XOM Configuration Manual

Application Note for 32-bit NuMicro® Family

Document Information

Abstract	Introduce XOM related address and registers configuration, and how to set or remove XOM region.
Apply to	NuMicro® M261 series.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

Table of Contents

1 Overview

In the NuMicro® M261 series, XOM (Execute-Only Memory) is a secure ROM region which forbids any data access. However, the code stores in the XOM region still could be executed by CPU due to it is accessed by instruction fetch. By forbidden data access path, the code in XOM could be protected from reading by CPU, DMA, ICE and Flash programing tool. By allow instruction fetching, the code still could be executed in XOM region. Furthermore, the M261 supports MOVW, MOVT instructions to encode data within a 32 bits instruction. Therefore, it is easy to build a "pure instruction code" in "C" language with a compiler which supporting "execute-only code" compiler option.

In general, XOM could be used to embed a software library into the M261. Due to the characteristic of XOM, the software library in it could be protected from disassemble and copy.

A XOM size can be set to the minimum size is 1 page (2 Kbytes) on the M261, and the maximum can set up to 255 pages (510 Kbytes). In addition, the M261 also allows users to set up to 4 XOM regions, as shown in Figure 1-1. This feature enables users to use XOM on the M261 freely and flexibly.

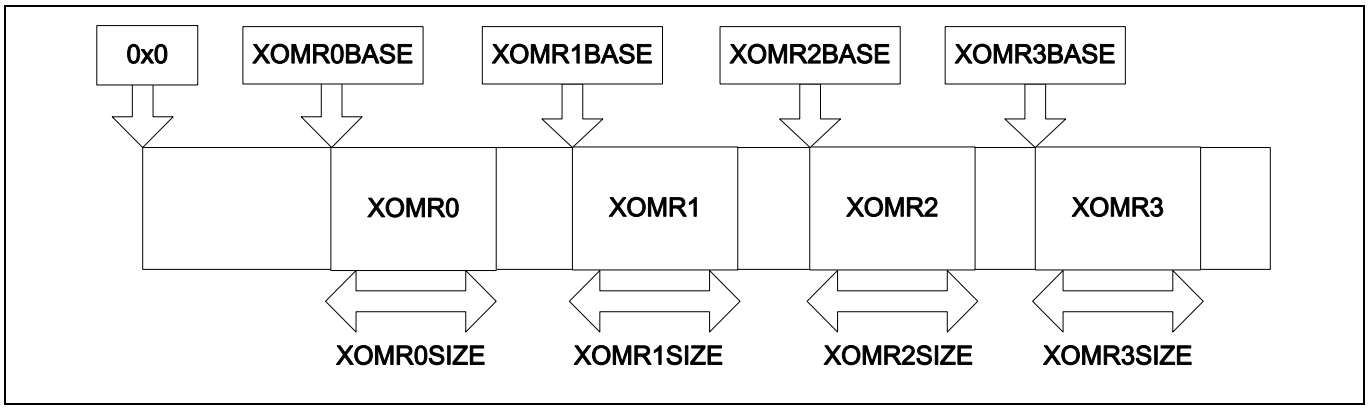

Figure 1-1 XOM Configuration

1.1 XOM Related Configure Address and Registers

The XOM region 0~3 configure address is shown in Table 1-1.

XOM Base	Address	XOM Size	Address	XOM CTRL	Address
XOMR0BASE	0x20_0000	XOMR0SIZE	0x20_0004	XOMR0CTRL	0x20_0008
XOMR1BASE	0x20_0010	XOMR1SIZE	0x20_0014	XOMR1CTRL	0x20_0018
XOMR2BASE	0x20_0020	XOMR2SIZE	0x20_0024	XOMR2CTRL	0x20_0028
XOMR3BASE	0x20_0030	XOMR3SIZE	0x20_0034	XOMR3CTRL	0x20_0038

Table 1-1 XOM Configure Address

- XOMRnBASE (n=0~3)
  It is XOM Region Base Address configuration. XOMRnBASE is used to set XOM region base address in APROM region.

- XOMRnSIZE (n=0~3)
  It is XOM Region Size configuration. XOMRnSIZE is used to set XOM size from 1 to 255 pages (2k ~510k bytes). The minimum setting is 1 page.

- XOMRnCTRL (n=0~3)
  It is XOM Region Control configuration, as shown in Table 1-2. XOMRnCTL is used to activate XOM region. The 0x5A is default value indicating that the XOM region is currently in OFF mode. If user writes non-0x5A to this configuration, the XOM region will be enabled and in normal mode. If user writes 0x50 to this configuration, the XOM region is in debug mode.

XOM Mode	XOM Active	XOM Region Control
Off mode	No	0x5A(default value)
Debug mode	Yes	0x50
Normal mode	Yes	Others

Table 1-2 XOM Control Configuration

After setting XOMRnBASE, XOMRnSIZE and XOMRnCTRL, the M261 series must reset to activate XOM regions.

FMC XOM status registers are listed in Table 1-3.

Register	Address
FMC_XOMR0STS	0x4000_C0D0
FMC_XOMR1STS	0x4000_C0D4
FMC_XOMR2STS	0x4000_C0D8
FMC_XOMR3STS	0x4000_C0DC
FMC_XOMSTS	0x4000_C0E0

Table 1-3 XOM Status Registers

- FMC_XOMRnSTS (n=0~3)
  It is XOM Setting Status register. These registers indicate the XOM region base address and page size.

- FMC_XOMSTS
  It is XOM Active Status register. This register indicate the active status of XOMRn (n=0~3)

2 Using BSP FMC ISP APIs to Configure XOM

2.1 Using FMC ISP APIs to Set and Activate XOM Region

Use the FMC ISP function to configure the XOM base address, size and active control on the M261. Follow the steps below to configure XOM region:

1. Execute SYS_UnlockReg() to unlock register protection.
2. Use FMC_Open() and FMC_ENABLE_AP_UPDATE() of BSP APIs to enable FMC ISP and APROM update functions.
3. Use the FMC_ConfigXOM() API to configure XOMRn (n=0~3), base address, XOM size and XOM active.
4. Program the user code into XOM region.
5. Reset chip to enable the XOM region.

2.2 Using FMC ISP APIs to Remove XOM Region

Follow the steps below to remove XOM region and settings:

1. Execute SYS_UnlockReg() to unlock register protect
2. Use FMC_Open() and FMC_ENABLE_AP_UPDATE() of BSP APIs to enable FMC ISP and APROM update functions
3. Use the FMC_EraseXOM() API to erase XOMRn (n=0~3) region and settings.
4. Do chip reset to remove XOM region and settings.

3 Using Keil® MDK to Configure XOM

Before using Keil® MDK to configure XOM, please install the latest version of Nuvoton Keil® driver first. The driver version needs to be v6650 or later.

3.1 Using Keil® MDK to Configure XOM Region

Follow the steps below to configure XOM region settings:

1. Open Keil® MDK and select “Project->Options for Target” to enter “Options” page, as shown in Figure 3-1.

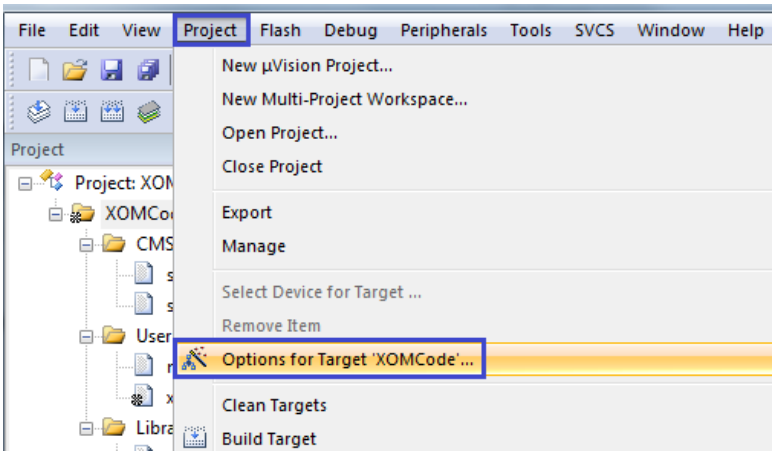

Figure 3-1 Open Keil® MDK Options

2. Click “Option”->”Debug”->”Settings” button to enter the debug setting page then select ”Chip Type” as M261, as shown in Figure 3-2.

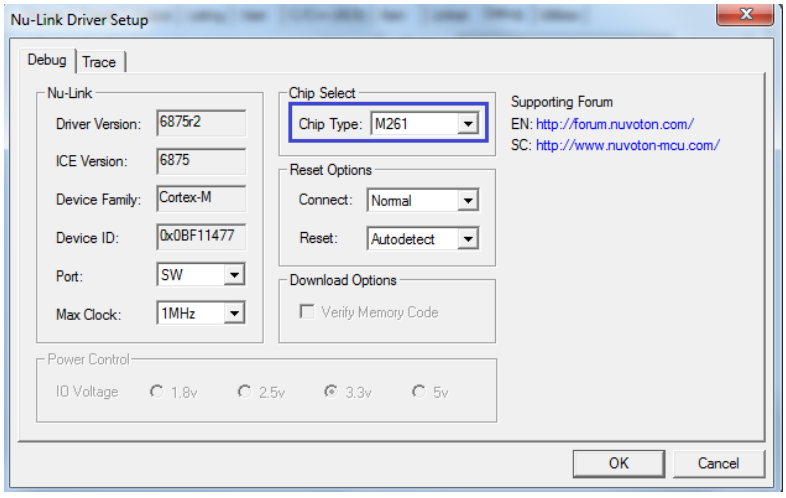

Figure 3-2 Select Chip Type

3. Click “Option”->”Utilities”->”Settings” button to enter Flash Download page. Click the “Setting” button in the “Chip Setting” to enter the “M261 Settings” page, as shown in Figure 3-3.

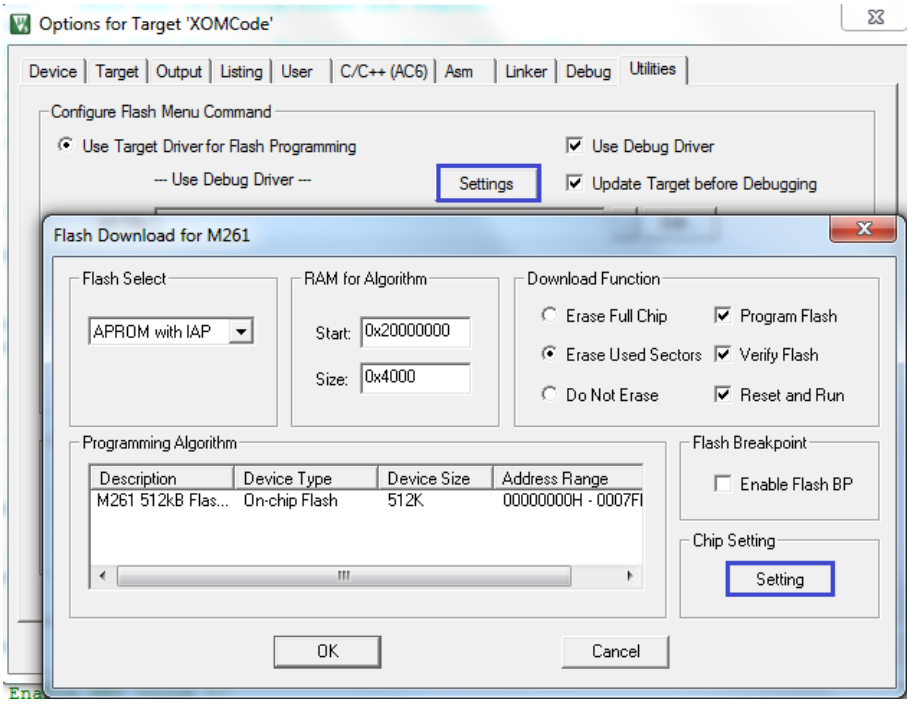

Figure 3-3 M261 Settings

4. After entering the “M261 Settings” page, select “XOM setting” page to enter XOM setting page, as shown in Figure 3-4.
5. Check XOM0~3 to set XOM “Start Address” and “Page Counts”. After configuring XOM setting then click “OK”.
6. To enable debug mode, check the “Debug Mode” option and repeat above steps.

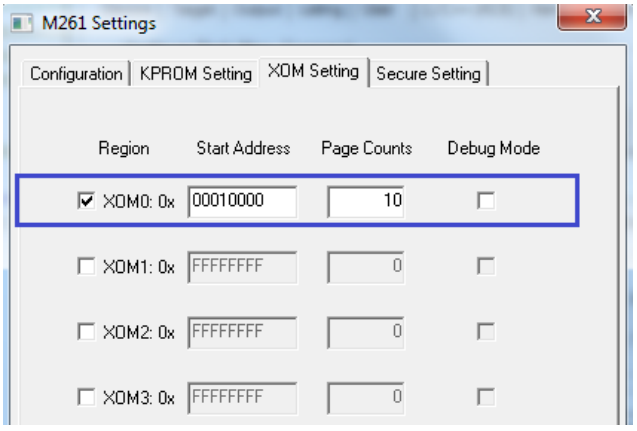

Figure 3-4 Keil® MDK XOM Setting

4 Using ICP Programming Tool to Configure XOM

Before using the ICP programming tool to configure XOM, please install the latest version of ICP programming tool first. The version needs to be v6650 or later.

4.1 Using ICP Programming Tool to Configure XOM Region

Follow the steps below to enter the XOM configure page in the ICP programming tool:

1. Open the ICP programming tool, select chip type “M261” and click “Connect” to enter the main page.
2. Click “Setting” button to open the “M261 Settings” page, as shown in Figure 4-1.

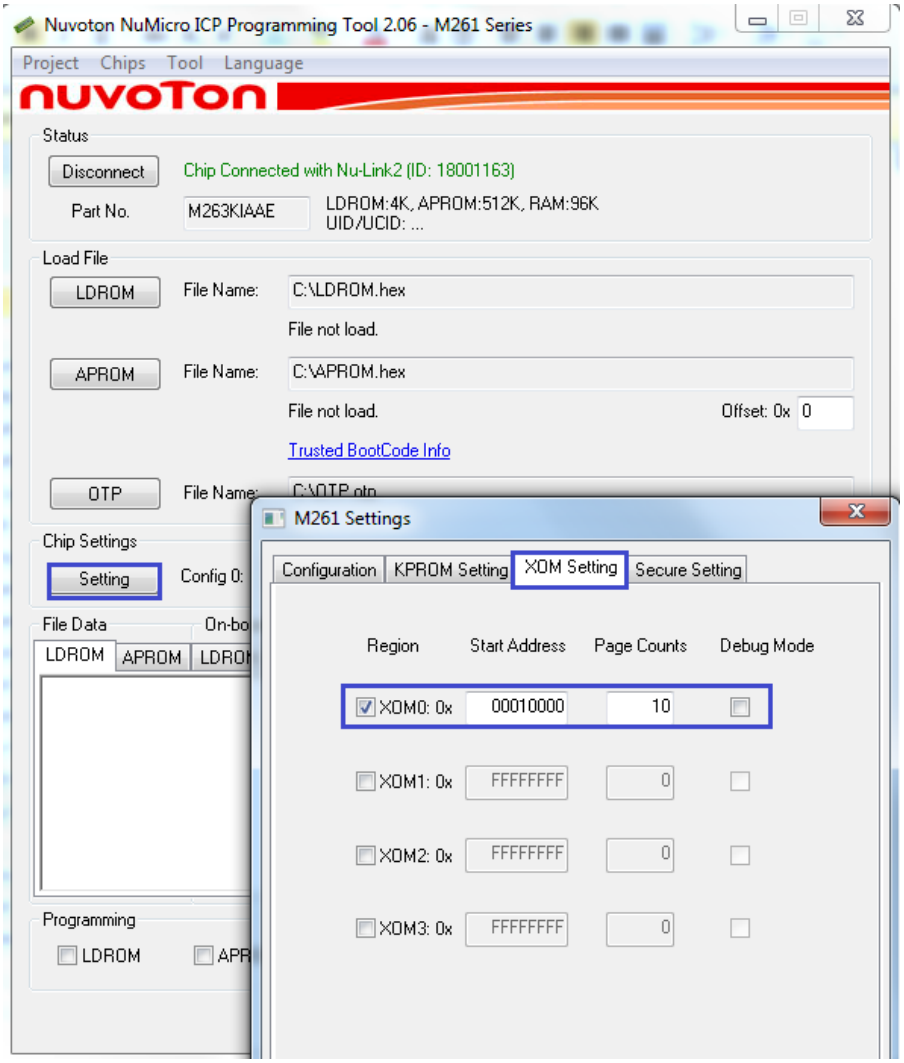

Figure 4-1 Open M261 Settings and XOM Setting

3. Enter “XOM Setting” page.
4. Check which XOM0~3 to fill the XOM “Start Address”, “Page Counts”. After configuration is finished, click “OK” to return to main page.
5. Click “Options” button to enter the programming setting page, as shown in Figure 4-2.

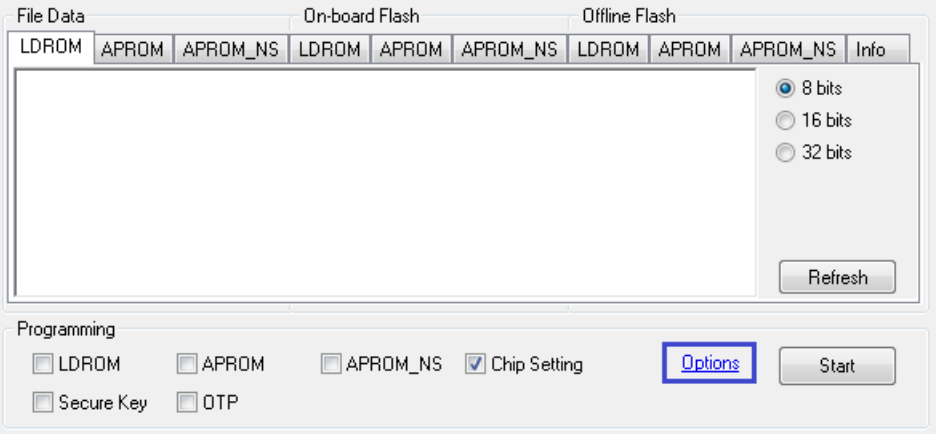

Figure 4-2 Programming Setting

6. Select “Erase”,” Program”, “Verify”, and “Reset Chip after Programming” options, as shown in Figure 4-3.
7. Click “OK” to return to main page.

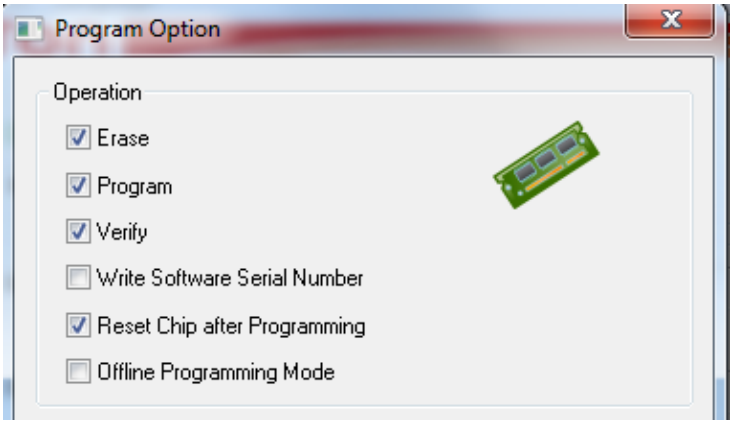

Figure 4-3 Program Option Select

8. Select “Chip Setting” and click “Start” button to program XOM configure settings and enable XOM, as shown in Figure 4-4.

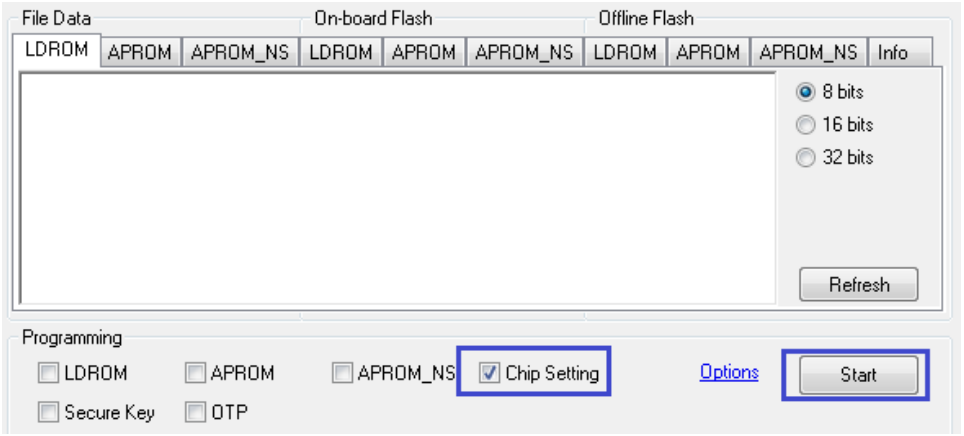

Figure 4-4 Start Programming

9. Re-enter “XOM Setting” page and uncheck the XOM select option, then return to main page.
10. Click “Refresh” and then re-enter “XOM Setting” page to check XOM configure status, as shown in Figure 4-5.
11. To enable debug mode, please check the “Debug Mode” option and repeat above steps.

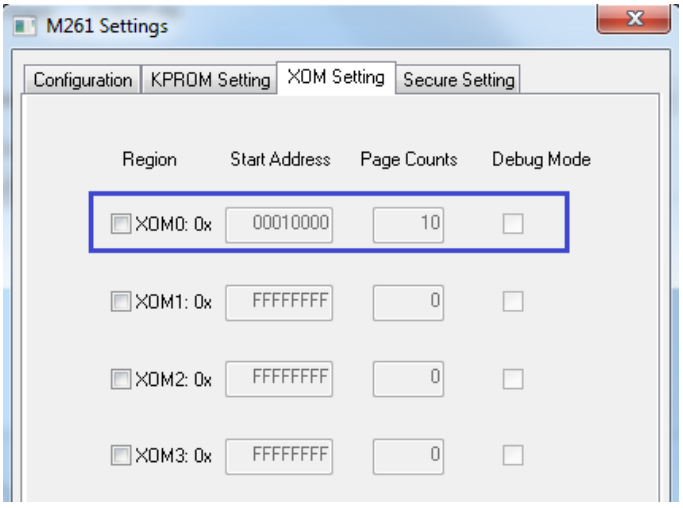

Figure 4-5 Check XOM Status

4.2 Using ICP Programming Tool to Remove XOM Region

1. Open the ICP programming tool, select chip type “M261” and click “Connect” to enter the main page.
2. Click “Setting” button to open the “M261 Settings” page, as shown in Figure 4-6.

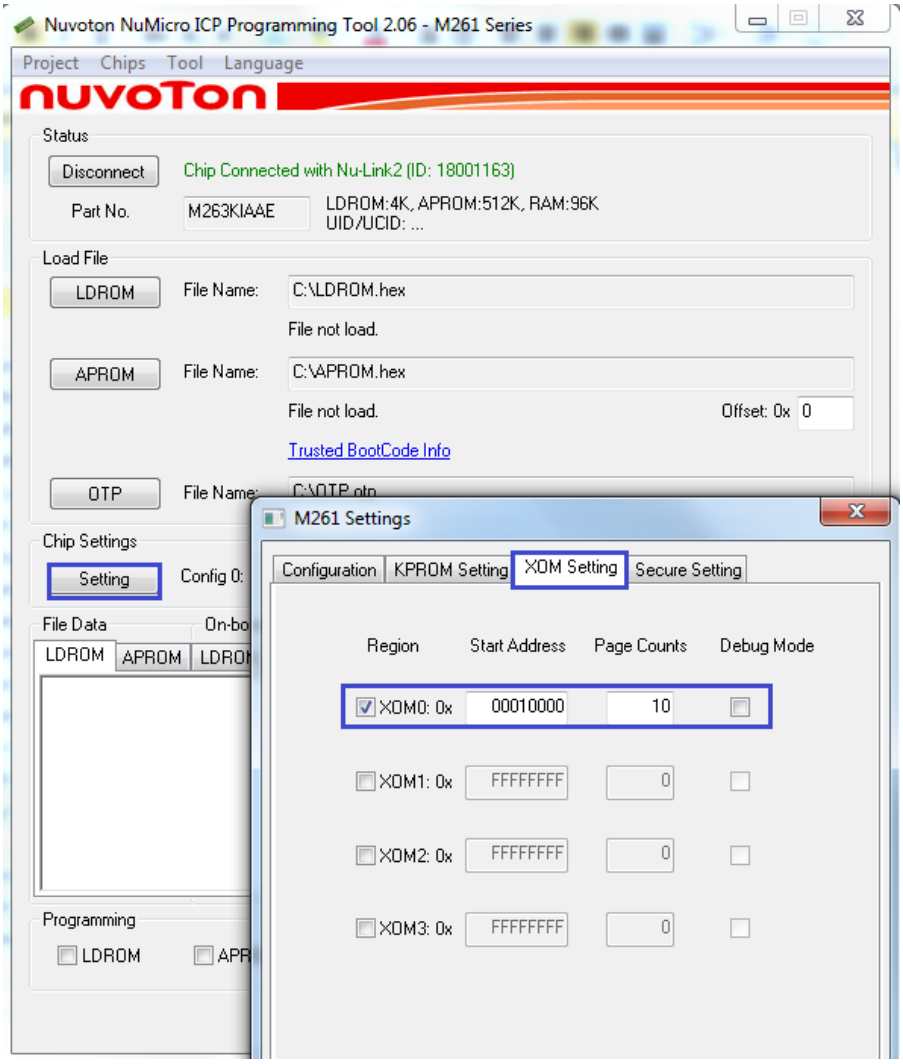

Figure 4-6 Open M261 Settings and XOM Setting

3. Enter “XOM Setting”.
4. Check which XOM region to be removed, and then click “OK” to return to the main page.
5. Select “Option” to enter “program option” page, as shown in Figure 4-7.
6. Select “Erase” and “Verify” options.
7. Click “OK” to return to the main page.

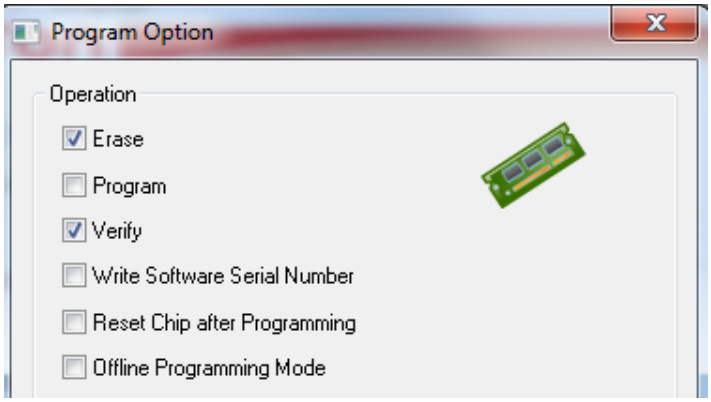

Figure 4-7 Program Option Selection

8. Select “Chip Setting” and click “Start” button to remove XOM region data and settings.
9. Enter “XOM Setting” page and uncheck XOM0.
10. Click “OK” to return to main page.
11. Click “Refresh” button then re-enter “XOM setting” page to check XOM configure status, as shown in Figure 4-8.

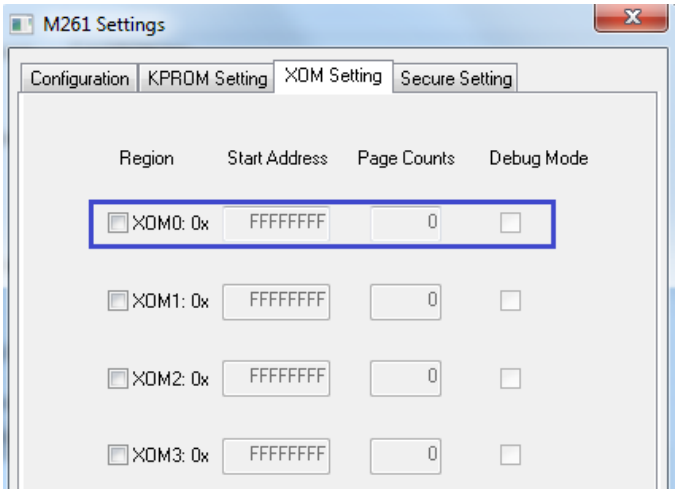

Figure 4-8 Check XOM Status

4.3 Using Erase Whole Target Chip Function to Remove XOM

The XOM region can also be removed using the erase whole chip function, “Tool”-> “Erase Whole Target Chip” in the ICP programming tool, as shown in Figure 4-9. If this function is used to remove XOM, all the XOMs region, user CONFIG0~3 and all the code and data on the Flash will be cleared.

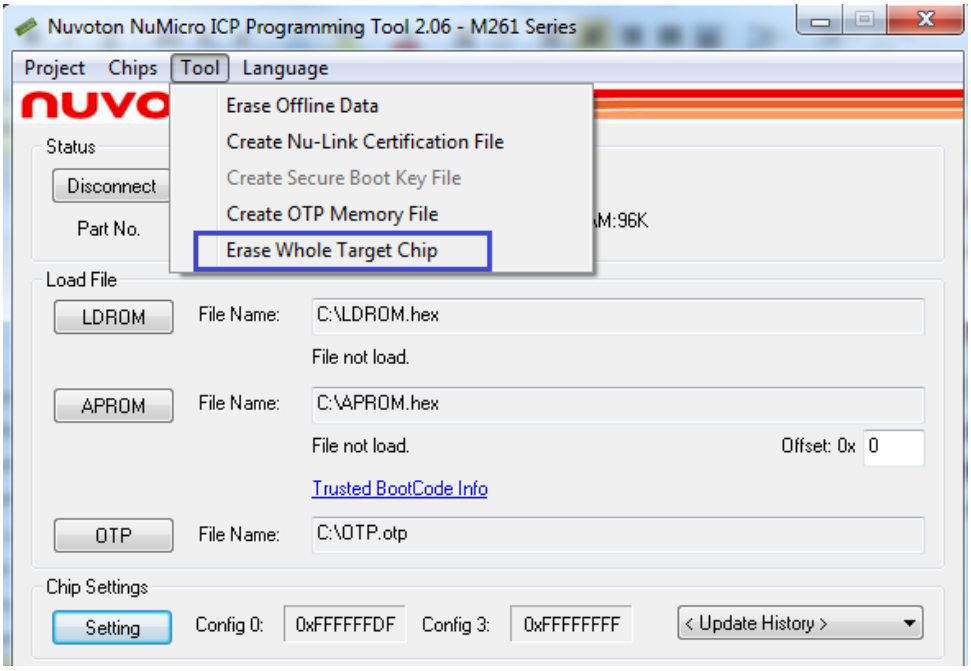

Figure 4-9 Erase Whole Chip Selection

5 Configuring XOM and Debugging on Keil® Compiler

5.1 Compiling Execute-only Code

Use Keil® compiler to compile .c file of XOM code. The .c file needs to be set to “Execute-only code” type. Follow the steps below, as shown in Figure 5-1:

1. Select .c file of XOM code on the Keil® “Project” window.
2. Open “Options” page of the .c file.
3. In “Option” -> “C/C++ (AC6) “-> “Language/Code Generation”, select “Execute-only code” option.

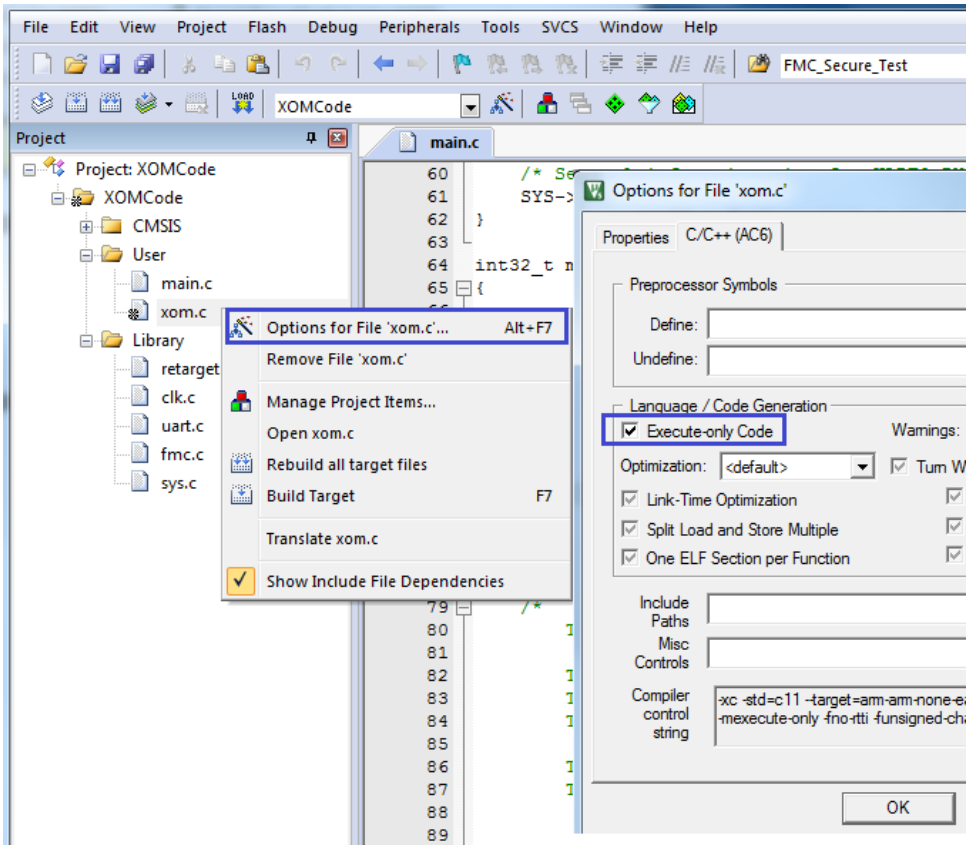

Figure 5-1 Compile .c File of XOM Code

5.2 Writing a Scatter File to Specify XOM Address

User can easily assign and program code to any address on Flash by using linker script, a scatter file description. User can open and edit scatter file by “Project” -> “Options of target” -> “Linker” -> “Edit”, as shown in Figure 5-2. When compiling a *.c file,* the compiler will assign the object code (.o) to specified address on Flash according to the scatter file description.

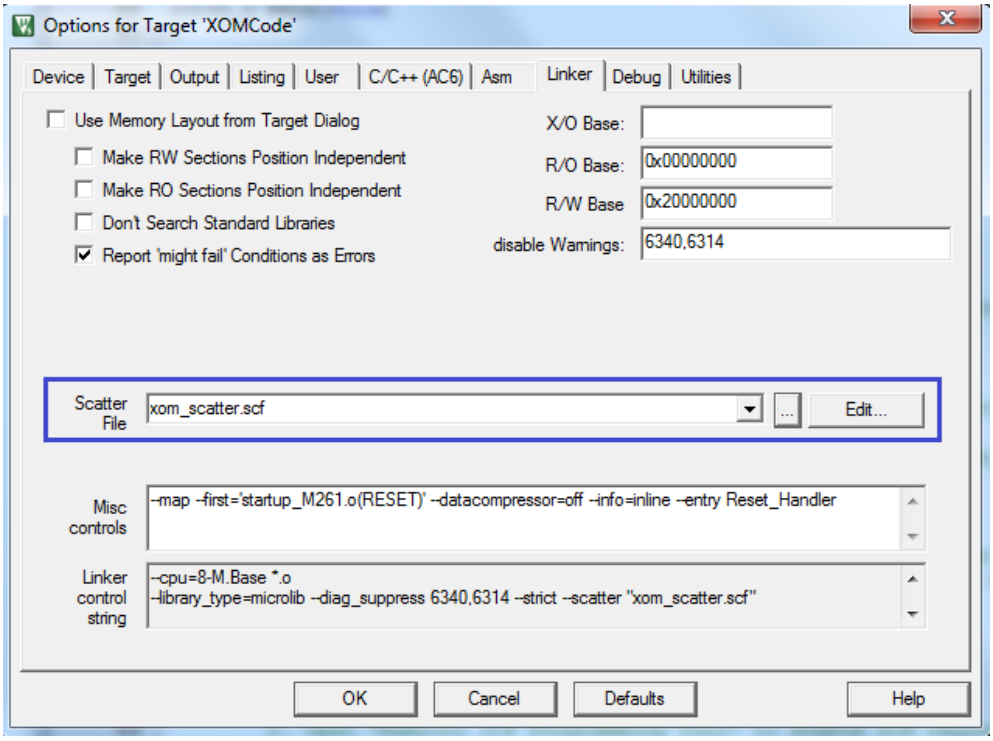

Figure 5-2 Open and Edit Scatter File

Below is a scatter file example. It declares a XOM0 region located at 0x10000 and assigns xom.o to this region.

```
; Scatter file

; XOM0 region base address
XOM0 0x10000
{
   ;xom.o base address in XOM0 region
   XOM0_ROM +0
   {
      xom.o
   }
}
```

5.3 Debugging XOM Region

Write a value other than 0x5A or write 0x50 to XOMRnCTL (n=0~3) control configuration address and reset chip, the XOM region will be enabled and enter normal mode (not 0x5A) or debug mode (0x50). In XOM debug mode, user can trace code by Step-Over or Step-Into to debug the code behavior. This is helpful to locate illegal data access to XOM region which is caused by the code in XOM region. In other words, the XOM debug mode forbids data access but doesn't forbid breakpoint in XOM region while XOM normal mode will forbid both data access and breakpoint.

XOM is a protect region which cannot be read in disassembly window. All values returned from XOM region by data read are 0xffff_ffff, as shown in Figure 5-3.

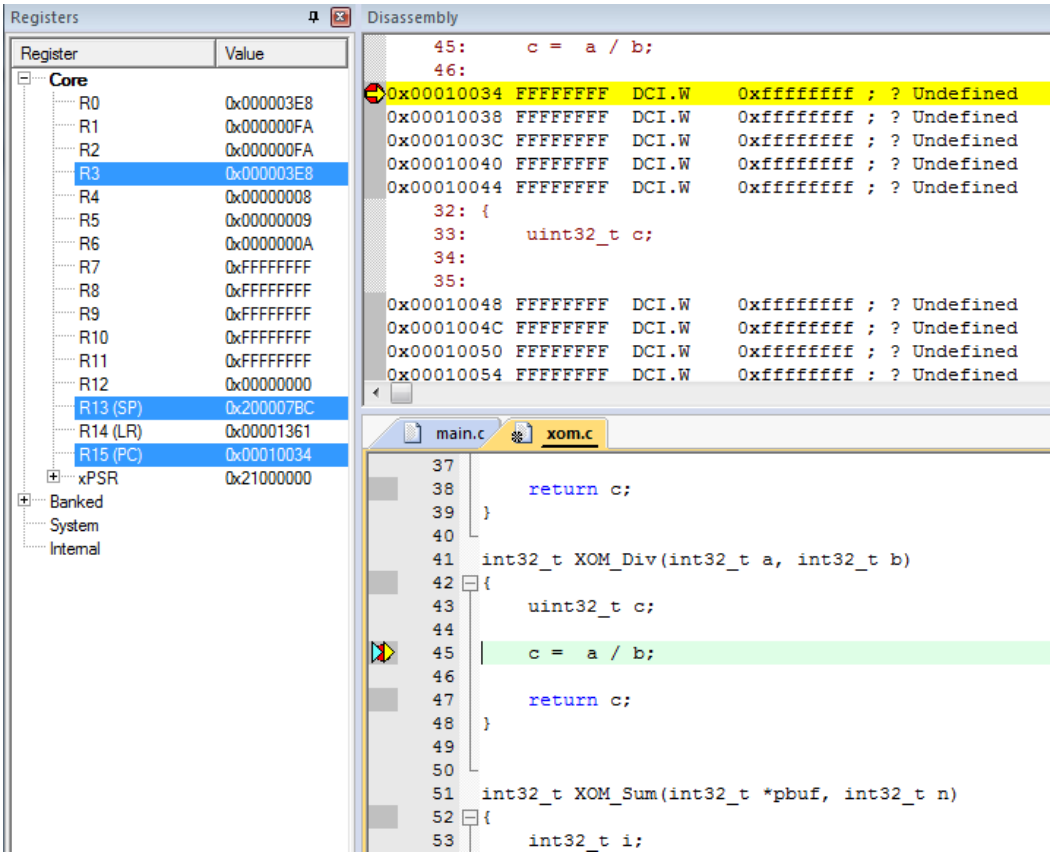

Figure 5-3 Trace XOM Code in Debug Mode

To make sure the code in XOM region is well protected, user must re-set XOM to normal mode after debugging. When the XOM region is in normal mode, the XOM region cannot be traced. It is forbidden to step into or set breakpoint in XOM region when XOM in normal mode, as shown in Figure 5-4.

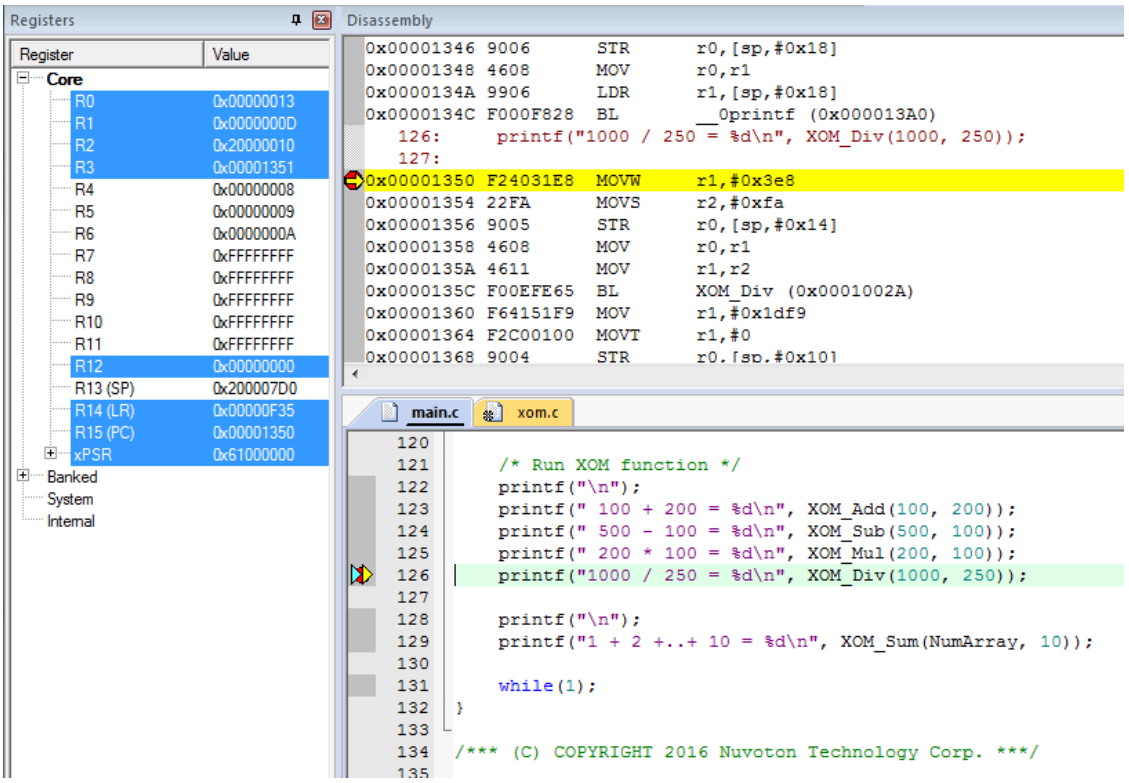

Figure 5-4 Trace XOM Code in Normal Mode

Note: If the code is trapped or executed too long in XOM region under normal mode, it may cause ICE error as ICE is forced to be disabled when code is running in a locked XOM region.

6 Precautions on Using XOM

Users need to pay attention to the following when using XOM:

1. XOM region base address can only be set in APROM and XOM region cannot be set over APROM.
2. Configuring XOM region needs to set XOMRnBASE，XOMRnSIZE and XOMRnCTRL (n= 0~3) at the same time. After finishing the configuration, chip has to reset to enable XOMs.
3. To disable an activated XOM region, it needs XOM page erase or whole chip erase. XOM page erase will erase the specified XOM region, and whole chip erase will erase all XOM regions.
4. It is recommended to write 0x0 to XOMRnCTRL (n=0~3) to activate XOM region.
5. Removing XOM region needs to use XOM page erase and fill correct XOM base address. After doing chip reset, the XOM region and settings will be cleared.
6. XOM regions cannot be overlapped.
7. XOM region base address must be page alignment.
8. Never set XOM region based to 0x0000_0000. Otherwise, it will fail to boot from APROM.
9. Because data read from XOM region will be 0xffff_ffff when XOM is activated, the “Verify Memory Code” of the IDE tool should be disabled when developing code within an activated XOM region.
10. When configuring XOM is failed, e.g. non-page alignment base address, chip needs to do “Whole Chip Erase” to remove all XOM region and settings.

7 XOM Sample Code

7.1 Sample Code Description

The FMC_XOMLib sample code demonstrates how to compile and build XOM library. There are two project files in FMC_XOMLib: XOMCode.uvprojx and XOMLib.uvprojx. FMC_XOMCode is used to build the code for XOM region.

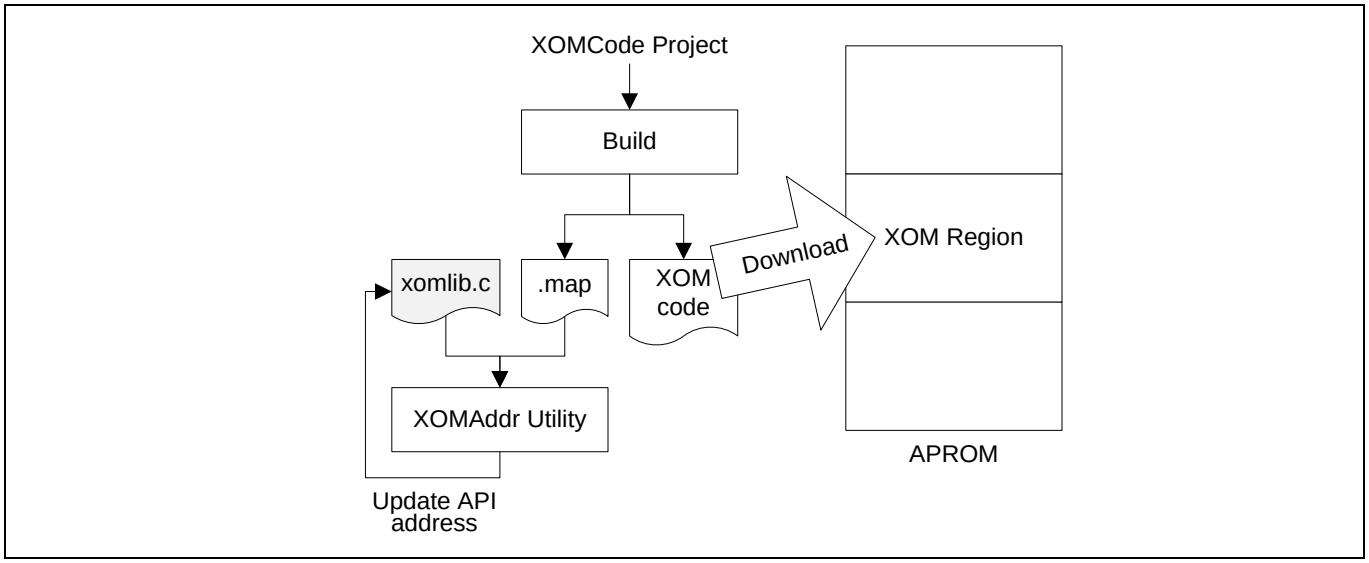

Figure 7-1 Build XOMCode Project

After XOMCode.uvprojx compilation is done, the IDE tool will execute XOMAddr.exe utility. The XOMAddr.exe refers to \list\XOMCode.map to update the API's address in xomlib.c in \XOMLib\, as shown in Figure 7-1.

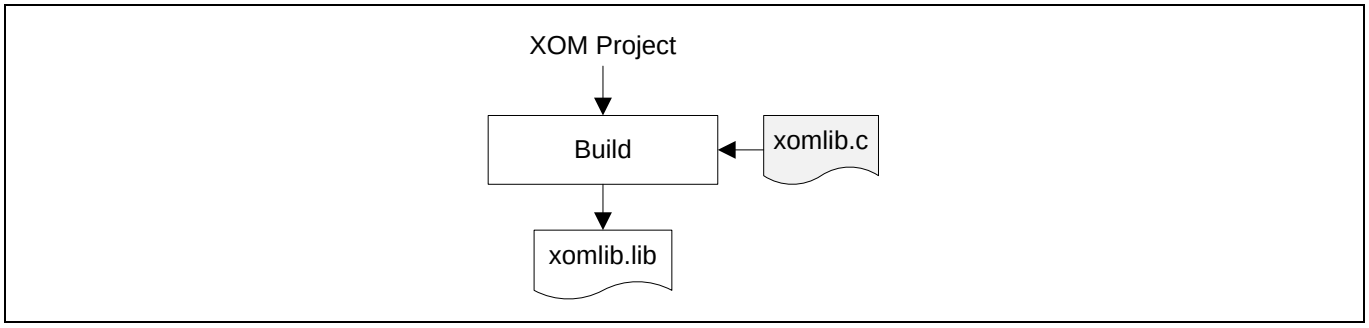

Figure 7-2 Build XOMLib Project

XOMLib is used to build the XOM library which could be included with other projects. XOMLib.uvprojx compiles xomlib.c and results in a xomib.lib file in \XOMLib\lib\, as shown in Figure 7-2. The FMC_XOMLibDemo sample code will use xomlib.h and xomlib.lib to call the functions in XOM region.

In FMC_XOMCode project, the code of XOM library in xom.c file option must be set as “Execute-only code“. The location of XOM library (xom.o) address is assigned at 0x10000 in xom_scatter.scf file. After compiling and downloading the XOMCode project, the XOM region is not enabled yet. User can enable XOM region manually by the methods mentioned in above chapters.

For example, use Options->Utilities->Settings->Setting->XOM Setting to set XOM base address and XOM size, as shown in Figure 7-3. After the XOM configure settings, user can execute chip reset to enable XOM region.

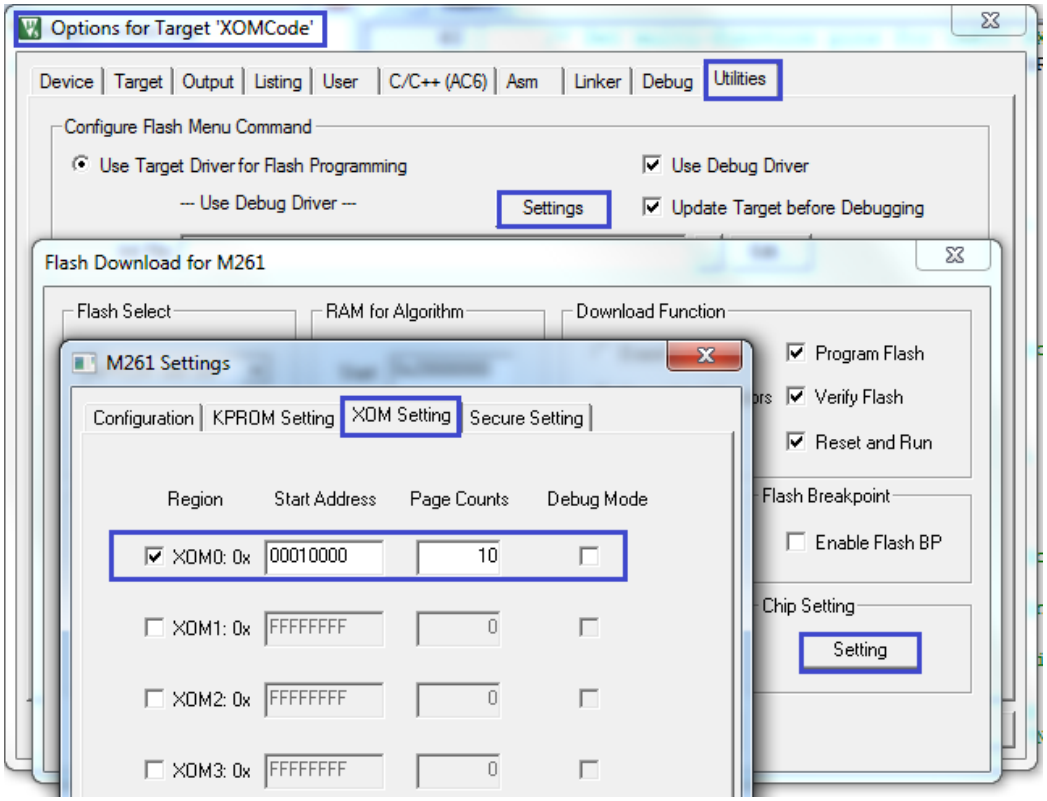

Figure 7-3 Use Keil® MDK to Set XOM Region

7.2 XOMCode Project

The XOMCode project is a complete project and can be executed alone. It is used to develop and test the XOM code. In main.c bellow, it initializes the MCU and call XOM functions to test them.

main.c

```
#include <stdio.h>
#include "M261.h"
#include "xomapi.h"

#define PLL_CLOCK       6400000
```

```
void SYS_Init(void)
{
    . . . .
}

int32_t main(void)
{
    uint32_t i, u32Data, u32Status, ret;
    int32_t NumArray[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

    /* Unlock protected registers */
    SYS_UnlockReg();

    /* Init System, IP clock and multi-function I/O. */
    SYS_Init();

    /* Configure UART: 115200, 8-bit word, no parity bit, 1 stop bit. */
    UART_Open(UART0, 115200);

    /*
        This sample code is used to show how to build an XOM library.

        The location of XOM region is defined by scatter file: xom_scatter.scf
        The API header file is xomapi.h
        The XOM functions are implemented in xom.c

        This project is only used to build code for XOM region and test its functions.
        To enable XOM region, please use "NuMicro ICP Programming Tool".

        example flow:
        1. Build XOMlib
        2. Download XOMlib code by press key "F8" in Keil® MDK.
        3. Open "NuMicro ICP Programming Tool" to enable XOM region and according to
xom_scatter.scf settings.
        4. The library (xomlib.o) and header (xomlib.h) is located at lib directory.
        5. Pass xomlib.o & xomlib.h to the people who will call the functions in XOM.

    */

    printf("\n\n");
    printf("+----------------------------------------+\n");
```

```
    printf("|      FMC XOM Library Build Example     |\n");
    printf("+----------------------------------------+\n");

    /* Unlock protected registers */
    SYS_UnlockReg();

    /* Enable FMC ISP function and enable APROM active*/
    FMC_Open();
    FMC_ENABLE_AP_UPDATE();

    /* Read User Configuration */
    printf("\n");
    printf("XOM Status = 0x%X\n", FMC->XOMSTS);

    /* Run XOM function */
    printf("\n");
    /* Call and execute XOM library APIs */
    printf(" 100 + 200 = %d\n", XOM_Add(100, 200));
    printf(" 500 - 100 = %d\n", XOM_Sub(500, 100));
    printf(" 200 * 100 = %d\n", XOM_Mul(200, 100));
    printf("1000 / 250 = %d\n", XOM_Div(1000, 250));

    printf("\n");
    printf("1 + 2 +..+ 10 = %d\n", XOM_Sum(NumArray, 10));

    while(1);
}
```

xom_scatter.scf

Specify the location of the XOM.o object file in XOM0 region.

```
; XOM0 region base address at 0x10000
XOM0 0x10000
{
   XOM0_ROM +0
   {
      ; Execute region of xom.o from 0x10000
      xom.o
   }
   . . . .
   . . . .
```

```
}
```

XOMAPI.h

Declare XOM library APIs header file.

```
#ifndef _XOMAPI_H_
#define _XOMAPI_H_

int32_t XOM_Add(int32_t a, int32_t b);
int32_t XOM_Sub(int32_t a, int32_t b);
int32_t XOM_Mul(int32_t a, int32_t b);
int32_t XOM_Div(int32_t a, int32_t b);
int32_t XOM_Sum(int32_t *pbuf, int32_t n);

#endif
```

xom.c

The XOM_Add(), XOM_Sub(), XOM_Mul(), XOM_Div(), XOM_Sum() are XOM library APIs. Because the code of xom.c needs to download to XOM region, this xom.c file must set “Execute-only code” compiler option to generate execute-only code, as shown in Figure 7-4.

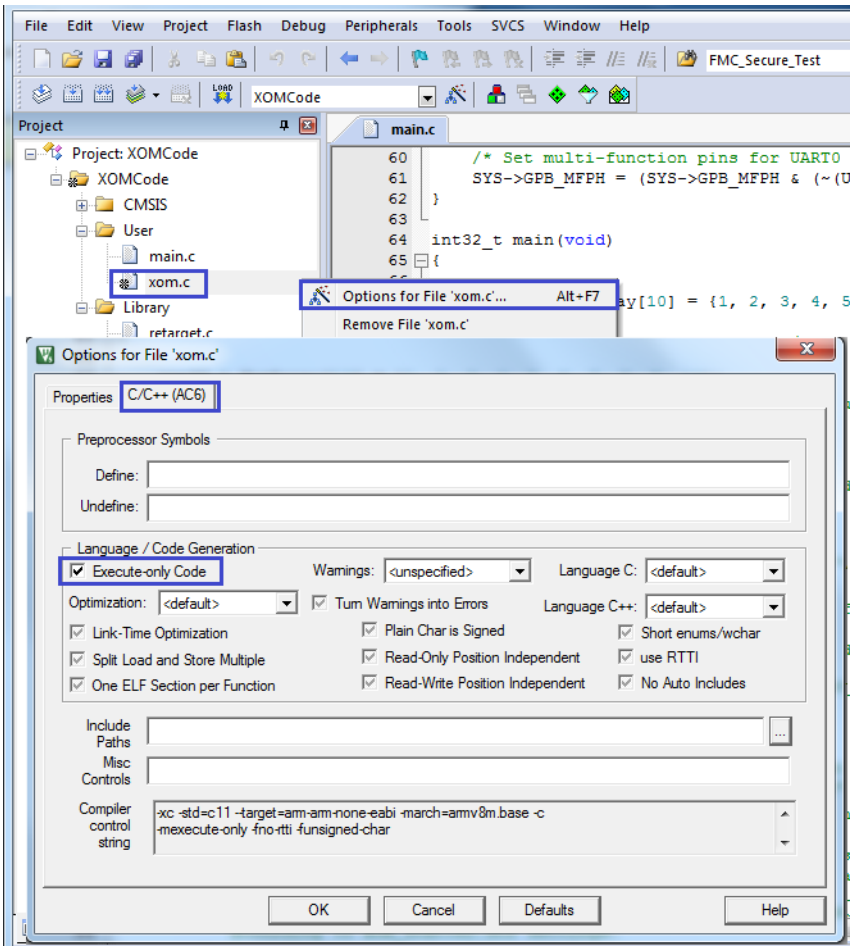

Figure 7-4 Generate Execute-Only Code

```
#include <stdio.h>
#include "M261.h"

int32_t XOM_Add(int32_t a, int32_t b)
{
    uint32_t c;

    c =  a + b;

    return c;
}

int32_t XOM_Sub(int32_t a, int32_t b)
{
    uint32_t c;
```

```
    c =  a - b;

    return c;
}

int32_t XOM_Mul(int32_t a, int32_t b)
{
    uint32_t c;

    c =  a * b;

    return c;
}

int32_t XOM_Div(int32_t a, int32_t b)
{
    uint32_t c;

    c =  a / b;

    return c;
}

int32_t XOM_Sum(int32_t *pbuf, int32_t n)
{
    int32_t i;
    int32_t i32Sum;

    i32Sum = 0;
    for(i=0;i<n;i++)
    {
        i32Sum += pbuf[i];
    }

    return i32Sum;
}
```

7.3 FMC_XOMLibDemo

The FMC_XOMLibDemo sample code demonstrates how to compile and link xomlib.lib which is generated from FMC_XOMLib. Before test FMC_XOMLibDemo project, XOM code must has been downloaded and activated in FMC_XOMCode project.

To call XOM functions from a new project, it should do:

1. First, add xomlib.lib to project and set the file location of xomlib.h in “Option” -> “C/C++ (AC6)” -> “Include Paths”, the compiler can find these two files when execute compile.
2. Include xomlib.h header file in main.c.
3. Call the XOM APIs.

Main.c

```
#include <stdio.h>
#include "M261.h"
#include "xomlib.h"

#define PLL_CLOCK       6400000

void SYS_Init(void)
{
    . . . .
}

int32_t main(void)
{
    extern void func(void);
    uint32_t i, u32Data, u32Status, ret;

    /* Unlock protected registers */
    SYS_UnlockReg();

    /* Init System, IP clock and multi-function I/O. */
    SYS_Init();

    /* Configure UART0: 115200, 8-bit word, no parity bit, 1 stop bit. */
    UART_Open(UART5, 115200);

    /*
        This sample code is used to show how to call XOM libary.
        The XOM library is built by XOMLib project.
        User need to add include path of xomlib.h and add object file xomlib.o to use XOM
library built by XOMLib project.
    */

    printf("\n\n");
```

```
    printf("+----------------------------------------+\n");
    printf("|  Demo how to use XOM library Sample Code  |\n");
    printf("+----------------------------------------+\n");

    /* Run XOM function */
    printf("\n");
    /* Call and execute XOM library APIs */
    printf(" 100 + 200 = %d\n", XOM_Add(100, 200));
    printf(" 500 - 100 = %d\n", XOM_Sub(500, 100));
    printf(" 200 * 100 = %d\n", XOM_Mul(200, 100));
    printf("1000 / 250 = %d\n", XOM_Div(1000, 250));

    printf("\n");
    printf("1 + 2 +..+ 10 = %d\n", XOM_Sum(NumArray, 10));

    while(1);
}
```

xomlib.h

XOM APIs library header file generates from compile FMC_XOMLib sample code.

```
/**************************************************************************
 * This file is auto-generated by Nuvoton XOM tool.
 * Copyright (C) 2016 Nuvoton Technology Corp. All rights reserved.
**************************************************************************/
#ifndef _XOMLIB_H_
#define _XOMLIB_H_

/* Include extern XOM library APIs header */
extern int32_t (*XOM_Add)(int32_t a, int32_t b);
extern int32_t (*XOM_Div)(int32_t a, int32_t b);
extern int32_t (*XOM_Mul)(int32_t a, int32_t b);
extern int32_t (*XOM_Sub)(int32_t a, int32_t b);
extern int32_t (*XOM_Sum)(int32_t *pbuf, int32_t n);

#endif
```

7.4 FMC_XOM

This sample code shows how to use ISP command to configure or remove XOM region.

The sample code is located in bsp\SampleCode\StdDriver folder. The procedure to use this sample code is as follows:

1. Compile sample code and download code to chip
2. Reset chip
3. Configure XOMR0 settings and reset chip to active XOMR0
4. Call Lib_XOM_ADD() function in XOMR0 region.
5. Remove XOMR0 setting and clear XOMR0 data.

main.c

```
#include <stdio.h>
#include "M261.h"

#define PLL_CLOCK       6400000

#define XOMR0_Base    0x10000

extern int32_t Lib_XOM_ADD(uint32_t a, uint32_t b);

void SYS_Init(void)
{
    . . . .
}

int32_t main(void)
{
    uint32_t i, u32Data, u32Status, ret;

    /* Unlock protected registers */
    SYS_UnlockReg();

    /* Init System, IP clock and multi-function I/O. */
    SYS_Init();

    /* Lock protected registers */
    SYS_LockReg();

    /* Configure UART0: 115200, 8-bit word, no parity bit, 1 stop bit. */
    UART_Open(UART0, 115200);

    /*
        This sample code is used to show how to use StdDriver API to enable/erase XOM.
```

```
*/

printf("\n\n");
printf("+--------------------------------------+\n");
printf("|  FMC XOM config & erase  Sample Code   |\n");
printf("+--------------------------------------+\n");

/* Unlock protected registers */
SYS_UnlockReg();

/* Enable FMC ISP function and enable APROM active*/
FMC_Open();
FMC_ENABLE_AP_UPDATE();

. . . .

printf("XOM Status = 0x%X\n", FMC->XOMSTS);
printf("Any key to continue...\n");
getchar();

/* Config XOMR0 */
if(FMC_GetXOMState (XOMR0) == 0)
{
   /* Configure XOMR0 base address and size */
   u32Status = FMC_ConfigXOM(XOMR0, XOMR0_Base, 1);
   if(u32Status)
      printf("XOMR0 Config fail...\n");
   else
      printf("XOMR0 Config OK...\n");

   printf("\nAny key to reset chip to enable XOM regions...\n");
   getchar();

   /* Reset chip to enable XOM region. */
   SYS_ResetChip();
   /* After chip reset, XOMR0 will be enabled*/
   while(1) {};
}

/* Run XOM function */
printf("\n");
```

```
    u32Data = Lib_XOM_ADD(100, 200);
  printf(" 100 + 200 = %d\n", u32Data);

  printf("\nXOMR0 active success....\n");
  printf("\nAny key to Erase XOM...\n");
  getchar();

  if((FMC->XOMSTS & 0x1) == 0x1)
  {
    /* Erase XOMR0 region and settings */
    if(FMC_EraseXOM(XOMR0) == 0)
      printf("Erase XOMR0....OK\n");
    else
      printf("Erase XOMR0....Fail\n");

    printf("\nAny key to reset chip...\n");
    getchar();

    /* Reset chip to finish erase XOM region. */
    SYS_ResetChip();
    /* After chip reset, XOMR0 will be cleared */
  }

  while(1);
}
```

XOM_Add.c

This is the XOM code for test in FMC_XOM project. The “Execute-only code” compiler option must be enabled.

```
#include <stdio.h>
#include "M261.h"

int32_t Lib_XOM_ADD(uint32_t a, uint32_t b)
{
  uint32_t c;
  c =  a + b;
  return c;
}
```

Xom_scatter.scf

It specifies the location of the xom_add.o object file in XOM0 region.

```
; Specified XOM0 base address at 0x10000
XOM0 0x10000
{
  XOM_ROM +0
  {
    ; Execution region of xom_add.o from 0x10000
     xom_add.o
  }
}
```

8 Conclusion

The XOM function in the NuMicro® M261 series provides a secure environment for project development. User can protect the source code not allowed to be accessed by setting the code region as a XOM region. No one can attempt the data in the XOM region. Such XOM feature ensures that the user can develop application more safely.

Revision History

Date	Revision	Description
2019.04.12	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.