3강. Timer/Counter

박 원 엽
010.5451.0113

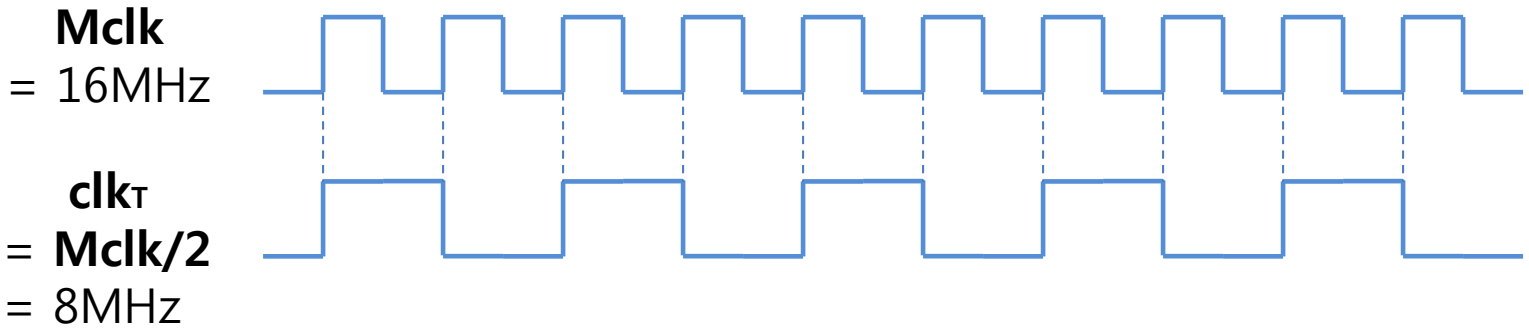
목 차

- **Timer/Counter란?**

  - 시간을 측정하기 위한 Peripheral 중 하나로써, 말 그대로 시간을 측정하는데 쓰이거나 PWM(Pulse Width Modulation)신호를 만드는데 사용된다.

- **Timer/Counter의 동작 원리?**

  - 시간 측정은 클럭(clkT)의 개수를 카운팅함으로써 가능해진다.
  - 예를 들어, clkT 가 100Hz의 주기로 펄스가 발생한다면 clkT 를 100번 카운팅하면 1초가 지난것이다.
  - 여기서 clkT 를 Timer/Counter의 clock source라고 하는데, 이 clkT 는 마스터클럭(Mclk)인 16MHz 크리스탈과 관계가 있다.
  - 타이머/카운터를 사용하기 위해선 clock source를 설정해주어야 하는데, 대개 마스터 클럭(Mclk)인 16MHz를 얼마만큼의 비율로 나누어서 사용한다. (외부트리거도 가능)
  - 예를 들어, clkT 를 Mclk의 1/2로 설정한다고 하면 clkT 는 8MHz의 클럭이 된다. 이때 2분주(Prescale) 라는 표현을 쓴다.



- **Timer/Counter의 동작 원리?**

  - clock source를 설정해주면 타이머 클럭(clkT) 속도에 동기화 되어 타이머가 동작하기 시작하는데, 타이머 클럭(clkT)을 하나씩 카운팅하며 이때 카운팅 된 숫자가 TCNTn 레지스터에 자동으로 저장된다. (n=0~3)

  - 즉, 타이머의 모든 설정을 완료하고 타이머를 동작시키면, 그 후에는 TCNTn 레지스터가 타이머 클럭(clkT) 속도에 맞춰 1씩 증가하는 것이다!

  **- Atmega128에는 총 4개의 타이머가 존재하는데,
  Timer/Counter0, 2는 8bit이고, Timer/Counter1, 3은 16bit이다.
  여기서 8bit와 16bit가 의미하는 것은 최대 카운팅할 수 있는 숫자이다.**

  - 즉, TCNT0과 TCNT2는 0~255(8bit) 까지의 값을 가질수 있고, TCNT1과 TCNT3은 0~65535(16bit) 까지의 값을 가질수 있다.

- **Timer/Counter 동작시키기 (8bit)**

  - Atmega128의 Timer/Counter의 동작 모드는 4가지가 있다.

  1. Normal Mode (데이터시트 98페이지)
  타이머/카운터의 가장 기본적인 모드이다.(위에서 설명한 내용) 클럭 인가시 카운터 레지스터(TCNTn)가 1씩 증가(또는 감소)하는 동작 모드이고, Up카운트 모드, Down카운트 모드가 있다.
  최대로 카운트 할 수 있는 숫자를 넘어가면 오버플로우(Overflow)가 발생하면서 인터럽트를 요청할 수 있다. 그 후 TCNTn는 다시 0부터 1씩 증가 한다.
  (Up카운트 모드 시. Down모드는 그 반대)

인터럽트(Interrupt)란?

프로그램 수행 시, 작성한 코드에 따라 절차지향(위에서부터 순서대로 한 문장씩 처리)적으로 처리되는데, 어떤 요인에 의해 인터럽트 요청이 발생되면 현재 수행하는 코드까지 실행하고, 인터럽트 서비스 루틴(ISR) 함수를 수행하게 된다.

자세한 건 외부 인터럽트에서 다시 다룬다.

- **Timer/Counter 동작시키기 (8bit)**

2. Clear Timer on Compare Match (CTC) Mode
CTC모드는 Normal모드와는 약간 다르게 동작한다.

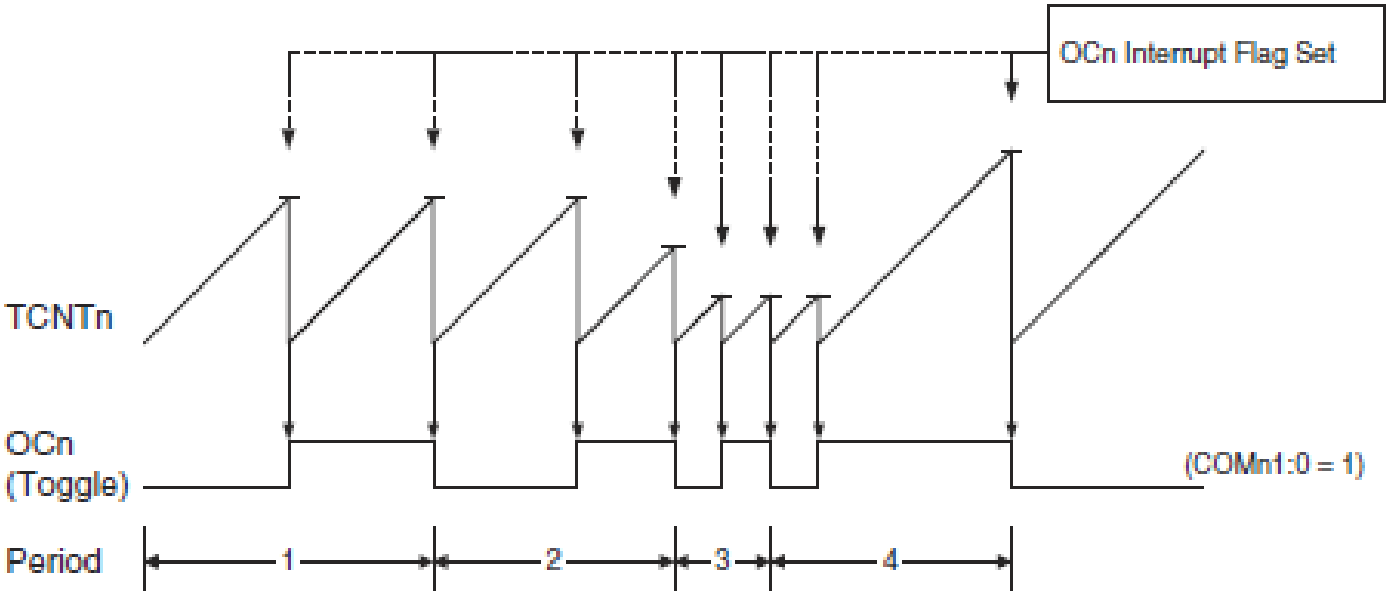
Normal모드는 TCNTn이 0부터 255까지 증가하고 256이 될 때 Overflow Interrupt가 요청된다.(업 카운트, 8bit 타이머일 때). 그 후, TCNTn은 0이 되고 다시 1씩 증가한다.

CTC모드는 TCNTn이 OCRn 레지스터의 값까지만 증가하고, TCNTn과 OCRn이 같아지면 Compare Match Interrupt가 요청된다. 그 후, TCNTn은 0부터 다시 증가.
또한, 비교매치때 Ocn핀을 토글시켜 출력할 수 있다.

- **Timer/Counter 동작시키기 (8bit)**

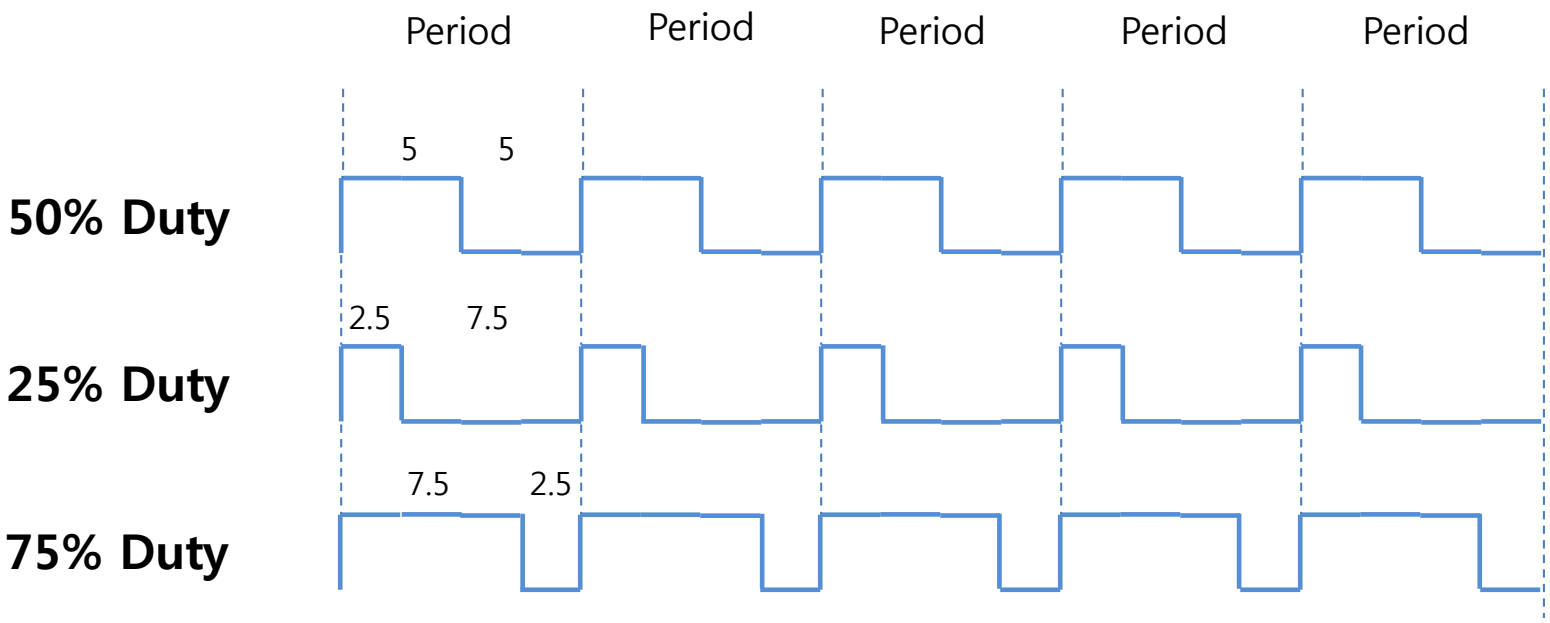
2. Clear Timer on Compare Match (CTC) Mode

Figure 38. CTC Mode, Timing Diagram



- **Timer/Counter 동작시키기 (8bit)**

- PWM이란?
PWM(Pulse Width Modulation)이란, 펄스폭 변조로 만들어지는 신호로서, 한 주기동안 H와 L의 비율(Duty rate)을 변화시켜 변조하는 방식이다.



- **Timer/Counter 동작시키기 (8bit)**

3. Fast PWM Mode

Timer/Counter를 이용하여 PWM신호를 출력할 수 있는 2가지 모드가 존재한다.

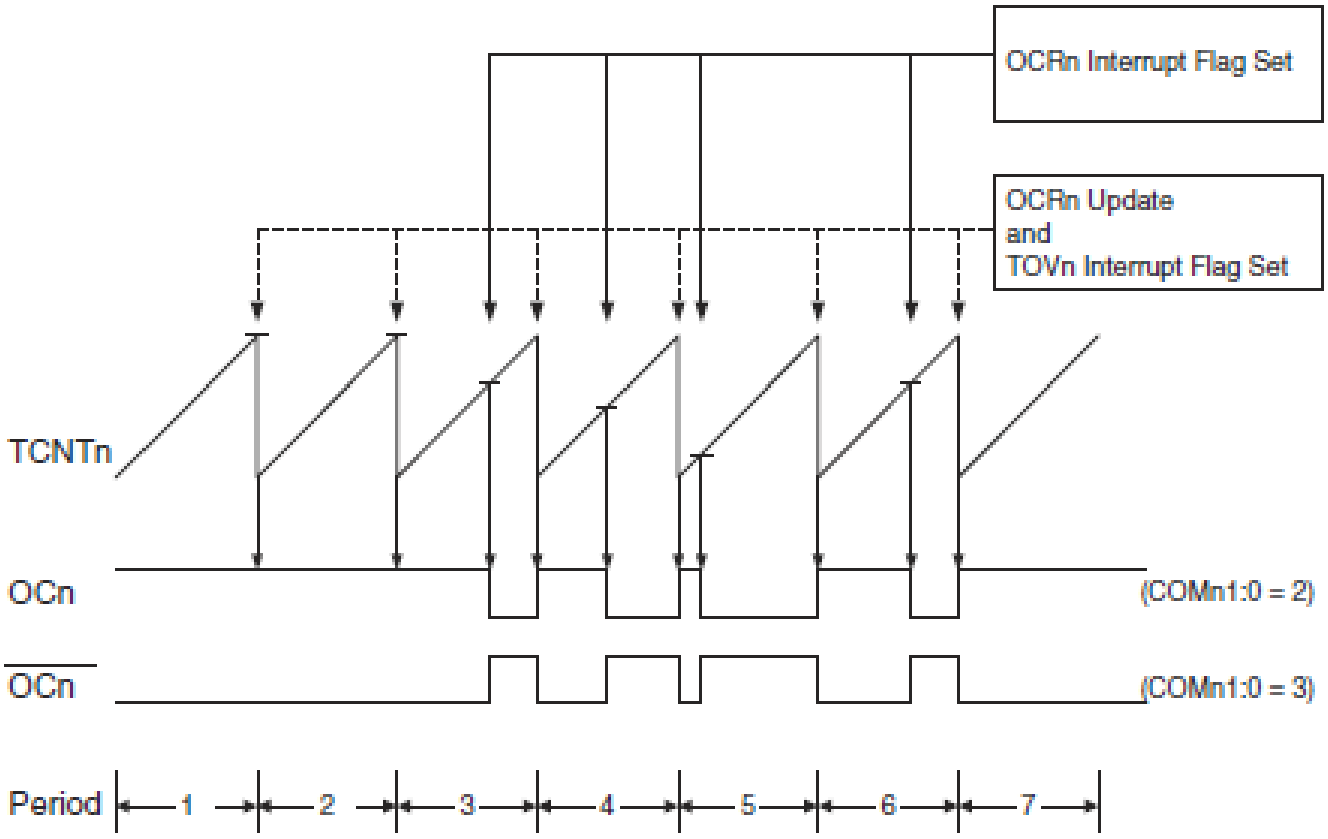
Fast PWM 모드를 시작하면 Ocn핀이 H(+5V)출력으로 시작된다. TCNTn이 증가하다가 OCRn과 같아지면 Ocn핀이 L(0V)로 떨어지고 TCNTn이 Overflow가 일어나서 0이 되면 다시 Ocn핀에서 H가 출력된다 (COMn1:0 = 2일때). 이후 반복.

즉, PWM의 한 주기는 TCNTn이 0~255(8bit일때)까지 증가하는데 걸리는 시간이 되며, Duty는 0~OCRn까지 증가하는데 걸리는 시간과 OCRn~255까지 걸리는 시간의 비율로 계산될 수 있다.

- **Timer/Counter 동작시키기 (8bit)**

3. Fast PWM Mode

Figure 39. Fast PWM Mode, Timing Diagram



- **Timer/Counter 동작시키기 (8bit)**

4. Phase correct PWM Mode

Phase correct PWM 모드는 TCNTn이 0부터 255까지 증가한 후, 다시 255부터 0까지 감소한다(반복).

Up 카운트시 OCRn과 같아지면 Ocn핀은 L가 출력되고,

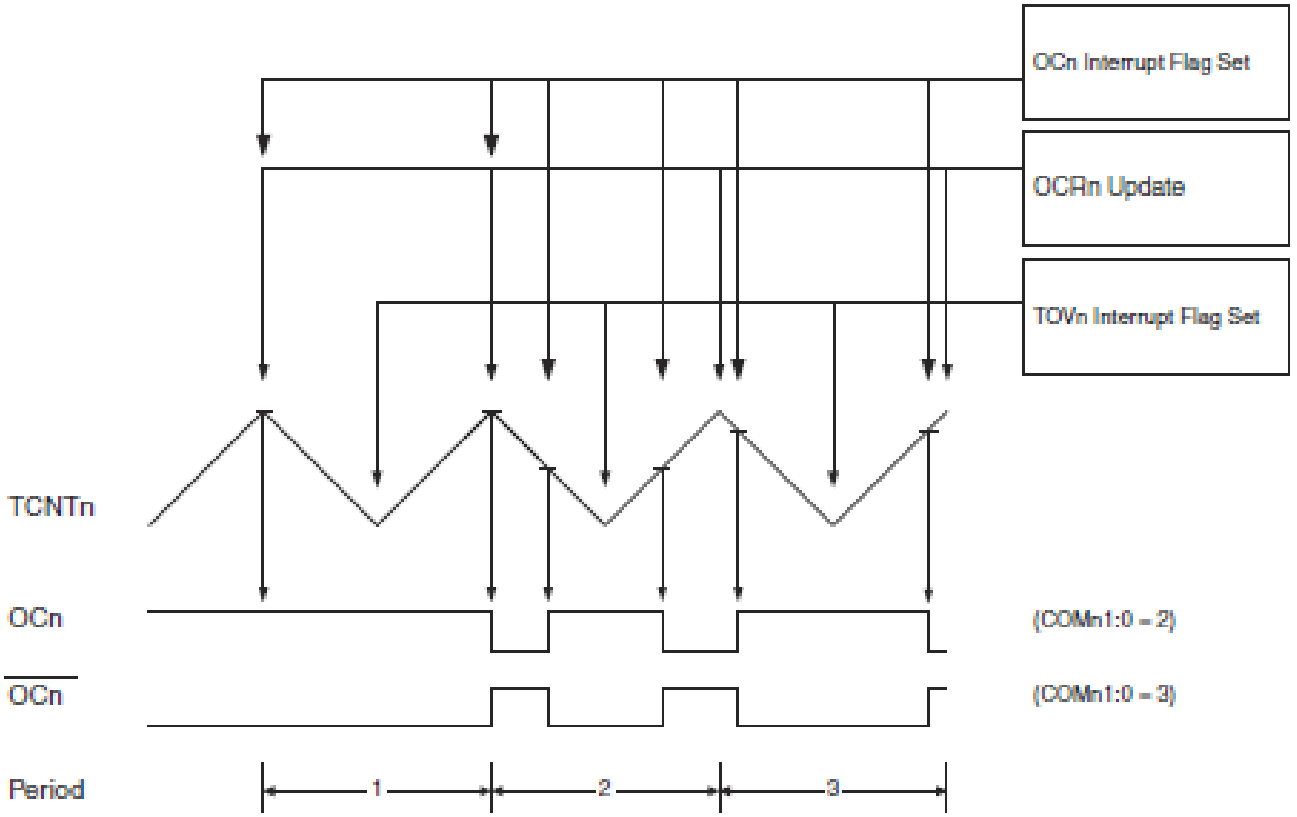
Down 카운트시 OCRn과 같아지면 Ocn핀은 H가 출력된다.

(COMn1:0 = 2일때)

- **Timer/Counter 동작시키기 (8bit)**

4. Phase correct PWM Mode

Figure 40. Phase Correct PWM Mode, Timing Diagram



3강 Timer/Counter

- 데이터시트 104페이지 부터

ATmega128에서 Timer/Counter0 기능과 관련된 레지스터들은 다음과 같다.

Timer/Counter Control Register – TCCR0

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	

Timer/Counter Register – TCNT0

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	

Output Compare Register – OCR0

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	

3강 Timer/Counter

- 데이터시트 104페이지 부터

Asynchronous Status Register – ASSR

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

Timer/Counter Interrupt Mask Register – TIMSK

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	

Timer/Counter Interrupt Flag Register – TIFR

Bit	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR
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	

Special Function IO Register – SFIOR

Bit	7	6	5	4	3	2	1	0	
	TSM	–	–	–	ACME	PUD	PSR0	PSR321	SFIOR
Read/Write	R/W	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

3강 Timer/Counter

Normal Mode로 Timer/Counter0를 설정하고 Overflow Interrupt 프로그램을 작성하여 LED 출력과 그때의 전압 출력 그래프를 오실로스코프를 통해 확인해본다.

다음과 같은 코드를 작성

```
#include <mega128.h>
#include <delay.h>
#include "usart.h"

main()
{
      /**   GPIO 입출력 설정  **/
      DDRB = 0x10; //PB4에 LED 연결
      PORTB = 0x00; //B포트 출력 초기화
      DDRE = 0x00; //PE6, 7에 스위치 연결
      /************************/

      //USART1_Init();

      /***** Timer/Counter0 기본모드 설정 *****/
      // Timer/Counter 0 initialization
      // Clock source: System Clock
      // Clock value: 15.625 kHz
      // Mode: Normal top=FFh
      // OC0 output: Disconnected
      ASSR=0x00;
      TCCR0=0x07;
      TCNT0=0x00;
      OCR0=0x00;

      // Timer(s)/Counter(s) Interrupt(s) initialization
      TIMSK=0x01;
      /****************************************/

      // Global enable interrupts
      //#asm("sei")
      SREG = 0x80;

      while(1)
      {
```

레지스터 설정의 의미를
한번 생각해보자.

이와 같이 설정하면
Timer/Counter0를
기본동작모드로
사용할 수 있게 된다!

다음과 같은 코드를 작성

```
30      // Global enable interrupts
31      //#asm("sei")
32      SREG = 0x80;
33
34      while(1)
35      {
36      }
37  }
38
39  // Timer 0 overflow interrupt service routine
40  interrupt [TIM0_OVF] void timer0_ovf_isr(void)
41  {
42      // Place your code here
43  }
```

SREG = 0x80;
이 코드의 의미를 찾아보자.

일반 함수와는 약간 다르게 생겼다. (CodeVision AVR용 ISR함수)
이를 interrupt service routine 이라고 하며
TIM0_OVF Interrupt가 요청되면 이 함수가 실행된다!!!

- 데이터시트 11페이지

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**

The Global Interrupt Enable bit must be set for the interrupts to be enabled. The individual interrupt enable control is then performed in separate control registers. If the Global Interrupt Enable Register is cleared, none of the interrupts are enabled independent of the individual interrupt enable settings. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts. The I-bit can also be set and cleared in software with the SEI and CLI instructions, as described in the instruction set reference.

Bit7 – I: 전역 인터럽트 활성화 비트

인터럽트를 활성화시키기 위해선 전역 인터럽트 활성화 비트를 Set(비트를 1로) 해야 한다.

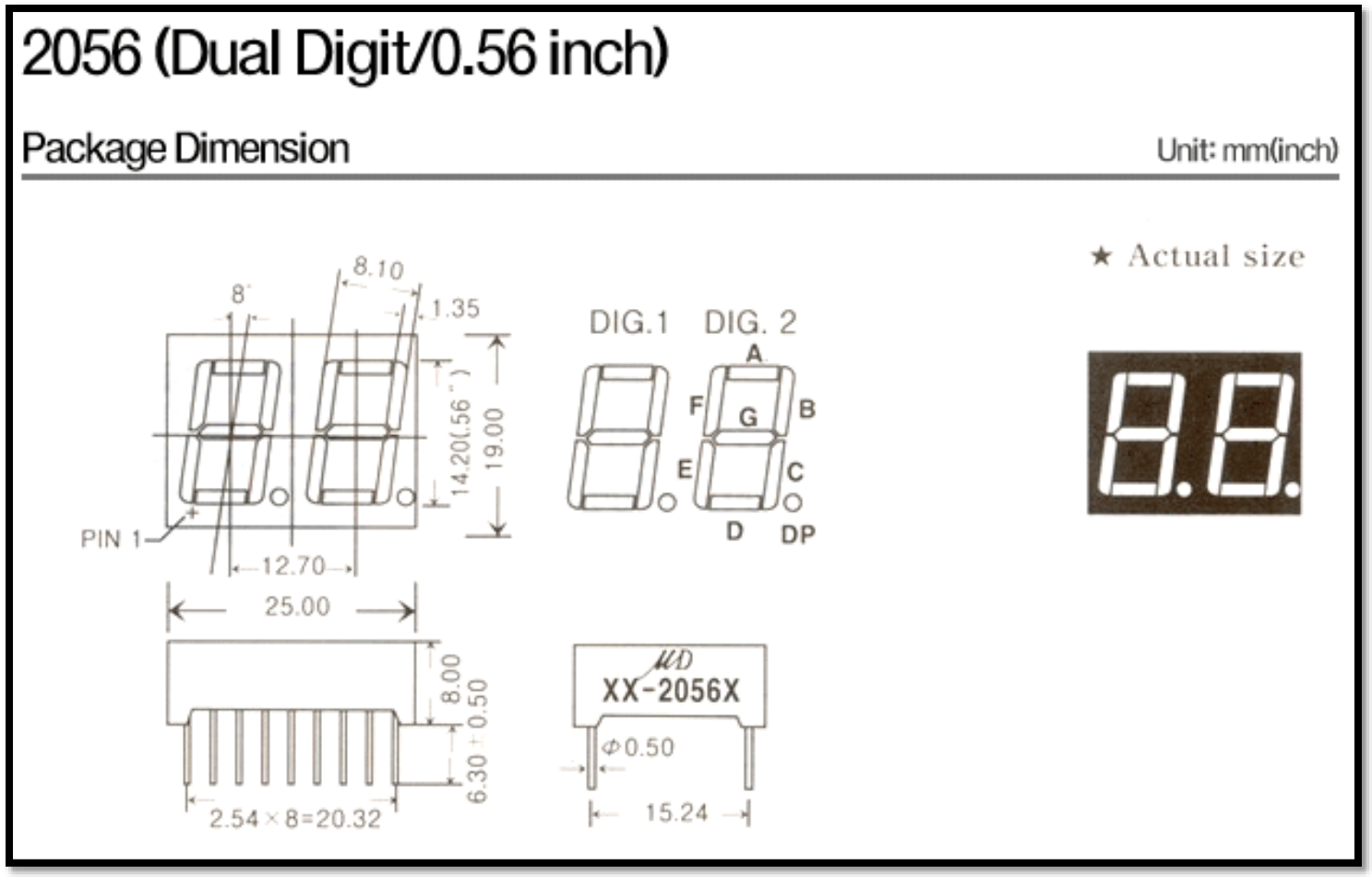
3강 Timer/Counter

1초를 만들어 보자! 어떻게 만들면 될까?

1. clkT의 클럭소스를 Mclk(fosc)의 몇 분주로 할 것인가?
   => TCNT가 1 증가하는데 걸리는 시간이 어떻게 되는가?
2. TCNT가 몇 까지 증가하면 OVF 인터럽트가 발생하는가?
   =>
3. 인터럽트가 몇 번 발생해야 1초가 지난 것인가?
   =>

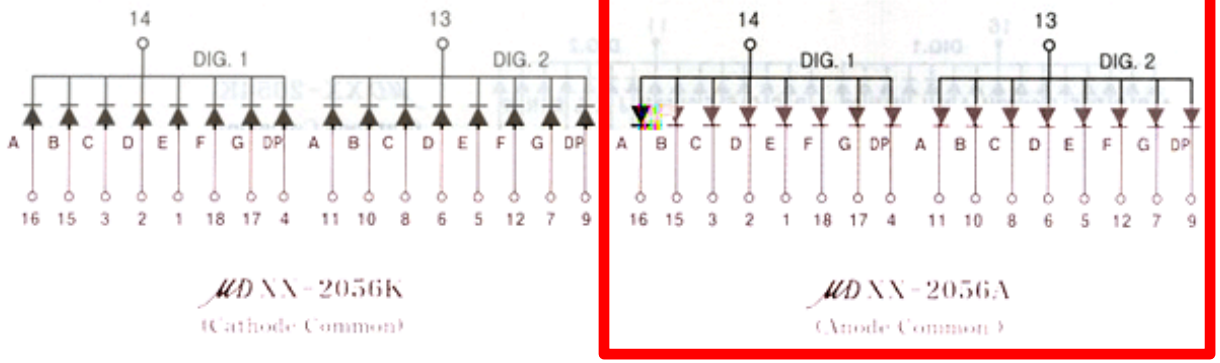
타이머를 이용해 1초에 한번씩 LED를 깜빡여보자!

- **7-Segment 핀 구성 - 1**



- **7-Segment 핀 구성 - 2**

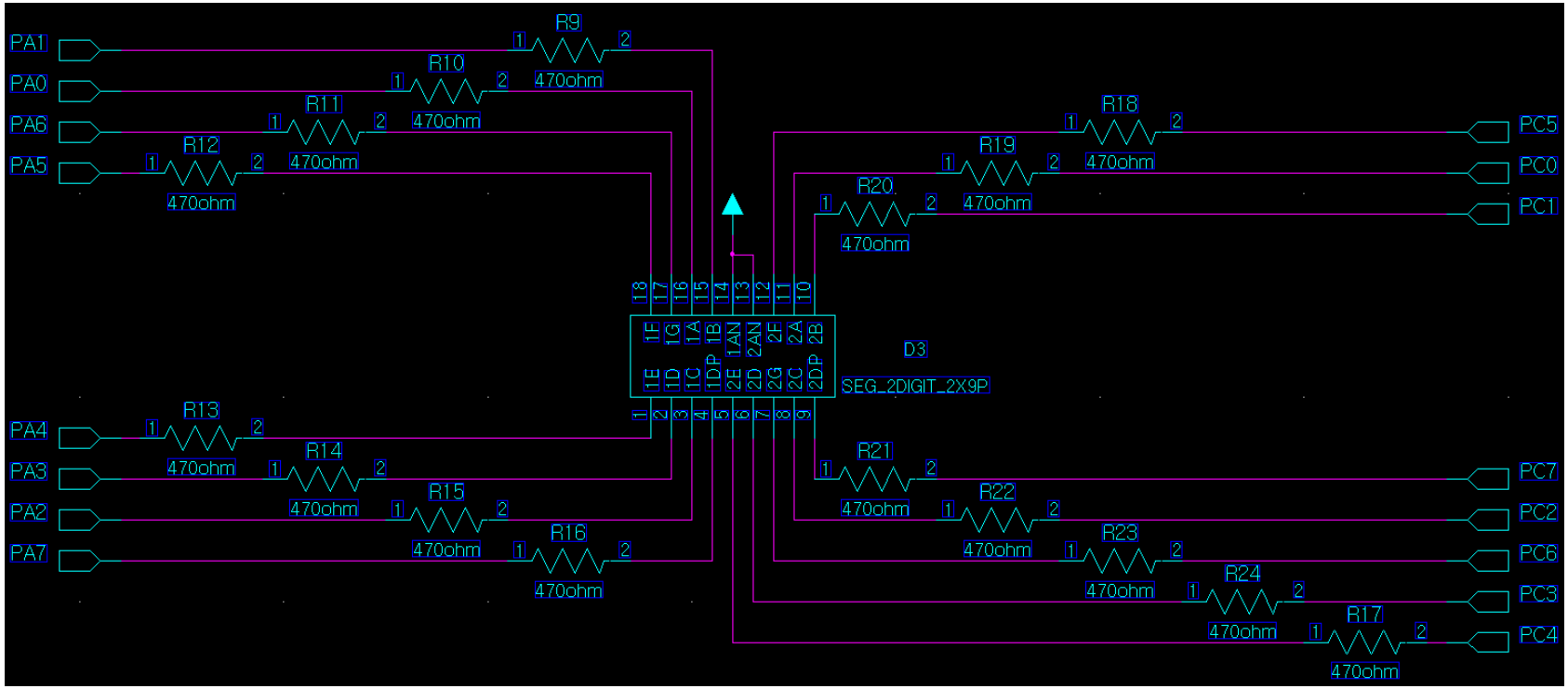
Internal Circuit Diagram



Device No.		R-2056X*	G-2056X*	SR-2056X*	Y-2056X*	UR-2056X*
CHIP	Materials	GaP/GaP	GaP/GaP	GaAsP/GaP	GaAsP/GaP	GaAlAs
	color	RED	GREEN	ORANGE	YELLOW	ULTRA RED
	λP	700	568	630	589	660
PD (mW)	TOTAL	640	640	640	640	640
	SEG.	10	10	10	10	10
VF (IF : 20mA)	Typ (V)	2.25	2.25	1.80	2.10	1.80
	Max (V)	2.60	2.60	2.10	2.60	2.00
IR	Max (μA)	10	10	10	10	10
	VR (V)	5	5	5	5	5
IV (IF : 10mA)	Min (μcd)	700	800	4,000	800	10,000
	Typ (μcd)	1,000	1,300	6,000	1,300	12,000

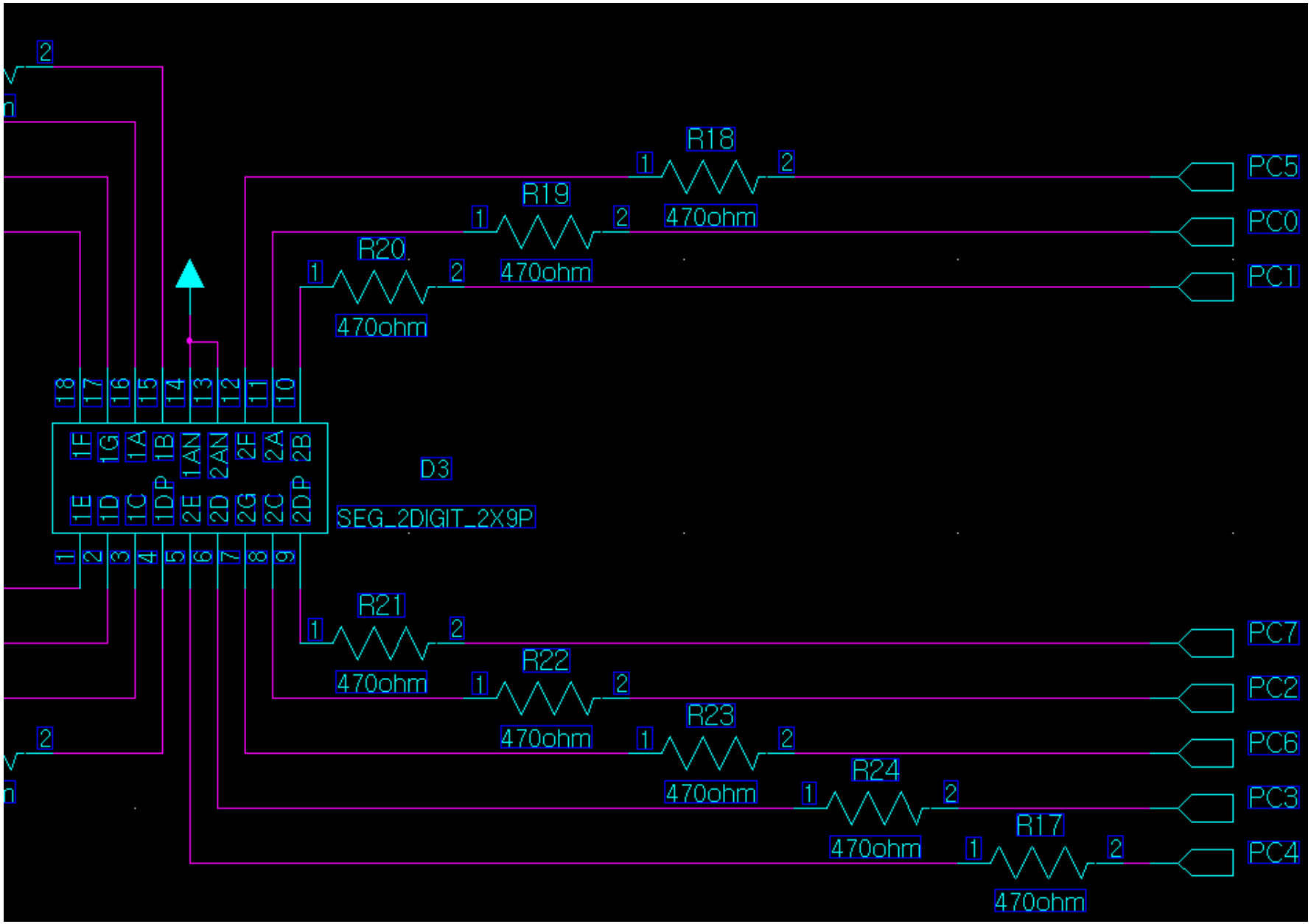
※ REMARK : X* > A = ANODE COMMON, K = CATHODE COMMON

- **7-Segment 연결 회로**

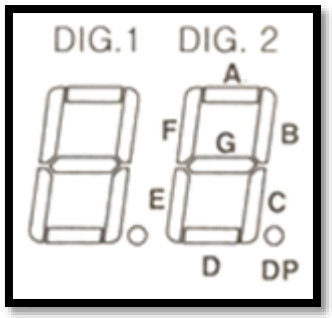


7-Segment는 여러 개의 LED가 집적되어 있다고 생각하면 된다.
우리가 사용할 7-Segment는 2자리를 가지고 있으며 Anode가 공통이다.
따라서 Anode 핀에 +5V(Vcc)를 연결해주고, 나머지 A,B,C,D,E,F,G,DP에
각각 저항(470ohm)을 거쳐 ATmega128의 포트에 연결시킨다.

3강 Timer/Counter



- **7-Segment에 숫자 표시하기**



이 그림에서
A, B, C, D, E, F만 켜면 '0', B, C만 켜면 '1', A, B, G, E, D만 켜면 '2'가 된다.

위의 회로도에서 다음과 같이 연결되어 있다.

2A – PC0	2E – PC4
2B – PC1	2F – PC5
2C – PC2	2G – PC6
2D – PC3	2DP – PC7

LED의 방향에 유의할 것! Anode가 공통이다!

다음과 같은 코드를 작성

```
#include <mega128.h>
#include <delay.h>
#include "usart.h"

main()
{
      /**   GPIO 입출력 설정   **/
      DDRB = 0x10; //PB4에 LED 연결
      PORTB = 0x00; //B포트 출력 초기화
      DDRE = 0x00; //PE6, 7에 스위치 연결
      DDRC = 0xff; //PC0~7에 7Segment 연결. 따라서 전부 출력으로 설정
      PORTC = 0xff; //C포트 출력 초기화
      /************************/

      //USART1_Init();

      /***** Timer/Counter0 기본모드 설정 *****/
      // Timer/Counter 0 initialization
      // Clock source: System Clock
      // Clock value: 15.625 kHz
      // Mode: Normal top=FFh
      // OC0 output: Disconnected
      ASSR=0x00;
      TCCR0=0x07;
      TCNT0=0x00;
      OCR0=0x00;

      // Timer(s)/Counter(s) Interrupt(s) initialization
      TIMSK=0x01;
      /*****************************************/

      // Global enable interrupts
      //#asm("sei")
      SREG = 0x80;
```

3강 Timer/Counter


```
36      while(1)
37      {                       //pgfedcba
38          PORTC = 0xc0; //0b11000000 => 0
39          delay_ms(500);
40          PORTC = 0xf9; //0b11111001 => 1
41          delay_ms(500);
42          PORTC = 0xa4; //0b10100100 => 2
43          delay_ms(500);
44          PORTC = 0xb0; //0b10110000 => 3
45          delay_ms(500);
46          PORTC = 0x99; //0b10011001 => 4
47          delay_ms(500);
48          PORTC = 0x92; //0b10010010 => 5
49          delay_ms(500);
50          PORTC = 0x83; //0b10000011 => 6
51          delay_ms(500);
52          PORTC = 0xd8; //0b11011000 => 7
53          delay_ms(500);
54          PORTC = 0x80; //0b10000000 => 8
55          delay_ms(500);
56          PORTC = 0x98; //0b10011000 => 9
57          delay_ms(500);
58      }
59  }
60
61  // Timer 0 overflow interrupt service routine
62  interrupt [TIM0_OVF] void timer0_ovf_isr(void)
63  {
64      // Place your code here
65  }
```

7 세그먼트에 0부터 9까지의 숫자를 표현한다!

3강 Timer/Counter

타이머 인터럽트를 사용하여 1초를 카운트 할 수 있는 프로그램을 만들어 매 초를 7-Segment에 숫자로 출력해보자!

문1) 1번 스위치를 누르면 타이머가 작동. 2번 스위치를 누르면 타이머가 정지하는 프로그램 작성

문2) 매 초마다 7-Segment에 초를 출력하고 짝수 초에 DP가 켜지고, 홀수 초에 DP가 꺼지는 프로그램 작성