<**입출력 포트**>

실습보드 제작 및

입출력 포트 코딩

(LED와 스위치)

1. **목적**

- 실습 보드를 직접 제작하기

- 교제에 나온 프로그램 언어에 대해 알아보고 직접 코딩 하기.

- LED를 통해서 코딩 결과를 확인해보자.

1. **코딩 실습**
2. **단순 LED 점등 (6.1)**

* **헤더파일”Avr/io.h”을 소스파일에 포함시키고 DDRA와 PORTA를 변수인 것처럼 사용하면 된다. 따라서 포트A를 출력 포트로 설정하고 데이터 0x5D 출력으로 하는 코드를 작성한다. 그리고 while문을 사용해서 무한루프를 일으켜서 전원이 꺼질 때까지 종료되지 않도록 한다**
* **예상:LED중에서 왼쪽부터 1,3,7번째에만 불이 들어오고 나머지는 불이 들어오지 않음.**
* **While: ()의 값이 참이면 {}에 들어있는 프로그램을 수행하고 아니면 while문에서 빠져나가게 된다.**

**<소스 코드>**

**#include <avr/io.h>**

**int main(void)**

**{**

**DDRA=0xFF; // 포트A를 모두 출력 포트로 설정**

**while (1) // 무한루프,**

**{**

**PORTA=0x5D; //점등 패턴으로 LED를 점등(01011101)**

**}**

**}**

전자기기, 회로, 컴퓨터이(가) 표시된 사진

자동 생성된 설명

* **결과: 아래사진은 실행 결과로 1,3,7번째에만 불이 들어온 것을 확인할 수 있습니다.**

1. **패턴을 변경해 가면서 LED 점등 (6.2)**

* **두번째에서는 첫번째 코드에서 while문안에 여러가지 출력 코드를 입력하여 패턴을 연속적으로 변경하는 LED 소스를 실행하는 것이다.**
* **예상:0xFE, 0xFD, 0xFB…. 0x7F를 소스를 넣고 실행할 경우에 처음에는 젤 오른쪽에 있는 것에만 불이 들어오고 다음 번에는 왼쪽으로 이동하면서 불이 들어오고 원래 들어온 불은 꺼지는 상태가 반복된다.**

**<소스 코드>**

**#include <avr/io.h>**

**int main()**

**{**

**DDRA=0xFF; //포트A를 모두 출력 포트로 설정**

**while (1) //무한루프**

**{**

**PORTA=0xFE; //PA0을 점등(11111110)**

**PORTA=0xFD; //PA1을 점등(11111101)**

**PORTA=0xFB; //PA2을 점등(11111011)**

**PORTA=0xF7; //PA3을 점등(11110111)**

**PORTA=0xEF; //PA4을 점등(11101111)**

**PORTA=0xDF; //PA5을 점등(11011111)**

**PORTA=0xBF; //PA6을 점등(10111111)**

**PORTA=0x7F; //PA7을 점등(01111111)**

**}**

**}**

전자기기, 회로, 컴퓨터이(가) 표시된 사진

자동 생성된 설명

* **결과: 아래 사진은 실행 결과로 패턴 이동이 너무 빨라 LED 점등이 모두 켜져 있는 것처럼 보이게 됩니다.**

1. **시간지연 함수를 이용한 LED 점등 (delay함수, 6.3)**

* **세번째는 두번째에 발생하는 점등 패턴이 변하지 않고 희미하게 모두 켜져 있는 것처럼 보이는 현상을 없애기 위해서 패턴과 패턴의 이동 사이에 시간 지연 함수인 msec\_delay()함수를 사용하였다.**
* **또한 void msec\_dalay(int n)을 선언하여서 msec\_dalay는 “함수이며 인자로서 int형 변수 하나를 받으면 변환값이 없다”라는 것을 프로그램에 인식시킨다.**
* **시간 지연이 필요한 경우 시간 지연 함수에 대한 반복루프를 만들어 내어야한다. for문을 사용하여서 시간 지연 함수내에서 루프를 반복시키고 있습니다.**
* **예상: 1초마다 LED의 불이 왼쪽으로 이동하는 것을 확인할 수 있습니다.**

**<소스 코드>**

**#include <avr/io.h>**

**void msec\_delay(int n); //시간지연함수를 선언**

**int main()**

**{**

**DDRA=0xFF; //포트 A를 모두 출력 포트로 설정**

**while (1) //무한루프**

**{**

**msec\_delay(1000); //1초의 지연이 발생**

**PORTA=0xFE;**

**msec\_delay(1000);**

**PORTA=0xFD;**

**msec\_delay(1000);**

**PORTA=0xFB;**

**msec\_delay(1000);**

**PORTA=0xF7;**

**msec\_delay(1000);**

**PORTA=0xEF;**

**msec\_delay(1000);**

**PORTA=0xDF;**

**msec\_delay(1000);**

**PORTA=0xBF;**

**msec\_delay(1000);**

**PORTA=0x7F;**

**}**

**}**

**void msec\_delay(int n) //시간 지연함수**

**{**

**int i;**

**for(;n>0;n--) //1msec 루프를 n회 반복**

**{**

**for(i=0;i<2000;i++) //1msec 시간 지연 루프**

**{**

**asm("nop"::); asm("nop"::);**

**//1클록을 쉬는 어셈블리 nop명령을 프로그램에서 수행하도록 하는 것이다.**

**}**

**}**

**}**

* **결과: 동영상에서 두번째에서 발생하는 문제점을 해결하여 1초다마 점등이 변하는 것을 확인할 수 있습니다.**

1. **배열패턴을 사용한 LED 점등 (6.4)**

* **세번째 프로그램에서 LED에 표시되는 형태가 바뀌면 while문 내의 LED 점등 부분을 일일이 변경하여야 할 뿐 아니라 패턴의 형태가 8가지보다 더 많게 되면 while문 내에 패턴 점등 부분을 추가하여야 한다. 패턴의 형태가 100개 정도 만들게 될 경우 프로그램 작성이 번거롭게 됩니다. 이러한 문제를 해결하기 위해서 네번째 프로그램에서는 패턴의 배열을 사용한 테이블에 저장하며 사용하는 unsigned char pattern[]을 사용합니다.**
* **if(++i==8)i=0; i의 값을 8이 될 때 까지 1를 증가시키는 것으로 8일 되면은 0으로 초기화하는 코드이다.**
* **예상: 프로그램이 구동하는 모습은 세번째와 일치합니다.**

**<소스 코드>**

**#include <avr/io.h>**

**void msec\_delay(int n); //시간 지연 함수 선언**

**unsigned char pattern[8] = {0xFE,0xFD,0xFB,0xF7,0xEF,0xDF,0xBF,0x7F};**

**// 8가지의 패턴을 배열 함수에 저장한다.**

**int main()**

**{**

**int i=0; //패턴 인덱스**

**DDRA=0xFF;**

**PORTA=pattern[i]; //처음 패턴으로 LED를 켠다.**

**while (1)**

**{**

**msec\_delay(1000); //1sec 지연**

**if(++i==8)i=0; //8번째패턴에서 인덱스 리셋**

**PORTA=pattern[i]; //i번째 패턴으로 점등**

**}**

**}**

**void msec\_delay(int n) //msec 단위 지연**

**{**

**int i;**

**for(;n>0;n--)**

**{**

**for(i=0;i<2000;i++)**

**{**

**asm("nop"::); asm("nop"::);**

**}**

**}**

**}**

* **결과: 동영상에서 확인할 수 있듯이 실행되는 결과는 세번째 프로그램과 동일 합니다. 그러나 세번째보다 패턴의 형태가 용이합니다.**

1. **스위치를 연결한 LED 점등 (6.5)**

* **5번째 프로그램에서는 LED에 스위치를 연결하여 보자.**
* **4번째 프로그램에 시간 지연 함수를 대신에 스위치의 누름을 감시하는 프로그램을 삽입하여 스위치를 누를 때마다 패턴을 이동시키는 프로그램을 만들어보자**
* **스위치를 입력함수로 사용(포트D)하고 LED를 출력함수(포트A)로 설정하고 패턴을 누를 때마다 패턴이 이동하므로 while(!(~PIND&0x01))를 사용한다.**
* **!: not을 의미하고 있는 연산자로 입력 값의 반대가 나오게 된다.**
* **&: and연산자를 의미하고 두 개의 입력 값이 둘다 1일 때 1을 출력한다.**
* **예상: 스위치를 누르고 있는 동안 LED의 불이 계속 해서 바뀔 것 이다.**

**<소스 코드>**

**#include <avr/io.h>**

**Unsigned char pattern[8]**

**= {0xFE,0xFD,0xFB,0xF7,0xEF,0xDF,0xBF,0x7F};**

**int main()**

**{**

**int i=0; // 패턴 인덱스**

**DDRA=0xFF; // 포트A를 모두 출력 포트로 설정**

**DDRD=0x00; // 포트D를 모두 입력 포트로 설정**

**PORTA=pattern[i]; // 처음 패턴으로 LED를 켠다**

**while (1)**

**{**

**while(!(~PIND&0x01)) // 스위치 누름을 기다림**

**if(++i==8)i=0; // 인덱스가 8이면 리셋**

**PORTA=pattern[i]; // i번째 패턴으로 LED를 켠다.**

**}**

**}**

* **결과: 동영상에서 확인할 수 있듯이 스위치를 누르게 되면은 LED에 들어오는 불이 계속 해서 바뀌는 것을 확인할 수 있습니다.**

1. **스위치를 누르면 연속해서 불이 들어오는 LED 점등 (6.6)**

* **프로그램5의 실행 결과를 보면은 스위치를 누르고 있는 동안 패턴이 회전하나 너무 빨라 감지할 수가 없고, 스위치를 떼면 회전이 멈추게 된다. 이를 보완하기 위하여 스위치를 뗐다 누르면 패턴이 회전하도록 하자.**
* **프로그램5의 while문 안에 while(~PIND&0x01)을 맨 마지막에 넣어 스위치가 떨어지는 것을 기다리도록 한다.**
* **예상: 5번째와는 다르게 스위치를 누르고 뗄 때마다 LED점등의 불의 위치가 변경할 것이다.**

**<소스 코드>**

**#include <avr/io.h>**

**unsigned char pattern[8]**

**= {0xFE,0xFD,0xFB,0xF7,0xEF,0xDF,0xBF,0x7F};**

**int main()**

**{**

**int i=0;**

**DDRA=0xFF;**

**DDRD=0x00;**

**PORTA=pattern[i];**

**while (1)**

**{**

**while(!(~PIND&0x01)) // 스위치 누름을 기다림**

**if(++i==8)i=0; // 인덱스가 8이면 리셋**

**PORTA=pattern[i]; // i번째 패턴으로 LED를 켠다.**

**while(~PIND&0x01); // 스위치 떨어짐을 기다림**

**}**

**}**

* **결과: 동영상에서 확인 할 수 있듯이 스위치를 누르고 뗄 때마다 LED의 불의 위치가 바뀌는 것을 확인 할 수 있습니다.**

1. **스위치 디바운싱을 이용한 연속하는 LED 점등 (6.7)**

* **프로그램6의 실행 결과를 보면은 한 번의 스위치 동작에 두 번 이상의 패턴 회전이 자주 발생한다. 프로그램7에서는 이런 현상은 방지하는 소스를 넣어보자.**
* **두 번 이상의 패턴 회전이 일어나게 하지 않기 위해서 while문 안에 시간 지연 함수를 추가 한다 .( 디바운싱)**
* **디바운싱을 할 때 지연 시간을 짧게 하면 디바운싱이 완전하지 않을 수 가 있다.**
* **프로그램6에서 발생하는 연속해서 LED 불이 이동하는 현상이 사라질 것이다.**

**<소스 코드>**

**#include <avr/io.h>**

**unsigned char pattern[8]**

**= {0xFE,0xFD,0xFB,0xF7,0xEF,0xDF,0xBF,0x7F};**

**int main()**

**{**

**int i=0;**

**DDRA=0xFF;**

**DDRD=0x00;**

**PORTA=pattern[i];**

**while (1)**

**{**

**while(!(~PIND&0x01)) // 스위치 누름을 기다림**

**msec\_delay(20); // 디바운싱을 위한 시간 지연**

**if(++i==8)i=0;**

**PORTA=pattern[i];**

**while(~PIND&0x01); // 스위치 떨어짐을 기다림**

**msec\_delay(20); // 디바운싱을 위한 시간 지연**

**}**

**}**

* **결과: 스위치가 눌리었다 뗄 때 마다 LED의 불이 이동하고 그 이동이 순차적으로 보이게 된다.**

1. **스위치 입력을 사용하여 LED를 순차적으로 점등 (6.6-C)**

* **8번째 프로그램에서는 #define 키워드를 사용하여 매크로를 정의한다.**
* **매크로를 사용하는 이유는 프로그램의 의미를 확실하게 전달하는 것과 하나의 상수를 여러 군데 사용할 때 용이하다.**
* **#define DEBOUNCING\_DELAY 20 이라는 매크로 함수를 사용하여 디바운싱을 위한 지연 시간을 나타내고 있다. 그러므로 원래 시간 값대신에 delay 함수에 DEBOUNCING\_DELAY를 입력한다.**
* **예상: 실행 결과는 프로그램 7과 일치할 것이라고 예상 됩니다.**

**#define F\_CPU 7372800UL //avr크리스탈 진동수**

**#include <avr/io.h> // I/O 레지스터 정의**

**#include <util/delay.h> // 시간 지연 함수용 헤더파일**

**#define DEBOUNCING\_DELAY 20 // 디바운싱 지연 시간**

**void msec\_delay(int n); // 시간 지연 함수**

**unsigned char pattern[8]**

**= {0xFE,0xFD,0xFB,0xF7,0xEF,0xDF,0xBF,0x7F};**

**int main()**

**{**

**int i=0;**

**DDRA=0xFF; // 포트 A를 출력으로 설정**

**DDRD=0x00; // 포트 B를 입력으로 설정**

**PORTA=pattern[i]; // 처음 패턴으로 LED를 켠다.**

**while (1)**

**{**

**while(!(~PIND&0x01)) // 스위치 누름을 기다림**

**msec\_delay(DEBOUNCING\_DELAY); // 시간 지연**

**if(++i==8)i=0; // 마지막 패턴에서 인덱스 리셋**

**PORTA=pattern[i]; // i번째 패턴으로 점등**

**while(~PIND&0x01); // 스위치 떨어짐을 기다림**

**msec\_delay(DEBOUNCING\_DELAY); // 시간 지연**

**}**

**}**

**void msec\_delay(int n)**

**{**

**for(;n>0;n--) // 1msec 시간 지연을 n회 반복**

**\_delay\_ms(1); // 1msec 시간 지연**

**}**

* **결과: 동영상에 확인 할 수 있든이 실행 결과는 프로그램 7과 일치하는 형태인 스위치가 눌러지거나 떼질 때마다 LED의 불의 위치가 바뀝니다.**