AVR 이론 : 인터럽트 (Interrupt)

→ 주작업을 하던중 방해하는 원인으로
  갔다 복귀하여 원래의 주작업을 수행하는 기법

인터럽트에서 가장 중요한 요소 3가지
1. 인터럽트 소스
2. 인터럽트 벡터
3. 인터럽트 우선순위

인터럽트 종류
1. 발생원인에 따라 인터럽트 [하드웨어 [외부인터럽트 / 내부인터럽트] / 소프트웨어]
2. 차단여부에 따라 { 차단가능[무시](INT), 차단불가능(NMI) }
3. 인터럽트 확인방법에 따라 [Polling (하드웨어구조 단순 / 장치의수 영향) / vector (하드웨어구조 복잡 / 장치의수영향X)]

AVR → [전체인터럽트 차단(DI) / 허용(EI) / 개별인터럽트 차단/허용]

EI 명령은 SEI
DI 명령은 CLI or SREG 레지스터의
1 비트를 세트/클리어하도 동일한 결과를 가져온다.

C언어에서는 sei(), cli()로 사용하시면 됩니다.

ATmega128의 리셋벡터와 인터럽트 벡터
1. 리셋 (1개)
2. 외부인터럽트 (8개)
3. 타이머/카운트 관련 인터럽트 (8개)
4. 각종 통신 관련 인터럽트 (8개)
5. ADC, EEPROM, 아날로그 비교기 (4개)

인터럽트 처리과정
1. 인터럽트 요청 신호 접수 → 해당장치의 인터럽트 플래그가 세트
2. 인터럽트 요청 허용 여부판단 → 동시에 발생한 인터럽트 우선순위 결정
3. 인터럽트 벡터 주소로 점프 → 해당주소로 점프하게 됩니다.
4. 복귀 정보 저장 → 동작중인 프로그램 PC값을 Stack에 저장합니다.
5. 인터럽트 서비스 루틴의 수행 → 서비스루틴으로 해당 프로그램을 수행한다
6. 주 프로그램으로 복귀 → RETI 명령을 만나 Stack of PC의 주프로그램으로 복귀.

AVR이론 : 타이머 / 카운터 개요

타이머를 동작하기 위해
1. 입력되는 클럭이 일정해야 한다.
2. 클럭의 주파수를 아는 것이 중요하다.

타이머 -> 일정주기로 신호를 준다.
카운터 -> 주기를 셈으로써 해당 기준치를 넘기면 (오버플로)발생한다.
-> 인터럽트가 발생되어 원하는 명령을 처리하는 방식이다.

타이머/카운터 인터럽트 처리
-> 타이머/카운터 인터럽트로 Overflow으로 사용하고,
1초를 만들기 위한 처리과정

step1
: 타이머가 일정한 신호이다.
: 신호를 주면, 카운터는 Overflow를 발생시킨다.
: 인터럽트가 발생하여 원하는 프로그램을 처리한다.

step2
: 주파수 -> 1초에 진동하는 횟수
: 주기 -> 진동한번/신호한번 1Hz가 들어오는 시간

타이머/카운터 인터럽트 레지스터
1. TIMSK
2. TCCR0A / TCCR0B
3. OCR0A / OCR0B
4. TIFR
5. etc..

ATmega 128 → 8개의 외부인터럽트 지원

1. EICRA
0 0 Low Level triggered
0 1 (reserved)
1 0 Falling Edge 비동기적으로 트리거
1 1 Rising Edge 비동기적으로 트리거

2. EICRB
0 0 Low Level 인터럽트 트리거
0 1 상승 or 하강에지 인터럽트 트리거
1 0 Falling Edge 인터럽트 비동기적 트리거
1 1 Rising Edge 인터럽트 비동기적 트리거

3. EIMSK INT 7 ~ 0
셋트를 하게 되면 "인터럽트" 허용
클리어를 하게 되면 "인터럽트 금지"

4. EIFR INT 7 ~ 0
인터럽트 신호가 입력되면 해당 인터럽트가 발생
인터럽트처리가 시작되면서 서브루틴으로
점프하게 되면 클리어 됩니다.