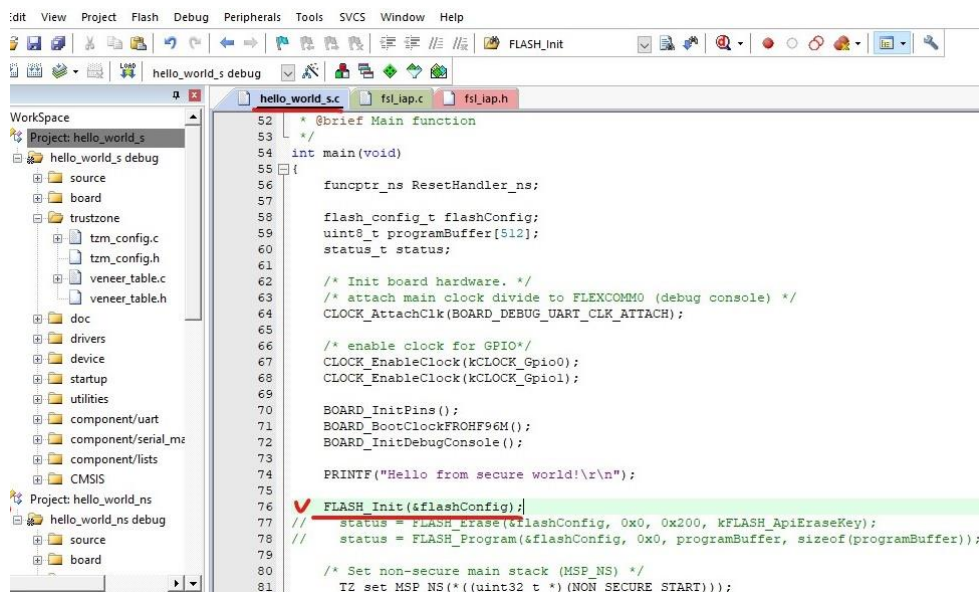


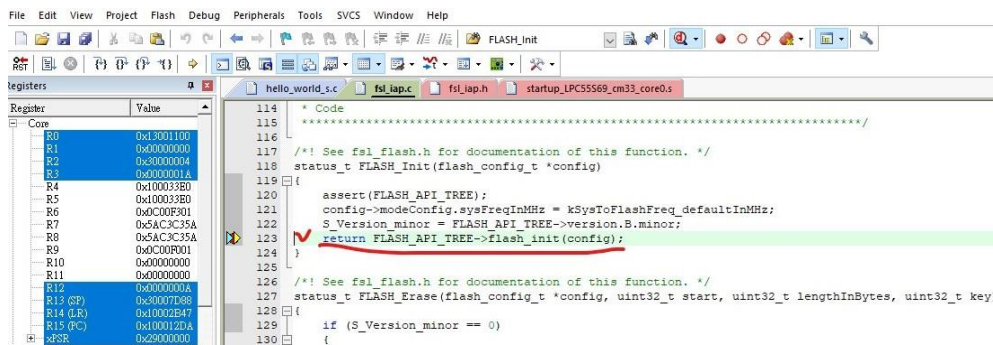
Operate peripherals under TrustZone

Some users cannot access MCU peripherals normally by adding peripheral initialization code to MCUXpresso SDK TrusZone demo. For example, when add Flash operation code in the security world, the program code jumps to HardFault_Handler after running to function FLASH_INIT(), and the execution of Flash erase and Flash program operations fails also, as follows:



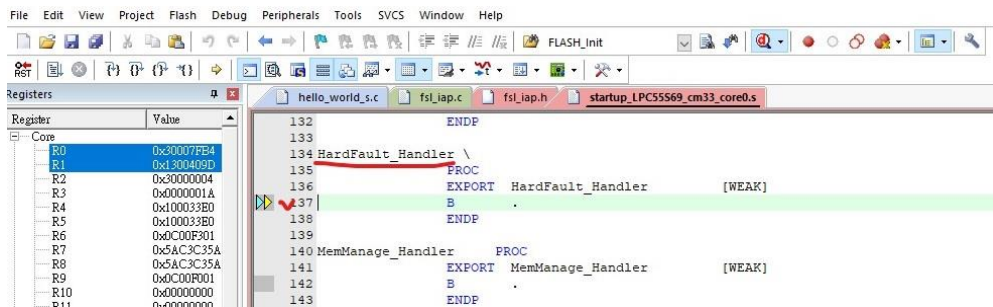
```
52  * @brief Main function
53  */
54  int main(void)
55  {
56      funcptr_ns ResetHandler_ns;
57
58      flash_config_t flashConfig;
59      uint8_t programBuffer[512];
60      status_t status;
61
62      /* Init board hardware. */
63      /* attach main clock divide to FLEXCOMM0 (debug console) */
64      CLOCK_AttachClk(BOARD_DEBUG_UART_CLK_ATTACH);
65
66      /* enable clock for GPIO*/
67      CLOCK_EnableClock(kCLOCK_Gpio0);
68      CLOCK_EnableClock(kCLOCK_Gpio1);
69
70      BOARD_InitPins();
71      BOARD_BootClockFROHF96M();
72      BOARD_InitDebugConsole();
73
74      PRINTF("Hello from secure world!\r\n");
75
76      FLASH_Init(&flashConfig);
77      status = FLASH_Erase(&flashConfig, 0x0, 0x200, kFLASH_ApiEraseKey);
78      status = FLASH_Program(&flashConfig, 0x0, programBuffer, sizeof(programBuffer));
79
80      /* Set non-secure main stack (MSP_NS) */
81      TZ_set_MSP_NS((uint32_t)(NON_SECURE_START));
```

Figure 1



```
114  * Code
115  .....
116
117  /** See fsl_flash.h for documentation of this function. */
118  status_t FLASH_Init(flash_config_t *config)
119  {
120      assert(FLASH_API_TREE);
121      config->modeConfig.sysFreqInMHz = kSysToFlashFreq_defaultInMHz;
122      S_Version_minor = FLASH_API_TREE->version.B.minor;
123      return FLASH_API_TREE->flash_init(config);
124  }
125
126  /** See fsl_flash.h for documentation of this function. */
127  status_t FLASH_Erase(flash_config_t *config, uint32_t start, uint32_t lengthInBytes, uint32_t key)
128  {
129      if (S_Version_minor == 0)
130      {
```

Figure 2



```
132      ENDF
133
134  HardFault_Handler \
135      PROC
136      EXPORT HardFault_Handler [WEAK]
137      B
138      ENDF
139
140  MemManage_Handler PROC
141      EXPORT MemManage_Handler [WEAK]
142      B
143      ENDF
```

Figure 3

```
SmarTTY - Raw Terminal
Connected to COM8 (115200 bps) Baud rate: 115200
Hello from secure world!

Flash driver API tree Demo Application...
Initializing flash driver...
Flash init successfull!!. Halting...

PFlash Example Start

Calling FLASH_Erase() API...
Unknown status

Done!
Calling FLASH_VerifyErase() API...
Unknown status

---- HALTED DUE TO FLASH ERROR! ----_
```

Figure 4

```
SmarTTY - Raw Terminal
Connected to COM8 (115200 bps) Baud rate: 115200
Hello from secure world!

Flash driver API tree Demo Application...
Initializing flash driver...
Flash init successfull!!. Halting...

PFlash Example Start
Calling FLASH_Program() API...
Alignment Error.

Calling FLASH_VerifyProgram() API...
Unknown status

---- HALTED DUE TO FLASH ERROR! ----_
```

Figure 5

Diagnosis

As shown in figure 2 and figure 3, when the program code runs to code `return VERSION_FLASH_API_TREE->flash_init(config)`, it automatically jumps to `HardFault_Handler`. `VERSION_FLASH_API_TREE` is located in the `0x1301fe00` address of the boot rom, the flash erase api is located in address `0x1300413bU`, and the flash program api is located in address `0x1300419dU` (the corresponding program code is shown in figure 6). **All above addresses are not security privilege.**

```
#define BOOTLOADER_API_TREE_POINTER ((bootloader_tree_t *)0x130010f0U)
runCmdFuncOption.commandAddr = 0x1300413bU; /*!< get the flash erase api location address in rom */
runCmdFuncOption.commandAddr = 0x1300419dU; /*!< get the flash program api location address in rom*/
```

Figure 6

From 7.5.3.1.2 *TrustZone preset data* chapter in user manual, after enabling the TrustZone configuration, users must configure the security level of the entire ROM address space to security priority (S-Priv) in order to ensure that the ROM area can be accessed normally by the security area code.

Table 212. TZM Preset value

TZM_PRESET value (offset 24, bit 13)	
0	TrustZone data not present
1	TrustZone data present

Note: Since the TrustZone configuration is enabled before a jump to the user application, the user's TrustZone configuration data must allow ROM code execution for successful transition from secure to normal mode and jump to user application. This means that user's TrustZone settings must include:

1. The whole ROM space (0x13000000-0x1301FFFF) must be configured as secure privilege.
2. When a secure MPU is used, the whole ROM space (0x13000000-0x1301FFFF) must be configured for code execution.

Figure 7

Solutions

Below is the steps of how to resolve this issue. The demo is based on MCUXpresso SDK demo hello_world_s.

Step 1: firstly we use the TEE tool integrated with MCUXpresso IDE to configure the security level of the Boot ROM address area, as shown in Figure 8, double-click the Boot-ROM area in the Memory attribution map window, and configure the sector's security level in the corresponding Security access configuration window on the left.

The screenshot shows the MCUXpresso IDE interface. The 'Memory attribution map' window is open, displaying a table of memory regions and their security configurations. The 'Boot-ROM (alias)' is highlighted in blue, indicating it is configured for 'S-Priv' security. The 'Security access configuration' window on the left shows the 'MPC Sectors' list with the 'Boot-ROM' sector selected and its security level set to 'S-Priv'. The 'Generated code' section on the right shows the 'trustzone/tzm_config.c' and 'trustzone/tzm_config.h' files.

Figure 8

Step 2: Second, when operating Flash or other peripherals in the security area, users must configure the security level of correlative peripherals to the security priority(S-Priv).

When operating flash in the SDK TrustZone demo, the MCU uses two slave peripherals, so users must configure their security level to S-Priv.

```
SysTick->CTRL = 0; /* Prevent generator systick handler */
```

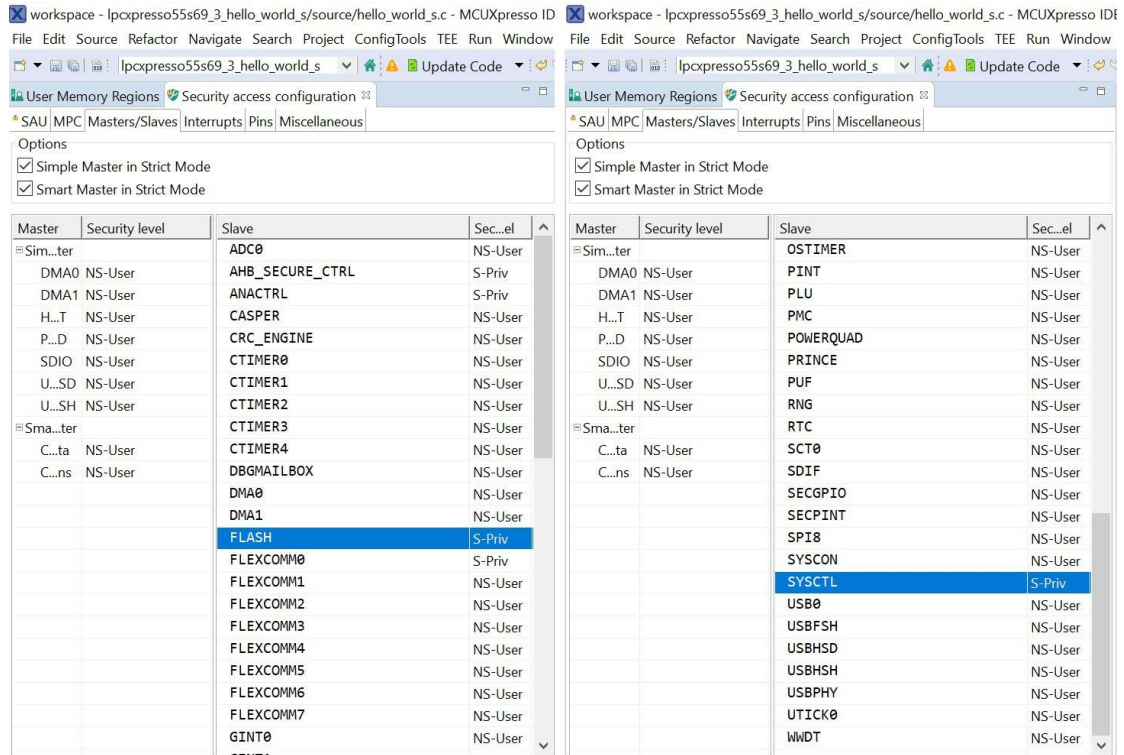


Figure 9

Please Note:

1. From the usermanual, when operating flash, the system clock frequency cannot exceed 100MHZ.
2. When using the function of `FLASH_Program()`, because the `s_buffer` is 512-byte aligned, the `BUFFER_LEN` is equal to $512/N$.

```
#define BUFFER_LEN 512 / 4
const uint32_t s_buffer[BUFFER_LEN]={1,2,3,4};
```

The above configuration of the security level can be configured through the TEE tool integrated the MCUXpresso IDE. After completing configuration, click Update Code to automatically update the relevant code in the `tzm_config.c` file, as shown in Figure 10.

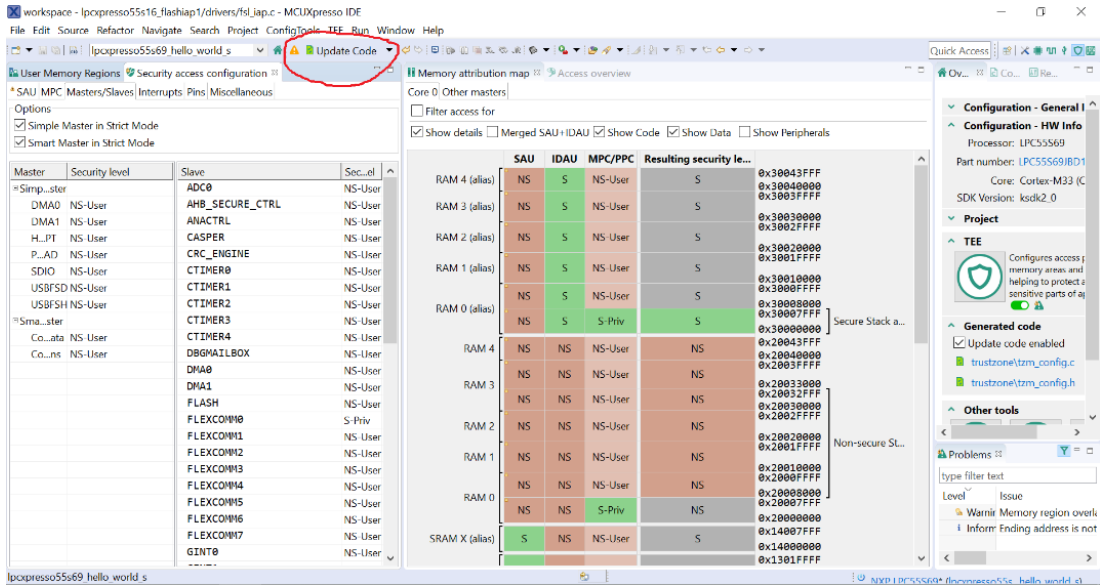


Figure 10

The updated code is shown in Figure 11 below. It is obvious that the security level settings of boot rom memory and peripheral (FLASH, SYSCTRL) have changed. If you do not use the TEE tool, you can also manually modify tzm_config.c to configure the same security options.

```

On disk: trustzone\tzm_config.c
171 /* Security level configuration of memories */
172 AHB_SECURE_CTRL->SEC_CTRL_FLASH_ROM[0].SEC_CTRL_FLASH_MEM_RULE[0] = 0x00000333U;
173 AHB_SECURE_CTRL->SEC_CTRL_FLASH_ROM[0].SEC_CTRL_FLASH_MEM_RULE[1] = 0;
174 AHB_SECURE_CTRL->SEC_CTRL_FLASH_ROM[0].SEC_CTRL_FLASH_MEM_RULE[2] = 0;
175 AHB_SECURE_CTRL->SEC_CTRL_FLASH_ROM[0].SEC_CTRL_ROM_MEM_RULE[0] = 0x33333333U;
176 AHB_SECURE_CTRL->SEC_CTRL_FLASH_ROM[0].SEC_CTRL_ROM_MEM_RULE[1] = 0x33333333U;
177 AHB_SECURE_CTRL->SEC_CTRL_FLASH_ROM[0].SEC_CTRL_ROM_MEM_RULE[2] = 0x33333333U;
178 AHB_SECURE_CTRL->SEC_CTRL_FLASH_ROM[0].SEC_CTRL_ROM_MEM_RULE[3] = 0x33333333U;
179 AHB_SECURE_CTRL->SEC_CTRL_RAMX[0].MEM_RULE[0] = 0;
180 AHB_SECURE_CTRL->SEC_CTRL_RAM0[0].MEM_RULE[0] = 0x33333333U;
181 AHB_SECURE_CTRL->SEC_CTRL_RAM0[0].MEM_RULE[1] = 0;
182 AHB_SECURE_CTRL->SEC_CTRL_RAM1[0].MEM_RULE[0] = 0;
183 AHB_SECURE_CTRL->SEC_CTRL_RAM1[0].MEM_RULE[1] = 0;
184 AHB_SECURE_CTRL->SEC_CTRL_RAM2[0].MEM_RULE[0] = 0;
185 AHB_SECURE_CTRL->SEC_CTRL_RAM2[0].MEM_RULE[1] = 0;
186 AHB_SECURE_CTRL->SEC_CTRL_RAM3[0].MEM_RULE[0] = 0;
187 AHB_SECURE_CTRL->SEC_CTRL_RAM3[0].MEM_RULE[1] = 0;
188 AHB_SECURE_CTRL->SEC_CTRL_RAM4[0].MEM_RULE[0] = 0;
189 AHB_SECURE_CTRL->SEC_CTRL_USB_HS[0].MEM_RULE[0] = 0;
190
191 /* Security level configuration of peripherals */
192 AHB_SECURE_CTRL->SEC_CTRL_APB_BRIDGE[0].SEC_CTRL_APB_BRIDGE0_MEM_CTRL0 = 0x00000333U;
193 AHB_SECURE_CTRL->SEC_CTRL_APB_BRIDGE[0].SEC_CTRL_APB_BRIDGE0_MEM_CTRL1 = 0;
194 AHB_SECURE_CTRL->SEC_CTRL_APB_BRIDGE[0].SEC_CTRL_APB_BRIDGE0_MEM_CTRL2 = 0x00030000U;
195 AHB_SECURE_CTRL->SEC_CTRL_APB_BRIDGE[0].SEC_CTRL_APB_BRIDGE1_MEM_CTRL0 = 0x00030000U;
196 AHB_SECURE_CTRL->SEC_CTRL_APB_BRIDGE[0].SEC_CTRL_APB_BRIDGE1_MEM_CTRL1 = 0;
197 AHB_SECURE_CTRL->SEC_CTRL_APB_BRIDGE[0].SEC_CTRL_APB_BRIDGE1_MEM_CTRL2 = 0x00030000U;
198 AHB_SECURE_CTRL->SEC_CTRL_APB_BRIDGE[0].SEC_CTRL_APB_BRIDGE1_MEM_CTRL3 = 0;
199 AHB_SECURE_CTRL->SEC_CTRL_AHB_PORTS_SLAVE0_RULE = 0x03000000U;
200 AHB_SECURE_CTRL->SEC_CTRL_AHB_PORTS_SLAVE1_RULE = 0x00030000U;

```

Figure 11

Third-party tools users:

Because many users are accustomed to using third-party development tools such as Keil or IAR, but these IDEs do not integrate the TEE tool, users need to check the configuration

requirements of related registers in user manual when modifying the security level of related areas and peripherals in TrusZone, and update the associated code in the `tzm_config.c` file (similar to Figure 11) to complete the related configuration. In addition, NXP released the MCUXpresso Config Tools, which integrates MCU-related configuration functions. Users can download and install this tool to perform configurations and update codes. The download link is as follows:

<https://www.nxp.com/design/software/development-software/mcuxpresso-software-and-tools/mcuxpresso-config-tools-pins-clocks-peripherals:MCUXpresso-Config-Tools>

Introduction of MCUXpresso Config Tools

After the tool is installed, open the configuration tool, select *Create a new configuration based on an SDK example or hello world project*, click *Next*, as shown in Figure 12:

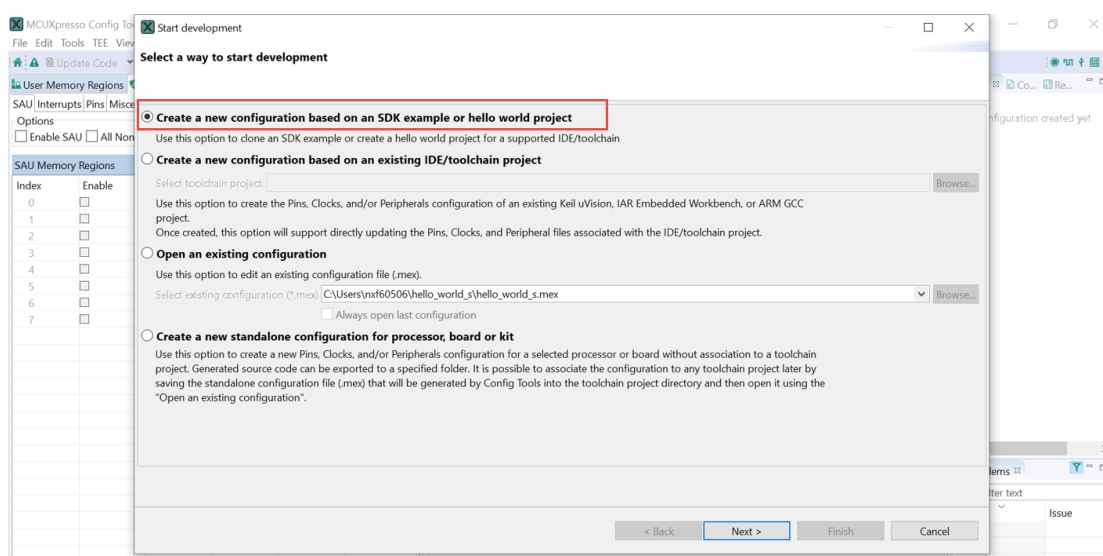


Figure 12

In *Start Development* window, follow below steps to generate project. As shown in Figure 13.

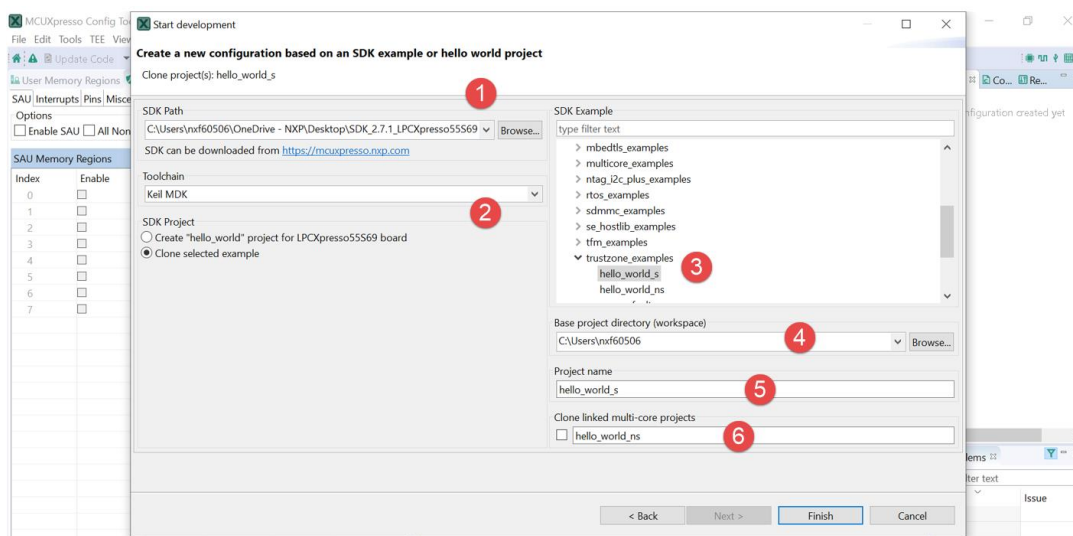


Figure 13

After the tzm_config.c file is updated, copy or import it to the corresponding folder of KEIL or IAR third-party development tools, and it can be used normally.