Hw4 배점 기준

1. 교재 2.6.3의 내용을 이해하여, late arrival이 발생하도록 코드를 작성하고, 실제 late arrival이 발 생함을 확인하고자 한다.
2. 전체 코드 (main.c)를 답안에 포함시키고, 어떤 원리에 의해서 late arrival이 발생하였음을 보 일수 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) .

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

1. Systick의 우선순위를 EXTI0 우선순위보다 높게 설정하고, (line23~24 코드가 포함되었는가 1점, 그룹 우선순위가 높아야 함)
2. line 30 ~ line 32의 코드를 이용해서 일정 시간 후에 SysTick interrupt가 발생하도록 동작 시킨다. 그 다음 EXTI0 interrupt를 발생시키면, **EXTI0 interrupt에 대한 stacking 하는 동안 systick interrupt가 발생**하도록 할 수가 있을 것이다. (SysTick->LOAD의 설정 값이 중요하다.) (코드와 설명 8점)

load(적절한 값 설정 중요 19~30 : b번 범위 안에 있을 것) 2점

코드에서 어떻게 late arrival이 발생하는가 설명 2점 (systick의 발생 시점을 조절함)

val 2점

ctrl = 0x07 (pending 이전에 나올 것) 1점

pending(enable 다음에 나올 것) 1점

1. CONTROL register 값을 ‘0’으로 만드는 코드는 라인 28에 있는 \_\_set\_control(0) 이란 코드이다. (코드가 포함되었는가 1점)

((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.

1. SysTick->LOAD에 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 (범위 5점) 0x13 ~ 0x1E

구간의 크기가 같은 경우 3점 – 길이:11

범위의 시작 끝이 동일하면 2점 (만약 코드상에 val이 다르면(없으면) 1점)

EXTI0 ISR과 SysTick ISR: SysTick ISR BP에서 먼저 중단됨.

* 이때 SP=0x20000628
* 이때 아래에서 보는 바와 같이 SysTick interrupt는 active 상태이고, EXT!0는 pending 상태 임. 즉 EXTI0 interrupt가 pending된 다음, systick interrupt 발생한 상태이므로 late arrival 이 발생한 상태임. (설명 3점, 캡처 2점)

(구간의 크기가 동일한 경우 솔루션과 동일한 설명이 아니어도 설명이 타당하면 설명 점수만 3점, 캡처는 점수 없음 // 솔루션과 다른 방식으로 접근한 경우 구간이 틀리면 설명도 틀린 것으로 간주 // 캡처의 경우 글로 EPA에 대해 설명해도 인정됨)

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

Stack 내용도 다음과 같다. 다음 2가지 형태가 존재한다. (LOAD 값에 따라서) – 여기 내용은 옵션 사항

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

텍스트, 폰트, 번호, 스크린샷이(가) 표시된 사진

자동 생성된 설명

1. 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

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

1. 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.

* 때 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 상 태이므로 late arrival이 아님. (설명 3점, 캡처 2점)

(구간의 크기가 동일한 경우 솔루션과 동일한 설명이 아니어도 설명이 타당하면 설명 점수만 3점, 캡처는 점수 없음, 솔루션과 다른 방식으로 접근한 경우 구간이 틀리면 설명도 틀린 것으로 간주)

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

Stack 내용도 다음과 같다. – 여기 내용은 옵션 사항



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.

텍스트, 폰트, 번호, 스크린샷이(가) 표시된 사진

자동 생성된 설명

1. (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. (설명 3점, 캡처 2점)

(구간의 크기가 동일한 경우 솔루션과 동일한 설명이 아니어도 설명이 타당하면 설명 점수만 3점, 캡처는 점수 없음, 솔루션과 다른 방식으로 접근한 경우 구간이 틀리면 설명도 틀린 것으로 간주)

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

Stack 내용도 다음과 같다. – 여기 내용은 옵션 사항

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

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

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명