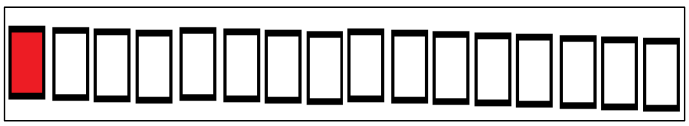
```
// 선행처리지시
#include "DSP28x_Project.h"
// Device Headerfile and Examples Include File

#define timeA 1000000
#define timeB 500000
#define timeC 500000
#define timeD 1000000
#define timeE 1000000
#define timeF 1000000
#define timeStop 500000

//   함수선언

interrupt void Xint3_isr(void);
interrupt void Xint4_isr(void);
interrupt void Xint5_isr(void);
interrupt void Xint6_isr(void);
int modeA(int);
int modeB(int);
int modeC(int);
int modeD(int);
int modeE(int);
int modeF(int);
int modeS(int);

// 시스템에서 사용할 전역 변수 선언

Uint16 Loop_cnt;
Uint16 SW1_cnt, SW2_cnt, SW3_cnt, SW4_cnt;
(스위치 4개 설정.)
volatile unsigned int i = 0;
volatile unsigned int mode = 0, stop = 0;
```

// 사용자 함수 **int modeA(int i)**

{LED가 16개이니까 I=1~16까지 값 설정.

```
        if (i == 1)
        {
        GpioDataRegs.GPCDAT.all = 32768;        DELAY_US(timeA);
        }
```



$$2^{15} = 32768$$

Table 70. GPIO Port C Data (GPCDAT) Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved		Reserved
2-0	GPIO87-GPIO64		Each bit corresponds to one GPIO port B pin (GPIO64-GPIO87) as shown in Figure 67
		0	Reading a 0 indicates that the state of the pin is currently low, irrespective of the mode the pin is configured for. Writing a 0 will force an output of 0 if the pin is configured as a GPIO output in the appropriate GPCMUX1 and GPCDIR registers; otherwise, the value is latched but not used to drive the pin.
		1	Reading a 1 indicates that the state of the pin is currently high irrespective of the mode the pin is configured for. Writing a 1 will force an output of 1 if the pin is configured as a GPIO output in the GPCMUX1 and GPCDIR registers; otherwise, the value is latched but not used to drive the pin.

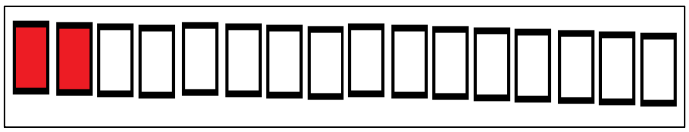
```
else if (i == 2)
        {
        GpioDataRegs.GPCDAT.all = 16384;        DELAY_US(timeA);
        }
        else if (i == 3)
        {
        GpioDataRegs.GPCDAT.all = 8192;         DELAY_US(timeA);
        }
        else if (i == 4)
        {
        GpioDataRegs.GPCDAT.all = 4096;         DELAY_US(timeA);
        }
        else if (i == 5)
        {
        GpioDataRegs.GPCDAT.all = 2048;         DELAY_US(timeA);
        }
        else if (i == 6)
        {
        GpioDataRegs.GPCDAT.all = 1024;         DELAY_US(timeA);
        }
        else if (i == 7)
        {
        GpioDataRegs.GPCDAT.all = 512;          DELAY_US(timeA);
        }
        else if (i == 8)
        {
        GpioDataRegs.GPCDAT.all = 256;
        DELAY_US(timeA);
        }
        else if (i == 9)
        {
        GpioDataRegs.GPCDAT.all = 128;
        DELAY_US(timeA);
        }
        else if (i == 10)
        {
        GpioDataRegs.GPCDAT.all = 64;           DELAY_US(timeA);
        }
        else if (i == 11)
        {
        GpioDataRegs.GPCDAT.all = 32;           DELAY_US(timeA);
        }
        else if (i == 12)
        {
        GpioDataRegs.GPCDAT.all = 16;           DELAY_US(timeA);
        }
        else if (i == 13)
        {
        GpioDataRegs.GPCDAT.all = 8;            DELAY_US(timeA);
        }
        else if (i == 14)
        {
        GpioDataRegs.GPCDAT.all = 4;            DELAY_US(timeA);
        }
        else if (i == 15)
        {
        GpioDataRegs.GPCDAT.all = 2;            DELAY_US(timeA);
        }
```

```
        else if (i == 16)
        {
        GpioDataRegs.GPCDAT.all = 1;        DELAY_US(timeA);
```

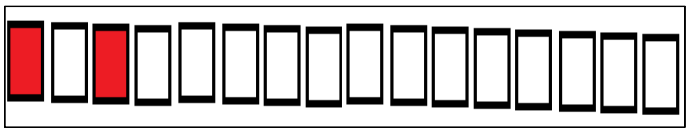
```
        }
        return 1;
}
```

코드 300줄	코드1000줄
반복문과 시프트 연산으로 표현.	레지스터GPCDAT로, 16개의 LED를 2진수로 표현. 총 16개이기에. $2^{15} \sim 2^{0}$의 덧셈으로 계산.

```
int modeB(int i)
{
        if (i == 1)
        {
        GpioDataRegs.GPCDAT.all = 32768;                DELAY_US(timeB);
        }
        else if (i == 2)
        {
        GpioDataRegs.GPCDAT.all = 49152;                DELAY_US(timeB);
        }
```



$2^{15} + 2^{14} = 49152 = 32768 + 16384$



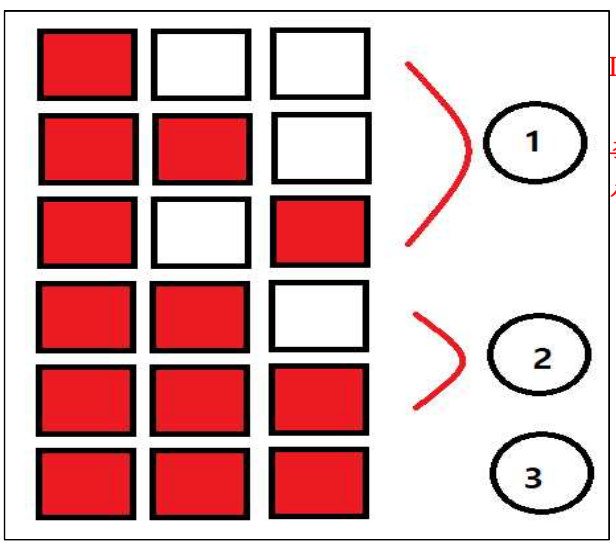
$2^{15} + 2^{13} = 40960 = 32768 + 8192$

```
        else if (i == 3)
        {
                GpioDataRegs.GPCDAT.all = 40960; DELAY_US(timeB);
        }
else if (i == 4)
        {
                GpioDataRegs.GPCDAT.all = 36864; DELAY_US(timeB);
        }
        else if (i == 5)
        {
                GpioDataRegs.GPCDAT.all = 34816; DELAY_US(timeB);
        }
        else if (i == 6)
        {
                GpioDataRegs.GPCDAT.all = 33792; DELAY_US(timeB);
        }
        else if (i == 7)
        {
                GpioDataRegs.GPCDAT.all = 33280; DELAY_US(timeB);
        }
        else if (i == 8)
        {
                GpioDataRegs.GPCDAT.all = 33024; DELAY_US(timeB);
        }
        else if (i == 9)
        {
                GpioDataRegs.GPCDAT.all = 32896; DELAY_US(timeB);
        }
        else if (i == 10)
        {
                GpioDataRegs.GPCDAT.all = 32832; DELAY_US(timeB);
        }
        else if (i == 11)
        {
                GpioDataRegs.GPCDAT.all = 32800; DELAY_US(timeB);
        }
        else if (i == 12)
        {
                GpioDataRegs.GPCDAT.all = 32784; DELAY_US(timeB);
        }
        else if (i == 13)
        {
                GpioDataRegs.GPCDAT.all = 32776; DELAY_US(timeB);
        }
        else if (i == 14)
        {
                GpioDataRegs.GPCDAT.all = 32772; DELAY_US(timeB);
        }
        else if (i == 15)
        {
                GpioDataRegs.GPCDAT.all = 32770; DELAY_US(timeB);
        }
        else if (i == 16)
        {
                GpioDataRegs.GPCDAT.all = 32769; DELAY_US(timeB);
        }
        else if (i == 17)
        {
                GpioDataRegs.GPCDAT.all = 49152; DELAY_US(timeB);
        }
        else if (i == 18)
        {
                GpioDataRegs.GPCDAT.all = 57344; DELAY_US(timeB);
        }
        else if (i == 19)
        {
                GpioDataRegs.GPCDAT.all = 53248; DELAY_US(timeB);
        }
        else if (i == 20)
        {
                GpioDataRegs.GPCDAT.all = 51200; DELAY_US(timeB);
        }
        else if (i == 21)
        {
                GpioDataRegs.GPCDAT.all = 50176; DELAY_US(timeB);
        }
        else if (i == 22)
        {
                GpioDataRegs.GPCDAT.all = 49664; DELAY_US(timeB);
        }
        else if (i == 23)
        {
                GpioDataRegs.GPCDAT.all = 49408; DELAY_US(timeB);
        }
        else if (i == 24)
        {
                GpioDataRegs.GPCDAT.all = 49280; DELAY_US(timeB);
        }
        else if (i == 25)
        {
                GpioDataRegs.GPCDAT.all = 49216; DELAY_US(timeB);
        }
        else if (i == 26)
        {
```

```
                GpioDataRegs.GPCDAT.all = 49184; DELAY_US(timeB);
}
else if (i == 27)
{
                GpioDataRegs.GPCDAT.all = 49168; DELAY_US(timeB);
}
else if (i == 28)
{
                GpioDataRegs.GPCDAT.all = 49160; DELAY_US(timeB);
}
else if (i == 29)
{
                GpioDataRegs.GPCDAT.all = 49156; DELAY_US(timeB);
}
else if (i == 30)
{
                GpioDataRegs.GPCDAT.all = 49154; DELAY_US(timeB);
}
else if (i == 31)
{
                GpioDataRegs.GPCDAT.all = 49153; DELAY_US(timeB);
}
else if (i == 32)
{
                GpioDataRegs.GPCDAT.all = 57344; DELAY_US(timeB);
}
else if (i == 33)
{
                GpioDataRegs.GPCDAT.all = 61440; DELAY_US(timeB);
}
else if (i == 34)
{
                GpioDataRegs.GPCDAT.all = 59392; DELAY_US(timeB);
}
else if (i == 35)
{
                GpioDataRegs.GPCDAT.all = 58368; DELAY_US(timeB);
}
else if (i == 36)
{
                GpioDataRegs.GPCDAT.all = 57856; DELAY_US(timeB);
}
else if (i == 37)
{
                GpioDataRegs.GPCDAT.all = 57600; DELAY_US(timeB);
}
else if (i == 38)
{
                GpioDataRegs.GPCDAT.all = 57472; DELAY_US(timeB);
}
else if (i == 39)
{
                GpioDataRegs.GPCDAT.all = 57408; DELAY_US(timeB);
}
else if (i == 40)
{
                GpioDataRegs.GPCDAT.all = 57376; DELAY_US(timeB);
}
else if (i == 41)
{
                GpioDataRegs.GPCDAT.all = 57360; DELAY_US(timeB);
}
else if (i == 42)
{
                GpioDataRegs.GPCDAT.all = 57352; DELAY_US(timeB);
}
else if (i == 43)
{
                GpioDataRegs.GPCDAT.all = 57348; DELAY_US(timeB);
}
else if (i == 44)
{
                GpioDataRegs.GPCDAT.all = 57346; DELAY_US(timeB);
}
else if (i == 45)
{
                GpioDataRegs.GPCDAT.all = 57345; DELAY_US(timeB);
}
else if (i == 46)
{
                GpioDataRegs.GPCDAT.all = 61440; DELAY_US(timeB);
}
else if (i == 47)
{
                GpioDataRegs.GPCDAT.all = 63488; DELAY_US(timeB);
}
else if (i == 48)
{
                GpioDataRegs.GPCDAT.all = 62464; DELAY_US(timeB);
}
else if (i == 49)
{
                GpioDataRegs.GPCDAT.all = 61952; DELAY_US(timeB);
}
else if (i == 50)
{
                GpioDataRegs.GPCDAT.all = 61696; DELAY_US(timeB);
}
else if (i == 51)
{
                GpioDataRegs.GPCDAT.all = 61568; DELAY_US(timeB);
}
else if (i == 52)
{
                GpioDataRegs.GPCDAT.all = 61504; DELAY_US(timeB);
}
else if (i == 53)
{
                GpioDataRegs.GPCDAT.all = 61472; DELAY_US(timeB);
}
else if (i == 54)
{
                GpioDataRegs.GPCDAT.all = 61456; DELAY_US(timeB);
}
else if (i == 55)
{
                GpioDataRegs.GPCDAT.all = 61448; DELAY_US(timeB);
}
else if (i == 56)
{
                GpioDataRegs.GPCDAT.all = 61444; DELAY_US(timeB);
}
else if (i == 57)
{
                GpioDataRegs.GPCDAT.all = 61442; DELAY_US(timeB);
}
else if (i == 58)
{
                GpioDataRegs.GPCDAT.all = 61441; DELAY_US(timeB);
}
else if (i == 59)
{
                GpioDataRegs.GPCDAT.all = 63488; DELAY_US(timeB);
}
else if (i == 60)
{
                GpioDataRegs.GPCDAT.all = 64512; DELAY_US(timeB);
}
else if (i == 61)
{
                GpioDataRegs.GPCDAT.all = 64000; DELAY_US(timeB);
}
else if (i == 62)
{
                GpioDataRegs.GPCDAT.all = 63744; DELAY_US(timeB);
}
else if (i == 63)
{
                GpioDataRegs.GPCDAT.all = 63616; DELAY_US(timeB);
}
else if (i == 64)
{
                GpioDataRegs.GPCDAT.all = 63552; DELAY_US(timeB);
}
else if (i == 65)
{
                GpioDataRegs.GPCDAT.all = 63520; DELAY_US(timeB);
}
else if (i == 66)
{
                GpioDataRegs.GPCDAT.all = 63504; DELAY_US(timeB);
}
else if (i == 67)
{
```

```
                GpioDataRegs.GPCDAT.all = 63496; DELAY_US(timeB);
}
else if (i == 68)
{
                GpioDataRegs.GPCDAT.all = 63492; DELAY_US(timeB);
}
else if (i == 69)
{
                GpioDataRegs.GPCDAT.all = 63490; DELAY_US(timeB);
}
else if (i == 70)
{
                GpioDataRegs.GPCDAT.all = 63489; DELAY_US(timeB);
}
else if (i == 71)
{
                GpioDataRegs.GPCDAT.all = 64512; DELAY_US(timeB);
}
else if (i == 72)
{
                GpioDataRegs.GPCDAT.all = 65024; DELAY_US(timeB);
}
else if (i == 73)
{
                GpioDataRegs.GPCDAT.all = 64768; DELAY_US(timeB);
}
else if (i == 74)
{
                GpioDataRegs.GPCDAT.all = 64640; DELAY_US(timeB);
}
else if (i == 75)
{
                GpioDataRegs.GPCDAT.all = 64576; DELAY_US(timeB);
}
else if (i == 76)
{
                GpioDataRegs.GPCDAT.all = 64544; DELAY_US(timeB);
}
else if (i == 77)
{
                GpioDataRegs.GPCDAT.all = 64528; DELAY_US(timeB);
}
else if (i == 78)
{
                GpioDataRegs.GPCDAT.all = 64520; DELAY_US(timeB);
}
else if (i == 79)
{
                GpioDataRegs.GPCDAT.all = 64516; DELAY_US(timeB);
}
else if (i == 80)
{
                GpioDataRegs.GPCDAT.all = 64514; DELAY_US(timeB);
}
else if (i == 81)
{
                GpioDataRegs.GPCDAT.all = 64513; DELAY_US(timeB);
}
else if (i == 82)
{
                GpioDataRegs.GPCDAT.all = 65024; DELAY_US(timeB);
}
else if (i == 83)
{
                GpioDataRegs.GPCDAT.all = 65280; DELAY_US(timeB);
}
else if (i == 84)
{
                GpioDataRegs.GPCDAT.all = 65152; DELAY_US(timeB);
}
else if (i == 85)
{
                GpioDataRegs.GPCDAT.all = 65088; DELAY_US(timeB);
}
else if (i == 86)
{
                GpioDataRegs.GPCDAT.all = 65056; DELAY_US(timeB);
}
else if (i == 87)
{
                GpioDataRegs.GPCDAT.all = 65040; DELAY_US(timeB);
}
else if (i == 88)
{
                GpioDataRegs.GPCDAT.all = 65032; DELAY_US(timeB);
}
else if (i == 89)
{
                GpioDataRegs.GPCDAT.all = 65028; DELAY_US(timeB);
}
else if (i == 90)
{
                GpioDataRegs.GPCDAT.all = 65026; DELAY_US(timeB);
}
else if (i == 91)
{
                GpioDataRegs.GPCDAT.all = 65025; DELAY_US(timeB);
}
else if (i == 92)
{
                GpioDataRegs.GPCDAT.all = 65280; DELAY_US(timeB);
}
else if (i == 93)
{
                GpioDataRegs.GPCDAT.all = 65408; DELAY_US(timeB);
}
else if (i == 94)
{
                GpioDataRegs.GPCDAT.all = 65344; DELAY_US(timeB);
}
else if (i == 95)
{
                GpioDataRegs.GPCDAT.all = 65312; DELAY_US(timeB);
}
else if (i == 96)
{
                GpioDataRegs.GPCDAT.all = 65296; DELAY_US(timeB);
}
else if (i == 97)
{
                GpioDataRegs.GPCDAT.all = 65288; DELAY_US(timeB);
}
else if (i == 98)
{
                GpioDataRegs.GPCDAT.all = 65284; DELAY_US(timeB);
}
else if (i == 99)
{
                GpioDataRegs.GPCDAT.all = 65282; DELAY_US(timeB);
}
else if (i == 100)
{
                GpioDataRegs.GPCDAT.all = 65281; DELAY_US(timeB);
}
else if (i == 101)
{
                GpioDataRegs.GPCDAT.all = 65408; DELAY_US(timeB);
}
else if (i == 102)
{
                GpioDataRegs.GPCDAT.all = 65472; DELAY_US(timeB);
}
else if (i == 103)
{
                GpioDataRegs.GPCDAT.all = 65440; DELAY_US(timeB);
}
else if (i == 104)
{
                GpioDataRegs.GPCDAT.all = 65424; DELAY_US(timeB);
}
else if (i == 105)
{
                GpioDataRegs.GPCDAT.all = 65416; DELAY_US(timeB);
}
else if (i == 106)
{
                GpioDataRegs.GPCDAT.all = 65412; DELAY_US(timeB);
}
else if (i == 107)
{
                GpioDataRegs.GPCDAT.all = 65410; DELAY_US(timeB);
}
else if (i == 108)
{
```

```
                GpioDataRegs.GPCDAT.all = 65409; DELAY_US(timeB);
        }
        else if (i == 109)
        {
                GpioDataRegs.GPCDAT.all = 65472; DELAY_US(timeB);
        }
        else if (i == 110)
        {
                GpioDataRegs.GPCDAT.all = 65504; DELAY_US(timeB);
        }
        else if (i == 111)
        {
                GpioDataRegs.GPCDAT.all = 65488; DELAY_US(timeB);
        }
        else if (i == 112)
        {
                GpioDataRegs.GPCDAT.all = 65480; DELAY_US(timeB);
        }
        else if (i == 113)
        {
                GpioDataRegs.GPCDAT.all = 65476; DELAY_US(timeB);
        }
        else if (i == 114)
        {
                GpioDataRegs.GPCDAT.all = 65474; DELAY_US(timeB);
        }
        else if (i == 115)
        {
                GpioDataRegs.GPCDAT.all = 65473; DELAY_US(timeB);
        }
        else if (i == 116)
        {
                GpioDataRegs.GPCDAT.all = 65504; DELAY_US(timeB);
        }
        else if (i == 117)
        {
                GpioDataRegs.GPCDAT.all = 65520; DELAY_US(timeB);
        }
        else if (i == 118)
        {
                GpioDataRegs.GPCDAT.all = 65512; DELAY_US(timeB);
        }
        else if (i == 119)
        {
                GpioDataRegs.GPCDAT.all = 65508; DELAY_US(timeB);
        }
        else if (i == 120)
        {
                GpioDataRegs.GPCDAT.all = 65506; DELAY_US(timeB);
        }
        else if (i == 121)
        {
                GpioDataRegs.GPCDAT.all = 65505; DELAY_US(timeB);
        }
        else if (i == 122)
        {
                GpioDataRegs.GPCDAT.all = 65520; DELAY_US(timeB);
        }
        else if (i == 123)
        {
                GpioDataRegs.GPCDAT.all = 65528; DELAY_US(timeB);
        }
        else if (i == 124)
        {
                GpioDataRegs.GPCDAT.all = 65524; DELAY_US(timeB);
        }
        else if (i == 125)
        {
                GpioDataRegs.GPCDAT.all = 65522; DELAY_US(timeB);
        }
        else if (i == 126)
        {
                GpioDataRegs.GPCDAT.all = 65521; DELAY_US(timeB);
        }
        else if (i == 127)
        {
                GpioDataRegs.GPCDAT.all = 65528; DELAY_US(timeB);
        }
        else if (i == 128)
        {
                GpioDataRegs.GPCDAT.all = 65532; DELAY_US(timeB);
        }
        else if (i == 129)
        {
                GpioDataRegs.GPCDAT.all = 65530; DELAY_US(timeB);
        }
        else if (i == 130)
        {
                GpioDataRegs.GPCDAT.all = 65529; DELAY_US(timeB);
        }
        else if (i == 131)
        {
                GpioDataRegs.GPCDAT.all = 65532; DELAY_US(timeB);
        }
        else if (i == 132)
        {
                GpioDataRegs.GPCDAT.all = 65534; DELAY_US(timeB);
        }
        else if (i == 133)
        {
        GpioDataRegs.GPCDAT.all = 65533; DELAY_US(timeB);
        }
        else if (i == 134)
        {
        GpioDataRegs.GPCDAT.all = 65534; DELAY_US(timeB);
        }
        else if (i == 135)
        {
        GpioDataRegs.GPCDAT.all = 65535; DELAY_US(timeB);
        }
        else if (i == 136)
        {
        GpioDataRegs.GPCDAT.all = 65535; DELAY_US(timeB);
        }

        return 1;
}
```



I=136까지 실행한 이유 : 1+2+3+...+15+16=136이기 때문에.

즉, led가 3개이면, 3+2+1=6으로 6가지의 경우에 대한 조건문을 설정 해야한다.

int modeC(int i)

```
{
        if (i == 1)
        {
GpioDataRegs.GPCDAT.all = 43690;    DELAY_US(timeC);
        }
        else if (i == 2)
        {
GpioDataRegs.GPCDAT.all = 21845;    DELAY_US(timeC);
        }

        return 1;
}
```

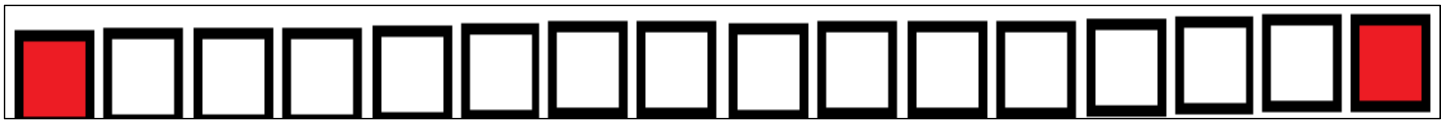
GpioDataRegs.GPCDAT.all = **43690;** $2^{15}+2^{13}+2^{11}+2^{9}+2^{7}+2^{5}+2^{3}+2^{1}=43690$

GpioDataRegs.GPCDAT.all = **21845;** $2^{16}+2^{14}+2^{12}+2^{10}+2^{8}+2^{6}+2^{4}+2^{2}=65535-43690=21845$

int modeD(int i)

```
{
        if (i == 1)
        {
GpioDataRegs.GPCDAT.all = 32769;    DELAY_US(timeD);
        }

        else if (i == 2)
        {
GpioDataRegs.GPCDAT.all = 16386;    DELAY_US(timeD);
        }

        else if (i == 3)
        {
GpioDataRegs.GPCDAT.all = 8196;     DELAY_US(timeD);
        }

        else if (i == 4)
        {
GpioDataRegs.GPCDAT.all = 4104;     DELAY_US(timeD);
        }

        else if (i == 5)
        {
                GpioDataRegs.GPCDAT.all = 2064;
DELAY_US(timeD);
        }

        else if (i == 6)
        {
GpioDataRegs.GPCDAT.all = 1056;     DELAY_US(timeD);
        }

        else if (i == 7)
        {
GpioDataRegs.GPCDAT.all = 576;      DELAY_US(timeD);
        }

        else if (i == 8)
        {
                GpioDataRegs.GPCDAT.all = 384;
DELAY_US(timeD);
        }

        else if (i == 9)
        {
GpioDataRegs.GPCDAT.all = 384;      DELAY_US(timeD);
        }

        else if (i == 10)
        {
GpioDataRegs.GPCDAT.all = 576;      DELAY_US(timeD);
        }

        else if (i == 11)
        {
GpioDataRegs.GPCDAT.all = 1056;     DELAY_US(timeD);
        }

        else if (i == 12)
        {
GpioDataRegs.GPCDAT.all = 2064;     DELAY_US(timeD);
        }

        else if (i == 13)
        {
GpioDataRegs.GPCDAT.all = 4104;     DELAY_US(timeD);
        }

        else if (i == 14)
        {
GpioDataRegs.GPCDAT.all = 8196;     DELAY_US(timeD);
        }

        else if (i == 15)
        {
GpioDataRegs.GPCDAT.all = 16386;    DELAY_US(timeD);
        }

        else if (i == 16)
        {
GpioDataRegs.GPCDAT.all = 32769;    DELAY_US(timeD);
        }

        return 1;
}
```

GpioDataRegs.GPCDAT.all = 32769; $2^{15}+2^{0}=1+32768=32769$



- “Mode 4”의 경우, led가 왼쪽, 오른쪽에서 출발하여 중앙으로 한 칸씩 이동하는 과정을 표현한 것이다.

(L, R) = (i번째 led, j번째 led) = (16,1), (15, 2), (14,3)…… (8, 7)

int modeE(int i)

```
{
        if (i == 1)
        {
        GpioDataRegs.GPCDAT.all = 32768; DELAY_US(timeE);
        }
        else if (i == 2)
        {
        GpioDataRegs.GPCDAT.all = 24576; DELAY_US(timeE);
        }
        else if (i == 3)
        {
        GpioDataRegs.GPCDAT.all = 7168; DELAY_US(timeE);
        }
        else if (i == 4)
        {
        GpioDataRegs.GPCDAT.all = 960; DELAY_US(timeE);
        }
        else if (i == 5)
        {
        GpioDataRegs.GPCDAT.all = 62; DELAY_US(timeE);
        }
        else if (i == 6)
        {
        GpioDataRegs.GPCDAT.all = 1; DELAY_US(timeE);
        }

        return 1;
}
```

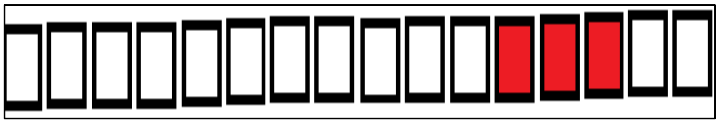
- 점등되는 LED의 개수가 5개까지 증가하며 우측으로 이동하는 것이다.

int modeF(int i)

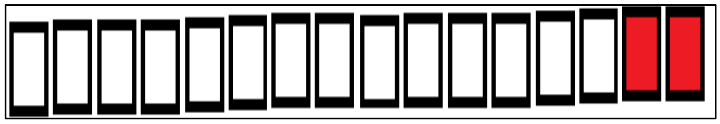
```
{
        if (i == 1)
        {
GpioDataRegs.GPCDAT.all = 32768;    DELAY_US(timeF);
        }

        else if (i == 2)
        {
GpioDataRegs.GPCDAT.all = 24576;    DELAY_US(timeF);
        }

        else if (i == 3)
        {
GpioDataRegs.GPCDAT.all = 7168;     DELAY_US(timeF);
        }

        else if (i == 4)
        {
GpioDataRegs.GPCDAT.all = 768;      DELAY_US(timeF);
        }

        else if (i == 5)
        {       2^7 = 128
GpioDataRegs.GPCDAT.all = 128;      DELAY_US(timeF);
        }

        else if (i == 6)
        {
        GpioDataRegs.GPCDAT.all = 96;       DELAY_US(timeF);
        }

        else if (i == 7)
        {
        GpioDataRegs.GPCDAT.all = 28;       DELAY_US(timeF);
        }

        else if (i == 8)
        {
        GpioDataRegs.GPCDAT.all = 6;        DELAY_US(timeF);
        }
        return 1;
}
```

$2^{14}+2^{13}=24576$

else if (i == 7) : GpioDataRegs.GPCDAT.all = 28;}



else if (i == 8) : GpioDataRegs.GPCDAT.all = 6;}



$i=8$의 경우, “i=7”의 경우 led가 3개가 켜지기에, 2개가 켜져야 하고, 그 값은 $2^{1}+2^{0}$=3이 되어야 하는데, 6으로 잘못 설정되어 있다고 분석하였습니다.

int modeS(int i)

```
{
        if (i == 1) {
GpioDataRegs.GPCDAT.all = 32768;     DELAY_US(timeStop);
        }
        else if (i == 2) {
GpioDataRegs.GPCDAT.all = 0;     DELAY_US(timeStop);
        }
}
```

초기조건 : #define timeStop 500000

따라서 "스위치 C"를 누르면, 시간 간격은 500ms이고, 아래의 led 번갈아 점등하여 정지 상태가 표현됩니다.

정지는 **"GpioDataRegs.GPCDAT.all = 0;"**으로 표현.

```
//   메인함수- 시작
void main(void)
{
        // Step 1. Disable Global Interrupt
        DINT;
        ====================================
        // Step 2. 시스템 컨트롤 초기화:
        InitSysCtrl();
==================================
        // Step 3. 인터럽트초기화:
        InitPieCtrl();
        IER = 0x0000;
        IFR = 0x0000;
        InitPieVectTable();

        // Vector Remapping
        EALLOW;
        PieVectTable.XINT3 = &Xint3_isr;
        PieVectTable.XINT4 = &Xint4_isr;
        PieVectTable.XINT5 = &Xint5_isr;
        PieVectTable.XINT6 = &Xint6_isr;
        EDIS;
```

* **DINT** : 칩의 활성 여부를 결정하는 전역 인터럽트 스위치를 Off하고, 칩의 비정상적인 작동을 방지한다.

* **InitSysCtrl** : 시스템 컨트롤을 초기화한다. 이 함수가 하는 역할은 다음과 같다.

(1) 시스템에 이상이 생기면 시스템을 리셋하는 Watchdog기능을 disable한다.
(2) PLLCR 레지스터를 설정하여 적당한 SYSCLKOUT을 생성한다.
(3) 프리스케일러를 설정하여 고속 주변장치 클럭과 저속 주변장치 클럭을 조정한다.
(4) 클럭을 ON하여 주변장치에 공급한다.

* **InitPieCtrl** : 함수를 통해 PIE Register을 초기화하여 PIE를 disable한다.

* **EALLOW-EDIS** : DSP에서 일정 Protected 영역에 값을 쓰기 위해서는 Protect를 해지하고 다시 Protect를 해주는 작업이 필요하다. HISPCP 레지스터는 Protected 영역에 있으므로 레지스터를 설정하기 위해서는 EALLOW로 보호설정을 풀어주고, EDIS로 보호를 해주어야 한다.

* **"IER = 0x0000; IFR = 0x0000;"** = 0x0000을 통하여 인터럽트가 없는 Register의 상태를 만들어 발생할 수 있는 오류를 방지하기 위해 사용한다.

소스1 내용) T1에서 정해놓은 인터럽트의 기본 루틴을 설정하기 위해, InitPieVectTable(); 함수를 이용한다.

PieVectTable.XINT3 = &Xint3_isr;

이것은 기본으로 설정 된 인터럽트 벡터를 변경해주는 코드이고, 위의 코드는 외부 인터럽트 3이 걸렸을 때, Xint3_isr이라는 ISR함수를 실행하도록 설정한 것이다. 이를 외부인터럽트 4, 5, 6에 대해서도 지정하였다.

```
// Step 4. GPIO 초기화
        EALLOW;
        GpioCtrlRegs.GPBMUX1.bit.GPIO44 = 0;
// 핀기능선택: GPIO44
        GpioCtrlRegs.GPBMUX1.bit.GPIO45 = 0;
// 핀기능선택: GPIO45
        GpioCtrlRegs.GPBMUX1.bit.GPIO46 = 0;
// 핀기능선택: GPIO46
        GpioCtrlRegs.GPBMUX1.bit.GPIO47 = 0;
// 핀기능선택: GPIO47
        GpioCtrlRegs.GPBDIR.bit.GPIO44 = 0;
// GPIO44 입출력선택: Input
        GpioCtrlRegs.GPBDIR.bit.GPIO45 = 0;
// GPIO45 입출력선택: Input
        GpioCtrlRegs.GPBDIR.bit.GPIO46 = 0;
// GPIO46 입출력선택: Input
        GpioCtrlRegs.GPBDIR.bit.GPIO47 = 0;
// GPIO47 입출력선택: Input

        GpioCtrlRegs.GPCMUX1.all = 0x00000000;
// GPIO64-GPIO79, GPIO 기능으로설정
        GpioCtrlRegs.GPCDIR.all = 0x0000FFFF;
// GPI064-GPIO79, 출력으로설정
        GpioDataRegs.GPCDAT.all = 0x0000FFFF;
        EDIS;
```

GPIO44 / XA4	External General-Purpose I/O Port	TACT Switch 4	EMIF Port (XA4)
GPIO45 / XA5	External General-Purpose I/O Port	TACT Switch 3	EMIF Port (XA5)
GPIO46 / XA6	External General-Purpose I/O Port	TACT Switch 2	EMIF Port (XA6)
GPIO47 / XA7	External General-Purpose I/O Port	TACT Switch 1	EMIF Port (XA7)

GPBQSEL2	0x6F94	2	GPIOB Qualifier Select 2 Register (GPIO48
GPBMUX1	0x6F96	2	GPIO B MUX 1 Register (GPIO32 to 47)
GPBMUX2	0x6F98	2	GPIO B MUX 2 Register (GPIO48 to 63)

소스1) GPIO Pin을 외부인터럽트로 지정하기 이전에 GPIO 설정을 진행하여야 한다.
input의 기능이 필요하므로 GPIO pin을 입력으로 설정하고, Led에 연결되는 Pin인 GPIO 64 - 79를 출력으로 설정한다.

GpioCtrlRegs.GPBDIR.bit.GPIO46 = 0;

Table 63. GPIO Port B Direction (GPBDIR) Register Field Descriptions

Bits	Field	Value	Description (1)
31-0	GPIO63-GPIO32		Controls direction of GPIO pin when GPIO mode is selected. Reading the register returns the current value of the register setting
		0	Configures the GPIO pin as an input. (default)
		1	Configures the GPIO pin as an output

(1) This register is EALLOW protected. See Section 7.2 for more information.

Figure 60. GPIO Port B Direction (GPBDIR) Register

31	30	29	28	27	26	25	24
GPIO63	GPIO62	GPIO61	GPIO60	GPIO59	GPIO58	GPIO57	GPIO56
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
GPIO55	GPIO54	GPIO53	GPIO52	GPIO51	GPIO50	GPIO49	GPIO48
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
GPIO47	GPIO46	GPIO45	GPIO44	GPIO43	GPIO42	GPIO41	GPIO40
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
GPIO39	GPIO38	GPIO37	GPIO36	GPIO35	GPIO34	GPIO33	GPIO32
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

GpioCtrlRegs.GPCDIR.all = 0x0000FFFF;

Table 64. GPIO Port C Direction (GPCDIR) Register Field Descriptions

Bits	Field	Value	Description (1)
31-0	GPIO87-GPIO64		Controls direction of GPIO pin when GPIO mode is selected. Reading the register returns the current value of the register setting
		0	Configures the GPIO pin as an input. (default)
		1	Configures the GPIO pin as an output

(1) This register is EALLOW protected. See Section 7.2 for more information.

Figure 61. GPIO Port C Direction (GPCDIR) Register

31							24
Reserved							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
GPIO87	GPIO86	GPIO85	GPIO84	GPIO83	GPIO82	GPIO81	GPIO80
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
GPIO79	GPIO78	GPIO77	GPIO76	GPIO75	GPIO74	GPIO73	GPIO72
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
GPIO71	GPIO70	GPIO69	GPIO68	GPIO67	GPIO66	GPIO65	GPIO64
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

```
EALLOW;
        GpioCtrlRegs.GPBCTRL.bit.QUALPRD1 = 0xFF;
  // (GPIO40~GPIO47) Qual period 설정
        GpioCtrlRegs.GPBQSEL1.bit.GPIO44 = 2;
    // Qualification using 6 samples
        GpioCtrlRegs.GPBQSEL1.bit.GPIO45 = 2;
// Qualification using 6 samples
        GpioCtrlRegs.GPBQSEL1.bit.GPIO46 = 2;
// Qualification using 6 samples
        GpioCtrlRegs.GPBQSEL1.bit.GPIO47 = 2;
// Qualification using 6 samples
        EDIS;
```

Table 57. GPIO Port B Qualification Control (GPBCTRL) Register Field Descriptions

Bits	Field	Value	Description (1)
31-24	QUALPRD3		Specifies the sampling period for pins GPIO56 to GPIO63
		0x00	Sampling Period = $T_{SYSCLKOUT}$ (2)
		0x01	Sampling Period = 2 × $T_{SYSCLKOUT}$
		0x02	Sampling Period = 4 × $T_{SYSCLKOUT}$
		...	...
		0xFF	Sampling Period = 510 × $T_{SYSCLKOUT}$
23-16	QUALPRD2		Specifies the sampling period for pins GPIO48 to GPIO55
		0x00	Sampling Period = $T_{SYSCLKOUT}$ (2)
		0x01	Sampling Period = 2 × $T_{SYSCLKOUT}$
		0x02	Sampling Period = 4 × $T_{SYSCLKOUT}$
		...	...
		0xFF	Sampling Period = 510 × $T_{SYSCLKOUT}$
15-8	QUALPRD1		Specifies the sampling period for pins GPIO40 to GPIO47
		0x00	Sampling Period = $T_{SYSCLKOUT}$ (2)
		0x01	Sampling Period = 2 × $T_{SYSCLKOUT}$
		0x02	Sampling Period = 4 × $T_{SYSCLKOUT}$
		...	...
		0xFF	Sampling Period = 510 × $T_{SYSCLKOUT}$
7-0	QUALPRD0		Specifies the sampling period for pins GPIO32 to GPIO39
		0x00	Sampling Period = $T_{SYSCLKOUT}$ (2)
		0x01	Sampling Period = 2 × $T_{SYSCLKOUT}$
		0x02	Sampling Period = 4 × $T_{SYSCLKOUT}$
		...	...
		0xFF	Sampling Period = 510 × $T_{SYSCLKOUT}$

(1) This register is EALLOW protected. See Section 7.2 for more information.
(2) $T_{SYSCLKOUT}$ indicates the period of SYSCLKOUT.

"GpioCtrlRegs.GPBCTRL.bit.QUALPRD1 = 0xFF;"

- OxFF : Sampling Period = 510 * $T_{SYSCLKOUT}$

GpioCtrlRegs.GPBQSEL1.bit.GPIO44 = 2; // Qualification using 6 samples

Figure 57. GPIO Port B Qualification Select 1 (GPBQSEL1) Register

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
GPIO47	GPIO46	GPIO45	GPIO44	GPIO43	GPIO42	GPIO41	GPIO40
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15 14	**13 12**	**11 10**	**9 8**	**7 6**	**5 4**	**3 2**	**1 0**
GPIO39	GPIO38	GPIO37	GPIO36	GPIO35	GPIO34	GPIO33	GPIO32
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 60. GPIO Port B Qualification Select 1 (GPBQSEL1) Register Field Descriptions

Bits	Field	Value	Description (1)
31-0	GPIO47-GPIO32		Select input qualification type for GPIO32 to GPIO47. The input qualification of each GPIO input is controlled by two bits as shown in Figure 55.
		00	Synchronize to SYSCLKOUT only. Valid for both peripheral and GPIO pins.
		01	Qualification using 3 samples. Valid for pins configured as GPIO or a peripheral function. The time between samples is specified in the GPACTRL register.
		10	Qualification using 6 samples. Valid for pins configured as GPIO or a peripheral function. The time between samples is specified in the GPACTRL register.
		11	Asynchronous. (no synchronization or qualification). This option applies to pins configured as peripherals only. If the pin is configured as a GPIO input, then this option is the same as 0,0 or synchronize to SYSCLKOUT.

(1) This register is EALLOW protected. See Section 7.2 for more information.

```
        // Step 5. XINT 초기화
        EALLOW;
        GpioIntRegs.GPIOXINT3SEL.bit.GPIOSEL = 47;
// 외부인터럽트XINT3로사용할핀선택: GPIO47
        GpioIntRegs.GPIOXINT4SEL.bit.GPIOSEL = 46;
// 외부인터럽트XINT4로사용할핀선택: GPIO46
        GpioIntRegs.GPIOXINT5SEL.bit.GPIOSEL = 45;
// 외부인터럽트XINT5로사용할핀선택: GPIO45
        GpioIntRegs.GPIOXINT6SEL.bit.GPIOSEL = 44;
// 외부인터럽트XINT6로사용할핀선택: GPIO44
        EDIS;
```

```
        XIntruptRegs.XINT3CR.bit.POLARITY = 2;      // XINT3 인터럽트발생조건설정: 입력신호의하강엣지
        XIntruptRegs.XINT4CR.bit.POLARITY = 1;      // XINT4 인터럽트발생조건설정: 입력신호의상승엣지
        XIntruptRegs.XINT5CR.bit.POLARITY = 1;      // XINT5 인터럽트발생조건설정: 입력신호의하강엣지
        XIntruptRegs.XINT6CR.bit.POLARITY = 1;      // XINT6 인터럽트발생조건설정: 입력신호의하강& 상승엣지
```

Bits	Field	Value	Description
15-4	Reserved		Reads return zero; writes have no effect.
3-2	Polarity		This read/write bit determines whether interrupts are generated on the rising edge or the falling edge of the signal on the pin.
		00	Interrupt generated on a falling edge (high-to-low transition)
		01	Interrupt generated on a rising edge low-to-high transition)
		10	Interrupt is generated on a falling edge (high to low transition)
		11	Interrupt generated on both a falling edge and a rising edge (high to low and low to high transition)
1	Select		Select the source for INT13

```
        XIntruptRegs.XINT3CR.bit.ENABLE = 1;        // XINT3 인터럽트: Enable
        XIntruptRegs.XINT4CR.bit.ENABLE = 1;        // XINT4 인터럽트: Enable
        XIntruptRegs.XINT5CR.bit.ENABLE = 1;        // XINT5 인터럽트: Enable
        XIntruptRegs.XINT6CR.bit.ENABLE = 1;        // XINT6 인터럽트: Enable
```

Table 83. XINT3 - XINT7 Interrupt Select and Configuration Registers

n	Interrupt	Interrupt Select Register	Configuration Register
3	XINT3	GPIOXINT3SEL	XINT3CR
4	XINT4	GPIOXINT4SEL	XINT4CR
5	XINT5	GPIOXINT5SEL	XINT5CR
6	XINT6	GPIOXINT6SEL	XINT6CR
7	XINT7	GPIOXINT7SEL	XINT7CR

```
        // 외부인터터트포합된백터활성화
        PieCtrlRegs.PIEIER12.bit.INTx1 = 1;
// PIE 인터럽트(XINT3) : Enable
        PieCtrlRegs.PIEIER12.bit.INTx2 = 1;
// PIE 인터럽트(XINT4) : Enable
        PieCtrlRegs.PIEIER12.bit.INTx3 = 1;
// PIE 인터럽트(XINT5) : Enable
        PieCtrlRegs.PIEIER12.bit.INTx4 = 1;
// PIE 인터럽트(XINT6) : Enable
        IER |= M_INT12;
// CPU 인터럽트(INT12) : Enable
```

```
        //===========================================================================================
```

Table 113. PIE Configuration and Control Registers

Name	Address	Size (x16)	Description
PIEIER12	0x0000 - 0CF8	1	PIE, INT12 Group Enable Register
PIEIFR12	0x0000 - 0CF9	1	PIE, INT12 Group Flag Register

소스1) GPIO 44~47 Pin을 각각 외부 인터럽트 3, 4, 5, 6으로 설정하였고, 적절한 상승 & 하강 엣지를 감지하도록 설정하였다. 외부인터럽트 3 ~ 6을 활성화하고, 외부인터럽트를 포함한 벡터를 활성화하였다.

```
// Step 6. Initialize Application Variables
        SW1_cnt = 0;        SW2_cnt = 0;        SW3_cnt = 0;        SW4_cnt = 0;
        Loop_cnt = 0;
```

소스1) 인터럽트 함수가 몇 번 동작하였는 지 카운트 해주는 변수 SWx_cnt와 main문이 얼마나 반복동작 했는 지 알려주는 Loop_cnt 변수를 선언하였다. 모든 설정이 끝났으므로, 인터럽트를 활성화시키고 동작을 시작했다.

```
// Enable global Interrupts and higher priority real-time debug events:

        EINT;   // Enable Global interrupt INTM
        ERTM;   // Enable Global realtime interrupt DBGM

        // IDLE loop. Just sit and loop forever :


        for (;;)
        {
                if (stop == 1)
                {
                        for (i = 1; i < 3; i++)
                        {
if (stop != 1) { break; }
                                modeS(i);
                        }
                }
                else if (mode == 1)
                {
                        for (i = 1; i < 17; i++)
                        {
                if (mode != 1 || stop != 0) { break; }
                                modeA(i);
                        }
                }
                else if (mode == 2)
                {
                        for (i = 1; i < 137; i++)
                        {
                if (mode != 2 || stop != 0) { break; }
                                modeB(i);
                        }
                }
                else if (mode == 3)
                {
                        for (i = 1; i < 3; i++)
                        {
                if (mode != 3 || stop != 0) { break; }
                                modeC(i);
                        }
                }
                else if (mode == 4)
                {
                        for (i = 1; i < 17; i++)
                        {
                                if (mode != 4 || stop !=
0) { break; }
                                modeD(i);
                        }
                }
                else if (mode == 5)
                {
                        for (i = 1; i < 7; i++)
                        {
                if (mode != 5 || stop != 0) { break; }
                                modeE(i);
                        }
                }
                else if (mode == 6)
                {
                        for (i = 1; i < 9; i++)
                        {
                if (mode != 6 || stop != 0) { break; }
                                modeF(i);
                        }
                }
                else if (mode >= 6)
                {
                        mode = 1;
                }
                else if (mode <= 1)
                {
                        mode = 6;
                }
                Loop_cnt++;
        }
}
```

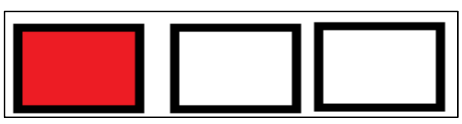
```
if (stop == 1)
{
        for (i = 1; i < 3; i++)
        {
if (stop != 1) { break; }
                modeS(i);
        }
}
else if (mode == 1)
```

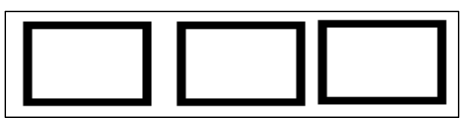
1000줄짜리는 stop과 동작으로 구분되기 때문에, stop이 1이 되는 경우를 if 문으로 설정하고, 그 이외의 값을 else if로 설정하였다.

stop에 “1”이 입력되면, modeS가 실행된다.

“for(;;)”을 통해, 무한 반복문임을 알 수 있는데, “i++”을 통해, 반복문이 진행될 때마다 I의 값이 늘어나고, 그 값은 LED 점등 상태의 개수를 의미한다.

- “stop mode”는 맨 왼쪽 led만 on, off가 반복하기에, 필요한 Led가 2개이다. 따라서, “i<3”라고 설정하였다.

 : on,

 : off

“modeS”의 탈출조건은 “If(stop !=1) {break;]”을 통해, 구현하였다. 즉, “stop = 0” 일 때이다.

```
else if (mode == 1)
{
        for (i = 1; i < 17; i++)
        {
if (mode != 1 || stop != 0) { break; }
            modeA(i);
        }
}
```

stop 모드 이후에는 동일한 방법으로 코드가 실행된다.
mode가 “1, 2, 3, 4, 5, 6”이냐에 따라 실행되는 동작은 다르고, 그 값에 따라 “A, B, C, D, E, F”가 실행 된다.
탈출조건도 stop 모드가 동일하게, 자신에 해당하는 mode와 다른 값이 실행되었을 때이다.

```
    else if (mode >= 6)
    {
        mode = 1;
    }

    else if (mode <= 1)
    {
        mode = 6;
    }

    Loop_cnt++;
```

소스1) 변수 mode가 1~6 사이의 값을 벗어나지 않도록, 만약 변수 mode가 6일 때 버튼 A가 눌려서 mode값이 7로 증가하거나, 또는 변수 mode가 1일 때 버튼 B가 눌려서 변수 mode가 0으로 변하는 경우 다시 변수 mode가 1 ~ 6 사이의 값을 유지하도록 조건문을 통해 변수 mode값을 1 ~ 6으로 제한하였다.

```
//   메인함수- 끝              //   ISR 함수정의
interrupt void Xint3_isr(void)
{
        SW1_cnt++;
        mode++;
        stop = 0;
        PieCtrlRegs.PIEACK.all = PIEACK_GROUP12;
}

interrupt void Xint4_isr(void)
{
        SW2_cnt++;
        mode--;
        stop = 0;
        PieCtrlRegs.PIEACK.all = PIEACK_GROUP12;
}
```

```
interrupt void Xint5_isr(void)
{
        SW3_cnt++;
        stop = 1;
        PieCtrlRegs.PIEACK.all = PIEACK_GROUP12;
}

interrupt void Xint6_isr(void)
{
        SW4_cnt++;
        stop = 0;
        mode = 1;
        PieCtrlRegs.PIEACK.all = PIEACK_GROUP12;
}
```

ISR함수 정의는 Xint5와 Xint6은 정지와 시작을 의미하기에, Xint5의 경우, stop모드가 실행되기 위해, “stop = 1”이 되고, Xint6의 경우, start를 하기 위해, “stop = 0”은 되지만, “mode = 1”이 된다.

Xint3과 Xint4는 mode값의 증가 감소이기에, “mode++”, “mode--”으로 표현하였다.
그리고 “stop=0”인 이유는 동작을 하는 함수이기에, 설정하였다.

Table 115. PIE Interrupt Acknowledge Register (PIEACK) Field Descriptions

Bits	Field	Value	Description
15-12	Reserved		Reserved
11-0	PIEACK		Each bit in PIEACK refers to a specific PIE group. Bit 0 refers to interrupts in PIE group 1 that are MUXed into $\overline{INT1}$ up to Bit 11, which refers to PIE group 12 which is MUXed into CPU $\overline{INT12}$
		bit x = 0 (1)	If a bit reads as a 0, it indicates that the PIE can send an interrupt from the respective group to the CPU.
			Writes of 0 are ignored.
		bit x = 1	Reading a 1 indicates if an interrupt from the respective group has been sent to the CPU and all other interrupts from the group are currently blocked.
			Writing a 1 to the respective interrupt bit clears the bit and enables the PIE block to drive a pulse into the CPU interrupt input if an interrupt is pending for that group.

(1) bit x = PIEACK bit 0 - PIEACK bit 11. Bit 0 refers to CPU $\overline{INT1}$ up to Bit 11, which refers to CPU $\overline{INT12}$