Basic timers

This tutorial demonstrates controlling the onboard LED1 and LED2 blinkers on the dev board using a **basic timer (TIM6)**.

1、software-hardware

- **STM32F103CubeIDE**
- **SSTM32 Robot Development Board**

  TIM: Chip internal peripherals

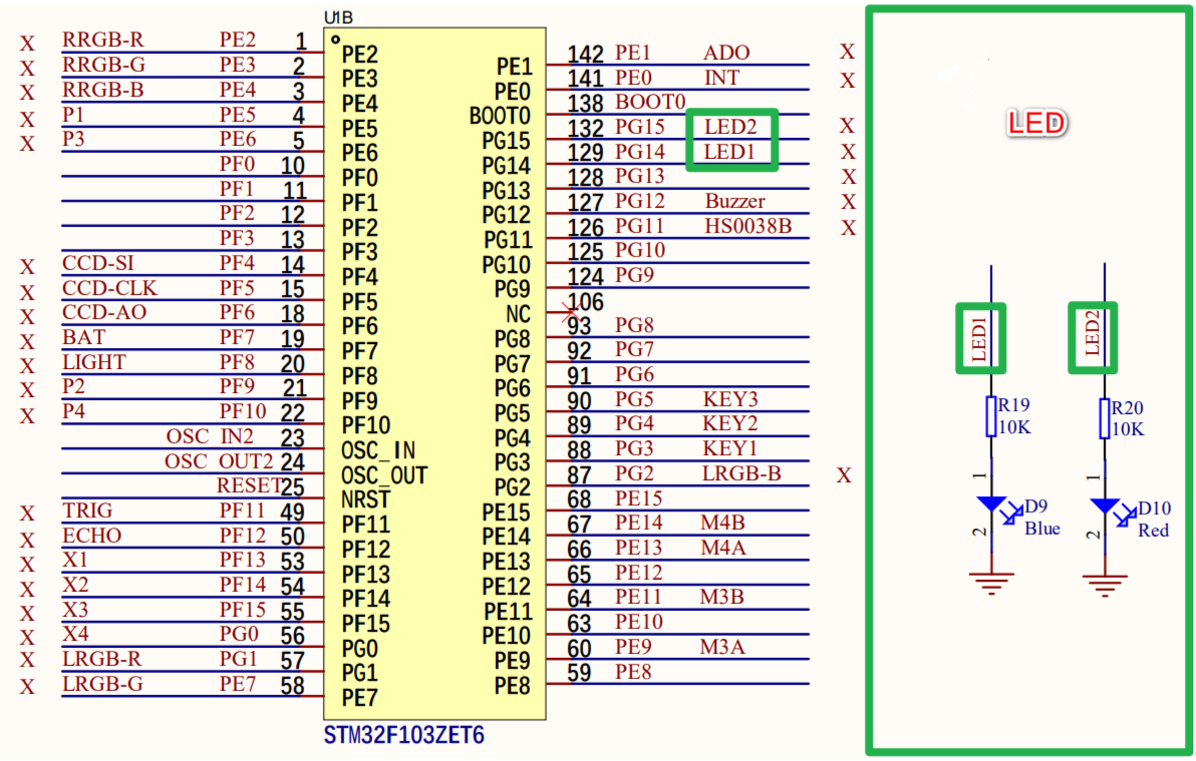
  LED light: onboard

- **Type-C cable or ST-Link**

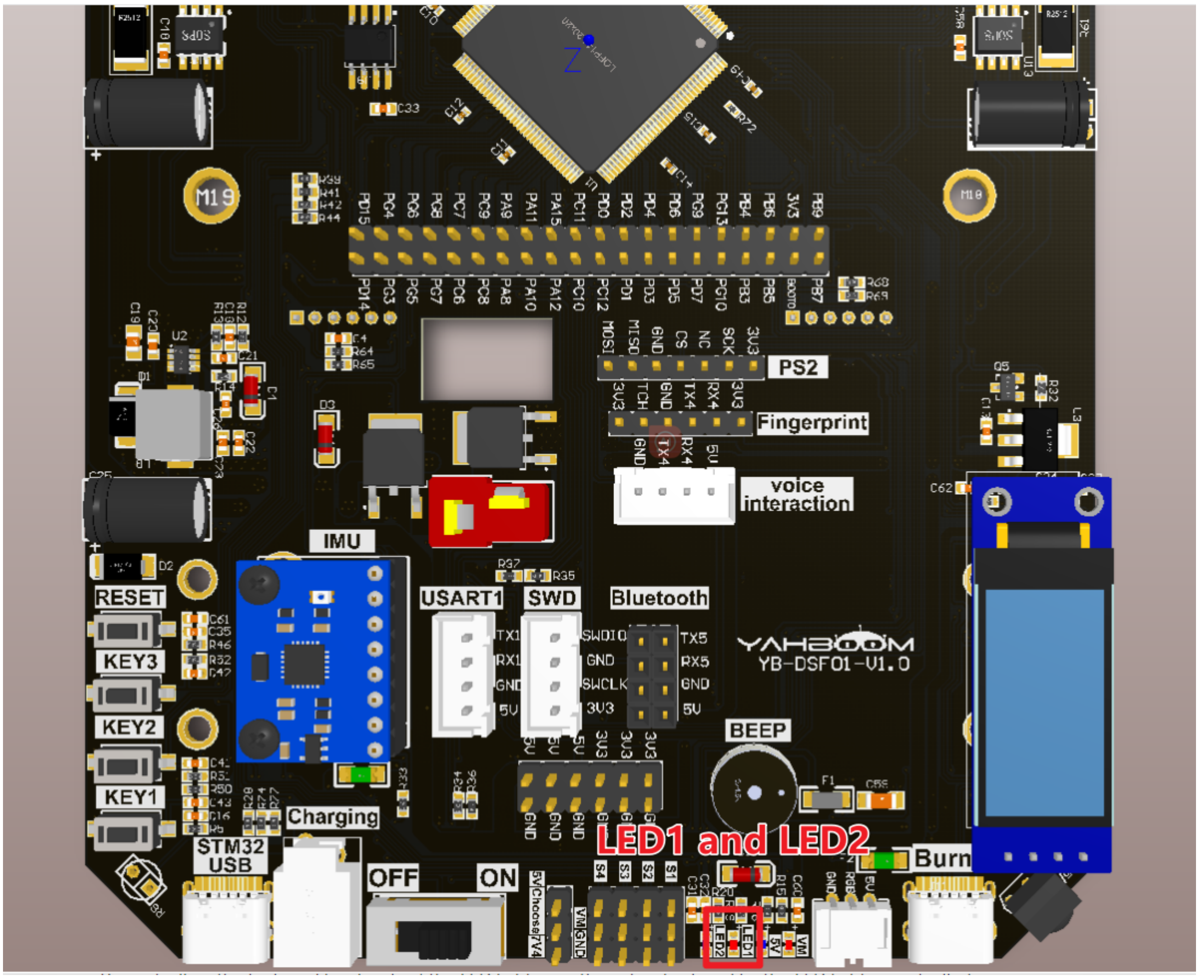
  Download or simulate the program of the development board

2、Brief principle

2.1、Hardware schematic diagram



2.2、Physical connection diagram



2.3、Principle of control

The GPIO output function is used to control the LED, and the timing function is realized through the basic timer.

- ### GPIO output

By controlling the high and low level of the LED lamp pin to control the color of the LED lamp display.

LED：High level light, low level off

LED（Schematic name)	Control pin	Functions
LED1	PG14	Control LED1 to turn on and off
LED2	PG15	Control LED2 to turn on and off

- ### Basic timers

The timing function of TIM6 on the STM32F103ZET6 development board was used

Timer types	Basic timer
The name of timer	TIM6、TIM7
Number of counter bits	16
Counting mode	incrementally
Predivision coefficient	1-65536
Generating DMA requests	Yes
Capture/compare channels	0
Complementary output	No
Clock frequency	72MHz (Max)
Mount bus	APB1

> **Time base unit**

register	Funciton
The counter register（TIMx_CNT)	The current value of the counter
Predivision register（TIMx_PSC)	Set frequency division coefficient (1-65536)
Automatically reload registers（TIMx_ARR)	The counter counts the boundary and the overloaded value

> **Timing formula**

$$T(s) = \frac{(ARR+1)*(PSC+1)}{TIM\_CLK(Hz)}$$

parameters	Meaning
T（s）	Timing time in seconds
ARR	Automatically reload the value
PSC	Predivision coefficient
TIM_CLK	The timer ticks in Hz

Timing time for the project: 10ms

$$T(s) = \frac{(ARR+1)*(PSC+1)}{TIM\_CLK(Hz)} = \frac{(99+1)*(7199+1)}{72000000(Hz)} = 0.01s$$

3、Engineering configuration

Project Configuration: Prompts for configuration options in the STM32CubeIDE project configuration process

3.1、Tips

Omitted project configuration: **New project, chip selection, project configuration, SYS for pin configuration, RCC configuration, clock configuration, and project configuration** content

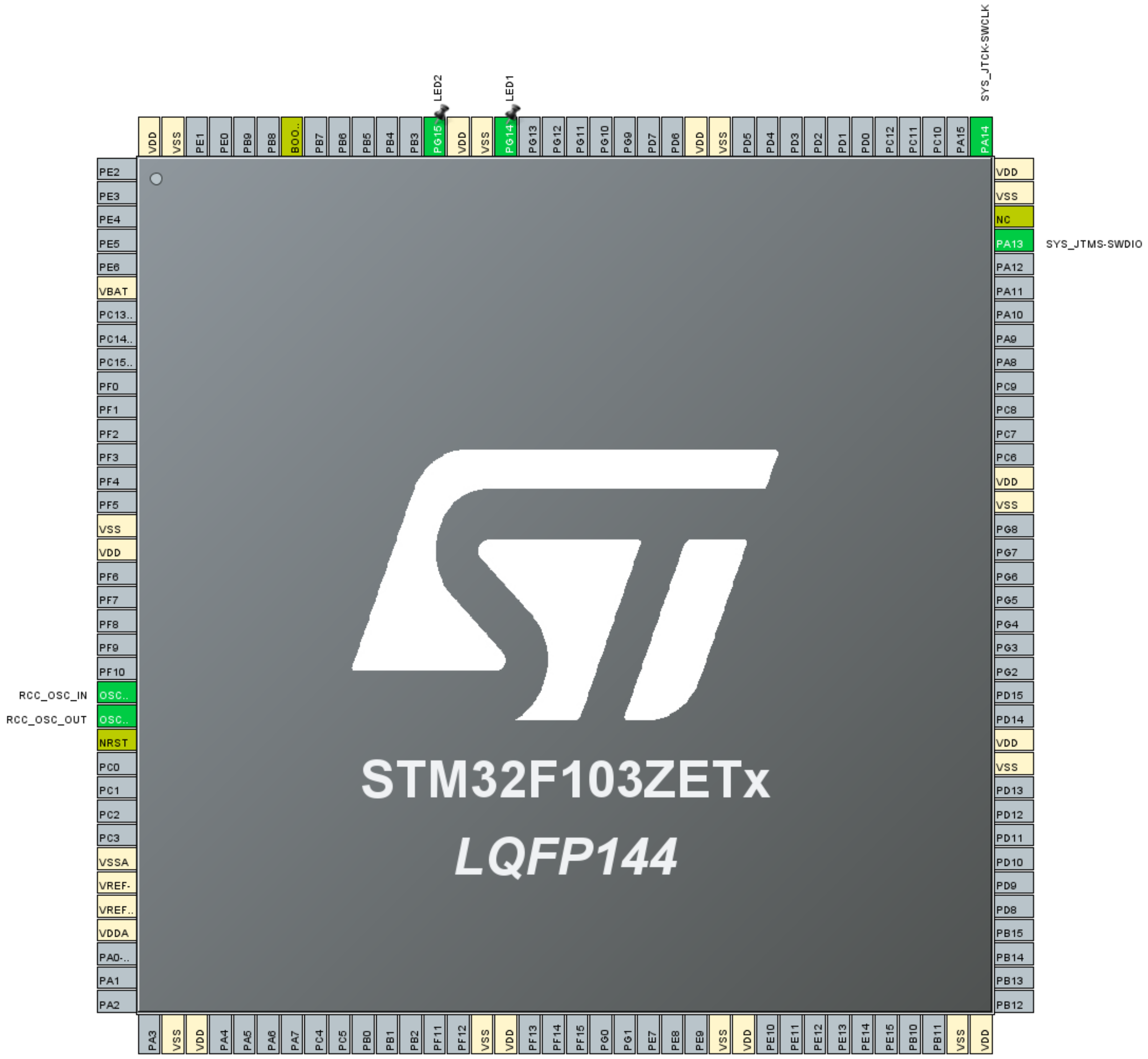
The project configuration part, which is not omitted, is the key point to configure in this tutorial.

```
Please refer to [2. Development environment construction and use: STM32CubeIDE installation and use] to understand how to configure the omitted parts of the project.
```

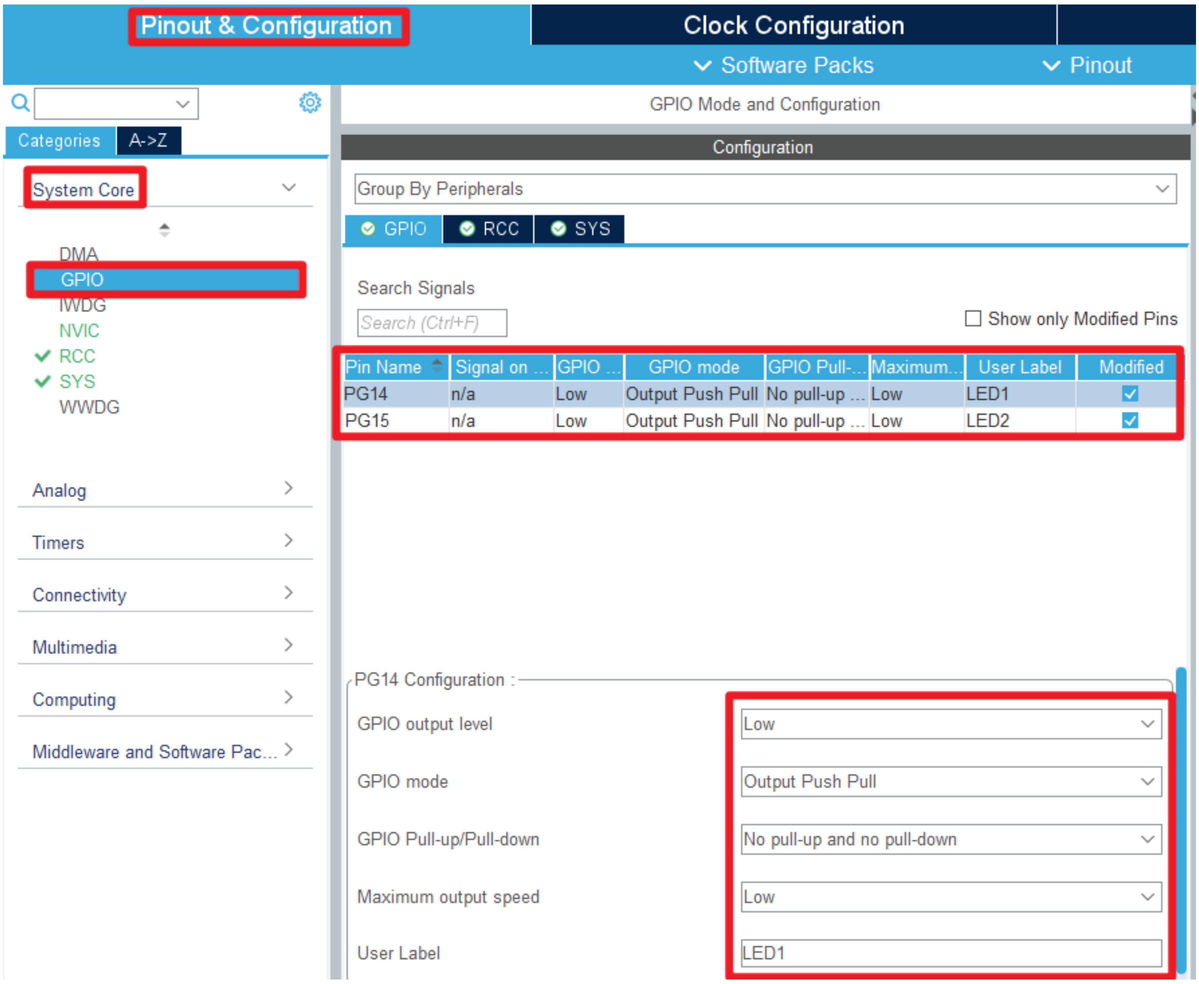
3.2、Pin configuration

- **Configure the specified pin function**

You can directly select the corresponding pin number in the pin view, and the corresponding option will appear when the mouse is left clicked


LED2
LED1
SYS_JTCK-SWCLK
PG15
PG14
PA14
PA13
SYS_JTMS-SWDIO
RCC_OSC_IN
RCC_OSC_OUT
OSC..
OSC..
NRST
BOO..
NC
STM32F103ZETx
LQFP144

- **GPIO**



GPIO Mode and Configuration

Configuration

Group By Peripherals

✓ GPIO ✓ RCC ✓ SYS

Search Signals

Search (Ctrl+F)

□ Show only Modified Pins

Pin Name	Signal on ...	GPIO ...	GPIO mode	GPIO Pull-...	Maximum...	User Label	Modified
PG14	n/a	Low	Output Push Pull	No pull-up ...	Low	LED1	☑
PG15	n/a	Low	Output Push Pull	No pull-up ...	Low	LED2	☑

PG14 Configuration :

GPIO output level: Low

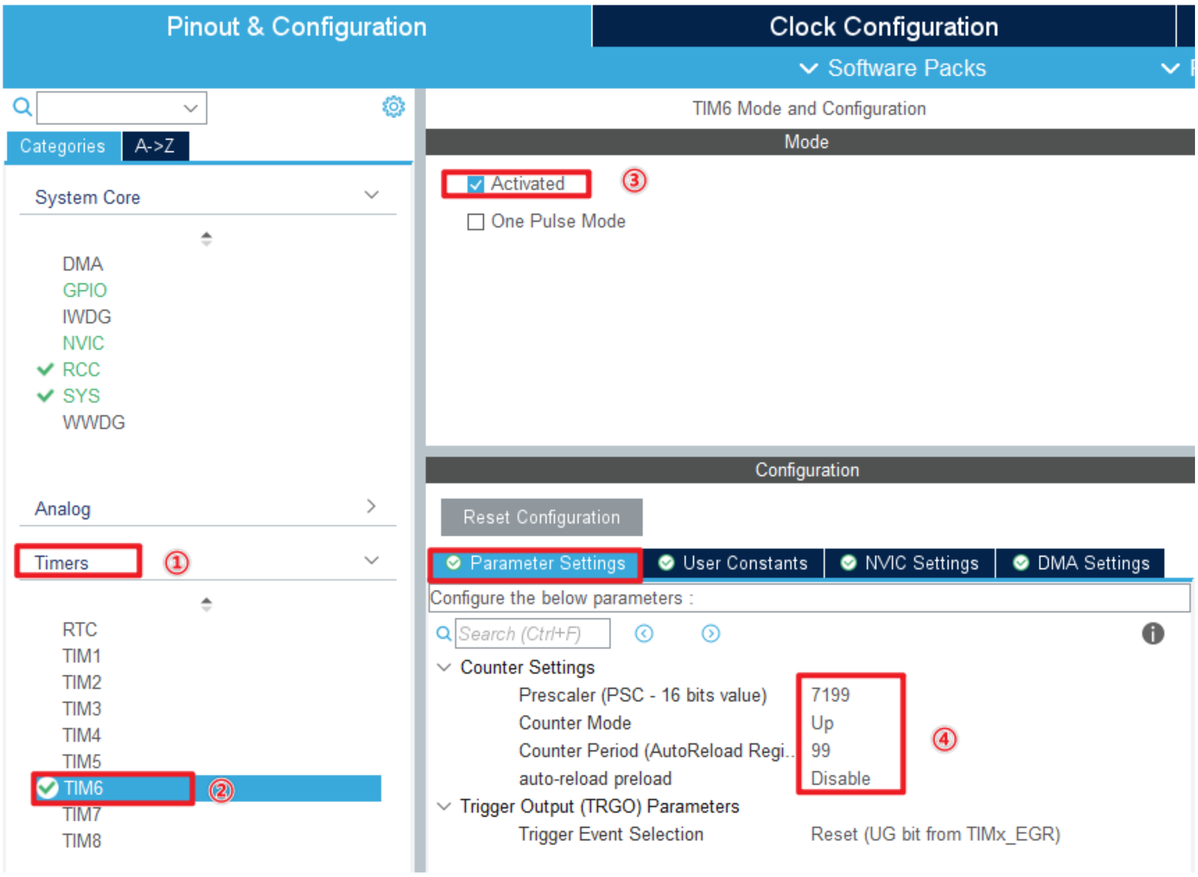
GPIO mode: Output Push Pull

GPIO Pull-up/Pull-down: No pull-up and no pull-down

Maximum output speed: Low

User Label: LED1

- **Timers**



TIM6 Mode and Configuration

Mode

☑ Activated

□ One Pulse Mode

Configuration

Reset Configuration

✓ Parameter Settings ✓ User Constants ✓ NVIC Settings ✓ DMA Settings

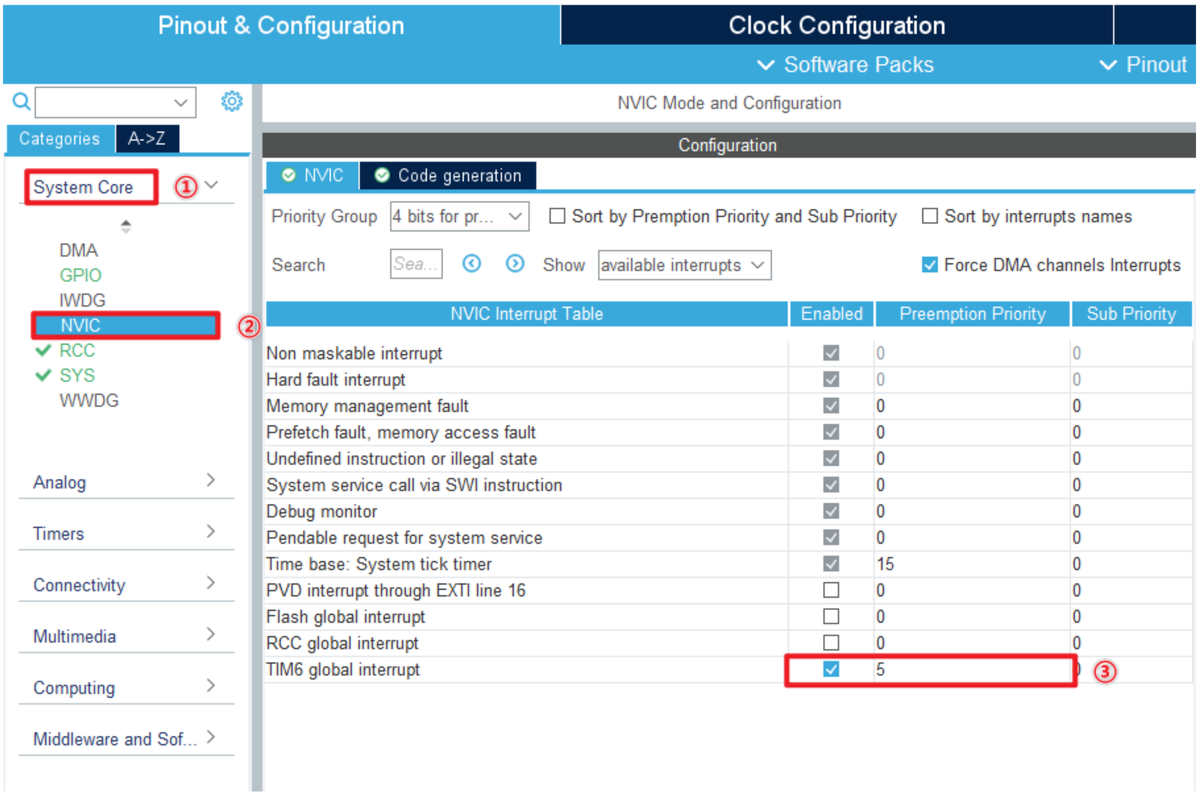
Configure the below parameters :

Search (Ctrl+F)

Counter Settings
- Prescaler (PSC - 16 bits value): 7199
- Counter Mode: Up
- Counter Period (AutoReload Regi..: 99
- auto-reload preload: Disable

Trigger Output (TRGO) Parameters
- Trigger Event Selection: Reset (UG bit from TIMx_EGR)

- **NVIC**

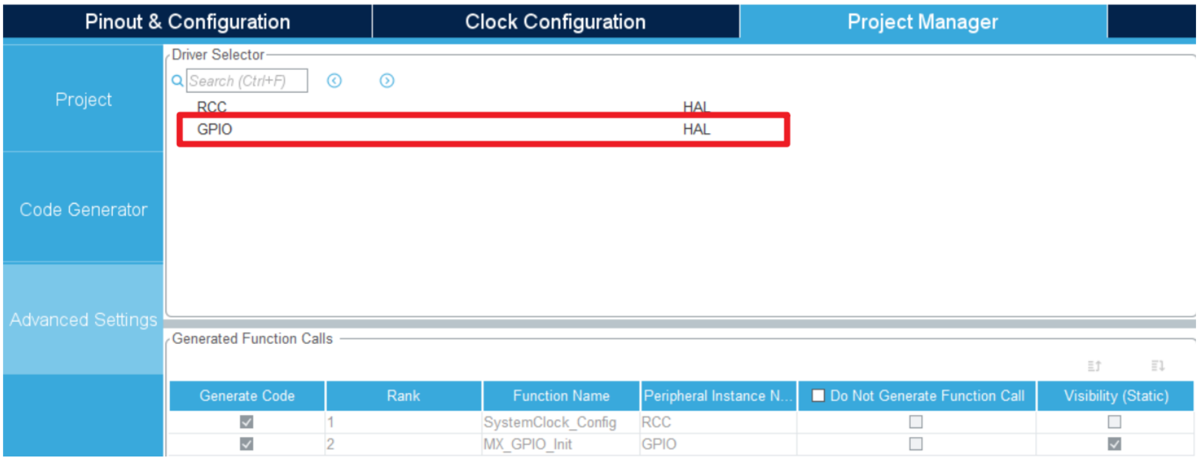


Priority Group 4 bits for pr... □ Sort by Premption Priority and Sub Priority □ Sort by interrupts names

Search Sea... Show available interrupts ☑ Force DMA channels Interrupts

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	☑	0	0
Hard fault interrupt	☑	0	0
Memory management fault	☑	0	0
Prefetch fault, memory access fault	☑	0	0
Undefined instruction or illegal state	☑	0	0
System service call via SWI instruction	☑	0	0
Debug monitor	☑	0	0
Pendable request for system service	☑	0	0
Time base: System tick timer	☑	15	0
PVD interrupt through EXTI line 16	□	0	0
Flash global interrupt	□	0	0
RCC global interrupt	□	0	0
TIM6 global interrupt	☑	5	0

- **Advanced Settings**



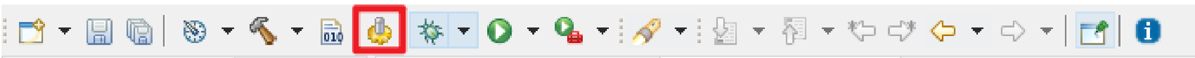
Driver Selector

RCC	HAL
GPIO	HAL

Generated Function Calls

Generate Code	Rank	Function Name	Peripheral Instance N...	□ Do Not Generate Function Call	Visibility (Static)
☑	1	SystemClock_Config	RCC	□	□
☑	2	MX_GPIO_Init	GPIO	□	☑

- **Generating code**



4、Main functions

This section mainly introduces the functional code written by users. **Detailed code can be opened by yourself in the project file we provide, and enter the Bsp folder to view the source code.**

4.1、User function

> HAL_TIM_IRQHandler：The timer interrupt service function

Function prototypes	**void HAL_TIM_IRQHandler(TIM_HandleTypeDef *htim)**
Functional Description	**The timer interrupt service function**

Function prototypes	**void HAL_TIM_IRQHandler(TIM_HandleTypeDef *htim)**
Input parameters	**htim**：Timer handle address
Return value	None
Notes	1. Internally, the function needs to determine the interrupt type and clear the corresponding interrupt flag, and finally call the callback function Define the USE_HAL_IRQ macro to switch between different interrupt handling functions

```
/*
If the HAL library is used, the HAL_TIM_IRQHandler function is invoked to
automatically handle timer interrupts.
If the HAL library is not used, the timer update interrupt is checked to see if
the interrupt bit is set.
*/
void TIM6_IRQHandler(void)
{
#if USE_HAL_IRQ
  HAL_TIM_IRQHandler(&htim6);//Using HAL_TIM_PeriodElapsedCallback is
automatically invoked
#else
  if (__HAL_TIM_GET_FLAG(&htim6, TIM_FLAG_UPDATE) != RESET)//Check if the TIM
update interrupt occurred
    {
      if (__HAL_TIM_GET_IT_SOURCE(&htim6, TIM_IT_UPDATE) != RESET) //Check if
interrupts for TIM6 are enabled
      {
        __HAL_TIM_CLEAR_IT(&htim6, TIM_IT_UPDATE);//Clearing interrupts
        g_time++;
        if(g_time % 100 == 0)//1s = 100*10ms
         {
             g_time = 1;
             HAL_GPIO_TogglePin(LED1_GPIO_Port,LED1_Pin);
         }
        else if(g_time % 50 == 0)//500ms = 50*10ms
         {
             HAL_GPIO_TogglePin(LED2_GPIO_Port,LED2_Pin);
         }
      }
    }
#endif
}
```

> **HAL_TIM_PeriodElapsedCallback**：**The timer update interrupt callback function**

Function prototypes	**void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)**
Functional Description	**The timer update interrupt callback function**
Input parameters	**htim**：Timer handle address
Return value	None
Notes	This function is called by HAL_TIM_IRQHandler, which allows you to write specific tasks

```
/*
HAL library internal timer update interrupt callback function
*/
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
     if(htim->Instance==TIM6)
      {
         g_time++;
      }
     if(g_time % 100 == 0)//1s = 100*10ms
     {
         g_time = 1;
         HAL_GPIO_TogglePin(LED1_GPIO_Port,LED1_Pin);
     }
     if(g_time % 50 == 0)//500ms = 50*10ms
     {
         HAL_GPIO_TogglePin(LED2_GPIO_Port,LED2_Pin);
     }
}
```

4.2、HAL library function parsing

The HAL library functions that were covered in the previous tutorial will not be covered

```
If you want to find the HAL library and LL library function analysis involved in
the entire tutorial, you can view the documents in the folder [8. STM32 Manual:
STM32F1_HAL Library and LL Library_User Manual]
```

> **function：HAL_TIM_Base_Init**

Function prototypes	**HAL_StatusTypeDef HAL_TIM_Base_Init(TIM_HandleTypeDef *htim)**
Functional Description	**Initialize the timer base unit**
Input parameters	**htim**：Timer handle address

Function prototypes	**HAL_StatusTypeDef HAL_TIM_Base_Init(TIM_HandleTypeDef *htim)**
Return value	**HAL status value**：HAL_OK、HAL_ERROR、HAL_BUSY、HAL_TIMEOUT
Notes	This calls the MCU low-level initialization function HAL_TIM_Base_MspInit to set the pins, clocks, and interrupts

> **function：HAL_TIM_Base_MspInit**

Function prototype	**void HAL_TIM_Base_MspInit(TIM_HandleTypeDef *htim);**
Functional description	**Initialize the peripheral clock, GPIO, and NVIC for the timer**
Input parameters	**htim**：Timer handle address
Return value	None

> **function：HAL_TIM_Base_MspDeInit**

Function prototype	**void HAL_TIM_Base_MspDeInit(TIM_HandleTypeDef *htim)**
Functional description	**Uninitialize the timer peripheral clock, GPIO, and NVIC**
Input parameters	**htim**：Timer handle address
Return value	None

> **function：HAL_TIM_Base_Start_IT**

Function prototype	**HAL_StatusTypeDef HAL_TIM_Base_Start_IT(TIM_HandleTypeDef *htim)**
Functional description	**Starts the timer and enables the interrupt function of the timer能**
Input parameters	**htim**：Timer handle address
Return value	**HAL status value**：HAL_OK、HAL_ERROR、HAL_BUSY、HAL_TIMEOUT
Notes	This function needs to be called by the user to enable and start the update interrupt of the timer

5、Experimental phenomenon

After downloading the program successfully, press the RESET button of the development board to observe the phenomenon of the development board

```
For program download, please refer to [2. Development environment construction
and use: program download and simulation]
```

现象：

LED1： The bright-off time interval is 1 second

LED2： The bright-off time interval is 0.5 seconds

```
Experimental phenomena can be found in [Basic Timer_Experimental Phenomenon.mp4]
```