
    SYS->IPRSTC1 |= SYS_IPRSTC1_CHIP_RST_Msk;

    return 0;
}
```

Read EEPROM function.

```
uint8_t SIM_EEPROM_READ(uint32_t address)
{
    uint32_t u32Data;

    u32Data = FMC_Read(((address/4)*4) + EEPROM_BASE);
    return ((u32Data>>((address%4)*8)) & 0xFF);
```

```
}
```

Write EEPROM function.

```
void SIM_EEPROM_WRITE(uint32_t address, uint8_t data)
{
    ……
    if((SIM_EEPROM_READ(address)==0xFF) || (SIM_EEPROM_READ(address)==data))
    {
        /* Original flash data is 0xFF, or data is matched. Do not perform flash page
erase. */

        /* Read oroginal data */
        u32Data = FMC_Read(((address/4)*4) + EEPROM_BASE);
        /* Assign 32-bit updated data */
        u32Data = ~(0xFF<<((address%4)*8));
        u32Data |= (data<<((address%4)*8));
        /* Write updated data */
        FMC_Write((((address/4)*4) + EEPROM_BASE), u32Data);
    }
    else
    {
        /* Need to update one page flash data. Perform flash page erase is mandatory
before writing updated data. */

        /* Assign 32-bit updated data */
        u32Data = FMC_Read(((address/4)*4) + EEPROM_BASE);
        u32Data &= ~(0xFF<<((address%4)*8));
        u32Data |= (data<<((address%4)*8));

        /* Dump target page data to SRAM buffer */
        u32Start = (EEPROM_BASE + ((address/FMC_FLASH_PAGE_SIZE)*FMC_FLASH_PAGE_SIZE)); //
page-size alignment
        u32End   = (u32Start + FMC_FLASH_PAGE_SIZE);
        pu32DataBuf = (uint32_t *)au32DataBuf;
        for(i=u32Start; i<u32End; i+=4)
        {
            *pu32DataBuf++ = FMC_Read(i);
        }

        /* Erase target page */
        FMC_Erase(EEPROM_BASE+((address/FMC_FLASH_PAGE_SIZE)*FMC_FLASH_PAGE_SIZE));
```

```
        /* Set updated data to SRAM buffer */
        au32DataBuf[((address%FMC_FLASH_PAGE_SIZE)/4)] = u32Data;

        /* Write target flash data from SRAM buffer */
        pu32DataBuf = (uint32_t *)au32DataBuf;
        for(i=u32Start; i<u32End; i+=4)
        {
            FMC_Write(i, *pu32DataBuf++);
        }
    }
}
```

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.

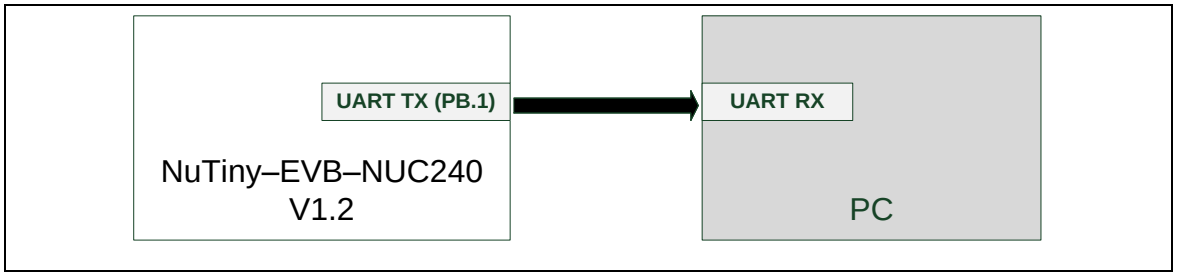

Figure 3-1 NUC240 UART TX and PC UART RX Connection

4 Directory Information

- EC_NUC240_ Simulate_EEPROM_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 Simulate_EEPROM.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 30, 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.