피지컬 컴퓨팅

Lec. 6. Timer

Heenam Yoon

Department of
Human-Centered Artificial Intelligence

E-mail) h-yoon@smu.ac.kr
Room) 0112

Review: 과제

```
void EXTI13_Configuration(void)
{
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_SYSCFG, ENABLE);
  SYSCFG_EXTILineConfig(EXTI_PortSourceGPIOC, EXTI_PinSource13);

  EXTI_InitTypeDef EXTI_InitStructure;
  NVIC_InitTypeDef NVIC_InitStructure;

  EXTI_InitStructure.EXTI_Line = EXTI_Line13;
  EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
  EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;
  EXTI_InitStructure.EXTI_LineCmd = ENABLE;
  EXTI_Init(&EXTI_InitStructure);

  NVIC_PriorityGroupConfig(NVIC_PriorityGroup_1);
  NVIC_InitStructure.NVIC_IRQChannel = EXTI15_10_IRQn;
  NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
  NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
  NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Init(&NVIC_InitStructure);
}
```

EXTI_Trigger_Rising
EXTI_Trigger_Falling
EXTI_Trigger_Rising_Falling
으로 바꾸어서 코딩해보고,
결과가 어떻게 달라지는지도
확인해보기

Review: Interrupt

```
void EXTI15_10_IRQHandler(void)
{
   /* Make sure that interrupt flag is set */
                                                    0
   if (EXTI_GetITStatus(EXTI_Line13) != RESET)
   {
      cnt ++;
      if(cnt % 2 == 1)
      {
          GPIO_SetBits(GPIOA, GPIO_Pin_5);
      }
      else
      {
          GPIO_ResetBits(GPIOA, GPIO_Pin_5);
      }
      /* Clear interrupt flag */
    EXTI_ClearITPendingBit(EXTI_Line13);
   }
}
```

```
int main()
{
   LED_init();
   SW_init();
   EXTI13_Configuration();

   while(1)
   {

   }
  return 0;
}
```

참고

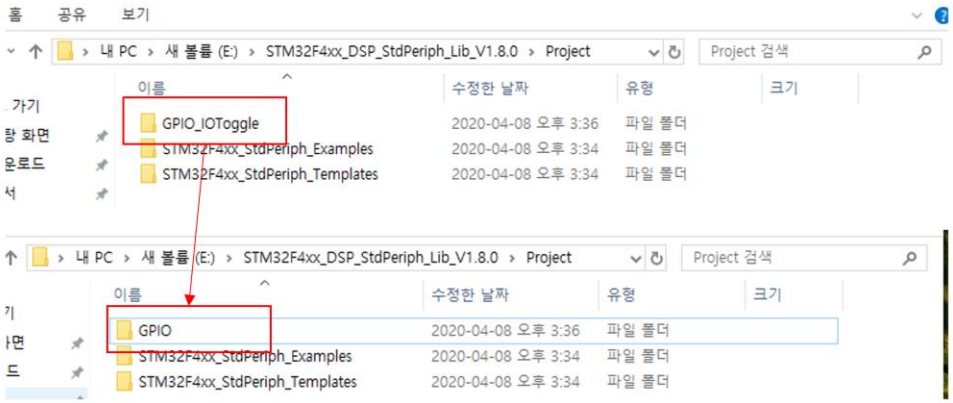
Firmware 개발 기본 흐름

- 필요 기능
- 어떤 포트/핀을 어떻게 사용할지 생각 (I/O, alternate)
- 프로젝트 생성 (환경 구축)
- 목적에 따라 레지스터 설정
- SW 구현

참고

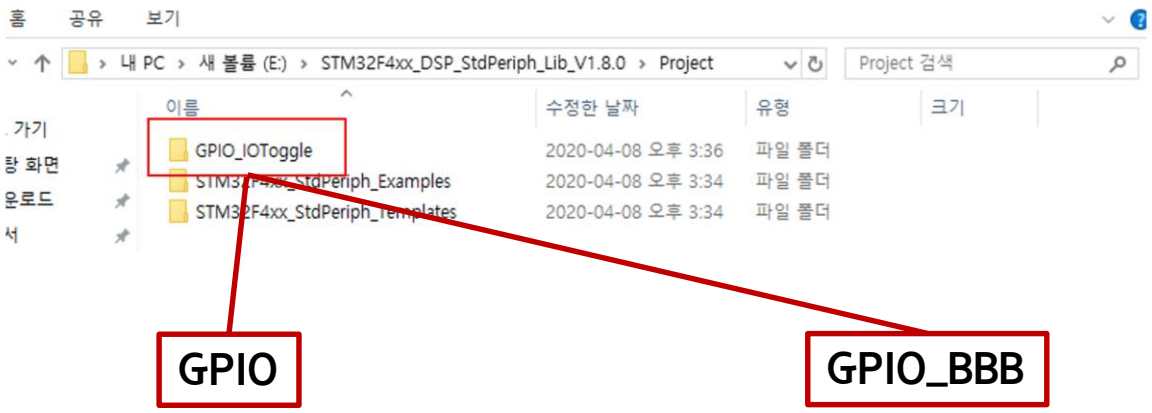
환경구축에 관하여

- 여기서부터 시작했음
  - SPL 압축해제 후 다음 폴더 복사
    - STM32F4xx_DSP_StdPeriph_Lib_V1.8.0₩Project₩STM32F4xx_StdPeriph_Examples₩GPIO₩GPIO_IOToggle
  - 아래 폴더에 붙여넣기 후 폴더명 변경
    - STM32F4xx_DSP_StdPeriph_Lib_V1.8.0₩Project



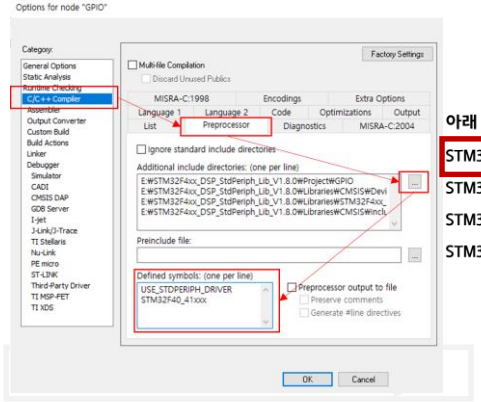
참고

환경구축에 관하여



Add files

경로	파일명
STM32F4xx_DSP_StdPeriph_Lib_V1.8.0\Project\GPIO	system_stm32f4xx.c
STM32F4xx_DSP_StdPeriph_Lib_V1.8.0\Project\GPIO	stm32f4xx_it.c
STM32F4xx_DSP_StdPeriph_Lib_V1.8.0\Libraries\CMSIS\Device\ST\STM32F4xx\Source\Templates\iar	startup_stm32f40_41xxx.s
STM32F4xx_DSP_StdPeriph_Lib_V1.8.0\Libraries\STM32F4xx_StdPeriph_Driver\src	stm32f4xx_gpio.c
STM32F4xx_DSP_StdPeriph_Lib_V1.8.0\Libraries\STM32F4xx_StdPeriph_Driver\src	stm32f4xx_rcc.c



아래 4개 경로 추가

STM32F4xx_DSP_StdPeriph_Lib_V1.8.0₩Project₩GPIO

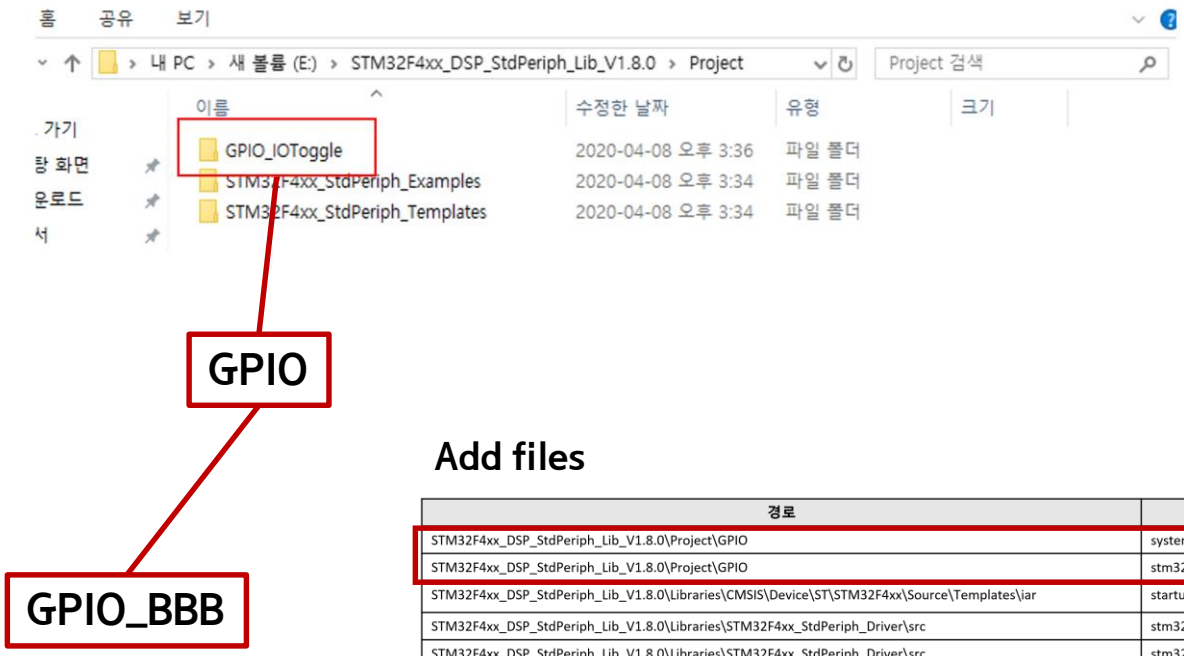
STM32F4xx_DSP_StdPeriph_Lib_V1.8.0₩Libraries₩CMSIS₩Device₩ST₩STM32F4xx₩Include

STM32F4xx_DSP_StdPeriph_Lib_V1.8.0₩Libraries₩STM32F4xx_StdPeriph_Driver₩inc

STM32F4xx_DSP_StdPeriph_Lib_V1.8.0₩Libraries₩CMSIS₩Include

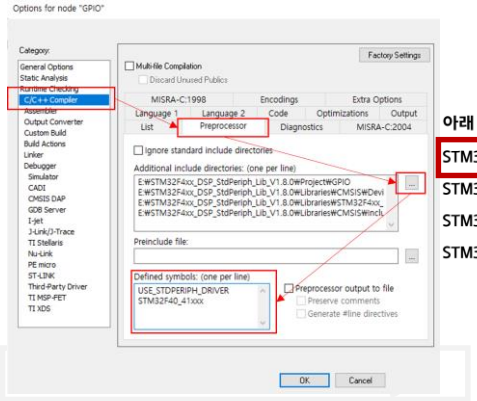
참고

환경구축에 관하여



Add files

경로	파일명
STM32F4xx_DSP_StdPeriph_Lib_V1.8.0\Project\GPIO	system_stm32f4xx.c
STM32F4xx_DSP_StdPeriph_Lib_V1.8.0\Project\GPIO	stm32f4xx_it.c
STM32F4xx_DSP_StdPeriph_Lib_V1.8.0\Libraries\CMSIS\Device\ST\STM32F4xx\Source\Templates\iar	startup_stm32f40_41xxx.s
STM32F4xx_DSP_StdPeriph_Lib_V1.8.0\Libraries\STM32F4xx_StdPeriph_Driver\src	stm32f4xx_gpio.c
STM32F4xx_DSP_StdPeriph_Lib_V1.8.0\Libraries\STM32F4xx_StdPeriph_Driver\src	stm32f4xx_rcc.c



아래 4개 경로 추가

STM32F4xx_DSP_StdPeriph_Lib_V1.8.0₩Project₩GPIO

STM32F4xx_DSP_StdPeriph_Lib_V1.8.0₩Libraries₩CMSIS₩Device₩ST₩STM32F4xx₩Include

STM32F4xx_DSP_StdPeriph_Lib_V1.8.0₩Libraries₩STM32F4xx_StdPeriph_Driver₩inc

STM32F4xx_DSP_StdPeriph_Lib_V1.8.0₩Libraries₩CMSIS₩Include

Timer

- Delay 함수의 입력 값은 무엇을 의미하는가?
- 정확한 시간 delay가 가능할까?
- MCU의 클럭 주파수가 다르다면?

```
void Delay(__IO uint32_t nCount)
{
  for(; nCount != 0; nCount--);
}


Delay(500000);
```

Timer

- Timer의 정의
  - A timer is a counter
  - MPU에서 클럭 펄스를 카운트하는 장치
    - Timer : 시간을(시스템 클럭) 카운트
    - Counter : 외부 혹은 내부의 클럭을 카운트

- Timer의 역할
  - 사용자 설정에 의해 타이머 동작을 하거나 카운터 동작을 하게 됨
  - 타이머 동작과 카운터 동작을 동시에 수행할 수 없음
  - 사용자는 타이머/카운터 장치를 타이머 동작을 하거나 카운터 동작을 하도록 설정해야 함

Timer

STM32 Timers

- Advanced control timers (TIM1 and TIM 8)
- General-purpose timers (TIM2 to TIM5, TIM9 to TIM14)
- Basic timers (TIM6 and TIM7)

STM32F411RE Timers

Timer type	Timer	Counter resolution	Counter type	Prescaler factor	DMA request generation	Capture/ compare channels	Complemen-tary output	Max. interface clock (MHz)	Max. timer clock (MHz)
Advanced -control	TIM1	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	Yes	100	100
General purpose	TIM2, TIM5	32-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	50	100
	TIM3, TIM4	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	50	100
	TIM9	16-bit	Up	Any integer between 1 and 65536	No	2	No	100	100
	TIM10, TIM11	16-bit	Up	Any integer between 1 and 65536	No	1	No	100	100

Timer

STM32 Timers

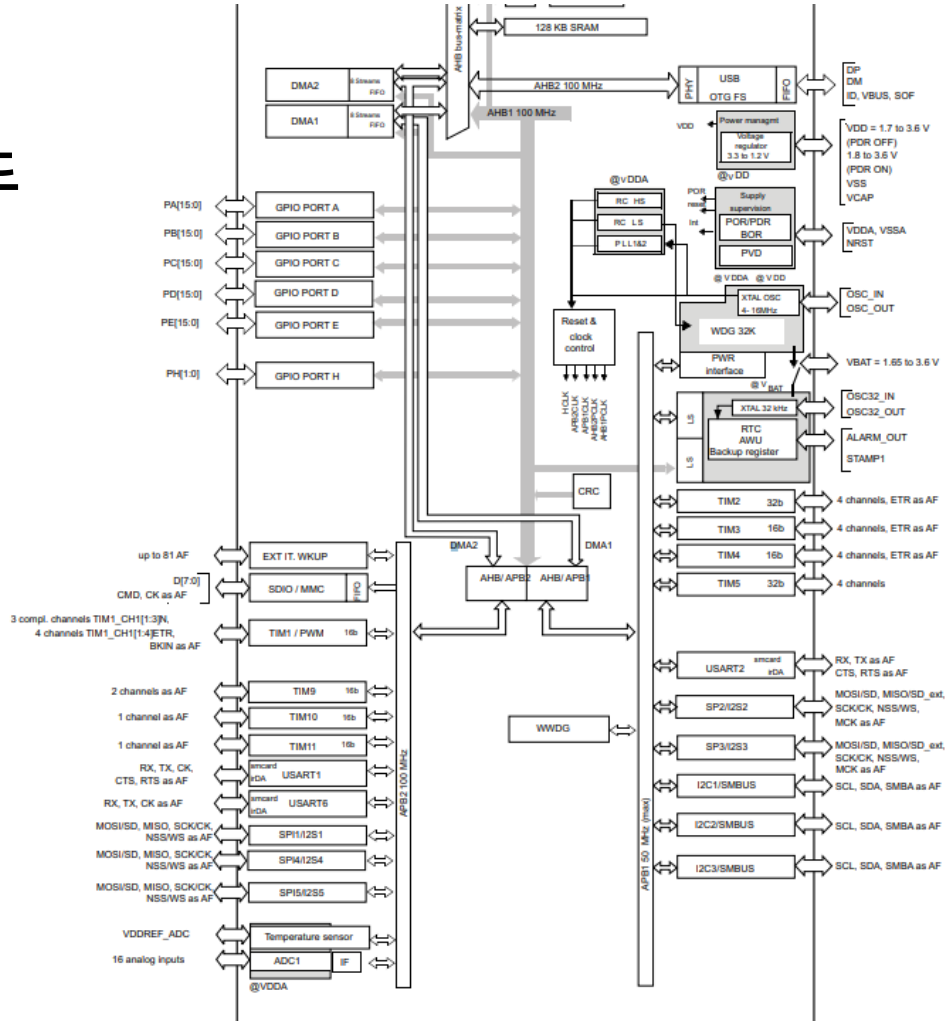
- Q. 초당 100,000,000개의 클럭 펄스를 모두 사용해야 하는가?
- A. No! 초당 원하는 개수만큼 클럭 펄스를 만들 수 있음

Timer type	Timer	Counter resolution	Counter type	Prescaler factor	DMA request generation	Capture/ compare channels	Complemen-tary output	Max. interface clock (MHz)	Max. timer clock (MHz)
Advanced -control	TIM1	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	Yes	100	100
General purpose	TIM2, TIM5	32-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	50	100
	TIM3, TIM4	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	50	100
	TIM9	16-bit	Up	Any integer between 1 and 65536	No	2	No	100	100
	TIM10, TIM11	16-bit	Up	Any integer between 1 and 65536	No	1	No	100	100

Timer

Timer 모드

- Counter 모드
- External Input Counter 모드
- PWM Output 모드
- Input Capture 모드
- Output Compare 모드

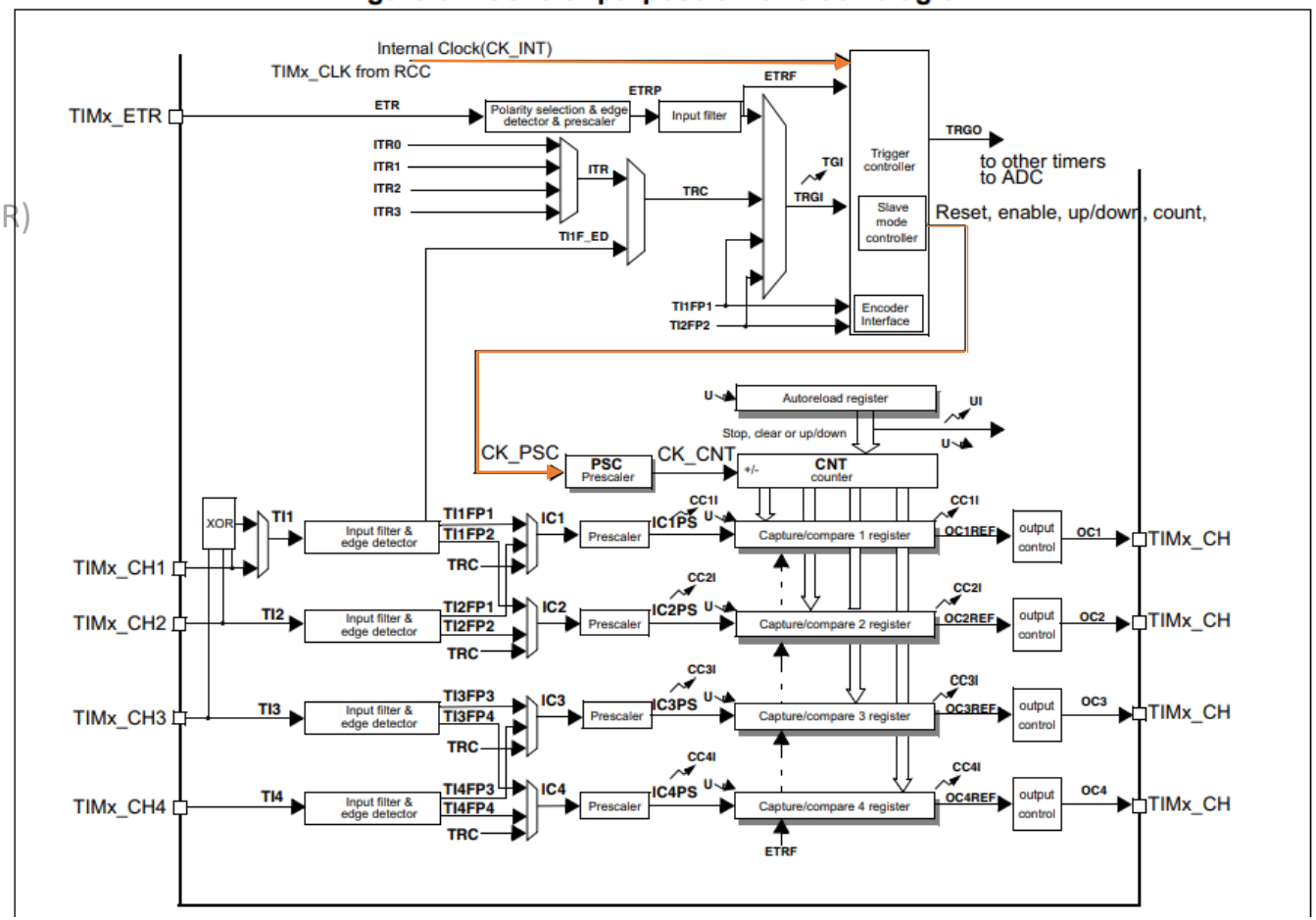


Timer

General-purpose timer

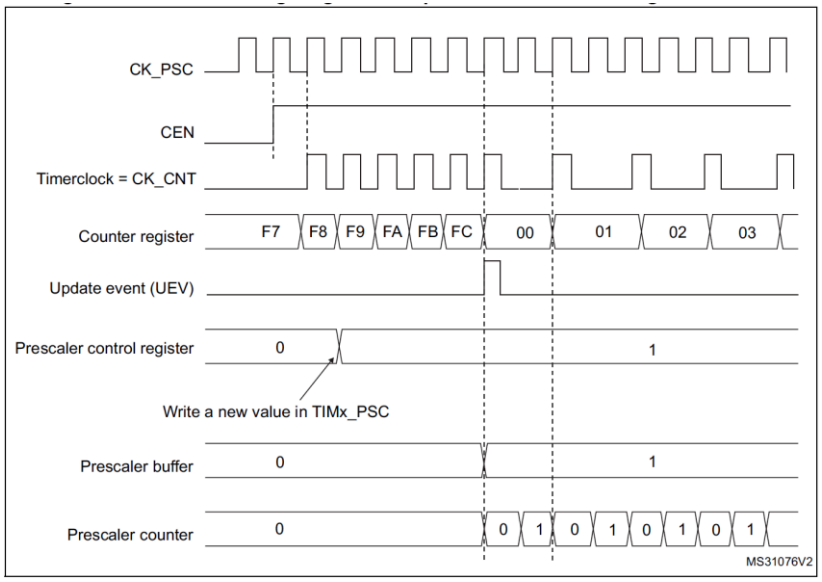
Counter Register (TIMx_CNT)
Prescaler Register (TIMx_PSC)
Auto-Reload Register (TIMx_ARR)

Figure 87. General-purpose timer block diagram

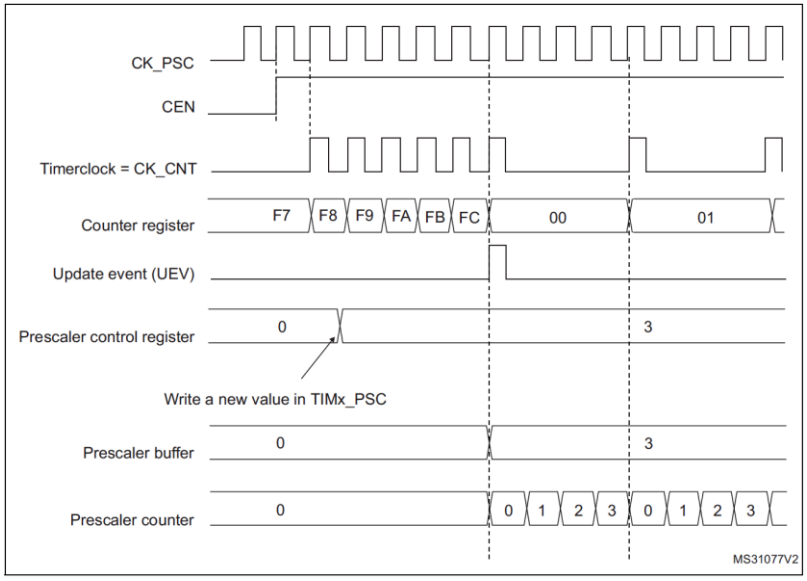


General-purpose timer

- Prescaler(프리스케일러)
  - 프리스케일러는 (TIMx_PSC 레지스터) 16 비트 레지스터를 통해 제어되는 16 비트 카운터에 기초하여 1 과 65536 (2^{16}) 사이의 인수만큼 카운터 클럭 주파수를 분할 할 수 있음



Counter timing diagram with prescaler division change from 1 to 2

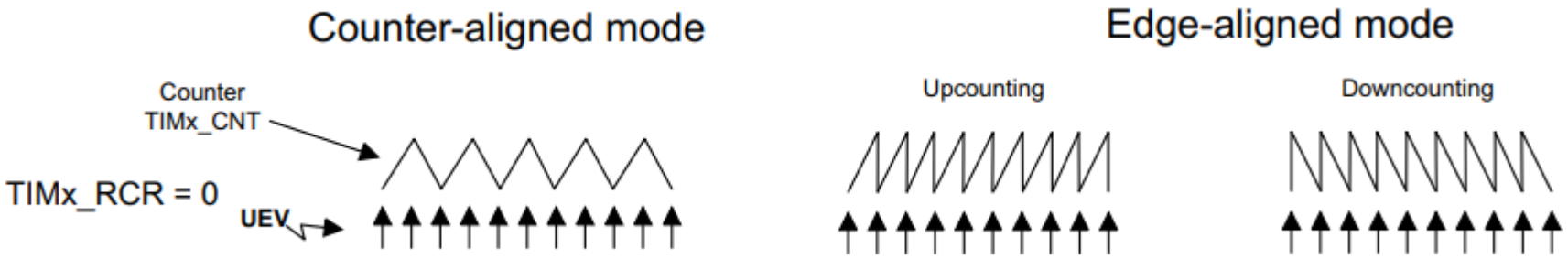


Counter timing diagram with prescaler division change from 1 to 4

Timer

General-purpose timer

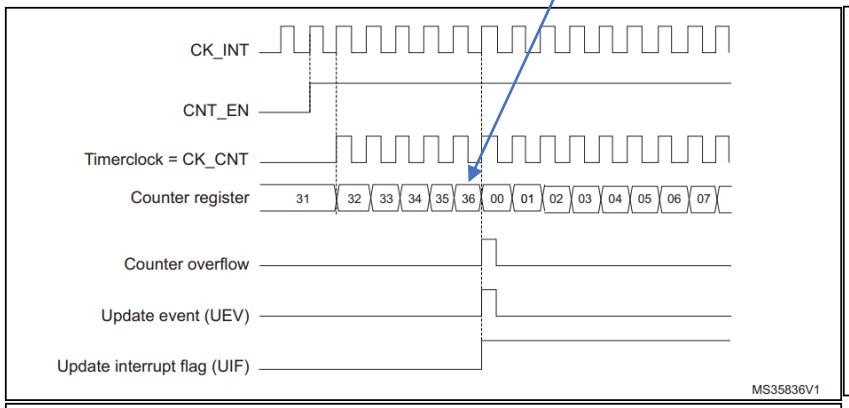
- Counter 모드
  - Up counting 모드
  - Down counting 모드
  - Center-aligned 모드 (up/down counting)

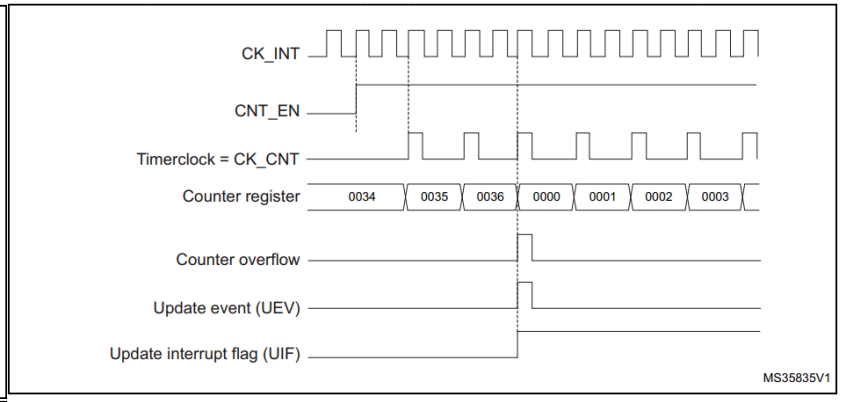


Timer

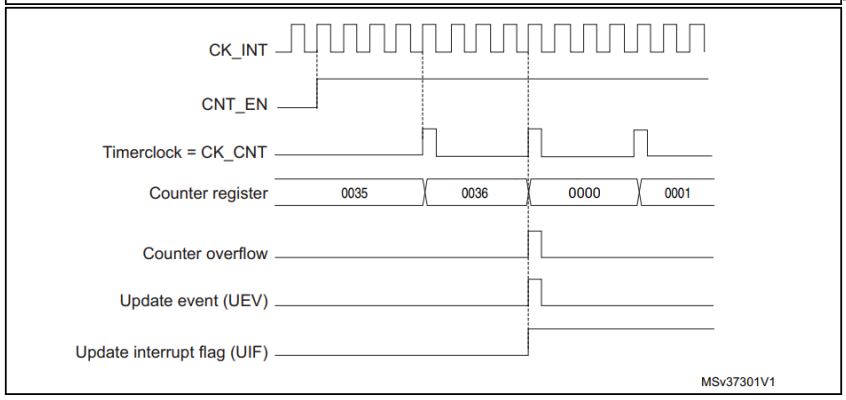
General-purpose timer

- Up counting 모드 (TIMx_ARR=0x36)





prescaler count

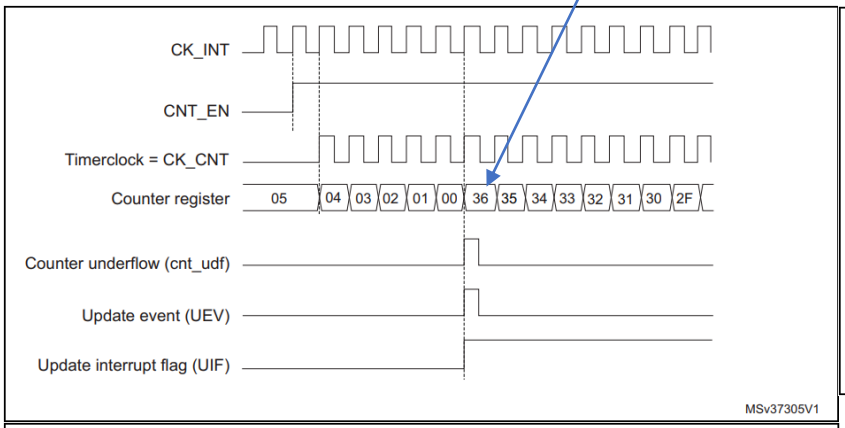


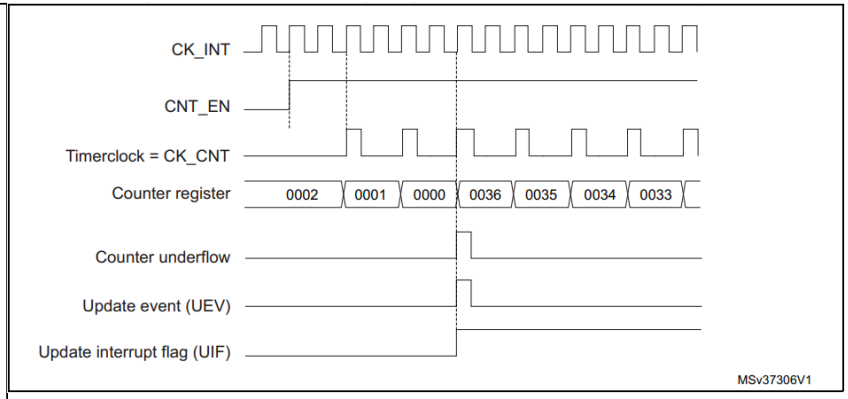
prescaler count

Timer

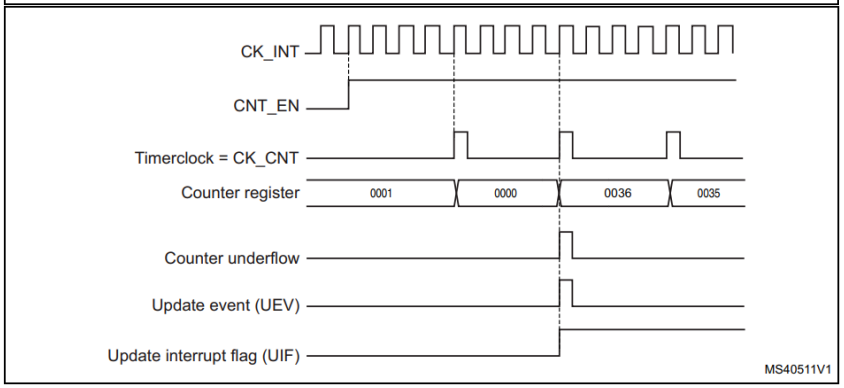
General-purpose timer

- down counting 모드 (TIMx_ARR=0x36)





prescaler count

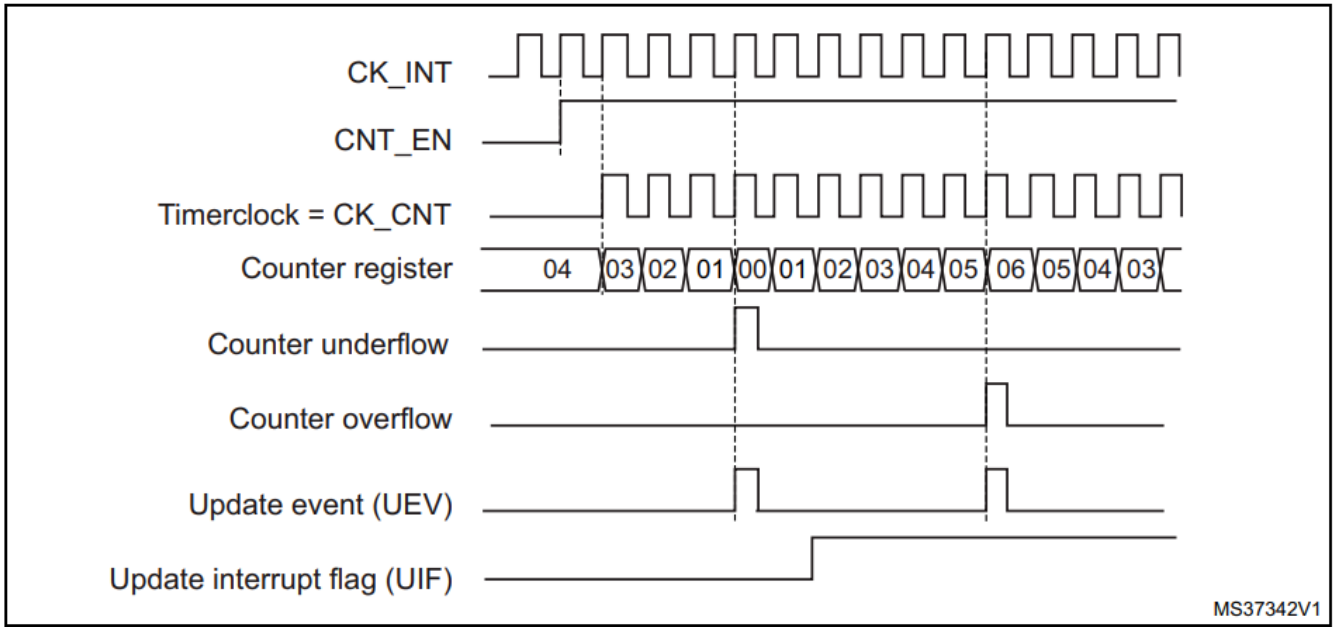


prescaler count

Timer

General-purpose timer

- Center-aligned 모드 (up/down counting (TIMx_ARR=0x06)

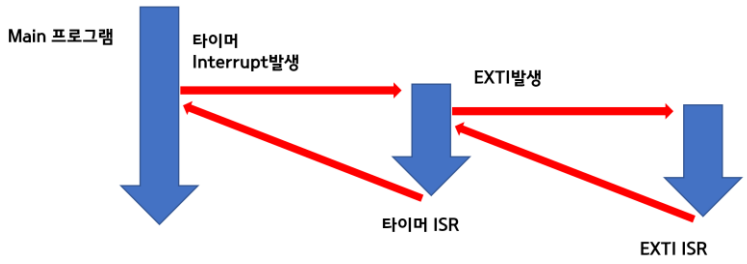


Timer

- 인터럽트 방식으로 동작함

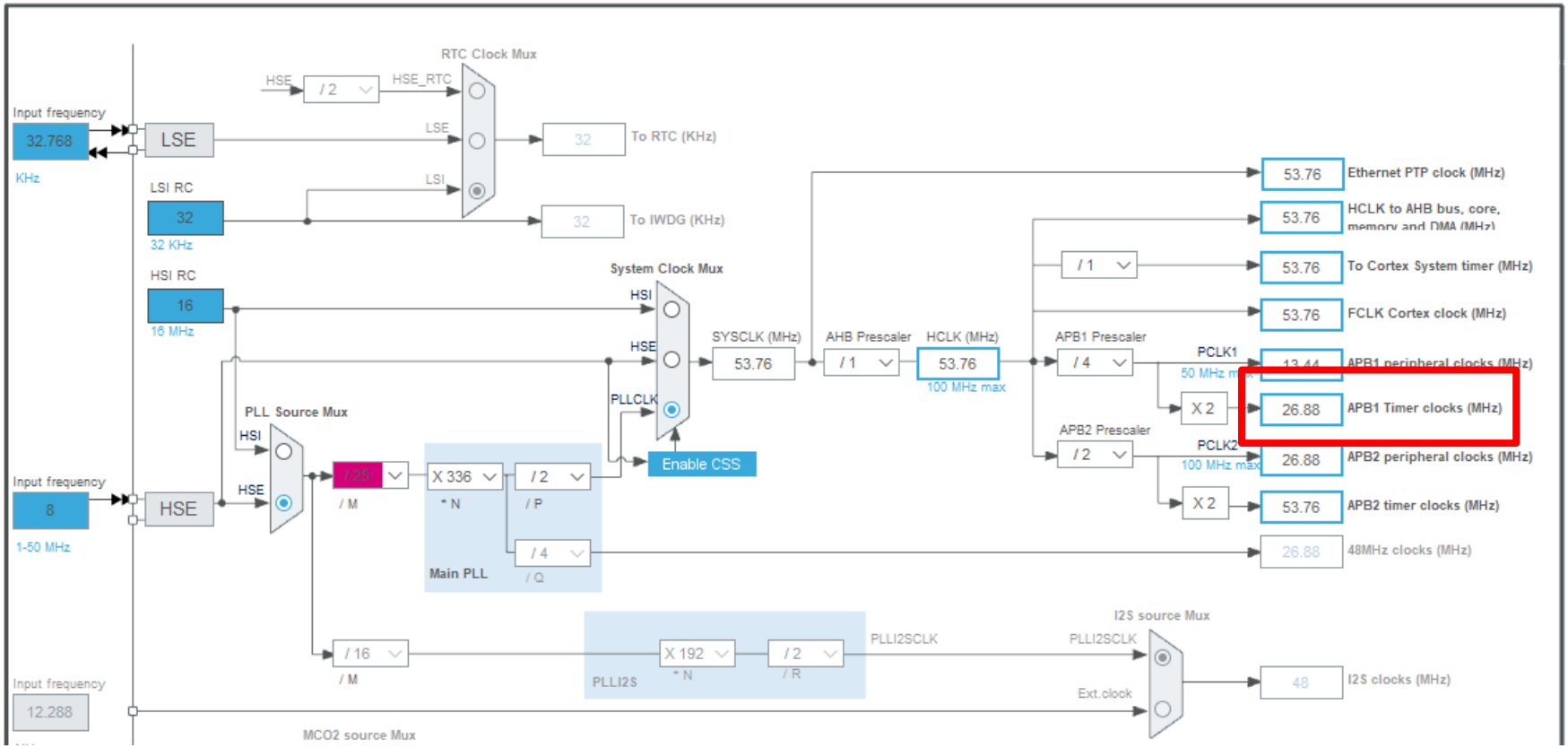
  Polling 방식으로 동작하는 것이 의미가 없음 (MCU는 클럭 카운트만 하게 될 것)

- 인터럽트 동작을 위한 레지스터 설정, 인터럽트가 발생했을 때의 handler구현이 필요함!
- + NVIC (Nested Vectored Interrupt Controller) 설정 필요
- + 언제 한번씩 인터럽트를 발생시킨 것인지 주기 결정 필요



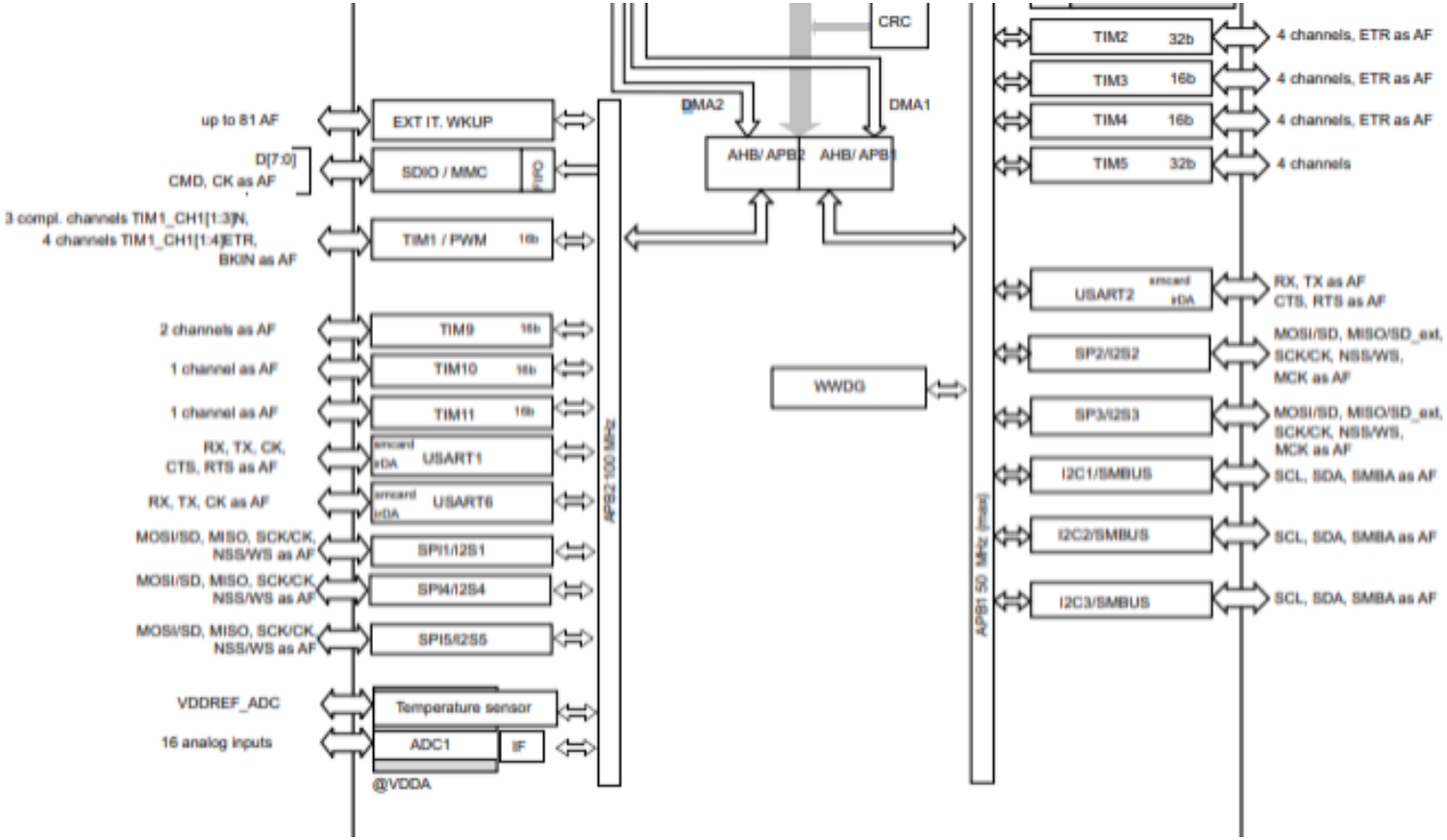
실습

- Nucleo-F411RE clock



실습

- Nucleo-F411RE Timer interface



실습

- Timer Interrupt Handler 이름
  - stm32f4xx.h 파일에 정의되어 있음

main.c | main_conf.h | **stm32f4xx.h [RO]** | startup_stm32f40_41xxx.s [RO]

```
  DMA1_Stream3_IRQn           = 14,     /*!< DMA1 Stream 3 global Interrupt                                    */
  DMA1_Stream4_IRQn           = 15,     /*!< DMA1 Stream 4 global Interrupt                                    */
  DMA1_Stream5_IRQn           = 16,     /*!< DMA1 Stream 5 global Interrupt                                    */
  DMA1_Stream6_IRQn           = 17,     /*!< DMA1 Stream 6 global Interrupt                                    */
  ADC_IRQn                    = 18,     /*!< ADC1, ADC2 and ADC3 global Interrupts                             */

#if defined(STM32F40_41xxx)
  CAN1_TX_IRQn                = 19,     /*!< CAN1 TX Interrupt                                                 */
  CAN1_RX0_IRQn               = 20,     /*!< CAN1 RX0 Interrupt                                                */
  CAN1_RX1_IRQn               = 21,     /*!< CAN1 RX1 Interrupt                                                */
  CAN1_SCE_IRQn               = 22,     /*!< CAN1 SCE Interrupt                                                */
  EXTI9_5_IRQn                = 23,     /*!< External Line[9:5] Interrupts                                     */
  TIM1_BRK_TIM9_IRQn          = 24,     /*!< TIM1 Break interrupt and TIM9 global interrupt                    */
  TIM1_UP_TIM10_IRQn          = 25,     /*!< TIM1 Update Interrupt and TIM10 global interrupt                  */
  TIM1_TRG_COM_TIM11_IRQn     = 26,     /*!< TIM1 Trigger and Commutation Interrupt and TIM11 global interrupt */
  TIM1_CC_IRQn                = 27,     /*!< TIM1 Capture Compare Interrupt                                    */
  TIM2_IRQn                   = 28,     /*!< TIM2 global Interrupt                                             */
  TIM3_IRQn                   = 29,     /*!< TIM3 global Interrupt                                             */
  TIM4_IRQn                   = 30,     /*!< TIM4 global Interrupt                                             */
  I2C1_EV_IRQn                = 31,     /*!< I2C1 Event Interrupt                                              */
  I2C1_ER_IRQn                = 32,     /*!< I2C1 Error Interrupt                                              */
  I2C2_EV_IRQn                = 33,     /*!< I2C2 Event Interrupt                                              */
  I2C2_ER_IRQn                = 34,     /*!< I2C2 Error Interrupt                                              */
  SPI1_IRQn                   = 35,     /*!< SPI1 global Interrupt                                             */
  SPI2_IRQn                   = 36,     /*!< SPI2 global Interrupt                                             */
  USART1_IRQn                 = 37,     /*!< USART1 global Interrupt                                           */
  USART2_IRQn                 = 38,     /*!< USART2 global Interrupt                                           */
  USART3_IRQn                 = 39,     /*!< USART3 global Interrupt                                           */
  EXTI15_10_IRQn              = 40,     /*!< External Line[15:10] Interrupts                                   */
  RTC_Alarm_IRQn              = 41,     /*!< RTC Alarm (A and B) through EXTI Line Interrupt                   */
  OTG_FS_WKUP_IRQn            = 42,     /*!< USB OTG FS Wakeup through EXTI line interrupt                     */
  TIM8_BRK_TIM12_IRQn         = 43,     /*!< TIM8 Break Interrupt and TIM12 global interrupt                   */
  TIM8_UP_TIM13_IRQn          = 44,     /*!< TIM8 Update Interrupt and TIM13 global interrupt                  */
  TIM8_TRG_COM_TIM14_IRQn     = 45,     /*!< TIM8 Trigger and Commutation Interrupt and TIM14 global interrupt */
  TIM8_CC_IRQn                = 46,     /*!< TIM8 Capture Compare Interrupt                                    */
  DMA1_Stream7_IRQn           = 47,     /*!< DMA1 Stream7 Interrupt                                            */
  FSMC_IRQn                   = 48,     /*!< FSMC global Interrupt                                             */
  SDIO_IRQn                   = 49,     /*!< SDIO global Interrupt                                             */
  TIM5_IRQn                   = 50,     /*!< TIM5 global Interrupt                                             */
  SPI3_IRQn                   = 51,     /*!< SPI3 global Interrupt                                             */
```

실습

- Timer Interrupt Handler 함수명
  - startup_stm32f40_41xxx.s 파일에 정의되어 있음

main.c | main_conf.h | stm32f4xx.h [RO] | **startup_stm32f40_41xxx.s [RO]** ×

```
DCD     EXTI2_IRQHandler                ; EXTI Line2
DCD     EXTI3_IRQHandler                ; EXTI Line3
DCD     EXTI4_IRQHandler                ; EXTI Line4
DCD     DMA1_Stream0_IRQHandler         ; DMA1 Stream 0
DCD     DMA1_Stream1_IRQHandler         ; DMA1 Stream 1
DCD     DMA1_Stream2_IRQHandler         ; DMA1 Stream 2
DCD     DMA1_Stream3_IRQHandler         ; DMA1 Stream 3
DCD     DMA1_Stream4_IRQHandler         ; DMA1 Stream 4
DCD     DMA1_Stream5_IRQHandler         ; DMA1 Stream 5
DCD     DMA1_Stream6_IRQHandler         ; DMA1 Stream 6
DCD     ADC_IRQHandler                  ; ADC1, ADC2 and ADC3s
DCD     CAN1_TX_IRQHandler              ; CAN1 TX
DCD     CAN1_RX0_IRQHandler             ; CAN1 RX0
DCD     CAN1_RX1_IRQHandler             ; CAN1 RX1
DCD     CAN1_SCE_IRQHandler             ; CAN1 SCE
DCD     EXTI9_5_IRQHandler              ; External Line[9:5]s
DCD     TIM1_BRK_TIM9_IRQHandler        ; TIM1 Break and TIM9
DCD     TIM1_UP_TIM10_IRQHandler        ; TIM1 Update and TIM10
DCD     TIM1_TRG_COM_TIM11_IRQHandler   ; TIM1 Trigger and Commutation and TIM11
DCD     TIM1_CC_IRQHandler              ; TIM1 Capture Compare
DCD     TIM2_IRQHandler                 ; TIM2
DCD     TIM3_IRQHandler                 ; TIM3
DCD     TIM4_IRQHandler                 ; TIM4
DCD     I2C1_EV_IRQHandler              ; I2C1 Event
DCD     I2C1_ER_IRQHandler              ; I2C1 Error
DCD     I2C2_EV_IRQHandler              ; I2C2 Event
DCD     I2C2_ER_IRQHandler              ; I2C2 Error
DCD     SPI1_IRQHandler                 ; SPI1
DCD     SPI2_IRQHandler                 ; SPI2
DCD     USART1_IRQHandler               ; USART1
DCD     USART2_IRQHandler               ; USART2
DCD     USART3_IRQHandler               ; USART3
DCD     EXTI15_10_IRQHandler            ; External Line[15:10]s
DCD     RTC_Alarm_IRQHandler            ; RTC Alarm (A and B) through EXTI Line
DCD     OTG_FS_WKUP_IRQHandler          ; USB OTG FS Wakeup through EXTI line
DCD     TIM8_BRK_TIM12_IRQHandler       ; TIM8 Break and TIM12
DCD     TIM8_UP_TIM13_IRQHandler        ; TIM8 Update and TIM13
DCD     TIM8_TRG_COM_TIM14_IRQHandler   ; TIM8 Trigger and Commutation and TIM14
DCD     TIM8_CC_IRQHandler              ; TIM8 Capture Compare
```

실습

실습 1: User LED를 1초 간격을 on/off

- 메인함수 헤더 추가
  - #include "stm32f4xx.h"
- 기존에 사용하는 함수 가져오기
  - LED_Init (port A, 5번핀을 output으로 제어)

```
void LED_init(void)
{
  RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);
  GPIO_InitTypeDef GPIO_InitStructure;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;
  GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
  GPIO_Init(GPIOA, &GPIO_InitStructure);
}
```

실습

실습 1: User LED를 1초 간격을 on/off

- **Timer 2번 사용**

```
void TIM2_Configuration(void)
{
 RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);

 TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
 NVIC_InitTypeDef NVIC_InitStructure;

 TIM_TimeBaseStructure.TIM_Prescaler = 26880 - 1;
 TIM_TimeBaseStructure.TIM_Period = 1000 - 1;

 TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
 TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);

 TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
 TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE);
 TIM_Cmd(TIM2, ENABLE);

  NVIC_PriorityGroupConfig(NVIC_PriorityGroup_1);
  NVIC_InitStructure.NVIC_IRQChannel = TIM2_IRQn;
  NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
  NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
  NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Init(&NVIC_InitStructure);
}
```

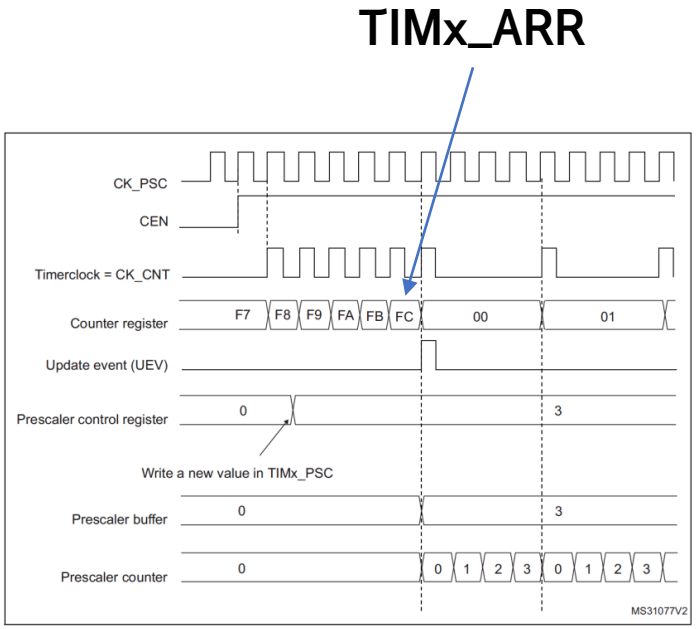
Timer2에 clock를 공급

Timer 사용을 위한 구조체 변수 선언

TIMx_PSC, TIMx_ARR 설정을 위한 값 지정

실습

실습 1: User LED를 1초 간격을 on/off




```
TIM_TimeBaseStructure.TIM_Prescaler = 26880 - 1;
TIM_TimeBaseStructure.TIM_Period = 1000 - 1;
```

TIMx_PSC, TIMx_ARR 설정을 위한 값 지정

$$\frac{SYSTEM\ CLOCK}{Prescaler값 \times Period값} = Freq. \Rightarrow \frac{26.88MHz}{26880 \times 1000} = 1Hz$$

실습

실습 1: User LED를 1초 간격을 on/off

```
void TIM2_Configuration(void)
{
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);

  TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
  NVIC_InitTypeDef NVIC_InitStructure;

  TIM_TimeBaseStructure.TIM_Prescaler = 26880 - 1;
  TIM_TimeBaseStructure.TIM_Period = 1000- 1;

  TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
  TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);


  TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
  TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE);
  TIM_Cmd(TIM2, ENABLE);

  …
}
```

Count를 증가시키는 모드
TIM_ConterMode_Up
TIM_ConterMode_Down

설정된 구조체를 이용하여 Timer2 초기화

Timer2가 설정된 count값까지 도달하면 interrupt가 발생하도록 활성화

Timer2의 count 시작 (CEN == 1)

실습

실습 1: User LED를 1초 간격을 on/off

Timer 2에 인터럽트가 발생하면, 이 함수로 연결

```
void TIM2_IRQHandler(void)
{
  if(TIM_GetITStatus(TIM2, TIM_IT_Update) != RESET)
  {
    GPIO_ToggleBits(GPIOA, GPIO_Pin_5);

    // clear interrupt flag
    TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
  }
}
```

Timer 2에 인터럽트가 발생했다면,

A포트 5번 핀은 toggle 시켜라
(0 → 1, 1 → 0)

인터럽트 플래그 초기화

```
int main()
{
  LED_init();
  TIM2_Configuration();
  while(1)
  {
  }

  return 0;
}
```

실습

실습 2: User LED를 원하는 시간 간격으로 on/off

$$\frac{SYSTEM\ CLOCK}{Prescaler값 \times Period값} = Freq. \Rightarrow \frac{26.88MHz}{26880 \times 1000} = 1Hz$$

```
void TIM2_Configuration(int interval_ms)
{
 RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);

 TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
 NVIC_InitTypeDef NVIC_InitStructure;

 TIM_TimeBaseStructure.TIM_Prescaler = 26880 - 1;
 TIM_TimeBaseStructure.TIM_Period = interval_ms - 1;

 TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
 TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);


 TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
 TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE);
 TIM_Cmd(TIM2, ENABLE);

 …
}
```

millisecond (ms)로 설정됨

```
int main()
{
 LED_init();
 TIM2_Configuration(1000);
 while(1)
 {
 }

 return 0;
}
```

실습

실습 3: 1초마다 FND의 출력 숫자 증가시키기

- Font[10] 배열 가져오기
- FND_Init() 함수 가져오기
- Delay() 함수 가져오기

```
int FND_cnt = 0;
unsigned char Font[10] = {0xC0, 0xF9, 0xA4, 0xB0, 0x99,
0x92, 0x82, 0xD8, 0x80, 0x90};
```

```
void FND_Init(void)
{
  RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOC, ENABLE);
  GPIO_InitTypeDef GPIO_InitStructure;
   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_1 | GPIO_Pin_2 | GPIO_Pin_3 |
GPIO_Pin_4 | GPIO_Pin_5 | GPIO_Pin_6 | GPIO_Pin_7;
  GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
   GPIO_Init(GPIOC, &GPIO_InitStructure);

  RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOB, ENABLE);
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_1 | GPIO_Pin_2 | GPIO_Pin_3;
  GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;

  GPIO_Init(GPIOB, &GPIO_InitStructure);
}
```

```
void Delay(__IO uint32_t nCount)
{
  for(; nCount != 0; nCount--);
}
```

실습

실습 3: 1초마다 FND의 출력 숫자 증가시키기

- Count_Progress(int d_3, int d_2, int d_1, int d_0) 함수 가져오기
  (안가져오고 직접 구현해도 됨)

```
void Count_Progress(int d_3, int d_2, int d_1, int d_0)
{
  GPIO_Write(GPIOB, 0x0008);
  GPIO_Write(GPIOC, Font[d_0]);
  Delay(1000);

  GPIO_Write(GPIOB, 0x0004);
  GPIO_Write(GPIOC, Font[d_1]);
  Delay(1000);

  GPIO_Write(GPIOB, 0x0002);
  GPIO_Write(GPIOC, Font[d_2]);
  Delay(1000);

  GPIO_Write(GPIOB, 0x0001);
  GPIO_Write(GPIOC, Font[d_3]);
  Delay(1000);
}
```

실습

실습 3: 1초마다 FND의 출력 숫자 증가시키기

- TIM2 번 사용
- TIM2_Configuration(int interval_ms)
- void TIM2_IRQHandler(void) // 내부 수정해야 함

  Timer 2에 인터럽트가 걸리면, FND_cnt변수를 증가시킴

```
void TIM2_IRQHandler(void)
{
 if(TIM_GetITStatus(TIM2, TIM_IT_Update) != RESET)
 {
  FND_cnt++;

  if(FND_cnt >= 10000)
  {
   FND_cnt = 0;
  }

  // clear interrupt flag
  TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
 }
}
```

실습

실습 3: 1초마다 FND의 출력 숫자 증가시키기

```
int main()
{
  FND_Init();
  TIM2_Configuration(1000);

  while(1)
  {
    Count_Progress(FND_cnt/1000, ((FND_cnt/100)%10), ((FND_cnt/10)%10), (FND_cnt%10));
  }

  return 0;
}
```

실습

실습 4: 1초마다 FND의 출력 숫자 증가시키기
+ User 스위치 눌리면 (interrupt로), FND 출력 값 0으로 초기화

- EXTI (external interrupt)를 사용해야 하므로 아래 프로젝트 파일 추가
  - 폴더 :

    STM32F4xx_DSP_StdPeriph_Lib_V1.8.0₩Libraries₩STM32F4xx_StdPeriph_Driver₩src
  - 파일 : misc.c, stm32f4xx_exti.c, stm32f4xx_tim.c, stm32f4xx_syscfg.c

실습

실습 4: 1초마다 FND의 출력 숫자 증가시키기
+ User 스위치 눌리면 (interrupt로), FND 출력 값 0으로 초기화

- **실습 3 코드 +**
- **지난 시간의 EXTI13_configuration(), EXTI15_10_IRQHandler(), SW_Init() 함수 가져오기**

실습

실습 4: 1초마다 FND의 출력 숫자 증가시키기
+ User 스위치 눌리면 (interrupt로), FND 출력 값 0으로 초기화

스위치가 눌리면 값을 0으로 바꿈

```
void EXTI15_10_IRQHandler(void)
{
  // Make sure that interrupt flag is set
  if (EXTI_GetITStatus(EXTI_Line13) != RESET)
  {
    FND_cnt = 0;
    EXTI_ClearITPendingBit(EXTI_Line13);
  }
}
```

```
int main()
{
 FND_Init();
 SW_init();
 TIM2_Configuration(1000);
 EXTI13_Configuration();

 while(1)
 {
  Count_Progress(FND_cnt/1000, ((FND_cnt/100)%10),
((FND_cnt/10)%10), (FND_cnt%10));
 }

 return 0;
}
```

과제

- Timer 2개 사용
  - 2번 타이머 (TIM2)로 FND 1초마다 출력 값 1씩 증가
  - 3번 타이머 (TIM3)로 User LED를 0.5초마다 점멸

- External interrupt로 스위치 제어
  - User switch가 눌리면 출력 값을 0으로 초기화

Thank you.