실험제목: 범용 입출력 장치 (GPIO: General-Purpose Input Output)

실험목적

C-언어를 사용하여 범용 입출력 장치를 제어하는 프로그램을 작성하는 능력을 배양한다.

실험 준비물

Microchip Studio (Atmel Studio 7)
Atmega328PB Xplained Mini Board
Seven Segment Display
Array Resistor (330 ohm x 7)
Breadboard

실험에 필요한 예비지식

1. C-언어 연산자

산술 연산자 (+, -, *, /, %)
대입 연산자 (+=, -=, *=, /=)
비교 연산자 (>, <, >=, <=, ==, !=)
논리 연산자 (&&, ||, !)
비트 연산자 (&, |, ^, ~)
쉬프트 연산자 (<<, >>)

1) 산술 연산자

연산자	기능	사용 예
+	더하기	a = b + c;
-	빼기	a = b - c;
*	곱하기	a = b * c;
/	나누기	a = b / c;
%	나머지	a = b % c;
++	1 증가	a++ 또는 ++a
--	1 감소	a-- 또는 --a

2) 대입 연산자

연산자	기능	사용 예
+=	더하기	a += 2;
-=	빼기	b -= 2;
*=	곱하기	a *= 2;
/=	나누기	b /= 2;

3) 비교 연산자

연산자	기능	사용 예
>	보다 크다	if (a > b)
<	보다 작다	if (a < b)
>=	이상	if (a >= b)
<=	이하	if (a <= b)
==	같다	if (a == b)
!=	다르다	if (a != b)

4) 논리 연산자

연산자	기능	사용 예
&&	논리 곱(그리고)	if ((a > b) && (a < c))
||	논리 합(또는)	if ((a > b) || (a < c))
!	논리 부정 (~가 아니다)	if (!(a > b))

5) 비트 연산자

연산자	기능	사용 예
&	AND	a = b & c;
|	OR	a = b | c;
^	XOR	a = b ^ c;
~	NOT	a = ~b;

6) 기타

연산자	기능	사용 예
<<	왼쪽 시프트	a = b << 4
>>	오른쪽 시프트	a = b >> 4
sizeof	바이트 사이즈 반환	a = sizeof(b);
?:	조건문	a = b > 1 ? 2 : 3;

2. C-언어에서 프로그램의 흐름 제어

1) 순차 진행

2) 조건 분기

(1) `if` 문 사용법

```
if (expression)
{
    statements
}
```

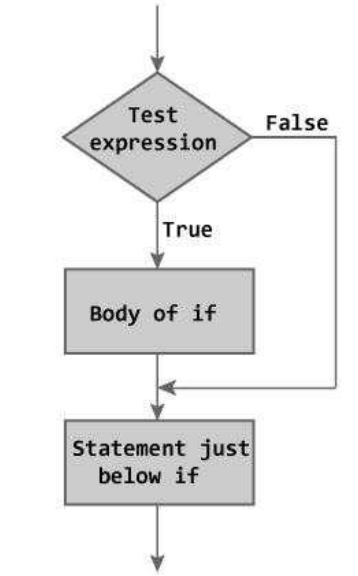


Figure: Flowchart of if Statement

<그림 3-1> `if` 문 흐름도

(2) if – else 문 사용법

```
if (expression)
{
    statements
}
else
{
    statements
}
```

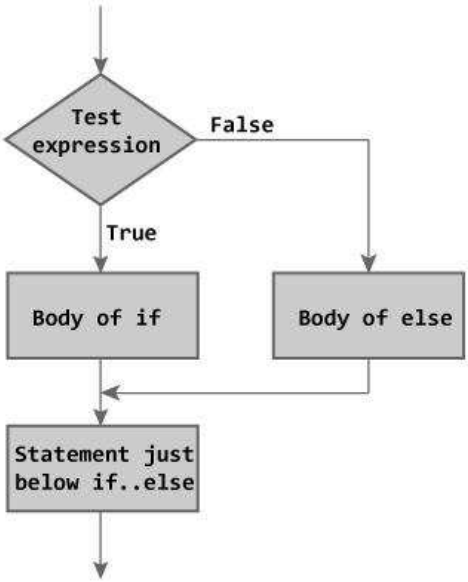


Figure: Flowchart of if...else Statement

<그림 3-2> if – else 문 흐름도

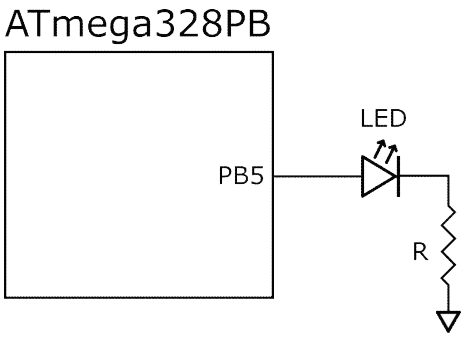
(3) 반복

while – loop
for – loop
do ~ while loop

실험내용

1. LED blinking 예제

ATmega328PB XMini Board에서 PB5에 연결된 LED가 100msec 동안 점등되고 이어서 500 msec 동안 소등되는 동작을 반복하도록 밑줄 친 부분을 채워 프로그램을 완성시킬 것.

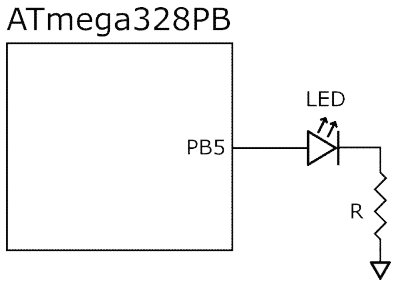


- `while` - loop
- Bitwise OR 연산자 (`|`)
- Bitwise NOT 연산자 (`~`)
- Bitwise AND 연산자 (`&`)
- Shift 연산자 (`<<`, `>>`)
- AVR GCC에서의 data types (종류 및 크기)
- library 함수 사용법: `_delay_ms()`

```
#define _________________   // _delay_ms()를 사용하기 위한 CPU 클럭 주파수 정의

#include <avr/io.h>         // 각종 레지스터의 주소가 선언된 header file 포함
#include <_____________>    // _delay_ms()의 prototype이 선언된 header file 포함

void main (void)
{
   DDRB  __________;         // (1) PB5를 OUTPUT mode로 설정
                             //     단, 다른 비트 값은 변경하지 말고 bit 5만을 설정
                             //     Bitwise OR 연산자 (|) 사용

   while (__)                // (2) 무한 loop가 되도록 ()속의 값을 설정
   {
      PORTB ___________;     // (3) PB5에 연결된 LED를 점등
                             //     단, 다른 비트 값은 변경하지 말고 bit 5만을 설정
                             //     Bitwise OR 연산자 (|) 사용
      _delay_ms(100);

      PORTB __________;      // (4) PB5에 연결된 LED를 소등
                             //     단, 다른 비트 값은 변경하지 말고 bit 5만을 설정
                             //     Bitwise NOT 연산자 (~) 및
                             //     Bitwise AND 연산자 (&) 사용
      _delay_ms(500);
   }
}
```

2. LED Toggling 예제

ATmega328PB XMini Board에서 PB5에 연결된 LED가 500msec 간격으로 점등과 소등을 무한히 반복하도록 밑줄 친 부분을 채워 프로그램을 완성시킬 것.

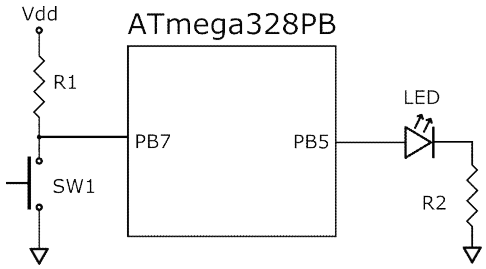


- Bitwise OR 연산자 (|)
- Bitwise XOR 연산자 (^)

```
#define _________________   // _delay_ms()를 사용하기 위한 CPU 클럭 주파수 정의

#include <avr/io.h>         // 각종 레지스터의 주소가 선언된 header file 포함
#include <_____________>    // _delay_ms()의 prototype이 선언된 header file 포함

void main (void)
{
   DDRB  __________;         // (1) PB5를 OUTPUT mode로 설정
                             //     단, 다른 비트 값은 변경하지 말고 bit 5만을 설정
                             //     Bitwise OR 연산자 (|) 사용

   while (__)                // (2) 무한 loop가 되도록 ()속의 값을 설정
   {
      PORTB ___________;     // (3) PB5에 연결된 LED를 점등
                             //     단, 다른 비트 값은 변경하지 말고 bit 5만을 설정
                             //     Bitwise XOR 연산자 (^) 사용
      _delay_ms(500);
   }
}
```

3. 입출력 장치의 동기화 예제

ATmega328PB XMini Board에서 PB7에 연결된 스위치를 누를 때에만 PB5에 연결된 LED가 점등되도록 주석을 참조하여 밑줄 친 부분을 채워 프로그램을 완성시킬 것.



- `if – else` statement
- 관계연산자 (`==`)

1) 스위치의 눌림 상태와 논리값의 관계

(1) 스위치 SW1이 눌리지 않았을 때:
GND 단자를 기준으로 한 PB7의 전압은 Vdd. 논리값으로는 '1'에 해당

(2) 스위치 SW1이 눌렸을 때:
GND 단자를 기준으로 한 PB7의 전압은 0V. 논리값으로는 '0'에 해당

(3) 따라서 PB7으로 입력되는 신호의 전압을 읽어 논리값으로 변환하면 스위치 SW1의 눌림 상태를 판단할 수 있다.

2) 프로그램을 통해 스위치의 눌림 상태를 판단하는 방법

(1) PB7으로 입력되는 신호의 논리값은 PINB 레지스터의 bit 7이 나타내고 있음.

(2) PINB 레지스터와 상수 0b10000000를 bitwise AND 연산을 취하여 그 결과가 '0'이라면 PB7으로 입력되는 신호의 논리값이 '0'이라는 뜻이고 이는 스위치 SW1이 닫혔음을 의미한다.

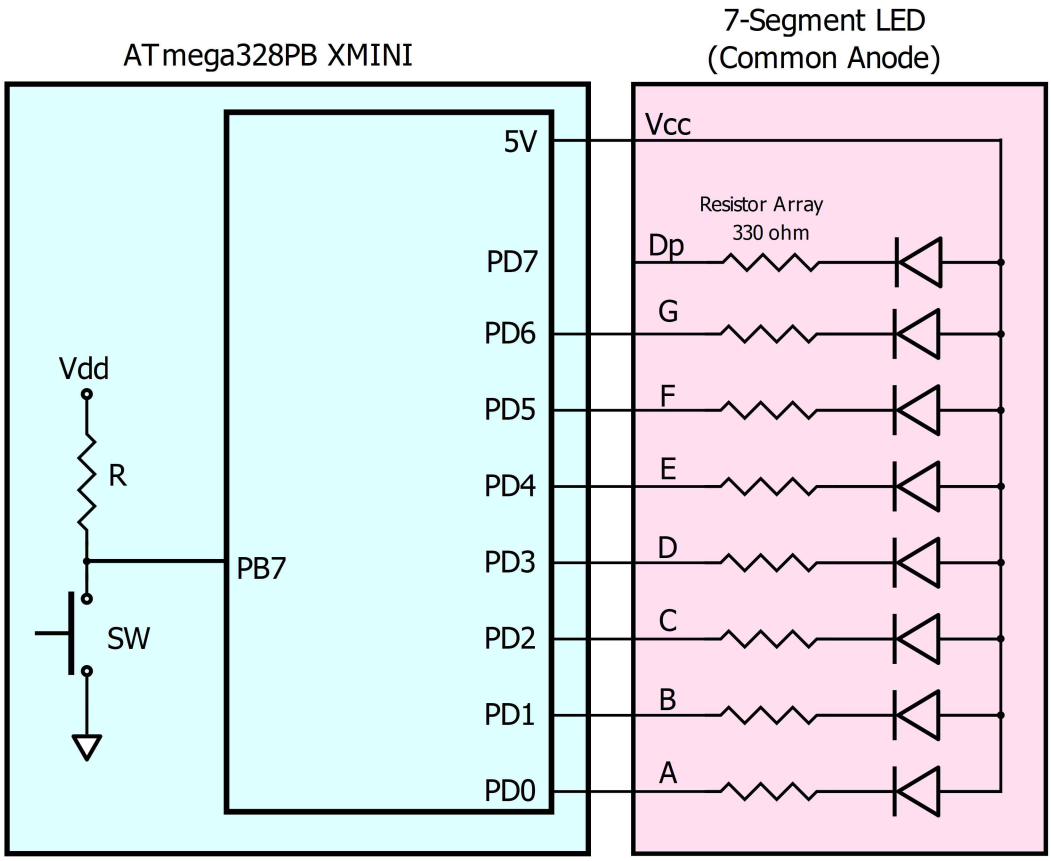
(3) PINB 레지스터와 상수 0b10000000를 bitwise AND 연산을 취하여 그 결과가 '0'이 아니라면 PB7으로 입력되는 신호의 논리값이 '1'이라는 뜻이고 이는 스위치 SW1이 열려있음을 의미한다.

(4) bitwise AND 연산자와 `if` statement를 사용하여 위의 방법을 C-언어로 구현하면 프로그램을 통해 스위치의 눌림 상태를 판단할 수 있다.

```
#include <avr/io.h>          // 각종 레지스터의 주소가 선언된 header file 포함

void main (void)
{
    DDRB  _________;         // (1) PB5를 OUTPUT mode로 설정
                             //     단, 다른 비트 값은 변경하지 말고 bit 5만을 설정
                             //     Bitwise OR 연산자 (|) 사용

    while (__)                   // (2) 무한 loop가 되도록 ()속의 값을 설정
    {
        if (________________)    // (3) 만일 PB7에 연결된 스위치가 눌렸으면(ON 이면)
        {
            _______________      //  (4) PB5에 연결된 LED 점등
        }
        else                     // 만일 PB7에 연결된 스위치가 OFF이면
        {
            _______________      // (5) PB5에 연결된 LED 소등
        }
    }
}
```

4. Mod-10 counter 예제

아래의 그림과 같이 회로를 구성하고, PB7에 연결된 스위치를 누를 때마다 10진 카운터(mod-10 counter)의 값을 1씩 증가시킨 후, 그 값이 PD[6..0]에 연결된 SSD(제품명: SSR-1056K)에 표시되도록 밑줄 친 부분을 채워 프로그램을 완성시키시오. 단, 스위치 바운싱(bouncing)은 무시하시오.



```
/* Lab_4.c */

#define SWITCH          7               // switch is connected to PB7

#define F_CPU           16000000UL
#define MAX_COUNT       10

#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
    uint8_t counter = 0;        // mod-10 counter

    // LED font array for digit 0 to 9
    uint8_t led_table[MAX_COUNT] = {_____________________________________________};

    // set PD[6..0] to OUTPUT mode
    DDRD ___________________;

    // display initial counter value on SSD
    PORTD ___________________;

    while (1)
    {
        // if SW is pressed
        if (_____________________________________)
        {
            // increment counter by 1
            ________________________;

            // reset counter when it reaches MAX_COUNT
            ____________________________;

            // display counter value to 7-seg LED
            PORTD ________________________;

            // wait until SW is released
            while (_____________________________);
        }
    }
}
```

숙제

Mod-10 down counter 예제

아래의 그림과 같이 회로를 구성하고, PB7에 연결된 스위치를 누를 때마다 10진 카운터(mod-10 counter)의 값을 1씩 감소시킨 후, 그 값이 PD[6..0]에 연결된 SSD에 표시되도록 프로그램을 작성하시오. 단, 카운터 초기값은 0이고, 9부터 0까지의 계수를 반복하며, 스위치 디바운싱(debouncing)은 무시하시오.