2.Timer

DR.SOMSIN THONGKRAIRAT


TIM IS JUST
A STOP WATCH
PWM, ENCODER, OS,
NEO PIXEL, FREQUENCY
DIVIDER, OVERCLOCK, RTC
imgflip.com

What is timer

- Programable counting module (up-down counter)

- Selective clock source (internal clock , external signal, programmatically)

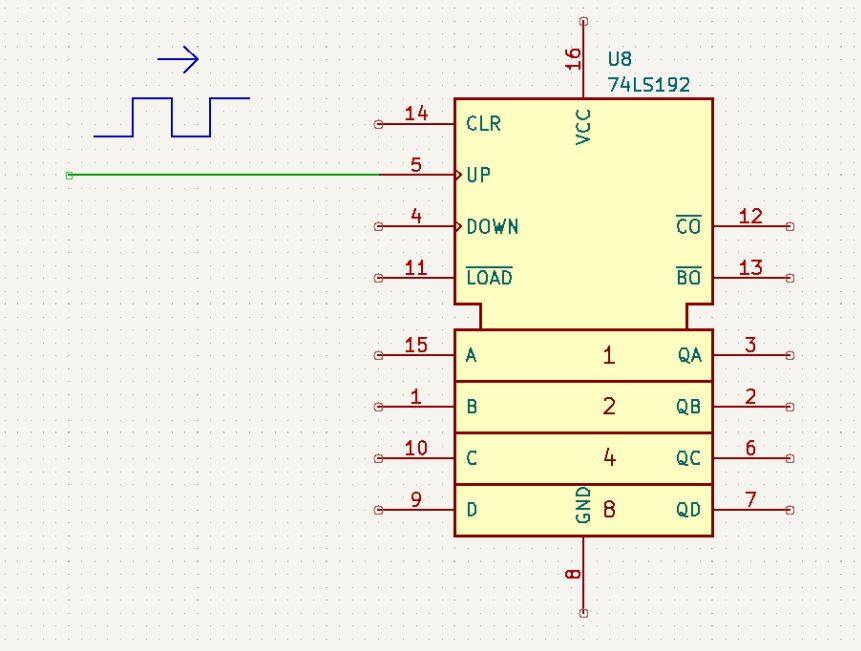
- Signal generator

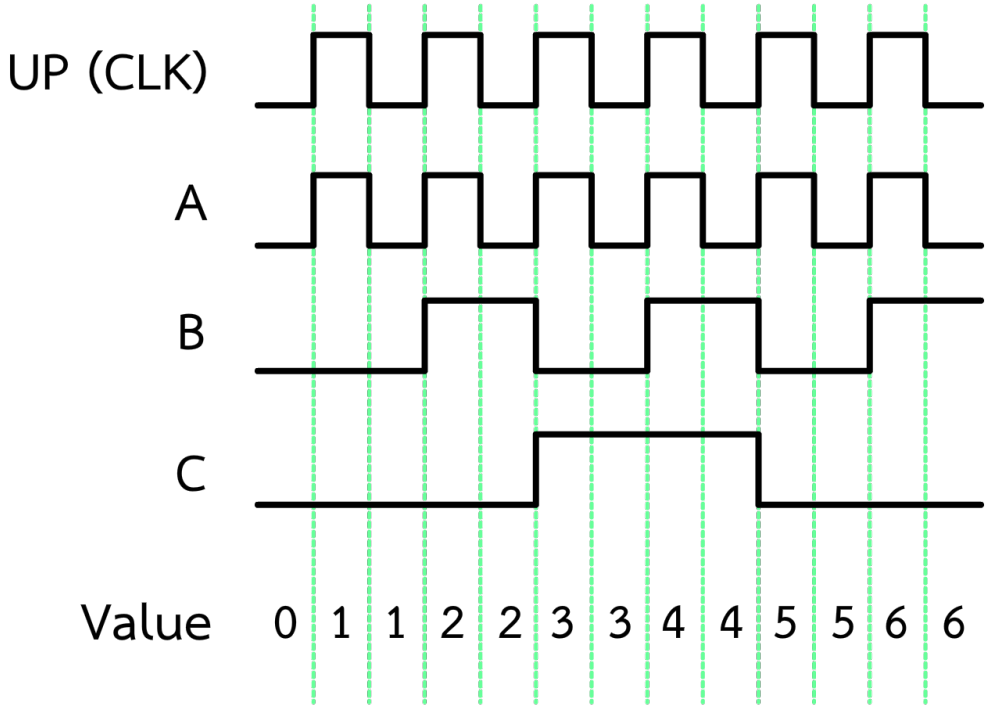
Application

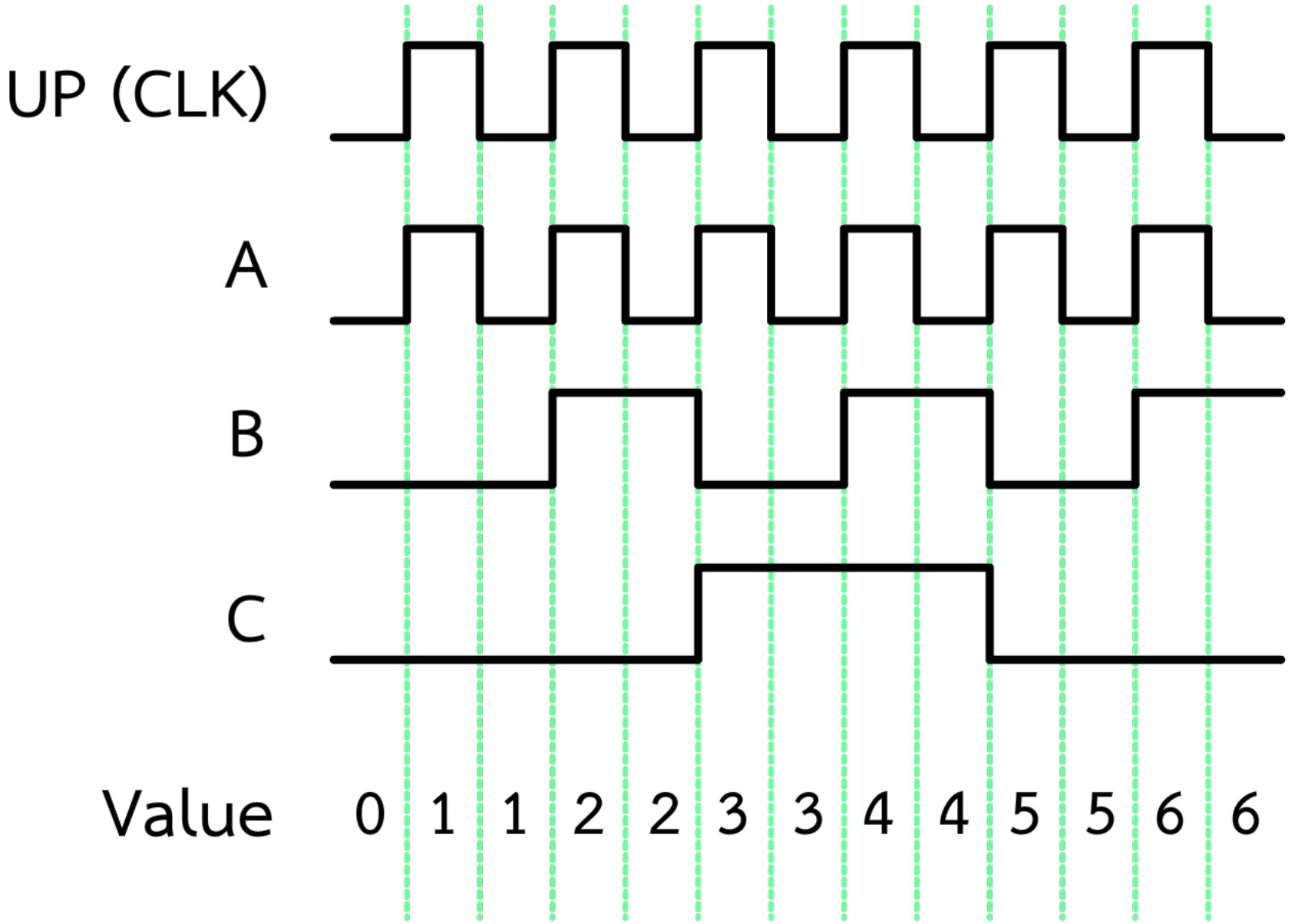
- Timer Period Trigger
- PWM
- Encoder manager
- Interrupt & Event

Counting meaning

- counting circuit (synchronous counter for example)

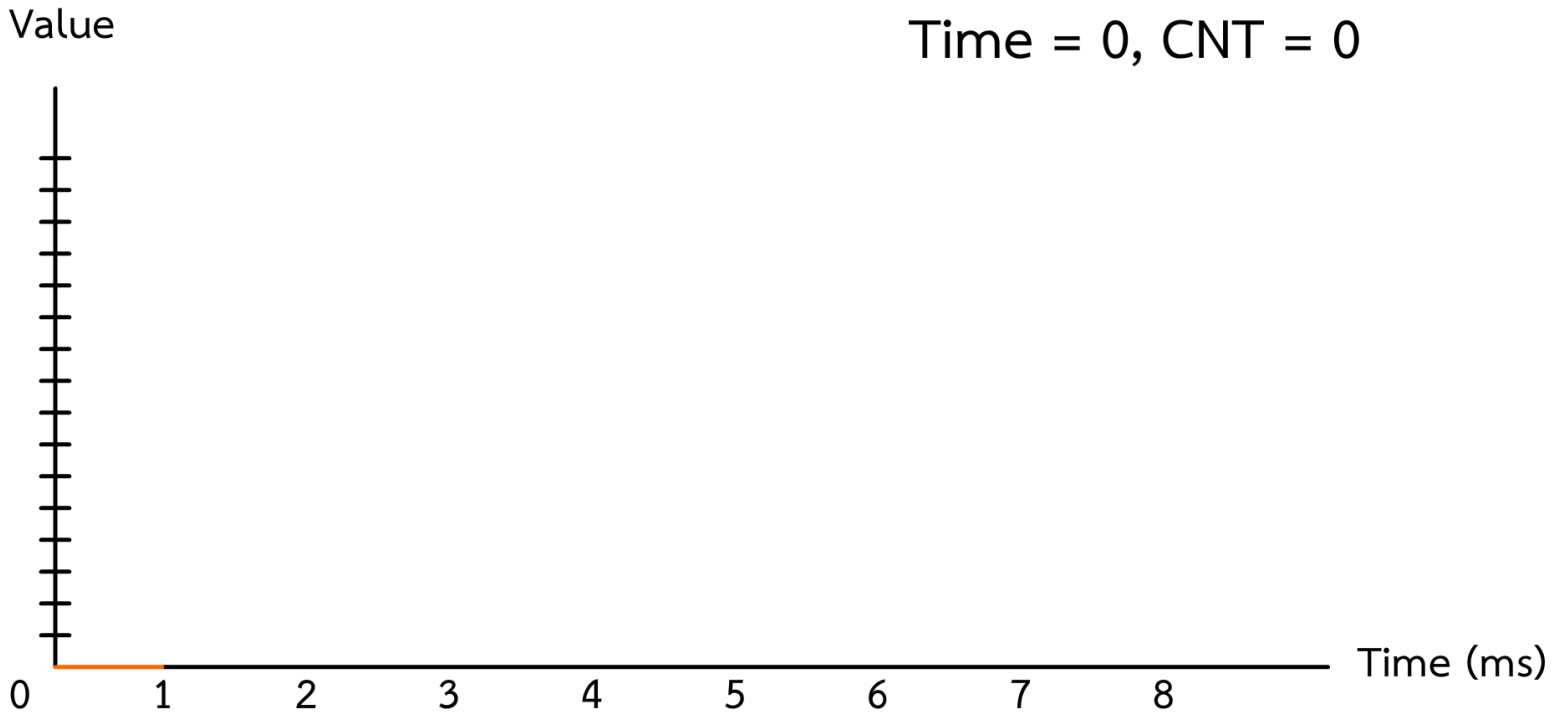





UP (CLK)
A
B
C
Value
0 1 1 2 2 3 3 4 4 5 5 6 6

Principle of counter module

- assume we continuously increase some variable call CNT 1 unit per 1 ms

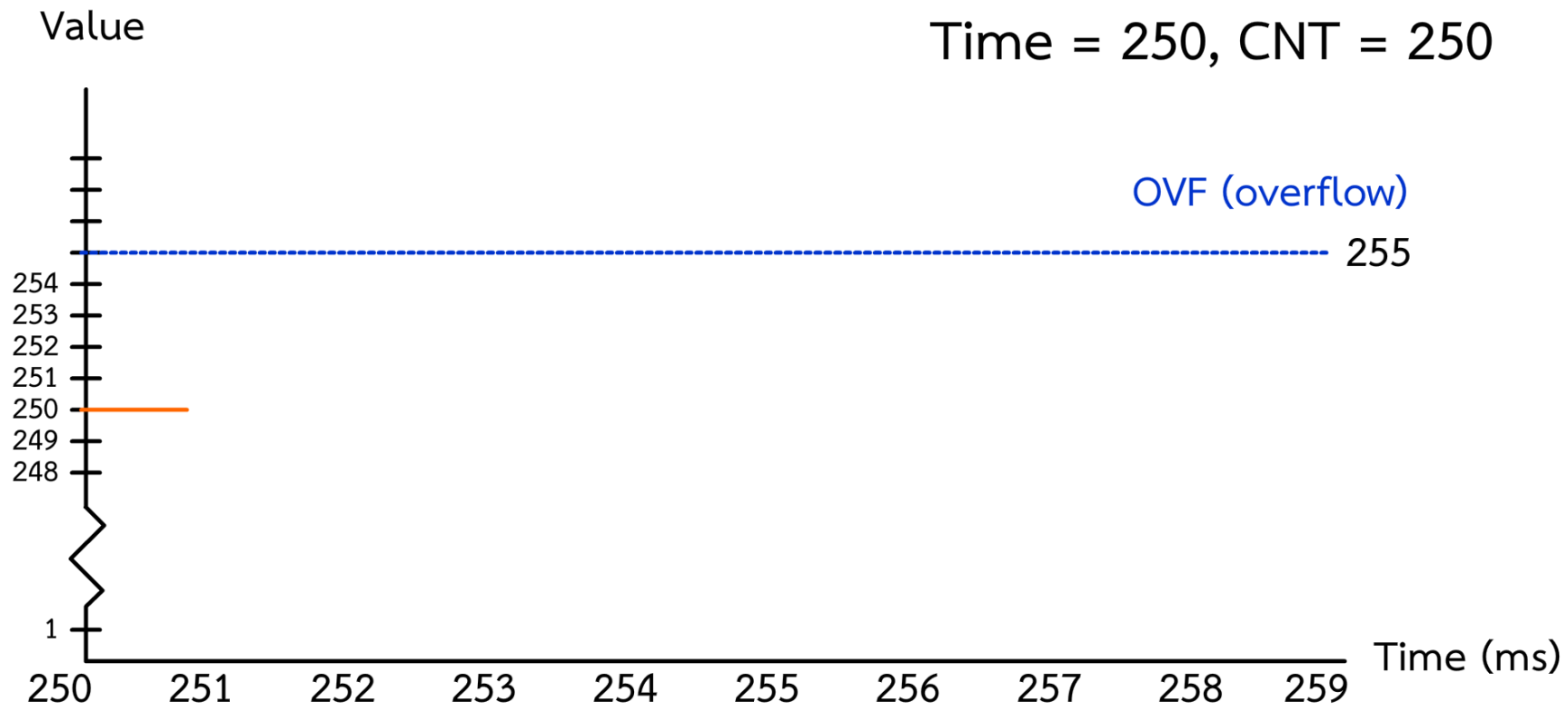


Overflow (OVF)

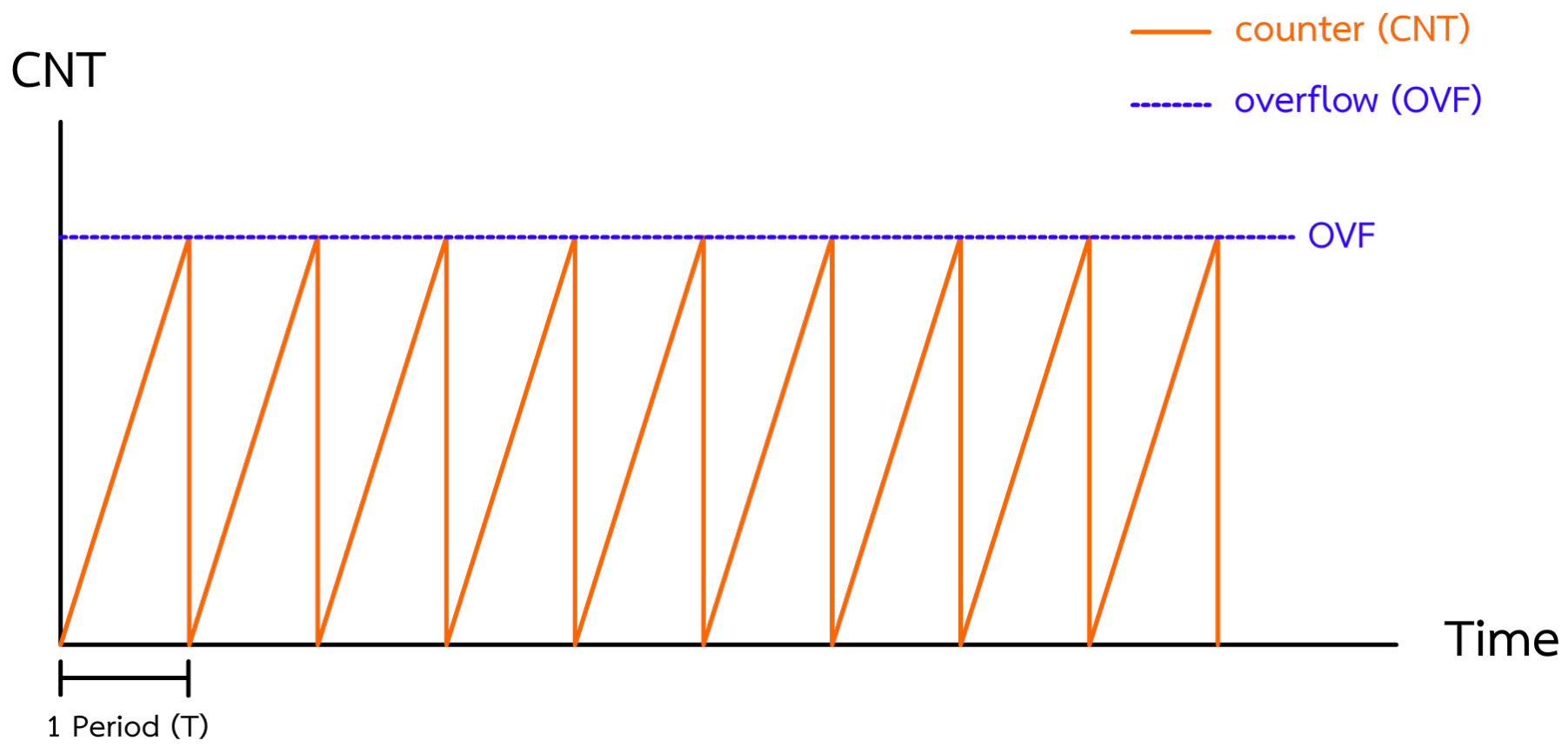
- ตัวกำหนดค่าสูงสุดของ CNT
- เมื่อ CNT count up ไปเกินกว่า OVF จะทำการ Rest ค่าของ CNT
- ทำให้เกิด overflow (OVF) interrupt

Principle of counter module

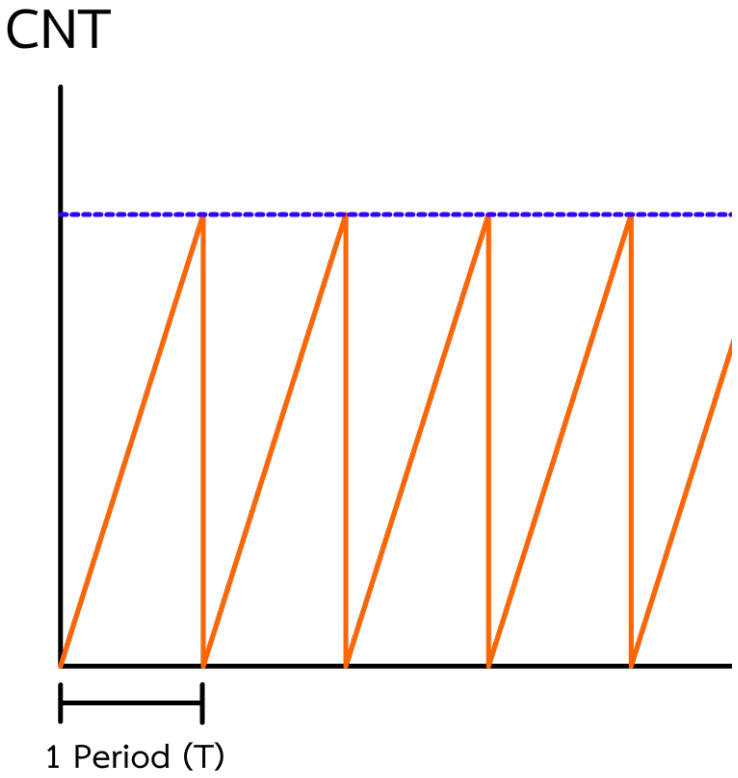
- assume we continuously increase some variable call CNT 1 unit per 1 ms



Principle of counter module



Period



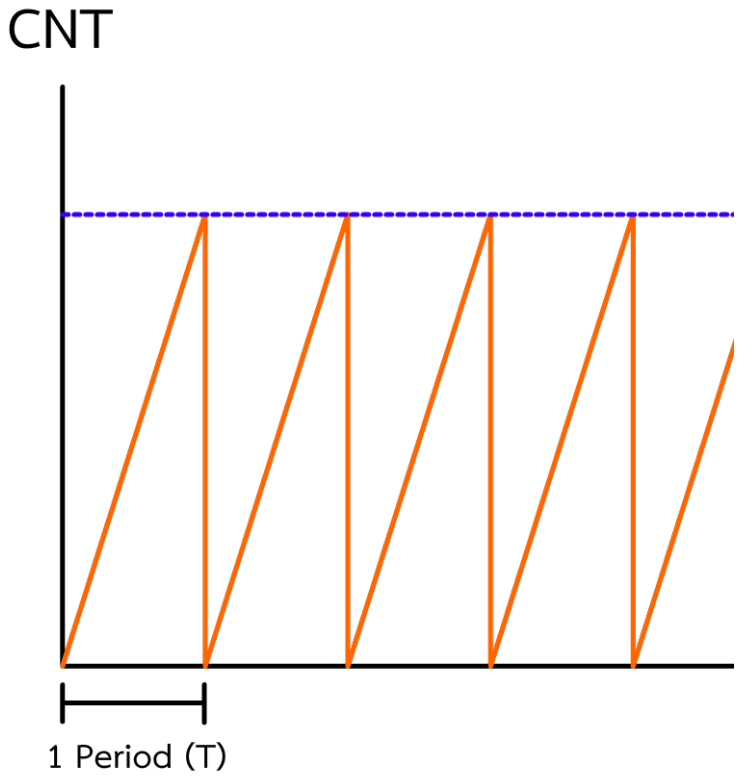
How long for 1 Period ?

1 Period นานเท่าไหร่ ?

Depend on counting rate (counting Frequency) and OVF

ขึ้นอยู่กับความเร็ว (ความถี่) ของการนับ และ OVF

Period



$$1\ T\ (1\ Period) = \frac{OVF}{Input\ clock\ (Hz)}$$

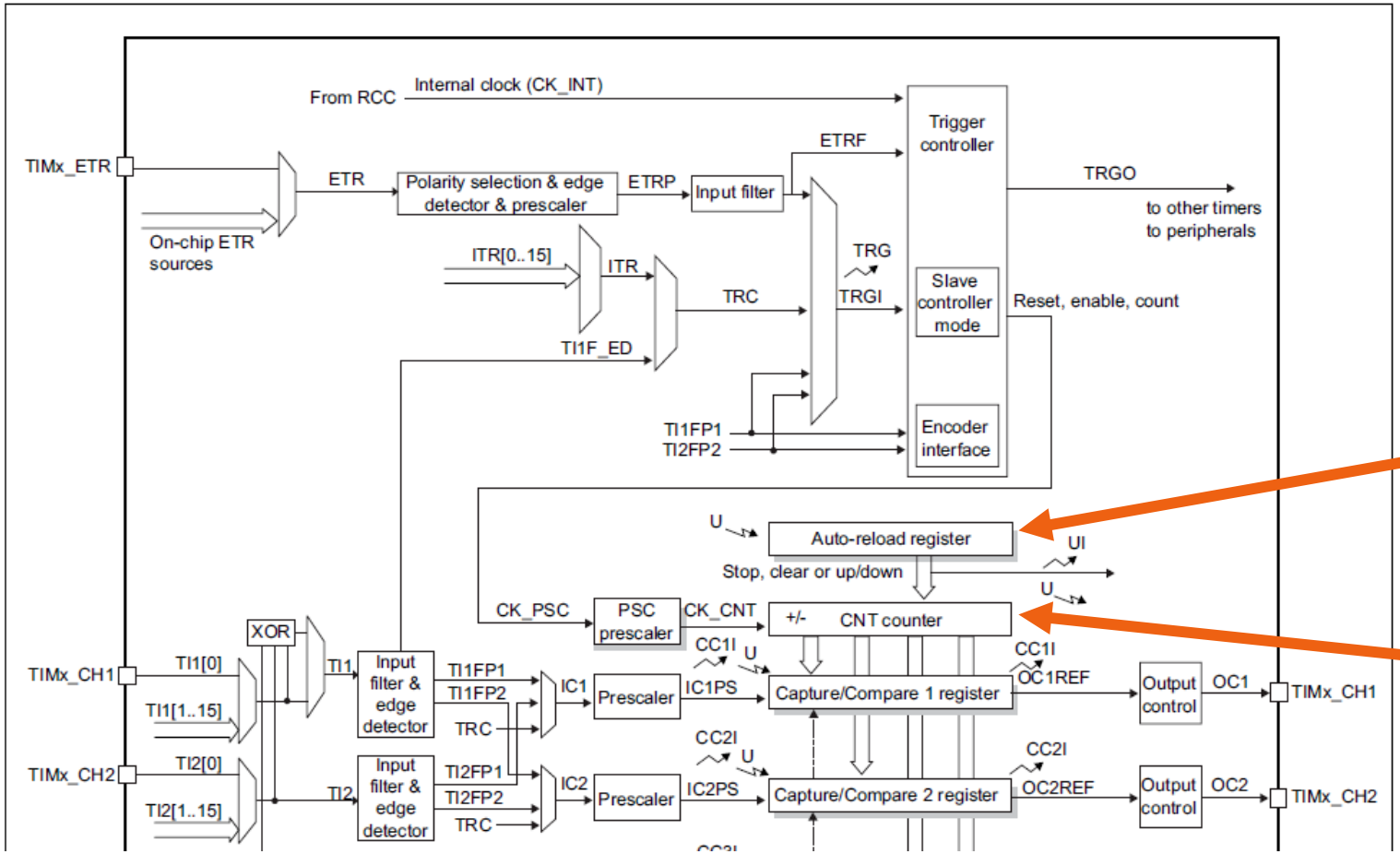
Ex1. Clock = 1MHz , OVF = 256

1 T = 0.000256 sec = 256uS

Ex2. Clock = 8MHz , OVF = 10000

1 T = 0.00125 sec = 1.25mS

Typical TIM infrastructure



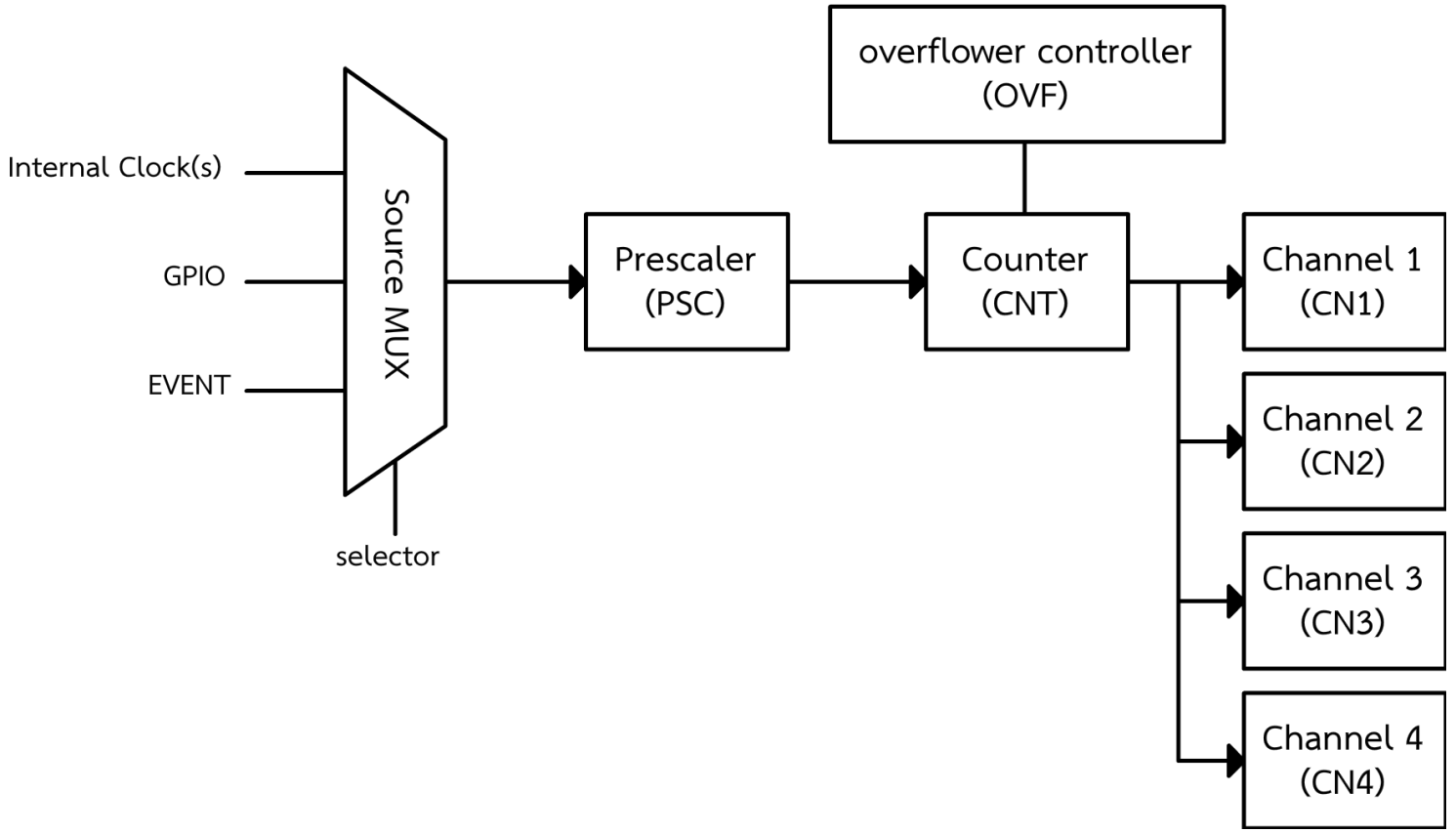
Too complicated ?

OVF

CNT

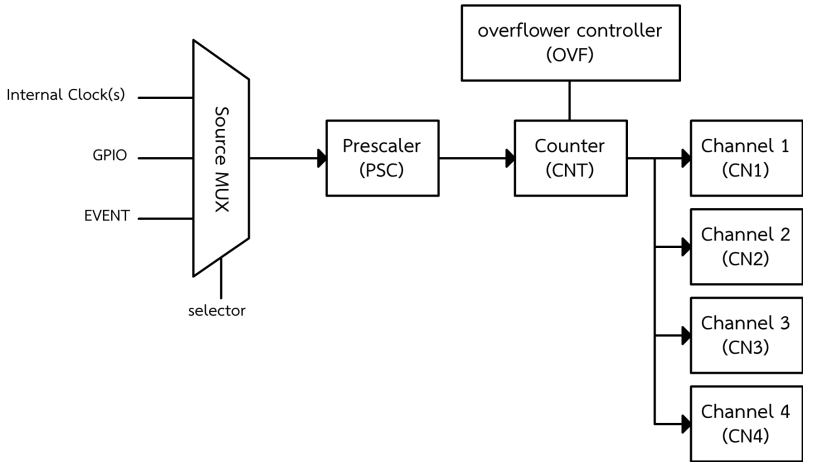
*From RM0444 [page 626]

Typical Clock infrastructure



TIM component

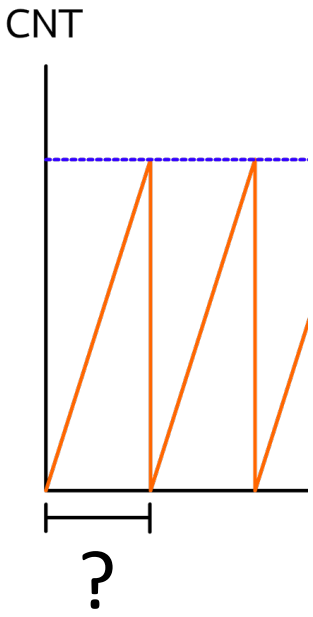
- Source MUX -> MUX เพื่อเลือก clock (สัญญาณ) ที่ใช้นับ
- Prescaler (PSC) -> ตัวหารก่อนเข้า Timer เพื่อให้ความถี่ลดลง
- Counter (CNT) -> ตัวนับ
- Overflow (OVF) -> ตัวเริ่มต้นนับใหม่
- Output channel (TIMx_CHx) -> output

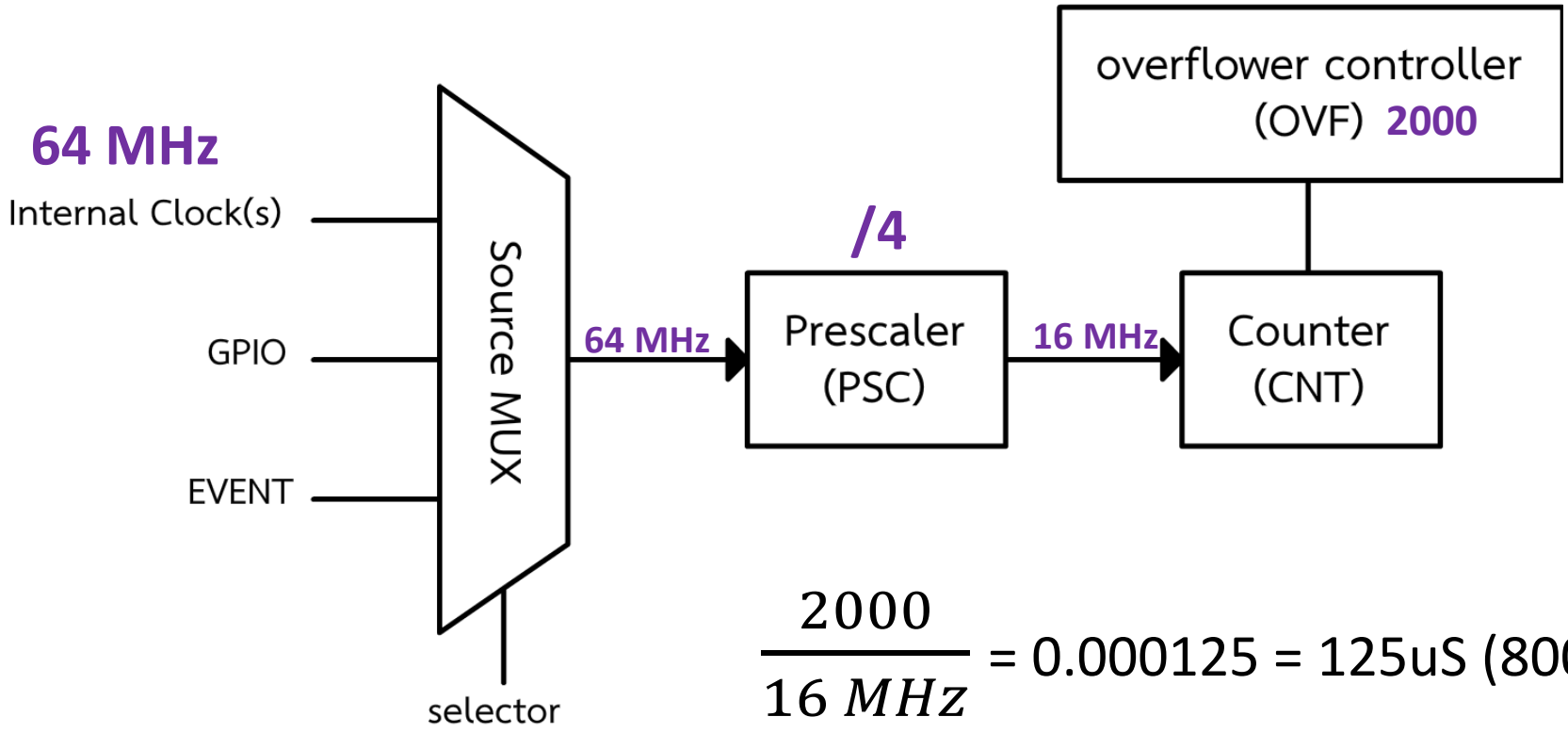


*X can be 1,2,3,4 (selected timer and channel)

Example

- clock source = PPL clock 64MHz , PSC = 4 , CNT = 32, OVF = 2000

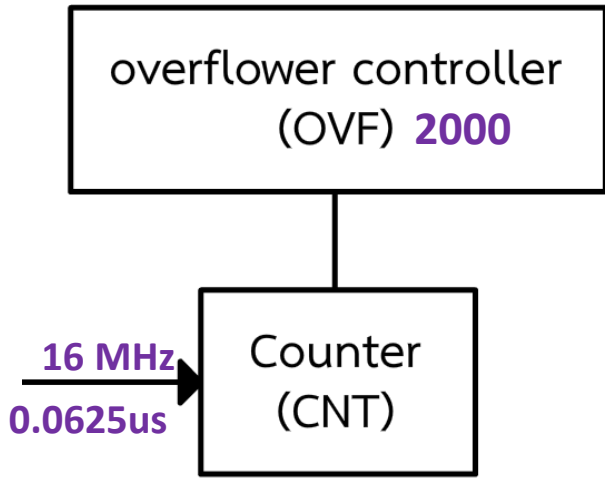


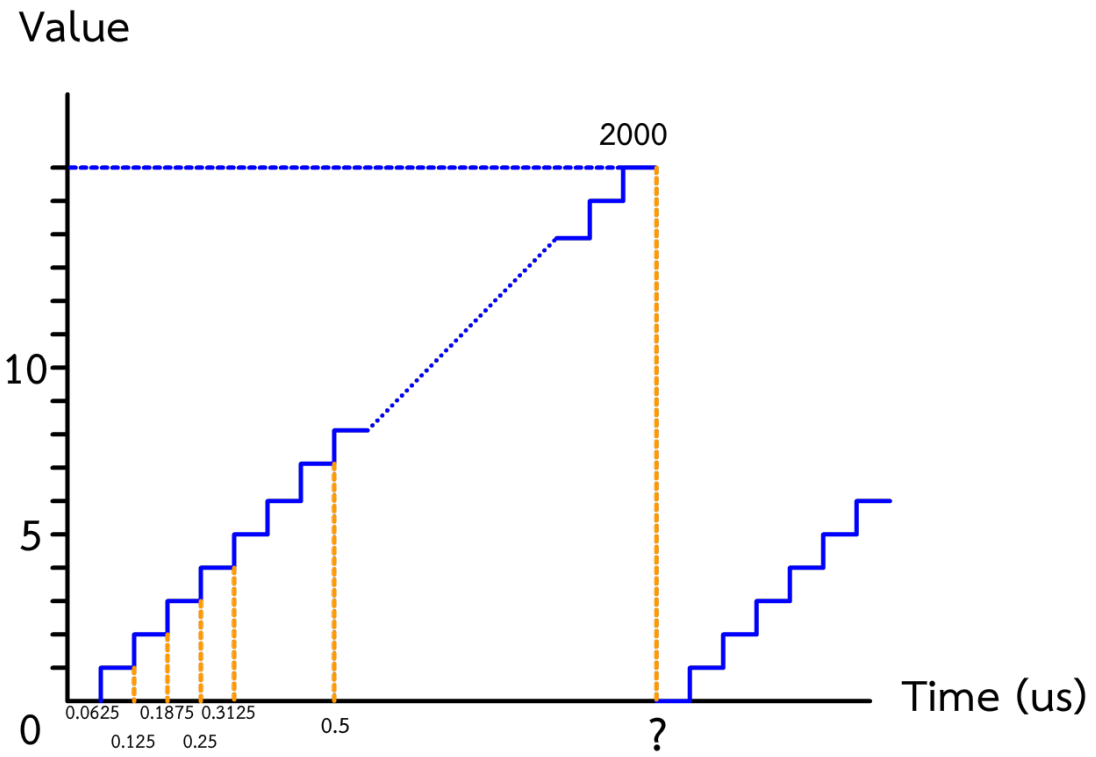


$$\frac{2000}{16\ MHz} = 0.000125 = 125uS\ (8000Hz)$$

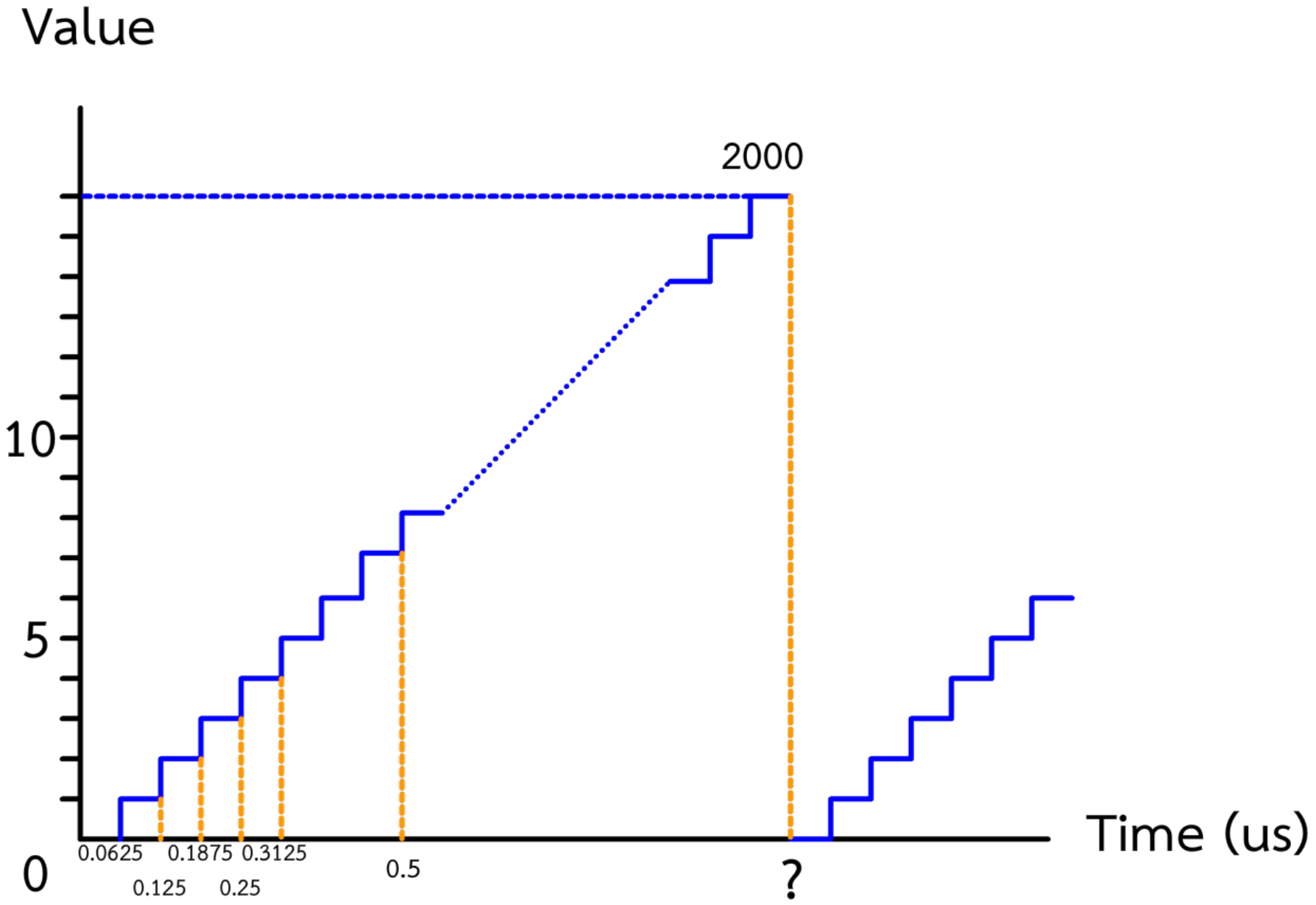
Example

- clock source = PPL clock 64MHz , PSC = 4 , CNT = 32, OVF = 2000

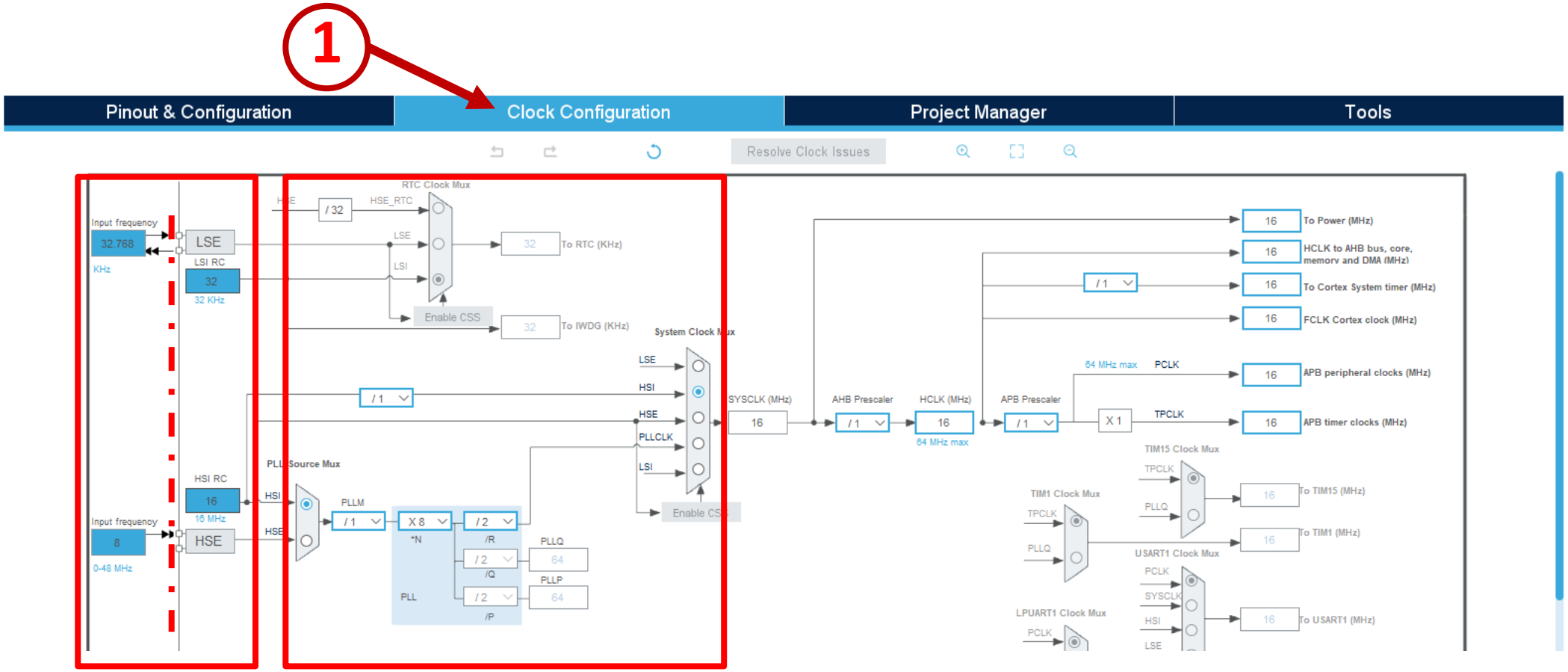




$$\frac{2000}{16\ MHz} = 0.000125 = 125uS\ (8000Hz)$$


Value
2000
10
5
0
0.0625
0.125
0.1875
0.25
0.3125
0.5
?
Time (us)

STM32 clock configuration menu



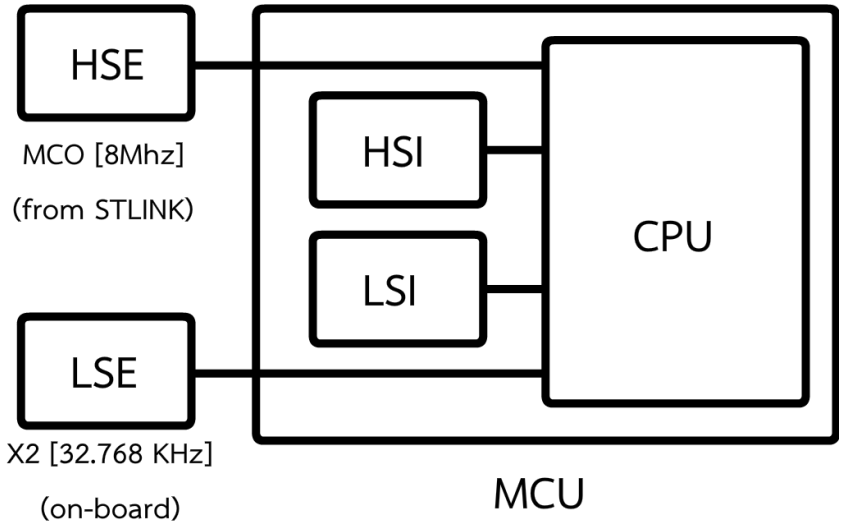
Input state

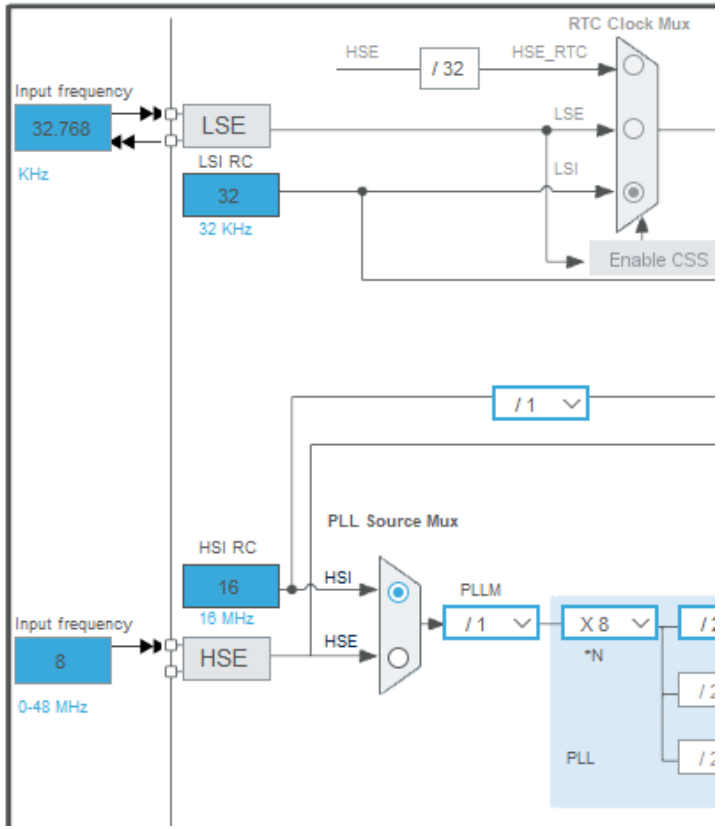
Frequency adjust state
(Phase lock loop)

*Left to right pattern

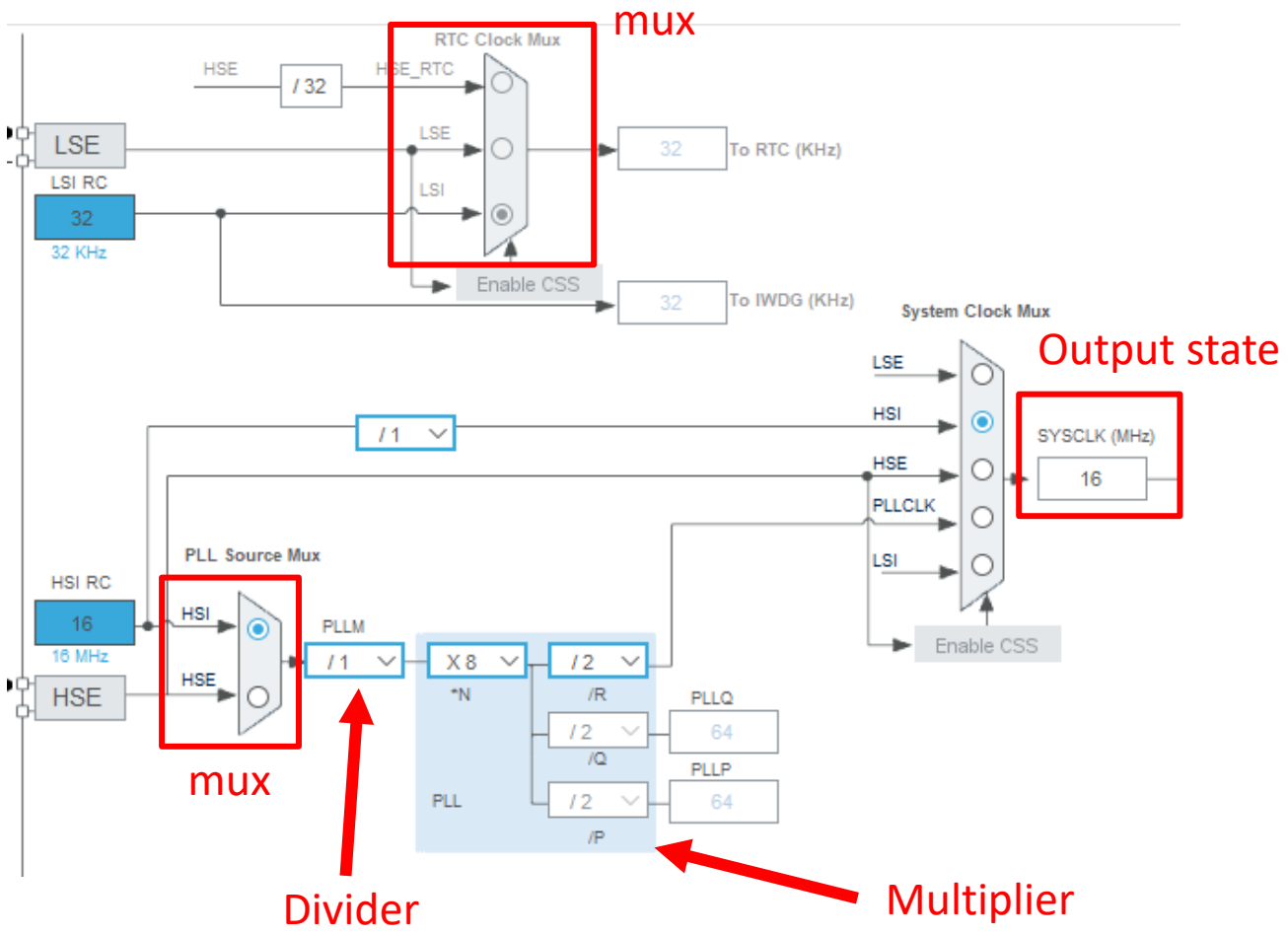
Stm32 Clock input state

LSE : Low Speed Clock (External)
LSI : Low Speed Clock (Internal)
HSE : High Speed Clock (External)
HSE : High Speed Clock (Internal)



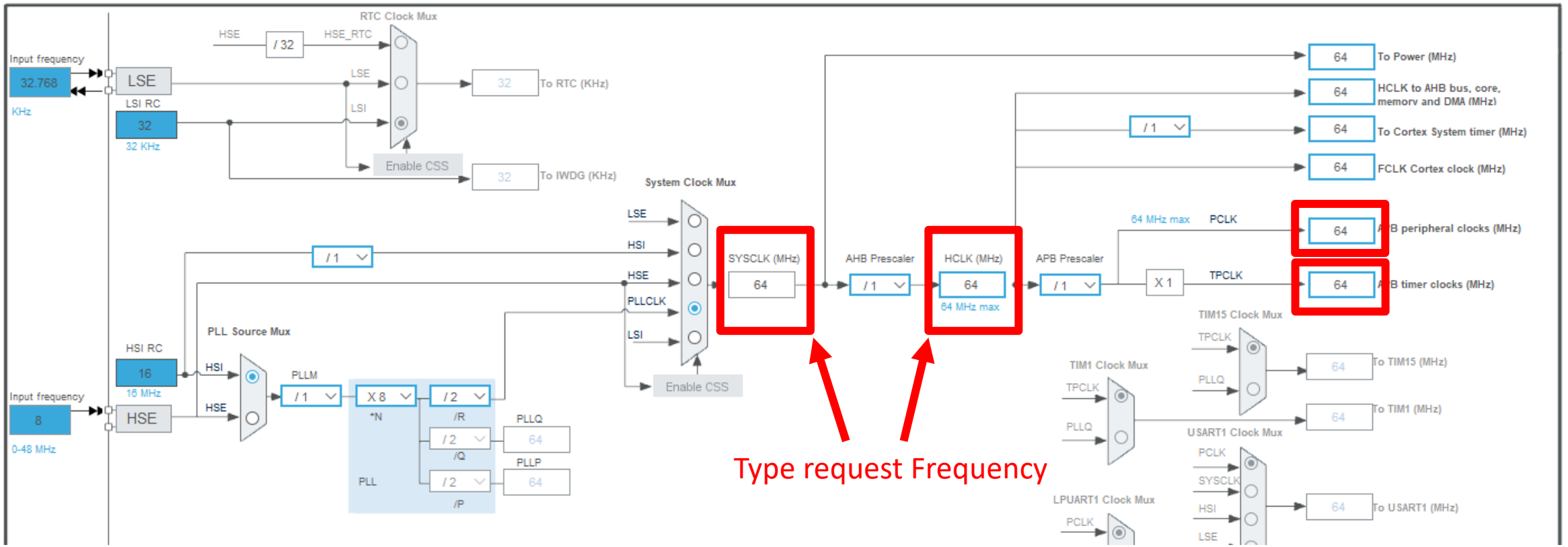


PPL : Phase lock loop



Cube MX setup

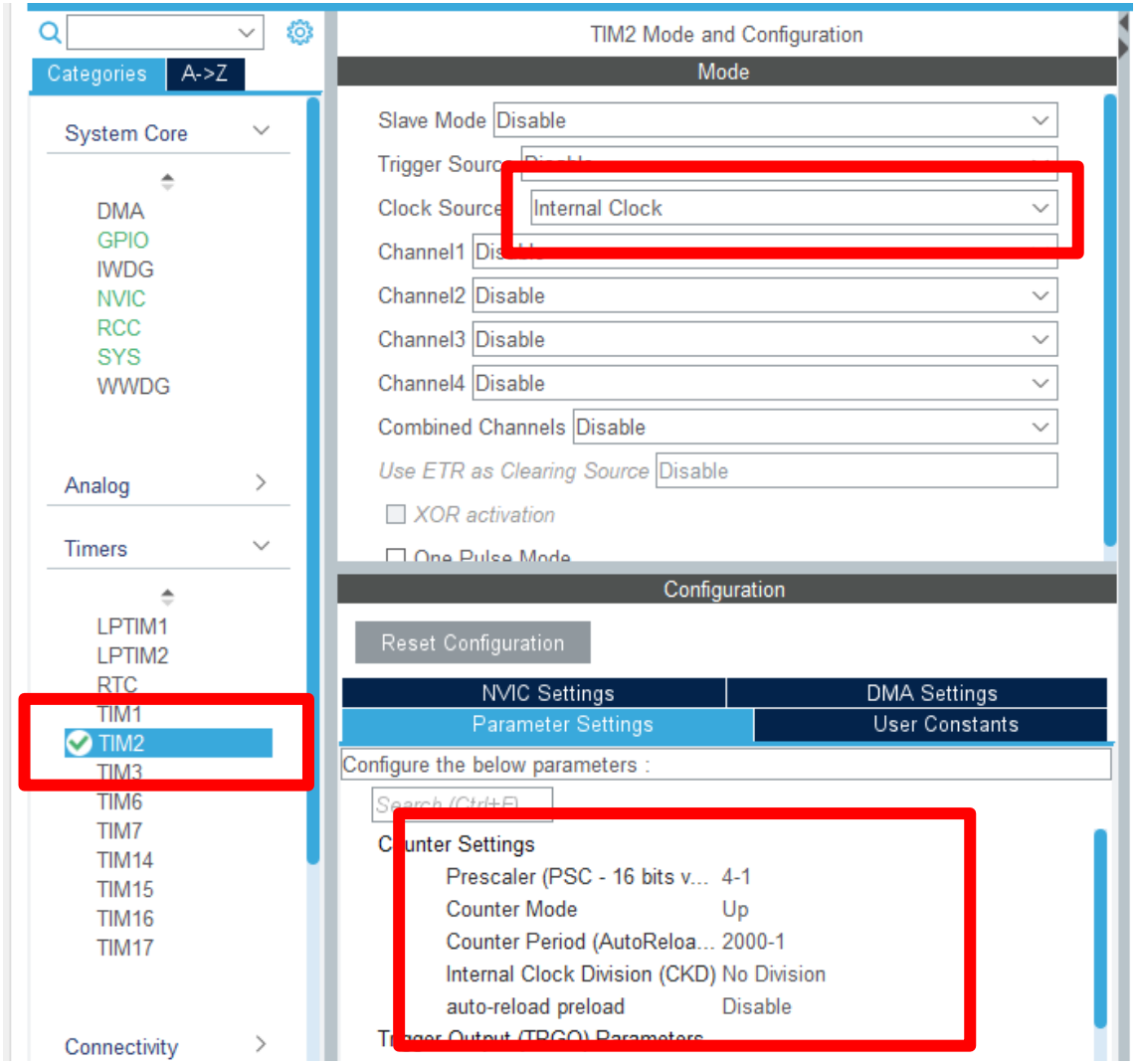
1. setup clock configuration



HCLK = 64MHz, APB timer clocks = 64MHz, APB peripheral clocks = 64MHz

Cube MX setup

2. setup TIM2 peripheral



Parameter TIM2

Clock Source : internal clock’

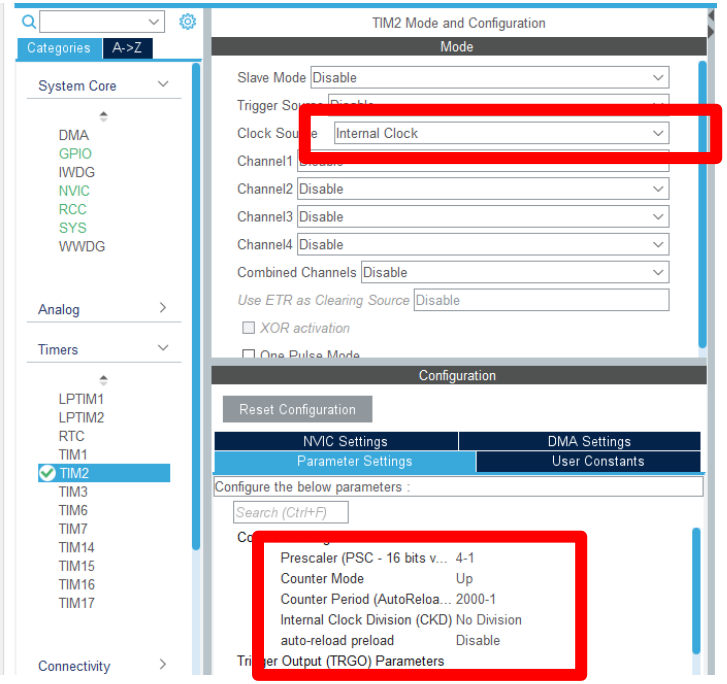
Prescaler : 4 – 1

Counter Period : 2000 - 1

Cube MX setup

2.1 parameter

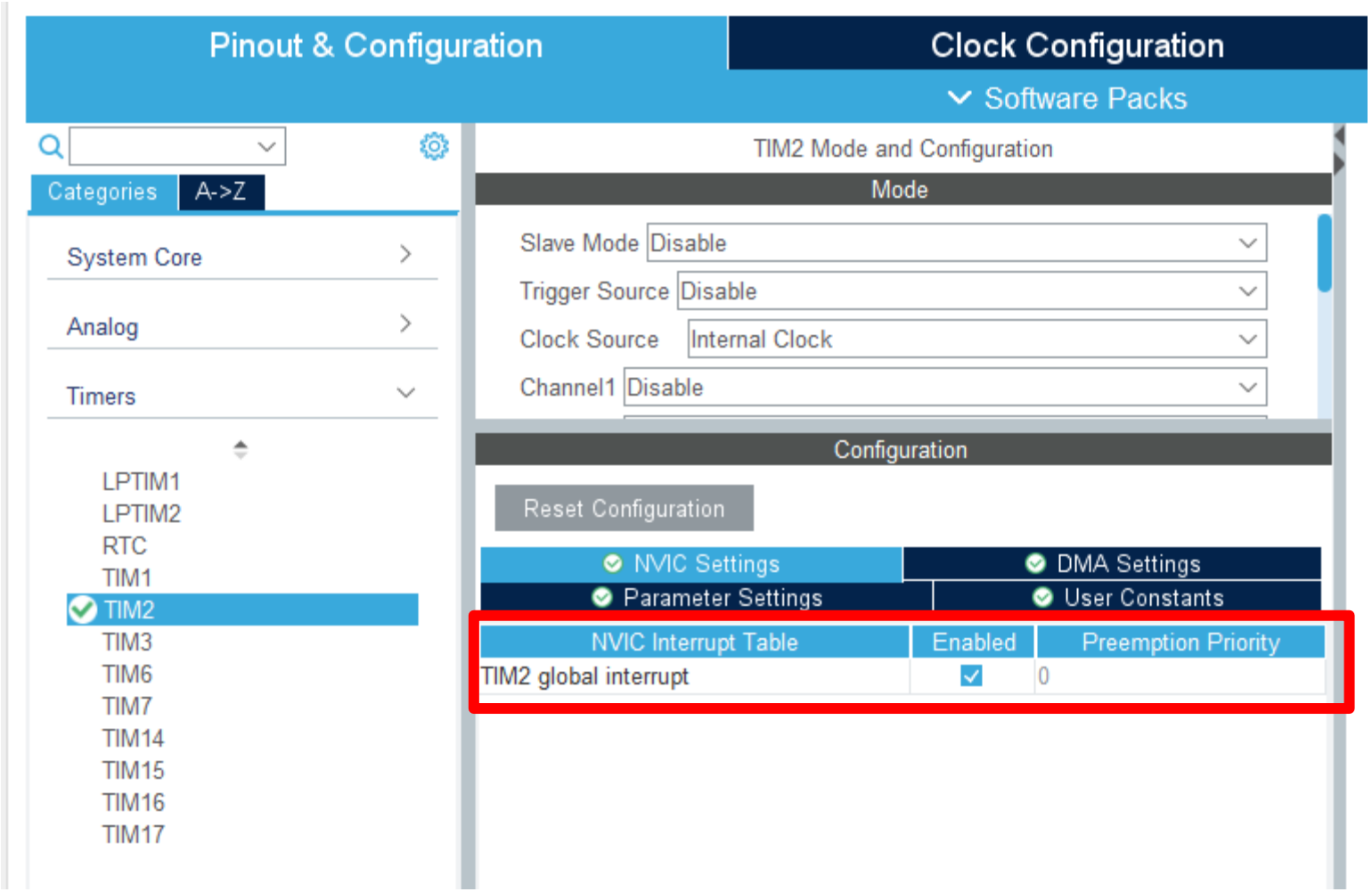
- Register ที่ตั้งค่าต้องใส่ค่า -1 เสมอ เนื่องจาก utilization of designed
- เช่น หากต้องการหารความถี่ด้วย 6 ต้องตั้งค่า PSC ให้เป็น 5 หรือสามารถใส่ 6 – 1 ในโปรแกรมได้เลย



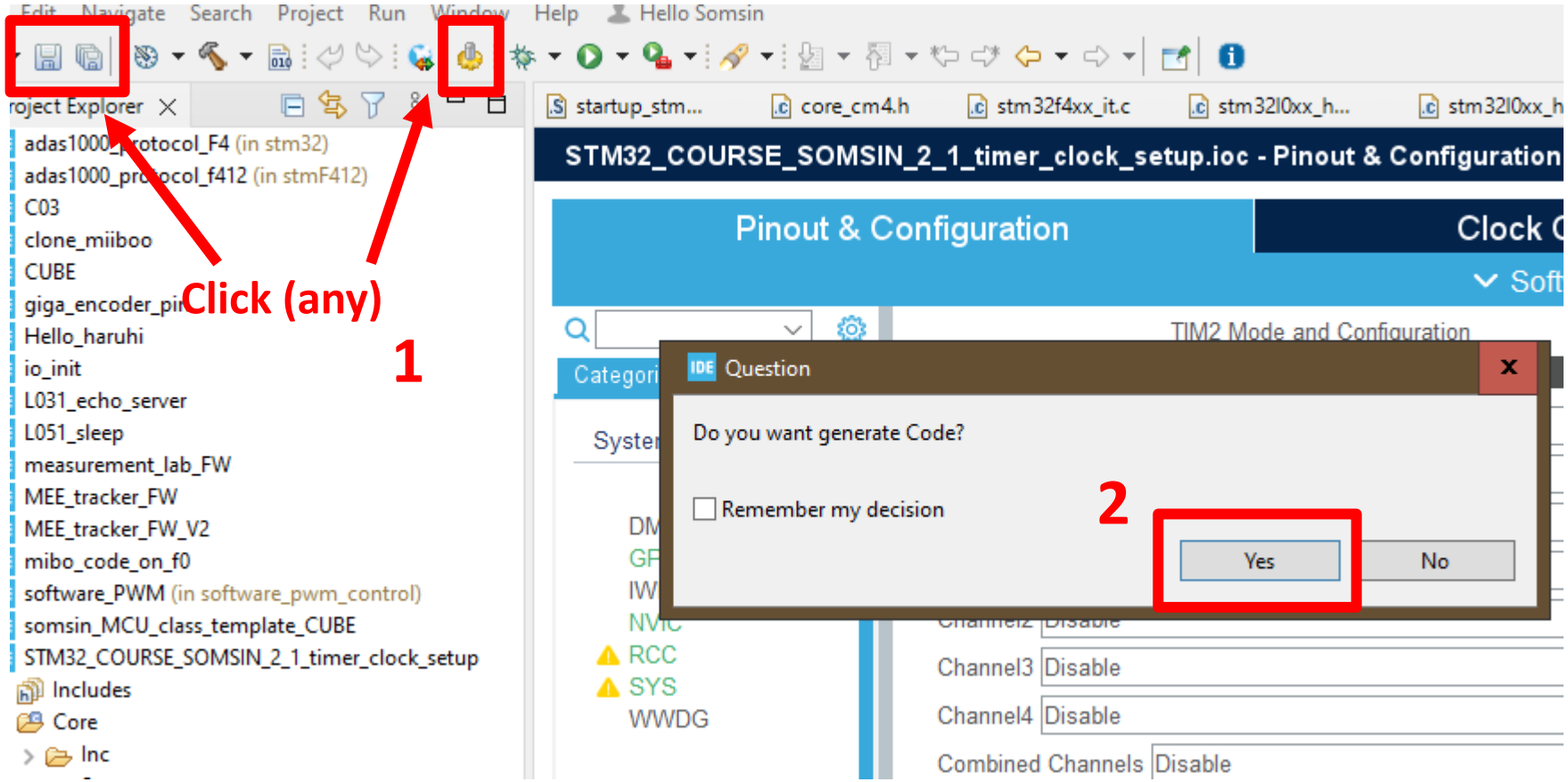
*- เนื่องจาก register นั้นมีข้อจำกัดคือต้องมีค่าตั้งแต่ 0 – max เท่านั้น เช่น register 8 bit จะมีค่าเป็นได้แค่เพียง 0 – 255 ดังนั้น ผู้พัฒนาต้องการให้ MCU มีความสามารถในการ ตั้งค่าได้ตั้งแต่ 1 – 256 จึงให้ ค่า 0 หมายความว่า 1 , 1 หมายความว่า 2 ,..... เป็นเหตุให้ต้องใส่ค่า – 1 เสมอ

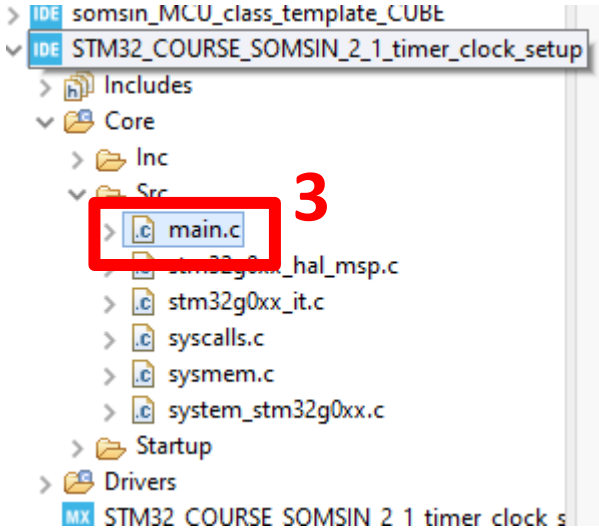
Cube MX setup

3. setup Interrupt



Cube MX setup





Cube MX setup

4. coding

4.1 start timer IT at section 2

```
 96
 97  /* USER CODE BEGIN 2 */
 98  HAL_TIM_Base_Start_IT(&htim2);
 99
100  /* USER CODE END 2 */
101
```

```
HAL_TIM_Base_Start_IT(&htim2);
```

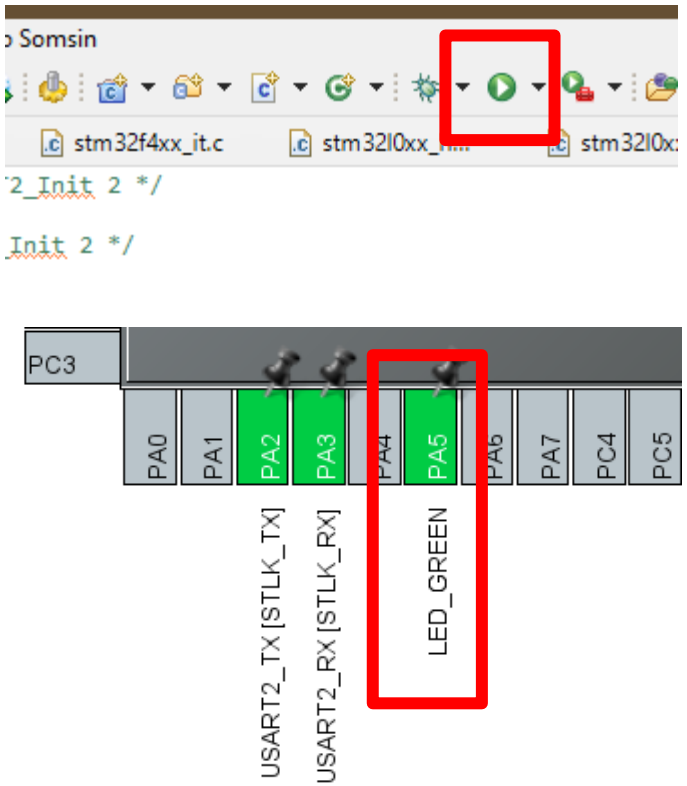
4.2 add timer callback function at section 4

```
293
294  /* USER CODE BEGIN 4 */
295  void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
296  {
297    if (htim == &htim2 )
298    {
299      HAL_GPIO_TogglePin(LED_GREEN_GPIO_Port, LED_GREEN_Pin);
300    }
301  }
302  /* USER CODE END 4 */
```

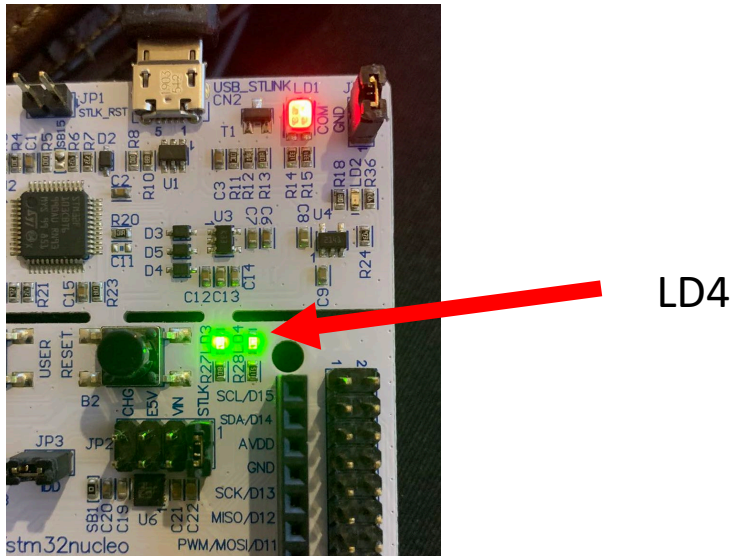
```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
     if (htim == &htim2 )
     {
     HAL_GPIO_TogglePin(LED_GREEN_GPIO_Port, LED_GREEN_Pin);
     }
}
```

Cube MX setup

Run and measure



ให้สังเกตความถี่ที่ LED กระพริบ นับว่าใน 1 วินาที LD4 กระพริบกี่ครั้ง

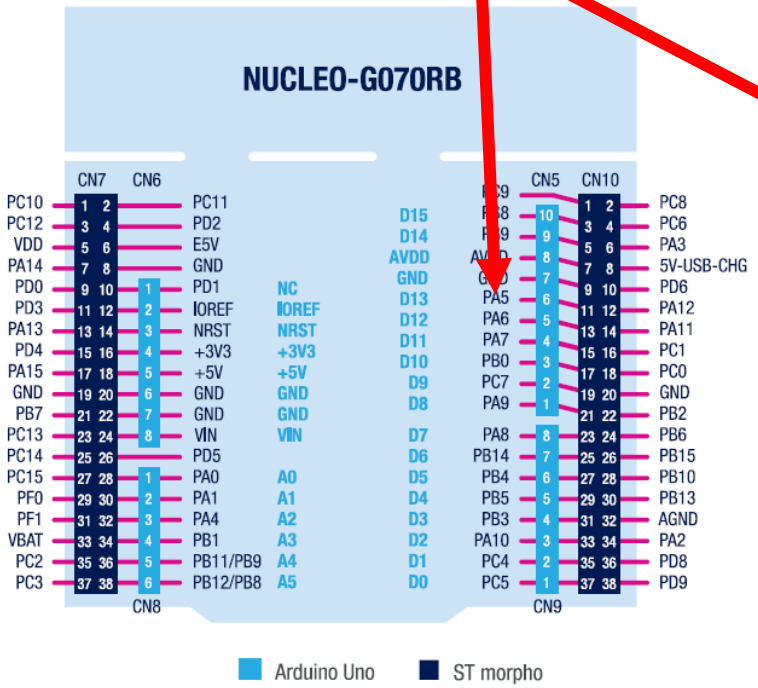


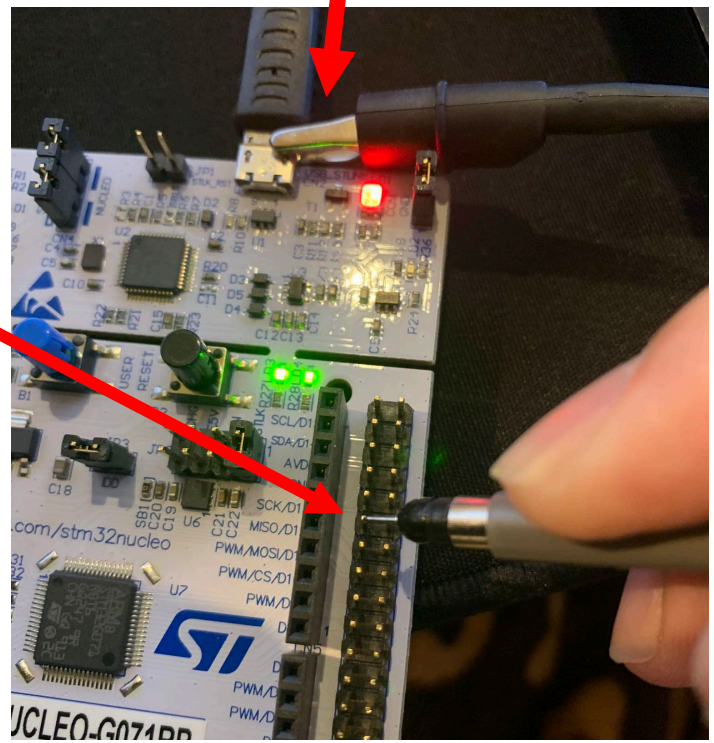
ล้อเล่นนะครับ ต้องใช้เครื่องช่วยวัดครับ นั้นก็คือ oscilloscope

oscilloscope

* Don’t forget to tap GND (any GND on Board)

LD4 = PA5

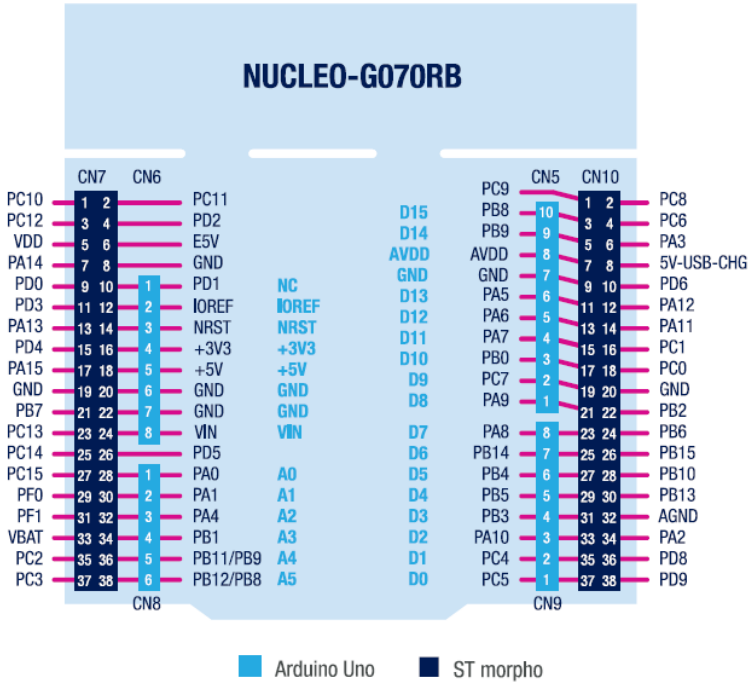


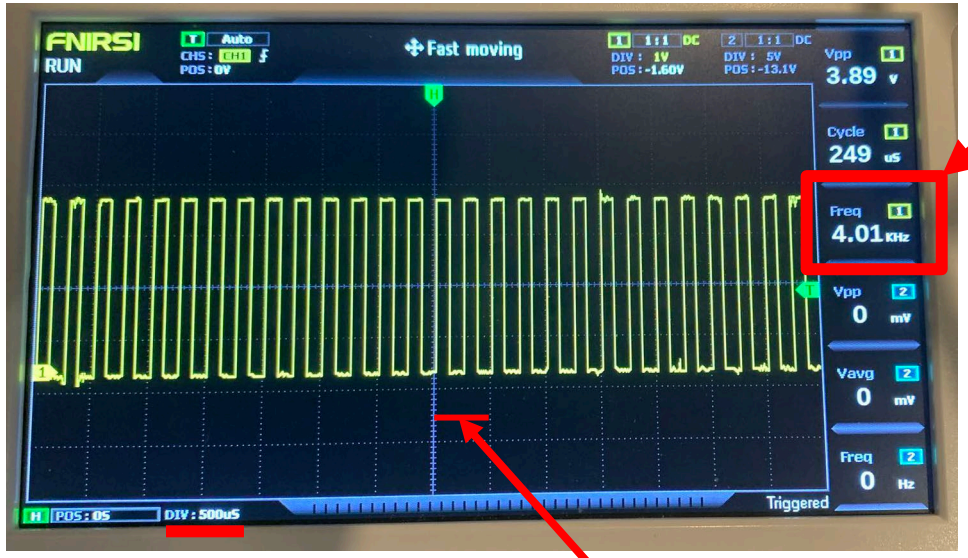


* Taping GND like this picture are highly NOT recommend in further study (but instructor is too lazy to do in proper way)

oscilloscope

LD4 = PA5





Frequency

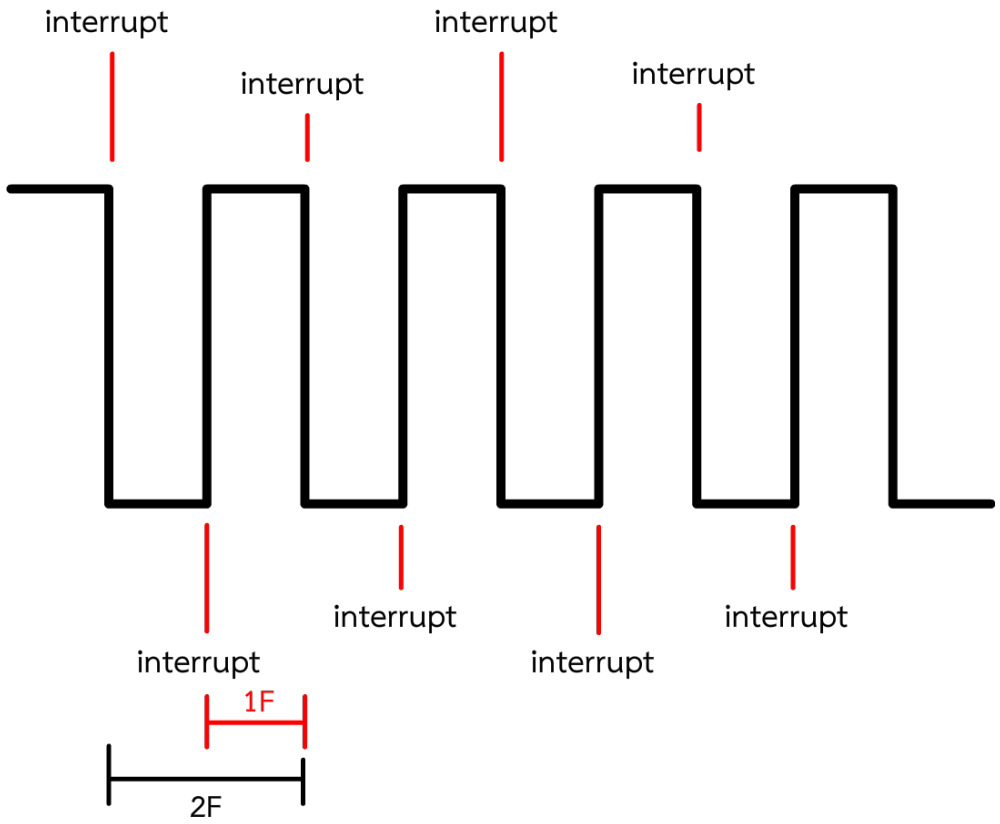
500 uS

2 cycle in 1 DIV

T = 500 / 2 = 250uS (4000 Hz)

* Instructor don’t have enough money to afford better oscilloscope so value may diff a bit

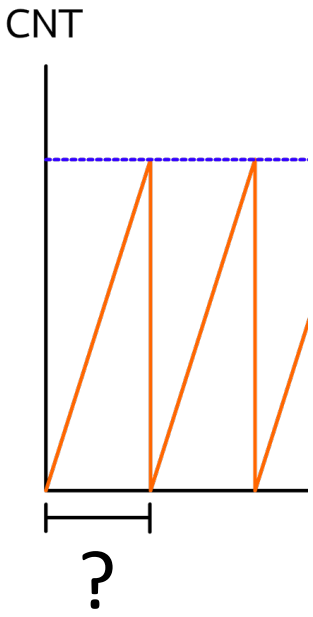
WHY F/2

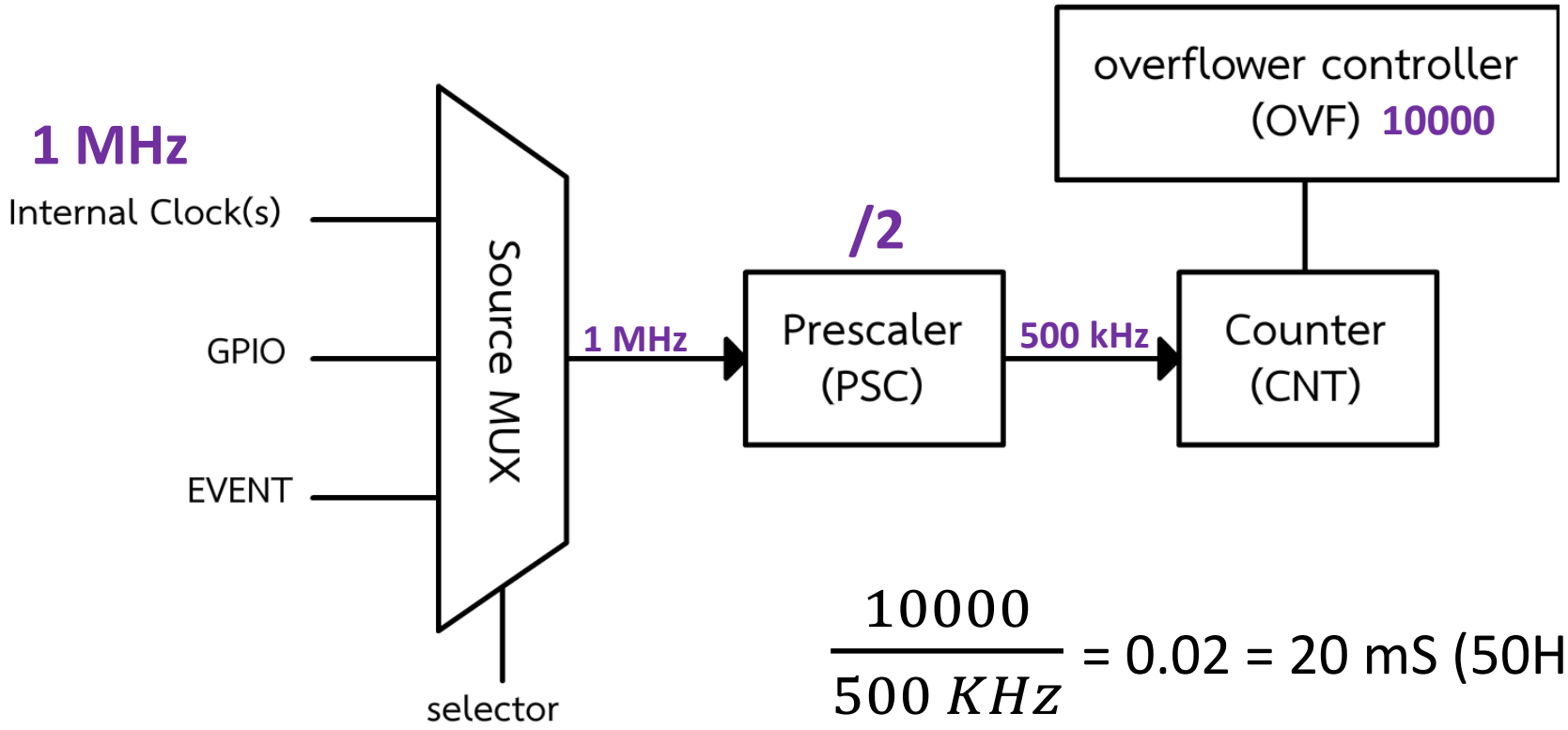


โปรแกรมต้องทำงาน 2 ครั้ง ถึงจะได้ Square wave 1 ลูกคลื่น oscilloscope เลยวัดได้ความถี่ / 2

Example2

- clock source = PPL clock 1MHz , PSC = 2 , CNT = 900, OVF = 10000



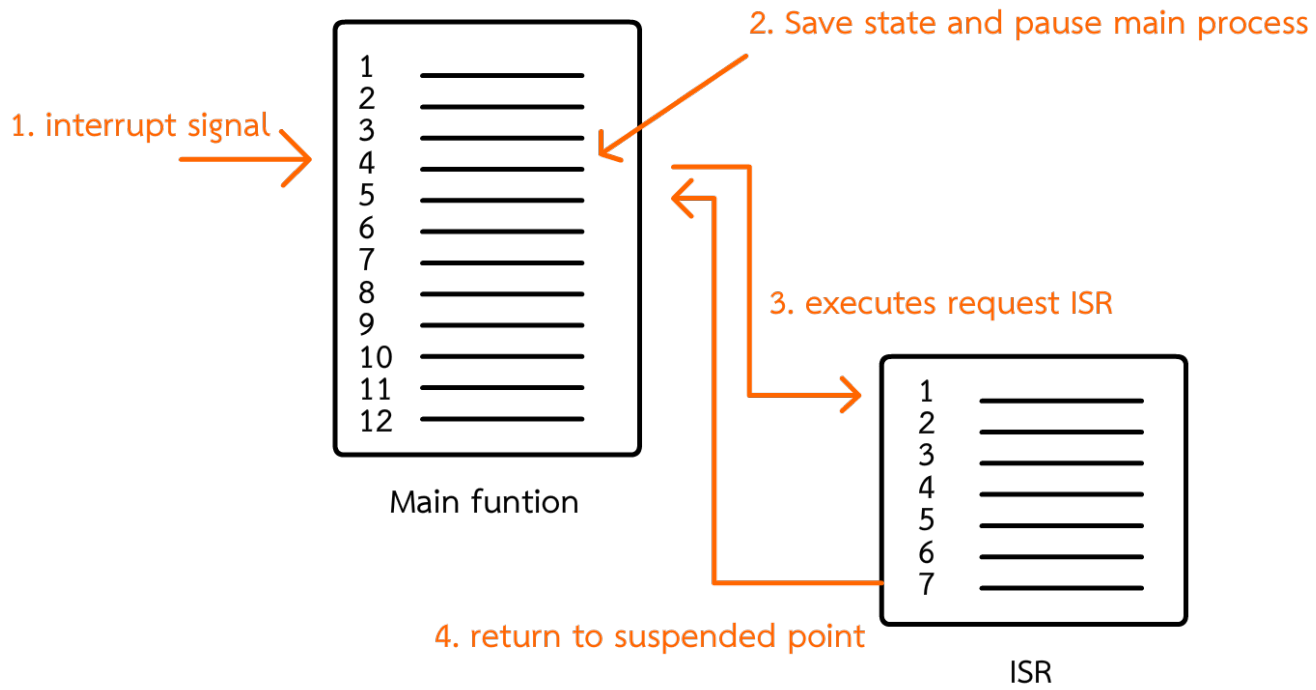


$$\frac{10000}{500\ KHz} = 0.02 = 20\ mS\ (50Hz)$$

Quiz

Interrupt

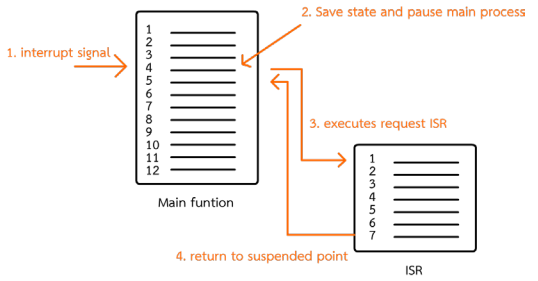
Interrupt คือ event (เหตุการณ์) หรือ signal ที่ส่งเข้ามายังระบบเพื่อ บอกให้ระบบหยุดทำงานหลัก (main function) และเตรียมตัวไปทำงานอื่นที่เตรียมไว้ (ISR : interrupt service routine)



Interrupt สามารถทำให้เกิดได้จากหลายเหตุการณ์ เช่น timer IO ADC หรือแม้กระทั้ง software

ถ้าเกิดขึ้นพร้อมกันจะเป็นอย่างไร ทับกัน? run แค่อันเดียว? รออีก 3 คาบ นะครับ

Interrupt



Timer ISR

```
293
294 /* USER CODE BEGIN 4 */
295 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
296 {
297   if (htim == &htim2 )
298   {
299     HAL_GPIO_TogglePin(LED_GREEN_GPIO_Port, LED_GREEN_Pin);
300   }
301 }
302 /* USER CODE END 4 */
```

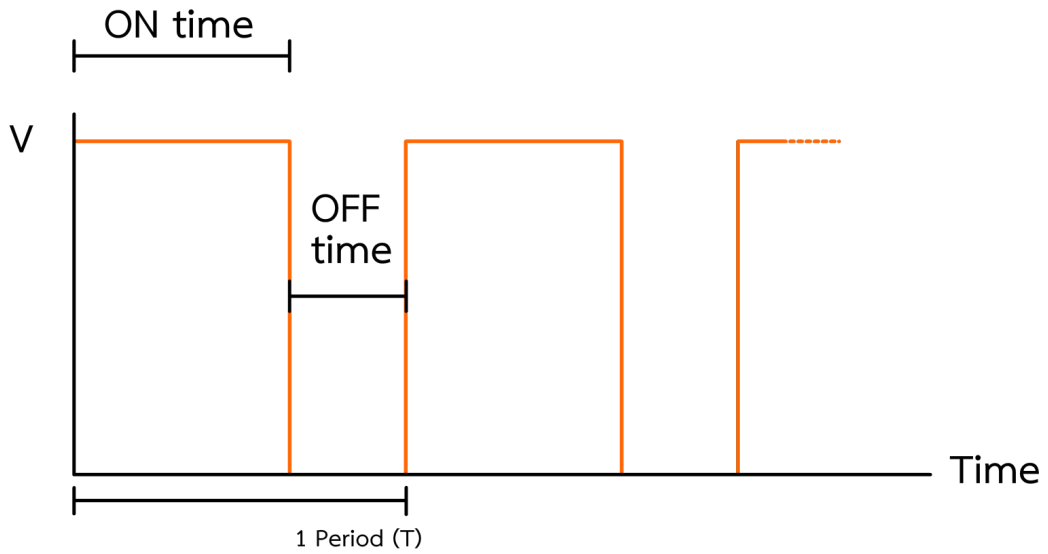
Timer application

- timer (counter)
- RTC (real time clock)
- Periodic interrupt
- pulse counter
- waveform generator
- generate event for another peripheral

Waveform generator

- Generate square wave with specific shape (frequency , duty cycle, phase)
- Our scope -> frequency , duty cycle
- Output compare Mode
- PWM generation Mode

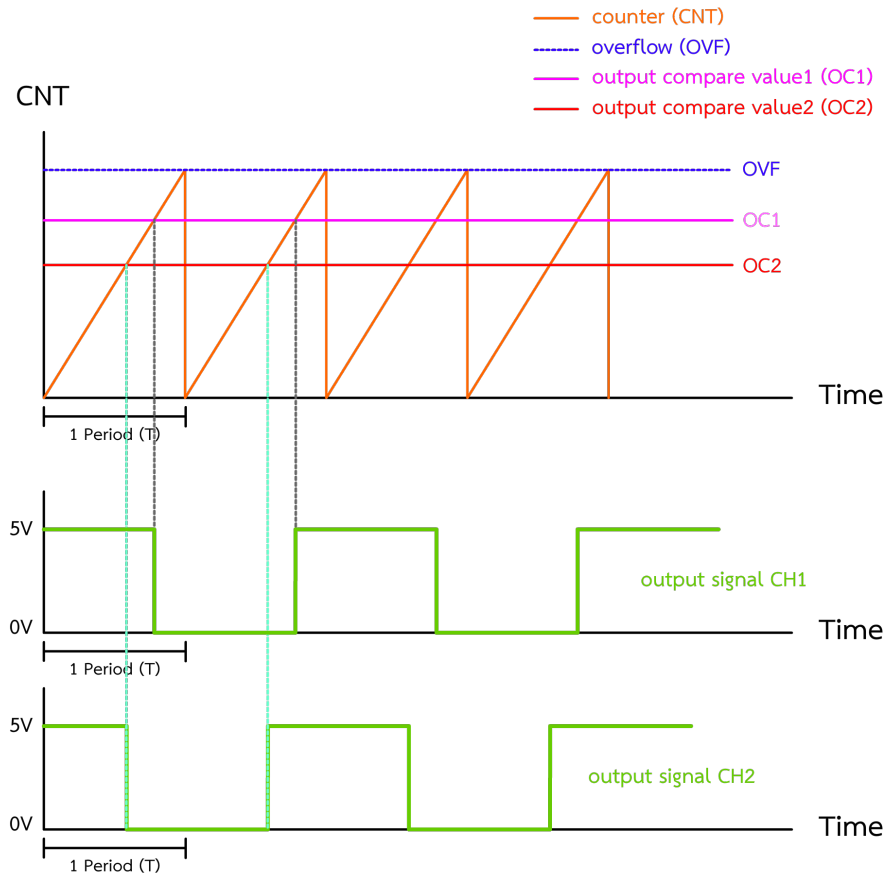
$$Duty\ Cycle\ = \frac{T_{on}}{Period}$$

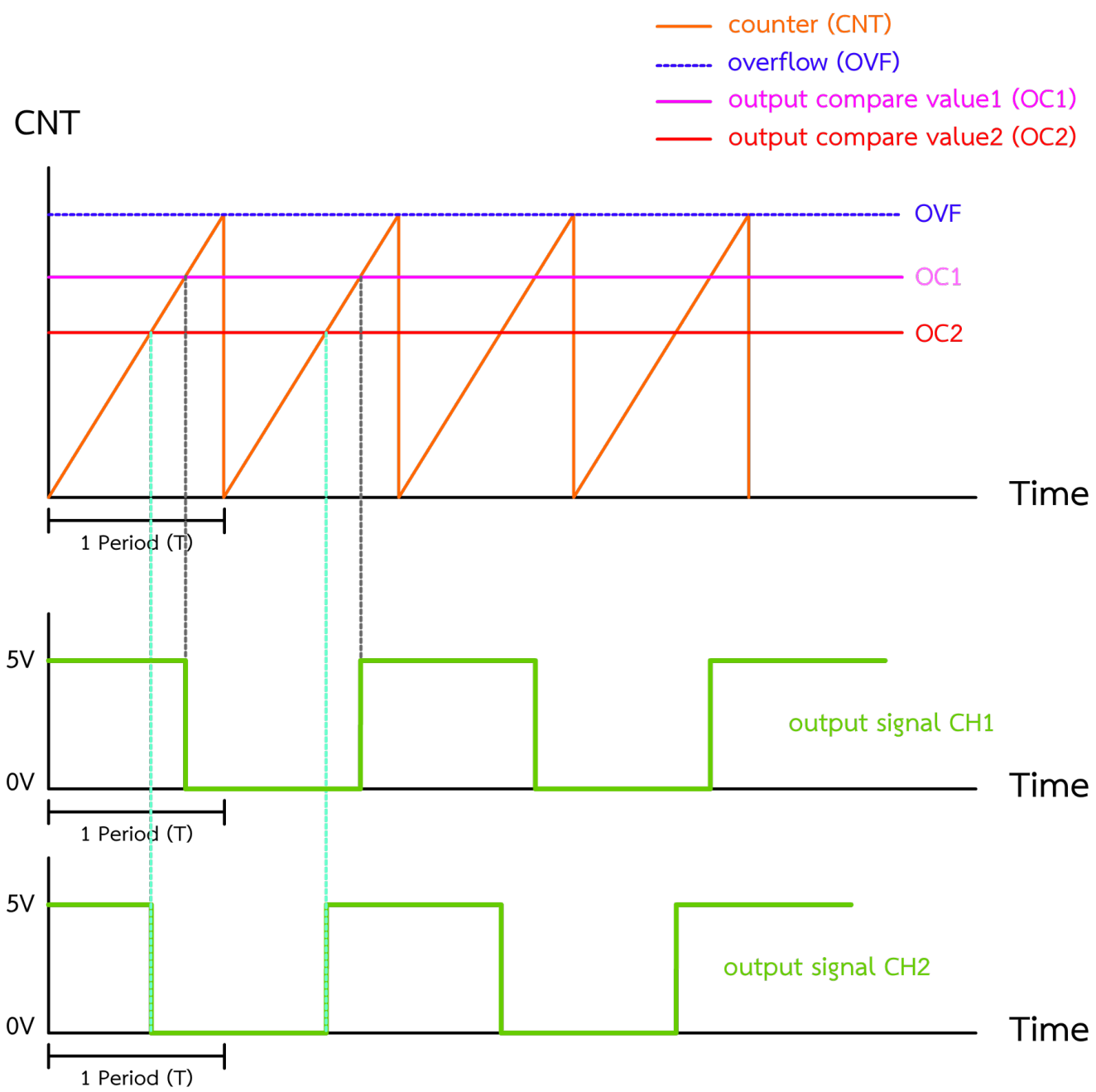


Output compare Mode

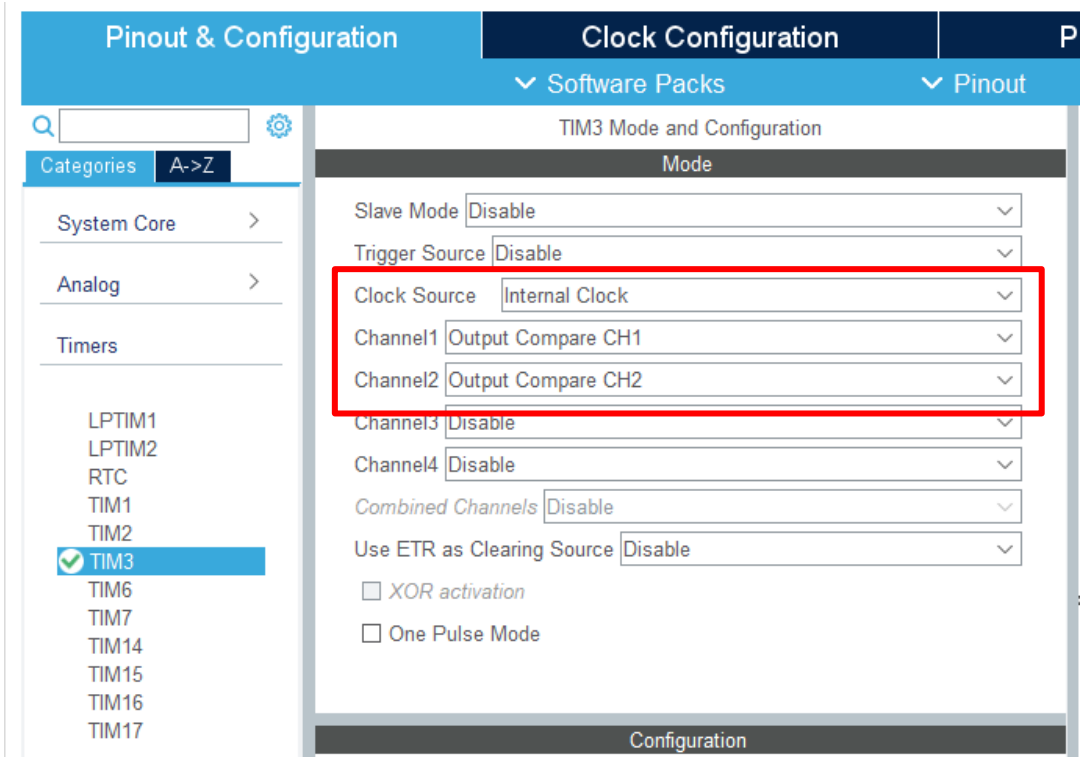
Generate event can be used to toggle signal

- When OC match CNT signal will toggle
- Can enable multi channel simultaneously

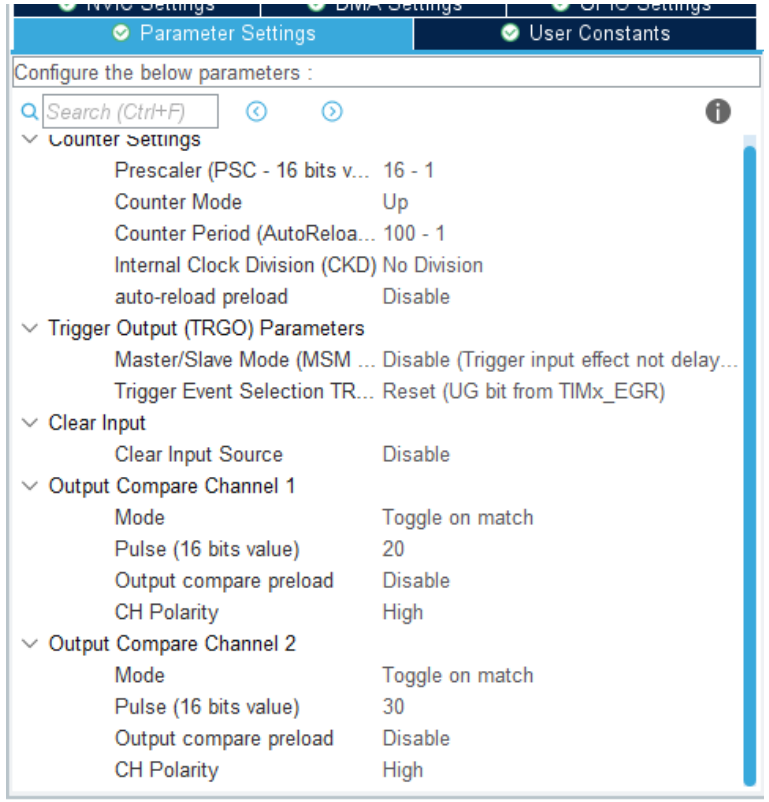



counter (CNT)
overflow (OVF)
output compare value1 (OC1)
output compare value2 (OC2)
CNT
OVF
OC1
OC2
Time
1 Period (T)
5V
0V
output signal CH1
Time
1 Period (T)
5V
0V
output signal CH2
Time
1 Period (T)

Output compare Mode Setup

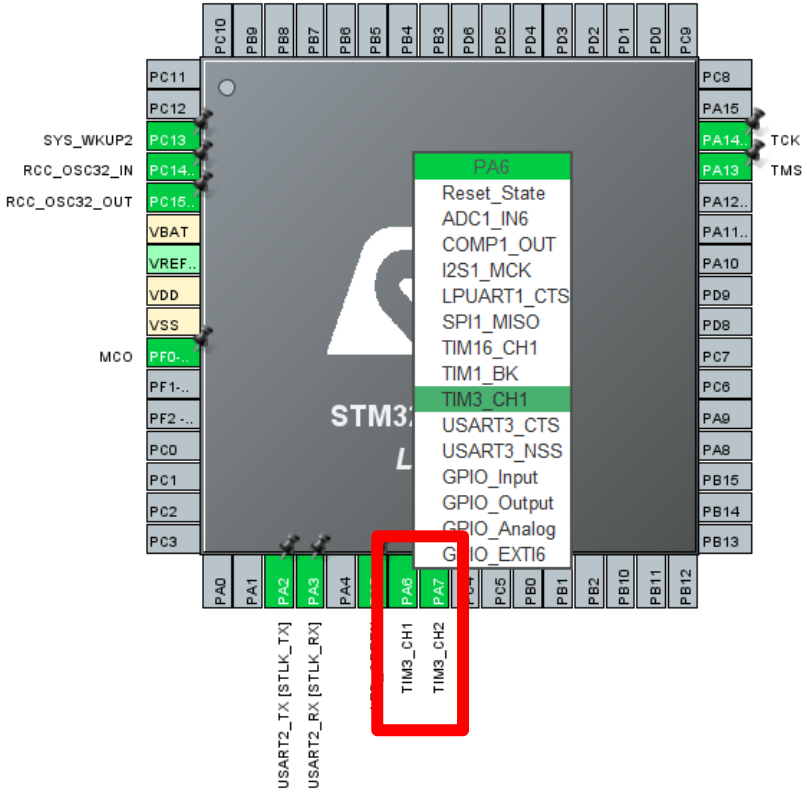


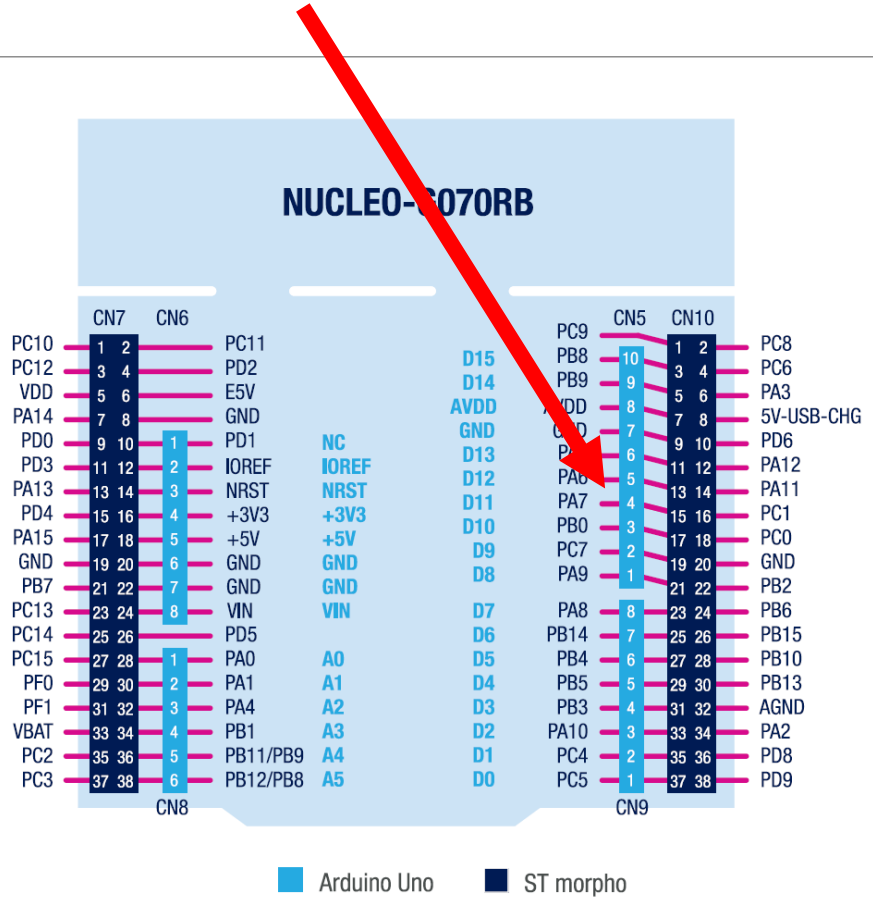
Enable TIM3 CH1, CH2 and Clock Source



Config Parameter

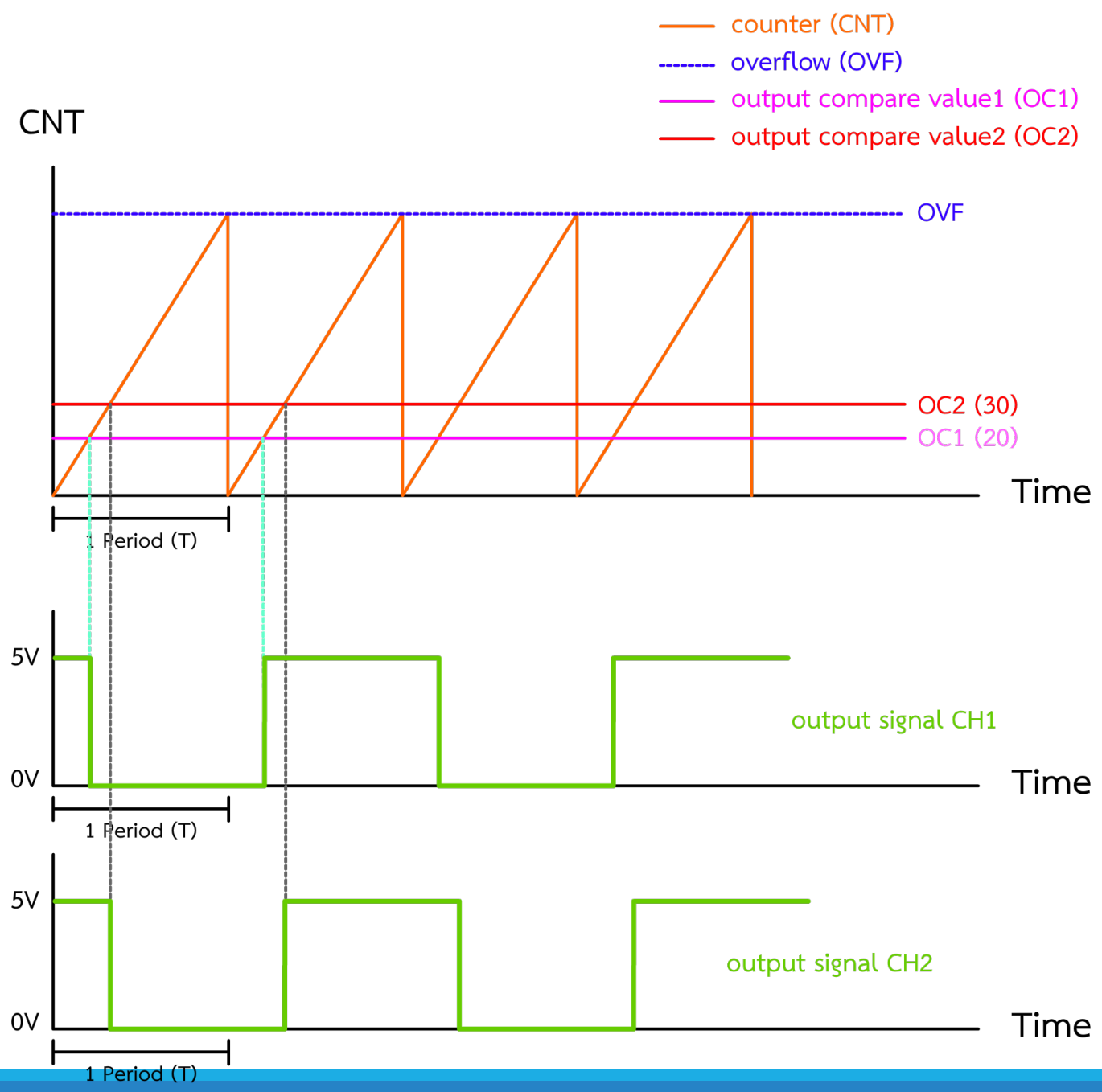
Config Pin

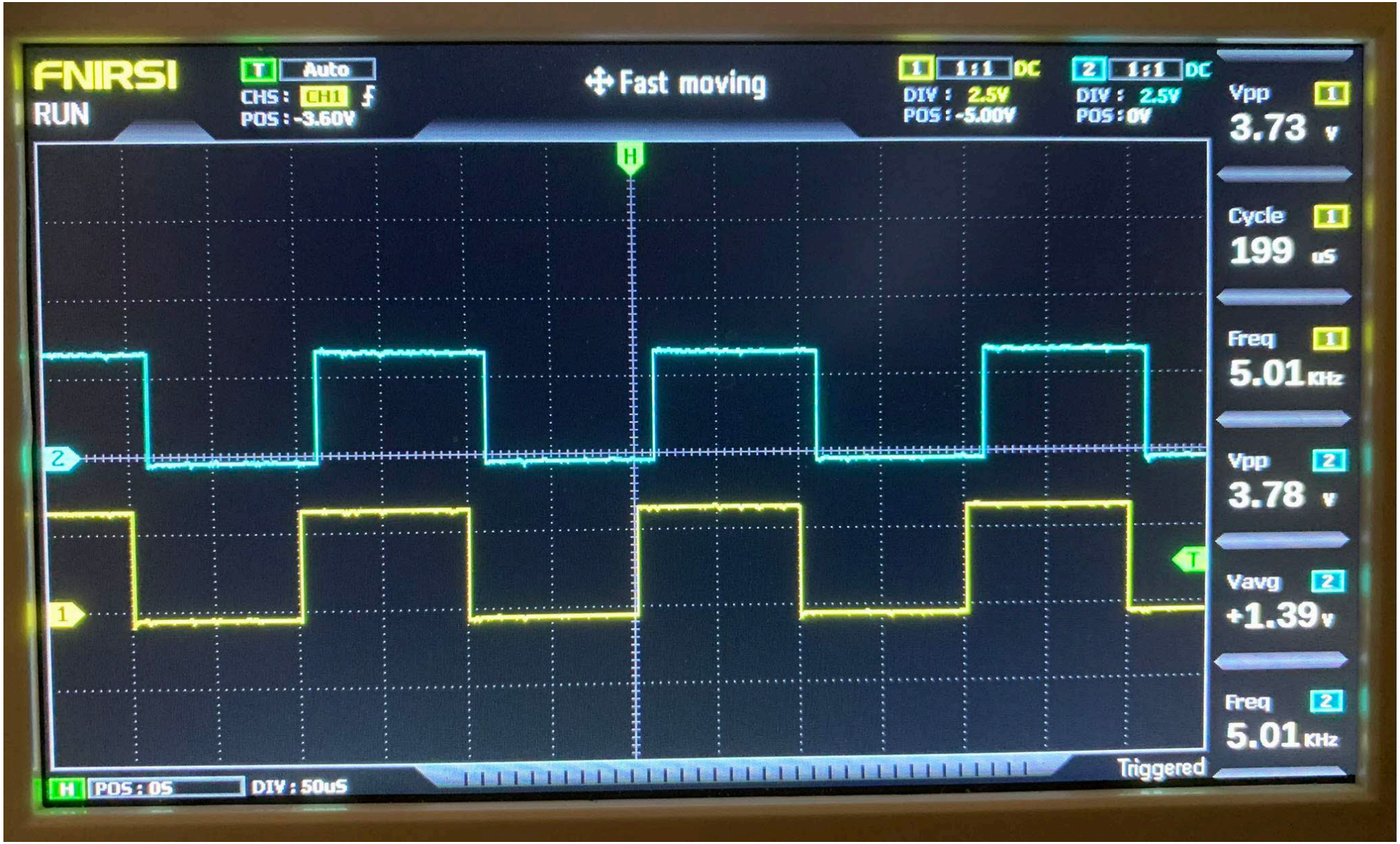

PA6
Reset_State
ADC1_IN6
COMP1_OUT
I2S1_MCK
LPUART1_CTS
SPI1_MISO
TIM16_CH1
TIM1_BK
TIM3_CH1
USART3_CTS
USART3_NSS
GPIO_Input
GPIO_Output
GPIO_Analog
GPIO_EXTI6
SYS_WKUP2
RCC_OSC32_IN
RCC_OSC32_OUT
MCO
TCK
TMS
USART2_TX [STLK_TX]
USART2_RX [STLK_RX]
TIM3_CH1
TIM3_CH2


CN7
CN6
CN5
CN10
CN8
CN9
PC10
PC12
VDD
PA14
PD0
PD3
PA13
PD4
PA15
GND
PB7
PC13
PC14
PC15
PF0
PF1
VBAT
PC2
PC3
PC11
PD2
E5V
GND
PD1
IOREF
NRST
+3V3
+5V
GND
GND
VIN
PD5
PA0
PA1
PA4
PB1
PB11/PB9
PB12/PB8
NC
IOREF
NRST
+3V3
+5V
GND
GND
VIN
A0
A1
A2
A3
A4
A5
D15
D14
AVDD
GND
D13
D12
D11
D10
D9
D8
D7
D6
D5
D4
D3
D2
D1
D0
PC9
PB8
PB9
PA7
PB0
PC7
PA9
PA8
PB14
PB4
PB5
PB3
PA10
PC4
PC5
PC8
PC6
PA3
5V-USB-CHG
PD6
PA12
PA11
PC1
PC0
GND
PB2
PB6
PB15
PB10
PB13
AGND
PA2
PD8
PD9
Arduino Uno
ST morpho

Coding

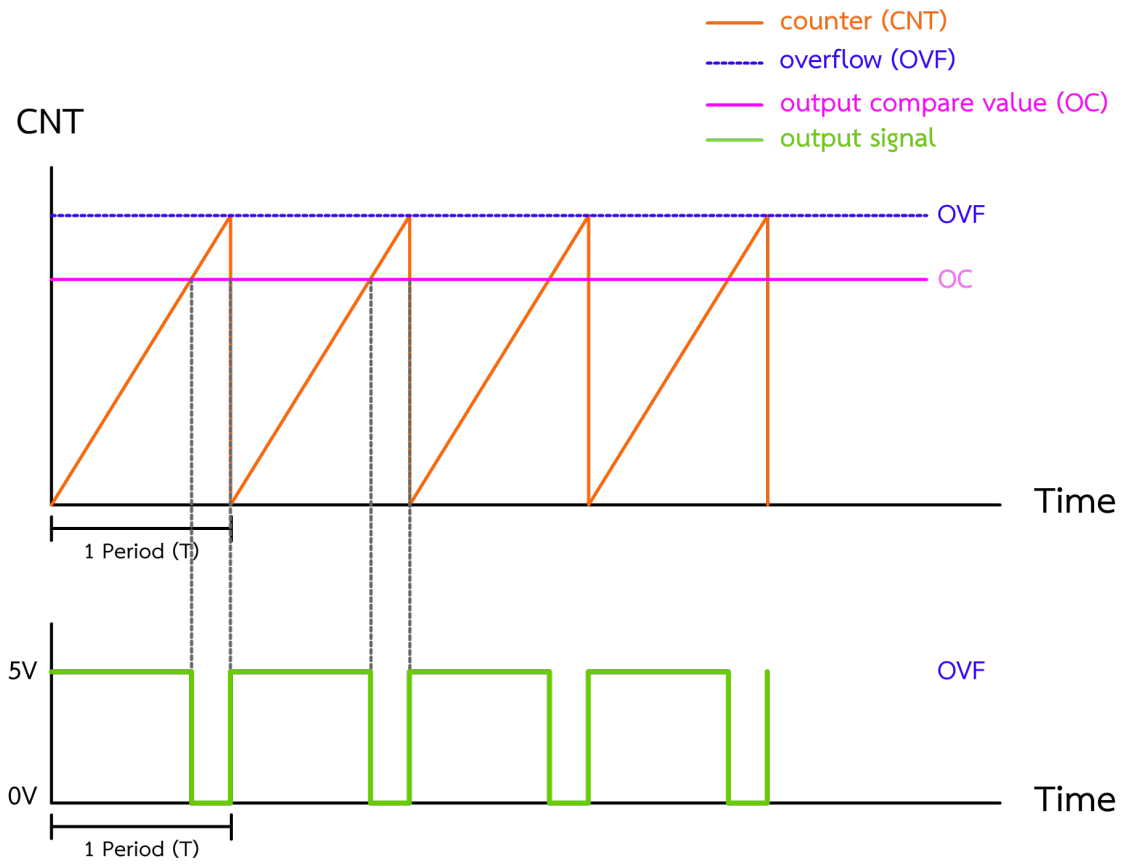
```
92    /* Initialize all configured peripherals */
93    MX_GPIO_Init();
94    MX_USART2_UART_Init();
95    MX_TIM3_Init();
96    /* USER CODE BEGIN 2 */
97    HAL_TIM_OC_Start(&htim3, TIM_CHANNEL_1);
98    HAL_TIM_OC_Start(&htim3, TIM_CHANNEL_2);
99    /* USER CODE END 2 */
100
101   /* Infinite loop */
102   /* USER CODE BEGIN WHILE */
103   while (1)
104   {
105     /* USER CODE END WHILE */
106
```


counter (CNT)
overflow (OVF)
output compare value1 (OC1)
output compare value2 (OC2)
CNT
OVF
OC2 (30)
OC1 (20)
Time
1 Period (T)
5V
0V
output signal CH1
Time
1 Period (T)
5V
0V
output signal CH2
Time
1 Period (T)

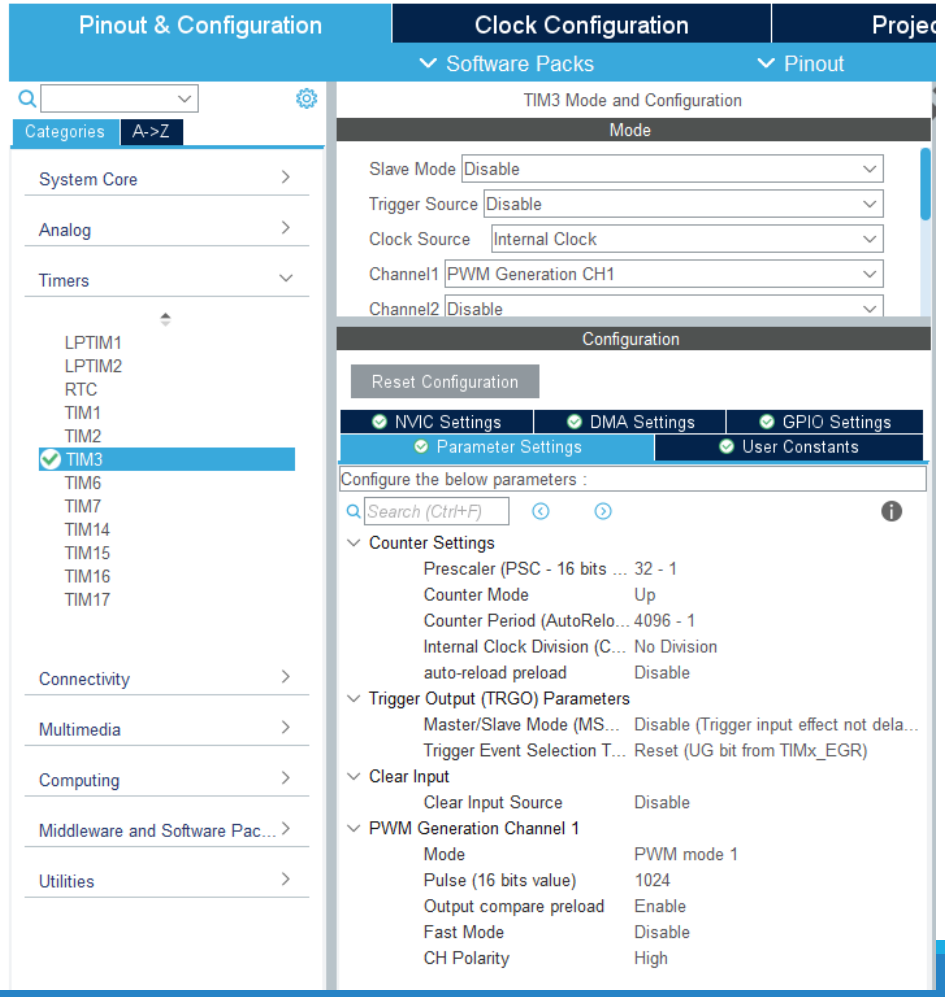

FNIRSI
RUN
T Auto
CHS: CH1
POS: -3.60V
Fast moving
1 1:1 DC
DIV: 2.5V
POS: -5.00V
2 1:1 DC
DIV: 2.5V
POS: 0V
Vpp 1
3.73 V
Cycle 1
199 uS
Freq 1
5.01 KHz
Vpp 2
3.78 V
Vavg 2
+1.39V
Freq 2
5.01 KHz
H POS: 0S
DIV: 50uS
Triggered

PWM generation Mode

- Output compare indicate what output logic
- If CNT < OC : output = high
- ELSE : output = low



Example (Output compare Mode)

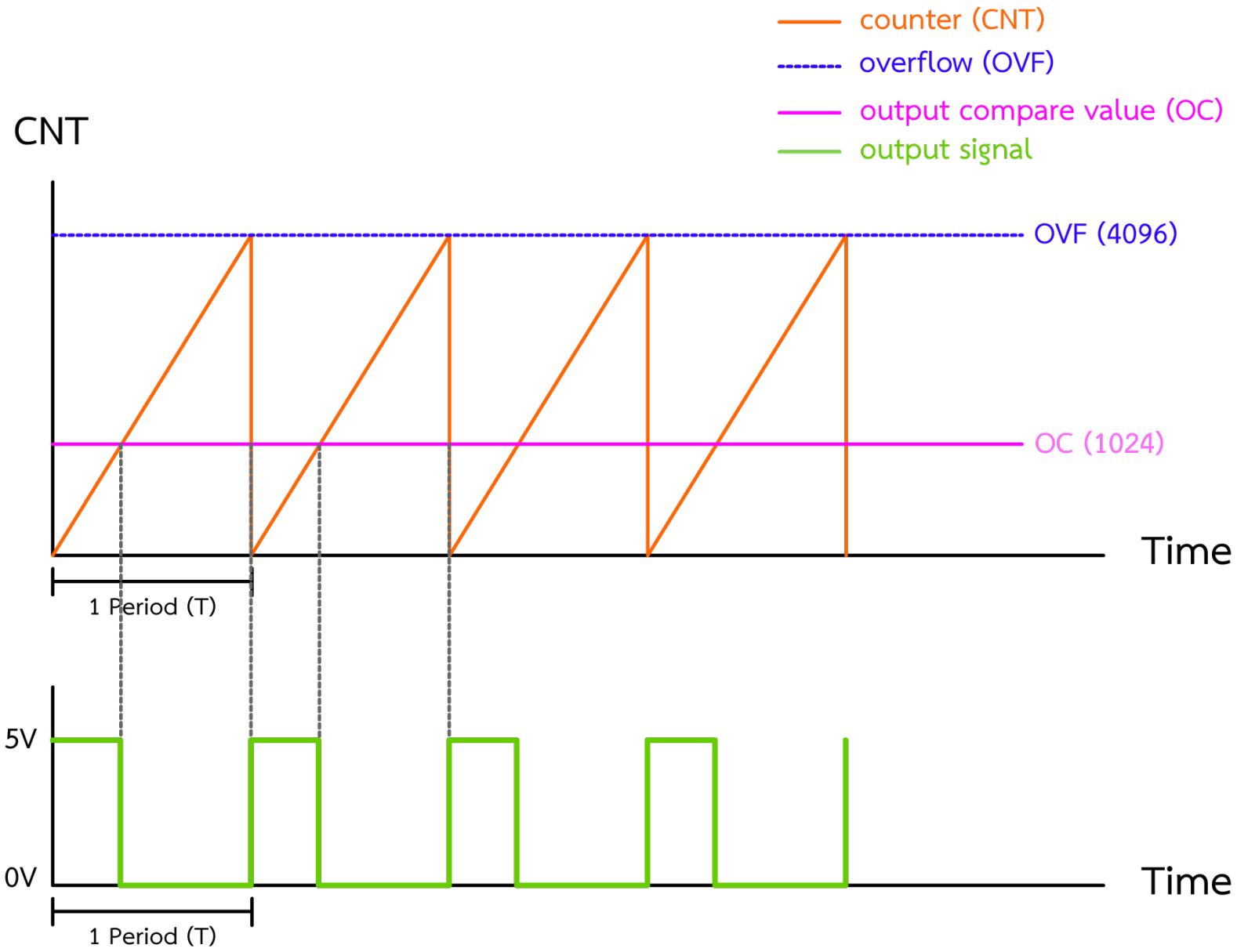


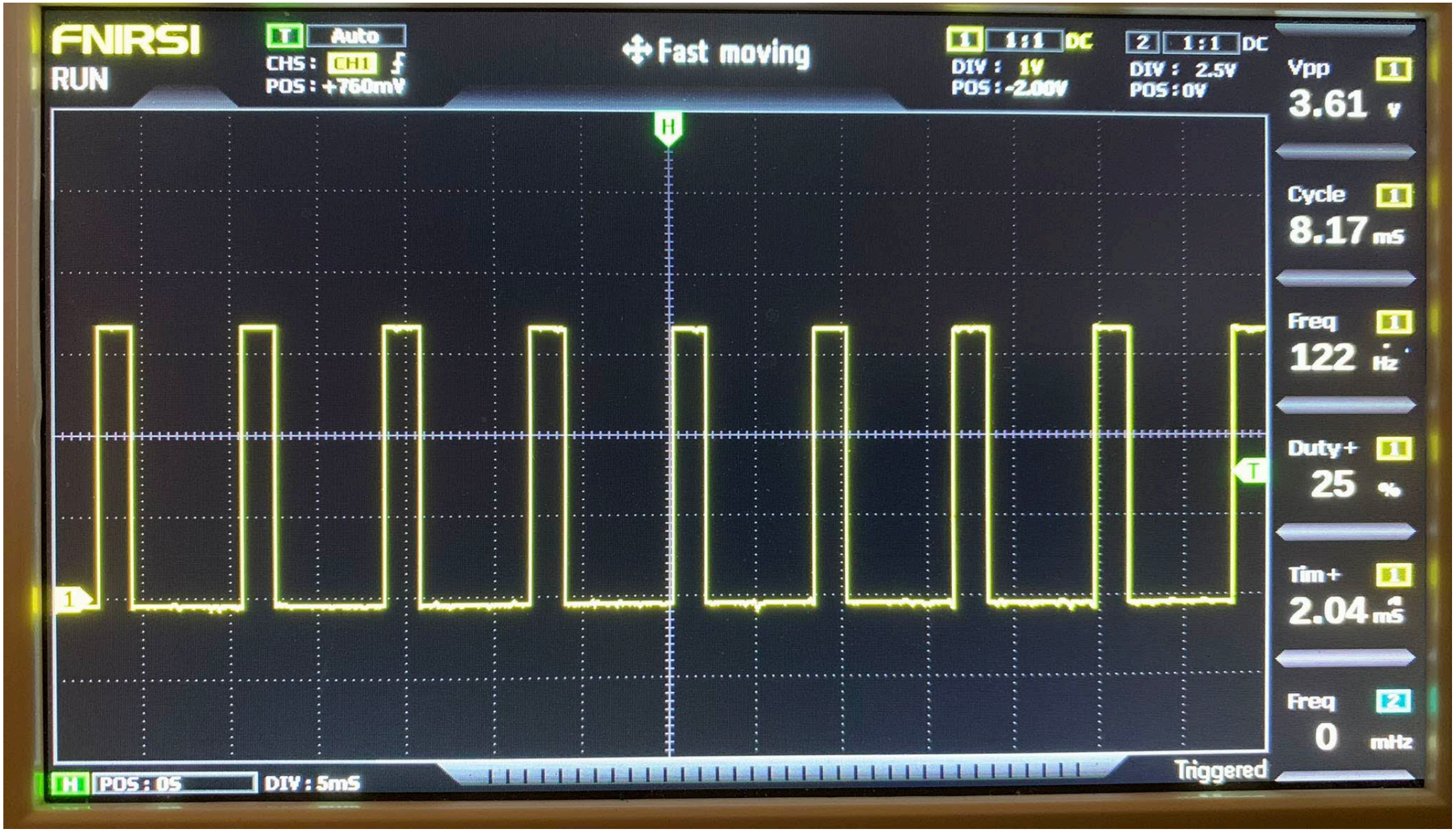
Setup PWM on TIM3 CH1

- OVF -> 4096 (12 bit)
- Pulse (OC) -> 1024 (25%)
- frequency = (16 Mhz / 32) / 4096 = 122.07Hz

Coding

```
90    /* USER CODE END SysInit */
91
92    /* Initialize all configured peripherals */
93    MX_GPIO_Init();
94    MX_USART2_UART_Init();
95    MX_TIM3_Init();
96    /* USER CODE BEGIN 2 */
97    HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_1);
98    /* USER CODE END 2 */
99
100   /* Infinite loop */
101   /* USER CODE BEGIN WHILE */
102   while (1)
103   {
104     /* USER CODE END WHILE */
```


counter (CNT)
overflow (OVF)
output compare value (OC)
output signal
CNT
OVF (4096)
OC (1024)
Time
1 Period (T)
5V
0V
Time
1 Period (T)


FNIRSI
RUN
Auto
CHS: CH1
POS: +760mV
Fast moving
1 1:1 DC
DIV: 1V
POS: -2.00V
2 1:1 DC
DIV: 2.5V
POS: 0V
Vpp 1
3.61 V
Cycle 1
8.17 mS
Freq 1
122 Hz
Duty+ 1
25 %
Tim+ 1
2.04 mS
Freq F1
0 mHz
H
POS: 0S
DIV: 5mS
Triggered

Change OC in run-time ?

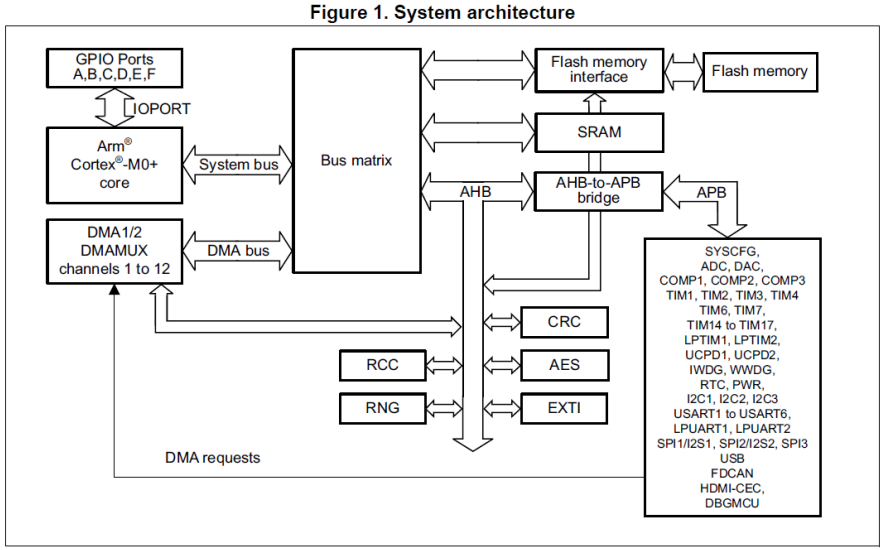
```
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */
      HAL_Delay(2000);
      __HAL_TIM_SET_COMPARE(&htim3,TIM_CHANNEL_1,2048);
      HAL_Delay(2000);
      __HAL_TIM_SET_COMPARE(&htim3,TIM_CHANNEL_1,1024);
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

```
_HAL_TIM_SET_COMPARE(HANDLE,CHANNEL,VALUE);
```

How about main function?

- จากตัวอย่างที่เรียนในคาบนี้ จะเห็นว่า การใช้งาน peripheral นั้น จะอยู่แยกส่วนกับ CPU และ RAM โดยสิ้นเชิง (ถ้าไม่นับว่า Register คือ memory ที่อยู่ใน CPU)

- การใช้งาน peripheral เยอะๆ นั้นส่งผมต่อความแรง (speed and latency) ของ CPU และ RAM หรือไม่?



* หน้า 56 ของ datasheet

Lab

Create PWM timer output 1 channel

- CH1 create ([student_ID] % 1000) Hz square ware with ([student_ID] % 20) percent of duty cycle

- EX student ID = 60601167

- CH1 -> 167Hz with 7% duty cycle

$$DUTY = \frac{OC}{OVF}$$

Lab con.

- Measure output using oscilloscope
- Explain about parameter value show calculation and fill up diagram