GigaDevice Semiconductor Inc.

GD32W51x 系列 TrustZone 开发指南

应用笔记

AN103

目录

图索引

表索引

1. 前言

本文介绍在 GD32W51x 系列上开发 TrustZone 程序。TrustZone 安全属性是 ARMv8-M 主扩展内容，提供硬件级安全分离和保护，示例代码也分为安全和非安全两部分，将分别介绍 Keil 和 IAR 两种环境下的开发流程和要点。

本文使用 GD32W515P-EVAL 开发板，主控芯片型号为 GD32W515PIQ6，采用带有 TrustZone 功能的 Cortex-M33 内核，主频高达 180MHz。TrustZone 安全功能由 EFUSE_TZCTL 寄存器的 TZEN 位或者选项字节中的 TZEN 位激活。示例代码在‘GD32W515P_EVAL_Demo_Suites / 23_Trustzone’中提供，由 GD32MCU.COM 软件包提供。

适用产品：GD32W51x 系列

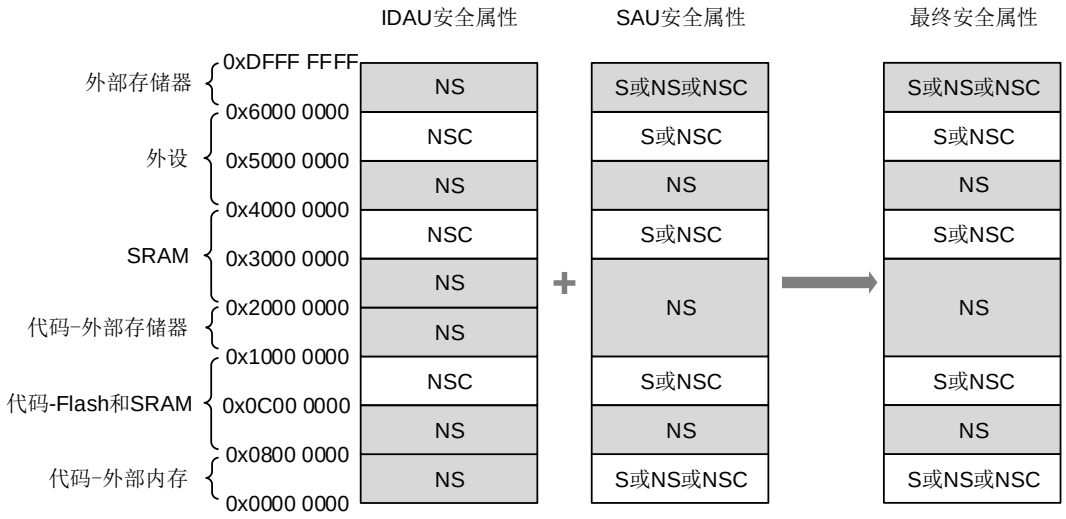
2. TrustZone 简介

带有 TrustZone 功能的 M33 内核具有安全和非安全两种状态，两种状态分别使用一组独立的内核寄存器和 Systick。存储区域分为安全（S）、非安全可调用（NSC）、非安全（NS）区域，安全区域可访问非安全区域，非安全区域只可访问非安全可调用区域（安全代码中），访问时提供硬件级安全保护。当 TrustZone 使能时，SAU（安全属性单元）和 IDAU（实施定义属性单元）用来共同设定内存地址的安全属性。最终安全等级为 SAU 和 IDAU 中定义的较高安全等级的属性（S>NSC>NS），见*图 2-1. 内存映射安全属性与 SAU 配置区域的示例*。

IDAU 提供一个硬件的非安全（NS）和非安全可调用（NSC）安全属性划分，软件无法更改，内存映射分区参考 GD32W51x_用户手册的 1.4 存储器映射章节。

SAU 可软件配置最多 8 个安全属性区域，可以将这些区域的安全（S）、非安全（NS）或非安全可调用（NSC）。

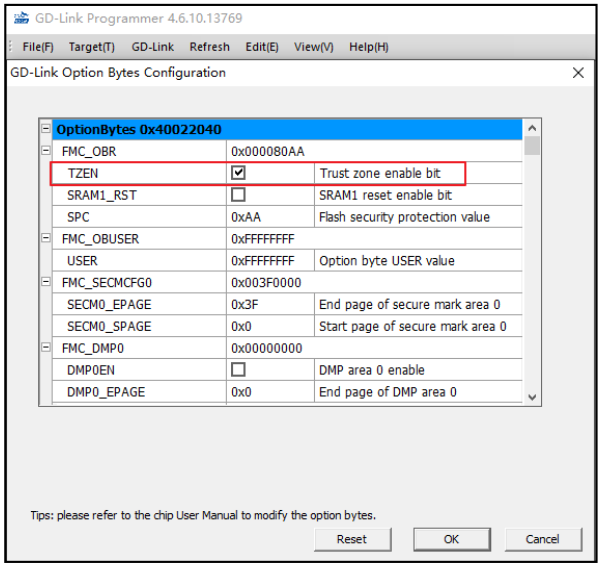
图 2-1. 内存映射安全属性与 SAU 配置区域的示例



GD32W51x 系列使用 TrustZone 保护控制器组（TZPCU）管理外设、SRAM、闪存的安全属性，并提供非法访问控制，配合内核实现 TrustZone 全功能。

在使用这些功能之前，请确保 EFUSE_TZCTL 寄存器的 TZEN 位或者选项字节中的 TZEN 位已使能。可使用 GD-Link Programmer 使能选项字节的 TZEN 位，见*图 2-2. 通过 GD-Link Programmer 使能 TrustZone*，GD-Link Programmer 软件可通过 GD32MCU.COM 获取。

图 2-2. 通过 GD-Link Programmer 使能 TrustZone



3. 软件开发

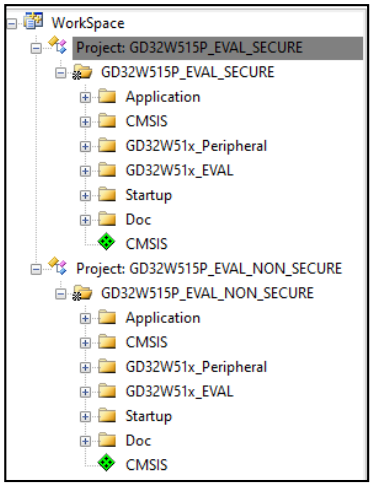
本文使用 MDK-ARM v5.28.0.0 和 IAR v8.50.9 两种开发环境。请确保已正确安装对应软件和器件支持包，开发环境软件可通过 **KEIL** 和 **IAR** 官网下载，器件支持包（GD32W51x_AddOn）可通过 GD32MCU.COM 下载。

示例工程中包含安全和非安全两部分，两部分分别使用不同的 Flash 和 SRAM。安全代码使用 FMC 中的前 256KB 空间（起始地址为 0x0C000000，大小 0x40000），内存使用 SRAM0 块（起始地址为 0x30000000，大小为 0x10000）。非安全代码使用 FMC 中的后 1792KB 空间（起始地址为 0x08040000，大小 0x1C0000），内存使用 SRAM1 块（起始地址为 0x20010000，大小为 0x10000）。

3.1. Keil 下开发 TrustZone

使用多工程方式可同时开发两个工程，文件结构见*图 3-1. Keil 工程文件结构*，工程路径为：GD32W51x_Demo_Suites\GD32W515P_EVAL_Demo_Suites\Projects\23_Trustzone\MDK-ARM\GD32W515P_EVAL_TRUSTZONE.uvmpw

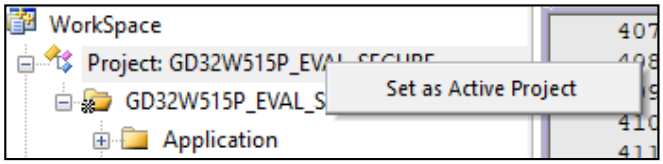
图 3-1. Keil 工程文件结构



3.1.1. 安全工程

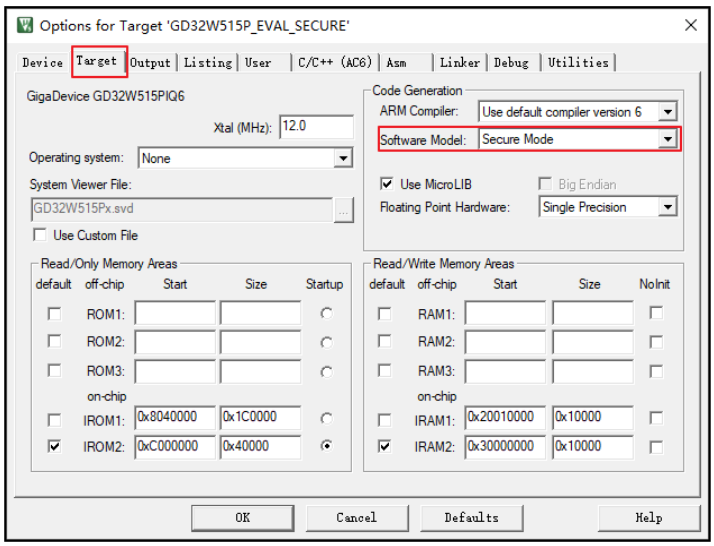
1、选择安全工程为当前工程，见*图 3-2. 选择安全工程*。

图 3-2. 选择安全工程



2、在 Project / Options for Target / Target 选项卡中，选择 Code Generation / Software Model 为 Secure Mode，见*图 3-3. 选择安全模式*。

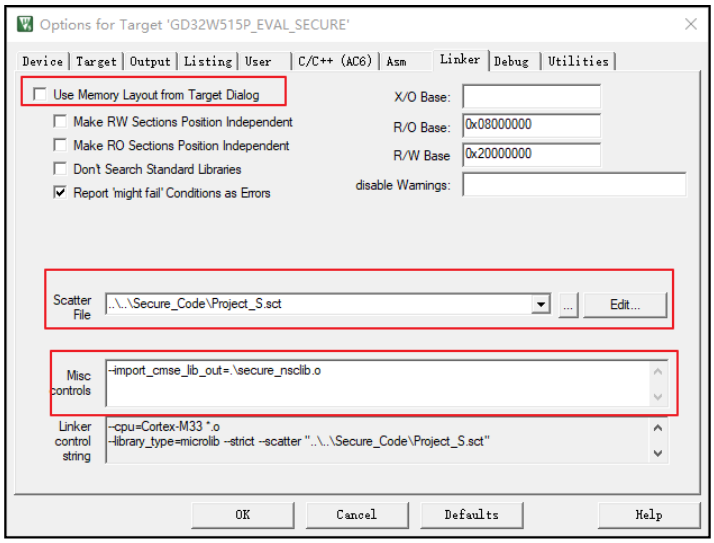
图 3-3. 选择安全模式



3、安全代码和非安全代码使用不同的 Flash 和 SRAM，使用分散加载文件为安全代码分配正确的 Flash 和 SRAM。在 Project / Options for Target / Linker 选项卡中，选择不使用 Target 界面分配地址，选择分散加载文件，并为 NSC 函数设置输出库。见*图 3-4. 选择分散加载文件和 NSC 输出库*。

输出命令为 --import_cmse_lib_out=.\secure_nsclib.o，编译器将在编译时将代码中__attribute((cmse_nonsecure_entry))标识代码编译到.\secure_nsclib.o 文件中，供非安全代码访问。

图 3-4. 选择分散加载文件和 NSC 输出库



4、安全代码使用 Flash 地址为 0x0C000000，大小为 0x40000，SRAM 地址为 0x30000000，大小为 0x10000。其中 NSC 函数分配到 0x0C03E000 地址，大小为 0x00002000。分散加载文件 Project_S.sct 代码见*表 3-1. Project_S.sct 代码*。

表 3-1. Project_S.sct 代码

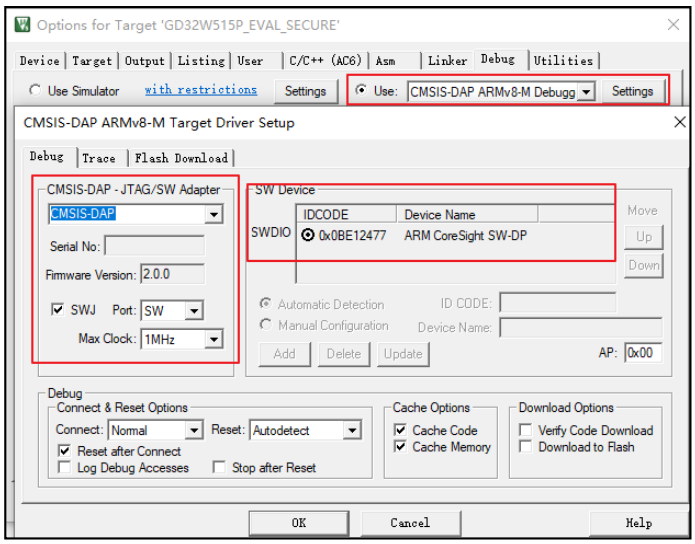
```
LR_IROM1  0x0C000000  0x00040000  {     ; load region size_region
  ER_IROM1  0x0C000000  0x0003E000  {  ; load address = execution address
```

```
    *.o (RESET,  +First)
    *(InRoot$$Sections)
    .ANY (+RO)
    .ANY (+XO)
  }
  RW_IRAM2  0x30000000 0x00010000  {  ; RW data
    .ANY (+RW +ZI)
  }
}

LR_IROM2  0x0C03E000  0x00002000  {
  ER_IROM2  0x0C03E000  0x00002000  {  ; load address = execution address
    *(Veneer$$CMSE)                    ; check w ith partition.h
  }
```

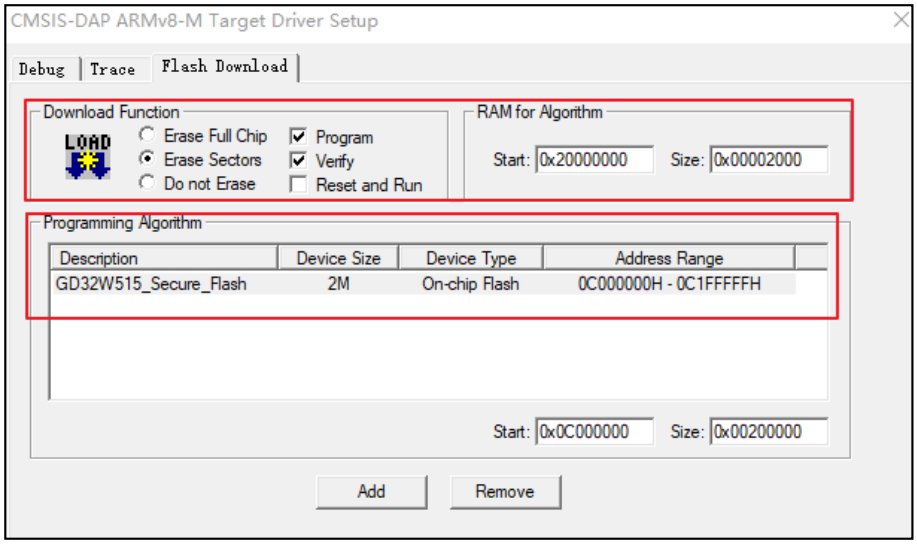
5、使用 GD-Link 进行调试和下载，将开发板的 GD-Link 接口与 PC 的 USB 口直接相连，在 Project / Options for Target / Debug 下选择 CMSIS-DAP ARMv8-M Debugger 选项，在 Settings 中识别能到调试器 ID，使用 SW 接口并设置最大时钟为 1MHz。见*图 3-5. 安全工程设置调试器*。

图 3-5. 安全工程设置调试器



6、在 Flash Download 选项卡下选择 Erase Sectors 并取消选中 Reset and Run，添加针对安全区域的下载算法 GD32W515_Secure_Flash，该下载算法由所安装的器件包提供。见*图 3-6. 安全工程设置下载算法和功能*。

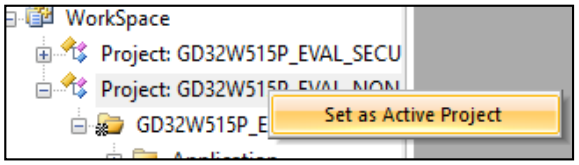
图 3-6. 安全工程设置下载算法和功能



3.1.2. 非安全工程

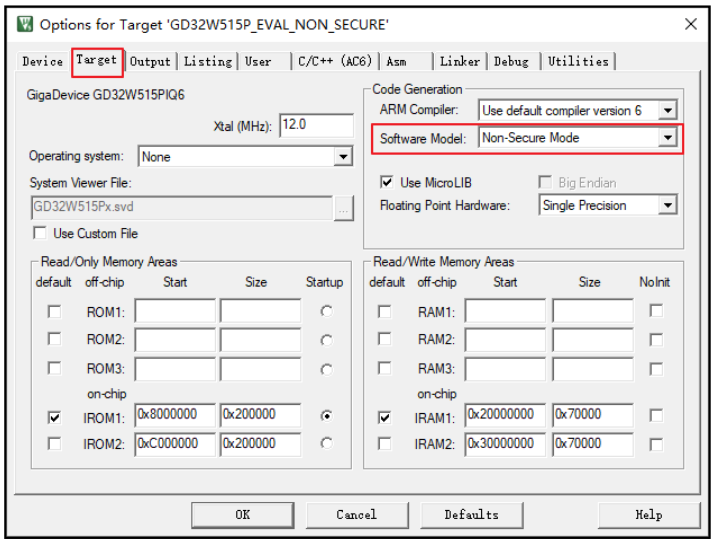
1、选择非安全工程为当前工程，见*图 3-7. 选择非安全工程*。

图 3-7. 选择非安全工程



2、在 Project / Options for Target / Target 选项卡中，选择 Code Generation / Software Model 为 Non-Secure Mode，见*图 3-8. 选择非安全模式*。

图 3-8. 选择非安全模式

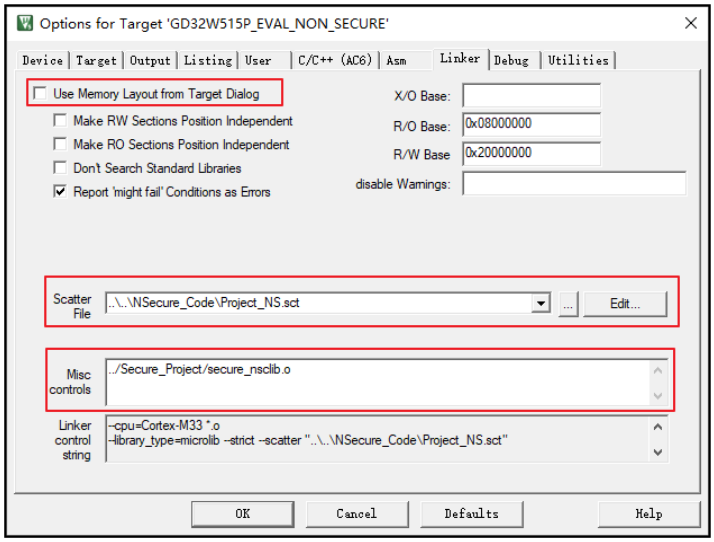


3、安全代码和非安全代码使用不同的 Flash 和 SRAM，使用分散加载文件为安全代码分配正确的 Flash 和 SRAM。在 Project / Options for Target / Linker 选项卡中，选择不使用 Target 界

面分配地址，选择分散加载文件，并使用安全工程导出的 NSC 函数库。见*图 3-9. 选择分散加载文件和导入 NSC 库*。

函数库导入命令../Secure_Project/secure_nsclib.o，可直接调用安全工程中__attribute((cmse_nonsecure_entry))标识代码，这部分代码包含在.\secure_nsclib.o 文件中。

图 3-9. 选择分散加载文件和导入 NSC 库



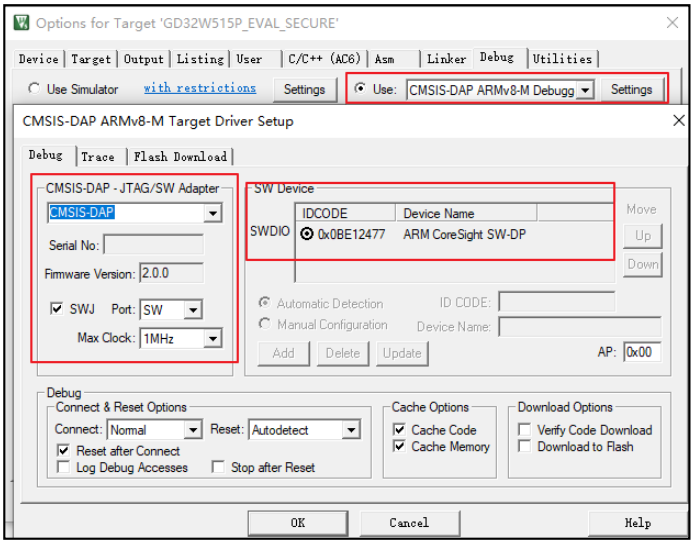
4、非安全代码使用 Flash 地址为 0x08040000，大小为 0x001C0000，SRAM 地址为 0x20010000，大小为 0x10000。分散加载文件 Project_NS.sct 代码见*表 3-2. Project_NS.sct 代码*。

表 3-2. Project_NS.sct 代码

```
LR_IROM1  0x08040000 0x001C0000   {     ; load region size_region
  ER_IROM1  0x08040000 0x001C0000   {   ; load address = execution address
   *.o (RESET,  +First)
   *(InRoot$$Sections)
   .ANY (+RO)
   .ANY (+XO)
  }
  RW_IRAM1  0x20010000 0x00010000   {   ; RW data
   .ANY (+RW +ZI)
  }
}
```

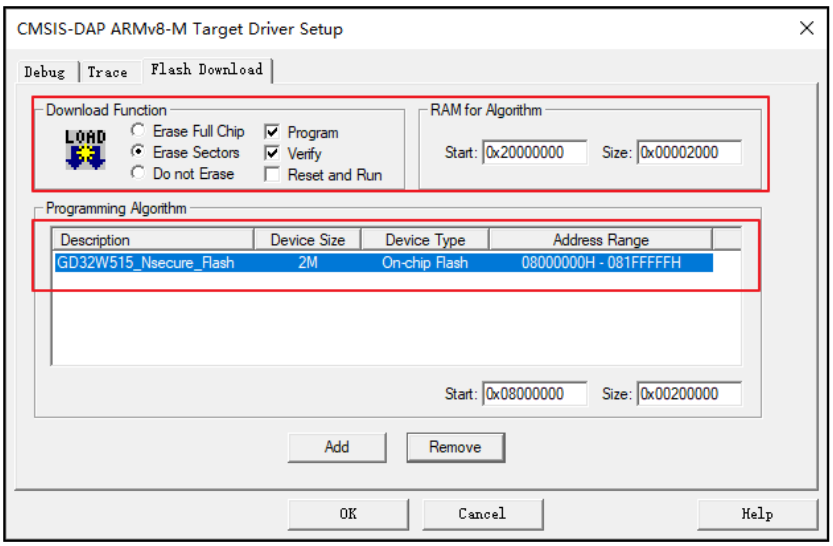
5、使用 GD-Link 进行调试和下载，将开发板的 GD-Link 接口与 PC 的 USB 口直接相连，在 Project / Options for Target / Debug 下选择 CMSIS-DAP ARMv8-M Debugger 选项，在 Settings 中识别能到调试器 ID，使用 SW 接口并设置最大时钟为 1MHz。见*图 3-10. 非安全工程设置调试器*。

图 3-10. 非安全工程设置调试器



6、在 Flash Download 选项卡下选择 Erase Sectors 并取消选中 Reset and Run，添加针对非安全区域的下载算法 GD32W515_Nsecure_Flash，该下载算法由所安装的器件包提供。见*图 3-11. 非安全工程设置下载算法和功能*。

图 3-11. 非安全工程设置下载算法和功能

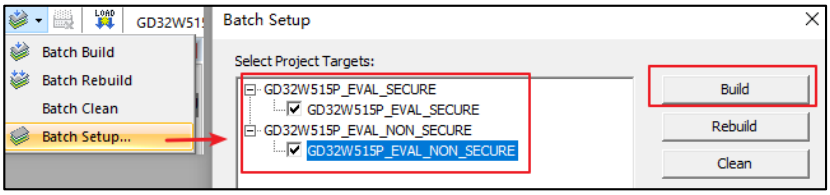


3.1.3. 编译工程

由于非安全工程需要使用安全工程生成 secure_nsclib.o 库，因此需要先编译安全工程生成该库，后编译非安全工程。可先设置安全工程为当前工程，编译安全工程，再设置非安全工程为当前工程，编译非安全工程。

也可设置 batch setup 选项依次编译两个工程，见*图 3-12. 工程编译*，勾选两个工程并点击 Build，编译顺序由工程目录顺序决定，可通过 Project / Manage / Multi-Project Workspace 选项修改。

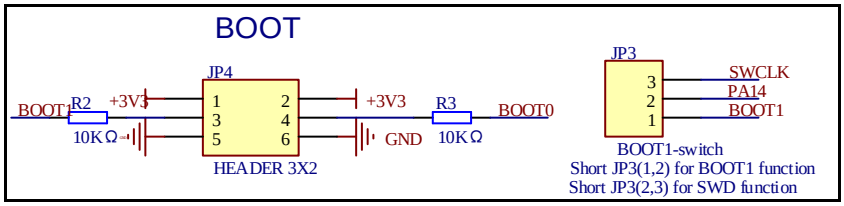
图 3-12. 工程编译



3.1.4. 下载工程

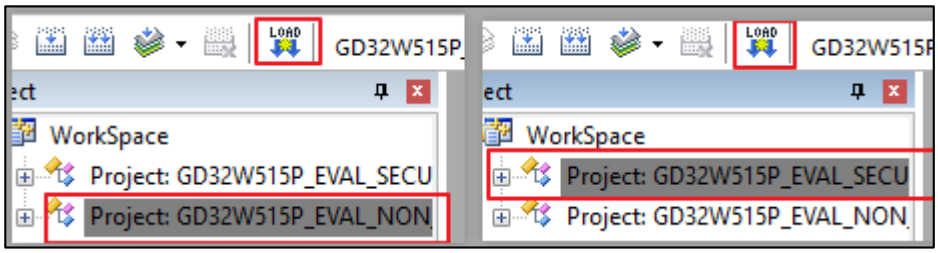
下载代码前需要先正确设置开发板，JP4 的 BOOT0/BOOT1 连接到 L，设置硬件 BOOT 选项从安全代码启动，JP3 连接到 SWD。见*图 3-13. BOOT 选项*。JP21 连接到 USART 用于串口打印。连接开发板 GD-Link 和 USART 到 PC 机上，并确保软件驱动已正确安装。

图 3-13. BOOT 选项



含有 TrustZone 功能的工程中，代码总是从安全代码启动，并跳转到非安全代码。因此建议先下载非安全代码，后下载安全代码。见*图 3-14. 工程下载*。

图 3-14. 工程下载

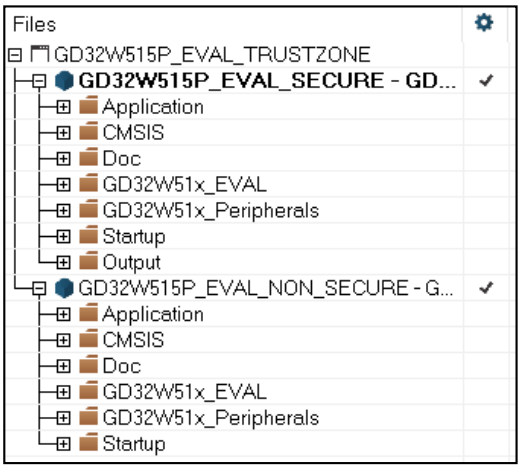


由于 Flash Download 选项中没有勾选 Reset and Run，因此需要手动按下 Reset 按键重启 MCU。重启后可观察到安全和非安全工程分别打印一条信息，并分别控制两个 LED 灯闪烁。

3.2. IAR 下开发 TrustZone

使用多工程方式可同时开发两个工程，文件结构见*图 3-15. IAR 工程文件结构*，工程路径为：GD32W51x_Demo_Suites\GD32W515P_EVAL_Demo_Suites\Projects\23_Trustzone\EWARM\GD32W515P_EVAL_TRUSTZONE.eww

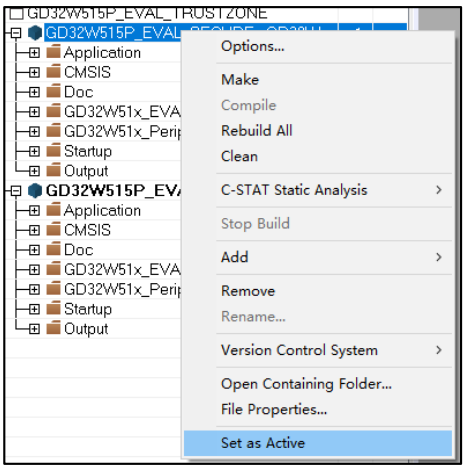
图 3-15. IAR 工程文件结构



3.2.1. 安全工程

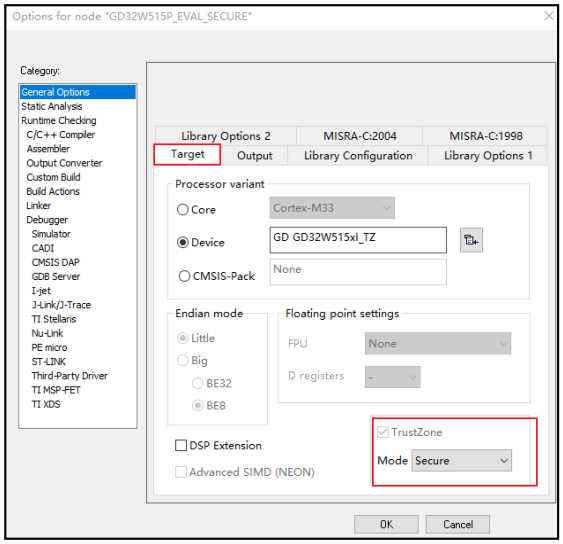
1、选择安全工程为当前工程，见*图 3-16. 选择安全工程*。

图 3-16. 选择安全工程



2、在 Project / Options / General Options / Target 选项卡中，选择 TrustZone / Mode 为 Secure。见*图 3-17.* **选择安全模式**。

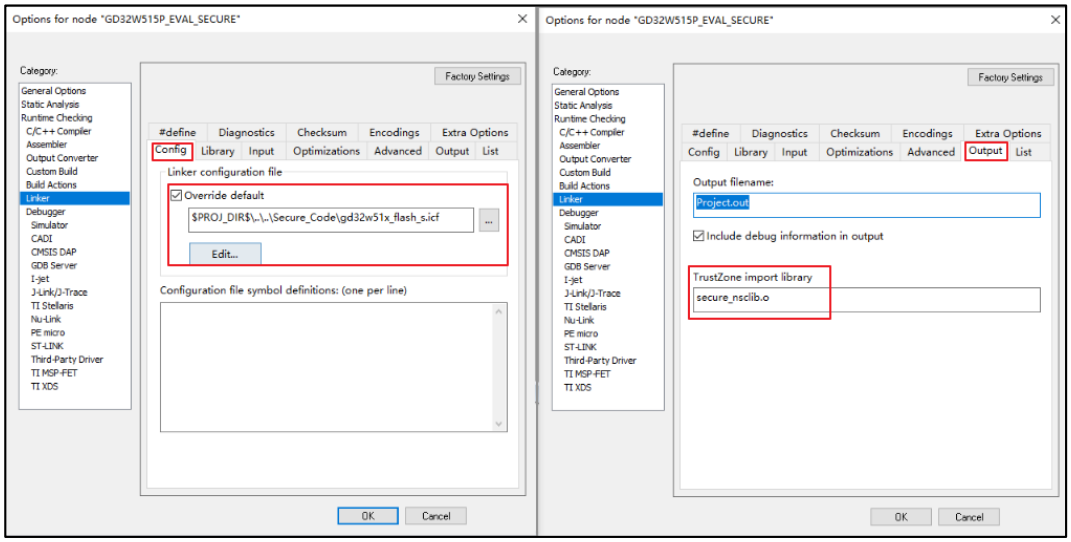
图 3-17. 选择安全模式



3、安全代码和非安全代码使用不同的 Flash 和 SRAM，使用分散加载文件为安全代码分配正确的 Flash 和 SRAM。在 Project/Options/Linker/Config 选项卡中，选择分散加载文件，并为 NSC 函数设置输出库。见*图 3-18. 选择分散加载文件和 NSC 输出库*。

输出库文件为 secure_nsclib.o，编译器将在编译时将代码中__attribute((cmse_nonsecure_entry))标识代码编译到 secure_nsclib.o 文件中，供非安全代码访问。

图 3-18. 选择分散加载文件和 NSC 输出库



4、安全代码使用 Flash 地址为 0x0C000000，大小为 0x40000，SRAM 地址为 0x30000000，大小为 0x10000。其中 NSC 函数分配到 0x0C03E000 地址，大小为 0x00002000。分散加载文件 gd32w51x_flash_s.icf 代码见*表 3-3. gd32w51x_flash_s.icf 代码*。

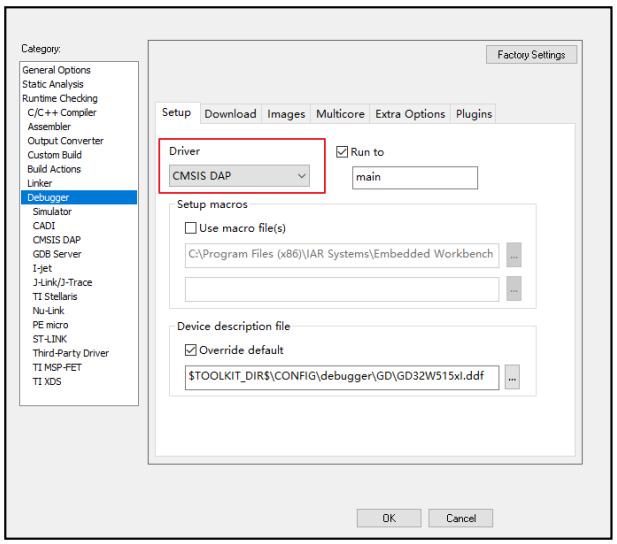
表 3-3. gd32w51x_flash_s.icf 代码

```
/*-Specials-*/
define symbol __ICFEDIT_intvec_start__  = 0x0C000000;
/*-Memory Regions-*/
```

```
define symbol __ICFEDIT_region_ROM_start__   = 0x0C000000;
define symbol __ICFEDIT_region_ROM_end__     = 0x0C03FFFF;
define symbol __ICFEDIT_region_RAM_start__   = 0x30000000;
define symbol __ICFEDIT_region_RAM_end__     = 0x3000FFFF;
......
define symbol __region_ROM_NSC_start__       = 0x0C03E000;
define symbol __region_ROM_NSC_end__         = 0x0C03FFFF;
......
define region ROM_region          = mem:[from __ICFEDIT_region_ROM_start__ to
__ICFEDIT_region_ROM_end__];
define region ROM_NSC_region      = mem:[from __region_ROM_NSC_start__ to
__region_ROM_NSC_end__];
define region RAM_region          = mem:[from __ICFEDIT_region_RAM_start__ to
__ICFEDIT_region_RAM_end__];
......
place in ROM_region        { readonly };
place in ROM_NSC_region    { section Veneer$$CMSE };
place in RAM_region        { readwrite, block HEAP, block CSTACK };
```

5、使用 GD-Link 进行调试和下载，在 Project/ Options/ Debugger 下选择 CMSIS-DAP 选项。见*图 3-19. 安全工程设置调试器*。

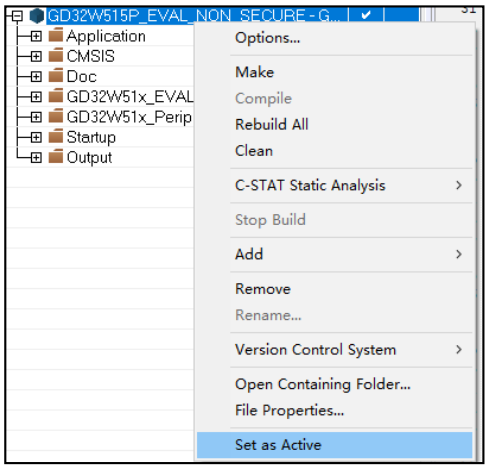
图 3-19. 安全工程设置调试器



3.2.2. 非安全工程

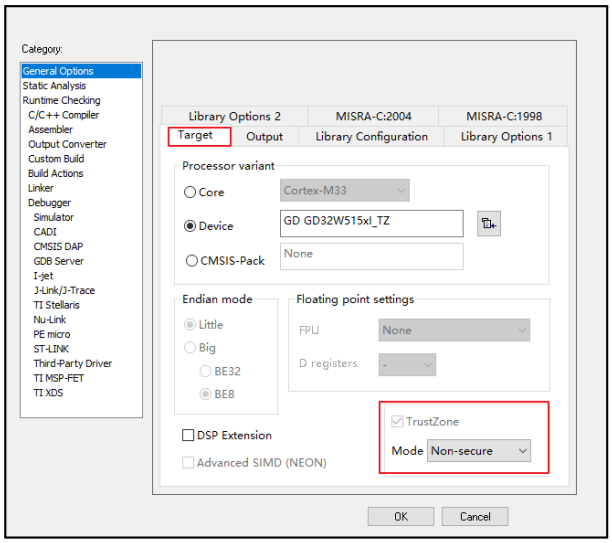
1、选择非安全工程为当前工程，见*图 3-20. 选择非安全工程*。

图 3-20. 选择非安全工程



2、在 Project / Options / General Options / Target 选项卡中，选择 TrustZone / Mode 为 Non-Secure。见*图 3-21. 选择非安全模式*。

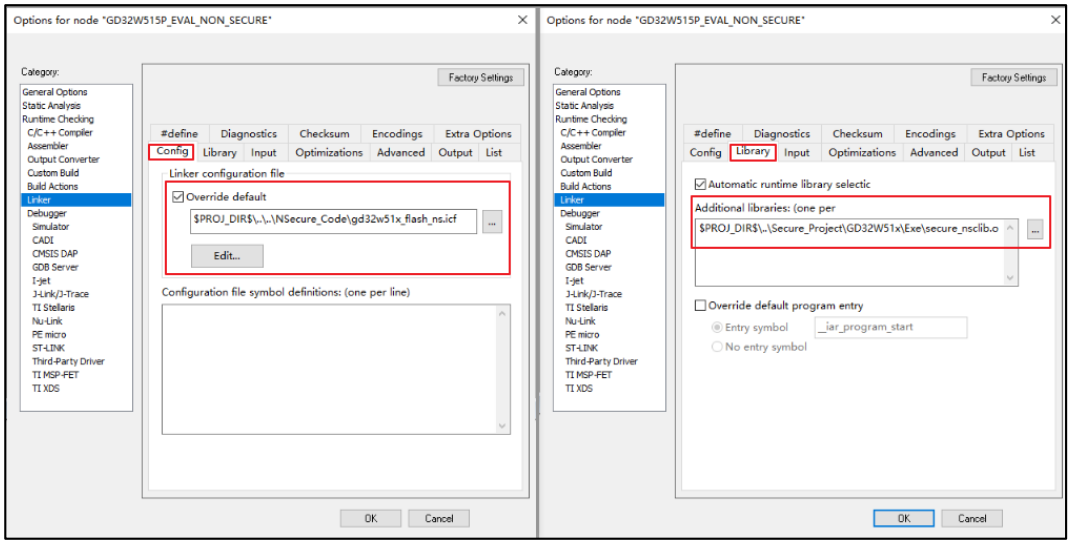
图 3-21. 选择非安全模式



3、安全代码和非安全代码使用不同的 Flash 和 SRAM，使用分散加载文件为安全代码分配正确的 Flash 和 SRAM。在 Project / Options / Linker / Config 选项卡中，选择分散加载文件，并使用安全工程导出的 NSC 函数库。见*图 3-22. 选择分散加载文件和导入 NSC 库*。

函数库导入$PROJ_DIR$\..\Secure_Project\GD32W51x\Exe\secure_nsclib.o，可直接调用安全工程中__attribute((cmse_nonsecure_entry))标识代码，这部分代码包含在 secure_nsclib.o 文件中。

图 3-22. 选择分散加载文件和导入 NSC 库



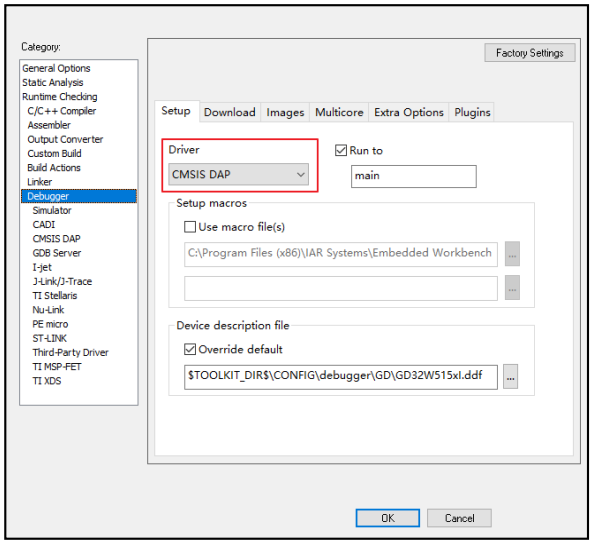
4、安全代码使用 Flash 地址为 0x08040000，大小为 0x001C0000，SRAM 地址为 0x20010000，大小为 0x10000。分散加载文件 Project_NS.sct 代码见*表 3-4. gd32w51x_flash_ns.icf 代码*。

表 3-4. gd32w51x_flash_ns.icf 代码

```
/*-Specials-*/
define symbol __ICFEDIT_intvec_start__  = 0x08040000;
/*-Memory Regions-*/
define symbol __ICFEDIT_region_ROM_start__  = 0x08040000;
define symbol __ICFEDIT_region_ROM_end__    = 0x081FFFFF;
define symbol __ICFEDIT_region_RAM_start__  = 0x20010000;
define symbol __ICFEDIT_region_RAM_end__    = 0x2001FFFF;
......
define region ROM_region = mem:[from __ICFEDIT_region_ROM_start__ to __ICFEDIT_region_ROM_end__];
define region RAM_region = mem:[from __ICFEDIT_region_RAM_start__ to __ICFEDIT_region_RAM_end__];
......
place in ROM_region    { readonly };
place in RAM_region    { readwrite, block HEAP, block CSTACK};
```

5、使用 GD-Link 进行调试和下载，在 Project/Options/Debugger 下选择 CMSIS-DAP 选项。见*图 3-23. 非安全工程设置调试器*。

图 3-23. 非安全工程设置调试器

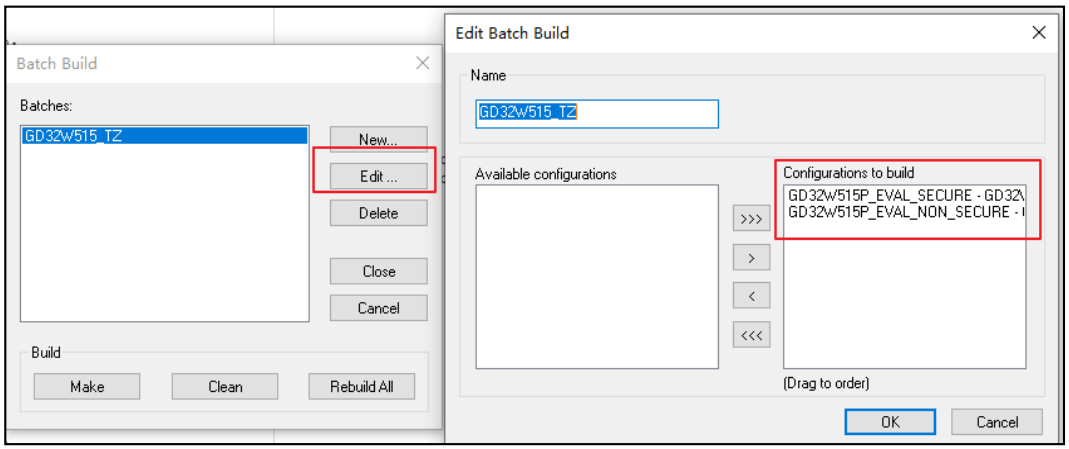


3.2.3. 编译工程

由于非安全工程需要使用安全工程生成 secure_nsclib.o 库，因此需要先编译安全工程生成该库，后编译非安全工程。可先设置安全工程为当前工程，编译安全工程，再设置非安全工程为当前工程，编译非安全工程。

也可设置 Project / Batch build 选项依次编译两个工程，见*图 3-24. 工程编译*，分别按顺序添加安全和非安全工程，完成后进行 Make 依次编译两个工程。

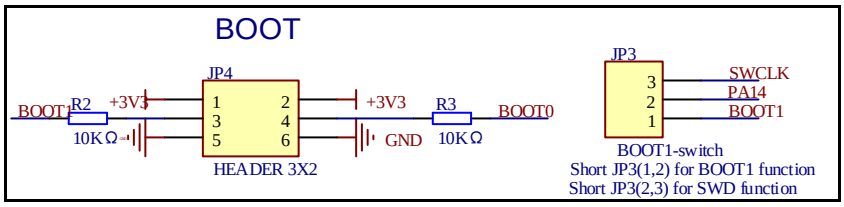
图 3-24. 工程编译



3.2.4. 下载工程

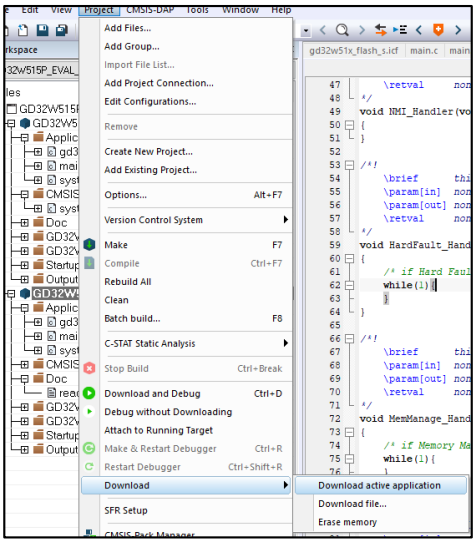
下载代码前需要先正确设置开发板，JP4 的 BOOT0/BOOT1 连接到 L，设置硬件 BOOT 选项从安全代码启动，JP3 连接到 SWD。见*图 3-25. BOOT 选项*。JP21 连接到 USART 用于串口打印。连接开发板 GD-Link 和 USART 到 PC 机上，并确保软件驱动已正确安装。

图 3-25. BOOT 选项



含有 TrustZone 功能的工程中，代码总是从安全代码启动，并跳转到非安全代码。因此先下载非安全代码，后下载安全代码。设置非安全工程为当前工程，使用 Project / Download / Download active application 下载代码，见*图 3-26. 工程下载*。

图 3-26. 工程下载



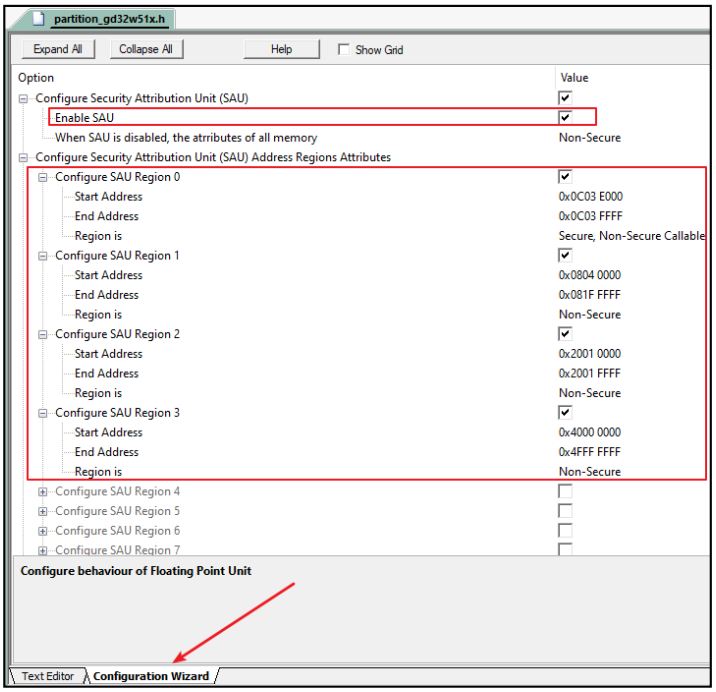
分别下载完成后需要手动按下 Reset 按键重启 MCU。重启后可观察到安全和非安全工程分别打印一条信息，并分别控制两个 LED 灯闪烁。

4. 代码解读

4.1. 安全工程

当在工程中 Project / Options for Target / Target 选项卡中，选择 Code Generation / Software Model 为 Secure Mode，则定义__ARM_FEATURE_CMSE 为 3，上电复位后执行 system_gd32w51x.c / SystemInit() / sau_region_config() 函数，该函数将根据 partition_gd32w51x.h 内容配置 SAU。partition_gd32w51x.h 文件可通过 Configuration Wizard 界面配置并生成 Text Editor 代码。如*图 4-1. SAU 配置*所示，使能 SAU，并配置四块区域，Region0 配置 NSC 函数地址为非安全可调用，与 Project_S.sct 配置对应。Region1 配置非安全代码的 Flash 地址为非安全，Region2 配置非安全代码的 SRAM 地址为非安全，与 Project_NS.sct 配置对应。Region3 配置外设地址为非安全，安全和非安全代码均可访问。

图 4-1. SAU 配置



安全工程 main 函数中进行安全域的中断使能、Systick 配置、FMC 配置、TZBMPC 配置、LED 配置、USART 配置，最后跳转非安全代码。见*表 4-1. 安全 main 代码*。

表 4-1. 安全 main 代码

```
int main(void)
{
    /* enable SecureFault handler */
    SCB->SHCSR |= SCB_SHCSR_SECUREFAULTENA_Msk;
    /* configure systick */
    systick_config();
    /* configure mark secure pages */
```

```
    if(SECM_SPAGE != (FMC_SECMCFG0 & 0x3FF) || SECM_EPAGE != ((FMC_SECMCFG0 >>
16) & 0x3FF)){
        fmc_secmark_config();
    }
    if(0U == (FMC_OBR & FMC_OBR_TZEN)){
        /* enable trustzone */
        fmc_trustzone_enable();
    }
    /* configure TZBMPC */
    tzbmpc_config();
    led_config();
    com_config();
    /* setup and jump to non-secure */
    nonsecure_init();
    while(1){
    }
}
```

fmc_secmark_config()函数配置 FMC 的 0-63 页，即前 256KB 为安全，该区域为安全代码使用。

表 4-2. FMC 配置

```
void fmc_secmark_config(void)
{
    fmc_unlock();
    ob_unlock();
    /* configure mark secure pages */
    ob_secmark_config(SECM_SPAGE, SECM_EPAGE, SECM_INDEX0);
    ob_start();
    while(0U != (FMC_SECSTAT & (FMC_SECSTAT_SECBUSY))){
    }
    ob_reload();
    ob_lock();
    fmc_lock();
}
```

tzbmpc_config()函数配置 SRAM1 为非安全，该区域为非安全代码使用。

表 4-3. SRAM 配置

```
void tzbmpc_config(void)
{
    uint16_t block_number = 0U;
    /* enable TZPCU clock */
    rcu_periph_clock_enable(RCU_TZPCU);
    /* SRAM1 is used to nonsecure code, so all blocks of SRAM1 should set to nonsecure */
```

```
    for(block_number = 0U; block_number <= TZBMPC1_BLOCK_NUMBER; block_number++){
        tzpcu_tzbmpc_block_secure_access_mode_config(TZBMPC1, block_number,
BLOCK_SECURE_ACCESS_MODE_NSEC);
    }
}
```

led_config()函数将 LED1 和 LED2 端口配置为非安全，由非安全代码控制。com_config()函数将 USART2（串口）配置为安全，由安全代码控制。

使用 __attribute((cmse_nonsecure_entry)) 标记函数 entry_cb_func_register() 和 non_secure_print()为非安全可调用，非安全代码可直接调用，用于传递非安全函数地址和信息打印。

若安全代码需要调用非安全函数时，先使用 cmse_nsfptr_create(func_addr)函数注册该地址，后使用__attribute((cmse_nonsecure_call))转化为非安全函数，最后调用该函数。非安全代码先调用安全域的 entry_cb_func_register()函数传入函数 toggle_led1 地址，后安全代码调用 nonsecure_func()实现 LED1 翻转，主要代码见*表 4-4. 安全代码调用非安全函数*。

表 4-4. 安全代码调用非安全函数

```
#define CMSE_NS_ENTRY  __attribute((cmse_nonsecure_entry))
#define CMSE_NS_CALL   __attribute((cmse_nonsecure_call))
typedef void CMSE_NS_CALL (*ns_fptr)(void);
ns_fptr nonsecure_func = (ns_fptr)NULL;
.......
CMSE_NS_ENTRY  void entry_cb_func_register(void *callback)
{
    if(callback != NULL){
    nonsecure_func = (ns_fptr)cmse_nsfptr_create(callback);
    }
}
```

4.2. 非安全工程

非安全代码导入 secure_nsclib.o 库，后可直接调用 NSC 函数。main()函数通过调用 entry_cb_func_register()函数将 toggle_led1 函数地址传给安全代码，配置非安全域 Systick、LED 端口初始化，while 循环中周期翻转 LED2 并调用 NSC 函数 non_secure_print 打印信息。

表 4-5. 安全代码调用非安全函数

```
extern void entry_cb_func_register(void *callback);
extern void non_secure_print(const char * str);
......
int main(void)
{
    entry_cb_func_register((void *)toggle_led1);
    /* configure systick */
```

```
    systick_config();
    gd_eval_led_init(LED1);
    gd_eval_led_init(LED2);
    while(1){
        /* toggle LED2 */
        gd_eval_led_toggle(LED2);
        non_secure_print("non-secure code toggle LED2.\r\n");
        delay_1ms(1000);
    }
}
```

5. 版本历史

表 5-1. 版本历史

版本号.	说明	日期
1.0	首次发布	2023 年 3 月 3 日

Important Notice

This document is the property of GigaDevice Semiconductor Inc. and its subsidiaries (the "Company"). This document, including any product of the Company described in this document (the “Product”), is owned by the Company under the intellectual property laws and treaties of the People’s Republic of China and other jurisdictions worldwide. The Company reserves all rights under such laws and treaties and does not grant any license under its patents, copyrights, trademarks, or other intellectual property rights. The names and brands of third party referred thereto (if any) are the property of their respective owner and referred to for identification purposes only.

The Company makes no warranty of any kind, express or implied, with regard to this document or any Product, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Company does not assume any liability arising out of the application or use of any Product described in this document. Any information provided in this document is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Except for customized products which has been expressly identified in the applicable agreement, the Products are designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only. The Products are not designed, intended, or authorized for use as components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, atomic energy control instruments, combustion control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or Product could cause personal injury, death, property or environmental damage ("Unintended Uses"). Customers shall take any and all actions to ensure using and selling the Products in accordance with the applicable laws and regulations. The Company is not liable, in whole or in part, and customers shall and hereby do release the Company as well as it’s suppliers and/or distributors from any claim, damage, or other liability arising from or related to all Unintended Uses of the Products. Customers shall indemnify and hold the Company as well as it’s suppliers and/or distributors harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of the Products.

Information in this document is provided solely in connection with the Products. The Company reserves the right to make changes, corrections, modifications or improvements to this document and Products and services described herein at any time, without notice.