<**인터럽트**>

인터럽트를 이용한

LED,FND 불 켜기

1. **목적**

- 인터럽트를 연결한 실습보드제작 및 FND 연결

- 교제에 나온 프로그램 언어에 대해 알아보고 직접 코딩 하기.

- LED와 FND를 통해서 코딩 결과를 확인해보자.

1. **코딩 실습**
2. **인터럽트를 이용한 LED 점등(예제)**

**Sw0이 눌려지는 순간에 외부 인터럽트0가 발생하여 포트A에 연결되어 있는 LED가 순차적으로 점멸되도록 하는 프로그램 입니다.(외부 인터럽트0는 하강 에지에서 발생)**

* **인터럽트를 사용하기 전에는 인터럽트를 프로그램이 인식하기 위해서 #include <avr/interrupt.h> 라는 인터럽트 헤더파일을 사용하여야한다.**
* **사용되는 함수 EICRA: 외부인터럽트 컨트롤 레지스터/ EIMSK: 외부인터럽트마스크 레지스터/ ISR(vector){// 인터럽트 서비스 루틴 코드를 삽입}**
* **예상: 스위치를 누를 때마다 패턴이 이동할 것이다.**

**<소스코드>**

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

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

**unsigned char led=0xfe;**

**// 외부 인터럽트0 서비스 루틴**

**ISR(INT0\_vect)**

**{**

**led=led<<1; //1비트 쉬프트(한번에 한칸씩 이동)**

**led=led l 0b00000001; //최하위 비트 셋**

**if(led==oxff)led=0xfe; //모두 off 상태면 초기화**

**PORTA=led; //포트 출력**

**}**

**void main(void)**

**{**

**//포트 초기화**

**DDRA=0xff; // 포트A에 출력 설정**

**PORTA=led; // 포트 A에 초기값 출력**

**// 외부 인터럽트 초기화**

**EICRA=(2<<ISC00); //외부 인터럽트0 하강에지**

**EIMSK=(1<<INT0); // 외부 인터럽트0 인에이블**

**sei(); //전역 인터럽트 인에이블 비트셋**

**while (1);**

**}**

**결과: LED의 불이 스위치를 누를 때마다 외쪽으로 이동하는 것을 확인 할 수 있다.**

1. **지역 자동 변수를 사용한 인터럽트 서비스 루틴(불완전한 동작(7.1))**

**인터럽트를 이용하여 LED 불이 바뀌는 프로그램을 만들어보자.**

**인터럽트를 사용하기 전과 비교햇을 때 while루푸 내에서 LED패턴을 이동시키는 부분이 인터럽트 서비스 루틴으로 이동시켜서 프로그램이 작성된다.**

**예상: 스위치를 누를 때마다 패턴이 이동할것이다.**

**<소스 코드>**

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

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

**unsigned char pattern[8]={0xfe,0xfd,0xfb,0xf7,0xef,0xdf,0xbf,0x7f};**

**ISR(INT0\_vect) // 외부 인터럽트 0 서비스 루틴**

**{**

**int i=0; //패턴 인덱스(자동 변수 사용)**

**if(++i==8)i=0; //증가 값이 8이면 0으로 리셋**

**PORTA=pattern[i]; //i번째 패턴으로 LED를 켠다**

**}**

**int main(void)**

**{**

**DDRA=0xff; //포트A를 출력**

**PORTA=pattern[0]; //처음 패턴으로 LED를 켠다.**

**EICRA=(2<<ISC00); //외부 인터럽트0 하강트리거**

**EIMSK=(1<<INT0); //외부 인터럽트0 인에이블**

**sei(); // 인터럽트 허용**

**while (1); // 무한루프**

**}**

**결과: 스위치를 누르면 처음 이후에는 LED의 불이 이동하지 않는 것을 확인 할 수 있습니다.**

**이러한 현상이 일어나는 원인은 인터럽트 서비스루틴 내의 변수 i를 지역 자동 변수로 선언하였기 때문이다. 지역 자동 변수는 변수가 선언된 중괄호를 벗어나면 소멸되므로 인터럽트 서비스 루틴인 ISR 함수의 수행이 끝날 때 변수i는 소멸된다.변수i는 인터럽트가 발생하여 인터럽트 서비스루틴으로 들어올 때마다 새로 생성되고 0으로 초기화된다.**

**즉, 스위치를 누르면 항상 pattern[1]의 패턴으로 LED 를 점등한다. 이 문제를 해결하기 위해서 지역 자동 변수 대신에 지역 정적 변수를 사용한다 . int i를 static int i로 바꾸어서 프로그램에 넣는다**

1. **지역 자동 변수를 사용한 인터럽트 서비스 루틴 (불완전한 채터링 방지(7.2))**

**세번째 프로그램에서는 자동 변수 대신에 정적 변수를 사용하고 채터링을 방지하기 위해서 인터럽트 신호에 디바운싱을 추가하여서 프로그램을 작성한다.**

**예상: 두번째와는 다르게 이번에는 LED패턴의 변화가 눈에 볼일 것이다.**

**<소스 코드>**

**#define F\_CPU 7372800UL ////atemga 128의 클럭수를 입력**

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

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

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

**unsigned char pattern[8]={0xfe,0xfd,0xfb,0xf7,0xef,0xdf,0xbf,0x7f};**

**ISR(INT0\_vect) //외부 인터럽트 0 서비스 루틴**

**{**

**static int i=0; // 패턴 인덱스(정적 변수사용)**

**if(++i==8)i=0;**

**PORTA=pattern[i];**

**\_delay\_ms(20); // 스위치 누름에 대한 디바운싱**

**while(~PIND&0x01); // 스위치 해체를 기다림**

**\_delay\_ms(20); // 스위치 뗌에 대한 디바운싱**

**}**

**int main(void)**

**{**

**DDRA=0xff;**

**PORTA=pattern[0];**

**EICRA=(2<<ISC00); //외부 인터럽트0 하강트리거**

**EIMSK=(1<<INT0); //외부 인터럽트0 인에이블**

**sei(); //전역 인터럽트 허용**

**while (1); //무한루프**

**}**

**결과: 스위치를 누를 때마다 LED의 불이 이동하는 것을 확인할 수 있다.**

**그러나 이 프로그램에도 문제점이 있는데 채터링을 방지하기 위해서 딜레이 함수를 사용하였으나 채터링이 방지가 되지 않는 다는 점이다.**

**이러한 현상이 발생하는 원인은 현재 수행 중인 인터럽트 서비스 루틴이 종료된 후 인터럽트는 다시 허용된다. 이때 채터링에 의해 인터럽트 플래그이 세트되어 있으므로 인터럽트가 발생하게 된다.**

**즉 스위치를 누르지 않았음에도 불구하고 인터럽트 서비스 루틴으로 재 진입하게 되어 채터링이 방지되지 않는다. 인터럽트에서 채터링을 방지하기 위해서는 인터럽트 플래그 INTF0을 리셋하고 인터럽트 서비스 루틴을 종료하여야 한다.**

**외부 인터럽트의 플래그는 EIFR레지스터의 0번째 비트에 1을 쓰면 플래그를 리셋할 수 있다.(EIFR=(1<<INTF0)**

1. **외부인터럽트 INT0을 사용하여 LED 패턴 이동(7.5.1)**

**네번째에서는 세번째에서 방지하지못한 채터링을 방지하기 위하여 외부 인터럽트의 플래그를 세트시키는 EIFR을 사용하여 프로그램을 작동시킨다. (EIFR=(1<<INTF0)**

**예상: LED패턴의 변화가 확실하게 이루어지는 것을 확인 할 수 있을 것이다.**

**<소스 코드>**

**#define F\_CPU 7372800UL ////atemga 128의 클럭수를 입력**

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

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

**#include <avr/interrupt.h> //외부인터럽트 헤드파일**

**void msec\_delay(int n); //시간 지연함수**

**static unsigned char pattern[8]={0xfe,0xfd,0xfb,0xf7,0xef,0xdf,0xbf,0x7f};**

**ISR(INT0\_vect) //외부 인터럽트 0 서비스 루틴**

**{**

**static int i=0; //패턴 인덱스**

**if(++i==8)i=0;**

**PORTA=pattern[i];**

**\_delay\_ms(20); //디바운싱**

**while(~PIND&0x01); //스위치 해체를 기다림**

**\_delay\_ms(20);**

**EIFR=(1<<INTF0); // 인터럽트 플래그 INTF0을 리셋**

**}**

**int main(void)**

**{**

**DDRA=0xff;**

**PORTA=pattern[0];**

**EICRA=(2<<ISC00); //외부 인터럽트0 하강트리거**

**EIMSK=(1<<INT0); //외부 인터럽트0 인에이블**

**sei();**

**while (1);**

**}**

**void msec\_delay(int n)**

**{**

**for(;n>0; n--)**

**\_delay\_ms(1);**

**}**

**결과: 스위치를 누를 때마다 LED 패턴이 변하는 것을 확인 할 수 있다.**

1. **외부인터럽트 INT0을 사용하여 LED 패턴 이동(7.5.2)**

**이번에는 인터럽트 서비스 루틴에서 포트A에 패턴을 출력하는 부분을 main()함수로 넘겨서 프로그램을 작성한다. 인터럽트 서비스 루틴의 인덱스를 main()함수에 전달하는 방법은 첫번째로 인덱스 변수를 외부 변수로 선언하여 범위를 파일 전체로 확장한다. 두번째 인덱스 변수의 범위가 확장됨에 따라 변수명을 i와 같이 범용으로 사용하는 이름을 쓰지 않는 것이 좋다.**

**밑에 소스코드는 4번째 코드에서 main()함수에 포트A를 전달 한것이다.**

**<소스 코드>**

**#define F\_CPU 7372800UL ////atemga 128의 클럭수를 입력**

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

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

**#include <avr/interrupt.h> // 외부인터럽트 헤드파일**

**void msec\_delay(int n)**

**static int index=0; //인덱스 변수명을 i대신 index사용**

**static unsigned char pattern[8]={0xfe,0xfd,0xfb,0xf7,0xef,0xdf,0xbf,0x7f};**

**ISR(INT0\_vect)**

**{**

**if(++index==8)index=0;**

**\_delay\_ms(20);**

**while(~PIND&0x01);**

**\_delay\_ms(20);**

**EIFR=(1<<INTF0); //외부인터럽트 플래그을 리셋**

**}**

**int main(void)**

**{**

**DDRA=0xff; //포트 A를 출력 포트로 설정**

**PORTA=pattern[0]; // 처음 패턴으로 LED를 켠다.**

**EICRA=(2<<ISC00); //외부 인터럽트0 하강트리거**

**EIMSK=(1<<INT0); //외부 인터럽트0 인에이블**

**sei(); //인터럽트 허용**

**while (1)**

**{**

**PORTA=pattern[index]; //인덱스에 있던 출력 함수를 main()함수**

**로 옮김**

**}**

**}**

**void msec\_delay(int n)**

**{**

**for(;n>0; n--)**

**\_delay\_ms(1);**

**}**

**결과: 스위치를 누를 때마다 LED의 불이 이동하는 것을 확인 할 수 있다.**

1. **과제- 외부인터럽트를 이용하여 세그먼트 작동 시키기**

**작동:sw0를 누를 때마다 세그먼트의 글자가 0>1>2>3>…>9>A>B>C>D>E>F로 순환하고 sw1을 누르면 반대방향으로 순환하는 프로그램을 완성하라**

**그러기 위해서 외부 인터럽트 서비스 루틴을 두개를 넣고 0번스위치의 경우는 INT0으로 1번 스위치의 경우는 INT1로 해서 프로그램을 작동한다.**

**예상: 스위치0을 누를 때마다 FND의 값이 0에서 F로 변할 것이면, 스위치1을 누를 때는 반대로 F에서부터 감소해서 0으로 변하는 것을 볼 수 있을 것이다.**

**#define F\_CPU 73728200UL //atemga 128의 클럭수를 입력**

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

**#include <avr/interrupt.h> //인터럽트 헤드파일**

**#include <util/delay.h> //딜레이 헤드파일**

**Unsigned char pattern\_num1[16]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x27,0x7f,0x6f,0x77,0x7c,0x39,0x5e,0x79,0x71}; //0~F출력**

**int i=0;**

**ISR(INT0\_vect)//0번스위치 누를때 마다 0->...->F**

**{**

**if(++i==16)i=0; //상승**

**\_delay\_ms(20);**

**while(~PIND&0x01);**

**\_delay\_ms(20);**

**EIFR=0x01;**

**}**

**ISR(INT1\_vect) //1번스위치 반대로**

**{**

**if(i--==0)i=16; //하강**

**\_delay\_ms(20);**

**while(~PIND&0x01);**

**\_delay\_ms(20);**

**EIFR=0x01;**

**}**

**int main(void)**

**{**

**DDRC=0xff; //C포트 모두 출력사용**

**PORTC=pattern\_num1[0]; //0으로 초기화**

**EICRA=0xaa; //0~3번스위치 하강엣지사용**

**EIMSK=0x0f; //0~3번스위치 인터럽트사용**

**sei(); //인터럽트 허용**

**while (1)**

**{**

**PORTD=~0x20; //트렌지스터**

**PORTC=~pattern\_num1[i];**

**}**

**}**

**결과: 스위치0을 누를 때마다 FND의 값이 0에서 F로 변할 것이면, 스위치1을 누를 때는 반대로 F에서부터 감소해서 0으로 변하는 것을 볼 수 있을 것이다.**