|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **결 과 보 고 서** | | | | |
| 학 과 | 학 년 | 학 번 | 조 | 성 명 |
| 전자공학과 | 3 | 12191505 | - | 윤수연 |
| 실험 제목 | LCD | | | |
| 1. **실험 과정**    1. 실습 1단계  |  | | --- | | #include <avr/io.h>  #include <util/delay.h>  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;  }  void display1(char\*str){  cmd(0x80);  int i = 0;  while (str[i] != '\0' ){  data(str[i++]);  }  }  volatile char str[1][16] = {{"YoonSuyeon"}}; // Initial LCD display  int main(){  DDRA=0xff;  DDRB=0xff;  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    display1(str[0]);    while(1){  }    return 0;  } |   1단계 실험은 이번 주에 새로 배운 소자인 LCD를 이용하는 실험이다. 위는 사용한 코드다. cmd함수를 이용해 초기 setting을 해주는데, 각각 레지스터 값을 설정하는 것이 까다로운 부분이었다. 각 위치에서의 역할에 따라 해당 값의 역할을 주석으로 설정했다.  Display 함수는 실제로 배열을 한 글자씩 올리는데 data함수를 통해 작업한다. data함수에는 cmd함수를 통해 레지스터에 간접 접근하여 올리는 내용의 코드가 들어가 있다.  이 때 display1 함수에서는 주소를 인수로 받는다. volatile char str[1][16] = {{"Your Name"}}; // Initial LCD display로 2차원배열로 선언하여 display1(str[0]); 과 display1(str); 이 똑같은 동작을 한다. 따라서 main함수에서 저렇게 불러온다.   * 1. 실습 2단계  |  | | --- | | #include <avr/io.h>  #include <util/delay.h>  void cmd(int command){//data를 lcd에 업로드  PORTB = command;// buffer에 command 전달  PORTA = 0x05;  \_delay\_ms(1);  PORTA = 0x00;// 통신 완료 후 닫기  }  //data receiver.  //0b00000100, IR선택, enable on  //0b00000000, IR선택,enable off  void data(char str){//portB에 data 전송  PORTB = str;  //0b00000101  PORTA = 0x05;  \_delay\_ms(1);  //0b00000001  PORTA = 0x01;  }  void display1(char\*str){//str is 1dimension  cmd(0x80);  int i = 0;  while (str[i] != '\0'){  data(str[i++]);    }  \_delay\_ms(1000);  int j=0;  while(j != 11){//11번 이동필요  j++;  // cmd(0x01);  cmd(0x1a);//00011000  // cmd(0x0c); //0000100 // Display on Cursor Off, Blink off  \_delay\_ms(1000);  }  }  volatile char str[1][50] = {{"ABCDEFGHIJKLMNOPQRSTUVWXYZ"}};// Initial LCD display  int main(){  //set output  DDRA=0xff;  DDRB=0xff;    //0b00111000  cmd(0x38); // Set 8bit 2Line 5x7 dots  //0b00000001  cmd(0x01); // Display clear  //0b10000000  cmd(0x80); // Set DDRAM address or cursor position on display==0번째  //0b00001100  cmd(0x0c); // Display on Cursor Off, Blink off  //0b00000110  cmd(0x06); // Entry Mode    //16글자씩 슬라이스      while(1){  display1(str[0]);    }    return 0;  } |   실습 2단계는 긴 str을 shift하며 dispaly하는 실험이다. 위는 사용한 코드다. 실험 1을 참고하여 몇 개의 레지스터 값을 바꿔 거의 유사한 코드다. 레지스터에 간접 접근하여 shift기능을 사용했다. 주석의 바이너리 코드를 보면 알 것이다. 더 자세한 설명은 이후 이어지는 고찰에 담았다.   1. **실험 결과**    1. 실습 1단계     그림 1. 실험 1 회로구성도    그림 2. 실험 1 실행 직후   * 1. 실습 2단계     그림 3. 실험 2 실행화면 1    그림 4. 실험 2 실행화면 2    그림 5. 실험 2 실행 화면 3   1. **고찰**   지난주 수업에 비해 다소 쉬운 주차였다. lcd소자는 평소 내가 정말 관심이 많은 소자인데, 생각해보니 직접 구동해볼 생각은 하지 못했었다. 그런데 이번 기회에 비록 proteus 프로그램을 이용해 실험한 것이지만 구동해보았다.  코드 작성 단계에서는 Base line이 주어졌기 때문에 16진수로 썼지만 코드 이해를 하는 과정에서는 0과 1을 무수히 써가며 2진수로 이해했다. 이번 주차의 내용에 대해 인터넷에 검색하여 추가로 더 알아봤는데, 우리가 사용한 cmd함수가 lcd구동에 있어서는 필수 부가결한 함수다. 그런데 그 함수 이름은 당연하게도 사람 마다 달랐는데, 안의 내용이 거의 같아서 그 정도면 관련 변수까지 define하여 하나의 헤더 파일을 만드는 것이 더 좋아 보였다. cmd함수니 data함수니 모두 lcd를 구동하는 데에 있어서 비슷하게 구현할 수 밖에 없는데 가독성 면에서 아쉬웠기 때문이다.  이번 주차 2단계는 레지스터의 기능을 이용하는 단계였는데, 처음에 레지스터 동작을 잘못 계산하여 잘 안된다고 판단하여 총 두 가지의 코드를 더 작성했었다. 처음에는 display함수를 두 개로 나누어 한 디스플레이에 16글자가 나오므로 str값을 16개의 문자를 갖는 sub string으로 슬라이스 하여 display하는 코드를 만들었었다. 그 다음으로는 이중 for문을 알파벳의 총 글자 수가 26자이기 때문에 11번 shift된다는 점에서 11번씩 다르게 출력하는 코드였다. 이러한 방법들 모두 lcd의 레지스터를 통해 조정하는 것이 아니라 그저 미리 설정해둔 값을 순서대로 display하는 것뿐이다. 눈속임이라고 볼 수도 있다. | | | | |
|  | | | | |