실험 4. 숫자를 표현해봅시다.

여러분 스스로 숙제를 확인하여 봅시다.

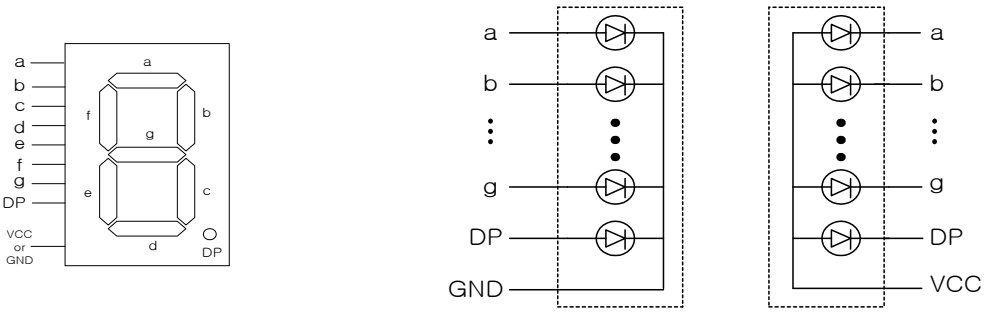
어떻게 확인하지요? 수업 시간보다 약간 일찍 와서 눈으로 확인해 보면 되지요.

확인 안된 프로그램을 어떻게 믿지요?

1. 7-SEGMENT란

LED는 ON/OFF 제어만 가능하므로 어떤 정보를 출력하기에는 부족함이 있지요. 그러면 디지털 값으로 최소한 숫자라도 표현해 보는 OUTPUT 장치는 없을까요? 이것이 7-segment입니다. 10진 숫자('0'에서 '9' 까지의 숫자)는 7개의 막대기로 표시가 가능하며, 둥그런 LED 앞면을 길게 만들면 막대 모양의 불을 만들 수 있지요. 이것을 7개 묶어 한 개의 소자로 만든 것이 7-segment 표시기입니다. 여러분이 흔히 보는 디지털 시계의 숫자 표시와 같은 것입니다. 7-segment는 원칙적으로 LED가 7개이나, 소수점을 나타내기 위하여 1개가 더 필요하여 실제로는 8개의 LED가 들어 있는 것이 표준입니다. 그러므로 7-segment를 사용하기 위해서 결선 해야되는 선이 2×8개가 되어 아주 복잡한 부품이 됩니다. 결선을 최소화시키는 방법이 없을까요?

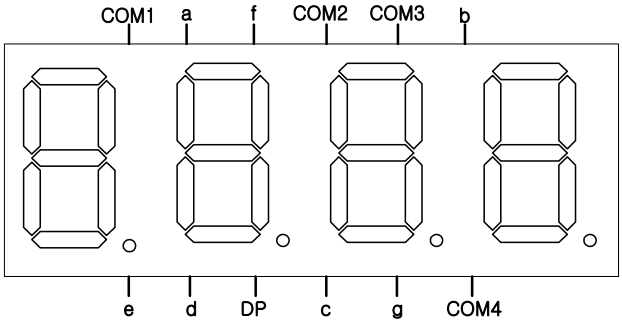
2. 7-SEGMENT의 결선방법



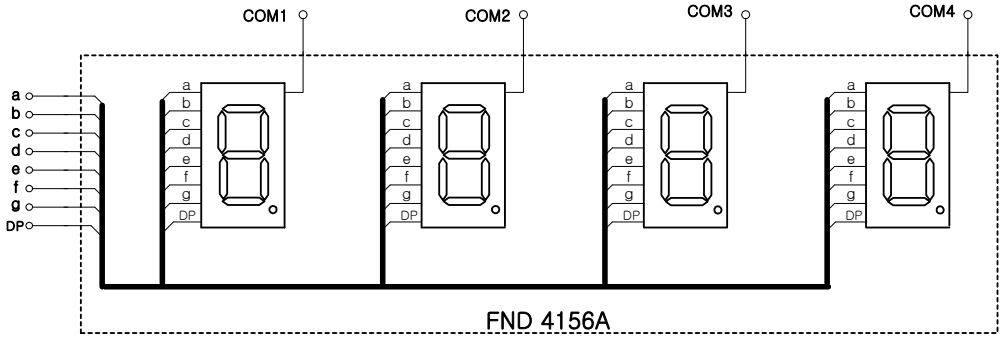
7-Segment C-C TYPE C-A TYPE

그림 E4-1. 7-SEGMENT의 구조 및 내부 회로

그림에서 볼 수 있듯이 LED 한쪽을 공통으로 연결하지요. 공통되는 것에 따라 C-C(Common Cathode), C-A(Common Anode) 두 종류가 있습니다. 그리고 7-segment를 1개만 사용하는 경우는 아주 드물고 여러 개를 사용하지요. 예를 들어 4개를 사용하면 4×9 개의 연결선이 필요하지요. 아주 많은 선입니다. 이것을 줄이기 위해서 각 segment를 다시 병렬로 연결합니다. 그러면 다음 그림과 같은 소자가 만들어집니다.



(a) 4-DIGIT 7-SEGMENT 외형



(b) (a) 4-DIGIT 7-SEGMENT 내부회로

그림 E4-2. 4-DIGIT 7-SEGMENT 내부회로

Digit를 구동하는 공통선은 여러 개의 LED을 구동하므로 전류가 많이 필요하지요. 일반적으로 1개의 LED는 약 10mA의 전류를 필요로 하며, 로직 회로 소자가 구동하는 전류는 약 20mA 입니다. 따라서 실험보드의 회로와 같이 전류 증폭 소자(트랜지스터)를 추가로 사용하는 경우가 많습니다.

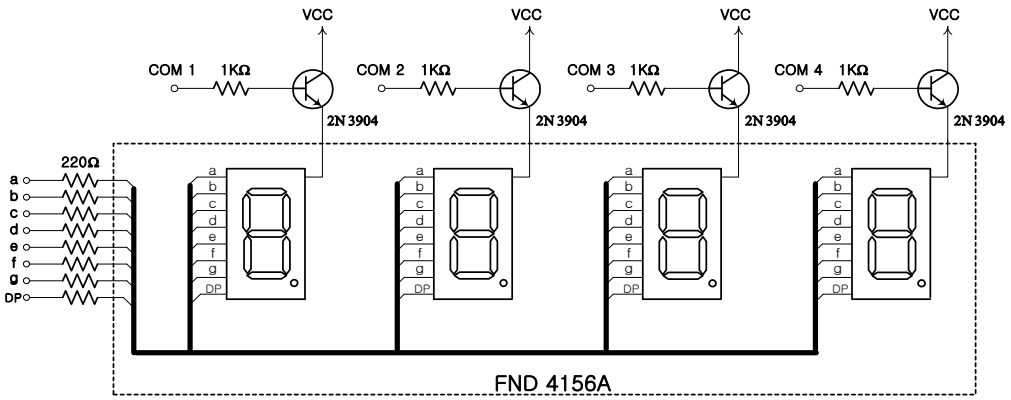


그림 E4-3. 실험보드에 있는 FND DISPLAY 회로도

앞에서 설명하였지만 결선을 줄이기 위해서 공통으로 연결하다보니 개별 소자 개념에서 특정 위치에 특정 숫자를 표시하기가 불가능해졌지요. 그래서 7-segment를 구동하는데는 조금 복잡한 개념이 사용됩니다. 즉 동시에 4개의 7-segment에 다른 숫자를 쓰는 것은 안되므로 특정 시간 동안만(아주 짧은 시간) 특정 위치에(회로도에서는 COM?) 특정 숫자(7-segment 입력 단자)를 표시하고, 위치를 이동하여 반복하는 것입니다. 그러면 사람은 LED의 잔상으로 LED 전체가 항상 불이 들어와 있는 것으로 인식하지요. 당연히 불 밝기는 어두워지겠지요. 실험 도중에 프로그램을 멈추면 하나만 불이 들어오고 아주 밝은 이유를 이해하겠지요.

3. 7-SEGMENT에 숫자 하나 써보기

실험보드에 있는 7-segment는 C-A type이므로 LED를 ON시키기 위해서 COM에 +5V('1')를 인가하고 segment에 0V('0')를 인가해야 합니다. 따라서, 가장 뒷자리에(DG4:COM4) '5'라는 숫자를 쓰려면 DG4(COM4) 단자에 '1'을 A, B, C, D, E, F, G, DP 단자에 '0, 1, 0, 0, 1, 0, 0, 1'을 인가하면 됩니다. 이것을 프로그램으로 하려면 회로는 **RA3, 2, 1, 0을 DG1(COM1), 2, 3, 4** 단자에 연결하고, **A, B, C, D, E, F, G, DP 는 RC7, 6, 5, 4, 3, 2, 1, 0에 연결**하고 출력 PORTA에 B'00000001'을 PORTC에 B'01001001'을 출력하면 됩니다. 이것을 셋째 digit(DG3)에 표시하려면 PORTA의 값을 B'00000010'로 하면 됩니다.

4. 7-SEGMENT에 서로 다른 두 개의 숫자 써보기

만약 마지막 두자리(DG3,4)에 서로 다른 숫자 '51'을 쓰기 위해서는 PORTC에 B'01001001'('5' code data), PORTA에 B'00000010'(DG3 선택)을 출력하고 최소 5msec 정도 지연시키고, 다시 PORTA에 B'00000000'을(자리 이동 중에 앞쪽 위치의 글자가 나타나는 것을 막기 위해서 digit 전부를 OFF 시킴), 그리고 PORTC에 B'10011111'('1' code data), PORTA에 B'00000001'(DG4 선택)을 출력하고 약간 지연시키고, 앞과 동일한 이유로 PORTA에 B'00000000'을 출력하고, 이를 계속 반복하도록 하면 '51'이 표시됩니다. 이 방법은 하드웨어적으로는 하나씩만 불이 들어오나 사람 눈의 잔상 효과를 이용하여 두 개가 동시에 켜져있는 것과 같은 효과를 나타내는 방식이며, 이를 scanning 이라고 부릅니다. 만약 시간 지연을 너무 짧게 하거나 길게 하면 발기가 어두워지거나 껌벅거릴 수 있습니다. 따라서 적당한 주기로 반복해야합니다. 이러한 scanning 표시 방법은 우리가 사용하는 모든 표시장치에 사용되는 공통적인 방법이므로 꼭 이해해 두어야합니다.

5. 변수 DISP1, DISP2에 들어있는 값으로 7-SEGMENT에 숫자 써보기

앞의 설명에서는 고정된 값을 PORTA, C에 출력하여 숫자를 표시하여 보았으나, 실제로는 고정된 숫자만 표시하는 것이 아니라 변하는 값을 표시해야 하므로 변수에 들어있는 내용을 표시할 수 있도록 하여 보자. 이 경우 digit 선택을 변수로 할 것이냐? 고정시킬 것이냐에 따라서 변수를 하나 더 사용 할 수도 있다. 고정시키는 경우의 프로그램을 아래에 나타내었다.

```
D_LOOP    MOVF      DISP1,W
          MOVWF     PORTC       ; 숫자 값 출력
          MOVLW     B'00000001'
          MOVWF     PORTA       ; 위치 결정 (COM1)
          CALL      DELAY       ; 표시하기 위한 DELAY 시간
          MOVLW     0
          MOVWF     PORTA       ; 전부 OFF
```

```
        MOVF    DISP2,W
        MOVWF   PORTC       ; 숫자 값 출력
        MOVLW   B'00000010'
        MOVWF   PORTA       ; 위치 결정 (COM2)
        CALL    DELAY       ; 표시하기 위한 DELAY 시간
        GOTO    D_LOOP
```

프로그램에서 CALL DELAY는 LED에 출력된 값이 일정시간 빛나도록 하기 위한 시간지연을 만드는 작용을 하며, 5msec가 되도록 설정하면 된다.

여기까지는 여러분이 쉽게 이해하였다고 봅니다. 그러나 위 프로그램이 정상적으로 동작하기 위해서는 DISP1, 2에 표시하고자 하는 숫자값이 들어가는 것이 아니고, on/off시키고자 하는 A, B, C, D, E, F, G, DP segment의 on/off 데이터가 들어 있어야 합니다. 다시 말하면, DISP1 변수에 '01' 이라는 값이 있으면 표시기에 '1'을 표시하지 않고 글자가 되지 않는 표시가 나옵니다. 이것을 '1'이 표시되도록 하려면 '01' 데이터를 B'10011111'로 변경하여 PORTC에 출력 해야합니다. 즉 DISP 변수에 있는 숫자값을 segment ON/OFF 값으로 변환시키는 작업이 필요합니다. 이를 table을 이용하면 간단히 구현할 수 있으며(이런 방법을 lookup table 방법이라 합니다.), 이 방식은 프로그램 작업에서 매우 많이 사용되는 기능이므로 이것을 용이하게 하기 위한 명령어가 준비되어 있습니다.

```
CONV    ANDLW   0FH     ; W의 low nibble 값을 변환하자.
        ADDWF   PCL,F   ; PCL+변환 숫자값 --> PCL
                        ; PC가 변경되므로 이 명령어 다음 수행 위
                        ; 치가 변경되지요.
        RETLW   B'00000011' ;'0'를 표시 하는 값이 W로 들어옴
        RETLW   B'10011111'   ; '1'를 표시 하는 값
        RETLW   B'00100101'   ; '2'를 표시 하는 값
        RETLW   B'00001101'   ; '3'를 표시 하는 값
        RETLW   B'10011001'   ; '4'를 표시 하는 값
        RETLW   B'01001001'   ; '5'를 표시 하는 값
        RETLW   B'01000001'   ; '6'를 표시 하는 값
        RETLW   B'00011011'   ; '7'를 표시 하는 값
        RETLW   B'00000001'   ; '8'를 표시 하는 값
```

```
        RETLW   B'00001001'   ; '9'를 표시 하는 값
        RETLW   B'11111101'   ; '-'를 표시 하는 값
        RETLW   B'11111111'   ; ' '를 표시 하는 값
        RETLW   B'11100101'   ; 'C'를 표시 하는 값
        RETLW   B'11111110'   ; '. '를 표시 하는 값
        RETLW   B'01100001'   ; 'E'를 표시 하는 값
        RETLW   B'01110001'   ; 'F'를 표시 하는 값
```

☞ ADDWF PCL,F 명령어는 현재 수행되는 프로그램의 수행 위치를 강제로 변경하는 것이므로 사용상에 세심한 주의가 요구된다. 특히 PCL 값이 넘쳐 CARRY가 발생하는 조건(data sheet에서 PCL과 PCLATH REG.에 대한 내용 참고)에서는 비정상적인 동작이 일어나게 된다.

따라서 위 프로그램을 다음과 같이 변경하면 됩니다.

```
        .
        .
D_LOOP  MOVF    DISP1,W     ; 변환할 값을 W에 넣기
        CALL    CONV        ;변환하기 위하여 추가된 부분
        MOVWF   PORTC       ; 숫자 값 출력
        MOVLW   B'00000001'
        MOVWF   PORTA       ; 위치 결정 (COM4)
        CALL    DELAY       ; 이 DELAY는 시간이 약
                            ; 5msec가 되도록 작성
        .
        .
```

■ 예비문제 4

1) 7-segment가 C-A type일 때, 숫자 '0'에서 '9' 까지를 나타내기 위한 segment 값을 써 봅시다.

2) 7-segment에 서로 다른 두 문자 써보기에서 중간 이동 중에 앞쪽 위치의 글자가 나타나는 것을 막기 위해서 전부를 OFF 시키는 명령어가 필요하다고 하였다. 이것을 생략하면 어떤 현상이 나오는가를 구체적으로 설명하시오.

3) 7-segment 4 digit에 '1234'가 표시 되도록 프로그램 하시오.

4) ADDWF PCL,F 와 RETLW의 명령어를 설명하시오.

5) 위의 CONV 부 프로그램에서 ANDLW 0FH 명령어가 필요한 이유를 설명하고, '0'에서 '9'까지에 해당하는 10개의 RETLW data 명령어만을 사용하는 경우 어떤 문제가 발생하는가를 설명하시오.

■ 실 험 4

1) 실험보드에 있는 7-segment는 무슨 type인지 전원을 인가하여 알아봅시다. 이때 그림 E4-1를 보면 최소한 2개의 단자에 전원을 넣어야 됩니다.

2) 다음 프로그램을 이용하여 7-segment에 '67' 이라는 숫자가 표시되도록 합시다.

```
D_LOOP  MOVF    DISP1, W
        MOVWF   PORTC       ; 숫자 값 출력
        MOVLW   B'00000001'
        MOVWF   PORTA       ; 위치 결정 (COM4 선택)
        CALL    DELAY       ; 이 DELAY는 시간이 약
                              5msec가 되도록
        MOVLW   0
        MOVWF   PORTA       ; 전부 OFF
        MOVF    DISP2, W
        MOVWF   PORTC       ; 숫자 값 출력
        MOVLW   B'00000010'
        MOVWF   PORTA       ; 위치 결정 (COM3 선택)
        CALL    DELAY       ; 이 DELAY는 시간이 약
                              5msec가 되도록
        MOVLW   0
        MOVWF   PORTA       ; 전부 OFF
        GOTO    D_LOOP
```

☞ 참고: 이것은 main 프로그램 만이므로 동작하기 위해서는 변수 선언 등과 나머지 설정 프로그램이 필요하죠.

3) 위 프로그램에서 DELAY 시간을 길게 하여 수행시켜 보고, 또 아주 짧게 하여 수행시켜 DELAY 시간의 변경에 따른 영향이 어떻게 나타나는가를 확인합시다. 그리하여 본인 생각에 가장 좋은 DELAY 시간을 설정합시다.

4) 위 프로그램에서 전부 OFF 하는 명령어 부분을 제거하고 수행시켜 예비문제 2)의 내용을 확인해 봅시다.

☞ 이때는 DELAY 시간이 짧은 것이 좋습니다. 극단적으로 DELAY 생략
☞ 아직도 source program에서 작성된 명령어를 제거하거나 생략하려면 진짜로 명령어를 일일이 지워야 합니까? -..-; 공부 좀 합시다.

5) 예비문제 3)를 동작시켜 봅시다.

6) Lookup table 방법을 이용하여 예비문제 3)을 구현하시오.

☞ Editor 화면에서 특정 부분을 복사하여 수정하면 프로그램 작성 시간이 많이 줄어들지요!!! SHIFT-CURSOR, CTRL-C, CTRL-V 등을 활용하시오.

♣ 지금까지의 내용을 충분히 이해하고 실험으로 확인하였으면, 한 단계 어려운 HOME WORK 4)을 하도록 하고, 아니면 이해할 때까지 GOTO 5장 7-segment 동작시키기로 -- (GOTO 명령어는 알고 있으니 다른 명령어도 공부해 봅시다.)

■ QUIZ 4

1) 백지에 지금까지 사용한 명령어를 쓰고 사용방법, 의미, 기능, 주의점 및 status 변화 등을 정리할 것.

2) 4개의 7-segment 표시기를 사용하여 가장자리 segment가 순차적으로 이동하면서 on되는 snake 모양의 프로그램을 작성하고 동작을 설명하시오.

■ HOME WORK 4

1) 여러분이 실험한 내용은 DISP1, DISP2, DISP3, DISP4에 들어있는 값을 7-segment에 표시하는 것만 작성하고 실험해 보았다. 그러나 일반적으로 어떤 값을 표시하는 것은 주 기능이 아니므로, 프로세서는 자기 일을 하면서 변화된 결과가 발생되면, 잠시 표시기를 동작시키는 일을 해야한다. 그러나 우리가 사용한 표시기는 하드웨어의 비중을 줄이기 위하여 모듈화된 7-segment을 사용하였으며, 따라서 한 순간에 한 숫자만을 표시할 수 있으므로 2 자리 이상을 표시하기 위해서는 앞의 예처럼 계속적으로 반복 표시해야만 한다.

☞ 이러한 반복을 'scanning: 주사' 라고 한다.

예를 들어 시계를 만든다면, 시간을 정확하게 얻어내는 것이 주된 일이고 표시는 그것을 외부에 알려 주는 것이다. 그러므로 시간을 만들면서(main program이 됨) 시간을 표시하는 일(부 program이 됨)을 동시에 해야한다. 가능한 방법이 무엇이 있겠는가? 구체적으로 설명해 보시오.

2) 실험에서는 숫자가 나오는 위치를 프로그램에서 상수로 고정하여 출력을 내 보냈는데, digit 선택을 변수를 사용하여 그 안에 들어있는 내용 값으로 위치와 숫자가 나타나도록 하는 부 프로그램을 작성해 보시오.

☞ Digit 변수의 high nibble(4bit) 값이 표시위치(0,1,2,3), low nibble(4bit) 값이 표시숫자가 되도록 할 것.

지금까지의 실험한 내용이 정리된 note를 다음 시간에 제출합시다.