

## 8 장 : FND로 자동차 주차용 전화번호표시기 만들기

---



**JCnet**  
제이씨넷

신상석

# 목차

---

- FND
- JKIT-128-1에서의 FND 연결 설계
- 실습 FND-1 : GPIO로 FND 1개 표시하기
- 실습 FND-2 : GPIO로 FND 4개 표시하기
- 실습 FND-3 : FND로 1/100초 stopwatch 만들기
- 실습 FND-4 : FND로 자동차주차용 전화번호표시기 만들기

# FND

---

## □ FND란?

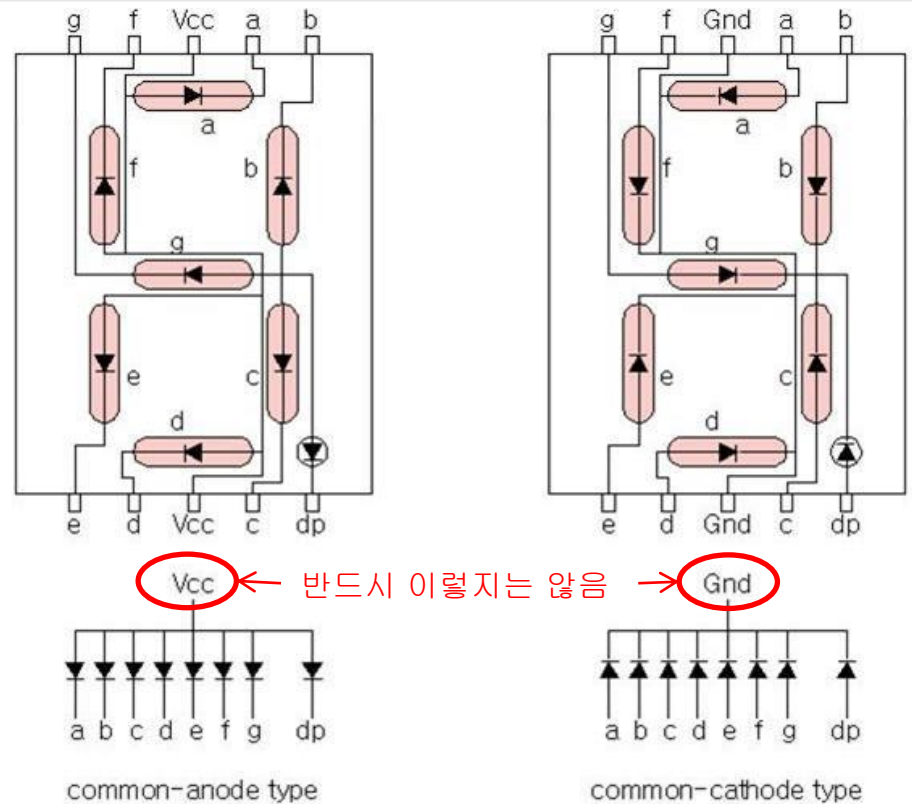
- Flexible Numeric Device의 약자
- 보통 7-Segment LED라고 칭함
- LED 7개(점 포함 8개)로 숫자를 표시하기 쉽도록 배열한 제품
- 1개, 2개, 3개, 4개를 함께 디스플레이하는 형태의 제품 판매
- 많이 사용하는 곳 : 엘리베이터 층 표시기, 임베디드 제품 상태 표시기



# FND

## □ FND 종류 및 구조

- Common Anode 타입 (CA 타입)
- Common Cathode 타입 (CC 타입)



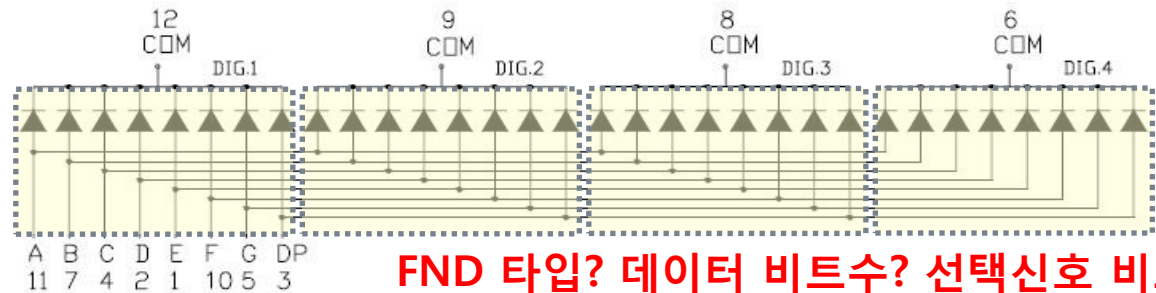
# FND

## □ 다중 FND

- 여러 개의 FND가 함께 묶여진 형태로, FND를 한 개씩 빠르게 돌아가면서 디스플레이하는 방식을 사용하면 눈의 잔상효과 때문에 모든 FND가 동시에 디스플레이되는 것처럼 보이는 현상을 이용한 제품 (최소 1/30 초 (33ms) 주기로 디스플레이 필요)
- 숫자를 표시하는 데이터 신호는 공통으로 사용하고, 각 FND를 선택하는 신호는 따로 할당하되, 선택 신호를 Common 노드에 할당



H. WCN4-XX36XX-C1X

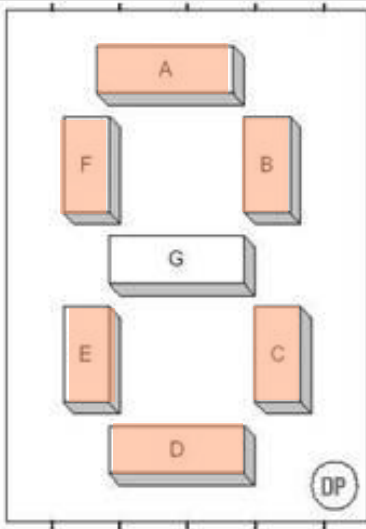


**FND 타입? 데이터 비트수? 선택신호 비트수?  
필요한 GPIO 개수는 ?**

# FND

## □ FND 숫자 표현 (Common-Cathode 경우)

8비트로 '0'을  
표현하려면 ?



16 진수	7-세그먼트의 비트값								데이터 값 ( HEX )
	DP	G	F	E	D	C	B	A	
0									
1									
2									
3	0	1	0	0	1	1	1	1	0X4F
4	0	1	1	0	0	1	1	0	0X66
5	0	1	1	0	1	1	0	1	0X6D
6	0	1	1	1	1	1	0	1	0X7D
7	0	0	1	0	0	1	1	1	0X27
8	0	1	1	1	1	1	1	1	0X7F
9	0	1	1	0	1	1	1	1	0X6F
A	0	1	1	1	0	1	1	1	0X77
B	0	1	1	1	1	1	0	0	0X7C
C	0	0	1	1	1	0	0	1	0X39
D	0	1	0	1	1	1	1	0	0X5E
E	0	1	1	1	1	0	0	1	0X79
F	0	1	1	1	0	0	0	1	0X71

# JKIT-128-1에서의 FND 연결 설계

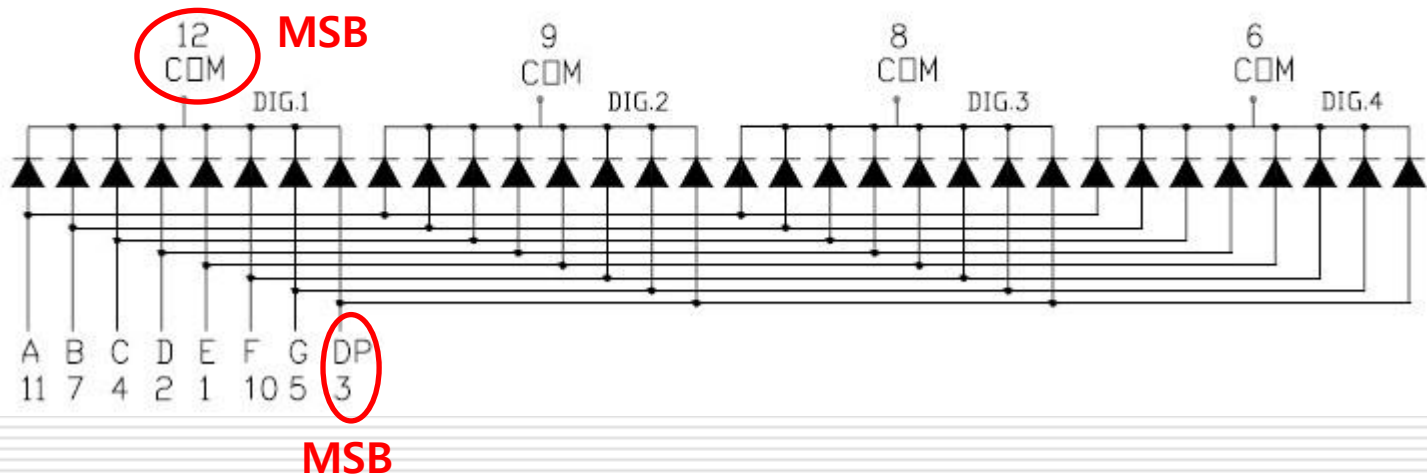
## □ FND 선택

- 요구사항 : 4중FND, CC 타입, 적은 전류, 작은 크기, 가격 저렴
- 선택 FND : WCN4-0036SR4-C11
  - Segment 당 전압은? (      )V, 전류는? (      )mA

[www.ic114.co.kr](http://www.ic114.co.kr)

H. WCN4-XX36XX-C1X

총 필요 전류량은?



# JKIT-128-1에서의 FND 연결 설계

□ FND : WCN4-0036SR4-C11

## ELECTRICAL/OPTICAL CHARACTERISTICS AT $T_a=25^{\circ}\text{C}$

WCN1-0036SR-A11U/C11U;WCN2-0036SR-A11/C11;WCN3-0036SR-A11/C11;WCN4-0036SR-A11/C11

PARAMETER	SYMBOL	MIN.	TYP.	MAX.	UNIT	TEST CONDITION
Luminous Intensity Per Segment	$I_v$	2.0	3.0	—	mcd	$I_f=10\text{mA}$
Dominant Wavelength	$\lambda_D$	—	643	—	nm	$I_f=20\text{mA}$
Peak Emission Wavelength	$\lambda_P$	—	660	—	nm	$I_f=20\text{mA}$
Spectral Line Half-Width	$\Delta\lambda$	—	20	—	nm	$I_f=20\text{mA}$
Forward Voltage Per Segment	$V_F$	—	1.8	2.0	V	$I_f=20\text{mA}$
Reverse Current Per Segment	$I_R$	—	—	100	$\mu\text{A}$	$V_R=5\text{V}$
Luminous Intensity Matching Ratio (Segment To Segment)	$I_{v-m}$			2:1		$I_f=10\text{mA}$



# JKIT-128-1에서의 FND 연결 설계

□ FND : WCN4-0036SR4-C11

## ABSOLUTE MAXIMUM RATINGS AT $T_a=25^{\circ}\text{C}$

PARAMETER	SH.RED	ORANGE	Yellow GREEN	UNIT
Power Dissipation Per Segment	50	65	65	mW
Peak Forward Current Per Segment (1/10duty cycle 0.1ms pulse width)	100	100	100	mA
Continuous Forward Current Per Segment	25	25	25	mA
Derating Linear From 25°C Per Segment	0.30	0.20	0.33	mA/°C
Reverse Voltage Per Segment	5	5	5	V
Operating Temperature Range	-35°C to + 85°C			
Storage Temperature Range	-35°C to + 85°C			
Solder Temperature 1/16 inch below seating plane for 3 seconds at 260°C				

# JKIT-128-1에서의 FND 연결 설계

---

## □ JKIT-128-1에서의 FND 연결 설계 개념

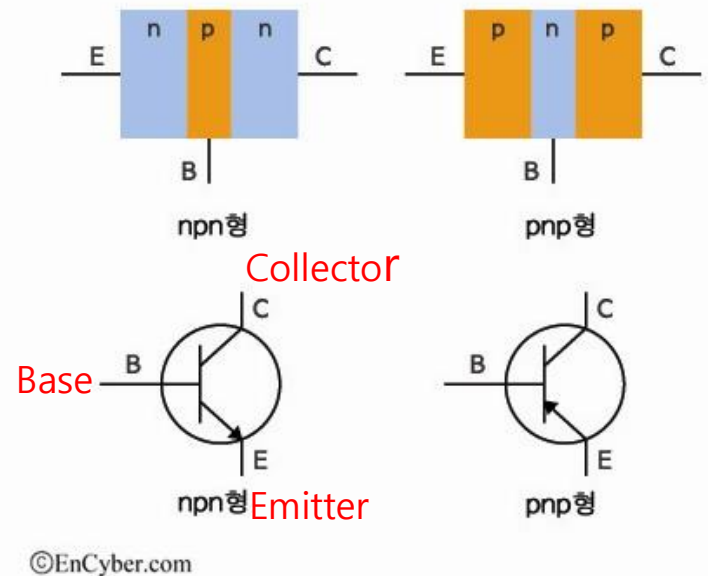
- FND의 데이터(8비트)는 동일한 입출력포트 C(PB)에 할당 (dp-g-f-e-d-c-b-a 순으로 8 비트 할당)
- FND의 선택신호(4비트)는 동일한 입출력포트 G(PG)에 할당 (왼쪽부터 PG3-0의 순으로 4비트 할당)
- COM 신호의 경우 8개의 segment를 동시에 드라이브하면  $25\text{mA} \times 8 = 200\text{mA}$ 의 전류를 처리할 수 있어야 하는데, ATmega128 입출력포트의 최대 능력 100mA를 초과하게 되므로 트랜지스터를 사용하여 전류량을 조절하여야 함. 즉, COM 신호 4개(PG3 ~ PG0)는 트랜지스터를 이용하여 구동
- 트랜지스터(Transistor) 선택 : PMBT2222 선택
- 시리얼 저항값 계산

$$R = (5\text{V} - 1.8\text{V} - 0.2\text{V}) / (12) \text{ mA} = 250 \Omega$$

# JKIT-128-1에서의 FND 연결 설계

## □ TR(Transistor)이란?

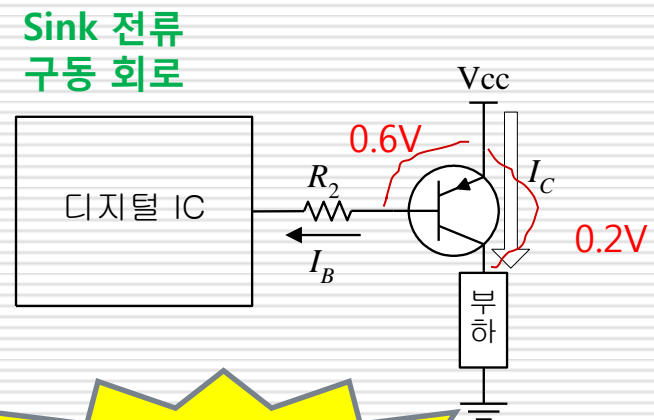
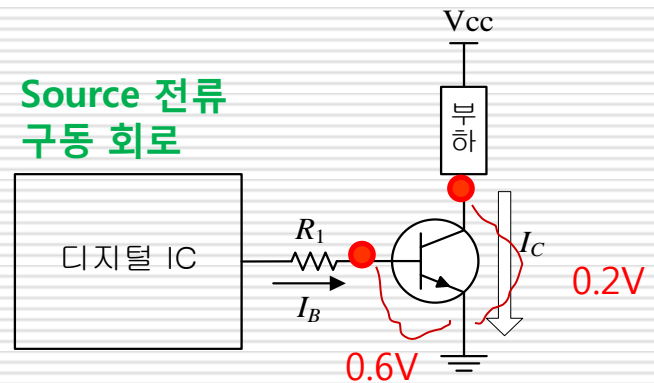
- 규소나 게르마늄으로 만들어진 반도체를 세 겹으로 접합하여 만든 전자회로 구성요소이며 전류나 전압흐름을 조절하여 증폭, 스위치 역할을 하는 전자부품
- 대부분의 전자회로에 사용되며 이를 고밀도로 집적하여 IC를 제작
- BJT(Bipolar Junction Transistor)와 FET(Field Effect Transistor)로 구분하는데 보통은 BJT를 의미함
- BJT는 다시 NPN형과 PNP형으로 구분되며 NPN은 Source 회로에, PNP는 Sink 회로에 사용



# JKIT-128-1에서의 FND 연결 설계

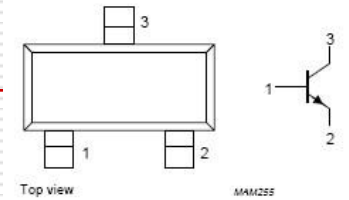
## □ TR(Transistor)을 이용한 부하 구동 회로

- FND나 릴레이, 모터 등의 부하 구동은 전류 소모가 많기 때문에 일반적인 TTL의 출력만으로 직접 구동하기 어려움
- 이런 경우 작은 전류로 큰 전류를 제어할 수 있는 TR을 오른쪽과 같이 연결하여 이용하면 이를 해결할 수 있음
- Source 전류 구동 회로를 예로 들면, 입출력신호를 B(Base)에 연결하고 부하에 연결할 신호를 C(Collector)에 연결
- R1, R2는 보통 약 1K~10K 정도



주의 : TR을 통과한 출력신호는 반전됨 !!!

# JKIT-128-1에서의 FND 연결 설계

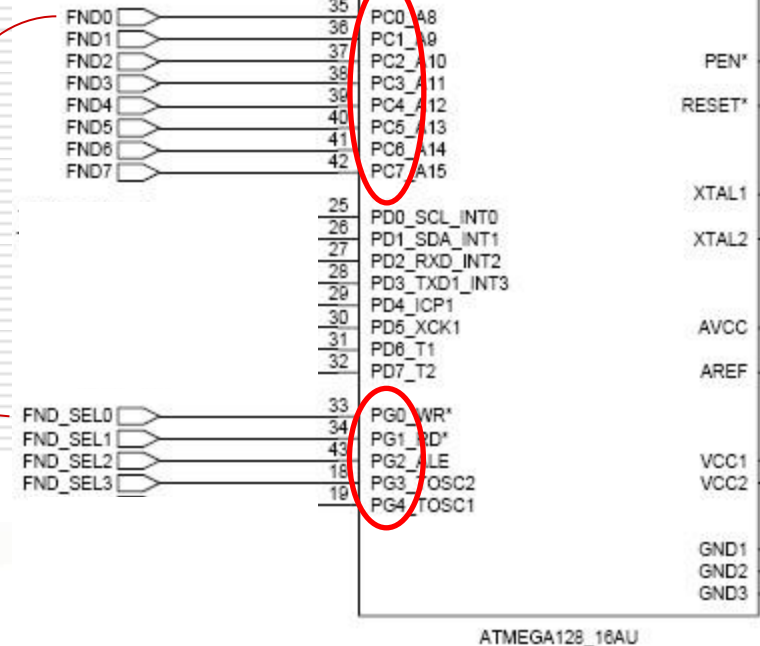
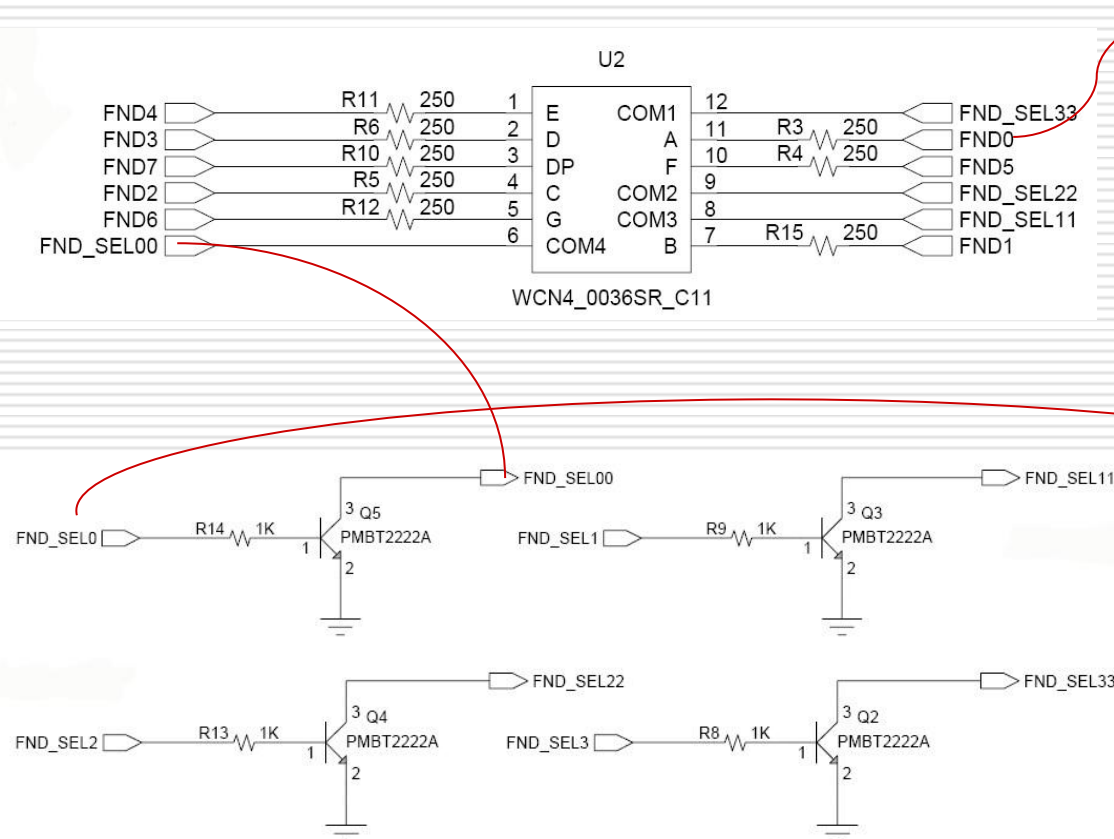


## □ PMBT2222

$h_{FE}$	DC current gain	$I_C = 0.1 \text{ mA}; V_{CE} = 10 \text{ V}$	35	–	
		$I_C = 1 \text{ mA}; V_{CE} = 10 \text{ V}$	50	–	
		$I_C = 10 \text{ mA}; V_{CE} = 10 \text{ V}$	75	–	
		$I_C = 10 \text{ mA}; V_{CE} = 10 \text{ V}; T_{amb} = -55 \text{ }^{\circ}\text{C}$	35	–	
		$I_C = 150 \text{ mA}; V_{CE} = 10 \text{ V}$	100	300	
		$I_C = 150 \text{ mA}; V_{CE} = 1 \text{ V}$	50	–	
	DC current gain PMBT2222 PMBT2222A	$I_C = 500 \text{ mA}; V_{CE} = 10 \text{ V}$	30 40	– –	
$V_{CEsat}$	collector-emitter saturation voltage	$I_C = 150 \text{ mA}; I_B = 15 \text{ mA}; \text{note 1}$			
	PMBT2222		–	400	mV
	PMBT2222A		–	300	mV
	collector-emitter saturation voltage	$I_C = 500 \text{ mA}; I_B = 50 \text{ mA}; \text{note 1}$			
	PMBT2222 PMBT2222A		– –	1.6 1	V V
$V_{BEsat}$	base-emitter saturation voltage	$I_C = 150 \text{ mA}; I_B = 15 \text{ mA}; \text{note 1}$			
	PMBT2222		–	1.3	V
	PMBT2222A		0.6	1.2	V
	base-emitter saturation voltage	$I_C = 500 \text{ mA}; I_B = 50 \text{ mA}; \text{note 1}$			
	PMBT2222 PMBT2222A		– –	2.6 2	V V

# JKIT-128-1에서의 FND 연

## □ JKIT-128-1 FND 회로도



# 실습 FND-1 : GPIO로 FND 1개 표시하기

---

## □ 실습 내용

- FND의 맨 오른쪽 digit에 '0' 표시하기
- FND의 맨 왼쪽 digit에 '7.' 표시하기 (각자 해보기)
- FND의 맨 오른쪽에 digit에 '0' ~ '9'를 1초 간격으로 연속하여 표시하기
- FND의 맨 오른쪽에서 부터 왼쪽으로 1회에 1 digit 씩 이동하면서 '0' ~ '9'를 연속하여 표시하기 (과제)

# 실습 FND-1 : GPIO로 FND 1개 표시하기

---

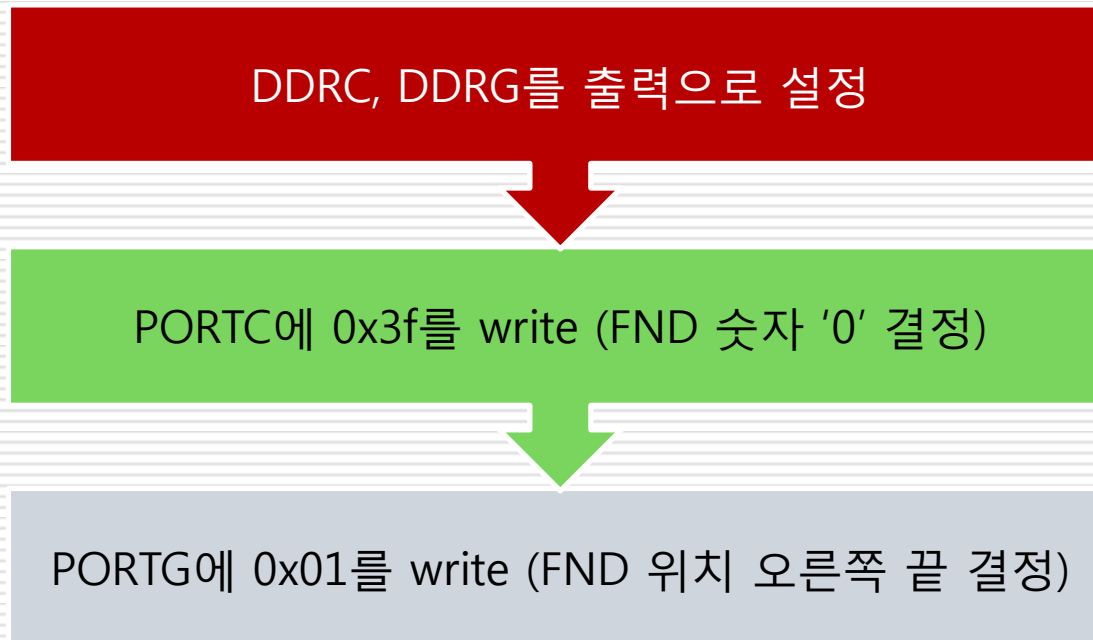
- 구동 프로그램 설계 : '0' Display (fnd\_1.c)
  - 숫자 '0'을 표시하기 위하여는 Port C에 '0'에 해당되는 값 '0x3f'를 보내면 됨
  - 맨 오른쪽 digit를 선택하려면 Port G에 '0x01'을 보내면 됨  
(주의 : Port G는 PG3 ~ PG0 의 4개 신호 중 '1'이 되는 신호가 선택됨. 왜냐하면 선택 신호 '1'이면 트랜지스터를 거치면서 반전되어 신호가 '0'이 되면서 선택이 이루어짐)



# 실습 FND-1 : GPIO로 FND 1개 표시하기

---

- 구동 프로그램 설계 : '0' Display (fnd\_1.c)



# 실습 FND-1 : GPIO로 FND 1개 표시하기

---

## □ 구동 프로그램 코딩 : '0' Display (fnd\_1\_1.c)

```
#include <avr/io.h>                                // ATmega128 register 정의

void main()
{
    DDRC = 0xff;                                     // C 포트는 모두 출력
    DDRG = 0x0f;                                     // G 포트도 4개는 출력
    PORTC = 0x3f;                                     // '0' 표현
    PORTG = 0x01;                                     // 가장 우측의 digit 선택
}
```

**더 해보기 : FND의 맨 왼쪽 digit에 '7.' 표시하기**

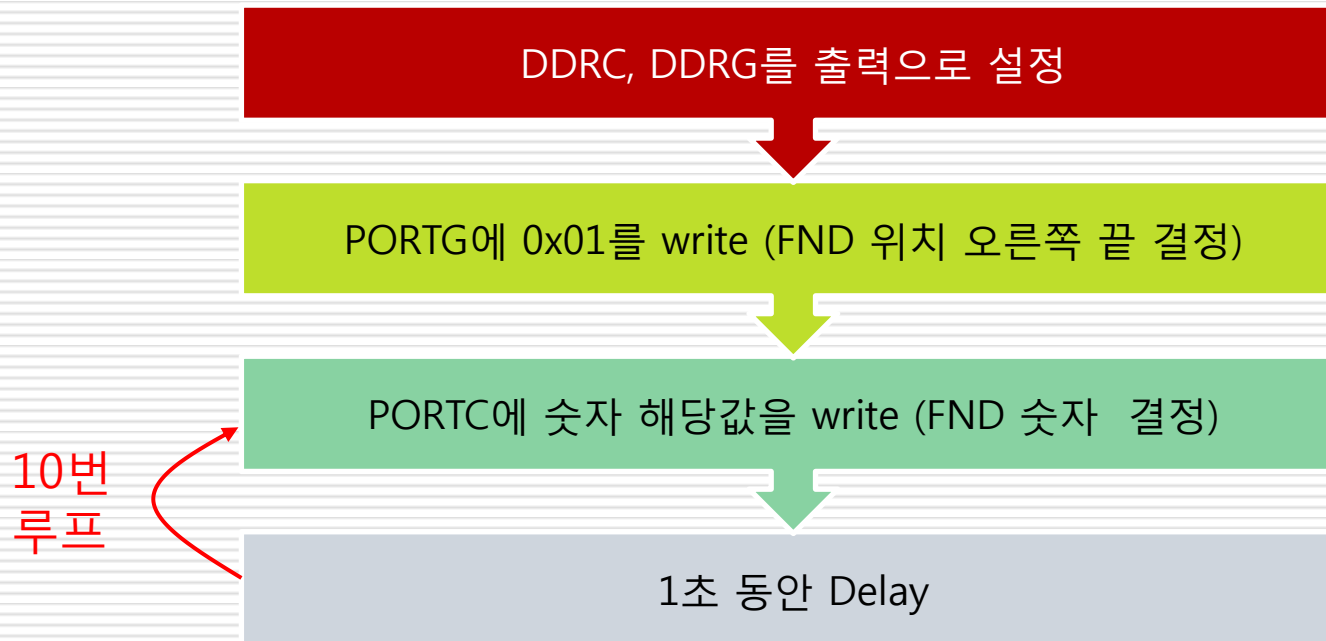
# 실습 FND-1 : GPIO로 FND 1개 표시하기

---

- 구동 프로그램 설계 : '0'~'9' Display (fnd\_1\_2.c)
  - 숫자 '0'~'9'를 표시하기 위하여는 Port C에 '0'~'9'에 해당되는 값 '0x3f'~'0x67'를 1초마다 하나씩 보내면 됨
  - 계속 같은 동작이 반복되므로 숫자 데이터 값을 배열(array)에 미리 저장해 놓았다가 Loop를 돌면서 배열을 하나씩 순서대로 꺼내서 Port C 로 출력
  - 맨 오른쪽 digit를 선택하려면 Port G에 '0x01'을 보내면 됨

# 실습 FND-1 : GPIO로 FND 1개 표시하기

□ 구동 프로그램 설계 : '0'~'9' Display (fnd\_1\_2.c)



# 실습 FND-1 : GPIO로 FND 1개 표시하기

## □ 구동 프로그램 코딩 : '0'~'9' Display (fnd\_1\_2.c)

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

```
#define F_CPU 16000000UL
```

```
#include <util/delay.h>
```

```
unsigned char digit[10] = {0x3f, 0x06,  
0x5b, 0x4f, 0x66, 0x6d, 0x7c, 0x07,  
0x7f, 0x67};
```

```
int main()
```

```
{
```

```
    int i=0;
```

```
    DDRC = 0xff;
```

```
    DDRC = 0x0f;
```

```
    PORTG = 0x01;
```

```
    for (i=0; i<10; i++)
```

```
    {
```

```
        PORTC = digit[i];
```

```
        // 숫자 '0'-'9'를 차례로 출력
```

```
        _delay_ms(1000);
```

```
    }
```

*\_delay\_ms( ) 함수 사용*

**더 해보기 : 무한히 반복하도록 수정**

# 실습 FND-2 : GPIO로 FND 4개 표시하기

---

## □ 실습 내용

- FND에 '1234' 표시하기
- 눈의 잔상효과가 나타나는 최소의 시간 찾기 (각자 해보기)

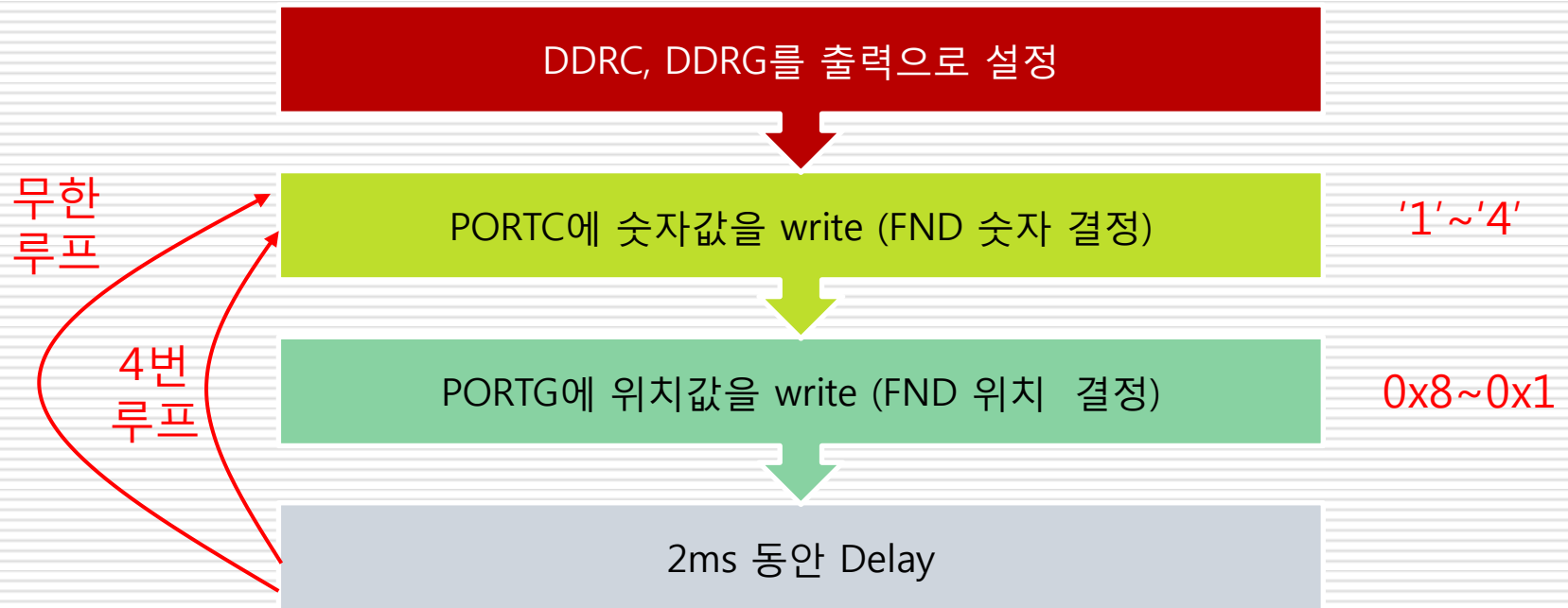
# 실습 FND-2 : GPIO로 FND 4개 표시하기

---

- 구동 프로그램 설계 : '1234' Display (fnd\_2.c)
  - 숫자 '1234'가 모두 동시에 보이게 하려면 4개의 digit을 순차적으로 디스플레이 하는 과정을 반복적으로 빠르게 수행하여야 함.
  - 즉 눈의 잔상효과를 얻기 위하여 1번의 Loop를 수행하는 시간은 최대 1/30초(약 33ms)를 넘어서는 안됨 (약 10ms로 처리)
  - '1'~'4'를 표시하기 위하여는 Port C에 '1'~'4'에 해당되는 값을 하나씩 보내면 됨
  - Digit의 위치도 왼쪽부터 한 칸씩 움직이면서 선택하려면 Port G에 '0x08' → '0x04' → '0x02' → '0x01'을 숫자와 짝을 맞추어 하나씩 보내면 됨

# 실습 FND-2 : GPIO로 FND 4개 표시하기

□ 구동 프로그램 설계 : '1234' Display (fnd\_2.c)





# 실습 FND-2 : GPIO로 FND 4개 표시하기

## □ 구동 프로그램 코딩 : '1234' Display (fnd\_2.c)

```
#include <avr/io.h>
#define F_CPU 16000000UL
#include <util/delay.h>
unsigned char digit[10] = {0x3f, 0x06,
0x5b, 0x4f, 0x66, 0x6d, 0x7c, 0x07,
0x7f, 0x67};
unsigned char fnd_sel[4] = {0x01,
0x02, 0x04, 0x08};
int main()
{
    int i=0;
    DDRC = 0xff;
    DDRG = 0x0f;
```

```
    while (1)
    {
        for (i=0; i<4; i++)
        {
            PORTC = digit[4-i];
            // 숫자 '1'-'4'를 출력
            PORTG = fnd_sel[i];
            _delay_ms(2);
        }
    }
}
```

**더 해보기 : 눈의 잔상효과 시간 찾기**

# 실습 FND-3 : 1/100초 초시계 만들기

---

## □ 실습 내용

- FND로 1/100초 디지털 초시계 만들기
- FND로 24시간 디지털 시계 만들기 (과제)

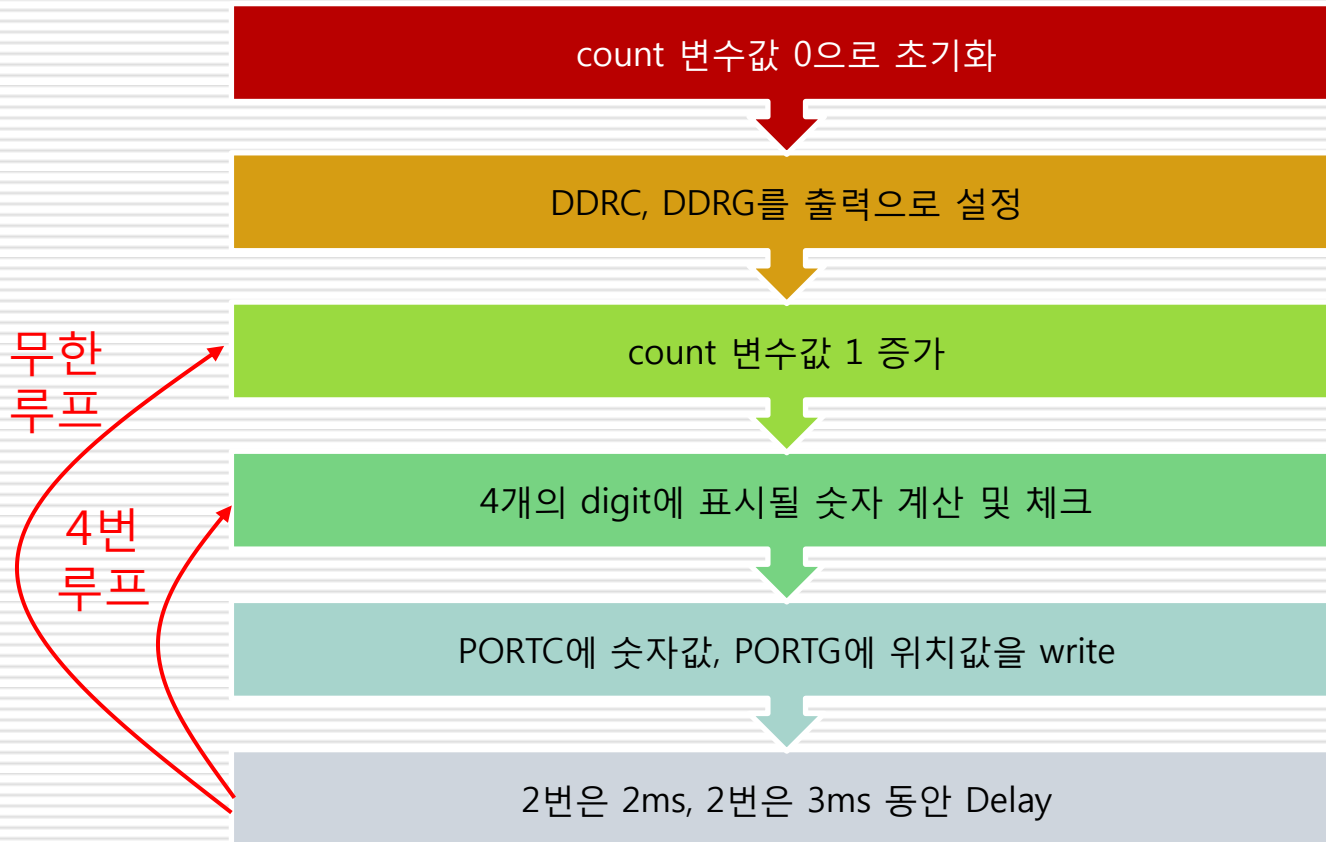
# 실습 FND-3 : 1/100초 초시계 만들기

---

- 구동 프로그램 설계 : 1/100 초 초시계(fnd\_3.c)
  - 디스플레이되는 값은 초기값 '00.00'에서 시작하여 1/100초마다 1씩 증가하는 10진 카운터처럼 동작
  - 99.99 이후는 00.00에서 다시 시작하도록 설정
  - 초기값이 0인 count 변수를 설정한 후, count를 1/100초마다 1씩 증가시킨 후 다음을 계산하고, 이 수를 디스플레이
    - FND 첫째자리 :  $(\text{count}/1000)\%10$
    - FND 둘째자리 :  $(\text{count}/100)\%10$
    - FND 셋째자리 :  $(\text{count}/10)\%10$
    - FND 넷째자리 :  $\text{count}\%10$
  - 주의 : 1/100초는 10ms인데, 4개 digit를 디스플레이하는 시간도 필요하므로 이 시간의 총합이 10ms가 되도록 설정함

# 실습 FND-3 : 1/100초 초시계 만들기

## □ 구동 프로그램 설계 : 1/100 초 초시계(fnd\_3.c)



# 실습 FND-3 : 1/100초 초시계 만들기

## □ 구동 프로그램 코딩 : 1/100 초 초시계(fnd\_3.c)

```
#include <avr/io.h>
#define F_CPU 16000000UL
#include <util/delay.h>
unsigned char digit[10] = {0x3f, 0x06,
0x5b, 0x4f, 0x66, 0x6d, 0x7c, 0x07,
0x7f, 0x67};
unsigned char fnd_sel[4] = {0x01,
0x02, 0x04, 0x08};
unsigned char fnd[4];
int main()
{
    int i=0, count=0;
    DDRC = 0xff;
    DDRG = 0x0f;
```

```
    while (1)
    {
        count++;
        if (count == 10000)
            count = 0;
        fnd[3] = (count/1000)%10;
        fnd[2] = (count/100)%10;
        fnd[1] = (count/10)%10;
        fnd[0] = count%10;
        for (i=0; i<4; i++)
        {
            PORTC = digit[fnd[i]];
            PORTG = fnd_sel[i];
            _delay_ms(2);
            if(i%2) _delay_ms(1); }
    }
```

# 실습 FND-4 : 전화번호표시기 만들기

---

## □ 실습 내용

- FND로 자동차주차용 전화번호표시기 만들기

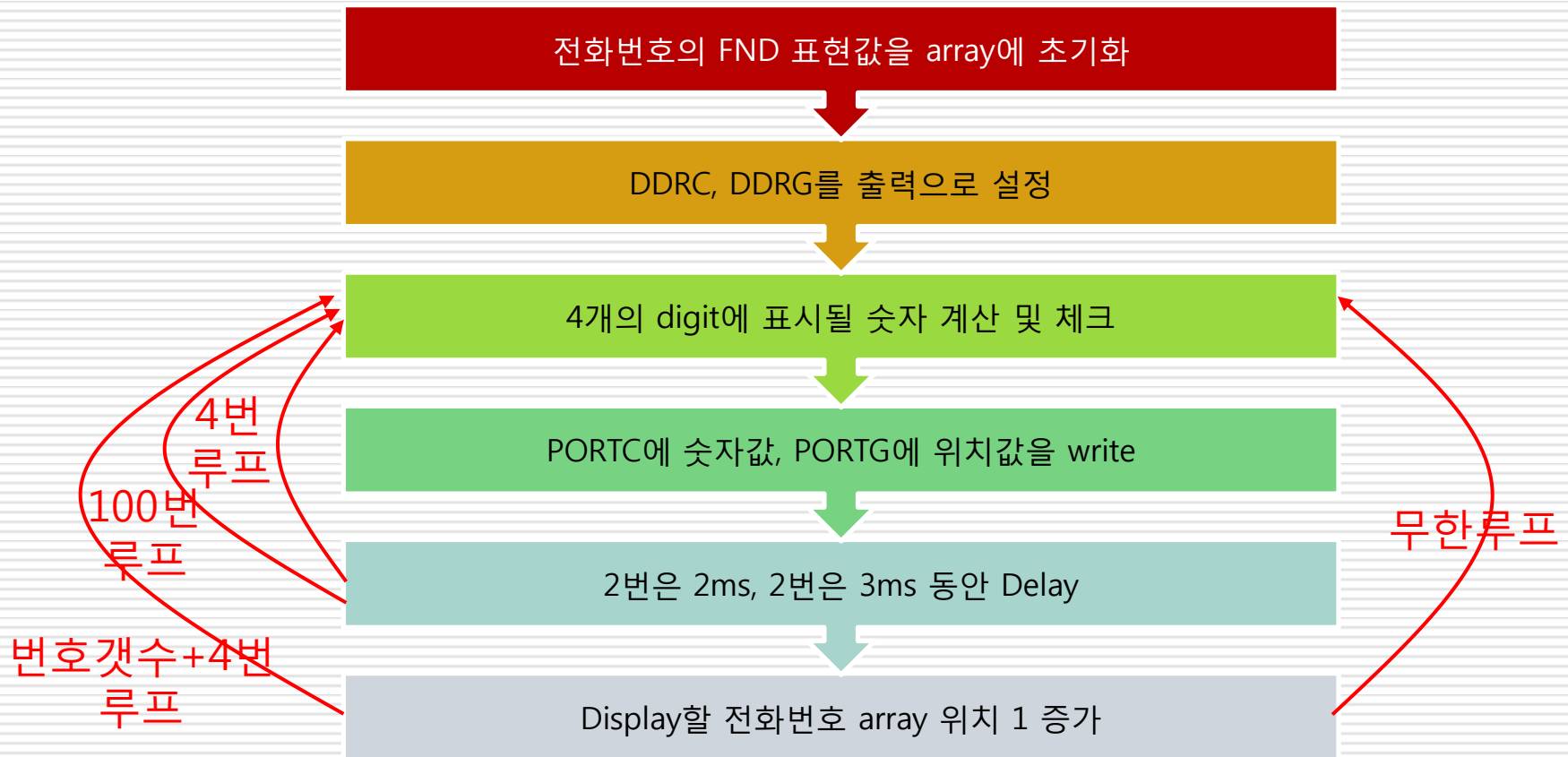
# 실습 FND-4 : 전화번호표시기 만들기

---

- 구동 프로그램 설계 : Tel. Num. Displayer (fnd\_4.c)
  - 4개의 digit에 표시할 내용(전화번호)을 먼저 결정  
(예 : 010-4016-1081) : array로 표시
  - 4개의 digit로 전화번호를 모두 표현할 수는 없으므로, 전화번호가 오른쪽에서 왼쪽으로 1초마다 왼쪽으로 물 흐르듯이 이동하면서 디스플레이되도록 프로그램함
  - 즉, 0 → 01 → 010 → 010- → 10-4 → 0-40 → -401 → 4016 → ... → 1081 → 081 → 81 → 1 → 형태로 loop 반복
  - 어떤 한 순간에 4개의 digit을 디스플레이하고, 이후 정해진 delay(0.5초) 후 디스플레이할 내용을 한 칸 이동하여 다시 4, 개의 digit를 디스플레이함

# 실습 FND-4 : 전화번호표시기 만들기

## □ 구동 프로그램 설계 : Tel. Num. Displayer (fnd\_4.c)





# 실습 FND-4 : 전화번호표시기 만들기

## □ 구동 프로그램 코딩 : Tel. Num. Displayer (fnd\_4.c)

```
#include <avr/io.h>
#define F_CPU 16000000UL
#include <util/delay.h>
unsigned char digit[21] = {0x00,
0x00, 0x00, 0x00, 0x3f, 0x06, 0x3f,
0x40, 0x66, 0x3f, 0x06, 0x7c, 0x40,
0x06, 0x3f, 0x7f, 0x06, 0x00, 0x00,
0x00, 0x00};
unsigned char fnd_sel[4] = {0x01,
0x02, 0x04, 0x08};
int main()
{
    int i, j, k;
    DDRC = 0xff;
    DDRG = 0x0f;
```

```
    while (1)
    {
        for (k=0; k<13+4; k++)
        {
            for (j=0; j<100; j++)
            {
                for (i=0; i<4; i++)
                {
                    PORTC = digit[i+k];
                    PORTG = fnd_sel[3-i];
                    _delay_ms(2);
                    if(i%2) _delay_ms(1);
                }
            }
        }
    }
}
```

# 숙제

---

- 제출 내용 : FND로 디지털 시계 만들기
  - 왼쪽 2개는 시간 표시, 오른쪽 2개는 분 표시, 시간과 분 사이에 점도 표시
  - 23시 59분까지 표시되며 이후는 다시 00시 00분부터 시작
  - 1초마다 한번씩 깜빡거리게 하기 (0.5초는 ON, 0.5초는 OFF)
- 제출 기한 : 다음 수업시간 종료 전까지
- 제출 방법 : eclass에 "학번-이름-FND.zip" 파일로 제출

# 문고 답하기

---

Q & A

