M480 ISP Firmware Update and Restore

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	This LDROM sample code divides APROM into run part and backup part. To restore the firmware in APROM, two parts are compared and updated after power cycling.
BSP Version	M480 Series BSP CMSIS V3.05.003
Hardware	NuMaker-PFM-M487 V3.0

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 and microprocessor 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. Overview

This LDROM sample code supports APROM update via UART, firmware verification and restoration is processed by separated bank. Since the M480 series microcontrollers (MCUs) are not equipped with bank re-mapping, two banks in APROM are acted as run part and backup part respectively as shown in Figure1-1 Flash Memory Allocation. When doing firmware update via ICE or UART, it is necessary to program an application to APROM_SIZE/2. Band0 and Bank1 are compared page by page after power cycling. If a difference is detected, sample code in LDROM starts updating the corresponding page in APROM. Besides, two banks are also compared after normal power cycling to correct the page fault.

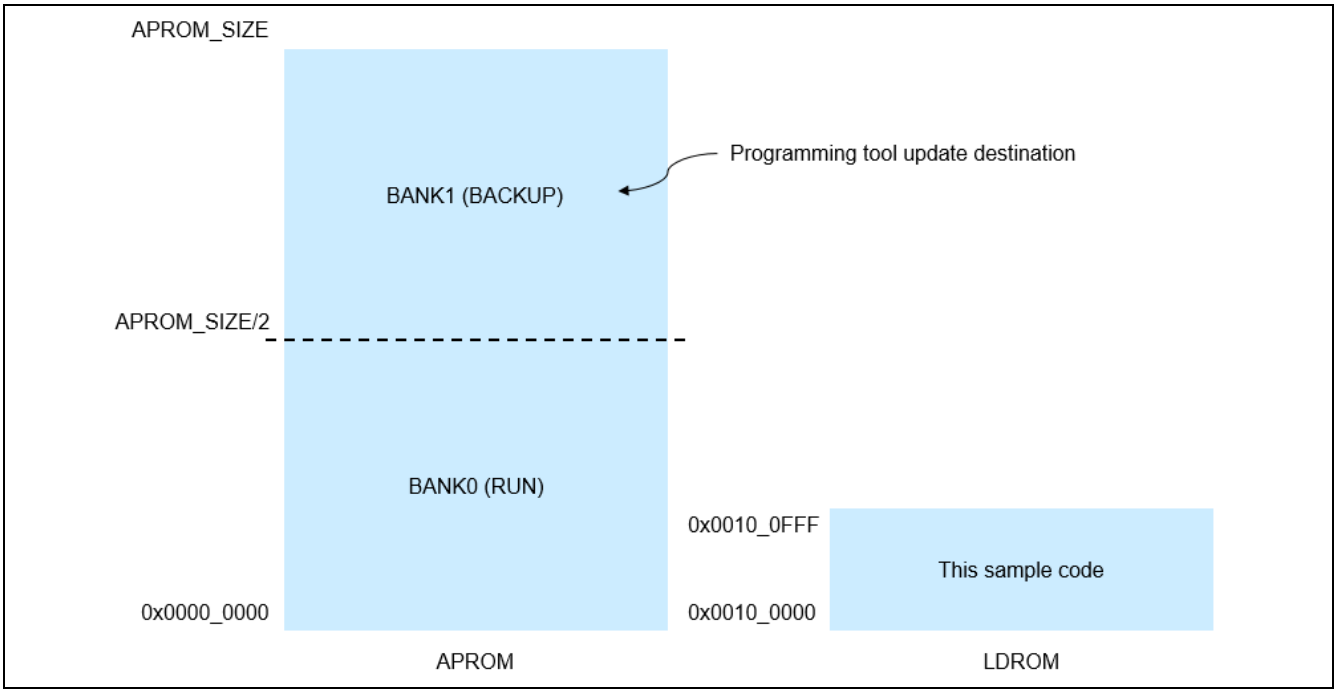

Figure1-1 Flash Memory Allocation

1.1 Software Architecture

After booting from LDROM shown as Figure1-2 Software Flow Chart, this sample code starts polling for UART command and comparing page by page between Bank0 and Bank1. If half of APROM_SIZE is compared and UART command is time-out, the software flow may be separated to: 1. If each page is identical, jump to APROM and run application. 2. If page fault is detected, erase and program the corresponding pages. Then jump to APROM and run application.

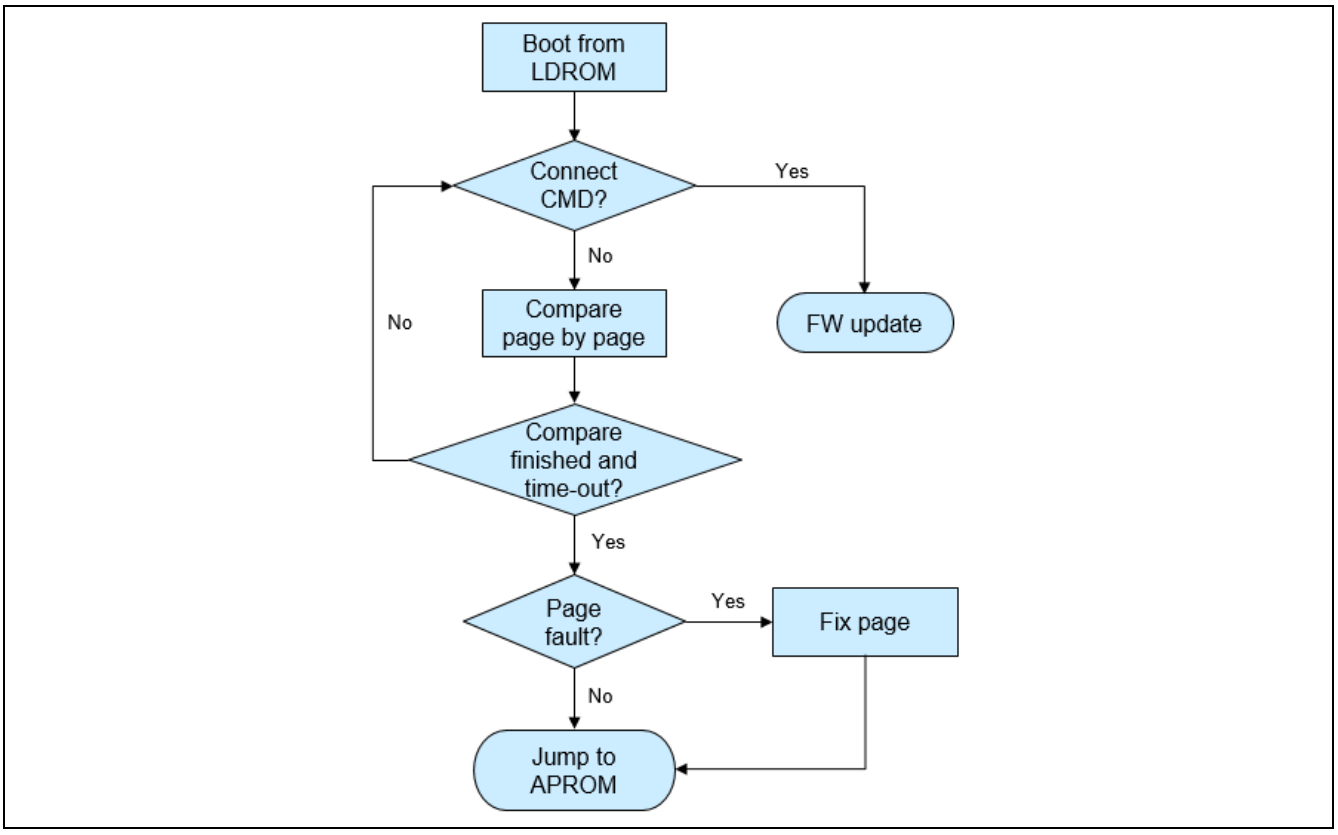

Figure1-2 Software Flow Chart

1.2 Function Description

1.2.1 Boot Mode Setting

- Boot mode:

  The M480 series supports multiple boot modes including: LDROM with IAP, LDROM without IAP, APROM with IAP, and APROM without IAP. The physical address to configure the boot mode is CONFIG0[7:6] CBS at 0x0030_0000. CBS is set to 1 in this LDROM sample code to make sure the independence between the loader and application.
  LDROM is mapped to address 0 if the boot mode, LDROM without IAP, is selected. In this case, it is necessary to access the address in APROM by FMC.

1.2.2 FMC Operation

- Erase:

  The erase unit of M480 series is page. In erase operation, write the erase command 0x22 into FMC_ISPCMD, write the physical base address of each page into FMC_ISPADDR, and write 1 into FMC_ISPTRG to finish it.
- Read:

  In read operation, write the read command 0x0 into MC_ISPCMD, write the physical address to read into FMC_ISPADDR, and write 1 into FMC_ISPTRG to finish it. 32-bit data can be read from FMC_ISPDAT each operation.
- Program:

  In program operation, write the 32-bit program command 0x21 into MC_ISPCMD, write the data to be programmed into FMC_ISPDAT, write the destination physical address into FMC_ISPADDR, and write 1 into FMC_ISPTRG to finish it.

1.2.3 Execution Instruction

1.2.3.1 Preparation

To make the chip boot from LDROM, it is necessary to connect an evaluation board with Nuvoton NuMicro ICP Programming Tool by ICE. First, program the sample code to LDROM. Then program 0x1 to CONFIG0[7:6] CBS. Besides, the code in APROM can also be programmed and updated via UART by Nuvoton NuMicro ISP Programming tool. A limitation of programming is that the size of APROM application should not exceed APROM_SIZE/2.

1.2.3.2 Execution Results

The following demonstrates a firmware update process to the chip with 512 KB APROM .

The offset at address 0x200 of Example #1 before firmware update is shown in Figure1-3 Run Part before Firmware Update.

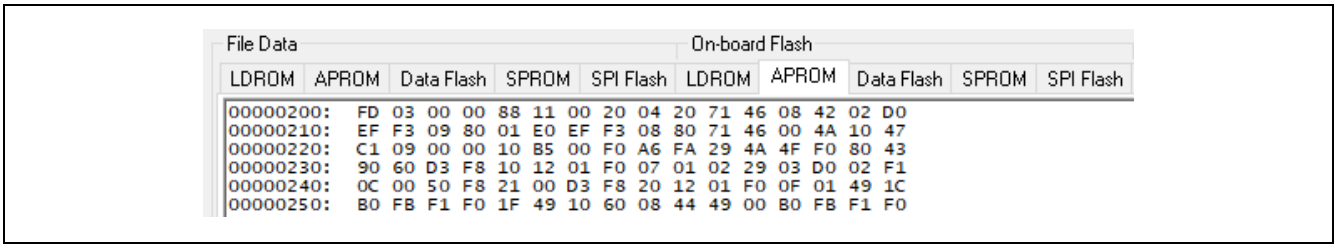

Figure1-3 Run Part before Firmware Update

The offset at address 0x40200 after program Example #2 to APROM_SIZE/2 is shown in Figure1-4 Backup Part after Firmware Update.

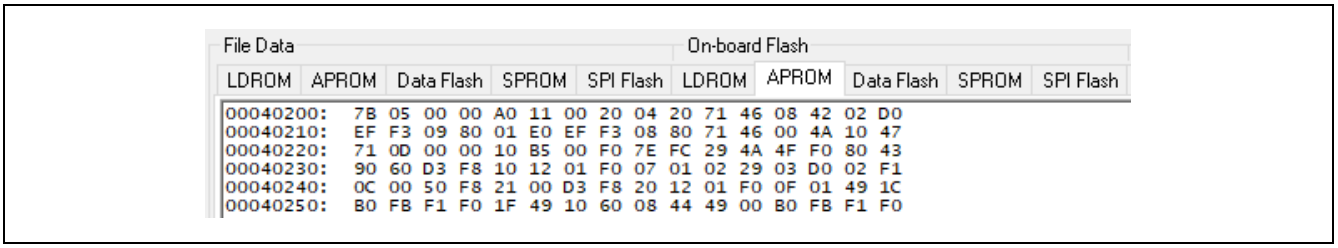

Figure1-4 Backup Part after Firmware Update

Two banks are compared and updated by sample code in LDROM after power cycling. Data at address 0x200 is identical with the address 0x40200 as shown in Figure1-5 Run Part after Firmware Update.

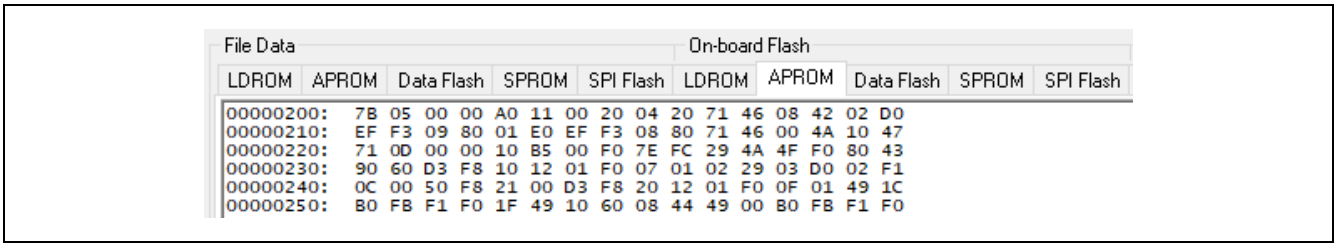

Figure1-5 Run Part after Firmware Update

2. Code Description

The code below meets the architecture in section 1.1, and the FMC operation process described in section 1.2.2 is implemented as well.

```
/* Word compare between source address and destination address */
int flashcmp(uint32_t desAddr, uint32_t srcAddr)
{
    uint32_t srcData, desData;
    FMC->ISPCMD = FMC_ISPCMD_READ;
    FMC->ISPADDR = srcAddr;
    FMC->ISPTRG = 0x1;
    __ISB();
    while (FMC->ISPTRG & 0x1) ;
    srcData = FMC->ISPDAT;

    FMC->ISPCMD = FMC_ISPCMD_READ;
    FMC->ISPADDR = desAddr;
    FMC->ISPTRG = 0x1;
    __ISB();
    while (FMC->ISPTRG & 0x1) ;
    desData = FMC->ISPDAT;

    return srcData != desData;
}

/* Word copy from source address to destination address */
void flashcpy(uint32_t desAddr, uint32_t srcAddr)
{
    uint32_t srcData;
    FMC->ISPCMD = FMC_ISPCMD_READ;
    FMC->ISPADDR = srcAddr;
    FMC->ISPTRG = 0x1;
    __ISB();
    while (FMC->ISPTRG & 0x1) ;
    srcData = FMC->ISPDAT;

    FMC->ISPCMD = FMC_ISPCMD_PROGRAM;
    FMC->ISPADDR = desAddr;
    FMC->ISPDAT = srcData;
    FMC->ISPTRG = 0x1;
    __ISB();
    while (FMC->ISPTRG & 0x1) ;
}

int main(void)
{
    uint16_t cp[4] = {0,0,0,0};
    uint32_t i = 0, j, checked = 0;
    uint32_t bsize, pnum;

    /* Unlock protected registers */
    SYS_UnlockReg();
    /* Init System, peripheral clock and multi-function I/O */
```

```
    SYS_Init();
    /* Init UART to 115200-8n1 */
    UART_Init();
    CLK->AHBCLK |= CLK_AHBCLK_ISPCKEN_Msk;
    FMC->ISPCTL |= FMC_ISPCTL_ISPEN_Msk;
    g_apromSize = GetApromSize();
    bsize = g_apromSize >> 1;
    pnum = bsize >> 12;
    GetDataFlashInfo(&g_dataFlashAddr, &g_dataFlashSize);
    SysTick->LOAD = 300000 * CyclesPerUs;
    SysTick->VAL   = (0x00);
    //using cpu clock
    SysTick->CTRL = SysTick->CTRL | SysTick_CTRL_CLKSOURCE_Msk | SysTick_CTRL_ENABLE_Msk;

    while (1)
    {
    /* Command catch */
        if ((bufhead >= 4) || (bUartDataReady == TRUE))
        {
            uint32_t lcmd;
            lcmd = inpw(uart_rcvbuf);

            if (lcmd == CMD_CONNECT)
            {
                goto _ISP;
            }
            else
            {
                bUartDataReady = FALSE;
                bufhead = 0;
            }
        }

    /* Command timeout and flash compare finished */
        if ((SysTick->CTRL & SysTick_CTRL_COUNTFLAG_Msk) && checked)
        {
         // Fix page
         for(i = 0; i < pnum; i++)
         {
             if(cp[i/16] & (1 << (i%16)))
             {
                 FMC->ISPCMD = FMC_ISPCMD_PAGE_ERASE;
                 FMC->ISPADDR = FMC_APROM_BASE + PAGE_SIZE * i;
                 FMC->ISPTRG = 0x1;
                 __ISB();
                 while (FMC->ISPTRG & 0x1) ;

                 for(j = 0; j < PAGE_SIZE; j += 4)
                 {
                     flashcpy(PAGE_SIZE * i + j, PAGE_SIZE * i + j + bsize);
                 }
             }
         }
            goto _APROM;
        }
```

```
    /* Compare flash page by page */
    if(!checked)
    {
        for(j = 0; j < PAGE_SIZE; j+=4)
        {
            if(flashcmp(i * PAGE_SIZE + j, i * PAGE_SIZE + j + bsize))
            {
                cp[i/16] |= 1 << (i%16);
                break;
            }
        }

        if(++i == pnum)
            checked = 1;
    }
    }

_ISP:

    while (1)
    {
        if (bUartDataReady == TRUE)
        {
            bUartDataReady = FALSE;
            ParseCmd(uart_rcvbuf, 64);
            PutString();
        }
    }

_APROM:
    SYS->RSTSTS = (SYS_RSTSTS_PORF_Msk | SYS_RSTSTS_PINRF_Msk);
    FMC->ISPCTL &= ~(FMC_ISPCTL_ISPEN_Msk | FMC_ISPCTL_BS_Msk);
    SCB->AIRCR = (V6M_AIRCR_VECTKEY_DATA | V6M_AIRCR_SYSRESETREQ);

    /* Trap the CPU */
    while (1);
}
```

3. Software and Hardware Requirements

3.1 Software Requirements

- BSP version
  - M480 Series BSP CMSIS V3.05.003
- IDE version
  - Keil uVersion 5.28
- Programming tools
  - Nuvoton NuMicro ICP Programming Tool
  - Nuvoton NuMicro ISP Programming Tool (Optional)

3.2 Hardware Requirements

- NuMaker-PFM-M487 V3.0

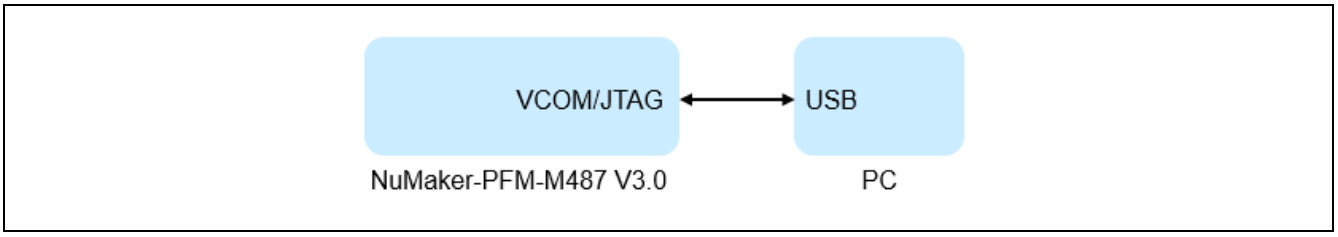

Figure3-1 Connect to PC

4. Directory Information

The directory structure is shown below.

EC_M480_ISP_Firmware_Update_and_Restore_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

Figure4-1 Directory Structure

5. Example Code Execution

1. Build the sample code.
2. Connect the evaluation board to Nuvoton NuMicro ICP Programming Tool with ICE. Next, select M480 Series.
   - Check “LDROM” and “Erase Whole Chip” in “Options”, click “LDROM” and select the binary compiled above, and click “Start” to start programming LDROM.
   - Check “Config”, uncheck “LDROM” and “Erase Whole Chip” in “Options”, click “Setting” and check “LDROM”, and click “Start” to start programming CONFIG.
   - Check “APROM” and uncheck “Config”, fill in the address offset APROM_SIZE/2 in “Offset”, click “APROM” and select any compiled sample code in BSP, and click “Start” to start programming APROM.
3. After the programming process, the chip boot from LDROM starts updating the run part, and the execution result meets the application programmed in backup part.

6. Revision History

Date	Revision	Description
2022.08.24	1.00	Initial version.

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.