**전자회로실험 보고서**



학과 전자공학과

학번 1218165

이름 김재욱

제출일 2020-10-11

**실습 이론**

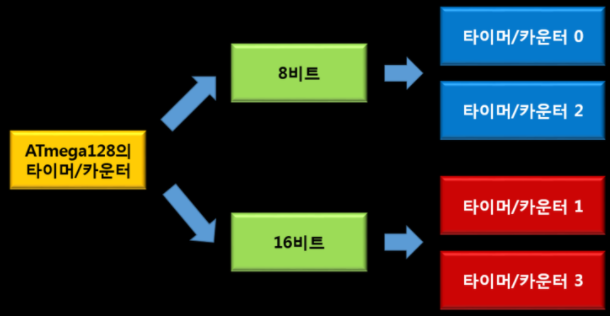
1. 타이머/카운터

타이머 모드에서의 동작

타이머가 내부 클럭을 이용하여 일정주기로 신호를 인가하고 카운터가 그 클럭의 계수를 합하여 카운터가 가질 수 있는 최대값을 넘기면 오버플로우가 발생하는데 이 오버플로우가 발생하면서 인터럽트가 발동되어 어떤 일을 처리 할수 있도록 하는 방식

내부 클럭을 사용하므로 일정한 주기의 신호가 인가되어 시간을 계산하기에 용이하다.

2. 타이머/카운터 종류



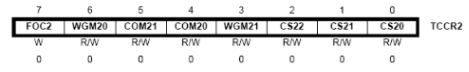
ATMEGA128에는 총 4가지의 타이머/인터럽트가 존재하며 각 타이머 인터럽트가 가질 수 있는 최대값에 따라 달라집니다. 최대값은 비트수에 결정된다

이번 실습에서 사용하는 타이머/카운터 2 를 사용하기 때문에 8비트이다

각 타이머/카운터의 기능이 다르기 때문에 레지스터 수와 종류도 다르다.

3. 타이머/카운터 2 레지스터

▶TCCR2



타이머/카운터 2의 제어 레지스터 이며 비트에따른 기능은 다음과 같다.

|  |  |
| --- | --- |
| 비트 | 기능 |
| 7 | PWM 모드가 아닌 경우에만 유효하며 1인경우 OC2를 통해 비트 5,4 번에 설정된값 출력 |
| 6,3 | 타이머 /카운터 2의동작 모드를 설정 (일반모드는 각각 0) |
| 5,4 | 타이머/카운터2의 출력단자 OC2의 동작제어 |
| 2,1,0 | 타이머/카운터2 분주비(프리스케일러) 결정 |

▶TCNT2

타이머/카운터 2의 카운터 값을 저장하는 레지스터 이며 0xFF(8비트 이므로)에서 0x00로 바뀔때 인터럽트가 발생된다.

▶OCR2

TCNT2 값과 비교하여 두 값이 일치했을 경우에만 OCR핀을 통해 설정된 값이 출력되거나 인터럽트가 발생한다.

▶TIFR

8비트의 타이머/카운터의 인터럽트 상태를 저장하는 레지스터

타이머/카운터 2가 가지는 비트는 7,6번이며 각각 출력 비교 인터럽트와 오버플로우 인터럽트에 대한 허가 가 설정된다.

ISR가 실행된다면 자동으로 0으로 설정된다.



▶TIMSK



타이머/카운터2의 인터럽트 모드를 설정한다.

|  |  |
| --- | --- |
| 비트 | 기능 |
| 7 | 타이머/카운터2 출력 비교매치 인터럽트 인에이블 비트 이며 SREG의 I 비트가 설정되어야 한다. |
| 6 | 타이머/카운터2 출력 오버플로우 인터럽트 인에이블 비트 이며 REG의 I 비트가 설정되어야 한다. |

**실습내용 및 결과**

실습1

|  |
| --- |
|  |
| /\*  \* week5-1.c  \*  \* Created: 2020-10-11 오전 10:14:00  \* Author : lobgd  \*/  #define F\_CPU 16000000  #include <avr/io.h>  #include <avr/interrupt.h>  // Normal Mode  unsigned char led = 0x00 ;  unsigned int count = 0;  ISR(TIMER2\_OVF\_vect){  count ++;  if (count == 60) {  if(led==0b10000000) {  led=0b00000001;  }  else {  led = led << 1;  }  count = 0;  }  }  int main(){  DDRB = 0xff ;  TCCR2 = 0b00000101 ; //프리스케일러 분주비 1024  TIMSK = 0b01000000 ; //오버플로우 인터럽트 허용  TCNT2 = 0 ; //타이머카운터0 초기화  SREG = 0x80 ; //전체 인터럽트 허가  led=0b00000001;  while(1){  PORTB=led;  }  } |
| 실습 1에대한 결과 영상 첨부 |

실습2

|  |
| --- |
|  |
| /\*  \* week5-2.c  \*  \* Created: 2020-10-11 오후 9:09:35  \* Author : lobgd  \*/  #define F\_CPU 16000000  #include <avr/io.h>  #include <util/delay.h>  #include <avr/interrupt.h>  volatile int count =0;  volatile int flag =0;  volatile int count\_time =0;  volatile char str[1][30] = {{"stop watch"}};  volatile char ch\_num[1][30] = {{"00:00:00"}};  volatile char reset[1][30] = {{"00:00:00"}};  ISR(TIMER2\_OVF\_vect){  if(flag){  count++;  if (count == 60) { //1초  count\_time+=1;  set();  count =0;  }  }  }  void set(){  hour\_change((count\_time/3600));  min\_change(count\_time/60);  sec\_change(count\_time%60);  }  ISR(INT5\_vect){ //시작  flag =1;  }  ISR(INT6\_vect){ //정지  flag=0;  }  ISR(INT7\_vect){ //  count\_time=0;  set();  }  void hour\_change(char num){  ch\_num[0][0] = num/10+'0';  ch\_num[0][1] = num%10+'0';  }  void min\_change(int num){  ch\_num[0][3] = num/10+'0';  ch\_num[0][4] = num%10+'0';  }  void sec\_change(int num){  ch\_num[0][6] = num/10+'0';  ch\_num[0][7] = num%10+'0';  }  void cmd(int command){  PORTB = command;  PORTD = 0x04; // 데이터 통신 enable, lcd 위치 설정 명령  \_delay\_ms(1);  PORTD = 0x00; // 데이터 통신 disable  }  void data(char str){  PORTB = str;  PORTD = 0x05; // 데이터 통신 enable, 레지스터에 저장해놨던 위치에 값을 쓰는 명령  \_delay\_ms(1);  PORTD = 0x01; // 데이터 통신 disable  }  void display1(char\*str){  cmd(0x80); // lcd 첫번째줄 위치 (0,0)  int i = 0;  while (str[i]!='\0')  {  data(str[i++]);  }  }  void display2(char\*str){  cmd(0x80 | 0x40); // lcd 두번째줄 위치 (1,0)  int i = 0;  while (str[i]!='\0')  {  data(str[i++]);  }  }  int main(){  DDRD=0xff;  DDRB=0xff;  DDRE=0x00;  PORTE=0b11100000;  cmd(0x38); // Set 8bit 2Line 5x7 dots  cmd(0x01); // Display clear  cmd(0x80); // Set DDRAM address or cursor position on display  cmd(0x0c); // Display on Cursor Off  cmd(0x06); // Entry Mode  // 인터럽트  EICRB = 0b10000000; //인터럽트 트리거 방식 설정  EIMSK = 0b11100000; //인터럽트 허용 설정  //타이머/카운터  TCCR2 = 0b00000101 ; //프리스케일러 분주비 1024  TIMSK = 0b01000000 ; //오버플로우 인터럽트 허용  TCNT2 = 0 ; //타이머카운터0 초기화  SREG = 0x80 ;    display1(str[0]);  while(1){  display2(ch\_num[0]);  }  } |
| 실습 2에대한 결과 영상 첨부 |

**고찰**

이번 실험은 입력되는 클럭의 개수를 세어 조건에 만족하면 인터럽트 를 발생시키는 타이머/카운터를 실습했다.

ATMEGA128은 총 4개의 타이머/카운터를 지원하며 각 레지스터에 어떤 값으로 세트 하느냐에 따라 기능들도 다르다 그중 내부 클럭 개수에 의한 시간을 측정하기위해 타이머/카운터 2의 일반 모드를 사용하였다.

이 또한 인터럽트의 한종류 이기때문에 ISR을 이용하여 처리 할 수 있다.

atmega 128의 메인 클럭이 16MHz이며 이번 실습에서 사용한 프리스케일러의 분주비가 1024이므로 1개의 클럭입력에 1/(16000000/1024) =0.000064 초가 걸린다. 8bit인타이머/카운터에서는 256번의 클럭입력이 들어와야 오버플로우로 인한 인터럽트가 걸리므로 한번의 인터럽트가 걸리면 0.000064 \* 256 =0.016384초가 걸린다. 1초를 세기 위해 1/0.016384 = 61번의 인터럽트 개수를 세야 한다. 이번 실습 에서는 반올림 하여 60으로 사용 된것 같다.

즉, 클럭의 분주비로 클럭 1개를 세는데 걸리는 시간을 조절 할수 있다. (일반모드이므로 내부 클럭으로 조절이 가능하다.)

타이머가 1을 세는데 걸린시간 = 1/(16000000/프리스케일러 분주비) = 프리스케일러 분주비/시스템 클럭

오버플러우가 발생하는 주기 = 타이머가 1을 세는데 걸린시간 \* 256(8비트)

T초에 필요한 인터럽트 발생횟수 = T/오버플로우가 발생하는 주기

이때 인터럽트 개수와 클럭의 개수를 헷갈리지 말아야 한다.