HW4-v1

1. 교재 2.6.3의 내용을 이해하여, late arrival이 발생하도록 코드를 작성하고, 실제 late arrival이 발생함을 확인하고자 한다.

(a) 전체 코드 (main.c)를 답안에 포함시키고, 어떤 원리에 의해서 late arrival이 발생하였음을 보일수 있는지 설명하십시오. main.c 코드에 interrupt를 발생시키기 전에 CONTROL register 값을 '0'으로 assign하는 코드를 포함시켜야 한다. (interrupt 시에 extended stack frame이 사용되지 않도록 하기 위하여.) (10 점)

```
1  #include "stm32f4xx.h"
2
3  volatile static uint32_t counter = 0;
4
5  void EXTI0_IRQHandler(void);
6  void SysTick_Handler(void);
7  int main(void);
8
9  void EXTI0_IRQHandler(void) {
10   counter++;
11 }
12
13 void SysTick_Handler(void) {
14   SysTick->CTRL &= ~(0x01UL);
15   counter += 10;
16 }
17
18 int main(void)
19 {
20
21   NVIC_SetPriorityGrouping(5);
22
23   NVIC_SetPriority(6, 0x60>>4);
24   NVIC_SetPriority(-1, 0x00>>4);
25
26   NVIC_EnableIRQ(6);
27
28   __set_CONTROL(0);
29
30   SysTick->LOAD = 25;
31   SysTick->VAL =0;
32   SysTick->CTRL = (0x01 | (0x01 << 1) | (0x01 <<2 ));
33
34   NVIC_SetPendingIRQ(6);
35
36   while(1) {
37   }
38 }
39
```

.

Systick의 우선순위를 EXTI0 우선순위보다 높게 설정하고, lne 30 ~ line 32의 코드를 이용해서 일정 시간 후에 SysTick interrupt가 발생하도록 동작시킨다. 그 다음 EXTI0 interrupt를 발생시키

면, EXTI0 interrupt에 대한 stacking 하는 동안 systick interrupt가 발생하도록 할 수가 있을 것이다. (SysTick->LOAD 의 설정 값이 중요하다.)

CONTROL register 값을 '0'으로 만드는 코드는 라인 28에 있는 __set_control(0) 이란 코드이다.

((b) ~ (d) 의 동작 비교를 위해서 EXTI0 ISR 시작점 (라인 9), SysTick ISR 시작점 (line 13)과 systick 설정 전인 라인 30에 BP를 잡고 디버깅을 한다.)

- line 30에서의 SP = 0x20000650 이다.

(b) SysTick->LOAD 에 assign 하는 값에 따라 late arrival이 발생하게 된다. 디버거를 통해서 late arrival이 발생하였음을 어떻게 구분할 수 있는가 설명하고, Late arrival이 발생하는 값의 범위를 구하시오. (10점)

Sol)

SysTick LOAD register 값의 범위: 19 ~ 30

EXTI0 ISR과 SysTick ISR: SysTick ISR BP에서 먼저 중단됨.

- 이때 SP=0x20000628
- 이때 아래에서 보는 바와 같이 SysTick interrupt는 active 상태이고, EXT!0는 pending 상태임. 즉 EXTI0 interrupt가 pending된 다음, systick interrupt 발생한 상태이므로 late arrival이 발생한 상태임..

Idx	Source	Name	E	P	A	Priority
15	System Tick Timer	SYSTICK	1	0	1	0 = 0 s0
17	PVD through EXTI line de...	PVD	0	0	0	0 = 0 s0
18	Tamper and TimeStamp i...	TAMP_STAMP	0	0	0	0 = 0 s0
19	RTC Wakeup interrupt thr...	RTC_WKUP	0	0	0	0 = 0 s0
20	FLASH global interrupt	FLASH	0	0	0	0 = 0 s0
21	RCC global interrupt	RCC	0	0	0	0 = 0 s0
22	EXTI Line0 interrupt	EXTI0	1	1	0	96 = 1 s32

Stack 내용도 다음과 같다. 다음 2가지 형태가 존재한다. (LOAD 값에 따라서)

(1)

```
0x20000628: 00000040 E000E200 00000000 E000E014 20000040 080004B9
0x20000640: 080003D2 41000200 00000000 06000005 00000000 00000006
```

즉 다음 실행될 코드의 주소가 0x080003D2이고 이 부분이 assembly code가 바로 NVIC ISPR의 해당 bit를 set 시킨 다음, 실행되는 assembly 코드이므로, EXTI0 interrupt가 pending된 상태임을 알 수 있다.

```
  1764:     NVIC->ISPR[(((uint32_t)IRQn) >> 5UL)] = (uint32_t)(1UL << (((uint32_t)IRQn) & 0x1FUL));
0x080003B8 F99D1003  LDRSB          r1,[sp,#0x03]
0x080003BC F001021F  AND            r2,r1,#0x1F
0x080003C0 2001      MOVS           r0,#0x01
0x080003C2 4090      LSLS           r0,r0,r2
0x080003C4 094A      LSRS           r2,r1,#5
0x080003C6 F24E2100  MOVW           r1,#0xE200
0x080003CA F2CE0100  MOVT           r1,#0xE000
0x080003CE F8410022  STR            r0,[r1,r2,LSL #2]
  1765:   }
0x080003D2 E7FF      B              0x080003D4
```

(2)

```
0x20000628: 00000040 E000E200 00000000 E000E014 20000040 080004B9
0x20000640: 080003D4 41000200 00000000 06000005 00000000 00000006
```

즉 다음 실행될 코드의 주소가 0x080003D4이고 이 부분이 assembly code가 바로 NVIC ISPR의 해당 bit를 set 시킨 다음, 2번째 실행되는 assembly 코드이므로, EXTI0 interrupt가 pending 된 상태임을 알 수 있다

```
  1764:     NVIC->ISPR[(((uint32_t)IRQn) >> 5UL)] = (uint32_t)(1UL << (((uint32_t)IRQn) & 0x1FUL));
0x080003B8 F99D1003  LDRSB          r1,[sp,#0x03]
0x080003BC F001021F  AND            r2,r1,#0x1F
0x080003C0 2001      MOVS           r0,#0x01
0x080003C2 4090      LSLS           r0,r0,r2
0x080003C4 094A      LSRS           r2,r1,#5
0x080003C6 F24E2100  MOVW           r1,#0xE200
0x080003CA F2CE0100  MOVT           r1,#0xE000
0x080003CE F8410022  STR            r0,[r1,r2,LSL #2]
  1765:   }
0x080003D2 E7FF      B              0x080003D4
  1766: }
0x080003D4 B001      ADD            sp,sp,#0x04
```

(c) (b)에서 구한 값보다 '1' 적은 값을 assign 하였을 때, late arrival이 발생하지 않았다고 추정한 이유를 설명하시오. (5 점)

Sol.)

SysTick LOAD register 값: 18

EXTI0 ISR과 SysTick ISR: SysTick ISR BP에서 먼저 중단됨.

- 이때 SP=0x20000628
- 이때 아래에서 보는 바와 같이 SysTick interrupt는 active 상태이고, EXT!0는 no pending and no active 상태임. 즉 EXTI0 interrupt가 pending되기 전에 systick interrupt 발생한 상태이므로 late arrival이 아님.

Idx	Source	Name	E	P	A	Priority
15	System Tick Timer	SYSTICK	1	0	1	0 = 0 s0
17	PVD through EXTI line de...	PVD	0	0	0	0 = 0 s0
18	Tamper and TimeStamp i...	TAMP_STAMP	0	0	0	0 = 0 s0
19	RTC Wakeup interrupt thr...	RTC_WKUP	0	0	0	0 = 0 s0
20	FLASH global interrupt	FLASH	0	0	0	0 = 0 s0
21	RCC global interrupt	RCC	0	0	0	0 = 0 s0
22	EXTI Line0 interrupt	EXTI0	1	0	0	96 = 1 s32

Stack 내용도 다음과 같다.

```
0x20000628: 00000040 E000E200 00000000 E000E014 20000040 080004B9
0x20000640: 080003CE 41000200 00000000 06000005 00000000 00000006
```

즉 다음 실행될 코드의 주소가 0x080003CE이고 이 부분이 assembly code가 바로 NVIC ISPR의 해당 bit를 set 시키는 코드이므로, 아직 EXTI0 interrupt가 pending되지 않은 상태임을 알 수 있다.

```
  1764:     NVIC->ISPR[(((uint32_t)IRQn) >> 5UL)] = (uint32_t)(1UL << (((uint32_t)IRQn) & 0x1FUL));
0x080003B8 F99D1003  LDRSB          r1,[sp,#0x03]
0x080003BC F001021F  AND            r2,r1,#0x1F
0x080003C0 2001      MOVS           r0,#0x01
0x080003C2 4090      LSLS           r0,r0,r2
0x080003C4 094A      LSRS           r2,r1,#5
0x080003C6 F24E2100  MOVW           r1,#0xE200
0x080003CA F2CE0100  MOVT           r1,#0xE000
0x080003CE F8410022  STR            r0,[r1,r2,LSL #2]
```

(d) (b)에서 구한 값보다 '1' 큰 값을 assign 하였을 때, late arrival이 발생하지 않았다고 추정한 이유를 설명하시오. (5 점)

Sol.)

SysTick LOAD register 값: 31

EXTI0 ISR과 SysTick ISR: SysTick ISR BP에서 먼저 중단됨.

- 이때 SP=0x20000608 (stacking이 2번 발생한 것을 알 수 있음.)
- 이때 아래에서 보는 바와 같이 SysTick interrupt는 active 상태이고, EXT!0도 active 상태임. 즉 디버거에서는 SysTick interrupt가 먼저 발생한 것처럼 보이지만 stack과 NVIC 상태를 보면 EXTI0 ISR이 시작되고 preemption이 발생하여 systick ISR이 실행되고 있음을 알 수 있다.. 따라서 이 경우는 late arrival 이 발생한 상황이 아니다.

Idx	Source	Name	E	P	A	Priority
15	System Tick Timer	SYSTICK	1	0	1	0 = 0 s0
17	PVD through EXTI line de...	PVD	0	0	0	0 = 0 s0
18	Tamper and TimeStamp i...	TAMP_STAMP	0	0	0	0 = 0 s0
19	RTC Wakeup interrupt thr...	RTC_WKUP	0	0	0	0 = 0 s0
20	FLASH global interrupt	FLASH	0	0	0	0 = 0 s0
21	RCC global interrupt	RCC	0	0	0	0 = 0 s0
22	EXTI Line0 interrupt	EXTI0	1	0	1	96 = 1 s32

Stack 내용도 다음과 같다.

```
Address: 0x20000608
0x20000608: 00000040 E000E200 00000000 E000E014 20000040 FFFFFFF9
0x20000620: 080002D8 41000016 00000040 E000E200 00000000 E000E014
0x20000638: 20000040 080004B9 080003D4 41000200 00000000 06000005
```

즉 다음 실행될 코드의 주소가 0x080002D8이고 이 부분의 assembly code가 바로 EXTI0 ISR 루틴의 시작 코드이므로, 아직 EXTI0 interrupt가 이미 동작된 상태임을 알 수 있다.

```
     9: void EXTI0_IRQHandler(void) {
    10:         counter++;
0x080002D8 F2400160  MOVW          r1,#0x60
0x080002DC F2C20100  MOVT          r1,#0x2000
0x080002E0 6808      LDR           r0,[r1,#0x00]
```