<**타이머/카운터**>

8비트 타이머/카운터

16비트 타이머/카운터

1. **목적**

- 실습보드제작 및 LED 연결

- 교제에 나온 타이머/카운터 프로그램 언어에 대해 알아보고 직접 코딩 하기.

- LED를 통해서 타이머/카운터를 포함한 프로그램 코딩 결과를 확인해보자.

1. **코딩 실습**
2. **표준모드에서 오버플로 인터럽트를 사용한 주기적 LED 패턴 이동(8.1)(8비트)**

* **타이머에 적절한 초기값을 쓰면 타이머 시작부터 타이머 오버플로가 발생할 때까지의 시간을 조절할 수 있다.**
* **10msec마다 타이머/카운터 0 오버플로 인터럽트를 걸고 인터럽트 횟수가 20일 대 LED패턴을 이동시키는 프로그램을 작성**
* **인터럽트가 걸린 후 인터럽트 서비스 루틴에 진입하는 데 시간이 걸리므로 오버플로 발생 후 약간 시간이 지난 후 타이머 재설정이 이루어진다. 즉, 타이머 오버플로 사이의 사간 간격으 10msec보다 약간 길다.**
* **분주비 1024를 선택하였을 때 10mesc 후에 인터럽트를 걸려면 타이머0에 설정하여야 할 값은 icount=10\*16\*1000/1024=156/ TCNT0=256-156=100**
* **TCNTn: 타이머/카운터 레지스터/ TIMSK; 타이머/카운터 인터럽트 마스크 레지스터 TOIENn: 타이머 오버플로 인터럽트 허용**

**<소스코드>**

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

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

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

**ISR(TIMER0\_OVF\_vect) // 타이머 0 오버플로 인터럽트 서비스 루틴**

**{**

**static int index=0; //패턴 인덱스**

**static char n\_enter; //인터럽트 횟수**

**TCNT0=100; //10msec 후에 인터럽트 발생**

**n\_enter++; //인터럽트 횟수 증가**

**if(n\_enter==20)**

**{**

**n\_enter=0;**

**PORTA=pattern[index++]’; //패턴이동**

**If(index==8) index=8;‘**

**}**

**}**

**int main()**

**{**

**DDRA=0xFF; //A포트를 출력으로 설정**

**PORTA=oxFF; //LED를 끈다**

**//타이머 카운터 0 설정**

**TCCR0=0x00; //표준 모드, 타이머 정지**

**TCNT0=100; // 타이머 초기값 설정**

**//인터럽트 설정**

**TIMSK=(1<<TOIE0); //타이머 0 오버플로 인터럽트 허용**

**sei() ; // 전역 인터럽트 허용**

**TCCR0 |=(7<<CS00); // 분주비 1024로 타이머 시작**

**while(1);**

**}**

**결과: LED의 불이 왼쪽으로 이동하면서 불이 들어오는 것을 확인 할수 있다.**

1. **표준 모드에서 출력 비교일치 인터럽트를 사용한 주기적LED패턴이동(8.2)8비트**

* **비교일치 인터럽트는 비교 레지스터OCR0과 타이머 레지스터 TCNT0이 일치할때 발생한다. 10msec마다 비교일치 인터럽트가 일어나도록 하는 프로그램을 작성한다.**
* **타이머를 동작시키면 8비트 타이머는 0->255까지 세고 다음에는 오버플로가 발생하면서 0부터 값을 센다.**
* **분주비가 1024일 때 10msec에 해당하는 카운터는 156이므로 인터럽트가 일어나는 TCNT0의 값은 156(0x9c),56(0x38),212(0xd4)…이러한 순서로 이루어지므로 OCR0를 값은 순으로 설정하면은 10msec마다 비교일치 인터럽트가 발생한다.**
* **OCRn: 출력비교 레지스터/ OCIEn: 출력 비교일치 인터럽트 허용**

**<소스코드>**

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

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

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

**ISR(TIMER0\_OVF\_vect) // 타이머 0 오버플로 인터럽트 서비스 루틴**

**{**

**static int index=0; //패턴 인덱스**

**static char n\_enter; //인터럽트 횟수**

**OCR+=156; //10msec 후에 인터럽트 발생**

**n\_enter++; //인터럽트 횟수 증가**

**if(n\_enter==20) //200msec**

**{**

**n\_enter=0;**

**PORTA=pattern[index++]’; //패턴이동**

**If(index==8) index=8;‘**

**}**

**}**

**int main()**

**{**

**DDRA=0xFF; //A포트를 출력으로 설정**

**PORTA=oxFF; //LED를 끈다**

**//타이머 카운터 0 설정**

**TCCR0=0x00; //표준 모드, 타이머 정지**

**OCR0=156; // 10msec 후에 비교일치 발생**

**//인터럽트 설정**

**TIMSK=(1<<OCIE0); //타이머 0 비교일치 인터럽트 허용**

**sei() ; // 전역 인터럽트 허용**

**TCCR0 |=(7<<CS00); // 분주비 1024로 타이머 시작**

**while(1);**

**}**

**결과: LED의 패턴이 이동속도가 위의 실험과 비슷한 것을 확인할 수 있다.**

1. **CTC모드를 사용한 주기적 LED패턴 이동 (8.3)8비트**

* **CTC모드를 설정하면 카운터값 TCNTn이 비교값 OCRn까지 증가하였다가 리셋되고 비교일치 인터럽트를 발생한다. 즉, 한 번 비교값을 설정하면 계속적으로 인터럽트가 발생한다.**
* **CTC모드에서는 인터럽트 서비스 루틴에서 OCR0를 다시 설정할 필요가 없다.**
* **10msec 주기로 비교일치 인터럽트를 발생시키고 인터럽트 횟수를 세어 20번 마다 LED패턴을 이동하도록 프로그램을 작성한다**
* **CTC모드에서 비교일치가 일어나는 주기는 OCRn+1이므로 OCR0=155**
* **CTC모드의 설정; WGM01=1/출력핀OC0를 반전모드로 설정: COM01:0=0x01/타이머 정지: cs02:0=0x00** => **TCCR0=(1<<WGM01)|(1<<COM00);**

**<소스코드>**

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

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

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

**ISR(TIMER0\_OVF\_vect) // 타이머 0 오버플로 인터럽트 서비스 루틴**

**{**

**static int index=0; //패턴 인덱스**

**static char n\_enter; //인터럽트 횟수**

**n\_enter++; //인터럽트 횟수 증가**

**}**

**if(n\_enter==20)**

**{**

**n\_enter=0;**

**PORTA=pattern[index++]’; //패턴이동**

**If(index==8) index=8;‘**

**}**

**}**

**int main()**

**{**

**DDRA=0xFF; //A포트를 출력으로 설정**

**DDRB |= (1<<DDB4); // PB4(OC0)를 출력으로 설정**

**PORTA=oxFF; //LED를 끈다**

**//타이머 카운터 0 설정:ctc모드, 타이머 정지, oc0핀 사용**

**TCCR0=(1<<WGM01)|(1<<COM00);**

**OCR0=156; // 10msec마다 비교일치 발생**

**//인터럽트 설정**

**TIMSK=(1<<OCIE0); //타이머 0 비교일치 인터럽트 허용**

**sei() ; // 전역 인터럽트 허용**

**TCCR0 |=(7<<CS00); // 분주비 1024로 타이머 시작**

**while(1);**

**}**

**결과: LED의 이동속도는 두번째 때와 비슷하며 그 외에 PB4와 연결된 LED는 계속해서 깜빡거리는 것을 반복한다.**

1. **도트매트릭스 켜기(과제1)**

* **5msec마다 타이머 인터럽트 서비스루틴에서 행을 바꾸며 LED 패턴을 변경하는 프로그램을 작성한다.**

**<소스 코드>**

**#define F\_CPU 1600000UL**

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

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

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

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

**ISR(TIMER0\_COMP\_vect)**

**{**

**static int index=0; //패턴 인덱스**

**static char n\_enter=0; //인터럽트 횟수**

**n\_enter++; //인터럽트 횟수 증가**

**if(n\_enter==20) //200msec**

**{**

**n\_enter=0;**

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

**PORTC=0x01<<index; //led, fnd패턴 이동**

**PORTA=pattern[index];**

**}**

**}**

**int main()**

**{**

**DDRA=0xff; //A포트를 출력으로 설정**

**DDRC=0xff; //C포트를 출력으로 설정**

**TCCR0=(1<<WGM01)|(1<<COM00); //타미어/카운터 0설정 : CTC모드,**

**타이머 정지, OC0핀 사용**

**OCR0 = 77; //5msec마다 비교일치 발생**

**TIMSK=(1<<OCIE0); //타이머 0비교일치 인터럽트 허용**

**sei(); //전역 인터럽트 허용**

**TCCR0|=(7<<CS00); //분주비 1024로 타이머 시작**

**while(1);**

**}**

**결과: 5msec 간격으로 LED의 패턴이 왼쪽으로 FND의 값이 순차적으로 변하는 것을 확인 할수 있다. 이 숫자의 값은 패턴에 들어있는 LED불에 들어오는 위치의 값과 일치한다.**

1. **PWN 신호로 LED 밝기 조절(8.4.c,8비트)**

* **5msec마다 타이머 인터럽트 서비스루틴에서 행을 바꾸며 LED 패턴을 변경하는 프로그램을 작성한다.**
* **분주비 8을 사용하여 FAST PWN모드로 OC0에 연결된 LED의 밝기를 조절하는 프로그램이다. PWM 신호의 듀티비는 외부인터럽트 INT0와 INT1을 사용하여 증가/감소 시킨다.**
* **FAST PWN 모드의 설정: WGM01=1, WGM00=1**
* **출력핀 OC0의 모드설정: COM01:0=0x02**
* **타이머정지: CS02:0=0x00**

**<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 지연 함수**

**static unsigned char duty=125; // 초기 듀티비**

**ISR(INT0\_vect) // 외부 인터럽트0 서비스루틴**

**{**

**duty+=5; //듀티비5증가**

**if(duty>=250) duty=250;**

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

**while(~PIND&0x01); // 스위치 해제를 기다림**

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

**EIFR=(1<<INTF0); // INT0 외부인터럽트 플래그을 리셋**

**}**

**ISR(INT1\_vect) // 외부 인터럽트1 서비스루틴**

**{**

**duty-=5; //듀티비5증가**

**if(duty>=250) duty=250;**

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

**while(~PIND&0x02); // 스위치 해제를 기다림**

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

**EIFR=(1<<INTF1); // INT1 외부인터럽트 플래그을 리셋**

**}**

**int main()**

**{**

**DDRB |=(1<<DDB4); //PB4(OC0)를 출력으로 설정**

**//타이머/ 카운터 0설절, FAST PWM 모드, 타이머 정지**

**TCCR0=(1<<WGM01)|(1<<WGM00)|(2<<COM00);**

**//인터럽트 설정, ISC11:0=2, ISC01:0=2(하강 모서리)**

**EICRA=(2<<ISC10)|(2<<ISC00) //외부 인터럽트0,1 하강트리거**

**EIMSK=(1<<INT0)| (1<<INT1); //외부 인터럽트0,1 인에이블**

**sei();**

**TCCR0 |= (0x02<<CS00); //분주비 8로 타이머 시작**

**while (1)**

**{**

**OCR0=duty; //PWM신호의 듀티비를 바꾼다.**

**}**

**}**

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

**{**

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

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

**}**

**결과: 스위치0을 누를때는 불빛이 점점 어두워지고 스위치1을 누를때는 불빛이 밝아 지는 것을 확인할 수있다**

1. **Phase Correct PWM 모드로 LED 밝기 조절(과제2)**

* **Phase correct pwm모드의 신호 반송주기는 Fast pwm모드의 약 두배이다**

**<소스코드>**

**#define F\_CPU 1600000UL**

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

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

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

**#define DEBOUNCING\_DELAY 20 //msec**

**void msec\_delay(int n); //시간지연함수**

**static volatile unsigned char duty=125; //초기 듀티비**

**ISR(INT0\_vect) //외부 인터럽트 0 서비스루틴**

**{**

**duty+=5; //듀티비 5증가**

**if(duty>=250)duty=250;**

**msec\_delay(DEBOUNCING\_DELAY);**

**while(~PIND&0x01); //스위치 해제를 기다림**

**msec\_delay(DEBOUNCING\_DELAY);**

**EIFR=(1<<INTF0); //INT0 인터럽트 플래그 리셋**

**}**

**ISR(INT1\_vect) //외부 인터럽트 1 서비스루틴**

**{**

**duty-=5; //듀티비 5감수**

**if(duty<=5)duty=5;**

**msec\_delay(DEBOUNCING\_DELAY);**

**while(~PIND&0x02); //스위치 해제를 기다림**

**msec\_delay(DEBOUNCING\_DELAY);**

**EIFR=(1<<INTF1); //INT1 인터럽트 플래그 리셋**

**}**

**int main()**

**{**

**DDRB|=(1<<DDB4); //PB4(OC0)를 출력으로 설정**

**//타이머/카운터0 설정, Phase Correct PWM모드, 타미어 정지**

**TCCR0=(0<<WGM01)|(1<<WGM00)|(2<<COM00);**

**EICRA=(2<<ISC10)|(2<<ISC00); //인터럽트 설정, 하강모서리**

**EIMSK=(1<<INT0)|(1<<INT1); //외부 인터럽트 0,1 허용**

**sei(); //전역 인터럽트 허용**

**TCCR0|=(0x02<<CS00); //분주비 8로 타이머 시작**

**while(1)**

**{**

**OCR0=duty; //PWM신호의 듀티비를 바꾼다.**

**}**

**}**

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

**{**

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

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

**}**

**결과: 스위치0을 누를때는 불빛이 점점 어두워지고 스위치1을 누를때는 불빛이 밝아 지는 것을 볼 수 있는데 이 속도가 Fast 모드에 비해서 느린 것을 확인 하므로써 phace correct 모드가 fast모드보다 반송주기가 크다는 것을 알 수 있다.**

1. **타이머/카운트1을 이용한 LED 제어 실험(일반 모드)**

* **타이머/카운터1을 일반 모드에서 1초 간격으로 오버플로우 인터럽트를 발생시켜 LED를 1비트씩 쉬프트 하면서 ON시키는 프로그램을 작성하시오**
* **TCNT: 타이머/카운터 1,3 레지스터/ TCCRnA:타이머/카운터n제어 레지스터**

**SREG: 상태레지스터(전체 인터럽트 인에이블 비트)**

**<소스 코드>**

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

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

**unsigned char led = 0xFE;//LED 출력 초기값**

**int main()**

**{**

**// I/O 포트 초기화**

**DDRA = 0xFF; // LED 연결 포트A 출력으로 설정**

**PORTA = led; // 포트A에 초기값 출력**

**// 타이머/카운터1 초기화**

**TIMSK = 0x04; // TOIE1 = 1(오버플로우 인터럽트 인에이블)**

**TCCR1A = 0x0;**

**TCCR1B = 0x05; // 일반모드, 1024분주**

**TCCR1C = 0x0;**

**TCNT1 = 58336; // 1/16us \* 1024 \* (65536-49911) = 1s**

**//sei();**

**SREG = 0x80; // 전역 인터럽트 인에이블 비트 I셋**

**//SREG = 0x00; // 전역 인터럽트 인에이블 비트 I셋**

**while(1); // 대기**

**}**

**// 타이머/카운터1􀀁오버플로우 서비스 루틴**

**// 인터럽트 발생 주기 1/7372800(AVR크리스탈 진동수) \* 1024분주 \* (65536 - 58336) =1s**

**ISR(TIMER1\_OVF\_vect)**

**{**

**// LED 순차 점멸**

**TCNT1 = 58336; // 초기값 재설정**

**led = (led << 1) | 0x01; // 1비트 쉬프트, 최하위 비트 셋**

**if(led == 0xFF) led = 0xFE; // 모두 off이면 초기값 재설정**

**PORTA = led; // 포트 출력**

**}**

**결과: LED의 불이 1초 간격으로 왼쪽으로 이동하는 것을 볼 수 있다.**

1. **타이머/카운트1을 이용한 LED 제어 실험(CTC 모드)**

* **타이머/카운터1을 CTC 모드(모드 4)에서 1초 간격으로 출력 비교 매치 인터럽트를 발생시켜 포트 C에 연결되어 있는 LED를 1비트씩 쉬프트 하면서 ON시키는 프로그램을 작성하시오.**

**<소스 코드>**

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

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

**unsigned char led = 0xFE;//LED 출력 초기값**

**int main()**

**{**

**// I/O 포트 초기화**

**DDRA = 0xFF; // LED 연결 포트A 출력으로 설정**

**PORTA = led; // 포트A에 초기값 출력**

**// 타이머/카운터1 초기화**

**TIMSK = 0x10; // OCIE1A = 1(출력비교매치인터럽트 A)**

**TCCR1A = 0x0;**

**TCCR1B = 0x0D; // CTC모드, 1024분주**

**TCCR1C = 0x0;**

**TCNT1 = 0;**

**OCR1A = 7199;**

**// 1/7372800(AVR 크리스탈 주파수) \* 1024 \* (1+7199) = 1s**

**//sei();**

**SREG = 0x80; // 전역 인터럽트 인에이블 비트I셋**

**//SREG = 0x00; // 전역 인터럽트 인에이블비트 I셋**

**while(1); // 대기**

**}**

**//타이머/카운터1 출력 비교 매치 인터럽트 루틴**

**//인터럽트 발생 주기 1/7372800(AVR크리스탈 주파수)\*1024분주\*(1+7199)=1s**

**ISR(TIMER1\_COMPA\_vect)**

**{**

**// LED 순차 점멸**

**OCR1A = 7199; // 초기값 재설정**

**led = (led << 1) | 0x01; // 1비트 쉬프트, 최하위비트 셋**

**if(led == 0xFF) led = 0xFE; // 모두 off이면초기값 재설정**

**PORTA = led; // 포트 출력**

**}**

**결과: LED의 불이 1초 간격으로 왼쪽으로 이동하는 것을 볼 수 있다.**

1. **타이머/카운트3을 이용한 LED 입력 캡처 실습 실험(CTC 모드)**

* **입력 캡쳐 실험을 위해 SW5가 눌려질 때마다 갱신되는 타이머/카운터3의 캡쳐 레지스터 ICR3의 16진수 첫째 자리값을 7-Segment에 표시되도록 하시오 (타이머/카운터3의 캡쳐 입력 신호 단자는 ICP3(PE7), 스위치 입력 회로의 SW5에 연결, 타이머/카운터3는 일반 모드,1024 분주로 동작시킴)**
* **ICRn:입력 캡처 레지스터/ ETIMSK:확장 타이머/카운터 인터럽트마스크**

**<소스코드>**

**#define F\_CPU 7372800UL**

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

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

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

**//16진수표시**

**unsigned char seg\_pat[16]=**

**{0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f,0x77,0x7c,0x39,0x5e,0x79,0x71};**

**int i=0;**

**void hex\_disp(void);**

**int main(void)**

**{**

**DDRC=0xff; //포트c 7세그먼트 출력**

**DDRE=0x0f; //포트E 상위4비트 입력**

**DDRD=0xf0; //포트D 상위4비트 출력(트랜지스터)**

**ETIMSK=0b00100000; //TICIE3=1(입력 캡쳐 인터럽트)**

**TCCR3A=0x0; //일반모드**

**TCCR3B=0x05; //1024분주**

**//ICNC3=ICES3=0(no필터, 하강 캡쳐)**

**TCCR3C=0x0;**

**TCNT3H=0; //TCNT3초기값**

**TCNT3L=0;**

**SREG=0x80; //전역 인터럽트 인에이블 비트 I셋**

**while(1) hex\_disp(); //캡쳐 값 표시하면서 대기**

**}**

**ISR(TIMER3\_CAPT\_vect)**

**{**

**);//타이머/카운터3 입력 캡쳐 인터럽트 서비스 루틴**

**i=ICR3L+(ICR3H<<8);**

**}**

**void hex\_disp(void)**

**{**

**int h1;**

**h1=i&0x000f; //16의 0승 자리 값**

**PORTD=~0x20; //PD5트랜지스터**

**PORTC=~seg\_pat[h1];**

**\_delay\_ms(10);**

**}**

**결과:스위치를 누를 때마다 FND의 나오는 값들이 변하는 것을 볼 수 있는데 이는 16진수에서 0승자리에 해당하는 값들이 출력되고 있다.**