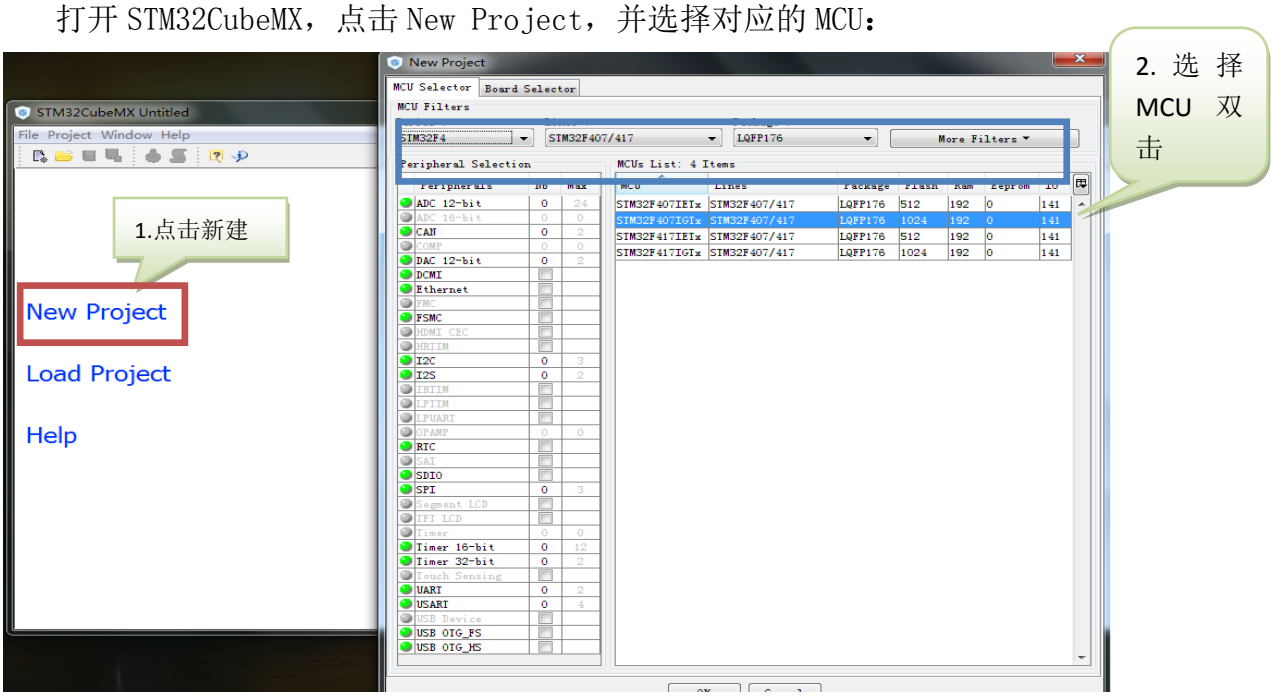
前提：使用 STM32 系列 MCU 很强大的辅助工具，最直接的图形配置和最新的库函数支持，最简单 KEIL 工程的建立。

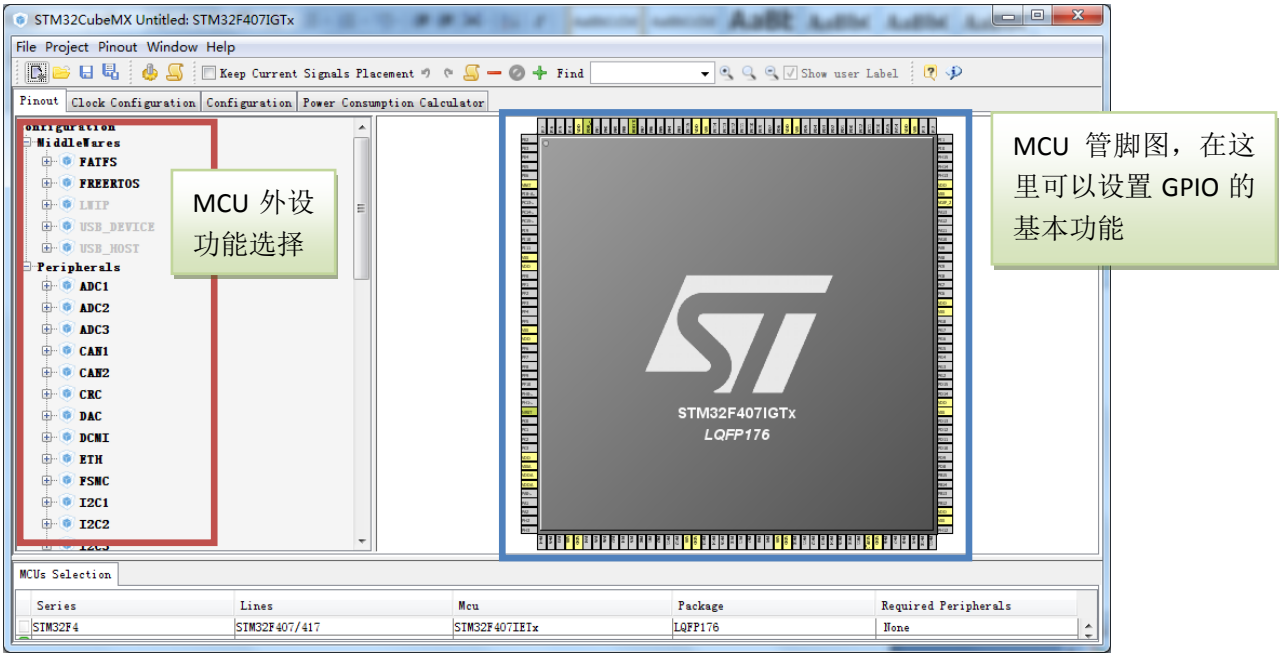
STM32CubeMX 最新版增加了对 STM32F1 系列 MCU 的支持，我的例程是以 STM32F4077IGXX 为准，其他的 STM32 系列的 MCU 设置也是大同小异的。本文乃闲时兴起所作，多有瑕疵，让大家见笑了。

Step 1：工程建立

打开 STM32CubeMX，点击 New Project，并选择对应的 MCU：

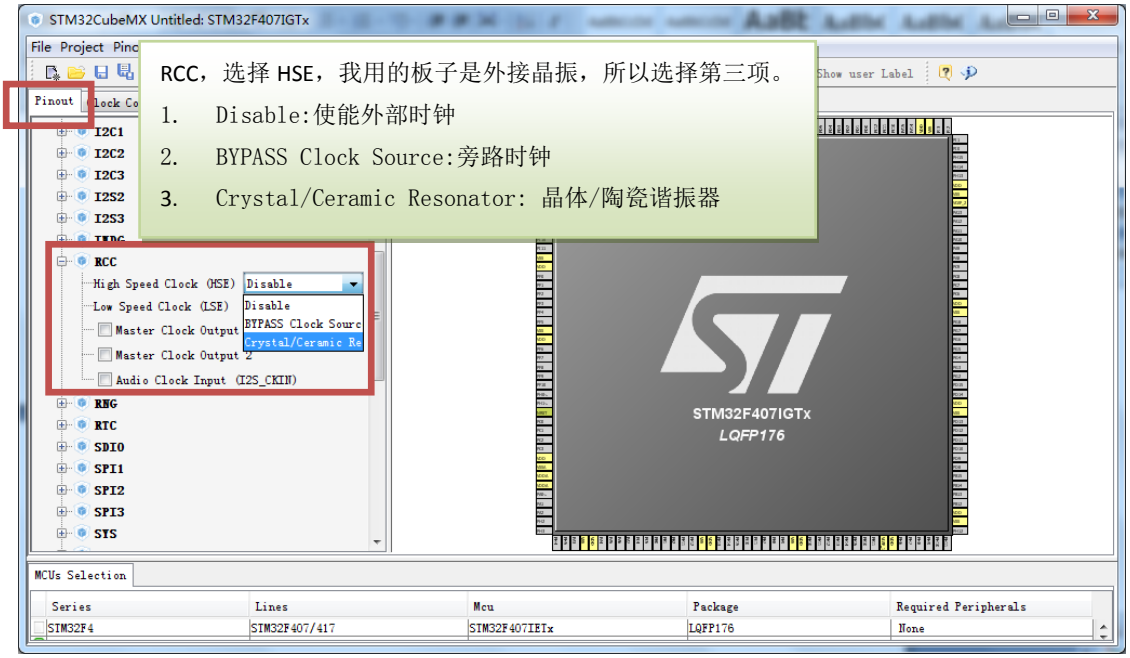


双击 MCU 进入工程界面，如图：

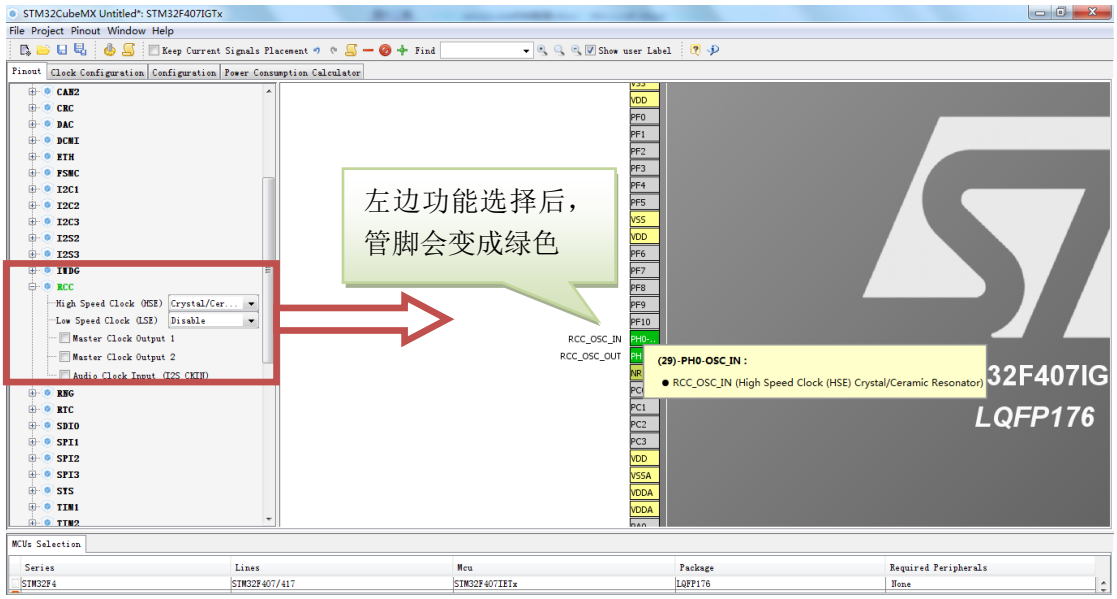


Step 2：外设功能选择（Pinout）

1.RCC 设置：



选择使能 RCC 之后，右边的 MCU 相对应的管脚会自动变成绿色，说明该管脚已经被使用：

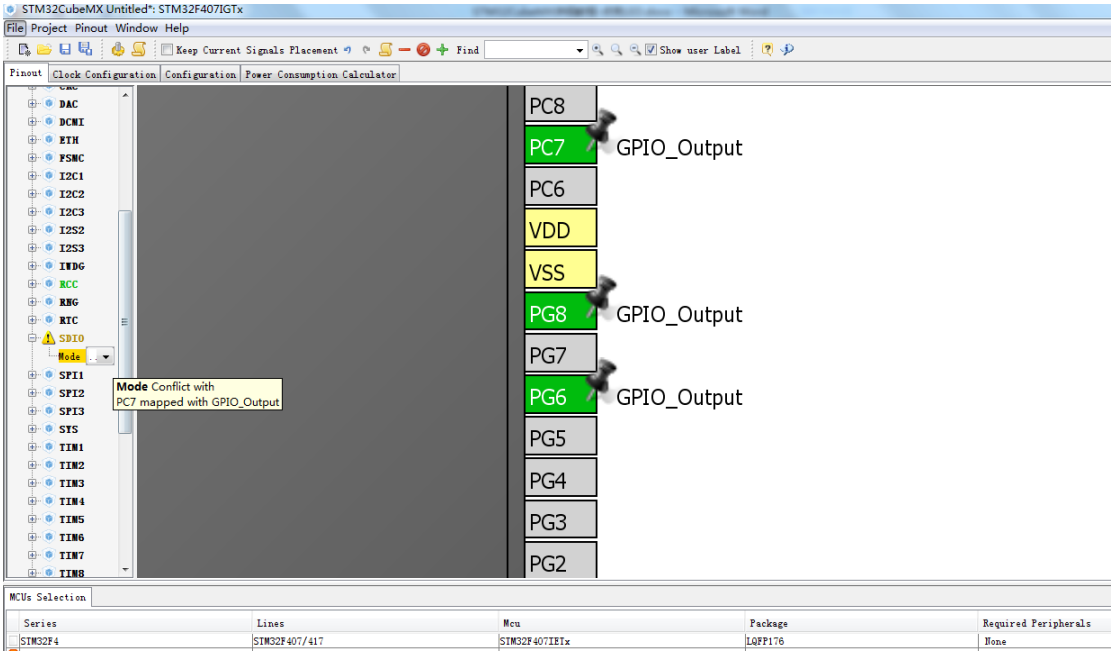


因为我们这里只是点灯一个简单动作，所以外设功能选择 RCC。

2.GPIO 口功能选择

GPIO 口的功能选择在这里面使用起来很简单。比如说我的板子是 PG6,PG8,PC7 作为 LED1,LED2,LED3 的 IO 口，那么我直接找到相对应的 IO 口，则会弹出该 IO 口所能设置的所有选项：

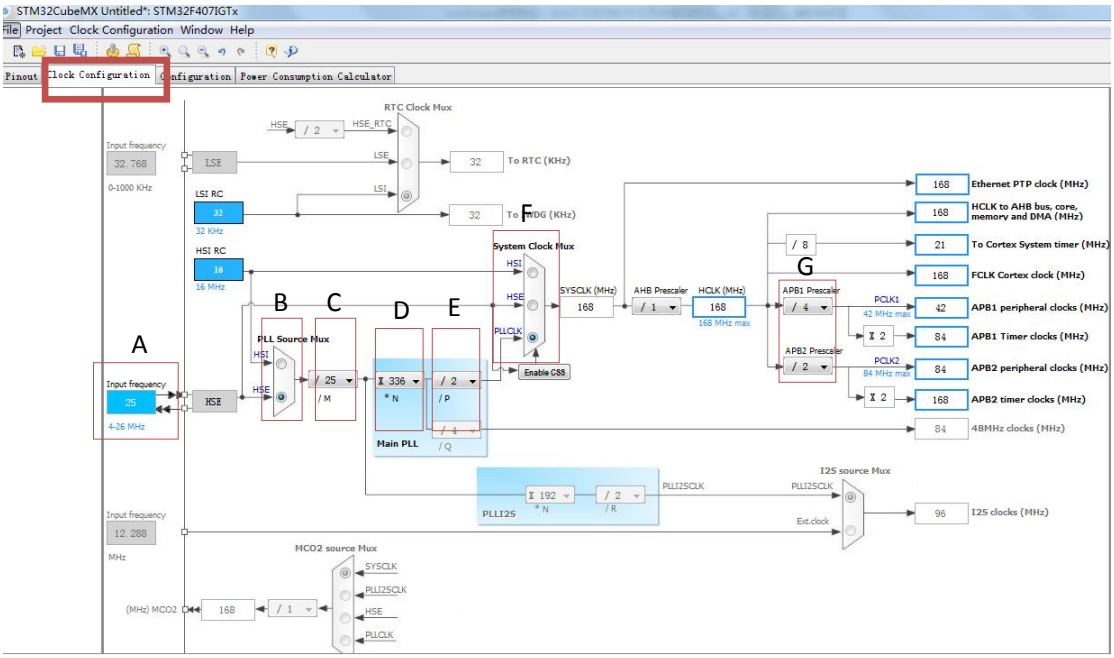
选择 GPIO_Output 即可。同理将 PG6，PG8 设置成 GPIO_Output 即可。



左边的黄色警告指的是该功能的 GPIO 已经被映射作用其他功能了，可以忽略。

Step 3：时钟配置（Clock Configuration）

Clock Configuration 配置完全采用图形方式，只要了解了 RCC 的配置原理你会觉得这种方式实在是太强大了，后面会有介绍。具体框图如下：



A：Input Frequency

B:PLL Source Mux

C:PLLM

D:PLLN

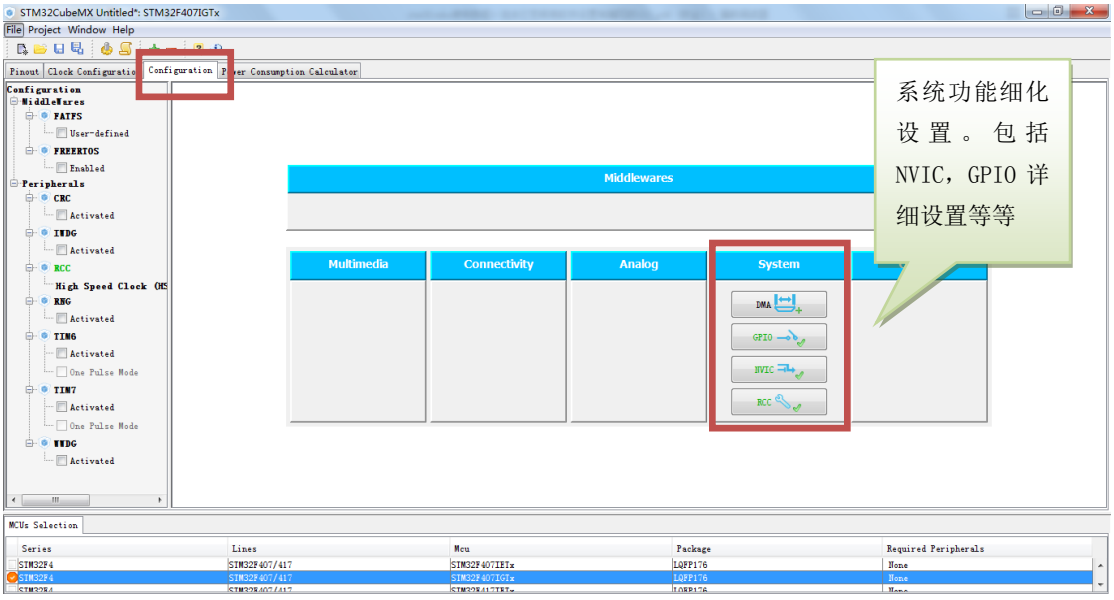
E:PLLP

F:System Clock Mux

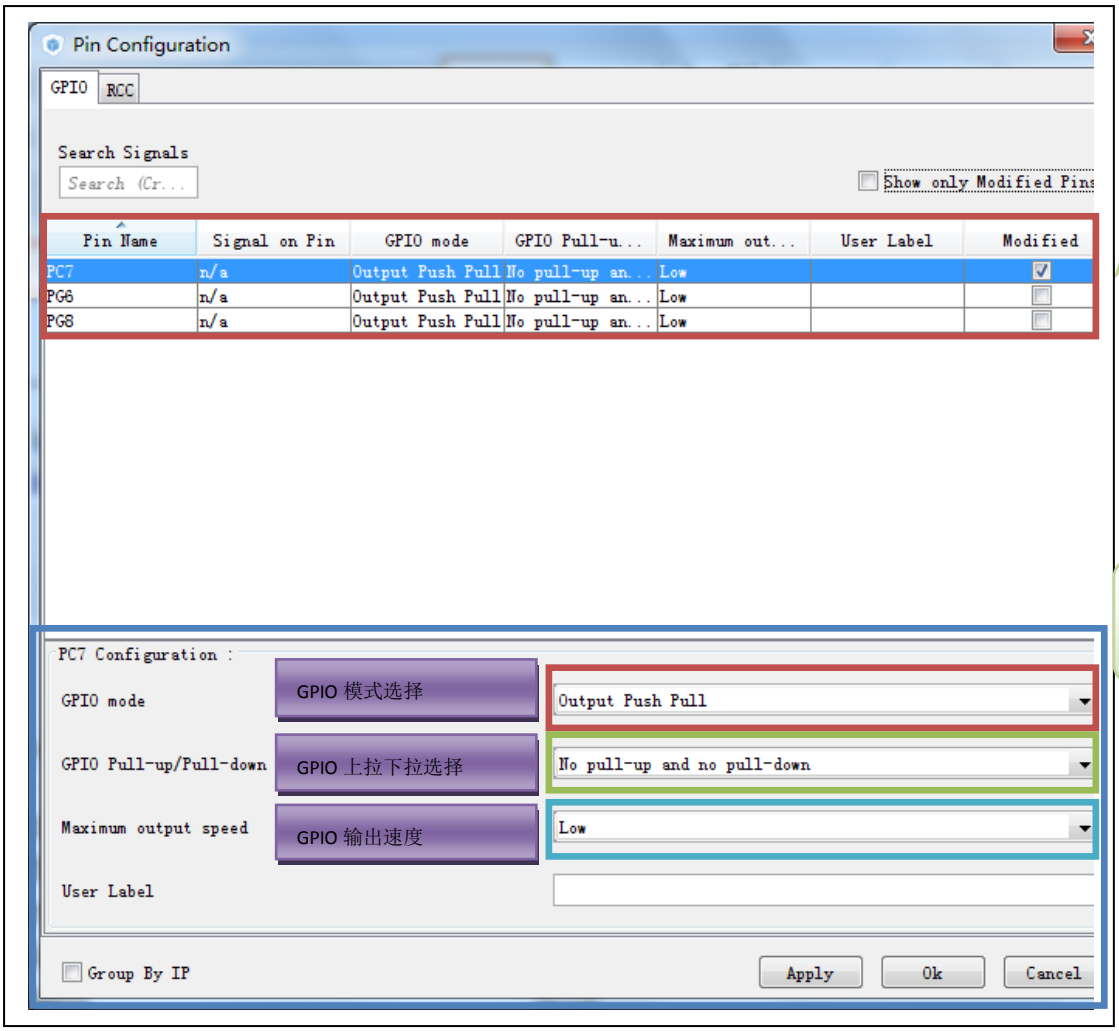
G:APB1/APB2 Perscaler

为什么要这样设置，自己可以去看 STM32F4 系列参考手册。我在后面会详细说明。

Step 4：深入配置（Configuration）



在这里我们只用到 GPIO 口，点击进入 GPIO 详细设置。如图：



GPIO mode：Output Push Pull 推挽输出（已选择）

Output Open Drain 开漏输出

GPIO Pull-up/Pull-dowm: NO pull-up and no pull-down 没有上下拉

pull-up 上拉（已选择）

pull-down 下拉

Maximum output speed: Low 慢

Medium 中

Fast 快

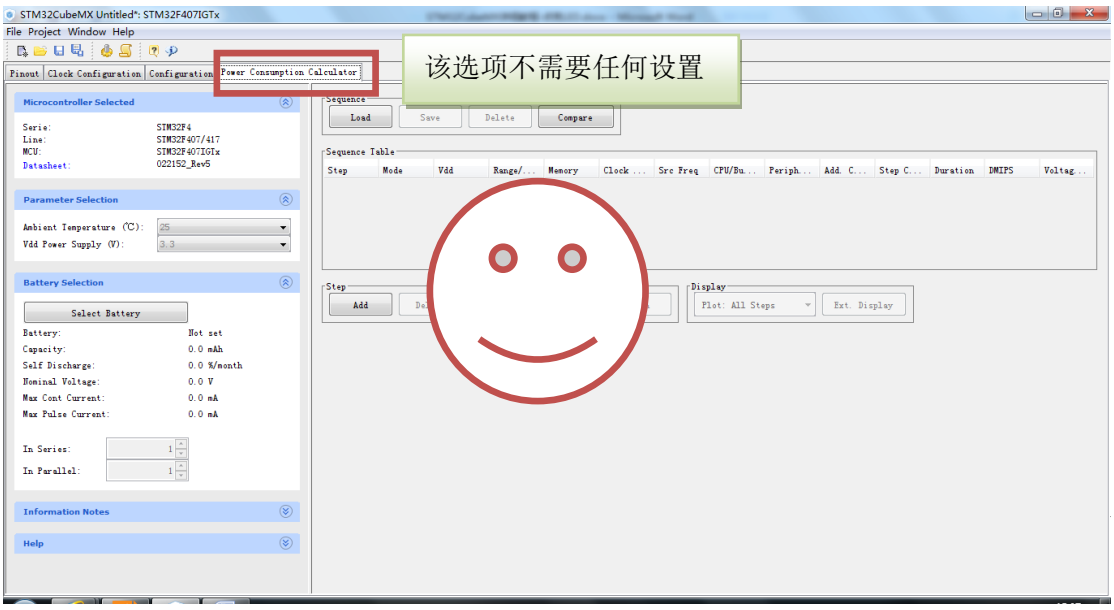
High 高（已选择）

大家可以根据自己的板子情况设置相关参数。

上面是我的 GPIO 口详细设置，因为忘记将设置的界面截图了。

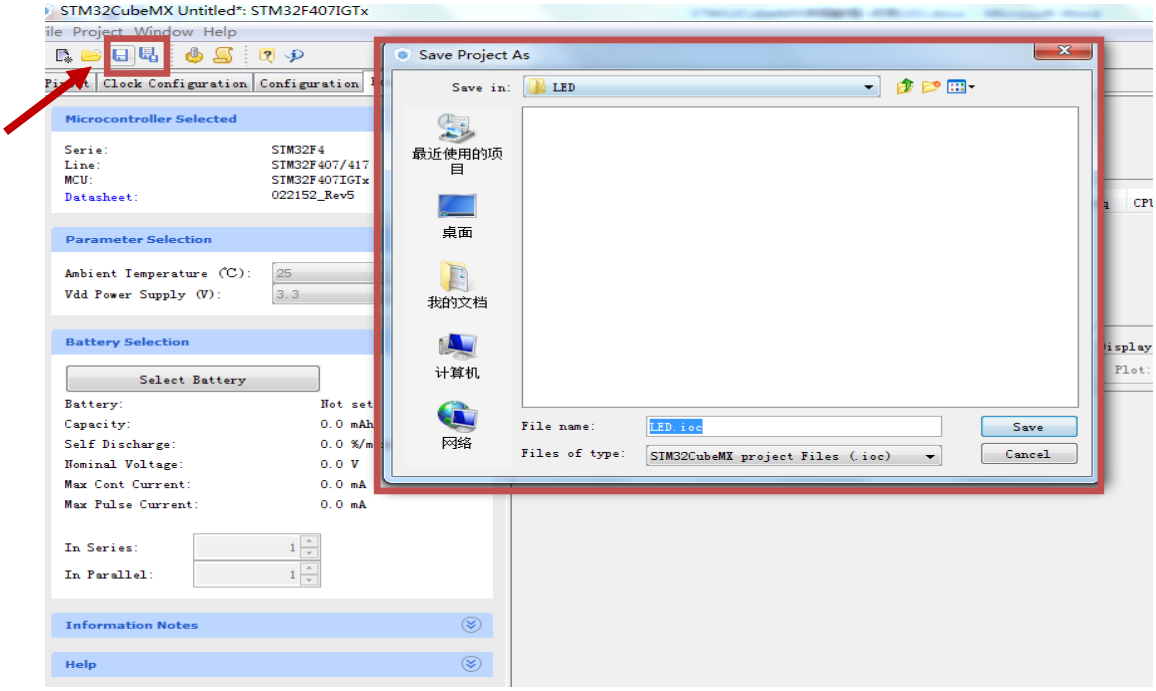
Step 5：低功耗计算（power consumption calculator）

该功能针对 F0，L 系列低功耗 MCU，这里不需要理会。

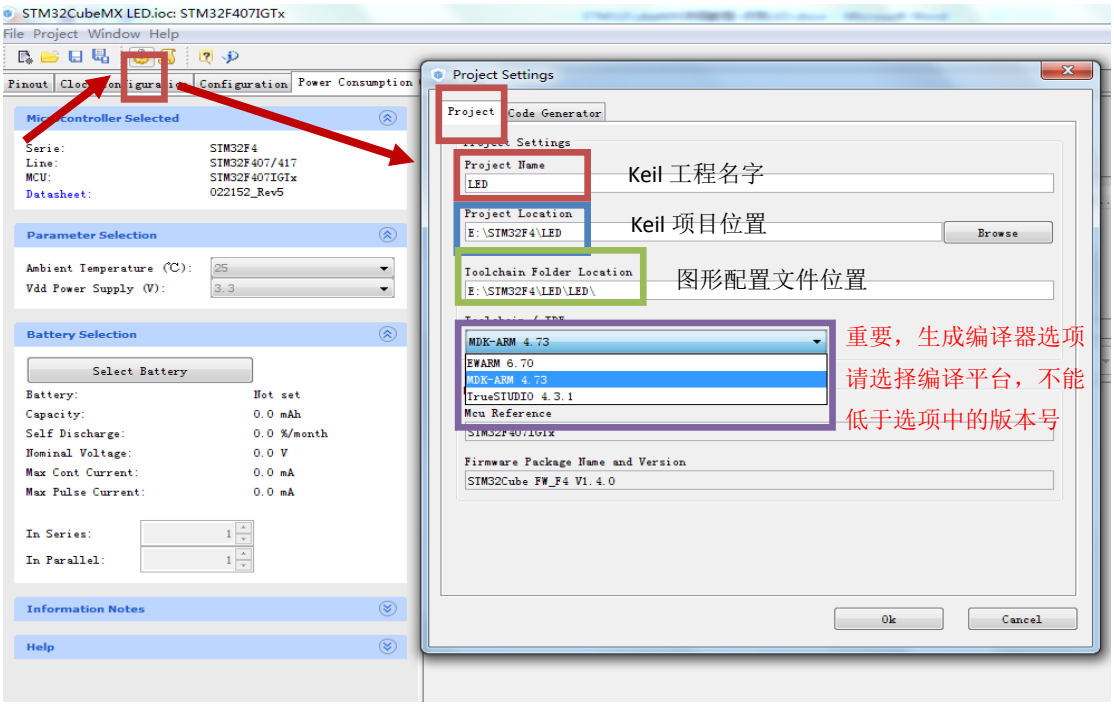


Step 6：保存配置和输出到工程目录

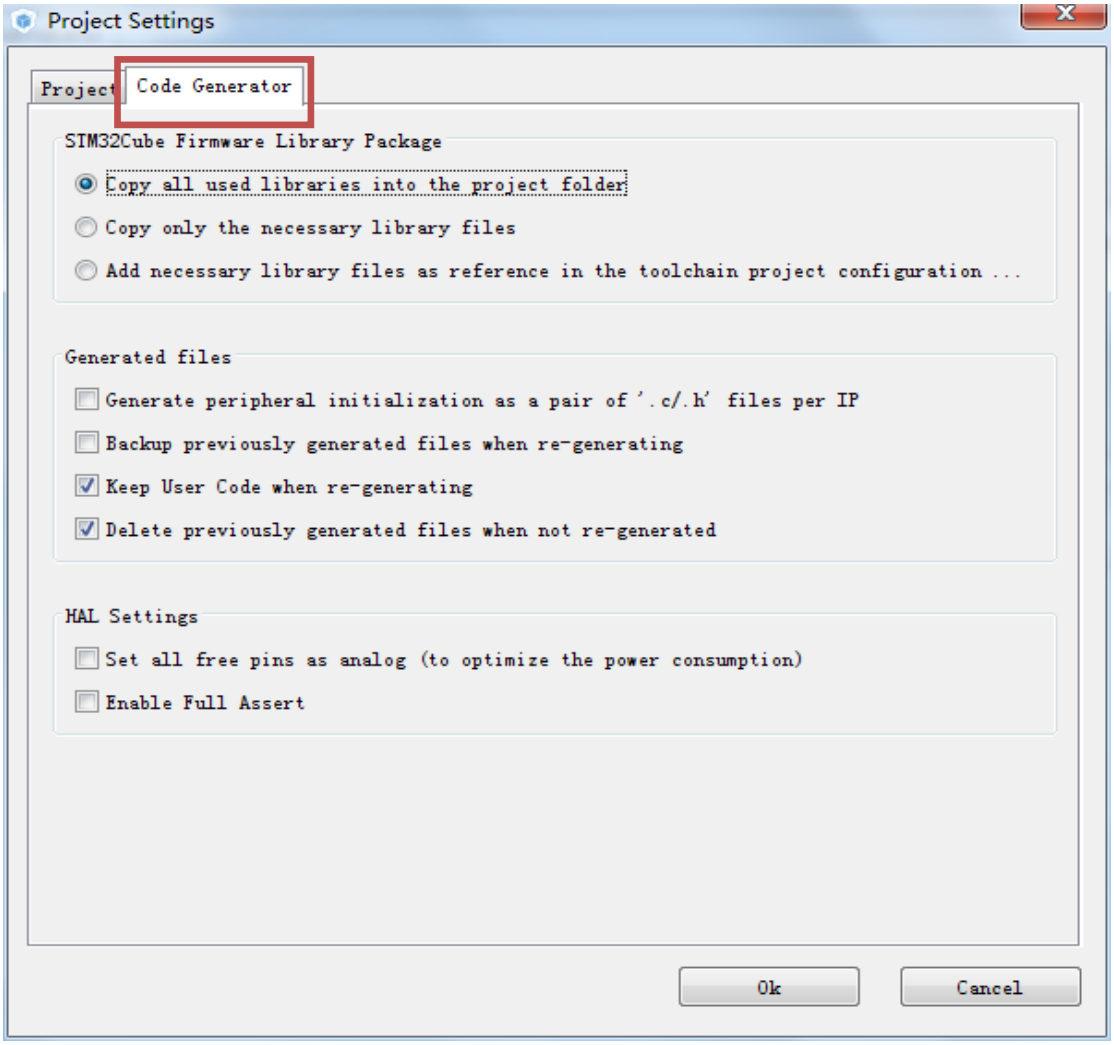
1. 保存配置（两个保存图标功能一样）



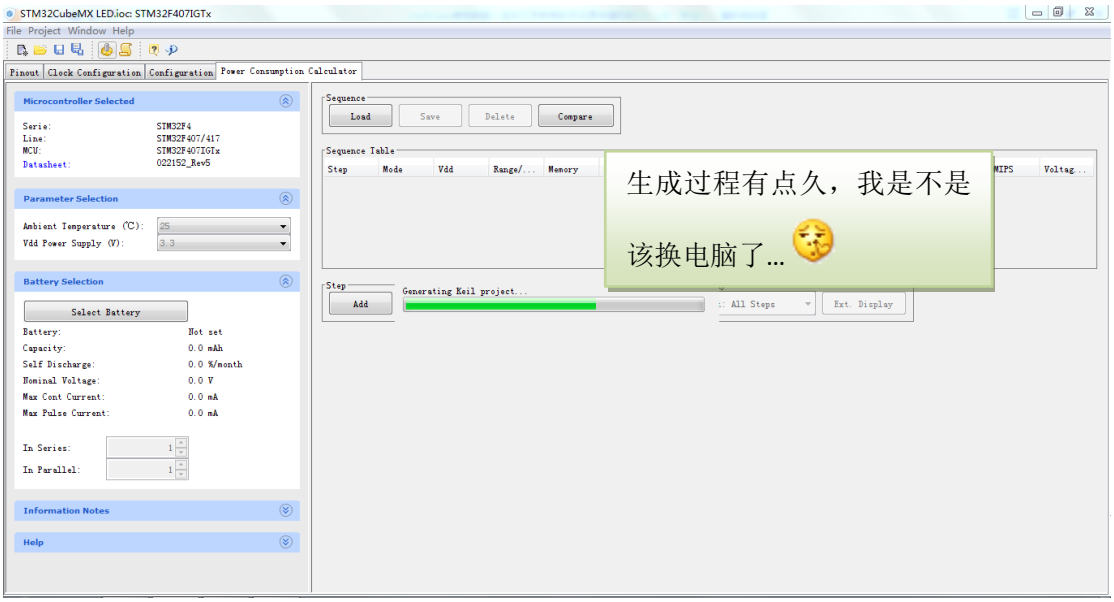
2. 根据用户的设置生成的源代码

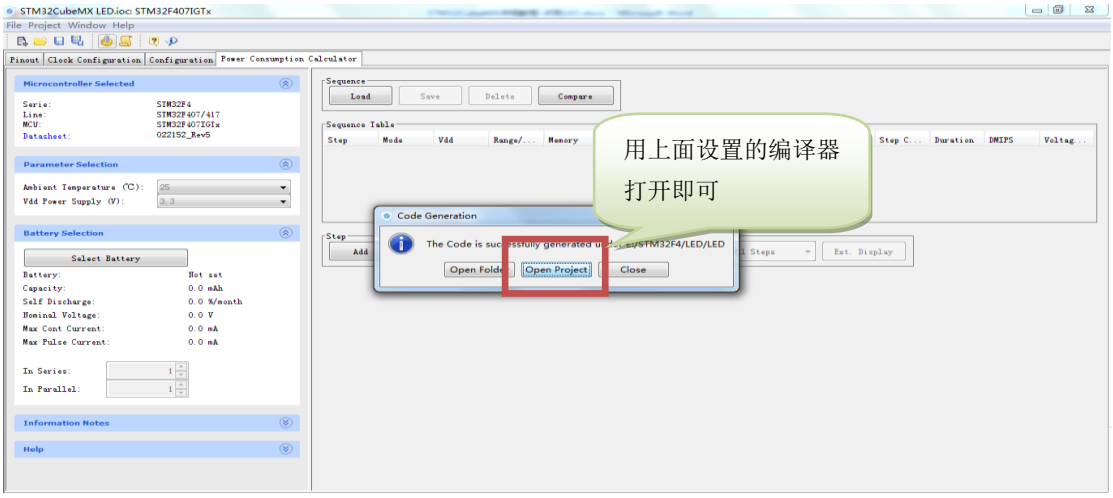

STM32CubeMX LED.ioc: STM32F407IGTx
File Project Window Help
Pinout Configuration Power Consumption
Project Settings
Project Code Generator
Project Name
LED
Keil 工程名字
Project Location
E:\STM32F4\LED
Browse
Keil 项目位置
Toolchain Folder Location
E:\STM32F4\LED\LED\
图形配置文件位置
MDK-ARM 4.73
EWARM 6.70
MDK-ARM 4.73
TrueSTUDIO 4.3.1
Mcu Reference
重要，生成编译器选项
请选择编译平台，不能
低于选项中的版本号
Firmware Package Name and Version
STM32Cube FW_F4 V1.4.0
Ok
Cancel
Parameter Selection
Battery Selection
Select Battery
Information Notes
Help

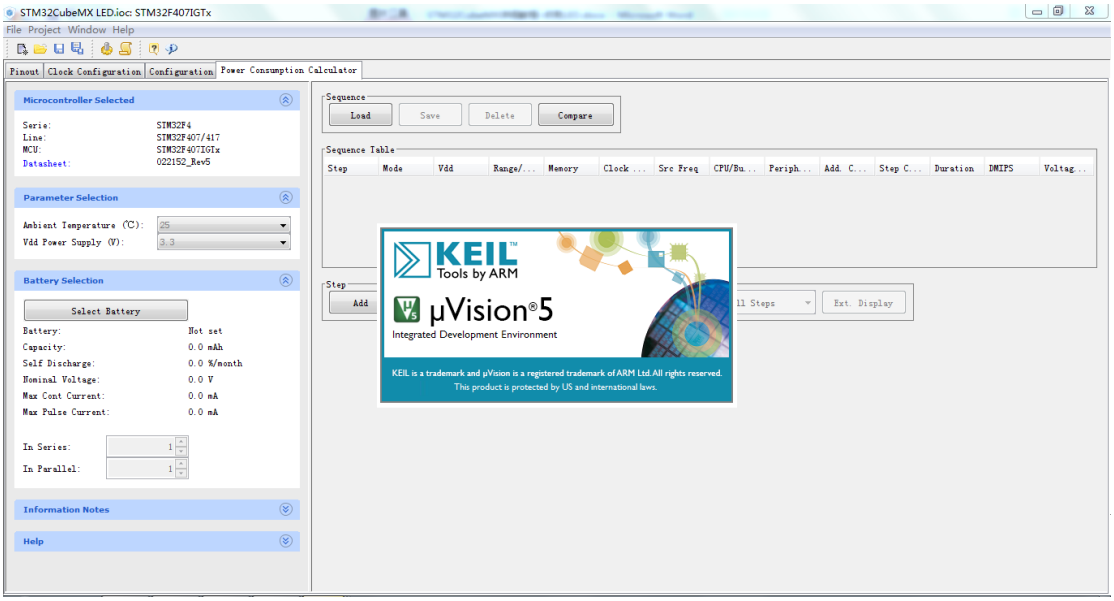
3. 代码生成设置

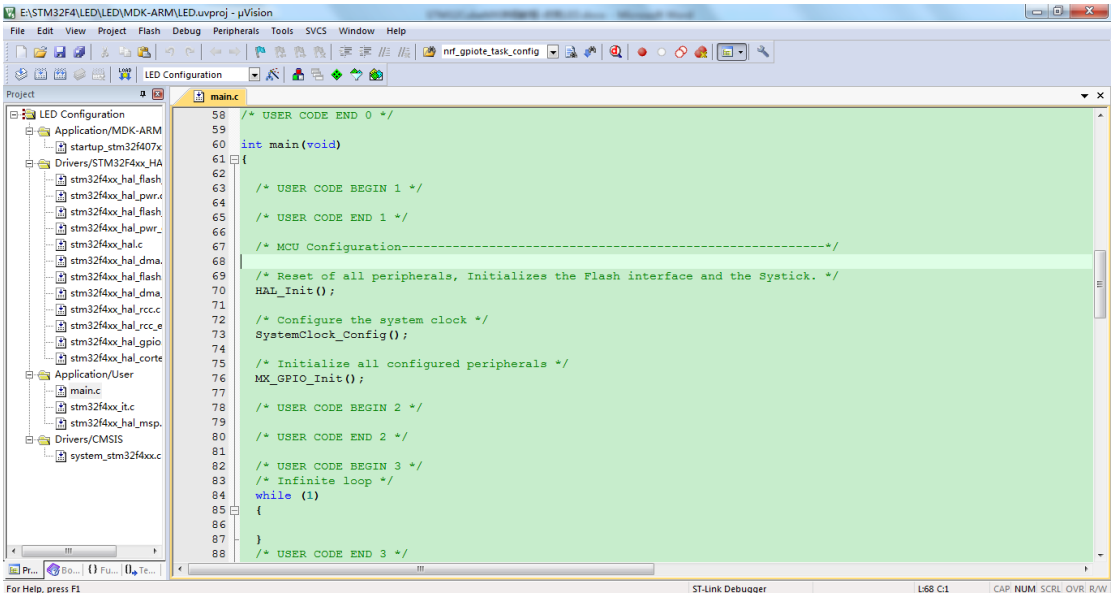

Project Settings
Project
Code Generator
STM32Cube Firmware Library Package
Copy all used libraries into the project folder
Copy only the necessary library files
Add necessary library files as reference in the toolchain project configuration ...
Generated files
Generate peripheral initialization as a pair of '.c/.h' files per IP
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
Ok
Cancel

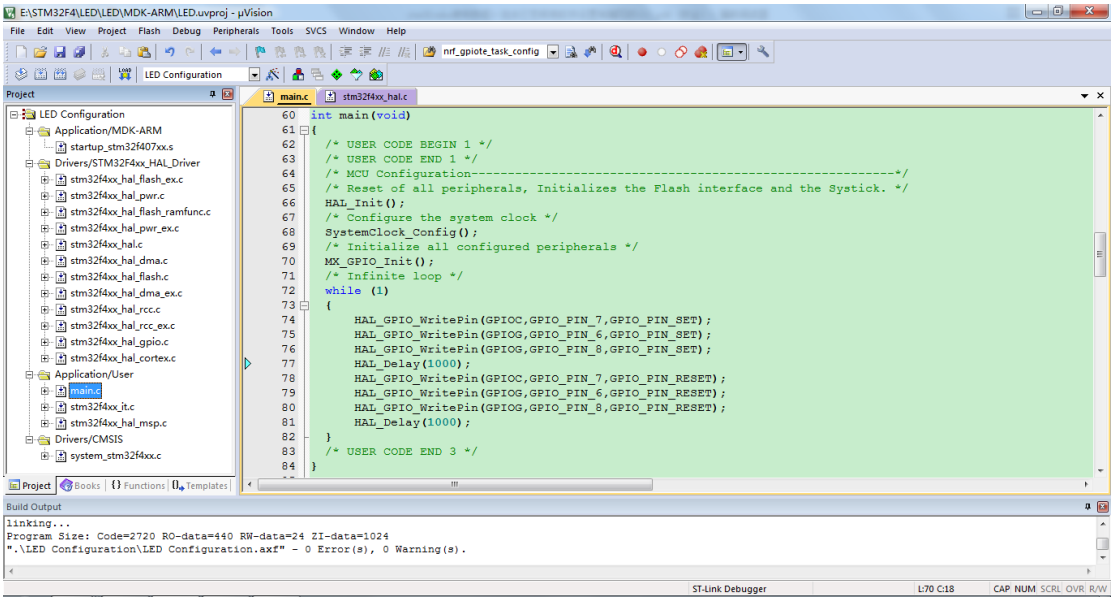
按照上面的勾选即可。


STM32CubeMX LED.ioc: STM32F407IGTx
File Project Window Help
Pinout Clock Configuration Configuration Power Consumption Calculator
Microcontroller Selected
Parameter Selection
Battery Selection
Select Battery
Information Notes
Help
Sequence
Load Save Delete Compare
Sequence Table
Step Mode Vdd Range/... Memory
Generating Keil project...
生成过程有点久，我是不是该换电脑了...


STM32CubeMX LED.ioc: STM32F407IGTx
File Project Window Help
Pinout Clock Configuration Configuration Power Consumption Calculator
Microcontroller Selected
Parameter Selection
Battery Selection
Select Battery
Information Notes
Help
Sequence
Load Save Delete Compare
Sequence Table
Code Generation
Open Project
Close
用上面设置的编译器打开即可

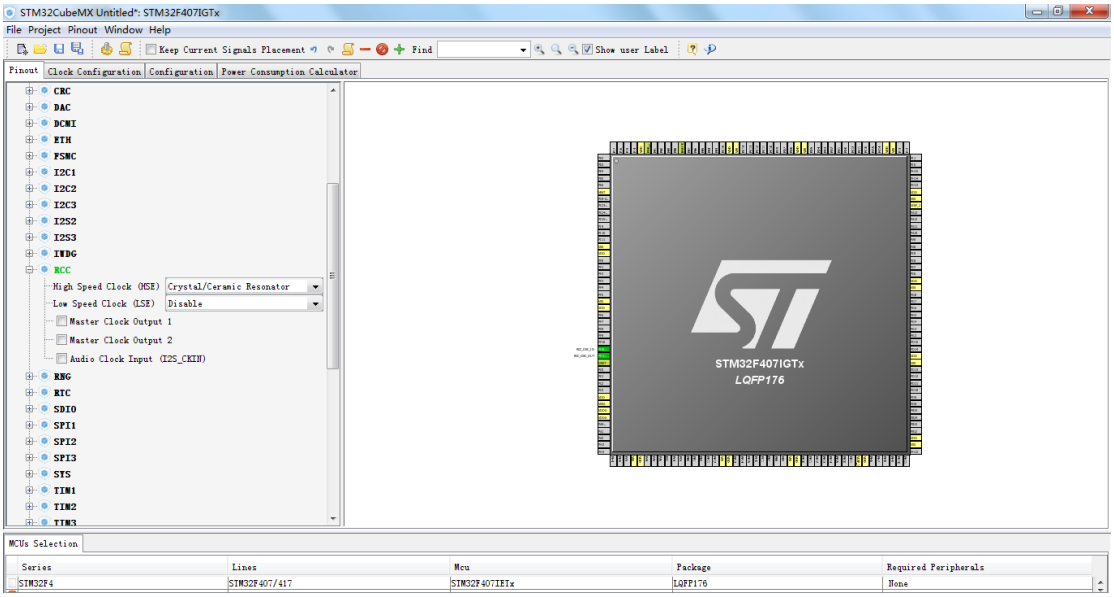

STM32CubeMX LED.ioc: STM32F407IGTx
File Project Window Help
Pinout Clock Configuration Configuration Power Consumption Calculator
Microcontroller Selected
Parameter Selection
Battery Selection
Select Battery
Information Notes
Help
Sequence
Load Save Delete Compare
Sequence Table
KEIL
Tools by ARM
µVision®5
Integrated Development Environment
KEIL is a trademark and µVision is a registered trademark of ARM Ltd. All rights reserved.
This product is protected by US and international laws.

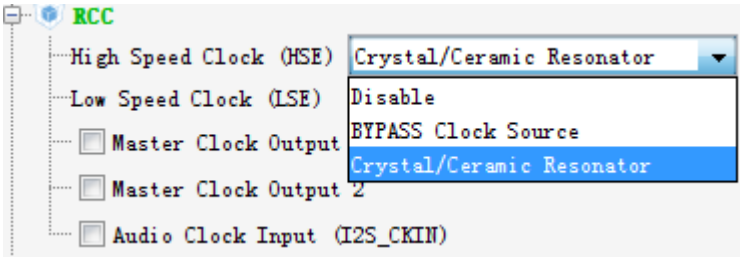




至此，用 STMCubeMX 新建一个点灯工程到此结束。虽然上面的步骤看起来很多，实际上我们只需要用鼠标点击几下即可。生成的工程文件里面，时钟和 GPIO 相关配置代码已经自动生成，是不是很强大很省心呢。在配置正确的情况下，我们的精力得到了释放，可以更加专注了具体功能的实现了。

附录：TMcubeMX 设置 STM32F407 系列 RCC 说明





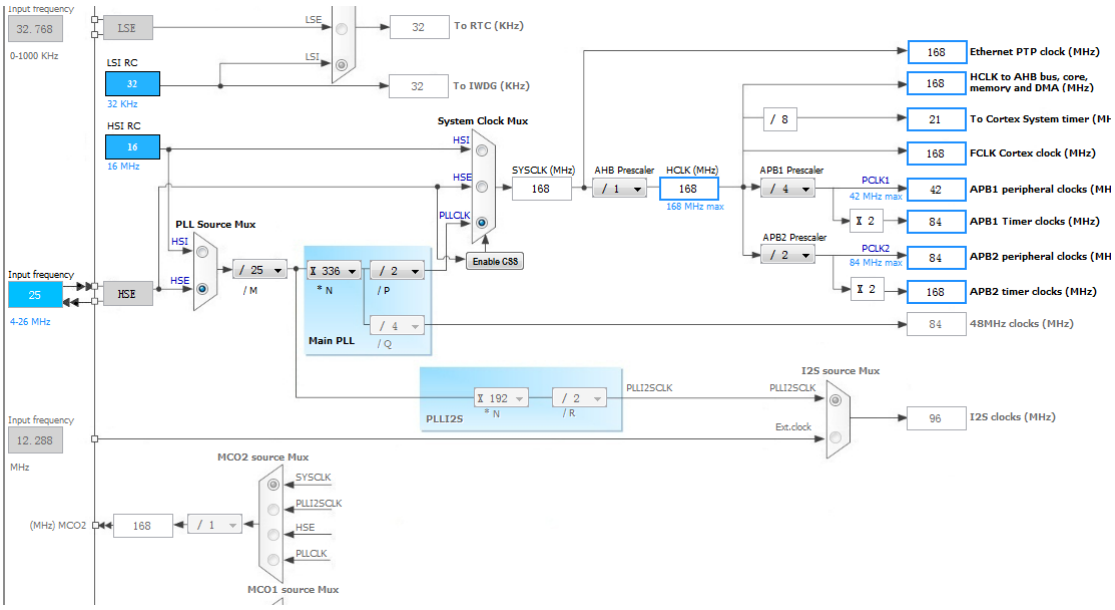
High Speed Clock(HSE):外部高速时钟，其中有三个选项，分别是：

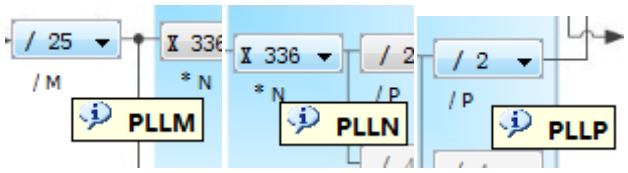
1. Disable:使能外部时钟
2. BYPASS Clock Source:旁路时钟
3. Crystal/Ceramic Resonator: 晶体/陶瓷谐振器（一般选择该选项）

Low Speed Clock(LSE):外部低速时钟。分别是：

- Disable:使能外部时钟
- BYPASS Clock Source:旁路时钟
- Crystal/Ceramic Resonator: 晶体/陶瓷谐振器

总体框图如下：





如上图所示：

M 表示 PLLM；

N 表示 PLLN；

P 表示 PLLP；

那么我们来看看数据手册来确认一下它们为什么要设置成这些数字，设置其他的可不可呢？

下面是 RCC PLL 配置寄存器的相关说明：

6.3.2 RCC PLL 配置寄存器 (RCC_PLLCFGR)

RCC PLL configuration register

偏移地址：0x04

复位值：0x2400 3010

访问：无等待周期，按字、半字和字节访问。

此寄存器用于根据公式配置 PLL 时钟输出：

- $f_{(VCO\ 时钟)} = f_{(PLL\ 时钟输入)} \times (PLLN / PLLM)$
- $f_{(PLL\ 常规时钟输出)} = f_{(VCO\ 时钟)} / PLLP$
- $f_{(USB\ OTG\ FS,\ SDIO,\ RNG\ 时钟输出)} = f_{(VCO\ 时钟)} / PLLQ$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				PLLQ3	PLLQ2	PLLQ1	PLLQ0	Reserved	PLLSRC	Reserved				PLLP1	PLLP0
				rw	rw	rw	rw		rw					rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	PLLN									PLLM5	PLLM4	PLLM3	PLLM2	PLLM1	PLLM0
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:28 保留，必须保持复位值。

位 27:24 **PLLQ**：主 PLL (PLL) 分频系数，适用于 USB OTG FS、SDIO 和随机数发生器时钟 (Main PLL (PLL) division factor for USB OTG FS, SDIO and random number generator clocks)

由软件置 1 或清零，用于控制 USB OTG FS 时钟、随机数发生器时钟和 SDIO 时钟的频率。这些位应仅在 PLL 已禁止时写入。

小心： 为使 USB OTG FS 能够正常工作，需要 48 MHz 的时钟。对于 SDIO 和随即数生成器，频率需要低于或等于 48 MHz 才可正常工作。

USB OTG FS 时钟频率 = VCO 频率 / PLLQ，并且 2 ≤ PLLQ ≤ 15

0000：PLLQ = 0，错误配置

0001：PLLQ = 1，错误配置

0010：PLLQ = 2

0011：PLLQ = 3

0100：PLLQ = 4

...

1111：PLLQ = 15

位 23 保留，必须保持复位值。

位 22 **PLLSRC**：主 PLL(PLL) 和音频 PLL (PLLI2S) 输入时钟源 (Main PLL(PLL) and audio PLL (PLLI2S) entry clock source)

由软件置 1 和清零，用于选择 PLL 和 PLLI2S 时钟源。此位只有在 PLL 和 PLLI2S 已禁止时才可写入。

0：选择 HSI 时钟作为 PLL 和 PLLI2S 时钟输入

1：选择 HSE 振荡器时钟作为 PLL 和 PLLI2S 时钟输入

位 21:18 保留，必须保持复位值。

位 17:16 **PLLP**：适用于主系统时钟的主 PLL (PLL) 分频系数 (Main PLL (PLL) division factor for main system clock)

由软件置 1 和清零，用于控制常规 PLL 输出时钟的频率。这些位只能在 PLL 已禁止时写入。

小心： 软件必须正确设置这些位，使其在此域中不超过 168 MHz。

PLL 输出时钟频率 = VCO 频率 / PLLP 并且 PLLP = 2、4、6 或 8

00：PLLP = 2
01：PLLP = 4
10：PLLP = 6
11：PLLP = 8

位 14:6 **PLLN**：适用于 VCO 的主 PLL (PLL) 倍频系数 (Main PLL (PLL) multiplication factor for VCO)

由软件置 1 和清零，用于控制 VCO 的倍频系数。这些位只能在 PLL 已禁止时写入。写入这些位时只允许使用半字和字访问。

小心： 软件必须正确设置这些位，确保 VCO 输出频率介于 192 和 432 MHz 之间。

VCO 输出频率 = VCO 输入频率 × PLLN 并且 192 ≤ PLLN ≤ 432

000000000：PLLN = 0，错误配置
000000001：PLLN = 1，错误配置
...
011000000：PLLN = 192
...
110110000：PLLN = 432
110110001：PLLN = 433，错误配置
...
111111111：PLLN = 511，错误配置

位 5:0 **PLLM**：主 PLL (PLL) 和音频 PLL (PLLI2S) 输入时钟的分频系数 (Division factor for the main PLL (PLL) and audio PLL (PLLI2S) input clock)

由软件置 1 和清零，用于在 VCO 之前对 PLL 和 PLLI2S 输入时钟进行分频。这些位只有在 PLL 和 PLLI2S 已禁止时才可写入。

小心： 软件必须正确设置这些位，确保 VCO 输入频率介于 1 和 2 MHz 之间。建议选择 2 MHz 的频率，以便限制 PLL 抖动。

VCO 输入频率 = PLL 输入时钟频率 / PLLM 并且 2 ≤ PLLM ≤ 63

000000：PLLM = 0，错误配置
000001：PLLM = 1，错误配置
000010：PLLM = 2
000011：PLLM = 3
000100：PLLM = 4
...
111110：PLLM = 62
111111：PLLM = 63

由上面我们得出这些结论：

PLLM：主 PLL（PLL）和音频 PLL（PLLI2S）输入时钟的分频系数，PLL 的设置是对 PLL 输入时钟源 HSE 进行分频系数，而这个系数在寄存器说明里面特别强调说来了，HSE 进入 PLL 的频率（VCO 输入频率）要介于 1 和 2MHz 之间：

VCO 输入频率 = PLL 输入时钟频率 / PLLM 并且 2 ≤ PLLM ≤ 63

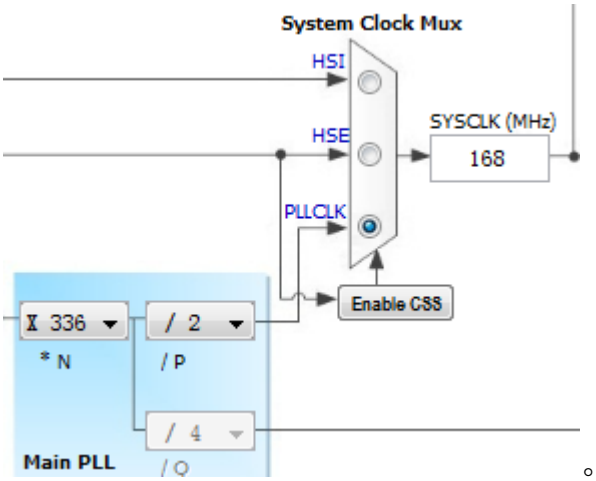
从这里我们就很清楚为啥 PLLM 要设置成 25 了，HSE 输入频率 25MHz 经过 25 分频变成了 1MHz，同理，如何用户的外部晶振为 16MHz，那么 PLLM 则选择 16 或者 8 都可以，建议尽量让 PLL 的输入频率为 1MHz，这样方便产生精确的震荡方波。

PLLN：适用于 VCO 的主 PLL（PLL）倍频系数。这里的计算公式为：VCO 输出频率 = VCO 输入频率 × PLLN 并且 192 ≤ PLLN ≤ 432。

从这里我们只能单纯地知道 PLLN 的范围，但是不知道为什么是 336 而不是

其他的。为了解决这个问题，我们还得了解 PLLP。

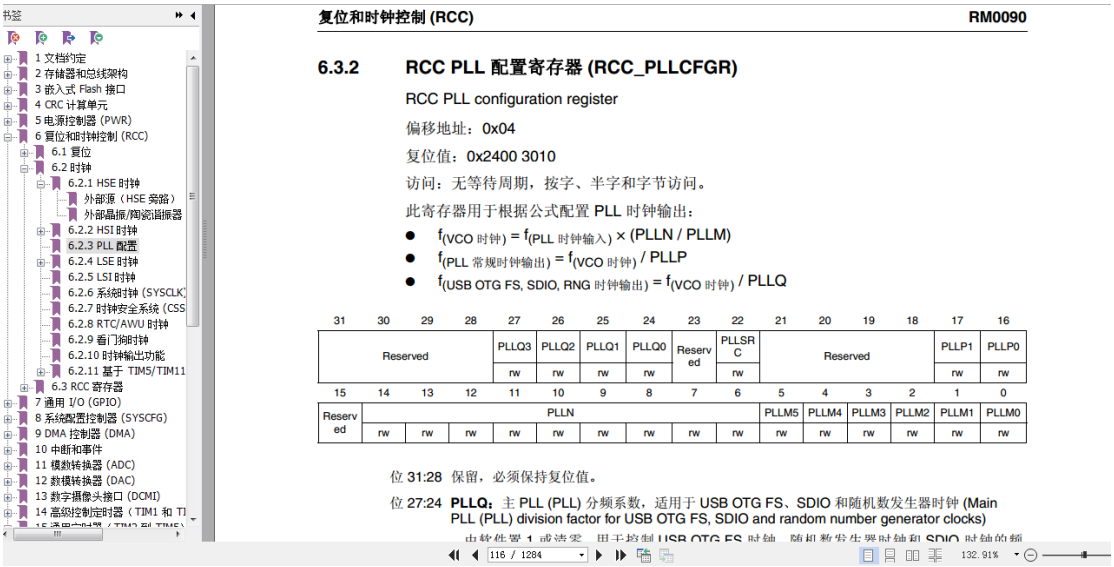
PLLP：适用于主系统时钟的主 PLL（PLL）分频系数。它的计算公式为：PLL 输出时钟频率 = VCO 频率 / PLLP 并且 PLLP = 2、 4、 6 或 8。事实上，大家可以从框图看出来，经过 PLLP 之后就是系统时钟了。



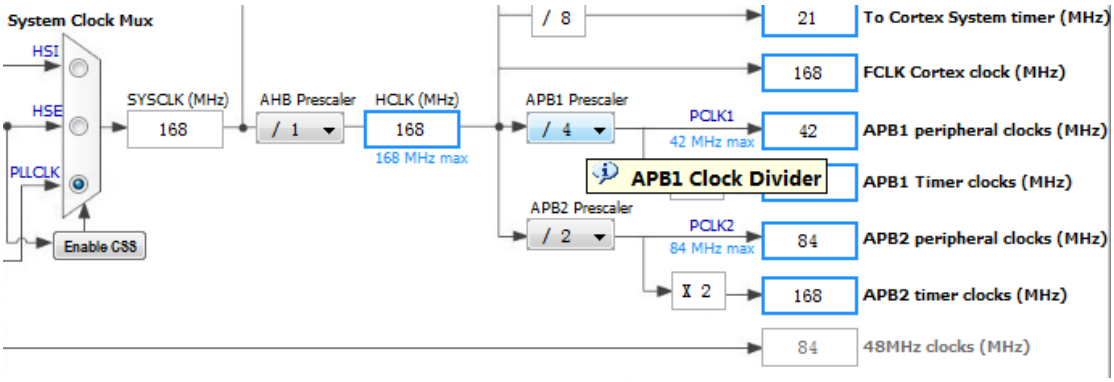
。

根据数据手册得知 STM32F407 系列，系统时钟 HCLK 最大值 = 168 MHz。即 SYSCLK=168M。这里我们倒推回去，168=X/PLLP（单位：MHz），并且 PLLP = 2、4、 6 或 8。当 PLLP=8 是，VCO=1344=1*PLLN，并且 192 ≤ PLLN ≤ 432，显然不成立。我们依次将 PLLP=6、4、2 带入上面的公式进行计算，唯有 2 是满足条件的。当 PLLP 为 2 是，PLLN=336.至此我们得知 M、N、P 的值了。

上面的所有公式在 STM32F4 用户手册都有详细的说明，大家可以自行对照推算一下



那么下面的图示，大家可以根据上面的思路去思考一下。



APB1 Divider 为啥要 4 分频，APB2 Divider 为何要 2 分频，请对照用户手册来比较推算一下。这些总线数值都可以在复位和时钟控制(RCC)这部分可以找到答案。