|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **결 과 보 고 서** | | | | |
| 학 과 | 학 년 | 학 번 | 조 | 성 명 |
| 전자공학과 | 3 | 12191505 | - | 윤수연 |
| 실험 제목 | Timer/Counter | | | |
| 1. **실험 과정**    1. 실습 1단계  |  | | --- | | #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 == 5) {    if(led==0b10000000) {  led=0b00000001;  }    else {  led = led << 1;  }    count = 0;  }  }  int main(){  DDRA = 0xff ;  TCCR2 = 0b00000101 ; //프리스케일러 분주비 1024  TIMSK = 0b01000000 ; //오버플로우 인터럽트 허용  TCNT2 = 0 ; //타이머카운터0 초기화  SREG = 0x80 ; //전체 인터럽트 허가    led=0b00000001;    while(1){  PORTA=led;  }  } |       그림 1. 실습 1단계 회로 구성도  실습 1단계는 내부 클럭을 이용하여 1초마다 켜지는 LED를 순서에 맞게 바꾸는 문제로, 분주비에 따른 Tin값과 시간이 흐를 때마다 커지는 TCNT2 레지스터의 오버 플로우 인터럽트를 통해 시간을 측정하는 방법에 대해 고민하는 실험 단계다.  위의 코드와 회로 구성도를 보면 실습 1단계에서는 주어진 코드를 거의 손보지 않았지만 한 부분에서 살짝 수정이 있었다. Timer/Counter가 내부 clock을 받기 때문에 일정한 주기를 갖는 펄스를 입력 받지만 실제 실험 결과를 볼 때도 1초에 하나의 LED 단위가 shift 되는 동작을 보기 위해서 코드의 count 변수를 수정했다.   * 1. 실습 2단계  |  | | --- | | #define F\_CPU 16000000  #include <avr/io.h>  #include <avr/interrupt.h>  #include <util/delay.h>  #include <stdio.h>  //A, B = LCD control, data  //D = interrupt  // Normal Mode  volatile unsigned int sec, min, hour=0;  volatile unsigned int count=0;  //initialize  volatile char str1[1][16] = {{"Stop Watch"}};  volatile char str2[1][16] ;  //fuction declaration about LCD  void cmd(int command);  void data(char str);  void display1(char\*str);  //fuctions about interrupt  ISR(TIMER2\_OVF\_vect) {//algorithm  count ++;  if(count ==3) {//about 1 sec  sec ++;  count = 0;  if (sec == 60) {  sec = 0;  min ++;  if (min == 60) {  min = 0;  hour ++;  }  }  }  }  // start  ISR(INT0\_vect) {  TIMSK = 0b01000000;  }  // stop  ISR(INT1\_vect) {  TIMSK = 0x00;  }  // reset  ISR(INT2\_vect) {  sec, min, hour = 0;  }  int main()  {  //initial setting for Timer/Counter  TCCR2 = 0b00000101; //prescaler 1024  TCNT2 = 0; //init\_0  //TIMSK = 0b01000000;  //initial setting for LCD  DDRA = 0xff;  DDRB = 0xff;    cmd(0b00111100); // set 8bit 2line 4\*10dot  cmd(0x01); // display clear  cmd(0x80); // set DDRAM address or cursor position on display  cmd(0x0c); // display on cursor off  cmd(0x06); // entry mode    //setting for button interrupt  DDRD = 0x00; // input  PORTD = 0b00000111; // pull up on  EICRA = 0b00101010;  EIMSK = 0b00000111;  SREG = 0x80; //allow global interrupt  display1(str1[0]);    while(1) {  sprintf(str2,"%02d:%02d:%02d",hour,min,sec);  display2(str2[0]);  }  }  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;  }  // LCD 1st line  void display1(char \*str) {  cmd(0x80);  int i = 0;  while(str[i] != '\0') {  data(str[i++]);  }  }  // LCD 2nd line  void display2(char \*str) {  cmd(0b11000000);  int i = 0;  while(str[i] != '\0') {  data(str[i++]);  }  } |   실습 2단계는 실습 1단계의 연장선으로 시간이 흐름에 따라 LCD패널을 이용하여 STOP WATCH를 작성하는 단계다. OVF INTERRUPT를 사용하여 시간을 흐름을 확인하는데, start, stop, reset으로 총 세 개의 버튼이 있다.  이번 실습 단계에서는 유독 함수가 많았다. 우선 평소와 같이 맨 처음 header files를 include하고, 필요한 전역 변수들을 정의 헸다. 1단계와 동일하게 count하기 위한 전역변수 count와 시, 분, 초를 각각 저장하기 위한 세 개의 변수들이다. 각 레지스터 값들은 이전에 사용했던 값들과 동일하기 때문에 다른 설명은 하지 않겠다.  LCD 관련 함수들은 코드 가독성을 위해 정의부와 구현부를 따로 떼어 인터럽트 함수들(ISR)이 가장 먼저 구현된다. 총 4개인데, 각 star, stop, reset 버튼에 대해 하나씩 구현하고 OVF interrupt에는 본격적인 알고리즘을 넣었다. 사실 if문이 중복으로 저렇게 들어가는 코드는 좋은 코드가 아니지만 알고리즘 자체가 간결해서 그냥 중복으로 대충 넣었다.  이번 주차는 유독 코드를 작성하며 주석을 많이 넣는 등 가독성에 신경을 썼는데, LCD에 표시되는 string변수를 update할 때 C 언어에서 배웠던 sprintf함수를 사용해서 손쉽게 string을 update할 수 있었다. 이러한 기능을 하는 함수가 있다는 것은 기억이 났지만 정확한 함수 명이 기억나지 않아서 아래 정보를 구글링하여 코드를 완성할 수 있었다.     1. **실험 결과**    1. 실습 1단계     그림 2. 실험 1 실행 결과1    그림 3. 실험 1 실행 결과2   * 1. 실습 2단계     그림 4. 실험 2 실행 직후    그림 5. 실험 2 실행 결과    그림 6. 실험 2 실행 결과   1. **고찰**   이번 주차에는 Timer/Counter에 대해 배웠다. STOP WATCH를 만들며 이름에 걸맞게 정직한 실험을 했다고 생각한다.  실습 2단계에서는 OVF INTERRUPT를 사용하여 STOP WATCH를 만드는 단계다. 하지만 처음에 OVF 인터럽트를 사용하지 않고 작동하도록 만들고 강의노트에 반드시 OVF 인터럽트를 사용하라는 말이 없어서 그대로 제출할까 잠시 고민했지만 조금 더 시간을 투자하여 OVF 인터럽트를 사용하자 훨씬 깔끔한 코드를 작성하게 되어 뿌듯했다.  다만 시간을 오래 투자해서 완벽하게 실행시켰는데 캡처만으로는 실행 결과를 보이기 부족하다고 생각하여 아래 링크에 실행결과 캡처 영상을 첨부하는 것이 좋을 것 같다. (제가 올린 유튜브 동영상입니다.)  https://youtu.be/8pyulExMaiQ | | | | |
|  | | | | |