GPIO (General-Purpose IO)

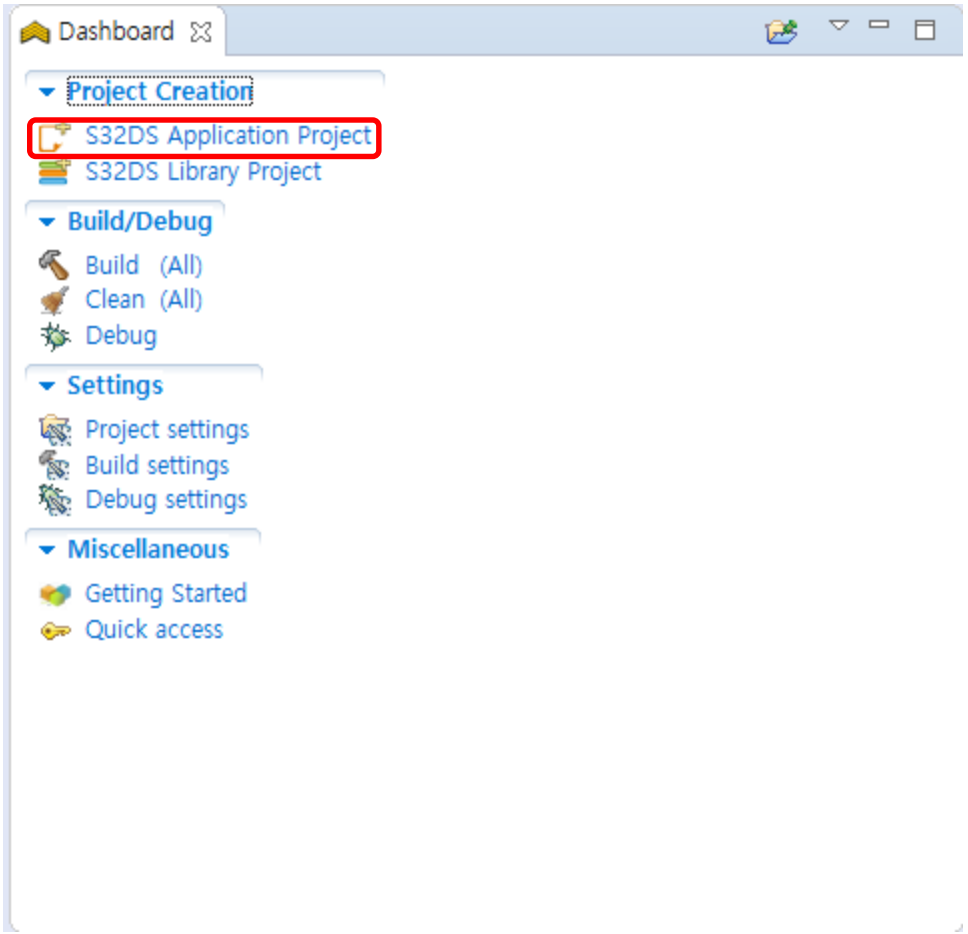
LED Blink / Switch Example

Hyeongrae Kim
Architecture and Compiler for Embedded System LAB.
School of Electronics Engineering, KNU, KOREA
2020-10-08

Setting

- **프로젝트 설정**

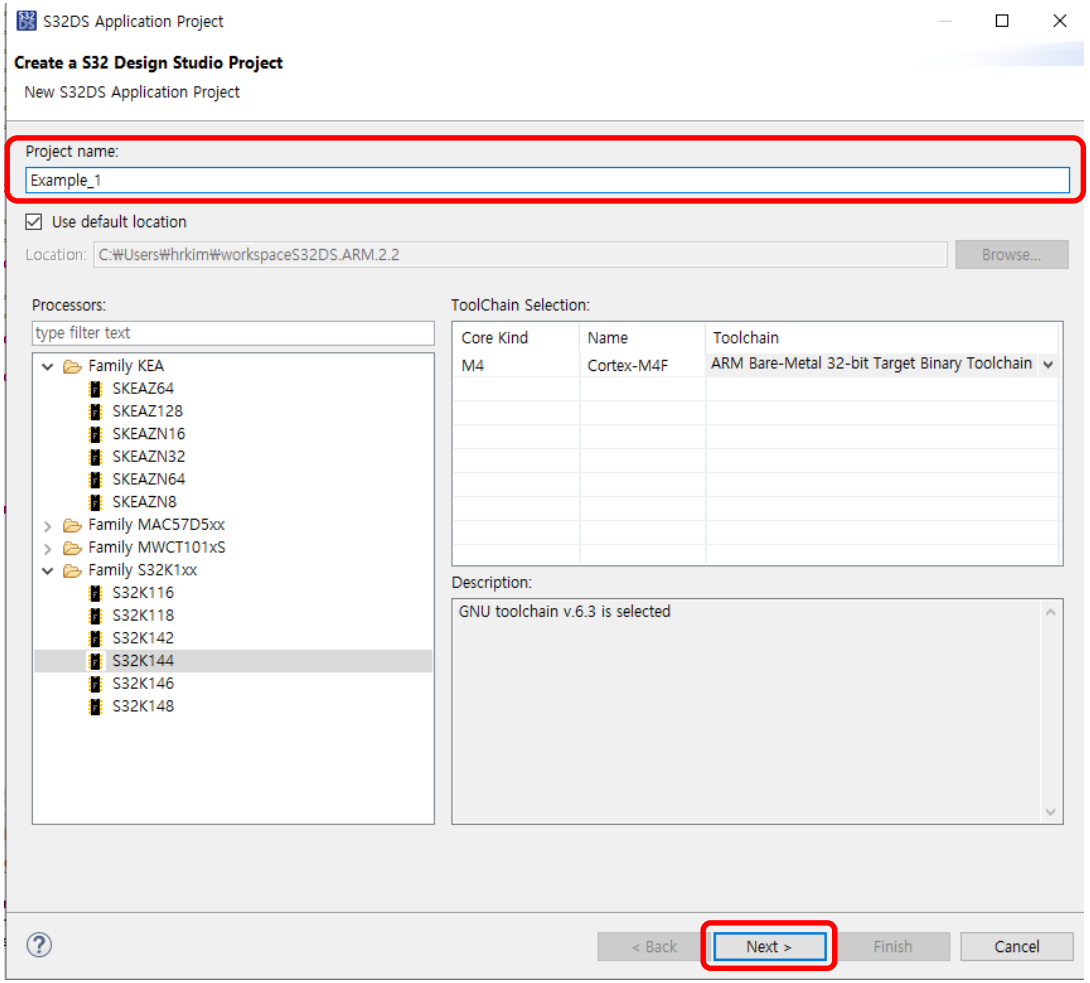
1. S32 Design Studio를 실행하고 왼쪽 하단의 **S32DS Application Project**를 클릭한다.



Setting

- **프로젝트 설정**

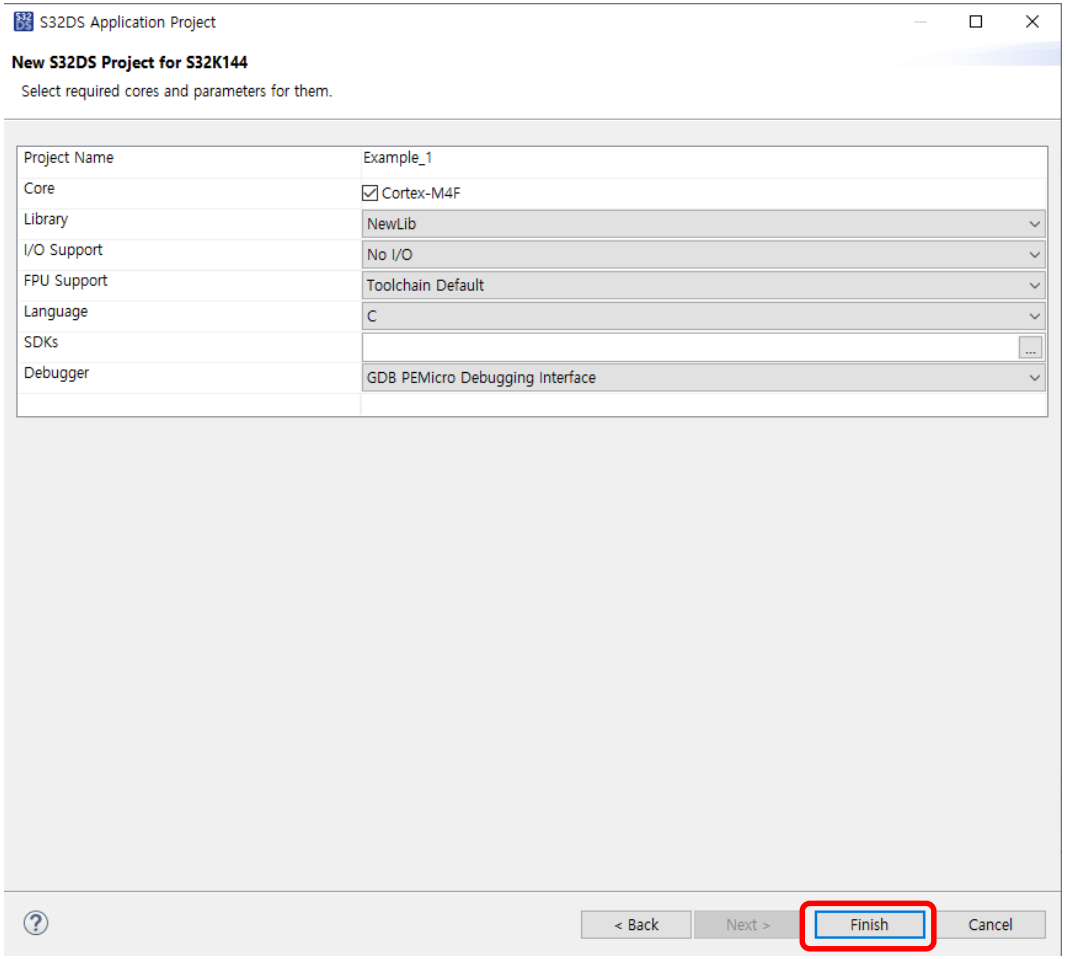
2. Project name을 **입력하고**, Family S32K1xx→S32K144를 **선택한 후** Next를 **클릭한다.**



Setting

- **프로젝트 설정**

  3. Default로 설정한 후 Finish (API를 사용할 때는 SDK 설정이 필요함)



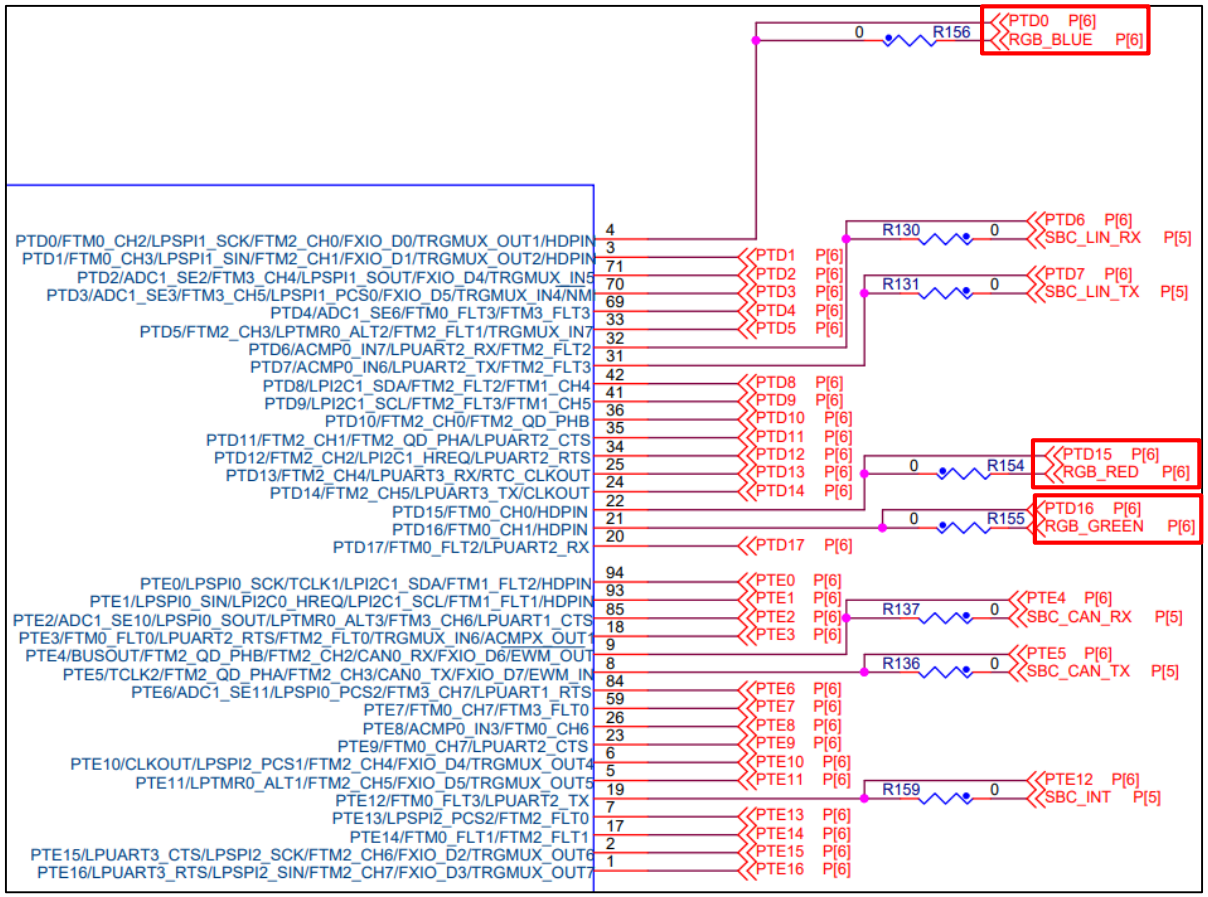
LED Example

- **타이머 주기에 따른** LED toggle
  1. **새로운 예제를 위한 프로젝트를 생성한다.**
  2. **원하는 동작을 위해 레지스터와 메모리에 직접 접근해서 값을 써야한다.**
  3. **해당 예제에서는** LED**를 사용해야하기 때문에** Board Schematic**에서** LED **정보를 파악한다.**
  4. LED**가 연결된** GPIO **모듈의 메모리 맵을 분석한다.**
  5. **보드 정보를 포함한 프로젝트를 생성했을 때, 해당 보드의 메모리 맵 정보가 헤더파일로 추가되기 때문에 이를 참고할 수 있다.**
  6. **분석 결과를 활용해 임베디드 프로그래밍을 한다.**

LED Example

1. Schematic 분석

   ✓ 해당 보드의 Schematic을 확인했을 때, User가 사용할 수 있는 LED는 PTD0/PTD15/PTD16 핀에 연결되어 있고, 각각 Blue, Red, Green이다.



LED Example

2. Data sheet 분석 : IO 설정

   ✓ LED를 사용하기 위해 연결된 핀의 IO 설정이 필요하다.

   ① PCC 및 Peripheral Register를 통해 Peripheral Clock 및 Peripheral 설정을 한다 (기본 설정을 따름).

   ② PCC_PORTx Register를 통해 핀을 포함하는 Port의 Clock 설정을 한다.

   ③ PORTx_PCRn을 통해 해당 핀의 Peripheral Pin 설정을 한다.

12.1.3 I/O configuration sequence

1. Ensure pins for the peripheral are in tristate sate (default out of reset).
2. Initialize peripheral clock in the Peripheral Clock Controller register(PCC) and peripheral specific clocking configurations.
3. Configure the peripheral
4. Initialize port clock for the peripheral pins in the Peripheral Clock Controller register (PCC_PORTx).
5. Configure the peripheral pins mux and features in the Port Control and Interrupts register (PORTx_PCRn).
6. Start communication

LED Example

2. Data sheet 분석 : Port Clock 설정

✓ PCC_PORTx Register에서 CGC bit를 set하여 Clock enable 설정을 한다.

PCC base address: 4006_5000h

Offset	Register	Width (In bits)	Access	Reset value
80h	PCC FTFC Register (PCC_FTFC)	32	RW	C000_0000h
84h	PCC DMAMUX Register (PCC_DMAMUX)	32	RW	8000_0000h
90h	PCC FlexCAN0 Register (PCC_FlexCAN0)	32	RW	8000_0000h
94h	PCC FlexCAN1 Register (PCC_FlexCAN1)	32	RW	8000_0000h
98h	PCC FTM3 Register (PCC_FTM3)	32	RW	8000_0000h
9Ch	PCC ADC1 Register (PCC_ADC1)	32	RW	8000_0000h
ACh	PCC FlexCAN2 Register (PCC_FlexCAN2)	32	RW	8000_0000h
B0h	PCC LPSPI0 Register (PCC_LPSPI0)	32	RW	8000_0000h
B4h	PCC LPSPI1 Register (PCC_LPSPI1)	32	RW	8000_0000h
B8h	PCC LPSPI2 Register (PCC_LPSPI2)	32	RW	8000_0000h
C4h	PCC PDB1 Register (PCC_PDB1)	32	RW	8000_0000h
C8h	PCC CRC Register (PCC_CRC)	32	RW	8000_0000h
D8h	PCC PDB0 Register (PCC_PDB0)	32	RW	8000_0000h
DCh	PCC LPIT Register (PCC_LPIT)	32	RW	8000_0000h
E0h	PCC FTM0 Register (PCC_FTM0)	32	RW	8000_0000h
E4h	PCC FTM1 Register (PCC_FTM1)	32	RW	8000_0000h
E8h	PCC FTM2 Register (PCC_FTM2)	32	RW	8000_0000h
ECh	PCC ADC0 Register (PCC_ADC0)	32	RW	8000_0000h
F4h	PCC RTC Register (PCC_RTC)	32	RW	8000_0000h
100h	PCC LPTMR0 Register (PCC_LPTMR0)	32	RW	8000_0000h
124h	PCC PORTA Register (PCC_PORTA)	32	RW	8000_0000h
128h	PCC PORTB Register (PCC_PORTB)	32	RW	8000_0000h
12Ch	PCC PORTC Register (PCC_PORTC)	32	RW	8000_0000h
130h	PCC PORTD Register (PCC_PORTD)	32	RW	8000_0000h
134h	PCC PORTE Register (PCC_PORTE)	32	RW	8000_0000h
150h	PCC SAI0 Register (PCC_SAI0)	32	RW	8000_0000h
154h	PCC SAI1 Register (PCC_SAI1)	32	RW	8000_0000h
168h	PCC FlexIO Register (PCC_FlexIO)	32	RW	8000_0000h
184h	PCC EWM Register (PCC_EWM)	32	RW	8000_0000h
198h	PCC LPI2C0 Register (PCC_LPI2C0)	32	RW	8000_0000h
19Ch	PCC LPI2C1 Register (PCC_LPI2C1)	32	RW	8000_0000h
1A8h	PCC LPUART0 Register (PCC_LPUART0)	32	RW	8000_0000h
1ACh	PCC LPUART1 Register (PCC_LPUART1)	32	RW	8000_0000h
1B0h	PCC LPUART2 Register (PCC_LPUART2)	32	RW	8000_0000h
1B8h	PCC FTM4 Register (PCC_FTM4)	32	RW	8000_0000h
1BCh	PCC FTM5 Register (PCC_FTM5)	32	RW	8000_0000h
1C0h	PCC FTM6 Register (PCC_FTM6)	32	RW	8000_0000h
1C4h	PCC FTM7 Register (PCC_FTM7)	32	RW	8000_0000h
1CCh	PCC CMP0 Register (PCC_CMP0)	32	RW	8000_0000h
1D8h	PCC QSPI Register (PCC_QSPI)	32	RW	8000_0000h
1E4h	PCC ENET Register (PCC_ENET)	32	RW	8000_0000h

29.6.24.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PR	CGC	Reserved	0		0			0							
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												0		0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

29.6.24.4 Fields

Field	Function
31	Present
PR	This bit shows whether the peripheral is present on this device. 0b - Peripheral is not present. 1b - Peripheral is present.
30	Clock Gate Control
CGC	This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified. 0b - Clock disabled. The current clock selection and divider options are not locked and can be modified. 1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified.

PCC_PORTD Register 구조

LED Example

2. Data sheet 분석 : Peripheral Pin 설정

✓ PORTx_PCRn에서 MUX bits를 설정하여 해당 핀이 GPIO로 사용될 수 있도록 한다.

PORT memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Pin Control Register n (PORT_PCR0)	32	R/W	See section	12.5.1/215
4	Pin Control Register n (PORT_PCR1)	32	R/W	See section	12.5.1/215
8	Pin Control Register n (PORT_PCR2)	32	R/W	See section	12.5.1/215
C	Pin Control Register n (PORT_PCR3)	32	R/W	See section	12.5.1/215
10	Pin Control Register n (PORT_PCR4)	32	R/W	See section	12.5.1/215
14	Pin Control Register n (PORT_PCR5)	32	R/W	See section	12.5.1/215
18	Pin Control Register n (PORT_PCR6)	32	R/W	See section	12.5.1/215
1C	Pin Control Register n (PORT_PCR7)	32	R/W	See section	12.5.1/215
20	Pin Control Register n (PORT_PCR8)	32	R/W	See section	12.5.1/215
24	Pin Control Register n (PORT_PCR9)	32	R/W	See section	12.5.1/215
28	Pin Control Register n (PORT_PCR10)	32	R/W	See section	12.5.1/215
2C	Pin Control Register n (PORT_PCR11)	32	R/W	See section	12.5.1/215
30	Pin Control Register n (PORT_PCR12)	32	R/W	See section	12.5.1/215
34	Pin Control Register n (PORT_PCR13)	32	R/W	See section	12.5.1/215
38	Pin Control Register n (PORT_PCR14)	32	R/W	See section	12.5.1/215
3C	Pin Control Register n (PORT_PCR15)	32	R/W	See section	12.5.1/215
40	Pin Control Register n (PORT_PCR16)	32	R/W	See section	12.5.1/215
44	Pin Control Register n (PORT_PCR17)	32	R/W	See section	12.5.1/215
48	Pin Control Register n (PORT_PCR18)	32	R/W	See section	12.5.1/215
4C	Pin Control Register n (PORT_PCR19)	32	R/W	See section	12.5.1/215
50	Pin Control Register n (PORT_PCR20)	32	R/W	See section	12.5.1/215
54	Pin Control Register n (PORT_PCR21)	32	R/W	See section	12.5.1/215
58	Pin Control Register n (PORT_PCR22)	32	R/W	See section	12.5.1/215
5C	Pin Control Register n (PORT_PCR23)	32	R/W	See section	12.5.1/215
60	Pin Control Register n (PORT_PCR24)	32	R/W	See section	12.5.1/215
64	Pin Control Register n (PORT_PCR25)	32	R/W	See section	12.5.1/215
68	Pin Control Register n (PORT_PCR26)	32	R/W	See section	12.5.1/215
6C	Pin Control Register n (PORT_PCR27)	32	R/W	See section	12.5.1/215
70	Pin Control Register n (PORT_PCR28)	32	R/W	See section	12.5.1/215
74	Pin Control Register n (PORT_PCR29)	32	R/W	See section	12.5.1/215
78	Pin Control Register n (PORT_PCR30)	32	R/W	See section	12.5.1/215
7C	Pin Control Register n (PORT_PCR31)	32	R/W	See section	12.5.1/215
80	Global Pin Control Low Register (PORT_GPCLR)	32	W (always reads 0)	0000_0000h	12.5.2/218
84	Global Pin Control High Register (PORT_GPCHR)	32	W (always reads 0)	0000_0000h	12.5.3/218

Address: 0h base + 0h offset + (4d × i), where i=0d to 31d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0							ISF	0				IRQC			
W								w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LK	0				MUX			0	DSE	Reserved	PFE	0	Reserved	PE	PS
W																
Reset	0	0	0	0	0	*	*	*	0	*	0	*	0	0	*	*

10–8 MUX	Pin Mux Control Not all pins support all pin muxing slots. Unimplemented pin muxing slots are reserved and may result in configuring the pin for a different pin muxing slot. The corresponding pin is configured in the following pin muxing slot as follows: 000 Pin disabled (Alternative 0) (analog). 001 Alternative 1 (GPIO). 010 Alternative 2 (chip-specific).

PORTx_PCR0 구조

LED Example

2. Data sheet 분석 : GPIO 설정

✓ GPIO로 사용되는 핀은 다음과 같은 특징을 가진다.

① PDDR을 통해 Input/Output 설정을 한다.

② Input으로 설정된 경우, PDIR을 통해 핀의 Logic level을 읽을 수 있다.

③ Output으로 설정된 경우, PDOR에 대응하는 set/clear/toggle register를 통해 Logic level을 쓸 수 있다.

13.2.1 Features

Features of the GPIO module include:

- Port Data Input register visible in all digital pin-multiplexing modes
- Port Data Output register with corresponding set/clear/toggle registers
- Port Data Direction register

LED Example

2. Data sheet 분석 : Output 설정

   ✓ GPIO로 사용되는 핀의 Input/Output 설정을 한다. LED는 Output으로 설정해야 한다.

13.3.1.1 GPIO memory map

GPIOA base address: 400F_F000h

GPIOB base address: 400F_F040h

GPIOC base address: 400F_F080h

GPIOD base address: 400F_F0C0h

GPIOE base address: 400F_F100h

Offset	Register	Width (In bits)	Access	Reset value
0h	Port Data Output Register (PDOR)	32	RW	0000_0000h
4h	Port Set Output Register (PSOR)	32	WORZ	0000_0000h
8h	Port Clear Output Register (PCOR)	32	WORZ	0000_0000h
Ch	Port Toggle Output Register (PTOR)	32	WORZ	0000_0000h
10h	Port Data Input Register (PDIR)	32	RO	0000_0000h
14h	Port Data Direction Register (PDDR)	32	RW	0000_0000h
18h	Port Input Disable Register (PIDR)	32	RW	0000_0000h

13.3.1.7.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PDD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PDD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

13.3.1.7.4 Fields

Field	Function
31-0 PDD	Port Data Direction Configures individual port pins for input or output. 0b - Pin is configured as general-purpose input, for the GPIO function. The pin will be high-Z if the port input is disabled in GPIOx_PIDR register. 1b - Pin is configured as general-purpose output, for the GPIO function.

PORTx_PDDR 구조

LED Example

3. 프로그래밍

1) LED가 연결된 핀에 대한 IO 설정을 한다.

```
PCC-> PCCn[PCC_PORTD_INDEX] |= PCC_PCCn_CGC_MASK;/* Enable clocks to peripherals (PORT modules) */
                                                 /* Enable clock to PORT D*/
PORTD->PCR[0] &= ~PORT_PCR_MUX_MASK;
PORTD->PCR[0] |= PORT_PCR_MUX(1);    /* Port D0: MUX = GPIO */
```

IO 설정 코드

```
/** PORT - Size of Registers Arrays */
#define PORT_PCR_COUNT                           32u

/** PORT - Register Layout Typedef */
typedef struct {
  __IO uint32_t PCR[PORT_PCR_COUNT];              /**< Pin Control Register n, array offset: 0x0, array step: 0x4 */
  __O  uint32_t GPCLR;                            /**< Global Pin Control Low Register, offset: 0x80 */
  __O  uint32_t GPCHR;                            /**< Global Pin Control High Register, offset: 0x84 */
  __O  uint32_t GICLR;                            /**< Global Interrupt Control Low Register, offset: 0x88 */
  __O  uint32_t GICHR;                            /**< Global Interrupt Control High Register, offset: 0x8C */
       uint8_t RESERVED_0[16];
  __IO uint32_t ISFR;                             /**< Interrupt Status Flag Register, offset: 0xA0 */
       uint8_t RESERVED_1[28];
  __IO uint32_t DFER;                             /**< Digital Filter Enable Register, offset: 0xC0 */
  __IO uint32_t DFCR;                             /**< Digital Filter Clock Register, offset: 0xC4 */
  __IO uint32_t DFWR;                             /**< Digital Filter Width Register, offset: 0xC8 */
} PORT_Type, *PORT_MemMapPtr;

 /** Number of instances of the PORT module. */
#define PORT_INSTANCE_COUNT                      (5u)


/* PORT - Peripheral instance base addresses */
/** Peripheral PORTA base address */
#define PORTA_BASE                               (0x40049000u)
/** Peripheral PORTA base pointer */
#define PORTA                                    ((PORT_Type *)PORTA_BASE)
/** Peripheral PORTB base address */
#define PORTB_BASE                               (0x4004A000u)
/** Peripheral PORTB base pointer */
#define PORTB                                    ((PORT_Type *)PORTB_BASE)
/** Peripheral PORTC base address */
#define PORTC_BASE                               (0x4004B000u)
/** Peripheral PORTC base pointer */
#define PORTC                                    ((PORT_Type *)PORTC_BASE)
/** Peripheral PORTD base address */
#define PORTD_BASE                               (0x4004C000u)
/** Peripheral PORTD base pointer */
#define PORTD                                    ((PORT_Type *)PORTD_BASE)
```

```
/** PCC - Size of Registers Arrays */
#define PCC_PCCn_COUNT                           116u

/** PCC - Register Layout Typedef */
typedef struct {
  __IO uint32_t PCCn[PCC_PCCn_COUNT];              /**< PCC Reserved Reg
} PCC_Type, *PCC_MemMapPtr;

 /** Number of instances of the PCC module. */
#define PCC_INSTANCE_COUNT                       (1u)


/* PCC - Peripheral instance base addresses */
/** Peripheral PCC base address */
#define PCC_BASE                                 (0x40065000u)
/** Peripheral PCC base pointer */
#define PCC                                      ((PCC_Type *)PCC_BASE)
/** Array initializer of PCC peripheral base addresses */
#define PCC_BASE_ADDRS                           { PCC_BASE }
/** Array initializer of PCC peripheral base pointers */
#define PCC_BASE_PTRS                            { PCC }
```

'S32K144.h'에 정의된 메모리 맵

LED Example

3. 프로그래밍

2) LED가 연결된 핀에 대한 GPIO 설정 (Output 설정)을 한다.

```
                                    /* Configure port D0 as GPIO output (LED on EVB) */
PTD->PDDR |= 1<<PTD0;               /* Port D0: Data Direction= output */
```

GPIO 설정 코드

```
/** GPIO - Register Layout Typedef */
typedef struct {
  __IO uint32_t PDOR;                              /**< Port Data Output Register, offset: 0x0 */
  __O  uint32_t PSOR;                              /**< Port Set Output Register, offset: 0x4 */
  __O  uint32_t PCOR;                              /**< Port Clear Output Register, offset: 0x8 */
  __O  uint32_t PTOR;                              /**< Port Toggle Output Register, offset: 0xC */
  __I  uint32_t PDIR;                              /**< Port Data Input Register, offset: 0x10 */
  __IO uint32_t PDDR;                              /**< Port Data Direction Register, offset: 0x14 */
  __IO uint32_t PIDR;                              /**< Port Input Disable Register, offset: 0x18 */
} GPIO_Type, *GPIO_MemMapPtr;

 /** Number of instances of the GPIO module. */
#define GPIO_INSTANCE_COUNT                      (5u)


/* GPIO - Peripheral instance base addresses */
/** Peripheral PTA base address */
#define PTA_BASE                                 (0x400FF000u)
/** Peripheral PTA base pointer */
#define PTA                                      ((GPIO_Type *)PTA_BASE)
/** Peripheral PTB base address */
#define PTB_BASE                                 (0x400FF040u)
/** Peripheral PTB base pointer */
#define PTB                                      ((GPIO_Type *)PTB_BASE)
/** Peripheral PTC base address */
#define PTC_BASE                                 (0x400FF080u)
/** Peripheral PTC base pointer */
#define PTC                                      ((GPIO_Type *)PTC_BASE)
/** Peripheral PTD base address */
#define PTD_BASE                                 (0x400FF0C0u)
/** Peripheral PTD base pointer */
#define PTD                                      ((GPIO_Type *)PTD_BASE)
```

'S32K144.h'에 정의된 메모리 맵

LED Example

3. **프로그래밍**

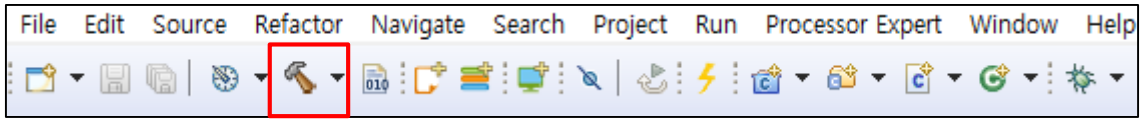
   3) **동작에 따라 'main' 함수를 구현한다.**

```
#include "device_registers.h"

/*! Port PTD0, bit 0: FRDM EVB output to blue LED
 */
#define PTD0  0


int main(void)
{

    /*!
     * Initialization
     * ================================================
     */



    PCC-> PCCn[PCC_PORTD_INDEX] = PCC_PCCn_CGC_MASK; /* Enable clock to PORT D*/

    PORTD->PCR[0] = PORT_PCR_MUX(1); /* Port D0: MUX = GPIO */

                                /* Configure port D0 as GPIO output (LED on EVB) */
    PTD->PDDR |= 1<<PTD0;       /* Port D0: Data Direction= output */



    /*!
     * Infinite for:
     * =======================
     */
        for(;;)
        {
            int cycles = 720000;
            while(cycles--);
            PTD-> PTOR |= 1<<PTD0;

        }
}
```

LED Example

4. 빌드

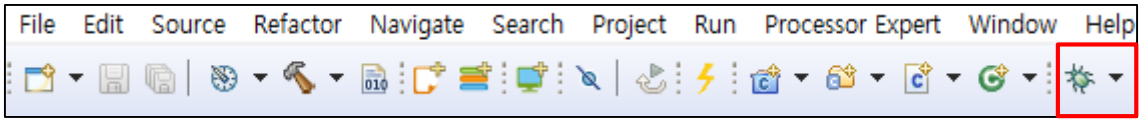
✓ 상단의 메뉴에서 망치 모양 툴을 눌러 빌드한다.



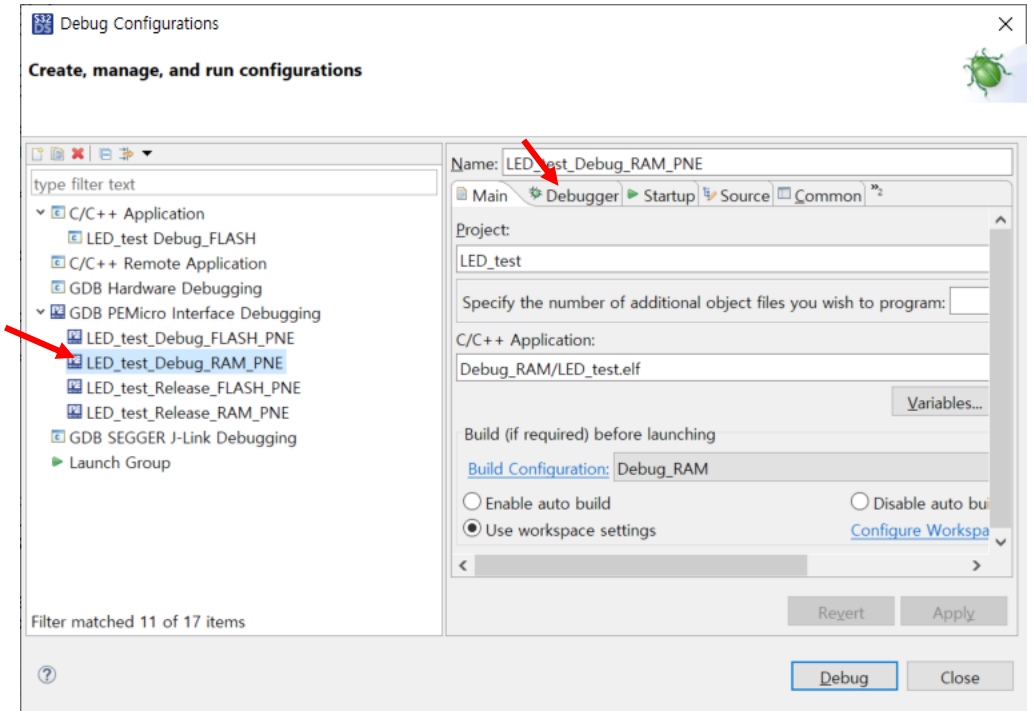
LED Example

5. 디버그

✓ 상단의 메뉴에서 디버그 툴의 ▼을 눌러 Debug configuration 메뉴에 들어간다.



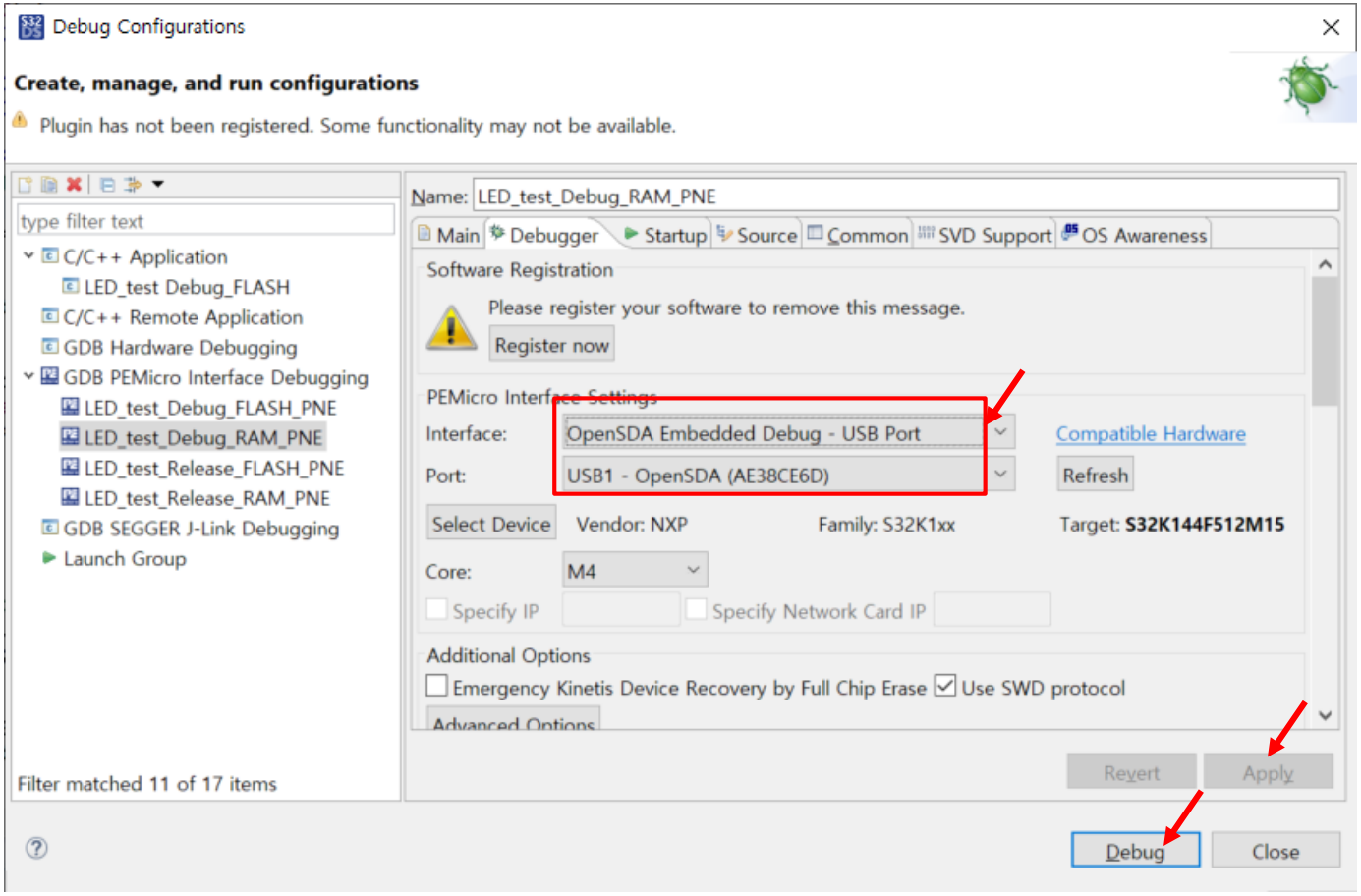
✓ GDB PEMicro Interface Debugging 하위에서 작업 프로젝트_Debug_RAM를 선택한후, Debugger 탭으로 이동한다.



LED Example

5. 디버그

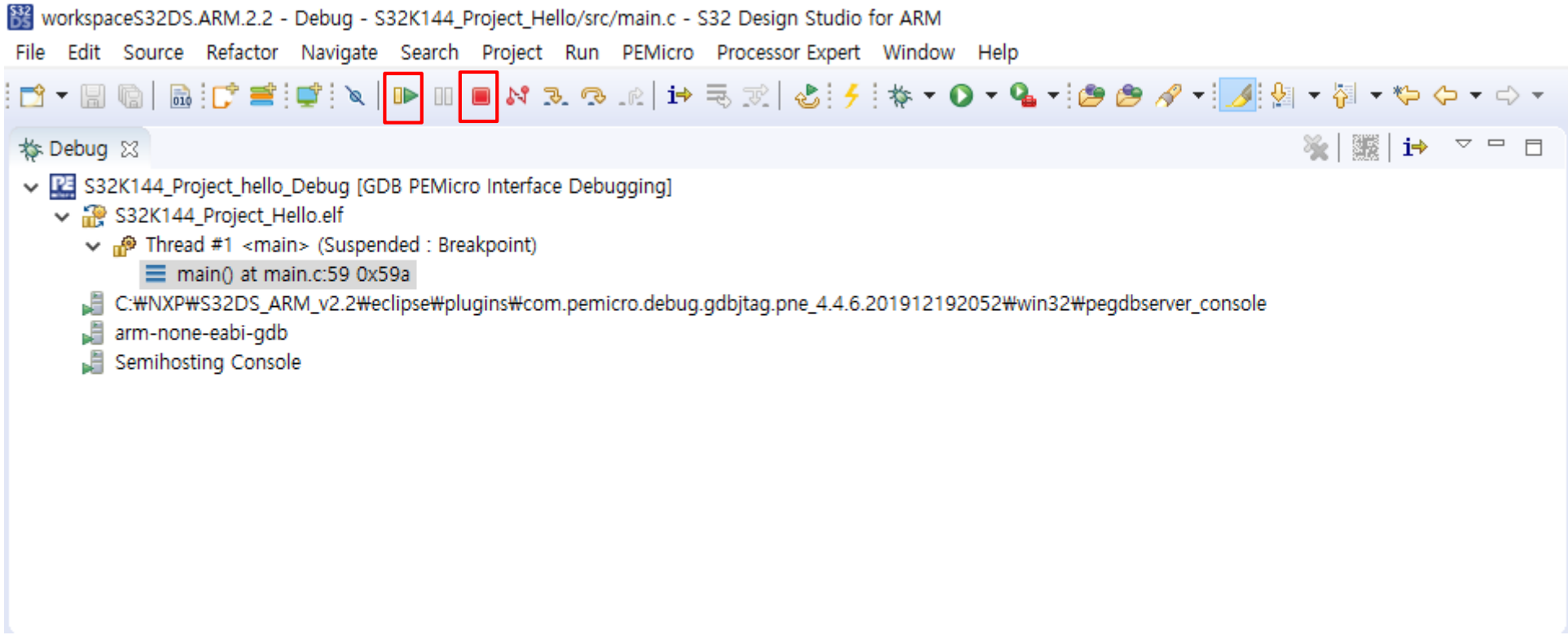
✓ Debugger탭 내에 인터페이스 선택부분에서 OpdenSDA 선택한 후, 적용 그리고 Debug버튼을 누른다.



LED Example

5. 디버그

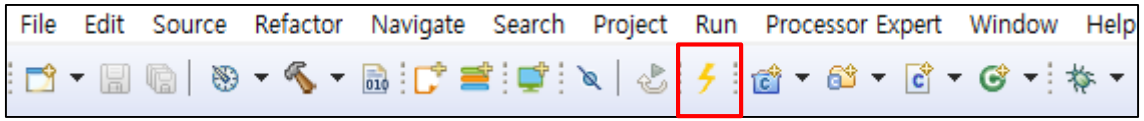
   ✓ Resume 및 Terminate 버튼을 통해 디버그를 시작하고 끝낼 수 있다. (Resume 시 일정 주기로 LED Blue가 Toggle됨을 확인)



LED Example

6. 다운로드

   ✓ 상단의 메뉴에서 번개 모양 툴을 눌러 보드의 플래시에 다운로드한다. (디버그 인터페이스와 동일)



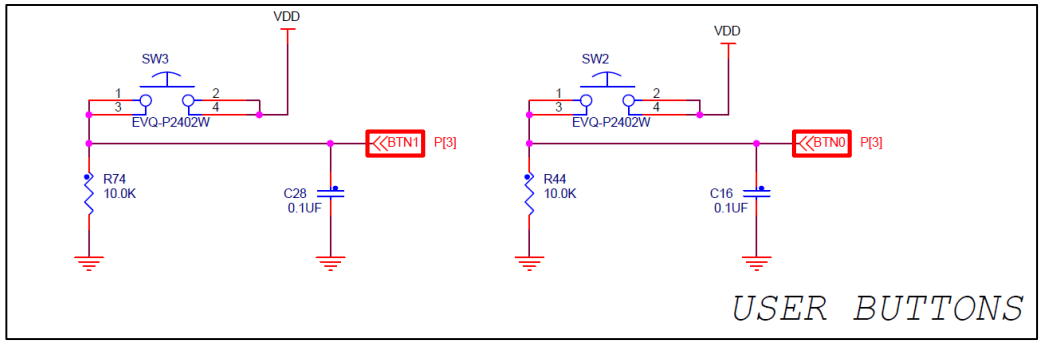
Switch Example

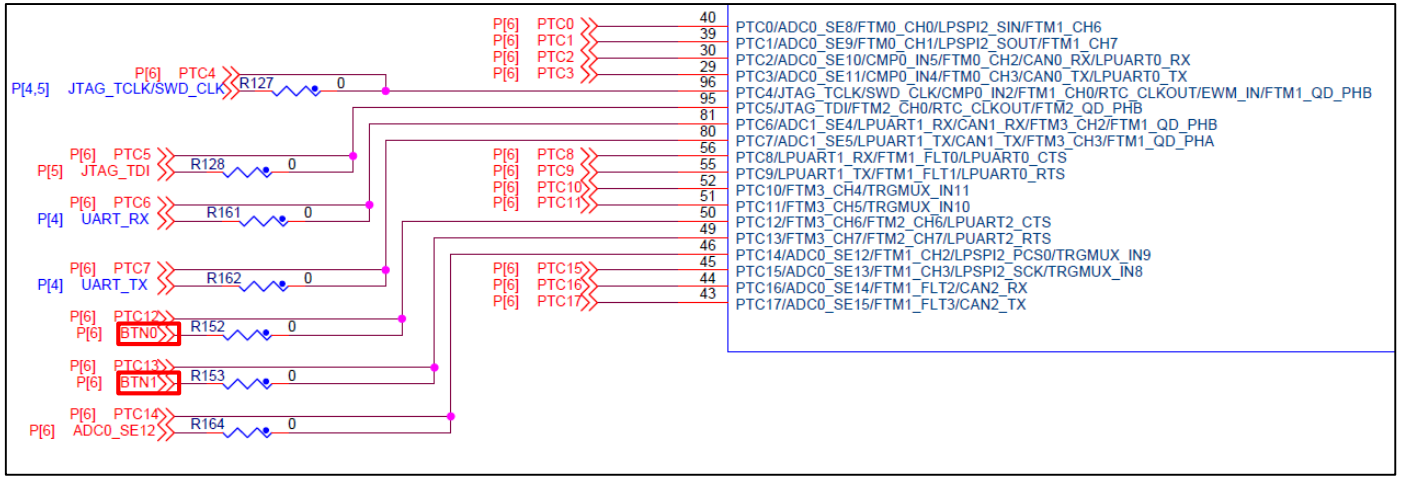
- Switch를 이용한 LED on/off 제어
  1. 새로운 예제를 위한 프로젝트를 생성한다.
  2. 원하는 동작을 위해 레지스터와 메모리에 직접 접근해서 값을 써야한다.
  3. 해당 예제에서는 Switch를 사용해야하기 때문에 Board Schematic에서 Switch 정보를 파악한다.
  4. Switch가 연결된 GPIO 모듈의 메모리 맵을 분석한다.
  5. 보드 정보를 포함한 프로젝트를 생성했을 때, 해당 보드의 메모리 맵 정보가 헤더파일로 추가되기 때문에 이를 참고할 수 있다.
  6. 분석 결과를 활용해 임베디드 프로그래밍을 한다.

Switch Example

1. Schematic 분석

   ✓ 해당 보드의 Schematic을 확인했을 때, User가 사용할 수 있는 Switch2/Switch3는 PTC12/PTC13 핀에 연결되어 있다.





Switch Example

2. Data sheet 분석 : IO 설정

   ✓ Switch를 사용하기 위해 연결된 핀의 IO 설정이 필요하다.

   ① PCC 및 Peripheral Register를 통해 Peripheral Clock 및 Peripheral 설정을 한다 (기본 설정을 따름).

   ② PCC_PORTx Register를 통해 핀을 포함하는 Port의 Clock 설정을 한다.

   ③ PORTx_PCRn을 통해 해당 핀의 Peripheral Pin 설정을 한다.

12.1.3 I/O configuration sequence

1. Ensure pins for the peripheral are in tristate sate (default out of reset).
2. Initialize peripheral clock in the Peripheral Clock Controller register(PCC) and peripheral specific clocking configurations.
3. Configure the peripheral
4. Initialize port clock for the peripheral pins in the Peripheral Clock Controller register (PCC_PORTx).
5. Configure the peripheral pins mux and features in the Port Control and Interrupts register (PORTx_PCRn).
6. Start communication

Switch Example

2. Data sheet 분석 : Port Clock 설정

✓ PCC_PORTx Register에서 CGC bit를 set하여 Clock enable 설정을 한다.

PCC base address: 4006_5000h

Offset	Register	Width (In bits)	Access	Reset value
80h	PCC FTFC Register (PCC_FTFC)	32	RW	C000_0000h
84h	PCC DMAMUX Register (PCC_DMAMUX)	32	RW	8000_0000h
90h	PCC FlexCAN0 Register (PCC_FlexCAN0)	32	RW	8000_0000h
94h	PCC FlexCAN1 Register (PCC_FlexCAN1)	32	RW	8000_0000h
98h	PCC FTM3 Register (PCC_FTM3)	32	RW	8000_0000h
9Ch	PCC ADC1 Register (PCC_ADC1)	32	RW	8000_0000h
ACh	PCC FlexCAN2 Register (PCC_FlexCAN2)	32	RW	8000_0000h
B0h	PCC LPSPI0 Register (PCC_LPSPI0)	32	RW	8000_0000h
B4h	PCC LPSPI1 Register (PCC_LPSPI1)	32	RW	8000_0000h
B8h	PCC LPSPI2 Register (PCC_LPSPI2)	32	RW	8000_0000h
C4h	PCC PDB1 Register (PCC_PDB1)	32	RW	8000_0000h
C8h	PCC CRC Register (PCC_CRC)	32	RW	8000_0000h
D8h	PCC PDB0 Register (PCC_PDB0)	32	RW	8000_0000h
DCh	PCC LPIT Register (PCC_LPIT)	32	RW	8000_0000h
E0h	PCC FTM0 Register (PCC_FTM0)	32	RW	8000_0000h
E4h	PCC FTM1 Register (PCC_FTM1)	32	RW	8000_0000h
E8h	PCC FTM2 Register (PCC_FTM2)	32	RW	8000_0000h
ECh	PCC ADC0 Register (PCC_ADC0)	32	RW	8000_0000h
F4h	PCC RTC Register (PCC_RTC)	32	RW	8000_0000h
100h	PCC LPTMR0 Register (PCC_LPTMR0)	32	RW	8000_0000h
124h	PCC PORTA Register (PCC_PORTA)	32	RW	8000_0000h
128h	PCC PORTB Register (PCC_PORTB)	32	RW	8000_0000h
12Ch	PCC PORTC Register (PCC_PORTC)	32	RW	8000_0000h
130h	PCC PORTD Register (PCC_PORTD)	32	RW	8000_0000h
134h	PCC PORTE Register (PCC_PORTE)	32	RW	8000_0000h
150h	PCC SAI0 Register (PCC_SAI0)	32	RW	8000_0000h
154h	PCC SAI1 Register (PCC_SAI1)	32	RW	8000_0000h
168h	PCC FlexIO Register (PCC_FlexIO)	32	RW	8000_0000h
184h	PCC EWM Register (PCC_EWM)	32	RW	8000_0000h
198h	PCC LPI2C0 Register (PCC_LPI2C0)	32	RW	8000_0000h
19Ch	PCC LPI2C1 Register (PCC_LPI2C1)	32	RW	8000_0000h
1A8h	PCC LPUART0 Register (PCC_LPUART0)	32	RW	8000_0000h
1ACh	PCC LPUART1 Register (PCC_LPUART1)	32	RW	8000_0000h
1B0h	PCC LPUART2 Register (PCC_LPUART2)	32	RW	8000_0000h
1B8h	PCC FTM4 Register (PCC_FTM4)	32	RW	8000_0000h
1BCh	PCC FTM5 Register (PCC_FTM5)	32	RW	8000_0000h
1C0h	PCC FTM6 Register (PCC_FTM6)	32	RW	8000_0000h
1C4h	PCC FTM7 Register (PCC_FTM7)	32	RW	8000_0000h
1CCh	PCC CMP0 Register (PCC_CMP0)	32	RW	8000_0000h
1D8h	PCC QSPI Register (PCC_QSPI)	32	RW	8000_0000h
1E4h	PCC ENET Register (PCC_ENET)	32	RW	8000_0000h

29.6.24.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PR	CGC	Reserved	0		0			0							
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												0		0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

29.6.24.4 Fields

Field	Function
31	Present
PR	This bit shows whether the peripheral is present on this device. 0b - Peripheral is not present. 1b - Peripheral is present.
30	Clock Gate Control
CGC	This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified. 0b - Clock disabled. The current clock selection and divider options are not locked and can be modified. 1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified.

PCC_PORTC Register 구조

Switch Example

2. Data sheet 분석 : Peripheral Pin 설정

✓ PORTx_PCRn에서 MUX bits를 설정하여 해당 핀이 GPIO로 사용될 수 있도록 한다.

PORT memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Pin Control Register n (PORT_PCR0)	32	R/W	See section	12.5.1/215
4	Pin Control Register n (PORT_PCR1)	32	R/W	See section	12.5.1/215
8	Pin Control Register n (PORT_PCR2)	32	R/W	See section	12.5.1/215
C	Pin Control Register n (PORT_PCR3)	32	R/W	See section	12.5.1/215
10	Pin Control Register n (PORT_PCR4)	32	R/W	See section	12.5.1/215
14	Pin Control Register n (PORT_PCR5)	32	R/W	See section	12.5.1/215
18	Pin Control Register n (PORT_PCR6)	32	R/W	See section	12.5.1/215
1C	Pin Control Register n (PORT_PCR7)	32	R/W	See section	12.5.1/215
20	Pin Control Register n (PORT_PCR8)	32	R/W	See section	12.5.1/215
24	Pin Control Register n (PORT_PCR9)	32	R/W	See section	12.5.1/215
28	Pin Control Register n (PORT_PCR10)	32	R/W	See section	12.5.1/215
2C	Pin Control Register n (PORT_PCR11)	32	R/W	See section	12.5.1/215
30	Pin Control Register n (PORT_PCR12)	32	R/W	See section	12.5.1/215
34	Pin Control Register n (PORT_PCR13)	32	R/W	See section	12.5.1/215
38	Pin Control Register n (PORT_PCR14)	32	R/W	See section	12.5.1/215
3C	Pin Control Register n (PORT_PCR15)	32	R/W	See section	12.5.1/215
40	Pin Control Register n (PORT_PCR16)	32	R/W	See section	12.5.1/215
44	Pin Control Register n (PORT_PCR17)	32	R/W	See section	12.5.1/215
48	Pin Control Register n (PORT_PCR18)	32	R/W	See section	12.5.1/215
4C	Pin Control Register n (PORT_PCR19)	32	R/W	See section	12.5.1/215
50	Pin Control Register n (PORT_PCR20)	32	R/W	See section	12.5.1/215
54	Pin Control Register n (PORT_PCR21)	32	R/W	See section	12.5.1/215
58	Pin Control Register n (PORT_PCR22)	32	R/W	See section	12.5.1/215
5C	Pin Control Register n (PORT_PCR23)	32	R/W	See section	12.5.1/215
60	Pin Control Register n (PORT_PCR24)	32	R/W	See section	12.5.1/215
64	Pin Control Register n (PORT_PCR25)	32	R/W	See section	12.5.1/215
68	Pin Control Register n (PORT_PCR26)	32	R/W	See section	12.5.1/215
6C	Pin Control Register n (PORT_PCR27)	32	R/W	See section	12.5.1/215
70	Pin Control Register n (PORT_PCR28)	32	R/W	See section	12.5.1/215
74	Pin Control Register n (PORT_PCR29)	32	R/W	See section	12.5.1/215
78	Pin Control Register n (PORT_PCR30)	32	R/W	See section	12.5.1/215
7C	Pin Control Register n (PORT_PCR31)	32	R/W	See section	12.5.1/215
80	Global Pin Control Low Register (PORT_GPCLR)	32	W (always reads 0)	0000_0000h	12.5.2/218
84	Global Pin Control High Register (PORT_GPCHR)	32	W (always reads 0)	0000_0000h	12.5.3/218

Address: 0h base + 0h offset + (4d × i), where i=0d to 31d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0							ISF	0				IRQC			
W								w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LK	0				MUX			0	DSE	Reserved	PFE	0	Reserved	PE	PS
W																
Reset	0	0	0	0	0	*	*	*	0	*	0	*	0	0	*	*

10–8 MUX	Pin Mux Control Not all pins support all pin muxing slots. Unimplemented pin muxing slots are reserved and may result in configuring the pin for a different pin muxing slot. The corresponding pin is configured in the following pin muxing slot as follows: 000 Pin disabled (Alternative 0) (analog). 001 Alternative 1 (GPIO). 010 Alternative 2 (chip-specific).

PORTx_PCR12 구조

Switch Example

2. Data sheet 분석 : GPIO 설정

✓ GPIO로 사용되는 핀은 다음과 같은 특징을 가진다.

① PDDR을 통해 Input/Output 설정을 한다.

② Input으로 설정된 경우, PDIR을 통해 핀의 Logic level을 읽을 수 있다.

③ Output으로 설정된 경우, PDOR에 대응하는 set/clear/toggle register를 통해 Logic level을 쓸 수 있다.

13.2.1 Features

Features of the GPIO module include:

- Port Data Input register visible in all digital pin-multiplexing modes
- Port Data Output register with corresponding set/clear/toggle registers
- Port Data Direction register

Switch Example

2. Data sheet 분석 : Input 설정

   ✓ GPIO로 사용되는 핀의 Input/Output 설정을 한다. Switch 신호를 입력받아야 하기 때문에 Input으로 설정해야 한다.

13.3.1.1 GPIO memory map

GPIOA base address: 400F_F000h

GPIOB base address: 400F_F040h

GPIOC base address: 400F_F080h

GPIOD base address: 400F_F0C0h

GPIOE base address: 400F_F100h

Offset	Register	Width (In bits)	Access	Reset value
0h	Port Data Output Register (PDOR)	32	RW	0000_0000h
4h	Port Set Output Register (PSOR)	32	WORZ	0000_0000h
8h	Port Clear Output Register (PCOR)	32	WORZ	0000_0000h
Ch	Port Toggle Output Register (PTOR)	32	WORZ	0000_0000h
10h	Port Data Input Register (PDIR)	32	RO	0000_0000h
14h	Port Data Direction Register (PDDR)	32	RW	0000_0000h
18h	Port Input Disable Register (PIDR)	32	RW	0000_0000h

13.3.1.7.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PDD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PDD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

13.3.1.7.4 Fields

Field	Function
31-0 PDD	Port Data Direction Configures individual port pins for input or output. 0b - Pin is configured as general-purpose input, for the GPIO function. The pin will be high-Z if the port input is disabled in GPIOx_PIDR register. 1b - Pin is configured as general-purpose output, for the GPIO function.

PORTx_PDDR 구조

Switch Example

3. 프로그래밍

1) Switch2가 연결된 핀에 대한 IO 설정을 한다.

```
PCC-> PCCn[PCC_PORTC_INDEX] |= PCC_PCCn_CGC_MASK;/* Enable clocks to peripherals (PORT modules) */
                                                 /* Enable clock to PORT C*/
PORTC->PCR[12] &= ~PORT_PCR_MUX_MASK;
PORTC->PCR[12] |= PORT_PCR_MUX(1);  /* Port C12: MUX = GPIO */
```

IO 설정 코드

```
/** PORT - Size of Registers Arrays */
#define PORT_PCR_COUNT                           32u

/** PORT - Register Layout Typedef */
typedef struct {
  __IO uint32_t PCR[PORT_PCR_COUNT];              /**< Pin Control Register n, array offset: 0x0, array step: 0x4 */
  __O  uint32_t GPCLR;                            /**< Global Pin Control Low Register, offset: 0x80 */
  __O  uint32_t GPCHR;                            /**< Global Pin Control High Register, offset: 0x84 */
  __O  uint32_t GICLR;                            /**< Global Interrupt Control Low Register, offset: 0x88 */
  __O  uint32_t GICHR;                            /**< Global Interrupt Control High Register, offset: 0x8C */
       uint8_t RESERVED_0[16];
  __IO uint32_t ISFR;                             /**< Interrupt Status Flag Register, offset: 0xA0 */
       uint8_t RESERVED_1[28];
  __IO uint32_t DFER;                             /**< Digital Filter Enable Register, offset: 0xC0 */
  __IO uint32_t DFCR;                             /**< Digital Filter Clock Register, offset: 0xC4 */
  __IO uint32_t DFWR;                             /**< Digital Filter Width Register, offset: 0xC8 */
} PORT_Type, *PORT_MemMapPtr;

 /** Number of instances of the PORT module. */
#define PORT_INSTANCE_COUNT                      (5u)


/* PORT - Peripheral instance base addresses */
/** Peripheral PORTA base address */
#define PORTA_BASE                               (0x40049000u)
/** Peripheral PORTA base pointer */
#define PORTA                                    ((PORT_Type *)PORTA_BASE)
/** Peripheral PORTB base address */
#define PORTB_BASE                               (0x4004A000u)
/** Peripheral PORTB base pointer */
#define PORTB                                    ((PORT_Type *)PORTB_BASE)
/** Peripheral PORTC base address */
#define PORTC_BASE                               (0x4004B000u)
/** Peripheral PORTC base pointer */
#define PORTC                                    ((PORT_Type *)PORTC_BASE)
/** Peripheral PORTD base address */
#define PORTD_BASE                               (0x4004C000u)
/** Peripheral PORTD base pointer */
#define PORTD                                    ((PORT_Type *)PORTD_BASE)
```

```
/** PCC - Size of Registers Arrays */
#define PCC_PCCn_COUNT                           116u

/** PCC - Register Layout Typedef */
typedef struct {
  __IO uint32_t PCCn[PCC_PCCn_COUNT];              /**< PCC Reserved Reg
} PCC_Type, *PCC_MemMapPtr;

 /** Number of instances of the PCC module. */
#define PCC_INSTANCE_COUNT                       (1u)


/* PCC - Peripheral instance base addresses */
/** Peripheral PCC base address */
#define PCC_BASE                                 (0x40065000u)
/** Peripheral PCC base pointer */
#define PCC                                      ((PCC_Type *)PCC_BASE)
/** Array initializer of PCC peripheral base addresses */
#define PCC_BASE_ADDRS                           { PCC_BASE }
/** Array initializer of PCC peripheral base pointers */
#define PCC_BASE_PTRS                            { PCC }
```

'S32K144.h'에 정의된 메모리 맵

Switch Example

3. 프로그래밍

2) Switch2가 연결된 핀에 대한 GPIO 설정 (Input 설정)을 한다.

```
                                /* Configure port C12 as GPIO input (BTN 0 [SW2] on EVB) */
PTC->PDDR &= ~(1<<PTC12);       /* Port C12: Data Direction= input (default) */
```

GPIO 설정 코드

```
/** GPIO - Register Layout Typedef */
typedef struct {
  __IO uint32_t PDOR;                              /**< Port Data Output Register, offset: 0x0 */
  __O  uint32_t PSOR;                              /**< Port Set Output Register, offset: 0x4 */
  __O  uint32_t PCOR;                              /**< Port Clear Output Register, offset: 0x8 */
  __O  uint32_t PTOR;                              /**< Port Toggle Output Register, offset: 0xC */
  __I  uint32_t PDIR;                              /**< Port Data Input Register, offset: 0x10 */
  __IO uint32_t PDDR;                              /**< Port Data Direction Register, offset: 0x14 */
  __IO uint32_t PIDR;                              /**< Port Input Disable Register, offset: 0x18 */
} GPIO_Type, *GPIO_MemMapPtr;

 /** Number of instances of the GPIO module. */
#define GPIO_INSTANCE_COUNT                      (5u)


/* GPIO - Peripheral instance base addresses */
/** Peripheral PTA base address */
#define PTA_BASE                                 (0x400FF000u)
/** Peripheral PTA base pointer */
#define PTA                                      ((GPIO_Type *)PTA_BASE)
/** Peripheral PTB base address */
#define PTB_BASE                                 (0x400FF040u)
/** Peripheral PTB base pointer */
#define PTB                                      ((GPIO_Type *)PTB_BASE)
/** Peripheral PTC base address */
#define PTC_BASE                                 (0x400FF080u)
/** Peripheral PTC base pointer */
#define PTC                                      ((GPIO_Type *)PTC_BASE)
/** Peripheral PTD base address */
#define PTD_BASE                                 (0x400FF0C0u)
/** Peripheral PTD base pointer */
#define PTD                                      ((GPIO_Type *)PTD_BASE)
```

'S32K144.h'에 정의된 메모리 맵

Switch Example

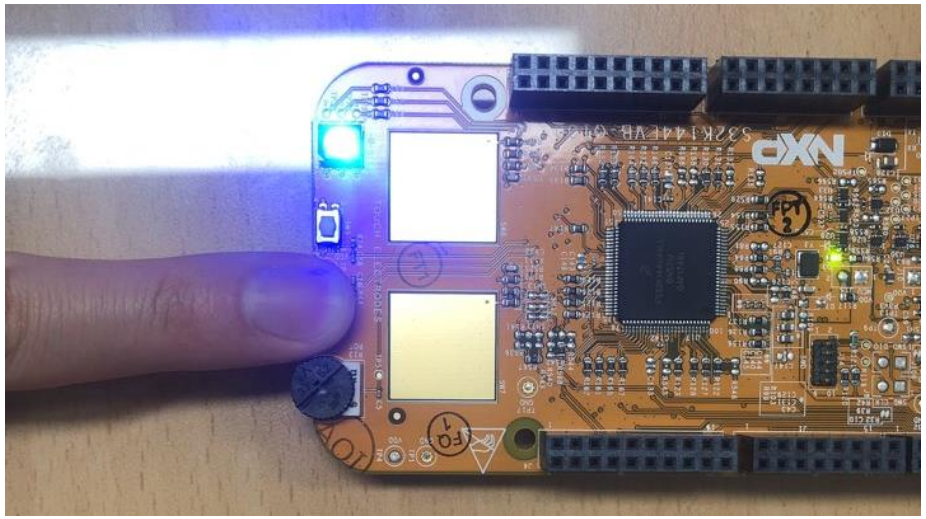
3. **프로그래밍**

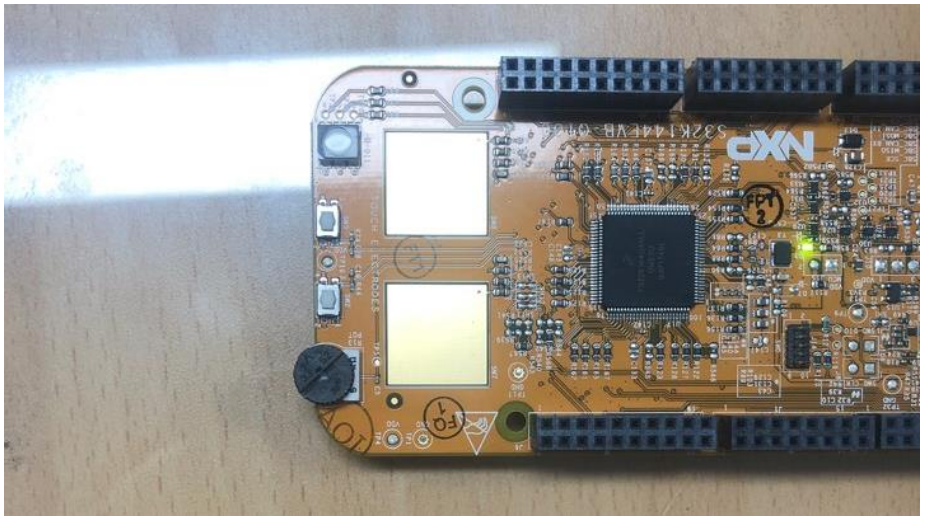
   3) **동작에 따라 'main' 함수를 구현한다.**

```
/*! Port PTD0, bit 0: FRDM EVB output to blue LED
 */
#define PTD0  0

/*! Port PTC12, bit 12: FRDM EVB input from BTN0 [SW2]
 */
#define PTC12 12


int main(void)
{
    int counter = 0;

     * Pins definitions

     * Initialization

    WDOG_disable();/* Disable Watchdog in case it is not done in startup code */

    PCC-> PCCn[PCC_PORTC_INDEX] = PCC_PCCn_CGC_MASK;/* Enable clocks to peripherals (PORT modules) */
                                                    /* Enable clock to PORT C*/
    PORTC->PCR[12] = PORT_PCR_MUX(1)
                                /* Configure port C12 as GPIO input (BTN 0 [SW2] on EVB) */
    PTC->PDDR &= ~(1<<PTC12);   /* Port C12: Data Direction= input (default) */


    PCC-> PCCn[PCC_PORTD_INDEX] = PCC_PCCn_CGC_MASK;/* Enable clocks to peripherals (PORT modules) */
                                                    /* Enable clock to PORT D*/
    PORTD->PCR[0] = PORT_PCR_MUX(1); /* Port D0: MUX = GPIO */

                                /* Configure port D0 as GPIO output (LED on EVB) */
    PTD->PDDR |= 1<<PTD0;       /* Port D0: Data Direction= output */

     * Infinite for:
       for(;;)
       {

           /*! -If Pad Data Input = 1 (BTN0 [SW2] pushed)
            *
            * Clear Output on port D0 (LED on)
            *
           */

           if (PTC->PDIR & (1<<PTC12)) {
               PTD-> PCOR |= 1<<PTD0;
           }
           else {                           /* -If BTN0 was not pushed*/
               PTD-> PSOR |= 1<<PTD0;       /*     Set Output on port D0 (LED off) */
           }
           counter++;
       }
}
```

Switch Example

- **동작 모습**
  - ✓ **빌드, 플래쉬 다운로드, 디버그 후 Switch를 눌렀을 때 LED가 켜지는 것을 확인한다.**





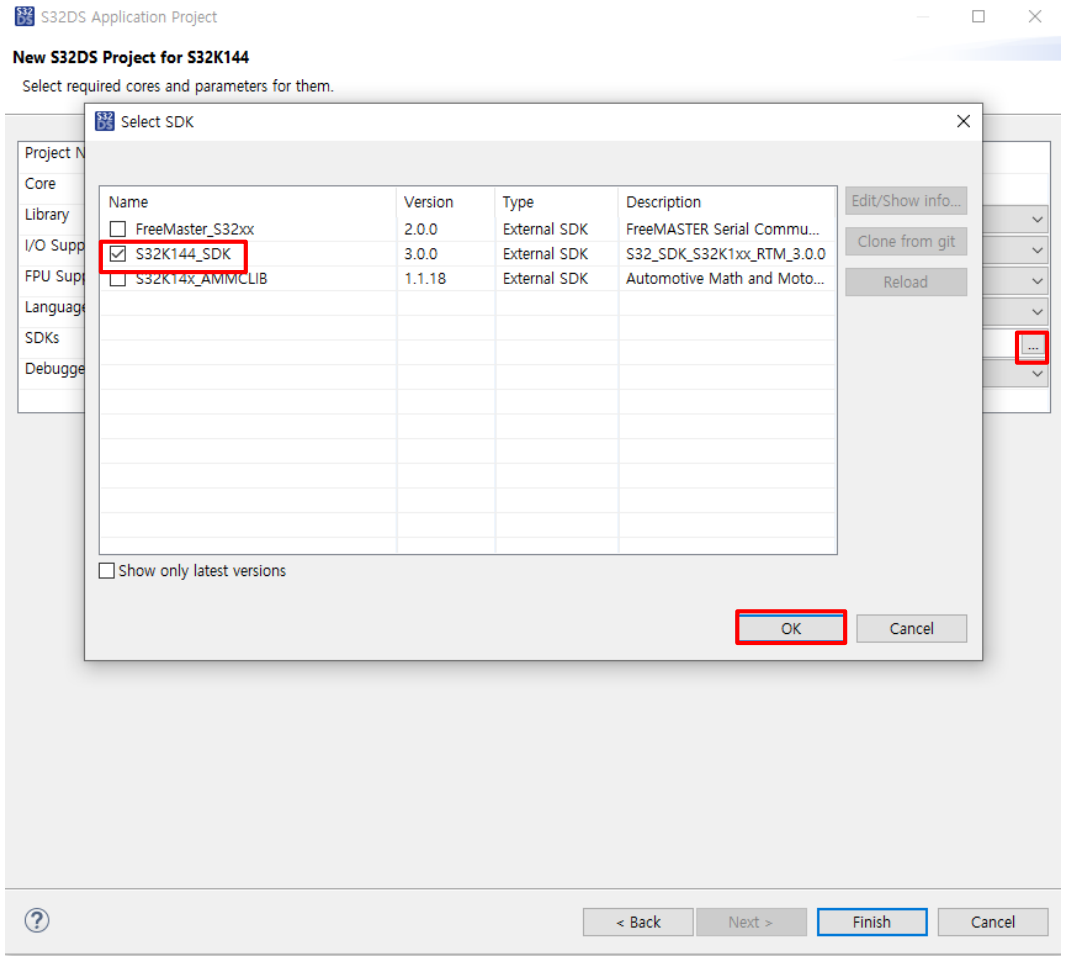
Switch Example with API

- Switch를 이용한 LED on/off 제어
  1. 새로운 예제를 위해 SDK가 포함된 프로젝트를 생성한다.
  2. 하드웨어 동작을 위해 레지스터와 메모리에 직접 접근하는 대신 주어진 API를 통해 접근한다.
  3. API를 활용해 임베디드 프로그래밍을 한다.

Switch Example with API

1. **프로젝트 생성**

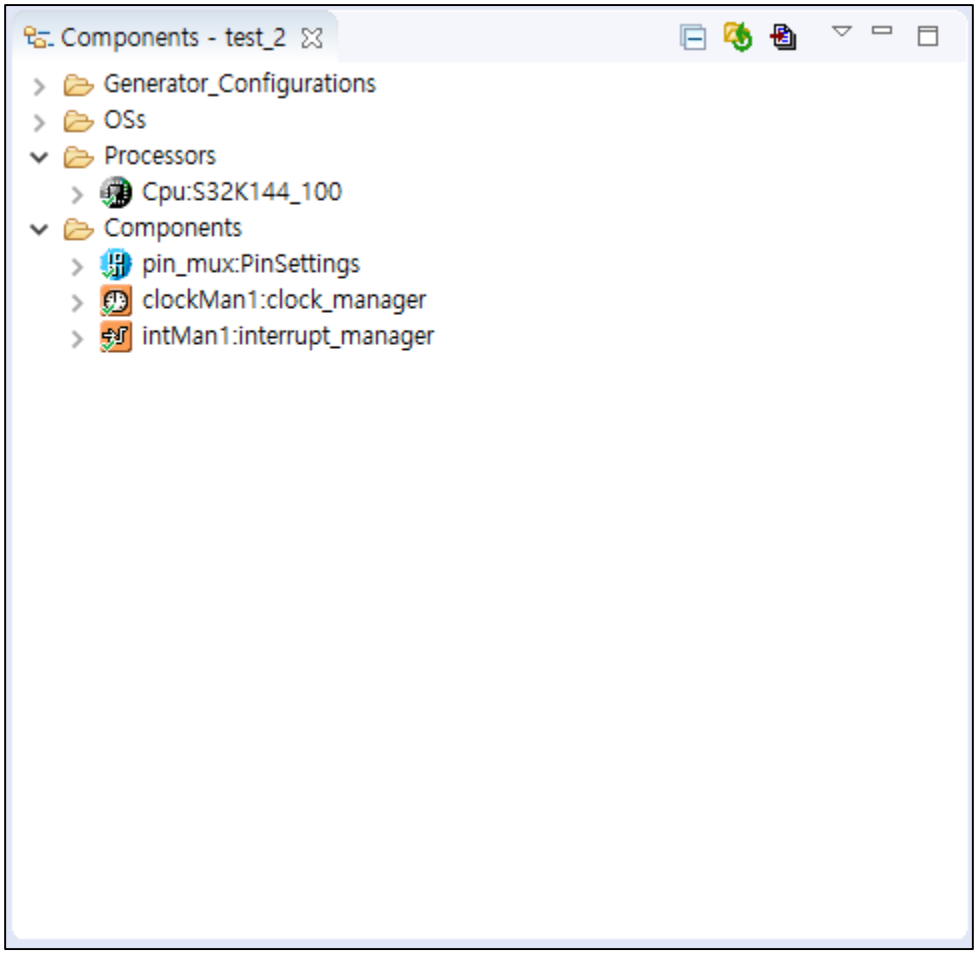
   ✓ **앞서 설명한 프로젝트 생성 과정 마지막에** SDKs 의 '...'**을 눌러** S32K144_SDK**를 체크하여 완료한다.**



Switch Example with API

1. **프로젝트 생성**

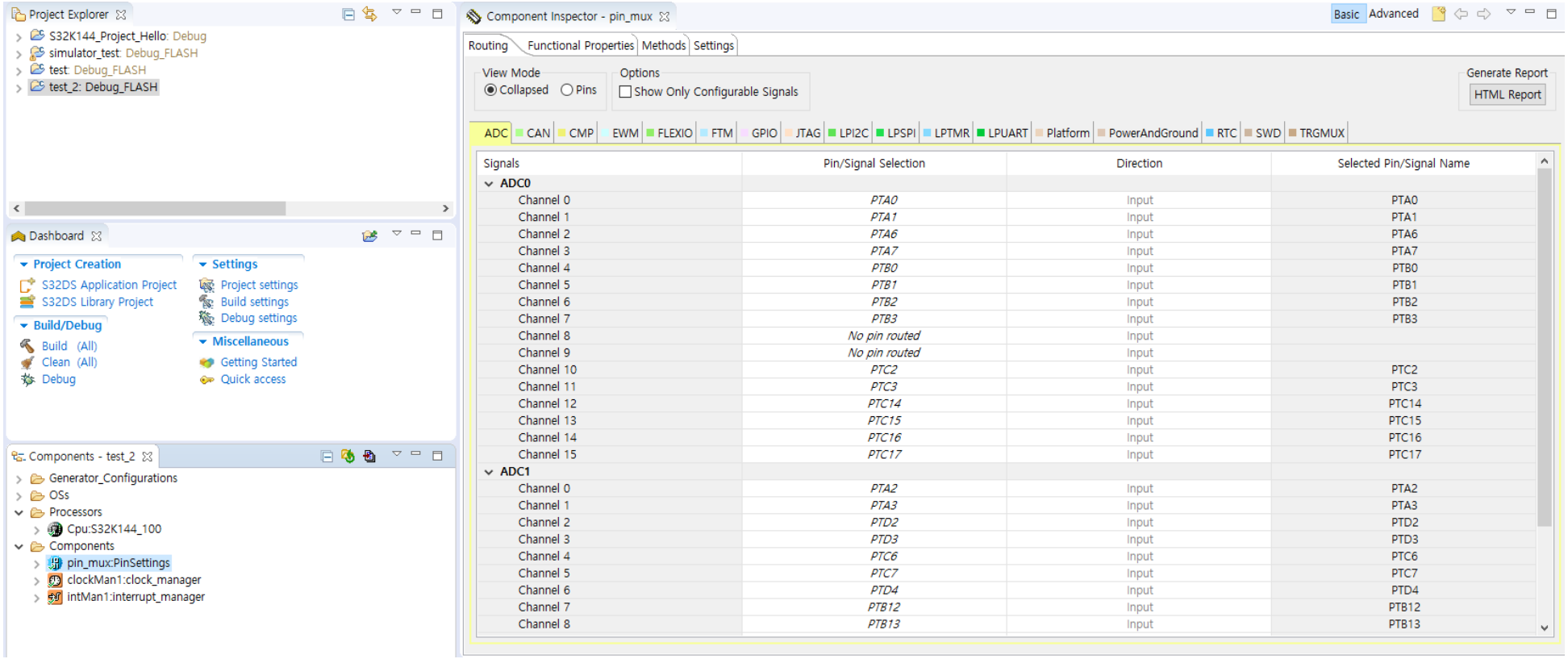
   ✓ **프로젝트를 선택했을 때 왼쪽 아래에 다음과 같은** Component **창이 뜨는 것을 확인한다.**



Switch Example with API

2. 핀 세팅

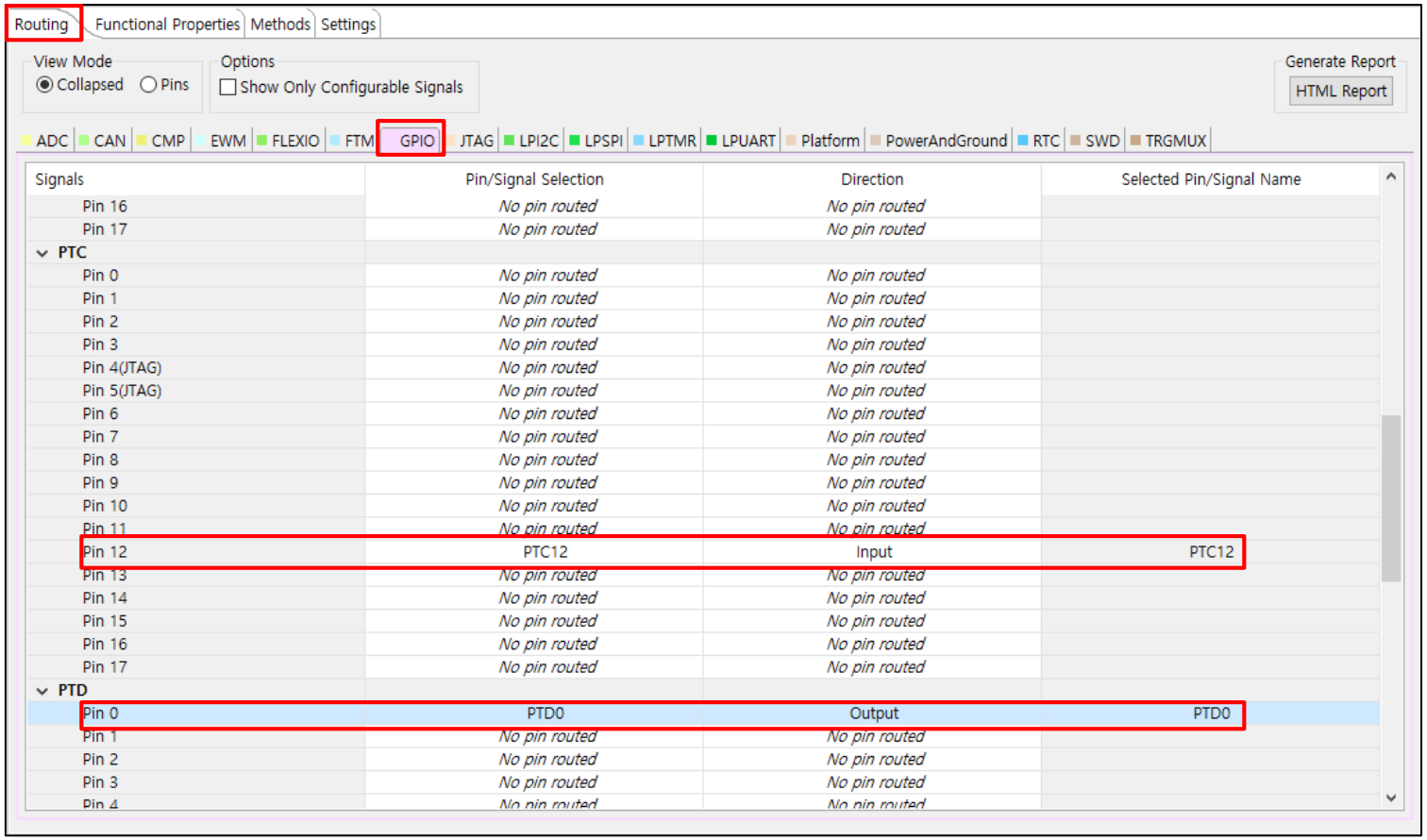
✓ Components→pin_mux:PinSettings를 더블클릭하면 다음과 같이 핀을 세팅할 수 있는 환경이 뜬다.



Switch Example with API

2. **핀 세팅**

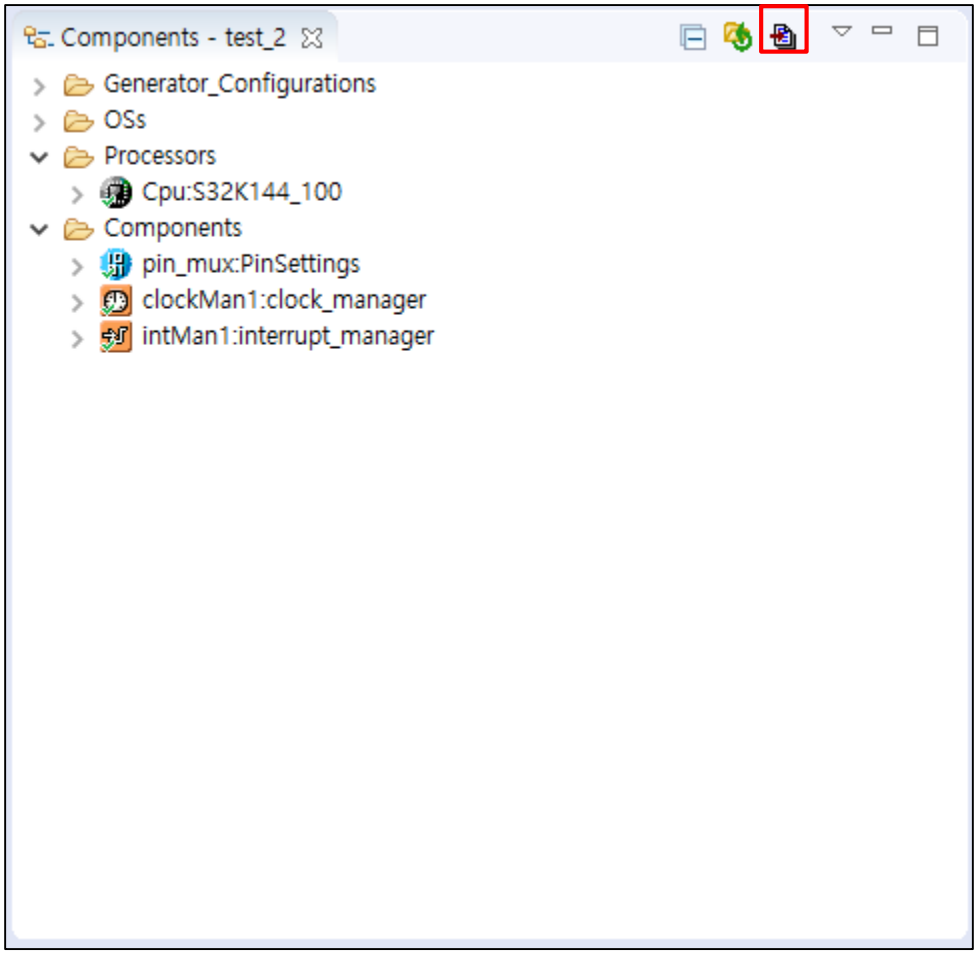
✓ Routing → GPIO → PTC12(SW2) : Input/PTD0(LED_Blue) : Output**으로 설정한다.**



Switch Example with API

2. **핀 세팅**

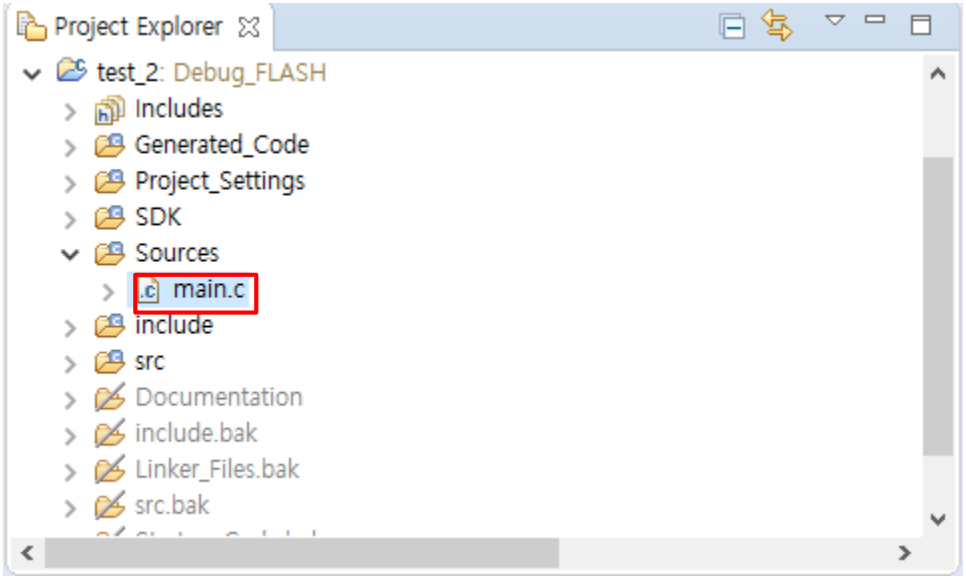
   ✓ **세팅을 저장하고, 왼쪽 하단의** Components view**에서** Generate Processor Expert Code**를 클릭한다.**



Switch Example with API

2. **핀 세팅**

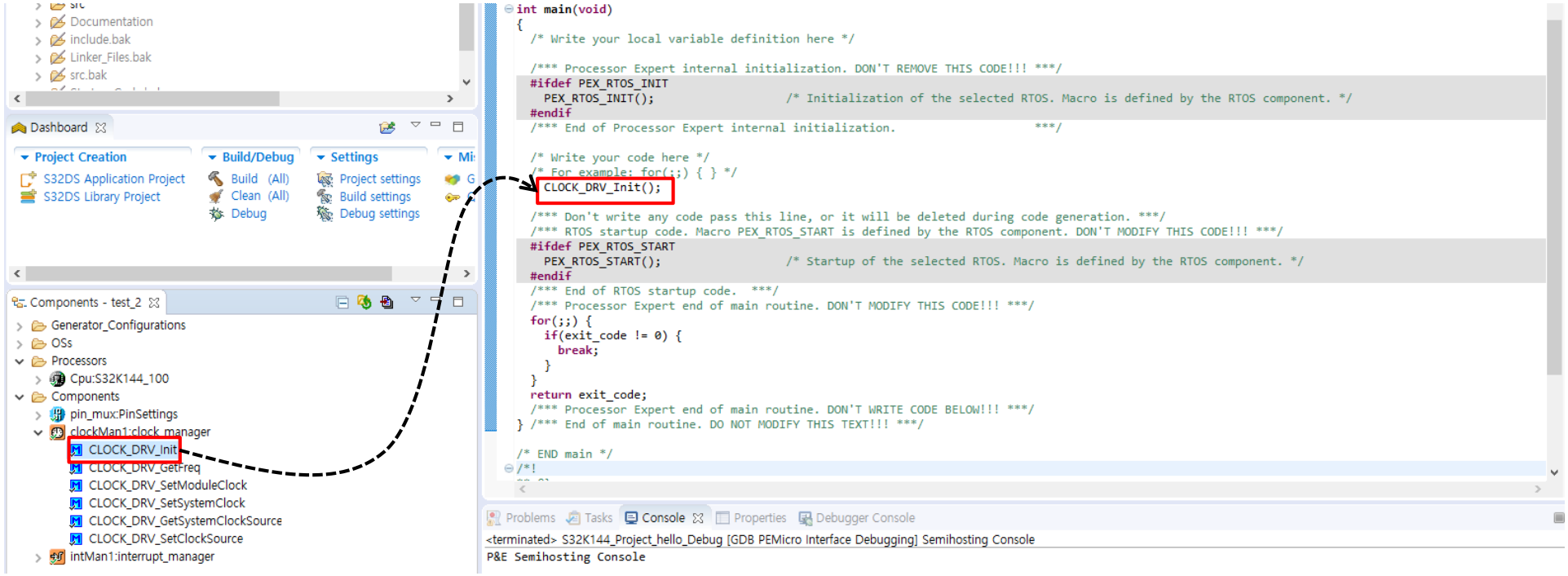
   ✓ **프로젝트의** Sources **하위에** main.c**가 생성됨을 확인한다.** (**생성되지 않았다면 프로젝트를 우클릭하고** Refresh)



Switch Example with API

3. 프로그래밍

1) CLOCK_DRV_Init 함수를 Drag and Drop으로 main.c에 가져온다.

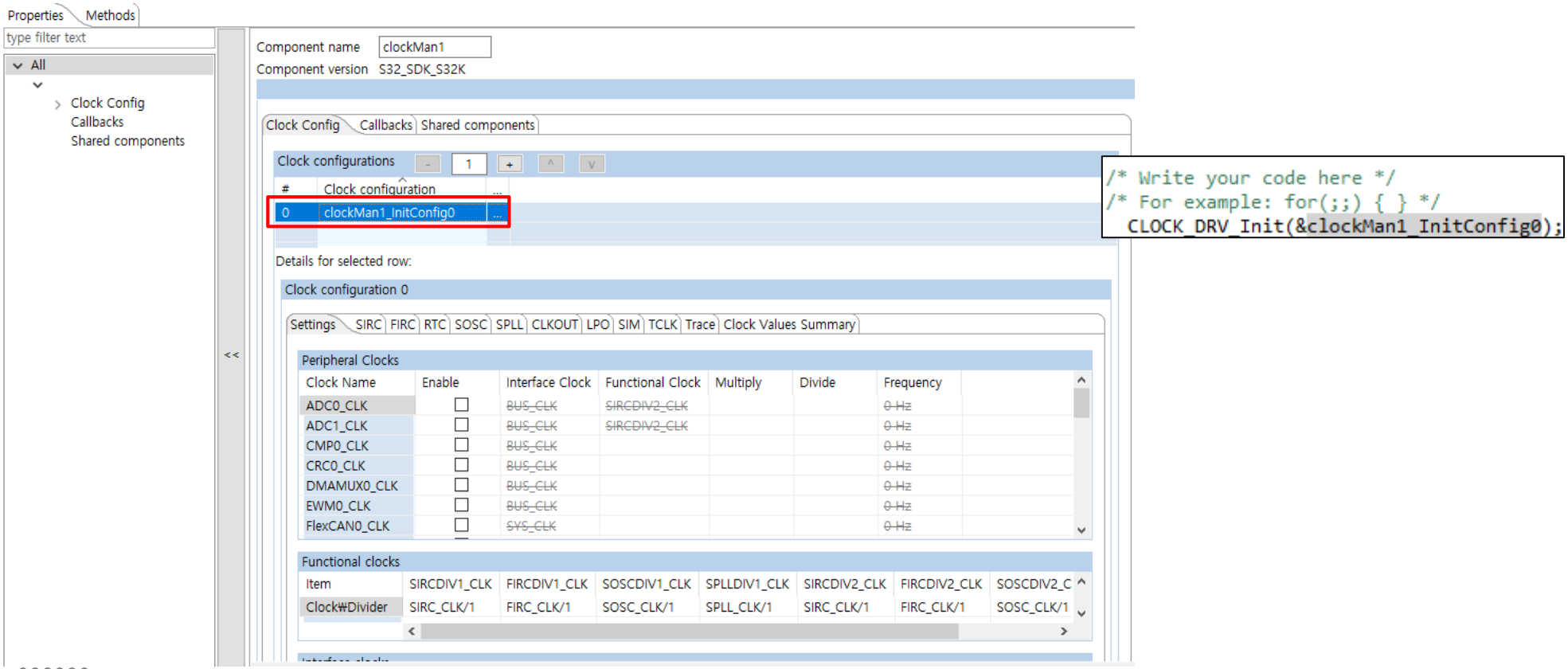


Switch Example with API

3. **프로그래밍**

   2) **해당 함수의 매개변수를 확인하기 위해** clockMan1:clock_manager**를 더블 클릭한다.**

   3) Clock configuration**의 이름을 확인하고 이 이름을** ClOCK_DRV_Init() **함수의 매개변수로 넣어준다.**



Switch Example with API

3. **프로그래밍**

4) **마찬가지로** PINS_DRV_Init() **함수를** Drag and Drop**으로 가져와 세팅한 핀 정보를 프로그램에 추가한다.**

```
/* Write your code here */
/* For example: for(;;) { } */
  CLOCK_DRV_Init(&clockMan1_InitConfig0);

  PINS_DRV_Init(NUM_OF_CONFIGURED_PINS, g_pin_mux_InitConfigArr);
```

Switch Example with API

3. **프로그래밍**

   4) **동일한 방법으로** PINS_DRV_SetPinDirection**을 추가한다.** SW2**와** LED_Blue **두 가지 핀을 사용할 것이기 때문에 두 개 추가한다.**

   5) **해당 함수의 매개변수는** GPIO base ptr, pin number, input(0)/output(1)**이다.**

   6) **비슷한 함수로** PINS_DRV_SetPinsDirection**이 있는데 이는 동일한** GPIO base ptr**의 여러** pin number**를 동시에** set **할 수 있다.**

```
PINS_DRV_SetPinDirection(PTD, 0, 1);  // PTD 0  output
PINS_DRV_SetPinDirection(PTC, 12, 0); // PTC 12 input
```

Switch Example with API

3. **프로그래밍**

   7) PINS_DRV_ReadPins() **함수는 해당** GPIO**의 모든 값을 리턴하므로 이를 이용해 스위치가 눌렸을 때의 조건식을 세운다.**

   8) PINS_DRV_ClearPins()**을 통해** LED_Blue**가 연결된** PTD0**를** clear**하여** LED**를 끈다.**

   9) PINS_DRV_SetPins()**를 통해** PTD0**를** set**하여** LED**를 켠다.**

```
  for(;;) {
  if(PINS_DRV_ReadPins(PTC) & (1<<12U)) // pull down resistor
      PINS_DRV_ClearPins(PTD, 1);
  else
      PINS_DRV_SetPins(PTD, 1);
}
```

Switch Example with API

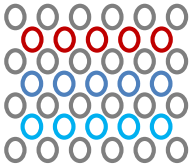
3. **프로그래밍**

   ✓ **전체 main.c는 다음과 같고, 빌드 및 디버그를 통해 동작을 확인한다.**

```
int main(void)
{
  /* Write your local variable definition here */

  /*** Processor Expert internal initialization. DON'T REMOVE THIS
  #ifdef PEX_RTOS_INIT
    PEX_RTOS_INIT();                 /* Initialization of the sel
  #endif
  /*** End of Processor Expert internal initialization.

  /* Write your code here */
  /* For example: for(;;) { } */
    CLOCK_DRV_Init(&clockMan1_InitConfig0);

    PINS_DRV_Init(NUM_OF_CONFIGURED_PINS, g_pin_mux_InitConfigArr);

    PINS_DRV_SetPinDirection(PTD, 0, 1);  // PTD 0  output
    PINS_DRV_SetPinDirection(PTC, 12, 0); // PTC 12 input

  /*** Don't write any code pass this line, or it will be deleted d
  /*** RTOS startup code. Macro PEX_RTOS_START is defined by the RT
  #ifdef PEX_RTOS_START
    PEX_RTOS_START();                /* Startup of the selected R
  #endif
  /*** End of RTOS startup code.  ***/
  /*** Processor Expert end of main routine. DON'T MODIFY THIS CODE

    for(;;) {
    if(PINS_DRV_ReadPins(PTC) & (1<<12U)) // pull down resistor
        PINS_DRV_ClearPins(PTD, 1);
    else
        PINS_DRV_SetPins(PTD, 1);
  }

  return exit_code;
  /*** Processor Expert end of main routine. DON'T WRITE CODE BELOW
} /*** End of main routine. DO NOT MODIFY THIS TEXT!!! ***/
```

Q & A

Thank you for your attention



Architecture and
Compiler
for Embedded Systems Lab.

School of Electronics Engineering, KNU

ACES Lab (hn02301@gmail.com)