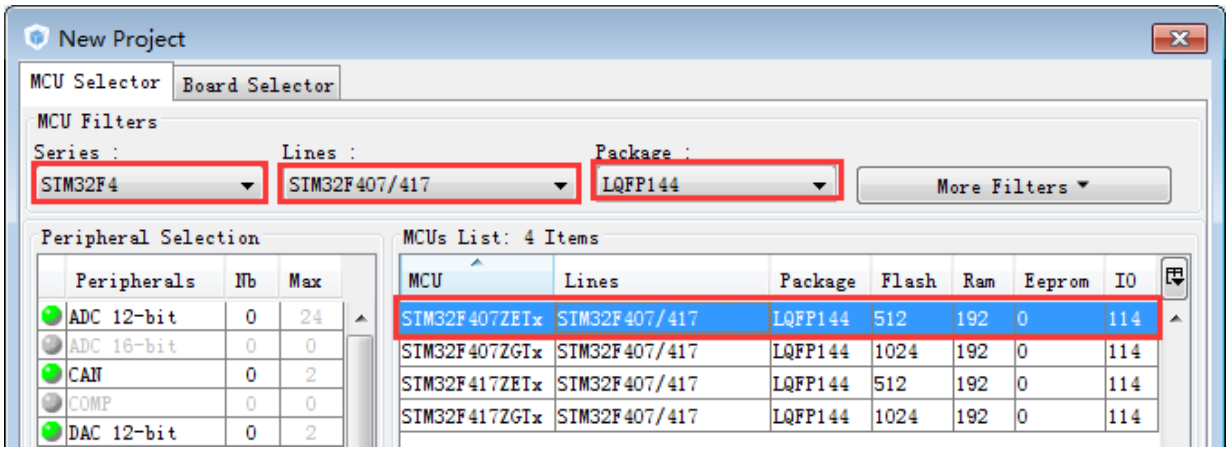
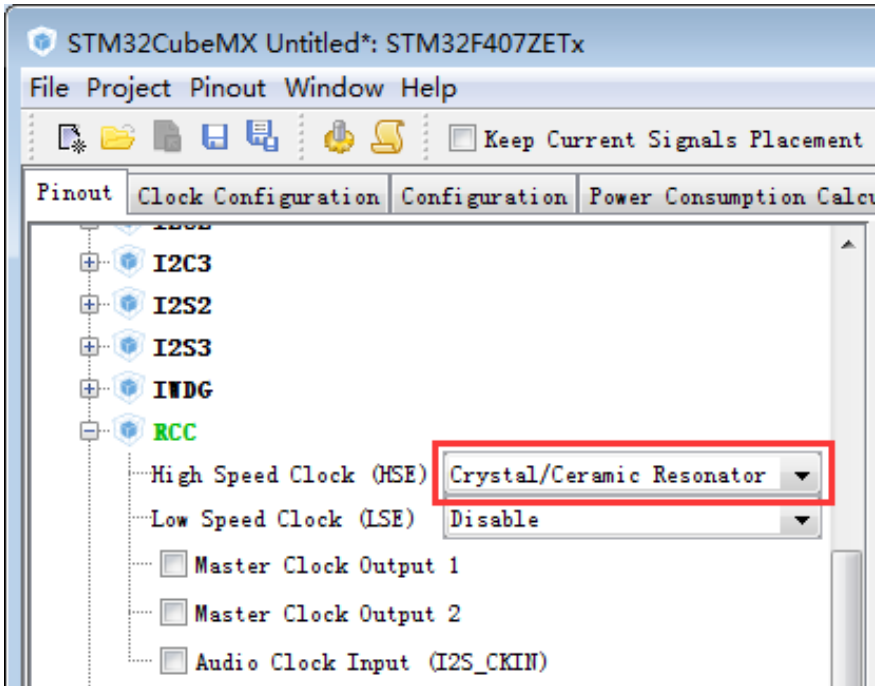
STM32Cube 学习之六：时钟树配置

假设已经安装好 STM32CubeMX 和 STM32CubeF4 支持包。

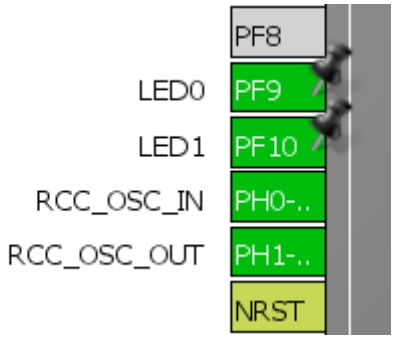
Step1.打开 STM32CubeMX，点击“New Project”，选择芯片型号，STM32F407ZETx。



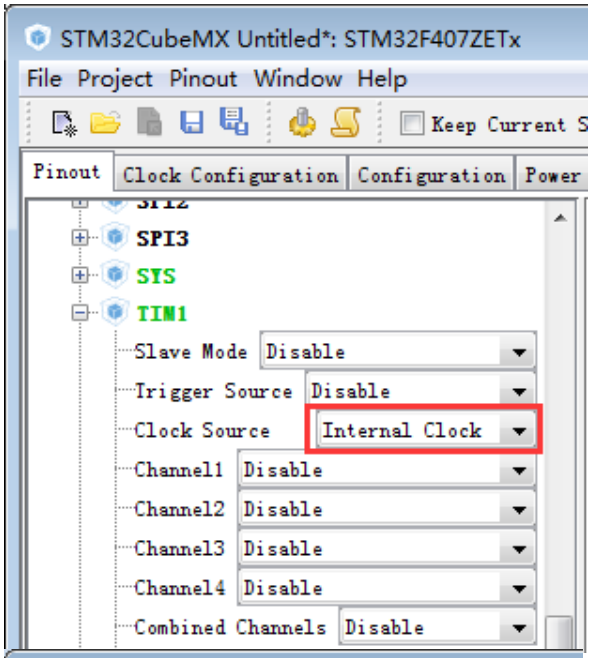
Step2. 在 Pinout 界面下配置使用外部晶振。

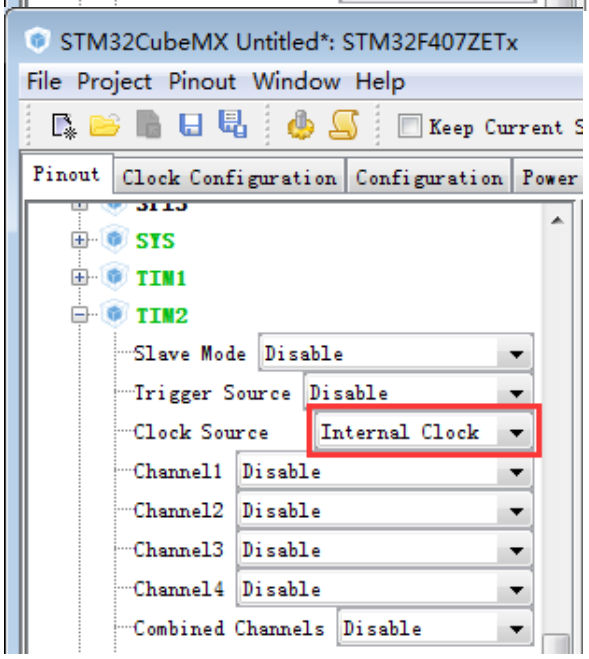


并配置 PF9，PF10 为输出模式，并输入用户标签 LED0，LED1。



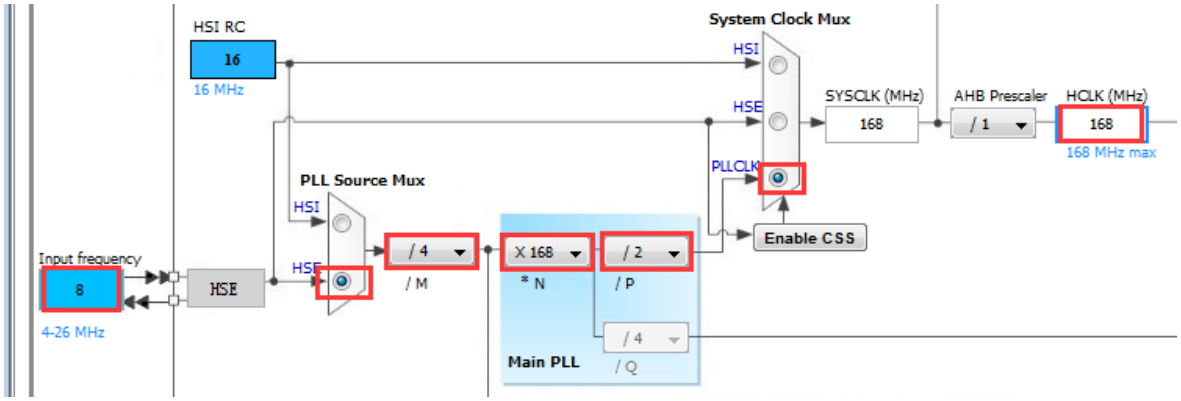
Step3.配置 TIM1，TIM2，都使用内部时钟源。



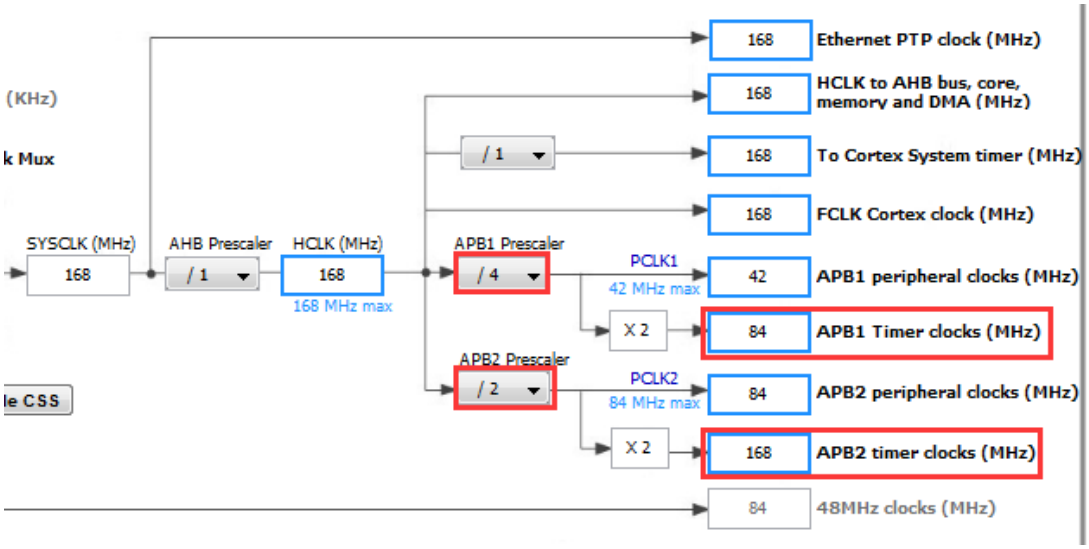


Step4.配置时钟树，使用外部 8M 晶振作为 PLL 输入，并使用 PLL 输出作为系统时钟。

根据下图配置，得到系统时钟为 168MHz。

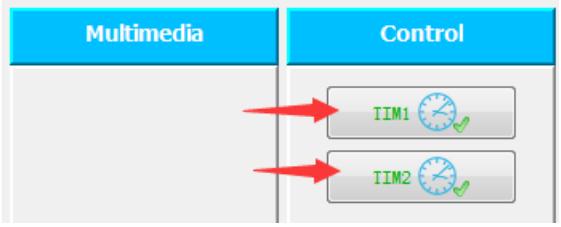


配置 APB1 和 APB2 上的定时器内部时钟源，分别为 84MHz 和 168MHz。

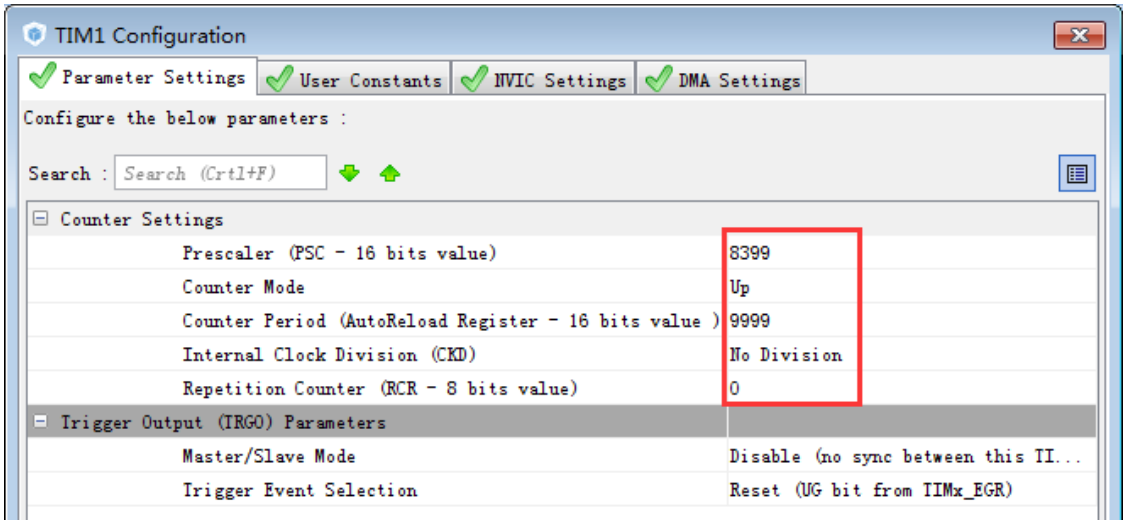


Step5.配置 TIM1 参数和 GPIO 的参数。

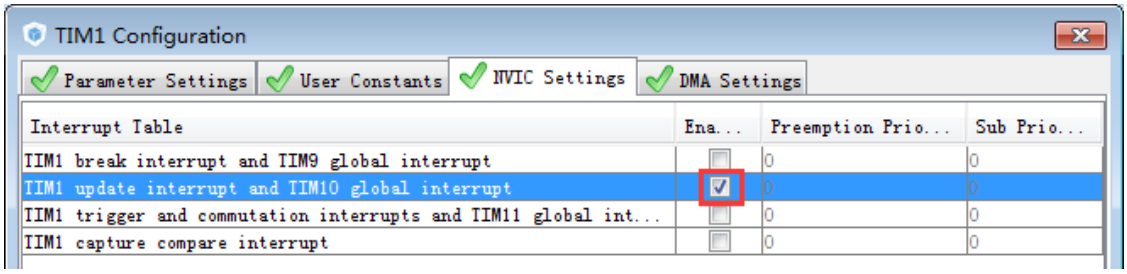
在 configuration 界面中点击 TIM1/ TIM2 按钮，可以进入参数配置界面。



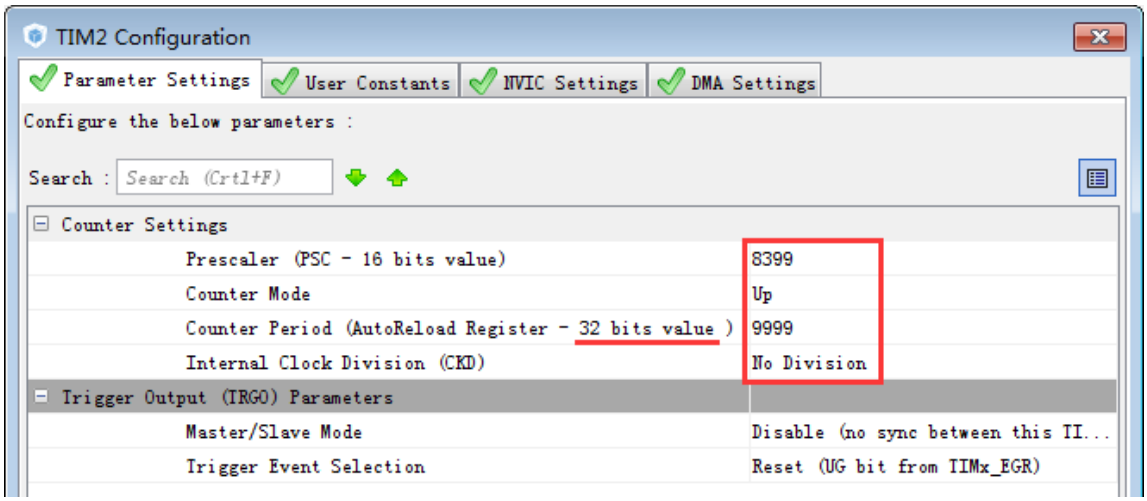
TIM1：在 Parameter Settings 页配置预分频系数为 8399，计数周期(自动加载值)为 9999，定时器溢出频率就是 168MHz/(8399+1)/(9999+1) = 2Hz。(为什么是 168MHz？后面解释。)



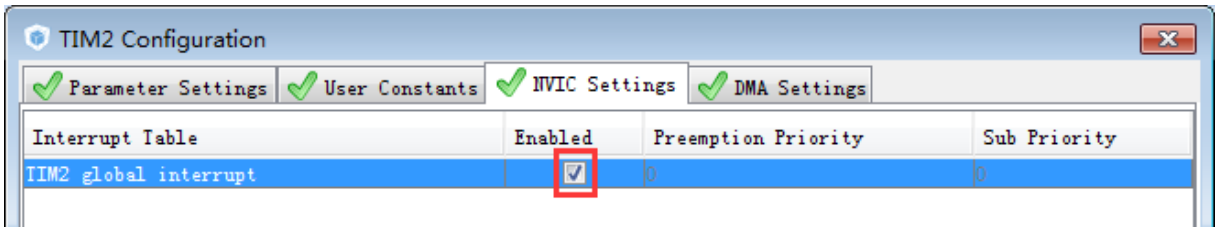
在 NVIC 页面使能 TIM1 的更新中断。



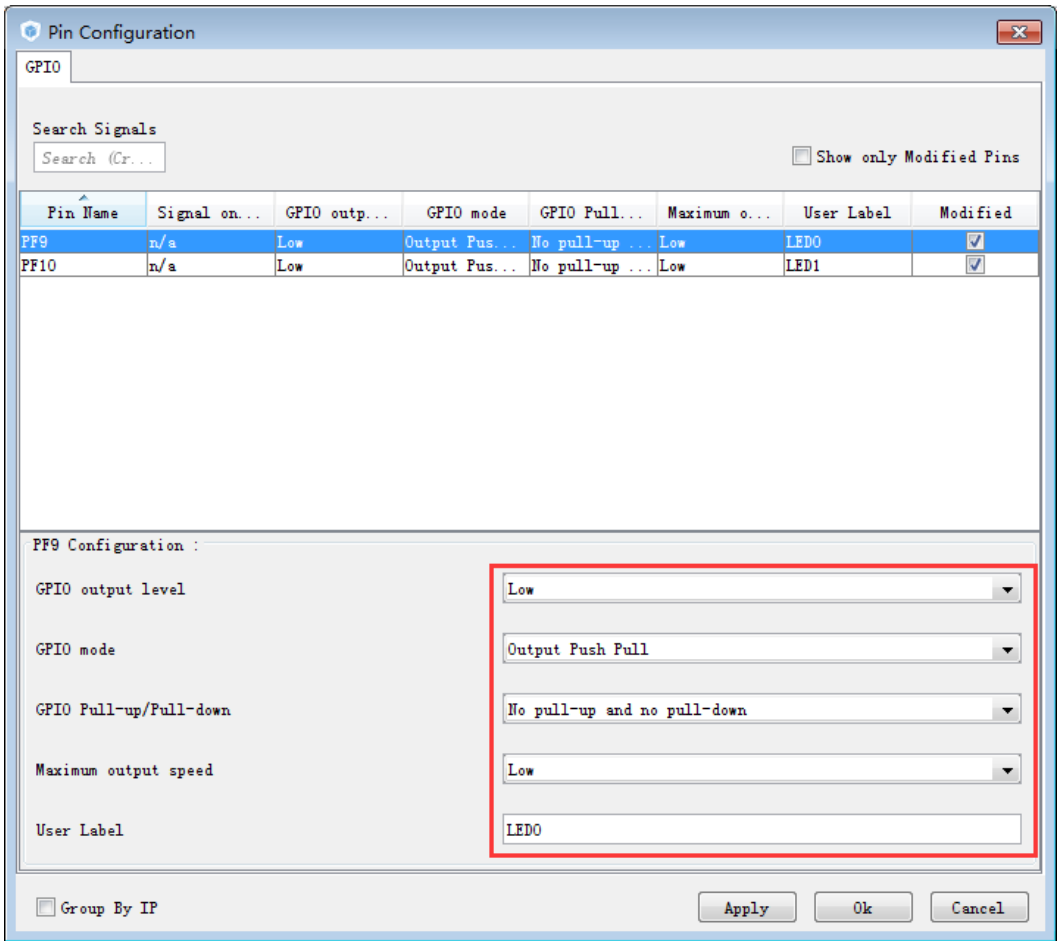
TIM2：在 Parameter Settings 页配置预分频系数为 8399，计数周期(自动加载值)为 9999，定时器溢出频率就是 84MHz/(8399+1)/(9999+1) = 1Hz。(为什么是 84MHz？后面解释。)



在 NVIC 页面使能 TIM2 的中断。

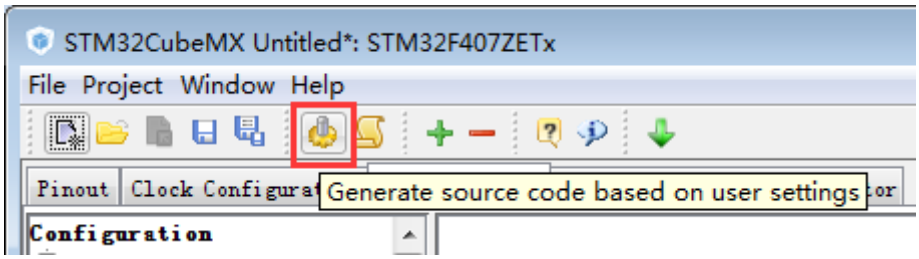


在 configuration 界面中点击 GPIO 按钮，配置 GPIO 的上拉电阻。在此 GPIO 配置默认即可。

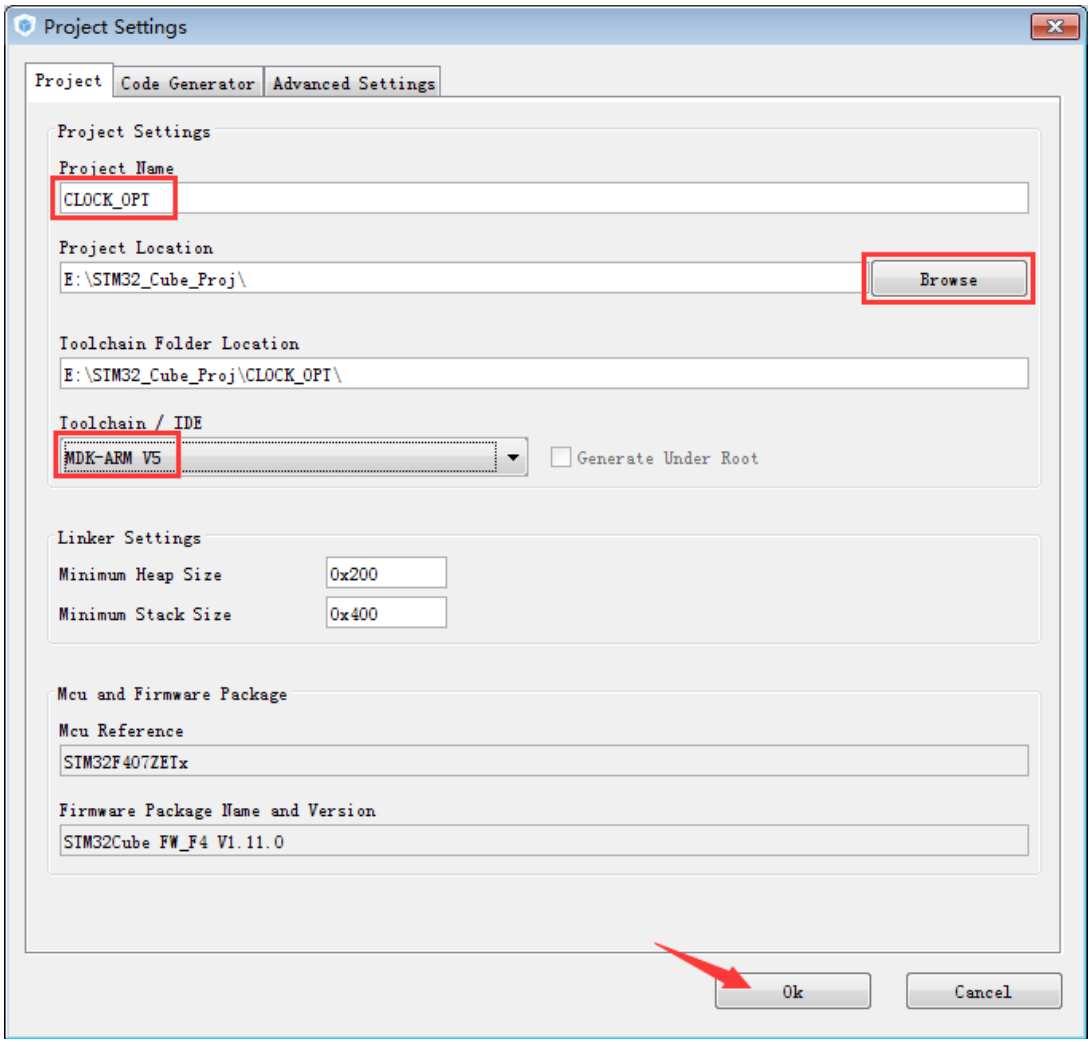


Step6.生成源代码。

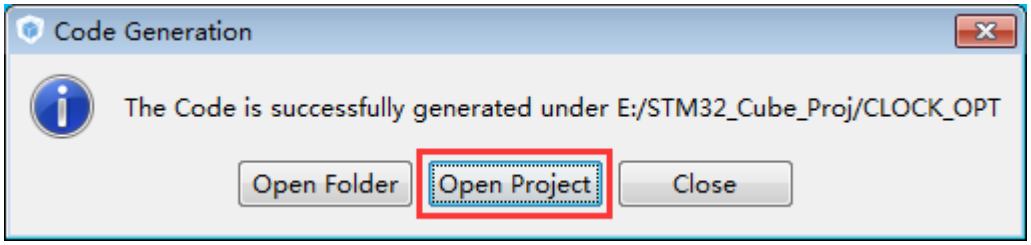
点击生成源代码按钮。



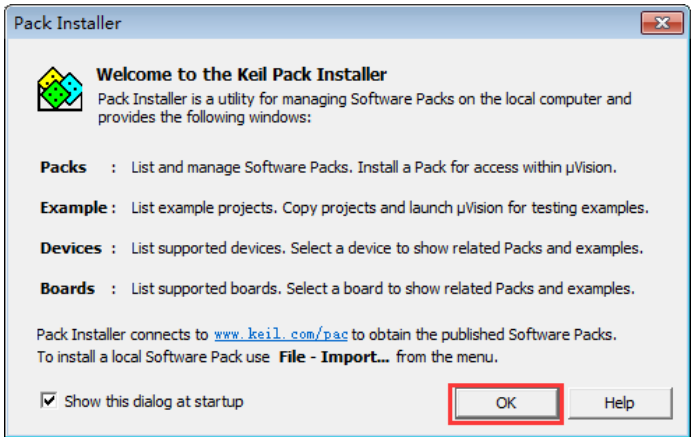
在设置界面中输入工程名，保存路径，工程 IDE 类型，点 OK 即可。



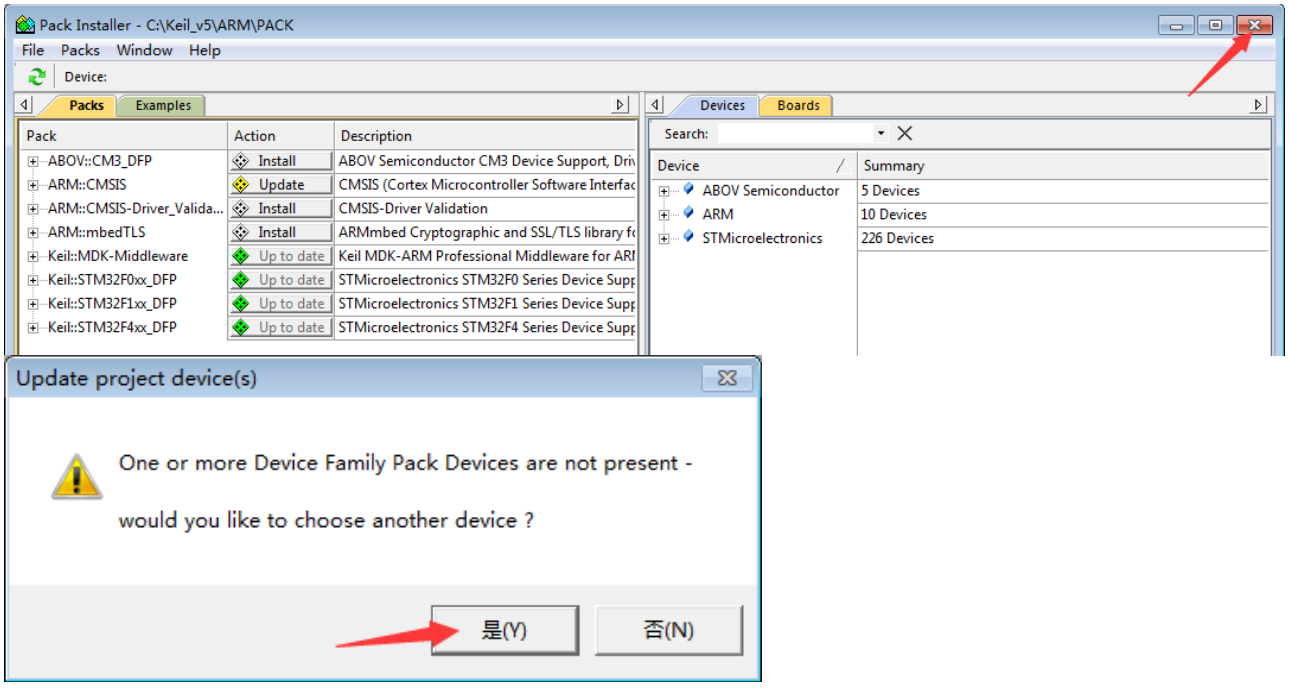
生成代码完成后可直接打开工程。



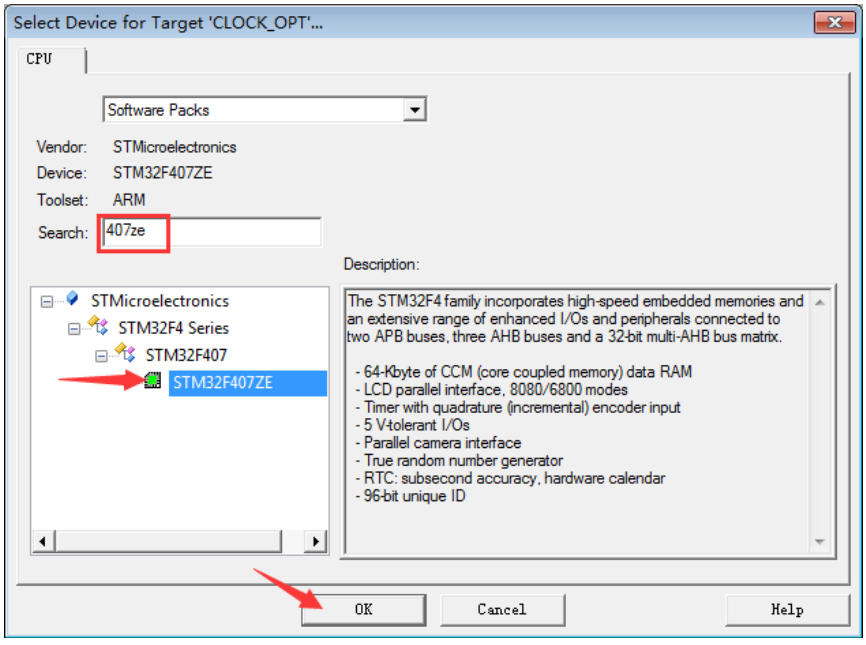
弹出如下对话框时，如果已经安装了 F4 的支持包，则点击 OK 关闭。如果没有安装，则点击界面中的 www.keil.com/...链接，找到芯片的支持包，然后安装。



关闭后面的界面。



点击“是”，然后选择芯片型号。可以在搜索框中输入关键字，加快选择速度。



Step7.添加功能代码。

在 main 文件/* USER CODE BEGIN 4 */和/* USER CODE END 4 */注释之间加入下面代码。

```
/* USER CODE BEGIN 4 */
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    if (TIM1 == htim->Instance) {
        HAL_GPIO_TogglePin(LED0_GPIO_Port, LED0_Pin);
    }
    if (TIM2 == htim->Instance) {
        HAL_GPIO_TogglePin(LED1_GPIO_Port, LED1_Pin);
    }
}
/* USER CODE END 4 */
```

在 main 函数的 while(1)之前启动 TIM1、TIM2 并使能它们的中断功能。

```
  /* USER CODE BEGIN 2 */
  HAL_TIM_Base_Start_IT(&htim1);
  HAL_TIM_Base_Start_IT(&htim2);
  /* USER CODE END 2 */
```

至此，完成整个工程。编译下载，现象就是 LED0 和 LED1 循环闪烁，LED0 亮 0.5 秒灭 0.5 秒，LED1 亮 1 秒灭 1 秒。

前面 Step5 留了一个问题，本例中为什么 TIM1 用到时钟是 168MHz，而 TIM2 用的是 84MHz？怎么才能知道某个定时器使用内部时钟源时，使用的是时钟树上 APB1 的分支还是 APB2 分支呢？

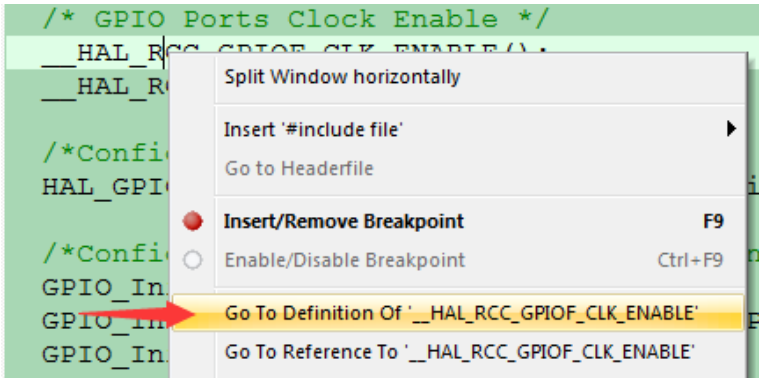
答案是，我也记不住 STM32 的时钟树是怎么分配的。但是有一个简单的方法，就是查看库文件中的宏定义。在任意一个用 CubeMX 生成的 MDK 工程中，打开 main.c 文件，找到任何一个使能外设时钟的语句，比如下图的使能 GPIO 时钟。

```
void MX_GPIO_Init(void)
{

  GPIO_InitTypeDef GPIO_InitStruct;

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOF_CLK_ENABLE();
  __HAL_RCC_GPIOH_CLK_ENABLE();
```

在该语句上点击右键，选择 Go To Definition...，找到该宏定义。



在该宏定义中，找到 RCC_开头的宏，继续右键 Go To Definition...

```
#define __HAL_RCC_GPIOF_CLK_ENABLE()   do { \
                                        __IO uint32_t tmpreg = 0x00U; \
                                        SET_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIOFEN);\
                                        /* Delay after an RCC peripheral clock enabling */ \
                                        tmpreg = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIOFEN);\
                                        UNUSED(tmpreg); \
                                        } while(0)
```

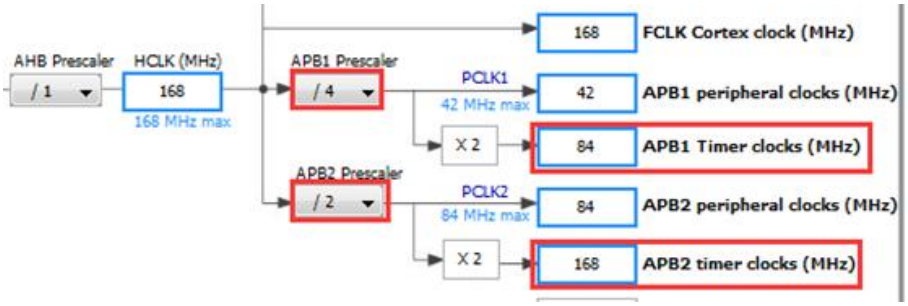
即可在 stm32f407xx.h 中找到。由下图可知，所有的 GPIO 时钟都来自 AHB1。

```
5076 /*********************  Bit definition for RCC_AHB1ENR register  ******
5077 #define  RCC_AHB1ENR_GPIOAEN                 ((uint32_t)0x00000001U)
5078 #define  RCC_AHB1ENR_GPIOBEN                 ((uint32_t)0x00000002U)
5079 #define  RCC_AHB1ENR_GPIOCEN                 ((uint32_t)0x00000004U)
5080 #define  RCC_AHB1ENR_GPIODEN                 ((uint32_t)0x00000008U)
5081 #define  RCC_AHB1ENR_GPIOEEN                 ((uint32_t)0x00000010U)
5082 #define  RCC_AHB1ENR_GPIOFEN                 ((uint32_t)0x00000020U)
5083 #define  RCC_AHB1ENR_GPIOGEN                 ((uint32_t)0x00000040U)
5084 #define  RCC_AHB1ENR_GPIOHEN                 ((uint32_t)0x00000080U)
5085 #define  RCC_AHB1ENR_GPIOIEN                 ((uint32_t)0x00000100U)
5086 #define  RCC_AHB1ENR_CRCEN                   ((uint32_t)0x00001000U)
5087 #define  RCC_AHB1ENR_BKPSRAMEN               ((uint32_t)0x00040000U)
5088 #define  RCC_AHB1ENR_CCMDATARAMEN            ((uint32_t)0x00100000U)
5089 #define  RCC_AHB1ENR_DMA1EN                  ((uint32_t)0x00200000U)
5090 #define  RCC_AHB1ENR_DMA2EN                  ((uint32_t)0x00400000U)
```

在该宏定义的上下查找，就可以找到和 TIM1，TIM2 相关的时钟使能宏定义，如下图：

```
5107
5108 /*********************  Bit definition for RCC_APB1ENR register  *****
5109 #define  RCC_APB1ENR_TIM2EN                  ((uint32_t)0x00000001U)
5110 #define  RCC_APB1ENR_TIM3EN                  ((uint32_t)0x00000002U)
5111 #define  RCC_APB1ENR_TIM4EN                  ((uint32_t)0x00000004U)
5112 #define  RCC_APB1ENR_TIM5EN                  ((uint32_t)0x00000008U)
5113 #define  RCC_APB1ENR_TIM6EN                  ((uint32_t)0x00000010U)
5114 #define  RCC_APB1ENR_TIM7EN                  ((uint32_t)0x00000020U)
5115 #define  RCC_APB1ENR_TIM12EN                 ((uint32_t)0x00000040U)
5116 #define  RCC_APB1ENR_TIM13EN                 ((uint32_t)0x00000080U)
5117 #define  RCC_APB1ENR_TIM14EN                 ((uint32_t)0x00000100U)
5118 #define  RCC_APB1ENR_WWDGEN                  ((uint32_t)0x00000800U)
5119 #define  RCC_APB1ENR_SPI2EN                  ((uint32_t)0x00004000U)
5120 #define  RCC_APB1ENR_SPI3EN                  ((uint32_t)0x00008000U)
5121 #define  RCC_APB1ENR_USART2EN                ((uint32_t)0x00020000U)
5122 #define  RCC_APB1ENR_USART3EN                ((uint32_t)0x00040000U)
5123 #define  RCC_APB1ENR_UART4EN                 ((uint32_t)0x00080000U)
5124 #define  RCC_APB1ENR_UART5EN                 ((uint32_t)0x00100000U)
5125 #define  RCC_APB1ENR_I2C1EN                  ((uint32_t)0x00200000U)
5126 #define  RCC_APB1ENR_I2C2EN                  ((uint32_t)0x00400000U)
5127 #define  RCC_APB1ENR_I2C3EN                  ((uint32_t)0x00800000U)
5128 #define  RCC_APB1ENR_CAN1EN                  ((uint32_t)0x02000000U)
5129 #define  RCC_APB1ENR_CAN2EN                  ((uint32_t)0x04000000U)
5130 #define  RCC_APB1ENR_PWREN                   ((uint32_t)0x10000000U)
5131 #define  RCC_APB1ENR_DACEN                   ((uint32_t)0x20000000U)
5132
5133 /*********************  Bit definition for RCC_APB2ENR register  *****
5134 #define  RCC_APB2ENR_TIM1EN                  ((uint32_t)0x00000001U)
5135 #define  RCC_APB2ENR_TIM8EN                  ((uint32_t)0x00000002U)
5136 #define  RCC_APB2ENR_USART1EN                ((uint32_t)0x00000010U)
5137 #define  RCC_APB2ENR_USART6EN                ((uint32_t)0x00000020U)
5138 #define  RCC_APB2ENR_ADC1EN                  ((uint32_t)0x00000100U)
5139 #define  RCC_APB2ENR_ADC2EN                  ((uint32_t)0x00000200U)
5140 #define  RCC_APB2ENR_ADC3EN                  ((uint32_t)0x00000400U)
5141 #define  RCC_APB2ENR_SDIOEN                  ((uint32_t)0x00000800U)
5142 #define  RCC_APB2ENR_SPI1EN                  ((uint32_t)0x00001000U)
5143 #define  RCC_APB2ENR_SYSCFGEN                ((uint32_t)0x00004000U)
5144 #define  RCC_APB2ENR_TIM9EN                  ((uint32_t)0x00010000U)
5145 #define  RCC_APB2ENR_TIM10EN                 ((uint32_t)0x00020000U)
5146 #define  RCC_APB2ENR_TIM11EN                 ((uint32_t)0x00040000U)
5147 #define  RCC_APB2ENR_SPI5EN                  ((uint32_t)0x00100000U)
5148 #define  RCC_APB2ENR_SPI6EN                  ((uint32_t)0x00200000U)
5149
```

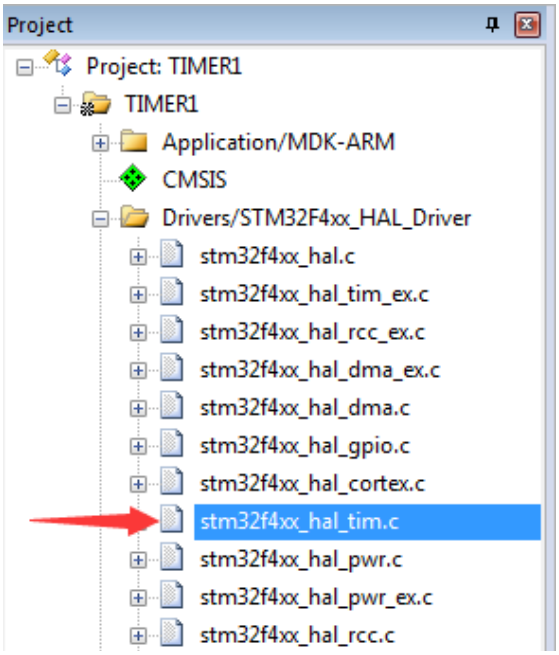
从图中可知，定时器 TIM2~TIM7 以及 TIM12~TIM14 的时钟来自 APB1，而 TIM1、TIM8、TIM9 的时钟来自 APB2。根据 Step4 的配置，APB1=84MHz，APB2=168MHz。



所以 Step5 配置 TIM1 时使用的是 168MHz，配置 TIM2 时使用的则是 84MHz。

特别说明：

因为 CubeMX 生成的 MDK 工程已经包含了配置中用到的外设相关文件，如下图：



打开 stm32f4xx_hal_tim.c，并点击右键，选择相应条目即可打开 stm32f4xx_hal_tim.h 文件，在以 HAL_开头的函数中，找到启动并使能定时器中断的函数，如下图：

```
1138 /* Time Base functions ************************************************
1139 HAL_StatusTypeDef HAL_TIM_Base_Init(TIM_HandleTypeDef *htim);
1140 HAL_StatusTypeDef HAL_TIM_Base_DeInit(TIM_HandleTypeDef *htim);
1141 void HAL_TIM_Base_MspInit(TIM_HandleTypeDef *htim);
1142 void HAL_TIM_Base_MspDeInit(TIM_HandleTypeDef *htim);
1143 /* Blocking mode: Polling */
1144 HAL_StatusTypeDef HAL_TIM_Base_Start(TIM_HandleTypeDef *htim);
1145 HAL_StatusTypeDef HAL_TIM_Base_Stop(TIM_HandleTypeDef *htim);
1146 /* Non-Blocking mode: Interrupt */
1147 HAL_StatusTypeDef HAL_TIM_Base_Start_IT(TIM_HandleTypeDef *htim);
1148 HAL_StatusTypeDef HAL_TIM_Base_Stop_IT(TIM_HandleTypeDef *htim);
1149 /* Non-Blocking mode: DMA */
1150 HAL_StatusTypeDef HAL_TIM_Base_Start_DMA(TIM_HandleTypeDef *htim, u
1151 HAL_StatusTypeDef HAL_TIM_Base_Stop_DMA(TIM_HandleTypeDef *htim);
```

定时器周期中断回调函数，在 1304 行。该函数的实现是用__weak 定义的，在用户文件中重定义即可，如 Step7 步骤所示。

```
1303 /* Callback in non blocking modes (Interrupt and DMA) **********
1304 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim);
1305 void HAL_TIM_OC_DelayElapsedCallback(TIM_HandleTypeDef *htim);
1306 void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim);
1307 void HAL_TIM_PWM_PulseFinishedCallback(TIM_HandleTypeDef *htim);
1308 void HAL_TIM_TriggerCallback(TIM_HandleTypeDef *htim);
1309 void HAL_TIM_ErrorCallback(TIM_HandleTypeDef *htim);
1310
```

另外，GPIO 操作的函数在 stm32f4xx_hal_gpio.c 和.h 文件中。

```
252 /* IO operation functions *****************************************
253 GPIO_PinState HAL_GPIO_ReadPin(GPIO_TypeDef* GPIOx, uint16_t GPIO
254 void HAL_GPIO_WritePin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin, GP
255 void HAL_GPIO_TogglePin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
256 HAL_StatusTypeDef HAL_GPIO_LockPin(GPIO_TypeDef* GPIOx, uint16_t
257 void HAL_GPIO_EXTI_IRQHandler(uint16_t GPIO_Pin);
258 void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin);
```

上面用到的 LED 端口宏定义，在 CubeMX 生成的工程目录\Inc\mxconstants.h 文件中。

```
41 #define LED0_Pin GPIO_PIN_9
42 #define LED0_GPIO_Port GPIOF
43 #define LED1_Pin GPIO_PIN_10
44 #define LED1_GPIO_Port GPIOF
```

官方例程请参考 stm32cubef4.zip 解压后 STM32Cube_FW_F4_V1.11.0\Projects\STM324xG_EVAL\Examples\RCC\RCC_ClockConfig 目录下的工程。

S.D.Lu 于 深圳
2016 年 8 月