

## 과제 1 번역본

- 10비트 해상도
- 0.5LSB 적분 비선형성
- $\pm 2$ LSB 절대 정확도
- 13 - 260  $\mu$  변환 시간
- 최대 76.9kSPS(최대 해상도에서 최대 15kSPS)
- 8개의 다중화된 단일 종단 입력 채널
- 7개의 차동 입력 채널
- 옵션 게인이 10배 및 200배인 2개의 차동 입력 채널
- ADC 결과 판독에 대한 왼쪽 조정(옵션)
- 0 - VCC ADC 입력 전압 범위
- 선택 가능한 2.56V ADC 기준 전압
- 자유 실행 또는 단일 변환 모드
- ADC 변환 완료 시 인터럽트
- 슬립 모드 노이즈 캔슬러 Atmel® AVR®ATMEGA128은 10비트 연속 근사 ADC를 갖추고 있습니다.

ADC는 포트 F의 핀으로 구성된 8개의 싱글 엔드 전압 입력을 허용하는 8채널 아날로그 멀티플렉서에 연결되며, 싱글 엔드 전압 입력은 0V(GND)를 나타냅니다. 이 장치는 또한 16개의 차동 전압 입력 조합을 지원합니다. 차동 입력 중 2개는 (ADC1, ADC0 및 ADC3, ADC2)에는 프로그래밍 가능한 게인 스테이지가 장착되어 있어, 다음과 같은 기능을 제공합니다. 차동 입력 전압에 대한 0dB(1x), 20dB(10x) 또는 46dB(200x)의 증폭 단계 A/D 변환 전에. 7개의 차동 아날로그 입력 채널이 공통의 음(-)을 공유합니다. 단자(ADC1), 기타 임의의 ADC 입력을 포지티브 입력 단자로 선택할 수 있습니다. 1x인 경우 또는 10배 이득을 사용하면 8비트 해상도를 기대할 수 있습니다. 200배 이득을 사용하면 7비트 해상도를 기대할 수 있습니다. 기대돼요. ADC에는 ADC에 대한 입력 전압이 다음과 같은 것을 보장하는 샘플 및 홀드 회로가 포함되어 있습니다. 변환 중에 일정한 레벨로 유지됩니다. ADC의 블록 다이어그램은 그림 108에 나와 있습니다. ADC에는 별도의 아날로그 공급 전압 핀인 AVCC가 있습니다. AVCC의 차이는 다음과 같습니다. VCC에서  $\pm 0.3$ V. 연결 방법은 236페이지의 "ADC Noise Canceller" 단락을 참조하십시오. 핀을 켜다. 명목

상 2.56V 또는 AVCC의 내부 기준 전압이 온칩에 제공됩니다. 전압 기준은 더 나은 노이즈 성능을 위해 커패시터에 의해 AREF 핀에서 외부로 분리될 수 있습니다.

공부한 내용: 아날로그 값을 최대 4개까지 가져올 수 있고 그중 핀에 따라 2의 10제곱 해 상도부터 2의 7제곱 해상도까지 신호를 증폭하여 가져올 수 있음. 전류가 일정하게 공급되지 않는 5v의 vcc와 달리 avcc포트는 2.56v의 전압을 일정하게 공급하여 값의 신뢰성을 보장할 수 있다. 또한 vcc에 연결되었는 gnd와 구분하여 사용할 수 있는 avcc포트가 있음

과제 2 소스코드

```
#define F_CPU 16000000

#include <avr/io.h>

#include <avr/interrupt.h>

#include <util/delay.h>

#include "LCD_Text.h"

int main()

{

    DDRF = 0x00;

    ADMUX = 0x40;

    ADCSRA = 0x87;

    DDRA = 0xFF;

    lcdInit();

    lcdClear();

    while(1)

    {

        unsigned int adcValue = 0;

        unsigned char channel = 0x00;

        ADMUX = 0x40 | channel;

        ADCSRA |= 0x40;
```

```

while((ADCSRA & 0x10) == 0);

{
    adcValue = ADC;
    _delay_ms(100);
}

float volt = adcValue * 5 / 1024;

int num = (int)volt;

int _num = (volt - num) * 1000;

lcdNumber(0, 0, adcValue);

lcdNumber(0, 7, num);

lcdString(0, 8, ".");

lcdNumber(0, 9, _num);

lcdString(1, 0, "19th_jj");

_delay_ms(100);

lcdClear();

if (adcValue > 896 && adcValue <= 1024)

{
    PORTA = 0b01111111;
}

else if (adcValue > 768 && adcValue <= 896)

{
    PORTA = 0b10111111;
}

else if (adcValue > 640 && adcValue <= 768)

{

```

```

        PORTA = 0b11011111;
    }
    else if (adcValue > 512 && adcValue <= 640)
    {
        PORTA = 0b11101111;
    }
    else if (adcValue > 384 && adcValue <= 512)
    {
        PORTA = 0b11110111;
    }
    else if (adcValue > 256 && adcValue <= 384)
    {
        PORTA = 0b11111011;
    }
    else if (adcValue > 128 && adcValue <= 256)
    {
        PORTA = 0b11111101;
    }
    else if (adcValue >= 0 && adcValue <= 128)
    {
        PORTA = 0b11111110;
    }
}
}

```

과제 3 소스코드

```
#define F_CPU 16000000
```

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

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

```
#include <util/delay.h>
```

```
#include "LCD_Text.h"
```

```
int main()
```

```
{
```

```
    DDRA = 0xff;
```

```
    DDRD = 0x00;
```

```
    int num1 = 0;
```

```
    int num2 = 0;
```

```
    int _case = 0;
```

```
    int result = 0;
```

```
    lcdInit();
```

```
    lcdClear();
```

```
    while(1)
```

```
    {
```

```
        if((PIND & 0x01) == 0)
```

```
        {
```

```
            _delay_ms(200); //다중 입력 방지
```

```
            num1++;
```

```
        }
```

```

if((PIND & 0x02) == 0)
{
    _delay_ms(200);
    _case++;
    _case = _case % 4;
}

```

```

if((PIND & 0x04) == 0)
{
    _delay_ms(200);
    num2++;
}

```

```

if((PIND & 0x08) == 0)
{

```

lcdClear();//예외처리 ex)이전 계산 결과가 3자리 넘어가는데 결과가 2자리수일경우 마지막 숫자가 지워지지 않고 그래도 남음

```

    _delay_ms(200);
    switch(_case)
    {
        case 0:
            result = num1 + num2;
            break;
        case 1:
            result = num1 - num2;
            break;
        case 2:

```

```
        result = num1 * num2;

        break;

    case 3:

        result = num1 / num2;

        break;

    }

}
```

```
lcdNumber(0, 0, num1);

switch(_case)
{

    case 0:

        lcdString(0, 3, "+");

        break;

    case 1:

        lcdString(0, 3, "-");

        break;

    case 2:

        lcdString(0, 3, "*");

        break;

    case 3:

        lcdString(0, 3, "/");

        break;

}

lcdNumber(0, 4, num2);
```

```
    lcdString(0, 7, "=");
```

```
    lcdNumber(0, 8, result);
```

```
  }
```

```
}
```