UART = 시리얼 통신을 관장하는 컨트롤러
시리얼 통신 = 전체수준은 시리얼 통신을 지칭

General-purpose input/output (GPIO) is a generic pin on an integrated circuit or computer board whose behavior—including whether it is an input or output pin—is controllable by the user at run time.

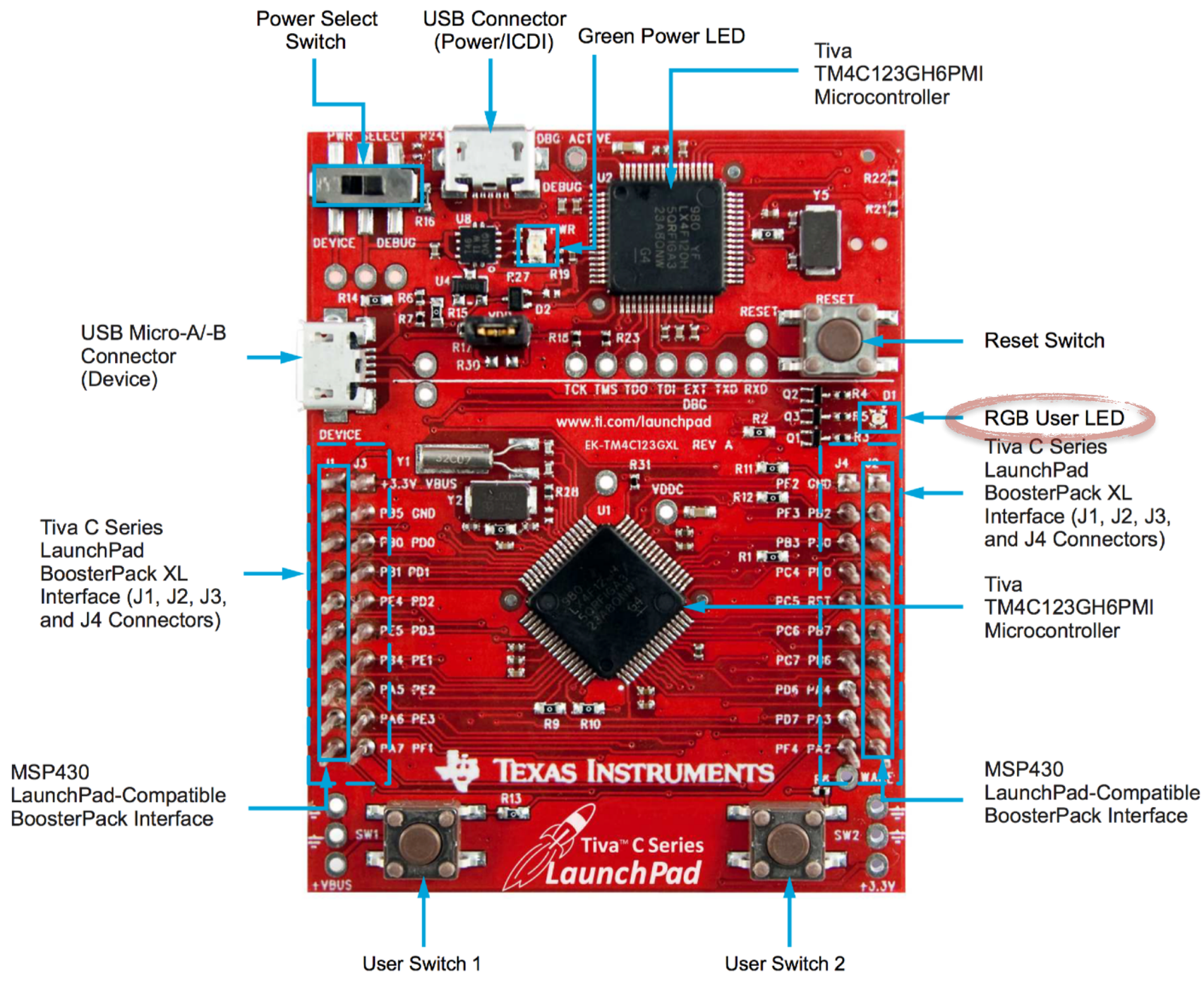
DR = data register

GPIO Programming

Lesson 02

Blinking LED

LED


Power Select Switch
USB Connector (Power/ICDI)
Green Power LED
Tiva TM4C123GH6PMI Microcontroller
USB Micro-A/-B Connector (Device)
Reset Switch
RGB User LED
Tiva C Series LaunchPad BoosterPack XL Interface (J1, J2, J3, and J4 Connectors)
Tiva C Series LaunchPad BoosterPack XL Interface (J1, J2, J3, and J4 Connectors)
Tiva TM4C123GH6PMI Microcontroller
MSP430 LaunchPad-Compatible BoosterPack Interface
MSP430 LaunchPad-Compatible BoosterPack Interface
User Switch 1
User Switch 2
www.ti.com/launchpad
EK-TM4C123GXL REV A
TEXAS INSTRUMENTS
Tiva™ C Series LaunchPad

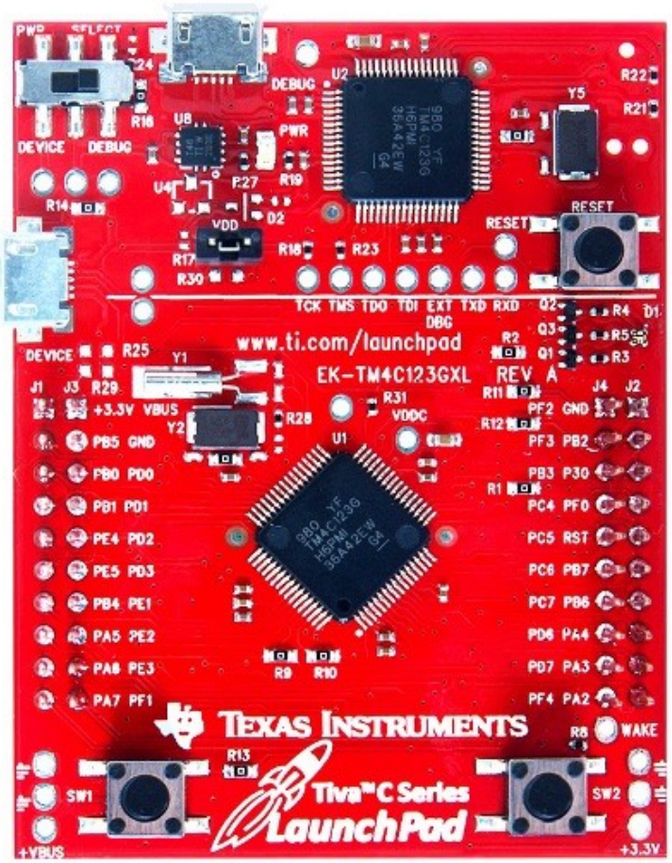
IO pins

TIVA™ C Series TM4C123G LaunchPad

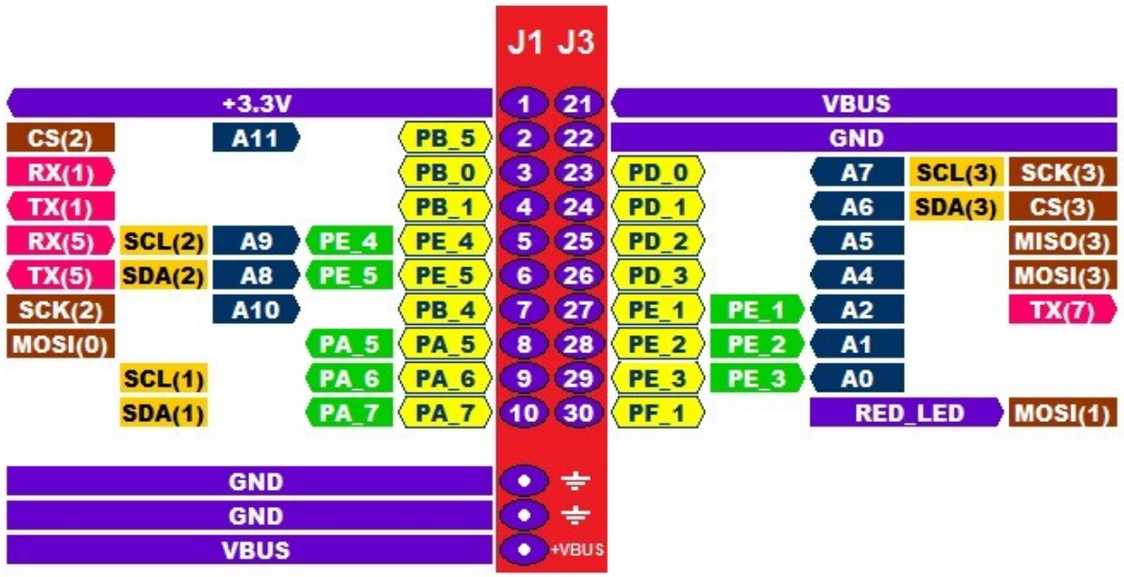
TEXAS INSTRUMENTS

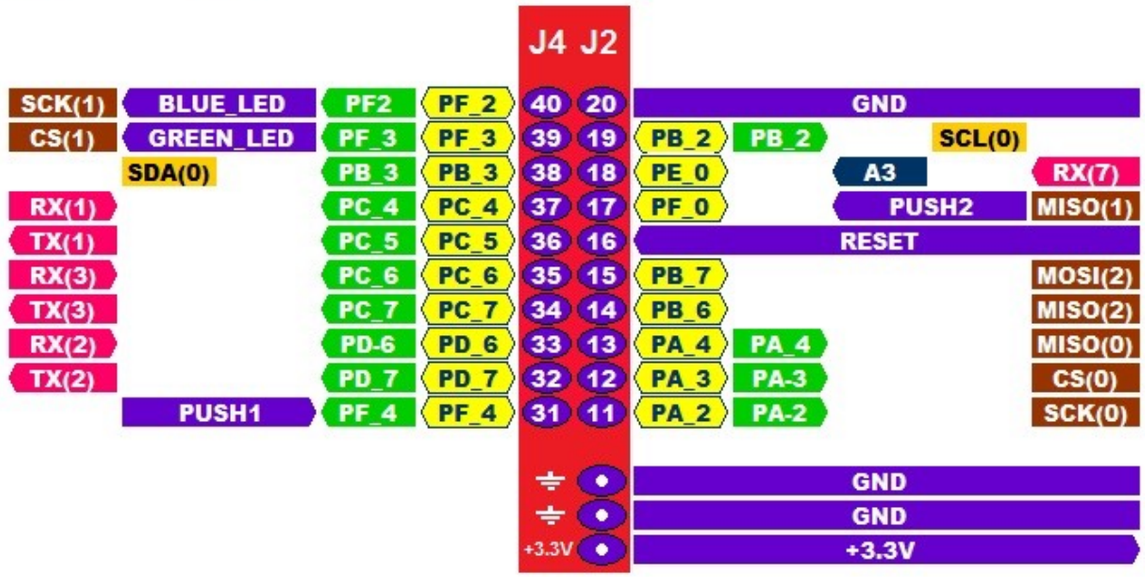
Energīa

Pinout Diagram Ver 1.0

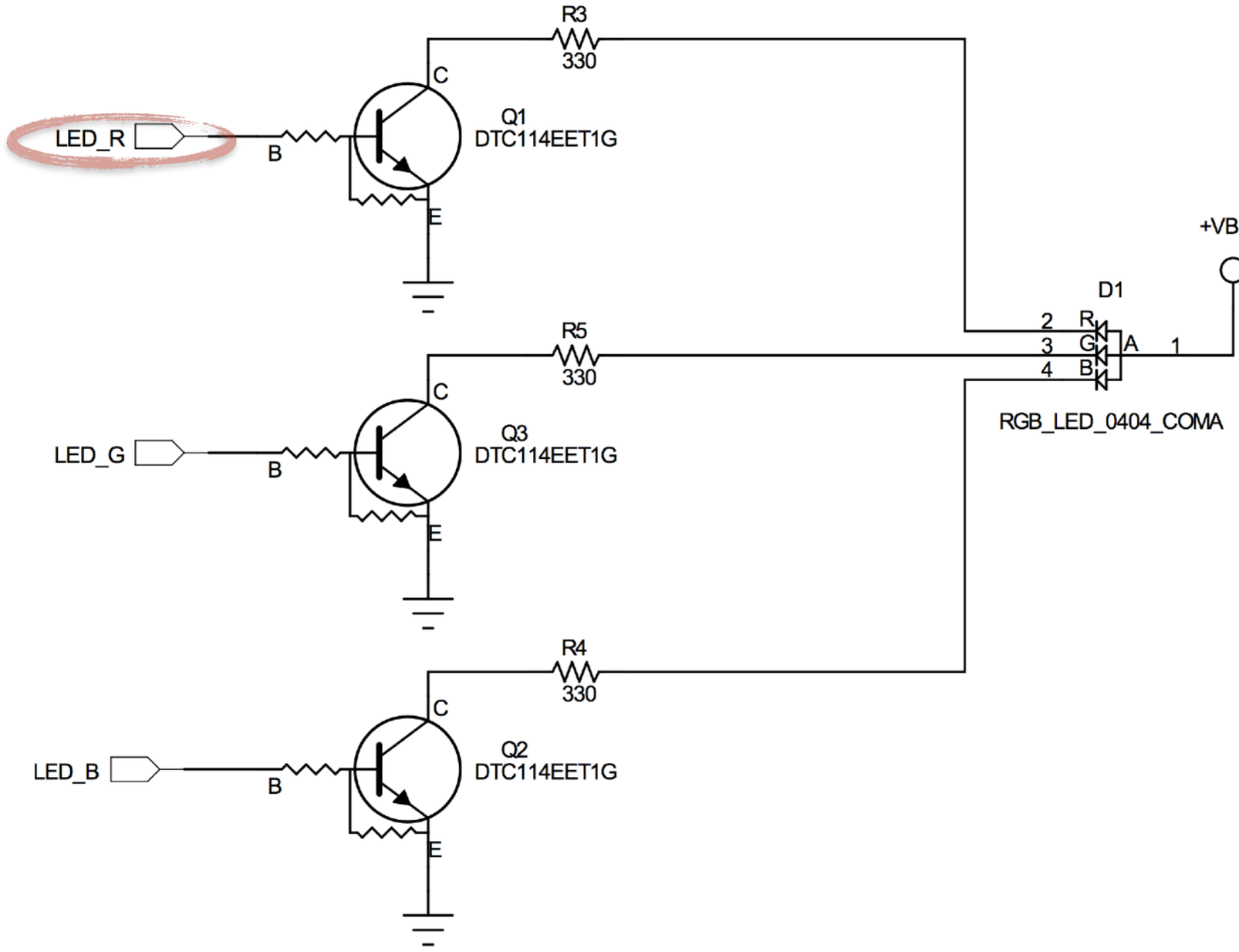


Hardware
digitalRead() and digitalWrite() PORTS
analogRead()
analogWrite()
I²C (TWI)
SPI
Hardware Serial

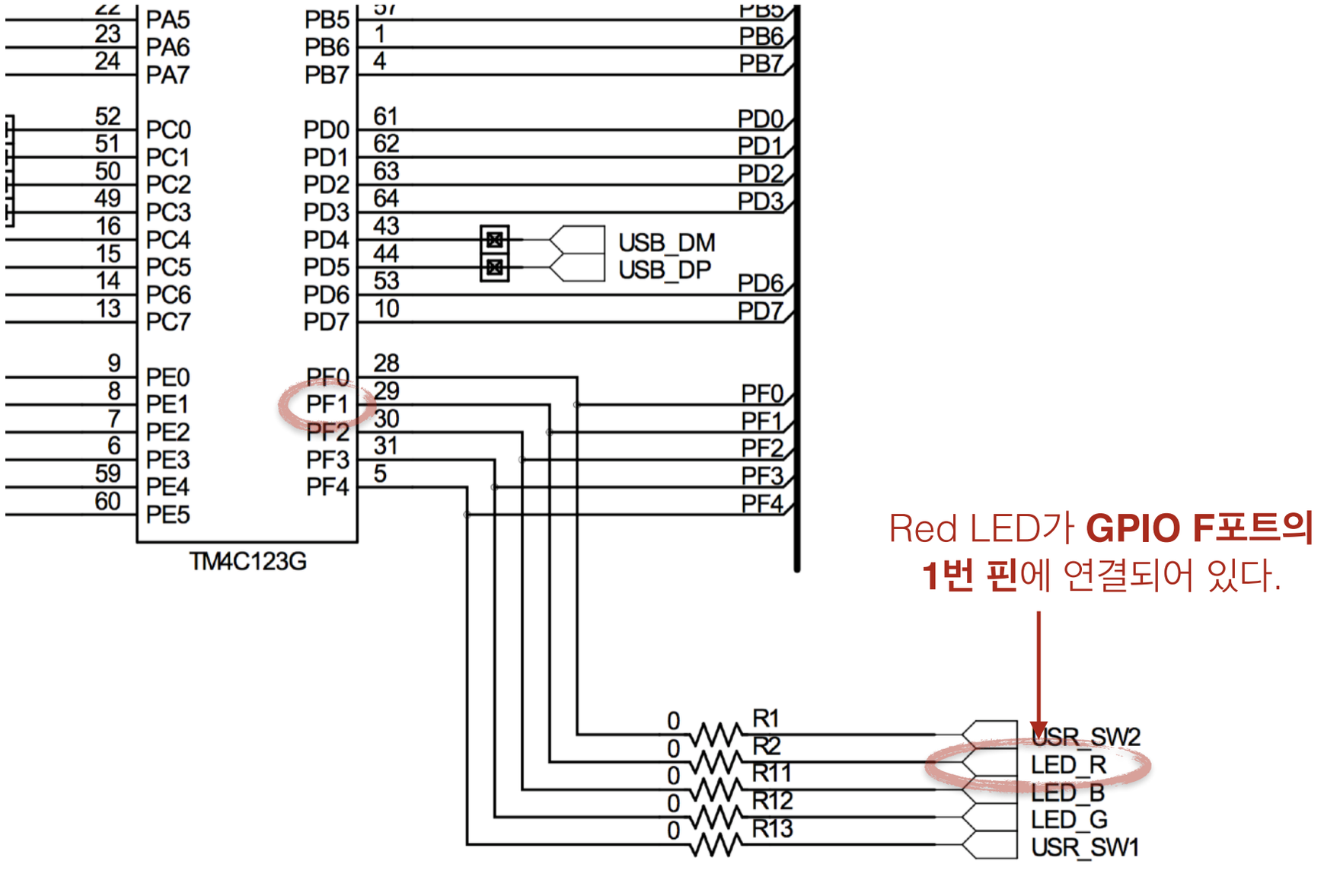




LED

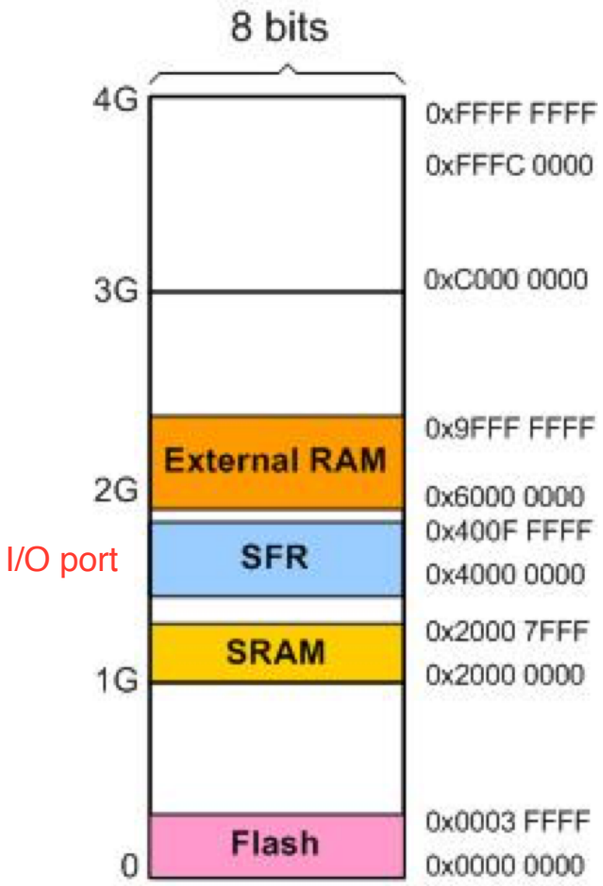

R3
330
C
Q1
DTC114EET1G
LED_R
B
E
+VBUS
D1
2 R
3 G
4 B
A
1
R5
330
C
Q3
DTC114EET1G
LED_G
B
E
RGB_LED_0404_COMA
R4
330
C
Q2
DTC114EET1G
LED_B
B
E

LED



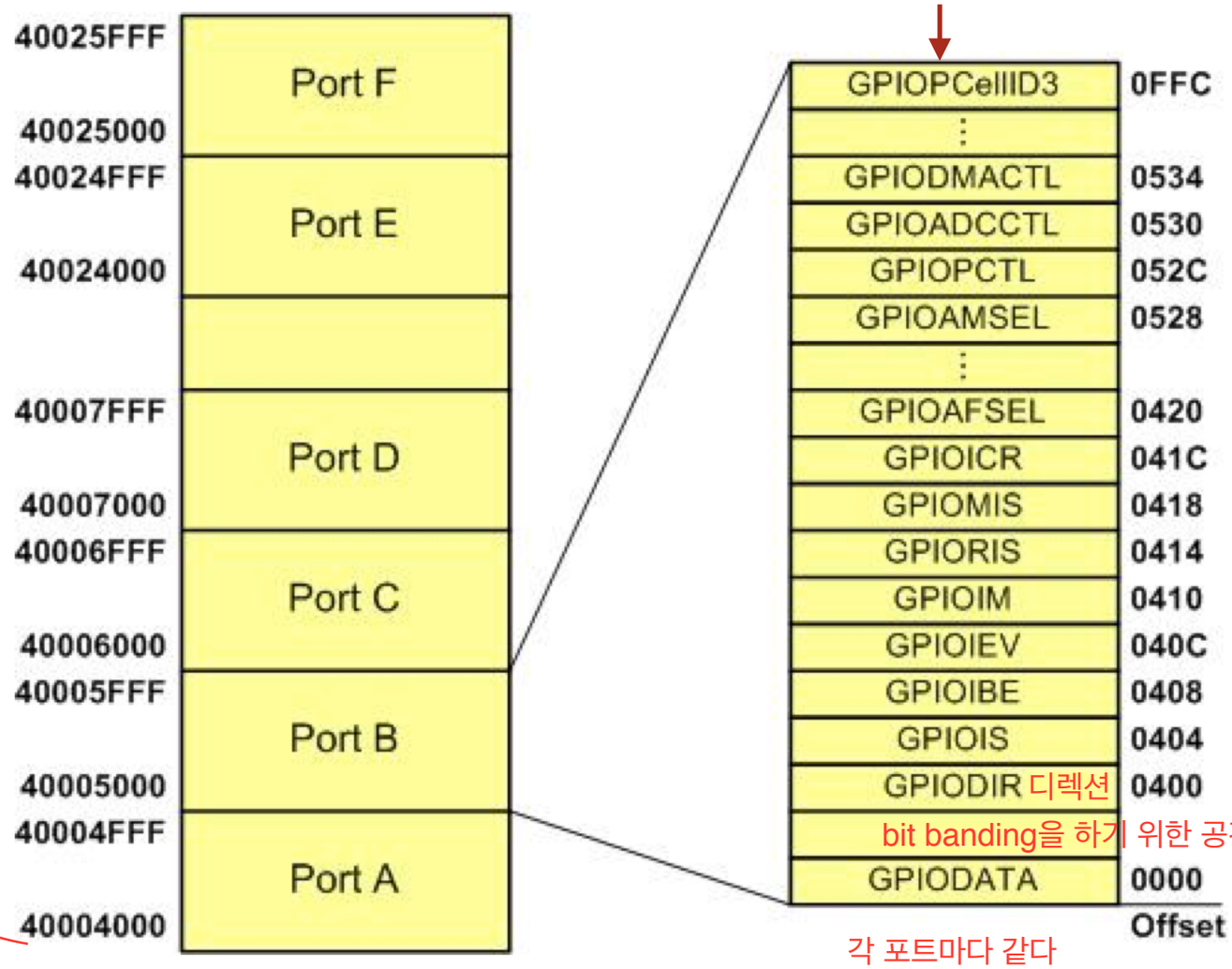
Red LED가 **GPIO F포트의 1번 핀**에 연결되어 있다.

GPIO Memory Map



I/O port

memorymap



Launchpad에는 A~F까지 6개의
GPIO 포트가 있고, 각 포트에는 최대 8개의 핀이 있다.

GPIO Port F Base Address

Table 2-4. Memory Map *(continued)*

Start	End	Description	For details, see page ...
0x4000.6000	0x4000.6FFF	GPIO Port C	658
0x4000.7000	0x4000.7FFF	GPIO Port D	658
0x4000.8000	0x4000.8FFF	SSI0	967
0x4000.9000	0x4000.9FFF	SSI1	967
0x4000.A000	0x4000.AFFF	SSI2	967
0x4000.B000	0x4000.BFFF	SSI3	967
0x4000.C000	0x4000.CFFF	UART0	903
0x4000.D000	0x4000.DFFF	UART1	903
0x4000.E000	0x4000.EFFF	UART2	903
0x4000.F000	0x4000.FFFF	UART3	903
0x4001.0000	0x4001.0FFF	UART4	903
0x4001.1000	0x4001.1FFF	UART5	903
0x4001.2000	0x4001.2FFF	UART6	903
0x4001.3000	0x4001.3FFF	UART7	903
0x4001.4000	0x4001.FFFF	Reserved	-
Peripherals			
0x4002.0000	0x4002.0FFF	I^2C 0	1017
0x4002.1000	0x4002.1FFF	I^2C 1	1017
0x4002.2000	0x4002.2FFF	I^2C 2	1017
0x4002.3000	0x4002.3FFF	I^2C 3	1017
0x4002.4000	0x4002.4FFF	GPIO Port E	658
0x4002.5000	0x4002.5FFF	GPIO Port F	658
0x4002.6000	0x4002.7FFF	Reserved	-

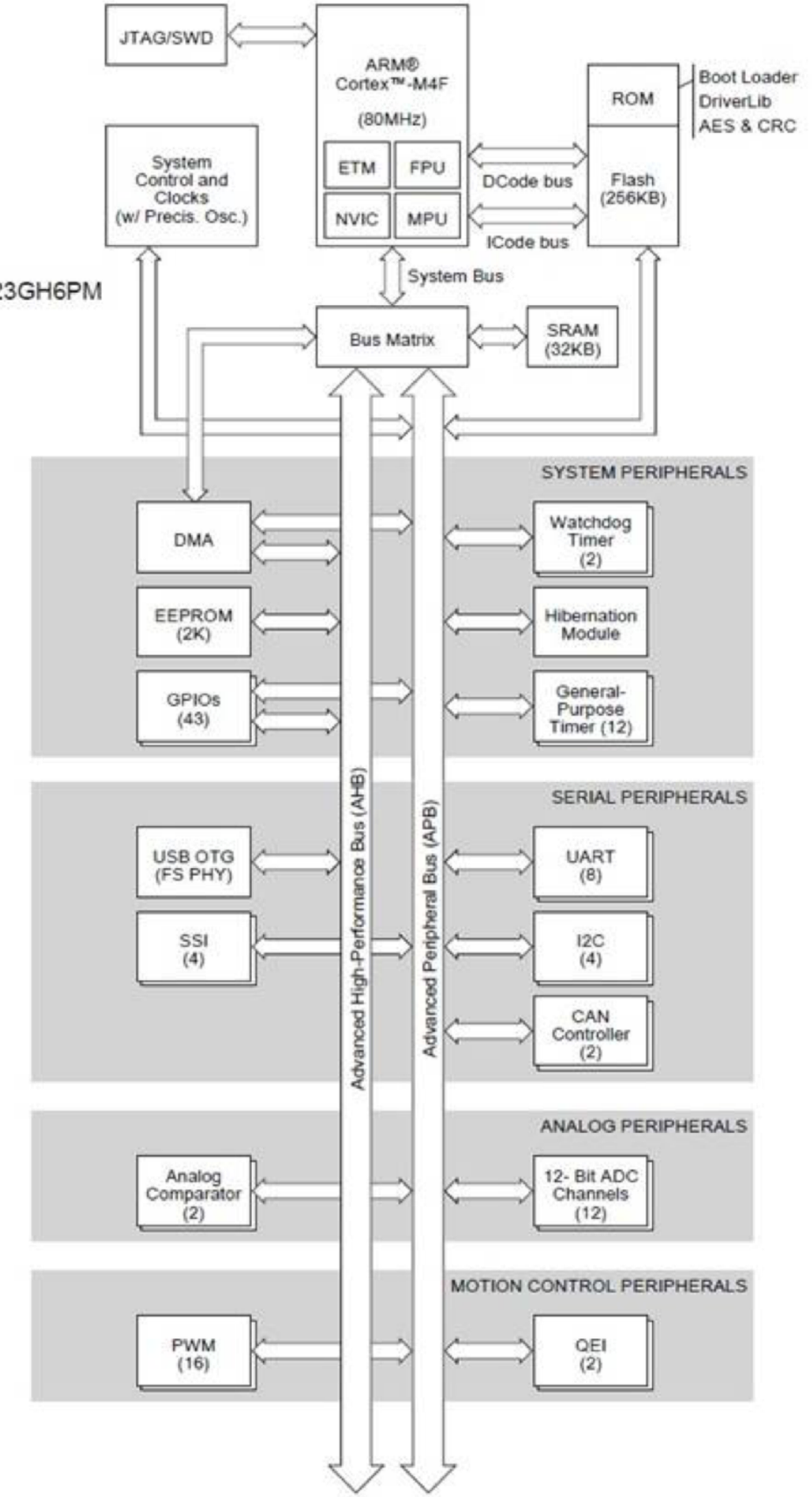
GPIO 포트 F의
base address를 확인한다.

Two Buses

- **2종류의 I/O Bus**
  - APB (Advanced Peripheral Bus)와 AHB (Advanced High-Performance Bus)

GPIO Ports Base Address for APB	
GPIO Port A (APB)	0x4000.4000
GPIO Port B (APB)	0x4000.5000
GPIO Port C (APB)	0x4000.6000
GPIO Port D (APB)	0x4000.7000
GPIO Port E (APB)	0x4002.4000
GPIO Port F (APB)	0x4002.5000
GPIO Ports Base Address for AHB	
GPIO Port A (AHB)	0x4005.8000
GPIO Port B (AHB)	0x4005.9000
GPIO Port C (AHB)	0x4005.A000
GPIO Port D (AHB)	0x4005.B000
GPIO Port E (AHB)	0x4005.C000
GPIO Port F (AHB)	0x4005.D000

TM4C123GH6PM



Direction and Data Registers

- 일반적으로 마이크로컨트롤러에서 각각의 **I/O** 포트에 대해서 적어도 **2**개의 레지스터가 있다.

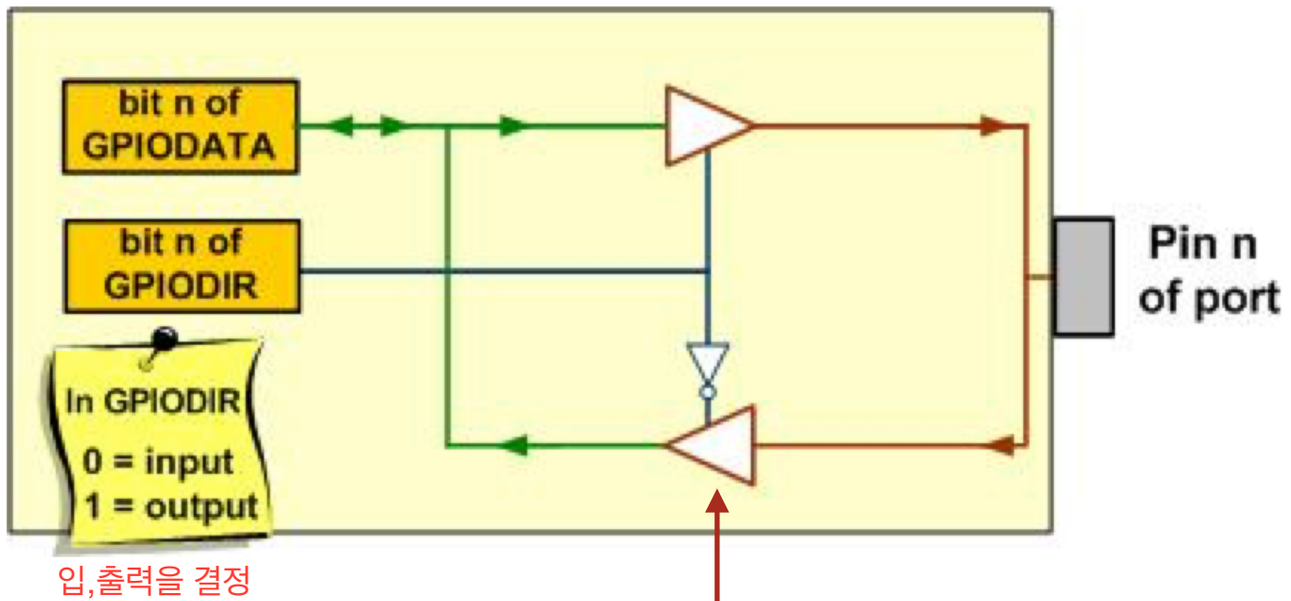


Figure 2-6: The Data and Direction Registers and a Simplified View of an I/O pin

GPIO 핀을 Digital Output으로 사용하려면

1. 해당 GPIO 포트의 Clock을 enable하고
2. 해당 핀의 direction을 output으로 설정하고
3. 해당 핀을 digital enable하고
4. 데이터를 출력한다.

Data Sheet
p.654 and Table 10-3 at p.655

클럭 게이팅: 회로에서 동작이 필요하지 않는 부분에 클럭을 공급하지 않음으로써 전력을 절감하는 기술

1. Clock Gating Control

0x400FE000 + 0x608번지의
5번째 비트에 1을 써야 한다.

General-Purpose Input/Output Run Mode Clock Gating Control (RCGCGPIO)

Base 0x400F.E000
Offset 0x608
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved										R5	R4	R3	R2	R1	R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

A~F까지 각 포트별로 1비트

find x from data sheet

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	R5	RW	0	GPIO Port F Run Mode Clock Gating Control

Value	Description
0	GPIO Port F is disabled.
1	Enable and provide a clock to GPIO Port F in Run mode.

Data Sheet
p.338

CCS의 Register View에서
SYSCTL->[0…99] 그룹에서
RCGCGPIO 레지스터의
값을 볼 수 있다.

```
*((unsigned int *)0x400FE608U) = 0x20U;
```

2. GPIO Direction Register

GPIO Direction (GPIODIR)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
Offset 0x400
Type RW, reset 0x0000.0000

LED에 연결된 PF1~PF3 핀의 direction 설정 비트들

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DIR							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

0~7번까지 각 핀별로 1비트

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DIR	RW	0x00	GPIO Data Direction

Value	Description
0	Corresponding pin is an input.
1	Corresponding pins is an output.

Data Sheet p.660

```
*((unsigned int *)0x40025400U) = 0x0EU;
```

0000 1110

3. GPIO Digital Enable Register

PF1~PF3 핀의 digital enable

GPIO Digital Enable (GPIODEN)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
Offset 0x51C
Type RW, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DEN							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DEN	RW	-	Digital Enable

Value	Description
0	The digital functions for the corresponding pin are disabled.
1	The digital functions for the corresponding pin are enabled.

The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in Table 10-1 on page 650.

Data Sheet p.680

```
*((unsigned int *)0x4002551CU) = 0x0EU;
```

4. GPIO Data Register

GPIO Data (GPIODATA)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
Offset 0x000
Type RW, reset 0x0000.0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved								DATA							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DATA	RW	0x00	GPIO Data This register is virtually mapped to 256 locations in the address space. To facilitate the reading and writing of data to these registers by independent drivers, the data read from and written to the registers are masked by the eight address lines [9:2]. Reads from this register return its current state. Writes to this register only affect bits that are not masked by ADDR[9:2] and are configured as outputs. See “Data Register Operation” on page 654 for examples of reads and writes.

Data Sheet
p.659
and p.652

Bit Banding in Data Register

search???

- **Cortex M4는 bit banding을 지원**
  - 데이터 레지스터의 8비트의 임의의 부분집합을 하나의 주소에 대응 (256개의 주소)
  - 32비트 아키텍처이므로 각 주소의 하위 2비트는 0이어야 한다. 그래서 주소공간에서 256 words (offset 0x000 - 0x3FC)를 차지한다.
  - 8비트 모두에 쓰려면 offset 0x3FC에 write한다.


```
*((unsigned int *)0x400253FCU) = 0x02U;  // On
*((unsigned int *)0x400253FCU) = 0x00U;  // Off
```

Blinking 01

```
int main() {
    *((unsigned int *)0x400FE608U) = 0x20U;
    *((unsigned int *)0x40025400U) = 0x0EU;
    *((unsigned int *)0x4002551CU) = 0x0EU;

    while (1) {
        *((unsigned int *)0x400253FCU) = 0x02U;

        int counter = 0;
        while (counter < 1000000) {
            ++counter;
        }

        *((unsigned int *)0x400253FCU) = 0x00U;
        counter = 0;
        while (counter < 1000000) {
            ++counter;
        }

    }
    return 0;
}
```

Enable clock

Select direction as output

Digital enable

Data 레지스터의 offset이 0x3FC라고 생각하면 됨

delay

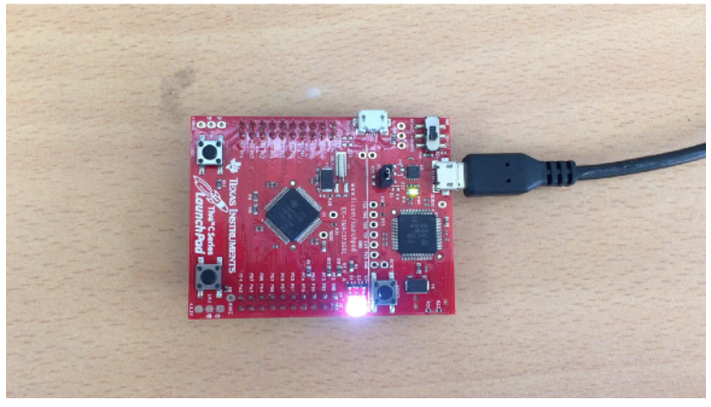
delay

Blinking 02

```
/*PF1 - red LED, PF2 - blue LED, PF3 - green LED  */

/* PORTF data register */
#define PORTFDAT (*((volatile unsigned int*)0x400253FC))
/* PORTF data direction register */
#define PORTFDIR (*((volatile unsigned int*)0x40025400))
/* PORTF digital enable register */
#define PORTFDEN (*((volatile unsigned int*)0x4002551C))
/* run mode clock gating register */
#define RCGCGPIO (*((volatile unsigned int*)0x400FE608))

void delayMs(int n);     /* function prototype for delay */
```

Blinking 02

```
int main(void)
{
                덮어씌움
    RCGCGPIO |= 0x20;  /* enable clock to GPIOF at clock gating register */
    PORTFDIR = 0x0E;       /* set PORTF pin3-1 as output pins */
    PORTFDEN = 0x0E;       /* set PORTF pin3-1 as digital pins */

    while(1)
    {
        PORTFDAT = 0x0E;       /* write PORTF to turn on all LEDs */
        delayMs(1000000);
        PORTFDAT = 0;          /* write PORTF to turn off all LEDs */
        delayMs(1000000);
    }
}

void delayMs(int n)
{
    volatile int i;
    for(i = 0 ; i < n; i++) {}
}
```



Assembly

```
          main():
0000033c:   B580                push        {r7, lr}
0000033e:   AF00                add         r7, sp, #0

14                RCGCGPIO |= 0x20;     /* set PORTF pin3-1 as output pins */
00000340:   4A0C                ldr         r2, [pc, #0x30]
00000342:   4B0C                ldr         r3, [pc, #0x30]
00000344:   681B                ldr         r3, [r3]
00000346:   F0430320            orr         r3, r3, #0x20
0000034a:   6013                str         r3, [r2]

15                PORTFDIR = 0x0E;  /* set PORTF pin3-1 as digital pins */
0000034c:   4B0A                ldr         r3, [pc, #0x28]
0000034e:   220E                movs        r2, #0xe
00000350:   601A                str         r2, [r3]

16                PORTFDEN = 0x0E;
00000352:   4B0A                ldr         r3, [pc, #0x28]
00000354:   220E                movs        r2, #0xe
00000356:   601A                str         r2, [r3]
```

return address (LR)과 현재 stack top의 주소 (R7)을 스택에 push한다.
스택에 push하면 SP는 자동으로 갱신된다.

새로운 stack top의 주소 (SP)를 R7에 복사한다.

pc+0x30 =0x372지만 4의 배수가 아니므로
0x374번지의 값을 R2로 load한다.

pc의 값은 2씩 자동 증가되므로 이번에도 0x374번지에
저장된 값을 R3로 load한다.

R3에 저장된 값은 RCGCGPIO 레지스터의 주소이다.
해당 레지스터의 현재 값을 R3로 읽어온다.

R3에 저장된 값에 0x20을 OR한다.

R3에 저장된 값을 R2에 저장된 주소, 즉 RCGCGPIO에
저장한다.

Assembly

```
19                  PORTFDAT = 0x0E;
00000358:   4B09                ldr         r3, [pc, #0x24]
0000035a:   220E                movs        r2, #0xe
0000035c:   601A                str         r2, [r3]

20                  delayMs(1000000);
0000035e:   4809                ldr         r0, [pc, #0x24]
00000360:   F000F812            bl          #0x388
```

PC+0x24=0x384번지에 저장된 상수 1000000을 R0에 저장한다.

return address를 LR에 저장하고 delayMs함수로 분기한다.

```
22                  PORTFDAT = 0;
00000364:   4B06                ldr         r3, [pc, #0x18]
00000366:   2200                movs        r2, #0
00000368:   601A                str         r2, [r3]

23                  delayMs(1000000);
0000036a:   4806                ldr         r0, [pc, #0x18]
0000036c:   F000F80C            bl          #0x388
```

Assembly

```
24              }
00000370:   E7F2            b           #0x358
00000372:   BF00            nop
00000374:   E608            b           #0xffffff88
00000376:   400F            ands        r7, r1
00000378:   5400            strb        r0, [r0, r0]
0000037a:   4002            ands        r2, r0
0000037c:   551C            strb        r4, [r3, r4]
0000037e:   4002            ands        r2, r0
00000380:   53FC            strh        r4, [r7, r7]
00000382:   4002            ands        r2, r0
00000384:   4240            rsbs        r0, r0, #0
00000386:   000F            movs        r7, r1
```

3개의 레지스터의 주소

1,000,000=0xF4240

0000 = 플래시 메모리 영역 read only영역

Assembly

```
27          void delayMs(int n){
            delayMs():
00000388:     B480                push        {r7}
0000038a:     B085                sub         sp, #0x14
0000038c:     AF00                add         r7, sp, #0
0000038e:     6078                str         r0, [r7, #4]

29            for(i = 0 ; i < n; i++) {}
00000390:     2300                movs        r3, #0
00000392:     60FB                str         r3, [r7, #0xc]
00000394:     E002                b           #0x39c
00000396:     68FB                ldr         r3, [r7, #0xc]
00000398:     3301                adds        r3, #1
0000039a:     60FB                str         r3, [r7, #0xc]
0000039c:     68FA                ldr         r2, [r7, #0xc]
0000039e:     687B                ldr         r3, [r7, #4]
000003a0:     429A                cmp         r2, r3
000003a2:     DBF8                blt         #0x396
30          }
000003a4:     3714                adds        r7, #0x14
000003a6:     46BD                mov         sp, r7
000003a8:     F85D7B04            ldr         r7, [sp], #4
000003ac:     4770                bx          lr
000003ae:     BF00                nop
255         {
```

이전 스택 top의 주소를 스택에 push한다.

스택에 매개변수와 지역변수를 위한 0x14바이트를 할당한다.

r7에 스택 top의 주소를 저장하고

매개변수의 값을 스택에 저장한다.
즉 변수 n의 주소는 r7+4이다.

변수 i의 주소는 r7+0c이다.

AAPCS (ARM Architecture Procedure Call Standard)

- **ARM ABI**의 일부이며 함수의 호출에 관련된 사항을 정의한 규약
- 레지스터 **r0, r1, r2, r3**가 함수 호출에서 입력 매개변수를 저장
- 함수의 출력은 레지스터 **r0**에 저장됨
- 함수가 실행된 후 **r4 ~ r12**는 함수 호출 전과 동일한 값을 유지해야
- **LR (link register)** 레지스터를 이용하여 리턴 주소를 저장
- 호출된 함수가 **r4 ~ r12** 레지스터를 이용한다면 그 레지스터의 값은 호출 전에 스택(stack)에 저장되어야 함
- **sp (stack pointer)** 레지스터가 스택의 주소를 저장하며, **push**와 **pop** 명령을 지원함

실습 01

기말 시험 문제

- **Launchpad에 하나의 외장 LED를 연결하여 점멸하는 프로그램과 회로를 작성하라. 어떤 핀에 연결할 것인지는 data sheet와 user guide를 참고하라.**

stdint.h과 구조체(struct)

Data Types in C99

첫 비트를 부호르 사용하는냐 안하느냐 차이.

Data type	Size	Range
int8_t	1 byte	-128 to 127
uint8_t	1 byte	0 to 255
int16_t	2 bytes	-32,768 to 32,767
uint16_t	2 bytes	0 to 65,535
int32_t	4 bytes	-2,147,483,648 to 2,147,483,647
uint32_t	4 bytes	0 to 4,294,967,295
int64_t	8 bytes	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
uint64_t	8 bytes	0 to 18,446,744,073,709,551,615

Table 1-2: ISO C99 integer data types and their ranges

- 임베디드시스템 프로그래밍에서 광범위하게 사용됨
- **stdint.h 파일을 include해야**

structure

```c
#include <stdint.h>

typedef struct {
    uint8_t y;
    uint16_t x;
} Point;
Point p1, p2;
```

_ 두개

__attribute__((packed))를 Point 앞에 추가하고 비교해본다.

```c
typedef struct {
    Point top_left;
    Point bottom_right;
} Window;

typedef struct {
    Point corners[3];
} Triangle;

Window w, w2;
Triangle t;
```

structure

```
int main() {
    Point *pp;
    Window *wp;

    p1.x = sizeof(Point);
    p1.y = 0xAAU;

    w.top_left.x = 1U;
    w.bottom_right.y = 2U;

    t.corners[0].x = 1U;
    t.corners[2].y = 2U;

    p2 = p1;
    w2 = w;
    pp = &p1;
    wp = &w2;

    (*pp).x = 1U;
    (*wp).top_left = *pp;
    pp->x = 1U;
    wp->top_left = *pp;
    return 0;
}
```

memory address를 짝수로 맞춰주기 위해서..
performence가 올라감

3이 아니라
sizeof(Point)가 4가 됨을 확인한다.
__attribute__((packed))를 추가한 후 비교해본다.

byte 혹은 half word
단위의 메모리 엑세스를 위해 strb 혹은 strh 명령이 사용된다.

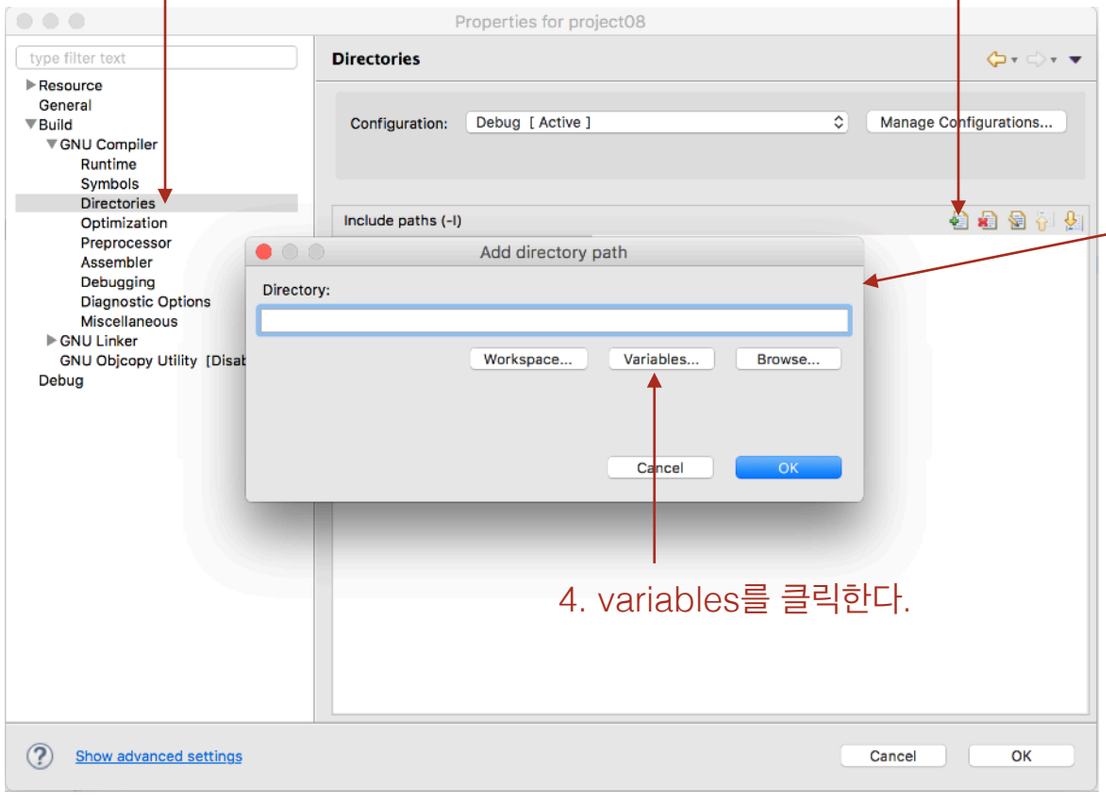
```
    p1.y = 0xAAU;
0348:  4B18        ldr     r3, [pc, #0x60]
034a:  22AA        movs    r2, #0xaa
034c:  701A        strb    r2, [r3]
    w.top_left.x = 1U;
034e:  4B18        ldr     r3, [pc, #0x60]
0350:  2201        movs    r2, #1
0352:  805A        strh    r2, [r3, #2]
    w.bottom_right.y = 2U;
0354:  4B16        ldr     r3, [pc, #0x58]
0356:  2202        movs    r2, #2
0358:  711A        strb    r2, [r3, #4]
    t.corners[0].x = 1U;
035a:  4B16        ldr     r3, [pc, #0x58]
035c:  2201        movs    r2, #1
035e:  805A        strh    r2, [r3, #2]
    t.corners[2].y = 2U;
0360:  4B14        ldr     r3, [pc, #0x50]
0362:  2202        movs    r2, #2
0364:  721A        strb    r2, [r3, #8]
```

- **TM4C123GH6PM 디바이스 해더 파일**
  - Download TM4C123GH6PM.h
  - 프로젝트의 소스 코드와 동일한 디렉토리에 복사한다.
- **CMSIS (Cortex Microcontroller Software Interface Standard)**
  - Vendor-independent hardware abstraction layer for the Cortex-M processor series
  - Download here.
  - 압축을 푼 후 CCS의 Workspace 디렉토리에 복사
  - CCS에서 Project Properties에서 include path를 설정한다 (다음 페이지 설명)

1. Build->GNU Compiler->Directories에서

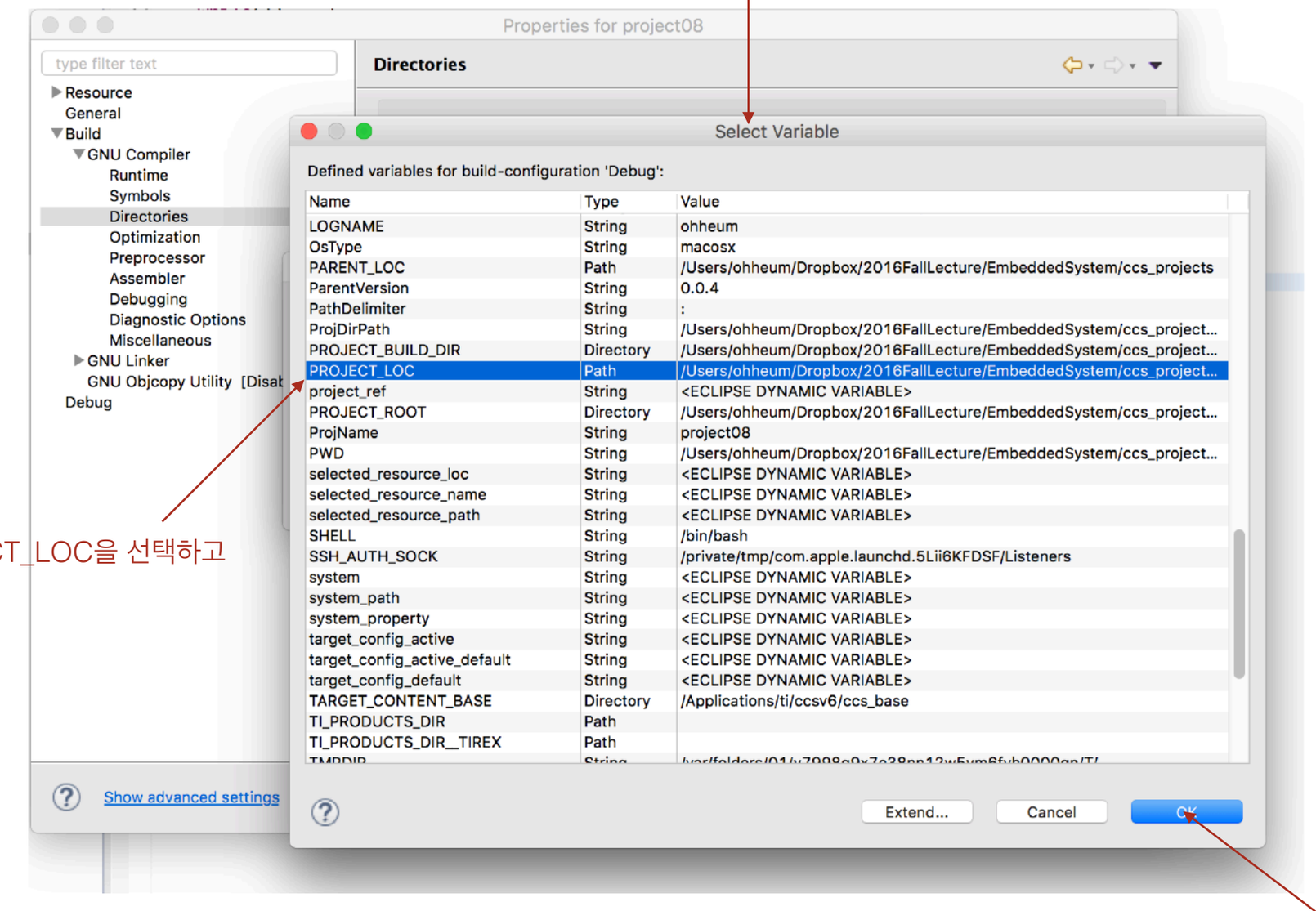
2. 이 버튼을 클릭하면

3. 이 대화상자가 열린다.



4. variables를 클릭한다.

5. Select Variables 창이 뜬다.

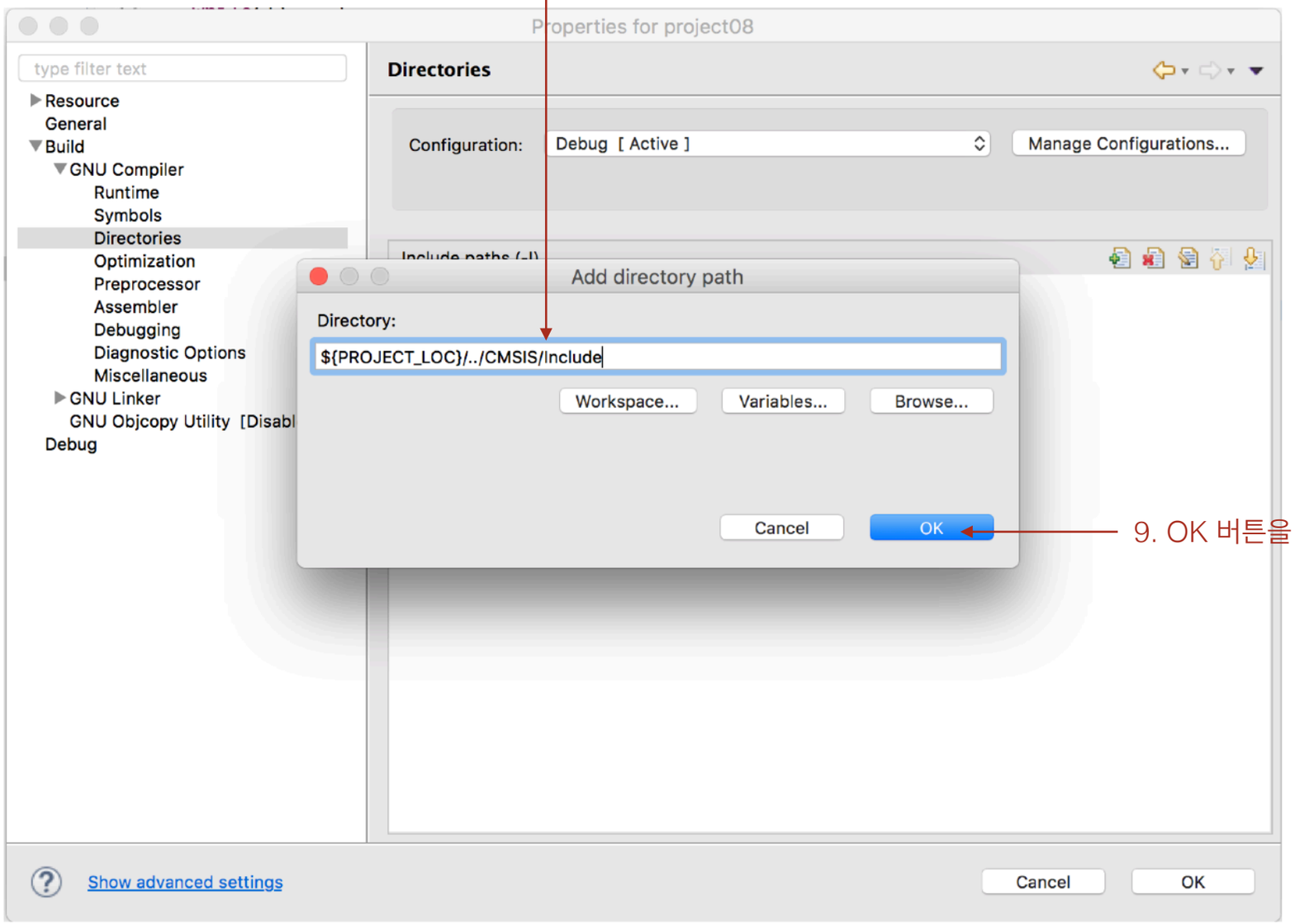


6. PROJECT_LOC을 선택하고

7. OK 버튼을 누른다.

/../ -> 위로 올간다라는 뜻이다.

8. 여기에 `/../CMSIS/Include`를 추가하고



9. OK 버튼을 누른다.

TM4C123GH6PM.h

bit banding을 위한 256개의 주소

DATA의 offset은 4*255=1020=0x3FC라고 생각한다.

DIR의 offset 0x400
=4*256=1024=0x400

```
typedef struct {
  __IO uint32_t  DATA_Bits[255];   /*!< GPIOA Structure                    */
  __IO uint32_t  DATA;             /*!< GPIO bit combinations              */
  __IO uint32_t  DIR;              /*!< GPIO Data                          */
  __IO uint32_t  IS;               /*!< GPIO Direction                     */
  __IO uint32_t  IBE;              /*!< GPIO Interrupt Sense               */
  __IO uint32_t  IEV;              /*!< GPIO Interrupt Both Edges          */
  __IO uint32_t  IM;               /*!< GPIO Interrupt Event               */
  __IO uint32_t  RIS;              /*!< GPIO Interrupt Mask                */
  __IO uint32_t  MIS;              /*!< GPIO Raw Interrupt Status          */
  __O  uint32_t  ICR;              /*!< GPIO Masked Interrupt Status       */
  __IO uint32_t  AFSEL;            /*!< GPIO Interrupt Clear               */
  __I  uint32_t  RESERVED1[55];    /*!< GPIO Alternate Function Select     */
  __IO uint32_t  DR2R;             /*!< GPIO 2-mA Drive Select             */
  __IO uint32_t  DR4R;             /*!< GPIO 4-mA Drive Select             */
  __IO uint32_t  DR8R;             /*!< GPIO 8-mA Drive Select             */
  __IO uint32_t  ODR;              /*!< GPIO Open Drain Select             */
  __IO uint32_t  PUR;              /*!< GPIO Pull-Up Select                */
  __IO uint32_t  PDR;              /*!< GPIO Pull-Down Select              */
  __IO uint32_t  SLR;              /*!< GPIO Slew Rate Control Select      */
  __IO uint32_t  DEN;              /*!< GPIO Digital Enable                */
  ⋮ __ >> gcc의 추가 기능이다.
} GPIOA_Type;
```

TM4C123GH6PM.h

```
#define GPIOF                   ((GPIOA_Type                *) GPIOF_BASE)
```

매크로이다. 구조체 모양의 포인터이다.

```
#define GPIOF_BASE              0x40025000UL
```

```
GPIOF->DIR = 0x0E;
```


GPIO_BASE + 0x400 = 0x40025400

Using Header File

```
/*  Turns on the red LED and toggles the blue LED. */
#include “TM4C123GH6PM.h”

void delayMs(int n);

int main(void)
{      구조체 모양의 포인터이다.
    SYSCTL->RCGCGPIO |= 0x20;      /* enable clock to GPIOF at clock gating control register */
    GPIOF->DIR = 0x0E;             /* enable the GPIO pins for the LED (PF3, 2 1) as output */
    GPIOF->DEN = 0x0E;             /* enable the GPIO pins for digital function */
    GPIOF->DATA = 0x02;            /* turn on red LED only and leave it on */
    while(1)
    {
        GPIOF->DATA |= 4;     /* turn on blue LED */
        delayMs(1000000);

        GPIOF->DATA &= ~4;    /* turn off blue LED */
        delayMs(1000000);
    }
}

void delayMs(int n)
{
    int i;
    for(i = 0 ; i < n; i++) {}
}
```

00100000

directory

digital enable

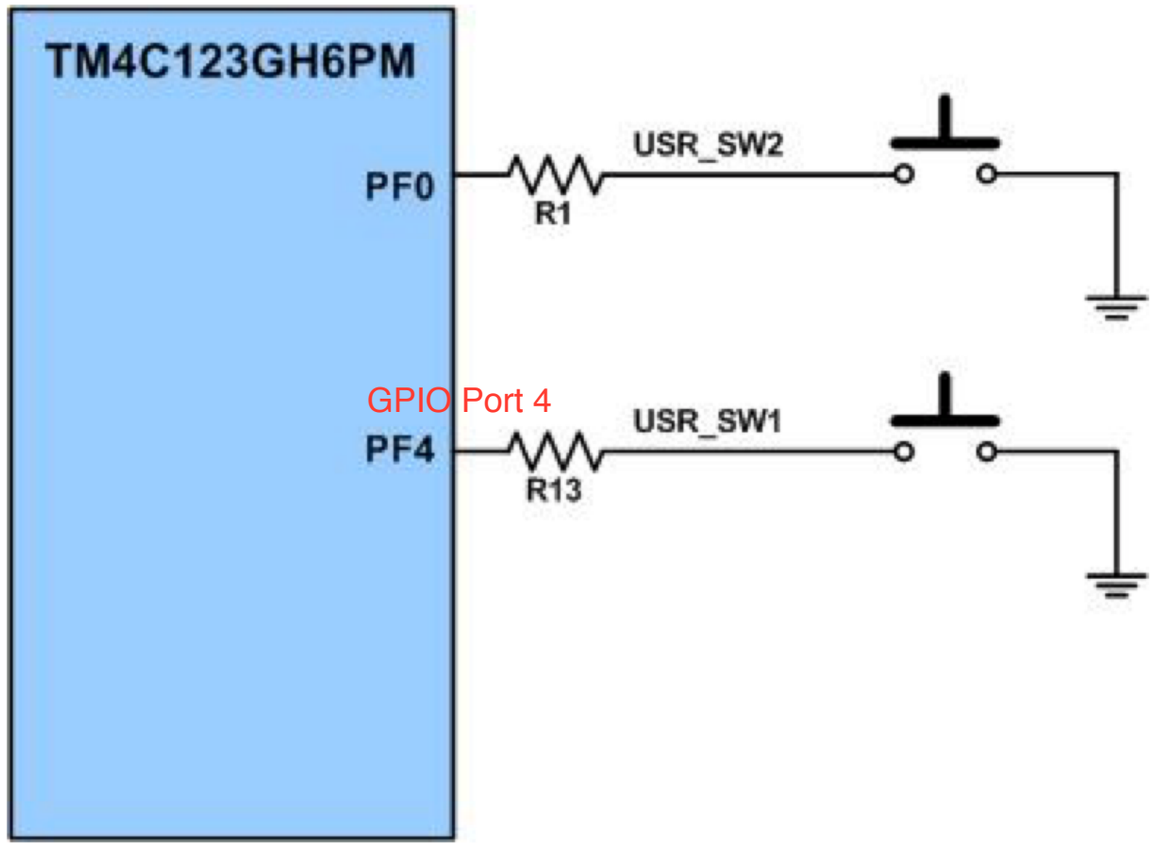
pin 1 red always turn on

rbg led = red 2 blue 4 green 8

Reading Switch

Reading a switch

- **Tina Launchpad에는 2개의 스위치가 내장**
  - SW1 push-button switch는 PF4 핀에 연결되어 있음
  - SW2 push-button switch는 PF0 핀에 연결되어 있음



GPIOPUR

- **SW1과 SW2를 사용하려면 PF0와 PF4에 대해서 internal pull-up resistor를enable해야 함**
- **GPIOPUR 레지스터**

GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
Offset 0x510
Type RW, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PUE							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-

PF4

Note: D0 to D7 are used to enable/disable the pull-up resistor for pins 0 to 7 of the port.
1: Enable pull-up
0: Disable pull-up

SW1을 읽어서 Green LED에 출력하기

1. Enable the clock to PORTF.
2. Set the Direction register PF4 as input and PF3 as output. green
3. Enable the digital I/O feature of PORTF.
4. Enable the pull up resistor option in PUR register.
5. Read SW1 on PORTF and invert the value .
6. Shift right the switch bit (PF4) to green LED bit (PF3) of the value. 4번 핀의 값을 3번 핀으로 출력해야하기 때문에 shift 해준다
7. Write the value to green LED of PORTF.
8. Repeat steps 5 to 7.

SW1을 읽어서 Green LED에 출력하기

```
/*  Reads SW1 and writes the inverse of the value to the green LED. */
#include “TM4C123GH6PM.h”

int main(void)
{
    unsigned int value;
    SYSCTL->RCGCGPIO |= 0x20;    /* enable clock to GPIOF */
    GPIOF->DIR = 0x08;           /* set PORTF3 pin as output (LED) pin */
    /* and PORTF4 as input, SW1 is on PORTF4 */
    GPIOF->DEN = 0x18;  00011000    /* set PORTF pins 4-3 as digital pins */
    GPIOF->PUR = 0x10;  00010000    /* enable pull up for pin 4 */

    while(1)
    {
        value = GPIOF->DATA;     /* read data from PORTF */
        value = ~value;          /* switch is low active; LED is high active */
        value = value >> 1;      /* shift it right to display on green LED */
        GPIOF->DATA = value;     /* put it on the green LED */
    }
}
```

shift

code04.c

SW2를 읽기

- SW2 스위치가 연결된 PORTF0 핀은 NMI (non-maskable interrupt)와 공유되어 있으며 그래서 lock되어 있음 read only
- lock을 풀기위해서는 TM4C123GH6PM.h 파일의 수정이 필요