M480 M4 Bit-Banding

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	Bit-banding is a term that ARM uses to describe a feature that is available on the NuMicro® M4 microcontroller. Bit-banding is performing atomic bitwise modifications to memory. This document demonstrates how you can use C code with a simple pre-processor macro to perform Bit-banding operation.
BSP Version	M480 Series BSP CMSIS V3.03.000
Hardware	NuMaker-ETM-M487 v1.1

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

Bit-banding is a feature of the Cortex-M4 processor that maps each word in an alias region of memory to a bit in a bit-band region of memory. This allows an individual bit in SRAM or peripheral register to be set/cleared by a single instruction that increases the operation speed. This sample demonstrates how to use Bit-banding operation in C with a simple pre-processor macro to perform the accesses.

1.2 Principle

Figure 1 shows examples of Bit-banding mapping between the Bit-banding alias region (32MB) and the Bit-banding region (1MB). For a more description, please see the ARM's link http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0439b/Behcjiic.html.

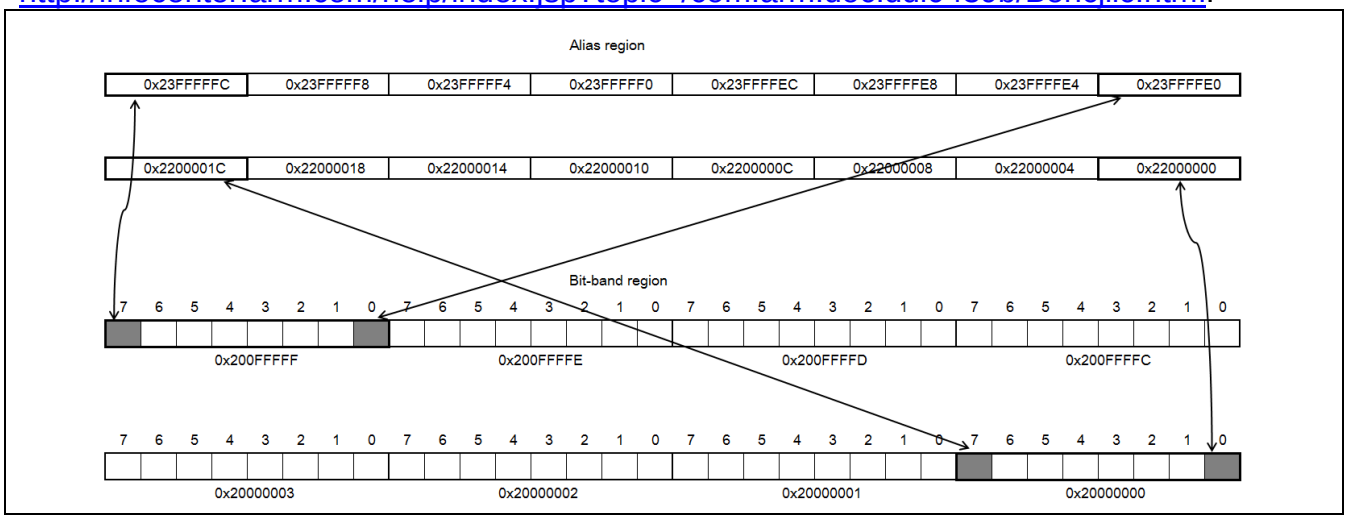


Figure 1 Bit-banding mapping

A mapping formula shows how to reference each word in the alias region to a corresponding bit, or target bit, in the Bit-banding region.

bit_word_addr = bit_band_base + ((byte_offset x 32) + (bit_number × 4))

where:

- bit_word_addr is the address of the word in the alias memory region that maps to the targeted bit.
- bit_band_base is the starting address of the alias region.
  0x20000000 : Base address of SRAM
  0x40000000 : Base address of peripheral register
- byte_offset is the number of the byte in the Bit- banding region that contains the targeted bit.
- bit_number is the bit position, 0 to 7, of the targeted bit.

1.3 Demo Result

After the program is compile completed, user can be downloaded it to the NuMaker-ETM-M487 development board for verification. In this example, setting and clearing to the third bit at the test peripheral register 0x4006000 using Bit-banding operation. The demo result as shown in Figure 2.

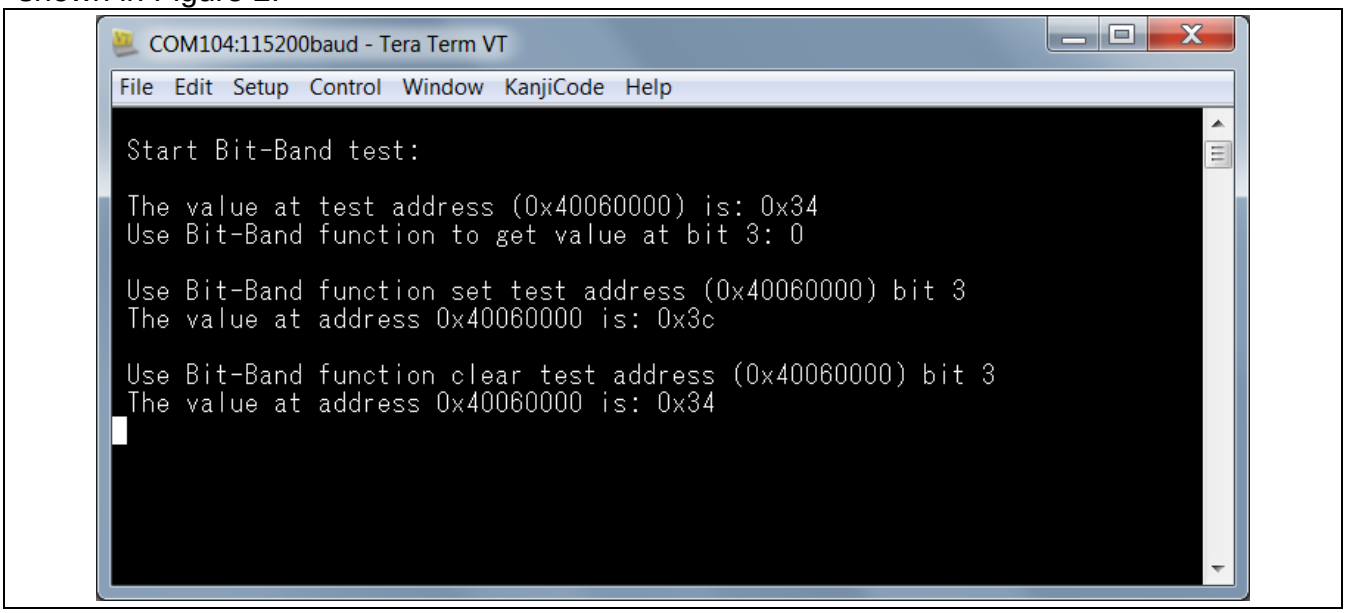


Figure 2 Bit-banding result

2 Code Description

Writing to one of the alias words will set or clear the corresponding bit in the Bit-banding region. The values of others bit in the alias region for any word are unmapped and will have no effect on the Bit-banding value.
In this example, convert peripheral register to alias region. The alias word at 0x42C0000C maps to bit [3] of the bit-band region at 0x40060000. The mapping formula is 0x42C0000C = 0x42000000 + ((0x40060000 & 0xFFFFF) * 32) + (3*4).
In this example, read the third bit at the test peripheral register 0x40060000 and set this bit. Read data from the peripheral register 0x40060000, the third bit has been set. Finally, clear the third bit of the test peripheral register 0x400060000, and read peripheral register 0x40060000 to get data is 0x34(bit[3] = 0). The following code lists how to do Bit-banding operation.

```
int main()
{
    /* Init System, IP clock and multi-function I/O */
    SYS_Init();

    /* Init UART0 for printf */
    UART0_Init();

    printf("\n Start Bit-Band test: \n");

    BitBand_Test();

    while(1);
}

void BitBand_Test(void)
{
    uint32_t u32Temp;
    uint32_t u32TestAddress;

    //SPI_CTL register address
    u32TestAddress = QSPI0_BASE;

    //Read peripheral device Address(0x40060000)
    u32Temp = MEM_ADDR(u32TestAddress);
    printf("\n The value at test address (0x40060000) is: 0x%x", u32Temp);

    //Use BitBand function to read SRAM bit value
    u32Temp = MEM_ADDR(BITBAND(u32TestAddress, 3));
```

```
    printf("\n Use Bit-Band function to get value at bit 3: %x \n", u32Temp);

    //Use BitBand function set bit
    printf("\n Use Bit-Band function set test address (0x40060000) bit 3 ");
    MEM_ADDR(BITBAND(u32TestAddress, 3)) = 1;
    //Read Test Address Value
    u32Temp = MEM_ADDR(u32TestAddress);
    printf("\n The value at address 0x40060000 is: 0x%x \n", u32Temp);

    //Use BitBand function clear bit
    printf("\n Use Bit-Band function clear test address (0x40060000) bit 3 ");
    MEM_ADDR(BITBAND(u32TestAddress, 3)) = 0;
    //Read Test Address Value
    u32Temp = MEM_ADDR(u32TestAddress);
    printf("\n The value at address 0x40060000 is: 0x%x \n", u32Temp);
}
```

3 Software and Hardware Environment

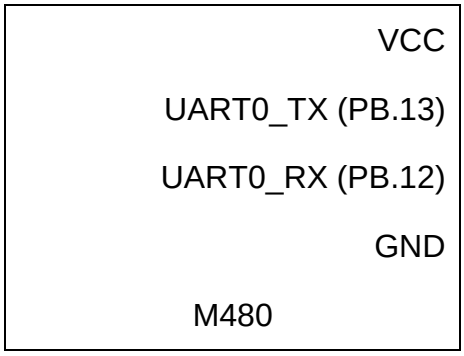
Software Environment

- BSP version
  - M480 Series BSP CMSIS V3.03.000
- IDE version
  - Keil uVersion 5.26

Hardware Environment

- Circuit components
  - NuMaker-ETM-M487
- Diagram

Using printf function to output message and the UART0 TX RX pin are PB.13 and PB.12 respectively.



4 Directory Information

- EC_M480_M4_Bit-Banding_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 M480_M4_Bit-Banding.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. 12, 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.