AVR – HW4

임베디드스쿨1기
Lv1과정
2020. 10. 09
박성환

1. Review-AVR UART

```c
unsigned char UART_receive(void)
{
    while(!(UCSR0A & (1<<RXC0))); //wait for data to be received
    return UDR0;    //get and return received data from buffer
}
```

RXC0 이 1이면
즉 UDR 레지스터에 수신될
데이터가 전달 완료되면 set됨

```c
void UART_transmit(unsigned char data)
{
    while(!(UCSR0A & (1<<UDRE0)));  //wait for empty transmit buffer
    UDR0 = data;    //put data into buffer, sends the data
}
```

UDR의 데이터가 전부 전송되어
empty 상태일 때 set이 됨

```c
void UART_string_transmit(char *string)
{
    while(*string != '\0')
    {
        UART_transmit(*string);
        string++;
    }
}
```

Receive, transmit 함수 활용하여
문자열 보내는 함수 생성

```c
void UART_PRINT(char *name, long val)
{
    char debug_buffer[UART_BUFLEN] = {'\0'};

    UART_string_transmit(name);
    UART_string_transmit(" = ");

    itoa(( val), debug_buffer, UART_BUFLEN); //int to ascii, val -> debug_buffer

    UART_string_transmit(debug_buffer);
    UART_string_transmit("\n");
}
```

문자열과 값을 보낼수 있도록
Printf 처럼 사용하도록 만든
함수

1. Review-itoa/sprintf

1) itoa() 함수

- stdlib.h를 선언한다. #include <stdlib.h>
- **사용법)** itoa(변환할 정수, 문자열로 변환한 값을 받을 문자열 변수, 진법 선택)
  itoa(int cnt, char * buffer,int n)
- **ex)**

```
int cnt=1;
char buffer[10];

itoa(cnt,buffer,10); // 숫자 1을 문자열로 변환하여 buffer 변수에 저장. 이 때 숫자는 10진수로 변환한다.
```

Itoa함수는 표준함수가 아니라 win32api라서 리눅스에서는 사용 불가능

참고로 atoa는 표준함수로 어디든 사용 가능

2) sprintf() 함수

- stdio.h를 선언한다. #include <stdio.h>
- **사용법)** sprintf(변환한 값을 받을 문자열 변수,변환할 정수)
- **ex)**

```
int cnt=1;
char buffer[10];

sprintf(buffer,"%d",cnt);  // buffer에는 1이 문자열로 변환되어 저장됨.
```

- **ex2)**

```
int cnt=1;
char buffer[10];

sprintf(buffer,"_%d",cnt);  // buffer에는 _1이 문자열로 저장됨.
```

Sprintf에 "%d" 대신 "0x%x" 하면 16진수로 변환됨(0x는 16진수 구분위함

2. Timer/Counter

1. 타이머/카운터란?

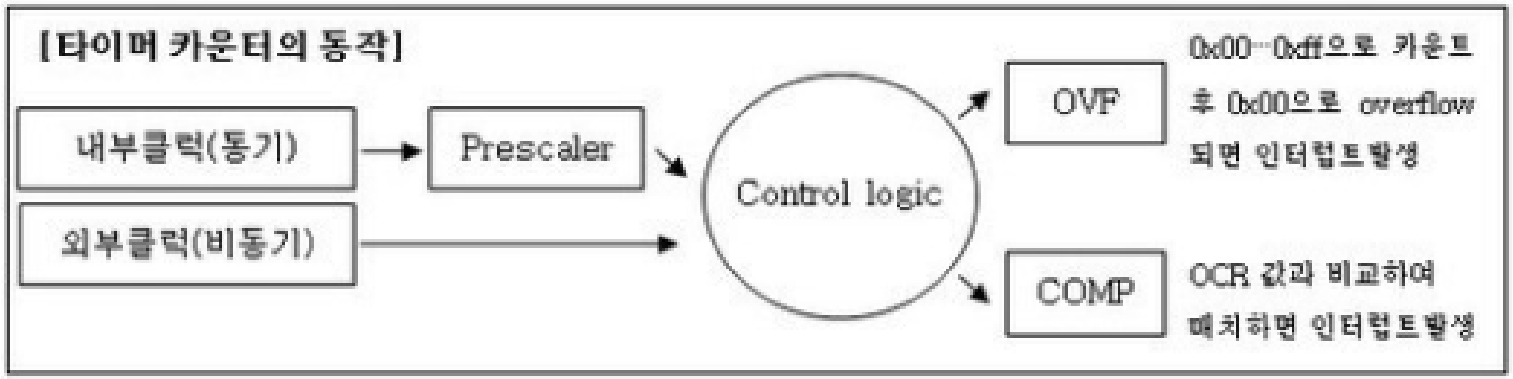
타이머: 내부 클럭을 이용하여 일정시간 간격의 펄스를 만들어 내거나 일정시간 경과 후에 인터럽트를 발생시키는 기능을 말함

카운터: 외부핀(TOSC1, TOSC2, T1, T2, T3)을 통해서 들어오는 펄스를 계수하여 Event Counter로서 동작되는 것을 말함

입력소스가 다르긴 한데 사용목적과 결과가 같은 이유로 사용되기 때문에 혼용되어 불린다.

타이머 – 내부클럭(빠름/분주가능:범위 내에서 클럭선택가능) – 동기모드

카운터 – 외부클럭(느림/분주불가능:외부클럭 그대로 사용) – 비동기모드



2. Timer/Counter

Normal모드
파형을 출력하지 않음

PWM모드
PWM모드는 파형을 일정한 주기로 출력하는 것입니다. 모터 속도 조절, LED밝기 조절에 사용됩니다.

CTC모드 (Clear Timer on Compare Match)
CTC모드는 원하는 주파수를 발생시키는 것입니다. 시계, 멜로디 발생에 사용됩니다.

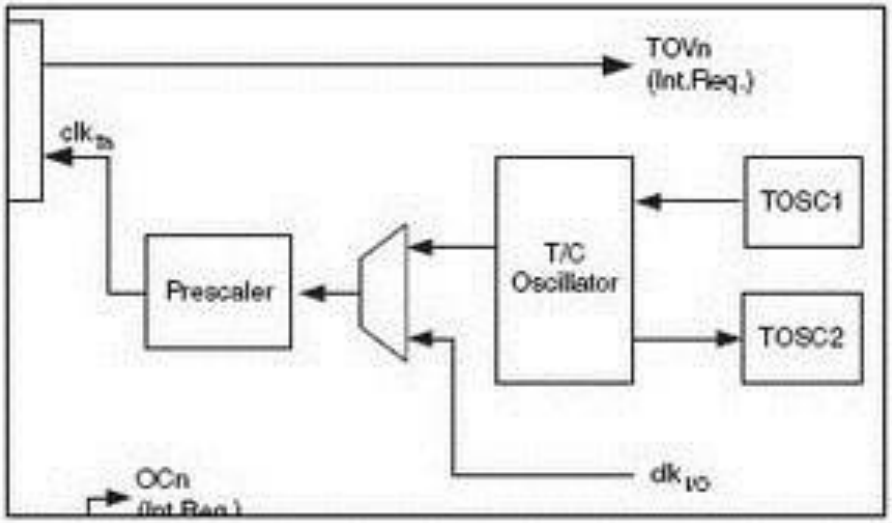
OVF 인터럽트(OverFlow)
TCNT가 가득 차면 OVF 인터럽트가 발생됩니다.

OCM 인터럽트(Output Compare Match)
TCNT와 특정값이 같으면 인터럽트가 발생됩니다.

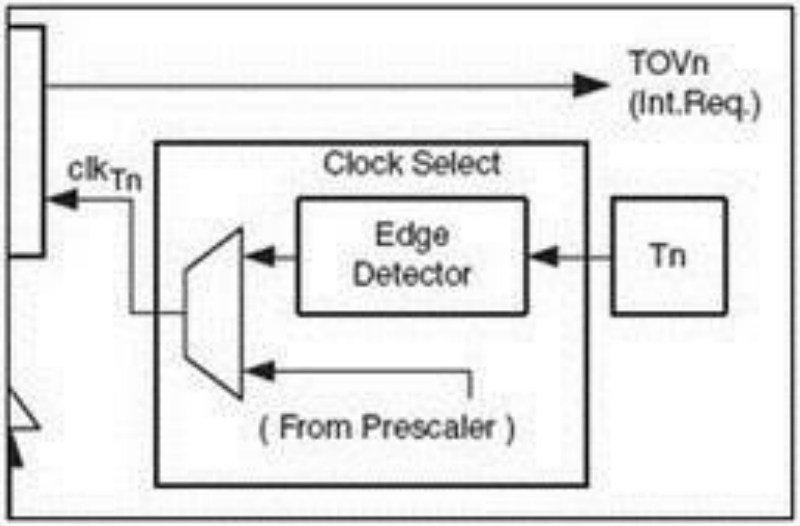
2-1. Timer/Counter0/2

1. 타이머/카운트 0/2

타이머/카운터0과2는8비트 구조로서 OVERFLOW, PWM 비슷한 기능을 가지고 제어방식도 비슷하다. 차이점이 있다면 타이머/카운터0 은32.768kHz의 크리스탈을 접속하는TOSC1 및TOSC2 단자를 가지고 있어서RTC의 기능을 갖도록 할 수 있으며 다른 타이머/카운터와는 틀리게 내부 클럭을 사용하든 외부의 클럭을 사용하든 모두 프리스케일러의 분주기능을 사용할 수 있다는 것이다.



타이머/카운터 0의 클럭 입력



타이머/카운터 2의 클럭 입력

2. Timer/Counter

- 8비트로 할 것인가? 16비트로 할 것인가?

- 내부 클럭을 입력으로 받을 것인가? 외부 신호 또는 클럭으로 입력받을 것인가?

- 만일 내부 입력이라면 분주비를 할 것인가? 말 것인가? 만일 하게 되면 몇 분주로 할 것인가?

- 외부 입력이라면 어떻게 받을 것인가? 클럭은 얼마로 할 것인가?

- 다양한 인터럽트 중에 어떤 것을 사용해야 할 것인가?

- 다양한 동작 모드 중에 어떤 모드를 사용할 것인가? 각 모드는 어떤 특징이 있는가?

- 위에 것들을 어떻게 설정해야 하는가....

2-1. Timer/Counter0/2

2. 타이머/카운트 0/2 특징(1)

- 8비트(0x00 ~ 0xff) 구조
- 10비트(0x000 ~ 0x3ff) 프리스케일러
- Overflow와Output Compare Match 인터럽트
- Output Compare Match 할 때 타이머가 클리어 된다.(CTC모드)
- 타이머/카운터0의 경우 다른 타이머/카운터와는 색다른 기능을 가진다.

> RTC기능
> 타이머와 카운터 모두 프리스케일러를 사용
> I/O 클럭과 독립된 외부32.768kHz 크리스탈에 동작가능

3. 타이머/카운트 0/2 특징(2)

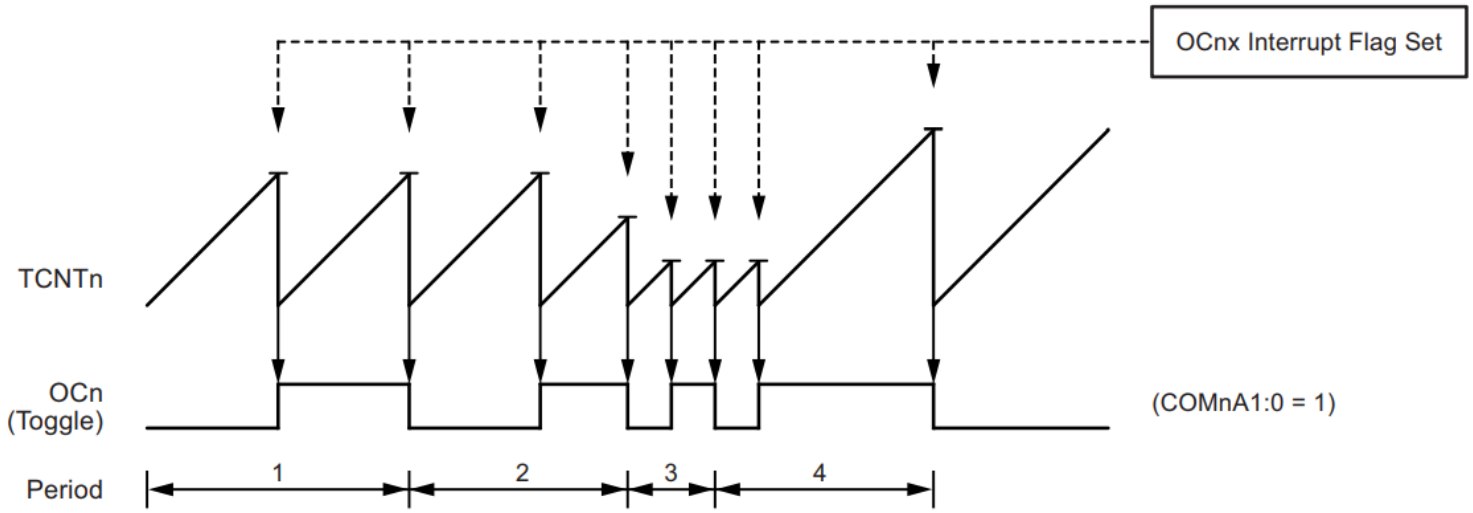
- 내/외부 클럭 중 하나를 선택하여 기준 클럭으로 삼는다.
- 타이머/카운터0, 2은0x00~0Xff까지Count하여Overflow되면OVF 인터럽트가 걸린다.
- 타이머/카운터0, 2은0x00~0Xff까지Count하다가 지속적으로TCNT값과OCR값을 비교하여 같으면COMP 인터럽트가 걸린다.

※용어설명
- BOTTOM : 카운터가 가질 수 있는 최소값0x00을 나타낸다.
- MAX : 카운터가 가질 수 있는 최대값0xff를 나타낸다.
- TOP : 각 동작모드에 따라 카운터가 실제로 도달하는 최대값을 나타낸다. (CTC모드의 경우TOP=OCR)
- COUNT : TCNT0의1씩 증가 또는 감소하는 것
- DIRECTION : COUNT의 증감/ 감소 설정
- CLEAR : Clear TCNTO
- CLKTO: 타이머/카운터 클럭

2-1. Timer/Counter0/2

① Normal Mode
- 일반적인 타이머 오버플로우 인터럽트가 필요할 때 사용
- 상향카운터
- 0x00 ~ 0xFF 계수동작 반복
- 카운트 도중Clear 없음
- 오버플로우(OVF) 인터럽트(MAX=0xFF값일 때 발생)
- 비교매치(COMP) 인터럽트(파형을 예상하지 못하기 때문에 추천하지 않음)

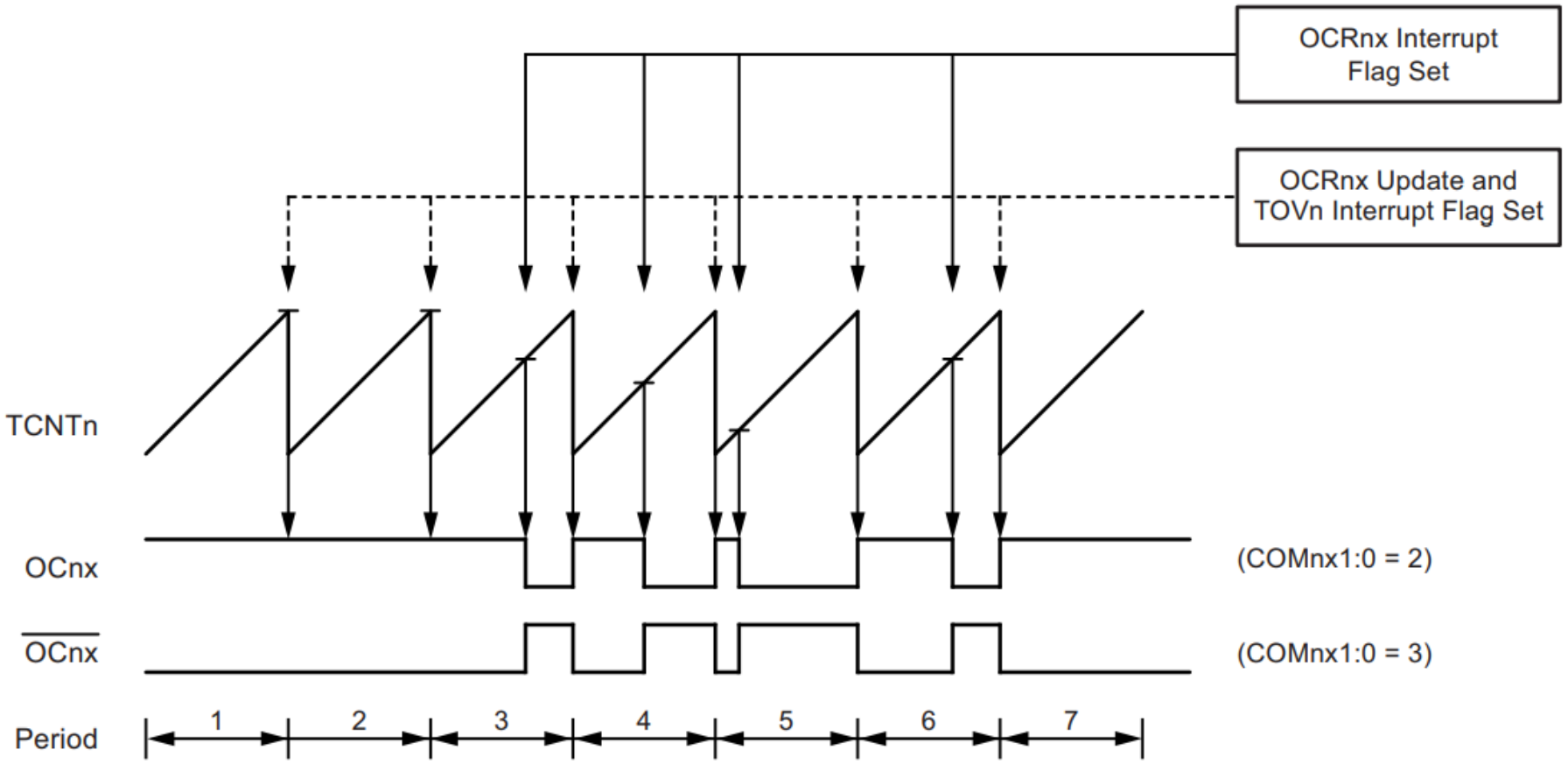
② CTC Mode(Compare Timer on Compare Match Mode)
- 주파수 분주 기능으로 주로 사용
- 상향카운터
- 0x00 ~ OCR0 계수 동작 반복
- OCR0값과TCNT0값이 같으면 카운트 도중Clear
- 오버플로우(OVF) 인터럽트
- (MAX=OCR0값일 때 발생, COMP인터럽트와 동일하게 작동되기 때문에 추천하지 않음)
- 비교매치(COMP) 인터럽트



2-1. Timer/Counter0/2

③ FAST PWM

- 높은 주파수PWM 파형발생이 필요할 때 사용
- 상향카운터(Single-Slope Operation)
- 0x00 ~ 0xFF 계수 동작 반복
- TCNT0과OCR0의Compare Match되면OC0에LOW출력(COM0 1:0 = 2)
- 0xFF →0x00 오버플로우 되면OC0에HIGH출력(COM0 1:0 = 2)

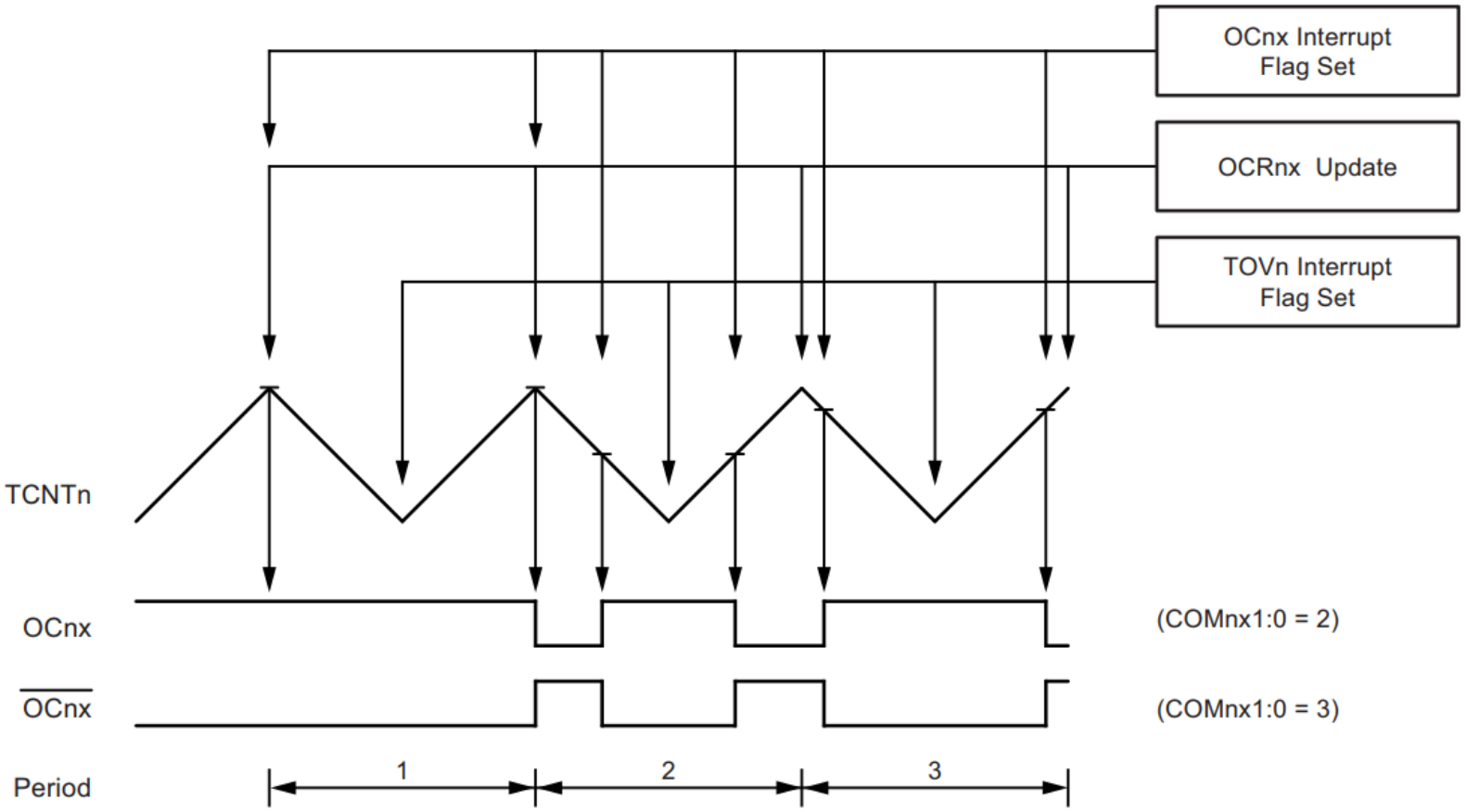


포기하면 얻는 건 아무것도 없다.

2-1. Timer/Counter0/2

④ Phase Correct PWM

- 높은 분해능의PWM출력 파형을 발생하는데 사용
- 상향카운터0x00 →0xFF
- 하향카운터0xFF →0x00
- 0x00 ~ 0xFF ~ 0x00 계수 동작 반복
- 상향카운터 비교매치 →OC0 = 0 출력(COM0 1:0 = 2)
- 하향카운터 비교매치 →OC0 = 1 출력(COM0 1:0 = 2)



포기하면 얻는 건 아무것도 없다.

감사합니다.