|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **결 과 보 고 서** | | | | |
| 학 과 | 학 년 | 학 번 | 조 | 성 명 |
| 전자공학과 | 3 | 12191505 | - | 윤수연 |
| 실험 제목 | GPIO | | | |
| 1. **실험 과정**    1. 실습 1단계  |  | | --- | | #include <inttypes.h>  #include <avr/io.h>  #include <avr/interrupt.h>  #include <avr/sleep.h>  #include <util/delay.h>  #define F\_CPU 16000000  unsigned char led\_on;  void PB\_LEDOnOff(void)  {  if(led\_on){  PORTA = 0b10000000;  led\_on = 0;  }    else{  PORTA = 0b00000000;  led\_on = 1;  }  }    int main()  {  unsigned char key;  DDRA = 0b10000000;  DDRE = 0b00000000;  PORTE = 0B10000000;  PORTA = 0b00000000;  led\_on = 1;  while (1)  {  key = (PINE & 0b10000000);    switch(key)  {  case 0b00000000:  PB\_LEDOnOff();  \_delay\_ms(500);  break;  default:  break;  }  }  } |   1개의 버튼 입력이 들어올 때마다 1개의 LED on/off를 반복하도록 구현하는 실험이다. 입력과 출력 port가 각각 하나씩 기능하여 각 기능에 port 하나씩을 부여했다. 코드는 led on/off를 조절하는 하나의 함수를 따로 설정하고 메인함수로 이루어져 있는데 비트연산을 통해 key가 입력됨을 무한 루프 안에서 계속 감지하고 입력 시 마다 동작한다.  button소자의 사용이 처음이라 이 부분이 어려울 것 같았는데 소자입력만 해주고 클릭으로 사용하는 소자라 전혀 어렵지 않았다.     * 1. 실습 2단계  |  | | --- | | #define F\_CPU 16000000  #include <avr/io.h>  #include <avr/interrupt.h>  #include <avr/sleep.h>  #include <util/delay.h>  int delay;  unsigned char isOn;  unsigned char state;  void DelayApp(int ms)  {  //ms를 소모하는 방식  while(ms != 0){  \_delay\_ms(1);  ms -= 1;  }  }  void InOnState(void){  if(state == 0b11111111){  state = 0b11111110;  }  PORTA=state;  DelayApp(delay);  state = state<<1|0b00000001;  DelayApp(delay);  }    void ON\_OFF(void){  if(isOn==1){  isOn=0;  PORTA = 0b00000000;  DelayApp(500);  }  else{ //repair  isOn=1;  }  }  int main(void)  {  unsigned char key;  DDRA = 0b11111111; // 출력 방향 설정  DDRE = 0b00000000; // 입력 방향 설정  PORTE = 0b11100000; // 풀업저항 설정  PORTA = 0b11111111;    delay = 20;  isOn = 0;//off  state = 0b11111110; //작동 시 초기위치는 0번 핀이다.    while (1){  if(isOn==1)  InOnState();    key = (PINE & 0b11100000);    switch(key){  case 0b01100000://1st pin  ON\_OFF();  break;    case 0b10100000://2nd pin  if(delay==2)//limit  break;  delay -= 7; //delay is down == speed up  break;    case 0b11000000: // 3rd pin  if(delay==200)//limit  break;  delay +=7;//delay is up == speed down  break;    default:  break;  }  }    return 0;  } |   위는 사용한 코드다. 전역변수로는 속도를 조절하게 될 speed에 반비례 관계를 가지는 delay변수와 현재 상태를 저장하여 다음 on시에 off할 때와 같은 상태를 저장하여 동작을 멈추었다 다시 시작했을 때도 진행 상황을 이어나갈 state변수, 마지막으로 On상태인지 off상태인지 binary로 저장할 변수 isOn이다. 가장 먼저 오는 함수는 header파일을 include하여 많이 쓰는 \_delay\_ms()함수에 매개변수로 변수를 입력하면 오류가 떠서 이를 반복문으로 다시 구현한 함수다. 원하는 ms단위의 시간을 넣고 그만큼 반복문을 돌려 ms를 다 소모할 때까지 반복문을 돌며 시간만큼 delay를 주는 방법이다.  다음으로 InOnState()는 동작을 실행할 때와 실행하지 않을 때로 나누어서 동작할 때의 동작을 구현했는데 비트연산을 사용해야했다. shift연산자를 사용하면 0으로 빈 자리를 메꾸게 되는데 원하는 동작을 하기 위해서는 비트 연산을 해서 값을 누적할 수 있도록 했다.  그리고 동작 버튼을 눌렸을 때 on/off 상태 변경을 담당하는 함수도 있다. isOn변수가 핵심인데, 각 상황을 반전시켜주는 역할을 한다. 그런데 이때 버튼을 누르는 건 사람인지라 누르는 시간이 좀 길어지면 버튼을 반복하여 누는 것처럼 동작하는 문제 때문에 이 함수 안에도 delay를 두었다.  마지막으로 main함수 안에는 초기 값을 설정하고 각 레지스터에 접근하여값을 설정해준다. 그리고 각 핀에 대한 역할을 부여하는데 첫번째로 8번핀, 가장 끝 핀은 on,off를 담당하는 핀으로 위에서 구현한 대부분 함수를 사용하여 역할을 부여했고 7번, 6번핀은 각각 속도를 올리고 낮추는 역할로, 전역변수인 delay에 접근하여 delay시간을 조절하여 speed를 올리고 낮춘다. 이 때 delay 와 speed는 반비례 관계임을 염두에 둔다.   1. **실험 결과**    1. 실습 1단계        * 1. 실습 2단계        1. **고찰**   패기와 다르게 2단계 구현만으로도 아주 힘들었다.1단계에서 Atmel Studio나 ToolChain 등 추가 프로그램을 설치하여 .hex파일로 동작해보려 했으나 오류가 너무 많이 나고 컴파일러의 충돌로 프로그램이 자꾸 꺼지는 관계로 proteus프로그램 내에서 컴파일했다. 또한 Atmega128 소자를 반전하는 것이 실험 결과에 영향을 미쳤는데, 이 부분 또한 어떠한 오류같다. 이런 자잘한 오류를 극복하는 과정에서 시간을 소비했다.  두번째 실험에서는 while반복문으로 delay를 처리했는데 컴퓨터가 처리하는 속도가 있어서인지 실제 delay 시간보다 더 걸리는 것을 보여주었다. 또 속도차이는 아주 작은 delay차이로도 확연히 느낄 수 있었는데, 정확히 속도를 잴 수는 없었지만 고작 7ms 다르게 했는데 느낄 수 있다는 것은 말이 안되고 아마도 컴퓨터 연산 속도가 꽤 느려서 while문을 도는 횟수가 주는 것이 체감이 바로 되는 것 같다. | | | | |
|  | | | | |