

펌웨어 개발 및 회로 설계 기초

-2-

2019-02-01

1. Interrupt

1. Interrupt



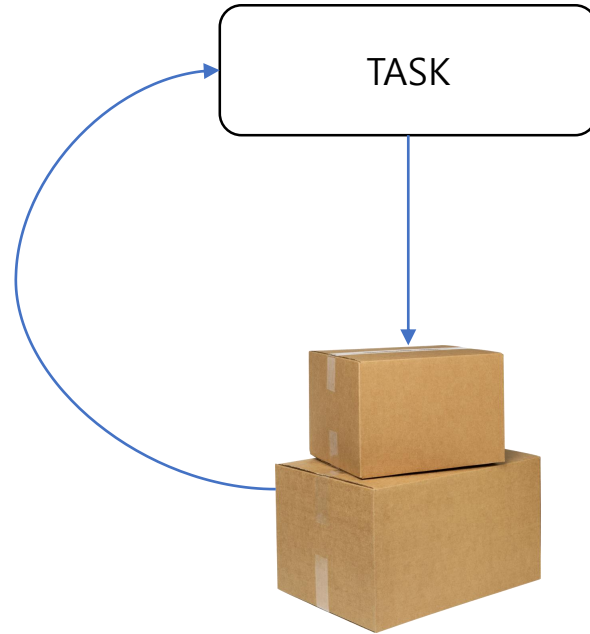
- 나는 지금 너무 바쁘데 지난주에 시킨 택배가 언제 올지 모른다.
- 택배가 왔는지 아닌지 확인하기 위하여 어떻게 해야 할까?

5. Interrupt



- 나는 지금 너무 바쁘데 지난주에 시킨 택배가 언제 올지 모른다.
- 택배가 왔는지 아닌지 확인하기 위하여 어떻게 해야 할까?

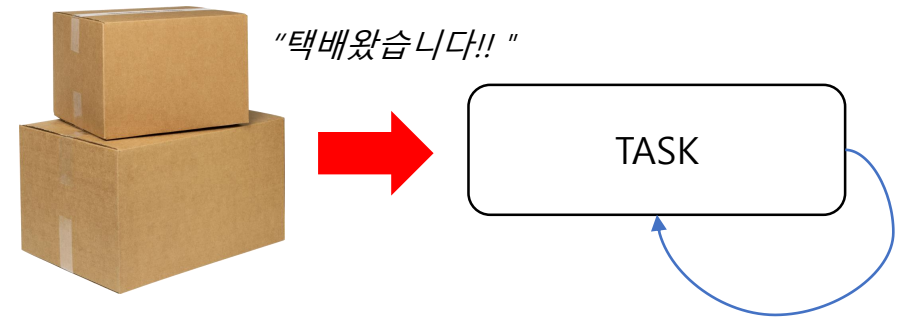
5. Interrupt



[Polling]

MCU에선 누구에게 이 사실을 알릴까? -> NVIC

"아저씨 택배 도착하면 010-XXXX-XXXX로 문자남겨주세요 "

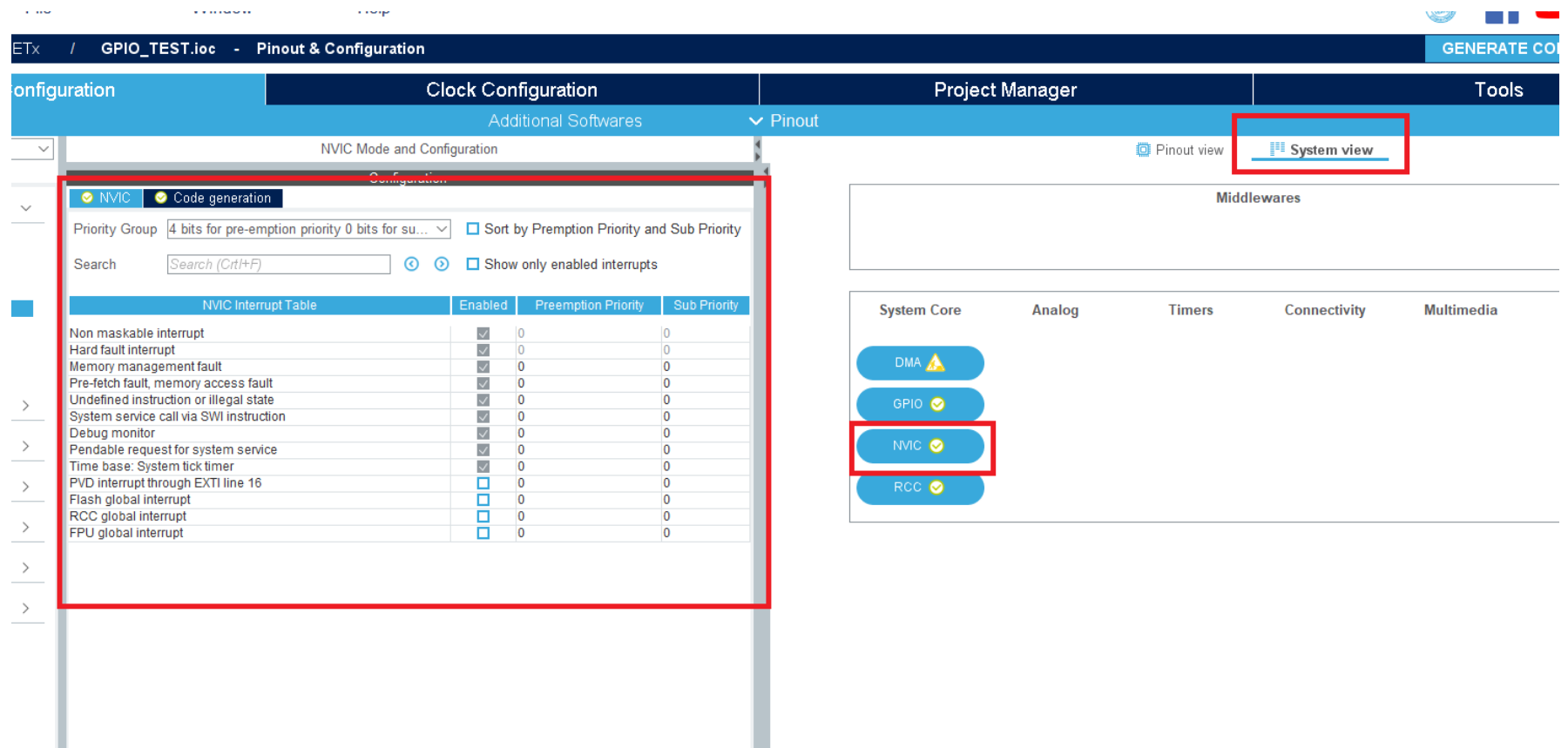


[Interrupt]

1. Interrupt

- NVIC(Nested Vector Interrupt Controller)

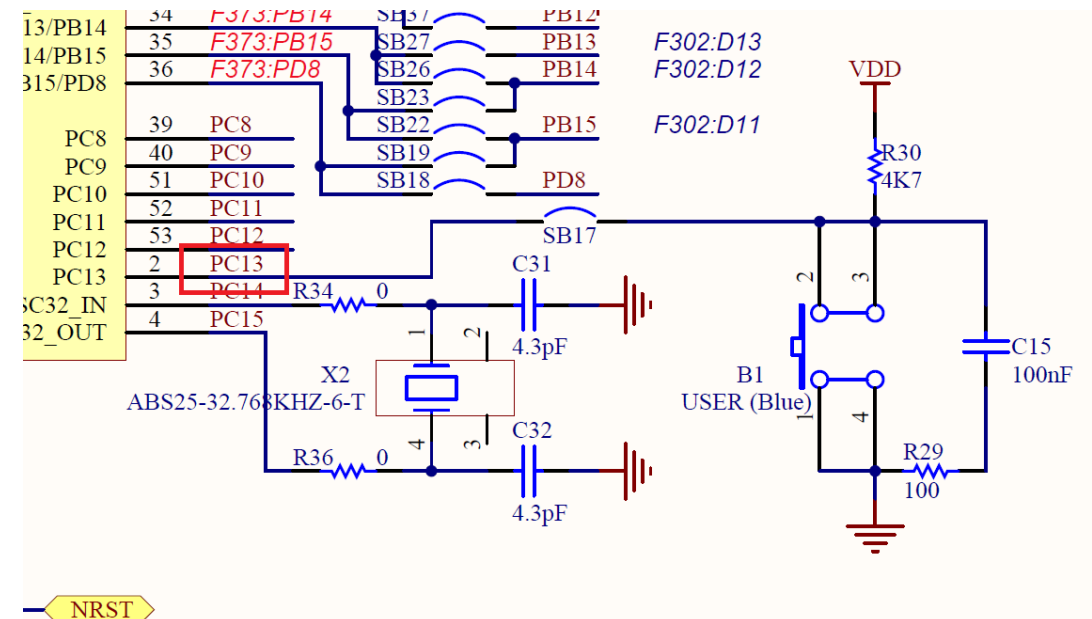
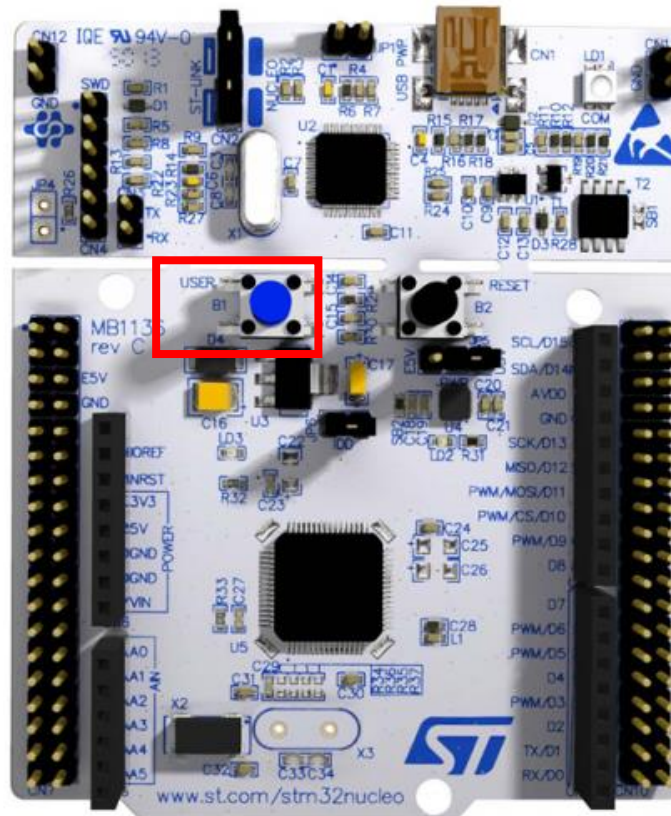
- 여러 종류의 인터럽트를 NVIC에 등록하면 NVIC가 인터럽트를 관리하기 시작한다.
- 인터럽트의 우선순위를 통해 여러 인터럽트의 순서를 정의한다.



1. Interrupt

• Interrupt 실습

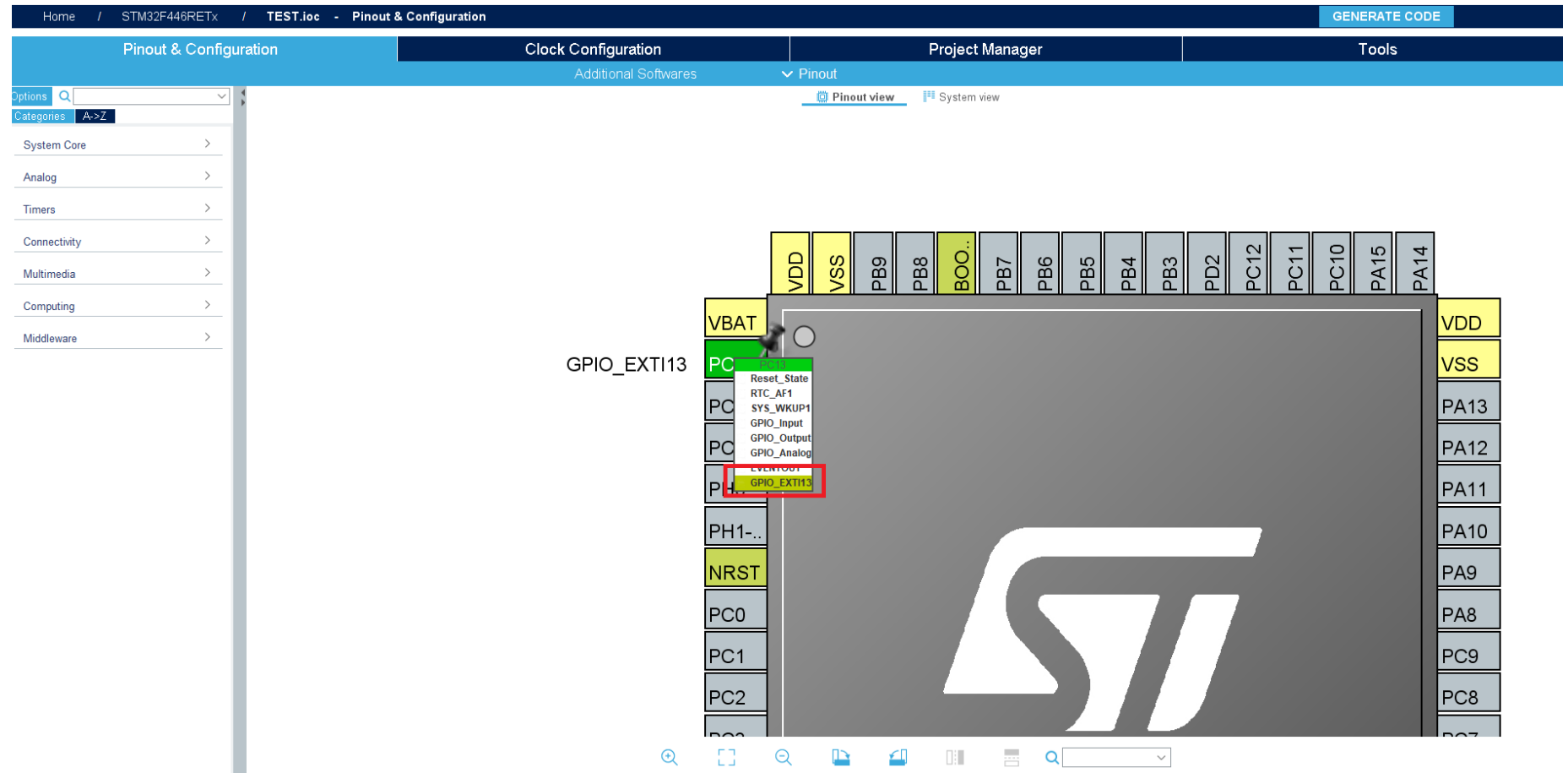
- 누클레오 보드 중앙에 있는 USER 버튼(파란색)은 내부적으로 칩의 PC13 핀에 연결되어 있다.
- 해당 버튼이 눌리는 순간을 인터럽트로 검출하여 버튼이 눌린 횟수를 저장해보자.



1. Interrupt

- Interrupt 실습

- CubeMX에서 PC13을 GPIO_EXTI13으로 설정



1. Interrupt

- Interrupt 실습

- System view -> NVIC 탭 클릭 -> EXTI Line[15:10] Interrupts를 활성화

The screenshot shows the STM32CubeMX IDE interface. The top bar includes 'le', 'Window', and 'Help' menus. Below the top bar, there are tabs for 'Configuration', 'Clock Configuration', 'Project Manager', and 'Tools'. The 'Configuration' tab is active, and the 'NVIC Mode and Configuration' section is expanded. The 'Configuration' sub-section is selected, showing the 'NVIC Interrupt Table'.

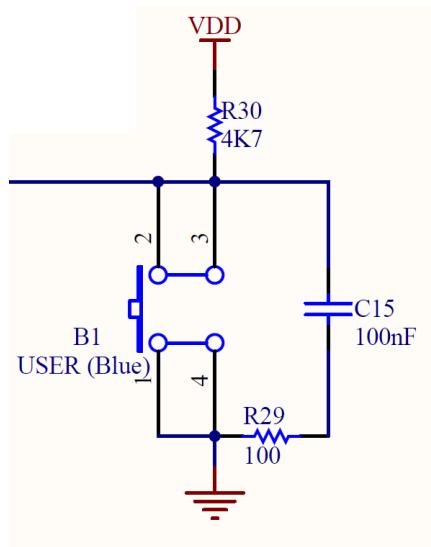
The 'NVIC Interrupt Table' is a table with the following columns: 'NVIC Interrupt Table', 'Enabled', 'Preemption Priority', and 'Sub Priority'. The table lists various interrupts, including 'Non maskable interrupt', 'Hard fault interrupt', 'Memory management fault', 'Pre-fetch fault, memory access fault', 'Undefined instruction or illegal state', 'System service call via SWI instruction', 'Debug monitor', 'Pendable request for system service', 'Time base: System tick timer', 'PVD interrupt through EXTI line 16', 'Flash global interrupt', 'RCC global interrupt', 'EXTI line[15:10] interrupts', and 'CPU global interrupt'. The 'EXTI line[15:10] interrupts' row is highlighted with a red box and labeled '3'.

The 'System view' tab is highlighted with a red box and labeled '1'. The 'NVIC' tab is highlighted with a red box and labeled '2'. The 'System Core' section shows the 'NVIC' component with a checkmark, indicating it is enabled.

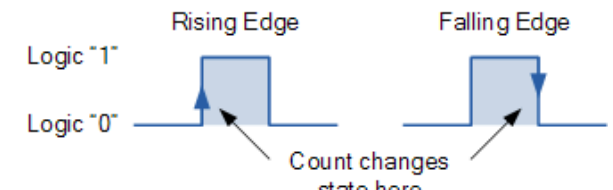
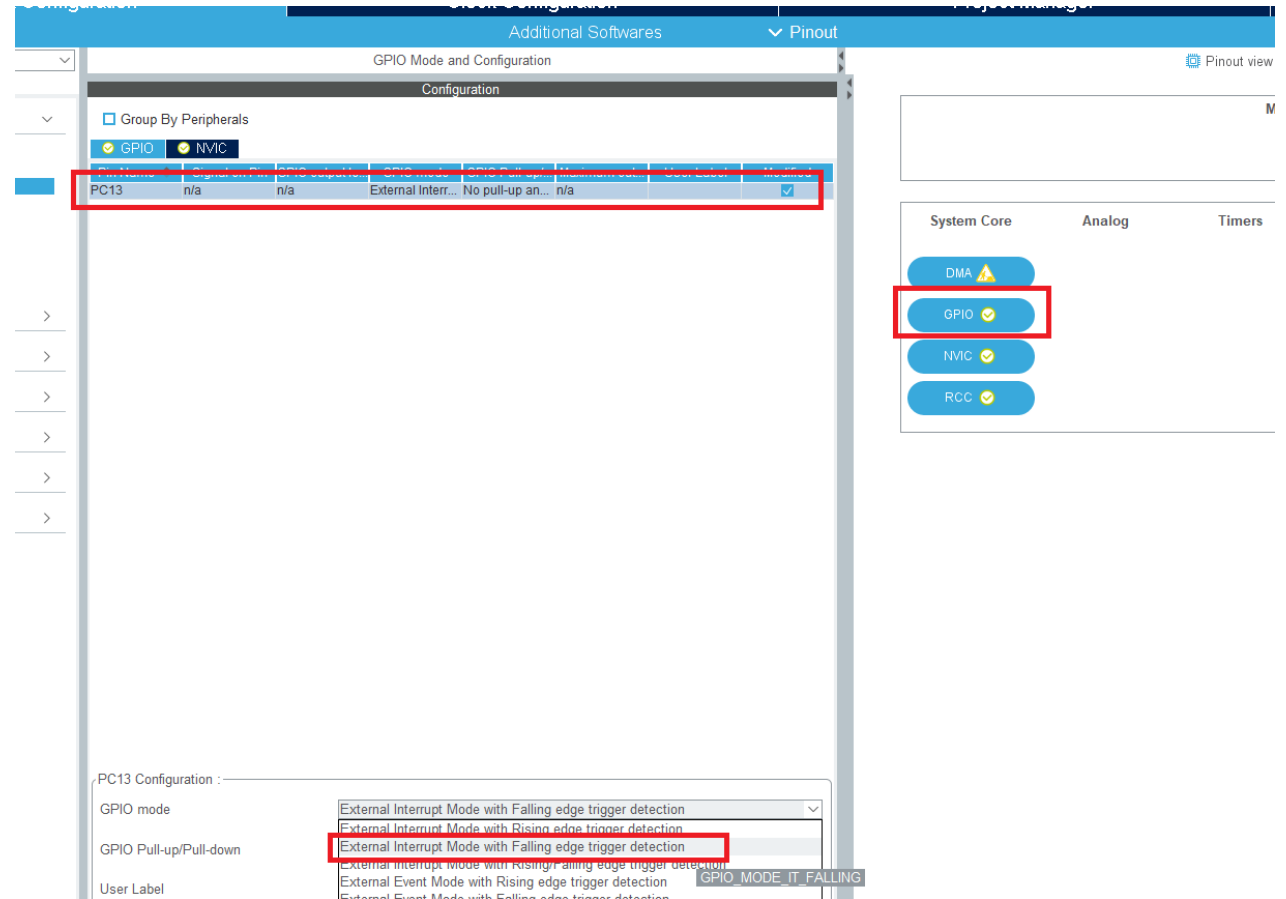
1. Interrupt

- Interrupt 실습

- GPIO 탭에서 PC13핀을 **Falling Edge trigger**로 설정 -> "GENERATE CODE"



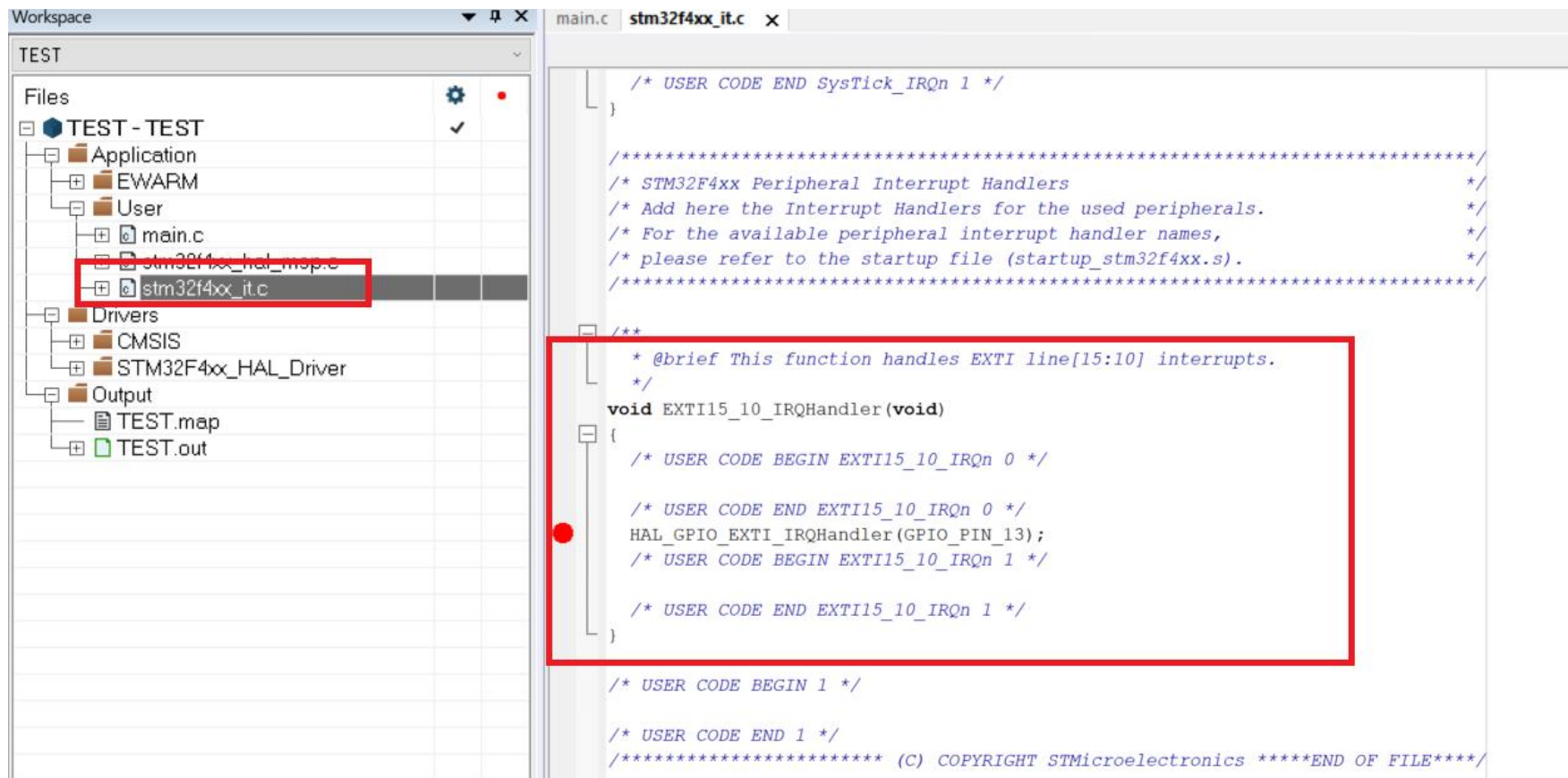
버튼 입력 부분은 풀업(Pull-up)되어있다가 눌리는 순간 Gnd가 된다.
따라서 눌리는 순간을 검출하기 위해서는 **Falling Edge**를 알아내야함.



1. Interrupt

• Interrupt 실습

- STM32의 모든 인터럽트 함수는 **stm32f4xx_it.c**에 존재한다.
- 해당 파일을 열고 우리가 활성화한 인터럽트에 대한 함수를 찾자.(EXTI15_10_IRQHandler)
- 이 함수 내에 종단점을 찍고, 프로그램 업로드 후, USER버튼을 눌렀을 때 해당 구문으로 들어오면 성공



```
Workspace
TEST
Files
TEST - TEST
  Application
  EWARM
  User
    main.c
    stm32f4xx_hal_map.c
    stm32f4xx_it.c
  Drivers
    CMSIS
    STM32F4xx_HAL_Driver
  Output
    TEST.map
    TEST.out

main.c  stm32f4xx_it.c  x

/* USER CODE END SysTick_IRQn 1 */
}

/*****
 * STM32F4xx Peripheral Interrupt Handlers
 * Add here the Interrupt Handlers for the used peripherals.
 * For the available peripheral interrupt handler names,
 * please refer to the startup file (startup_stm32f4xx.s).
 *****/

/**
 * @brief This function handles EXTI line[15:10] interrupts.
 */
void EXTI15_10_IRQHandler(void)
{
    /* USER CODE BEGIN EXTI15_10_IRQn 0 */

    /* USER CODE END EXTI15_10_IRQn 0 */
    HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_13);
    /* USER CODE BEGIN EXTI15_10_IRQn 1 */

    /* USER CODE END EXTI15_10_IRQn 1 */
}

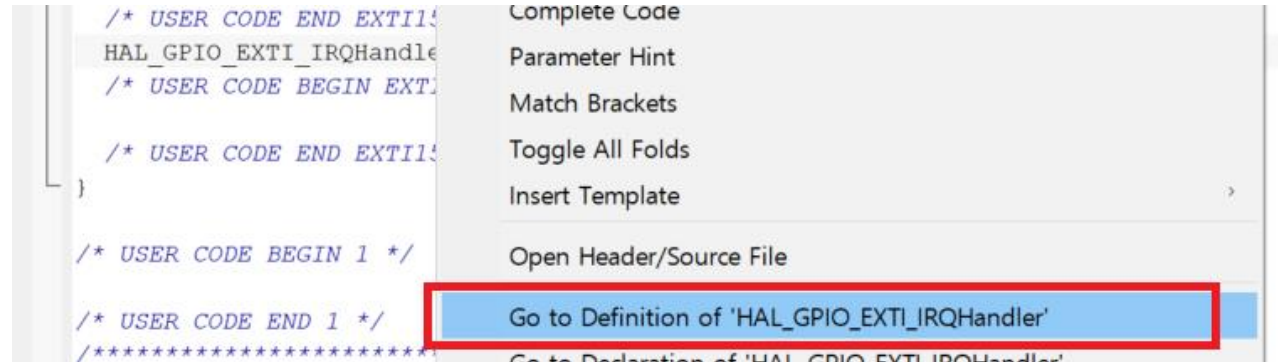
/* USER CODE BEGIN 1 */

/* USER CODE END 1 */
/***** (C) COPYRIGHT STMicroelectronics *****END OF FILE*****/
```

1. Interrupt

- Interrupt 실습

- HAL_GPIO_EXTI_IRQHandler 함수를 오른쪽 클릭하고 **Go to Definition**을 클릭하면, 해당 함수가 정의된 곳으로 갈 수 있다.



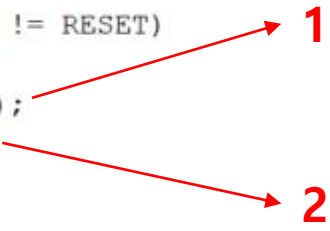
```
void HAL_GPIO_EXTI_IRQHandler(uint16_t GPIO_Pin)
{
    /* EXTI line interrupt detected */
    if( __HAL_GPIO_EXTI_GET_IT(GPIO_Pin) != RESET)
    {
        __HAL_GPIO_EXTI_CLEAR_IT(GPIO_Pin);
        HAL_GPIO_EXTI_Callback(GPIO_Pin);
    }
}
```

Two red arrows point from the code to the numbers 1 and 2. Arrow 1 points to the line `HAL_GPIO_EXTI_CLEAR_IT(GPIO_Pin);` and arrow 2 points to the line `HAL_GPIO_EXTI_Callback(GPIO_Pin);`.

1. Interrupt

• Interrupt 실습

```
void HAL_GPIO_EXTI_IRQHandler(uint16_t GPIO_Pin)
{
    /* EXTI line interrupt detected */
    if(__HAL_GPIO_EXTI_GET_IT(GPIO_Pin) != RESET)
    {
        __HAL_GPIO_EXTI_CLEAR_IT(GPIO_Pin);
        HAL_GPIO_EXTI_Callback(GPIO_Pin);
    }
}
```

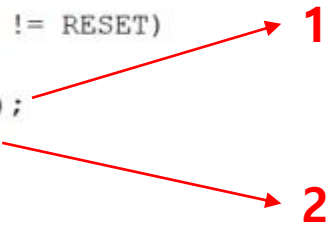


- 이 함수에선 크게 두 가지 일을 한다. [1] 인터럽트 플래그 삭제, [2] 인터럽트 콜백 구문 실행
- [1] 인터럽트가 실행되면, 내부적으로 해당 인터럽트가 발생했다는 플래그를 세운다.
- 만약 동일한 인터럽트가 또 발생되었는데, 그 인터럽트에 대한 플래그가 여전히 활성화 되었으면 그 인터럽트는 무시된다. 따라서 사용자는 인터럽트 구문 내부에서, 해당 인터럽트 플래그를 삭제 해 주어야 다음 인터럽트를 또 받을 수 있다.

1. Interrupt

• Interrupt 실습

```
void HAL_GPIO_EXTI_IRQHandler(uint16_t GPIO_Pin)
{
    /* EXTI line interrupt detected */
    if(__HAL_GPIO_EXTI_GET_IT(GPIO_Pin) != RESET)
    {
        __HAL_GPIO_EXTI_CLEAR_IT(GPIO_Pin);
        HAL_GPIO_EXTI_Callback(GPIO_Pin);
    }
}
```



- 이 함수에선 크게 두 가지 일을 한다. [1] 인터럽트 플래그 삭제, [2] 인터럽트 콜백 구문 실행
- [2] 인터럽트에 대한 콜백 함수가 실행되는데, 해당 함수를 오른쪽 클릭하고 **Go to Definition**을 눌러, 해당 함수가 정의된 곳으로 가보자.

```
__weak void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    /* Prevent unused argument(s) compilation warning */
    UNUSED(GPIO_Pin);
    /* NOTE: This function Should not be modified, when the callback is needed,
       the HAL_GPIO_EXTI_Callback could be implemented in the user file
    */
}
```

1. Interrupt

- Interrupt 실습

```
__weak void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    /* Prevent unused argument(s) compilation warning */
    UNUSED(GPIO_Pin);
    /* NOTE: This function Should not be modified, when the callback is needed,
       the HAL_GPIO_EXTI_Callback could be implemented in the user file
    */
}
```

- Weak 함수는, 만약 함수가 정의되어 있다면 그 함수로 대체되고, 정의되어 있지 않으면 CPU는 '아무 일도 하지 않음(NOP)'으로 대체된다.
- 해당 함수명을 복사하여, main.c 파일의 적당한 곳에 다음과 같이 작성하고, 업로드 하자.

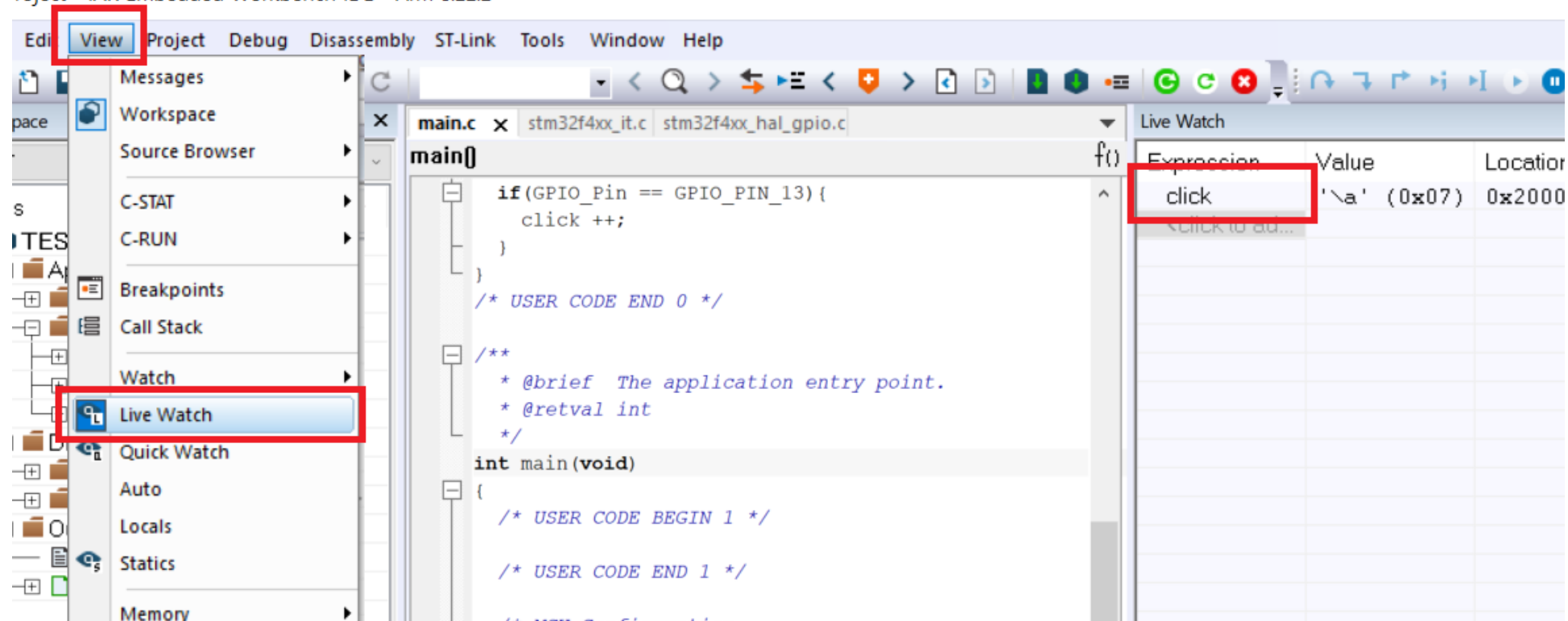
```
/* Private user code -----*/
/* USER CODE BEGIN 0 */
uint8_t click;
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin){
    if(GPIO_Pin == GPIO_PIN_13){
        click++;
    }
}
/* USER CODE END 0 */
```

1. Interrupt

• Interrupt 실습

- 프로그램을 시작시킨 후, **View -> Live Watch**를 클릭하고, 생성된 탭에서 **click** 변수를 등록한다.
- 그 이후에, USER 버튼을 눌러 click 변수가 버튼을 누를 때 마다 1씩 증가하는 것을 확인한다.

Project - IAR Embedded Workbench IDE - Arm 8.22.2

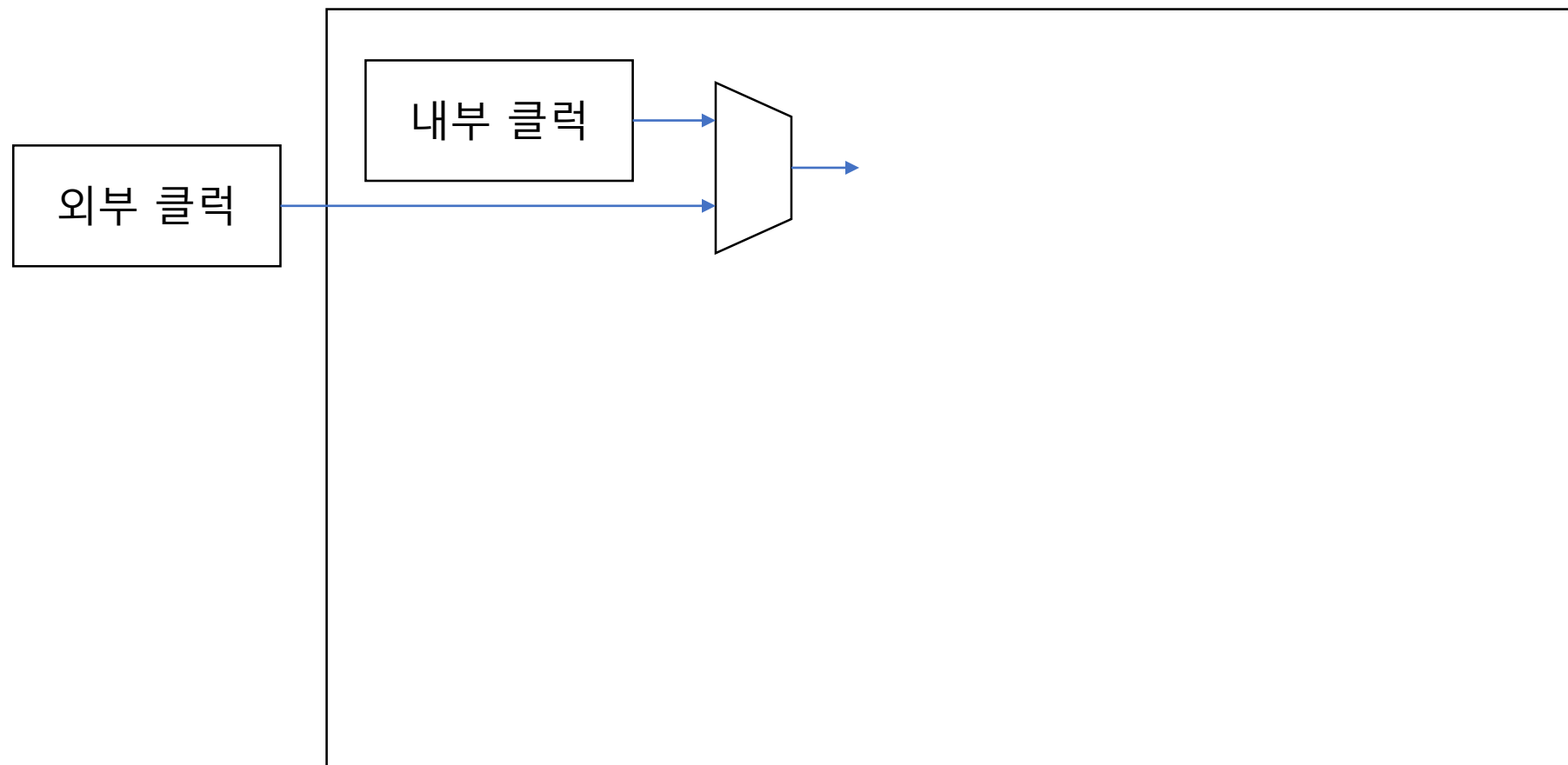


2. Clock

2. Clock

- Clock

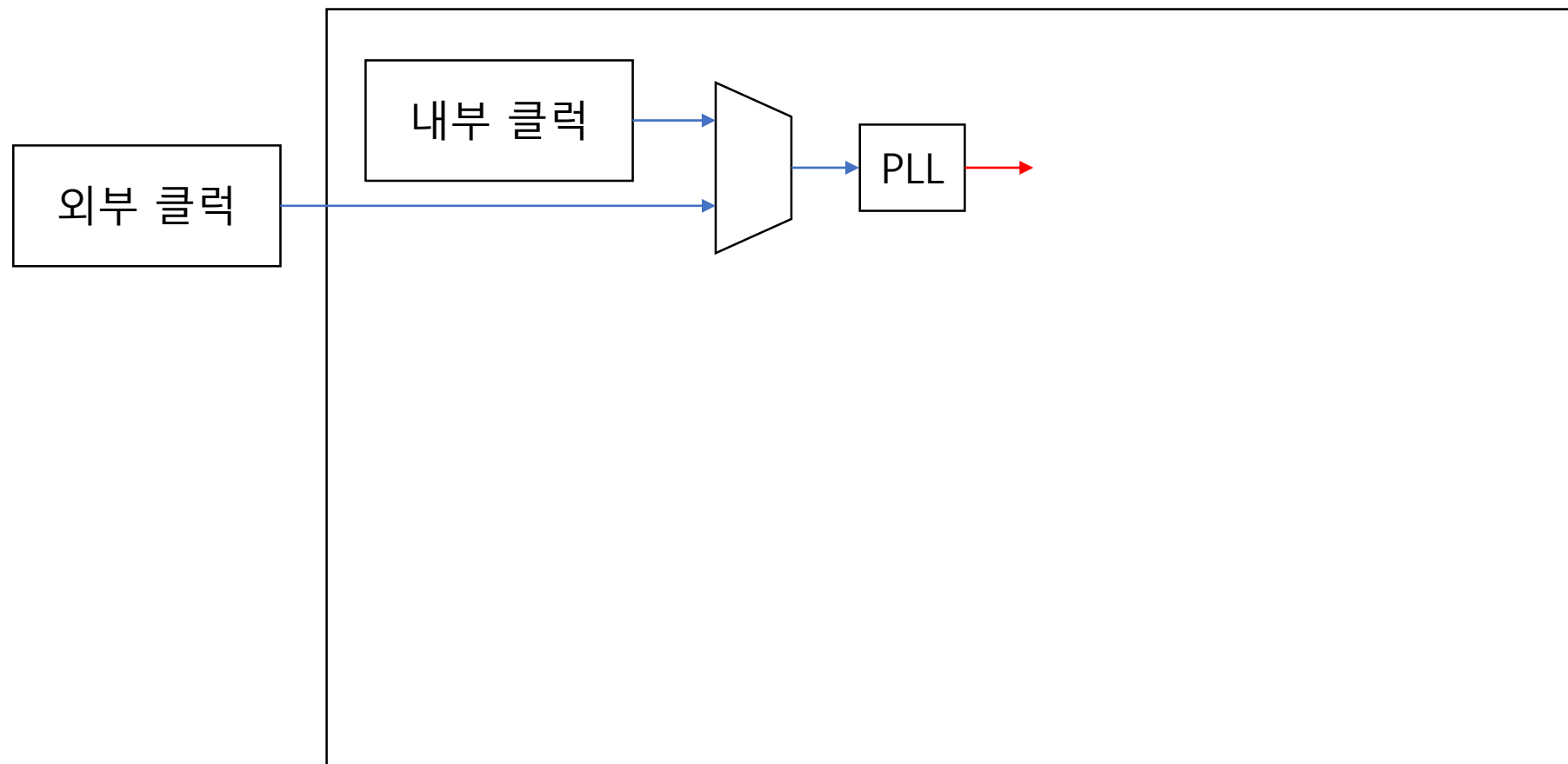
- Clock(클럭)이란 시스템이 돌아가는 속도를 결정하는 아주 중요한 요소이다.
- 클럭 소스는 칩 내부에 있는 **내부 클럭**과, 외부에서 들어오는 **외부 클럭**이 있을 수 있으며, 두 클럭 원 중, MCU가 선택할 수 있다.



2. Clock

- Clock

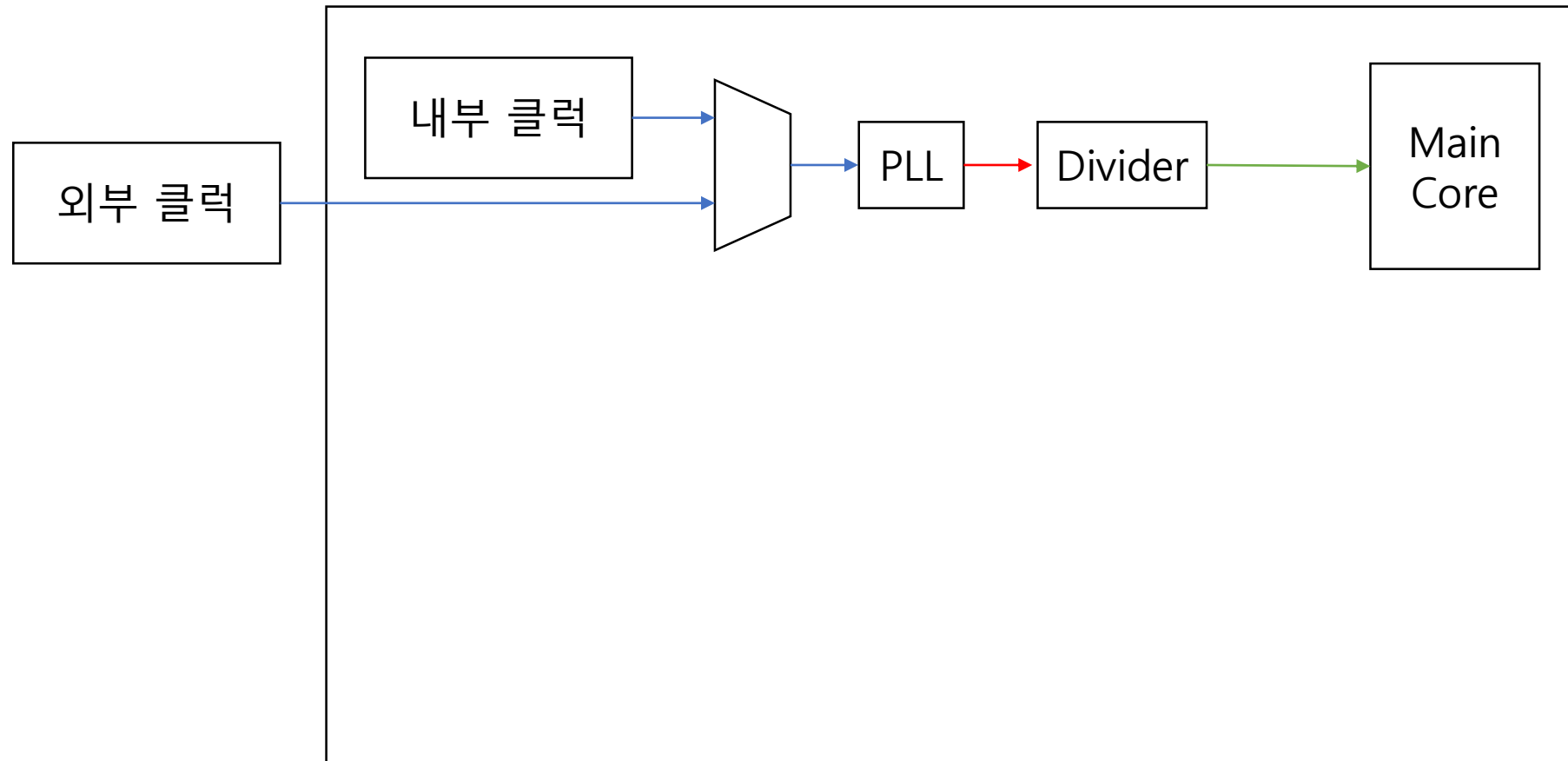
- 선택된 클럭원은 시스템 클럭으로 바로 사용될 수 있지만, 일반적으로 PLL이라는 회로를 통해 훨씬 더 빠른 클럭으로 대체된다.



2. Clock

- Clock

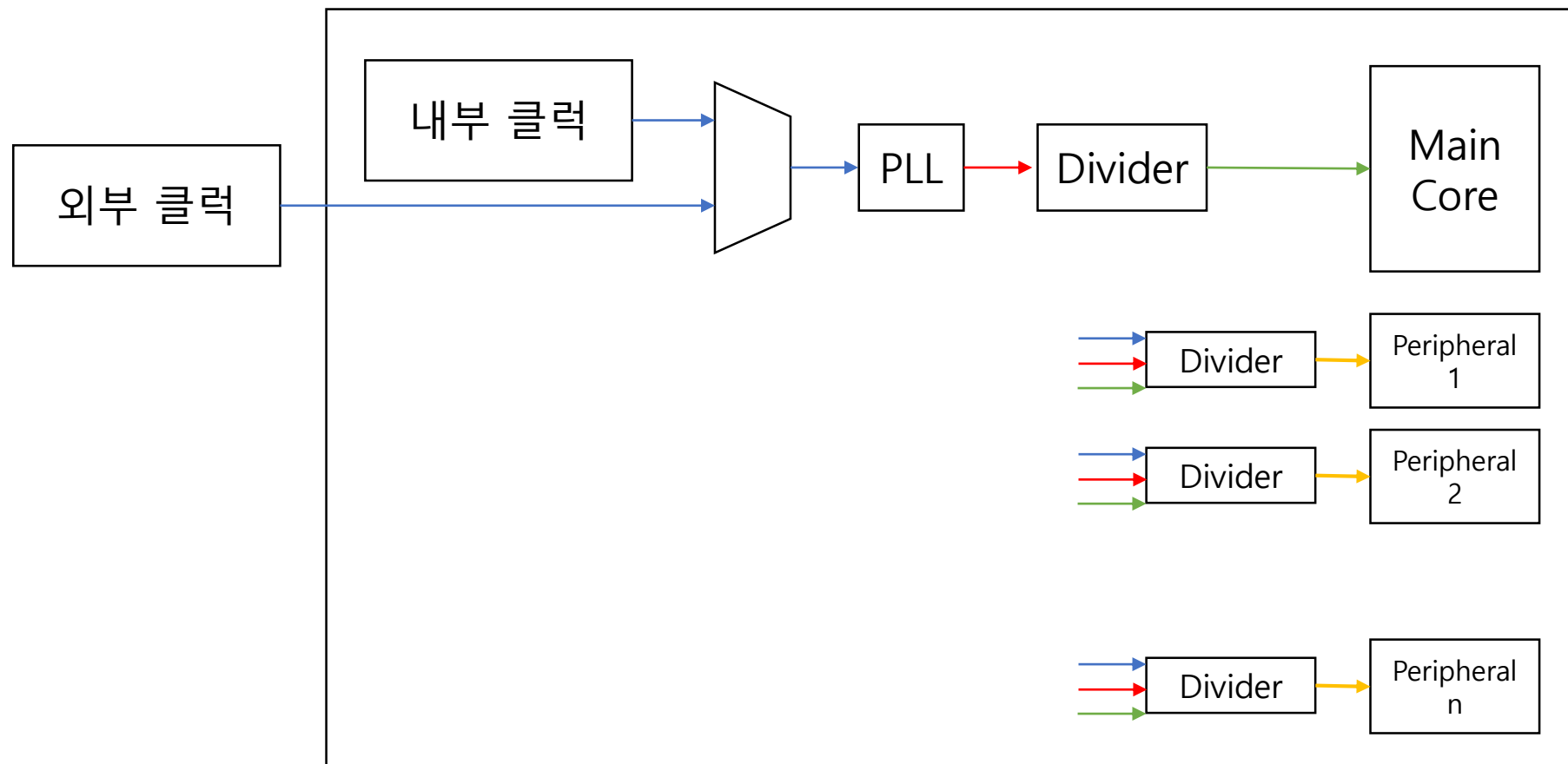
- PLL 회로를 통해 더 빨라진 클럭을 시스템 클럭으로 바로 사용할 수 있으나, 해당 클럭을 다시 정수로 나누어 CPU 코어로 들어가기도 한다.



2. Clock

- Clock

- 메인 코어 뿐만 아니라, 각각의 Peripheral들에게도 클럭 소스가 들어간다.
- 여기에서 중요한 점은 각각의 **Peripheral**들은 메인 코어와 클럭 속도가 다를 수 있다는 것이다.

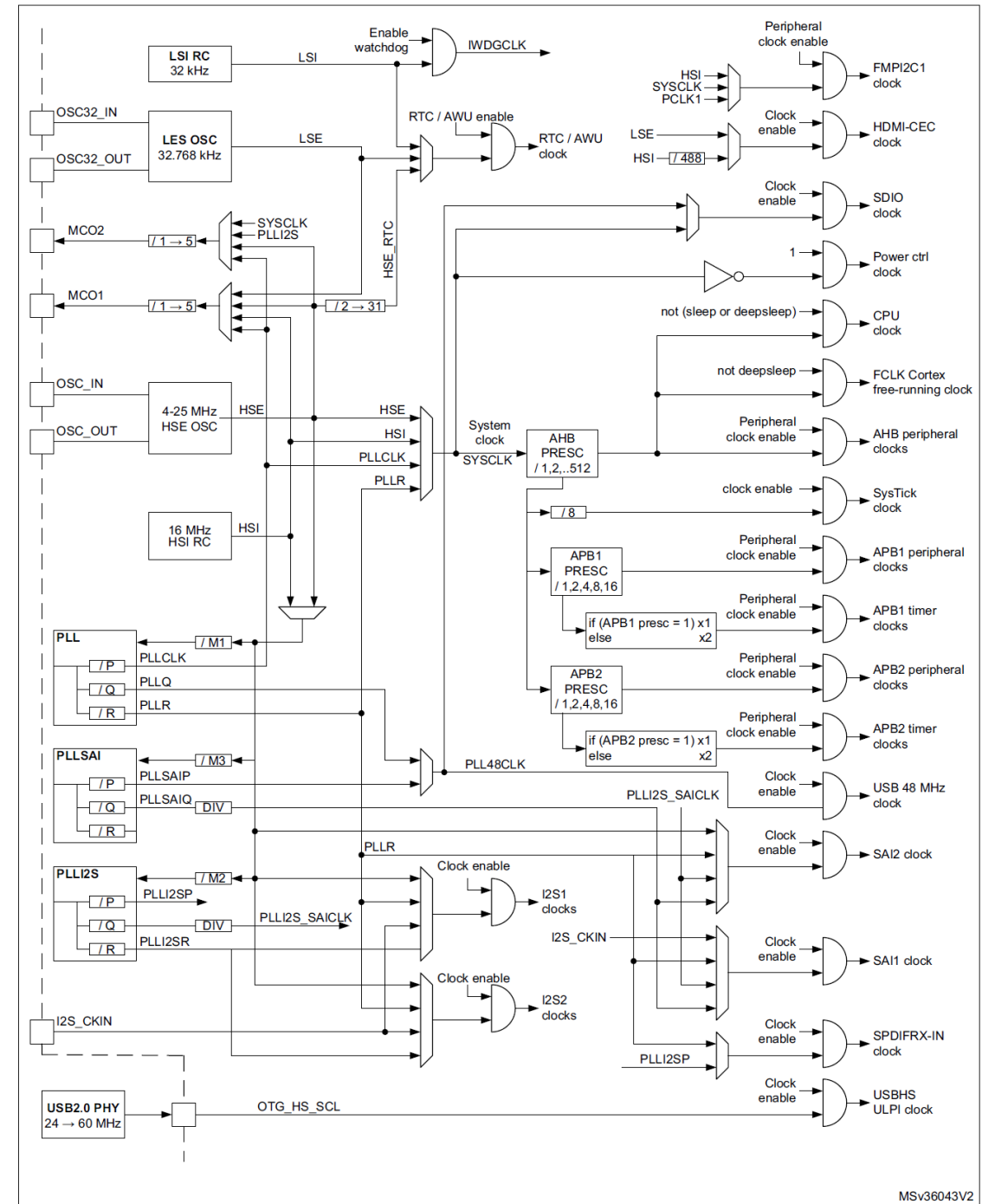


2. Clock

- STM32F4 Clock System



Figure 14. Clock tree



2. Clock

- **C2000 Clock System**

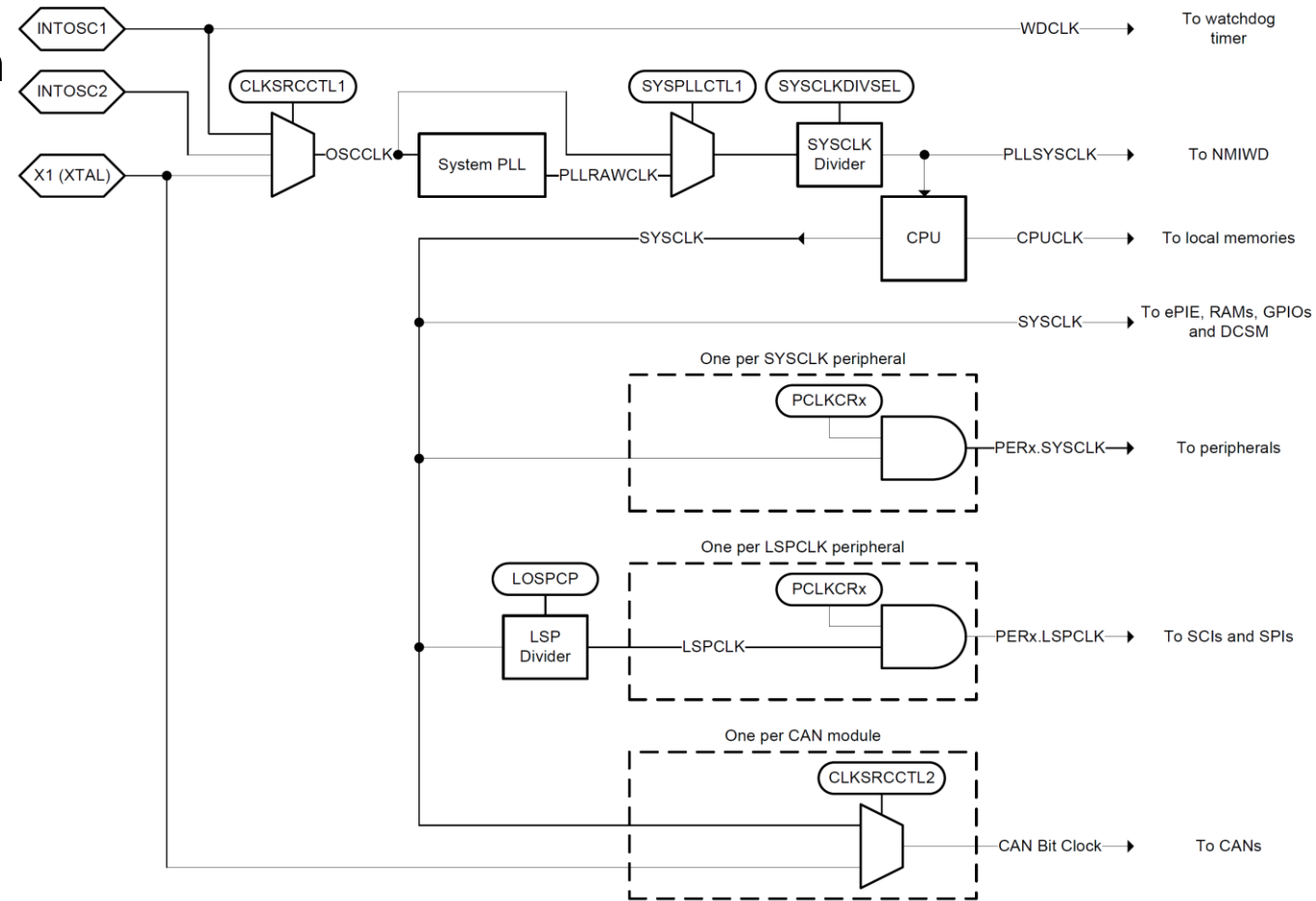
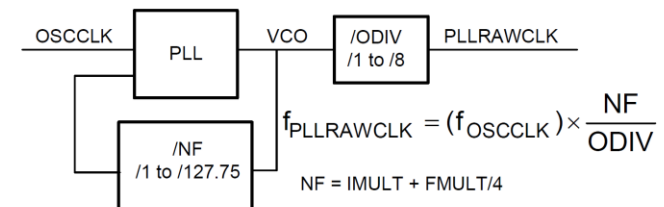


Figure 3-4. System PLL



2. Clock

- CubeMX에서 Clock 세팅하기

2CubeMX TEST.ioc*: STM32F446RETx

