3 포트 입·출력

이번 장부터 본격적으로 MPLAB X tool을 사용하여 PIC 마이크로컨트롤러를 코딩하는 방법에 대한 설명이 이루어진다. 이번 장에서는 먼저 가장 기본적인 포트 입·출력 제어를 연습해 보기로 하자. 입·출력 제어를 연습할 때 시각적인 효과를 위해서 발광소자인 LED(Light Emitting Diode)를 활용하기로 한다. LED를 점등하는 패턴을 다양한 형태로 제어해 봄으로써 프로그래밍의 기초 능력을 배양하기로 한다.

3.1 사용부품 소개

상당수의 마이크로프로세서 또는 마이크로컨트롤러는 수 mA에 불과한 포트 출력 전류를 가지고 있어서 직접적으로 LED를 구동하기에 무리가 있다. 그에 비해, PIC 마이컴의 경우에는 포트 출력 전류가 수십 mA까지 가능하므로 LED의 직접 구동이 가능하다는 장점을 가진다. 따라서 본 교재에서는 LED를 출력 포트에 직접 연결하는 방식으로 실습을 진행하기로 한다. LED 이외에도 다양한 소자를 이용하여 포트 제어관련 실습이 가능하며 다음은 본 장에서 포트제어 실습을 위해 사용할 부품들에 대한 소개이다.

[1] LED(Light Emitting Diode)

LED는 두 단자 형태의 반도체 발광소자로서 적정한 전류를 단자에 인가하게 되면 빛을 발생하는 p-n접합(p-n junction)형 다이오드이다. 전류가 단자를 흐를 때 소자내부에서 전자들(electrons)이 정공들(holes)과 재결합을 하게 되면서 광자(photons) 형태로 에너지를 방출하게 된다. LED가 방출하는 색깔은 광자에너지(photon energy)와 연관되어 있으며 반도체가 가지고 있는 에너지 밴드갭(band gap)에 의해서 결정된다. 일반적으로 LED 자체는 $1mm^2$이하의 작은 크기를 가지고 있으며 빛을 방사하는 패턴을 결정하기 위한 광학 요소(optical components)와 결합되어 진다.

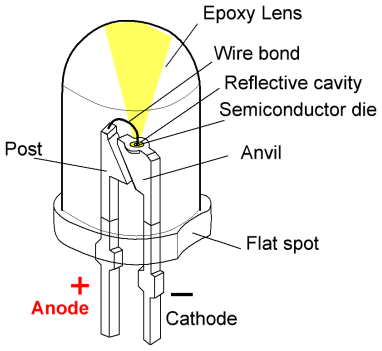
초기에는 LED를 단순히 기존의 소형 전구가 하던 전자장치의 표시기 역할을 대치하는 정도로 사용하는 수준이었다. 그러나 그 이후에 7-세그먼트 등과 같은 숫자 표시 형태로 발전하여 디지털시계 등에 널리 활용되기 시작하였다. 오늘날에 이르러서는 디스플레이 장치를 비롯한 각종 센서 등에 유용하게 쓰이면서 그 활용 범위를 넓혀가고 있는 중

이다.

다양한 색깔을 발생하는 LED의 구성이 가능하며 이는 반도체 소자 제조 시에 AlGaAs, GaAsP, ZnSe 등의 화합물을 다양하게 조합함으로써 가능하게 된다. 이렇게 제조된 LED 소자는 화합물의 종류에 따라 전압 인가 시에 다른 파장의 빛을 발생하게 된다.

LED는 구조상 극성을 가지고 있기 때문에 사용 시에 주의해야한다. <그림 3-1>, <그림 3-2>는 다양한 LED와 LED 내부 구조를 보인 것이다.

<그림 3-1> 다양한 LED



<그림3-2> LED의 내부 구조

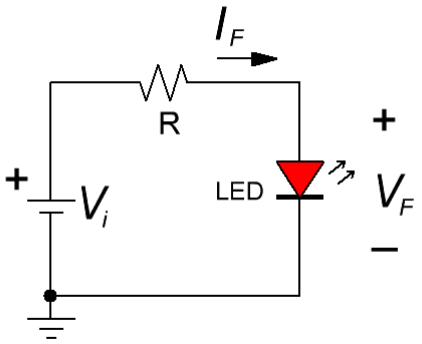
LED는 비록 회로 상에서 다이오드 모양으로 표현되지만 순방향 전압강하(forward voltage drop) 값은 일반적인 다이오드와 다르다는 사실에 유의해야 한다. 또한 LED의 순방향 전압은 컬러(color)에 따라서도 차이가 존재한다. <표 3-1>은 LED의 색깔에 따른 순방향 전압강하(forward voltage drop) 값의 범위를 나타낸 것이다.

<표 3-1> LED 색깔별 순방향 전압강하 범위

Color of LED	Forward Voltage Drop[V]
Red(빨강)	1.63 ~ 2.03
Yellow(노랑)	2.10 ~ 2.18
Orange(오렌지)	2.03 ~ 2.10
Blue(파랑)	2.48 ~ 3.7
Green(녹색)	1.9 ~ 4.0
Violet(보라색)	2.76 ~ 4.0
UV(자외선 방출 LED)	3.1 ~ 4.4
White(흰색)	3.2 ~ 3.6

ABSOLUTE MAXIMUM RATINGS (T_{amb} = 25 °C, unless otherwise specified) **VLHW5100**				
PARAMETER	**TEST CONDITION**	**SYMBOL**	**VALUE**	**UNIT**
Reverse voltage		V_R	5	V
DC forward current		I_F	30	mA
Peak forward current	at 1 kHz, t_p/T = 0.1	I_{FSM}	0.1	A
Power dissipation		P_V	100	mW
Zener reverse current		I_Z	100	mA
Junction temperature		T_j	100	°C
Operating temperature range		T_{amb}	- 40 to + 100	°C
Storage temperature range		T_{stg}	- 40 to + 100	°C
Soldering temperature	t ≤ 5 s	T_{sd}	260	°C
Thermal resistance junction/ambient		R_{thJA}	400	K/W

<그림 3-3> White LED Datasheet



<그림 3-4> LED 회로 구성

<그림 3-3>은 백색 LED에 대한 데이터시트의 일부분이다. 표시된 바와 같이 최대 전류 정격(maximum current rating)에 따라 <그림 3-4>와 같은 회로에서 최소한의 저항의 크기를 구하면 다음과 같다.

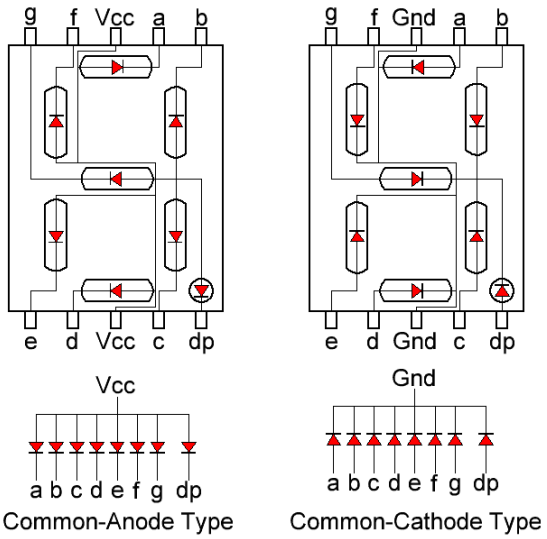
$$R = \frac{V}{I} = \frac{V_i - V_F}{I_{F_{\max}}} = \frac{5\,V - 3.2\,V}{30mA} = 60\Omega$$

하지만 데이터시트에 표기된 정격 전류는 최대로 흘릴 수 있는 정격으로 일반적으로는 이보다 훨씬 적은 전류로도 원하는 밝기를 얻을 수 있으므로 소모 전력을 낮추기 위해서 적절한 저항 값을 선정하는 것이 바람직하다.

[2] 7-세그먼트(7-Segment)

7-세그먼트는 일명 SSD(seven segment display)라고 불리는 소자로써 십진 숫자(decimal numerals)를 표시하기 위한 전자 디스플레이 소자의 한 형태이며 수를 표현하기 위한 a~g의 7개 단자들과 점을 표현하기 위한 DP(decimal point) 단자를 가지고 있다. 오늘날 7-세그먼트는 디지털시계를 비롯한 숫자정보를 취급하는 각종 전자 장치에 널리 활용되고 있다.

<그림 3-5>은 7-세그먼트의 내부 구조를 보여주고 있다. 7-세그먼트는 내부 LED의 극성에 따라 공통 애노드(common anode) 형식과 공통 캐소드(common cathode) 형식의 두 가지로 나눌 수 있다. 본 교재에서는 주로 공통 캐소드(common cathode)형을 사용하였으므로 <표 3-2>에서 제시한 논리 값을 바탕으로 제어를 해야 한다. 앞서 언급했듯이 PIC 마이컴의 경우 특수한 포트 구조로 인하여 출력전류가 LED를 직접 구동할 수 있을 정도로 충분히 크다. 그러나 7-세그먼트처럼 내부적으로 다수의 LED를 가진 소자의 경우에는 포트에서 제공되는 출력 전류만으로는 충분한 밝기를 얻기가 어려울 수 있다. 이러한 경우 트랜지스터(transistor) 등을 활용한 전류증폭 회로를 사용해야 한다.



<그림 3-5> 7-Segment의 2가지 종류

<표 3-2> 공통 캐소드(common cathode) 7-Segment 숫자 표

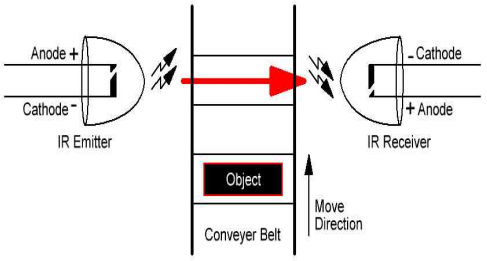
숫자	a	b	c	d	e	f	g	2진 값	16진 값
0	1	1	1	1	1	1	0	0111111	3F
1	0	1	1	0	0	0	0	0000110	06
2	1	1	0	1	1	0	1	1011011	5B
3	1	1	1	1	0	0	1	1001111	4F
4	0	1	1	0	1	1	1	1100110	66
5	1	0	1	1	0	1	1	1101101	6D
6	0	0	1	1	1	1	1	1111100	7C
7	1	1	1	0	0	1	0	0100111	27
8	1	1	1	1	1	1	1	1111111	7F
9	1	1	1	0	0	1	1	1100111	67

[3] 적외선 센서(Infrared Ray Sensor)

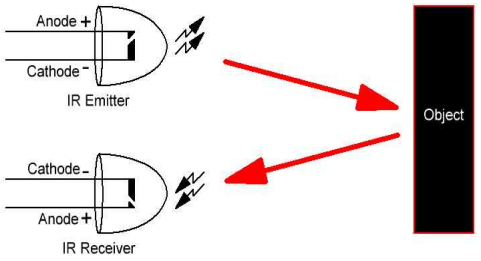
적외선 센서는 LED의 구조를 이용한 센서 중의 하나로 발광소자(infrared emitting diode)와 수광소자(infrared ray receiver) 두 가지로 구성된다. 발광소자는 앞서 설명한 LED와 같은 원리의 소자로써 빛의 파장이 700nm ~ 1mm 사이인 빛(적외선)을 방사하는 소자이다. 적외선파는 인간의 시각으로는 볼 수 없으며 전자기 스펙트럼 상에서 가시광선과 마이크로파 영역 사이에 위치한다. 전자기파의 파장은 대략 0.75에서 1000μm 사이에 해당한다. 파장의 범위가 0.75~3μm에 속하는 경우를 근적외선(near-infrared)이라 하고 3~6μm에 속하는 경우에는 이를 중적외선(mid-infrared)이라 한다. 그리고 6μm 이상의 파장을 가지는 경우를 원적외선(far-infrared)이라 부른다.

TV의 리모컨에서와 같이 적외선 기술은 우리 생활 곳곳에서 유용하게 사용되고 있다. 적외선 기술이 이처럼 널리 사용되는 이유는 소비 전력이 적고 회로의 구성이 용이하다는 장점을 가지고 있기 때문이다.

<그림 3-6>과 <그림 3-7>은 적외선 센서를 활용하는 대표적인 두 가지 경우를 그림으로 표현한 것이다. <그림 3-6>의 응용은 발광소자와 수광소자가 서로 마주보고 있는 형태로 평상시에 수광부 회로에서 전류의 흐름이 이루어진다. 그러다가 물체가 이들 사이를 지나면서 적외선파가 순간적으로 차단되는 경우에 이를 감지할 수 있게 된다. <그림 3-7>은 발광소자와 수광소자를 나란히 배열하여 사용하는 형태로 평상시에는 수신부에 전류가 흐르지 않게 된다. 그러다가 물체가 가까이 근접하게 되면 발광부에서 방사한 적외선 신호가 반사되어 수신 감지됨으로써 수신부에서 전류가 흐르게 된다.

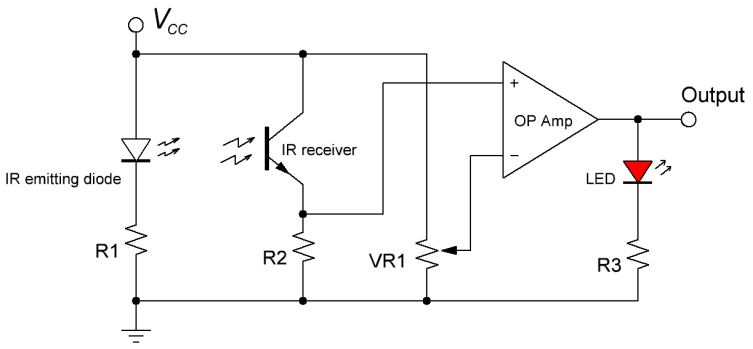


<그림 3-6> Break Beam Sensor



<그림 3-7> Reflectance Sensor

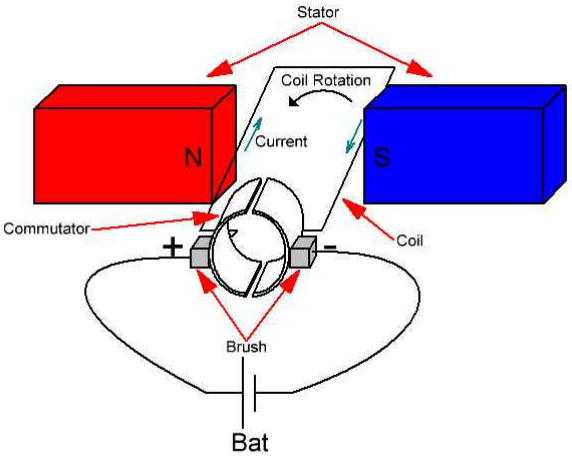
<그림 3-8>는 적외선 센서를 구동하는 드라이버 회로의 한 예를 보인 것이다. 적외선 발광소자에서 방사되는 신호가 수신 센서부에 수신될 경우 연산증폭기(operational amplifier)의 출력이 HIGH가 되도록 하는 회로이다. 수신감도 조절은 연산증폭기의 음(-)의 단자에 연결된 가변저항을 조정함으로써 가능하다.



<그림 3-8> 적외선 센서 구동 회로

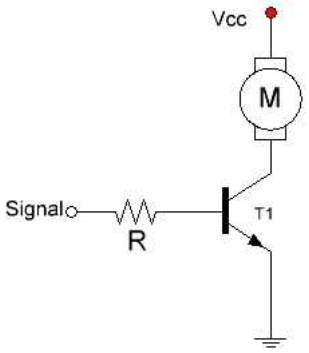
[4] 직류 전동기(DC Motor)

직류 전동기(direct current Motor) 혹은 일반적으로 DC 모터라고 불리며 영구자석으로 구성되는 고정자(stator)와 코일로 이루어진 회전자(rotator)로 구성된 모터이다. 회전자에 전류가 흐름으로써 형성되는 자장이 고정자가 형성하는 자장의 영향을 받아서 회전력을 발생하게 되는 원리를 이용한 전동기이다. 회전자에 흐르는 전류의 방향전환은 브러시(brush)라고 하는 기계장치를 통하여 이루어진다. 그러나 브러시의 마모로 인하여 내구성이 약하다는 단점을 가지고 있다. 그럼에도 불구하고 제어가 용이하고, 제어용 모터로서 우수한 특성을 가지고 있어서, 모형 자동차, 무선조정용 장난감 등을 비롯한 여러 방면에서 사용되고 있다. <그림 3-9>는 직류 전동기의 구조와 동작을 간략히 나타낸 그림이다.



<그림 3-9> 직류 전동기의 구조

일반적으로 MCU의 입·출력 포트에서 나오는 전류 신호의 크기는 20mA 근처이다. 이러한 작은 전류로 DC 모터를 직접 구동하기에는 무리가 있다. 따라서 <그림 3-10>에서 보인 바와 같은 전류증폭 방법을 사용해야 한다. 트랜지스터의 동작은 포화영역(saturation region), 활성영역(active region), 차단영역(cut-off region)으로 나누어진다. 트랜지스터를 증폭기로 활용할 때에는 활성영역에서 동작시키게 된다. 그에 반해서 <그림 3-10>에서와 같이 트랜지스터를 스위치로 작동시키기 위해서는 포화영역과 차단영역을 사용해야 한다.



<그림 3-10> 직류 전동기 단방향 구동 회로

3.2 입·출력 포트 (I/O Ports)

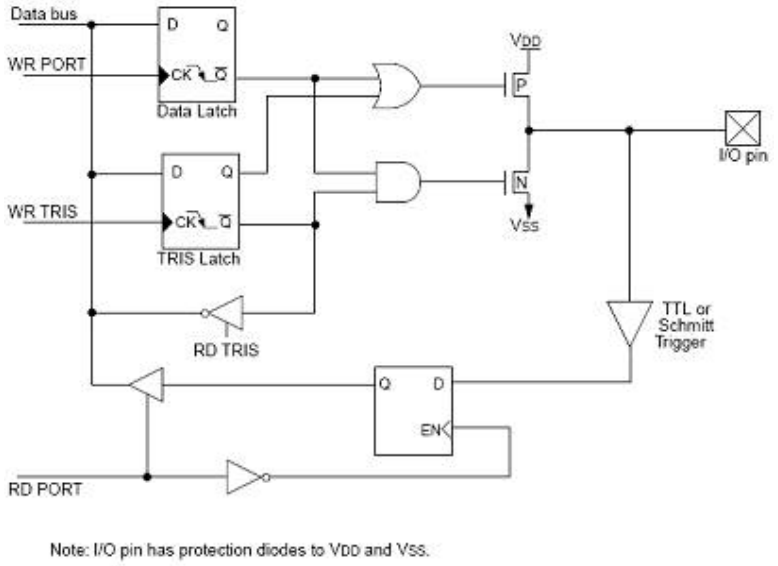
PIC 마이컴의 포트 입·출력제어 실습에 들어가기에 앞서 본 교재에서 사용하는 마이컴 PIC16F628A의 입·출력 포트의 기본적인 구조에 대하여 설명하기로 한다.

[1] 일반적인 개요

입·출력 핀은 일반적으로 외부 장치를 제어하기 위한 목적으로 사용되며 이들 중 일부는 특수 기능을 수행하기 위한 목적으로도 혼용하여 쓰이기도 한다. 각 디바이스들이 가지는 입·출력 핀들의 특수 기능은 데이터시트(datasheet)를 참조하면 된다. 일반적으로 특정 핀을 특수 기능 목적으로 사용하는 경우에는 제어용 입·출력 용도로 사용하는 것이 제한된다.

대부분의 핀들은 TRIS레지스터라 불리는 데이터 직접 레지스터에 의해 제어된다. TRISx 레지스터는 PORTx를 직접 제어하며 레지스터가 0으로 설정되면 출력으로, 1로 설정되면 입력으로 활용된다.

일반적인 I/O의 포트 구조를 아래의 <그림 3-11>에 나타내었다.



<그림 3-11> 일반적인 I/O 포트 구조

(1) 구조 (Structure)

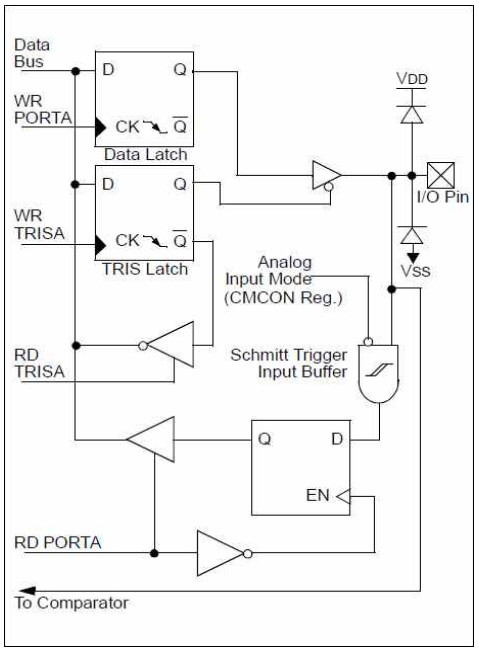
■ PORTA와 TRISA 레지스터

8핀으로 구성된 포트 A(RA)의 RA5 핀은 일반적으로 Master Clear(Reset)를 위한 입력 핀으로 사용하며 RA6, RA7 핀은 크리스탈 오실레이터를 사용하기 위한 입·출력 핀으로 사용한다.

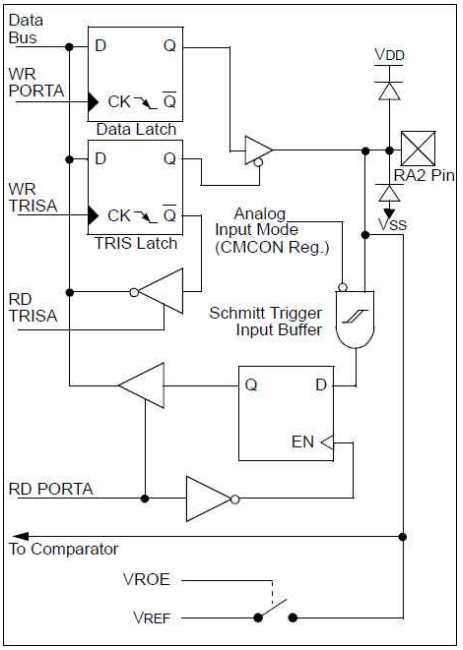
포트 A의 대다수의 핀들은 TTL 입력 레벨과 CMOS 출력 구동 구조를 가지고 있으며 이들은 TRIS 레지스터의 설정으로 입·출력이 결정한다. 그러나 이들과 달리 RA4 핀은 슈미트 트리거 입력과 오픈드레인 출력 구조를 가지고 있다. 따라서 RA4 핀을 출력으로 사용할 경우에는 외부에 풀업 저항을 연결해주어야 한다.

TRISA 레지스터를 set(1) 하면 입력을 받아들이기 위하여 High-Impedance 모드로 바뀌고 clear(0) 하면 출력 래치(latch)가 선택된다.

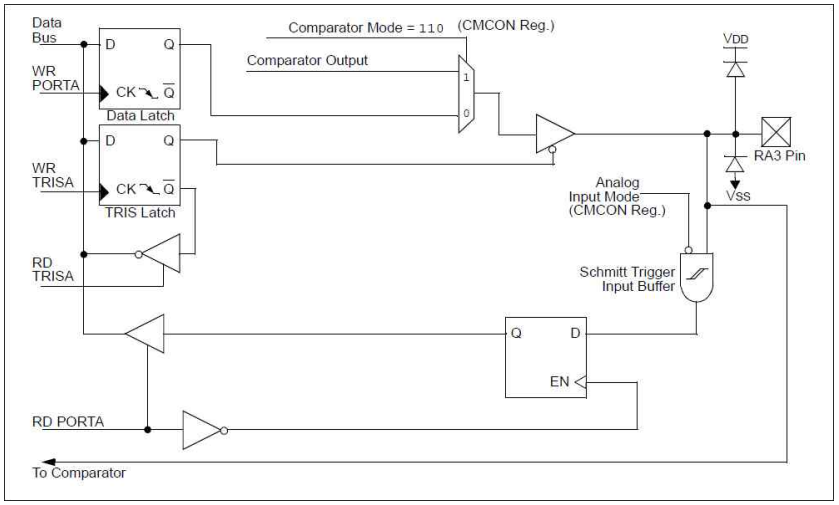
각 핀마다 특수한 기능을 가지고 있는 경우가 있는데, 이에 대해서는 후에 각 기능을 다루는 부분에서 다루기로 한다.



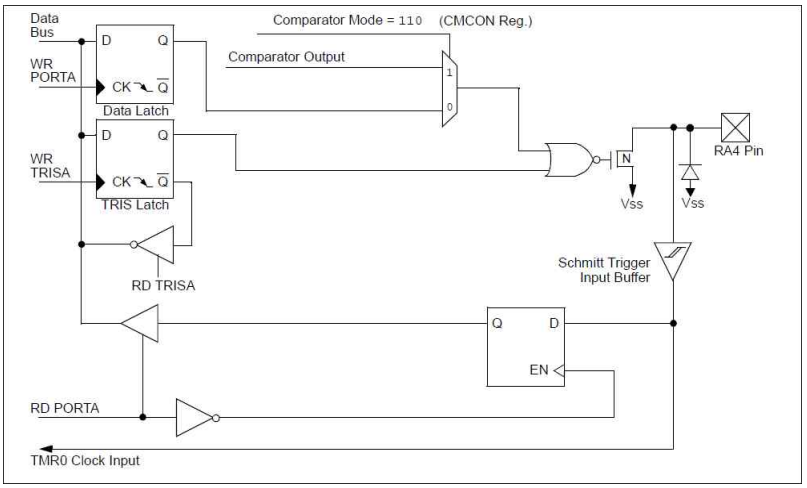
<그림 3-12> RA0:RA1의 블록 다이어그램



<그림 3-13> RA2 핀의 블록 다이어그램



<그림 3-14> RA3 핀의 블록 다이어그램

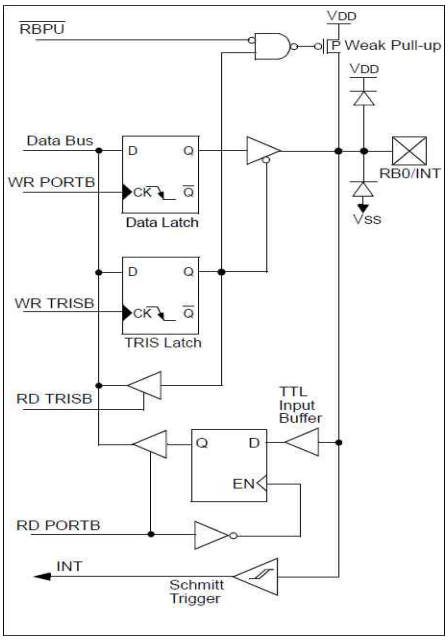


<그림 3-15> RA4 핀의 블록 다이어그램

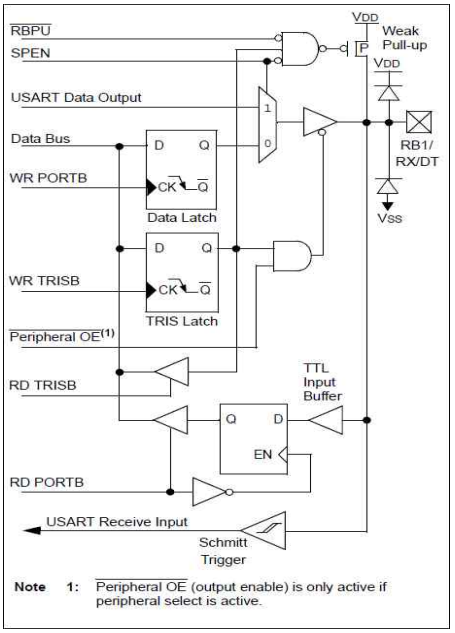
■ PORTB와 TRISB 레지스터

포트 B(RB)는 8-bit로 양방향 입·출력이 가능하고 TRISB에 의해 제어된다. TRISB 레지스터를 set(1) 하면 입력을 받아들이기 위한 High-Impedance 모드로 바뀌고 clear(0) 하면 출력 래치가 선택된다.

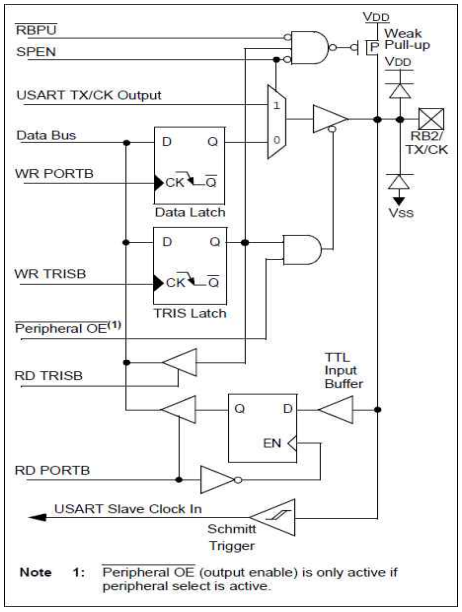
포트 B는 내부적으로 풀업이 되어 있으며 레지스터를 활용하여 풀업의 동작 상태를 제어할 수 있다. 포트 B를 출력으로 사용할 경우 자동으로 풀업은 사라지며 Power-on Reset의 경우에도 활성화 되지 않는다.



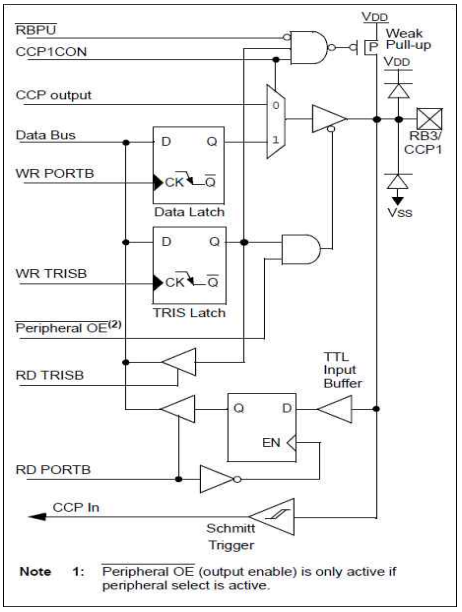
<그림 3-16> RB0 핀의 블록 다이어그램



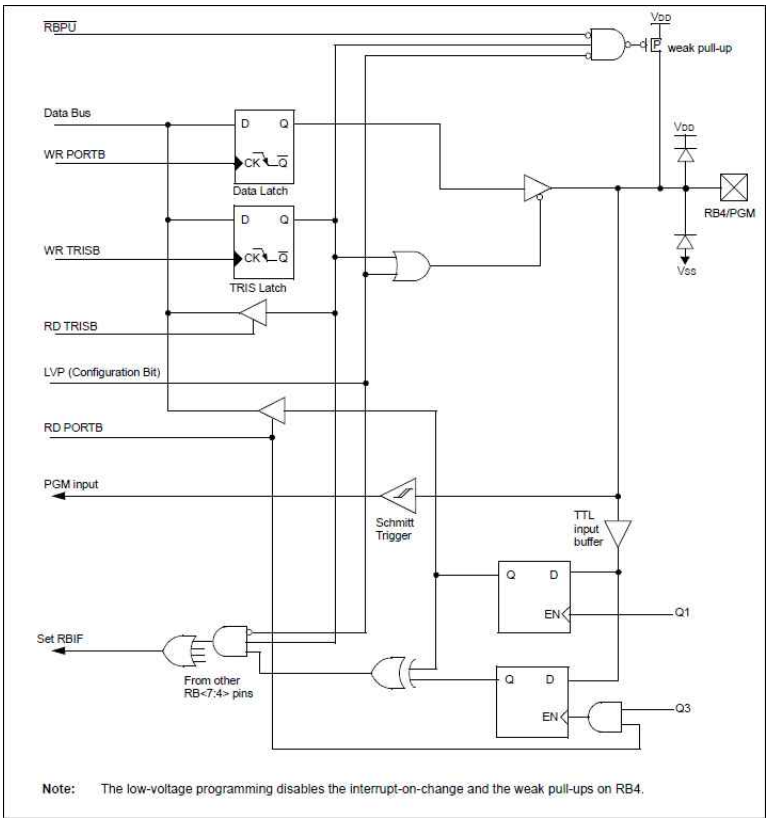
<그림 3-17> RB1 핀의 블록 다이어그램



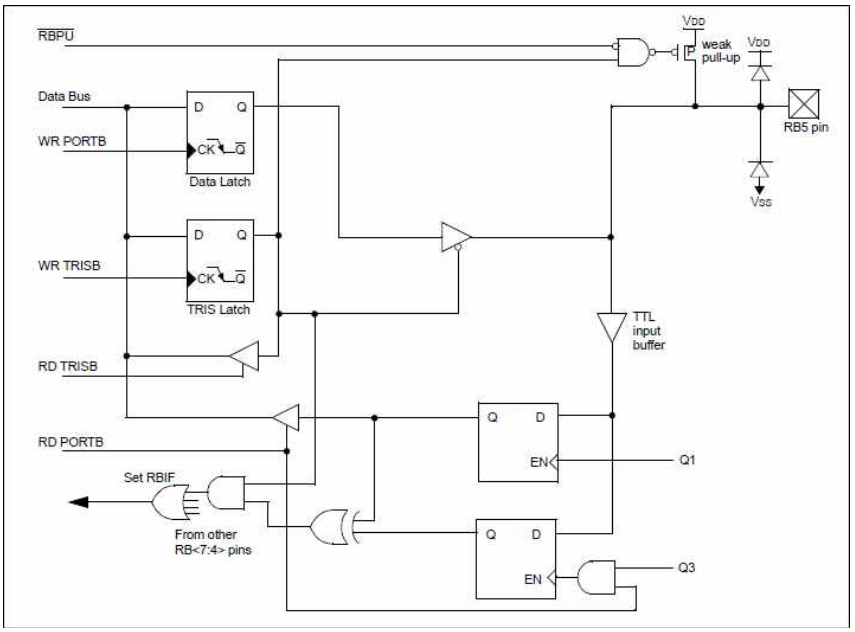
<그림 3-18> RB2 핀의 블록 다이어그램



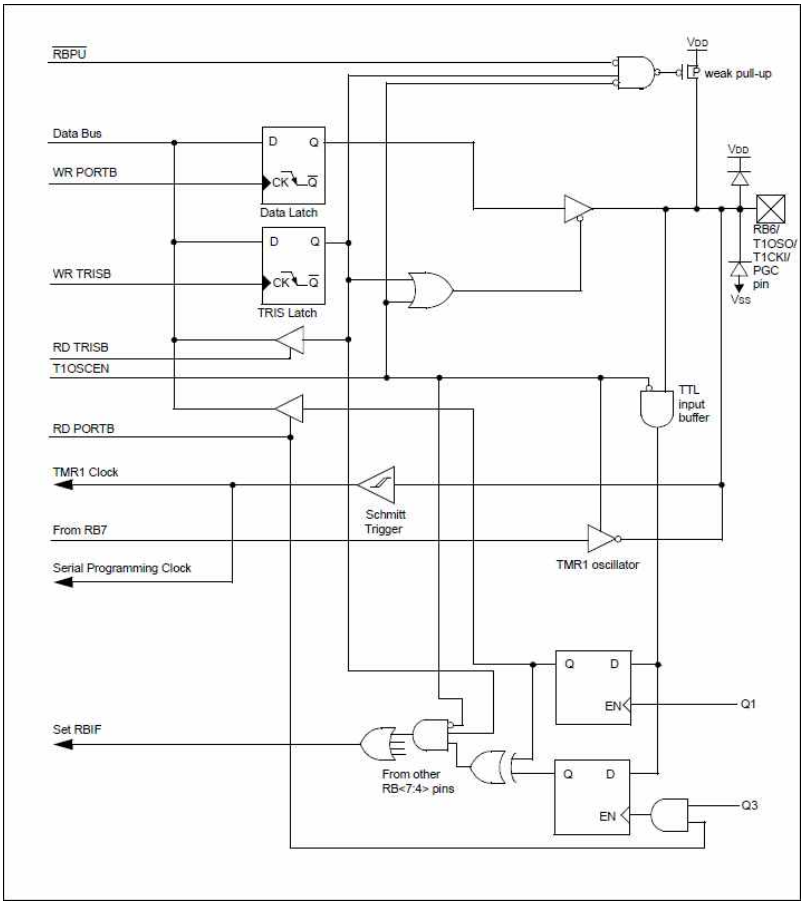
<그림 3-19> RB3 핀의 블록 다이어그램



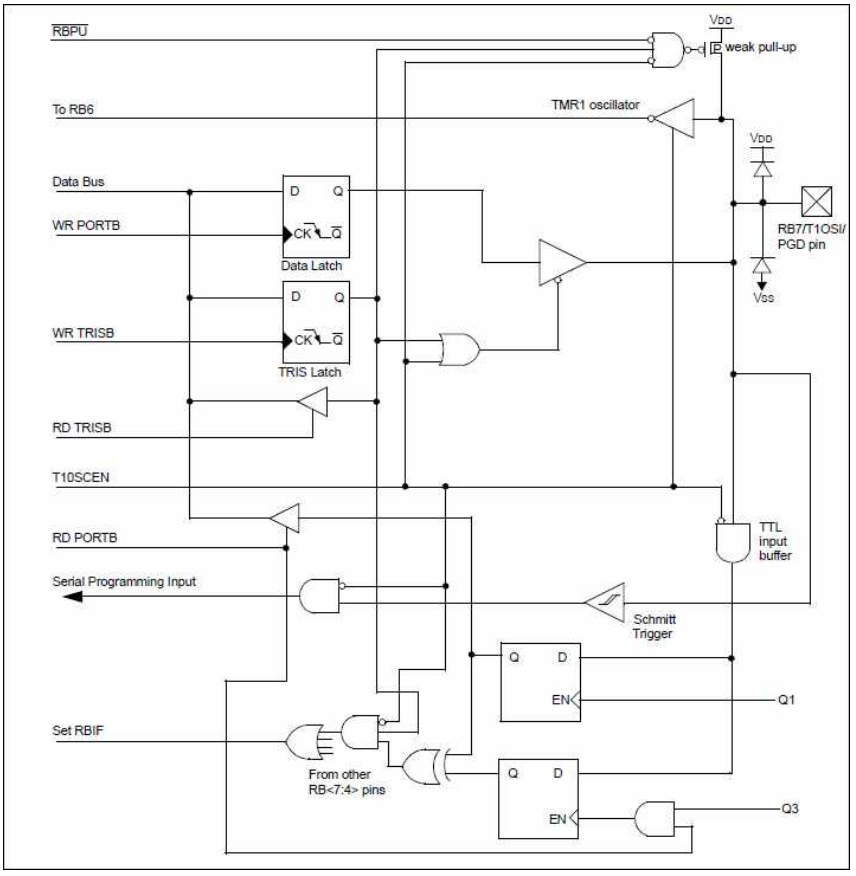
<그림 3-20> RB4 핀의 블록 다이어그램



<그림 3-21> RB5 핀의 블록 다이어그램



<그림 3-22> RB6 핀의 블록 다이어그램



<그림 3-23> RB7 핀의 블록 다이어그램

포트 B(RB) 역시 포트 A(RA)와 마찬가지로 각 핀마다 특수한 기능을 가지고 있는 경우가 있는데, 이에 대해서는 후에 각 기능을 다루는 장에서 언급하기로 한다.

3.3 포트 출력 제어

같은 동작을 하는 프로그램이라 하더라도 그 구성은 개발자에 따라 서로 다른 형태를 가질 수 있다. 그러나 효율적인 프로그램을 위해서는 정해진 조건에 부합하는 최적의 구성이 무엇보다도 중요하다. 더욱이 사용하고자 하는 마이크로프로세서가 기능면이나 메모리의 크기 면에 있어서 많은 제약이 있는 경우에는 더욱 더 프로그램의 효율성에 신중을 기해야 한다. 사용하는 컴파일러에 대한 이해는 효율적인 프로그램 구성에 있어서 매우 중요한 영향을 미치므로 사전에 공부를 하여 두는 것이 바람직 할 것이다.

이 절에서는 같은 기능을 하는 프로그램을 3가지 형식으로 구성하여 보았다. 이들 각각에 대하여 프로그램 구성의 효율성을 염두에 두고 면밀히 관찰하면서 연습해 보기 바란다. 그러면 추후에 복잡한 구조의 시스템을 효율적으로 프로그래밍 하고자 할 때 많은 도움이 될 것이다.

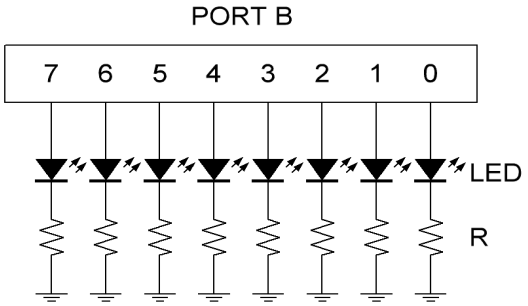
이제 PIC 마이컴의 입·출력 제어에 가장 기초가 되는 포트 출력 방법에 대하여 LED를 활용하여 연습해보기로 하자.

[1] 실습 1 - LED 제어

이번 실습의 목적은 포트 B에 8개의 LED를 연결하고서 점등(ON)되는 LED가 LSB(Least Significant Bit)에서부터 MSB(Most Significant Bit)까지 순차적으로 이동하도록 하는 것이다. 단, 점등되는 LED의 이동은 매 0.5초(500ms) 간격으로 이루어지도록 한다. 그리고 이러한 일련의 동작이 무한 반복되도록 구성한다.

(1) 회로 구성

다음은 본 실습을 위한 회로 구성이다. PIC16F628A 포트 B의 8개 단자(B.0 ~ B.7)에 <그림 3-24>에 보인 바와 같이 LED를 각각 연결한다. 저항 값은 사용한 LED의 데이터시트를 참조하여 선택 사용하면 된다. 참고적으로 본 실험에서는 구동저항으로 470 Ω을 사용하였으며 만일 고휘도 LED를 사용하는 경우에는 이 보다 더 큰 저항 값을 사용해도 될 것이다.



<그림 3-24> 회로 구성

(2) 프로그램

■ [EX 1]

다음은 가장 단순한 형태의 프로그램으로 컴파일러 CCS-C의 기본적인 함수만을 사용하여 작성한 경우이다.

```
#include <16f628a.h>    //include device header file
#fuses XT, NOWDT, PUT, NOPROTECT
#use    delay(clock=4000000)    //oscillator assignment

#byte porta = 0x05       //port A aiiasing
#byte portb = 0x06       //port B aiiasing

void main(void)
{
        set_tris_b(0x00); // setting port B as output
        portb=0x00;      // initial value for port B ( All LED off)

        while(1) //infinite loop
        {
                portb=0x01; // port B output : (MSB) 00000001 (LSB)
                delay_ms(500);
                portb=0x02;      //(MSB) 00000010 (LSB)
                delay_ms(500);
                portb=0x04;      //(MSB) 00000100 (LSB)
                delay_ms(500);
                portb=0x08;      //(MSB) 00001000 (LSB)
```

```
                delay_ms(500);
                portb=0x10;     //(MSB) 00010000 (LSB)
                delay_ms(500);
                portb=0x20;     //(MSB) 00100000 (LSB)
                delay_ms(500);
                portb=0x40;     //(MSB) 01000000 (LSB)
                delay_ms(500);
                portb=0x80;     //(MSB) 10000000 (LSB)
                delay_ms(500);
        }
}
```

■ [EX 2]

다음은 앞의 경우에 비해 보다 간결한 형태의 프로그램으로 Standard C 언어에서 사용되는 함수를 사용한 경우이다. (초기 값이 다름에 주의)

```
#include <16f628a.h> //device header file include
#fuses XT, NOWDT, PUT, NOPROTECT
#use delay(clock=4000000)

#byte porta = 0x05 //port A aliasing
#byte portb = 0x06 //port B aliasing

void main(void)
{
        int i=0;  //define variable and assign the initial value
        set_tris_b(0x00); //setting port B as all output
        portb=0x01; //output port B with initial value

        while(1) //infinite loop
        {
                delay_ms(500);  //time delay 500ms
                portb=portb<<1; //shift left to MSB side
                i++;            //increase the variable
```

```
                if(i==8)  //when it's MSB, back to initial state
                {
                        i=0;      //initialize the variable
                        portb=0x01; //output port B with initial value
                }
        }
}
```

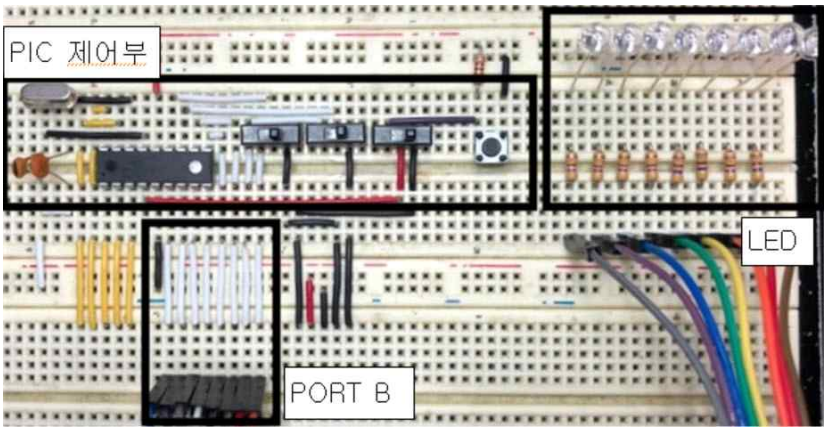
■ [EX 3]

다음 프로그램은 CCS-C가 제공하는 고유의 내장함수(Built-in function)인 rotate_left()를 사용하여 보다 효율적으로 구성한 경우이다.

```
#include <16f628a.h> //include device header file
#fuses XT, NOWDT, PUT, NOPROTECT
#use delay(clock=4000000) //oscillator define
#byte porta = 0x05 //port A aliasing
#byte portb = 0x06 //port B aliasing

void main(void)
{
        set_tris_b(0x00); //setting all pin of port B as output
        portb=0x01; // output port B with initial value

        while(1)   //infinite loop
        {
                delay_ms(500); //time delay 500mS
                rotate_left(&portb,1); //shift left with value of port B
        }
}
```

(3) 실습 결과

<그림 3-25>는 실습에 사용한 실제 회로 구성을 보인 것이다. 본 교재에서 사용하는 다운로더(downloader) 장치는 PicKit3이다. 컴파일된 헥사(Hexa)파일을 다운로드 하기 위한 PIC 마이컴 주변회로 구성은 1장에서 이미 소개한 바 있으니 참조하기 바란다.

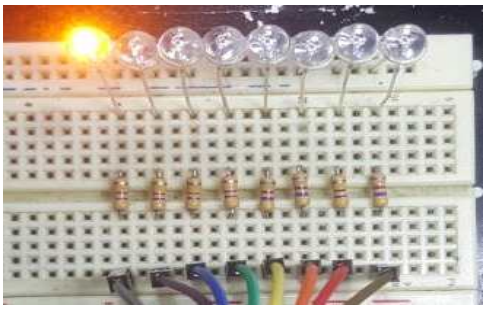
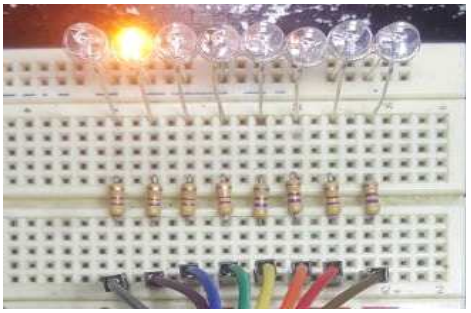
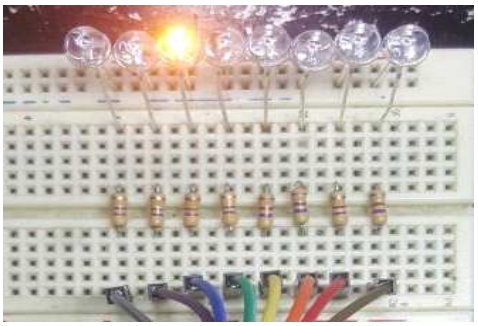
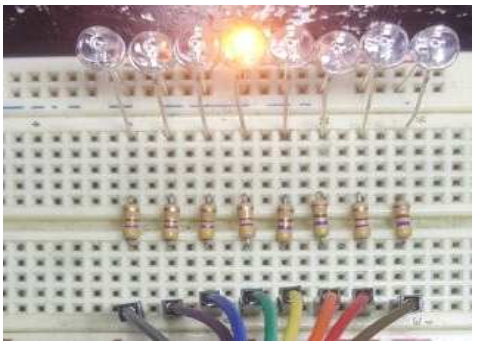
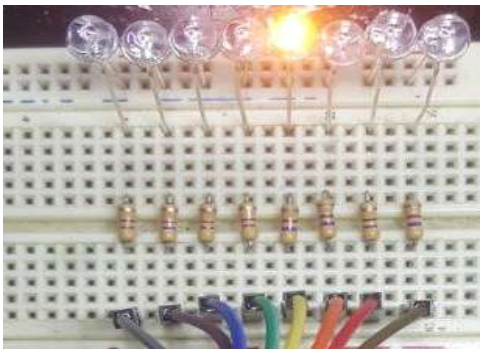
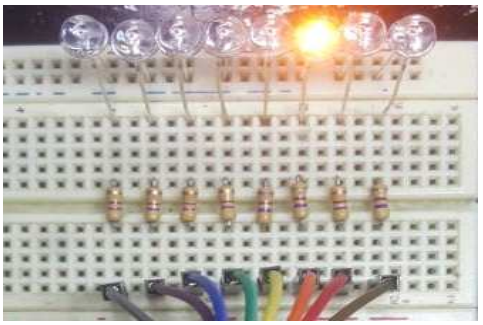
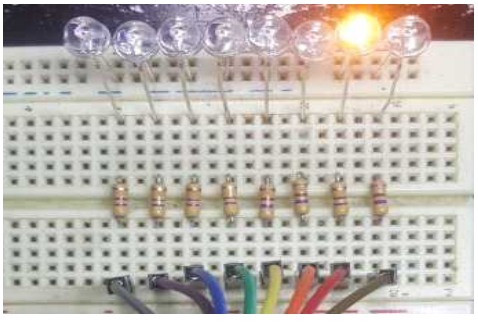
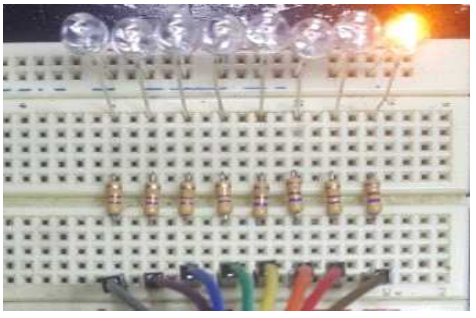


<그림 3-25> 실제 회로 구성

<그림 3-26>은 실제 실험을 수행했을 때 B.0핀과 B.1핀에서 출력되는 신호를 오실로스코프를 통해 측정한 결과이다. 파형을 보면 주기는 LED 8개가 순차적으로 켜지는 전체 시간인 4초에 해당하며 포트 B의 각 핀은 500ms동안 HIGH 신호를 발생하는 것을 알 수 있다. 전체 동작과정은 <표 3-3>을 통하여 확인할 수 있다.

<그림 3-26> 실습 결과 파형

<표 3-3> 실습 결과

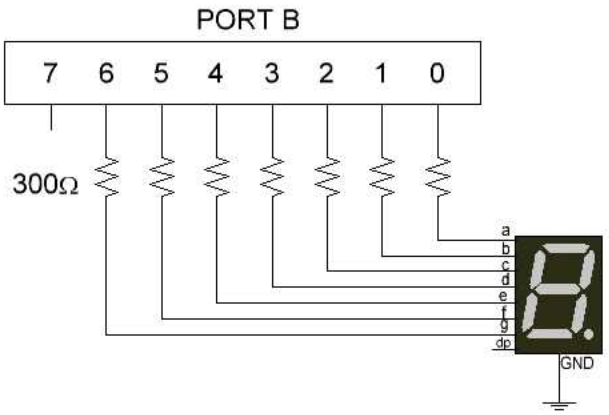









[2] 실습 2 - 7-세그먼트(7-Segment) 제어

본 실습의 목적은 포트 B에 7-세그먼트를 연결하고서 7-세그먼트에 표시되는 숫자가 0부터 9까지 순차적으로 증가하도록 하는 것이다. 단, 숫자의 변화는 1초 단위로 발생하도록 한다. 7-세그먼트의 숫자가 9에 도달하면 그 1초 후에는 0부터 다시 시작하는 형식으로 구성하고 이를 무한히 반복하도록 구성한다.

(1) 회로 구성

다음은 본 실습을 위한 회로 구성이다. 포트 B의 7개 단자 즉, B.0에서 B.6까지를 사용하여 아래 <그림 3-27>에 보인 바와 같이 7-Segment(Common Cathode Type)의 A~G 단자를 각각 순서대로 연결한다. 7-세그먼트의 GND 단자는 그라운드에 연결하되 DP(Decimal Point)단자는 연결하지 않는다.



<그림 3-27> 회로 구성

(2) 프로그램

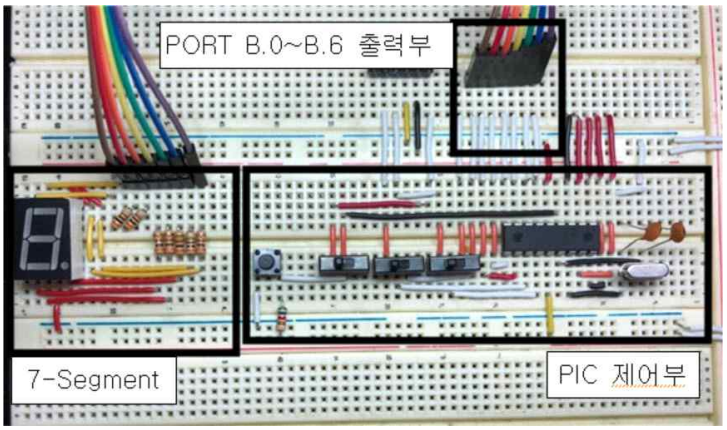
다음은 <표 3-2>를 바탕으로 각 숫자에 해당하는 데이터를 배열(array)로 만들어 7-세그먼트에 0부터 9까지의 숫자가 1초 단위로 증가하도록 구성하고 이를 무한히 반복하도록 한 코딩이다.

```
#include <16f628a.h> //include device header file
#fuses XT, NOWDT, PUT, NOPROTECT
#use delay(clock=4000000) //oscillator setting
```

```
#byte porta = 0x05 //port A aliasing
#byte portb = 0x06 //port B aliasing

//Array for the 7-segment display
byte digit[10]={0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7C,0x27,0x7F,0x67};

void main(void)
{
        int i=0;  //define variable and assign the initial value
        set_tris_b(0x00); //setting all pins of port B as output
        portb=0x00; //output port B with initial value

        while(1) //infinite loop
        {
                for(i=0 ; i<10 ; i++) //setting interval range as 1 ~ 9
                {
                  portb=digit[i]; //out the proper value according to I
                  delay_ms(1000); //stay the present state for 1 sec.
                }
        }
}
```

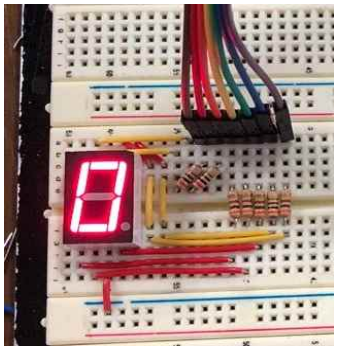
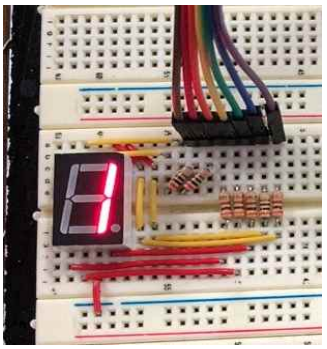
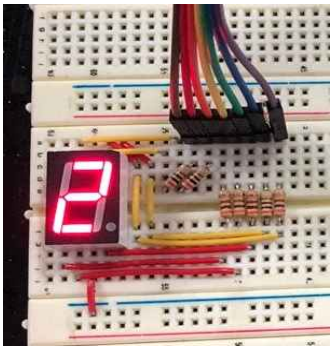
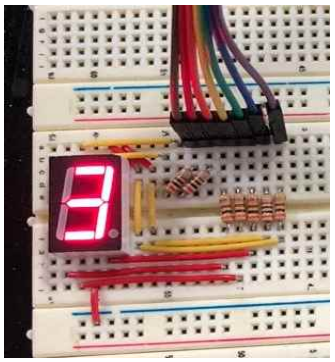
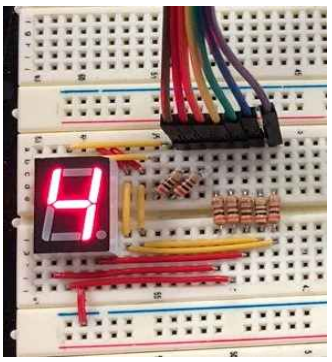
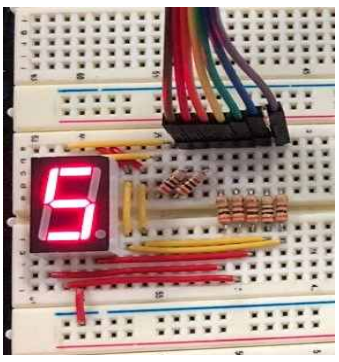
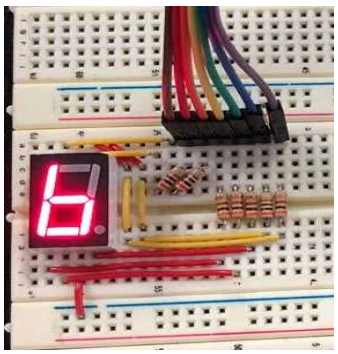
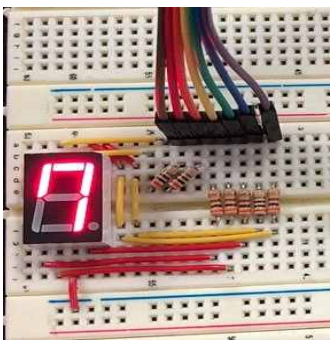
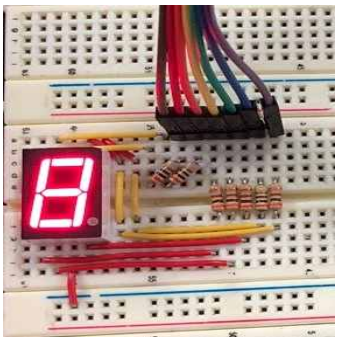
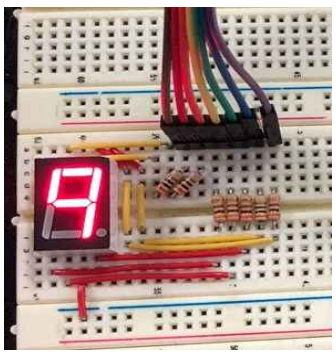
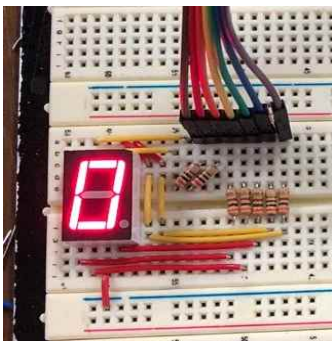
(3) 실습 결과

아래에 보인 바와 같이 실제 실습을 진행하였으며 그 결과를 <표 3-4>에 나타내었다.



<그림 3-28> 실제 회로 구성

<표 3-4> 실습 결과

		
		
		
		이후 무한 반복

3.4. 포트 입·출력 제어

평상시에 정해진 일련의 동작을 하던 마이크로프로세서가 특정 입력 포트에서 발생하는 상태 변화 즉 이벤트(event)를 인식하고 이에 대한 처리를 우선적으로 수행해야 하는 경우가 종종 발생한다. 이처럼 프로세서가 외부의 이벤트에 반응하는 방식에는 폴링(Polling)방식과 인터럽트(Interrupt)방식이 있다.

폴링방식은 외부에서 들어오는 이벤트의 상태를 확인하기 위하여 정해진 시간 혹은 순번에 맞추어 주기적으로 점검을 하는 방식이다. 그에 비해 인터럽트방식은 main문을 수행하는 도중에 외부로부터 특정 인터럽트 핀에 신호가 들어오면 프로세서는 현재하고 있는 업무를 즉시 멈추고 해당 인터럽트의 서비스 루틴을 수행하는 방식이다. 그리고 인터럽트 서비스 루틴이 끝나면 다시 원래하던 작업을 이어서 수행하게 된다.

전체적으로 동작이 단순한 시스템의 경우에는 비교적 단순한 구조를 가지는 폴링방식이 유용하게 쓰일 수 있다. 그러나 체크해야 할 포트가 많거나 여러 작업을 동시에 수행해야 하는 시스템에 있어서 폴링방식은 다른 작업의 수행에 영향을 미치거나 프로세서에 부담을 주게 되므로 효율적인 작업 수행이 어렵게 된다. 또한 폴링방식은 순서대로 일을 처리하기 때문에 비상상태나 급하게 처리해야 할 업무에 대해 신속한 대응이 어렵다는 문제점을 가지고 있다. 따라서 다소 복잡하고 다양한 업무를 처리해야 하는 시스템에서는 외부 이벤트에 대하여 보다 신속하고 효율적으로 업무를 처리해야 할 필요성이 요구된다. 이러한 요구사항을 만족시키기 위한 방안이 바로 인터럽트를 활용하는 방식이다. 인터럽트는 프로세서에 우선순위를 정해 놓고 우선순위가 높은 일이 발생하면 현재 작업을 즉시 중단하고 우선순위에 따른 업무를 효율적으로 처리한 다음 원래하던 작업을 계속해서 수행할 수 있는 방식이다.

이번 장에서는 외부 이벤트에 대한 처리 방식 중에서 비교적 간단한 방식인 폴링(polling) 방식에 대하여 연습해 보기로 한다. 인터럽트(interrupt) 방식에 대한 실습은 다음 장(4장)에서 다룰 예정이다.

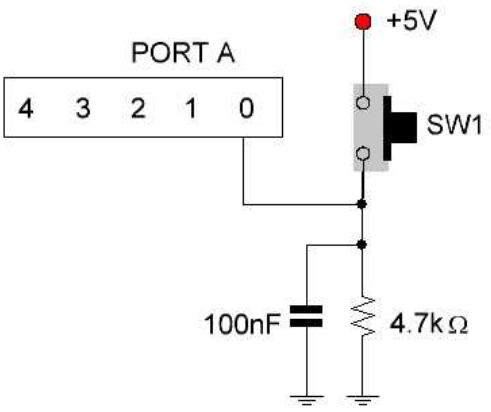
[1] 실습 1 - LED 제어

본 실습의 목적은 포트 A의 A.0핀을 입력으로 하여 이 핀의 상태에 따라서 포트 B에 연결된 LED들이 다른 동작을 하도록 하는 것이다. 본 실습의 동작 원리를 정리하면 다음과 같다.

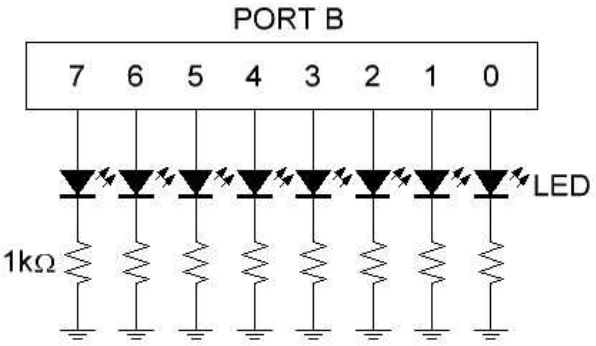
① 처음에 전원을 인가하면 A.0핀의 입력상태가 LOW(0)이고 포트 B에 연결된 LED들이 꺼진 상태를 유지한다.
② 버튼(button) SW1을 누르고 있으면 A.0핀에 HIGH(1) 신호가 인가되게 된다. 즉, 버튼을 계속 눌러서 A.0핀에 입력되는 신호가 HIGH를 유지하는 한 LED가 LSB에서 MSB 방향으로 0.5초마다 순차적으로 이동하는 동작을 한다.
③ 그러한 도중에 버튼에서 손을 떼어 A.0핀에 인가되는 전압이 LOW가 되면 LED는 그 직전의 상태를 유지해야 한다.
④ 그 상태에서 다시 버튼을 누르는 작업을 하면 LED는 그 전 상태에 이어서 동작을 재개하게 된다.
⑤ 점등 되는 LED의 진행 순서는 B.7 이후에는 B.0에서 다시 시작하는 형태로 하라.

(1) 회로 구성

<그림 3-29>와 <그림 3-30>은 본 실습을 위한 회로 구성이다. 포트 B의 B.0~B.7 핀에 각각 LED를 연결하고 A.0 핀에 버튼(button)을 이용하는 회로를 구성한다.<그림 3-29>에서 사용한 버튼은 평소에는 개방(open)되어 있다가 버튼이 눌려져 있을 때에만 단락(short)되는 형태이다. 즉, Normally-OFF Pushed-ON 형태이다. 그리고 회로의 동작상 평상시에는 핀 A.0의 상태가 LOW(0)을 유지하게 되고 버튼을 누르고 있는 동안에만 HIGH(1) 상태가 A.0핀에 나타나게 된다.



<그림 3-29> 포트 A 회로 구성



<그림 3-30> 포트 B 회로 구성

(2) 프로그램

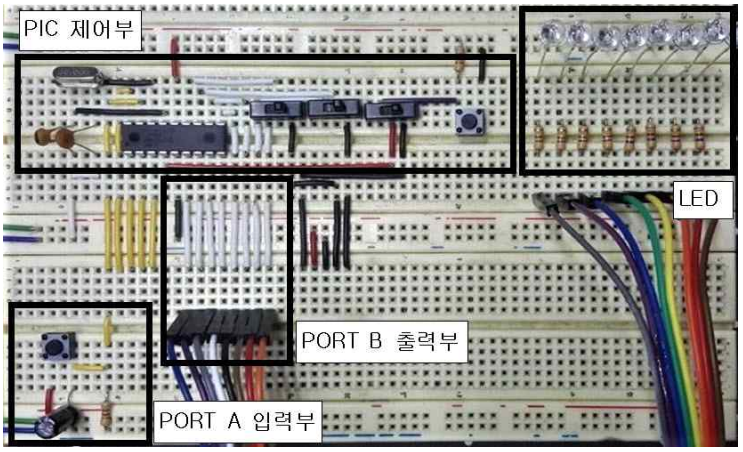
다음은 포트 A의 A.0 핀 상태 값을 주기적으로 읽어서 A.0핀의 상태에 합당한 동작을 하도록 구성한 프로그램이다. 평상시에는 A.0핀의 입력 값이 LOW(0) 상태여서 포트 B에 연결된 LED가 현재 상태를 유지하게 된다. 만일 버튼을 눌러서 A.0핀에 입력되는 신호가 HIGH(1)가 되도록 하면 포트 B에서 점등되는 LED가 이동하게 된다. 이러한 동작은 버튼이 눌려져 있는 동안 계속된다.

```
#include <16f628a.h> //include device header file
#fuses XT, NOWDT, PUT, NOPROTECT
#use delay(clock=4000000) //oscillator setting

#byte porta = 0x05 //port A aliasing
#byte portb = 0x06 //port B aliasing

#bit input = porta.0 //input button pin aliasing

void main(void)
{
        int cnt, temp;  //define variable
        set_tris_a(0xff); //set all pins of port A as input
        set_tris_b(0x00); //set all pins of port B as output

        temp = 0x00;
        portb = temp;  //output port B with initial value
        cnt = 0;
```

```
        while(1) //infinite loop
        {
                if(input==1) //if button is pressed
                {
                  if(cnt==0) temp=0x01; //load temp with initial value
                  else temp=temp<<1;   //shift temp value to the  left

                   portb=temp; //output port B with temp value

                   cnt++;
                   if(cnt>7) cnt=0; //set cnt value with initial value
                   delay_ms(500); //time delay 500ms
                 }
         }
}
```

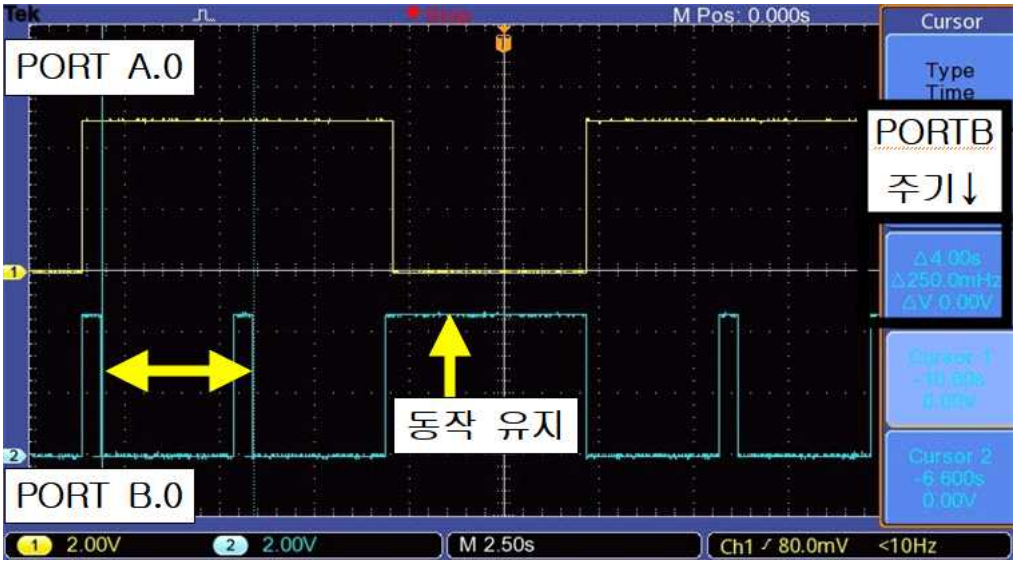
(3) 실습 결과

아래에 보인 바와 같이 실습을 진행하였다.



<그림 3-31> 실제 회로 구성

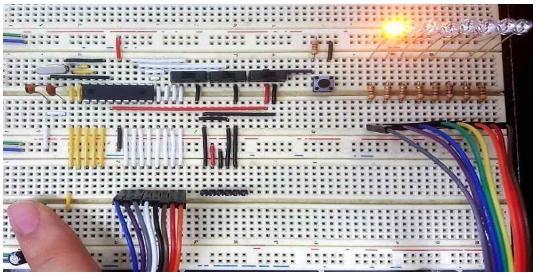
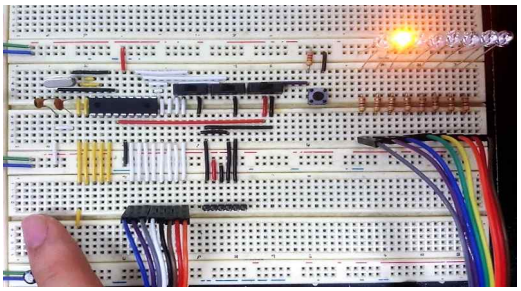
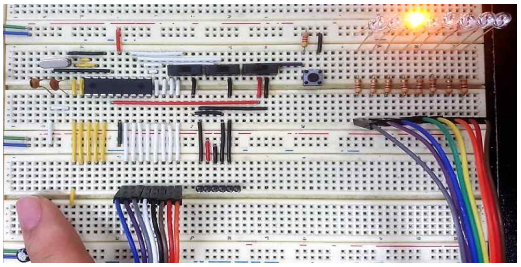
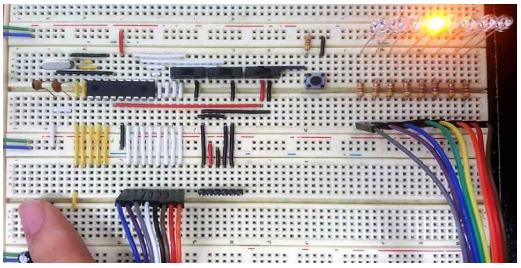
<그림 3-32>는 실습과정 중에 A.0핀과 B.0핀에서 출력되는 신호를 오실로스코프로 측정한 결과이다. 결과파형에서 알 수 있듯이 A.0핀이 HIGH(1) 상태를 유지하는 동안 B.0핀에서 LED 점멸이 진행되는 것을 볼 수 있다. 그리고 B.0가 점등되어 있는 순간에 버튼을 놓아서 A.0의 신호를 LOW 상태로 만들면 B.0는 직전의 상태를 유지함을 알 수 있다.

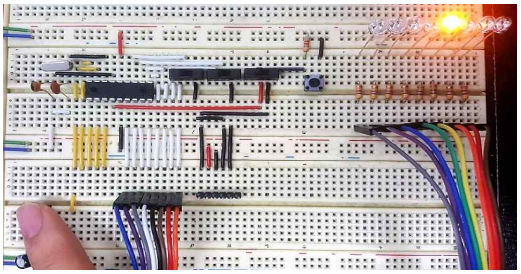
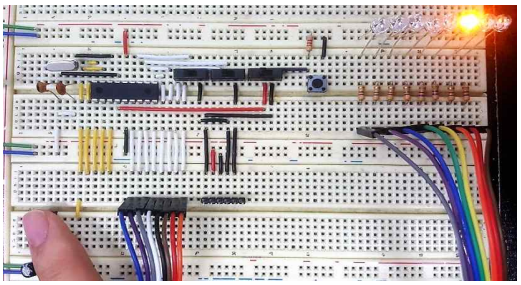
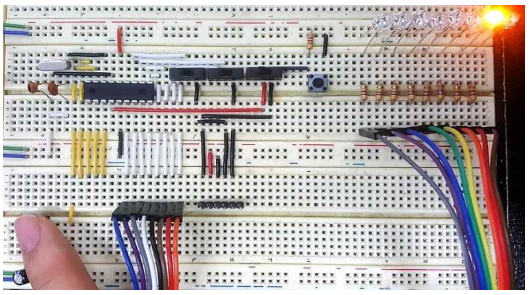
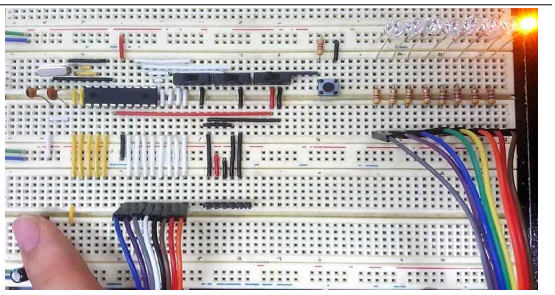
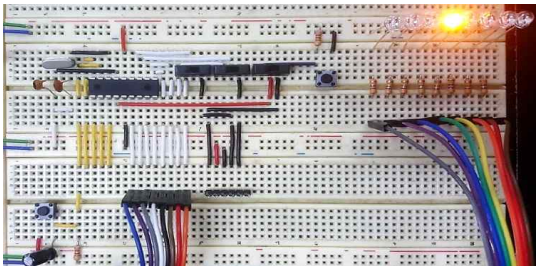


<그림 3-32> 실습 결과 파형 (A.0와 B.0핀의 파형)

아래의 <표 3-5>을 통하여 전체 동작과정을 보였다.

<표 3-5> 실습 결과

버튼을 계속 누르고 있을 때의 LED 동작	
	
	

<table>
<tr><td>  </td><td>  </td></tr>
<tr><td>  </td><td> 
이후 계속 반복</td></tr>
<tr><td colspan="2">버튼에서 손을 떼었을 때의 LED 동작 – 직전 상태 유지</td></tr>
<tr><td colspan="2">  </td></tr>
</table>

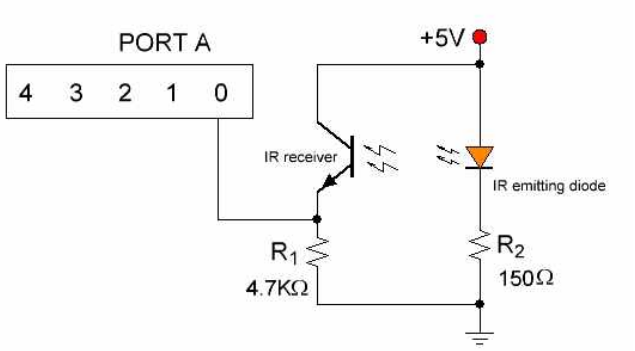
[2] 실습 2 - 직류 전동기(DC Motor) 구동

이번 실습은 외부에 적외선 센서 회로를 구성하고 적외선 센서의 수신부 상태를 포트 A의 A.0핀을 통하여 입력받아서 이 핀의 상태에 따라 B.0핀에 연결된 직류 전동기를 구동시키는 것이 목적이다. 다음은 본 실습의 동작조건이다.

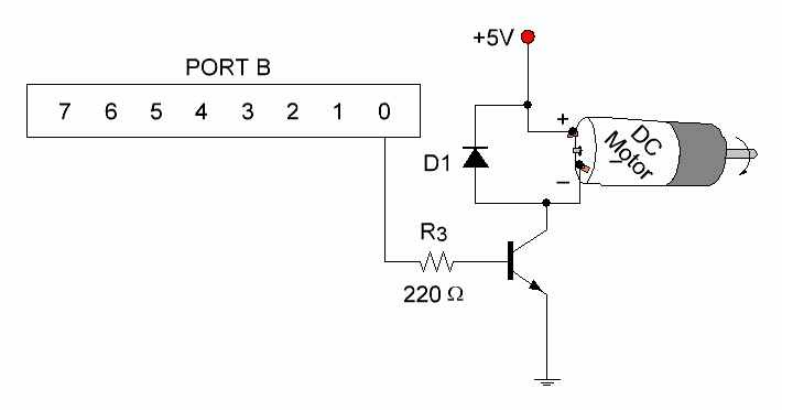
① 본 실습에서 사용한 적외선 센서의 발광부와 수광부는 서로 마주보는 형태로 배치한다. 평상시에는 적외선 센서의 발광부와 수광부 사이의 신호전달이 원활하여 입력으로 설정된 핀 A.0에 HIGH(1) 신호가 인가된다.
② 이들 발광부와 수광부 사이의 신호 전달이 장애물로 인하여 차단되면 A.0핀의 상태가 LOW(0)가 된다. 이러한 회로구성 방식을 break beam sensor 방식이라고 한다.
③ 만일 A.0핀의 입력이 HIGH인 경우 다시 말해서 장애물이 존재하지 않아서 적외선 센서간의 신호전달이 원활한 경우에는 모터 구동핀인 B.0핀의 출력을 LOW로, 장애물의 존재로 인하여 A.0핀의 입력이 LOW인 경우에는 모터 드라이브 핀 B.0에 HIGH 값을 출력하여 직류 전동기를 회전시킨다.

(1) 회로 구성

<그림 3-33>과 <그림 3-34>는 본 실습을 위한 구동회로이다. <그림 3-33>에 보인 적외선 센서 구동 회로는 평상시에는 발광부에서 발생하는 적외선 신호가 수광부에 원활하게 전달되도록 하였다. 따라서 수광부의 도통으로 인하여 저항 R_1에서 전압강하가 발생한다. 그러나 만일 장애물이 이들 두 센서 사이에 놓이게 되면 수광부가 차단상태가 되어 R_1에 전류가 흐르지 않게 되므로 A.0핀에 LOW(0)신호가 인가된다. <그림 3-34>에 보인 회로는 출력 핀 B.0에서 발생한 신호를 통하여 직류 전동기를 구동할 수 있도록 하는 전류증폭회로이다. PIC 마이컴의 출력 전류는 최대 20mA 정도로 직류 전동기를 직접 구동시키기에는 다소 부족한 면이 있다. 따라서 트랜지스터를 사용하여 전류를 증폭할 수 있는 드라이브 회로가 필요하다. 이 때 사용되는 트랜지스터는 스위치의 역할을 해야 하기 때문에 트랜지스터의 포화(saturation)영역과 차단(cut-off)영역을 활용해야 한다.



<그림 3-33> 포트 A 회로 구성



<그림 3-34> 포트 B 회로 구성

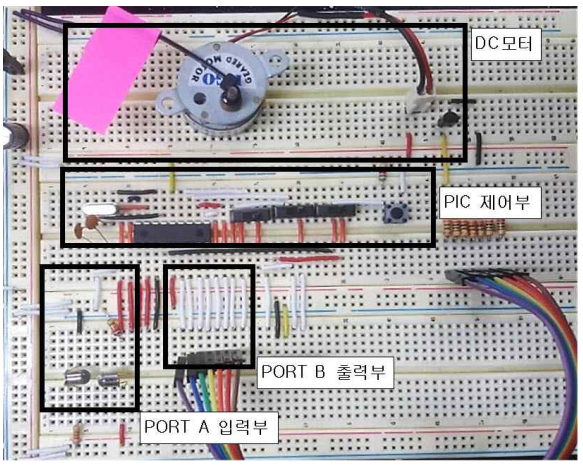
(2) 프로그램

```
#include <16f628a.h>    //include device header file
#fuses XT, NOWDT, PUT, NOPROTECT
#use    delay(clock=4000000)

#byte porta = 0x05      //port A aliasing
#byte portb = 0x06      //port B aliasing

#bit input = porta.0  //input pin aliasing - infrared signal(active low)
#bit output = portb.0 //output pin aliasing - DC motor drive(active high)

void main(void)
{
        set_tris_a(0xff); //set all pin of port A as input
```

```
        set_tris_b(0x00); ////set all pin of port B as output

        while(1) //infinite loop
        {
                if(input==0) //if the infrared signal is blocked
                {
                        output = 1; //make DC motor ON
                }
                else
                {
                        output = 0; //make DC motor OFF
                }
        }
}
```

(3) 실습 결과

아래와 같이 실습을 진행하였으며 그 결과를 <그림 3-36>과 <표 3-6>에 나타내었다.

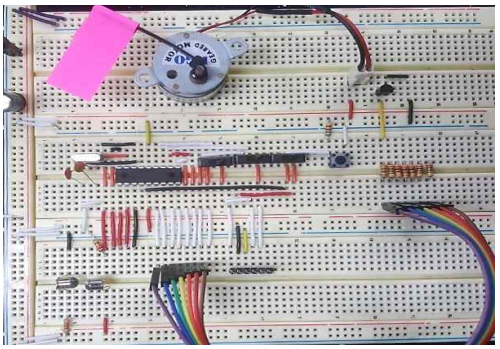
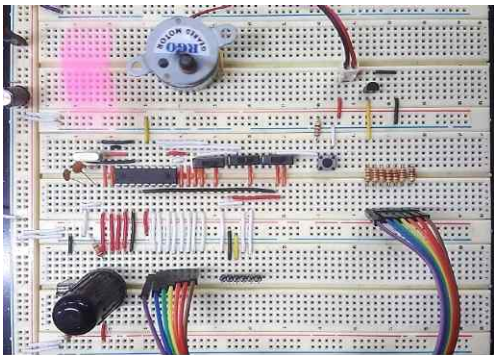


<그림 3-35> 실제 회로 구성

<그림 3-36>은 입력 A.0핀과 출력 B.0핀에서의 신호를 오실로스코프를 통해 측정한 결과 파형이다. 결과파형을 통하여 적외선 센서 사이를 물체가 가로막았을 때 즉, A.0핀의 입력 신호가 HIGH에서 LOW가 되는 순간, 출력 B.0핀의 상태가 LOW(0)에서 HIGH(1)로 전환되는 것을 확인할 수 있다.

<그림 3-36> 실습 결과 파형

<표 3-6> 실습 결과

	수광부에 적외선 신호가 들어올 때 (모터 정지 상태)
	수광부에 적외선 신호가 들어오지 않을 때 (모터 회전 상태)