
마이크로컨트롤러 기능

제 6장 TC를 이용한 타이머



ICAT

Integrated Circuits for Advanced Technology Lab.

TC를 이용한 타이머

1. 클럭과 카운터
2. ATmega128A의 타이머/카운터
3. 8비트 타이머/카운터의 일반 동작모드
4. 타이머로 LED 점멸시키기
5. 타이머로 디지털 시계 만들기



타이머/카운터

□ 타이머와 카운터

- ❖ 정확한 시간의 측정이 필요하다. (자명종과 스톱워치)
- ❖ 임베디드 시스템에서 타이머와 카운터가 시간측정의 일을 담당한다.
- ❖ 타이머/카운터는 일정한 개수만큼의 클럭을 세어 시간을 측정하므로, **정확한 시간 재기**가 가능하다.
- ❖ 타이머는 필요한 시간을 미리 레지스터에 설정하고, 다른 작업과 병행하게 타이머가 동작하고, 설정한 조건에서 인터럽트 발생하게 함으로써, **MCU의 효율을 극대화** 시킬 수 있다.



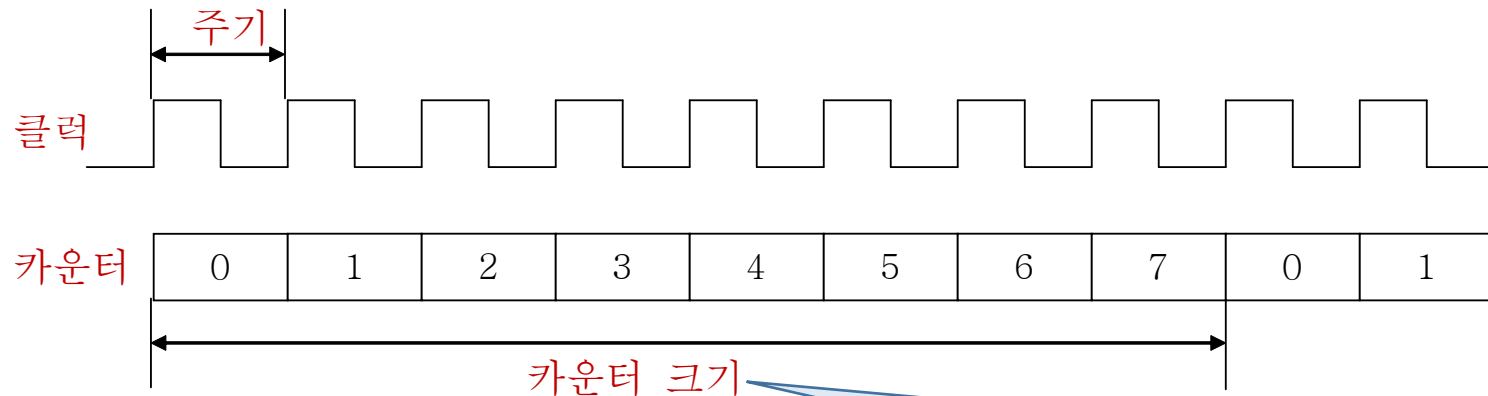
ICAT

Integrated Circuits for Advanced Technology Lab.

클럭과 카운터

□ 클럭

- ❖ 시계
- ❖ 일정한 시간 간격으로 0과 1의 값이 번갈아 나타남
- ❖ 주어진 일을 순서대로 정확한 시간에 처리하기 위해 사용



카운터의 비트수에 따라 달라짐 :
3비트 카운터 : $2^3 = 8$

□ 카운터

- ❖ 클럭을 세는 장치



ICAT

Integrated Circuits for Advanced Technology Lab.

ATMega128A 타이머/카운터

□ 타이머와 카운터

- ❖ **타이머** : MCU 내부 클럭을 세는 장치
 - 동기모드
 - 타이머는 MCU의 내부클럭을 세어 일정시간 간격의 펄스를 만들어 내거나 일정시간 경과 후에 인터럽트를 발생
- ❖ **카운터** : MCU의 외부에서 입력되는 클럭을 세는 장치
 - 비동기모드
 - 카운터는 외부 핀(TOSC1, TOSC2, T1, T2, T3)을 통해서 들어오는 펄스를 계수(Edge Detector)하여 Event Counter로서 동작



ATMega128A 타이머/카운터

□ ATMega128A의 타이머 카운터

- ❖ 타이머 0~3 모두 4개의 타이머/카운터를 보유
 - 타이머 0,2 : 8비트 타이머로 서로 기능 유사
 - 타이머 1,3 : 16비트 타이머로 서로 기능 유사
- ❖ 인터럽트 기능
 - **오버플로우 인터럽트** : 카운터의 값의 오버플로우에 의해 발생
 - **출력비교 인터럽트** : 카운터 값이 출력비교 레지스터의 값과 같게 되는 순간에 발생
 - **입력 캡처 인터럽트** : 외부로부터의 트리거 신호에 의해서 카운터의 초기값을 입력캡처
- ❖ PWM 출력 기능
 - Pulse Width Modulation



8비트 타이머/카운터

□ 8비트 타이머/카운터의 특징

- ❖ 4개의 타이머/카운터 중 0번과 2번 타이머/카운터
- ❖ PWM 및 비동기 동작 모드를 갖는 8비트 업/다운(Up/Down) 카운터
- ❖ 8비트 카운터 : $2^8 = 256$, 즉 0~255까지 셀 수 있음
- ❖ 10비트의 프리스케일러(prescaler) 보유
- ❖ 각종 인터럽트 기능
 - 오버플로우 인터럽트(overflow interrupt)
 - 출력비교 인터럽트(output compare match interrupt)
- ❖ PWM 기능 제공
- ❖ 타이머 0는 부가적으로 14.7456MHz의 수정 진동자를 TOSC1,2 단자에 연결해 Real Time Clock으로 사용할 수 있는 기능 제공



ICAT

Integrated Circuits for Advanced Technology Lab.

8비트 타이머/카운터

□ 프리스케일러(Prescaler)

- ❖ 고속의 클럭을 사용하여 타이머를 동작시킬 때 나타나는 문제를 해결하기 위해 클럭을 분주하여 더 느린 타이머 클럭을 만듦.
 - 4MHz 클럭을 사용하는 경우 그 클럭의 주기는 250ns
 - 이 클럭으로 256까지 센다고 해도 64us 이하를 세는 카운터 밖에는 만들 수가 없음.
- ❖ 10비트, 프리스케일러 보유(최대 $2^{10} = 1024$ 배 가능)
 - 프리스케일러를 1024로 하면 4MHz 클럭의 타이머 클럭은 주기가 $250\text{ns} \times 1024 = 256\mu\text{s}$ 가 되고, 이 클럭을 256까지 센다면 $256\mu\text{s} \times 256 = 65.536\text{ms}$ 크기의 타이머를 만들 수 있음.



8비트 타이머/카운터

□ 프리스케일러(Prescaler)

❖ 타이머 0

- 클럭소스 : TOSC1에 입력된 시스템 클럭 주파수의 1/4미만의 외부클럭, TOSC1/TOSC2단자에 연결된 수정 진동자에 의해 발생된 클럭, 내부 클럭
- 분주비 : 1, 8, 32, 64, 128, 256, 1024

❖ 타이머 2

- 클럭소스 : T2핀으로 입력되는 외부 클럭, 내부 클럭
- 분주비 : 1, 8, 64, 256, 1024



8비트 타이머/카운터

□ 8비트 타이머/카운터 레지스터

- ❖ 타이머/카운터 제어 레지스터(TCCRn) – 동작모드 설정
- ❖ 타이머/카운터 레지스터(TCNTn) – 타이머n의 8비트 카운터 값 저장
- ❖ 타이머/카운터 인터럽트 플래그 레지스터(TIFR) – 타이머가 발생하는 인터럽트 플래그를 저장
 - TCNTn가 오버플로우가 발생하면 TIFR의 TOVn 플래그가 세트 되고 오버플로우 인터럽트 발생
 - 출력 비교 레지스터(OCRn) 과 TCNTn 값과 같으면 TIFR의 OCFn 플래그를 세트, 출력비교 인터럽트 요청 후, Ocn 출력핀으로 출력
- ❖ 타이머/카운터 인터럽트 마스크 레지스터(TIMSK) – 타이머/카운터가 발생하는 인터럽트를 개별적으로 인에이블하도록 설정



ICAT

Integrated Circuits for Advanced Technology Lab.

8비트 타이머/카운터

□ 8비트 타이머/카운터의 일반 동작 모드

- ❖ ATmega128A는 4가지의 타이머 동작모드를 가짐
- ❖ 가장 기본적인 동작모드가 일반동작 모드임.

□ 8비트 타이머/카운터의 동작

- ❖ 동작 모드 결정.
- ❖ 타이머에 사용할 클럭 소스와 프리스케일러 결정
- ❖ 원하는 타이머의 주기 및 그 주기 동안의 시간을 정확히 세기 위한 타이머 클럭의 수 결정.

$$\text{Timer Period} = \frac{1}{\text{Clock Frequency} / \text{Prescaler}} \cdot \text{TCNT}$$

- ❖ 카운터 레지스터의 카운터 시작 값 설정(오버플로우 인터럽트 사용)



8비트 타이머/카운터

□ 8비트 타이머/카운터의 동작 예

- ❖ 동작 모드 결정 : 일반동작모드
- ❖ 타이머에 사용할 클럭 소스와 프리스케일러 결정
 - 내부클럭 : 14.7456MHz
 - 프리스케일러 : 1024
 - 타이머 클럭 주파수 : $14745600/1024 = 14.4\text{KHz}$
 - 타이머 클럭 주기 : 약 69us ($1/14400 = 0.000069444$)
- ❖ 원하는 타이머의 주기 및 그 주기 동안의 시간을 정확히 세기 위한 타이머 클럭의 수 결정.
 - 1 주기당 타이머 클럭 개수 = 타이머 클럭 주파수 * 타이머 시간
 - 만약 10ms의 타이머를 만들기 원한다면 $14400 * 0.01 = 144$
- ❖ 카운터 레지스터의 카운터 시작 값 설정
 - 카운터의 시작위치를 계산하여 카운터 레지스터를 설정
 - 카운터시작 위치 = $255 - \text{타이머 클럭 개수} (255 - 144 = 112)$
 - 카운터 레지스터(TCNT) 값을 112로 초기화 시키고 타이머를 동작시키면 144번 증가후 255값을 넘어서 인터럽트가 걸리게 됨.



8비트 타이머/카운터

□TCCRn(Timer/Counter Control Register n)

- ❖ 타이머/카운터 제어 레지스터 $n(n=0 \text{ or } 2)$
- ❖ 동작 모드, 프리스케일러등 타이머/카운터의 전반적인 동작 형태를 결정

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|-------|-------|-------|-------|------|------|------|
| FOCn | WGMn0 | COMn1 | COMn0 | WGMn1 | CSn2 | CSn1 | CSn0 |

- 비트 7 : FOCn
- 비트 6, 3 : WGM
- 비트 5, 4 : COM
- 비트 2, 1, 0 : CSn



8비트 타이머/카운터

□ TCCRn(Timer/Counter Control Register n)

❖ FOC(Force Output Compare)

- FoC가 1로 세트되면 출력 비교 실시(OC핀 작동)
- FoC가 0로 세팅되면 출력 비교를 실시하지 않음.

❖ WGM(Waveform Generation Mode)

- 타이머/카운터의 동작모드 설정
- 카운터의 카운팅 방향, 최대 카운터 값, Waveform Generation 방식 등을 결정

| 모드 | WGMn1 (CTCn) | WGMn0 (PWMn) | 타이머/카운터 동작모드 | TOP | OCRn의 업데이트 | TOVn Flag Set-on |
|----|-----------------|-----------------|--------------------|------|---------------|------------------|
| 0 | 0 | 0 | Normal | 0xFF | Immediate | MAX |
| 1 | 0 | 1 | PWM, Phase Correct | 0xFF | TOP | BOTTOM |
| 2 | 1 | 0 | CTC | OCRn | Immediate | MAX |
| 3 | 1 | 1 | Fast PWM | 0xFF | TOP | MAX |



8비트 타이머/카운터

□TCCRn(Timer/Counter Control Register n)

❖ COM(Compare Output Mode)

- OCn핀의 동작을 조정, 이 핀을 출력으로 사용하기 위해 DDR레지스터를 출력으로 설정해야 함
- COMn1/COMn0에 따른 Ocn 핀의 동작
 - **Compare Output**, Non-PWM 모드

| COMn1 | COMn0 | 내용 |
|-------|-------|-----------------------------|
| 0 | 0 | normal 포트 동작, OCn 연결을 끊다. |
| 0 | 1 | Toggle OCn on compare match |
| 1 | 0 | Clear OCn on compare match |
| 1 | 1 | Set OCn on compare match |



8비트 타이머/카운터

□TCCRn(Timer/Counter Control Register n)

- Compare Output, Fast-PWM 모드

| COMn1 | COMn0 | 내용 |
|-------|-------|--|
| 0 | 0 | normal 포트 동작, OCn 연결을 끊다. |
| 0 | 1 | 예약 |
| 1 | 0 | Clear OCn on compare match, set OCn at TOP |
| 1 | 1 | Set OCn on compare match, clear OCn at TOP |

- Compare Output, PC PWM Mode

| COMn1 | COMn0 | 내용 |
|-------|-------|--|
| 0 | 0 | normal 포트 동작, OCn 연결을 끊다. |
| 0 | 1 | 예약 |
| 1 | 0 | up카운팅일때 Clear OCn on compare match, down카운팅일때 set Ocn |
| 1 | 1 | up카운팅일때 Set OCn on compare match, down카운팅일때 clear Ocn |



8비트 타이머/카운터

□TCCRn(Timer/Counter Control Register n)

❖ CSn : 클럭 선택(Clock Select)

- 타이머/카운터에 사용할 클럭과 프리스케일러를 선택
- 타이머/카운터 0

| CS02 | CS01 | CS00 | 설명 |
|------|------|------|-----------|
| 0 | 0 | 0 | 클럭소스가 없다. |
| 0 | 0 | 1 | No 프리스케일러 |
| 0 | 1 | 0 | 8분주 |
| 0 | 1 | 1 | 32분주 |
| 1 | 0 | 0 | 64분주 |
| 1 | 0 | 1 | 128분주 |
| 1 | 1 | 0 | 256분주 |
| 1 | 1 | 1 | 1024분주 |



8비트 타이머/카운터

□TCCRn(Timer/Counter Control Register n)

- 타이머/카운터 2

| CS22 | CS21 | CS20 | 설명 |
|------|------|------|----------------------|
| 0 | 0 | 0 | 클럭소스가 없다. |
| 0 | 0 | 1 | No 프리스케일러 |
| 0 | 1 | 0 | 8분주 |
| 0 | 1 | 1 | 64분주 |
| 1 | 0 | 0 | 256분주 |
| 1 | 0 | 1 | 1024분주 |
| 1 | 1 | 0 | 외부클럭의 하강에지에서 카운터 2동작 |
| 1 | 1 | 1 | 외부클럭의 상승에지에서 카운터 2동작 |



8비트 타이머/카운터

□ TCNTn (Timer/Counter Register n)

- ❖ 타이머/카운터 레지스터 n (n은 0 or 2)
 - 타이머/카운터 n의 8비트 카운터 값을 저장하고 있는 레지스터
 - 이 레지스터는 쓸 수도 있고 읽을 수도 있다.
 - 임의의 값을 써주면 타이머의 주기를 더 빠르게 할 수 있다.
 - 레지스터의 값은 인터럽트가 걸리면 자동으로 0으로 클리어 되고 다시 분주된 주기마다 한 개씩 값이 늘어난다.

| | | | | | | | |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TCNT7 | TCNT6 | TCNT5 | TCNT4 | TCNT3 | TCNT2 | TCNT1 | TCNT0 |



8비트 타이머/카운터

□TIMSK(Timer Interrupt Mask)

- ❖ 타이머 인터럽트 마스크 레지스터
- ❖ 타이머/카운터0, 타이머/카운터1, 타이머/카운터2가 발생하는 인터럽트를 개별적으로 enable하는 레지스터

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

- ❖ 비트 1, 7 : OCIE0/OCIE2
 - 타이머/카운터0/2의 출력비교 인터럽트 인에이블
- ❖ 비트 0, 6 : TOIE0/TOIE2
 - 타이머/카운터0/2 오버플로우 인터럽트 인에이블



8비트 타이머/카운터

□TIFR(Timer Interrupt Flag Register)

- ❖ 타이머 인터럽트 플래그 레지스터
- ❖ 타이머/카운터0, 타이머/카운터1, 타이머/카운터2가 발생하는 인터럽트의 플래그를 저장하는 레지스터

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

- ❖ 비트 1, 7 : OCF0/OCF2
 - TCNT0/2레지스터와 출력비교 레지스터 OCR0/2값을 비교해서, 같으면 이 비트가 "1"로 세트되어 출력 비교인터럽트가 발생.
- ❖ 비트 0, 6 : TOV0/TOV2
 - 타이머/카운터0/2에서 오버플로우가 발생하면 이 비트가 "1"로 세트되어 오버플로우 인터럽트가 발생.



8비트 타이머/카운터

□ASSR(ASynchronous Status Register)

- ❖ 비동기 상태 레지스터
- ❖ 타이머/카운터0이 외부 클럭에 의하여 비동기 모드로 동작하는 경우 관련된 기능을 수행하는 레지스터

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|-----|--------|-------------|--------|
| - | - | - | - | AS0 | TCN0UB | OCR0UB B | TCR0UB |

- 비트 3(AS0 : ASynchronous timer 0)-1이면 비동기모드로 동작
- 비트 2(TCN0UB : Timer/CouNter0 Update Busy) -**TCNT0**
- 비트 1(OCR0UB : Output Compare Register 0 Update Busy) -**OCR0**
- 비트 0(TCR0UB : Timer CouNter Register 0 Update Busy) -**TCCR0**
- 비트 2, 1, 0은 비동기모드로 동작 시 **각 레지스터**에 새로운 값을 기록하면 1로 세트



8비트 타이머/카운터

□SFIOR(Special Function I/O Register)

- ❖ 특수 기능 I/O 레지스터
- ❖ 타이머/카운터들을 동기화 하는데 관련된 기능을 담당하는 레지스터

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|---|---|---|------|-----|------|--------|
| TSM | - | - | - | ACME | PUD | PSR0 | PSR321 |

- **비트 7(TSM : Timer Synchronization Mode)**
 - '0'으로 클리어시 : PSR0, PSR321비트를 하드웨어적으로 클리어,
 - 이로 인해, 모든 타이머들이 동시에 카운팅 동작을 시작한다
- **비트 1(PSR0 : Prescaler Reset Timer 0)**
- **비트 0(PSR321 : Prescaler Reset Timer 321)**
- TSM '1'로 세트 시 : PSR0, PSR321비트를 값을 유지하고 대응하는 프리스케일러를 리셋하여 해당 타이머 동작을 정지시키고 같은 값이 되도록 설정



8비트 타이머/카운터

□ATMega128A 타이머 오버플로우 인터럽트 프로그램

❖ 타이머 오버플로우 인터럽트 프로그램의 틀

```
#include <io.h>
#include <interrupt.h> //인터럽트 관련 시스템 헤더 파일
SIGNAL(SIG_OVERFLOW0) /* Timer 0의 Overflow Interrupt 에 대한 Interrupt Service
                        Routine 선언*/
{
cli();
/* 인터럽트가 발생되었을 때 실행할 명령어 세트를 선언 */
sei();
}
int main(void)
{
TIMSK |= 1 << TOIE0; //오버플로우 인터럽트 허용
TIFR |=1 << TOV0; //TOV0 Timer/Counter0 overflow flag 클리어
sei(); /* 전체 인터럽트 Enable */
for (;;) { /* main loop 명령어 세트 */
}
}
```



ICAT

Integrated Circuits for Advanced Technology Lab.

실습 7 : 타이머로 LED 점멸

□ 실습 개요

- ❖ ATmega128A 마이크로컨트롤러의 타이머 기능을 이용하여 LED를 점멸시키는 실습
- ❖ 타이머를 이용하여 정확히 1초 마다 LED가 점멸하도록 함.
- ❖ 타이머는 타이머/카운터 0의 일반 동작 모드를 사용

□ 실습 목표

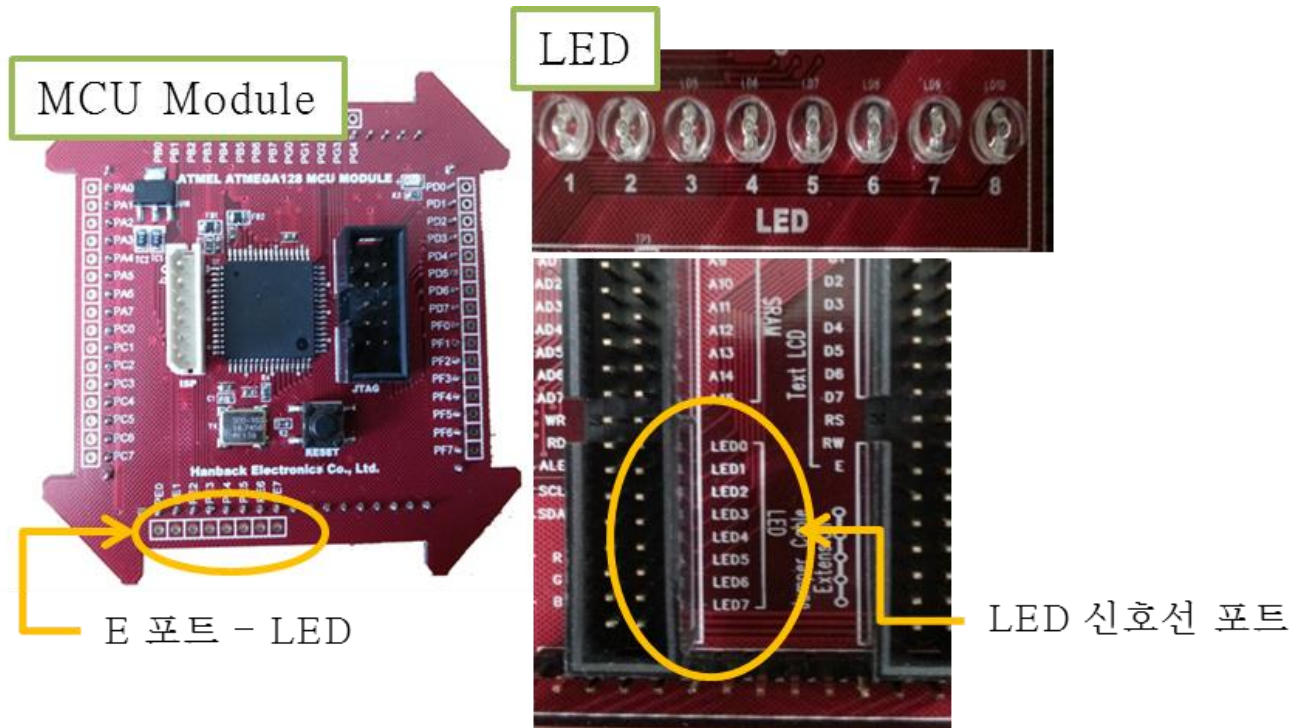
- ❖ 타이머0의 동작원리 이해
- ❖ 타이머0 제어 방법의 습득(관련 레지스터 이해)
- ❖ 오버플로우 인터럽트 제어 프로그램 방법 습득



실습 7 : 타이머로 LED 점멸

□ 사용 모듈

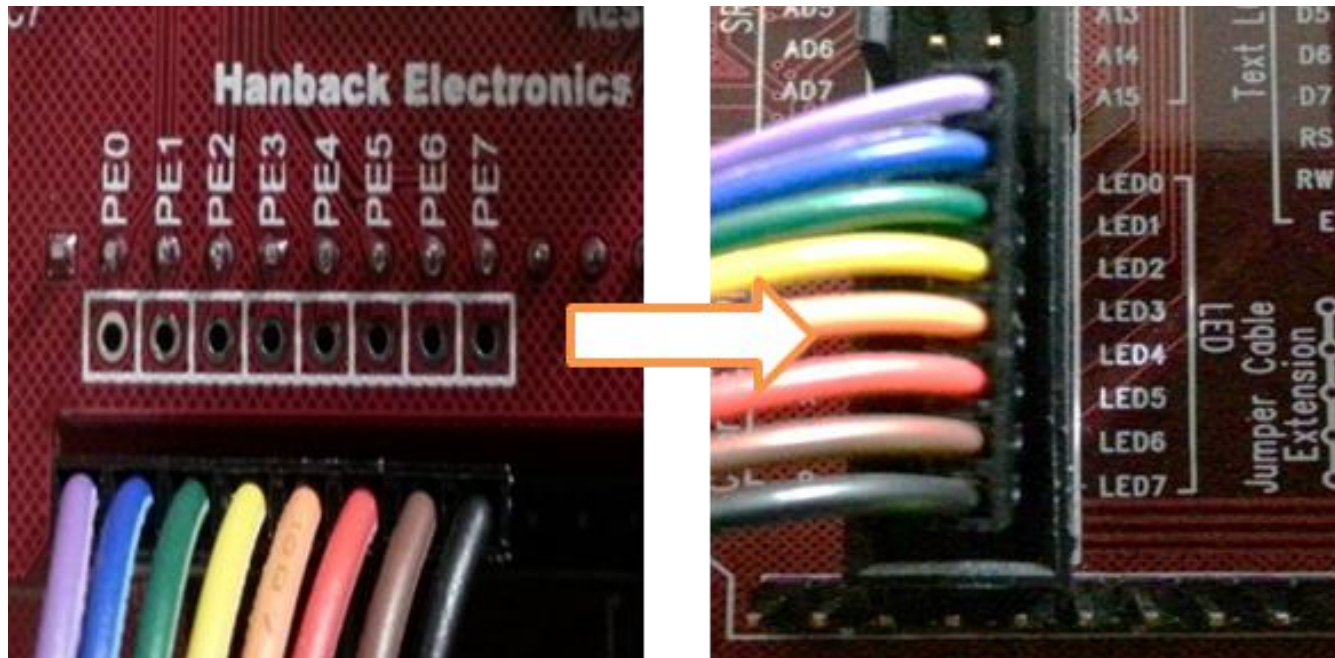
- ❖ MCU 모듈, LED 모듈



실습 7 : 타이머로 LED 점멸

□ 모듈 결선 방법

- ❖ MCU 모듈 포트 E의 PE0~PE7을 LED 모듈의 LED0~LED7까지 연결



실습 7 : 타이머로 LED 점멸

- ❖ 구동 프로그램 : 타이머/카운터 를 동작시키기 위한 사전 지식
 - ❖ 사용할 타이머/카운터 결정
 - 여기서는 8비트 타이머/카운터인 타이머/카운터 0를 사용
 - ❖ 동작 모드 결정
 - 여기서는 일반 동작 모드 사용
 - TCCR 레지스터의 CS를 제외한 모든 비트들을 0으로 세트.
 - ❖ 타이머 클럭 결정(클럭소스 및 프리스케일러 결정)
 - 여기서는 내부 클럭(14.7456MHz)을 사용
 - 프리스케일러는 최대한 주기를 길게 하기 위해 1024를 사용
 - ASSR /SFIOR는 디폴트, TCCR0의 CS비트를 “111”로 설정
 - ❖ 타이머 주기 결정
 - 1초를 만들기 위해 10ms을 타이머 주기로 결정
 - 인터럽트는 10ms마다 발생, 인터럽트가 100번 발생하면 1초로 간주
 - 10ms를 맞추기 위해서 세어야 하는 카운터 개수를 계산하고 그 카운터 개수를 만족시키기 위한 TCNT 초기값을 TCNT0에 넣음. (112)
 - ❖ 인터럽트 인에이블
 - TIMSK 레지스터를 설정하여 오버플로우 인터럽트를 Enable하고, TIFR 레지스터의 Timer/Counter0 overflow flag를 클리어



실습 7 : 타이머로 LED 점멸

□ 구동 프로그램 : 소스 분석

❖ Timer_led.c

| | |
|----|--|
| 1) | <pre>#include<avr/io.h> #include<avr/interrupt.h> volatile char LED_Data = 0x00; unsigned char timer0Cnt=0;</pre> |
| 2) | <pre>SIGNAL(SIG_OVERFLOW0); //Timer0 Overflow0 ISP int main(){</pre> |
| 3) | <pre>DDRE = 0xFF; //포트 E를(0~7비트까지 모두) 출력 포트로 사용</pre> |



ICAT

Integrated Circuits for Advanced Technology Lab.

실습 7 : 타이머로 LED 점멸

| | |
|----|--|
| 4) | <pre>TCCR0 = 0x07; /* 0~7비트 까지의 모든 register중에서 0~2까지의 레지스터를 사용하여 출력한다. 1024prescaler (Timer/Counter0 Control Register: Timer/Count0의 동작을 설정하고 프리스케일러를 설정하는 등의 기능을 수행한다.) */ TCNT0 = 112; /* (1 / (14.7456 Mhz / 1024 prescaler)) * 144=>0.01s 0.01s를 얻기위한 카운트 값 (Timer/Count0 Register: Timer/Count0의 8비트 카운터 값 을 저장하고 있는 레지스터이다.) */ TIMSK = 0x01; //오버플로우 인터럽트 허용 TIFR = 1 << TOV0; //TOV0 Timer/Counter0 overflow flag 클리어</pre> |
| 5) | <pre>sei(); while(1){</pre> |



실습 7 : 타이머로 LED 점멸

| | |
|----|---|
| 6) | <pre>PORTE = LED_Data; //LED_Data를 포트 E로 보낸다. } return 0; }</pre> |
| 7) | <pre>SIGNAL(SIG_OVERFLOW0){ cli(); TCNT0 =112; timer0Cnt++; if(timer0Cnt == 100){ //0.01s * 100 = 1s 1초를 얻기 위한 카운트 횟수 LED_Data++; timer0Cnt=0; } sei(); }</pre> <div data-bbox="871 982 1570 1250" style="border: 1px solid #add8e6; border-radius: 15px; padding: 20px; text-align: center; margin: 20px auto; width: fit-content;"><p>인터럽트 서비스 루틴</p></div> |



실습 7 : 타이머로 LED 점멸

□ 실행 결과



실습 8 : 타이머를 이용한 디지털 시계

□실습 개요

- ❖ 타이머를 이용하여 디지털 시계의 기능을 설계
- ❖ Array-FND 모듈에 마이크로 컨트롤러 출력 포트를 연결하고, 클럭을 이용하여 일정 카운트 기능을 수행
- ❖ 타이머/카운터 0의 일반 모드 동작을 사용한다
- ❖ 편이 상 시계는 초와 분까지만 표시

□실습 목표

- ❖ 타이머/카운터 활용 방법의 습득(관련 레지스터 이해)
- ❖ 디지털 시계(초/분) 구현 방법 이해



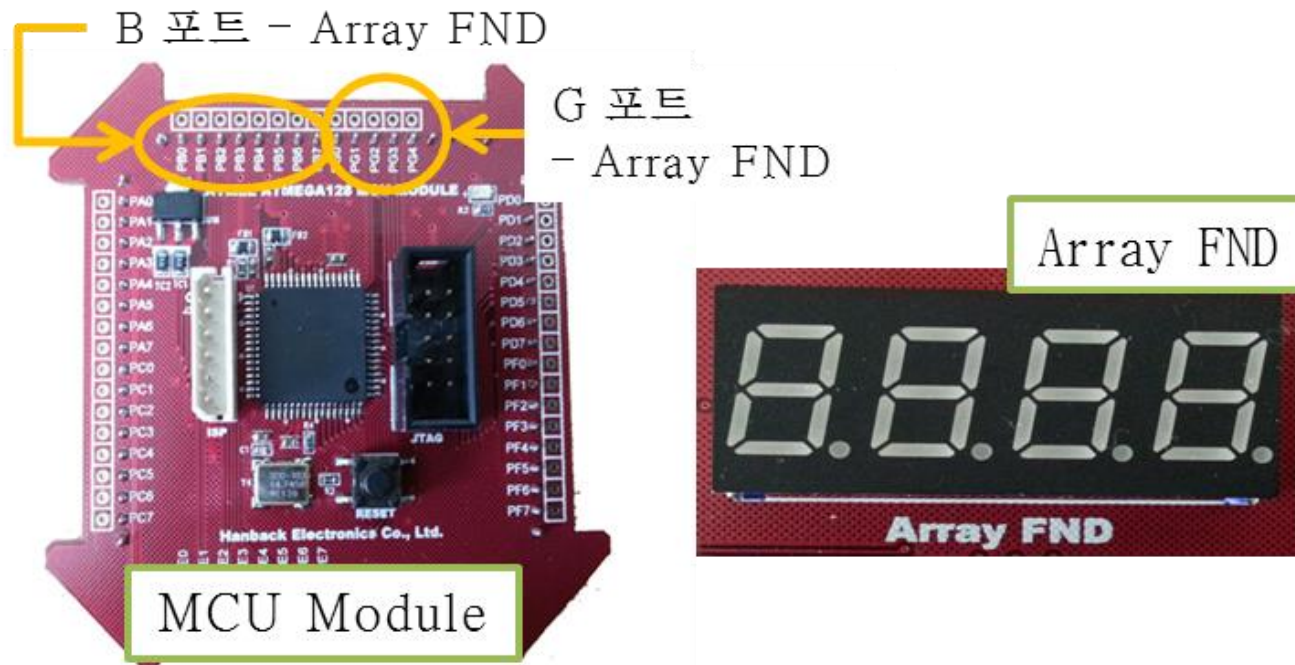
ICAT

Integrated Circuits for Advanced Technology Lab.

실습 8 : 타이머를 이용한 디지털 시계

□ 사용 모듈

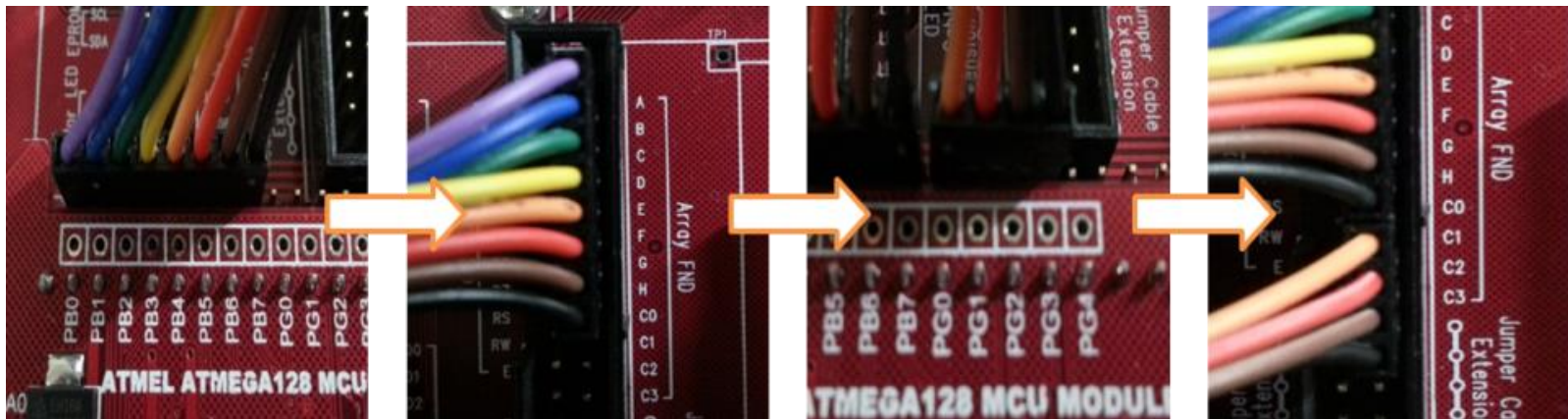
- ❖ MCU 모듈, ArrayFND 모듈



실습 8 : 타이머를 이용한 디지털 시계

□ 모듈 결선 방법

- ❖ MCU 모듈의 포트 B의 PB0~PB7를 Array FND 모듈의 A~H까지 연결
- ❖ MCU 모듈의 포트 G의 PG0~PG3는 Array FND 모듈의 C0~C3까지 연결



실습 8 : 타이머를 이용한 디지털 시계

□ 구동 프로그램 : 사전 지식

- ❖ 타이머를 이용하여 디지털 시계
- ❖ 타이머로 LED 점멸하기 예제와 거의 유사
- ❖ 시계 표시를 위해 Array-FND를 사용
- ❖ 타이머를 이용하여 정확히 1초를 카운트



ICAT

Integrated Circuits for Advanced Technology Lab.

실습 8 : 타이머를 이용한 디지털 시계

□ 구동 프로그램 : 소스 분석

❖ Timer_digitalclock.c

| | |
|----|---|
| 1) | <pre>#include<avr/io.h> #include<avr/interrupt.h> #include<util/delay.h> enum{C0,C1,C2,C3}; unsigned char FND_DATA_TBL[]={0x3F,0X06,0X5B,0X4F,0X66,0X6D,0X7C,0X07,0 X7F, 0X67,0X77,0X7C,0X39,0X5E,0X79,0X71,0X08,0X80}; //표시문자에 대한 FND 문자표 volatile unsigned char time_s=0; unsigned char time_m=0,timer0Cnt=0; void print_FND(unsigned char selCx, unsigned char data); /* 원하는 FND 포트에 data 출력 */</pre> |
| 2) | <pre>SIGNAL(SIG_OVERFLOW0); //Timer0 Overflow0 ISP int main(){</pre> |



ICAT

Integrated Circuits for Advanced Technology Lab.

실습 8 : 타이머를 이용한 디지털 시계

| | |
|----|---|
| 3) | <pre> DDR_B = 0xFF; /* data port 0~7비트까지의 모든 register를 사용하여 출력한다. (Data Direction Register) */ PORT_B = 0x00; /* 0~7비트까지의 모든 register를 사용하여 입력한다. */ DDR_G = 0x0F; /* control port 0-3 PIN 0~4비트까지의 register중에서 0~3비트까지의 register를 사용하여 출력한다. */ PORT_G = 0x0F; </pre> |
| 4) | <pre> TCCR0 = 0x07; /* 0~7비트까지의 모든 register중에서 0~2까지의 레지스터를 사용하여 출력한다. 1024prescaler (Timer/Counter0 Control Register: Timer/Counter0의 동작을 설정하고 프리스케일러를 설정하는 등의 기능을 수행한다.) */ TCNT0 = 112 ; /* (1 / (14.7456 Mhz / 1024 prescaler)) * 144 =>0.01s 0.01s를 얻기위한 카운트 값 (Timer/Counter0 Register: Timer/Counter0의 8비트 카운터 값을 저장하고 있는 레지스터이다.) */ TIMSK = 1 << TOIE0; //오버플로우 인터럽트 허용 TIFR = 1 << TOV0; //TOV0 Timer/Counter0 overflow flag 클리어 </pre> |



실습 8 : 타이머를 이용한 디지털 시계

| | |
|----|--|
| 5) | <pre>sei(); while(1){</pre> |
| 6) | <pre>print_FND(C3, time_s%10); //4번째 COM 에 1초단위 출력 _delay_ms(1); print_FND(C2, time_s/10); //3번째 COM에 10초단위 출력 _delay_ms(1); print_FND(C1, time_m%10); //2번째 COM에 1분 단위 출력 _delay_ms(1); print_FND(C0, time_m/10); //1번째 COM에 10분단위 출력 _delay_ms(1); } return 0; }</pre> |



실습 8 : 타이머를 이용한 디지털 시계

```
7) SIGNAL(SIG_OVERFLOW0){
    cli();
    TCNT0 = 112;
    timer0Cnt++;
    if(timer0Cnt == 100){           //0.01s * 100 = 1s //1초를 얻기 위한 카운트 횟수
        if(time_s >= 59){
            time_m++;
            time_s = 0;
        }
        else
            time_s++;
        if(time_m > 59){
            time_m = 0;
        }
        timer0Cnt = 0;
    }
    sei();
}
```

인터럽트 서비스 루틴



ICAT

Integrated Circuits for Advanced Technology Lab.

실습 8 : 타이머를 이용한 디지털 시계

```
8) void print_FND(unsigned char selCx, unsigned char data){
    switch(selCx){
    case C3:           //4번째 FND (좌측기준)
        PORTG &= 0xF0; //하나씩만 입력하기 위해서 그 전의 내용을 초기화 시킴
        PORTG |= 0x07; //포트 G의 G0을 사용하여 C3로 출력
        break;
    case C2:           //3번째 FND
        PORTG &= 0xF0; //이전 내용 초기화
        PORTG |= 0x0B; //포트 G의 G1을 이용하여 C2로 출력
        break;
    case C1:           //2번째 FND
        PORTG &= 0xF0;
        PORTG |= 0x0D; //포트 G의 G2를 이용하여 C1으로 출력
        break;
    case C0:           //1번째 FND
        PORTG &= 0xF0;
        PORTG |= 0x0E; //포트 G의 G3를 이용하여 C0로 출력
        Break;
    }
```



실습 8 : 타이머를 이용한 디지털 시계

```
9) default:
PORTG &= 0xF0;
PORTG |= 0x0F; //포트 G의 G4를 사용하지 않는다.
break;
}
if(selCx == C1 && time_s%2 == 0) //C1에 일정하게 . 을 찍는다
PORTB =FND_DATA_TBL[data] | 0x80;
else
PORTB =FND_DATA_TBL[data];
}
```



실습 8 : 타이머를 이용한 디지털 시계

□ 실행 결과

