인터럽트

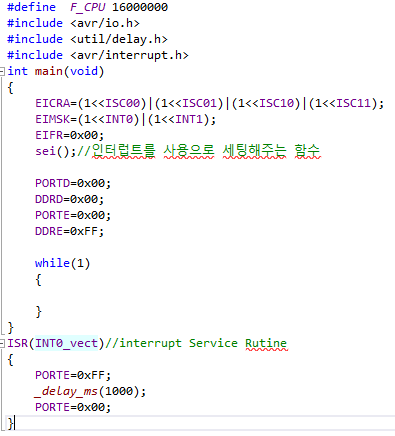
이번에는 인터럽트라는 내용에 대해서 다루어 본다. 인터럽트는 특정한 신호에 따라서 현재 작업을 중단하고 정해진 내용을 수행하게 하는 기능이다. 이기능은 다중작업을 위해서 생긴 기능이다.

저번 장 들에서 레지스터를 다룰 때 보면 항상 인터럽트 플래그 레지스터 라는게 있었다. 이 레지스터는 특정 상황에서 어떠한 상황을 알려주기 위한 레지스터들인 경우가 많았는데 이번장에서는 이 부분에 대해서 다룬다.

인터럽트는 크게 2종류로 나눌수있다.

* 폴링=코드에서 직접 인터럽트를 검사해서 발동된 인터럽트를 확인하고 이를 처리함. 우리가 지금까지 인터럽트를 확인하는 코드를 짜준 건 이쪽에 해당된다 처리 순서에 한계가 있고 무었보다 성능을 잡아먹는다.
* 벡터=하드웨어적으로 인터럽트가 발생하면 이에 대한 벡터를 CPU에 전송하고 이 벡터에 있는 코드를 실행시키는 인터럽트. 하드웨어적으로 이에 대한 회로가 있어야 되지만 다른 동작을 무시하고 발동 시킬수 있기 때문에 처리가 빠르다.

이번에 다루는 내용은 대부분 인터럽트 벡터를 사용해서 다루는 내용이다.

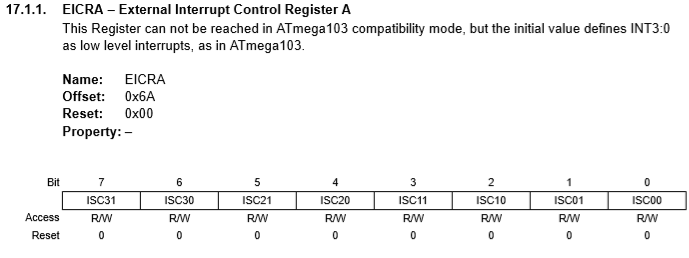
이 코드는 특이하게 while문 안에 아무런 코드가 없어서 아무것도 안하는 것처럼 보일 수 있다. 하지만 실제로는 이 코드는 PORTD에 있는 외부 인터럽트 핀으로부터 입력을 받아서 PORTE를 제어하는 코드이다.

여기서 특이점은 while을 이용해서 PORTD를 읽는 식이 아니라 인터럽트 함수를 사용했다는 것이다. 이제부터 이것을 어떻게 사용하는지에 대해서 이야기할 예정이다.

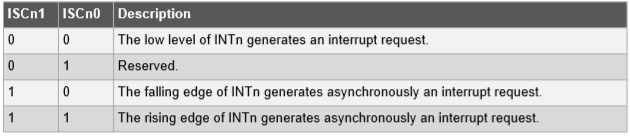
일단 물리적인 이야기를 하고 넘어가야 된다. 하드웨어 인터럽트는 물리적으로 구성되어 있는 기능중 하나이기 때문에 사용을 위한 핀이 물리적으로 정해져 있다. 핀아웃을 확인해보면 외부 인터럽트 핀의 위치를 확인해 볼수 있는데 우리가 사용할 인터럽트 핀인 INT0,1은 각각 PORTD의 0과 1번핀에 할당되어 있는 것을 알수 있다. 따라서 외부 인터럽트 0과 1을 사용하기 위해서는 이핀들을 사용해야 된다

코드에 나와있는 것 중에서 avr/interrupt.h가 눈에 띌텐데 이 헤더는 인터럽트 관련 기능을 모아놓은 헤더로 인터럽트를 사용하기 위해서는 꼭 추가시켜야된다.

다음은 외부 인터럽트 관련 레지스터에 대해서 다룬다.

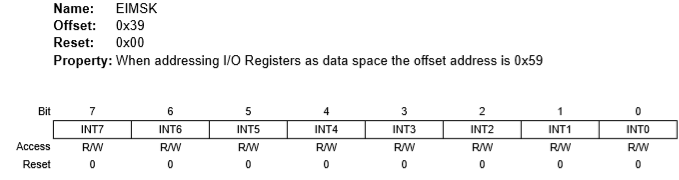


EICR레지스터는 A와 B2개가 있는데 여기에서는 A만 사용하기 때문에 A만 다룬다. 여기에는 핀 1개를 세팅하기 위한 비트가 2개씩 있는데 각각 이런 식으로 작동한다.

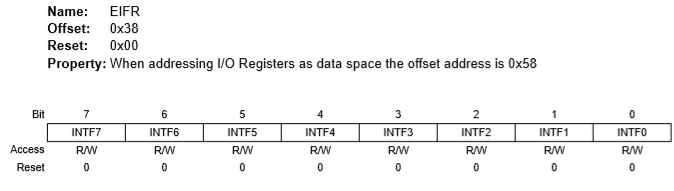


* 0/0=높은 신호가 아닌 낮은 신호에서 동작하는 인터럽트. 주로 항상 켜져있는 신호가 꺼꺼질 때 확인하는 용도로 쓰인다.
* 1/0=신호가 들어왔을 때 신호파형이 하강하는 과정에서 인터럽트가 동작한다. 버튼을 누를때 버튼을 누른뒤 다시 올라갈 때 동작한다고 생각하면 쉽다
* 1/1=신호가 들어왔을 때 신호파형이 상승할 때 인터럽트가 동작한다. 버튼을 눌렀을 때 바로 동작한다고 이해하면 쉽다.

코드 에서는 0과 1번 핀 세팅을 전부 상승 트리거로 잡아 놨다. 따라서 신호가 상승할 때 동작한다



다음은 EIMSK레지스터이다. 이 레지스터는 1로 작성해 놓은 비트의 핀의 인터럽트를 허용한다. 따라서 이 코드에서는 INT0과 INT1의 코드를 활성화 시켜놨다



마지막으로는 EIFR레지스터인데 이 레지스터는 외부 인터럽트 플래그들 가지고있는 레지스터로 실제로 외부 인터럽트가 동작했을 때 작동하는 레지스터이다. 코드에서는 맨 처음에 초기화만 해놓고 사용하지 않는데 그 이유는 실제로 인터럽트가 발생할 경우 EIFR레지스터에 올라온 플래그는 인터럽트 루틴을 벗어나면 자동으로 헤제된다.

코드를 보면 이 레지스터들을 사용한 직후 sei()라는 함수를 사용했는데 이는 인터럽트 헤더에 작성된 함수로 시스템 레지스터를 조작해서 하드웨어 인터럽트 벡터를 사용하는 것을 허용하게 세팅하는 함수이다. 하드웨어 인터럽트를 사용하려면 꼭 이 함수를 쓰던가 개별적으로 SERG라는 레지스터를 세팅해줘야된다.

이후 PORTD는 입력으로 초기화 시켜놓고 PORTE는 출력으로 초기화 시킨다.

코드에서는 하는일이 없이 계속 무한루프만 돈다. 하지만 인터럽트가 걸리게 될 경우 인터럽트를 발생시킨 하드웨어는 자신의 인터럽트 벡터를 CPU에 전송하고 CPU는 현재 진행중인 작업을 일단 저장하고 인터럽트 벡터가 가르키고 있는 인터럽트 서비스 루틴을 찼아서 해당되는 루틴에 있는 코드를 수행한다. 여기에서는 INT0\_vect가 가리키고 있는 ISR함수의 PORTE를 조작하는 내용의 코드를 실행시킨 뒤에 코드가 다 실행되었다면 다시 원래의 코드로 돌아온다. 이경우에는 무한루프를 돌게된다.

여기에서는 외부 인터럽트를 사용하는 방법을 다루었는데 우리가 지금까지 다룬 기능들이 가진 인터럽트 플래그를 볼수 있었듯이 이 기능은 여기에만 국한되지 않고 거의 마이크로 컨트롤러가 가진 대부분의 기능에 적용할수 있다. ATmega128A가 지원한 인터럽트 벡터는 총 35개로 물리적인 제약사항이 발생하지 않는다면 이들 전부를 사용해 볼수 있다. 다른 기능의 인터럽트를 사용하고 싶다면 인터럽트 벡터 표를 참고해서 자신이 사용하고 싶은 인터럽트 벡터를 ISR함수에 넣어서 사용하면 된다.

하드웨어 인터럽트는 다중 작업을 처리하려면 꼭 배워야 되는 기술이다. 폴링으로만 작업을 처리하려면 정확한 시간에 정확한 작업을 하는 것이 불가능한 경우가 많은 만큼 이를 해결하려면 꼭 하드웨어 인터럽트 사용에 숙달될 필요가 있다.