[마이크로프로세서및실험]

학부:	학번 : 20066062	성명 : 이명민

주제: 타이머인터럽트(Timer/counter0 Overflow)

원광대학교 전기전자 및 정보공학부 이 상 설 교수

■ 실험목적

○ 타이머 인터럽트의 이용 프로그래밍

1. TIMSK 레지스터의 TOIE0 비트를 1로 셋팅하면 타이머0 인터럽트가 활성화 된다.

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	

```
TIMSK = TIMSK | 1<<TOIE0;
```

3. 타이머카운터0의 클럭소스를 선택한다.

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	

Table 56. Clock Select Bit Description

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)

※ MPU에 인가되는 클럭의 1024 개 마다 카운터클럭으로 사용하려면 CS02, CS01, CS00를 "1 1 1"로 셋팅하여야 한다.

```
TCCR0 = (TCCR0 & 0xF8) | 7;
```

4. 인터럽트 자체를 활성화하기 위해 실행할 명령

```
sei();
```

5. 인터럽트 서비스루틴의 설정

(1) 프로그램의 앞부분에 include 파일을 추가한다.

#include <avr/interrupt.h>

(2) 타이머카운터0 인터럽트에 대한 서비스프로그램을 작성한다.

SIGNAL(SIG_OVERFLOW0)

{ 타이머 카운터0 인터럽트 서비스 프로그램 }

6. main 프로그램과 인터럽트서비스루틴에서 공통으로 사용되는 변수는 volatile 제약어를 앞에 사용하여 외부변수로 선언한 후 이용한다.

■ 실험내용

1. 약 1초 마다 2개의 세븐세그먼트에서 숫자가 1씩 증가하게 프로그램 하라.

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

unsigned      char digit[]={0x11, ... }; // 각 팀의 7 세그먼트 LED용 값을 사용한다.
volatile      long   timer = 0;
volatile      long   number = 0;


SIGNAL(SIG_OVERFLOW0)
{
        // 매 오버플로 인터럽트 마다 timer를 1씩 증가시키다가
        // 약 1초가 되었을 때 number 값을 1증가시키게 프로그램 한다.
        timer++;
        if(timer % 61 == 0)
                number++;
}

//////////////////////////////////////////////////////
//
//////////////////////////////////////////////////////
int     main(void)
{
        DDRA = 0xFF;

        TIMSK |= 1<<TOIE0;
        TCCR0 = (TCCR0 & 0xF8) | 7;

        sei();                                  // start global interrupt

        while(1){
                PORTA = digit[number % 10];
        }
}
```

2. (HomeWork) LED를 임의의 핀에 연결하고 매 0.5초 마다 On/Off 하게 하라.

실행코드 설명과 함께 리포트 제출(1주일 후 까지)

[마이크로프로세서및실험]

학부:	학번 :	성명 :

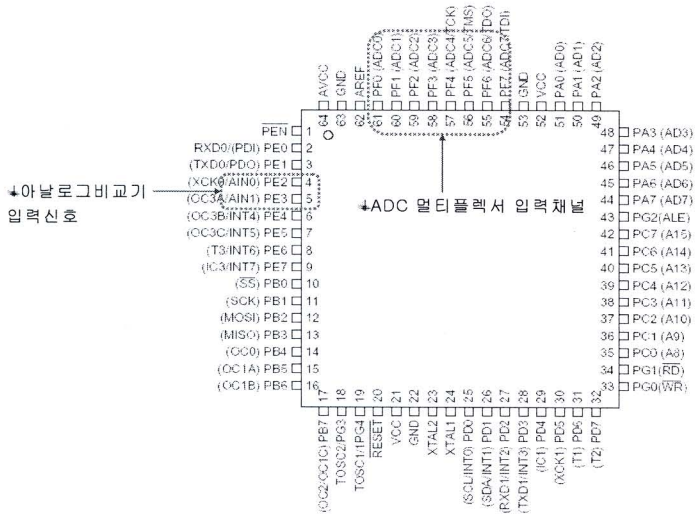
주제: Analog Comparator

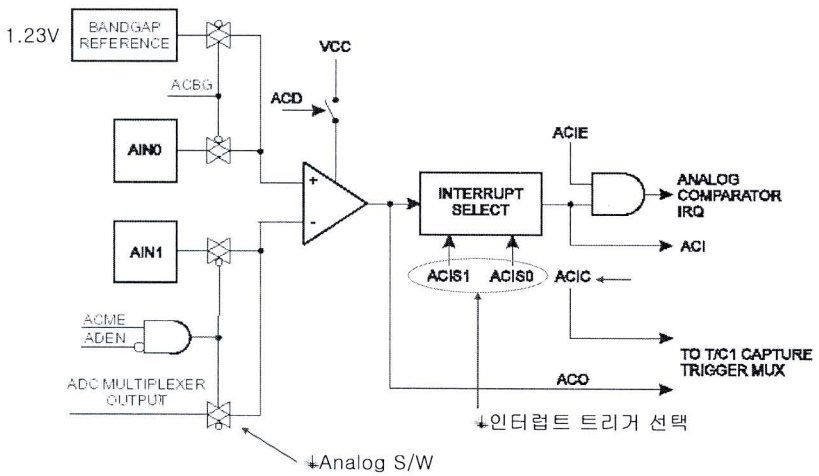
원광대학교 전기전자 및 정보공학부 이 상 설 교수

■ 실험목적

- 아날로그 비교기의 동작을 실험한다.
- 아날로그 비교기 인터럽트 발생을 위한 셋팅과 인터럽트 서비스루틴을 작성한다.

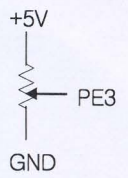
■ 기초지식





Bit	7	6	5	4	3	2	1	0	
	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	ACSR
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	N/A	0	0	0	0	0	

1. 오른쪽 그림과 같이 가변저항을 이용 5V ~ 0V 의 전압변화를 PE3로 연결한다.



2. 아래 프로그램을 작성하여 가동시킨 후 가변저항을 움직이면서 결과를 관찰한다.

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

//세븐세그먼트용 배열설정은 여기에


volatile int         ac_result=0;

SIGNAL(SIG_COMPARATOR)
{       // 의미 : 01 = 0->1,  10 = 1->0
        ac_result = ( ACSR & (1<<ACO) ) ? 1 : 2;
}

void enable_analog_compare_interrupt(void)
{
        // V+ : 1.23V
        // V- : AIN1 PE3 pin
        // Analog compare interrupt enable
        // Interrupt on toggle
        ACSR = (1<<ACBG) | (1<<ACIE);
}

////////////////////////////////////////////////////////////
int     main(void)
{
        // 세븐세그먼트 작동을 위한 설정은 여기에

        enable_analog_compare_interrupt();
        sei();          // start global interrupt

        while(1){
                // ac_result 변수 정보를 세븐세그먼트에 디스플레이 동작은 여기에


        }
}
```