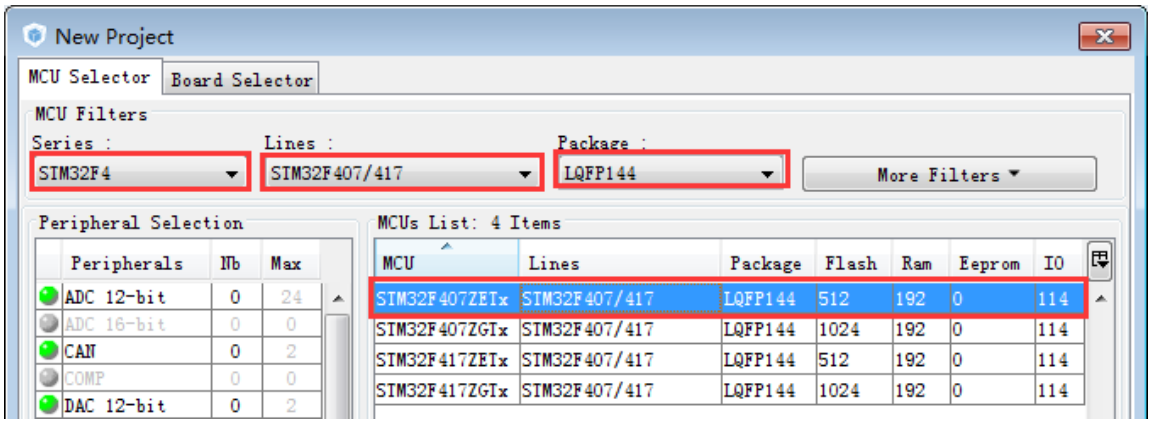
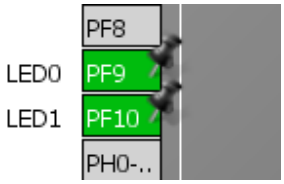
STM32Cube 学习之五：定时器中断

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

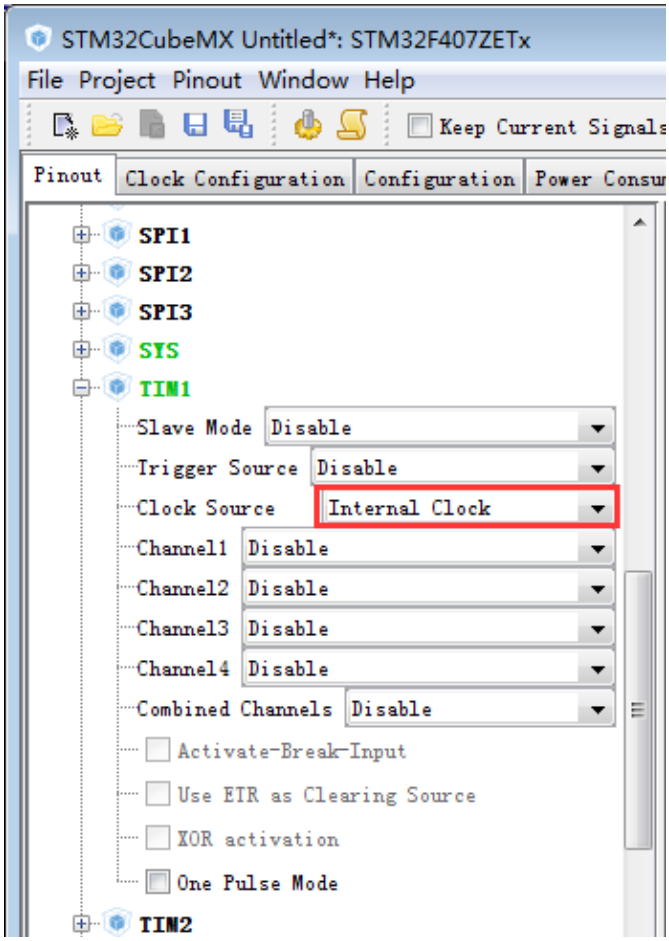
Step1.打开 STM32CubeMX，点击“New Project”，选择芯片型号，STM32F407ZETx。



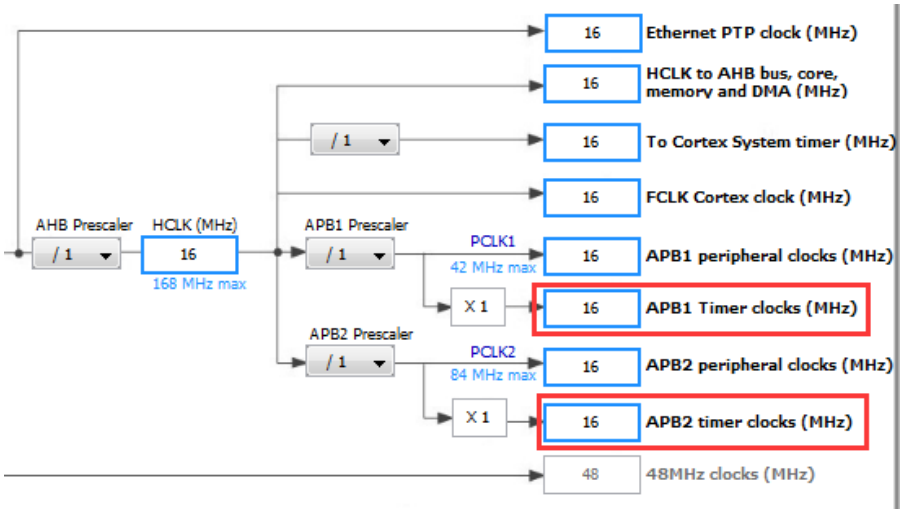
Step2.在 Pinout 界面下配置 PF9，PF10 为输出模式，并输入用户标签 LED0，LED1。



Step3.配置 TIM1，使用内部时钟源。



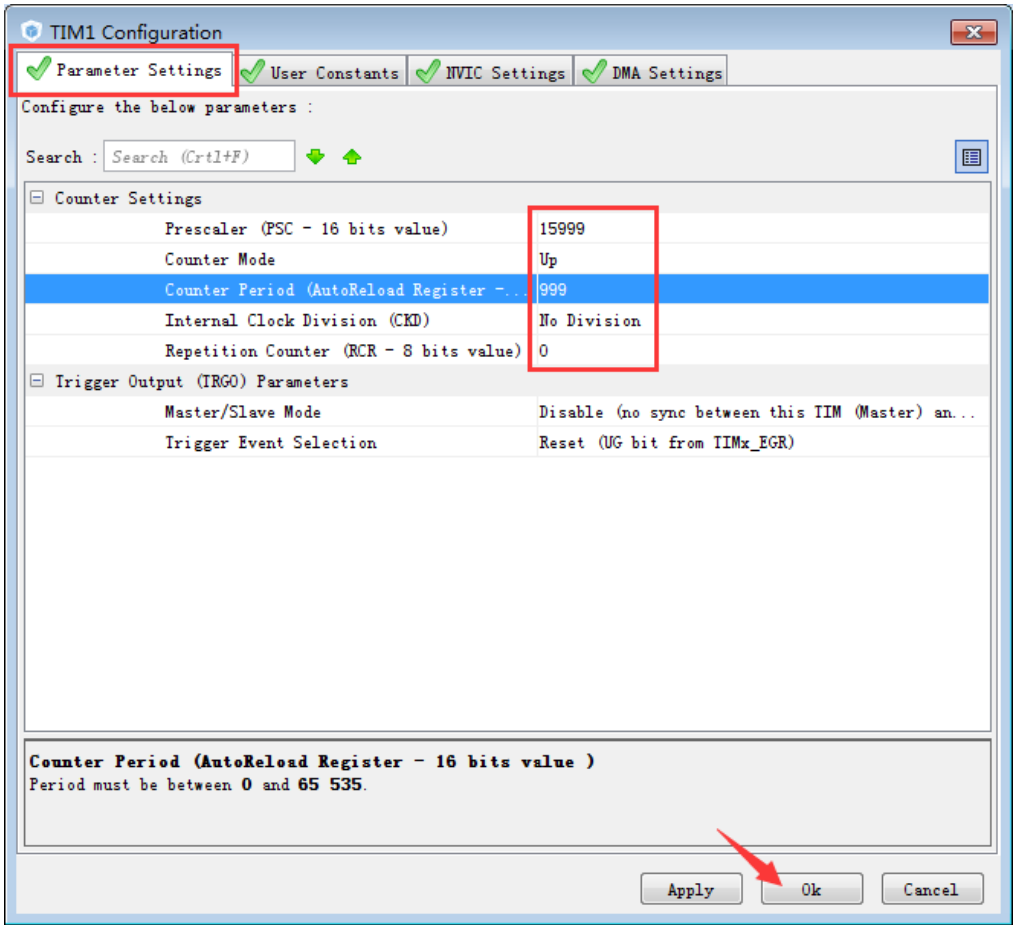
Step4.配置时钟树，在此使用默认值，16MHz。



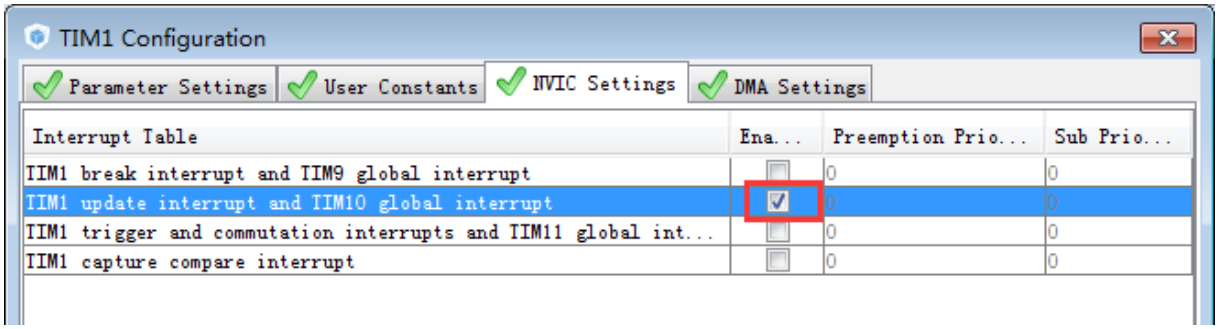
Step5.配置 TIM1 参数和 GPIO 的参数。

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

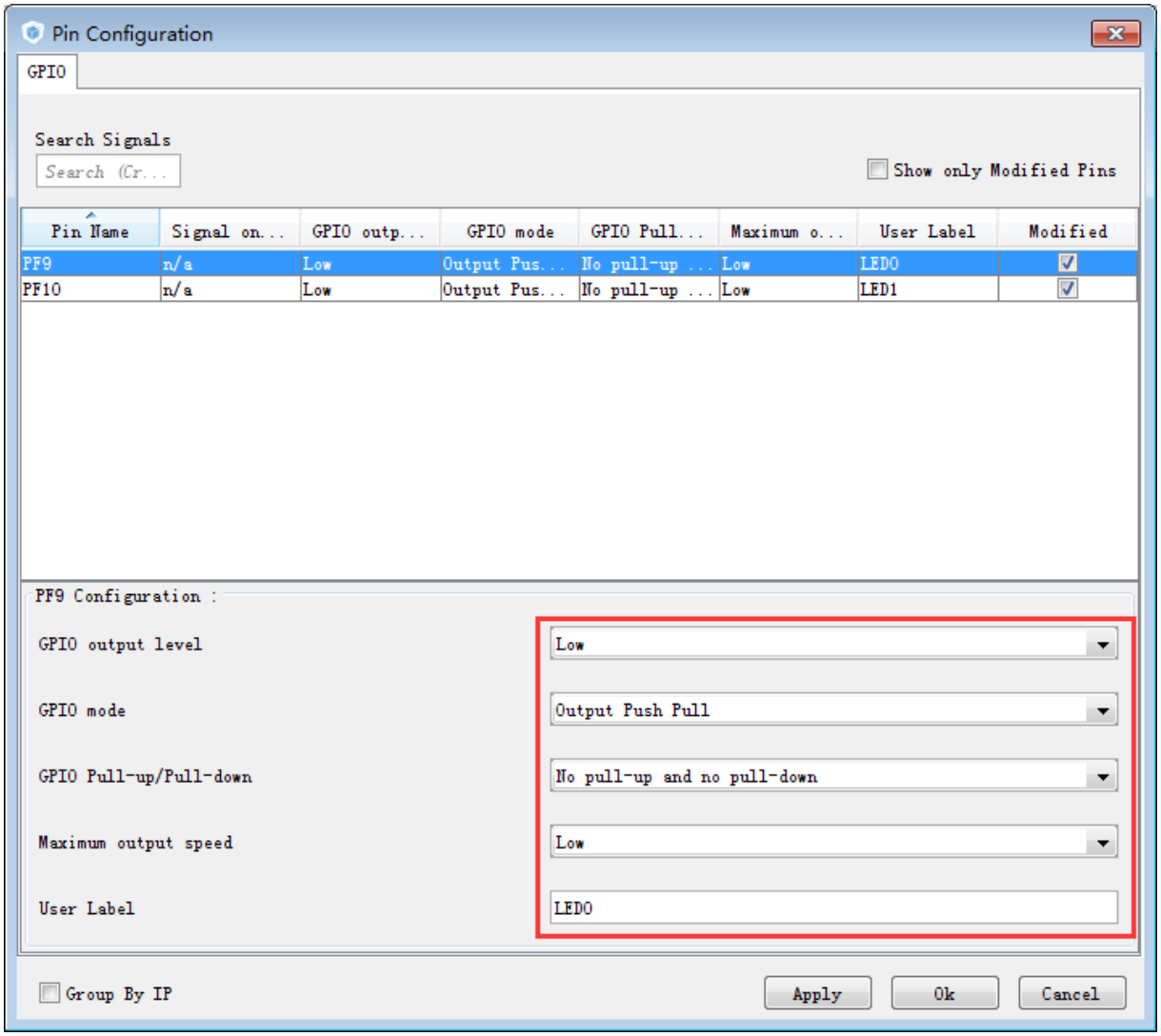
在 Parameter Settings 页配置预分频系数为 15999，计数周期(自动加载值)为 999，定时器溢出频率就是 16MHz/(15999+1)/(999+1) = 1Hz。



在 NVIC 页面使能 TIM1 的更新中断。

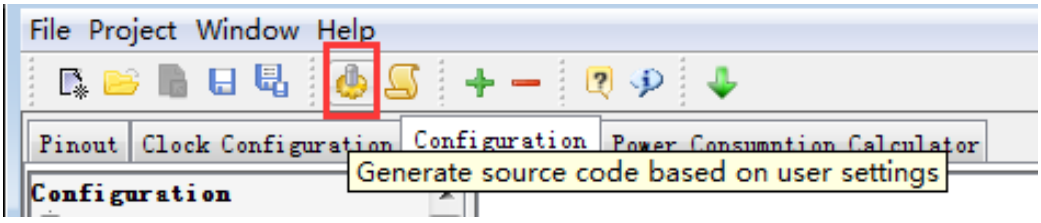


在 configuration 界面中点击 GPIO 按钮，配置 GPIO 的上拉电阻。在此 GPIO 配置默认即可。

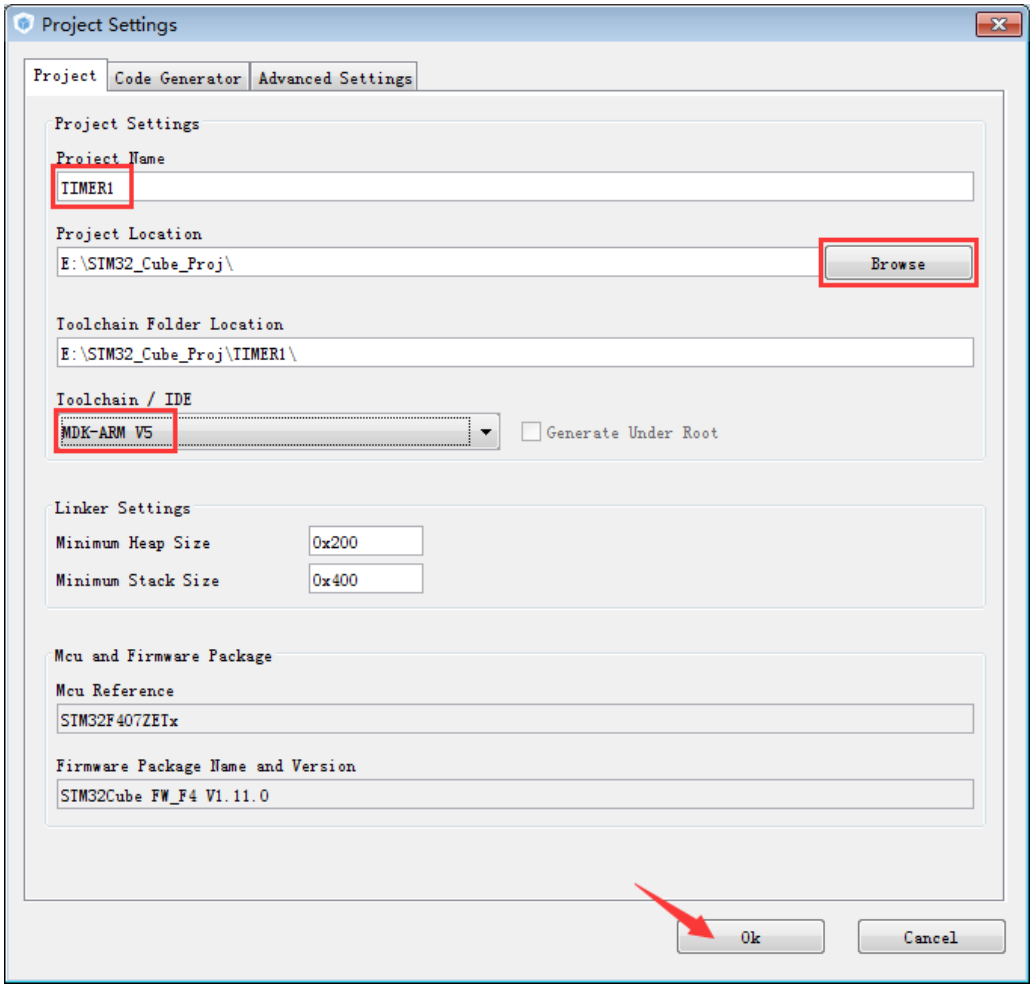


Step6.生成源代码。

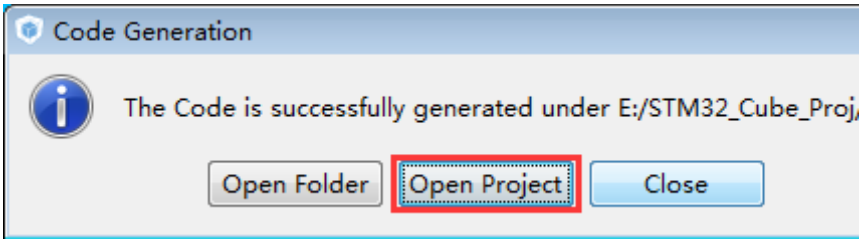
点击生成源代码按钮。



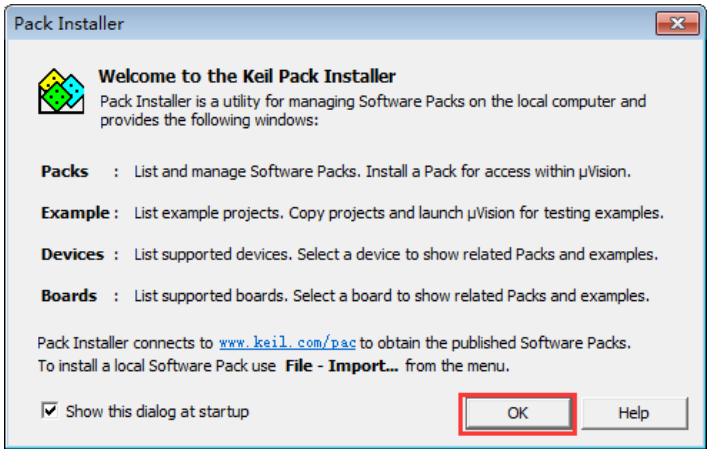
在设置界面中输入工程名，保存路径，工程 IDE 类型，点 OK 即可。



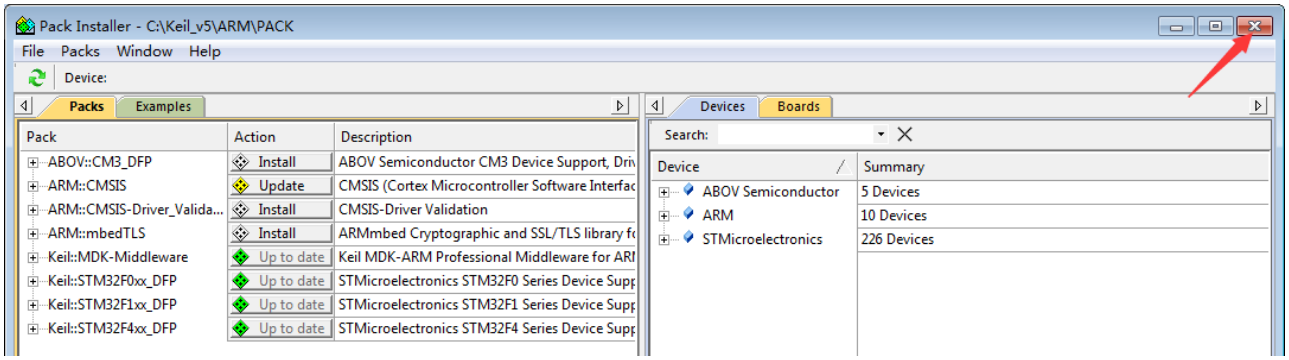
生成代码完成后可直接打开工程。



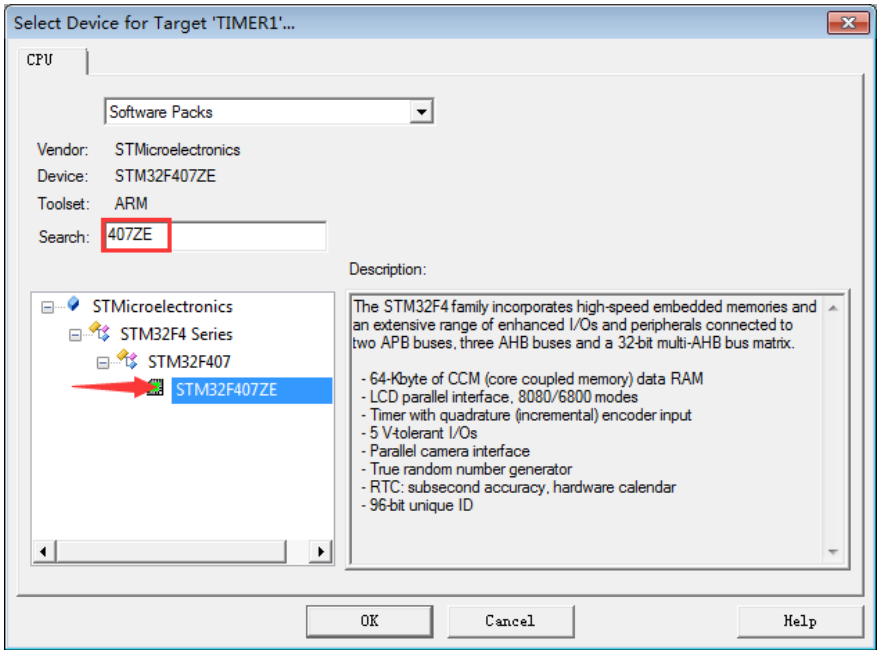
弹出如下对话框时，如果已经安装了 F4 的支持包，则点击 OK 关闭。如果没有安装，则点击界面中的 www.keil.com/...链接，找到芯片的支持包，然后安装。



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。

```
183 /* USER CODE BEGIN 4 */
184 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
185 {
186   if (TIM1 == htim->Instance) {
187       HAL_GPIO_TogglePin(LED0_GPIO_Port, LED0_Pin);
188       HAL_GPIO_TogglePin(LED1_GPIO_Port, LED1_Pin);
189   }
190 }
191 /* USER CODE END 4 */
```

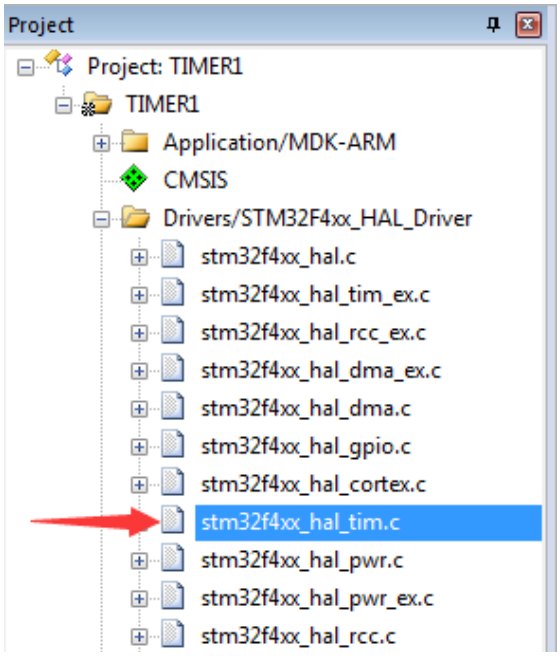
在 main 函数的 while(1)之前启动 TIM1 并使能其中断功能。

```
80
81   /* USER CODE BEGIN 2 */
82     HAL_TIM_Base_Start_IT(&htim1);
83   /* USER CODE END 2 */
84
85   /* Infinite loop */
86   /* USER CODE BEGIN WHILE */
87   while (1)
88   {
89   /* USER CODE END WHILE */
90
91   /* USER CODE BEGIN 3 */
92
93   }
```

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

特别说明：

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 HAL_开头的函数中，找到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
```

应注意，上图的这一组回调函数接口，是所有定时器共用的。所以，如果在实际应用中开启了多个定时器中断，则需要像 Step7 步骤中的代码一样，先判断是哪一个定时器的中断。

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\TIM\TIM_TimeBase 目录下的工程。

S.D.Lu 于 深圳
2016 年 8 月