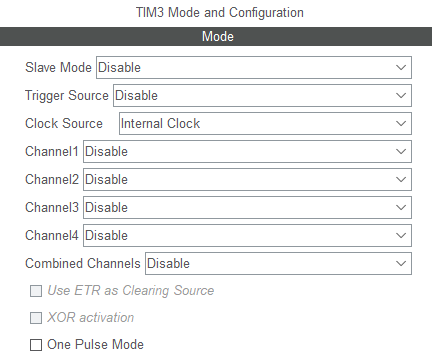
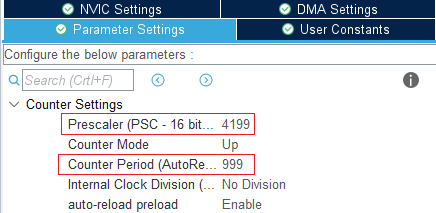
**STM32F407ZGT6-5-TIM3-Basic**

The purpose of this code is to generate a TIM3 Update Event every 100ms and use this period to control the LED.

TIM3 is attached to APB1 with 42MHz CLK (See system diagram in datasheet), so below setting achieves a 100ms period. **Pay attention here, pre-scale and ARR should be 1 less than desired value.**





So here, 42MHz / 4200 = 10KHz, with 1000 counts, period = 100ms every Event Update.

# void MX\_TIM3\_Init(void)

**The global part of main() file has below declaration:**

**TIM\_HandleTypeDef htim3;**

**Its detail is:**

**#if** (USE\_HAL\_TIM\_REGISTER\_CALLBACKS == 1)

**typedef** **struct** \_\_TIM\_HandleTypeDef

**#else**

**typedef** **struct**

**#endif** /\* USE\_HAL\_TIM\_REGISTER\_CALLBACKS \*/

{

**TIM\_TypeDef** \*Instance; /\* **Register base address** \*/

**TIM\_Base\_InitTypeDef** Init; /\*!< TIM Time Base required parameters \*/

HAL\_TIM\_ActiveChannel Channel; /\*!< Active channel \*/

DMA\_HandleTypeDef \*hdma[7]; /\*!< DMA Handlers array

This array is accessed by a @ref DMA\_Handle\_index \*/

HAL\_LockTypeDef Lock; /\*!< Locking object \*/

\_\_IO HAL\_TIM\_StateTypeDef State; /\*!< TIM operation state \*/

**#if** (USE\_HAL\_TIM\_REGISTER\_CALLBACKS == 1)

**void** (\* Base\_MspInitCallback)(**struct** \_\_TIM\_HandleTypeDef \*htim); /\*!< TIM Base Msp Init Callback \*/

**void** (\* Base\_MspDeInitCallback)(**struct** \_\_TIM\_HandleTypeDef \*htim); /\*!< TIM Base Msp DeInit Callback \*/

**void** (\* PeriodElapsedCallback)(**struct** \_\_TIM\_HandleTypeDef \*htim); /\*!< TIM Period Elapsed Callback \*/

**void** (\* PeriodElapsedHalfCpltCallback)(**struct** \_\_TIM\_HandleTypeDef \*htim); /\*!< TIM Period Elapsed half complete Callback \*/

**void** (\* TriggerCallback)(**struct** \_\_TIM\_HandleTypeDef \*htim); /\*!< TIM Trigger Callback \*/

**void** (\* TriggerHalfCpltCallback)(**struct** \_\_TIM\_HandleTypeDef \*htim); /\*!< TIM Trigger half complete Callback \*/

**#endif** /\* USE\_HAL\_TIM\_REGISTER\_CALLBACKS \*/

} **TIM\_HandleTypeDef**;

This code examples just use TIM3 to generate basic Update Event, which is called base configuration. We will consider below structures:

**TIM\_TypeDef**

This is the list of all TIM3 registers;

**TIM\_Base\_InitTypeDef**

/\*\*

\* @brief TIM Time base Configuration Structure definition

\*/

**typedef** **struct**

{

uint32\_t Prescaler;

/\* Specifies the prescaler value used to divide the TIM clock.

This parameter can be a number between Min\_Data = 0x0000 and Max\_Data = 0xFFFF \*/

uint32\_t CounterMode;

/\* Specifies the counter mode.

This parameter can be a value of @ref TIM\_Counter\_Mode \*/

uint32\_t Period;

/\* **Specifies the period value to be loaded into the active Auto-Reload Register at the next update event.**

This parameter can be a number between Min\_Data = 0x0000 and Max\_Data = 0xFFFF. \*/

uint32\_t ClockDivision;

/\* Specifies the clock division.

This parameter can be a value of @ref TIM\_ClockDivision \*/

uint32\_t RepetitionCounter;

/\* Specifies the repetition counter value. Each time the RCR downcounter reaches zero, an update event is generated and counting restarts from the RCR value (N).

This means in PWM mode that (N+1) corresponds to:

- the number of PWM periods in edge-aligned mode

- the number of half PWM period in center-aligned mode

GP timers: this parameter must be a number between Min\_Data = 0x00 and Max\_Data = 0xFF.

Advanced timers: this parameter must be a number between Min\_Data = 0x0000 and Max\_Data = 0xFFFF. \*/

uint32\_t AutoReloadPreload;

/\* Specifies the auto-reload preload.

This parameter can be a value of @ref **TIM\_AutoReloadPreload** \*/

} TIM\_Base\_InitTypeDef;

/\*\* @defgroup **TIM\_AutoReloadPreload** TIM Auto-Reload Preload

\*/

**#define** TIM\_AUTORELOAD\_PRELOAD\_DISABLE 0x00000000U

/\* TIMx\_ARR register is not buffered \*/

**#define** TIM\_AUTORELOAD\_PRELOAD\_ENABLE TIM\_CR1\_ARPE

/\* TIMx\_ARR register is buffered \*/void MX\_TIM3\_Init(void)

**void** **MX\_TIM3\_Init**(**void**) Body

**static** **void** **MX\_TIM3\_Init**(**void**)

{

/\* USER CODE BEGIN TIM3\_Init 0 \*/

/\* USER CODE END TIM3\_Init 0 \*/

TIM\_ClockConfigTypeDef sClockSourceConfig = {0};

TIM\_MasterConfigTypeDef sMasterConfig = {0};

/\* USER CODE BEGIN TIM3\_Init 1 \*/

/\* USER CODE END TIM3\_Init 1 \*/

htim3.Instance = TIM3;

htim3.Init.Prescaler = 4199;

htim3.Init.CounterMode = TIM\_COUNTERMODE\_UP;

htim3.Init.Period = 999;

htim3.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;

htim3.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_ENABLE;

**if** (**HAL\_TIM\_Base\_Init**(&htim3) != *HAL\_OK*)

{

Error\_Handler();

}

sClockSourceConfig.ClockSource = TIM\_CLOCKSOURCE\_INTERNAL;

**if** (HAL\_TIM\_ConfigClockSource(&htim3, &sClockSourceConfig) != *HAL\_OK*)

{

Error\_Handler();

}

sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_RESET;

sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;

**if** (HAL\_TIMEx\_MasterConfigSynchronization(&htim3, &sMasterConfig) != *HAL\_OK*)

{

Error\_Handler();

}

/\* USER CODE BEGIN TIM3\_Init 2 \*/

/\* USER CODE END TIM3\_Init 2 \*/

}

**HAL\_TIM\_Base\_Init() initializes the registers with designed value;**

# HAL\_TIM\_Base\_Start\_IT()

In side this function, the core sentence are:

**\_\_HAL\_TIM\_ENABLE\_IT(htim, TIM\_IT\_UPDATE);**

**\_\_HAL\_TIM\_ENABLE(htim)**

## \_\_HAL\_TIM\_ENABLE\_IT()

/\*\* @brief Enable the specified TIM interrupt.

\* @param \_\_HANDLE\_\_ specifies the TIM Handle.

\* @param \_\_INTERRUPT\_\_ specifies the TIM interrupt source to enable.

\* This parameter can be one of the following values:

\* @arg **TIM\_IT\_UPDATE**: Update interrupt

\* @arg TIM\_IT\_CC1: Capture/Compare 1 interrupt

\* @arg TIM\_IT\_CC2: Capture/Compare 2 interrupt

\* @arg TIM\_IT\_CC3: Capture/Compare 3 interrupt

\* @arg TIM\_IT\_CC4: Capture/Compare 4 interrupt

\* @arg TIM\_IT\_COM: Commutation interrupt

\* @arg TIM\_IT\_TRIGGER: Trigger interrupt

\* @arg TIM\_IT\_BREAK: Break interrupt

\* @retval None

\*/

**#define** \_\_HAL\_TIM\_ENABLE\_IT(\_\_HANDLE\_\_, \_\_INTERRUPT\_\_) ((\_\_HANDLE\_\_)->Instance->DIER |= (\_\_INTERRUPT\_\_))

Then we have:

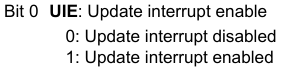
**#define** TIM\_IT\_UPDATE TIM\_DIER\_UIE

**#define** TIM\_DIER\_UIE\_Pos (0U)

**#define** TIM\_DIER\_UIE\_Msk (0x1UL << TIM\_DIER\_UIE\_Pos)

**#define** TIM\_DIER\_UIE TIM\_DIER\_UIE\_Msk

So, here we set DIER[0] to 1, which is:



## \_\_HAL\_TIM\_ENABLE()

There is another function macro called \_\_HAL\_TIM\_ENABLE(), though we didn’t use it here.

/\*\*

\* @brief Enable the TIM peripheral.

\* @param \_\_HANDLE\_\_ TIM handle

\* @retval None

\*/

**#define** \_\_HAL\_TIM\_ENABLE(\_\_HANDLE\_\_) ((\_\_HANDLE\_\_)->Instance->CR1 |= (TIM\_CR1\_CEN))

**#define** TIM\_CR1\_CEN\_Pos (0U)

**#define** TIM\_CR1\_CEN\_Msk (0x1UL << TIM\_CR1\_CEN\_Pos)

**#define** TIM\_CR1\_CEN TIM\_CR1\_CEN\_Msk

**So this function enables CEN.**

