# External SRAM Access with M487 SPIM

Example Code Introduction for 32-bit NuMicro® Family

## Information

| | |
|---|---|
| Application | This example uses the M487 SPIM interface to access the external SRAM in DMM mode. |
| BSP Version | M480_BSP_CMSIS_V3.05.001 |
| 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 example uses the M487 to access the external SRAM (VTI7064) through the SPIM (DMM) interface.

## 1.1 Principle

The VTI7064 of Vilsion is a 64 Mbit serial SRAM that supports 1.8V and 3.3V SPI communications.
The VTI7064 read operation command (0x03) is Figure 1-1.

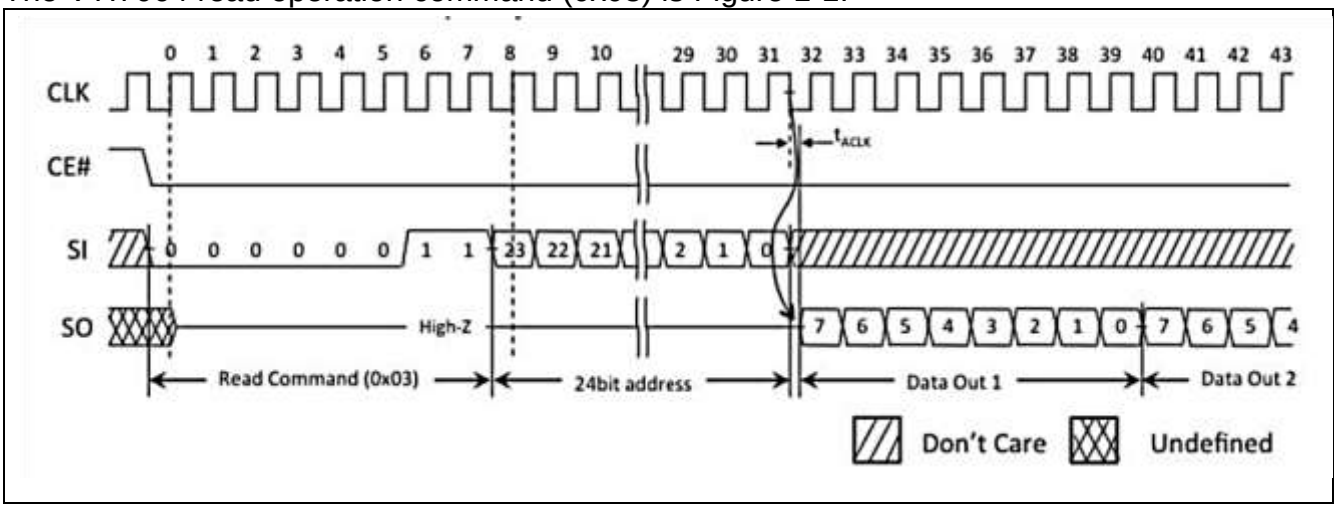


Figure 1-1 VTI7064 Read Operation Sequence Diagram

The VTI7064 write command (0x02) is Figure 1-2.

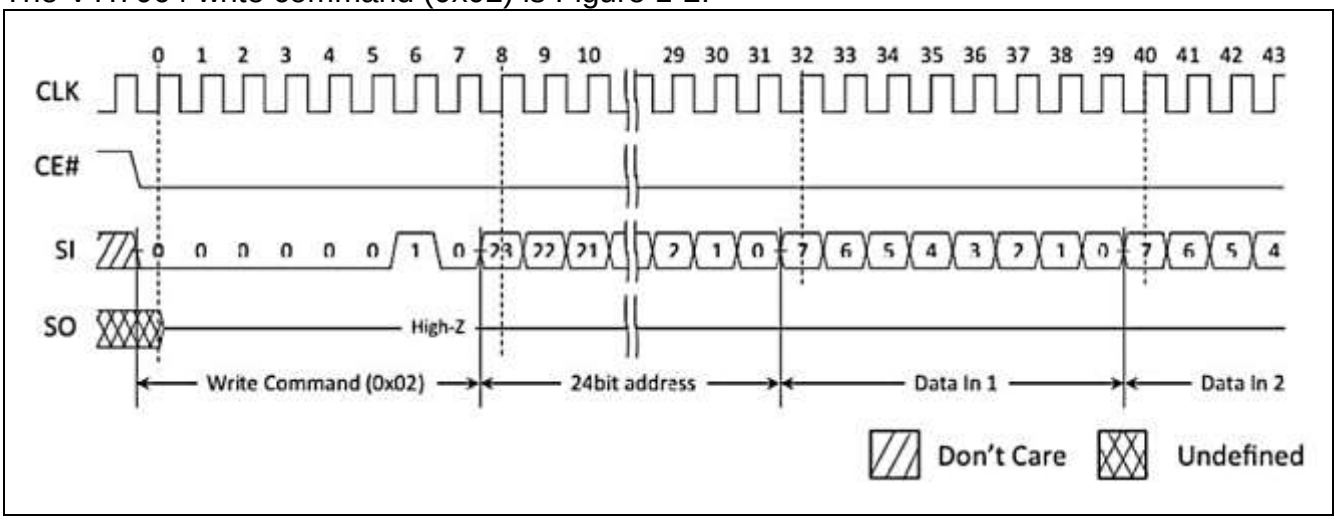


Figure 1-2 VTI7064 Write Operation Sequence Diagram

The VTI7064 reset operation is as follows: the operation consists of two commands: reset enable (0x66) and reset (0x99).

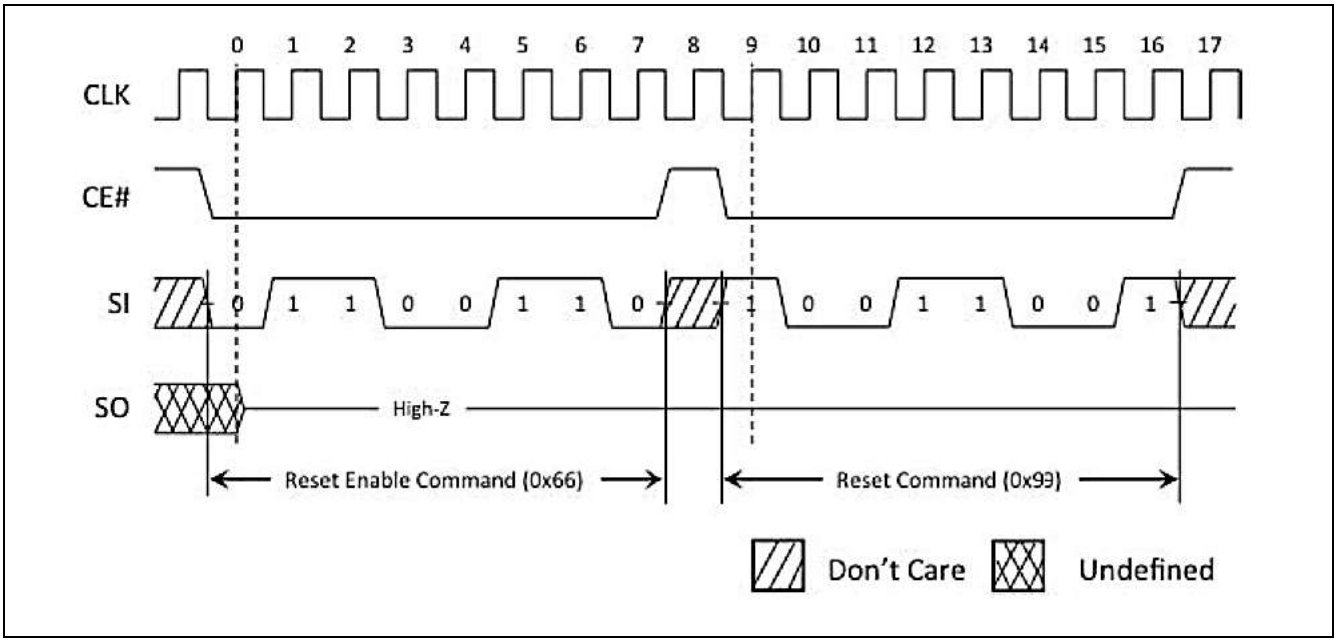


Figure 1-3 VTI7064 Reset Operation Sequence Diagram

**Note:** Figure 1-1, Figure 1-2 and Figure 1-3 are derived from VTI7064 Datasheet. For more information about VTI7064, please refer to the VTI7064 Datasheet.

This example uses the SPIM interface to access the external SRAM(VTI7064). In the direct memory mapping mode, when the microcontroller (MCU) sets the CMDCODE (SPIM_CTL0[31:24]) command code, the controller will access the SRAM through the AHB, and the MCU is not required to set the relevant SRAM commands.

With the NuMaker-PFM-M487 evaluation board, the VTI7064 can be welded to the circuit board at U6 to access SRAM (VTI7064) through SPIM interface. The relevant circuit diagram of VTI7064 is shown in Figure 1-4.

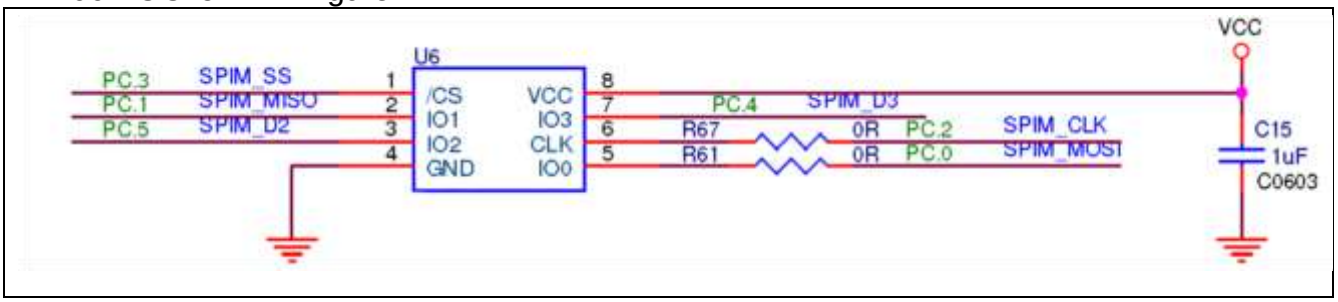


Figure 1-4 VTI7064 Circuit Diagram

## 1.2 Demo Result

The operation results are shown in Figure 1-5. It can be seen that the SRAM is initialized successfully, and the data written to the SRAM is consistent with the read data.

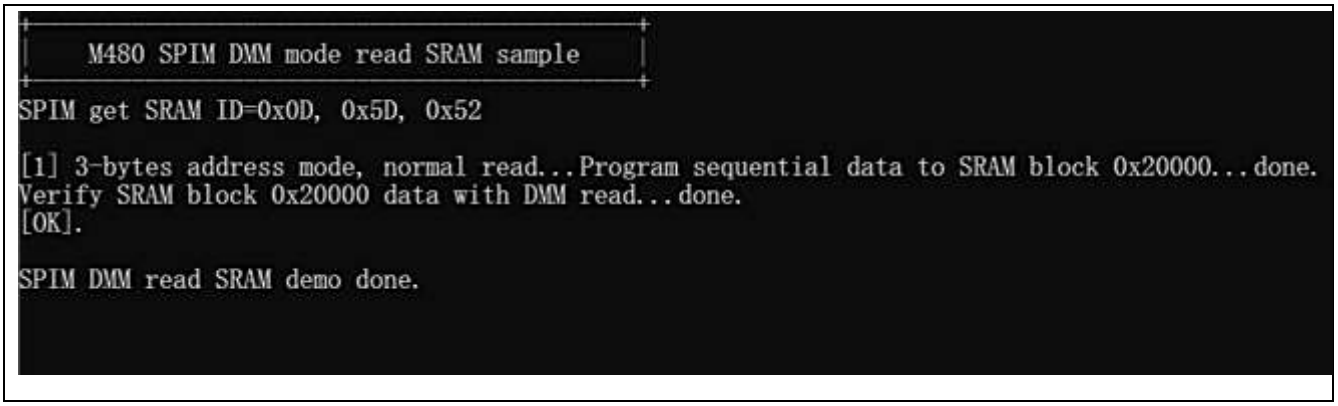


Figure 1-5 Programming Result Printing

# 2 Code Description

Initialize VTI7064 and read SRAM ID.

```
    if (SPIM_InitSRAM(1) != 0)        /* Initialized SRAM */
    {
        printf("SPIM SRAM initialize failed!\n");
        goto lexit;
    }
    SPIM_ReadSRAMId(idBuf, sizeof(idBuf), 1);
    printf("SPIM get SRAM ID=0x%02X, 0x%02X, 0x%02X\n", idBuf[0], idBuf[1], idBuf[2]);
```

Write the data stream to the specified start address 0x20000 of SRAM through the I/O mode of SPIM interface.

```
    printf("Program sequential data to SRAM block 0x%x...", TEST_BLOCK_ADDR);
    for (offset = 0; offset < SRAM_BLOCK_SIZE; offset += BUFFER_SIZE)
    {
        pData = (uint32_t *)g_buff;
        for (i = 0; i < BUFFER_SIZE; i += 4, pData++)
            * pData = (i << 16) | (TEST_BLOCK_ADDR + offset + i);
        SPIM_IO_Write_SRAM(TEST_BLOCK_ADDR + offset, 0, BUFFER_SIZE, g_buff, OPCODE_PP, 1,
1, 1);
        }
    printf("done.\n");
```

Read the data stream with the specified start address 0x20000 of SRAM through the DMM mode of SPIM interface and compare it with the written data. If there is different data, it will print out “FAILED”.

```
     printf("Verify SRAM block 0x%x data with DMM read...", TEST_BLOCK_ADDR);
     SPIM_EnterDirectMapMode(is4ByteAddr, u32RdCmd, 8);
     for (offset = 0; offset < SRAM_BLOCK_SIZE; offset += BUFFER_SIZE)
     {
    memcpy(g_buff, (uint8_t *)(SPIM_DMM_MAP_ADDR + TEST_BLOCK_ADDR + offset), BUFFER_SIZE);
    pData = (uint32_t *)g_buff;
    for (i = 0; i < BUFFER_SIZE; i += 4, pData++)
    {
          if (*pData != ((i << 16) | (TEST_BLOCK_ADDR + offset + i)))
         {
            printf("FAILED!\n");
            printf("SRAM address 0x%x, read 0x%x, expect 0x%x!\n", TEST_BLOCK_ADDR + i,
*pData, (i << 16) | (TEST_BLOCK_ADDR + offset + i));
                    SPIM_ExitDirectMapMode();
                    return -1;
                }
```

```
            }
        }
    printf("done.\n");
```

# 3 Software and Hardware Requirements

## 3.1 Software Requirements

- BSP version
  - M480_BSP_CMSIS_V3.05.001
- IDE version
  - Keil uVersion 5.35

## 3.2 Hardware Requirements

- Circuit components
  - NuMaker-PFM-M487 V3.0
  - SRAM VTI7064MSMI
- Pin Connect

| M487 | | SRAM VTI7064 |
|---|---|---|
| VCC | ↔ | VCC |
| SPIM_SS(PC.3) | → | CS |
| SPIM_MISO (PC.1) | ↔ | IO1 |
| SPIM _D2(PC.5) | ↔ | IO2 |
| SPIM _D3(PC.4) | ↔ | IO3 |
| SPIM _CLK(PC.2) | → | CLK |
| SPIM _MOSI(PC.0) | ↔ | IO0 |
| GND | ↔ | GND |

Figure 3-1 Connection Diagram between M487 and VTI7064

# 4 Directory Information

The directory structure is shown below.

| | |
|---|---|
| EC_M487_SPIM_DMM_SRAM_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 |

Figure 4-1 Directory Information

# 5 Example Code Execution

1. Browse the sample code folder as described in the Directory Information section and double-click *SPIM_DMM_SRAM.uvproj*.
2. Enter Keil compile mode.
   - Build
   - Download
   - Start/Stop debug session
3. Open the serial port tool and select the corresponding serial port.
   - Press the reset key and view the serial port debug information

# 6 Revision History

| Date | Revision | Description |
|---|---|---|
| 2021.12.13 | 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.*