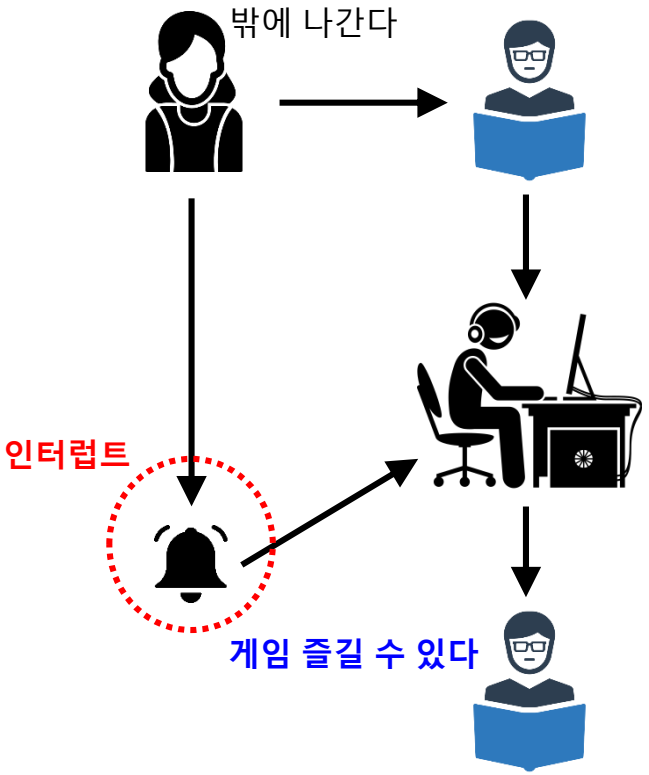
Lecture 06

외부 인터럽트

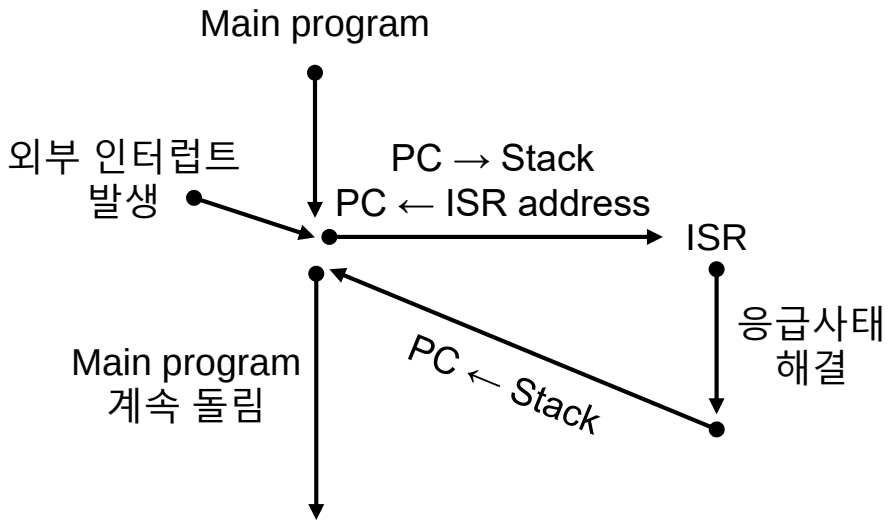
외부 인터럽트란?

- CPU의 즉각적인 처리를 필요로 하는 이벤트 알림



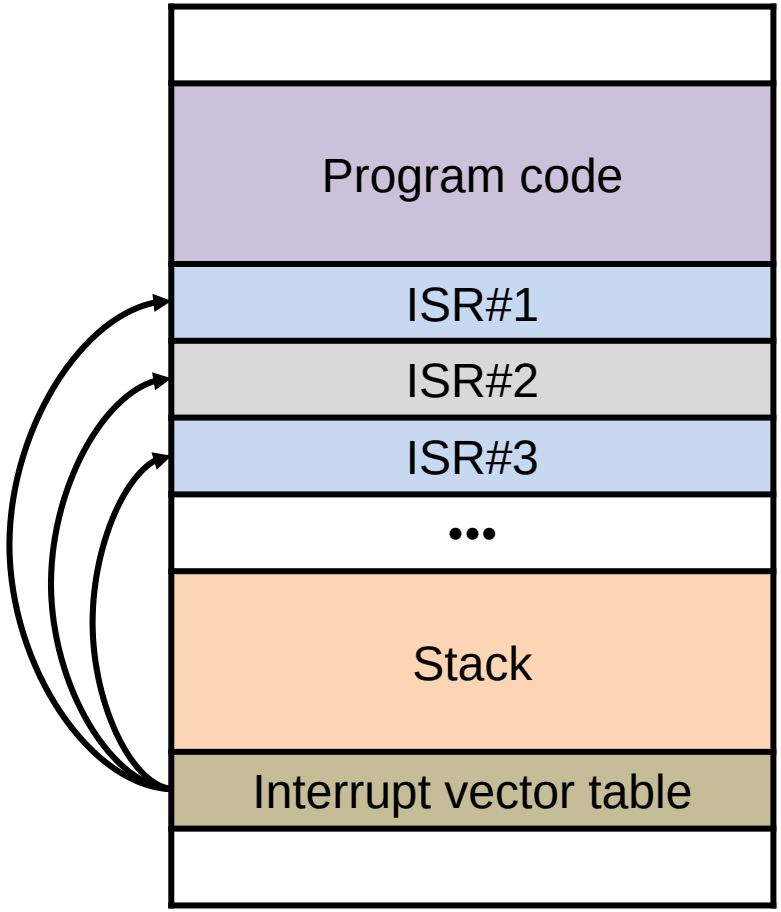
인터럽트 처리

- ISR(Interrupt Service Routine)
  - 인터럽트 발생 시 처리 해주는 코드
- 인터럽트 처리절차
  1. 해내고 있는 코드를 멈춘다
  2. PC값을 stack에 넣는다
  3. ISR주소값을 얻는다
  4. ISR로 건너뛴다
  5. ISR코드를 실행한다
  6. 다 처리하면 PC값을 되돌린다
  7. 이전실행위치로 되돌아간다



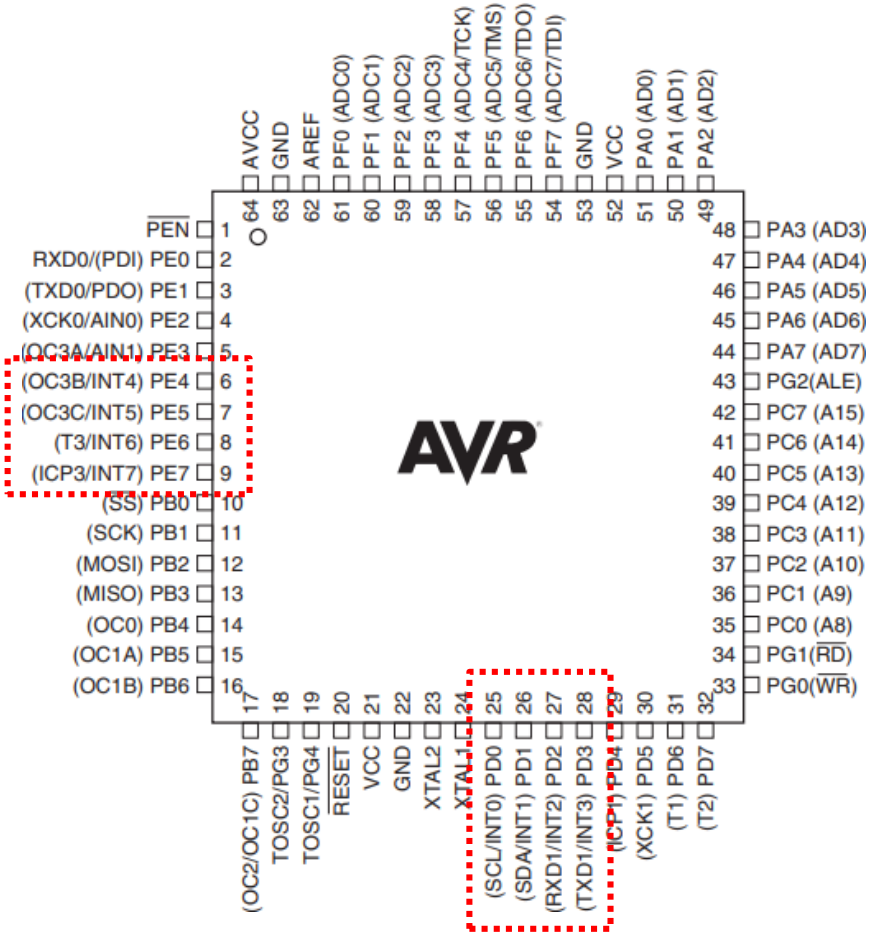
ATmega128 인터럽트 종류

- 외부 인터럽트
  - INT7~INT0 8개
- 내부 인터럽트
  1. Reset
  2. 타이머/카운터
  3. SPI
  4. UART
  5. I2C
  6. SPM
  7. ADC
  8. Analog comparator
  9. EEPROM



ATmega128 인터럽트 종류

- 외부 인터럽트
  - INT7~INT0 8개
    - INT0 : PD0
    - INT1 : PD1
    - INT2 : PD2
    - INT3 : PD3
    - INT4 : PE4
    - INT5 : PE5
    - INT6 : PE6
    - INT7 : PE7



관련 레지스터

- SREG : Status Register

Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

  - Bit 7 : I (Global Interrupt Enable)
    - 인터럽트 기능을 사용하기 위해 SREG의 7번 비트를 1로 설정해 야 함
    - SREG의 7번 비트가 0이면 어떤 인터럽트도 사용할 수 없게 됨
    - 인터럽트 발생 후 하드웨어는 SREG의 7번 비트를 0로 바꿔 줌 → 어떤 인터럽트를 처리하는 중 다른 인터럽트를 받지 않다는 의미
    - 인터럽트 처리 후 하드웨어는 SREG의 7번 비트를 다시 1로 바꿔 줌 → 인터럽트를 다시 받아 주는 의미

관련 레지스터

- EIMSK : External Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
	INT7	INT6	INT5	INT4	INT3	INT2	INT1	IINT0	EIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

  - Bits 7~0 : INT7~INT0 (External Interrupt Request Enable)
    - SREG의 7번 비트를 1로 설정하고 나서 개별적인 인터럽트를 사용하기 위해 EIMSK에서 해당 비트를 1로 설정해 야 함
    - 예: 인터럽트 6을 사용하려면 다음과 같이 설정해 야 함

```
SREG |= 0x80; // Global Interrupt Enable
EIMSK |= 0x40; // INT6 Enable
```

관련 레지스터

- EIFR : External Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
	INTF7	INTF6	INTF5	INTF4	INTF3	INTF2	INTF1	IINTF0	EIFR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

  - Bits 7~0 : INTF7~INTF0 (External Interrupt Flags)
    - SREG의 7번 비트와 EIMSK에 개별 인터럽트를 설정한 후 해당 인터럽트가 발생하면 EIFR의 해당 인터럽트 flag는 1로 됨
    - 해당 ISR 실행이 완료되면 EIFR의 해당 인터럽트 flag는 0로 됨
    - 또한 EIFR의 인터럽트 flag를 수동적으로 클리어하려면 해당 비트에 1를 써 넣어 야 함

관련 레지스터

- EICRA : External Interrupt Control Register A

Bit	7	6	5	4	3	2	1	0	
	ISC31	ISC30	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00	EICRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

  - Bits 7~0 : ISC31~ISC00 (External Interrupt 3~0 Sense Control)
    - SREG의 7번 비트와 EIMSK에 개별 인터럽트를 설정한 후 해당 인터럽트의 trigger 방식을 결정함

ISCn1	ISCn0	인터럽트 trigger 방식
0	0	INTn 핀의 low 레벨 신호 입력에 의해 인터럽트 trigger
0	1	reserved
1	0	INTn 핀의 하강에지(falling edge)에서 인터럽트 trigger
1	1	INTn 핀의 상승에지(rising edge)에서 인터럽트 trigger

관련 레지스터

- EICRB : External Interrupt Control Register B

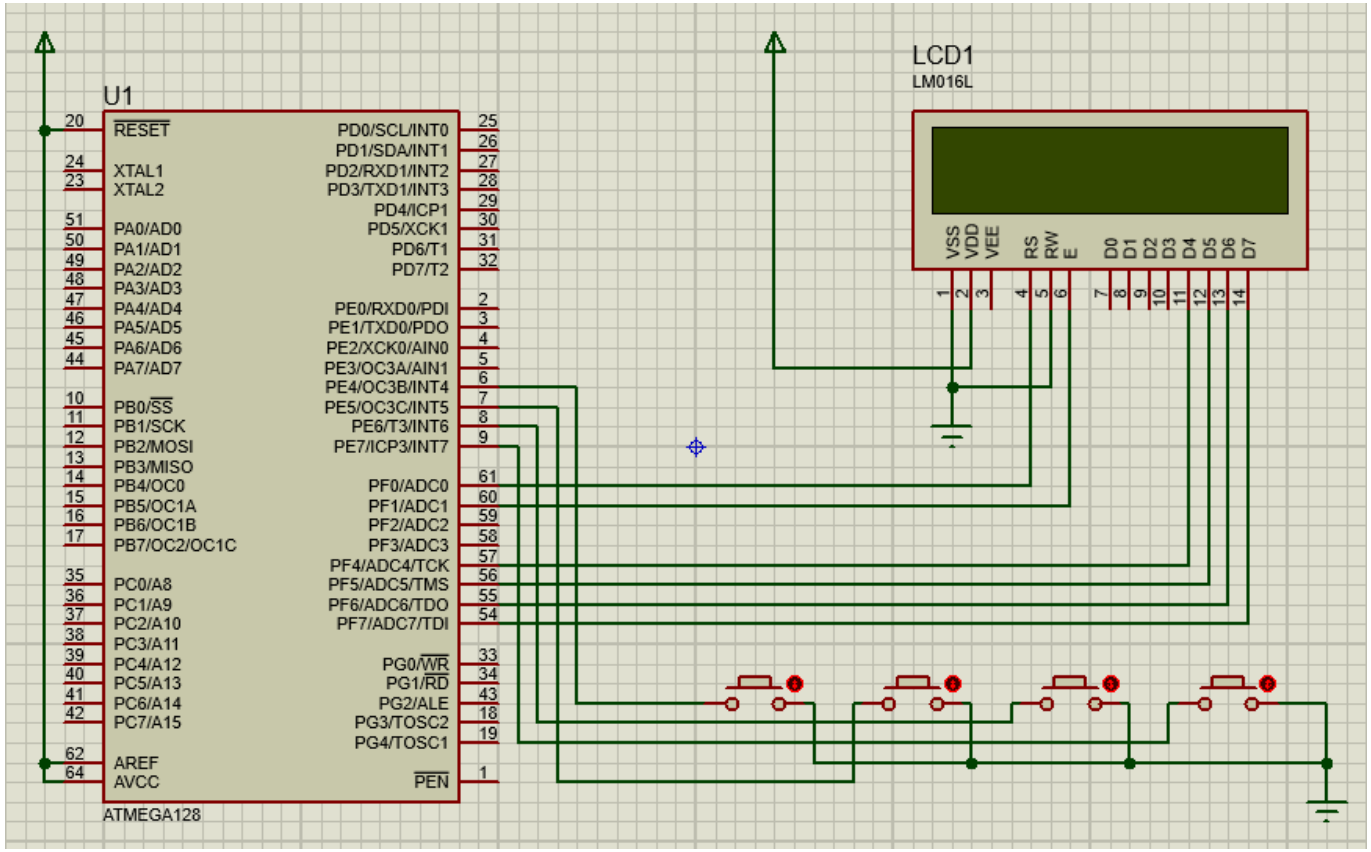
Bit	7	6	5	4	3	2	1	0	
	ISC71	ISC70	ISC61	ISC60	ISC51	ISC50	ISC41	ISC40	EICRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

  - Bits 7~0 : ISC71~ISC40 (External Interrupt 7~4 Sense Control)
    - SREG의 7번 비트와 EIMSK에 개별 인터럽트를 설정한 후 해당 인터럽트의 trigger 방식을 결정함

ISCn1	ISCn0	인터럽트 trigger 방식
0	0	INTn 핀의 low 레벨 신호 입력에 의해 인터럽트 trigger
0	1	INTn 핀의 논리값의 변화에 의해 인터럽트 trigger
1	0	INTn 핀의 하강에지(falling edge)에서 인터럽트 trigger
1	1	INTn 핀의 상승에지(rising edge)에서 인터럽트 trigger

실습1: 하강에지 인터럽트

- Proteus



실습1: 하강에지 인터럽트

- 포트 설정
  - 버튼 입력
    - 포트 E 입력 모드 설정: DDRE = 0x00
    - 포트 E pull-up 설정 : PORTE = 0xff
  - LCD 제어
    - 포트 F 출력 모드 설정: DDRF = 0xff
- 인터럽트 설정
  - Global interrupt enable : SREG |= 0x80
  - INT7~INT4 사용 : EIMSK = 0xf0
  - 하강에지 trigger 설정 : EICRB = 0xaa

실습1: 하강에지 인터럽트

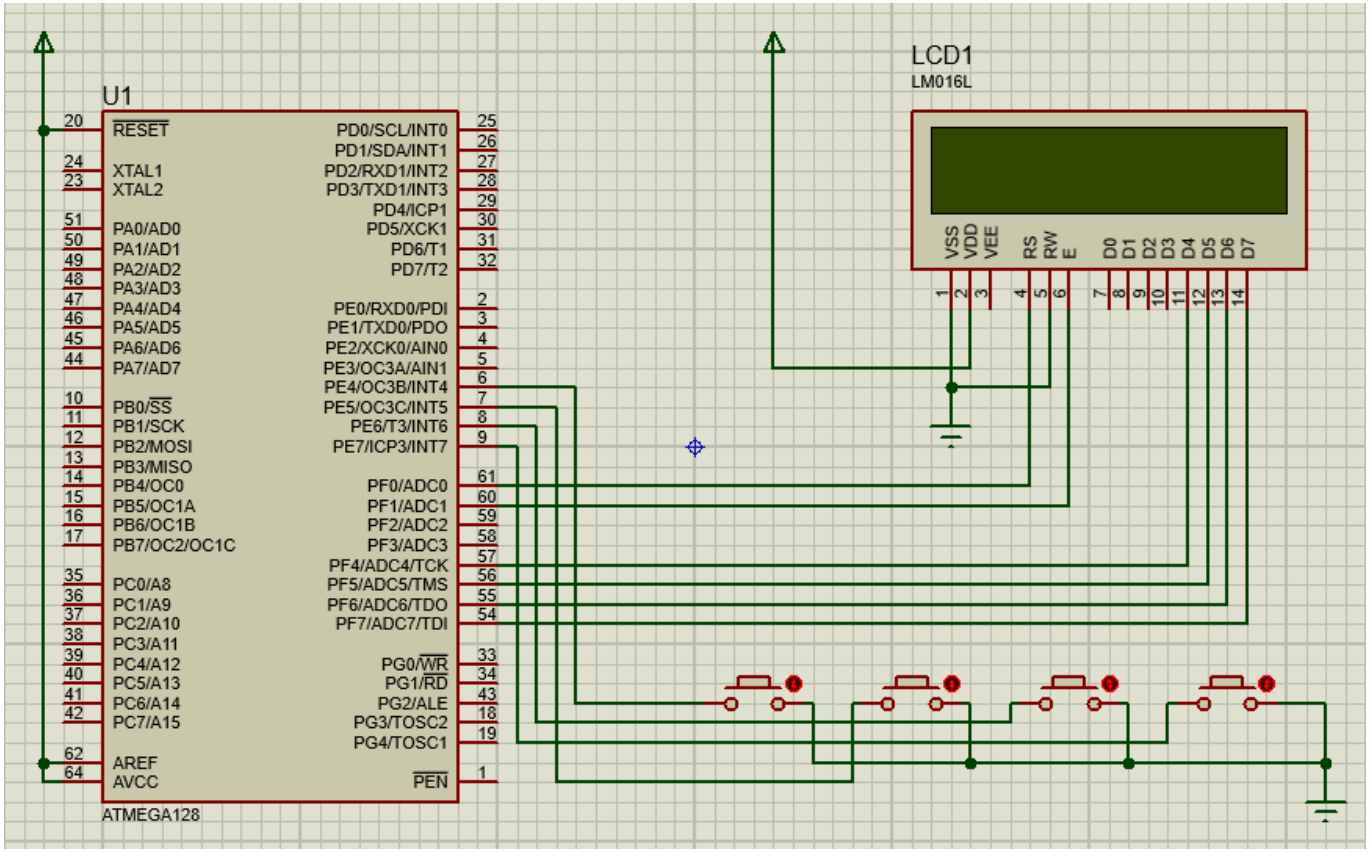
- 소스 코드

```c
#include <xc.h>
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include "lcd4.h"
#include "stdio.h"

volatile unsigned char interruptNo = 0;

ISR(INT4_vect) { interruptNo = 4; }
ISR(INT5_vect) { interruptNo = 5; }
ISR(INT6_vect) { interruptNo = 6; }
ISR(INT7_vect) { interruptNo = 7; }
```

```c
int main(void) {
    char msg[20];

    DDRF = 0xff;
    DDRE = 0x00;
    PORTE = 0xff;

    SREG |= 0x80;
    EIMSK = 0xf0;
    EICRB = 0xaa;

    init_lcd4();
    writeString_lcd4(0, 0, "Interrupt No: ");

    while(1) {
        if (interruptNo) {
            sprintf(msg, "%d checked", interruptNo);
            interruptNo = 0;
            writeString_lcd4(0, 1, msg);
        }
    }
}
```

실습2: 인터럽트 flag 체크

- Proteus



실습2: 인터럽트 flag 체크

- 포트 설정
  - 버튼 입력
    - 포트 E 입력 모드 설정: DDRE = 0x00
    - 포트 E pull-up 설정 : PORTE = 0xff
  - LCD 제어
    - 포트 F 출력 모드 설정: DDRF = 0xff
- 인터럽트 설정
  - ISR를 사용하지 않아서 global interrupt enable(SREG |= 0x80)을 안 함
  - INT7~INT4 사용 : EIMSK = 0xf0
  - 하강에지 trigger 설정 : EICRB = 0xaa

실습2: 인터럽트 flag 체크

- 소스 코드

```c
#include <xc.h>
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include "lcd4.h"
#include "stdio.h"

int main(void) {
   unsigned char interruptNo = 0;
   char msg[20];

   DDRF = 0xff;
   DDRE = 0x00;
   PORTE = 0xff;
   EIMSK = 0xf0;
   EICRB = 0xaa;
   init_lcd4();
   writeString_lcd4(0, 0, "Interrupt No: ");

   while(1) {
      interruptNo = EIFR;
      if (interruptNo) {
         switch (interruptNo) {
            case 0x10: interruptNo = 4; EIFR |= 0x10; break;
            case 0x20: interruptNo = 5; EIFR |= 0x20; break;
            case 0x40: interruptNo = 6; EIFR |= 0x40; break;
            case 0x80: interruptNo = 7; EIFR |= 0x80; break;
         }
         sprintf(msg, "%d checked", interruptNo);
         interruptNo = 0;
         writeString_lcd4(0, 1, msg);
      }
   }
}
```