

**전자회로실험 보고서**

학과 전자공학과

학번 1218165

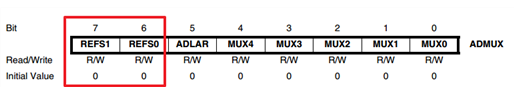
이름 김재욱

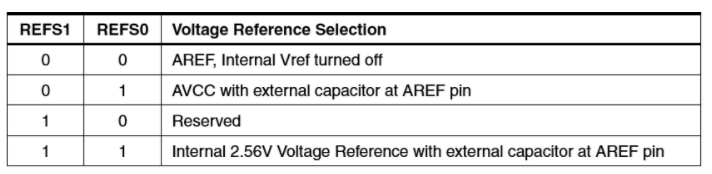
제출일 2020-11-29

**실습 이론**

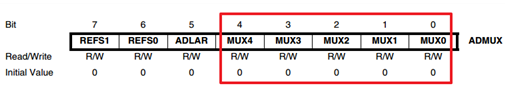
기준전압 설정

ADC가 동작하기 위해서 기준전압을 설정해야하며 기준전압을 설정하기 위한 3개의 핀(AREF, AVCC, GND)이 존재한다. 각 AREF, AVCC 핀으로 들어오는 입력값으로 기준전압을 설정할 수 있으며 내부의 2.56V를 기준전압으로 사용 할 수 있다.

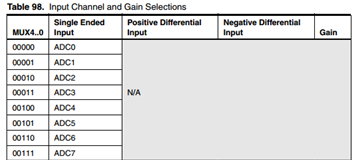




ADC를 사용하는 핀번호 설정



ATMEGA128에는 총 8개의 ADC채널이 존재하며 사용시 활성화하기위해 aaaa 레지스터의 비트4~0을 사용한다.

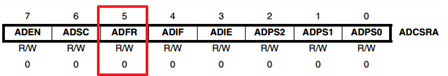


ADC 변환준비/완료

ADCSRA레지스터의 비트 7을 1로 세팅하면 읽어들일 준비가 되었다는 뜻이다. 변환이 완료되어 데이터 레지스터에 값이 업데이트 되면 ADCSRA레지스터의 비트4가 1로 설정된다. 즉 변환완료 확인을 이 4 비트를 사용하여 체크할 수 있다.

Single Conversion 모드에서는 모드를 설정이 필요 없으며 위에 설명된 과정을 통해 변환이 이뤄진다.

Free Running 모드에서는 모드 설정을 위해 ADFR비트를 1로 설정해야 하며 자동으로 데이터 레지스터로 값이 업데이트된다.

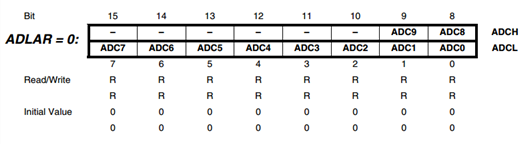


ADC pin에 연결된 값을 읽는 방법

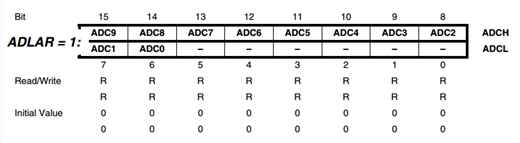
ADC는 10비트 결과 값을 가지기 때문에 두개의 데이터 레지스터가 필요하다.

이때 10개의 비트를 왼쪽부터 정렬할 것인지 오른쪽에서 정렬할 것인지에 따라 ADMUX 레지스터의 비트 에서 설정되어야 한다.

0일경우 ADCH, ADCL레지스터에 각각 2비트 8비트씩 할당되어 값을 저장한다.

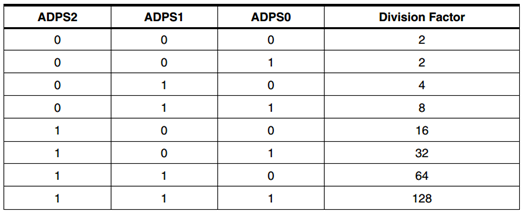


1일경우 ADCH, ADCL레지스터에 각각 8비트 2비트씩 할당되어 값을 저장한다.



ADC에서의 클록 주파수 설정

일반적인ADC는 50KHz ~ 200KHz의 범위에서 동작한다. Cpu 클록 주파수가 16MHz이므로 값을 낮춰주기위해 prescaler를 사용한다. 이를 위해 ADCSRA 레지스터의 2~0 비트를 사용한다.



선택된 핀의 입력 전압과 전압기준에 따라 변환결과는 다음과 같다.



**실습내용 및 결과**

실습1

|  |
| --- |
|  |
| /\*  \* Week12-1.c  \*  \* Created: 2020-11-27 오전 11:15:55  \* Author : lobgd  \*/  #define F\_CPU 16000000  #include <inttypes.h>  #include <avr/io.h>  #include <util/delay.h>  #include <avr/interrupt.h>  #include <avr/sleep.h>  #include <stdio.h>  void cmd(int);  void data(char);  void display1(char\*);  void display2(char\*);  char str[2][16] = {{"ADC VALUE :"},{" "}};  void ADC\_SingleConversion\_Init(unsigned char port) {  ADMUX = port; //ADMUX값을 설정  }    int ADC\_SingleConversion\_read() {  int output;  ADCSRA |= (1<<ADSC) | (1<<ADEN) ; //Start single conversion  while(ADCSRA & (1<<ADSC)); //wait Conversion to complete  output = ADCL + (ADCH << 8);  ADCSRA &= ~(1<<ADEN); //disable ADC  return output;  }  void cmd(int command){  PORTB = command;  PORTA= 0x04;  \_delay\_ms(1);  PORTA= 0x00;  }  void data(char str){  PORTB= str;  PORTA= 0x05;  \_delay\_ms(1);  PORTA= 0x01;  }  void display1(char \*str){  cmd(0x80);  int i =0;  while(str[i]!='\0'){  data(str[i++]);  }  }  void display2(char \*str){  cmd(0xc0);  int i =0;  while(str[i]!='\0'){  data(str[i++]);  }  }  int main()  {  int ADC\_Output;  float volt;  DDRA = 0xff; // set port A as output  DDRB = 0xff; // set port B as output  DDRF = 0xfe; // set port F as intput  EICRA = 0xff;  EIMSK = 0x0f;  SREG = 0x80;  cmd(0x38);  cmd(0x01);  cmd(0x80);  cmd(0x0c);  cmd(0x06);  //위 내용은 LCD강의 참고  while (1){  display1(str[0]); // 1행 출력  display2(str[1]); // 2행 출력  ADC\_SingleConversion\_Init(0x00); // ADC0 설정  ADC\_Output = ADC\_SingleConversion\_read(); // 변환된 digtal 값 읽기  volt = 5.\*(ADC\_Output/1023.); // 가변저항으로 얻은 전압값 계산  sprintf(str[1],"%d.%d%d%dV",((int)volt),((int)(volt\*10))%10,  ((int)(volt\*100))%10,((int)(volt\*1000))%10); // 전압 값 출력  \_delay\_ms(10);  }  return 0;  } |
| 오프라인 수업에서 조교님께 결과를 확인 받음. |

실습2

|  |
| --- |
|  |
| /\*  \* Week12-2.c  \*  \* Created: 2020-11-27 오전 12:42:12  \* Author : lobgd  \*/  #define F\_CPU 16000000  #include <avr/io.h>  #include <avr/interrupt.h>  #include <util/delay.h>  #include <stdio.h>  volatile unsigned char rx\_data ;  volatile unsigned char tx\_data ;  char value[3][20] ={" "," "," " };  char xx[8]={'x','-','a','x','i','s',':',' '};  char yy[8]={'y','-','a','x','i','s',':',' '};  char zz[8]={'z','-','a','x','i','s',':',' '};  float result =0.12412;  char num[30];  void xprint(){  for(int i=0;i<8;i++){  UDR0 = xx[i];  \_delay\_ms(10);  }  }  void yprint(){  for(int i=0;i<8;i++){  UDR0 = yy[i];  \_delay\_ms(10);  }  }  void zprint(){  for(int i=0;i<8;i++){  UDR0 = zz[i];  \_delay\_ms(10);  }  }  void vxprint(){  for(int i=0;i<6;i++){  UDR0 = value[0][i];  \_delay\_ms(10);  }  }  void vyprint(){  for(int i=0;i<6;i++){  UDR0 = value[1][i];  \_delay\_ms(10);  }  }  void vzprint(){  for(int i=0;i<6;i++){  UDR0 = value[2][i];  \_delay\_ms(10);  }  }  void calculate(int voltx,int volty,int voltz){  sprintf(value[0],"%dV",(int)voltx);    sprintf(value[1],"%dV",(int)volty);    sprintf(value[2],"%dV",(int)voltz);  }  void print(){  xprint();  vxprint();  yprint();  vyprint();  zprint();  vzprint();  }  void ADC\_SingleConversion\_Init(unsigned char port) {  ADMUX = port; //ADMUX값을 설정  }  int ADC\_SingleConversion\_read() {  int output;  ADCSRA |= (1<<ADSC) | (1<<ADEN) ; //Start single conversion  while(ADCSRA & (1<<ADSC)); //wait Conversion to complete  output = ADCL + (ADCH << 8);  ADCSRA &= ~(1<<ADEN); //disable ADC  return output;  }  int main(void)  { int ADC\_Outputx;  int ADC\_Outputy;  int ADC\_Outputz;    UCSR0A = 0x00;  UCSR0B = 0b10011000; // RXCIE0=1, TXEN0=1, RXEN0=1 ==> 수신완료 인터럽트 허가, 수신허가.송신허가  UCSR0C = 0b10000110; // 비동기 통신, 패리티 없음, ==>고정  // 데이터 비트 : 8비트, 정지비트 : 1비트  UBRR0H = 0; //보오 레이트 레지스터  UBRR0L = 103; //보오 레이트 레지스터, fosc=14.7456MHz, BAUD=9600bps,  SREG = 0x80;  float volt1;  float volt2;  float volt3;    while(1){  ADC\_SingleConversion\_Init(0x00); // ADC0 설정  ADC\_Outputx = ADC\_SingleConversion\_read(); // 변환된 digtal 값 읽기  volt1 = ADC\_Outputx;  \_delay\_ms(10);  ADC\_SingleConversion\_Init(0x01); // ADC1 설정  ADC\_Outputy = ADC\_SingleConversion\_read(); // 변환된 digtal 값 읽기  volt2 = ADC\_Outputy;  \_delay\_ms(10);  ADC\_SingleConversion\_Init(0x02); // ADC2 설정  ADC\_Outputz = ADC\_SingleConversion\_read(); // 변환된 digtal 값 읽기  volt3 = ADC\_Outputz;  \_delay\_ms(10);  calculate(volt1,volt2,volt3);  print();  };    } |
| 오프라인 수업에서 조교님께 결과를 확인 받음. |

**고찰**

ADC 변환에서 기준전압을 설정하는 것은 입력 전압의 범위와 같다. 만약 0~5V 사이의 아날로그 신호를 1024개의 디지털 신호로 변환하기 위해 기준전압은 5V가 되어야 한다.

데이터를 읽을 때 단일핀에서의 ADC변환이 있고 두개의 핀에서의 신호차를 얻는 것이 존재하는데 이번 실험에서는 단일 핀에서의 ADC변환을 사용하였다.

실습코드중 가장중요한 READ부분을 설명하면

int ADC\_SingleConversion\_read() {

int output;

ADCSRA |= (1<<ADSC) | (1<<ADEN) ; //Start single conversion

while(ADCSRA & (1<<ADSC)); //wait Conversion to complete

output = ADCL + (ADCH << 8);

ADCSRA &= ~(1<<ADEN); //disable ADC

return output;

}

ADCSRA 레지스터에 ADEN에 1을 설정하여 ADC변환을 활성화 시키고 ADSC를 1로 설정하여 단일 변환 모드에서의 ADC변환 준비가 되었음을 알린다.

변환이 완료되기를 기다리기 위해 ADSC가 1로 설정되어있음을 확인하고 0이 된다면 다음 코드를 수행한다.

값을 읽기위해 두개의 데이터 레지스터를 사용하며 데이터가 우측부터 정렬되었으므로 ADCL전체값과 ADCH일부분 값을 읽어들인다.

다음으로 ADC변환을 비활성화 시킨다.

실습 1에서는 가변저항에 따른 값을 0~5V로 표현하기위해 읽어들인 값을 1024로 나누고 5를 곱해 범위를 변환해주었다.

그러나 실습 2에서는 가속도의 값을 다른 값의 범위로 변환해줄 필요없이 기준전압에 따른 1024개로 나눈 값을 출력해야하므로 부가적인 연산이 필요없었다.

이차이를 인지하지 못해 실습시 다른 값이 나와 제대로 확인 하지 못하였지만 조교님의 도움으로 해결할 수 있었다.