



[04_ATmega128 IO / Interrupt / Timer]

한국폴리텍대학교 성남캠퍼스

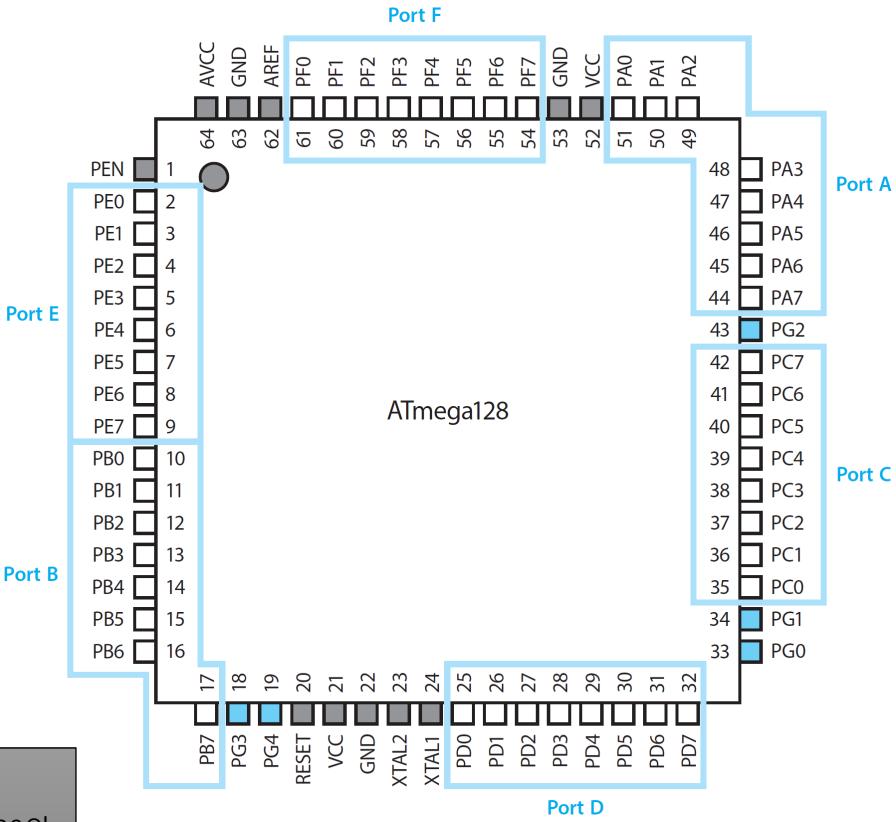
1 Digital Data In & Out

Digital Data In & Out

ATmega128 IO Port

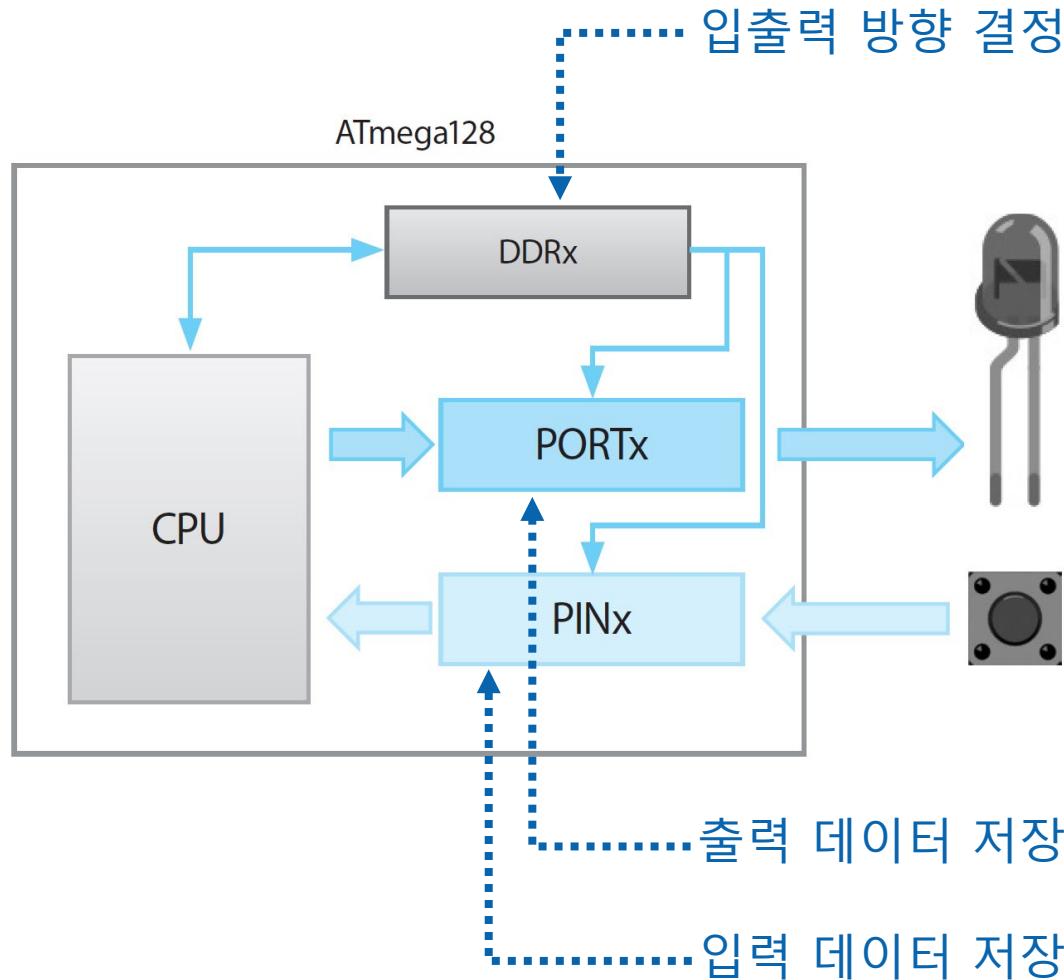
❖ ATmega128 내부의 CPU는 워드 단위인 8비트 단위로 데이터를 처리

- 8개의 입출력 핀을 묶은 포트(port) 단위로 관리
- A에서 G까지 7개의 포트를 가짐
- A~F 포트는 8개 핀이 할당되어 있지만 G 포트는 5개 핀만 할당되어 있음
- G 포트는 연속된 핀 번호에 할당되어 있지 않음
- CPU와 물리적인 ATmega128의 핀 사이에서 데이터 교환 및 제어 역할



Digital Data In & Out

ATmega128 IO Register



Digital Data In & Out

ATmega128 IO Port Register

- ◆ ATmega128 칩의 핀으로 출력할 데이터 저장

비트	7	6	5	4	3	2	1	0
비트 이름	PORTx7	PORTx6	PORTx5	PORTx4	PORTx3	PORTx2	PORTx1	PORTx0
읽기/쓰기	R/W							
초깃값	0	0	0	0	0	0	0	0

(a) PORTA~PORTF

비트	7	6	5	4	3	2	1	0
비트 이름	-	-	-	PORTG4	PORTG3	PORTG2	PORTG1	PORTG0
읽기/쓰기	R	R	R	R/W	R/W	R/W	R/W	R/W
초깃값	0	0	0	0	0	0	0	0

(b) PORTG

Digital Data In & Out

ATmega128 IO DDR Register(Data Direction)

❖ 핀의 입출력 방향 선택 (0 : 입력, 1 : 출력)

비트	7	6	5	4	3	2	1	0
비트 이름	DDx7	DDx6	DDx5	DDx4	DDx3	DDx2	DDx1	DDx0
읽기/쓰기	R/W							
초깃값	0	0	0	0	0	0	0	0

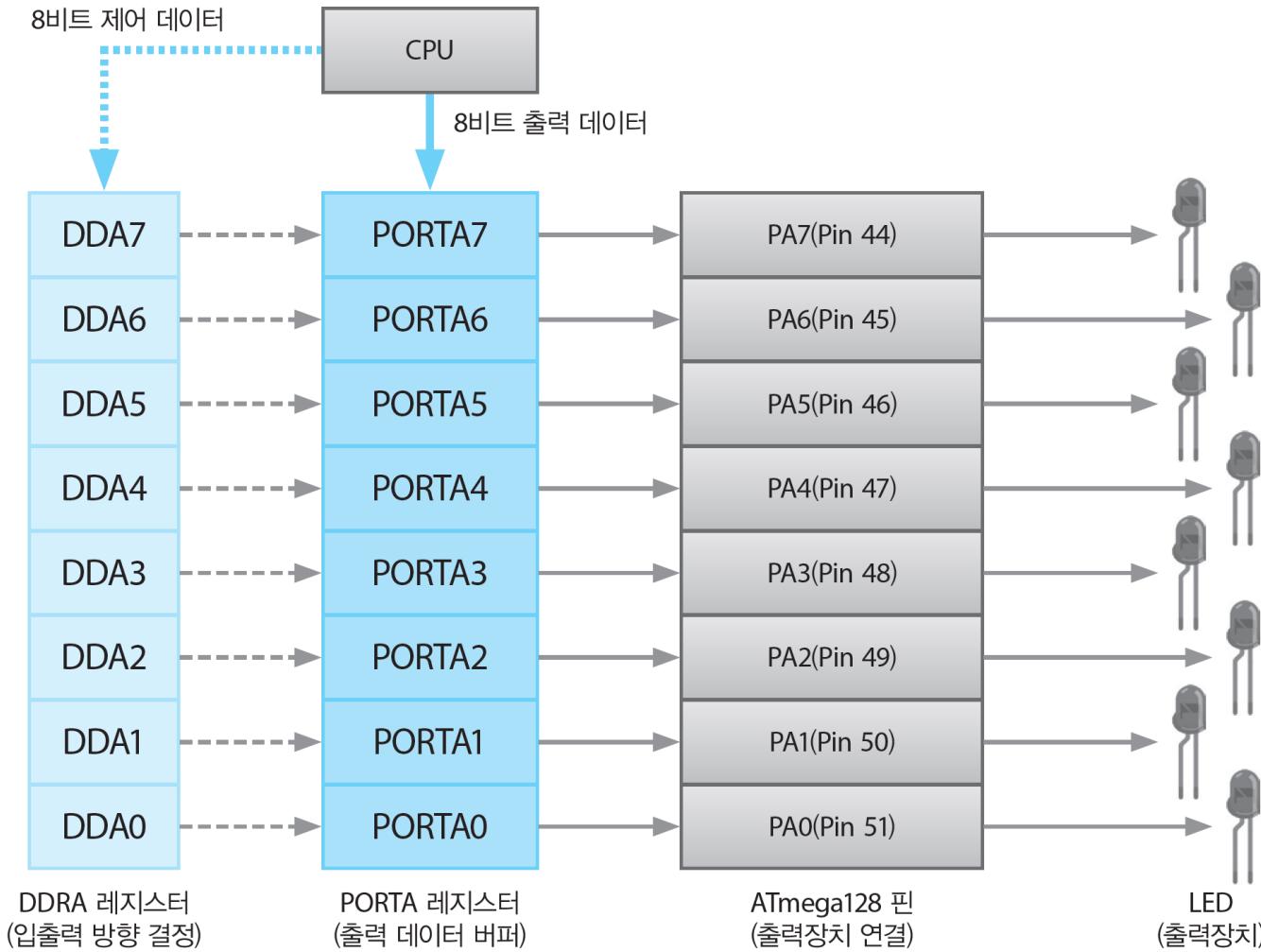
(a) DDRA~DDRF

비트	7	6	5	4	3	2	1	0
비트 이름	-	-	-	DDG4	DDG3	DDG2	DDG1	DDG0
읽기/쓰기	R	R	R	R/W	R/W	R/W	R/W	R/W
초깃값	0	0	0	0	0	0	0	0

(b) DDRG

Digital Data In & Out

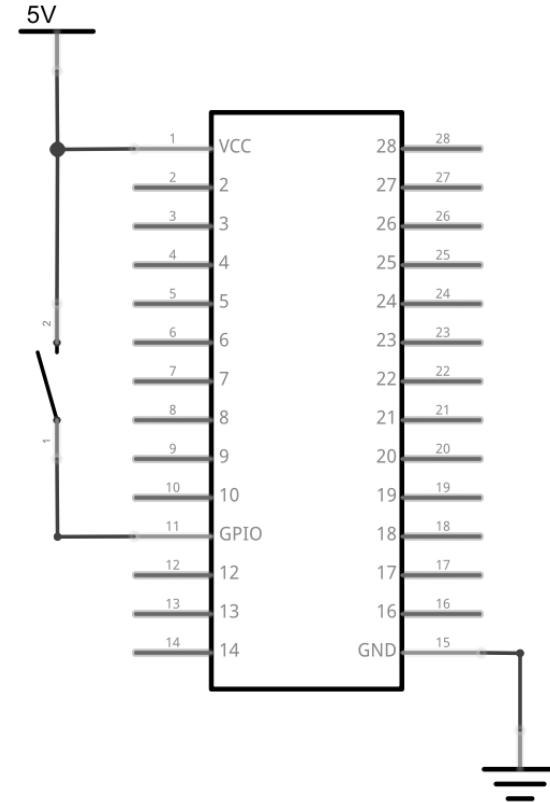
데이터 출력 구조



■ 데이터 입력 기본 구조

- ❖ 버튼이 눌러진 경우 입력(GPIO) 핀에는 5V (HIGH)가 가해 짐

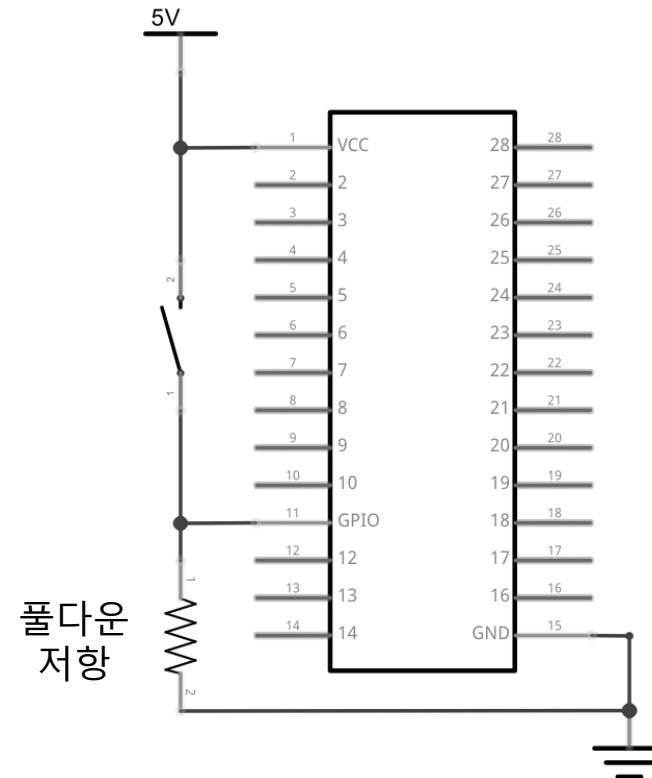
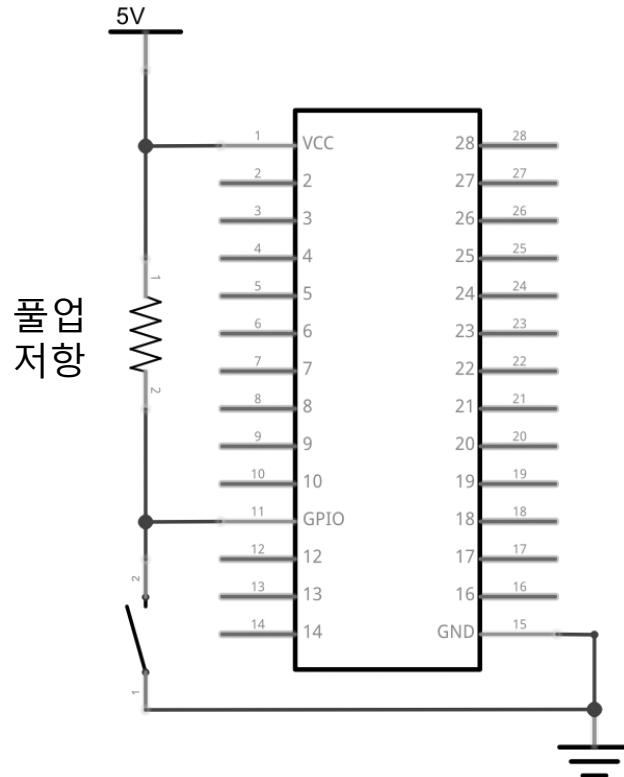
- ❖ 버튼이 눌러지지 않은 경우 입력 핀에 가해지는 값은 알 수 없음
 - 회로가 개방된 경우 플로팅(floating)되어 있다고 이야기함
 - 회로가 개방(open circuit)되는 경우는 피해야 함



■ 데이터 입력 기본 구조

❖ 풀업 : 버튼이 눌러지지 않은 경우 입력 핀에 5V(HIGH)가 가해지도록 보장

❖ 풀다운 : 버튼이 눌러지지 않은 경우 입력 핀에 GND(LOW)가 가해지도록 보장



Digital Data In & Out

ATmega128 IO PIN Register

◆ ATmega128 칩의 핀으로 입력 받은 데이터 저장

비트	7	6	5	4	3	2	1	0
비트 이름	PINx7	PINx6	PINx5	PINx4	PINx3	PINx2	PINx1	PINx0
읽기/쓰기	R	R	R	R	R	R	R	R
초깃값	N/A							

(a) PINA~PINF

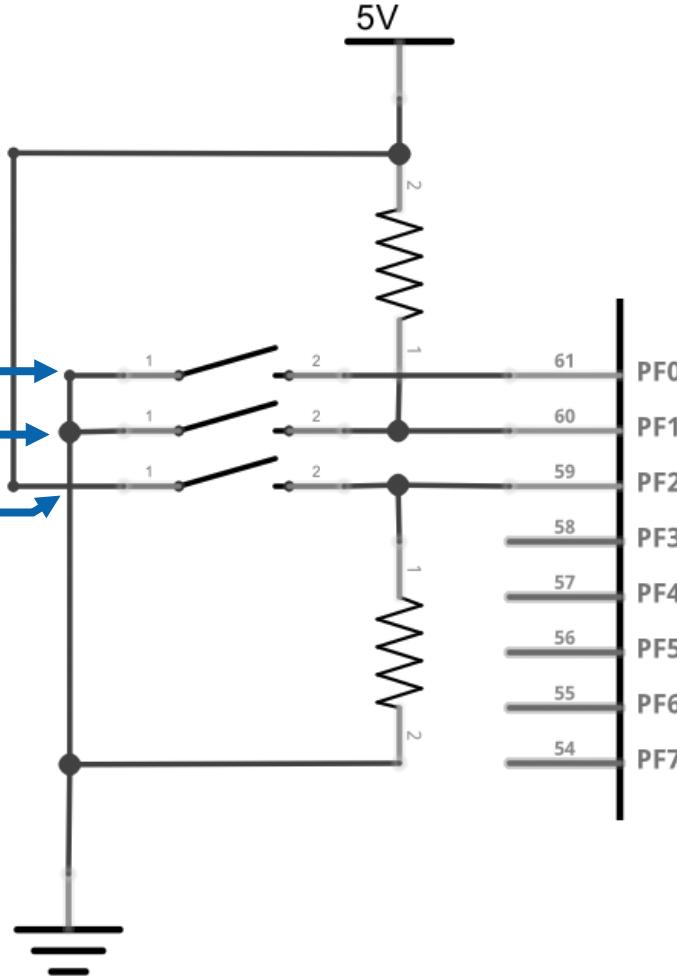
비트	7	6	5	4	3	2	1	0
비트 이름	PING7	PING6	PING5	PING4	PING3	PING2	PING1	PING0
읽기/쓰기	R	R	R	R	R	R	R	R
초깃값	0	0	0	N/A	N/A	N/A	N/A	N/A

(b) PING

■ ATmega128 IO 출력 Full-Up/Down 설정

(DDR 설정으로 입력 상태일 때
출력 레지스터 PORT로
내부 풀업 저항 사용 여부 결정)

- 내부 풀업 저항 사용
- 외부 풀업 저항 사용
- 외부 풀다운 저항 사용



Digital Data In & Out

■ ATmega128 IO 출력 Full-Up/Down 설정

```
#define F_CPU 16000000L
#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
    // 포트 B의 3개 핀(PB0~PB2)만을 출력으로 설정하고
    // 나머지 5개 핀은 디폴트 값을 유지한다.
    DDRB |= 0x07;

    // 포트 F의 3개 핀(PF0~PF2)만을 입력으로 설정하고
    // 나머지 5개 핀은 디폴트 값을 유지한다.
    DDRF &= ~0x07;

    // 포트 F의 PF0 핀에 연결된 내부 풀업 저항을 사용하도록 설정한다.
    PORTF |= 0x01;

    while (1)
    {
        // 포트 B에 연결된 8개 핀 중 PB0~PB2까지의 3개 LED에만
        // 버튼의 상태를 반영하여 출력하고
        // 나머지 5개 핀의 출력은 이전 상태로 유지한다.
        PORTB = (PORTB & 0xF8) + (PINB & 0x07);
    }

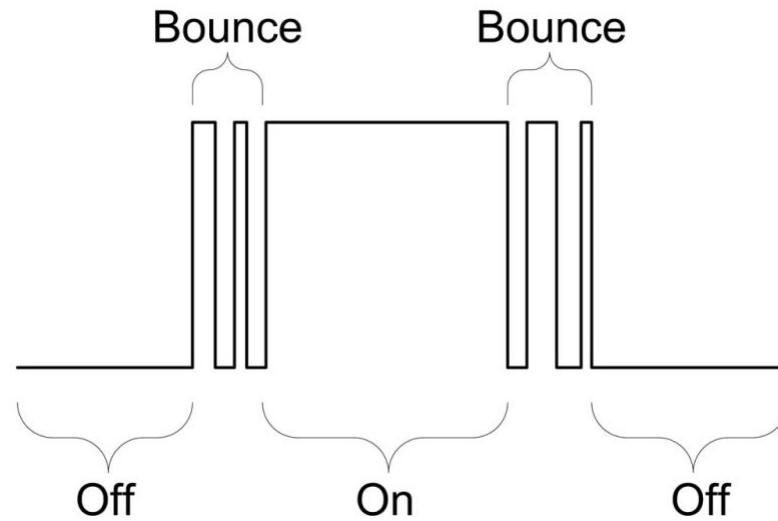
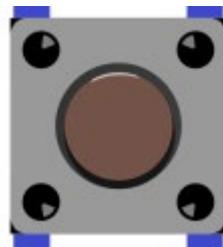
    return 0;
}
```

Digital Data In & Out

Bounce / Chattering

◆ 채터링[chattering]

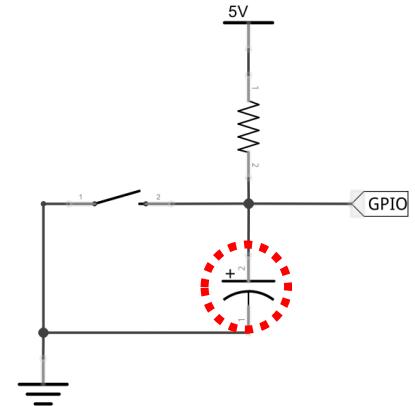
- 전자 회로 내의 스위치 접점이 닫히거나 열리는 순간에 기계적인 진동에 의해 매우 짧은 시간안에 스위치가 붙었다가 떨어지는 것을 반복하는 현상



Bounce / Chattering

❖ 디바운스 : 바운스 현상을 제거하는 방법

- 소프트웨어에 의한 방법 : 시간 지연을 통해 버튼의 빠른 상태 변화 무시
- 하드웨어에 의한 방법 : 커패시터를 통해 버튼의 빠른 상태 변화 무시



```

while (1)
{
    state_current = (PINF & 0x04) >> 2;      // 버튼 상태 읽기
    // 버튼이 눌러지지 않은 상태에서 눌러진 상태로 바뀌는 경우
    if(state_current == 1 && state_previous == 0){
        _delay_ms(30);                      // 디바운스
        pattern = circular_shift_left(pattern);
        PORTB = pattern;
    }
    state_previous = state_current;          // 버튼 상태 업데이트
}

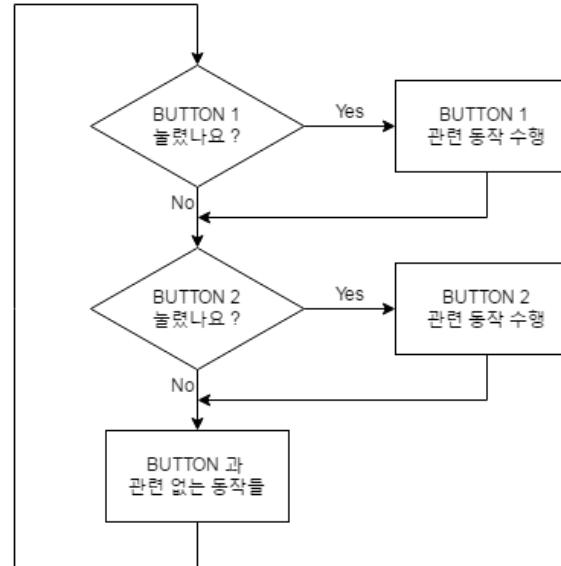
```

2 Interrupt

■ Polling vs. Interrupt

❖ Polling

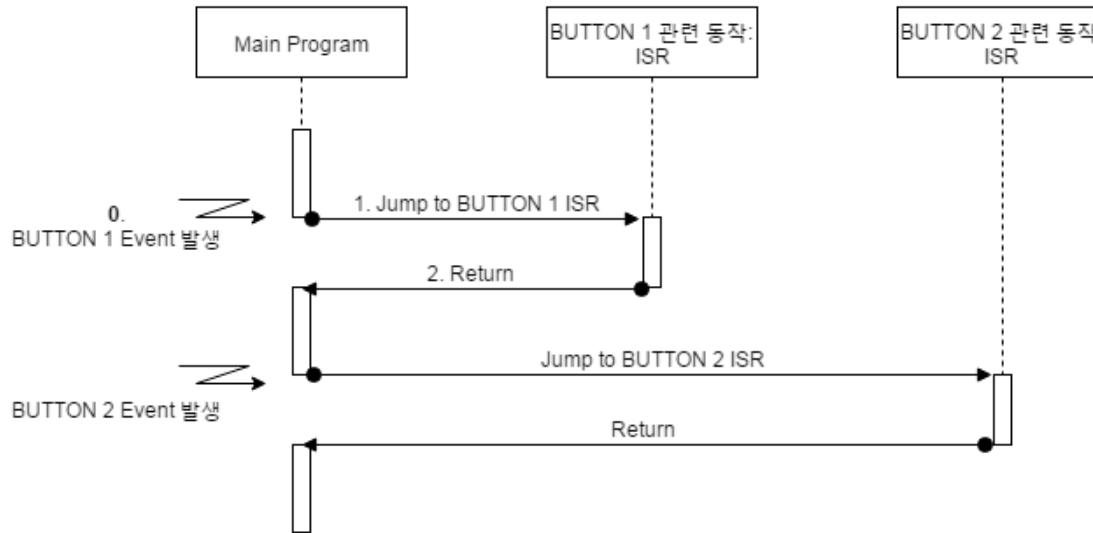
- 코드 나열 순서에 의해 실행 순서 결정
- 모든 코드는 동일한 실행 우선 순위를 가짐
- 코드 A에 의해 코드 B의 실행이 지연될 수 있음
- 정해진 순서에 따라 실행되는 구조로 하드웨어의 지원은 필요하지 않음
- 코드 작성 및 이해가 쉬움



■ Polling vs. Interrupt

❖ Interrupt

- 우선 순위에 따라 실행 순서 결정
- 인터럽트에 따라 서로 다른 실행 우선 순위를 가짐
- (우선 순위가 낮은) 코드 A에 의해 (우선 순위가 높은) 코드 B의 실행이 지연되지 않음
- (우선 순위가 높은) 비정상적인 코드를 먼저 실행되는 구조로 하드웨어에 의해 우선 순위에 따른 처리 지원
- 코드 작성 및 이해가 복잡하고 어려움



ATmega128 Interrupt

◆ ALU(Arithmetic Logic Unit) – Status Register

- Status register는 Instruction set에 따른 명령어를 실행한 후 ALU에 대한 정보를 표시
- Status register의 사용을 통해 전용 비교 명령어를 사용하지 않아 간결한 코드를 작성할 수 있음
- Status register는 Interrupt Routine에 진입할 때 자동으로 저장되며 Interrupt Routine에서 복귀할 때는 자동으로 복원되지 않으며, 소프트웨어로 처리해야 함

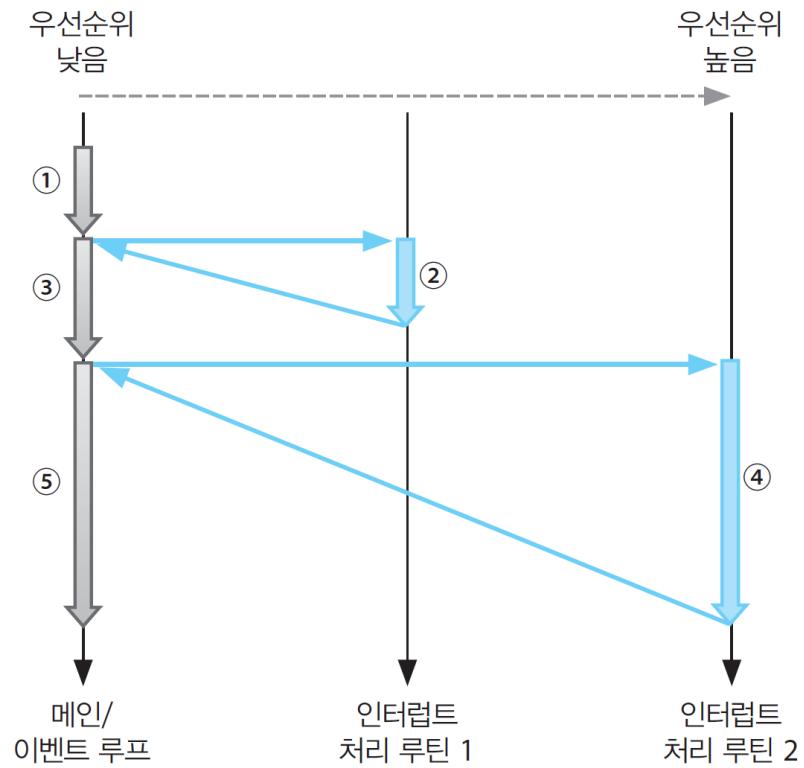
비트	7	6	5	4	3	2	1	0
읽기/쓰기	I R/W 0	T R/W 0	H R/W 0	S R/W 0	V R/W 0	N R/W 0	Z R/W 0	C R/W 0
초깃값	0	0	0	0	0	0	0	0

비트 번호	비트 이름	설명
7	I	Global Interrupt Enable: 전역적인 인터럽트 발생을 허용한다.
6	T	Bit Copy Storage: 비트 복사를 위한 BLD(Bit Load), BST(Bit Store) 명령에서 사용한다. BST 명령에 의해 비트 값을 비트 T에 저장할 수 있으며, BLD 명령에 의해 비트 T의 내용을 읽어 올 수 있다.
5	H	Half Carry Flag: 산술 연산에서의 보조 캐리 발생을 나타낸다. 보조 캐리는 바이트 단위 연산에서 하위 니블(nibble)로부터 발생하는 캐리를 말한다.
4	S	Sign Bit: 부호 비트로, 음수 플래그(N)와 2의 보수 오버플로 플래그(V)의 배타적 논리합(XOR)으로 설정된다($S = N \oplus V$).
3	V	2's Complement Overflow Flag: 2의 보수를 이용한 연산에서 자리 올림이 발생하였음을 나타낸다.
2	N	Negative Flag: 산술 연산이나 논리 연산에서 결과가 음수임을 나타낸다.
1	Z	Zero Flag: 산술 연산이나 논리 연산에서 결과가 0임을 나타낸다.
0	C	Carry Flag: 산술 연산이나 논리 연산에서 캐리가 발생하였음을 나타낸다.

■ ATmega128 Interrupt

❖ ISR – Interrupt Service Routine

- 마이크로 컨트롤러가 특정 작업을 즉시 처리하도록 요구하는 비정상적인 사건
- 하드웨어에 의해 호출되는 “함수”로 이해 가능
- 인터럽트가 발생하면 현재 실행 중인 코드를 정지하고 Interrupt Service Routine(ISR)으로 즉시 이동하여 인터럽트를 먼저 처리
- 동시에 여러 개의 인터럽트가 발생하면 우선 순위가 높은 인터럽트를 우선 처리
- 인터럽트 처리 순서
 - ✓ 인터럽트 발생
 - ✓ 인터럽트 벡터 테이블의 해당 인터럽트 벡터가 저장된 인터럽트 벡터 테이블 위치로 이동
 - ✓ 인터럽트 벡터 값에 해당하는 ISR로 이동
 - ✓ ISR 처리
 - ✓ 인터럽트 처리 이전에 수행하던 전의 코드로 되돌아감



■ ATmega128 Interrupt

❖ AVR 마이크로 컨트롤러의 대표 특징

- 35개의 인터럽트 사용 가능
- 인터럽트 벡터 : 인터럽트가 발생하였을 때 처리가 옮겨질 해당 ISR의 메모리 주소
- 인터럽트 벡터 테이블 : 35개 인터럽트에 대한 인터럽트 벡터를 모아놓은 테이블
 - ✓ 플래시 메모리의 0x0000 번지에서 0x0045 번지까지 기록되어 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

Interrupt

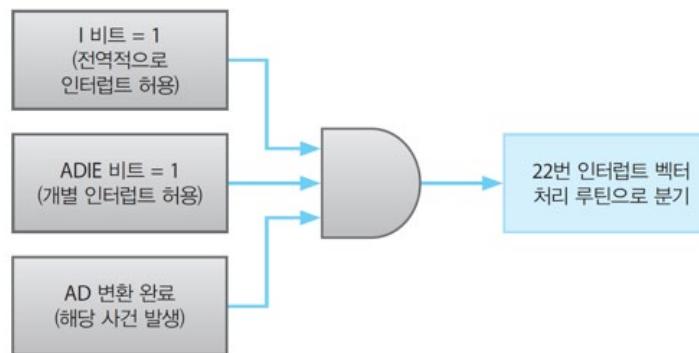
ATmega128 Interrupt

Vector No.	Program Address ⁽²⁾	Source	Interrupt Definition
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/Counter1 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

ATmega128 Interrupt

❖ Interrupt 처리 조건

- 전역적 인터럽트 비트 세트
 - ✓ 상태 레지스터(status register) SREG의 7번 'I' 비트 세트
 - ✓ 세트를 위해 sei(), 클리어를 위해 cli() 함수 사용
- 개별 인터럽트 활성화 비트 세트
 - ✓ 디폴트로 인터럽트는 비활성화 상태에 있음
 - ✓ 개별 인터럽트 별로 인터럽트 활성화 비트 존재
- 인터럽트 발생 조건 충족(ADC)



■ ATmega128 ISR(Interrupt Service Routine)

❖ Interrupt Service Routine

- 인터럽트가 발생하였을 때 처리 루틴
- 하드웨어에 의해서만 호출할 수 있음
 - ✓ 코드 상에서는 ISR을 호출할 수 있는 코드가 없으며, 해당 Interrupt가 발생할 경우 지정된 Interrupt vector로 이동하여 지정된 소프트웨어 처리

```
#include <avr/interrupt.h>

ISR(ADC_vect)
{
    // 인터럽트 처리 코드
}
```

- 반환값이 없고 매개변수의 데이터 형이 없는 함수
 - ✓ 모든 ISR은 동일한 이름을 가짐
 - ✓ 처리할 인터럽트의 종류를 인터럽트 벡터 이름인 매개변수로 구분

Interrupt

■ ATmega128 ISR(Interrupt Service Routine)

❖ Interrupt Vector Table

벡터 번호	벡터 이름	인터럽트 정의
1	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, JTAG AVR Reset
2	INT0_vect	External Interrupt Request 0
3	INT1_vect	External Interrupt Request 1
4	INT2_vect	External Interrupt Request 2
5	INT3_vect	External Interrupt Request 3
6	INT4_vect	External Interrupt Request 4
7	INT5_vect	External Interrupt Request 5
8	INT6_vect	External Interrupt Request 6
9	INT7_vect	External Interrupt Request 7
10	TIMER2_COMP_vect	Timer/Counter2 Compare Match
11	TIMER2_OVF_vect	Timer/Counter2 Overflow
12	TIMER1_CAPT_vect	Timer/Counter1 Capture Event
13	TIMER1_COMPA_vect	Timer/Counter1 Compare Match A
14	TIMER1_COMPB_vect	Timer/Counter1 Compare Match B
15	TIMER1_OVF_vect	Timer/Counter1 Overflow
16	TIMERO_COMP_vect	Timer/Counter0 Compare Match
17	TIMERO_OVF_vect	Timer/Counter0 Overflow
18	SPI_STC_vect	SPI Serial Transfer Complete
19	USART0_RX_vect	USART0 Rx Complete
20	USART0_UDRE_vect	USART0 Data Register Empty

Interrupt

■ ATmega128 ISR(Interrupt Service Routine)

❖ Interrupt Vector Table

벡터 번호	벡터 이름	인터럽트 정의
21	USART0_TX_vect	USART0 Tx Complete
22	ADC_vect	ADC Conversion Complete
23	EE_READY_vect	EEPROM Ready
24	ANALOG_COMP_vect	Analog Comparator
25	TIMER1_COMPC_vect	Timer/Counter1 Compare Match C
26	TIMER3_CAPT_vect	Timer/Counter3 Capture Event
27	TIMER3_COMPA_vect	Timer/Counter3 Compare Match A
28	TIMER3_COMPB_vect	Timer/Counter3 Compare Match B
29	TIMER3_COMPC_vect	Timer/Counter3 Compare Match C
30	TIMER3_OVF_vect	Timer/Counter3 Overflow
31	USART1_RX_vect	USART1 Rx Complete
32	USART1_UDRE_vect	USART1 Data Register Empty
33	USART1_TX_vect	USART1 Tx Complete
34	TWI_vect	Two-wire Serial Interface
35	SPM_READY_vect	Store Program Memory Ready

■ ATmega128 ISR(Interrupt Service Routine)

❖ Interrupt 사용 시 주의사항 - 1

- 중첩된 인터럽트
 - ✓ 인터럽트는 인터럽트될 수 없음
 - ✓ ISR 실행 중 발생되는 인터럽트는 처리되지 못함
 - ✓ ISR은 가능한 짧게 작성하는 것이 바람직함
- 인터럽트 우선순위
 - ✓ 번호가 낮은 인터럽트가 우선순위가 높음
 - ✓ 우선순위가 높은 인터럽트가 먼저 처리됨
 - ✓ 우선순위가 가장 높은 인터럽트는 리셋 인터럽트임

■ ATmega128 ISR(Interrupt Service Routine)

❖ Interrupt 사용 시 주의사항 - 2

- C언어의 컴파일러는 아래 코드의 최적화를 위해 while문이 필요 없다고 판단하고 i에 10을 할당

```
int i = 0;
while (i < 10)
    i++;
printf("%d\n", i); // 10
```



```
int i = 10;      // 반복문을 없애버리고 10을 할당
printf("%d\n", i); // 10
```

- 최적화 방지의 필요성
 - ✓ ISR은 코드 내에서 호출되는 부분이 없으므로 ISR은 메인 코드와 무관한 코드로 간주될 수 있음
 - ✓ 컴파일러의 최적화 과정에서 ISR에서 변수 값을 변경하는 것은 무의미할 수 있음
 - ✓ ISR에서 값을 변경하는 변수는 'volatile' 키워드를 사용하여야 함

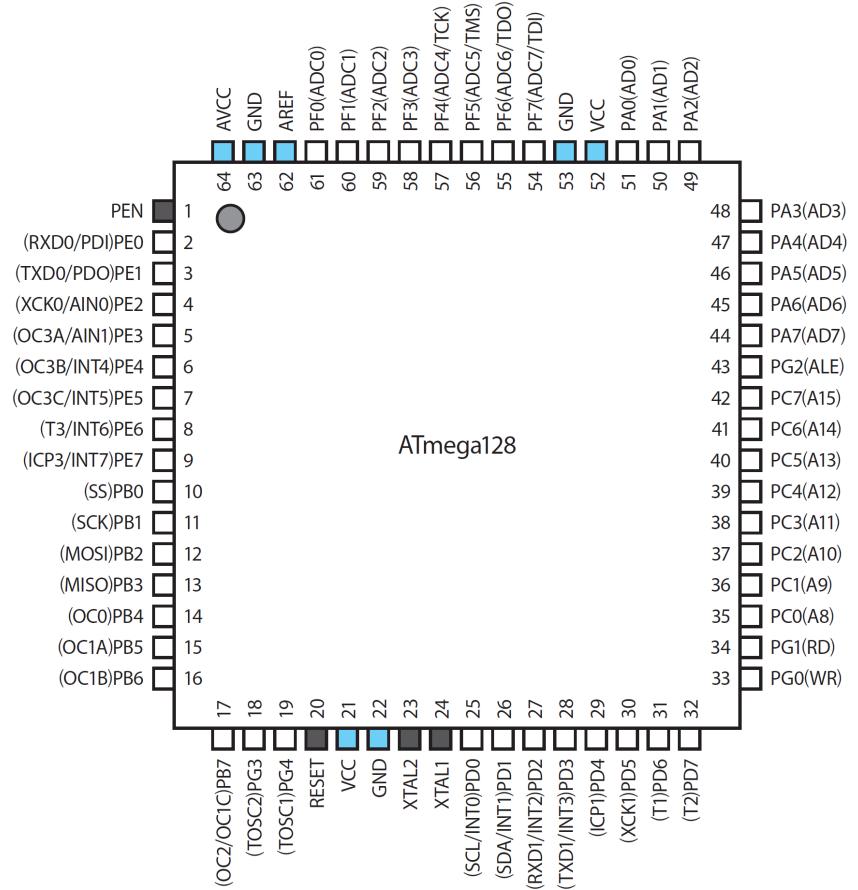
```
volatile int value = 0;

ISR(INTERRUPT_vect)
{
    value = (value + 1) % 2;
}
```

ATmega128 External Interrupt

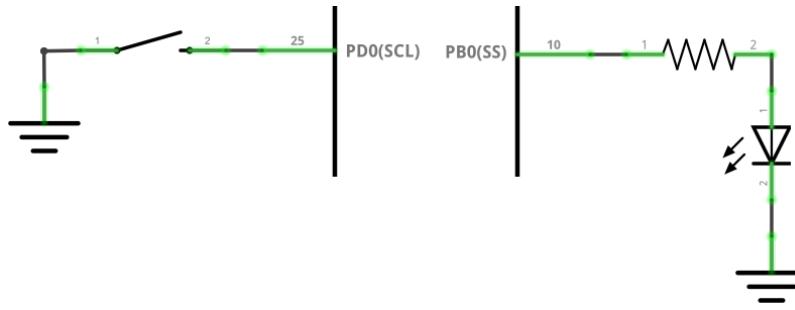
- RESET을 제외하고 가장 우선순위가 높은 인터럽트
- 범용 입출력 핀의 값이나 상태 변화 시에 인터럽트 발생
 - ✓ 정해진 8개 핀으로만 외부 인터럽트 발생 가능

외부 인터럽트	포트	ATmega128 핀 번호
INT0	PD0	25
INT1	PD1	26
INT2	PD2	27
INT3	PD3	28
INT4	PE4	6
INT5	PE5	7
INT6	PE6	8
INT7	PE7	9



■ ATmega128 External Interrupt

- INT0로 Interrupt 신호를 입력한 뒤 PORTB0의 LED를 이용하여 Interrupt 처리 확인 예시



- main 함수 내에는 INIT_INT0() 함수를 호출하여 Interrupt를 활성화시킨 후 별도로 Interrupt Service Routine을 호출하지 않음

```
ISR(INT0_vect)
{
    state = (state + 1) % 2;           // LED 상태 전환
}
```

```
void INIT_INT0(void)
{
    EIMSK |= (1 << INT0);          // INT0 인터럽트 활성화
    EICRA |= (1 << ISC01);         // 하강 에지에서 인터럽트 발생
    sei();                           // 전역적으로 인터럽트 허용
}
```

ATmega128 External Interrupt

◆ EIMSK Register

- External Interrupt Mask Register
 - ✓ INT7 - INT0 비트가 1로 설정되고 상태 레지스터(SREG)의 I 비트가 1로 설정되면 해당 외부 핀 Interrupt가 활성화

Bit	7	6	5	4	3	2	1	0	EIMSK
Read/Write	R/W								
Initial Value	0	0	0	0	0	0	0	0	

◆ EICRA/EICRB Register

- External Interrupt Control Register A / B
 - ✓ 외부 인터럽트 제어 레지스터(EICRA 및 EICRB)의 인터럽트 감지 제어 비트는 외부 인터럽트가 Rising/Falling Edge에서 활성화, 또는 Level 감지되는지를 정의

Bit	7	6	5	4	3	2	1	0	EICRA	ISCn1	ISCn0	인터럽트 발생 시점
Read/Write	R/W											
Initial Value	0	0	0	0	0	0	0	0				
Bit	7	6	5	4	3	2	1	0	EICRB	ISCn1	ISCn0	인터럽트 발생 시점
Read/Write	R/W											
Initial Value	0	0	0	0	0	0	0	0				

3 Timer

■ Timer / Counter

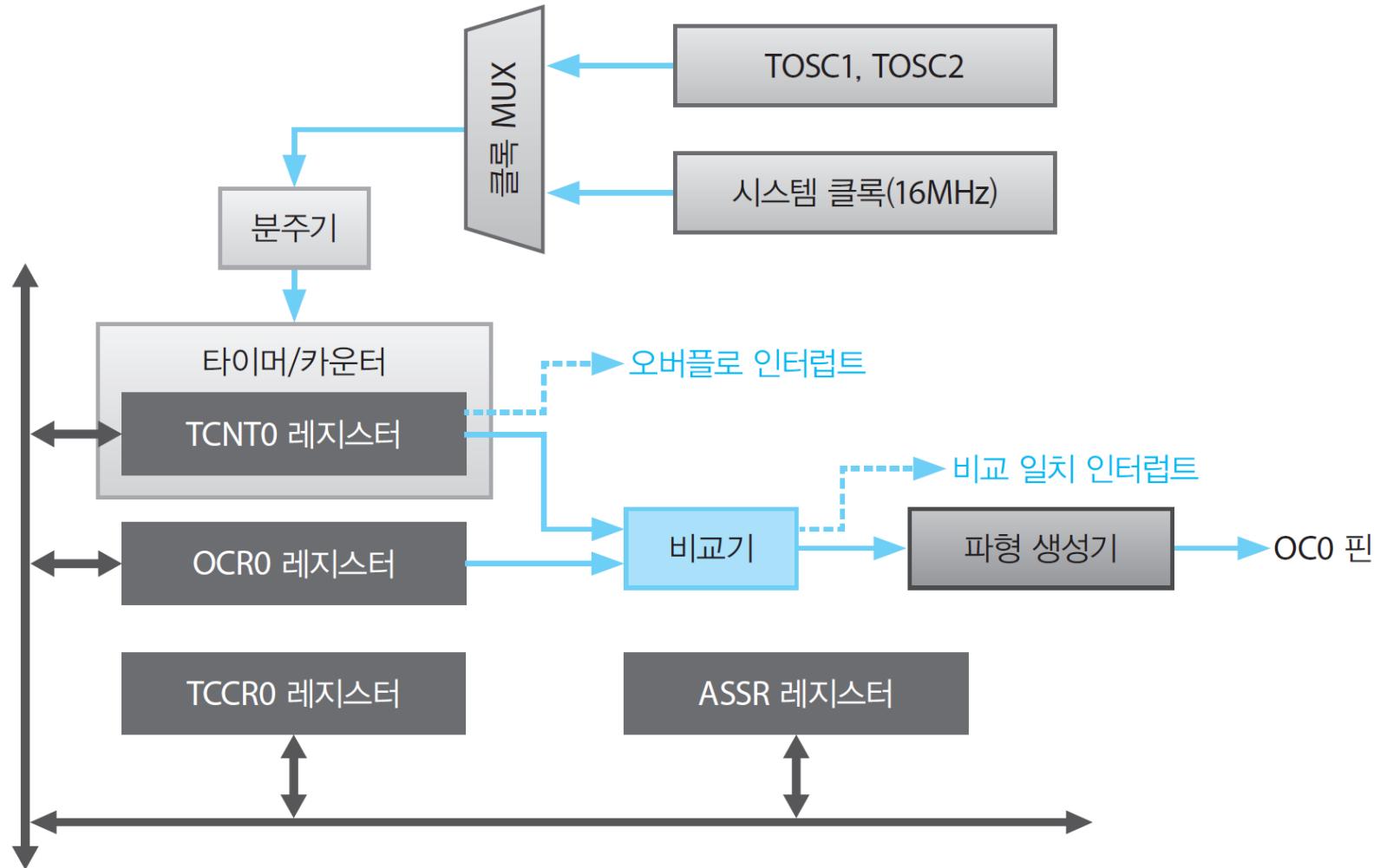
❖ Timer / Counter 개요

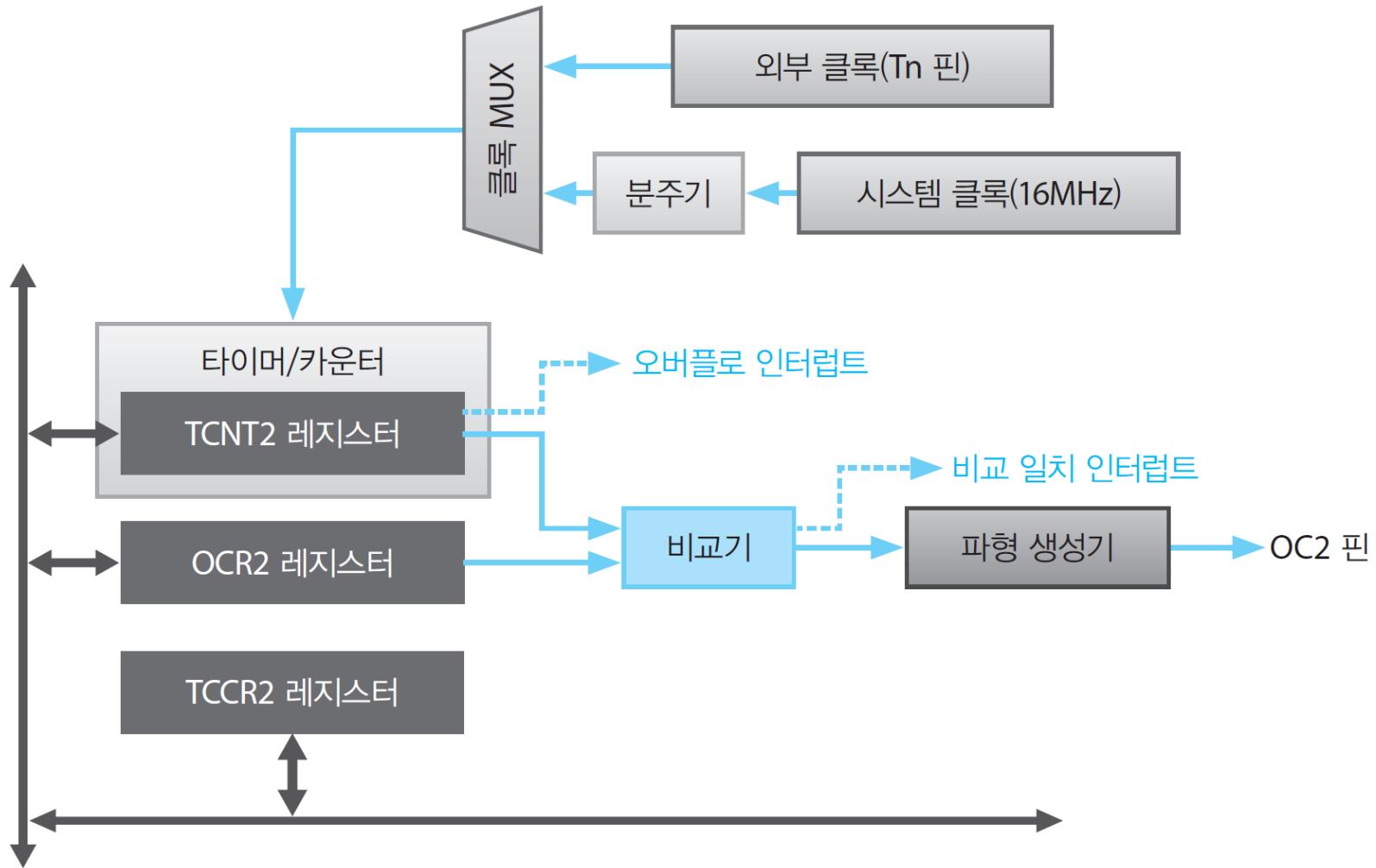
- 입력 펄스를 세는 장치, 즉, 카운터임
- 일정한 주기의 펄스를 셈으로써 시간 측정, 즉, 타이머 역할 수행 가능
- Micro-Controller의 시스템 클럭 사용 가능

❖ ATmega128 Timer/Counter

- 4개의 타이머/카운터 제공
- 0번과 2번의 8비트 타이머/카운터 (0~255 카운트)
- 1번과 3번의 16비트 타이머/카운터 (0~65535 카운트)
- 16MHz 시스템 클럭을 사용하는 경우 256개 클럭의 발생 시간은 0.016ms(8비트 1-Cycle)
- 시스템 클럭을 분주하여 측정 시간 조절 가능

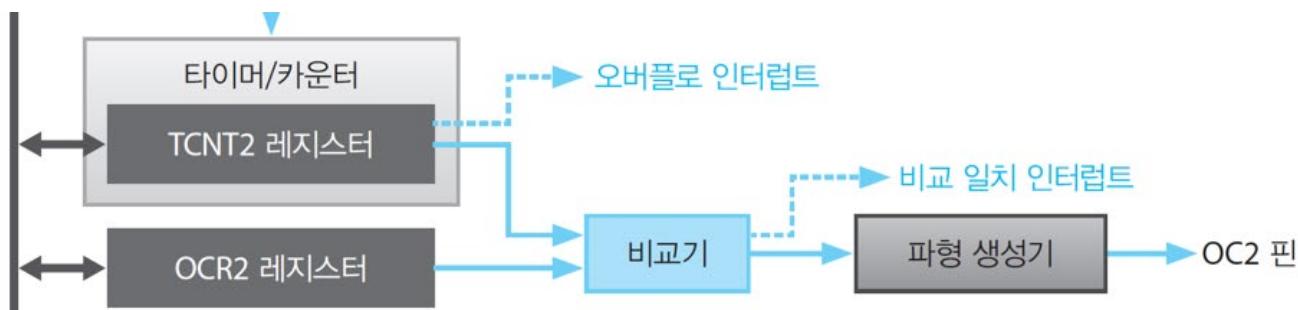
■ Timer/Counter0 (8-bit)



 Timer/Counter2 (8-bit)

■ Timer / Counter Interrupt

- 현재까지의 펄스 수는 TCNT_n 레지스터에 저장됨
- 오버플로 인터럽트
 - ✓ 최대로 셀 수 있는 펄스의 수를 넘어설 때 TCNT_n 레지스터가 0으로 바뀌면서 발생
- 비교 일치 인터럽트
 - ✓ TCNT_n(Timer/Counter Register n)의 값이 미리 설정된 OCR_n(Output Compare Register n) 레지스터의 값과 일치하는 경우 발생
 - ✓ 비교 일치 인터럽트가 발생할 때 OC_n 핀을 통해 파형 출력 가능



비트	7	6	5	4	3	2	1	0
TCNT0[7:0]								
읽기/쓰기	R/W							
초깃값	0	0	0	0	0	0	0	0

■ 8-bit Timer/Counter

❖ Overflow Interrupt Example-2

```
int main(void)
{
    DDRB = 0x01;                      // PB0 핀을 출력으로 설정
    PORTB = 0x00;                      // LED는 끈 상태에서 시작

    // 분주비를 1024로 설정
    TCCR0 |= (1 << CS02) | (1 << CS01) | (1 << CS00);

    TIMSK |= (1 << TOIE0);           // 오버플로 인터럽트 허용

    sei();                            // 전역적으로 인터럽트 허용

    while(1){ }
    return 0;
}
```

8-bit Timer/Counter

- ❖ TCCRn (Timer Counter Control Register, 분주비 설정)

비트	7	6	5	4	3	2	1	0
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00
읽기/쓰기	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
초깃값	0	0	0	0	0	0	0	0

CS02	CS01	CS00	설명
0	0	0	클록 소스 없음(타이머/카운터 정지)
0	0	1	분주비 1(분주 없음)
0	1	0	분주비 8
0	1	1	분주비 32
1	0	0	분주비 64
1	0	1	분주비 128
1	1	0	분주비 256
1	1	1	분주비 1,024

8-bit Timer/Counter

❖ TIMSK (Timer Counter Interrupt Mask Register, Interrupt 활성화)

비트	7	6	5	4	3	2	1	0
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0
읽기/쓰기	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
초깃값	0	0	0	0	0	0	0	0

- 디폴트로 인터럽트는 비활성화 상태에 있음
 - ✓ OCIE0 비트 : 비교일치 인터럽트 활성화 비트
 - ✓ TOIE0 비트 : 오버플로 인터럽트 활성화 비트

벡터 번호	인터럽트	벡터 이름	인터럽트 허용 비트	
10	비교 일치 인터럽트	TIMER2_COMP_vect	OCIE2	Timer/Counter 2 Output Compare Match Interrupt Enable
11	오버플로 인터럽트	TIMER2_OVF_vect	TOIE2	Timer/Counter 2 Overflow Interrupt Enable
16	비교 일치 인터럽트	TIMER0_COMP_vect	OCIE0	Timer/Counter 0 Output Compare Match Interrupt Enable
17	오버플로 인터럽트	TIMER0_OVF_vect	TOIE0	Timer/Counter 0 Overflow Interrupt Enable

8-bit Timer/Counter

❖ 비교일치 Interrupt Example - 1

- TCNT_n 값과 비교할 값을 OCR_n 레지스터에 설정

비트	7	6	5	4	3	2	1	0
	OCR0[7:0]							
읽기/쓰기	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
초깃값	0	0	0	0	0	0	0	0

- 비교 일치 인터럽트 발생 후 TCNT_n 값을 0으로 초기화되지 않음

```
#include <avr/io.h>
#include <avr/interrupt.h>
int count = 0; // 비교일치가 발생한 횟수
int state = 0; // LED 점멸 상태
ISR(TIMER0_COMP_vect)
{
    count++;
    TCNT0 = 0; // 자동으로 0으로 변하지 않는다
    if(count == 64){ // 비교일치 64회 발생 = 약 0.5초 경과
        count = 0; // 카운터 초기화
        state = !state; // LED 상태 반전
        if(state) PORTB = 0x01; // LED 켜기
        else PORTB = 0x00; // LED 끄기
    }
}
```

■ 8-bit Timer/Counter

❖ 비교일치 Interrupt Example - 2

```
int main(void)
{
    DDRB = 0x01;                      // PB5 핀을 출력으로 설정
    PORTB = 0x00;                      // LED는 끈 상태에서 시작

    // 분주비를 1024로 설정
    TCCR0 |= (1 << CS02) | (1 << CS01) | (1 << CS00);

    OCR0 = 128;                        // 비교일치 기준값

    TIMSK |= (1 << OCIE0);           // 비교일치 인터럽트 허용

    sei();                             // 전역적으로 인터럽트 허용

    while(1){ }

    return 0;
}
```

■ 8-bit Timer/Counter

❖ 출력파형 생성 모드 - 1

비트	7	6	5	4	3	2	1	0
	FOCO	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00
읽기/쓰기	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
초깃값	0	0	0	0	0	0	0	0

- TCCR0 레지스터의 WGM(Waveform Generation Mode) 비트에 의해 제어
 - 1번과 3번의 PWM 관련 모드는 추후 학습할 예정
 - 정상 모드 : 비교 일치 발생 시 TNCTn 값을 초기화하지 않음
 - CTC 모드 : 비교 일치 발생 시 TNCTn 값을 초기화함 (Clear Timer on Compare match)
 - 정상 및 CTC 모드에서 파형 출력은 COM 비트에 의해 제어됨

모드 번호	WGM01 (CTC)	WGM00 (PWM)	타이머/카운터 모드	TOP	설명		
					COM01	COM00	
0	0	0	정상	0xFF	0	0	OC0 핀으로 데이터가 출력되지 않으며, OC0 핀은 일반적인 범용 입출력 핀으로 동작한다.
1	0	1	위상 교정 PWM	0xFF	0	1	비교 일치가 발생하면 OC0 핀의 출력은 반전된다.
2	1	0	CTC	OCR0	1	0	비교 일치가 발생하면 OC0 핀의 출력은 LOW 값으로 바뀐다(clear).
3	1	1	고속 PWM	0xFF	1	1	비교 일치가 발생하면 OC0 핀의 출력은 HIGH 값으로 바뀐다(set).

■ 8-bit Timer/Counter

❖ 파형 출력 정상모드 Example - 1

- 비교 일치 인터럽트가 발생하는 경우 Ocn 핀으로 파형 출력 가능
 - ✓ 8비트 타이머/카운터는 1개, 16배트 타이머/카운터는 3개의 비교 일치 인터럽트를 사용할 수 있으며 파형 출력 핀 개수 역시 동일함
- 다음 코드의 경우 LED를 점멸하는 코드가 존재하지 않음
 - ✓ 파형 출력에 의해 LED가 점멸되며 이는 소프트웨어 코드가 아닌 하드웨어에 의해 제어

```
#include <avr/io.h>
#include <avr/interrupt.h>

volatile int count = 0;           // 비교일치가 발생한 횟수
int state = 0;                  // LED 점멸 상태

ISR(TIMER0_COMP_vect)
{
    count++;
    TCNT0 = 0;                 // 자동으로 0으로 변하지 않는다
}
```

타이머/카운터	파형 출력 핀	ATmega128 핀 번호
0	OC0	PB4
	OC1A	PB5
	OC1B	PB6
	OC1C	PB7
1	OC2	PB7
	OC3A	PE3
	OC3B	PE4
	OC3C	PE5

8-bit Timer/Counter

❖ 파형 출력 정상모드 Example - 2

```
int main(void)
{
    // 파형 출력 핀인 OC0(PB4) 핀을 출력으로 설정
    DDRB = 0x10;
    PORTB = 0x00;                                // LED는 끈 상태에서 시작

    // 분주비를 1024로 설정
    TCCR0 |= (1 << CS02) | (1 << CS01) | (1 << CS00);

    OCR0 = 255;                                  // 비교일치 기준값

    // 비교일치 인터럽트 발생 시 OC0 핀의 출력을 반전
    TCCR0 |= (1 << COM00);

    TIMSK |= (1 << OCIE0);                      // 비교일치 인터럽트 허용

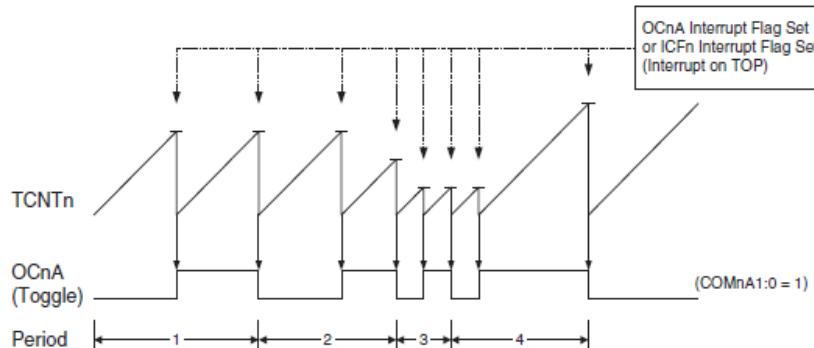
    sei();                                       // 전역적으로 인터럽트 허용

    while(1){ }
    return 0;
}
```

8-bit Timer/Counter

❖ 파형 출력 CTC모드 Example - 1

- 비교 일치 발생 시 TCNT_n 값을 초기화함 (Clear Timer on Compare match)



```
#include <avr/io.h>
#include <avr/interrupt.h>

volatile int count = 0;           // 비교일치가 발생한 횟수
int state = 0;                  // LED 점멸 상태

ISR(TIMER0_COMP_vect)
{
    count++;
}
```

8-bit Timer/Counter

❖ 파형 출력 CTC모드 Example - 2

```
int main(void)
{
    // 파형 출력 핀인 OC0(PB4) 핀을 출력으로 설정
    DDRB = 0x10;
    PORTB = 0x00;                                // LED는 끈 상태에서 시작

    // 분주비를 1024로 설정
    TCCR0 |= (1 << CS02) | (1 << CS01) | (1 << CS00);

    TCCR0 |= (1 << WGM01);                      // CTC 모드

    OCR0 = 255;                                  // 비교일치 기준값

    // 비교일치 인터럽트 발생 시 OC0 핀의 출력을 반전
    TCCR0 |= (1 << COM00);

    TIMSK |= (1 << OCIE0);                      // 비교일치 인터럽트 허용

    sei();                                       // 전역적으로 인터럽트 허용

    while(1){ }
    return 0;
}
```

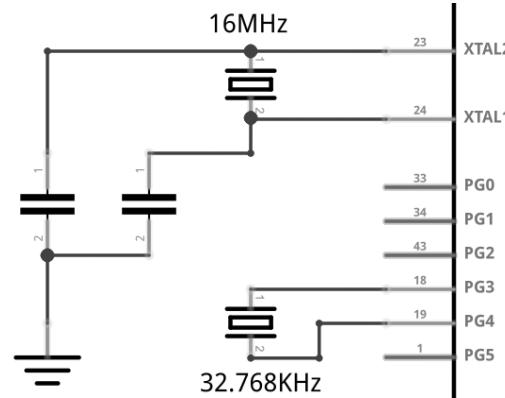
8-bit Timer/Counter

❖ ASSR (Asynchronous Status Register)

Bit	7	6	5	4	3	2	1	0	
Read/Write	-	-	-	-	AS0	TCN0UB	OCR0UB	TCR0UB	ASSR
Initial Value	0	0	0	0	0	0	0	0	

- 0번 타이머/카운터에서만 사용 가능
- 시스템 클럭이 아닌 외부 오실레이터를 클럭 소스로 사용 가능
 - ✓ TOSC 핀에 외부 클럭 연결
 - ✓ 시스템 클럭과 별개의 클럭을 사용하므로 비동기(asynchronous)라고 이야기 함
 - ✓ ASSR 레지스터의 AS0 비트를 세트하여 외부 클럭 사용

CS02	CS01	CS00	분주비	오버플로 인터럽트 발생 간격
0	0	0	-	-
0	0	1	1(분주 없음)	1/128s
0	1	0	8	1/16s
0	1	1	32	1/4s
1	0	0	64	1/2s
1	0	1	128	1s
1	1	0	256	2s
1	1	1	1,024	8s



Timer/Counter-1, 3 (16-bit)

◆ 오버플로 인터럽트

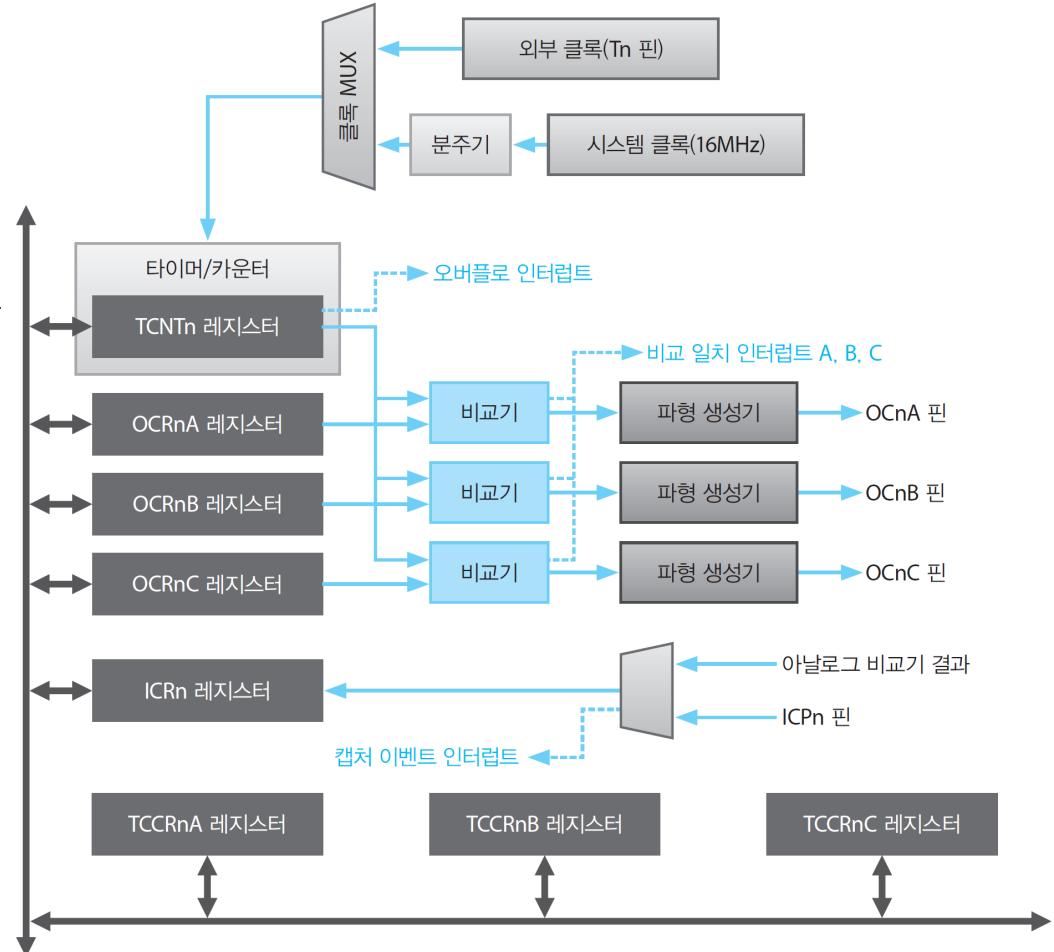
- $0 \sim 2^{16} - 1$ 까지 카운트 가능

◆ 3개의 비교 일치 인터럽트 사용 가능

- 3개의 서로 다른 비교 일치 값 설정을 위한 OCRnX 레지스터
- 3개의 파형 출력 핀

◆ 입력캡처

- 특정 사건이 발생한 경우 현재 카운터 값인 TCNTn 레지스터 값 저장



■ 16-Bit Timer/Counter

❖ 오버플로 인터럽트

- TCNT_n 레지스터 = TCNT_{nH} + TCNT_{nL}
 - ✓ 0 ~ 65,535까지 셀 수 있는 16비트 레지스터
 - ✓ 분주하지 않은 경우 4.096 ms 간격의 인터럽트 발생
 - ✓ 256으로 분주한 경우 약 1초 간격으로 인터럽트 발생

비트	15	14	13	12	11	10	9	8
TCNT _{nH}	TCNT _n [15:8]							
TCNT _{nL}	TCNT _n [7:0]							
비트	7	6	5	4	3	2	1	0
읽기/쓰기	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
초깃값	0	0	0	0	0	0	0	0

■ 16-Bit Timer/Counter

❖ Overflow Interrupt Example (1초에 1번 점멸)

```
#include <avr/io.h>
#include <avr/interrupt.h>
int state = 0;

ISR(TIMER1_OVF_vect){
    state = !state;                      // LED 상태 반전
    if(state) PORTB = 0x01;               // LED 켜기
    else PORTB = 0x00;                   // LED 끄기
}

int main(void){
    DDRB = 0x01;                         // PB0 핀을 출력으로 설정
    PORTB = 0x00;                         // LED는 끈 상태에서 시작

    TCCR1B |= (1 << CS12);             // 분주비를 256으로 설정

    TIMSK |= (1 << TOIE1);            // 오버플로 인터럽트 허용

    sei();                               // 전역적으로 인터럽트 허용

    while(1){ }
    return 0;
}
```

■ 16-Bit Timer/Counter

❖ TCCR1B Register – 분주비 설정

비트	7	6	5	4	3	2	1	0
	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10
읽기/쓰기	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
초깃값	0	0	0	0	0	0	0	0

CS12	CS11	CS10	설명
0	0	0	클록 소스 없음(타이머/카운터 정지)
0	0	1	분주비 1(분주 없음)
0	1	0	분주비 8
0	1	1	분주비 64
1	0	0	분주비 256
1	0	1	분주비 1,024
1	1	0	T1 핀의 외부 클록을 사용하며 하강 에지에서 동작함
1	1	1	T1 핀의 외부 클록을 사용하며 상승 에지에서 동작함

■ 16-Bit Timer/Counter

❖ TIMSK Register – Interrupt 활성화

비트	7	6	5	4	3	2	1	0
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0
읽기/쓰기	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
초깃값	0	0	0	0	0	0	0	0
비트	7	6	5	4	3	2	1	0
	-	-	TICIE3	OCIE3A	OCIE3B	TOIE3	OCIE3C	OCIE1C
읽기/쓰기	R	R	R/W	R/W	R/W	R/W	R/W	R/W
초깃값	0	0	0	0	0	0	0	0

- 5개 인터럽트에 대해 각각의 활성화 비트 존재
 - ✓ TOIE1 비트 : 오버플로 인터럽트 활성화
 - ✓ 비교 일치 인터럽트 활성화
 - OCIE1B : 비교 일치 B
 - OCIE1A : 비교 일치 A
 - ETIMSK 레지스터의 OCIE1C : 비교 일치 C
 - ✓ TICIE1 : 입력 캡처 인터럽트 활성화

16-Bit Timer/Counter

◆ 16-Bit Timer/Counter Interrupt Vectors

벡터 번호	인터럽트	벡터 이름	인터럽트 허용 비트	
12	입력 캡처 인터럽트	TIMER1_CAPT_vect	TICIE1	Timer/Counter1 Input Capture Interrupt Enable
13	비교 일치 A 인터럽트	TIMER1_COMPA_vect	OCIE1A	Timer/Counter 1 Output Compare Match A Interrupt Enable
14	비교 일치 B 인터럽트	TIMER1_COMPB_vect	OCIE1B	Timer/Counter 1 Output Compare Match B Interrupt Enable
15	오버플로 인터럽트	TIMER1_OVF_vect	TOIE1	Timer/Counter 1 Overflow Interrupt Enable
25	비교 일치 C 인터럽트	TIMER1_COMPC_vect	OCIE1C	Timer/Counter 1 Output Compare Match C Interrupt Enable
26	입력 캡처 인터럽트	TIMER3_CAPT_vect	TICIE3	Timer/Counter3 Input Capture Interrupt Enable
27	비교 일치 A 인터럽트	TIMER3_COMPA_vect	OCIE3A	Timer/Counter 3 Output Compare Match A Interrupt Enable
28	비교 일치 B 인터럽트	TIMER3_COMPB_vect	OCIE3B	Timer/Counter 3 Output Compare Match B Interrupt Enable
29	비교 일치 C 인터럽트	TIMER3_COMPC_vect	OCIE3C	Timer/Counter 3 Output Compare Match C Interrupt Enable
30	오버플로 인터럽트	TIMER3_OVF_vect	TOIE3	Timer/Counter 3 Overflow Interrupt Enable

■ 16-Bit Timer/Counter

❖ 비교일치 Interrupt Example - 1

비트	15	14	13	12	11	10	9	8
OXR1xH	OXR1x[15:8]							
OXR1xL	OXR1x[7:0]							
비트	7	6	5	4	3	2	1	0
읽기/쓰기	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
초깃값	0	0	0	0	0	0	0	0

```
#include <avr/io.h>
#include <avr/interrupt.h>

int state = 0;

ISR(TIMER1_COMPA_vect)
{
    state = !state;                      // LED 상태 반전
    if(state) PORTB = 0x01;                // LED 켜기
    else PORTB = 0x00;                    // LED 끄기

    TCNT1 = 0;                           // 자동으로 0으로 변하지 않는다
}
```

■ 16-Bit Timer/Counter

❖ 비교일치 Interrupt Example - 2

```
int main(void)
{
    DDRB = 0x01;                      // PB0 핀을 출력으로 설정
    PORTB = 0x00;                      // LED는 끈 상태에서 시작

    OCR1A = 0xFFFF;                    // 비교일치 값 설정

    TCCR1B |= (1 << CS12);          // 분주비를 256으로 설정

    TIMSK |= (1 << OCIE1A);         // 비교일치 A 인터럽트 허용

    sei();                            // 전역적으로 인터럽트 허용

    while(1){ }
    return 0;
}
```

■ 16-Bit Timer/Counter

❖ 서로 다른 점멸주기 Example - 1

- 3개의 비교 일치 Interrupt와 오버플로 Interrupt를 사용
 - ✓ 4개의 LED가 0.25초, 0.5초, 0.75초, 1초 간격으로 점멸

인터럽트	비교 일치 A	비교 일치 B	비교 일치 C	오버플로
TCNT1 값	0x3FFF	0x7FFF	0xBFFF	0xFFFF
인터럽트 간격	0.25초	0.5초	0.75초	1초
PB0				○
PB1		○		○
PB2	○	○	○	○

■ 16-Bit Timer/Counter

❖ 서로 다른 점멸주기 Example - 2

```
#include <avr/io.h>
#include <avr/interrupt.h>

int state0 = 0, state1 = 0, state2 = 0;

ISR(TIMER1_COMPA_vect)
{
    state2 = !state2;           // PB2 핀의 LED 제어
    if(state2) PORTB |= 0x04;
    else PORTB &= ~0x04;
}

ISR(TIMER1_COMPB_vect)
{
    state2 = !state2;           // PB2 핀의 LED 제어
    if(state2) PORTB |= 0x04;
    else PORTB &= ~0x04;

    state1 = !state1;           // PB1 핀의 LED 제어
    if(state1) PORTB |= 0x02;
    else PORTB &= ~0x02;
}
```

■ 16-Bit Timer/Counter

❖ 서로 다른 점멸주기 Example - 3

```
ISR(TIMER1_COMPC_vect)
{
    state2 = !state2;                      // PB2 핀의 LED 제어
    if(state2) PORTB |= 0x04;
    else PORTB &= ~0x04;
}

ISR(TIMER1_OVF_vect)
{
    state2 = !state2;                      // PB2 핀의 LED 제어
    if(state2) PORTB |= 0x04;
    else PORTB &= ~0x04;

    state1 = !state1;                      // PB2 핀의 LED 제어
    if(state1) PORTB |= 0x02;
    else PORTB &= ~0x02;

    state0 = !state0;                      // PB0 핀의 LED 제어
    if(state0) PORTB |= 0x01;
    else PORTB &= ~0x01;
}
```

■ 16-Bit Timer/Counter

❖ 서로 다른 점멸주기 Example - 4

```
int main(void)
{
    DDRB = 0x07;                      // PB0~PB2 핀을 출력으로 설정
    PORTB = 0x00;                      // LED는 끈 상태에서 시작

    OCR1A = 0x3FFF;                   // 비교일치 A (1/4초 간격)
    OCR1B = 0xFFFF;                   // 비교일치 B (2/4초 간격)
    OCR1C = 0xBFFF;                   // 비교일치 C (3/4초 간격)

    TCCR1B |= (1 << CS12);          // 분주비를 256으로 설정

    // 비교일치 A & B, 오버플로 인터럽트 허용
    TIMSK |= (1 << OCIE1A) | (1 << OCIE1B) | (1 << TOIE1);
    // 비교일치 C 인터럽트 허용
    ETIMSK |= (1 << OCIE1C);

    sei();                            // 전역적으로 인터럽트 허용

    while(1){ }
    return 0;
}
```

16-Bit Timer/Counter

❖ 파형 생성 Example - 1

- 3개의 비교 일치 각각 파형이 출력되는 핀이 서로 다름
 - ✓ 1번 타이머/카운터의 비교 일치 C와 2번 타이머/카운터의 비교 일치 시 파형이 출력되는 핀은 동일함

타이머/카운터	파형 출력 핀	ATmega128 핀 번호
0	OC0	PB4
	OC1A	PB5
1	OC1B	PB6
	OC1C	PB7
2	OC2	PB7
	OC3A	PE3
3	OC3B	PE4
	OC3C	PE5

■ 16-Bit Timer/Counter

❖ 파형 생성 Example - 2

```
#include <avr/io.h>
#include <avr/interrupt.h>
int state = 0;

ISR(TIMER1_COMPA_vect){
    TCNT1 = 0;                                // 자동으로 0으로 변하지 않는다
}

int main(void){
    DDRB = 0x20;                             // PB5 핀을 출력으로 설정
    PORTB = 0x00;                            // LED는 끈 상태에서 시작
    OCR1A = 0xFFFF;                          // 비교일치 값 설정
    TCCR1B |= (1 << CS12);                // 분주비를 256으로 설정

    // 비교일치 인터럽트 발생 시 OC1A 핀의 출력을 반전
    TCCR1A |= (1 << COM1A0);
    TIMSK |= (1 << OCIE1A);                // 비교일치 A 인터럽트 허용

    sei();                                    // 전역적으로 인터럽트 허용

    while(1){ }
    return 0;
}
```

■ 16-Bit Timer/Counter

❖ 파형 생성 제어 – TCCR1A, TCCR1B Register

비트	7	6	5	4	3	2	1	0
	COM1A1	COM1A0	COM1B1	COM1B0	COM1C1	COM1C0	WGM11	WGM10
읽기/쓰기	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
초깃값	0	0	0	0	0	0	0	0

비트	7	6	5	4	3	2	1	0
	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10
읽기/쓰기	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
초깃값	0	0	0	0	0	0	0	0

- 7번부터 2번까지 2개 비트가 각 비교 일치 시 파형 생성을 제어
- 1번과 0번 WGM1n 비트는 TCCR1B 레지스터의 WGM1n 비트와 함께 파형 생성 모드 결정

■ 16-Bit Timer/Counter

❖ 파형 생성 모드

- PWM 관련 내용은 추후 학습
- 0번 모드 : 정상 모드
 - ✓ 비교 일치 발생 시 TCNT_n 레지스터가 리셋되지 않음
- 4번 모드 : CTC 모드
 - ✓ 비교 일치 발생 시 TCNT_n 레지스터가 리셋됨
 - ✓ 비교 일치 값은 OCR1A 레지스터에 저장
- 12번 모드 : CTC 모드
 - ✓ 비교 일치 발생 시 TCNT_n 레지스터가 리셋됨
 - ✓ 비교 일치 값은 ICR1 레지스터에 저장

모드 번호	TCCR1B 레지스터		TCCR1A 레지스터		타이머/카운터 모드	TOP
	WGM13	WGM12 (CTC)	WGM11 (PWM)	WGM10 (PWM)		
0	0	0	0	0	정상	0xFFFF
1	0	0	0	1	8비트 위상 교정 PWM	0x00FF
2	0	0	1	0	9비트 위상 교정 PWM	0x01FF
3	0	0	1	1	10비트 위상 교정 PWM	0x03FF
4	0	1	0	0	CTC	OCR1A
5	0	1	0	1	8비트 고속 PWM	0x00FF
6	0	1	1	0	9비트 고속 PWM	0x01FF
7	0	1	1	1	10비트 고속 PWM	0x03FF
8	1	0	0	0	위상 및 주파수 교정 PWM	ICR1
9	1	0	0	1	위상 및 주파수 교정 PWM	OCR1A
10	1	0	1	0	위상 교정 PWM	ICR1
11	1	0	1	1	위상 교정 PWM	OCR1A
12	1	1	0	0	CTC	ICR1
13	1	1	0	1	-	-
14	1	1	1	0	고속 PWM	ICR1
15	1	1	1	1	고속 PWM	OCR1A

■ 16-Bit Timer/Counter

❖ 파형 생성 동작

- 정상 모드와 CTC 모드에서의 동작은 동일하게 TCCR1A 레지스터의 COM1An 비트에 의해 결정

COM1A1	COM1A0	설명
0	0	OC1A 핀으로 데이터가 출력되지 않으며, OC1A 핀은 일반적인 범용 입출력 핀으로 동작한다.
0	1	비교 일치가 발생하면 OC1A 핀의 출력은 반전된다.
1	0	비교 일치가 발생하면 OC1A 핀의 출력은 LOW 값으로 바뀐다(clear).
1	1	비교 일치가 발생하면 OC1A 핀의 출력은 HIGH 값으로 바뀐다(set).

❖ TCCR1C Register

- 비교 일치가 발생한 것과 동일한 효과를 얻기 위해 FOC1x(Force Out Compare) 비트 세트를 통해 강제로 매칭 출력을 설정할 수 있음
 - ✓ 비교 매칭을 강제로 실행해도 OCFnx 플래그가 설정되거나 타이머가 다시 로드/클리어 되지는 않지만, OCnx 핀은 실제 비교 매칭이 발생한 것처럼 업데이트 됨

비트	7	6	5	4	3	2	1	0
	FOC1A	FOC1B	FOC1C	-	-	-	-	-
읽기/쓰기	W	W	W	R	R	R	R	R
초깃값	0	0	0	0	0	0	0	0



Q & A
