life.augmented

UM1718
用户手册

STM32CubeMX用于STM32配置和初始化C代码生成

引言

STM32CubeMX是用于STM32微控制器的图形工具。它属于STM32Cube计划（参见*第 1 节*），既可作为独立应用，也可在STM32CubeIDE工具链中使用。

STM32CubeMX有以下主要特性：

- **微控制器选择方便**，覆盖整个STM32产品
- 可从一系列意法半导体的开发板中**选择板子**
- **微控制器配置简单**（引脚、时钟树、外设、中间件）以及生成对应的初始化C代码
- 将以前保存的配置导入新的MCU项目即可**轻松地转换到其他微控制器**
- **将当前配置轻松地导出到兼容的MCU**
- **生成配置报告**
- 为一系列集成开发环境工具链**生成嵌入C项目**　STM32CubeMX项目包括生成的初始化C代码、兼容MISRA 2004的HAL驱动程序、用户配置所需的中间件协议栈，以及在选择的IDE中打开和建立项目的所有相关文件。
- 用户定义应用序列的**功耗计算**
- **自动更新**功能确保用户随时更新STM32CubeMX
- 下载和更新用户应用开发所需的STM32Cube嵌入式软件（关于STM32Cube嵌入式软件产品的详情，参见*附录E*）

虽然STM32CubeMX提供了一个用户界面并且生成的C代码兼容STM32 MCU设计和固件解决方案，但用户仍需要参考产品技术文档，以了解关于微控制器外设和固件实际实现的详情。

以下文档可从*www.st.com*获得：

- STM32微控制器参考手册和数据手册
- STM32F0（UM1785）、STM32F1（UM1850）、STM32F2（UM1940）、STM32F3（UM1786）、STM32F4（UM1725）、STM32F7（UM1905）、STM32G0（UM2303）、STM32G4（UM2570）、STM32H7（UM2217）、STM32L0（UM1749）、STM32L1（UM1816）、STM32L4/L4+（UM1884）、STM32L5（UM2659）、STM32MP1（https://wiki.st.com/stm32mpu）和STM32WB（UM2442）的STM32Cube HAL/LL驱动程序用户手册。

目录

表格索引

图片目录

1 STM32Cube 概述

STMCube 源自意法半导体，旨在通过减少开发工作量、时间和成本，让开发人员的生活更轻松。 STM32Cube 基于 Arm®(a) Cortex® 内核，涵盖 STM32 微控制器整个产品系列。

STM32Cube 包括：

- 图形软件配置工具STM32CubeMX，可通过图形向导生成初始化C代码。
- 综合的嵌入式软件平台，并针对每个系列提供单独的库文件（例如STM32CubeF2用于STM32F2系列，STM32CubeF4用于STM32F4系列）
  - STM32抽象层嵌入式软件STM32Cube HAL，确保在STM32各个产品之间实现最大限度的可移植性
  - 底层API（LL）提供了一个专家级的快速轻量级层，它比HAL更靠近硬件。
  - 一套一致的中间件，比如RTOS、USB、TCP/IP
  - 提供了一套完整示例以及嵌入式软件工具。

arm

a. Arm是Arm Limited（或其子公司）在美国和/或其他地区的注册商标。

2 STM32CubeMX入门

2.1 原理

客户需要快速识别最符合其要求的MCU（核心架构、特性、存储器容量、性能.....）。开发板设计人员主要关注的是如何针对板布局优化微控制器引脚配置并满足应用要求（选择外设工作模式）；嵌入式系统开发人员更感兴趣的是为特定目标设备开发新应用，以及将现有设计迁移至不同的微控制器。

迁移到新平台并将C代码更新到新固件驱动程序需要耗费时间，这可能会导致项目出现不必要的延迟。STM32CubeMX 基于STM32Cube计划而开发，旨在满足客户关键要求，从而最大限度地重复使用软件并缩短创建目标系统的时间：

- STM32Cube固件解决方案提出了跨STM32产品的通用硬件抽象层API，可实现软件重复使用和应用程序设计的可移植性。
- 凭借STM32CubeMX内置的STM32微控制器、外设和中间件（LwIP和USB通信协议栈，用于小型嵌入式系统的FatFs文件系统，FreeRTOS）知识，迁移时间得到了优化。

STM32CubeMX 图形界面执行以下功能：

- 快速简便地配置所选外设和中间件的MCU引脚、时钟树和工作模式
- 为开发板设计人员生成引脚配置报告
- 生成一个完整项目，包含所有必需的库和初始化C代码，以在用户定义的工作模式下设置设备。可以在选定的应用开发环境中直接打开项目（适用于一系列支持的IDE），以继续进行应用程序开发（参见*图 1*）。

在配置过程中，STM32CubeMX检测冲突和无效设置，并使用有意义的图标和有用的工具提示突出显示这些冲突和设置。

图1. STM32CubeMX C代码生成流程概述

2.2 主要特性

STM32CubeMX 具备以下特性：

- **项目管理**

  STM32CubeMX 允许用户创建、保存和加载先前保存的项目：

  – 当启动STM32CubeMX时，用户可以选择创建新项目或加载先前保存的项目。

  – 项目保存操作可将项目内执行的用户设置和配置保存在.ioc文件中，在STM32CubeMX下次加载项目时便可使用该文件。

  STM32CubeMX 允许用于在新项目中导入先前保存的项目。

  STM32CubeMX 项目具有两个配置选项：

  – 仅MCU配置：.ioc文件保存在专用项目文件夹中。

  – 具有C代码生成的MCU配置：此时，.ioc文件与生成的源C代码一起保存在专用项目文件夹中。每个项目只能有一个.ioc文件。

- **轻松选择MCU和STMicroelectronics板**

  在开始新项目时，会打开一个专用窗口，用户可从STM32产品中选择微控制器或STMicroelectronics板。提供不同的筛选选项，以简化MCU和开发板选择。通过与竞争对手产品系列进行特性比较，还可以通过“交叉选择器”选项卡选择MCU。比较标准可以调整。

- **轻松执行引脚排列配置**

  – 在**“引脚排列”**视图中，用户可以从列表中选择外设，并配置应用程序所需的外设模式。STM32CubeMX 相应地对引脚进行分配和配置。

  – 对高级用户而言，还可以使用**“引脚布局”**视图，直接将外设功能映射到物理引脚。信号可以锁定在引脚上，以防止STM32CubeMX冲突解算器将信号移动到另一个引脚。

  – 引脚排列配置可以导出为.csv文件。

- **完整的项目生成**

  项目生成包括一组IDE的引脚排列、固件和中间件初始化C代码。它基于STM32Cube嵌入式软件库。可以执行以下操作：

  – 用户可以从先前定义的引脚排列开始，继续配置中间件、时钟树、服务（RNG、CRC等）和外设参数。STM32CubeMX 生成相应的初始化C代码。由此，用户可获得一个项目目录，包括生成的main.c文件和用于配置和初始化的C头文件、必要的HAL和中间件库的副本，以及用于所选IDE的特定文件。

  – 用户可以在用户专用扇区中添加用户定义的C代码，从而修改生成的源文件。STM32CubeMX确保在下一次C代码生成时保留用户C代码（如果用户C代码不再与当前配置相关，则对其添加注释）。

- STM32CubeMX 可以使用用户定义的freemarker .ftl模板文件生成用户文件。
- 在“**项目设置**”菜单中，用户可以选择必须为哪些开发工具链（IDE）生成 C 代码。 STM32CubeMX 确保将 IDE 相关的项目文件添加到项目文件夹中，以便可以将项目作为第三方 IDE 中的新项目（IAR™ EWARM、Keil™ MDK-ARM、Atollic® TrueSTUDIO® 和用于 STM32 的系统工作台）直接导入。

- **功耗计算**

  用户可以首先选择微控制器产品编号和电池类型，进而定义表示应用生命周期和参数的一系列步骤（频率选择、使能的外设、步长持续时间）。STM32CubeMX 功耗计算器返回相应的功耗和电池寿命估算值。

- **时钟树配置**

  STM32CubeMX 提供了时钟树的图示，可以参阅设备参考手册。用户可以更改默认设置（时钟源、预分频器和频率值）。然后相应地更新时钟树。使用工具提示突出显示无效的设置和限制。使用求解器功能可以解决时钟树配置冲突。当无法找到给定用户配置的完全匹配时，STM32CubeMX提出最接近的解。

- **自动更新STM32CubeMX和STM32Cube MCU软件包**

  STM32CubeMX 附带更新程序机制，可配置为自动或按需检查更新。它支持STM32CubeMX自动更新以及STM32Cube固件库软件包的更新。更新程序机制还允许删除先前安装的软件包。

- **报告生成**

  可以生成.pdf和.csv报告，用于记录用户配置工作。

- **图形仿真器**

  对于具有图形功能的微控制器，STM32CubeMX允许用户模拟图形配置并调整图形参数以优化性能。一旦结果符合要求，就可以相应地调整当前项目配置。

- **支持CMSIS-Pack格式的嵌入式软件包**

  STM32CubeMX可以获取和下载以CMSIS-Pack格式提供的嵌入式软件包的更新。然后可以将属于这些新版本的所选软件组件添加到当前项目中。

- **上下文帮助**

  将鼠标悬停在“内核”、“系列”、“外设”和“中间件”上，可以显示上下文帮助窗口。它们提供了简短介绍，以及与所选项目相对应的相关文档的链接。

2.3 规则和限制

- C代码生成仅涵盖外设和中间件初始化。它基于STM32Cube HAL固件库。
- STM32CubeMX C代码生成仅涵盖用于外设和中间件组件的初始化代码，这些外设和中间件使用STM32Cube嵌入式软件包中包含的驱动程序。尚不支持某些外设和中间件组件的代码生成。
- 有关引脚分配规则的说明，请参见*附录A*。
- 有关STM32CubeMX C代码生成的设计选择和限制的说明，请参见*附录B*。

3 安装和运行 STM32CubeMX

3.1 系统要求

3.1.1 支持的操作系统和架构

- Windows® 7：32 位 (x86), 64 位 (x64)
- Windows® 8：32 位 (x86), 64 位 (x64)
- Windows® 10：32 位 (x86), 64 位 (x64)
- Linux®：32 位 (x86) 和 64 位 (x64)（已在 RedHat、 Ubuntu 和 Fedora 上测试）
  由于STM32CubeMX是32位应用程序，因此部分版本的Linux 64位发行版需要安装32位兼容软件包，例如ia32-libs。
- macOS®：64 位 (x64)（已在 OS X El Capitan 和 Sierra 上测试）

3.1.2 内存必要条件

- 建议最低RAM：2 GB。

3.1.3 软件要求

必须安装Java™运行环境1.8。

请注意，不支持Java 9和Java 10，而且使用Java 11可进行的验证是有限的。

在与“Oracle JDK 8公开更新结束”有关的Oracle公告之后，可以通过 https://adoptopenjdk.net/访问OpenJDK 8。

3.2 安装/卸载 STM32CubeMX 独立版本

3.2.1 安装STM32CubeMX 独立版本

如要安装STM32CubeMX，需遵循以下步骤：

1. 从www.st.com/stm32cubemx下载STM32CubeMX安装包。
2. 将stm32cubemx.zip整个软件包提取（解压缩）到同一目录中。
3. 检查您的访问权限并启动安装向导：

   在 Windows® 上：

   a) 确保您拥有管理员权限。

   b) 双击SetupSTM32CubeMX-VERSION.exe文件，启动安装向导。

   在 Linux® 上：

   a) 确保您具有目标安装目录的访问权限。您可以将安装程序作为root（或sudo）运行，以在共享目录中安装STM32CubeMX。

   b) 执行“chmod 777 SetupSTM32CubeMX-5.0.0.linux”以更改属性，从而使文件可执行。

c) 双击SetupSTM32CubeMX-VERSION.linux文件，或从控制台窗口启动。

在 macOS® 上：

a) 确保您拥有管理员权限。

b) 双击SetupSTM32CubeMX-VERSION应用文件，启动安装向导。
如果出现错误，请使用以下命令启动exe文件：
sudo java -jar SetupSTM32CubeMX-4.14.0.exe。

4. 在Windows上成功安装STM32CubeMX后，桌面上会显示STM32CubeMX图标，可以在“**程序**”菜单中找到STM32CubeMX应用程序。STM32CubeMX.ioc文件显示为多维数据集图标。双击这些文件，使用STM32CubeMX打开文件。

5. 从磁盘中删除zip的内容。

注： *如果未安装适当版本的Java™运行时环境（版本1.7_45或更高版本），则向导建议下载并停止。Java™安装完成后重新启动STM32CubeMX安装。安装JRE时，如有问题，请参见第 18 节：FAQ。*

*在Windows上操作时，“**程序**”菜单仅会使能STM32CubeMX的最新安装版本。如果指定了不同的安装文件夹，则可以在PC上保留先前的版本（不推荐）。否则，新安装将覆盖先前的版本。*

3.2.2 从命令行安装STM32CubeMX

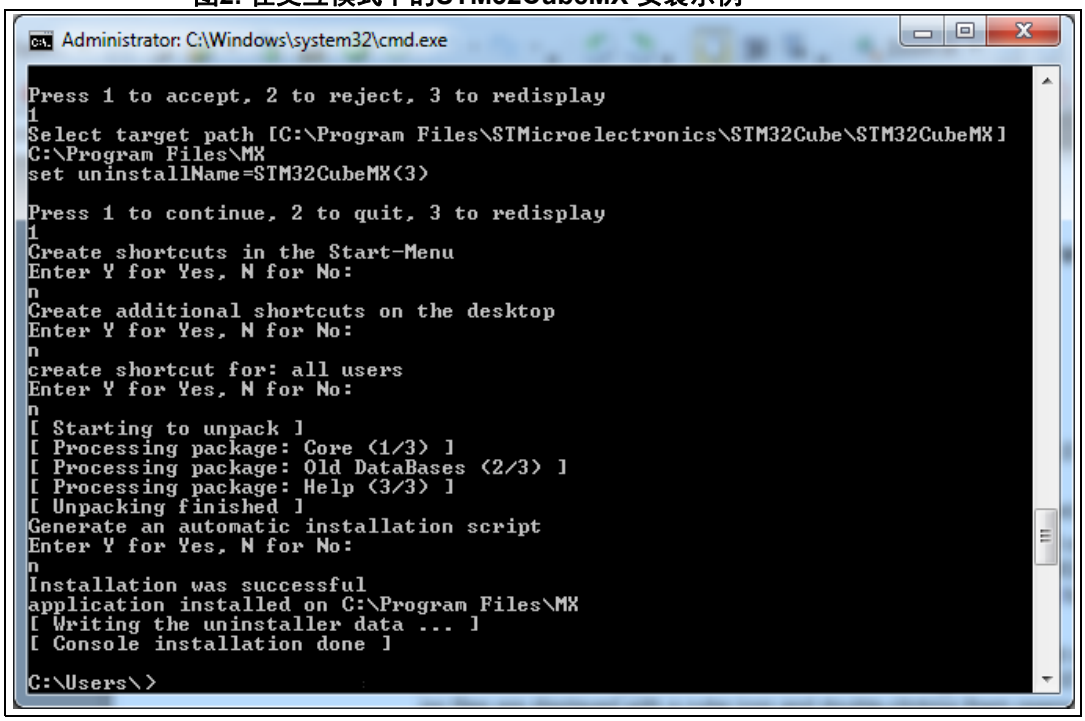
有两种方法从控制台窗口启动安装：在控制台交互模式下或通过脚本。

交互模式

要执行交互式安装，请键入以下命令：
java –jar SetupSTM32CubeMX-4.14.0.exe –console

在每个安装步骤中，都需要请求得到应答（参见*图 2*）。

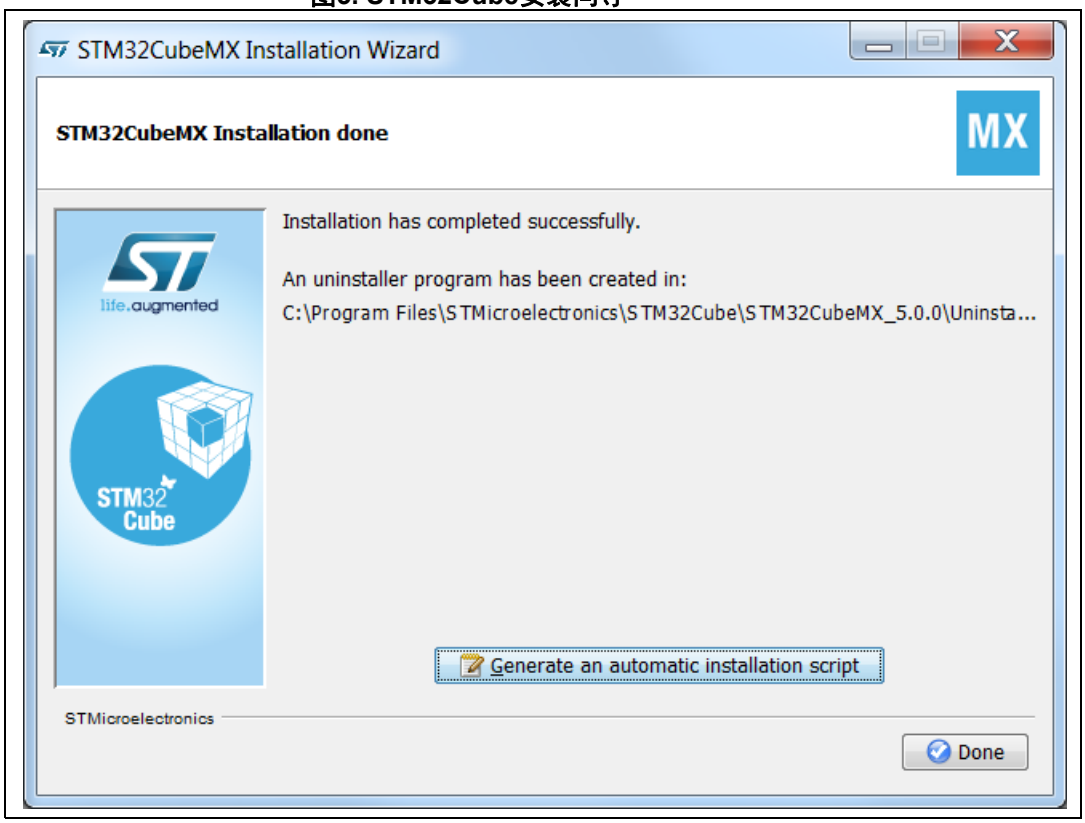
图2. 在交互模式下的STM32CubeMX 安装示例



自动安装模式

在安装结束时，使用STM32CubeMX图形向导或控制台模式，可以生成包含用户首选项的自动安装脚本（参见*图 3*）。

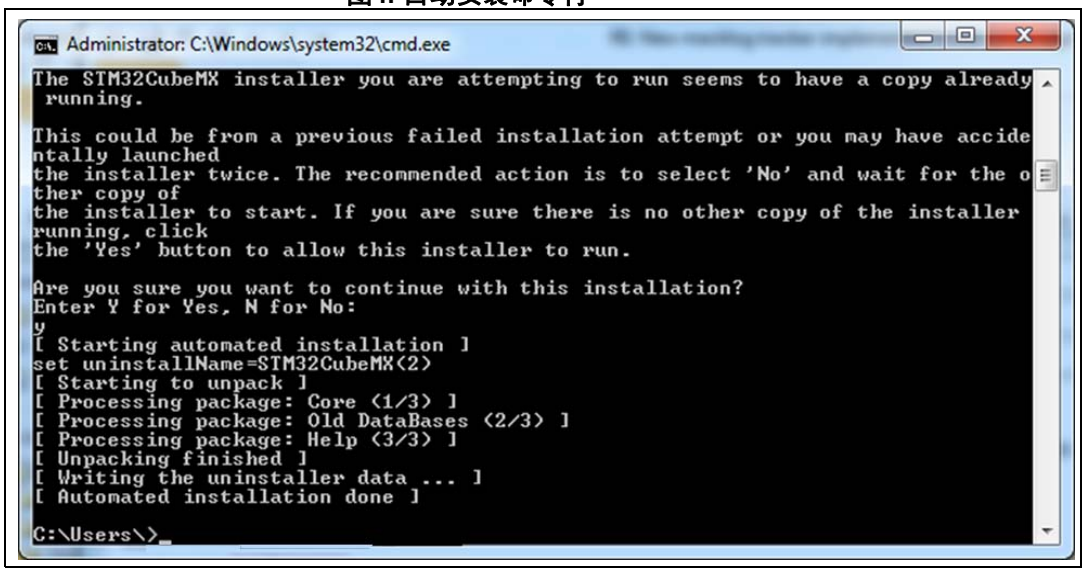
图3. STM32Cube安装向导



然后，您可以通过键入以下命令来启动安装：

java –jar SetupSTM32CubeMX-4.14.0.exe auto-install.xml

图4. 自动安装命令行



3.2.3 卸载STM32CubeMX独立版本

在macOS®上卸载STM32CubeMX

如要在macOS上卸载 STM32CubeMX，使用以下命令行：

java -jar <STM32CubeMX installation path>/Uninstaller/uninstaller.jar。

在Linux®上卸载STM32CubeMX

在Linux上有三种卸载STM32CubeMX的方法：

- 使用以下命令行

  java -jar <STM32CubeMX installation path>/Uninstaller/uninstaller.jar。
- 通过Windows Explorer窗口：
  a) 使用文件资源管理器。
  b) 转到STM32CubeMX安装的卸载程序目录。
  c) 双击启动，卸载桌面快捷方式。

在Windows®上卸载STM32CubeMX

在Windows上有三种卸载STM32CubeMX的方法：

- 使用以下命令行

  java -jar <STM32CubeMX installation path>/Uninstaller/uninstaller.jar。
- 通过Windows Explorer窗口：
  a) 使用文件资源管理器。
  b) 转到STM32CubeMX安装的卸载程序目录。
  c) 双击启动，卸载桌面快捷方式。
- 通过**Windows控制**面板：
  a) 从**Windows控制**面板中选择**“程序和功能”**，显示计算机上安装的程序列表。
  b) 右击 STM32CubeMX 并选择**“卸载”**。

3.3 启动STM32CubeMX

3.3.1 作为独立应用程序运行

如要将STM32CubeMX 作为独立应用程序运行于Windows：

- 在“程序文件” > ST Microelectronics > STM32CubeMX中选择STM32CubeMX。
- 或者双击桌面上的 STM32CubeMX图标。

如要将 STM32CubeMX作为独立应用程序运行于Linux，从STM32CubeMX 安装目录启动可执行的STM32CubeMX 。

如要将 STM32CubeMX 作为独立应用程序运行于 macOS®，从启动面板启动 STM32CubeMX 应用程序。

注： *在 macOS® 上不存在 STM32CubeMX 桌面图标。*

3.3.2 在命令行模式下运行STM32CubeMX

为了便于与其他工具集成，STM32CubeMX提供了命令行模式。使用一组命令，您可以：

- 加载MCU
- 加载现有配置
- 保存当前配置
- 设置项目参数，生成对应代码
- 根据模板生成用户代码
- 加载通过其产品编号标识的板
- 刷新嵌入式软件包列表（软件包和STM32Cube MCU软件包），然后安装/删除软件包
- 选择其他软件（软件包）组件以添加到项目中。

有三种可用的命令行模式：

- 如要在交互式命令行模式下运行STM32CubeMX，请使用以下命令行：
  - 在Windows上：

    ```
    java -jar STM32CubeMX.exe –i
    ```

  - 在 Linux® 和 macOS® 上：

    ```
    java -jar STM32CubeMX –i
    ```

  然后显示“MX>”提示，表明应用程序已准备好接受命令。
- 如要在从脚本获得命令的命令行模式下运行STM32CubeMX，请使用以下命令行：
  - 在Windows上：

    ```
    java -jar STM32CubeMX.exe –s <脚本文件名>
    ```

  - 在Linux和macOS上：

    ```
    java -jar STM32CubeMX –s <脚本文件名>
    ```

  必须在脚本文件中列出要执行的所有命令。脚本文件内容的示例如下所示：

  ```
  load STM32F417VETx
  project name MyFirstMXGeneratedProject
  project toolchain "MDK-ARM v4"
  project path C:\STM32CubeProjects\STM32F417VETx
  project generate
  exit
  ```

- 如要在从脚本获得命令且并无UI的命令行模式下运行STM32CubeMX，请使用以下命令行：
  - 在Windows上：

    java -jar STM32CubeMX.exe –q <脚本文件名>
  - 在Linux和macOS上：

    java -jar STM32CubeMX –q <脚本文件名>

  同样，用户可以在显示MX提示时输入命令。

如需可用的命令列表，请参见*表 1*。

表1. 命令行摘要

命令行	目的	示例
help	显示可用的命令列表。	help
swmgr刷新	刷新可下载的嵌入式软件包版本列表。	swmgr refresh
swmgr install stm32cube_<series> _<version> <license-mode (accept|ask)>	安装指定的STM32Cube MCU软件包版本。 第二个参数许可证模式是强制性的，但对于随附许可证的软件包而言确实很重要： – **接受**：自动接受许可证。 – **询问**：许可证在弹出窗口中显示，供用户接受。	swmgr install stm32cube_f1_1.8.0 accept
swmgr remove stm32cube_<series> _<version>	删除指定的STM32Cube MCU软件包版本。	swmgr remove stm32cube_f1_1.8.0
swmgr install <packVendor>.<packName>. <packVersion> <license-mode (accept|ask)>	安装指定的软件包版本。 第二个参数许可证模式是强制性的，但对于随附许可证的软件包而言确实很重要： – **接受**：自动接受许可证。 – **询问**：许可证在弹出窗口中显示，供用户接受。	swmgr install <packVendor>.<packName>. X-CUBE-NFC4.1.4.1 ask
<packVersion> <license-mode (accept|ask)>	删除指定的软件包版本。	swmgr remove STMicroelectronics. X-CUBE-BLE1.4.2.0
swmgr install <filename path> <license-mode (accept|ask)	安装嵌入式软件包。	swmgr install "C:\repo\packs\STMicroelectronics. X-CUBE-BLE1.4.2.0.pack” accept
pack enable <vendor> <pack>[/bundle] <version> <class> <group>[/subgroup] [variant]	选择要添加到项目中的软件包组件。 第二个和/或第五个参数中存在的“/”分别表示明确提及的套件和/或子组（参考：ARM CMSIS pack pdsc格式）。 要找出要启用的组件的软件包/套件/类别/组/子组名称，请选择该组件，然后从“其他软件”窗口中单击“隐藏/显示详细信息”。	pack enable STMicroelectronics “X-CUBE-BLE1/BlueNRG-MS” 1.0.0 "Wireless" "Controller"
软件包验证	自上次调用“pack validate”命令以来，在项目中应用了所有已启用的软件包组件。	pack validate

表1. 命令行摘要（续）

命令行	目的	示例
load <mcu>	加载所选的MCU。	load STM32F101RCTx load STM32F101Z(F-G)Tx
load <board part number> <allmodes|nomode>	在所有外设均配置为默认模式（所有模式）或没有任何外设配置（无模式）的情况下加载所选板。	loadboard NUCLEO-F030R8 allmodes loadboard NUCLEO-F030R8 nomode
config load <filename>	加载先前保存的配置。	config load C:\Cube\ccmram\ccmram.ioc
config save <filename>	保存当前配置。	config save C:\Cube\ccmram\ccmram.ioc
config saveext <filename>	保存当前配置及所有参数，包括已将值保持为默认值的参数（用户不可修改）。	config saveext C:\Cube\ccmram\ccmram.ioc
config saveas <filename>	以新名称保存当前项目。	config saveas C:\Cube\ccmram2\ccmram2.ioc
csv pinout <filename>	将当前引脚配置导出为csv文件。（稍后）可以将此文件导入板布局工具。	Csv pinout mypinout.csv
script <filename>	运行脚本文件的所有命令。每行必须有一个命令。	script myscript.txt
project couplefilesbyip <0|1>	此代码生成选项允许用于在0或1之间进行选择，0表示在main中生成外设初始化，1表示在专用.c/.h文件中生成每个外设初始化。	project couplefilesbyip 1
setDriver <Peripheral Name> <HAL | LL>	对于支持的系列，STM32CubeMX可以基于LL驱动程序或HAL驱动程序生成外设初始化代码。 该命令行使用户可以为每个外设在基于HAL的代码生成和基于LL的代码生成之间进行选择。 默认情况下，代码生成基于HAL驱动程序。	setDriver ADC LL setDriver I2C HAL
generate code <path>	仅生成“STM32CubeMX已生成”代码，而不是包含STM32Cube固件库和工具链项目文件的完整项目。 如要生成项目，请使用“project generate”。	generate code C:\mypath
set tpl_path <path>	设置包含.ftl用户模板文件的源文件夹的路径。 存储在此文件夹中的所有模板文件都用于代码生成。	set tpl_path C:\myTemplates\
set dest_path <path>	设置目标文件夹的路径，该文件夹将保存根据用户模板生成的代码。	set dest_path C:\myMXProject\inc\
get tpl_path	获取用户模板源文件夹的路径名	get tpl_path
get dest_path	获取用户模板目标文件夹的路径名。	get dest_path

表1. 命令行摘要（续）

命令行	目的	示例
project toolchain <toolchain>	指定待用于项目的工具链。 使用“project generate”命令为该工具链生成项目。	project toolchain EWARM project toolchain “MDK-ARM V4” project toolchain “MDK-ARM V5” project toolchain TrueSTUDIO project toolchain SW4STM32
project name <name>	指定项目名称。	project name ccmram
project path <path>	指定用于生成项目的路径。	project path C:\Cube\ccmram
project generate	生成完整项目。	project generate
exit	结束STM32CubeMX进程。	exit

3.4 使用STM32CubeMX进行更新

STM32CubeMX 执行机制，用于访问互联网，以及：

- 下载基于 Arm® CMIS 包格式的嵌入式软件包：STM32Cube MCU 软件包 （完整版和补丁）和第三方软件包 （.pack）
- 管理用户定义的第三方包列表
- 检查STM32CubeMX和嵌入式软件包更新
- 执行 STM32CubeMX的自动更新
- 刷新STM32 MCU的描述和文档。

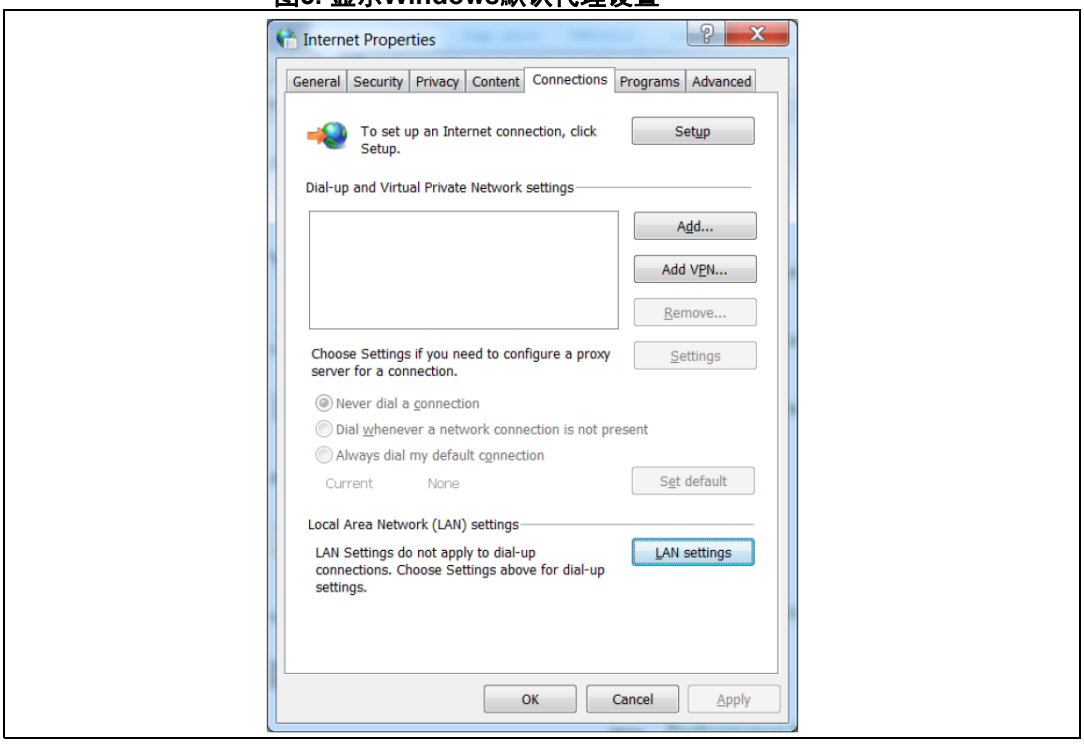
“帮助”菜单下提供了安装和更新相关的子菜单。

也可以在无法连接互联网的计算机上执行离线更新（参见*第 3.4.2 节*）。通过浏览文件系统并选择可用的STM32Cube MCU软件包，即可实现。

如果运行STM32CubeMX的PC使用代理服务器连接到计算机网络，则STM32CubeMX需要连接到该服务器之后才能访问互联网，获取自动更新并下载固件包。有关该连接配置的描述，请参见*第 3.4.1 节*。

如要查看Windows默认代理设置，请从“控制面板”中选择“Internet选项”，然后从**“连接”**选项卡中选择“LAN设置”（参见*图 5*）。

图5. 显示Windows默认代理设置



提供多种代理类型，包括不同的计算机网络配置：

- 无代理：应用程序直接访问Web（Windows默认配置）。
- 无需登录名/密码的代理
- 使用登录名/密码的代理：使用互联网浏览器时，会打开一个对话框，提示用户输入其登录名/密码。
- 使用登录名/密码的Web代理：使用互联网浏览器时，会打开一个网页，提示用户输入其登录名/密码。

如有必要，请与IT管理员联系，获取代理信息（代理类型、http地址、端口）。

STM32CubeMX 不支持Web代理。此时，用户无法受益于更新机制，必须从http://www.st.com/stm32cube手动复制STM32Cube MCU软件包到存储库。为此，需遵循以下步骤：

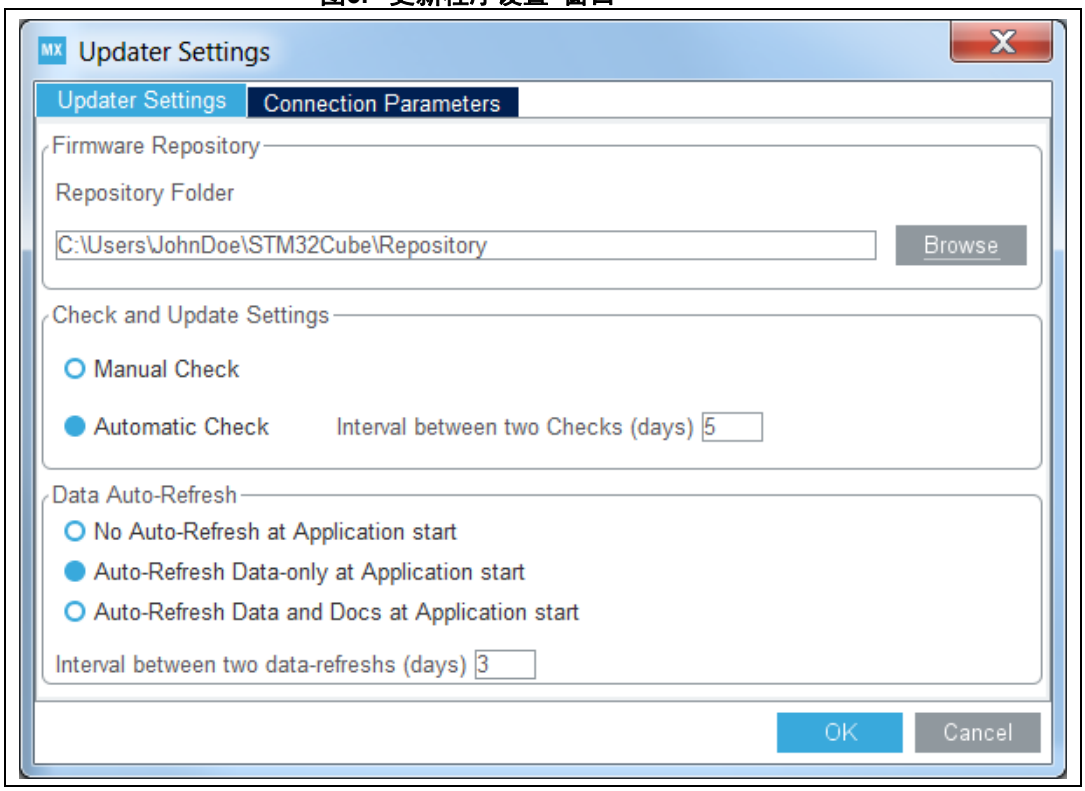
1. 访问http://www.st.com/stm32cube，从“*相关的软件*”部分下载相关的STM32Cube MCU软件包。
2. 将zip包解压缩到STM32Cube存储库。在**“更新程序设置”**选项卡中找到默认存储库文件夹位置，如*图 6*所示（您可能需要更新默认文件夹位置才能使用其他位置或名称）。

3.4.1 更新程序配置

要执行STM32Cube新库包安装或更新，必须按如下方式配置该工具：

1. 选择**“帮助 > 更新程序设置”**，打开**“更新程序设置”**窗口。
2. 在**“更新程序设置”**选项卡（参见*图 6*）
   a) 指定用于存储已下载软件包的存储库目标文件夹。
   b) 启用/禁用自动检查更新。

图6. “更新程序设置”窗口



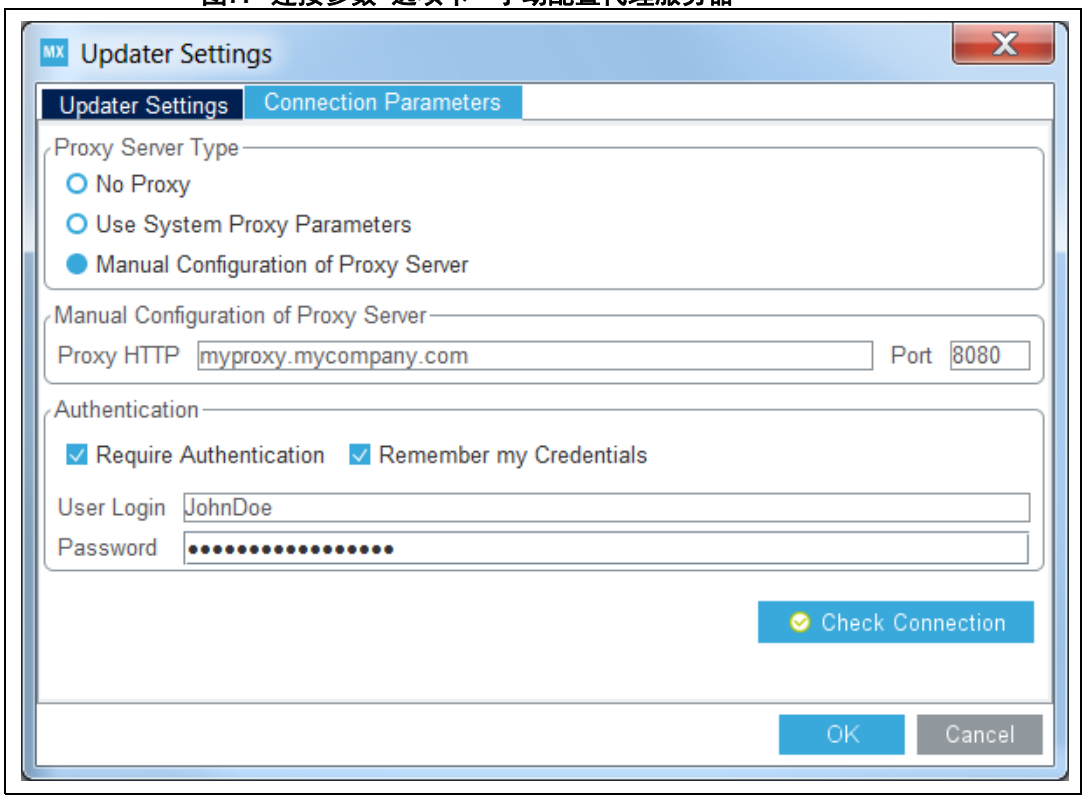
3. 在**“连接参数”**选项卡中，选择以下一种代理类型，指定适合您网络配置的代理服务器设置（参见*图 7*）：
   - 无代理
   - 使用系统代理参数

     在Windows上，从PC系统设置中获取代理参数。

     如果使用无需登录名/密码的代理服务器配置，请取消勾选“需要身份验证”。
   - 手动配置代理服务器

     输入代理服务器的http地址和端口号。如果使用无需登录名/密码的代理服务器配置，请输入登录名/密码信息，或取消勾选“需要身份验证”。
4. 或者取消勾选**“记住我的凭据”**，防止STM32CubeMX将加密的登录名/密码信息保存在文件中。这意味着每次启动STM32CubeMX时都要重新输入登录名/密码信息。
5. 点击**“检查连接”**按钮，验证连接是否有效。出现绿色复选标记，确认连接正常运行

图7. “连接参数”选项卡 - 手动配置代理服务器



6. 选择**“帮助”>“安装新库”**子菜单，选择待安装的软件包列表。
7. 如果该工具配置为手动检查，请选择**“帮助”>“检查更新”**，查找可安装的新工具版本或固件库补丁。

3.4.2 安装STM32 MCU软件包

要下载新的STM32 MCU软件包，请按照以下步骤操作：

1. 选择**“帮助”>“管理嵌入式软件包”**，打开**“嵌入式软件包管理器”**（参见*图 8*），或使用主页上的“安装/删除”键。

   “展开/折叠”按钮 分别用于展开/折叠软件包列表。

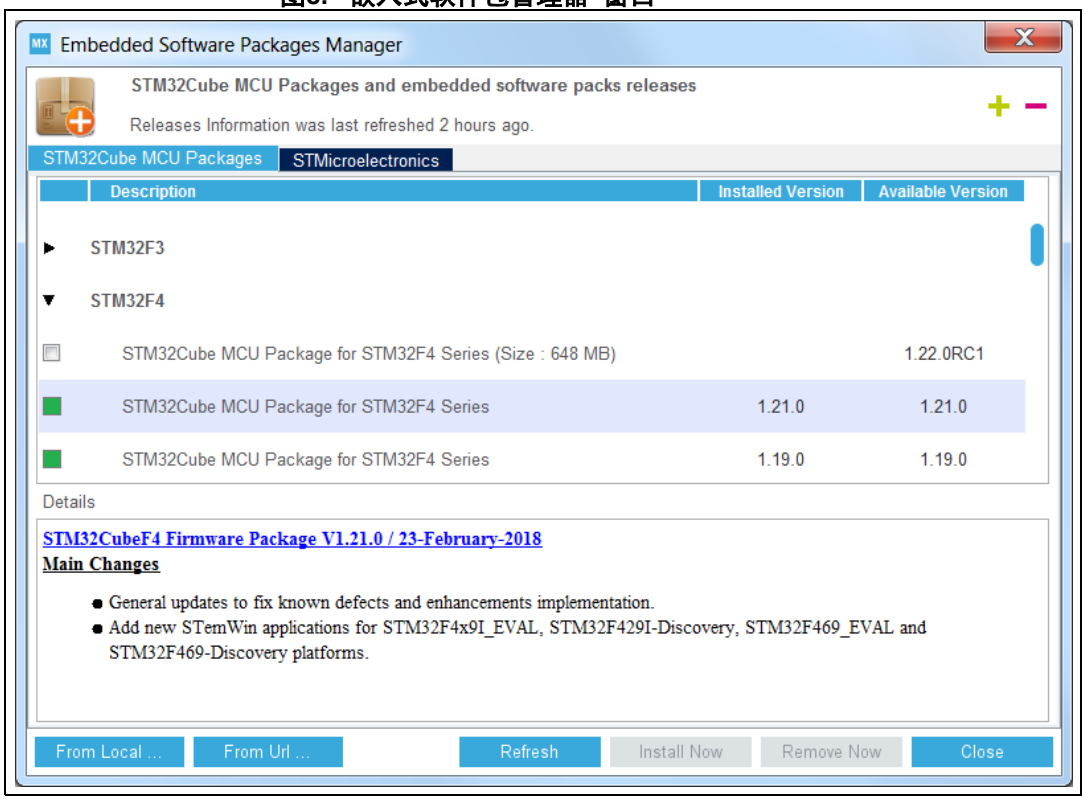
   如果使用STM32CubeMX进行安装，则显示所有可供下载的软件包及其版本，包括用户PC当前安装的版本（若有），以及*www.st.com*提供的最新版本。

   如果此时无法访问互联网，选择“从本地...”，然后浏览并选择先前下载的所需STM32Cube MCU软件包的zip文件。对文件执行完整性检查，确保STM32CubeMX完全支持该文件。

   当安装的版本与*www.st.com*提供的最新版本匹配时，程序包将标记为绿色。
2. 点击复选框，选择软件包，然后点击“立即安装”，开始下载。

相关示例，请参见*图 8*。

图8. “嵌入式软件包管理器”窗口



3.4.3 安装STM32 MCU软件包补丁

使用*第 3.4.2 节*所述流程，下载STM32 MCU软件包补丁。

可以通过版本号轻松识别库补丁，例如STM32Cube_FW_F7_1.4.1，其中第三个数字为非空（例如，1.4.1版本为“1”）。

该补丁不是一个完整的库软件包，只是一组需要更新的库文件。补丁文件位于原始包的顶部（例如，STM32Cube_FW_F7_1.4.1是对STM32Cube_FW_F7_1.4.0软件包的补充）。

在4.17版本之前，STM32CubeMX复制原始基线目录中的补丁（例如，STM32Cube_FW_F7_V1.4.1补丁文件被复制到STM32Cube_FW_F7_V1.4.0目录中）。

自STM32CubeMX4.17开始，下载补丁会创建专用目录。例如，下载STM32Cube_FW_F7_V1.4.1补丁会创建STM32Cube_FW_F7_V1.4.1目录，其中包含原始STM32Cube_FW_F7_V1.4.0基线以及STM32Cube_FW_F7_V1.4.1包中所含补丁文件。

然后，用户可以选择在一部分项目中继续使用原始软件包（不含补丁），在其他项目中升级到补丁版本。

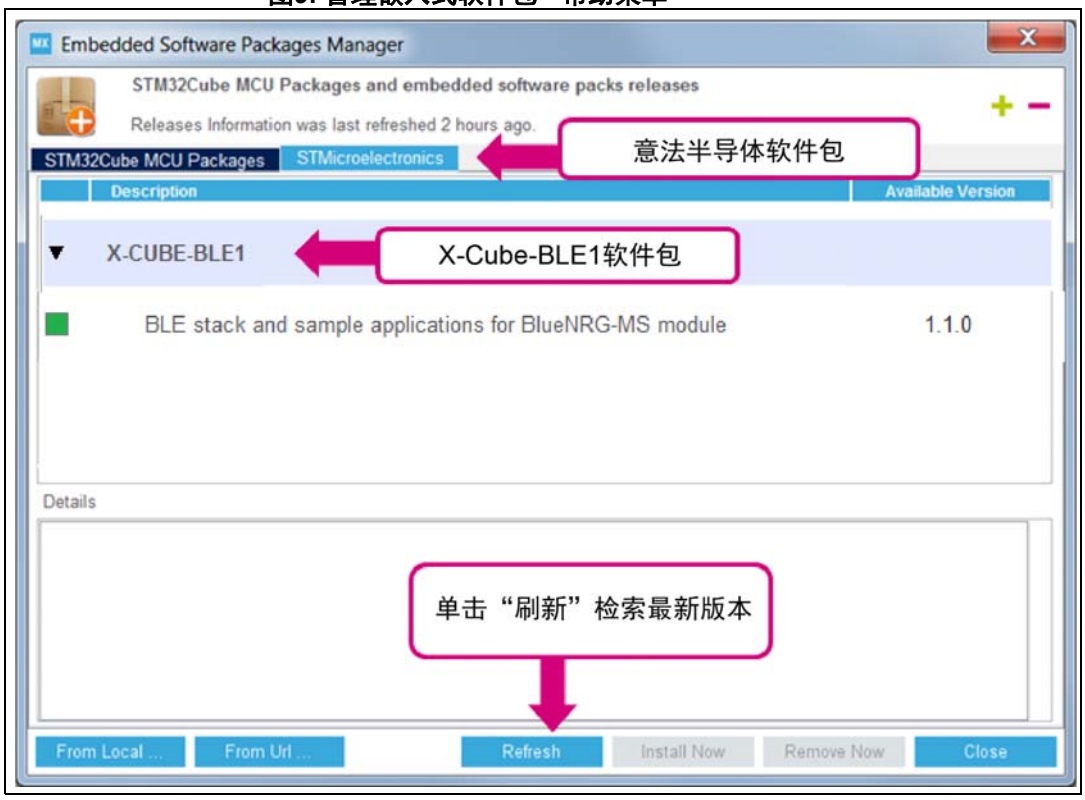
3.4.4 安装嵌入式软件包

自 4.24 版本开始， STM32CubeMX 可以选择 Arm® Keil™ CMSIS 包格式 （.pack）的第三方嵌入式软件包，其内容请见包描述 （.pdsc）文件。可从 http://www.keil.com 获取参考文档。

1. 选择**“帮助”**>**“管理嵌入式软件包”**，打开**“新库管理器”**窗口（参见*图 9*），或使用主页上的“安装/删除”键。

   使用展开/折叠按钮  ，展开或折叠软件包列表。

图9. 管理嵌入式软件包 - 帮助菜单



2. 点击**“从本地...”**按钮，浏览计算机文件系统，并选择嵌入式软件包。STM32Cube MCU软件包以zip档案形式提供，嵌入式软件包以.pack档案形式提供。

   在下列情况下需要执行此操作：

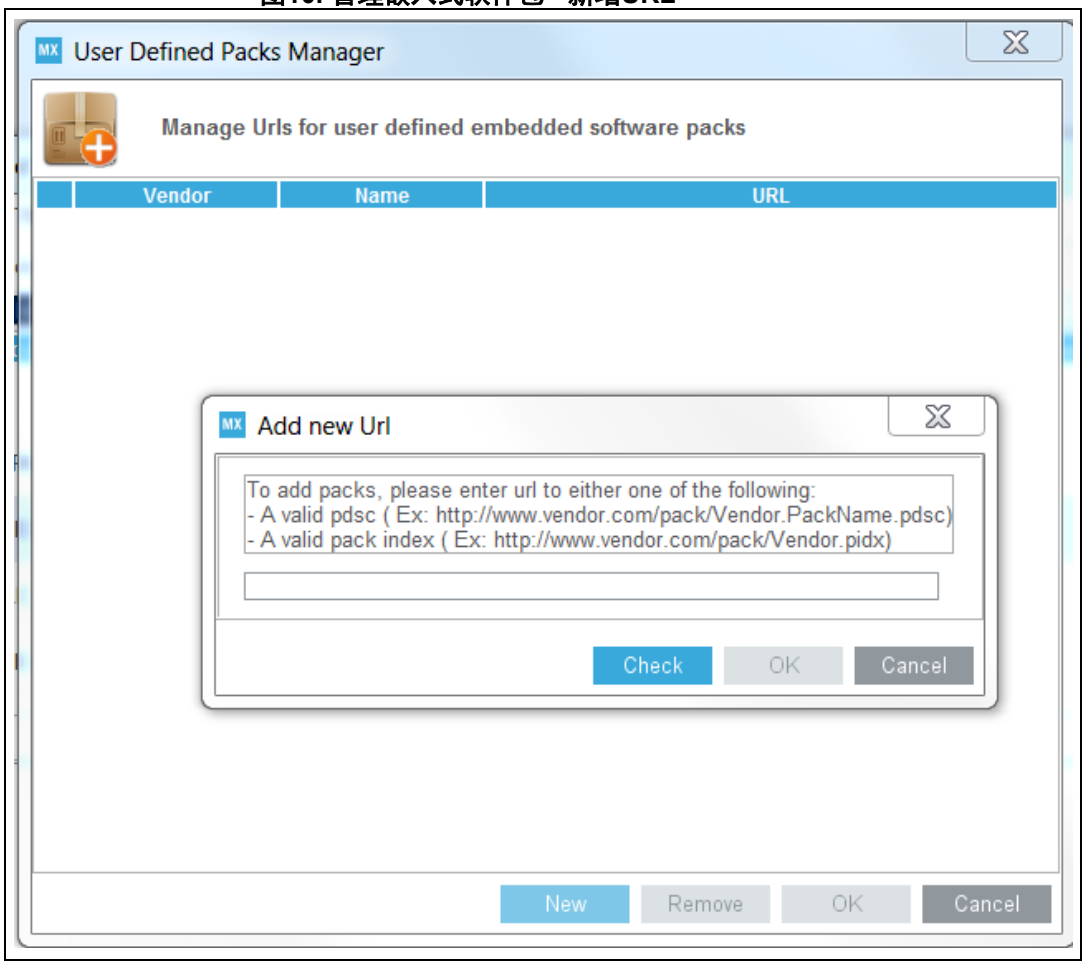
   – 无法访问互联网，但可在本地计算机上使用嵌入式软件包。
   – 嵌入式软件包不公开，因此无法在互联网上使用。对于此类软件包，STM32CubeMX无法检测并建议更新。

3. 点击**“从URL...”**按钮，为软件包.pdsc文件或外部包索引（.pidx）指定互联网下载位置。

   按照以下步骤继续：

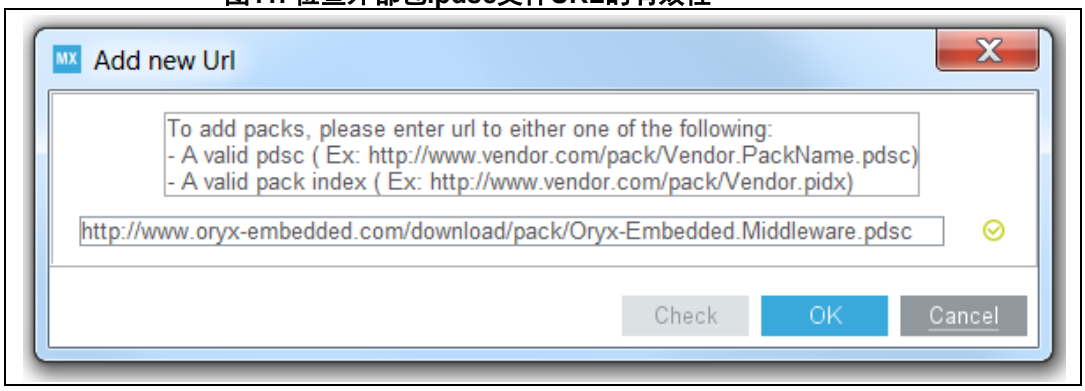
   a) 选**“从URL...”**，点击“**新建**”（参见*图 10*）。
   b) 指定.pdsc文件URL。例如，Oryx嵌入式中间件包的URL是https://www.oryx-embedded.com/download/pack/Oryx-Embedded.Middleware.pdsc（参见*图 11*）。

图10. 管理嵌入式软件包 - 新增URL



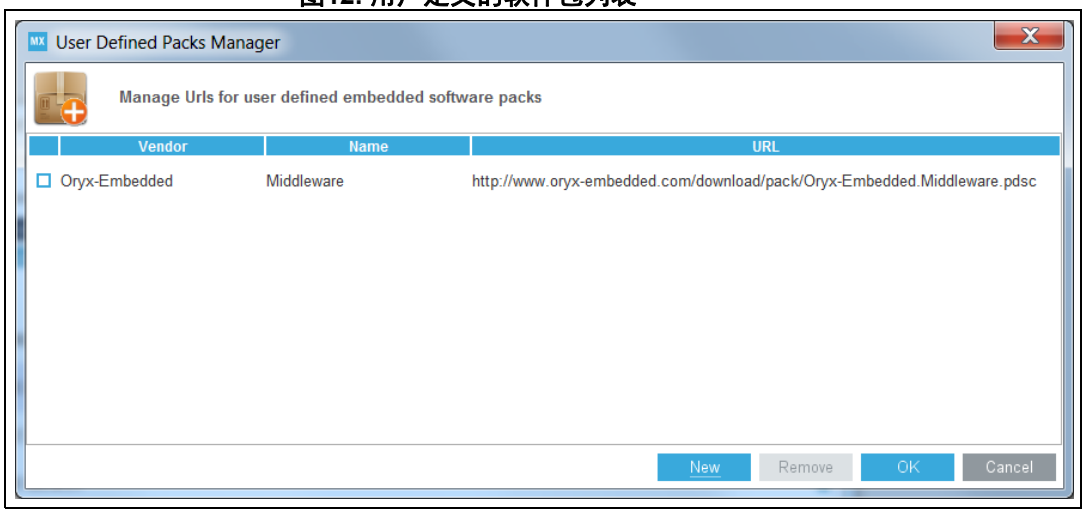
c) 点击**“检查”**按钮，验证所提供的URL是否有效（参见*图 11*）。

图11. 检查外部包.pdsc文件URL的有效性



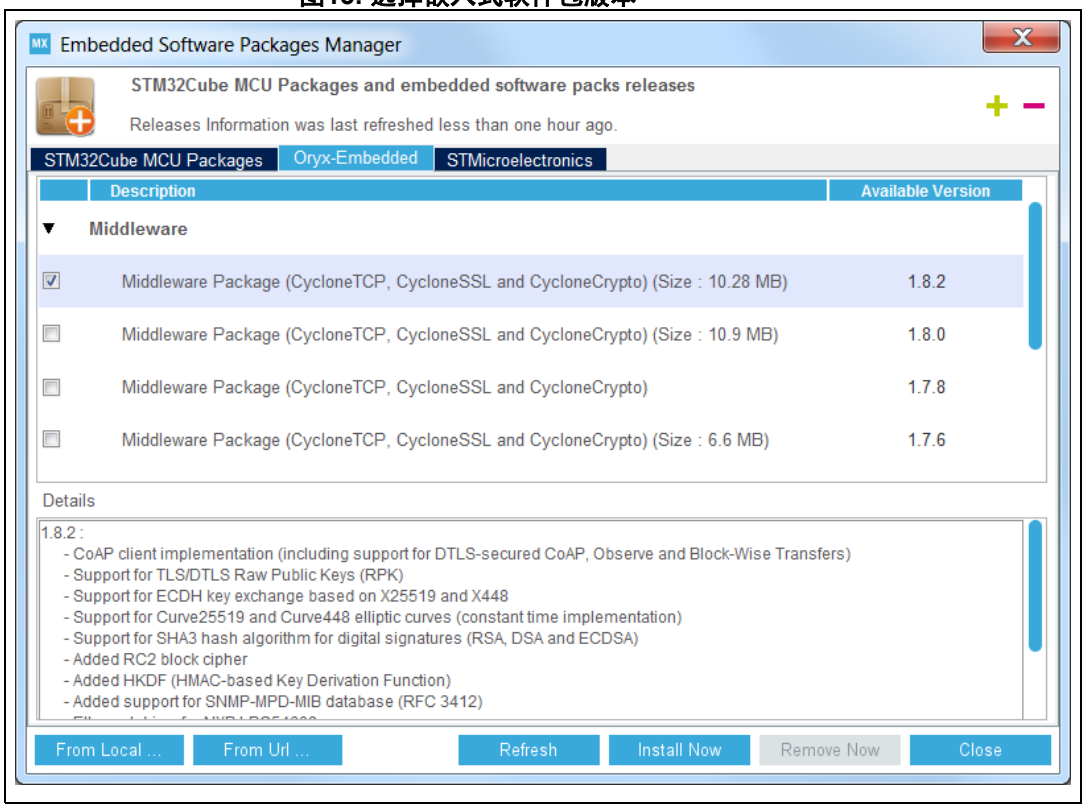
d) 点击**“确定”**。现在可在用户定义的包列表中找到包pdsc信息（参见*图 12*）。要从列表中删除URL，选中URL复选框，然后点击**“删除”**。

图12. 用户定义的软件包列表



e) 点击“**确定**”，关闭窗口并开始检索psdc信息。成功完成后，可用的包版本将显示在可安装的库列表中。使用相应的复选框，选择给定的版本。

图13. 选择嵌入式软件包版本



f) 点击**“立即安装”**，开始下载软件包。将打开进度条，指示安装进度。如果软件包随附许可协议，将会弹出一个窗口，要求用户接受（请参阅*图 14*）。安装成功后，复选框变为绿色（参见*图 15*）。

然后，用户可以将此包中的软件组件添加到其项目中。

图14. 接受许可协议

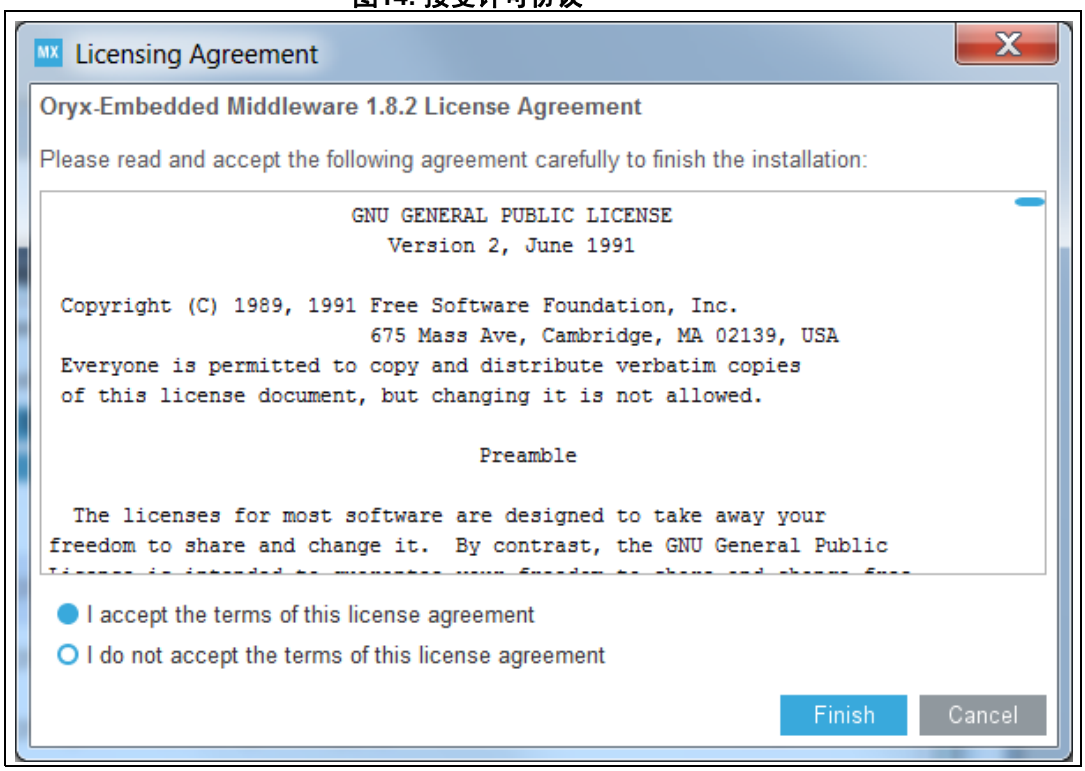
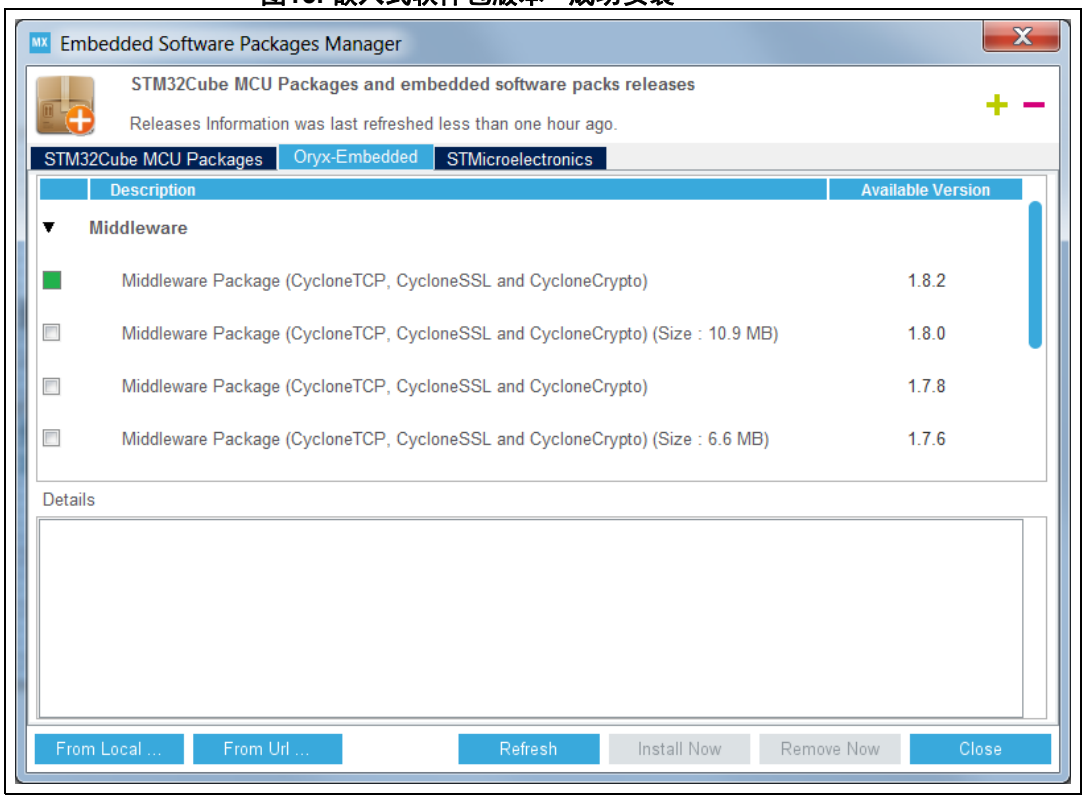


图15. 嵌入式软件包版本 - 成功安装



3.4.5 删除已安装的嵌入式软件包

请按以下步骤（参见图16至18）从旧库版本清理存储库，从而节省磁盘空间：

1. 选择**“帮助”>“管理嵌入式软件包”**，打开**“嵌入式软件包管理器”**，或使用主页上的“安装/删除”键。
2. 点击绿色复选框，选择stm32cube存储库中可用的软件包。
3. 点击**“立即删除”**按钮并确认。然后跳出进度窗口，显示删除状态。

图16. 删除库

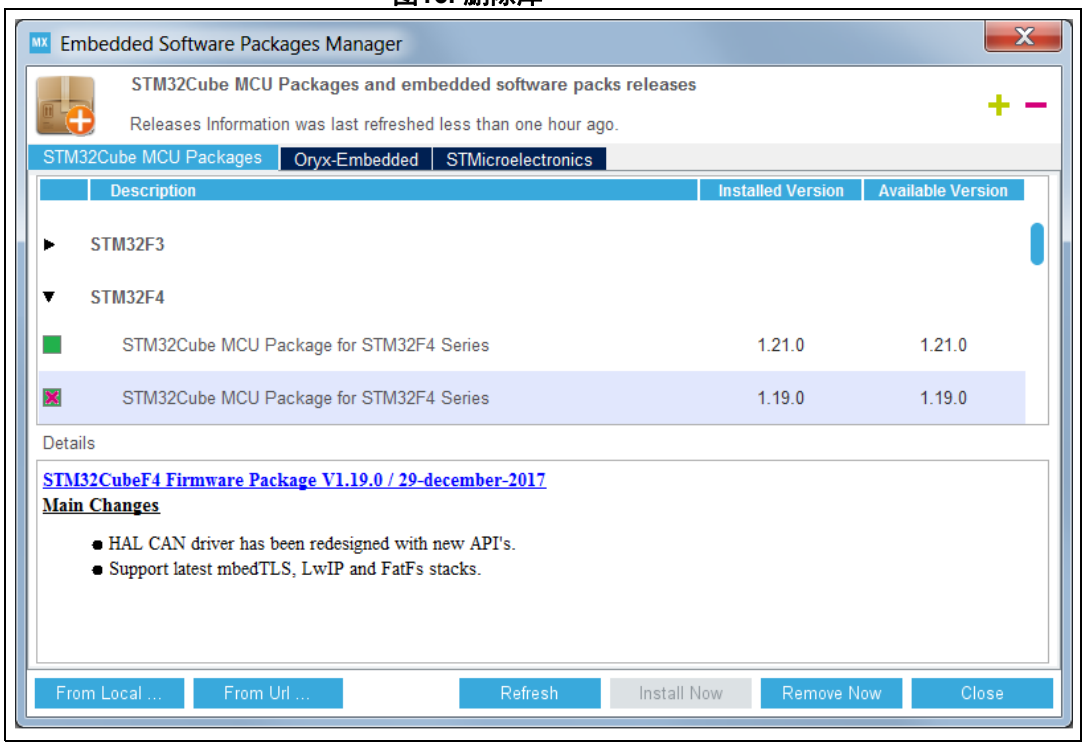


图17. 删除库确认消息

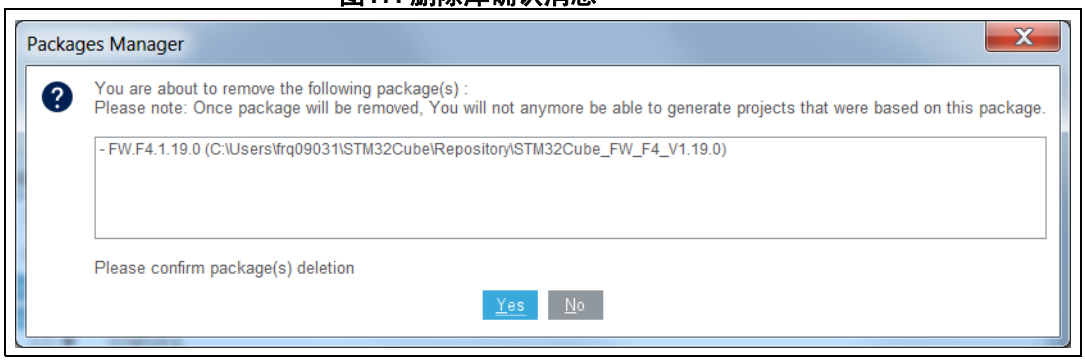
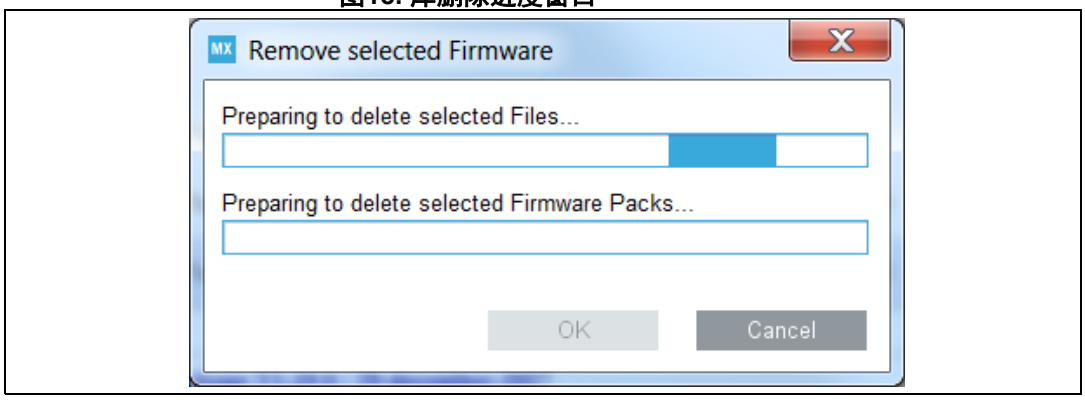


图18. 库删除进度窗口



3.4.6 检查更新

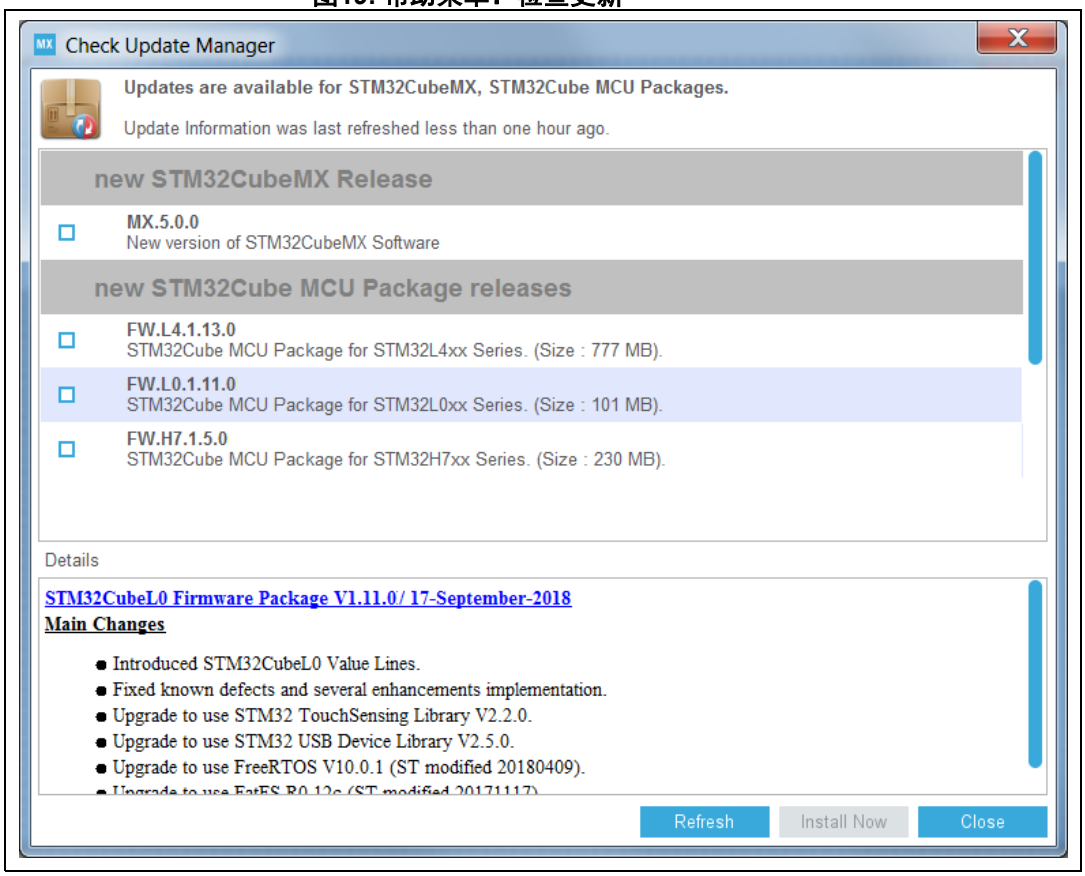
STM32CubeMX可以检查更新是否可用于STM32CubeMX当前安装的版本或安装在存储库文件夹中的嵌入式软件包（*图 19*）。

当更新程序配置为自动检查时，它会定期验证更新是否可用。

如果在更新程序设置窗口中禁用了自动检查，则用户可以手动检查更新是否可用：

1. 点击图标，打开**“更新管理器”**窗口或选择**“帮助”>“检查更新”**。列出了用户当前安装程序的所有可用更新。
2. 点击复选框，选择软件包，然后点击“立即安装”，下载更新。

图19. 帮助菜单：检查更新



4 STM32CubeMX用户界面

STM32CubeMX用户界面具有三个主视图，用户可以使用方便的breadcrumbs（breadcrumb）进行浏览：

1. **主**页
2. **新项目**窗口
3. 项目页面

具有面板、按键和菜单，用户只需单击即可进行操作并进行配置选择。

用户界面将在以下各节中详细介绍。

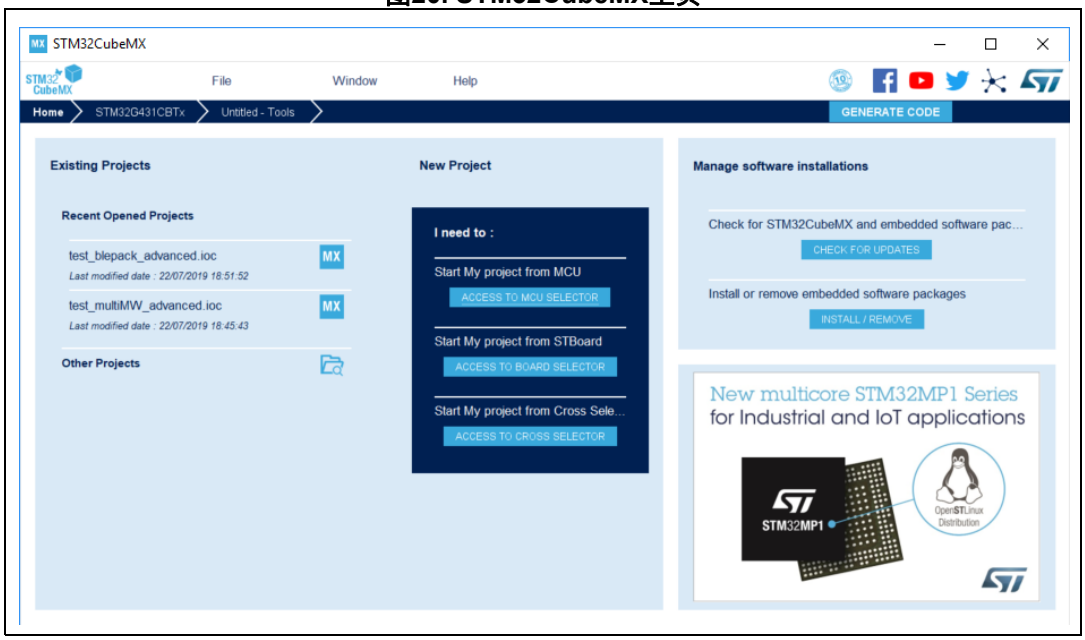
对于C代码生成，虽然用户可以在不同的配置视图之间来回切换，但建议遵循以下顺序：

1. 在**“项目管理器”**视图中，配置项目设置。
2. 在**“引脚布局和配置”**视图中的**模式**面板，通过启用外部时钟、主输出时钟、音频输入时钟（若与您的应用相关）来配置RCC外设。这会在**“时钟配置”**视图上自动显示更多选项（参见*图 95*）。然后，选择与应用相关的特性（外设、中间件）及其工作模式。
3. 如有必要，在时钟配置视图中调整时钟树配置。
4. 在**“引脚布局和配置”**视图的“配置”面板中，配置初始化外设和中间件操作模式所需的参数。
5. 点击 GENERATE CODE 生成初始化C代码。

4.1 主页

主页是启动STM32CubeMX程序时打开的第一个窗口（参见*图 20*）。关闭该页面即会关闭应用程序。它提供了一些顶层菜单的快捷键，还可访问社交网站。顶层菜单和社交网络链接仍可从后续项目页面访问，将在以下各节进行详细说明。

图20. STM32CubeMX主页



4.1.1 文件菜单

有关**文件菜单**和快捷键的说明，请参见*表 2*。

表2. 主页快捷键

名称 ***键盘快捷键***	**说明**	**主页快捷键**
新项目... *Ctrl-N*	打开一个新项目窗口，显示所有支持的MCU和一组可供选择的意法半导体的开发板(1)。	要从板开始创建新项目，请单击  要从MCU开始创建新项目，请单击 
加载项目... *Ctrl-L*	选择STM32CubeMX配置.ioc文件，加载已有的STM32CubeMX项目配置（参见*注意：*）。	在“其他项目”下，单击浏览图标 
导入项目... *Ctrl-I*	打开一个新窗口，选择要导入的配置文件以及导入设置。仅当使用空MCU配置时才可以导入。否则，菜单被禁用(2)。	无
保存项目 *Ctrl-S*	将当前项目配置（引脚排列、时钟树、外设、中间件、功耗计算器）另存为新项目。 根据用户自定义的项目设置，该操作将创建包含.ioc文件的项目文件夹。	无
项目另存为... *Ctrl-A*	保存当前项目。	无
关闭项目 *Ctrl-C*	关闭当前项目并切换回欢迎页面。	无
近期项目 *无*	显示最近保存的五个项目的列表。	在**“最近项目”**下，单击项目名称旁边的  图标。
生成报告 *Ctrl-R*	将项目的当前配置保存为两个文档（pdf格式和文本格式）。	无
退出 *Ctrl-X*	建议保存项目（如有必要），然后关闭应用程序。	点击，关闭窗口和应用程序。 

1. 在**“新建项目”**时：为避免在此阶段弹出任何错误消息，请确保互联网连接可用（“帮助”>“更新程序设置”菜单下的“连接参数”选项卡）或“数据自动刷新”设置为“不
在应用程序启动时自动刷新（“帮助”>“更新程序设置”菜单下的“更新程序设置”选项卡）。
2. 在**导出**上，状态窗口显示检查导入冲突时检测到的警告或错误。用户可以决定是否取消导入。

注意： 在**项目加载**时：STM32CubeMX会检测该项目是否使用该工具的旧版本创建，如果是，则会建议用户迁移，以使用最新的STM32CubeMX数据库和STM32Cube固件版本，或者继续。
在STM32CubeMX 4.17之前，点击继续仍然会升级到与项目使用的SMT32Cube固件版本“兼容”的最新数据库。
自STM32CubeMX 4.17开始，点击继续将保持用于创建项目的数据库不变。如果计算机上没有所需的数据库版本，则会自动下载该版本。
升级到STM32CubeMX新版本时，请确保在加载新项目之前始终备份项目（特别是当项目包含用户代码时）。

4.1.2 “窗口”菜单和“输出”选项卡

“窗口”菜单允许用户访问**输出**功能。

表3. 窗口菜单

名称	说明
输出	从**“窗口”**菜单中选择/取消选择**“输出”**会在STM32CubeMX项目页面底部隐藏/显示以下“输出”选项卡（请参阅*图 21*） – 最后选定某一MCU时(1)，**MCU选择**选项卡列出某一特定系列中匹配用户标准（系列、外设、封装..）的MCU。 – **“输出”**选项卡显示了关于所执行的操作、出现的错误以及在用户操作时发现的警告等信息（参见*图 22*）的简单列表。
字体大小	可以更改STM32CubeMX字体大小设置。必须重新启动STM32CubeMX，使更改生效。

1. 从列表中选择不同的MCU将重置当前项目配置并切换到新的MCU。然后系统会提示用户在继续下一步之前确认此操作。

图21. 窗口菜单

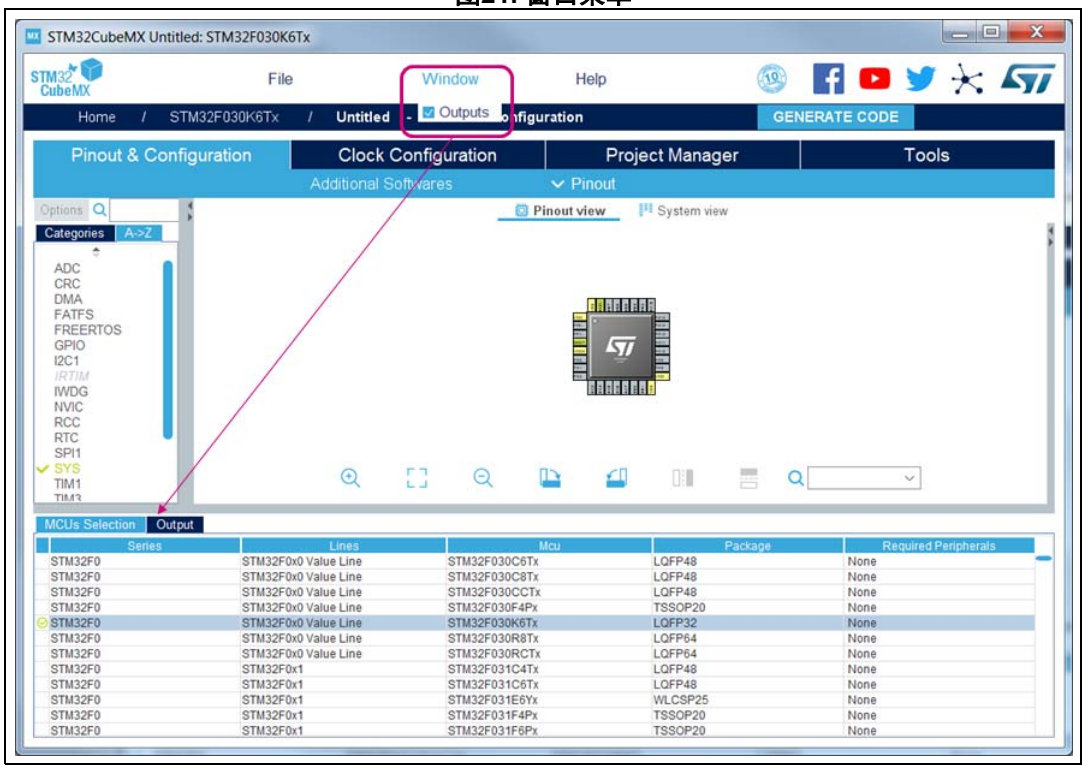
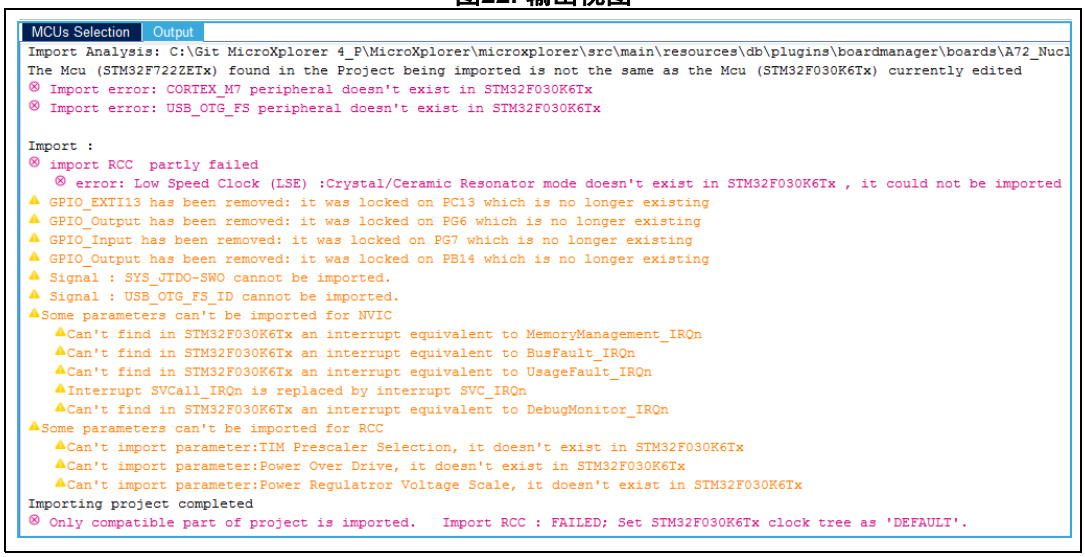


图22. 输出视图



4.1.3 Help菜单

有关**帮助**菜单和快捷键的描述，请参见*表 4*。

表4. “帮助”菜单快捷键

名称 *键盘快捷键*	说明	主页快捷键
帮助 ***F1***	打开STM32CubeMX用户手册。	无
关于 ***Alt-A***	显示版本信息。	无
文档和资源 ***Alt-D***	显示可用于当前项目中所用MCU的官方文档。	无
刷新数据 *Alt-R*	打开一个对话窗口，该窗口建议使用STM32MCU最新信息刷新STM32CubeMX数据库（描述和官方文档列表），并允许用户一次性下载所有官方文档。	无
检查更新 *Alt-C*	显示可供下载的软件和固件版本更新。	点击 CHECK FOR UPDATES
管理嵌入式软件包 ***Alt-U***	显示可供安装的所有嵌入式软件包。 绿色复选框表示软件包已经安装在用户存储库文件夹（在“帮助>更新程序设置”菜单中指定存储库文件夹位置）。	点击 INSTALL/REMOVE
更新程序设置... ***Alt-S***	打开更新程序设置窗口，配置手动或自动更新、互联网连接的代理设置、用于存储已下载的软件和固件版本的存储库文件夹。	无
用户首选项	打开用户首选项窗口以启用或禁用收集功能使用统计信息。	无

4.1.4 社交链接

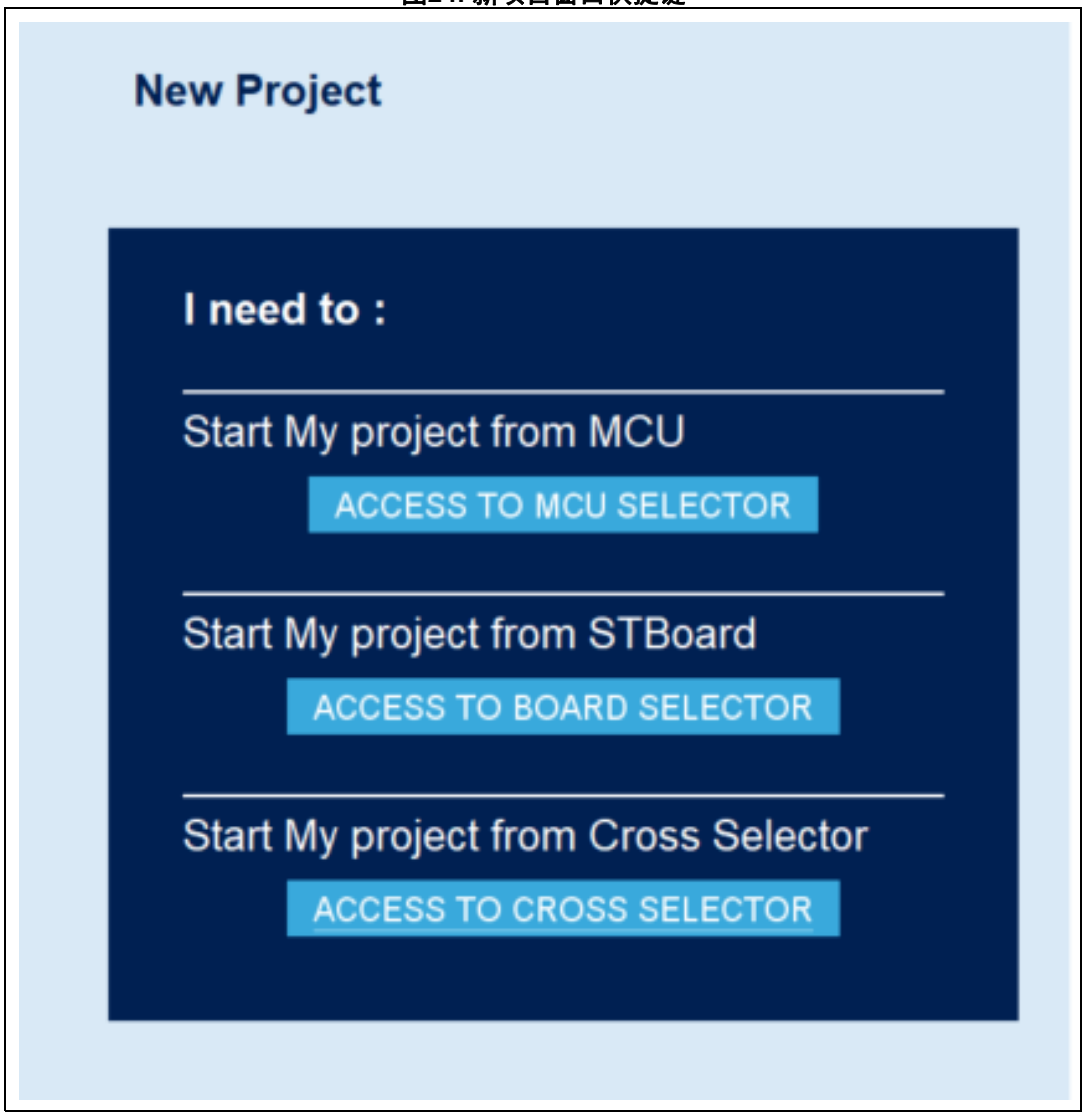
可通过STM32CubeMX工具栏访问建于Facebook™、Twitter™、STM32 YouTube™频道和ST社区等流行社交平台上的开发者社区（参见*图 23*）。

图23. 社交平台链接

4.2 新项目窗口

通过“文件”菜单或直接通过主页上的快捷键访问“新项目”窗口（请参阅*图 24*）。

图24. 新项目窗口快捷键



主要用于从STM32产品中选择最适合用户应用需求的微控制器或开发板料号。

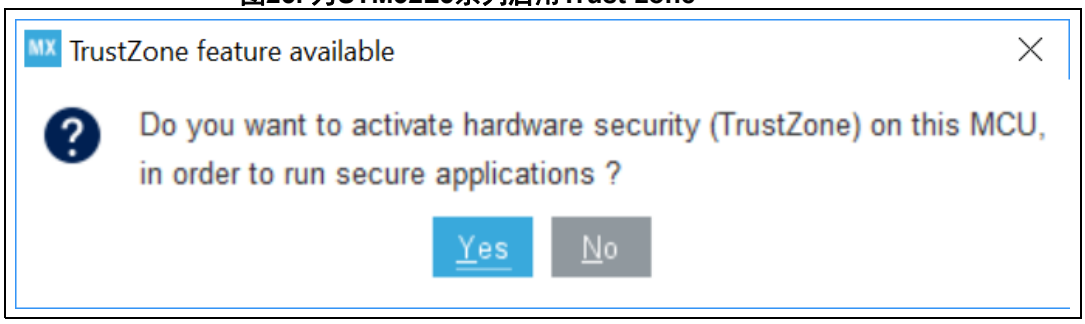
此窗口显示三个选项卡供您选择：

- **“MCU选择器”**选项卡（提供目标处理器列表）
- **“板选择器”**选项卡（显示意法半导体开发板列表）
- **“交叉选择器”**选项卡（使用户可以针对给定的MCU/MPU料号和一组条件找到STM32产品组合中的最佳替代品）

对于STM32L5系列，Arm Cortex-M33处理器的安全特性及其面向Armv8-M架构的TrustZone与意法半导体安全实现方案相结合。选择STM32L5 MCU或板需要选择是否启用TrustZone（硬件安全性）（请参阅*图 25*）。相应地调整项目：

- 如果未启用Trustzone，解决方案与其他STM32Lx系列相同
- 如果已启用TrustZone，项目配置和生成的项目将显示与安全功能相关的特性（请参阅本手册的专用部分）。

图25. 为STM32L5系列启用Trust-zone

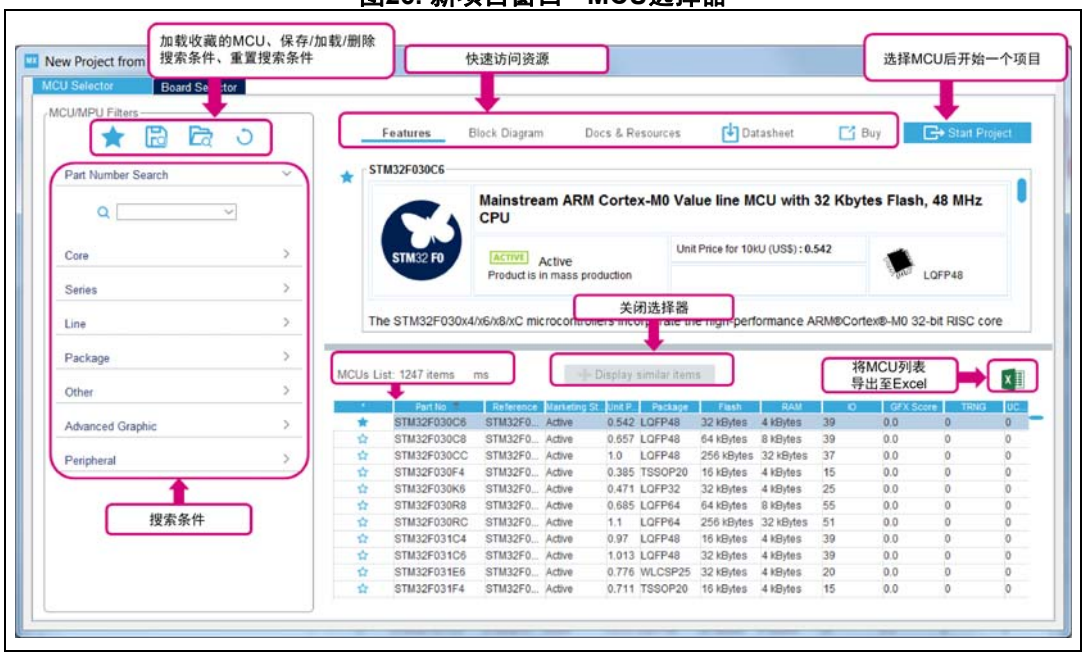


4.2.1 MCU选择器

MCU选择

“MCU选择器”允许基于一系列条件进行筛选：系列、产品线、封装、外设或其他特性，如价格、存储器容量或I/O数量（参见*图 26*）以及图形功能。

图26. 新项目窗口 - MCU选择器

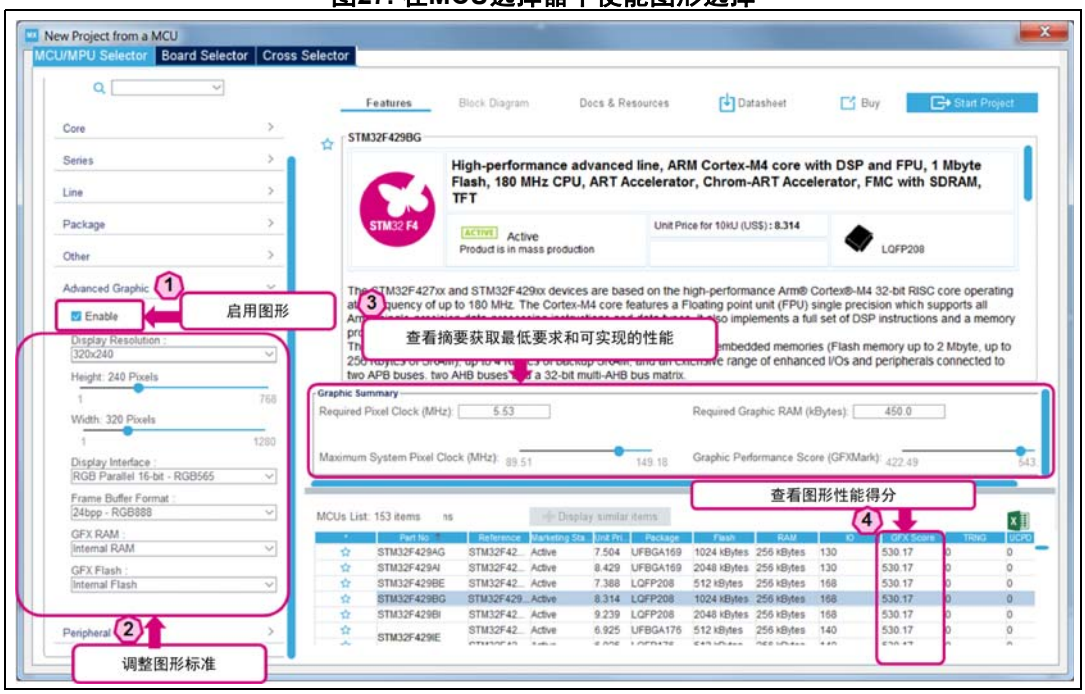


基于图形条件选择MCU

选中复选框，启用**“图形选择”**，使用以下条件来刷新“MCU选择器”视图（如*图 27*所示）：

1. 一组**图形特定的筛选条件**
2. 符合这些条件及其图形性能得分的**MCU列表**。**图形性能得分**是指，在所选图形系统配置中可使用MCU实现的图形性能，这是一种指示性估计结果：得分越高，性能越佳。图形性能得分显示在GFX 列中。此外，从该列表中选择MCU，即可在项目中使用图形协议栈。
3. **图形摘要面板**，显示满足所选图形条件的像素时钟和图形RAM大小的最低要求。
   它还显示了当前MCU列表可以实现的性能范围（最大系统时钟和图形性能得分）。
   工具提示中提供了参数说明（如要显示：将鼠标悬停在参数名称上）。

图27. 在MCU选择器中使能图形选择



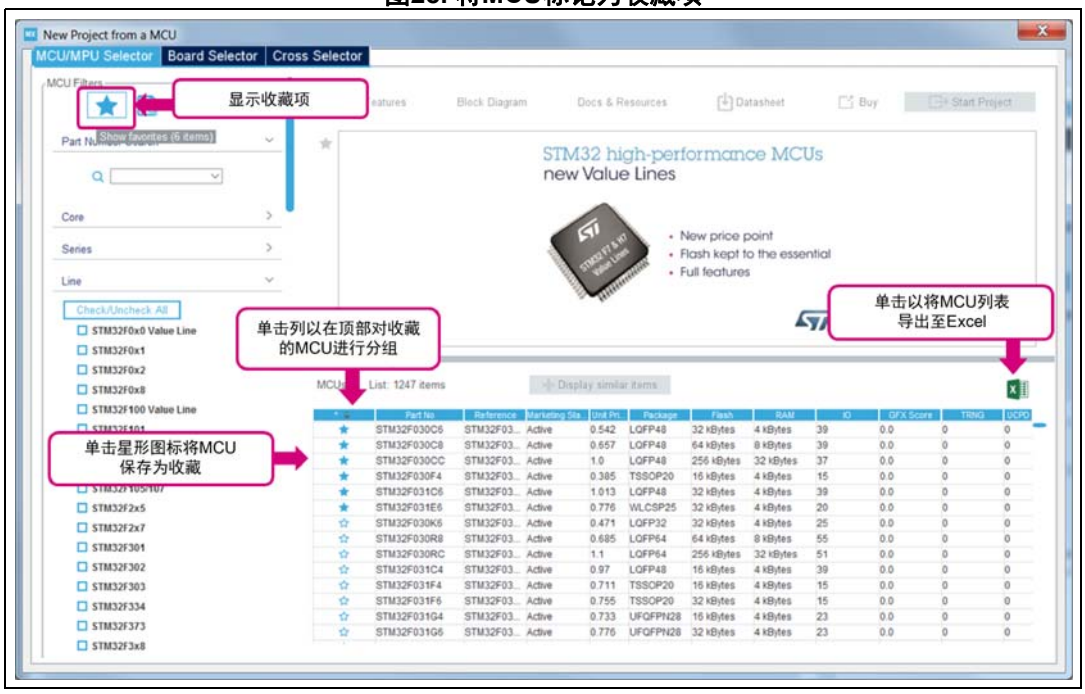
导出至Excel功能

点击图标  ，用户即可将MCU表信息保存到Excel文件。

显示收藏的MCU特性

点击MCU列表中的图标  ，将其标记为收藏项，请参见*图 28*。

图28. 将MCU标记为收藏项



MCU相近选择器特性

当找到的MCU数量低于50时，选择器会列出具有相近特性的MCU（参见*图 29*）。点击**“显示类似项目”**按钮，显示这些相近特性（参见*图 30*）：默认情况下，MCU首先按匹配率排序，然后按料号排序。对于相近的MCU（匹配率低于100%），MCU所在行显示灰色，而非匹配的单元格则突出显示为深灰色。

图29. 新项目窗口 - 具有相近特性的MCU列表

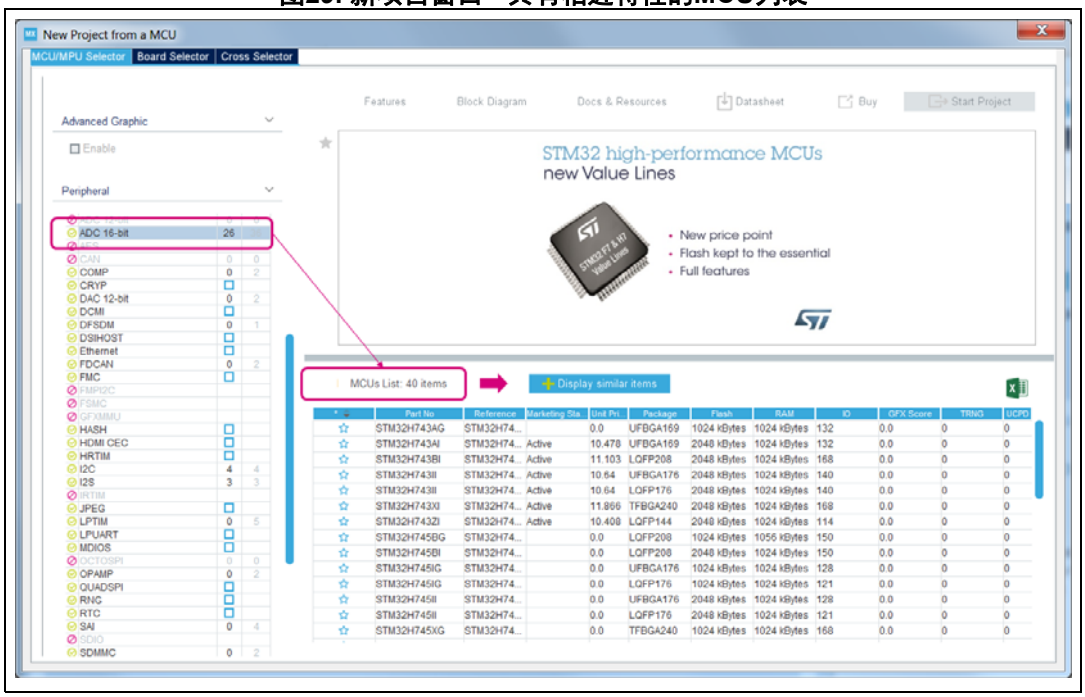
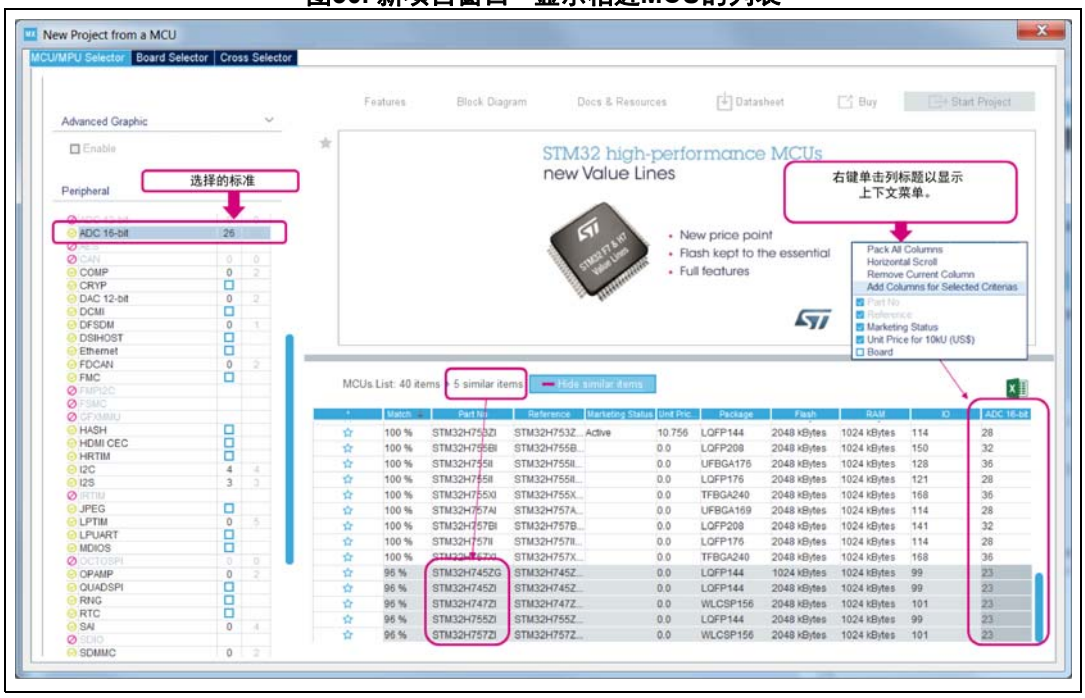


图30. 新项目窗口 - 显示相近MCU的列表



注： *为每个用户选择的条件计算匹配百分比，例如：*

- 根据CAN标准，当请求CAN外设的四个实例时，仅拥有三个实例的MCU会达到75 %的匹配率。

- 如果选择最大价格标准，则给定MCU的匹配率等于最大请求价格除以实际MCU价格。在最

低价格标准的情况下，匹配率等于MCU价格除以最低要求价格。
最后，将所有条件比率取平均值，得出"匹配"列百分比值。

4.2.2 板选择

"板选择器"允许对STM32板类型、系列和外设进行筛选（参见*图 31*）。仅建议使用默认板配置。不支持通过重新配置跳线或使用焊桥获得的替代板配置。

选择板时，**"引脚布局"**视图将使用相关MCU料号以及LCD、按钮、通信接口、LED和其他功能的引脚分配进行初始化。或者，用户可以选择使用默认外设模式对其进行初始化。

当选择板配置时，信号变为"引脚已固定"，即，不能通过STM32CubeMX约束条件求解器自动移动（外设树上的用户操作，例如选择外设模式，不会移动信号）。这可确保用户配置与开发板保持兼容。

图31. 新项目窗口 - 板选择器

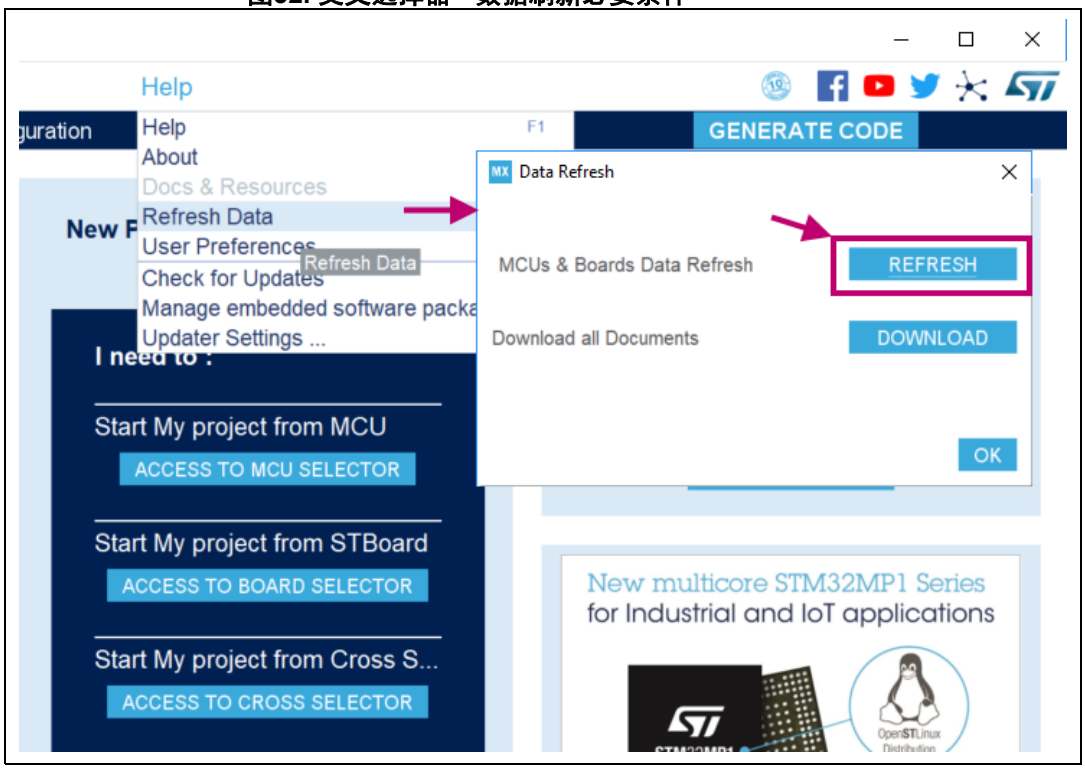
4.2.3 交叉选择器

料号选择

交叉选择器使用户可以找到STM32产品组合中可以替代当前使用的MCU或MPU（来自意法半导体或其他芯片供应商）的最佳产品。

要访问此功能，STM32CubeMX数据必须是最新的。使用"帮助"菜单中的"刷新数据"可以确保这一点（请参阅*图 32*）。

图32. 交叉选择器 - 数据刷新必要条件

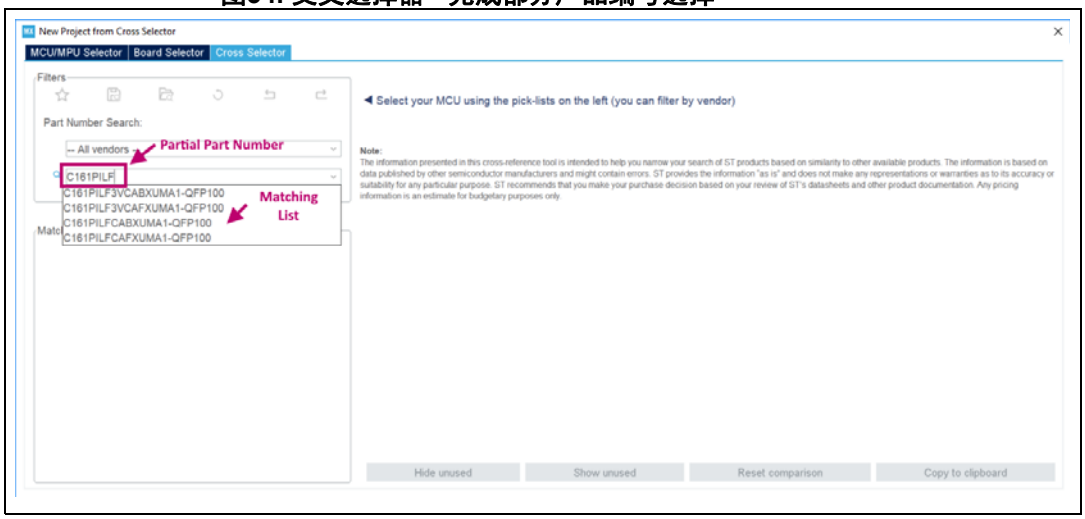


单击主页“从交叉选择器启动我的项目”部分下的“访问交叉选择器”，可在“交叉选择器”选项卡上打开“新建项目”窗口。

用户可以使用两个下拉菜单选择供应商以及要比较的产品的料号（请参阅*图 33*）。也可以部分输入产品编号：STM32CubeMX会列出匹配产品列表（请参阅*图 34*）。

图33. 交叉选择器 - 按供应商的料号选择

图34. 交叉选择器 - 完成部分产品编号选择



比较购物篮

选择料号后，就会在“匹配的ST候选产品”面板中显示出匹配的意法半导体料号候选列表及其匹配率。

在默认情况下，将选择三个最接近的匹配项，并将其与要比较的料号一起添加到比较购物篮中（请参阅*图 35*）。

图35. 交叉选择器 - 比较购物篮

这一选择可以随时在“匹配的意法半导体候选产品”面板中进行更改。

可以自定义比较：在认为不相关时可以取消选择要用于比较的功能，还可以调整其重要性。这些选择会影响计算出的匹配率。

对于待比较料号不支持的功能，或者当功能信息不可用时，将禁用比较。

按键可用于操作和保存比较购物篮视图的副本：

- 隐藏不用于比较的条件或显示所有条件。
- 返回STM32CubeMX默认比较设置
- 将当前比较购物篮视图复制并粘贴到文档或电子邮件中。

选择新项目的MCU/MPU

从比较购物蓝中单击STM32料号，可在MCU/MPU选择器选项卡中将其选中，然后单击 Start Project 可创建该料号的新项目（请参阅*图 36*）。

图36. 交叉选择器 - 新项目的料号选择

单击“交叉选择器”选项卡，用户可以返回购物篮并更改当前选择以用于另一个料号。

4.3 项目页面

选择了STM32料号或板后，或者已经加载了先前保存的项目后，将会打开项目页面，显示以下一组视图（有关其详细说明，请参阅专用部分）：

- **引脚布局和配置**
- **时钟配置**
- **项目管理器**
- **工具**

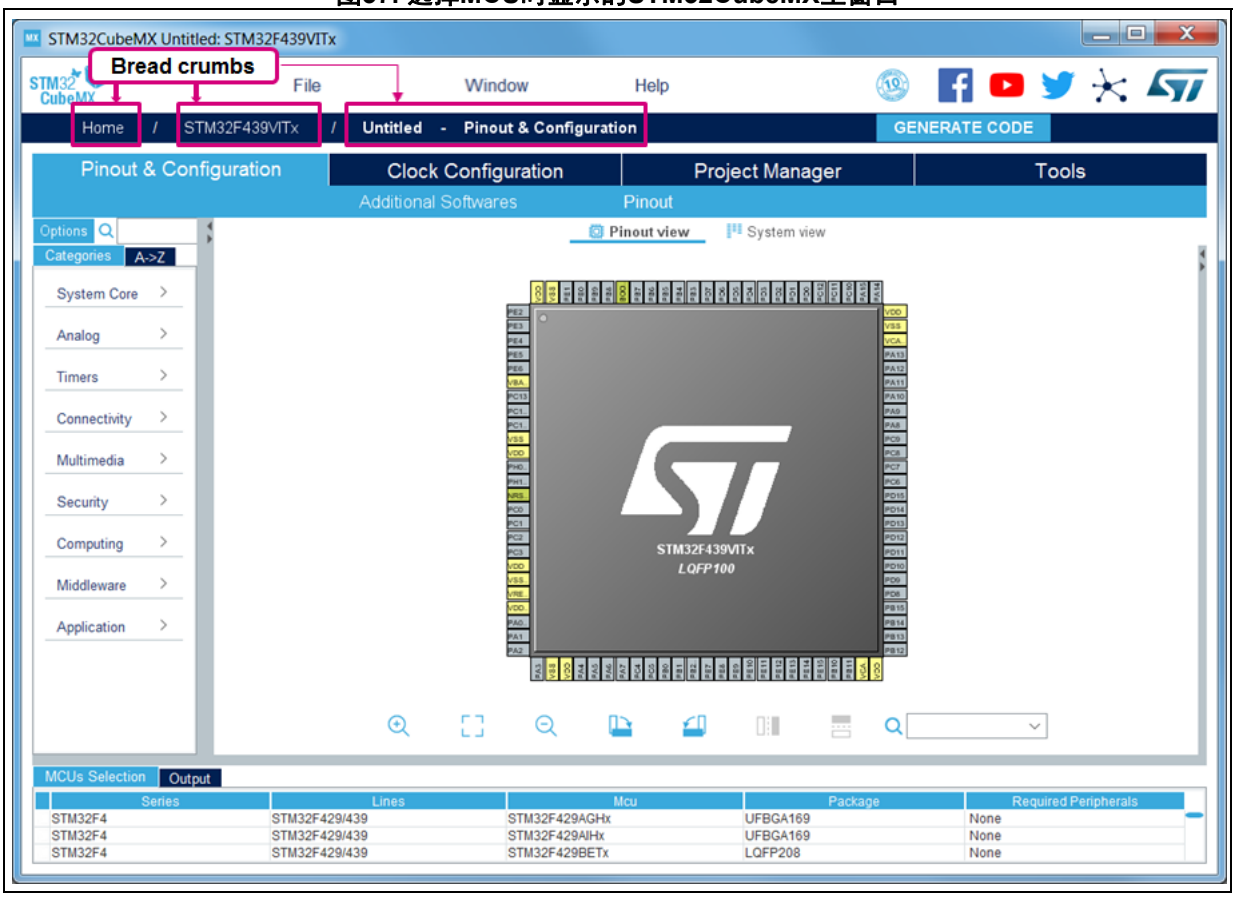
用户可以在其之间移动而不会影响当前所保存的配置。

用户始终可以单击 GENERATE CODE 按键，可以生成与当前项目配置相对应的代码。

此外，借助便捷的breadcrumbs（请参阅*图 37*），用户可以在STM32CubeMX用户界面中检测其当前位置，然后可以移至其他位置：

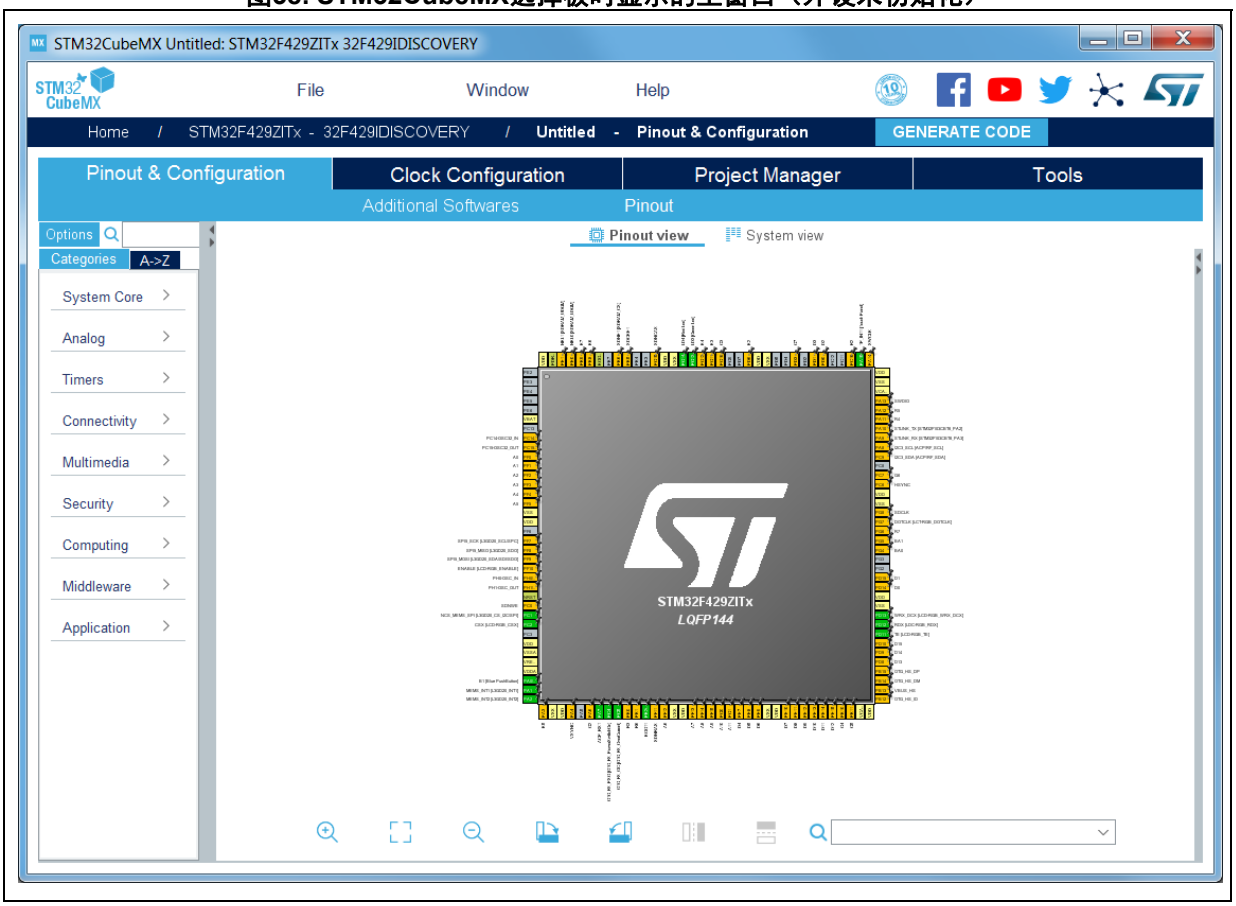
- 单击主页breadcrumbs进入主页
- 单击料号进入“新建项目”窗口
- 单击项目名称（如果该项目还没有名称，则为“无标题”）可以返回到项目页面。

图37. 选择MCU时显示的STM32CubeMX主窗口



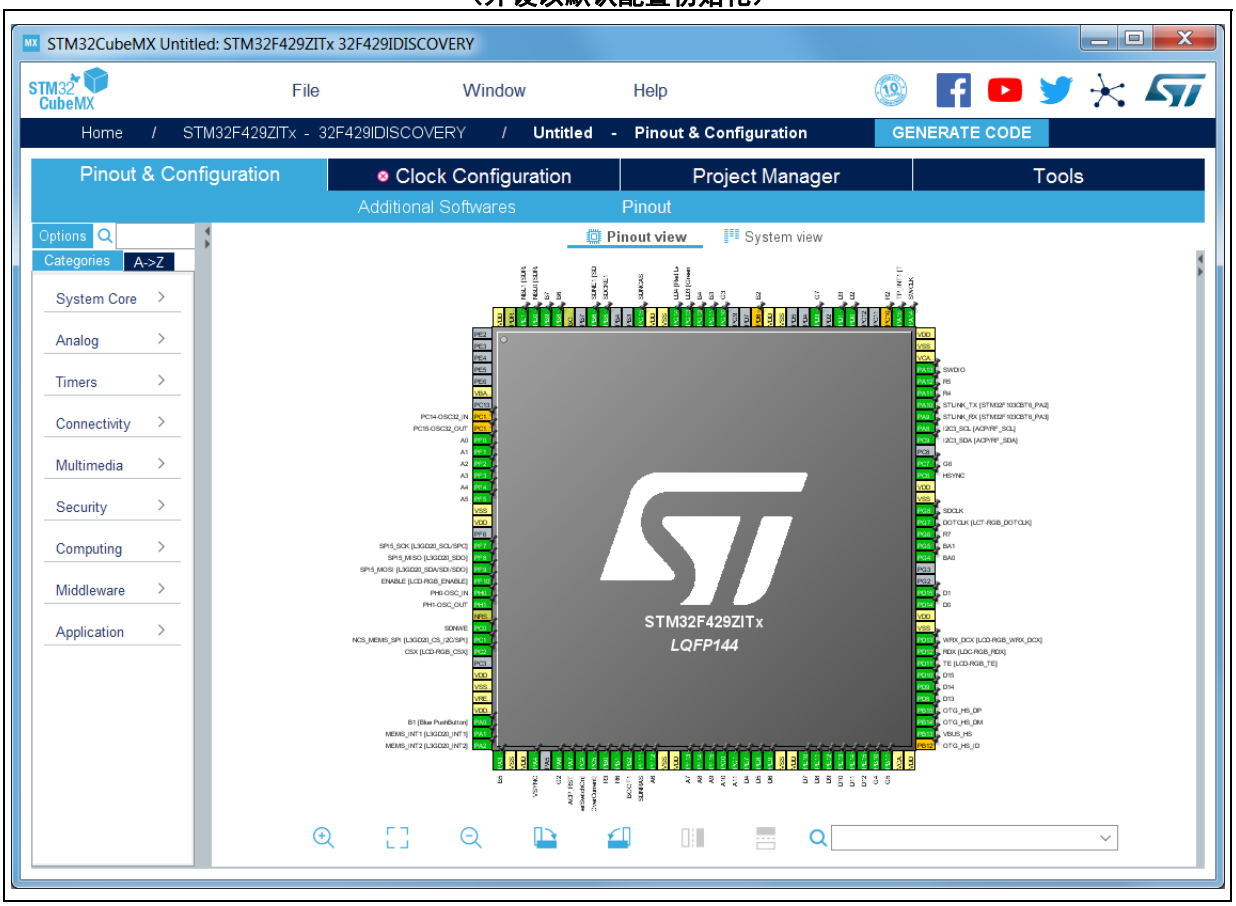
选择一个板，在对话窗口要求将所有外设初始化为其默认模式时回答**“否”**，从而自动设置该开发板的引脚排列。但是，只有设置为GPIO的引脚才会被标记为已配置，即以绿色突出显示，而未设置外设模式。接下来，用户可以从外设树中手动选择其应用所需的外设模式（参见*图 38*）。

图38. STM32CubeMX选择板时显示的主窗口（外设未初始化）



选择一个开发板，接受将所有外设初始化为其默认模式，会自动设置开发板上可用的外设引脚布局和默认模式。这意味着STM32CubeMX为开发板上可用的所有外设生成C初始化代码，而不仅仅是与用户应用程序相关的外设（参见*图 39*）。

图39. 选择板时显示的STM32CubeMX主窗口（外设以默认配置初始化）



4.4 “引脚布局和配置”视图

“引脚布局和配置”视图具有以下主面板、功能和菜单：

- 可以按字母顺序和类别查看的**“组件”列表**。在默认情况下，其包括所选MCU支持的外设和中间件列表。从该列表中选择一个组件将打开另外两个面板（**“模式”**和**“配置”**），用户可以借此设置其功能模式，并配置将包含在生成的代码中的初始化参数。
- **引脚布局**以图形方式表示所选封装的引脚布局（例如，BGA、QFP…），其中每个引脚都用其名称（例如，PC4）和当前备用功能分配（如有）来表示。
- **“系统”视图**提供了所有软件可配置组件的概览：GPIO，外设，中间件和其他软件组件。使用可单击的按键可以打开给定组件的配置选项（“模式”和“配置”面板）。按键图标颜色反映配置状态。

- **“附加软件”**功能可为当前项目选择默认情况下不可用的软件组件。选择附加软件组件将会相应地更新**“引脚布局和配置”**视图。
- **“引脚布局”**菜单，用户可借此执行与引脚布局相关的操作，例如清除引脚布局配置或将引脚布局配置导出为csv文件。

建议

- 可以随意调整不同面板的大小：将鼠标悬停在板边框上会显示一个双向箭头：右键单击，将会向某个方向拉动来扩展或缩小面板。
- 可以使用 打开 和关闭 箭头显示/隐藏“配置”、“模式”、“引脚”和“系统”视图。

4.4.1 元件清单

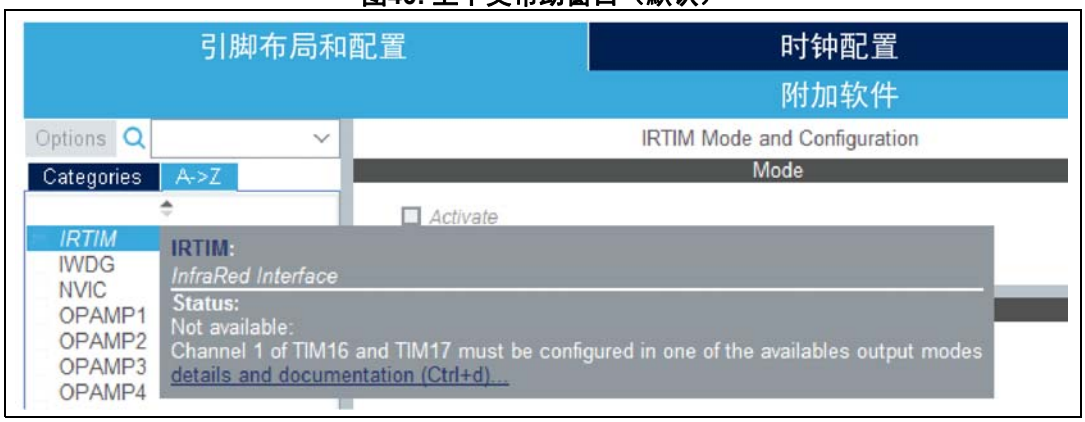
组件列表显示了该项目可用的所有组件。从组件列表中选择一个组件，将会打开“模式”和“配置”面板。

上下文帮助

将鼠标悬停在外设或中间件的短名称上，将显示**“上下文帮助”**窗口。

在默认情况下，该窗口显示扩展名和配置冲突源（如果有）（请参阅*图 40*）。

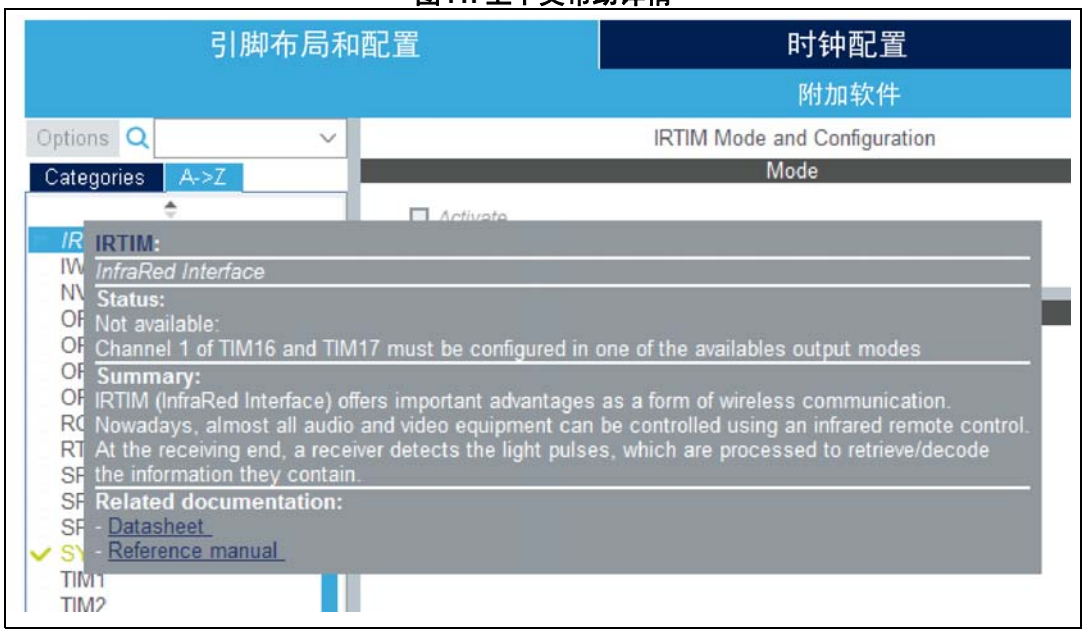
图40. 上下文帮助窗口（默认）



单击*详细信息和文档*链接（或CTRL + d）可提供额外信息，例如摘要和参考文档链接（请参阅*图 41*）。对于给定外设，单击*“数据手册”*或*“参考手册”*会在相关章节中打开存储在STM32CubeMX存储库文件夹中的相应文档。由于微控制器的数据手册和参考手册仅在用户请求时才会下载到STM32CubeMX存储库，因此需要功能性互联网连接：

- 要检查互联网连接，请从**“帮助”>“更新程序设置”**菜单中打开**“连接”**选项卡。
- 要请求下载当前所选微控制器的参考文档，请从**“帮助”>“刷新数据”**菜单窗口中单击**“刷新”**。

图41. 上下文帮助详情



图标和颜色方案

*表 5*显示组件列表视图中使用的图标和配色方案以及“模式”面板中的相应配色方案。

表5. 元件列表、模式图标和颜色方案

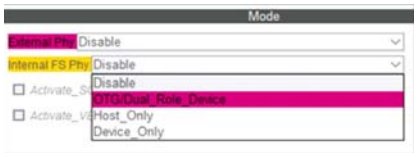
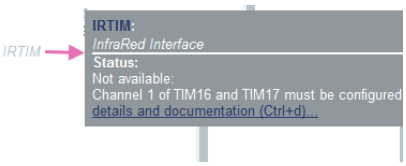
显示	组件状态	对应的“模式”视图/工具提示
纯黑色文本 示例： UART5	外设未配置（没有设置模式），所有模式都可用。	
灰色斜体文本 示例： *LWIP*	外设不可用，因为某些约束未求解。请参见工具提示。	
 示例： ETH	外设已配置（至少设置了一个模式），所有其他模式都可用。绿色复选标记表示所有参数均已正确配置，叉号表示未正确配置。	
 示例： USB_OTG_HS	外设未配置（没有设置模式），至少有一个模式不可用。	

表5. 元件列表、模式图标和颜色方案（续）

显示	组件状态	对应的“模式”视图/工具提示
示例： USB_OTG_HS	外设已配置（设置了一个模式），其他模式中至少有一个模式不可用。	
示例： I2C2	外设未配置（没有设置模式），没有模式可用。 将鼠标移到外设名称上，以显示描述冲突的工具提示。	
示例：IRTIM	外设由于限制条件而不可用。	

4.4.2 “组件模式”面板

从左侧面板上的组件列表中选择一个组件以打开**“模式”**面板。

模式面板帮助用户根据外设的选择和它们的操作模式配置MCU引脚。由于STM32MCU允许不同外设和多个功能（备用功能）使用相同的引脚，因此该工具搜索最适合用户选择的外设集的引脚排列配置。STM32CubeMX突出显示无法自动解决的冲突（参见*表 5*）。

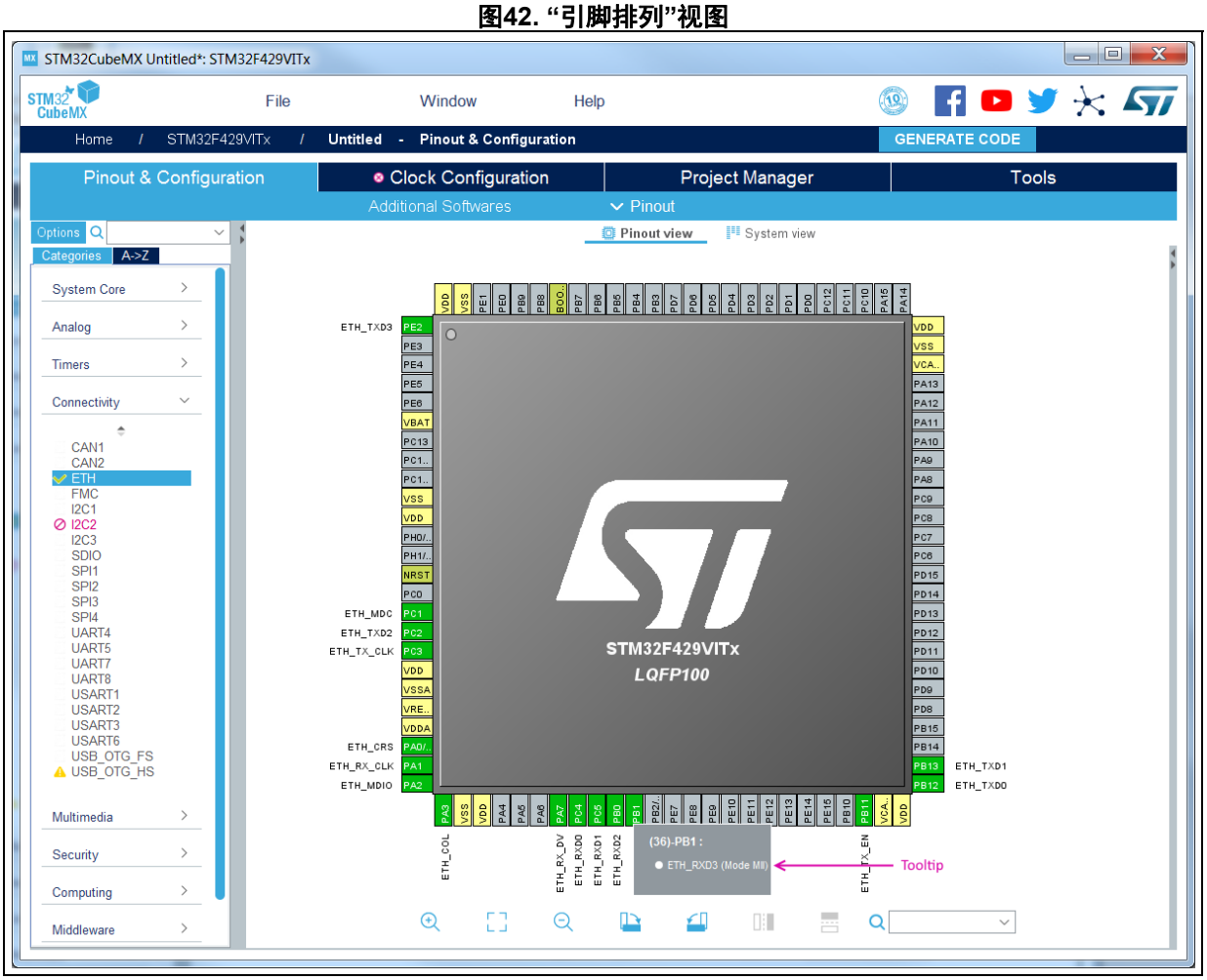
“模式”面板还可以为项目启用中间件和其他软件组件。

注： *对于某些中间件（USB、FATS、LwIP），在激活中间件模式之前必须使能外设模式。工具提示指导用户完成配置。对于FatFs，已经引入用户定义模式。该模式允许STM32CubeMX在没有预定义外设模式的情况下生成FatFs代码。然后，用户可以选择通过必要的代码更新生成的user_diskio.c/.h驱动文件，从而将中间件与用户定义的外设连接。*

4.4.3 “引脚排列”视图

选择 Pinout view 以显示所选料号，所选封装（例如BGA，QFP ...）的引脚布局的图示，其中每个引脚均以其名称（例如PC4）、其配置状态及其当前备用功能分配（如果有，例如ETH_MII_RXD0）来表示，请参见*图 42*获取示例。

图42. “引脚排列”视图



自动刷新**“引脚布局”**视图，以匹配**“模式”**面板中执行的用户组件配置。

直接通过**“引脚布局”**视图而不是**“模式”**面板分配引脚需要充分了解MCU，因为每个单独的引脚都可以分配给特定功能。

提示和技巧

若需获取菜单和快捷键列表，请参阅*表 2：主页快捷键*。

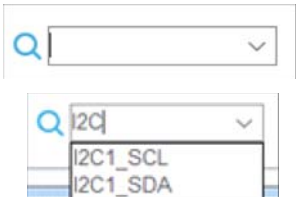
- 使用鼠标滚轮放大和缩小。
- 单击并拖动芯片图以将其移动。
- 点击最适合将其重设到最适合的位置和大小。
- 使用“引脚布局”>“导出引脚布局”菜单以将引脚布局配置导出为.csv文本格式。
- 某些基本控制， 例如引脚一致性，都是内置式。详细信息，请参见*附录ASTM32CubeMX 引脚分配规则*。

4.4.4 “引脚布局”菜单和快捷键

表6. “引脚布局”菜单和快捷键

名称或图标	快捷键	说明
保持当前信号位置	*Ctrl-K*	防止移动引脚分配，以匹配新的外设工作模式。建议使用可单独阻止每个引脚分配的新固定特性，，不勾选此复选框。
显示用户标签	无	在“引脚布局”视图中显示用户定义的标签。
撤消模式和引脚布局	*Ctrl-Z*	撤销前面的配置步骤（逐个）。
恢复模式和引脚布局	*Ctrl-Y*	恢复已经撤销的步骤（逐个）。 **警告**（限制）：平台设置选项卡中的配置未恢复。
禁用所有模式	*Ctrl-D*	将已使能的所有外设和中间件模式复位为“已禁用”。相应地，在这些模式下配置的引脚（绿色）被复位为“未使用”（灰色）。 外设和中间件标签从绿色变为黑色（未使用时）或灰色（不可用时）。
清除引脚排列	*Ctrl-P*	在“引脚布局”视图中清除用户引脚布局配置。 请注意，此操作会将所有已配置的引脚恢复为复位状态，并禁用先前启用的所有外设和中间件模式（无论其是否使用引脚上的信号）。
Pins/Signals选项	*Ctrl-O*	打开一个窗口，显示所有已配置引脚的列表以及引脚上的信号名称，并显示一个“标签”字段，便于用户为列表的每个引脚指定标签名称。 要激活此菜单，必须至少配置一个引脚。 点击引脚图标，单独固定/取消固定引脚信号。 选中多行后点击右键，打开上下文菜单，并选择相应的操作，以便一次性固定或取消固定所有已选择的引脚信号。 点击列标题名称，按名称的字母顺序排序或按MCU上的位置排序。
清除单一映射信号	*Ctrl-M*	针对没有关联模式的信号（以橙色突出显示，非引脚固定），清除分配到引脚的信号。
列出与引脚排列兼容的MCU	*Alt-L*	提供最符合当前项目引脚配置的MCU列表。匹配可以是： – 精确匹配 – 具备硬件兼容性的部分匹配：引脚位置相同，引脚名称可能已更改 – 不具备硬件兼容性的部分匹配：所有信号都可以映射，不是全部在相同引脚位置 请参见*第 15 节：教程5：将当前项目配置导出到兼容MCU兼容MCU*。
导出引脚布局 有Alt.功能	-	将引脚配置生成为.csv文本文件，包括备用功能信息。
导出引脚布局 无Alt.功能	*Ctrl-U*	将引脚配置生成为.csv文本文件，不包括备用功能信息。

表6. “引脚布局”菜单和快捷键（续）

名称或图标	快捷键	说明
复位已使用的GPIO	*Alt-G*	打开一个窗口，在已配置的GPIO引脚总数中指定待释放的GPIO数。
设置未使用的GPIO	*Ctrl-G*	打开一个窗口，在尚未使用的GPIO引脚总数中指定待配置的GPIO数。 指定模式：输入、输出或模拟（推荐的优化功耗配置）。 **注意：** 使用此菜单之前，确保设置调试引脚（在SYS外设下可用），以访问微控制器调试设施。
布局重置	-	-
	-	放大“引脚布局”视图。
	-	将芯片引脚排列图调整到最佳尺寸。
	-	缩小“引脚布局”视图。
	-	顺时针旋转90度。
	-	逆时针旋转90度。
	-	在仰视图和俯视图之间水平翻转。
	-	在仰视图和俯视图之间垂直翻转。
	-	此搜索字段允许用户在“引脚布局”视图中搜索引脚名称、信号名称或信号标签。 找到后，“引脚布局”视图上闪烁显示符合搜索条件的引脚或引脚组。 点击“引脚布局”视图，停止闪烁。

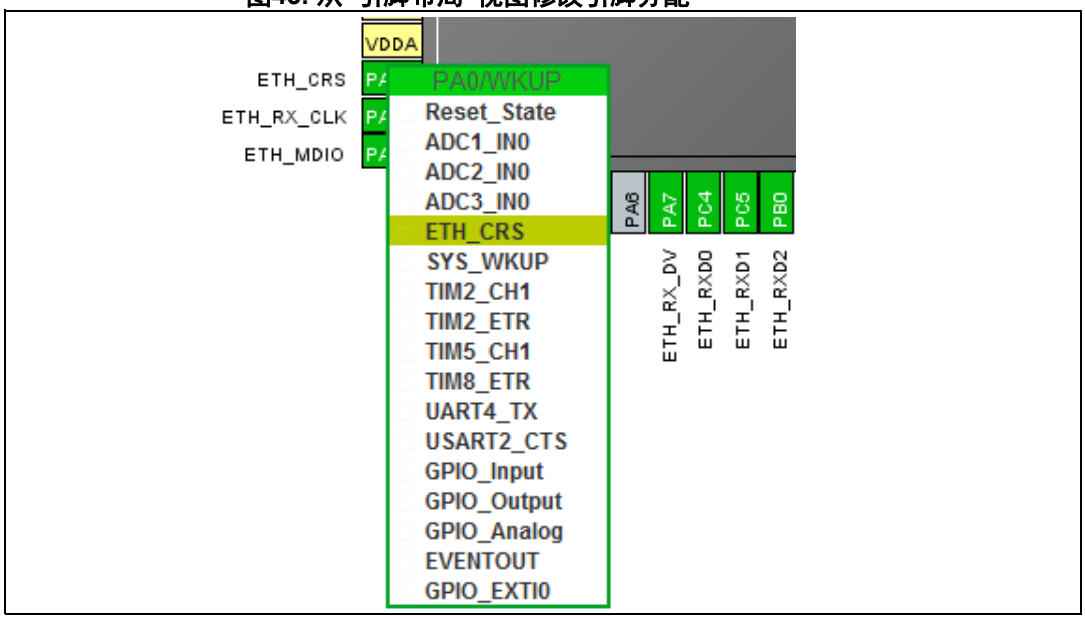
4.4.5 “引脚布局”视图高级操作

手动修改引脚分配

要手动修改引脚分配，请遵循以下顺序：

1. 在**引脚布局**视图中点击引脚以显示所有其他可能的替代功能列表，以及以蓝色突出显示的当前分配（参见*图 43*）。
2. 单击以选择要分配给引脚的新功能。

图43. 从“引脚布局”视图修改引脚分配



手动将功能重新映射到另一个引脚

要手动将功能重新映射到另一个引脚，请遵循以下顺序：

1. 按CTRL键并点击**引脚布局**视图中的引脚。可能用于重新定位的引脚（如有）用蓝色突出显示。
2. 将功能拖到目标引脚。

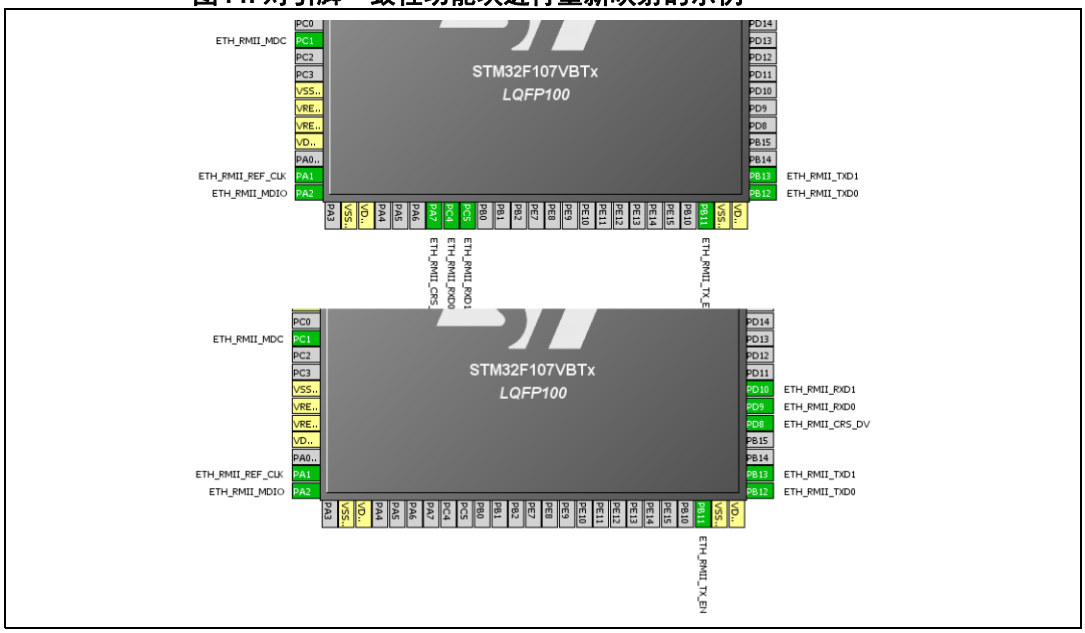
注意： 从“引脚布局”视图执行的引脚分配覆盖以前的任何分配。

手动重新映射有歧义的目标引脚

对于具有引脚一致性功能块的MCU（STM32F100x/F101x/F102x/F103xandSTM32F105x/F107x），目标引脚可能有歧义，例如可以有多个目标功能块（包括目标引脚）。要显示所有可能的备用重新映射功能块，请将鼠标移动到目标引脚上。

注： *“引脚功能块”是一组必须被分配到一起以实现特定外设模式的引脚。如*图 44中所示，STM32F107xx芯片上有两个引脚功能块，用于在RMII同步模式下配置以太网外设：{PC1,PA1,PA2,PA7,PC4,PC5,PB11,PB12,PB13,PB5}和{PC1,PA1,PA2,PD10,PD9,PD8,PB11, PB12, PB13, PB5}。

图44. 对引脚一致性功能块进行重新映射的示例



解决引脚冲突

为了解决当一些外设模式使用相同的引脚时可能发生的引脚冲突，STM32CubeMX尝试将外设模式功能重新分配给其他引脚。引脚冲突无法解决的外设用紫红色突出显示，并附有描述冲突的工具提示。

如果无法通过重新映射模式来解决冲突，用户可以尝试以下方法：

- 如果勾选了 ☑ Keep Current Signals Placement 框，尝试以不同的顺序选择外设。
- 取消选择**保持当前信号位置**框并让STM32CubeMX 尝试所有重新映射组合以找到解决方案。
- 当您不能使用外设模式时，将其**手动重新映射**，因为那个模式的一个信号没有引脚可用。

4.4.6 保持当前信号位置

该复选框可从**“引脚布局”**菜单中使用。可以在配置期间的任何时候选择或取消选择该复选框。默认情况下，该复选框处于未选中状态。

建议取消选中该复选框以优化外设的位置（同时使用的最大外设数量）。

当目标是匹配一个开发板设计时，应选中**保持当前信号位置**复选框。

“保持当前信号位置”未选中

由此STM32CubeMX将先前映射的功能块重新映射到其他引脚，以便服务与当前引脚排列配置冲突的新请求（选择新的外设模式或新的外设模式功能）。

“保持当前信号位置”已选中

这样可确保与给定外设模式对应的所有功能仍然分配（映射）到给定的引脚。一旦分配完成，STM32CubeMX不能将外设模式功能从一个引脚移动到另一个引脚。如果在当前引脚配置中可行，将提供新的配置请求。

该功能可用于：

- 锁定所有与外设相对应的引脚，这些引脚是使用**外设**面板配置的。
- 从**引脚布局**视图进行手动重新映射时，维持一个映射到引脚的功能。

建议

如果某个模式不可用（以紫红色突出显示），请尝试为该模式找到另一个引脚重新映射配置，步骤如下:

1. 从引脚布局视图逐一取消选择指定的功能，直到模式再次可用为止。
2. 然后，再次选择模式，并使用新的序列继续引脚排列配置（参见*附录A STM32CubeMX引脚分配规则*获取重新映射示例）。因为该操作很耗时，建议取消选择**保持当前信号位置**复选框。

注： *即使未选中“保持当前信号位置”，STM32CubeMX也不会移动GPIO_功能（除GPIO_EXTI功能外）。*

4.4.7 在引脚上锁定和标记信号

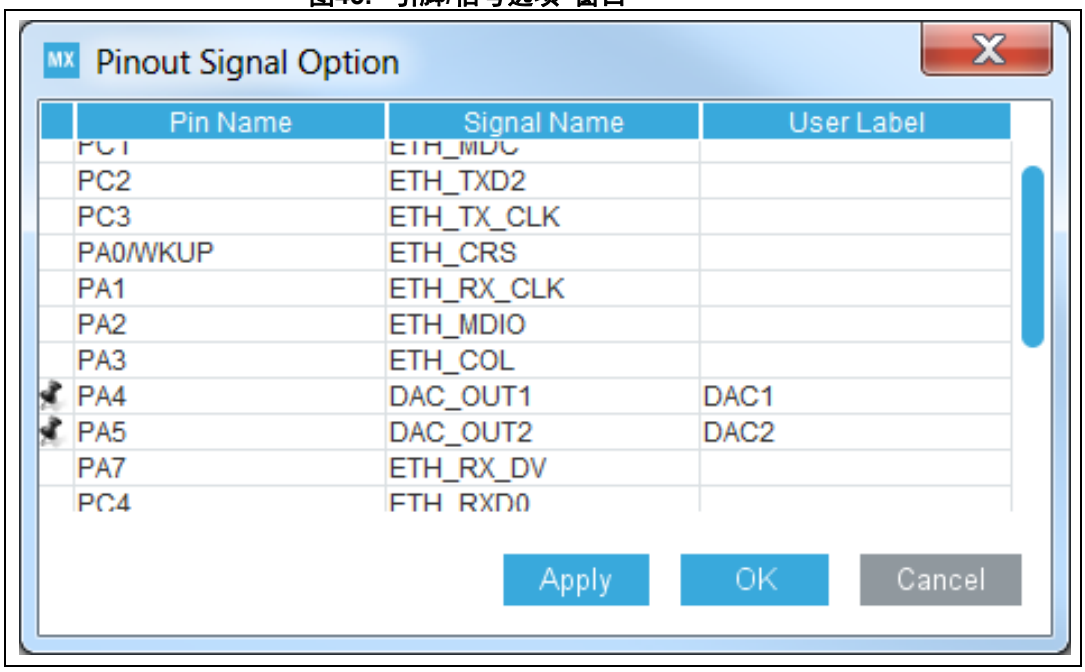
STM32CubeMX有为用户提供了一项功能以选择性地锁定信号到引脚。这样可防止STM32CubeMX在解决冲突时自动将锁定的信号移动到其他引脚。用于代码生成的标签也可以分配给信号（参阅*第 6.1 节*获取详细信息）。

有几种方法可以锁定、取消锁定和标记信号：

1. 从**引脚布局**视图，右键单击带信号分配的引脚。该操作会打开一个上下文菜单：
   a) 对于取消锁定的信号，选择**信号锁定**可以锁定信号。然后，相关引脚上会显示引脚图标。信号不能再自动移动（例如在解决引脚分配冲突时）。
   b) 对于锁定的信号，选择**信号取消锁定**可以取消锁定信号。引脚图标将被去除。从现在开始，要解决一个冲突（比如外设模式冲突），该信号可以移动到另一个引脚，前提是取消选中“保持用户位置”选项。
   c) 选择**输入用户标签**以指定此信号的用户定义标签。新标签替换**引脚布局**视图中的默认信号名称。

2. 从**引脚排列**菜单，选择**引脚/信号选项**
“引脚/信号选项”窗口（参见*图 45*）列出所有已经配置的引脚。

图45. “引脚/信号选项”窗口



a) 单击第一列以分别锁定/取消锁定信号。
b) 选择多行，右键单击以打开上下文菜单，并选择**“信号锁定”**或**“取消锁定”**。
c) 选择“用户标签”字段编辑该字段并输入用户定义的标签。
d) 单击列标题，按引脚或信号名称字母顺序排列列表顺序。再点击一次回到默认状态，即按MCU上的引脚位置排序。

注： *即使信号已锁定，仍然可以从**引脚布局**视图手动更改引脚信号分配：单击引脚以显示此引脚的其他可能信号并选择相关信号。*

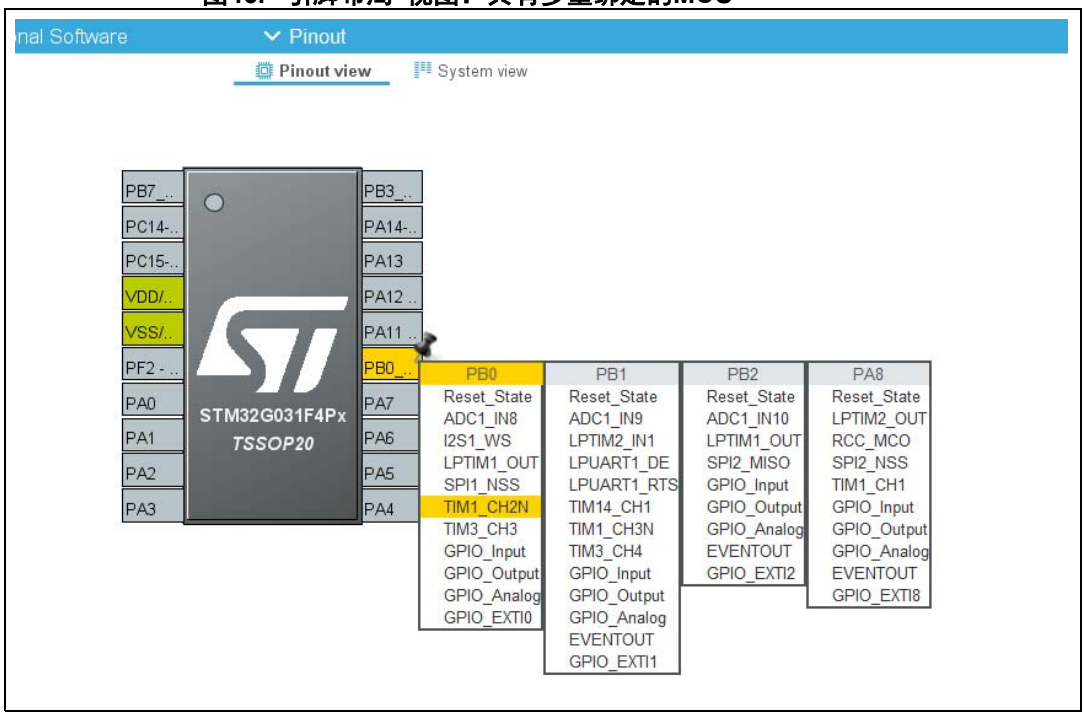
4.4.8 多重绑定封装的引脚布局

对于低引脚数（少于20个引脚）的封装，如SO8N、TSSOP20和WLCSP18封装，引入了多重绑定。它是由多个MCU焊盘在封装上共享同一引脚。

STM32G0系列引入了多重绑定以用于STM32G031/G041 MCU。

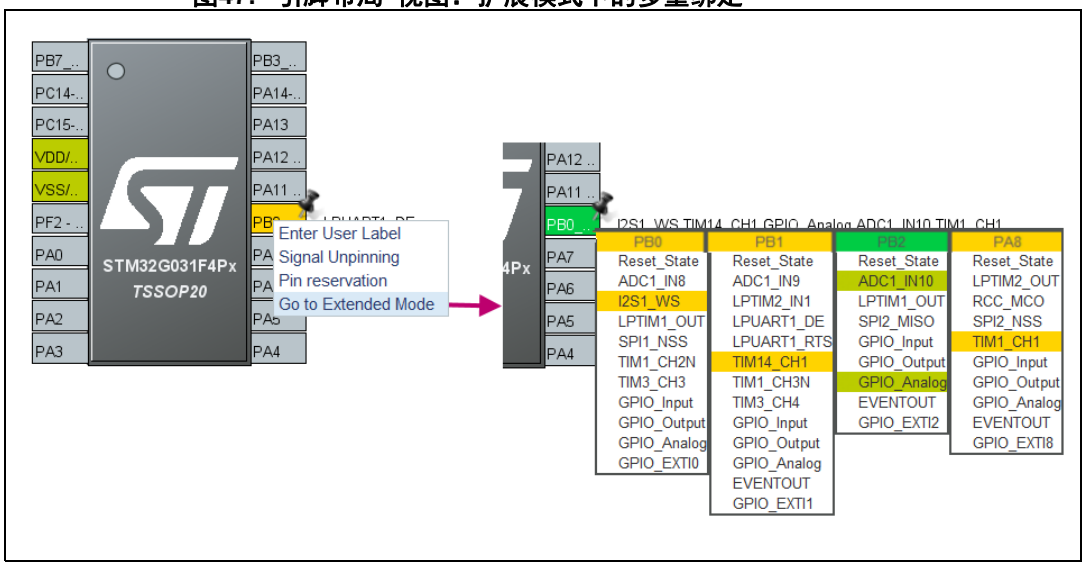
STM32CubeMX“引脚布局”视图可显示到达引脚的所有信号，还允许每个引脚只选择一个信号，但可以与其他模拟GPIO组合的模拟信号除外。

图46. “引脚布局”视图：具有多重绑定的MCU



STM32CUbeMX还提供了一种通过右键单击引脚来选择的扩展模式：该模式允许每个引脚选择多个信号。该模式用于测试目的，例如环回测试。请小心使用，因为这可能导致电气冲突或功耗增加，造成器件损坏。

图47. “引脚布局”视图：扩展模式下的多重绑定



4.4.9 系统视图

选择 System view 以显示所有软件可配置组件：GPIO、外设和中间件。可单击的按键使用户可以打开组件的模式和配置选项。按键图标反映了组件的配置状态（有关配置状态和“图系统”视图，请参阅*表 7*）。

当用户从“配置”面板更改组件配置时，系统视图将自动刷新成新的配置状态。

如果用户从“模式”面板禁用该组件，系统视图将自动刷新，并且不再显示该组件的按键。

图48. 系统视图

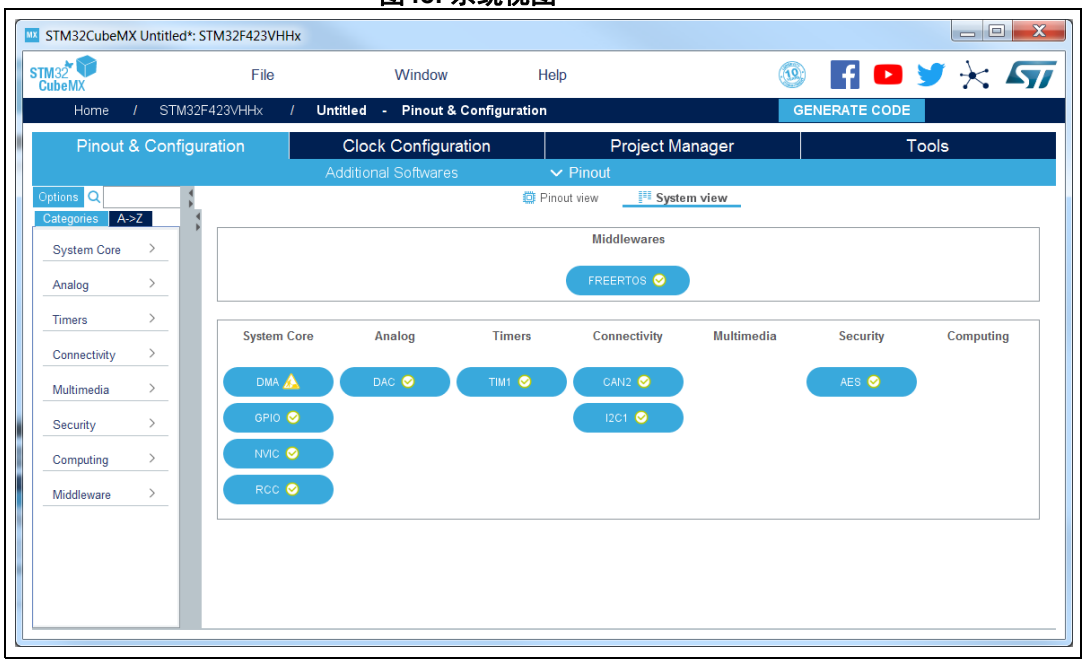
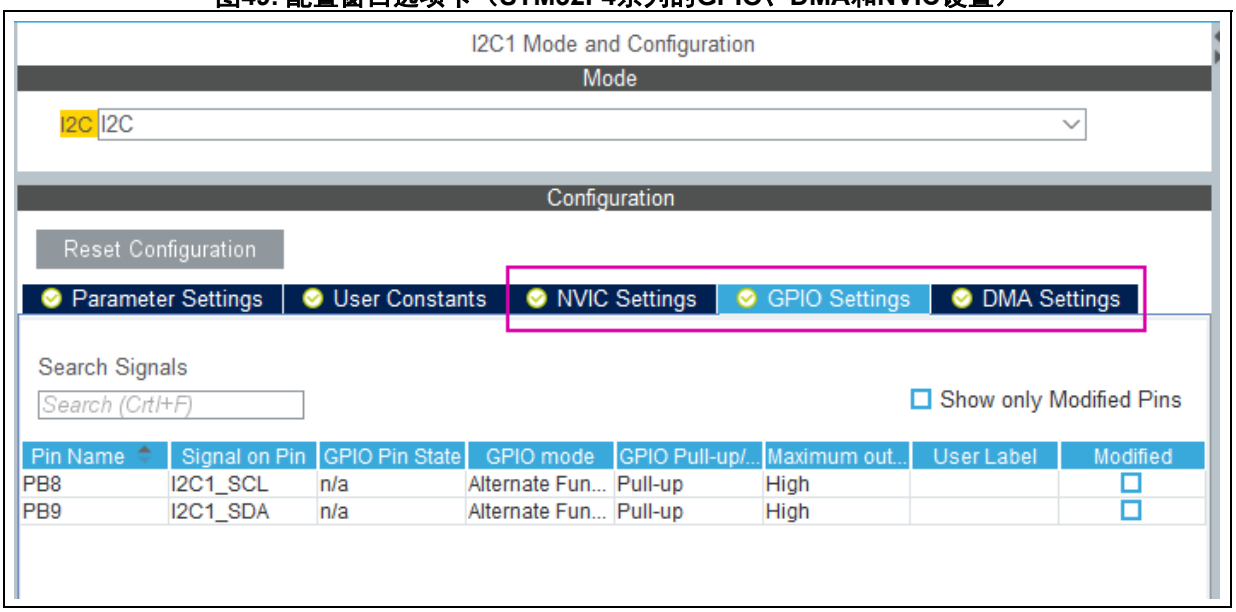


表7. 配置状态

图标	说明
	配置完成且正确。
	配置正确，但某些部分仍有待配置（可能是可选的）。
	配置无效，需要修复才能使生成的C项目正常运行。

GPIO、DMA和NVIC设置可以通过专用按键（如其他外设）或经由“配置”面板中的选项卡进行访问（请参阅*图 49*）。

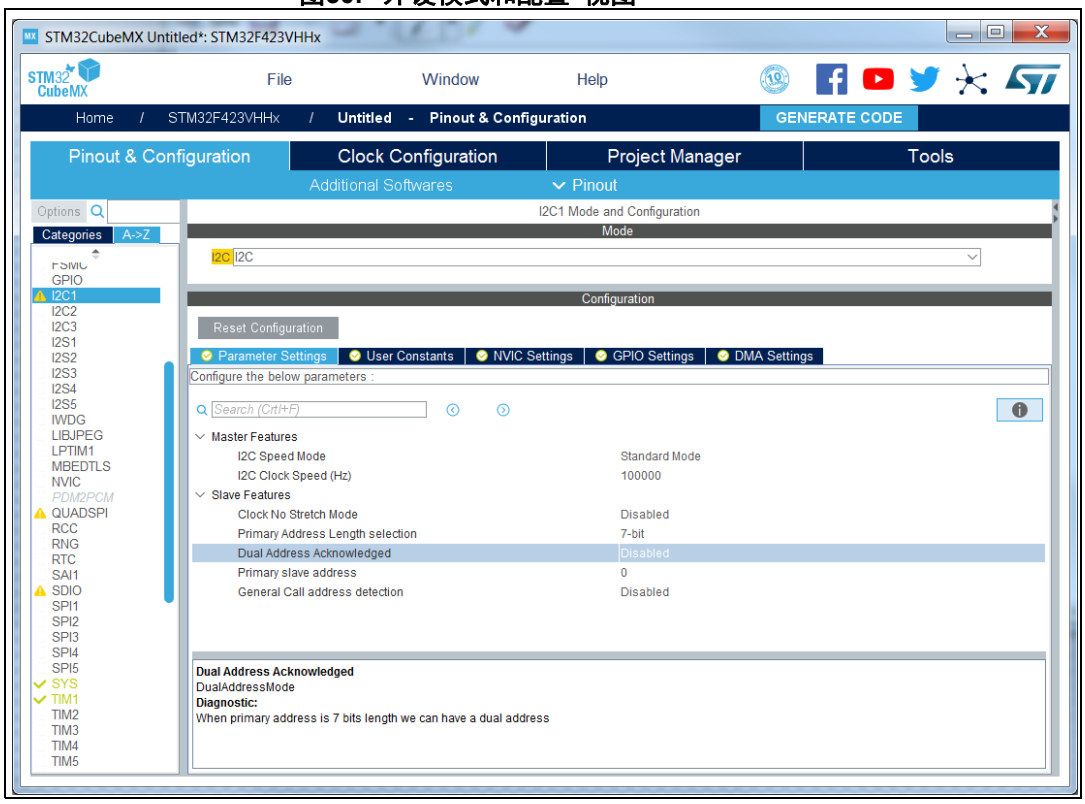
图49. 配置窗口选项卡（STM32F4系列的GPIO、DMA和NVIC设置）



4.4.10 组件配置面板

单击左侧面板中的组件名称时，将显示该面板。用户可使用该面板配置在选定的操作模式下初始化外设或中间件所需的功能参数（请参阅*图 50*）。STM32CubeMX使用这些设置来生成相应的初始化C代码。

图50. “外设模式和配置”视图



该配置窗口包含多个选项卡：

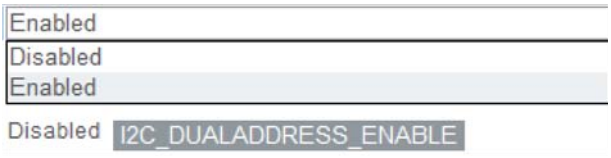
- **参数设置**为选定的外设或中间件配置库专用参数，
- **NVIC、GPIO和DMA设置**为所选外设设置参数（有关配置详细信息，参见*第 4.4.14 节*、*第 4.4.12 节*和*第 4.4.13 节*）。
- **用户常量** 创建一个或多个用户定义的常量，这对整个项目来说是通用的（有关配置详细信息，参见*第 4.4.11 节*）。

检测到无效设置，包括：

- 如果用户选择分别小于/大于最小/最大阈值，则重置为最小/最大有效值
- 如果先前的值既不是最大阈值也不是最小阈值，则重置为以前的有效值
- 以紫红色突出显示。

表 8 描述外设和中间件配置按钮和消息。

表8. “外设和中间件配置”窗口按钮与工具提示

按钮和消息	动作
	显示和隐藏描述面板。
工具提示 	引导用户以有效的最小-最大值范围完成参数设置。 如要显示工具提示，将鼠标移到可能值列表中的参数值上。
	单击齿轮图标可以选择是显示十六进制值或十进制值，还是未选中任何值（“未选中”选项）。
	搜索
	将组件重置为其默认配置（STM32CubeMX的初始设置）。

无检查选项

默认情况下，STM32CubeMX检查用户输入的参数值是否有效。您可以为给定参数选择无检查选项，以绕开该检查。这就允许输入任何STM32CubeMX配置可能不知道的值（比如常量）。

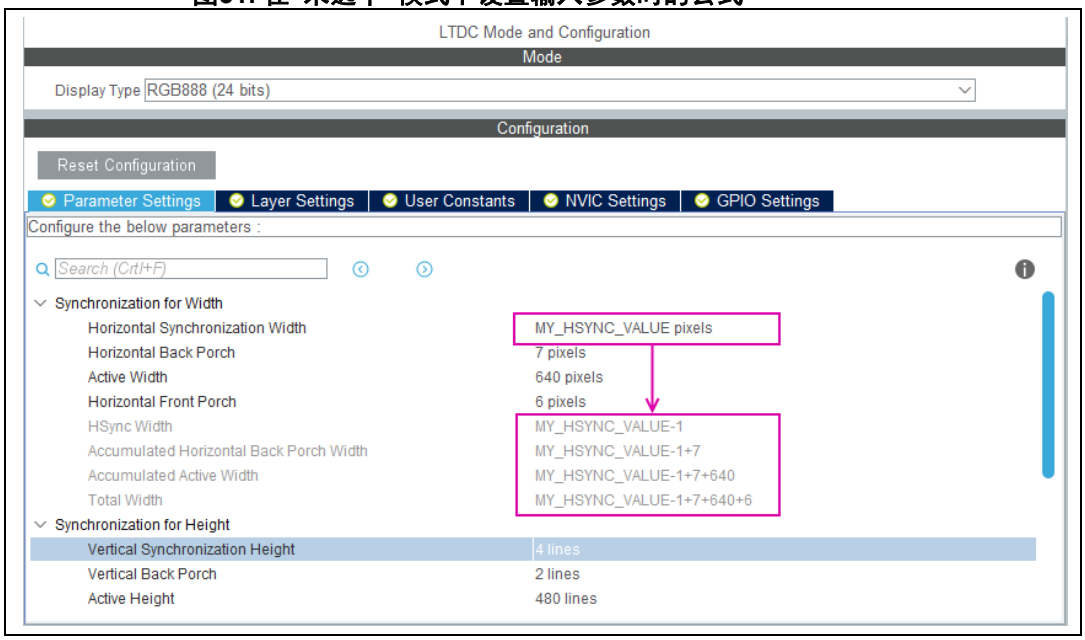
只有整数类型（十六进制或十进制）的参数才能绕过有效性检查。来自预定义的可能值列表的参数和非整数型或文本类型的参数均不能绕过有效性检查。

要回到默认模式（启用有效性检查的十进制或十六进制值），输入十进制或十六进制值并检查相关选项（十六进制或十进制检查）。

注意： 当一个参数依赖于另一个设为无检查的参数时：

- 一个参数依赖于另一个参数来评估其可能的最小值或最大值的情况：如果将其他参数设置为无检查，不再评估和检查最小值或最大值。
- 一个参数依赖于另一个参数来评估其当前值的情况：一个参数依赖于另一个参数来评估其当前值的情况。该值不再自动推导。相反，它被替换为公式文本，将设置为无检查的参数字符串显示为变量（参见*图 51*）。

图51. 在“未选中”模式下设置输入参数时的公式



4.4.11 ”用户常量“配置窗口

用户常量选项卡用于定义用户常量（参见*图 52*）。常量自动生成于main.h文件中的STM32CubeMX用户项目中（参见*图 53*）。一旦定义完成，它们就可以用于配置外设和中间件参数（参见*图 54*）。

图52. “用户常量”选项卡

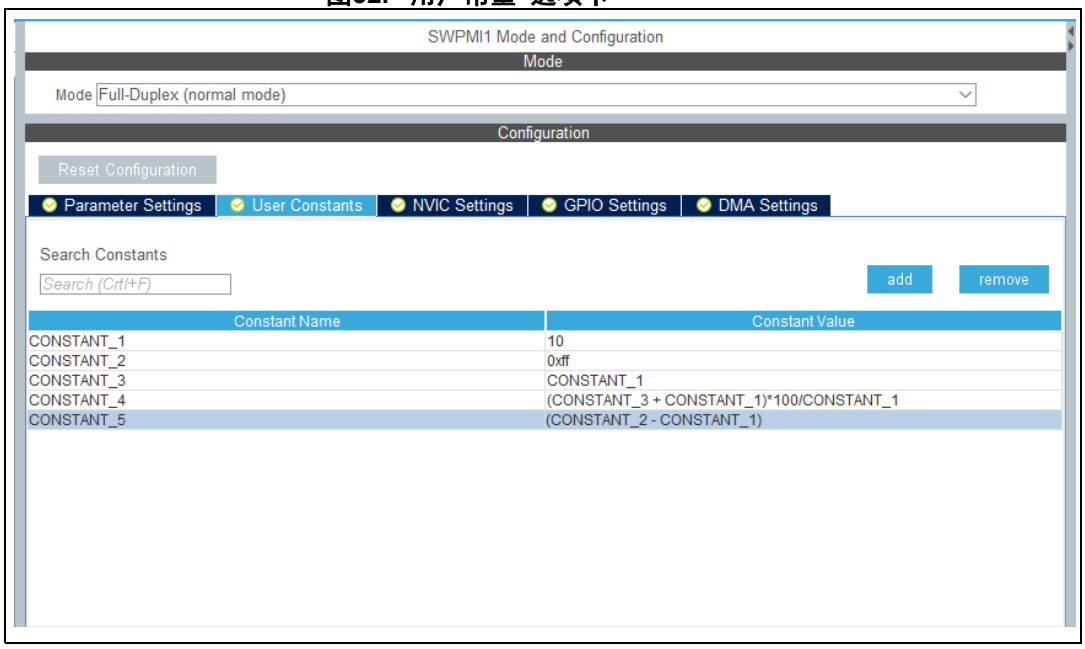
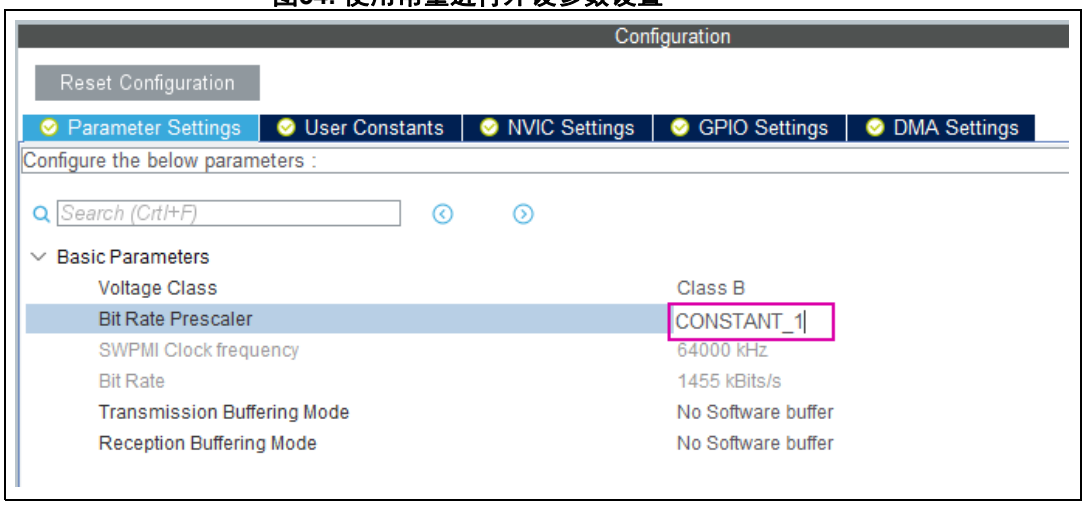


图53. 摘录生成的main.h文件

```
/* Includes ------------------------------------------------------------------*/

/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private define ------------------------------------------------------------*/
#define CONSTANT_1 10
#define CONSTANT_2 0xff
#define CONSTANT_3 CONSTANT_1
#define CONSTANT_4 (CONSTANT_3+CONSTANT_1)*100/CONSTANT_1
#define CONSTANT_5 (CONSTANT_2 - CONSTANT_1)

/* USER CODE BEGIN Private defines */

/* USER CODE END Private defines */
```

图54. 使用常量进行外设参数设置



创建/编辑用户常量

点击**添加**按钮打开**用户常量**选项卡并创建新的用户定义常量（参见*图 55*）。

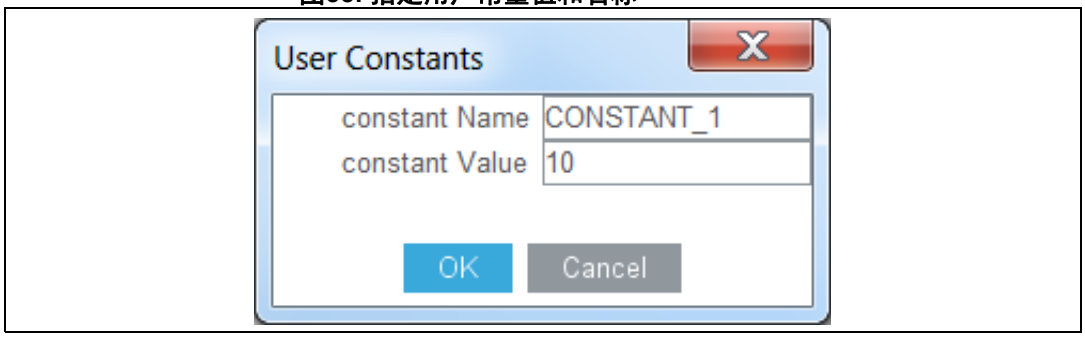
常量包括：

- 名称，必须符合下列规则：
  - 必须是唯一的。
  - 不应为C/C++关键字。
  - 不能包含空格。
  - 不应以数字开头。
- 一个值

  常量值可以是：（参见*图 52*获取示例）：
  - 一个简单的十进制或十六进制值
  - 一个先前定义的常量
  - 使用算术运算符（减法、加法、除法、乘法和余数）和数值或用户定义的数值常量作为操作数的公式
  - 一个字符串：字符串值必须在双引号之间（例如：“constant_for_usart”）。

一旦定义了常量，其名称和/或值仍然可以更改：双击指定要修改的用户常量的行。该操作会打开**用户常量**选项卡以便编辑。常量名的变更会应用到任何该常量使用的地方。这不会影响外设或中间件配置状态。然而，更改常量值会影响使用常量值的参数，并可能导致无效设置（例如，超过最大阈值）。无效的参数设置以紫红色突出显示。

图55. 指定用户常量值和名称



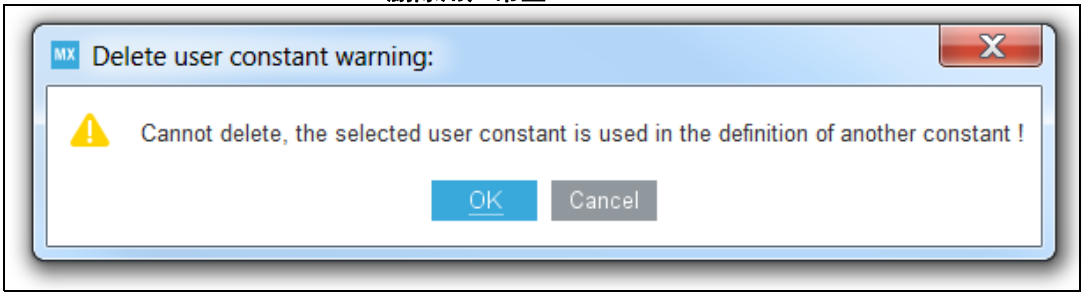
删除用户常量

点击**删除**按钮可删除现有的用户定义常量。

然后，用户常量被自动删除，除非遇到以下情况：

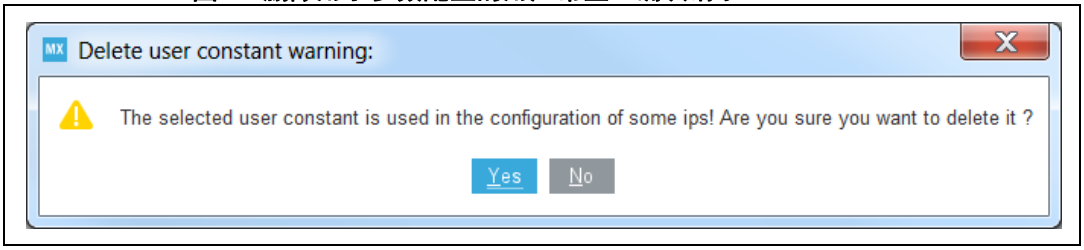
- 当常量用于定义另一个常量时。在这种情况下，弹出窗口将显示解释性消息（参见*图 56*）。

图56. 在常量已经用于另一个常量定义时，禁止删除用户常量



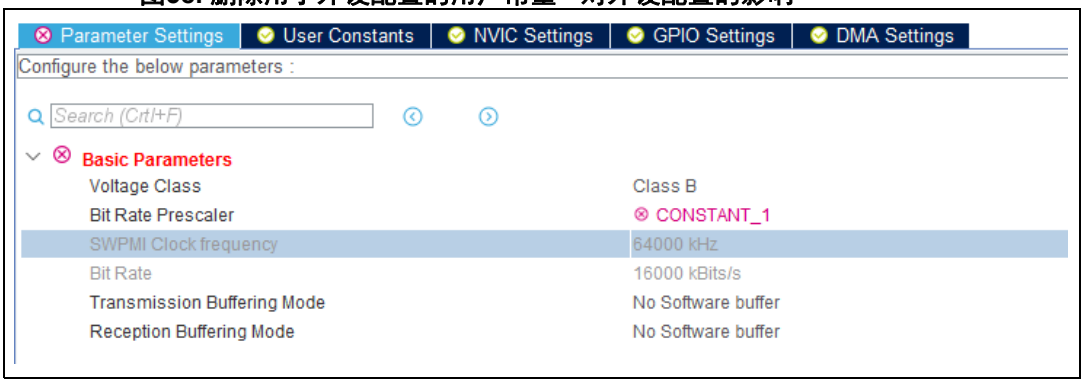
- 当常量用于外设库或中间件库参数的配置时。在这种情况下，请求用户确认删除，因为删除常量会导致无效的外设或中间件配置（参见*图 57*）。

图57. 删除用于参数配置的用户常量 - 确认请求



单击**“Yes（是）”**会导致无效的外设配置（参见*图 58*）

图58. 删除用于外设配置的用户常量 - 对外设配置的影响



检索用户常量

“搜索常量”字段可以在用户常量的完整列表中搜索常量名称或值（请参阅*图 59*和*图 60*）。

图59. 在用户常量列表中搜索常量名称

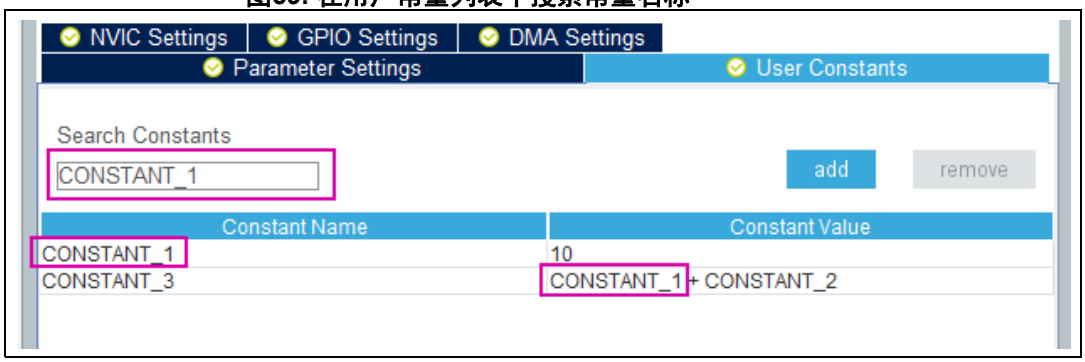
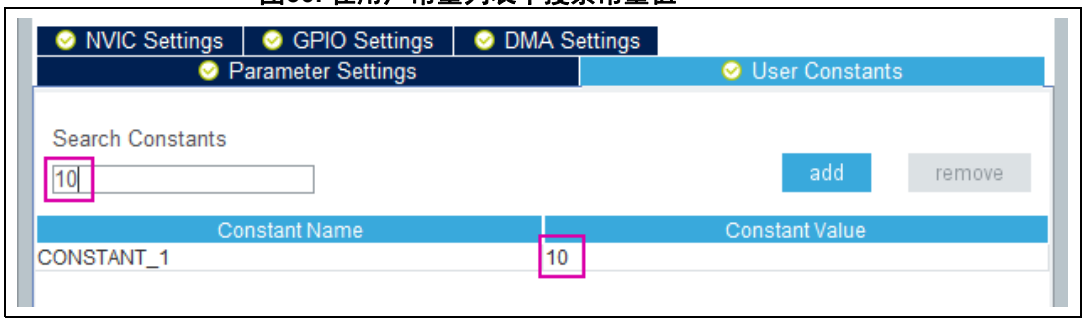


图60. 在用户常量列表中搜索常量值

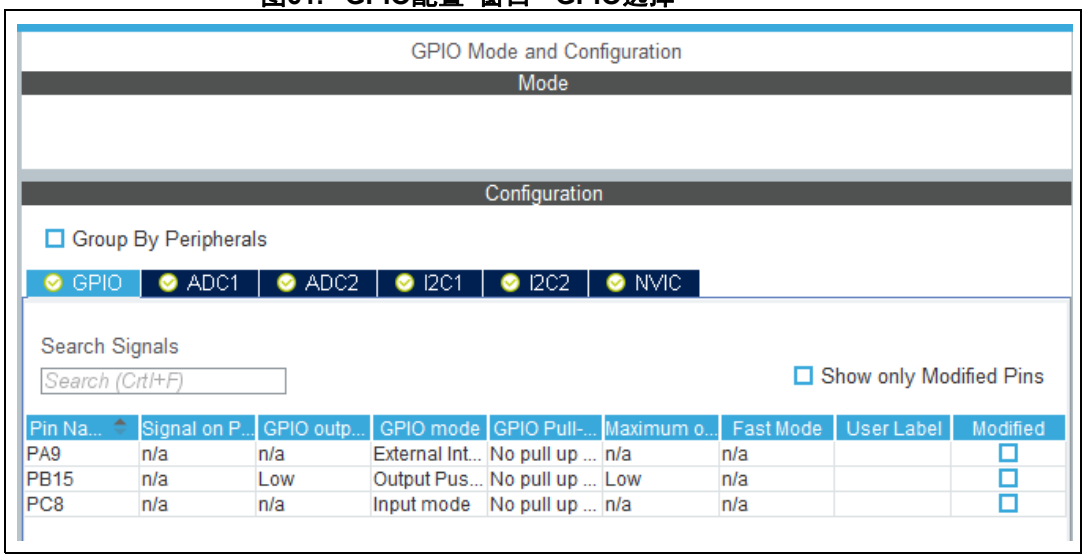


4.4.12 “GPIO配置”窗口

单击**“系统”视图**面板中的**GPIO**，以打开**GPIO配置**窗口，该窗口可用于配置GPIO引脚设置（请参阅*图 61*）。配置中填充的默认值可能不适用于某些外设配置。特别是，检查GPIO速度是否足以满足外设通信速度，并在需要时选择内部上拉。

注： *也可以通过外设实例配置窗口中的专用窗口访问特定外设实例的GPIO设置。另外，可以在输出模式（默认输出电平）下配置GPIO。将相应地更新生成的代码。*

图61. “GPIO配置”窗口 - GPIO选择

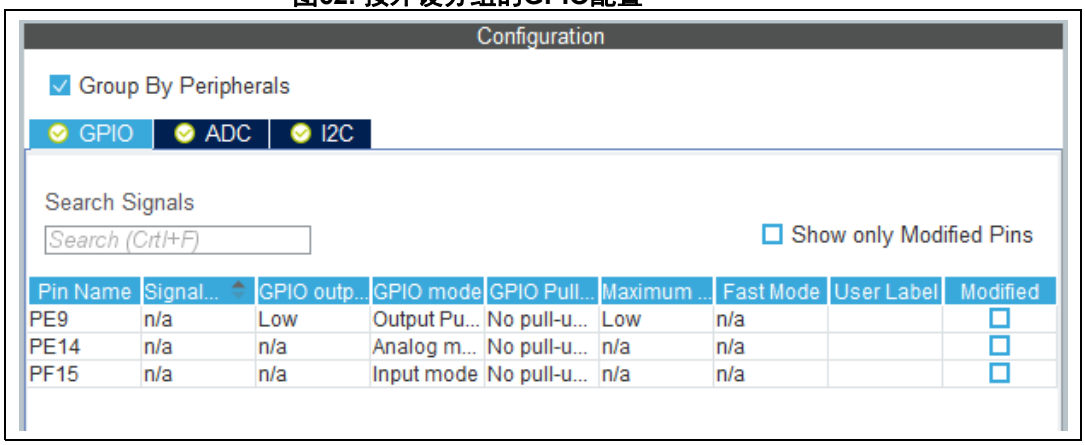


单击一行或选择一组行以显示相应的GPIO参数：

- **GPIO引脚状态**
  它更改GPIO输出电平的默认值。默认情况下，它被设置为低，可以更改为高。
- **GPIO模式**（模拟、输入、输出、备用功能）
  在**引脚排列**视图中选择外设模式会自动以相关的备用功能和GPIO模式配置引脚。
- **GPIO上拉/下拉**
  它被设置为一个默认值，可以在其他选项可用时进行配置。
- **GPIO最大输出速度**（仅面向通信外设）
  默认情况下，它被设置为低（为了功耗优化），可以更改为更高的频率以适应应用需求。
- **用户标签**
  它将默认名称（例如，GPIO_input）更改为用户定义的名称。**引脚布局**视图相应地更新。可以通过**“查找”**菜单在该新名称下找到GPIO。

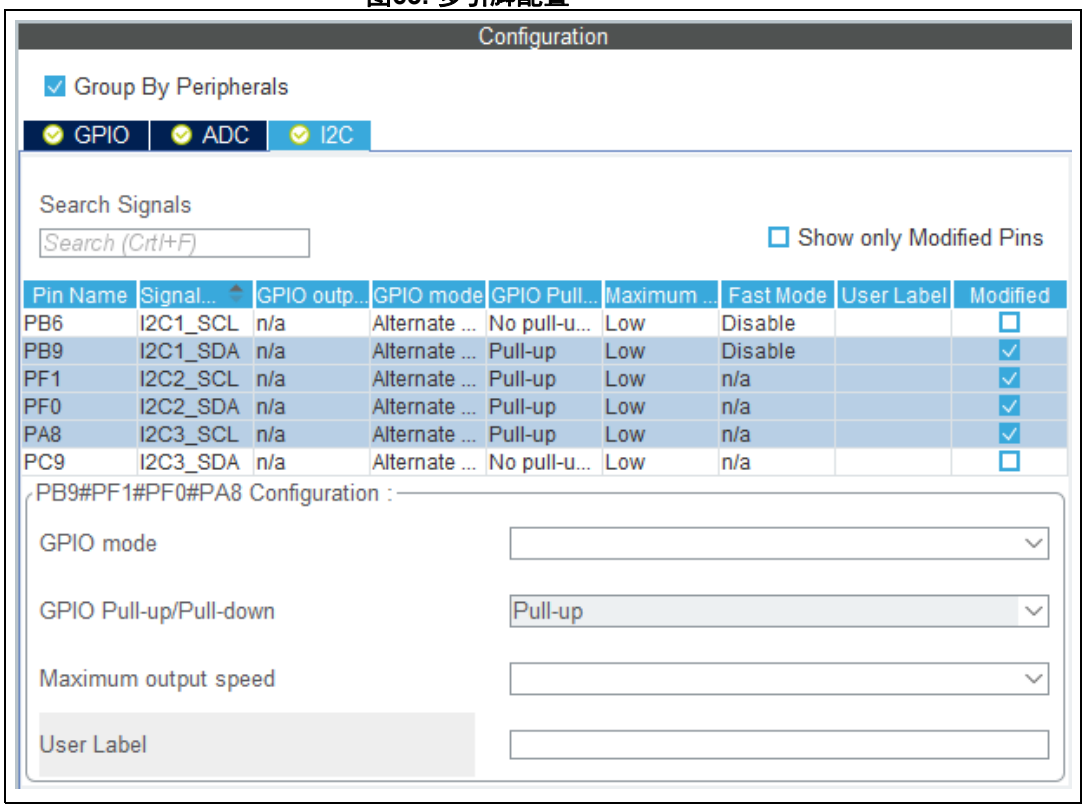
按外设分组复选框允许用户在同一个窗口下对所有外设实例进行分组（参见*图 62*）。

图62. 按外设分组的GPIO配置



如*图 63*中所示，**可**以对行执行多重选择，以同时将一组引脚更改为给定配置。

图63. 多引脚配置



4.4.13 “DMA配置”窗口

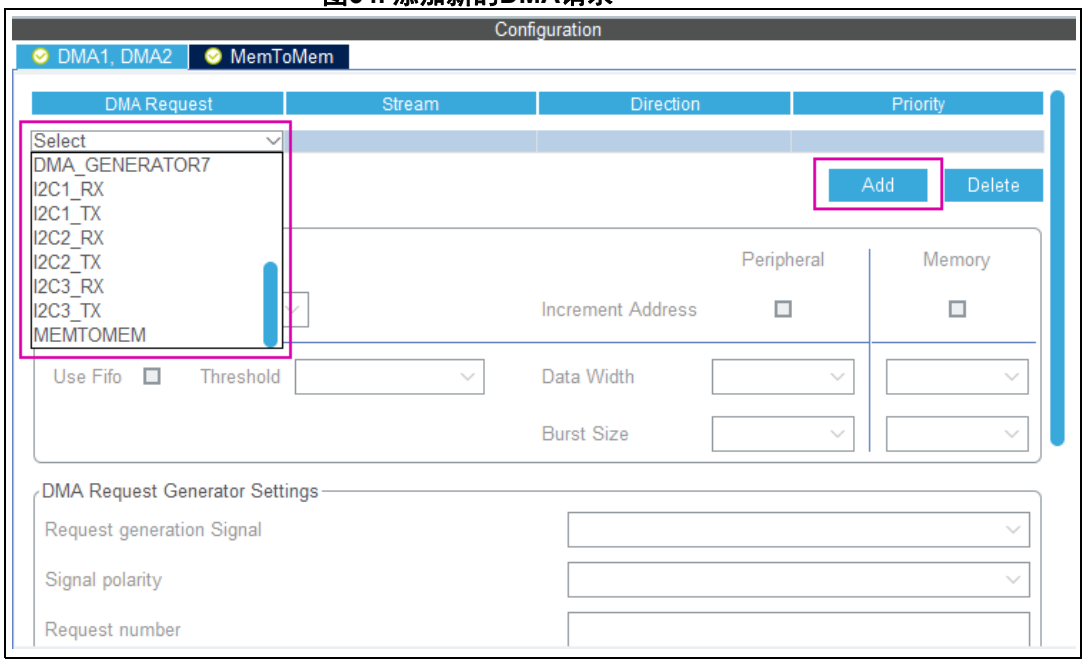
在**系统**视图中点击**DMA**可打开**DMA配置**窗口。

该窗口用于配置MCU上可用的通用DMA控制器。DMA接口允许在CPU运行时在内存和外设之间执行数据传输，以及内存到内存传输（如支持）。

注： *部分外设（例如**USB**或**以太网**）拥有自己的DMA控制器，默认情况下已使能或通过“外设配置”窗口使能。*

在**DMA配置**窗口点击**添加**在DMA配置窗口的末尾添加一个新行，其中有一个组合框，建议在映射到外设信号的可能的**DMA 请求**之间进行选择（参见*图 64*）。

图64. 添加新的DMA请求

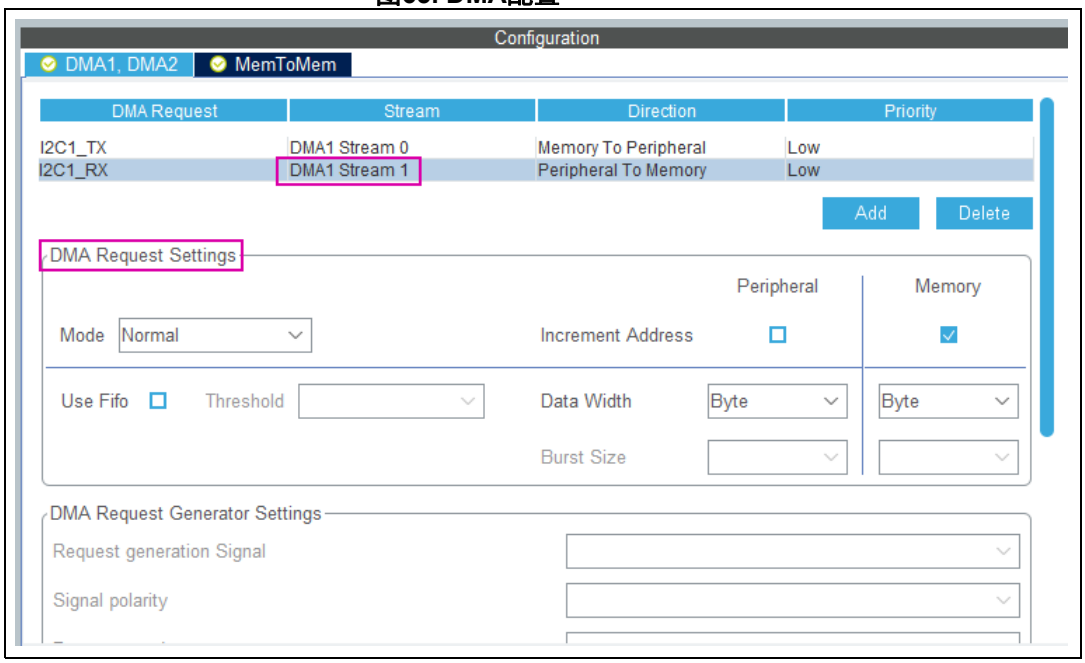


选择DMA请求将自动在所有可用的流、方向和优先级之间分配一个流。在配置DMA通道时，完全描述DMA传输运行时参数（如起始地址）取决于应用程序代码。

DMA请求（对于STM32F4 MCU而言，称为通道）用于保留一个数据流，以便在外设和内存之间传输数据（参见*图 65*）。流优先级用于决定为下一个DMA传输选择哪个流。

DMA控制器支持首先使用软件优先级的双重优先级系统，在软件优先级相同的情况下，硬件优先级由流编号提供。

图65. DMA配置

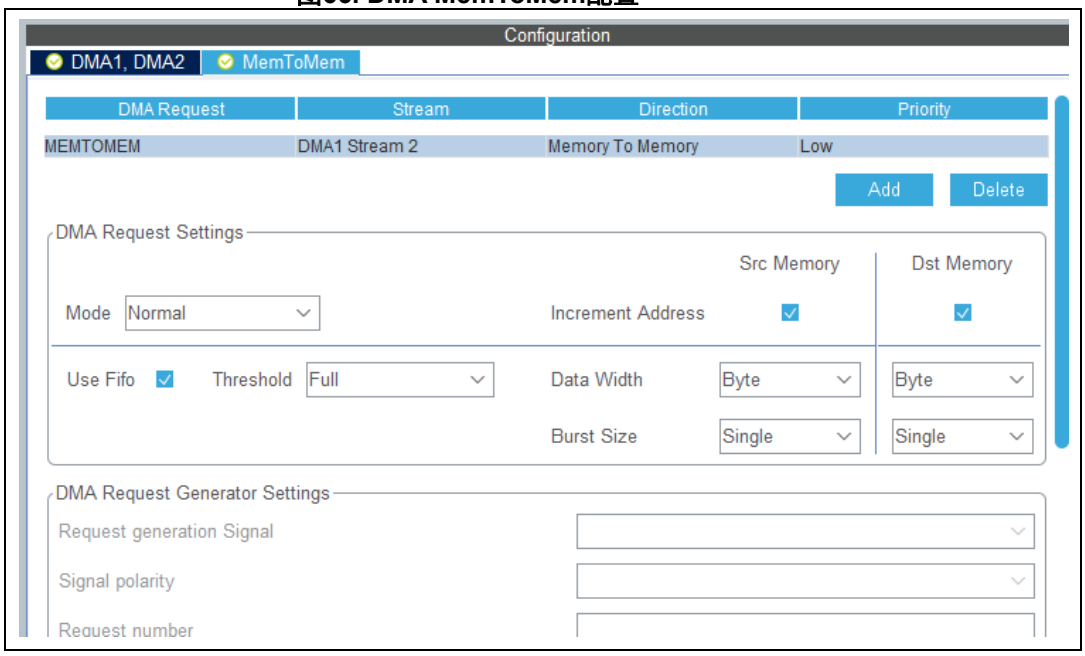


可以通过**DMA配置**窗口完成其他的DMA配置设置：

- ***模式***：常规模式、循环模式或外设流控制器模式（仅可用于SDIO外设）。
- ***递增添加***：外设地址和内存地址增量（固定或后递增，在这种情况下，地址在每次传输后都会递增）的类型。单击复选框以启用后递增模式。
- ***外设数据宽度***: 8、16或32位
- 从默认的直接模式切换到带可编程*阈值*的*FIFO模式*：
  a) 点击**使用FIFO**复选框。
  b) 然后，配置***外设和内存数据宽度***（8、16或32位）。
  c) 在***单一传输***和***批量传输***之间选择。如果选择批量传输，请选择批量数据大小（1、4、8或16）。

对于内存到内存传输（MemToMem），DMA配置适用于源内存和目标内存。

图66. DMA MemToMem配置



4.4.14 “NVIC配置”窗口

在**系统**视图中点击**NVIC**打开嵌套向量中断控制器配置窗口（参见*图 67*）。

中断非掩蔽和中断处理程序在两个选项卡中进行管理：

- **NVIC**选项卡允许在NVIC控制器中启用外设中断并设置其优先级。
- **代码生成**选项卡允许选择与中断相关的代码生成选项。

使用NVIC选项卡视图启用中断

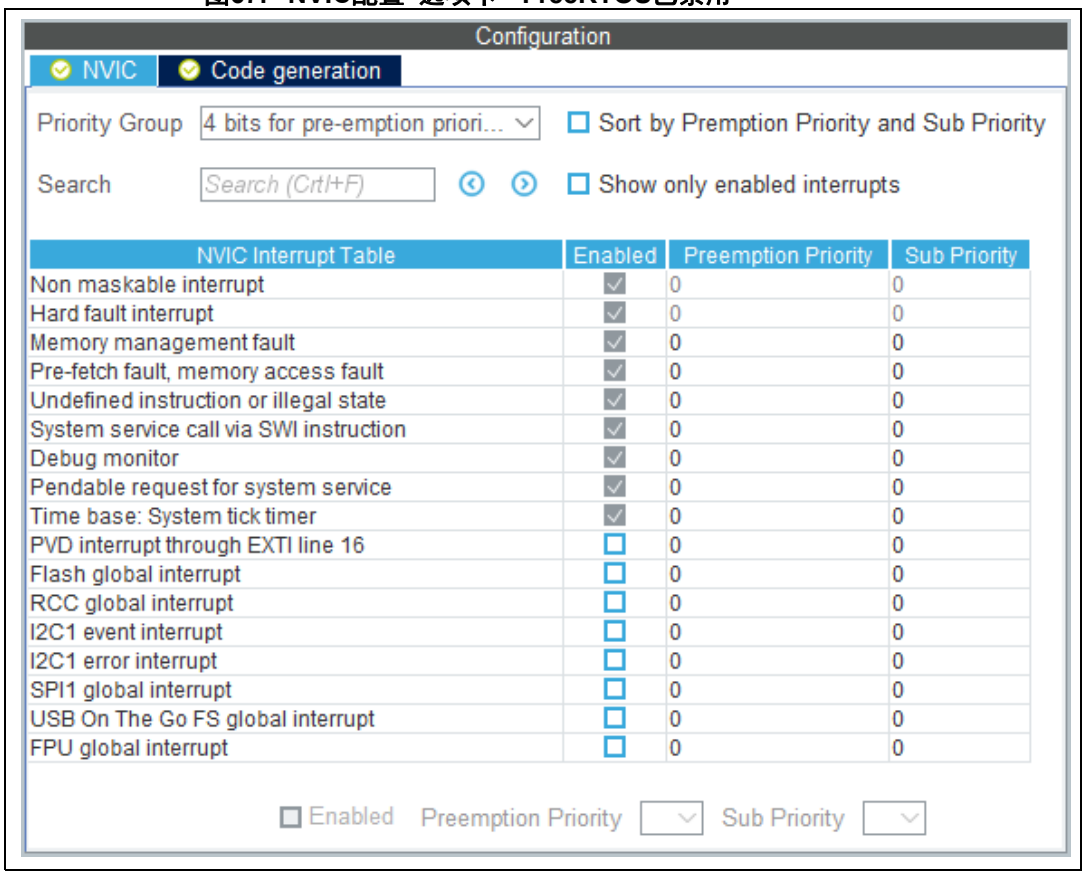
NVIC视图（参见*图 67*）不显示所有可能的中断，只显示在**引脚布局和配置**面板中选择的外设可用的中断。显示系统中断，但永远不能被禁用。

选中/取消选中**只显示使能的中断**框以筛选未启用的中断。

使用**搜索字段**根据字符串值筛选出中断向量表。例如，在从**引脚排列**面板启用UART外设后，在NVIC搜索字段中输入UART并点击旁边的绿色箭头：然后会显示所有UART中断。

启用**外设中断**将生成NVIC功能并为该外设调用**HAL_NVIC_SetPriority**和**HAL_NVIC_EnableIRQ**。

图67. “NVIC配置”选项卡 - FreeRTOS已禁用

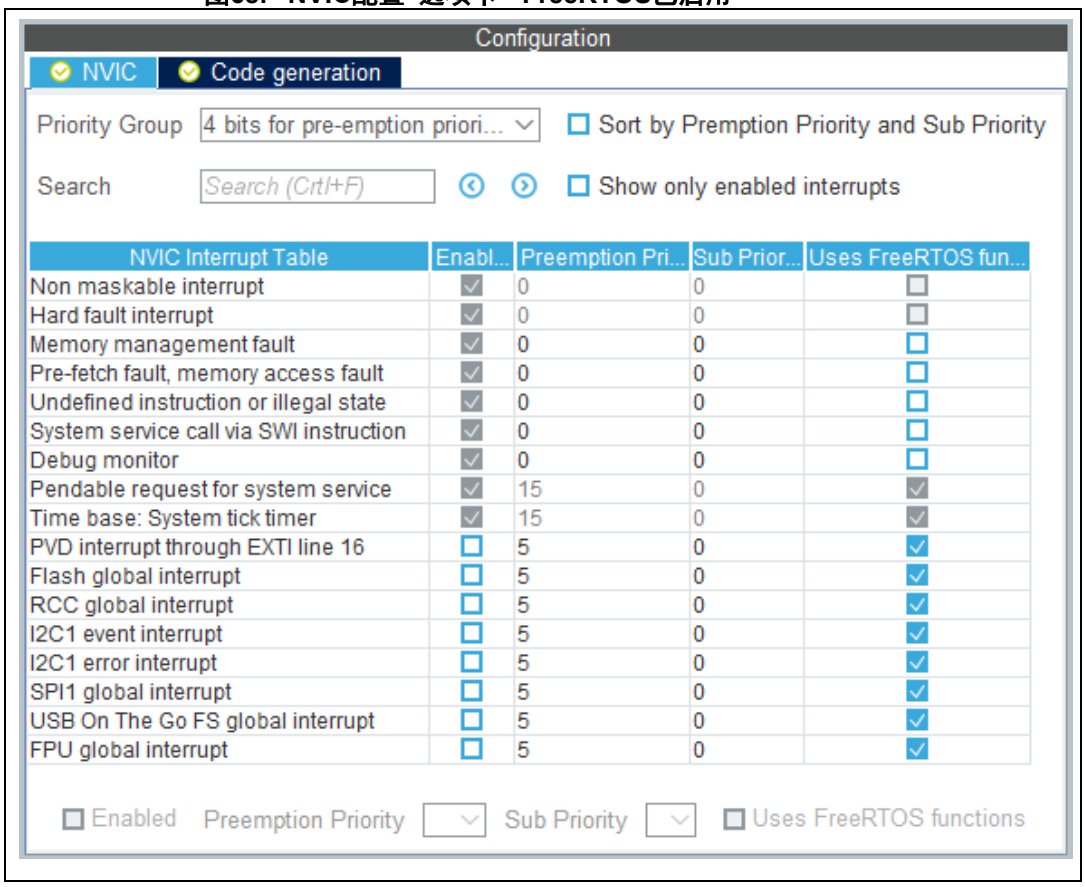


如果FreeRTOS已启用，将显示额外的列（参见*图 68*）。

在这种情况下，所有调用中断安全FreeRTOS API的中断服务程序（ISR）的优先级须低于LIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY参数中定义的优先级（值最高，优先级最低）。勾选相应复选框确保应用约束。

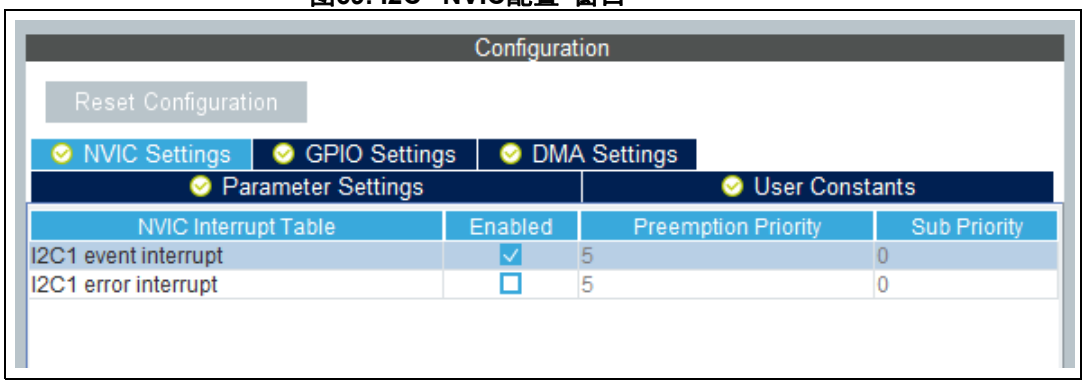
如果ISR不使用这些功能，则可以取消选中复选框并设置任何优先级。可以选中/取消选中多个行（参见在*图 68*中以蓝色突出显示的行）。

图68. “NVIC配置”选项卡 - FreeRTOS已启用



外设专用中断也可以通过“外设配置”窗口中的NVIC窗口访问（参见*图 69*）。

图69. I2C “NVIC配置”窗口



STM32CubeMX NVIC配置包括选择优先级组、启用/禁用中断和配置中断优先级（抢占和次优先级）：

1. 选择一个**优先级组**

   多个位，可用于定义NVIC优先级。这些位被分成两个优先级组，对应于两种优先级类型：抢占优先级和次优先级。例如，对于STM32F4 MCU，NVIC优先级组0对应于0位抢占和4位子优先级。

2. 在中断表中，单击一个或多个行以选择一个或多个中断向量。使用中断表下面的小部件以一次一个或一次几个的方式配置向量：
   - **启用复选框**：选中/取消选中以启用/禁用中断。
   - **抢占优先级**：选择一个优先级。抢占优先级定义一个中断中断另一个中断的能力。
   - **次优先级**：选择一个优先级。次优先级定义中断优先级。

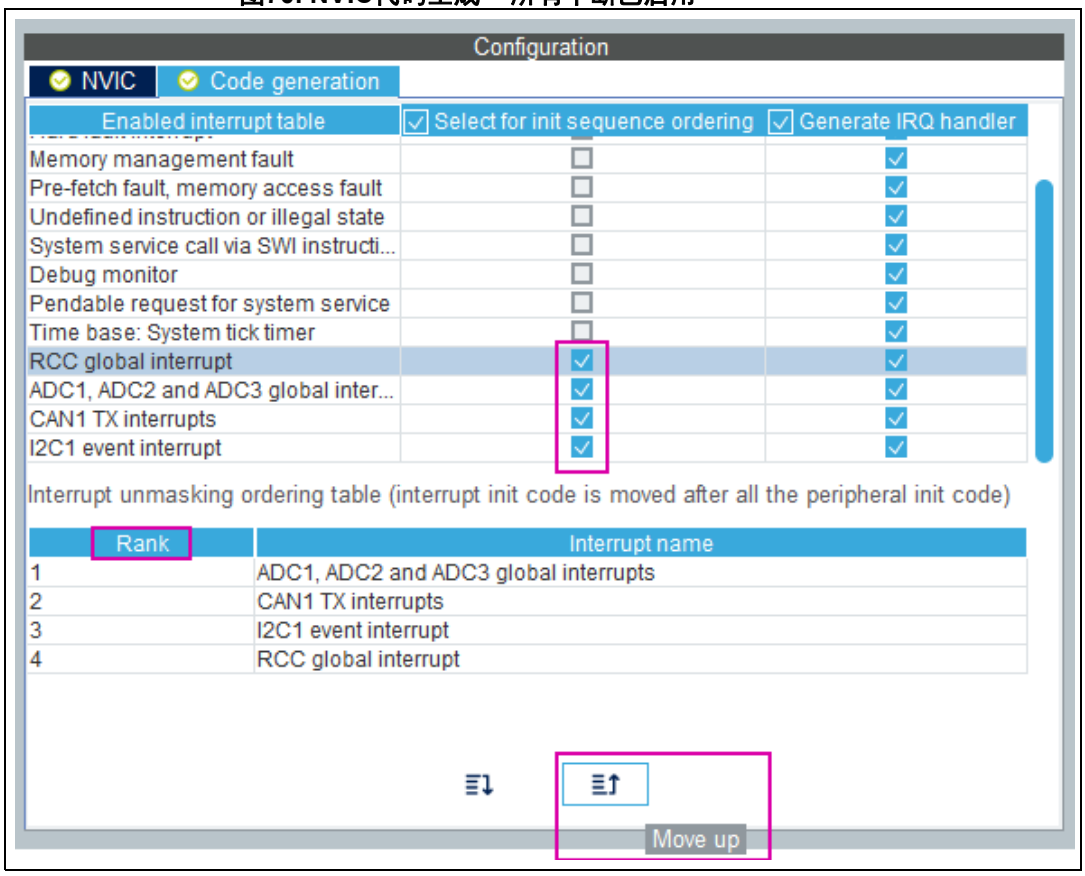
用于中断处理的代码生成选项

代码生成视图允许定制为中断初始化和中断处理程序生成的代码：

- **为序列排序和IRQ处理程序代码生成而选择/取消选择所有中断**

  使用列名前面的复选框一次配置所有中断（参见*图 70*）。注意，系统中断不适合初始化序列重排，因为软件解决方案并不对其进行控制。

图70. NVIC代码生成 – 所有中断已启用



- **中断的默认初始化序列**

  默认情况下，在配置GPIO和启用外设时钟之后，中断作为外设MSP初始化函数的一部分被启用。

  如下面的CAN示例所示，此例中的*HAL_NVIC_SetPriority*和*HAL_NVIC_EnableIRQ*函数在外设*msp_init*函数中的stm32xxx_hal_msp.c文件中被调用。

  中断启用代码以粗体显示：

```
void HAL_CAN_MspInit(CAN_HandleTypeDef* hcan)
{
GPIO_InitTypeDef GPIO_InitStruct;
if(hcan->Instance==CAN1)
{
/* 外设时钟启用 */
__CAN1_CLK_ENABLE();
/**CAN1 GPIO配置
PD0     ------> CAN1_RX
PD1     ------> CAN1_TX
*/
GPIO_InitStruct.Pin = GPIO_PIN_0|GPIO_PIN_1;
GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
```

```
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
GPIO_InitStruct.Alternate = GPIO_AF9_CAN1;
HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

/* 外设中断初始化 */
HAL_NVIC_SetPriority(CAN1_TX_IRQn, 2, 2);
HAL_NVIC_EnableIRQ(CAN1_TX_IRQn);
 }
}
```

只有对于**EXTI GPIO**，中断才在*MX_GPIO_Init*函数中启用：

```
/*配置GPIO引脚：MEMS_INT2_Pin */
GPIO_InitStruct.Pin = MEMS_INT2_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_EVT_RISING;
GPIO_InitStruct.Pull = GPIO_NOPULL;
HAL_GPIO_Init(MEMS_INT2_GPIO_Port, &GPIO_InitStruct);

/* EXTI中断初始化*/
HAL_NVIC_SetPriority(EXTI15_10_IRQn, 0, 0);
HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);
```

对于一些外设，应用程序仍然需要调用另一个函数以实际激活中断。以定时器外设为例，需要调用*HAL_TIM_IC_Start_IT*函数以便在中断模式下启动计时器输入捕获（IC）测量。

- **中断初始化序列配置**

  为一组外设选中**选择初始化序列排序**，将每个外设的HAL_NVIC函数调用移到main.c文件中定义的名为**MX_NVIC_Init**的相同专用函数。此外，每个外设的HAL_NVIC函数按照**代码生成**视图底部中指定的顺序被调用（参见*图 71*）。

  例如，*图 71*中所示的配置生成了以下代码：

```
/** NVIC配置
*/
void MX_NVIC_Init(void)
{
 /* CAN1_TX_IRQn中断配置*/
 HAL_NVIC_SetPriority(CAN1_TX_IRQn, 2, 2);
 HAL_NVIC_EnableIRQ(CAN1_TX_IRQn);
 /* PVD_IRQn中断配置*/
 HAL_NVIC_SetPriority(PVD_IRQn, 0, 0);
 HAL_NVIC_EnableIRQ(PVD_IRQn);
 /* FLASH_IRQn中断配置*/
 HAL_NVIC_SetPriority(FLASH_IRQn, 0, 0);
 HAL_NVIC_EnableIRQ(CAN1_IRQn);
 /* RCC_IRQn中断配置*/
 HAL_NVIC_SetPriority(RCC_IRQn, 0, 0);
 HAL_NVIC_EnableIRQ(CAN1_IRQn);
 /* ADC_IRQn中断配置*/
```

```
HAL_NVIC_SetPriority(ADC_IRQn, 0, 0);
HAL_NVIC_EnableIRQ(ADC_IRQn);
}
```

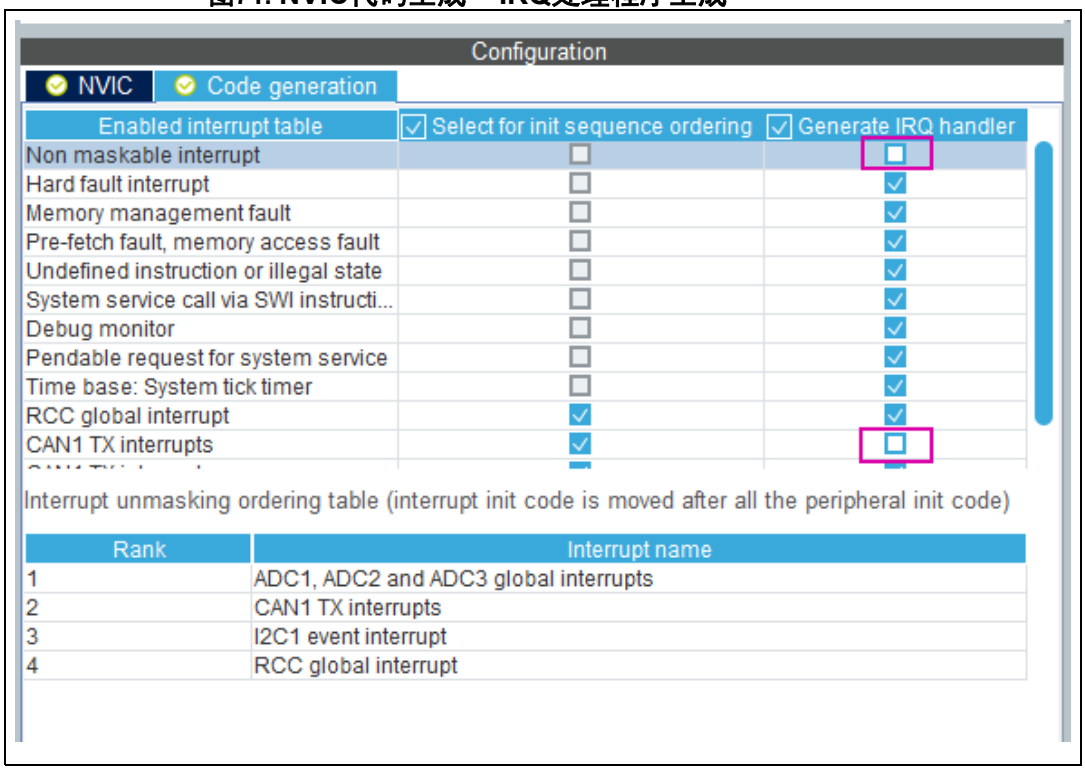
- **中断处理程序代码生成**

  默认情况下，STM32CubeMX在stm32xxx_it.c文件内生成中断处理程序。例如：

```
void NMI_Handler(void)
{
 HAL_RCC_NMI_IRQHandler();
}
void CAN1_TX_IRQHandler(void)
{
 HAL_CAN_IRQHandler(&hcan1);
}
```

  “生成IRQ处理程序”列允许用户控制是否生成中断处理程序函数调用。从**生成IRQ处理程序**列取消选择CAN1_TX 和NMI中断，如*图 71*中所示从stm32xxx_it.c文件删除前面提到的代码。

图71. NVIC代码生成 – IRQ处理程序生成

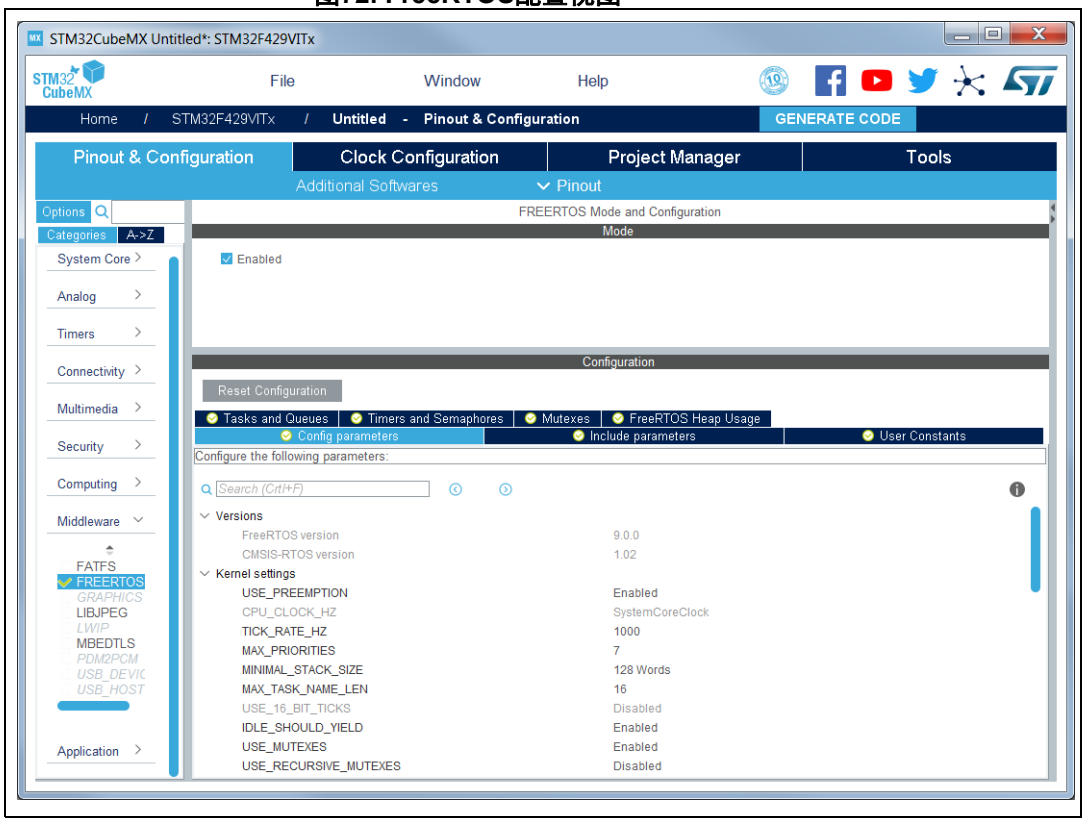


4.4.15 FreeRTOS配置面板

通过STM32CubeMX FreeRTOS配置窗口，用户可以配置实时操作系统应用所需的所有资源，并保留相应的堆。FreeRTOS元件在使用CMSIS-RTOSAPI函数生成的代码中定义和创建。其步骤如下：

1. 在**“引脚布局和配置”**选项卡中，单击FreeRTOS以显示“模式”和“配置”面板（请参阅*图 72*）。
2. 在模式面板中启用FreeRTOS。
3. 转到“配置”面板以继续配置FreeRTOS本机参数和对象，如任务、计时器、队列和信号标。在“配置”选项卡中，配置“内核”和“软件”设置。在“包含参数”选项卡中，选择应用程序所需的API函数，并以此优化代码大小。配置和包含参数均为FreeRTOSConfig.h文件的一部分。

图72. FreeRTOS配置视图



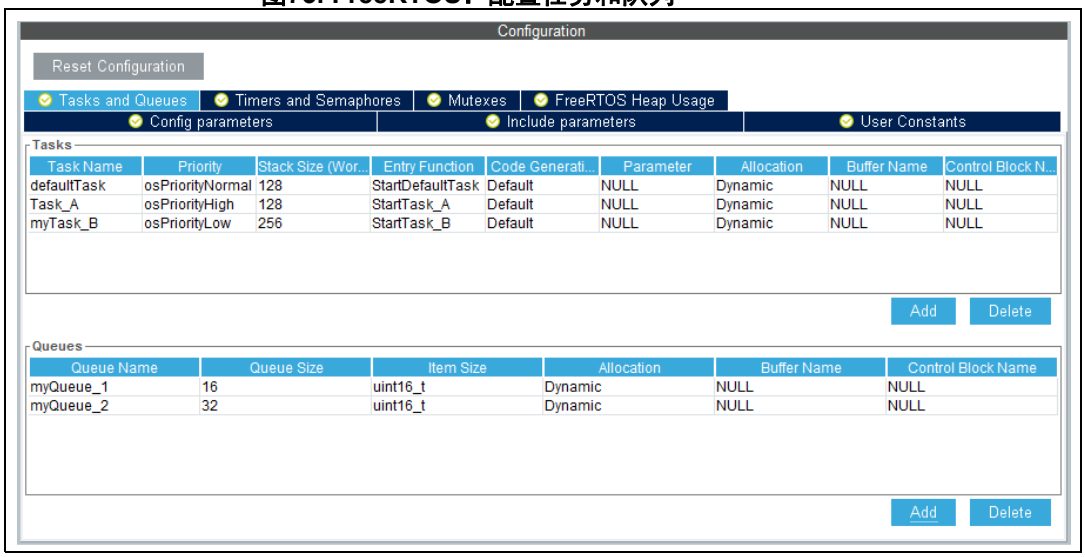
任务和队列选项卡

与任何RTOS一样，FreeRTOS允许将实时应用构建为一组独立任务，在给定时间只执行一个任务。队列用于任务间通信：它们允许在各任务之间或在中断与任务之间交换消息。

在STM32CubeMX中，**FreeRTOS任务和队列**选项卡允许创建和配置此类任务和队列（参见*图 73*）。如果在**项目设置**菜单中选择“生成作为每个外设和中间件的.c/.h文件对的代码”选项，则会在main.c或freeRTOS.c中生成相应的初始化代码。

如果在**“项目管理器”**菜单中选择“生成作为每个外设和中间件的.c/.h文件对的代码”选项，则会在main.c（默认）或freeRTOS.c中生成相应的初始化代码。

图73. FreeRTOS：配置任务和队列



- 任务

  在**任务**部分下，点击**“添加”**按钮，以打开**新建任务**窗口，可在其中配置任务**名称**、**优先级**、**堆栈大小**和**入口函数**（参见*图 74*）。可在任何时候更新这些设置：双击任务行可再次打开供编辑的新任务窗口。

  入口函数可以作为弱函数或外部函数生成：

  – 当任务作为**弱**任务生成时，用户可以提出非默认生成的另一个定义。

  – 当任务为**外部**任务时，由用户提供其函数定义。

  默认情况下，生成函数定义时包括允许定制的用户部分。

- 队列

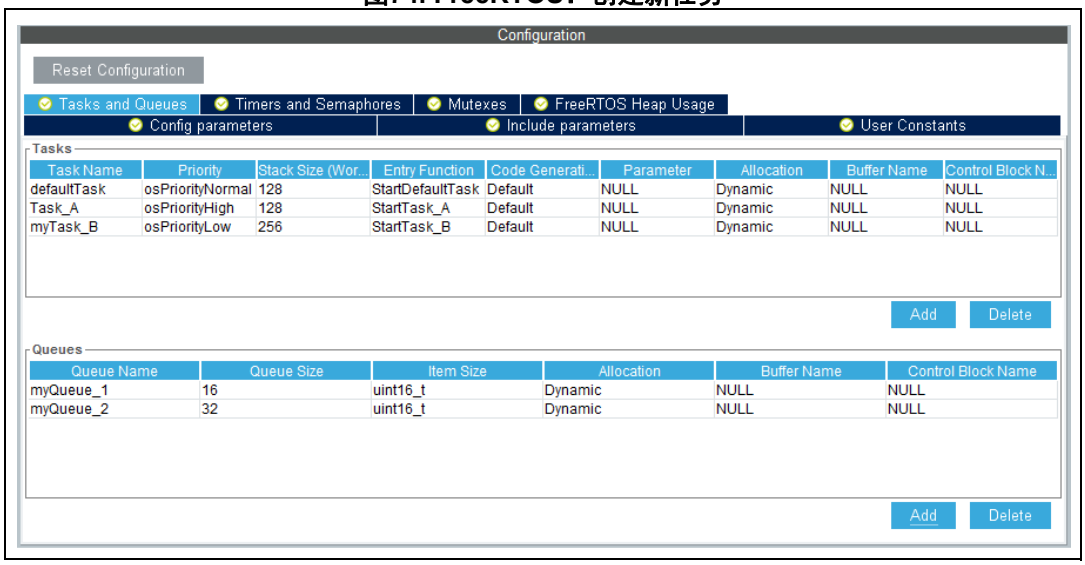
  在**队列**部分下，点击**添加**按钮，以打开**新建队列**窗口，可在其中配置队列**名称**、**大小**和**项目大小**（参见*图 74*）。队列大小对应于队列中一次可容纳的最大项目，而项目大小是存储在队列中的每个数据项的大小。项目大小可以用字节数或数据类型表示：

- 1字节对应uint8_t、int8_t、char和portCHAR类型
- 2字节对应uint16_t、int16_t、short和portSHORT类型
- 4字节对应uint32_t、int32_t、int、long和float类型
- 8字节对应uint64_t、int64_t和double类型

  默认情况下，当无法通过用户输入自动得出项目大小时，FreeRTOS堆用量计算器使用四字节。

  可在任何时候更新这些设置：双击队列行可再次打开供编辑的新队列窗口。

图74. FreeRTOS：创建新任务



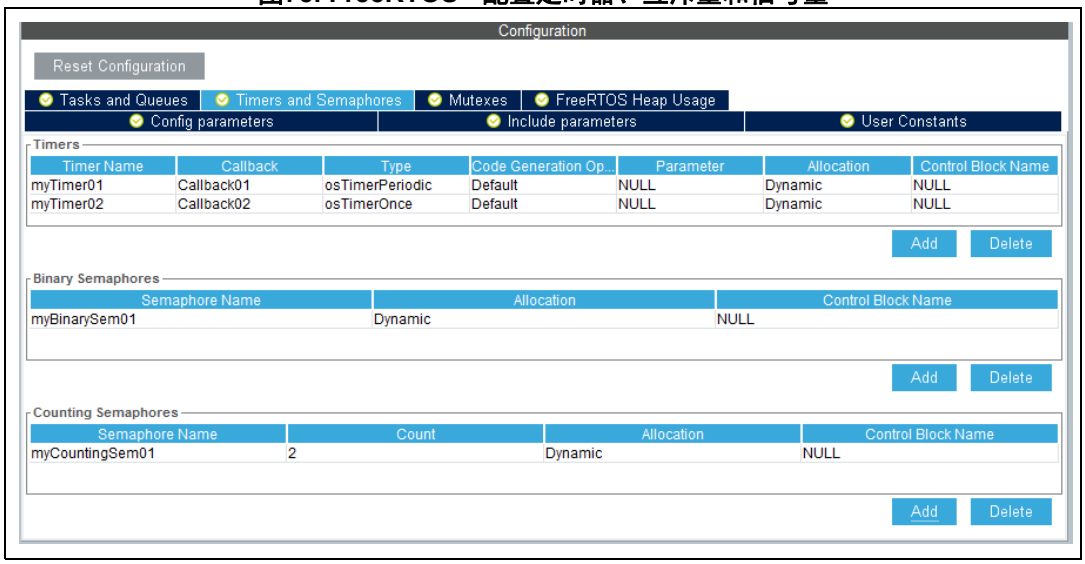
以下代码段显示了与*图 73*对应的生成代码。

```
/* 创建线程 */
 /* 定义和创建defaultTask */
 osThreadDef(defaultTask, StartDefaultTask, osPriorityNormal, 0, 128);
 defaultTaskHandle = osThreadCreate(osThread(defaultTask), NULL);

 /* 定义和创建Task_A */
 osThreadDef(Task_A, StartTask_A, osPriorityHigh, 0, 128);
 Task_AHandle = osThreadCreate(osThread(Task_A), NULL);

 /* 定义和创建Task_B */
 osThreadDef(Task_B, StartTask_B, osPriorityLow, 0, 256);
 Task_BHandle = osThreadCreate(osThread(Task_B), NULL);

 /* 创建队列 */
 /* 定义和创建myQueue_1 */
 osMessageQDef(myQueue_1, 16, 4);
 myQueue_1Handle = osMessageCreate(osMessageQ(myQueue_1), NULL);

 /* 定义和创建myQueue_2 */
 osMessageQDef(myQueue_2, 32, 2);
 myQueue_2Handle = osMessageCreate(osMessageQ(myQueue_2), NULL);
```

定时器、互斥量和信号量

FreeRTOS定时器、互斥量和信号量可通过FreeRTOS**定时器和互斥量**选项卡创建。首先需要从“配置”选项卡将其启用（请参阅*图 75*）。

图75. FreeRTOS - 配置定时器、互斥量和信号量



在每个对象专用部分下，点击**添加**按钮，以打开相应的**新建<对象>**窗口，可在其中指定对象特定参数。可在任何时候修改对象设置：双击相应行可再次打开供编辑的**新建<对象>**窗口。

注： *如果新创建的对象不可见，请展开窗口。*

- 时序

  在创建定时器前，必须在**配置参数**选项卡的**软件定时器定义部分**使能其使用（USE_TIMERS定义）。在同样的部分，也可以配置定时器任务优先级、队列长度和堆栈深度。

  可将定时器创建为单次触发（运行一次）或自动重载（周期性）。必须指定定时器名称和相应的回调函数名称。在调用CMSIS-RTOS osTimerStart函数时，由用户来填写回调函数代码和指定定时器周期（从定时器启动到执行其回调函数的时间）。

- 互斥量/信号量

  在创建互斥量、递归信号量和计数信号量之前，必须在**配置参数**选项卡的**内核设置**部分使能其使用（USE_ MUTEXES、USE_RECURSIVE_MUTEXES、USE_COUNTING_SEMAPHORES定义）。

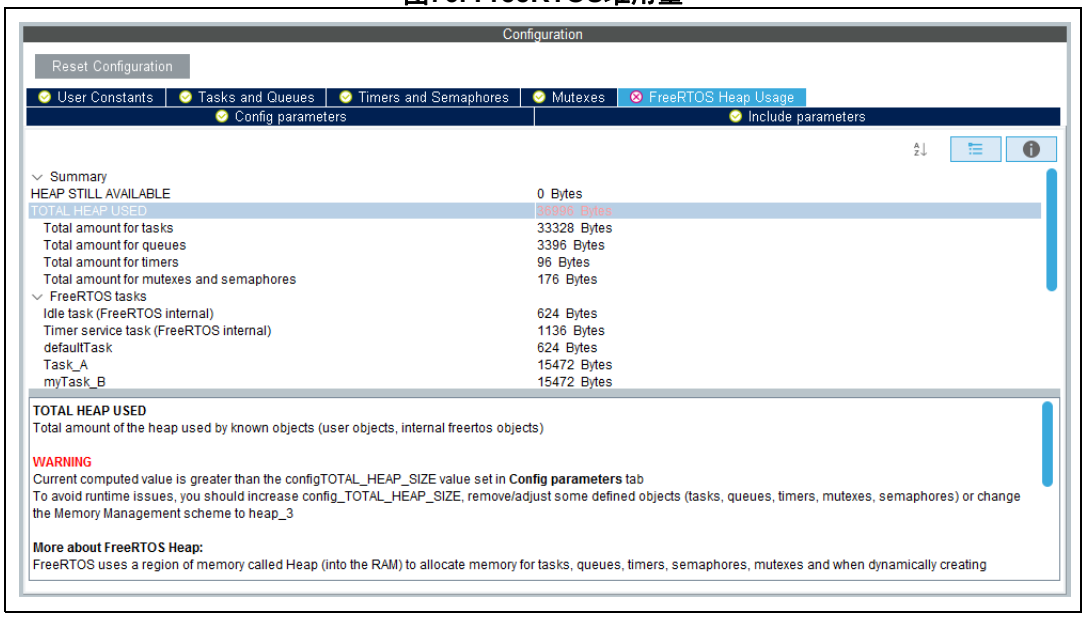
  以下代码段显示了与*图 75*对应的生成代码。

```
/* 创建信号量 */
/* 定义和创建myBinarySem01 */
osSemaphoreDef(myBinarySem01);
myBinarySem01Handle = osSemaphoreCreate(osSemaphore(myBinarySem01), 1);

/* 定义和创建myCountingSem01 */
osSemaphoreDef(myCountingSem01);
myCountingSem01Handle = osSemaphoreCreate(osSemaphore(myCountingSem01), 7);

 /* 创建定时器 */
/* 定义和创建myTimer01 */
osTimerDef(myTimer01, Callback01);
myTimer01Handle = osTimerCreate(osTimer(myTimer01), osTimerPeriodic, NULL);

/* 定义和创建myTimer02 */
osTimerDef(myTimer02, Callback02);
myTimer02Handle = osTimerCreate(osTimer(myTimer02), osTimerOnce, NULL);

/* 创建互斥量 */
/* 定义和创建myMutex01 */
osMutexDef(myMutex01);
myMutex01Handle = osMutexCreate(osMutex(myMutex01));

/* 创建递归互斥量 */
/* 定义和创建myRecursiveMutex01 */
osMutexDef(myRecursiveMutex01);
myRecursiveMutex01Handle = osRecursiveMutexCreate(osMutex(myRecursiveMutex01));
```

FreeRTOS堆用量

FreeRTOS堆用量选项卡显示了当前使用的堆，并将其与在**配置参数**选项卡中设置的TOTAL_HEAP_SIZE参数进行比较。当使用的总堆超过TOTAL_HEAP_SIZE最大阈值时，将以紫红色显示，而且选项卡上将显示相同颜色的叉（请参阅*图 76*）。

图76. FreeRTOS堆用量



4.4.16 设置HAL时基源

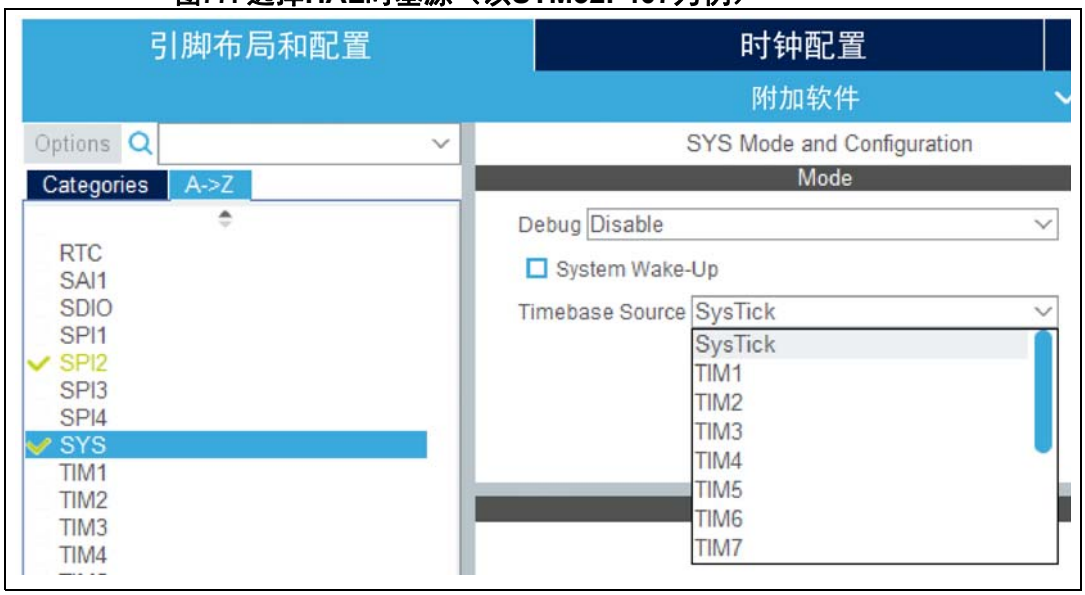
在默认情况下， STM32Cube HAL 围绕唯一的时基源 （即 Arm® Cortex® 系统计时器 （SysTick））构建。

但是，HAL时基相关函数被定义为弱函数，因而，这些函数可重载，以便使用另一个硬件时基源。当应用使用RTOS时，强烈建议这样做，因为中间件可完全控制SysTick配置（节拍和优先级），并且大多数RTOS强制将SysTick优先级设置为最低。

如果应用考虑HAL编程模型，那么使用SysTick仍可以接受，也就是说，在中断服务请求上下文中（没有死锁问题）不执行对HAL时基服务的任何调用。

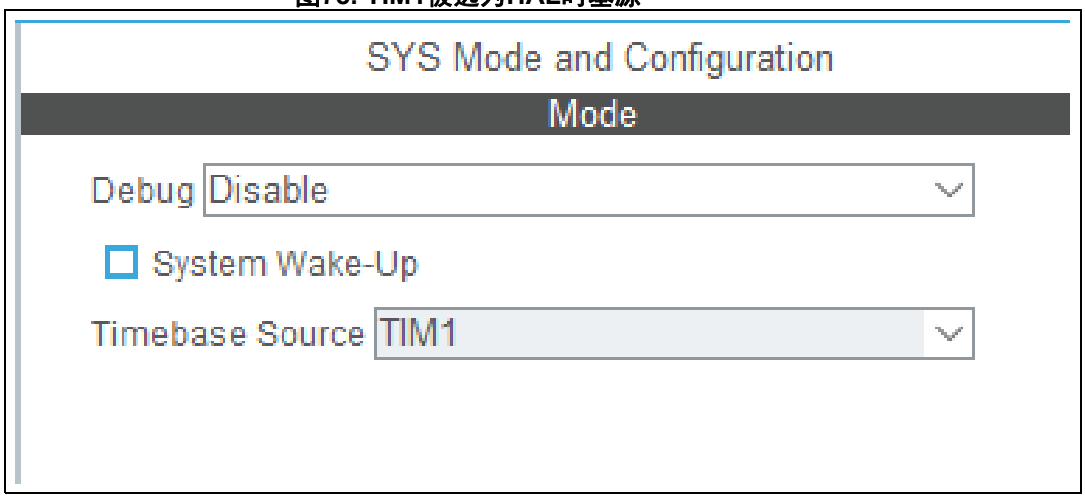
要更改HAL时基源，请转到**“组件列表”**面板中的“系统外设”，然后在可用源中选择一个时钟：SysTick，TIM1，TIM2 ...（请参阅*图 77*）。

图77. 选择HAL时基源（以STM32F407为例）



当用作时基源时，给定的外设是灰色的，不能再被选中（参见*图 78*）。

图78. TIM1被选为HAL时基源

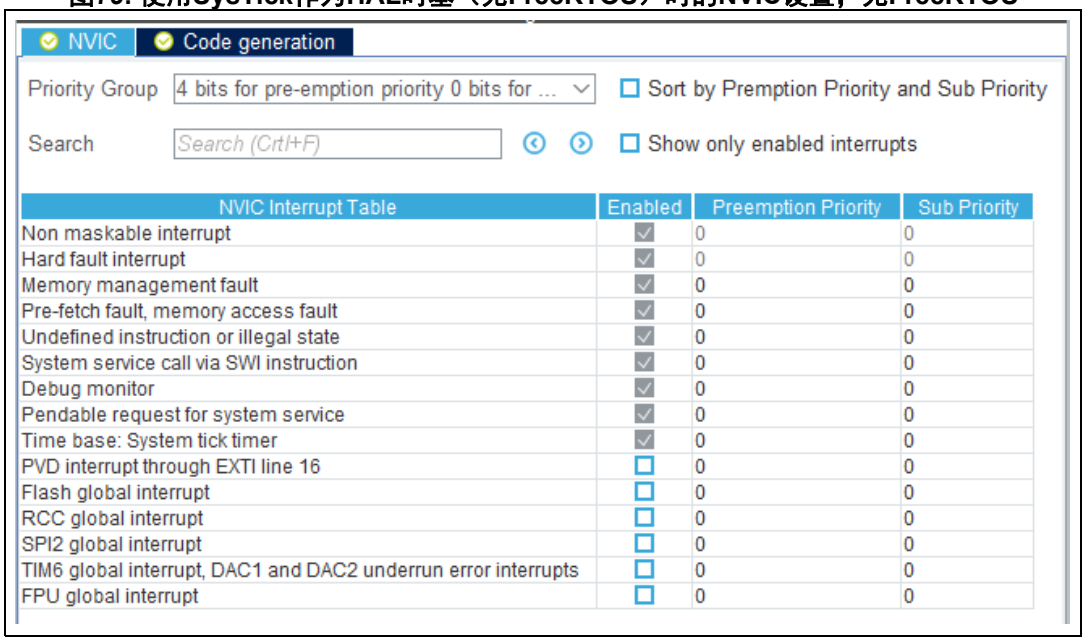


如下例所示，HAL时基源的选择和FreeRTOS的使用会影响生成的代码。

使用SysTick（无FreeRTOS）的配置示例

正如*图 79*中所示，使用SysTick（无FreeRTOS）时，SysTick的优先级设为0（高）。

图79. 使用SysTick作为HAL时基（无FreeRTOS）时的NVIC设置，无FreeRTOS



NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	☑	0	0
Hard fault interrupt	☑	0	0
Memory management fault	☑	0	0
Pre-fetch 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	☑	0	0
PVD interrupt through EXTI line 16	☐	0	0
Flash global interrupt	☐	0	0
RCC global interrupt	☐	0	0
SPI2 global interrupt	☐	0	0
TIM6 global interrupt, DAC1 and DAC2 underrun error interrupts	☐	0	0
FPU global interrupt	☐	0	0

中断优先级（在main.c中）和处理程序代码（在stm32f4xx_it.c中）相应生成：

- main.c文件

```
/* SysTick_IRQn 中断配置 */
  HAL_NVIC_SetPriority(SysTick_IRQn, 0, 0);
```

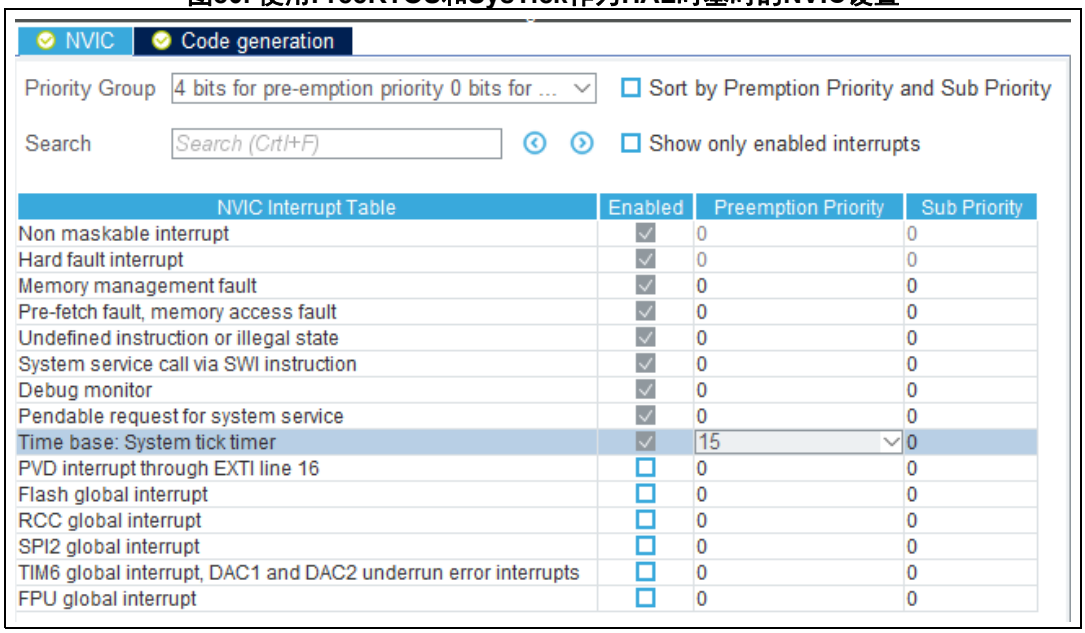
- stm32f4xx_it.c文件

```
/**
* @简要说明此函数处理系统定时器。
*/
void SysTick_Handler(void)
{
  /* 用户代码开始 SysTick_IRQn 0 */
 /* 用户代码结束 SysTick_IRQn 0 */
  HAL_IncTick();
  HAL_SYSTICK_IRQHandler();
  /* 用户代码开始 SysTick_IRQn 1 */

  /* 用户代码结束 SysTick_IRQn 1 */
}
```

使用SysTick和FreeRTOS的配置示例

正如*图 80*中所示，使用SysTick和FreeRTOS时，SysTick的优先级设为15（低）。

图80. 使用FreeRTOS和SysTick作为HAL时基时的NVIC设置



如下面的代码片段所示，SysTick中断处理程序被更新为使用CMSIS-os osSystickHandler函数。

- main.c文件

```
/* SysTick_IRQn 中断配置 */
HAL_NVIC_SetPriority(SysTick_IRQn, 15, 0);
```

- stm32f4xx_it.c文件

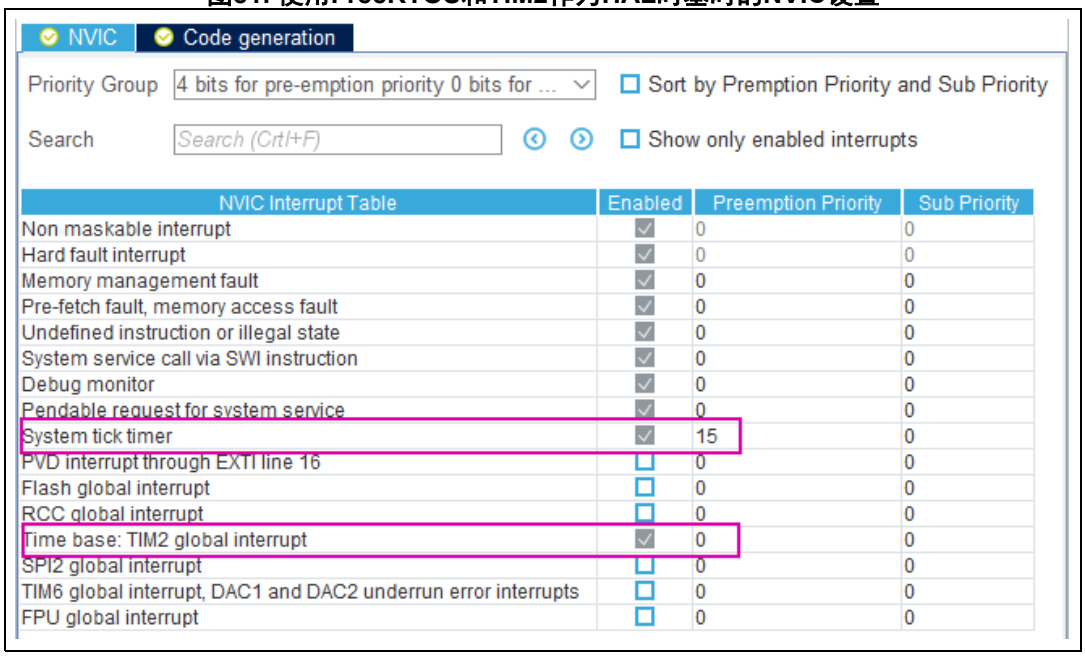
```
/**
* @简要说明此函数处理系统定时器。
*/
void SysTick_Handler(void)
{
 /* 用户代码开始 SysTick_IRQn 0 */

 /* 用户代码结束 SysTick_IRQn 0 */
 HAL_IncTick();
 osSystickHandler();
 /* 用户代码开始 SysTick_IRQn 1 */

 /* 用户代码结束 SysTick_IRQn 1 */
}
```

使用TIM2作为HAL时基源的配置示例

当TIM2用作HAL时基源时，生成一个新的stm32f4xx_hal_timebase_TIM.c文件以过载HAL时基相关函数，包括将TIM2配置为HAL时基源的*HAL_InitTick*函数。

TIM2时基中断的优先级设为0（高）。如果使用FreeRTOS，SysTick的优先级设为15（低），否则设为0（高）。

图81. 使用FreeRTOS和TIM2作为HAL时基时的NVIC设置



NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	☑	0	0
Hard fault interrupt	☑	0	0
Memory management fault	☑	0	0
Pre-fetch 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
System tick timer	☑	15	0
PVD interrupt through EXTI line 16	☐	0	0
Flash global interrupt	☐	0	0
RCC global interrupt	☐	0	0
Time base: TIM2 global interrupt	☑	0	0
SPI2 global interrupt	☐	0	0
TIM6 global interrupt, DAC1 and DAC2 underrun error interrupts	☐	0	0
FPU global interrupt	☐	0	0

相应地生成stm32f4xx_it.c文件：

- 使用FreeRTOS时，SysTick_Handler调用osSystickHandler，否则将调用HAL_SYSTICK_IRQHandler。
- 生成TIM2_IRQHandler以处理TIM2全局中断。

4.5 STM32MP1系列的“引脚布局和配置”视图

对于STM32MP1系列，用户和通过**“引脚布局和配置”**视图进行以下操作：

- 将组件分配给一个或多个运行内核
- 将外设配置为启动设备
- 选择要由启动加载程序管理的外设
- 将GPIO分配给一个运行内核（请参阅*图 83*）。

这些功能在“组件树”面板的两个不同面板中提供（请参阅*图 82*），

1. 列出了所有支持的外设和中间件（必须启用“显示内核”选项）
2. 从每个组件模式面板中，单击组件名称打开。

图82. STM32MP1启动设备和运行内核

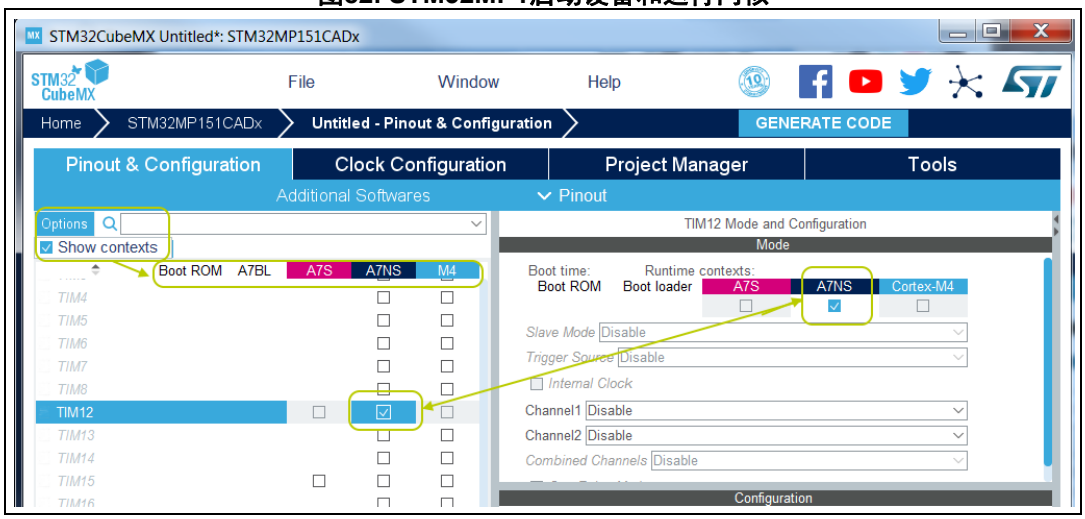
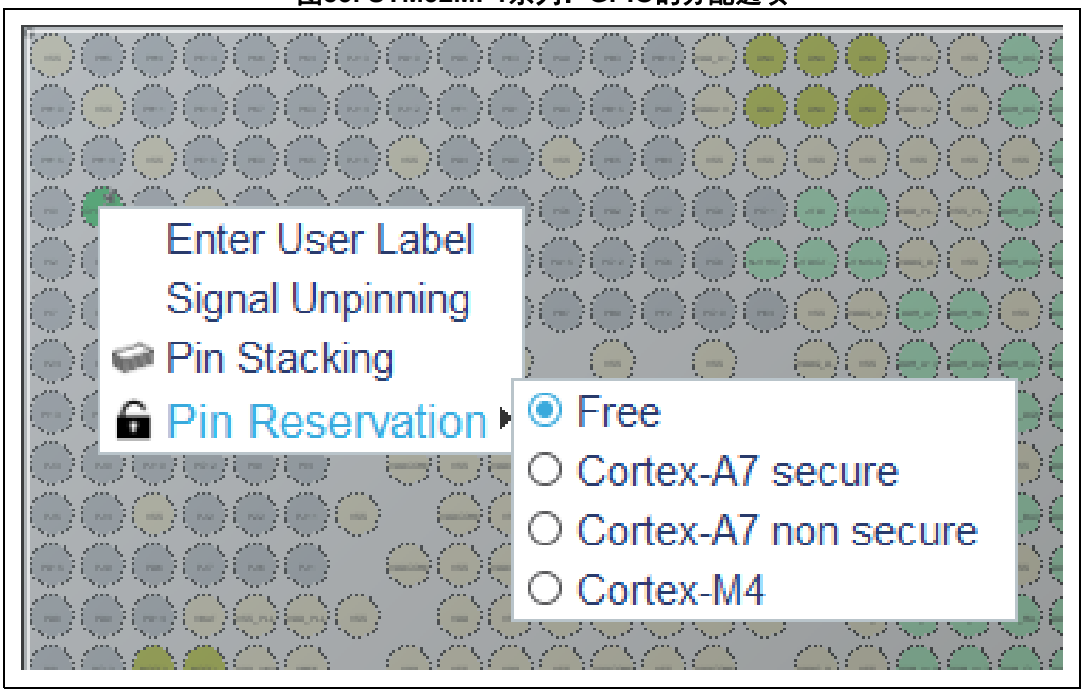


图83. STM32MP1系列：GPIO的分配选项



4.5.1 运行配置

STM32MP1 设备为多核 （Arm® Cortex®-A7 双核和 Cortex-®M4）多固件设备，每个固件都在其中一个内核上执行。固件和内核之间的关联定义了固件执行其代码的运行内核。

有三个运行内核可用：

1. 运行Linux内核的Cortex-A7 Non Secure
2. 运行SP_min的Cortex-A7 Secure
3. 运行STM32Cube固件的Cortex-M4。

将组件分配给运行内核表示指定了将在运行哪一个内核控制组件。对Cortex-A7内核的分配反映在设备树代码生成中，而对Cortex-M4内核的分配反映在基于STM32Cube的C代码生成中（有关更多详细信息，请参见代码生成部分）。

在内核专用列中完成对内核的组件分配。

4.5.2 启动阶段配置

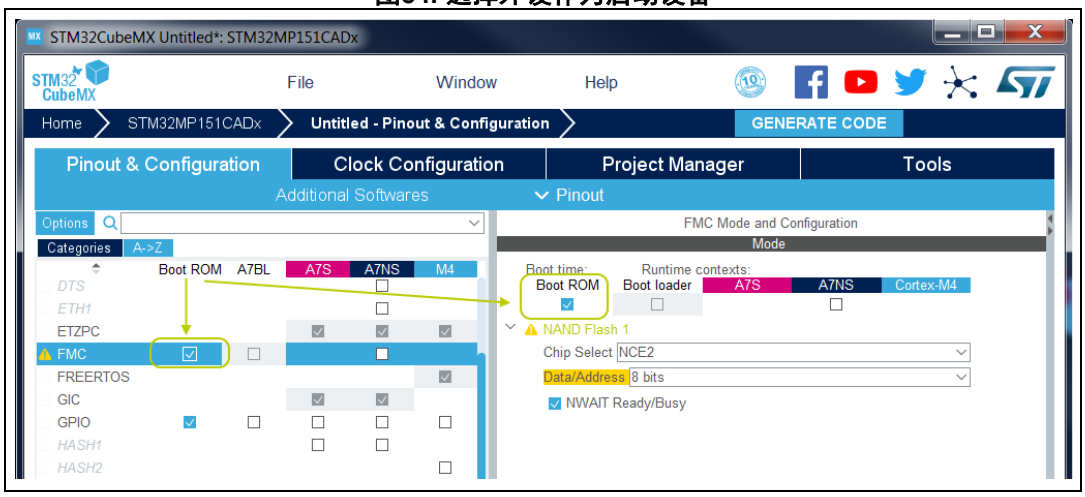
启动ROM外设选择

微处理器需要多个执行阶段进行启动和运行。

首先要执行的是ROM中嵌入的二进制代码。它使用默认配置来初始化时钟树以及启动检测中涉及的所有外设。

可以选择由启动ROM程序管理的外设作为启动器件。该选择在“启动ROM”列中完成（请参阅*图 84*）。

图84. 选择外设作为启动设备



将外设设置为启动设备时，它会加设特定的引脚布局：某些信号必须专门映射到启动ROM可见的引脚上，而且启动ROM程序仅考虑这些信号/引脚。

当设置了ROM可启动外设的功能模式时，链接到该模式的引脚布局与运行时的引脚布局相同，启动ROM代码加设在特定引脚上的信号除外。

在启动步骤期间（在执行启动ROM代码期间），外设仅通过可启动信号和引脚的子集运行。启动后，在运行期间，由外设运行所选功能模式所需的所有信号。

启动加载程序（A7BL）外设选择

板启动时，启动执行固件的每个Cortex-A7运行时（安全和非安全）（SP_min用于Cortex-A7 Secure，Linux内核用于Cortex-A7 Non Secure），然后执行早期启动阶段，即在DDR中U-Boot重定位之前。

启动加载程序（A7BL）列用于定义在该启动加载程序阶段可以管理的那些设备。

该分配反映在生成的不同设备树中（更多详细信息，请参见代码生成部分）。

4.6 STM32H7双核产品系列的“引脚布局和配置”视图

某些STM32H7产品系列具有Arm Cortex-M7内核、ARM Cortex-M4内核和三个电源域。

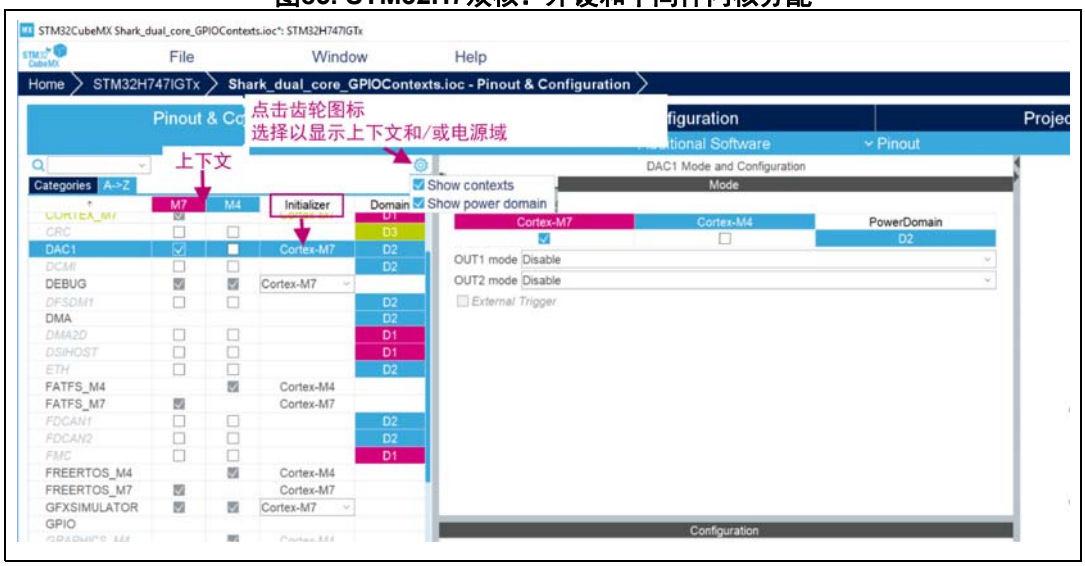
对于此类产品，用户可通过**“引脚布局和配置”**视图进行以下操作：

- 对于每种外设和中间件：尽可能将其分配给一个内核或同时分配给两个内核。如果选择了两个内核，则分配“初始化程序”内核，以指示应在哪个内核上调用外设或中间件初始化功能。
- 对于每个外设：查看其所属的电源域。
- 对于GPIO：将其分配给内核，或将其留给其他可能需要它的组件。在后一种情况下，GPIO初始化在保留它的组件的相同内核上执行（相应地生成代码）。

对于外设和中间件，这些功能在“组件树”面板中两个不同的面板中提供

1. 列出了所有支持的外设和中间件（单击齿轮图标以启用“显示内核”选项），请参阅 *图 85*
2. 从每个组件模式面板中，单击组件名称打开。

图85. STM32H7双核：外设和中间件内核分配



对于GPIO（请参阅*图 86*），可直接或稍后通过**“引脚布局”**视图进行分配，并通过在中间件的平台设置面板中进行选择来自动进行分配。

图86. STM32H7双核：GPIO内核分配

4.7 在“引脚布局和配置”视图中启用安全性（仅适于STM32L5系列）

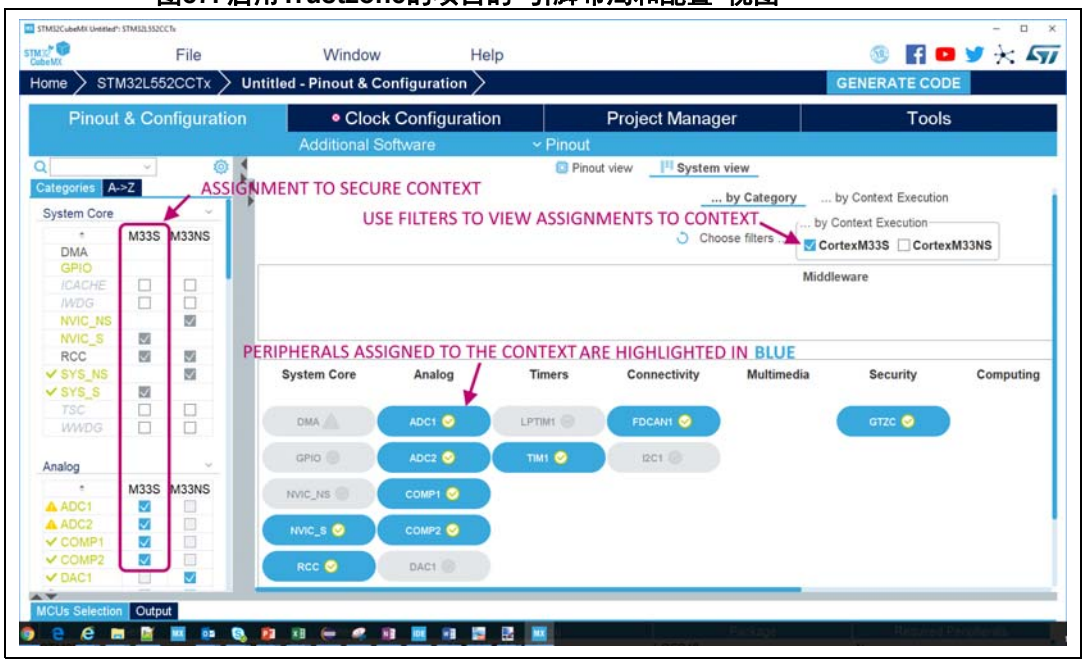
STM32L5系列MCU利用了Arm Cortex-M33处理器的安全特性，并将其面向Armv8-M架构的TrustZone与意法半导体安全实现方案相结合。

支持STM32L5 MCU

- 两级特权
  - 非特权：软件对系统资源的访问受到限制
  - 特权：软件可以完全访问系统资源，但受到安全限制
- 两种安全状态，即安全状态和非安全状态：在FLASH_OPTR寄存器中设置TZEN选项位后，TrustZone安全性被激活。安全状态与模式和特权正交，因此，每个安全状态都支持在两种模式下以两种特权级别执行。

在STM32CubeMX中，激活TrustZone的选择是在项目创建时进行的（请参阅*第 4.2 节：新项目窗口*）。启用TrustZone时，将在安全内核（M33S）和非安全内核（M33NS）之间进行拆分，并使用更多与安全性相关的配置选项，相应地调整STM32CubeMX的“引脚布局和配置”视图（请参阅*图 87*）。

图87. 启用Trustzone的项目的“引脚布局和配置”视图



4.7.1 外设、GPIO EXTI和DMA请求的特权访问

STM32CubeMX独立于TrustZone进行特权访问：

- 对于每个外设：在GTZC配置面板中（请参阅*第 4.7.5 节*），如*图 88*中所示
- 对于每个GPIO EXTI：在GPIO配置面板中，如*图 89*中所示
- 对于每个DMA通道：在DMA配置面板中（请参阅*第 4.7.4 节*），如*图 90*中所示。

注： *当TrustZone处于激活状态时，所有RCC寄存器均可置于特权模式或均不可置于特权模式。在STM32CubeMX中，这是通过选择RCC模式面板中的“仅特权属性”复选框来完成的（请参阅图 91）。在特权模式下，所有RCC寄存器配置都通过PWR_CR_PRIVEN位保留给特权应用程序，该位在激活Trustzone时得到保护。*

图88. 设置外设特权

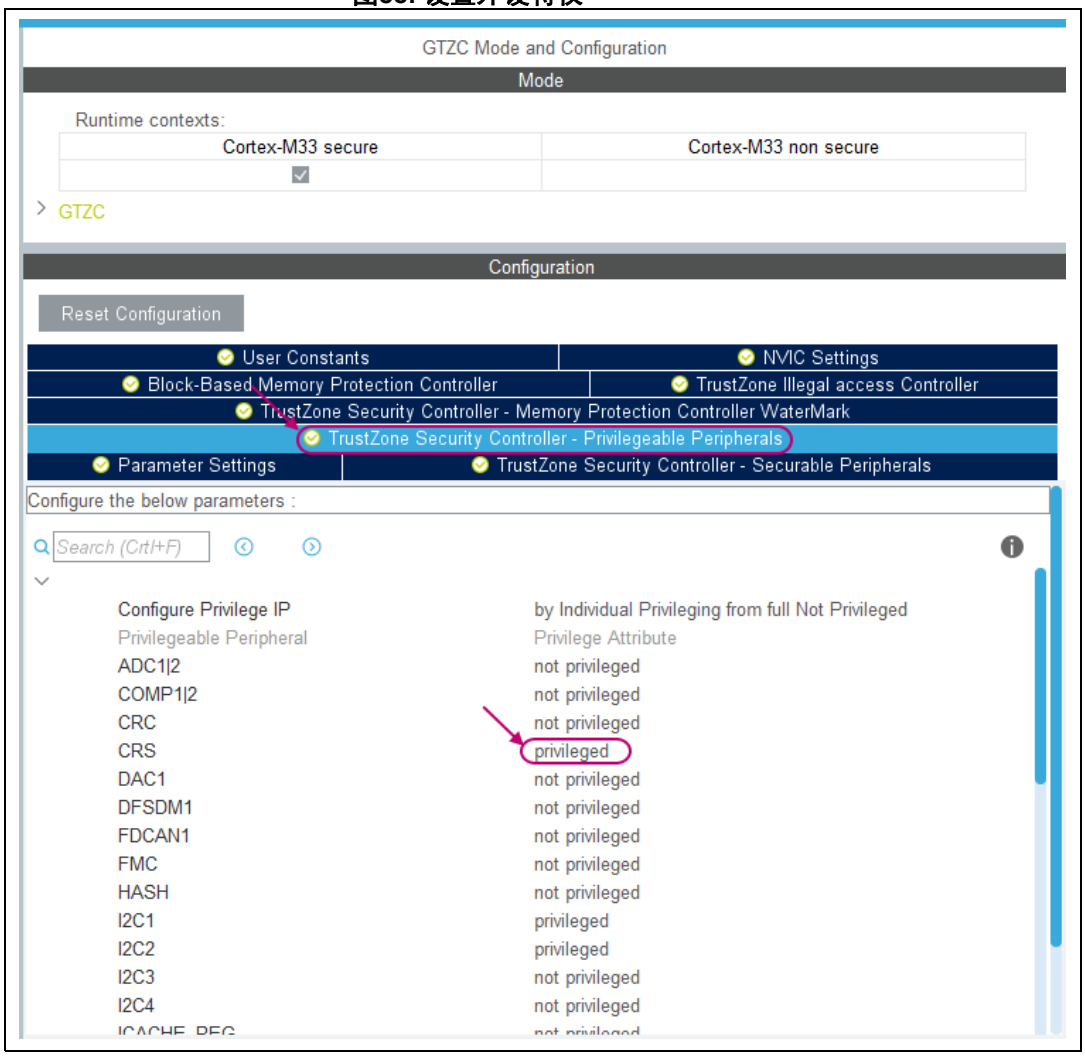

GTZC Mode and Configuration
Mode
Runtime contexts:
Cortex-M33 secure
Cortex-M33 non secure
GTZC
Configuration
Reset Configuration
User Constants
NVIC Settings
Block-Based Memory Protection Controller
TrustZone Illegal access Controller
TrustZone Security Controller - Memory Protection Controller WaterMark
TrustZone Security Controller - Privilegeable Peripherals
Parameter Settings
TrustZone Security Controller - Securable Peripherals
Configure the below parameters :
Search (Crtl+F)
Configure Privilege IP
by Individual Privileging from full Not Privileged
Privilegeable Peripheral
Privilege Attribute
ADC1|2
not privileged
COMP1|2
not privileged
CRC
not privileged
CRS
privileged
DAC1
not privileged
DFSDM1
not privileged
FDCAN1
not privileged
FMC
not privileged
HASH
not privileged
I2C1
privileged
I2C2
privileged
I2C3
not privileged
I2C4
not privileged

图89. 设置GPIO EXTI的特权

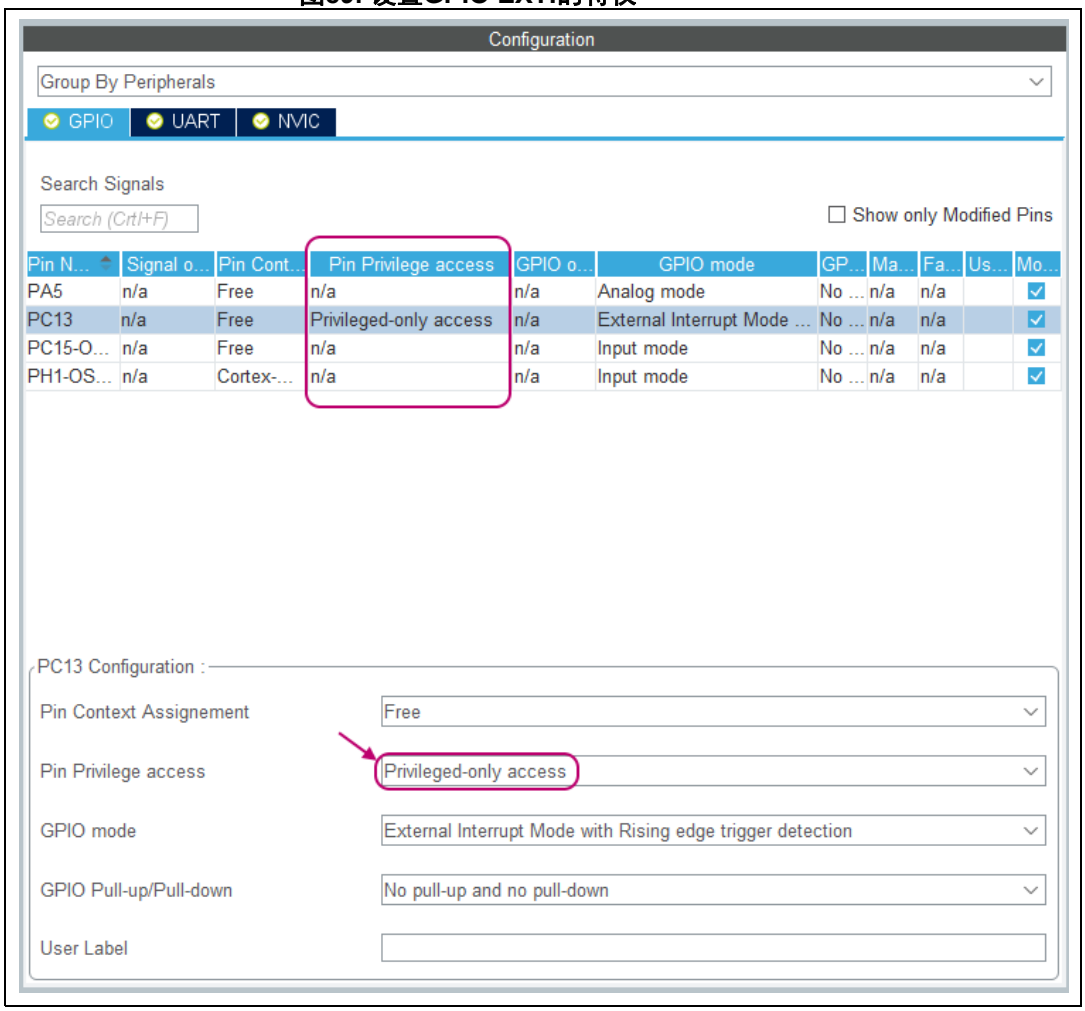


图90. 配置DMA请求的安全性和权限

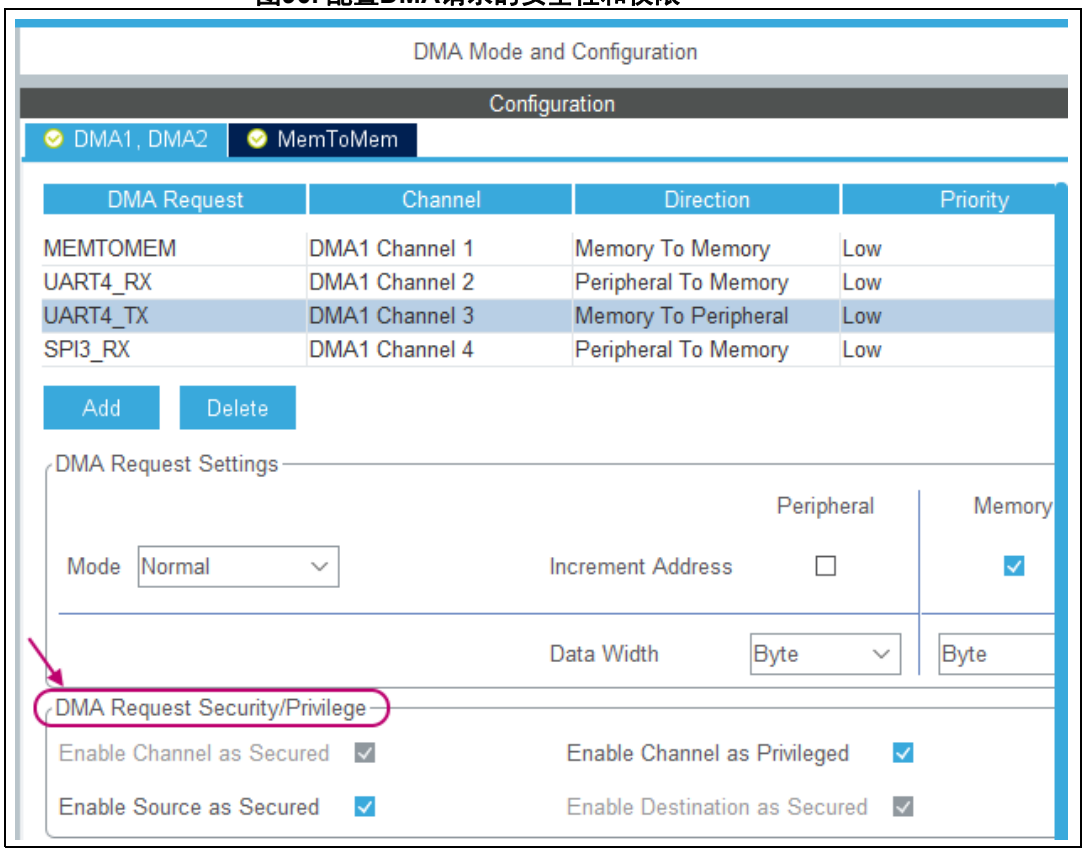
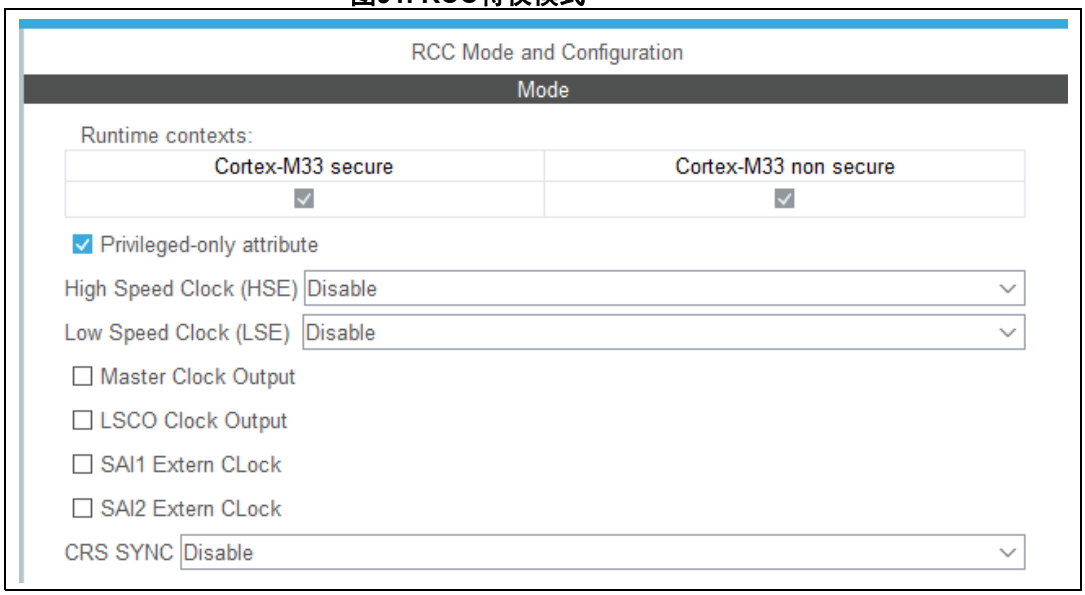


图91. RCC特权模式



4.7.2 GPIO/外设/中间件的安全/非安全内核分配

STM32CubeMX允许用户

- 将每个外设和中间件分配给其中一个内核。
- 将GPIO输入或输出分配给其中一个内核或将其留给可能需要它的其他组件使用。在后一种情况下，GPIO分配与保留它的组件处于相同的内核中。在默认情况下，所有IO都处于安全状态。

分配在不同的面板中完成：

- 仅对于外设和中间件：从启用“显示内核”选项（单击齿轮图标）的组件树面板中或从模式面板中。
- 仅对于外设：从GTZC配置面板中（仅适于外设）。
- 仅对于GPIO：从配置面板中或从“引脚布局”视图中，通过右键单击GPIO引脚，然后选择“保留引脚”。
- 对于DMA请求：从DMA配置面板中。

注： *可以通过“时钟配置”视图来保护RCC资源（请参阅第 4.8.2 节）。*

注： *对于需要外设的中间件，只能将中间件分配给已经分配了外设的内核。*

4.7.3 外设中断的NVIC和内核分配

启用TrustZone时，中断控制器分为非安全内核的NVIC_NS和安全内核的NVIC_S。还可以使用两个SysTick实例，每个实例用于一个内核：二者分别在SYS_NS和SYS_S下可见。

在默认情况下，所有中断都处于安全状态。

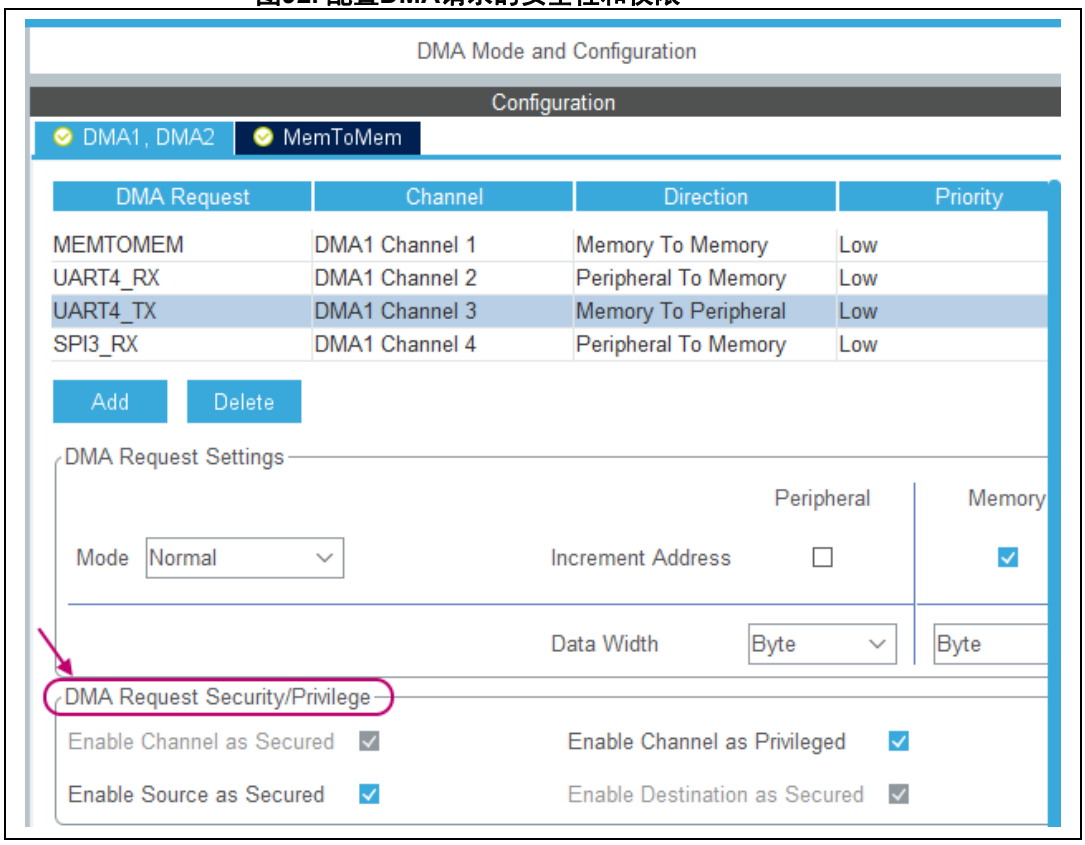
外设中断被自动分配给与内核相关的中断控制器：

- 对于分配给非安全内核的外设，在NVIC_NS上启用中断。
- 对于分配给安全内核的外设，在NVIC_S上启用中断。

4.7.4 DMA（内核分配和特权访问设置）

STM32CubeMX允许用户将DMA通道设置为特权通道，并且在某些情况下，可以保护DMA通道、源和目标站，请参阅图 92。

图92. 配置DMA请求的安全性和权限



在默认情况下，将DMA通道设置为非特权通道。将其设置为特权通道的选择始终可用。

保护DMA通道、源和目标站的选择取决于请求的特征。

有以下四种情况：

- 该请求可以是内存间的传输请求，也可以是DMA生成器请求：在默认情况下，通道处于非安全状态，但可以处于安全状态。仅当通道处于安全状态时，才能使源和目标站受保护。
- 该请求针对分配给非安全内核的外设：通道、源和目标站无法受保护（禁用复选框），因此将其强制进入非安全内核。
- 该请求是针对分配给安全内核的外设的外设到内存的请求：通道和源自动受保护（启用复选框，无法将其禁用），同时可以选择是否保护目标。
- 该请求是针对分配给安全内核的外设的内存到外设的请求：通道和目标站自动受保护（启用复选框，无法将其禁用），同时可以选择是否保护源。

4.7.5 GTZC

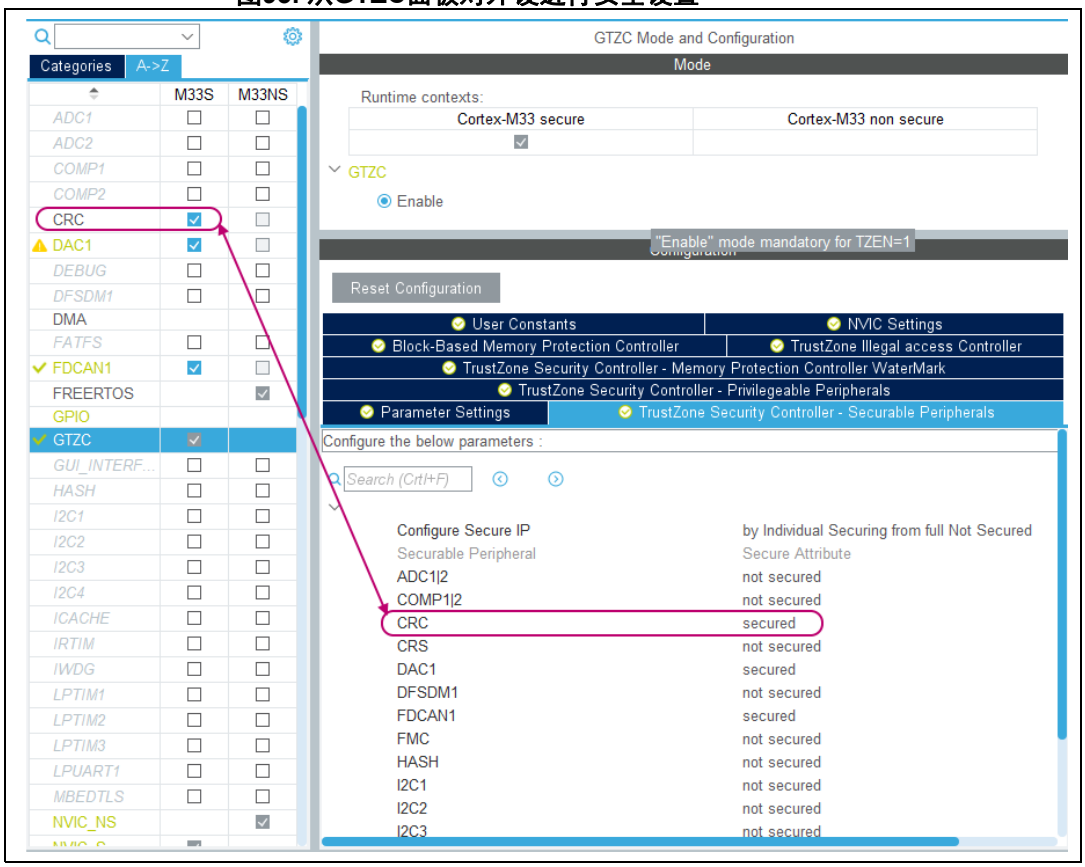
为了配置TrustZone系统安全性，STM32L5系列配备了Global TrustZone安全控制器（GTZC）。请参考参考手册RM0438获取更多详细信息。

在STM32CubeMX中，对于激活了TrustZone的项目，在默认情况下启用GTZC且无法将其禁用。对于没有激活Trustzone的项目，可以启用GTZC，并且仅可以设置特权。

GTZC由三个块组成，这三个块可以使用GTZC配置面板中的专用选项卡通过CubeMX进行配置：

- TZSC（TrustZone安全控制器）
  - 定义了安全的和/或具有特权的外设，并控制水印内存外设控制器（MPCWM）的非安全区域大小。TZSC模块通过与RCC和I/O逻辑共享，将每个可进行安全设置的外设的安全状态通知某些外设（如RCC或GPIO）。
  - 在TrustZone安全控制器的“可设置特权的外设”选项卡中设置特权。
  - 安全状态在TrustZone安全控制器的“安全外设”选项卡中设置（它们与在树形视图或“模式”面板中完成的内核分配（M33S或M33NS）相匹配）。
  - MPCWM配置通过TrustZone安全控制器的“内存保护控制器水印”选项卡完成。
- MPCBB（基于块的内存保护控制器）
  - 控制相关SRAM的所有块（256字节页）的安全状态。这是通过基于块的“内存保护控制器”选项卡进行配置的。
- TZIC（TrustZone非法访问控制器）
  - 收集系统中的所有非法访问事件，然后生成针对NVIC的安全中断。这是通过TrustZone非法访问控制器选项卡进行配置的。

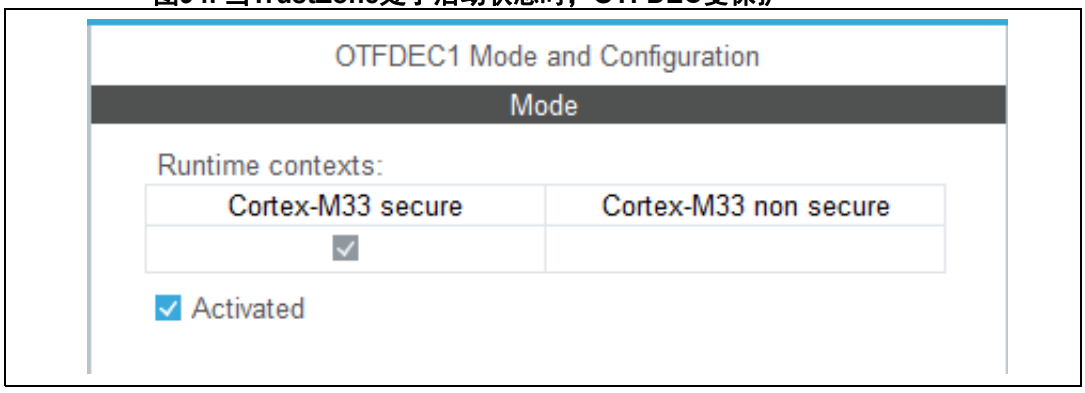
图93. 从GTZC面板对外设进行安全设置



4.7.6 OTFDEC

用户可通过即时解密引擎（OTFDEC）基于所读取的请求地址信息对即时AHB流量进行解密。在产品中启用安全性后，只能通过安全主机对OTFDEC进行编程。

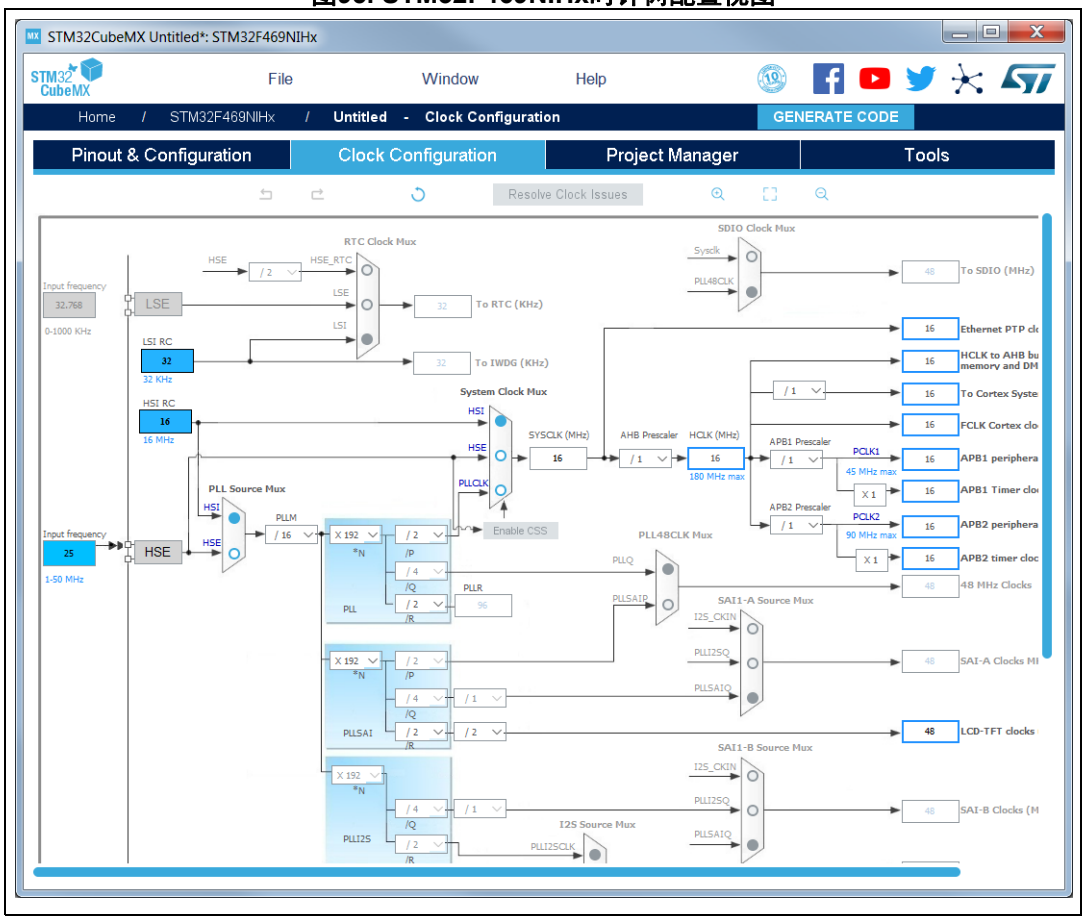
图94. 当TrustZone处于活动状态时，OTFDEC受保护



4.8 “时钟配置”视图

STM32CubeMX**时钟配置**窗口（参见*图 95*）提供了时钟路径、时钟源、分频器和倍频器的原理概览。下拉菜单和按键可用于修改实际的时钟树配置，以满足应用需求。

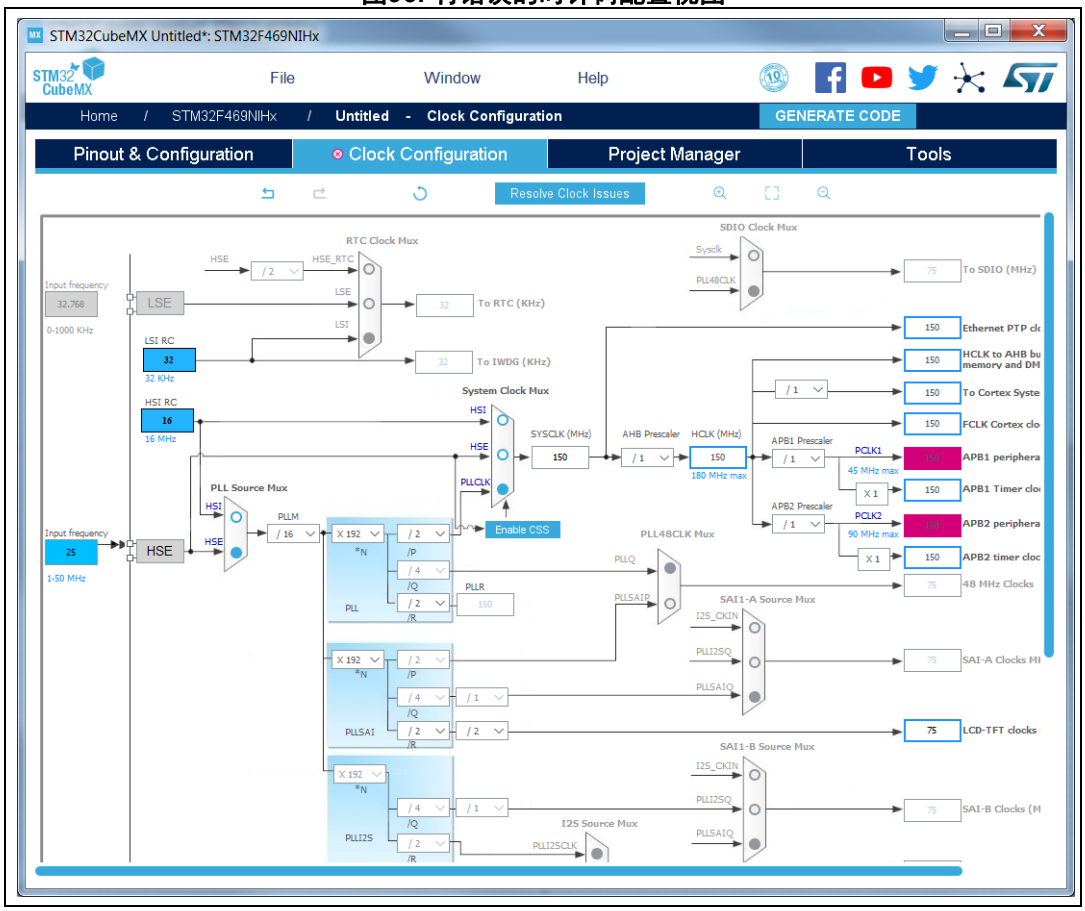
图95. STM32F469NIHx时钟树配置视图



实际时钟速度显示并处于活动状态。使用时钟信号以蓝色突出显示。

如*图 96*中所示，突出显示了超出范围的配置值以标记潜在问题。建议使用解算器功能，以自动解决此类配置问题。

图96. 有错误的时钟树配置视图



支持反向路径：只需在蓝色字段中输入所需的时钟速度，STM32CubeMX会尝试重新配置倍频器和分频器，以提供所需值。然后，可以通过右键点击该字段来锁定生成的时钟值，以防止修改。

STM32CubeMX生成相应的初始化代码：

- 支持相关HAL_RCC结构初始化和函数调用的main.c
- 适用于振荡器频率和V_{DD}值的stm32xxxx_hal_conf.h。

4.8.1 时钟树配置功能

外部时钟源

在使用外部时钟源时，用户必须先通过RCC外设下可用的**引脚排列**视图将其使能。

外设时钟配置选项

对应于时钟外设的其他路径灰显。要激活这些路径，必须在**引脚排列**视图（如USB）中正确配置外设。此视图允许用户：

- **输入CPU时钟（HCLK）、总线或外设时钟的频率值**

  STM32CubeMX尝试推荐达到所需频率的时钟树配置，同时调整预分频器和分频器，并考虑其他外设约束（如USB时钟的最小值）。如果找不到解决方案，STM32CubeMX建议切换到不同的时钟源，或甚至可以得出结论：没有符合所需频率的解决方案。
- **锁定应保留当前值的频率字段**

  右键单击频率字段并选择**锁定**，以便在STM32CubeMX要搜索新的时钟配置解决方案时保留当前分配的值。

  在不需要保留时，用户可以解锁锁定的频率字段。
- **选择将驱动系统时钟的时钟源（SYSCLK）**
  - 适用于用户定义频率的外部振荡器时钟（HSE）。
  - 适用于定义的固定频率的内部振荡器时钟（HSI）。
  - 主PLL时钟
- **选择次级时钟源（适用于产品）**
  - 低速内部（LSI）或外部（LSE）时钟
  - I2S 输入时钟
  - 其他源
- **选择预分频器、分频器和倍频器值**
- **当MCU支持时，在HSE上使能时钟安全系统（CSS）**

  仅当直接或间接通过PLL将HSE时钟用作系统时钟源时，此功能才可用。通过此功能可检测HSE失效，并向软件通知该情况，从而使MCU能够执行救援操作。
- **当MCU支持时，在LSE上使能CSS**

  仅在已使能LSE和LSI并选择LSE或LSI作为RTC或LCD的时钟源时，此功能才可用。
- **使用工具栏重置按钮重置时钟树默认设置**

  此功能可重新加载STM32CubeMX默认时钟树配置。
- **使用工具栏的“撤消/恢复”按键来撤消/恢复用户配置步骤**
- **检测并解决配置问题**

  在生成代码前检测到错误的时钟树配置。以紫红色突出显示错误，**时钟配置**视图标有紫红叉（参见*图 96*）。

  可通过点击**解决时钟问题**按钮手动或自动解决问题，该按钮仅在检测到问题时使能。

  基础解决过程遵循特定顺序：

  a) 将HSE频率设为最大值（可选）。

  b) 依次将HCLK频率与外设频率设为最大值或最小值（可选）。

  c) 更改多路复用器输入（可选）。

d) 最后，通过倍频器/分频器值迭代来解决问题。如果找到解决方案，则从紫红色突出显示中清除时钟树，否则将显示错误消息。

注： *要在时钟树中可用，须在**引脚布局**视图的RCC配置中使能外部时钟、I2S输入时钟和主时钟。该信息也可以通过工具提示提供。*

此工具自动执行以下操作：

- 根据用户选择的时钟源、时钟频率和预分频器/倍频器/分频器值，调节总线频率、定时器、外设和主输出时钟。
- 检查用户设置的有效性。
- 以紫红色突出显示无效设置，并提供工具提示，以指导用户实现有效配置。

根据RCC设置（在RCC**引脚布局和配置**视图中配置）调整**时钟配置**视图，反之亦然：

- 如果在RCC**引脚布局**视图中使能外部和输出时钟，则可在**时钟配置**视图中对其进行配置。
- 如果在RCC配置视图中使能定时器预分频器，则调整定时器时钟倍频器的选择。

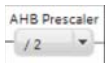
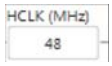
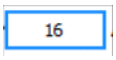
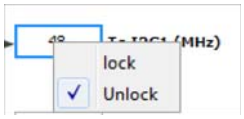
相反，时钟树配置可能会影响配置视图中的某些RCC参数：

- 闪存延迟：自动通过V_{DD}电压、HCLK频率和功率超载状态获取等待状态数。
- 功率调节器电压级别：自动根据HCLK频率得出。
- 自动根据HCLK频率使能功率超载。当使能功率驱动时，AHB和APB域的最大可能频率值会增加。它们在**时钟配置**视图中显示。

在*system_stm32f4xx.c*文件中定义启动时使用的默认最佳系统设置。该文件由STM32CubeMX从STM32CubeF4 MCU包中复制。之后在主要功能中切换到用户定义的时钟设置。

*图 95*提供了STM32F429x MCU的时钟树配置示例，*表 9*介绍了可用于配置每个时钟的小工具。

表9. 时钟配置视图小工具

格式	外设实例的配置状态
	有效时钟源
	模糊或灰显的不可用设置（时钟源、分频器等）
	预分频器、分频器、倍频器选择的灰色下拉列表。
	倍频器选择
	用户定义的频率值
	自动得出的频率值
	用户可修改的频率字段
	右键单击蓝色边框矩形，以锁定/解锁频率字段。将其锁定以在时钟树配置更新期间保留频率值。

4.8.2 保护时钟资源（仅适于STM32L5系列）

在激活TrustZone安全性后，RCC可以通过安全配置寄存器来防止对系统时钟资源的非安全访问。

因此，STM32CubeMX允许用户将其配置为安全：

- 具有固定频率的系统时钟源：HIS、LSI和RC48
- 具有可配置频率的系统时钟源：HSE (+CSS)，MSI和LSE（+CSS）
- 两个多路复用器：CLK48时钟多路复用器，系统时钟（+ MCO源）多路复用器
- 其他系统配置：PLLSYS，PLLSAI1，PLLSAI2锁相环和AHB/APB1/APB2总线预分频器

在“时钟配置”视图中，这些安全的资源以钥匙图标突出显示。使用右键单击资源访问的“安全”复选框，可以启用安全性。资源处于安全状态后，将以绿色方块突出显示。

可以锁定可配置资源以防进一步更改配置：这可以通过选择通过右键单击资源访问的“锁定”复选框来完成。

还有一个快捷按键可用于一键锁定/解锁所有可安全设置和可配置的资源。

当外设配置为安全时，其相关的时钟、复位、时钟源和时钟启用也处于安全状态。在STM32CubeMX中，在“引脚布局和配置”视图中将外设配置为安全，其时钟源在“时钟配置”视图中使用绿色方块自动突出显示为安全。

表10. 时钟配置安全设置

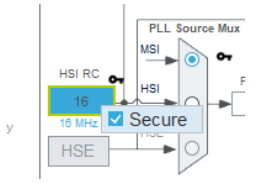
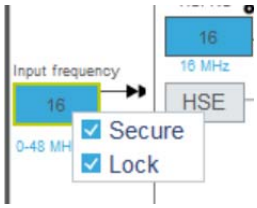
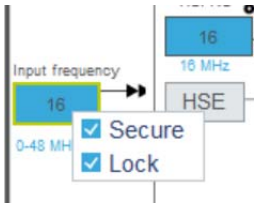
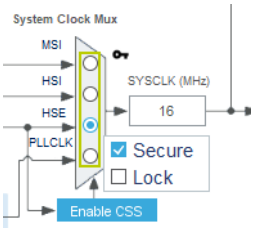
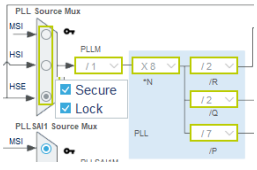
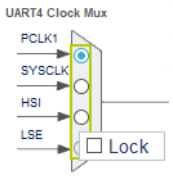
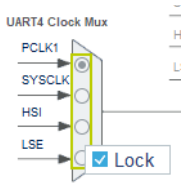
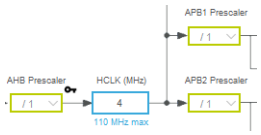
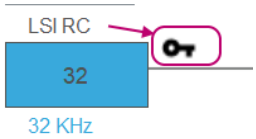
视图	说明
	安全的不可配置系统时钟资源的示例。
	安全且仍处于开放状态以供编辑的系统时钟HSE时钟源的示例：可以更改频率值。
	安全且已锁定编辑的系统时钟HSE时钟源的示例：无法修改频率值。
	安全且未锁定的系统时钟多路复用器的示例：可以更改时钟源。
	安全且已锁定的主PLL多路复用器的示例。时钟源为HSE，无法更改。PLLxxM、PLLxxN、PLLxxP、PLLxxQ和PLLxxR也处于安全状态且已锁定编辑。

表10. 时钟配置安全设置（续）

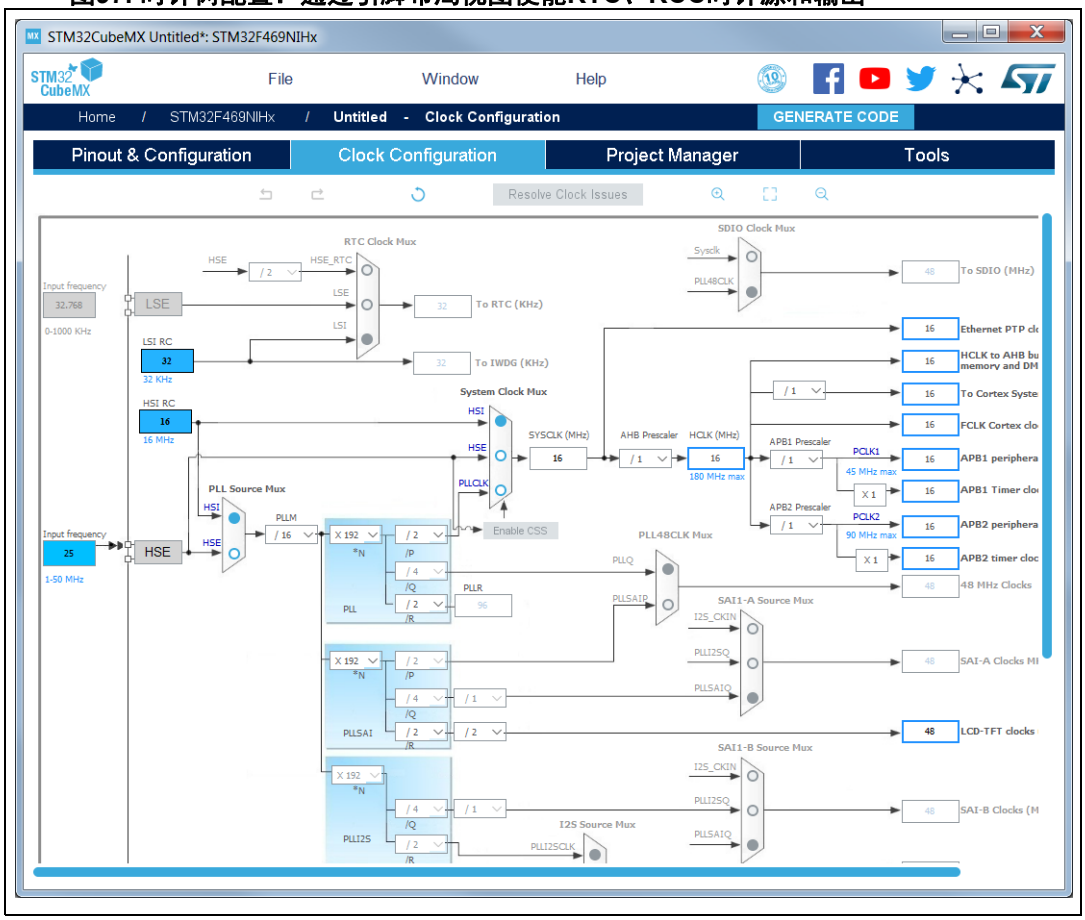
视图	说明
	UART4时钟源多路复用器示例：时钟源是安全的，因为UART4外设在“引脚布局和配置”视图中被配置为安全。将其设置为PCLK1，可以在未选中“锁定”复选框时进行更改。
	UART4时钟源多路复用器示例：时钟源是安全的，因为UART4外设在“引脚布局和配置”视图中被配置为安全。将其设置为PCLK1，但不能在“锁定”开启时进行更改。
	保护和锁定对AHB预分频器的访问的示例。APB1和APB2预分频器也被锁定。
	使用密钥图标突出显示为可保护资源的LSI的示例。
	“全部锁定/解锁”按键（仅对安全资源有效）。

4.8.3 建议

“时钟配置”视图不是唯一的时钟配置条目，还可以配置RCC和RTC外设。

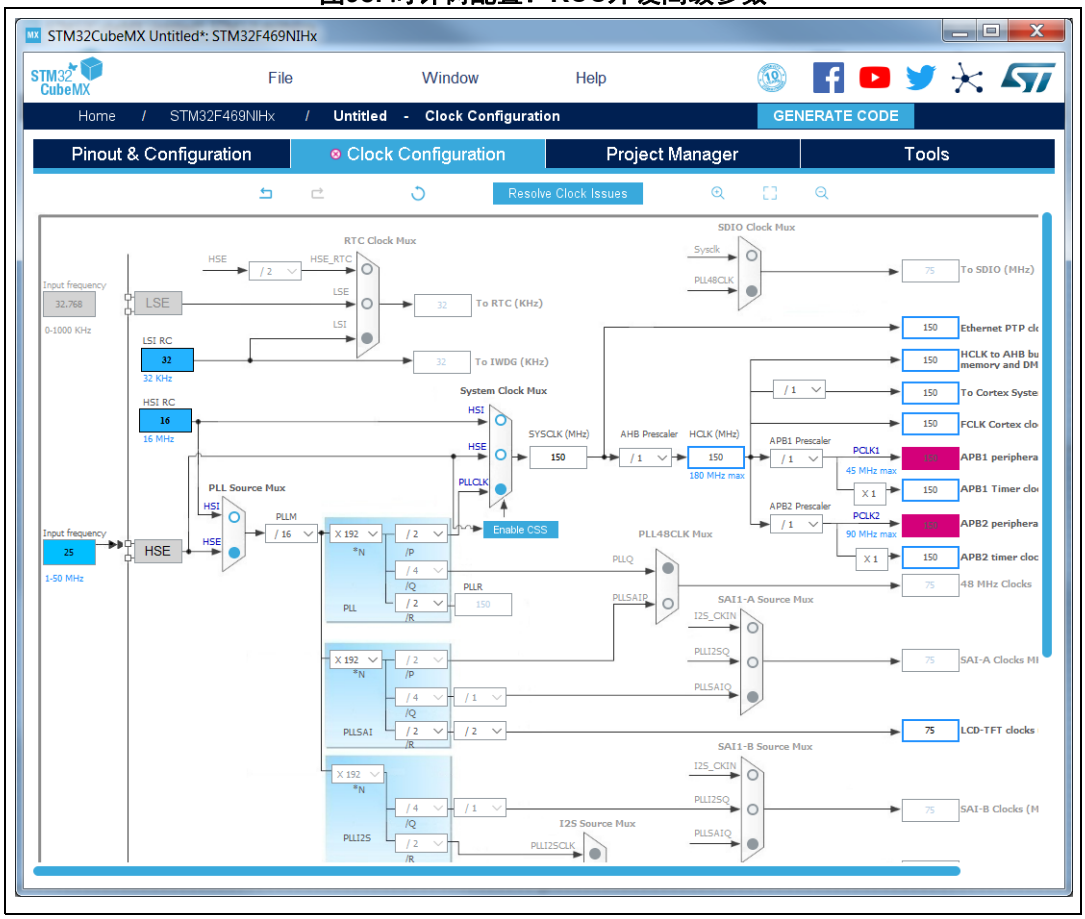
1. 从**“引脚布局和配置”**视图转到RCC模式面板，以便根据需要启用时钟：外部时钟、主输出时钟和音频I2S输入时钟（如果可用）。然后转到RCC配置面板，并根据需要调整默认设置。更改会反映在**“时钟配置”**视图中。定义的设置也可能会更改RCC配置中的设置（请参阅*图 97*）。

图97. 时钟树配置：通过引脚布局视图使能RTC、RCC时钟源和输出



2. 前往**引脚布局和配置**视图中的**RCC配置**。这里为高级配置定义的设置会反映在**时钟配置**视图中。 定义的设置可能会更改RCC配置中的设置。

图98. 时钟树配置：RCC外设高级参数



4.8.4 STM32F43x/42x功率超载功能

STM32F42x/43x MCU可实现功率超载功能，以便在施加足够的V_{DD}电源电压时（如V_{DD} > 2.1 V）允许在最大AHB/APB总线频率（如HCLK为180 MHz）下工作。

*表 11*列出了与功率超载功能有关的不同参数及其在STM32CubeMX用户界面中的可用性。

表11. 电压调节与功率超载和HCLK频率

参数	STM32CubeMX板	值
V_{DD}电压	配置（RCC）	预定义范围内的用户定义。影响功率超载。
功率调节器 电压调节		根据HCLK频率和功率超载自动得出（参见*表 12*）。
功率超载		该值取决于HCLK和V_{DD}值（参见*表 12*）。它只能在V_{DD} ≥ 2.2 V时使能。 当V_{DD} ≥2.2 V时，可自动根据HCLK获得该值，如果有多种选择，也可以由用户配置该值（如HCLK = 130 MHz）
HCLK/AHB时钟 最大频率值	时钟配置	用蓝色显示，以指示可能的最大值。例如：当无法激活功率超载时（V_{DD} ≤ 2.1 V），HCLK的最大值为168 MHz，否则为180 MHz。
APB1/APB2时钟 最大频率值		用蓝色显示，以指示可能的最大值。

*表 12*给出了功率超载模式与HCLK频率之间的关系。

表12. 功率超载模式与HCLK频率之间的关系

HCLK频率范围： 要使能功率超载（POD），需要V_{DD} > 2.1 V	相应的电压调节 和功率超载(POD)
≤120 MHz	级别 3 禁止POD
120到144 MHz	级别 2 可禁用或使能POD
144到168 MHz	禁用POD时为级别1 使能POD时为级别2
168到180 MHz	必须使能POD 级别1（否则不支持频率范围）

4.8.5 时钟树词汇表

表13. 词汇表

缩略语	定义
HSI	高速内部振荡器：复位后使能，精度低于HSE
HSE	高速外部振荡器：需要外部时钟电路
PLL	锁相环：用于倍增上述时钟源
LSI	低速内部时钟：通常用于看门狗定时器的低功耗时钟
LSE	低速外部时钟：由外部时钟供电
SYSCLK	系统时钟
HCLK	内部AHB时钟频率
FCLK	Cortex自由运行时钟

表13. 词汇表（续）

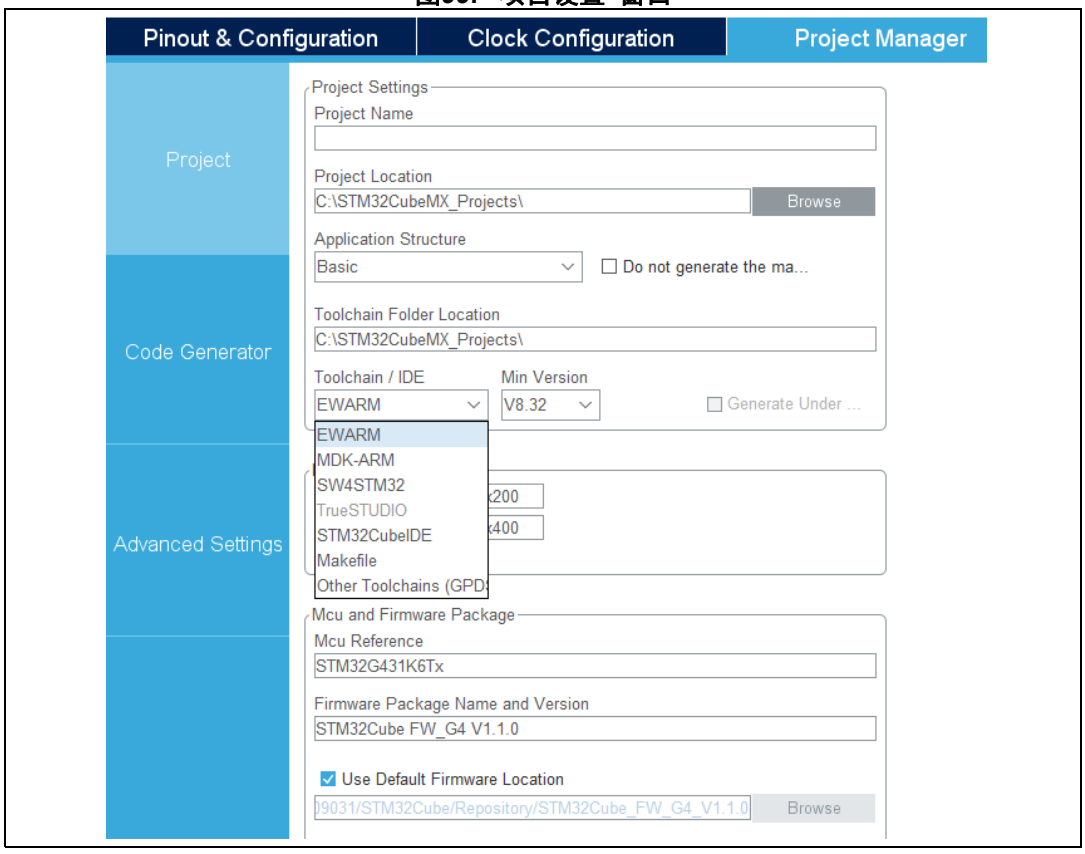
缩略语	定义
AHB	高级高性能总线
APB1	低速高级外设总线
APB2	高速高级外设总线

4.9 “项目管理器”视图

该视图（请参阅*图 99*）具有三个选项卡：

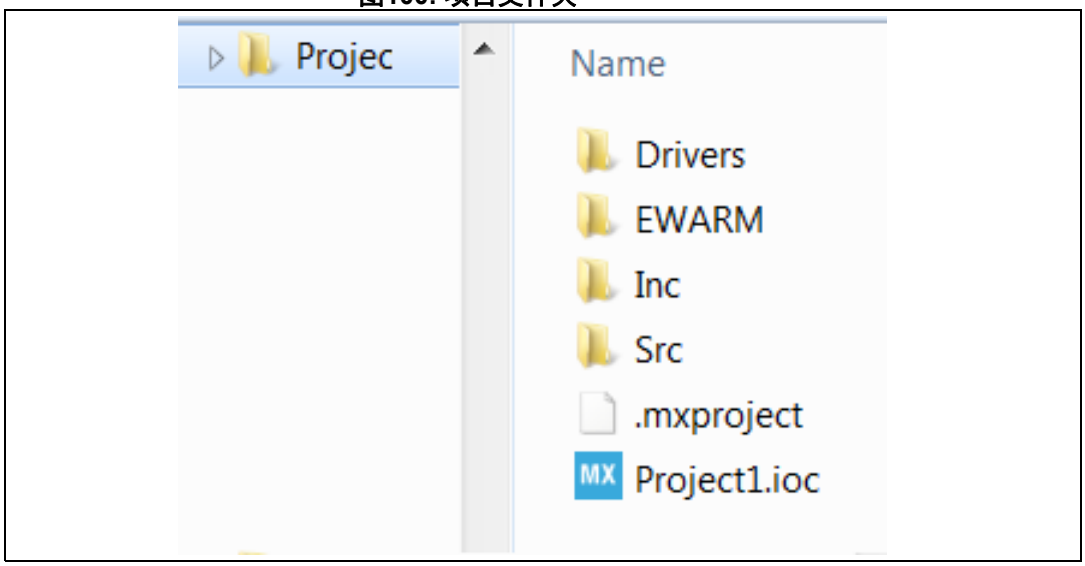
- 常规项目设置：指定项目名称、位置、工具链和固件版本。
- 代码生成：允许设置代码生成选项，例如外设初始化代码的位置、库复制/链接选项，以及为自定义代码选择模板。
- 高级设置：用于排序STM32CubeMX初始化功能调用。

图99. “项目设置”窗口



代码在*图 100*中所示的项目文件夹树中生成。

图100. 项目文件夹



注： *保存项目后，某些项目设置选项将变为只读。要修改这些选项，必须使用“**文件**”>“**将项目另存为**”菜单将项目另存为新项目。*

4.9.1 “项目”选项卡

“项目设置”窗口的**“项目”**选项卡允许配置以下选项（参见*图 99*）：

- 项目设置：
  - 项目名称：用于创建项目文件夹的名称以及给定项目位置的.ioc文件名
  - 项目位置：项目文件夹的存储目录。
  - 应用程序结构：在“基本”和“高级”选项之间进行选择。

    基本结构：建议用于使用一个中间件或不使用中间件的项目。这种结构将IDE配置文件夹与源文件放置在同一级别，并按源文件进行组织，并包含子文件夹（请参阅*图 101*）。

    高级结构：建议在项目中使用多个中间件组件时使用。它让中间件应用程序的集成变得更加简单（请参阅*图 102*）
  - 工具链文件夹位置：在默认情况下，它位于项目文件夹中，与.ioc文件位于同一级别。
  - 工具链/IDE：选定的工具链
  - 仅对于STM32MP1系列，OpenSTLinux设置：生成的设备树的位置以及清单版本和当前项目的内容（请参阅*图 103*）。这些信息可以使正确的软件组件版本与用于Cortex®M和Linux的STM32CubeMP1、tf-a、用于Cortex®A的u-boot同步。重要的是要将其考虑在内，特别是要确保一个Cube固件版本在OpenAMP/RPM链接和资源管理API方面与用于Cortex®A的软件组件一致。

在Toolchain/IDE下选择*"Makefile"*将生成一个基于gcc的通用Makefile。

选择*其他工具链（GPDSC）*生成一个gpdsc文件。Gpdsc文件提供项目的通用描述，包括构建项目所需的驱动程序和其他文件（如启动文件）的列表和路径。因此，可以将STM32CubeMX项目生成扩展到任何支持gpdsc的工具链，因为该工具链能够通过处理gpdsc文件信息来加载STM32CubeMX生成的C项目。为标准化嵌入项目的描述，gpdsc解决方案基于CMSIS-Pack。

- 面向SW4STM32和Atollic® TrueSTUDIO®工具链的其他项目设置：

  选择可选的**在根文件夹中生成**复选框，以在STM32CubeMX 用户项目根文件夹中生成工具链项目文件，或者取消选择该复选框，以在专用工具链文件夹中生成工具链项目文件。

  在根文件夹下生成STM32CubeMX项目，有助于在使用基于Eclipse的IDE（如SW4STM32和TrueStudio®）时受益于以下Eclipse特性：

  – 导入项目时可选择将项目复制到Eclipse工作区中。
  – 使用Eclipse工作区中的GIT或SVN等源代码控制系统。

  选择将项目复制到工作区中可防止在Eclipse中完成的更改与在STM32CubeMX中完成的更改之间发生任何进一步的同步，因为项目将有两个不同的副本。
- 链接器设置：为应用分配的最小堆和栈的大小值。对于堆大小和栈大小，建议的默认值分别为0x200和0x400。当应用使用中间件栈时，这些值可能需要增加。
- 当有多个版本可用时的固件包选择（当后续版本实现相同的API并支持相同的MCU时，就会出现这种情况）。默认情况下，使用最新的可用版本。
- 固件位置选择选项

  默认位置是在**帮助>更新程序设置**菜单下指定的位置。

  取消选择**使用默认固件位置** 复选框，则用户可为项目将要使用的固件指定不同的路径（参见*图 104*）。

图101. 选择基本应用程序结构

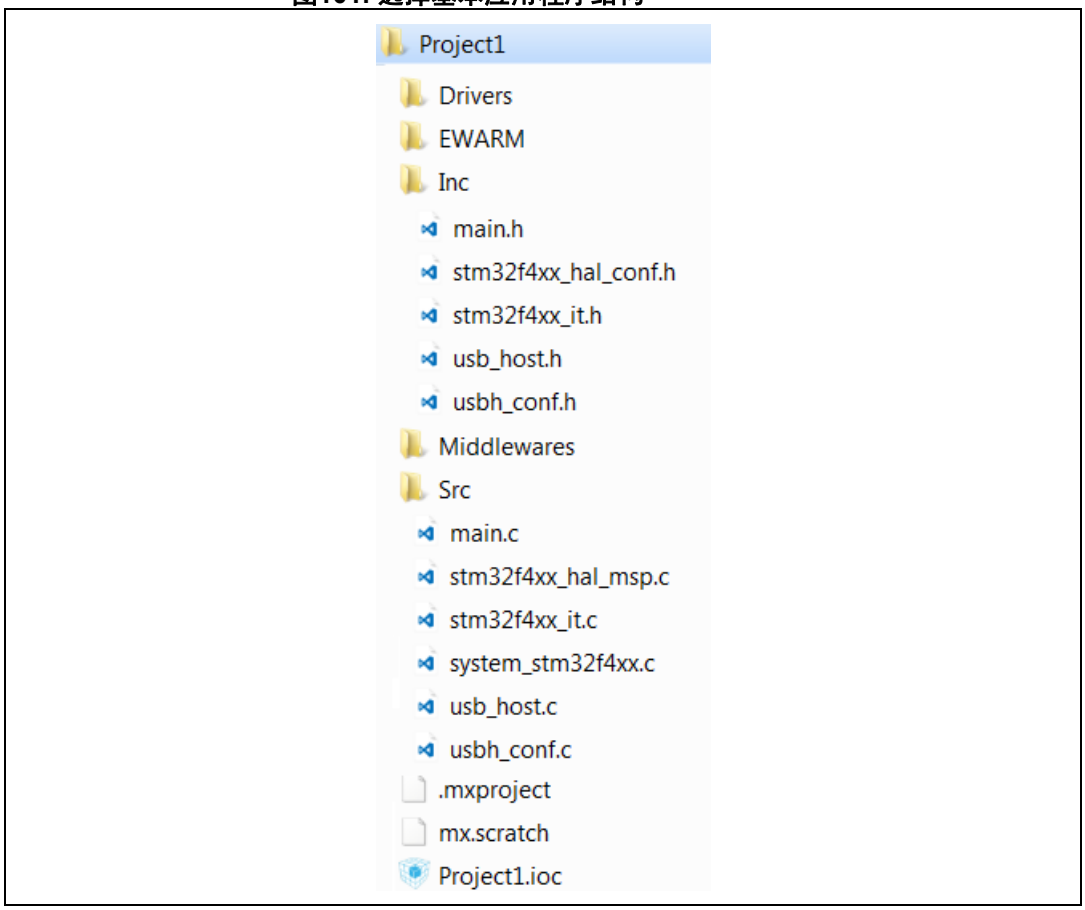


图102. 选择高级应用程序结构

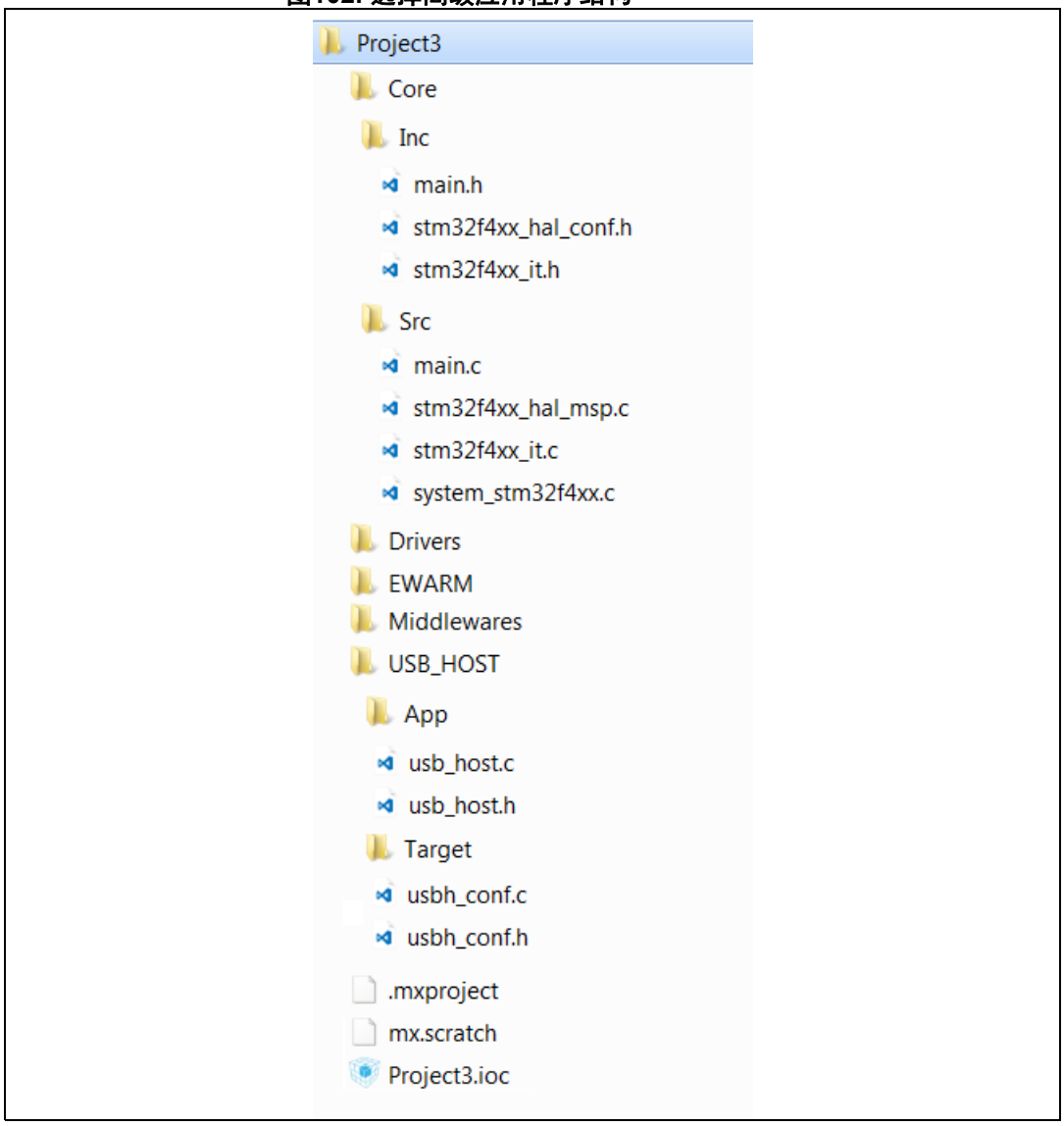


图103. OpenSTLinux设置（仅适于STM32MP1系列）

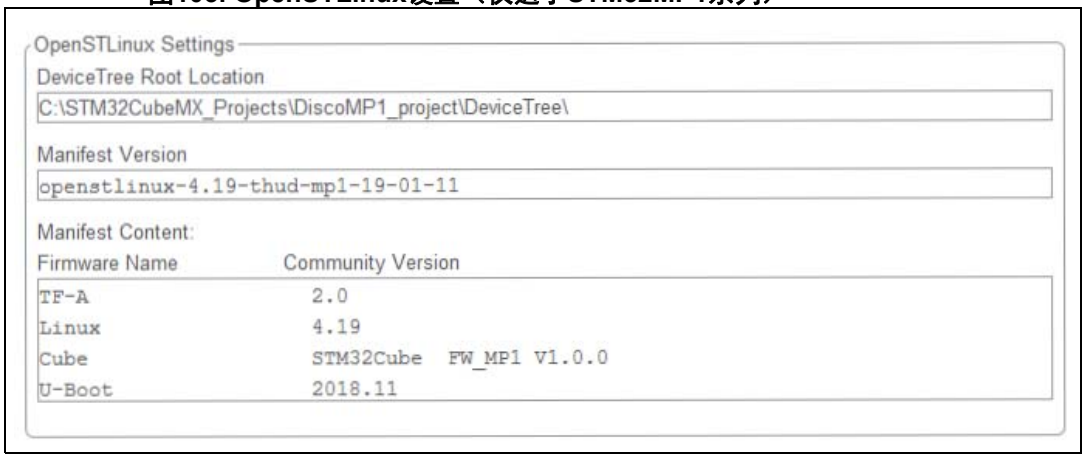
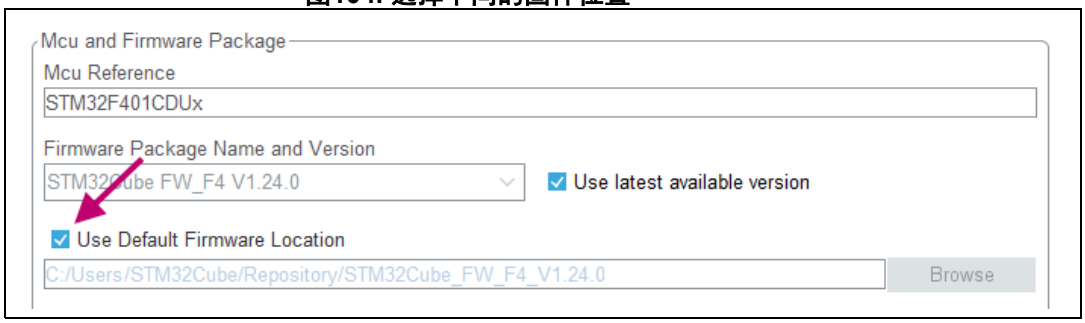
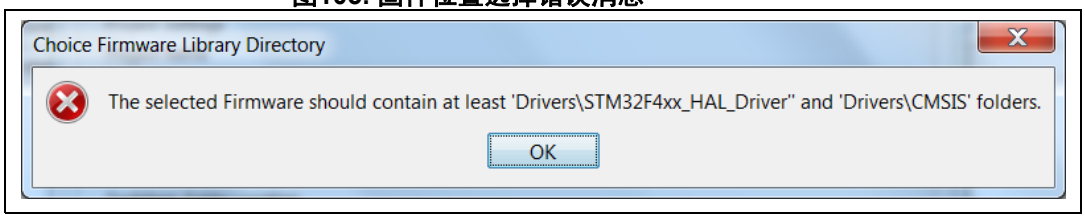


图104. 选择不同的固件位置



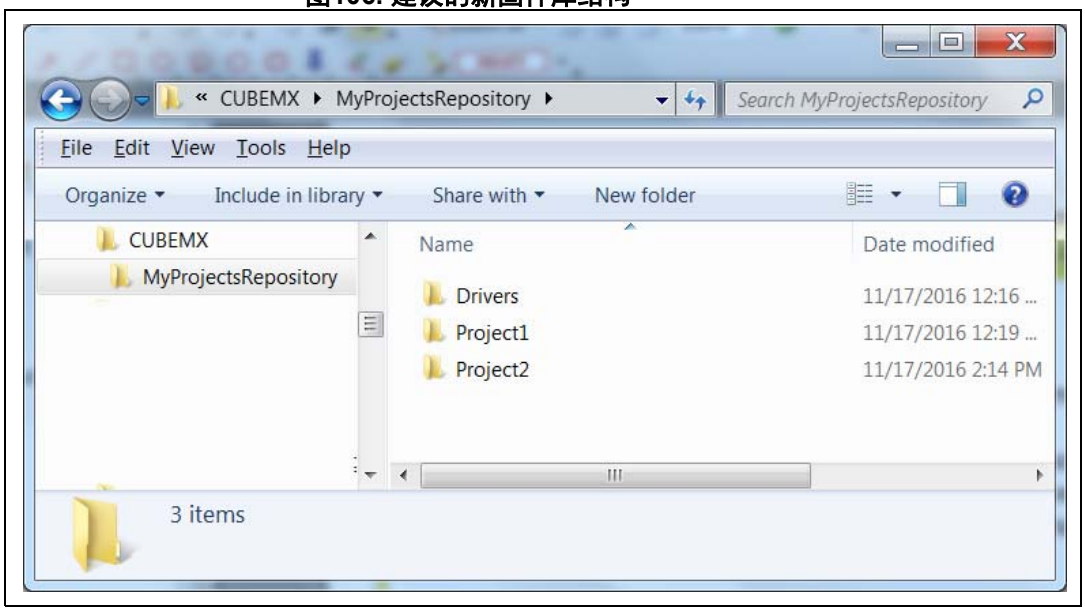
新位置须至少包含一个*驱动*目录，其中包含来自相关STM32Cube MCU封装的HAL和CMSIS驱动。如果找不到文件夹，将弹出错误消息（参见*图 105*）。

图105. 固件位置选择错误消息



要在所有使用相同固件位置的项目中复用相同的*驱动*文件夹，从**代码生成器**选项卡选择**添加库文件作为参考**（参见*图 106*）。

图106. 建议的新固件库结构



注意： STM32CubeMX仅为该默认位置管理固件更新。选择另一个位置可使用户无法从自动更新中获益。用户必须手动将新的驱动版本复制到其项目文件夹中。

4.9.2 “代码生成器”选项卡

“代码生成器”选项卡允许指定以下代码生成选项（参见*图 107*）：

- “STM32Cube固件库包”选项
- “生成的文件”选项
- “HAL设置”选项
- “自定义代码模板”选项

“STM32Cube固件库包”选项

可以进行以下操作：

- 将所有使用过的库复制到项目文件夹中

  STM32CubeMX将与用户配置有关的驱动程序库（HAL、CMSIS）和中间件库复制到用户项目文件夹（例如，FatFs、USB）。
- 只复制必需的库文件：

  STM32CubeMX只将与用户配置有关的库文件复制到用户项目文件夹（例如，HAL库中的SDIO HAL驱动程序）。
- 添加工具链项目配置文件中引用的所需库

  默认情况下，需要的库文件被复制到用户项目中。选择此选项以使配置文件指向STM32CubeMX库中的文件：用户项目文件夹将不保存库文件的副本，而只保存对STM32CubeMX库中文件的引用。

“生成的文件”选项

用户可在此区域定义以下选项：

- 生成的外设初始化为一对 .c/.h文件，或者将所有外设初始化保存在main.c文件中。
- 将之前生成的文件备份在备份目录中

  为之前生成的.c/.h文件添加.bak扩展名。

  在重新生成C代码时保留用户代码。

  此选项仅适用于STM32CubeMX生成文件中的用户部分。它不适用于可能被手动添加或通过ftl模板生成的用户文件。
- 在当前配置不再需要以前生成的文件时，删除这些文件。例如，如果在以前的代码生成中启用的UART外设目前在当前配置中被禁用，则删除uart.c/.h文件。

“HAL设置”选项

此区域允许选择以下任意一个HAL设置选项：

- 将所有不用的引脚设为模拟类型，以优化功耗
- 启用/禁用*Full Assert*功能：stm32xx_hal_conf.h配置文件中的定义语句添加注释或不加注释。

“自定义代码模板”选项

要生成自定义代码，单击**“模板设置”**下的**“设置”**按钮，打开“模板设置”窗口（参见*图 108*）。

然后提示用户选择要从中选择代码模板的源目录，以及生成相应代码的目标目录。

默认源目录指向STM32CubeMX安装文件夹中的extra_template目录，该目录用于存储所有用户定义的模板。默认的目标文件夹位于用户项目文件夹中。

STM32CubeMX 然后使用选定的模板生成用户自定义代码（参见*第 6.3节：自定义代码生成*）。

*图 109*显示*图 108*上显示的模板配置结果：根据sample_h.ftl 模板定义生成一个sample.h文件。

图107. 项目设置代码生成器

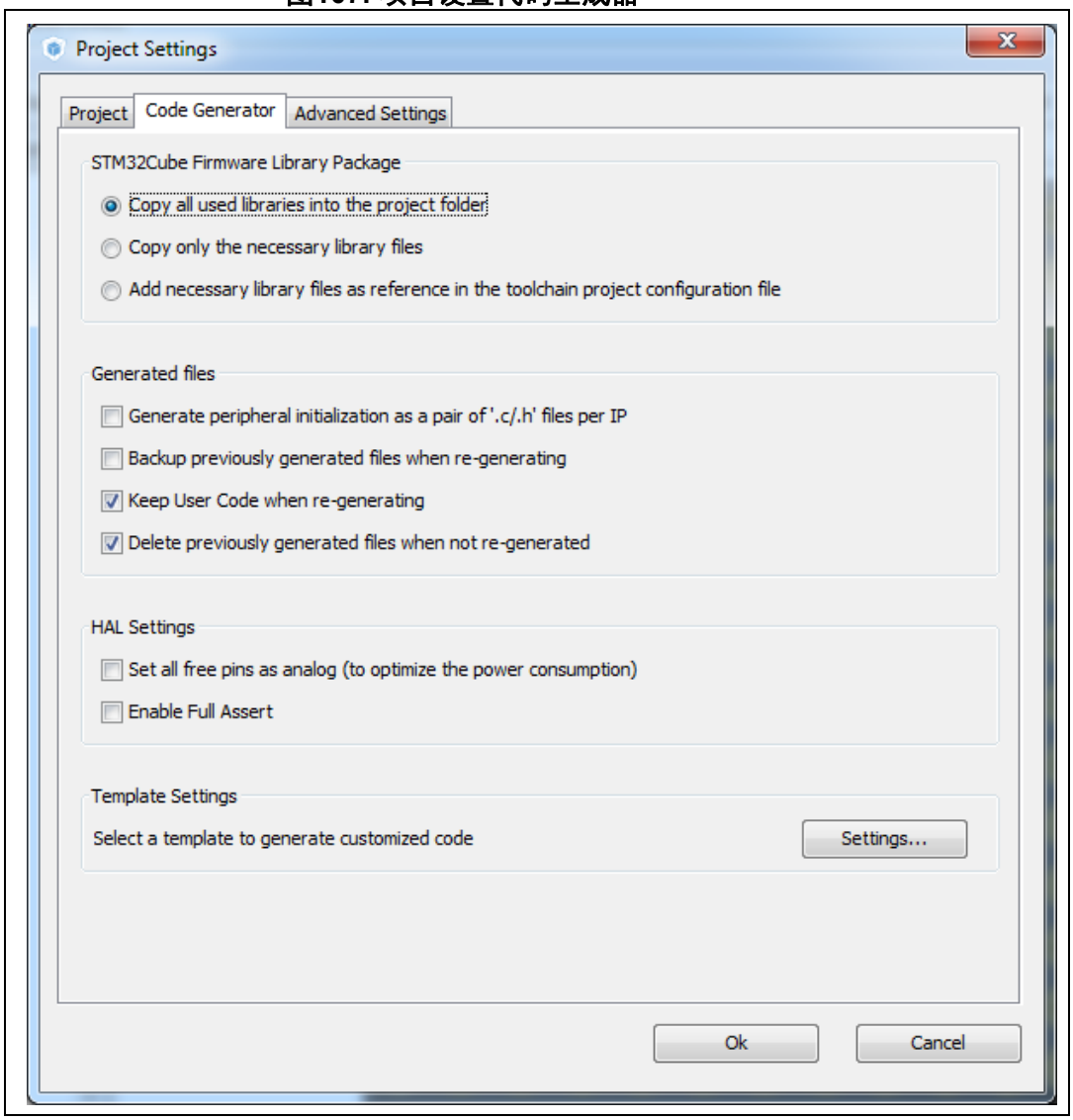


图108. “模板设置”窗口

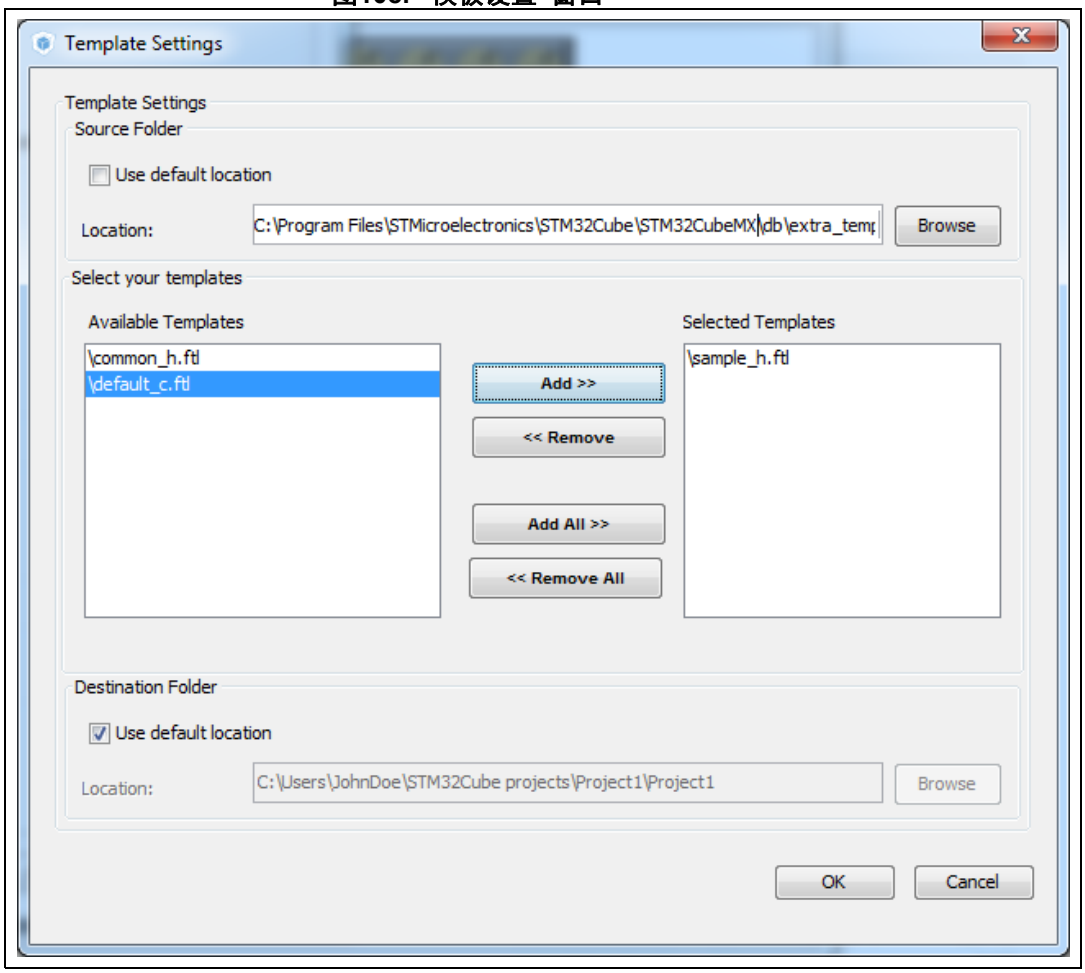
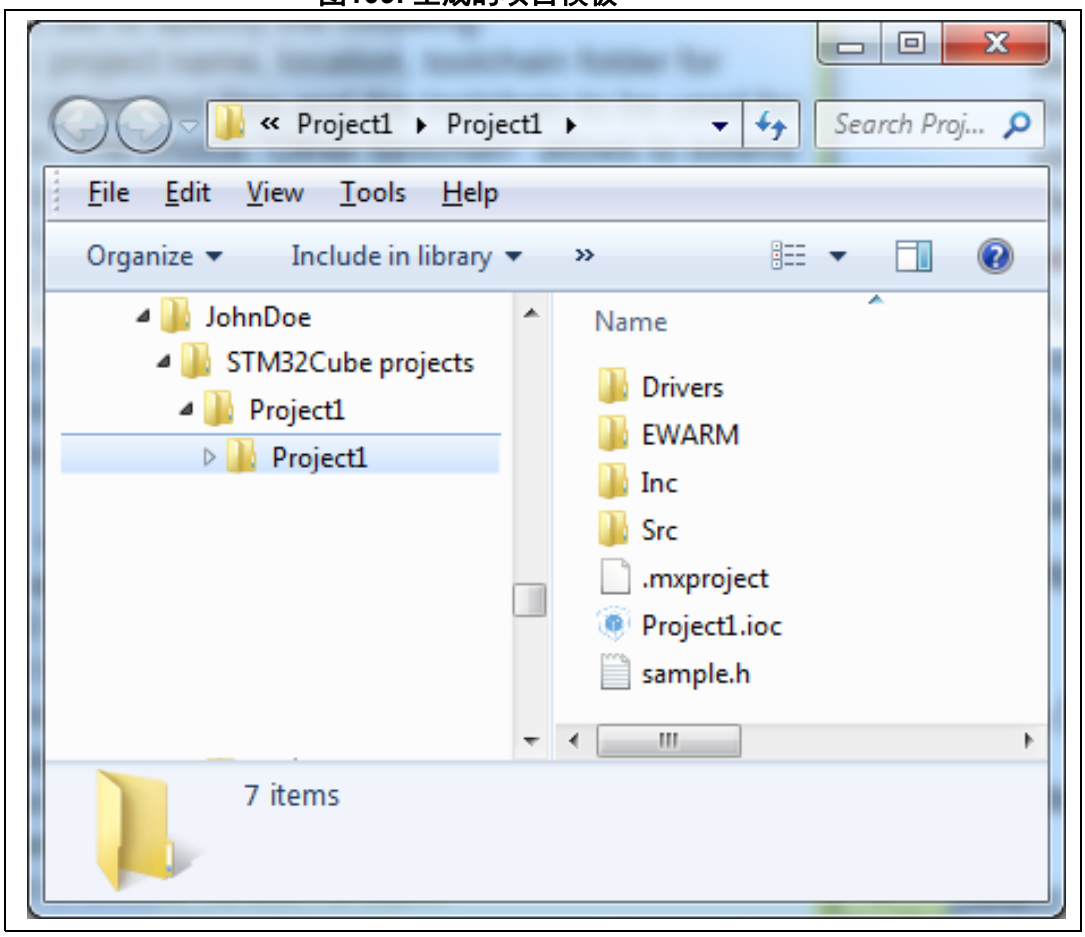

Template Settings
Template Settings
Source Folder
Use default location
Location:
C:\Program Files\STMicroelectronics\STM32Cube\STM32CubeMX\db\extra_temp
Browse
Select your templates
Available Templates
Selected Templates
\common_h.ftl
\default_c.ftl
\sample_h.ftl
Add >>
<< Remove
Add All >>
<< Remove All
Destination Folder
Use default location
Location:
C:\Users\JohnDoe\STM32Cube projects\Project1\Project1
Browse
OK
Cancel

图109. 生成的项目模板



4.9.3 “高级设置”选项卡

*图 110*显示为项目选择的外设和/或中间件。

初始化函数调用排序

默认情况下，生成的代码按照STM32CubeMX中外设和中间件的使能顺序调用外设/中间件初始化函数。然后，用户可以使用向上和向下箭头按钮修改排名编号以重新排序。

重置按钮允许切换回字母顺序。

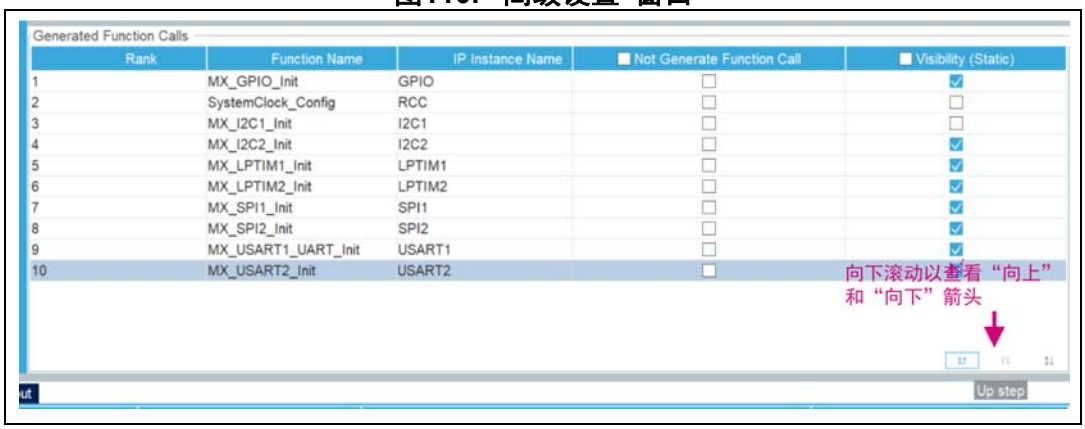
禁用对初始化函数的调用

如果勾选“**不生成**”复选框，则STM32CubeMX 不生成对相应外设初始化函数的调用。这取决于用户代码。

为给定的外设实例选择基于HAL或LL的代码生成

从STM32CubeMX4.17和STM32L4系列开始，STM32CubeMX使一些外设可以生成基于低层（LL）驱动程序而不是HAL驱动程序的初始化代码：用户可以在**驱动程序选择器**部分中选择LL或HAL驱动程序。从而相应地生成代码（参见*第 6.2 节：使用底层驱动程序生成STM32Cube代码*）。

图110. “高级设置”窗口



取消选择**可见性（Static）**选项（如*图 110*中所示的MX_I2C1_init函数操作）允许在不使用static关键字的情况下生成函数定义，从而将其可见性扩展到当前文件之外（参见*图 111*）。

图111. 在不使用C语言“static”关键字的情况下生成的初始化函数

```
/* Private function prototypes -----------
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_LPTIM1_Init(void);
static void MX_LPTIM2_Init(void);
void MX_I2C1_Init(void);
static void MX_I2C2_Init(void);
static void MX_SPI1_Init(void);
static void MX_SPI2_Init(void);
static void MX_USART1_UART_Init(void);
static void MX_USART2_Init(void);
```

注意： 仅适用于STM32MP1系列
在默认情况下，由于未激活“项目管理器/高级设置”面板中的“不生成函数调用”框，因此在STM32Cube Cube固件main()函数中调用SystemClock_Config函数（请参阅*图 110*）。
该配置对于在工程模式（Cortex-M4独立模式）下运行STM32Cube固件有效。
该配置对于在生产模式下运行STM32Cube固件无效：必须选中“项目管理器/高级设置”面板下的“不生成函数调用”框，因此不会调用*main()*函数中的*SystemClock_Config()*。

4.10 “导入项目”窗口

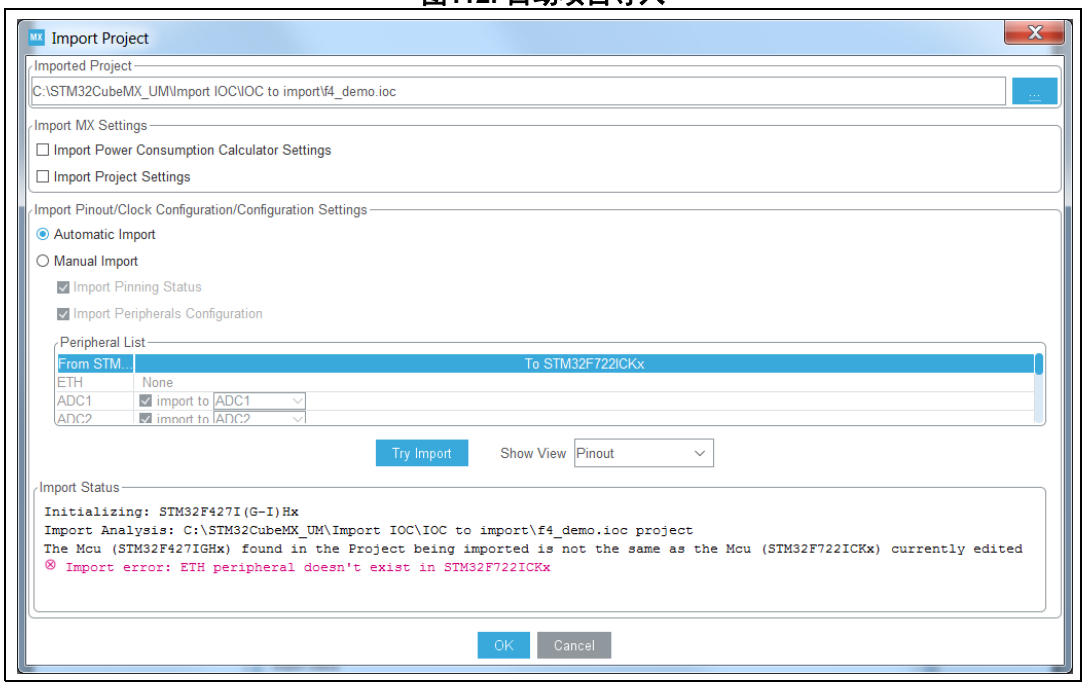
“导入项目”菜单简化了先前保存的配置到另一个MCU的移植。在默认情况下，导入以下设置：

- **“引脚排列”**选项卡：MCU引脚和相应的外设模式。如果目标MCU中没有相同的外设实例，则导入失败。
- **“时钟配置”**选项卡：时钟树参数。
- **“配置”**选项卡：外设和中间件库初始化参数。
- **“项目”设置**：选择工具链和代码生成选项。

如要导入项目，请按照以下步骤进行：

1. 在启动新项目并选择MCU后，选择显示在**文件**菜单下的**导入项目** 图标  。
   只要在选择MCU之后没有为新项目定义用户配置设置，菜单就保持有效。一旦对项目配置执行了用户操作，该菜单将被禁用。
2. 选择**文件** > **导入项目**将会打开专用的“导入项目”窗口。此窗口允许指定以下选项：
   – 在当前空项目之上导入的项目的STM32CubeMX配置文件（.ioc）路径名。
   – 是否导入**“功耗计算器”**选项卡中定义的配置。
   – 是否导入通过**“项目 > 设置”**菜单定义的项目设置：IDE选择、代码生成选项和高级设置。
   – 是否导入通过**项目** > **设置**菜单定义的项目设置：IDE选择和代码生成选项。
   – 是尝试导入整个配置（自动导入）还是只导入一个子集（手动导入）。
   a) 自动项目导入（参见*图 112*）

图112. 自动项目导入

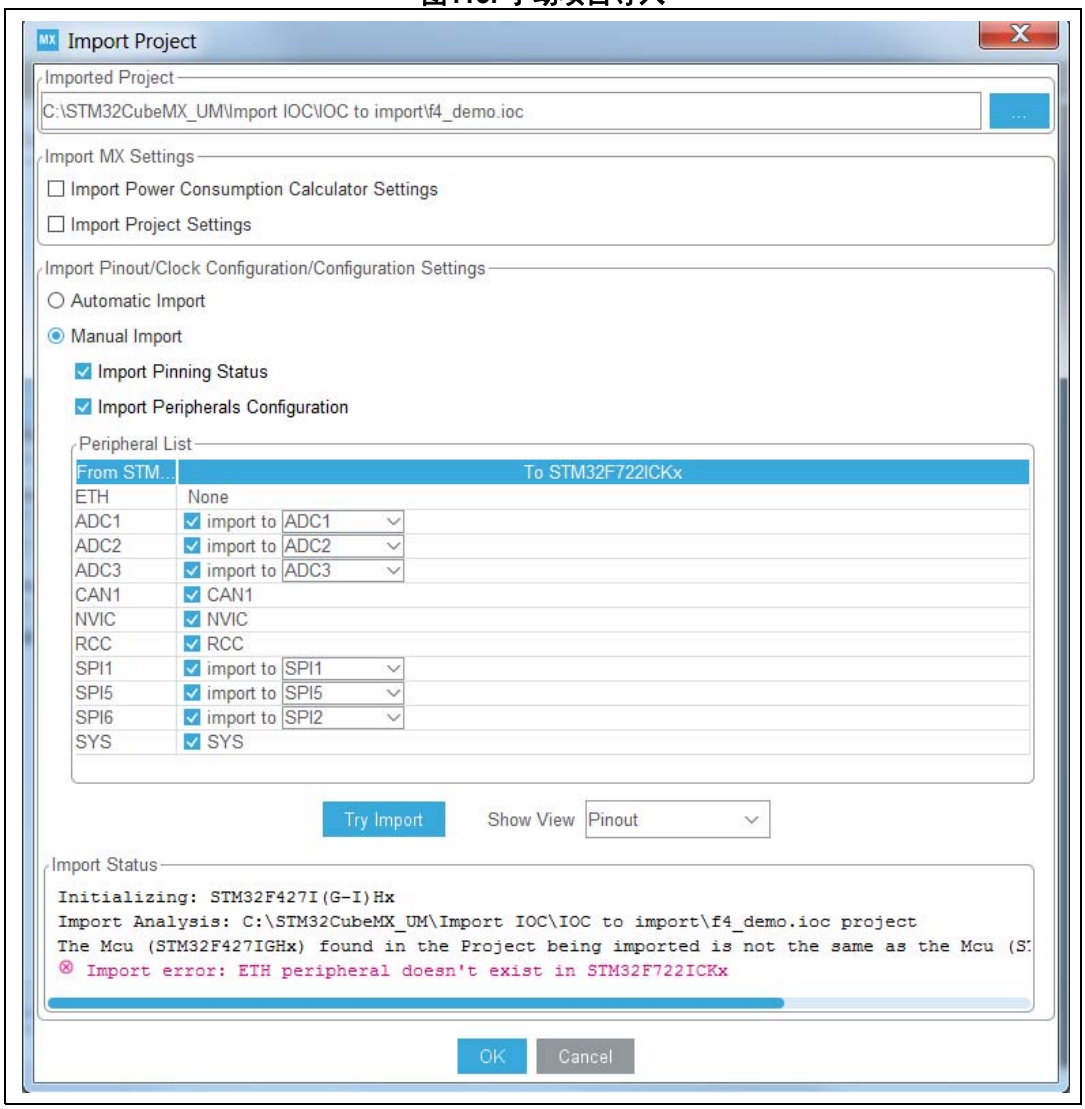

Import Project
Imported Project
C:\STM32CubeMX_UM\Import IOC\IOC to import\f4_demo.ioc
Import MX Settings
Import Power Consumption Calculator Settings
Import Project Settings
Import Pinout/Clock Configuration/Configuration Settings
Automatic Import
Manual Import
Import Pinning Status
Import Peripherals Configuration
Peripheral List
From STM...
To STM32F722ICKx
ETH
None
ADC1
import to
ADC1
ADC2
import to
ADC2
Try Import
Show View
Pinout
Import Status
Initializing: STM32F427I(G-I)Hx
Import Analysis: C:\STM32CubeMX_UM\Import IOC\IOC to import\f4_demo.ioc project
The Mcu (STM32F427IGHx) found in the Project being imported is not the same as the Mcu (STM32F722ICKx) currently edited
Import error: ETH peripheral doesn't exist in STM32F722ICKx
OK
Cancel

b) 手动项目导入

在这种情况下，用户可以通过复选框手动选择外设集（参见*图 113*）。

选择**尝试导入**选项进行尝试导入。

图113. 手动项目导入



外设列表显示：

– 要导入的项目中配置的外设实例

– 如果当前选定的MCU存在任何外设实例，必须将配置导入这些实例。如果有多个外设实例可供导入，用户需要选择其中一个外设。

如果导入引脚更少的更小封装或外设选项更少的低端MCU，可能会发生冲突。

点击**尝试导入**按钮以检查此类冲突：“导入状态”窗口和“外设”列表刷新后可指示错误（参见*图 114*）、警告，以及导入是否已经成功：

– 警告图标表示用户已多次选择一个外设实例，其中一个导入请求将不被执行。

– 交叉符号表示存在引脚排列冲突，在这种情况下，配置将无法导入。

可以通过手动导入微调导入选项并解决导入尝试中遇到的问题。*图 115*是一个成功的导入尝试示例，在取消对某些外设的导入请求后获得成功。

显示视图功能允许在不同的配置选项卡（引脚排列、时钟树、外设配置）之间切换，以便在实际部署之前检查“尝试导入”操作对当前项目的影响（参见*图 115*）。

图114. “导入项目” 菜单 - 尝试导入时出现错误

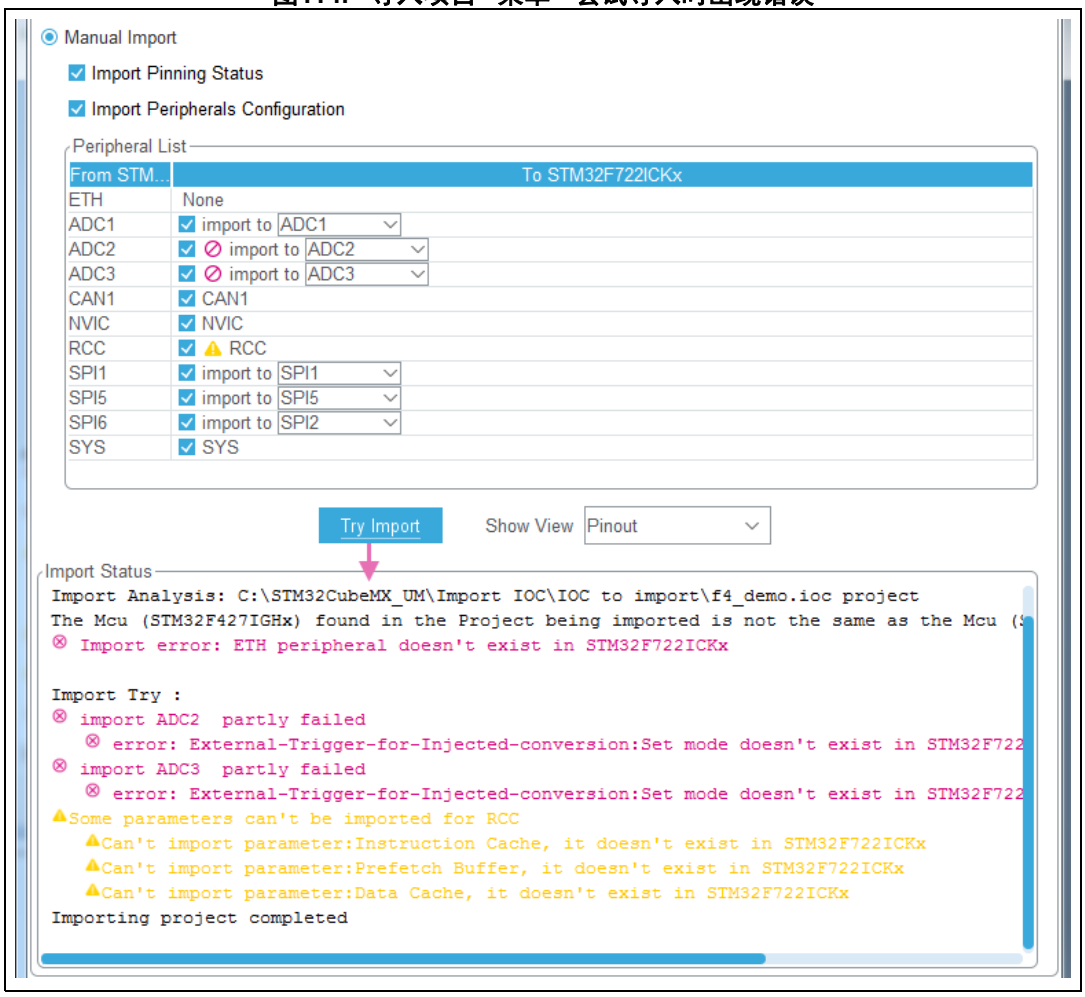
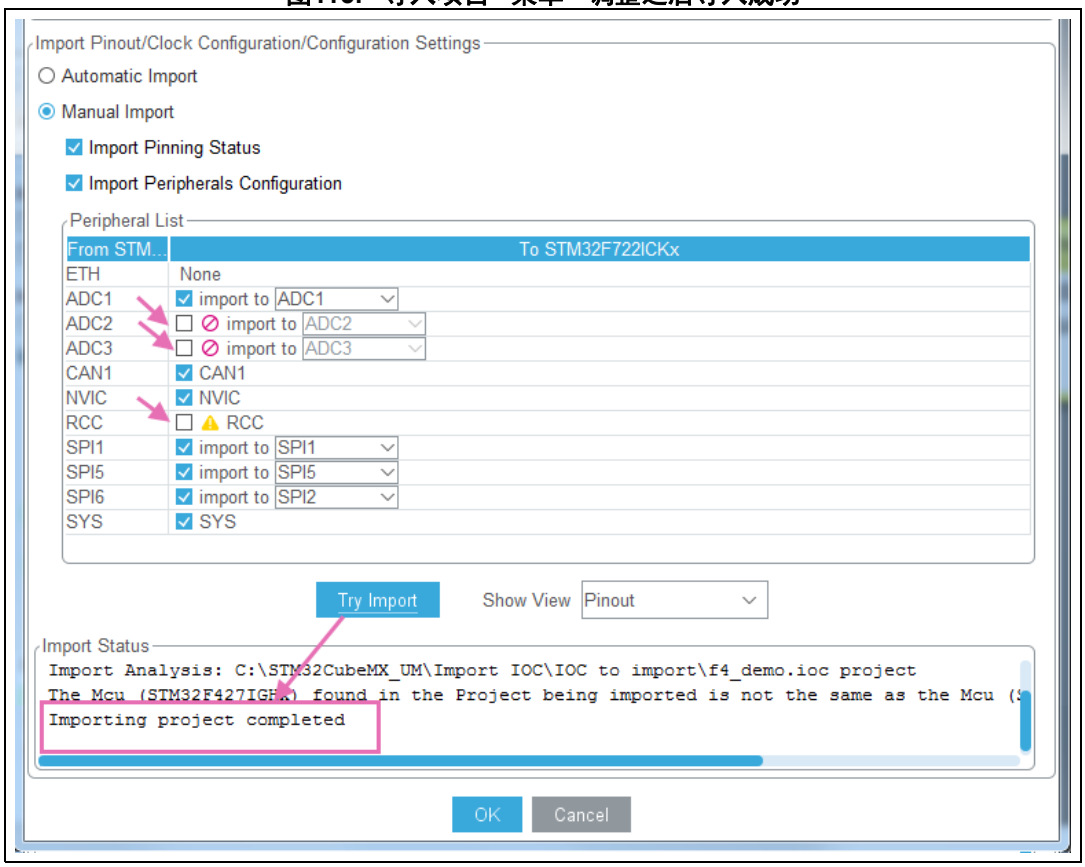


图115. “导入项目” 菜单 - 调整之后导入成功



3. 选择**确定**以当前状态导入，或选择**取消**返回空项目，无需进行导入。
在导入时，“导入”图标会变灰，因为MCU此时已经配置好，不可能再导入一个非空配置。

4.11 “设置未使用 / 重置已使用GPIO”窗口

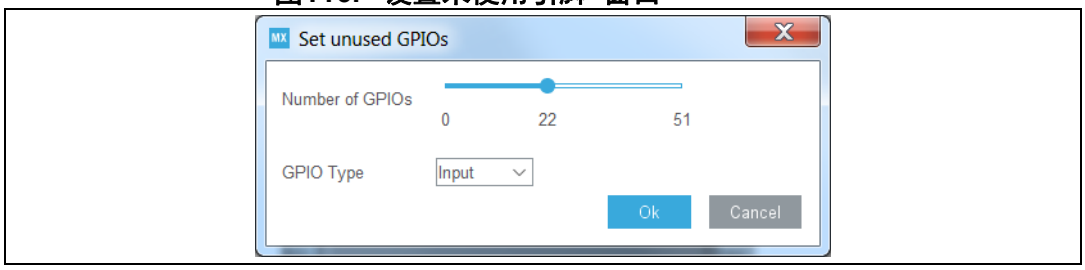
这些窗口用于在同一GPIO模式下同时配置多个引脚。

如要打开它们：

- 从STM32CubeMX菜单栏选择**引脚排列** > **设置未使用GPIO** 。

注： *用户选择GPIO的数量，并允许STM32CubeMX在可用的引脚中选择要配置或重置的实际引脚。*

图116. “设置未使用引脚”窗口



- 从STM32CubeMX菜单栏选择**引脚排列 > 重置已使用GPIO** 。

  根据是否在工具栏上选中了“保持当前信号位置”选项，STM32CubeMX 冲突解算器能够/不能将GPIO信号移至其他未使用的GPIO：

  – 如果未勾选“保持当前信号位置”选项，STM32CubeMX冲突解算器能够将GPIO信号移至未使用的引脚，以便适应另一种外设模式。

  – 如果勾选了“保持当前信号布置”选项，GPIO信号将不能移动，可能的外设模式数量受限。

  参照*图 118*和*图 119*并检查可用外设模式的限制。

图117. “重置已使用引脚”窗口

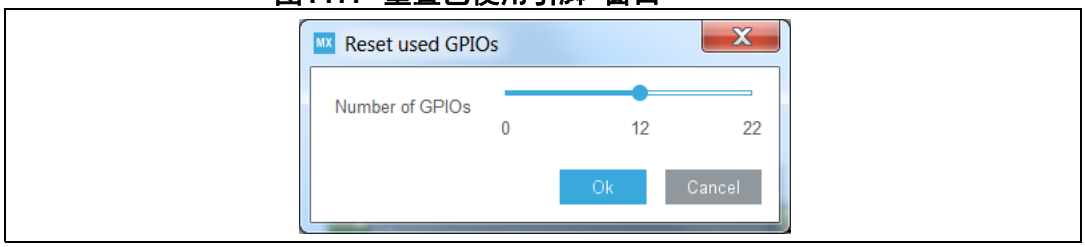


图118. 在勾选了“保持当前信号位置”选项的情况下设置未使用的GPIO引脚

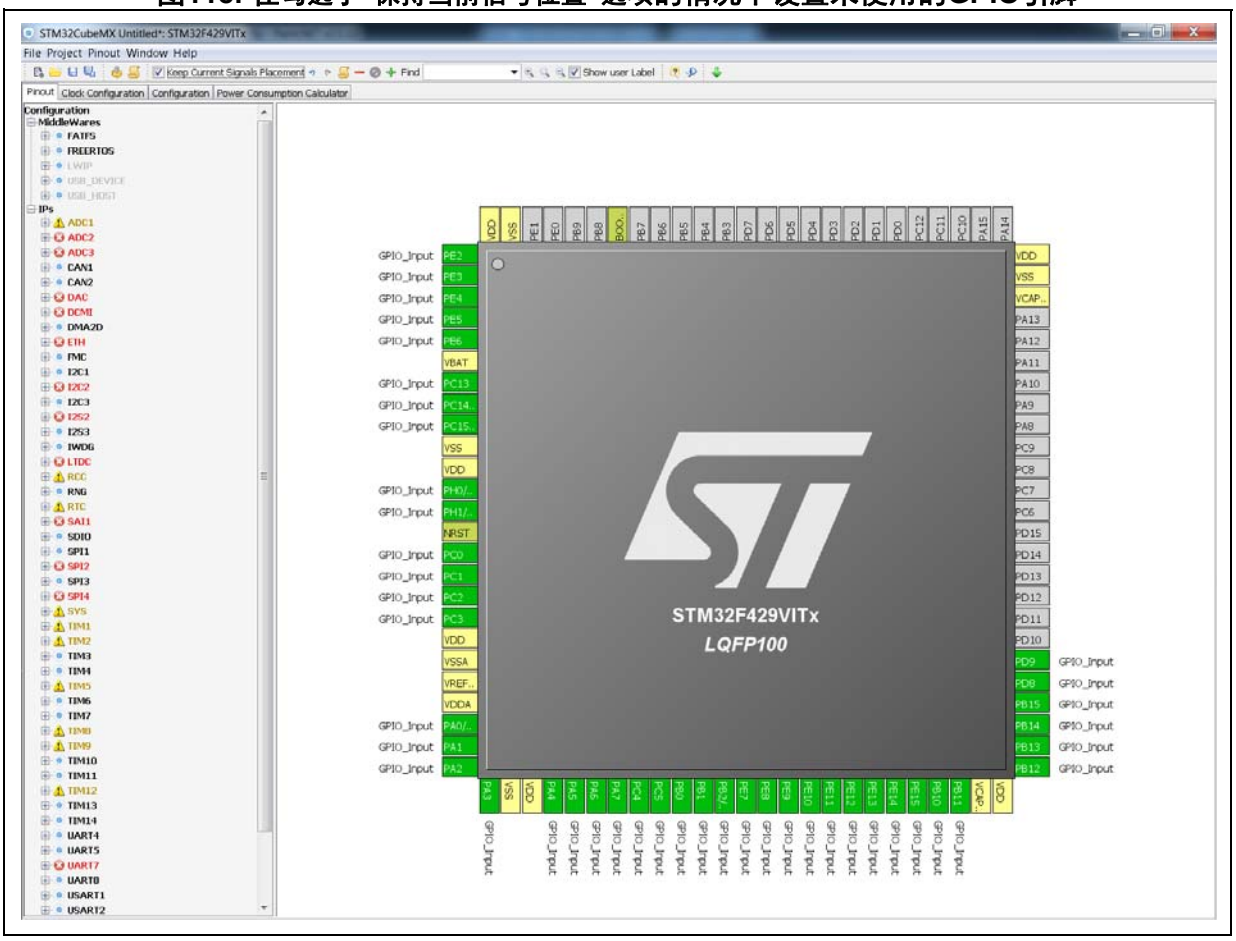
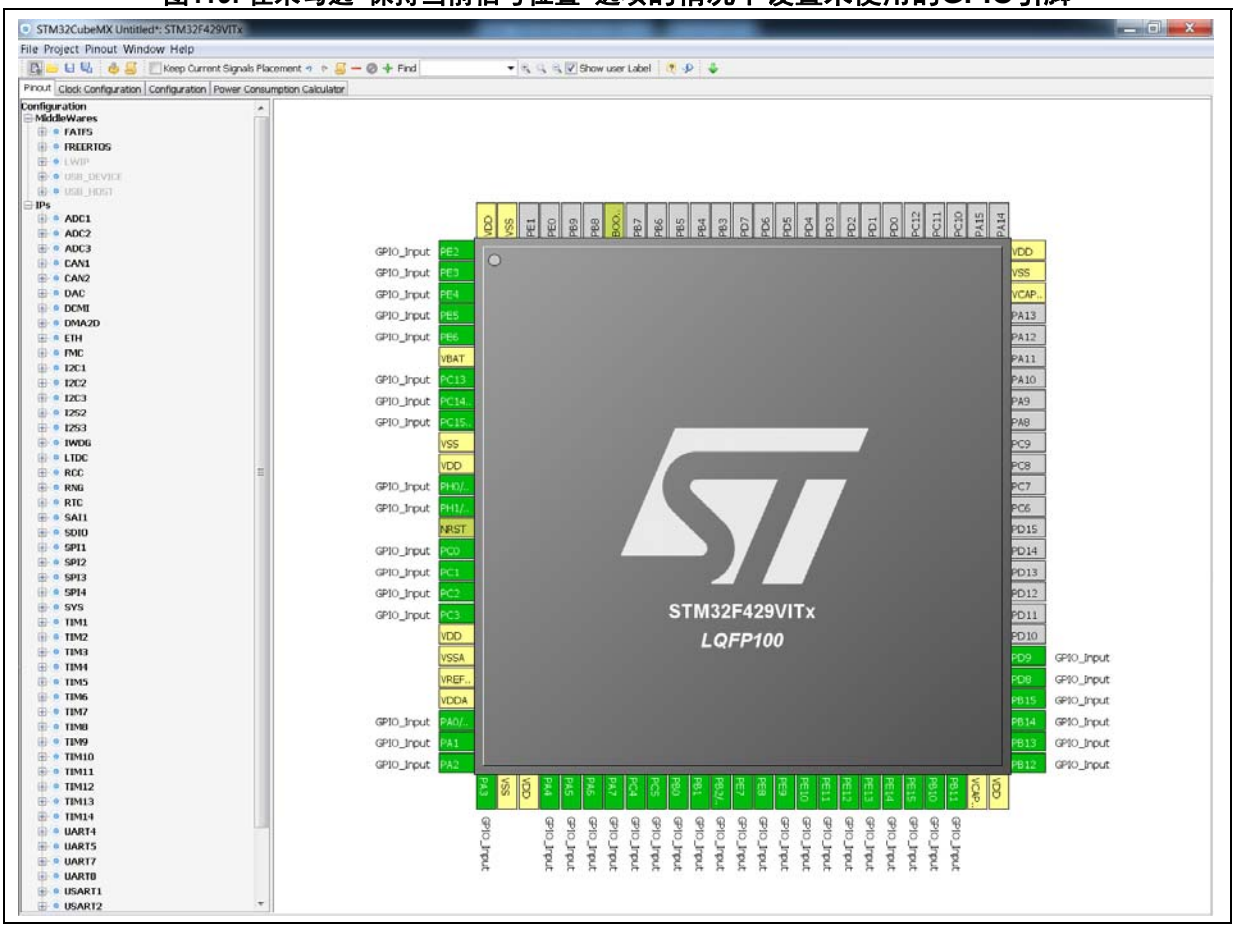


图119. 在未勾选“保持当前信号位置”选项的情况下设置未使用的GPIO引脚



4.12 “更新管理器”窗口

可以通过STM32CubeMX菜单栏中的**“帮助”**菜单访问三个窗口：

1. 选择**帮助 > 检查更新**以打开**检查更新管理器**窗口并查找可下载的最新软件版本。
2. 选择**帮助 > 管理嵌入式软件包**以打开**嵌入式软件包管理器**窗口并查找可下载的嵌入式软件包。它还允许检查包更新和删除以前安装的软件包。
3. 选择**帮助 > 更新程序设置**以打开**更新程序设置**窗口并配置更新机制设置（代理设置、手动更新或自动更新、存储嵌入式软件包的存储库文件夹）。

有关这些窗口的详细描述，请参见*第 3.4 节：使用STM32CubeMX进行更新*。

4.13 附加软件组件选择窗口

进行项目操作时，可以随时在**“引脚布局和配置”**选项卡中单击**“其他软件”**来打开“其他软件组件选择”窗口。它允许用户为当前项目选择附加软件组件。多核产品当前不支持此功能。

它分为四个面板，如*图 120*中所示：

- **筛选器**面板

  可以使用“显示/隐藏筛选器”按键隐藏。它位于窗口的左侧，提供一组用于筛选包组件列表的条件。

- **软件包**面板

  这是主面板，显示了每个可以为项目选择的软件包的软件组件列表。

- **组件依赖性**面板

  可以使用“显示/隐藏相关性”按键隐藏。它会显示“软件包”面板中所选组件的依赖性（如果有）。会在找到任何解决方案时，提出解决方案。

  未解决的依赖性以紫红色图标突出显示。

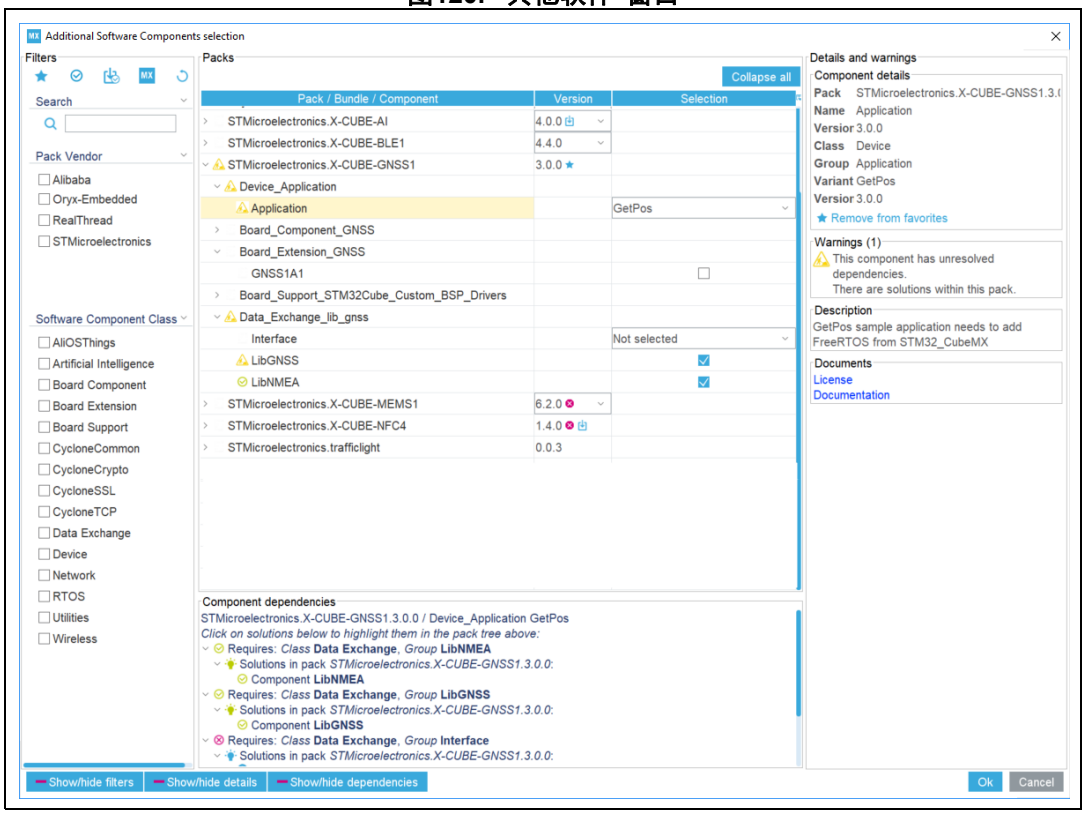
  解决依赖性后（通过在候选解决方案中选择一个组件），将以绿色图标突出显示。

- **“详细信息和警告”**面板

  可以使用“显示/隐藏详细信息”按键隐藏。它位于右侧。其中提供了在“软件包”面板中选择的元件信息。

  该元件可以是软件包、套件或组件。其中可以对可用但尚未安装的软件包版本进行安装，还允许用户将当前项目迁移到软件包的较新版本，这会引起无法自动解决的不兼容问题。

图120. “其他软件”窗口



有关如何通过STM32CubeMX CMSIS-Pack集成对其他软件组件进行处理的更多详细信息，请参阅*第 10节：支持使用 CMSIS-Pack 标准的其他软件组件*。

4.13.1 软件组件简介

Arm®Keil™ CMSIS-Pack 标准定义了要作为软件包分发的软件组件的软件包 （*.pdsc）格式。软件包是一个包含 *.pdsc 描述文件的 zip 文件。

STM32CubeMX对pack .pdsc文件进行解析，以提取软件组件列表。该列表显示在“软件包”面板中。

Arm® Keil™ CMSIS-Pack 标准将软件组件定义为文件列表。组件或每个相应的单独文件都可以选择引用必须解析为 true 的条件，否则组件或文件在给定上下文中不适用。这些条件在 “**组件依赖性** ” 面板中列出。

没有组件名称。相反，每个组件都以类名称、组名称和版本的组合作为给定供应商包的唯一标识。可以分配其他类别，如子组和型号。这些详细信息在**“详细信息和警告”**面板中列出。

4.13.2 筛选器面板

如要筛选软件组件列表，请选择包供应商名称和软件组件类别，或在搜索字段中输入文本字符串。

软件组件搜索结果表已经折叠。点击左箭头将其展开并显示所有符合筛选条件的组件。

表14. “其他软件”窗口-筛选器图标

图标	说明
	仅显示收藏的软件包。 通过单击在**“详细信息和警告”**面板中将一个软件包设置为收藏 
	仅显示选定的组件。 当同一组件有多个实现选择时，通过复选框或变量选择在“软件包”面板中选择组件。
	仅显示已安装的软件包。 启用以显示或隐藏尚未安装的软件包。 尚未安装的软件包以图标区分 
	仅显示与该版本的STM32CubeMX兼容的软件包。 与该版本不兼容的软件包以图标区分 
	重置所有筛选器

4.13.3 “软件包”面板

在默认情况下，“软件包”面板显示一个折叠视图：显示所有已知软件包的名称以及一个给定版本（默认为最新版本）。图标仅用于突出显示软件包版本或组件的状态（请参阅表软件包面板图标）。**“详细信息和警告”**以及**“组件依赖性”**面板用于提供详细信息。

单击左箭头可以展开默认视图，以显示下一个级别，该级别可以是“套件”或顶层部组件。最低级别为组件级别。

注： *某些软件包在Arm®内核、STM32系列或STM32 MCU上可能是有条件的，而且仅在所选的MCU符合标准时才可见。例如，说明‘<accept Dcore="Cortex-M4"/>”条件的软件包仅适用于具有Arm®的MCU。*
Cortex®-M4内核。

表15. “其他软件”窗口-“软件包”面板列

列名称	说明
软件包/套件/组件	在软件包级别，显示<软件包名称> 在套件包级别，显示<类别名称>_ <套件名称（如果有）> 在组件级别，显示<组名>/<子组名（如果有）>。 类名称由Arm CMSIS标准标准化(1)
版本	显示从软件包的一个或多个可用版本列表中选择的版本。 套件和组件可以沿用软件包的版本，也可以具有其特定版本。版本显示在**“详细信息和警告”**面板中。
选择	当只有一种实现方式可用时，通过复选框选择组件；如果存在变体，则从列表中选择组件。

1. Arm® Keil™ CMIS-Pack 网站（http://www.keil.com）列出以下类别：
   数据交换：用于数据交换的软件组件
   文件系统：文件驱动支持和文件系统
   图形：用于用户界面的图形库
   网络：使用 Internet 协议的网络堆栈
   RTOS：实时操作系统
   安全：根据安全标准测试应用软件的组件
   保密：为安全通信或存储而加密
   USB：通用串行总线栈
   无线：Bluetooth®、 WiFi®、以及 ZigBee® 等通信协议栈。

表16. “其他软件”窗口-“软件包”面板图标

图标	说明
	已将软件包添加到用户收藏的软件包列表中。 使用**“详细信息和警告”**面板从收藏列表中添加/删除软件包。
	软件包版本与该STM32CubeMX版本不兼容。 解决方案：选择兼容版本。
	软件包版本尚未安装。 解决方案：转到**“详细信息和警告”**面板，下载软件包版本以将其用于项目。
	该组件不可选择。 解决方案：下载该组件所属的软件包。
	选择一个组件，至少还有一个条件需要解决。 选择具有该图标的组件行，以刷新**“组件依赖性”**面板的相关性、状态和解决方案的列表（如果有）。
	至少选择一个组件，并且满足所有条件（如果有）。
	可以切换到其他软件包版本。 解决方案：使用**“详细信息和警告”**面板进行更改。

4.13.4 “组件依赖性”面板

条件是应用于给定软件组件的依赖规则。选择一个组件时，面板会刷新，会提供有关要解决的依赖性和可用解决方案（如果找到）的详细信息（请参阅*表 17*）。

表17. “组件相关性”面板上下文帮助

上下文帮助	说明
This component has no dependency.	没有依赖性要解决。
⊗ Requires: *Class* **CMSIS**, *Group* **CORE** ⊗ **No solution found**	有依赖性要解决但未找到解决方案。
⊗ Requires: *Class* **Data Exchange**, *Group* **Interface** Solutions in pack *STMicroelectronics.X-CUBE-GNSS1.3.0.0*: ? Component **Interface** / Variant **Basic** ? Component **Interface** / Variant **Template**	有依赖性要解决且至少找到一套解决方案。 单击解决方案建议，将其自动重定向到“软件包”面板中的组件选择行。
Requires: *Class* **Data Exchange**, *Group* **LibNMEA** Solutions in pack *STMicroelectronics.X-CUBE-GNSS1.3.0.0*: Component **LibNMEA**	存在依赖性且已解决（已选择满足条件的组件）。

4.13.5 “详细信息和警告”面板

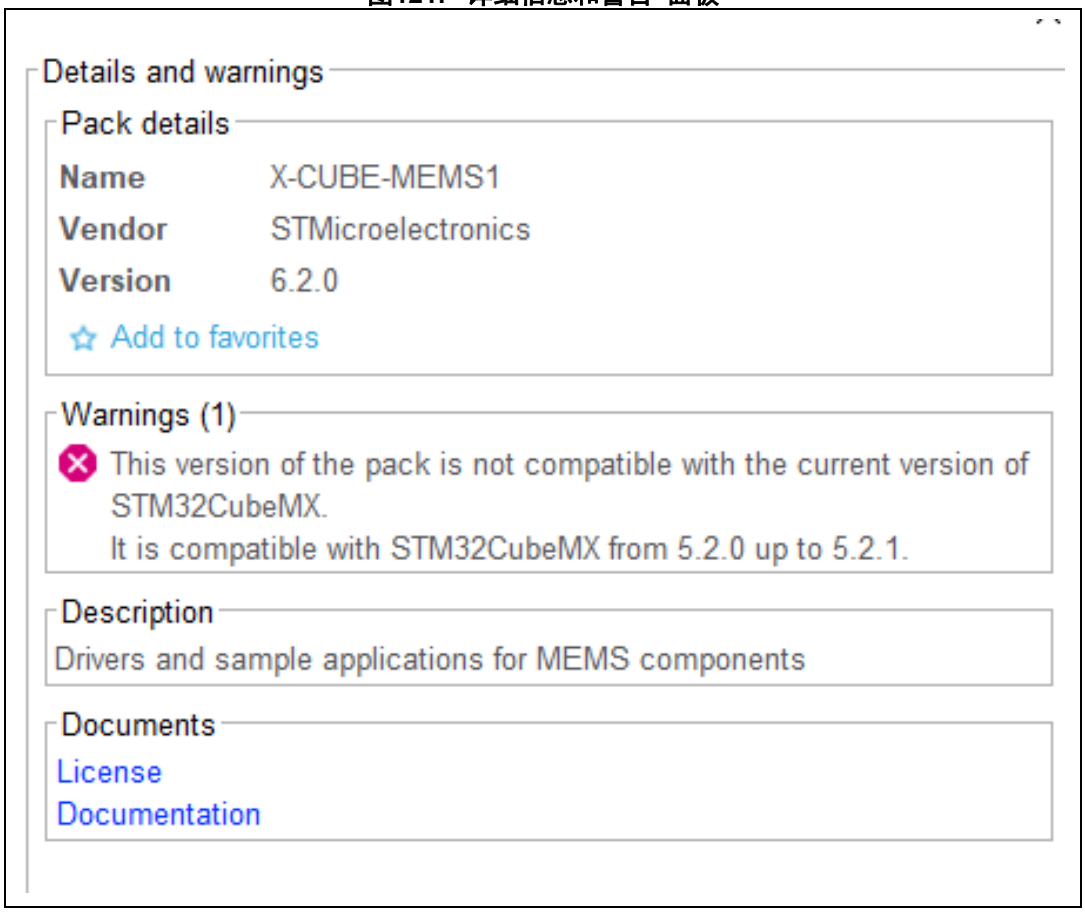
单击 Show/hide details 以取消隐藏面板（请参阅*图 121*）。

从**“软件包”**面板中选择一行后，将刷新该面板。

可以在该面板中进行一些操作，即将软件包添加到收藏的软件包列表中/从收藏的软件包列表中删除软件包，安装软件包，通过链接访问软件包文档。

如果检测到任何问题，将在**“警告”**部分提供说明。

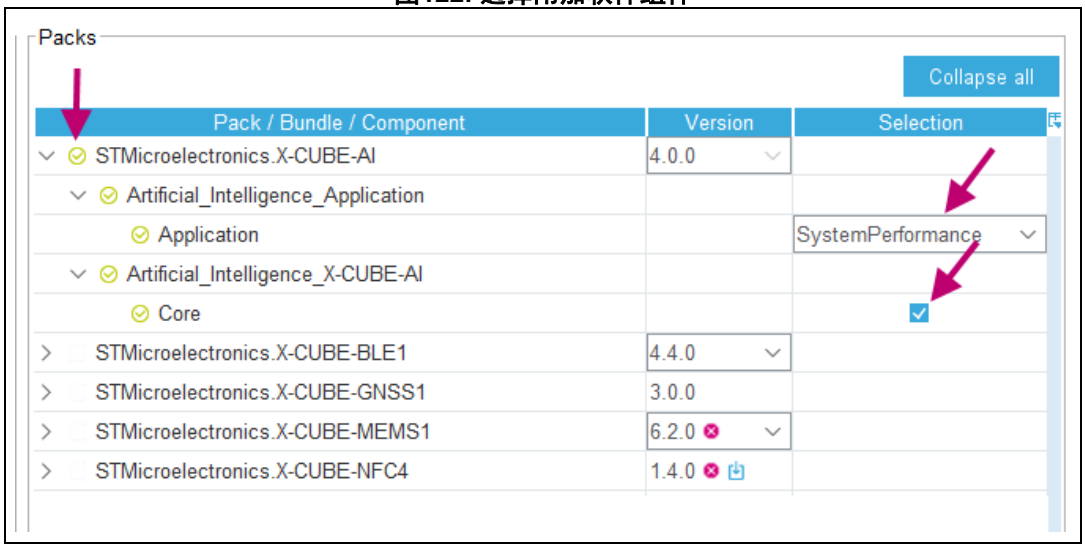
图121. “详细信息和警告”面板



4.13.6 针对其他软件组件更新树形视图

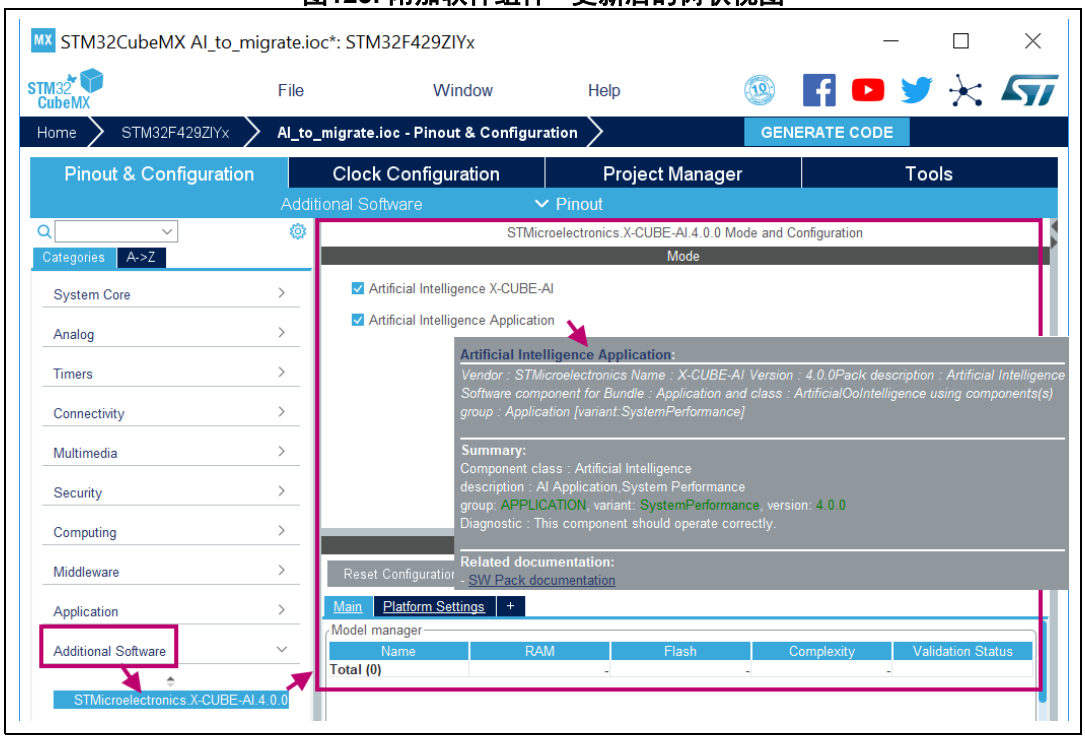
一旦完成应用所需的软件组件的选择（参见*图 122*），点击**确定**以刷新STM32CubeMX窗口：所选的组件出现在附加软件下的树状视图中（参见*图 123*）。

图122. 选择附加软件组件



当前选择的其他软件组件将显示在树形视图中（请参阅*图 123*）。必须在“模式”面板中启用软件组件，而且如果在配置面板中指定了任何参数，则可以进行进一步配置。将鼠标悬停在组件名称上，以显示上下文帮助以及文档链接。

图123. 附加软件组件 - 更新后的树状视图



4.14 “关于”窗口

该窗口显示STM32CubeMX版本信息。

如要打开它，从STM32CubeMX菜单栏选择**帮助** > **关于**。

图124. “关于”窗口

5 STM32CubeMX工具

5.1 功耗计算器视图

对于日益增长的嵌入式系统而言，功耗是一个主要问题。为了帮助将其最小化，STM32CubeMX提供了**"功耗计算器"**选项卡（请参阅*图 125*），其会根据微控制器、电池型号和
用户定义的电源序列，提供以下结果：

- 平均电流消耗

  功耗值可从数据手册中获取，也可以根据用户指定的总线或核心频率进行插值。
- 电池寿命
- 平均DMIP

  DMIP直接从MCU数据手册获得，既不进行插值，也不进行推断。
- 最高环境温度（T_{AMAX}）

  工具根据芯片的内部功耗、封装类型和105°C的最高结温计算最高环境温度，以确保良好的操作条件。

  当前的T_{AMAX}实现未考虑I/O消耗。为精确地估计T_{AMAX}，必须使用额外消耗字段指定I/O消耗。I/O动态电流消耗公式在微控制器数据手册中指定。

功耗计算器视图使开发人员能够以可视化的方式估计嵌入式应用的消耗，并在每个功耗系列步骤中进一步降低消耗：

- 尽可能使用低功耗模式
- 根据步骤要求调整时钟源和频率。
- 为每个阶段使能所需的外设。

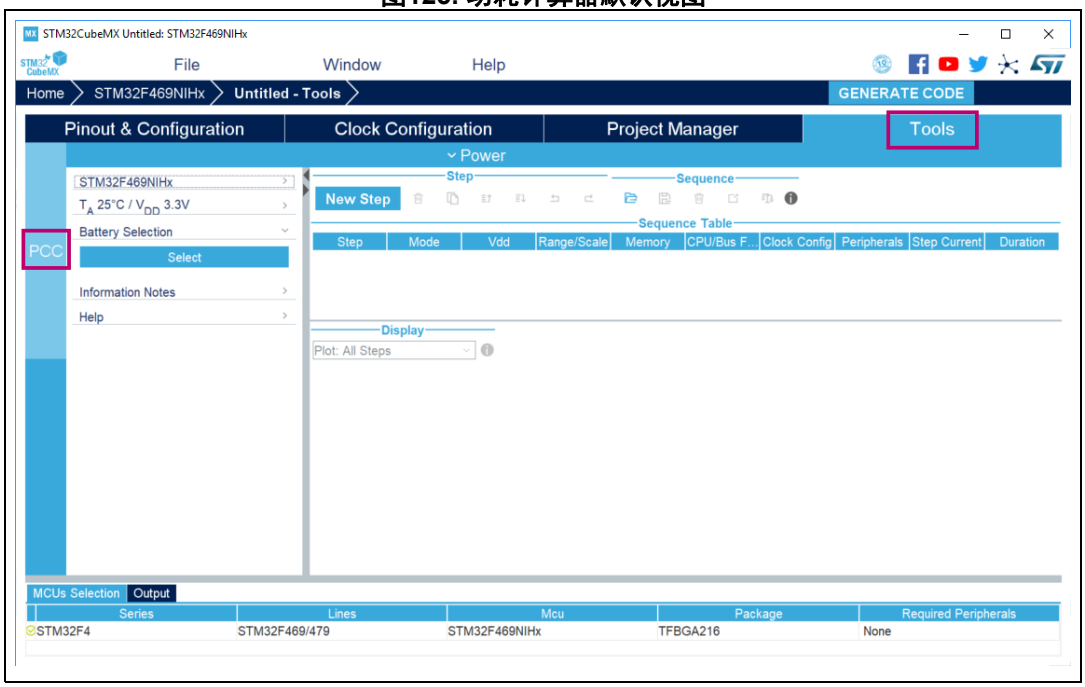
在每个步骤中，用户可以选择将VBUS作为可能的电源，而不是电池。这将影响电池寿命估算。如果可以在不同的电压水平下进行功耗测量，STM32CubeMX还将推荐电压值选择（参见*图 128*）。

STM32L0、STM32L1、STM32L4、STM32L4+、STM32G0、STM32G4、STM32H7和STM32WB系列还提供了另一个选项，即跳变检查器。当使能时，跳变检查器会检测当前配置的序列中的无效跳变。这可以确保在添加新步骤时仅将可能的转换推荐给用户。

5.1.1 建立功耗系列

默认开始视图如*图 125*中所示。

图125. 功耗计算器默认视图



选择V_{DD}值

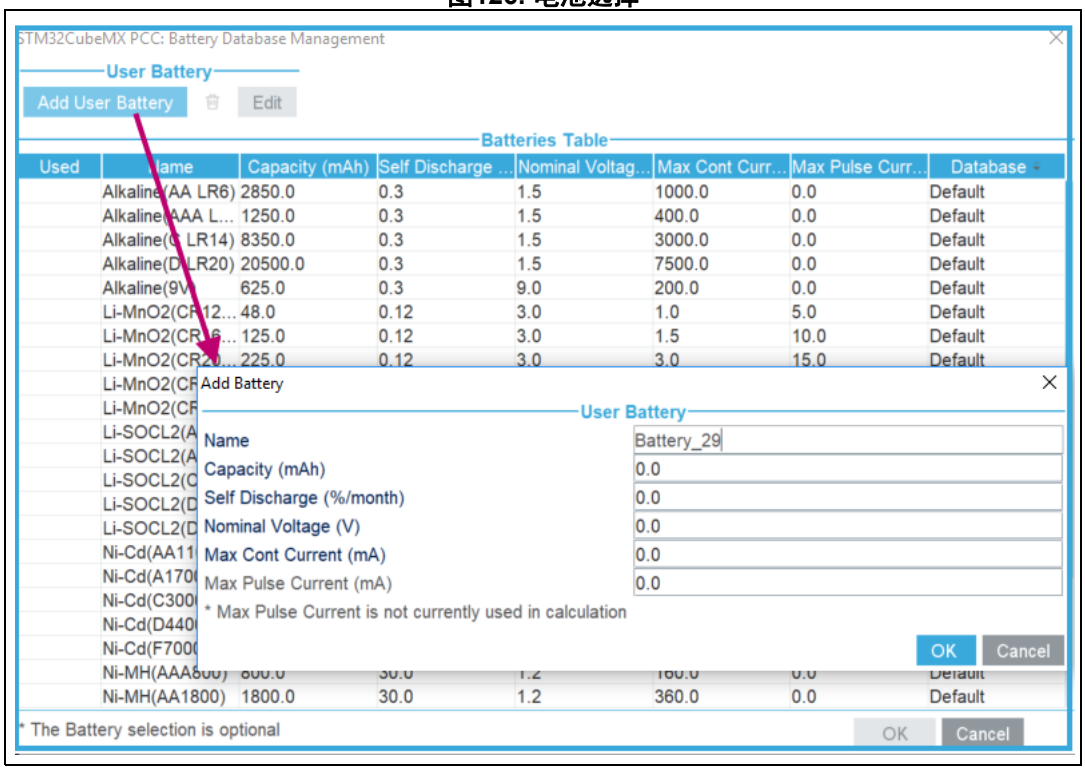
在该视图中，当多个选项可用时，用户必须选择V_{DD}值。

选择电池型号（可选）

用户可以视需要选择电池模型。也可以在配置功耗系列后执行该操作。

用户可以选择预定义的电池或选择指定与其应用最匹配的新电池（参见*图 126*）。

图126. 电池选择



功耗系列默认视图

用户现在可以继续建立功耗系列。

管理序列步骤

可使用一组步骤按钮在一个序列中组织步骤（**添加**新步骤、**删除**步骤、**复制**步骤、在序列中**向上**或**向下**移动）（参见*图 127*）。

通过点击功耗计算器视图中的**撤销**按钮或主工具栏中的撤销图标，用户可以撤销或恢复上一个配置操作

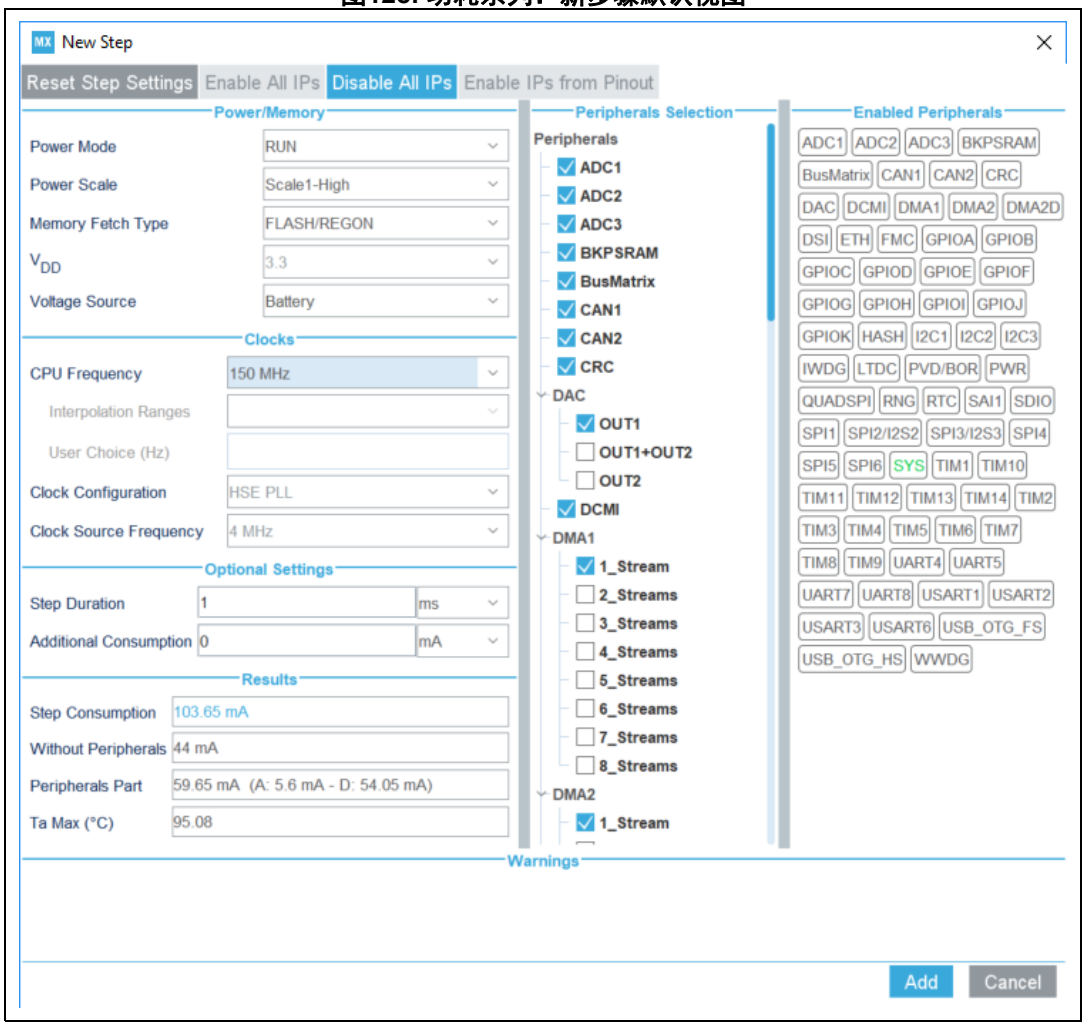
图127. 步骤管理功能

添加一个步骤

可通过两种方法添加新步骤：

- 在功耗面板中点击**添加**。**新步骤**窗口将打开，其中包含空步骤设置。
- 或者，从序列表中选择一个步骤，并点击**复制**。将打开一个**新步骤**窗口，用于复制步骤设置（参见*图 128*）。

图128. 功耗系列：新步骤默认视图



在配置完步骤后，窗口中将显示当前消耗和T_{AMAX}值。

编辑步骤

要编辑步骤，请在序列表中双击它，这将打开**“编辑步骤”**窗口。

移动步骤

默认情况下，在序列末尾添加新步骤。点击序列表中的步骤，以选择步骤，并使用**向上**和**向下**按钮将其移动到序列中的其他位置。

删除步骤

选择要删除的步骤，然后点击**删除**按钮。

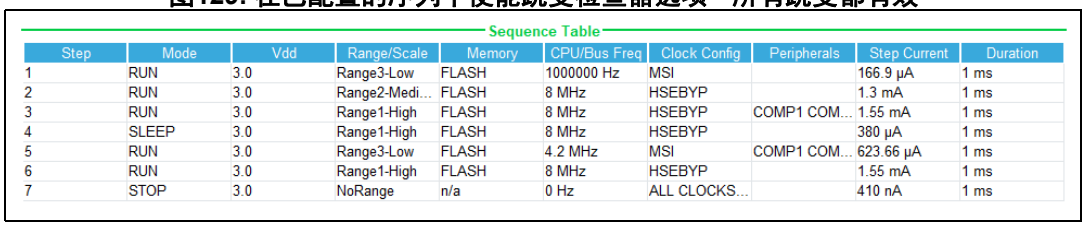
使用跳变检查器

并非功率模式之间的所有转换都可行。“功耗计算器”的电源菜单建议使用跳变检查器来检测无效跳变，或将序列配置仅适于制为有效跳变。

在执行序列配置之前使能跳变检查器选项，以确保用户只能选择有效跳变步骤。

在已配置序列上使能跳变检查器选项后，如果所有跳变有效，将以绿框突出显示（参见*图 129*），如果至少有一个跳变无效，则以紫红色突出显示（紫红框加上以紫红色突出显示的无效步骤描述，参见*图 130*）。后一种情况下，用户可以点击**显示日志**按钮，以了解如何解决跳变问题（参见*图 131*）。

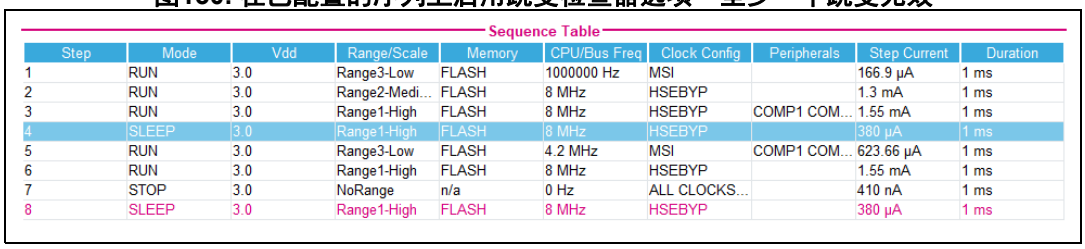
图129. 在已配置的序列中使能跳变检查器选项 - 所有跳变都有效



Sequence Table

Step	Mode	Vdd	Range/Scale	Memory	CPU/Bus Freq	Clock Config	Peripherals	Step Current	Duration
1	RUN	3.0	Range3-Low	FLASH	1000000 Hz	MSI		166.9 µA	1 ms
2	RUN	3.0	Range2-Medi...	FLASH	8 MHz	HSEBYP		1.3 mA	1 ms
3	RUN	3.0	Range1-High	FLASH	8 MHz	HSEBYP	COMP1 COM...	1.55 mA	1 ms
4	SLEEP	3.0	Range1-High	FLASH	8 MHz	HSEBYP		380 µA	1 ms
5	RUN	3.0	Range3-Low	FLASH	4.2 MHz	MSI	COMP1 COM...	623.66 µA	1 ms
6	RUN	3.0	Range1-High	FLASH	8 MHz	HSEBYP		1.55 mA	1 ms
7	STOP	3.0	NoRange	n/a	0 Hz	ALL CLOCKS...		410 nA	1 ms

图130. 在已配置的序列上启用跳变检查器选项－至少一个跳变无效



Sequence Table

Step	Mode	Vdd	Range/Scale	Memory	CPU/Bus Freq	Clock Config	Peripherals	Step Current	Duration
1	RUN	3.0	Range3-Low	FLASH	1000000 Hz	MSI		166.9 µA	1 ms
2	RUN	3.0	Range2-Medi...	FLASH	8 MHz	HSEBYP		1.3 mA	1 ms
3	RUN	3.0	Range1-High	FLASH	8 MHz	HSEBYP	COMP1 COM...	1.55 mA	1 ms
4	SLEEP	3.0	Range1-High	FLASH	8 MHz	HSEBYP		380 µA	1 ms
5	RUN	3.0	Range3-Low	FLASH	4.2 MHz	MSI	COMP1 COM...	623.66 µA	1 ms
6	RUN	3.0	Range1-High	FLASH	8 MHz	HSEBYP		1.55 mA	1 ms
7	STOP	3.0	NoRange	n/a	0 Hz	ALL CLOCKS...		410 nA	1 ms
8	SLEEP	3.0	Range1-High	FLASH	8 MHz	HSEBYP		380 µA	1 ms

图131. 转换检查器选项 - 显示日志

5.1.2 配置功耗系列中的步骤

在**编辑步骤**和**新建步骤**窗口中执行步骤配置。图形界面通过强制设置参数的预定义顺序来引导用户。

其命名可能因所选MCU系列而异。有关每个参数的详细信息，请参阅*第 5.1.4 节*词汇表和*附录DSTM32微控制器功耗参数*，或参阅数据手册的电气特性部分。

当只有一个可能的值时，工具会自动设置参数（在这种情况下，参数无法修改并且灰显）。工具仅推荐与所选MCU相关的配置选择。

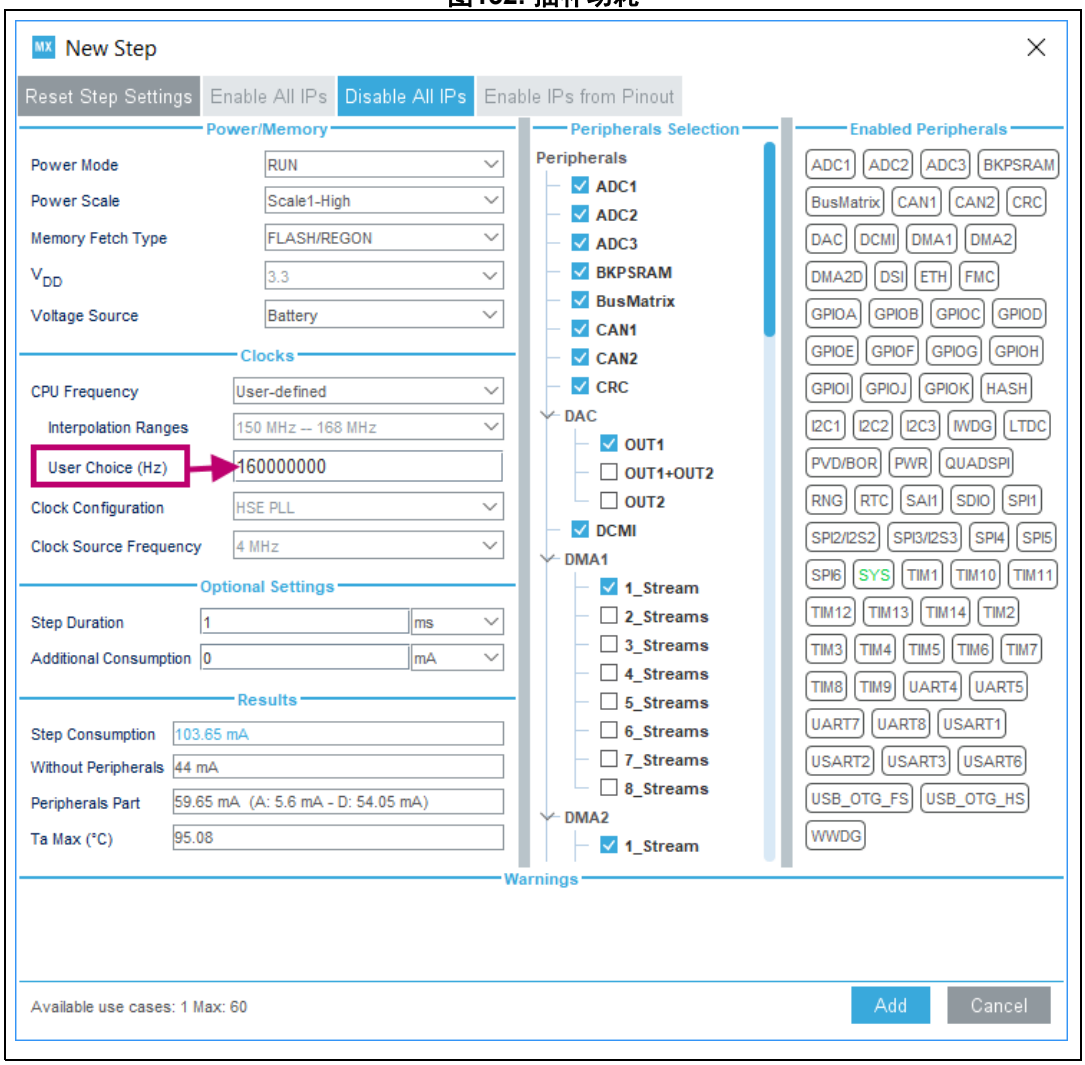
要配置一个新步骤：

1. 单击**添加**或**复制**打开**新步骤**窗口，或双击序列表中的步骤打开**编辑步骤**窗口。
2. 在打开的步骤窗口中，按以下顺序选择：
   - 功率模式
     更改功率模式会重置整个步骤配置。
   - 外设
     可在配置功率模式之后的任何时候选择/取消选择外设。
   - 电源级别
     电源级别与功耗范围（STM32L1）或电源级别（STM32F4）对应。
     更改功率模式或功耗范围会丢弃所有后续配置。
   - 内存提取类型
   - V_{DD}值（如果有多种选择）
   - 电压电源（电池或VBUS）
   - 时钟配置
     更改时钟配置会进一步重置频率选择。
   - 具有多个可用选项时选择**CPU频率**（STM32F4）和**AHB总线频率/CPU频率**（STM32L1），或在激活模式下选择用户指定频率。在这种情况下，将对消耗进行插值（参见*使用插值*）。
3. 可选设置
   - **步骤持续时间**（默认值为1 ms）
   - **额外消耗**值（以mA表示），例如，用于反映应用所使用的外部元件（外部调节器、外部上拉电阻、LED或其他显示器）。添加到微控制器功耗中的该值将影响步骤的总体功耗。
4. 一旦配置完成，**添加**按钮将变为激活。点击该按钮可创建步骤并将其添加到序列表中。

使用插值

对于为激活模式配置的步骤（运行、睡眠），通过选择CPU频率作为用户定义频率并输入以Hz为单位的频率来支持频率插值（参见*图 132*）。

图132. 插补功耗



导入引脚排列

图 133 在**“引脚布局”**视图中示出了ADC配置的示例：在“功耗计算器”视图中单击**“从引脚布局启用IP”**，可选择ADC外设和GPIO A (*图 134*)。

“从引脚布局启用IP”按键使用户可以自动选择在**“引脚布局”**视图中配置的外设。

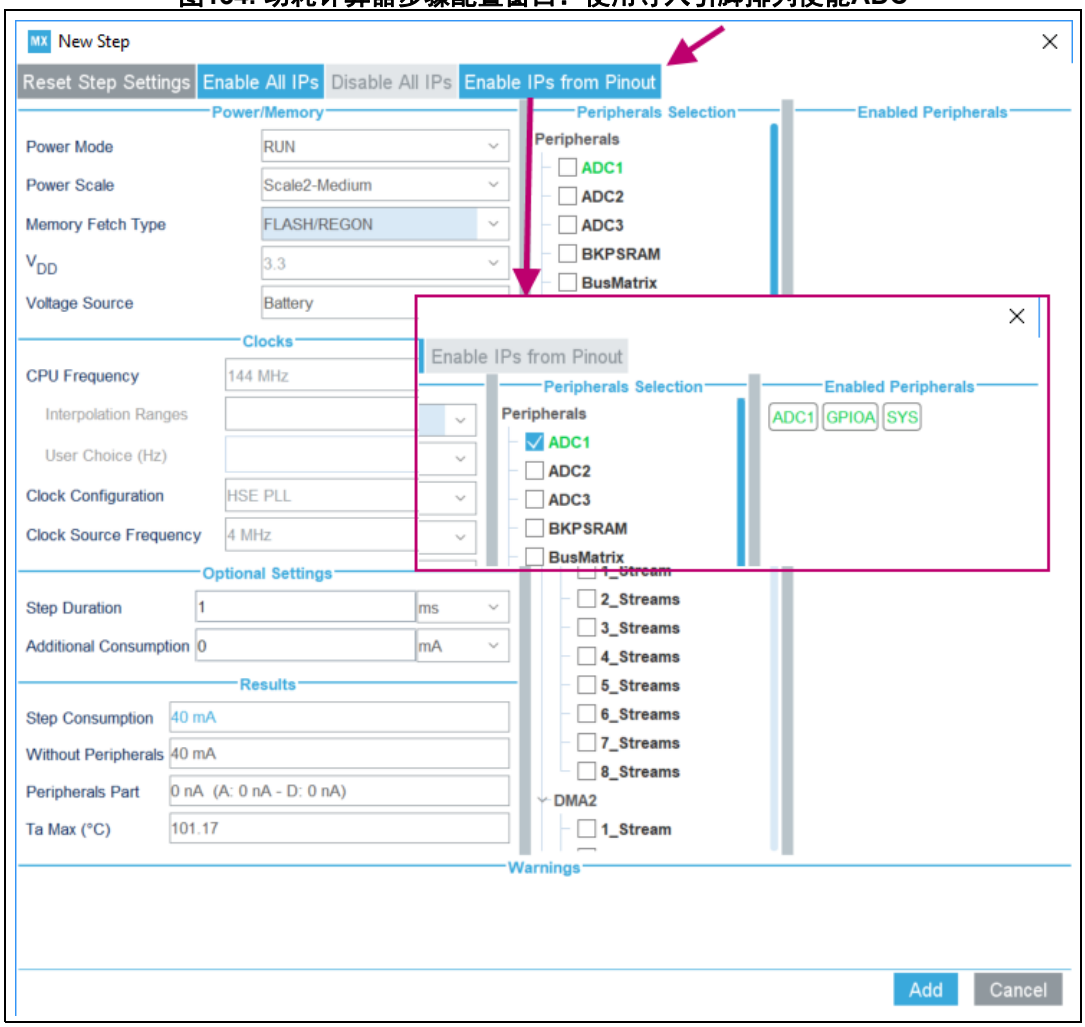
图133. 在引脚排列视图中选择的ADC

选择/取消选择所有外设

单击**“启用所有IP”**使用户可以一次选择所有外设。

单击**“禁用所有IP”**可删除那些产生功耗的IP。

图134. 功耗计算器步骤配置窗口：使用导入引脚排列使能ADC

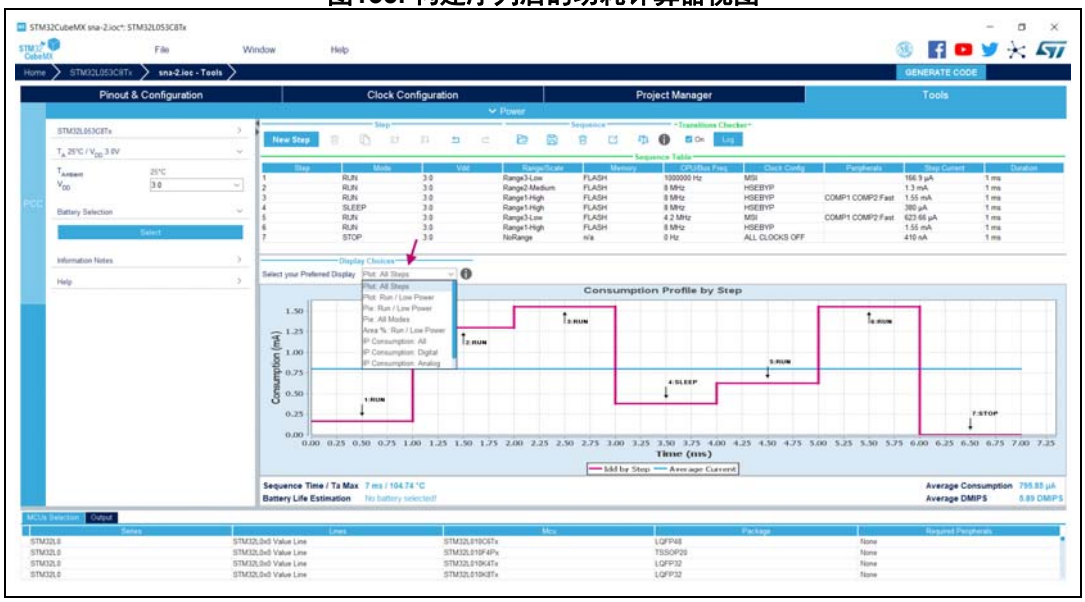


5.1.3 管理用户定义的功耗系列并审查结果

配置功耗系列会引起功耗计算器视图更新（参见*图 135*）：

- 序列表显示所有步骤和步骤参数值。类别列指示消耗值是从数据手册中获取或通过插值得出。
- 序列图区域根据显示类型显示了功耗系列的不同视图（如绘制所有步骤、绘制低功率与运行模式）
- 结果总结提供了总序列时间，最高环境温度（T_{AMAX}），以及选择有效电池配置后的平均功耗、DMIPS和电池寿命估计。

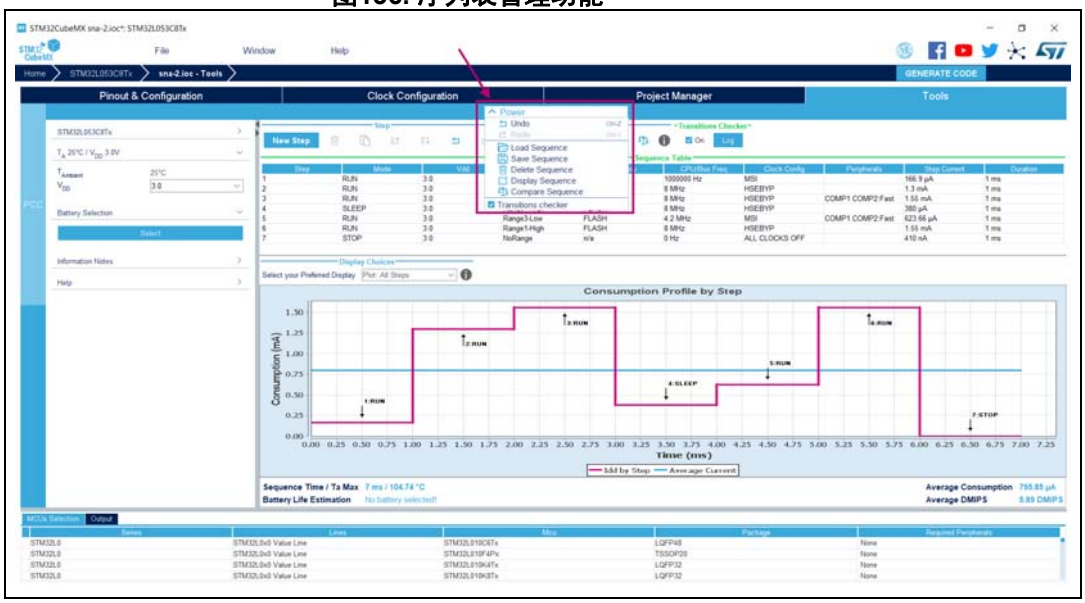
图135. 构建序列后的功耗计算器视图



管理整个序列（加载、保存和比较）

从电源菜单（请参阅*图 136*），可以保存、删除当前序列或将其与以前保存的序列进行比较，其中以前保存的序列将显示在专用弹出窗口中。

图136. 序列表管理功能

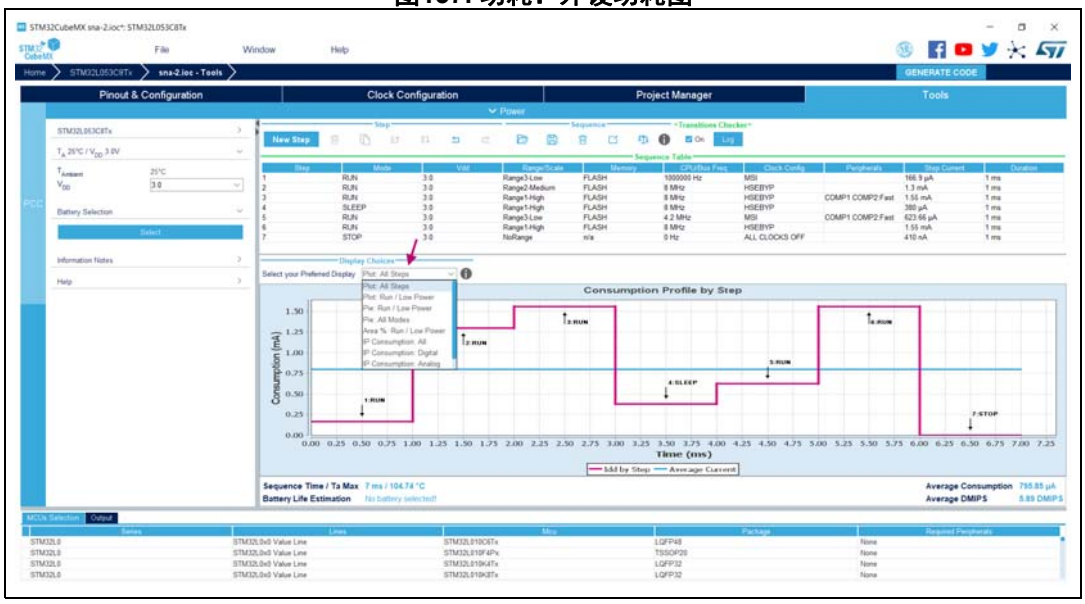


管理结果图表和显示选项

在显示区域中，选择要显示的图表类型（如序列步骤、饼图、每个外设的消耗）。您也可以点击**外部显示**，以在专用窗口中打开图表（参见*图 137*）。

右键单击图表以访问上下文菜单：**属性**、**复制**、**另存为**png图片文件、**打印**、**缩放**菜单，以及**自动范围**，以在执行缩放操作之前重置为原始视图。也可以通过用鼠标从左到右选择图表中的区域来实现**缩放**，通过点击图表并向左拖动鼠标可实现**缩放重置**。

图137. 功耗：外设功耗图

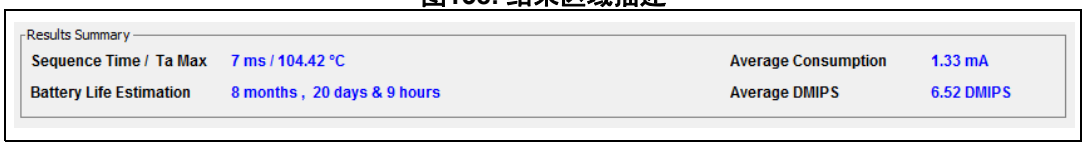


结果总结区域概述

该区域提供以下信息（参见*图 138*）：

- 作为序列步骤持续时间总和的总序列时间。
- 作为由步骤持续时间加权的每个步骤消耗总和的平均消耗。
- 基于Dhrystone基准并强调了定义序列的CPU性能的平均DMIPS（每秒百万条Dhrystone指令）。
- 基于平均功耗与电池自放电的选定电池型号电池寿命估算。
- T_{AMAX}：序列期间所遇到的最高环境温度值。

图138. 结果区域描述



5.1.4 功耗系列步骤参数词汇表

用于表征功耗系列步骤的参数如下（有关更多信息，请参见*附录DSTM32微控制器功耗参数*）：

- 功耗模式

  为节省能耗，建议将微控制器工作模式从需要最大功率的运行模式切换到使用有限资源的低功耗模式。

- V_{CORE}范围（STM32L1）或功率级别（STM32F4）

  这些参数由用于控制数字外设的电源范围的软件设置。

- 内存提取类型

  该字段推荐了应用C代码执行的可能内存位置。它可能是RAM、闪存或ARTON或OFF的闪存（仅适用于具有专有的自适应实时（(ART）内存加速器的系列，当通过闪存执行时，内存加速器可以加快程序执行速度）。

  凭借ART加速器所获得的性能相当于闪存以0个等待周期执行程序。就功耗而言，相当于从RAM执行程序。此外，STM32CubeMX使用相同的选项来覆盖两种设置，即ART ON的RAM和闪存。

- 时钟配置

  该操作设置用于计算微控制器功耗的AHB总线频率或CPU频率。当只有一种可能的选择时，频率会自动配置。

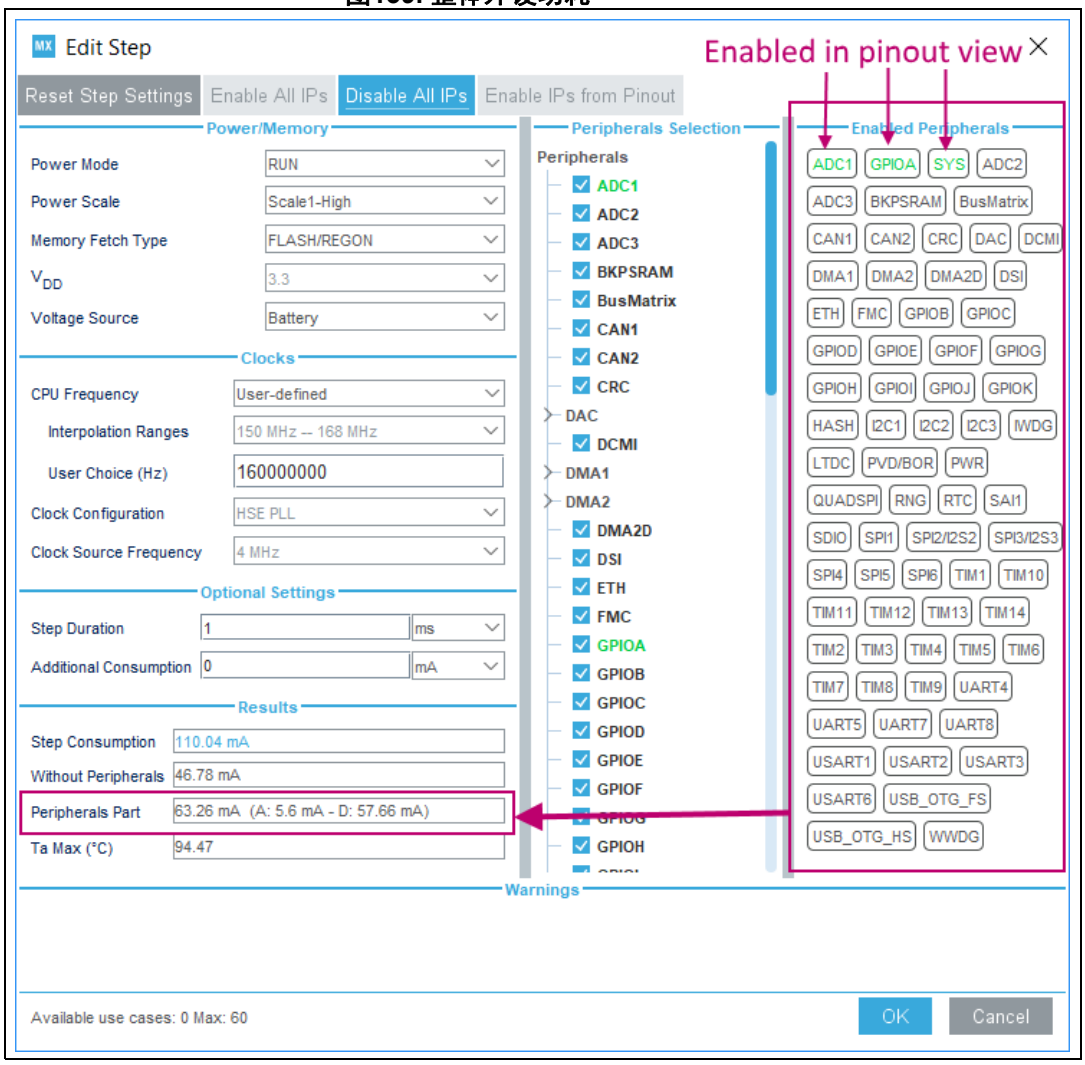
  时钟配置下拉列表允许配置应用时钟：

  – 内部或外部振荡源：MSI、HSI、LSI、HSE或LSE
  – 振荡频率
  – 其他确定参数，其中包括PLL ON、LSE旁路，AHB预分频器值，使用的LCD

- 外设

  外设列表显示了可用于所选功耗模式的外设。功耗在假定外设仅处于时钟模式的情况下给出（如未被运行程序使用）。可使能或禁用每个外设。在工具提示中显示外设的单独功耗。在步骤结果区域中提供外设模拟和数字部分所引起的总功耗（参见*图 139*）。

图139. 整体外设功耗



用户可以选择与应用相关的外设：

– 无（**全部禁用**），
– 一些（使用外设专用的复选框），
– 全部（**全部激活**),
– 或之前定义的引脚排列配置中的全部外设（**导入引脚排列**）。

在计算功耗时，仅考虑选定和已使能的外设。

- 步骤持续时间

用户可以更改默认步骤持续时间值。在构建序列时，用户可以根据应用的实际功率序列创建步骤，或者将其定义为在每种模式下花费的百分比。例如，如果某个应用程序在“运行”模式下消耗30%，在“睡眠”模式下消耗20%，并在“停止”模式下消耗50%，则用户须配置一种3步序列，其中包括运行30 ms，睡眠20 ms和停止50 ms。

- 其他消耗
  该字段允许输入特定用户配置所产生的额外消耗（如为其他连接的设备提供电源的MCU）。

5.1.5 电池词汇表

- 容量（mAh）
  单次电池放电可以提供的能量。
- 自放电（%/月）
  表示在特定时期内未使用电池时（开路条件）的内部泄漏所造成的电池容量损失。
- 标称电压（V）
  充满电的电池所提供的电压。
- 最大值 连续电流（mA）
  该电流对应在电池寿命期间可在不损坏电池的情况下提供的最大电流。
- 最大值 脉冲电流（mA）
  表示在异常情况下可提供的最大脉冲电流，如在启动阶段打开应用时。

5.1.6 SMPS特性

一些微控制器（例如STM32L496xxxxP）允许用户连接外部开关模式电源（SMPS），以进一步降低功耗。

对于此类微控制器，功耗计算器工具提供以下功能：

- 为当前项目选择SMPS
  在左侧面板中，选中**使用SMPS**复选框，以使用SMPS（参见*图 140*）。默认情况下，使用ST SMPS模型。
- 点击**更改**按钮选择另一个SMPS模型
  该操作将打开SMPS数据库管理窗口，用户可以在其中添加新的SMPS模型（参见*图 141*）。然后，用户可以为当前序列选择不同的SMPS模型（参见*图 142*、*图 143*和*图 144*）
- 通过使能SMPS检查器检查当前序列中的无效SMPS转换
  要执行该操作，请通过选中复选框来使能检查器，并单击**帮助**按钮，以打开参考状态图（参见*图 145*）。
- 为每个步骤配置SMPS模式（参见*图 146*）
  如果使能了SMPS检查器，则只推荐对当前步骤有效的SMPS模式。

图140. 为当前项目选择SMPS

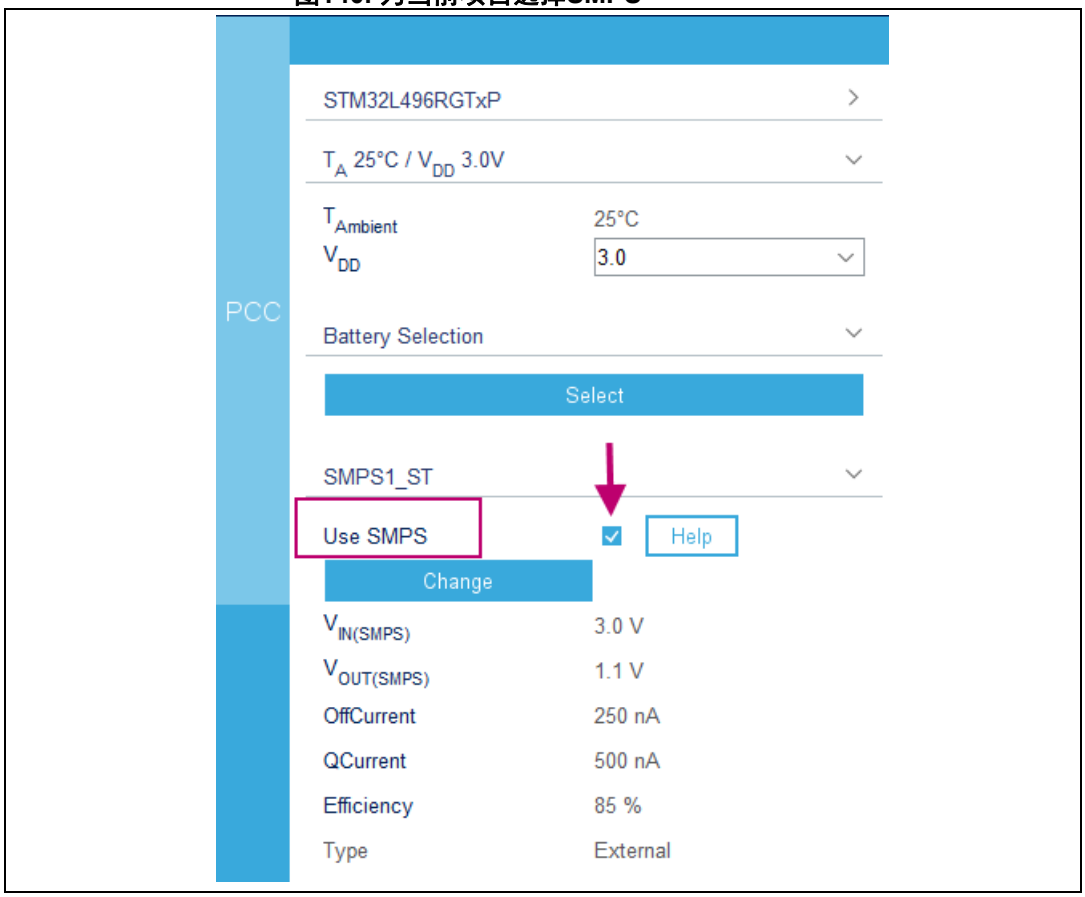


图141. SMPS数据库 - 添加新的SMPS模型

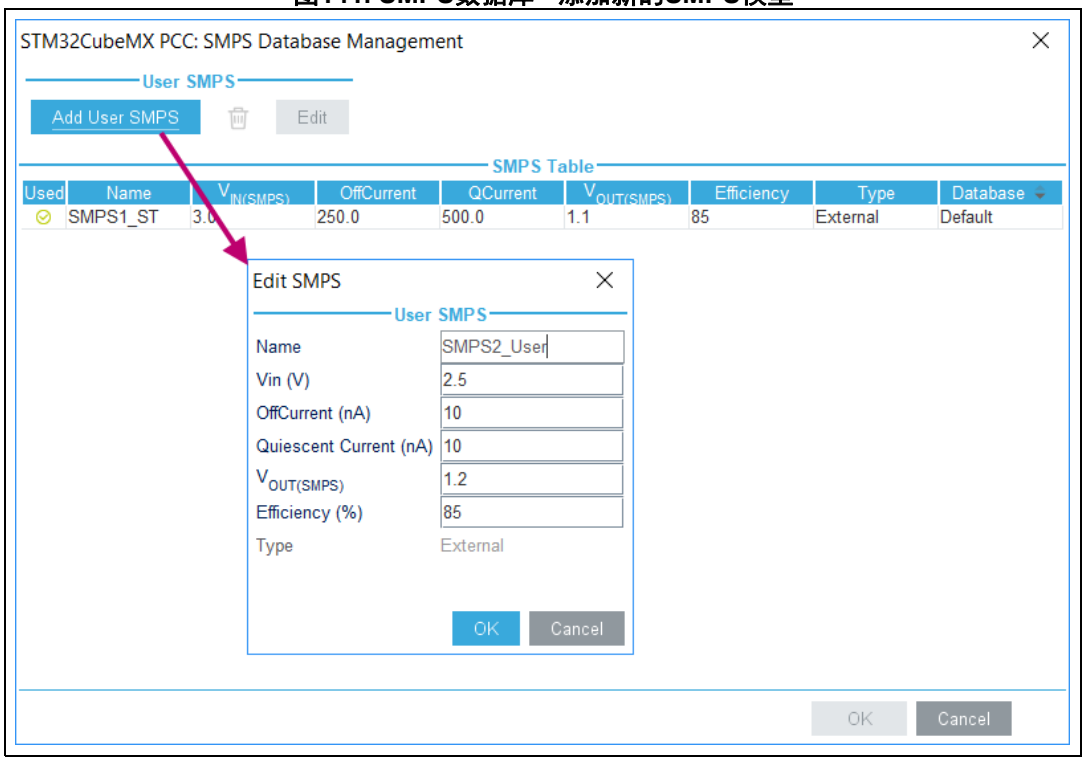


图142. SMPS数据库 - 选择不同的SMPS模型

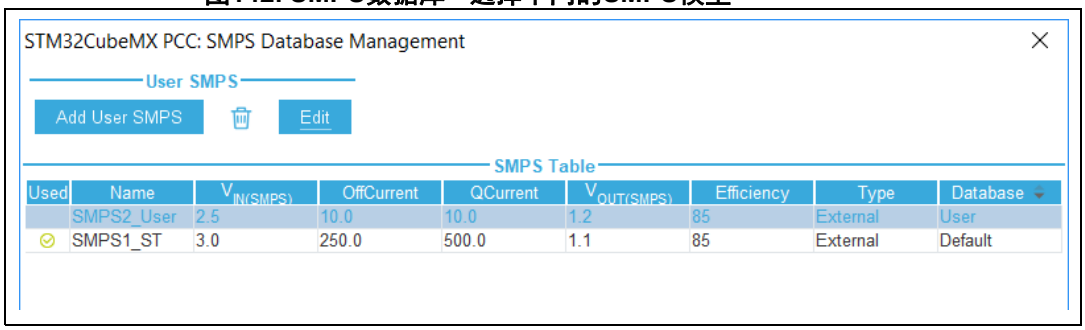


图143. 使用新的SMPS模型更新当前项目配置

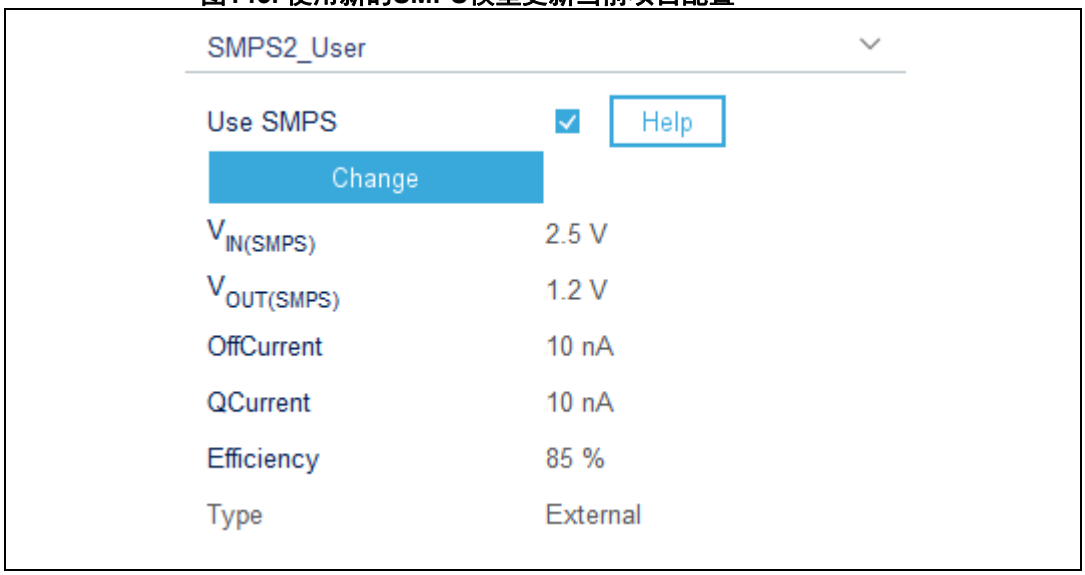


图144. 已选择新模型的SMPS数据库管理窗口

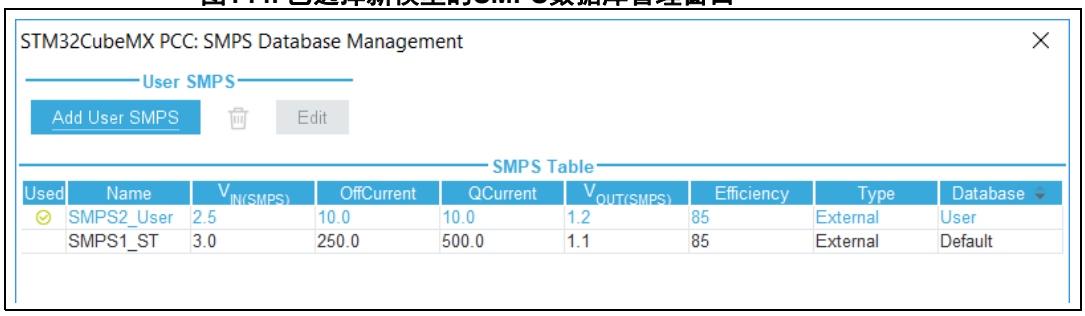


图145. SMPS转换检查器和状态图助手窗口

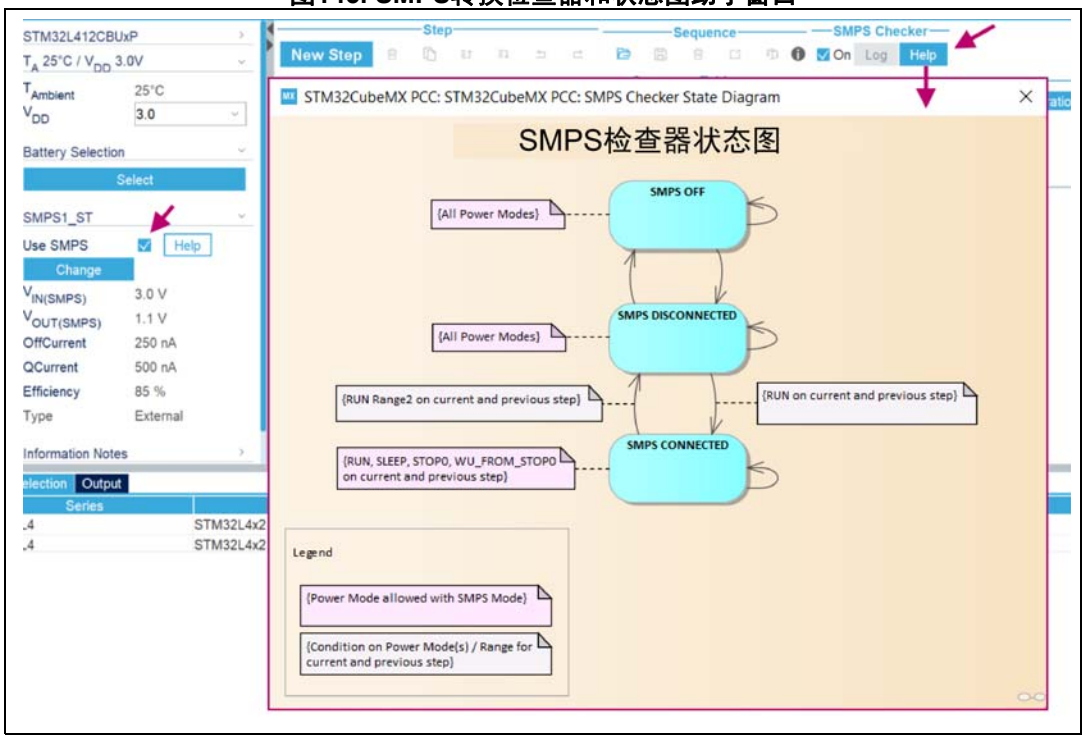
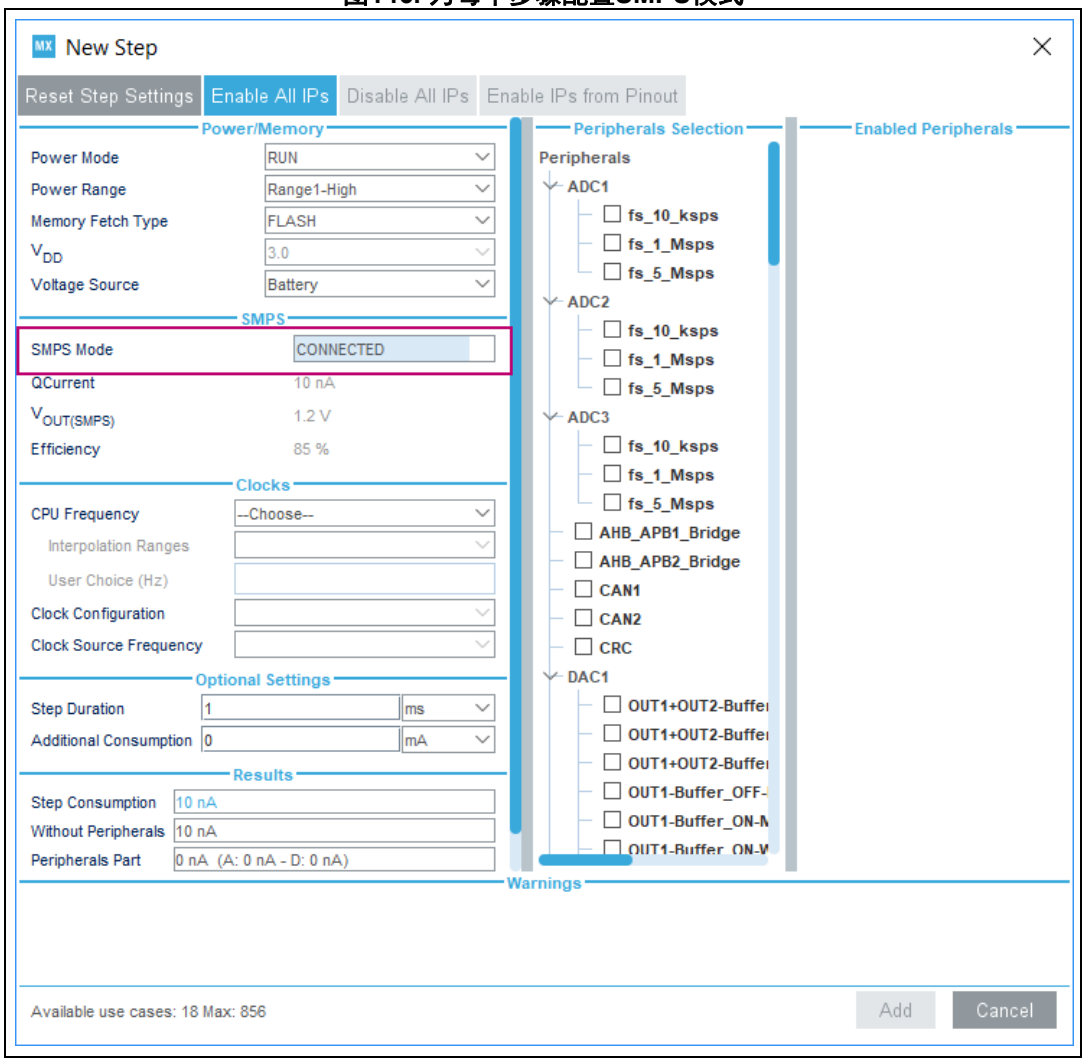


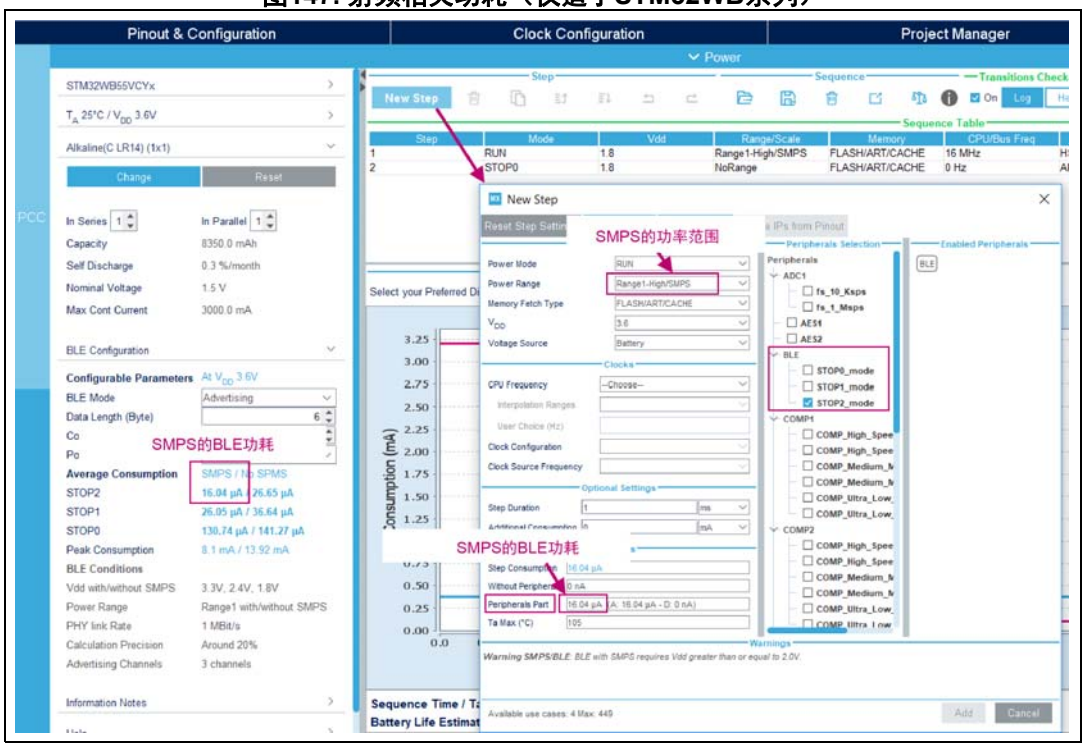
图146. 为每个步骤配置SMPS模式



5.1.7 支持BLE（仅适于STM32WB系列）

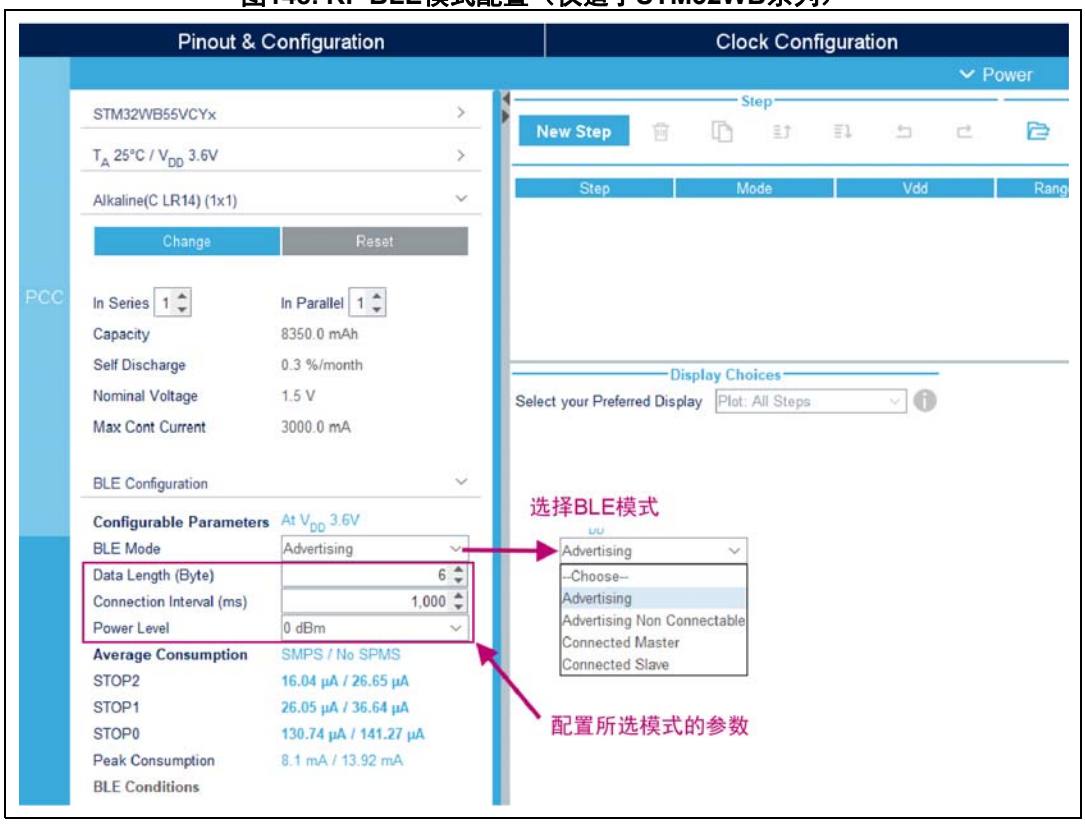
功耗工具允许用户将与RF外设和相应BLE功能模式有关的功耗，以及SMPS功能的使用考虑在内。

图147. 射频相关功耗（仅适于STM32WB系列）



可以从左侧面板选择BLE模式，并将其配置成反映用户的应用程序相关设置。

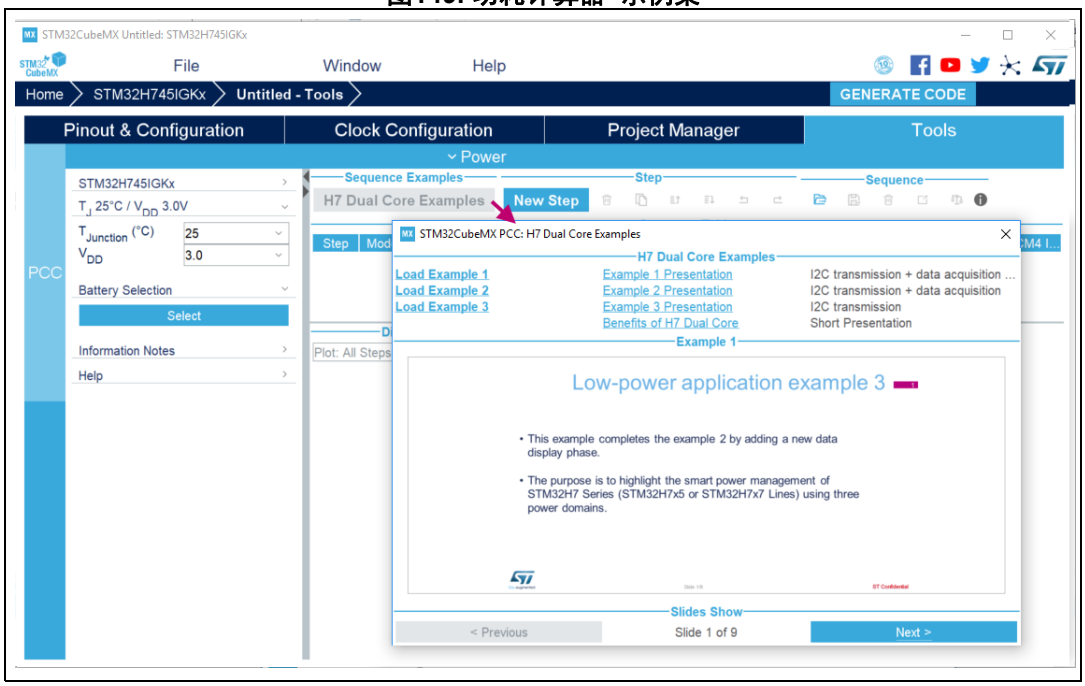
图148. RF BLE模式配置（仅适于STM32WB系列）



5.1.8 功能示例（仅适于STM32MP1和STM32H7双核）

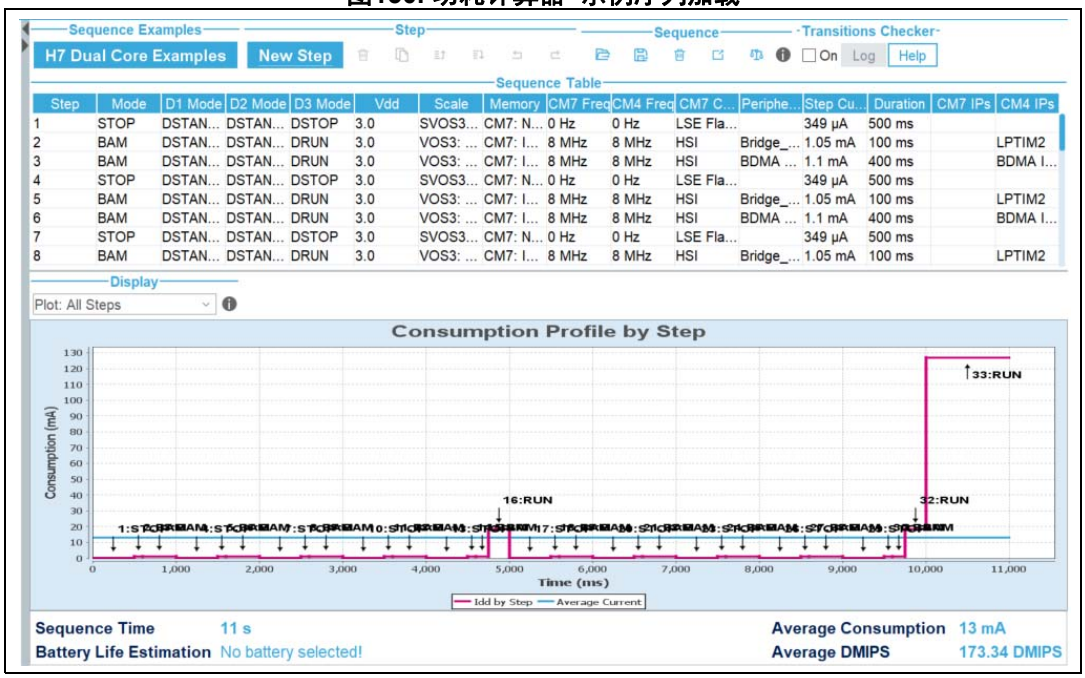
在“序列示例”部分下，PCC工具允许访问示例：每个示例都具有说明性的幻灯片组，以及可加载到PCC中的现成的序列（请参阅*图 149*）。

图149. 功耗计算器–示例集



单击“加载示例N”将加载与示例N对应的序列（请参阅*图 150*）。

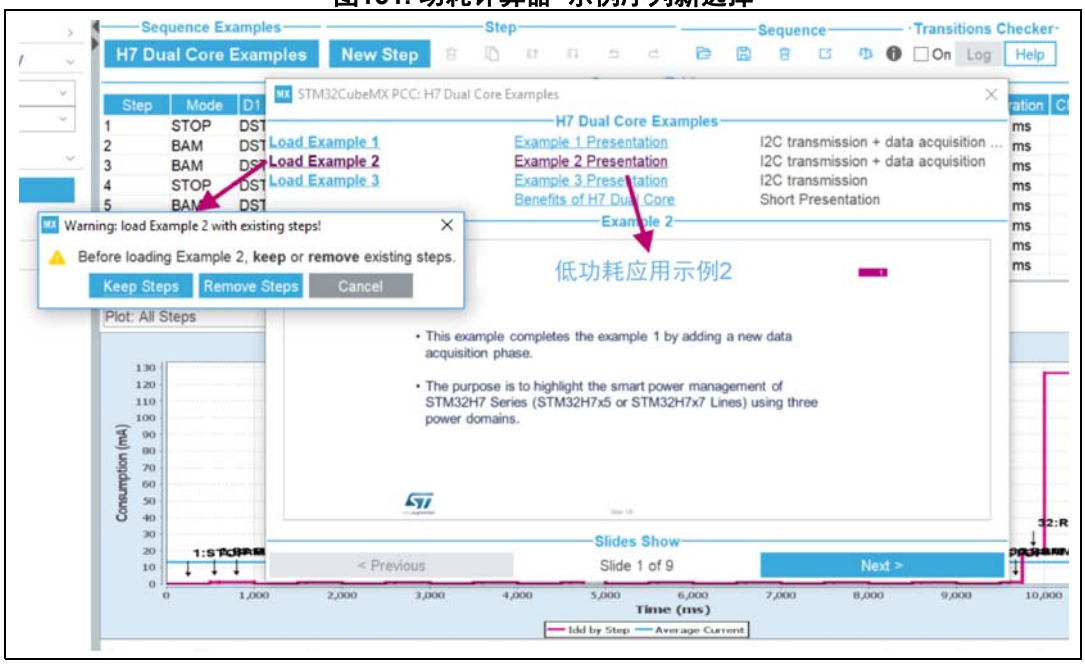
图150. 功耗计算器–示例序列加载



单击“示例N演示”将显示该示例的说明。

示例可以随时更改，新序列可以添加到或替换当前序列（参见*图 151*）。

图151. 功耗计算器–示例序列新选择



注： *这些示例用于给定的料号，当用于其他料号时可能需要进行调整。而且，加载后，建议编辑每个步骤并检查设置。*

5.2 DDR套件（仅适于STM32MP1系列）

DDR SDRAM属于高速精密器件，需要精心进行PCB设计。

STM32MP15器件支持以下DDR类型：

- LPDDR2
- LPDDR3
- DDR3 / DDR3L

它们由JEDEC标准（接口、命令、时序、封装和焊球布局的标准化）规定。

STM32CubeMX已进行扩展，可为STM32MP1 DDR子系统提供详尽的工具套件。它具有以下主要功能。

- **DDR控制器和PHY寄存器的配置**可基于减少的可编辑参数集自动进行管理。
- **DDR测试**基于丰富的测试列表而提供。测试从基本测试到应力测试。用户还可以开发自己的测试。
- 可对字节通道延迟进行**DDR调整**，以补偿板设计的缺陷。

DDR配置可以像**“引脚布局和配置”**视图中的其他外设一样进行访问：从组件面板单击DDR，打开“模式和配置”面板。

可以从**“工具”**视图中获得**DDR测试套件的测试和调整**功能。

DDR套件基于两个重要概念：

- **DDR时序**是DDR控制器和PHY配置的重要输入
- 调整DDR信号以补偿板设计的缺陷。

5.2.1 DDR 配置

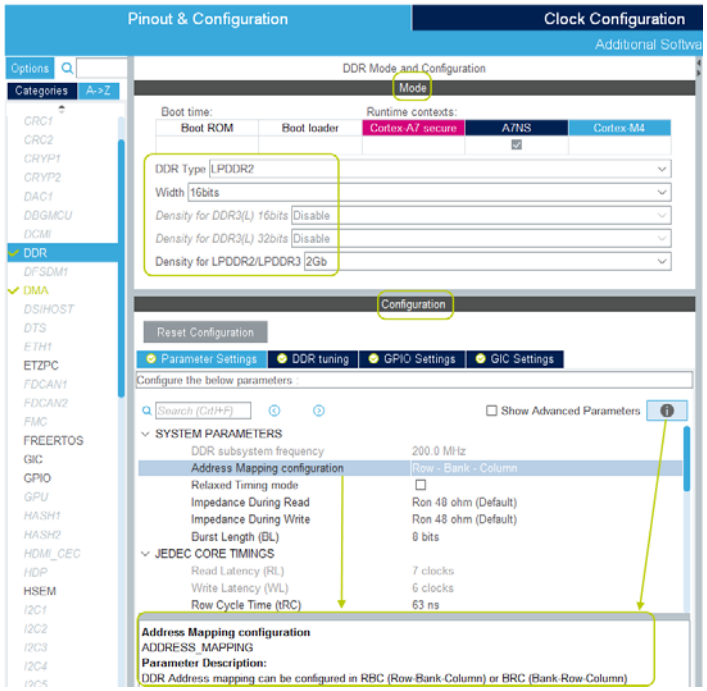
STM32CubeMX允许设置DDR系统参数和JEDEC内核时序。时序参数可在DDR数据手册中找到。

DDR类型、宽度和密度

进行DDR配置步骤时，需要对DDR类型、宽度和密度参数进行设置。在**“引脚布局和配置”**视图中选择DDR后，可以在“模式”面板中完成此操作。

有关LPDDR2设置的示例，请参阅*图 152*。

图152. DDR引脚布局和配置设置



另一个示例：对于具有两个“DDR3 16位2 Gb”芯片的配置，设置为“DDR3/DDR3L”、“32位”和4 Gb”。

注： *DDR IP的上下文无法更改，DDR与该工具中标识为“Cortex-A7 NS”的“Cortex-A7非安全”绑定在一起。*

DDR 配置

单击参数将在“DDR配置”页脚中显示其他详细信息。

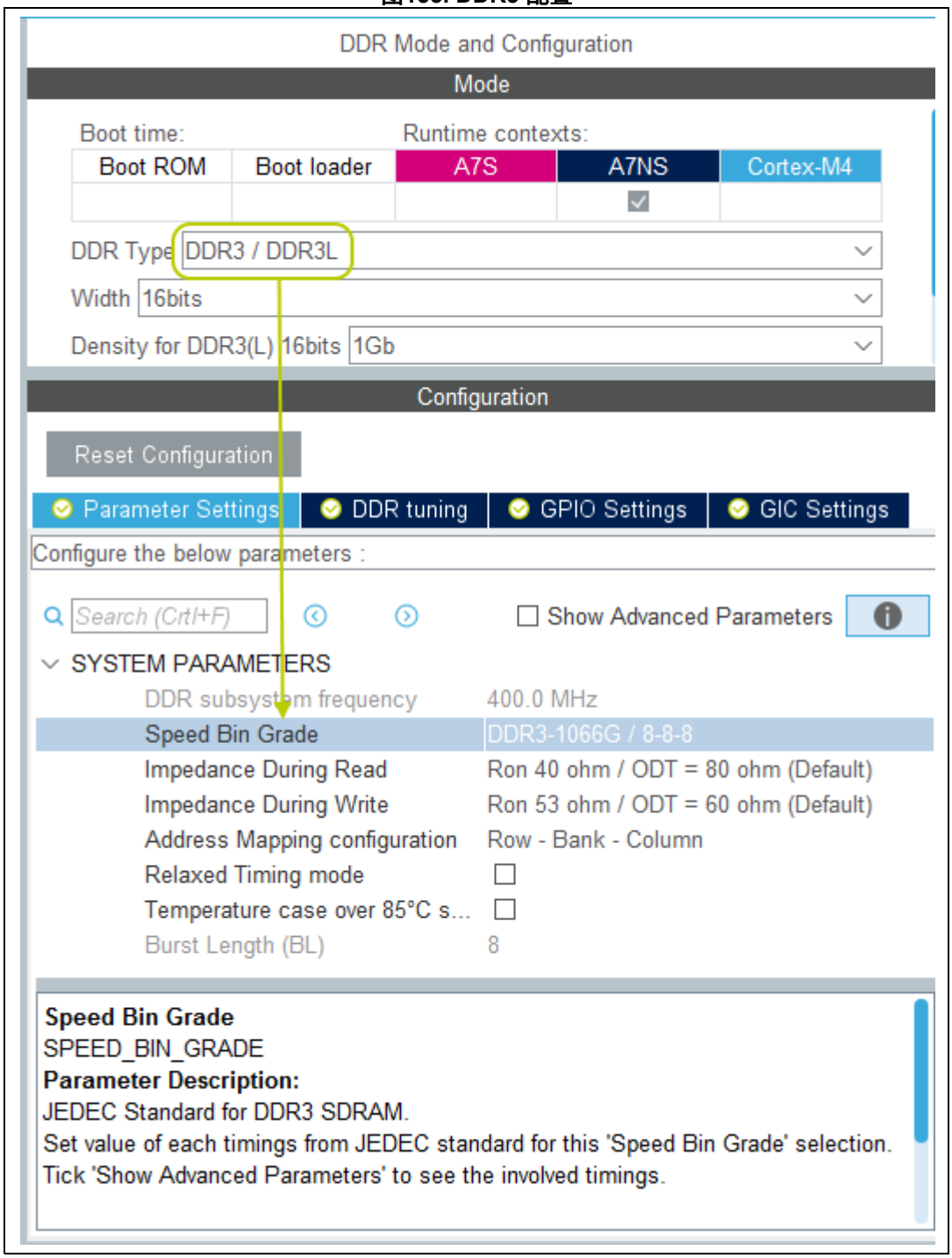
- DDR频率是从“时钟配置”选项卡中获取的，无法在DDR配置中更改。
- “休闲时序”模式在初启阶段用于尝试休闲密钥DDR DDR时序值（在t_{RC}、t_{RCD}和t_{RP}时序中添加一个t_{CK}）
- 其他参数须从用户DDR数据手册中进行检索。
- 一些参数是只读的：仅用于参考，并取决于DDR类型。

单击“生成代码”会基于这些参数自动计算器件树的DDR节点（DDR控制器和DDR PHY寄存器值）。

DDR3 配置

对于DDR3，通过选择**“速度档位/等级”**组合可以简化配置，无需手动编辑时序参数。

图153. DDR3 配置



“速度档位/等级”组合必须与所选的DDR相匹配。如果确切的组合不在选择列表中，须选择“1066E/6-6-6”以达到更快的DDR“速度档位/等级”，而“1066G/8-8-8”可以用作休闲配置。

然后，时序版本是可选的并保留给高级用户使用：选择“显示高级参数”以显示列表。

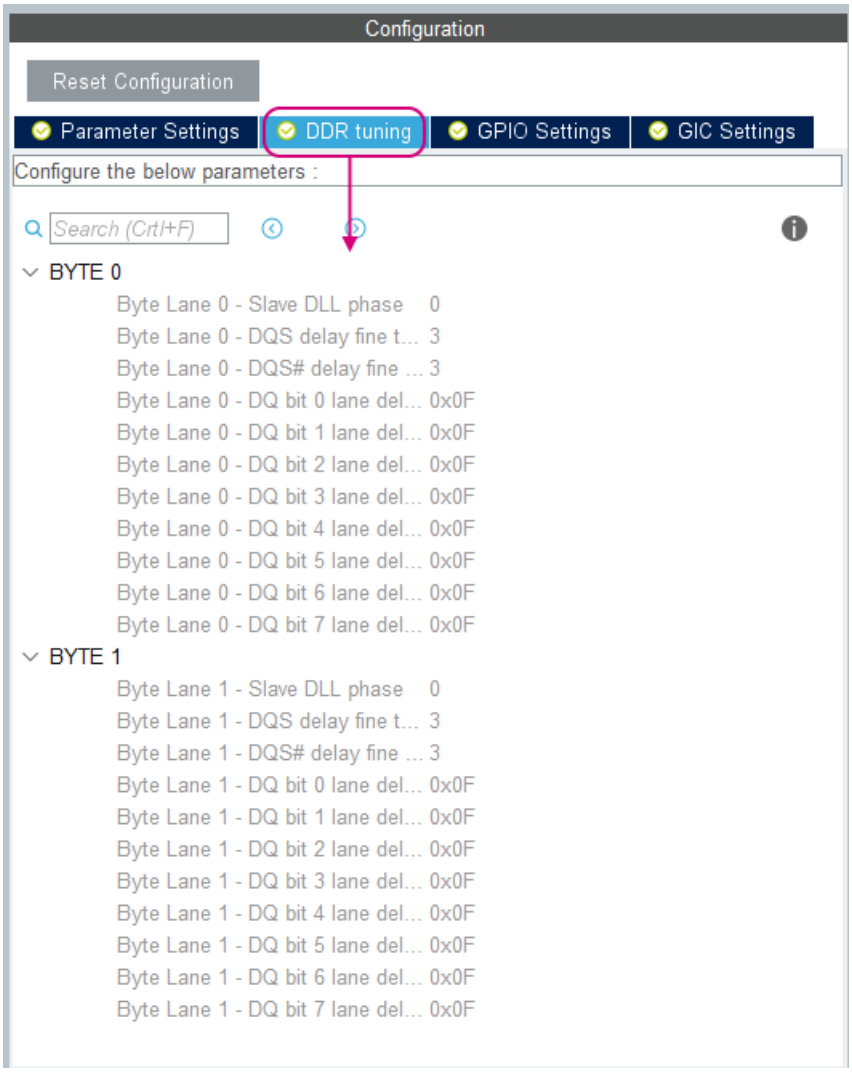
“DDR调整”选项卡（只读）

用户可以通过调谐选项卡检查对调整参数的修改。这些参数在DDR配置面板中是只读的（请参阅*图 154*），在调整操作之后进行了修改，并且与DQS位置和DQ线路延迟有关：

– “从属DDL阶段”，“DQS延迟微调”和“DQS＃延迟微调”定义了特定字节的DQS选通信号的位置。对于DQ线路眼图，该位置是最佳位置。

– “DQ x位通道延迟微调”定义了应用于特定字节的x位的延迟，用于补偿该特定位的潜在线长变化

图154. DDR调谐参数



5.2.2 连接目标和加载DDR寄存器

为了管理DDR测试和调谐，STM32CubeMX须使用**DDR交互协议**与目标尤其是与**U-Boot SPL**建立连接：

- DDR交互协议仅在**基本启动方案U-Boot SPL**二进制文件中可用，而且UART4外设实例支持该协议
- 当U-Boot SPL在UART4上检测到与STM32CubeMX的连接时，将停止其初始化过程并接受来自STM32CubeMX的命令。

有两个连接选项：

1. U-Boot SPL二进制文件在闪存中可用
2. 由于尚未对DDR进行测试或调整（因此尚未完全发挥作用），因此需要将U-Boot SPL加载到SYSRAM。

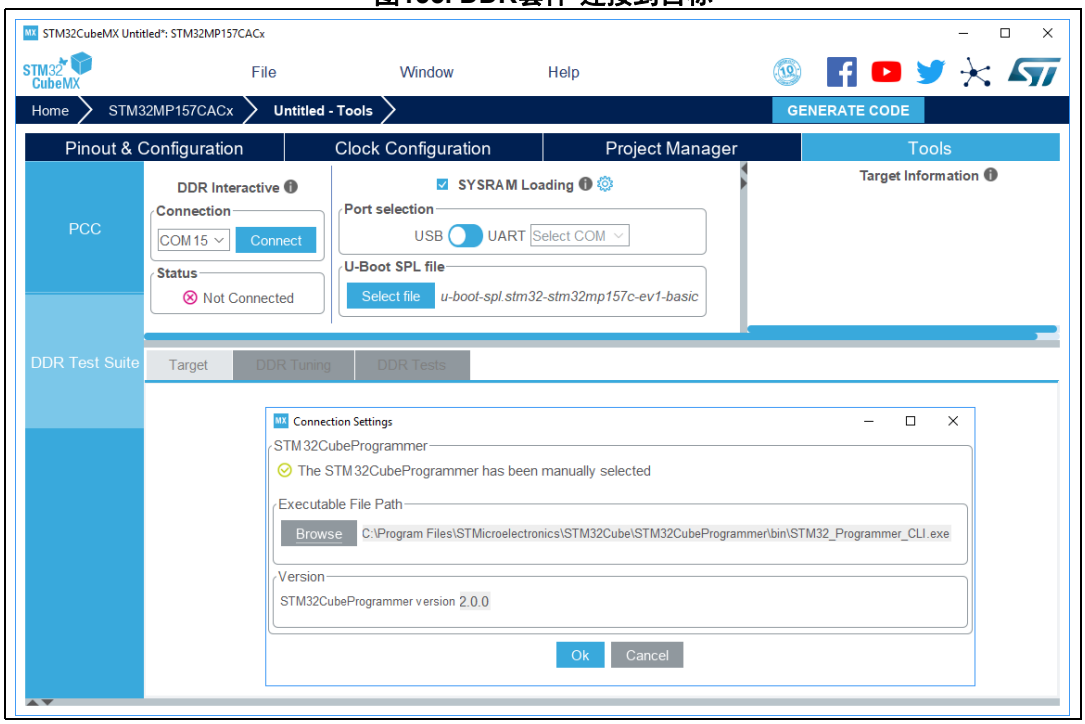
先决条件

- 安装ST-LinkUSB驱动程序以执行固件升级：对于Windows，必须使用最新版本的STSW-LINK009。对于Linux，必须使用STSW-LINK007驱动程序。二者均可从*www.st.com*下载。
- 安装STM32CubeProgrammer（仅适于SYSRAM加载）：可以从*www.st.com*下载安装程序。

连接到目标

必须选择COM端口以连接到目标，如*图 155*中所示。

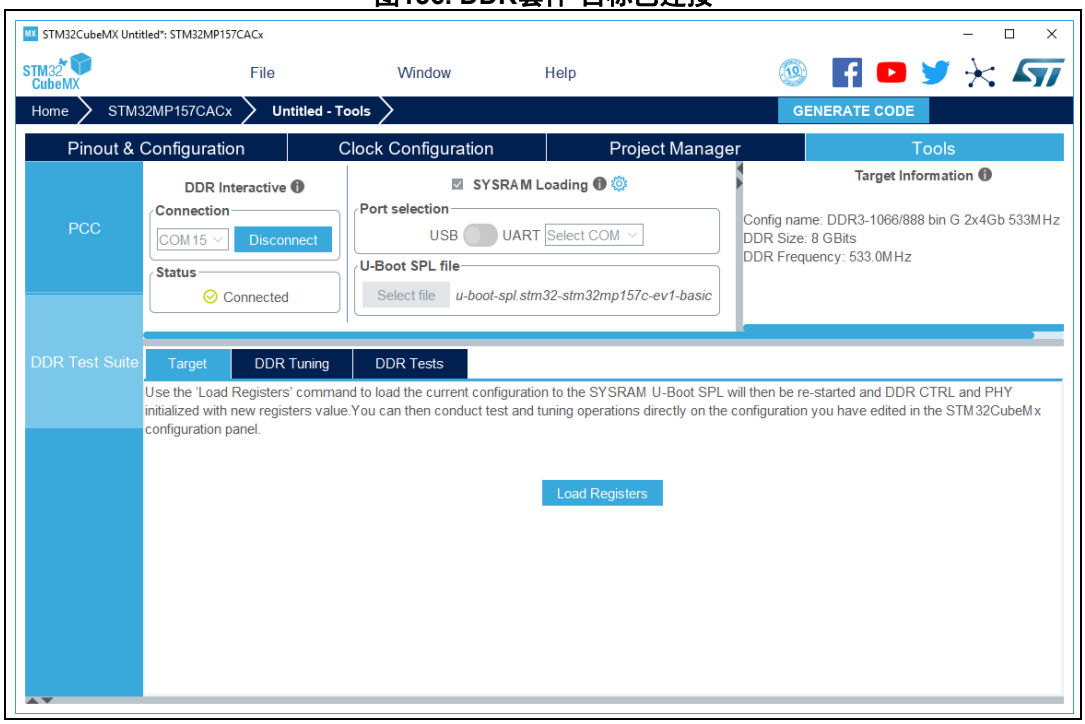
图155. DDR套件-连接到目标



如果需要在SysRAM中加载U-Boot SPL，可以使用STM32CubeProgrammer工具通过UART或USB进行。如果STM32CubeMX没有自动检测到，须在“连接设置”窗口中指定STM32CubeProgrammer工具的位置：单击 将其打开。须在构建映像文件夹中手动选择U-Boot SPL文件。

连接后，将提供各种服务和目标信息（请参阅*图 156*）。

图156. DDR套件-目标已连接



输出/日志消息

STM32CubeMX输出与DDR套件相关的活动日志（请参阅*图 157*）和交互协议通信日志（请参阅*图 158*）。这些通过启用“窗口”菜单中的输出进行显示。

图157. DDR活动日志

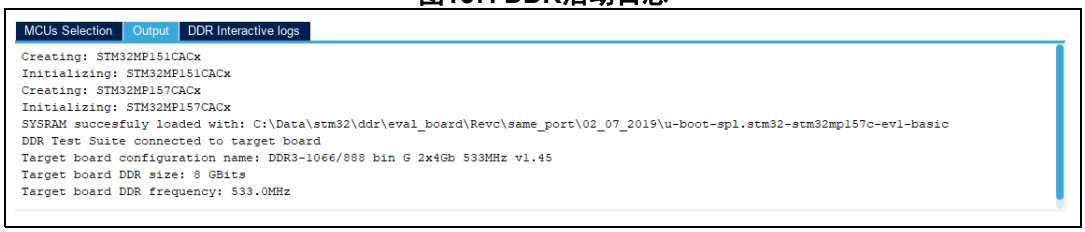
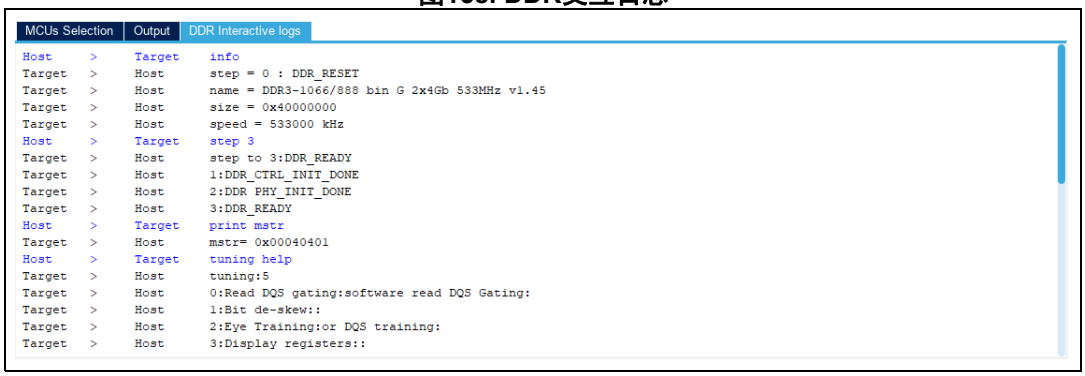


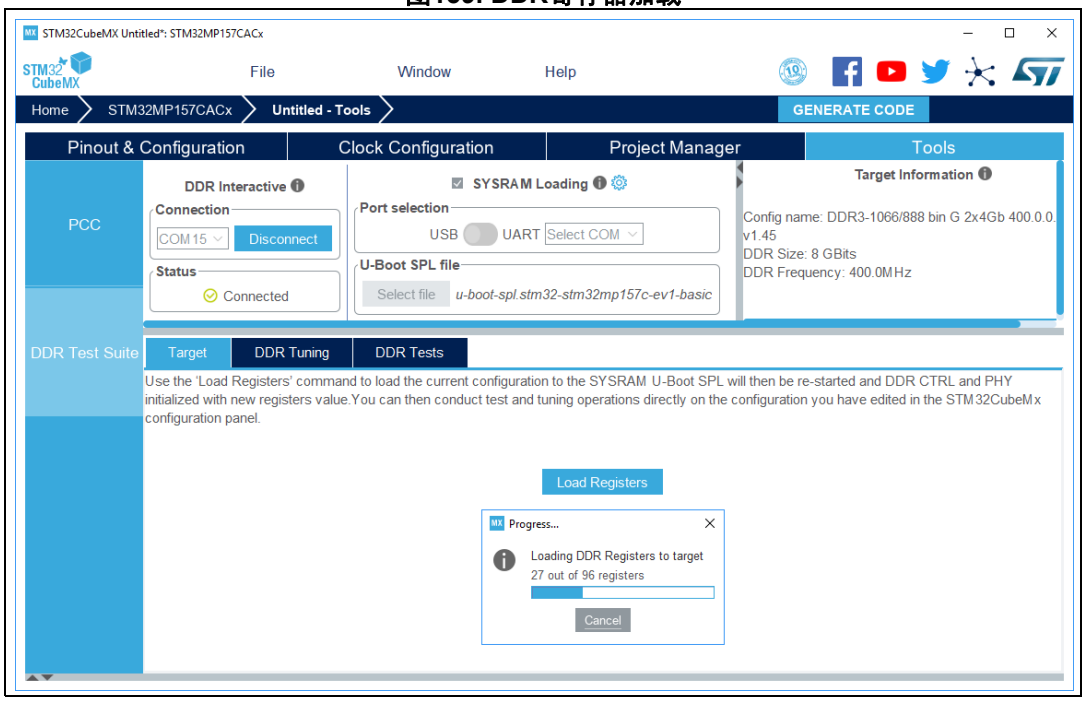
图158. DDR交互日志



DDR寄存器加载（可选）

一旦在DDR交互模式下连接，用户即可在SYSRAM中加载当前DDR配置。

图159. DDR寄存器加载



如果使用的U-Boot SPL已经包含所需的DDR配置，该步骤则是可选项。它使用这些寄存器触发DDR控制器和PHY初始化，还允许用户快速测试配置，无需生成设备树和专用的U-Boot SPL二进制文件。

5.2.3 DDR测试

先决条件

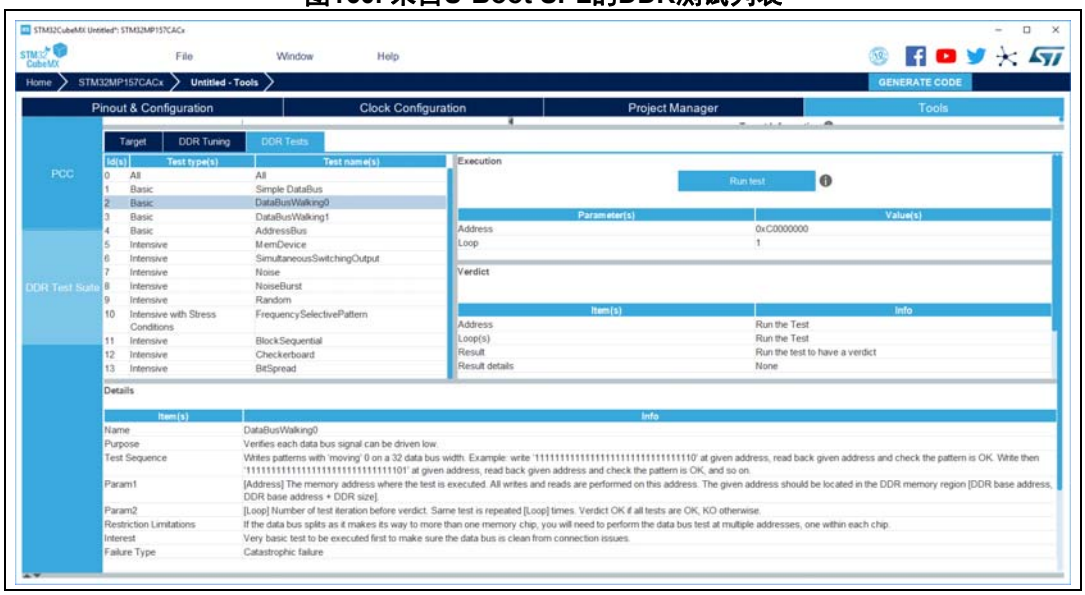
要进行DDR测试：

- DDR套件须处于连接状态
- DDR配置须在存储器中可用，既可以和U-Boot SPL（在器件树中带有DDR寄存器文件）一起，也可以在DDR寄存器中使用（请参阅*第 5.2.2 节*）。

DDR测试列表

DDR测试是U-Boot SPL的一部分（请参阅*图 160*）。

图160. 来自U-Boot SPL的DDR测试列表



可以修改U-boot SPL来添加新测试。

大多数测试的参数都是在执行之前设置的，例如：

- 地址：执行测试的存储器地址。所有写入和读取均在该地址上执行。给定的地址须位于DDR存储器区域中[DDR基址，DDR基址+ DDR大小]。
- 在STM32MP15上，DDR基址为0xC0000000（例如，4 GB的DDR大小为0x20000000）。
- 循环：判决前的测试迭代次数。同一测试重复[循环]次。如果所有测试都通过，判决则为OK，否则为KO。
- 大小：待测试区域的字节大小。大小必须为4的倍数（对32位无符号整数执行读/写操作）且最小值等于4。大小最多可为DDR大小。
- 模式：用于读取/写入操作的32位模式。

DDR套件内置自动校正功能，可防止用户指定错误的值。

进行所有测试时，“数据缓存”处于禁用状态，“指令缓存”处于启用状态。

DDR测试结果

测试判决由U-Boot SPL报告：重新调用测试所用的参数以及“通过/失败”状态和结果详细信息（请参阅*图 161*）。测试历史可在“输出”和“日志”面板中找到（请参阅*图 162*）。

图161. DDR测试套件结果

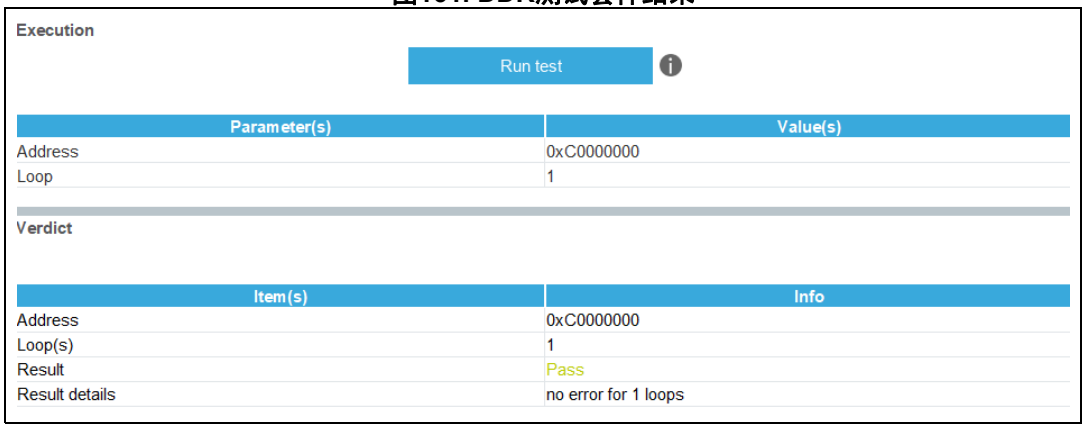
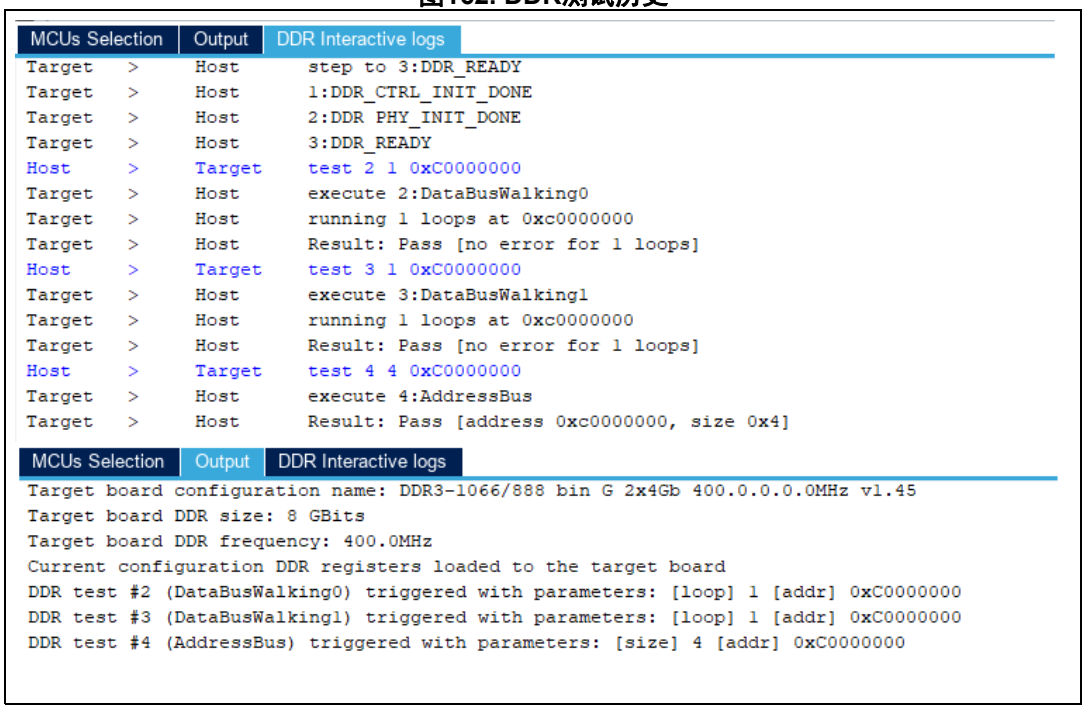


图162. DDR测试历史



5.2.4 DDR调整

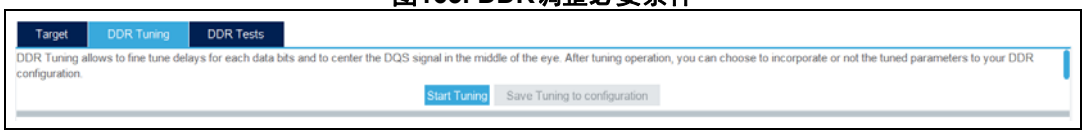
先决条件

进行DDR调整的必要条件如下：

- DDR套件处于连接状态
- 有效DDR配置在存储器中可用，既可以和U-Boot SPL（通过设备树中的DDR寄存器文件）一起，也可以在DDR寄存器中（请参阅*DDR寄存器加载（可选）*）。

由于进行了DDR调整，因此可以补偿硬件设计的不完善之处，以实现最佳操作（有关DDR设计布线指南，请参见*www.st.com*上的AN5122）。

图163. DDR调整必要条件



可调信号

可调信号包括：

- DQS信号：每个数据字节的位置
- 8个DQ位：每个数据字节的延迟。

一些DDR寄存器专用于存储相应的调谐后的设置：

- “从属DDL阶段”，“DQS延迟微调”和“DQS＃延迟微调”定义了特定字节的DQS选通信号的位置：该位置是有关DQ线路眼图的最佳位置
- “DQ x位通道延迟微调”定义了应用于特定字节的x位的延迟，用于补偿该特定位的潜在线长变化。

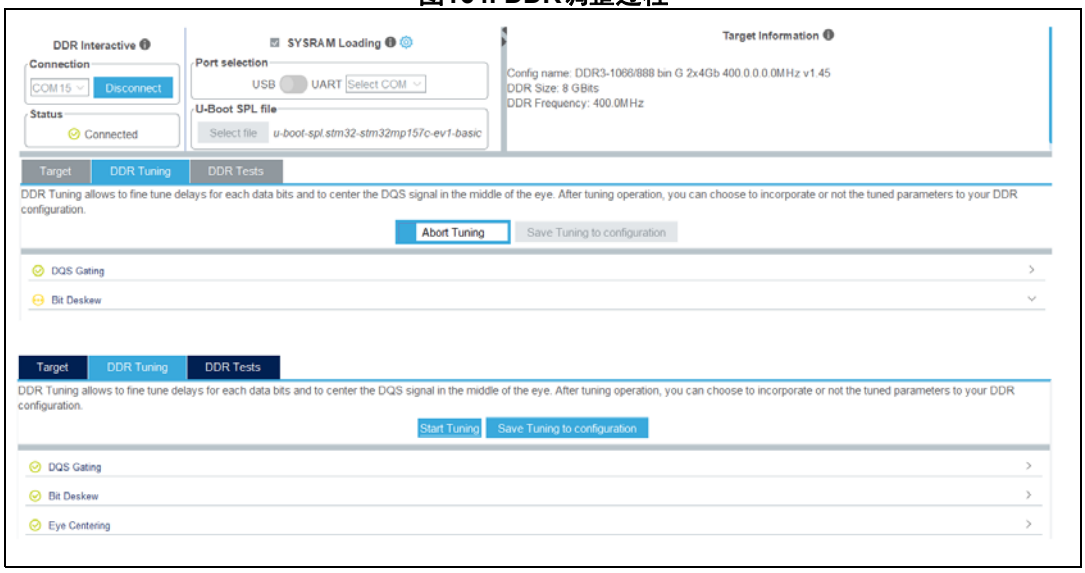
注： *建议在多个板上进线路调整，以确保调整后的参数变化受到限制。*

调整过程

调整是通过三个连续的步骤完成的（请参阅*图 164*）：

1. DQS门控
2. 位去时滞
3. 眼图训练

图164. DDR调整过程



位去时滞

“位去时滞”面板（请参阅*图 165*）图示了

- 给定字节的最佳DQS信号位置，以便调整DQ线路延迟
- 用于所考虑字节的每个DQ线路的延迟。单位延迟值为20.56 ps。有四个步骤。因此，位通道延迟的调谐范围是0-61.68 ps。

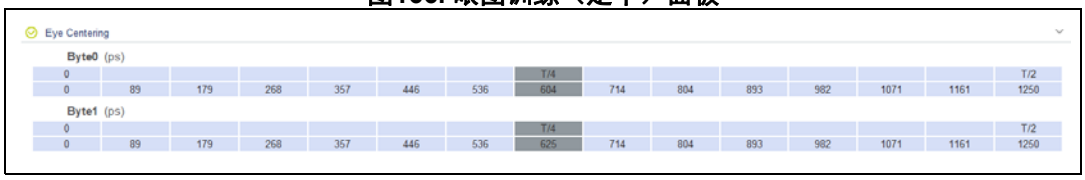
图165. 位去时滞

眼图训练（定中）

眼图训练（定中）面板（*图 166*）给出了DQS信号在每个字节的半周期内的最终最佳位置：

- DQS位置从36度到144度粗略变化（四分之一周期为90度）
- 然后，DQS位置围绕粗略位置以8步在-61.68 ps到+82.24 ps的范围内精细变化

图166. 眼图训练（定中）面板



传播调整结果

调整完成后，DDR套件允许用户将调整后的参数传播到当前DDR配置（请参阅*图 167*）。“DDR调整”选项卡将相应刷新（请参阅*图 168*）。

图167. DDR调整-保存到配置

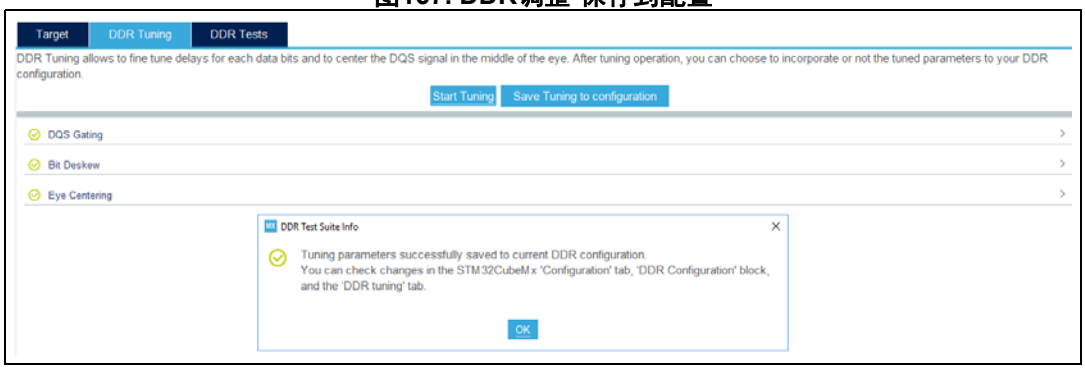
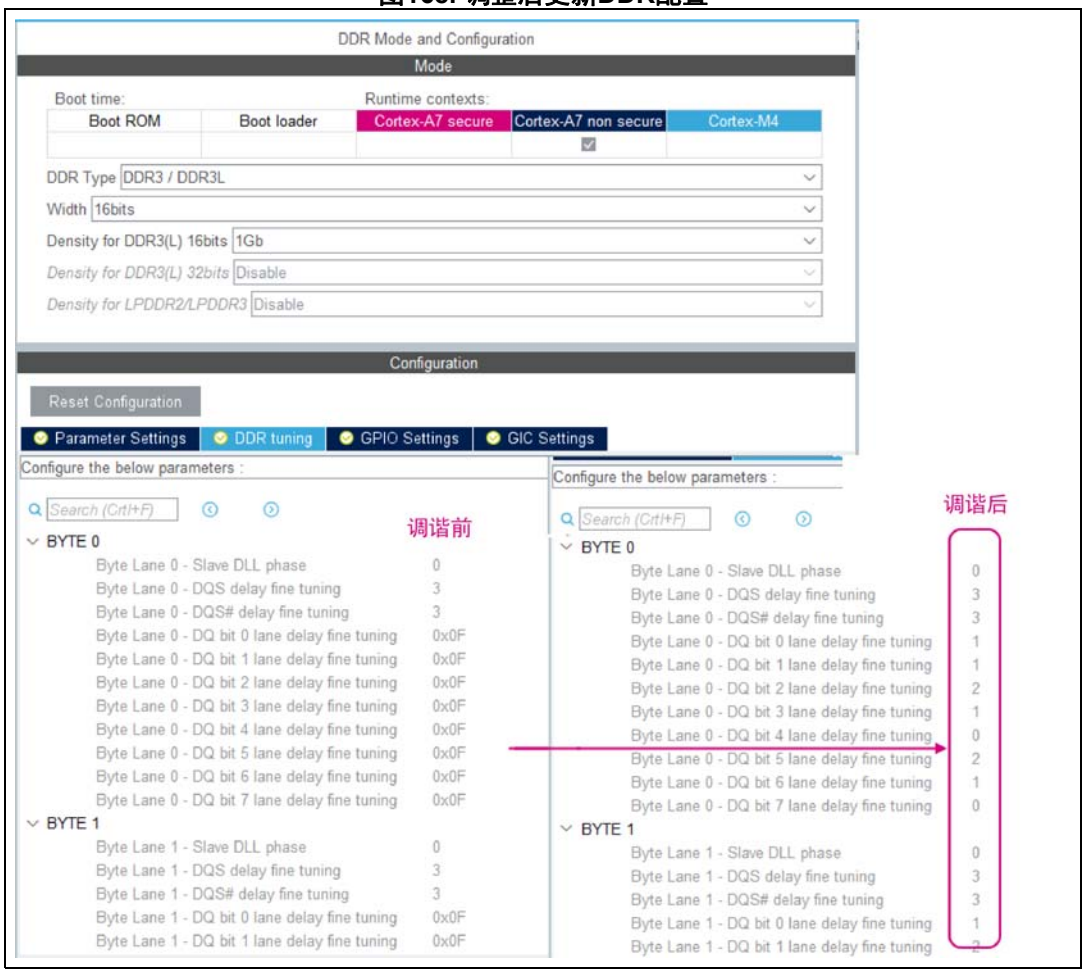


图168. 调整后更新DDR配置



6 STM32CubeMXC代码生成概述

6.1 仅使用HAL驱动程序生成STM32Cube代码（默认模式）

在C代码生成过程中，STM32CubeMX执行以下操作：

1. 如果缺失，将从用户存储库下载相关的STM32Cube MCU包。STM32CubeMX存储库文件夹在**帮助 > 更新程序设置**菜单中指定。
2. 它从固件包以及*Drivers/CMSIS*、*Drivers/STM32F4_HAL_Driver*文件夹和*中间件*文件夹中（如果选择了中间件）的相关文件中复制。
3. 它生成与用户MCU配置相对应的初始化C代码（.c/.h文件），并将其存储在*Inc*和*Src*文件夹中。默认情况下，包含以下文件：
   - **stm32f4xx_hal_conf.h**文件：此文件定义了使能的HAL模块，并将一些参数（如外部高速振荡器频率）设为预定义的默认值，或根据用户配置（时钟树）进行设置。
   - **stm32f4xx_hal_msp.c**（MSP=MCU支持包）：此文件定义了所有初始化函数，以便根据用户配置（引脚分配、使能时钟、使用DMA和中断）配置外设实例。
   - **main.c**负责：

     通过调用重置所有外设、初始化闪存接口和SysTick的*HAL_init()*函数将MCU重置为已知状态。

     配置和初始化系统时钟。

     配置和初始化外设未使用的GPIO。

     为每个已配置的外设定义和调用外设初始化函数，该函数定义了将传递给相应外设*HAL init*函数（转而调用外设HAL MSP初始化函数）的句柄结构。请注意，当使用LwIP（各自的USB）中间件时，底层以太网（各自的USB外设）的初始化C代码将从main.c移至LwIP（各自的USB）初始化C代码本身。
   - **main.h**文件：

     此文件包含与通过**引脚排列**选项卡所设置的引脚标签相对应的定义语句，以及通过**配置**选项卡所添加的用户项目常量（有关示例，请参见*图 169*与*图 170*）：

```
#define MyTimeOut          10
#define LD4_Pin            GPIO_PIN_12
#define LD4_GPIO_Port      GPIOD
#define LD3_Pin            GPIO_PIN_13
#define LD3_GPIO_Port      GPIOD
#define LD5_Pin            GPIO_PIN_14
#define LD5_GPIO_Port      GPIOD
#define LD6_Pin            GPIO_PIN_15
#define LD6_GPIO_Port      GPIOD
```

图169. 生成定义语句的引脚标签

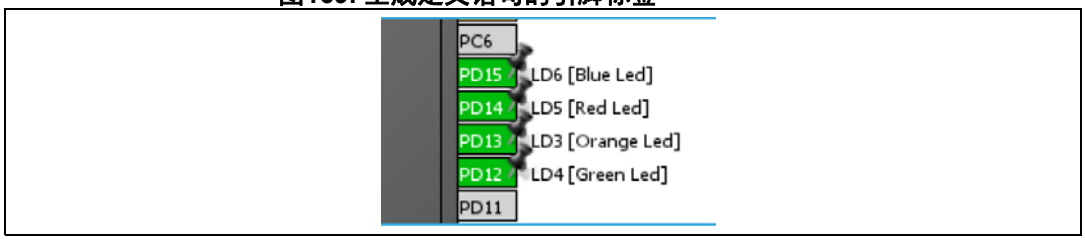
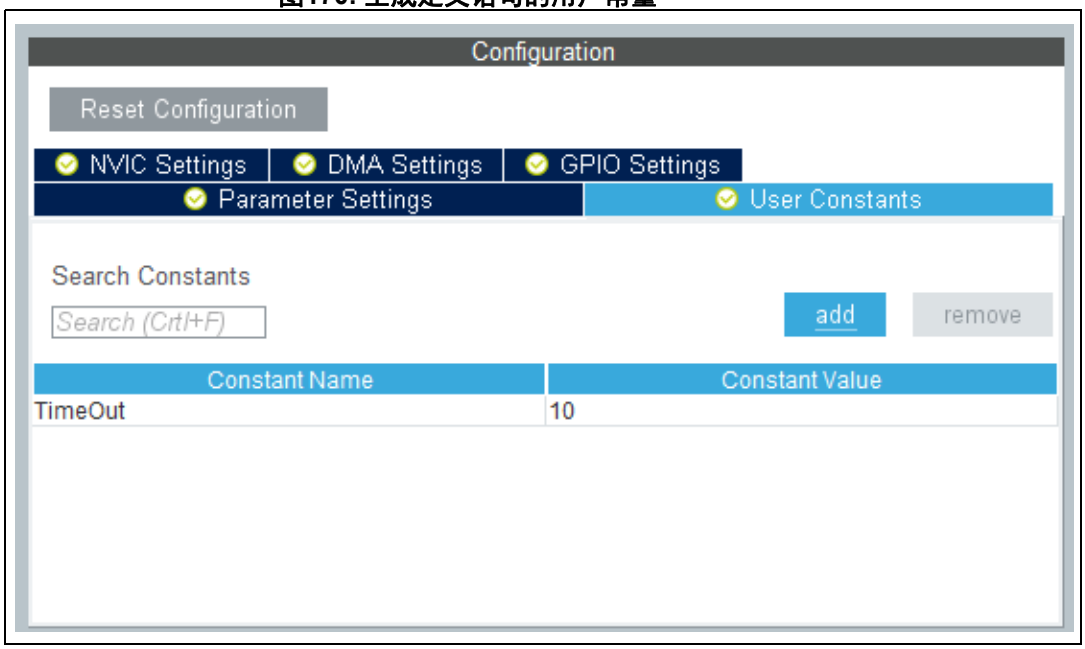


图170. 生成定义语句的用户常量

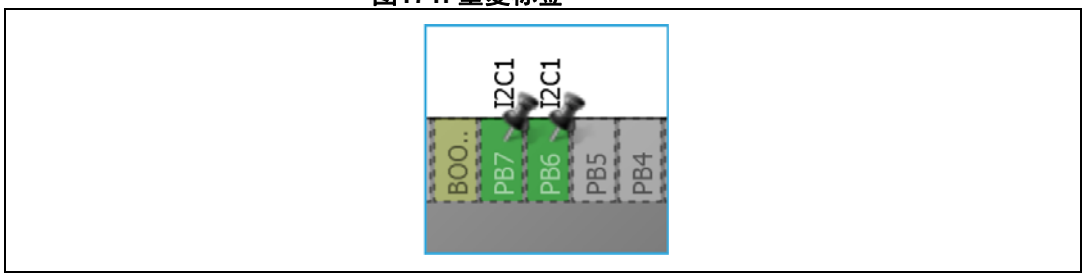


如果出现重复的标签，则会添加由引脚端口字母与引脚索引号组成的唯一后缀，并将其用于产生相关定义语句。

在*图 171*中所显示的重复I2C1标签示例中，通过代码生成功能产生了以下代码，保留原始端口B引脚6定义语句上的I2C1标签，并在引脚7定义语句上添加B7后缀：

```
#define I2C1_Pin              GPIO_PIN_6
#define I2C1_GPIO_Port        GPIOB
#define I2C1B7_Pin            GPIO_PIN_7
#define I2C1B7_GPIO_Port      GPIOB
```

图171. 重复标签



为了编译生成的项目，定义语句应遵守严格的命名规则。它们应以字母或下划线以及相应的标签开始。此外，它们不应包括任何特殊字符，如减号、圆括号或方括号。标签中的任何特殊字符均将被自动替换为定义名称中的下划线。

如果标签包含“[]”或“()”之间的字符串，则仅将列出的第一个字符串用于定义名称。例如，标签“**LD6** [Blue Led]”对应于以下定义语句：

```
#define LD6_Pin   GPIO_PIN_15
#define LD6_GPIO_Port   GPIOD
```

定义语句用于配置生成的初始化代码中的GPIO。在以下示例中，使用相应的定义语句对标记为*Audio_RST_Pin*和*LD4_Pin*的引脚进行初始化：

```
/*配置GPIO引脚：LD4_Pin Audio_RST_Pin */
GPIO_InitStruct.Pin = LD4_Pin | Audio_RST_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_LOW;
HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);
```

4. 最后将生成*项目*文件夹，其中包含与用户项目设置匹配的工具链特定文件。双击IDE特定项目文件将启动IDE并加载准备好编辑、构建和调试的项目。

6.2 使用底层驱动程序生成STM32Cube代码

对于除STM32H7和STM32MP1系列以外的所有STM32系列，STM32CubeMX允许用户基于外设HAL驱动程序或外设低层（LL）驱动程序生成外设初始化代码。

通过“项目管理器”视图进行选择（请参阅*第 4.9.3节：“高级设置”选项卡*）。

LL驱动仅适用于需要优化访问且没有复杂软件配置的外设。LL服务允许通过更改相关外设寄存器内容来执行原子操作：

- 支持的外设示例：RCC、ADC、GPIO、I2C、SPI、TIM、USART等
- LL驱动不支持的外设示例：USB、SDMMC、FSMC。

LL驱动在STM32CubeL4包中提供：

- 它们位于*STM32Cube_FW_L4_V1.6\Drivers\STM32L4xx_HAL_Driver*文件夹的*Inc*和*Src*目录中的HAL驱动(**stm32l4_hal_<peripheral_name>**)旁边。
- 通过其命名规则可轻松识别它们：**stm32l4_ll_<peripheral_name>**

有关HAL和LL驱动的更多信息，请参见*STM32L4 HAL和底层驱动*用户手册（UM1884）。

因为使用LL还是HAL驱动是取决于外设，所以用户可以在同一项目中混合使用HAL和LL驱动。

下表显示了三种可能的STM32CubeMX项目生成选项之间的主要区别：仅限HAL、仅限LL、HAL和LL混合代码。

表18. LL与HAL代码生成：STM32CubeMX项目中包含的驱动

要包含的项目配置和驱动	仅限HAL	仅限LL	混合HAL和LL	注释
CMSIS	有	有	有	-
STM32xxx_HAL_Driver	仅HAL驱动文件	仅LL驱动文件	HAL和LL混合驱动文件	如果将项目设置选项设为“仅复制必要的文件”，则只复制给定配置（外设选择）所需的驱动文件。否则（“所有使用的库”选项），将复制一系列完整的驱动文件。

表19. LL与HAL代码生成：STM32CubeMX生成的头文件

已生成头文件	仅限HAL	仅限LL	HAL和LL混合	注释
main.h	有	有	有	此文件包含include语句和为用户常量（GPIO标签和用户常量）生成的定义语句。
stm32xxx_hal_conf.h	有	无	有	此文件使能项目所需的HAL模块。
stm32xxx_it.h	有	有	有	中断处理程序的头文件
stm32xx_assert.h	无	有	有	该文件包含断言宏和用于检查函数参数的函数。

表20. LL与HAL：STM32CubeMX生成的源文件

已生成源文件	仅限HAL	仅限LL	HAL和LL混合	注释
main.c	有	有	有	该文件包含主函数和可选的STM32CubeMX生成函数。
stm32xxx_hal_msp.c	有	无	有	该文件包含以下功能： – HAL_MspInit – 对于使用HAL驱动的外设： HAL_<Peripheral>_MspInit、 HAL_<Peripheral>_MspDeInit， 这些函数仅适用于使用HAL驱动的外设。
stm32xxx_it.c	有	有	有	中断处理程序的源文件

表21. LL与HAL：STM32CubeMX生成函数与函数调用

生成的源文件	仅限HAL	仅限LL	HAL和LL混合	注释
Hal_init()	在main.c中调用	未使用	在main.c中调用	此文件执行以下功能： – 闪存预取及指令数据缓存的配置 – 选择SysTick计时器作为时基源 – 设置NVIC组优先级 – MCU低级初始化。
Hal_msp_init()	在stm32xxx_hal_msp.c中生成并被HAL_init()调用	未使用	在stm32xxx_hal_msp.c中生成 并被HAL_init()调用	此函数执行外设资源配置(1)。
MX_<Peripheral>_Init()	[1]：外设配置和调用HAL_<Peripheral>_Init()	[2]：使用LL函数配置外设和外设资源配置(1) 调用LL_Peripheral_Init()	– 当为<外设>选择HAL驱动时，将执行以下函数生成和调用*[1]：外设配置和调用HAL_<Peripheral>_Init()* – 当为<外设>选择LL驱动时，将执行以下函数生成和调用*[2]：使用LL函数配置外设和外设资源配置*	该文件负责外设配置。 当为<外设>选择LL驱动时，还执行外设资源配置(1)。
HAL_<Peripheral>_MspInit()	[3]：当为<外设>选择HAL驱动时，在stm32xxx_hal_msp.c中生成	未使用	只能为<外设>选择HAL驱动：执行以下函数生成和调用*[3]：当为<外设>选择HAL驱动时，在stm32xxx_hal_msp.c中生成*	外设资源配置(1)
HAL_<Peripheral>_MspDeInit()	[4]：当为<外设>选择HAL驱动时，在stm32xxx_hal_msp.c中生成	未使用	只能为<外设>选择HAL驱动：执行以下函数生成和调用*[4]：当为<外设>选择HAL驱动时，在stm32xxx_hal_msp.c中生成*	此函数只能用于释放外设资源。

1. 外设资源包括：
   - 外设时钟
   - 引脚布局配置（GPIO）
   - 外设DMA请求
   - 外设中断请求和优先级。

图172. 基于HAL的外设初始化：usart.c代码片段

```
基于HAL初始化USART外设
void MX_USART1_UART_Init(void)
{                                                    外设配置
  huart1.Instance = USART1;
  huart1.Init.BaudRate = 115200;
  huart1.Init.WordLength = UART_WORDLENGTH_7B;
  huart1.Init.StopBits = UART_STOPBITS_1;
  ...
  if (HAL_UART_Init(&huart1) != HAL_OK)
  {
    Error_Handler();
  }
}
void HAL_UART_MspInit(UART_HandleTypeDef* uartHandle)
{                                                    外设资源配置
  GPIO_InitTypeDef GPIO_InitStruct;
  if(uartHandle->Instance==USART1)
  {
    /* Peripheral clock enable */
    __HAL_RCC_USART1_CLK_ENABLE();
    /* USART1 GPIO Configuration */
    GPIO_InitStruct.Pin = GPIO_PIN_10;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull = GPIO_PULLUP;
    ...
    HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
  }
}
void HAL_UART_MspDeInit(UART_HandleTypeDef* uartHandle)
{                                                    外设资源释放
  if(uartHandle->Instance==USART1)
  {
     /* Peripheral clock disable */
    __HAL_RCC_USART1_CLK_DISABLE();
    /* USART1 GPIO Configuration */
    HAL_GPIO_DeInit(GPIOA, GPIO_PIN_10);
    HAL_GPIO_DeInit(GPIOB, GPIO_PIN_6);
  }
}
```

图173. 基于LL的外设初始化：usart.c代码片段

```
使用LL驱动程序初始化USART外设
void MX_USART1_UART_Init(void)
{
  LL_USART_InitTypeDef USART_InitStruct;
  LL_GPIO_InitTypeDef GPIO_InitStruct;
  /* Peripheral clock enable */
  LL_APB2_GRP1_EnableClock(LL_APB2_GRP1_PERIPH_USART1);

    /**USART1 GPIO Configuration                    外设资源配置
    PA10      ------> USART1_RX
    PB6      ------> USART1_TX
    */
  GPIO_InitStruct.Pin = LL_GPIO_PIN_10;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_ALTERNATE;
  GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_VERY_HIGH;
  GPIO_InitStruct.Pull = LL_GPIO_PULL_UP;
  GPIO_InitStruct.Alternate = LL_GPIO_AF_7;
  LL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  GPIO_InitStruct.Pin = LL_GPIO_PIN_6;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_ALTERNATE;
  GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_VERY_HIGH;
  GPIO_InitStruct.Pull = LL_GPIO_PULL_UP;
  GPIO_InitStruct.Alternate = LL_GPIO_AF_7;
  LL_GPIO_Init(GPIOB, &GPIO_InitStruct);
                                                  外设配置
  USART_InitStruct.BaudRate = 115200;
  USART_InitStruct.DataWidth = LL_USART_DATAWIDTH_7B;
  USART_InitStruct.StopBits = LL_USART_STOPBITS_1;
  USART_InitStruct.Parity = LL_USART_PARITY_NONE;
  USART_InitStruct.TransferDirection = LL_USART_DIRECTION_TX_RX;
  USART_InitStruct.HardwareFlowControl = LL_USART_HWCONTROL_NONE;
  USART_InitStruct.OverSampling = LL_USART_OVERSAMPLING_16;

  LL_USART_Init(USART1, &USART_InitStruct);
  LL_USART_ConfigAsyncMode(USART1);
}
```

图174. HAL与LL：main.c代码片段

基于HAL的main.c

```
  */
/* Includes ------------------------------------------
#include "main.h"
#include "stm32l4xx_hal.h"
#include "usart.h"
#include "gpio.h"

void SystemClock_Config(void);
void Error_Handler(void);

int main(void)
{
  /* Reset of all peripherals,
  Initializes the Flash interface and the Systick.
  **/
  HAL_Init();

  /* Configure the system clock */
  SystemClock_Config();

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART1_UART_Init();
```

基于LL的main.c

```
  */
/* Includes ------------------------------------------
#include "main.h"

#include "usart.h"
#include "gpio.h"

void SystemClock_Config(void);
void Error_Handler(void);

int main(void)
{
  /* Reset of all peripherals,
  Initializes the Flash interface and the Systick.
  **/
  LL_Init();

  /* Configure the system clock */
  SystemClock_Config();

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART1_UART_Init();
```

6.3 自定义代码生成

STM32CubeMX支持通过FreeMarker模板引擎生成自定义代码（参见http://www.freemarker.org）。

6.3.1 FreeMarker用户模板的STM32CubeMX数据模型

STM32CubeMX可以为以下任何MCU配置信息生成基于FreeMarker模板文件（.ftl扩展名）的自定义代码：

- 用户配置使用的MCU外设列表
- 这些外设的参数值列表
- 这些外设使用的资源列表：GPIO、DMA请求和中断。

用户模板文件必须与STM32CubeMX数据模型兼容。这意味着模板必须用以下行开头：

```
[#ftl]
[#list configs as dt]
[#assign data = dt]
[#assign peripheralParams =dt.peripheralParams]
[#assign peripheralGPIOParams =dt.peripheralGPIOParams]
[#assign usedIPs =dt.usedIPs]
```

并用以下行结束：

```
[/#list]
```

提供了用于指导的示例模板文件（参见*图 175*）。

如果模板中有任何可用的代码，STM32CubeMX还将生成用户特定代码。

如下例所示，使用示例模板时，ftl命令作为其所生成数据旁边的注释提供：

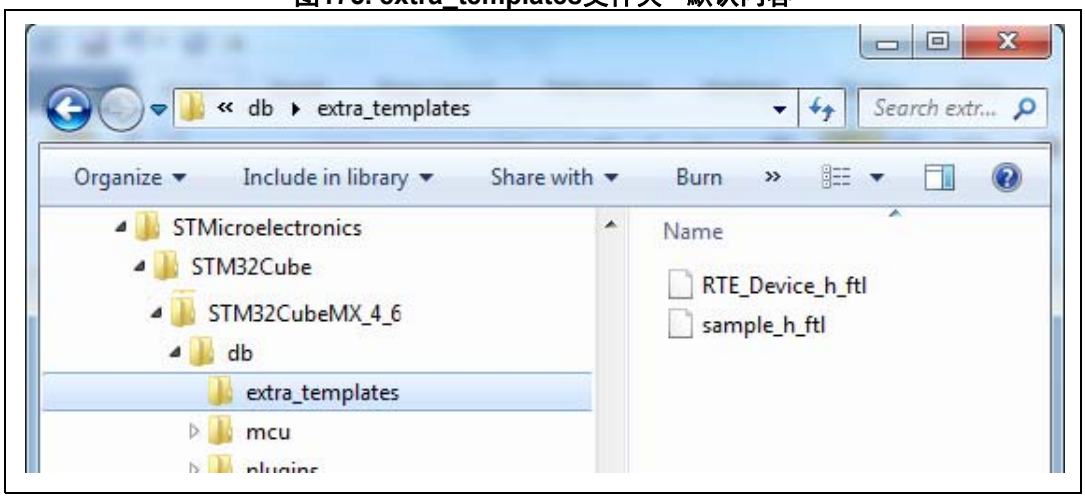
模板中的FreeMarker命令：

```
${peripheralParams.get("RCC").get("LSI_VALUE")}
```

生成的代码：

```
LSI_VALUE : 32000      [peripheralParams.get("RCC").get("LSI_VALUE")]
```

图175. extra_templates文件夹 - 默认内容



6.3.2 保存并选择用户模板

用户可以将FreeMarker模板文件放在db/extra_templates文件夹或任何其他文件夹中的STM32CubeMX安装路径下。

然后，对于给定的项目，用户将通过**“项目管理器”**视图菜单中的“代码生成器”选项卡中的**“模板设置”**窗口，选择与其项目相关的模板文件（请参阅*第 4.9 节*）。

6.3.3 自定义代码生成

要生成自定义代码，用户必须将FreeMarker模板文件放在db/extra_templates文件夹中的STM32CubeMX安装路径下（参见*图 176*）。

模板文件名必须遵守命名规则<用户文件名>_<文件扩展名>.ftl，以便按<用户文件名>.<文件扩展名>生成相应的自定义文件。

默认情况下，在用户项目根文件夹中的.ioc文件旁生成自定义文件（参见*图 177*）。

要在不同的文件夹中生成自定义代码，用户应匹配extra_template文件夹中的目标文件夹树结构（参见*图 178*）。

图176. 包含用户模板的extra_templates文件夹

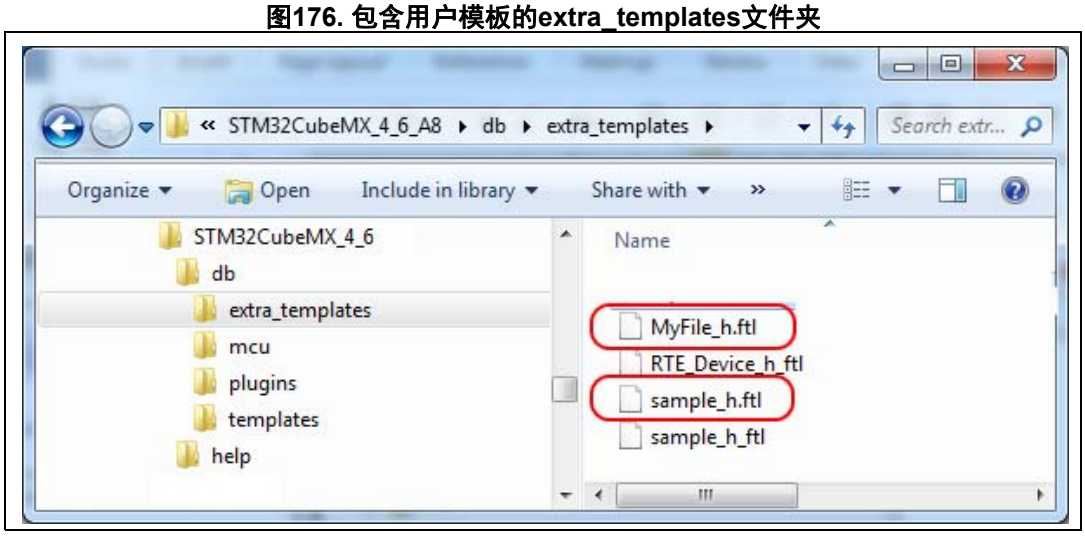


图177. 具有相应的自定义生成文件的项目根文件夹

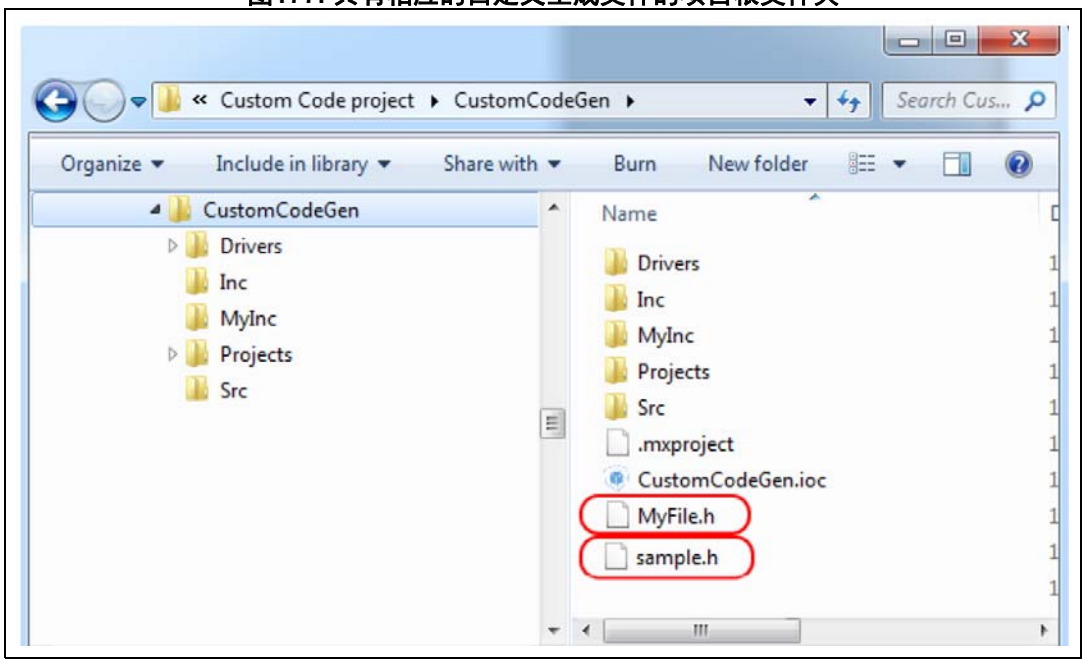
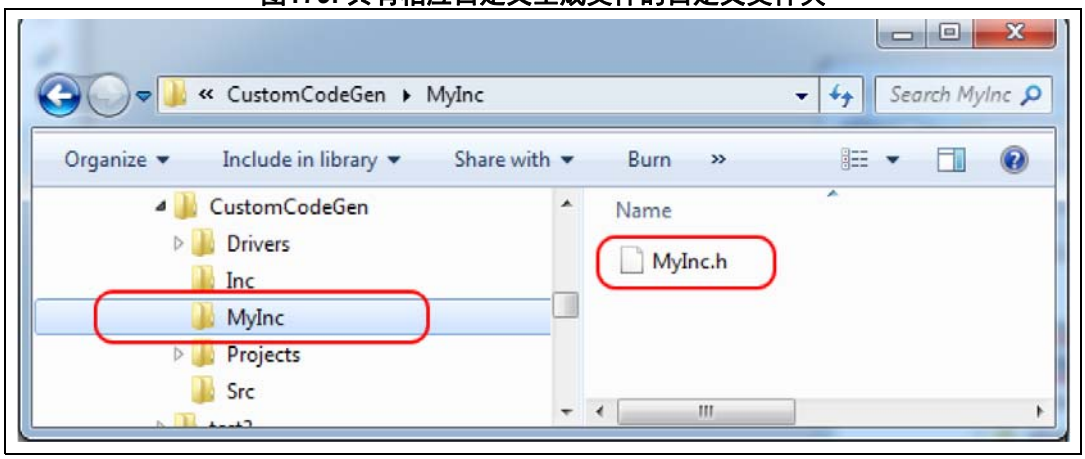


图178. 模板的用户自定义文件夹

图179. 具有相应自定义生成文件的自定义文件夹



6.4 其他C项目生成设置

STM32CubeMX允许通过.extSettings文件指定其他项目设置。此文件必须放在与.ioc文件相同的项目文件夹以及相同的层级中。

例如，当外部工具调用STM32CubeMX生成项目并需要特定项目设置时，可以使用其他设置。

可能的条目和语法

所有条目均为可选。它们按以下三种类别来组织：项目文件、组或其他。

- **[项目文件]**：用于在其中指定其他包括目录的部分

  语法

  HeaderPath = <include directory 1 path>;< include directory 2 path >

  示例

  HeaderPath=../../IIR_Filter_int32/Inc ;

- **[组]**：用于在其中创建新的文件组和/或将文件添加到组中的部分

  语法

  <Group name> = <file pathname1>;< file pathname2>

  示例

  Doc=$ PROJ_DIR$\..\readme.txt

  Lib=C:\libraries\mylib1.lib; C:\libraries\mylib2.lib;

  Drivers/BSP/MyRefBoard = C:\MyRefBoard\BSP\board_init.c; C:\MyRefBoard\BSP\board_init.h;

- **[其他]**用于在其中使能HAL模块和/或指定预处理器定义语句的部分
  - 使能预处理器定义语句

    可以在[其他]行之后使用以下语法指定预处理器定义语句：

    语法

    Define = <define1_name>;<define2_name>

    示例

Define= USE_STM32F429I_DISCO

– 在生成的stm32f4xx_hal_conf.h中使能HAL模块
可以在[其他]行之后使用以下语法使能HAL模块：
语法
HALModule = <ModuleName1>; <ModuleName1>;
示例
HALModule=I2S;I2C

.extSettings文件示例和生成的结果

就本示例而言，通过从板选择器选择
来自STM32CubeMX板选择器的STM32F429I-DISCO板。在**“项目管理器”**视图的“项目”选项卡中选择EWARM工具链。将项目保存为*MyF429IDiscoProject*。.extSettings文本文件放在项目文件夹中所生成的.ioc文件旁，其中包含以下内容：

[组]

Drivers/BSP/STM32F429IDISCO=C:\Users\frq09031\STM32Cube\Repository\STM32Cube_FW_F4_V1.14.0\Drivers\BSP\STM32F429I-Discovery\stm32f429i_discovery.c;
C:\Users\frq09031\STM32Cube\Repository\STM32Cube_FW_F4_V1.14.0\Drivers\BSP\STM32F429I-Discovery\stm32f429i_discovery.h

Lib=C:\Users\frq09031\STM32Cube\Repository\STM32Cube_FW_F4_V1.14.0\Middlewares\Third_Party\FreeRTOS\Source\portable\IAR\ARM_CM4F\portasm.s

Doc=$PROJ_DIR$\..\readme.txt

[其它]

Define = USE_ STM32F429I_DISCO

HALModule = UART;SPI

在生成项目时，此.extSettings文件的存在会触发以下更新：

- EWARM文件夹中的MyF429IDiscoProject.ewp项目文件（参见*图 180*）
- 项目Inc文件夹中的stm32f4xx_hal_conf.h文件（参见*图 181*）
- EWARM用户界面中的项目视图，如*图 182*和*图 183*中所示。

图180. 为预处理定义语句更新.ewp项目文件（EWARM IDE）用于预处理器define语句

```
<settings>
  <name>ICCARM</name>
  <archiveVersion>2</archiveVersion>
  <data>
    <version>28</version>
    <wantNonLocal>1</wantNonLocal>
    <debug>1</debug>
    <option>
      <name>CCDefines</name>
      <state>USE_HAL_DRIVER</state>
      <state>STM32F429xx</state>
      <state>USE_STM32F429I_DISCO</state>
    </option>
```

图181. 更新stm32f4xx_hal_conf.h文件以使能选定模块

stm32f4xx_hal_conf.h

```
/* #define HAL_RTC_MODULE_ENABLED    */
/* #define HAL_SAI_MODULE_ENABLED    */
/* #define HAL_SD_MODULE_ENABLED     */
/* #define HAL_MMC_MODULE_ENABLED    */
#define HAL_SPI_MODULE_ENABLED
/* #define HAL_TIM_MODULE_ENABLED    */
#define HAL_UART_MODULE_ENABLED
/* #define HAL_USART_MODULE_ENABLED    */
/* #define HAL_IRDA_MODULE_ENABLED    */
/* #define HAL_SMARTCARD_MODULE_ENABLED    */
/* #define HAL_WWDG_MODULE_ENABLED    */
/* #define HAL_PCD_MODULE_ENABLED    */
/* #define HAL_HCD_MODULE_ENABLED    */
```

图182. 添加到EWARM IDE中的组的新组和新文件

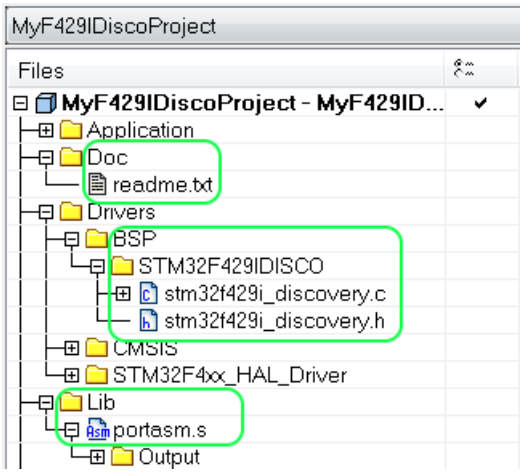
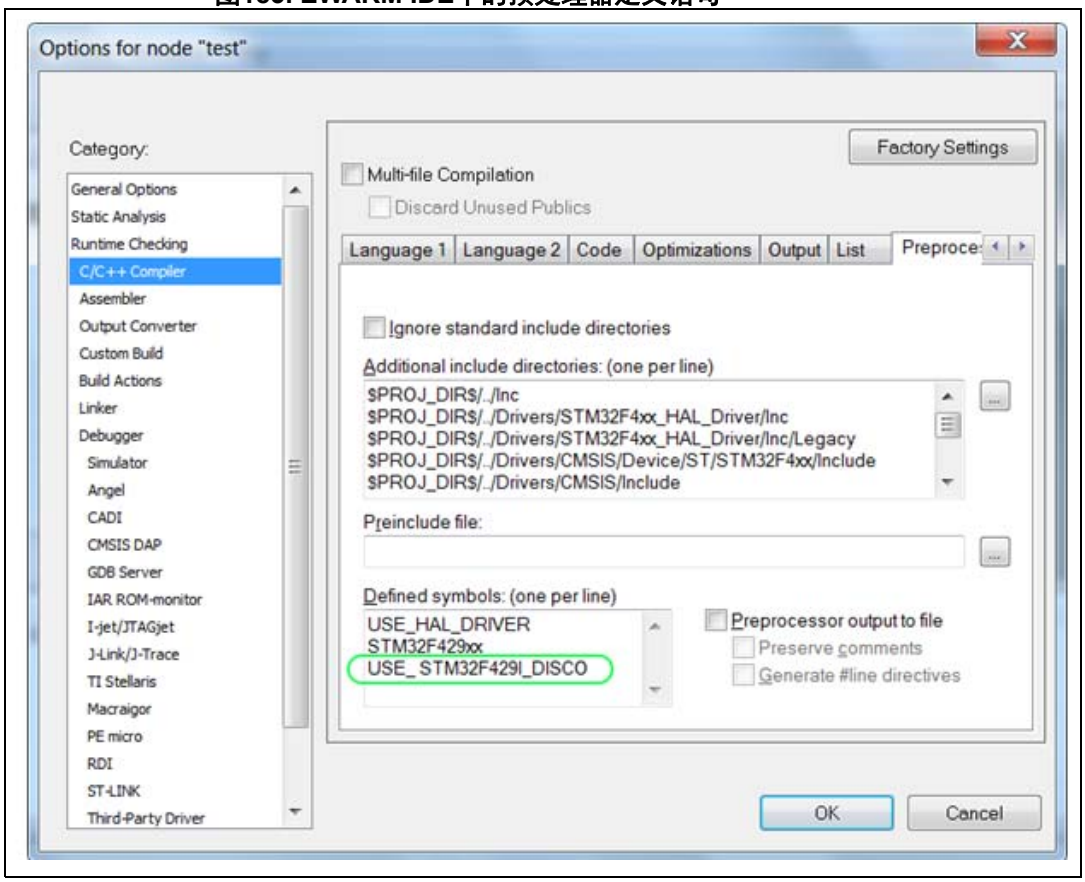


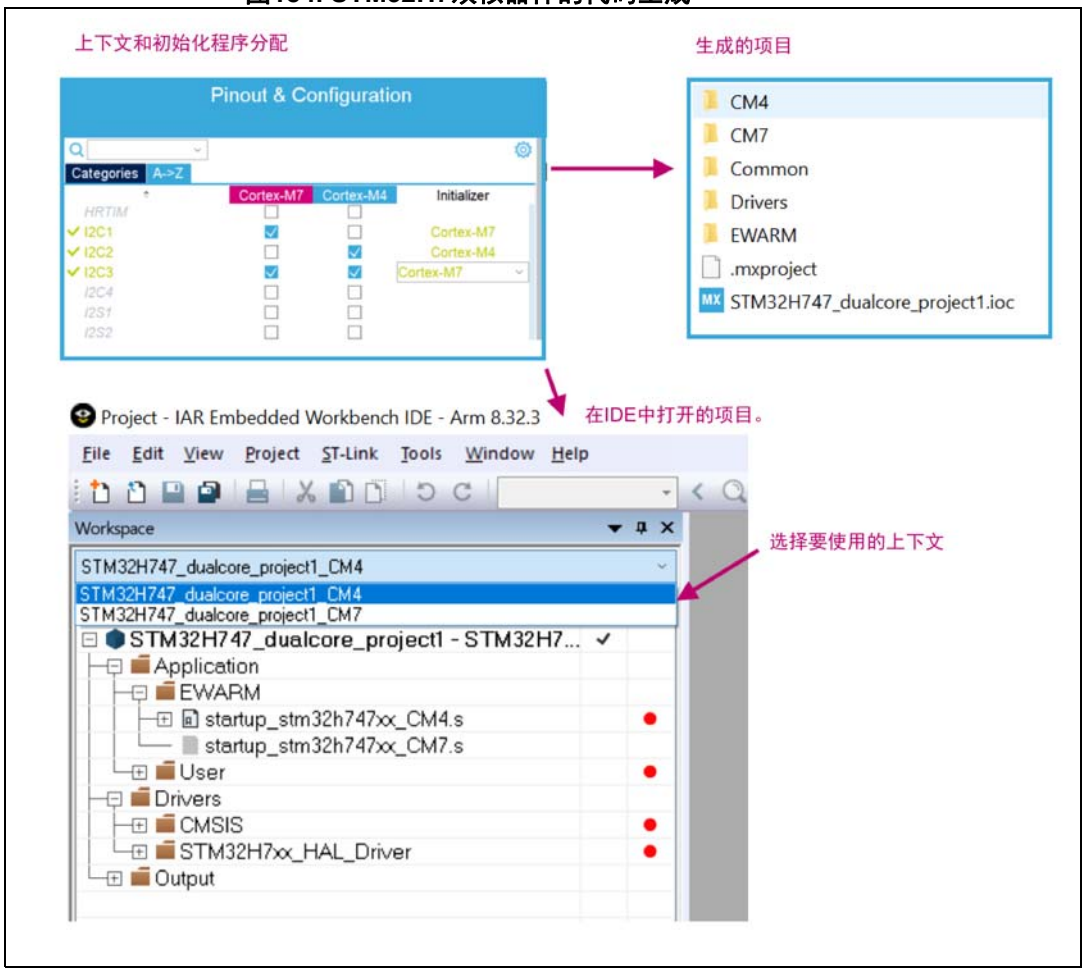
图183. EWARM IDE中的预处理器定义语句



7 双核MCU的代码生成（仅适于STM32H7双核产品系列）

为了与Arm Cortex-M双核产品一起使用，STM32CubeMX会根据内核分配和在用户界面中进行的初始化程序选择，自动为两个内核生成代码（详细信息请参阅*第 4.6节：STM32H7双核产品系列的"引脚布局和配置"视图*）。

图184. STM32H7双核器件的代码生成



生成的初始化代码

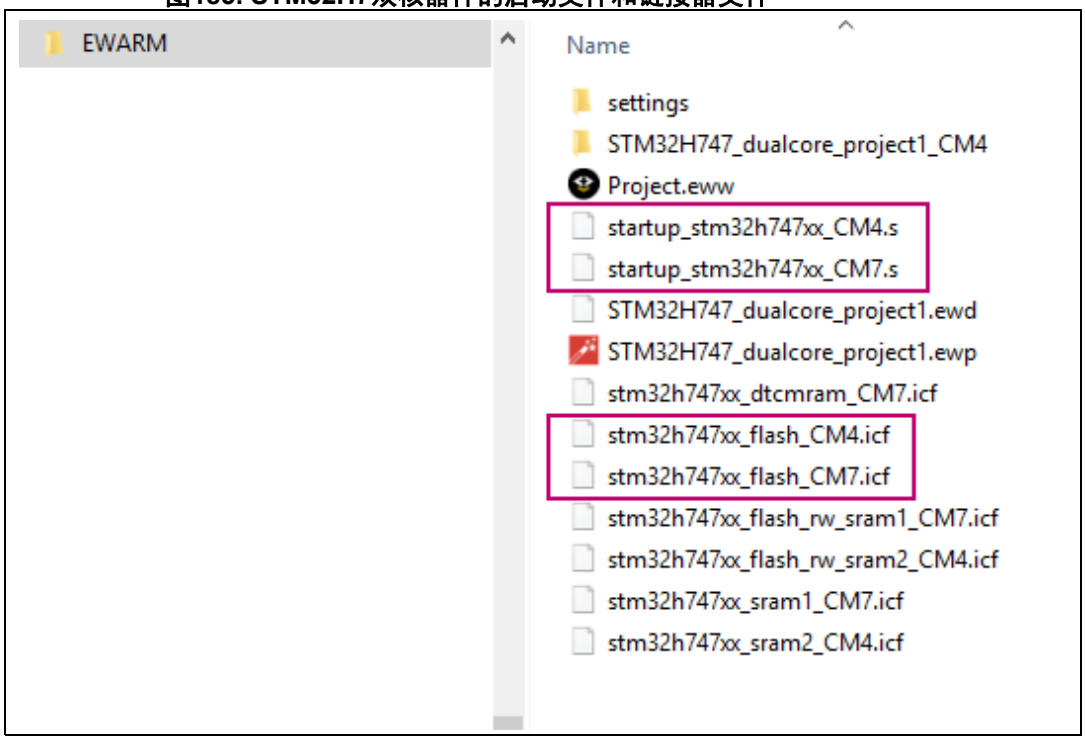
该代码在"CM4"、"CM7"和"常规"文件夹中生成。"常规"文件夹包含system_stm32h7xx.c，其中包含时钟树设置。

将外设或中间件分配给这两个内核时，将为这两个内核生成函数MX_<name>_init，但将仅从初始化程序端进行调用。

生成的启动文件和链接器文件

项目的每个配置（_M4或_M7）都应带有一个启动文件和一个链接器文件，每个文件都分别带有_M4或_M7的后缀。

图185. STM32H7双核器件的启动文件和链接器文件



生成的启动模式代码

STM32CubeMX目前仅支持一种启动模式，即两个ARM Cortex-M内核立即启动。

其他启动模式将在以后在“项目管理器”视图中作为项目选项引入：

- Arm Cortex-M7内核启动，Arm Cortex-M4 gated
- Arm Cortex-M4 core booting, Arm Cortex-M7 gated
- 从闪存执行的第一内核启动将第二内核代码加载到SRAM，然后使第二内核启动。

STM32CubeMX使用STM32CubeH7 MCU软件包随附的模板文件作为参考。

8 启用Trustzone的代码生成（仅适于STM32L5系列）

在STM32CubeMX的“项目管理器”视图中，所有项目生成选项仍可用。

然而，工具链的选择仅适于支持以下各项的IDE/编译器：
Cortex®-M33内核：

- EWARM v8.32或更高版本
- MDK-ARM v5.27或更高版本（ARM编译器6）
- STM32CubeIDE（GCC v4.2或更高版本）

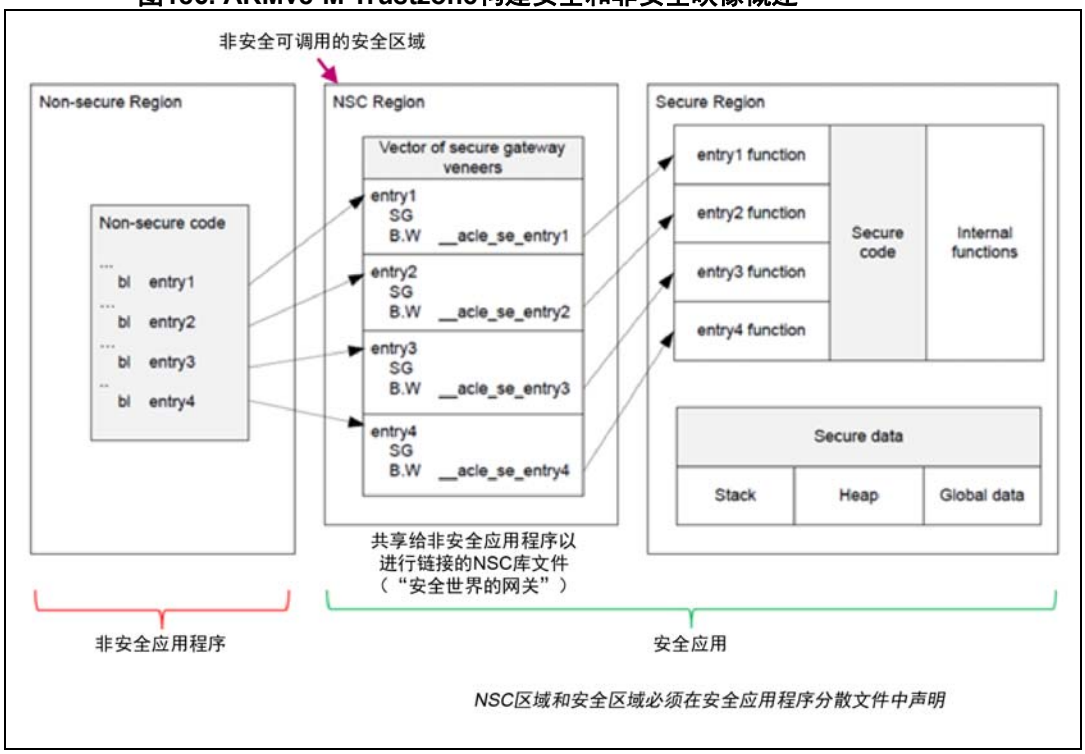
在选择产品时，STM32CubeMX需要在是否启用TrustZone之间进行选择。

- 启用TrustZone后，STM32CubeMX会生成两个C项目：一个安全项目和一个非安全项目。编译后，有两个图像可供下载，其中每个内核一个图像。
- 禁用TrustZone后，STM32CubeMX会生成一个非安全的C项目，这对不支持TrustZone的其他产品都是一样的。

功能特性

启用Trustzone时，须调整项目生成，以确保可以构建安全和非安全的映像。

图186. ARMv8-M Trustzone构建安全和非安全映像概述



为项目启用TrustZone时，STM32CubeMX会生成三个文件夹：

- NonSecure用于非安全代码
- Secure用于安全代码
- Secure_nsclib用于非安全可调用区域

请参阅*图 187*（使用TZ_BasicStructure_project_inCubeIDE.png）和*图 188*（使用STM32L5_STM32CubeMX_Project_settings_inCubeIDE.png）。

图187. 启用STM32L5 TrustZone的项目的项目浏览器视图

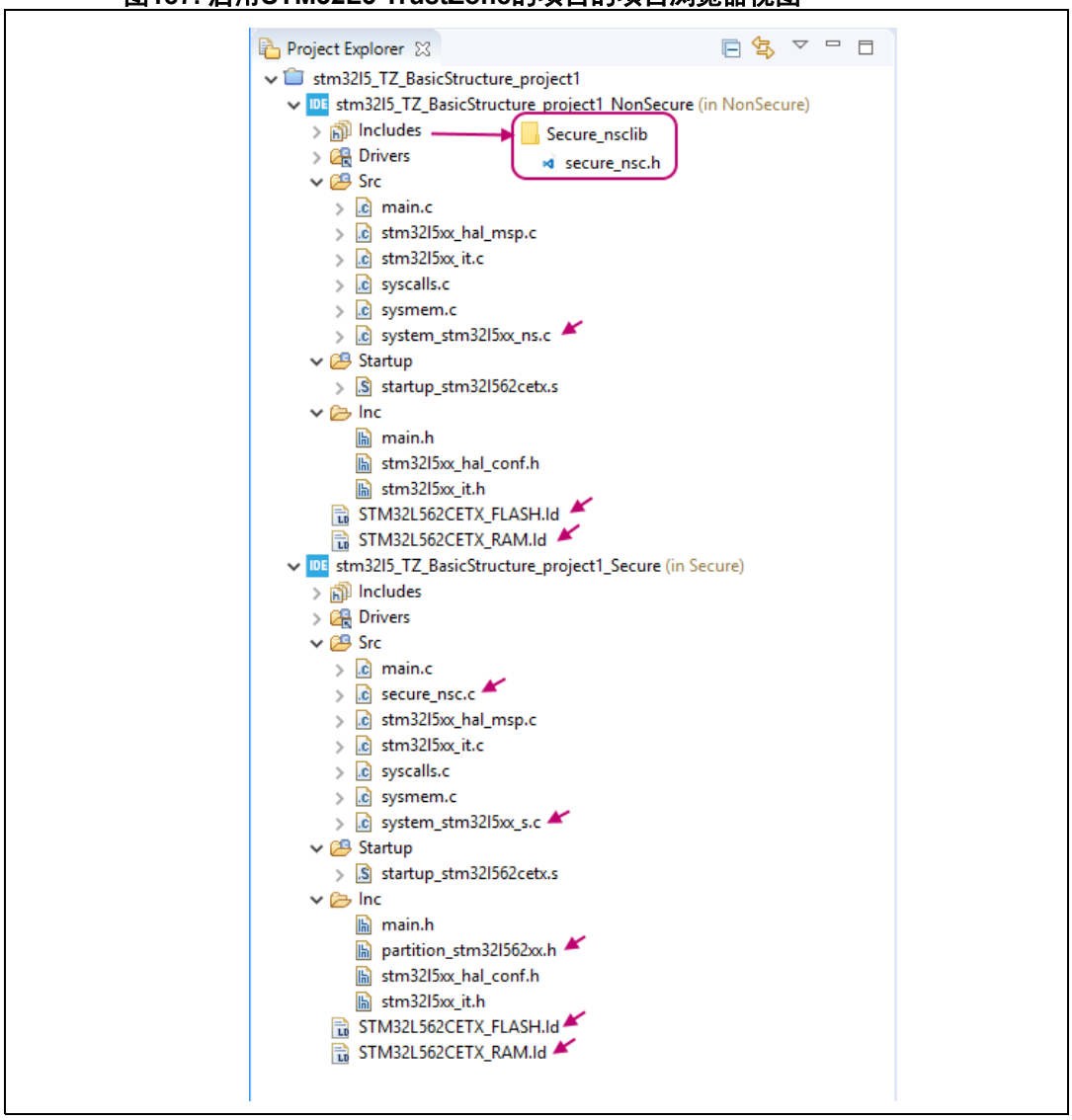
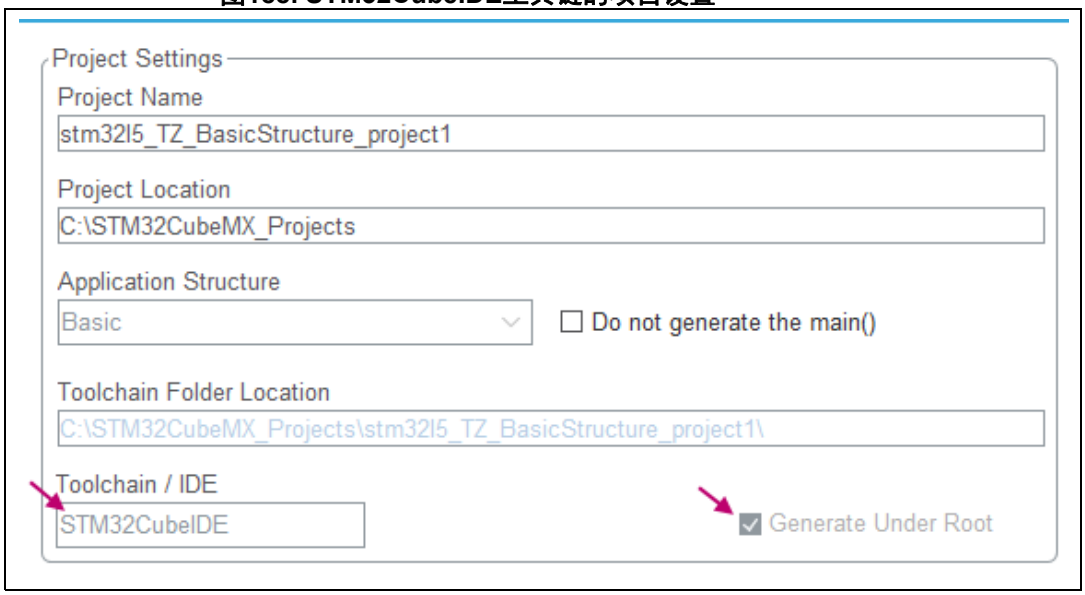


图188. STM32CubeIDE工具链的项目设置



STM32CubeMX还会生成特定文件，在*表 22*中进行了详细介绍。

表22. 启用TrustZone 时生成的文件

文件	文件夹	详细信息
产品内核安全/非安全分区.h“模板”文件 示例：partition_stm32l552xx.h	安全	基于CMSIS CORE V5.3.1 partition_ARMCM33.h模板的ARMCM33安全/非安全区域的初始设置。 它对安全归属单元（SAU）CTRL寄存器、休眠和异常处理的设置操作、浮点单元和中断目标进行初始化。
secure_nsc.h文件	Secure_nsclib	用户须填写 非安全可调用API列表。 Templates\TrustZone\Secure_nsclib文件夹的STM32L5Cube嵌入式软件包中有模板可作为参考。
System_stm32l5xx_s.c	安全	当系统实现安全性时，将在安全应用程序中使用的CMSIS Cortex-M33器件外设访问层系统源文件。
System_stm32l5xx_ns.c	非安全	系统实现安全性时，将在非安全应用程序中使用的CMSISCortex-M33器件 外设访问层系统源文件。
STM32L562CETX_FLASH STM32L562CETX_RAM 或 STM32L552CETX_FLASH STM32L552CETX_RAM	安全，非安全	用于安全和非安全存储器布局的链接器文件。 文件扩展名和命名约定： – .icf（EWARM） – .sct（MDK-ARM），或 – .ld（GCC编译器工具链）

9 设备树生成（仅适于STM32MP1系列）

Linux中的设备树用于提供一种描述不可发现的硬件的方法。意法半导体为所有平台配置数据（包括DDR配置）广泛使用设备树。

Linux开发人员可以手动编辑器件树源文件（dts），但作为替代，STM32CubeMX提供了部分器件树生成服务，以减少工作量并减轻新手压力。STM32CubeMX将生成与板级配置相对应的部分器件树。“部分”表示并非生成整个（板级）设备树，而仅仅生成主要部分，这些部分通常表示需要较大工作量而且可能导致编译错误和功能异常：

- 文件夹结构和文件到文件夹的分布
- dtsi和数据头包含物
- pinCtrl和时钟生成
- 片上系统器件节点定位
- 多核相关配置（Etzpc绑定，资源管理器绑定，外设分配）

9.1 设备树概述

要正常运行，所有软件都需要获取执行软件的平台的硬件描述，包括CPU类型、存储器大小和引脚配置。当前Linux内核和U-boot已将此类不可发现的硬件描述置于单独的二进制文件中，即器件树BLOB（dtb）。使用OpenSTLinux发行版随附的dtc编译器，从设备树源文件（dts）编译设备树BLOB。

设备树结构由板级文件（.dts）组成，该文件包含两个设备树源包含文件（.dtsi）：soc级文件和–pinctrl文件，其中–pinctrl文件列出了引脚复用配置。

设备树结构非常接近C语言的多级结构，其中
“根”（/）是最高级别，然后“外设”是分层结构中进一步描述的小节点（请参阅图*189*、*190*和*191*）。

当用户配置需要时，STM32CubeMX生成使用广泛超载的机制来完成或更改某些SOC器件定义。

图189. STM32CubeMX生成的DTS－提取1

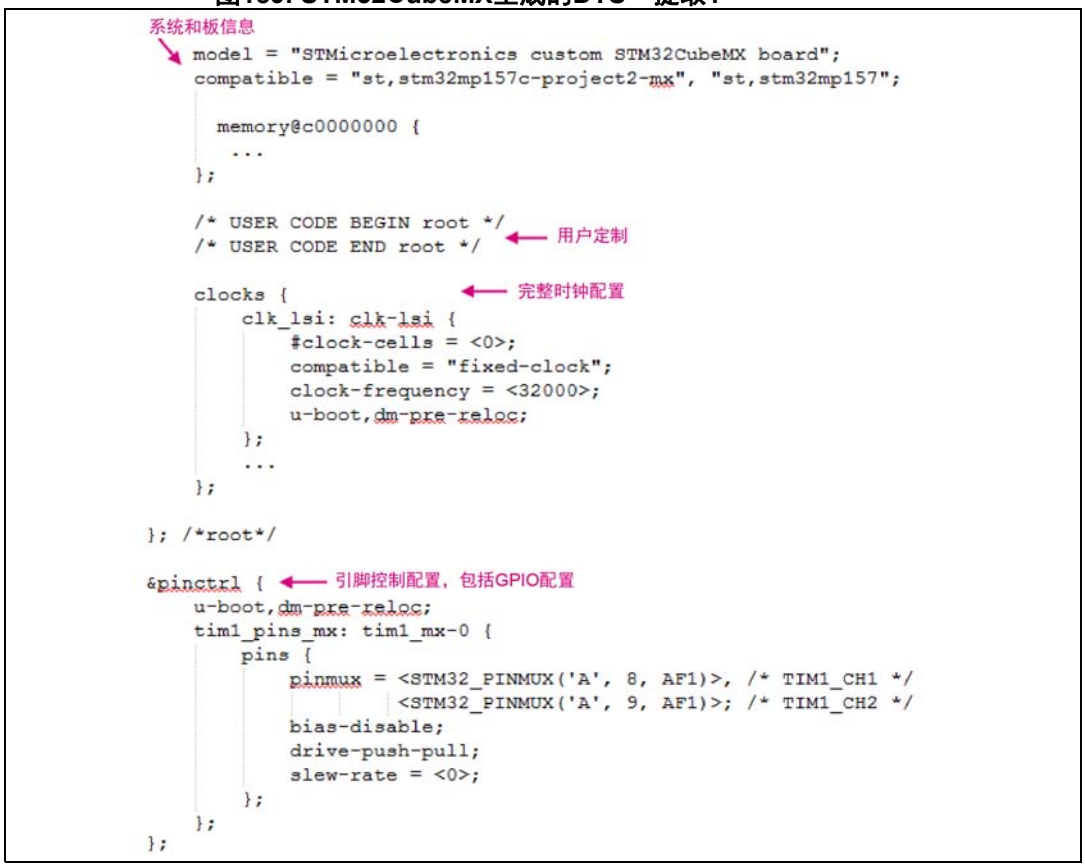


图190. STM32CubeMX生成的DTS－提取2

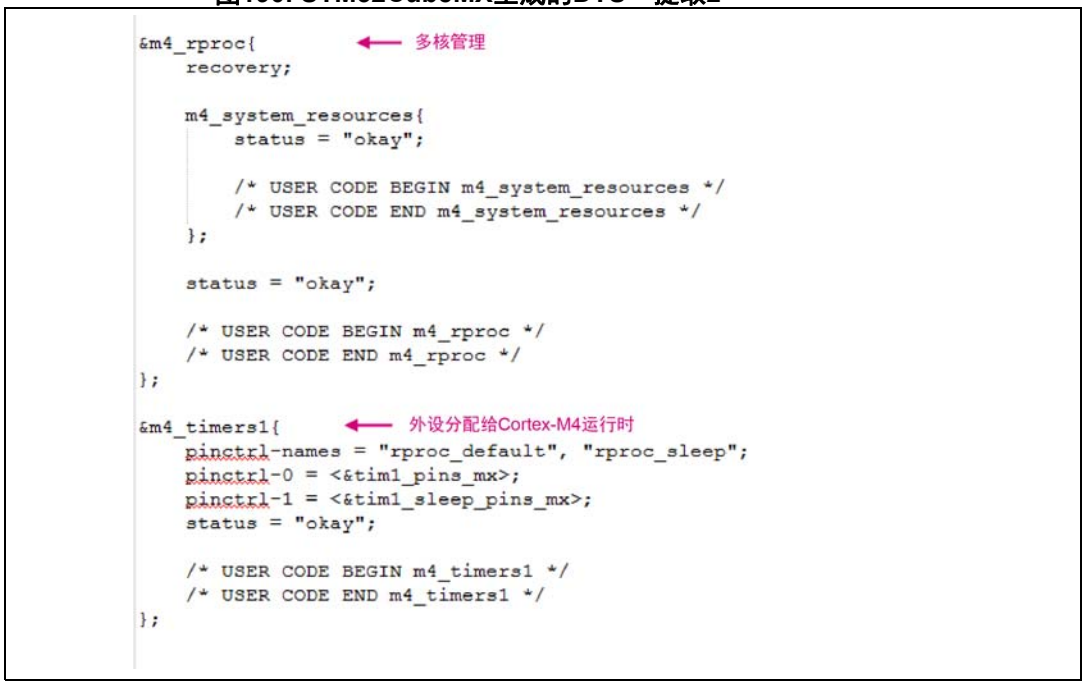
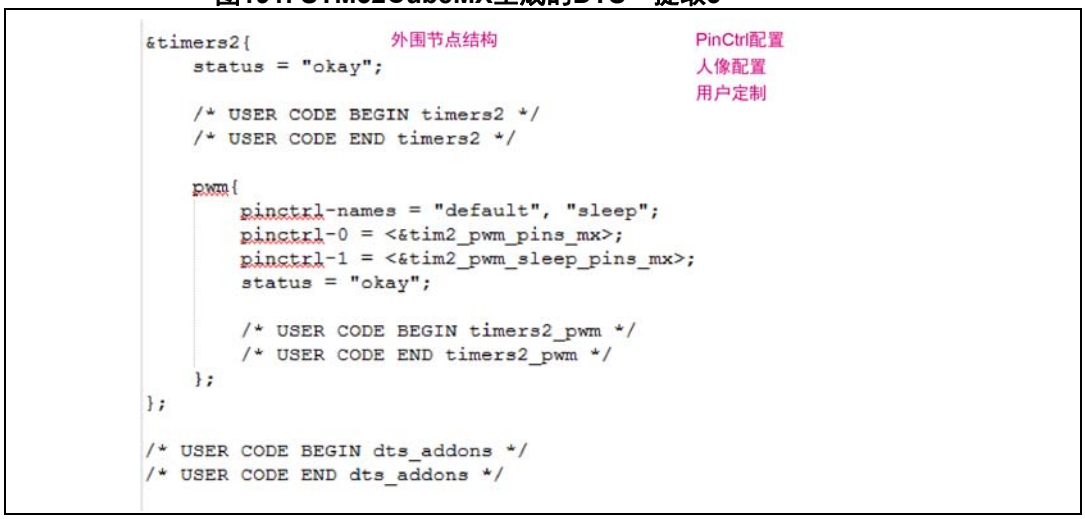


图191. STM32CubeMX生成的DTS－提取3



有关更多详细信息，请参阅Thomas Petazzoni的“Device Tree for Dummies”，可在https://elinux.org上获得。

有关STM32MP1系列器件树特性的更多信息，请参阅ST Wiki（https://wiki.st.com/stm32mpu）。

9.2 STM32CubeMX设备树的生成

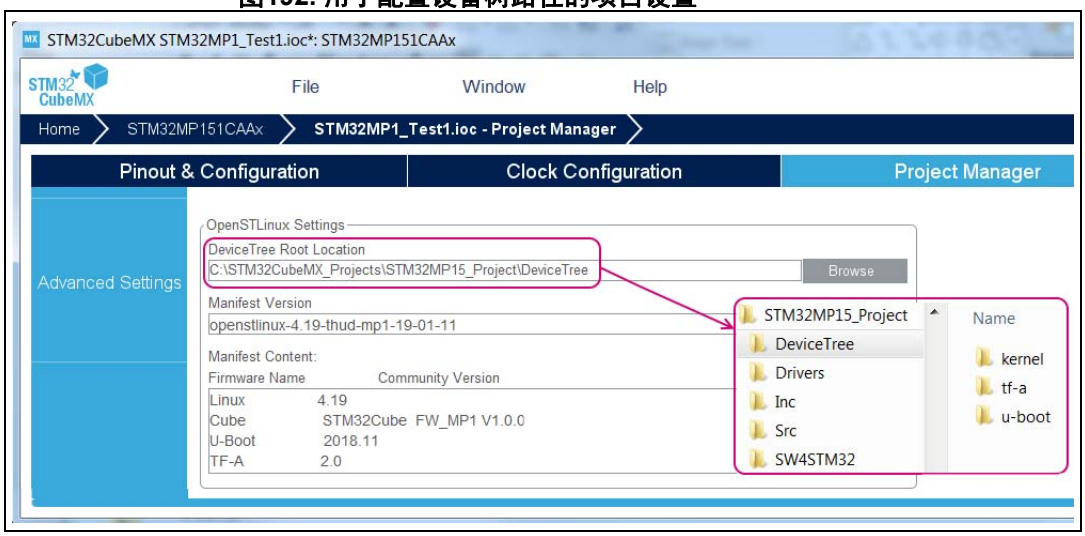
对于STM32MP1系列，已扩展STM32CubeMX代码生成功能，可生成针对所支持的固件的器件树（DT）：

- 用于配置TF-A和SP_min的单个DT
- 用于配置U-Boot的DT
- 用于配置Linux内核的DT

可通过同一 GENERATE CODE 按键访问DTS生成。

可以从“项目管理器”视图的“高级设置”选项卡中的“OpenSTLinux设置”下配置DT生成路径（请参阅*图 192*）。STM32CubeMX为每个设备树生成设备树源（DTS）文件。

图192. 用于配置设备树路径的项目设置



设备树结构包括：

- 完整的时钟树
- 完整的引脚控制
- 完整的多核参考定义
- 一组设备节点和小节点
- 可以填充的用户部分，以便得到完整的可启动设备树（内容不会在下一代丢失）。

生成的DTS文件反映了用户配置，例如将外设分配给运行时和启动加载程序，或时钟树设置。

STM32CubeMX DT生成可确保不同DT之间的一致性。此外，它会生成DDR配置文件，作为TF-A和U-Boot器件树的一部分。

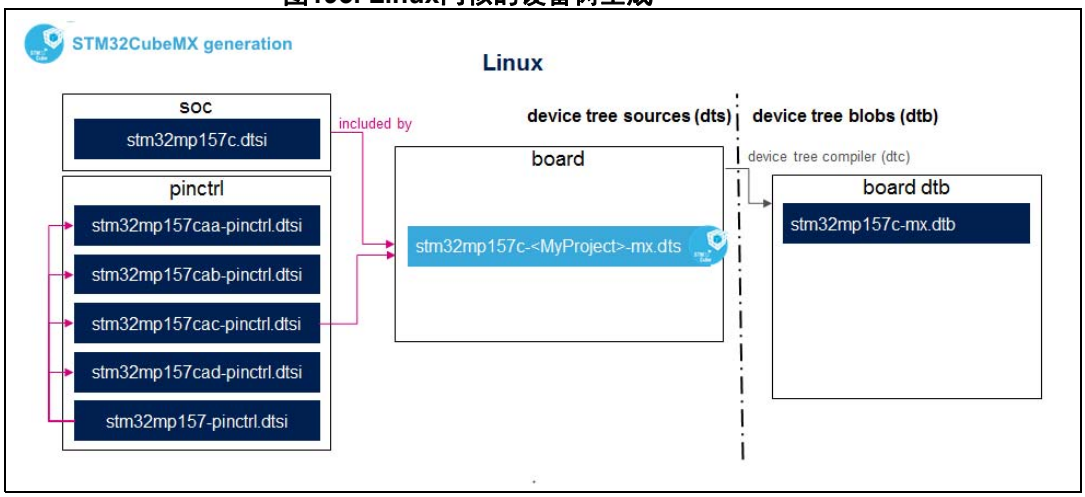
上述文件及其包含的文件将进行编译以创建目标固件的设备树BLOB。

9.2.1 Linux内核的器件树生成

STM32CubeMX仅为Linux生成“板”文件。该文件包括与所选软件包相对应的“soc”文件和“pinctrl”文件。

根据Linux内核源代码Documentation/devicetree/bindings/文件夹中可用的设备树绑定，可以通过填充用户部分来完善STM32CubeMX生成的设备树节点。

图193. Linux内核的设备树生成

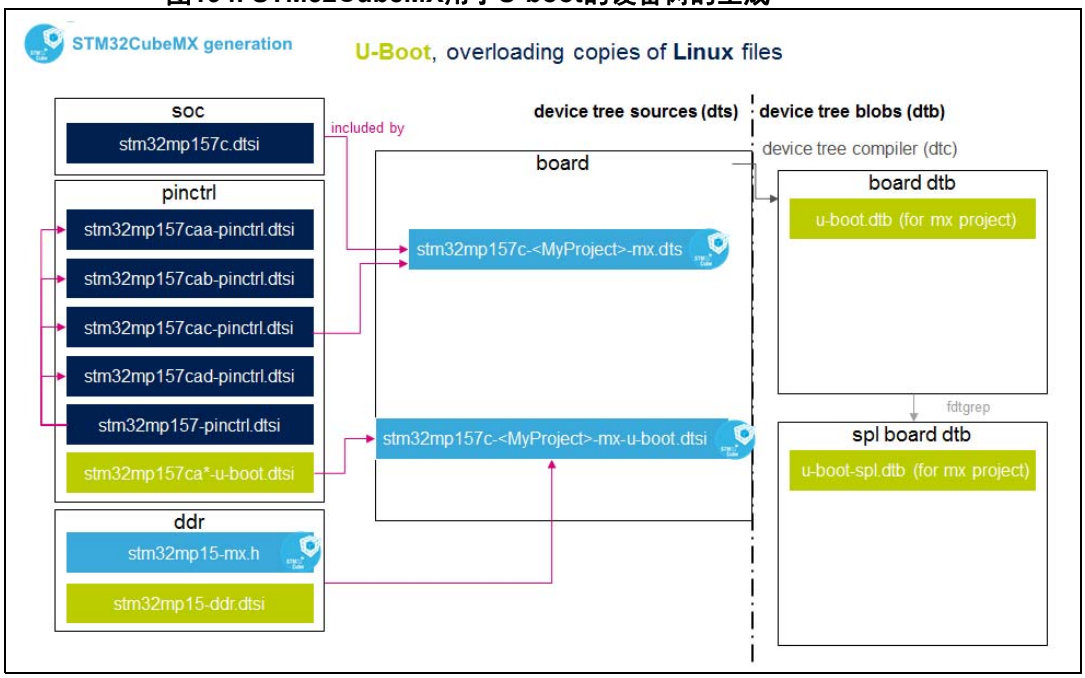


9.2.2 用于U-boot的器件树生成

STM32CubeMX为U-Boot复制Linux dts文件，并用两个新文件进行完善：一个文件用于“ddr”配置，另一个用于U-Boot附加元件，主要包括在需要时使用“u-boot,dm-pre-reloc”属性。

根据U-Boot源代码Documentation/devicetree/bindings/文件夹中可用的设备树绑定，可以通过填充用户部分来完善STM32CubeMX生成的设备树节点。

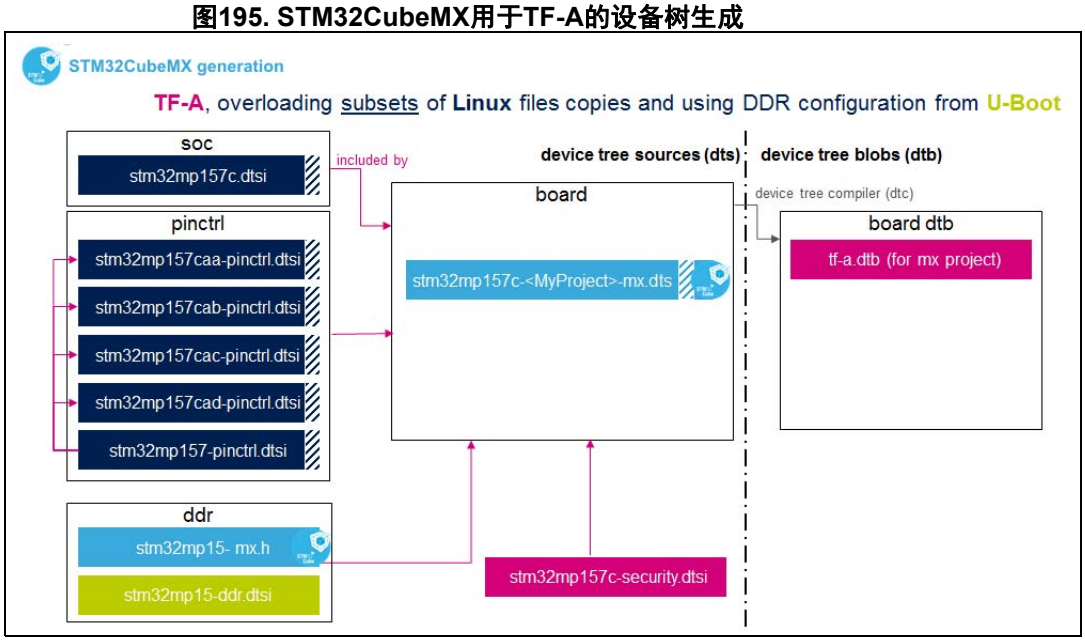
图194. STM32CubeMX用于U-boot的设备树的生成



9.2.3 用于TF-A的设备树生成

为了节省空间，STM32CubeMX为TF-A生成“board”dts文件，该文件Linux“板”dts文件的简化版。该文件包括TF-A随附的在“soc”和“pinctrl”侧的简化版dtsi文件。为U-Boot生成的相同“ddr”配置文件重新用于TF-A。

图195. STM32CubeMX用于TF-A的设备树生成



根据TF-A源代码docs/devicetree/bindings/文件夹中可用的设备树键合，可以通过填充用户部分来完善STM32CubeMX生成的设备树节点。

10 支持使用CMSIS-Pack 标准的其他软件组件

CMSIS-Pack标准描述了软件组件、设备参数和评估板支持的交付机制。

基于XML的软件包描述（pdsc）文件描述了软件包的内容（文件集合）。它包括源代码、头文件、软件库、文档和源代码模板。软件包包括完整的文件集以及以ZIP格式提供的pdsc文件。安装软件包后，所有随附的软件组件均可用于开发工具。

软件组件是源模块、头文件和配置文件以及库的集合。包含软件组件的软件包还可以包括示例项目和用户代码模板。

有关更多详细信息，请访问http://www.keil.com网站。

STM32CubeMX支持以软件包形式交付的第三方和意法半导体其他嵌入式软件解决方案。STM32CubeMX能够：

1. 安装软件包并检查更新（请参阅*第 3.4.4 节*）。
2. 选择当前项目的软件组件（请参阅*第 4.13 节*）。完成此项操作后，所选的组件将显示在树形视图中（请参阅*图 196*）。
3. 从树形视图中启用软件组件（请参阅*图 197*）。使用上下文帮助可获取有关选择的更多详细信息。
4. 配置软件组件（请参阅*图 197*）。该功能仅适用于随附STM32CubeMX专有格式文件的组件。
5. 为选定的工具链生成C项目（请参阅*图 198*）。
   a) 软件组件文件会自动复制到项目中。
   b) 会自动生成软件组件配置和初始化代码。该功能仅适用于随附STM32CubeMX私有格式文件的组件。

图196. 选择CMSIS-Pack软件组件

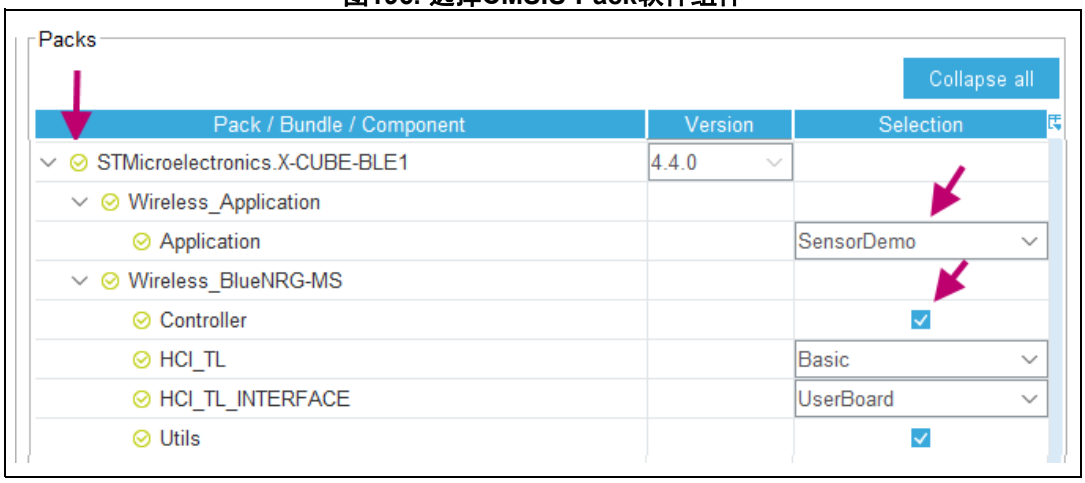


图197. 启用并配置CMSIS-Pack软件组件

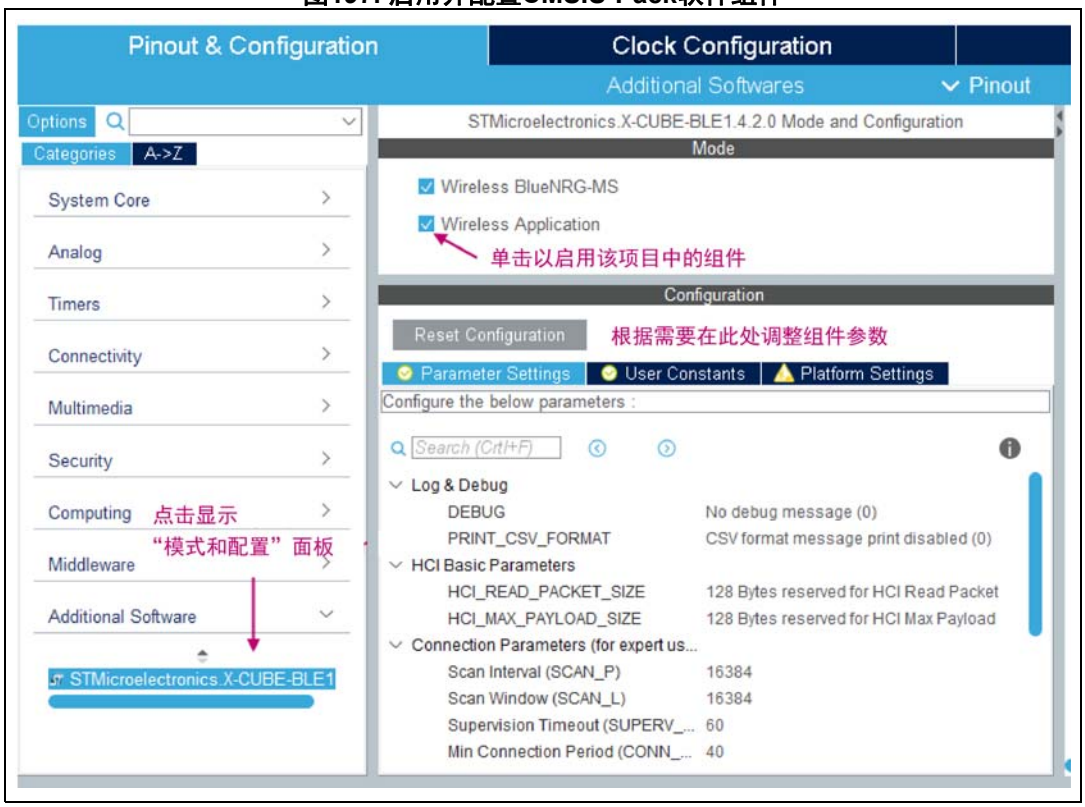
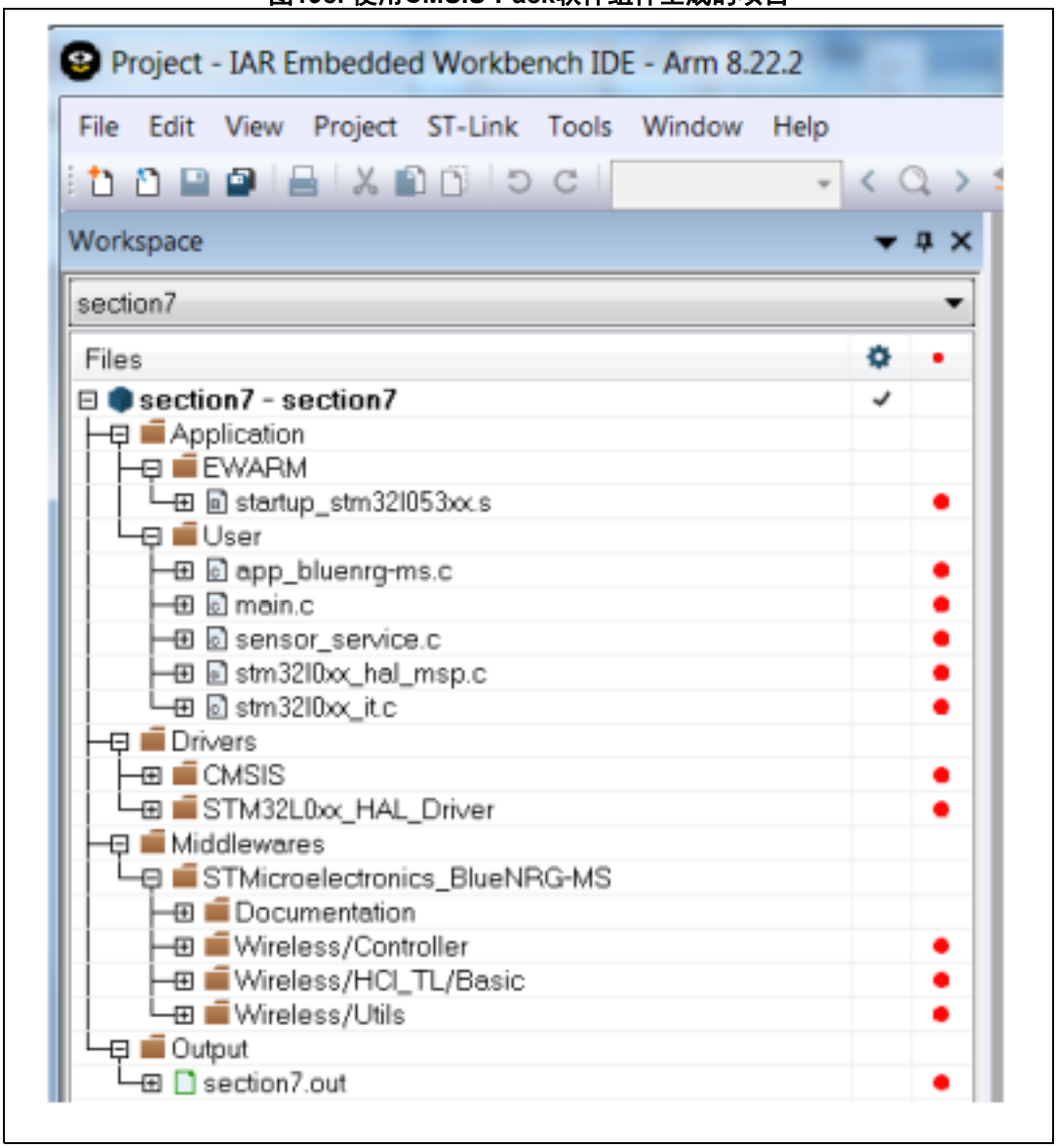


图198. 使用CMSIS-Pack软件组件生成的项目



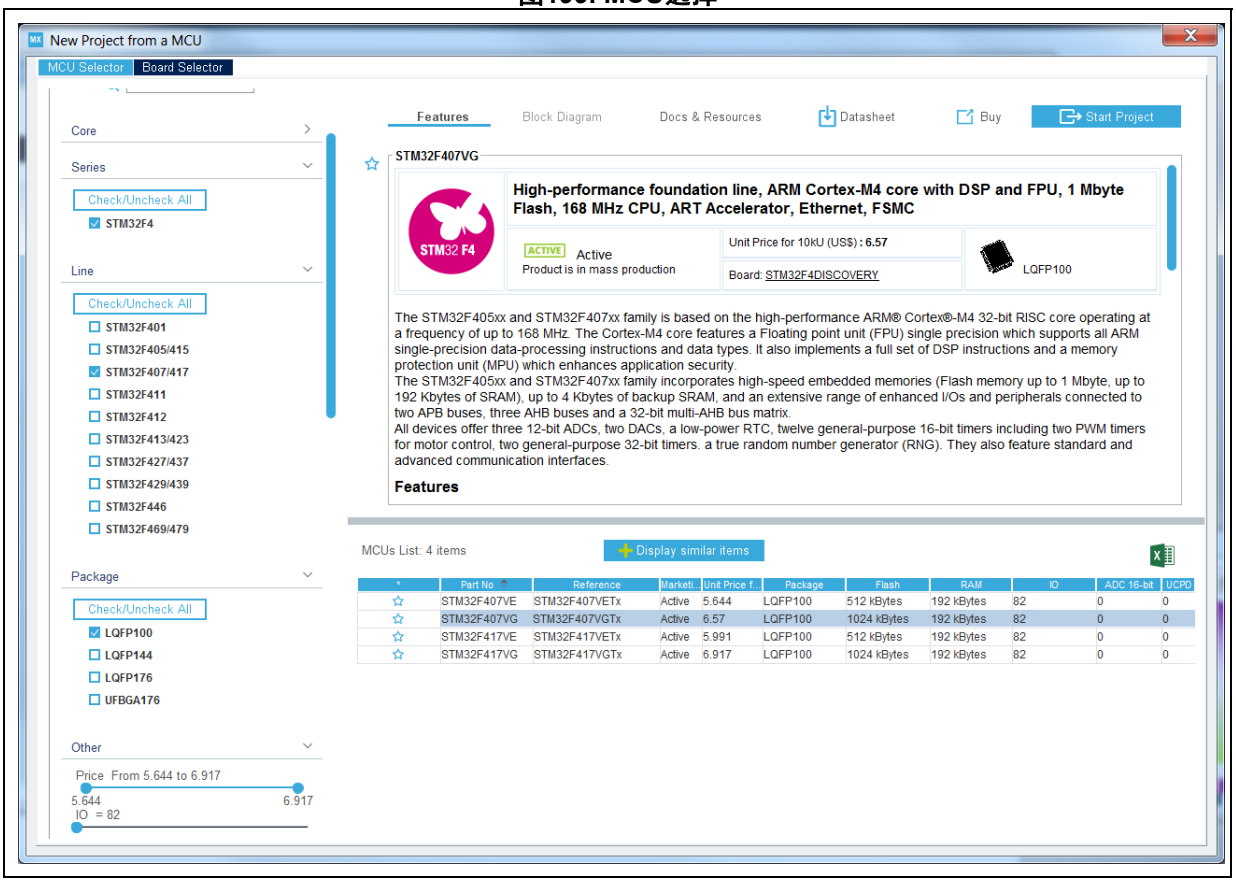
11 教程1：使用STM32F4 MCU从引脚排列到生成项目C代码

本节介绍了配置和C代码生成过程。以STM32F4DISCOVERY板上运行的简单LED切换应用为例。

11.1 创建一个新STM32CubeMX项目

1. 从主菜单栏中选择**文件 > 新建项目**，或从主页选择**新建项目**。
2. 选择**MCU选择器**选项卡，并通过将STM32F4选为“系列”、将STM32F407选为“产品线”和将LQFP100选为“封装”来筛选STM32产品（参见*图 199*）。
3. 从MCU列表中选择STM32F407VGTx，然后点击**“确定”**。

图199. MCU选择



然后使用选定的MCU数据库填充STM32CubeMX视图（*图 200*）。可根据需要通过取消选择**窗口 > 输出**子菜单删除MCU选择底部窗口（参见*图 201*）。

图200. 具有MCU选择的引脚排列视图

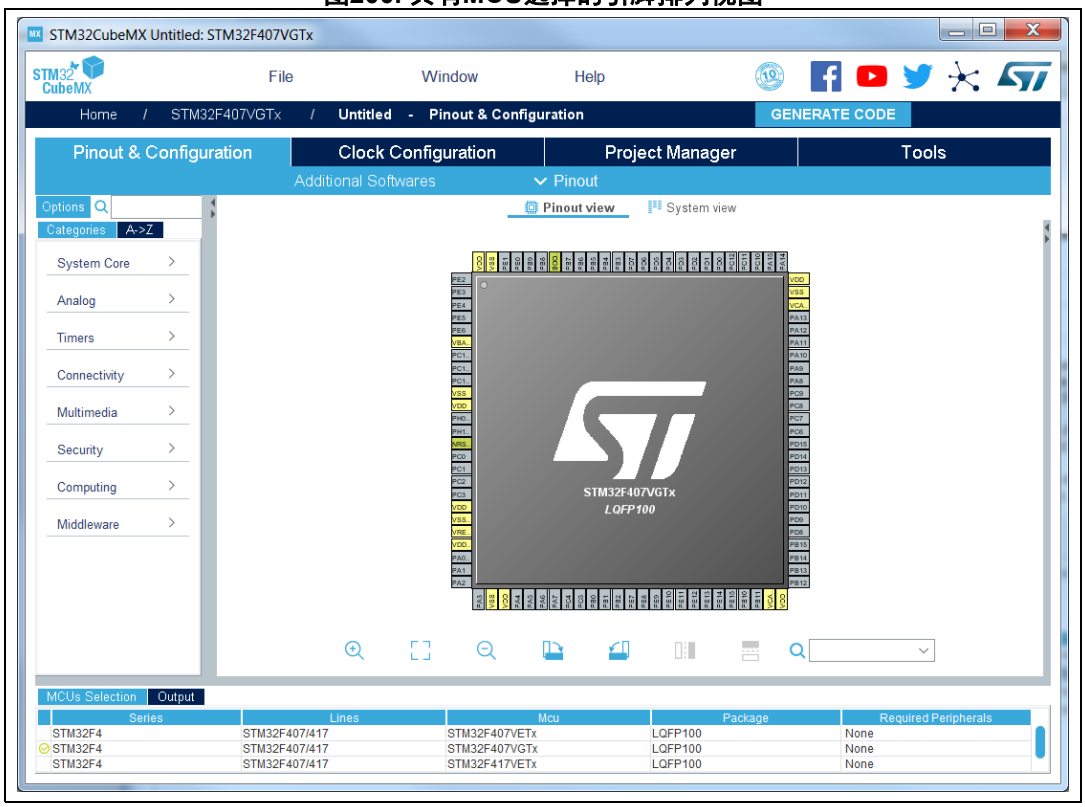
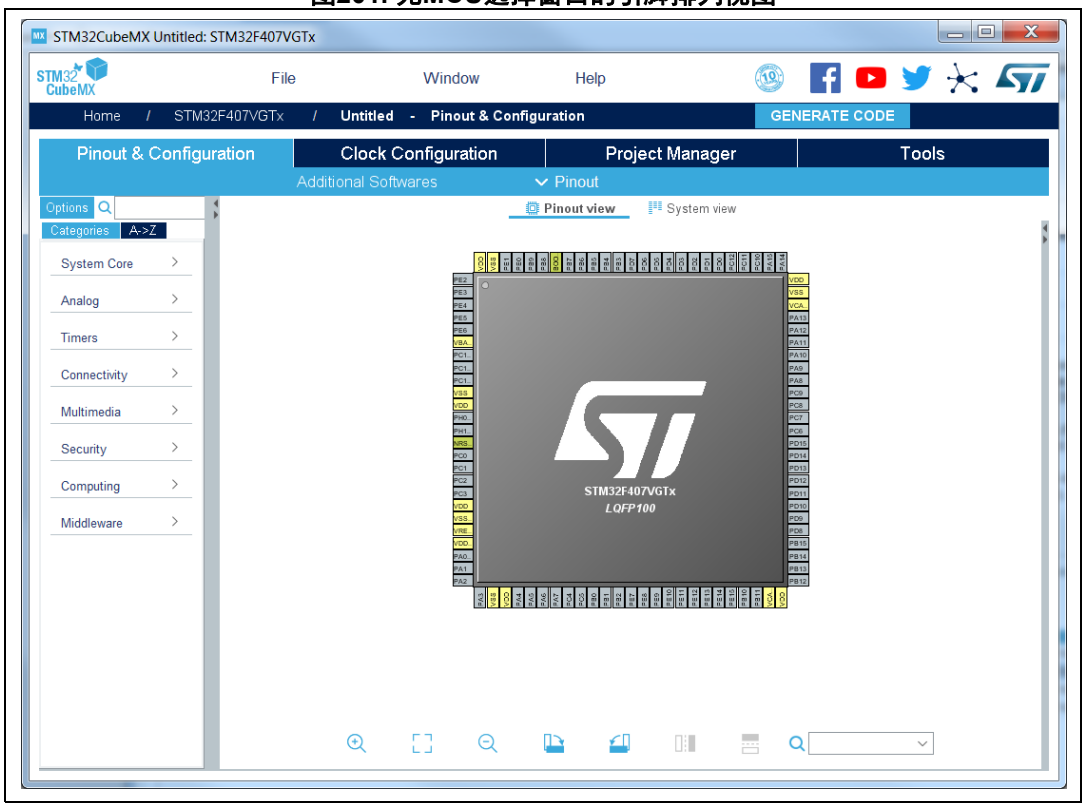


图201. 无MCU选择窗口的引脚排列视图



11.2 配置MCU引脚排列

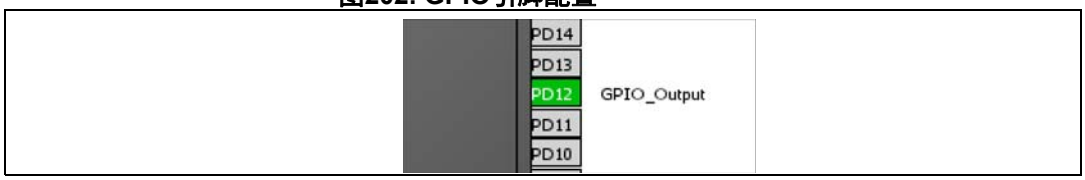
有关菜单、高级操作和冲突解决方案的详细说明，请参见*第 4 节*和*附录A*。

1. 默认情况下，STM32CubeMX显示**引脚排列**视图。
2. 默认情况下，Keep Current Signals Placement未选中，以允许STM32CubeMX移动外设函数和找到最佳引脚分配，即可支持最大外设模式数量的引脚分配。

   由于MCU引脚配置必须与STM32F4DISCOVERY板相匹配，通过使能STM32CubeMX的Keep Current Signals Placement可保持给定引脚的外设函数分配（映射）。

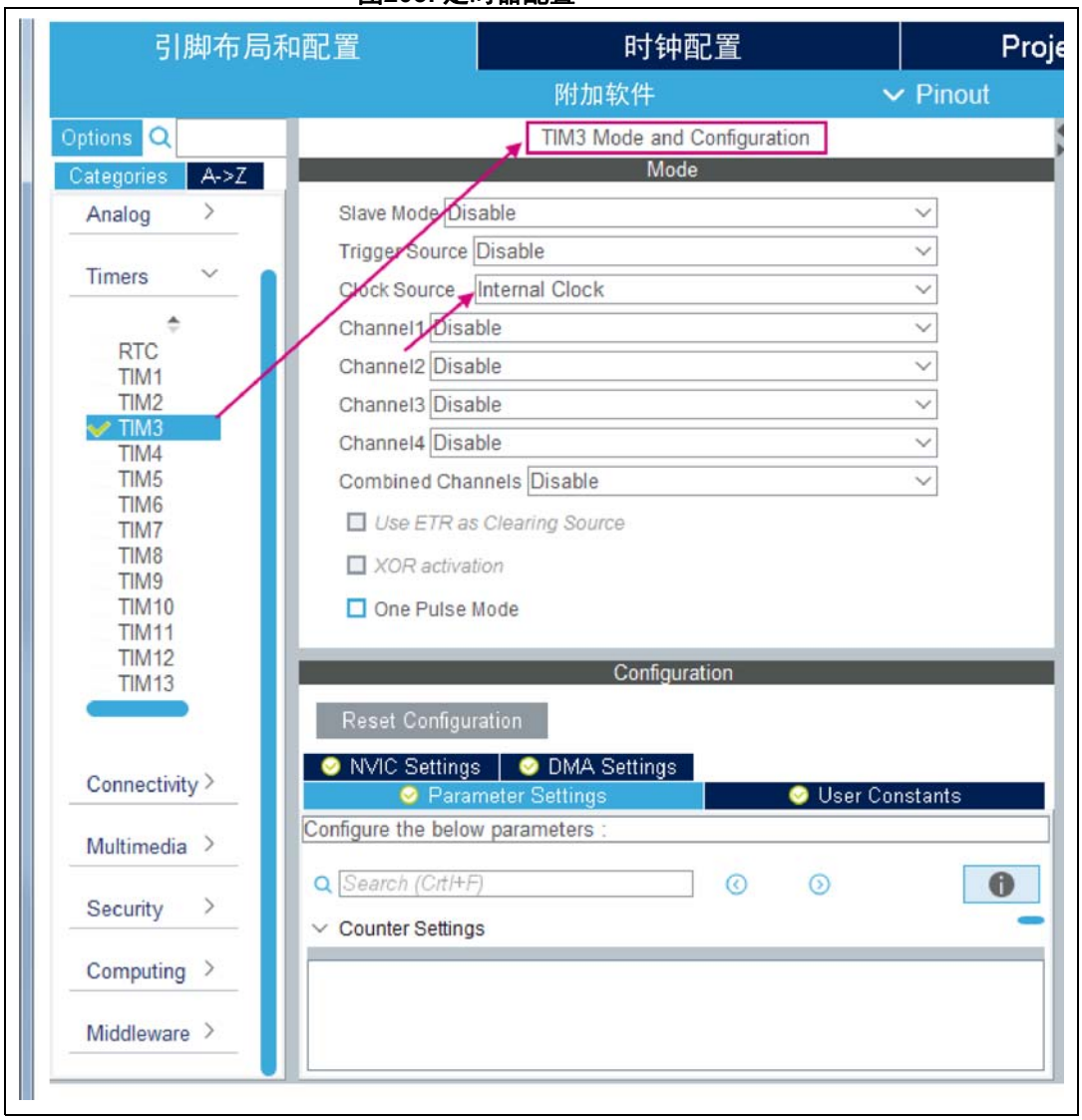
   该设置将另存为用户偏好，以便在重新打开工具或加载其他项目时进行恢复。
3. 选择所需的外设和外设模式：
   a) 通过在**引脚布局**视图中右键单击PD12，可配置GPIO在STM32F4DISCOVERY绿色LED上输出信号，然后选择GPIO_output：

图202. GPIO引脚配置



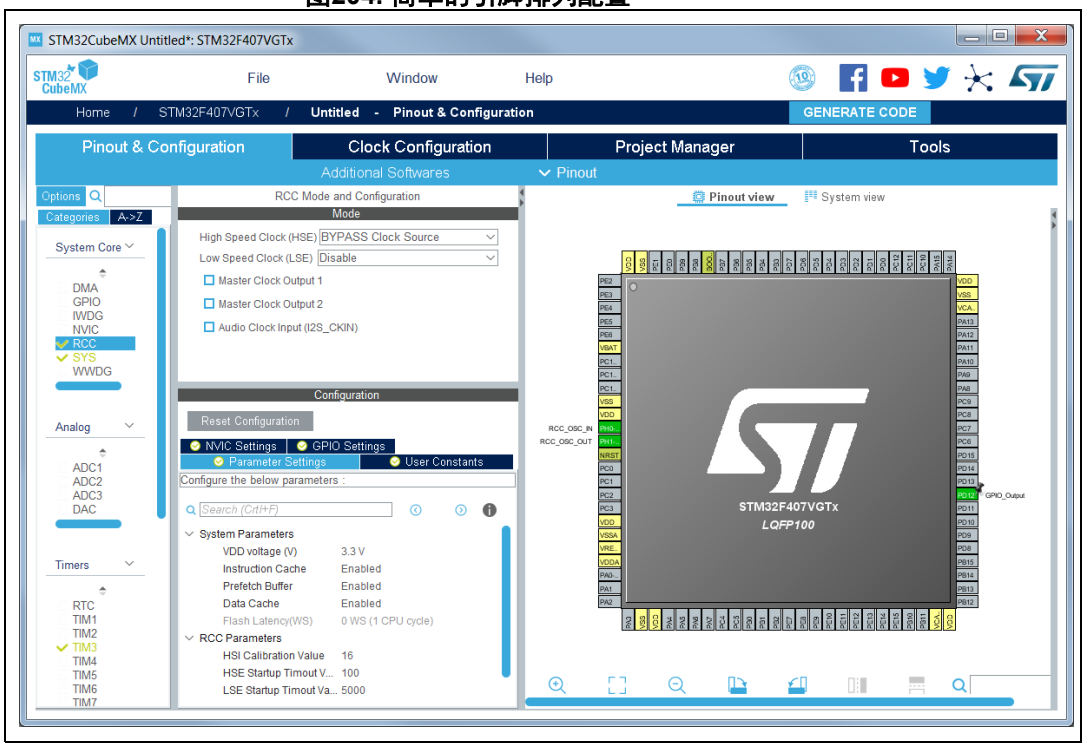
b) 使能将用作LED切换时基的定时器。可通过在外设树中将内部时钟选作TIM3时钟源来完成此操作（参见*图 203*）。

图203. 定时器配置



c) 您也可以配置RCC，将外部振荡器用作潜在时钟源（参见*图 204*）。

图204. 简单的引脚排列配置



这样就完成了本示例的引脚排列配置。

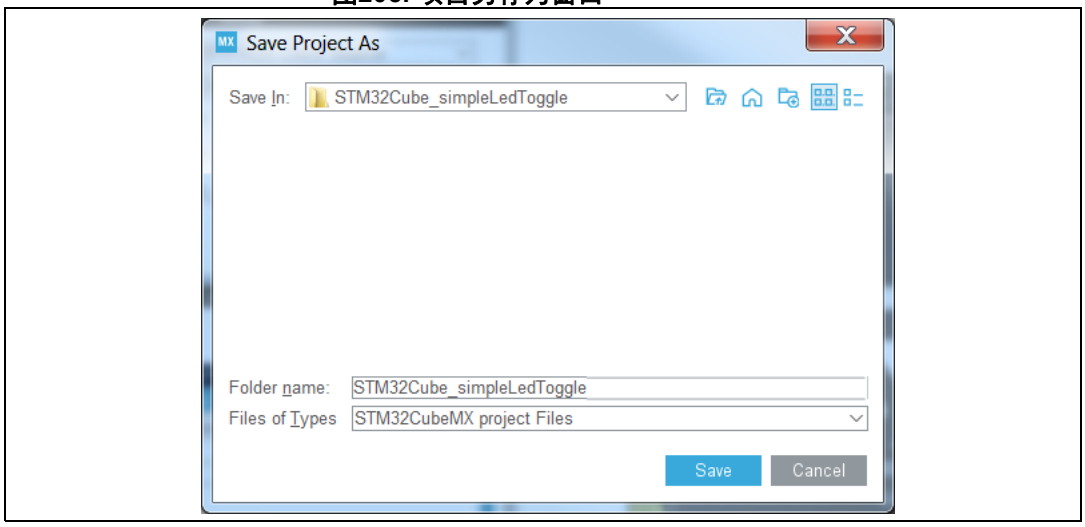
注： *从STM32CubeMX 4.2开始，用户可通过直接从“**板选择器**”选项卡直接加载ST探索板配置来跳过引脚排列配置。*

11.3 保存项目

1. 单击以保存项目。

   首次保存时，请选择项目的目标文件夹和文件名。将自动添加.ioc扩展名，以指示这是一个STM32CubeMX配置文件。

图205. 项目另存为窗口



2. 单击以使用其他名称或位置保存项目。

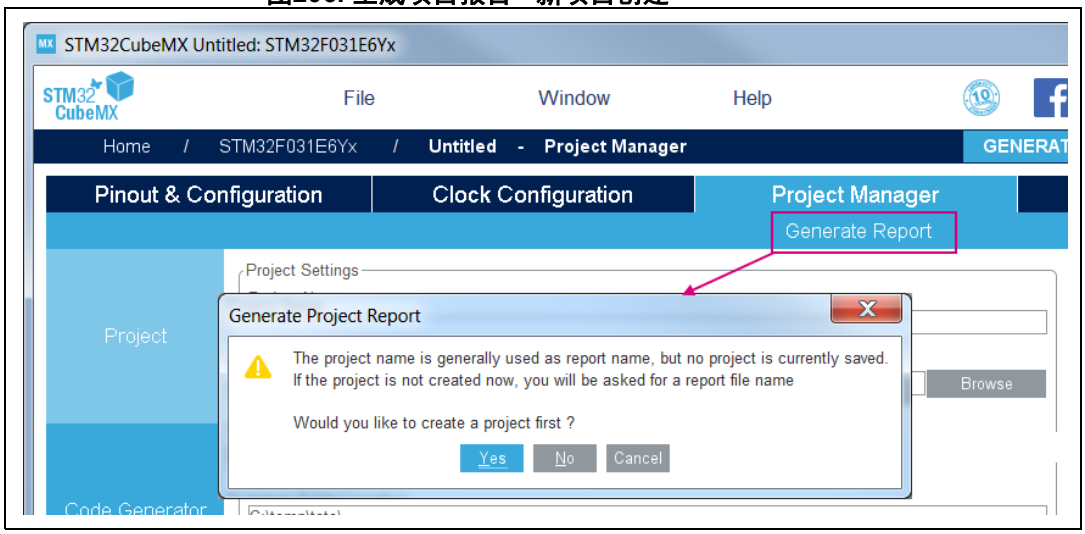
11.4 生成报告

可在配置期间随时生成报告：

1. 单击以生成.pdf和.txt报告。

   如果尚未创建项目文件，则会出现一个警告，以提示用户先保存项目，并要求提供项目名称和目标文件夹（参见*图 206*）。然后为项目生成.ioc文件以及具有相同名称的.pdf和.txt报告。

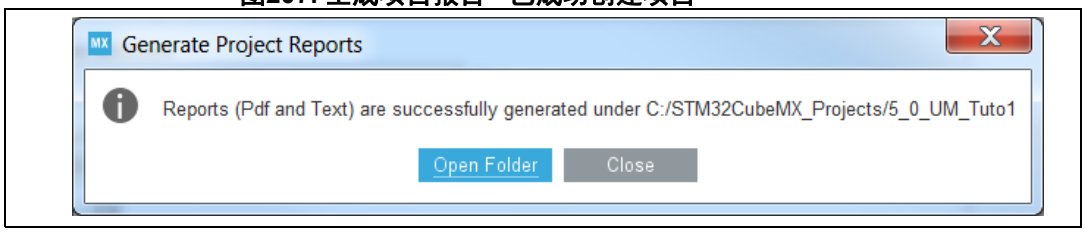
图206. 生成项目报告 - 新项目创建



如果选择**“否”**，则只要求提供报告的名称和位置。

如*图 207*中所示，操作成功后会显示一条确认消息。

图207. 生成项目报告 - 已成功创建项目



2. 使用Adobe Reader打开.pdf报告或使用您最喜欢的文本编辑器打开.txt报告。报告总结了为项目执行的所有设置和MCU配置。

11.5 配置MCU时钟树

以下序列介绍了如何基于STM32F4 MCU配置应用所需的时钟。

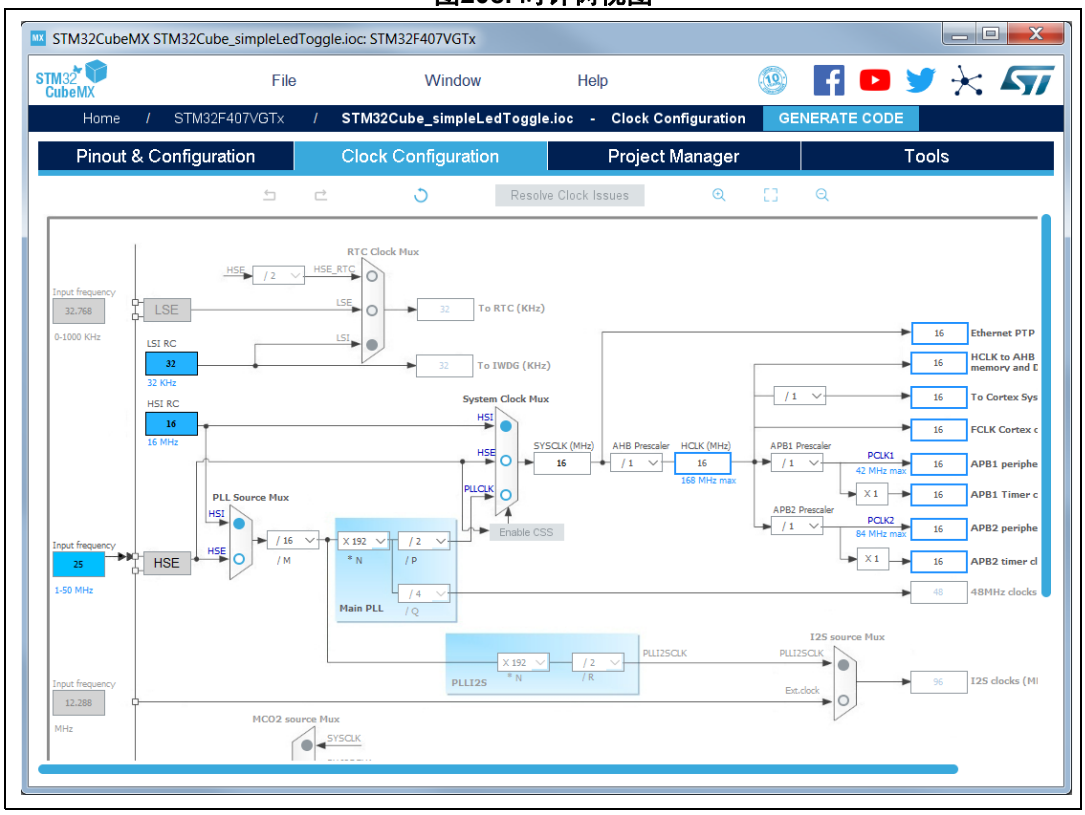
STM32CubeMX通过用户选择的时钟源和预分频器自动产生系统CPU和AHB/APB总线频率。通过最小和最大条件的动态验证，检测出错误设置并以紫红色突出显示。实用的提示详细说明了当设置不可用或错误时需执行的操作。用户频率选择可能会影响某些外设参数（例如UART波特率限制）。

STM32CubeMX使用时钟树视图中定义的时钟设置为每个外设时钟生成初始化C代码。作为项目main.c和stm32f4xx_hal_conf.h中RCC初始化的一部分，在生成的C代码中执行时钟设置（HSE、HSI和以赫兹表示的外部时钟值）。

按照以下顺序配置MCU时钟树：

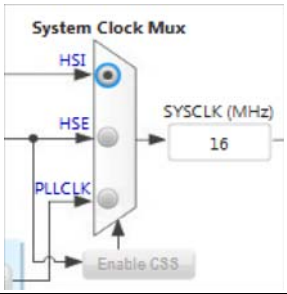
1. 点击**时钟配置**选项卡以显示时钟树（参见*图 208*）。
   内部（HSI、LSI）、系统（SYSCLK）和外设时钟频率字段无法编辑。系统和外设时钟可通过选择时钟源来调节，也可以选择使用PLL、预分频器和倍频器来调节。

图208. 时钟树视图



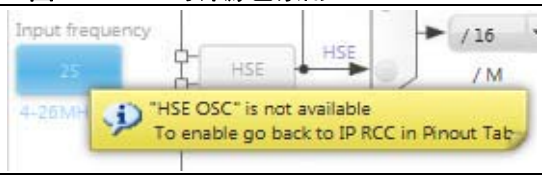
2. 首先选择将驱动微控制器系统时钟的时钟源（HSE、HSI或PLLCLK）。
在本教程的示例中，选择HSI以使用内部16 MHz时钟（参见*图 209*）。

图209. HSI时钟使能



要使用外部时钟源（HSE或LSE），应在**引脚排列**视图中配置RCC外设，因为引脚将用于连接外部时钟晶振（参见*图 210*）。

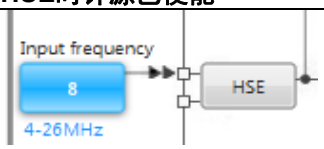
图210. HSE时钟源已禁用



STM32F4DISCOVERY板的其他时钟配置选项：

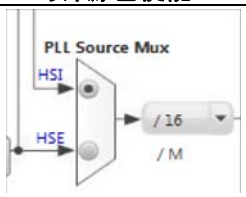
– 选择外部HSE源并在HSE输入频率框中输入8，因为探索板连接了8 MHz晶振：

图211. HSE时钟源已使能



– 选择外部PLL时钟源和HSI或HSE作为PLL输入时钟源。

图212. 外部PLL时钟源已使能



3. 使用HSI将内核和外设时钟保持在16 MHz，不使用PLL和预分频。

注： *也可以选择使用PLL、预分频器和倍频器进一步调整系统和外设时钟：*

其他独立于系统时钟的时钟源可按以下方式配置：

- *USB OTG FS、随机数发生器和SDIO时钟由PLL的独立输出驱动。*
- *I2S外设带有自己的内部时钟（PLLI2S），也可以采用独立的外部时钟源。*
- *USB OTG HS和以太网时钟采用外部时钟源。*

4. 也可以选择为允许向外部电路输出两个时钟的微控制器时钟输出（MCO）引脚配置预分频器。
5. 单击 以保存项目。
6. 转至**配置**选项卡以继续进行项目配置。

11.6 配置MCU初始化参数

注意： STM32CubeMX生成的C代码涵盖了使用STM32Cube固件库进行的MCU外设和中间件初始化。

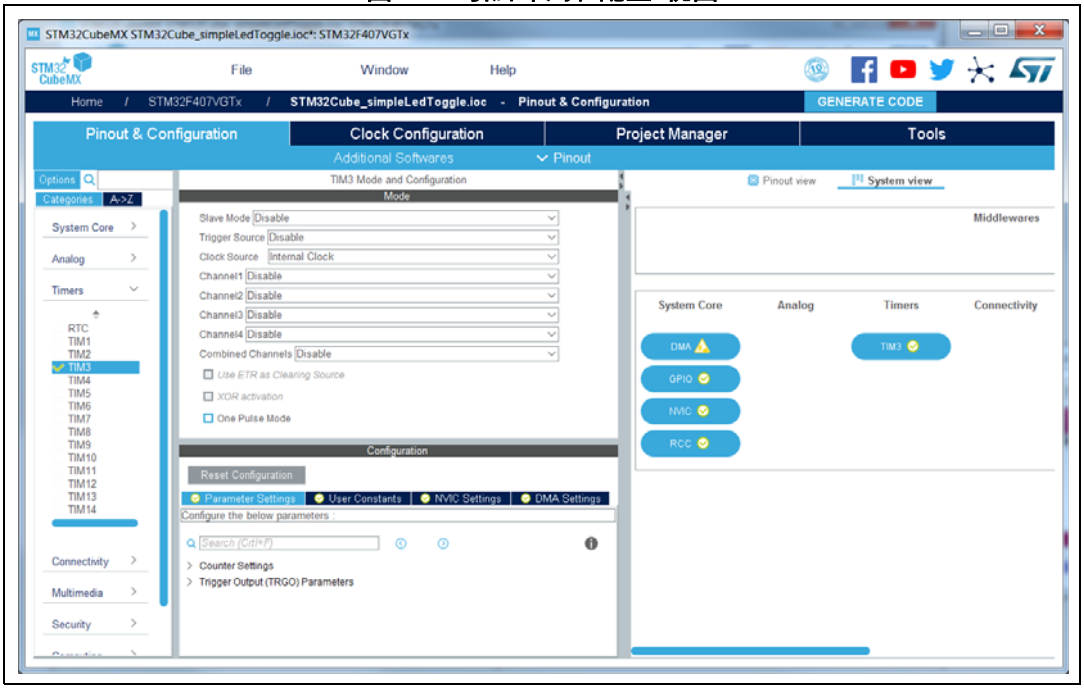
11.6.1 初始条件

在**“引脚布局和配置”**选项卡上，使用**“模式”**和**“配置”**面板选择并配置（逐个）应用程序所需的每个组件（外设、中间件、其他软件）（请参阅*图 213*）。

未正确配置外设时，将显示工具提示和警告消息（有关详细信息，请参见*第 4 节：STM32CubeMX用户界面*）。

注： ***RCC****外设初始化将使用在此视图中完成的参数配置以及在****时钟树****视图中完成的配置（时钟源、频率、预分频值等）。*

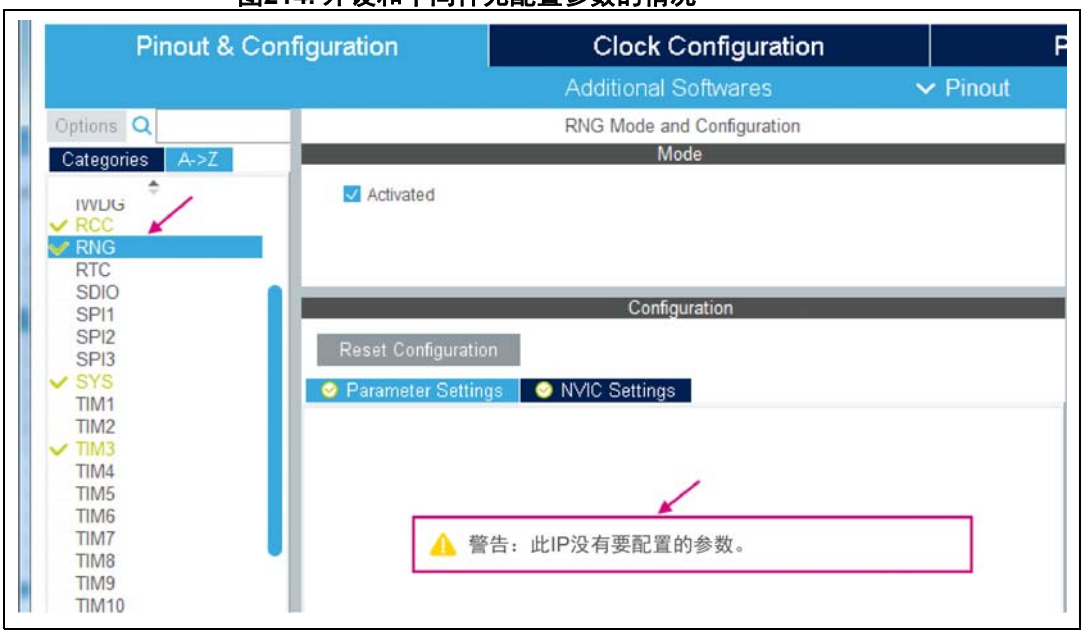
图213. “引脚布局和配置”视图



11.6.2 配置外设

每个外设实例与主面板中的专用按钮对应。某些外设模式没有可配置的参数，如下图所示。

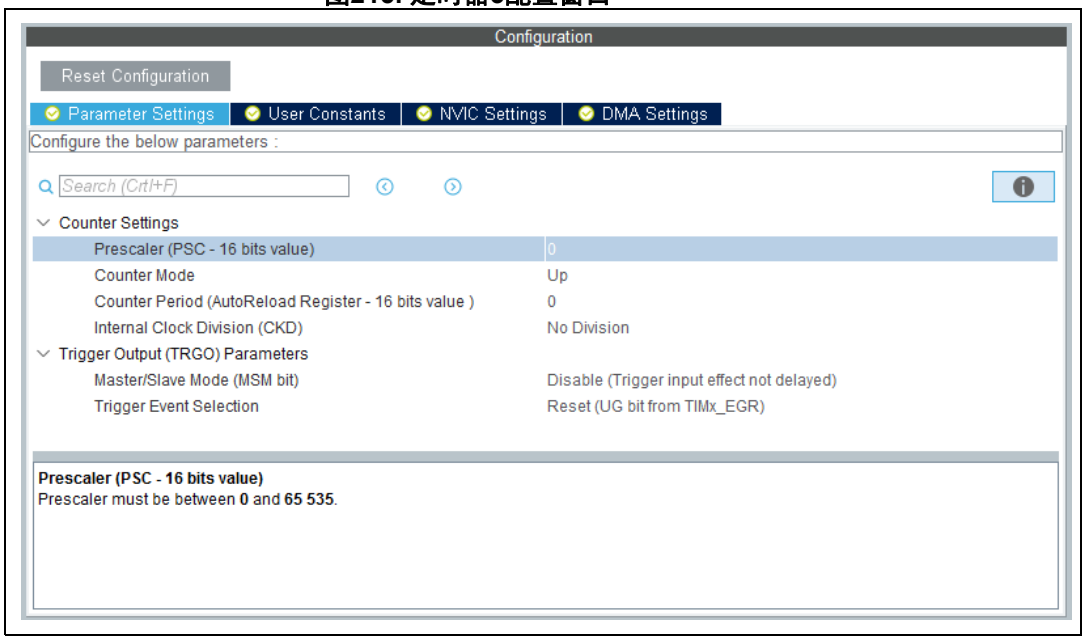
图214. 外设和中间件无配置参数的情况



按照以下步骤继续进行外围设备配置：

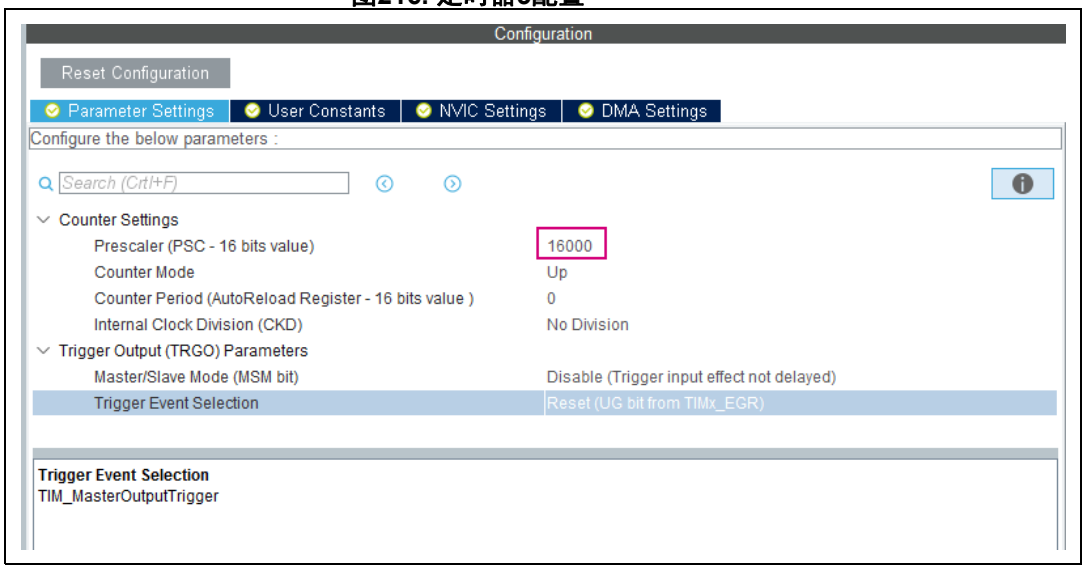
1. 单击外设按钮以打开相应的配置窗口。
   在我们的示例中
   a) 点击**TIM3**以打开定时器配置窗口。

图215. 定时器3配置窗口



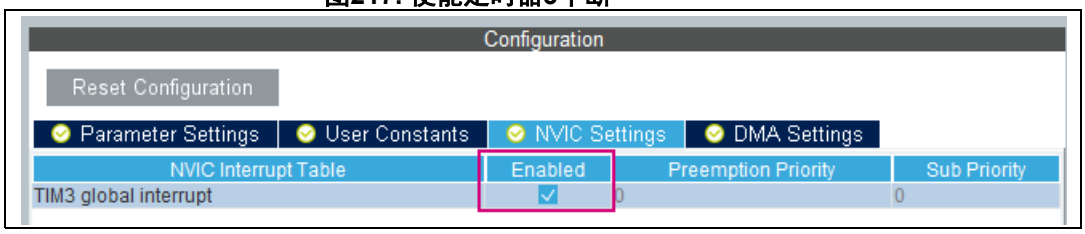
b) 通过16 MHz APB时钟（时钟树视图），将预分频器设为16000，并将计数器周期设为1000，以使LED每毫秒闪烁一次。

图216. 定时器3配置



2. 或者，如果可用，选择：
   - **NVIC设置**选项卡，以显示NVIC配置，并使能此外设的中断。
   - **DMA设置**选项卡，以显示DMA配置，并配置此外设的DMA传输。
     在教程示例中，未使用DMA，GPIO设置保持不变。使能中断，如*图 217*中所示。
   - **GPIO设置**选项卡用于显示GPIO配置，并为此外设配置GPIO。
   - 插入一个项目：
   - **用户常量**选项卡用于指定要在项目中使用的常量。

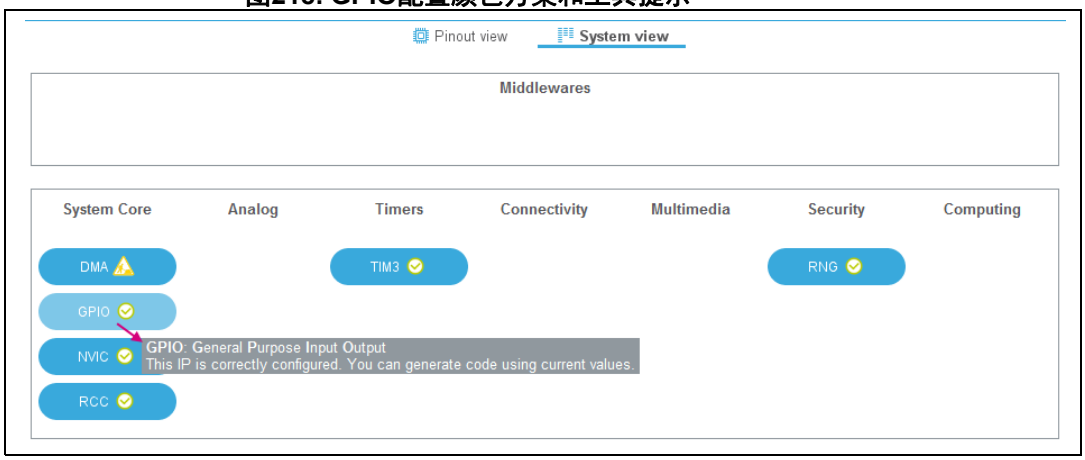
图217. 使能定时器3中断



11.6.3 配置GPIO

用户可通过此窗口调整所有引脚配置。通过小图标以及工具提示指示配置状态。

图218. GPIO配置颜色方案和工具提示

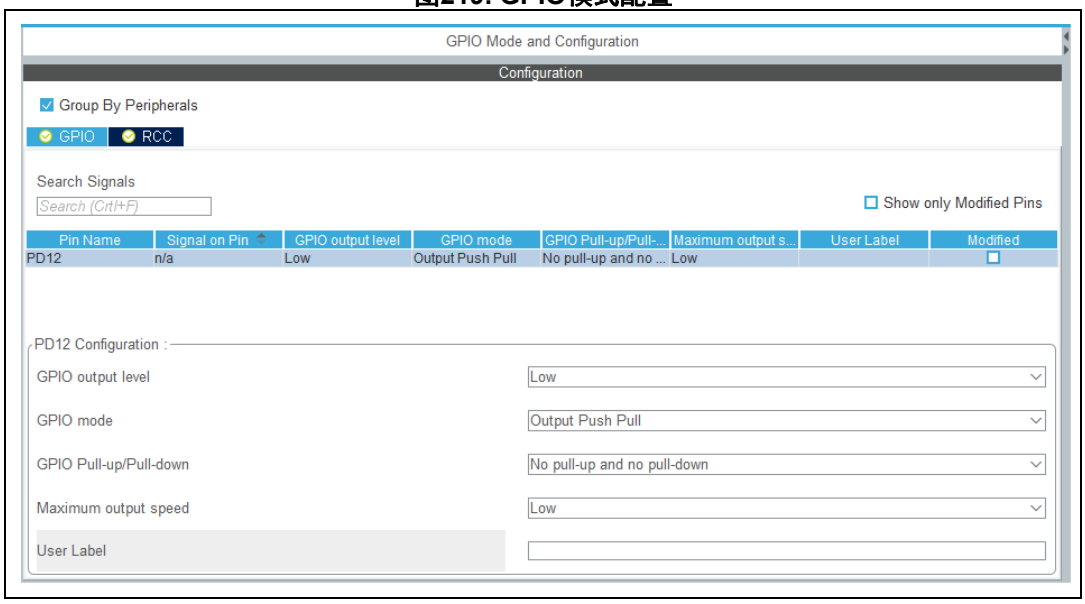


按照以下顺序配置GPIO：

1. 点击配置视图中的**GPIO按钮**，打开下面的**引脚配置**窗口。
2. 第一个选项卡显示已分配GPIO模式、但不用于专用外设和中间件的引脚。选择“引脚名称”，打开该引脚的配置。

   在教程示例中，选择PD12并将其配置为在输出推挽模式下驱动STM32F4DISCOVERY LED（参见*图 219*）。

图219. GPIO模式配置



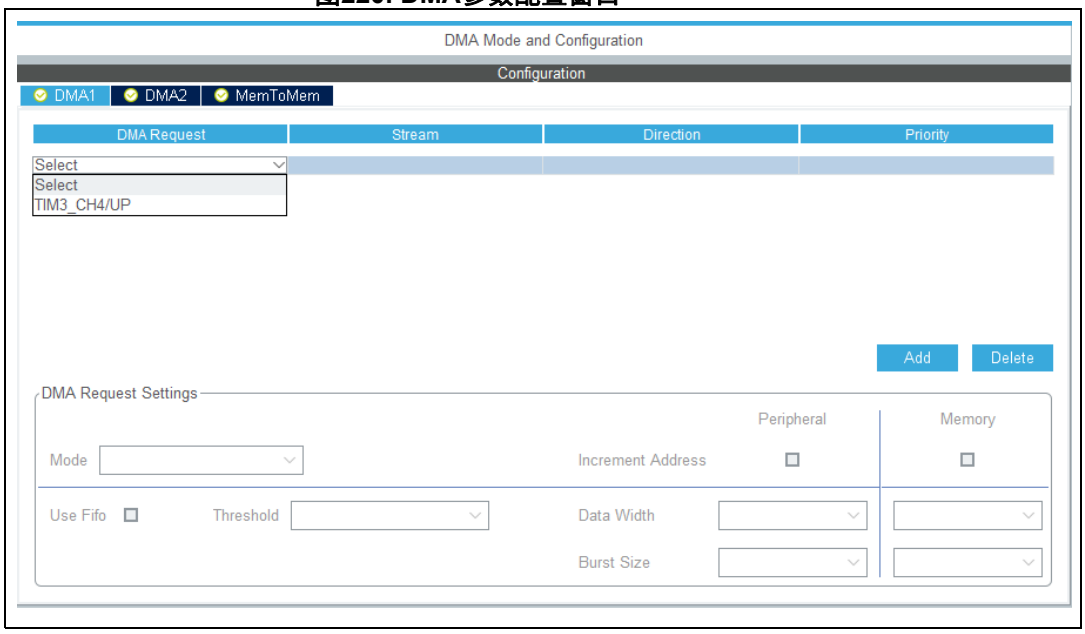
11.6.4 配置DMA

该示例无需此配置。建议使用DMA传输为CPU减荷。“DMA配置”窗口提供快速、简便的方式配置DMA（参见*图 220*）：

1. 添加新DMA请求，并在可行配置列表中选择。
2. 在现有的流中选择。
3. 选择“方向：内存到外设”或“外设到内存”。
4. 选择优先级。
5. 启用FIFO。

注： *还可使用“外设和中间件”配置窗口为给定外设和中间件配置DMA。*

图220. DMA参数配置窗口



11.6.5 配置中间件

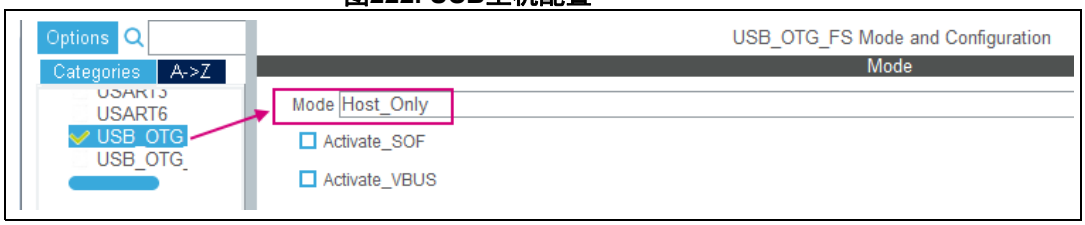
教程示例不需要配置中间件。

如果中间件模式需要使用外设，则必须在**引脚排列**视图中配置外设才能提供中间件模式。工具提示可以指导用户，如下所示。

图221. 中间件工具提示

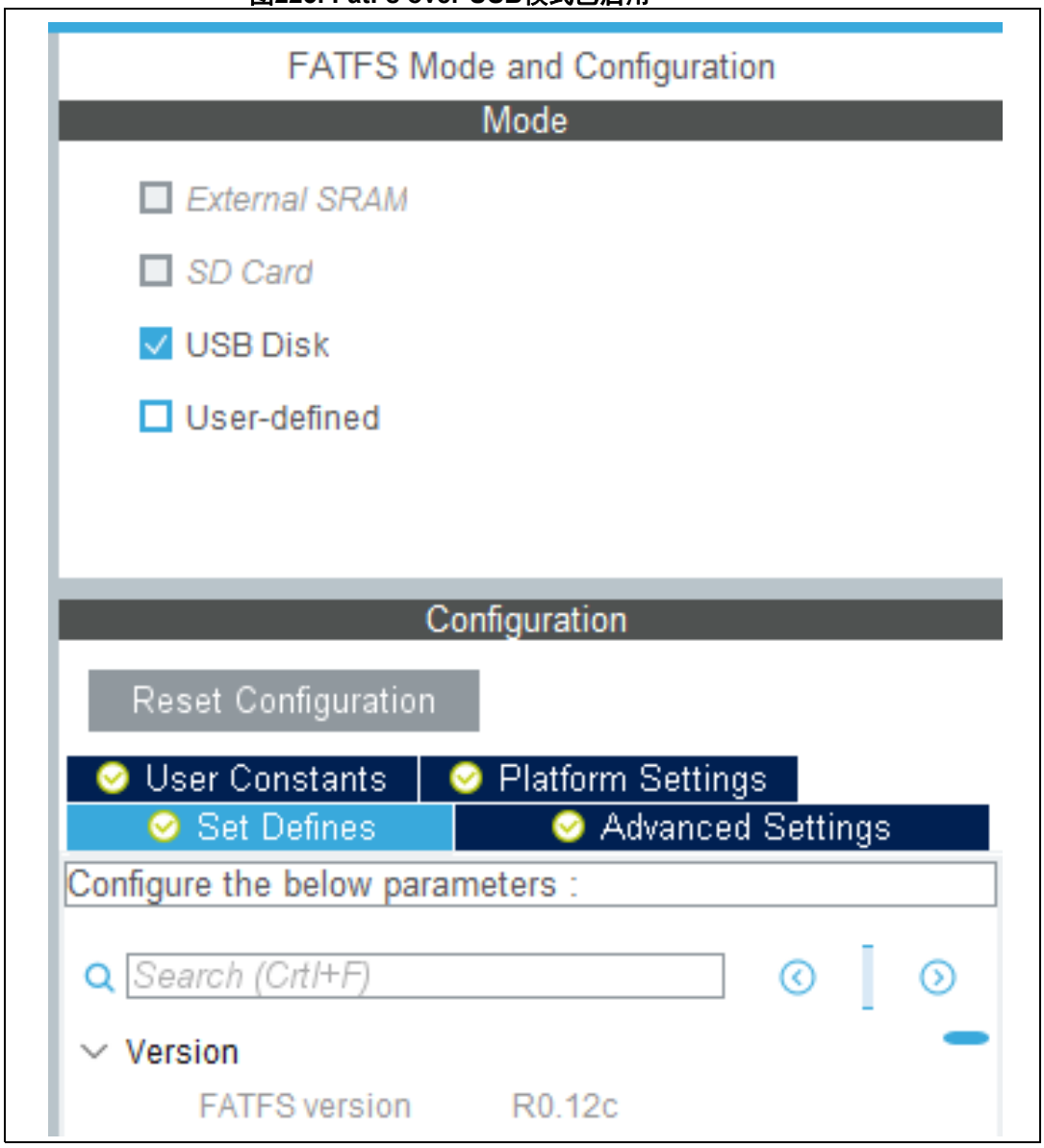
1. 通过**引脚排列**视图配置USB外设。

图222. USB主机配置



2. 从USB主机中间件选择MSC_FS类。
3. 选中此复选框在树形面板中启用FatFs USB模式。

图223. FatFs over USB模式已启用

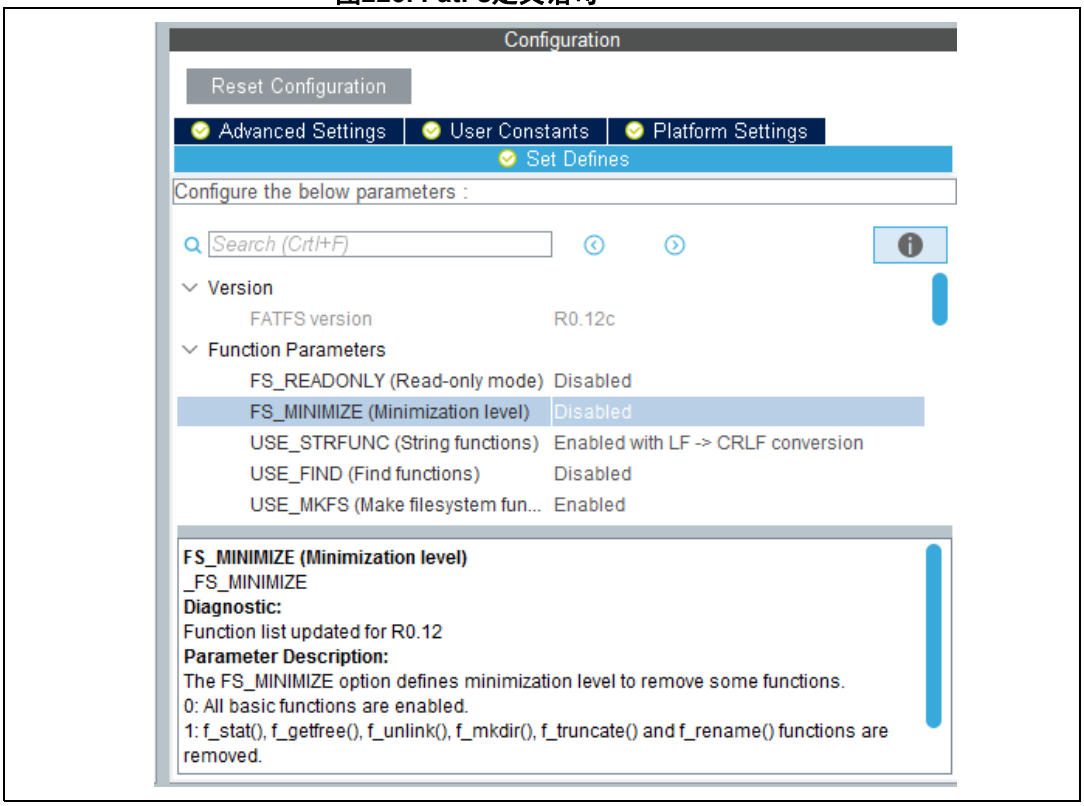


4. 选择**配置**视图。会显示FatFs和USB按钮。

图224. 启用FatF和USB的“系统”视图

5. 使用默认设置的FatFs和USB已标记为已配置 。点击**FatFs**和**USB**按钮显示默认配置设置。还可按照窗口底部提供的指南进行更改。

图225. FatFs定义语句



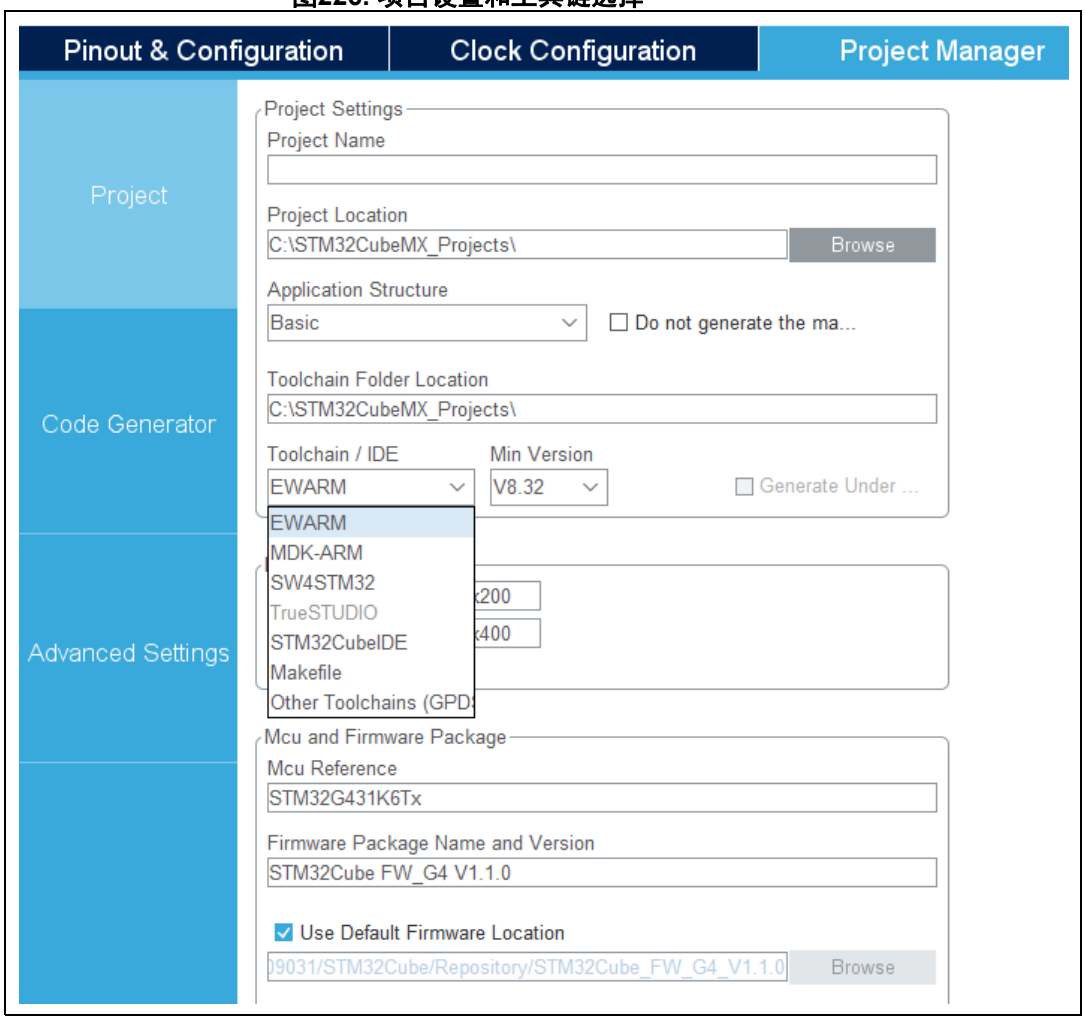
11.7 生成完整的C项目

11.7.1 设置项目选项

可以在生成C代码之前调整默认项目设置，如*图 226*中所示。

1. 选择“项目管理器”视图以更新项目设置和生成选项。
2. 选择**“项目”选项卡**，然后选择项目**名称**、**位置**、**工具链**和**工具链版本**以生成项目（请参阅*图 226*）。

图226. 项目设置和工具链选择

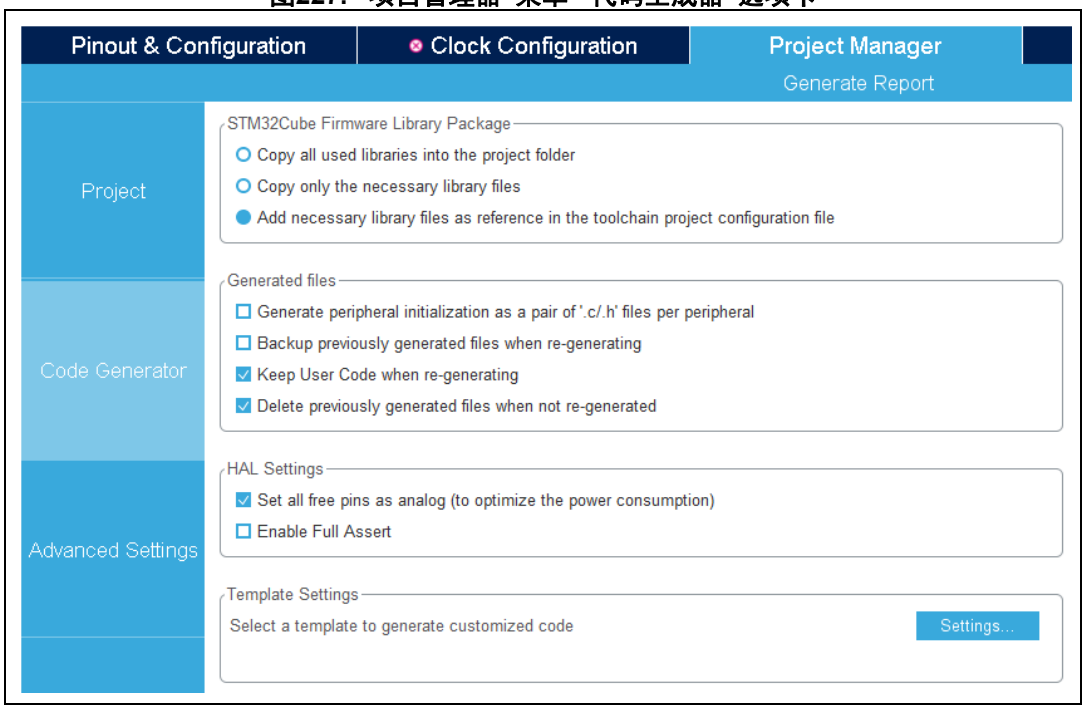


3. 选择**代码生成器**选项卡，以选择各种C代码生成选项：
   - 复制到*项目*文件夹的库文件。
   - 重新生成C代码（例如重新生成C代码过程中保留或备份的内容）。
   - HAL特定操作（例如，将所有空闲引脚设为模拟I/O，以降低MCU功耗）。

   在本教程示例中，选择*图 227*中显示的设置，然后单击**“确定”**。

注： *如果固件包缺失，会出现对话框窗口。请转到下一节了解如何下载固件包。*

图227. “项目管理器”菜单-“代码生成器”选项卡

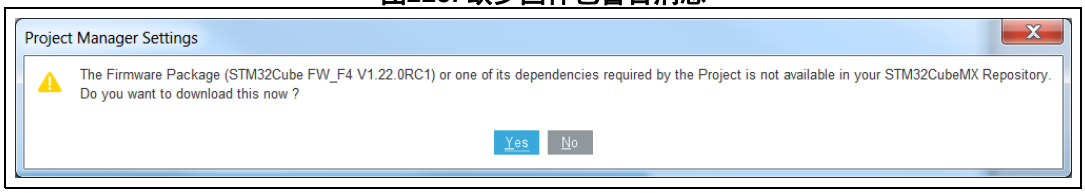


11.7.2 下载固件包和生成C代码

1. 点击 GENERATE CODE 生成C代码。

   C代码生成过程中，STM32CubeMX会将相关STM32Cube MCU包中的文件复制到项目文件夹，以便对项目进行编译。首次生成项目时，固件包在用户PC上不可用，会显示警告消息：

图228. 缺少固件包警告消息

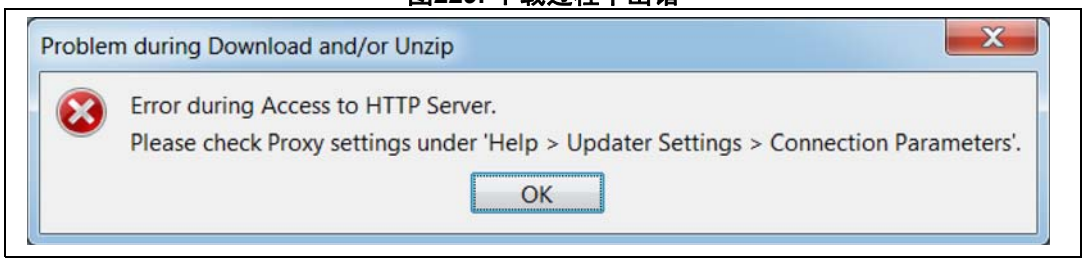


2. STM32CubeMX可供下载相关固件包或继续进行操作。点击**下载**获取完整项目，即准备好用于所选ID的项目。

   点击**继续**后，仅会创建*Inc*和*Src*文件夹，保留STM32CubeMX生成的初始化文件。需要手动复制必要的固件和中间件库才能获得完整项目。

   如果下载失败，会显示错误消息。

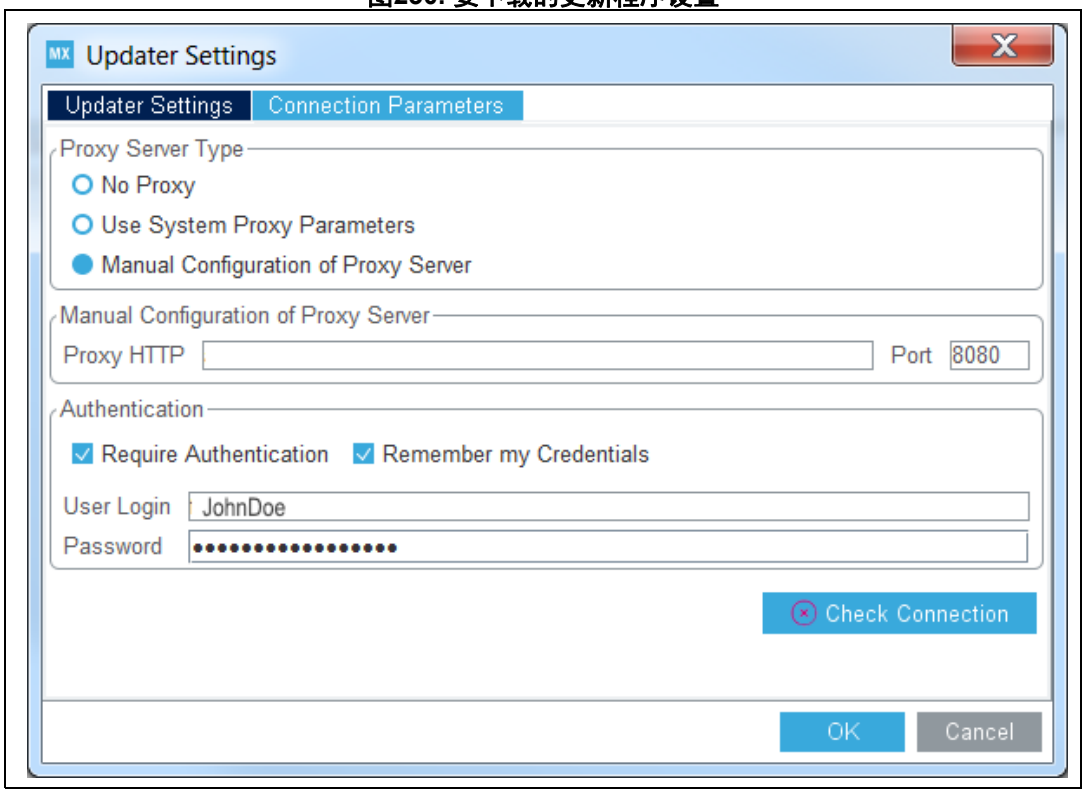
图229. 下载过程中出错



要解决此问题，请执行下面的两步操作。否则请跳过。

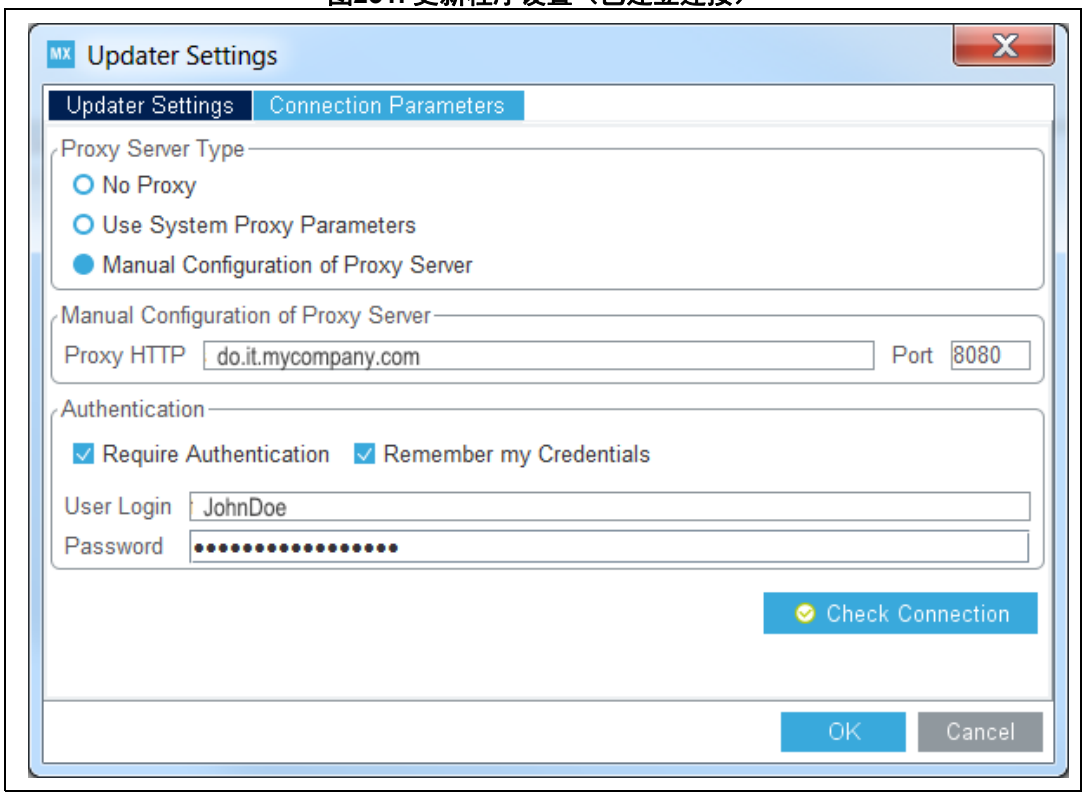
3. 选择**帮助 > 更新程序设置菜单**，并调整连接参数，使其与您的网络配置相匹配。

图230. 要下载的更新程序设置



4. 点击**检查连接。**连接建立后，选中标记会立即变为绿色。

图231. 更新程序设置（已建立连接）



5. 连接建立后，点击 GENERATE CODE 生成C代码。C代码生成过程开始并显示进度（请参见下图）。

图232. 下载固件包

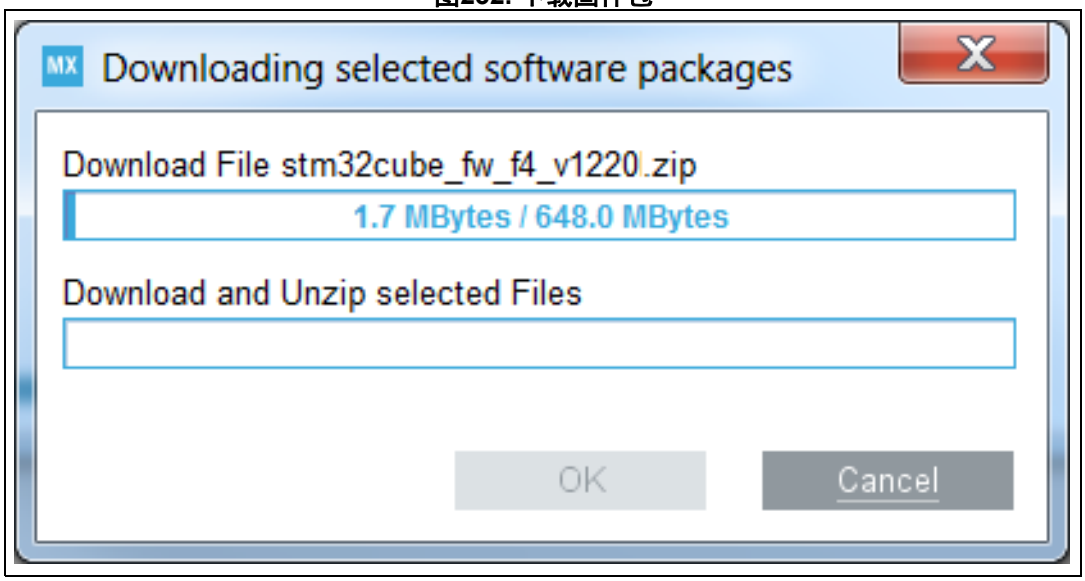
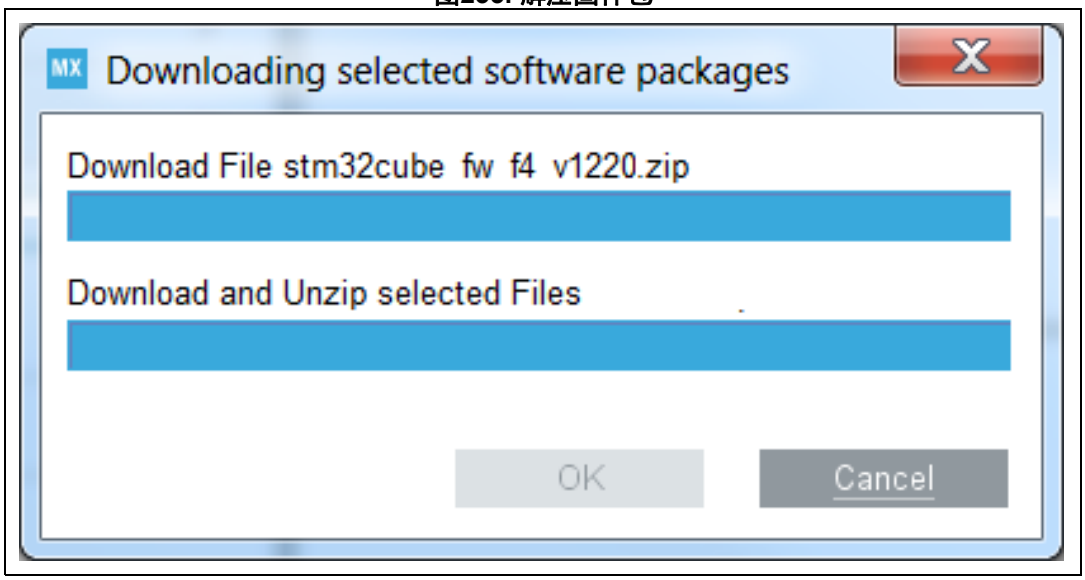
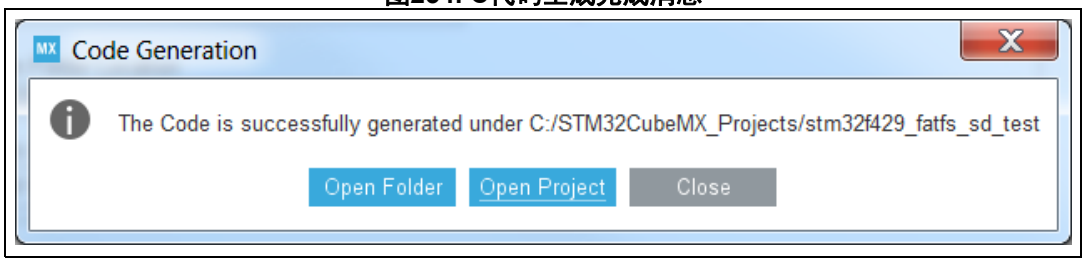


图233. 解压固件包



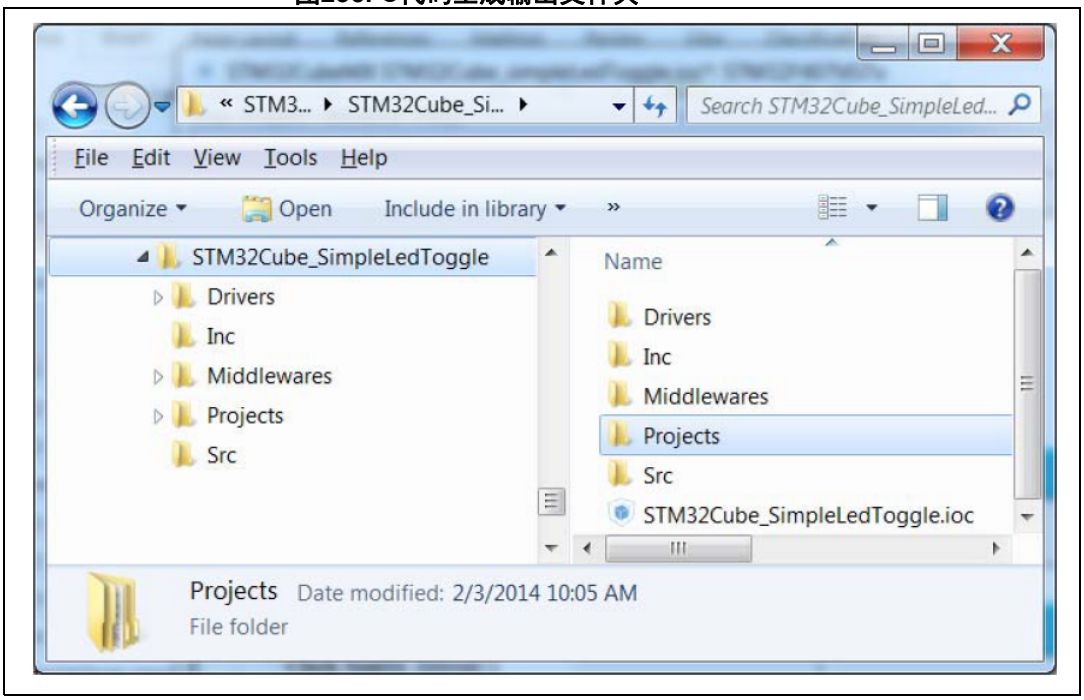
6. 最后会显示确认消息，指示C代码已成功生成。

图234. C代码生成完成消息



7. 点击**打开文件夹**显示生成的项目内容，或点击**打开项目**直接在IDE中打开项目。然后继续执行*第 11.8 节*中的操作。

图235. C代码生成输出文件夹



生成的项目包括：

- 位于根文件夹中的STM32CubeMX .ioc项目文件。
  该文件包含通过STM32CubeMX用户界面生成的项目用户配置和设置。
- *驱动程序*和*中间件*文件夹包含与用户配置相关的固件包文件的副本。
- *项目*文件夹包含IDE特有的文件夹，其中包含在IDE中进行项目开发和调试所需的所有文件。
- *Inc*和*Src*文件夹包含STM32CubeMX为中间件、外设和GPIO初始化生成的文件，包括main.c文件。STM32CubeMX生成的文件包含用户专用的允许插入用户自定义C代码的部分。

注意： 在该用户部分中编写的C代码会在下一次生成C代码时保留，而在这些部分以外编写的C代码则会被覆盖。

如果用户部分被移动或用户部分定界符被重命名，用户C代码将丢失。

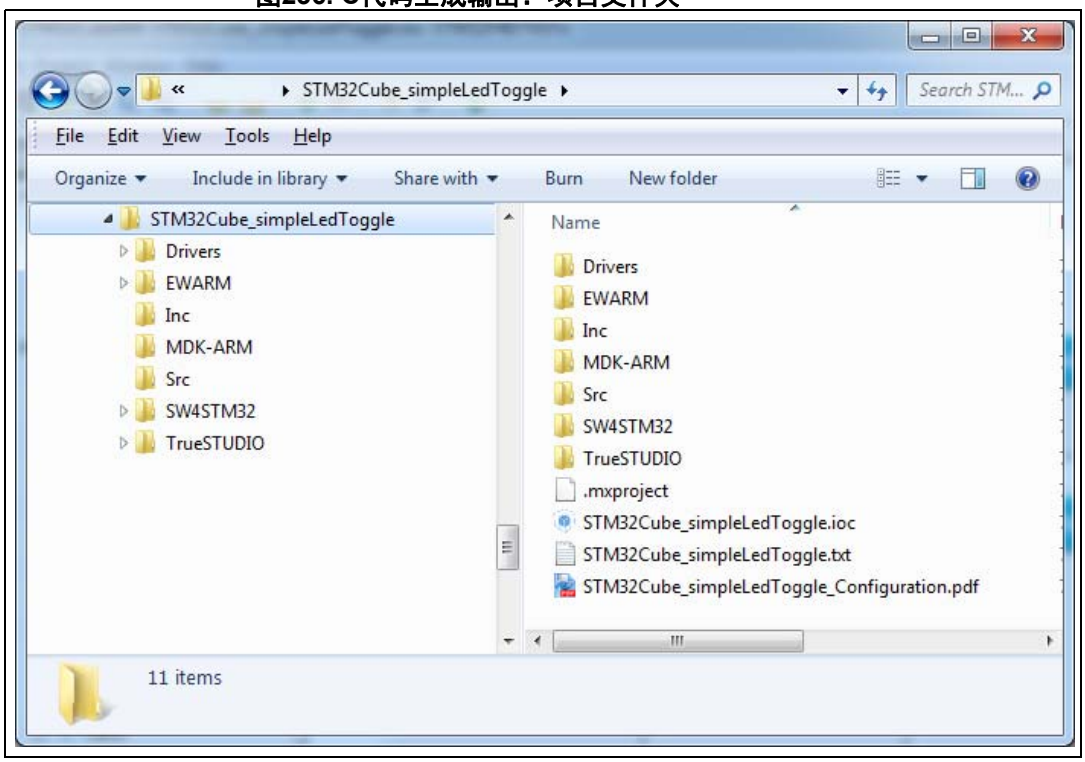
11.8 构建和更新C代码项目

本例介绍了如何在IAR™ EWARM工具链中使用生成的初始化C代码并完成项目，以使LED按照TIM3频率闪烁。

提供了一个文件夹可用于为生成C代码选择的工具链：可从**“项目管理器”**菜单中选择另一工具链并再次点击“生成代码”，为多个工具链生成项目。

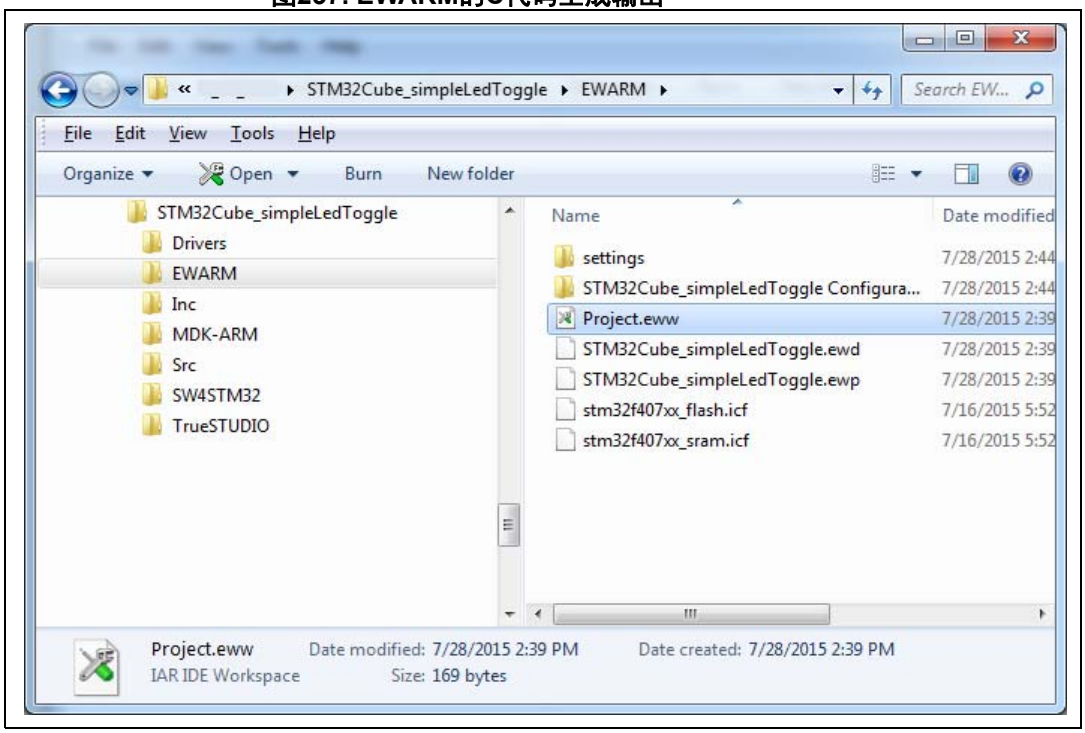
1. 点击对话框窗口中的**打开项目**，或双击STM32CubeMX下的工具链文件夹中提供的相关IDE文件，直接在IDE工具链中打开项目（参见*图 234*）。

图236. C代码生成输出：项目文件夹



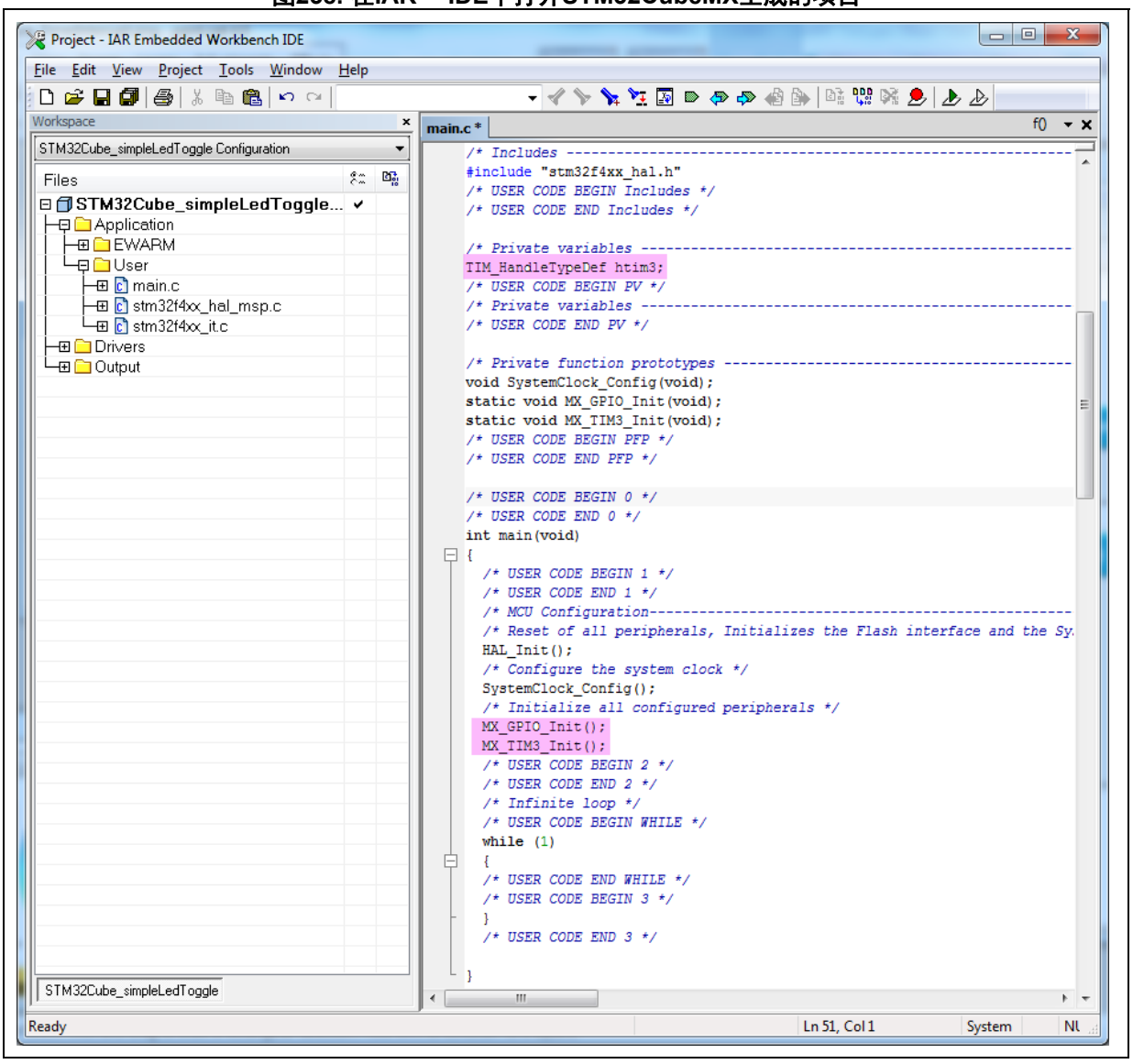
2. 例如，选择.eww文件可加载IAR™ EWARM IDE中的项目。

图237. EWARM的C代码生成输出



3. 选择main.c文件可在编辑器中打开该文件。

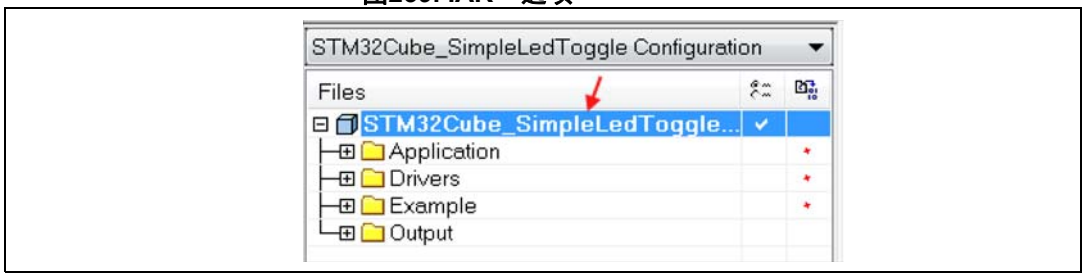
图238. 在IAR™ IDE中打开STM32CubeMX生成的项目



定义了htim3结构句柄、系统时钟、GPIO和TIM3初始化函数。初始化函数在main.c中调用。C代码部分目前为空。

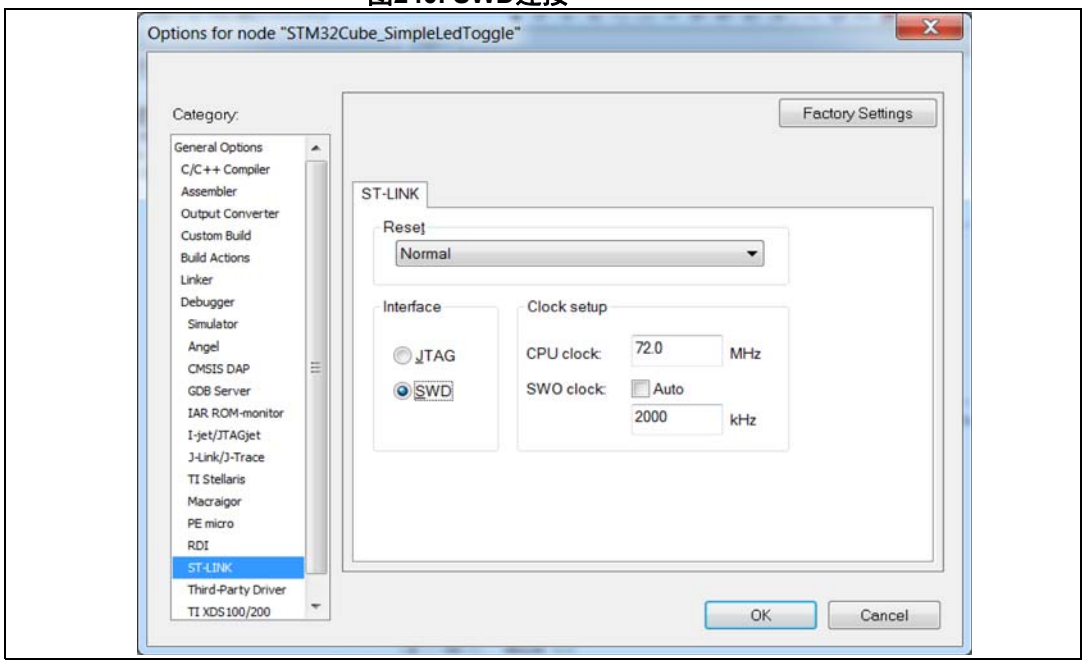
4. 在IAR™ IDE中，右键单击项目名称并选择**选项**。

图239. IAR™选项



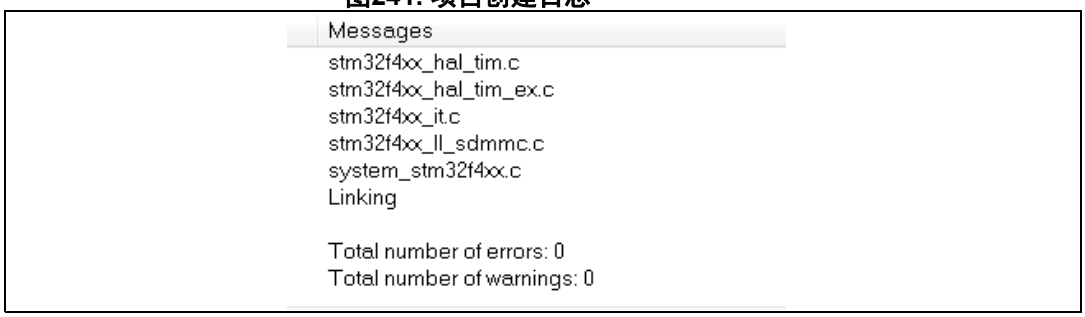
5. 点击ST-LINK类别，并确保已选择SWD与STM32F4DISCOVERY板进行通信。点击**确定**。

图240. SWD连接



6. 选择**项目> 全部重建**。检查项目创建是否成功。

图241. 项目创建日志



7. **仅可**在用户专用部分添加用户C代码。

注： *main while(1)循环放置在用户部分。*

例如：

a) 编辑main.c文件。

b) 要启动定时器3，请在用户部分2中更新以下C代码：

图242. 用户部分2

```
  HAL_Init();
  /* Configure the system clock */
  SystemClock_Config();
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_TIM3_Init();

  /* USER CODE BEGIN 2 */
  HAL_TIM_Base_Start_IT(&htim3);
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
```

c) 然后在用户部分4中添加以下C代码：

图243. 用户部分4

```
/* USER CODE BEGIN 4 */

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
  if ( htim->Instance == htim3.Instance )
  {
    HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_12);
  }
}
/* USER CODE END 4 */
```

这段C代码会实现在HAL定时器驱动程序(stm32f4xx_hal_tim.h)中定义的弱回调，在定时器计数周期结束时切换驱动绿色LED的GPIO引脚。

8. 使用  对板子进行重建和编程。确保SWD ST-LINK选项已被选中作为“项目”选项，否则板子编程将失败。
9. 使用  启动程序。STM32F4DISCOVERY板上的绿色LED将每秒闪烁一次。
10. 要更改MCU配置，请返回STM32CubeMX用户界面，执行更改并重新生成C代码。如果启用了“项目管理器”的“代码生成器”选项卡中的

选项，将更新项目，并在用户部分保留C代码。

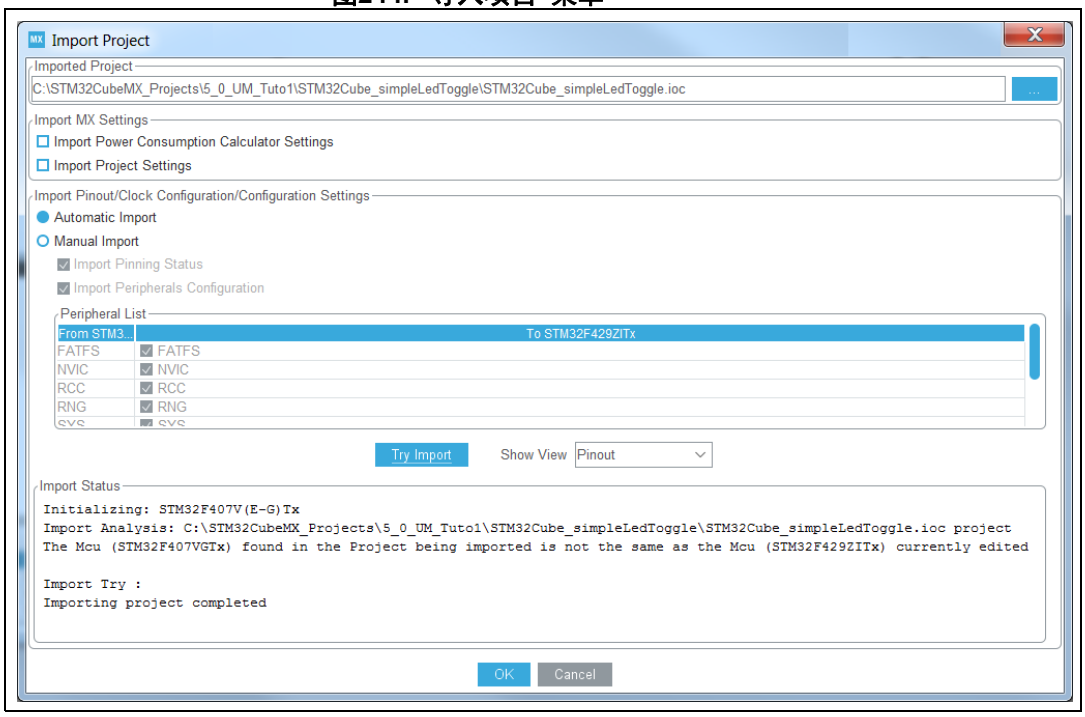
11.9 切换到另一MCU

STM32CubeMX允许将项目配置加载到同系列MCU上。

请按如下步骤操作：

1. 选择**文件 > 新项目**。
2. 选择属于相同系列的MCU。举例来说，可选择作为32F429IDISCOVERY板内核MCU的STM32F429ZITx。
3. 选择**文件 > 导入项目**。在**导入项目**窗口中，浏览到要加载的.ioc文件。显示一条消息，警告您当前选择的MCU(STM32F429ZITx)与在.ioc文件中指定的MCU(STM32F407VGTx)不同。提供多种导入选项建议（参见*图 244*）。
4. 点击**尝试导入**按钮并检查导入状态，以确认导入是否已成功。
5. 点击**“确定”**导入项目。随即会显示输出选项卡，报告导入结果。
6. 2F429IDISCOVERY板上的绿色LED连接至PG13：按CTRL+右键单击**PD12**，并将其拖放到PG13上。
7. 在**“项目管理器”**的“项目”选项卡中，配置新的项目名称和文件夹位置。点击**生成图标**保存项目并生成代码。
8. 在对话框窗口中选择**打开项目**，使用用户代码更新用户部分，请务必更新PG13的GPIO设置。构建项目并刷写板子。启动程序并检查LED是否每秒闪烁一次。

图244. “导入项目”菜单



12 教程2 - 使用STM32429I-EVAL评估板的SD卡上的FatFs示例STM32429I-EVAL 评估板

教程包括在使用FatFs文件系统中间件的STM32429I-EVAL1 SD卡上创建文件和写入文件。

要生成项目并运行教程2，请按照以下顺序操作：

1. 启动STM32CubeMX。
2. 选择**文件 > 新项目**。“项目”窗口将打开。
3. 点击**板选择器**选项卡显示ST板列表。
4. 选择**评估板**作为板子类型，选择**STM32F4**作为“系列”，对列表进行筛选。
5. 回答“是”，以默认方式初始化所有外设，以便仅为应用程序使用的外设生成代码。
6. 选择STM32429I-EVAL板并点击**“确定”**。在询问是否将所有外设初始化为其默认模式的对话框中回答“否”（参见*图 245*）。载入**引脚排列**视图，该视图与评估板上的MCU引脚排列配置匹配（参见*图 246*）。

图245. 板子外设初始化对话框

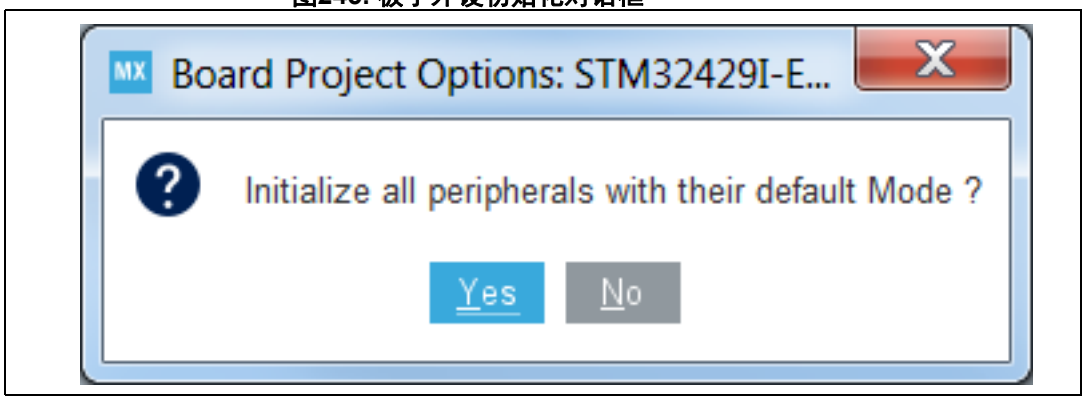
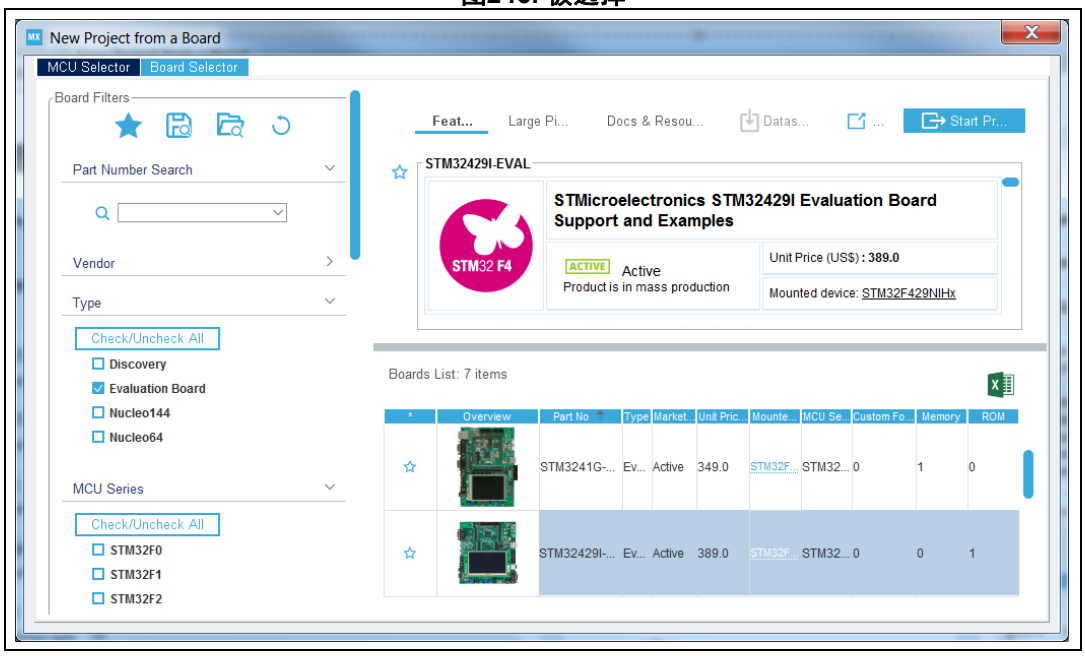
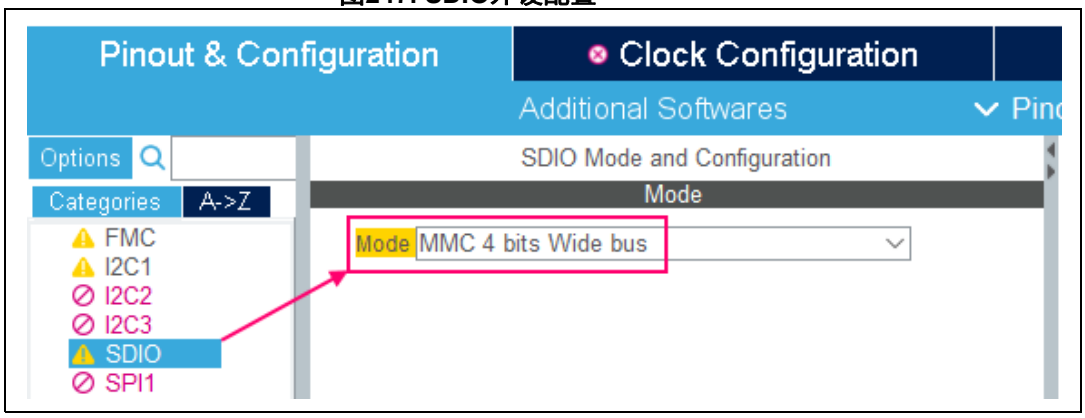


图246. 板选择



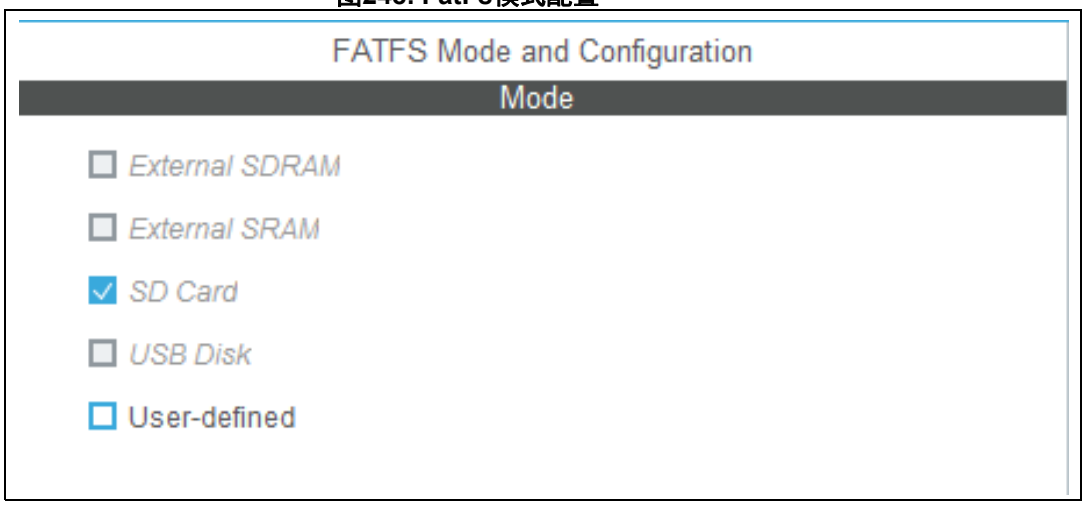
7. 在左侧的“外设”树中，展开SDIO外设并选择SD 4位宽总线（参见*图 247*）。

图247. SDIO外设配置



8. 在“中间件”类别下，FatFs模式选择**“SD卡”**（参见*图 248*）。

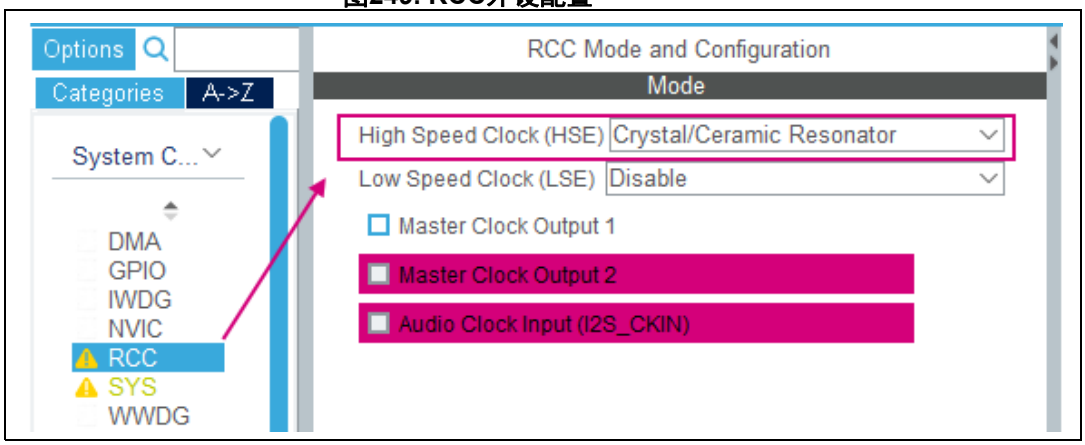
图248. FatFs模式配置



9. 按以下方法配置时钟：

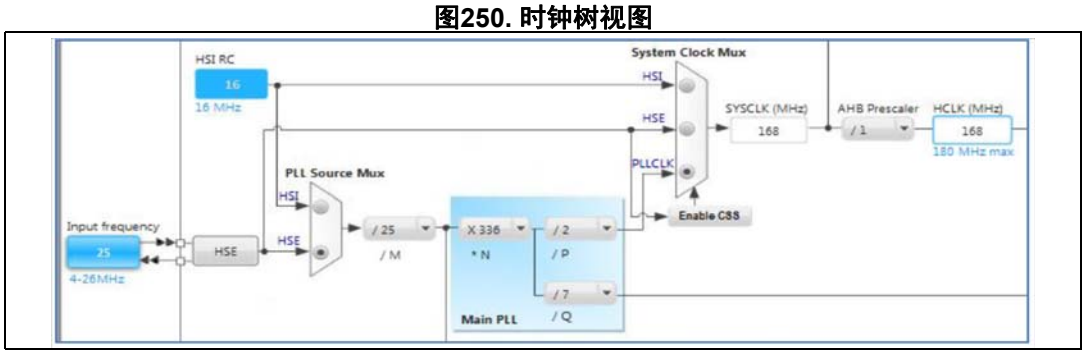
a) 从**引脚排列**视图中选择RCC外设（参见*图 249*）。

图249. RCC外设配置



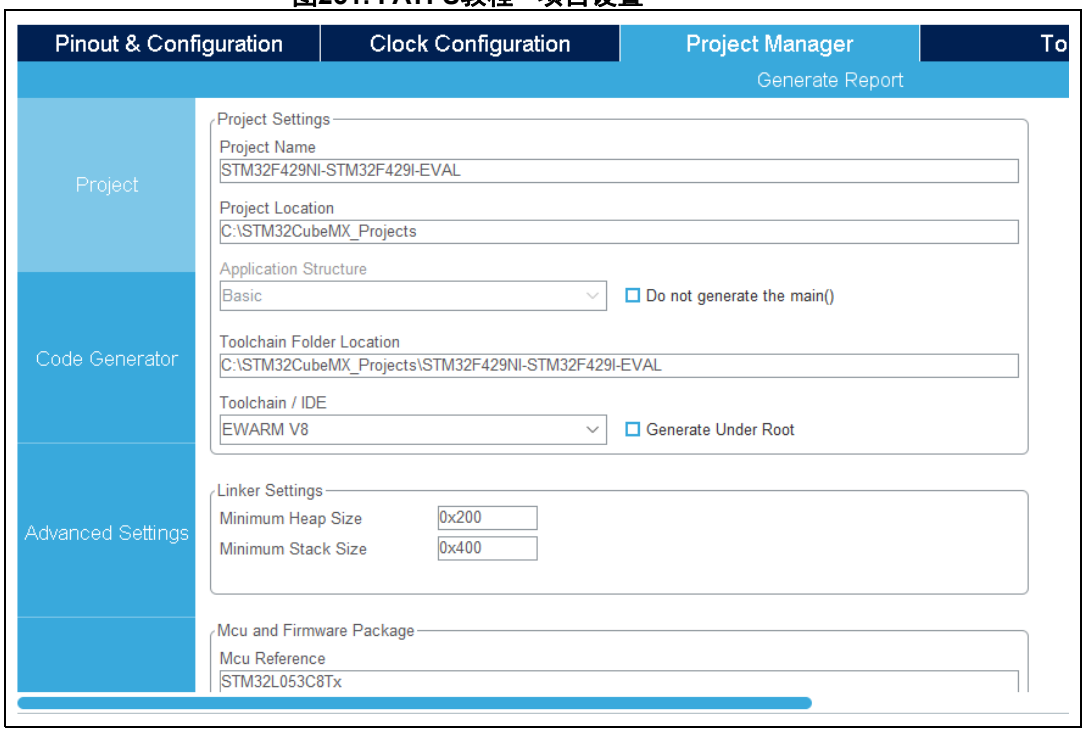
b) 在时钟选项卡中配置时钟树（参见*图 250*）。

图250. 时钟树视图



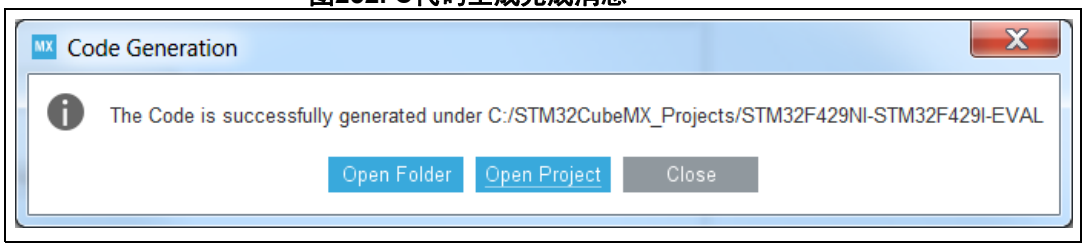
10. 在**项目**选项卡中，指定项目名称和目标文件夹。然后选择EWARM IDE工具链。

图251. FATFS教程 - 项目设置



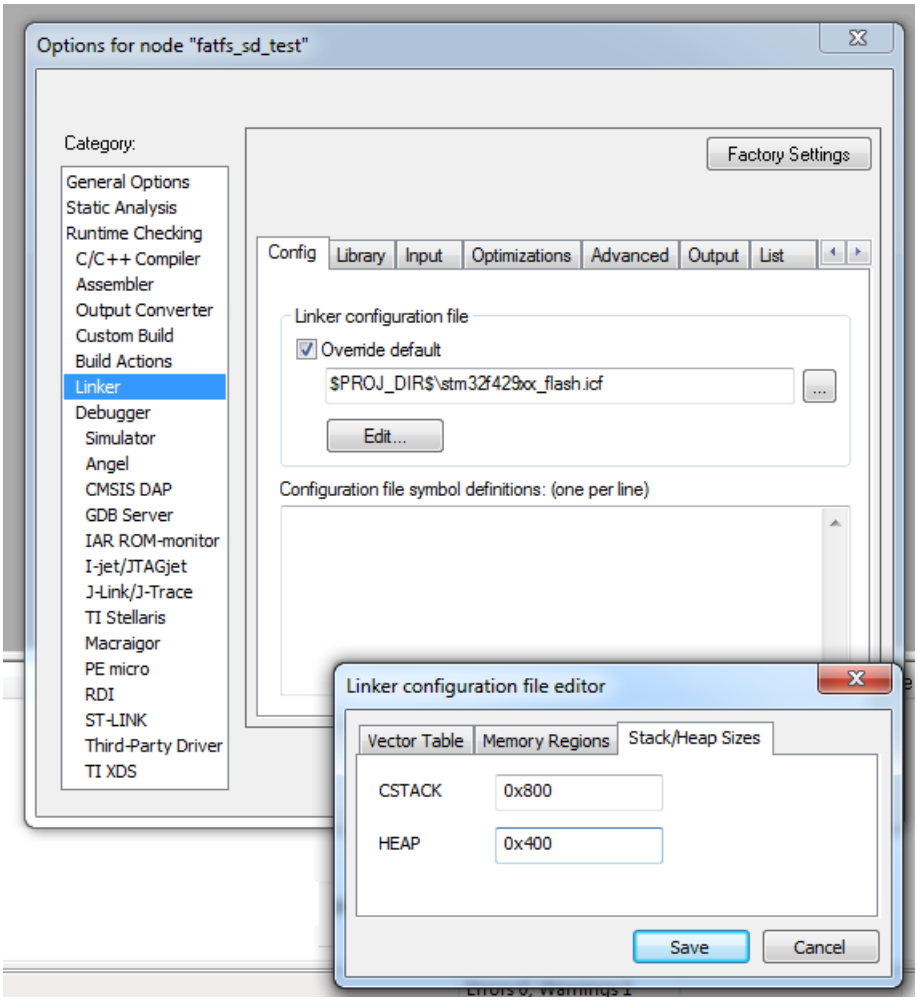
11. 点击**确定**。然后在工具栏菜单中点击 GENERATE CODE 生成项目。
12. 代码生成完毕后，点击**代码生成**对话框窗口中的**打开项目**（参见*图 252*）。此操作会直接在IDE中打开项目。

图252. C代码生成完成消息



13. 在IDE中，检查堆和堆栈大小是否足够大：右键单击项目名称并选择**“选项”**，然后选择**“链接器”**。选中**覆盖默认**，使用STM32CubeMX生成的项目文件夹中的icf文件。如果尚未通过CubeMX用户界面完成操作（在“项目管理器”的“项目”选项卡的“链接器设置”下），则调整堆和堆栈的大小（请参阅*图 253*）。

图253. IDE工作空间



注： *使用MDK-Arm工具链时，进入“应用/MDK-ARM”文件夹并双击startup_xx.s文件，在其中编辑和调整堆和堆栈大小。*

14. 进入“应用/用户”文件夹。双击main.c文件并对其进行编辑。
15. 教程包括在使用FatFs文件系统中间件的评估板SD卡上创建文件和写入文件：
    a) 启动时，所有LED均熄灭。
    b) 红色LED亮起，表示出现生错误（FatFs初始化、文件读取/写入访问错误...）。
    c) 橙色LED亮起，表示FatFs链接已成功安装在SD驱动程序上。
    d) 蓝色LED亮起，表示文件已成功写入到SD卡。
    e) 绿色LED亮起，表示已成功从SD卡中读取文件。
16. 要实现用例，请用以下代码更新main.c：
    a) 在专用用户代码部分插入main.c私有变量：

```
/* 用户代码开始 PV */
/* 私有变量 ---------------------------------------------*/
FATFS SDFatFs;  /* SD卡逻辑驱动器的文件系统对象 */
FIL MyFile;     /* 文件对象 */
const char wtext[] = "Hello World!";
const uint8_t image1_bmp[] = {
0x42,0x4d,0x36,0x84,0x03,0x00,0x00,0x00,0x00,0x00,0x36,0x00,0x00,0x00,
0x28,0x00,0x00,0x00,0x40,0x01,0x00,0x00,0xf0,0x00,0x00,0x00,0x01,0x00,
0x18,0x00,0x00,0x00,0x00,0x00,0x00,0x84,0x03,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x29,0x74,
0x51,0x0e,0x63,0x30,0x04,0x4c,0x1d,0x0f,0x56,0x25,0x11,0x79,0x41,0x1f,
0x85,0x6f,0x25,0x79,0x7e,0x27,0x72,0x72,0x0b,0x50,0x43,0x00,0x44,0x15,
0x00,0x4b,0x0f,0x00,0x4a,0x15,0x07,0x50,0x16,0x03,0x54,0x22,0x23,0x70,
0x65,0x30,0x82,0x6d,0x0f,0x6c,0x3e,0x22,0x80,0x5d,0x23,0x8b,0x5b,0x26};
/* 用户代码结束 PV */
```

    b) 插入主函数本地变量：

```
int main(void)
{

  /* 用户代码开始 1 */
  FRESULT res;            /* FatFs函数公共结果代码 */
  uint32_t byteswritten, bytesread; /* 文件写/读计数 */
  char rtext[256];            /* 文件读取缓冲区*/
  /* 用户代码结束 1 */

  /* MCU配置----------------------------------------*/

  /* 复位所有外设，初始化Flash接口和Systick。 */
HAL_Init();
```

    c) 在主函数的初始化调用之后、while循环之前插入用户代码，对SD卡执行实际读取/写入操作：

```
int main(void)
{
```

```
....
MX_FATFS_Init();

  /* 用户代码开始 2 */
/*##-0- 关闭所有LED（红、绿、橙和蓝） */
   HAL_GPIO_WritePin(GPIOG, (GPIO_PIN_10 | GPIO_PIN_6 | GPIO_PIN_7 |
GPIO_PIN_12), GPIO_PIN_SET);
/*##-1- FatFS：链接SD磁盘I/O驱动程序 ##########*/
   if(retSD == 0){
          /* 成功：设置橙色LED开启*/
     HAL_GPIO_WritePin(GPIOG, GPIO_PIN_7, GPIO_PIN_RESET);
/*##-2- 将文件系统对象寄存到FatFs模块 ###*/
   if(f_mount(&SDFatFs, (TCHAR const*)SD_Path, 0) != FR_OK){
    /* FatFs初始化错误：设置红色LED开启 */
      HAL_GPIO_WritePin(GPIOG, GPIO_PIN_10, GPIO_PIN_RESET);
      while(1);
    }  else   {
/*##-3- 在逻辑驱动器#上创建一个FAT文件系统*/
 /* 警告：格式化uSD卡将删除设备上的所有内容 */
  if(f_mkfs((TCHAR const*)SD_Path, 0, 0) != FR_OK){
   /* FatFs格式错误：设置红色LED开启 */
   HAL_GPIO_WritePin(GPIOG, GPIO_PIN_10, GPIO_PIN_RESET);
   while(1);
  } else {
/*##-4- 创建并打开一个有写权限#的新文本文件*/
  if(f_open(&MyFile, "Hello.txt", FA_CREATE_ALWAYS | FA_WRITE) != FR_OK){
  /* 打开并写入'Hello.txt'文件错误：设置红色LED开启 */
  HAL_GPIO_WritePin(GPIOG, GPIO_PIN_10, GPIO_PIN_RESET);
  while(1);
     } else {
 /*##-5- 将数据写入文本文件 ####################*/
   res = f_write(&MyFile, wtext, sizeof(wtext), (void *)&byteswritten);
  if((byteswritten == 0) || (res != FR_OK)){
   /* 'Hello.txt'文件写入或EOF错误：设置红色LED开启 */
   HAL_GPIO_WritePin(GPIOG, GPIO_PIN_10, GPIO_PIN_RESET);
   while(1);
   } else {
 /*##-6- 成功打开/写入：设置蓝色LED开启 */
   HAL_GPIO_WritePin(GPIOG, GPIO_PIN_12, GPIO_PIN_RESET);
   f_close(&MyFile);
 /*##-7- 打开有读取权限#的文本文件*/
   if(f_open(&MyFile, "Hello.txt", FA_READ) != FR_OK){
  /* 打开并读取'Hello.txt'文件错误：设置红色LED开启 */
   HAL_GPIO_WritePin(GPIOG, GPIO_PIN_10, GPIO_PIN_RESET);
   while(1);
   } else {
 /*##-8- 从文本文件#########读取数据*/
   res = f_read(&MyFile, rtext, sizeof(wtext), &bytesread);
   if((strcmp(rtext,wtext)!=0)|| (res != FR_OK)){
  /* 'Hello.txt'文件读取或EOF错误：设置红色LED开启 */
   HAL_GPIO_WritePin(GPIOG, GPIO_PIN_10, GPIO_PIN_RESET);
   while(1);
```

```
    } else {
   /* 成功读取：设置绿色LED开启 */
    HAL_GPIO_WritePin(GPIOG, GPIO_PIN_6, GPIO_PIN_RESET);
  /*##-9- 关闭打开的文本文件#################*/
    f_close(&MyFile);
  }}}}}}}
  /*##-10- 断开micro SD磁盘I/O驱动程序#########*/
    FATFS_UnLinkDriver(SD_Path);

  /* 用户代码结束 2 */

  /* 无限循环 */
  /* 用户代码开始WHILE */
while (1)
```

13 教程 3 - 使用功耗计算器优化嵌入式应用功耗等

13.1 教程概述

本教程侧重于介绍STM32CubeMX功耗计算器的特性及其优势，以评估节能技巧对给定应用序列的影响。

降低给定应用功耗的主要考虑因素包括：

- 降低工作电压
- 缩短耗能模式的用时

  由开发人员选择可在低功耗和性能之间实现最佳平衡的配置。
- 最大限度地延长非活动模式和低功耗模式的时间
- 使用最佳时钟配置

  如果微控制器需长期处于有效工作模式以执行指定操作，则降低工作频率会提高能耗，因而内核应始终以相对良好的速度运行。
- 仅启用与当前应用状态相关的外设，并对其他外设进行时钟选通
- 在适当的情况下，使用具有低功耗特性的外设（例如通过I2C唤醒微控制器）
- 最大限度地减少状态转换次数
- 优化代码执行过程中的存储器访问
  - 首选执行RAM中的代码，而非Falsh存储器中的代码
  - 在适当的情况下，考虑使CPU频率与Flash存储器的工作频率匹配，以实现零等待状态。

以下教程介绍了如何借助STM32CubeMX功耗计算器功能对应用进行调试，以最大限度地降低其功耗并延长电池使用寿命。

注： *功耗计算器不考虑I/O动态电流消耗以及同样会影响电流消耗的外部板组件。为此，为用户提供了"其他功耗"字段来指定此类功耗值。*

13.2 应用程序示例说明

设计应用时使用的NUCLEO-L476RG板基于STM32L476RGTx器件，并由2.4 V电池供电。

该应用的主要目的是执行ADC测量并通过UART传输转换结果。该应用使用：

- 多种低功耗模式：低功耗运行、低功耗睡眠、睡眠、停机和待机
- 多个外设：USART、DMA、定时器、COMP、DAC和RTC
  - RTC用于运行日历，以及在指定时间结束时将CPU从待机模式下唤醒。
  - DMA会将ADC测量结果从ADC传送到存储器
  - USART与DMA结合使用，通过虚拟COM端口发送/接收数据，并将CPU从停机模式中唤醒。

优化此类复杂应用的过程是先引入只具有相应功能的序列，然后分步引入STM32L476RG微控制器提供的低功耗特性。

13.3 使用功耗计算器

13.3.1 创建功耗系列

请按照以下步骤创建序列（参见*图 254*）：

1. 启动STM32CubeMX。
2. 点击**新项目**，并从**板**选项卡中选择Nucleo-L476RG板。
3. 点击**功耗计算器**选项卡，选择“功耗计算器”视图。随即会创建第一个序列作为参考。
4. 对其进行调整，以最大限度地降低整体电流消耗。为此：
   a) 选择2.4 V V_{DD}电源。该值可逐步调整（参见*图 255*）。
   b) 选择Li-MnO2(CR2032)电池。该步骤可选。电池类型可稍后更改（参见*图 255*）。

图254. 功耗计算示例

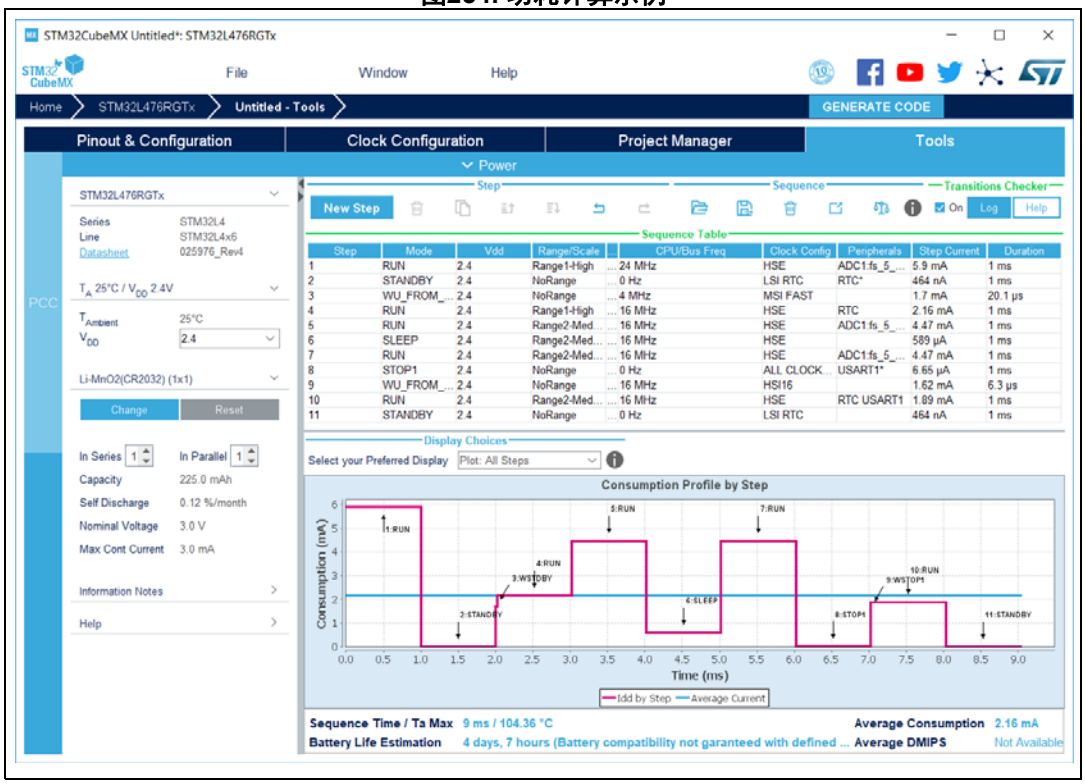
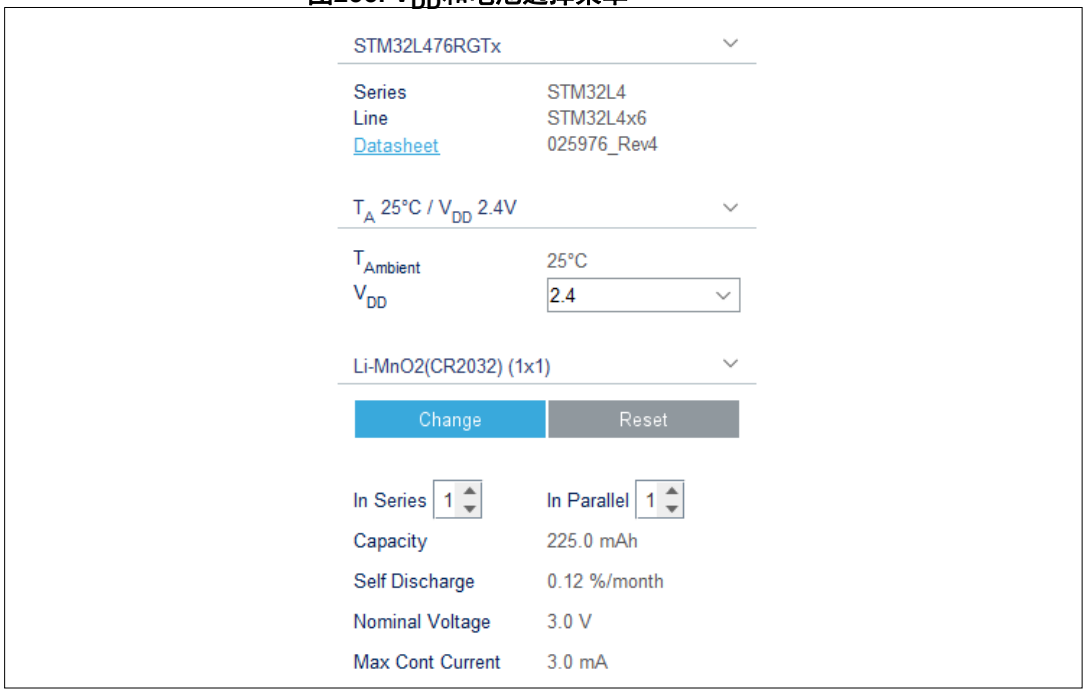


图255. V_{DD}和电池选择菜单



5. 使能**跳变检查器**确保序列有效（参见*图 255*）。此选项可用于验证序列是否符合允许在STM32L476RG中实现的转换。
6. 点击**添加**按钮添加与*图 255*中介绍的序列相匹配的步骤。
   – 各步骤默认持续1 ms，但使用产品数据手册中指定的转换时间预设的唤醒转换除外（参见*图 256*）。
   – 一些功耗未知或可忽略不计的外设将以“*”突出显示（参见*图 256*）。

图256. 序列表

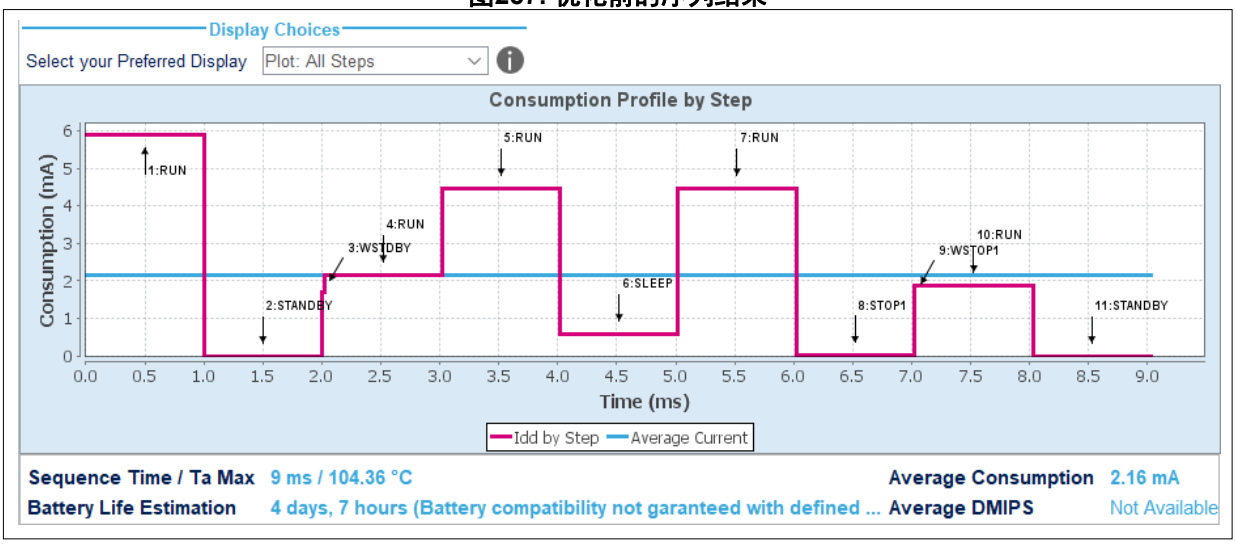
Sequence Table

Step	Mode	Vdd	Range/Scale	...	CPU/Bus Freq	Clock Config	Peripherals	Step Current	Duration
1	RUN	2.4	Range1-High	...	24 MHz	HSE	ADC1:fs_5_...	5.9 mA	1 ms
2	STANDBY	2.4	NoRange	...	0 Hz	LSI RTC	RTC*	464 nA	1 ms
3	WU_FROM_...	2.4	NoRange	...	4 MHz	MSI FAST		1.7 mA	20.1 µs
4	RUN	2.4	Range1-High	...	16 MHz	HSE	RTC	2.16 mA	1 ms
5	RUN	2.4	Range2-Med...	...	16 MHz	HSE	ADC1:fs_5_...	4.47 mA	1 ms
6	SLEEP	2.4	Range2-Med...	...	16 MHz	HSE		589 µA	1 ms
7	RUN	2.4	Range2-Med...	...	16 MHz	HSE	ADC1:fs_5_...	4.47 mA	1 ms
8	STOP1	2.4	NoRange	...	0 Hz	ALL CLOCK...	USART1*	6.65 µA	1 ms
9	WU_FROM_...	2.4	NoRange	...	16 MHz	HSI16		1.62 mA	6.3 µs
10	RUN	2.4	Range2-Med...	...	16 MHz	HSE	RTC USART1	1.89 mA	1 ms
11	STANDBY	2.4	NoRange	...	0 Hz	LSI RTC		464 nA	1 ms

7. 点击**保存**按钮将序列保存为SequenceOne。

会生成应用功耗曲线。曲线显示9 ms内序列的平均总功耗为2.01 mA，电池使用寿命仅为4天（参见*图 257*）。

图257. 优化前的序列结果



13.3.2 优化应用功耗

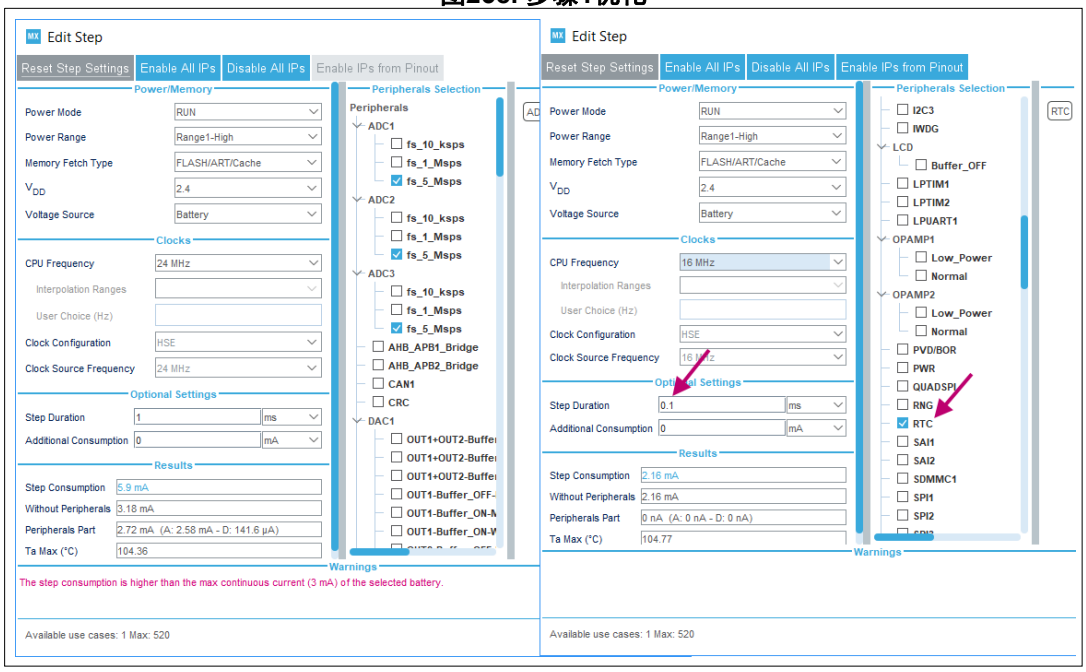
我们现在通过一些操作优化总功耗和电池使用寿命。这些操作在步骤1、4、5、6、7、8和10中执行。

下面的图形中，左侧图形显示原始步骤，右侧图形显示更新了多个优化操作的步骤。

步骤1（运行）

- 结果
  所有外设均已启用，但应用只需要使用RTC。
- 动作
  – 降低工作频率。
  – 只启用RTC外设。
  – 要降低平均电流消耗，请缩短在该模式下的用时。
- 结果
  电流从9.05 mA降低至2.16 mA（参见*图 258*）。

图258. 步骤1优化



步骤4（运行，RTC）

- 动作
  将该模式下的用时缩短为0.1 ms。

步骤5（运行，ADC，DMA，RTC）

- 动作
  - 切换为低功耗运行模式。
  - 降低工作频率。
- 结果

  电流消耗从6.17 mA降低至271 µA（参见*图 259*）。

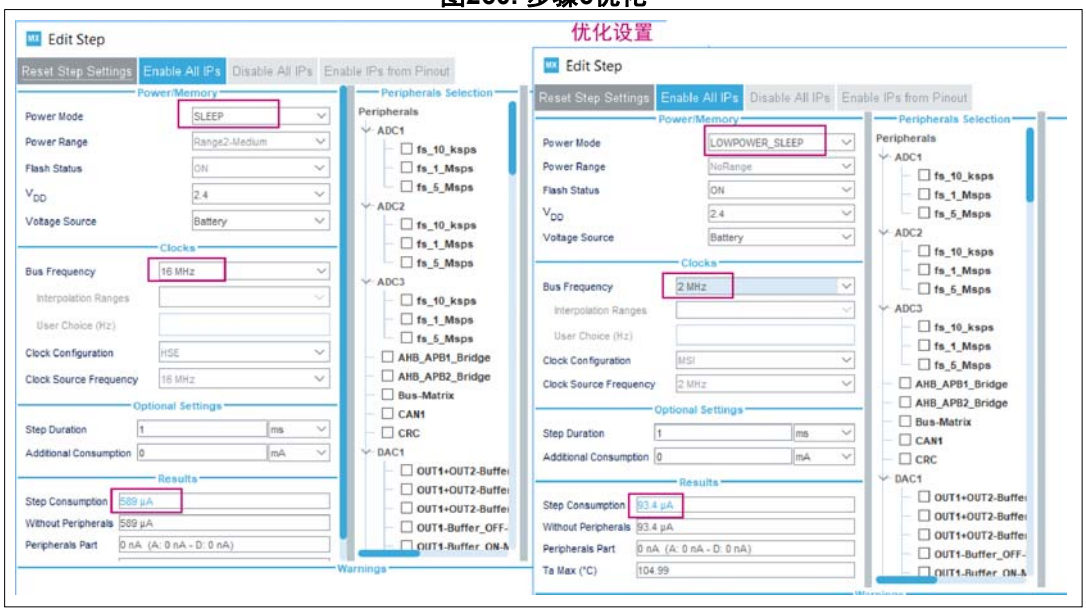
图259. 步骤5优化

步骤6（睡眠，DMA，ADC，RTC）

- 动作
  - 切换为功耗更低的睡眠模式（BAM模式）
  - 将工作频率降低至2 MHz。
- 结果

  电流消耗从703 µA降低至93 µA（参见*图 260*）。

图260. 步骤6优化

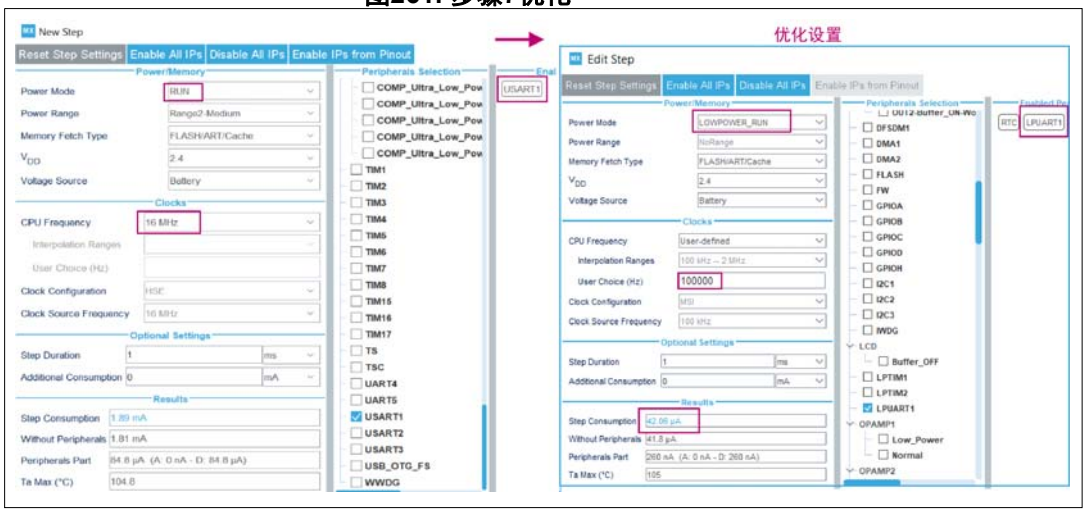


步骤7（运行，DMA，RTC，USART）

- 动作
  - 切换为功耗更低的运行模式。
  - 使用高能效的LPUART外设。
  - 使用插值功能将工作频率降低至1 MHz。
- 结果

  电流消耗从1.92 µA降低至42 µA（参见*图 261*）。

图261. 步骤7优化

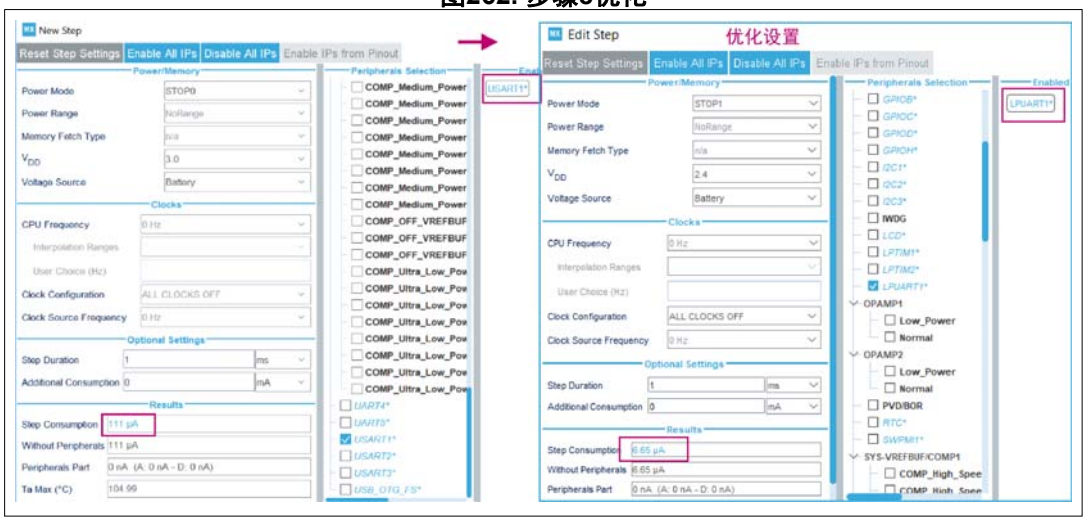


步骤8（停机0，USART）

- 动作
  - 切换为Stop1低功耗模式。
  - 使用高能效的LPUART外设。
- 结果

  减少了电流消耗（请参阅*图 262*）。

图262. 步骤8优化



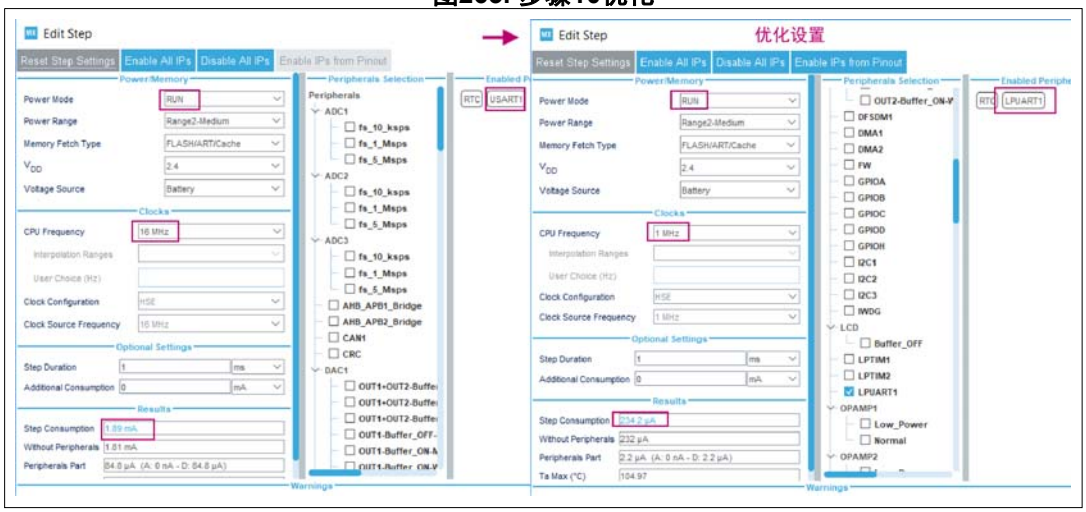
步骤10（RTC，USART）

- 动作
  - 使用高能效的LPUART外设。
  - 将工作频率降低至1 MHz。
- 结果

  电流消耗从1.89 mA降低至234 µA（参见*图 263*）。

  *图 264*中提供的示例显示平均电流消耗降低155 µA.。

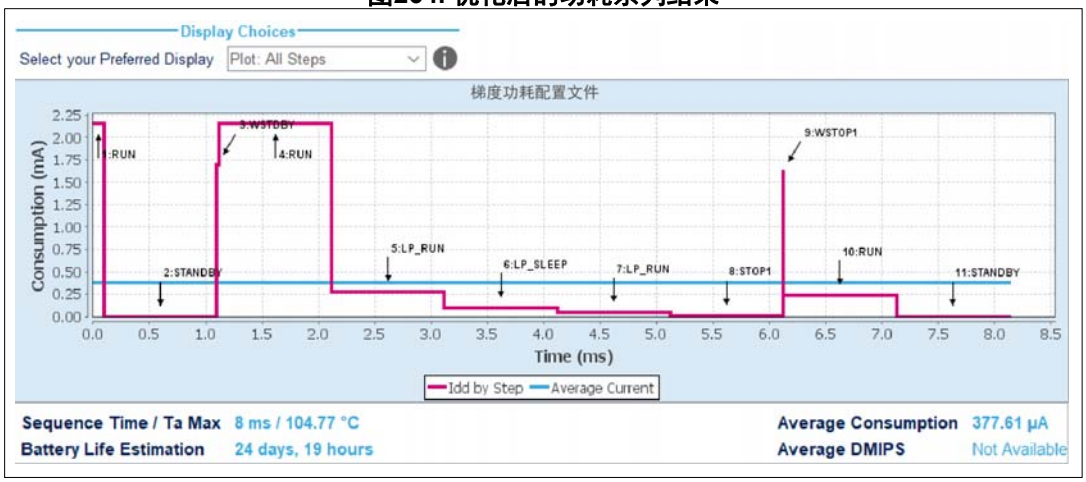
图263. 步骤10优化



有关序列整体结果，请参见*图 264*：持续时间为7 ms，电池使用寿命约为2个月，平均电流消耗为165.25 µA。

使用**比较**按钮将当前结果与另存为SequenceOne.pcs的原始结果进行比较。

图264. 优化后的功耗系列结果



14 教程4 - 通过串口与STM32L053xx Nucleo板通信示例 STM32L053xx Nucleo板的通信示例

本教程旨在演示如何使用STM32CubeMX为NUCLEO-L053R8板创建UART串行通信应用。

本例需要使用Windows PC。ST-Link USB连接器用于MCU上的串行数据通信、固件下载和调试。板与计算机之间必须连接Type-A转Mini-B USB连接线。USART2外设使用的PA2和PA3引脚连接到ST-Link连接器。此外，选择USART2通过ST-Link虚拟COM端口与PC进行通信。需要在PC上安装串行通信客户端（比如Tera Term），以显示通过虚拟通信端口从板子接收到的消息。

14.1 教程概述

教程4将介绍以下步骤：

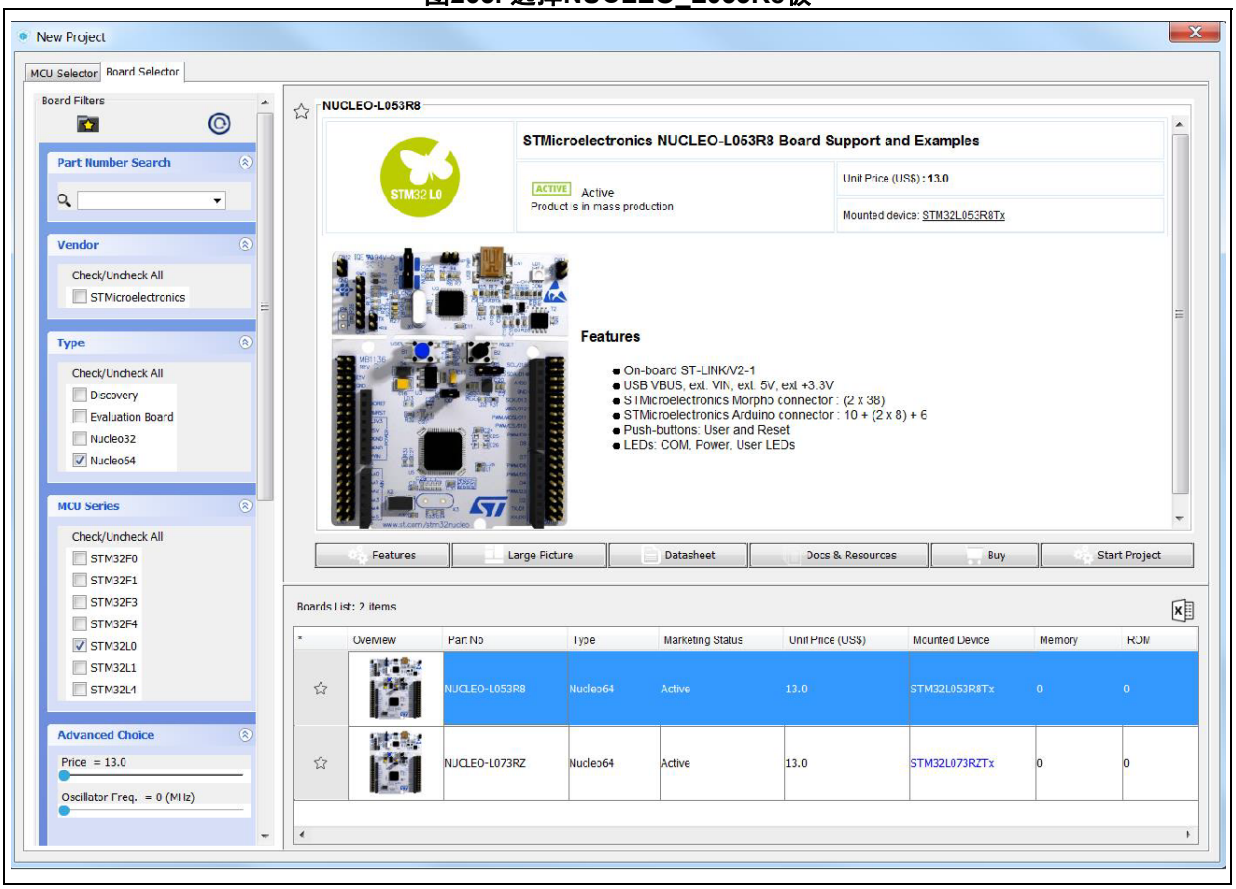
1. 从**新项目**菜单中选择NUCLEO-L053R8板。
2. 从**引脚排列**视图中选择所需功能（调试、USART、定时器）：外设工作模式以及引脚上的相关信号分配。
3. 在**时钟配置**视图中配置MCU时钟树。
4. 在**配置**视图中配置外设参数
5. 在**“项目管理器”**菜单中配置项目设置并生成项目（仅限初始化代码）。
6. 为项目更新对应于UART通信的用户应用代码示例。
7. 在板子上编译和执行项目。
8. 将Tera Term软件配置为PC上的串行通信客户端。
9. 结果显示在PC屏上。

14.2 创建一个新STM32CubeMX项目并选择Nucleo板

为此，需遵循以下步骤：

1. 从主菜单栏中选择**文件 > 新项目**。此操作会打开**新项目**窗口。
2. 进入**板选择器**选项卡并对STM32L0系列进行筛选。
3. 选择NUCLEO-L053R8，并点击**“确定”**在STM32CubeMX用户界面中加载板子（参见*图 265*）。

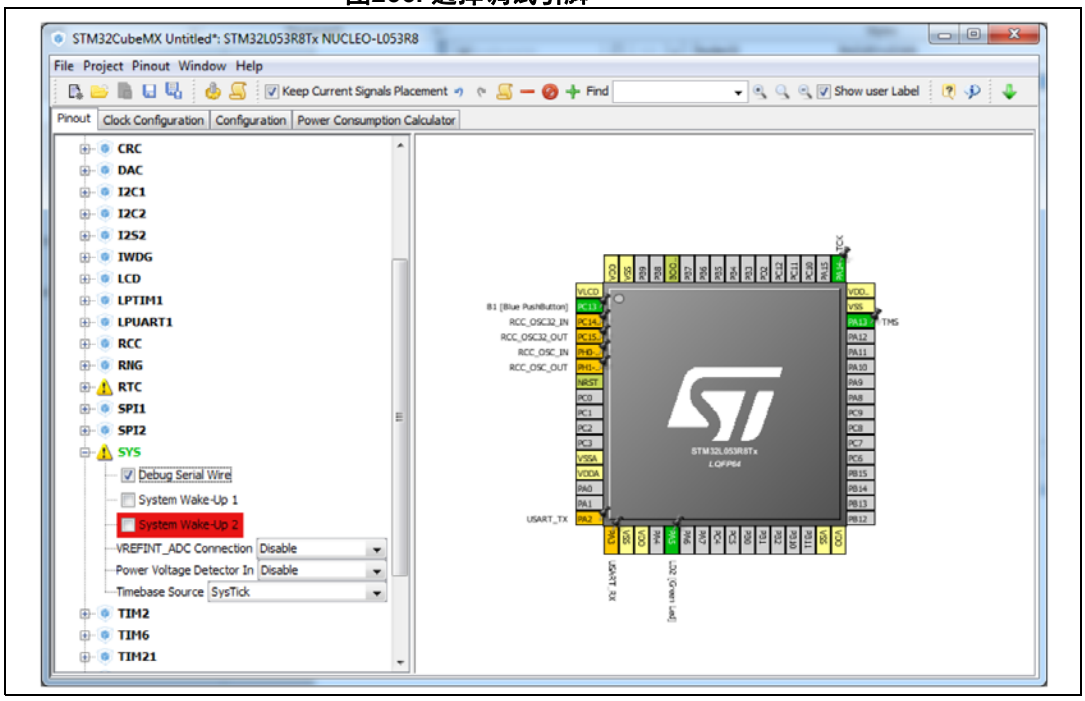
图265. 选择NUCLEO_L053R8板



14.3 从“引脚排列”视图中选择功能

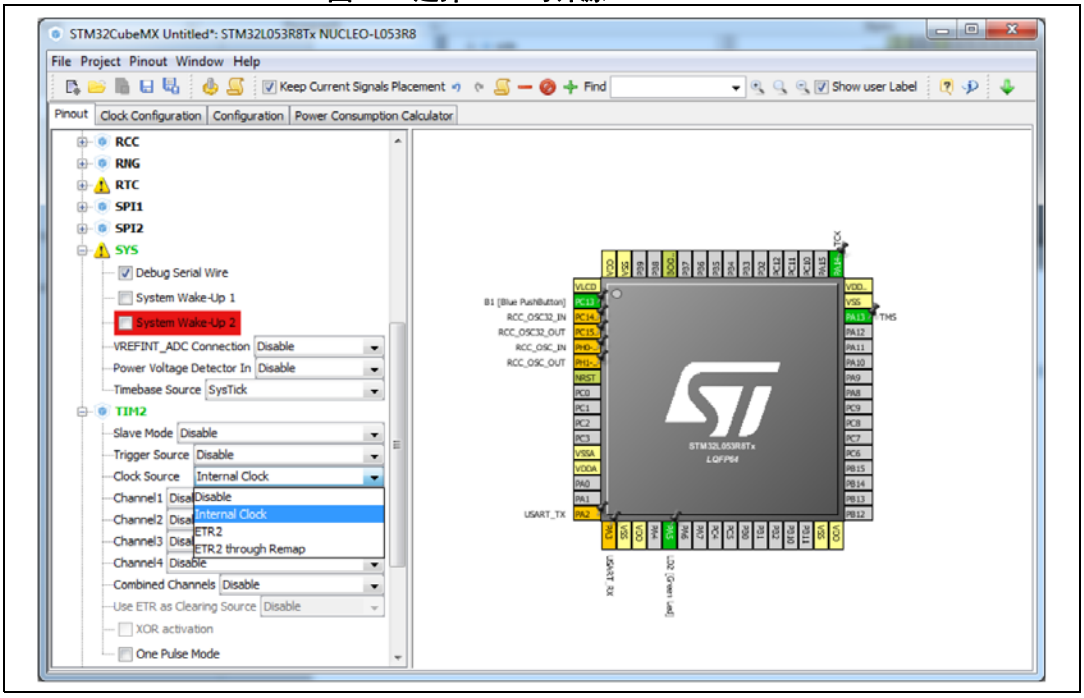
1. 在SYS下选择“调试串行线”（参见*图 266*）。

图266. 选择调试引脚



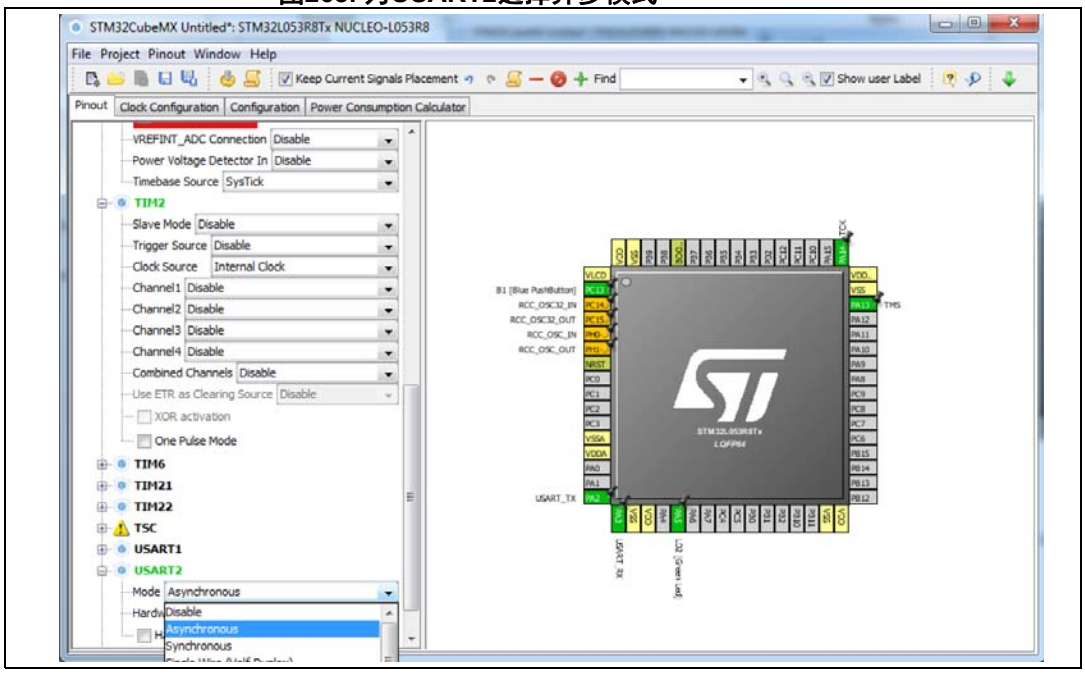
2. 在TIM2外设下选择“内部时钟”作为时钟源（参见*图 267*）。

图267. 选择TIM2时钟源



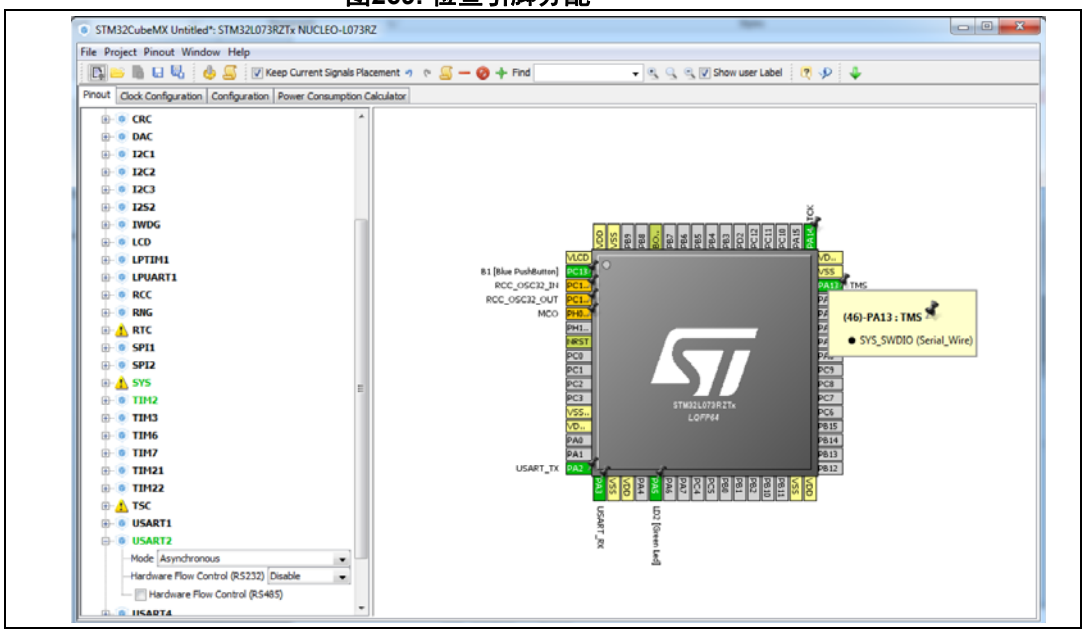
3. 为USART2外设选择异步模式（参见*图 268*）。

图268. 为USART2选择异步模式



4. 检查为各引脚分配的信号是否正确（参见*图 269*）：
   - 为PA13分配SYS_SWDIO
   - 为PA14分配TCK
   - 为PA2分配USART_TX
   - 为PA3分配USART_RX

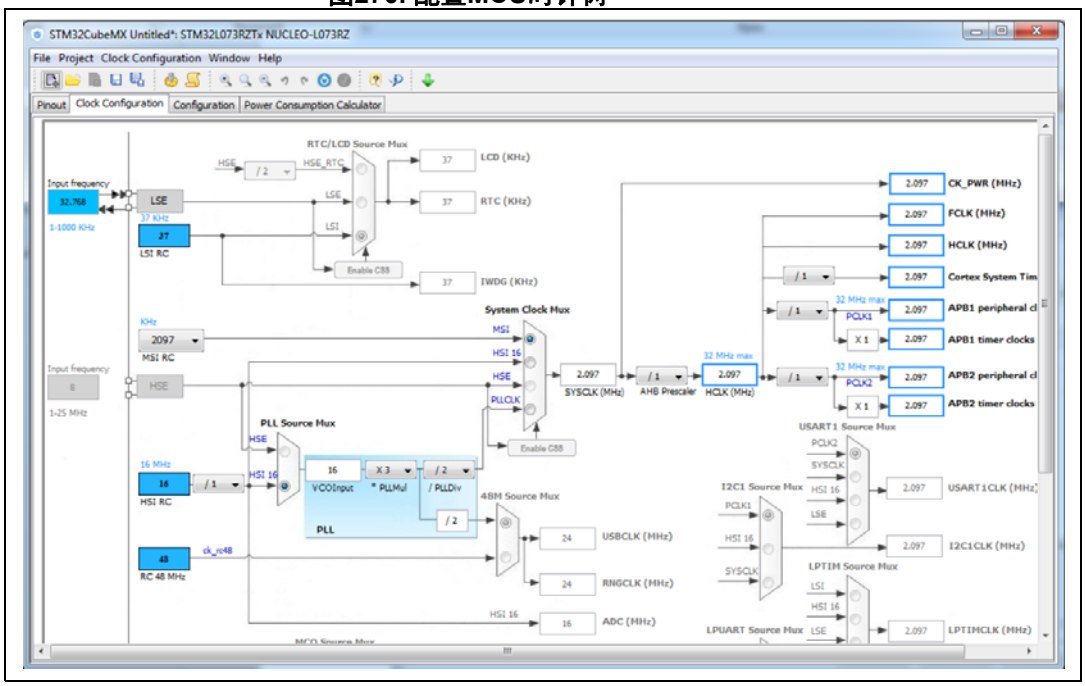
图269. 检查引脚分配



14.4 在“时钟配置”视图中配置MCU时钟树

1. 进入**时钟配置**选项卡并保留配置，以使用MSI作为输入时钟，并使用2.097 MHz的HCLK（参见*图 270*）。

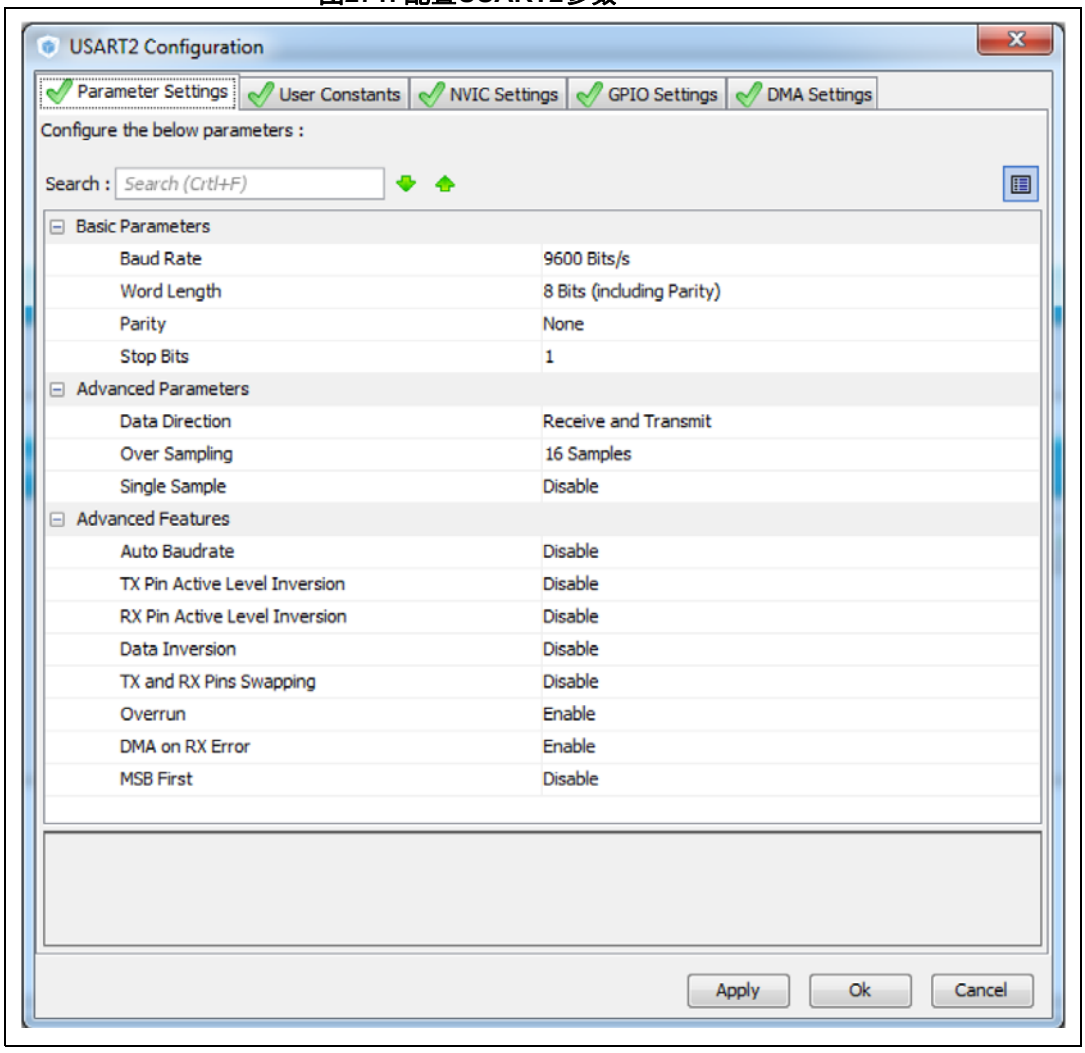
图270. 配置MCU时钟树



14.5 在“配置”视图中配置外设参数

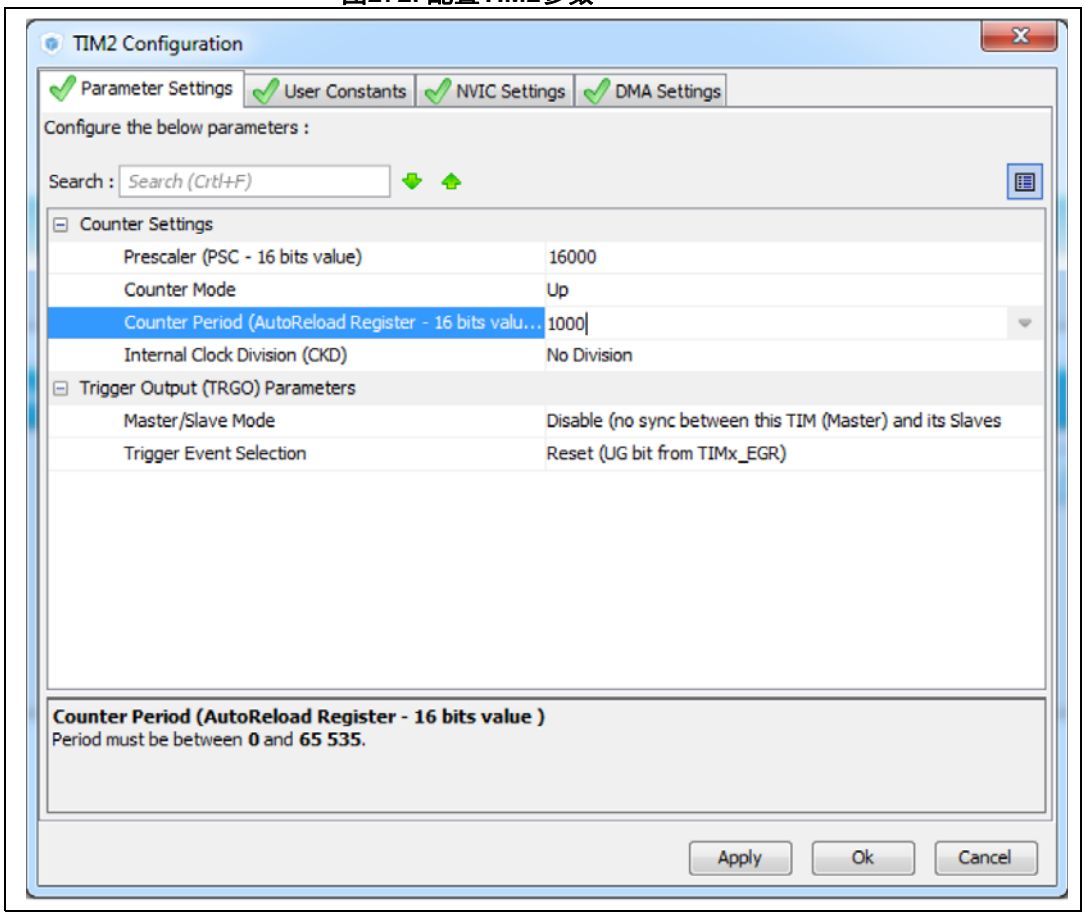
1. 在**配置**选项卡中，点击**USART2**打开外设**参数设置**窗口，并将波特率设为9600。确保“数据方向”设为“接收和发送”（参见*图 271*）。
2. 点击**“确定”**应用更改并关闭窗口。

图271. 配置USART2参数



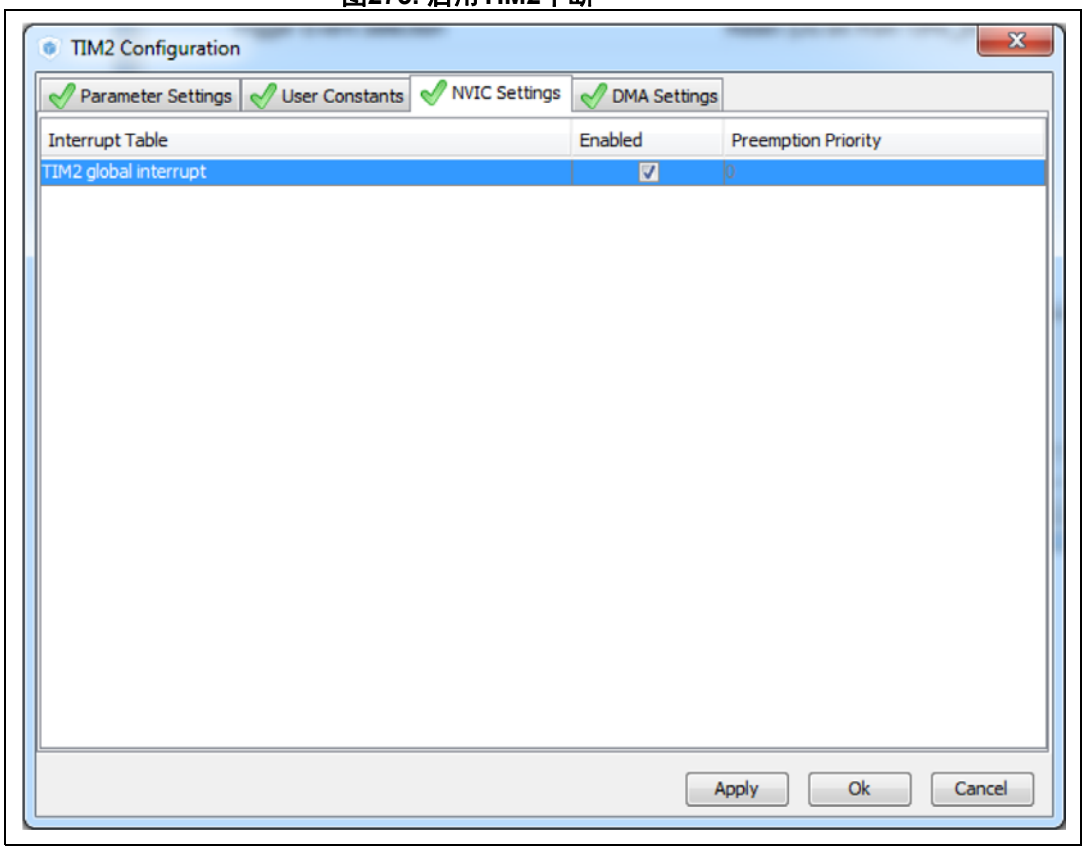
3. 点击**TIM2**并将预分频器改为16000，将“字长”改为8位，将“计数器周期”改为1000（参见*图 272*）。

图272. 配置TIM2参数



4. 在**NVIC设置**选项卡中启用TIM2全局中断（参见*图 273*）。

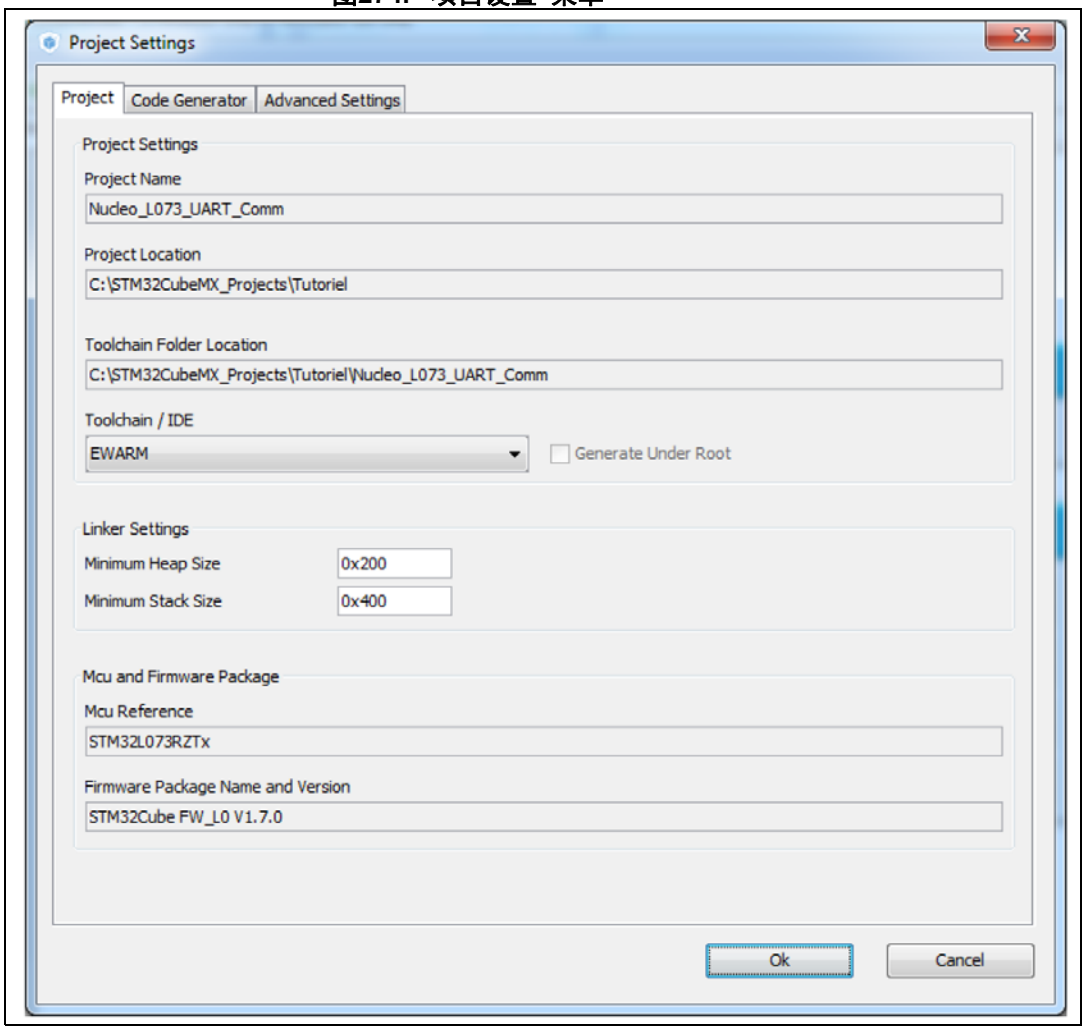
图273. 启用TIM2中断



14.6 配置项目设置并生成项目

1. 在**项目设置**菜单中，指定项目名称、目标文件夹，并选择EWARM IDE工具链（参见*图 274*）。

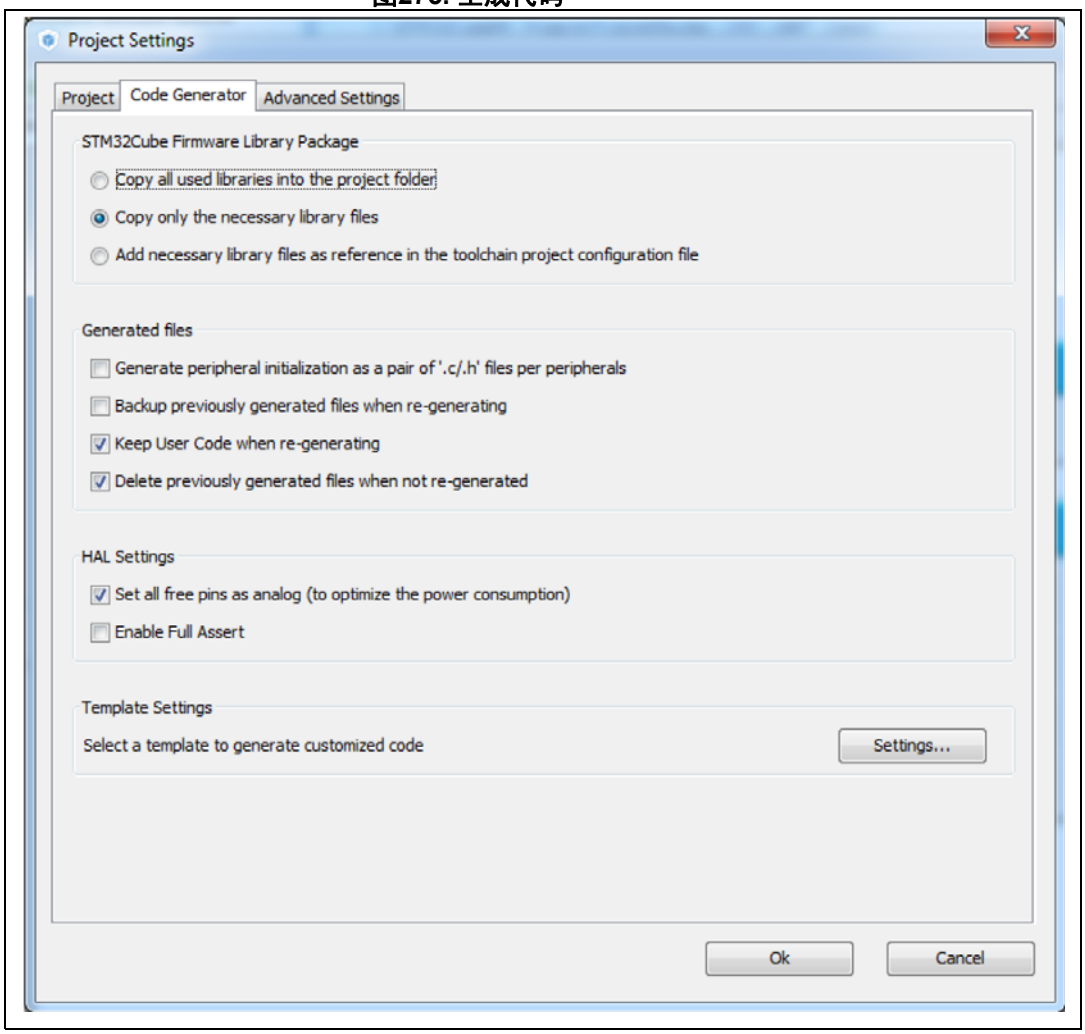
图274. “项目设置”菜单



如果固件包版本在用户PC中尚不可用，则会打开一个进度窗口，以显示固件包下载进度。

2. 在**代码生成器**选项卡中，按*图 275*所示配置要生成的代码，并点击**确定**以生成代码。

图275. 生成代码



14.7 使用用户应用代码更新项目

添加以下用户代码：

```
/* 用户代码开始 0 */
#include "stdio.h"
#include "string.h"
/* 用于传输的缓冲区和传输次数*/
char aTxBuffer[1024];
int nbtime=1;
/* 用户代码结束 0 */
```

在主函数中，启动定时器事件生成函数，具体如下：

```
/* 用户代码开始 2 */
 /* 启动定时器事件生成 */
```

```
  HAL_TIM_Base_Start_IT(&htim2);
 /* 用户代码结束 2 */

/* 用户代码开始 4 */
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim){
sprintf(aTxBuffer,"STM32CubeMX rocks %d times \t", ++nbtime);
HAL_UART_Transmit(&huart2,(uint8_t *) aTxBuffer, strlen(aTxBuffer), 5000);
}
/* 用户代码结束 4 */
```

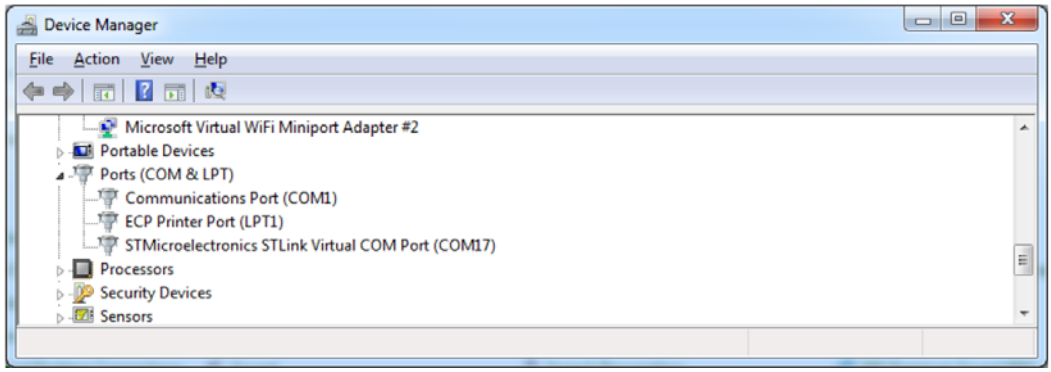
14.8 编译并运行项目

1. 在首选IDE中编译项目。
2. 将项目下载到板子中。
3. 运行程序。

14.9 将Tera Term软件配置为PC上的串行通信客户端 串行通信客户端

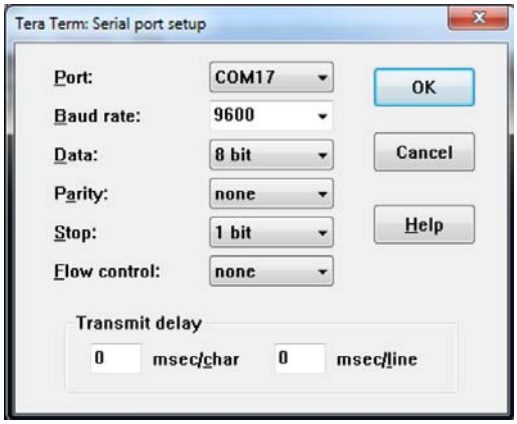
1. 在计算机上的“设备管理器”窗口中检查ST Microelectronics使用的虚拟通信端口（参见*图 276*）。

图276. 检查通信端口



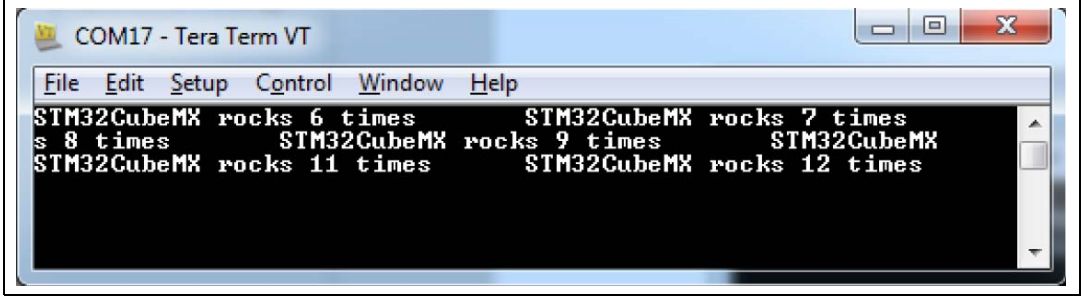
2. 要将Tera Term配置为侦听相关虚拟通信端口，请对参数进行调整，以使其与MCU上的USART2参数设置相匹配（参见*图 277*）。

图277. 设置Tera Term端口参数



3. Tera Term窗口会在几秒的周期内显示板子发出的消息（参见*图 278*）。

图278. 设置Tera Term端口参数



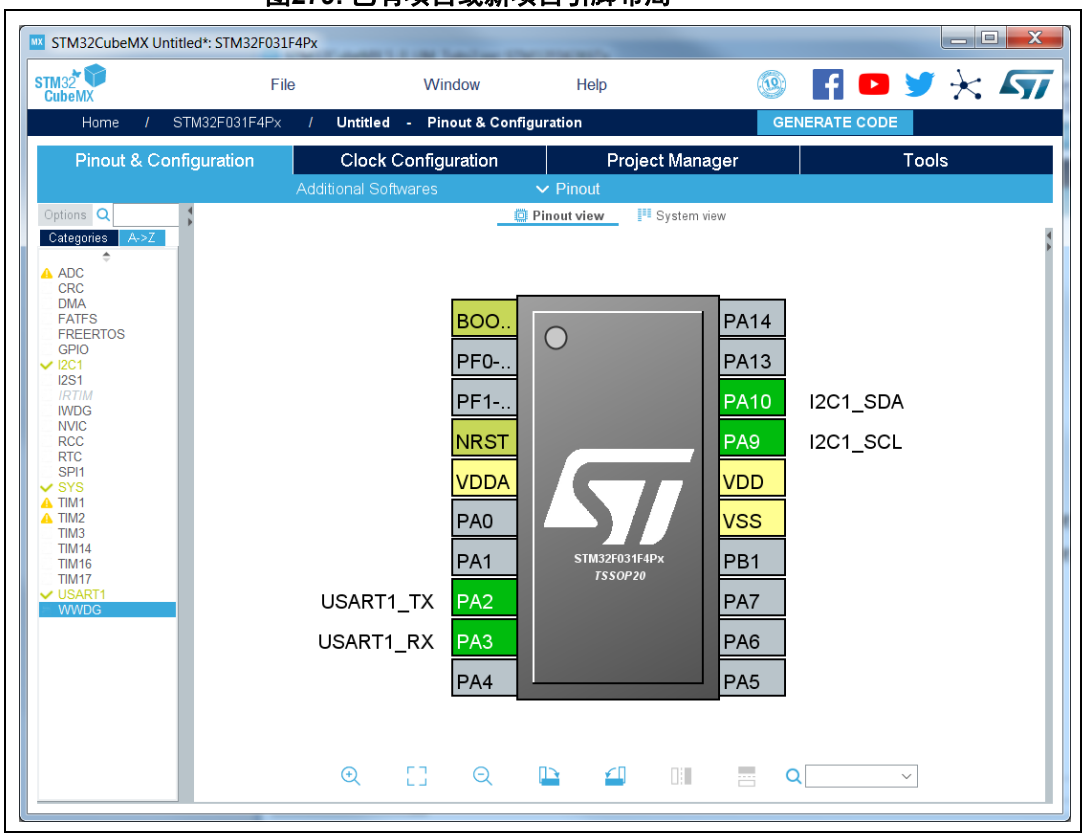
15 教程5：将当前项目配置导出到兼容MCU兼容MCU

如果在**引脚排列**菜单中选择**列出与引脚排列兼容的MCU**，则STM32CubeMX会获取与当前项目配置兼容的MCU列表，并提供选项将当前配置导出到新选择的兼容MCU。

本教程介绍了如何显示兼容MCU列表以及如何将当前项目配置导出到兼容MCU：

1. 加载已有项目，或创建并保存新项目：

图279. 已有项目或新项目引脚布局



2. 进入**引脚排列**菜单并选择**列出与引脚排列兼容的MCU**。弹出**引脚排列兼容**窗口（参见*图 280*和*图 281*）。

   可根据需要修改搜索条件和筛选选项，并点击**搜索**按钮重新开始搜索过程。

   颜色阴影和*注释*列表示匹配度：

   – 完全匹配：MCU与当前项目完全匹配（相关示例，请参见*图 281*）。
   – 部分匹配且硬件兼容：可确保硬件兼容性，但一些引脚名称不能保留。将鼠标悬停在所需MCU上即可显示解释性工具提示（相关示例，请参见*图 280*）。
   – 部分匹配但硬件不兼容：并非所有信号均可分配给完全相同的引脚位置，需要进行重新映射。将鼠标悬停在所需MCU上即可显示解释性工具提示（相关示例，请参见*图 281*）。

图280. 与引脚排列兼容的MCU列表 - 部分匹配且硬件兼容

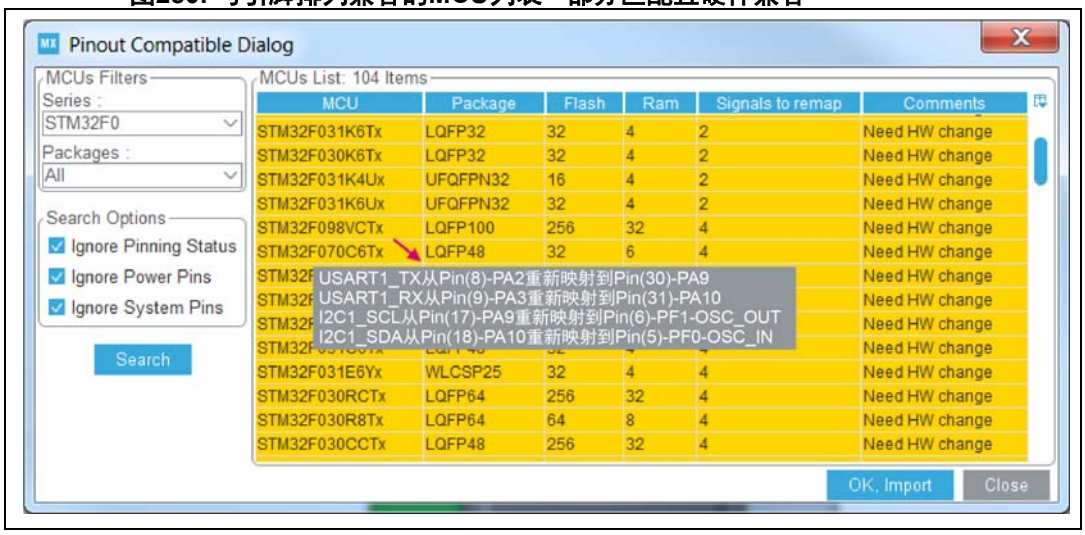
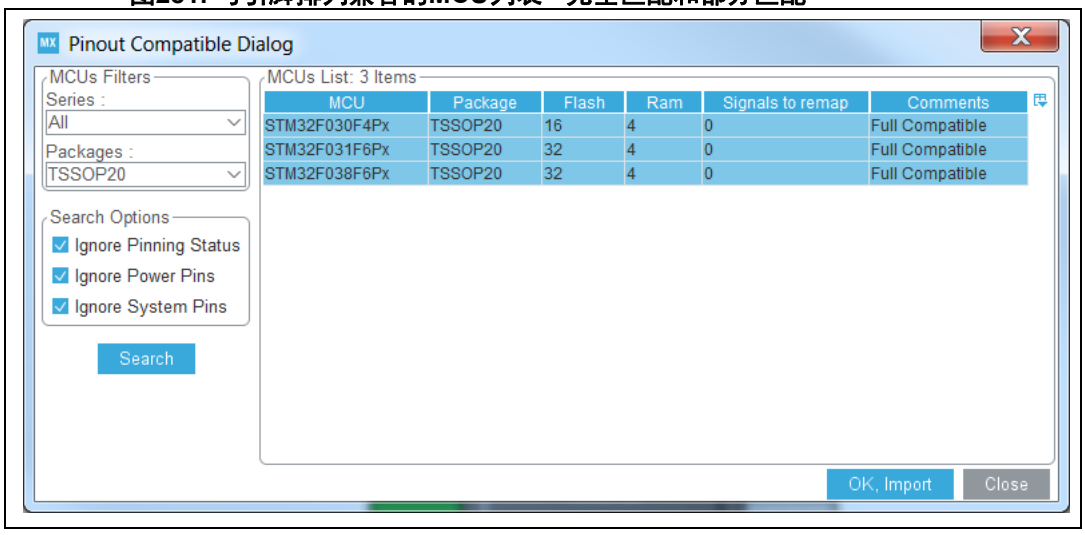
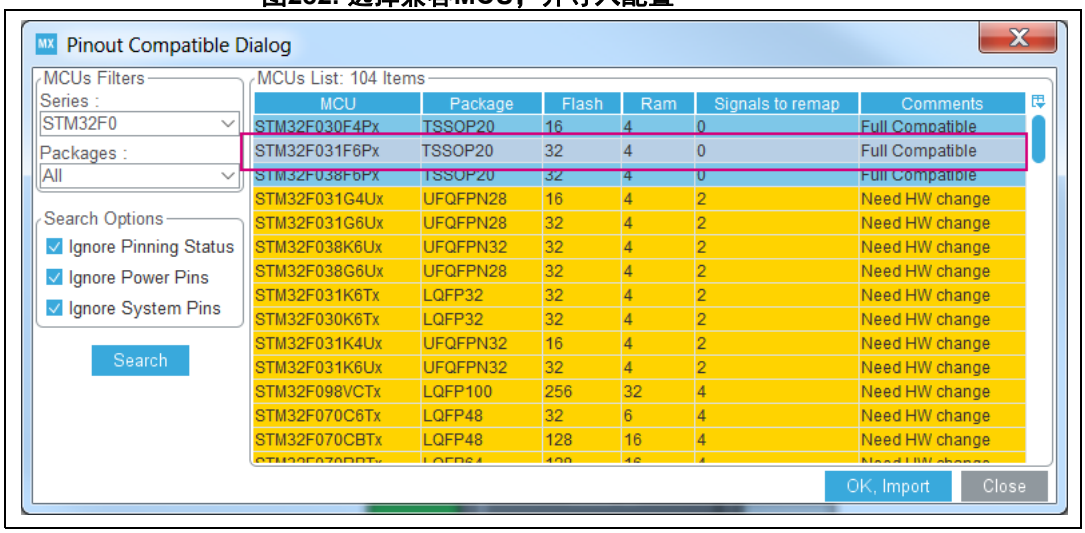


图281. 与引脚排列兼容的MCU列表 - 完全匹配和部分匹配



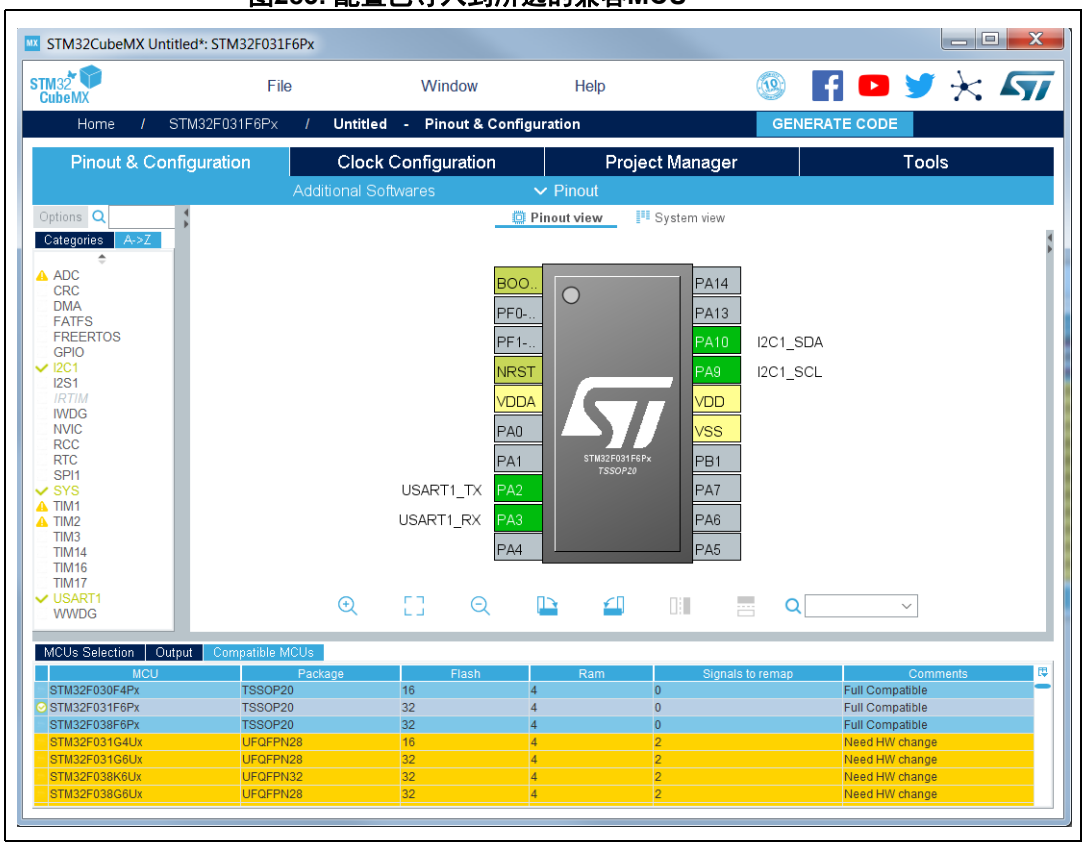
3. 然后选择要导入当前配置的MCU，并点击**确定，导入**：

图282. 选择兼容MCU，并导入配置



配置现在可用于所选MCU：

图283. 配置已导入到所选的兼容MCU



4. 为了能够随时查看兼容MCU列表，请在**窗口**菜单下选择**输出**。
   要将当前配置加载到另一兼容MCU，请双击兼容MCU列表。
5. 要删除对搜索条件的一些限制，可采用多种方法：
   – 选中**忽略引脚状态**复选框可忽略引脚状态（已锁定引脚）。
   – 选中**忽略电源引脚**复选框则不会考虑电源引脚。
   – 选中**忽略系统引脚**复选框则不会考虑系统引脚。将鼠标悬停在复选框上可显示工具提示，其中会列出当前MCU上可用的系统引脚。

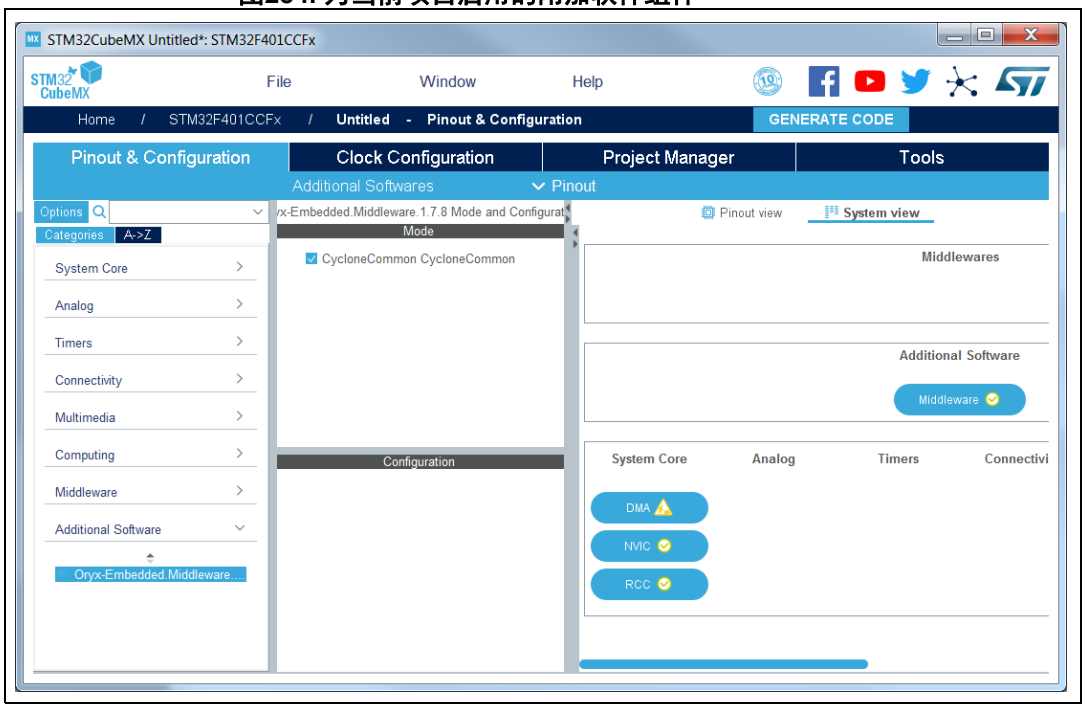
16 教程6 – 将嵌入式软件包添加到用户项目

在该教程中，以Oryx-Embedded.Middleware.1.7.8.pack为例演示了如何将软件包组件添加到STM32CubeMX项目。但不应理解为意法半导体建议使用此软件包。

要将嵌入式软件包添加到项目中，请按以下步骤操作：

1. 使用http://www.oryx-embedded.com中提供的.pdsc文件安装Oryx-Embedded.Middleware.1.7.8.pack（参见*第 3.4.4节：安装嵌入式软件包*）。
2. 选择**新项目**。
3. 从**MCU选择器**中选择STM32F01CCFx。
4. 从**“引脚布局和配置”**视图中选择**“其他软件”**，以打开“其他软件组件”窗口，然后选择以下软件组件：CycloneCommon套件中的“编译器支持”，“RTOS端口/无”和“日期时间帮助程序例程”（请参阅*第 4.13节：附加软件组件选择窗口*）。
5. 在树形视图中点击**确定**可显示所选组件，点击复选框可为当前项目启用该软件组件（参见*图 284*）。

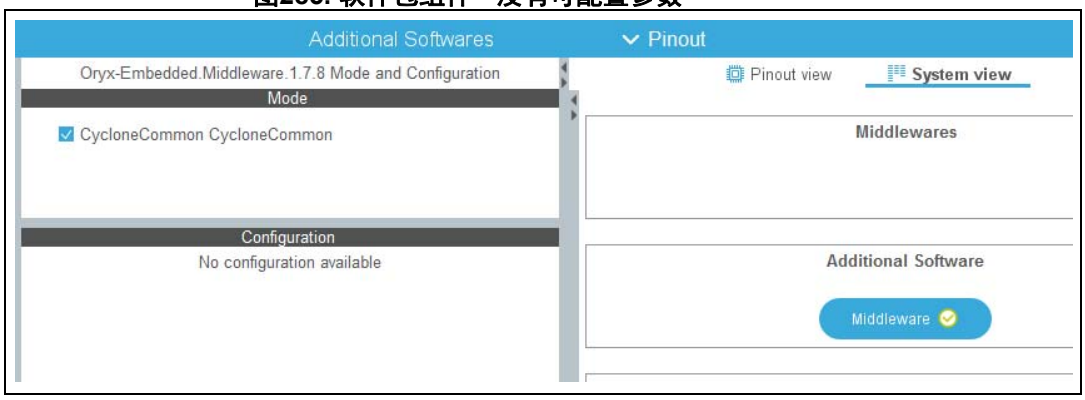
图284. 为当前项目启用的附加软件组件



如果软件包名称以绿色突出显示，说明所选软件组件的所有条件均已满足。如果至少有一个条件未满足，则软件包名称会以橙色突出显示。

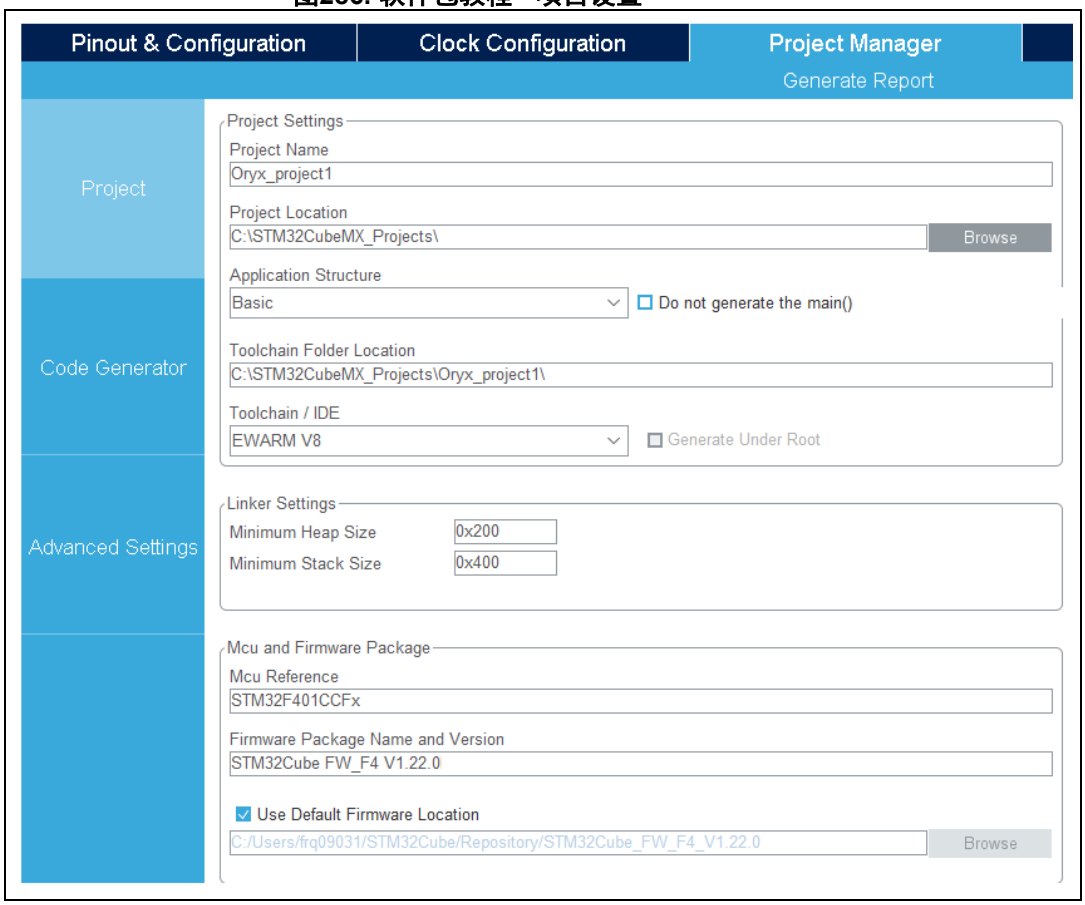
6. 检查是否没有参数可在**配置**选项卡中进行配置（参见*图 285*）。

图285. 软件包组件 - 没有可配置参数



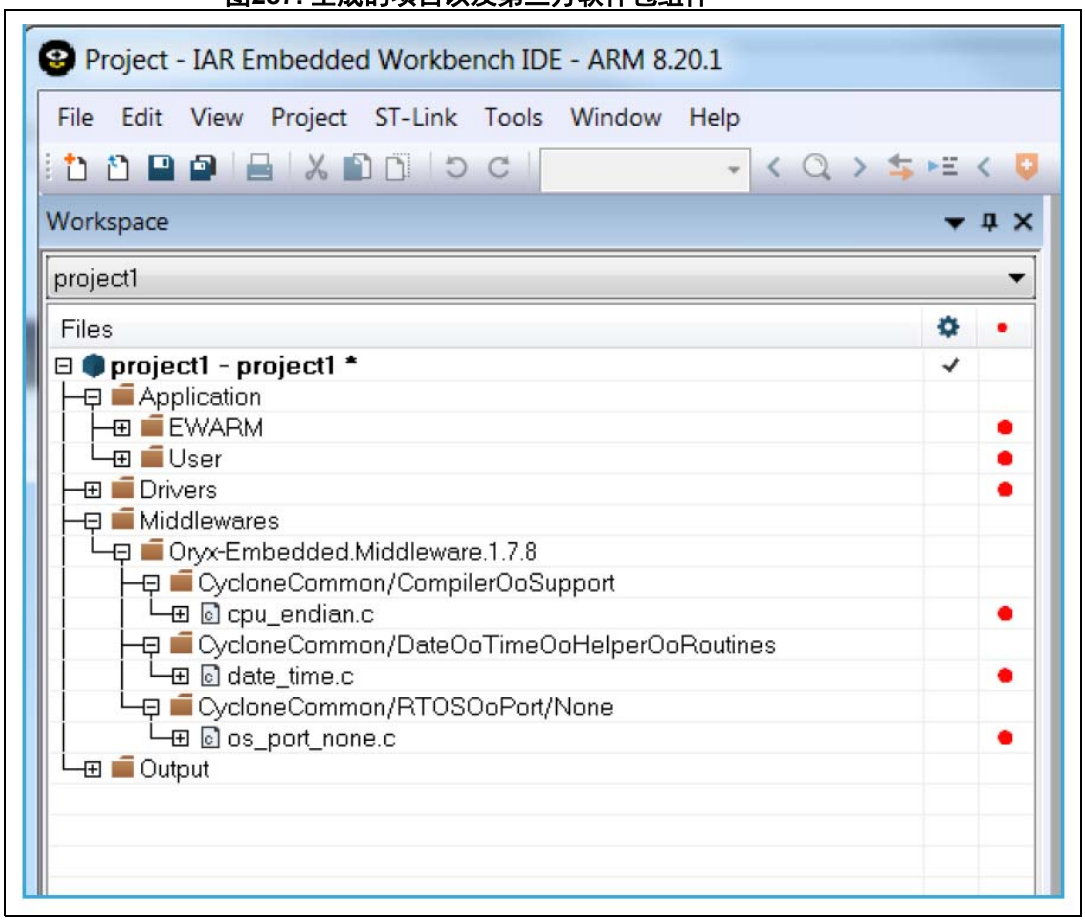
7. 选择**项目管理器**项目选项卡，指定项目参数（参见*图 286*），并选择IAR™EWARM作为IDE。

图286. 软件包教程 - 项目设置



8. 点击 GENERATE CODE 生成项目。如果点击接受，则会在STM32Cube中不存在STM32CubeF4 MCU包的情况下下载此软件包。
9. 点击**打开项目**。Oryx软件组件会显示在生成的项目中（参见*图 287*）。

图287. 生成的项目以及第三方软件包组件



17 教程7–使用X-Cube-BLE1软件包

本教程演示了如何使用X-Cube-BLE1软件包来实现功能项目。

下面是运行本教程的必要条件：

- 硬件：NUCLEO-L053R8、X-NUCLEO-IDB05A1和迷你USB电缆（请参阅*图 288*）
- 工具：STM32CubeMX，IDE（Atollic®或STM32CubeMX所支持的任何其他工具链）
- 嵌入式软件包：STM32CubeL0（版本1.10.0或更高版本），X-Cube-BLE1 1.1.0（请参阅*图 289*）。
- 移动应用程序（请参阅*图 290*：适用于iOS®或Android™的意法半导体BlueNRG应用程序

图288. 必要的硬件条件

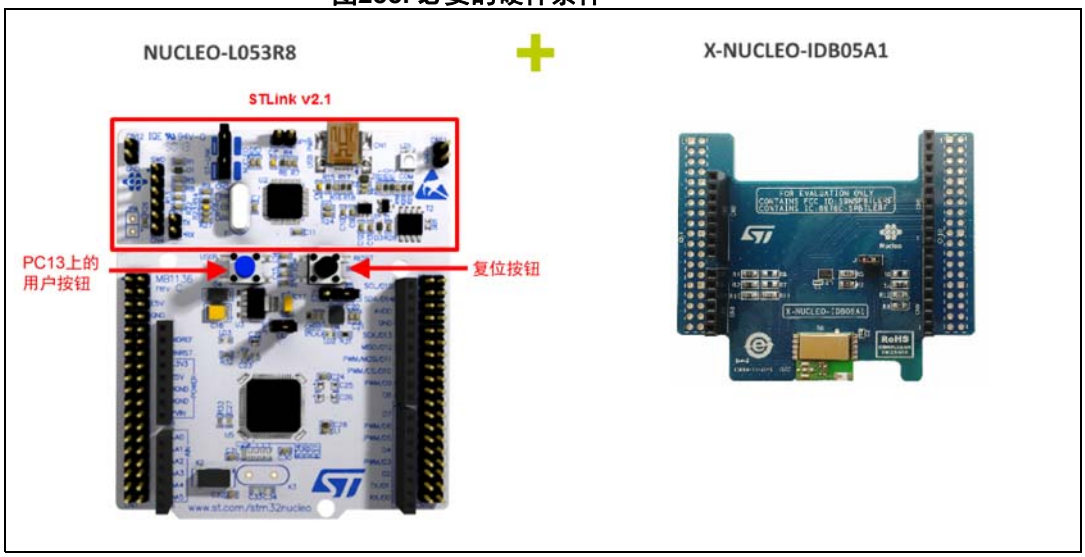


图289. 嵌入式软件包

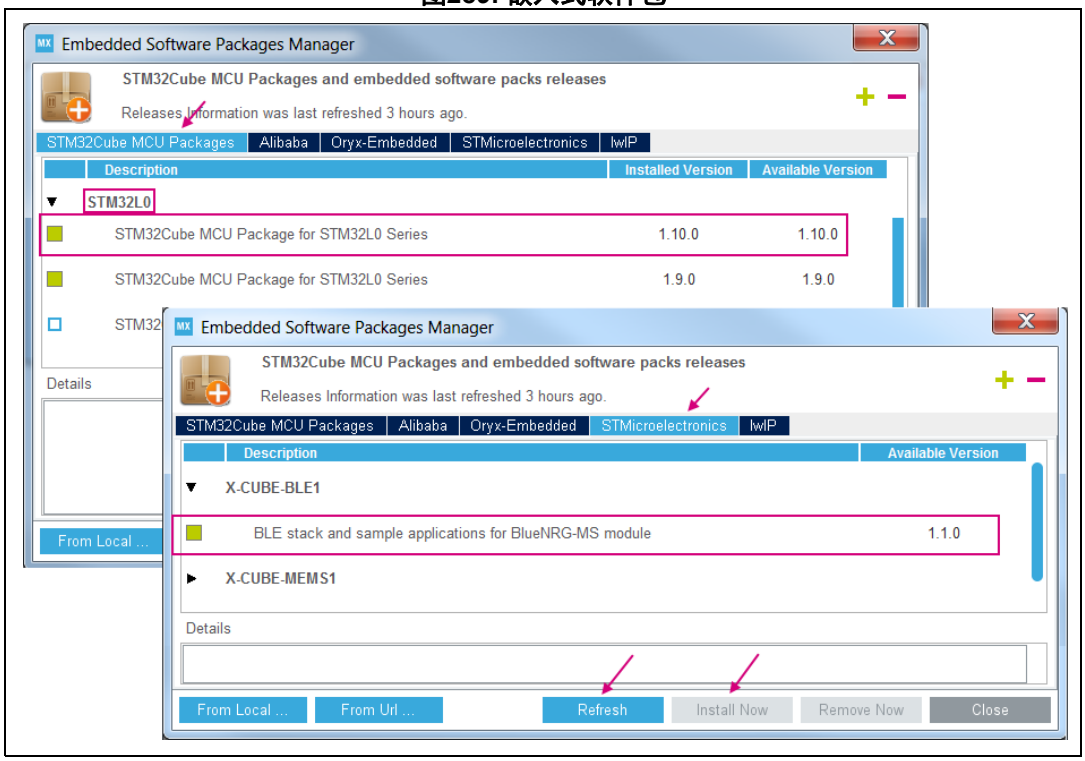
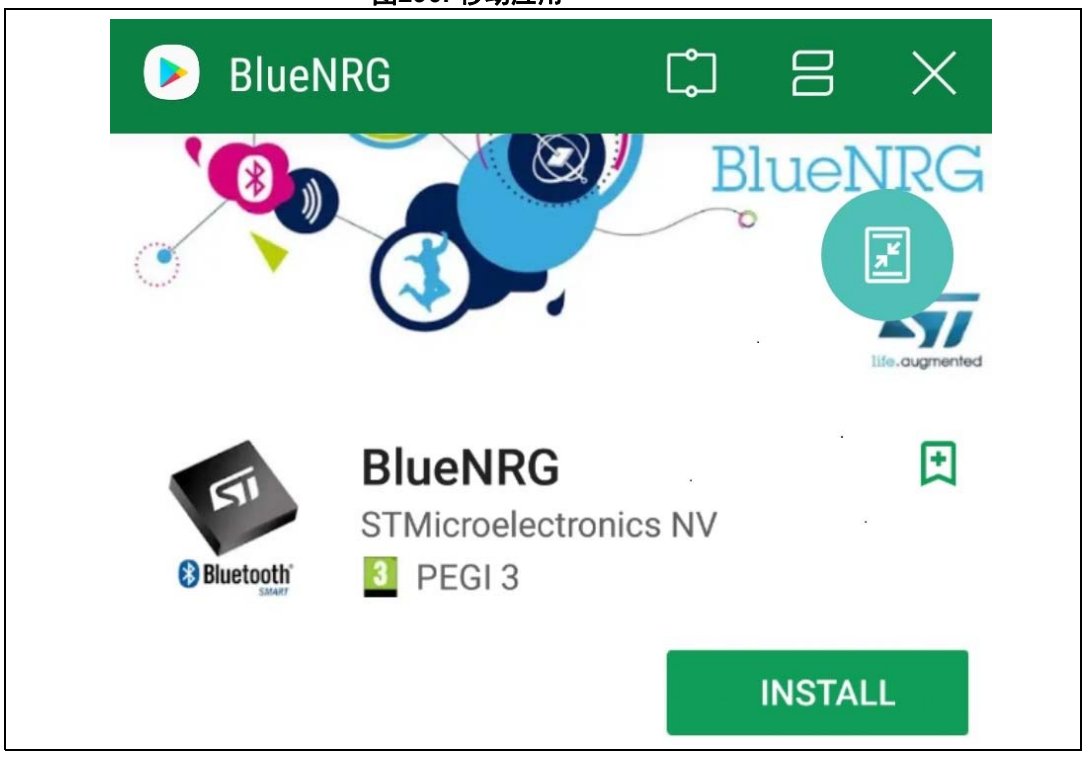


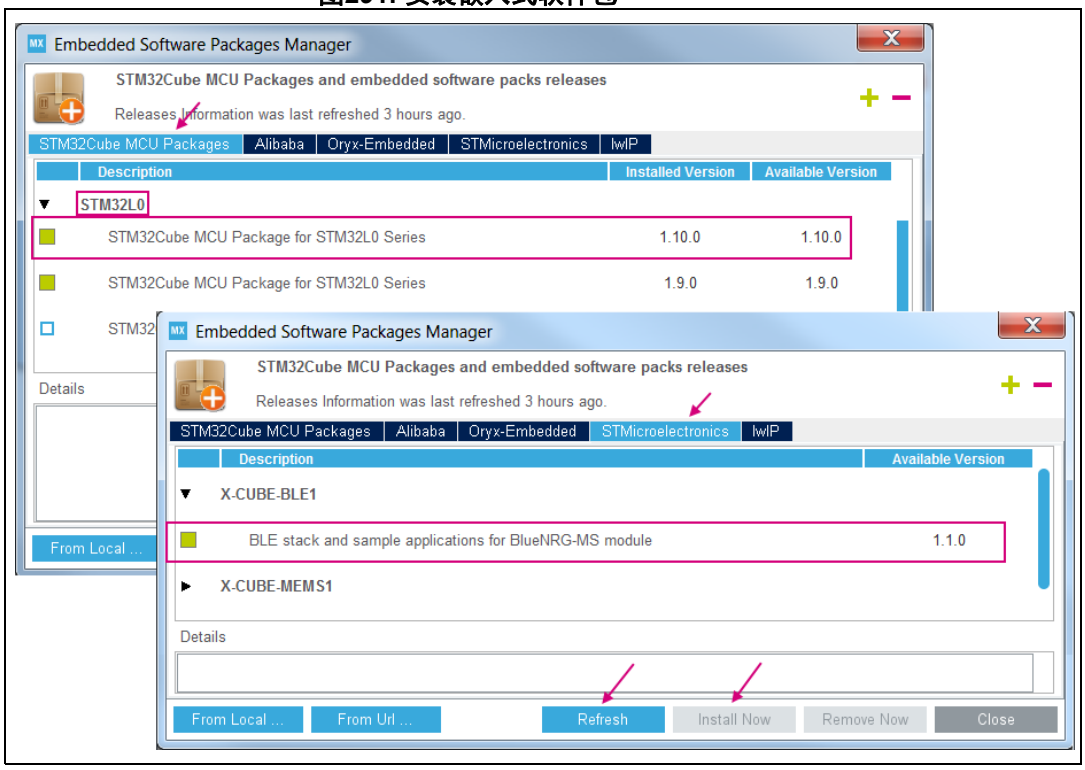
图290. 移动应用



按照以下步骤安装和运行本教程：

1. 检查STM32CubeMX互联网连接：
   a) 选择**“帮助”>“更新程序设置”**菜单，打开更新程序窗口。
   b) 在**“连接”**选项卡中验证是否已配置并已建立互联网连接。
2. 安装所需的嵌入式软件包（请参阅*图 291*）：
   a) 选择**“帮助”>“管理嵌入式软件包”**菜单，打开**“嵌入式软件包管理器”**窗口。
   b) 单击**“刷新”**按键，以使用最新的可用软件包版本刷新列表。
   c) 选择**STM32Cube MCU软件包**选项卡，并检查是否已安装STM32CubeL0固件软件包1.10.0或更高版本（复选框必须为绿色）。否则，请选中复选框并单击**“立即安装”**。
   d) 选择**“意法半导体”**选项卡，并检查是否已安装X-Cube-BLE1软件包版本1.0.0（复选框必须为绿色）。否则，请选中复选框并单击**“立即安装”**。

图291. 安装嵌入式软件包



3. 开始一个新项目：
   a) 选择**“新建项目”**以打开新项目窗口。
   b) 选择**“板选择器”**选项卡。
   c) 板类型选择Nucleo64，MCU系列选择STM32L0。
   d) 从结果板列表中选择NUCLEO-L053R8（请参阅*图 292*）。
   e) 当提示以默认模式初始化所有外设时，请回答**“否”**（请参阅*图 293*）。

图292. 开始一个新项目-选择NUCLEO-L053R8板

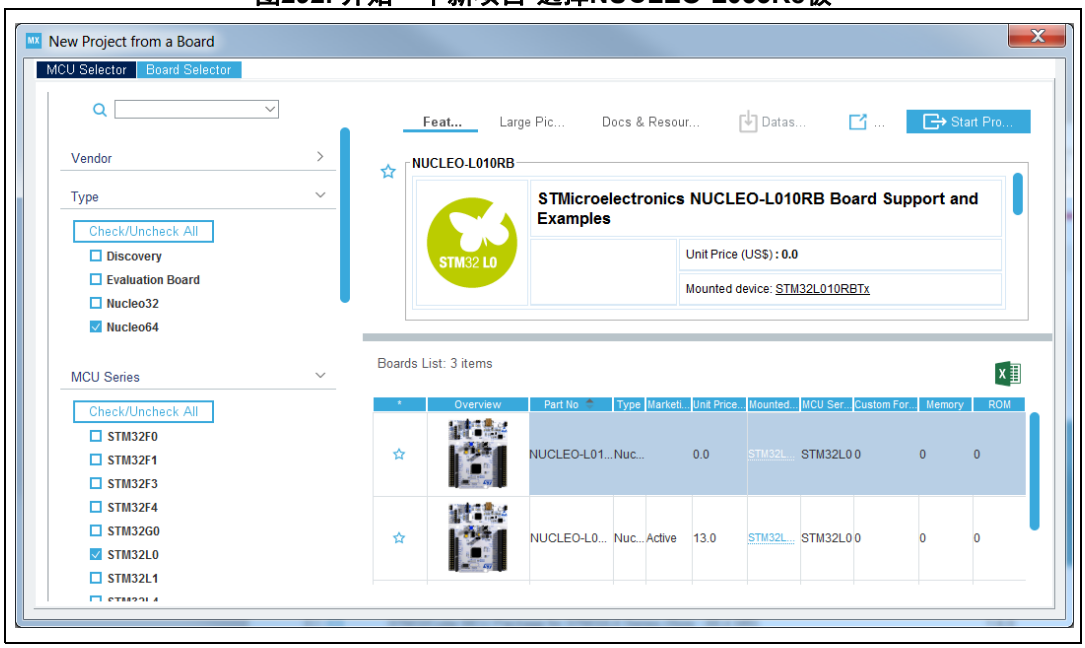
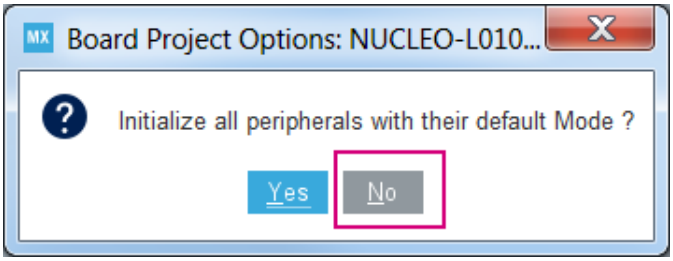


图293. 开始一个新项目-初始化所有外设



4. 将X-Cube-BLE1组件添加到项目中：

   a) 从**“引脚布局和配置”**视图中单击**“其他软件”**，以打开**“其他软件组件选择”**窗口。

   b) 选择相关组件（请参阅*图 294*）

   “应用程序组”随附应用程序列表：C文件实现应用程序循环，即*Process()*函数。从“应用程序组”中，选择**“SensorDemo”**应用程序。

   选择**控制器**和**实用程序组件**

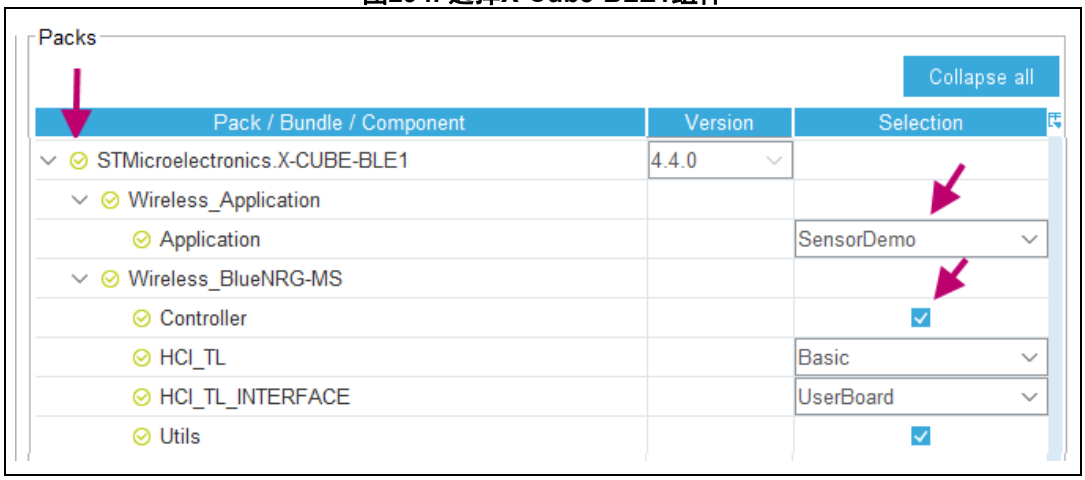
   为**HCI_TL**组件选**择“基本版本”。**“基本版本”提供了意法半导体实现的HCI_TL API，而“模板”选项则要求用户实施自己的代码。

   选择**UserBoard**变体作为**HCI_TL_INTERFACE**组件。 使用UserBoard选项会生成<boardname>_bus.c文件，即本教程的nucleo_l053r8_bus.c文件；而“模板”选项会生成custom_bus.c文件并要求用户提供自己的实施方式。

   有关软件组件的更多详细信息，请参阅X-Cube-BLE1软件包文档。

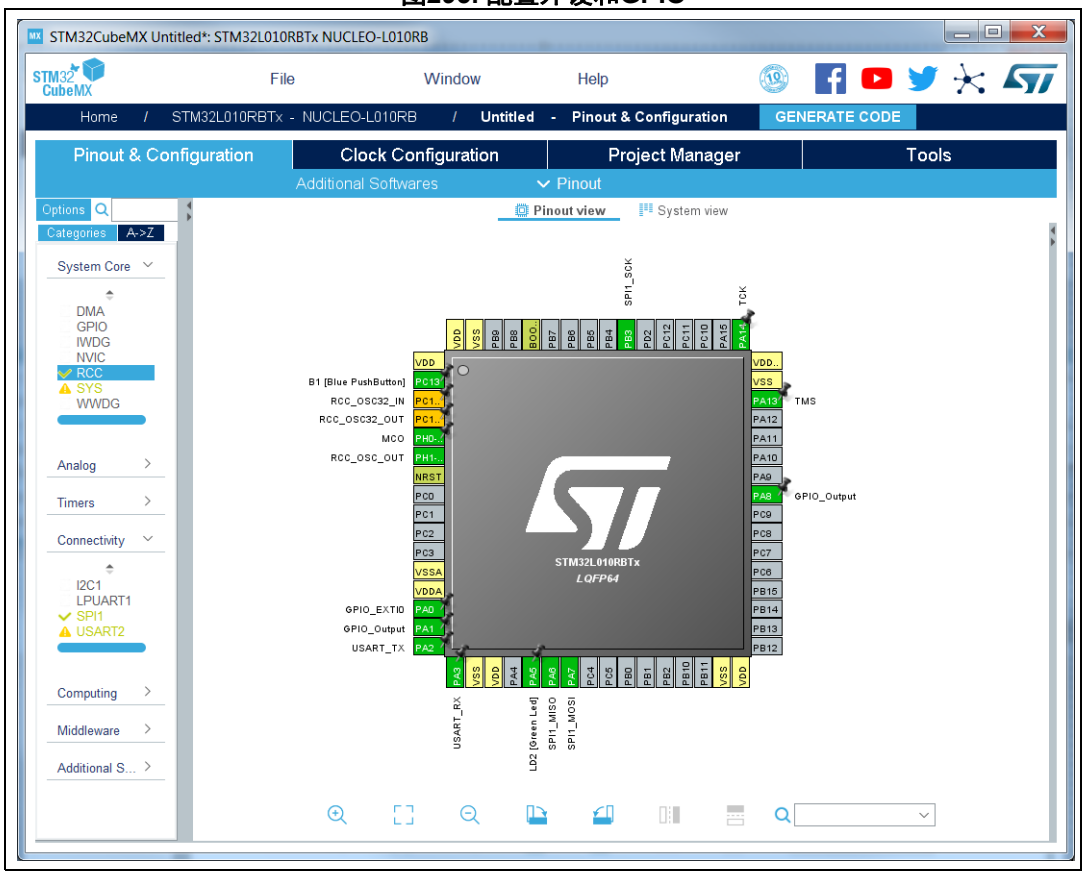
c) 单击**“确定”**将选择应用到项目并关闭窗口。左侧面板的**“其他软件”**部分将进行相应的更新。

图294. 选择X-Cube-BLE1组件



5. 从**“引脚布局”**选项卡启用外设和GPIO（请参阅*图 295*）：
   a) 在**“异步模式”**下配置**USART2**。
   b) 在**“全双工主模式”**下配置**SPI1**。
   c) 左键单击以下引脚，然后针对所需的GPIO对其进行设置：
      PA0：GPIO_EXTI0
      PA1：GPIO_Output
      PA8：GPIO_Output
   d) 在**SYS**外设下启用**“调试串行线”**。

图295. 配置外设和GPIO



6. 从“配置”选项卡**配置**外设：

   a) 单击**“系统”**部分下的**NVIC**按键以打开**NVIC配置**窗口。启用EXTI第0行和第1行中断，然后单击**OK**（请参阅*图 296*）。

   b) 单击**“连接性”**部分下的**SPI**按键以打开**SPI配置**窗口。检查数据大小是否设置为8位并且预分频器的值是否设置为16，以使HCLK除以预分频器的值小于或等于8 MHz。

   c) 单击**“连接性”**部分下的**USART2**以打开**“配置”**窗口，并检查以下参数设置：

   在**“参数设置”**下：

   波特率：115200比特/秒

   字长：8位（包括奇偶校验）

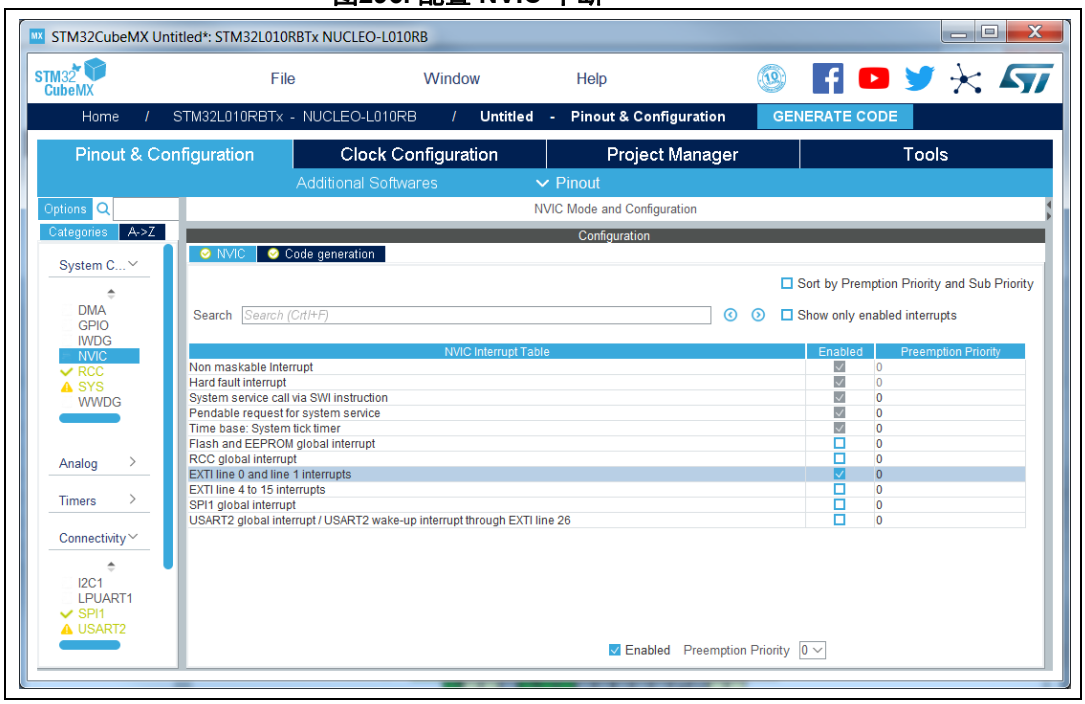
   校验位：无

   停止位：1

   在**“GPIO设置”**下：

   用户标签：USART_TX和USART_RX

图296. 配置 NVIC 中断



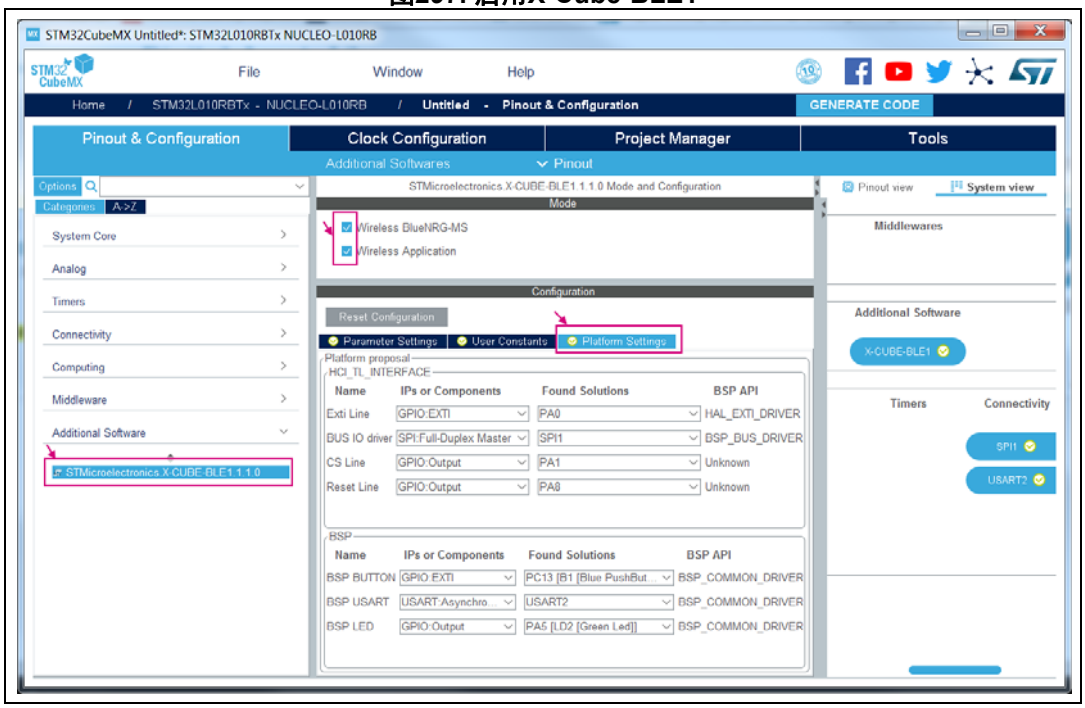
7. 从**“引脚布局和配置”**视图启用和配置X-Cube-BLE1软件包组件：
   a) 单击左侧面板中的软件包项目以显示模式和配置选项卡。
   b) 单击“模式”面板中的复选框以启用X-Cube-BLE1，将显示“配置”面板，其中显示了要配置的参数。橙色三角形表示某些参数未配置。正确配置所有参数后，橙色三角形会变成绿色的复选标记（请参阅*图 297*）。
   c) “参数设置”选项卡保持不变。
   d) 转到“平台设置”选项卡，按照*图 297*和*表 23*中所述，使用硬件资源配置连接。

表23. 连接硬件资源

名称	IP或组件	找到的解决方案
BUS IO驱动器	全双工主模式下的SPI	SPI1
EXTI线	GPIO：EXTI	PA0
CS系列	GPIO：输出	PA1
重置行	GPIO：输出	PA8
BSP LED	GPIO：输出	PA5
BSP按钮	GPIO：EXTI	PC13
BSP USART	异步模式下的USART	USART2

检查图标是否变为  。点击**“确定”**关闭**“配置”**窗口。

图297. 启用X-Cube-BLE1



8. 生成SensorDemo项目：

   a) 点击 GENERATE CODE 生成代码。如果项目尚未保存，则打开**“项目设置”**窗口。

   b) 正确配置了项目设置后，单击 GENERATE CODE 以生成代码（请参阅*图 298*）。生成完成后，将出现一个对话框窗口，要求打开项目文件夹（“打开文件夹”）或打开IDE工具链中的项目（“打开项目”）。选择**“打开项目”**（请参阅*图 299*）。

   c) 如果.cproject文件与Atollic® TrueStudio®相关联，通过单击**“打开项目”**可自动启动TrueStudio®：在TrueStudio启动窗口中，创建或选择现有工作区（请参阅*图 300*），然后单击**“确定”**。STM32CubeMX生成的项目将显示在TrueStudio® **“项目资源管理器”**面板中（请参阅*图 301*）。

图298. 配置SensorDemo项目

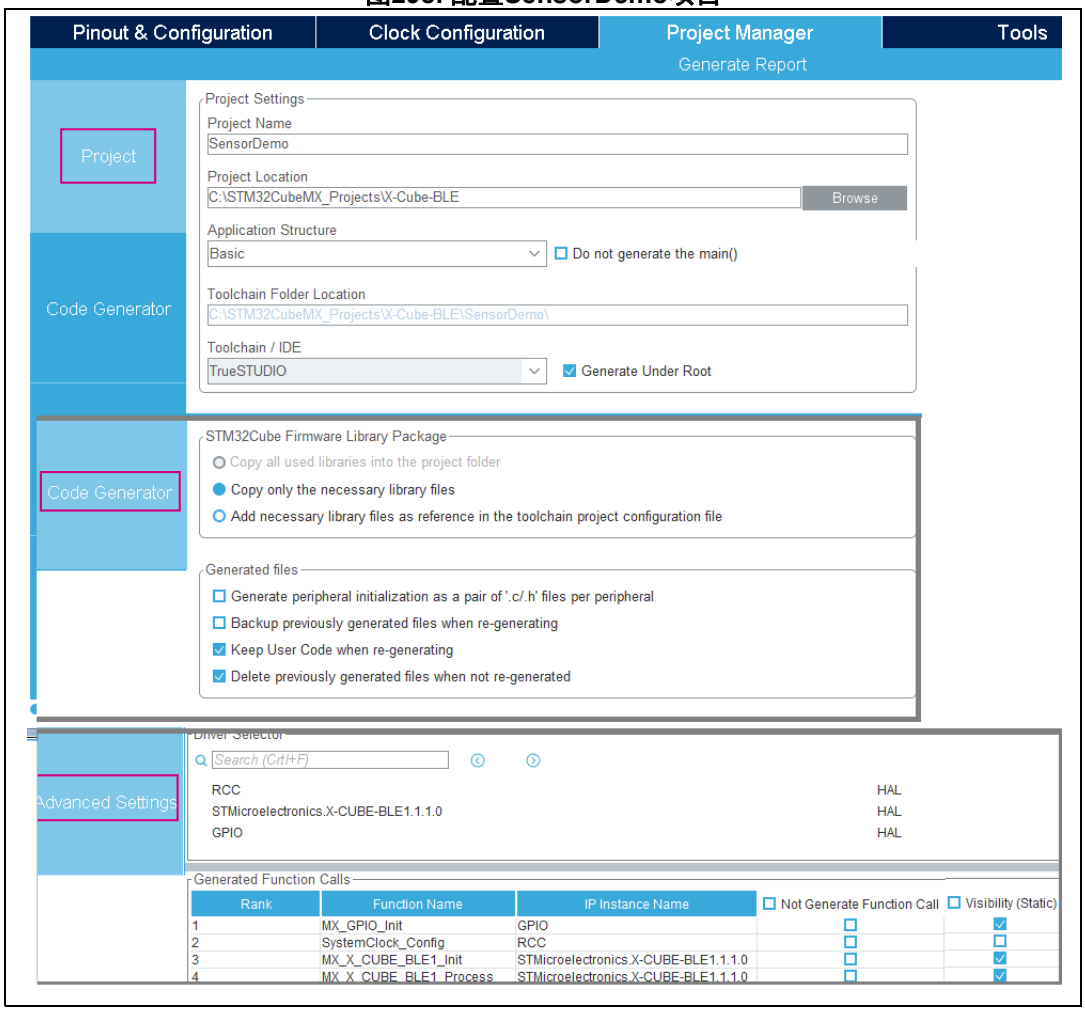


图299. 打开IDE工具链中的SensorDemo项目

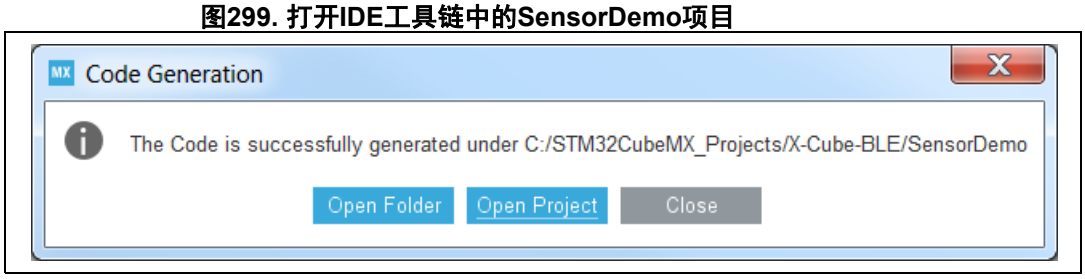


图300. 在Atollic® TrueStudio®中启动SensorDemo项目

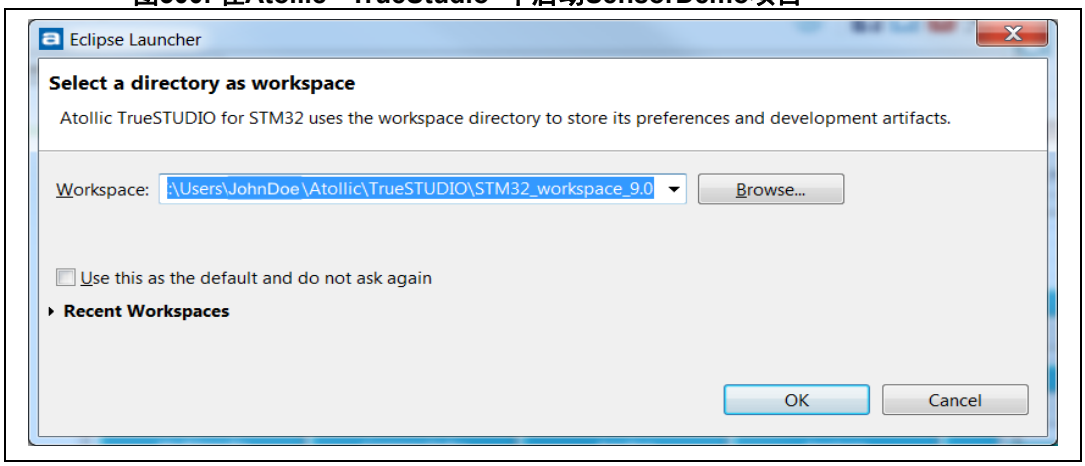
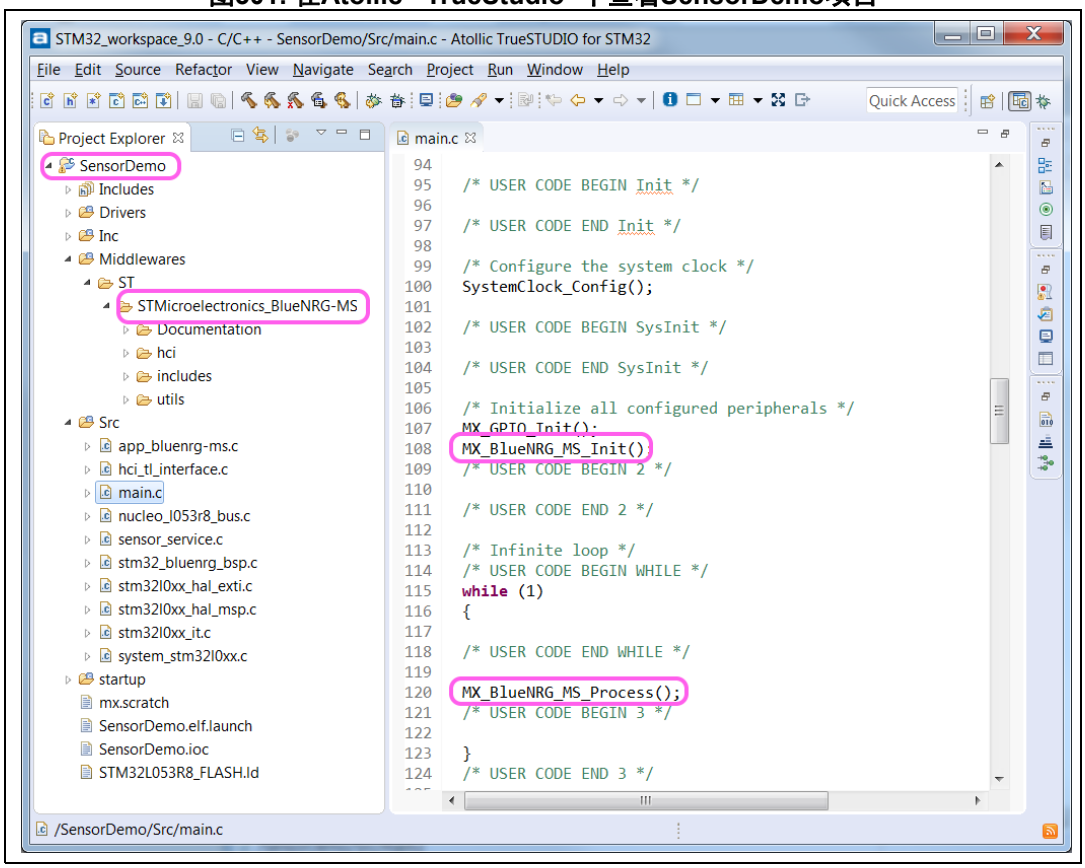
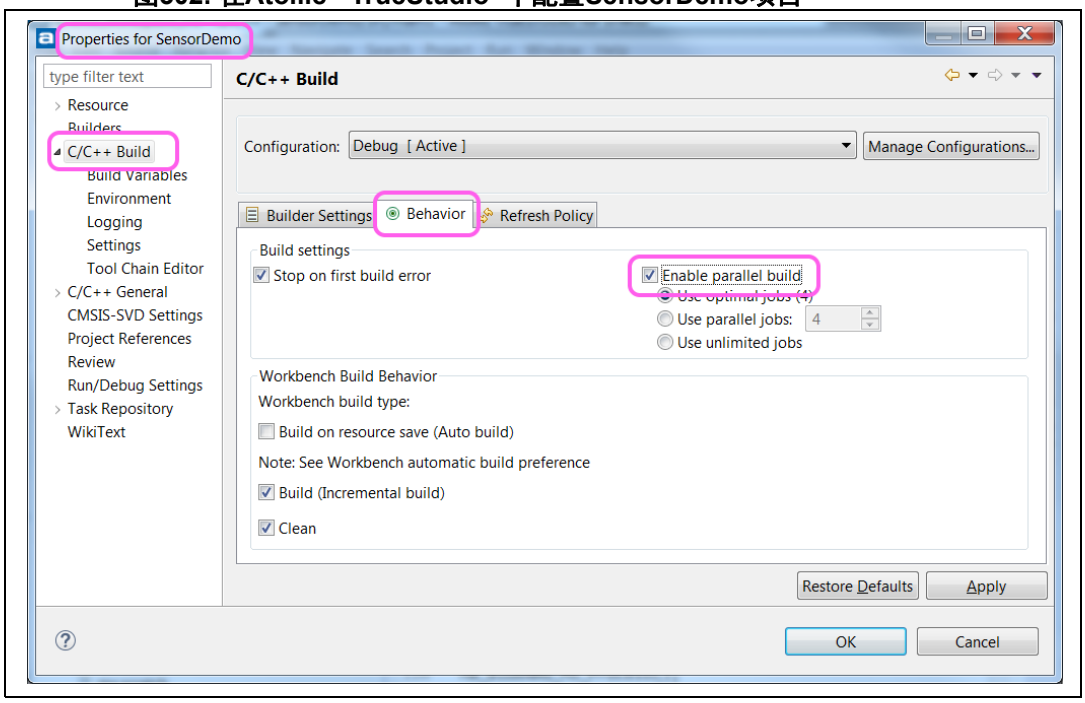


图301. 在Atollic® TrueStudio®中查看SensorDemo项目



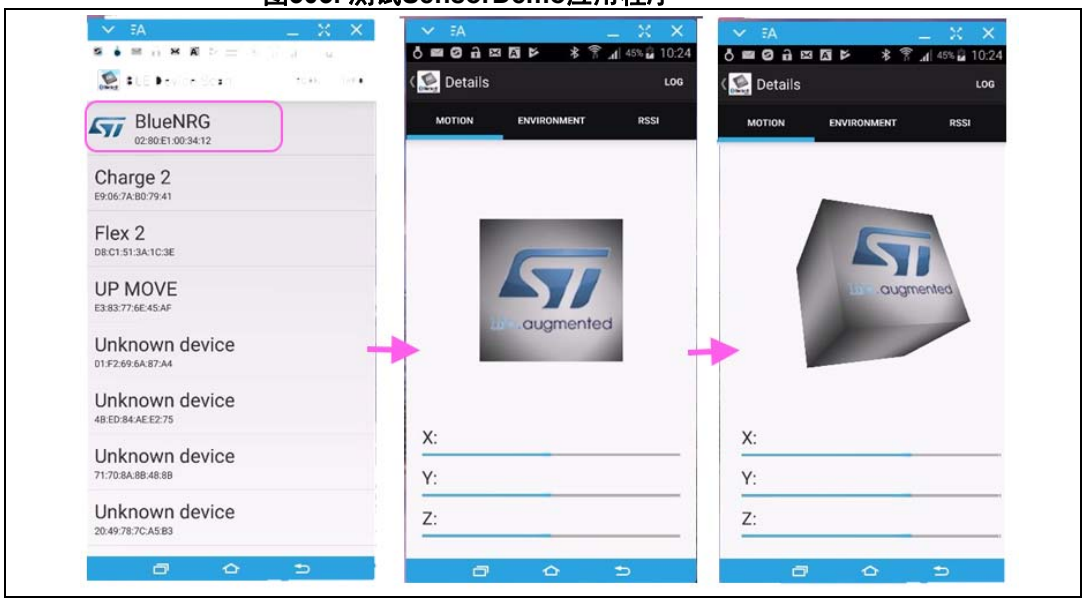
9. 从Atollic® TrueStudio®构建并运行SensorDemo应用程序：
   a) 配置项目属性 （请参阅*图 302*）
      在**“项目资源管理器”**面板中，右键单击项目名称（SensorDemo），然后选择**“属性”**以打开**“属性”**窗口。
      选择**“C/C ++构建”**，然后从**“行为”**选项卡中启用并行构建，以加快构建过程。
   b) 单击构建图标，，以构建项目。
   c) 通过USB电缆将计算机连接到Nucleo板ST-link连接器。
   d) 从**“运行”**菜单中**单击**以在板上运行项目。

图302. 在Atollic® TrueStudio®中配置SensorDemo项目



10. 通过在电话上启动BlueNRG应用程序来测试STM32 SensorDemo应用程序：
    e) 扫描附近的器件。
    f) 选择BlueNRG器件。
    g) 由于硬件上没有MEM感应元件，请按**蓝色按键**以模拟MEMs数据：每次按下该按键，ST Cube会旋转固定值（请参阅*图 303*）。

图303. 测试SensorDemo应用程序



18 FAQ

18.1 在“引脚排列配置”面板中，在我添加新的外设模式时，为什么STM32CubeMX会移动一些功能？

您可能已取消选择 Keep Current Signals Placement 。在这种情况下，工具会执行自动重新映射，以优化放置位置。

18.2 我如何手动强制进行功能重新映射？

使用**手动重新映射**功能。

18.3 为什么引脚布局视图中有一些引脚以黄色或浅绿色突出显示？为什么我不能更改一些引脚的功能（点击一些引脚时没有任何反应）？

这些引脚属于特定引脚（如电源或BOOT引脚），不可用作外设信号。

18.4 安装“Java 7更新45”或更新版的JRE时，为何会出现“Java 7更新45”错误？

此问题一般发生在64位Windows操作系统中，出错时，计算机中安装了多个Java™版本，并且64位Java™安装版本过旧。

STM32CubeMX安装过程中，计算机会搜索Java™的64位安装版本。

- 如果搜索到，则会检查“Java 7更新45”最低版本要求。如果安装的版本较旧，则会显示错误，要求进行升级。
- 如果未找到64位版本，STM32CubeMX会搜索32位版本。如果找到且版本过旧，则会显示“Java 7更新45”错误。用户必须更新安装版本才能解决此问题。

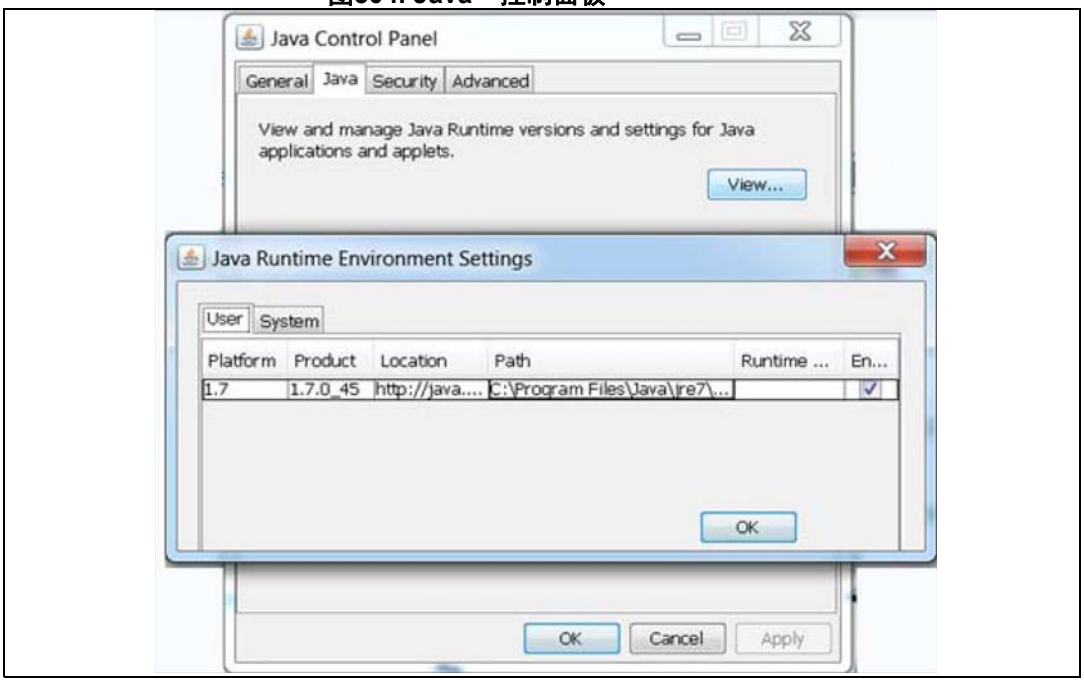
为避免出现这一问题，建议执行下列其中一项操作：

1. 删除所有Java™安装版本，然后只重新安装一个版本（32位或64位）（Java 7 update 45或更新版本）。
2. 保留32位和64位安装版本，但确保64位版本至少为Java 7 update 45。

注： *一些用户（例如Java开发者）可能需要检查定义硬编码Java路径的PC环境变量（JAVA_HOME或PATH）并对其进行更新，使其指向最新Java安装版本。*

在Windows 7上，可使用控制面板检查Java安装版本。为此，请双击控制面板\全部控制面板中的 Java 图标，以打开Java™设置窗口（参见*图 304*）。

图304. Java™控制面板



还可输入MS-DOS命令*"java –version"*以检查最新安装的Java版本（在此调用的Java程序是在C:\Windows\System32下安装的程序的副本）：

java version "1.7.0_45"

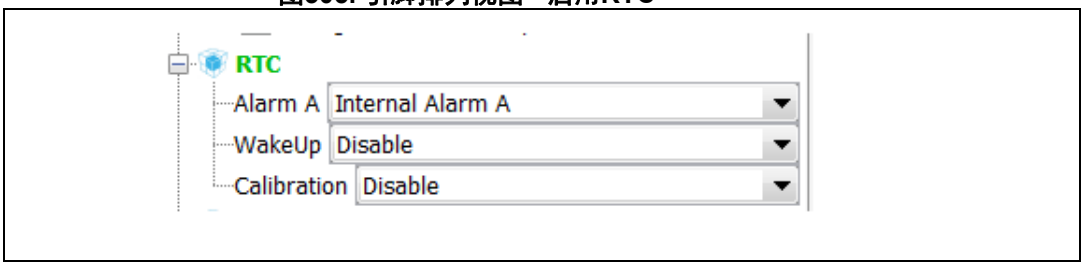
Java (TM) SE Runtime Environment (build 1.7.0_45-b18)

Java HotSpot (TM) 64-Bit Server VM (build 24.45-b08, mixed mode)

18.5 为何RTC复用器在时钟树视图中仍无效？

要启用RTC复用器，用户应在**引脚排列**视图中启用RTC外设，如下图所示。

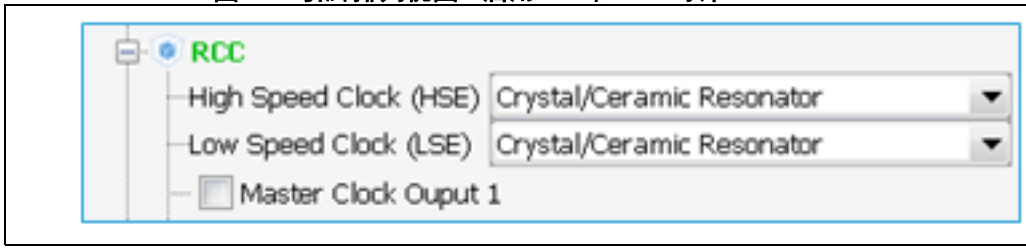
图305. 引脚排列视图 - 启用RTC



18.6 如何选择LSE和HSE作为时钟源并更改频率？

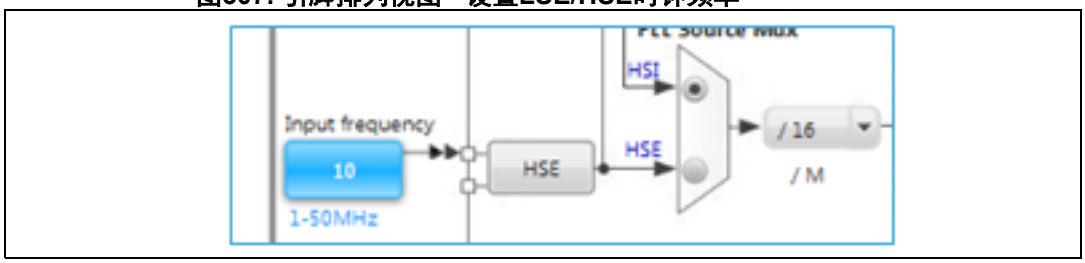
在**引脚排列**视图中依此对RCC进行配置后，LSE和HSE时钟会立即生效。相关示例，请参见*图 306*。

图306. 引脚排列视图 - 启用LSE和HSE时钟



随后可编辑时钟源频率并选择外部源，参见*图 307*。

图307. 引脚排列视图 - 设置LSE/HSE时钟频率



18.7 在PC13、PC14、PC15和PI8之一已配置为输出的情况下，为什么STM32CubeMX不允许我将其配置为输出？

STM32CubeMX执行在参考手册的“输出电压特征”表中以注脚形式记录的限制条件：

“PC13、PC14、PC15和PI8通过电源开关供电。由于该开关的灌电流能力有限（3mA），因此在输出模式下使用GPIO PC13到PC15和PI8时存在以下限制：速率不得超过2 MHz，最大负载为30 pF，这些I/O不能用作电流源（如用于驱动LED）。”

18.8 以太网配置：为什么有时候我不能指定DP83848或LAN8742A？

对于大多数系列，STM32CubeMX根据所选的以太网模式调整可能的PHY组件驱动程序的列表：

- 如果选择以太网MII模式，用户能够在DP83848组件驱动程序与“用户Phy”之间选择。
- 如果选择以太网RMII模式，用户能够在LAN8742A组件驱动程序与“用户Phy”之间选择。

如果选择“用户Phy”，用户必须手动添加组件驱动程序才能在其项目中使用。

注： *对于STM32H7系列，PHY被视为外部组件，不再在以太网外设配置下指定。用户可在LwIP“**平台设置**”选项卡下选择PHY。但由于STM32H7固件包只提供在所有STM32H7评估板和Nucleo板上可用的LAN8742A组件的驱动程序代码，因此STM32CubeMX用户界面仅用于在“用户Phy”与LAN8742之间选择。*

如果选择LAN8742，BSP驱动程序代码会复制到生成的项目中。

附录A STM32CubeMX引脚分配规则

STM32CubeMX中实行以下引脚分配规则：

- 规则1：块一致性
- 规则2：块相关性
- 规则3：一个块 = 一种外设模式
- 规则4：块重新映射（仅限STM32F10x）
- 规则5：功能重新映射
- 规则6：块转移（仅限STM32F10x）
- 规则7：设置或清除外设模式
- 规则8：分别映射功能（如果未选中“保留当前放置位置”）
- 规则9：GPIO信号映射

A.1 块一致性

设置引脚信号时（假定相应外设模式没有不明确的内容），会映射该模式需要使用的所有引脚/信号，并且引脚会以绿色显示（否则配置的引脚会以橙色显示）。

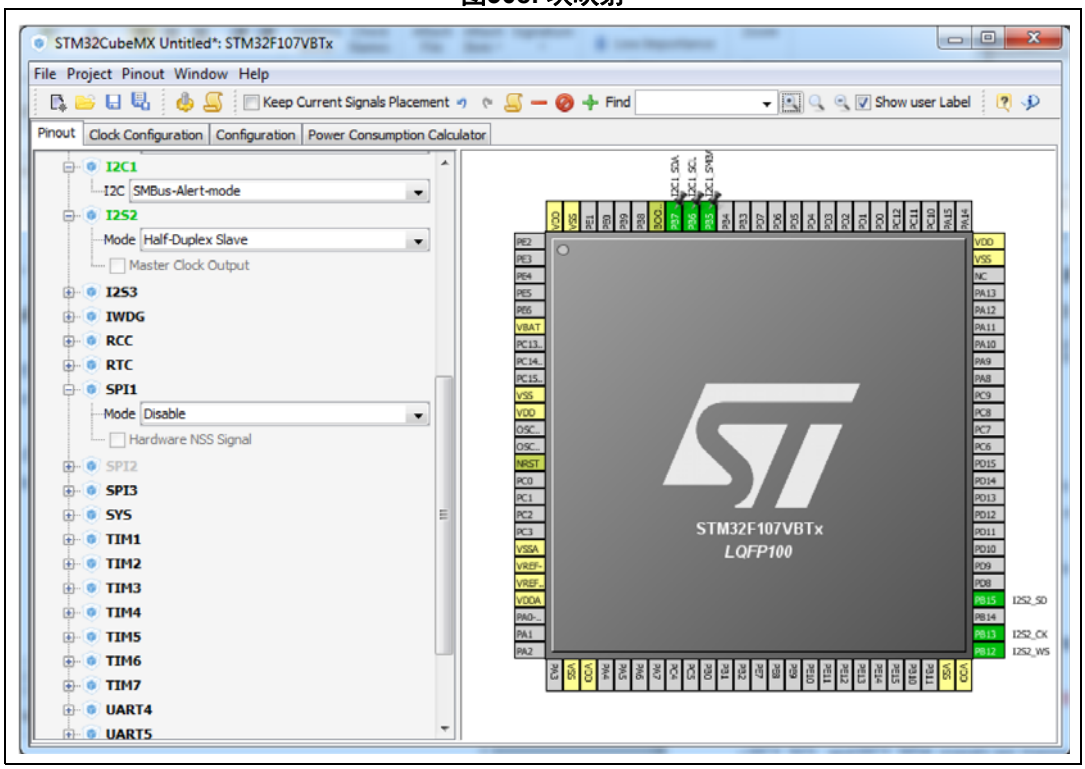
清除引脚信号时，会同时取消该模式需要使用的所有引脚/信号的映射，引脚变回灰色。

STM32F107x MCU块映射示例

如果用户为PB5分配I2C1_SMBA功能，则STM32CubeMX对引脚和模式进行如下配置：

- I2C1_SCL和I2C1_SDA信号分别映射到PB6和PB7引脚（参见*图 308*）。
- I2C1外设模式设为SMBus-Alert模式。

图308. 块映射

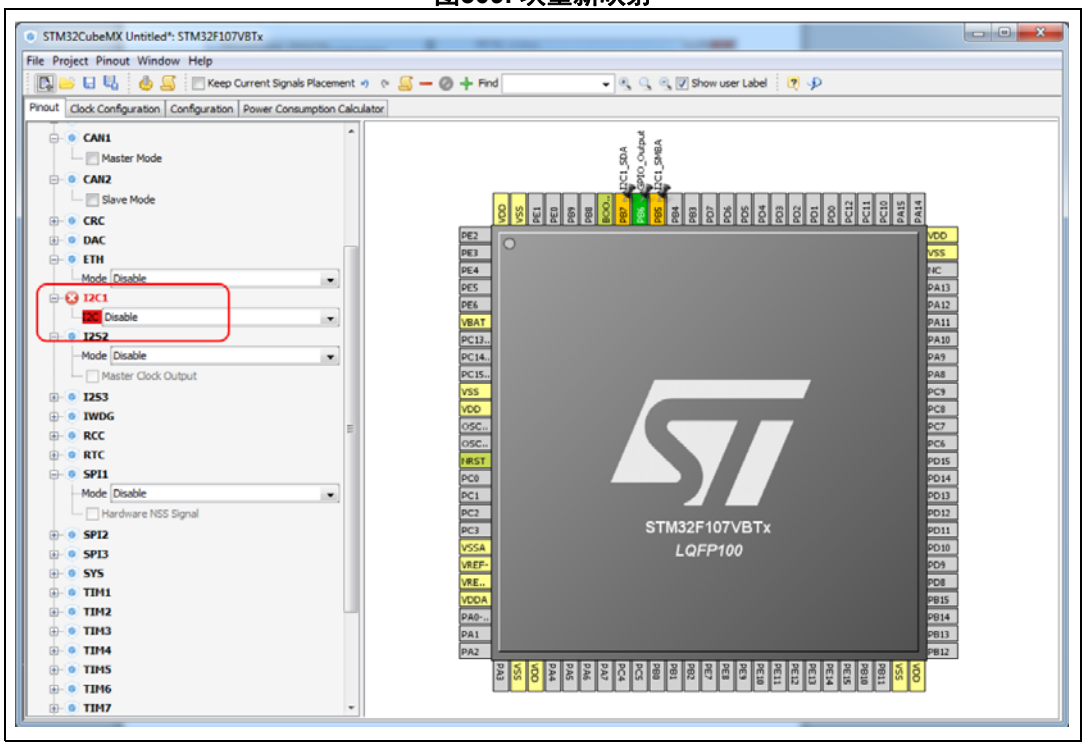


STM32F107x MCU块重新映射示例

如果用户为PB6分配GPIO_Output，STM32CubeMX会在外设树形视图中自动禁用I2C1 SMBus-Alert外设模式，并更新其他I2C1引脚（PB5和PB7），具体如下：

- 如果引脚已取消固定，则引脚配置复位（引脚以灰色显示）。
- 如果引脚已固定，则保留分配给引脚的外设信号，引脚会以橙色突出显示，因为引脚不再与外设模式匹配（参见*图 309*）。

图309. 块重新映射



为使STM32CubeMX找到I2C外设模式的替代解决方案，用户需要取消固定I2C1引脚并从外设树形视图中选择I2C1模式（参见*图 310*和*图 311*）。

图310. 块重新映射 - 示例1

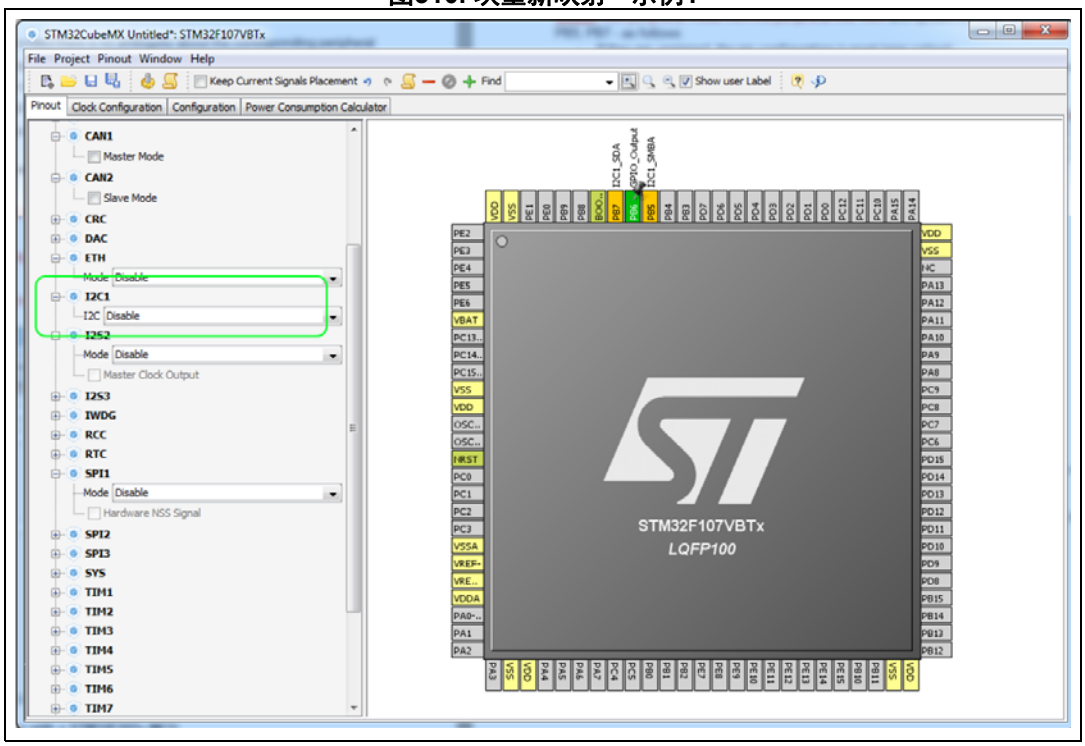
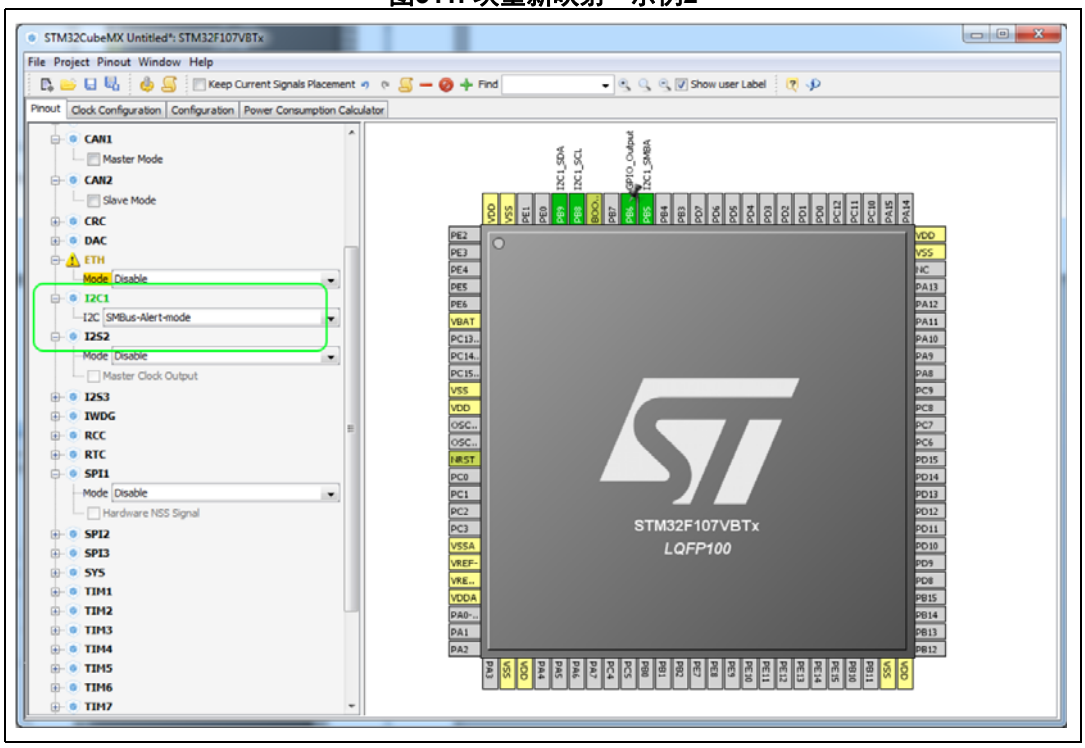


图311. 块重新映射 - 示例2



A.2 块相关性

在**引脚布局**视图中，同一信号可显示为多个引脚的替代功能。但该信号仅可映射一次。

因此，对于STM32F1 MCU，不能同时为同一外设模式选择两个引脚块：如果选择一个块/块发出的信号，则会清除备用块。

STM32F107x MCU在全双工主模式下SPI的块重新映射示例

如果在树形视图中选择SPI1全双工主模式，则相应的SPI信号会默认分配给PB3、PB4和PB5引脚（参见*图 312*）。

如果用户为PA6分配的是当前分配给PB4的SPI1_MISO功能，则STM32CubeMX会清除PB4引脚的SPI1_MISO功能以及为该块配置的其他所有引脚，并将相应的SPI1功能移至相关引脚，这些引脚的功能块与PB4引脚相同（参见*图 313*）。

（按下CTRL并点击PB4可以蓝色显示PA6替代功能，然后将信号拖放到引脚PA6）

图312. 块相关性 - SPI信号分配给PB3/4/5

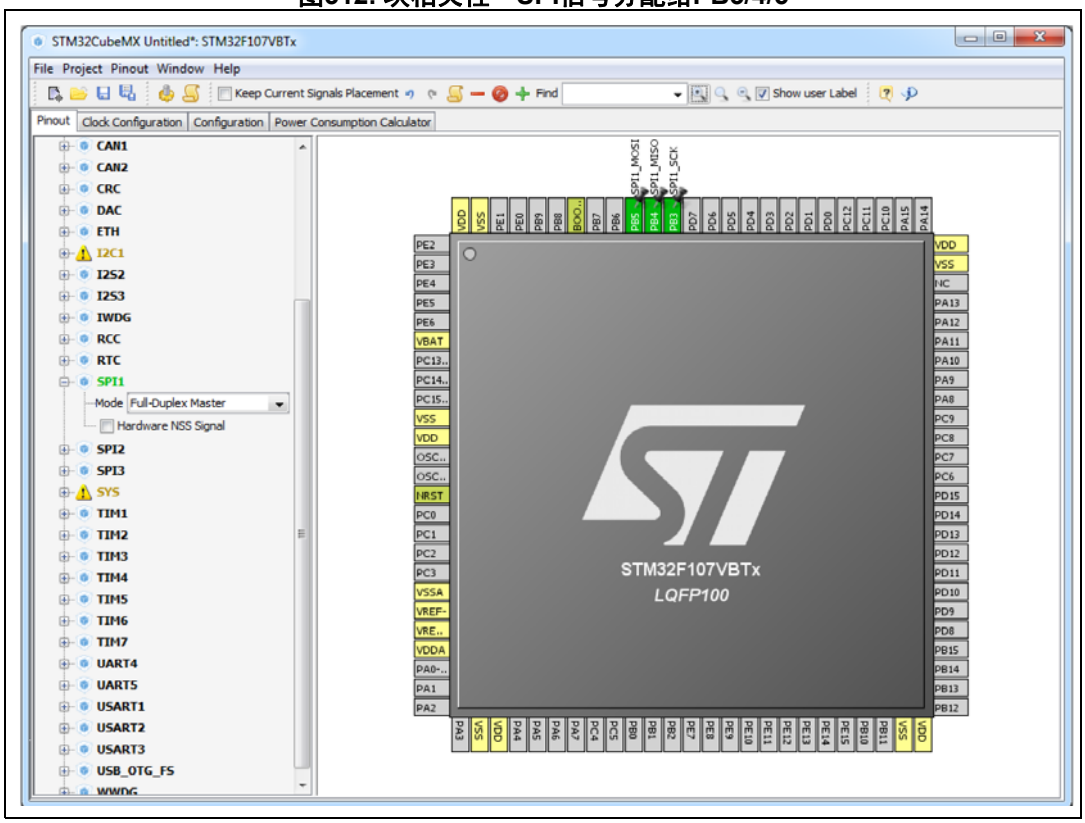
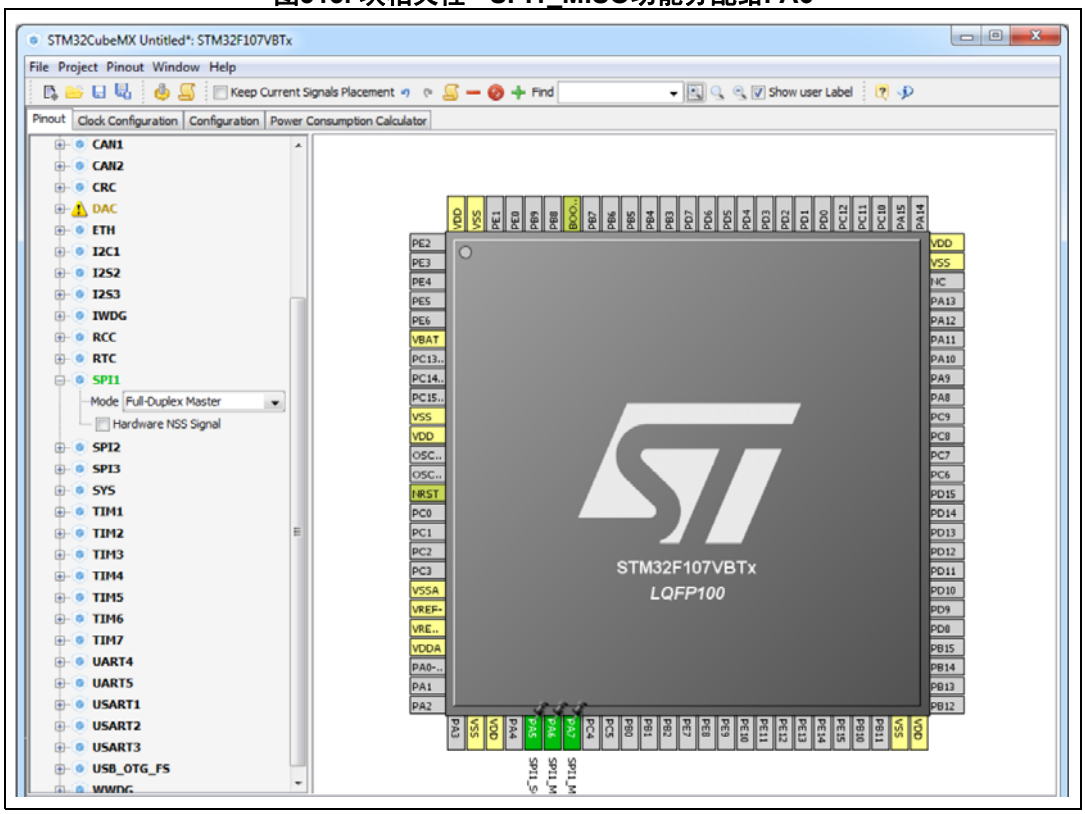


图313. 块相关性 - SPI1_MISO功能分配给PA6



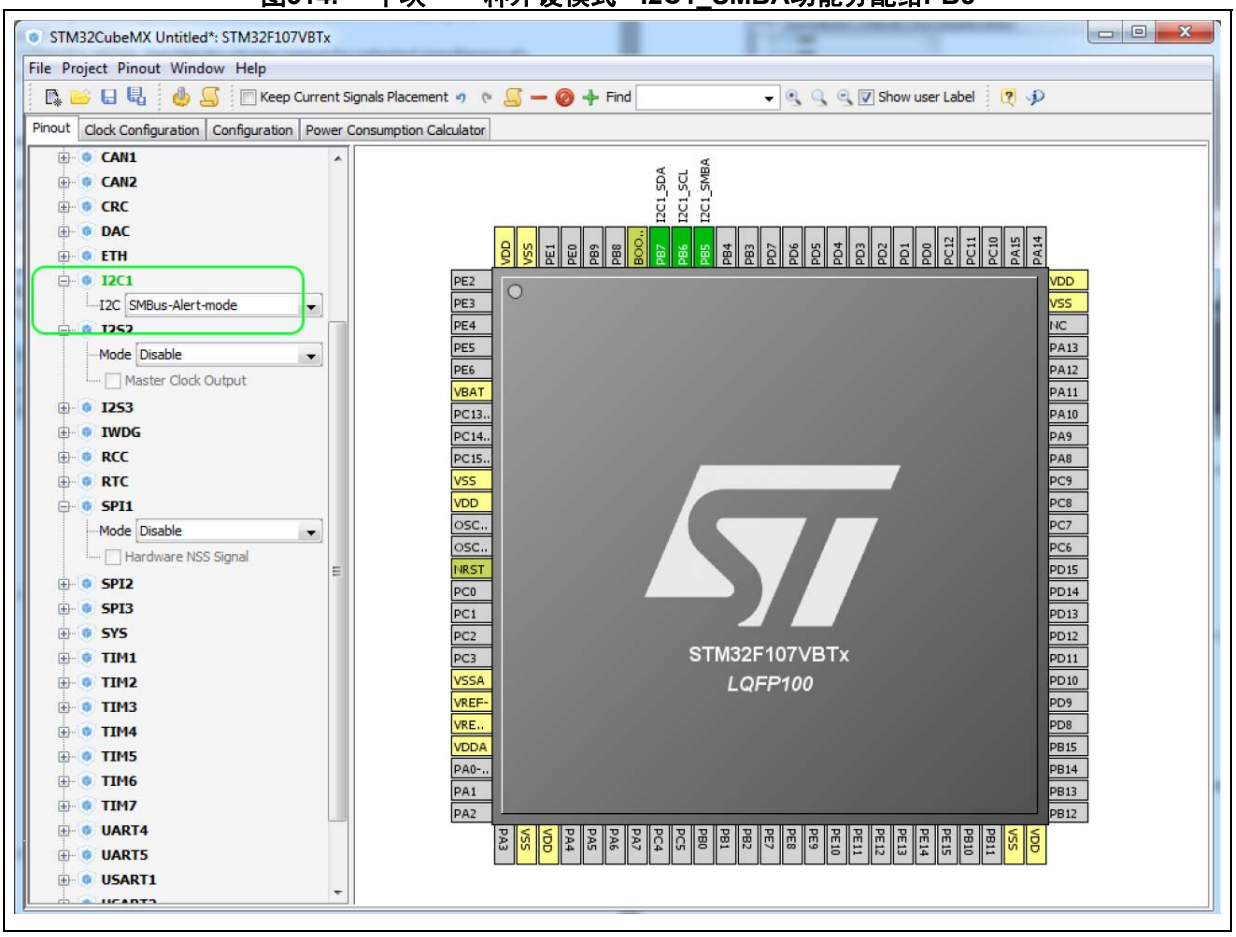
A.3 一个块 = 一种外设模式

如果已在**引脚布局**视图中对引脚块进行全面配置（以绿色显示），则会在外设树中自动设置相关外设模式。

STM32F107x MCU示例

将I2C1_SMBA功能分配给PB5会自动将I2C1外设配置为SMBus-Alert模式（参见*图 314*中的外设树）。

图314. 一个块 = 一种外设模式 - I2C1_SMBA功能分配给PB5



A.4 块重新映射（仅限STM32F10x）

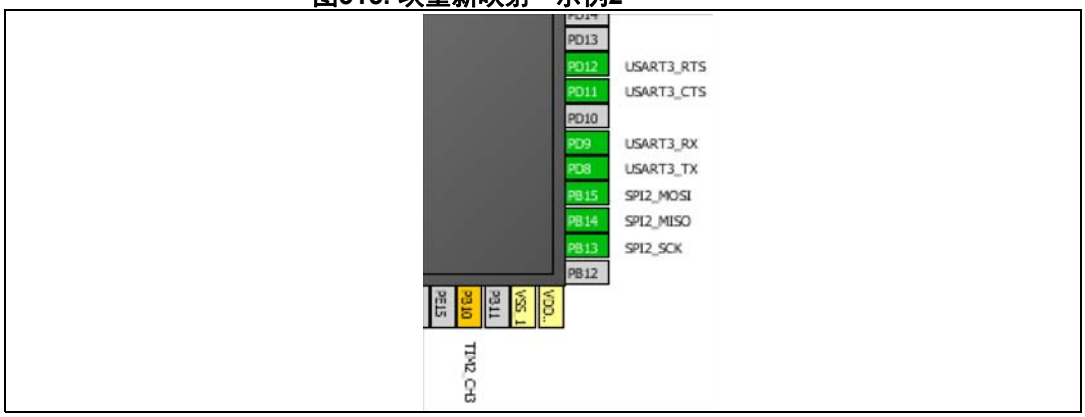
要配置外设模式，STM32CubeMX会选择引脚块，并将各模式信号分配给此块中的一个引脚。这样它便会搜索模式可映射到的第一个空闲块。

设置外设模式时，如果默认块中至少有一个引脚已使用，STM32CubeMX会尝试找到备用块。如果未找到替代快，则选择另一序列中的功能，或取消选中 Keep Current Signals Placement，并重新映射所有块，以找到解决方案。

示例

STM32CubeMX将USART3硬件流程控制模式重新映射到（PD8-PD9-PD11-PD12）块，因为USART3默认块的PB14已分配给SPI2_MISO功能（参见*图 315*）。

图315. 块重新映射 - 示例2



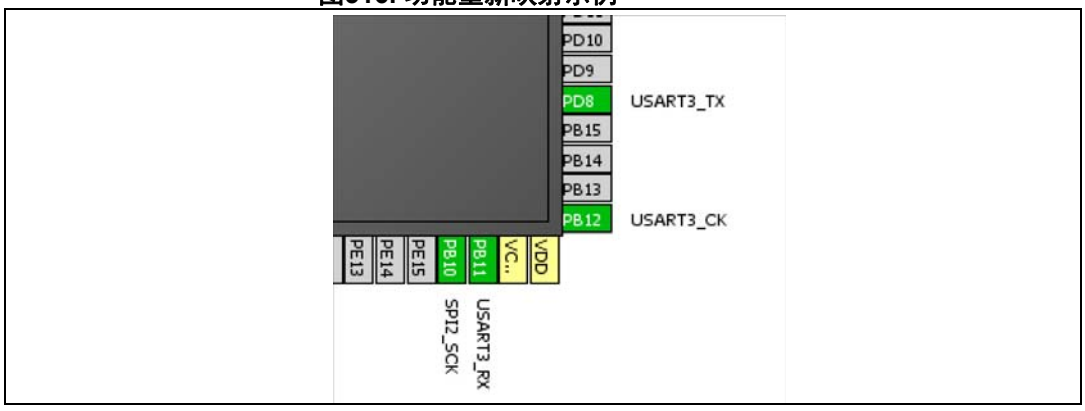
A.5 功能重新映射

要配置外设模式，STM32CubeMX会将模式的各个信号分配给引脚。这样它便会搜索信号可映射到的第一个空闲引脚。

使用STM32F415x的示例

为同步模式配置USART3时，STM32CubeMX发现USART3_TX信号的默认PB10引脚已被SPI使用。因此会将其重新映射为PD8（参见*图 316*）。

图316. 功能重新映射示例



A.6 块转移（仅适用于STM32F10x，且“保留当前信号放置位置”已取消选中）

如果块无法映射，并且不存在可用的替代解决方案，STM32CubeMX会尝试重新映射受共用引脚影响的所有外设模式，以释放引脚。

示例

在启用“保留当前信号放置位置”的情况下，如果先设置了USART3同步模式，则会映射异步默认块（PB10-PB11），以太网将不可用（以红色显示）（参见*图 317*）。

取消选中 Keep Current Signals Placement ，STM32CubeMX可转移块并为以太网MII模式释放块。（见*图 318*）。

图317. 未应用块转移

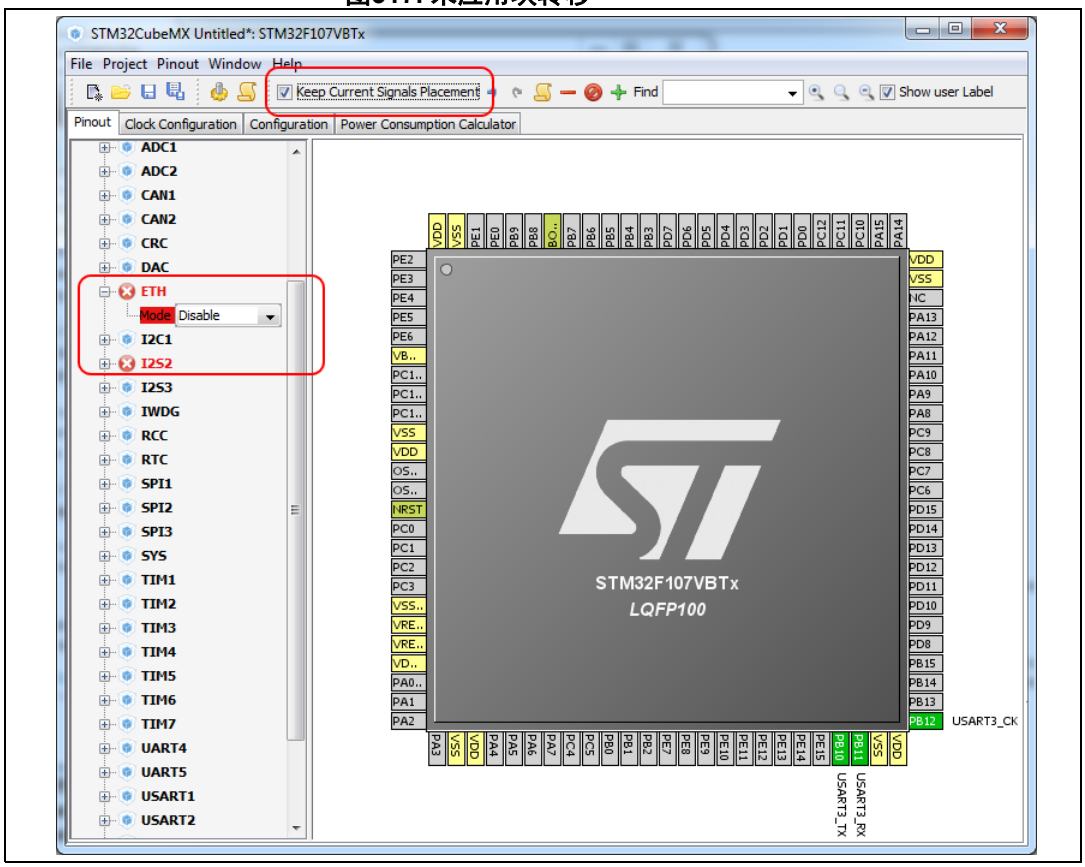
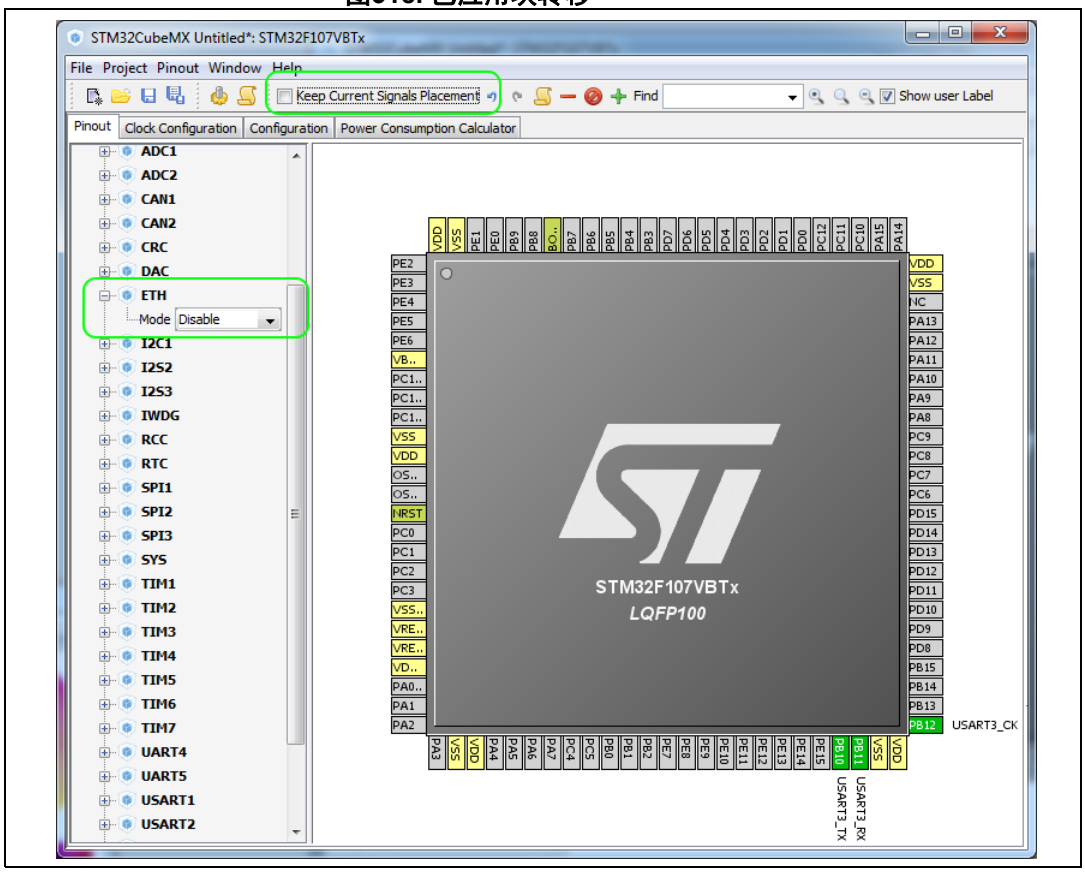


图318. 已应用块转移



A.7 设置或清除外设模式

“外设”面板和**引脚布局**视图相互关联：如果已设置或清除外设模式，则会设置或清除对应的引脚功能。

A.8 分别映射功能

如果STM32CubeMX需要已手动分配某一功能的引脚（未设置外设模式），则仅在 Keep Current Signals Placement 已取消选中且此功能未固定（无固定图标）的情况下，才能将此功能移至另一引脚。

A.9 GPIO信号映射

I/O信号（GPIO_Input、GPIO_Output、GPIO_Analog）可通过**引脚布局**视图手动分配给引脚，也可以通过**“引脚布局”**菜单自动分配。此类引脚不能再自动分配给另一信号：STM32CubeMX信号自动放置不会再考虑此引脚，因为STM32CubeMX不会将I/O信号转移到其他引脚。

引脚仍可手动分配给另一信号或分配为复位状态。

附录B STM32CubeMXC代码生成设计选择和限制

B.1 STM32CubeMX生成的C代码和用户部分

STM32CubeMX生成的C代码提供用户部分，如下所示。用户部分允许插入用户C代码并在下一次生成C代码时保留。

不应对用户部分进行移动或重命名。仅会保留由STM32CubeMX定义的用户部分。下次生成C代码时，用户创建的部分将被忽略并丢弃。

```
/* 用户代码开始 0 */
(..)
/* 用户代码结束 0 */
```

注： *STM32CubeMX可在一些用户部分生成C代码。将由用户决定是否清除该部分中可能不适用的部分。例如，主函数中的while(1)循环放置在用户部分内，如下所示：*

```
/* 无限循环 */
 /* 用户代码开始WHILE */
 while (1)
 {
 /* 用户代码结束WHILE */

 /* 用户代码开始 3 */
 }
/* 用户代码结束 3 */
```

B.2 STM32CubeMX外设初始化设计选择

STM32CubeMX生成的外设_*Init*函数可通过 MX_前缀轻松识别：

```
static void MX_GPIO_Init(void);
static void MX_<Peripheral Instance Name>_Init(void);
static void MX_I2S2_Init(void);
```

每个由用户选择的外设实例都有*MX_<外设实例名称>_Init*函数（例如*MX_I2S2_Init*）。该函数会执行HAL驱动程序初始化（例如*HAL_I2S_Init*）所需的相关句柄结构初始化（例如，&hi2s2代表I2S的第二个实例），并会实际调用此函数：

```
void MX_I2S2_Init(void)
{
 hi2s2.Instance = SPI2;
 hi2s2.Init.Mode = I2S_MODE_MASTER_TX;
 hi2s2.Init.Standard = I2S_STANDARD_PHILLIPS;
 hi2s2.Init.DataFormat = I2S_DATAFORMAT_16B;
 hi2s2.Init.MCLKOutput = I2S_MCLKOUTPUT_DISABLE;
 hi2s2.Init.AudioFreq = I2S_AUDIOFREQ_192K;
 hi2s2.Init.CPOL = I2S_CPOL_LOW;
 hi2s2.Init.ClockSource = I2S_CLOCK_PLL;
 hi2s2.Init.FullDuplexMode = I2S_FULLDUPLEXMODE_ENABLE;
 HAL_I2S_Init(&hi2s2);
```

```
}
```

外设初始化默认在*main.c*中完成。如果外设用于中间件模式，则外设初始化可在中间件对应的.c文件中完成。

自定义*HAL_<外设名称>_MspInit()* 函数在stm32f4xx_hal_msp.c文件中创建，用于配置所选外设的低级硬件（GPIO、CLOCK）。

B.3 STM32CubeMX中间件初始化设计选择和限制中间件初始化

B.3.1 概述

STM32CubeMX不支持在中间件堆栈本机文件中插入C用户代码（虽然LwIP等堆栈在一些用例中可能需要使用C用户代码）。

STM32CubeMX生成的中间件*Init*函数可通过 MX_前缀轻松识别：

```
MX_LWIP_Init(); // defined in lwip.h file
MX_USB_HOST_Init(); // defined in usb_host.h file
MX_FATFS_Init(); // defined in fatfs.h file
```

但需要注意下列例外情况：

- 除非用户在“项目设置”窗口中选择生成*Init*函数作为.c/.h文件对，否则不会为FreeRTOS生成*Init*函数， 而是在*main.c*文件中定义*StartDefaultTask*函数，并在主函数中调用CMSIS-RTOS本机函数(*osKernelStart*)。
- 如果FreeRTOS已启用，则从main.c文件中的*StartDefaultTask*函数调用其他正在使用的中间件的*Init*函数。

  示例：

```
void StartDefaultTask(void const * argument)
{
    /* FATFS初始化代码 */
    MX_FATFS_Init();
    /* LWIP初始化代码 */
    MX_LWIP_Init();
 /* USB_HOST初始化代码 */
 MX_USB_HOST_Init();
 /* 用户代码开始 5 */
 /* 无限循环 */
 用于(;;)
 {
   osDelay(1);
 }
 /* 用户代码结束 5 */
}
```

B.3.2 USB 主机

USB外设初始化在*usbh_conf.c*文件中的中间件初始化C代码中执行，而USB堆栈初始化则在*usb_host.c*文件中执行。

使用USB主机中间件时，用户负责在生成的*usb_host.c*文件中实现*USBH_UserProcess*回调函数。

如果应用需要在各个类之间进行动态切换，用户可通过STM32CubeMX用户界面选择注册一个类或全部类。

B.3.3 USB设备

USB外设初始化在*usbd_conf.c*文件中的中间件初始化C代码中执行，而USB堆栈初始化则在*usb_host.c*文件中执行。

USB VID、PID和字符串描述符通过STM32CubeMX用户界面配置，并在*usbd_desc.c*生成的文件中可用。其他标准描述符（配置、接口）在同一文件中被硬编码，以免支持USB复合设备。

使用USB设备中间件时，用户负责在所有设备类的*usbd_<classname>_if.c*类接口文件（例如usbd_storage_if.c）中实现函数。

不支持USB MTP和CCID类。

B.3.4 FatFs

FatFs是适用于小型嵌入式系统的通用FAT/exFAT文件系统解决方案。

FatFs配置在*ffconf.h*生成的文件中提供。

SDIO外设（用于FatFs SD卡模式）和FMC外设（用于FatFs外部SDRAM和外部SRAM模式）的初始化保留在*main.c*文件中。

一些文件需要由用户修改，以适应用户板子的特殊性（可将STM32Cube嵌入式软件包中的BSP作为示例）：

- 使用FatFs SD卡模式时*bsp_driver_sd.c/.h*生成的文件
- 使用FatFs外部SRAM模式时*bsp_driver_sram.c/.h*生成的文件
- 使用FatFs外部SDRAM模式时*bsp_driver_sdram.c/.h*生成的文件。

支持多驱动FatFs，这意味着应用可使用多个逻辑驱动（外部SDRAM、外部SRAM、SD卡、U盘、用户自定义驱动）。但不支持特定逻辑驱动的多个实例（例如，使用USB主机的两个实例或多个RAM磁盘的FatFs）。

不支持NOR和NAND Flash存储器。在这种情况下，用户应选择FatFs用户自定义模式，并更新为在中间件与所选外设之间实现接口而生成的*user_diskio.c*驱动文件。

B.3.5 FreeRTOS

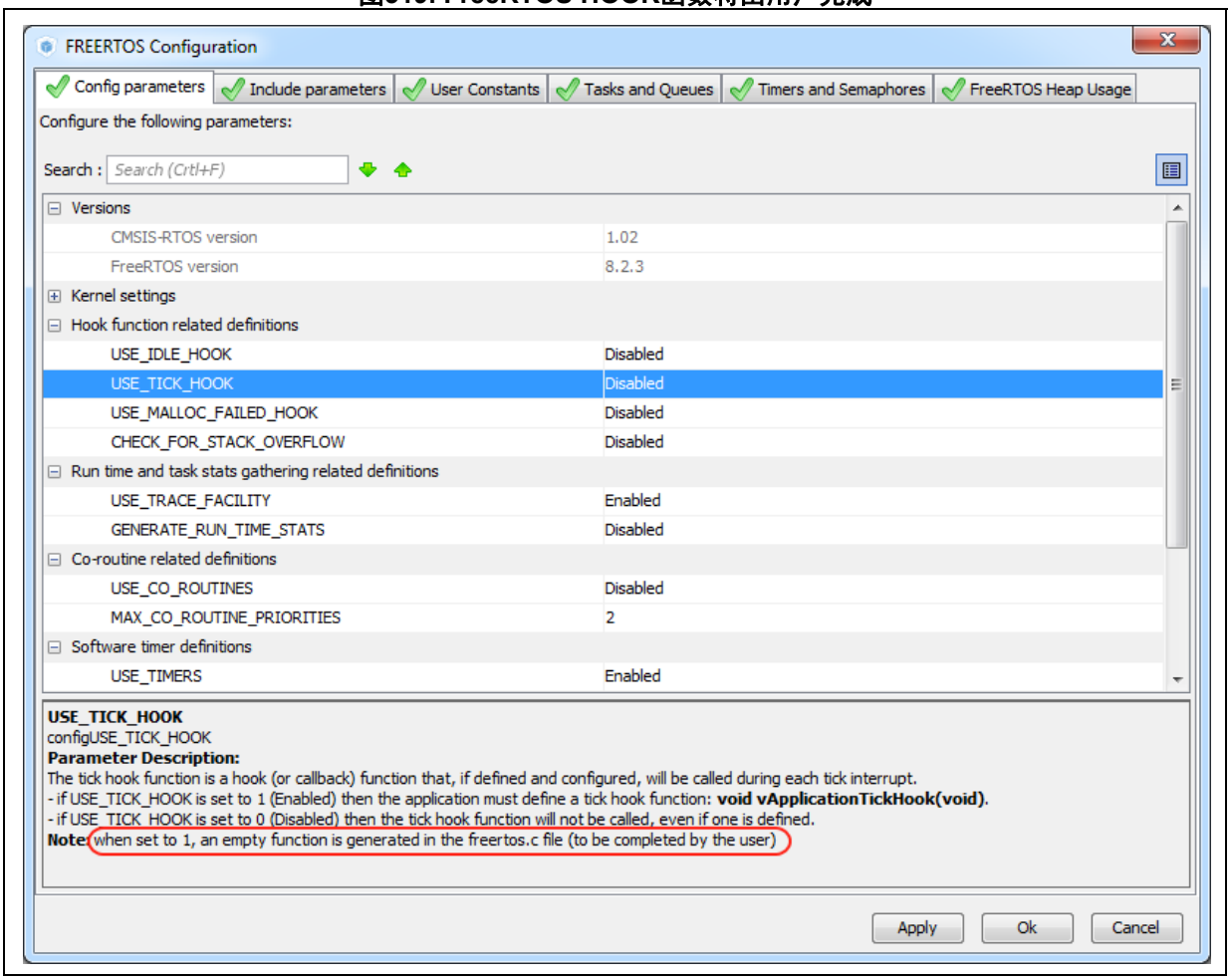
FreeRTOS是适用于微控制器的免费实时嵌入式操作系统。

FreeRTOS配置在*FreeRTOSConfig.h*生成的文件中提供。

如果已启用FreeRTOS，选择的其他所有中间件模式（例如LwIP、FatFs、USB）将在main.c文件中的同一FreeRTOS线程中进行初始化。

如果GENERATE_RUN_TIME_STATS、CHECK_FOR_STACK_OVERFLOW、USE_IDLE_HOOK、USE_TICK_HOOK和USE_MALLOC_FAILED_HOOK参数已激活，STM32CubeMX会生成*freertos.c*文件，其中包含用户应实现的空函数。工具提示会对此突出显示（参见*图 319*）。

图319. FreeRTOS HOOK函数将由用户完成



B.3.6 LwIP

LwIP是TCP/IP协议栈的小规模独立实现：它减少了RAM的使用，因此适用于RAM为几十KB的嵌入式系统。

LwIP初始化函数在*lwip.c*中定义，而LwIP配置则在*lwipopts.h* 生成的文件中提供。

STM32CubeMX仅支持以太网LwIP。以太网外设初始化在中间件初始化C代码中完成。

STM32CubeMX不支持在堆栈本机文件中插入用户C代码。但一些LwIP用例需要修改堆栈本机文件（例如*cc.h*、*mib2.c*）：应对用户修改进行备份，否则将会在下一次生成STM32CubeMX时丢失。

从LwIP版本1.5开始，STM32CubeMX LwIP支持IPv6（参见*图 321*）。

必须禁用DHCP才能配置静态IP地址。

图320. LwIP 1.4.1配置

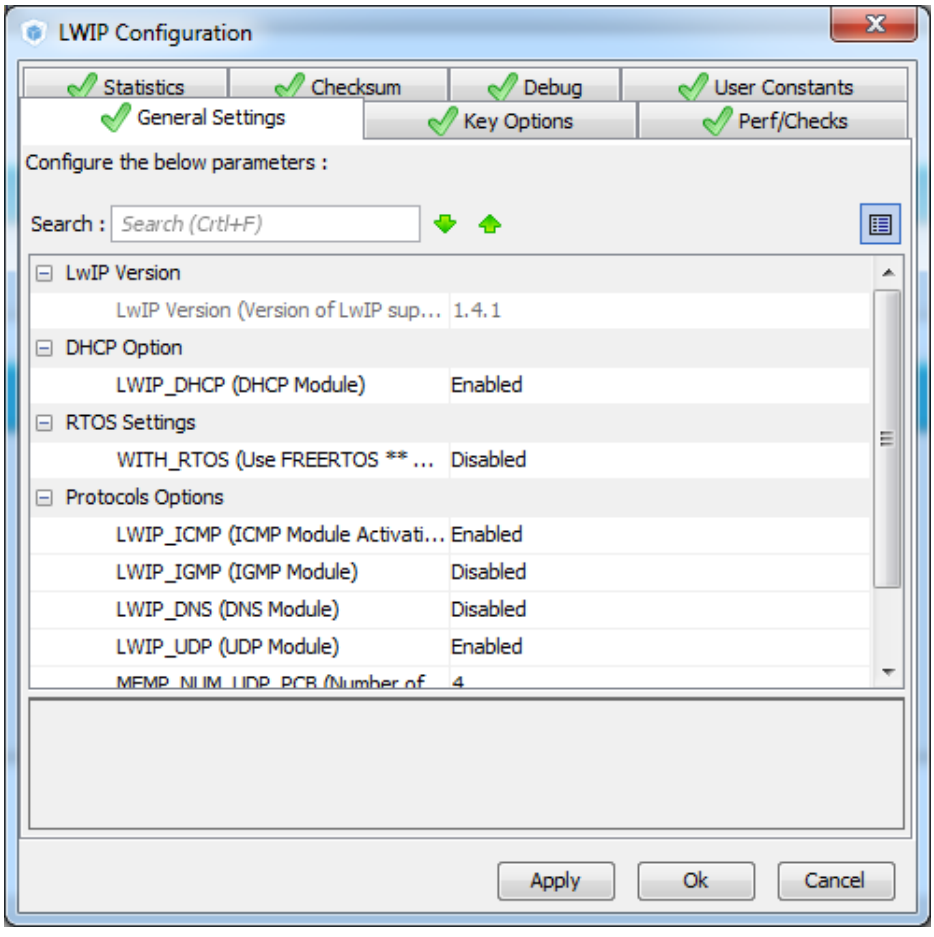
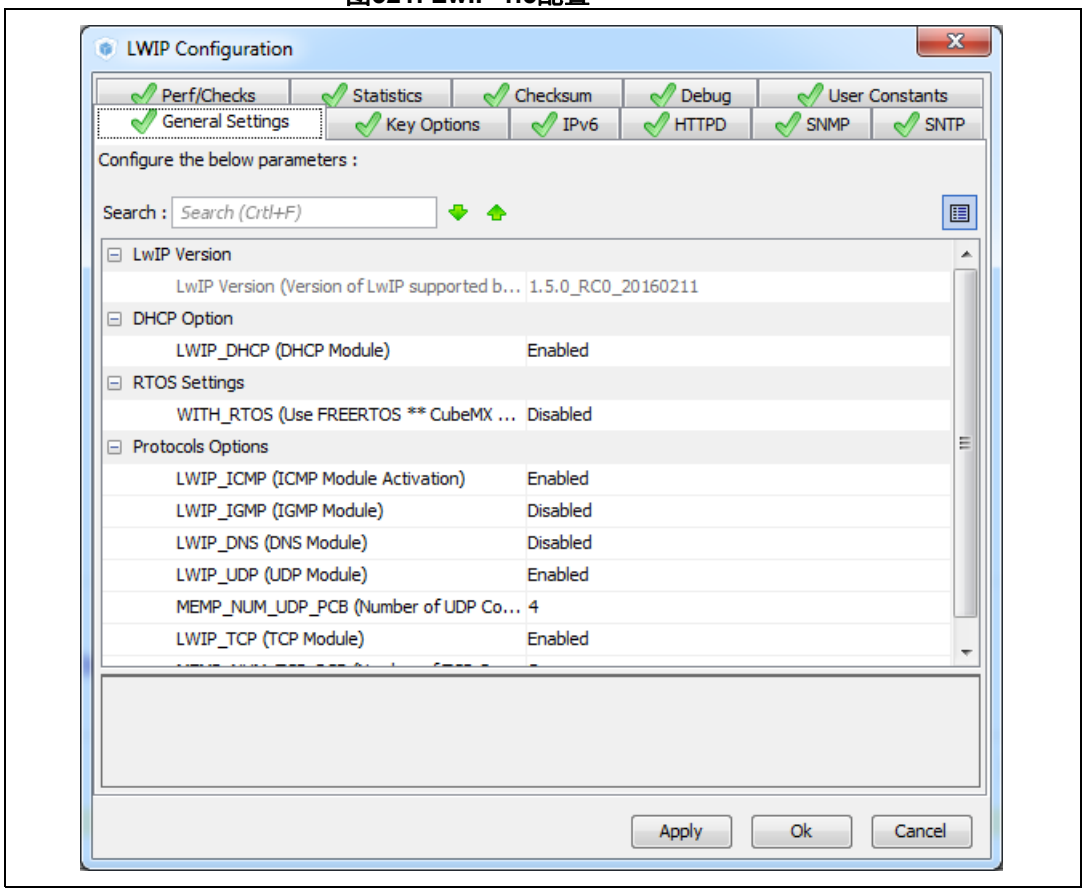


图321. LwIP 1.5配置



如果特定参数已启用（默认禁用），STM32CubeMX生成的C代码将报告编译错误。用户必须使用堆栈补丁（从网站下载）或用户C代码解决问题。以下参数会生成错误：

- MEM_USE_POOLS：用户C代码将添加到*lwipopts.h*或*cc.h*（堆栈文件）中。
- PPP_SUPPORT、PPPOE_SUPPORT：需要用户C代码
- MEMP_SEPARATE_POOLS，MEMP_OVERFLOW_CHECK > 0：需要堆栈补丁
- MEM_LIBC_MALLOC & RTOS启用：需要堆栈补丁
- LWIP_EVENT_API：需要堆栈补丁

在STM32CubeMX中，用户必须启用FreeRTOS才能通过netconn和套接字API使用LwIP。这些API需要使用线程，因此需要使用操作系统。如果不启用FreeRTOS，则仅可使用受LwIP事件驱动的Raw API。

B.3.7 Libjpeg

Libjpeg这一广泛使用的C库允许读取和写入JPEG文件。该库在STM32CubeF7、STM32CubeH7、STM32CubeF2以及STM32CubeF4嵌入式软件包中提供。

STM32CubeMX生成以下文件，用户可通过STM32CubeMX用户界面配置文件内容：

- **libjpeg.c/.h**

  *MX_LIBJPEG_Init()*初始化函数在libjpeg.c文件中生成。该函数为空。由用户决定是否在用户部分输入代码并调用应用需要的libjpeg函数。
- **jdata_conf.c**

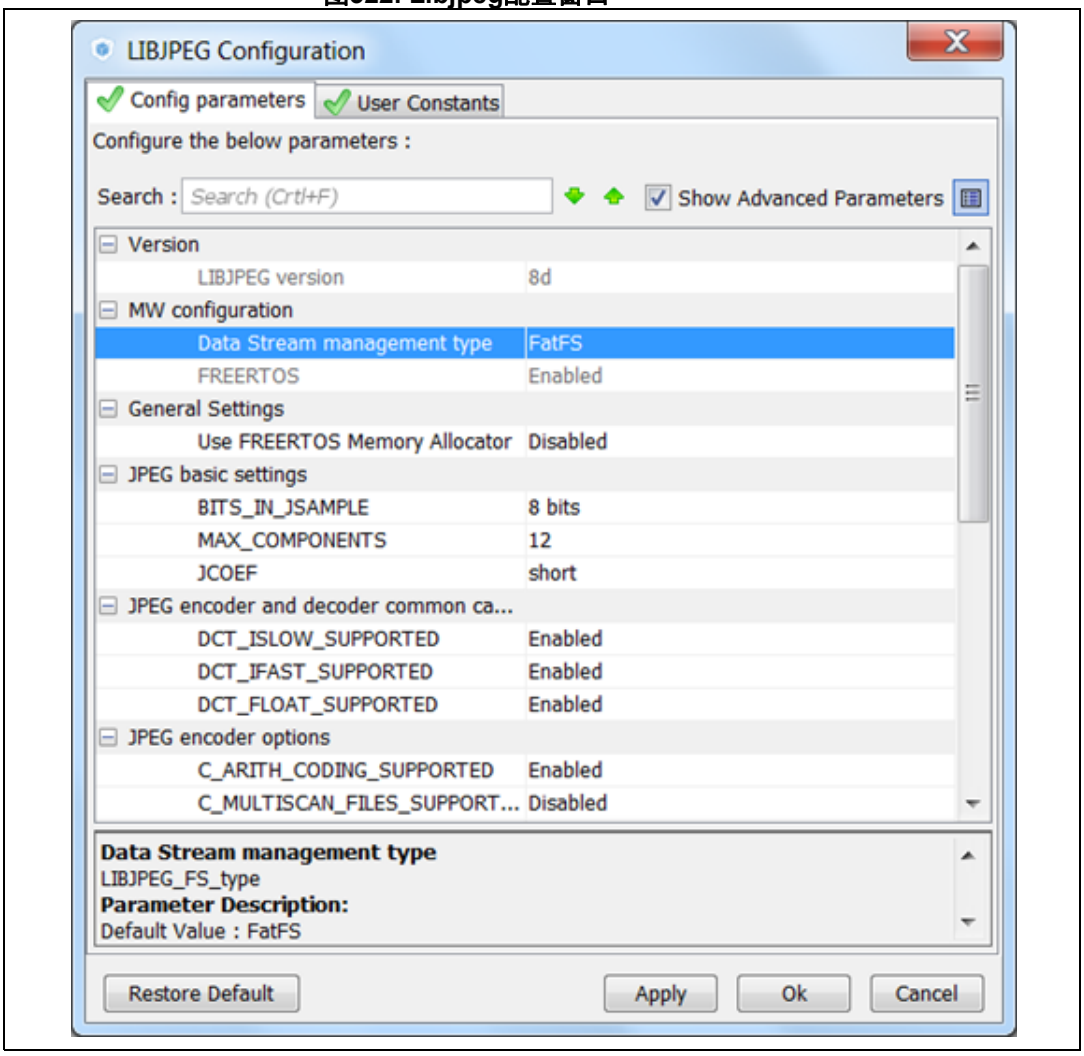
  仅当FatFs被选为数据流管理类型时，才会生成此文件。
- **jdata_conf.h**

  该文件的内容会根据选择的数据流管理类型进行调整。
- **jconfig.h**

  该文件由STM32CubeMX生成。但无法配置。
- **jmorecfg.h**

  该文件中包含的一些（并非全部）定义语句可通过STM32CubeMX libjpeg配置菜单修改。

图322. Libjpeg配置窗口



B.3.8 Mbed TLS

Mbed TLS这一C库允许为嵌入式产品添加加密功能。它用于处理安全套接层（SSL）和传输层安全（TLS）协议，这两种协议用于通过安全网络在两者之间建立安全、加密、经过认证的链接。Mbed TLS配有直观的API，并最大限度地减少了编码足迹。如需了解详细信息，请访问https://tls.mbed.org/。

Mbed TLS在STM32CubeF2、STM32CubeF4、STM32CubeF7以及STM32CubeH7嵌入式软件包中提供。

Mbed TLS可以在没有LwIP堆栈的情况下运行（参见 *图 323：Mbed TLS（无LwIP）*）。

如果使用LwIP堆栈，还必须启用FreeRTOS（参见 *图 324：Mbed TLS（有LwIP和FreeRTOS）*）。

STM32CubeMX生成以下文件，用户可通过STM32CubeMX用户界面（参见 *图 325：Mbed TLS配置窗口*）和/或使用代码自身的用户部分修改这些文件的内容：

- *mbedtls_config.h*
- *mbedtls.h*
- *net_sockets.c* （仅在LwIP启用的情况下生成）
- *mbedtls.c*

图323. Mbed TLS（无LwIP）

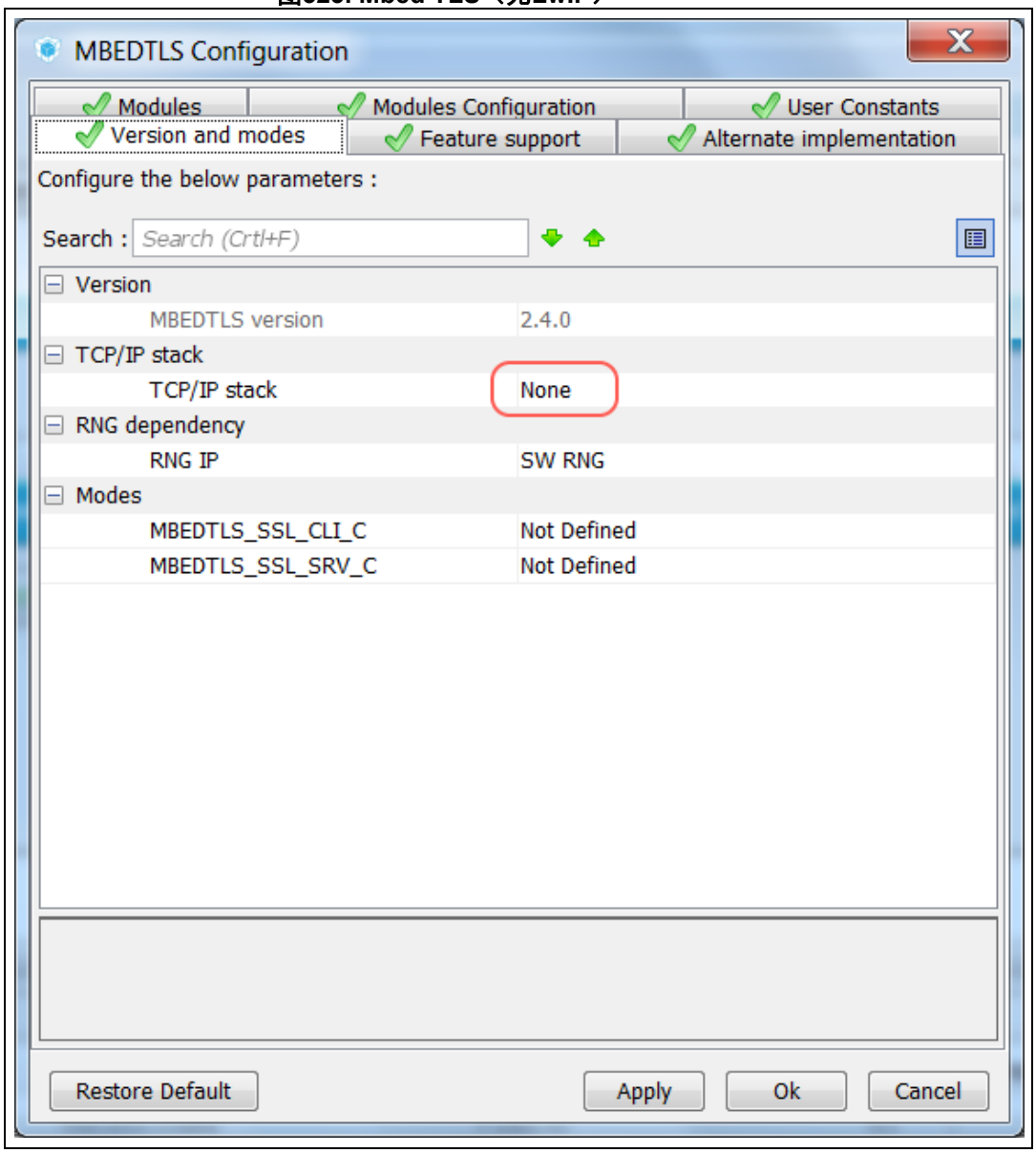


图324. Mbed TLS（有LwIP和FreeRTOS）

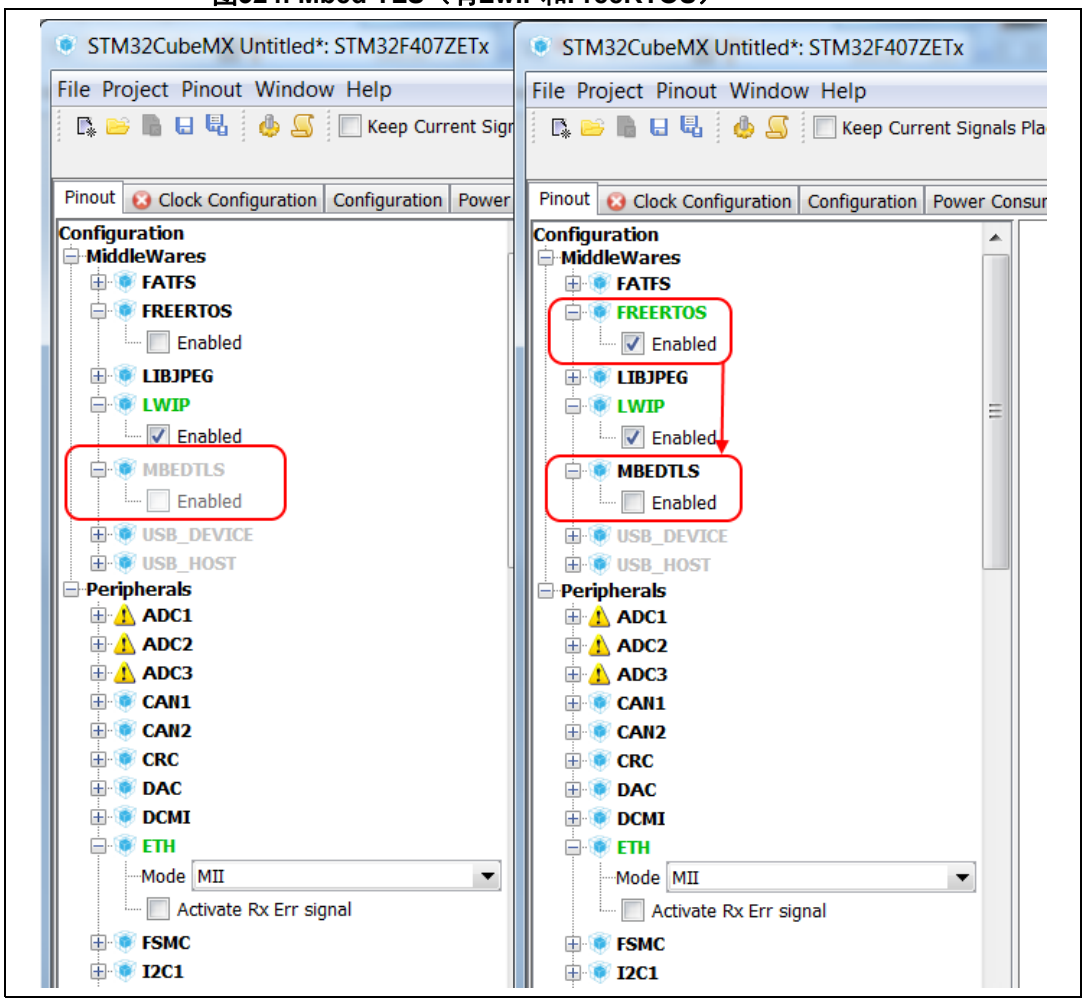
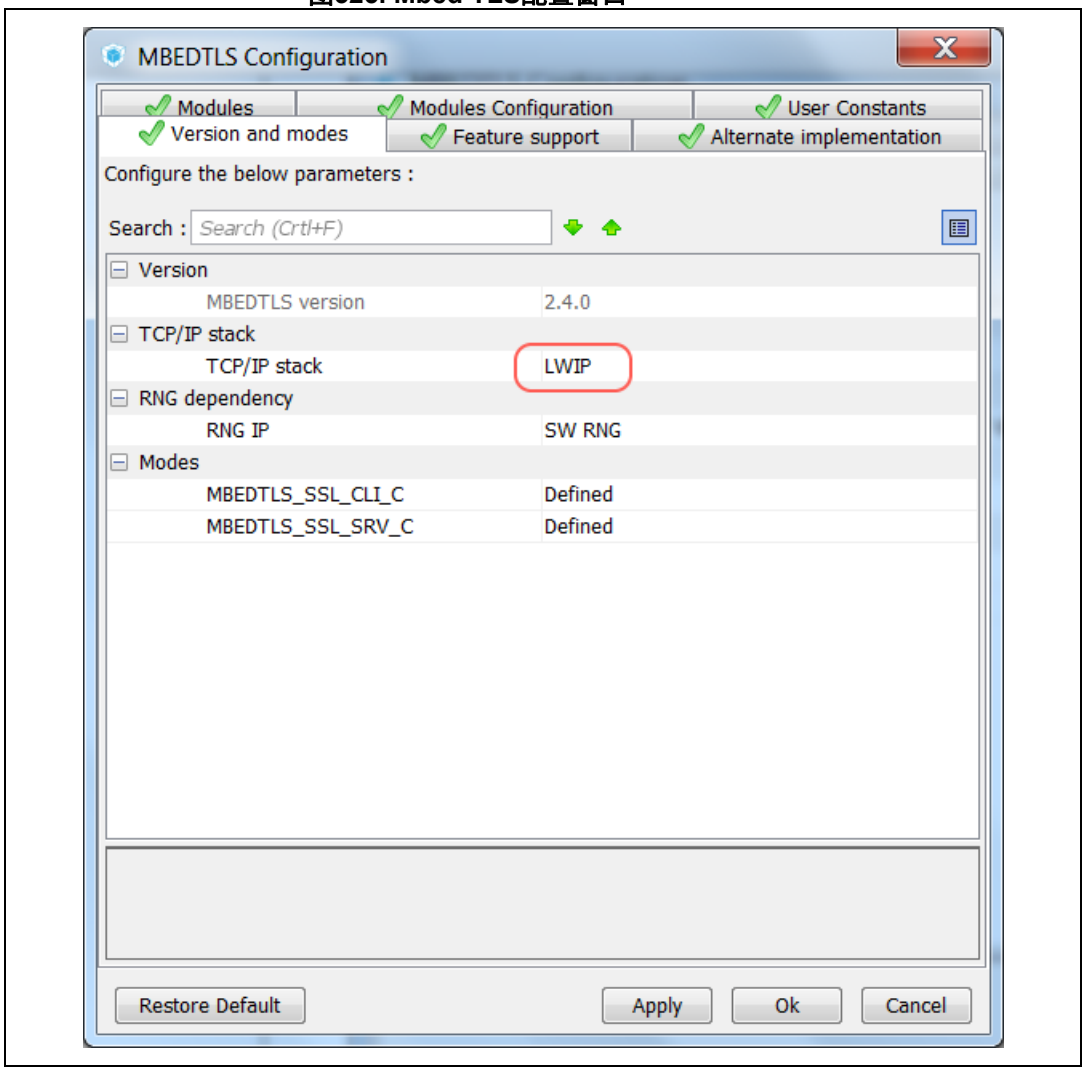


图325. Mbed TLS配置窗口



B.3.9 TouchSensing

STM32TouchSensing库属于C库，用于将传统的电动机械开关替换为STM32微控制器的电容传感器，从而可创建更高端的人机界面。

该库要求在微控制器上配置触摸感应外设。

STM32CubeMX生成以下文件，用户可通过STM32CubeMX用户界面（参见*图 326：启用TouchSensing外设*、*图 327：触摸感应传感器选择面板*和*图 328：TouchSensing配置面板*）和/或使用代码自身的用户部分修改这些文件的内容：

- *touchsensing.c/.h*
- *tsl_user.c/.h*
- *tsl_conf.h*

图326. 启用TouchSensing外设

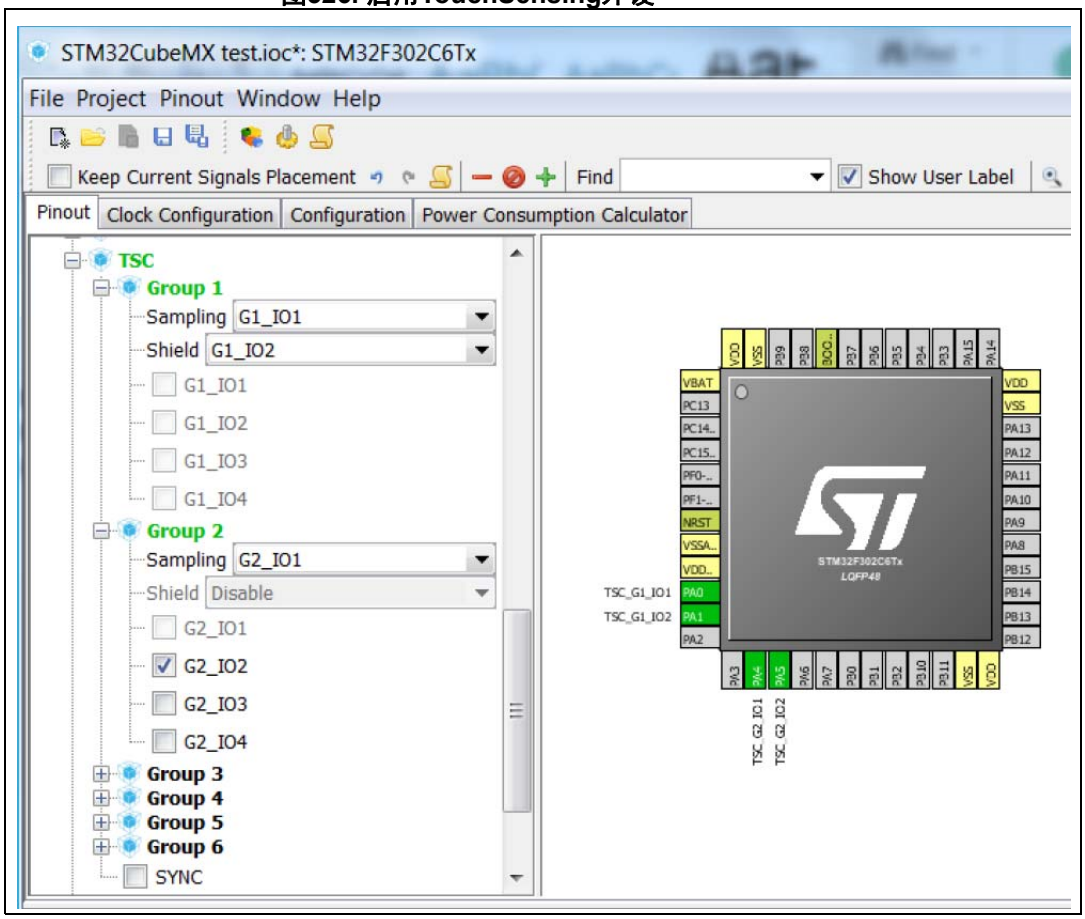


图327. 触摸感应传感器选择面板

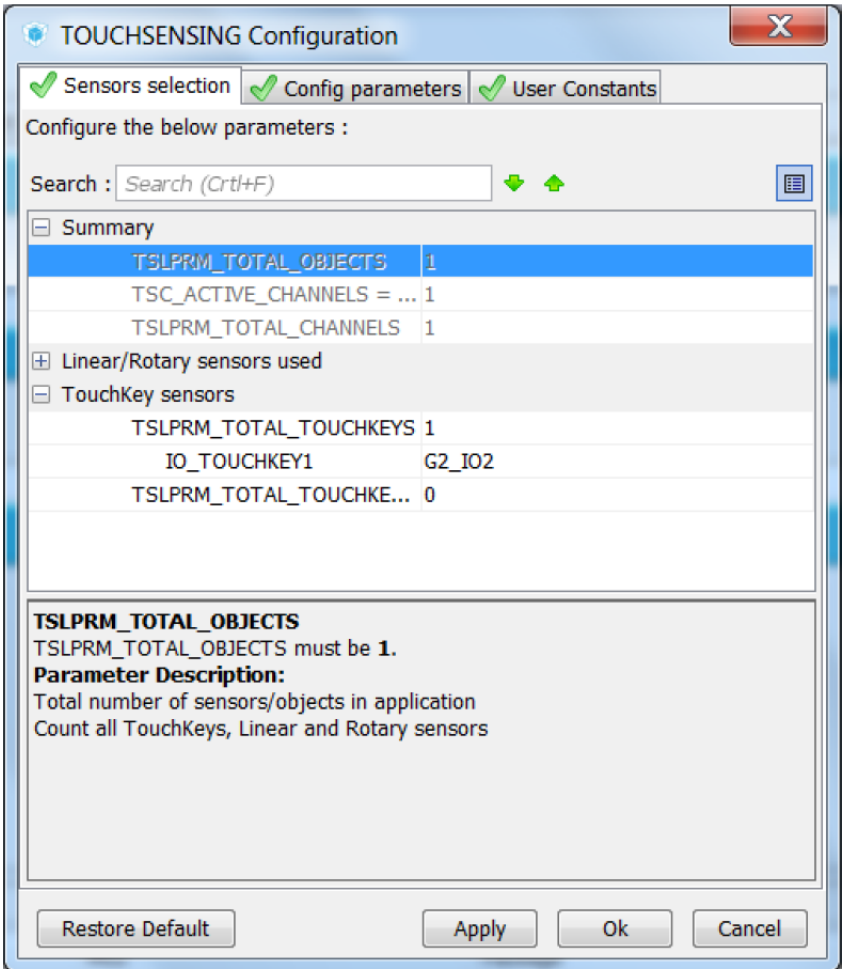
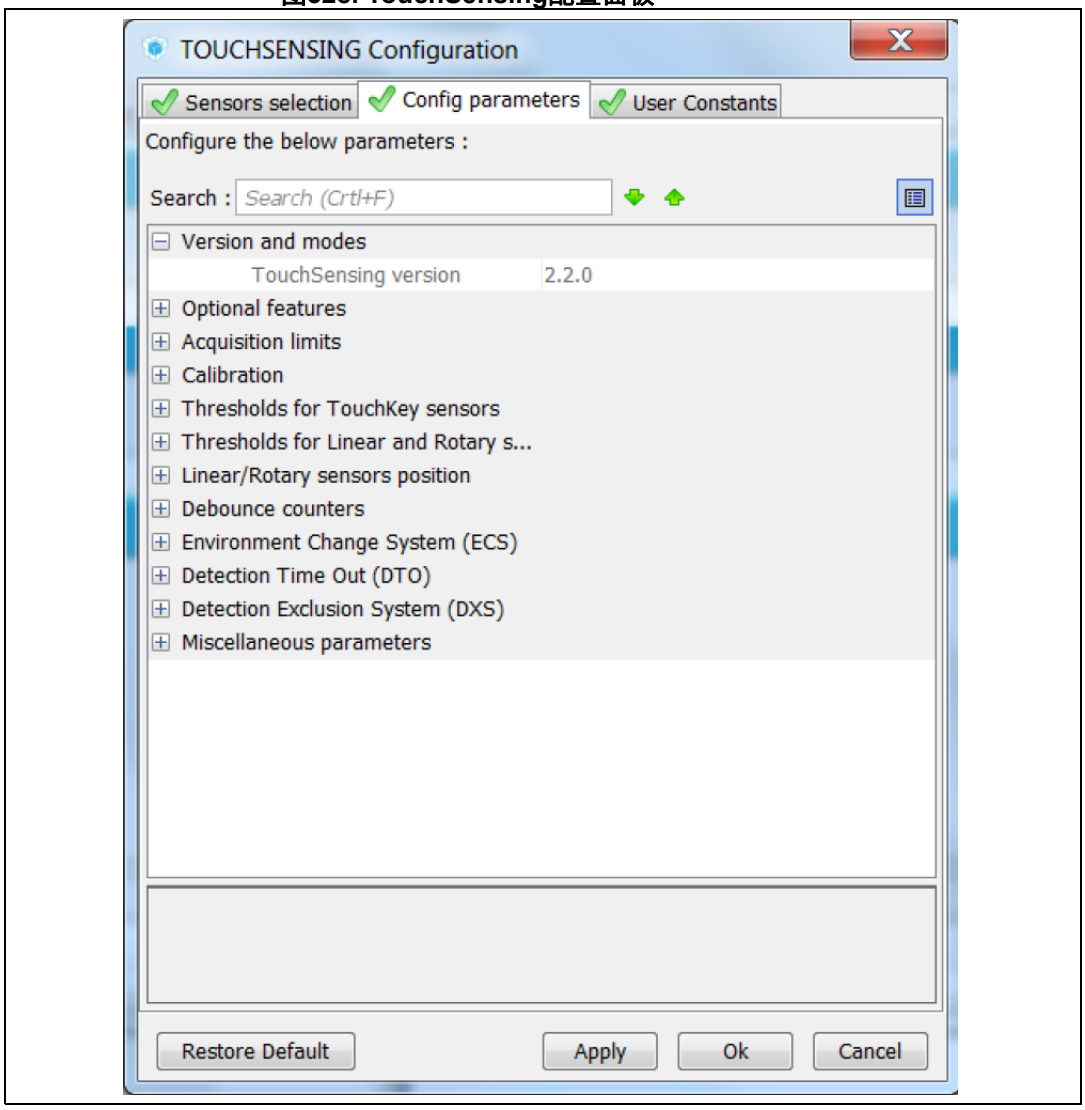


图328. TouchSensing配置面板



B.3.10 PDM2PCM

PDM2PCM库属于C库，用于将脉冲密度调制（PDM）数据输出转换为16位脉冲编码调制（PCM）格式。该库要求启用CRC外设。

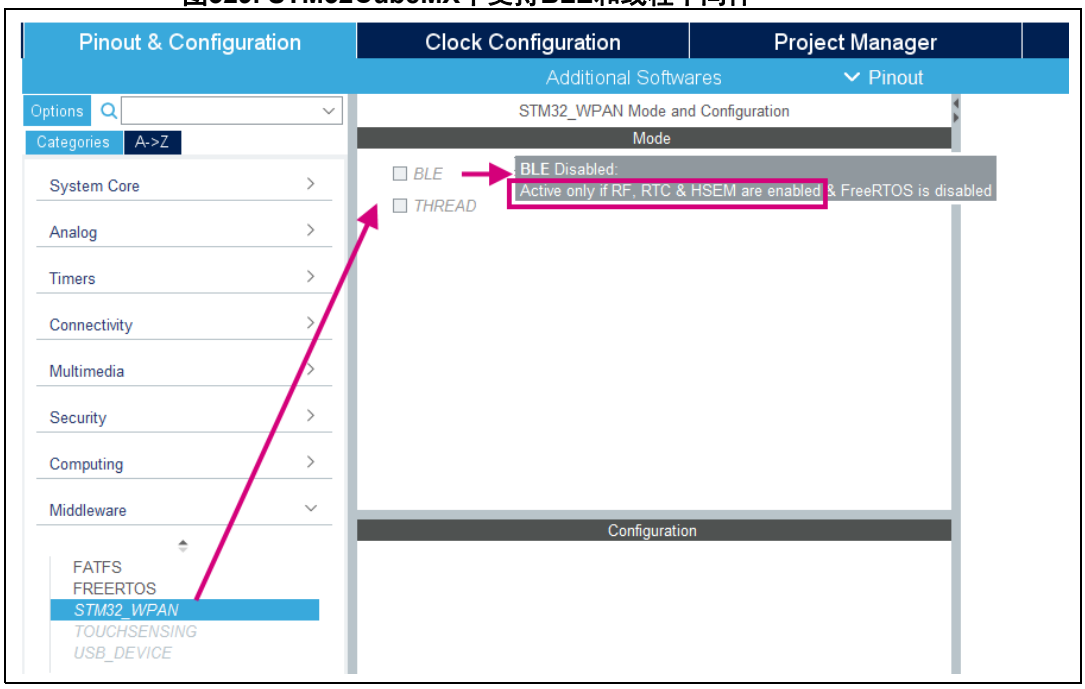
STM32CubeMX生成以下文件，用户可通过STM32CubeMX用户界面和/或使用代码自身的用户部分修改这些文件的内容：

- *pdm2pcm.h/.c*

B.3.11 STM32WPAN BLE/线程（仅适于STM32WB系列）

STM32CubeMX现在支持STM32WPAN BLE和线程中间件。

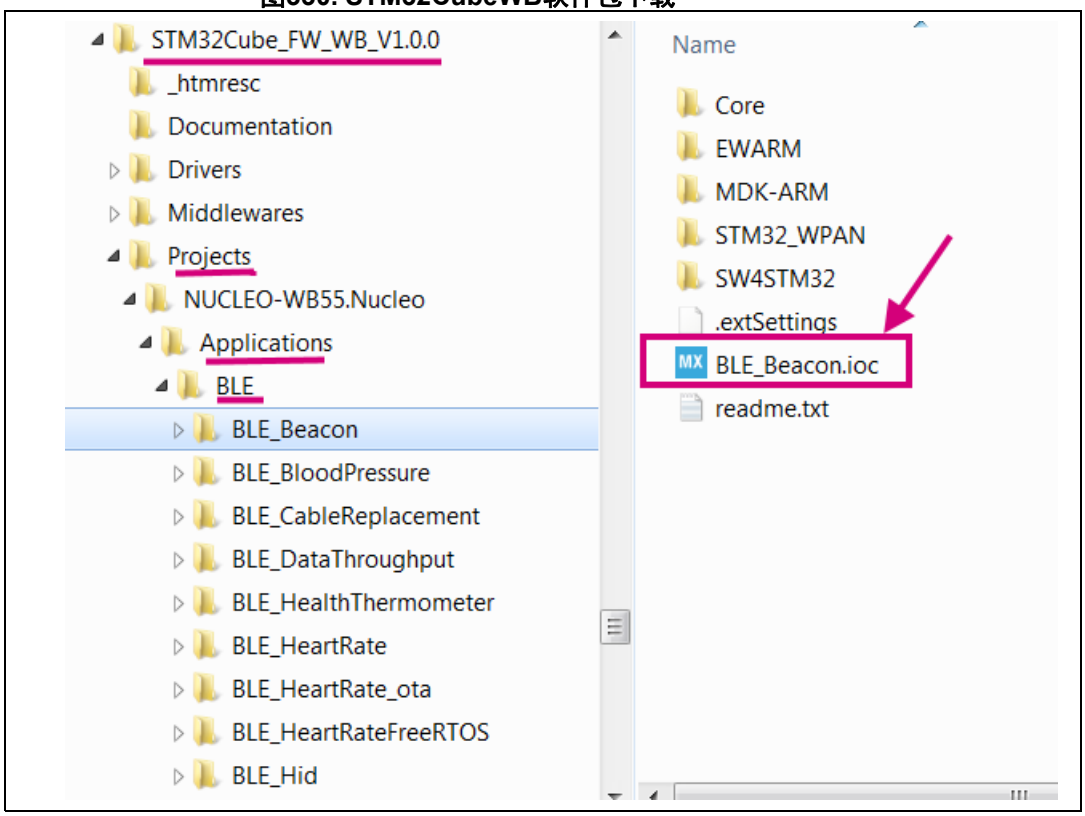
图329. STM32CubeMX中支持BLE和线程中间件



它们在给定的项目中是专有的，而且尚不支持具有FreeRTOS的配置。

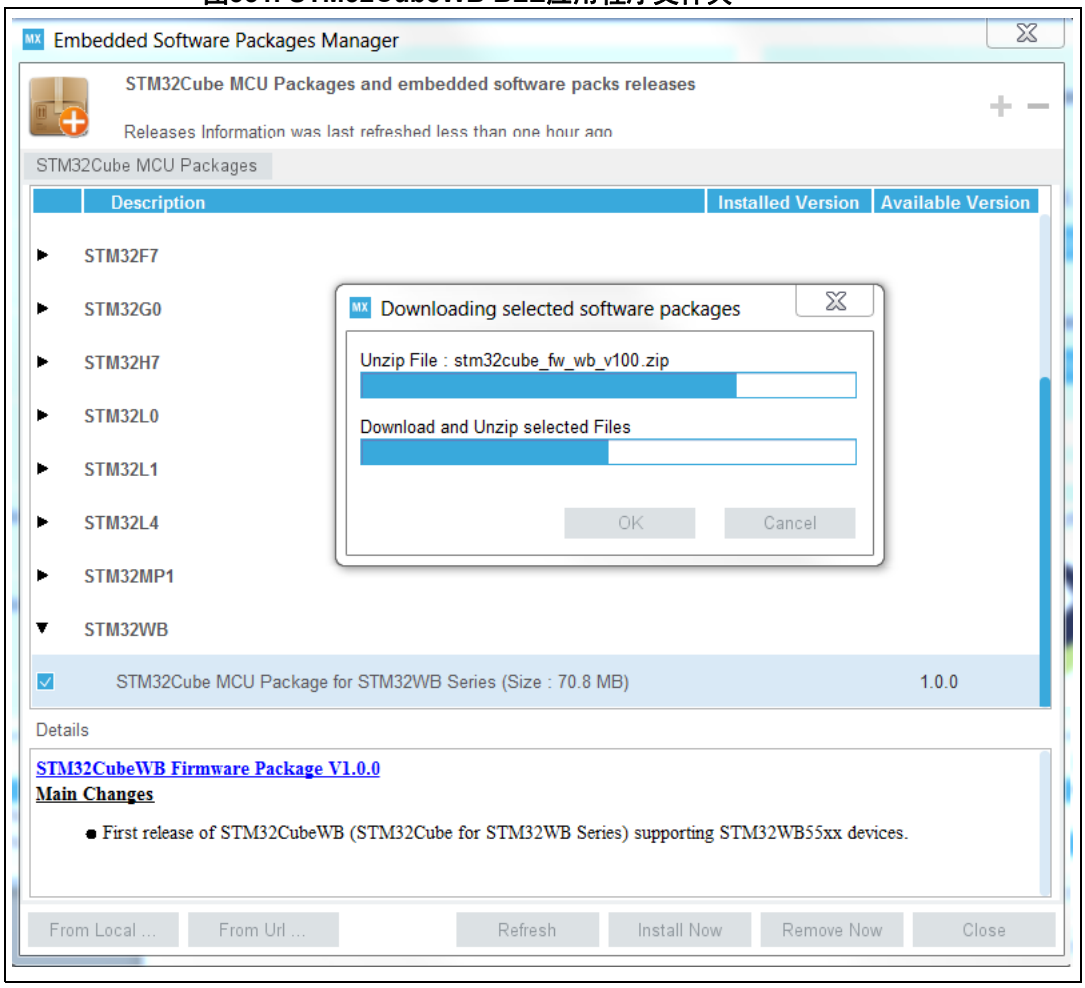
可以在STM32CubeWB MCU软件包的项目文件夹中找到用STM32CubeMX生成的应用程序项目。

图330. STM32CubeWB软件包下载



可以通过STM32CubeMX按照*第 3.4.2 节：安装STM32 MCU软件包*中所述的标准步骤安装该软件包。

图331. STM32CubeWB BLE应用程序文件夹



BLE 配置

要启用BLE，首先必须激活某些外设（RTC，HSEM，RF）。

然后，必须选择一种应用程序类型，可以是透明模式、服务器配置文件、路由器配置文件或客户端配置文件中的一种。

最后，必须配置与该应用程序类型相关的模式和其他参数。

注： *BLE透明模式和所有线程应用程序也需要配置USART或LPUART外设。*

图332. BLE服务器配置文件选择

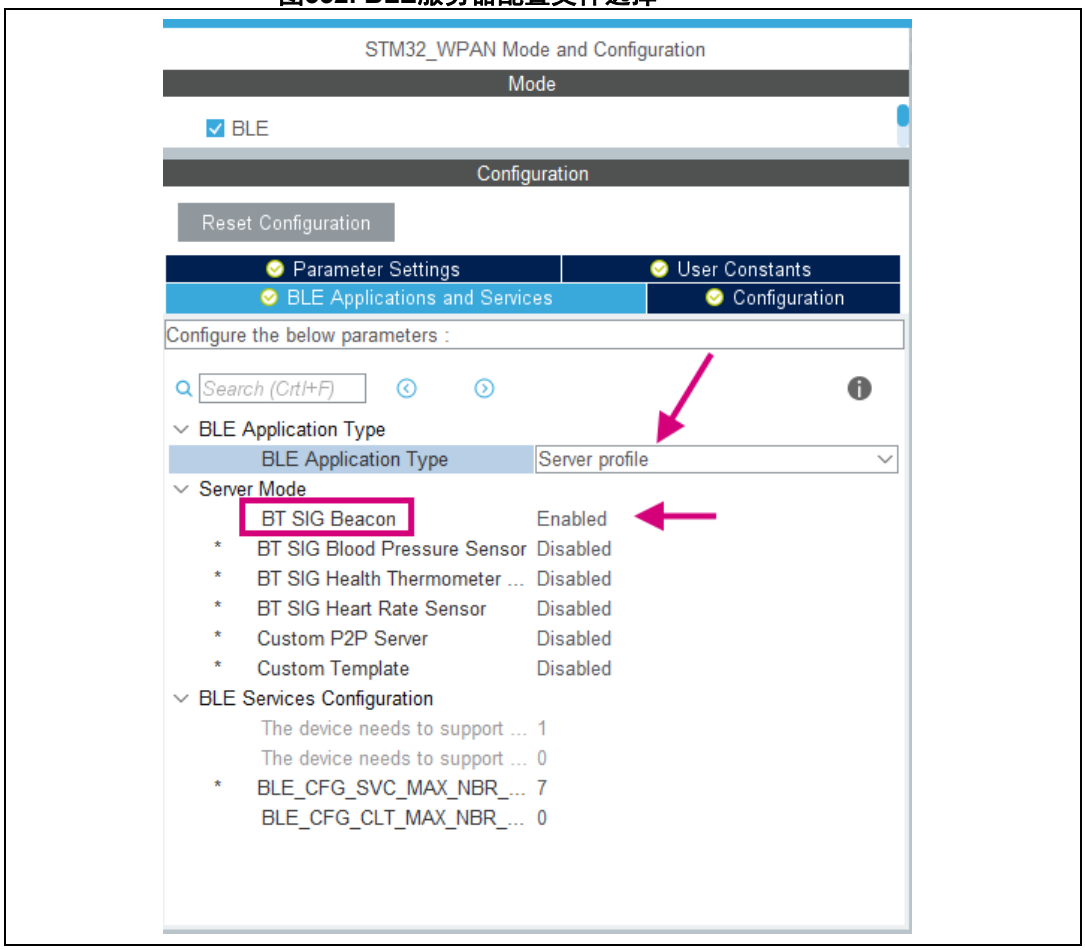
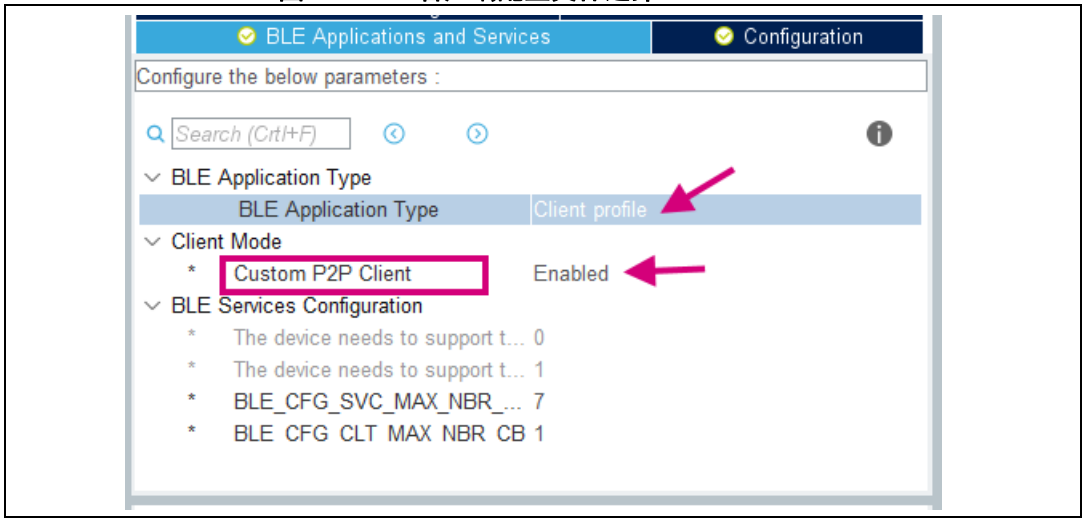


图333. BLE客户端配置文件选择

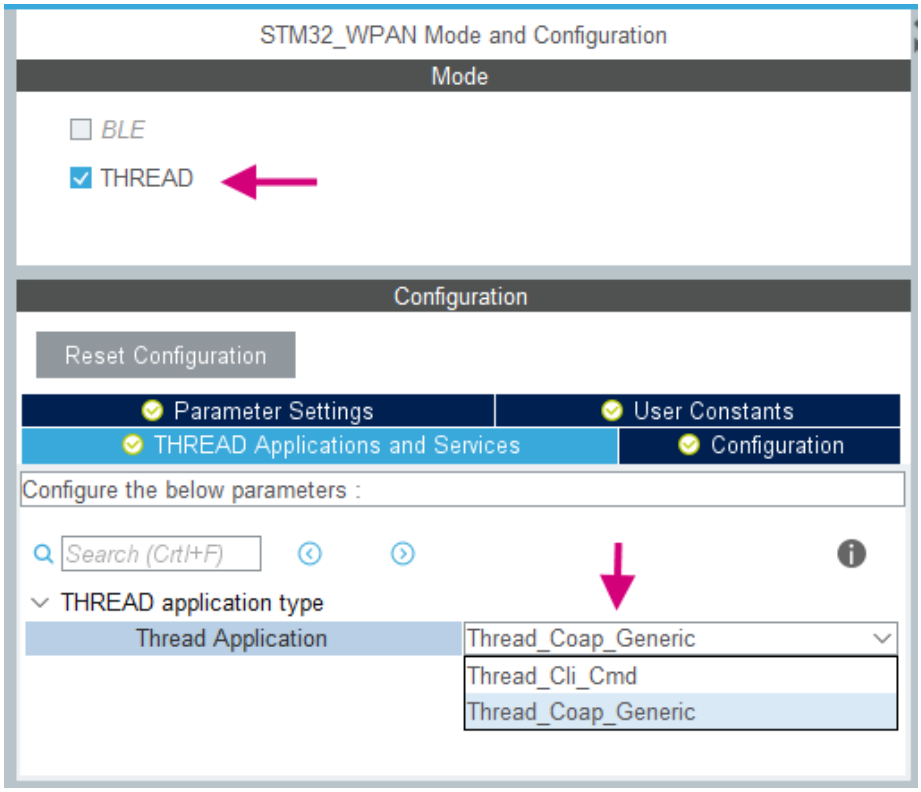


线程配置

要启用线程，首先必须激活某些外设（RTC，HSEM，RF）。

然后，必须选择应用程序类型并配置相关参数。

图334. 线程应用程序选择



B.3.12 OpenAmp和RESMGR_UTILITY（STM32MP1系列和STM32H7双核产品系列）

在双核产品上引入了新的软件和硬件，以实现多核协作。

- 仅适于STM32MP1系列：用于在两个处理器实例之间交换数据的处理器间通信控制器（IPCC）基于以下事实：共享存储器缓冲区分配在MCU SRAM中，并且每个处理器拥有特定的寄存器组和中断。
- 仅适用于STM32MP1系列：用于Cortex-A和Cortex-M内核之间互通的OpenAMP中间件实现RPMsg消息传递协议（请参阅*图 335*）。
- 用于系统资源管理的资源管理器库（RESMGR_UTILITY）：多处理器器件可以在多个内核上运行独立的固件（请参阅*图 336*）。这表示一个内核可以使用某些外设而无需了解这些外设的用法：资源管理库的作用是控制外设分配给专用内核，并提供一种配置用于操作该外设的系统资源的方法（请参阅*图 337*）。

图335. 为STM32MP1器件启用OpenAmp

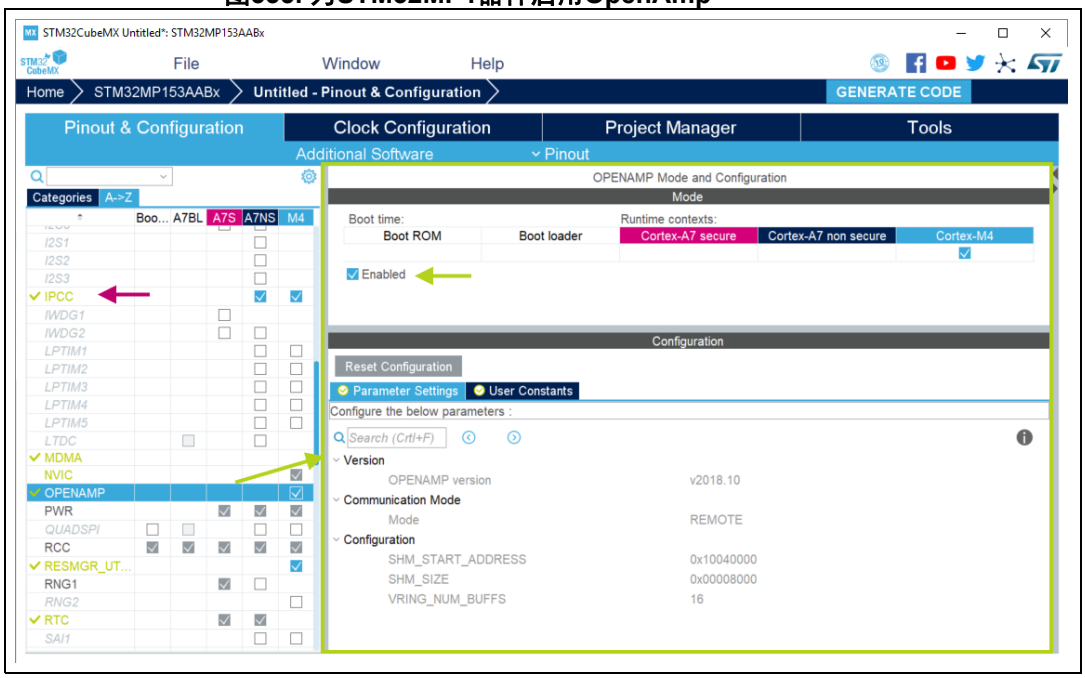


图336. 为STM32MP1器件启用资源管理器

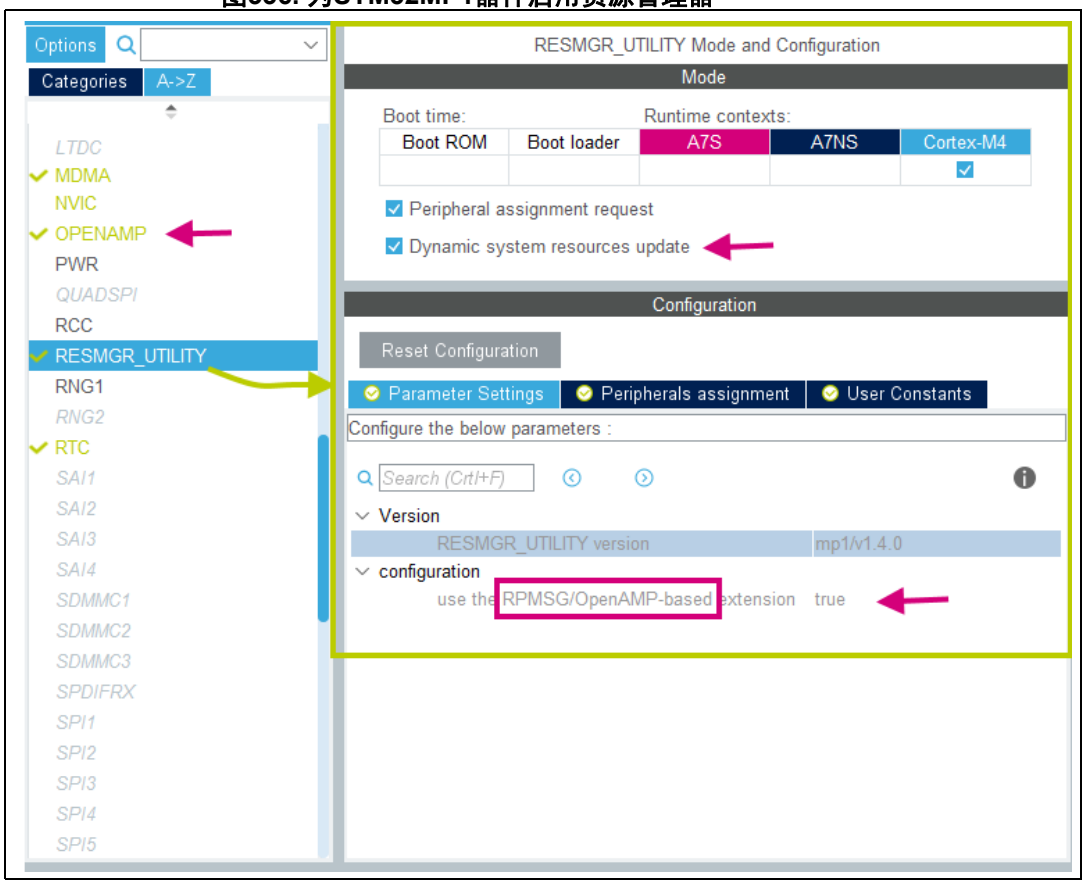
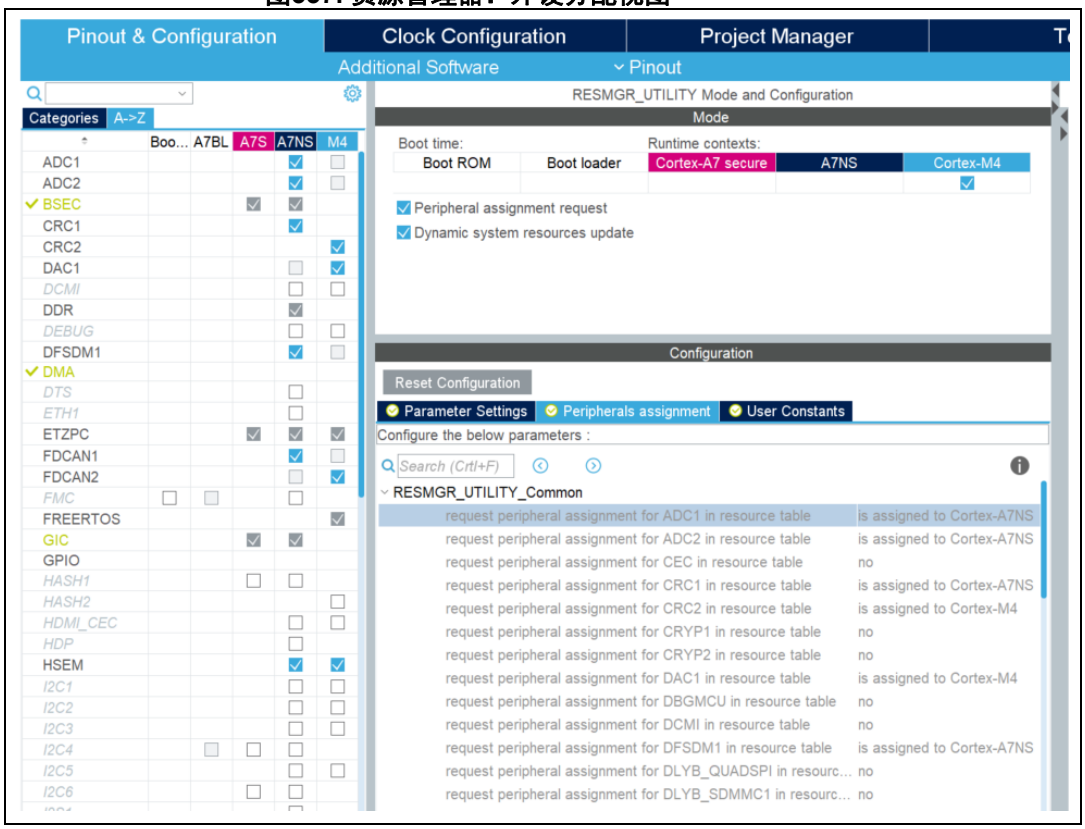


图337. 资源管理器：外设分配视图



有关更多详细信息，请访问STM32MP1专用Wiki网站（https://wiki.st.com/stm32mpu）。

附录C STM32微控制器命名规则

STM32微控制器产品编号按照以下命名规则编订：

- 器件子系列

  数字越大，提供的功能越多。

  例如，STM32L0系列包含STM32L051、L052、L053、L061、L062、L063子系列，其中，STM32L06x产品编号配有AES，而STM32L05x没有AES。

  最后一位数表示功能级别。上例中：

  – 1 = 基本型系列
  – 2 = 带USB
  – 3 = 带USB和LCD。

- 引脚数
  – F = 20个引脚
  – G = 28个引脚
  – K = 32个引脚
  – T = 36个引脚
  – S = 44个引脚
  – C = 48 个引脚
  – C = 64（或66）个引脚
  – M = 80个引脚
  – O = 90个引脚
  – V = 100 个引脚
  – Q = 132个引脚（例如 例如 STM32L162QDH6）
  – Z = 144 个引脚
  – I = 176（+25）个引脚
  – B = 208个引脚（例如 例如 STM32F429BIT6）
  – N = 216个引脚

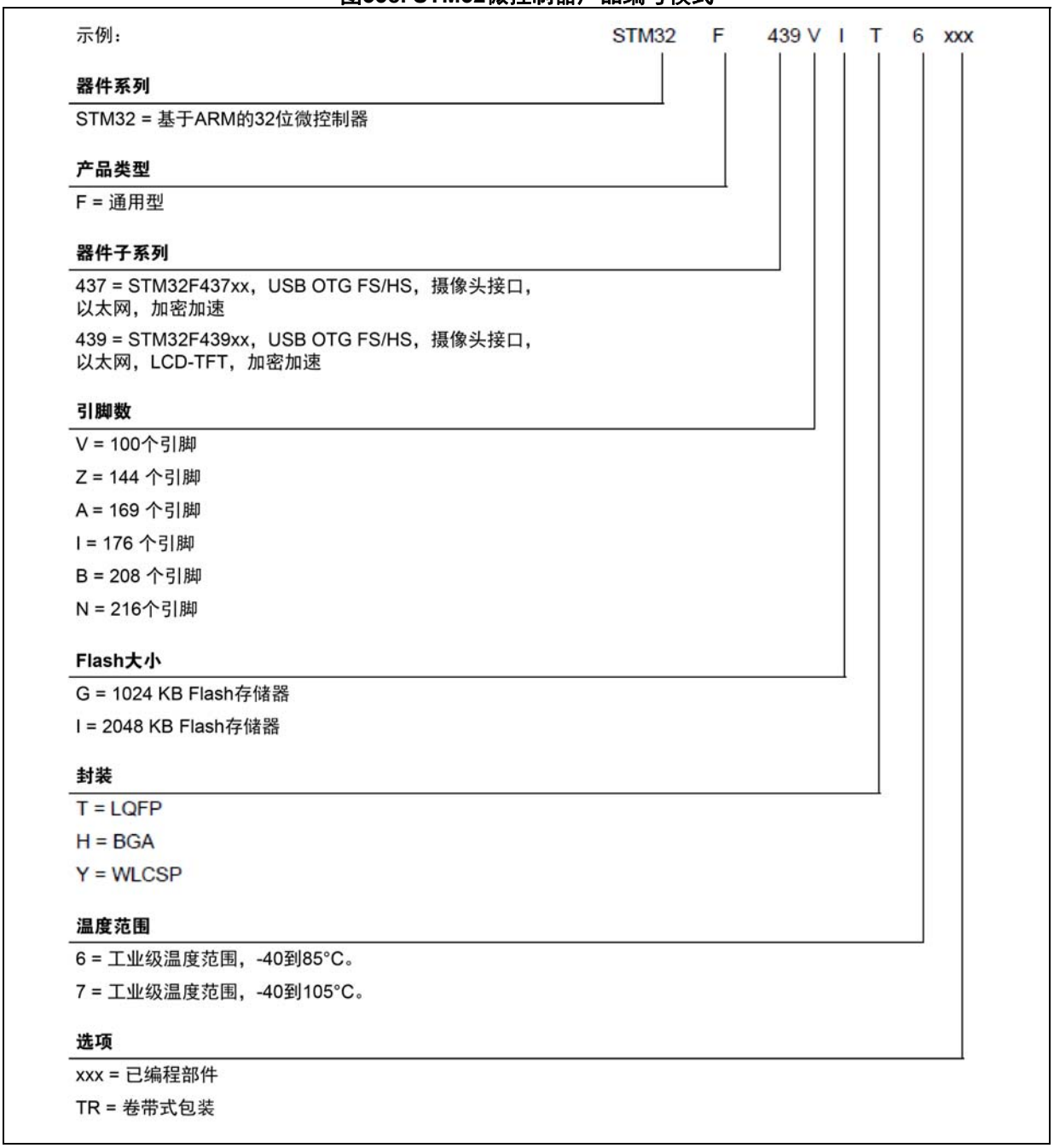
- Flash存储器大小
  – 4 = 16 KB Flash存储器
  – 6 = 32 KB Flash存储器
  – 8 = 64 KB Flash
  – B = 128 KB Flash
  – C = 256 KB Flash
  – D = 384 KB Flash存储器
  – E = 512 KB Flash
  – F = 768 KB Flash存储器
  – G = 1024 KB Flash存储器
  – I = 2048 KB Flash存储器

- 封装
  – B = SDIP
  – H = BGA

- M = SO
- P = TSSOP
- T = LQFP
- U = VFQFPN
- Y = WLCSP

图 338 举例介绍了STM32微控制器产品的编号模式。

图338. STM32微控制器产品编号模式



附录D STM32微控制器功耗参数

本节概括介绍了如何使用STM32CubeMX功耗计算器。

微控制器功耗取决于芯片规格、电源电压、时钟频率以及工作模式。嵌入式应用通过降低时钟频率（不需要进行快速处理的情况下）、选择最佳工作模式及运行电压范围的方式优化STM32 MCU功耗。下文介绍了STM32功耗模式和电压范围。

D.1 功耗模式

STM32 MCU支持不同的功耗模式（有关完整说明，请参见STM32 MCU数据手册）。

D.1.1 STM32L1系列

STM32L1微控制器共有6种功耗模式，包括5种低功耗模式：

- **运行**模式

  此模式可通过HSE/HSI时钟源提供最高性能。CPU的最高运行频率为32 MHz，并会启用调压器。

- **睡眠**模式

  该模式使用HSE或HSI作为系统时钟源。调压器已启用，CPU停止运行。所有外设继续运行并可在发生中断/事件时唤醒 CPU。

- **低功耗运行**模式

  该模式使用的多速内部(MSI) RC振荡器设为最低时钟频率（131 kHz），内部调压器处于低功耗模式。时钟频率和已启用外设数均受到限制。

- **低功耗睡眠**模式

  该模式通过进入睡眠模式实现。内部调压器处于低功耗模式。时钟频率和已启用外设数均受到限制。典型示例是以32 kHz频率运行的定时器。

  如果唤醒是通过事件或中断触发的，系统会恢复运行模式，且调压器会开启。

- **停机**模式

  该模式可实现最低功耗，同时会保留RAM和寄存器内容。时钟停止。可使用频率为32 kHz/37 kHz的LSE/LSI对实时时钟（RTC）进行备份。已启用外设数受到限制。调压器处于低功耗模式。

  可通过任意EXTI线将器件从停机模式唤醒。

- **待机**模式

  该模式可实现最低功耗。此时，内部调压器关闭，因此整个V_{CORE}域将断电。时钟停止，实时时钟（RTC）可通过频率为32 kHz/37 kHz的LSE/LSI进行保留。RAM和寄存器内容会丢失，但待机电路中的寄存器除外。对已启用外设数的限制要比停机模式下更严格。

器件在复位、三个WKUP引脚中有一个引脚出现上升沿、或发生RTC事件（若RTC启用）的情况下退出待机模式。

注： *退出停机或待机模式进入运行模式时，STM32L1 MCU会经历将MSI振荡器用作时钟源的状态。该转换对全局功耗有着重要影响。因此，功耗计算器引入两个转换步骤：**WU_FROM_STOP**和**WU_FROM_STANDBY**。在这两步中，时钟自动配置为MSI。*

D.1.2 STM32F4系列

STM32F4微控制器共有5种功耗模式，包括4种低功耗模式：

- **运行**模式

  该模式是开机或系统复位后的默认模式。此模式可通过HSE/HSI时钟源提供最高性能。CPU能够以最大频率运行，具体视所选功率级别而定。

- **睡眠**模式

  只有CPU停止运行。所有外设继续运行并可在发生中断/事件时唤醒CPU。时钟源为进入睡眠模式前所设置的时钟。

- **停止**模式

  此模式以RC振荡器作为时钟源，可实现极低功耗。1.2 V域中的所有时钟都会停止，CPU和外设也会停止运行。PLL、HSI RC和HSE晶振被禁用。寄存器和内部SRAM的内容会保留下来。

  可以将调压器置于主调压器模式（MR）或低功耗调压器模式（LPR）。选择处于低功耗调压器模式的调压器将延长唤醒时间。

  可使Flash存储器进入停机模式，以实现快速唤醒，也可使其进入深度断电模式，以较慢的唤醒时间实现较低功耗。

  停机模式有两种子模式：

  – **在正常模式下停机（默认模式）**

    在该模式下，1.2 V域保持在标称漏电流模式下，最低V12电压为1.08 V。

  – **欠载模式下停机**

    在该模式下，1.2 V域保持在减小漏电流模式下，V12电压低于1.08 V。调压器（主模式或低功耗模式）处于欠载或低压模式。Flash存储器必须处于深度断电模式。唤醒时间比正常模式下长100 µs左右。

- **待机**模式

  停机模式以RC振荡器作为时钟源，可以实现极低的功耗。内部调压器关闭，因此整个1.2 V域将断电。CPU和外设停止运行。PLL、HSIRC和HSE晶振被禁用。除选择的备份域和4字节备份SRAM中的寄存器外，SRAM和寄存器的内容都将丢失。只有RTC和LSE振荡器块上电。发生外部复位（NRST引脚）、IWDG复位、WKUP引脚上出现上升沿或者触发RTC报警/唤醒/篡改/时间戳事件时，器件退出待机模式。

- **V_{BAT}操作**
  与待机模式相比，此模式可显著减小功耗。仅当为备份域供电的V_{BAT}引脚连接到由电池或其他电源供电的可选待机电压时，该模式才可用。V_{BAT}域会保留（RTC寄存器、RTC备份寄存器和备份SRAM），RTC和LSE振荡器块会上电。与待机模式的主要区别在于外部中断和RTC报警/时间不会使器件退出V_{BAT}操作。增大V_{DD}达到最小阈值。

D.1.3 STM32L0系列

STM32L0微控制器共有8种功耗模式，包括7种低功耗模式，可在低功耗、短启动时间和可用唤醒源之间获得最佳平衡：

- **运行模式**
  此模式可通过HSE/HSI时钟源提供最高性能。CPU的最高运行频率为32MHz，并会启用调压器。
- **睡眠模式**
  该模式使用HSE或HSI作为系统时钟源。调压器已启用，只有CPU停止运行。所有外设继续运行并可在发生中断/事件时唤醒 CPU。
- **低功耗运行模式**
  此模式在低功耗模式下使用内部调压器，多速内部（MSI）RC振荡器设为最小时钟频率（131 kHz）。在低功耗运行模式下，时钟频率和已启用外设数均受到限制。
- **低功耗睡眠模式**
  要实现此模式，可在内部调压器处于低功耗模式的情况下进入睡眠模式。时钟频率和已启用外设数均受到限制。事件或中断可使系统恢复为运行模式（调压器开启）。
- **停机模式（使用RTC）**
  停机模式下可以实现最低功耗，同时可保留RAM寄存器内容和实时时钟。调压器处于低功耗模式。LSE或LSI仍在运行中。此时，$_{CORE}$域中的所有时钟都会停止，PLL、MSI RC、HSE晶振和HSI RC振荡器也被禁止。
  一些具有唤醒功能的外设可在停机模式期间启用HSI RC来检测其唤醒条件。可通过任一EXTI线在3.5 µs内将器件从停机模式唤醒，处理器可用作中断或恢复代码。
- **停机模式（不使用RTC）**
  该模式与“停机模式（使用RTC）”相同，唯一不同的是在此处停止的RTC时钟。
- **待机模式（使用RTC）**
  在实时时钟运行的情况下，待机模式可实现最低功耗。此时，内部调压器关闭，因此整个V_{CORE}域将断电。PLL、MSI RC、HSE晶振和HSI RC振荡器也会关闭。LSE或LSI仍在运行中。

进入待机模式后，RAM和寄存器内容会丢失，但待机电路中的寄存器内容除外（唤醒逻辑、IWDG、RTC、LSI、LSE晶体32 KHz振荡器、RCC_CSR寄存器）。

发生外部复位（NRST引脚）、IWDG复位、三个WKUP引脚中的一个引脚上出现上升沿或者触发RTC报警（报警A或报警B）、RTC篡改事件、RTC时间戳事件或RTC唤醒事件时，器件会在60 µs内退出待机模式。

发生RTC篡改事件、RTC时间戳事件或RTC唤醒事件。

- **待机模式（不使用RTC）**

  此模式与待机模式（使用RTC）完全相同，唯一不同的是RTC、LSE和LSI时钟会停止。

  发生外部复位（NRST引脚）或三个WKUP引脚中的一个引脚上出现上升沿时，器件会在60 µs内退出待机模式。

注： *进入停机或待机模式时，RTC、IWDG和相应的时钟源不会自动停止。LCD进入停机模式时不会自动停机。*

D.2 功耗范围

利用动态电压调节功能，可对STM32 MCU的功耗进行进一步调整：为逻辑（CPU、数字外设、SRAM和Flash存储器）供电的内部主调压器输出电压V12可通过在软件中选择功率范围（STM32L1和STM32L0）或功率级别（STM32 F4）的方式进行调整。

提供的功耗范围定义如下（有关完整的详细信息，请参考STM32 MCU数据手册）。

D.2.1 STM32L1系列有三种V_{CORE}范围

- 高性能**范围1**（V_{DD}范围限制为2.0-3.6 V），CPU最高运行频率为32 MHz

  只要V_{DD}输入电压超过2.0 V，调压器就会输出1.8 V电压（典型值）。可执行Flash编程和擦除操作。
- 中等性能**范围2**（整个V_{DD}范围），CPU最大频率为16 MHz

  在1.5 V下，Flash应可工作，但其读取访问时间适中。仍可执行Flash编程和擦除操作。
- 低性能**范围3**（整个V_{DD}范围），CPU最大频率范围限制为4 MHz（仅通过多速内部RC振荡器时钟源生成）

  在1.2 V下，Flash存储器仍可工作，但其读取访问时间较长。不可执行Flash编程和擦除操作。

D.2.2 STM32F4系列有多种V_{CORE}级别

仅当PLL关闭且选择HSI或HSE作为系统时钟源时，才能修改级别。

- **级别1**（V12电压范围限制为1.26 - 1.40 V），此模式为复位后的默认模式。
  HCLK频率范围 = 144 MHz到168 MHz（180 MHz时过载）。
  此模式为复位后的默认模式。
- **级别2**（V12电压范围限制为1.20 - 1.32 V）。
  HCLK频率范围最大为144 MHz（168 MHz时过载）。
- **级别3**（V12电压范围限制为1.08 - 1.20 V），此模式为退出停机模式后的默认模式。
  HCLK频率 ≤120 MHz。

电压缩放如下调整为f_{HCLK}频率：

- **STM32F429x/39x MCU**：
  - **级别1**：最大168 MHz（达到180 MHz时过载）
  - **级别2**：120到144 MHz（达到168 MHz时过载）
  - **级别3**：最大120 MHz。
- **STM32F401x MCU**：
  非级别1
  - **级别2**：60到84 MHz
  - **级别3**：最大60 MHz。
- **STM32F40x/41x MCU**：
  - **级别1**：最大168 MHz
  - **级别2**：最大144 MHz

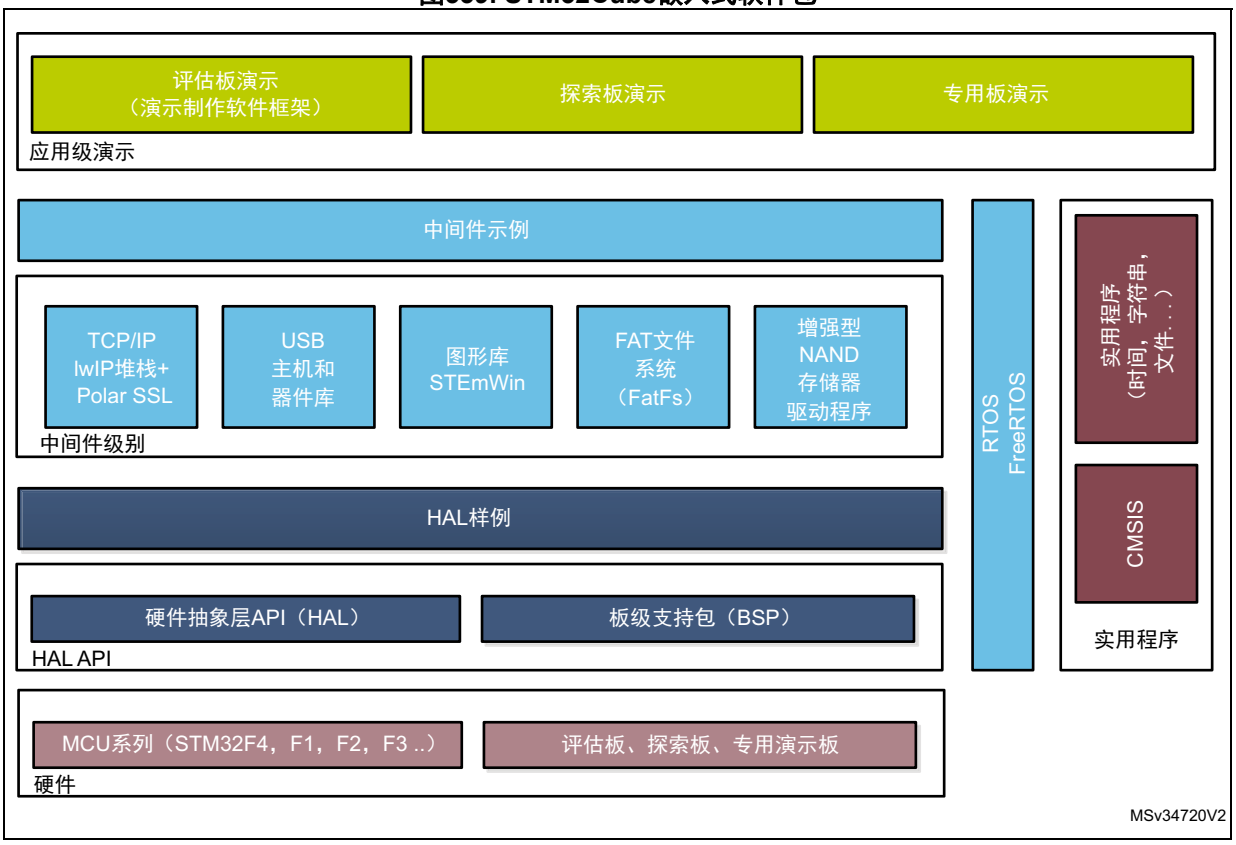
D.2.3 STM32L0系列有三种V_{CORE}范围

- 范围1（V_{DD}范围限制为1.71到3.6 V），CPU最高运行频率为32 MHz
- 范围2（整个V_{DD}范围），CPU最大频率为16 MHz
- 范围3（整个V_{DD}范围），CPU最大频率限制为4.2 MHz。

附录E STM32Cube嵌入式软件包

嵌入式软件包与STM32CubeMX C代码生成器均属于STM32Cube产品的一部分（参考*DB2164简明数据手册*）：这些软件包包括涵盖微控制器硬件的低级硬件抽象层（HAL），以及大量在STMicroelectronics板上运行的示例集（参见*图 339*）。这些组件可在STM32系列之间实现高度可移植性。软件包完全兼容STM32CubeMX生成的C代码。

图339. STM32Cube嵌入式软件包



注： *STM32CubeF0、STM32CubeF1、STM32CubeF2、STM32CubeF3、STM32CubeF4、STM32CubeL0和STM32CubeL1嵌入式软件包在st.com上提供。这些软件包基于STM32Cube版本v1.1（其他系列将逐步引入），并包含STM32CubeMX用于生成初始化C代码的嵌入式软件库。*

用户应使用STM32CubeMX生成初始化C代码，软件包中提供的示例可用于入门级STM32应用开发。

19 版本历史

表24. 文档版本历史

日期	版本	STM32CubeMX 版本号	变更
2014年2月17日	1	4.1	初始版本。
2014年4月4日	2	4.2	在封面、*第 2.2 节：主要特性*、*“第5.14.1 节：外设和中间件配置窗口”*、以及*附录ESTM32Cube嵌入式软件包*中，增加了对STM32CubeF2和STM32F2系列的支持。 更新了*第 11.1 节：创建一个新STM32CubeMX项目*、*第 11.2 节：配置MCU引脚排列*、*第 11.6 节：配置MCU初始化参数*。 “生成GPIO初始化C代码”部分移至*第8 节：教程3- 生成GPIO初始化C代码（仅限STM32F1系列）*并更新了内容。 增加了*第 18.4 节：安装“Java 7更新45” 或更新版的JRE时，为何会出现“Java 7更新45”错误？*。
2014年4月24日	3	4.3	在封面、*第 2.2 节：主要特性*、*第 2.3 节：规则和限制*和“第5.14.1节：外设和中间件配置窗口”中增加了对STM32CubeL0和STM32L0系列的支持 在*“表13：文件菜单功能”*、*“第5.7.3 节：引脚布局菜单和”*和*第 4.2 节：新项目窗口*中增加了板选择。更新了*表15：引脚布局菜单*。 更新了*图 125：功耗计算器默认视图*，并在*第 5.1.1 节：建立功耗系列*中增加了电池选择。 更新了*第 5.1 节：功耗计算器视图* 中的注释 更新了*第 11.1 节：创建一个新STM32CubeMX项目*。 增加了*第 18.5 节：为何RTC复用器在时钟树视图中仍无效？*，*第 18.6 节：如何选择LSE和HSE作为时钟源并更改频率？*，以及*第 18.7 节：在PC13、PC14、PC15和PI8之一已配置为输出的情况下，为什么STM32CubeMX不允许我 将其配置为输出？*。

表24. 文档版本历史

日期	版本	STM32CubeMX 版本号	变更
2014年6月19日	4	4.4	在封面、*第 2.2 节：主要特性*、*第 2.3 节：规则和限制*中增加了对STM32CubeF0、STM32CubeF3、STM32F0和STM32F3系列的支持。 在*第 2.2 节：主要特性*、*表 2：主页快捷键*、*第 4.2 节：新项目窗口*、*“第5.7 节：工具栏和菜单”*、*第 4.11 节：“设置未使用 / 重置已使用GPIO”窗口*、*第 4.9 节：“项目管理器”视图*以及*“第5.15 节：引脚布局视图”*中增加了板选择功能和引脚锁定功能。增加了*“第5.15.1 节：在引脚上锁定和标记信号”*。 更新了*“第5.16 节：配置视图”*和*第 4.8 节：“时钟配置”视图*以及*第 5.1 节：功耗计算器视图*。 更新了*图 37：选择MCU时显示的STM32CubeMX主窗口*、*图 99：“项目设置”窗口*、*图 124：“关于”窗口*、*“图140：STM32CubeMX引脚布局视图”*、*“图120：芯片视图”*、*图 125：功耗计算器默认视图*、*图 126：电池选择*、*“图87：构建功耗序列”图 128：功耗系列：新步骤默认视图*、*图 135：构建序列后的功耗计算器视图*、*图 136：序列表管理功能*、*“图88：PCC编辑步骤窗口”*、*“图83：功耗序列：已配置新步骤（STM32F4示例）”*、*图 133：在引脚排列视图中选择的ADC*、*图 134：功耗计算器步骤配置窗口：使用导入引脚排列使能ADC*、*图 138：结果区域描述*、*“图100：外设功耗工具提示”*、*图 254：功耗计算示例*、*“图155：序列表”*和*“图156：功耗计算结果”*。 更新了*“图142：STM32CubeMX配置视图”*和*“图39：STM32CubeMX配置视图”*中的STM32F1系列标题。 在*第 5.1 节：功耗计算器视图*中增加了 STM32L1。 将图*使用PCC面板添加新步骤*从*第8.1.1 节：添加一个步骤*中删除。 将图*向序列添加新步骤*从*第 5.1.2 节：配置功耗系列中的步骤*中删除。 更新了*第8.2 节：检查结果*。 更新了附录*B.3.4FatFs*和*附录DSTM32微控制器功耗参数*。增加了附录*D.1.3STM32L0系列*和*D.2.3STM32L0系列有三种VCORE范围*。

表24. 文档版本历史

日期	版本	STM32CubeMX 版本号	变更
2014年9月19日	5	4.5	在封面*第 2.2 节：主要特性*、*第 2.3 节：规则和限制*中增加了对STM32CubeL1的支持。 更新了*第 3.2.3 节：卸载STM32CubeMX独立版本*。 在*第 3.4 节：使用STM32CubeMX进行更新*中增加了离线更新，修改了*图 8：“嵌入式软件包管理器”窗口*和*第 3.4.2 节：安装STM32 MCU软件包*。 更新了*第 4 节：STM32CubeMX用户界面*前言、*表 2：主页快捷键*和*第 4.2 节：新项目窗口*。 增加了*图 31：新项目窗口 - 板选择器*。 更新了*图 107：项目设置代码生成器*。 修改了*第 4.9 节：“项目管理器”视图*中的步骤3。 更新了*图39：STM32CubeMX配置视图 - STM32F1系列*。 在*“第5.14.1 节：外设和中间件配置窗口”*中增加了STM32L1。 更新了*图 61：“GPIO配置”窗口 - GPIO选择*、*第 4.4.12 节：“GPIO配置”窗口*和*图 66：DMA MemToMem配置*。 更新了*第 4.8 节：“时钟配置”视图*的前言。更新了*第 4.8.1 节：时钟树配置功能*和*第 4.8.3 节：建议*、*第 5.1 节：功耗计算器视图*、*图 128：功耗系列：新步骤默认视图*、*图 135：构建序列后的功耗计算器视图*、*图83：功耗系列：配置新步骤（STM32F4示例）*和*图 134：功耗计算器步骤配置窗口： 使用导入引脚排列使能ADC*。 增加了*图 137：功耗：外设功耗图*并更新了*“图100：外设功耗工具提示”*。更新了*第 5.1.4 节：功耗系列步骤参数词汇表*。 更新了*第 6 节：STM32CubeMXC代码生成概述*。 更新了*第 11.1 节：创建一个新STM32CubeMX项目*和*第 11.2 节：配置MCU引脚排列*。 增加了*第 12 节：教程2 - 使用STM32429I-EVAL评估板的SD卡上的FatFs示例 STM32429I-EVAL 评估板*，并更新了*第8 节：教程3- 生成GPIO初始化C代码（仅限STM32F1系列）*。 更新了*第 5.1.2 节：配置功耗系列中的步骤*。

表24. 文档版本历史

日期	版本	STM32CubeMX版本号	变更
2015年1月19日	6	4.6	完整项目生成、功耗计算和时钟树配置现在可用于所有STM32系列。 更新了*第 2.2节：主要特性*和*第 2.3节：规则和限制*。 更新了*第 3.1.3节：软件要求*中的Eclipse IDE。 更新了*图 6：“更新程序设置”窗口*、*图 8：“嵌入式软件包管理器”窗口*和*图 31：新项目窗口 - 板选择器*，更新了*第 4.9节：“项目管理器”视图*和*第 4.12节：“更新管理器”窗口*。 更新了*图 124：“关于”窗口*。 删除了图 *STM32CubeMX配置视图 - STM32F1系列*。 更新了*“表17：STM32CubeMX芯片视图”*中的图标和配色方案。 更新了*“第5.14.1节：外设和中间件配置窗口”*。 更新了*图 64：添加新的DMA请求*和*图 66：DMA MemToMem配置*。 更新了*第 4.8.1节：时钟树配置功能*。 更新了*图 126：电池选择*、*“图87：构建功耗序列”*、*“图88：PCC编辑步骤窗口”*。 增加了*第 6.3节：自定义代码生成*。 更新了*图 208：时钟树视图*和*图 213：“引脚布局和配置”视图*。 更新了外设配置顺序和 *第 11.6.2节：配置外设*中的*图 215：定时器3配置窗口*。 删除了教程3：生成GPIO初始化C代码（仅限STM32F1系列）。 更新了*图 219：GPIO模式配置*。 更新了*图 254：功耗计算示例*和*图155：序列表*。 更新了附录*A.1块一致性*、*A.2块相关性*和*A.3一个块 = 一种外设模式*。 附录*A.4块重新映射（仅限STM32F10x）*：更新了*第 节：示例*。 附录*A.6块转移（仅适用于STM32F10x，且“保留当前信号放置位置”已取消选中）*：更新了*示例* 更新了附录*A.8分别映射功能*。 更新了附录*B.3.1概述*。 更新了附录*D.1.3STM32L0系列*。

表24. 文档版本历史

日期	版本	STM32CubeMX版本号	变更
2015年3月19日	7	4.7	*第 2.2节：主要特性*：删除了*STM32F1系列的引脚排列初始化C代码生成*；更新了*完整的项目生成*。 更新了*图 8：“嵌入式软件包管理器”窗口*和*图 31：新项目窗口 - 板选择器*。 更新了*第 4.9节：“项目管理器”视图*中的IDE列表，并修改了*图 99：“项目设置”窗口*。 更新了*第 4.8.1节：时钟树配置功能*。更新了*图 95：STM32F469NIHx时钟树配置视图*。 *第 5.1节：功耗计算器视图*：增加了转换检查器选项。更新了*图 125：功耗计算器默认视图*、*图 126：电池选择*、以及*“图87：建立功耗序列”*。增加了*图 129：在已配置的序列中使能跳变检查器选项 - 所有跳变都有效*、*图 130：在已配置的序列上启用跳变检查器选项－至少一个跳变无效*和*图 131：转换检查器选项-显示日志*。更新了*图 135：构建序列后的功耗计算器视图*。更新了*第 节：管理序列步骤*和*第 节：管理整个序列（加载、保存和比较）*。更新了*图88：PCC编辑步骤窗口*和*图 138：结果区域描述*。 更新了*图 254：功耗计算示例*、*图155：序列表*、*图156：功耗计算结果*和*图158：功耗结果 - IP功耗图*。 更新了附录*B.3.1概述*和*B.3.5FreeRTOS*。
2015年5月28日	8	4.8	增加了*第 3.2.2节：从命令行安装STM32CubeMX*和*第 3.3.2节：在命令行模式下运行STM32CubeMX*。
2015年7月9日	9	4.9	增加了STLM32F7和STM32L4微控制器系列。 增加了*导入项目*功能。在“表13：文件菜单功能”中添加了*“导入”*功能。增加了*第 4.10节：“导入项目”窗口*。更新了*图 128：功耗系列：新步骤默认视图*、*“图88：PCC编辑步骤窗口”*，*“图83：功耗序列：配置新步骤”*（以STM32F4为例）”、*图 134：功耗计算器步骤配置窗口： 使用导入引脚排列使能ADC*和*“图87：外设功耗工具提示”*。 更新了*第 3.3.2节：在命令行模式下运行STM32CubeMX*中运行STM32CubeMX的命令行。 更新了*“第5.16节：配置视图”*中的注释。 在*第 4.8.1节*中增加了新时钟树配置功能。 更新了*图 221：中间件工具提示*。 修改了附录*B.1STM32CubeMX生成的C代码和用户部分*中的代码示例。 更新了附录*B.3.1概述*。 更新了附录*B.3.4FatFs*中生成的.h文件。

表24. 文档版本历史

日期	版本	STM32CubeMX 版本号	变更
2015年8月27日	10	4.10	将*引言*中的UM1742替换为UM1940。 更新了*第 3.3.2 节：在命令行模式下运行STM32CubeMX*中在命令行模式下运行STM32CubeMX的命令行。修改了*表 1：命令行摘要*。 更新了*第 4.2 节：新项目窗口*中的板子选择。 更新了*“第5.16 节：配置视图概述”*。更新了*“第5.14.1 节：外设和中间件配置窗口”*、*第 4.4.12 节：“GPIO配置”窗口*以及*第 4.4.13 节：“DMA配置”窗口*。增加了*第 4.4.11 节：”用户常量“配置窗口*。 更新了*第 4.8 节：“时钟配置”视图*，并添加了存储路径。 更新了*第 11.1 节：创建一个新STM32CubeMX项目*、*第 11.5 节：配置MCU时钟树*、*第 11.6 节：配置MCU初始化参数*、*第 11.7.2 节：下载固件包和生成C代码*和*第 11.8 节：构建和更新C代码项目*。增加了*第 11.9 节：切换到另一MCU*。 更新了*第 12 节：教程2 - 使用STM32429I-EVAL评估板的SD卡上的FatFs示例 STM32429I-EVAL 评估板*，并将STM32F429I-EVAL替换为STM32429I-EVAL。
2015年10月16日	11	4.11	更新了*图 8：“嵌入式软件包管理器”窗口*和*第 3.4.6 节：检查更新*。 *第 4.4.11 节：”用户常量“配置窗口*中支持字符串常量。 更新了*第 4.8 节：“时钟配置”视图*。 更新了*第 5.1 节：功耗计算器视图*。 修改了*图 254：功耗计算示例*。 更新了*第 13 节：教程 3 - 使用功耗计算器 优化嵌入式应用功耗等*。 在*第 3.1.3 节：软件要求*中增加了Eclipse Mars
2015年12月3日	12	4.12	代码生成选项现在受“项目设置”菜单支持。 更新了*第 3.1.3 节：软件要求*。 在*第 4.10 节：“导入项目”窗口*中增加了项目设置。更新了*图 112：自动项目导入*；修改了*手动项目导入*步骤，并更新了*图 113：手动项目导入*和*图 114：“导入项目” 菜单 - 尝试导入时出现错误*；修改了导入序列的第三步。 更新了*“图83：带有错误的时钟树配置视图”*。 在*第 6.1 节：仅使用HAL驱动程序生成STM32Cube代码 （默认模式）*中增加了mxconstants.h。 更新了*图 254：功耗计算示例*至*图 263：步骤10优化*。 更新了*图 264：优化后的功耗系列结果*。

表24. 文档版本历史

日期	版本	STM32CubeMX 版本号	变更
2016年2月3日	13	4.13	更新了*第 2.2 节：主要特性*： – 与.ioc文件相关的信息。 – 时钟树配置 – 自动更新STM32CubeMX和STM32Cube。 STM32CubeMX更新了*第 2.3 节：规则和限制*中与C代码生成相关的限制。 在*第 3.1.1 节：支持的操作系统和架构*中增加了Linux。更新了*第 3.1.3 节：软件要求*中的Java Run Time Environment版本号。 更新了*第 3.2.1 节：安装STM32CubeMX 独立版本*、*第 3.2.3 节：卸载STM32CubeMX独立版本*以及*“第3.3.1 节：下载STM32CubeMX插件安装包”*。 更新了*第 3.3.1 节：作为独立应用程序运行*。 更新了*第 4.9 节：“项目管理器”视图*和*第 4.12 节：“更新管理器”窗口*。 更新了*“第5.15.1 节：在引脚上固定和标记信号”*。 增加了*第 4.4.16 节：设置HAL时基源* 更新了*“图143：GPIO、DMA和NVIC设置的配置窗口选项卡（STM32F4系列）”*。 在*第 4.4.12 节：“GPIO配置”窗口*中增加了与输出模式下的GPIO配置相关的注释；更新了*图 61：“GPIO配置”窗口 - GPIO选择*。 修改了*图 125：功耗计算器默认视图*、*“图86：构建功耗序列”*、*图 127：步骤管理功能*、*图 129：在已配置的序列中使能跳变检查器选项 - 所有跳变都有效*、*图 130：在已配置的序列上启用跳变检查器选项－至少一个跳变无效*。 在*导入引脚排列*中增加了导入引脚列按钮图标。 增加了*选择/取消选择所有外设*。修改了*图 135：构建序列后的功耗计算器视图*。更新了*管理整个序列（加载、保存和比较）*。更新了*图 138：结果区域描述*和*“图100：外设功耗工具提示”*。 更新了*图 254：功耗计算示例*和*图 256：序列表*。 更新了*第 6.3 节：自定义代码生成*。 更新了*第 11.1 节：创建一个新STM32CubeMX项目*中的*图 200：具有MCU选择的引脚排列视图*和*图 201：无MCU选择窗口的引脚排列视图*。 更新了*第 11.6.2 节：配置外设*。 更新了*第 11.7.1 节：设置项目选项*中的*图 226：项目设置和工具链选择*和*图 227：“项目管理器”菜单-“代码生成器”选项卡*，并更新了*第 11.7.2 节：下载固件包和生成C代码*中的*图 228：缺少固件包警告消息*。

表24. 文档版本历史

日期	版本	STM32CubeMX版本号	变更
2016年3月15日	14	4.14	将STM32CubeMX版本号升级为4.14.0。 在*第 2.2 节：主要特性*中增加了导入之前保存的项目以及通过模板生成用户文件的内容。 在*第 3.1.1 节：支持的操作系统和架构*、*第 3.2.1 节：安装STM32CubeMX 独立版本*、*第 3.2.3 节：卸载STM32CubeMX独立版本*和*“第3.4.3 节：从Eclipse IDE 中运行STM32CubeMX插件”*中增加了MacOS。 在*第 3.3.2 节：在命令行模式下运行STM32CubeMX*中增加了允许通过模板生成用户文件的内容。 更新了*第 3.4.1 节：更新程序配置*中的新库安装序列。 更新了*“第5.7.3 节：引脚布局菜单”*中的*“图107：引脚布局菜单”*（已选择引脚布局选项卡）和*“图108：引脚布局菜单”*（未选中引脚选项卡）。 修改了*“表16：窗口菜单”*。 更新了*“第5.7 节：输出窗口”*。 更新了*图 99：“项目设置”窗口*和*第 4.9.1 节：“项目”选项卡*。 更新了*第 4.4.16 节：设置HAL时基源*中的*图 79：使用SysTick作为HAL时基（无FreeRTOS）时的NVIC设置，无FreeRTOS*和*图 80：使用FreeRTOS和SysTick作为HAL时基时的NVIC设置*。 更新了*第 4.4.11 节：”用户常量“配置窗口*中的*图 52：“用户常量”选项卡*和*图 53：摘录生成的main.h文件*。 *第 4.4.12 节：“GPIO配置”窗口*：更新了*图 61：“GPIO配置”窗口 - GPIO选择*、*图 62：按外设分组的GPIO配置*和*图 63：多引脚配置*。 更新了*第 4.4.14 节：“NVIC配置”窗口*。
2016年5月18日	15	4.15	导入项目功能不再局限于相同系列的MCU（请参见*第 2.2 节：主要特性*、*“第5.7.1 节：文件菜单”*和*第 4.10 节：“导入项目”窗口*）。 更新了*第 3.3.2 节：在命令行模式下运行STM32CubeMX*中的命令行。 *表 1：命令行摘要*：修改了所有与config命令相关的示例以及set dest_path <path>示例。 在*“表13：文件菜单功能”*中为*“加载项目”菜单*增加了注意事项。 更新了*“表14：项目菜单”*中的*“生成代码”菜单*描述。 更新了*“表15：引脚布局菜单”*中的*“设置未使用的GPIO”*菜单。 在*第 节：使用NVIC选项卡视图启用中断*中增加了启用FreeRTOS的实例。 增加了*第 4.4.15 节：FreeRTOS配置面板*。 更新了附录*B.3.5FreeRTOS*和*B.3.6LwIP*。

表24. 文档版本历史

日期	版本	STM32CubeMX版本号	变更
2016年9月23日	16	4.17	将整个文档中的*mxconstants.h*替换为*main.h*。 更新了*引言*、*第 3.1.1 节：支持的操作系统和架构*和*第 3.1.3 节：软件要求*。 增加了*第 3.4.3 节：安装STM32 MCU软件包补丁*。 更新了*表 2：主页快捷键*中的加载项目描述。 更新了*“表15：引脚布局菜单”*中的“清除引脚布局功能”。 更新了*第 4.9.3 节：“高级设置”选项卡*，增加底层驱动程序。 在*“表17：外设和中间件配置窗口中的按键和工具提示”*中增加了*“无检查”*和*“十进制和十六进制检查”*选项。 更新了*任务和队列选项卡*和*图 76：FreeRTOS堆用量*。 更新了*图 61：“GPIO配置”窗口 - GPIO选择*。 在整个文档中将PPC替换为功耗计算器。 增加了*第 6.2 节：使用底层驱动程序生成STM32Cube代码*；更新了*表 20：LL与HAL：STM32CubeMX生成的源文件*和*表 21：LL与HAL：STM32CubeMX生成函数与函数调用*。 更新了*图 305：引脚排列视图 - 启用RTC*。 增加了*第 14 节：教程4 - 通过串口与STM32L053xx Nucleo板通信示例* *STM32L053xx Nucleo板的通信示例*。 增加了STM32CubeMX版本号与文档修订版本之间的对应关系。
2016年11月21日	17	4.18	删除了*第 3.1.3 节：软件要求*中的Windows XP并增加了Windows 10。 更新了*第 3.2.3 节：卸载STM32CubeMX独立版本*。 在*表 1：命令行摘要*中增加了setDriver命令行。 增加了列出引脚排列兼容MCU的功能： – 更新了*表15：引脚布局菜单*。 – 增加了*第 15 节：教程5：将当前项目配置导出到兼容MCU兼容MCU* 在*第 4.9.1 节：“项目”选项卡*和*图 99：“项目设置”窗口*中增加了固件位置选择选项。 增加了恢复默认值功能： – 更新了*表 8：“外设和中间件配置”窗口按钮与工具提示* – 更新了*图 54：使用常量进行外设参数设置*。
2017年1月12日	18	4.19	项目导入不再限于同系列微控制器：更新了*引言*、*图 112：自动项目导入*、*图 113：手动项目导入*、*图 114：“导入项目” 菜单 - 尝试导入时出现错误*和*图 115：“导入项目” 菜单 - 调整之后导入成功*。 修改了附录*B.3.4FatFs*、*B.3.5FreeRTOS*和*B.3.6LwIP*。增加了附录*B.3.7Libjpeg*。

表24. 文档版本历史

日期	版本	STM32CubeMX版本号	变更
2017年3月2日	19	4.20	*表17：STM32CubeMX“芯片”视图 - 图标和颜色方案*: – 更新了替代功能示例列表。 – 更新了对应于引脚上映射的功能的示例和描述。 – 增加了共用相同引脚的模拟信号的示例和描述。 更新了*“图87：外设配置窗口”（STM32F4系列）图 52：“用户常量”选项卡*、*图 58：删除用于外设配置的用户常量 - 对外设配置的影响*、*图 59：在用户常量列表中搜索常量名称*和*图 60：在用户常量列表中搜索常量值*。 增加了*第 5.1.6 节：SMPS特性*。 增加了*第 6.4 节：其他C项目生成设置*。 在附录*B.3.7Libjpeg*中向包含Libjpeg的软件包列表中添加了STM32CubeF4。
2017年5月5日	20	4.21	在*第 1 节：STM32Cube 概述*中做了少量改动。 更新了*图 26：新项目窗口 - MCU选择器*和*图 99：“项目设置”窗口*。 更新了*第 4.9.1 节：“项目”选项卡*中项目设置的描述。 更新了*图 110：“高级设置”窗口*。 在附录*B.3.7Libjpeg*中嵌入了Libjpeg的软件包列表中增加了STM32CubeF2和STM32CubeH7。 修改了*图 339：STM32Cube嵌入式软件包*的外观。
2017年7月6日	21	4.22	将STM32H7增加到受支持STM32系列列表中。 在*第 3.4 节：使用STM32CubeMX进行更新*中增加了MCU数据和文档刷新功能，并更新了*图 6：“更新程序设置”窗口*。 在*第 4.2 节：新项目窗口*中增加了识别靠近MCU的功能，更新了*图 26：新项目窗口-MCU选择器*、增加了*图 29：新项目窗口-具有相近特性的MCU列表*和*图 30：新项目窗口-显示相近MCU的列表*，更新了*图 199：MCU选择*。 更新了*图 37：选择MCU时显示的STM32CubeMX主窗口*。 在*“表15：引脚布局菜单”*中增加了“顺时针旋转/逆时针旋转和顶视图/底视图”。 增加了*第 4.1.4 节：社交链接*。 更新了*图 146：为每个步骤配置SMPS模式*。 更新了*第 6.2 节：使用底层驱动程序生成STM32Cube代码*。 更新了*图 226：项目设置和工具链选择*。
2017年9月5日	22	4.22.1	在*引言*、*第 5.1 节：功耗计算器视图*和*第 6.2 节：使用底层驱动程序生成STM32Cube代码*中增加了STM32L4+系列。 在*第 3.3.1 节：作为独立应用程序运行*中增加了在MacOS上运行STM32CubeMX的指南。从*“第3.4.3节：从Eclipse IDE运行STM32CubeMX插件”*删除了MacOS。 增加了*第 18.8 节：以太网配置：为什么有时候我不能指定DP83848或LAN8742A？*

表24. 文档版本历史

日期	版本	STM32CubeMX版本号	变更
2017年10月18日	23	4.23	增加了*“第1节：常规信息”*。 将*第 4.2节：新项目窗口*中的**显示相近**按钮重命名为**显示类似项目**。 在**“第5.7.5节：帮助菜单”**中增加了**“刷新数据”**和*“文档与资源”菜单*。 在*第 6.2节：使用底层驱动程序生成STM32Cube代码*中增加了STM32F2、STM32F4和STM32F7系列。 增加了附录*B.3.8Mbed TLS*。 更新了用户手册修订版本22对应的STM32CubeMX版本号。
2018年1月16日	24	4.24	用“STM32Cube MCU封装”替换了“STM32Cube固件封装”。 更新了*第 1节：STM32Cube 概述*。 更新了*第 3.1.1节：支持的操作系统和架构*中的MacOS。更新了*第 3.1.3节：软件要求*中的Eclipse要求。 *第 3.4节：使用STM32CubeMX进行更新*： – 更新了“前言”一节 – 更新了*“图13：连接参数选项卡”的无代理* – *第 3.4.2节*重命名为“安装STM32 MCU软件包”并进行了更新。 – *第 3.4.3节*重命名为“安装STM32 MCU软件包补丁” – 增加了*第 3.4.4节：安装嵌入式软件包* – 更新了*第 3.4.6节：检查更新* 更新了*图 31：新项目窗口 - 板选择器*。 更新了*图 38：STM32CubeMX选择板时显示的主窗口（外设未初始化）*和介绍性文字。 更新了*图 39：选择板时显示的STM32CubeMX主窗口（外设以默认配置初始化）*和介绍性文字。 在*“表14：项目菜单”*中增加了“选择其他软件组件”菜单。 “安装新库”菜单重命名为“管理嵌入式软件包”，并在*“表17：帮助菜单”*中更新了相应的描述。 更新了*第 3.4.5节：删除已安装的嵌入式软件包*。 更新了*第 4.12节：“更新管理器”窗口* 增加了*第 4.13节：附加软件组件选择窗口*。 在*“表17：STM32CubeMX芯片视图”的图标和配色方案”*中增加了引脚栈功能。 *第 6.2节：使用底层驱动程序生成STM32Cube代码*：在支持低级驱动器的产品系列列表中增加了STM32F0、STM32F3、STM32L0。 *第 12节：教程2 - 使用STM32429I-EVAL评估板的SD卡上的FatFs示例STM32429I-EVAL评估板*：更新了*图 246：板选择*，并修改了生成项目和运行教程2的顺序中的步骤6。 *第 14节：教程4 - 通过串口与STM32L053xx Nucleo板通信示例STM32L053xx Nucleo板的通信示例*：更新了*图 265：选择NUCLEO_L053R8板*。 增加了*第 16节：教程6 – 将嵌入式软件包添加到用户项目*。
2018年1月16日	24（续）	4.24	增加了附录*B.3.9TouchSensing*和*B.3.10PDM2PCM*。 *第 4.4.14节：“NVIC配置”窗口*/中断的默认初始化序列：将中断使能代码对应的颜色由绿色改为黑色粗体。

表24. 文档版本历史

日期	版本	STM32CubeMX 版本号	变更
2018 年3月 7日	25	4.25	更新了*引言*、*第 1 节：STM32Cube 概述*、*第 2.3 节：规则和限制*、*第 3.2.1 节：安装STM32CubeMX 独立版本*、*第 4 节：STM32CubeMX用户界面*、*第 4.9.1 节：“项目”选项卡*以及*和“5.13.1 节：外设和中间件树面板”*。 对整个文档进行了少量文字修订。 更新了*“表13：文件菜单功能”*和*表 12：功率超载模式与HCLK频率之间的关系*。 更新了*图 26：新项目窗口 - MCU选择器*、*图 27：在MCU选择器中使能图形选择*、*图 99：“项目设置”窗口*、*图 104：选择不同的固件位置*、*“图77：启用STemWin框架”*、*“图116：图形的配置视图”*、*图 306：引脚排列视图 - 启用LSE和HSE时钟*以及*图 307：引脚排列视图 - 设置LSE/HSE时钟频率*。 增加了*导出至Excel功能*、*显示收藏的MCU特性*以及*“第4.4.16 节：图形框架和仿真器”*。 增加了*“第17节：教程8–使用STemWin Graphics框架”*，*“第18节：教程9：使用STM32CubeMX图形仿真器及其小节”*。 增加了*“第B.3.11 节：图形”*。
2018年9月 5日	26	4.27	在封面上更新了STM32Cube标志。 在整个文档中使用STM32Cube™ 替换STMCube™。更新了*第 1 节：STM32Cube 概述*。 更新了*图 1：STM32CubeMX C代码生成流程概述*。 更新了*第 2.2 节：主要特性*，以增加新功能：图形仿真器功能，支持CMSIS-Pack格式的嵌入式软件包和上下文帮助。 将*第 3.4 节*标题更改为“使用STM32CubeMX获取更新”。禁止显示的图形*“连接参数”选项卡-无代理*和*“连接参数”选项卡-使用系统代理参数*。更新了*图 9：管理嵌入式软件包 - 帮助菜单*。 在*第 3.4.4 节：安装嵌入式软件包*中，更新嵌入式软件包安装顺序的步骤3f并增加了*图 14：接受许可协议*。 *第 4.2 节：新项目窗口*：更新了*图 26：新项目窗口 - MCU选择器*、*图 28：将MCU标记为收藏项*和*图 31：新项目窗口 - 板选择器*。 *“第5.7.1 节：文件菜单”*：在*“表13：文件菜单功能”*中增加了有关“新项目”的注意事项。更新了*“图107：引脚布局菜单”*（选择了“引脚布局”选项卡）和*“图108：引脚布局菜单”*（未选择“引脚布局”选项卡）。 *第 4.9 节：“项目管理器”视图*: – 增加了与项目保存相关的注释（步骤3）。 – 更新了*图 99：“项目设置”窗口* – 更新了*第 4.9.1 节：“项目”选项卡*和*图 104：选择不同的固件位置*。 增加了*第 4.13.4 节：“组件依赖性”面板*，*上下文帮助*，*第 10 节：支持使用 CMSIS-Pack 标准的其他软件组件*以及*第 17 节：教程7–使用X-Cube-BLE1软件包*。

表24. 文档版本历史

日期	版本	STM32CubeMX 版本号	变更
2018年11月12日	27	4.28	更新了*第 3.4.2节：安装STM32 MCU软件包*、*第 3.4.4节：安装嵌入式软件包*、*第 3.4.5节：删除已安装的嵌入式软件包*、*第 3.4.6节：检查更新*和其中的数字。 更新了*第 4节：STM32CubeMX用户界面*、其小节以及其中的图和表格。 更新了*第 10节：支持使用 CMSIS-Pack 标准的其他软件组件*、*11.6.1*部分更新为*11.6.5*、*第 11.7.1节：设置项目选项*、*第 11.7.2节：下载固件包和生成C代码*、*第 11.8节：构建和更新C代码项目*、*第 11.9节：切换到另一MCU*、*第 12节：教程2 - 使用STM32429I-EVAL评估板的SD卡上的FatFs示例STM32429I-EVAL 评估板*以及其中的图、*第 15节：教程5：将当前项目配置导出到兼容MCU兼容MCU*以及其中的图、*第 16节：教程6 – 将嵌入式软件包添加到用户项目*以及*第 17节：教程7–使用X-Cube-BLE1软件包*图。

表24. 文档版本历史

日期	版本	STM32CubeMX 版本号	变更
2018年11月12日	27（续）	5.0	增加了“*第19节：教程10：使用ST-TouchGFX框架*”及其小节。 更新了*表 21：LL与HAL：STM32CubeMX生成函数与函数调用*。 删除了之前的“*图164：启用和配置CMSIS-Pack软件组件*”、“*图192：FatFs外设实例*”、“*图213：项目导入状态*”、“*图254：将软件组件选择保存为用户首选项*”、以及“*图268：配置X-Cube-BLE1*”。 更新了*图 1：STM32CubeMX C代码生成流程概述*、*图 3：STM32Cube安装向导*、“*图7：关闭STM32CubeMX透视图*”、“*图9：打开Eclipse插件*”、“*图10：STM32CubeMX透视图*”、*图 139：整体外设功耗*、*图 170：生成定义语句的用户常量*、*图 196：选择CMSIS-Pack软件组件*、*图 197：启用并配置CMSIS-Pack软件组件*、*图 198：使用CMSIS-Pack软件组件生成的项目*、*图 199：MCU选择*、*图 200：具有MCU选择的引脚排列视图*、*图 201：无MCU选择窗口的引脚排列视图*、*图 203：定时器配置*、*图 204：简单的引脚排列配置*、*图 205：项目另存为窗口*、*图 206：生成项目报告 - 新项目创建*、*图 207：生成项目报告 - 已成功创建项目*、*图 208：时钟树视图*、*图 213：“引脚布局和配置”视图*、*图 214：外设和中间件无配置参数的情况*、*图 215：定时器3配置窗口*、*图 216：定时器3配置*、*图 217：使能定时器3中断*、*图 218：GPIO配置颜色方案和工具提示*、*图 219：GPIO模式配置*、*图 220：DMA参数配置窗口*、*图 221：中间件工具提示*、*图 222：USB主机配置*、*图 222：USB主机配置*、*图 223：FatFs over USB模式已启用*、*图 224：启用FatF和USB的“系统”视图*、*图 225：FatFs定义语句*、*图 226：项目设置和工具链选择*、*图 227：“项目管理器”菜单-“代码生成器”选项卡*、*图 228：缺少固件包警告消息*、*图 230：要下载的更新程序设置*、*图 231：更新程序设置（已建立连接）*、*图 232：下载固件包*、*图 233：解压固件包*、*图 234：C代码生成完成消息*、*图 244：“导入项目”菜单*, *图 274：“项目设置”菜单*、*图 284：为当前项目启用的附加软件组件*、*图 285：软件包组件 - 没有可配置参数*、*图 286：软件包教程 - 项目设置*、*图 289：嵌入式软件包*、*图 291：安装嵌入式软件包*、*图 292：开始一个新项目-选择NUCLEO-L053R8板*、*图 293：开始一个新项目-初始化所有外设*、*图 294：选择X-Cube-BLE1组件*、*图 295：配置外设和GPIO*、*图 296：配置 NVIC 中断*、*图 297：启用X-Cube-BLE1*、*图 297：启用X-Cube-BLE1*、*图 298：配置SensorDemo项目*以及“*图312：图形仿真器用户界面*”。

表24. 文档版本历史

日期	版本	STM32CubeMX 版本号	变更
2019年2月19日	28	5.0	更新了*引言*、*第 1节：STM32Cube 概述*、*第 2.2节：主要特性*、*第 3.1.3节：软件要求*、*第 3.4.2节：安装STM32 MCU软件包*、*第 4节：STM32CubeMX用户界面*、*解决引脚冲突*、*第 4.4.10节：组件配置面板*、*第 4.8节："时钟配置"视图*、*第 4.9节："项目管理器"视图*、*第 4.9.1节："项目"选项卡*、*第 4.9.3节："高级设置"选项卡*、*使用转换检查器*、*第 9.2节：STM32CubeMX设备树的生成*、*第 6.3.2节：保存并选择用户模板*、*.extSettings文件示例和生成的结果*和*第 11.6.4节：配置DMA*。 增加了*第 4.5节：STM32MP1系列的"引脚布局和配置"视图*、*第 4.5.2节：启动阶段配置*、*Section 5: STM32CubeMX tools*、*第 9节：设备树生成（仅适于STM32MP1系列）*、*第 B.3.11节：STM32WPAN BLE/线程（仅适于STM32WB系列）*、*第 B.3.12节：OpenAmp和RESMGR_UTILITY（仅适于STM32MP1系列和STM32H7双核产品系列）*及其小节。 删除了以前的*"第1节：一般信息"*。 更新了*表 2：主页快捷键*、*表 5：元件列表*、*模式图标和颜色方案*、*表 6："引脚布局"菜单和快捷键*以及*表 9：时钟配置视图小工具*的标题。 更新了*图 99："项目设置"窗口*、*图 100：项目文件夹*、*图 104：选择不同的固件位置*、*图 112：自动项目导入*、*图 113：手动项目导入*、*图 114："导入项目" 菜单 - 尝试导入时出现错误*、*图 115："导入项目" 菜单 - 调整之后导入成功*、*图 116："设置未使用引脚"窗口*、*图 117："重置已使用引脚"窗口*、*图 124："关于"窗口*、*图 191：STM32CubeMX生成的DTS－提取3*、*图 196：选择CMSIS-Pack软件组件*、*图 197：启用并配置CMSIS-Pack软件组件*、*图 251：FATFS教程 - 项目设置*和*图 252：C代码生成完成消息*。
2019年4月16日	29	5.1	更新了*引言*。*第 3.1.3节：软件要求*、*第 4.2节：新项目窗口*、*MCU相近选择器特性*、*外部时钟源*、*导入引脚排列*、*选择/取消选择所有外设*、*第 4.5节：STM32MP1系列的"引脚布局和配置"视图*、*第 4.13节：附加软件组件选择窗口*、*第 5.2.1节：DDR 配置*、*第 6.2节：使用底层驱动程序生成STM32Cube代码*、*BLE 配置*和*第 B.3.12节：OpenAmp和RESMGR_UTILITY （STM32MP1系列和STM32H7双核产品系列）*。 增加了*第 4.2.1节：MCU选择器*，*第 4.2.2节：板选择*，*第 4.2.3节：交叉选择器*，*第 4.6节：STM32H7双核产品系列的"引脚布局和配置"视图*、*第 5.1.8节：功能示例（仅适于STM32MP1和STM32H7双核）*和*第 7节：双核MCU的代码生成 （仅适于STM32H7双核产品系列）*。 删除了先前的*"第3.3节：安装STM32CubeMX插件版本"*及其小节，以及先前的*"第3.4.3节：从Eclipse IDE运行STM32CubeMX插件"*。
2019年4月16日	29（续）	5.1	更新了*表 3：窗口菜单*。 将图*27*更新为*31*、*图 110："高级设置"窗口*、图*125*更新为*132*、*134*更新为*137*以及*139*更新为*148*、*图 226：项目设置和工具链选择*以及图*254*更新为*264*， 增加了*图 24：新项目窗口快捷键*，*图 83：STM32MP1系列：GPIO的分配选项*，*图 337：资源管理器：外设分配视图*以及*图 339：STM32Cube嵌入式软件包*。

表24. 文档版本历史

日期	版本	STM32CubeMX 版本号	变更
2019年10月1日	30	5.2	更新了*引言*。*第 2.2 节：主要特性*、*第 3.3.2 节：在命令行模式下运行STM32CubeMX*、*料号选择*、*第 4.13 节：附加软件组件选择窗口*、*第 4.13.1 节：软件组件简介*、*第 4.13.2 节：筛选器面板*、*第 4.13.3 节：“软件包”面板*、*第 4.13.4 节：“组件依赖性”面板*、*第 4.13.6 节：针对其他软件组件更新树形视图*、*第 5.1 节：功耗计算器视图*和*第 6.2 节：使用底层驱动程序生成STM32Cube代码*。 更新了*表 1：命令行摘要*、*表 6：“引脚布局”菜单和快捷键*、*表 16：“其他软件”窗口-“软件包”面板图标*和*表 17：“组件相关性”面板上下文帮助*。 更新了*图 20：STM32CubeMX主页*、*图 122：选择附加软件组件*、*图 123：附加软件组件 - 更新后的树状视图*、*图 196：选择CMSIS-Pack软件组件*和*图 294：选择X-Cube-BLE1组件*。 增加了*第 4.4.8 节：多重绑定封装的引脚布局*和*第 4.13.5 节：“详细信息和警告”面板*。 增加了*表 15：“其他软件”窗口-“软件包”面板列*
2019年12月13日	31	5.4	更新了*引言*、*第 1 节：STM32Cube 概述*、*第 4.2 节：新项目窗口*、*选择新项目的MCU/MPU*和*第 11.7.1 节：设置项目选项*。 增加了*第 4.7 节：在“引脚布局和配置”视图中启用安全性 （仅适于STM32L5系列）*及其小节、*第 4.8.2 节：保护时钟资源（仅适于STM32L5系列）*以及*第 8 节：启用Trustzone的代码生成（仅适于STM32L5系列）*。 删除了先前的*“第4.4.16 节：图形框架和模拟器”*、*“第17 节：教程8–使用STemWin图形框架”*、*“第18 节：教程9：使用STM32CubeMX图形仿真器”*、*“第19 节：教程10：使用ST-TouchGFX框架”*和*“第B.3.11 节：图形”*。 对整个文档进行了少量文字修订。 更新了*表 1：命令行摘要*。 更新了*图 46：“引脚布局”视图：具有多重绑定的MCU*、*图 47：“引脚布局”视图：扩展模式下的多重绑定*、*图 83：STM32MP1系列：GPIO的分配选项*、*图 99：“项目设置”窗口*、*图 155：DDR套件-连接到目标*、*图 156：DDR套件-目标已连接*、*图 157：DDR活动日志*、*图 158：DDR交互日志*、*图 159：DDR寄存器加载*、*图 160：来自U-Boot SPL的DDR测试列表*、*图 161：DDR测试套件结果*、*图 162：DDR测试历史*、*图 163：DDR调整必要条件*、*图 164：DDR调整过程*、*图 165：位去时滞*、*图 166：眼图训练（定中）面板*、*图 167：DDR调整-保存到配置*、*图 188：STM32CubeIDE工具链的项目设置*和*图 226：项目设置和工具链选择*。 增加了*图 25：为STM32L5系列启用Trust-zone*。

表25. 中文文档版本历史

日期	版本	STM32CubeMX 版本号	变更
2020年10月5日	1		中文初始版本。

重要通知 - 请仔细阅读

意法半导体公司及其子公司 （“ST”）保留随时对 ST 产品和 / 或本文档进行变更、更正、增强、修改和改进的权利，恕不另行通知。买方在订货之前应获取关于 ST 产品的最新信息。 ST 产品的销售依照订单确认时的相关 ST 销售条款。

买方自行负责对 ST 产品的选择和使用， ST 概不承担与应用协助或买方产品设计相关的任何责任。

ST 不对任何知识产权进行任何明示或默示的授权或许可。

转售的 ST 产品如有不同于此处提供的信息的规定，将导致 ST 针对该产品授予的任何保证失效。

ST 和 ST 徽标是 ST 的商标。若需 ST 商标的更多信息，请参考 *www.st.com/trademarks*。所有其他产品或服务名称均为其各自所有者的财产。

本文档中的信息取代本文档所有早期版本中提供的信息。