STM32CubeMx 使用详解

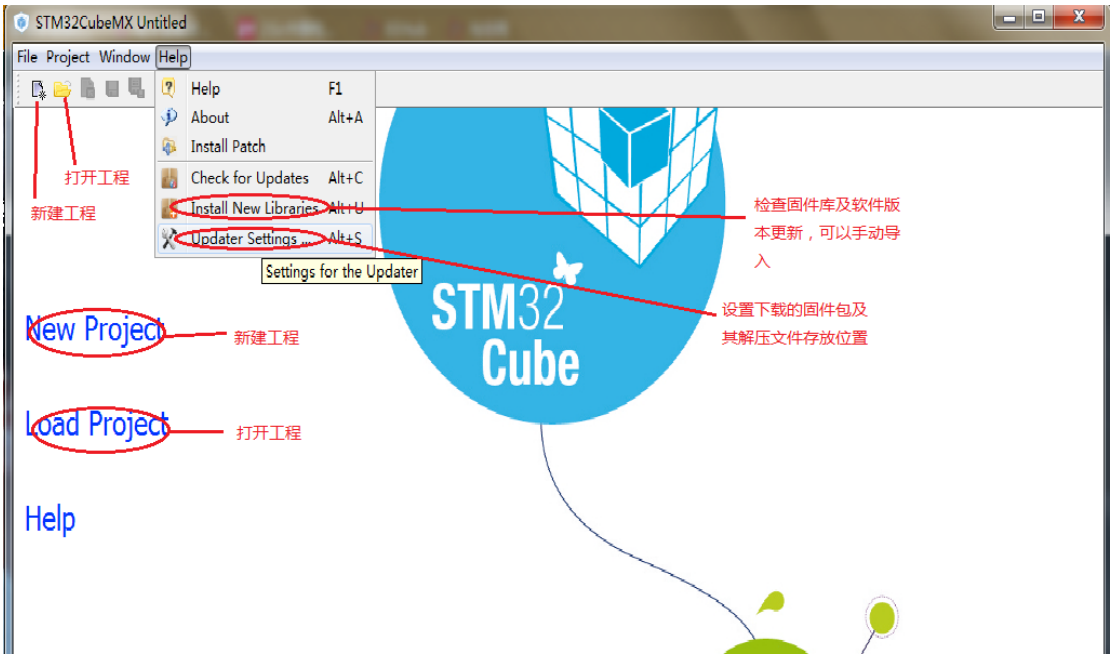
——cuyebiren

——意法半导体 STM32/STM8 社区

STM32CubeMx 是 STM32 系列单片机初始化代码工程生成工具。我们可以用它搜索选择满足我们需求的芯片，用它配置芯片外设引脚和功能，用它配置使用如 LWIP、FAT32、FreeRTOS 等第三方软件系统，还可以用它做功耗评估。STM32CubeMx 不仅能生成初始化代码工程，也能生成引脚配置信息的 pdf 和 txt 文档，方便查阅和设计原理图。——我相信 STM32CubeMx 的强大会使玩过它的人赞不绝口，毅然决然地放弃使用标准库，转而使用基于 HAL 库的它和 HAL 库。

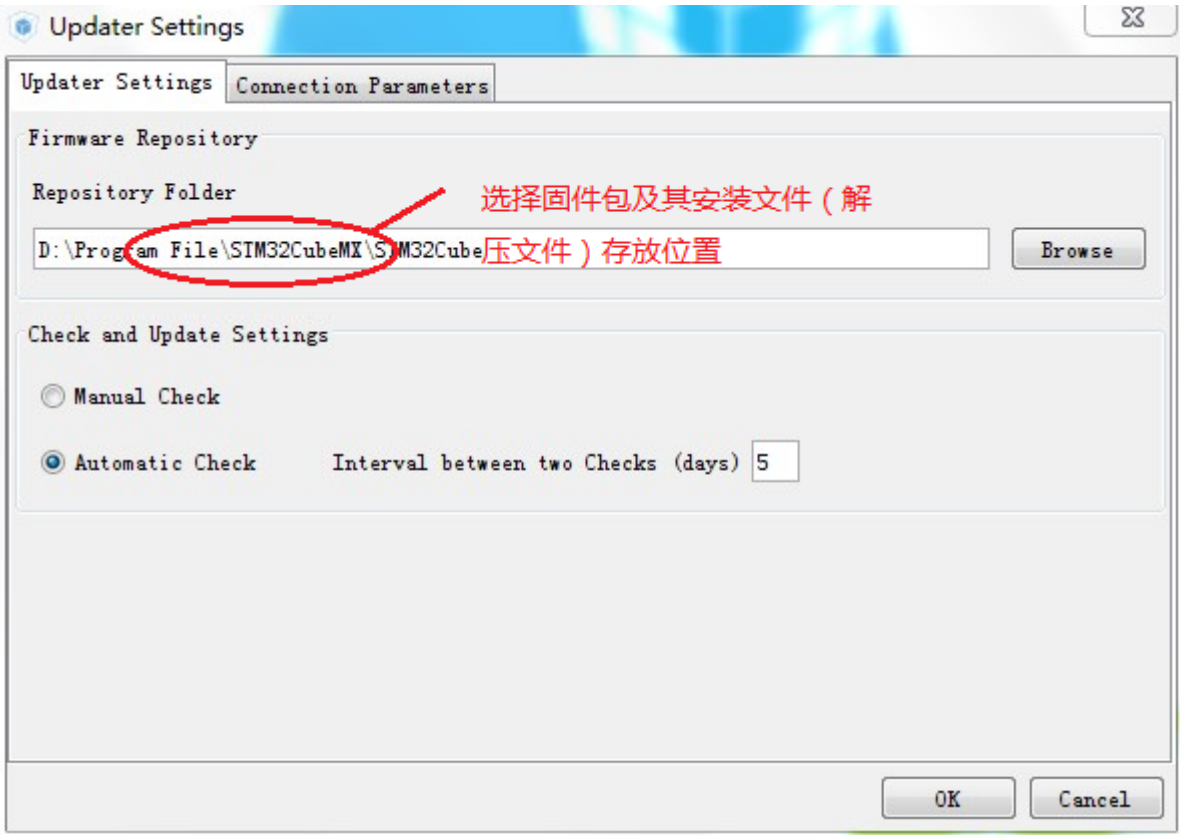
下面就开始介绍 STM32CubeMx 的使用：

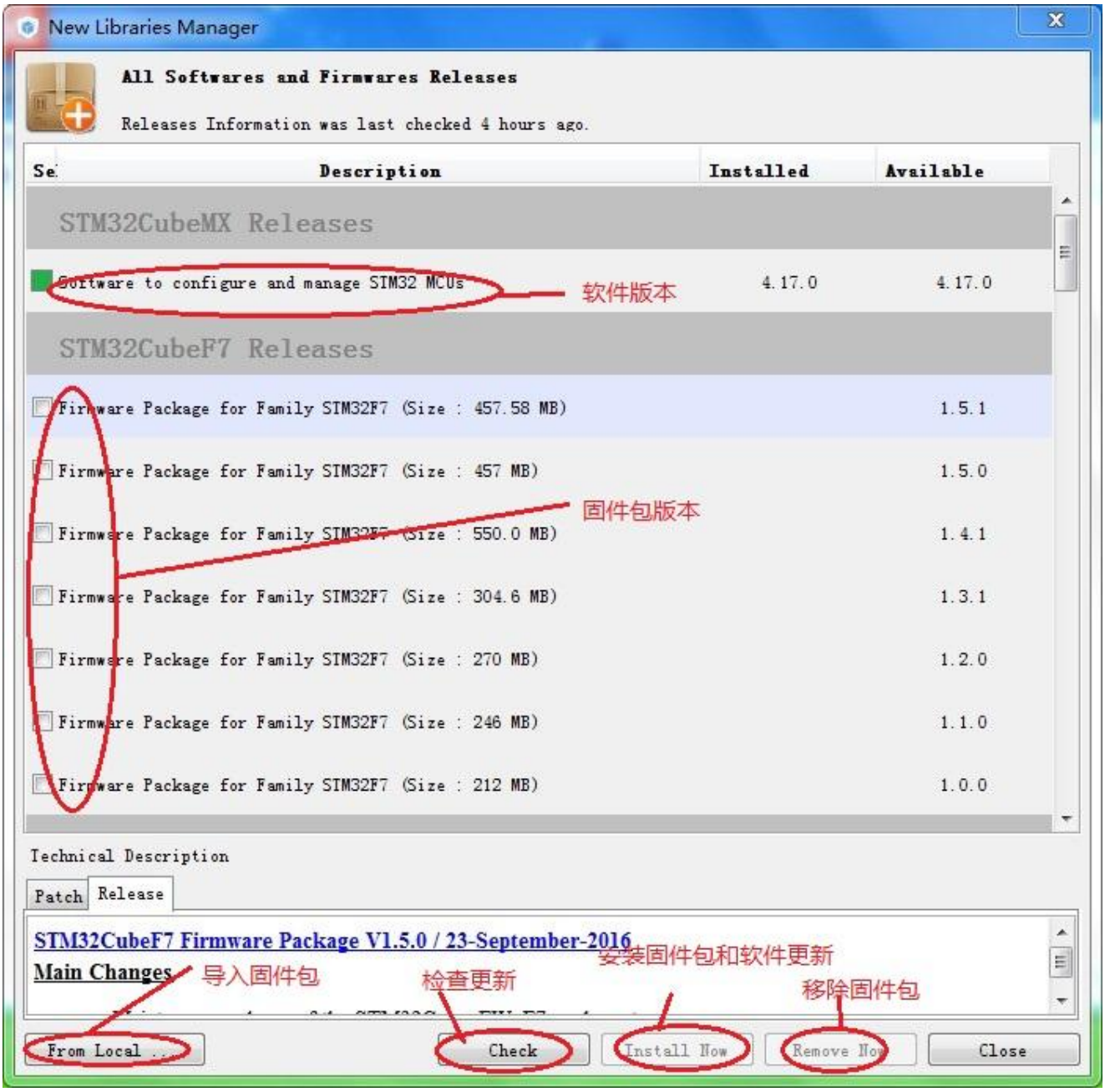
一、打开软件后的界面，如下。



这里主要介绍“Help”菜单。“Updater Setings”可以设置下载的固件库及其解压文件的存放位置，这样就可以找到软件下载的固件库到底存放到哪了。“Install New Libraries”可以检查并下载固件库和软件更新情况，以及历史版

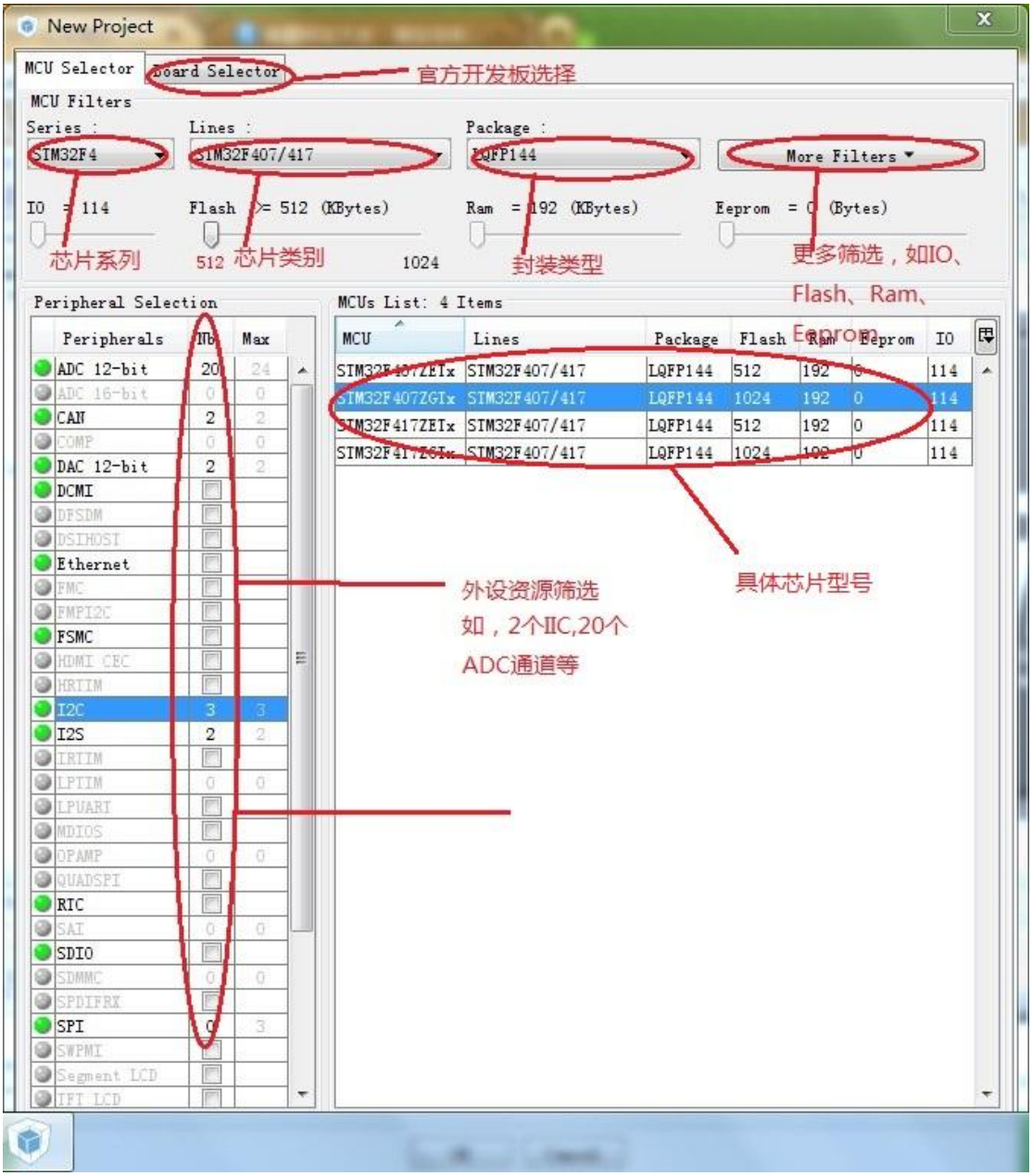
本，也可以手动导入固件库。



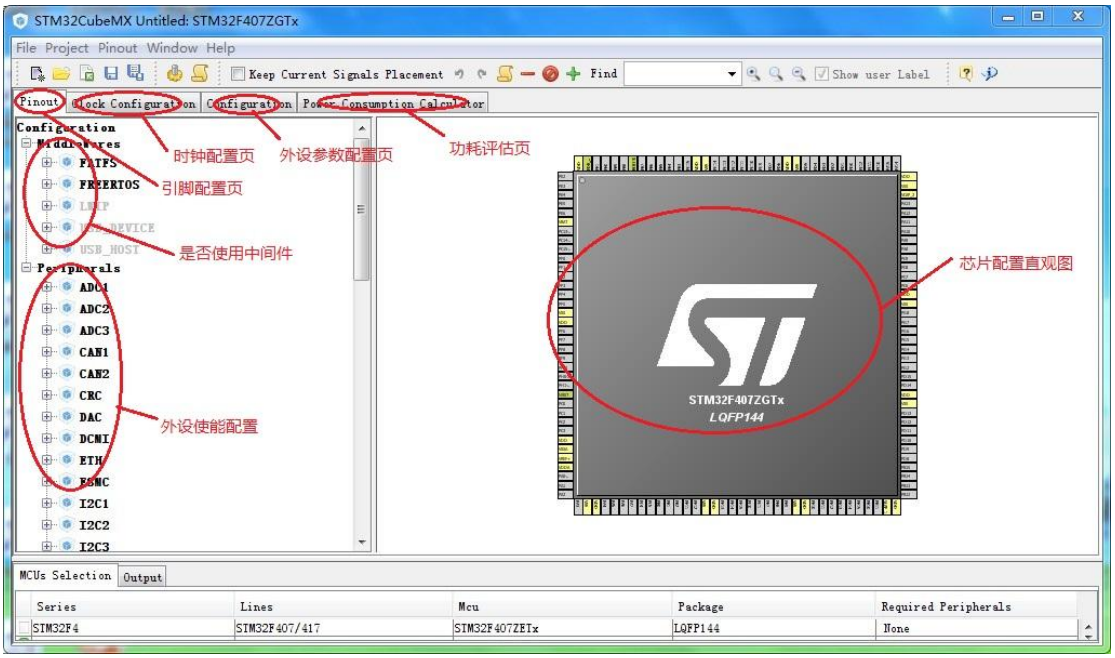


二、点击“New Projet”进入芯片选择界面。

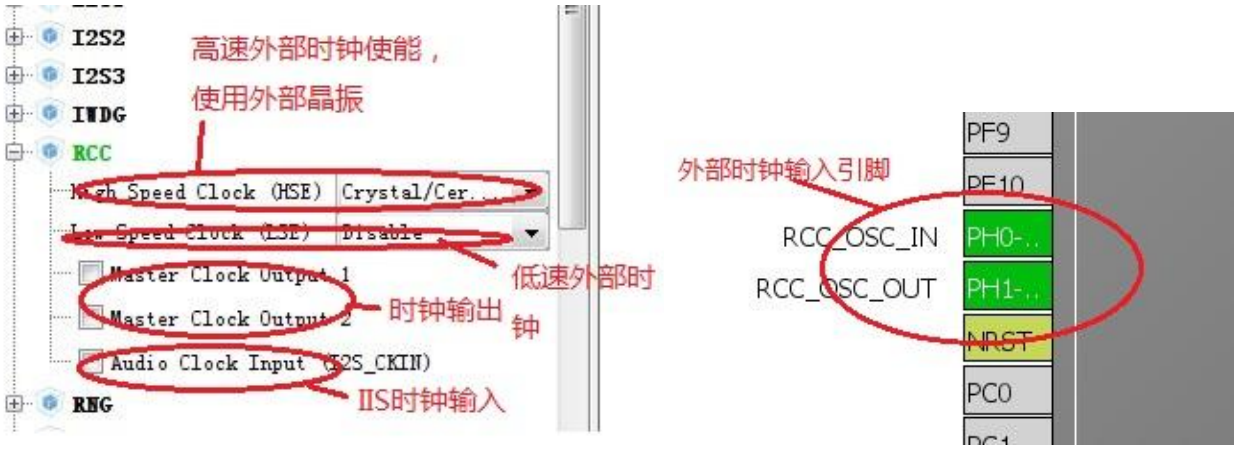
这里选择 STM32F407ZGTx（因为我的开发板是这个型号）。



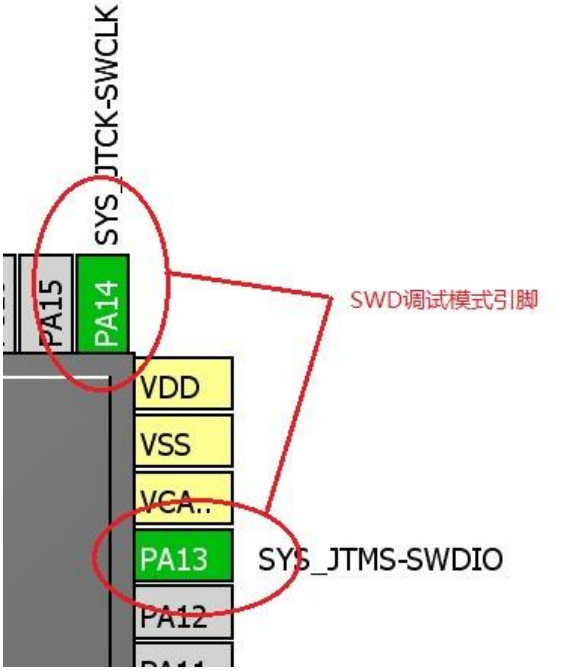
三、选择芯片型号双击或点“OK”，进入工程配置。


STM32CubeMX Untitled: STM32F407ZGTx
File Project Pinout Window Help
Keep Current Signals Placement
Find
Show user Label
Pinout
Clock Configuration
Configuration
Power Consumption Calculator
Configuration
Middlewares
FATFS
FREERTOS
LWIP
USB_DEVICE
USB_HOST
Peripherals
ADC1
ADC2
ADC3
CAN1
CAN2
CRC
DAC
DCMI
ETH
FSMC
I2C1
I2C2
I2C3
时钟配置页
外设参数配置页
功耗评估页
引脚配置页
是否使用中间件
外设使能配置
芯片配置直观图
STM32F407ZGTx
LQFP144
MCUs Selection
Output
Series
Lines
Mcu
Package
Required Peripherals
STM32F4
STM32F407/417
STM32F407ZETx
LQFP144
None

四、外设使能及引脚选择。

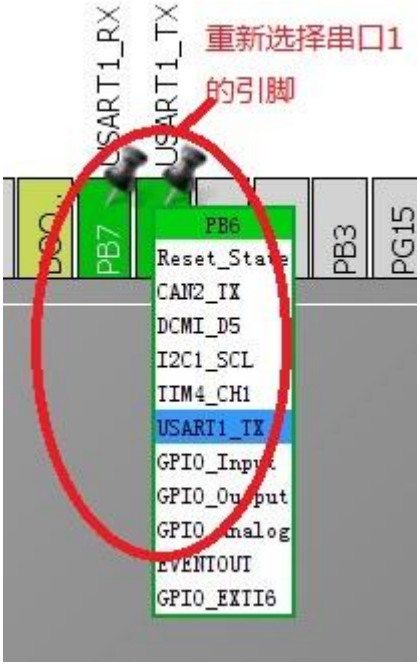

I2S2
I2S3
IWDG
RCC
High Speed Clock (HSE)
Crystal/Cer...
Low Speed Clock (LSE)
Disable
Master Clock Output 1
Master Clock Output 2
Audio Clock Input (I2S_CKIN)
RNG
高速外部时钟使能，
使用外部晶振
低速外部时钟
时钟输出
IIS时钟输入
外部时钟输入引脚
RCC_OSC_IN
RCC_OSC_OUT
PF9
PF10
PH0-..
PH1-..
NRST
PC0

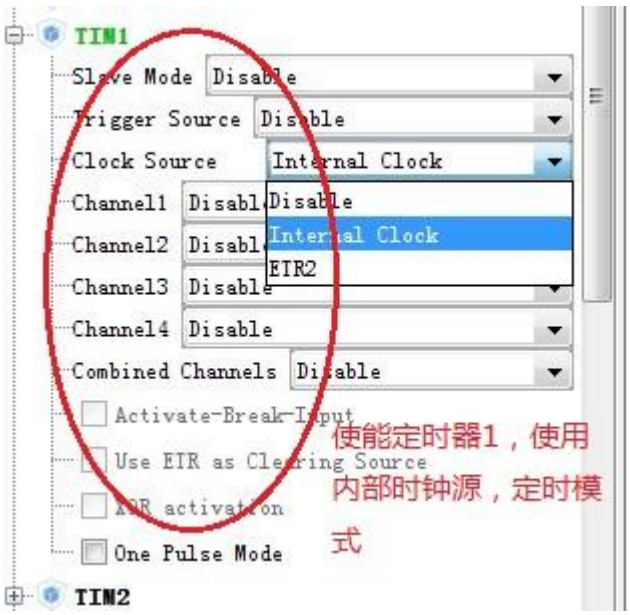

SPI2
SPI3
SYS
Debug
Serial Wire
System Wake-Up
Timebase Source
SysTick
TIM1
TIM2
使用SWD调试模式
HAL_Delay 延时函数所使用的硬件定时器

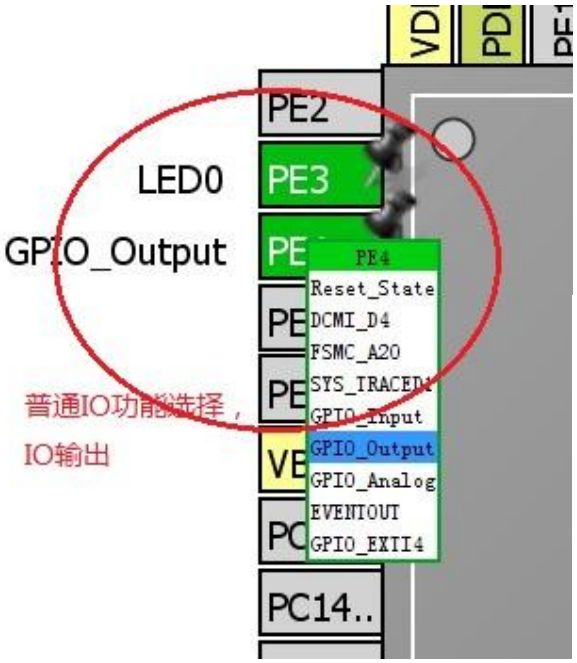

SYS_JTCK-SWCLK
PA15
PA14
SWD调试模式引脚
VDD
VSS
VCA..
PA13
SYS_JTMS-SWDIO
PA12

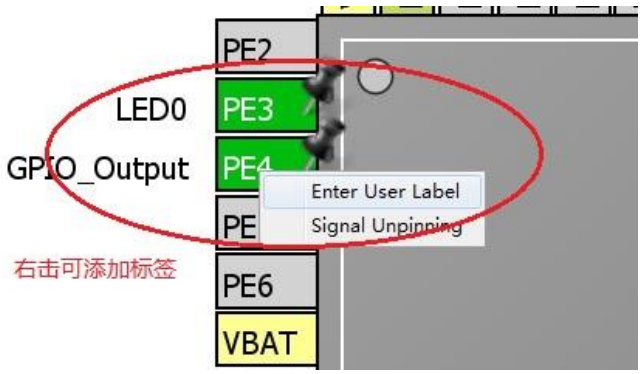

UART4
USART1
使能串口1，异步模式
Mode
Asynchronous
Disable
Asynchronous
Synchronous
Multiprocessor Communication
IrDA
LIN
SmartCard
MCUs Selection

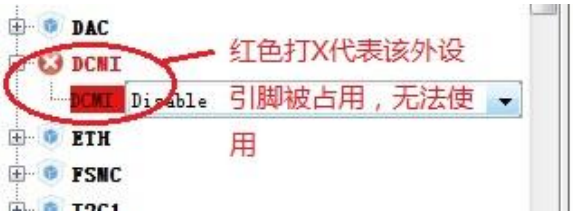

串口1所使用的引脚
PA11
PA10
USART1_RX
PA9
USART1_TX
PA8


USART1_RX
USART1_TX
重新选择串口1的引脚
PB7
PB3
PG15
PB6
Reset_State
CAN2_TX
DCMI_D5
I2C1_SCL
TIM4_CH1
USART1_TX
GPIO_Input
GPIO_Output
GPIO_Analog
EVENTOUT
GPIO_EXTI6

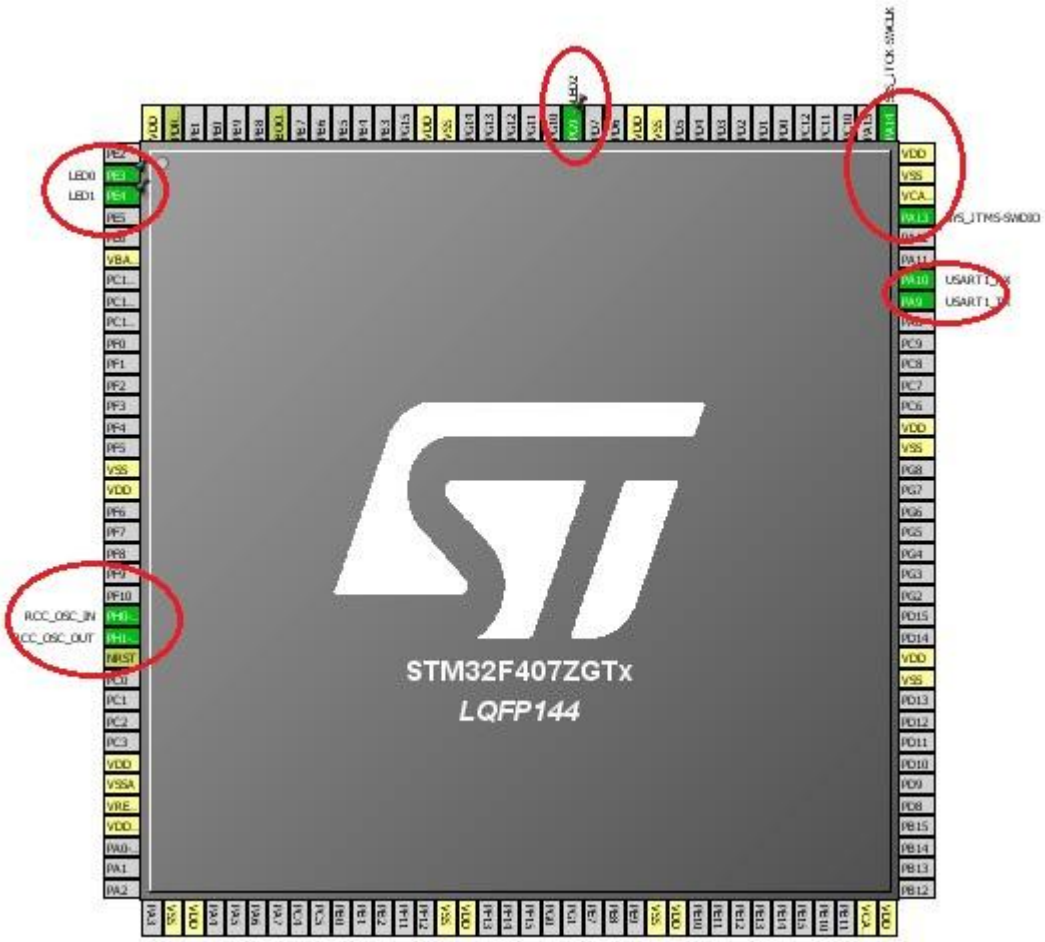

TIM1
Slave Mode
Disable
Trigger Source
Disable
Clock Source
Internal Clock
Channel1
Channel2
Channel3
Channel4
Disable
Internal Clock
ETR2
Combined Channels
Activate-Break-Input
Use ETR as Clearing Source
One Pulse Mode
使能定时器1，使用内部时钟源，定时模式
TIM2


LED0
GPIO_Output
PE2
PE3
PE4
PE4
Reset_State
DCMI_D4
FSMC_A20
SYS_IRACED1
GPIO_Input
GPIO_Output
GPIO_Analog
EVENTOUT
GPIO_EXTI4
PC14..
普通IO功能选择，
IO输出

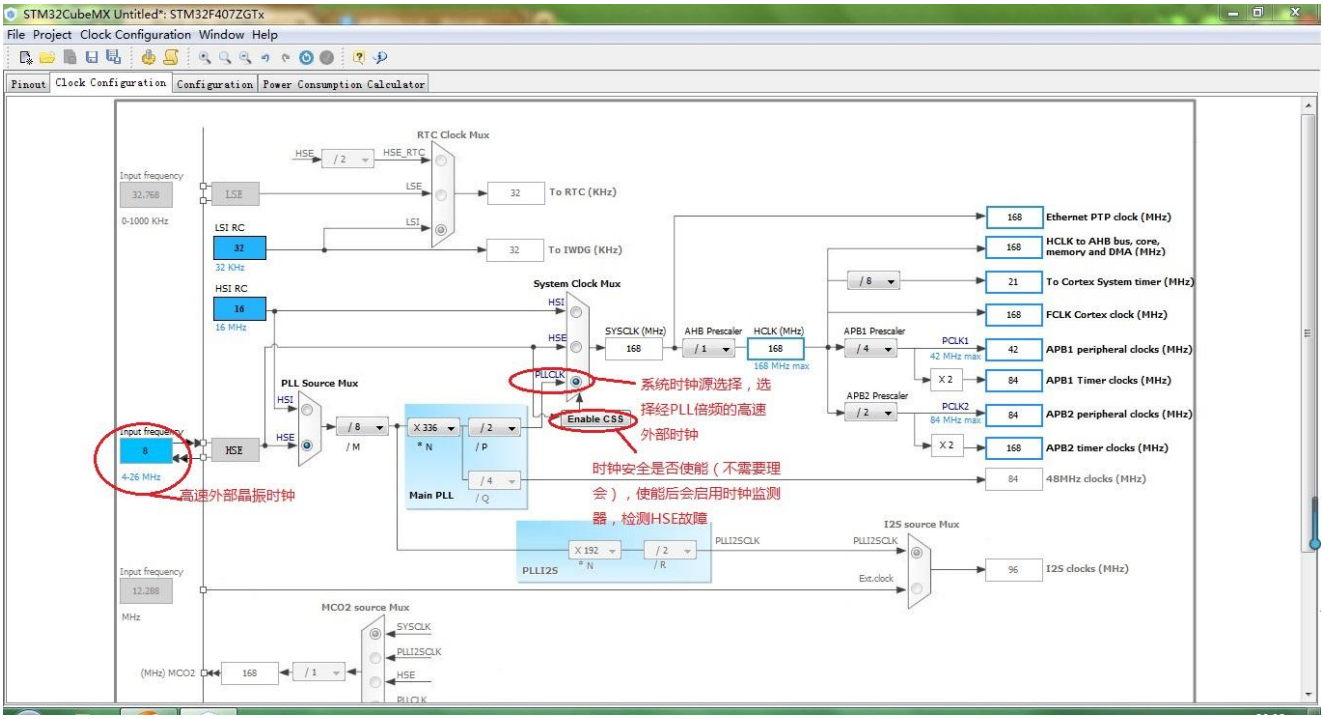

LED0
GPIO_Output
PE2
PE3
PE4
Enter User Label
Signal Unpinning
PE6
VBAT
右击可添加标签


DAC
DCMI
DCMI Disable
ETH
FSMC
红色打X代表该外设
引脚被占用，无法使
用

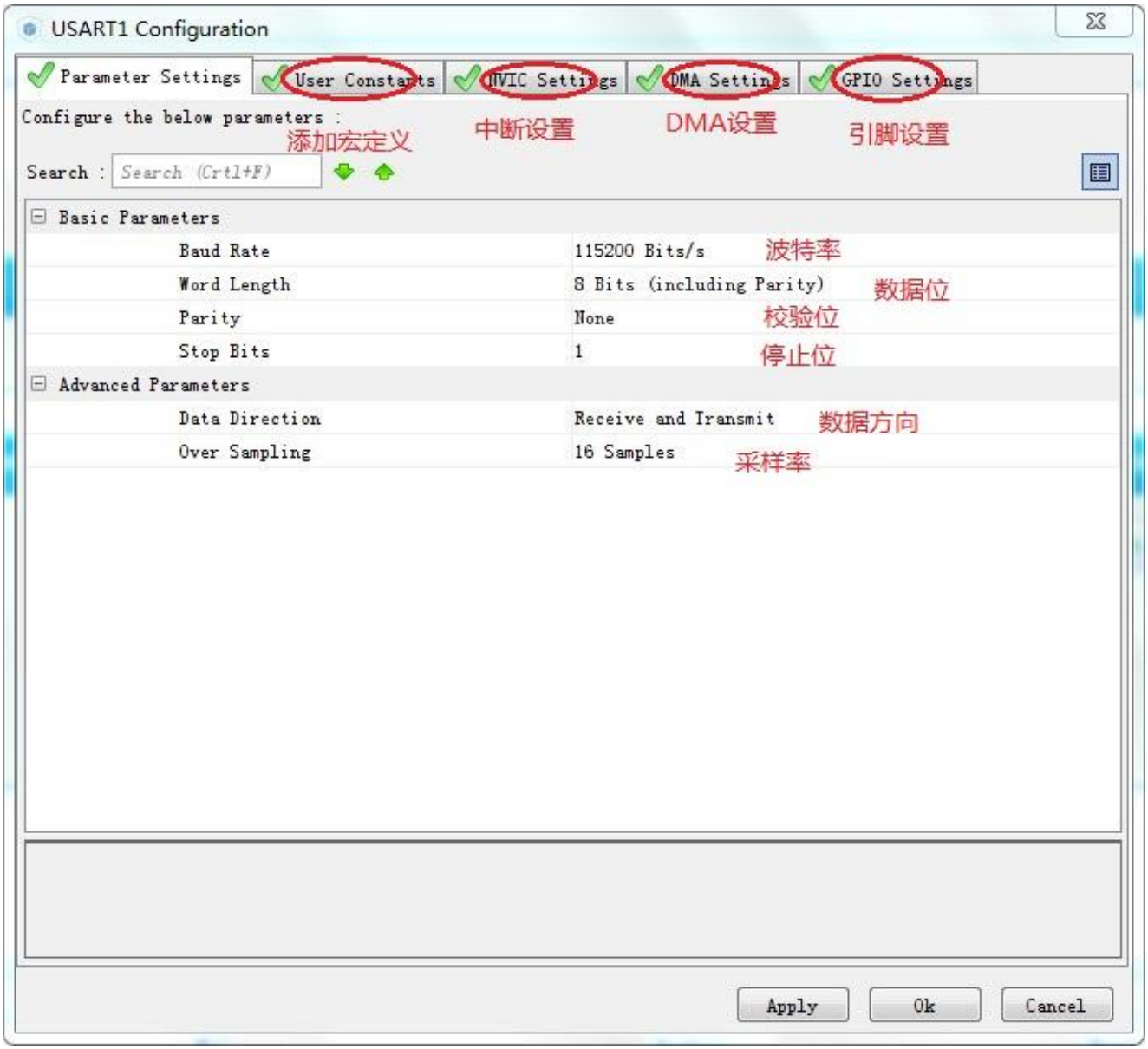

TIM6
TIM7
TIM8
Slave Mode Disable
Trigger Source Disable
Clock Source Disable
Channel1 Disable
Channel2 Disable
Channel3 Disable
Channel4 Disable
Combined Channels Disable
Activate Break Input
Use ITR as Clearing Source
XOR activation
One Pulse Mode
TIM9
黄色！，代表外设引脚
部分被占用，部分功能
无法实现

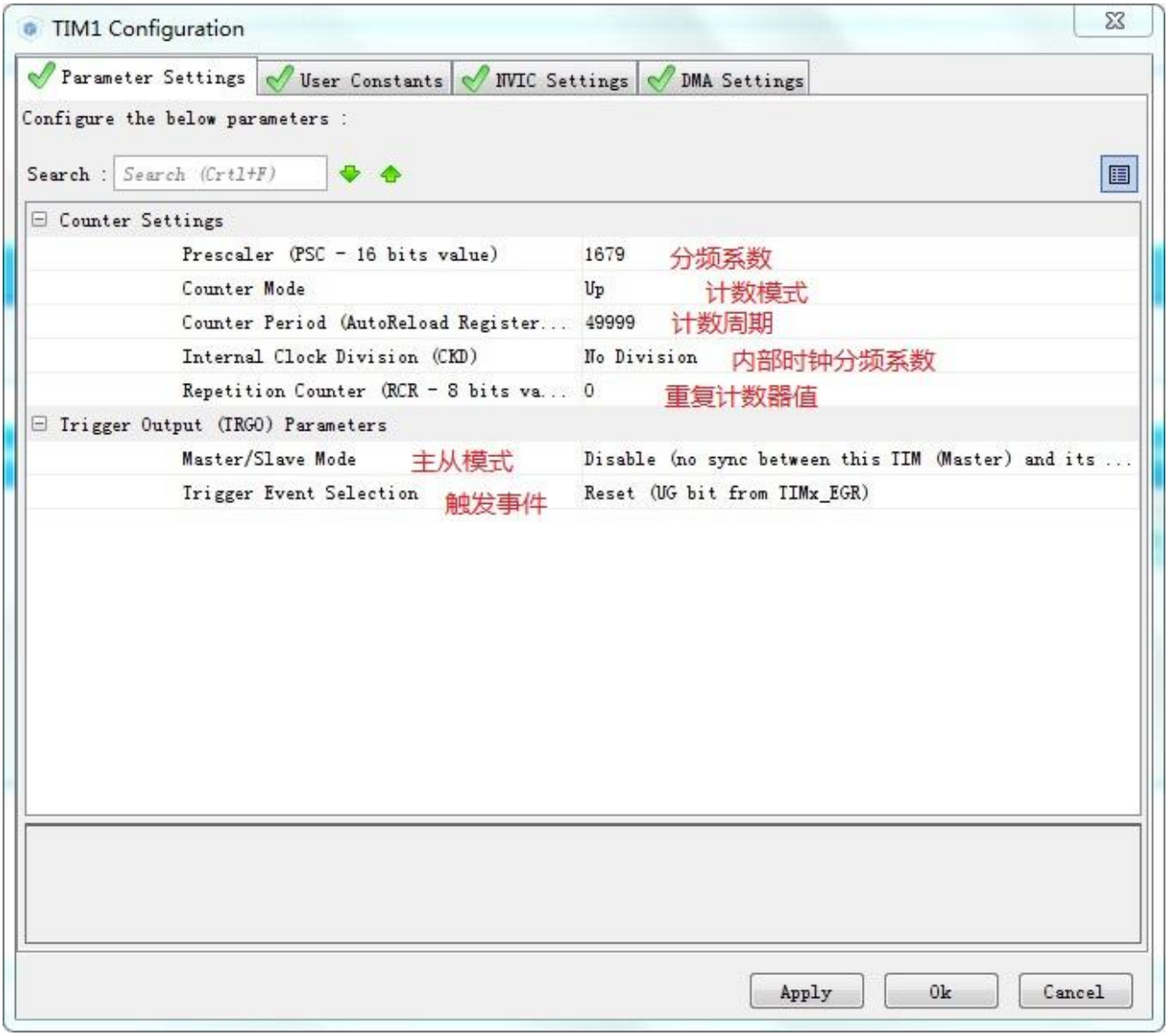

LED0
LED1
RCC_OSC_IN
RCC_OSC_OUT
USART1_TX
USART1_RX
SYS_JTMS-SWDIO
STM32F407ZGTx
LQFP144

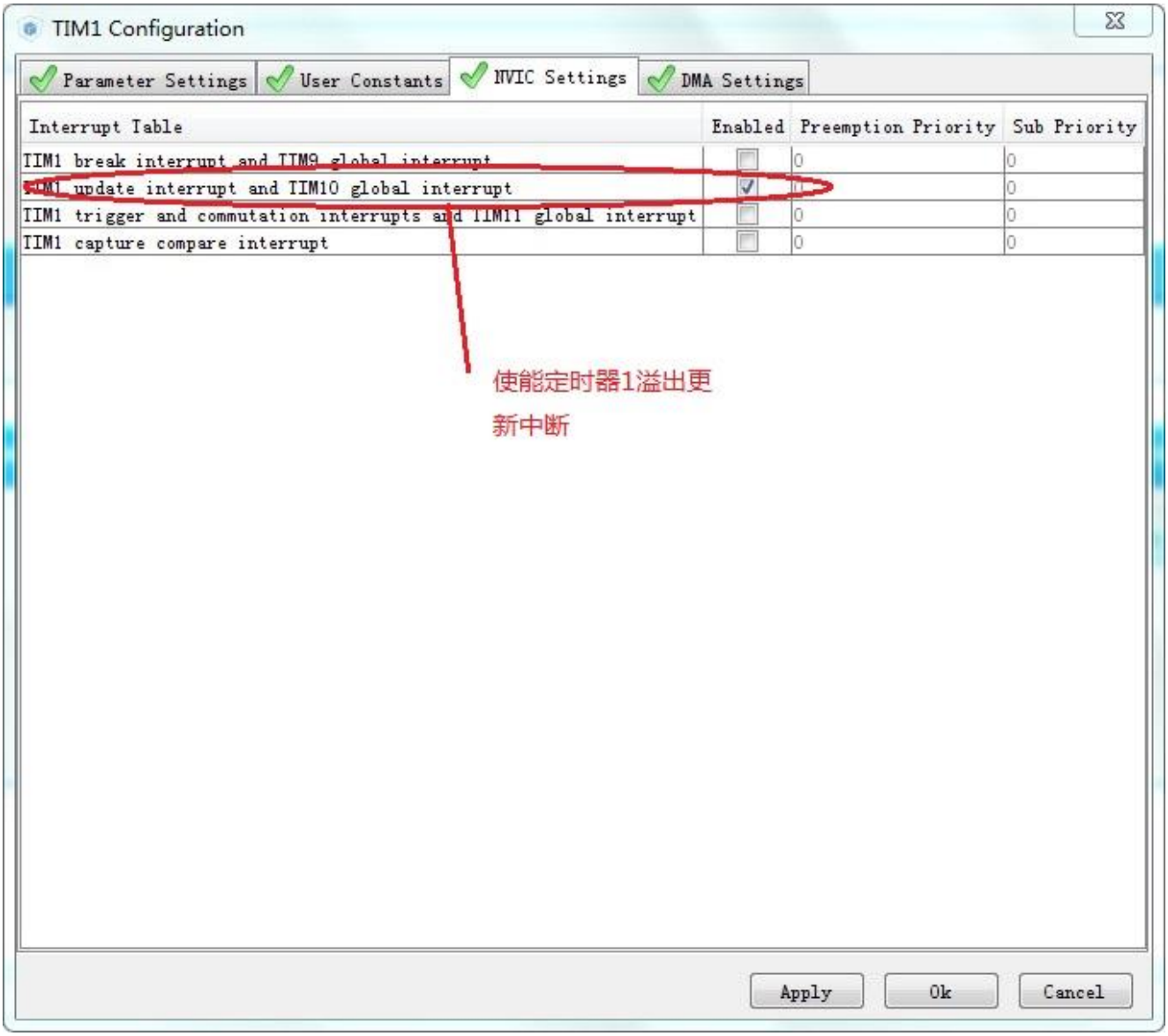
五、时钟配置。

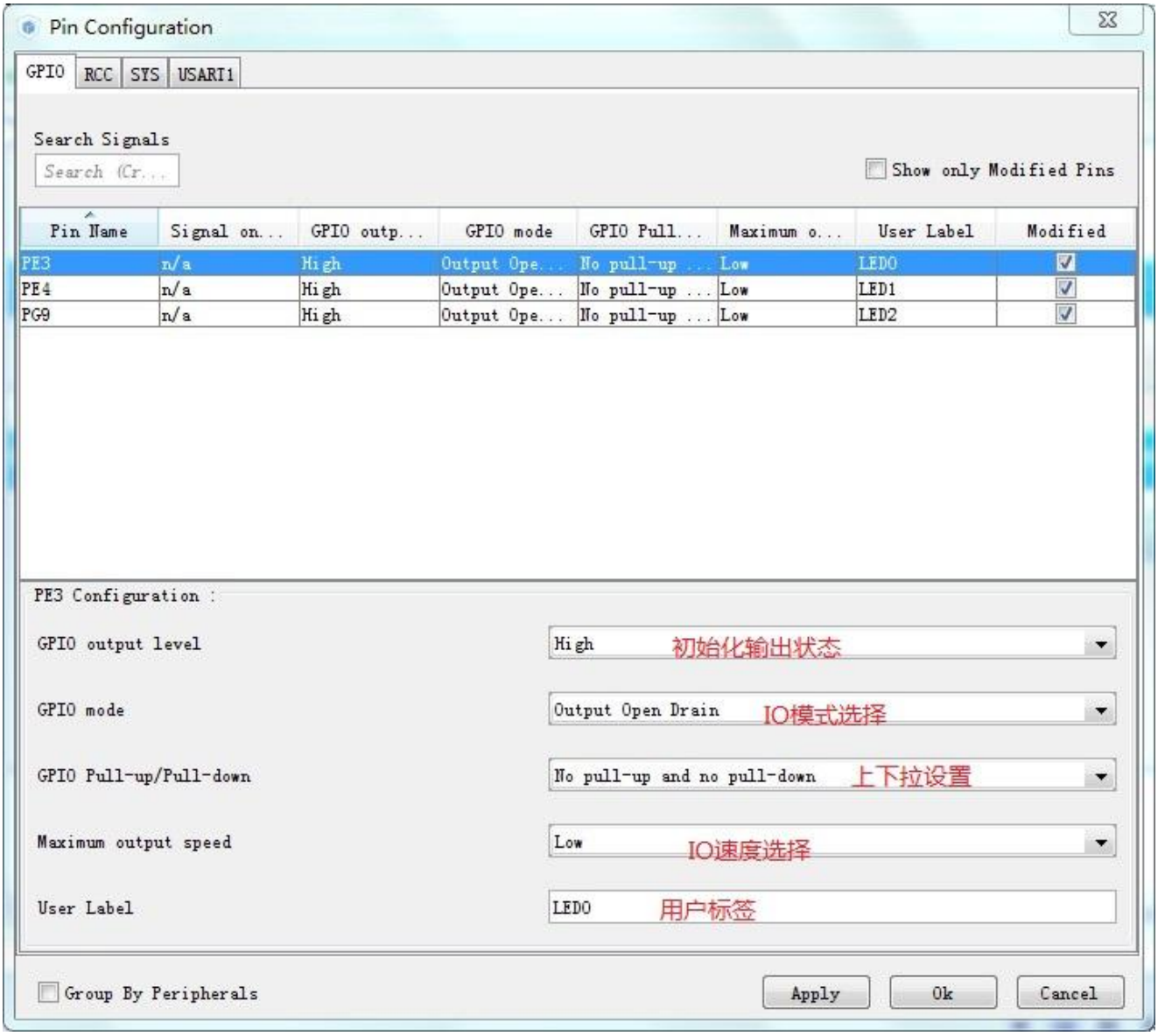

STM32CubeMX Untitled*: STM32F407ZGTx
File Project Clock Configuration Window Help
Pinout
Clock Configuration
Configuration
Power Consumption Calculator
RTC Clock Mux
To RTC (KHz)
To IWDG (KHz)
System Clock Mux
PLL Source Mux
Main PLL
PLLI2S
I2S source Mux
MCO2 source Mux
Ethernet PTP clock (MHz)
HCLK to AHB bus, core, memory and DMA (MHz)
To Cortex System timer (MHz)
FCLK Cortex clock (MHz)
APB1 peripheral clocks (MHz)
APB1 Timer clocks (MHz)
APB2 peripheral clocks (MHz)
APB2 timer clocks (MHz)
48MHz clocks (MHz)
I2S clocks (MHz)
系统时钟源选择，选
择经PLL倍频的高速
外部时钟
时钟安全是否使能（不需要理
会），使能后会启用时钟监测
器，检测HSE故障
高速外部晶振时钟

六、外设及中间件参数配置。

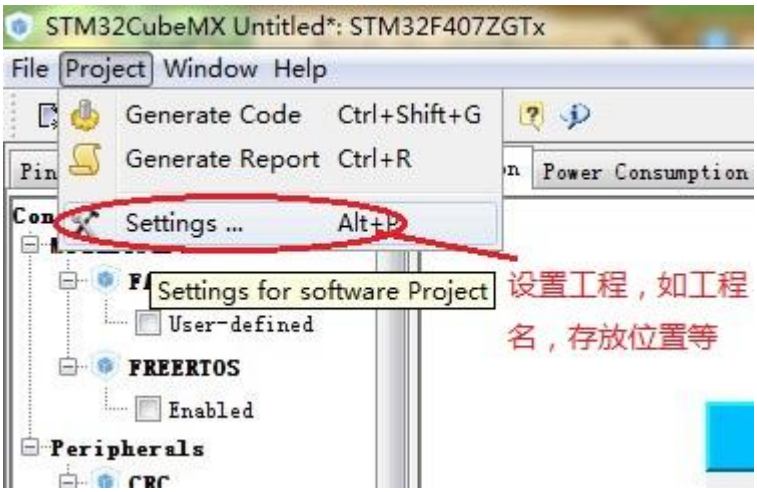

USART1 Configuration
Parameter Settings
User Constants
NVIC Settings
DMA Settings
GPIO Settings
Configure the below parameters :
添加宏定义
中断设置
DMA设置
引脚设置
Search :
Search (Crtl+F)
Basic Parameters
Baud Rate
115200 Bits/s
波特率
Word Length
8 Bits (including Parity)
数据位
Parity
None
校验位
Stop Bits
1
停止位
Advanced Parameters
Data Direction
Receive and Transmit
数据方向
Over Sampling
16 Samples
采样率
Apply
Ok
Cancel

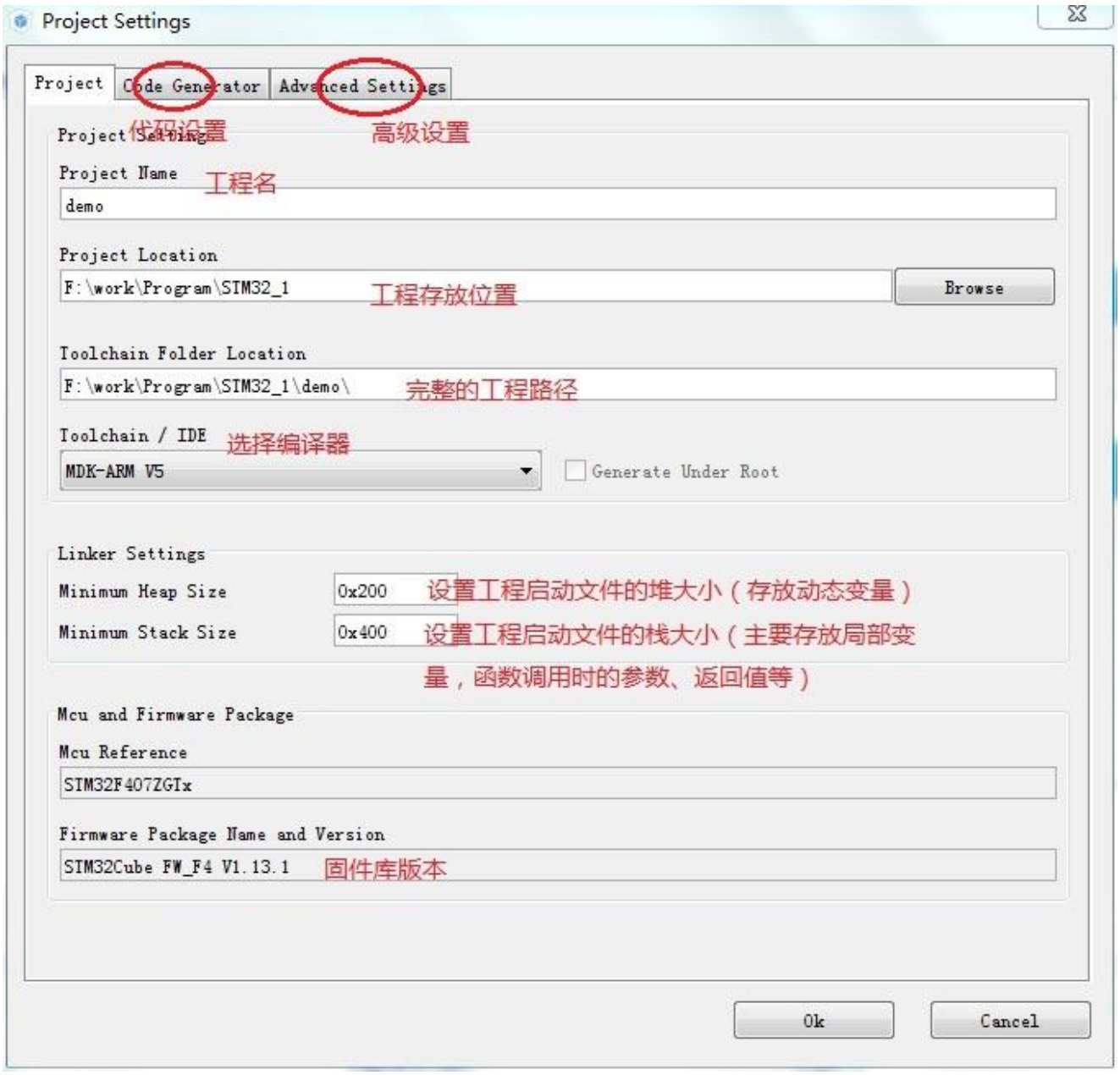

TIM1 Configuration
Parameter Settings
User Constants
NVIC Settings
DMA Settings
Configure the below parameters :
Search :
Search (Crtl+F)
Counter Settings
Prescaler (PSC - 16 bits value)
1679
分频系数
Counter Mode
Up
计数模式
Counter Period (AutoReload Register...
49999
计数周期
Internal Clock Division (CKD)
No Division
内部时钟分频系数
Repetition Counter (RCR - 8 bits va...
0
重复计数器值
Trigger Output (TRGO) Parameters
Master/Slave Mode
主从模式
Disable (no sync between this TIM (Master) and its ...
Trigger Event Selection
触发事件
Reset (UG bit from TIMx_EGR)
Apply
Ok
Cancel

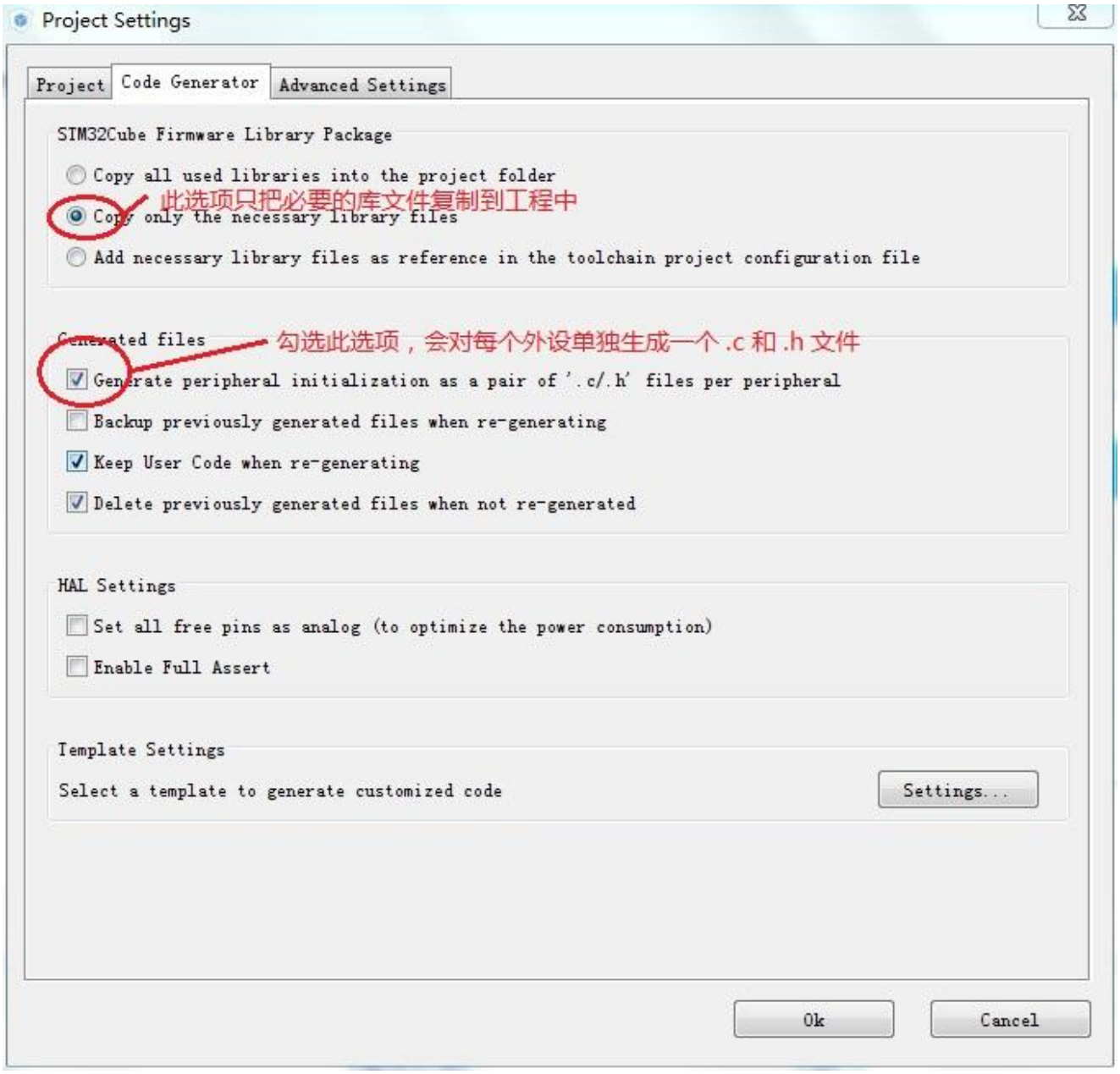

TIM1 Configuration
Parameter Settings
User Constants
NVIC Settings
DMA Settings
Interrupt Table
Enabled
Preemption Priority
Sub Priority
TIM1 break interrupt and TIM9 global interrupt
TIM1 update interrupt and TIM10 global interrupt
TIM1 trigger and commutation interrupts and TIM11 global interrupt
TIM1 capture compare interrupt
使能定时器1溢出更
新中断
Apply
Ok
Cancel

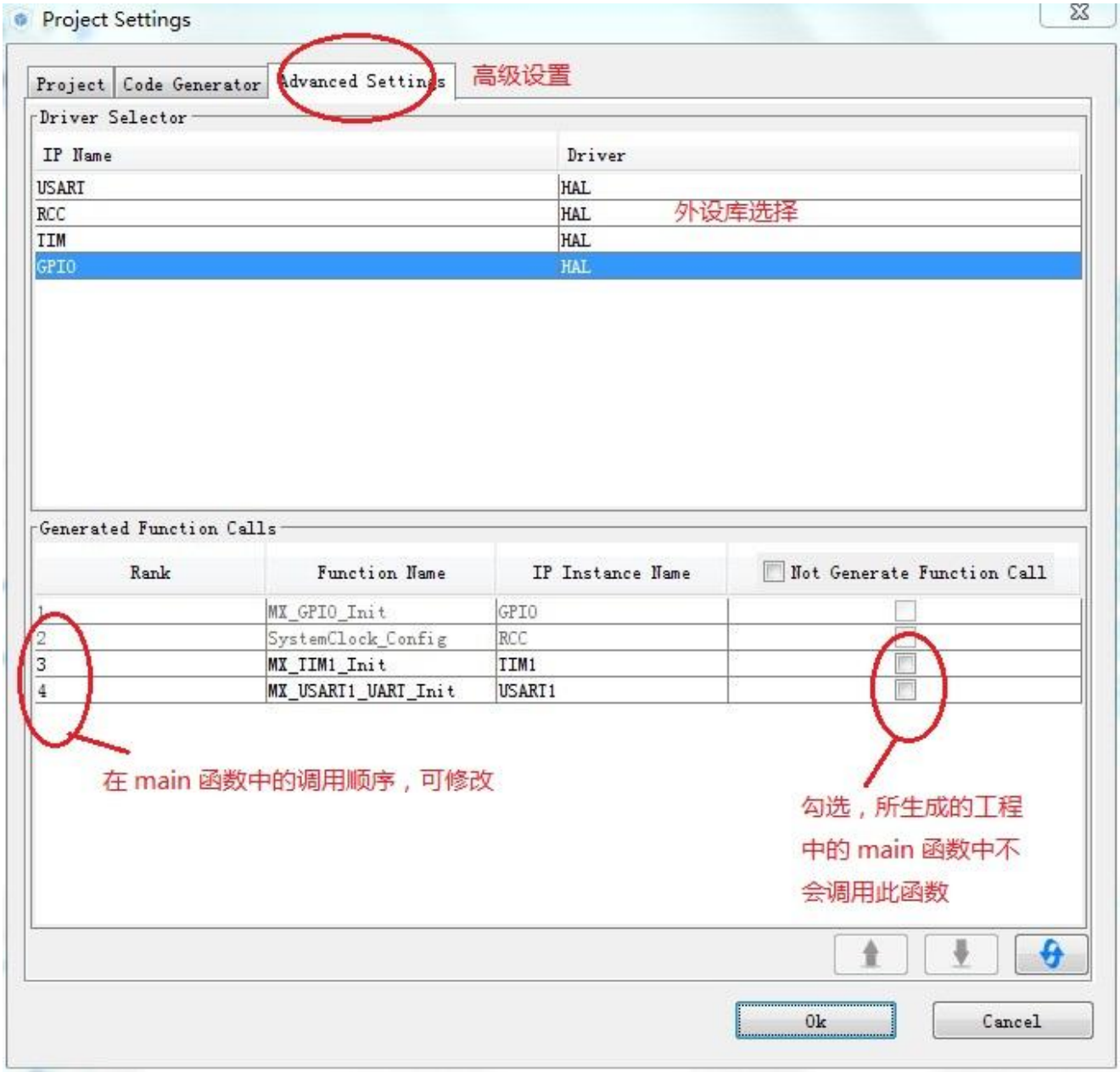

Pin Configuration
GPIO
RCC
SYS
USART1
Search Signals
Search (Cr...
Show only Modified Pins
Pin Name
Signal on...
GPIO outp...
GPIO mode
GPIO Pull...
Maximum o...
User Label
Modified
PE3
n/a
High
Output Ope...
No pull-up ...
Low
LED0
PE4
n/a
High
Output Ope...
No pull-up ...
Low
LED1
PG9
n/a
High
Output Ope...
No pull-up ...
Low
LED2
PE3 Configuration :
GPIO output level
High
初始化输出状态
GPIO mode
Output Open Drain
IO模式选择
GPIO Pull-up/Pull-down
No pull-up and no pull-down
上下拉设置
Maximum output speed
Low
IO速度选择
User Label
LED0
用户标签
Group By Peripherals
Apply
Ok
Cancel

七、工程设置。


STM32CubeMX Untitled*: STM32F407ZGTx
File
Project
Window
Help
Generate Code
Ctrl+Shift+G
Generate Report
Ctrl+R
Settings ...
Alt+P
Power Consumption
Settings for software Project
设置工程，如工程名，存放位置等
User-defined
FREERTOS
Enabled
Peripherals
CRC


Project Settings
Project
Code Generator
Advanced Settings
代码设置
高级设置
Project Setting
Project Name
工程名
demo
Project Location
F:\work\Program\STM32_1
工程存放位置
Browse
Toolchain Folder Location
F:\work\Program\STM32_1\demo\
完整的工程路径
Toolchain / IDE
选择编译器
MDK-ARM V5
Generate Under Root
Linker Settings
Minimum Heap Size
0x200
设置工程启动文件的堆大小（存放动态变量）
Minimum Stack Size
0x400
设置工程启动文件的栈大小（主要存放局部变量，函数调用时的参数、返回值等）
Mcu and Firmware Package
Mcu Reference
STM32F407ZGIx
Firmware Package Name and Version
STM32Cube FW_F4 V1.13.1
固件库版本
Ok
Cancel

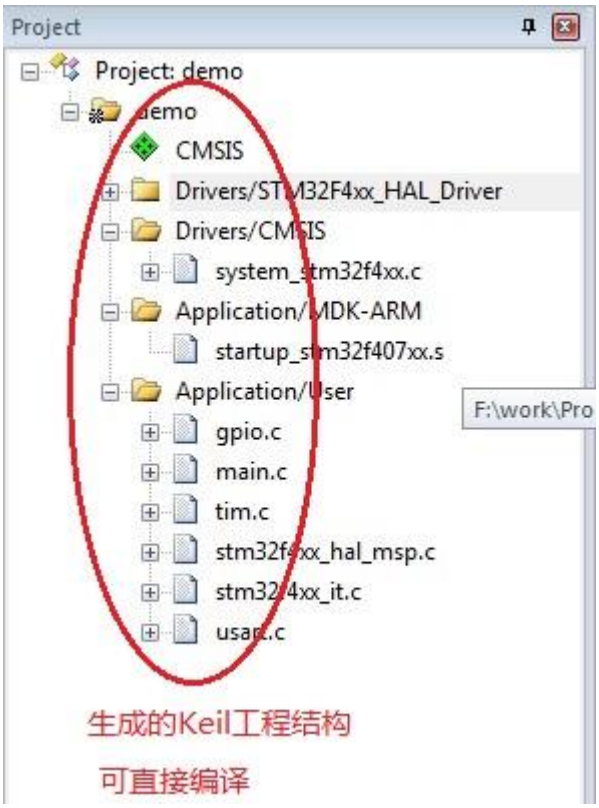

Project Settings
Project
Code Generator
Advanced Settings
STM32Cube Firmware Library Package
Copy all used libraries into the project folder
此选项只把必要的库文件复制到工程中
Copy only the necessary library files
Add necessary library files as reference in the toolchain project configuration file
Generated files
勾选此选项，会对每个外设单独生成一个 .c 和 .h 文件
Generate peripheral initialization as a pair of '.c/.h' files per peripheral
Backup previously generated files when re-generating
Keep User Code when re-generating
Delete previously generated files when not re-generated
HAL Settings
Set all free pins as analog (to optimize the power consumption)
Enable Full Assert
Template Settings
Select a template to generate customized code
Settings...
Ok
Cancel


Project Settings
Project
Code Generator
Advanced Settings
高级设置
Driver Selector
IP Name
Driver
USART
HAL
RCC
HAL
外设库选择
TIM
HAL
GPIO
HAL
Generated Function Calls
Rank
Function Name
IP Instance Name
Not Generate Function Call
1
MX_GPIO_Init
GPIO
2
SystemClock_Config
RCC
3
MX_TIM1_Init
TIM1
4
MX_USART1_UART_Init
USART1
在 main 函数中的调用顺序，可修改
勾选，所生成的工程中的 main 函数中不会调用此函数
Ok
Cancel

八、生成 Keil 工程。


File Project Window Help
Pinout
Clock Configuration
Configuration
Power Consumption Calculator
Configuration
MiddleWares
FATFS
User-defined
FREERTOS
Enabled
点击生成Keil工程
点击产生引脚、外设配置情况的 pdf 和 txt 文件

九、生成的 Keil 工程分析。



main.c

```
37  #include "usart.h"
38  #include "gpio.h"
39
40  /* USER CODE BEGIN Includes */
41
42  /* USER CODE END Includes */
43
44  /* Private variables ---------------------------
45
46  /* USER CODE BEGIN PV */
47  /* Private variables ---------------------------
48
49  /* USER CODE END PV */
50
51  /* Private function prototypes -----------------
52  void SystemClock_Config(void);
53  void Error_Handler(void);
54
55  /* USER CODE BEGIN PFP */
56  /* Private function prototypes -----------------
57
58  /* USER CODE END PFP */
59
60  /* USER CODE BEGIN 0 */
61
62  /* USER CODE END 0 */
63
64  int main(void)
65  {
66
67    /* USER CODE BEGIN 1 */
68
```

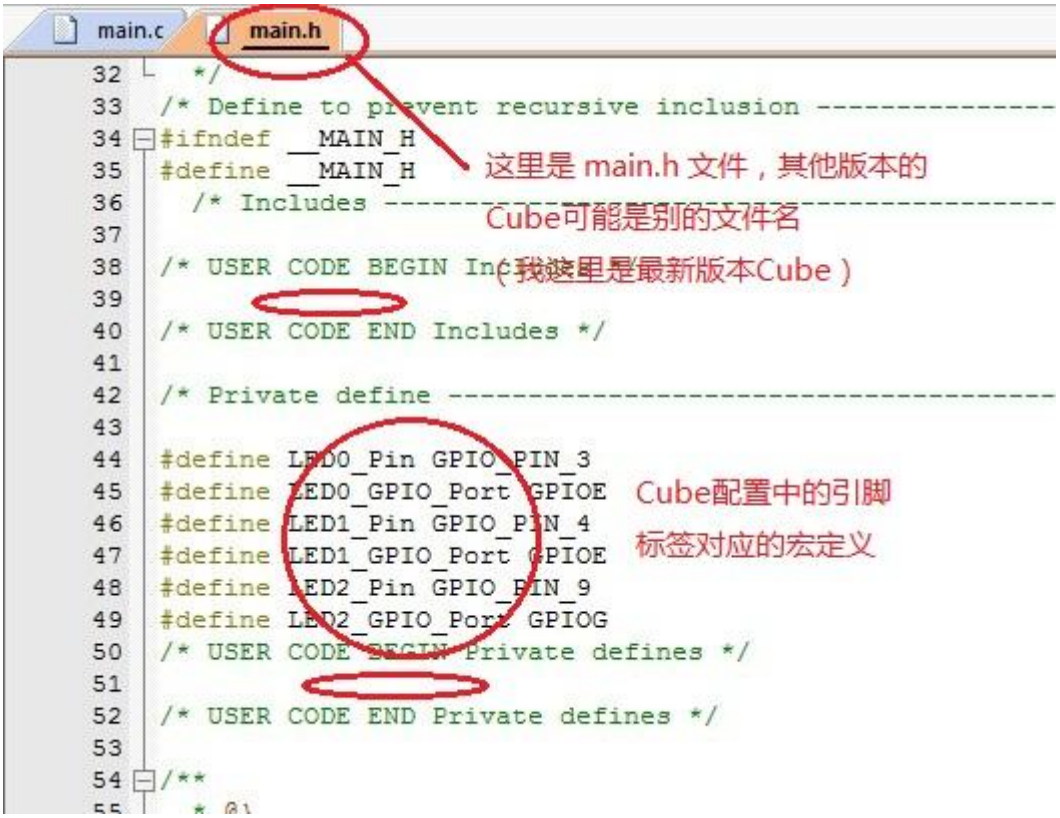
在“BEGIN” “END”之间添加代码，不会因 Cube 配置更改重新生成Keil工程而被清掉

```
63
64  int main(void)
65  {
66
67    /* USER CODE BEGIN 1 */
68
69    /* USER CODE END 1 */
70
71    /* MCU Configuration--------------------------
72
73    /* Reset of all peripherals, Initializes the F
74    HAL_Init();
75
76    /* Configure the system clock */
77    SystemClock_Config();
78
79    /* Initialize all configured peripherals */
80    MX_GPIO_Init();
81    MX_TIM1_Init();
82    MX_USART1_UART_Init();
83
84    /* USER CODE BEGIN 2 */
85
86    /* USER CODE END 2 */
87
88    /* Infinite loop */
89    /* USER CODE BEGIN WHILE */
90    while (1)
91    {
92    /* USER CODE END WHILE */
```

只能在“BEGIN”“END”之间添加代码

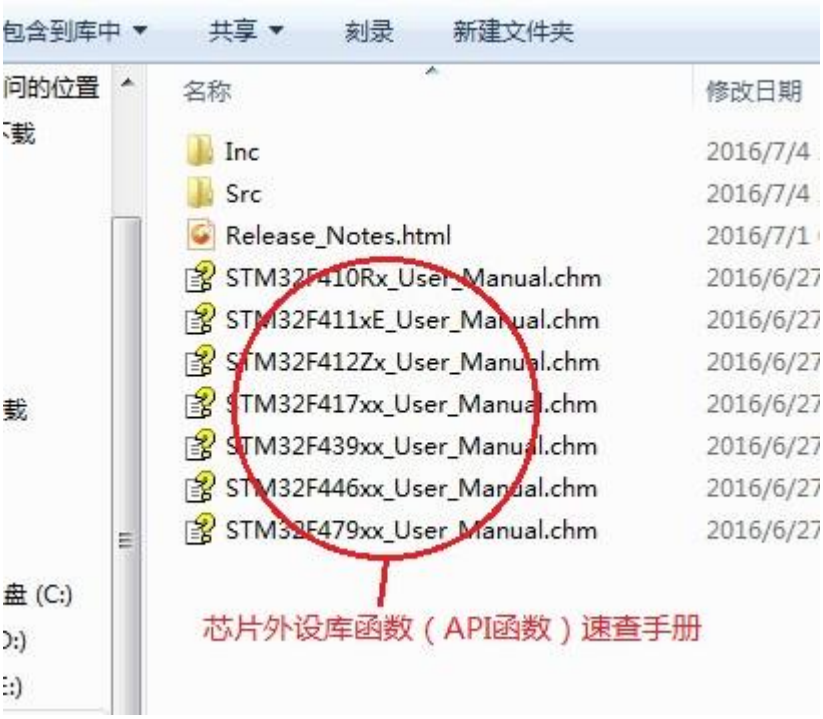
```
86    /* USER CODE END 2 */
87
88    /* Infinite loop */
89    /* USER CODE BEGIN WHILE */
90    while (1)
91    {
92    /* USER CODE END WHILE */
93
94    /* USER CODE BEGIN 3 */
95
96    }
97    /* USER CODE END 3 */
98
99  }
100
101 /** System Clock Configuration
102 */
```

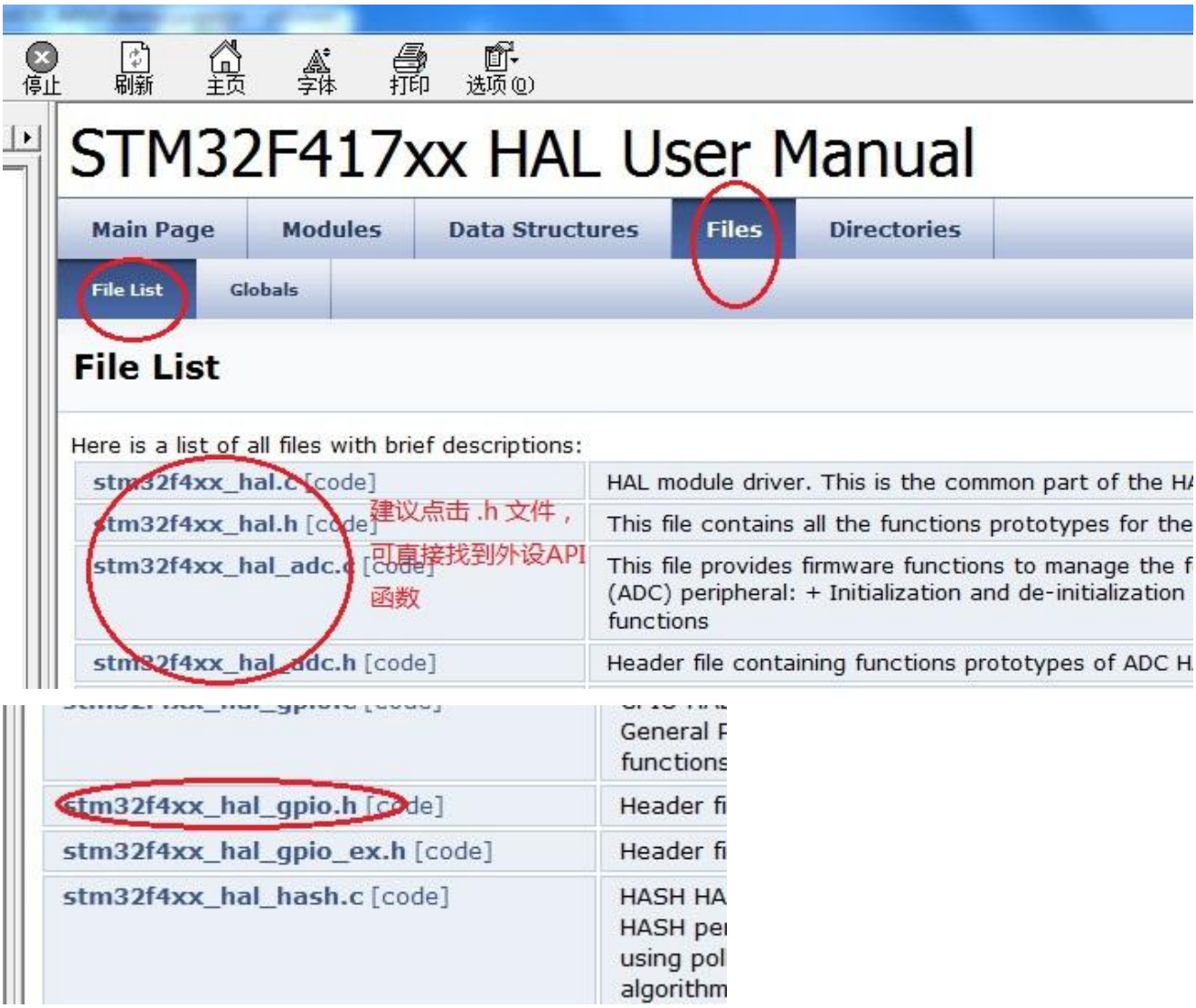
只能在“BEGIN”“END”之间添加代码



十、HAL 库函数（API 函数）查找方法。

解压 Cube 固件包，打开找到 Drivers 文件夹，再打开如下。 .chm 文件就是 HAL 库的 API 速查手册。方法如下：





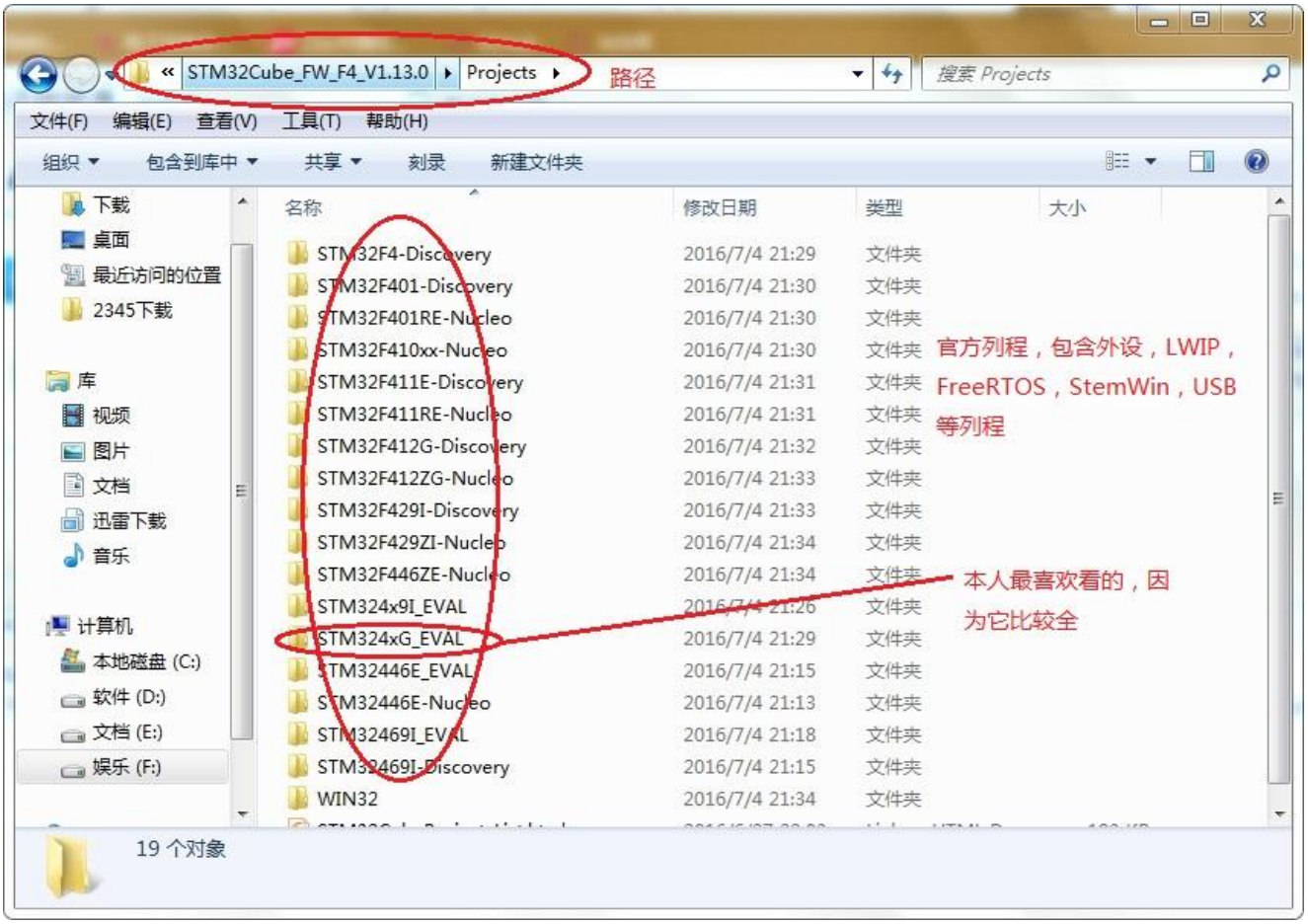
Functions

普通 IO API 函数声明

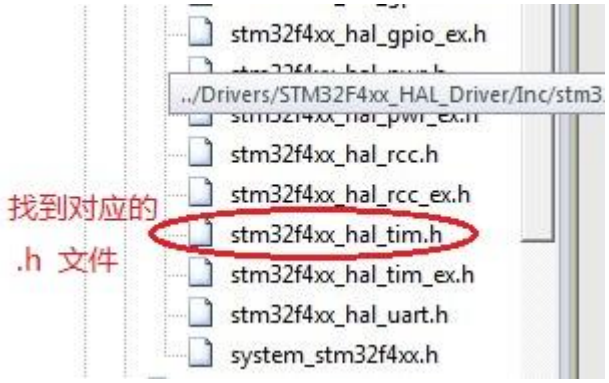
void	**HAL_GPIO_Init** (GPIO_TypeDef *GPIOx, **GPIO_InitTypeDef** *GPIO_Init) Initializes the GPIOx peripheral according to the specified parameters in the GPIO_Init.
void	**HAL_GPIO_DeInit** (GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin) De-initializes the GPIOx peripheral registers to their default reset values.
GPIO_PinState	**HAL_GPIO_ReadPin** (GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin) Reads the specified input port pin.
void	**HAL_GPIO_WritePin** (GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin, **GPIO_PinState** PinSta Sets or clears the selected data port bit.
void	**HAL_GPIO_TogglePin** (GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin) Toggles the specified GPIO pins.
HAL_StatusTypeDef	**HAL_GPIO_LockPin** (GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin) Locks GPIO Pins configuration registers.
void	**HAL_GPIO_EXTI_IRQHandler** (uint16_t GPIO_Pin) This function handles EXTI interrupt request.
__weak void	**HAL_GPIO_EXTI_Callback** (uint16_t GPIO_Pin) EXTI line detection callbacks.

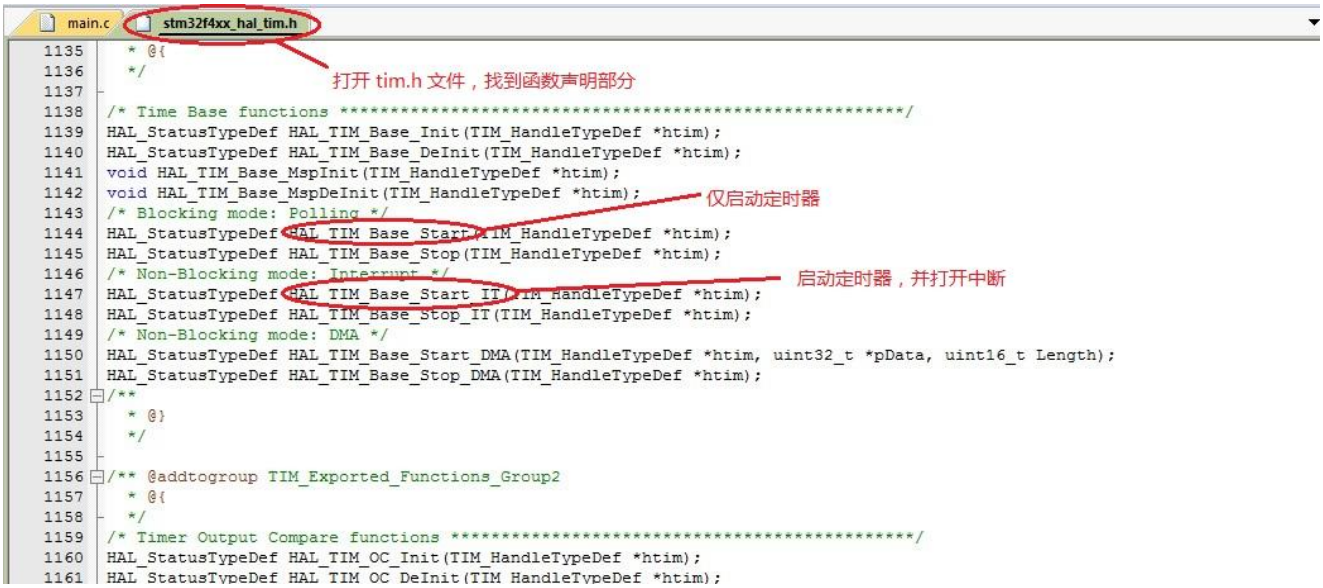
十一、学习 HAL 库的最好方法就是学习官方列程。

打开 Cube 固件包，找到 Projects 文件夹，里边有好多列程，都是官方出的开发板的 HAL 库列程，不过是手动建的工程，不是用 CubeMx 生成的。

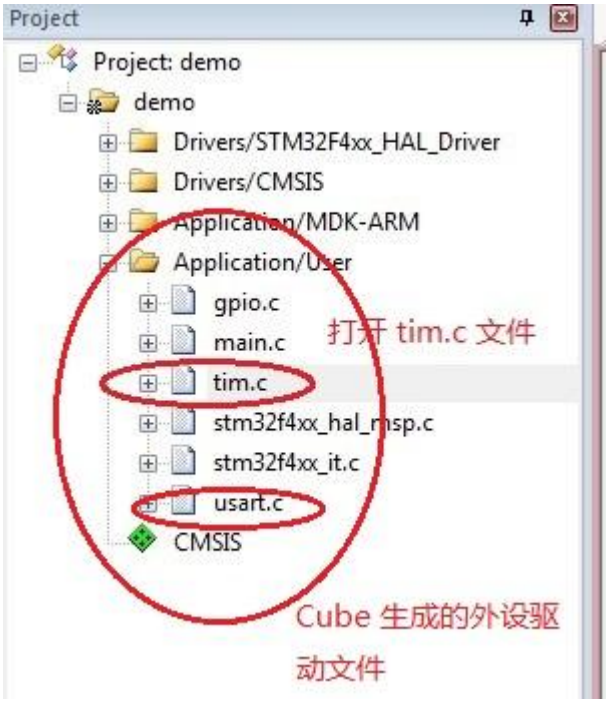


十二、添加应用程序。




main.c
stm32f4xx_hal_tim.h
打开 tim.h 文件，找到函数声明部分
仅启动定时器
启动定时器，并打开中断
1135 * @{
1136 */
1137
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, uint32_t *pData, uint16_t Length);
1151 HAL_StatusTypeDef HAL_TIM_Base_Stop_DMA(TIM_HandleTypeDef *htim);
1152 /**
1153 * @}
1154 */
1155
1156 /** @addtogroup TIM_Exported_Functions_Group2
1157 * @{
1158 */
1159 /* Timer Output Compare functions ***/
1160 HAL_StatusTypeDef HAL_TIM_OC_Init(TIM_HandleTypeDef *htim);
1161 HAL_StatusTypeDef HAL_TIM_OC_DeInit(TIM_HandleTypeDef *htim);


main.c
stm32f4xx_hal_tim.h
tim.c
tim 计数器溢出中断的回调函数
其他中断的回调函数
1300 /** @addtogroup TIM_Exported_Functions_Group9
1301 * @{
1302 */
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
1311 /**
1312 * @}
1313 */
1314
1315 /** @addtogroup TIM_Exported_Functions_Group10
1316 * @{


Project
Project: demo
demo
Drivers/STM32F4xx_HAL_Driver
Drivers/CMSIS
Application/MDK-ARM
Application/User
gpio.c
main.c
tim.c
stm32f4xx_hal_msp.c
stm32f4xx_it.c
usart.c
CMSIS
打开 tim.c 文件
Cube 生成的外设驱动文件

main.c | stm32f4xx_hal_tim.h | tim.c

```
34
35  /* Includes ------------------------------------------
36  #include "tim.h"
37
38  /* USER CODE BEGIN 0 */
39
40  /* USER CODE END 0 */
41
42  TIM_HandleTypeDef htim1;
43
44  /* TIM1 init function */
45  void MX_TIM1_Init(void)
46  {
47    TIM_ClockConfigTypeDef sClockSourceConfig;
48    TIM_MasterConfigTypeDef sMasterConfig;
49
50    htim1.Instance = TIM1;
51    htim1.Init.Prescaler = 1679;
52    htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
```

找到 tim 结构体变量定义 htim1

main.c

```
78
79    /* Initialize all configured peripherals */
80    MX_GPIO_Init();
81    MX_TIM1_Init();
82    MX_USART1_UART_Init();
83
84    /* USER CODE BEGIN 2 */
85    HAL_TIM_Base_Start_IT(&htim1);
86    /* USER CODE END 2 */
87
88    /* Infinite loop */
89    /* USER CODE BEGIN WHILE */
90    while (1)
91    {
92    /* USER CODE END WHILE */
93
94    /* USER CODE BEGIN 3 */
95
96      HAL_Delay(500);
97    }
98    /* USER CODE END 3 */
99
100 }
101
102 /** System Clock Configuration
```

调用 Cube 自动生成 的外设初始化函数

添加启动定时器函数，并开启中断

HAL库自带的延时函数，单位 ms ，以 系统滴答定时器为时钟基准（就是 Cube 引脚配置页 中的 SYS 项 的 Time Base 选项所选的时钟）。
注意，HAL库的外设驱动函数中会调用它


main.c
500ms 定时器中断，参数配置可看 Cube 配置过程
重新定义 tim 周期移除中断函数，实现我们想要的功能，
（会覆盖 HAL 库中的同名函数）
普通 IO 状态翻转，LED0_Pin等是Cube引脚配置中的引脚
标识符
区分不同定时器中断（当只有一个定时
器中断时，可省略）
串口输出字符串
/* USER CODE BEGIN 4 */
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
if(htim == &htim1)
{
HAL_GPIO_TogglePin(LED0_GPIO_Port, LED0_Pin);
HAL_GPIO_TogglePin(LED1_GPIO_Port, LED1_Pin);
HAL_GPIO_TogglePin(LED2_GPIO_Port, LED2_Pin);
printf("\n\r UART Printf Example: retarget the C library printf function to the UART\n\r");
}
}
/* USER CODE END 4 */


main.c
stm32f4xx_hal_tim.h
tim.c
usart.c
Cube 生成的外设驱动文件
/* USER CODE END USART1_MspDeInit 1 */
}
/* USER CODE BEGIN 1 */
区分编译器
#ifdef __GNUC__
/* With GCC/RAISONANCE, small printf (option LD Linker->Libraries->Small printf
set to 'Yes') calls __io_putchar() */
#define PUTCHAR_PROTOTYPE int __io_putchar(int ch)
#else
#define PUTCHAR_PROTOTYPE int fputc(int ch, FILE *f)
#endif /* __GNUC__ */
/**
* @brief Retargets the C library printf function to the USART.
* @param None
* @retval None
*/
PUTCHAR_PROTOTYPE
print 函数重定向
{
/* Place your implementation of fputc here */
/* e.g. write a character to the EVAL_COM1 and Loop until the end of transmission */
HAL_UART_Transmit(&huart1, (uint8_t *)&ch, 1, 0xFFFF);
return ch;
}
/* USER CODE END 1 */

十三、串口打印效果。

十四、小结。

CubeMx 生成的 Keil 工程，可以像我们平时用 标准库 建的工程一样添加 工程文件 、工程文件夹 和工程路径。但有一点要注意，就是 在 CubeMx 生成 的 文件 中添加代码时 ，一定要在“BEGIN”“END”之间添加，否则，修改 CubeMx 工程配置 重新生成 Keil 工程时，会把 “BEGIN”“END”之间 之外的东西清掉。