

Lecture3

CORTEX M_RESET SEQUENCE

By: Abdelrahman matarawy

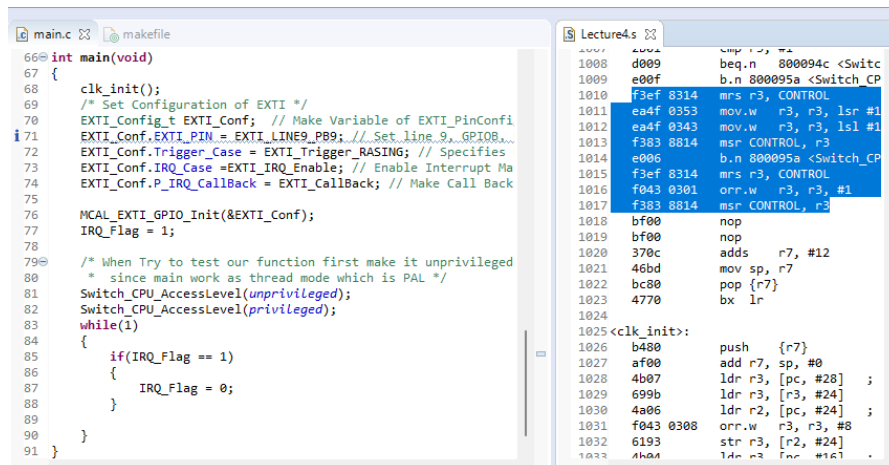
✚ Switch Between Privileged and Unprivileged:

○ Code:

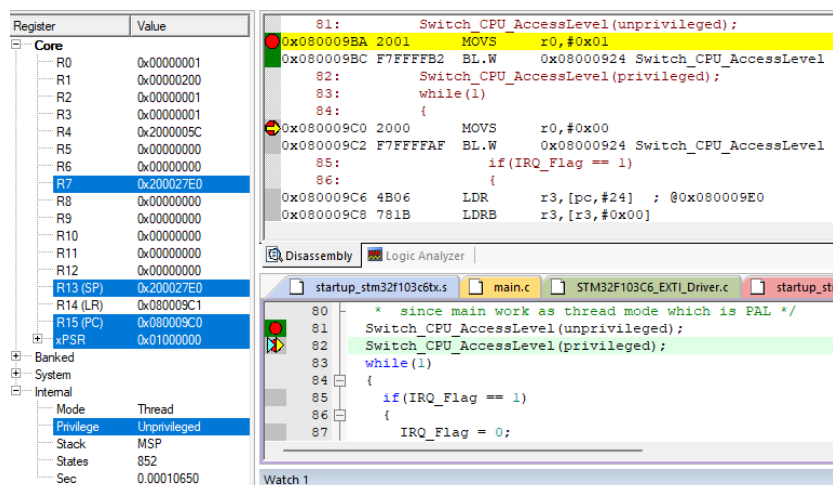
```

34 enum CPU_AccessLevel{
35     privileged,
36     unprivileged
37 };
38
39 void Switch_CPU_AccessLevel(enum CPU_AccessLevel LEVEL)
40 {
41     switch(LEVEL)
42     {
43     case privileged:
44         /* Clear Bit 0 in CONTROL register */
45         __asm("MRS R3, CONTROL \n\t" // Store value of Control register to R3
46             "LSR R3, R3, #0x01 \n\t" // Shift R3 to right to remove bit 0
47             "LSL R3, R3, #0x01 \n\t" // Shift R3 to left to clear bit 0
48             "MSR CONTROL, R3"); // load R3 value to Control Register after edit value of bit 0
49         break;
50     case unprivileged:
51         /* Set Bit 0 in CONTROL register */
52         __asm("MRS R3, CONTROL \n\t" // Store value of Control register to R3
53             "ORR R3, R3, #0x01 \n\t" // Set bit 0
54             "MSR CONTROL, R3"); // load R3 value to Control Register after edit value of bit 0
55         break;
56     }
57 }

```



○ Simulation:



➤ Note:

Not allowed to write in CONTROL register when you are unprivileged, so we handle it in ISR as handler mode work as PAL to it have access to write on Control.

What would happen if we switched to unprivileged before configuring NVIC?

○ Code:

```
65
66 int main(void)
67 {
68     clk_init();
69
70     /* Test what will happen if change NPAL before init the EXTI */
71     Switch_CPU_AccessLevel(unprivileged);
72
73     /* Set Configuration of EXTI */
74     EXTI_Config_t EXTI_Conf; // Make Variable of EXTI_PinConfig_t structure
75     EXTI_Conf.EXTI_PIN = EXTI_LINE9_PB9; // Set line 9, GPIOB, Pin9 an EXTI
76     EXTI_Conf.Trigger_Case = EXTI_Trigger_RISING; // Specifies Rising Trigger
77     EXTI_Conf.IRQ_Case = EXTI_IRQ_Enable; // Enable Interrupt Mask
78     EXTI_Conf.P_IRQ_Callback = EXTI_Callback; // Make Call Back Function point to ISR
79
80     MCAL_EXTI_GPIO_Init(&EXTI_Conf);
81     IRQ_Flag = 1;
82
83     /* When Try to test our function first make it unprivileged
84      * since main work as thread mode which is PAL */
85     //Switch_CPU_AccessLevel(unprivileged);
86     Switch_CPU_AccessLevel(privileged);
```

○ Simulation:

The screenshot displays a debugger interface with two main panes. The left pane, titled 'Registers', shows the state of the processor's registers. The right pane, titled 'Disassembly', shows the assembly code being executed. The code includes a 'break;' statement at line 55, followed by a 'NOP' instruction at line 56. The code then switches to unprivileged mode and attempts to write to the CONTROL register, which causes a hard fault. The stack trace shows the program branching to a hard fault handler.

➤ Note:

The Stack branch to strange address, we find that it loops in hard fault handler because not allow to access NVIC during unprivileged.

Switch Between them using ISR:

Code:

```
59 void EXTI_Callback(void)
60 {
61     IRQ_Flag = 1;
62     Switch_CPU_AccessLevel(privileged);
63 }
64
65 int main(void)
66 {
67     clk_init();
68
69     /* Set Configuration of EXTI */
70     EXTI_Config_t EXTI_Conf; // Make Variable of EXTI_PinConfig_t structure
71     EXTI_Conf.EXTI_PIN = EXTI_LINE9_PB9; // Set line 9, GPIOB, Pin9 an EXTI
72     EXTI_Conf.Trigger_Case = EXTI_Trigger_RISING; // Specifies Rising Trigger
73     EXTI_Conf.IRQ_Case = EXTI_IRQ_Enable; // Enable Interrupt Mask
74     EXTI_Conf.P_IRQ_Callback = EXTI_Callback; // Make Call Back Function point to ISR
75
76     MCAL_EXTI_GPIO_Init(&EXTI_Conf);
77     IRQ_Flag = 1;
78
79     /* When Try to test our function first make it unprivileged
80      * since main work as thread mode which is PAL */
81     Switch_CPU_AccessLevel(unprivileged);
82     //Switch_CPU_AccessLevel(privileged);
83     while(1)
```

Simulation:

The first screenshot shows the IDE interface with the C code from the previous block. The execution is paused at line 85, which is an if-statement checking the IRQ_Flag. The Register window on the left shows the current state of the processor registers, including R0 through R15, SP, and PSR. The Watch window at the bottom shows the current value of the IRQ_Flag variable.

The second screenshot shows the same IDE interface, but with the execution paused at line 62, where the Switch_CPU_AccessLevel function is called with the privileged argument. The Register window shows the state of the registers, and the Watch window shows the current value of the IRQ_Flag variable. The General Purpose I/O (GPIO) configuration window is also visible, showing the settings for the EXTI line 9.