Høgskulen på Vestlandet

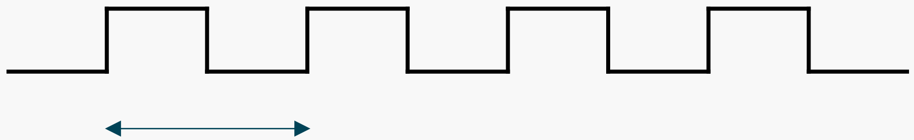
ELE201 Mikrokontrollere og datanett
Mikrokontroller

Klokker, timere og timer-interrupt

Eivind Vågslid Skjæveland
esk@hvl.no

Klokker i datasystem

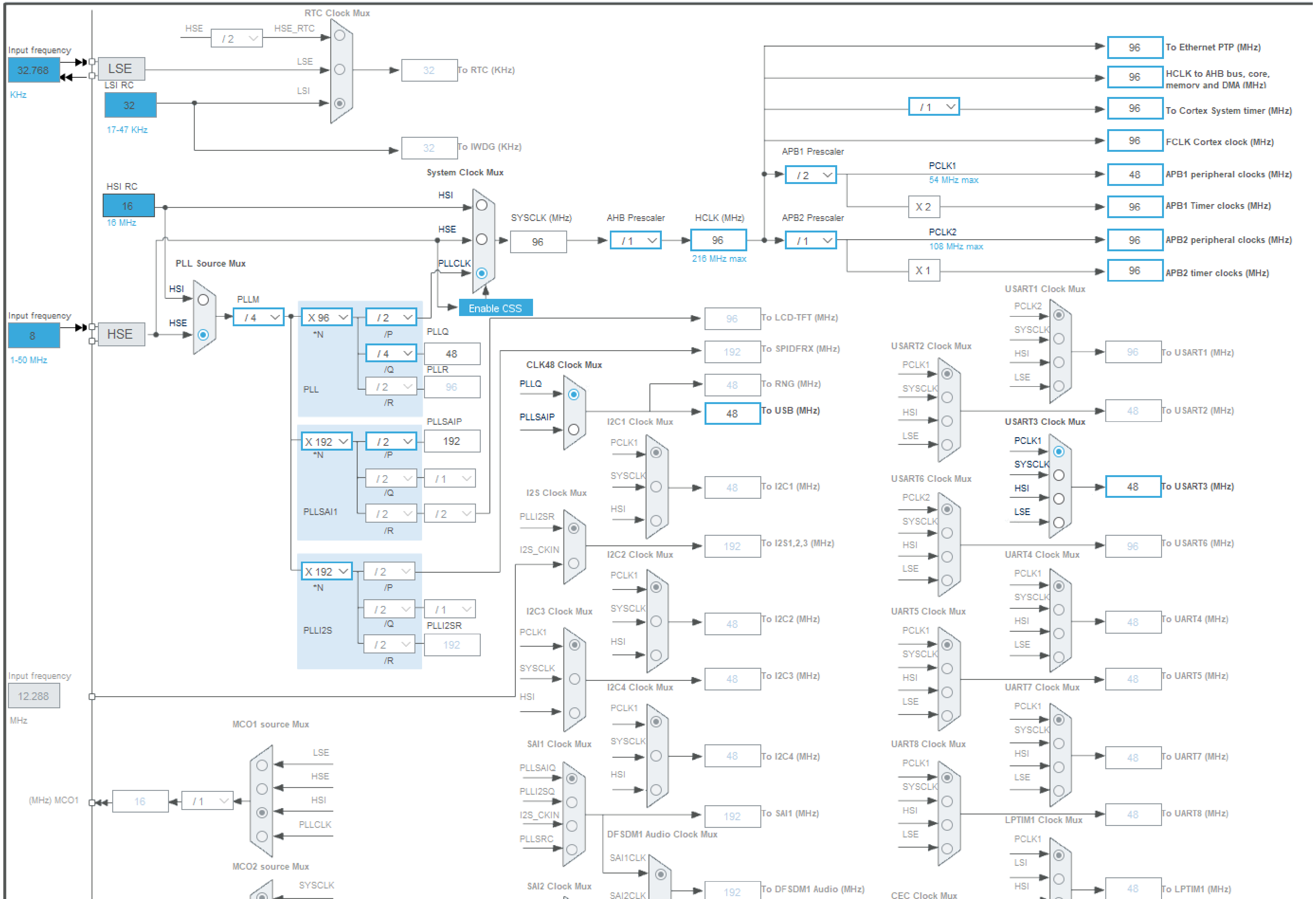
- Klokker
  - Pulstog av firkantbølger med konstant frekvens’



Klokkeperiode T_s

Klokkefrekvens $f_s = \frac{1}{T_S}$

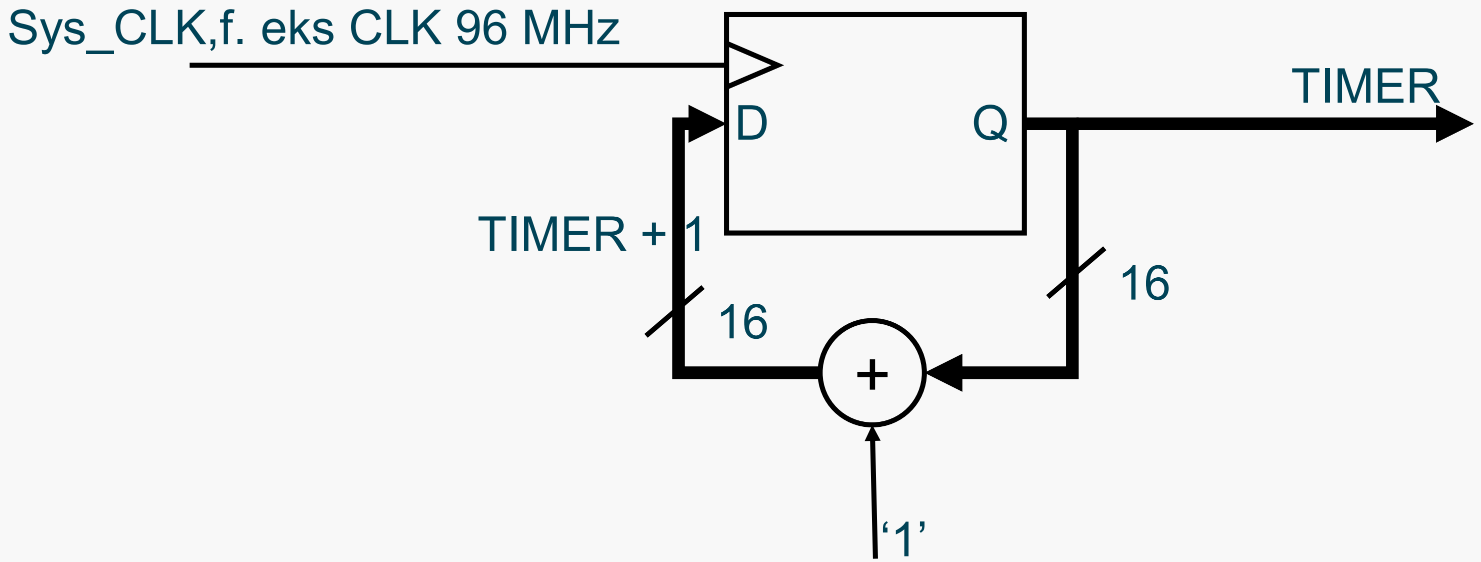
- Styrer tidspunkt og rekkefølge på utføring av instruksjonar
  - Program-element
- gir informasjon om tid til program
  - Kan generere interrupt
- Kan vera mange ulike klokker i eit system



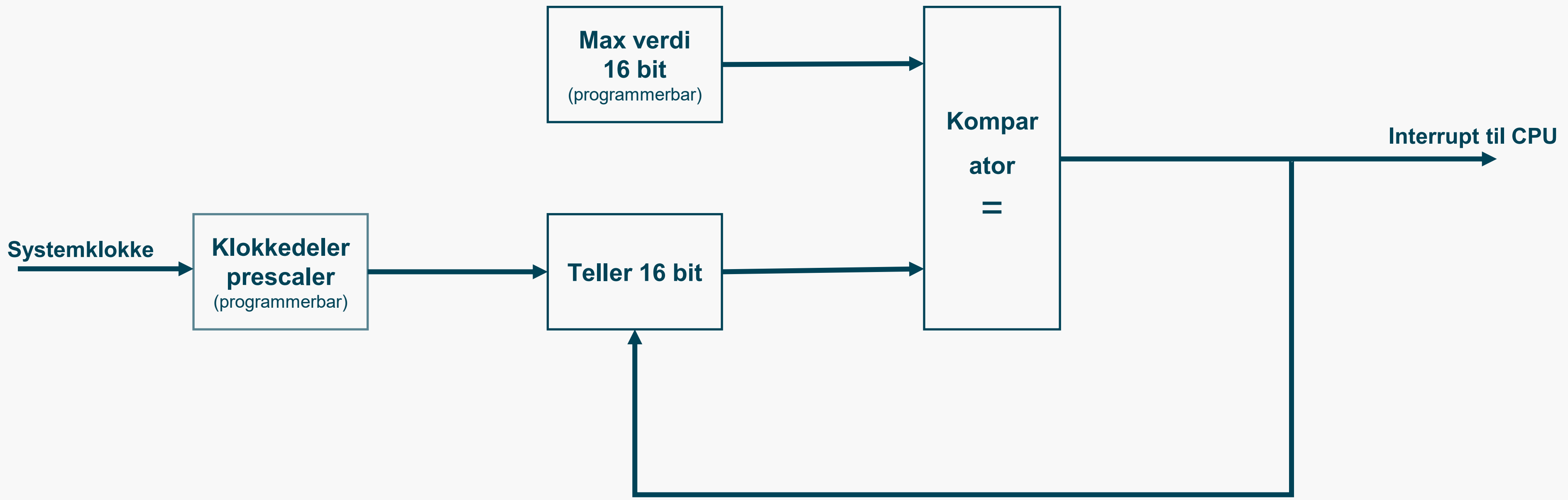
RTC Clock Mux
HSE
/ 2
HSE_RTC
Input frequency
32.768
KHz
LSE
LSE
LSI RC
32
17-47 KHz
LSI
32
To RTC (KHz)
32
To IWDG (KHz)
System Clock Mux
HSI RC
16
16 MHz
HSI
HSE
PLLCLK
SYSCLK (MHz)
96
AHB Prescaler
/ 1
HCLK (MHz)
96
216 MHz max
Enable CSS
PLL Source Mux
HSI
HSE
PLLM
/ 4
Input frequency
8
1-50 MHz
HSE
X 96
*N
/ 2
/P
PLLQ
/ 4
/Q
48
PLLR
/ 2
/R
96
PLL
PLLSAIP
X 192
*N
/ 2
/P
192
/ 2
/Q
/ 1
/ 2
/R
/ 2
PLLSAI1
X 192
*N
/ 2
/P
/ 2
/Q
/ 1
PLLI2SR
/ 2
/R
192
PLLI2S
Input frequency
12.288
MHz
MCO1 source Mux
LSE
HSE
HSI
PLLCLK
(MHz) MCO1
16
/ 1
MCO2 source Mux
SYSCLK
96
To Ethernet PTP (MHz)
96
HCLK to AHB bus, core, memory and DMA (MHz)
/ 1
96
To Cortex System timer (MHz)
96
FCLK Cortex clock (MHz)
APB1 Prescaler
/ 2
PCLK1
54 MHz max
48
APB1 peripheral clocks (MHz)
X 2
96
APB1 Timer clocks (MHz)
APB2 Prescaler
/ 1
PCLK2
108 MHz max
96
APB2 peripheral clocks (MHz)
X 1
96
APB2 timer clocks (MHz)
96
To LCD-TFT (MHz)
192
To SPIDFRX (MHz)
CLK48 Clock Mux
PLLQ
PLLSAIP
48
To RNG (MHz)
48
To USB (MHz)
I2C1 Clock Mux
PCLK1
SYSCLK
HSI
48
To I2C1 (MHz)
I2S Clock Mux
PLLI2SR
I2S_CKIN
192
To I2S1,2,3 (MHz)
I2C2 Clock Mux
PCLK1
SYSCLK
HSI
48
To I2C2 (MHz)
I2C3 Clock Mux
PCLK1
SYSCLK
HSI
48
To I2C3 (MHz)
I2C4 Clock Mux
PCLK1
SYSCLK
HSI
48
To I2C4 (MHz)
SAI1 Clock Mux
PLLSAIQ
PLLI2SQ
I2S_CKIN
PLLSRC
192
To SAI1 (MHz)
DFSDM1 Audio Clock Mux
SAI1CLK
SAI2CLK
192
To DFSDM1 Audio (MHz)
SAI2 Clock Mux
USART1 Clock Mux
PCLK2
SYSCLK
HSI
LSE
96
To USART1 (MHz)
USART2 Clock Mux
PCLK1
SYSCLK
HSI
LSE
48
To USART2 (MHz)
USART3 Clock Mux
PCLK1
SYSCLK
HSI
LSE
48
To USART3 (MHz)
USART6 Clock Mux
PCLK2
SYSCLK
HSI
LSE
96
To USART6 (MHz)
UART4 Clock Mux
PCLK1
SYSCLK
HSI
LSE
48
To UART4 (MHz)
UART5 Clock Mux
PCLK1
SYSCLK
HSI
LSE
48
To UART5 (MHz)
UART7 Clock Mux
PCLK1
SYSCLK
HSI
LSE
48
To UART7 (MHz)
UART8 Clock Mux
PCLK1
SYSCLK
HSI
LSE
48
To UART8 (MHz)
LPTIM1 Clock Mux
PCLK1
LSI
HSI
48
To LPTIM1 (MHz)
CEC Clock Mux

Timer

- Ein timer er ein HW-eining som held orden på tida i kretsen.
- Ein timer er ein teljar som aukar verdien sin med ein for kvar klokkesyklus;
  - Dedikert elektronikk i timer:
    - Går uavhengig av program-utføring



TIMER med klokkedeler og komparator



Frå databladet til STM32F767

- The devices include two advanced-control timers, eight general-purpose timers, two basic timers and two watchdog timers.

- Advanced-control: These are the most feature-rich timers, typically used for complex applications like motor control, power conversion, and high-resolution PWM.
- General purpose: These are versatile timers suitable for a wide range of applications, including general-purpose timing, PWM generation, input capture, output compare, and more.
- Basic: These are simpler timers, primarily used for basic timing and delay generation.

Table 6. Timer feature comparison

Timer type	Timer	Counter resolution	Counter type	Prescaler factor	DMA request generation	Capture/ compare channels	Complementary output	Max interface clock (MHz)	Max timer clock (MHz)(1)
Advanced -control	TIM1, TIM8	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	Yes	108	216
General purpose	TIM2, TIM5	32-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	54	108/216
	TIM3, TIM4	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	54	108/216
	TIM9	16-bit	Up	Any integer between 1 and 65536	No	2	No	108	216
	TIM10, TIM11	16-bit	Up	Any integer between 1 and 65536	No	1	No	108	216
	TIM12	16-bit	Up	Any integer between 1 and 65536	No	2	No	54	108/216
	TIM13, TIM14	16-bit	Up	Any integer between 1 and 65536	No	1	No	54	108/216
Basic	TIM6, TIM7	16-bit	Up	Any integer between 1 and 65536	Yes	0	No	54	108/216

1. The maximum timer clock is either 108 or 216 MHz depending on TIMPRE bit configuration in the RCC_DCKCFGR register.

HCLK (AHB Clock)

- It is a clock derived from SYSCLK. Clocks the CPU core, the AHB bus, and some AHB peripherals. Its frequency is typically lower than SYSCLK, as it is divided down using a **prescaler**. This helps:
- Optimize power consumption
- Allow different peripherals to run at different speeds If **SYSCLK** = 100 MHz and the prescaler is set to divide by 2, then **HCLK** = 50 MHz.

- **APB1 (Advanced Peripheral Bus 1):** This bus typically runs at a lower frequency than HCLK, set by a prescaler. It connects to peripherals like timers (TIM2–TIM7, TIM12–TIM14), USART2/3, I2C1/2/3, SPI2/3, and others. The lower frequency helps reduce power consumption for slower peripherals.
- **APB2 (Advanced Peripheral Bus 2):** This bus can run at the same frequency as HCLK or at a divided rate, depending on the prescaler setting. It connects to higher-speed peripherals such as TIM1, TIM8, USART1/6, SPI1, and the ADCs.

SysTick timer

- The SysTick timer on the STM32F767 microcontroller is a 24-bit down-counting timer embedded within the Cortex-M7 core itself, making it a highly integrated and essential component for real-time operating systems (RTOS) and general-purpose timing. It offers a simple yet effective mechanism for generating periodic interrupts, typically configured to fire at a regular interval (e.g., every millisecond) to drive the OS tick. Its preloader value is derived directly from the system clock (HCLK), ensuring precise and synchronized timing.

Bruka av timere i program

- HAL_Delay(x)
  - Venter i x millisekund
- x = HAL_GetTick();
  - Retunerer antall milliskund sidan programstart/reset.

- Timer –interrupt
  - Timer kan generer interrupt når
    - Timer-teller Når maksverdi
    - Timer-teller når bestemt verdi
    - Med jevne mellomrom
      - F.eks interrupt kvart µs

- PWM
  - Puls-bredde-modulering
  - Klokking av ADC
  - Klokking av DAC

HAL_GetTick()

- Styr LED med HAL_GetTick()
  - Med HAL_Delay() kan ikkje programmet gjera andre ting i ventetida
  - Med HAL_GetTick()er maskinen ledig til andre oppgåver.

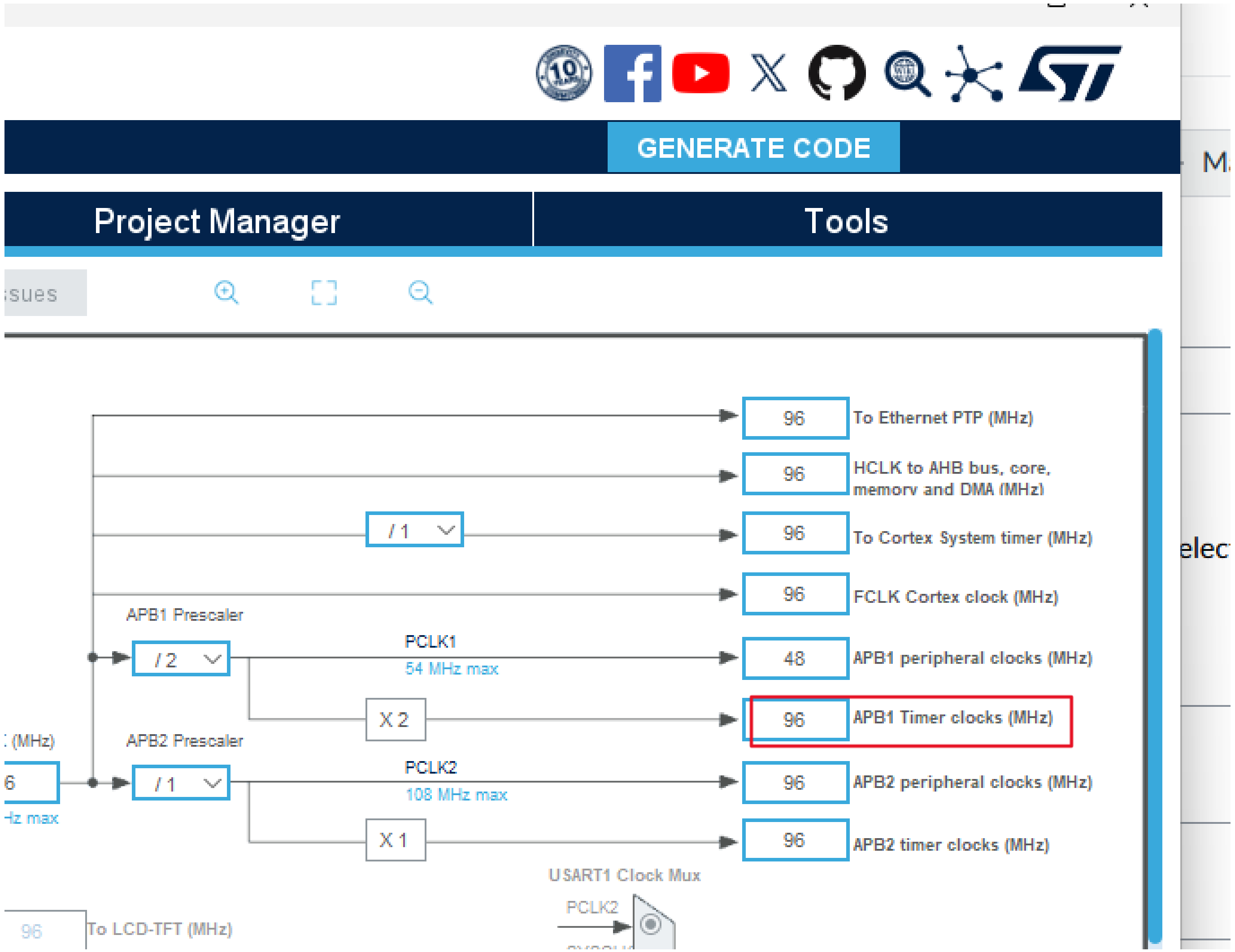
```c
  /* USER CODE BEGIN 2 */
uint32_t start_tid = HAL_GetTick();
const uint32_t ventetid = 1000;
/* USER CODE END 2 */
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
    uint32_t tid = HAL_GetTick();
    if (tid - start_tid > ventetid)
    {
        HAL_GPIO_TogglePin(LD1_GPIO_Port, LD1_Pin);
        start_tid = HAL_GetTick();
    }
}
/* USER CODE END 3 */
```

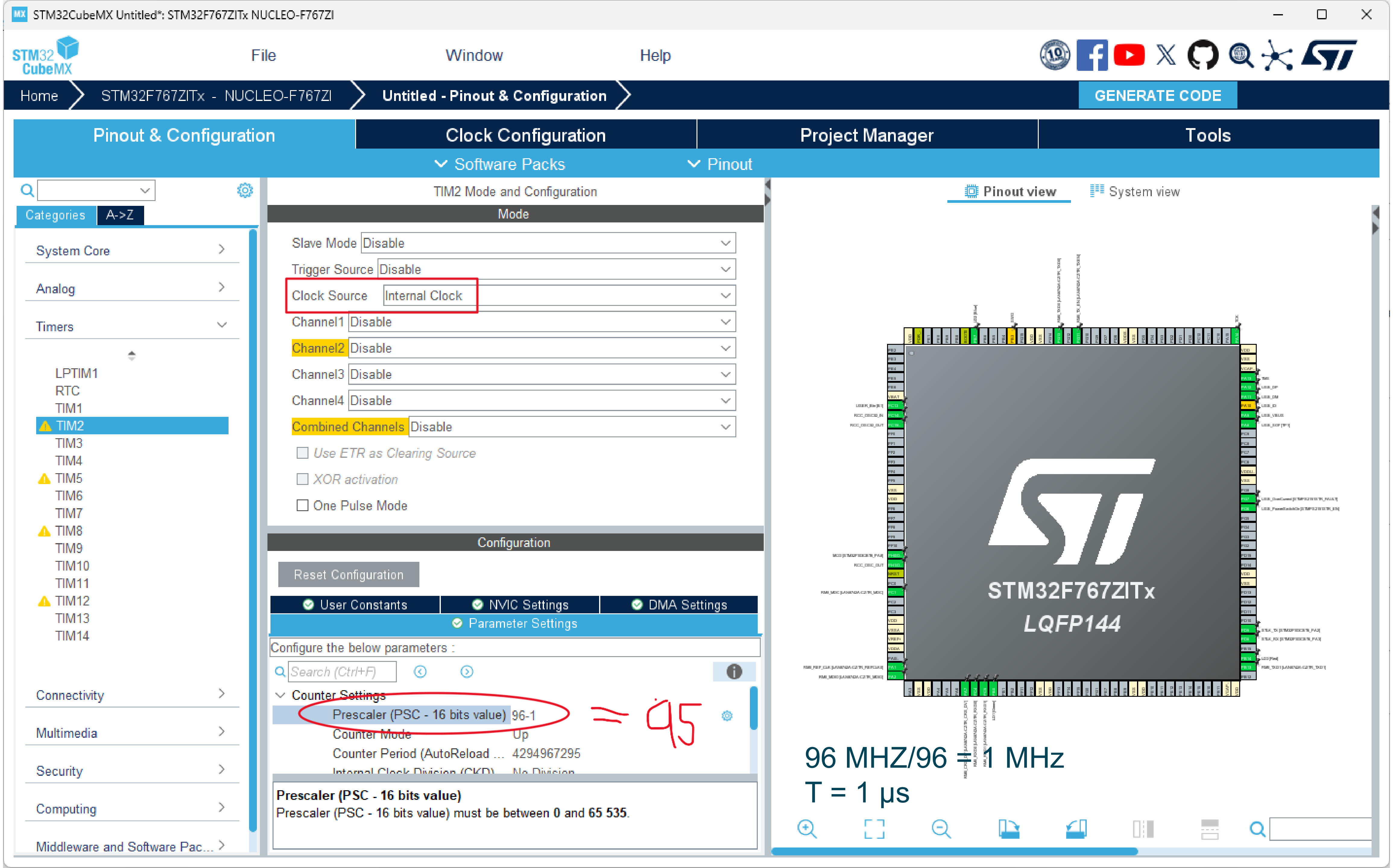
Oppgåver

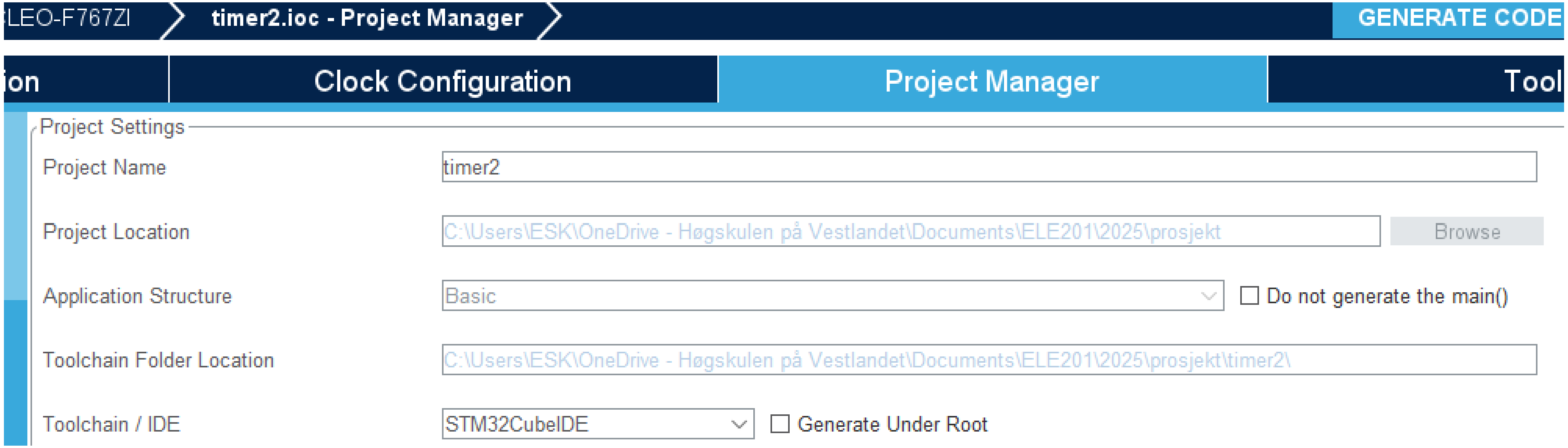
1. Styr LED med HAL_GetTick() i staden for HAL_Delay()
   - 3 LED med uavhengig blinkefrekvens
2. Styr LED med generell Timer
   - TIM2
3. Timer-interrrupt
   - Interrupt kvart 100 ms
     - Tidels sekund.

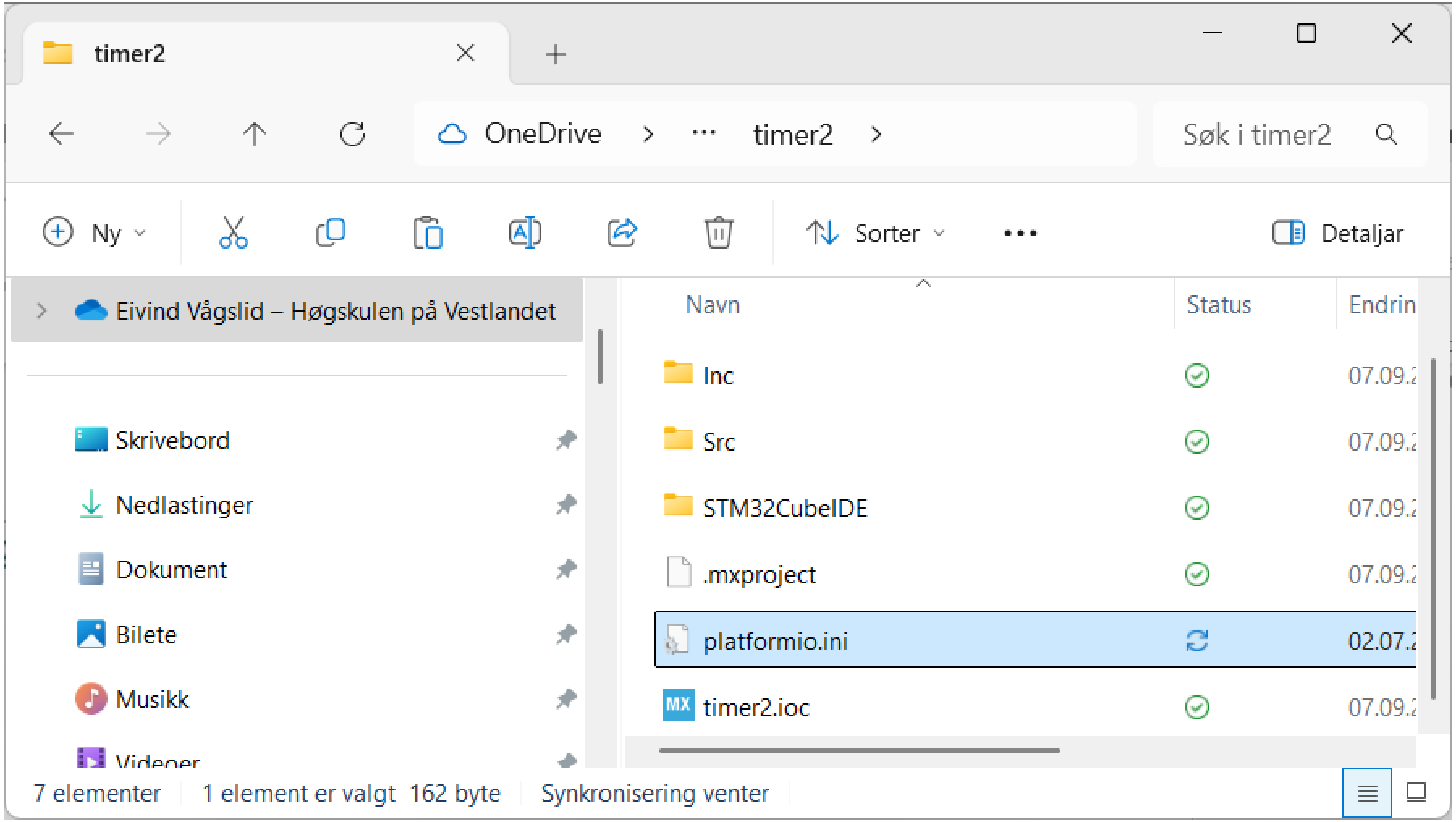
Timer 2 , APB1 CLK

Styr LED med Timer 2.




STM32CubeMX Untitled*: STM32F767ZITx NUCLEO-F767ZI
File
Window
Help
Home
STM32F767ZITx - NUCLEO-F767ZI
Untitled - Pinout & Configuration
GENERATE CODE
Pinout & Configuration
Clock Configuration
Project Manager
Tools
Software Packs
Pinout
Categories
A->Z
System Core
Analog
Timers
LPTIM1
RTC
TIM1
TIM2
TIM3
TIM4
TIM5
TIM6
TIM7
TIM8
TIM9
TIM10
TIM11
TIM12
TIM13
TIM14
Connectivity
Multimedia
Security
Computing
Middleware and Software Pac...
TIM2 Mode and Configuration
Mode
Slave Mode Disable
Trigger Source Disable
Clock Source Internal Clock
Channel1 Disable
Channel2 Disable
Channel3 Disable
Channel4 Disable
Combined Channels Disable
Use ETR as Clearing Source
XOR activation
One Pulse Mode
Configuration
Reset Configuration
User Constants
NVIC Settings
DMA Settings
Parameter Settings
Configure the below parameters :
Search (Ctrl+F)
Counter Settings
Prescaler (PSC - 16 bits value) 96-1
= 95
Counter Mode Up
Counter Period (AutoReload ... 4294967295
Internal Clock Division (CKD) No Division
Prescaler (PSC - 16 bits value)
Prescaler (PSC - 16 bits value) must be between 0 and 65 535.
Pinout view
System view
STM32F767ZITx
LQFP144
96 MHZ/96 = 1 MHz
T = 1 µs


LEO-F767ZI
timer2.ioc - Project Manager
GENERATE CODE
ion
Clock Configuration
Project Manager
Tool
Project Settings
Project Name
timer2
Project Location
C:\Users\ESK\OneDrive - Høgskulen på Vestlandet\Documents\ELE201\2025\prosjekt
Browse
Application Structure
Basic
Do not generate the main()
Toolchain Folder Location
C:\Users\ESK\OneDrive - Høgskulen på Vestlandet\Documents\ELE201\2025\prosjekt\timer2\
Toolchain / IDE
STM32CubeIDE
Generate Under Root


timer2
OneDrive
timer2
Søk i timer2
Ny
Sorter
Detaljar
Eivind Vågslid – Høgskulen på Vestlandet
Skrivebord
Nedlastinger
Dokument
Bilete
Musikk
Videoer
Navn
Status
Endrin
Inc
07.09.2
Src
07.09.2
STM32CubeIDE
07.09.2
.mxproject
07.09.2
platformio.ini
02.07.2
timer2.ioc
07.09.2
7 elementer
1 element er valgt 162 byte
Synkronisering venter

PIO Home | platformio.ini

platformio.ini

```ini
[env:nucleo_f767zi]
platform = ststm32
board = nucleo_f767zi
framework = stm32cube
build_flags =  -I./Inc -D HSE_VALUE=8000000
monitor_speed = 115200
```

```c
int main(void)
{

  /* USER CODE BEGIN 1 */
      // volatile keyword is very important!
      // it is not the MCU but a timer responsible in changing this variable
      // So your compiler optimizes this variable out
      // thinking that it is unused. Yeah, pretty stupid.
      volatile uint32_t timer_val;
  /* USER CODE END 1 */
```

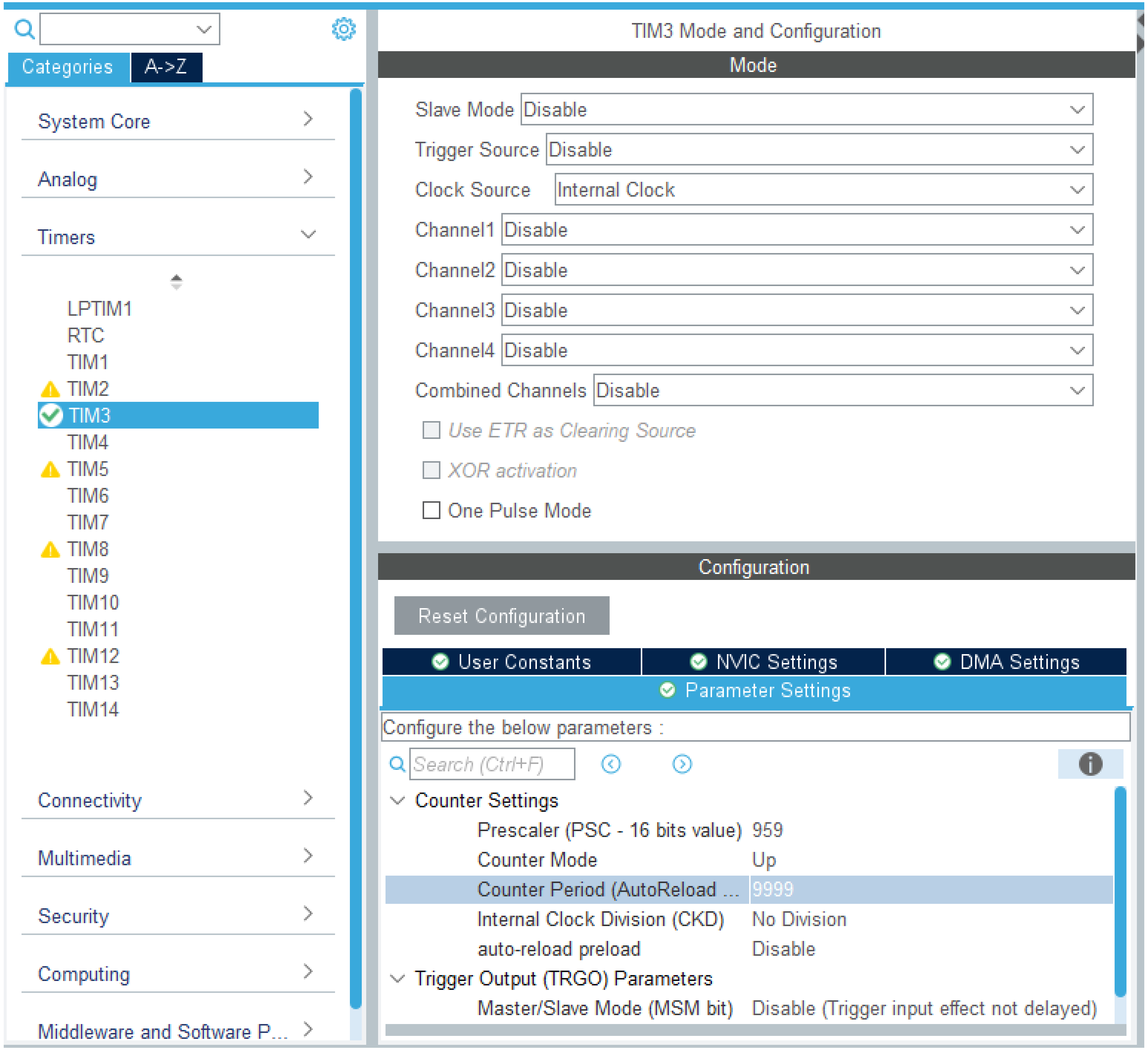
```c
  /* USER CODE BEGIN 2 */
      // Start timer
      HAL_TIM_Base_Start(&htim2);


      // Get current time (microseconds)
      timer_val =
__HAL_TIM_GET_COUNTER(&htim2);
  /* USER CODE END 2 */
```

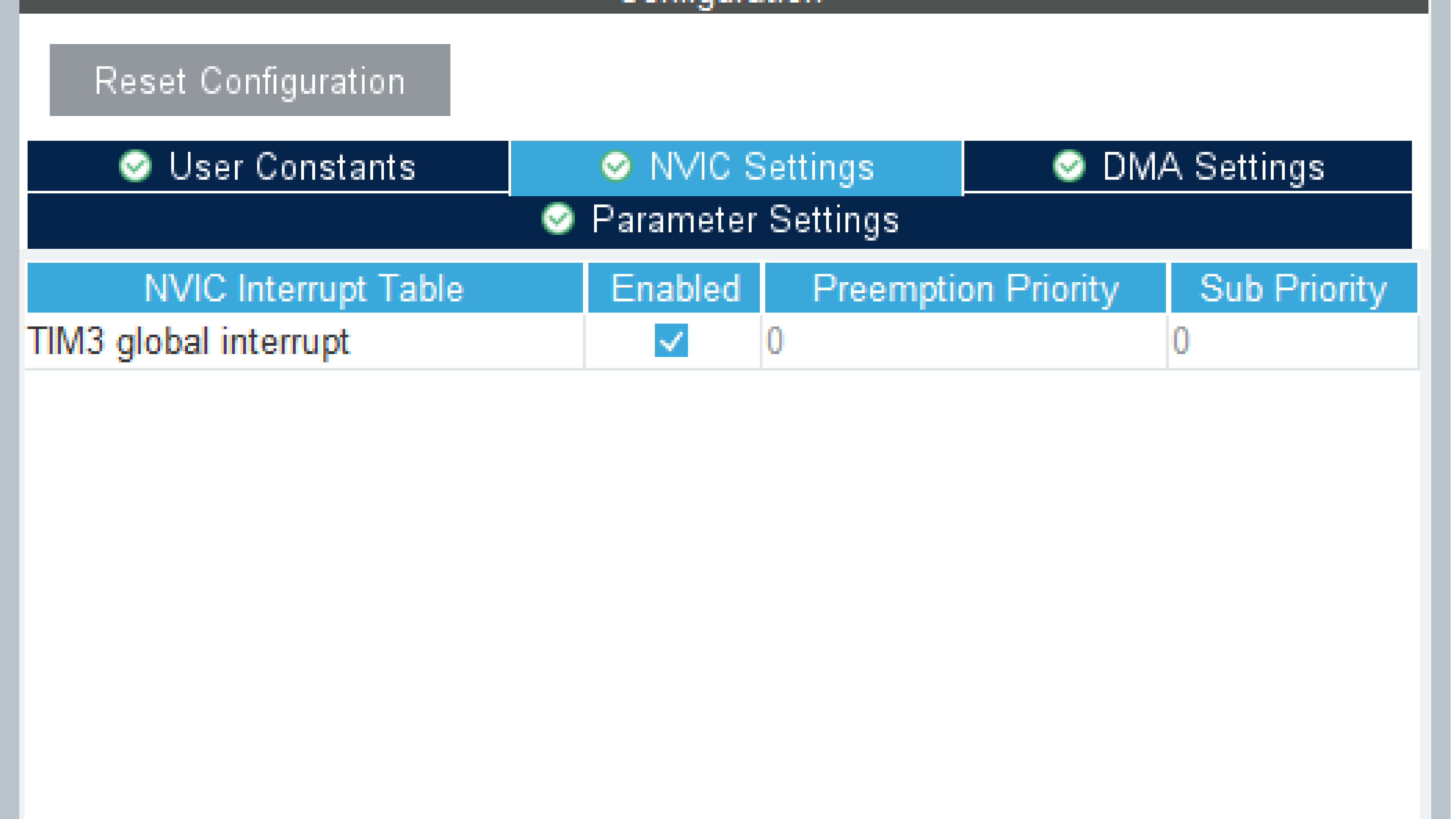
```c
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    // 1 million mikrosekund = 1 sekund
    if (__HAL_TIM_GET_COUNTER(&htim2) - timer_val >= 1000000)
    {
        HAL_GPIO_TogglePin(LD2_GPIO_Port, LD2_Pin);
        timer_val = __HAL_TIM_GET_COUNTER(&htim2);
    }
}
/* USER CODE END 3 */
```

Timer-interrupt

- Vi kan konfigurera ein timer til å gi interrupt med jevne mellomrom.
- Vi skal konfigurere TIMER3 til å gi interrupt kvart 100 ms (1/10 sekund),
- Og bruka dette til å Toggla LED3

- $T_{int} = 100ms = 100 \cdot 10^{-3}s$
- $f_{ABD1} = 96 \text{ MHz} = 96 \cdot 10^{6}Hz$
- $T_{int} = \frac{\{(TimerCountPeriod+1)\cdot(Presacle+1)\}}{f_{ADB1}}$
- $100\ 10^{-3} \cdot 96\ 10^{6}Hz = (TimerCountPeriod + 1) \cdot (Presacle + 1)$
- $(TimerCountPeriod + 1) \cdot (Presacle + 1) = 9600000 = 10\ 000 \cdot 960$
- For timer 3 (16 bit) er $TimerCountPeriod \leq 65535$
- Velger $TimerCountPeriod = 9999$
- $Prescale = 959$


TIM3 Mode and Configuration
Categories
A->Z
System Core
Analog
Timers
LPTIM1
RTC
TIM1
TIM2
TIM3
TIM4
TIM5
TIM6
TIM7
TIM8
TIM9
TIM10
TIM11
TIM12
TIM13
TIM14
Connectivity
Multimedia
Security
Computing
Middleware and Software P...
Mode
Slave Mode Disable
Trigger Source Disable
Clock Source Internal Clock
Channel1 Disable
Channel2 Disable
Channel3 Disable
Channel4 Disable
Combined Channels Disable
Use ETR as Clearing Source
XOR activation
One Pulse Mode
Configuration
Reset Configuration
User Constants
NVIC Settings
DMA Settings
Parameter Settings
Configure the below parameters :
Search (Ctrl+F)
Counter Settings
Prescaler (PSC - 16 bits value) 959
Counter Mode Up
Counter Period (AutoReload ... 9999
Internal Clock Division (CKD) No Division
auto-reload preload Disable
Trigger Output (TRGO) Parameters
Master/Slave Mode (MSM bit) Disable (Trigger input effect not delayed)


Reset Configuration
User Constants
NVIC Settings
DMA Settings
Parameter Settings
NVIC Interrupt Table
Enabled
Preemption Priority
Sub Priority
TIM3 global interrupt
0
0

- Automatisk generert kode
- I main.c:

```c
TIM_HandleTypeDef htim3;

static void MX_TIM3_Init(void);
```

- I stm32f7xx_it.c

```c
void TIM3_IRQHandler(void)
{
  /* USER CODE BEGIN TIM3_IRQn 0 */

  /* USER CODE END TIM3_IRQn 0 */
  HAL_TIM_IRQHandler(&htim3);
  /* USER CODE BEGIN TIM3_IRQn 1 */

  /* USER CODE END TIM3_IRQn 1 */
}
```

HAL_TIM_IRQHandler(&htim3);

```
  /* TIM Update event */
  if ((itflag & (TIM_FLAG_UPDATE)) ==
   (TIM_FLAG_UPDATE))
  {
    if ((itsource & (TIM_IT_UPDATE)) ==
   (TIM_IT_UPDATE))
    {
      __HAL_TIM_CLEAR_FLAG(htim,
   TIM_FLAG_UPDATE);
#if (USE_HAL_TIM_REGISTER_CALLBACKS == 1)
      htim->PeriodElapsedCallback(htim);
#else
      HAL_TIM_PeriodElapsedCallback(htim)
   ;
#endif /* USE_HAL_TIM_REGISTER_CALLBACKS
   */
    }
  }
```

- Vi må skriva funksjonen
- `HAL_TIM_PeriodElapsedCallback(htim)`
- I main.c

```c
/* USER CODE BEGIN 4 */
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef* htim)
{
    HAL_GPIO_TogglePin(LD3_GPIO_Port,LD2_Pin);
}
/* USER CODE END 4 */
```

- OBS, ingenting i while(1)-løkka!

- Også i main.c

```c
/* USER CODE BEGIN 2 */
  // Start timer 3
HAL_TIM_Base_Start_IT(&htim3);
/* USER CODE END 2 */
```