M480 PDMA ExSRAM Basic Mode

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	M480 EBI interface write/read data to External SRAM with PDMA basic mode example code.
BSP Version	M480 Series BSP CMSIS V3.03.001
Hardware	NuMaker-PFM-M487 V3.0 & M487 Advance Ver4.0

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 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 chip is equipped with an external bus interface (EBI) for external device use. In order to connect between an external device and a chip, EBI is operating at address bus and data bus multiplex mode. The EBI supports three chip selects that can connect three external devices with different timing setting requirements. This example code uses M480 EBI interface to access external SRAM with PDMA on the NuMaker-PFM-M487 V3.0 board.

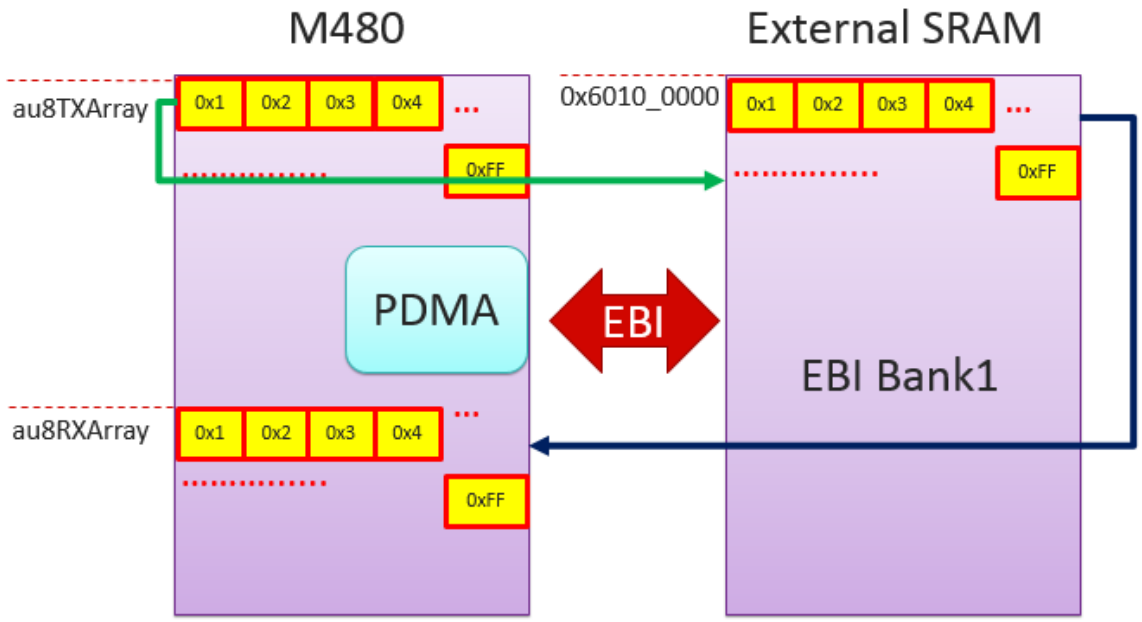
For more information on the NuMaker-PFM-M487 board and NuMaker-M487-Advance board platform, please visit Nuvoton website to download the User Manual document UM_NuMaker-PFM_M487_User_Manual and UM_NuMaker_M487_Advance_User_Manual.

https://www.nuvoton.com/hq/products/microcontrollers/arm-cortex-m4-mcus/User-Manual/?__locale=en&resourcePage=Y&category=&pageIndex=3

NuMaker emWin M487 Board

1.2 Principle

The mainly principle for user need to know about two part including hardware and software. The hardware section is to connect between M480 EBI interface and external SRAM. The software is to setup EBI initialization and PDMA.



1.3 Demo Result

After executed this example code, user will be seeing below picture. M480 EBI interface will write and read 256 bytes to external SRAM with PDMA.

```
CPU @ 192000000Hz
+-----------------------------------------------------+
|    PDMA Memory to External SRAM Sample Code         |
+-----------------------------------------------------+

----Write 256 byte value to External SRAM-------------
Test done for writting 256 byte value to External SRAM

----Read 256 byte value from External SRAM------------
Test done for reading 256 byte value from External SRAM
```

2 Code Description

This main() function is used to initialize platform for EBI interface and PDMA. After that, it will be doing two demo, one is to write data to external SRAM, the other one is to read data from external SRAM.

```
void main(void)
{
    /* Init System, IP clock and multi-function I/O */
    SYS_Init();
    ……
    /* Initialize EBI bank1 to access external SRAM (S62WV51216BLL) */
    EBI_Open(EBI_BANK1, EBI_BUSWIDTH_16BIT, EBI_TIMING_NORMAL, 0, EBI_CS_ACTIVE_LOW);

    printf("\n\nCPU @ %dHz\n", SystemCoreClock);
    printf("+------------------------------------------------------+ \n");
    printf("|    PDMA Memory to External SRAM Sample Code          | \n");
    printf("+------------------------------------------------------+ \n");

    printf("\n----Write 256 byte value to External SRAM------------ \n");
    for(i=0; i<BUFSIZE; i++) au8TXArray[i] = i;
    /* Open Channel 2 */
    PDMA_Open(PDMA,1 << 2);
    /* Transfer count is PDMA_TEST_LENGTH, transfer width is 32 bits(one word) */
    PDMA_SetTransferCnt(PDMA,2, PDMA_WIDTH_32, PDMA_TEST_LENGTH);
    /* Set source address is au8TXArray, destination address is EBI_BANK1_BASE_ADDR,
Source/Destination increment size is 32 bits(one word) */
    PDMA_SetTransferAddr(PDMA,2, (uint32_t)au8TXArray, PDMA_SAR_INC, EBI_BANK1_BASE_ADDR,
PDMA_DAR_INC);
    /* Request source is memory to memory */
    PDMA_SetTransferMode(PDMA,2, PDMA_MEM, FALSE, 0);
    /* Transfer type is burst transfer and burst size is 4 */
    PDMA_SetBurstType(PDMA,2, PDMA_REQ_BURST, PDMA_BURST_4);
    /* Enable interrupt */
    PDMA_EnableInt(PDMA,2, PDMA_INT_TRANS_DONE);
    /* Enable NVIC for PDMA */
    NVIC_EnableIRQ(PDMA_IRQn);
    g_u32IsTestOver = 0;
    /* Generate a software request to trigger transfer with PDMA channel 2  */
    PDMA_Trigger(PDMA,2);
```

```
    /* Waiting for transfer done */
    while(g_u32IsTestOver == 0);
    /* Check transfer result */
    if(g_u32IsTestOver == 1)
        printf("Test done for writting 256 byte value to External SRAM \n");
    else if(g_u32IsTestOver == 2)
        printf("Target abort...\n");

    printf("\n----Read 256 byte value from External SRAM------------ \n");
    for(i=0; i<BUFSIZE; i++) au8RXArray[i] = 0;
     /* Open Channel 3 */
    PDMA_Open(PDMA,1 << 3);
    /* Transfer count is PDMA_TEST_LENGTH, transfer width is 32 bits(one word) */
    PDMA_SetTransferCnt(PDMA,3, PDMA_WIDTH_32, PDMA_TEST_LENGTH);
    /* Set source address is EBI_BANK1_BASE_ADDR, destination address is au8RXArray,
Source/Destination increment size is 32 bits(one word) */
    PDMA_SetTransferAddr(PDMA,3, EBI_BANK1_BASE_ADDR, PDMA_SAR_INC, (uint32_t)au8RXArray,
PDMA_DAR_INC);
    /* Request source is memory to memory */
    PDMA_SetTransferMode(PDMA,3, PDMA_MEM, FALSE, 0);
    /* Transfer type is burst transfer and burst size is 4 */
    PDMA_SetBurstType(PDMA,3, PDMA_REQ_BURST, PDMA_BURST_4);
    /* Enable interrupt */
    PDMA_EnableInt(PDMA,3, PDMA_INT_TRANS_DONE);
    /* Enable NVIC for PDMA */
    NVIC_EnableIRQ(PDMA_IRQn);
    g_u32IsTestOver = 0;
    /* Generate a software request to trigger transfer with PDMA channel 3  */
    PDMA_Trigger(PDMA,3);

    /* Waiting for transfer done */
    while(g_u32IsTestOver == 0);
    /* Check transfer result */
    if(g_u32IsTestOver == 1)
        printf("Test done for reading 256 byte value from External SRAM \n");
    else if(g_u32IsTestOver == 2)
        printf("Target abort...\n");
```

This PDMA_IRQHandler () function is used to detect status of each PDMA including how to handle abort and finish.

```
void PDMA_IRQHandler(void)
{
    uint32_t status = PDMA_GET_INT_STATUS(PDMA);
    if(status & PDMA_INTSTS_ABTIF_Msk)    /* abort */
    {
        /* Check if channel 2, 3 has abort error */
        if(PDMA_GET_ABORT_STS(PDMA) & PDMA_ABTSTS_ABTIF2_Msk)
        {
            /* Clear abort flag of channel 2 */
            PDMA_CLR_ABORT_FLAG(PDMA,PDMA_ABTSTS_ABTIF2_Msk);
        }
        else if(PDMA_GET_ABORT_STS(PDMA) & PDMA_ABTSTS_ABTIF3_Msk)
        {
            /* Clear abort flag of channel 3 */
            PDMA_CLR_ABORT_FLAG(PDMA,PDMA_ABTSTS_ABTIF3_Msk);
        }
        g_u32IsTestOver = 2;
    }
    else if(status & PDMA_INTSTS_TDIF_Msk)      /* done */
    {
        /* Check transmission of channel 2, 3 has been transfer done */
        if(PDMA_GET_TD_STS(PDMA) & PDMA_TDSTS_TDIF2_Msk)
        {
            /* Clear transfer done flag of channel 2 */
            PDMA_CLR_TD_FLAG(PDMA,PDMA_TDSTS_TDIF2_Msk);
        }
        else if(PDMA_GET_TD_STS(PDMA) & PDMA_TDSTS_TDIF3_Msk)
        {
            /* Clear transfer done flag of channel 3 */
            PDMA_CLR_TD_FLAG(PDMA,PDMA_TDSTS_TDIF3_Msk);
        }
        g_u32IsTestOver = 1;
    }
}
```

3 Software and Hardware Environment

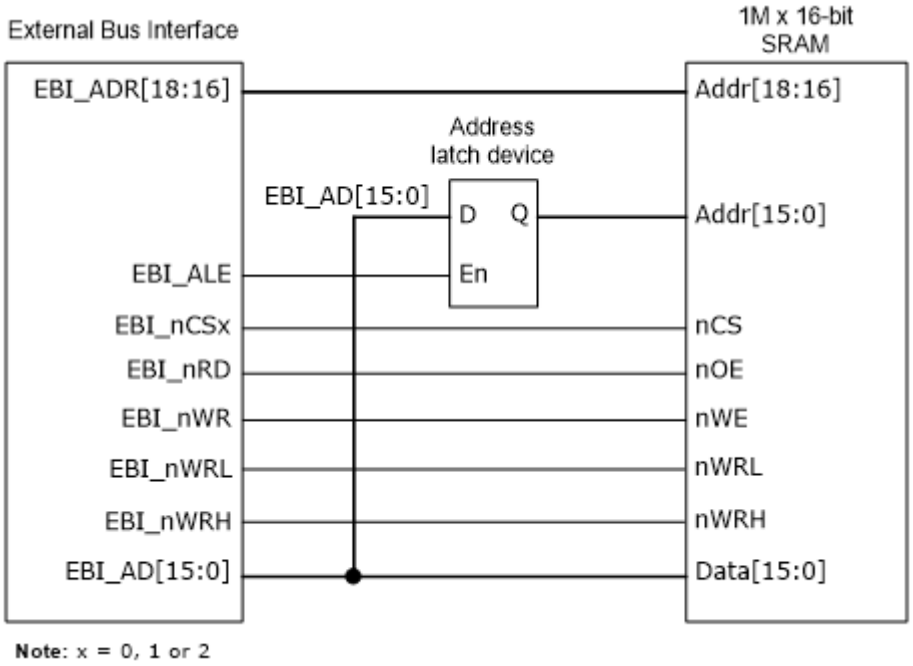
Software Environment

- BSP version
  - M480 Series BSP CMSIS V3.03.001
- IDE version
  - Keil uVersion 5.26

Hardware Environment

- Circuit components
  - NuMaker-PFM-M487 V3.0 board
  - M487 Advance Ver4.0
- Diagram

The below diagram is to show how to connect external SRAM with M487 including EBI interface for accessing data in multiplex mode.



4 Directory Information

EC_M480_PDMA_ExSRAM_BasicMode_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 PDMA_ExSRAM_BasicMode.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. 18, 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.