**전자회로실험 보고서**



학과 전자공학과

학번 1218165

이름 김재욱

제출일 2020-09-24

**실습 이론**

1. GPIO

- General Purpose Input Ouput 의 줄임말으로 마이크로 프로세서가 다른 장치와 통신하기 위해 사용되는 입출력 포트 이다.

- GPIO는 프로그래밍 이전에 역할이 할당 되지 않기 때문에 반드시 입,출력을 설정을 통해 결정되어야 한다.

- GPIO는 입,출력을 제어하기 위해 DDRx,PORTx,PINx 라는 총 3개의 레지스터로 이루어져 있다.

- ATMEGA128에는 PORTA~PORTF : 8bit , PORTG: 5bit 총 53개의 핀이 GPIO핀이다.

- 각 레지스터는 bit 단위로 제어가 가능하다.

DDRx

- 입출력의 방향을 설정하는 레지스터

- DDRA~DDRG

- 1: 출력 0:입력

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| W | W | W | W | R | R | R | R |

=

포트 A의 PA7-PA4 까지 출력포트로 ,PA3-PA0 까지 입력포트로 설정

PORTx

- DDRx포트에서 출력으로 설정된 핀의 데이터를 출력하는 레지스터

- DDRx포트에서 입력으로 설정된 핀의 풀업저항을 설정하는 레지스터

- 1: HIGH/PULL UP 0: LOW/PULL DOWN

DDRA PORTA

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |

포트 A의 PA7,PA4는 HIGH 값 출력, PA6,PA5 는 LOW 값 출력

포트 A의 PA3-PA0 는 풀업저항으로 설정

PULL UP : 스위치를 누르지 않은 상태 HIGH, 스위치를 누른상태 LOW

PULL DOWN : 스위치를 누르지 않은 상태 LOW, 스위치를 누른상태 HIGH

입력인 경우 풀업저항으로 설정하여 플로팅 현상을 방지한다.

PINx

- DDRx포트에서 입력으로 설정되어 있는 핀의 데이터를 입력하는 레지스터

- PINx 레지스터를 통해 입력으로 설정된 핀으로 입력되는 데이터 확인

- 1: 해당핀 입력 데이터 읽기 0: 해당핀 입력 데이터 읽지않기

DDRA PORTA

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

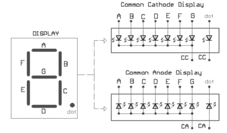
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

하위 4bit 만 입력포트로 설정되었기 때문에 PINA는 하위 4bit에서만 데이터를 읽을 수 있다.

즉 PINx&0b00001111을 통해 하위 4비트가 유효함을 표시할 수 있다.

실습에서 사용된 7segment는 캐소드 7segment이므로 A-G까지 디지털 핀에 연결된 LED들은 VCC가 인가되어야 불이 켜진다. 반대로 애노드 7segments는 A-G까지 디지털 핀에 연결된 LED들은 GND가 인가되어야한다.

즉 각 16진수에 대한 7segment변환은 아래 표와 같다.



**실습내용 및 결과**

**실습 1**

|  |
| --- |
|  |
| **/\***  **\* week1-1.c**  **\***  **\* Created: 2020-09-15 오후 2:15:19**  **\* Author : lobgd**  **\*/**  **#define F\_CPU 16000000**  **#include<avr/io.h>**  **#include<util/delay.h>**  **unsigned char led\_on;**  **void PB\_LEDOnOff(void) // LED ON/OFF**  **{**  **if (led\_on) {**  **PORTB=0b10000000;// LED OFF**  **led\_on = 0;**  **}**  **else {**  **PORTB=0b00000000;// LED ON**  **led\_on = 1;**  **}**  **}**  **int main(void){**  **unsigned char key;**  **DDRB = 0b10000000; // 입출력 방향 설정**  **DDRE = 0b00000000; // 입출력 방향 설정**  **PORTE = 0b10000000; // 풀업저항 설정**  **PORTB = 0b00000000;**  **led\_on = 1;**  **while(1)**  **{**  **key = (PINE & 0b10000000); // 버튼 스위치 값을 읽기**  **switch(key) // key와 값이 일치하면 해당 case 실행문을 실행**  **{**  **case 0b00000000: // 스위치가 눌렸음을 확인**  **PB\_LEDOnOff();**  **\_delay\_ms(500); // 0.5초 시간 지연 발생**  **break;**  **default: // 변수와 값이 불일치하면 default문 실행**  **break;**  **}**  **}**  **}** |

**실습 2**

|  |
| --- |
|  |
| **#define F\_CPU 16000000**  **#define \_\_DELAY\_BACKWARD\_COMPATIBLE\_\_**  **#include<avr/io.h>**  **#include<util/delay.h>**  **unsigned char led\_on;**  **int i =0;**  **double interval =500.0;**  **unsigned char ledArray[8] = {0b10000000,**  **0b01000000,**  **0b00100000,**  **0b00010000,**  **0b00001000,**  **0b00000100,**  **0b00000010,**  **0b00000001};**  **void PB\_LEDOnOff(void) {**  **if(led\_on){**  **PORTB =ledArray[i];**  **\_delay\_ms(interval);**  **PORTB =0b00000000;**  **i++;**  **if(i>8){**  **i=0;**  **}**  **}**  **else{**  **PORTB =0b00000000;**  **}**  **}**  **void change(){**  **if (led\_on) { // LED OFF**  **led\_on = 0;**  **}**  **else { // LED ON**  **led\_on = 1;**  **}**  **}**  **int main(void){**  **unsigned char key;**  **unsigned char up;**  **unsigned char down;**  **DDRB = 0b11111111; // 입출력 방향 설정**  **DDRE = 0b00000000; // 입출력 방향 설정**  **PORTE = 0b11100000; // 풀업저항 설정**  **PORTB = 0b00000000;**  **led\_on = 0;**    **while(1)**  **{**  **key = (PINE & 0b10000000); // 버튼 스위치 값을 읽기**  **up =(PINE & 0b01000000);**  **down =(PINE & 0b00100000);**    **PB\_LEDOnOff();**    **if(key==0b00000000){**  **change();**  **PORTB=0b11111111;**    **}**  **if(up==0b00000000){**  **interval+=500;**  **}**  **if (down==0b00000000){**  **if(interval>0){**  **interval-=500;**  **}**  **}**  **}**  **}** |

|  |
| --- |
|  |
| **/\***  **\* week1-3.c**  **\***  **\* Created: 2020-09-15 오후 2:44:36**  **\* Author : lobgd**  **\*/**  **#define F\_CPU 16000000**  **#include <avr/io.h>**  **#include<util/delay.h>**  **int num =0;**  **unsigned char segment[16] ={0b00111111,0b00000110,0b01011011,0b01001111,**  **0b01100110,0b01101101,0b01111101,0b00100111,**  **0b01111111,0b01100111,0b01110111,0b01111100,**  **0b01011000,0b01011110,0b01111001,0b01110001};**  **int main(void)**  **{**  **DDRB = 0b01111111;**  **DDRE = 0b00000000;**  **PORTB =0b00000000;**  **PORTE = 0b11110000;**    **while (1)**  **{**  **unsigned char first;**  **unsigned char seconde;**  **unsigned char third;**  **unsigned char forth;**  **first = (PINE & 0b00010000);**  **seconde= (PINE & 0b00100000);**  **third = (PINE & 0b01000000);**  **forth = (PINE & 0b10000000);**  **// 버튼이 닫히면 1 스위치가 열리면 0**  **if(first == 0b00000000){ //닫히면**  **num+=1;**  **}**  **if(seconde == 0b00000000){ //닫히면**  **num+=2;**  **}**  **if(third == 0b00000000){ //닫히면**  **num+=4;**  **}**  **if(forth == 0b00000000){ //닫히면**  **num+=8;**  **}**  **PORTB = segment[num];**  **\_delay\_ms(500);**  **num =0;**  **}**  **}** |

**고찰**

**atmega128은 마이크로프로세서로 프로그래밍을 통해 입출력 설정이 가능하다. 입출력을 담당하는 레지스터의 종류에는 3가지가 있으며 각 레지스터마다의 역할이 다름을 알게 되었고 각 레지스터네 0 또는 1을 할당할때의 의미에 대해서도 알수 있었다. 아두이노에서는 내장함수를 통해 INPUT,OUTPUT을 결정하고 데이터를 읽고 출력 할수 이었는데 atmega에서는 핀 하나하나에 대해 비트로 설정하여서 복잡하기도 했지만 처음사용해보는 것이라 익숙하지 않아서 생각했으며 코딩하면서 특히 비트와 핀의 숫자를 헷갈리지 말아야겠다 느꼈다.**

**실습 1에서atmeat128의 17번 핀이 led의 +방향과 연결되어있고 led의 -방향을 접지시켰다.**

**17번 핀은 PB7 이므로 입출력을 설정하는 DDRB에 0b10000000을 전달해 PB7을 출력 모드로 설정한다.**

**즉 PORTB 에 연결된 led를 켜기 위해선 5v를 출력해야 하므로 0b10000000으로 전달한다.**

**반대로 끄기 위해선 0v 를 출력하면 되므로 0b00000000으로 전달한다.**

**반면 버튼은 atmega128의 9번핀인 PE7로 데이터 입력을 받아야 하기 때문에 DDRE에 0b0000000를 전달해 7번째 비트가 입력을 설정할수 있도록 전달한다. PORTE를 통해 풀업저항을 설정하면 버튼은 버튼을 눌렀을때 LOW가 되며 누르지 않았을때 HIGH가된다. if 문을 통해 버튼이 눌렸음을 PINE의 7번째 비트가 0임을 확인하여 알수 있다. 특히 key&(PINE & 0b10000000)는 오직 7번째 비트만 입력값을 읽어들임을 의미한다.**

**그후 led의 전원을 제어한다.**

**실습 2에선 버튼 3개를 이용해 전원제어, delay증가, delay 감소를 구현하고 8개의led를 차례로 껐다 켜는 제어이다. 먼저 배열을 생성해서 index별로 하나의 led를 제어할 수 있게 하였다. 그리고 led제어 함수인 PB\_LEDOnOff에서 배열의 index를 증가시켜 차례대로 led점멸한 후 배열의 마지막 index에서는 0으로 초기화 하여 계속해서 점멸할수 있도록 했다.**

**key,up,down변수를 이용해 PINE의 데이터를 읽었다. 각 변수의 데이터값을 if 문을 통해 전원,delay up delay down 을 동작하게 되었다. interval이라는 매개변수를 이용하여 0.5초씩 증감 하도록 햇으며 추가로 delay down의 조건으로 interval이 0보다 작을경우는 감소하지 않게 하여 매개변수로 음수가 전달 되지 않게 하였다.**

**특히 \_delay\_ms()함수에 매개변수로 상수가 아닌 변수를 쓰게 될경우 컴파일 오류가 발생하게 된다. #define \_\_DELAY\_BACKWARD\_COMPATIBLE\_\_를 #include<util/delay.h>앞에 추가 해주어 컴파일 오류를 해결할 수 있다.**

**실습 3에서는 캐소드 7 세그먼트를 이용해 2진수를 16진수로 표현하였다. 기본 알고리즘은 PE4,PE5,PE6,PE7순서대로 각 2진수의 4자리를 할당하였으며(차례로 1의자리,2의자리,4의자리, 8의자리) 2진수를 10진수로 변환하는 방법인 각자리의 2의n승을 이용해 자리수마다 수를 더해 10진수를 만들었다. 예를들어 PE4,PE5,PE6,PE7순서대로 1,1,0,1이면 1+2+2\*2\*2 =11 로 계산하였다 16진수를 표현하는 7세그먼트의 조합은 배열로 만들어 놓고 변환된 10진수가 7세그먼트의 조합의 배열의 인덱스이다. PORTB 에 7세그먼트의 조합의 배열을 대입해 출력할 수 있었다.**