

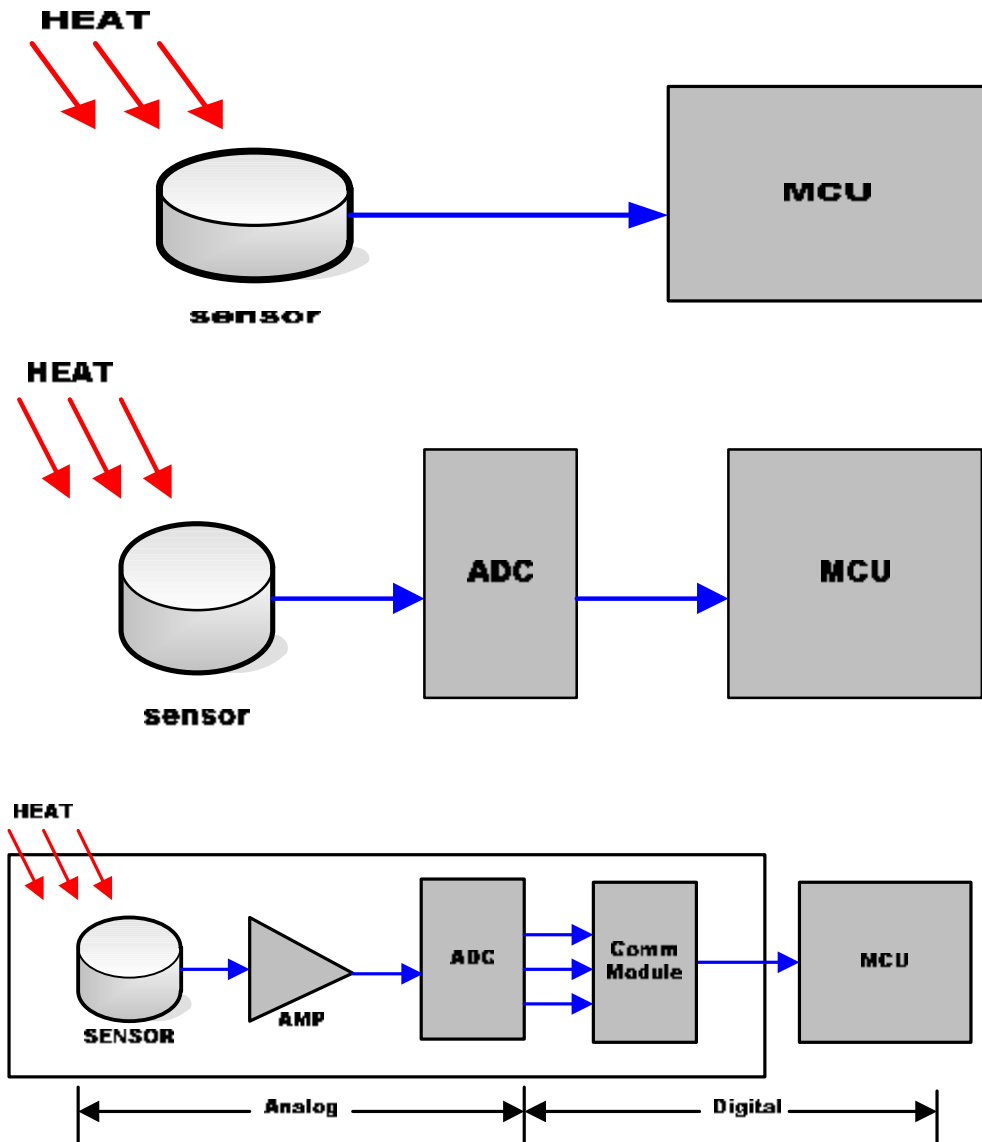
11.1 센서

11.1.1 센서란?

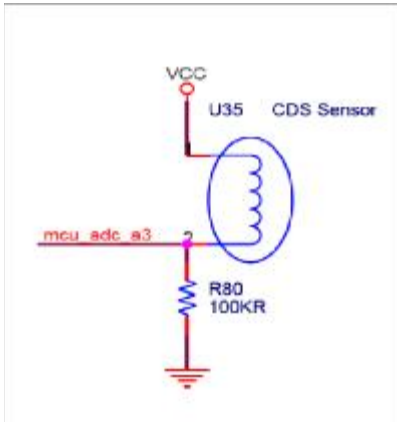
- 여러 환경 요소의 변화를 감지하여 전기적 신호로 변환하는 기기

Q) 온도의 변화에 따른 출력 결과가 무엇일까?

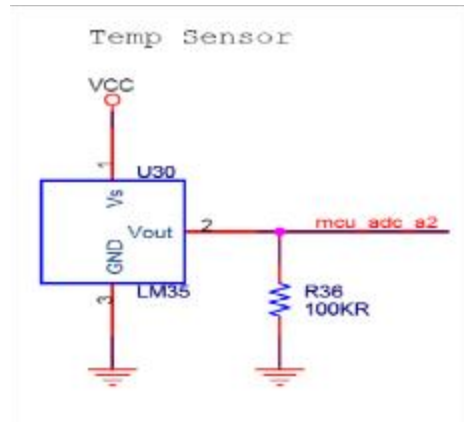
Q) MCU는 이 출력의 변화를 어떻게 알 수 있을까?



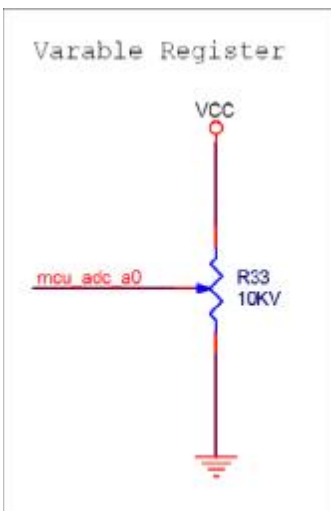
11.1.2 센서의 종류



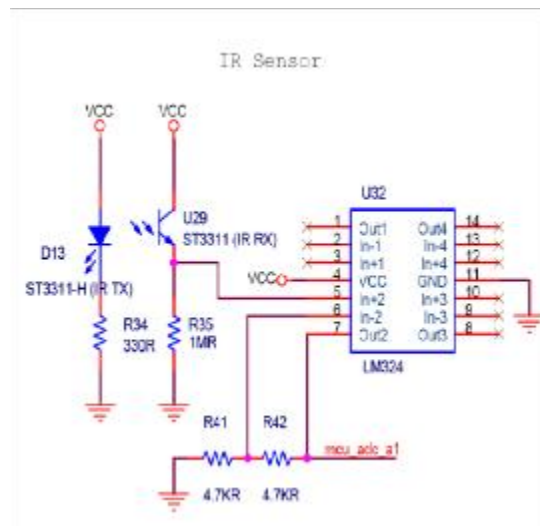
- 조도 센서



- 온도 센서



- 가변 저항



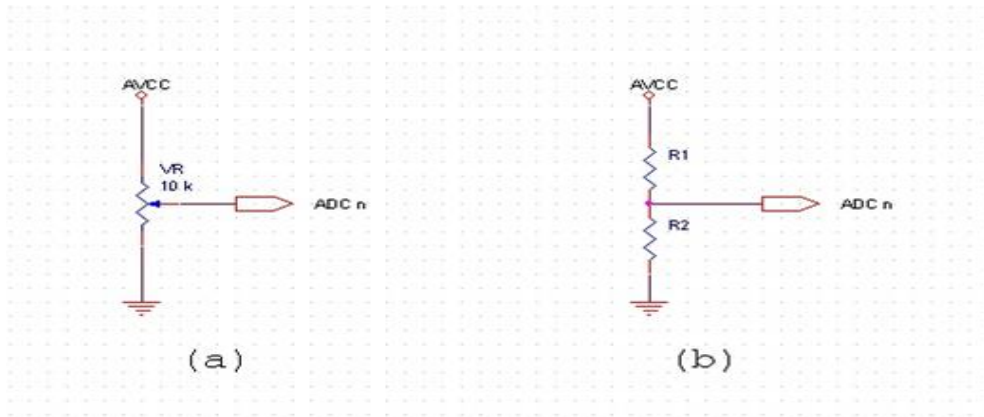
- 적외선 센서

센서들의 ADC 변환 원리는 동일하다. 여기서는 가변 저항을 가지고 ADC하는 원리를 설명함으로써 다른 모든 아날로그 센서의 이해를 돕고자 한다.

ATmega128은 8채널 10비트 분해능의 A/D 컨버터를 내장하고 있다. 이들 8채널의 아날로그 입력은 모두 포트 F와 동일한 단자를 사용하고 있으며 MCU 내부의 아날로그 멀티플렉서에 의해서 선택되도록 되어 있다.

11.1.3 ADC의 원리

- 가변저항의 예



$R1 = 3K\ \Omega$ / $R2 = 7K\ \Omega$ 일 경우 $R1$ 에 걸리는 전압 V_{R1} 은

$$V_{R1} = (3k / (3k + 7k)) * 5V = 1.5\ V$$

$$V_{R2} = (7k / (3k + 7k)) * 5V = 3.5\ V$$

ADC로 입력되는 값은 이 3.5 V가 입력되는데, ADC 결과가 3.5라는 값이 나온다는 얘기가 아니고 1024 분의 얼마에 해당하는 비율로 나오게 된다.

즉, $R2$ 값이 변화하는 비율 (여기서는 5V에서 3.5V로 변하는 비율)에 따라서 ADC의 입력이 1024에 대한 어떤 비율로 변하게 되는 것이다.

가변저항이 1V 변화할 때마다 ADC는 $1024/5 = 204$ 칸의 단위로 움직이게 된다. 따라서, 저항 $R2$ 가 3.5V 이므로 $204.8 * 3.5 = 716.8$ 이라는 값이 나온다. 이 716.8 은 말 그대로 716.8 이지 저항 값이나 전류가 아니라는 것을 기억하자.

11.2 ADC

11.2.1 ADC의 특징

- 8채널
- 10비트 분해능
- 측차비교형(변환시간 수십us의 종속형 A/D Converter)
- 내부 아날로그 멀티플렉서 탑재
- 샘플/홀드회로 탑재로 인하여 A/D 동작동안 전압 고정화
- 단극성 아날로그 입력 / 차동입력 선택
- 차동입력에서 10배~200배의 증폭 A/D 가능

11.2.2 분해능이란?

- 마이컴에서 분해능이란 보통 AD 컨버터에서 주로 사용되는 용어.
- 외부에서 아날로그 신호가 들어 오게 되면 sin파가 입력되게 됨.
- 그런데 예를들어 이 아날로그 신호가 어떻게 변화가 되는지를 1초에 한번씩 체크하는 것과 0.1초마다 체크하는 것, 그리고 0.01 초마다 체크 하는 것은 다름.
- 얼마나 세밀하게 체크를 하는가가 얼마나 "분해"를 하는 것인가와 의미가 같음.
- 좀더 세밀하게 체크할 수 있다면 보다 정확한 데이터를 구할수가 있음. 이것이 분해할 수 있는 능력이라고 함.

11.2.3 ADC의 구성

11.2.3.1 범용 PORTF의 특수 기능

- ADC0 ~ ADC7 : 8채널 10비트 A/D 컨버터의 아날로그 입력단자
- ADC 정확도 성능 향상을 위한 독립 전원 구성
- AVCC : Analog Supply Voltage(VCC의 전압의 $\pm 0.3V$ 유지해야함)
- AGND : Analog Ground (반드시 GND와 연결)
- AREF : Analog Reference Voltage

11.2.3.2 입력 전압의 범위

- 일반 모드 입력전압의 범위($0V \sim V_{ref}$)
- 차동입력 모드 입력전압의 범위($-V_{ref} \sim V_{ref}$)
- V_{ref} 의 범위

전원전압 VCC를 초과할 수 없다.($V_{REF}=V_{GND} \sim V_{CC}$, 내부기준전압 2.56V)

11.3 ADC의 동작

- 단일변환모드(Single conversion mode)
ADC 동작을 한번만 수행하게 된다.
- 프리런닝코드(Free running mode)
ADC 동작을 반복적으로 수행하게 된다

11.3.1 AD 컨버터 초기화 설정

- ADC 활성화 (ADEN=1)
- ADC 클럭설정 (ADPS 2~0)

- ADC 기준전압설정 (REF 1~0)
- ADC 입력채널설정 (MUX 4~0)
- ADC 동작모드설정 (ADFR)
- ADC 변환 완료 인터럽트 활성화 (ADIE)

11.3.2 A/D 컨버터 스타트

- ADC 시작신호 (ADSC)

11.3.3 A/D 변환

- ADC 변환 결과

$$ADC = \frac{V_{IN} \cdot 1024}{V_{REF}}$$

[단극성 입력]

$$ADC = \frac{(V_{POS} - V_{NEG}) \cdot GAIN \cdot 512}{V_{REF}}$$

[차등입력]

11.3.4 A/D 변환 완료

- ADC 변환이 완료되면 변환값을 ADCH/L 에 저장하게 된다.

11.3.5 A/D 변환 완료 인터럽트 요청

- ADC Conversion Complete Interrupt 요청

11.3.6 ADC 상태플래그 셋

- AIDF = 1

11.3.7 ADC 다음 동작 결정

- 단일/연속동작인지 구분하여 다음 동작 수행을 결정한다. (ADFR)

11.4 ADC 관련 레지스터

11.4.1 ADMUX(ADC Multiplexer Selection Register)

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
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	

MUX4	MUX3	MUX2	MUX1	MUX0	선택되는 입력
0	0	0	0	0	ADC0
0	0	0	0	1	ADC1
0	0	0	1	0	ADC2
0	0	0	1	1	ADC3
0	0	1	0	0	ADC4
0	0	1	0	1	ADC5
0	0	1	1	0	ADC6
0	0	1	1	1	ADC7

REFS1	REFS0	Voltage Reference Selection
0	0	AREF에 연결된 전압, 내부 Vref는 꺼짐
0	1	AVCC에 연결된 전압, AREF에 연결된 캐패시터 이용
1	0	Reserved
1	1	내부 2.56V Vref 사용, AREF에 연결된 캐피시터 이용

- ADLAR : ADC Left Adjust Result 변환결과를 좌측으로 몰것인지, 우측으로 몰것인지를 결정

11.4.2 ADCSRA(ADC Control and Status Register)

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
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 -ADEN (ADC Enable) : ADEN = 1 : ADC 활성화

Bit 6 -ADSC (ADC Start Conversion) : ADSC = 1 : ADC의 변환이 시작된다.

(단일 변환모드일때 단 한번만 작동 / 프리런닝모드일때 변환동작반복)

Bit 5 -ADFR (ADC Free Running Select) : ADFR = 1 : 프리런닝모드,
ADFR = 0 : 단일변환모드

Bit 4 -ADIF (ADC Interrupt Flag) : ADC변환완료 인터럽트가 요청되고 그 상태를 이 비트에 표시한다.

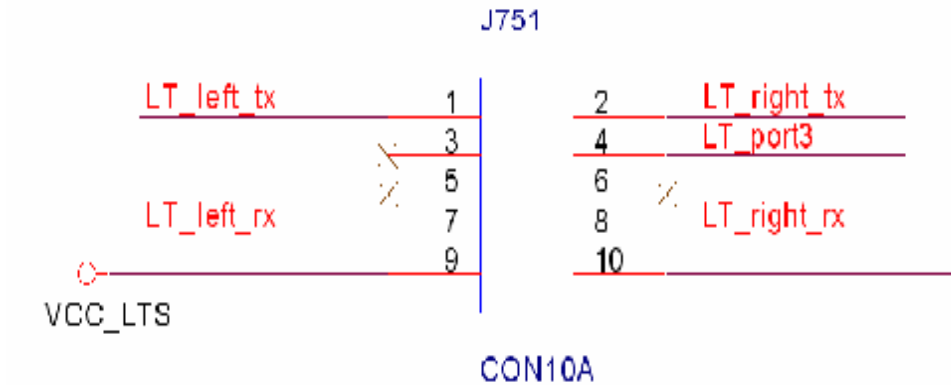
Bit 3 -ADIE (ADC Interrupt Enable) : ADIE = 1 : ADC Interrupt 활성화

ADPS2	ADPS1	ADPS0	분주비
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

11.4.3 ADCH/L (ADC Data Register)

Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	-	-	ADC9	ADC8	ADCH
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
Read/Write	R	R	R	R	R	R	R	R	ADLAR=0
Initial Value	0	0	0	0	0	0	0	0	인 경우

- ADC의 변환결과를 저장한다.



※적외선 발광 센서 동작시키기

AM-IRS 적외선 센서보드를 동작시키려면 먼저 적외선 발광 센서를 동작 시켜야한다.

적외선 발광 센서는 LED와 사용법이 비슷하다. 차이점은 LED에서 가시광선(빨강, 파랑 등 눈에 보이는 색)이 나오는데 반해 적외선 센서는 눈에 보이지 않는 적외선이 나온다. 이것은 디지털 카메라 또는 핸드폰 카메라로 찍을 경우 동작을 잘하는지 확인할 수 있다. 디지털 카메라로 전원이 들어가고 동작이 되는 센서 발광부를 찍어보면 밝게 빛나고 있는 것을 확인할 수 있다.

AM-IRS 적외선 센서보드 에서는 발광 센서를 동작시키기 위해서 해당 포트에 출력으로 내보내야 한다.

※ 적외선 수광 센서 동작시키기

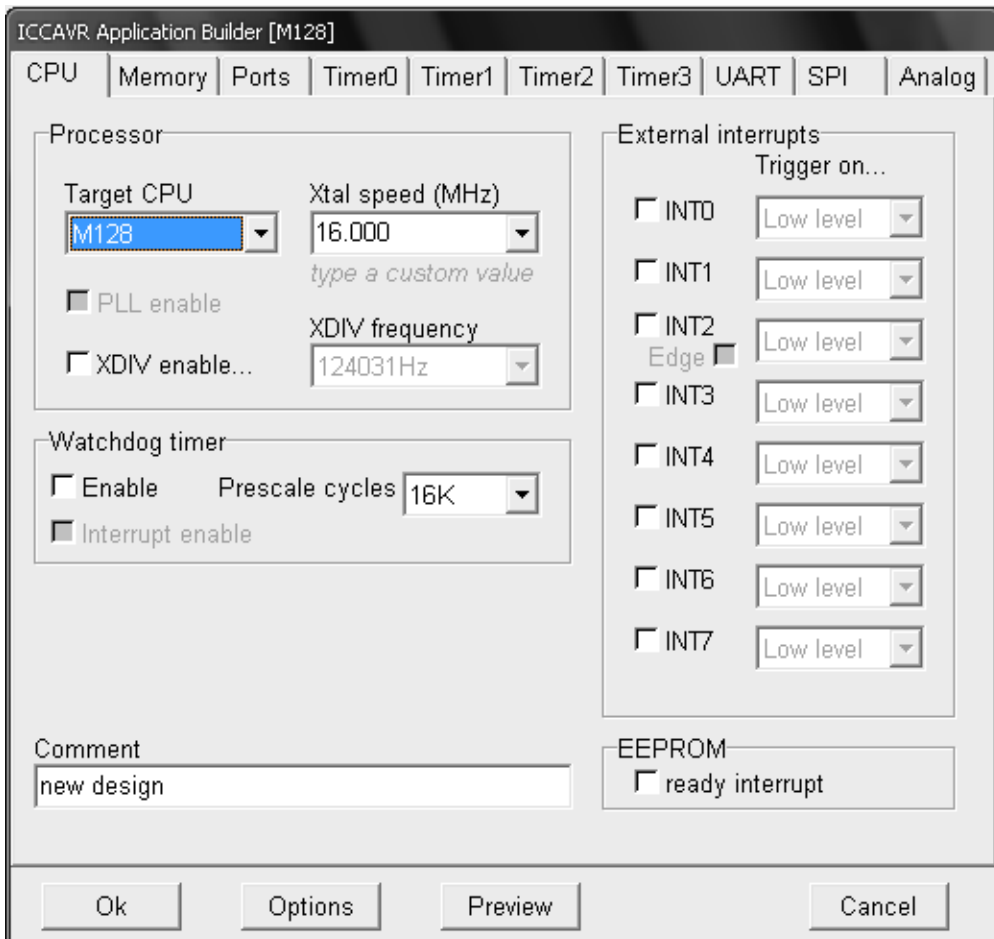
적외선 수광 센서의 출력은 아날로그 신호이다. 이를 ADC(전용 IC 또는 MCU 내부 ADC 핀)에 연결하여 사용한다. 물체가 가까워 지거나 흰색이 앞에 있으면 전압은 올라가고 물체가 멀어지거나 흰색이 멀어지면 전압은 내려가게 된다.

11.6 ADC 실습

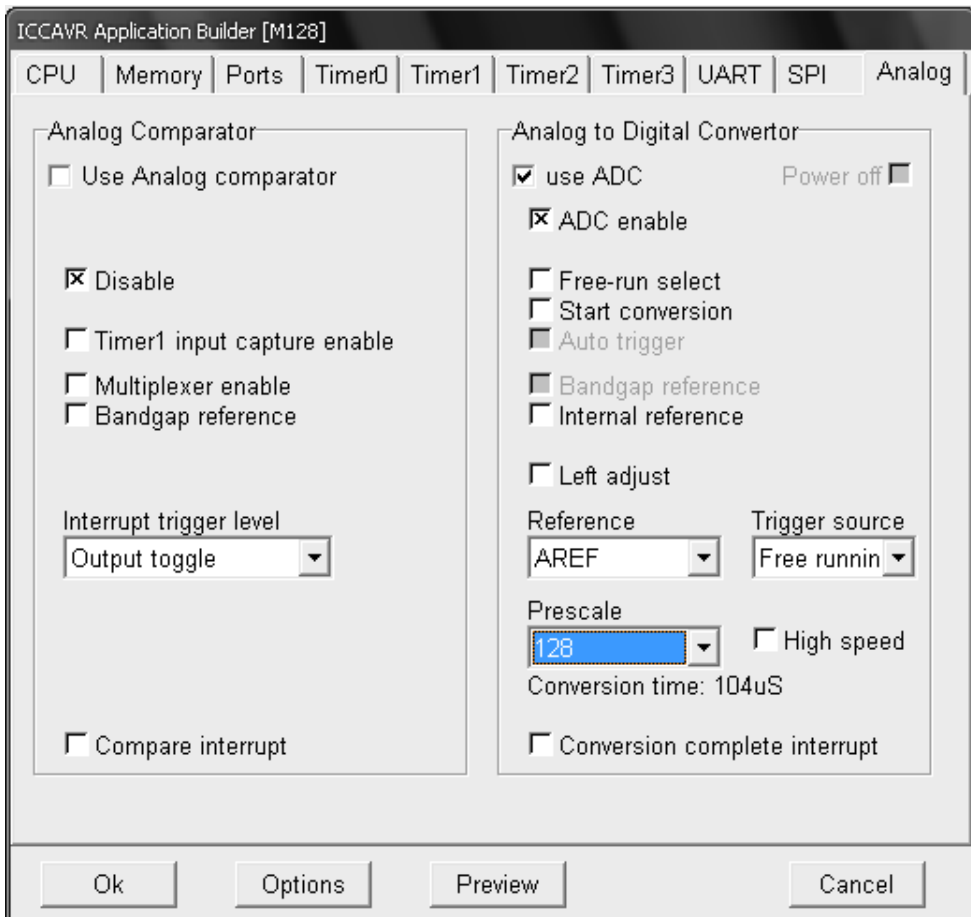
11.6.1 ICCAVR 세팅

ICCAVR을 실행시키고 프로젝트를 생성한다음 Tools >> Application Builder를 선택하여 Target CPU 에 M128, Xtal speed에 16MHz를 선택한다.

경우에 따라 타이머를 사용해야 할 경우가 있지만 생략하고 Analog탭을 선택한다.



다음에 같이 셋팅하고 Options에서 Include "main()"를 선택한 다음 Click.



11.6.2 ADC 실습

- 센서 실습

```
void main() {    //기본적으로 메인함수에서 실행하는 것으로 한다.
volatile unsigned int ch0,ah0,al0; // 전역변수에서 선언하는 것이 좋다.
init_devices();
while(1) {    // ADC()함수형태로 묶어주는 것이 좋지만 이해를 위해..
ADCSRA &= 0x7f; //ADEN 레지스터만 0으로 셋팅
ADMUX = 0x40;   //외부의 AVcc로 입력된 단자를 사용하며, 0번 채널이용
                // 1번 채널은 0x41, 2번 채널은 0x42, ...이런식으로..
ADCSRA |= 0xC0; //AD Start...
al0 = ADCL;     //ADC 하위 데이터 8bit를 al0에 대입
```

```

ah0 = ADCH;      //ADC 상위 2bit를 ah0에 대입
ch0 += al0+ah0*256; //ADC 결과 저장
printf("data : %d \r\n", ch0);
delay(500);
ch0 = 0;
}
}

```

AD 변환 예제 (터미널에출력) - ICCAVR

```

#include <iom128v.h>
#include <macros.h>
typedef unsigned char byte;
volatile int value=0;
void uart0_init(void)
{UCSR0B = 0x00; //disable while setting baud rate
  UCSR0A = 0x00;
  UCSR0C = 0x06;
  UBRR0L = 0x19; //set baud rate lo
  UBRR0H = 0x00; //set baud rate hi
  UCSR0B = 0x18;}
void adc_init(void)
{
  ADCSRA = 0x00; //disable adc
  ADMUX = 0x00; //select adc input 0
  ACSR = 0x80;
  ADCSRA = 0x0F;}
void delay(int cnt)
{
  int i, j;
  for(i=0; i<cnt; i++)
    for(j=0; j<600; j++) ;
}

```

AD 변환 예제 (터미널에출력) - ICCAVR

```
#pragma interrupt_handler adc_isr:22
void adc_isr(void)
{
    int value;
    value=ADCL;
    value|=(int)ADCH<<8;
    value&=0x03ff;
    value=(value/10)+20;
    printf("value=%d\n\r", value);
}
void main()
{
    int val;
    init_devices();
    ADCSRA=0x8e;
    SREG|=0x80;
    printf("test");
    while(1)
    {
        ADMUX=0x40;
        ADCSRA|=0x40;
        delay(1000);
    }
}
```

이상으로 ADC와 센서에 관한 기초 사항들을 마무리 하겠습니다. 보다 많은 소스는 <http://www.NTC.co.kr>에 오시면 Q&A에 등록되어 있습니다. 많은 소스를 접하다보면 자신도 모르게 발전하는 자신을 느낄수 있을것 같습니다. 공부하시다가 궁금하신 점 있으시면, <http://www.NTC.co.kr>로 방문하셔서 Q&A를 통하여 질문하시거나, 자료실이나 강의 등을 참고하시면 됩니다. 감사합니다.

12. 스피커로 소리내기

12.1 스피커로 사운드 내기(원하는 주파수)

타이머 1을 이용하여 주파수를 정해주면 소리를 내는 소스이다. "학교종이 땡땡땡"을 예제로 만들었다. 원리를 살펴보면, Timer1을 16비트 타이머로 셋팅 하면 timer에서 TCNT1H와 TCNT1L 값을 얼마로 주느냐에 따라서 오버플로우가 일어나는 타이밍이 달라지는데, 이 오버플로우 함수에서 스피커 연결선을 토글해주면 원하는 주파수에서 소리가 나게 되는 것이다.

이 때, 카운트 하나당 클럭 하나가 된다.(즉, CPU 클럭을 16MHz를 사용한 여기서는 대략 16번이 1us 정도 된다.)

예를 들어서, 16번에 오버플로우 한번이면 1us에 한 번 토글이고, 160번에 오버플로우 한번이면 10us에 한 번 토글이며, 100 kHz 1,600번에 오버플로우 한번이면 100us에 한 번 토글이며, 10 kHz 가 되는 것이다.

즉, 클럭 몇번당 * 원하는 주파수 = CPU 클럭 이 된다.

그럼, 증가 시키는 기본이 되는 시작점인, TCNT1H 값이랑 TCNT1L 값은 어떻게 구할까?

우선 원하는 주파수를 freq라고 하자. 클럭 XX번에 오버플로우 한번이라면 클럭 몇번당 = CPU 클럭 / 주파수 가 된다.

즉, 적는 값은 0xFFFF- 클럭 몇번당 이 되며,

$TCNT1H = (0xFFFF - \text{클럭 몇번당}) / 0x100;$

$TCNT1L = 0xFFFF - \text{클럭 몇번당} - 0xFF00;$ 가 된다.

이 sound 함수를 쓰려면, delay 함수 정확히 만들려면, 이런걸 동시에 쓰려면, 타이머가 여러개 있어야 한다.

```
#pragma interrupt_handler timer0_ovf_isr:17
```

```
void timer0_ovf_isr(void)
```

```
{ TCNT0 = sori; //reload counter value
```

```
  if(toggle == 0){
```

```
    toggle = 1;
```

```
    ONS; } else{
```

```
    toggle = 0;
```

```
    OFFS; }
```

```
} // 타이머에서 토글 예제..
```

스피커 소리내기 예제 - ICCAVR

```

//ICC-AVR application builder : 2008-01-14 오전 11:11:48
// Target : M128
// Crystal: 16.000Mhz
/*
    sori = 0x40;//도
    sori = 0x49;//레
    sori = 0x52;//미
    sori = 0x56;//파
    sori = 0x5a;//솔
    sori = 0x64;//라
    sori = 0x6e;//시
    sori = 0x73;//도
*/
#include <iom128v.h>
#include <macros.h>
#define ONS                                (PORTG = PORTG|0x10)
#define OFFS                               (PORTG = PORTG&0xef)
#define C                                  0x0e
#define D                                  0x28
#define E                                  0x40
#define F                                  0x49
#define G                                  0x5d
#define A                                  0x6f
#define B                                  0x81
#define C2                                  0x87
#define d_time                            500
#define d_time2                            1000
#define d_time3                            1500
#define d_time4                            2000
volatile int toggle=0, sori=0, status=0;
volatile char keyin=0;
int delaytime[4] = {d_time,d_time2,0,d_time4};

```

스피커 소리내기 예제 계속 - ICCAVR

```

void timer0_init(void)
{
    TCCR0 = 0x00; //stop
    ASSR = 0x00; //set async mode
    TCNT0 = 0x5A; //set count
    OCR0 = 0xA6;
    TCCR0 = 0x03; //start timer
}

#pragma interrupt_handler timer0_ovf_isr:17
void timer0_ovf_isr(void)
{
    TCNT0 = sori; //reload counter value
    if(toggle == 0){
        toggle = 1;
        ONS;
    }else{
        toggle = 0;
        OFFS;
    }
}

void delay(int cnt)
{
    int i, j;
    for(i=0; i<cnt; i++){
        for(j=0; j<1000; j++){
            ; }
        }
}

```

스피커 소리내기 예제 계속 - ICCAVR

```
void schoolsong(){  
    unsigned char  
school[30]={G,G,A,A,G,G,E,G,G,E,E,D,G,G,A,A,G,G,E,G,E,D,E,C};  
    unsigned char length[30]={1,1,1,1,1,1,2,1,1,1,4,1,1,1,1,2,1,1,1,1,4};  
    unsigned int i=0;  
    for(i=0;i<30;i++){  
        sori=school[i];  
        delay(delaytime[length[i]-1]);  
    }  
}  
  
void butterfly(){  
    unsigned char nore[30 ]= {G, E, E, G, E, C, D, E, D, C, E, G, C2, G, C2,  
        G, C2, G, E, G, D, F, E, D, C};  
    unsigned char length[30]={2,1,1,1,1,2,2,1,1,1,1,2,1,1,1,1,1,2,2,1,1,1,2};  
    unsigned int i=0;  
    for(i=0;i<30;i++){  
        sori=nore[i];  
        delay(delaytime[length[i]-1]);  
    }  
}  
  
void sun(){  
    unsigned char nore[50] = {C, E, G, G, G, G, E, F, G, E, C, C, C2, B, C2,  
A, G, A, G, E, C2, B, C2, A, G, A, G,E,A,A,G,G,E,E,G,G,D,D,D,E,C,C,C};  
    unsigned char length[50] = {1,1,1,1,1,1,1,1,1,1,2,1,1,1,1,1,1,1,1,1,1,1,  
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1};  
    unsigned int i=0;  
    for(i=0;i<43;i++){  
        sori=nore[i];  
        delay(delaytime[length[i]-1]);  
    }  
}
```

스피커 소리내기 예제 계속 - ICCAVR

```

void song(){
    unsigned char school[30]={C,D,E,F,G,A,B,C2};
    unsigned char length[30]={1,1,1,1,1,1,1,1};
    unsigned int i=0;
    for(i=0;i<8;i++){
        sori=school[i];
        delay(delaytime[length[i]-1]);
    }
}

void main(void)
{
    init_devices();
    //randomize();
    //insert your functional code here...
    status=1;
    while(1){
        song();
        schoolsong();
        butterfly();
        sun();
    }
}

```

이상으로 AVR 기초 사항들을 마무리 하겠습니다. 보다 많은 자료와 소스들은 <http://www.ntc.co.kr>에 오시면 강좌 및 Q&A에 등록되어 있습니다. AVR에 대한 기초개념을 확실히 하고 응용에 들어간다면 작은 인연(AVR과의 만남)이 인생의 큰 전환점이 될수도 있을거라 감히 말할 수 있습니다. 세상을 살면서 만나는 모든 인연이 다 소중할 수는 없겠지만 만나는 한사람 한사람에게 좋은 이미지로 기억된다면 참 아름다운 인생이 될 것입니다. 공부하시다가 궁금하신 점 있으시면, <http://www.NTC.co.kr>로 방문하셔서 Q&A를 통하여 질문하시거나, 자료실이나 강의 등을 참고하시면 됩니다. 감사합니다. 꾸벅.....