Interrupt Priority

2010.02.08

Interrupt Priority 시험

- 3가지 예제 코드
  - IntPriority1_PriorityGroup
  - IntPriority2_SubPriority
  - IntPriority3_ChangePriority
- Priority group과 Sub priority에 대한 것을 2개의 예제를 통해서 살펴보고, 동작 중에 이 priority를 변화시켜보는 예제까지 다룰 예정

Cortex-M3 Priority group

- Cortex-M3는 ARM7, ARM9에 비해 월등히 많은 수의 인터럽트 지원
  - 시스템에서 지원하는 것과 외부 인터럽트를 포함해서 256개까지 사용 가능
  - 또한 각각 **따로 우선 순위를 지정할 수 있다**.
- 많은 인터럽트가 우선 순위를 갖게 되면 priority control이 무척 복잡
  - 그래서 NVIC는 **priority grouping을 지원**.
- Application Interrupt and Reset Control Register (SCB_AIRCR)의 PRIGROUP field를 사용해서 모든 PRI_N을 pre-emption priority field와 subpriority field로 나누어서 사용
- pre-emption priority group을 group priority로서 사용

PRIGROUP[2:0]	Binary point position	Pre-emption field	Subpriority field	Number of pre-emption priorities	Number of subpriorities
b000	bxxxxxxx.y	[7:1]	[0]	128	2
b001	bxxxxxx.yy	[7:2]	[1:0]	64	4
b010	bxxxxx.yyy	[7:3]	[2:0]	32	8
b011	bxxxx.yyyy	[7:4]	[3:0]	16	16
b100	bxxx.yyyyy	[7:5]	[4:0]	8	32
b101	bxx.yyyyyy	[7:6]	[5:0]	4	64
b110	bx.yyyyyyy	[7]	[6:0]	2	128
b111	b.yyyyyyyy	None	[7:0]	0	256

STM32 CPU group priority

- 프로세서 설계 시 이 부분을 좀더 간략하게 만들 수 있다.
  - 낮은 부분의 비트는 모두 0으로 만들어서 사용. 예를 들면 PRI_N[7:4]만 사용하고 나머지는 다 0으로 만들어서 사용하도록 설계. STM32 CPU 방식

PRIGROUP [2:0]	Interrupt priority level value, PRI_*N*[7:4]			Number of	
	Binary point[(1)]	Group priority bits	Subpriority bits	Group priorities	Sub priorities
0b100	0bxxx.y	[7:5]	[4]	8	2
0b101	0bxx.yy	[7:6]	[5:4]	4	4
0b110	0bx.yyy	[7]	[6:4]	2	8
0b111	0b.yyyy	None	[7:4]	1	16

- PRIGROUP이 가질 수 있는 값은 0b100부터 0b111로 설정
  - Group priority 설정 부분은 3 비트부터 하나도 없는 것까지 설정 가능

b011	bxxxx.yyyy	[7:4]	[3:0]	16	16

- 설정할 수 있는 부분이 한가지 더 있다.
  - [7:4] 비트 부분을 모두 x로 사용하는 것
  - Group priority만으로 모두 사용하고, Sub priority는 없도록 만든다.

Priority group vs. Sub priority

- pre-emption priority group을 group priority로서 사용
  - **Pre-emption**이 가능하다는 것. 즉, 어떤 인터럽트가 동작 중일 때 그보다 높은 group priority를 가진 (숫자는 낮은 값) 인터럽트가 발생하게 되면 이전 인터럽트의 동작은 멈추고 새로 발생한 높은 group priority의 인터럽트가 동작하게 되는 것
- Sub priority는 Pre-emption 불가능
  - 새로 생긴 인터럽트가 같은 group priority를 가졌다면 지금 실행중인 인터럽트는 동작을 멈추지 않는다. Sub priority는 Pre-emption과는 무관하다.
- Sub priority 용도
  - 만약 높은 priority를 가진 인터럽트가 동작 중인데 새로 2개의 인터럽트가 새로 발생했다고 가정. 새로 발생한 2개의 인터럽트는 같은 group priority를 가졌지만 Sub priority는 다르다.
  - 이 경우 지금 수행 중이던 높은 priority를 가진 인터럽트가 동작을 끝내고 대기 중이던 **2**개의 인터럽트를 처리하려고 하는 순간 **Sub priority**가 작동하게 된다. 즉, 높은 Sub priority를 가지는 (낮은 숫자 값을 가지는) 인터럽트가 먼저 수행된다.

Priority Test 1 - Priority Group

- 2개의 키에 서로 다른 Priority Group을 설정
- 각각의 interrupt handler에서는 무한히 어떤 동작을 반복하도록 설정. 다른 색깔의 LED를 무한히 점등하는 작업
- Priority Group이 다르기 때문에 어떤 키가 눌린 상태에서 더 높은 Group priority를 가진 인터럽트가 발생하지 않으면 계속 자신의 무한한 interrupt handler를 동작하고 있을 것이다.
- 2개의 키에 서로 다른 Group priority를 지정할 것이기 때문에 아마도 키가 눌리는 순서에 따라 다른 현상이 나타날 것이라고 예상할 수 있다.
  - 만약 Key0에 더 높은 Group priority를 지정하였을 경우, Key0를 먼저 누르면 아무리 Key1을 눌러도 반응이 없을 것이고, Key1을 먼저 누르게 되면, Key1에 해당하는 LED가 점등 중이라도, 더 높은 Group priority를 가진 Key0가 눌리게 되면 Key1의 LED는 그 상태 그대로 있고, Key0의 LED가 점등되는 상황이 발생할 것이다.

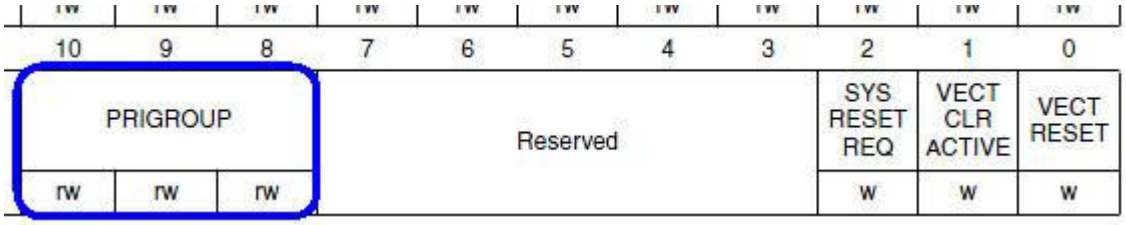
Priority Test 1 소스 분석 (1)

```
/* Configure one bit for preemption priority */
NVIC_PriorityGroupConfig(NVIC_PriorityGroup_1);

/* Enable the EXTI0 Interrupt */
NVIC_InitStructure.NVIC_IRQChannel = EXTI0_IRQn;
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStructure);

/* Enable the EXTI1 Interrupt */
NVIC_InitStructure.NVIC_IRQChannel = EXTI1_IRQn;
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 1;
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStructure);
```

Priority Test 1 소스 분석 (2)

```
#define NVIC_PriorityGroup_0         ((uint32_t)0x700) /*!< 0 bits for pre-emption priority
                                                            4 bits for subpriority */
#define NVIC_PriorityGroup_1         ((uint32_t)0x600) /*!< 1 bits for pre-emption priority
                                                            3 bits for subpriority */
#define NVIC_PriorityGroup_2         ((uint32_t)0x500) /*!< 2 bits for pre-emption priority
                                                            2 bits for subpriority */
#define NVIC_PriorityGroup_3         ((uint32_t)0x400) /*!< 3 bits for pre-emption priority
                                                            1 bits for subpriority */
#define NVIC_PriorityGroup_4         ((uint32_t)0x300) /*!< 4 bits for pre-emption priority
                                                            0 bits for subpriority */
```

- Priority Group을 설정하는 값은 Priority Group을 아예 사용하지 않는 NVIC_PriorityGroup_0부터 모든 비트를 Priority Group으로만 사용하는 NVIC_PriorityGroup_4까지 모두 5가지 이다.



- SCB_AIRCR 레지스터의 PRIGROUP 비트 부분을 설정하는데 사용

Priority Test 1 소스 분석 (3)

```
void EXTI1_IRQHandler(void) {
    if(EXTI_GetITStatus(GPIO_EXTI_Line_KEY2) != RESET) {
        EXTI_ClearITPendingBit(GPIO_EXTI_Line_KEY2);
        volatile int i;
        while(1) {
            for(i = 1000000; i > 0; i --);
            LED_Toggle_Blue();
        }
    }
}
```

```
void EXTI0_IRQHandler(void) {
    if(EXTI_GetITStatus(GPIO_EXTI_Line_KEY1) != RESET) {
        EXTI_ClearITPendingBit(GPIO_EXTI_Line_KEY1);
        volatile int i;
        while(1) {
            for(i = 1000000; i > 0; i --);
            LED_Toggle_Red();
        }
    }
}
```

Priority Test 2 - Sub Priority

- 이 시험을 위해서는 하나의 인터럽트 소스가 더 필요하다.
- 다른 하나의 인터럽트 소스로는 UART1을 사용한다.
- UART1으로 입력을 받게 되면 인터럽트가 발생하도록 만든 이후에 키를 동작 시켜서 Sub Priority에 대한 시험을 진행한다.

- UART1의 인터럽트 처리는 일정 시간이 지난 후에 끝나도록 한다.
- 그리고 그 인터럽트 처리 시간 동안에 두 키를 순서대로 눌러 본다.
- 물론 두 키의 인터럽트는 Group priority는 동일하게 하면서 Sub Priority를 다르게 설정해 놓는다.
- 두 키가 어떤 순서로 눌리더라도 Sub Priority가 높은 것이 먼저 실행될 것으로 예상할 수 있다.

Priority Test 2 소스 분석 (1)

```
/* Enable USART1 Receive and Transmit interrupts */
USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);
```

- USART1_Init() 함수에서 UART 인터럽트 처리를 위해서 위와 같은 코드가 초기화 과정에서 수행되도록 한다.
- UART 쪽 RX에 대해서 인터럽트가 발생할 수 있도록 만드는 것

Priority Test 2 소스 분석 (2)

```
/* Enable the USART1 Interrupt */
NVIC_InitStructure.NVIC_IRQChannel = USART1_IRQn;
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStructure);

/* Enable the EXTI0 Interrupt */
NVIC_InitStructure.NVIC_IRQChannel = EXTI0_IRQn;
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 1;
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStructure);

/* Enable the EXTI1 Interrupt */
NVIC_InitStructure.NVIC_IRQChannel = EXTI1_IRQn;
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 1;
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStructure);
```

Priority Test 2 소스 분석 (3)

```
void USART1_IRQHandler(void) {
    if(USART_GetITStatus(USART1, USART_IT_RXNE) != RESET) {
        USART_ClearITPendingBit(USART1, USART_IT_RXNE);
        int k;
        volatile int i;
        for(k = 0; k <10; k ++) {
            for(volatile int i = 1000000; i > 0; i --);
            LED_Toggle_Yellow();
}}}
```

- UART handler는 노란색 LED 점등, 10회만 반복
- 먼저 터미널에서 엔터를 치면 위 USART1_IRQHandler가 수행 되어서 노란색 LED가 점등
- 노란색 LED가 점등되는 동안 2개의 키를 순서대로 누른다.
- WKUP/USER 순으로 혹은, USER/WKUP 순으로 누른다.
- 어떠한 순서로 눌러도 붉은색 **LED**만 점등되는 상태가 된다.

Priority Test 3 - 동작 중 Priority 변경

- 인터럽트의 **Priority**는 고정된 것일까? 동작 중에도 변경이 가능한가?
- NVIC_PriorityGroup_1으로 설정하고, EXTI0는 Preemption Priority를 0으로 가지고 있고, EXTI1은 Preemption Priority를 1로 가진다

```
/* Configure one bit for preemption priority */
NVIC_PriorityGroupConfig(NVIC_PriorityGroup_1);

/* Enable the EXTI0 Interrupt */
NVIC_InitStructure.NVIC_IRQChannel = EXTI0_IRQn;
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority
                    = 0;
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStructure);

/* Enable the EXTI1 Interrupt */
NVIC_InitStructure.NVIC_IRQChannel = EXTI1_IRQn;
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority
                    = 1;
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStructure);
```

```
void NVIC_Configuration_ChangeDuringWorking(void) {
    NVIC_InitTypeDef NVIC_InitStructure;

    /* Enable the EXTI0 Interrupt */
    NVIC_InitStructure.NVIC_IRQChannel = EXTI0_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority
                        = 1;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);

    /* Enable the EXTI1 Interrupt */
    NVIC_InitStructure.NVIC_IRQChannel = EXTI1_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority
                        = 0;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
}
```

Priority Test 3 소스 분석

```
int main(void) {
..... ..... ..... ..... ..... .....
        printf("5> Interrupt Priority Change Test₩n");
..... ..... ..... ..... ..... .....
        case '5':
            NVIC_Configuration_ChangeDuringWorking();
            break;
        }
..... ..... ..... ..... ..... .....
}
```

- main 함수에 5번을 눌렀을 때 이것이 동작하도록 설정
- 최초에는 붉은색 LED 부분이 동작하는 것이 priority가 높았다가, 5번 메뉴로 Priority 변경을 실행한 이후에는 우선순위가 변경
- 프로그램의 동작 중에 얼마든지 Priority를 변경할 수 있는 것을 확인할 수 있다.