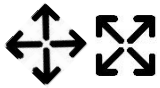
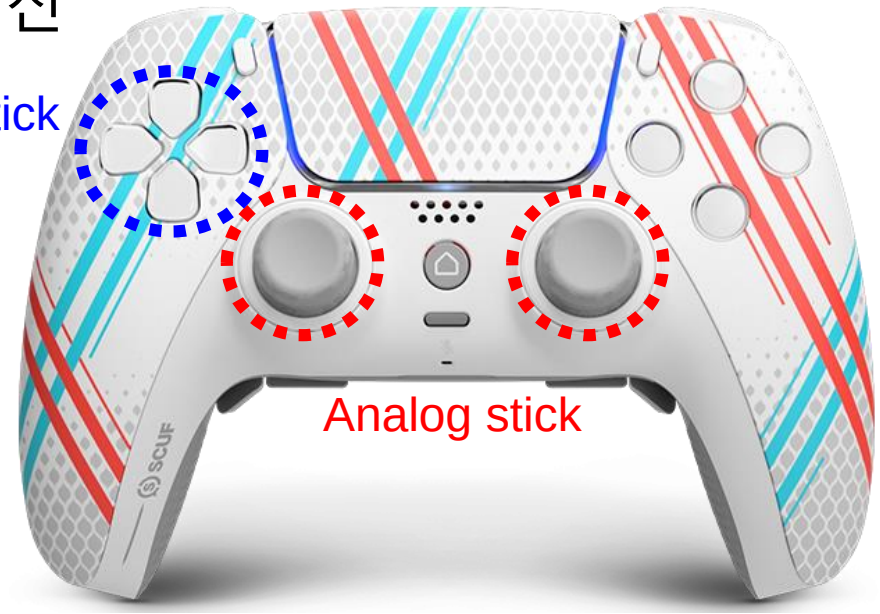
Lecture 10~11

ADC

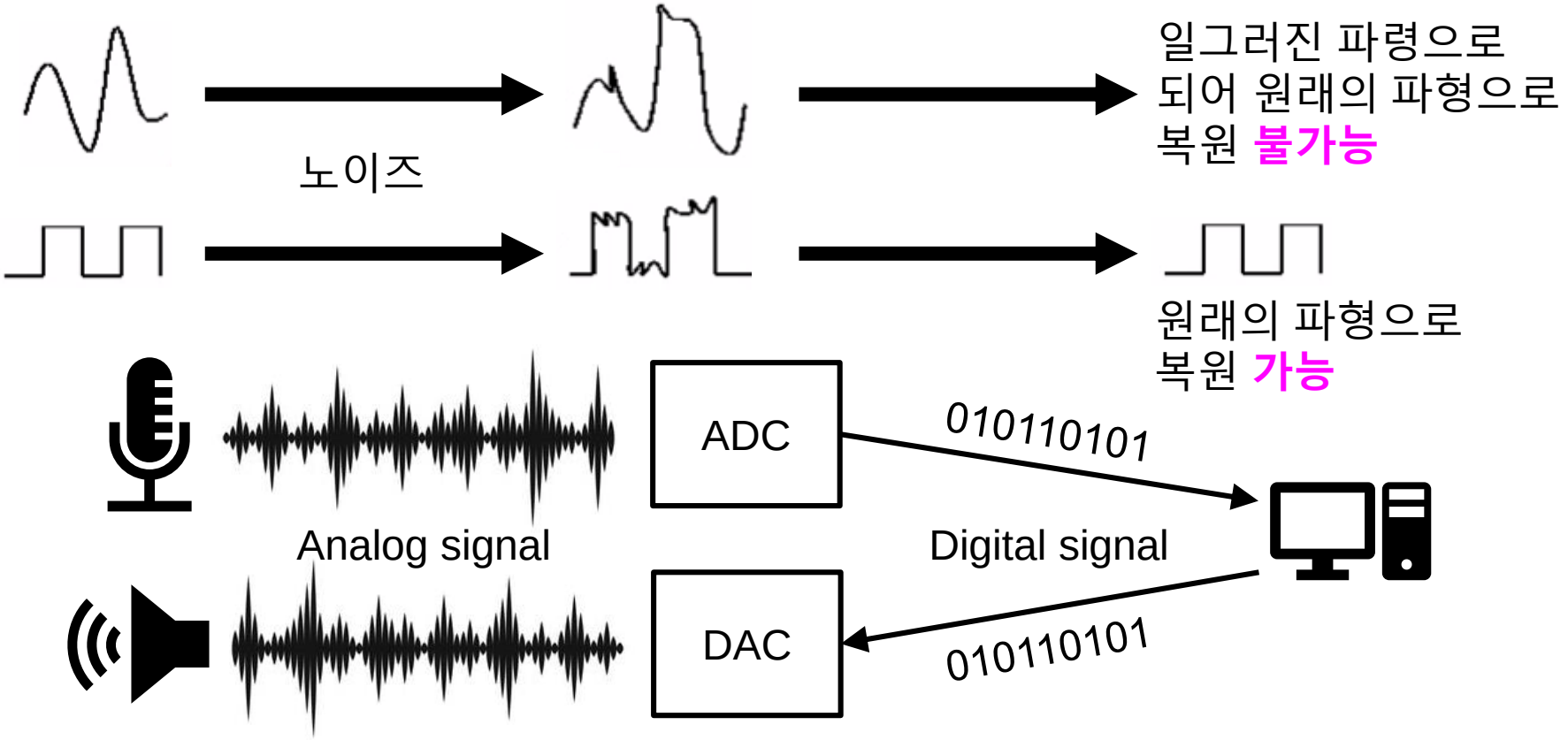
아날로그 및 디지털 신호

- **아날로그 신호**는 연속적으로 변하는 신호로서, 정해진 범위 내의 모든 값이 신호값으로 나타남
- **디지털 신호**는 비연속적으로 변하는 신호로서, 데이터를 일련의 이산 값들로 표현함
- 예: PS5 controller
  - Digital stick : 
  - Analog stick : 모두 방향 이동 가능



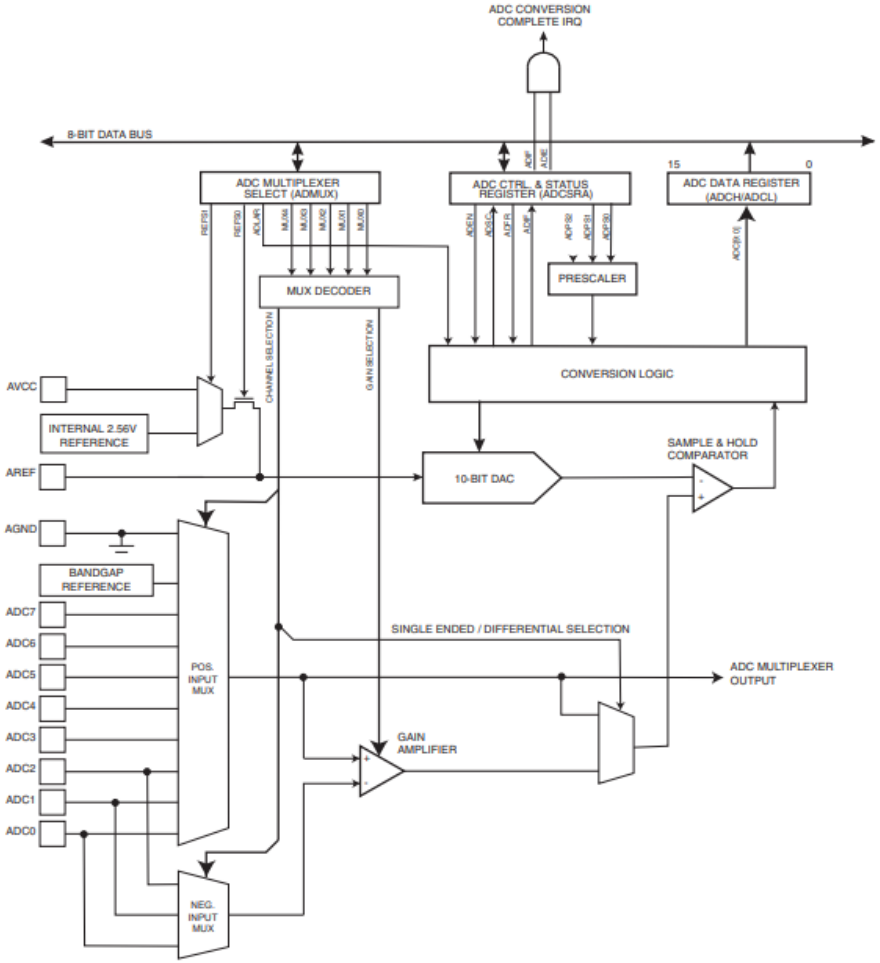
아날로그 디지털 변화

- 아날로그 디지털 변화가 왜 필요한가?



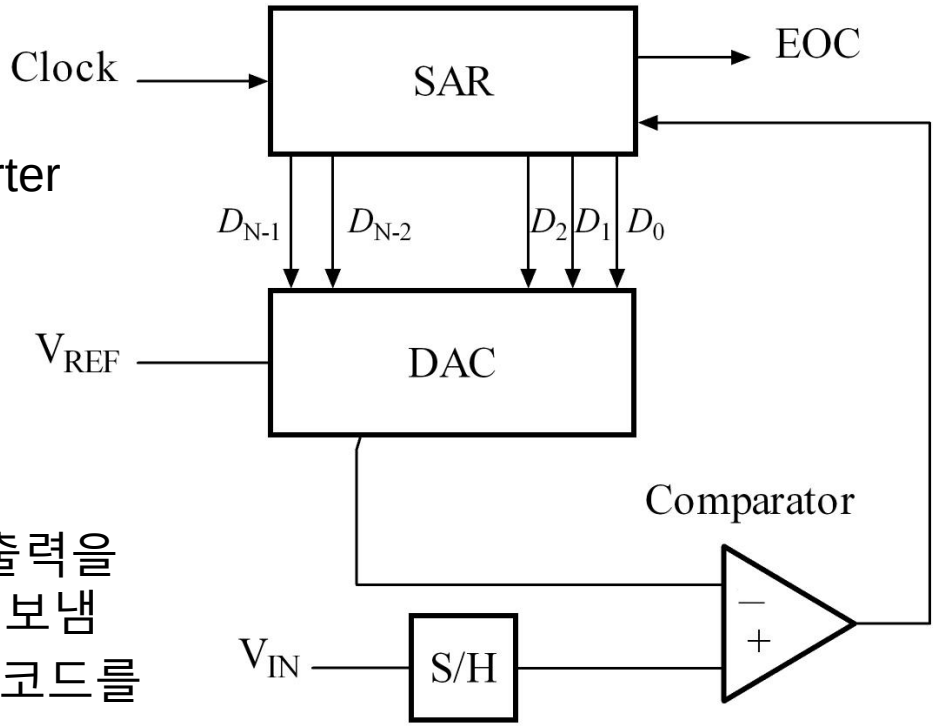
ATmega128의 ADC

- 10-bit resolution
- ±2LSB absolute accuracy
- 13~260μs conversion time
- Up to 76.9kSPS
- 2 differential input channels with 10× and 200× optional gain
- Free-running or single conversion mode
- Interrupt on ADC conversion complete



SAR ADC

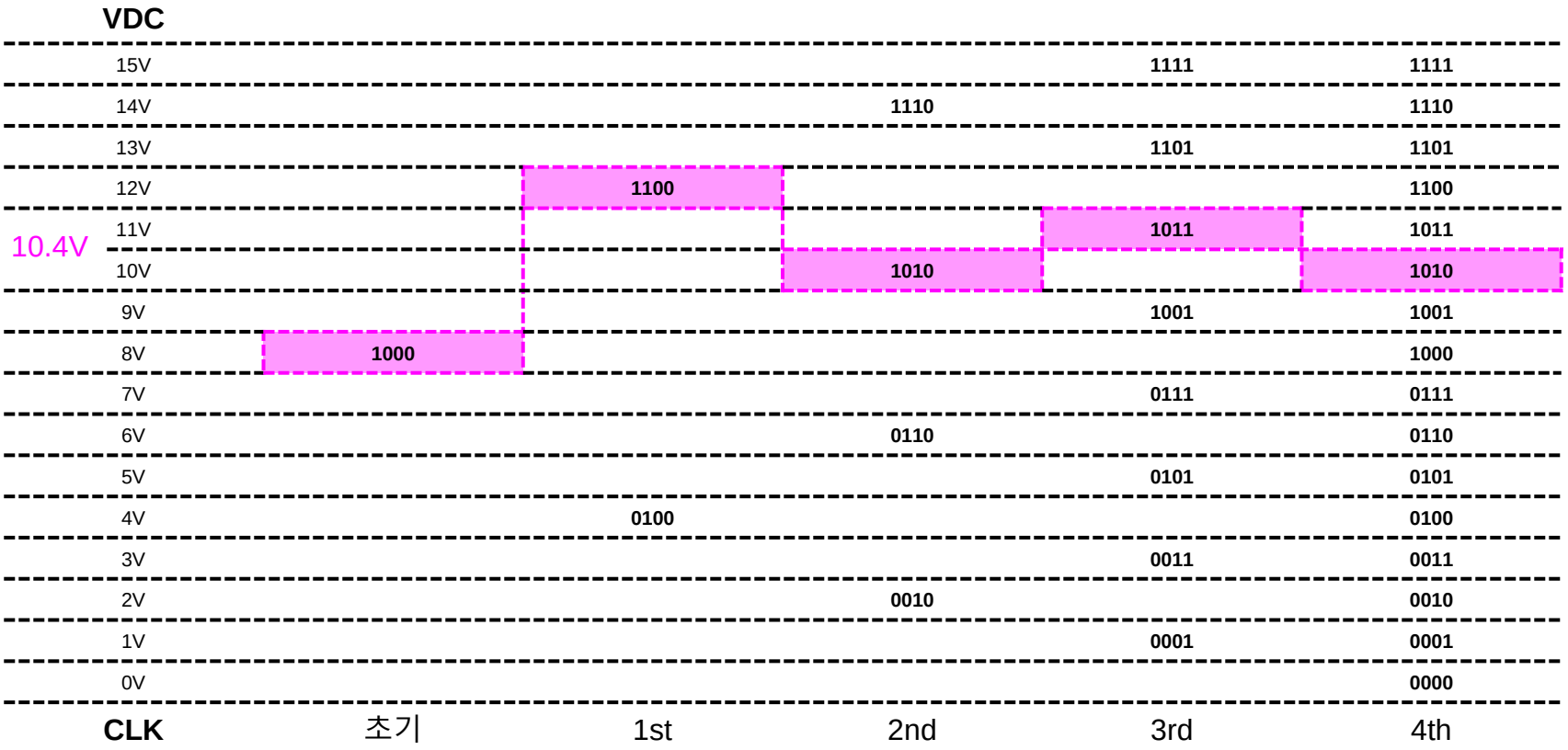
- Successive-approximation ADC
  - SAR : successive approximation register
  - DAC : digital-to-analog converter
  - EOC : end of conversion
  - S/H : sample and hold



- SAR ADC 동작
  - S/H 회로는 V_{IN}을 획득함
  - Comparator는 V_{IN}과 DAC의 출력을 비교하여 비교 결과를 SAR에 보냄
  - SAR는 V_{IN}과 해당하는 2진수 코드를 만들어서 DAC에 보냄
  - DAC는 받은 2진수 코드를 아날로그로 변화하여 Comparator에 보냄

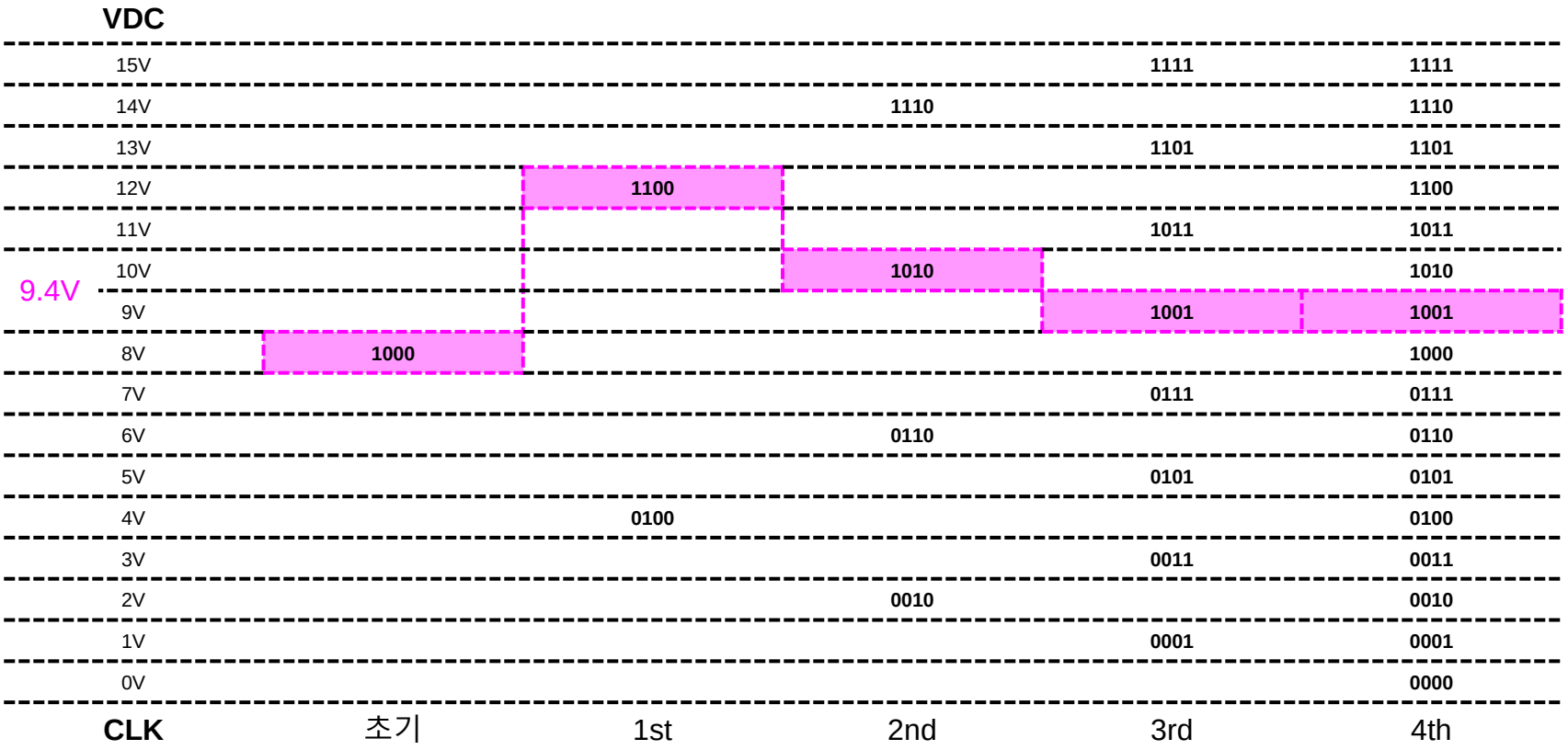
SAR ADC

- 예, 4-bit resolution SAR ADC, 입력 10.4V



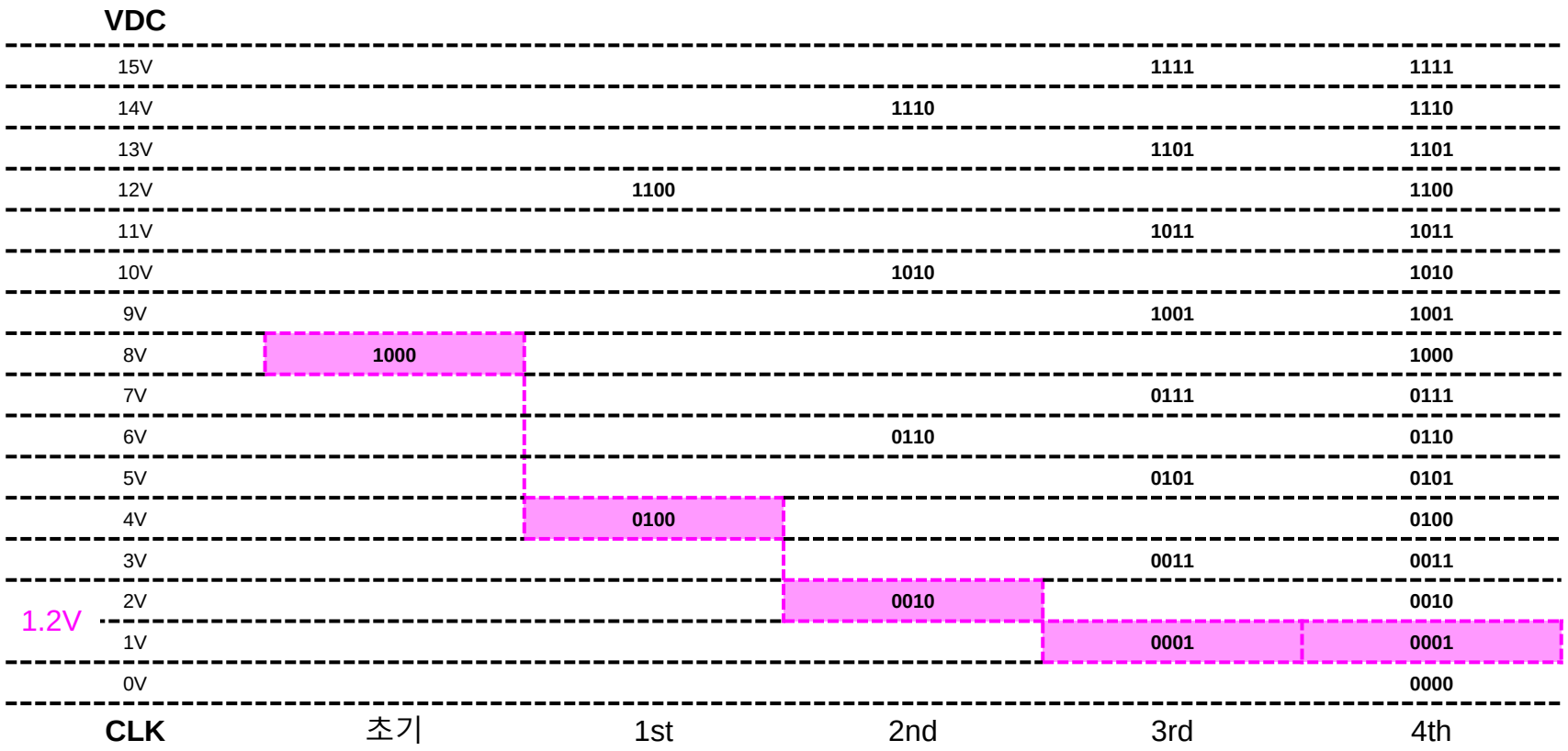
SAR ADC

- 예, 4-bit resolution SAR ADC, 입력 9.4V



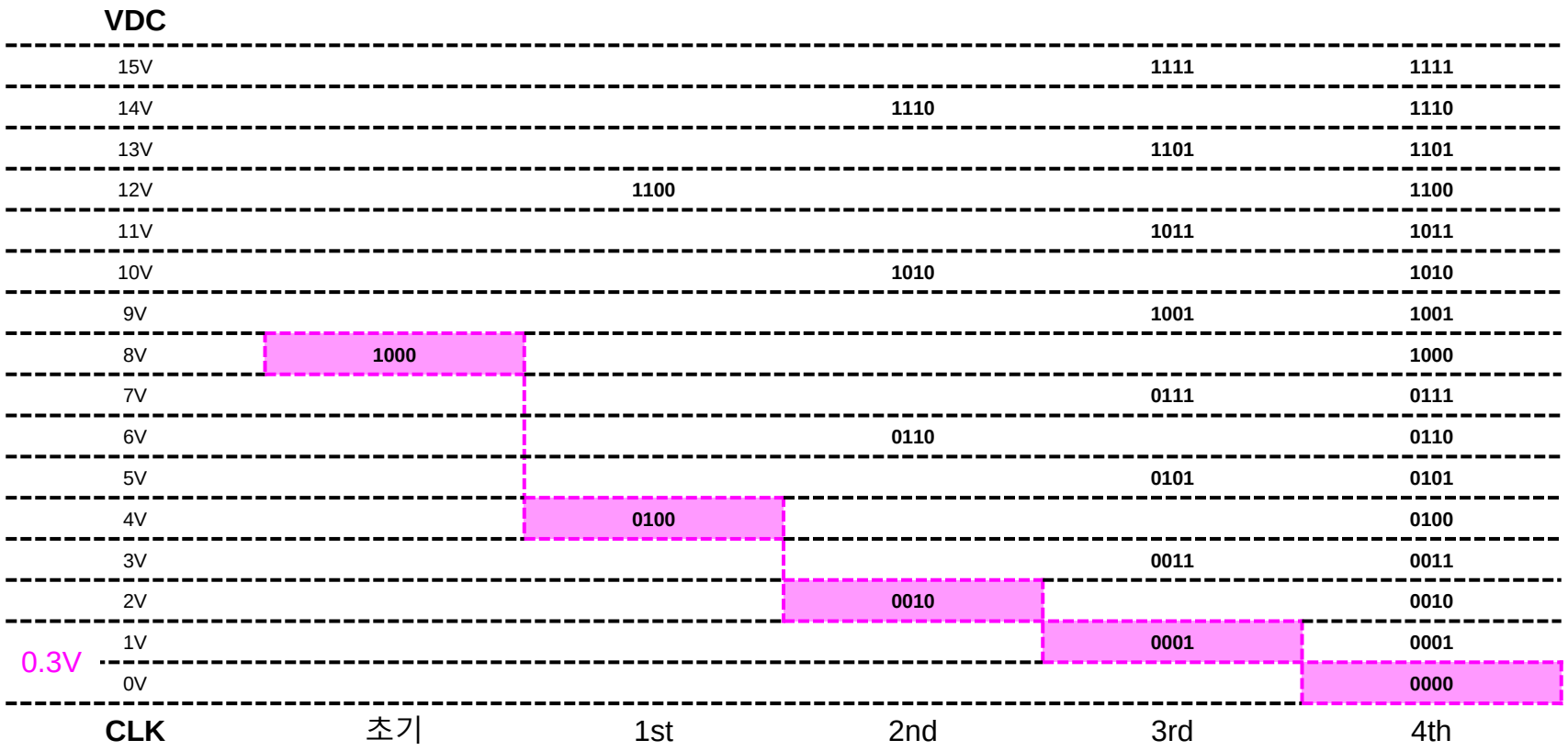
SAR ADC

- 예, 4-bit resolution SAR ADC, 입력 1.2V



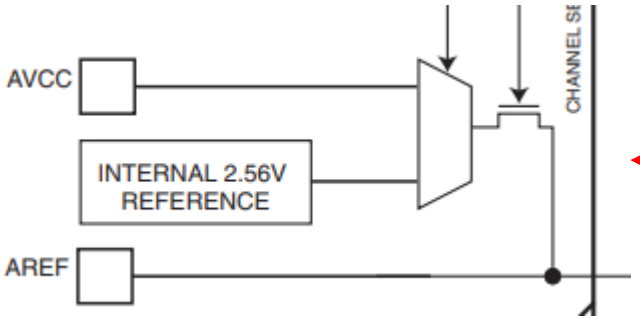
SAR ADC

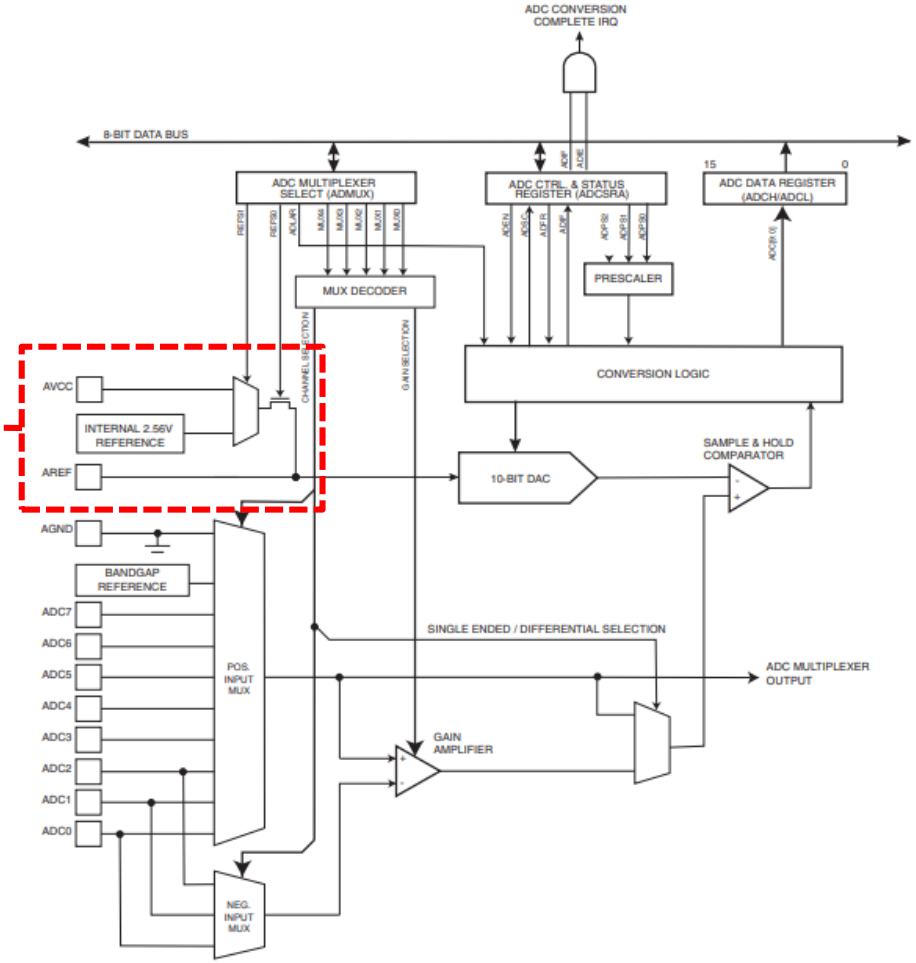
- 예, 4-bit resolution SAR ADC, 입력 0.3V



ATmega128의 ADC

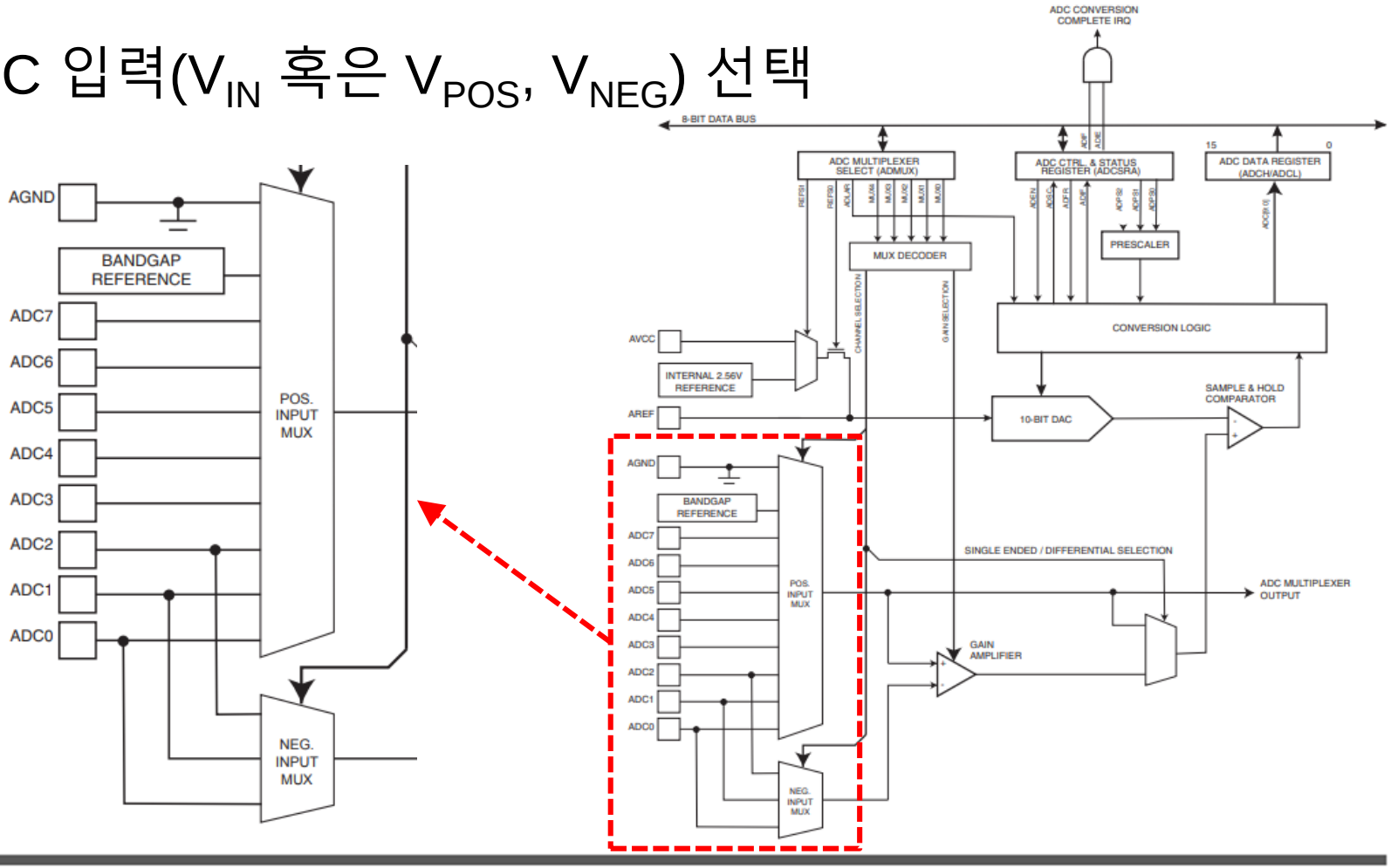
- V_{REF} 선택





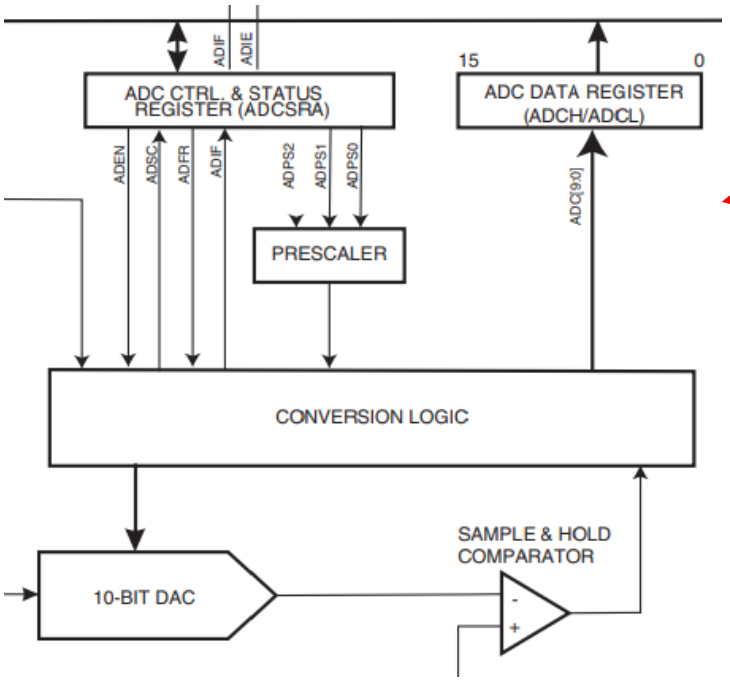
ATmega128의 ADC

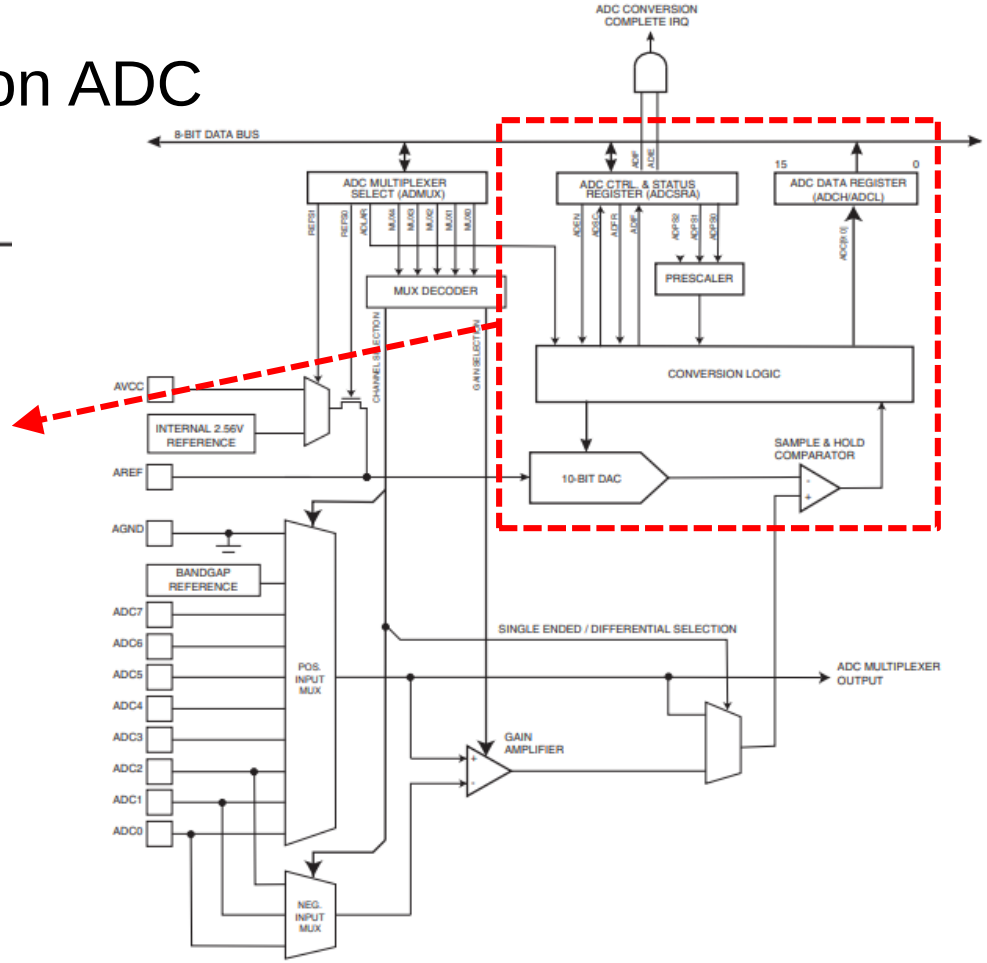
- ADC 입력(V_{IN} 혹은 V_{POS}, V_{NEG}) 선택



ATmega128의 ADC

- Successive-approximation ADC





ATmega128의 ADC

- ADC 변화
  - Single ended conversion(단일 입력 V_{IN})

$$\mathrm{ADC} = \frac{\mathrm{V_{IN}} \cdot 1023}{\mathrm{V_{REF}}}$$

  - Differential channels(양의 입력 V_{POS} 및 음의 입력 V_{NEG})

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

컨트롤 레지스터

- 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	

  - Bits 7:6 – REFS1:0 : ADC에서 사용하는 기준전압을 설정함

REFS1	REFS0	기준전압
0	0	외부의 AREF 단자로 입력된 전압을 사용함
0	1	외부의 AVCC 단자로 입력된 전압을 사용함
1	0	–
1	1	내부의 2.56V를 사용함

컨트롤 레지스터

- 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	

  - Bit 5 – ADLAR : ADC Left Adjust Result
    - 1 : 변환결과값을 ADCH/L에 저장할 때 좌측으로 끝을 맞추어 저장됨

15	14	13	12	11	10	9	8	
ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
ADC1	ADC0	–	–	–	–	–	–	ADCL
7	6	5	4	3	2	1	0	

    - 0 : 변환결과값을 ADCH/L에 저장할 때 우측으로 끝을 맞추어 저장됨

15	14	13	12	11	10	9	8	
–	–	–	–	–	–	ADC9	ADC8	ADCH
ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
7	6	5	4	3	2	1	0	

컨트롤 레지스터

- 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	

  - Bits 4:0 – MUX4:0 : Analog Channel and Gain Selection Bits

MUX4..0	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
00000	ADC0	N/A		
00001	ADC1			
00010	ADC2			
00011	ADC3			
00100	ADC4			
00101	ADC5			
00110	ADC6			
00111	ADC7			

컨트롤 레지스터

MUX4..0	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
01000[1]		ADC0	ADC0	10x
01001		ADC1	ADC0	10x
01010[1]		ADC0	ADC0	200x
01011		ADC1	ADC0	200x
01100		ADC2	ADC2	10x
01101		ADC3	ADC2	10x
01110		ADC2	ADC2	200x
01111		ADC3	ADC2	200x
10000		ADC0	ADC1	1x
10001		ADC1	ADC1	1x
10010	N/A	ADC2	ADC1	1x
10011		ADC3	ADC1	1x
10100		ADC4	ADC1	1x
10101		ADC5	ADC1	1x
10110		ADC6	ADC1	1x
10111		ADC7	ADC1	1x
11000		ADC0	ADC2	1x
11001		ADC1	ADC2	1x
11010		ADC2	ADC2	1x
11011		ADC3	ADC2	1x
11100		ADC4	ADC2	1x

MUX4..0	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
11101		ADC5	ADC2	1x
11110	1.23V (V_{BG})	N/A		
11111	0V (GND)			

컨트롤 레지스터

- ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADFR	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 → 변환이 시작됨
    - 단일 변환 모드(single conversion) : 단 한 번만 작동
    - Free running 모드 : 변환동작 반복
  - Bit 5 – ADFR : ADC Free Running Select
    - 1 : free running 모드
    - 0 : 단일 변환 모드

컨트롤 레지스터

- ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADFR	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 4 – ADIF : ADC Interrupt Flag
    - ADC변환이 완료되어 ADC 데이터 레지스터 값이 갱신되고 나면 이것이 1로 설정되면서 변환완료 인터럽트를 요청함
    - ADIE가 1로 설정되고 SREG의 글로벌 인터럽트 허용 비트가 1로 설정되어 있다면 변환완료 인터럽트가 발생되어 처리됨
    - 인터럽트가 처리되면 ADIF가 0으로 클리어 되며, ADIF에 1을 써넣어도 클리어 됨

컨트롤 레지스터

- ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADFR	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 3 – ADIE : ADC Interrupt Enable
    - ADC 변환완료 인터럽트를 개별적으로 허용함
  - Bits 2:0 – ADPS2:0 : ADC Prescaler Select Bits

ADPS2	ADPS1	ADPS0	분주비
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8

ADPS2	ADPS1	ADPS0	분주비
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

컨트롤 레지스터

- ADCH/L – ADC Data Register

ADLAR = 0

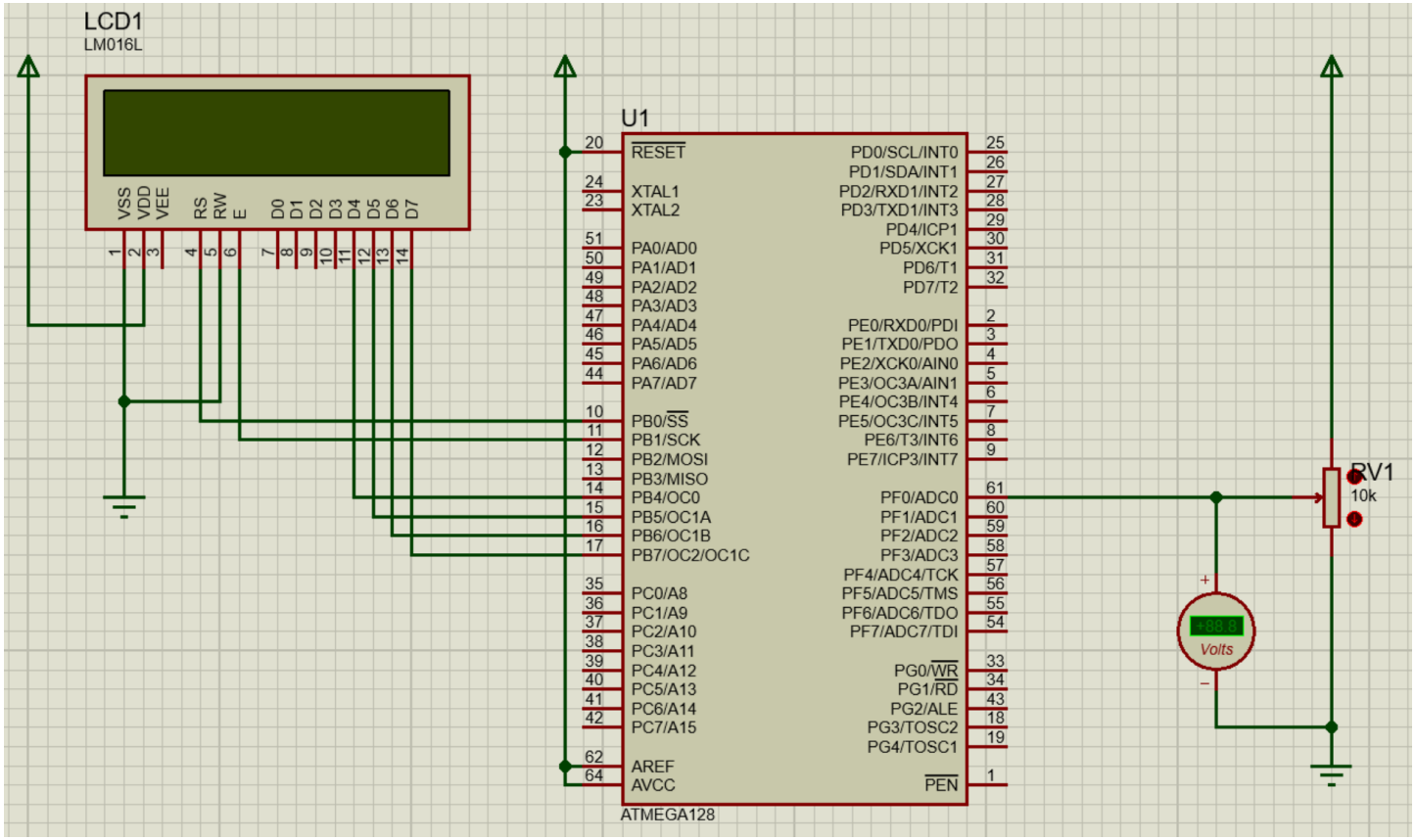
Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	ADC9	ADC8	ADCH
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
	ADC1	ADC0	–	–	–	–	–	–	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

실습1

- Free running 모드에서 채널 0을 통해 가변저항의 전압 측정



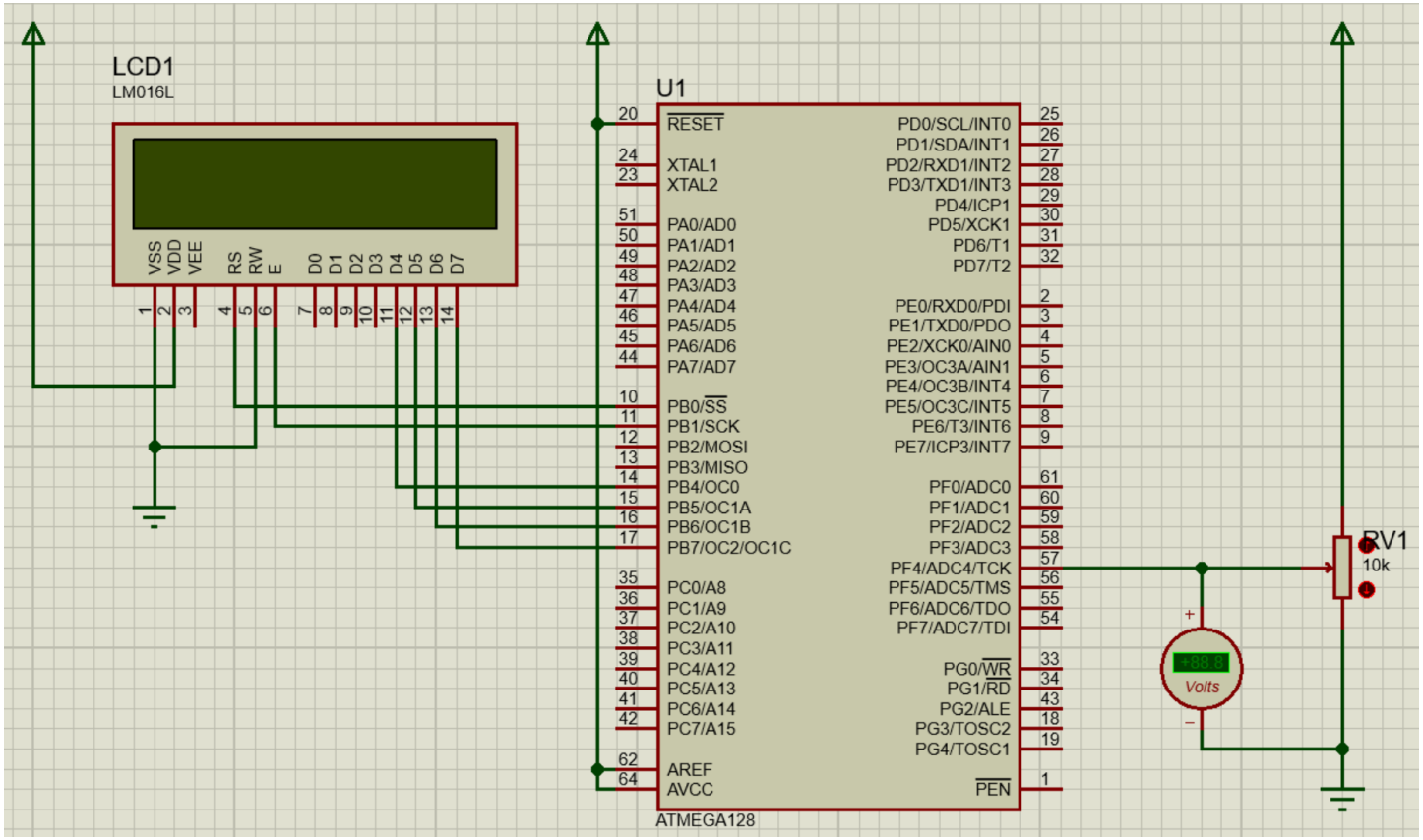
실습1

- 소스코드

```
#include <xc.h>
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>
#include "lcd4.h"

int main(void) {
    char msg[20];
    long result;
    int v;

    DDRB = 0xFF; // LCD control
    DDRF = 0x00; // input mode

    init_lcd4();
    _delay_ms(10);
    writeString_lcd4(0, 0, "Input voltage:");

    ADCSRA = 0xE6; // ADC enable, ADC conversion, free running, prescaler /64
    ADMUX = 0x00; // ADC0, AREF VCC

    while(1) {
        result = ADC;
        result = result*5000/1023;
        v = result;
        sprintf(msg, "val: %d.%03dv", v/1000, v%1000);
        writeString_lcd4(0, 1, msg);
        _delay_ms(500);
    }
}
```

실습2

- 단일 변환 모드에서 채널 4을 통해 가변저항의 전압 측정



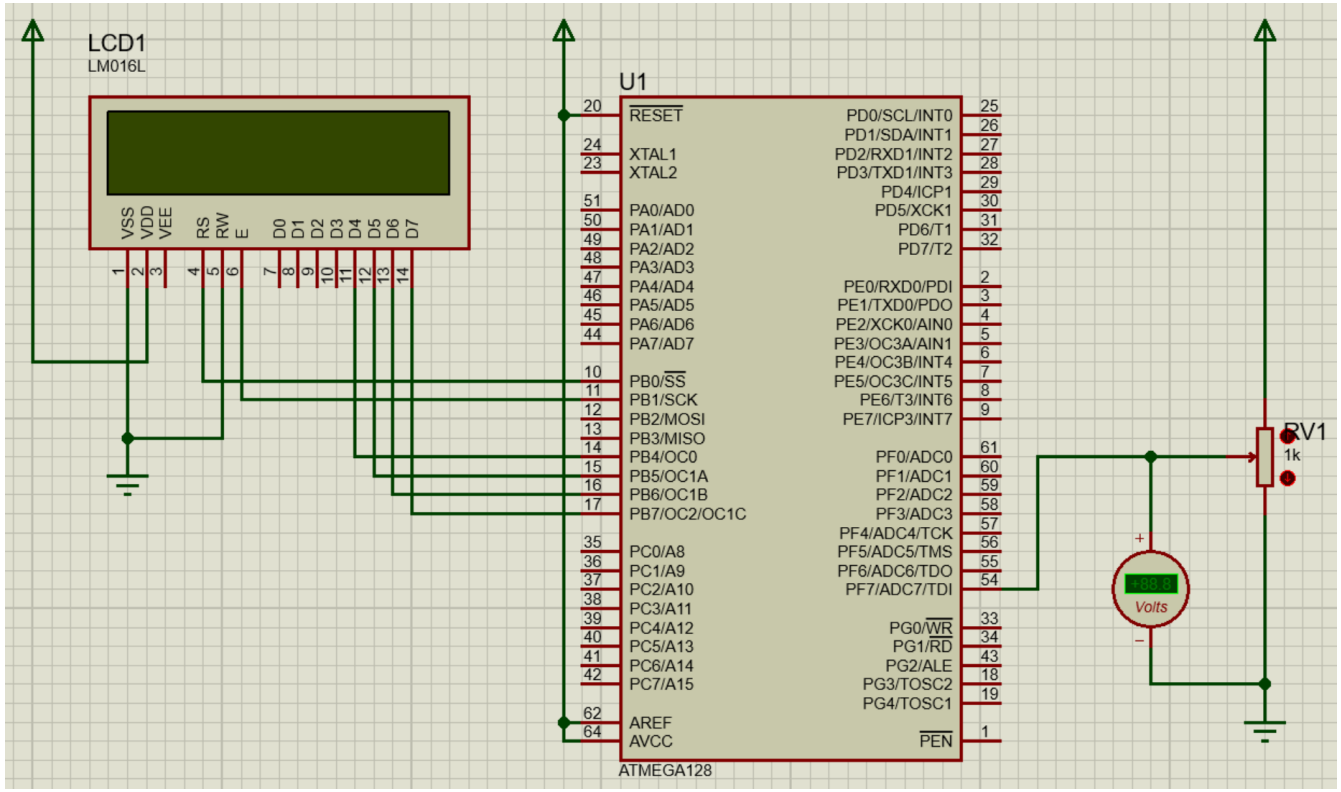
실습2

- 소스코드

```
#include <xc.h>
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>
#include "lcd4.h"

int main(void) {
    long result;
    int v;
    char msg[20];

    DDRB = 0xFF; // LCD control
    DDRF = 0x00; // ADC

    init_lcd4();
    writeString_lcd4(0, 0, "Single Mode");

    ADCSRA = (1<<ADEN) | (1<<ADPS2) | (1<<ADPS1) | (1<<ADPS0);
    ADMUX = 0x04;

    while(1) {
        ADCSRA |= (1<<ADSC); // trigger ADC conversion
        _delay_ms(1);
        result = ADC;
        result = result*5000/1023;
        v = result;
        sprintf(msg, "val: %d.%03dv", v/1000, v%1000);
        writeString_lcd4(0, 1, msg);
        _delay_ms(500);
    }
}
```

실습3

- Free running 모드에서 인터럽트 플래그 인식 방법으로 채널 4을 통해 가변저항의 전압 측정



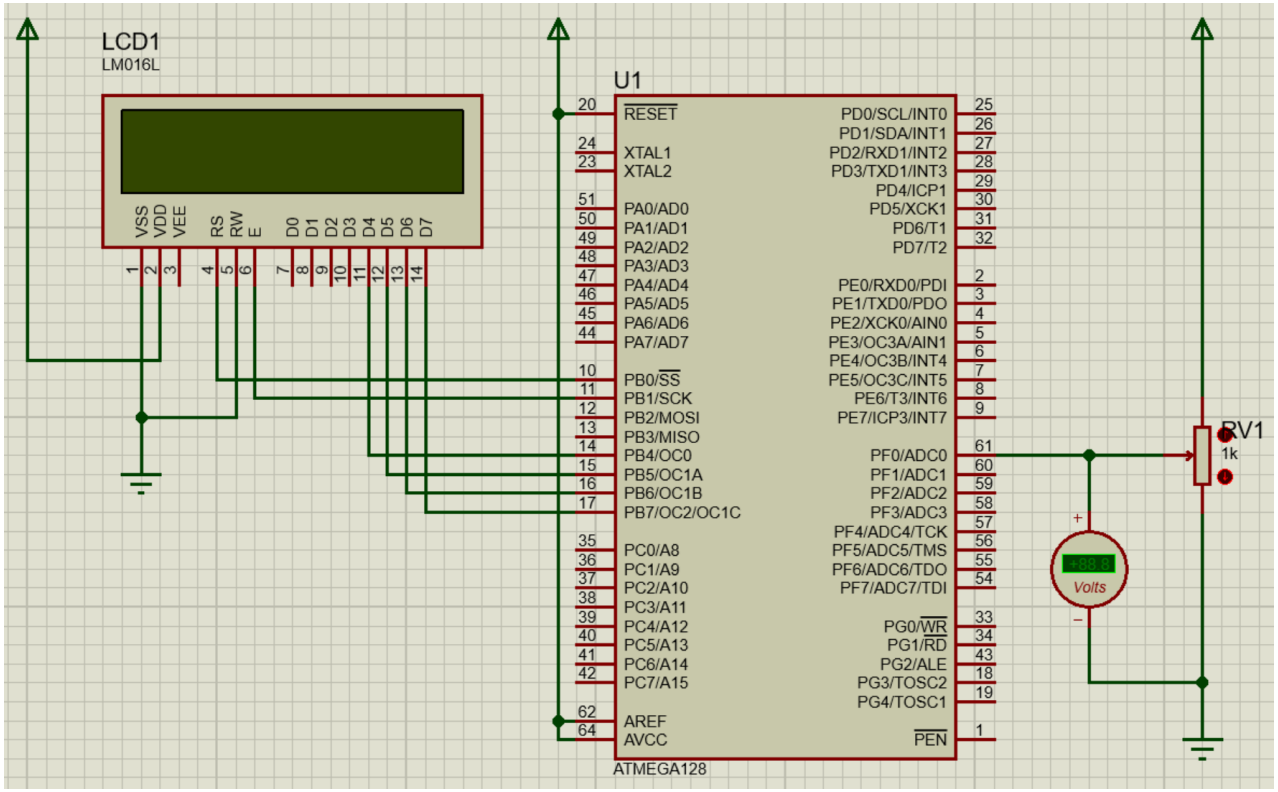
실습3

- 소스코드

```
#include <xc.h>
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>
#include "lcd4.h"

int main(void) {
    long result;
    int v;
    char msg[20];

    DDRB = 0xFF; // LCD control
    DDRF = 0x00; // ADC input

    init_lcd4();
    writeString_lcd4(0, 0, "Free IntFlag Mode");

    ADCSRA = (1<<ADEN) | (1<<ADSC) | (1<<ADFR) | (1<<ADPS2) | (1<<ADPS1);
    ADMUX = 0x07;

    while(1) {
        ADCSRA |= (1<<ADIF);
        while ((ADCSRA & (1<<ADIF))==0x00) {
            result = ADC;
            result = result*5000/1023;
            v = result;
            sprintf(msg, "val: %d.%03dv", v/1000, v%1000);
            writeString_lcd4(0, 1, msg);
            _delay_ms(500);
        }
    }
}
```

실습4

- 단일 변환 모드에서 인터럽트 서비스 루틴 사용 방법으로 채널 0을 통해 가변저항의 전압 측정



실습4

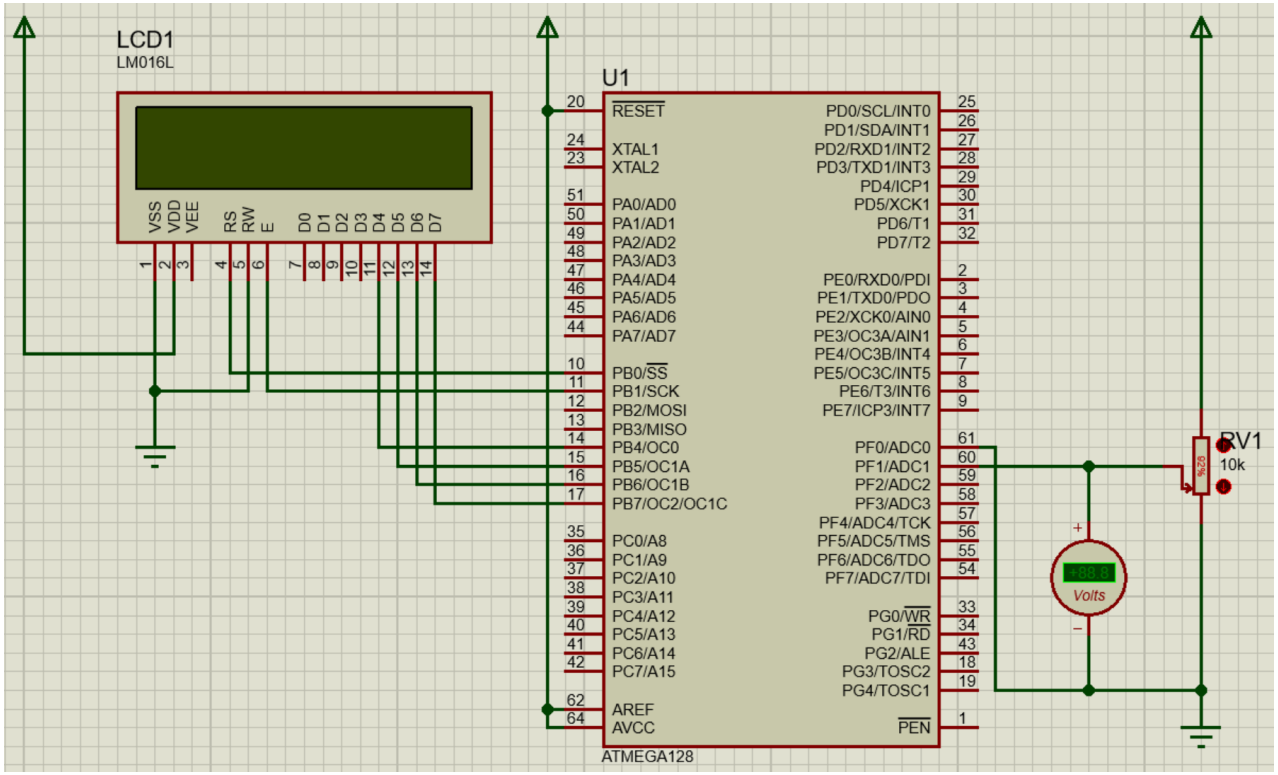
- 소스코드

```
#include <xc.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdio.h>
#include "lcd4.h"

long result;
int v;
char msg[20];

ISR(ADC_vect) {
   result = ADC; r
   esult = result*5000/1023;
   v = result;
   sprintf(msg, "val: %d.%03dv", v/1000, v%1000);
   writeString_lcd4(0, 1, msg);
}
```

```
int main(void) {
   DDRB = 0xFF; // LCD control
   DDRF = 0x00; // ADC input
   init_lcd4();
   writeString_lcd4(0, 0, "Interrupt Mode");
   ADCSRA = (1<<ADEN) | (1<<ADIE) | (1<<ADPS2) | (1<<ADPS1);
   ADMUX = 0x00;
   sei();

   while(1) {
      ADCSRA |= (1<<ADSC);
      _delay_ms(300);
   }
}
```

실습5

- 단일 변환 모드에서 인터럽트 서비스 루틴 사용 방법을 사용하고, 10× 확대모드를 사용하여, 채널 1을 통해 가변저항의 전압 측정



실습5

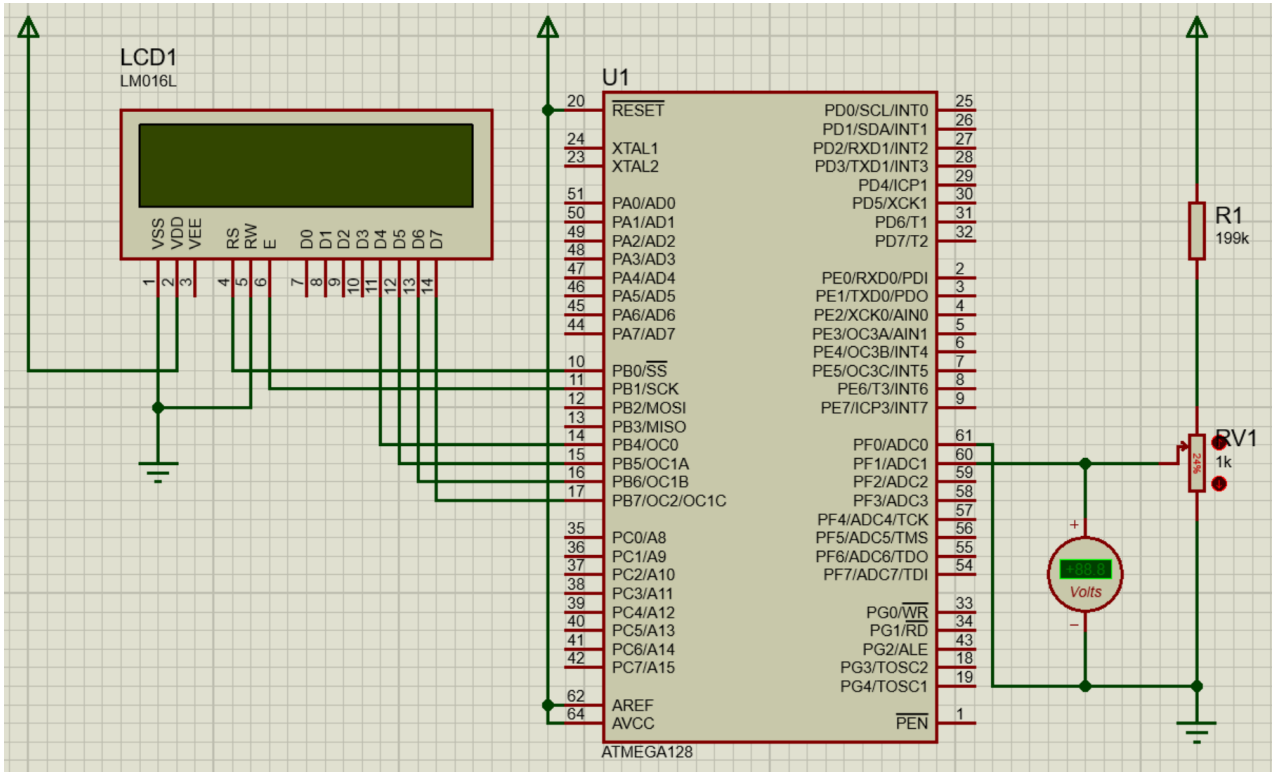
- 소스코드

```
#include <xc.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdio.h>
#include "lcd4.h"

long result;
int v;
char msg[20];


ISR(ADC_vect) {
   result = ADC;
   result = result*5000/1023;
   v = result;
   sprintf(msg, "val: %d.%04dv", v/10000, v%10000);
   writeString_lcd4(0, 1, msg);
}
```

```
int main(void) {
   DDRB = 0xFF; // LCD control
   DDRF = 0x00; // ADC input

   init_lcd4();
   writeString_lcd4(0, 0, "ADC 01001 Mode");

   ADCSRA = (1<<ADEN) | (1<<ADIE) | (1<<ADPS2) | (1<<ADPS1);
   ADMUX = 0x09;
   sei();

   while(1) {
      ADCSRA |= (1<<ADSC);
      _delay_ms(300);
   }
}
```

실습6

- 단일 변환 모드에서 인터럽트 서비스 루틴 사용 방법을 사용하고, 200× 확대모드를 사용하여, 채널 1을 통해 가변저항의 전압 측정



실습6

- 소스코드

```
#include <xc.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdio.h>
#include "lcd4.h"

long result;
int v;
char msg[20];


ISR(ADC_vect) {
   result = ADC;
   result = result*2500/1023; }
   v = result;
   sprintf(msg, "val: 0.%05dv", v%100000);
   writeString_lcd4(0, 1, msg);
}
```

```
int main(void) {
   DDRB = 0xFF; // LCD control
   DDRF = 0x00; // ADC input

   init_lcd4();
   writeString_lcd4(0, 0, "ADC 01011 (x200)");

   ADCSRA = (1<<ADEN) | (1<<ADIE) | (1>>ADPS2) | (1<<ADPS1);
   ADMUX = 0x0B;
   sei();

   while(1) {
      ADCSRA |= (1<<ADSC);
      _delay_ms(300);
   }
```