Setting Interrupt

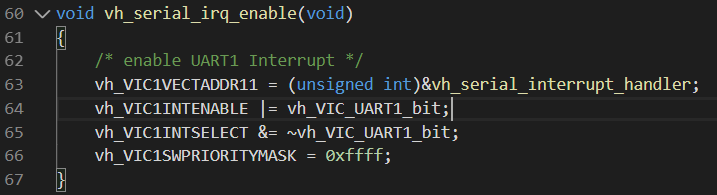
컴퓨터 소프트웨어학과

강정구

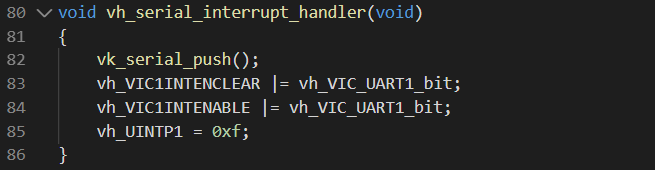
1. HW interrupt

VIC 는 vector 방식의 interrupt 레지스터 세트를 지원하는 방식으로 interrupt source 마다 레지스터 하나씩을 할당하여 해당하는 interrupt routine 의 주소값을 저장하고 있다. 그래서 만약 interrupt 가 발생하면 controller 은 해당하는 register 의 address 로 자동으로 갈 수 있도록 해준다. 이런 방식으로 하면 interrupt 지연 시간을 줄여 빠르게 ISR 을 실행할 수 있다는 장점을 가지고 있다.

S5PC100 은 3개의 VIC 를 가지고 있는데 우리는 그 중 UART1 을 구현해 볼 것이다.

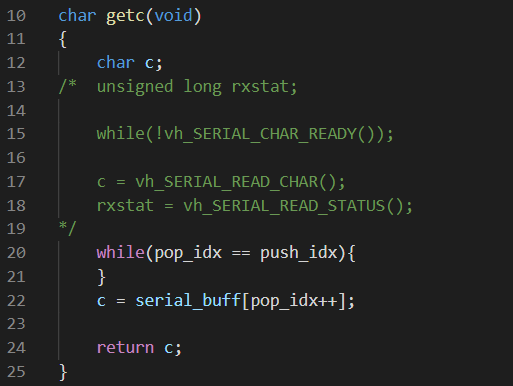
**serial.c / vh\_serial\_irq\_enable()**

위의 코드는 main 함수에서 set\_interrupt 를 실행시키면 set\_interrupt 가 실행시키는 함수이다. 이는 UART1 Interrupt 를 활성화시키는 함수로 VIC1VECTADDR11 레지스터에 ISR 주소를 저장하고 VIC1INTSELECT 레지스터에서 UART1 인터럽트를 IRQ 로 설정, VIC1SWPRIORITYMASK 레지스터를 모두 1로 mask 하는 과정을 담고 있다.

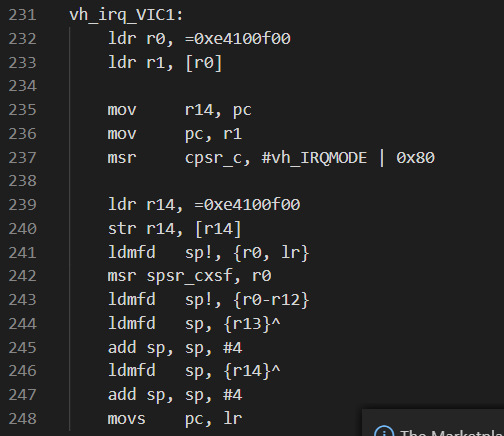
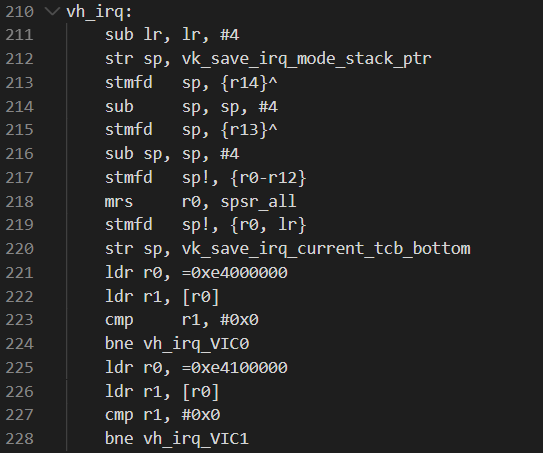
**Serial.c / vh\_serial\_interrupt\_handler()**

이 함수는 키보드의 입력을 ㅂ다아 버퍼에 저장하는 역할을 한다. 실행순서는 URXH 레지스터에 수신된 키보드 문자 데이터를 버퍼에 저장하고 VICINTENCLEAR 레지스터에서는 UART1 인터럽트를 비활성화, VIC1INTENABLE 에서는 인터럽트를 활성화 시키고 UINTP1 레지스터에서 UART1 인터럽트를 다시 설정하는 역할을 수행한다.

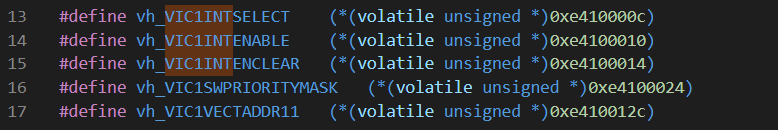
**Serial.c / getc()**



**startup.S / vh\_irq startup.S / vh\_irq\_VIC1**



왼쪽의 vh\_irq 는 interrupt 진입 루틴으로 이전의 레지스터들인 r14, r13, r0~r12 들을 stack 에다가 넣어주고 SPSR 과 link register 까지 넣어준다. 그리고 VICIRQSTATUS 를 확인하여 어디에서 인터럽트를 발생했는지 확인하여 pc 에 VICADDRESS 값을 저장하게 된다. 여기에서는 VIC0 과 VIC1 의 STATUS 만 확인하여 넘어가게ㅔ 된다. 그리고 오른쪽에는 VIC1 의 INT 진입과 복귀 루틴이 나와있다. VIC0ADDR 에 저장된 인터럽트 핸들러 주소로 점프하고 진입할때와 마찬가지로 stack 에서 값들을 복구하고 return 하게된다.

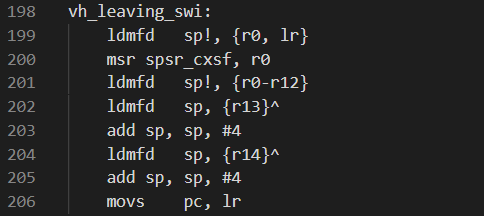
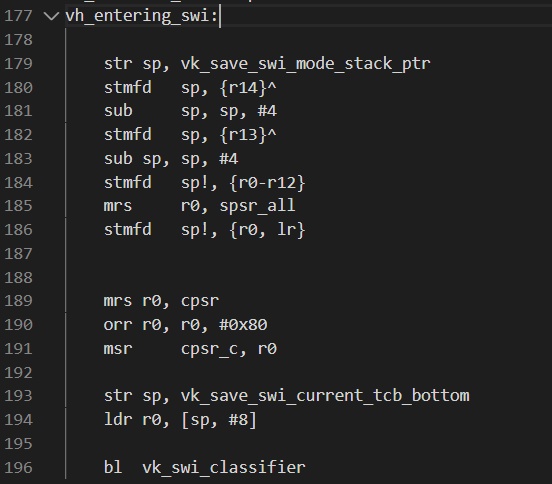
**Hal.h / irq\_registers**

위에 쓰인 레지스터들의 값을 저장해준 모습이다.

2. SW interrupt

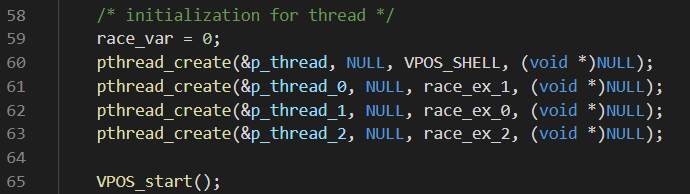
sw interrupt 는 프로그램의 실행을 그 프로그램에 포함시킨 명령어로 중단시키는 것으로 hw 상에서 자동적으로 처리해 준 것과 다르다는 것을 알 수 있다. 이는 커널 함수를 호출하기 위해 privileged mode 에 진입하고자 할때 사용한다. Software Interrupt 는 decode 단계에서 swi 명령어를 인식하여 발생한다. SWI 란 프로세스 모드를 privileged mode 로 변경시켜 운영체제가 하는 일이 더 높은 권한을 가질 수 있게 해준다. 이는 일단 VPOS scheduler 에서 사용되는 것을 볼 수 있다.

**startup.S / vh\_entering\_swi startup.S / vh\_leaving\_swi**



vh\_entering\_swi 에서는 swi 진입루틴을 나타낸다. hw IRQ mode 와 마찬가지로 stack 에다가 원래 쓰던 register 값들을 저장하고 SPSR, lr 을 stack 에 저장한다. 이 때 좀 다른 점이 IRQ exception 을 비활성화 해야하는데 r0 에 cpsr 값을 넣어주고 인터럽트 마스크 비트를 0x80 으로 수정하고 cpsr\_c 에다가 넣어준다. 그 다음 r0 값을 다시 stack 에 저장한다음 swi\_classifier 함수로 branch 한다.

Vh\_leaving\_swi 에서는 swi 복귀 루틴으로 스택에 저장했던 모든 레지스터를 복원하고 원래의 루틴으로 복귀하는 과정이 나타나 있다.

**kernel\_start.c / pthread\_create**

3. 실행화면

