Generating interrupt in NON-security world

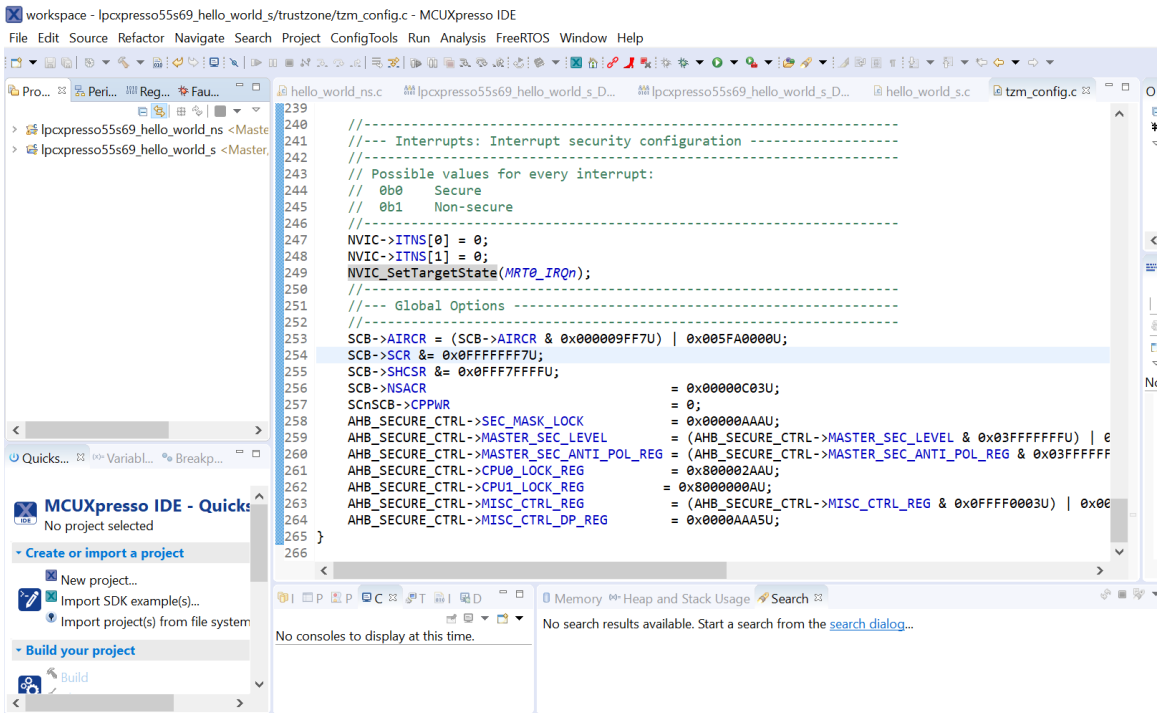
For the CM33 of LPC55S6x family, the trust zone module is integrated, the memory space and peripherals are classified as security and non-security space. In order to generate interrupt in non-security mode, the NVIC module especially the NVIC_ITNSx register must be initialized in security mode so that interrupt module can generate interrupt in non-security mode.

The example demos that MRT0 module generates interrupt in non-security mode, the NVIC module is initialized at security mode, MRT0 is initialized at non-security mode.

The project is based on MCUXpresso IDE ver11.1 tools, LPC55S69-EVK board and SDK_2.x_LPCXpresso55S69 SDK package version 2.7.1.

1) security and non-security mode introduction

The SDK package for LPC55S6x has an example:



In thetzm_config.c, add the following code:

```
//enable SYSCON module to be accessed by non-secure side
AHB_SECURE_CTRL->SEC_CTRL_APB_BRIDGE[0].SEC_CTRL_APB_BRIDGE0_MEM_CTRL0 =
0xFCCCCCFCU;
//enable MRT module to be accessed by non-secure side
AHB_SECURE_CTRL->SEC_CTRL_APB_BRIDGE[0].SEC_CTRL_APB_BRIDGE0_MEM_CTRL1 =
0xFCCCFFCCU;
//enable PIO1_7 pin to be accessed by non-secure side
AHB_SECURE_CTRL->SEC_GPIO_MASK1&=~(1<<7);
//enable MRT0 interrupt on non-secure side
NVIC_SetTargetState(MRT0_IRQn);
```

The lpcxpresso55s69_hello_world_s is the code that the LPC55S69 runs in

the security mode.

The lpcxpresso55s69_hello_world_ns is the code that the lpc55S69 runs in non-security mode.

Users can compile the two project, then download ONLY the lpcxpresso55s69_hello_world_s project, the lpcxpresso55s69_hello_world_ns is loaded to flash automatically by the lpcxpresso55s69_hello_world_s project.

The lpcxpresso55s69_hello_world_s is loaded into PROGRAM_FLASH (rx) : ORIGIN = 0x10000000, so the LPC55S69 runs in security mode after Reset.

2) NVIC-ITNSx registers introduction

For the CM33 core, there is additional NVIC-ITNSx registers,

Table 4-35 NVIC registers summary

Address	Name	Type	Required privilege	Reset value	Description
0xE000E100-0xE000E13C	NVIC_ISER0-NVIC_ISER15	RW	Privileged	0x00000000	*4.4.2 Interrupt Set Enable Registers* on page 4-305
0XE000E180- 0xE000E1BC	NVIC_ICER0-NVIC_ICER15	RW	Privileged	0x00000000	*4.4.3 Interrupt Clear Enable Registers* on page 4-306
0XE000E200- 0xE000E23C	NVIC_ISPR0-NVIC_ISPR15	RW	Privileged	0x00000000	*4.4.4 Interrupt Set Pending Registers* on page 4-307
0XE000E280- 0xE000E2BC	NVIC_ICPR0-NVIC_ICPR15	RW	Privileged	0x00000000	*4.4.5 Interrupt Clear Pending Registers* on page 4-307
0xE000E300-0xE000E33C	NVIC_IABR0-NVIC_IABR15	RW	Privileged	0x00000000	*4.4.6 Interrupt Active Bit Registers* on page 4-308
0xE000E380-0xE000E3BC	NVIC_ITNS0-NVIC_ITNS15	RW[v]	Privileged	0x00000000	*4.4.7 Interrupt Target Non-secure Registers* on page 4-308.
0xE000E400-0xE000E5DC	NVIC_IPR0-NVIC_IPR119	RW	Privileged	0x00000000	*4.4.8 Interrupt Priority Registers* on page 4-309
0xE000EF00	STIR	WO	Configurable[w]	0x00000000	*4.4.9 Software Trigger Interrupt Register* on page 4-310

Table 4-42 NVIC_ITNSn bit assignments

Bits	Name	Function
[31:0]	ITNS	Interrupt Targets Non-secure bits. For ITNS[m] in NVIC_ITNSn, this field indicates and allows modification of the target Security state for interrupt 32n+m. **0** The interrupt targets Secure state. **1** The interrupt targets Non-secure state.

The CM33 core has two interrupt vector table located at different memory space, one is for security mode, another is for non-security mode. In other words, each interrupt source has two interrupt vector, one is located in security interrupt vector table, another is located at non-security interrupt vector table.

If user wants to generate interrupt in non-security mode, user has to set the interrupt source bit in NVIC_ITNSx register in the secure project. In this way, the security project can control whether the interrupt source is allowed or not allowed in non-security project.

Generally, the code to initialize the NVIC_ITNSx register is located in the tzm_config.c to initialize the trust zone in the security project.

3) MRT0 and NVIC code

3.1 The following code is required to be located at the lpcxpresso55s69_hello_world_ns project, I copy the hello_world_ns here

In non-secure side, the code initializes the PIO1_7 as GPIO output pin, which drives a LED on LPC55S69-EVK board.

It also initializes the MRT0 module so that it can generate interrupt. In the ISR of MRT0, a LED is toggling.

It initializes the NVIC so that MRT0 can trigger interrupt.

Result: after the code runs, the green LED toggles.

Hello_world_ns.c code:

```
#define PRINTF_NSE DbgConsole_Printf_NSE
void Init_PIO1_7_NS(void);
void Init_MRT0_NS(void);
void NS_InterruptInit(void);
void SystemInit(void)
{

}

/*!
 * @brief Main function
 */
int main(void)
{
    int result;

    /* set BOD VBAT level to 1.65V */
    POWER_SetBodVbatLevel(kPOWER_BodVbatLevel1650mv, kPOWER_BodHystLevel50mv,
false);

    PRINTF_NSE("Welcome in normal world!\r\n");
    PRINTF_NSE("This is a text printed from normal world!\r\n");

    result = StringCompare_NSE(&strcmp, "Test1\r\n", "Test2\r\n");
    if (result == 0)
    {
        PRINTF_NSE("Both strings are equal!\r\n");
    }
    else
    {
        PRINTF_NSE("Both strings are not equal!\r\n");
    }
    Init_PIO1_7_NS();
    __asm("nop");
    Init_MRT0_NS();
    __asm("nop");
    NS_InterruptInit();
    __asm("nop");
    while (1)
```

```
    {
    //GPIO->NOT[1]=1<<7;
    __asm("nop");
    }
}

void Init_PIO1_7_NS(void)
{
    //enable gated GPIOP1 clock
    __asm("nop");
    SYSCON->AHBCLKCTRL.AHBCLKCTRL0|=1<<15;
    //set the mux
    __asm("nop");
    //set GPIO direction reg
    GPIO->DIR[1]|=1<<7;
    //toggle the PIO1_4
    GPIO->NOT[1]=1<<7;
    __asm("nop");
    GPIO->NOT[1]=1<<7;
    __asm("nop");
    GPIO->NOT[1]=1<<7;

}


void Init_MRT0_NS(void)
{
    SYSCON->AHBCLKCTRL.AHBCLKCTRL1|=1<<0;
    //set up PIT0
    MRT0->CHANNEL[0].INTVAL=12000000;
    //repeated interrupt mode
    MRT0->CHANNEL[0].CTRL=0x00;
    //enable MRT channel0
    MRT0->CHANNEL[0].CTRL=1<<0;
}


void NS_InterruptInit(void)
{
    NVIC->IPR[9]=0x00;
    __asm("nop");
    NVIC->ISER[0]|=1<<9;
    __asm("nop");
    NVIC->ICPR[0]|=1<<9;
    __asm("cpsie i");
}

void MRT0_IRQHandler(void)
{
    //toggle LED
    //clear flag
    MRT0->CHANNEL[0].STAT|=1<<0;
    GPIO->NOT[1]=1<<7;

}
```

4. conclusion

In order to generate interrupt in non-security side, in the secure project, customer has to set up the registers in Trusted Execution Environment so that the corresponding peripherals can be accessed by the non-secure side, for example MRT, GPIO pin, and enabling a specific interrupt source.

In the non-security project, the PIO1_7, MRT0 and NVIC are initialized so that MRT0 can trigger interrupt on non-security project, in the MRT0 ISR, a PIIO1_7 pin is toggled, a Green LED is flashing.