M2351 Collaborative Secure Software Development

Application Note for 32-bit NuMicro® Family

Document Information

Abstract	Introduce the concept of Collaborative Secure Software Development (CSSD), and the precautions to be considered by the first and the second developer during development.
Apply to	NuMicro® M2351 series

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

www.nuvoton.com

Table of Contents

1 OVERVIEW 3

2 FIRST DEVELOPMENT 4

2.1 First Development Flow 4

2.2 SCRLOCK – Secure Region Lock 7

2.2.1Keil® MDK 7

2.2.2ICP Programming Tool 8

3 COLLABORATIVE DEVELOPMENT 10

3.1 Collaborative Development Flow 10

3.2 XOM – Execute-Only Memory 14

3.2.1Keil® MDK 14

3.2.2ICP Programming Tool 15

3.3 Download Code 16

3.3.1Keil® MDK 16

3.3.2ICP Programming Tool 19

3.4 ARLOCK – All Region Lock 20

3.4.1Keil® MDK 20

3.4.2ICP Programming Tool 21

4 SAMPLE CODE 22

4.1 Operation Procedure 22

4.2 Sample Code Description 22

4.2.1CSSD_LED – Secure 22

4.2.2CSSD_LED – NonSecure 27

5 CONCLUSION 30

1 Overview

In the past, software intellectual property (IP) provider needs to release their library to their customer. Then the customer links the library to build the firmware for end products. By this flow, it is hard to control how many end products produced by the customer, because the software IP has been embedded into the firmware and could be copied as many as customer's want. Since the library has been released to customer side, it is also hard to prevent the customer to reverse engineering the software IP from the library.

The NuMicro® M2351 series based on the Armv8-M architecture supports Arm® TrustZone® Technology to separate the M2351 into secure world and non-secure world. All services provided in the secure world could be called from the non-secure world but the services are never exposed in the view of non-secure world. It means the customer can never copy or trace the software IP in the secure world while the software IP providing their services. This development flow is called Collaborative Secure Software Development (CSSD). The software IP provider is called "first developer" and its customer is called "second developer".

The CSSD indicates that the second developer deploys APIs based on mature products provided by the first developer to expand features and speed up the development of a certain type of product. For example, the first developer provides fingerprint identification algorithm library, such that the second developer can use it for the applications of door lock, car lock and access control system, and so on.

To meet the Arm® TrustZone® technology, Flash memory controller (FMC) provides the secure and non-secure world with secure region and non-secure region respectively under the TrustZone® environment. In the M2351 series microcontroller (MCU), there are non-secure regions which can be dominated by the second developer and secure regions which cannot be dominated by the second developer. In secure regions, the first developer partitions a block used to place library, and the second developer can only call and execute the APIs in this block by non-secure callable. The overall concept is shown in Figure 1-1.

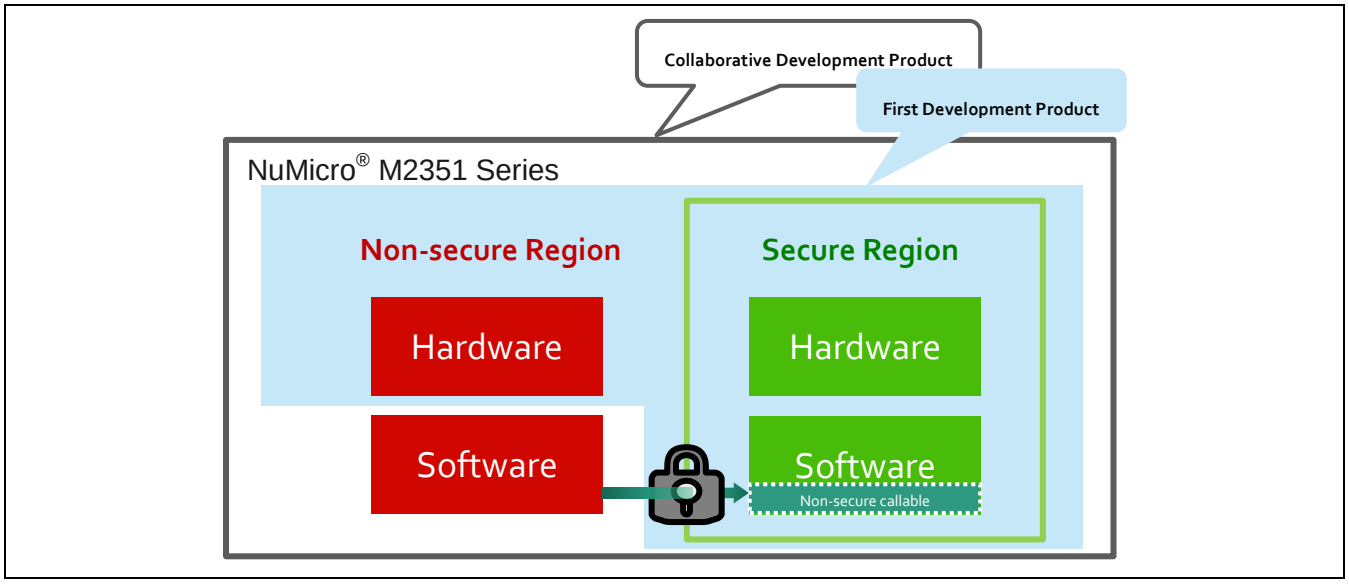

Figure 1-1 Concept of CSSD

2 First Development

This chapter describes the precautions on development flow to be considered by the first developer and the environment setup before, in, after or when secure region development is completed.

2.1 First Development Flow

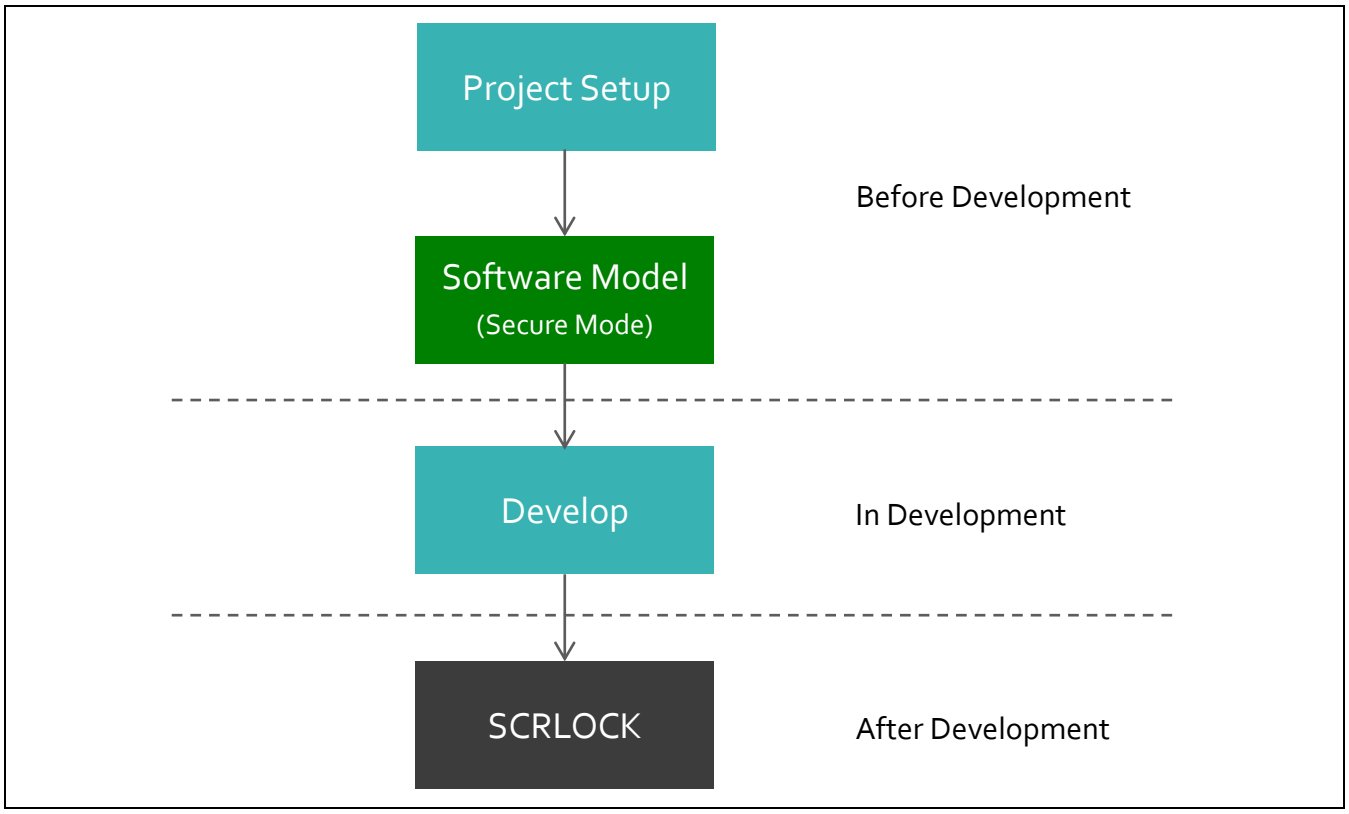

Figure 2-1 First Development Flow Chart

According to the development flow chart as shown in Figure 2-1, the development setup flow in Keil® MDK is as follows.

- Before development
  - Click [Options for Target – Target] as shown in Figure 2-2, and the first developer must check if the Software Model is in Secure Mode.

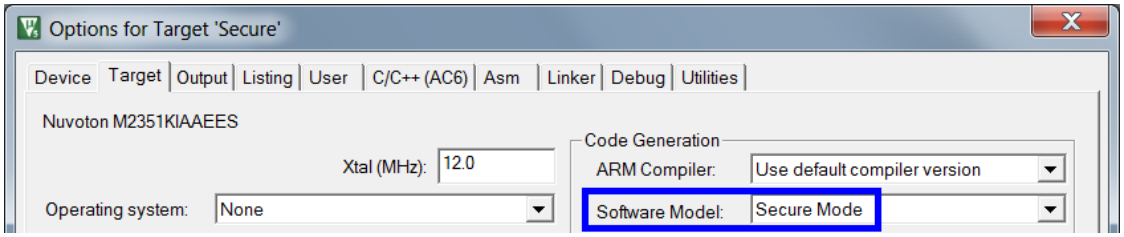

Figure 2-2 Check Software Model in Keil® MDK

- Click [Options for Target – Linker] as shown in Figure 2-3.
  - Edit secure script component file in Scatter File, which is used to define the range of secure APROM, secure SRAM and placing non-secure callable APIs.
  - Specify a path to save the object file generated by non-secure callable provided to the second developer in Misc controls.

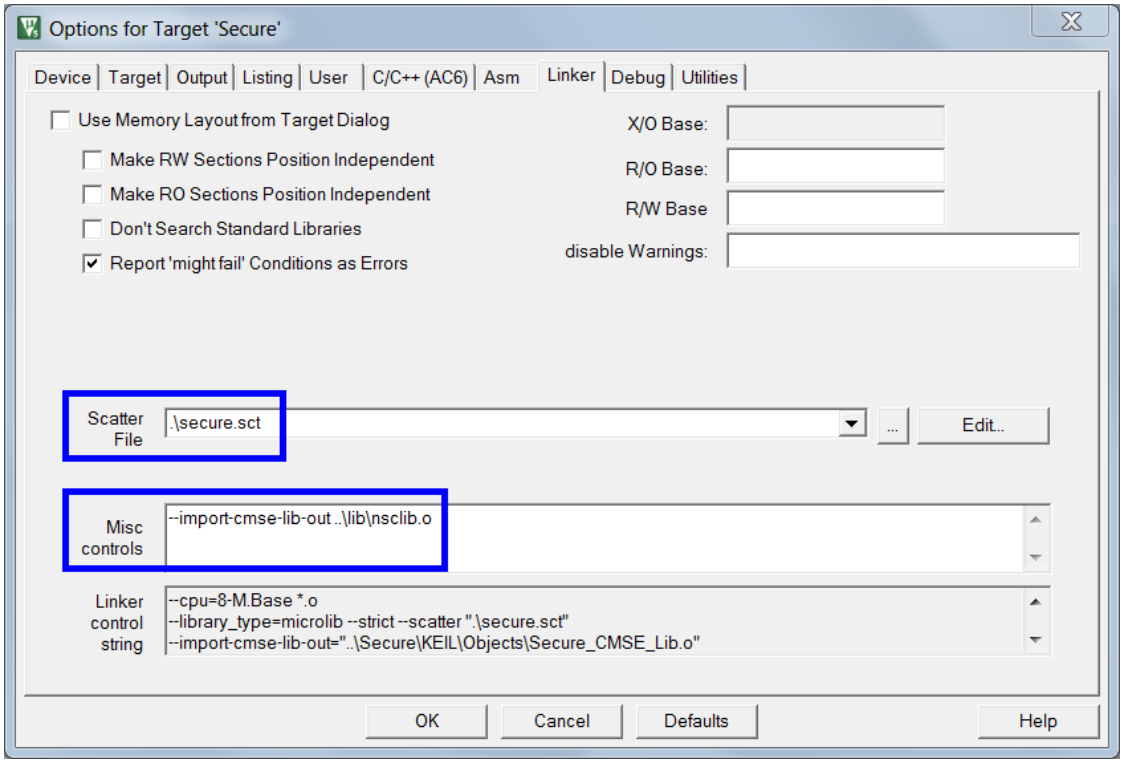

Figure 2-3 Specify Scatter File and Misc Controls in Keil® MDK

- In development

The first developer must configure security attributions in development. For example, the size of SRAM, the setup of non-secure region starting address, and the secure or non-secure configuration of peripheral and interrupt, etc. In addition, the first developer must use non-secure callable to pack APIs built in the peripheral that secure configured to non-secure use.

For details about the environment setup and the usage of non-secure callable with the first developer before or in development mentioned above, please refer to M2351 TrustZone® Program Development document.

- After development

To protect code in the secure region, the first developer must open SCRLOCK to avoid code from being tampered or destroyed in secure region. Section 2.2 will introduce the concept of SCRLOCK and how to set up environment in Keil® MDK and ICP Programming Tool.

- Development completed

Finally, when development is completed, the first developers must pack the following

information to the second developers, such that the second developers are able to develop.

- Non-secure APROM and SRAM base address
- The object file generated by non-secure callable
- Non-secure callable APIs header file

2.2 SCRLOCK – Secure Region Lock

If SCRLOCK address in 0x0020_0804 is not read as 0x5A when MCU detects there is ICE connected, the MCU will start read and write protection mechanism, namely user can only read 0xFFFF_FFFF through ICE in secure region, and all program to secure region will be ignored. It is noteworthy that the first developer can use this setup to protect code in secure region, and the second developer can only use library in secure region by calling non-secure callable functions but cannot know or re-write code in secure region. Additionally, the sizes of secure and non-secure regions are determined by the set value of non-secure boundary value, NSCBA in 0x0020_0800. Consequently, the second developers must pay attention to that the range of non-secure region they are able to develop is from NSCBA to the APROM size, 512KB.

The following shows the setting methods of NSCBA and SCRLOCK in Keil® MDK and ICP Programming Tool. In addition, please note that chip will do erase when setting NSCBA.

2.2.1 Keil® MDK

Open a project in the Keil® MDK environment.

1. Click [Options for Target – Utilities], and click [Settings] button to open Flash Download page. Then click [Setting] button under Chip Setting to open the Secure Setting page, as shown in Figure 2-4.
2. The method of setting NSCBA is that select [Non-secure region], specify Start Address and then click [OK]. Select [Secure Region Lock] and click [OK] and then secure region can be locked.

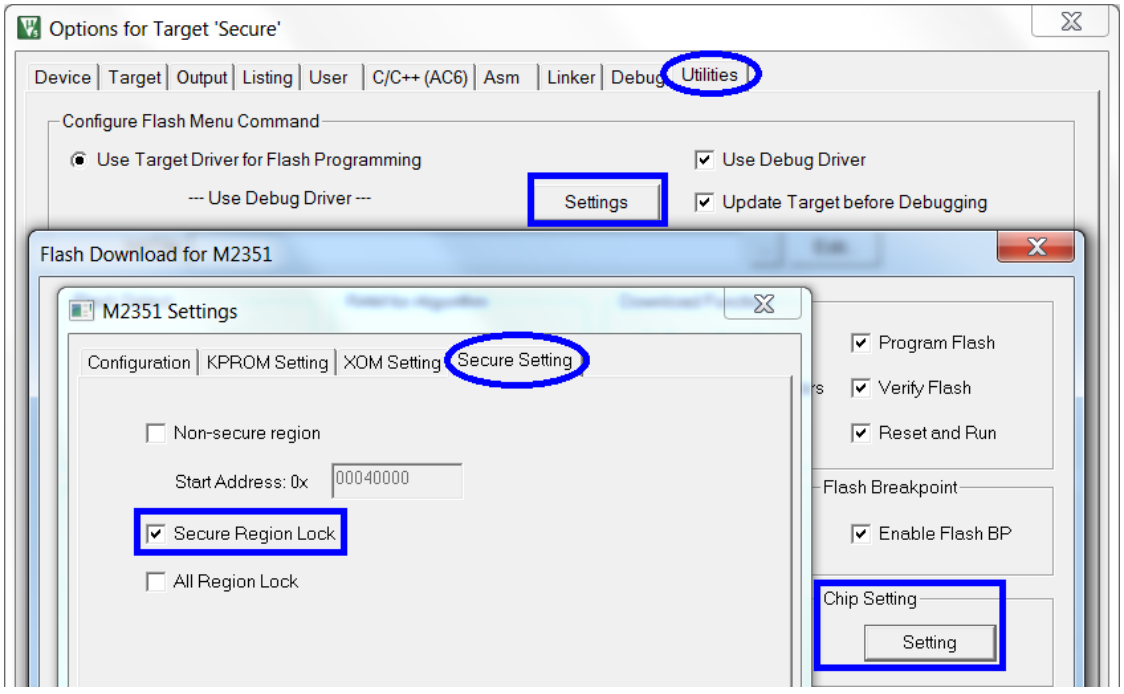

Figure 2-4 Setting Methods of NSCBA and SCRLOCK in Keil® MDK

2.2.2 ICP Programming Tool

After selecting a target chip correctly, connect the ICP Programming Tool with Nu-Link. When connected:

1. Click [Setting] button under Chip Settings to open the Secure Setting page, as shown in Figure 2-5.
2. Select [Non-Secure Region (APROM_NS)] and specify Start Address and then click [OK], or select [Secure Region Lock] and then click [OK].
3. Select [Chip Setting] under Programming and click [Start] to start programming, then the value of NSCBA can be set or secure region can be locked.

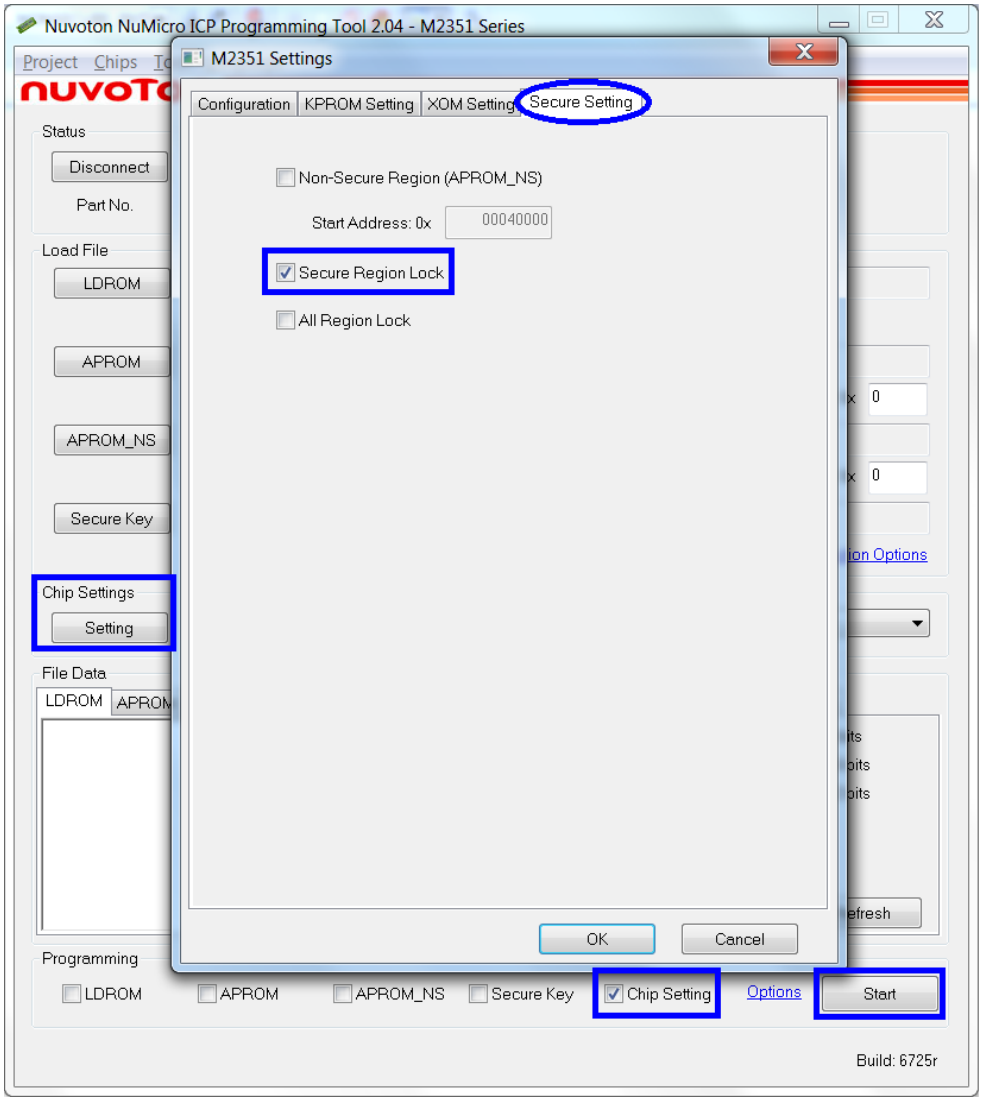

Figure 2-5 Setting Methods of NSCBA and SCRLOCK in ICP Programming Tool

If users find that they need to modify code after secure region is locked, in response to this situation, the ICP Programming Tool provides the function that can erase the whole chip. User just needs to click [Erase Whole Target Chip] in [Tool] as shown in Figure 2-6 and then SCRLOCK can be unlocked. But this action will erase whole data in this chip; therefore, the second developer must pay attention to not clicking this function item by mistake.

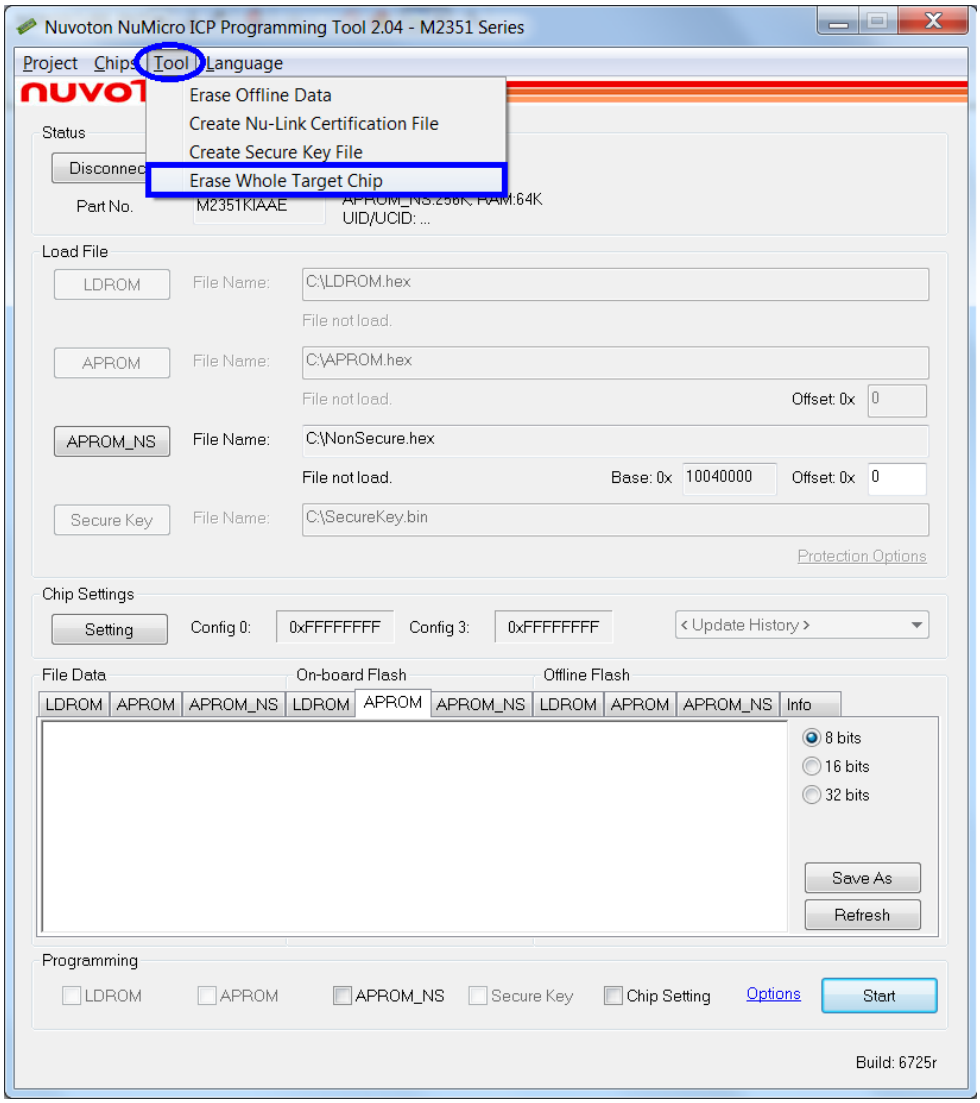

Figure 2-6 Erase Whole Target Chip in ICP Programming Tool

3 Collaborative Development

This chapter describes the concept of the collaborative development flow and the precautions including environment setup and restrictions on use before, in or after non-secure region development.

3.1 Collaborative Development Flow

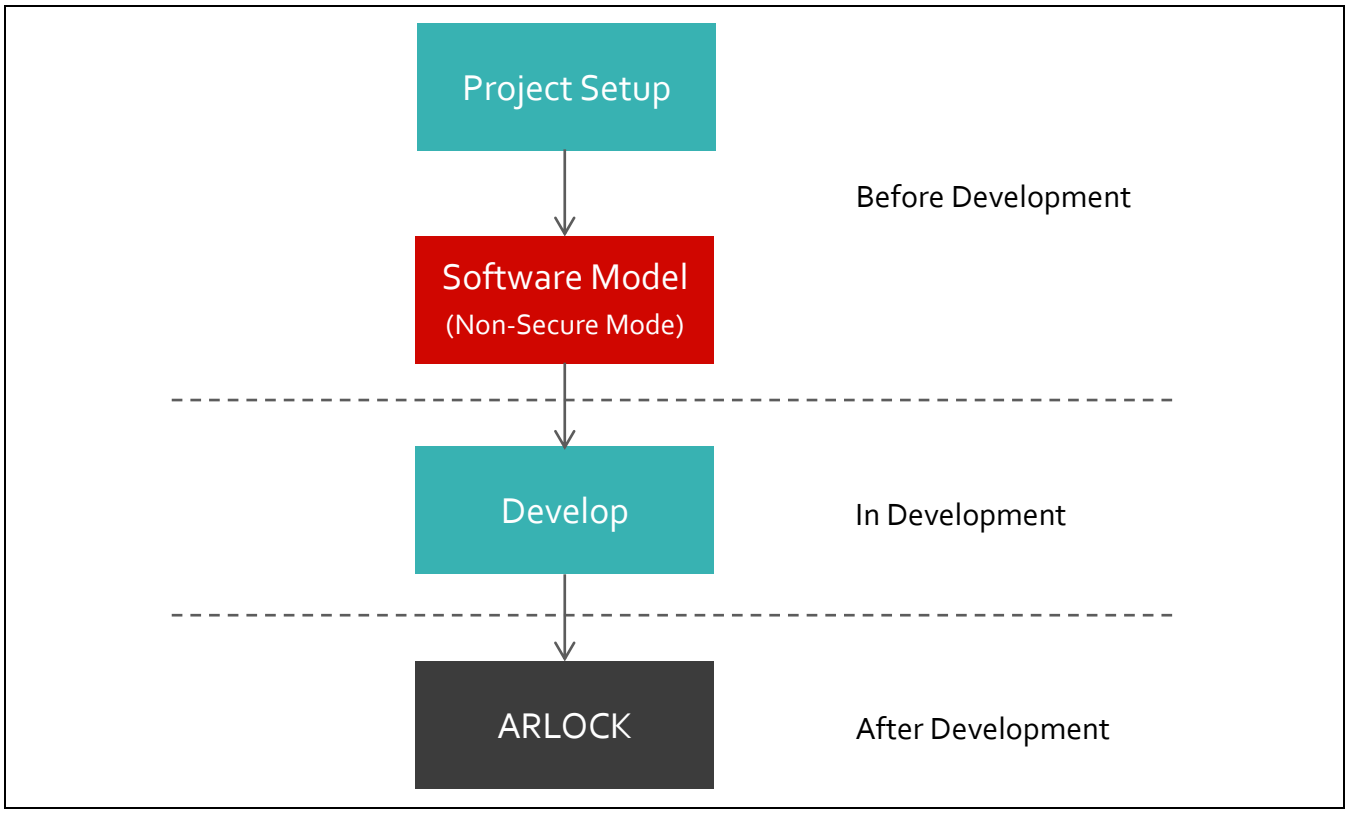

Figure 3-1 Collaborative Development Flow Chart

According to the development flow chart as shown in Figure 3-1, the development setup flow in Keil® MDK is as follows.

- Before development
  - Click [Options for Target – Target] as shown in Figure 3-2, and the second developer must check if the Software Model is in Non-Secure Mode.

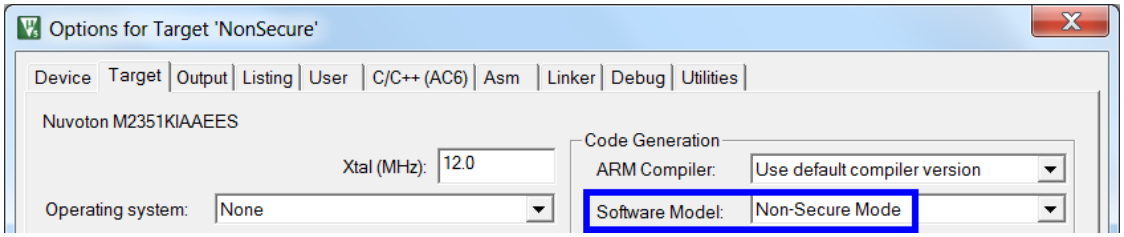

Figure 3-2 Check Software Model in Keil® MDK

- Click [Options for Target – Linker] as shown in Figure 3-3.
  - According to the first developer provides the information of non-secure APROM and SRAM base address, the second developer can edit non-secure script component file in Scatter File, which is used to define the range of non-secure APROM and non-secure SRAM.

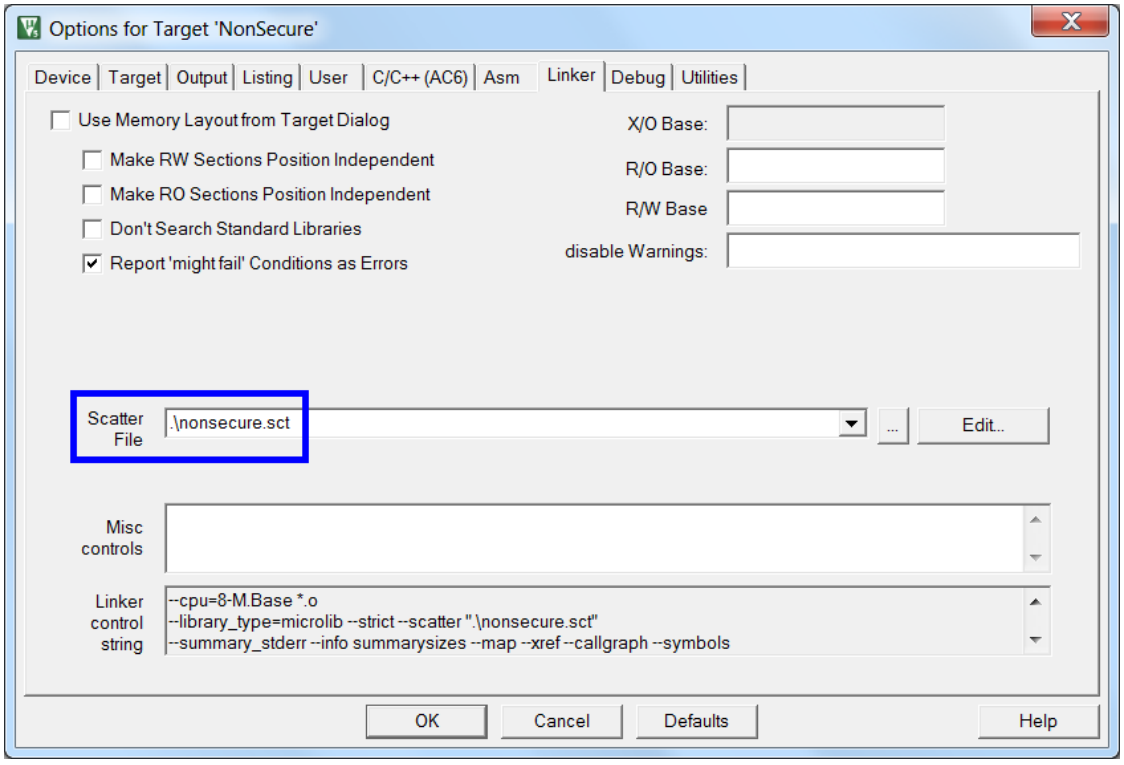

Figure 3-3 Specify Scatter File in Keil® MDK

- Click [Options for Target – Utilities] and click [Setting] button, and then Flash Download page will appear as shown in Figure 3-4.
  - The second developer can read the information that non-secure region range is from 0x1000_0000 to 0x1007_FFFF in the red box. In the later example, the first developer divides the first 256KB of APROM as secure region; therefore, the development range that can be programmed by the second developer is from 0x1004_0000 to 0x1007_FFFF.
  - In addition, regardless of whether secure region is locked by the first developer or not, all of the following need to be checked to ensure that ICE download is available.
    - Flash Select has been set to [APROM_NS].
    - Non-secure SRAM base address has been specified in the Start field under RAM for Algorithm.

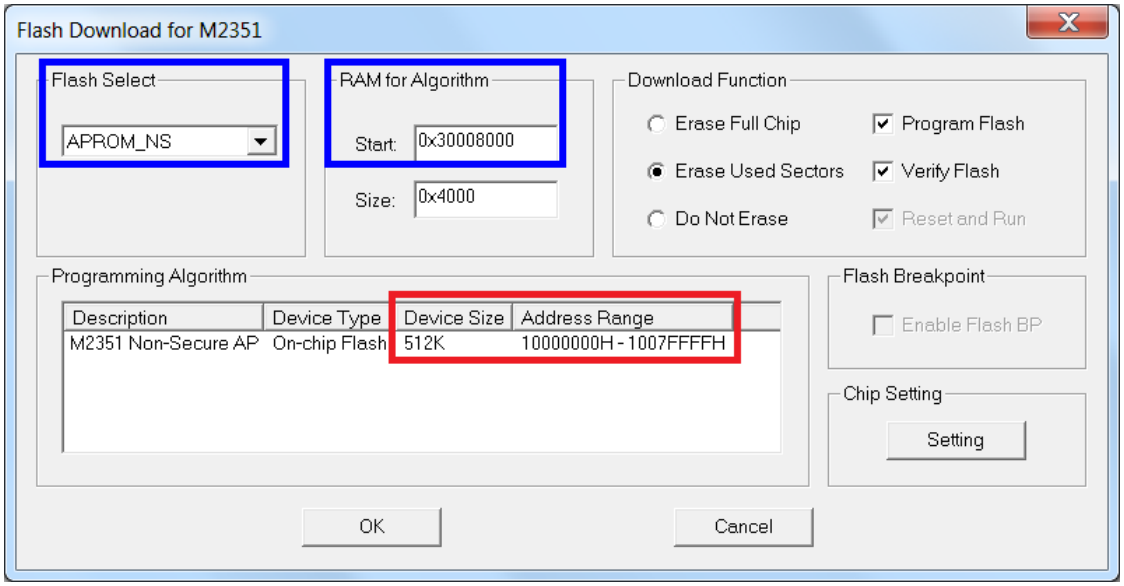

Figure 3-4 Check Download Setting in Keil® MDK

- Click [Manage Project Items] and then add an object file which is generated by non-secure callable provided by the first developer to this project, as shown in Figure 3-5.

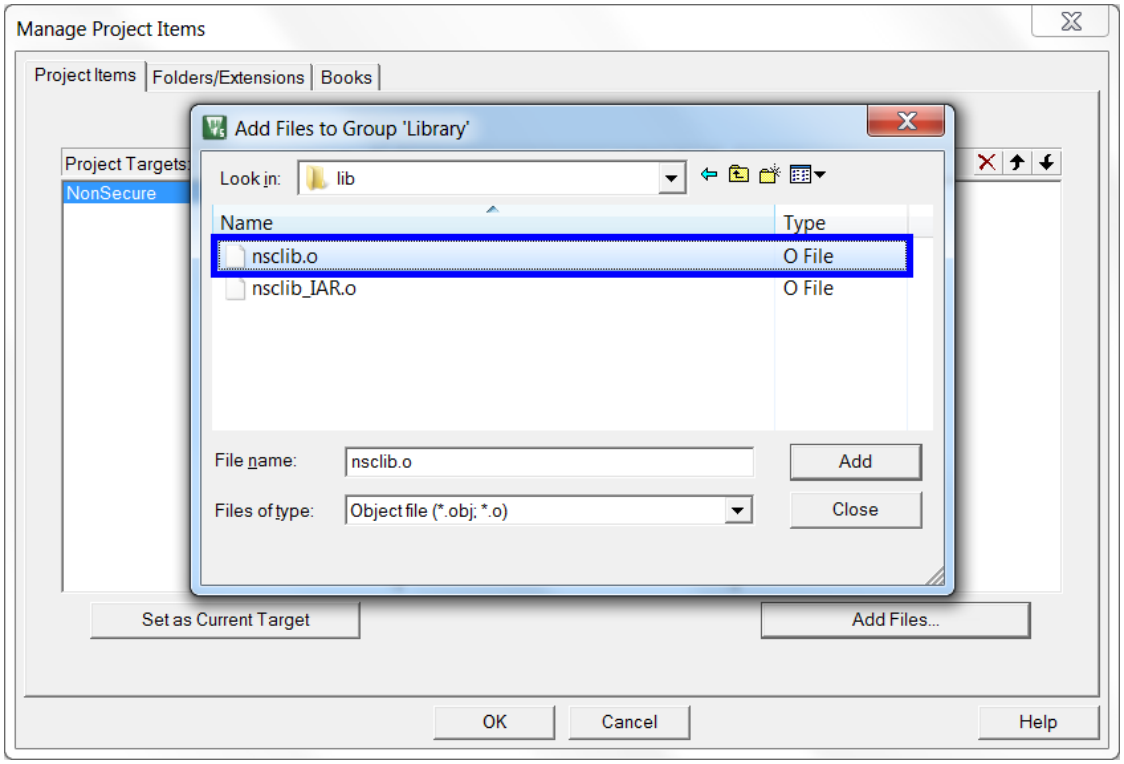

Figure 3-5 Add Non-secure Callable Library to Project in Keil® MDK

- In development

When the second developer is in development, it can develop the peripheral that is configured

to non-secure by the first developer directly, and can only access the peripheral that secure configured by calling non-secure callable function when user wants to use or control it. In NuMicro® M2351 series, for instance, the clock controller is fixed to secure only peripheral, if non-secure callable APIs does not contain the function of clock source select provided by the first developer, the clock of the peripheral that the second developer wants to develop can only operate in the setup already configured by the first developer in advance. Furthermore, the related precautions of downloading code please refer to section 3.3.

- After development

To protect code in non-secure region, the second developer can open ARLOCK to avoid code from being tampered or destroyed in secure and non-secure regions after completing development stage. Section 3.4 will introduce the concept of ARLOCK and how to set up environment in Keil® MDK and ICP Programming Tool.

3.2 XOM – Execute-Only Memory

XOM is a region only can execute code, and the data in this region cannot be accessed by any interface or tool. Hence, no matter the first or the second developers, they can put the source code not released to the customer in this region. In the case of not being read can achieve the effect of protect code effectively. For example, the first developers can put library which they provide to the second developers in XOM, and the second developers can call and execute the instructions but cannot know what code is in the library.

The following will introduce the XOM setup method in Keil® MDK and ICP Programming Tool. For details about the XOM configuration method and the precautions on use, please refer to XOM Configure Manual document.

3.2.1 Keil® MDK

Open a project in the Keil® MDK environment.

1. Click [Options for Target – Utilities] and click [Settings] button, and Flash Download page will appear. Then click [Setting] button under Chip Setting to open the XOM Setting page, as shown in Figure 3-6.
2. Select [XOM0/1/2/3], specify Start Address and Page Counts of XOM and then click [OK].
3. Also select [Debug Mode] for program debugging.

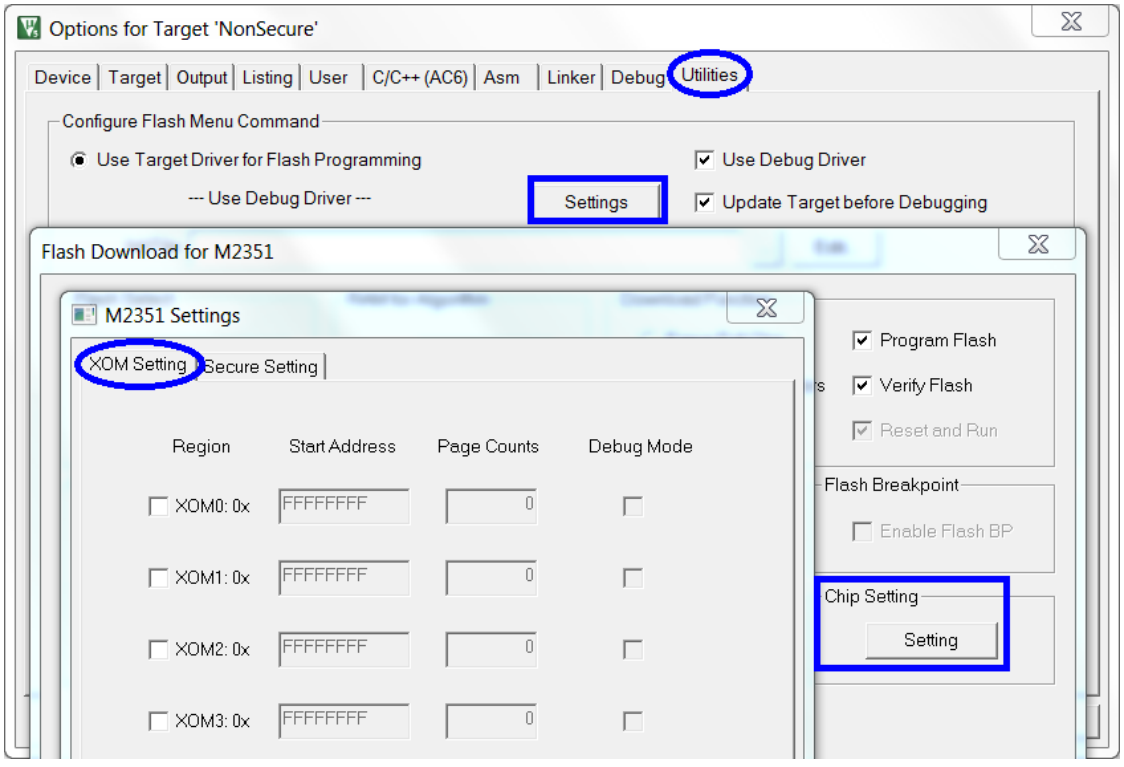

Figure 3-6 Configure XOM in Keil® MDK

3.2.2 ICP Programming Tool

After selecting a target chip correctly, connect the ICP Programming Tool with Nu-Link. When connected:

1. Click [Setting] button under Chip Settings to open the XOM Setting page, as shown in Figure 3-7.
2. Select [XOM0/1/2/3], specify Start Address and Page Counts of XOM and then click [OK].
3. Also select [Debug Mode] for program debugging.
4. Select [Chip Setting] under Programming and then click [Start] to start programming.

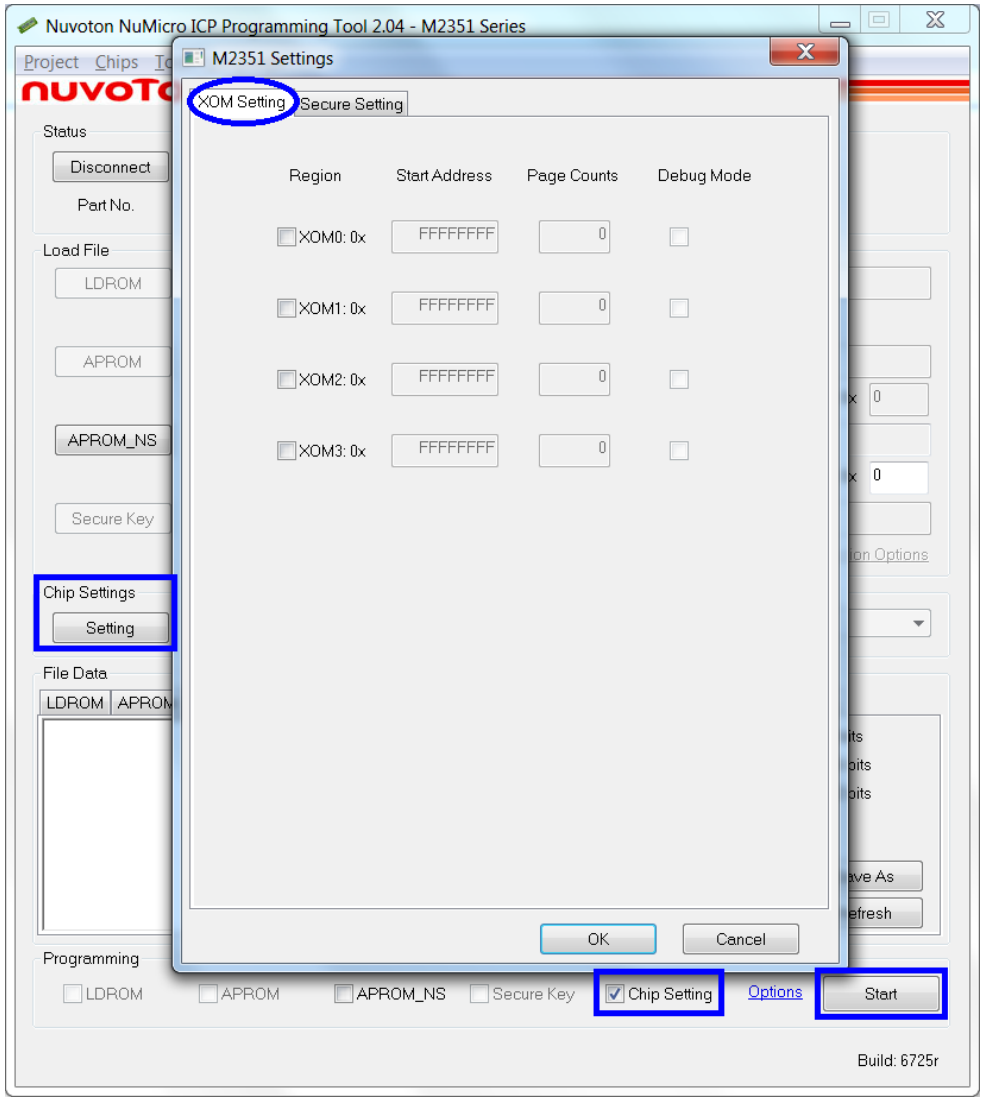

Figure 3-7 Configure XOM in ICP Programming Tool

3.3 Download Code

Nuvoton provides the debugging tool to adjust and download code on MCU. Because the first developer protects secure region code by using SCRLOCK, the second developer will have the restrictions on setup and use when debugging. The following will introduce the environment setup in Keil® MDK and ICP Programming Tool respectively.

3.3.1 Keil® MDK

The second developers must confirm that all setup flows before development are set correctly according to section 3.1, and then click [Download] under [Flash] to start downloading code, or click [Start/Stop Debug Session] under [Debug] to debug code. It should be noted that, when using [Memory Windows] under [View] in the debugging interface, the second developer can only read and write the peripheral that non-secure configured. In addition, the second developer can only do step over in debug mode, if user do step in, code will step out immediately. And if code run to secure region and did not return to non-secure region, ICE will be disconnected. The following describes the environment in Keil® MDK with the sample code in chapter 4.

In the NuMicro® M2351 series, the secure peripheral is configured in the address of 0x4XXX_XXXX, and the non-secure peripheral is configured in the address of 0x5XXX_XXXX. The second developer cannot access all control registers of secure region directly, hence user can only read the value of 0xFFFF_FFFF in all address of 0x4XXX_XXXX. As shown in Figure 3-8, 0x4000_2000 and 0x4000_4000 are the base address of clock controller and general purpose I/O (GPIO) in secure region respectively.

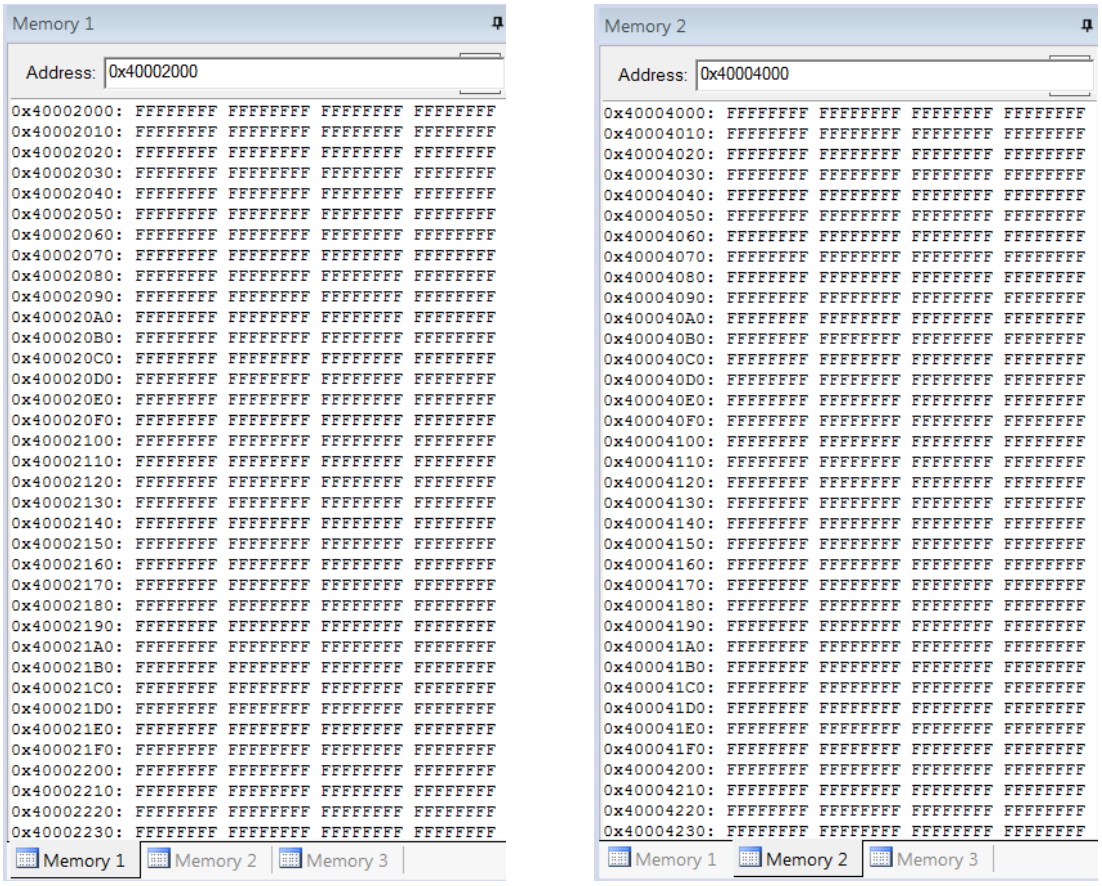

Figure 3-8 Read 0xFFFF_FFFF in Secure Region Control Registers

The base address of GPIO in non-secure region is 0x5000_4000 as shown in Figure 3-9, the control registers of PA is in the green box and the control registers of PC is in the red box. In the following example, the first developer configures PC as non-secure peripheral and PA as secure peripheral, hence the second developer can read the programmed value of PC in the red box directly and can also specify value to control the register directly, but can only read 0x0000_0000 and cannot write any value to the control registers in the green box after entering debugging interface.

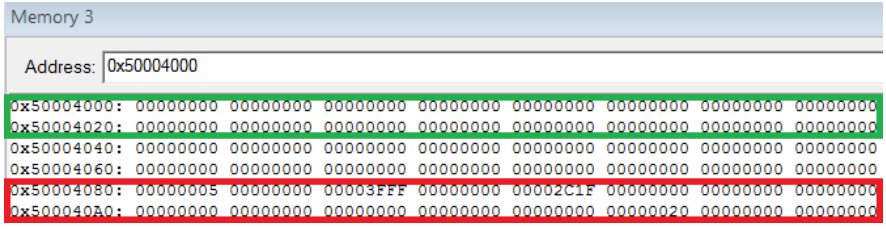

Figure 3-9 GPIO Non-secure Region Control Registers

Give another example, 0x5007_0000 and 0x5007_1000 are the base address of UART0 and UART1 respectively in non-secure region as shown in Figure 3-10. In the following example, the first developer configures UART0 as non-secure peripheral and UART1 as secure peripheral, hence, the second developer can read and write the control registers of UART0 directly, but can only read 0xFFFF_FFFF in the control registers of UART1 where cannot write any value after entering debugging interface.

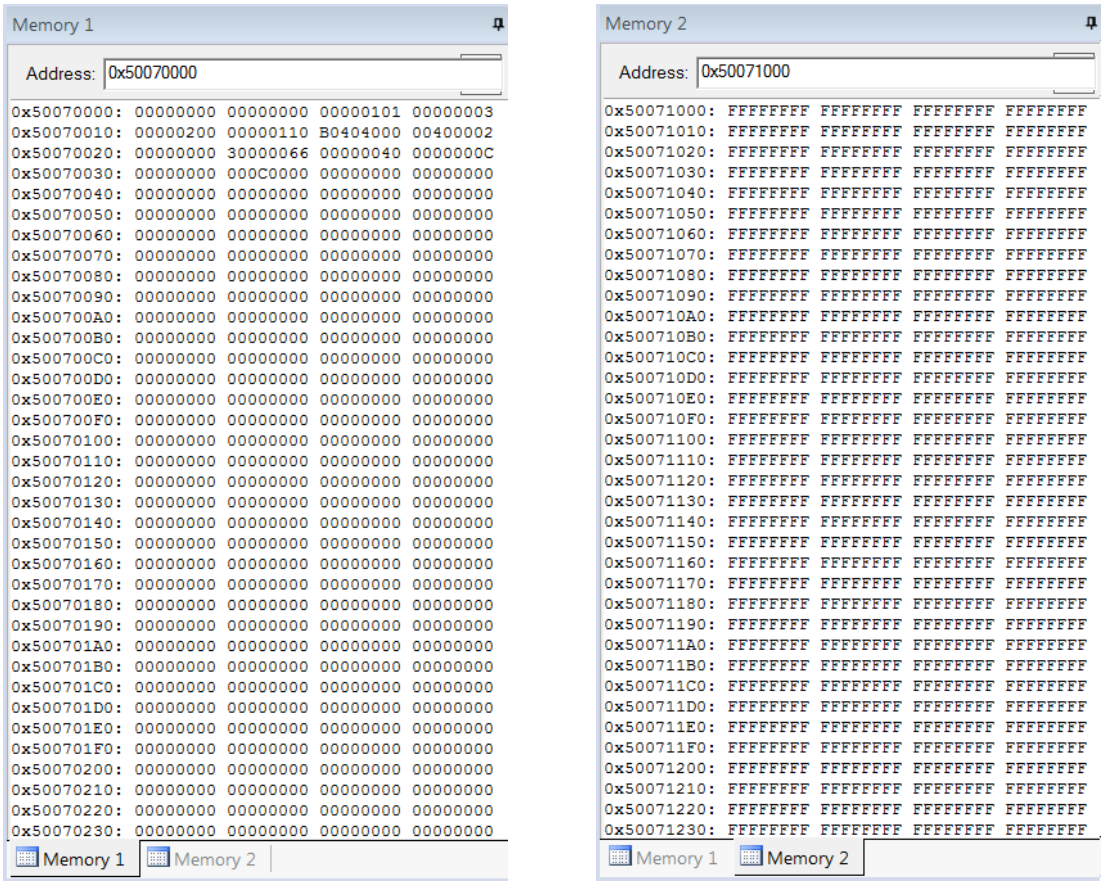

Figure 3-10 UART0/1 Non-secure Region Control Registers

3.3.2 ICP Programming Tool

After selecting a target chip correctly, connect the ICP Programming Tool with Nu-Link. When connected:

1. Add a non-secure file to [APROM_NS].
2. Select [APROM_NS] under Programming and click [Start] to start programming, then user can download code to non-secure region.

In addition, the second developer can read the initial settings on this chip programmed by the first developer in the red box shown in Figure 3-11. For example, the size of non-secure SRAM is 64KB, the size of non-secure APROM is 256KB and the base address of non-secure region is 0x1004_0000; therefore, the range can be developed is from 0x1004_0000 to 0x1007_FFFF. Moreover, because secure region is locked by the first developer, the second developer cannot program the secure region, such as LDROM and APROM. And user cannot read the related information about secure region in On-board Flash.

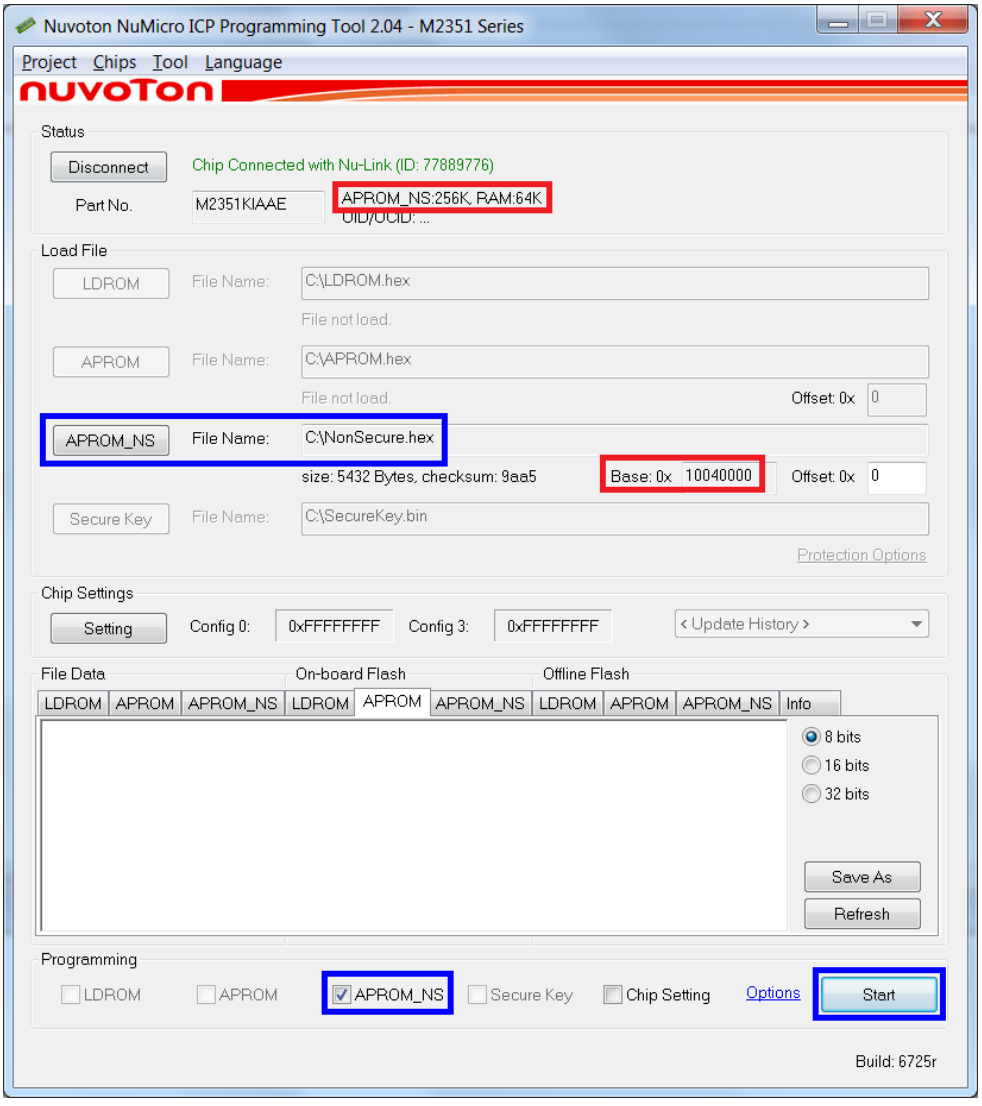

Figure 3-11 Download Non-secure Region Code in ICP Programming Tool

3.4 ARLOCK – All Region Lock

All region lock implies that it will lock code in all regions. When MCU detects there is ICE connected, If ARLOCK address in 0x0021_0804 is not read as 0x5A, the MCU will start read and write protection mechanism on secure region and non-secure region. This function can be enabled after the second developer completes the product development stage. This prevents code in product from being invaded to get or tampered in any way by people with bad intention.

In the Keil® MDK environment and ICP Programming Tool, the Secure Setting page contains options for setting NSCBA, SCRLOCK and ARLOCK. Among them, NSCBA and SCRLOCK are limited to be used by the first developer and ARLOCK is available for the second developer for choose. The following introduces the operation method.

3.4.1 Keil® MDK

Open a project in the Keil® MDK environment.

1. Click [Options for Target – Utilities], and click [Settings] button to open Flash Download page. Then click [Setting] button under Chip Setting to open the Secure Setting page, as shown in Figure 3-12.
2. Select [All Region Lock] and click [OK] then secure and non-secure regions can be locked.

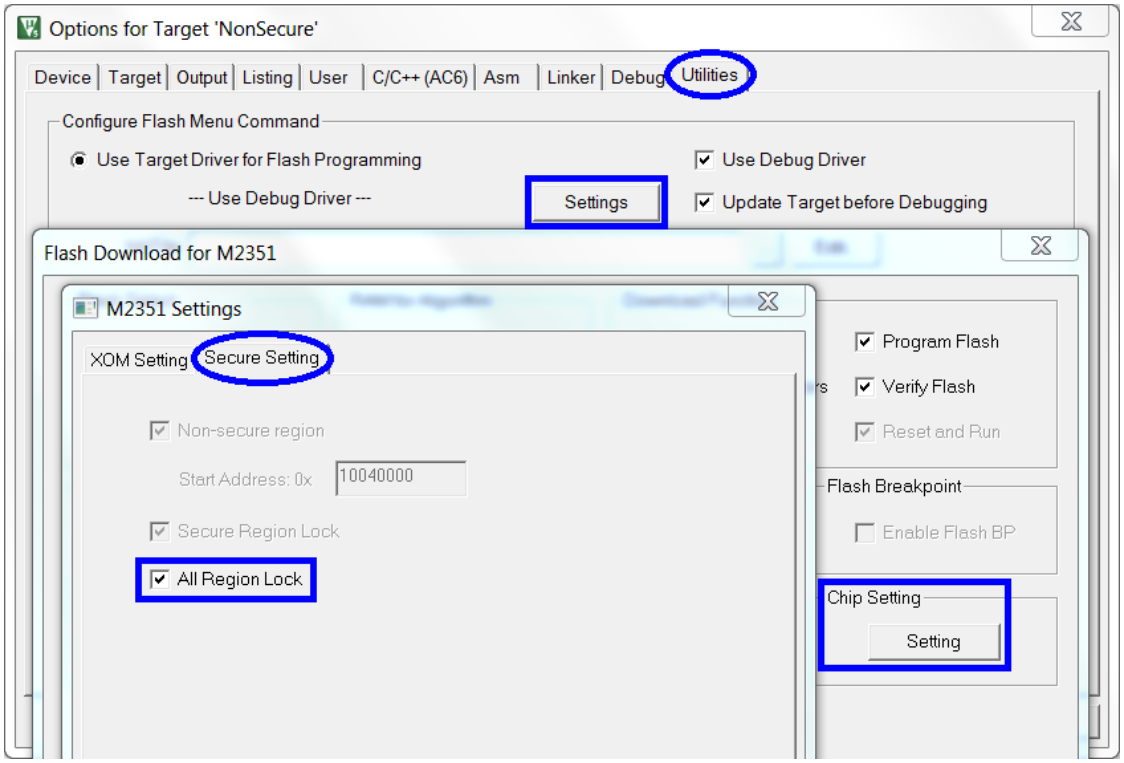

Figure 3-12 Setting Method of ARLOCK in Keil® MDK

3.4.2 ICP Programming Tool

After selecting a target chip correctly, connect the ICP Programming Tool with Nu-Link. When connected:

1. Click [Setting] button under Chip Settings to open the Secure Setting page, as shown in Figure 3-13.
2. Select [All Region Lock] and then click [OK].
3. Select [Chip Setting] under Programming and click [Start] to start programming, then secure and non-secure regions can be locked.

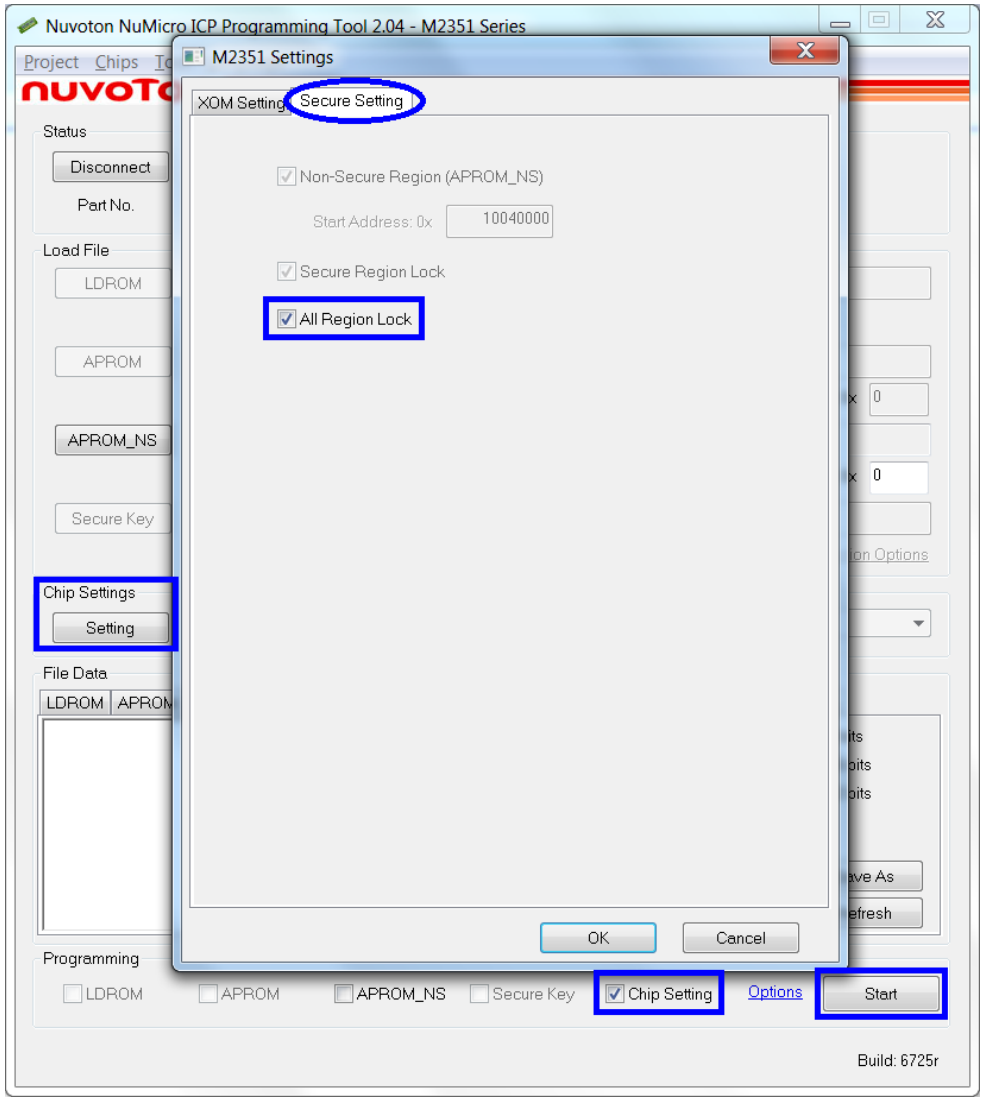

Figure 3-13 Setting Method of ARLOCK in ICP Programming Tool

4 Sample Code

4.1 Operation Procedure

The folder of sample CSSD_LED contains two packs of program: Secure and NonSecure. Firstly, download Secure code and open SCRLOCK according to the introduction in section 2.2, this can be used to simulate MCU provided by the first developer. After that, download NonSecure code to simulate the behavior that the second developer conducts collaborative development.

4.2 Sample Code Description

4.2.1 CSSD_LED – Secure

- System frequency is 64 MHz; UART0_NS serial ports TX/RX is PB.13/PB.12; Baud rate is 115200 bps.
- Non-secure APROM/SRAM base address is set as 0x1004_0000/0x3000_8000.
- The secure function to be provided to the second developer is placed in non-secure callable.

```
#include <arm_cmse.h>
#include <stdio.h>
#include "NuMicro.h"                        /* Device header */
#include "partition_M2351.h"

#define NEXT_BOOT_BASE  0x10040000
#define JUMP_HERE       0xe7fee7ff      /* Instruction Code of "B ." */

/* typedef for NonSecure callback functions */
typedef __NONSECURE_CALL int32_t (*NonSecure_funcptr)(uint32_t);

void SYS_Init(void)
{
    /* Enable PLL */
    CLK->PLLCTL = CLK_PLLCTL_128MHz_HIRC;

    /* Waiting for PLL stable */
    while((CLK->STATUS & CLK_STATUS_PLLSTB_Msk) == 0);

    /* Set HCLK divider to 2 */
    CLK->CLKDIV0 = (CLK->CLKDIV0 & (~CLK_CLKDIV0_HCLKDIV_Msk)) | 1;
```

```
    ...

    /* Set multi-function pins for UART0 RXD and TXD */
    SYS->GPB_MFPH = (SYS->GPB_MFPH & (~(UART0_RXD_PB12_Msk | UART0_TXD_PB13_Msk))) |
                    UART0_RXD_PB12 | UART0_TXD_PB13;
}

void DEBUG_PORT_Init(void)
{
    /* Configure UART and set UART Baudrate */
    DEBUG_PORT->BAUD = UART_BAUD_MODE2 | UART_BAUD_MODE2_DIVIDER(__HIRC, 115200);
    DEBUG_PORT->LINE = UART_WORD_LEN_8 | UART_PARITY_NONE | UART_STOP_BIT_1;
}

void Nonsecure_Init(void)
{
    NonSecure_funcptr fp;

    /* SCB_NS.VTOR points to the Non-secure vector table base address. */
    SCB_NS->VTOR = NEXT_BOOT_BASE;

    /* 1st Entry in the vector table is the Non-secure Main Stack Pointer. */
    __TZ_set_MSP_NS(*((uint32_t *)SCB_NS->VTOR)); /* Set up MSP in Non-secure code */

    /* 2nd entry contains the address of the Reset_Handler (CMSIS-CORE) function */
    fp = ((NonSecure_funcptr)(*(((uint32_t *)SCB_NS->VTOR) + 1)));

    /* Clear the LSB of the function address to indicate the function-call
       will cause a state switch from Secure to Non-secure */
    fp = cmse_nsfptr_create(fp);

    /* Check if the Reset_Handler address is in Non-secure space */
    if(cmse_is_nsfptr(fp) && (((uint32_t)fp & 0xf0000000) == 0x10000000))
    {
        printf("Execute non-secure code ...\n");
        fp(0); /* Non-secure function call */
    }
    else
    {
        /* Something went wrong */
```

```
        printf("No code in non-secure region!\n");
        printf("CPU will halted at non-secure state\n");

        /* Set nonsecure MSP in nonsecure region */
        __TZ_set_MSP_NS(NON_SECURE_SRAM_BASE + 512);

        /* Try to halted in non-secure state (SRAM) */
        M32(NON_SECURE_SRAM_BASE) = JUMP_HERE;
        fp = (NonSecure_funcptr)(NON_SECURE_SRAM_BASE + 1);
        fp(0);

        while(1);
    }
}

__NONSECURE_ENTRY
int32_t Secure_PA11_LED_On(uint32_t num)
{
    printf("Secure PA11 LED On call by secure\n");
    PA11 = 0;
    return 0;
}

__NONSECURE_ENTRY
int32_t Secure_PA11_LED_Off(uint32_t num)
{
    printf("Secure PA11 LED Off call by secure\n");
    PA11 = 1;
    return 1;
}

...

__NONSECURE_ENTRY
uint32_t GetSystemCoreClock(void)
{
    printf("System core clock = %d.\n", SystemCoreClock);
    return SystemCoreClock;
}

int32_t LED_On(void)
```

```
{
    printf("Secure/Non-secure LED On call by Secure\n");
    PA10 = 0;
    PC1_NS = 0;
    return 0;
}

int32_t LED_Off(void)
{
    printf("Secure/Non-secure LED Off call by Secure\n");
    PA10 = 1;
    PC1_NS = 1;
    return 1;
}

void SysTick_Handler(void)
{
    static uint32_t ticks;

    switch(ticks++)
    {
        case   0:
            LED_On(0u);
            break;

        ...

        case 600:
            ticks = 0;
            break;

        default:
            if(ticks > 600)
            {
                ticks = 0;
            }
    }
}

int main(void)
{
```

```
    SYS_UnlockReg();

    SYS_Init();

    /* UART is configured as non-secure for debug in both secure and non-secure region */
    DEBUG_PORT_Init();

    printf("Secure code is running ...\n");

    /* Init GPIO Port A for secure LED control */
    GPIO_SetMode(PA, BIT13 | BIT12 | BIT11 | BIT10, GPIO_MODE_OUTPUT);

    /* Init GPIO Port C for non-secure LED control */
    GPIO_SetMode(PC_NS, BIT1, GPIO_MODE_OUTPUT);

    /* Generate Systick interrupt each 10 ms */
    SysTick_Config(SystemCoreClock / 100);

    Nonsecure_Init();

    do
    {
        __WFI();
    }
    while(1);
}
```

- Defined in secure scatter file
  - The range of secure APROM is 0x0000_0000~0x0003_FFFF.
  - The range of secure SRAM is 0x2000_0000~0x2000_7FFF.
  - The range of placing non-secure callable APIs is 0x0003_F000~0x0003_FFFF.

```
LR_ROM 0x0
{
    EXE_ROM +0 0x40000
    {
        *.o(RESET, +First)
        *(+RO)
    }

    EXE_RAM 0x20000000 0x8000
```

```
    {
        *(+RW, +ZI)
    }
}

LR_NSC 0x3F000
{
    NSC +0
    {
         *(Veneer$$CMSE)
    }
}
```

4.2.2 CSSD_LED – NonSecure

- In the NonSecure folder, cssd_lib.h contains non-secure callable APIs provided by the first developer to be used by the second developer.

```
extern int32_t Secure_PA11_LED_On(uint32_t num);
extern int32_t Secure_PA11_LED_Off(uint32_t num);
extern int32_t Secure_PA12_LED_On(uint32_t num);
extern int32_t Secure_PA12_LED_Off(uint32_t num);
extern int32_t Secure_PA13_LED_On(uint32_t num);
extern int32_t Secure_PA13_LED_Off(uint32_t num);
extern uint32_t GetSystemCoreClock(void);
```

- LED_ON and LED_OFF are the non-secure functions developed by the second developer.
- Take hybrid display of secure and non-secure LED lights as a collaborative development example, the location of second developer handle in SysTick_Handler can be configured to non-secure functions or non-secure callable functions by the second developer.

```
#include <arm_cmse.h>
#include "NuMicro.h"                    /* Device header */
#include "cssd_lib.h"

void LED_On(uint32_t us)
{
    printf("NS LED On call by NS\n");
    PC0_NS = 0;
}
```

```
void LED_Off(uint32_t us)
{
    printf("NS LED Off call by NS\n");
    PC0_NS = 1;
}

void SysTick_Handler(void)
{
    static uint32_t ticks;

    switch(ticks++)
    {
        case   0:
            // second developer handle
            LED_On(7u);
            Secure_PA11_LED_On(0u);
            break;

        ...

        case 600:
            ticks = 0;
            break;

        default:
            if(ticks > 600)
            {
                ticks = 0;
            }
    }
}

int main(void)
{
    printf("\nNonsecure code is running ...\n");

    /* Init GPIO Port C for non-secure LED control */
    GPIO_SetMode(PC_NS, BIT0, GPIO_MODE_OUTPUT);

    /* Call secure API to get system core clock */
```

```
    SystemCoreClock = GetSystemCoreClock();

    /* Generate Systick interrupt each 10 ms */
    SysTick_Config(SystemCoreClock / 100);

    /* Waiting for secure/non-secure Systick interrupt */
    while(1);
}
```

- Defined in non-secure scatter file
  - The range of non-secure APROM is 0x1004_0000~0x1007_FFFF.
  - The range of non-secure SRAM is 0x3000_8000~0x3001_7FFF.

```
LR_ROM 0x10040000 0x10000
{
    EXE_ROM +0
    {
        *.o(RESET, +First)
        *(+RO)
    }

    EXE_RAM 0x30008000 0x10000
    {
        *(+RW, +ZI)
    }
}
```

5 Conclusion

As described above, the first developer must engage in development under the project environment in secure mode. After the development is completed, the first developer must provide the second developer with non-secure APROM and SRAM base address, the object file generated by non-secure callable and non-secure callable APIs header file.

The second developer must engage in development under the project environment in non-secure mode. Note that in the development, the peripheral configured to non-secure by the first developer can be accessed directly; otherwise, only the peripheral configured to secure can be accessed by calling the non-secure callable function provided by the first developer.

Revision History

Date	Revision	Description
2018.09.06	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.