**전자회로실험 보고서**



학과 전자공학과

학번 1218165

이름 김재욱

제출일 2020-10-08

**실습 이론**

1. 인터럽트

인터럽트란 외부장치나 sw에 의해서 처리해야할 일을 마이크로프로세서에게 처리 할 수 있도록 알려주는 역할이다.

마이크로 프로세서는 어떤 곳에서 인터럽트가 발생했는지 알아야 하며 또 어떤 일을 처리해야 하는지 알아야 하기 때문에 인터럽트가 제공하는 인터럽트 벡터 주소와 인터럽트 서비스 루틴또는 인터럽트 핸들러를 이용해 처리한다.

2. 인터럽트 종류

하드웨어 인터럽트

외부 장치의 입/출력으로 인해 발생되며 하드웨어에 의한 인터럽트이다

소프트웨어 인터럽트

SystemCall 이나 예외처리(0으로 나누기)등으로 인해 발생하며 소프트웨어에 의한 인터럽트이다.

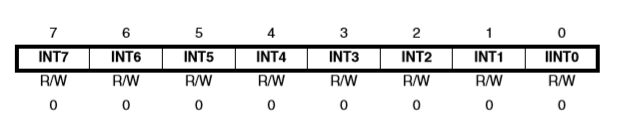
타이머 인터럽트

스케쥴링 알고리즘(RR)에 의해 cpu가 사용되는 시간에 일정 시간마다 인터럽트가 발생하여 다른 프로세스로 문맥교환 하도록 하는 인터럽트이다.

3. 인터럽트 레지스터

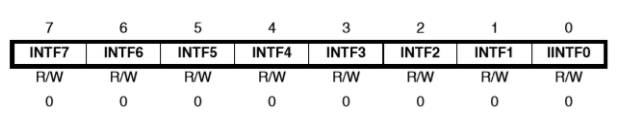
ATMEGA128에는 총 8개의 외부 인터럽트가 있다.

EIMSK 레지스터



사용하고자하는 외부 인터럽트 설정해주는 레지스터 이며 각 비트를 1로 설정해 사용함을 알린다.

EIFR 레지스터

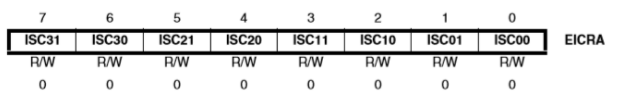


외부 인터럽트가 엣지 트리거에에 의해 요청된 경우 허용여부 상관 없이 1로 설정되며 이 값을 읽고 쓰는 것 또한 가능하다. 하지만 이번 실습에서는 ISR을이용하여 처리 할 일을 명시할 것이기 때문에 이 레지스터의 값을 읽어올 필요가 없다.

EICRA 레지스터

**INT0~INT3** 인터럽트의 트리거 방식을 설정해주는 레지스터이며

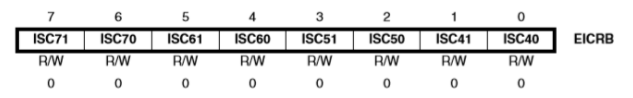
하나의 인터럽트당 EICRA레지스터의 2개의 비트를 할당받아 트리거 방식을 설정한다



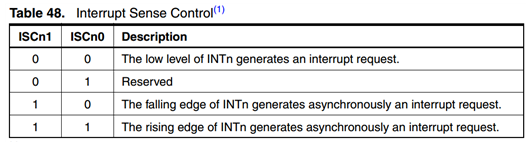
EICRB 레지스터

**INT4~INT7** 인터럽트의 트리거 방식을 설정해주는 레지스터이며

하나의 인터럽트당 EICRA레지스터의 2개의 비트를 할당받아 트리거 방식을 설정한다



트리거 방식은 아래의 표로 나타낸다.



실습1

**실습내용 및 결과**

|  |
| --- |
|  |
| /\*  \* week4-1.c  \*  \* Created: 2020-10-03 오전 12:00:21  \* Author : lobgd  \*/  #define F\_CPU 16000000  #include<avr/io.h>  #include<util/delay.h>  #include <avr/interrupt.h>  unsigned char led\_on;  ISR(INT3\_vect){  if (led\_on) {  PORTB=0b10000000; // LED OFF  led\_on = 0;  }  else {  PORTB=0b00000000; // LED ON  led\_on = 1;  }  }  int main(void){  DDRB = 0b10000111; // 출력 설정  DDRD = 0b00000000; // 입력 설정  PORTD = 0b00001000; // 풀업저항 설정  EICRA = 0b10000000; //인터럽트 트리거 방식 설정  EIMSK = 0b00001000; //인터럽트 허용 설정  SREG |= 0x80; //전체 인트럽트 허가  led\_on = 1;  while(1) {  PORTB=0b10000111; // LED OFF  \_delay\_ms(2000);  PORTB=0b00000000; // 모든 LED on  \_delay\_ms(500);  }  } |
| 실습 1의 결과 영상 첨부 |

실습2

|  |
| --- |
|  |
| #define F\_CPU 16000000  #define \_\_DELAY\_BACKWARD\_COMPATIBLE\_\_  #include <avr/interrupt.h>  #include<avr/io.h>  #include<util/delay.h>  unsigned char led\_on =1;  int i =0;  double interval =500.0;  unsigned char ledArray[8] = {0b10000000,  0b01000000,  0b00100000,  0b00010000,  0b00001000,  0b00000100,  0b00000010,  0b00000001};  void play(){  if(led\_on){  PORTB =ledArray[i];  \_delay\_ms(interval);  PORTB =0b00000000;  i++;  if(i>8){  i=0;  }  }  else{  PORTB =0b00000000;  }  }  ISR(INT5\_vect){  interval+=200;  }  ISR(INT6\_vect){  if (interval>0){  interval-=200;  }  }  ISR(INT7\_vect){  if (led\_on) { // LED OFF  led\_on = 0;  }  else { // LED ON  led\_on = 1;  }  }  int main(void){  // led  DDRB = 0b11111111; // 입출력 방향 설정  PORTB = 0b00000000;  // 버튼  DDRE = 0b00000000; // 입출력 방향 설정 인터럽트 사용  PORTE = 0b11100000; // 풀업저항 설정  EICRB = 0b10101000 ; //인터럽트 트리거 방식 설정  EIMSK = 0b11100000; //인터럽트 허용 설정  SREG |= 0x80; //전체 인트럽트 허가  while(1)  {  play();  }  } |
| 실습 2의 결과 영상 첨부 |

**고찰**

이번 실험은 인터럽트에 대한 실습이 이뤄졌다 OS를 배우고 있는데 인터럽트의 종류는 크게 3가지가 있으며 그중에서도 외부 장치(버튼 이나 스위치)에 의한 인터럽트를 실습하였다.

인터럽트에 의해 일을 처리하기 위해서는 인터럽트를 일으킨 곳의 주소를 알아야하며 어떤 일을 처리해야할지 알려주어야 한다. 각각 인터럽트 벡터와 인터럽트 서비스 루틴을 사용하는데 ATMEGA128에서는 인터럽트 벡터를 INT0\_VECT ~INT7\_VECT 로 표시하며 인터럽트 서비스 루틴은 ISR(){/\*동작 명시\*/}로 이뤄진다.

이때 ISR의 파라미터로 인터럽트 벡터 주소가 들어가야 하며 인터럽트 될때마다 ISR을 호출시켜 동작 할 수 있게 해야한다. 그러므로 인터럽트가 발생했는지 확인 할 필요가 없다

지난 주차에서 사용했던 ISR(USART0\_RX\_VECT)도 마찬가지로 읽을 데이터가 있는지 확인 하지 않더라도 ISR을 통해 자동 호출되어 데이터를 읽어 올수 있었다.

인터럽트의 가장 중요한 점은 main 코드가 실행되는 도중을 가로채어 서비스 할 수 있다는 점이다.