뻔뻔한 AVR 프로그래밍

The 4th Lecture

유 명 환 (✉ yoo@netplug.co.kr)

INDEX

1 시간(Time)에 대한 정의

시간

절대적인 시간

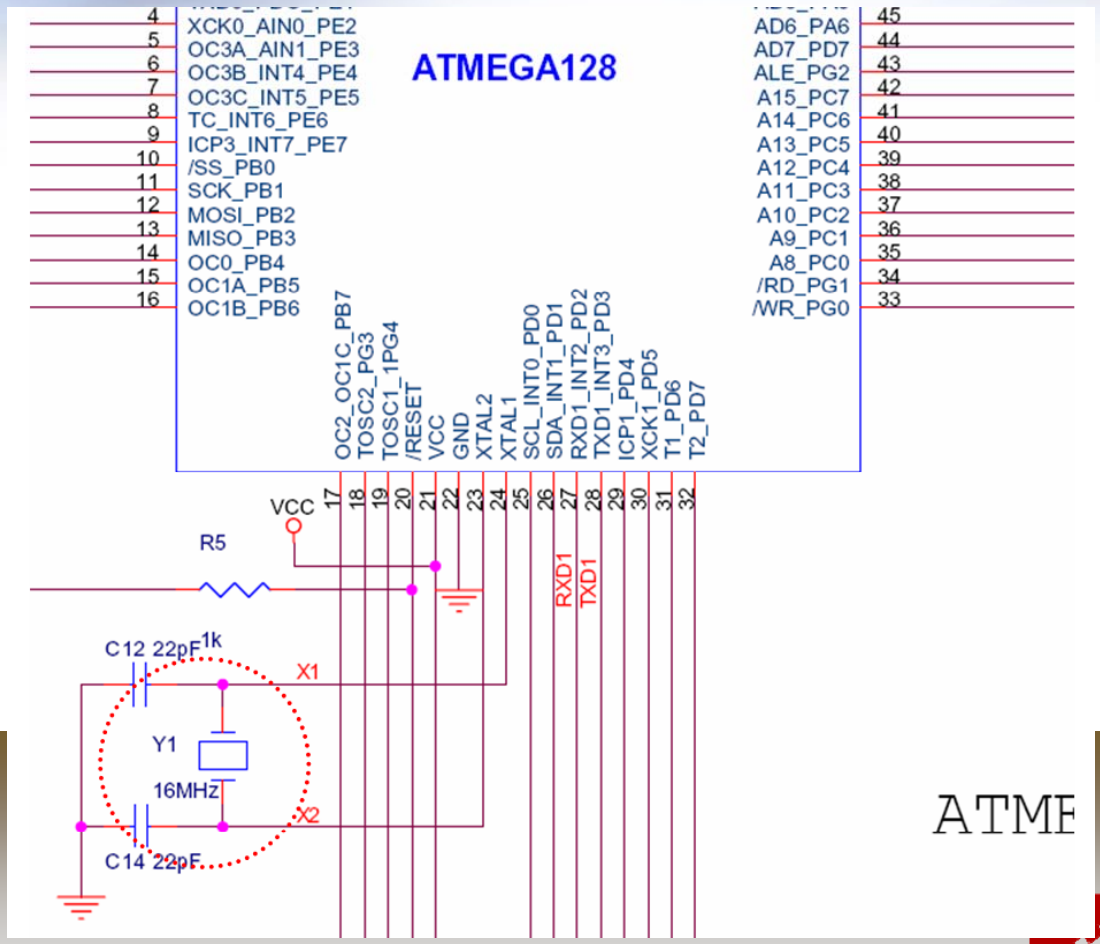
- 주변 환경과 상관없이 독립적으로 제어되는 시간
- 예) RTC : Real-Time Clock

상대적인 시간

- 주변 환경과 맞물려 제어되어야 하는 시간
- “지금으로부터 ~ 후”
- 예) 1초마다 한번씩 LED가 On/Off

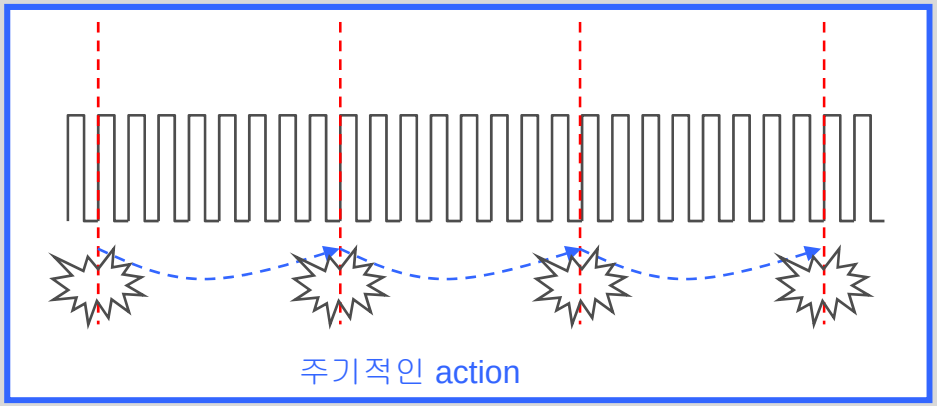
2 왜 타이머(Timer)와 카운터(Counter)인가?

1. 3분 뒤에 라면을 먹는 것이 목표다.
2. 3분이란 시간을 측정해야 한다.
3. 3분이란 약속된 시간이 지난 후 반드시 이를 알려줘야 한다.
4. 3분이란 시간을 누군가(타이머) 알려주면 이제까지 하던 일을 멈추고 1번의 목표를 실행한다.



2 왜 타이머(Timer)와 카운터(Counter)인가?

Situation



(주파수의) 진동 수 = $\frac{1}{\text{(시간의) 주기}}$

(Hz)

(second)

Definition

- 원하는 주기를 설정 → Timer
- 클럭의 입력 횟수 카운팅 → Counter

2 왜 타이머(Timer)와 카운터(Counter)인가?

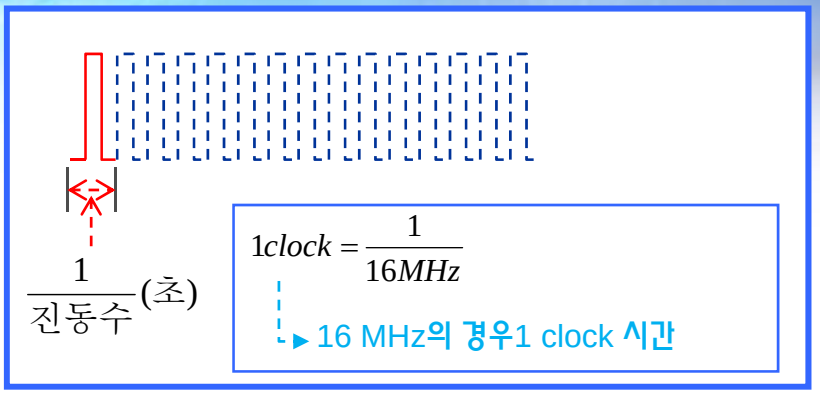
공식 (1)

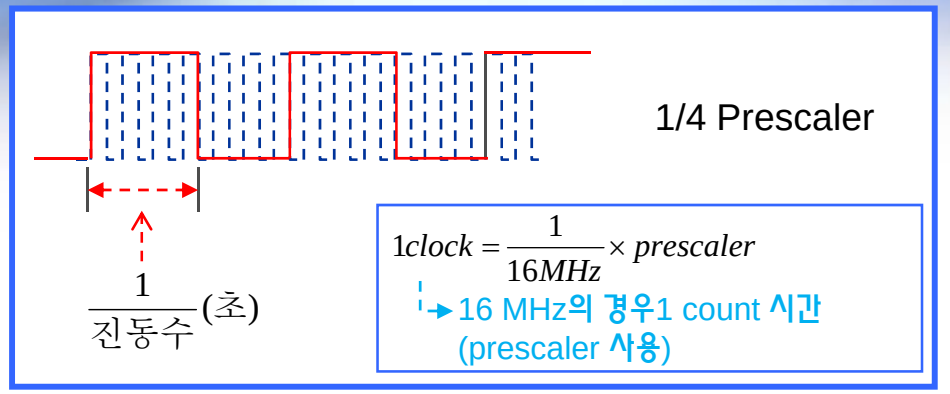
$$\underset{①}{1ms} = \underset{②}{\left[\frac{1}{16M}\right]} \times \underset{③}{n}$$

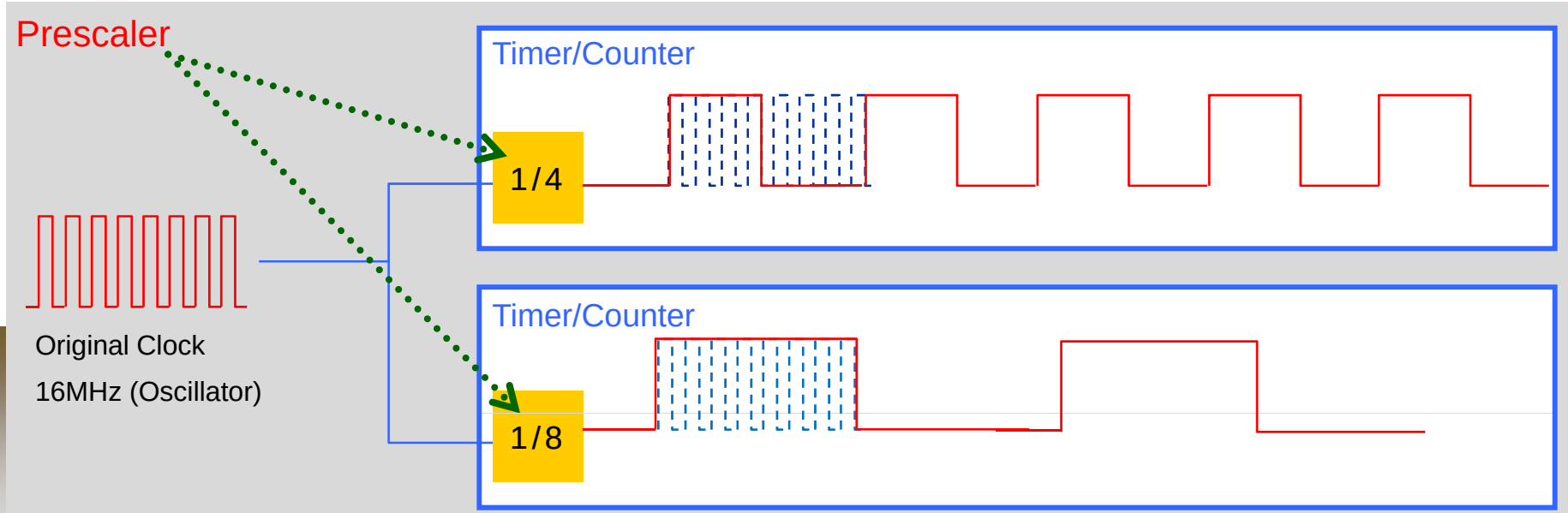
공식 (2)

$$\underset{①}{1ms} = \underset{③}{\left\{\underset{②}{\left[\frac{1}{16M}\right]} \times Prescaler\right\}} \times \underset{④}{n}$$

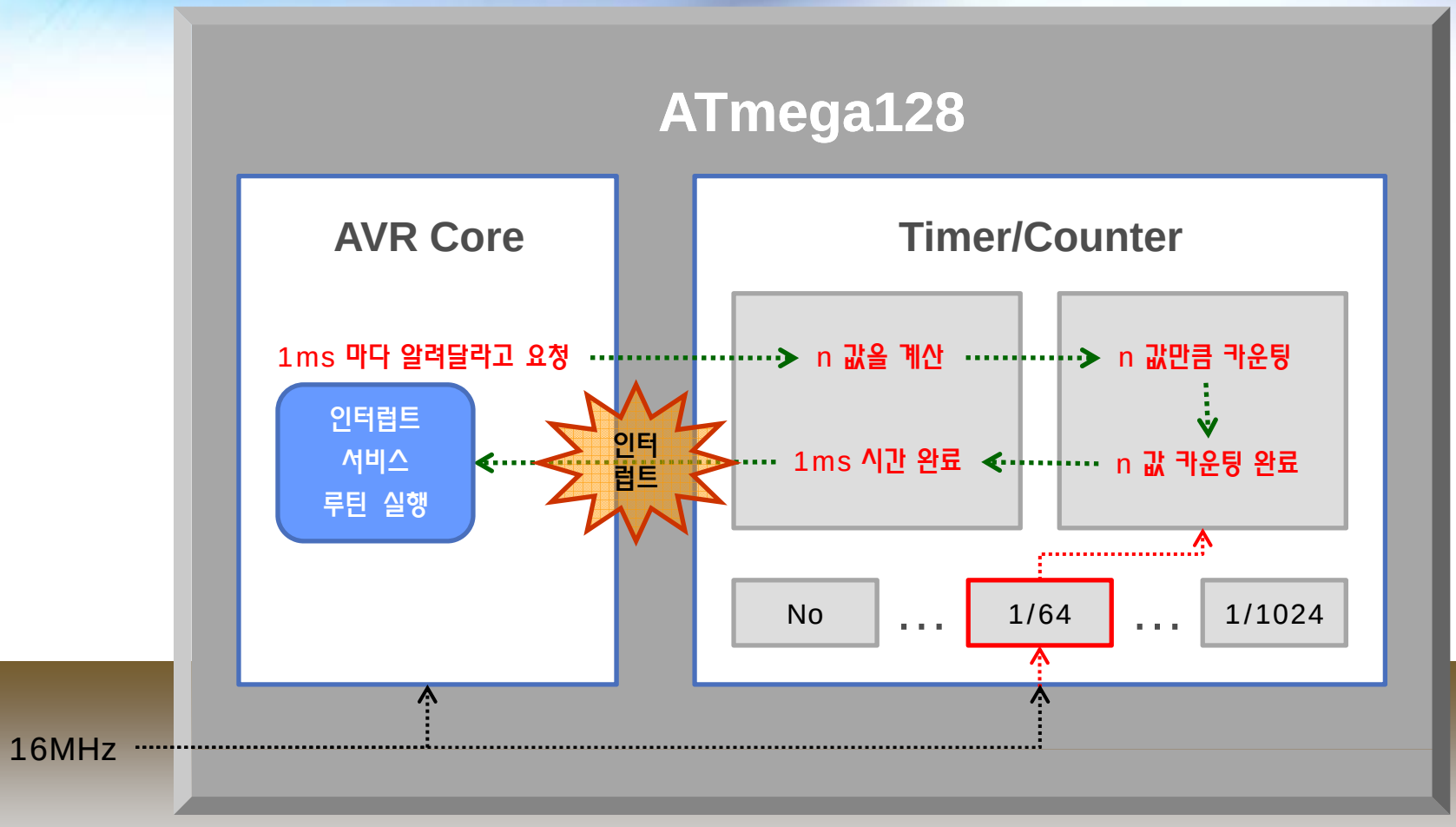
2 왜 타이머(Timer)와 카운터(Counter)인가?





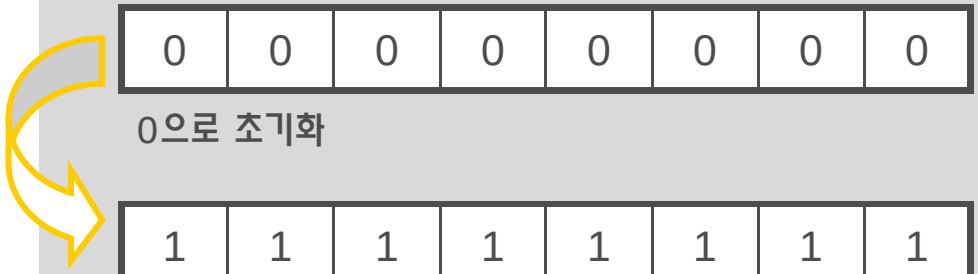


3 ATmega128 타이머/카운터 동작 구조

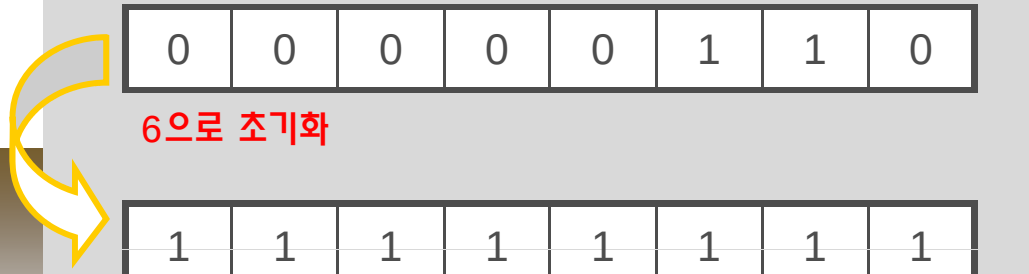


3 ATmega128 타이머/카운터 동작 구조

Overflow Interrupt

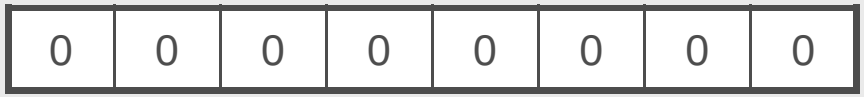


- 8bit register 한 개만 필요
- Overflow는 255에서 발생
- 0~255 ➔ 256 counting



- n = 250 일 때 인터럽트 발생
- 0이 아닌 6으로 초기화
- 6~255 ➔ 250 counting

Output Compare Match Interrupt



0으로 초기화

249 (n – 1) 로 초기화

- 두 개의 register 필요
  - counter register (횟수 기록)
  - compare register (n 값 기록)
- counter register가 증가되다 compare register와 그 값이 서로 같으면 interrupt발생

counter register 값을 증가시킨 후 compare register와 비교

- counter register를 증가하며 compare register와 비교

(counter register == compare register) ➔ interrupt 발생

- counter register와 compare register를 비교하여 값이 서로 같을 때 interrupt 발생
- 250 counting

4 ATmega128 타이머/카운터 관련 레지스터

TCCR0 :
Timer/Counter Control Register

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	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	

Register	설명	실제 구현 예

Bit6, 3 – WGM01:0 Waveform Generation Mode	• 출력할 파형 설정
	• Overflow interrupt 일 때
	– Normal mode로 설정
	– Interrupt가 발생할 때 마다 반드시 TCNT0 값을 초기화 해야 함
	
	• Compare match interrupt 일 때
	– CTC mode로 설정
	– TOP을 OCR0로 사용 -> OCR0 값에서 인터럽트 발생
	– Interrupt가 발생하면 자동적으로 TCNT0 값이 “0”으로 초기화 됨
		`TCCR0 =`
		`(0 << WGM00) | (1 << WGM01);`
		`// CTC mode`
		
		`TCCR0 =`
		`(0 << WGM00) | (0 << WGM01);`
		`// Normal mode`

Mode	WGM01(1) (CTC0)	WGM00(1) (PWM0)	Timer/Counter Mode of Operation	TOP	Update of OCR0 at	TOV0 Flag Set on
0	0	0	Normal	0xFF	Immediate	MAX
1	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	1	0	CTC	OCR0	Immediate	MAX
3	1	1	Fast PWM	0xFF	TOP	MAX

4 ATmega128 타이머/카운터 관련 레지스터

Bit 2:0 – CS02: Clock select

• Prescaler 설정

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	clk_{T0S}/(No prescaling)
0	1	0	clk_{T0S}/8 (From prescaler)
0	1	1	clk_{T0S}/32 (From prescaler)
1	0	0	clk_{T0S}/64 (From prescaler)
1	0	1	clk_{T0S}/128 (From prescaler)
1	1	0	clk_{T0S}/256 (From prescaler)
1	1	1	clk_{T0S}/1024 (From prescaler)

```
TCCR0 =
(1<<CS02) | (0<<CS01) | (0<<CS00);
// 1/64 Prescaler
```

4 ATmega128 타이머/카운터 관련 레지스터

TCNT0 :
Timer/Counter Register

Bit	7	6	5	4	3	2	1	0	
	TCNT0[7:0]								TCNT0
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	

Register	설명	실제 구현 예

TCNT0	• Prescaler가 적용된 클럭 주기마다 1씩 증가하는 8비트 값
	• 하드웨어(= 카운터)에 의해 증가됨
	• Overflow Interrupt 발생 시점 = 256 – n
	• Output Compare Match Interrupt 발생 시점 = 0
		`TCNT0 = 256 – (CPU_CLOCK/TICKS_PER_SEC/64);`
		
		`// CPU_CLOCK : 16000000`
		`// TICKS_PER_SEC : 1000`
		`// 64 : 1/64 Prescaler`
		`// (CPU_CLOCK/TICKS_PER_SEC/64) == n`

4 ATmega128 타이머/카운터 관련 레지스터

OCR0 :

Output Compare Register

Bit	7	6	5	4	3	2	1	0	
	OCR0[7:0]								OCR0
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	

Register	설명	실제 구현 예
OCR0	• TCNT0와 비교가 되는 8비트 값 • Output Compare Match Interrupt 에서 사용 • 초기화 값 = n - 1	`OCR0 = (CPU_CLOCK/TICKS_PER_SEC/64) - 1;`

4 ATmega128 타이머/카운터 관련 레지스터

TIMSK :

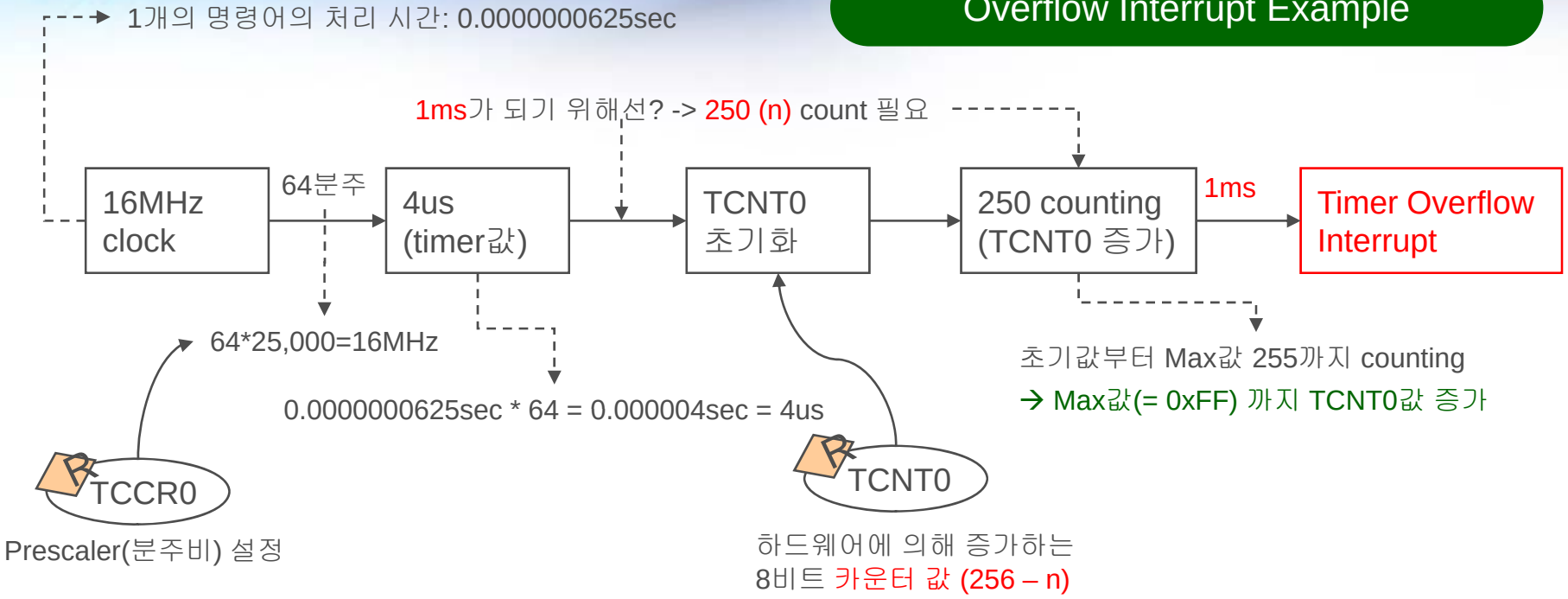
Timer/Counter Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
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	

Register	설명	실제 구현 예
Bit 1 – OCIE0: Timer/Counter0 Output Compare Match Interrupt Enable	• Output Compare Match Interrupt 활성화 • OCIE0와 SREG_I 비트가 1로 설정되어 있어야만 Output Compare Match Interrupt가 활성화 됨	`TIMSK = (1<<OCIE0) | (0<<TOIE0);` `// Output Compare Match Interrupt`
Bit 0 – TOIE0: Timer/Counter0 Overflow Interrupt Enable	• Overflow Interrupt 활성화 • TOIE0와 SREG_I 비트가 1로 설정되어 있어야만 Overflow Interrupt가 활성화 됨	`TIMSK = (0<<OCIE0) | (1<<TOIE0);` `// Overflow Interrupt`

4 ATmega128 타이머/카운터 관련 레지스터

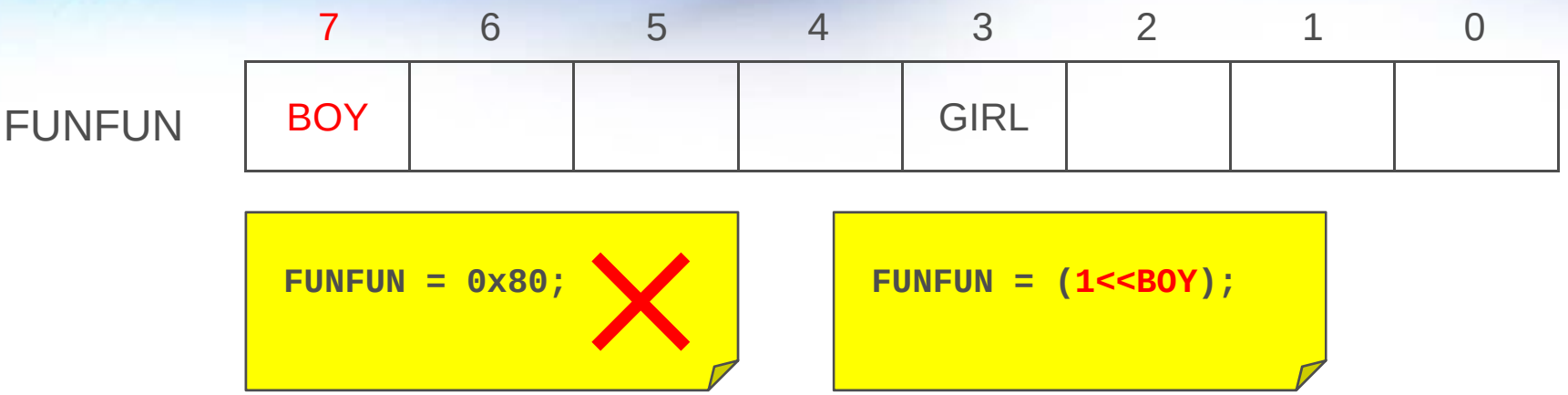
Overflow Interrupt Example



TIMSK

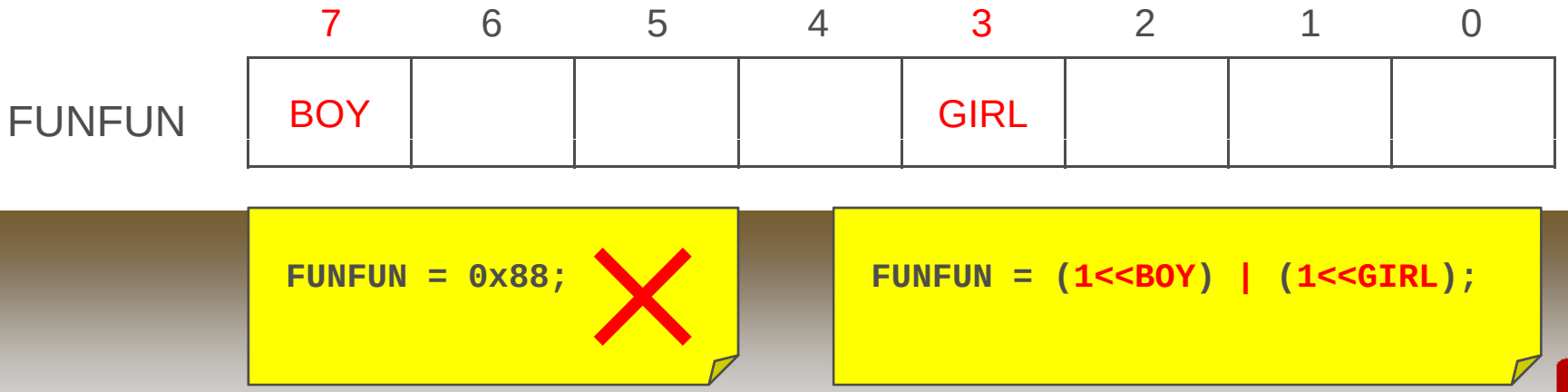
Overflow interrupt를 활성화 시키기 위해
TIMSK 레지스터의 TOIE0 비트를 1로 설정

5 뻔뻔한 노하우 : 레지스터 비트 설정 방법




```
FUNFUN = 0x80;
```

```
FUNFUN = (1<<BOY);
```




```
FUNFUN = 0x88;
```

```
FUNFUN = (1<<BOY) | (1<<GIRL);
```

6 ATmega128 타이머/카운터 실습

물리적 연결

DK128-MAIN과 DK128-EXT 연결

초기화 & 설정 루틴

Timer/Counter 관련 레지스터 설정

초기화 : n 값 계산
설정 : TCCR0 설정 -> Normal mode, 1/64 Prescaler
활성화 : TIMSK 설정 -> Overflow Interrupt 활성화

Overflow Interrupt 발생

서비스 루틴

Timer/Counter 인터럽트 서비스 루틴
(Interrupt Vector Table사용)

```
SIGNAL(SIG_OVERFLOW0)
```

6 ATmega128 타이머/카운터 실습

ATmega128 Data Sheet

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

C:\AvrEdit\WinAvr\avr\include\avr\iom128.h

```
/* Interrupt vectors */

#define SIG_INTERRUPT0          _VECTOR(1)
#define SIG_INTERRUPT1          _VECTOR(2)
#define SIG_INTERRUPT2          _VECTOR(3)
#define SIG_INTERRUPT3          _VECTOR(4)
#define SIG_INTERRUPT4          _VECTOR(5)
#define SIG_INTERRUPT5          _VECTOR(6)
#define SIG_INTERRUPT6          _VECTOR(7)
#define SIG_INTERRUPT7          _VECTOR(8)
#define SIG_OUTPUT_COMPARE2     _VECTOR(9)
#define SIG_OVERFLOW2           _VECTOR(10)
#define SIG_INPUT_CAPTURE1      _VECTOR(11)
#define SIG_OUTPUT_COMPARE1A    _VECTOR(12)
#define SIG_OUTPUT_COMPARE1B    _VECTOR(13)
#define SIG_OVERFLOW1           _VECTOR(14)
#define SIG_OUTPUT_COMPARE0     _VECTOR(15)
#define SIG_OVERFLOW0           _VECTOR(16)
```