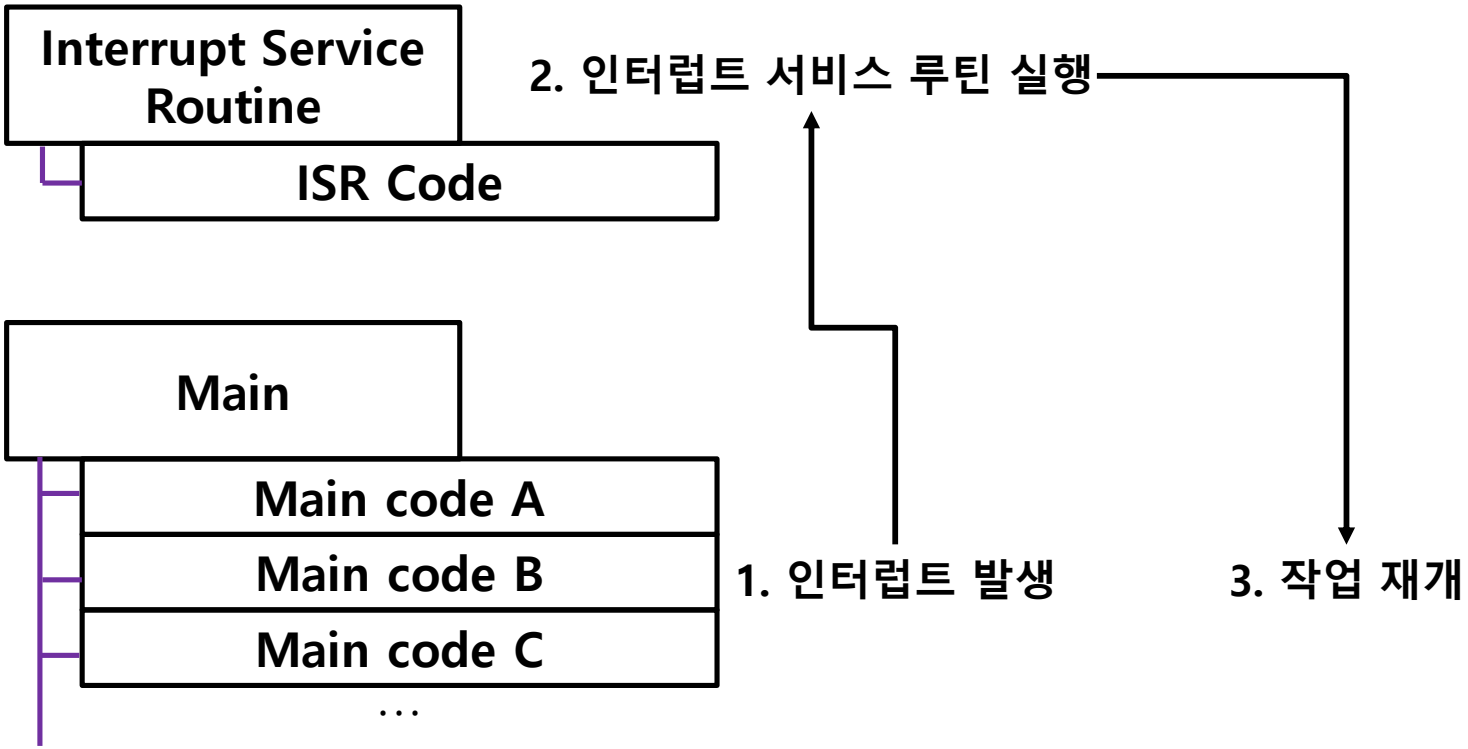
전자회로실험2

Lab5.
Interrupt

Interrupts


Interrupt Service Routine
ISR Code
2. 인터럽트 서비스 루틴 실행
Main
Main code A
Main code B
Main code C
...
1. 인터럽트 발생
3. 작업 재개

Interrupts

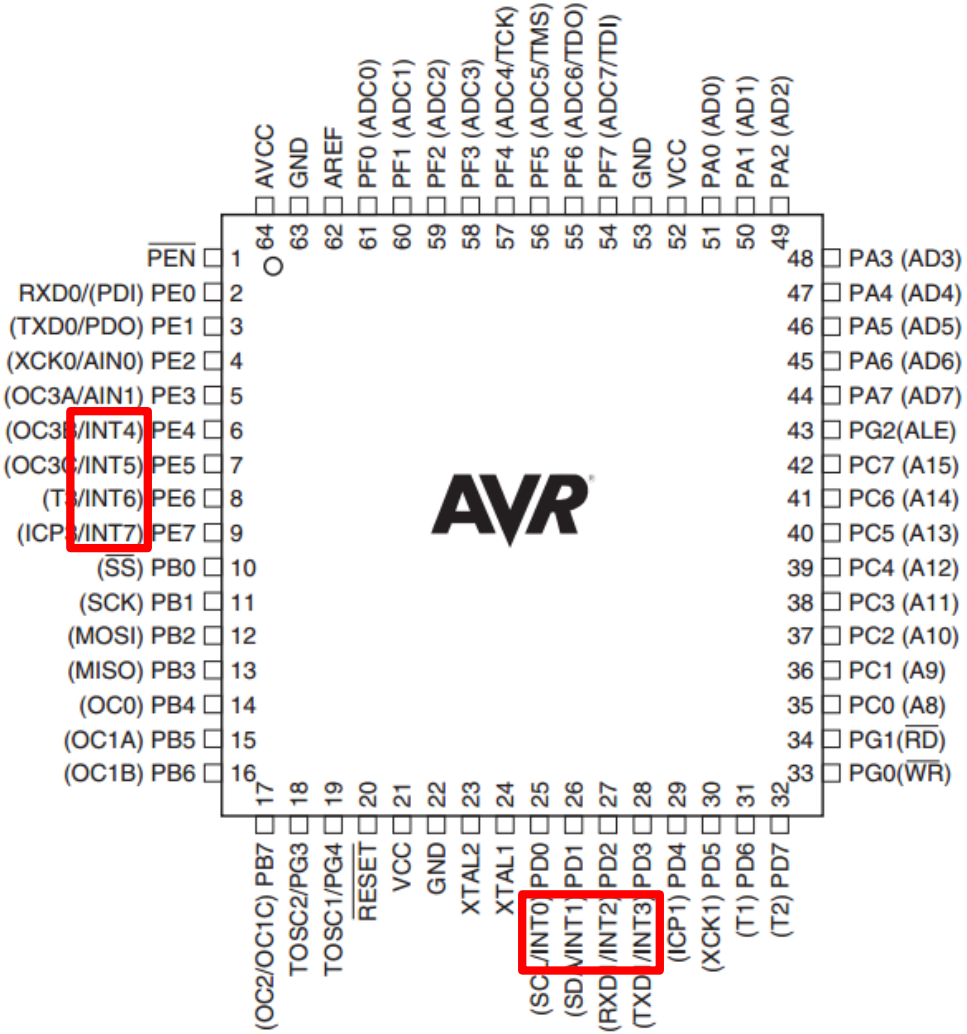
Interrupt Vector: 인터럽트 발생시 이동할 주소를 가지고 있음

Table 23. Reset and Interrupt Vectors

Vector No.	Program Address[2]	Source	Interrupt Definition
1	$0000[1]	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	$0002	INT0	External Interrupt Request 0
3	$0004	INT1	External Interrupt Request 1
4	$0006	INT2	External Interrupt Request 2
5	$0008	INT3	External Interrupt Request 3
6	$000A	INT4	External Interrupt Request 4
7	$000C	INT5	External Interrupt Request 5
8	$000E	INT6	External Interrupt Request 6
9	$0010	INT7	External Interrupt Request 7
10	$0012	TIMER2 COMP	Timer/Counter2 Compare Match
11	$0014	TIMER2 OVF	Timer/Counter2 Overflow
12	$0016	TIMER1 CAPT	Timer/Counter1 Capture Event
13	$0018	TIMER1 COMPA	Timer/Counter1 Compare Match A
14	$001A	TIMER1 COMPB	Timer/Counter1 Compare Match B
15	$001C	TIMER1 OVF	Timer/Counter1 Overflow
16	$001E	TIMER0 COMP	Timer/Counter0 Compare Match
17	$0020	TIMER0 OVF	Timer/Counter0 Overflow
18	$0022	SPI, STC	SPI Serial Transfer Complete
19	$0024	USART0, RX	USART0, Rx Complete
20	$0026	USART0, UDRE	USART0 Data Register Empty
21	$0028	USART0, TX	USART0, Tx Complete
22	$002A	ADC	ADC Conversion Complete
23	$002C	EE READY	EEPROM Ready
24	$002E	ANALOG COMP	Analog Comparator
25	$0030[3]	TIMER1 COMPC	Timer/Countre1 Compare Match C
26	$0032[3]	TIMER3 CAPT	Timer/Counter3 Capture Event
27	$0034[3]	TIMER3 COMPA	Timer/Counter3 Compare Match A
28	$0036[3]	TIMER3 COMPB	Timer/Counter3 Compare Match B
29	$0038[3]	TIMER3 COMPC	Timer/Counter3 Compare Match C
30	$003A[3]	TIMER3 OVF	Timer/Counter3 Overflow
31	$003C[3]	USART1, RX	USART1, Rx Complete
32	$003E[3]	USART1, UDRE	USART1 Data Register Empty
33	$0040[3]	USART1, TX	USART1, Tx Complete
34	$0042[3]	TWI	Two-wire Serial Interface
35	$0044[3]	SPM READY	Store Program Memory Ready

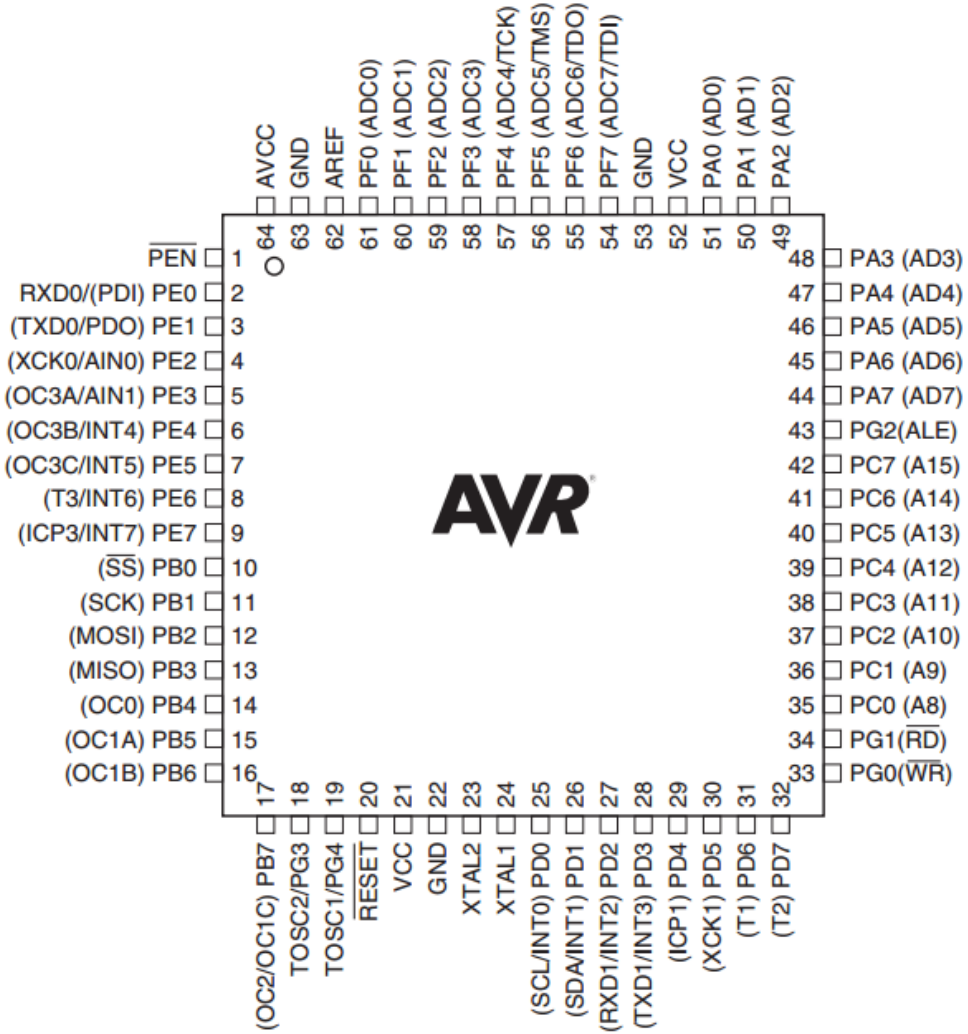
Interrupts

```
/* Interrupt vectors */

/* External Interrupt Request 0 */
#define INT0_vect            _VECTOR(1)
#define SIG_INTERRUPT0           _VECTOR(1)

/* External Interrupt Request 1 */
#define INT1_vect            _VECTOR(2)
#define SIG_INTERRUPT1           _VECTOR(2)

/* External Interrupt Request 2 */
#define INT2_vect            _VECTOR(3)
#define SIG_INTERRUPT2           _VECTOR(3)

/* External Interrupt Request 3 */
#define INT3_vect            _VECTOR(4)
#define SIG_INTERRUPT3           _VECTOR(4)

/* External Interrupt Request 4 */
#define INT4_vect            _VECTOR(5)
#define SIG_INTERRUPT4           _VECTOR(5)

/* External Interrupt Request 5 */
#define INT5_vect            _VECTOR(6)
#define SIG_INTERRUPT5           _VECTOR(6)

/* External Interrupt Request 6 */
#define INT6_vect            _VECTOR(7)
#define SIG_INTERRUPT6           _VECTOR(7)

/* External Interrupt Request 7 */
#define INT7_vect            _VECTOR(8)
#define SIG_INTERRUPT7           _VECTOR(8)
```



Interrupts

```
/* Timer/Counter2 Compare Match */
#define TIMER2_COMP_vect        _VECTOR(9)
#define SIG_OUTPUT_COMPARE2     _VECTOR(9)

/* Timer/Counter2 Overflow */
#define TIMER2_OVF_vect         _VECTOR(10)
#define SIG_OVERFLOW2           _VECTOR(10)

/* Timer/Counter1 Capture Event */
#define TIMER1_CAPT_vect        _VECTOR(11)
#define SIG_INPUT_CAPTURE1      _VECTOR(11)

/* Timer/Counter1 Compare Match A */
#define TIMER1_COMPA_vect       _VECTOR(12)
#define SIG_OUTPUT_COMPARE1A        _VECTOR(12)

/* Timer/Counter Compare Match B */
#define TIMER1_COMPB_vect       _VECTOR(13)
#define SIG_OUTPUT_COMPARE1B        _VECTOR(13)

/* Timer/Counter1 Overflow */
#define TIMER1_OVF_vect         _VECTOR(14)
#define SIG_OVERFLOW1           _VECTOR(14)

/* Timer/Counter0 Compare Match */
#define TIMER0_COMP_vect        _VECTOR(15)
#define SIG_OUTPUT_COMPARE0     _VECTOR(15)

/* Timer/Counter0 Overflow */
#define TIMER0_OVF_vect         _VECTOR(16)
#define SIG_OVERFLOW0           _VECTOR(16)
```



Interrupts

```
/* SPI Serial Transfer Complete */
#define SPI_STC_vect            _VECTOR(17)
#define SIG_SPI                 _VECTOR(17)

/* USART0, Rx Complete */
#define USART0_RX_vect          _VECTOR(18)
#define SIG_USART0_RECV         _VECTOR(18)
#define SIG_UART0_RECV          _VECTOR(18)

/* USART0 Data Register Empty */
#define USART0_UDRE_vect        _VECTOR(19)
#define SIG_USART0_DATA         _VECTOR(19)
#define SIG_UART0_DATA          _VECTOR(19)

/* USART0, Tx Complete */
#define USART0_TX_vect          _VECTOR(20)
#define SIG_USART0_TRANS        _VECTOR(20)
#define SIG_UART0_TRANS         _VECTOR(20)
```

Bit	7	6	5	4	3	2	1	0	
	RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n	UCSRnB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

UCSRnB

- **Bits 7:RX Complete Interrupt Enable**
- **Bits 6:TX Complete Interrupt Enable**

Interrupts

```
ISR(Interrupts_vector)
{
        ISR code...
}
```

```
ISR(TIMER0_COMP_vect)
{
    msec++;
    if(msec == 1000)
    {
        msec = 0;
        sec++;
    }
    if(sec == 100)
        sec = 0;
}
```

```
ISR(INT0_vect)
{
    PORTC = 0xFF;
}
```

Interrupts

- **Control Register**
  - 외부 인터럽트 상태 레지스터 A - EICRA
    - 외부 인터럽트 0~3의 트리거 동작 모드를 설정하는 레지스터
    - 비동기 방식으로 동작하며, 외부인터럽트의 펄스폭이 최소 50 ns 이상이어야 함

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	

ISCn1	ISCn0	설 명
0	0	외부 핀 INTn의 논리 '0'일 때 인터럽트를 요청
0	1	예약됨
1	0	외부 핀 INTn이 하강 에지(논리 '1'에서 논리 '0'으로 변할 때)일 때 비동기 인터럽트를 요청
1	1	외부 핀 INTn이 상승 에지(논리 '0'에서 논리 '1'로 변할 때)일 때 비동기 인터럽트를 요청

  - 외부 인터럽트 상태 레지스터 B - EICRB
    - 외부 인터럽트 4~7의 트리거 동작 모드를 설정하는 레지스터
    - 동기 방식으로 동작하며, 외부인터럽트의 펄스폭이 최소 1클록 사이클 이상이어야 함

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	

ISCn1	ISCn0	설 명
0	0	외부 핀 INTn의 논리 '0'일 때 인터럽트를 요청
0	1	외부 핀 INTn이 하강에지 또는 상승에지 일 때 인터럽트를 요청
1	0	외부 핀 INTn이 하강 에지(논리 '1'에서 논리 '0'으로 변할 때)일 때 인터럽트를 요청
1	1	외부 핀 INTn이 상승 에지(논리 '0'에서 논리 '1'로 변할 때)일 때 인터럽트를 요청

Interrupts

- **Control Register**

External Interrupt Mask Register – EIMSK

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 7 - 0 Enable**

- 외부 인터럽트 마스크 레지스터 - EIMSK
  - 외부 인터럽트의 인터럽트 마스크 레지스터(개별적인 인터럽트 허용)
  - 1로 세트 되면 해당 외부 인터럽트 허용
  - 0으로 클리어 되면 해당 외부 인터럽트 금지

ex) 외부인터럽트0의 허용
```
1. EIMSK |= 0X01;
2. EIMSK |= (1 << INT0);
```

ex) 외부인터럽트0의 금지
```
1. EIMSK &= ~0X01;
2. EIMSK &= ~(1 << INT0);
```

Interrupts

- **Control Register**

External Interrupt Flag Register – EIFR

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 7 - 0**

- 외부 인터럽트 플래그 레지스터 – EIFR
  - 외부 인터럽트의 발생 상황을 알 수 있는 플래그 레지스터
  - 인터럽트 트리거 발생시 1로 세트
  - 인터럽트 서브 루틴 진입시 0으로 자동 클리어
  - 프로그램에서 클리어 하기 위해서는 해당 비트를 1로 쓰기
    - ex) 외부 인터럽트0 플래그의 클리어
      - ; // EIFR == 0X01 (외부 인터럽트 0의 요청이 있음)
      - EIFR |= 0X01; // EIFR == 0X00 (외부 인터럽트 0의 요청이 없음)

Interrupts

- **Control Register**

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	

- 전역 인터럽트 허용 (SREG. I = 1), SREG는 AVR의 상태 레지스터, I 는 7번 비트

- AVR 상태 레지스터
  - - Bit 7 -> I : 전역 인터럽트 허용
    - 인터럽트 요청시 인터럽트 벡터로 점프하도록 허용하는 전역 인터럽트 허용 비트
    - 인터럽트 루틴 진입시 0 으로 클리어
    - 인터럽트 루틴에서 빠져 나갈 때 다시 1로 세트

ex) 프로그램에서 강제 세트 방법

```
SREG |= 0X80;

#include <avr/interrupt.h>
sei();    // C언어 매크로

SEI    ; 어셈블리어
```

ex) 프로그램에서 강제 클리어 방법

```
SREG &= ~0X80;

#include <avr/interrupt.h>
cli();    // C언어 매크로

CLI    ; 어셈블리어
```

Interrupts

```c
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

int msec = 0, sec = 0;


void USART_Init(unsigned int ubrr){
    /*
     *  Baud Rate 9600, Stop Bit 1
     *  Character Size: 8-Bit
     *  No Parity
     */
    UBRR0H = (unsigned char)(ubrr >> 8);
    UBRR0L = (unsigned char)ubrr;
    UCSR0B = (1<<RXEN0) | (1<<TXEN0);
    UCSR0C = (3 << UCSZ0);
}

void USART_Transmit(char data){
    while(!((UCSR0A) & (1<<UDRE0)));    //Data Register Not Empty: Wait
    UDR0 = data;                        //Data Register Empty: Transmit
}


void USART_Transmit_String(char *str){
    while(*str != '₩0')
    USART_Transmit(*str++);
}

char USART_Receive(){
    while(!(UCSR0A & (1<<RXC0)));   //Receive Not complete: Wait
    return UDR0;                    //Receive Complete : Receive
}
```

```c
void Timer_Init(){
    TCCR0 = ( 3 << CS0 );
    TIMSK = (1 << OCIE0 );
    OCR0 = 249;
}

void Interrupt_Init(){
    EIMSK = ( 1 << INT0 ) | ( 1 << INT1 );
    EICRA = ( 1 << ISC01 ) | ( 1 << ISC11 );
}

ISR(TIMER0_COMP_vect){
    msec++;
    if(msec == 1000){
        msec = 0;
        sec++;
    }
    if(sec == 100){
        sec = 0;
    }
}

ISR( INT0_vect ) {
    TIMSK = 0;
    PORTA = 0xff;
}

ISR( INT1_vect ){
    TIMSK = ( 1 << OCIE0 );
    PORTA = 0x00;
}
```

- 스위치(D0)를 누르면 타이머 스톱, 스위치(D1)을 누르면 타이머 재개

Interrupts

```
int main(void)
{
    int a, b;

    DDRA = 0xFF;
    DDRD = 0x00;

    USART_Init(MYUBRR);
    Timer_Init();
    Interrupt_Init();

    USART_Transmit_String("Timer: ");
    _delay_ms(10);
    SREG = 0x80;

    while (1)
    {
        a = sec/10 + 48;
        b = sec%10 + 48;
        USART_Transmit_String("Timer : ");
        USART_Transmit(a);
        USART_Transmit(b);
        USART_Transmit('₩r');
        _delay_ms(100);
    }

}
```

- 스위치(D0)를 누르면 타이머 스톱, 스위치(D1)을 누르면 타이머 재개

과제

1. USART Interrupt를 이용하여 LED On/Off

2. 스위치를 사용하여 interrupt를 통해서 스위치를 누르면 LED가 왕복하면서 움직이고, 다시 스위치를 누르면 LED가 멈추는 프로그램 작성

3. 2번의 프로그램을 ISR 내부에서 작동하도록하고 ISR 내부에서 interrupt가 동작하는지 확인해보고, 동작한다면 어떤 문제가 있는지, 동작하지 않는다면 왜 동작하지 않는지 생각해보기