

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STMCube™系列（参见[第2节](#)），既可作为独立应用，也可作为Eclipse插件集成到集成开发环境（IDE）中。

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中打开和编译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：STM32Cube嵌入式软件包](#)）

虽然STM32CubeMX提供了一个用户界面并且生成的C代码兼容STM32MCU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),
STM32L0 (UM1749), STM32L1 (UM1816), STM32L4/L4+ (UM1884)和
STM32H7 (UM2217)的STM32Cube HAL/LL驱动程序用户手册。*



目录

1	概述	15
2	STM32Cube 概述	15
3	STM32CubeMX入门	16
3.1	原理	16
3.2	主要特性	18
3.3	规则和限制	19
4	安装和运行 STM32CubeMX	20
4.1	系统要求	20
4.1.1	支持的操作系统和架构	20
4.1.2	内存必要条件	20
4.1.3	软件要求	20
4.2	安装/卸载 STM32CubeMX 独立版本	20
4.2.1	安装STM32CubeMX 独立版本	20
4.2.2	从命令行安装STM32CubeMX	22
4.2.3	卸载STM32CubeMX独立版本	24
4.3	安装STM32CubeMX 插件版本	25
4.3.1	下载STM32CubeMX 插件安装包	25
4.3.2	在Eclipse IDE中安装STM32CubeMX 插件	25
4.3.3	在Eclipse IDE中卸载STM32CubeMX 插件	26
4.4	启动STM32CubeMX	28
4.4.1	STM32CubeMX 作为独立应用程序运行	28
4.4.2	在命令行模式下运行STM32CubeMX	28
4.4.3	从Eclipse IDE运行STM32CubeMX 插件	31
4.5	获取STM32Cube和第三方软件发布和更新	32
4.5.1	更新程序配置	34
4.5.2	安装STM32 MCU软件包	37
4.5.3	安装STM32 MCU软件包补丁	38
4.5.4	安装嵌入式软件包	38
4.5.5	删除已安装的嵌入式软件包	43
4.5.6	检查更新	45

5	STM32CubeMX用户界面	46
5.1	欢迎页面	46
5.2	新项目窗口	48
5.3	主窗口	53
5.4	工具栏和菜单	56
5.4.1	文件菜单	56
5.4.2	项目菜单	57
5.4.3	引脚布局菜单	57
5.4.4	窗口菜单	60
5.4.5	帮助菜单	60
5.4.6	社交链接	61
5.5	输出窗口	61
5.5.1	MCU选择面板	61
5.5.2	输出面板	62
5.6	“导入项目”窗口	62
5.7	“设置未使用 / 重置已使用GPIO”窗口	68
5.8	“项目设置”窗口	70
5.8.1	“项目”选项卡	72
5.8.2	“代码生成器”选项卡	76
5.8.3	“高级设置”选项卡	79
5.9	“更新管理器”窗口	81
5.10	附加软件组件选择窗口	81
5.10.1	软件组件简介	81
5.10.2	筛选器面板	83
5.10.3	软件组件表格	83
5.10.4	软件组件条件	84
5.11	“关于”窗口	87
5.12	“引脚布局”视图	88
5.12.1	“外设和中间件树”面板	90
5.12.2	“芯片”视图	91
5.12.3	“芯片”视图高级操作	95
5.12.4	保持当前信号布置	97
5.12.5	在引脚上锁定和标记信号	98
5.12.6	设置HAL时基源	99
5.13	配置视图	105

5.13.1	“外设和中间件配置”窗口	107
5.13.2	“用户常量”配置窗口	110
5.13.3	“GPIO配置”窗口	115
5.13.4	“DMA配置”窗口	118
5.13.5	“NVIC配置”窗口	121
5.13.6	FreeRTOS中间件配置视图	129
5.13.7	图形框架和仿真器	135
5.14	时钟树配置视图	139
5.14.1	时钟树配置功能	139
5.14.2	建议	144
5.14.3	STM32F43x/42x功率超载功能	145
5.14.4	时钟树词汇表	146
5.15	功耗计算器视图	147
5.15.1	建立功耗系列	148
5.15.2	配置功耗系列中的步骤	155
5.15.3	管理用户定义的功耗系列并审查结果	159
5.15.4	功耗系列步骤参数词汇表	162
5.15.5	电池词汇表	165
5.15.6	SMPS特性	165
6	STM32CubeMXC代码生成概述	170
6.1	仅使用HAL驱动程序生成STM32Cube代码（默认模式）	170
6.2	使用底层驱动程序生成STM32Cube代码	172
6.3	自定义代码生成	177
6.3.1	FreeMarker用户模板的STM32CubeMX数据模型	177
6.3.2	保存并选择用户模板	177
6.3.3	自定义代码生成	178
6.4	其他C项目生成设置	181
7	教程—使用STM32F4从引脚布局到生成项目C代码	185
7.1	创建一个新STM32CubeMX项目	185
7.2	配置MCU引脚布局	188
7.3	保存项目	189
7.4	生成报告	190
7.5	配置MCU时钟树	190
7.6	配置MCU初始化参数	193

7.6.1	初始条件	193
7.6.2	配置外设	194
7.6.3	配置GPIO	197
7.6.4	配置DMA	198
7.6.5	配置中间件	199
7.7	生成完整的C项目	202
7.7.1	设置项目选项	202
7.7.2	下载固件包和生成C代码	204
7.8	构建和更新C代码项目	209
7.9	切换到另一MCU	214
8	教程2 - 使用STM32429I-EVAL评估板的SD卡上的FatFs示例	216
9	教程 3 - 使用功耗计算器优化嵌入式应用功耗等	224
9.1	教程概述	224
9.2	应用程序示例说明	225
9.3	使用功耗计算器	225
9.3.1	创建功耗系列	225
9.3.2	优化应用功耗	228
10	教程4 - 通过串口与STM32L053xx Nucleo板通信示例	236
10.1	教程概述	236
10.2	创建一个新STM32CubeMX项目并选择Nucleo板	236
10.3	从“引脚布局”视图中选择功能	238
10.4	在“时钟配置”视图中配置MCU时钟树	241
10.5	在“配置”视图中配置外设参数	242
10.6	配置项目设置并生成项目	245
10.7	使用用户应用代码更新项目	246
10.8	编译并运行项目	247
10.9	将Tera Term软件配置为PC上的串行通信客户端	247
11	教程5：将当前项目配置导出到兼容MCU	249
12	教程6 – 将嵌入式软件包添加到用户项目	253
13	教程 7 – 使用STemWin图形框架	257

13.1	步骤1：为图形选择MCU	257
13.2	步骤2：在引脚布局视图中启用STemWin	257
13.3	步骤3：从配置窗口中配置STemWin参数	259
13.4	步骤4：在配置窗口中使用STemWin GUIBuilder工具	259
13.5	步骤5：生成嵌入式C项目和更新	261
14	教程8：使用STM32CubeMX图形仿真器	263
15	FAQ	266
15.1	在“引脚布局配置”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？	266
15.2	我如何手动强制进行功能重新映射？	266
15.3	为什么芯片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）？	266
15.4	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？	266
15.5	为何RTC复用器在时钟树视图中仍无效？	267
15.6	如何选择LSE和HSE作为时钟源并更改频率？	268
15.7	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？	268
15.8	以太网配置：为什么有时候我不能指定DP83848或LAN8742A？	269
附录A	STM32CubeMX引脚分配规则	270
A.1	块一致性	270
A.2	块间依赖性	274
A.3	一个块 = 一种外设模式	277
A.4	块重新映射（仅限STM32F10x）	277
A.5	功能重新映射	278
A.6	块转移（仅适用于STM32F10x，且“保留当前信号布置”已取消选中）	279
A.7	设置或清除外设模式	280
A.8	分别映射功能	280
A.9	GPIO信号映射	280
附录B	STM32CubeMXC代码生成设计选择和限制	281
B.1	STM32CubeMX生成的C代码和用户部分	281
B.2	STM32CubeMX外设初始化设计选择	281

B.3	STM32CubeMX中间件初始化设计选择和限制	282
B.3.1	概述	282
B.3.2	USB 主机	283
B.3.3	USB设备	283
B.3.4	FatFs	283
B.3.5	FreeRTOS	284
B.3.6	LwIP	285
B.3.7	Libjpeg	287
B.3.8	Mbed TLS	288
B.3.9	TouchSensing	291
B.3.10	PDM2PCM	294
B.3.11	图形	294
附录C	STM32微控制器命名规则	296
附录D	STM32微控制器功耗参数	298
D.1	功耗模式	298
D.1.1	STM32L1系列	298
D.1.2	STM32F4系列	299
D.1.3	STM32L0系列	300
D.2	功耗范围	301
D.2.1	STM32L1系列有三种VCORE范围	301
D.2.2	STM32F4系列有多种VCORE级别	302
D.2.3	STM32L0系列有三种VCORE范围	302
附录E	STM32Cube嵌入式软件包	303
16	版本历史	304

表格索引

表1.	命令行摘要	29
表2.	欢迎页面快捷方式	47
表3.	文件菜单功能	56
表4.	项目菜单	57
表5.	引脚布局菜单	58
表6.	窗口菜单	60
表7.	帮助菜单	60
表8.	软件组件	83
表9.	软件组件条件图标	84
表10.	“外设和中间件树”面板 - 图标和颜色方案	90
表11.	STM32CubeMX “芯片”视图 - 图标和颜色方案	92
表12.	外设和中间件配置按钮	106
表13.	“外设和中间件配置”窗口按钮与工具提示	108
表14.	时钟树视图小工具	143
表15.	电压调节与功率超载和HCLK频率	146
表16.	功率超载模式与HCLK频率之间的关系	146
表17.	词汇表	146
表18.	LL与HAL代码生成: STM32CubeMX项目中包含的驱动	173
表19.	LL与HAL代码生成: STM32CubeMX生成的头文件	173
表20.	LL与HAL: STM32CubeMX生成的源文件	174
表21.	LL与HAL: STM32CubeMX生成函数与函数调用	174
表22.	文档版本历史	304
表23.	中文文档版本历史	315

图片索引

图1.	STM32CubeMX C代码生成流程概述	17
图2.	在交互模式下的STM32CubeMX 安装示例	22
图3.	STM32Cube安装向导	23
图4.	自动安装命令行	24
图5.	添加STM32CubeMX插件档案	25
图6.	安装STM32CubeMX插件	26
图7.	关闭STM32CubeMX视图	26
图8.	卸载STM32CubeMX插件	27
图9.	打开Eclipse插件	31
图10.	STM32CubeMX 视图	32
图11.	显示Windows默认代理设置	33
图12.	“更新程序设置”窗口	34
图13.	“连接参数”选项卡 - 无代理	35
图14.	“连接参数”选项卡 - 使用系统代理参数	36
图15.	“连接参数”选项卡 - 手动配置代理服务器	36
图16.	“嵌入式软件包管理器”窗口	37
图17.	管理嵌入式软件包 - 帮助菜单	39
图18.	管理嵌入式软件包 - 新增URL	40
图19.	检查外部包.pdsc文件URL的有效性	40
图20.	用户定义的软件包列表	41
图21.	选择嵌入式软件包版本	42
图22.	嵌入式软件包版本 - 成功安装	43
图23.	删除库	44
图24.	删除库确认消息	44
图25.	库删除进度窗口	44
图26.	帮助菜单: 检查更新	45
图27.	STM32CubeMX欢迎页面	47
图28.	新项目窗口 - MCU选择器	48
图29.	在MCU选择器中使能图形选择	49
图30.	将MCU标记为收藏项	50
图31.	新项目窗口 - 具有相近MCU特性的MCU列表	51
图32.	新项目窗口 - 显示相近MCU的MCU列表	51
图33.	新项目窗口 - 板选择器	52
图34.	选择MCU时显示的STM32CubeMX主窗口	53
图35.	选择板时显示的STM32CubeMX主窗口 (未选中外设默认选项)	54
图36.	选择板时显示的STM32CubeMX主窗口 (选中外设默认选项)	55
图37.	引脚布局菜单 (已选择引脚布局选项卡)	57
图38.	引脚布局菜单 (未选择引脚布局选项卡)	58
图39.	社交平台链接	61
图40.	MCU选择菜单	61
图41.	输出面板	62
图42.	自动项目导入	63
图43.	手动项目导入	64
图44.	“导入项目”菜单 - 尝试导入时出现错误	66
图45.	“导入项目”菜单 - 调整之后导入成功	67
图46.	“设置未使用引脚”窗口	68
图47.	“重置已使用引脚”窗口	68
图48.	在勾选了“保持当前信号布置”选项的情况下设置未使用的GPIO引脚	69

图49.	在未勾选“保持当前信号布置”选项的情况下设置未使用的GPIO引脚.....	70
图50.	“项目设置”窗口	71
图51.	项目文件夹	72
图52.	选择不同的固件位置	74
图53.	固件位置选择错误消息	75
图54.	建议的新固件库结构	75
图55.	项目设置代码生成器	77
图56.	“模板设置”窗口	78
图57.	生成的项目模板	79
图58.	“高级设置”窗口	80
图59.	在不使用C语言“static”关键字的情况下生成的初始化函数	80
图60.	附加软件组件-显示包和捆绑的压缩视图	82
图61.	附加软件组件 - 展开的视图显示组件详细信息	82
图62.	依赖性解决: 找到解决方案	84
图63.	依赖性解决: 未找到解决方案	85
图64.	软件组件条件 - 解决方案建议及指引	85
图65.	已解析的软件组件条件	86
图66.	选择附加软件组件	86
图67.	附加软件组件 - 更新后的树状视图	87
图68.	“关于”窗口	87
图69.	STM32CubeMX“引脚布局”视图	89
图70.	“芯片”视图	91
图71.	红色突出显示和工具提示示例: 没有可用的模式配置	95
图72.	橙色突出显示和工具提示示例: 一些配置不可用	95
图73.	工具提示示例: 全部配置不可用	95
图74.	从“芯片”视图修改引脚分配	96
图75.	对引脚一致性功能块进行重新映射的示例	97
图76.	“引脚/信号选项”窗口	99
图77.	选择HAL时基源 (以STM32F407为例)	100
图78.	TIM2被选为HAL时基源	100
图79.	使用SysTick作为HAL时基 (无FreeRTOS) 时的NVIC设置, 无FreeRTOS	101
图80.	使用FreeRTOS和SysTick作为HAL时基时的NVIC设置	102
图81.	使用FreeRTOS和TIM2作为HAL时基时的NVIC设置	104
图82.	STM32CubeMX“配置”视图	105
图83.	面向GPIO、DMA和NVIC设置 (STM32F4系列) 的配置窗口	106
图84.	“外设配置”窗口 (STM32F4系列)	107
图85.	“用户常量”窗口	110
图86.	摘录生成的main.h文件	111
图87.	使用常量进行外设参数设置	111
图88.	指定用户常量值和名称	112
图89.	当常量已经用于另一个常量的定义时, 不允许删除用户常量	113
图90.	删除用于参数配置的用户常量 - 确认请求	113
图91.	删除用于外设配置的用户常量 - 对外设配置的影响	114
图92.	在用户常量列表中搜索常量名称	114
图93.	在用户常量列表中搜索常量值	115
图94.	“GPIO配置”窗口 - GPIO选择	116
图95.	“GPIO配置”窗口 - 显示GPIO设置	117
图96.	按外设分组的GPIO配置	117
图97.	多引脚配置	118
图98.	添加新的DMA请求	119
图99.	DMA配置	120
图100.	DMA MemToMem配置	121

图101.	“NVIC配置”选项卡 - FreeRTOS已禁用	122
图102.	“NVIC配置”选项卡 - FreeRTOS已启用	123
图103.	I2C “NVIC配置”窗口	123
图104.	NVIC代码生成 – 所有中断已启用	125
图105.	NVIC代码生成 – 中断初始化序列配置	127
图106.	NVIC代码生成 – IRQ处理程序生成	128
图107.	FreeRTOS配置视图	129
图108.	FreeRTOS: 配置任务和队列	130
图109.	FreeRTOS: 创建新任务	131
图110.	FreeRTOS - 配置定时器、互斥量和信号量	133
图111.	FreeRTOS堆用量	135
图112.	使能STemWin框架	136
图113.	图形配置视图	137
图114.	图形配置窗口	137
图115.	保存更改	138
图116.	STM32F429xx时钟树配置视图	142
图117.	有错误的时钟树配置视图	142
图118.	时钟树配置：通过引脚布局视图使能RTC、RCC时钟源和输出	144
图119.	时钟树配置：RCC外设高级参数	145
图120.	功耗计算器默认视图	148
图121.	电池选择	149
图122.	建立功耗系列	150
图123.	步骤管理功能	150
图124.	功耗系列：新步骤默认视图	151
图125.	编辑步骤窗口	152
图126.	在已配置的序列中使能跳变检查器选项 - 所有跳变都有效	153
图127.	在已配置的序列中使能跳变检查器选项 - 至少有一个跳变无效	154
图128.	跳变检查器选项 - 显示日志	154
图129.	插补功耗	156
图130.	在引脚布局视图中选择的ADC	157
图131.	功耗计算器步骤配置窗口：使用导入引脚布局使能ADC	158
图132.	构建序列后的功耗计算器视图	159
图133.	序列表管理功能	160
图134.	功耗：外设功耗图	161
图135.	结果区域描述	162
图136.	外设功耗工具提示	164
图137.	为当前项目选择SMPS	166
图138.	SMPS数据库 - 添加新的SMPS模型	166
图139.	SMPS数据库 - 选择不同的SMPS模型	167
图140.	使用新的SMPS模型更新当前项目配置	167
图141.	已选择新模型的SMPS数据库管理窗口	167
图142.	SMPS跳变检查器和状态图助手窗口	168
图143.	为每个步骤配置SMPS模式	169
图144.	生成定义语句的引脚标签	171
图145.	生成定义语句的用户常量	171
图146.	重复标签	171
图147.	基于HAL的外设初始化：uart.c代码片段	175
图148.	基于LL的外设初始化：uart.c代码片段	176
图149.	HAL与LL：main.c代码片段	176
图150.	extra_templates文件夹 - 默认内容	178
图151.	包含用户模板的extra_templates文件夹	179
图152.	具有相应的自定义生成文件的项目根文件夹	179

图153.	模板的用户自定义文件夹	180
图154.	具有相应自定义生成文件的自定义文件夹	180
图155.	为预处理定义语句更新.ewp项目文件（EWARM IDE）	182
图156.	更新stm32f4xx_hal_conf.h文件以使能选定模块	183
图157.	添加到EWARM IDE中的组的新组和新文件	183
图158.	EWARM IDE中的预处理器定义语句	184
图159.	MCU选择	185
图160.	具有MCU选择的引脚布局视图	186
图161.	无MCU选择窗口的引脚布局视图	187
图162.	GPIO引脚配置	188
图163.	定时器配置	188
图164.	简单的引脚布局配置	189
图165.	项目另存为窗口	189
图166.	生成项目报告 - 新项目创建	190
图167.	生成项目报告 - 已成功创建项目	190
图168.	时钟树视图	191
图169.	HSI时钟使能	192
图170.	HSE时钟源已禁用	192
图171.	HSE时钟源已使能	192
图172.	外部PLL时钟源已使能	192
图173.	配置视图	194
图174.	外设和中间件无配置参数的情况	194
图175.	定时器3配置窗口	195
图176.	定时器3配置	196
图177.	使能定时器3中断	197
图178.	GPIO配置颜色方案和工具提示	197
图179.	GPIO模式配置	198
图180.	DMA参数配置窗口	199
图181.	FatFs禁用	199
图182.	USB主机配置	200
图183.	FatFs over USB模式已启用	200
图184.	已启用FatFs和USB的配置视图	200
图185.	FatFs外设实例	201
图186.	FatFs定义语句	201
图187.	项目设置和工具链选择	202
图188.	“项目设置”菜单 - “代码生成器”选项卡	203
图189.	缺少固件包警告消息	204
图190.	下载过程中出错	204
图191.	要下载的更新程序设置	205
图192.	更新程序设置（已建立连接）	206
图193.	下载固件包	206
图194.	解压固件包	207
图195.	C代码生成完成消息	207
图196.	C代码生成输出文件夹	208
图197.	C代码生成输出：项目文件夹	209
图198.	EWARM的C代码生成输出	210
图199.	在IAR™ IDE中打开STM32CubeMX生成的项目	211
图200.	IAR™选项	212
图201.	SWD连接	212
图202.	项目创建日志	213
图203.	用户部分2	213
图204.	用户部分4	214

图205.	“导入项目”菜单	215
图206.	项目导入状态	215
图207.	板子外设初始化对话框	216
图208.	板选择	217
图209.	SDIO外设配置	217
图210.	FatFs模式配置	218
图211.	RCC外设配置	218
图212.	时钟树视图	218
图213.	“项目设置”菜单 - “代码生成器”选项卡	219
图214.	C代码生成完成消息	219
图215.	IDE工作空间	220
图216.	功耗计算示例	226
图217.	VDD和电池选择菜单	227
图218.	序列表	227
图219.	优化前的序列结果	228
图220.	步骤1优化	229
图221.	步骤5优化	230
图222.	步骤6优化	231
图223.	步骤7优化	232
图224.	步骤8优化	233
图225.	步骤10优化	234
图226.	优化后的功耗系列结果	235
图227.	选择NUCLEO_L053R8板	237
图228.	选择调试引脚	238
图229.	选择TIM2时钟源	239
图230.	为USART2选择异步模式	240
图231.	检查引脚分配	240
图232.	配置MCU时钟树	241
图233.	配置USART2参数	242
图234.	配置TIM2参数	243
图235.	启用TIM2中断	244
图236.	“项目设置”菜单	245
图237.	生成代码	246
图238.	检查通信端口	247
图239.	设置Tera Term端口参数	248
图240.	设置Tera Term端口参数	248
图241.	已有项目或新项目引脚布局	249
图242.	与引脚布局兼容的MCU列表 - 部分匹配且硬件兼容	250
图243.	与引脚布局兼容的MCU列表 - 完全匹配和部分匹配	250
图244.	选择兼容MCU，并导入配置	251
图245.	配置已导入到所选的兼容MCU	252
图246.	为当前项目启用的附加软件组件	253
图247.	将软件组件选择保存为用户首选项	254
图248.	软件包组件 - 没有可配置参数	254
图249.	软件包教程 - 项目设置	255
图250.	生成的项目以及第三方软件包组件	256
图251.	教程 - 为图形选择MCU	257
图252.	图形框架工具提示	258
图253.	启用STemWin框架	258
图254.	STemWin图形框架配置窗口	259
图255.	STemWin GUIBuilder配置面板	260
图256.	STemWin GUIBuilder	261

图257.	StemWin生成的项目和文件	262
图258.	“配置”视图中的GFXSIMULATOR	263
图259.	“图形仿真器”用户界面	264
图260.	图形仿真器 - “当前配置”字段	265
图261.	Java™控制面板	267
图262.	引脚布局视图 - 启用RTC	267
图263.	引脚布局视图 - 启用LSE和HSE时钟	268
图264.	引脚布局视图 - 设置LSE/HSE时钟频率	268
图265.	块映射	271
图266.	块重新映射	272
图267.	块重新映射 - 示例1	273
图268.	块重新映射 - 示例2	274
图269.	块相关性 - SPI信号分配给PB3/4/5	275
图270.	块相关性 - SPI1_MISO功能分配给PA6	276
图271.	一个块 = 一种外设模式 - I2C1_SMBA功能分配给PB5	277
图272.	块重新映射 - 示例2	278
图273.	功能重新映射示例	278
图274.	未应用块转移	279
图275.	已应用块转移	280
图276.	FreeRTOS HOOK函数将由用户完成	284
图277.	LwIP 1.4.1配置	285
图278.	LwIP 1.5配置	286
图279.	Libjpeg配置窗口	288
图280.	Mbed TLS (无LwIP)	289
图281.	Mbed TLS (有LwIP和FreeRTOS)	290
图282.	Mbed TLS配置窗口	291
图283.	启用TouchSensing外设	292
图284.	触摸感应传感器选择面板	293
图285.	TouchSensing配置面板	294
图286.	图形应用架构	295
图287.	STM32微控制器产品编号模式	297
图288.	STM32Cube嵌入式软件包	303

1 概述

STM32CubeMX 支持 32 位基于 ARM® Cortex® 的微控制器。



2 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是STMCube™的实现，涵盖了整个STM32产品。

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。LL API仅对一部分外设可用。
 - 一套一致的中间件，比如RTOS、USB、TCP/IP、图形库
 - 提供了一套完整示例以及嵌入式软件工具。

3 STM32CubeMX入门

3.1 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，从而最大限度地复用软件并缩短创建目标系统的时间：

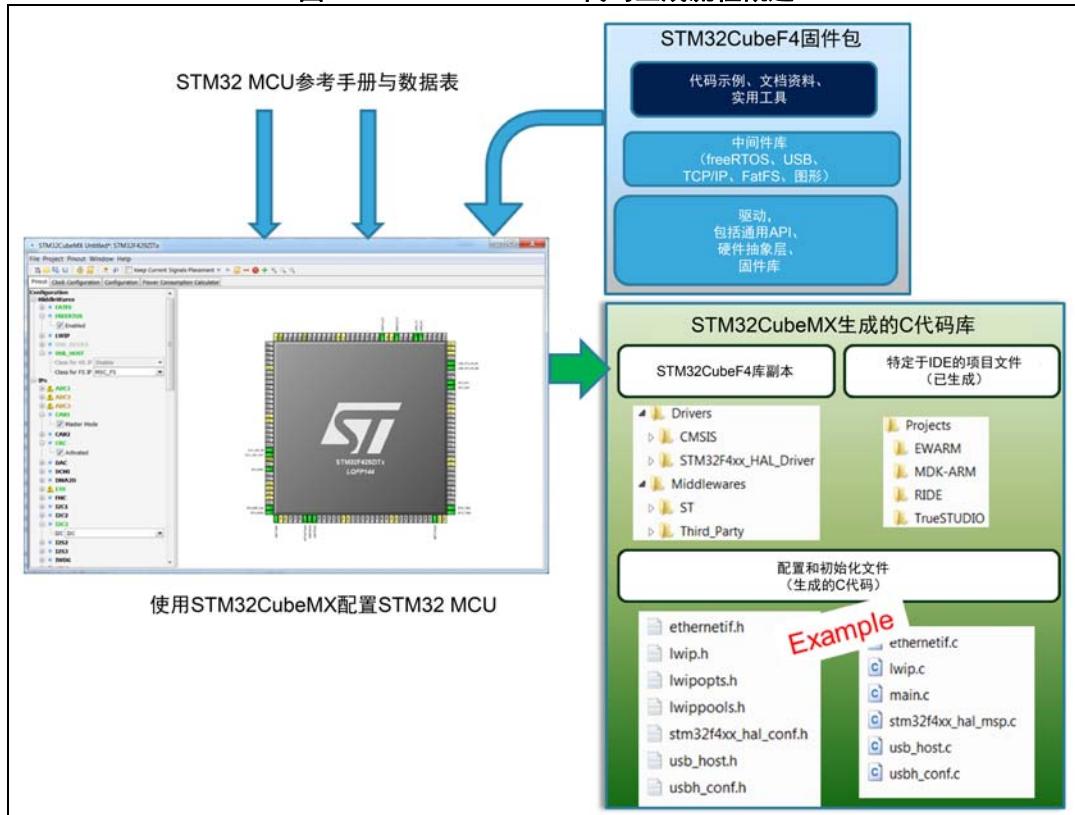
- STM32Cube固件解决方案提出了跨STM32产品的通用硬件抽象层API，可实现软件复用和应用程序设计的可移植性。
- 凭借STM32CubeMX内置的STM32微控制器、外设和中间件（LwIP和USB通信协议栈，用于小型嵌入式系统的FatFs文件系统，FreeRTOS）知识，迁移时间得到了优化。

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



3.2 主要特性

STM32CubeMX 具备以下特性：

- **项目管理**

STM32CubeMX 可创建、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 还允许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：

- 仅限MCU配置：.ioc文件保存在与其他.ioc文件相邻的任何位置。
- 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。

- **轻松执行引脚布局配置**

- 在“引脚布局”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“芯片”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。
- 引脚布局配置可以导出为.csv文件。

- **完整的项目生成**

项目生成包括一组IDE的引脚布局、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布局开始，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文件夹中添加用户定义的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，用户可以选择具体的开发工具链（IDE）以生成 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 AC6 System Workbench）直接导入。

- **功耗计算**

用户可以首先选择微控制器料号和电池类型，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。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精确匹配时，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。

3.3 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的代码生成。
- 有关引脚分配规则的说明，请参见 [附录A](#)。
- 有关 STM32CubeMXC 代码生成的设计选择和限制的说明，请参见 [附录B](#)。

4 安装和运行 STM32CubeMX

4.1 系统要求

4.1.1 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)

4.1.2 内存必要条件

- 建议最低RAM: 2 GB。

4.1.3 软件要求

必须安装以下软件：

- 对于Windows和Linux，请安装1.7.0_45或更高版本的Java™运行时环境
如果您的计算机上未安装Java™或者您已安装了旧版本，则STM32CubeMX安装程序将打开Java™下载网页并停止。
- 对于macOS，请安装Java™开发套件1.7.0_45或更高版本
- 对于Eclipse插件安装，请安装以下IDE之一：
 - Eclipse Mars (4.5)
 - Eclipse Neon (4.6)
 - Eclipse Oxygen (4.7)

4.2 安装/卸载 STM32CubeMX 独立版本

4.2.1 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) 双击SetupSTM32CubeMX-VERSION.linux文件（或从控制台窗口启动）。
- 在macOS上：
- 确保您拥有管理员权限。
 - 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.`
- 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文件显示为立方体图标。双击这些文件，使用STM32CubeMX打开文件。
 - 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），则向导将建议下载并停止。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。

4.2.2 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。

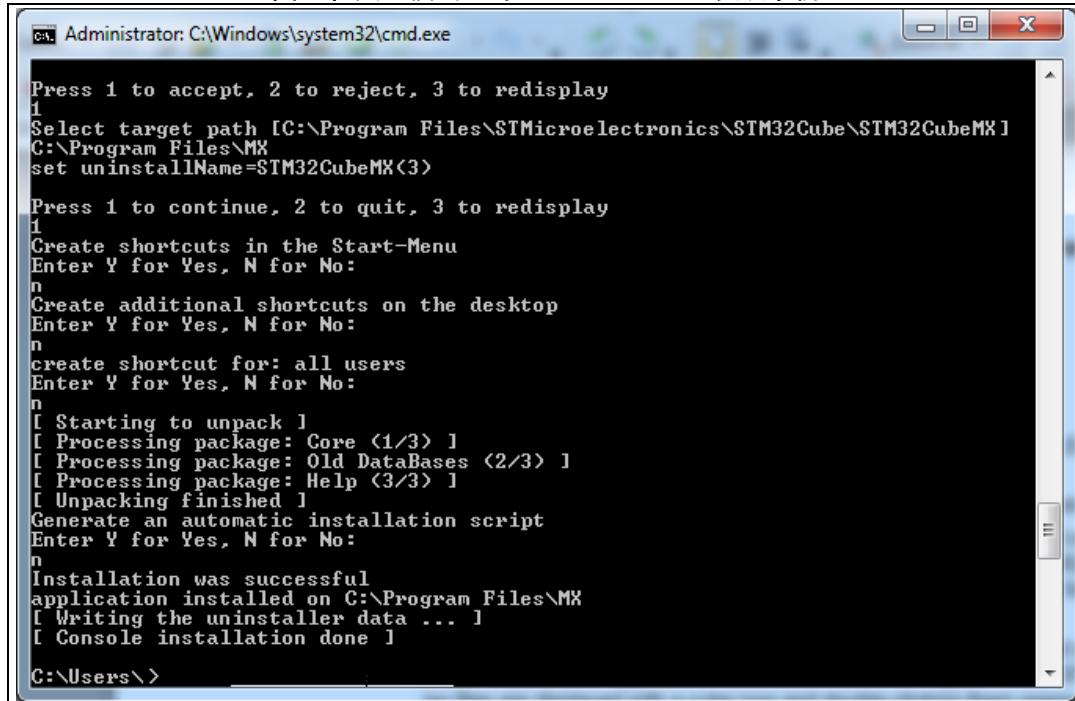
交互模式

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



The screenshot shows a Windows Command Prompt window titled "Administrator: C:\Windows\system32\cmd.exe". The window contains the following text:

```
Press 1 to accept, 2 to reject, 3 to redisplay
1
Select target path [C:\Program Files\STMicroelectronics\STM32Cube\STM32CubeMX]
C:\Program Files\MX
set uninstallName=STM32CubeMX<3>

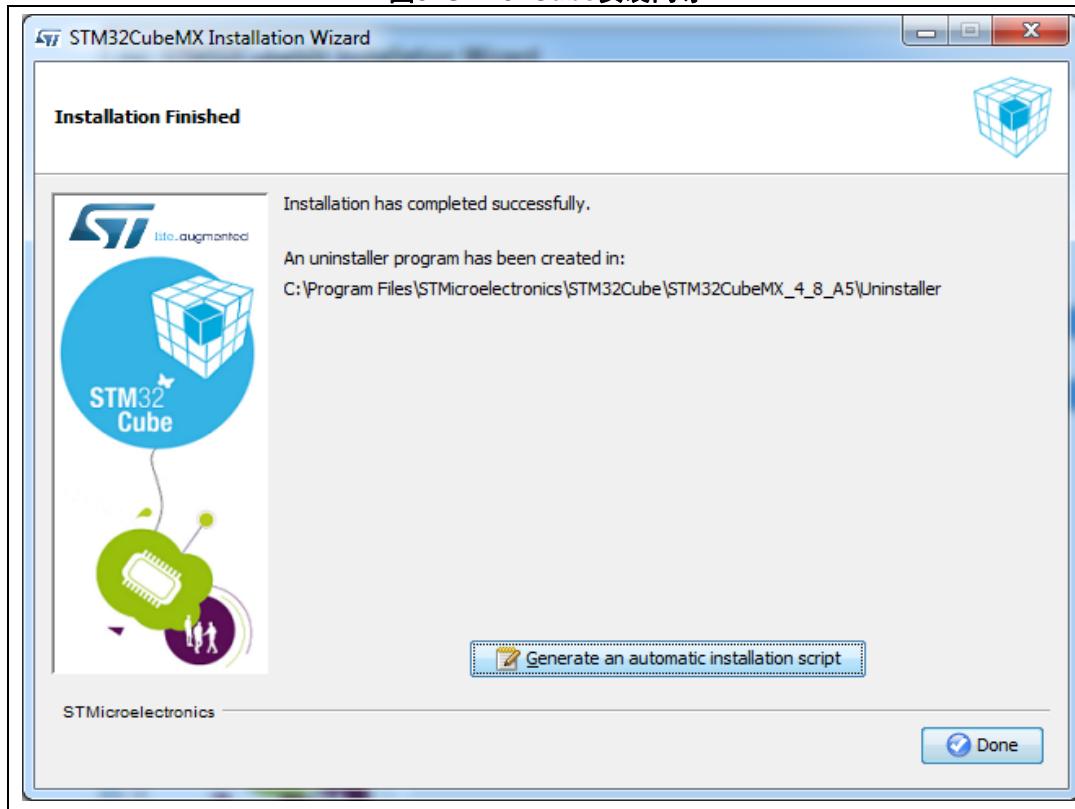
Press 1 to continue, 2 to quit, 3 to redisplay
1
Create shortcuts in the Start-Menu
Enter Y for Yes, N for No:
n
Create additional shortcuts on the desktop
Enter Y for Yes, N for No:
n
create shortcut for: all users
Enter Y for Yes, N for No:
n
[ Starting to unpack ]
[ Processing package: Core <1/3> ]
[ Processing package: Old DataBases <2/3> ]
[ Processing package: Help <3/3> ]
[ Unpacking finished ]
Generate an automatic installation script
Enter Y for Yes, N for No:
n
Installation was successful
application installed on C:\Program Files\MX
[ Writing the uninstaller data ... ]
[ Console installation done ]

C:\Users\>
```

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，可以生成包含用户安装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

```

Administrator: C:\Windows\system32\cmd.exe
The STM32CubeMX installer you are attempting to run seems to have a copy already
running.

This could be from a previous failed installation attempt or you may have accidentally
launched the installer twice. The recommended action is to select 'No' and wait for the other
copy of the installer to start. If you are sure there is no other copy of the installer
running, click the 'Yes' button to allow this installer to run.

Are you sure you want to continue with this installation?
Enter Y for Yes, N for No:
y
[ Starting automated installation ]
set uninstallName=STM32CubeMX<2>
[ Starting to unpack ]
[ Processing package: Core <1/3> ]
[ Processing package: Old DataBases <2/3> ]
[ Processing package: Help <3/3> ]
[ Unpacking finished ]
[ Writing the uninstaller data ... ]
[ Automated installation done ]

C:\Users\>

```

4.2.3 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”。

4.3 安装STM32CubeMX 插件版本

STM32CubeMX 插件可以安装在Eclipse IDE 开发工具链中。本节描述了与安装相关的步骤。

4.3.1 下载STM32CubeMX 插件安装包

如要下载STM32CubeMX插件，需遵循以下顺序：

1. 前往<http://www.st.com/stm32cubemx>。
2. 下载STM32CubeMX- Eclipse-plug-in .zip文件到本地磁盘。

4.3.2 在Eclipse IDE中安装STM32CubeMX 插件

如要在Eclipse IDE安装STM32CubeMX 插件，需遵循以下顺序：

1. 启动Eclipse环境。
2. 从主菜单栏中选择“帮助”>“安装新软件”。出现“可用软件”窗口。
3. 点击“添加”。打开“添加存储库”窗口。
4. 点击“存档”。打开“存储库存档”浏览器。
5. 选择您下载的STM32CubeMX-Eclipse-plug-in .zip文件，然后点击“打开”（参见图 5）。
6. 在“添加存储库”对话框中点击“确定”，
7. 选中STM32CubeMX_Eclipse_插件并点击“下一步”（参见图 6）。
8. 在“安装详情”对话框中点击“下一步”。
9. 点击“审阅许可”对话框中的“我接受许可协议的条款”，然后点击“完成”。
10. 点击“安全警告”菜单中的“确定”。
11. 请求重启Eclipse IDE时点击“确定”（参见[在命令行模式下运行STM32CubeMX](#)）。

图5. 添加STM32CubeMX插件档案

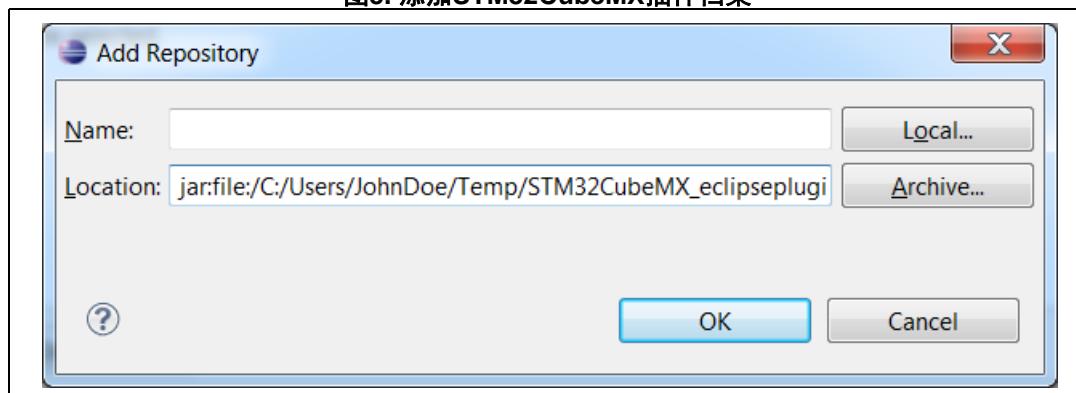
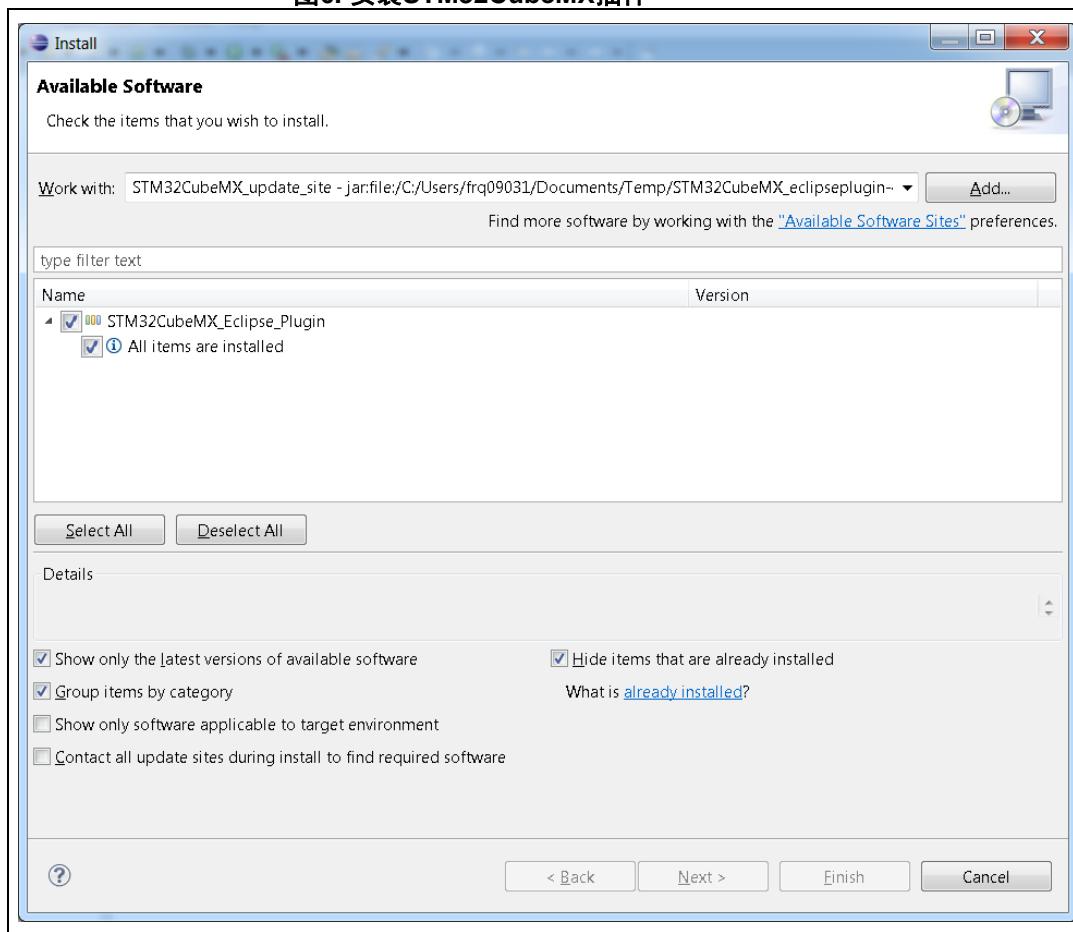


图6. 安装STM32CubeMX插件



4.3.3 在Eclipse IDE中卸载STM32CubeMX 插件

如要在Eclipse IDE卸载STM32CubeMX 插件，需遵循以下顺序：

1. 在Eclipse中，右键点击STM32CubeMX视图图标（参见 图 7），选择“关闭”。
2. 在Eclipse“帮助”菜单中，选择“安装新软件”。
3. 点击“已安装软件”选项卡，并选择 STM32CubeMX，随后点击“卸载”。
4. 在“卸载详情”菜单中点击“完成”（参见 图 8）。

图7. 关闭STM32CubeMX视图

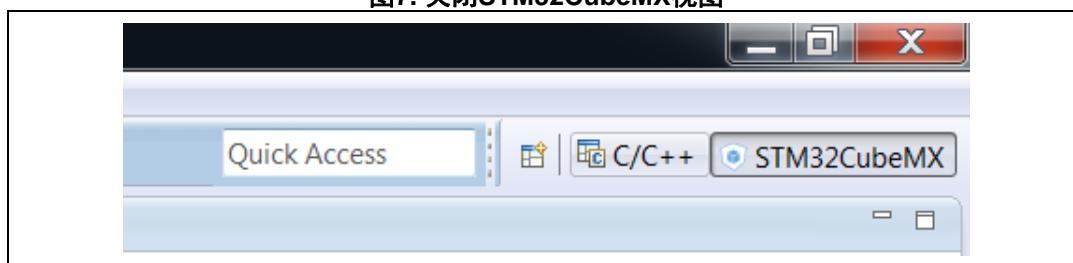
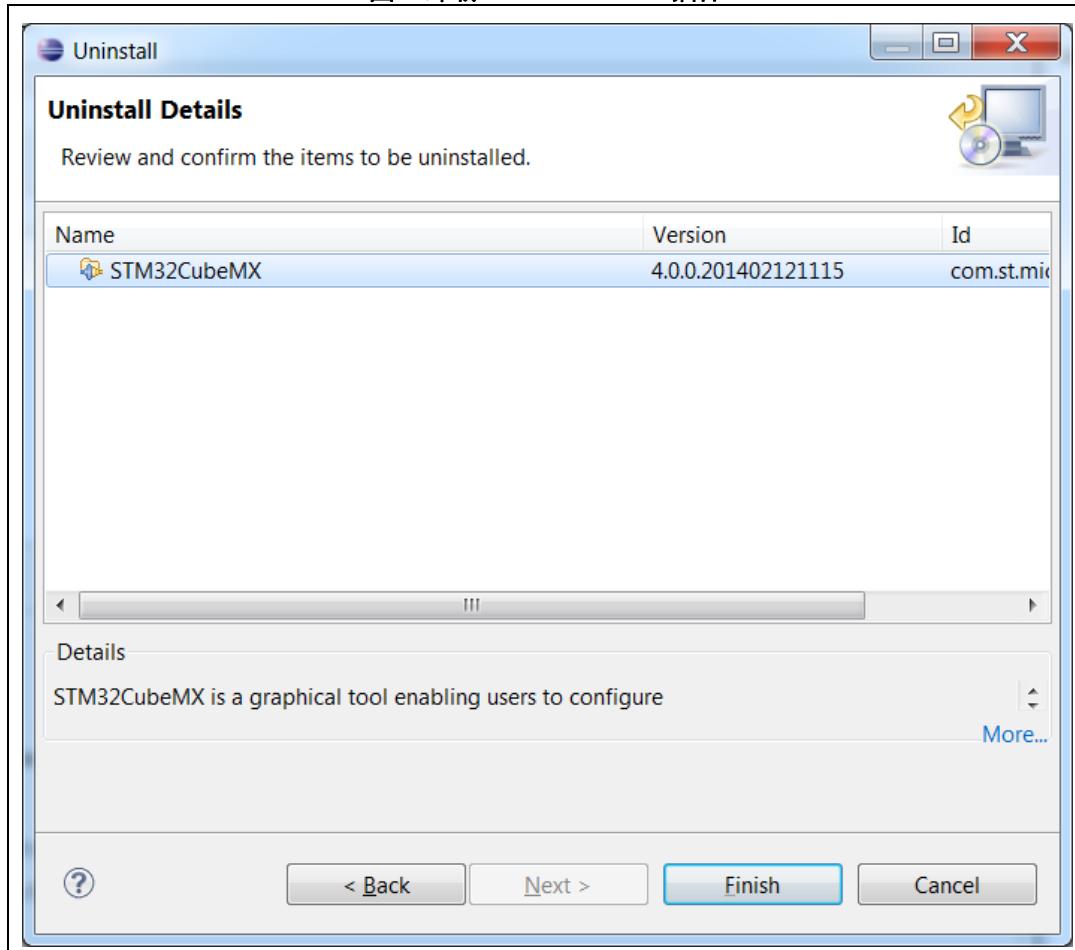


图8. 卸载STM32CubeMX插件



4.4 启动STM32CubeMX

4.4.1 STM32CubeMX 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。

4.4.2 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。

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。

参见[表 1](#)了解可用命令。

表1. 命令行摘要

命令行	目的	示例
help	此命令显示可用命令的列表	help
load <mcu>	此命令加载已选定的MCU	load STM32F101RCTx load STM32F101Z(F-G)Tx
config load <filename>	此命令加载先前保存的配置	config load C:\Cube\ccmram\ccmram.ioc
config save <filename>	此命令保存当前配置	config save C:\Cube\ccmram\ccmram.ioc
config saveext <filename>	此命令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>	此命令以新名称保存当前项目	config saveas C:\Cube\ccmram2\ccmram2.ioc
csv pinout <filename>	此命令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>	此命令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在0或1之间进行选择，0表示在main.c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。 此命令行允许为每个外设选择基于HAL和基于LL的代码生成。 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>	此命令仅生成“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>	此命令设置包含.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可用于代码生成。	set tpl_path C:\myTemplates\

表1. 命令行摘要（续）

命令行	目的	示例
set dest_path <path>	此命令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	此命令获取用户模板源文件夹的路径名	get tpl_path
get dest_path	此命令获取用户模板目标文件夹的路径名。	get dest_path
project toolchain <toolchain>	此命令指定待用于项目的工具链。然后，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>	此命令指定项目名称	project name ccmram
project path <path>	此命令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

4.4.3 从Eclipse IDE运行STM32CubeMX 插件

如要从Eclipse运行 STM32CubeMX 插件：

1. 启动Eclipse环境。
2. 打开Eclipse IDE之后，点击开启新的视图： 。
3. 选择STM32CubeMX，在视图中打开STM32CubeMX（参见 [图 9](#)）。
4. STM32CubeMX视图打开（参见 [图 10](#)）。通过“欢迎”菜单进入STM32CubeMX用户界面。

图9. 打开Eclipse插件

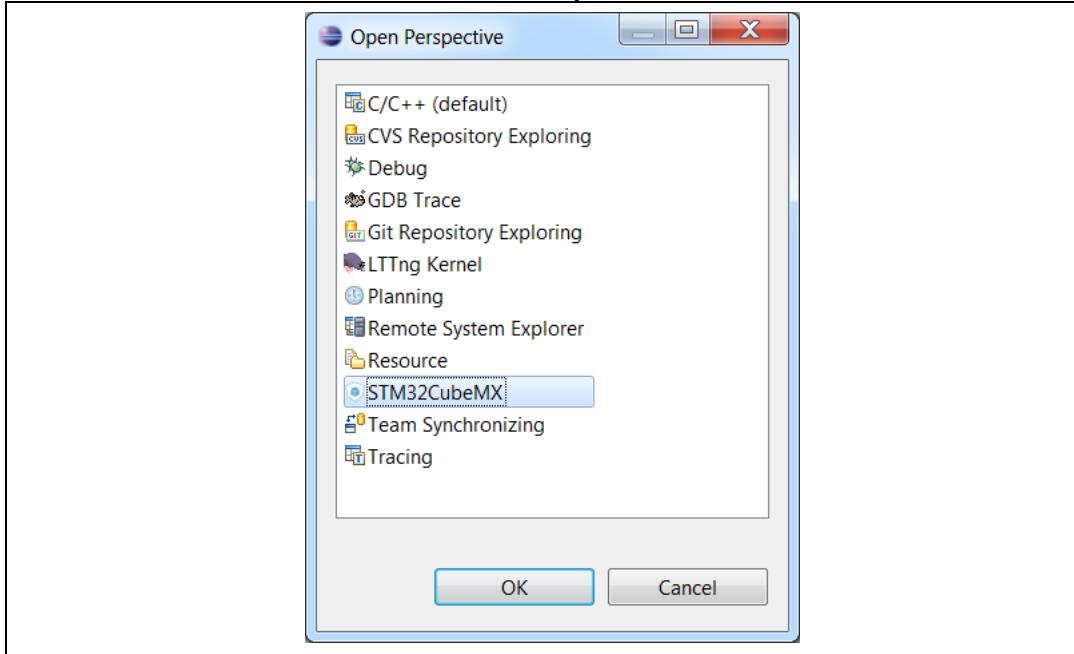
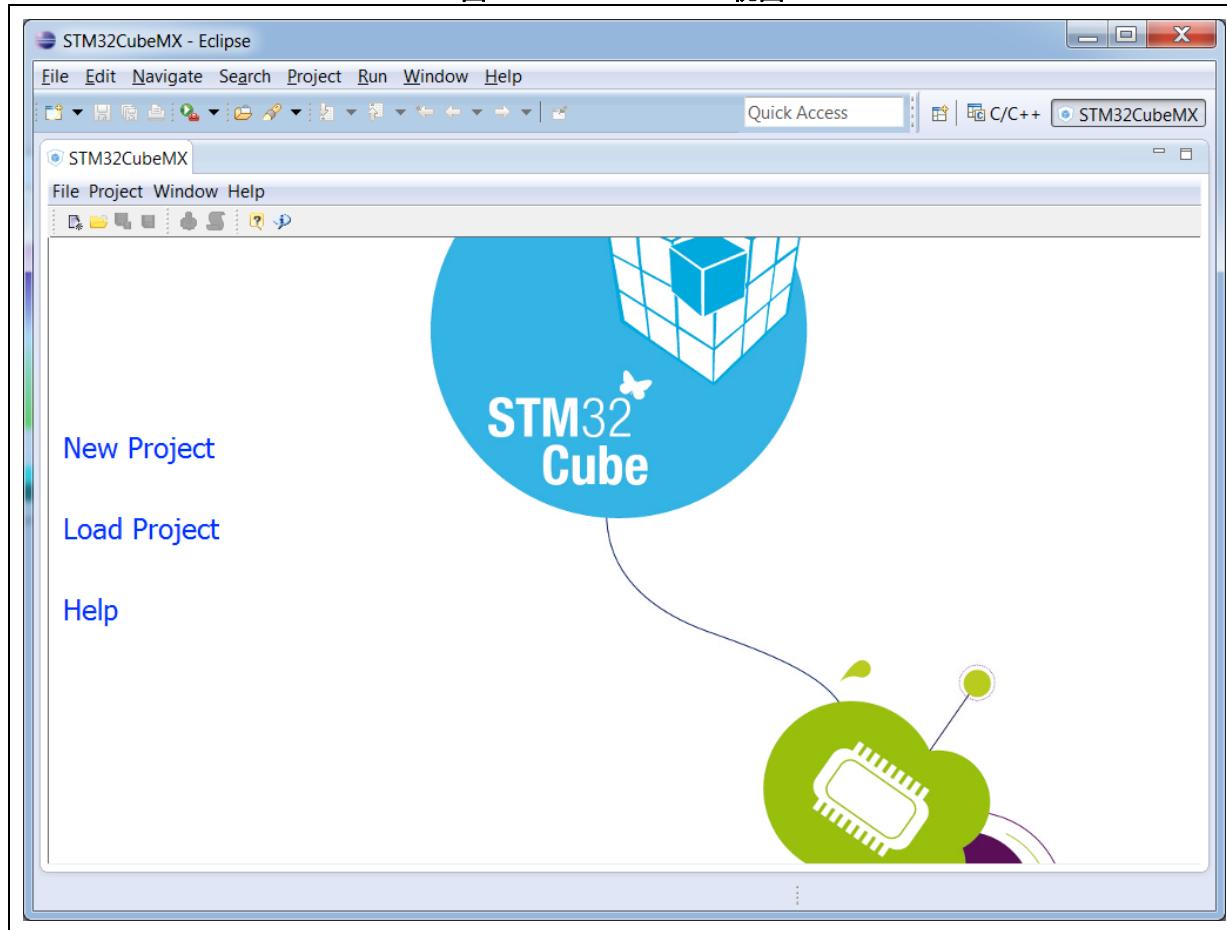


图10. STM32CubeMX 视图



4.5 获取STM32Cube和第三方软件发布和更新

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 的自身更新，
- 刷新 STM32 MCU 的描述和文档。

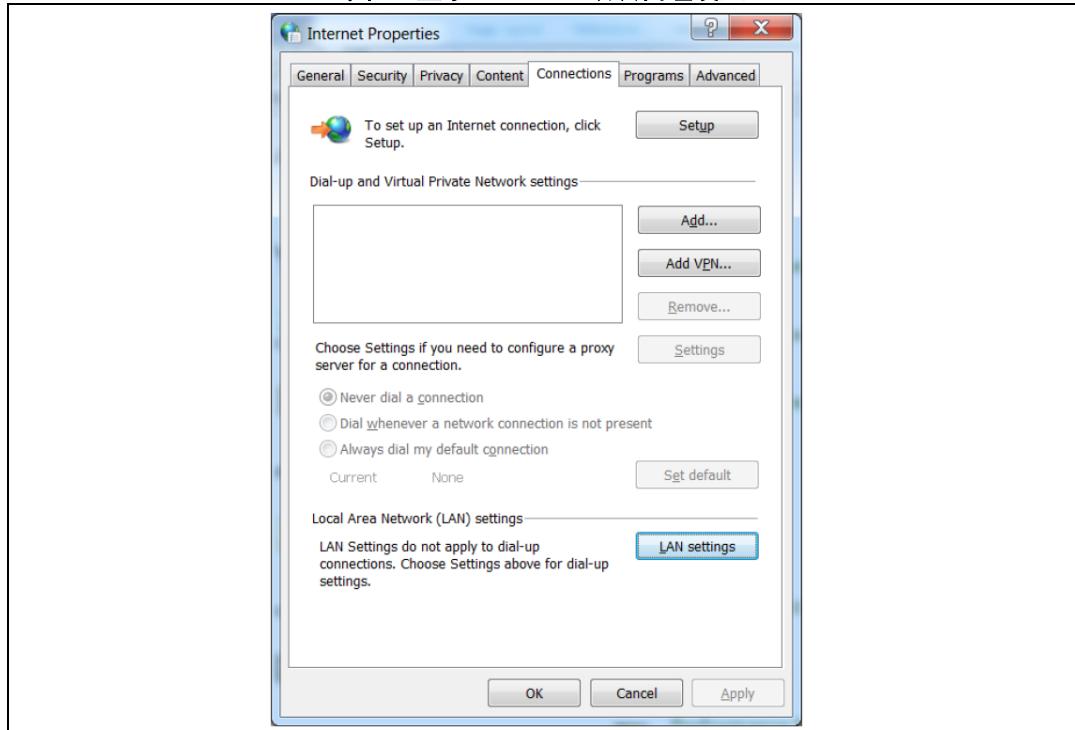
“帮助”菜单下提供了安装和更新相关的子菜单。

也可以在无法连接互联网的计算机上执行离线更新（参见图 16）。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，获取自身更新并下载固件包。有关该连接配置的描述，请参见第 4.5.1 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 11）。

图11. 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，用户将无法受益于更新机制，需要从 <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：

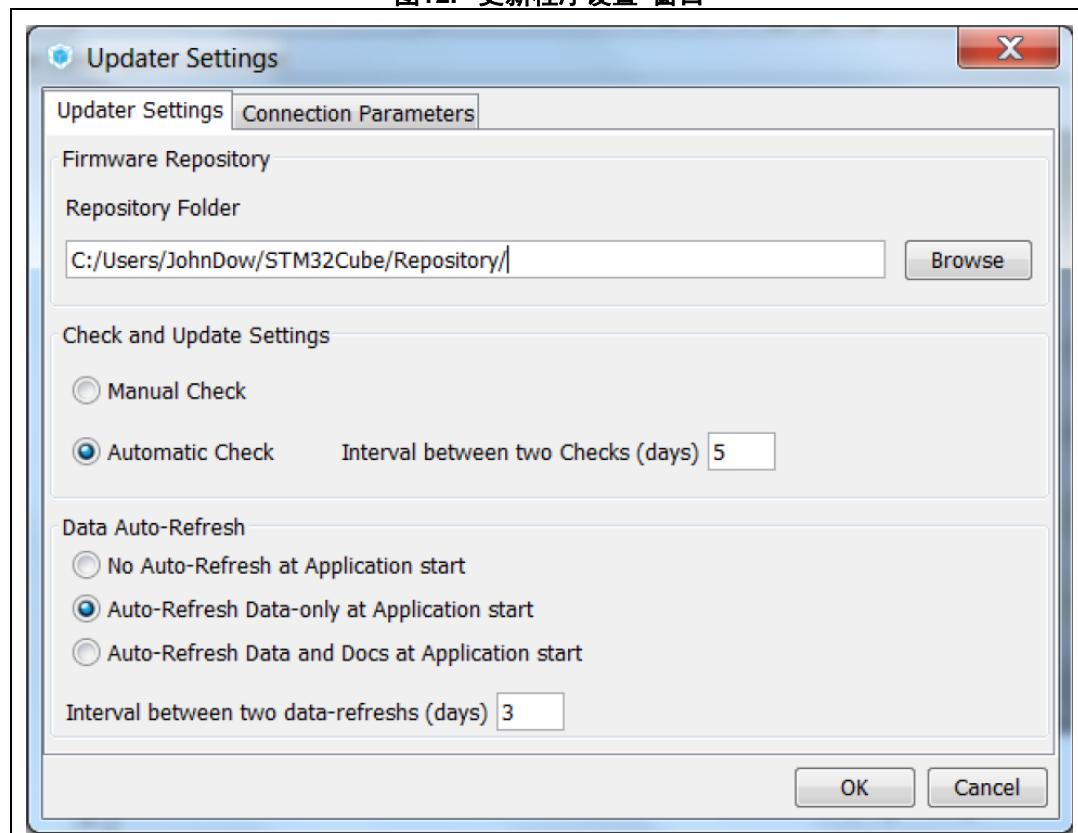
1. 访问 <http://www.st.com/stm32cube>，从“相关的软件”部分下载相关的STM32Cube MCU软件包。
2. 将zip包解压缩到STM32Cube存储库。在“更新程序设置”选项卡中找到默认存储库文件夹位置，如图 12 所示（您可能需要更新默认文件夹位置才能使用其他位置或名称）。

4.5.1 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 12）
 - 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12.“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：
 - 无代理（参见图 13）
 - 使用系统代理参数（参见图 14）
在Windows上，将从PC系统设置中获取代理参数。

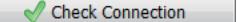
- 如果使用无需登录名/密码的代理服务器配置，请取消勾选“需要身份验证”。
- 手动配置代理服务器（参见 [图 15](#)）
输入代理服务器的http地址和端口号。如果使用无需登录名/密码的代理服务器配置，请输入登录名/密码信息，或取消勾选“需要身份验证”。
4. 取消勾选“**记住我的凭据**”，防止STM32CubeMX将加密的登录名/密码信息保存在文件中。这意味着每次启动STM32CubeMX时都要重新输入登录名/密码信息。
 5. 点击“检查连接”按钮，验证连接是否有效。出现绿色复选标记，确认连接正常运行


图13. “连接参数”选项卡 - 无代理

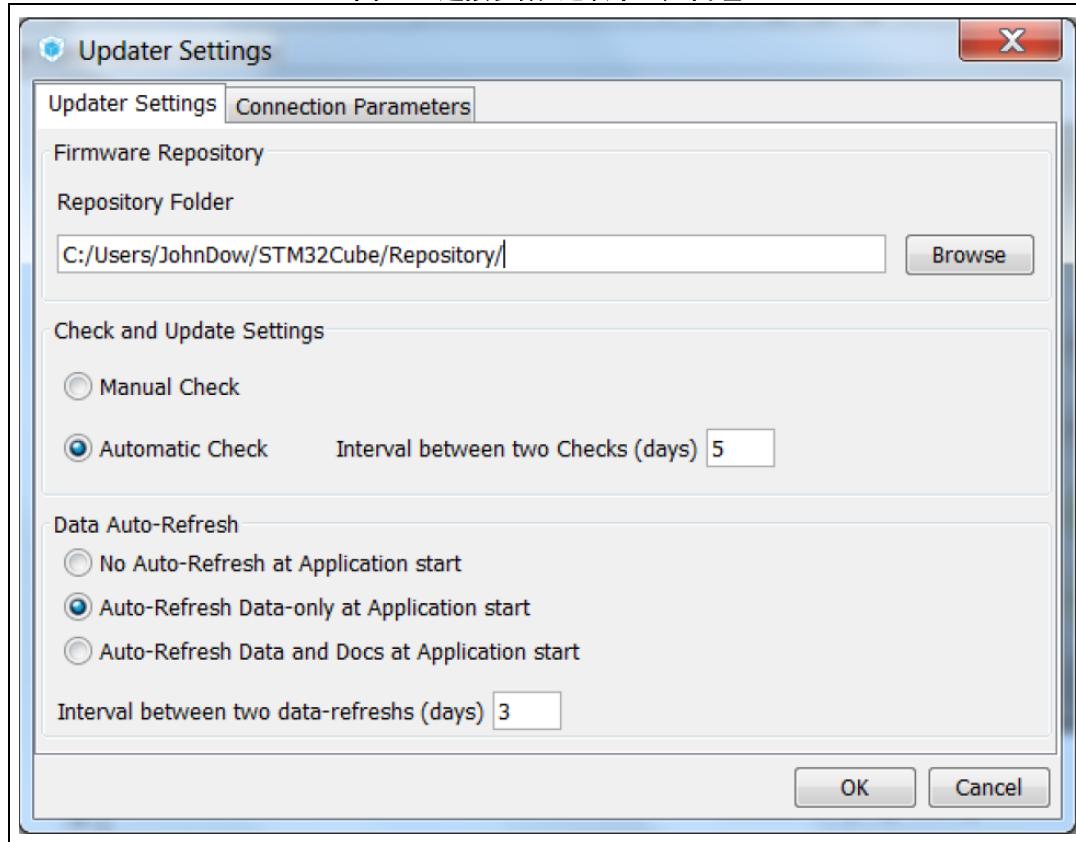


图14. “连接参数”选项卡 - 使用系统代理参数

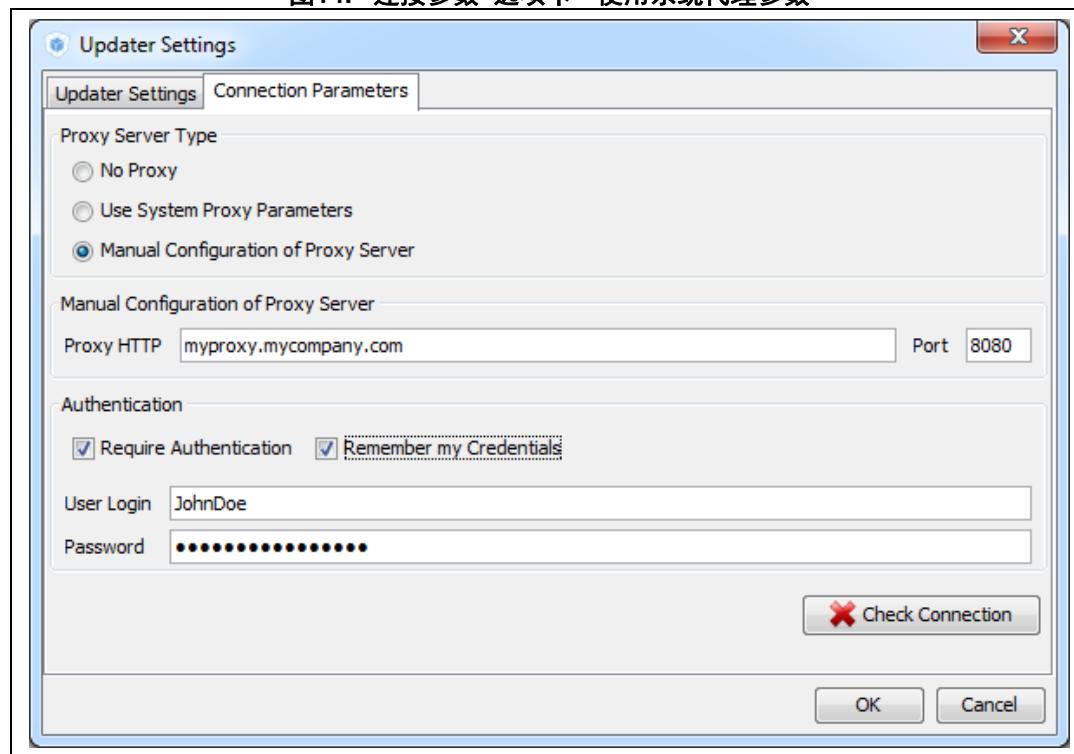
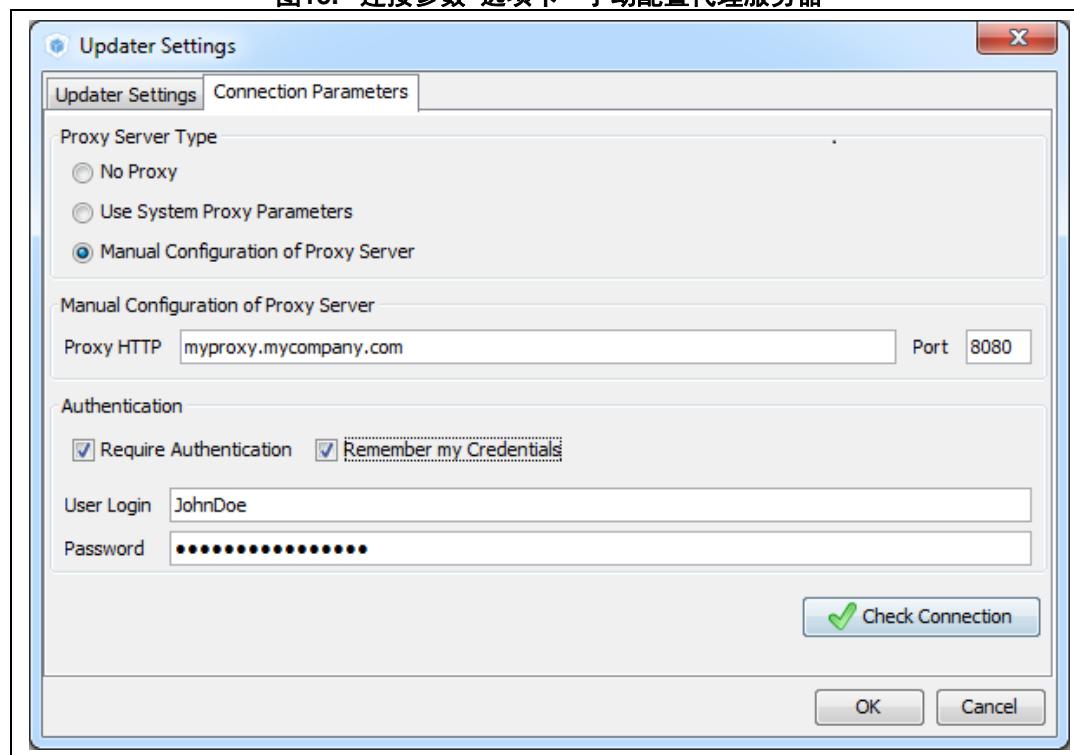


图15. “连接参数”选项卡 - 手动配置代理服务器



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。

4.5.2 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 16）。

“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<http://www.st.com>提供的最新版本。

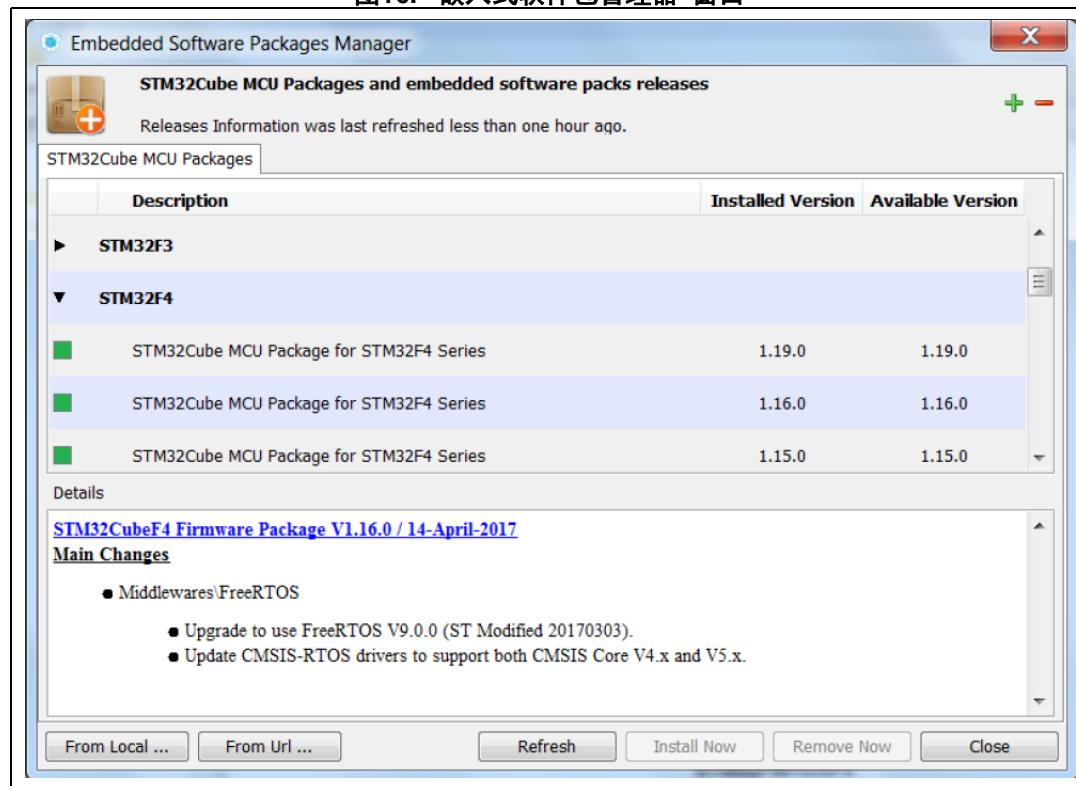
如果此时无法访问互联网，请选择“本地文件”。接下来，浏览并选择先前从st.com下载的所需STM32Cube MCU软件包的zip文件。对文件执行完整性检查，确保STM32CubeMX完全支持该文件。

当安装的版本与<http://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 16。

图16. “嵌入式软件包管理器”窗口



4.5.3 安装STM32 MCU软件包补丁

使用第 4.5.2 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。

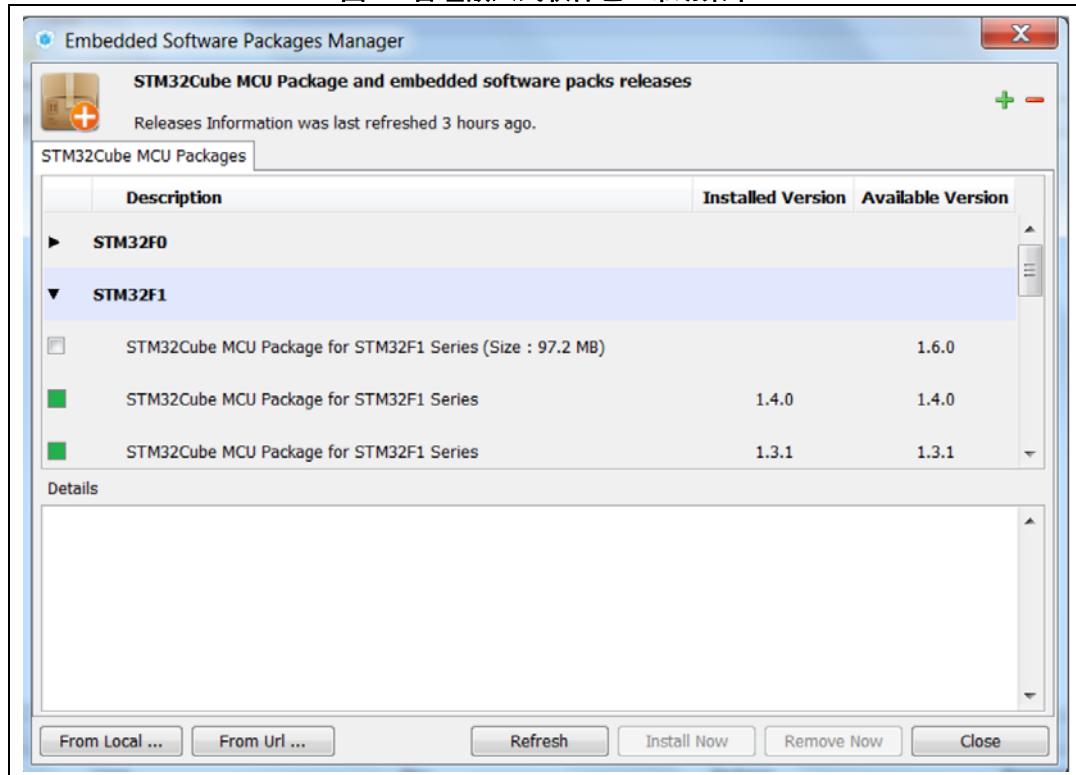
4.5.4 安装嵌入式软件包

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 17）。

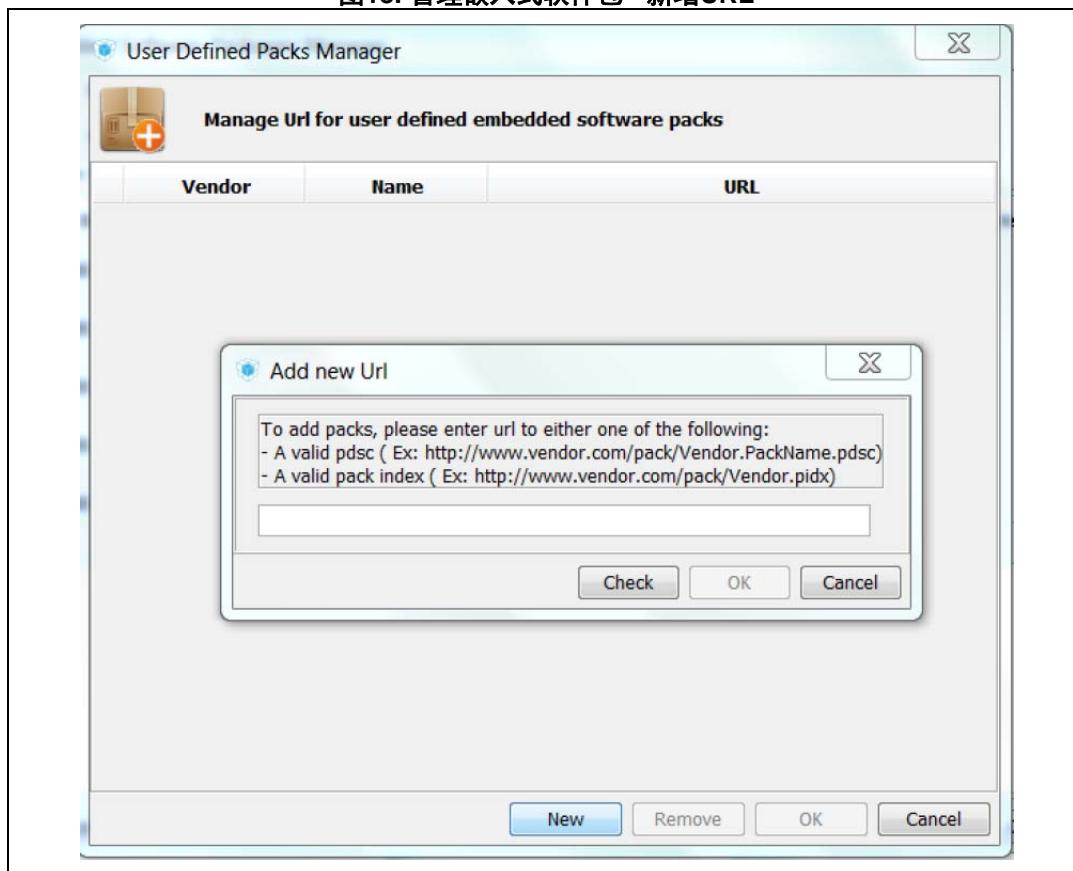
使用展开/折叠按钮 ，展开或折叠软件包列表。

图17. 管理嵌入式软件包 - 帮助菜单



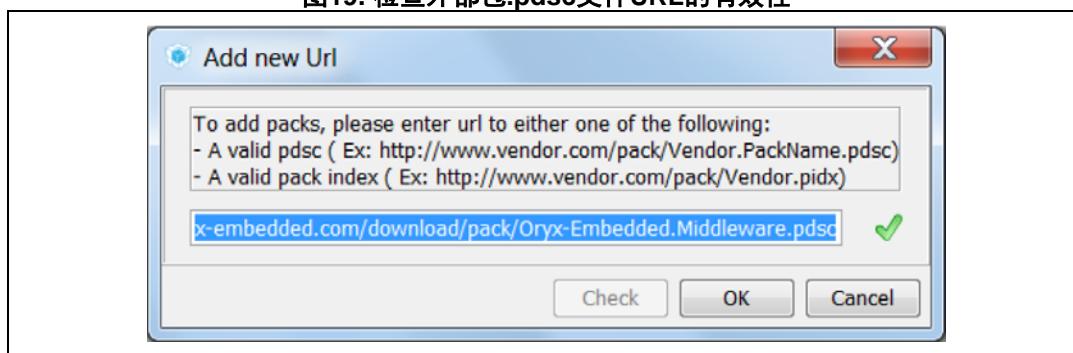
2. 点击“从本地...”按钮，浏览计算机文件系统，并选择嵌入式软件包。STM32Cube MCU软件包以zip档案形式提供，嵌入式软件包以.pack档案形式提供。
在下列情况下需要执行此操作：
 - 无法访问互联网，但可在本地计算机上使用嵌入式软件包。
 - 嵌入式软件包不公开，因此无法在互联网上使用。对于此类软件包，STM32CubeMX 无法检测并建议更新。
3. 点击“从URL...”按钮，为软件包.pdsc文件或外部包索引(.pidx)指定互联网下载位置。
按照以下步骤继续：
 - a) 选“从URL...”，点击“新建”（参见图 18）。
 - b) 指定.pdsc文件URL。例如，Oryx嵌入式中间件包的URL是https://www.oryx-embedded.com/download/pack/Oryx-Embedded.Middleware.pdsc（参见图 19）。

图18. 管理嵌入式软件包 - 新增URL



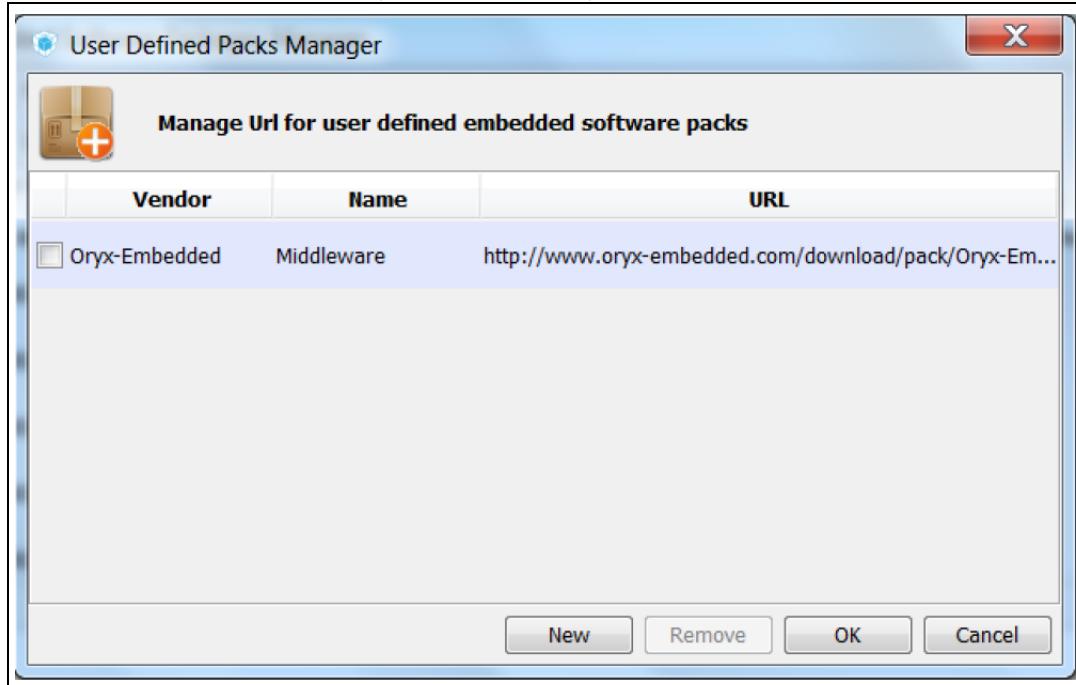
- c) 点击“检查”按钮，验证所提供的URL是否有效（参见图 19）。

图19. 检查外部包.pdsc文件URL的有效性



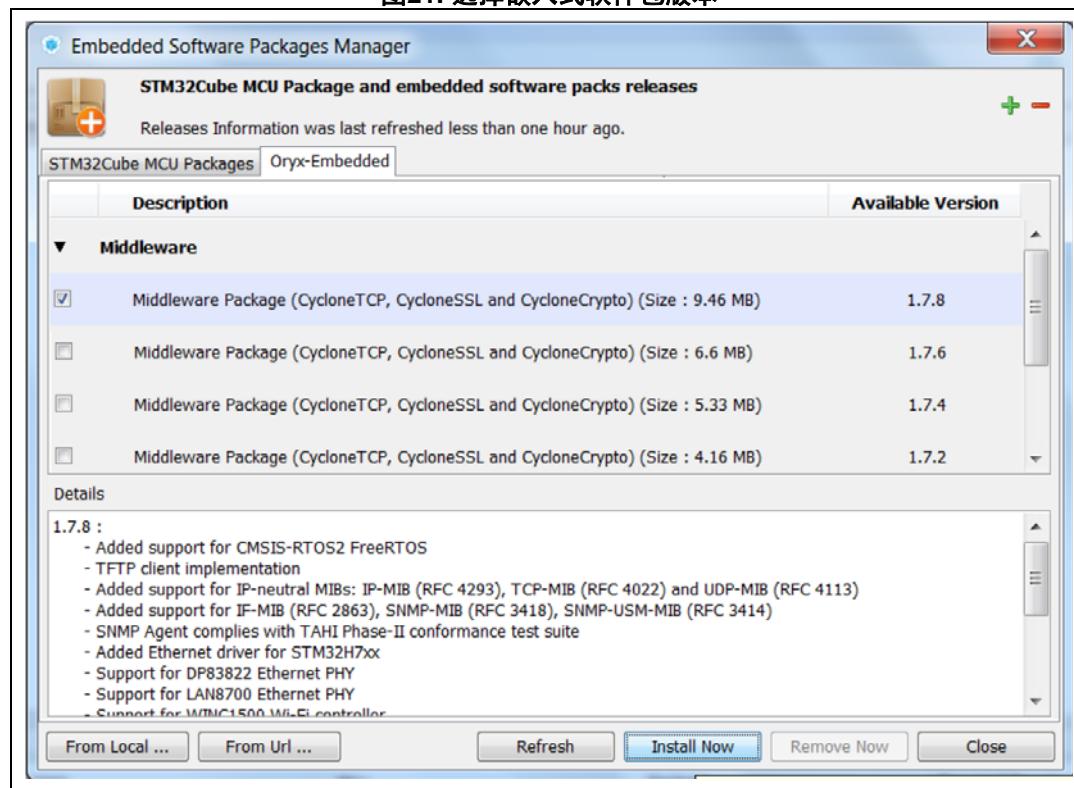
- d) 点击“确定”。现在可在用户定义的包列表中找到包pdsc信息（参见图 20）。
要从列表中删除URL，选中URL复选框，然后点击“删除”。

图20. 用户定义的软件包列表



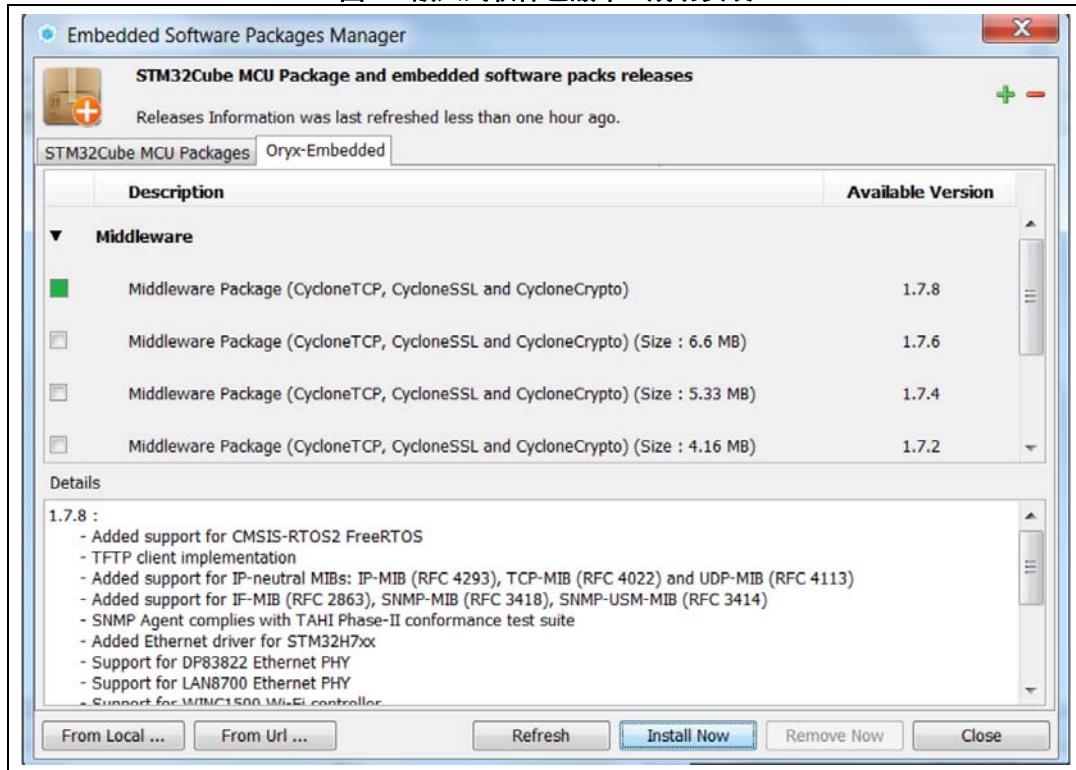
- e) 点击“确定”，关闭窗口并开始获取psdc信息。成功完成后，可用的包版本将显示在可安装的库列表中。使用相应的复选框，选择给定的版本（参见图 21）。

图21. 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。安装成功后，复选框变为绿色（参见 [图 22](#)）。
- 然后，用户可以将此包中的软件组件添加到其项目中。

图22. 嵌入式软件包版本 - 成功安装



4.5.5 删除已安装的嵌入式软件包

请按以下步骤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”。
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。

有关示例，请参见[图 23至图 25](#)。

图23. 删除库

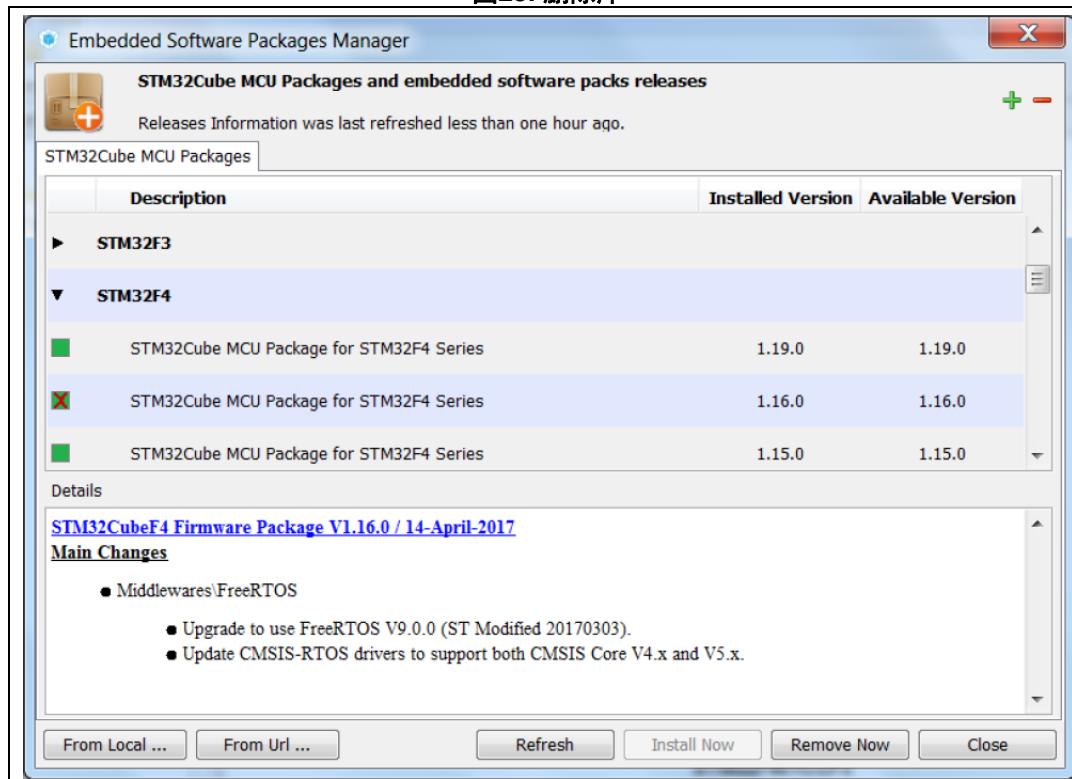


图24. 删除库确认消息

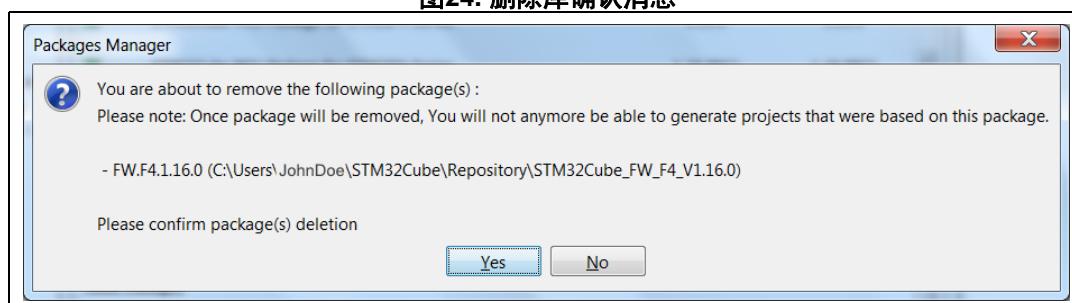
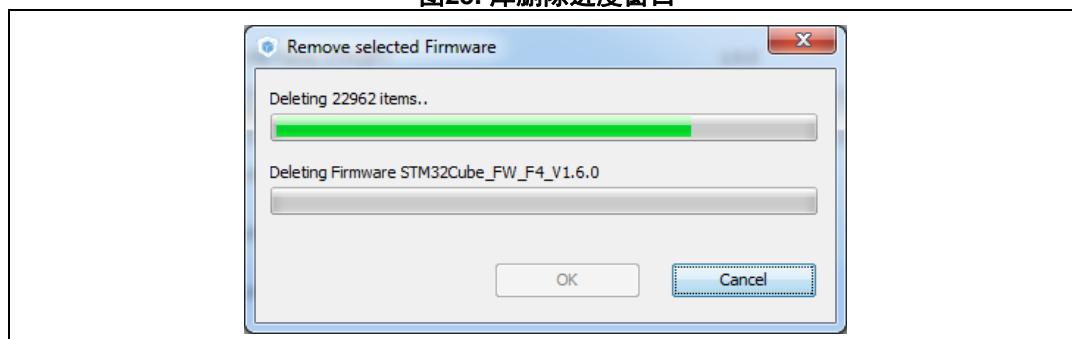


图25. 库删除进度窗口



4.5.6 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（参见图 26）。

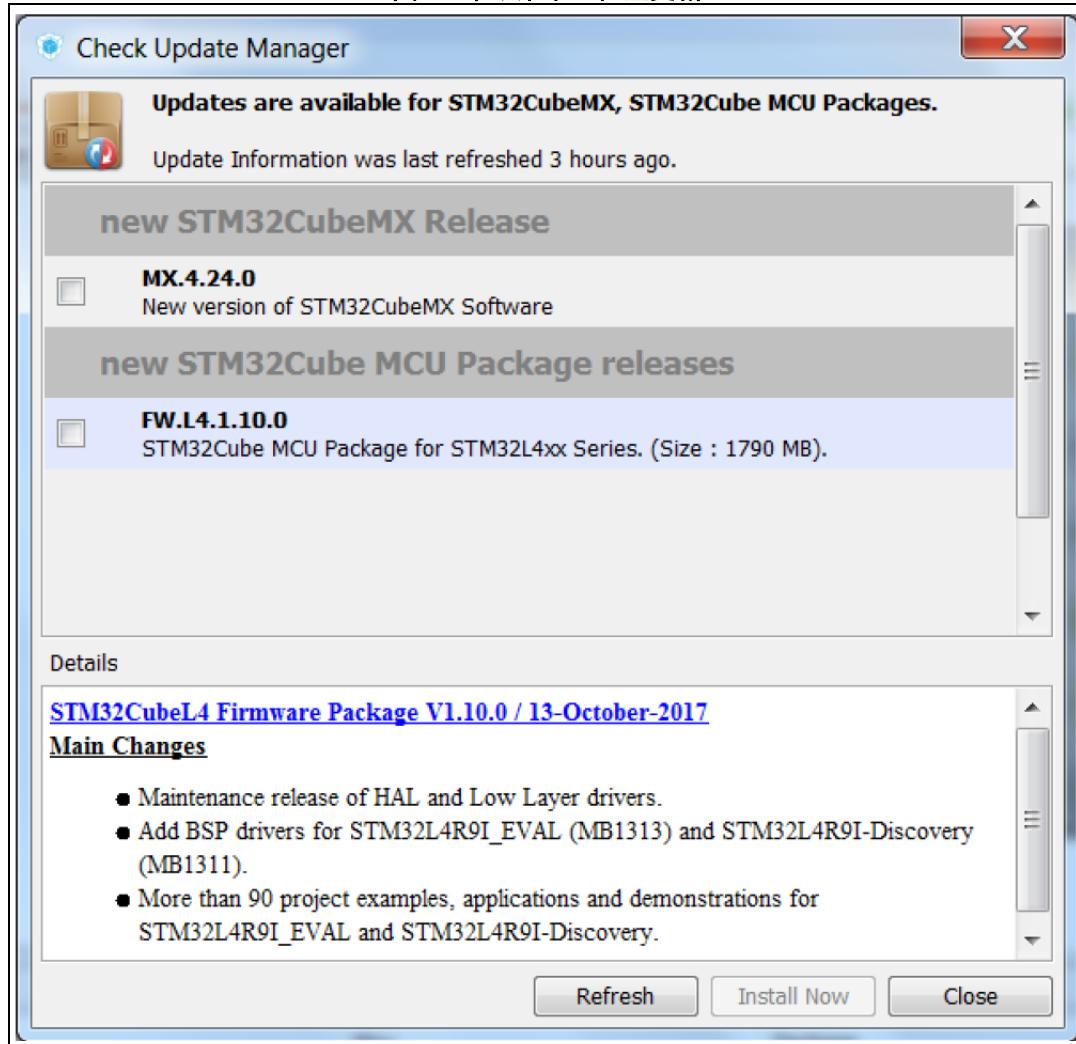
当更新程序配置为自动检查时，它会定期验证更新是否可用。

此时，工具栏上会出现绿色箭头图标 。

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26. 帮助菜单：检查更新



5 STM32CubeMX用户界面

STM32CubeMX用户界面包含以下内容：一、一个欢迎页面，用户可以在该页面决定启动一个新项目或加载一个近期项目；二、一个新项目窗口，用户应在该窗口中选择要用于项目的开发板或微控制器料号；三、项目视图及其主窗口、菜单栏、工具栏、四个配置视图（引脚布局、配置、时钟配置、功耗计算器）和一组帮助窗口（MCU选择、更新管理器、关于）。以下各节介绍了所有这些用户界面组件。

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. 从“引脚布局”视图中选择相关特性（外设、中间件）及其工作模式。
2. 在时钟配置视图中配置时钟树。
在“引脚布局”视图中，通过使能外部时钟、主输出时钟、音频输入时钟（与您的应用相关）来配置RCC外设。这会在“时钟树”视图上自动显示更多选项（参见
[图 34](#)）。
3. 在配置视图中，配置初始化外设和中间件工作模式所需的参数。
4. 生成初始化C代码。

5.1 欢迎页面

欢迎页面是启动STM32CubeMX程序时打开的第一个窗口。只要应用程序正在运行，它就会保持打开状态。关闭该页面即会关闭应用程序。有关“欢迎”页面的说明，请参见[图 27](#)和[表 2](#)。

图27. STM32CubeMX欢迎页面

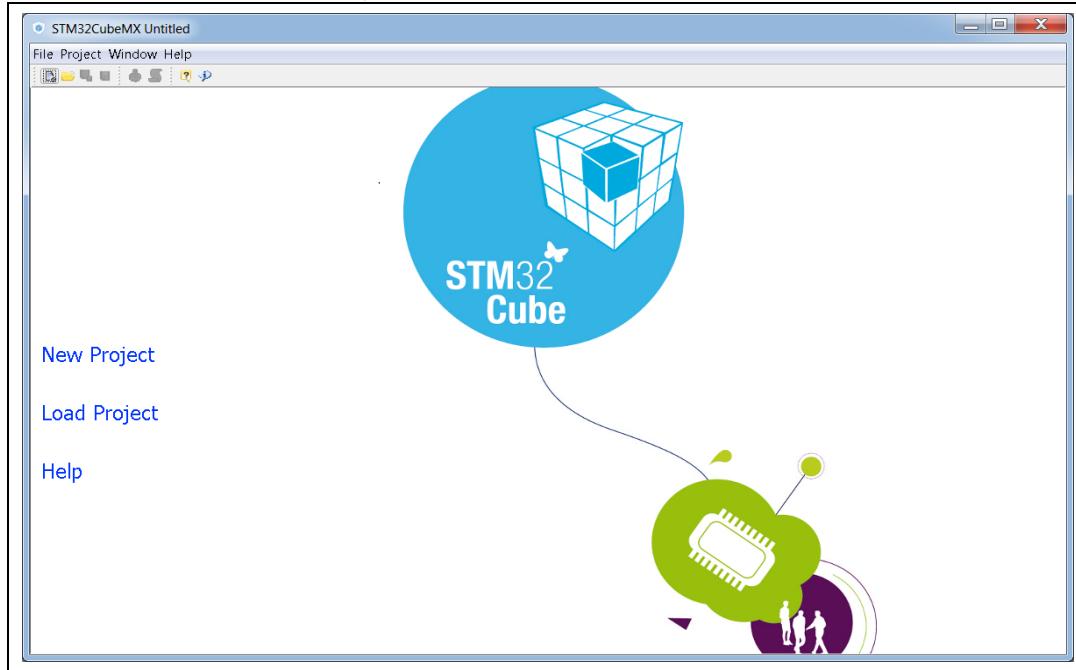


表2. 欢迎页面快捷方式

名称	说明
新项目	此快捷方式通过打开“新项目”窗口来启动STM32CubeMX新项目创建（从“MCU选择器”选项卡中选择MCU或从“板选择器”选项卡中选择板配置）。
加载项目	此快捷方式打开一个浏览器窗口，用于选择并加载先前保存的配置（.ioc文件）。 加载使用旧STM32CubeMX版本创建的项目时，用户可以选择迁移，以将其迁移到最新的STM32CubeMX可用数据库和STM32Cube固件版本，也可以选择继续。 注意： 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STM32CubeMX4.17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。 注意： 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）。
帮助	此快捷方式可打开用户手册。

5.2 新项目窗口

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。

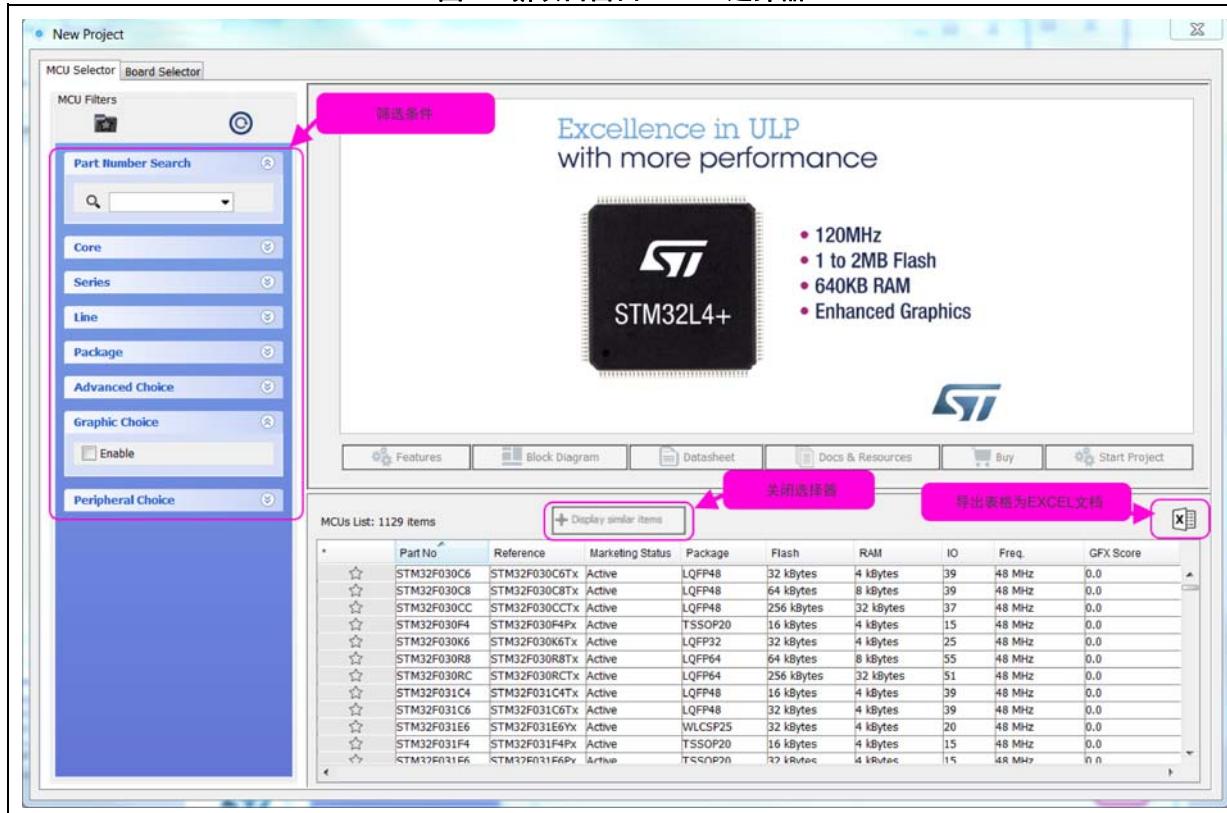
此窗口显示两个选项卡供您选择：

- “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。

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、其他MCU特性，如价格、存储器容量或I/O数量（参见图 28）以及MCU图形功能。

图28. 新项目窗口 - MCU选择器



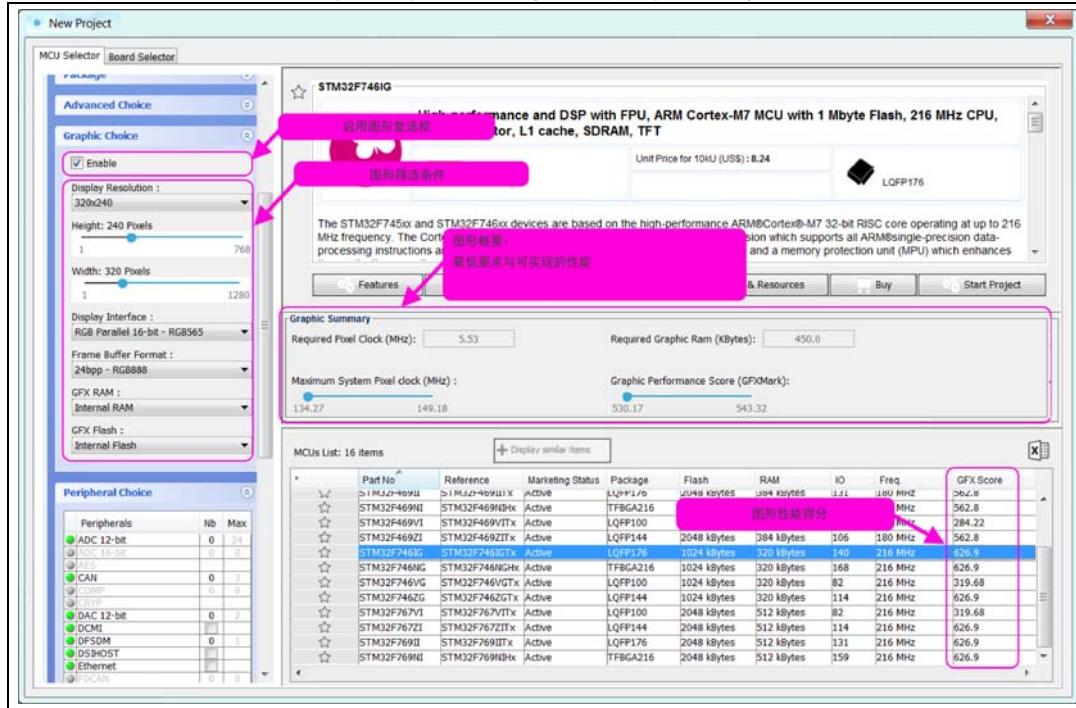
基于图形条件选择MCU

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 29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29. 在MCU选择器中使能图形选择



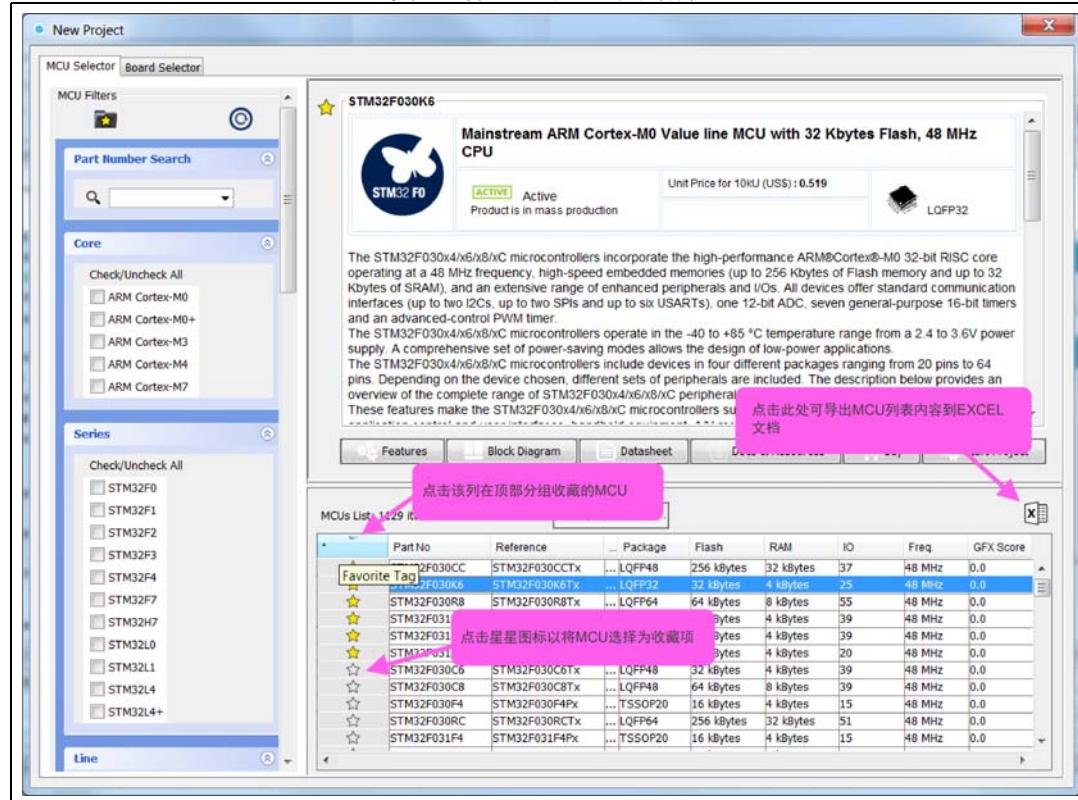
导出至Excel功能

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 30。

图30. 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 31）。点击“显示类似项目”按钮，显示这些相近特性（参见图 32）：默认情况下，MCU首先按匹配度排序，然后按料号排序。对于相近的MCU（匹配度低于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31. 新项目窗口 - 具有相近MCU特性的MCU列表

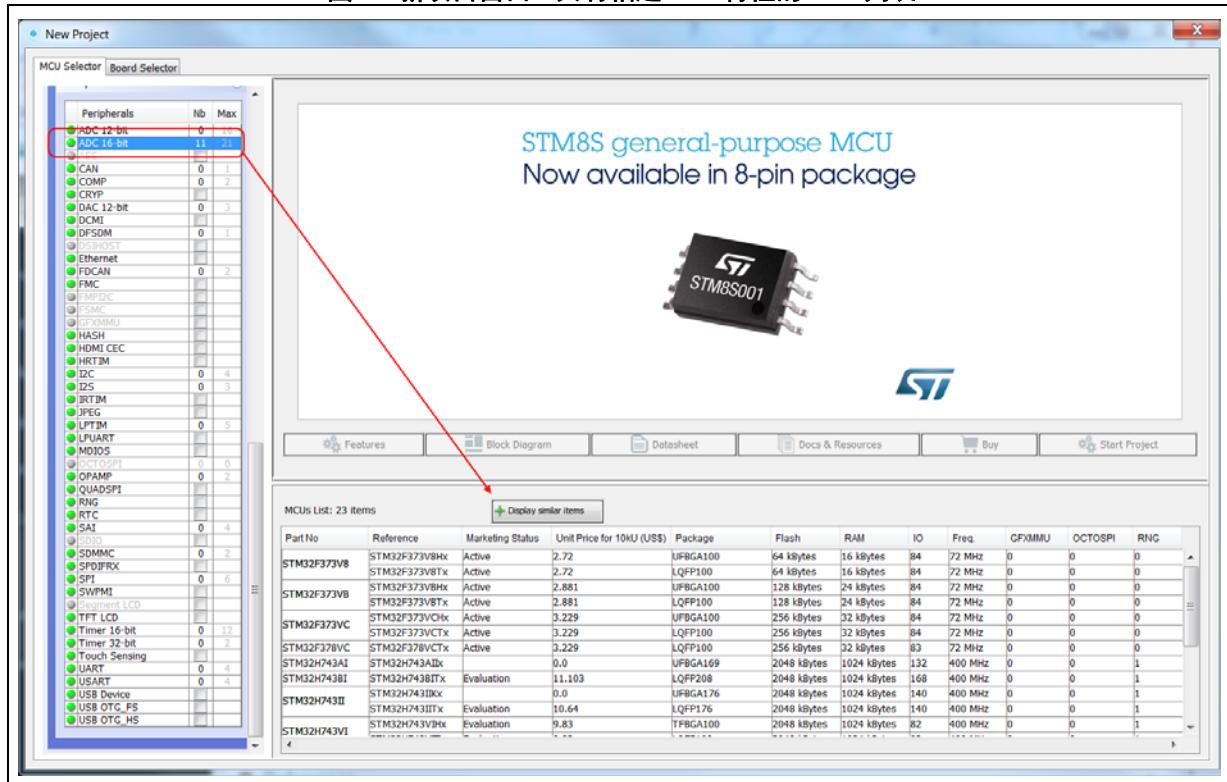
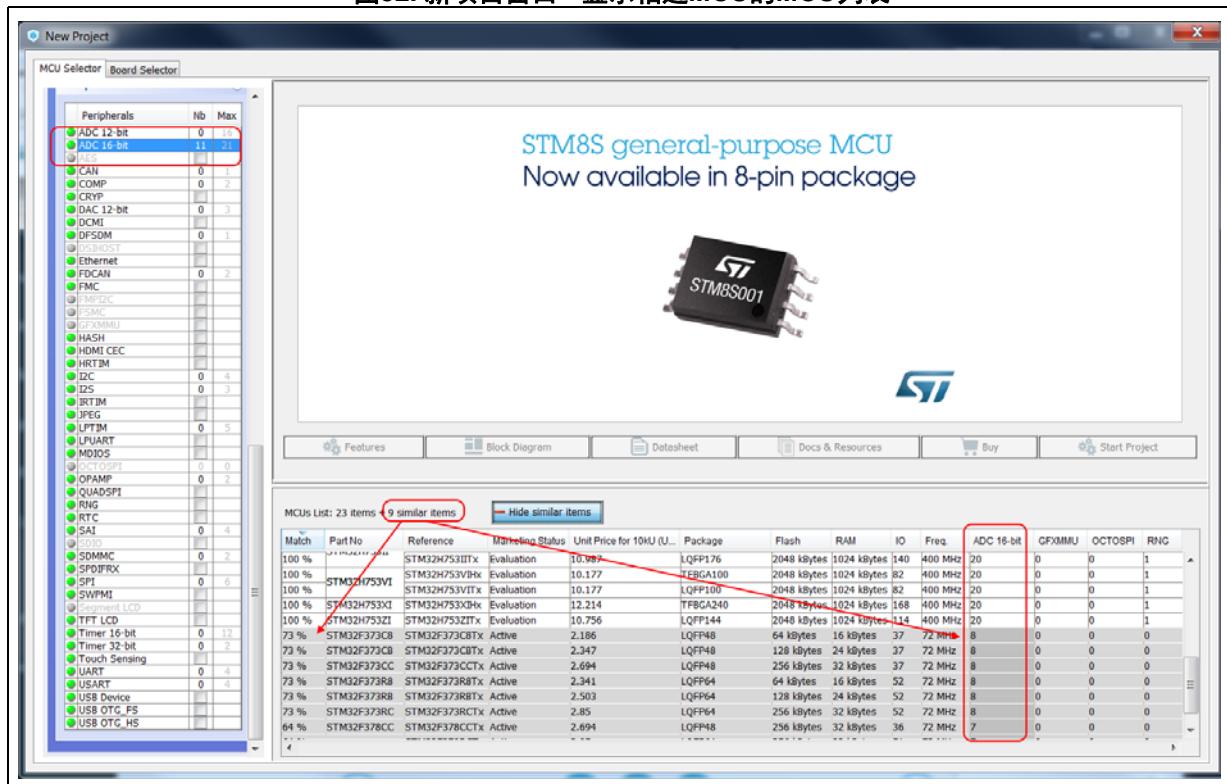


图32. 新项目窗口 - 显示相近MCU的MCU列表



注:

为每个用户选择的标准计算匹配百分比，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将达到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，将所有标准比率取平均值，得出“匹配”列百分比值。

“板选择器”允许对STM32板类型、系列和外设进行筛选（参见图 33）。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（参见图 35）。或者，用户可以选择使用默认外设模式对其进行初始化（参见图 36）。

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。

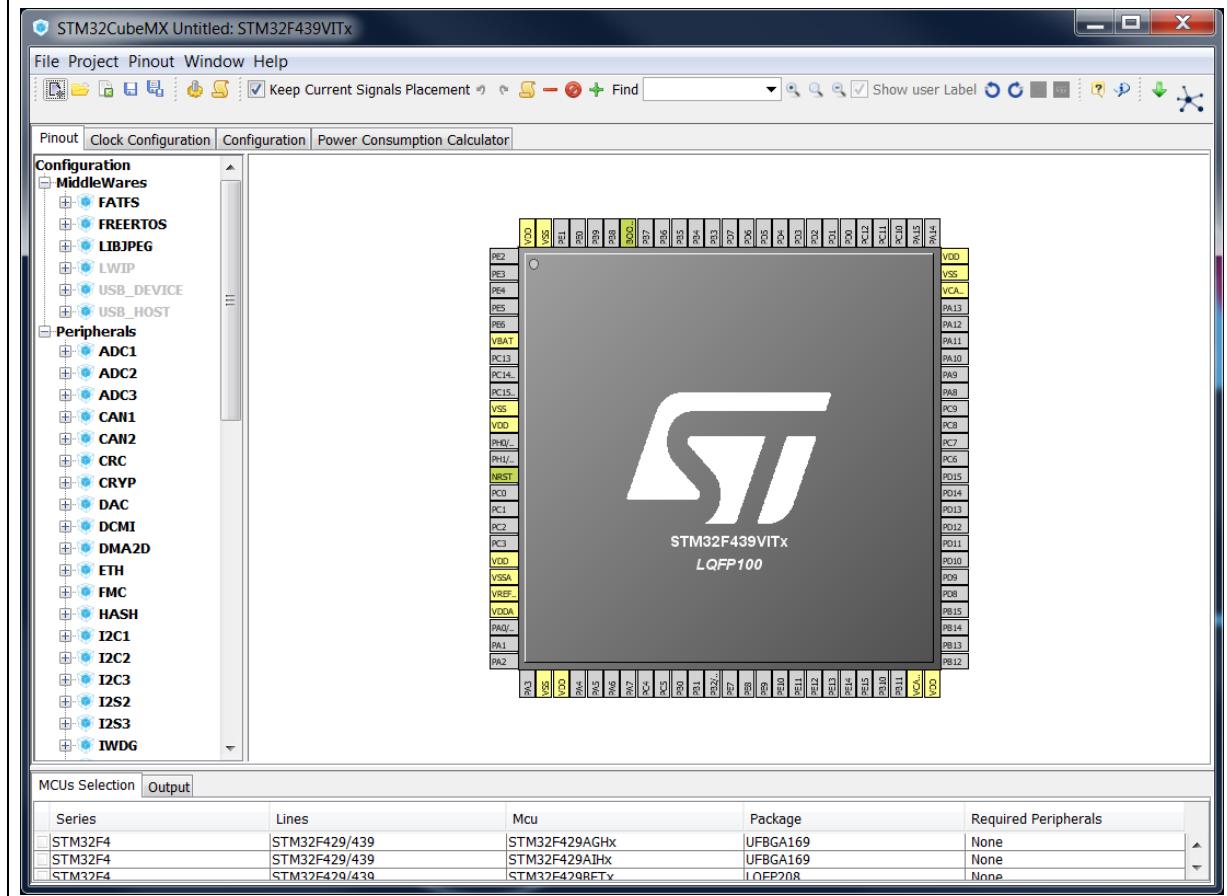
图33. 新项目窗口 - 板选择器



5.3 主窗口

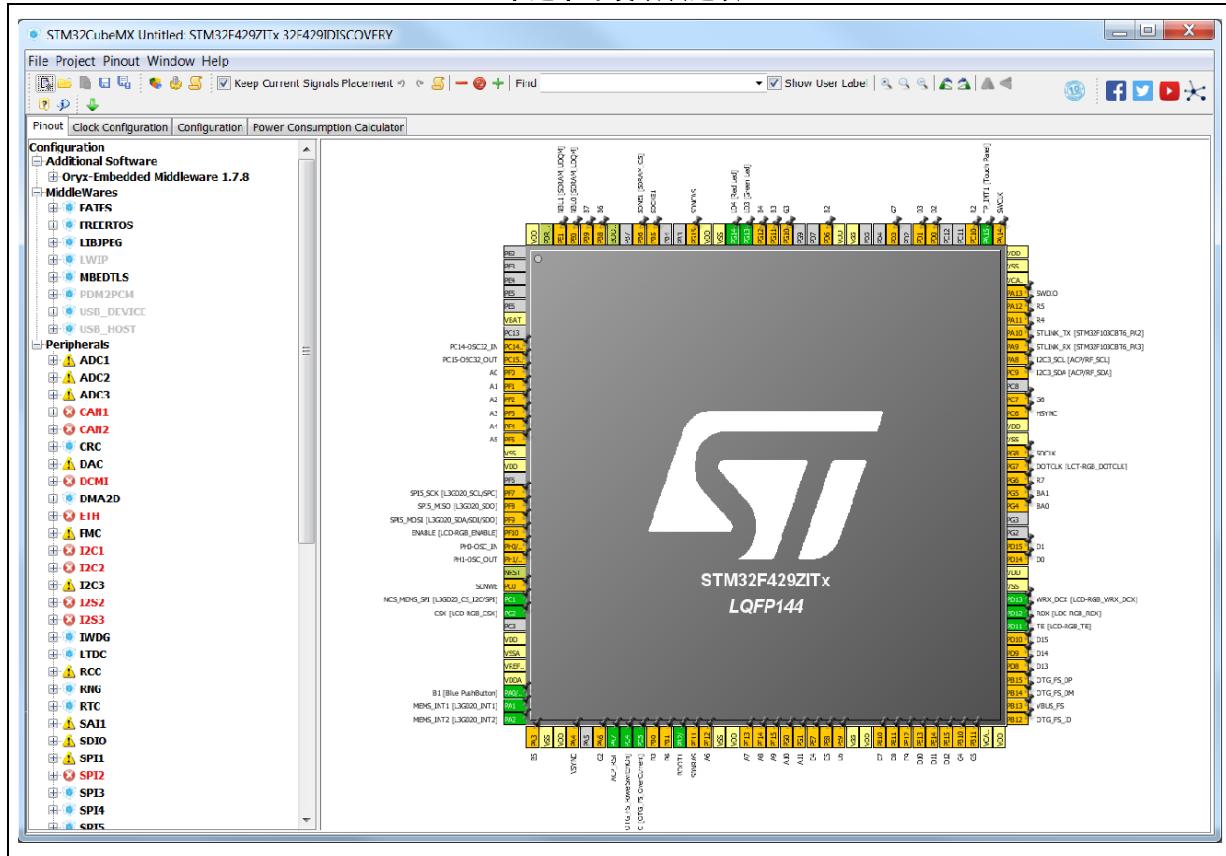
一旦STM32料号或开发板选择了，或者加载了之前保存的项目，主窗口将显示STM32CubeMX所有组件和菜单（参见图 34）。有关工具栏和菜单的详细说明，请参见第 5.3 节。

图34. 选择MCU时显示的STM32CubeMX主窗口



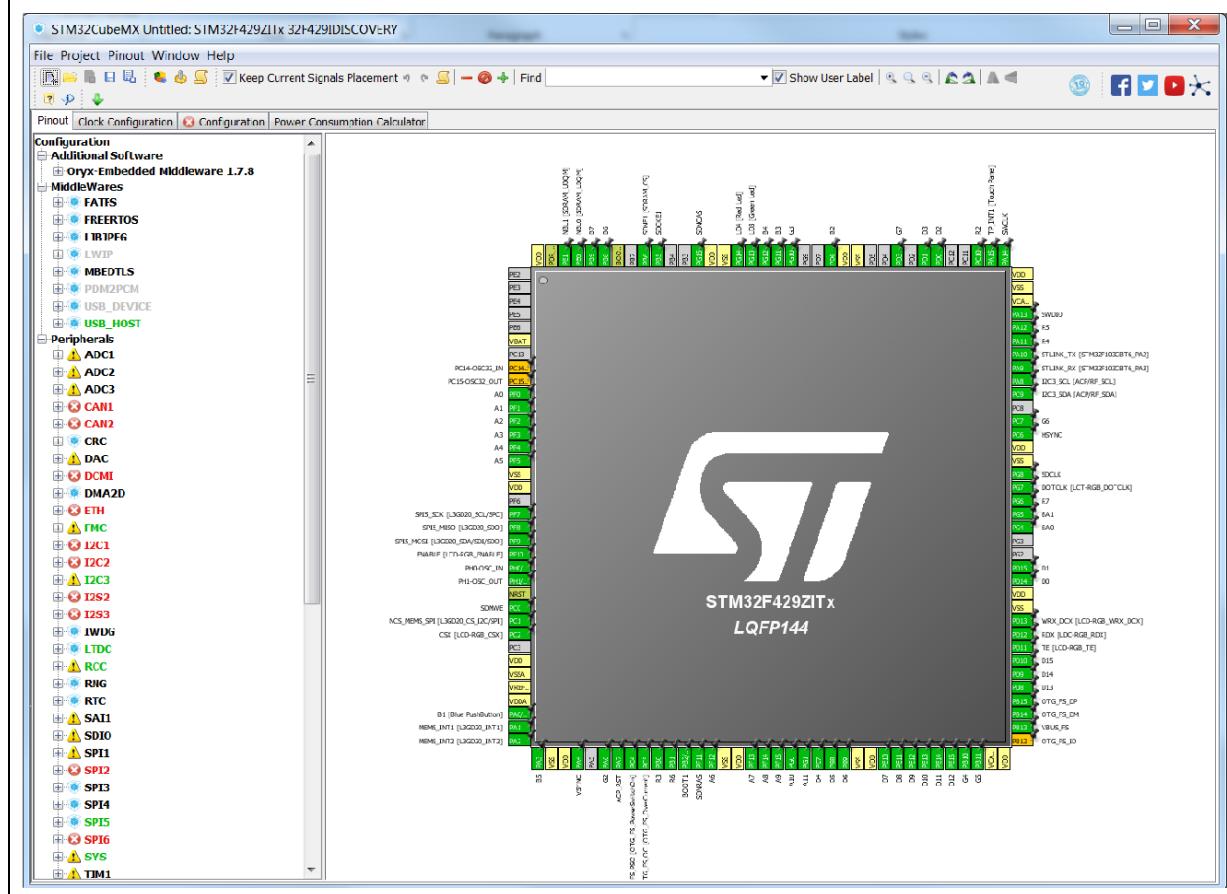
选择一个板，在对话窗口要求将所有外设初始化为其默认模式时回答“否”，从而自动设置该开发板的引脚布局。但是，只有设置为GPIO的引脚才会被标记为已配置，即以绿色突出显示，而未设置外设模式。接下来，用户可以从外设树中手动选择其应用所需的外设模式（参见图 35）。

**图35. 选择板时显示的STM32CubeMX主窗口
(未选中外设默认选项)**



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将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 36）。

**图36. 选择板时显示的STM32CubeMX主窗口
(选中外设默认选项)**



5.4 工具栏和菜单

STM32CubeMX菜单栏中提供以下菜单：

- 文件菜单
- 项目菜单
- 引脚布局菜单（仅当选中引脚布局视图时显示）
- 窗口菜单
- 帮助菜单

以下章节将介绍STM32CubeMX工具栏和菜单。

5.4.1 文件菜单

有关文件菜单和图标的说明，请参见表 3。

表3. 文件菜单功能

图标	名称	说明
	新项目	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STMicroelectronics板。
	加载项目...	选择STM32CubeMX配置.ioc文件，加载已有的STM32CubeMX项目配置。 注意： 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）。
	导入项目...	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。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。
	项目另存为 ...	将当前项目配置（引脚布局、时钟树、外设、中间件、功耗计算器）另存为新项目。此操作将创建一个.ioc文件，该文件由用户定义名称，位于目标文件夹中。
	保存项目	保存当前项目。
无图标	关闭项目	关闭当前项目并切换回欢迎页面。
无图标	近期项目 >	显示最近保存的五个项目的列表
无图标	退出	如有必要，建议保存项目，然后关闭应用程序。

5.4.2 项目菜单

有关项目菜单和图标描述, 请参见表 4。

表4. 项目菜单

图标	名称	说明
	生成代码	该菜单为当前配置（引脚布局、时钟、外设和中间件）生成C初始化C代码。如果先前尚未定义这些配置，请打开项目设置窗口。 注：升级到STM32CubeMX新版本时，建议备份当前项目。用户将收到迁移至新固件库版本的提示（若可用）。选择“继续”，保持先前使用的版本。
	生成报告 ⁽¹⁾	此菜单将当前项目配置生成为pdf文件和文本文件。
	设置	此菜单打开项目设置窗口，用于配置项目名称、文件夹、选择工具链和C代码生成选项
	选择附加软件组件	此菜单可用于选择嵌入式软件包提供的软件组件。用户可以通过选择“帮助”菜单下的“管理嵌入式软件包”来安装此类软件包。

- 如果先前保存了项目，则会在与项目配置.ioc文件相同的位置生成报告。否则，用户可以选择目标文件夹，以及是否将项目配置另存为.ioc文件。

5.4.3 引脚布局菜单

仅当选择了“引脚布局”选项卡时，“引脚布局”菜单和子菜单快捷方式方可使用（参见图 37）。否则它们处于隐藏模式（参见图 38）。有关引脚布局菜单和图标描述，请参见表 5。

图37. 引脚布局菜单（已选择引脚布局选项卡）

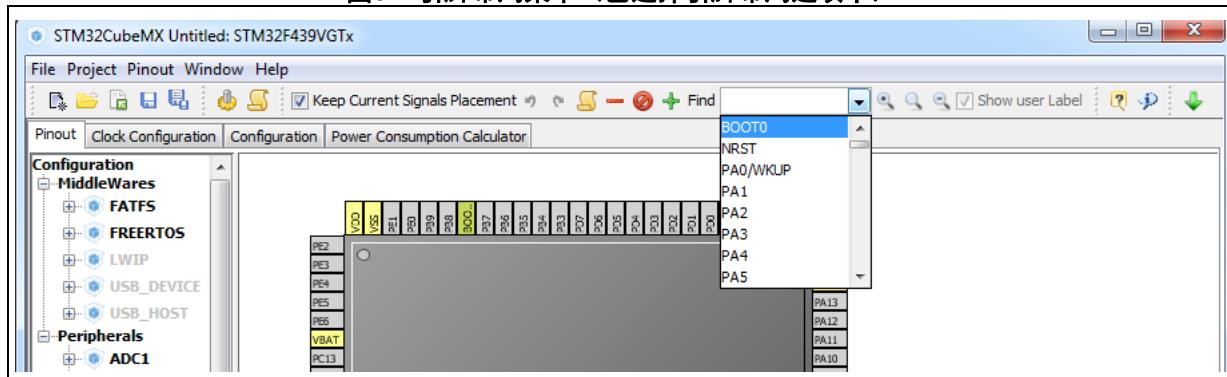


图38. 引脚布局菜单（未选择引脚布局选项卡）

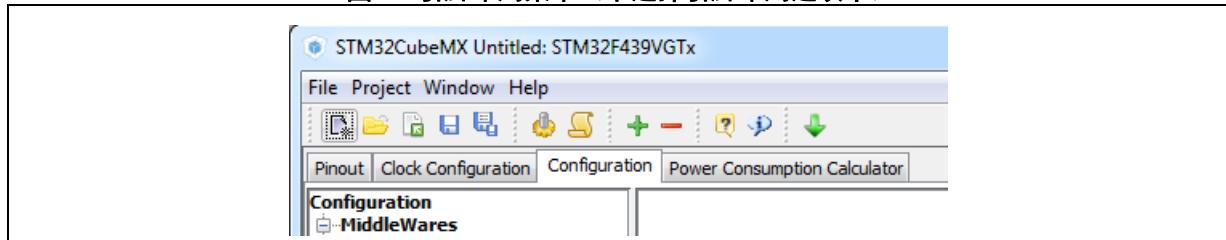


表5. 引脚布局菜单

图标	名称	说明
	撤销	撤销前面的配置步骤（逐个）
	恢复	恢复已经撤销的步骤（逐个）
无图标	引脚/信号选项	<p>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。</p> <p>点击引脚图标，单独固定/取消固定引脚信号。</p> <p>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。</p> <p>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。</p>
	引脚布局搜索字段	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，“芯片”视图上闪烁显示符合搜索条件的引脚或引脚组。点击“芯片”视图，停止闪烁。
	显示用户标签	允许在“芯片”视图上显示用户定义的标签，而不是分配给引脚的信号名称。
无图标	清除引脚布局	<p>在“引脚布局”窗口中清除用户引脚布局配置。</p> <p>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）。</p>
无图标	清除单一映射信号	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。
无图标	设置未使用的GPIO	<p>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）。</p> <p>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。</p>
无图标	复位已使用的GPIO	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。
	生成引脚布局的csv文本文件	将引脚配置生成为.csv文本文件

表5. 引脚布局菜单 (续)

图标	名称	说明
无图标	列出与引脚布局兼容的MCU	<p>提供最符合当前项目引脚配置的MCU列表。匹配可以是：</p> <ul style="list-style-type: none"> - 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，但不能全部在同一个引脚位置 <p>有关如何使用此特性的详细信息，请参见 教程5：将当前项目配置导出到兼容MCU。</p>
	全部折叠	折叠外设/中间件树视图
	禁用模式	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）。
	展开全部	展开外设/中间件树视图，显示所有功能模式。
	放大	放大芯片引脚布局图
	最佳适配	将芯片引脚布局图调整到最佳尺寸
<input checked="" type="checkbox"/> Keep Current Signals Placement	缩小	缩小芯片引脚布局图
	保持当前信号布置	<p>仅可从工具栏使用。</p> <p>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。</p>
	顺时针旋转 逆时针旋转	<p>在当前视图中，将MCU内核及其引脚和映射信号顺时针或逆时针旋转90度。</p> <p>点击两次可旋转180度，三次旋转270度。</p>
	水平翻转/垂直翻转	<p>仅可用于BGA封装。</p> <p>允许在底视图和顶视图之间水平或垂直翻转。（然后保持文本原样）。底视图对应于BGA封装的底侧（球侧朝上），顶视图对应于另一侧（球侧朝下，ST标识清晰可见（如有））。</p>

5.4.4 窗口菜单

“窗口”菜单允许访问输出功能（参见表 6）。

表6. 窗口菜单

名称	说明
输出	打开STM32CubeMX主窗口底部的MCU选择窗口。 打开STM32CubeMX主窗口底部的两个选项卡： – “MCU选择”选项卡，用于列出符合用户条件的MCU，用户条件可通过MCU选择器来选择。 – “输出”选项卡，显示用户操作时遇到的STM32CubeMX消息、警告和错误。

5.4.5 帮助菜单

有关帮助菜单和图标的描述，请参见表 7。

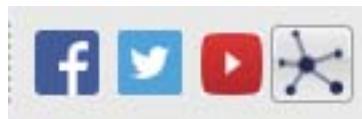
表7. 帮助菜单

图标	名称	说明
	帮助内容	打开 STM32CubeMX 用户手册
	关于...	显示版本信息
	检查更新	显示可供下载的软件和固件版本更新。
	管理嵌入式软件包	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）。
	更新程序设置...	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。
无图标	刷新数据	打开一个对话窗口，该窗口建议使用STM32 MCU最新信息刷新STM32CubeMX数据库（MCU描述和官方文档列表），并允许一次性下载所有官方文档
无图标	文档和资源	显示可用于当前项目中所用MCU的官方文档。

5.4.6 社交链接

可通过STM32CubeMX工具栏（参见图39）访问建于Facebook、Twitter、STM32uTube频道和ST社区等流行社交平台上的开发者社区。

图39. 社交平台链接



5.5 输出窗口

5.5.1 MCU选择面板

当选定某一MCU时，该窗口列出某一特定系列中匹配用户标准（系列、外设、封装..）的所有MCU。

注：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。系统将提示用户在继续下一步之前确认此操作。

图40. MCU选择菜单

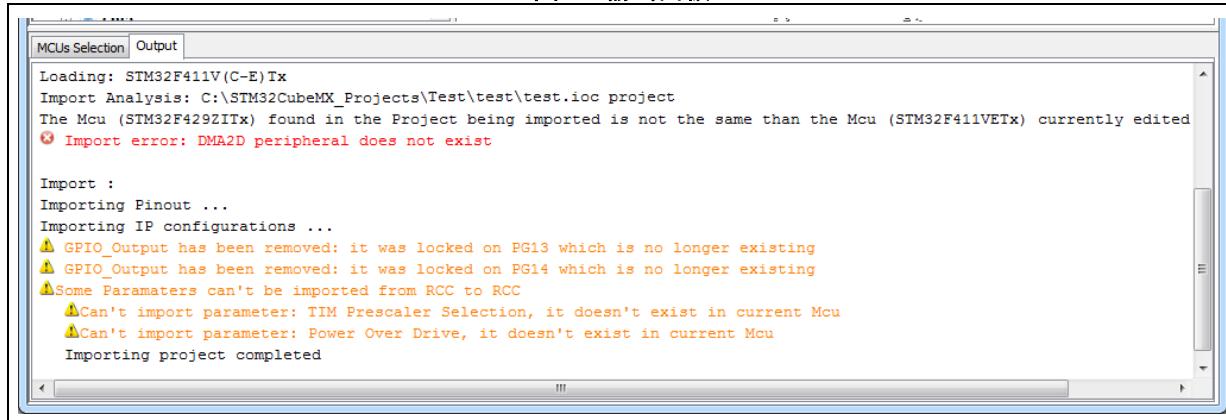
MCUs Selection					
Series	Lines	Mcu	Package	Required Peripherals	
<input type="checkbox"/> STM32F4	STM32F429/439	STM32F429VETx	LQFP100	RTC,SAI,SDIO	
<input type="checkbox"/> STM32F4	STM32F429/439	STM32F429VGTx	LQFP100	RTC,SAI,SDIO	
<input type="checkbox"/> STM32F4	STM32F429/439	STM32F429VITx	LQFP100	RTC,SAI,SDIO	
<input type="checkbox"/> STM32F4	STM32F429/439	STM32F429ZETx	LQFP144	RTC,SAI,SDIO	
<input type="checkbox"/> STM32F4	STM32F429/439	STM32F429ZGTx	LQFP144	RTC,SAI,SDIO	
<input type="checkbox"/> STM32F4	STM32F429/439	STM32F429ZITx	LQFP144	RTC,SAI,SDIO	
<input type="checkbox"/> STM32F4	STM32F429/439	STM32F429ZEYx	WL CSP143	RTC,SAI,SDIO	
<input type="checkbox"/> STM32F4	STM32F429/439	STM32F429ZGYx	WL CSP143	RTC,SAI,SDIO	
<input type="checkbox"/> STM32F4	STM32F429/439	STM32F429ZIYx	WL CSP143	RTC,SAI,SDIO	
<input type="checkbox"/> STM32F4	STM32F429/439	STM32F439BGTx	LQFP208	RTC,SAI,SDIO	
<input type="checkbox"/> STM32F4	STM32F429/439	STM32F439BITx	LQFP208	RTC,SAI,SDIO	
<input type="checkbox"/> STM32F4	STM32F429/439	STM32F439IGHx	UF BGA176	RTC,SAI,SDIO	
<input type="checkbox"/> STM32F4	STM32F429/439	STM32F439IIHx	UF BGA176	RTC,SAI,SDIO	
<input type="checkbox"/> STM32F4	STM32F429/439	STM32F439IGTx	LQFP176	RTC,SAI,SDIO	
<input type="checkbox"/> STM32F4	STM32F429/439	STM32F439IITx	LQFP176	RTC,SAI,SDIO	
<input type="checkbox"/> STM32F4	STM32F429/439	STM32F439NGHx	TF BGA216	RTC,SAI,SDIO	
<input type="checkbox"/> STM32F4	STM32F429/439	STM32F439NIHx	TF BGA216	RTC,SAI,SDIO	
<input type="checkbox"/> STM32F4	STM32F429/439	STM32F439VGTx	LQFP100	RTC,SAI,SDIO	
<input checked="" type="checkbox"/> STM32F4	STM32F429/439	STM32F439VITx	LQFP100	RTC,SAI,SDIO	
<input type="checkbox"/> STM32F4	STM32F429/439	STM32F439ZGTx	LQFP144	RTC,SAI,SDIO	
<input type="checkbox"/> STM32F4	STM32F429/439	STM32F439ZITx	LQFP144	RTC,SAI,SDIO	
<input type="checkbox"/> STM32F4	STM32F429/439	STM32F439ZGYx	WL CSP143	RTC,SAI,SDIO	
<input type="checkbox"/> STM32F4	STM32F420/430	STM32F4307TVx	WL CSP143	RTC,SAI,SDIO	

从窗口菜单中选择/取消选择输出可以显示/隐藏该窗口。

5.5.2 输出面板

此面板显示包含执行的操作、错误和弹出警告的非详尽列表（参见图 41）。

图41. 输出面板



5.6 “导入项目”窗口

导入项目菜单简化了将以前保存的配置移植到另一个MCU的过程。

默认会导入以下设置：

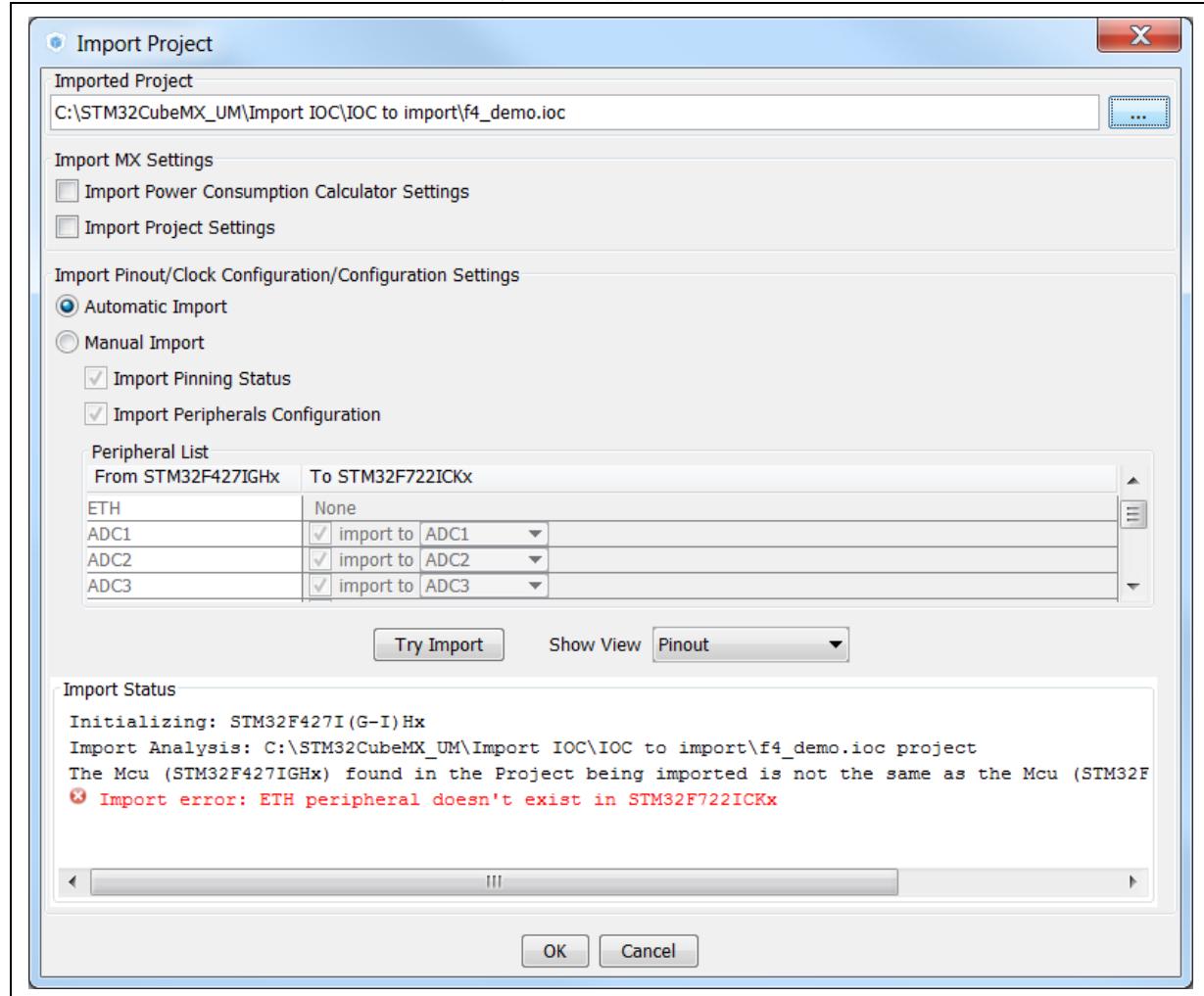
- “引脚布局”选项卡：MCU引脚和相应的外设模式。如果目标MCU中没有相同的外设实例，则导入失败。
- “时钟配置”选项卡：时钟树参数。
- “配置”选项卡：外设和中间件库初始化参数。
- “项目”设置：选择工具链和代码生成选项。

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 42）

图42. 自动项目导入

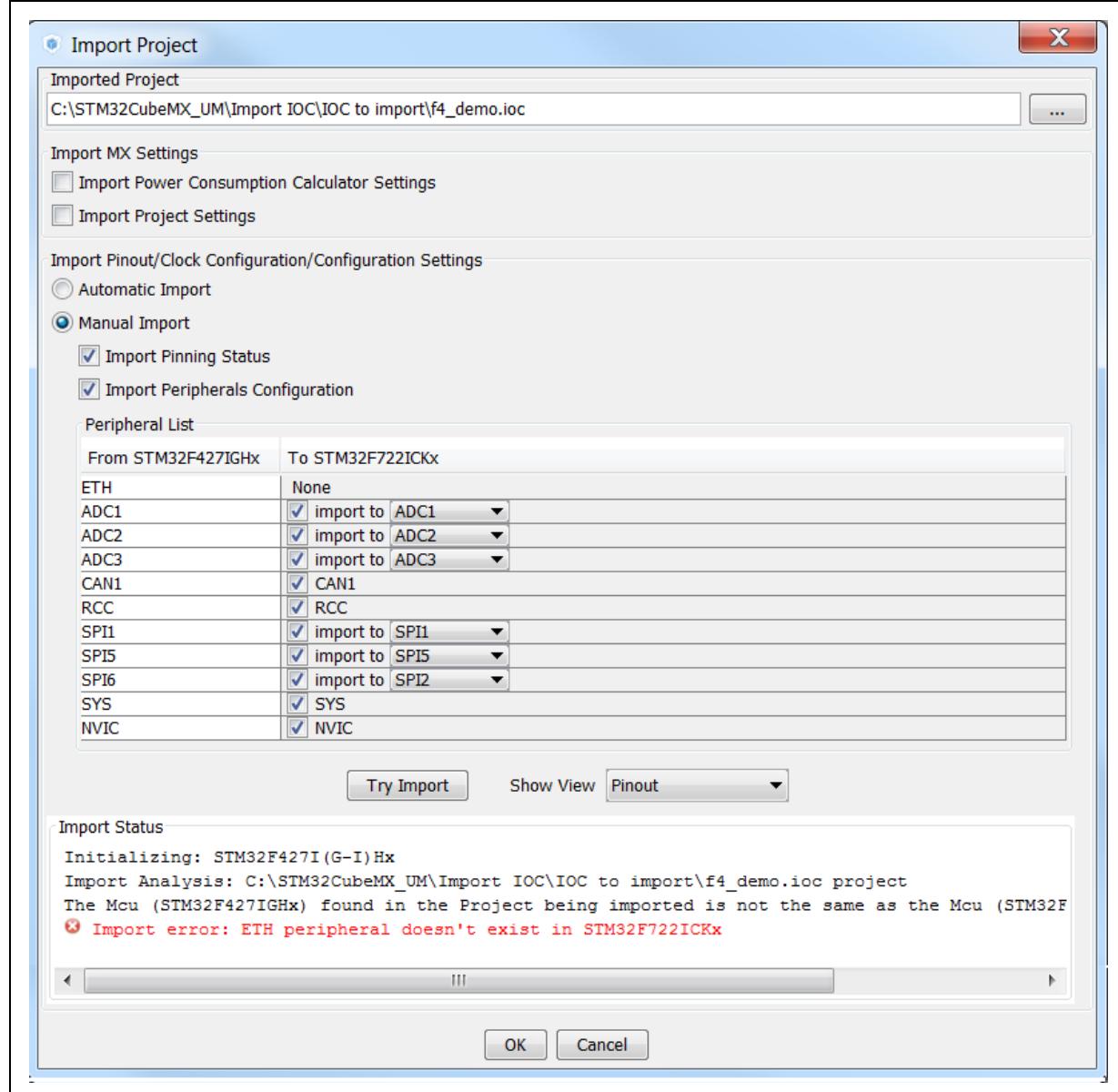


b) 手动项目导入

在这种情况下，可以通过复选框手动选择外设集（参见图 43）。

选择尝试导入选项进行尝试导入。

图43. 手动项目导入



外设列表显示：

- 待导入的项目中配置的外设实例
- 如果当前选定的MCU存在外设实例，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 44）、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布局冲突，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 45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（引脚布局、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 45）。

图44. “导入项目”菜单 - 尝试导入时出现错误

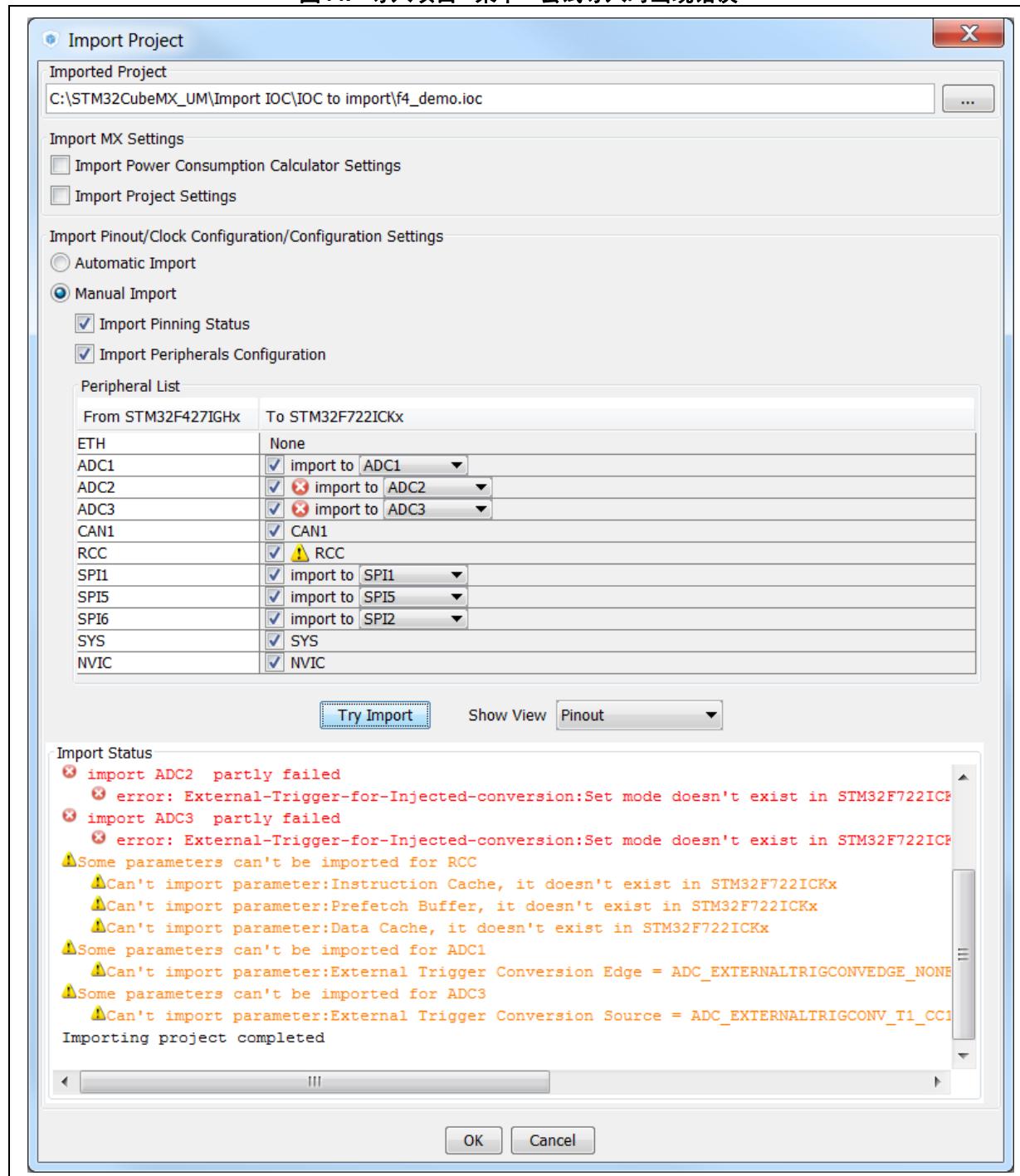
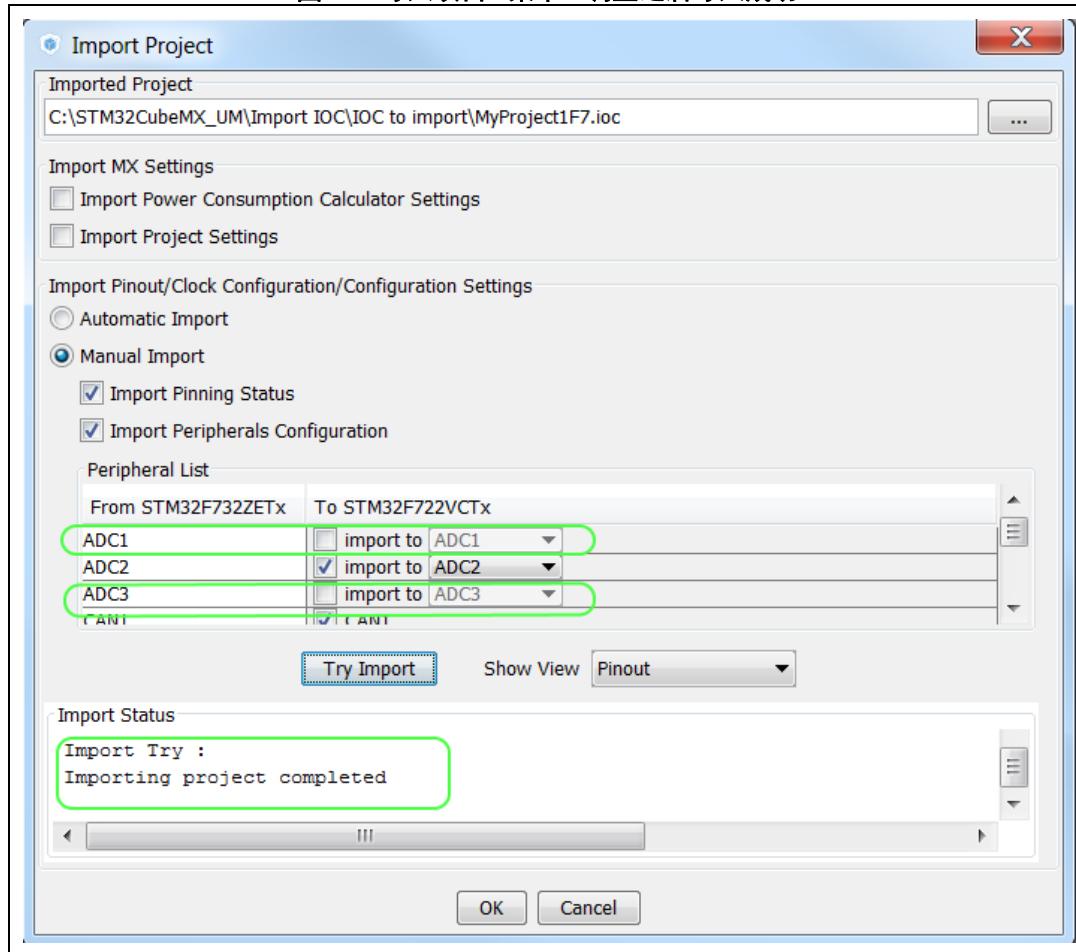


图45. “导入项目”菜单 - 调整之后导入成功



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。
一旦导入，“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。

5.7 “设置未使用 / 重置已使用GPIO”窗口

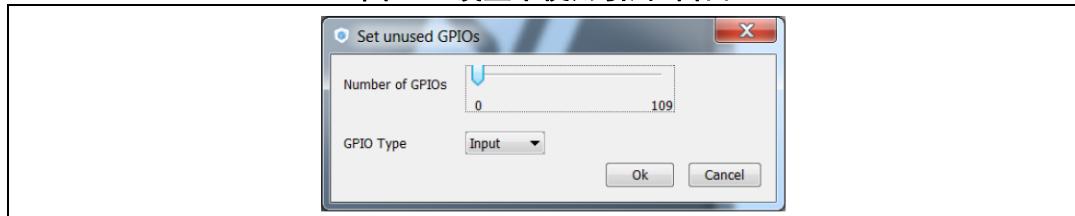
这些窗口允许在相同的GPIO模式下一次配置多个引脚。

如要打开它们：

- 从STM32CubeMX菜单栏选择引脚布局 > **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46. “设置未使用引脚”窗口



- 从STM32CubeMX菜单栏选择引脚布局 > **重置已使用GPIO**。

根据是否在工具栏上选中了“保持当前信号布置”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“保持当前信号布置”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 48](#)和[图 49](#)并检查可用外设模式的限制。

图47. “重置已使用引脚”窗口

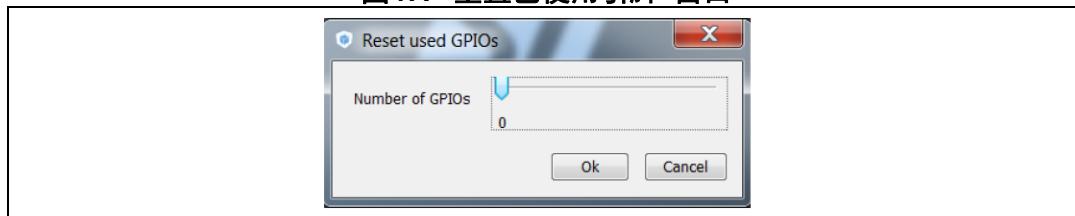


图48. 在勾选了“保持当前信号布置”选项的情况下设置未使用的GPIO引脚

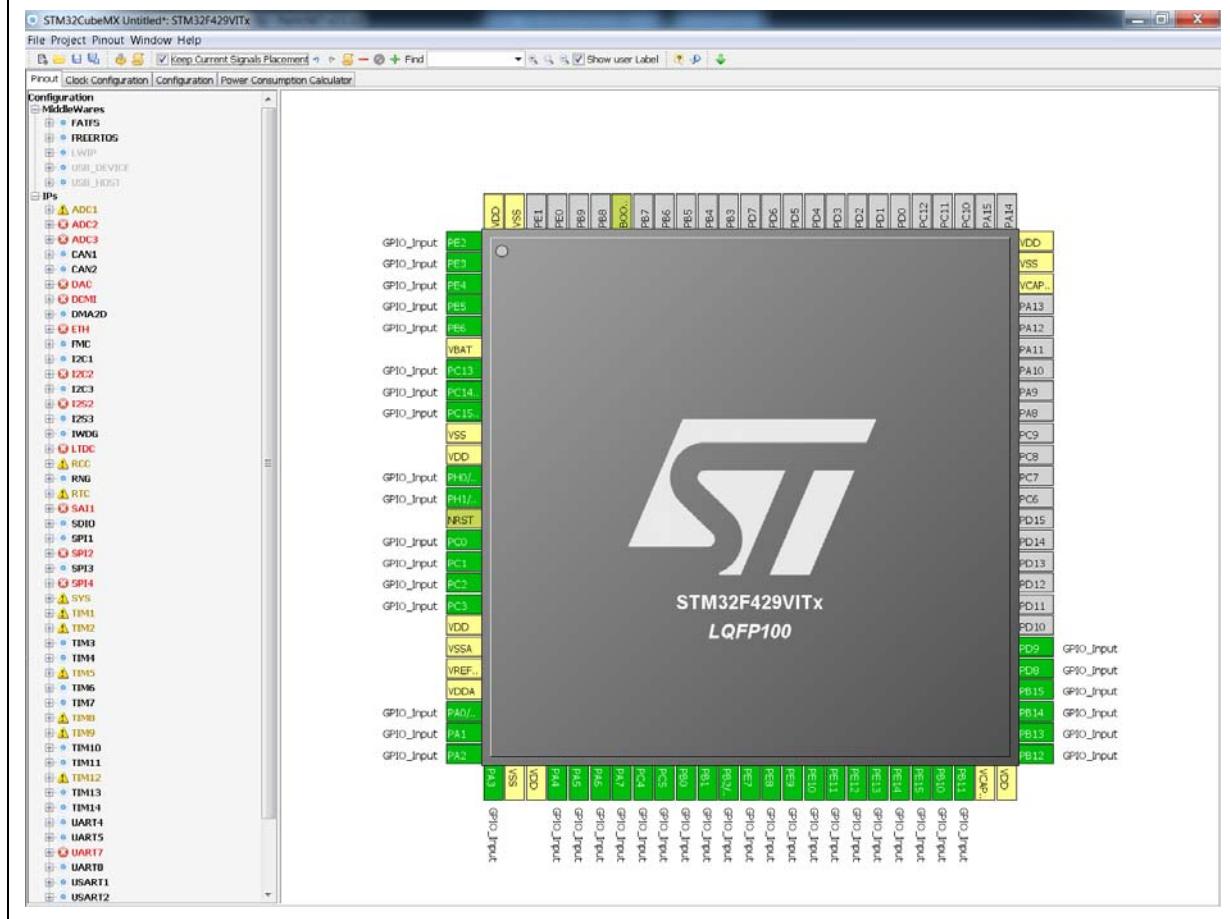
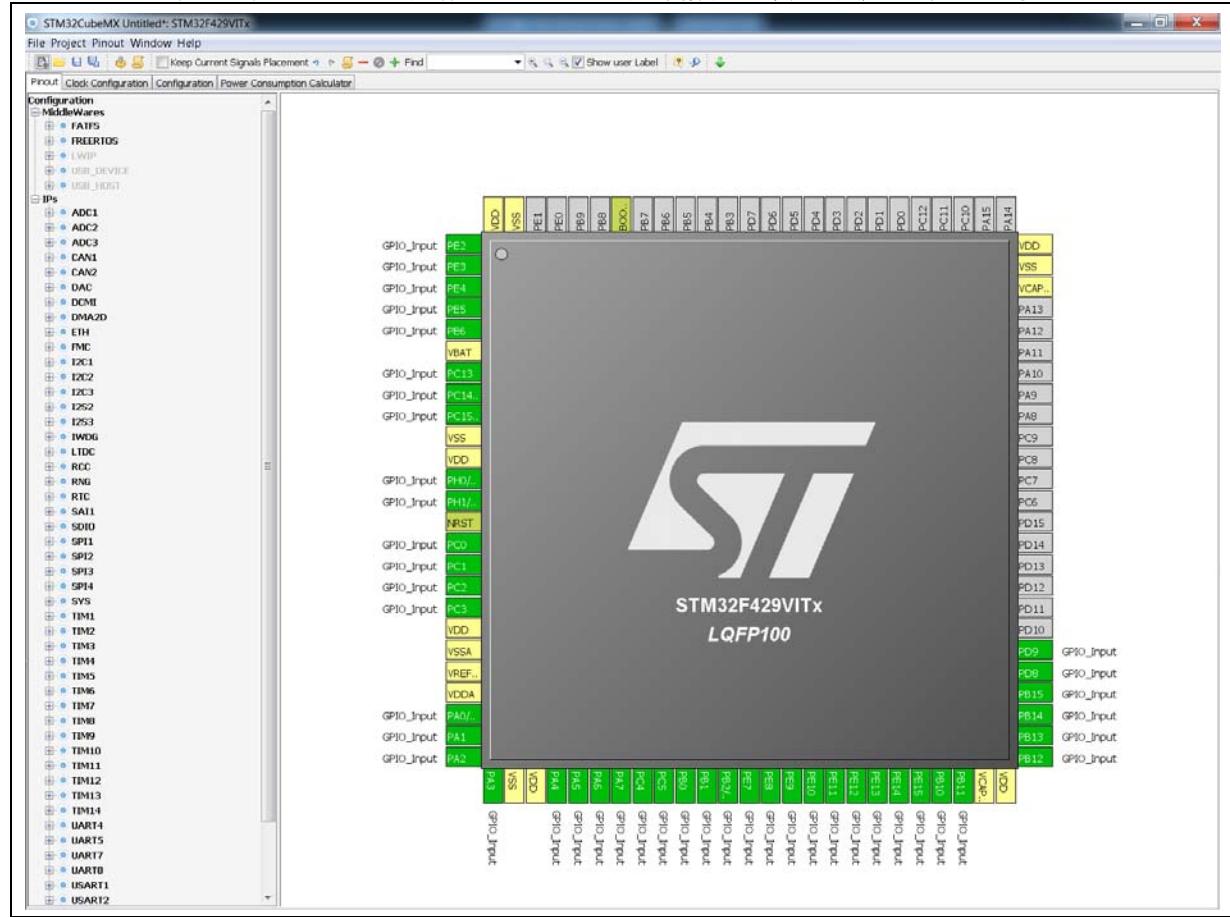


图49. 在未勾选“保持当前信号布置”选项的情况下设置未使用的GPIO引脚



5.8 “项目设置”窗口

该“项目设置”窗口包含三个选项卡：

- 通用项目设置选项卡，允许指定项目名称、位置、工具链和固件版本。
- 代码生成选项卡，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。
- 高级设置选项卡，用于排序STM32CubeMX初始化功能调用。

有多个方法可以打开“项目设置”窗口：

- 从STM32CubeMX菜单栏选择**项目>设置**（参见图 50）。然后，代码生成将在图 51中显示的项目文件夹树中生成。
- 通过第一次点击**项目>生成代码**。
- 通过为包含C代码生成（不仅是引脚配置）的项目选择**另存为**。

图50. “项目设置”窗口

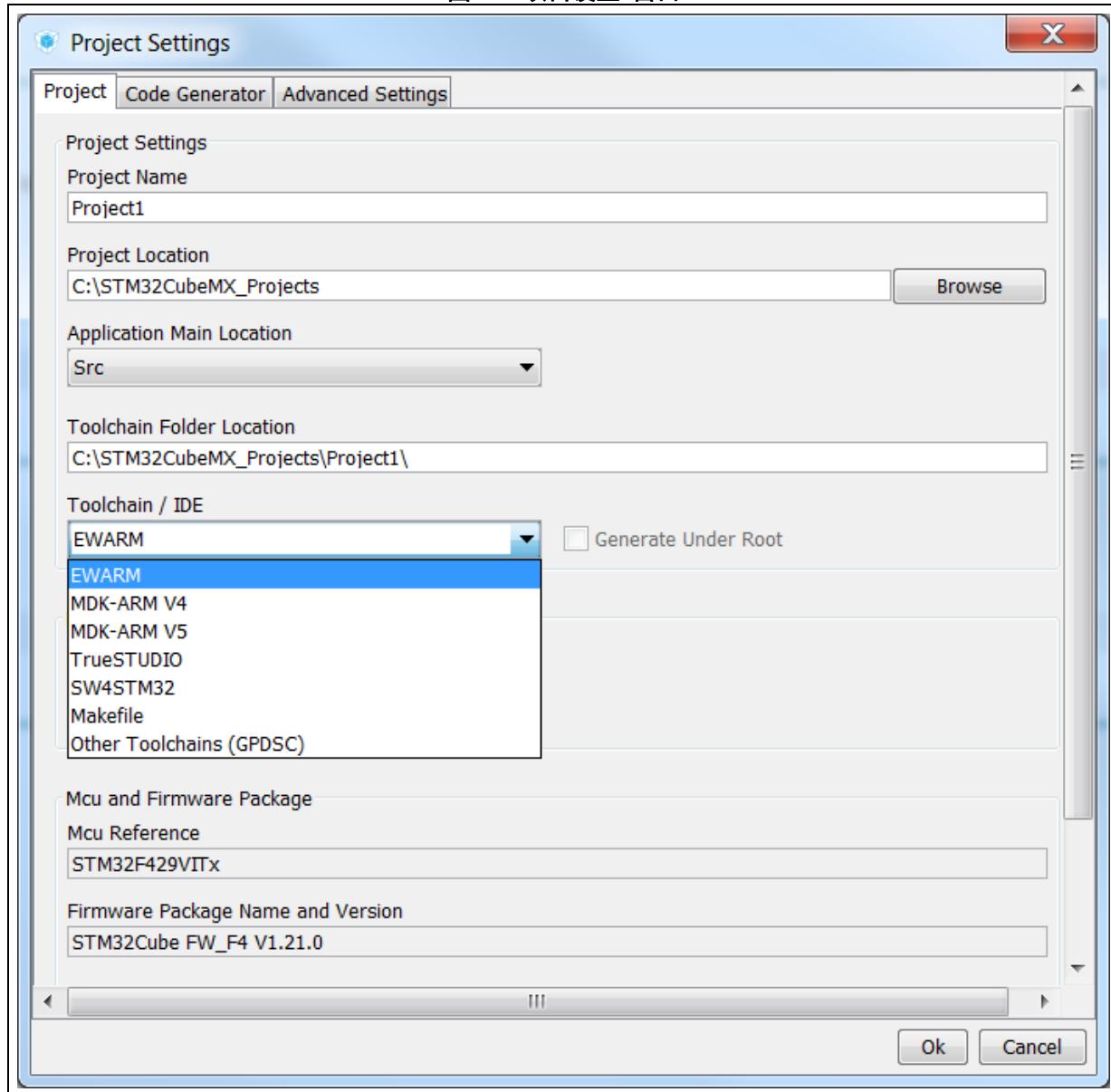
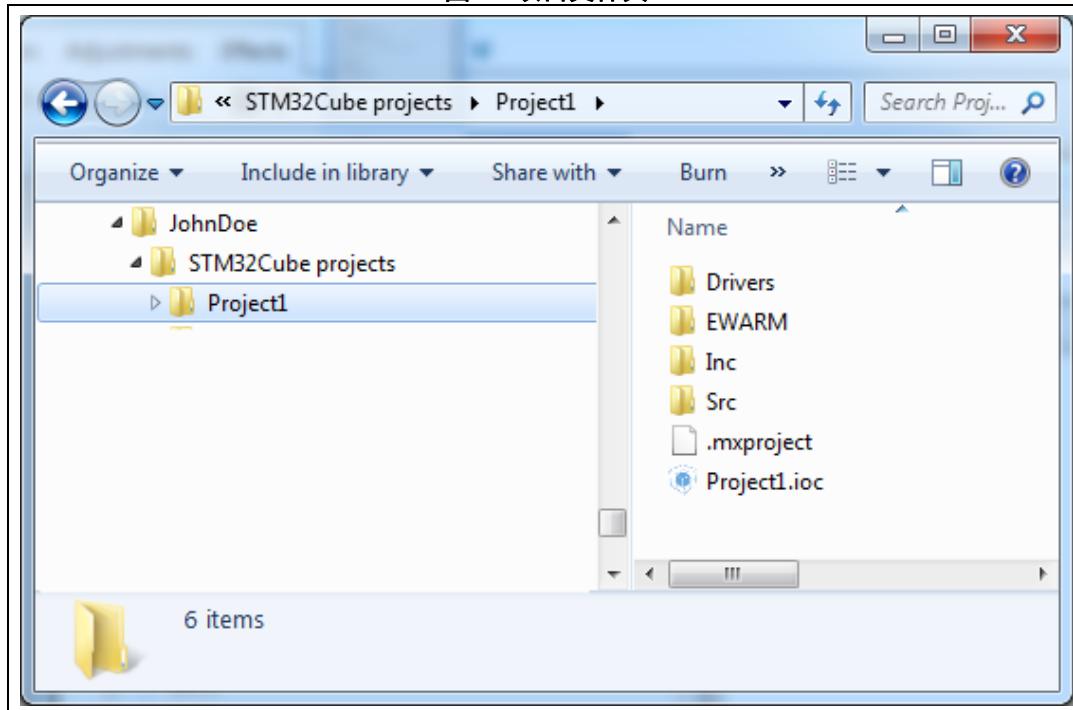


图51. 项目文件夹



5.8.1 “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 50）：

- 项目设置：项目名称、项目位置、应用主文件位置（Inc、Src或Core/Inc、Core/Src文件夹）、工具链文件夹以及其他用于项目生成的要素。

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。这就允许将STM32CubeMX项目生成扩展到支持gpdsc的任何工具链，因为工具链可以通过处理gpdsc文件信息的方式加载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。

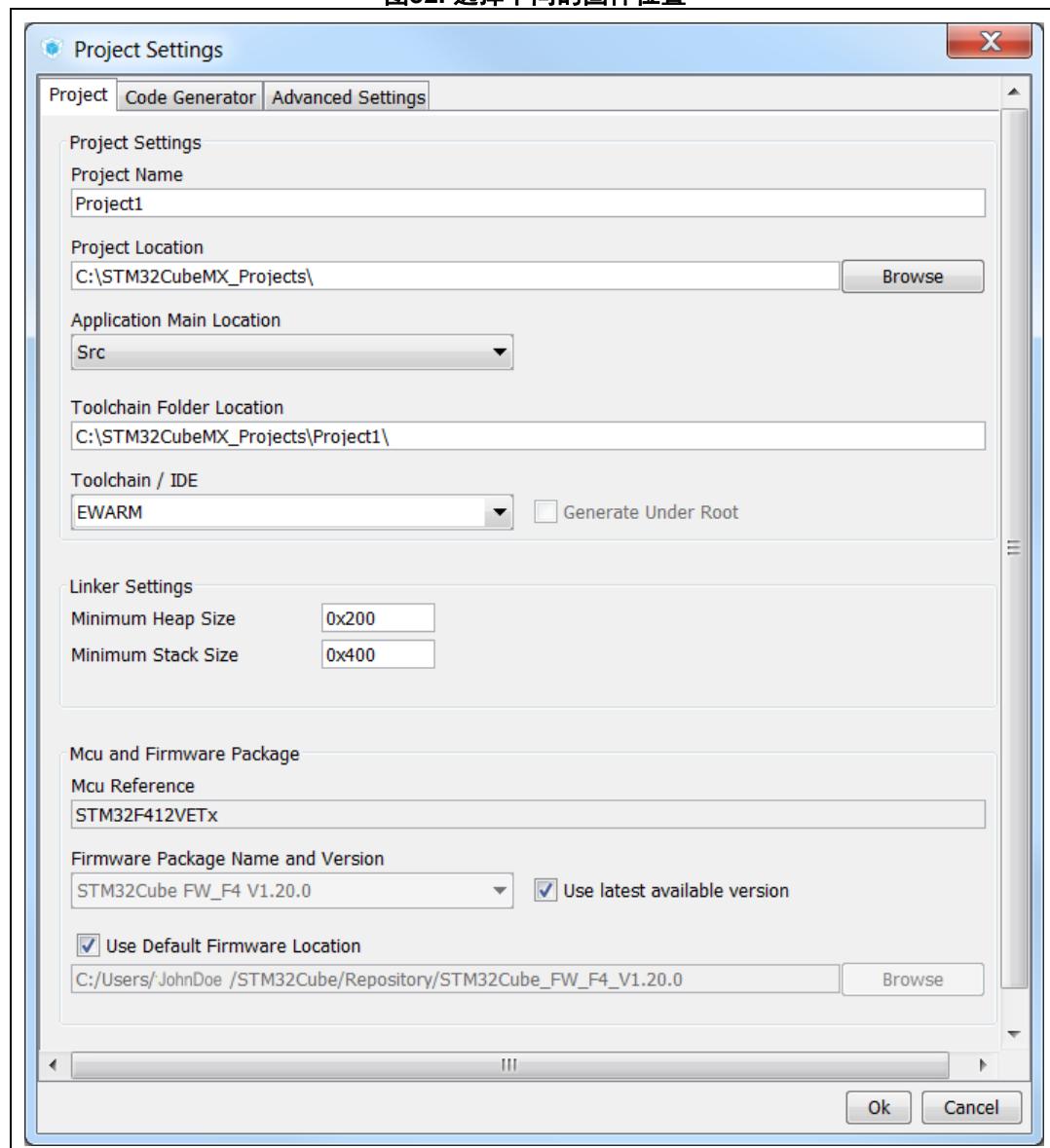
但是，需要注意的是，选择将项目复制到工作区中将防止在Eclipse中完成的更改与在STM32CubeMX中完成的更改之间发生任何进一步的同步，因为项目将有两个不同的副本。

- 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。

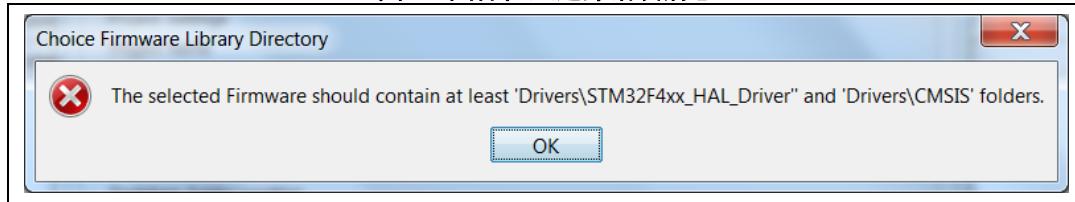
取消选择**使用默认固件位置**复选框，则可为项目将要使用的固件指定不同的路径（参见图 52）。

图52. 选择不同的固件位置



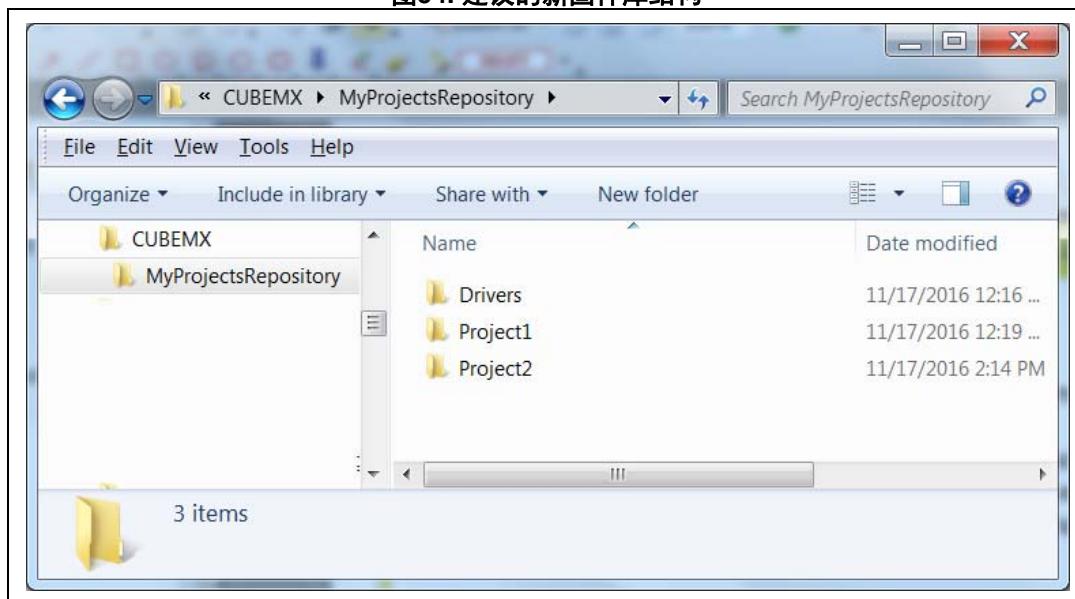
新位置应至少包含一个驱动目录，其中包含来自相关STM32Cube MCU封装的HAL和CMSIS驱动。如果找不到文件夹，将弹出错误消息（参见图 53）。

图53. 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重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 54）。

图54. 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。选择另一个位置将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。

5.8.2 “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 55）：

- “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

此区域允许定义以下选项：

- 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fsl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将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 56）。

然后将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 然后将使用选定的模板生成用户自定义代码（参见[自定义代码生成](#)）。

图 57显示图 56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55. 项目设置代码生成器

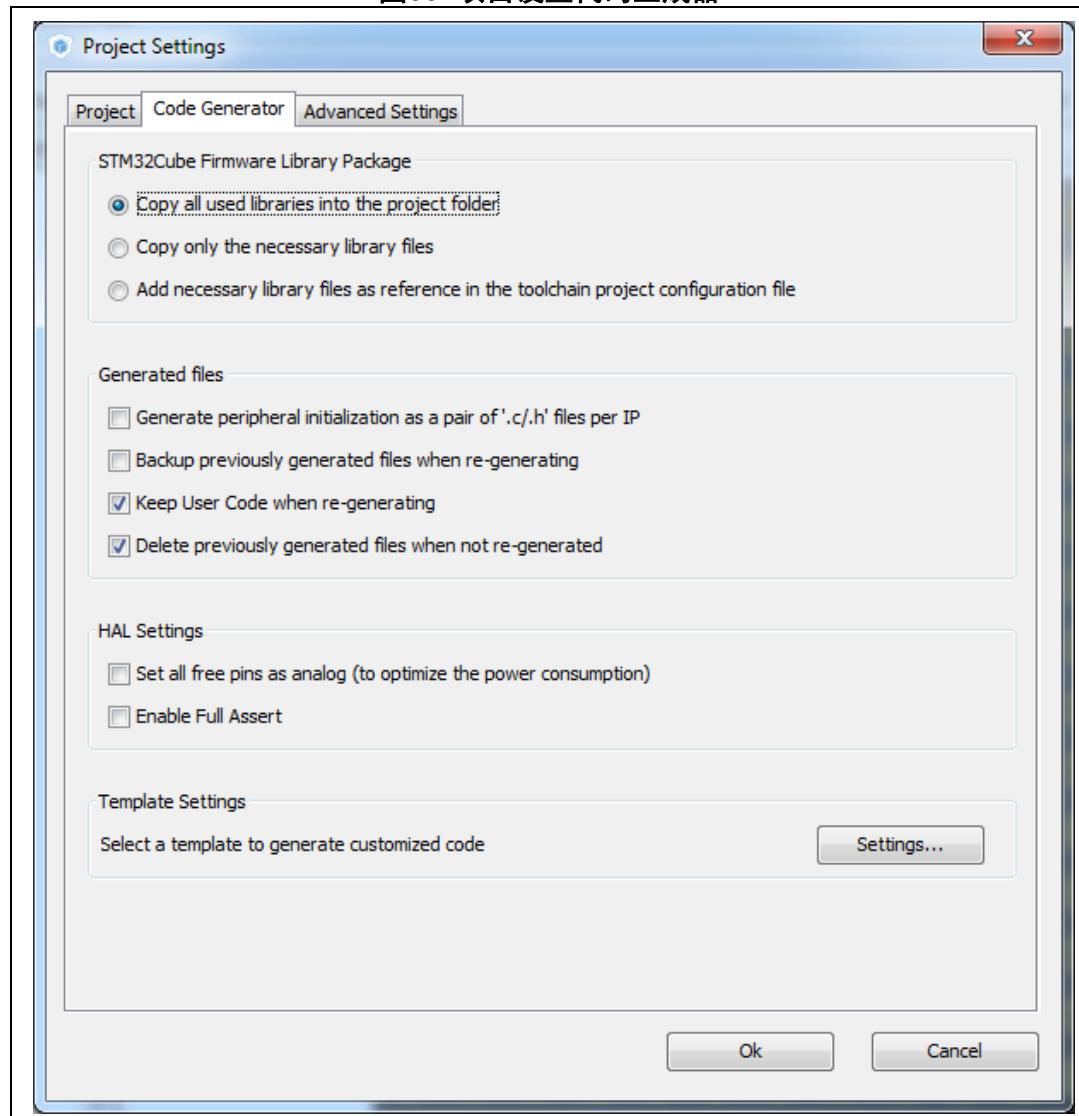


图56. “模板设置”窗口

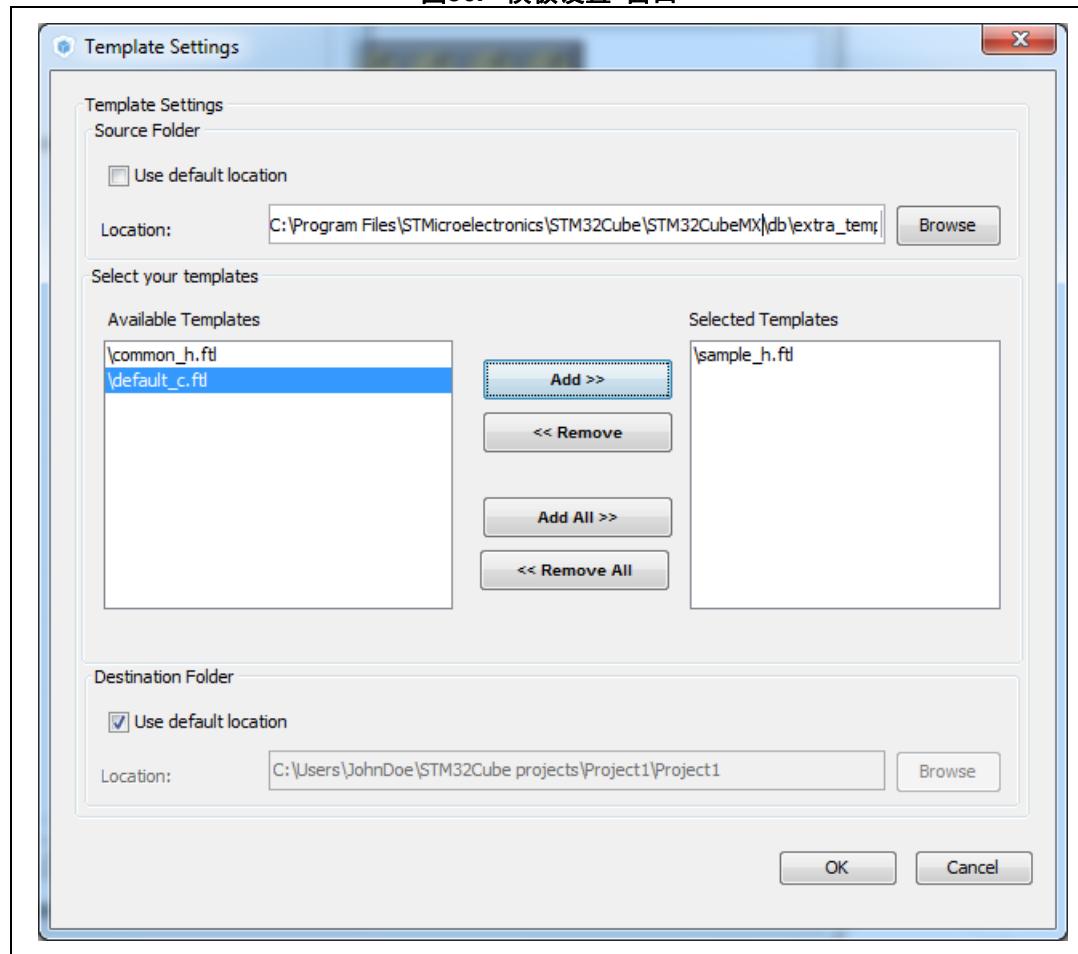
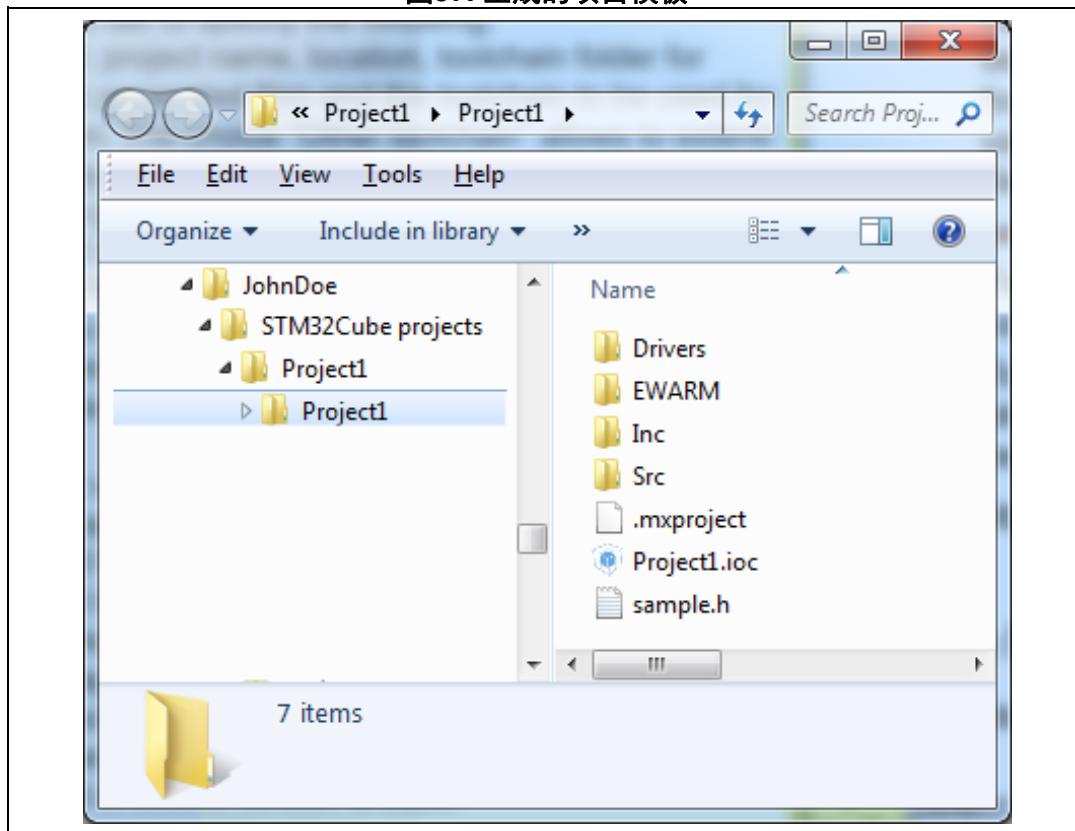


图57. 生成的项目模板



5.8.3 “高级设置”选项卡

图 58 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。

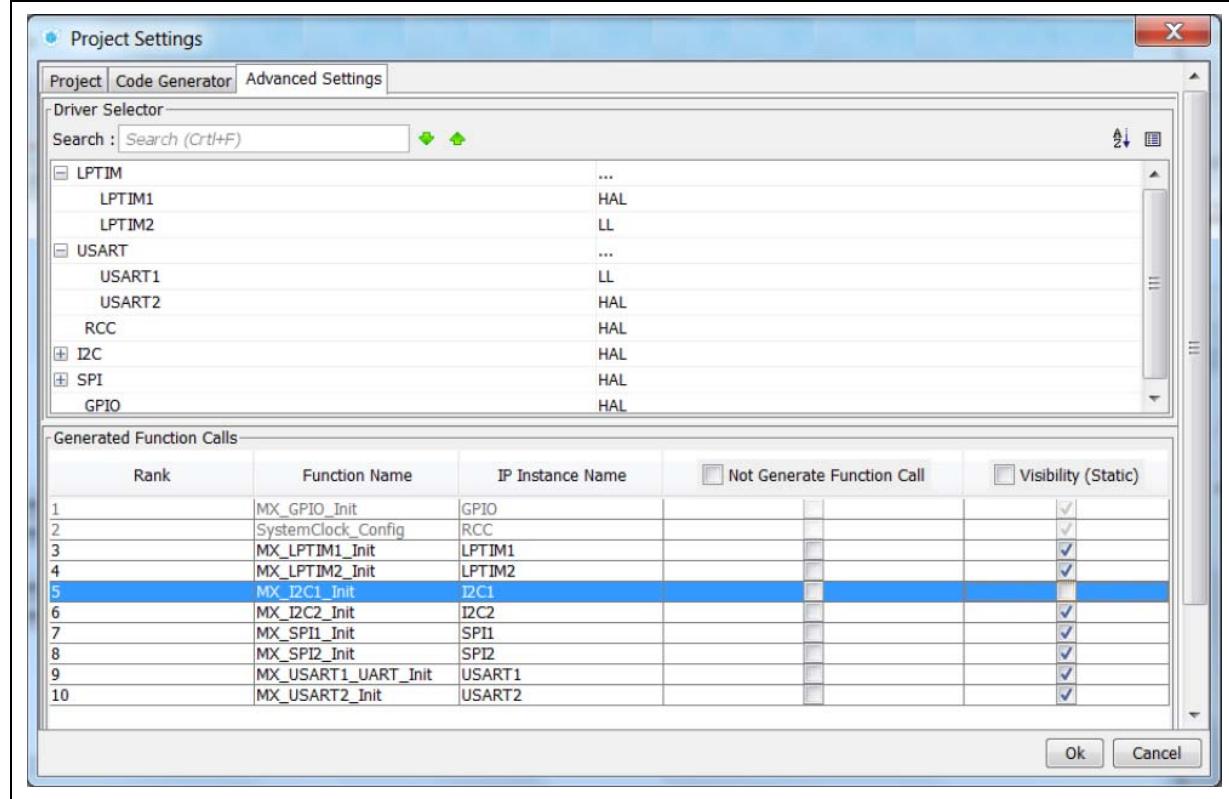
禁用对初始化函数的调用

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。将相应地生成代码（参见[使用底层驱动程序生成STM32Cube代码](#)）。

图58. “高级设置”窗口



取消选择可见性（Static）选项（如图 58 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 59）。

图59. 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); // This line is highlighted with a red box
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);
```

5.9 “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[获取STM32Cube和第三方软件发布和更新](#)。

5.10 附加软件组件选择窗口

在进行项目操作时，该窗口可以随时打开。它允许为当前项目选择附加软件组件。

它包含三个面板：

- **软件组件面板**
该面板位于窗口的右上方。它显示可以为项目选择的软件组件列表。
- **筛选器面板**
它位于窗口的左侧，提供一组用于筛选包组件列表的条件。
- **条件解析面板**
它位于窗口的右下角，通过单击要解析的条件进行显示。它提供条件的详细描述，并提出解决方案（如果在软件组件列表中可以找到任何解决方案）。

有三种方法可以打开附加软件组件选择窗口：

- 选择项目，然后点击选择附加软件组件
- 点击工具栏上的菜单图标 
- 使用Ctrl+E快捷键。

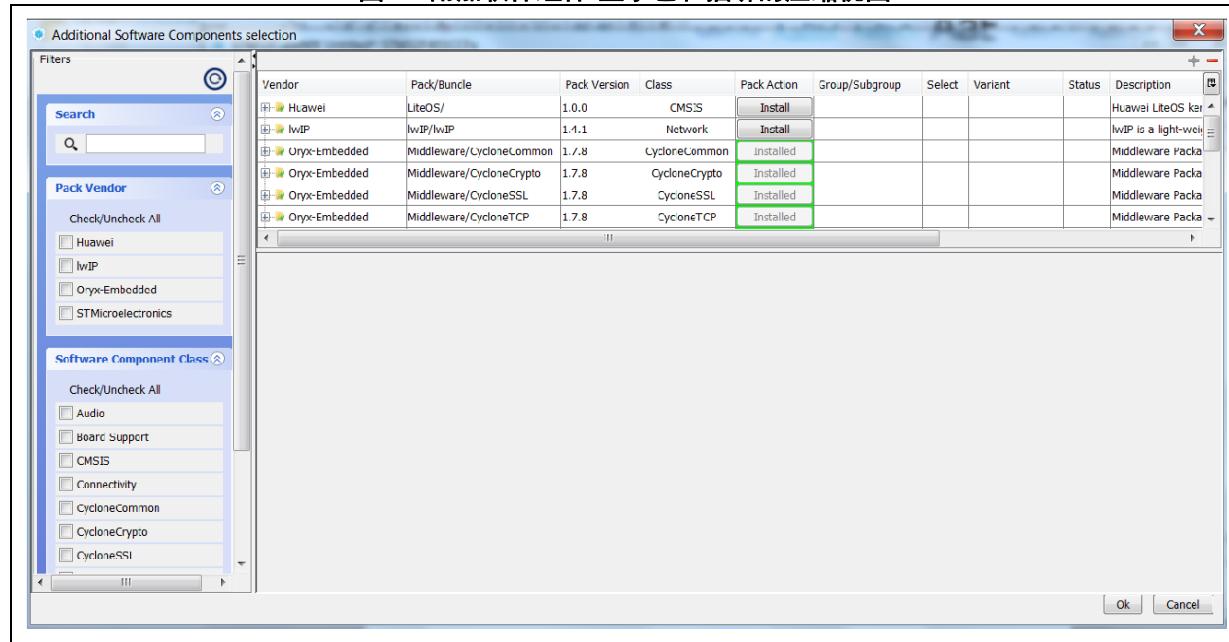
5.10.1 软件组件简介

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。

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。

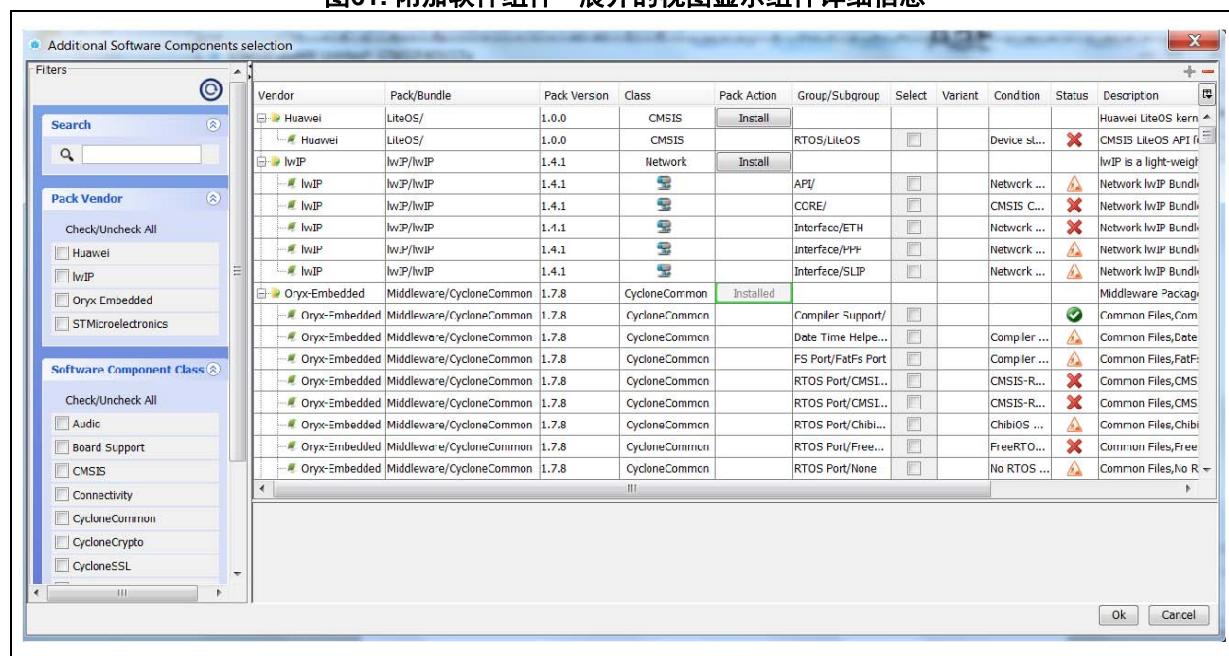
默认情况下，窗口会显示一个按供应商名称（第一列）和包或捆绑版本（第二列）分组的组件列表。有关示例，请参见[图 60](#)。

图60. 附加软件组件-显示包和捆绑的压缩视图



点击 按钮展开视图并显示所有软件组件的详细信息（参见图 61）。

图61. 附加软件组件 - 展开的视图显示组件详细信息



5.10.2 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。

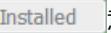
软件组件搜索结果表已经折叠。点击  按钮将其展开并显示所有符合筛选条件的组件。

点击  清空所有筛选器复选框或点击 **Check/Uncheck All** 清空给定类别的筛选器（包供应商或组件类别）。

5.10.3 软件组件表格

STM32CubeMX 解析pack .pdsc文件以提取嵌入式软件组件的列表和相应的特性。

表8. 软件组件

列名称	说明
供应商	包供应商名称，可在包名称中找到。
包 / 绑绑	包或捆绑名称。
版本	包 / 绑绑版本。
分类 ⁽¹⁾	组件所属的组件类别。
包操作	点击  以安装包，  意味着该包已经安装。
组/子组	软件组件所属的组和子组（可选）。
	允许向当前项目添加软件组件的选择复选框。
型号	适用于存在不同型号（如调试和发布）的软件组件的可选字段。
条件	描述条件的缩写名（比如CMSIS-RTOS Dependencies）。
状态	待满足条件的状态（参见 表 9 获取不同的条件状态）。单击状态将在底部面板中打开条件描述。
说明	软件组件的简短描述。

1. Arm® Keil™ CMS-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

无线：Bluetooth®、WiFi®、以及 ZigBee® 等通信协议栈。

5.10.4 软件组件条件

条件是应用于给定软件组件的依赖规则。使用表9中显示的图标可显示软件组件条件状态。

表9. 软件组件条件图标

图标	说明
✓	所有软件组件条件（如有）都已解析。
✗	不能通过当前包列表或为当前选定的MCU解析一个或多个软件组件条件。
⚠	必须使用同一个包的另一个软件组件解析一个或多个条件。
⚠	必须使用另一个包的软件组件来解析一个或多个条件。

- 单击条件状态图标，在底部面板中显示要解析的条件，以及它们的描述和可以解决此问题的软件组件（如有）（参见图62和图63）。

图62. 依赖性解决：找到解决方案

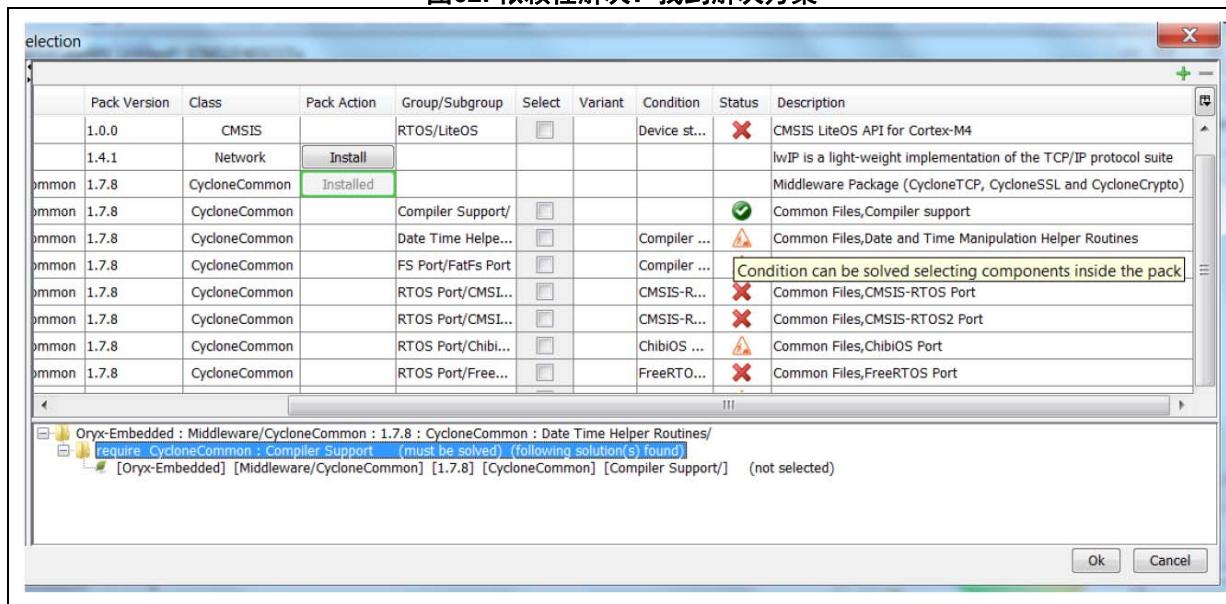


图63. 依赖性解决：未找到解决方案

Pack Version	Class	Pack Action	Group/Subgroup	Select	Variant	Condition	Status	Description
1.0.0	CMSIS		RTOS/LiteOS	<input type="checkbox"/>		Device st...	X	CMSIS LiteOS API for Cortex-M4
1.4.1	Network	Install						IwIP is a light-weight implementation of the TCP/IP protocol suite
Common 1.7.8	CycloneCommon	Installed						Middleware Package (CycloneTCP, CycloneSSL and CycloneCrypto)
Common 1.7.8	CycloneCommon		Compiler Support/	<input type="checkbox"/>			✓	Common Files,Compiler support
Common 1.7.8	CycloneCommon		Date Time Helper...	<input type="checkbox"/>		Compiler ...	⚠	Common Files,Date and Time Manipulation Helper Routines
Common 1.7.8	CycloneCommon		FS Port/FatFs Port	<input type="checkbox"/>		Compiler ...	⚠	Common Files,FatFs Port
Common 1.7.8	CycloneCommon		RTOS Port/CMSI...	<input type="checkbox"/>		CMSIS-R...	X	Common Files,CMSIS-RTOS Port
Common 1.7.8	CycloneCommon		RTOS Port/CMSI...	<input type="checkbox"/>		CMSIS-R...	X	Missing components CMSIS-RTOS2 Port
Common 1.7.8	CycloneCommon		RTOS Port/Chibi...	<input type="checkbox"/>		ChibiOS ...	⚠	Common Files,ChibiOS Port
Common 1.7.8	CycloneCommon		RTOS Port/Free...	<input type="checkbox"/>		FreeRTOS ...	X	Common Files,FreeRTOS Port
Common 1.7.8	CycloneCommon		RTOS Port/None	<input type="checkbox"/>		No RTOS ...	⚠	Common Files,No RTOS
Common 1.7.8	CycloneCommon		RTOS Port/RTX ...	<input type="checkbox"/>		RTX Dep...	⚠	Common Files,Keil RTX Port
Common 1.7.8	CycloneCommon		RTOS Port/embed...	<input type="checkbox"/>		embOS D...	⚠	Common Files,Segger embOS Port
Common 1.7.8	CycloneCommon		RTOS Port/uCOS...	<input type="checkbox"/>		uCOS-II ...	X	Common Files,Micrium uC/OS-II Port
Common 1.7.8	CycloneCommon		RTOS Port/uCOS...	<input type="checkbox"/>		uCOS-III ...	X	Common Files,Micrium uC/OS-III Port
Common 1.7.8	CycloneCommon		Resource Manag...	<input type="checkbox"/>		Compiler ...	⚠	Common Files,FLASH Resource Manager

树视图显示了以下信息：

- Oryx-Embedded : Middleware/CycloneCommon : 1.7.8 : CycloneCommon : RTOS Port/CMSIS-RTOS Port
 - require CycloneCommon : Compiler Support (must be solved) (following solution(s) found)
 - [Oryx-Embedded] [Middleware/CycloneCommon] [1.7.8] [CycloneCommon] [Compiler Support/] (not selected)
 - require CMSIS : RTOS (must be solved) (following solution(s) found)
 - [Huawei] [LiteOS/] [1.0.0] [CMSIS] [RTOS/LiteOS] (not selected)
 - require CMSIS : CORE (must be solved) (no solution found)

2. 在底部面板中单击建议的解决方案将突出显示上面组件表格中的软件组件选择复选框（参见图 64）。

图64. 软件组件条件 - 解决方案建议及指引

Pack Version	Class	Pack Action	Group/Subgroup	Select	Variant	Condition	Status	Description
1.0.0	CMSIS		RTOS/LiteOS	<input type="checkbox"/>		Device st...	X	CMSIS LiteOS API for Cortex-M4
1.4.1	Network	Install						IwIP is a light-weight implementation of the TCP/IP protocol suite
Common 1.7.8	CycloneCommon	Installed						Middleware Package (CycloneTCP, CycloneSSL and CycloneCrypto)
Common 1.7.8	CycloneCommon		Compiler Support/	<input checked="" type="checkbox"/>			✓	Common Files,Compiler support
Common 1.7.8	CycloneCommon		Date Time Helper...	<input checked="" type="checkbox"/>		Compiler ...	⚠	Common Files,Date and Time Manipulation Helper Routines
Common 1.7.8	CycloneCommon		FS Port/FatFs Port	<input type="checkbox"/>		Compiler ...	⚠	Common Files,FatFs Port
Common 1.7.8	CycloneCommon		RTOS Port/CMSI...	<input type="checkbox"/>		CMSIS-R...	X	Common Files,CMSIS-RTOS Port
Common 1.7.8	CycloneCommon		RTOS Port/CMSI...	<input type="checkbox"/>		CMSIS-R...	X	Common Files,CMSIS-RTOS2 Port
Common 1.7.8	CycloneCommon		RTOS Port/Chibi...	<input type="checkbox"/>		ChibiOS ...	⚠	Common Files,ChibiOS Port
Common 1.7.8	CycloneCommon		RTOS Port/Free...	<input type="checkbox"/>		FreeRTOS ...	X	Common Files,FreeRTOS Port
Common 1.7.8	CycloneCommon		RTOS Port/None	<input type="checkbox"/>		No RTOS ...	⚠	Common Files,No RTOS
Common 1.7.8	CycloneCommon		RTOS Port/RTX ...	<input type="checkbox"/>		RTX Dep...	⚠	Common Files,Keil RTX Port
Common 1.7.8	CycloneCommon		RTOS Port/embed...	<input type="checkbox"/>		embOS D...	⚠	Common Files,Segger embOS Port
Common 1.7.8	CycloneCommon		RTOS Port/uCOS...	<input type="checkbox"/>		uCOS-II ...	X	Common Files,Micrium uC/OS-II Port
Common 1.7.8	CycloneCommon		RTOS Port/uCOS...	<input type="checkbox"/>		uCOS-III ...	X	Common Files,Micrium uC/OS-III Port
Common 1.7.8	CycloneCommon		Resource Manag...	<input type="checkbox"/>		Compiler ...	⚠	Common Files,FLASH Resource Manager
Common 1.7.8	CycloneCommon		String Helper Ro...	<input type="checkbox"/>		Compiler ...	⚠	Common Files,String and Path Manipulation Helper Routines

树视图显示了以下信息：

- Oryx-Embedded : Middleware/CycloneCommon : 1.7.8 : CycloneCommon : Date Time Helper Routines/
 - require CycloneCommon : Compiler Support (must be solved) (following solution(s) found)
 - [Oryx-Embedded] [Middleware/CycloneCommon] [1.7.8] [CycloneCommon] [Compiler Support/] (not selected)

3. 选择推荐的软件组件解析该条件。条件状态变为已解析，，（参见图 65）。

图65. 已解析的软件组件条件

Pack Version	Class	Pack Action	Group/Subgroup	Select	Variant	Condition	Status	Description
1.0.0	CMSIS		RTOS/LiteOS	<input type="checkbox"/>		Device st...		CMSIS LiteOS API for Cortex-M4
1.4.1	Network	Install						IwIP is a light-weight implementation of the TCP/IP protocol suite
CycloneCommon 1.7.8	CycloneCommon	Installed						Middleware Package (CycloneTCP, CycloneSSL and CycloneCrypto)
CycloneCommon 1.7.8	CycloneCommon		Compiler Support/	<input checked="" type="checkbox"/>				Common Files,Compiler support
CycloneCommon 1.7.8	CycloneCommon		Date Time Helper...	<input type="checkbox"/>		Compiler ...		Common Files,Date and Time Manipulation Helper Routines
CycloneCommon 1.7.8	CycloneCommon		FS Port/FatFs Port	<input type="checkbox"/>		Compiler ...		Common Files,FatFs Port
CycloneCommon 1.7.8	CycloneCommon		RTOS Port/CMSI...	<input type="checkbox"/>		CMSIS-R...		Common Files,CMSIS-RTOS Port
CycloneCommon 1.7.8	CycloneCommon		RTOS Port/CMSI...	<input type="checkbox"/>		CMSIS-R...		Common Files,CMSIS-RTOS2 Port
CycloneCommon 1.7.8	CycloneCommon		RTOS Port/Chibi...	<input type="checkbox"/>		ChibiOS ...		Common Files,ChibiOS Port
CycloneCommon 1.7.8	CycloneCommon		RTOS Port/Free...	<input type="checkbox"/>		FreeRTOS...		Common Files,FreeRTOS Port
CycloneCommon 1.7.8	CycloneCommon		RTOS Port/None	<input type="checkbox"/>		No RTOS ...		Common Files,No RTOS
CycloneCommon 1.7.8	CycloneCommon		RTOS Port/RTX ...	<input type="checkbox"/>		RTX Dep...		Common Files,Keil RTX Port
CycloneCommon 1.7.8	CycloneCommon		RTOS Port/embed...	<input type="checkbox"/>		embOS D...		Common Files,Segger embOS Port
CycloneCommon 1.7.8	CycloneCommon		RTOS Port/uCOS...	<input type="checkbox"/>		uCOS-II ...		Common Files,Micrium uC/OS-II Port
CycloneCommon 1.7.8	CycloneCommon		RTOS Port/uCOS...	<input type="checkbox"/>		uCOS-III ...		Common Files,Micrium uC/OS-III Port
CycloneCommon 1.7.8	CycloneCommon		Resource Manag...	<input type="checkbox"/>		Compiler ...		Common Files,FLASH Resource Manager
CycloneCommon 1.7.8	CycloneCommon		String Helper Ro...	<input type="checkbox"/>		Compiler ...		Common Files,String and Path Manipulation Helper Routines

Bottom pane shows a tree view of selected components: Oryx-Embedded : Middleware/CycloneCommon : 1.7.8 : CycloneCommon : Date Time Helper Routines/ require CycloneCommon : Compiler Support (solved) [Oryx-Embedded] [Middleware/CycloneCommon] [1.7.8] [CycloneCommon] [Compiler Support/] (selected)

4. 一旦完成应用所需软件组件的选择（参见图 66），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。软件组件的详细信息可以在工具提示中找到（参见图 67）。

图66. 选择附加软件组件

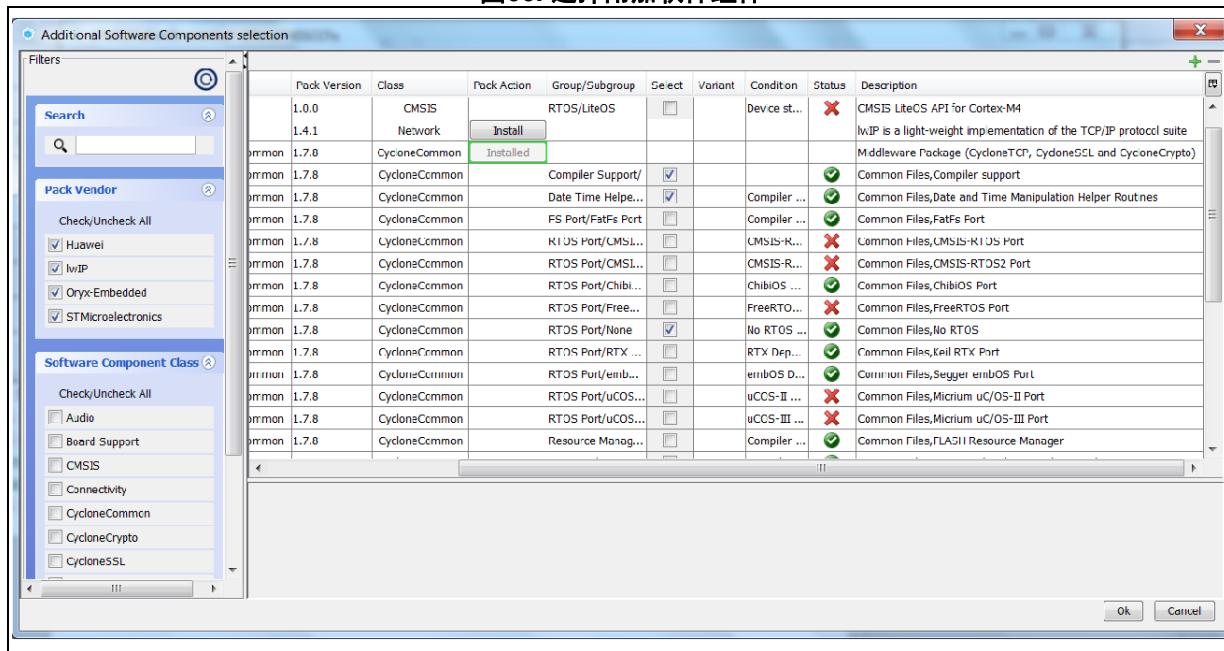
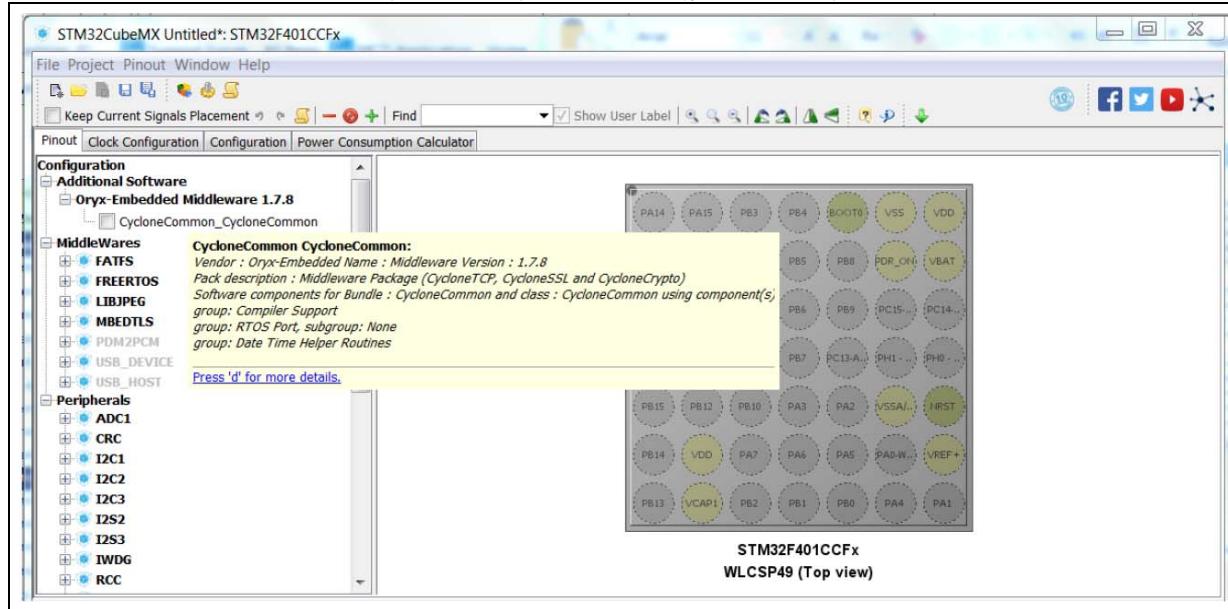


图67. 附加软件组件 - 更新后的树状视图



5.11 “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68. “关于”窗口



5.12 “引脚布局”视图

引脚布局视图帮助用户根据外设/中间件的选择和它们的操作模式配置MCU引脚。

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。

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布局配置。STM32CubeMX突出显示无法自动解决的冲突。

引脚布局视图左侧面板显示**外设和中间件树**，而右侧面板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。

STM32CubeMX提供两种微控制器配置方法：

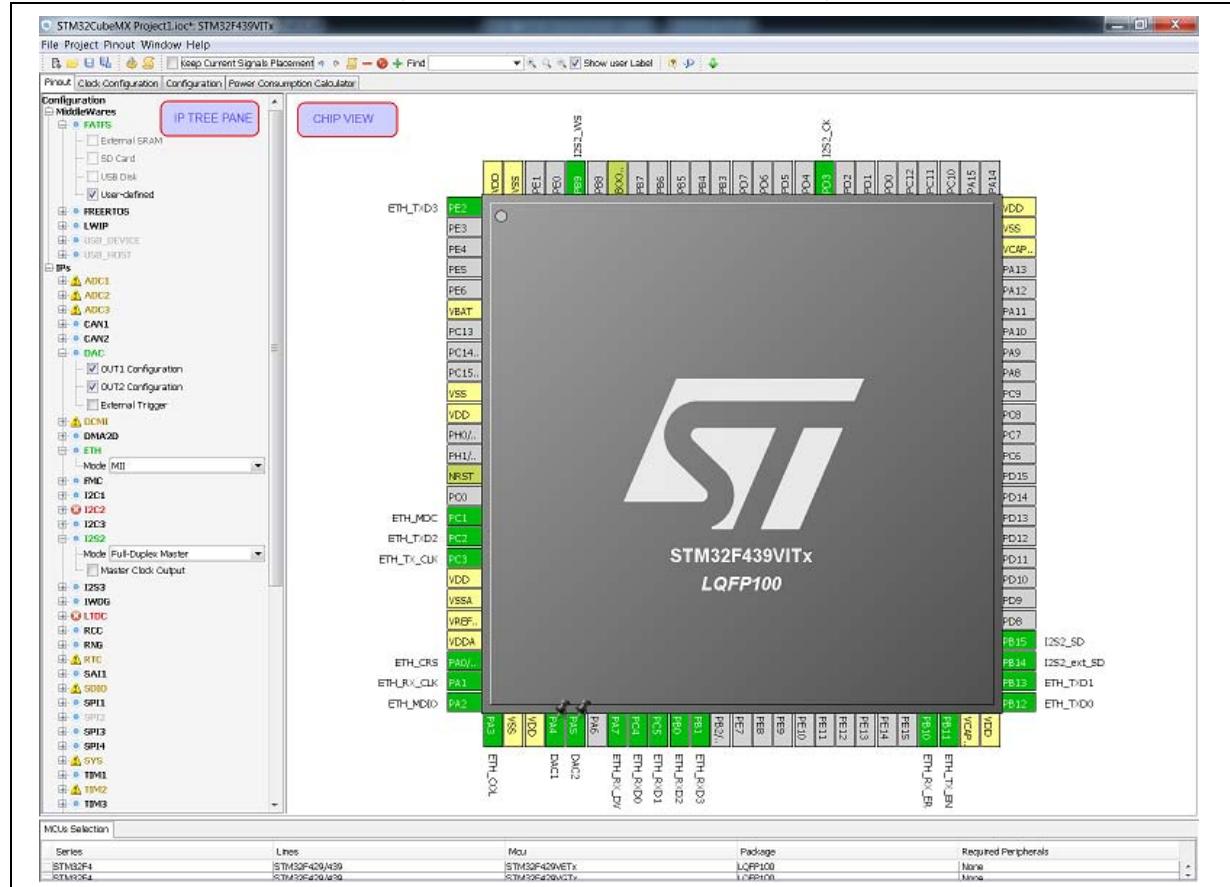
- 从**外设和中间件树**点击外设名称并选择操作模式（参见[第 5.12.1 节](#)）。
- 对于高级用户，可以点击**芯片**视图上的单个引脚，以便手动将其映射到某个外设功能（参见[第 5.12.2 节](#)）。

此外，选择**引脚布局 > 设置未使用的GPIO**允许在给定的GPIO模式下一次性配置多个未使用的引脚。

注：**引脚布局**视图将自动刷新以显示生成的引脚布局配置。

当**引脚布局**视图处于激活状态时，与引脚布局有关的菜单和快捷键可用（关于**引脚布局**菜单的详细信息，请参阅菜单部分）。

图69. STM32CubeMX“引脚布局”视图



5.12.1 “外设和中间件树”面板

在此面板中，用户可在与应用相对应的模式中选择外设、服务（DMA、RCC...）、中间件。此面板显示当前选定处理器可用的所有外设、服务（DMA、RCC）和中间件。

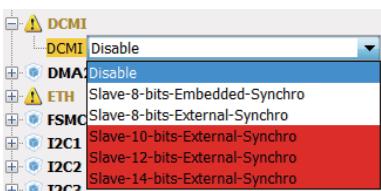
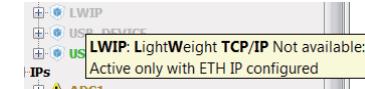
用户可以在与项目相关的模式中选择一个集合。

注：也可以从配置视图访问外设树面板。但是，只有不影响引脚布局的外设和中间件模式才能通过此菜单进行配置。

图标和颜色方案

表 10 显示外设和中间件树面板中使用的图标和颜色方案。

表10.“外设和中间件树”面板 - 图标和颜色方案

显示	外设状态
CAN1	外设未配置（没有设置模式），所有模式都可用。
ADC1	外设已配置（至少设置了一个模式），所有其他模式都可用。
ADC3	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。
ADC2	外设未配置（没有设置模式），至少有一个模式不可用。
ETH	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。
CAN1 Mode Disable	可用的外设模式配置以纯黑色显示。
	黄色警告图标表示至少有一个模式配置不再可用。
	当给定的外设模式不再保留任何配置时，该外设以红色突出显示。
	有些模式取决于其他外设或中间件模式的配置。工具提示解释了条件未满足时的依赖关系。

5.12.2 “芯片”视图

芯片视图为所选的料号显示：

- 采用特定封装（BGA、LQFP...）的MCU
 - 以图形方式表示其引脚布局，每个引脚都用其名称（例如，PC4： GPIO端口C的引脚4）和当前功能分配（例如，ETH MII RXD0）表示，可参见图 70获取示例。

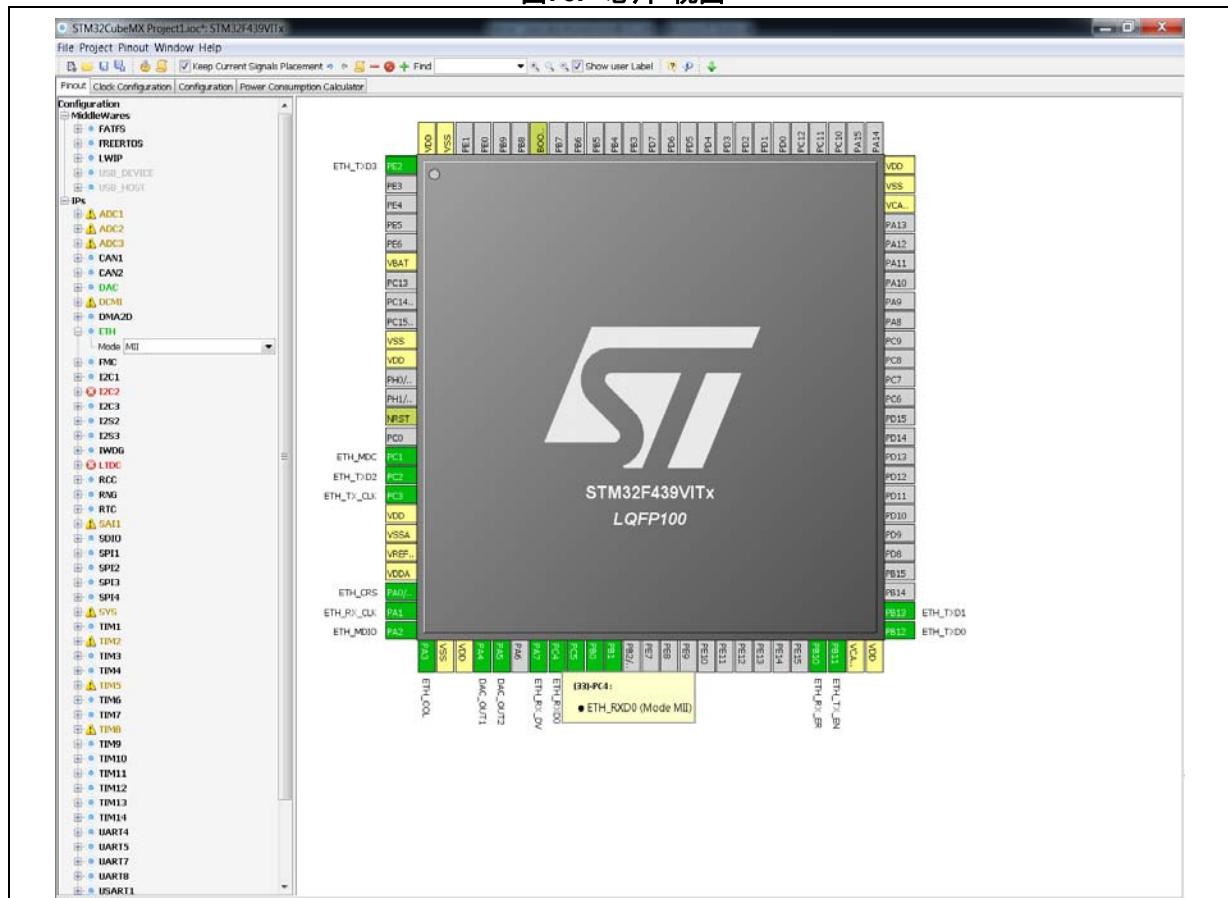
芯片视图会自动刷新，以匹配通过外设树执行的用户配置。它显示引脚当前的配置状态。

通过芯片视图而不是外设面板分配引脚，需要对MCU有深入的了解，因为每个引脚都可以分配一个特定的功能。

提示和技巧

- 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（参见表 5）。
 - 使用引脚布局 > 通用CSV引脚布局文本文件将引脚布局配置导出为文本格式。
 - 某些基本控件，例如引脚一致性的保障块，都是内置式控件。详细信息，请参见[附录A: STM32CubeMX引脚分配规则](#)。

图70. “芯片”视图



图标和颜色方案

表 11 显示芯片视图中使用的图标和颜色方案。

表11. STM32CubeMX “芯片”视图 - 图标和颜色方案

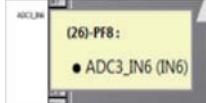
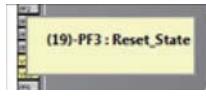
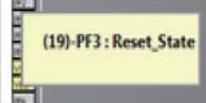
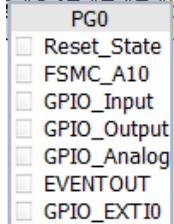
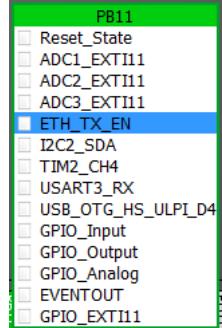
显示	引脚信息
  	<p>工具提示指示选择的引脚当前配置：备用功能名称、复位状态或GPIO模式。 将鼠标移到引脚名称上以使其显示。 当引脚具有与当前所选功能相对应的备用引脚时，弹出消息提示用户执行“CTRL + click”操作以使其显示。 可用的备用引脚以蓝色突出显示。</p>
	<p>可为给定引脚选择的备用功能列表。默认情况下，没有配置备用功能（引脚处于复位状态）。 单击引脚名称以显示列表。</p>
	<p>当数字外设功能已映射到引脚时，它将以蓝色突出显示。 当它对应于配置良好的外设模式时，列表标题显示为绿色。</p>

表11. STM32CubeMX “芯片”视图 - 图标和颜色方案（续）

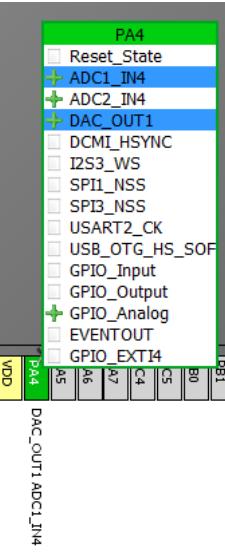
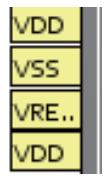
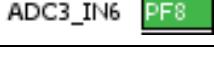
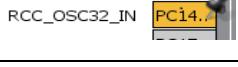
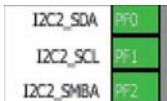
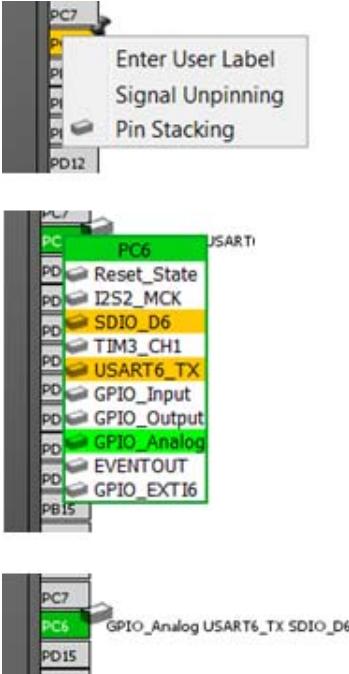
显示	引脚信息
	模拟信号可以共享同一个引脚。例如，在PA4上首先启用了DAC_OUT1，然后将ADC_IN4分配给该引脚。
	启动和复位引脚以卡其色突出显示。其配置不能更改。
	电源专用引脚以黄色突出显示。其配置不能更改。
	未配置的引脚显示为灰色（默认状态）。
	当一个信号分配对应于一个没有歧义的外设模式时，引脚颜色切换为绿色。
	当信号分配不对应于有效的外设模式配置时，引脚以橙色显示。需要配置额外的引脚以实现有效的模式配置。

表11. STM32CubeMX “芯片”视图 - 图标和颜色方案（续）

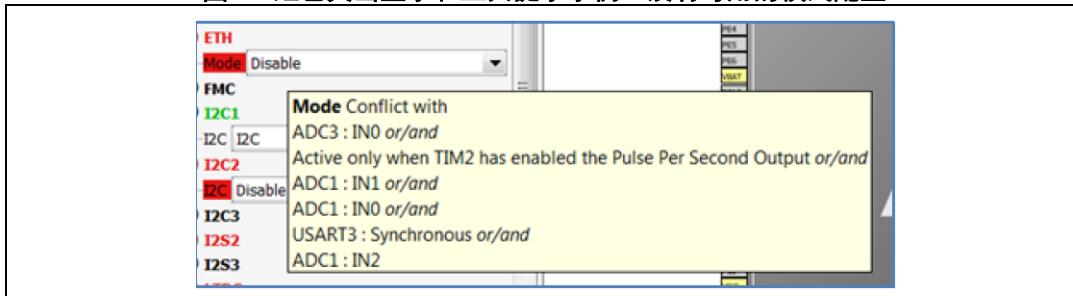
显示	引脚信息
	当一个信号分配对应于一个没有歧义的外设模式时，引脚以绿色显示。 例如，将PF2引脚分配给I2C2_SMBA信号，可以明确地与I2C2模式匹配，并且STM32CubeMX自动配置其他引脚（PF0和PF1）以完成引脚模式配置。
	引脚堆叠功能允许开发板设计者在给定的引脚上指定多个信号，对应于板支持的多个配置（通常通过焊桥选择）。 右键单击引脚以选择引脚堆叠。然后，单击以选择要叠加在该引脚上的信号。

工具提示

将鼠标移到无法使用或部分可用的外设和外设模式上，以显示描述冲突来源的工具提示，即哪些外设正在使用哪些引脚。

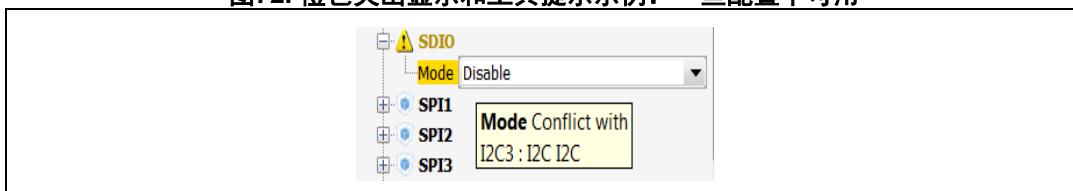
例如（请参阅[图 71](#)），以太网（ETH）外设不再可用，因为没有剩下可能的模式配置。工具提示指示被分配到该模式所需引脚的信号（ADC1-IN0信号、USART3同步信号，等等）。

图71. 红色突出显示和工具提示示例：没有可用的模式配置



在下一个示例（参见图 72）中，SDIO外设部分可用，因为它至少有一种模式不可用：必要的引脚已经分配给I2C3外设的I2C模式。

图72. 橙色突出显示和工具提示示例：一些配置不可用



在最后一个示例（参见图 73）中，I2C2外设不可用，因为没有模式功能可用。每个功能都有一个工具提示，其中所有重新映射的引脚都已分配（USART3同步模式）。

图73. 工具提示示例：全部配置不可用



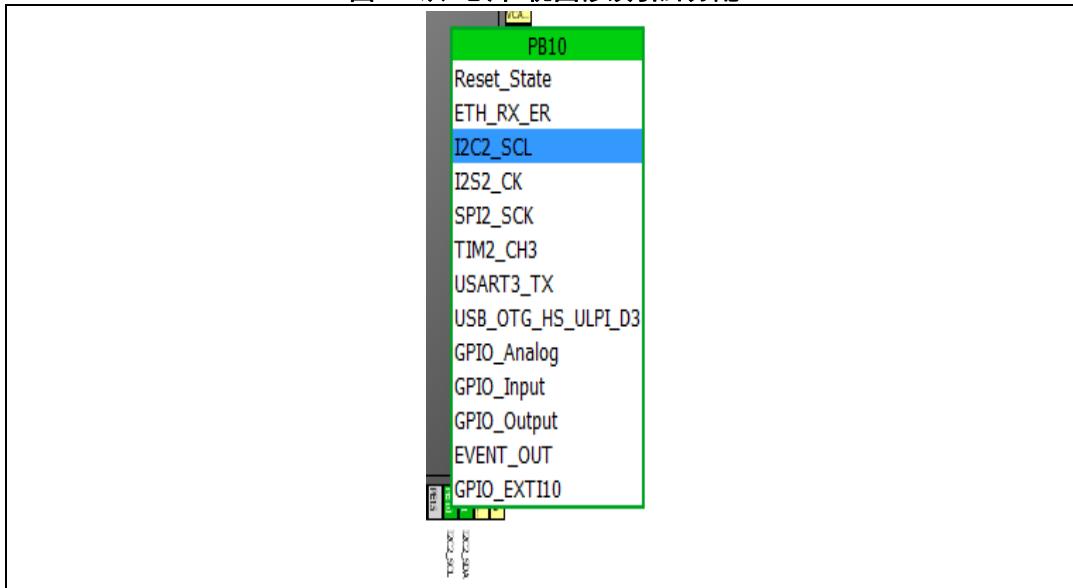
5.12.3 “芯片”视图高级操作

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. 在芯片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 74）。
2. 单击以选择要分配给引脚的新功能。

图74. 从“芯片”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键并点击芯片视图中的引脚。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“芯片”视图执行的引脚分配覆盖以前的任何分配。

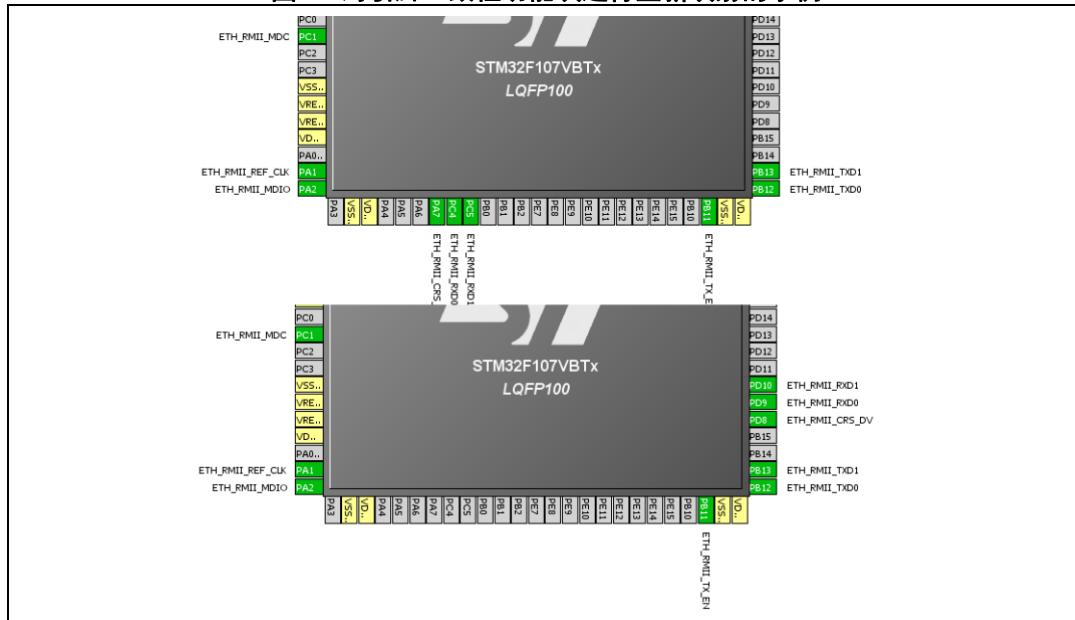
手动重新映射有歧义的目标引脚

对于具有引脚一致性功能块的MCU（STM32F100x/ F101x/ F102x/ F103x和STM32F105x/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 75 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75. 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红色或橙色突出显示，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布置框并让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。

5.12.4 保持当前信号布置

当引脚布局视图已选择时，可从工具栏访问该复选框（参见图 37 和表 5）。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布置复选框。

“保持当前信号布置”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布局配置冲突的新请求（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**芯片**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**芯片**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布置配置（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“保持当前信号布置”，STM32CubeMX也不会移动GPIO_ 功能（除GPIO_EXTI 功能外）。

5.12.5 在引脚上锁定和标记信号

STM32CubeMX有一个功能，用户可通过它选择性地将信号锁定到引脚：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。也可给信号贴上标签：用户标签用于代码生成（请参阅5.1部分获取详细信息）。

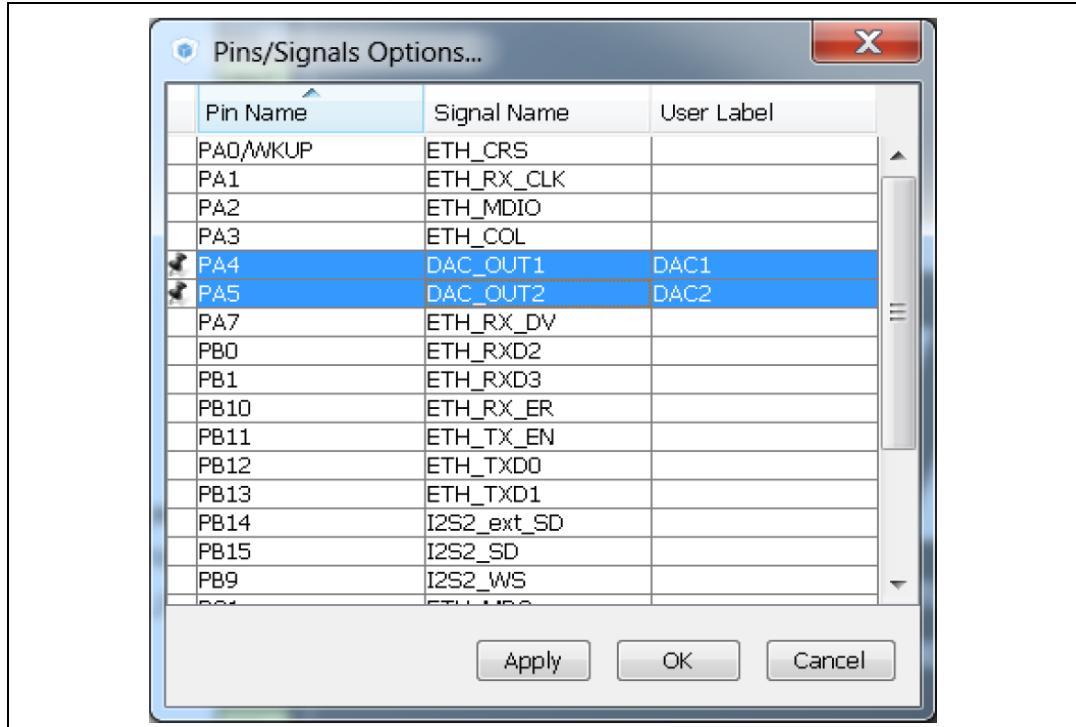
STM32CubeMX有为用户提供了一项功能以选择性地锁定信号到引脚。这样可防止STM32CubeMX在解决冲突时自动将锁定的信号移动到其他引脚。用于代码生成的标签也可以分配给信号（参阅[第 6.1 节](#)获取详细信息）。

有几种方法可以锁定、取消锁定和标记信号：

1. 从**芯片**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“保持用户布置”选项。

- c) 选择输入用户标签以指定此信号的用户定义标签。新标签将替换芯片视图中的默认信号名称。
2. 从引脚布局菜单，选择引脚/信号选项
 “引脚/信号选项”窗口（参见图 76）列出所有已经配置的引脚。
 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”。

图76. “引脚/信号选项”窗口



- 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，仍然可以从芯片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。

5.12.6 设置HAL时基源

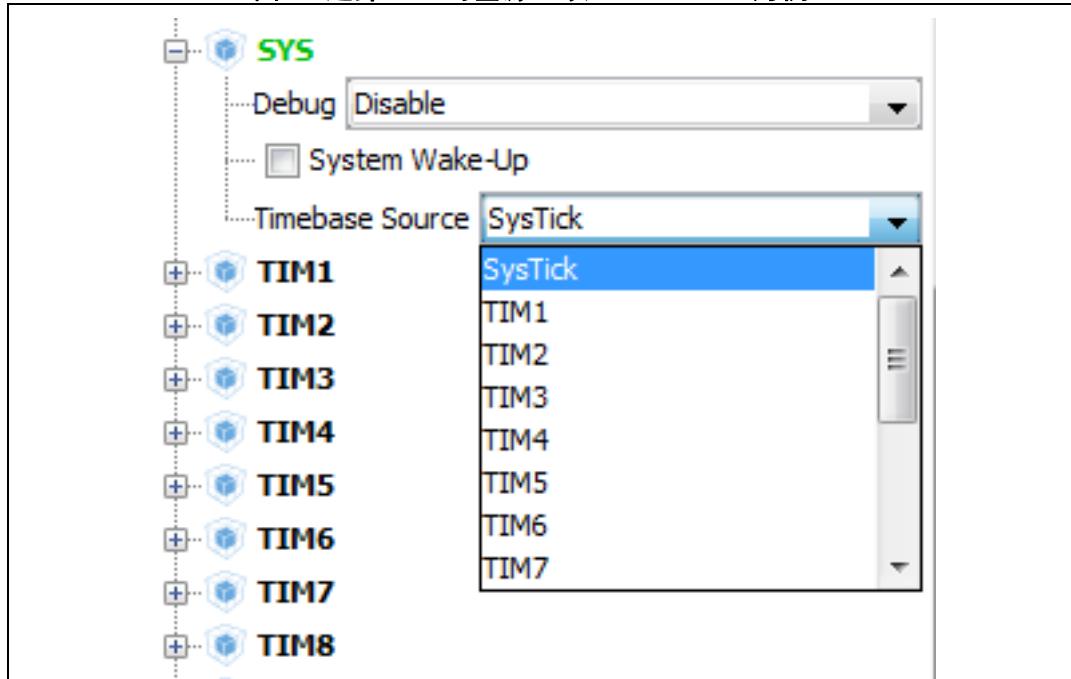
默认情况下，STM32Cube HAL 是围绕一个独特的时基源构建的，也就是 Arm®-Cortex® 系统时钟（SysTick）。

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，请访问**外设和中间件树**面板中的系统外设，并在可用的时钟源（SysTick、TIM1、TIM2）中选择一个时钟...（参见图 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. TIM2被选为HAL时基源

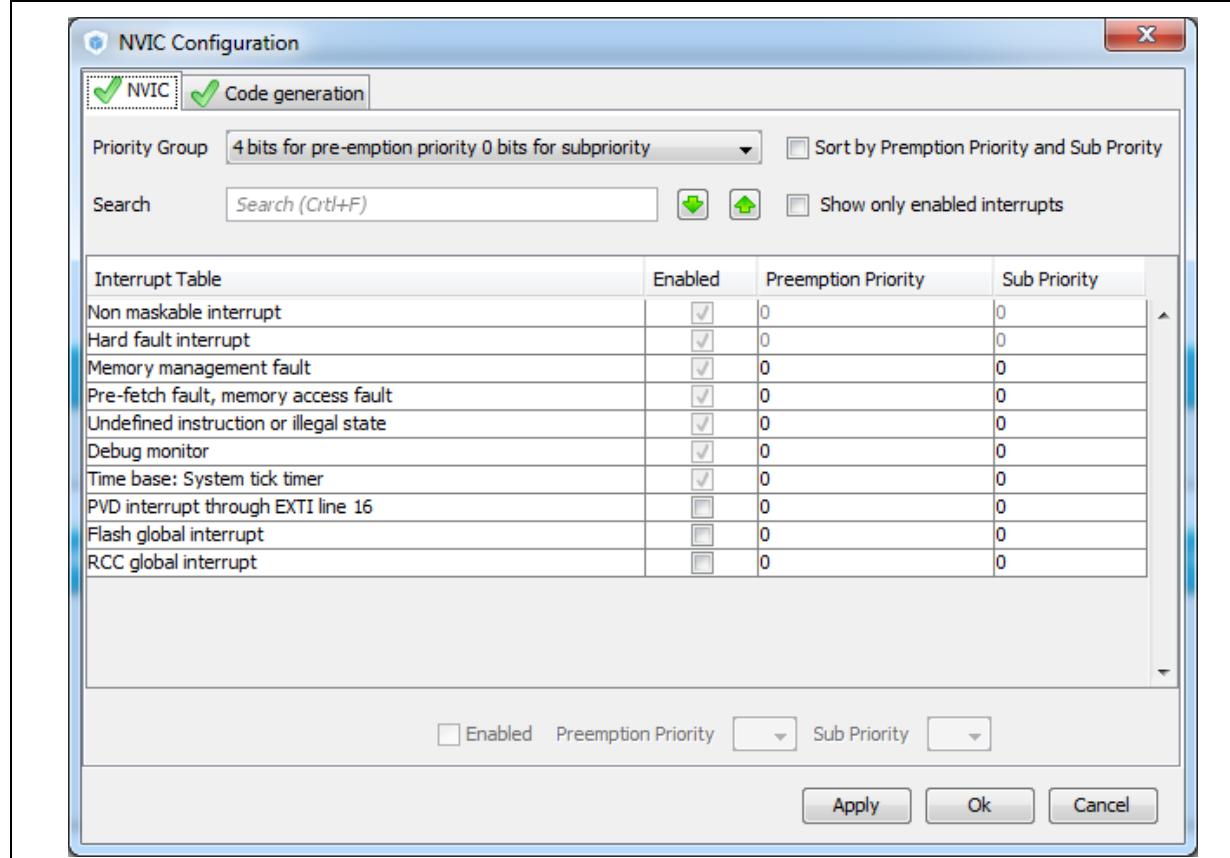


如下例所示，HAL时基源的选择和FreeRTOS的使用会影响生成的代码。

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



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();
}
```

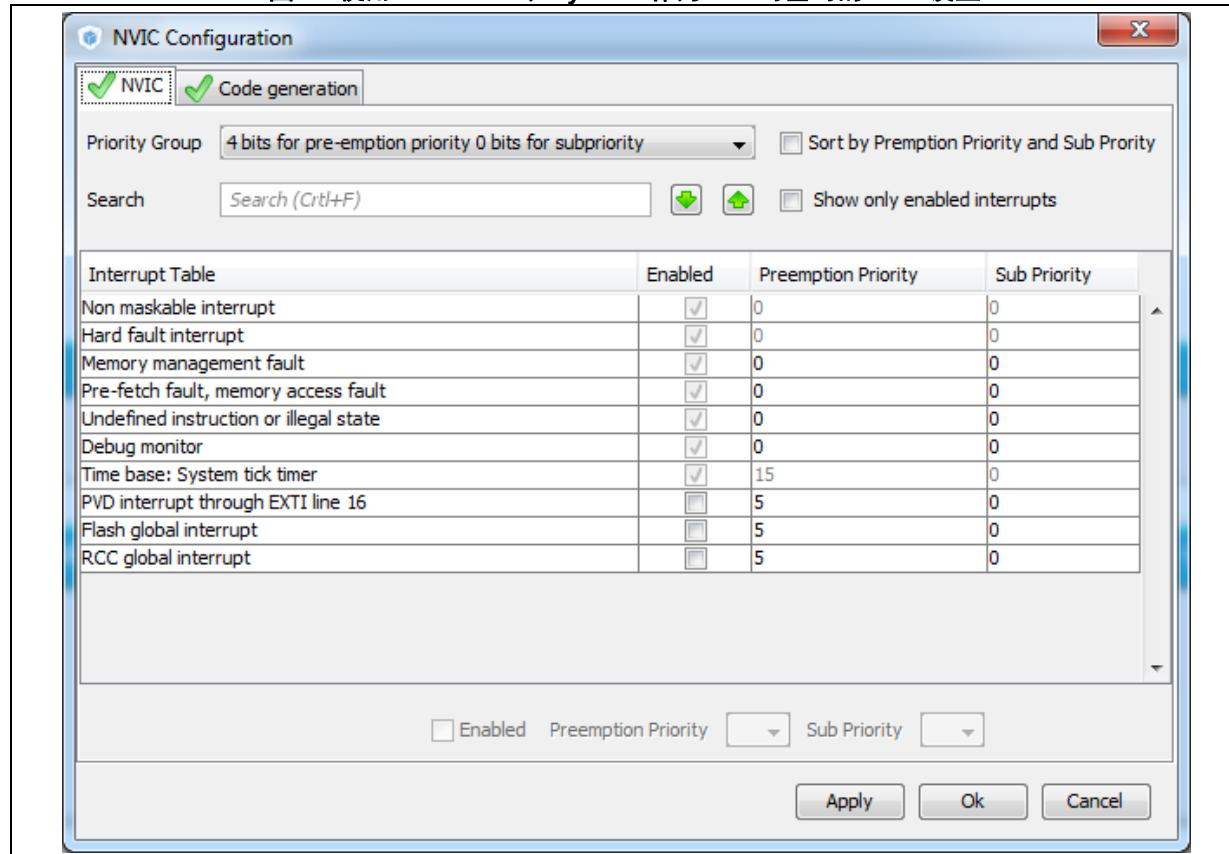
```
HAL_SYSTICK_IRQHandler();
/* 用户代码开始 SysTick_IRQn 1 */

/* 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_IRQHandler 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 */

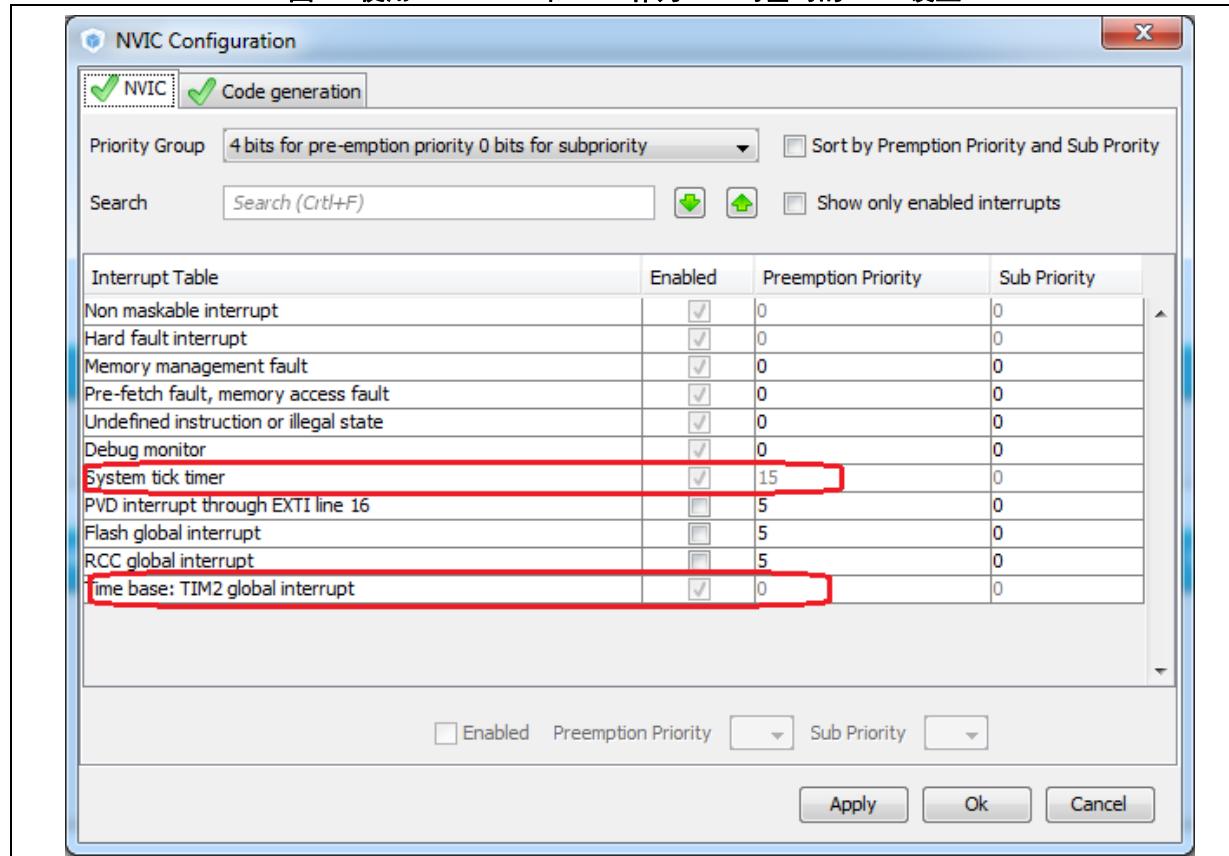
    /* 用户代码结束 SysTick_IRQn 1 */
}
```

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文件以重载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



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。

5.13 配置视图

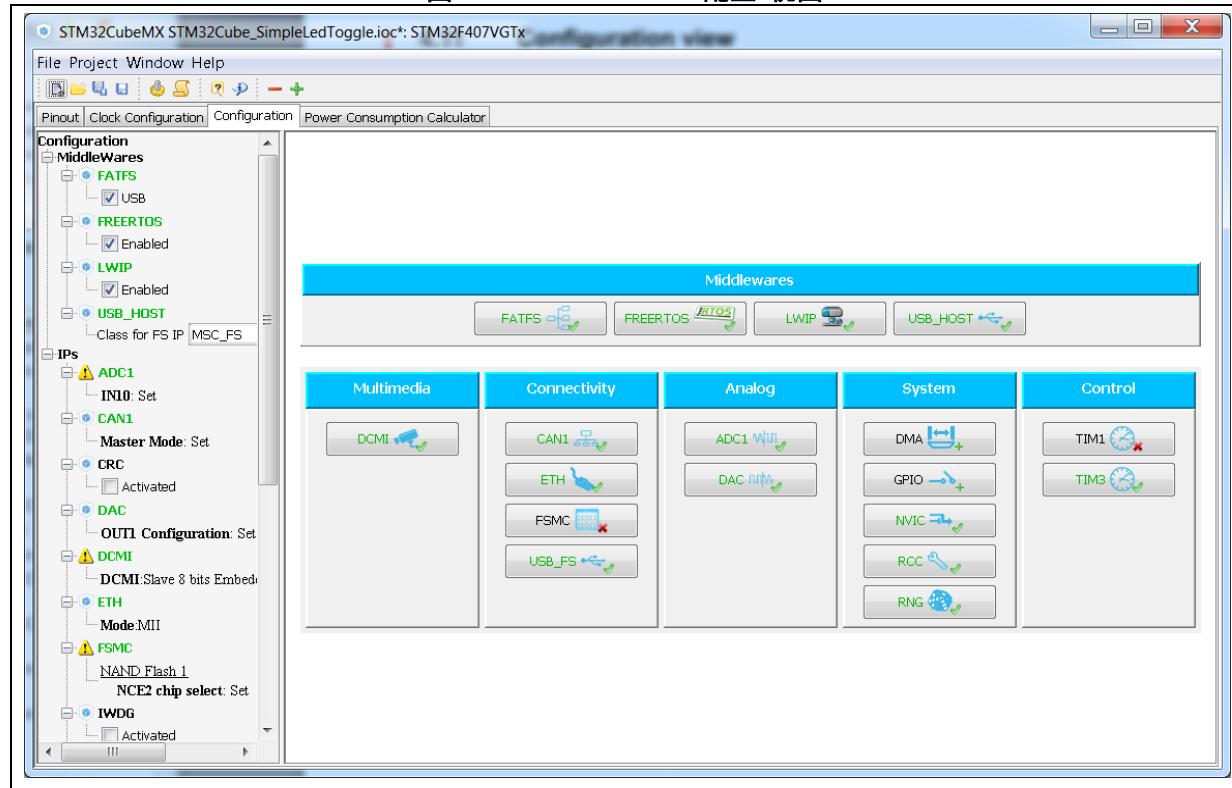
STM32CubeMX 配置窗口（参见图 82）概述所有软件可配置组件：GPIO、外设和中间件。可点击的按钮可用于选择组件初始化参数的配置选项，这些参数将包含在生成的代码中。按钮图标颜色反映配置状态：

- 绿色对号：正确配置
- 警告标记：不完整但仍然是功能配置
- 红色叉号：无效配置。

注：影响引脚布局的GPIO和外设模式只能从引脚布局视图进行设置。它们在“配置”视图中为只读模式。

在该视图中，MCU通过其**外设和中间件树**显示在左侧面板上，通过中间件、多媒体、连接、模拟、系统和控制类别中组织的外设和中间件列表显示在右侧面板上。每个外设实例都有一个专用按钮来编辑其配置：例如，TIM1和TIM3 在图 82 中显示为专用按钮。

图82. STM32CubeMX“配置”视图



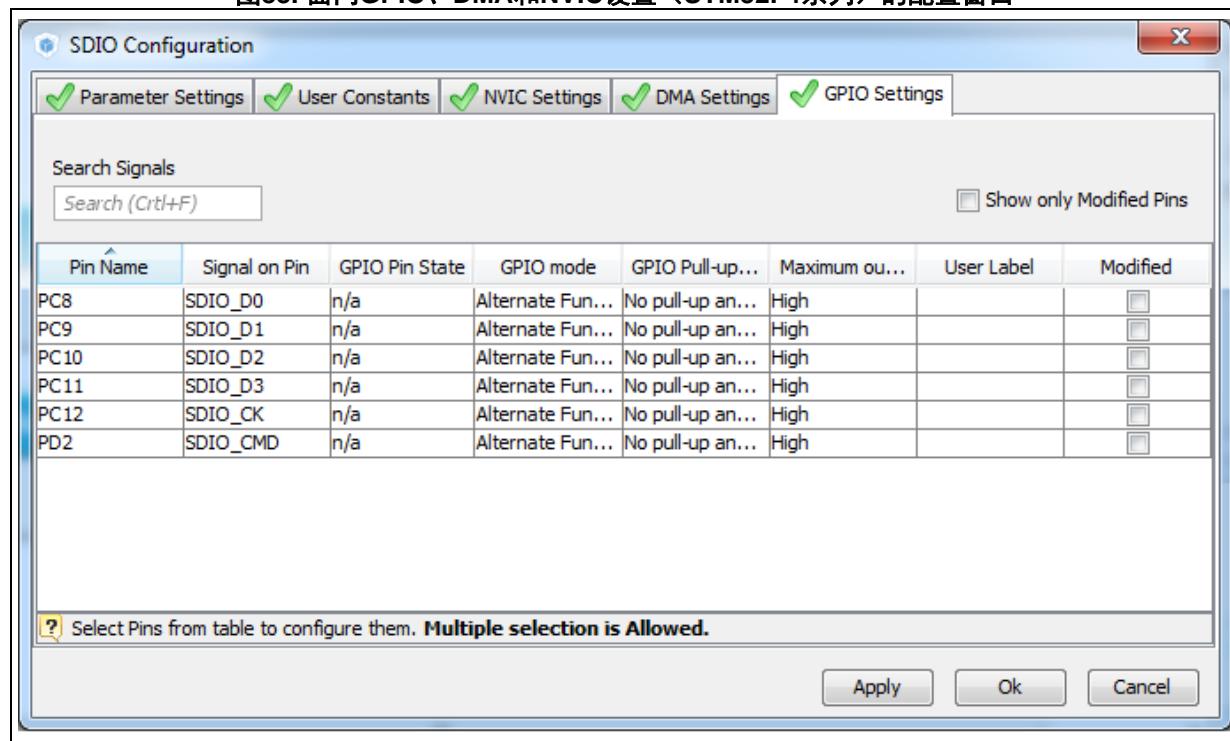
配置按钮关联到配置窗口中的每个外设（参见表 12）。

表12. 外设和中间件配置按钮

格式	外设实例配置状态
	可用但没有完全配置。点击可打开配置窗口。
	已通过默认或用户定义的设置正确配置，允许继续生成相应的初始化C代码。点击可打开配置窗口。
	使用了一些错误的参数值进行了错误的配置。单击可显示以红色突出显示的错误。 其他示例（UART）： Baud Rate 1000000 Bits/s
	说明错误来源的对话框。应在另一个视图中修复。

GPIO、DMA和NVIC设置可以通过像其他外设一样的专用按钮访问，也可以通过使用其外设的其他配置窗口中的选项卡访问（参见图 83）。

图83. 面向GPIO、DMA和NVIC设置（STM32F4系列）的配置窗口



5.13.1 “外设和中间件配置”窗口

从配置面板点击外设实例或中间件名称可打开该窗口。它可用于配置在选定的操作模式中初始化外设或中间件所需的功能参数。此配置用于生成相应的初始化C代码。请参见[图 84](#)获取“外设配置”窗口示例。

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[“NVIC配置”窗口](#)、[“GPIO配置”窗口](#)和[“DMA配置”窗口](#)）。
- **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[“用户常量”配置窗口](#)）。

检测到无效设置，并且：

- 如果用户选择小于最小阈值，则重置为最小有效值，
- 如果用户选择大于最大阈值，则重置为最大有效值，
- 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，
- 以红色突出显示： 1000000 Bits/s

[表 13](#)描述外设和中间件配置按钮和消息。

图84. “外设配置”窗口（STM32F4系列）

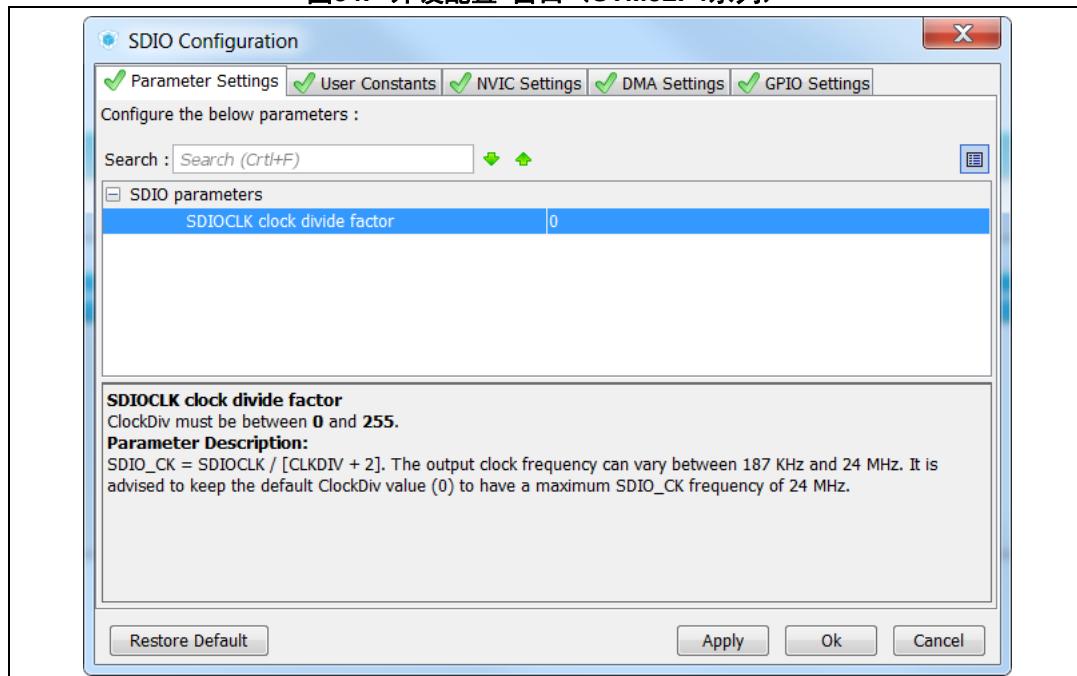


表13. “外设和中间件配置”窗口按钮与工具提示

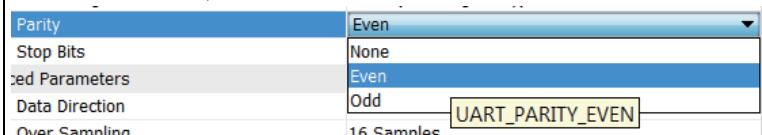
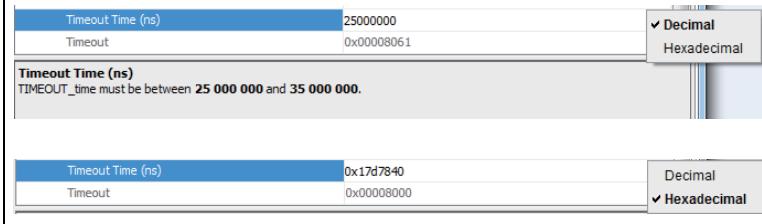
按钮和消息	动作
应用	在不关闭窗口的情况下保存更改
OK	保存并关闭窗口
取消	关闭并重置先前保存的参数设置
恢复默认值	<p>对于选中的外设，整个用户配置被重置为STM32CubeMX默认设置，而不清除“用户常量”选项卡中定义的项目用户常量：在外设参数、GPIO、NVIC和DMA选项卡上恢复STM32CubeMX默认值。</p> <p>注： 对于GPIO/DMA和NVIC窗口，恢复默认值按钮不可用，因为它们涉及多个外设。</p> <p>注： “恢复默认值后单击取消“关闭窗口并保留用户配置。</p> <p>注意： 在恢复默认值后点击应用保存更改。在这种情况下，无法恢复用户原始配置。</p>
	显示和隐藏描述面板
工具提示	<p>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。</p> 
十六进制值与十进制值	<p>单击右侧箭头，选择显示该字段为十六进制值或十进制值：</p> 

表13. “外设和中间件配置”窗口按钮与工具提示（续）

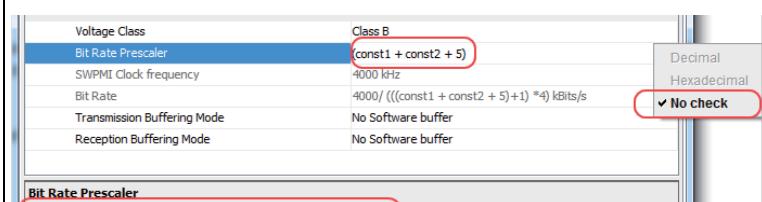
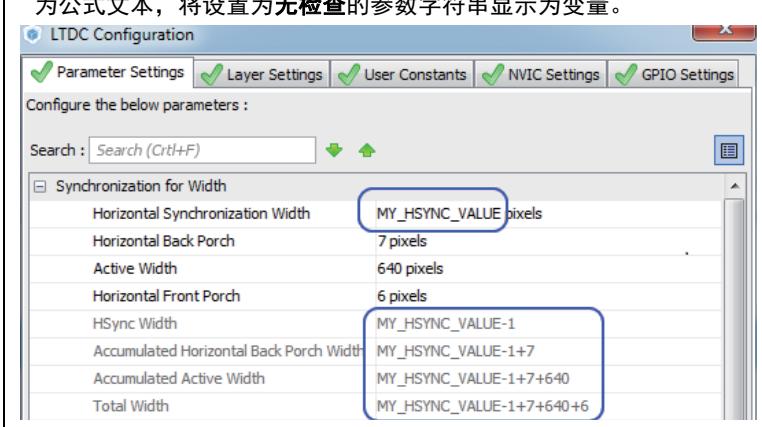
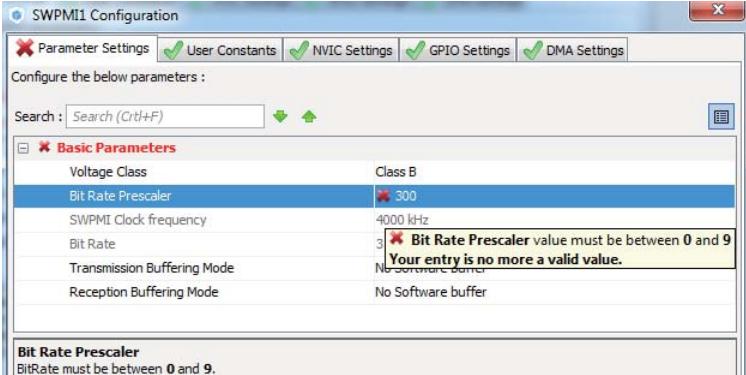
按钮和消息	动作
无检查选项	<p>默认情况下，STM32CubeMX 检查用户输入的参数值是否有效。您可以为给定参数选择无检查选项，以绕开该检查。这就允许输入任何 STM32CubeMX 配置可能不知道的值（比如常量）。</p>  <p>注： 只有整数类型（十六进制或十进制）的参数才能绕过有效性检查。来自预定义的可能值列表的参数和非整数型或文本类型的参数均不能绕过有效性检查。</p> <p>要回到默认模式，即启用有效性检查的十进制或十六进制值，输入十进制或十六进制值并检查相关选项（十六进制或十进制检查）。</p> <p>注意： 当一个参数依赖于另一个设为无检查的参数时：</p> <ul style="list-style-type: none"> 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 如果将其他参数设置为无检查，该值不再自动派生。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。 

表13. “外设和中间件配置”窗口按钮与工具提示（续）

按钮和消息	动作
十进制和十六进制检查工具 提示	如果用户输入的值超出范围，错误将以红色高亮显示，并附有说明性工具提示： 

5.13.2 “用户常量”配置窗口

用户常量窗口用于定义用户常量（参见图 85）。常量自动生成于main.h文件中的STM32CubeMX用户项目中（参见图 86）。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 87）。

图85. “用户常量”窗口

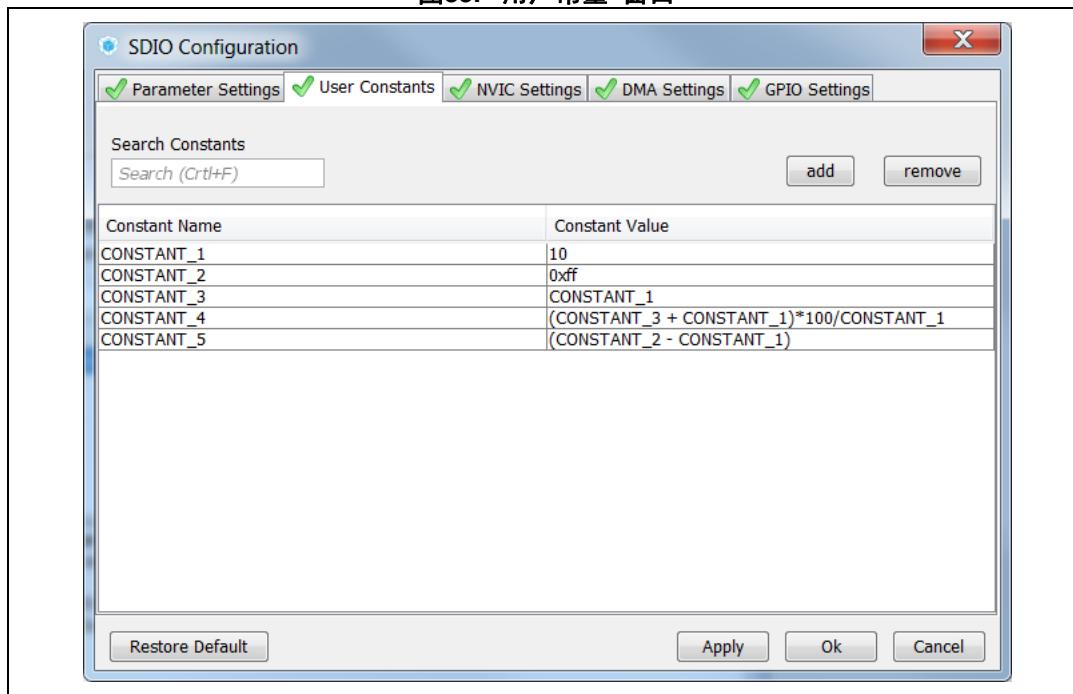
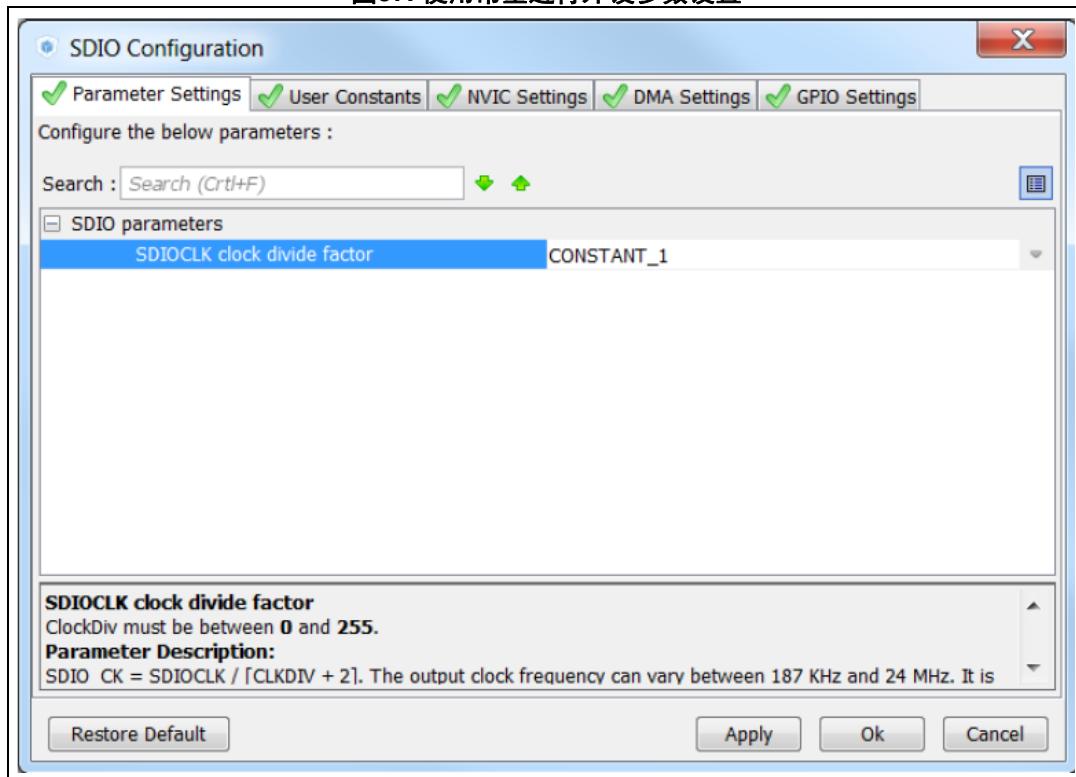


图86. 摘录生成的main.h文件

```
/* 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87. 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窗口并创建新的用户定义常量（参见图 88）。

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。

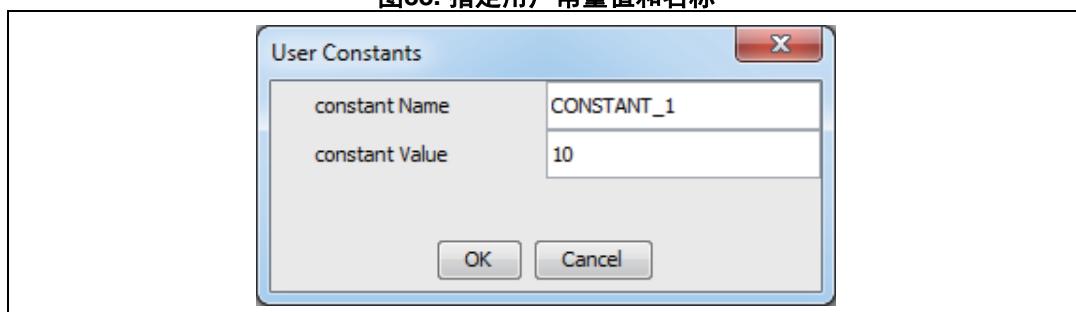
- 一个值

常量值可以是：（参见图 85 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窗口以便编辑。常量名称的变更将会应用到使用了常量的地方。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）。无效的参数设置将以红色突出显示，并带有红色叉号。

图88. 指定用户常量值和名称



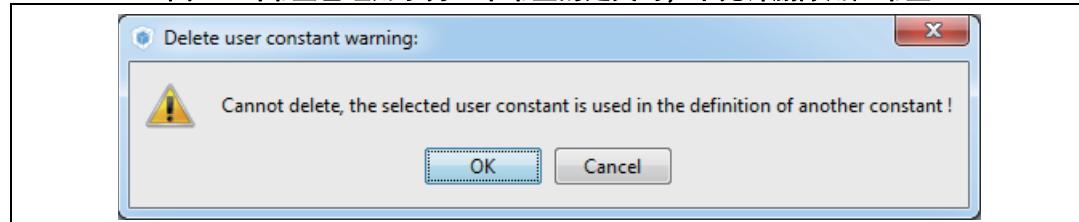
删除用户常量

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：

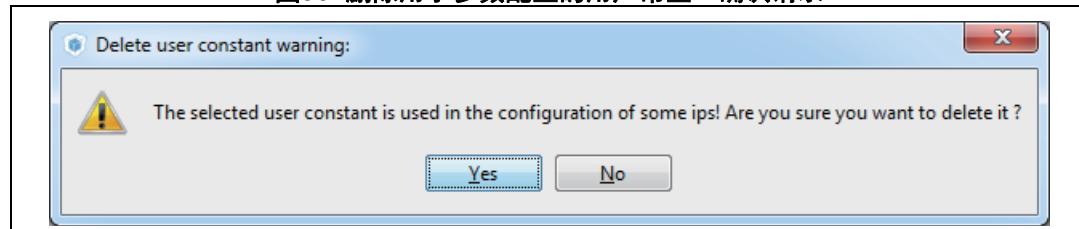
- 当常量用于定义另一个常量时。在这种情况下，弹出窗口将显示解释性消息（参见图 89）。

图89. 当常量已经用于另一个常量的定义时，不允许删除用户常量



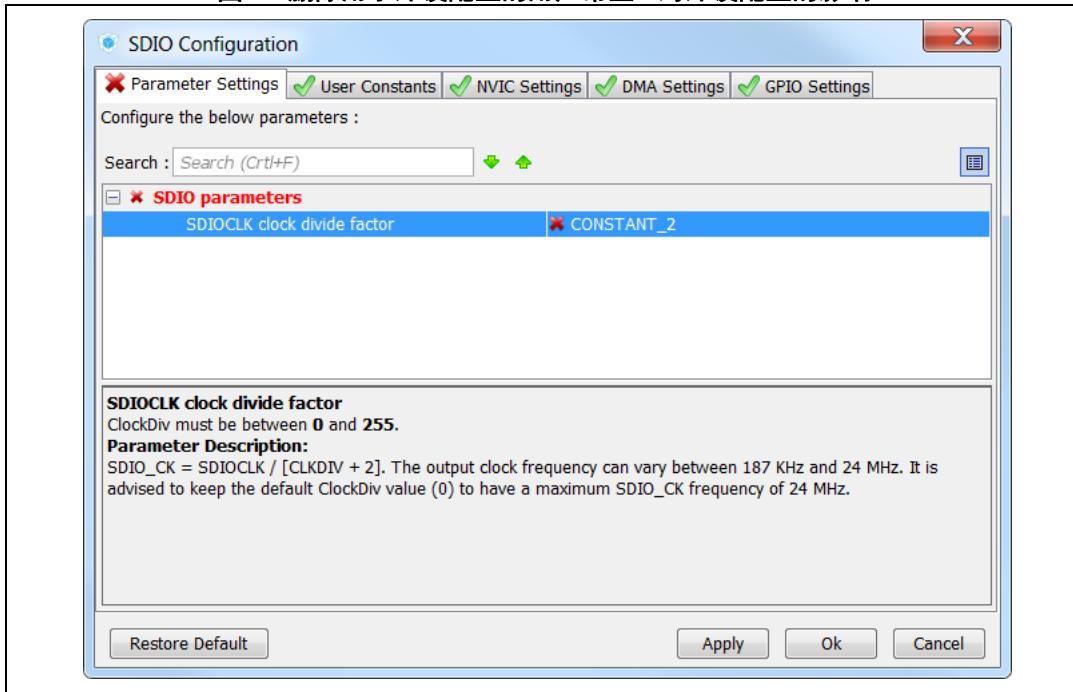
- 当常量用于外设库或中间件库参数的配置时。在这种情况下，请求用户确认删除，因为删除常量将导致无效的外设或中间件配置（参见图 90）。

图90. 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 91）

图91. 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

检索常量字段可在完整的用户常量列表中搜索常量名称或值（参见[图 92](#)和[图 93](#)）。

图92. 在用户常量列表中搜索常量名称

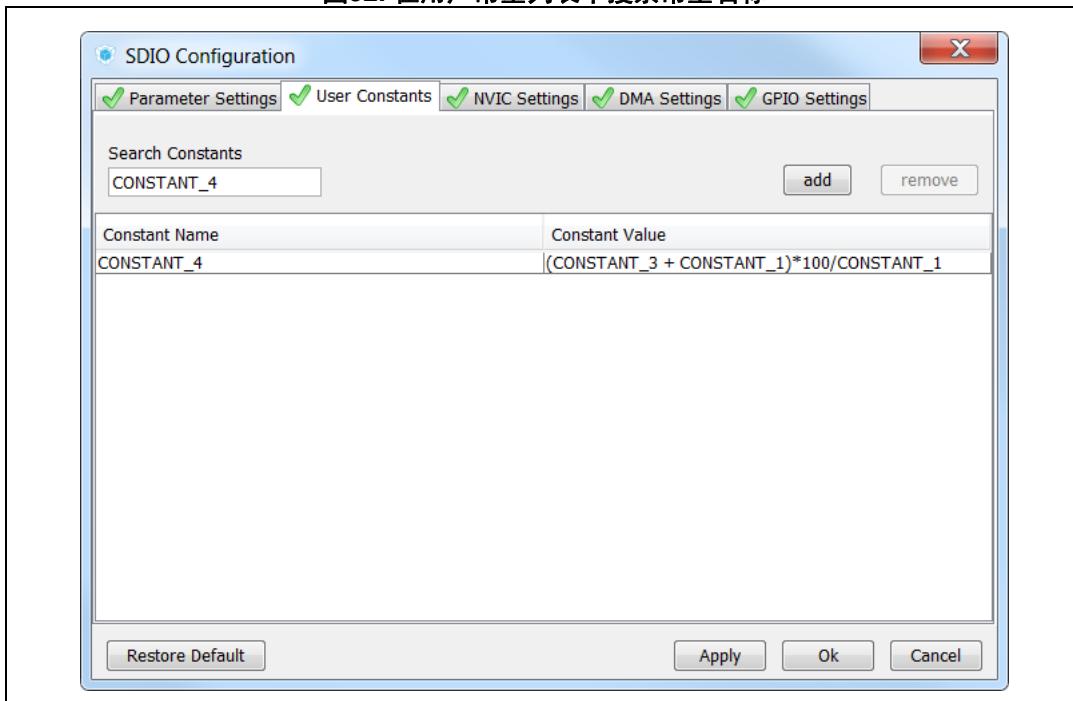
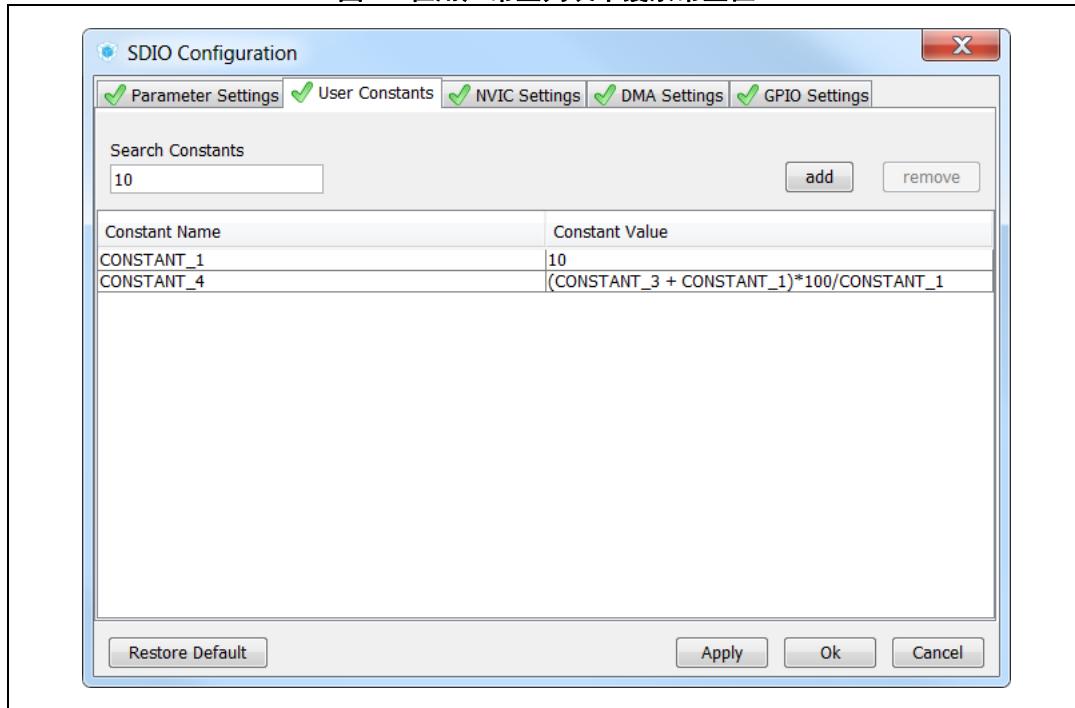


图93. 在用户常量列表中搜索常量值

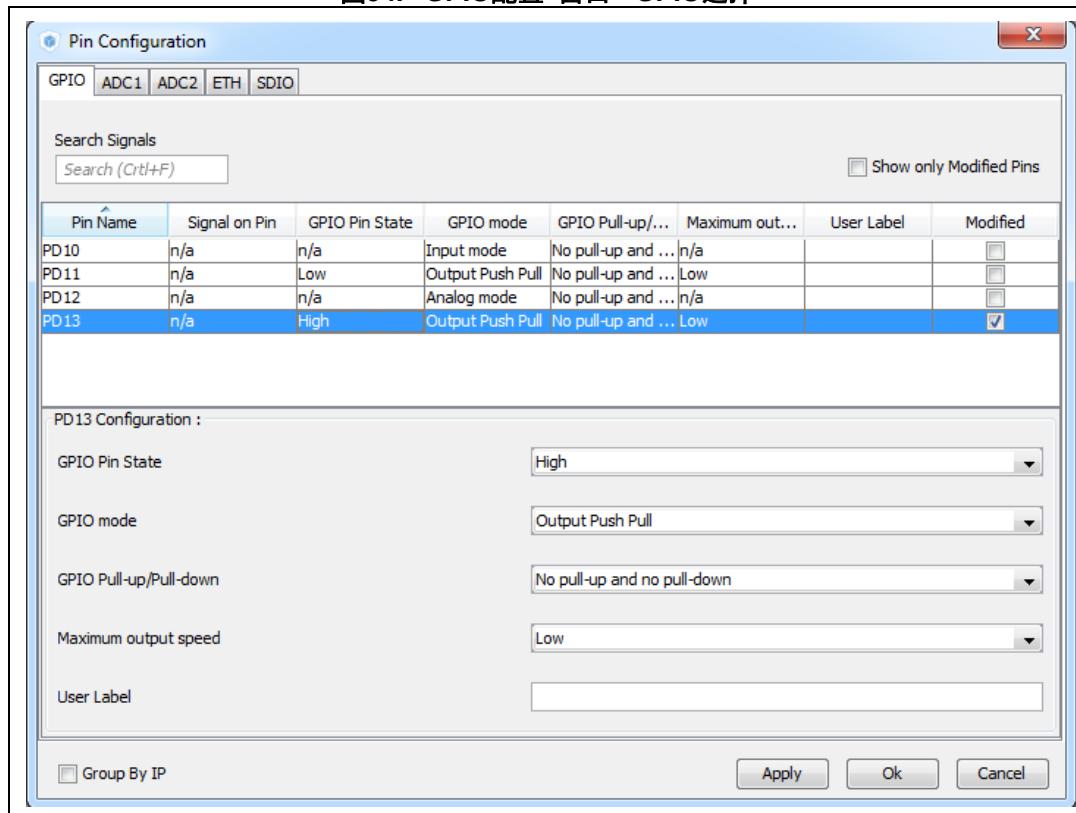


5.13.3 “GPIO配置”窗口

点击配置面板中的**GPIO**打开**GPIO配置**窗口，该窗口可配置GPIO引脚设置（参见图 94）。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GPIO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。生成的代码将相应地更新。

图94. “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（参见图 95）：

- **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）

在引脚布局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。

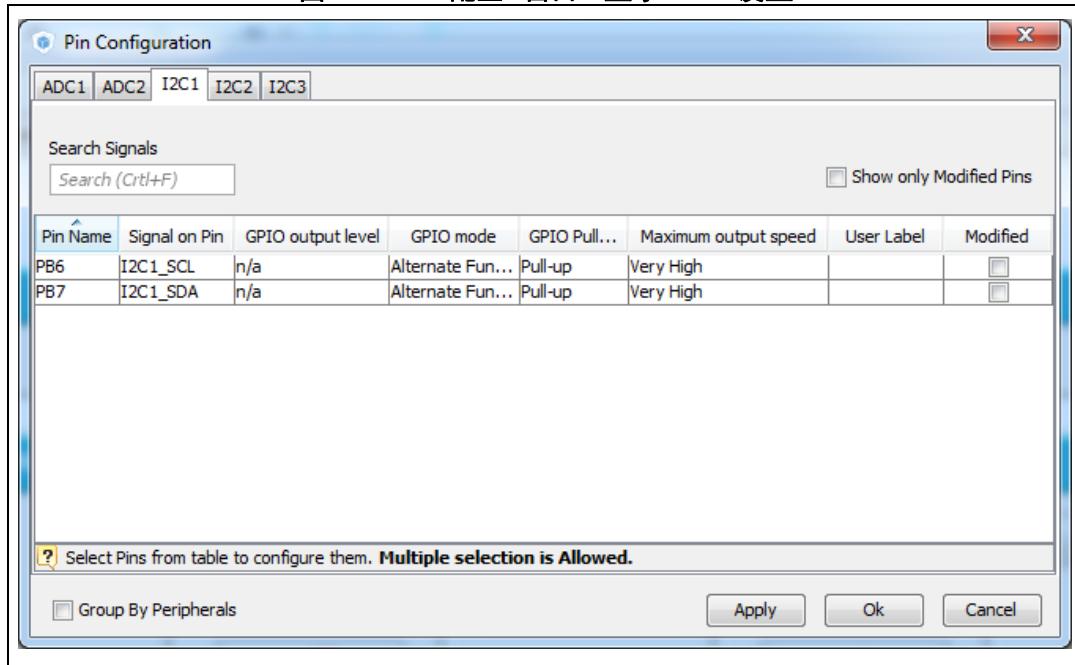
- **GPIO最大输出速度**（仅面向通信外设）

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**

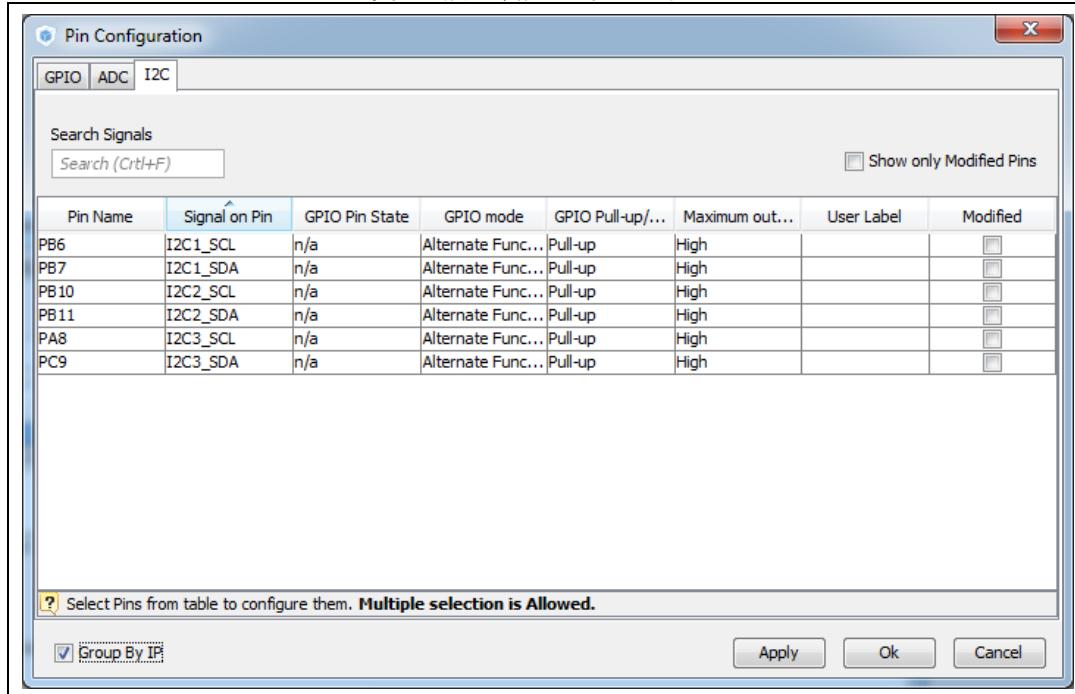
它将默认名称（例如，GPIO_input）更改为用户定义的名称。芯片视图相应地更新。可以通过“查找”菜单在该新名称下找到GPIO。

图95. “GPIO配置”窗口 - 显示GPIO设置



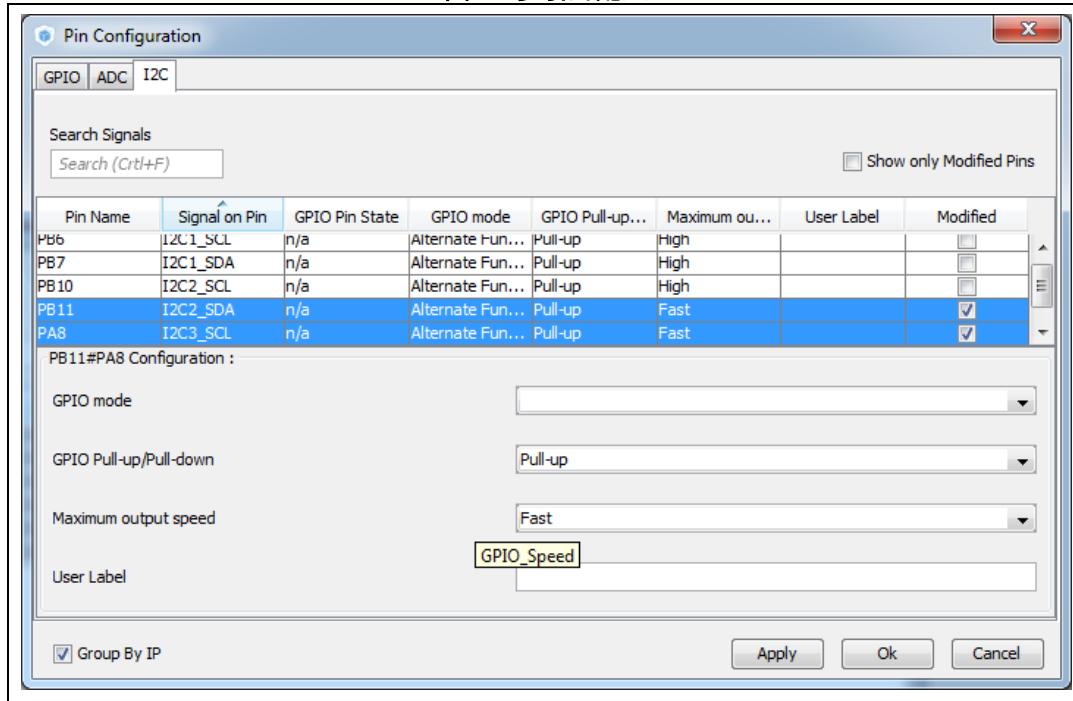
按外设分组复选框允许在同一个窗口下对所有外设实例进行分组（参见图 96）。

图96. 按外设分组的GPIO配置



如图 97 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97. 多引脚配置



5.13.4 “DMA配置”窗口

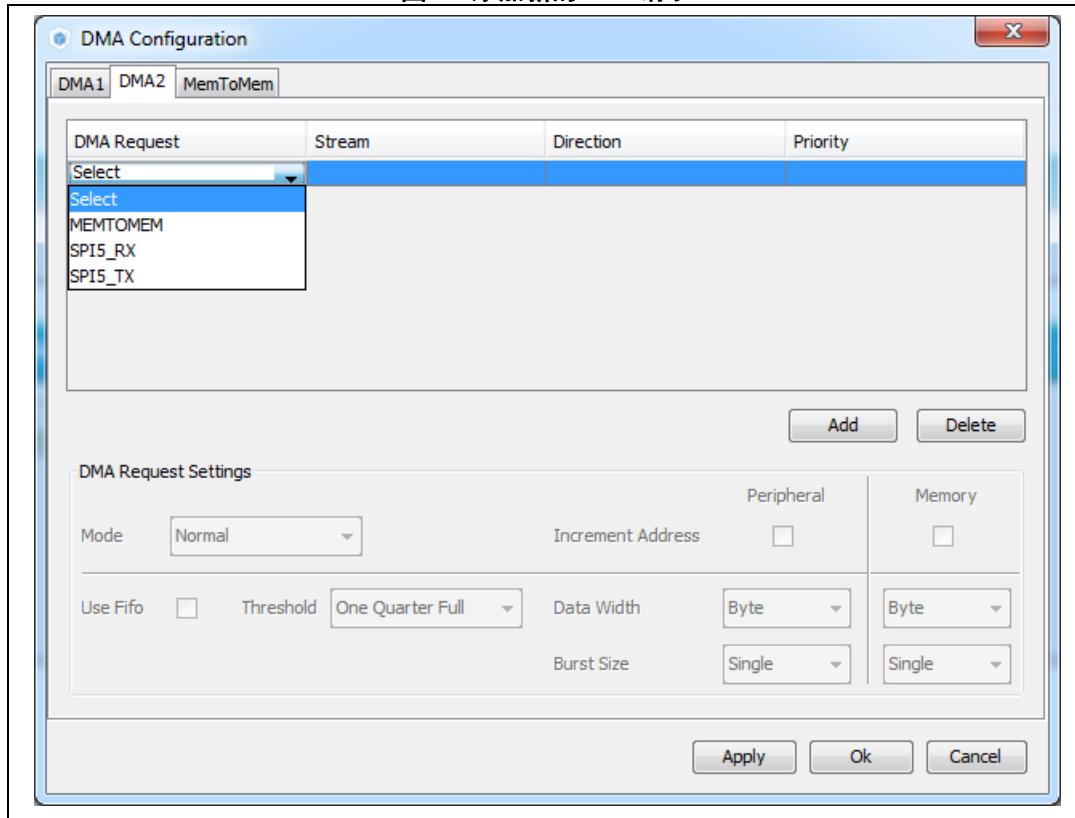
在配置面板中点击**DMA**可打开**DMA配置**窗口。

该窗口允许配置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：
USB或以太网等部分外设拥有自己的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 98）。

图98. 添加新的DMA请求

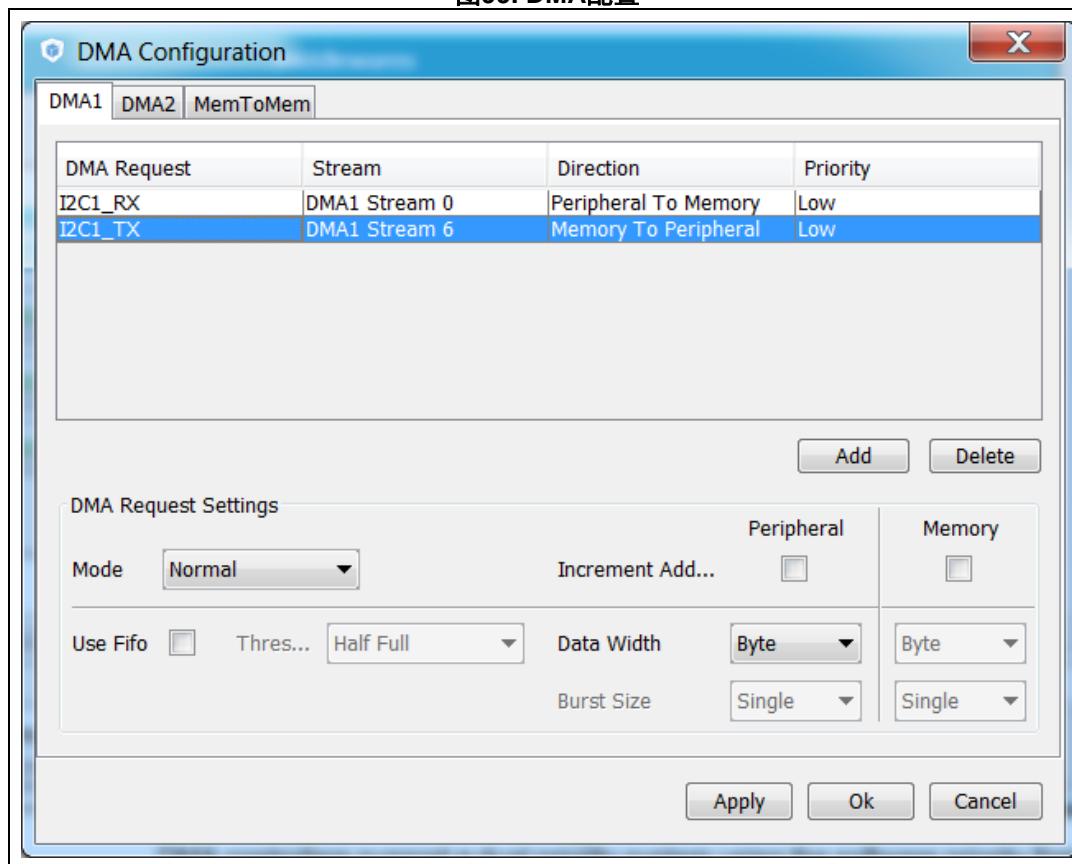


选择DMA请求将自动在所有可用的流、方向和优先级之间分配一个流。在配置DMA通道时，完全描述DMA传输运行时参数（如起始地址等）取决于应用程序代码。

DMA请求（为STM32F4 MCU调用通道）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 99）。流优先级将用于决定为下一个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99. DMA配置

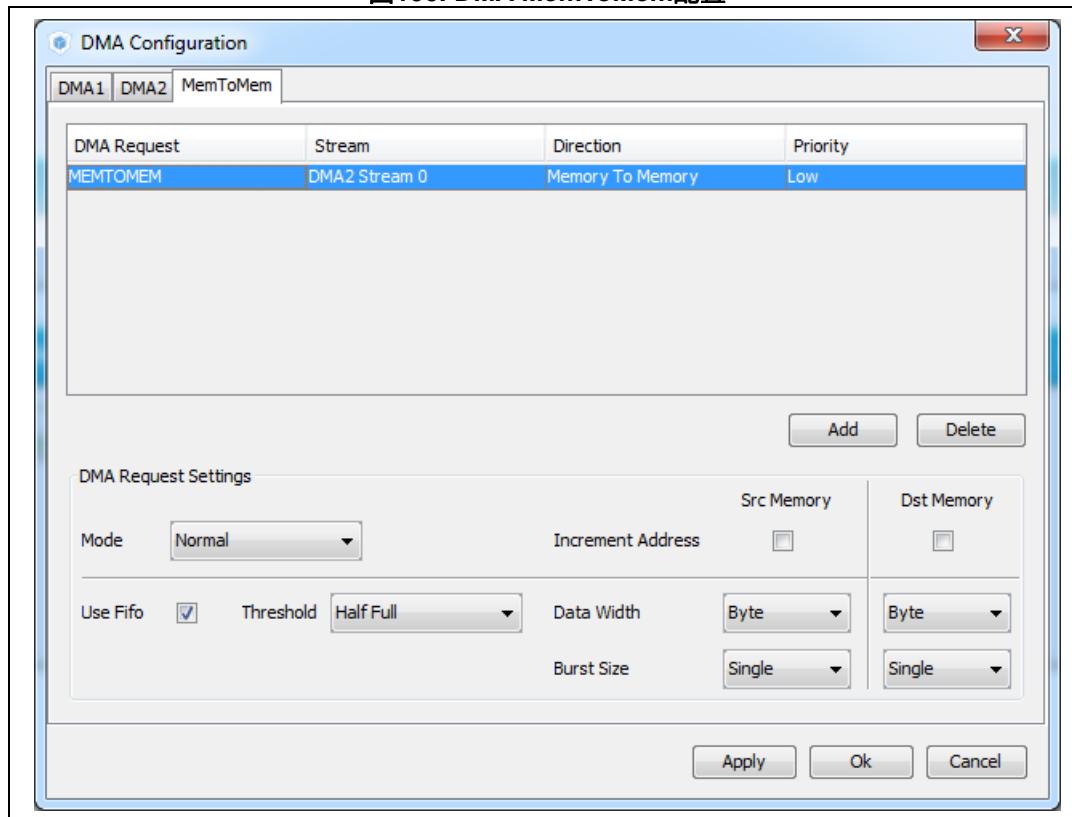


可以通过**DMA配置**窗口完成其他的DMA配置设置：

- 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**：
 - 点击**使用FIFO**复选框。
 - 然后，配置**外设和内存数据宽度**（8、16或32位）。
 - 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100. DMA MemToMem配置



5.13.5 “NVIC配置”窗口

在配置面板中点击**NVIC**打开嵌套向量中断控制器配置窗口（参见图 101）。

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

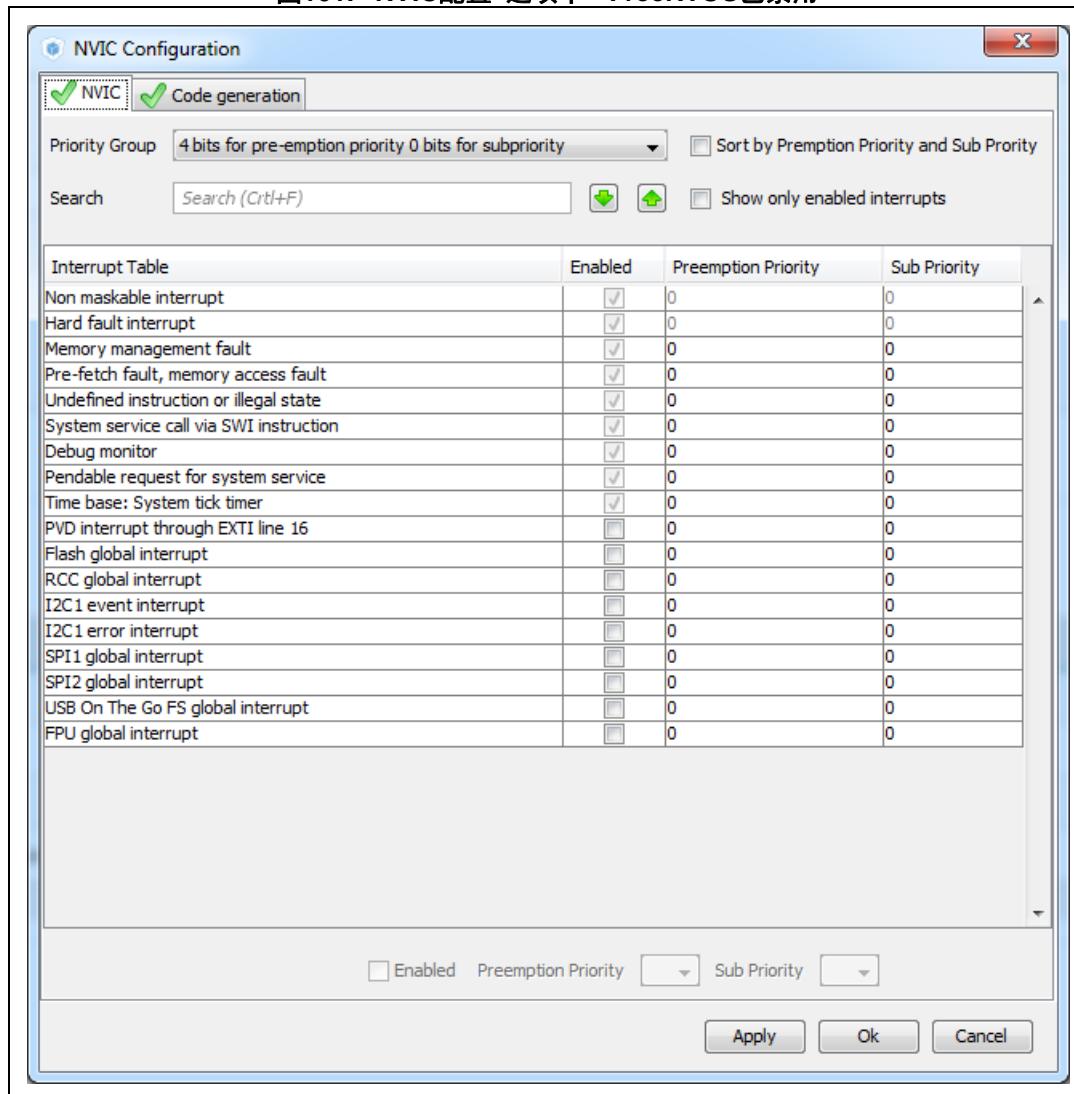
NVIC视图（参见图 101）不显示所有可能的中断，只显示在引脚布局和配置面板中选择的外设可用的中断。显示系统中断，但永远不能被禁用。

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，在从引脚布局面板启用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101. “NVIC配置”选项卡 - FreeRTOS已禁用



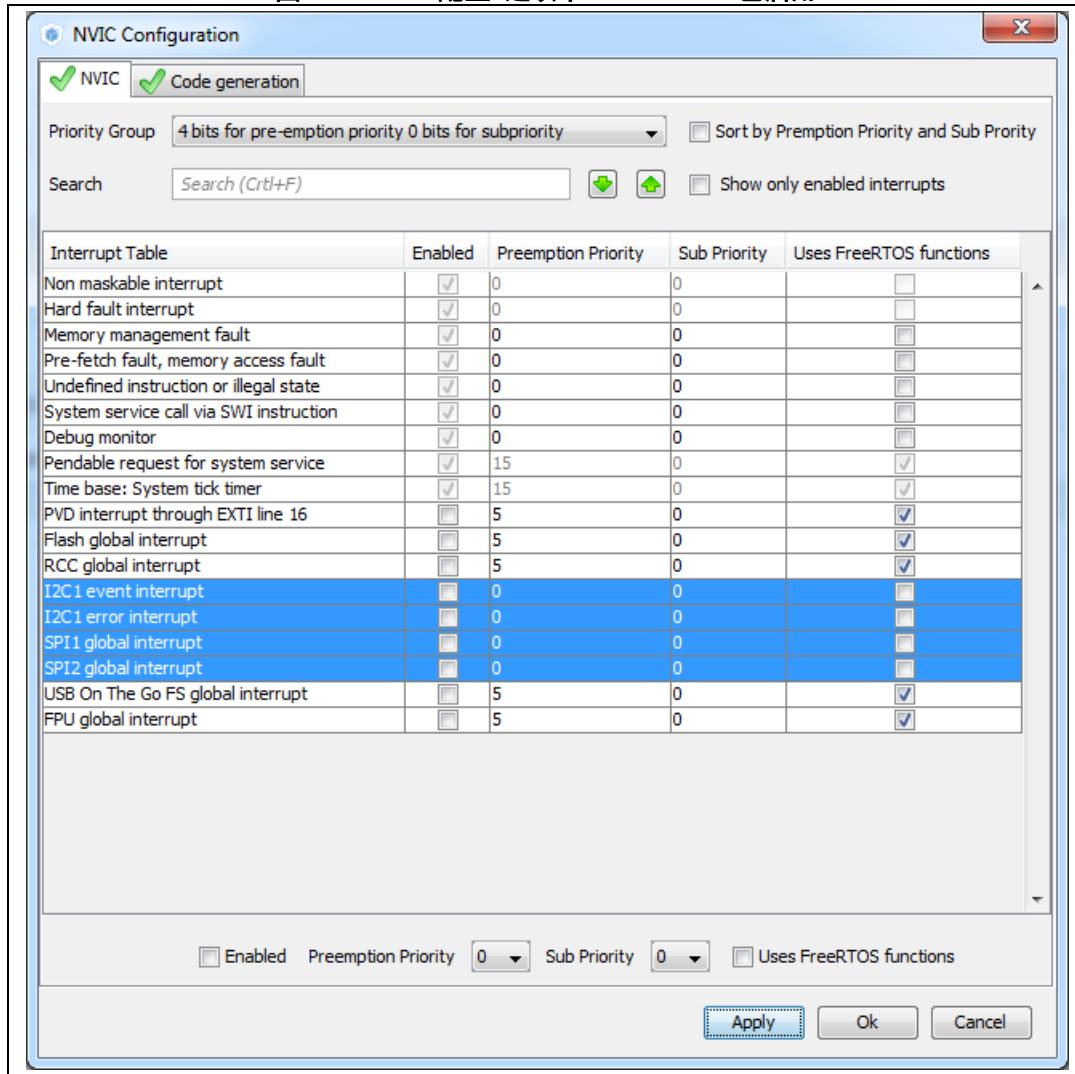
如果FreeRTOS已禁用，将显示额外的列（参见[图 102](#)）。在这种情况下，所有调用中断安全FreeRTOS API的中断服务程序（ISR）的优先级应低于

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。

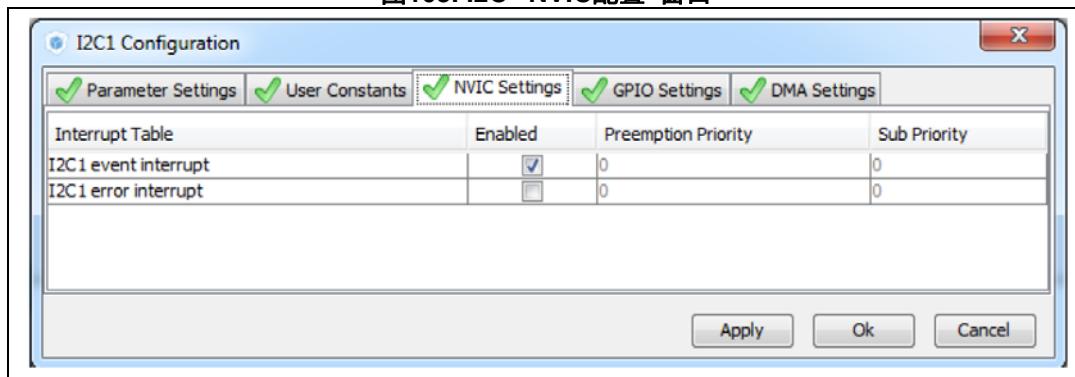
一次可以选中/取消选中多个行（参见在[图 102](#)中以蓝色突出显示的行）。

图102. “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 103）。

图103. 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，在STM32F4 MCU中，NVIC优先级组0对应于0位抢占优先级和4位次优先级。

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。
- 点击应用以保存更改，然后点击确定以关闭窗口。

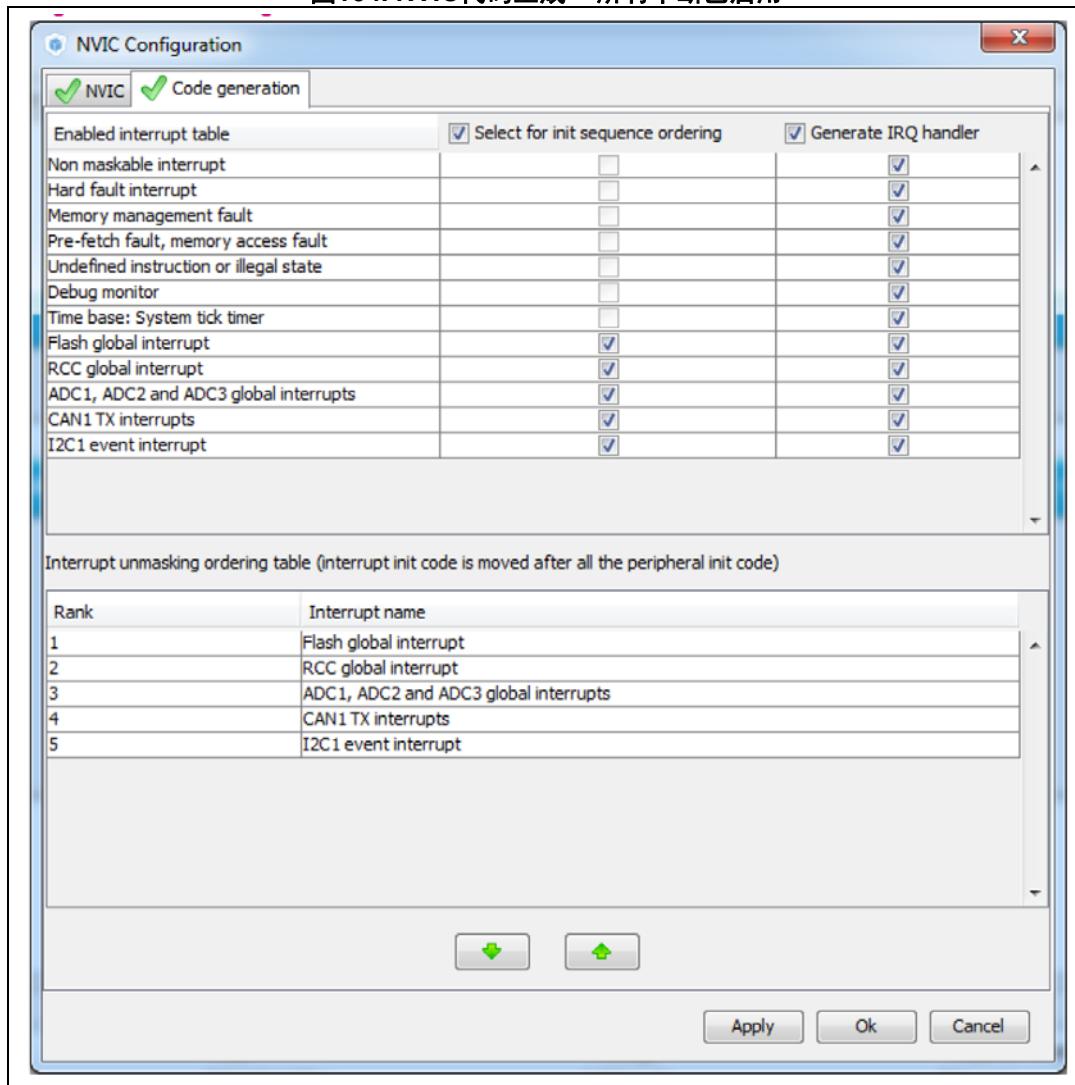
用于中断处理的代码生成选项

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 104）。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104. 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 105](#))。

例如, [图 105](#)中所示的配置生成了以下代码:

```

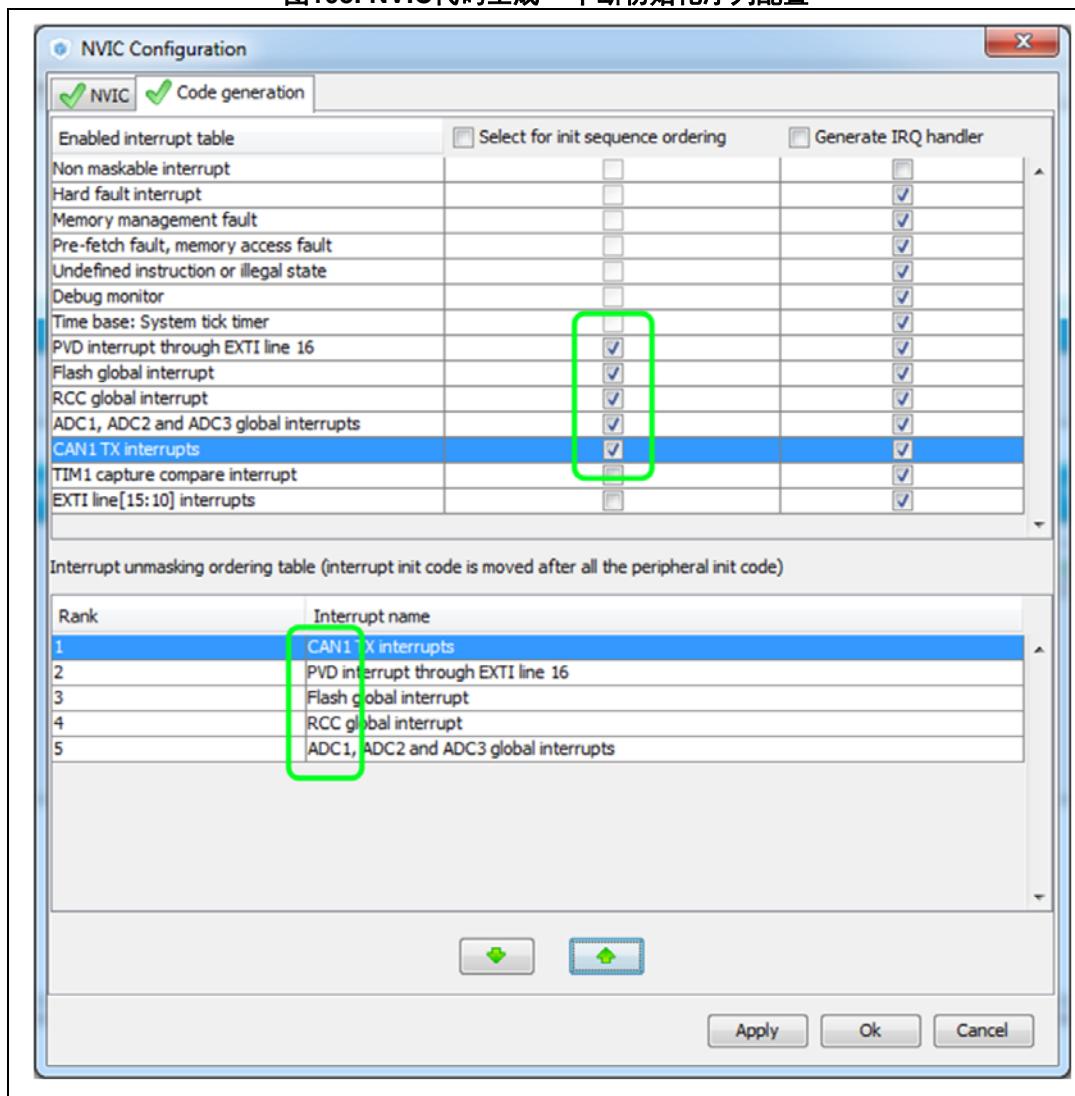
/** NVIC配置
*/
void MX_NVIC_Init(void)
{
/* CAN1_TX_IRQn中断配置*/
HAL_NVIC_SetPriority(CAN1_TX_IRQn, 2, 2);
HAL_NVIC_EnableIRQ(CAN1_TX_IRQn);

/* 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);  
}
```

图105. NVIC代码生成 – 中断初始化序列配置



- 中断处理程序代码生成

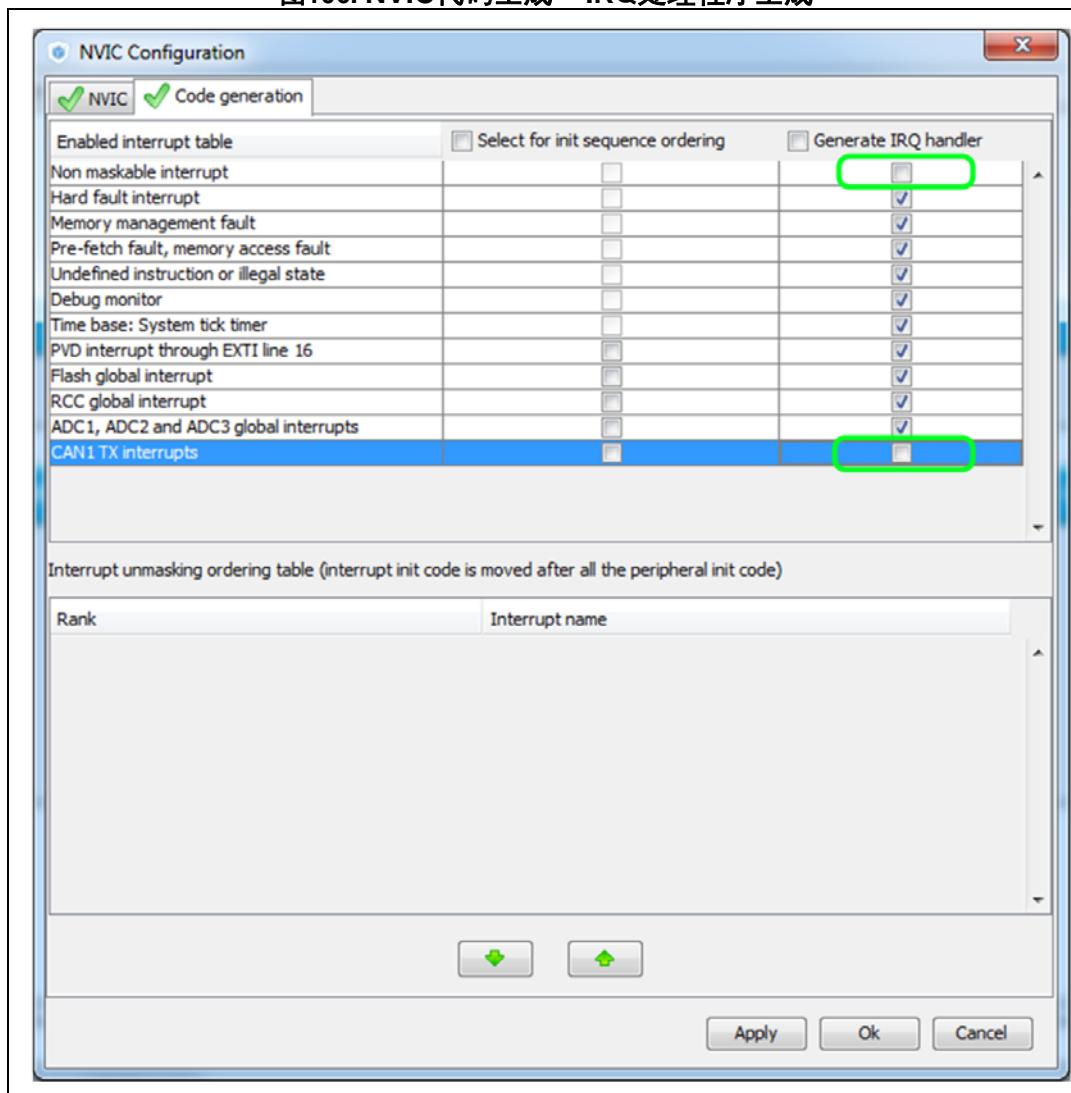
默认情况下，STM32CubeMX在stm32xxx_it.c文件内生成中断处理程序。例如：

```
void NMI_Handler(void)
{
    HAL_RCC_NMI_IRQHandler();
}

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

生成IRQ处理程序列可以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 105中所示从stm32xxx_it.c文件删除前面提到的代码。

图106. NVIC代码生成 – IRQ处理程序生成



5.13.6 FreeRTOS中间件配置视图

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-RTOS API函数生成的代码中定义和创建。其步骤如下：

1. 在配置选项卡中，通过树状视图使能FreeRTOS。
2. 点击配置面板中的FreeRTOS，以打开FreeRTOS配置窗口（参见图 107）。

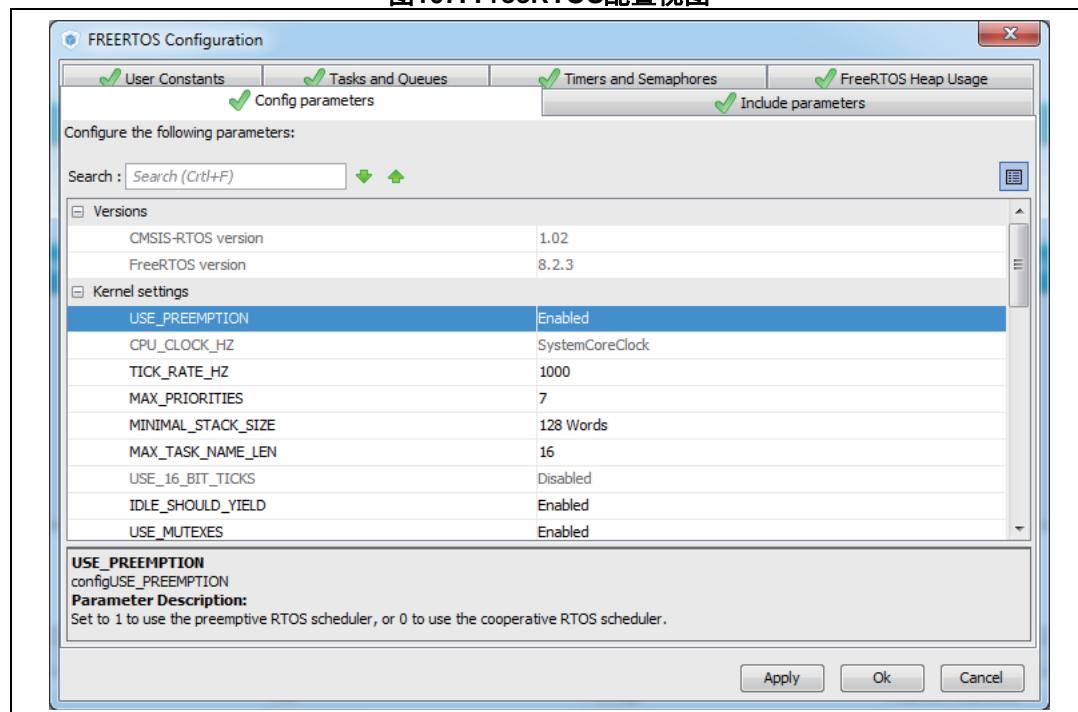
除用户常量选项卡以外，可通过任何其他选项卡配置FreeRTOS本地配置参数和对象，如任务、定时器、队列和信号量。

配置参数值允许配置内核和软件设置。

包含参数选项卡允许只选择应用所需的API函数，以优化代码量。

Config和Include包含参数均为FreeRTOSConfig.h文件的一部分。

图107. FreeRTOS配置视图



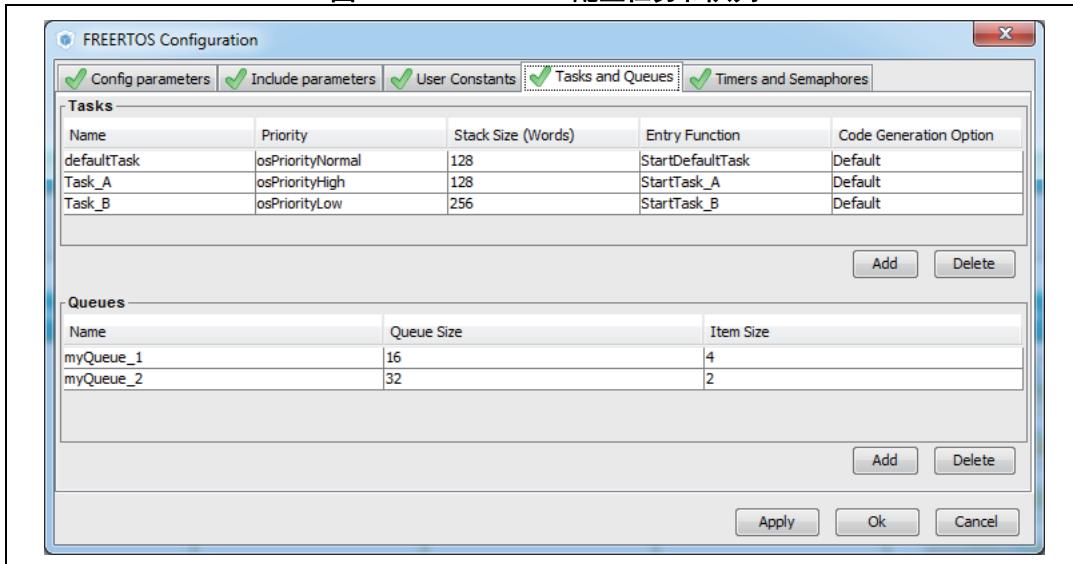
任务和队列选项卡

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 108](#)）。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./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“项目设置”菜单中选择“生成作为每个外设和中间件的.c./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108. 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 109](#)）。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

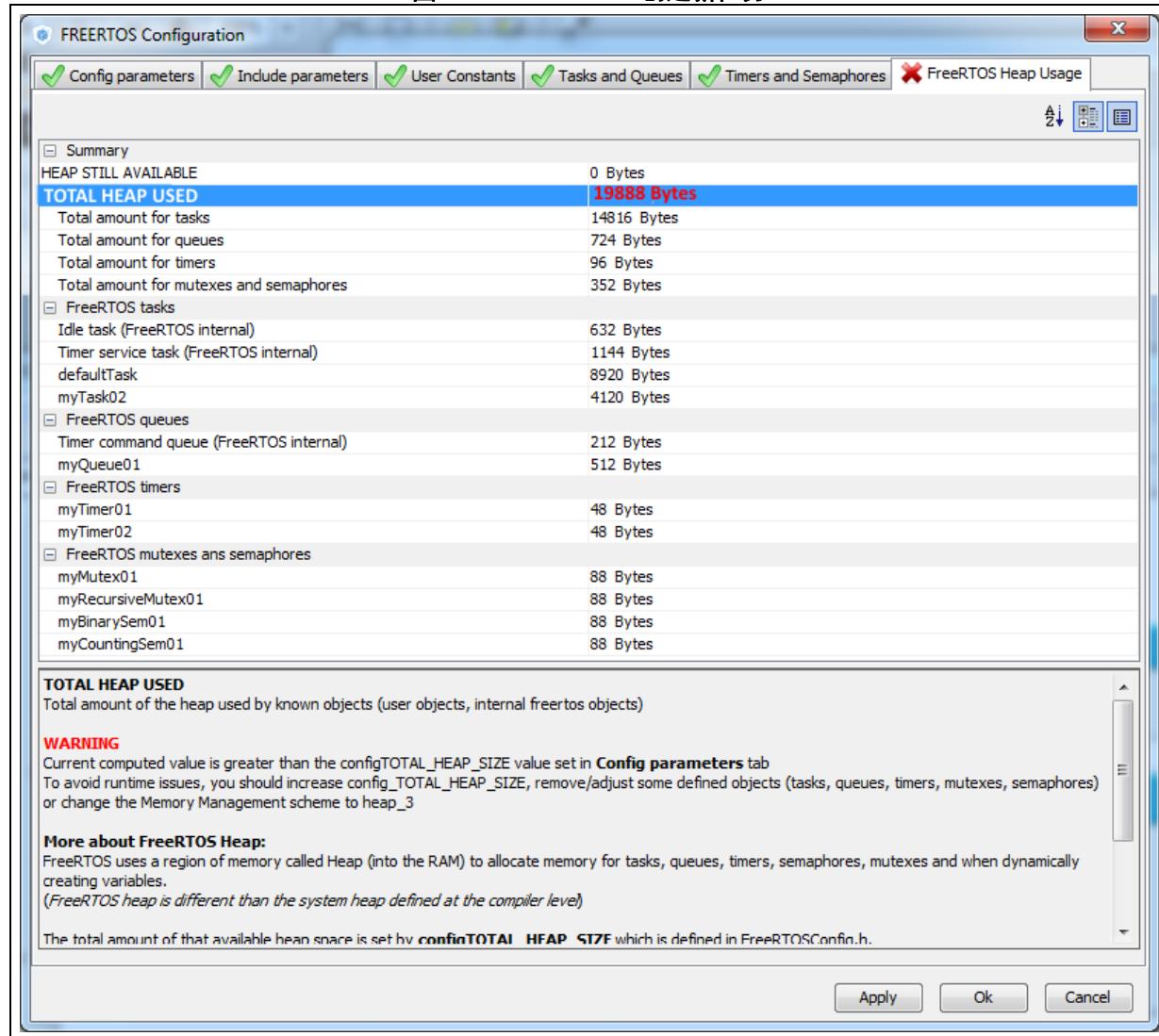
在队列部分下，点击添加按钮，以打开**新建队列**窗口，可在其中配置队列名称、大小和项目大小（参见[图 109](#)）。队列大小对应于队列中一次可容纳的最大项目，而项目大小是存储在队列中的每个数据项的大小。项目大小可以用字节数或数据类型表示：

- 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4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109. FreeRTOS：创建新任务



以下代码段显示了与[图 108](#)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);
```

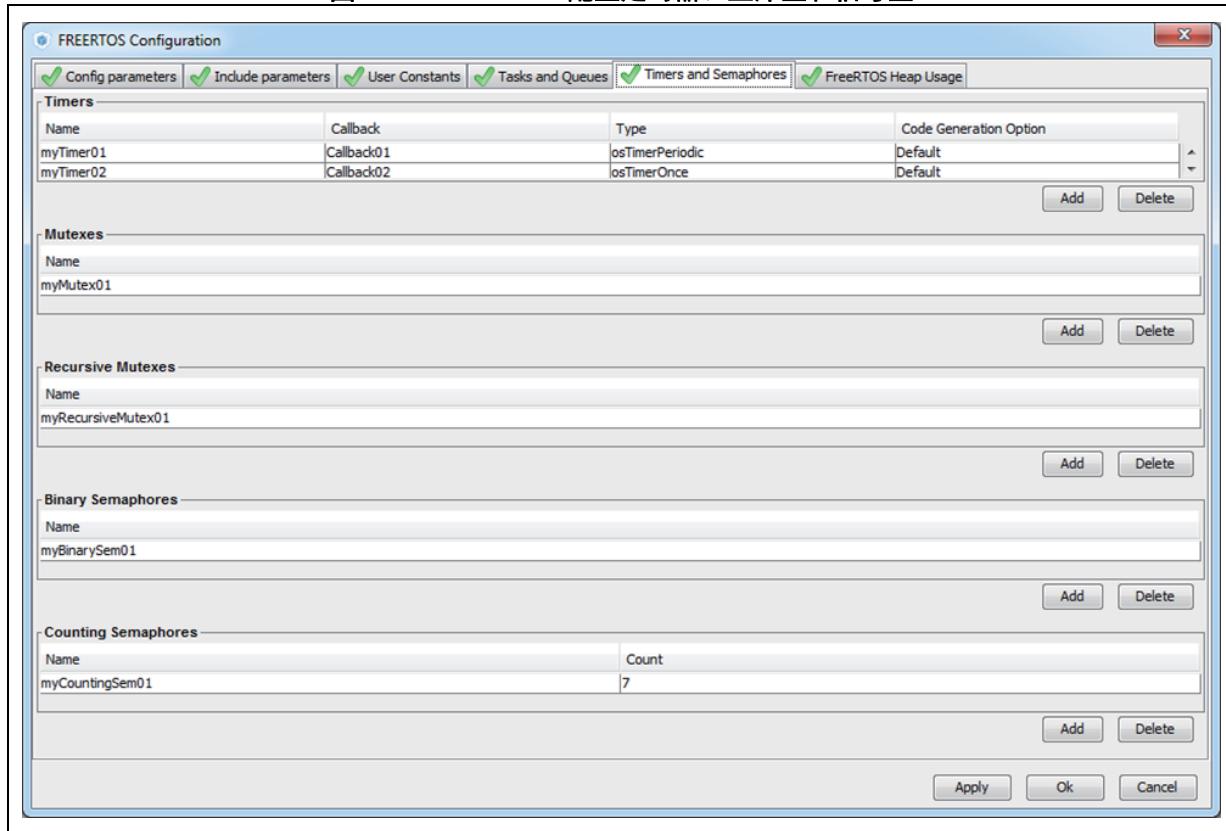
定时器、互斥量和信号量

FreeRTOS定时器、互斥量和信号量可通过FreeRTOS**定时器和互斥量**选项卡配置（参见[图 110](#)）。

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。

图110. FreeRTOS - 配置定时器、互斥量和信号量



- 定时器

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_RECURSIVE_MUTEXES`、`USE_COUNTING_SEMAPHORES` 定义）。

以下代码段显示了与图 110: FreeRTOS - 配置定时器、互斥量和信号量对应的生成代码。

```
/* 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);

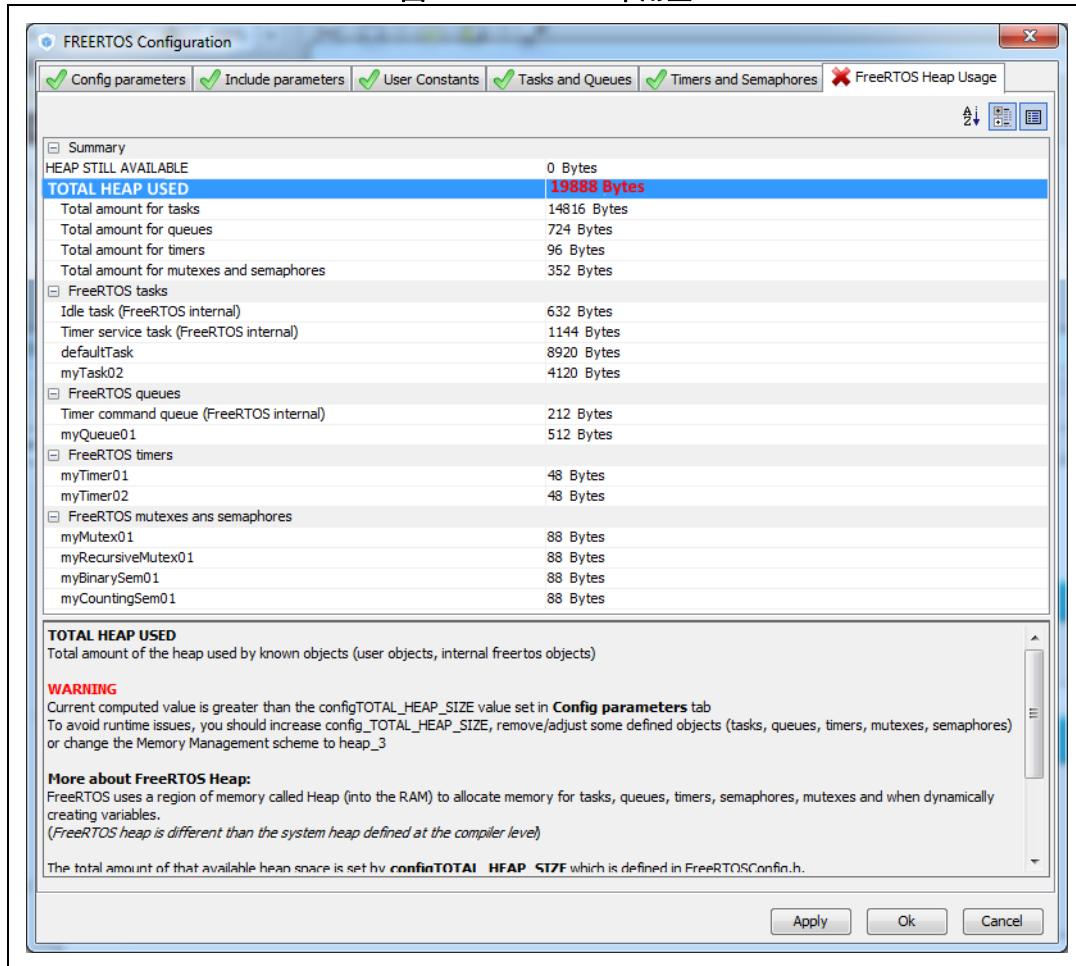
/* 创建互斥量 */
/* 定义和创建myMutex01 */
osMutexDef(myMutex01);
myMutex01Handle = osMutexCreate(osMutex(myMutex01));

/* 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。当使用的堆总量超过TOTAL_HEAP_SIZE最大阈值时，将以红色显示并在选项卡上显示红叉（参见图 111）。

图111. FreeRTOS堆用量



5.13.7 图形框架和仿真器

在选择支持图形处理的微控制器（参见[基于图形条件选择MCU](#)）后，用户可以选择STemWin图形框架。

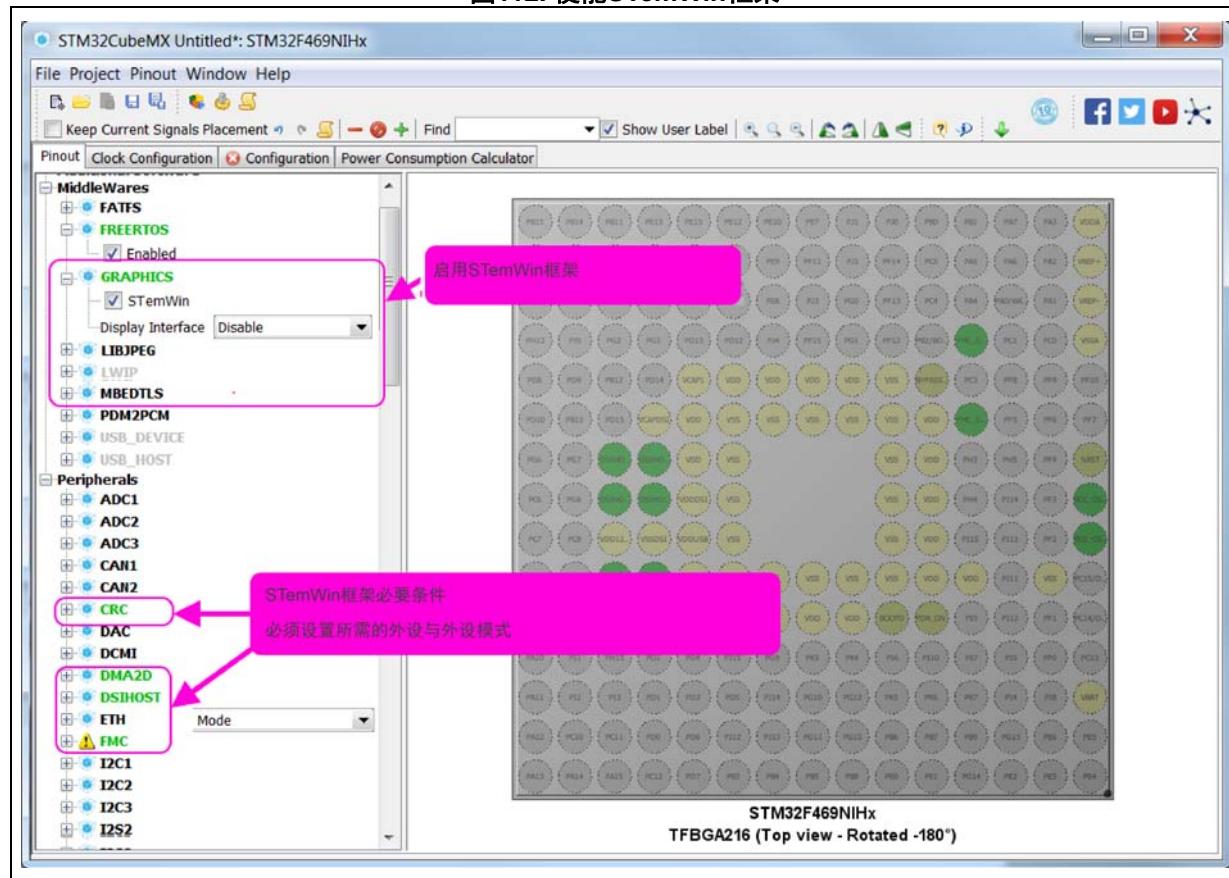
有关详细信息，请参见STM32CubeF4和STM32CubeF7嵌入式软件包中提供的相关文档与许可条款。

通过使用图形仿真器，用户可以将图形当前的配置性能与仿真设置进行比较。随后可以导入这些结果，以便用于项目。

引脚布局视图：使能图形框架和必要条件

在引脚布局视图中，必须正确配置必要的外设和中间件，以使图形框架完全可用。将鼠标悬停在框架名称之上，以显示详细说明必要条件的工具提示（参见[图 112](#)）。

图112. 使能STemWin框架



配置视图：图形配置窗口

在引脚布局视图中选择图形框架和相关显示接口后，图形配置按钮在配置视图中以绿色显示。

点击“图形”按钮可打开图形配置窗口，可在其中配置图形公共参数、框架特定参数和启动框架特定工具。有关详细信息，请参见每个框架的专用教程。

图113. 图形配置视图

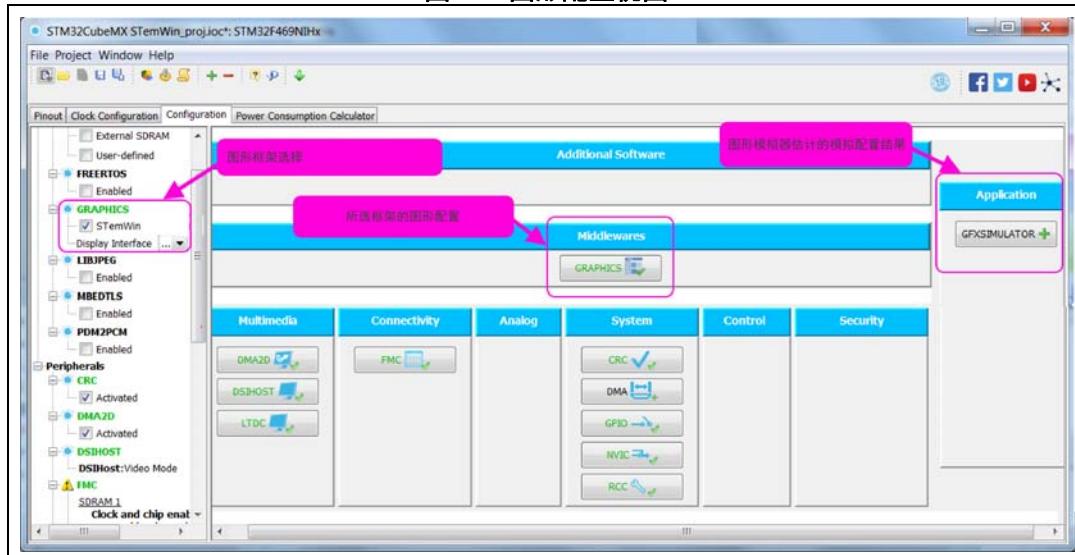
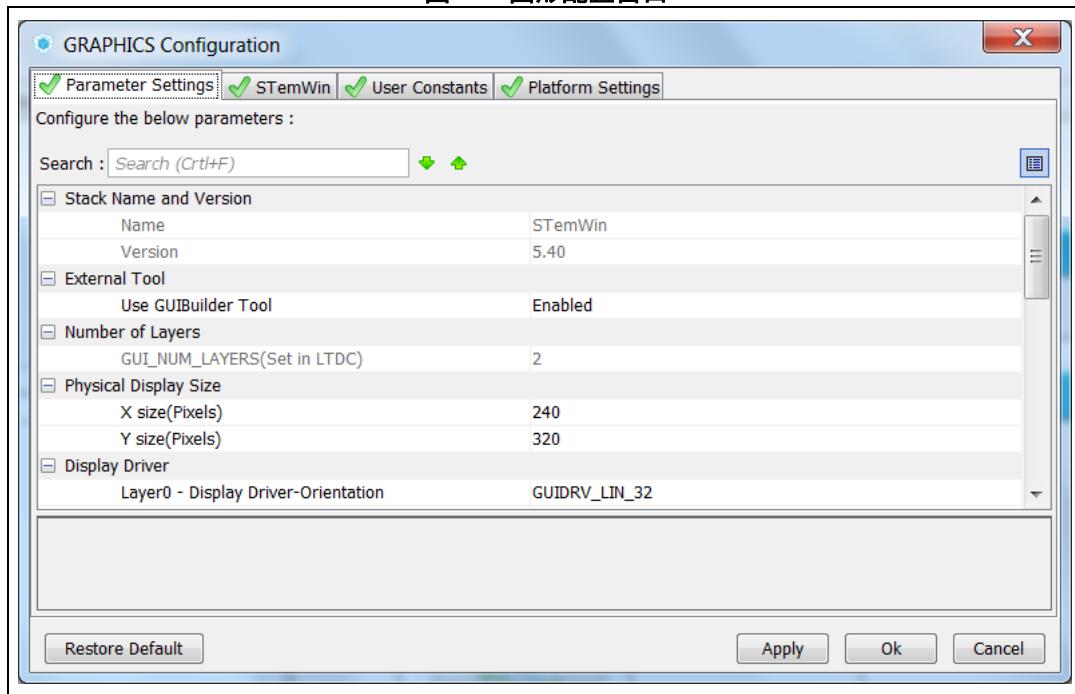


图114. 图形配置窗口



配置视图：图形仿真器

点击GFXSIMULATOR按钮可打开图形仿真器用户界面。

在左侧面板中，用户可编辑参数，以更改仿真器默认配置。

在右侧面板中，用户可以可视化当前项目的图形配置。

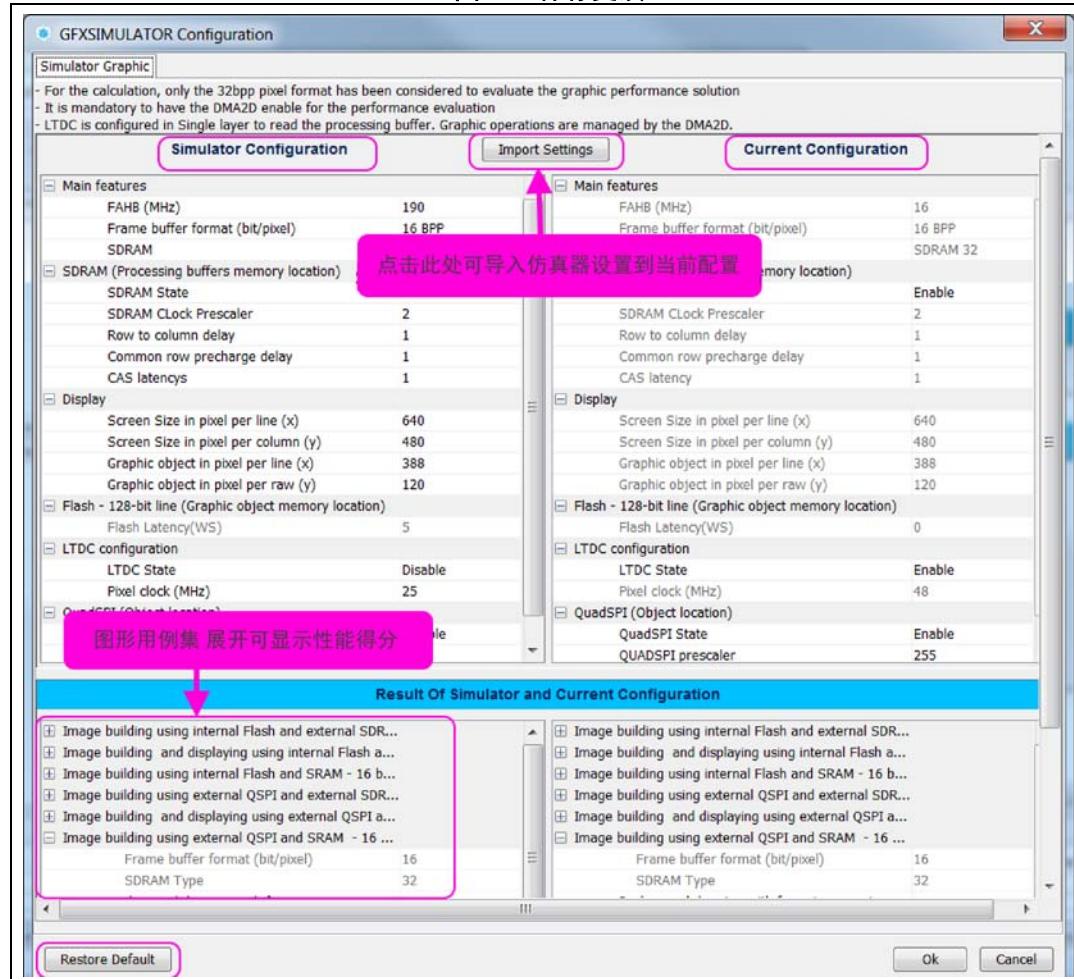
在底部面板中，用户可以比较一组图形用例的配置性能结果。

如果对仿真器的结果满意，用户可以点击“导入设置”按钮，以导入当前配置中的仿真器设置。

点击“恢复默认”可恢复仿真器配置的STM32CubeMX默认设置。

点击“确定”可保存在仿真器配置面板中所完成的更改，并在下次重新打开GFXSIMULATOR窗口时使用保存的设置。

图115. 保存更改



5.14 时钟树配置视图

STM32CubeMX时钟配置窗口（参见图 116）提供了时钟路径、时钟源、分频器和倍频器的原理概览。通过下拉菜单和按钮可修改实际时钟树配置，以满足用户应用要求。

实际时钟速度显示并处于活动状态。使用的时钟信号以蓝色突出显示。

超出范围的配置值以红色突出显示，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（参见图 117）。

支持反向路径：只需在蓝色字段中输入所需的时钟速度，STM32CubeMX将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。

5.14.1 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外设下可用的引脚布局视图将其使能。

外设时钟配置选项

一些对应于时钟外设的其他路径灰显。要激活这些路径，必须在**引脚布局**视图（如USB）中正确配置外设。该视图允许：

- **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将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
- ...

- **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。

- **使用工具栏撤销/恢复用户配置步骤**

撤销/恢复按钮 ()

- **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。以红色突出显示错误，**时钟配置**视图标有红叉（参见图 117）。

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：

- 将HSE频率设为最大值（可选）。
- 依次将HCLK频率与外设频率设为最大值或最小值（可选）。
- 更改多路复用器输入（可选）。
- 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，将清除时钟树的红色突出显示。否则会显示错误消息。

注：

要在时钟树中可用，应在**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。

此工具将自动执行以下操作：

- 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）调整时钟树视图，反之亦然：

- 如果在RCC引脚布局视图中使能外部和输出时钟，则可在时钟树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，则将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。它们在时钟树视图中显示。

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main函数中切换到用户定义的时钟设置。

[图 116](#)提供了STM32F429x MCU的时钟树配置视图示例，[表 14](#)介绍了可用于配置每个时钟的小工具。

图116. STM32F429xx时钟树配置视图

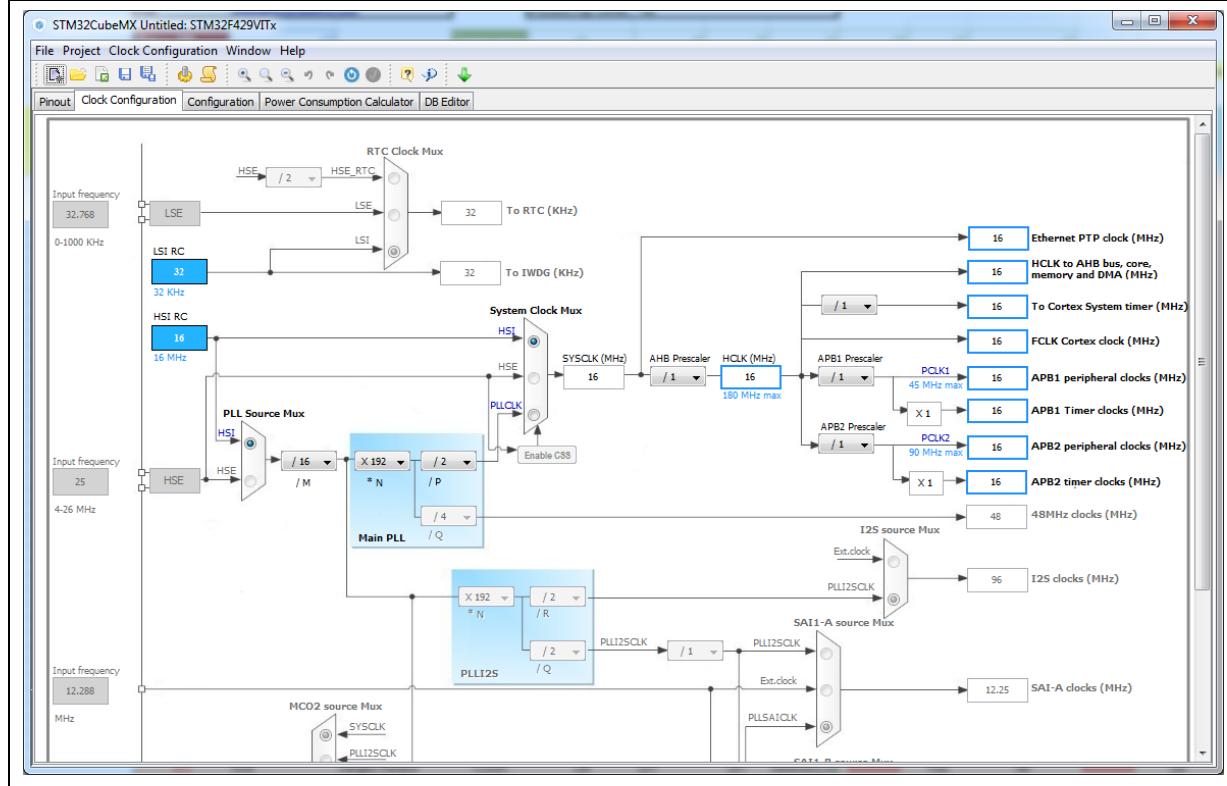


图117. 有错误的时钟树配置视图

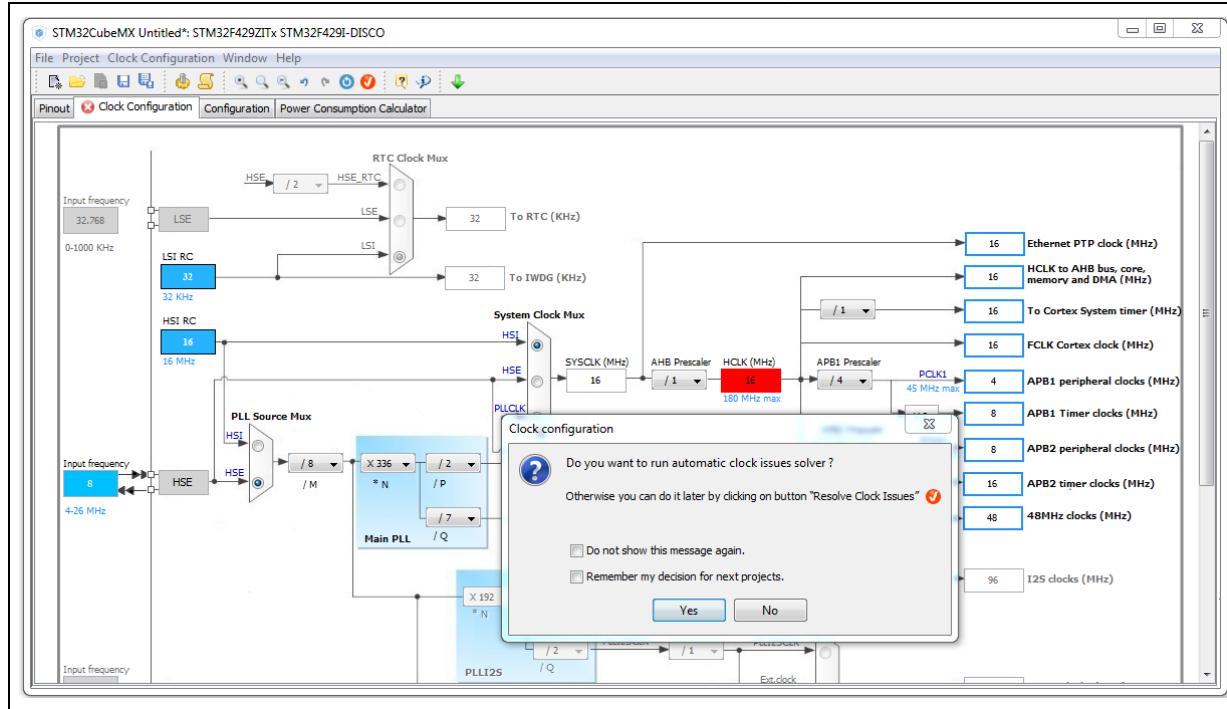


表14. 时钟树视图小工具

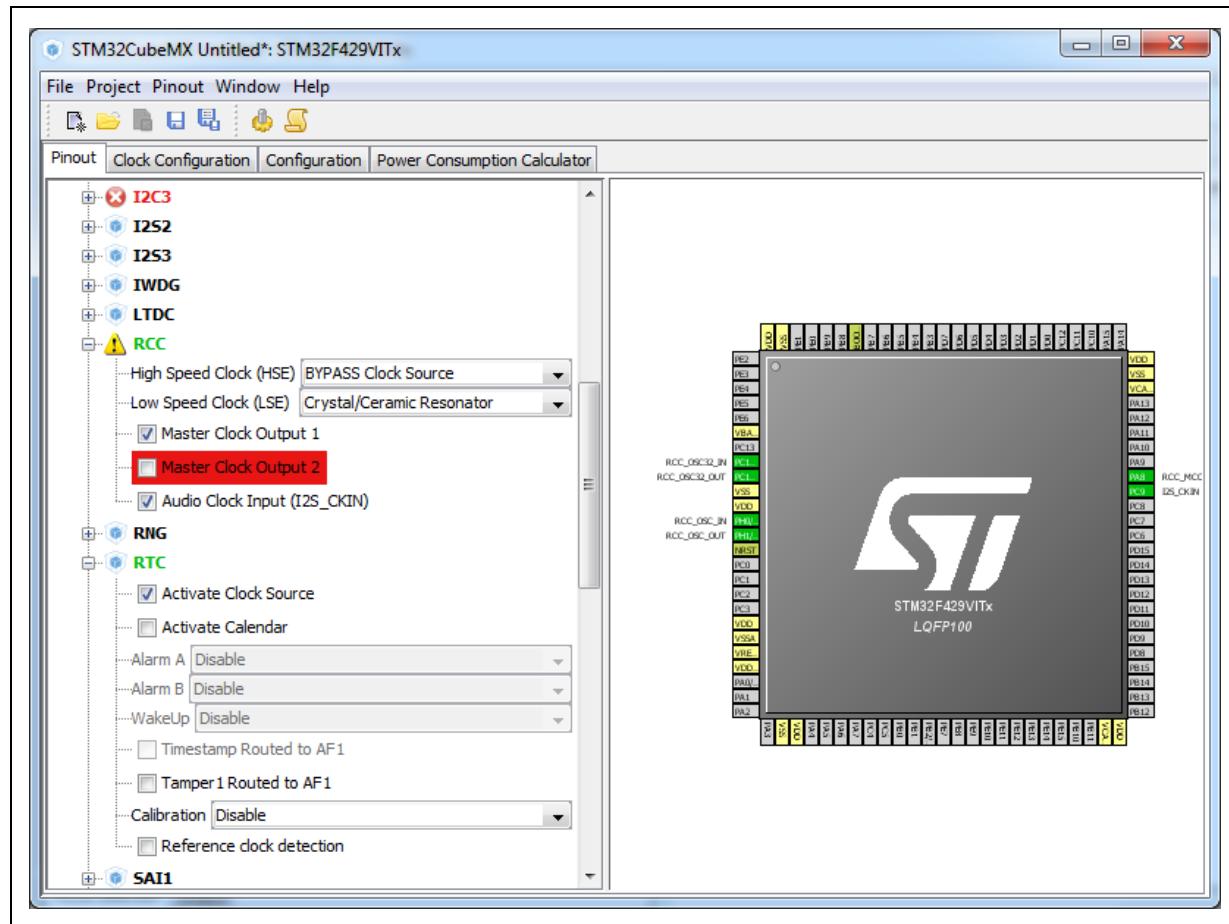
格式	外设实例的配置状态
	有效时钟源
	模糊或灰显的不可用设置（时钟源、分频器等）
	预分频器、分频器、倍频器选择的灰色下拉列表。
	倍频器选择
	用户定义的频率值
	自动得出的频率值
	用户可修改的频率字段
	右键单击蓝色边框矩形，以锁定/解锁频率字段。将其锁定以便在时钟树配置更新期间保留频率值。

5.14.2 建议

时钟树视图不是时钟配置的唯一入口。

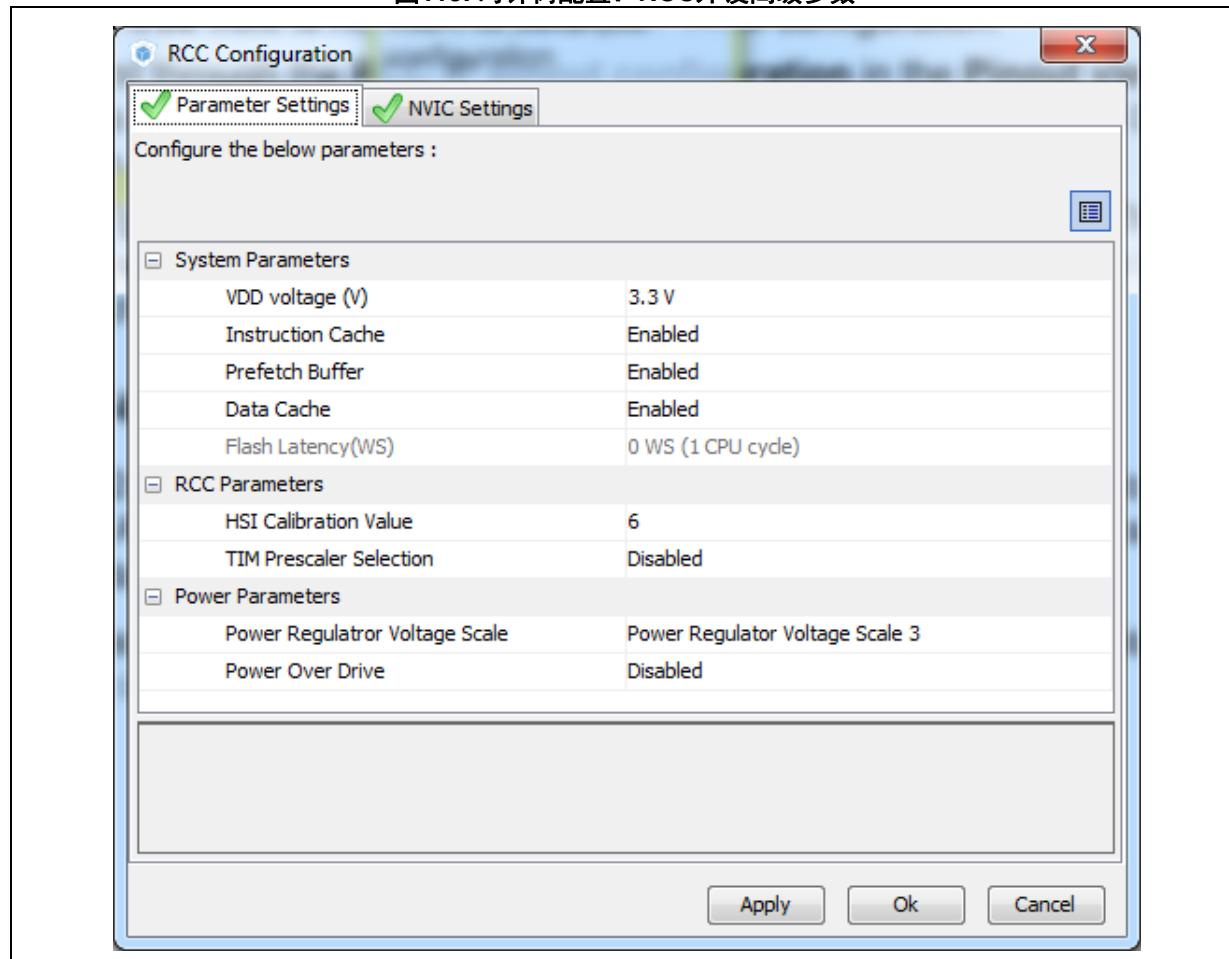
- 首先通过引脚布局视图中的RCC引脚布局配置根据需要使能可用的时钟：外部时钟、主输出时钟和音频I2S输入时钟（参见图 118）。

图118. 时钟树配置：通过引脚布局视图使能RTC、RCC时钟源和输出



2. 然后前往配置视图中的RCC配置。这里为高级配置定义的设置将反映在时钟树视图中。时钟树视图中定义的设置可能会改变RCC配置中的设置（参见图 119）。

图119. 时钟树配置：RCC外设高级参数



5.14.3 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 15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15. 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	配置 (RCC)	根据HCLK频率和功率超载自动得出（参见 表 16 ）。
功率超载	配置 (RCC)	该值取决于HCLK和V _{DD} 值（参见 表 16 ）。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	时钟配置	用蓝色显示，以指示可能的最大值

[表 16](#)给出了功率超载模式与HCLK频率之间的关系。

表16. 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）

5.14.4 时钟树词汇表

表17. 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

表17. 词汇表（续）

缩略语	定义
FCLK	Cortex自由运行时钟
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

5.15 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。为帮助最大限度地降低功耗，STM32CubeMX提供了**功耗计算器**选项卡（参见[图 120](#)），该选项卡根据微控制器、电池模型以及用户定义的功耗系列提供了以下结果：

- 平均电流消耗
功耗值可从数据表中获取，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数据表获得，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表中指定。

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。

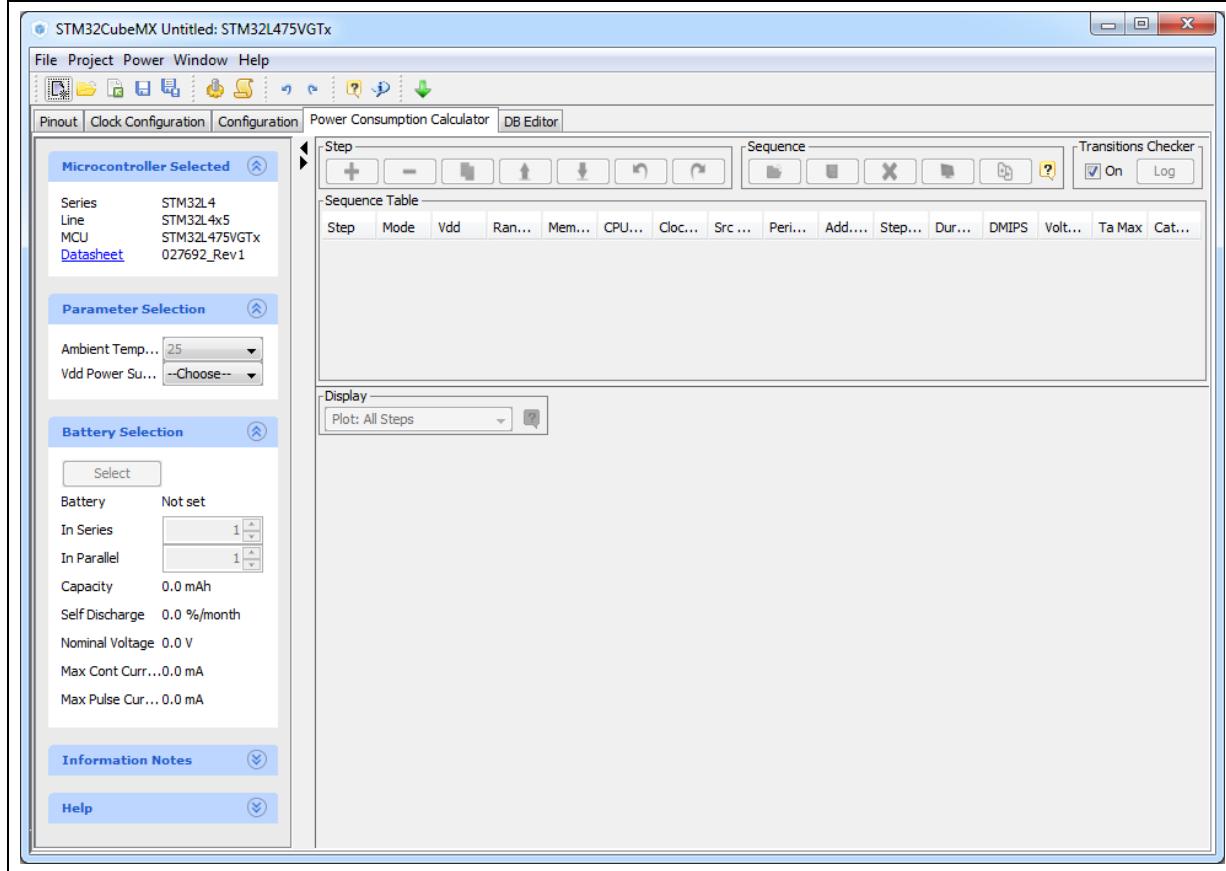
在每个步骤中，用户可以选择将VBUS作为可能的电源，而不是电池。这将影响电池寿命估算。如果可以在不同的电压水平下进行功耗测量，STM32CubeMX还将推荐电压值选择（参见[图 124](#)）。

另一个选项（跳变检查器）可用于STM32L0、STM32L1、STM32L4和STM32L4+系列。当使能时，跳变检查器会检测当前配置的序列中的无效转换。这可以确保在添加新步骤时仅将可能的转换推荐给用户。

5.15.1 建立功耗系列

默认开始视图如图 120 中所示。

图120. 功耗计算器默认视图



选择V_{DD}值

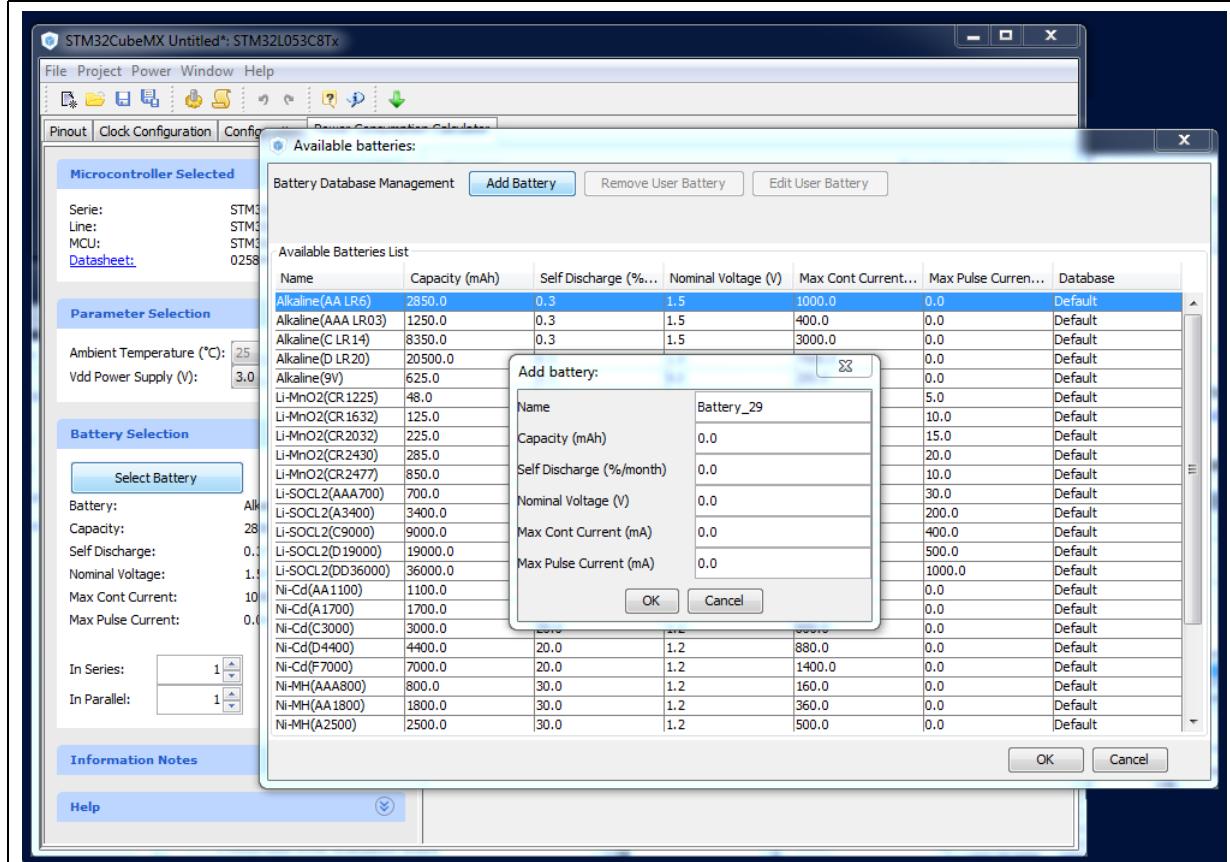
在该视图中，当多个选项可用时，用户必须选择V_{DD}值。

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 121）。

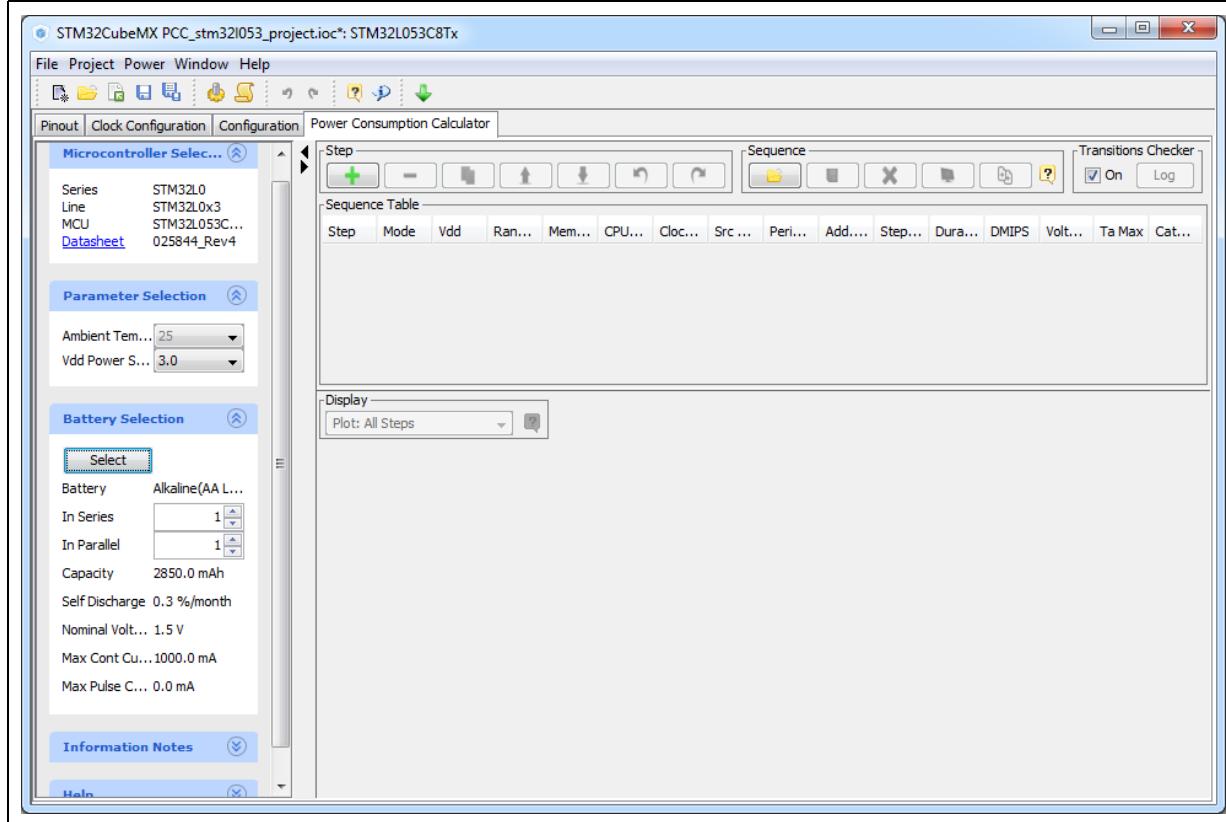
图121. 电池选择



功耗系列默认视图

用户现在可以继续建立功耗系列（参见图 122）。

图122. 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 123）。

通过点击功耗计算器视图中的**撤销**或**恢复**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123. 步骤管理功能

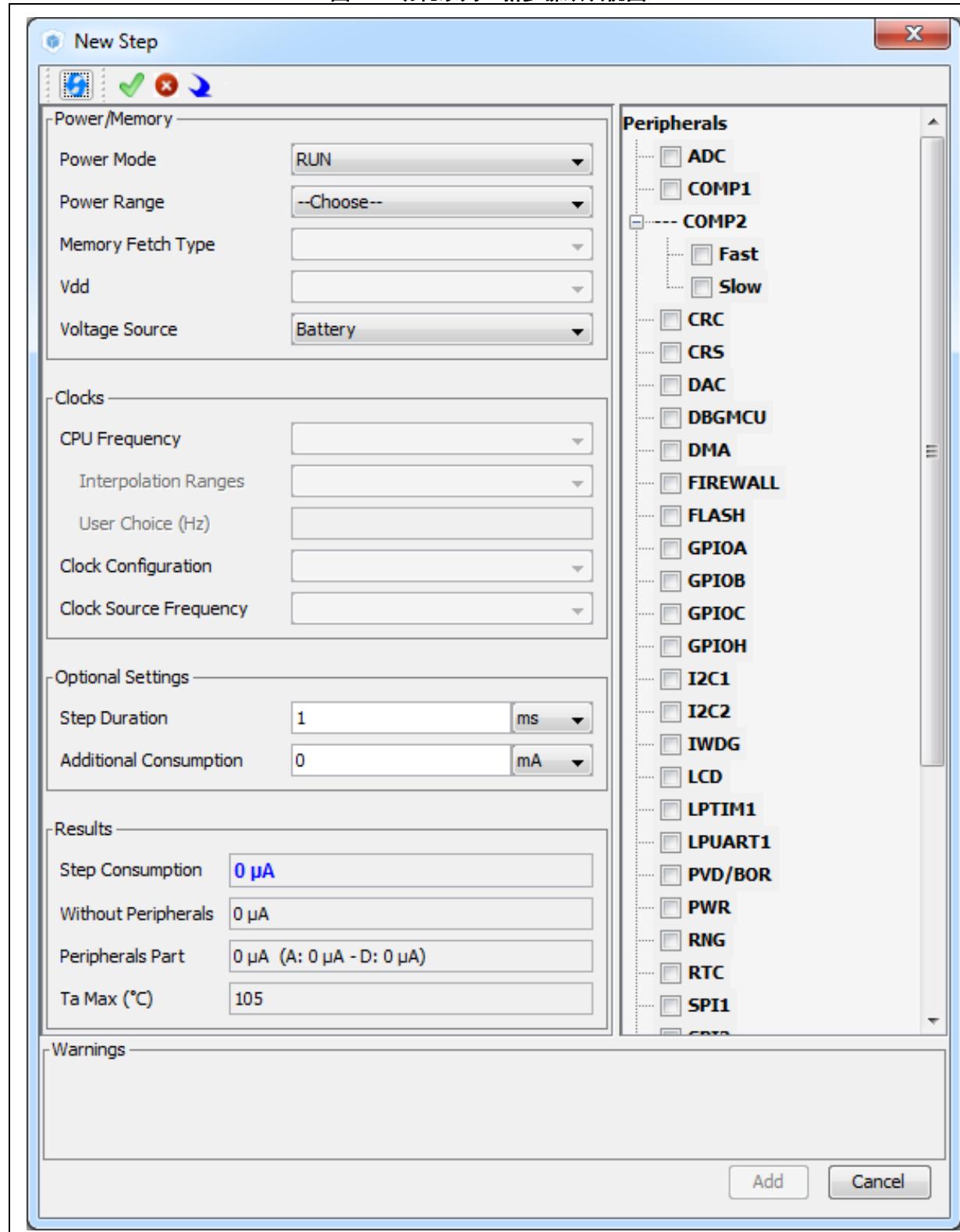


添加一个步骤

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。（见图 124）。

图124. 功耗系列：新步骤默认视图

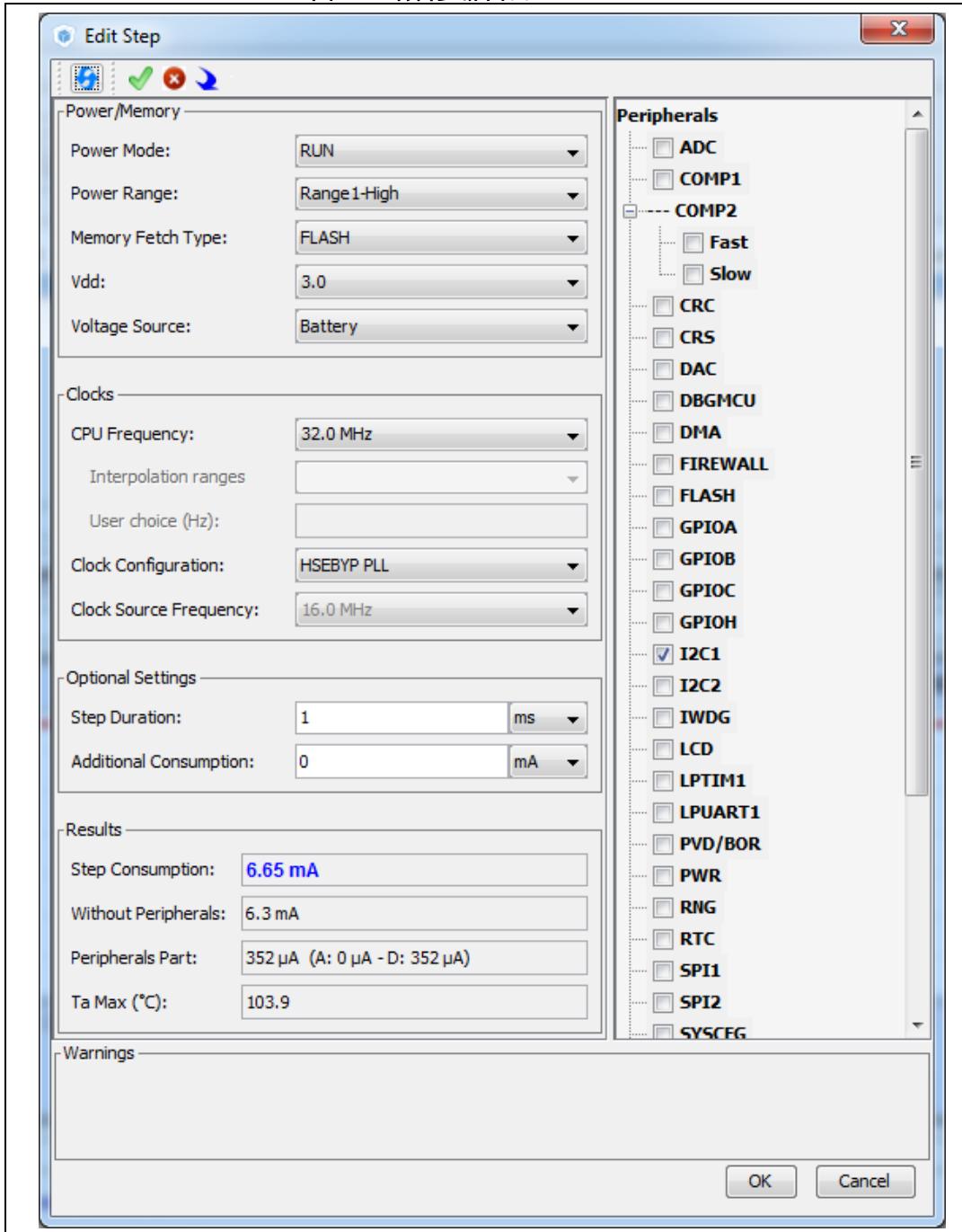


在配置完步骤后，窗口中将显示当前消耗和 $T_{A\text{MAX}}$ 值。

编辑步骤

要编辑步骤，在序列表中双击步骤。编辑步骤窗口将打开（参见图 125）。

图125. 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跳变都可行。功耗计算器提供跳变检查器以检测无效的跳变，或将序列配置限制为仅限有效跳变。

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。

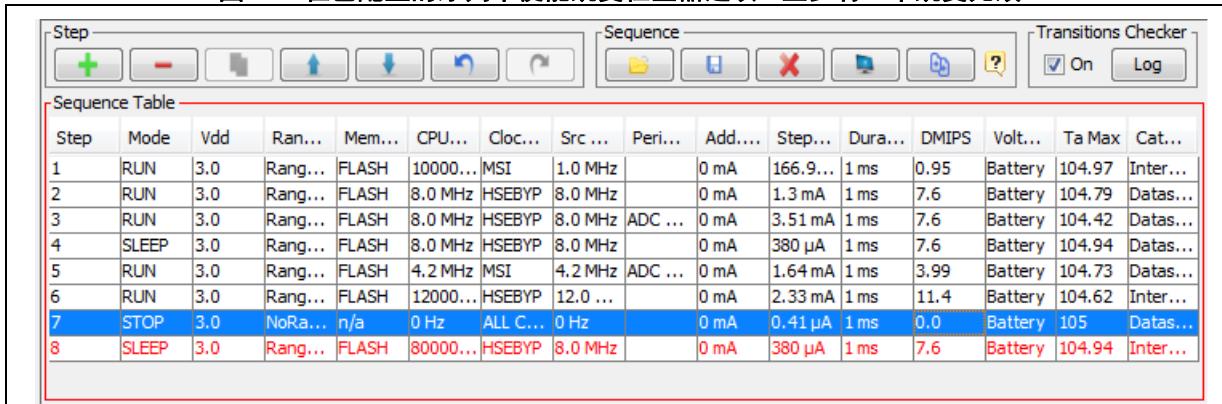
在已配置序列上使能跳变检查器选项后，如果所有跳变有效，将以绿色突出显示（绿框）（参见图 126），如果至少有一个跳变无效，则以红色突出显示（红框加上以红色突出显示的无效步骤描述）（参见图 127）。

在这种情况下，用户可以点击显示日志按钮，以了解如何解决转换问题（参见图 128）。

图126. 在已配置的序列中使能跳变检查器选项 - 所有跳变都有效

Step	Mode	Vdd	Ran...	Mem...	CPU...	Cloc...	Src ...	Peri...	Add...	Step...	Dur...	DMIPS	Volt...	Ta ...	Cat...
1	RUN	3.0	Rang...	FLASH	1000...	MSI	1.0 MHz		0 mA	166.9...	1 ms	0.95	Battery	104.97	Inter...
2	RUN	3.0	Rang...	FLASH	8.0 MHz	HSEBYP	8.0 MHz		0 mA	1.3 mA	1 ms	7.6	Battery	104.79	Data...
3	RUN	3.0	Rang...	FLASH	8.0 MHz	HSEBYP	8.0 MHz	ADC ...	0 mA	3.51 mA	1 ms	7.6	Battery	104.42	Data...
4	SLEEP	3.0	Rang...	FLASH	8.0 MHz	HSEBYP	8.0 MHz		0 mA	380 µA	1 ms	7.6	Battery	104.94	Data...
5	RUN	3.0	Rang...	FLASH	4.2 MHz	MSI	4.2 MHz	ADC ...	0 mA	1.64 mA	1 ms	3.99	Battery	104.73	Data...
6	RUN	3.0	Rang...	FLASH	1200...	HSEBYP	12.0 ...		0 mA	2.33 mA	1 ms	11.4	Battery	104.62	Inter...
7	STOP	3.0	NoRa...	n/a	0 Hz	ALL C...	0 Hz		0 mA	0.41 µA	1 ms	0.0	Battery	105	Data...

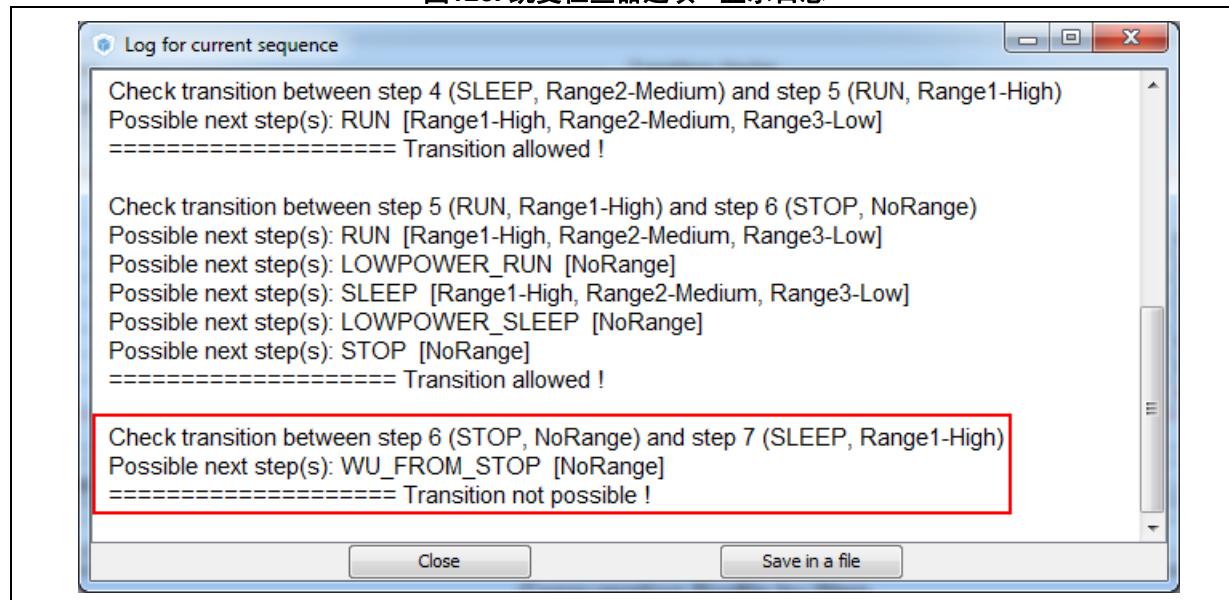
图127. 在已配置的序列中使能跳变检查器选项 - 至少有一个跳变无效



The screenshot shows the STM32CubeMX Sequence Editor interface. At the top, there are buttons for Step management (Add, Remove, Insert, Move Up, Move Down, Undo, Redo) and Sequence management (Save, Load, Delete, Check). To the right, there is a 'Transitions Checker' section with a checkbox labeled 'On' and a 'Log' button. Below these are tabs for 'Sequence Table' and 'Sequence Graph'. The 'Sequence Table' tab is active, displaying a grid of 8 rows (steps) and 16 columns (parameters). Step 7 is highlighted in blue. The columns include: Step, Mode, Vdd, Range, Memory, CPU, Clock, Source, Period, Address, Step, Duration, DMIPS, Voltage, Ta Max, and Category.

Step	Mode	Vdd	Ran...	Mem...	CPU...	Cloc...	Src ...	Peri...	Add...	Step...	Dura...	DMIPS	Volt...	Ta Max	Cat...
1	RUN	3.0	Rang...	FLASH	10000...	MSI	1.0 MHz		0 mA	166.9...	1 ms	0.95	Battery	104.97	Inter...
2	RUN	3.0	Rang...	FLASH	8.0 MHz	HSEBYP	8.0 MHz		0 mA	1.3 mA	1 ms	7.6	Battery	104.79	Data...
3	RUN	3.0	Rang...	FLASH	8.0 MHz	HSEBYP	8.0 MHz	ADC ...	0 mA	3.51 mA	1 ms	7.6	Battery	104.42	Data...
4	SLEEP	3.0	Rang...	FLASH	8.0 MHz	HSEBYP	8.0 MHz		0 mA	380 µA	1 ms	7.6	Battery	104.94	Data...
5	RUN	3.0	Rang...	FLASH	4.2 MHz	MSI	4.2 MHz	ADC ...	0 mA	1.64 mA	1 ms	3.99	Battery	104.73	Data...
6	RUN	3.0	Rang...	FLASH	12000...	HSEBYP	12.0 ...		0 mA	2.33 mA	1 ms	11.4	Battery	104.62	Inter...
7	STOP	3.0	NoRa...	n/a	0 Hz	ALL C...	0 Hz		0 mA	0.41 µA	1 ms	0.0	Battery	105	Data...
8	SLEEP	3.0	Rang...	FLASH	80000...	HSEBYP	8.0 MHz		0 mA	380 µA	1 ms	7.6	Battery	104.94	Inter...

图128. 跳变检查器选项 - 显示日志



5.15.2 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 5.15.4节](#)词汇表和[附录D：STM32微控制器功耗参数](#)，或参阅数据表的电气特性部分。

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。

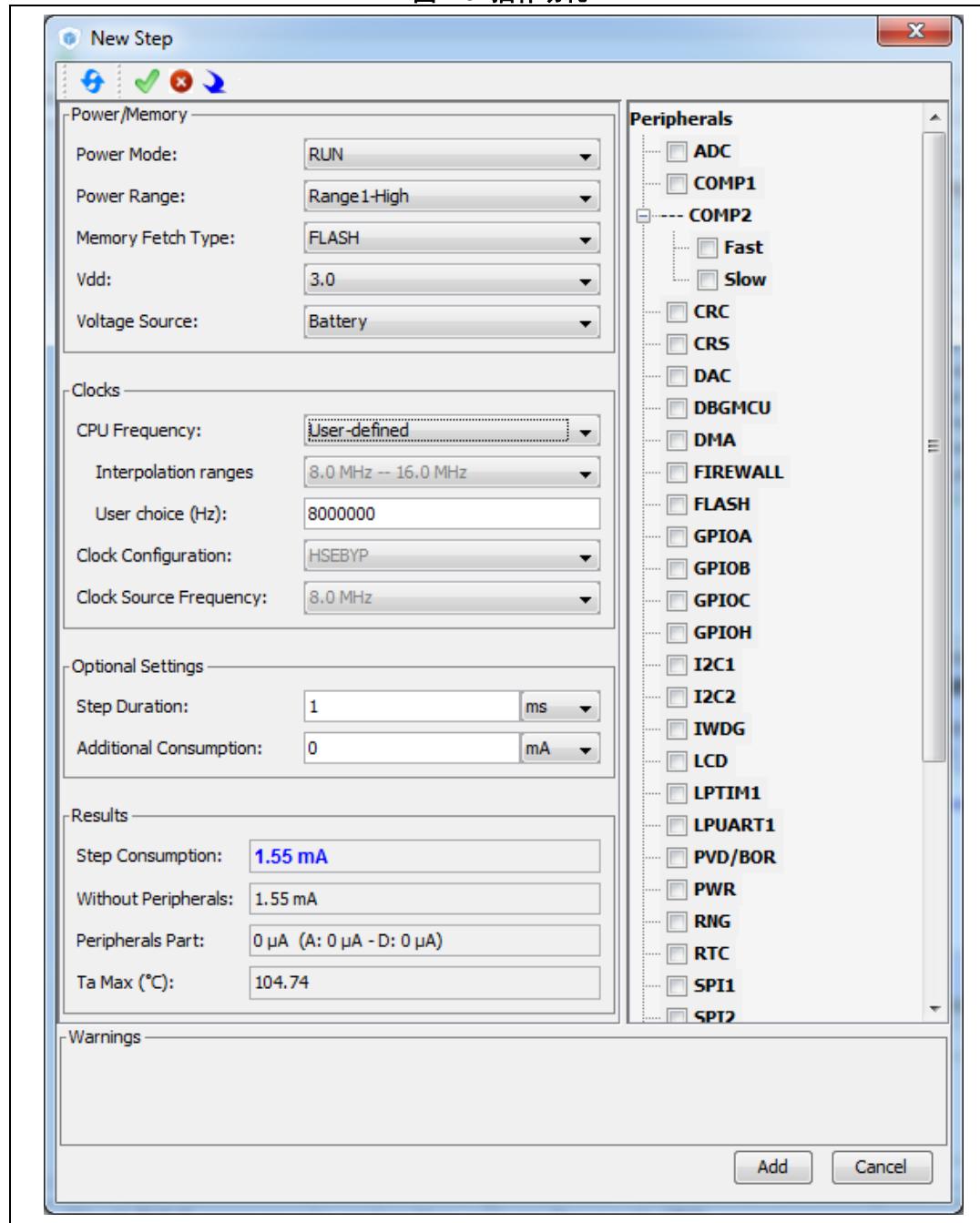
按以下步骤配置新步骤：

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 129）。

图129. 插补功耗

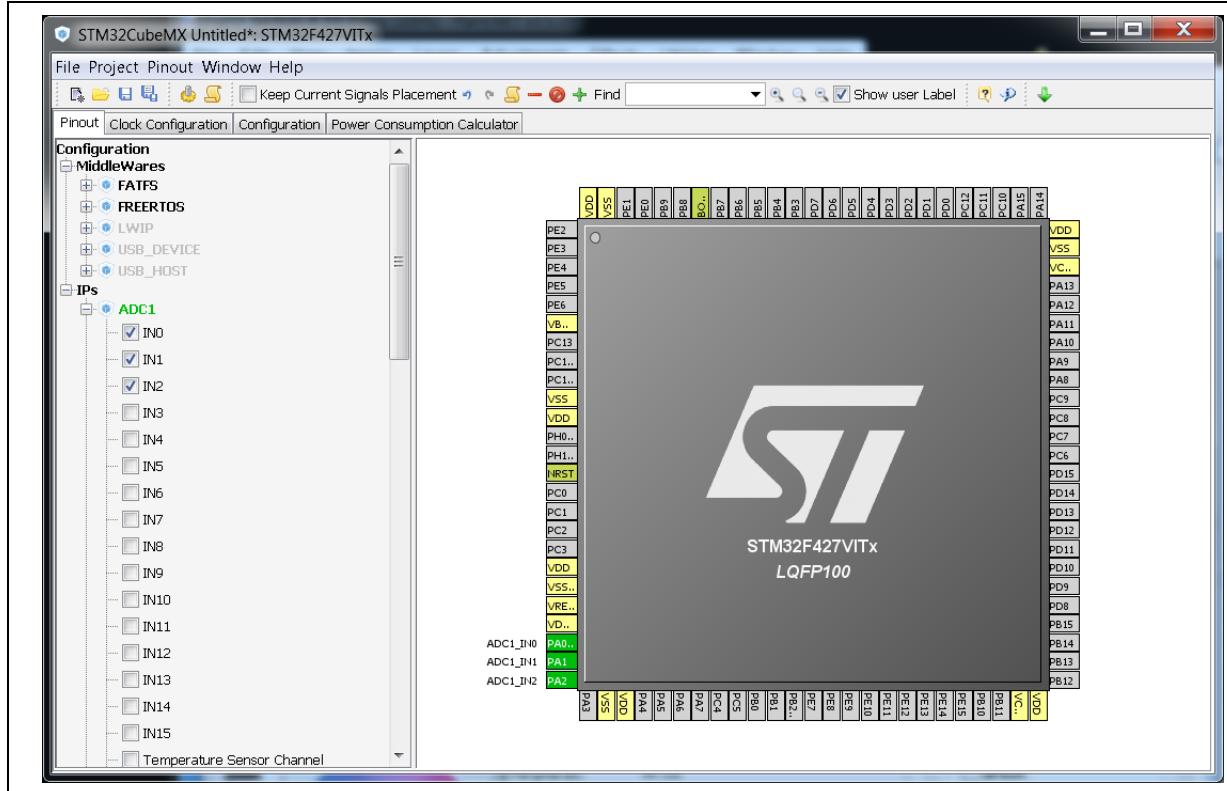


导入引脚布局

图 130 显示了引脚布局视图中的ADC配置示例：在功耗计算器视图中单击导入引脚布局，以选择ADC外设和GPIO A (图 131)。

导入引脚布局按钮  允许自动选择已在引脚布局视图中配置过的外设。

图130. 在引脚布局视图中选择的ADC

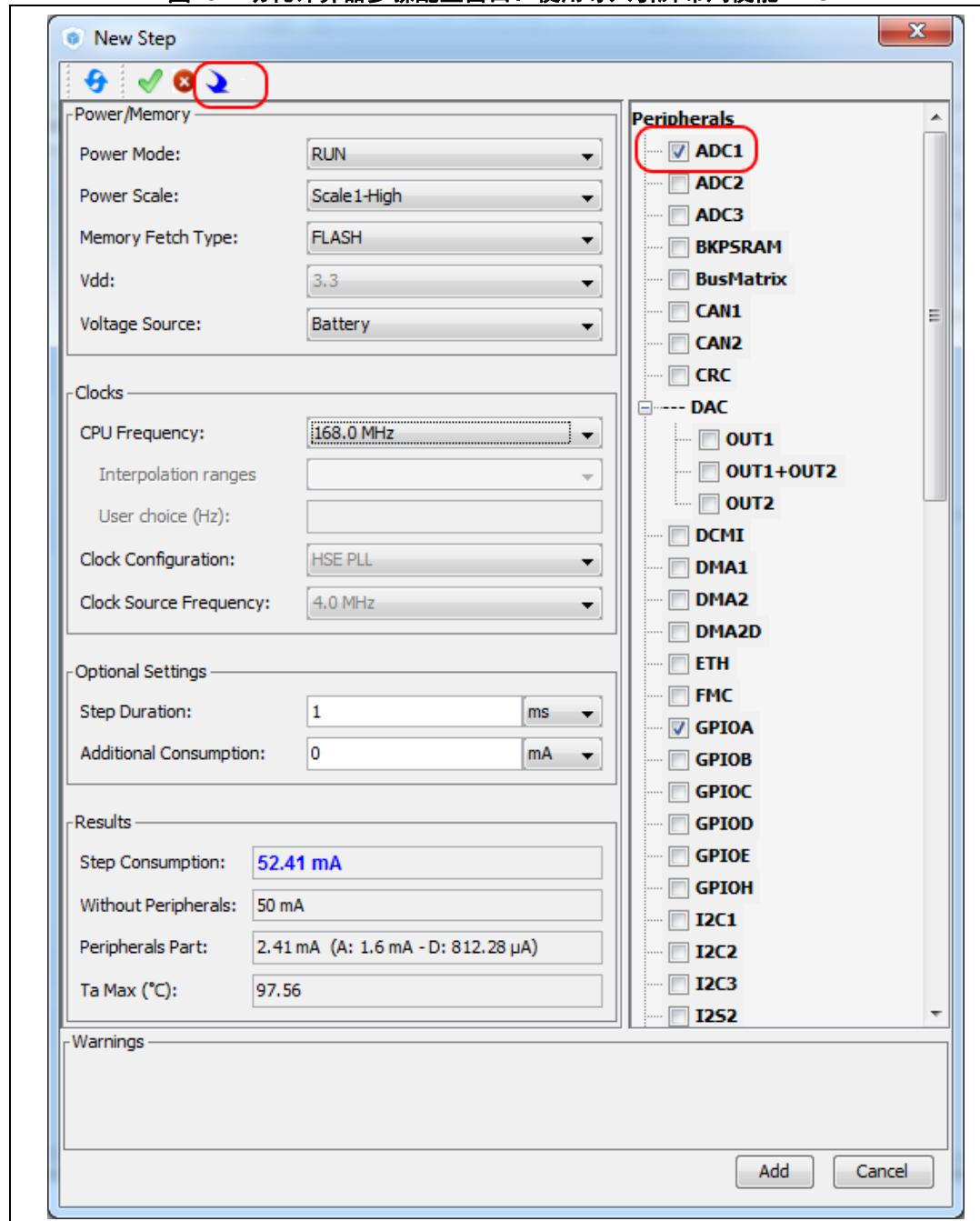


选择/取消选择所有外设

单击全选按钮 可一次选择所有外设。

单击取消全选 可删除作为消耗贡献者的外设。

图131. 功耗计算器步骤配置窗口：使用导入引脚布局使能ADC

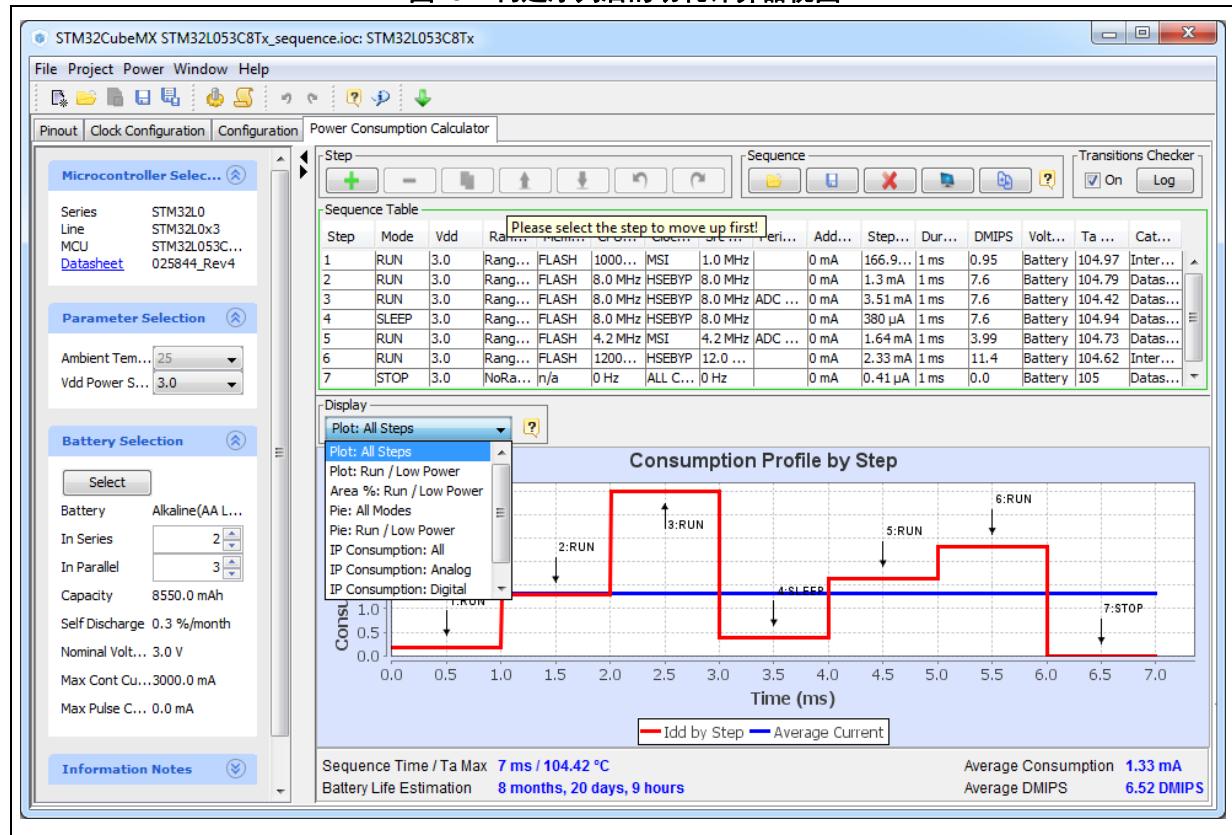


5.15.3 管理用户定义的功耗系列并审查结果

配置功耗系列会引起功耗计算器视图更新（参见图 132）：

- 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表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等）
- 结果总结提供了总序列时间，最高环境温度 ($T_{A\text{MAX}}$)，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132. 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）

通过分别点击  和  可保存或删除当前序列。

此外，可在当前视图中加载以前保存的序列或通过点击  打开进行比较（参见 [图 133](#)）。

图133. 序列表管理功能



要加载以前保存的序列：

1. 单击加载按钮 .
2. 浏览以选择要加载的序列。

要打开以前保存的序列以进行比较：

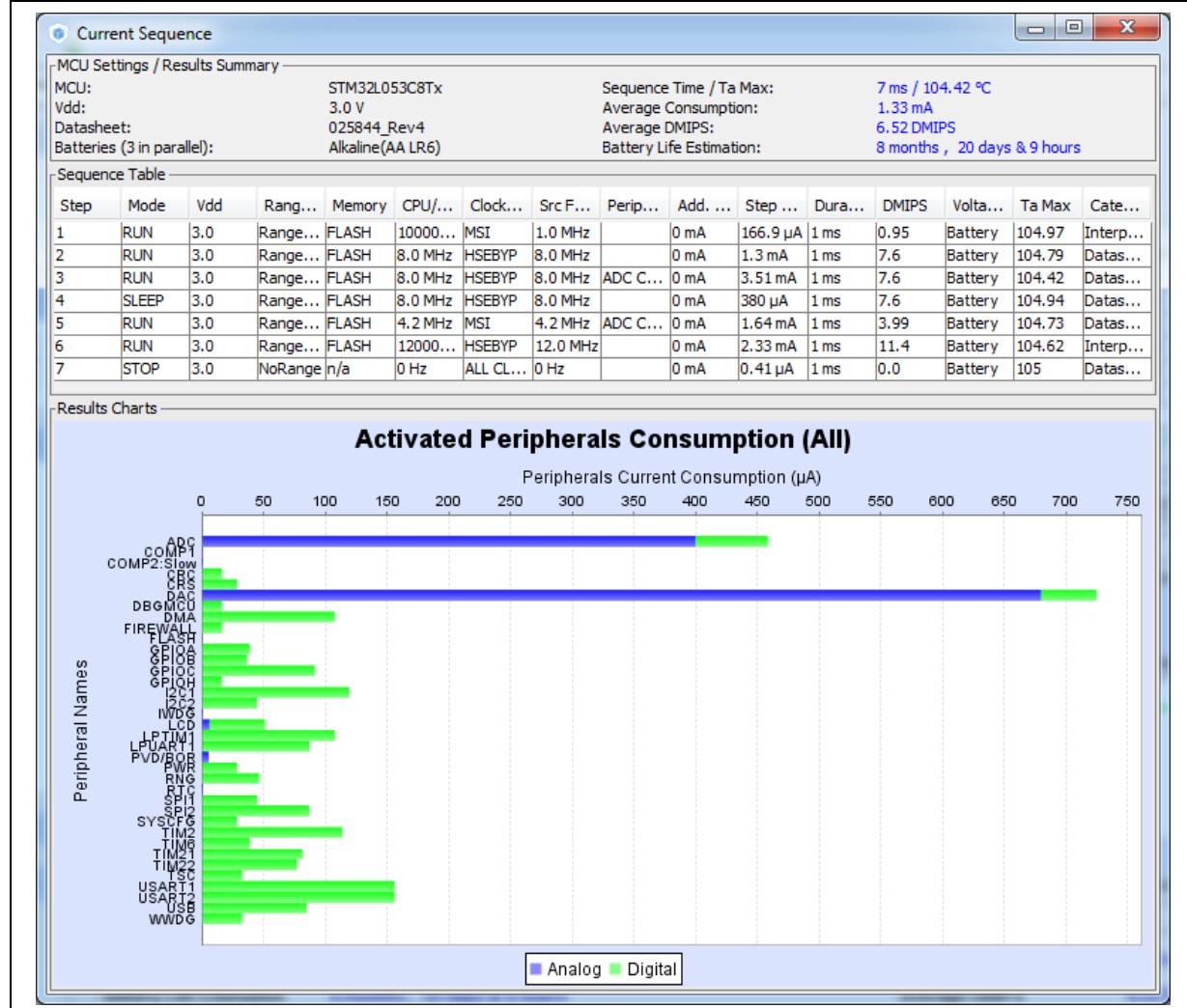
1. 点击比较按钮 .
2. 浏览并选择要与当前序列进行比较的.pcs序列文件。将打开一个新窗口，该窗口显示所选序列详情。

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（序列步骤、饼图、每个外设的消耗等）。您也可以点击**外部显示**，以在专用窗口中打开图表（参见图 134）。

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134. 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 135）：

- 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135. 结果区域描述

Results Summary			
Sequence Time / T_a Max	7 ms / 104.42 °C	Average Consumption	1.33 mA
Battery Life Estimation	8 months , 20 days & 9 hours	Average DMIPS	6.52 DMIPS

5.15.4 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D：STM32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：PLL ON、LSE旁路、AHB预分频值、具有一定占空比的LCD等

- 外设

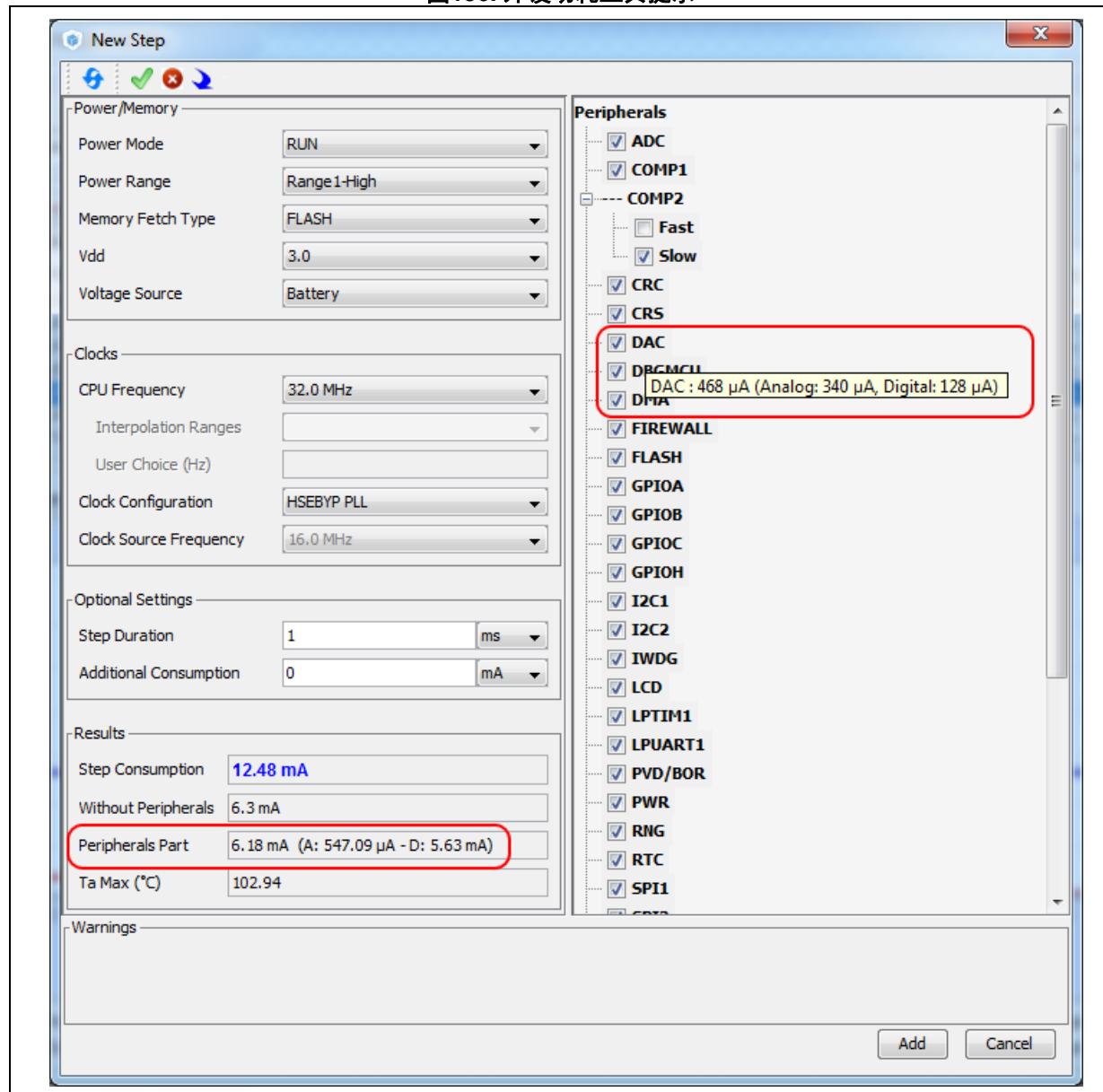
外设列表显示了可用于所选功耗模式的外设。功耗在假定外设仅处于时钟模式的情况下给出（如未被运行程序使用）。可使能或禁用每个外设。在工具提示中显示外设的单独功耗。在步骤结果区域中提供外设模拟和数字部分所引起的总功耗（参见图 136）。

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**），
- 或之前定义的引脚布局配置中的全部外设（**导入引脚布局**）。

在计算功耗时，仅考虑选定和已使能的外设。

图136. 外设功耗工具提示



- 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，如果应用在运行模式花费30%的时间、在睡眠模式花费20%的时间，在停止模式花费50%的时间，则用户必须配置由30 ms的运行、20 ms的睡眠和50 ms的停止组成的3步序列。

- 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）。

5.15.5 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。

5.15.6 SMPS特性

某些微控制器（如STM32L496xxxxP）允许连接外部切换模式电源（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 137](#)）。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 138](#)）。然后，用户可以为当前序列选择不同的SMPS模型（参见[图 139](#)、[图 140](#)和[图 141](#)）。
- 通过使能SMPS检查器检查当前序列中的无效SMPS跳变。
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 142](#)）。
- 为每个步骤配置SMPS模式（参见[图 143](#)）。
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137. 为当前项目选择SMPS

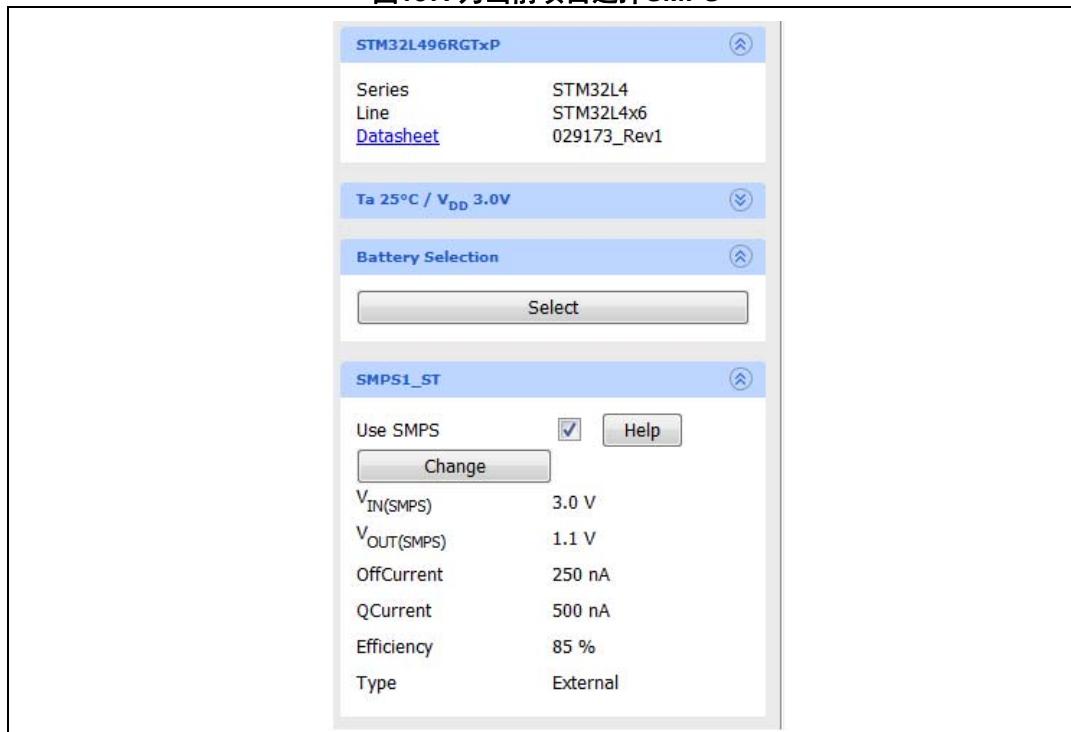


图138. SMPS数据库 - 添加新的SMPS模型

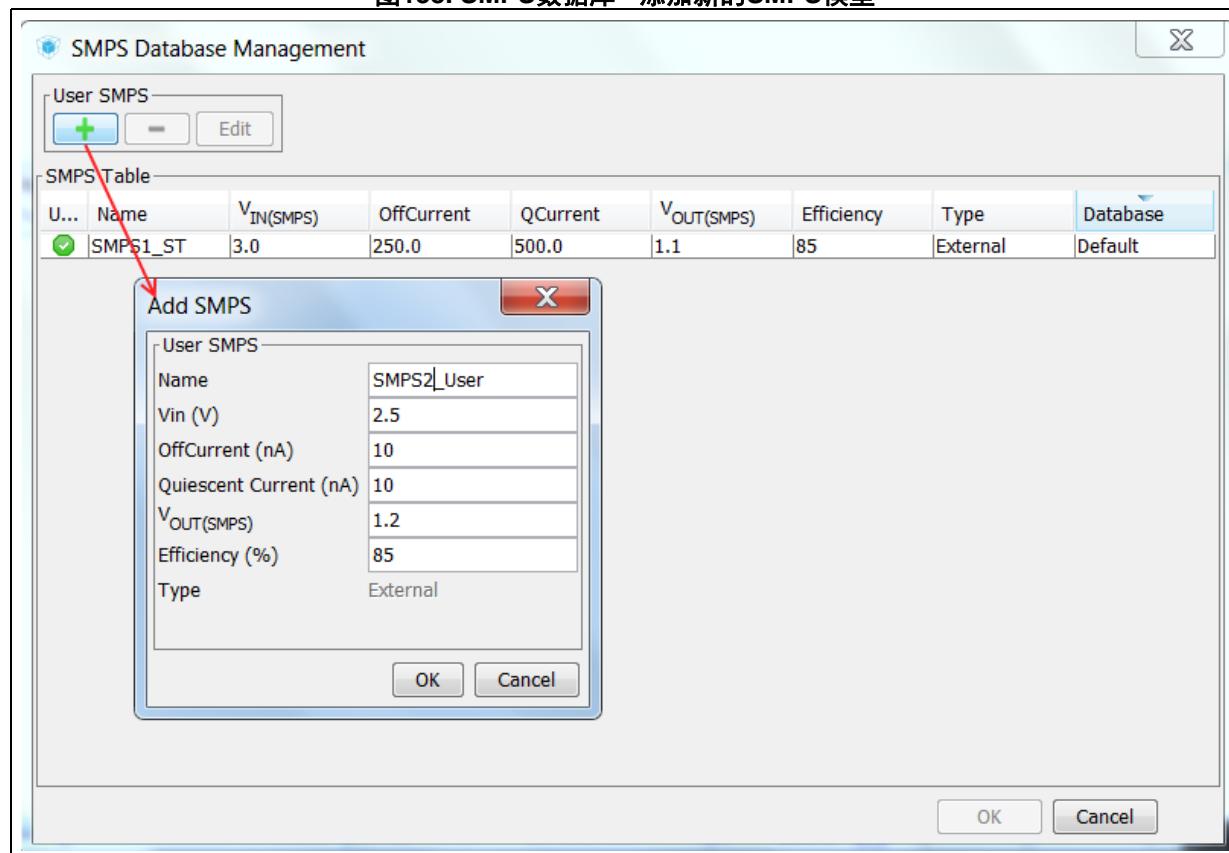


图139. SMPS数据库 - 选择不同的SMPS模型

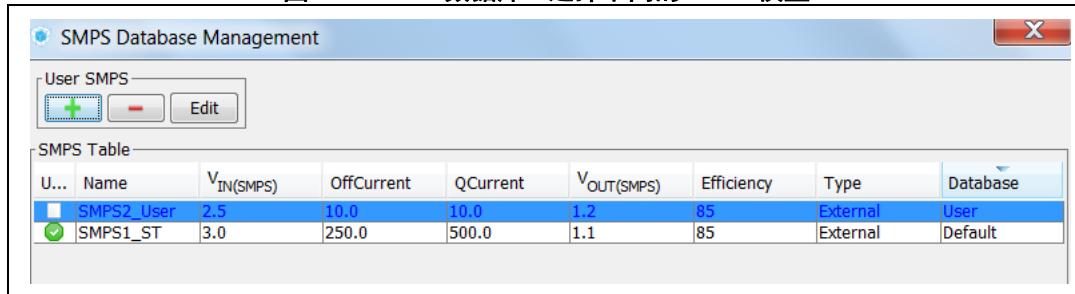


图140. 使用新的SMPS模型更新当前项目配置

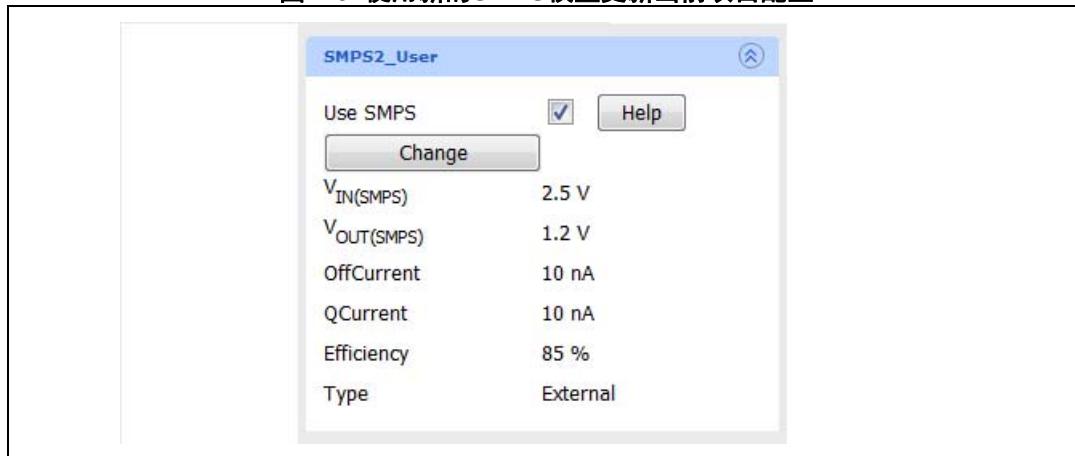


图141. 已选择新模型的SMPS数据库管理窗口

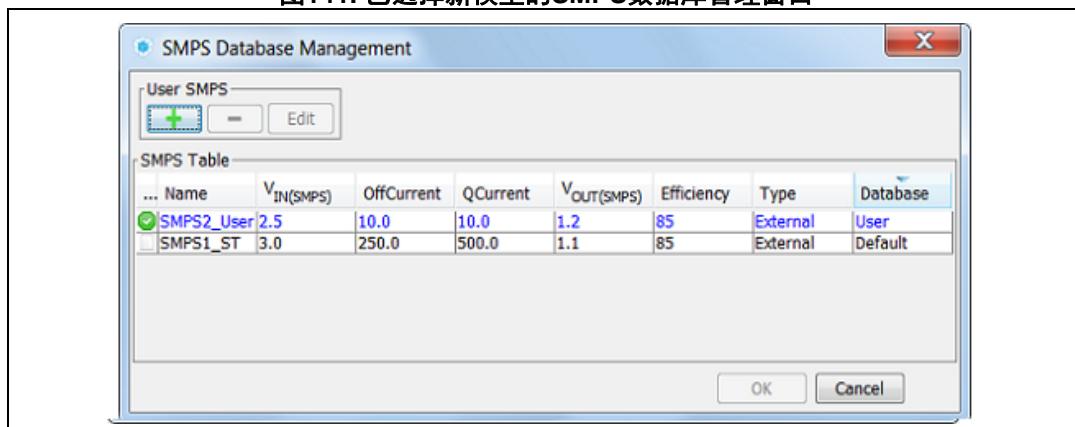


图142. SMPS跳变检查器和状态图助手窗口

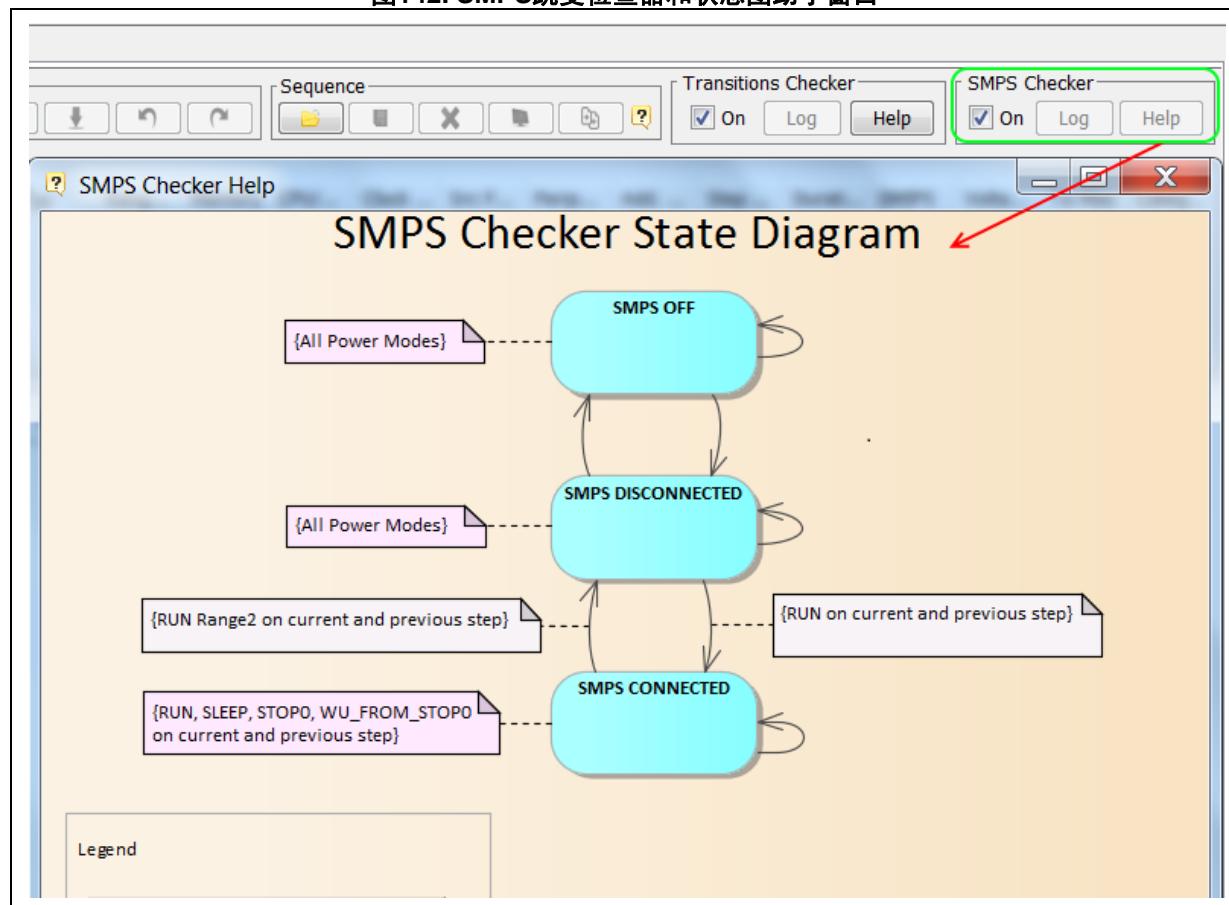
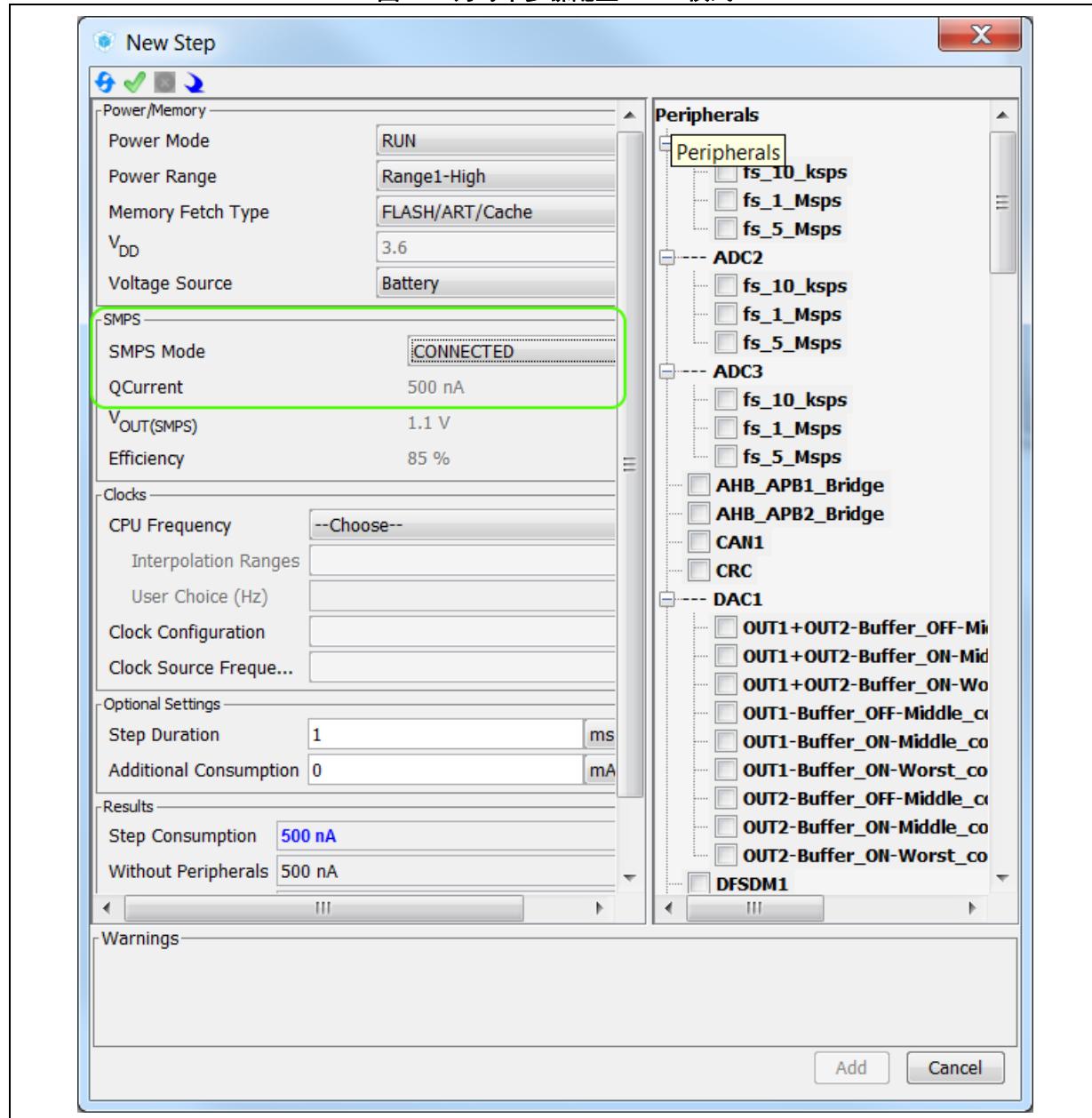


图143. 为每个步骤配置SMPS模式



6 STM32CubeMXC代码生成概述

有关与代码生成和C项目设置相关的主题，请参见[项目菜单](#)。

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**（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布局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 144与图 145）：

```
#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144. 生成定义语句的引脚标签

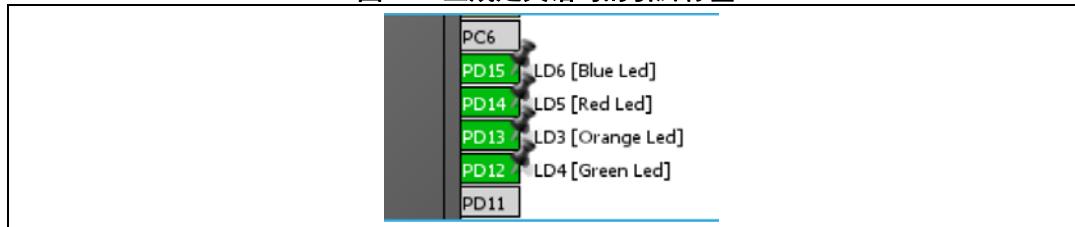
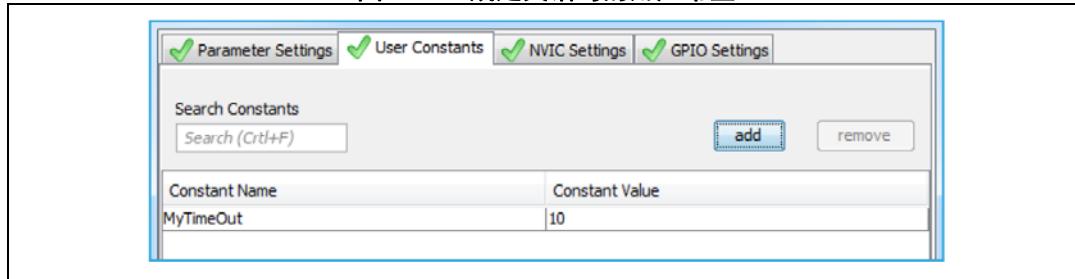


图145. 生成定义语句的用户常量

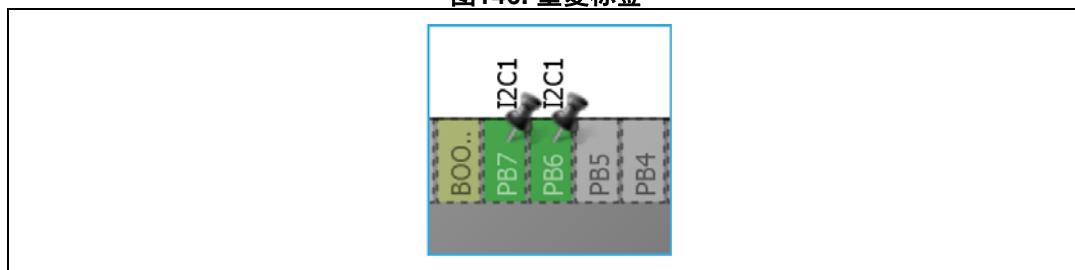


如果出现重复的标签，则会添加由引脚端口字母与引脚索引号组成的唯一后缀，并将其用于产生相关定义语句。

在图 146 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146. 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STM32F0、STM32F3、STM32L0、STM32L1、STM32L4、STM32L4+、STM32F2、STM32F4和STM32F7系列，STM32CubeMX允许产生基于外设HAL驱动或外设底层（LL）驱动的外设初始化代码。

通过“项目设置”窗口进行选择（参见“[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	HAL和LL混合	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

生成的头文件	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

生成的源文件	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： – 配置Flash预取、指令和数据缓存 – 选择SysTick计时器作为时基源 – 设置NVIC组优先级 – MCU低级MCU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⁾ 。
LL_init()	未使用	在main.c中生成和调用	未使用	系统和内存中断配置
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] – 当为<外设>选择LL驱动时，将执行以下函数生成和调用[2]	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]	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]	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147. 基于HAL的外设初始化：uart.c代码片段

```
USART外设初始化 — 基于HAL
void MX_USART1_UART_Init(void)
{
    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)
{
    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)
{
    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148. 基于LL的外设初始化：uart.c代码片段

```

使用LL驱动进行USART外设初始化
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
    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);

    外设资源配置
    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;

    外设配置
    LL_USART_Init(USART1, &USART_InitStruct);
    LL_USART_ConfigAsyncMode(USART1);
}

```

图149. HAL与LL: main.c代码片段

main.c HAL-based <pre> /* * 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(); } </pre>	main.c LL-based <pre> /* * 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(); } </pre>
--	--

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 150: extra_templates文件夹 - 默认内容](#)）。

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")]

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/extratemplates文件夹或任何其他文件夹中的STM32CubeMX安装路径下。

然后，用户将通过**模板设置**窗口为给定项目选择与其项目有关的模板文件，可通过**项目设置**菜单访问此窗口（参见[“项目设置”窗口](#)）

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 151：包含用户模板的extra_templates文件夹](#)）。

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 152：具有相应的自定义生成文件的项目根文件夹](#)）。

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 153：模板的用户自定义文件夹](#)）。

图150. extra_templates文件夹 - 默认内容

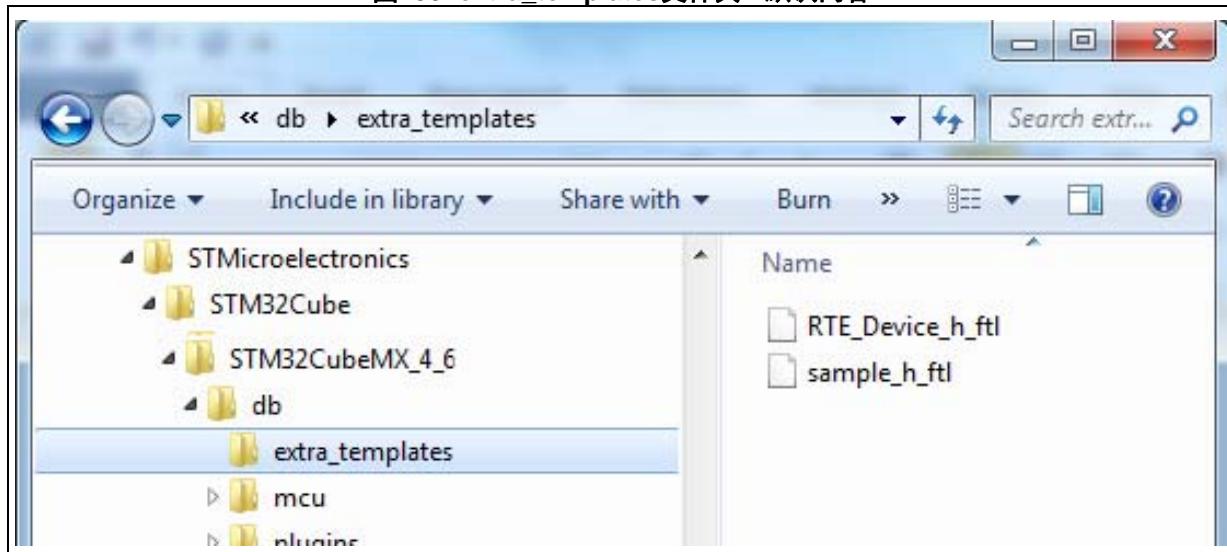


图151. 包含用户模板的extra_templates文件夹

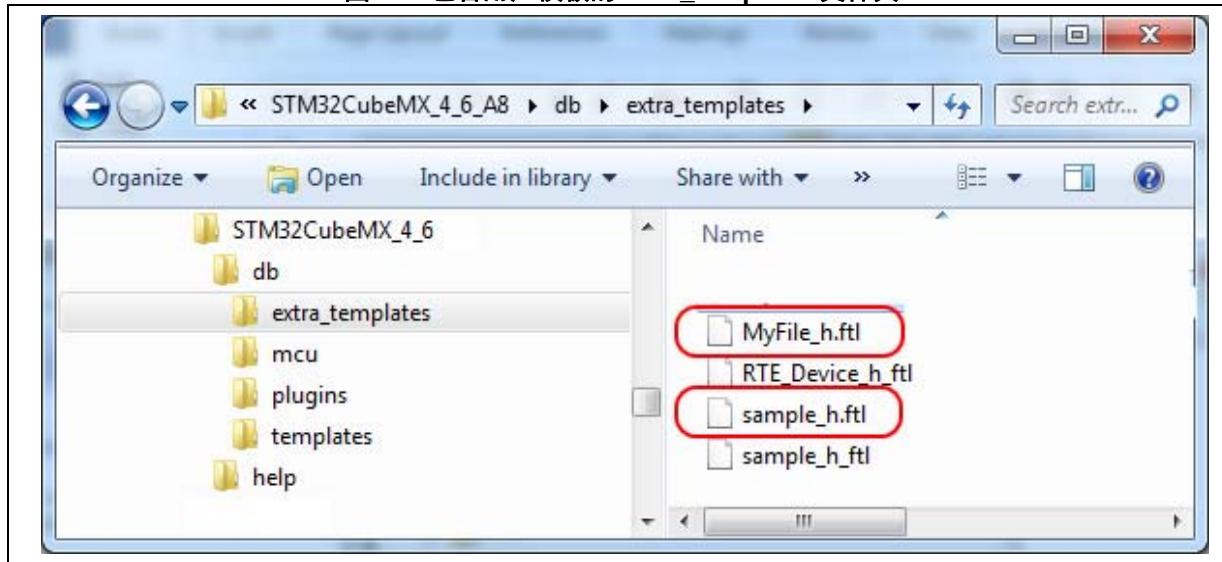


图152. 具有相应的自定义生成文件的项目根文件夹

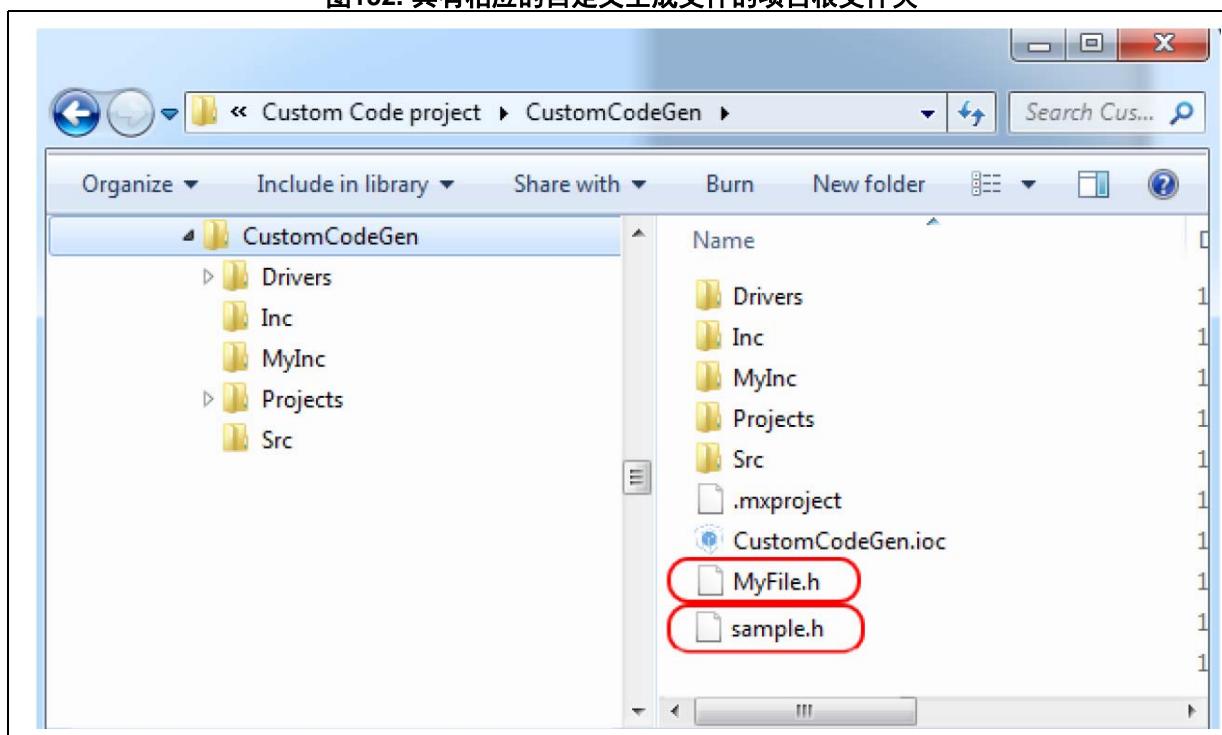


图153. 模板的用户自定义文件夹

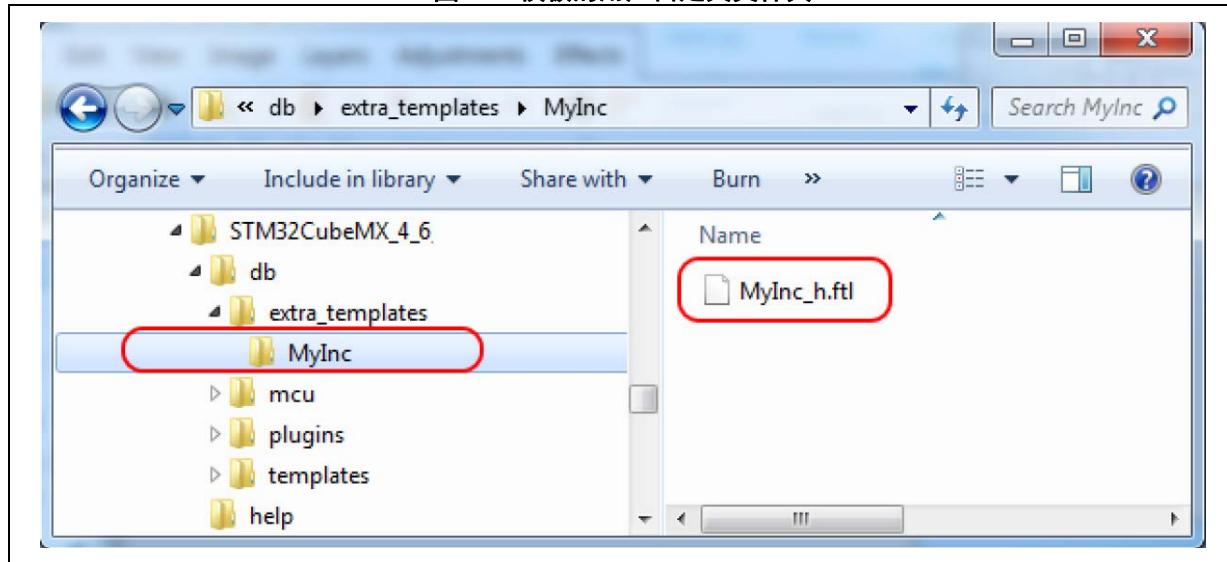
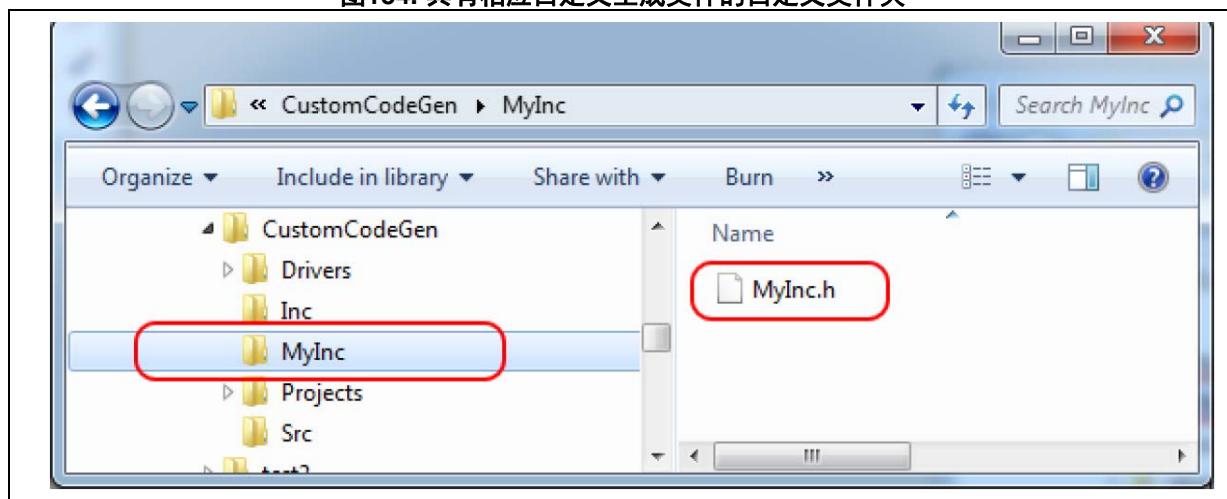


图154. 具有相应自定义生成文件的自定义文件夹



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

在本例中，通过使用STM32CubeMX板选择器选择STM32F429I-DISCO板来创建新项目。在项目设置下选择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：

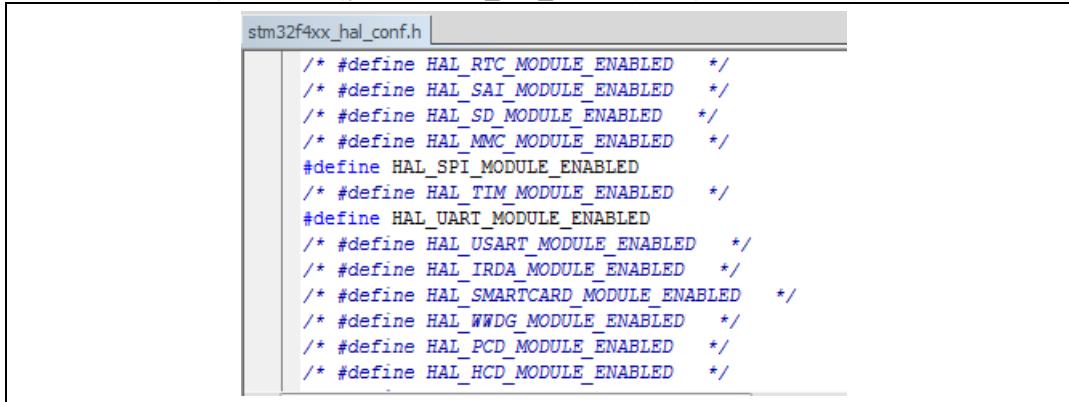
- EWARM文件夹中的MyF429IDiscoProject.ewp项目文件（参见图 155），
- 项目Inc文件夹中的stm32f4xx_hal_conf.h文件（参见图 156）
- EWARM用户界面中的项目视图，如图 157和图 158中所示。

图155. 为预处理定义语句更新.ewp项目文件 (EWARM IDE)



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

图156. 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157. 添加到EWARM IDE中的组的新组和新文件

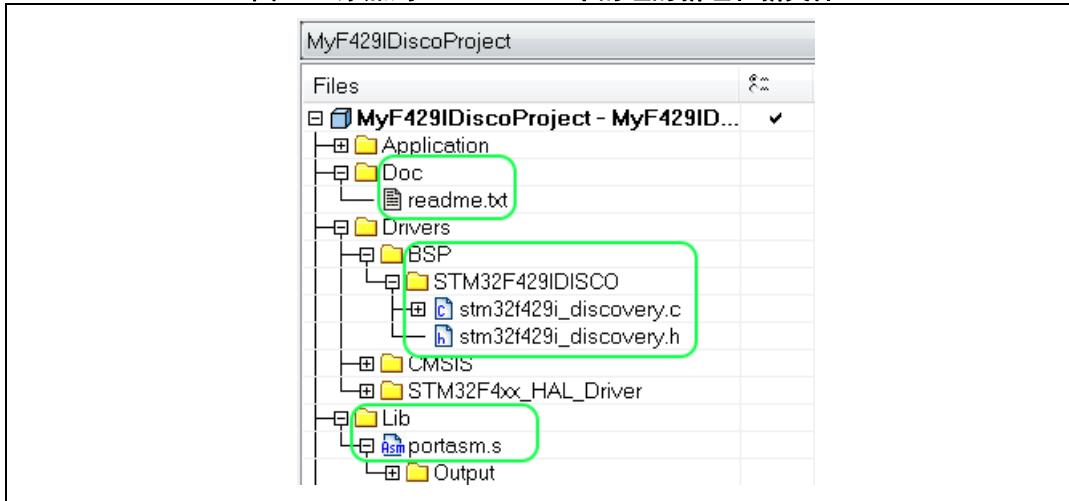
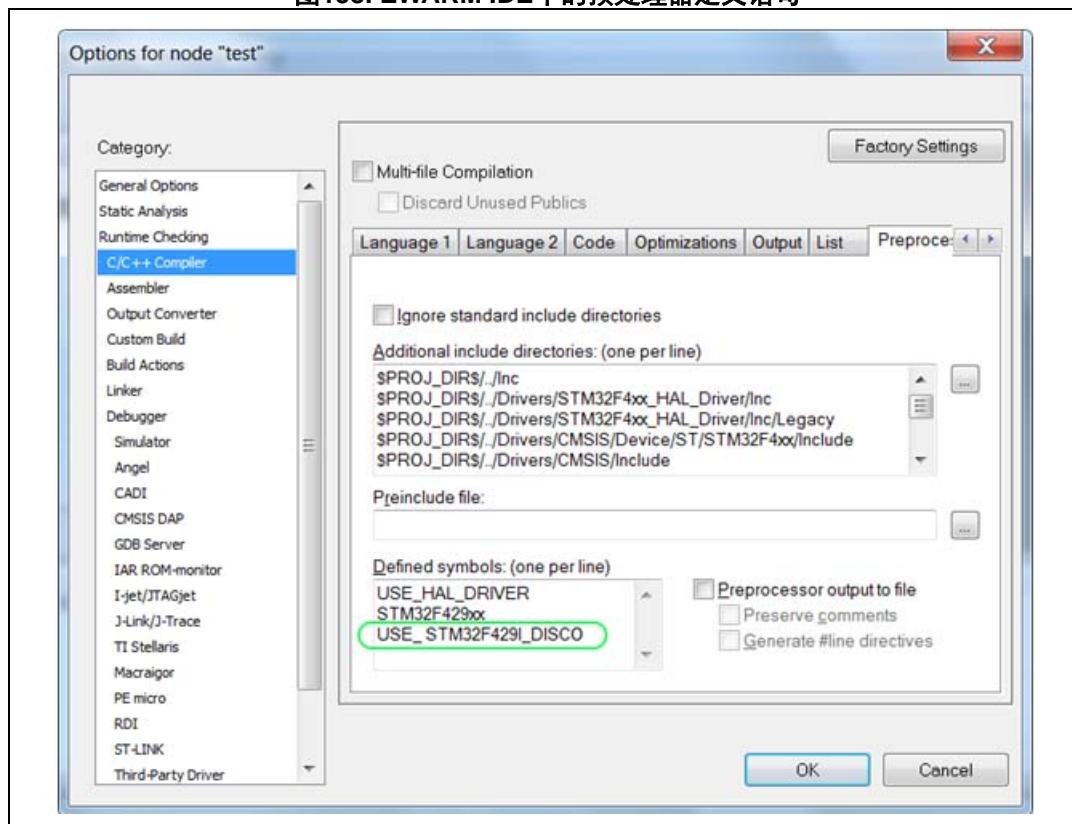


图158. EWARM IDE中的预处理器定义语句



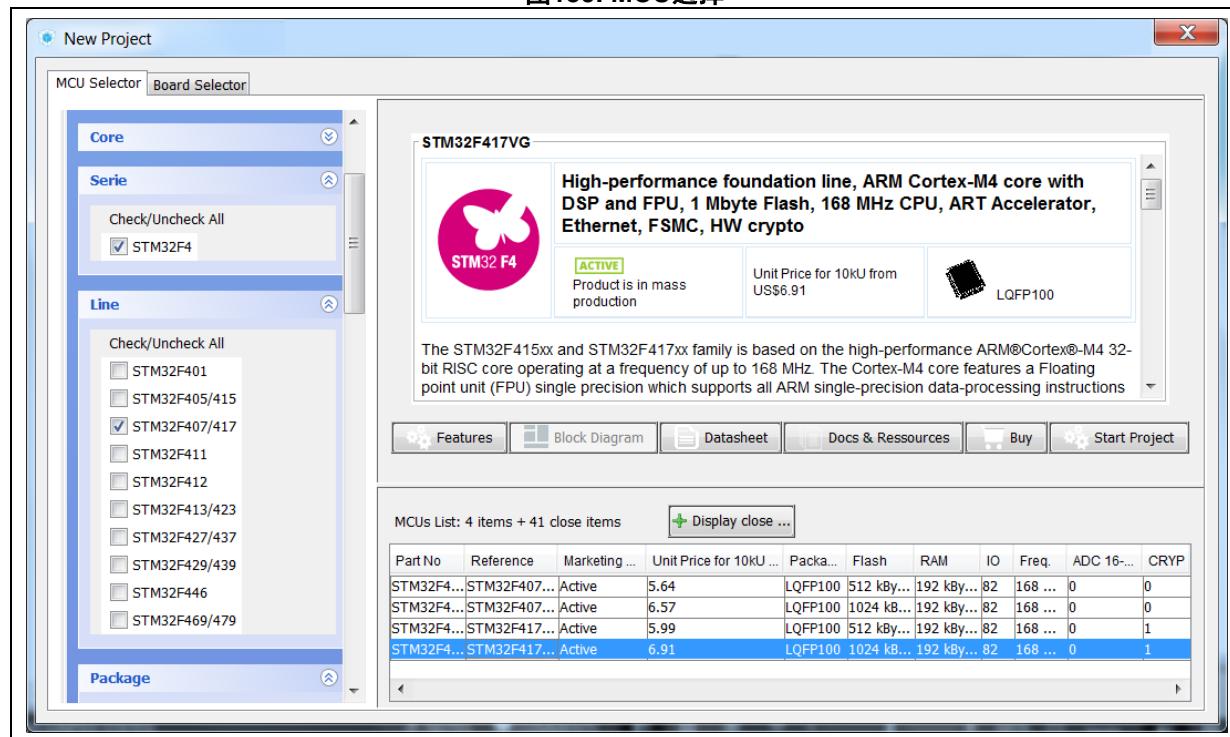
7 教程一使用STM32F4从引脚布局到生成项目C代码

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翻转应用为例。

7.1 创建一个新STM3CubeMX项目

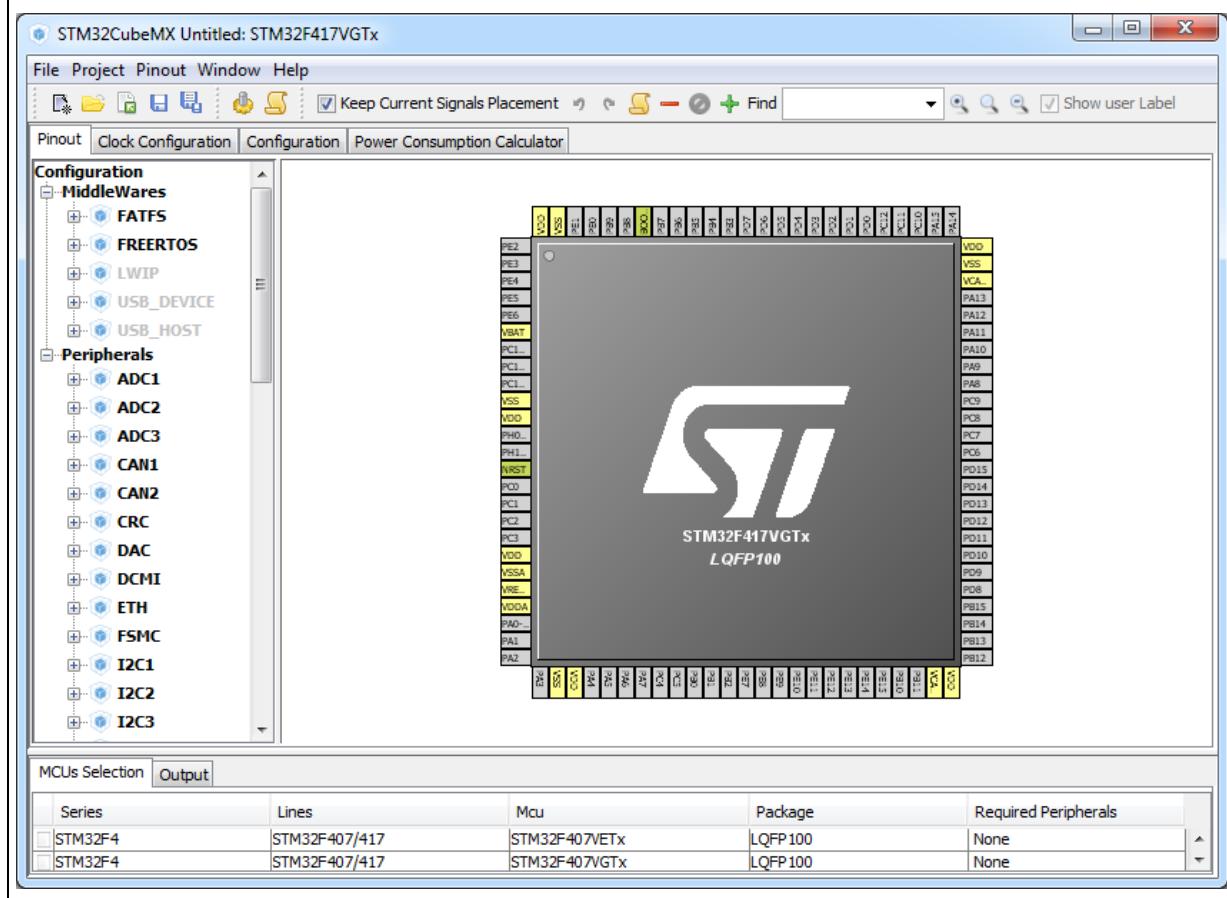
1. 从主菜单栏中选择文件 > 新建项目，或从欢迎页面选择新建项目。
2. 选择MCU选择器选项卡，并通过将STM32F4选为“系列”、将STM32F407选为“产品线”和将LQFP100选为“封装”来筛选STM32产品（参见图 159）。
3. 从MCU列表中选择STM32F407VGTx，然后点击“确定”。

图159. MCU选择



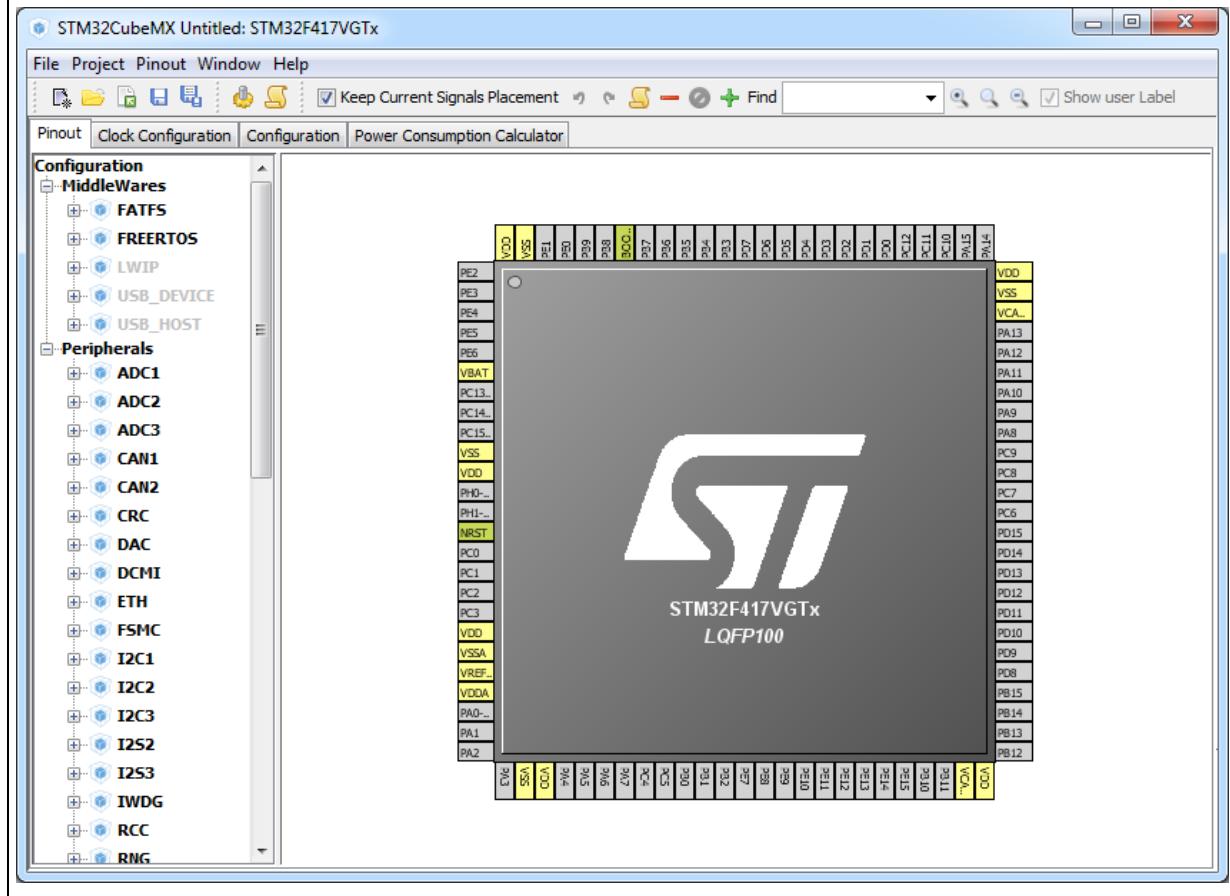
然后使用选定的MCU数据库填充STM3CubeMX视图（参见图 160）。

图160. 具有MCU选择的引脚布局视图



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 161）。

图161. 无MCU选择窗口的引脚布局视图

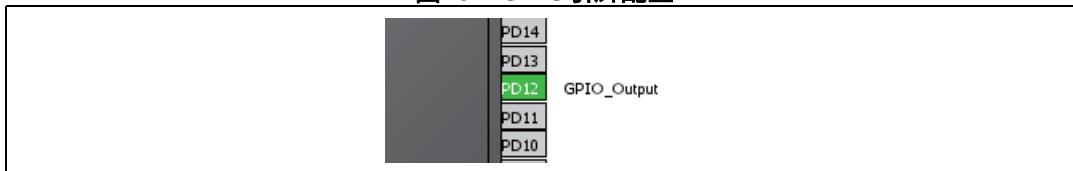


7.2 配置MCU引脚布局

有关菜单、高级操作和冲突解决方案的详细说明，请参见[STM32CubeMX用户界面](#)和[附录A：STM32CubeMX引脚分配规则](#)。

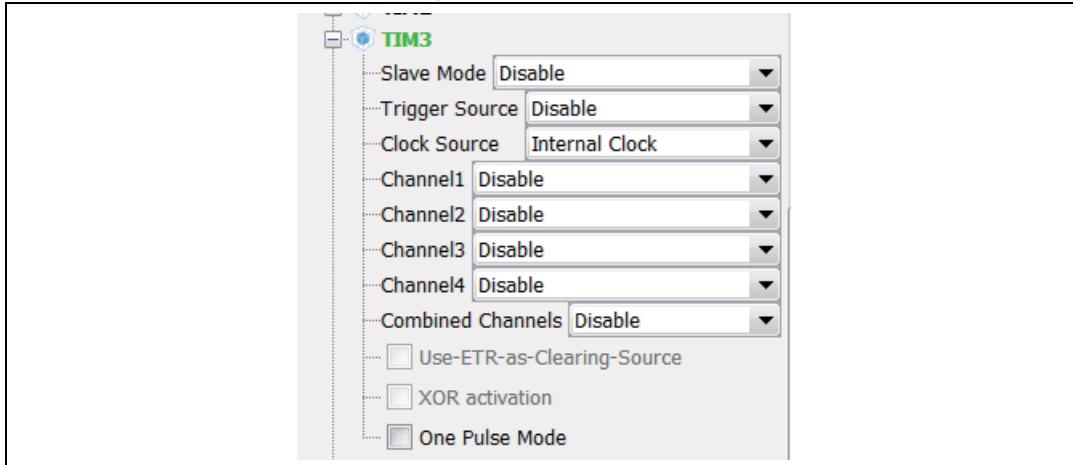
1. 默认情况下，STM32CubeMX显示引脚布局视图。
2. 默认情况下， Keep Current Signals Placement 未选中，以允许STM32CubeMX移动外设功能和找到最佳引脚分配，即可支持最大外设模式数量的引脚分配。
由于MCU引脚配置必须与STM32F4DISCOVERY板相匹配，通过使能STM32CubeMX的 Keep Current Signals Placement 可保持给定引脚的外设函数分配（映射）。
该设置将另存为用户偏好，以便在重新打开工具或加载其他项目时进行恢复。
3. 选择所需的外设和外设模式：
 - a) 通过在芯片视图中右键单击PD12，可配置GPIO在STM32F4DISCOVERY绿色LED上输出信号，然后选择GPIO_output：

图162. GPIO引脚配置



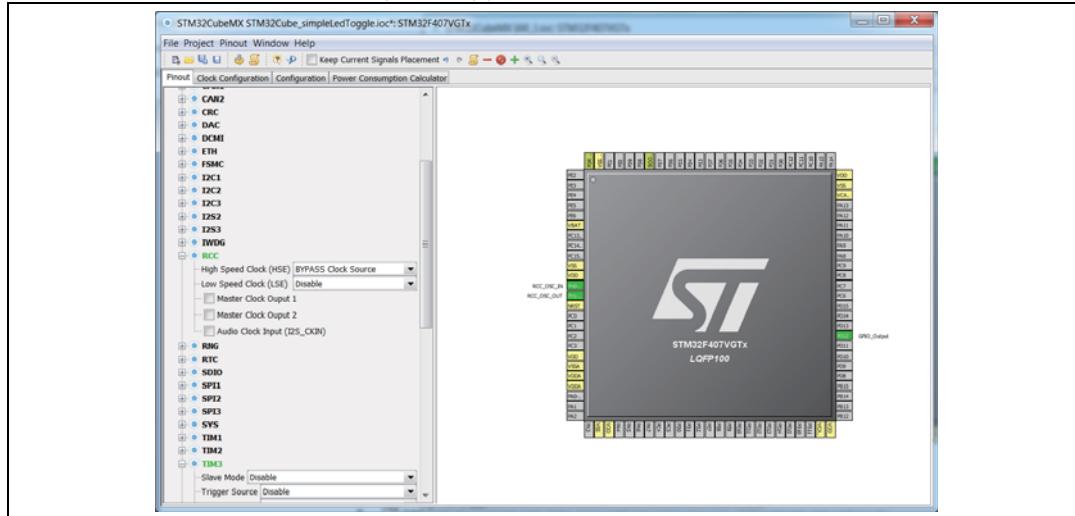
- b) 使能将用作LED翻转时基的定时器。可通过在外设树中将内部时钟选作TIM3时钟源来完成此操作（参见图 163）。

图163. 定时器配置



- c) 您也可以配置RCC，以将外部振荡器用作潜在时钟源（参见图 164）。这样就完成了本示例的引脚布局配置。

图164. 简单的引脚布局配置



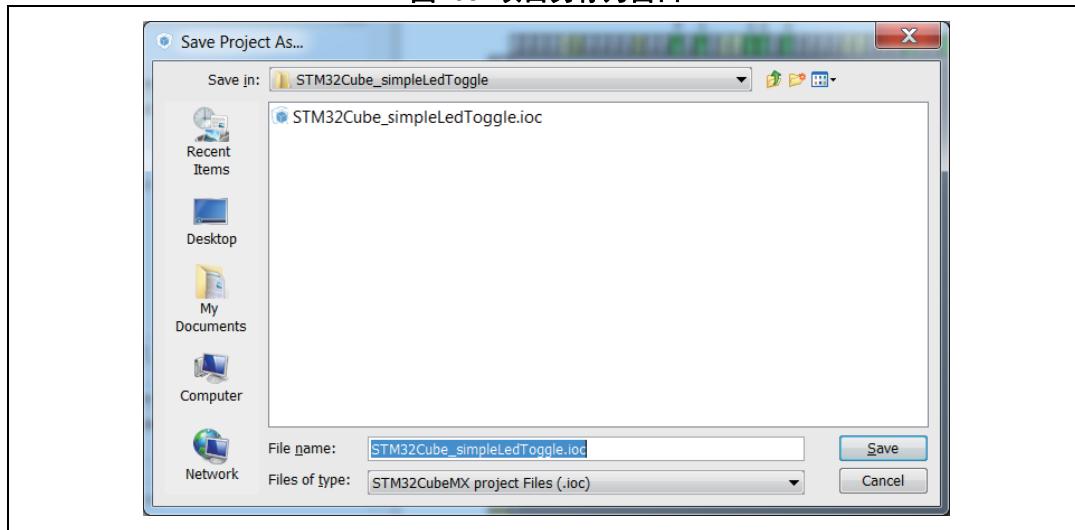
注：从STM32CubeMX 4.2开始，用户可通过直接从“板选择器”选项卡直接加载ST探索板配置来跳过引脚布局配置。

7.3 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165. 项目另存为窗口



2. 单击 以使用其他名称或位置保存项目。

7.4 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 166）。然后为项目生成.ioc文件以及具有相同名称的.pdf和.txt报告。

如果选择“否”，则只要求提供报告的名称和位置。

操作成功后会显示确认信息（参见图 167）。

图166. 生成项目报告 - 新项目创建

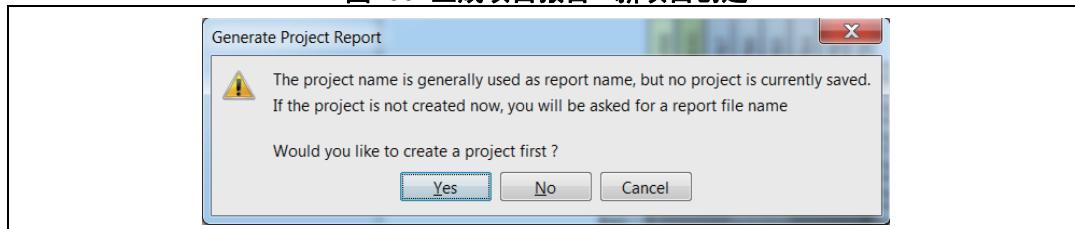
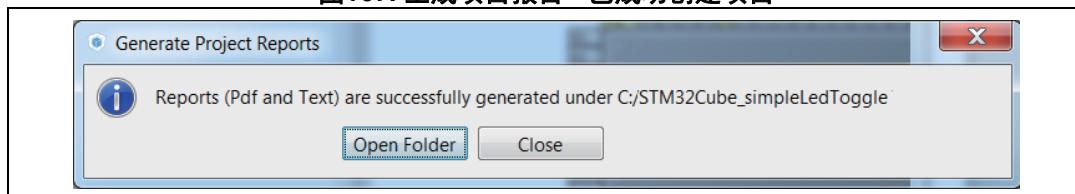


图167. 生成项目报告 - 已成功创建项目



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。

7.5 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）。

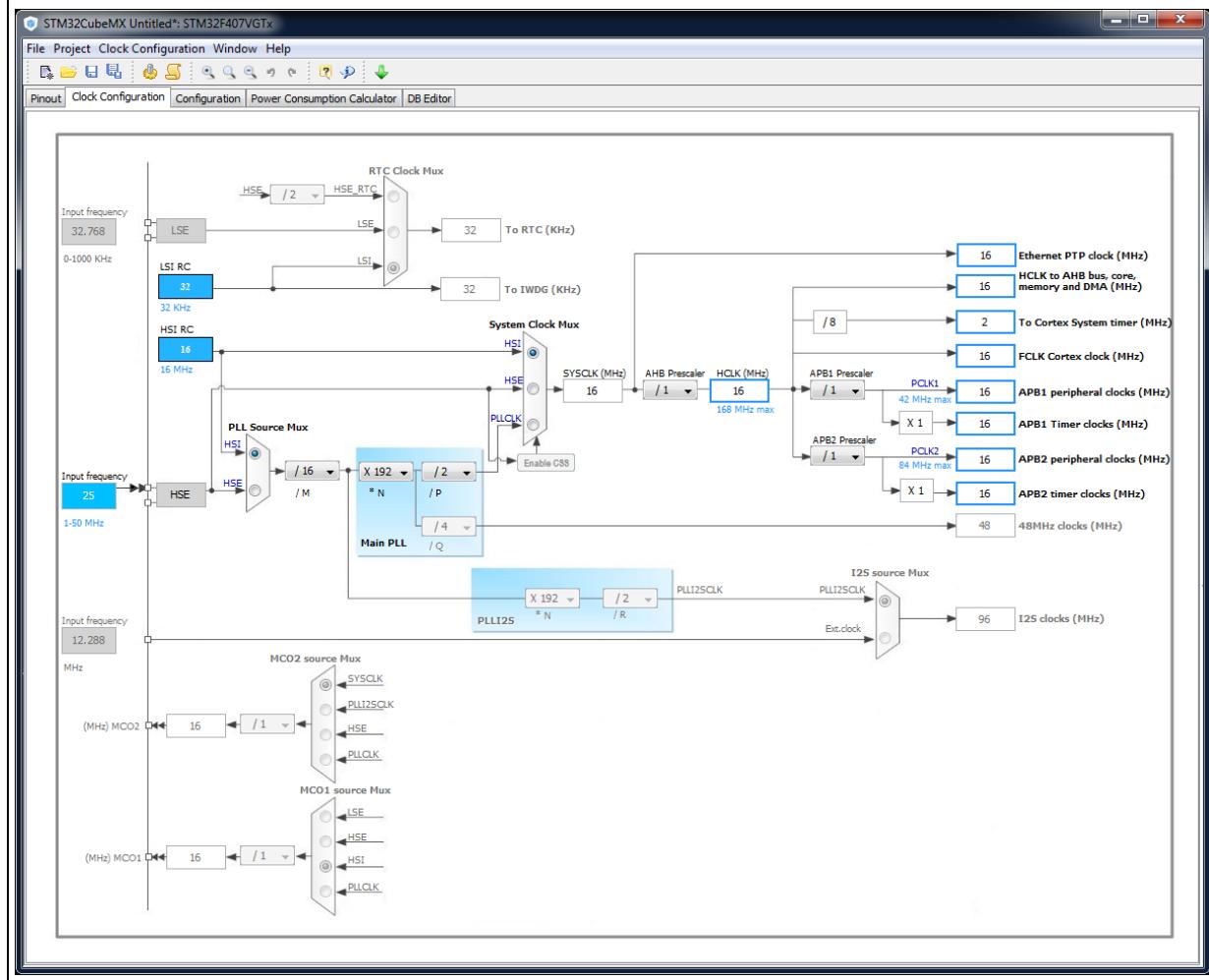
STM32CubeMX使用时钟树视图中定义的时钟设置为每个外设时钟生成初始化C代码。作为项目main.c和stm32f4xx_hal_conf.h中RCC初始化的一部分，在生成的C代码中执行时钟设置（HSE、HSI和以赫兹表示的外部时钟值）。

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 168）。

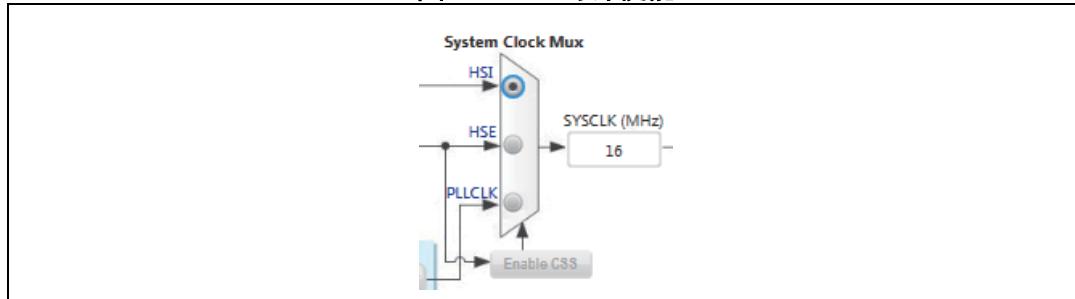
内部（HSI、LSI）、系统（SYSCLK）和外设时钟频率字段无法编辑。系统和外设时钟可通过选择时钟源来调节，也可以选择使用PLL、预分频器和倍频器来调节。

图168. 时钟树视图



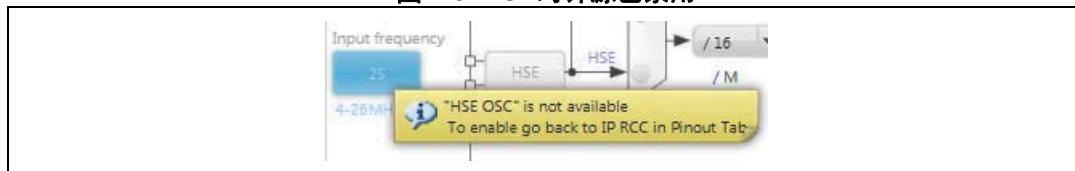
2. 首先选择将驱动微控制器系统时钟的时钟源（HSE、HSI或PLLCLK）。
在本教程的示例中，选择HSI以使用内部16 MHz时钟（参见图 169）。

图169. HSI时钟使能



要使用外部时钟源（HSE或LSE），应在引脚布局视图中配置RCC外设，因为引脚将用于连接外部时钟晶振（参见图 170）。

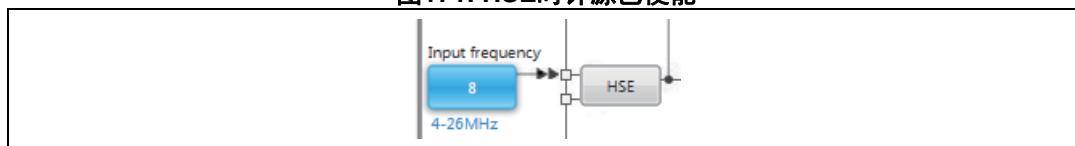
图170. HSE时钟源已禁用



STM32F4DISCOVERY板的其他时钟配置选项包括：

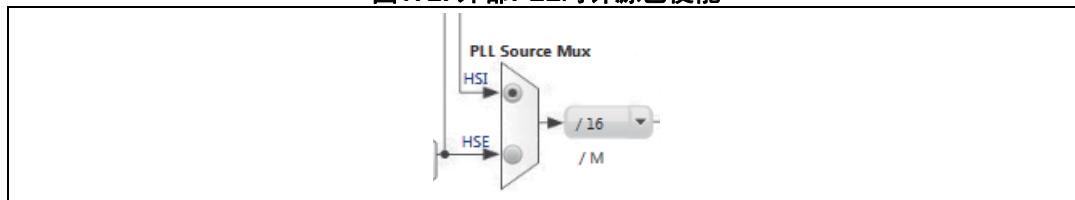
- 选择外部HSE源并在HSE输入频率框中输入8，因为探索板连接了8 MHz晶振：

图171. 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172. 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。

7.6 配置MCU初始化参数

提醒

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。

7.6.1 初始条件

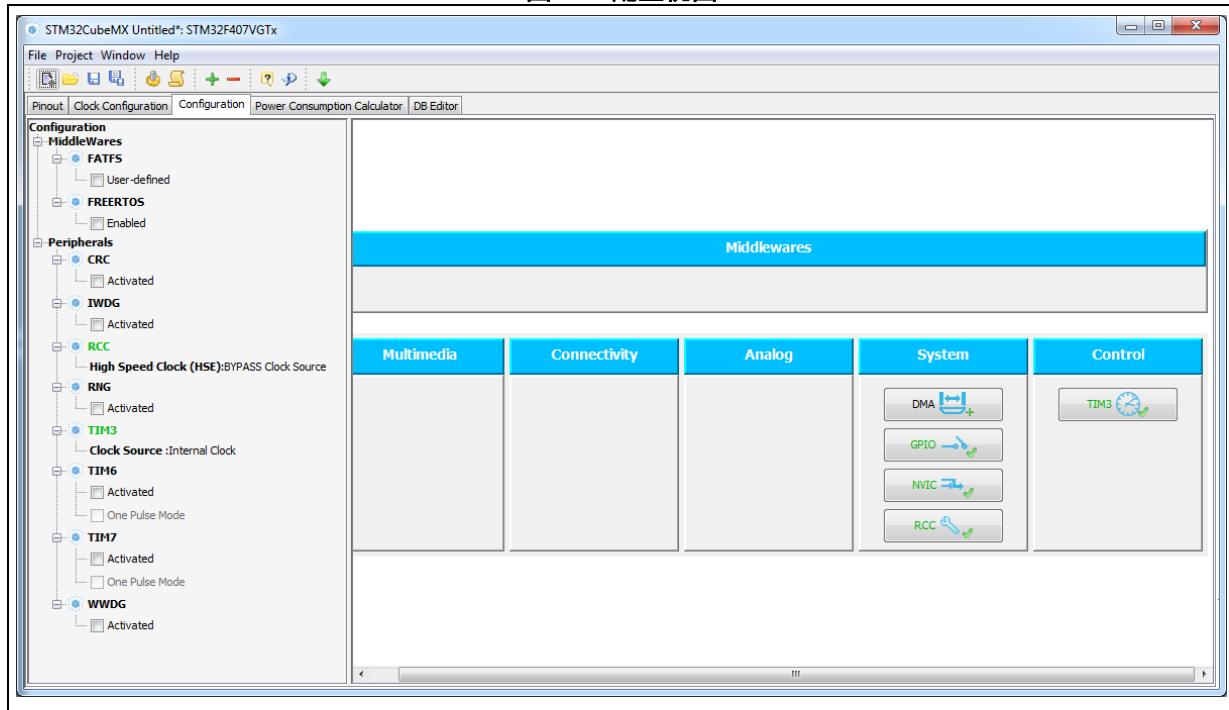
选择配置选项卡，以显示配置视图（参见图 173）。

可在**外设和中间件树**面板中禁用或使能不影响引脚布局的外设和中间件模式。只能通过**引脚布局**选项卡选择影响引脚分配的模式。

未正确配置外设时，将在主面板中显示工具提示和警告消息（有关详细信息，请参见 [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）。

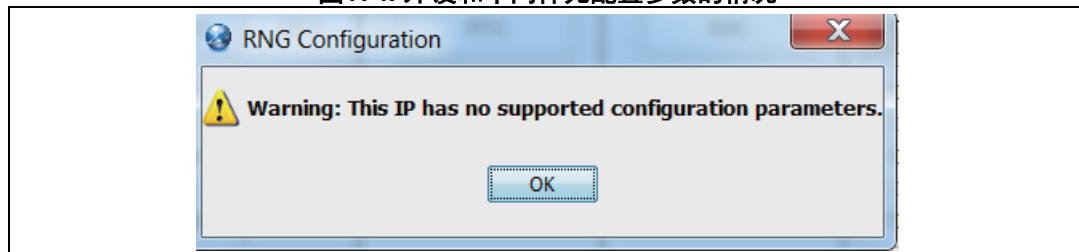
图173. 配置视图



7.6.2 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174. 外设和中间件无配置参数的情况



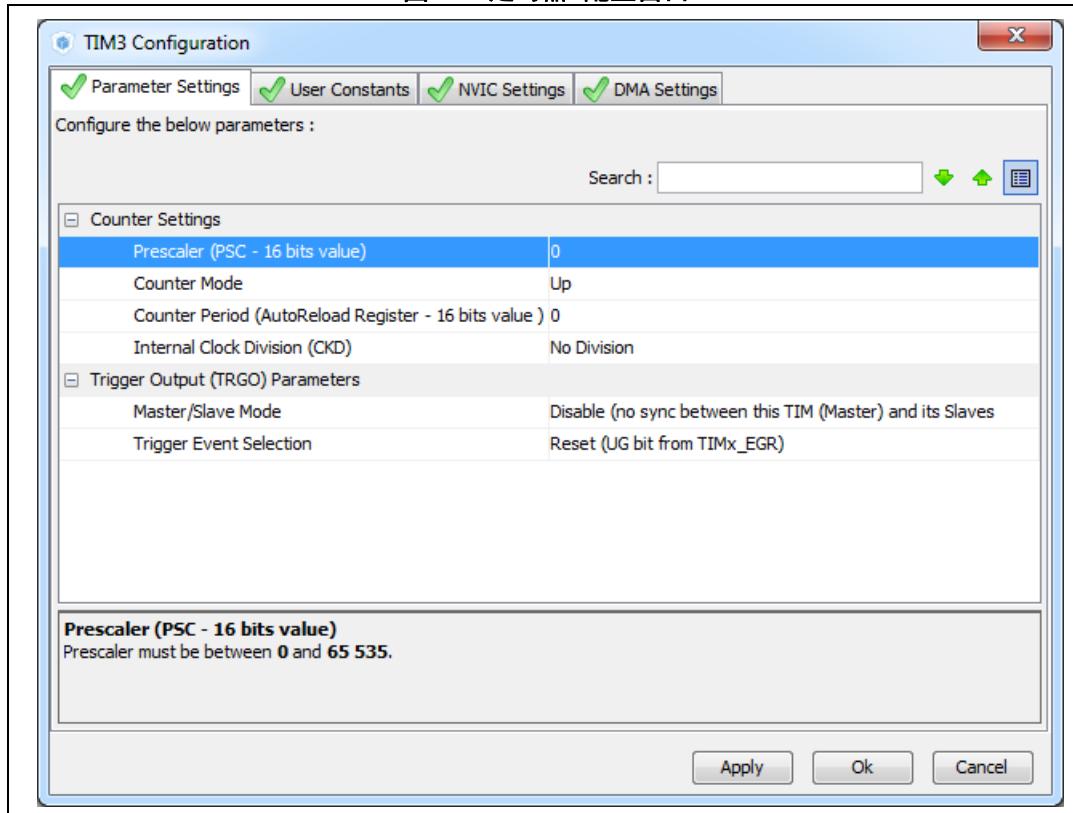
按照以下步骤继续进行外围设备配置：

1. 单击外设按钮以打开相应的配置窗口。

在我们的示例中，

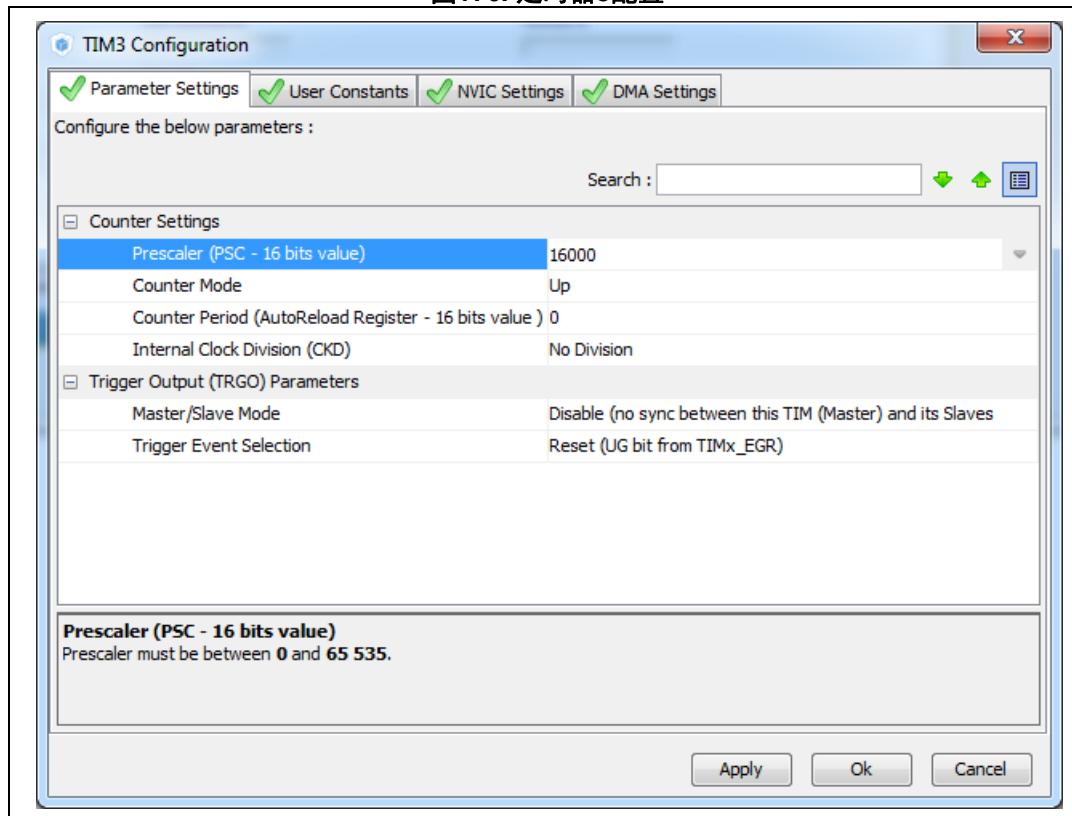
- a) 点击TIM3以打开定时器配置窗口。

图175. 定时器3配置窗口



- b) 通过16MHz 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176. 定时器3配置

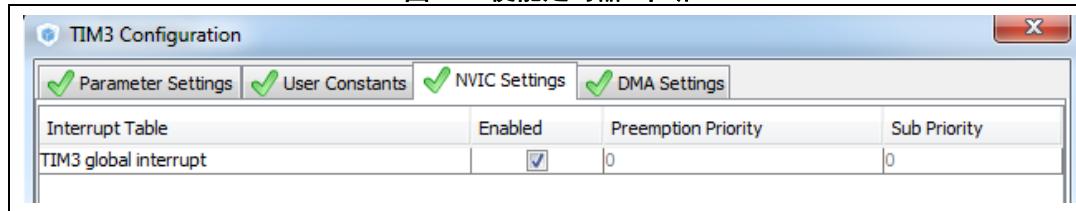


2. 或者，如果可用，选择：

- **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 177 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。

3. 修改并单击“应用”或“确定”以保存修改。

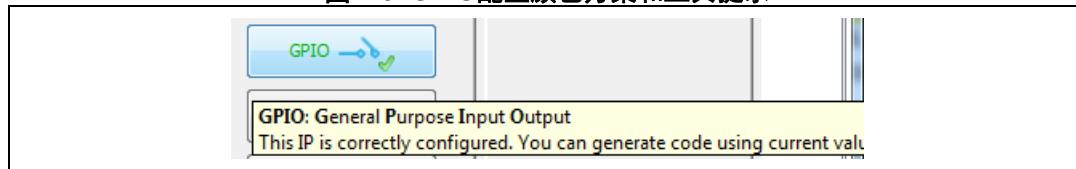
图177. 使能定时器3中断



7.6.3 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。

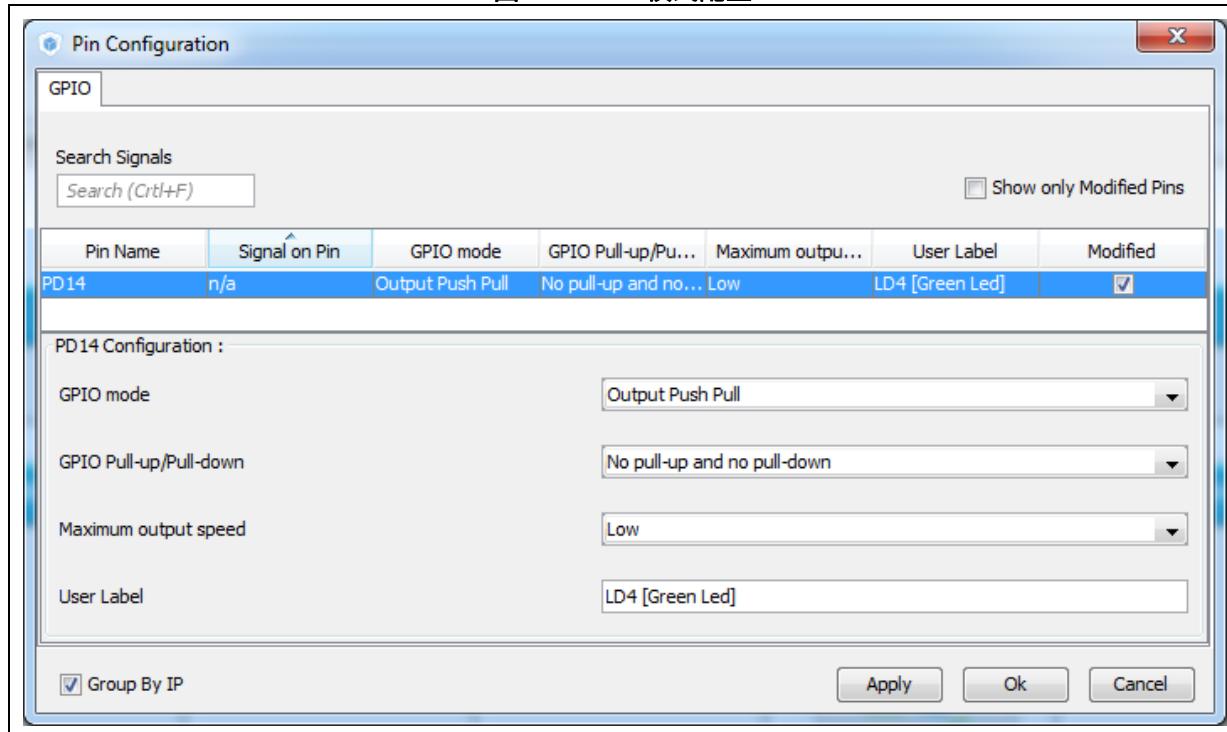
图178. GPIO配置颜色方案和工具提示



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 179）。

图179. GPIO模式配置



3. 点击应用，然后点击确定关闭窗口。

7.6.4 配置DMA

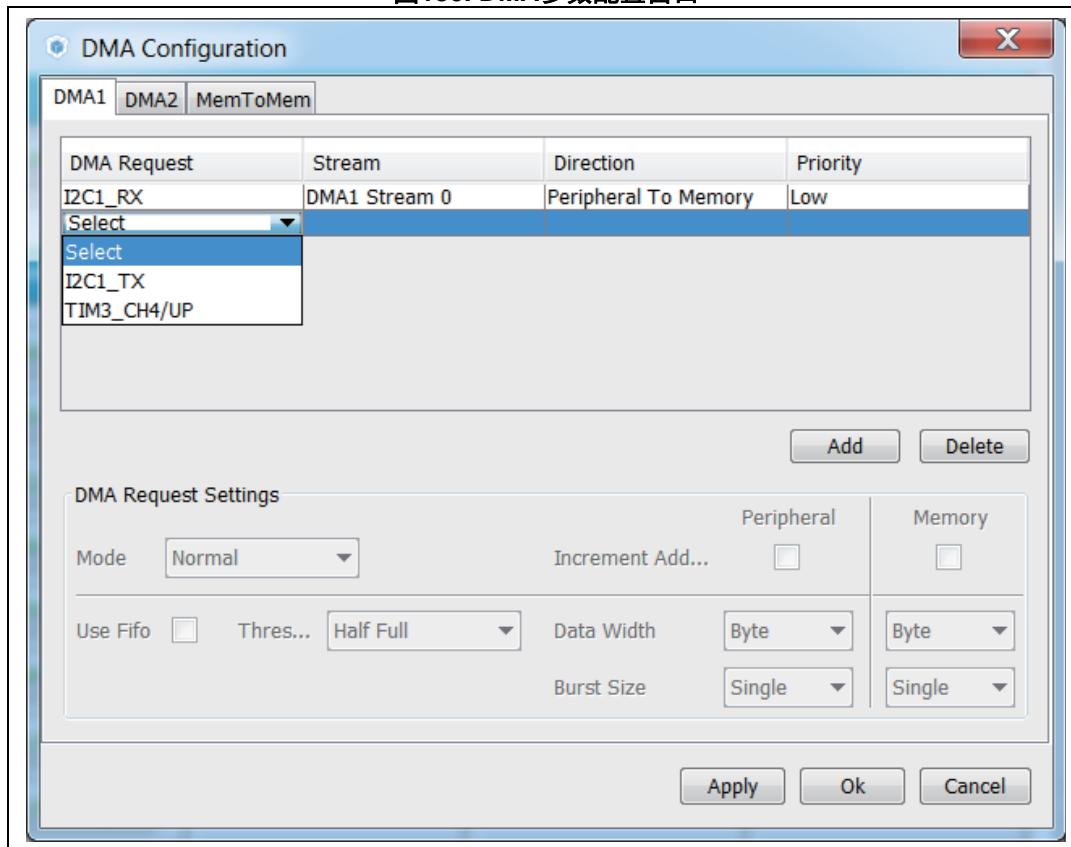
教程示例不需要配置中间件。

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 180）。

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：存储器到外设”或“外设到存储器”。
4. 选择优先级。

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180. DMA参数配置窗口

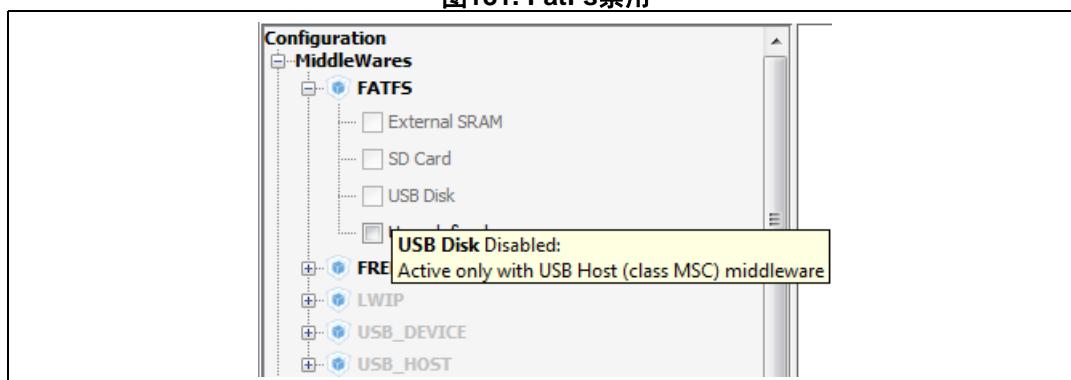


7.6.5 配置中间件

教程示例不需要配置中间件。

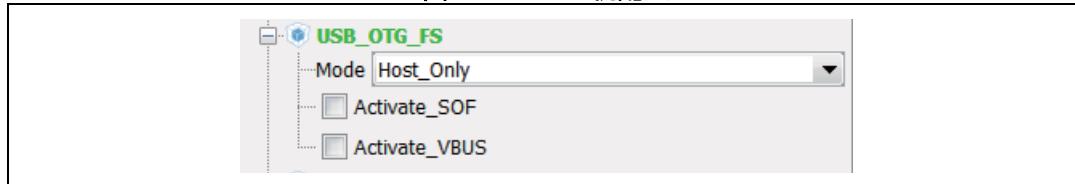
如果中间件模式需要使用外设，则必须在引脚布局视图中配置外设才能提供中间件模式。工具提示可引导用户完成操作，如下面的FatFs示例所示：

图181. FatFs禁用



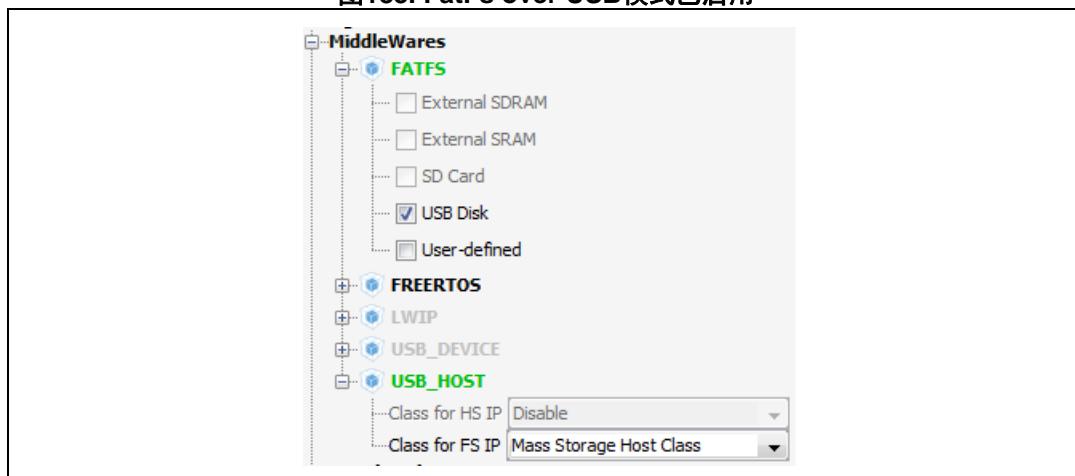
1. 通过引脚布局视图配置USB外设。

图182. USB主机配置



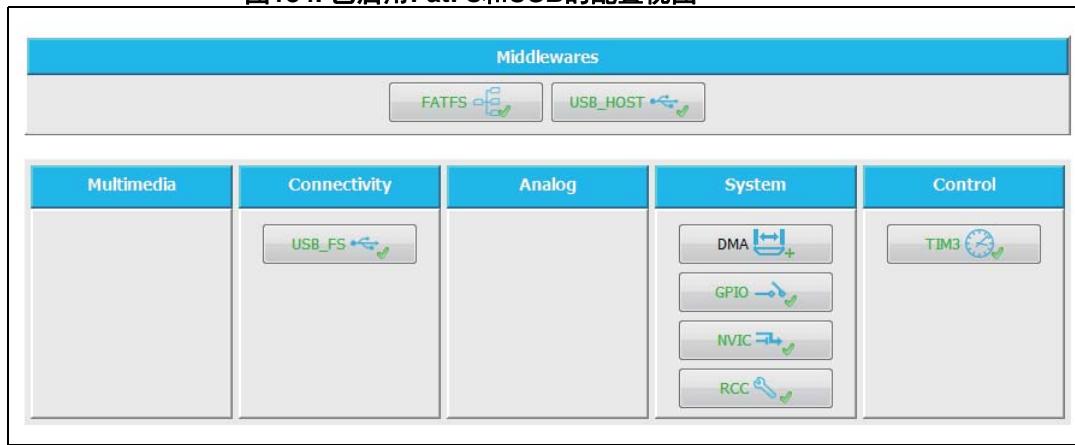
2. 从USB主机中间件选择MSC_FS类。
3. 选中此复选框在树形面板中启用FatFs USB模式。

图183. FatFs over USB模式已启用



4. 选择配置视图。会显示FatFs和USB按钮。

图184. 已启用FatFs和USB的配置视图



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185. FatFs外设实例

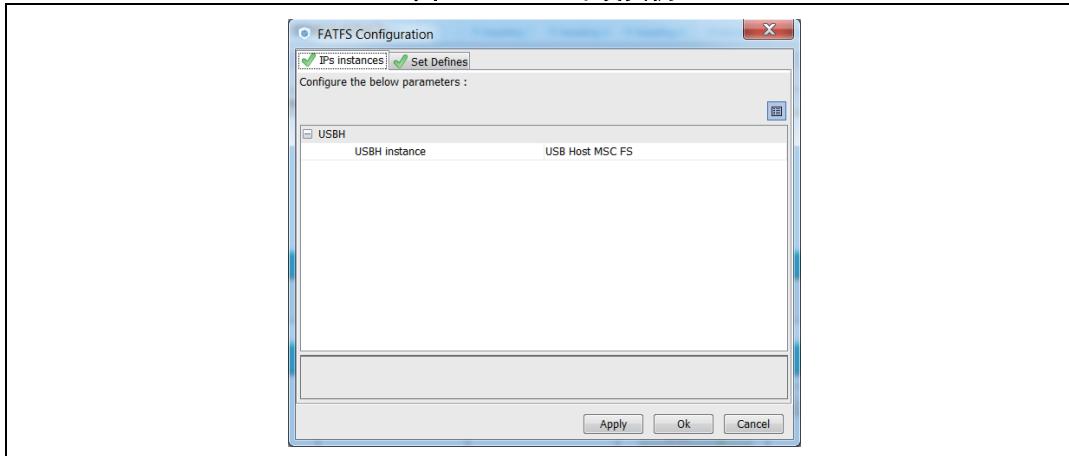
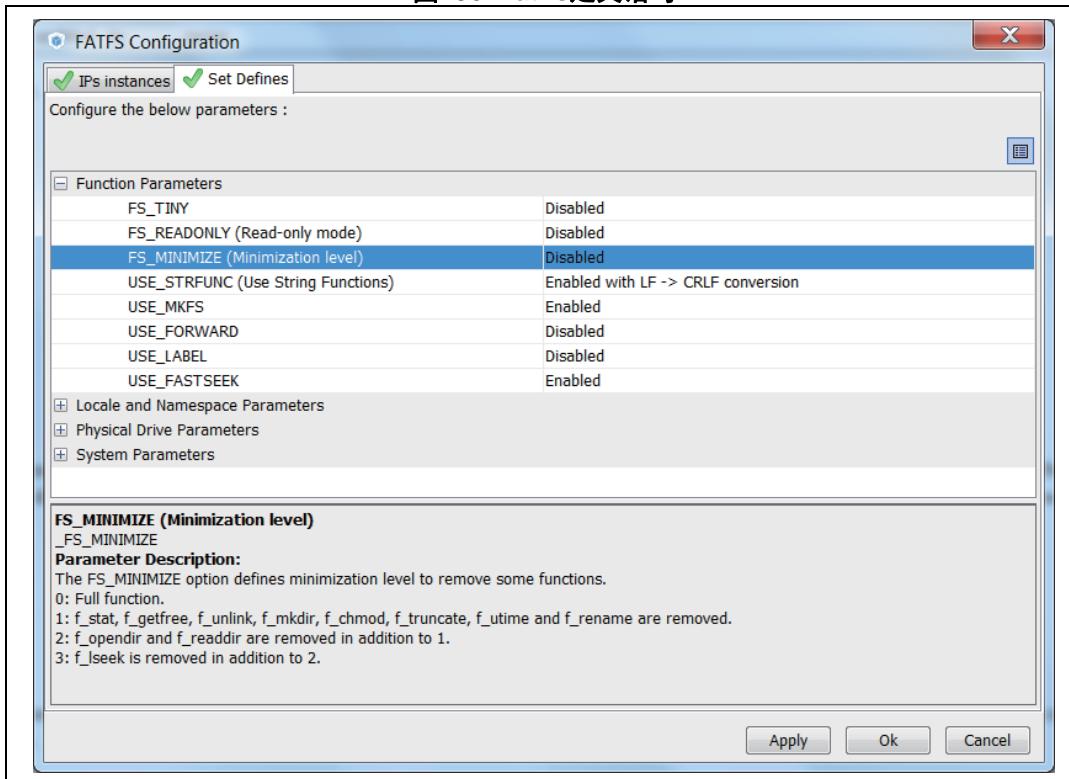


图186. FatFs定义语句



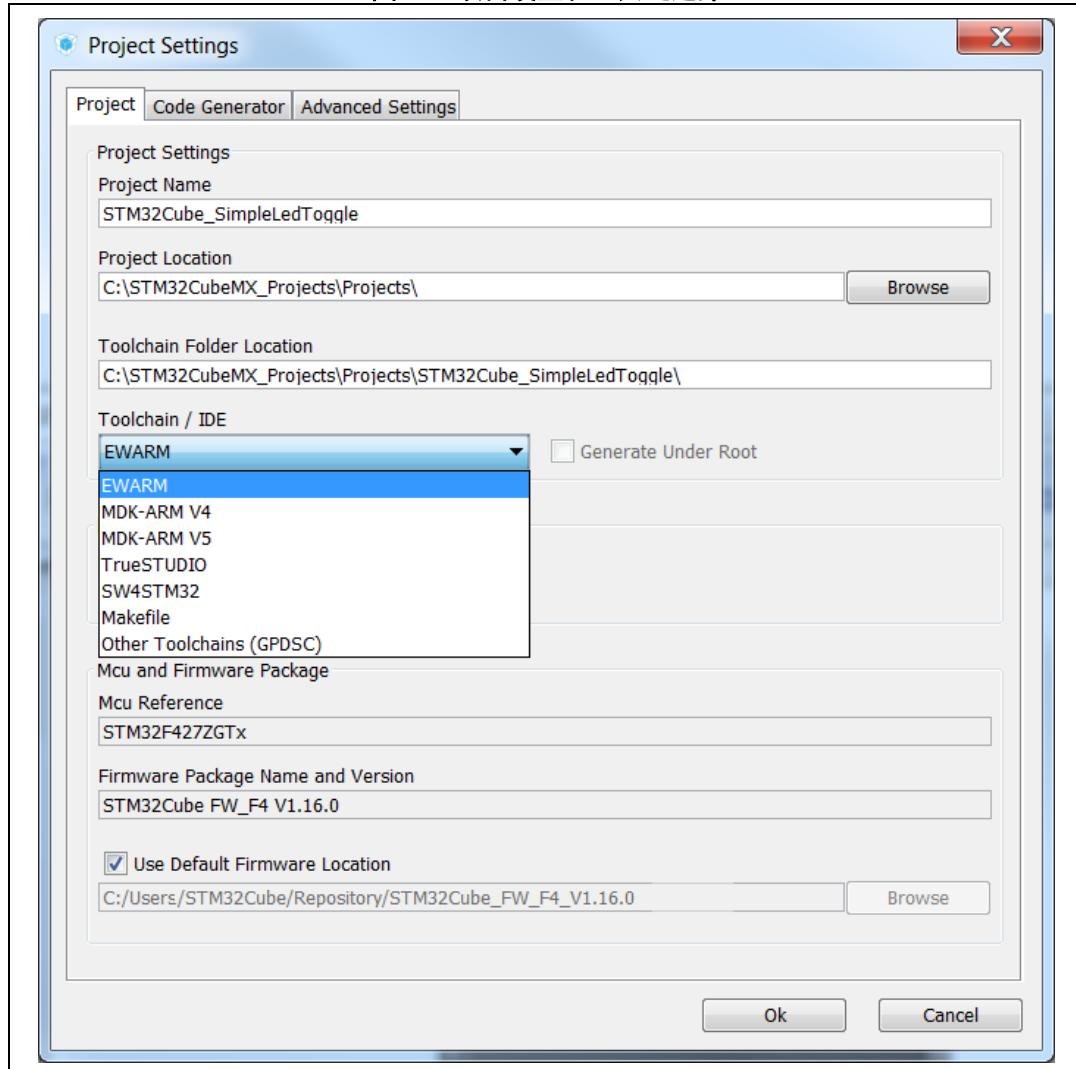
7.7 生成完整的C项目

7.7.1 设置项目选项

在生成C代码之前，可按照图 187中的说明调整默认项目设置。

1. 从项目菜单中选择设置，打开“项目设置”窗口。
2. 选择项目选项卡，并选择项目名称、位置和工具链，以生成项目（参见图 187）。

图187. 项目设置和工具链选择



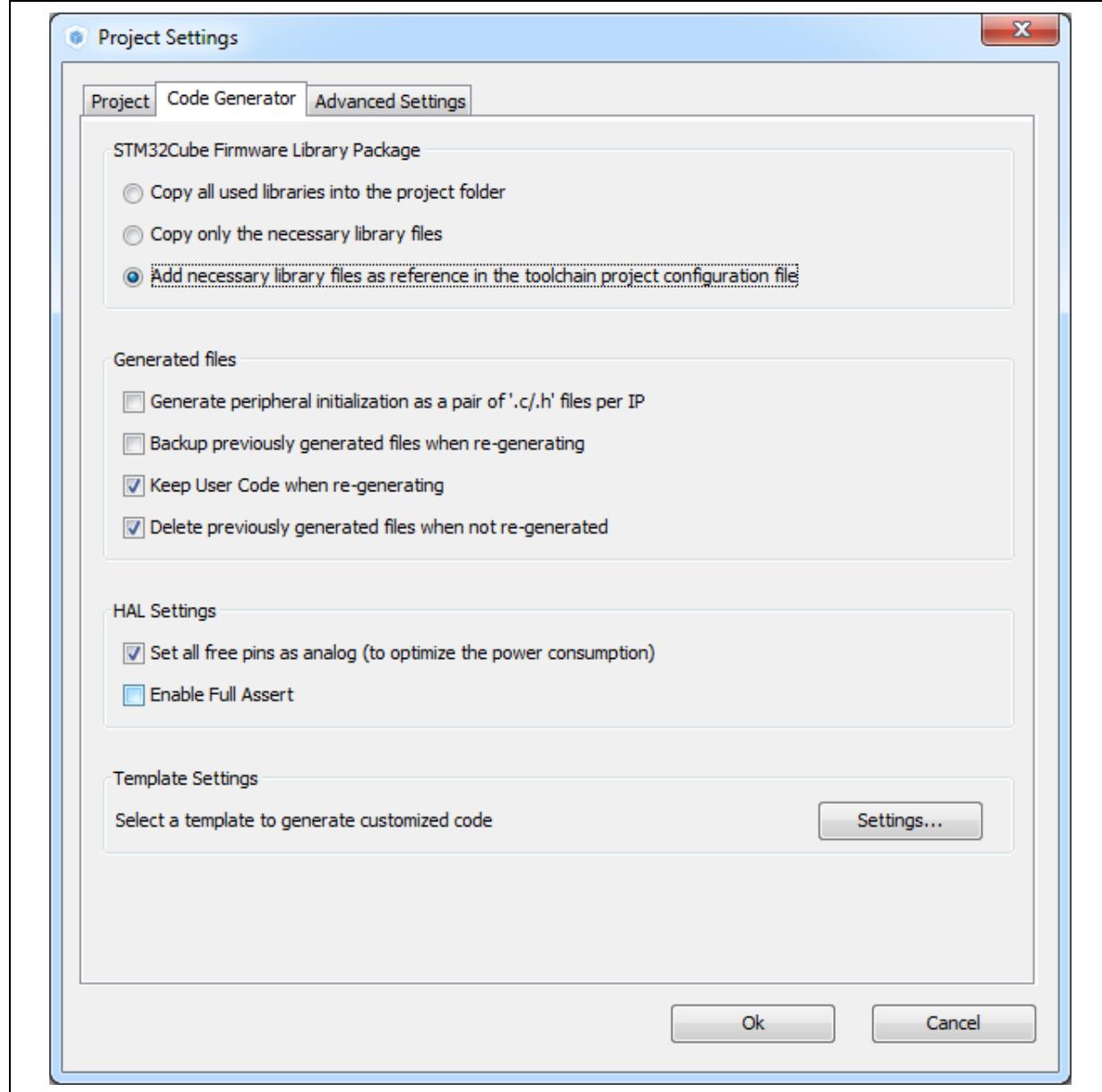
3. 选择代码生成器选项卡，以选择各种C代码生成选项：

- 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）。

在教程示例中，按下图所示选择设置，并点击“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188. “项目设置”菜单 -“代码生成器”选项卡

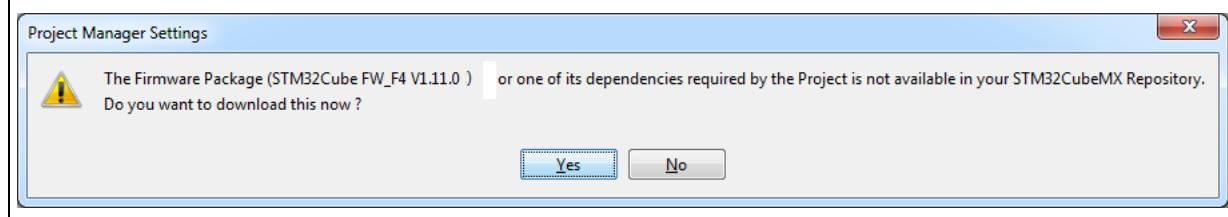


7.7.2 下载固件包和生成C代码

1. 点击生成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189. 缺少固件包警告消息

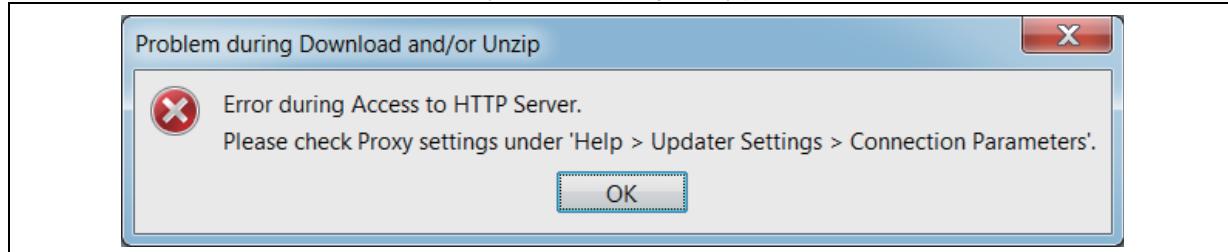


2. STM32CubeMX可供下载相关固件包或继续进行操作。点击[下载](#)获取完整项目，即准备好用于所选IDE的项目。

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，会显示以下错误消息：

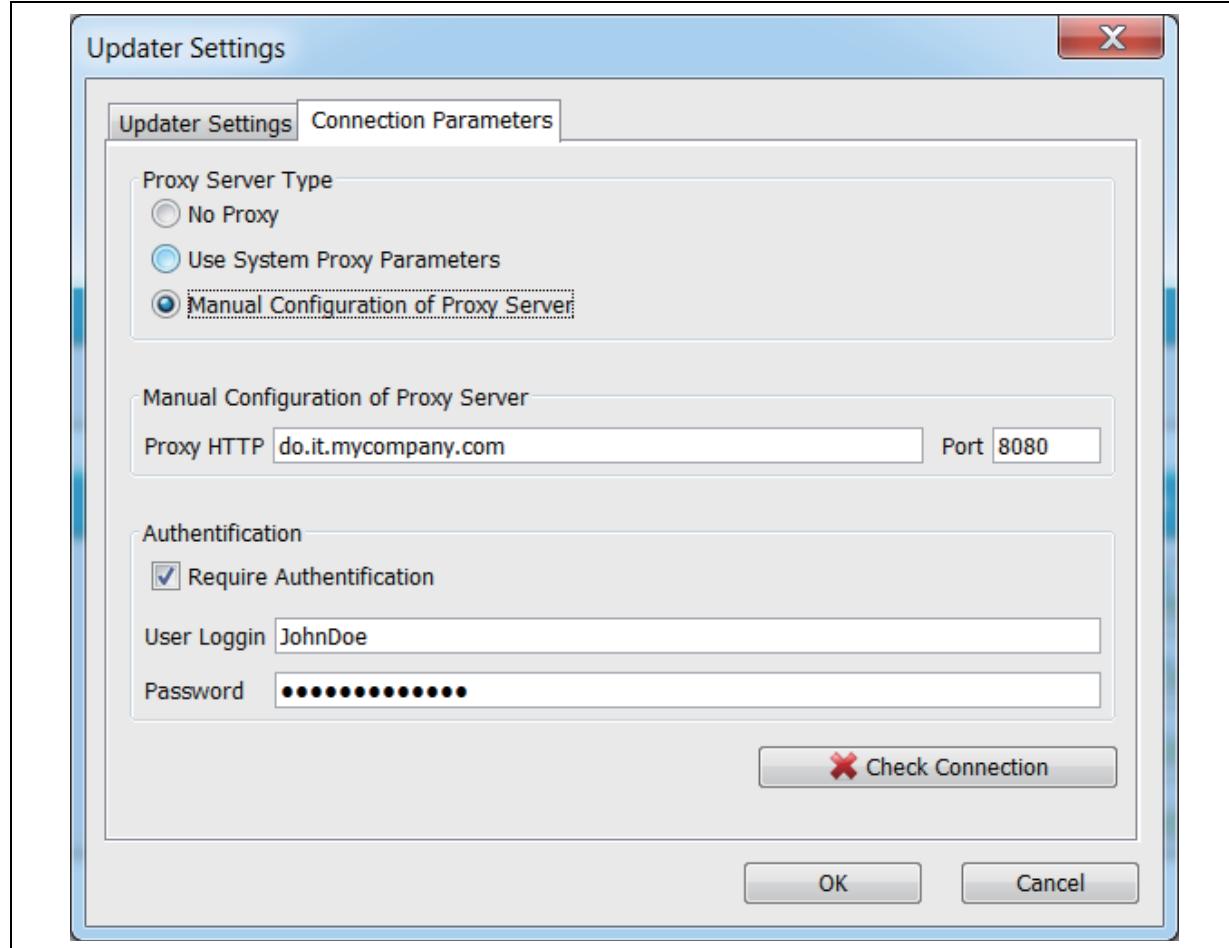
图190. 下载过程中出错



要解决此问题，请执行下面的两步操作。否则请跳过。

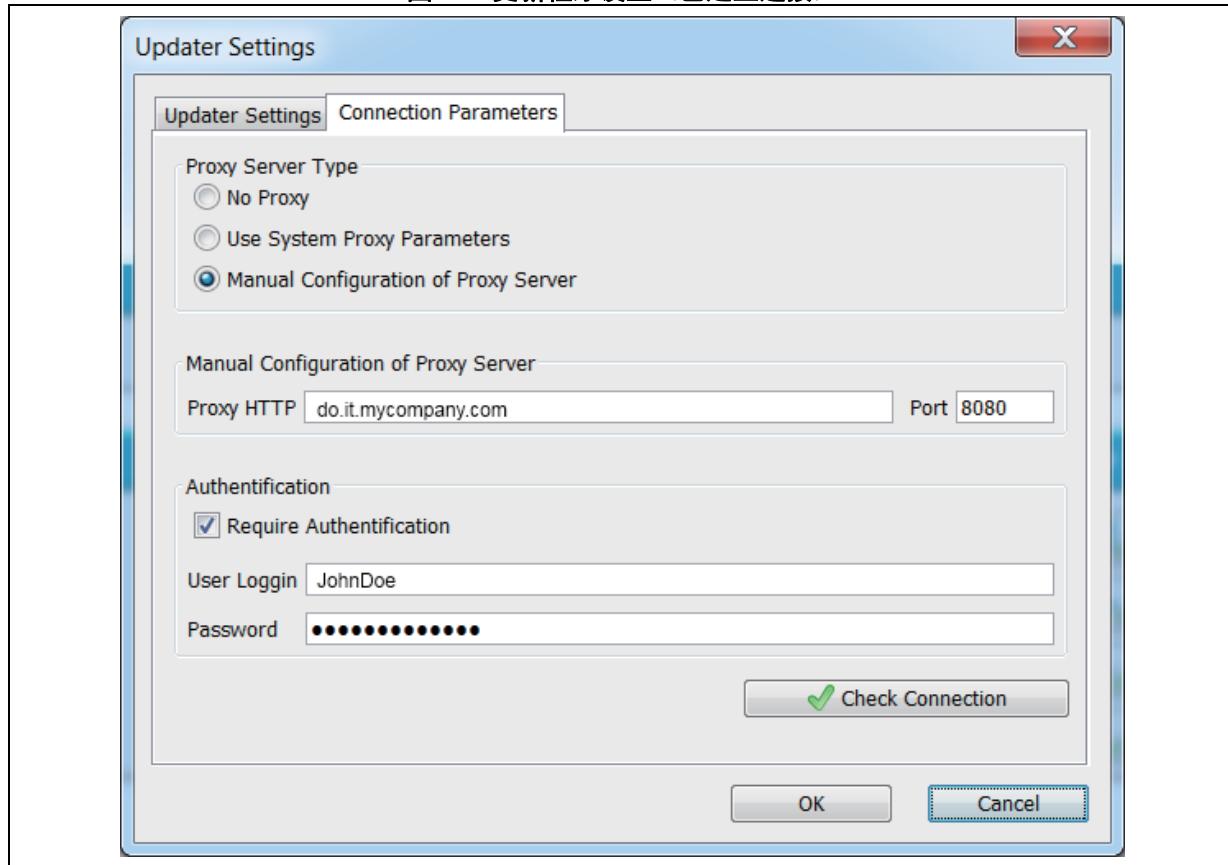
3. 选择帮助 > 更新程序设置菜单，并调整连接参数，使其与您的网络配置相匹配。

图191. 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192. 更新程序设置（已建立连接）



5. 连接建立后，点击生成C代码。C代码生成过程开始，进度显示如下图所示。

图193. 下载固件包

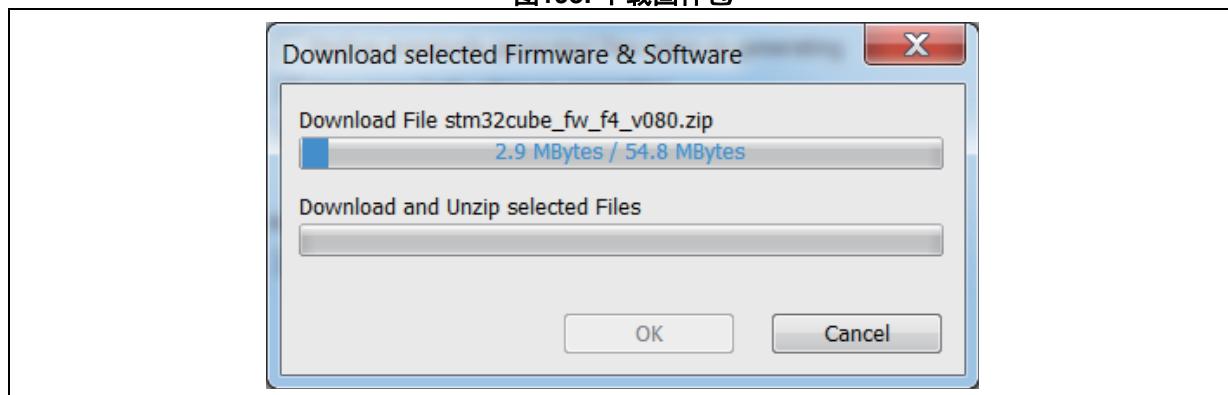
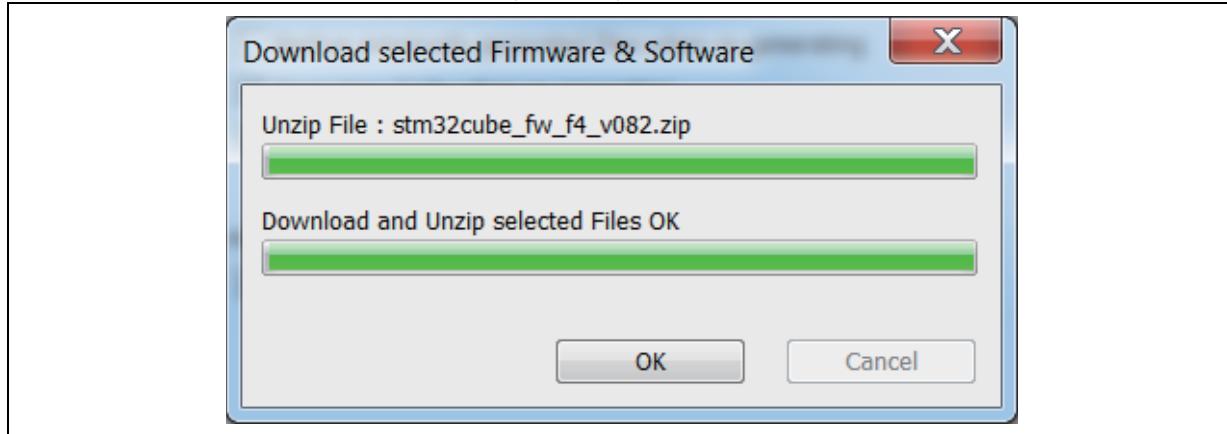
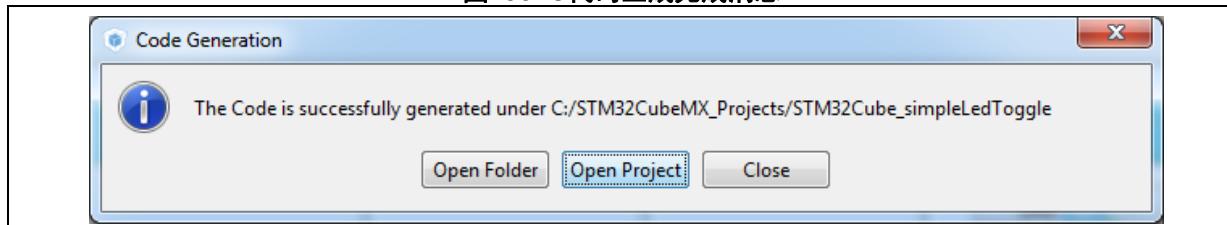


图194. 解压固件包



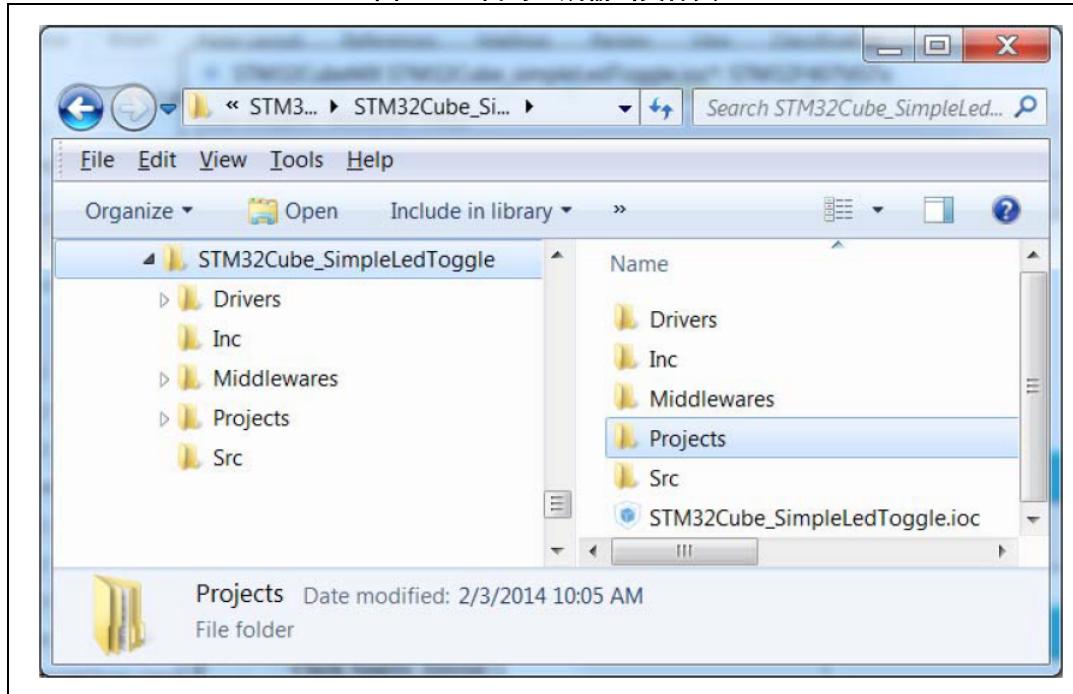
6. 最后会显示确认消息，指示C代码已成功生成。

图195. 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 7.8 节中的操作。

图196. 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。

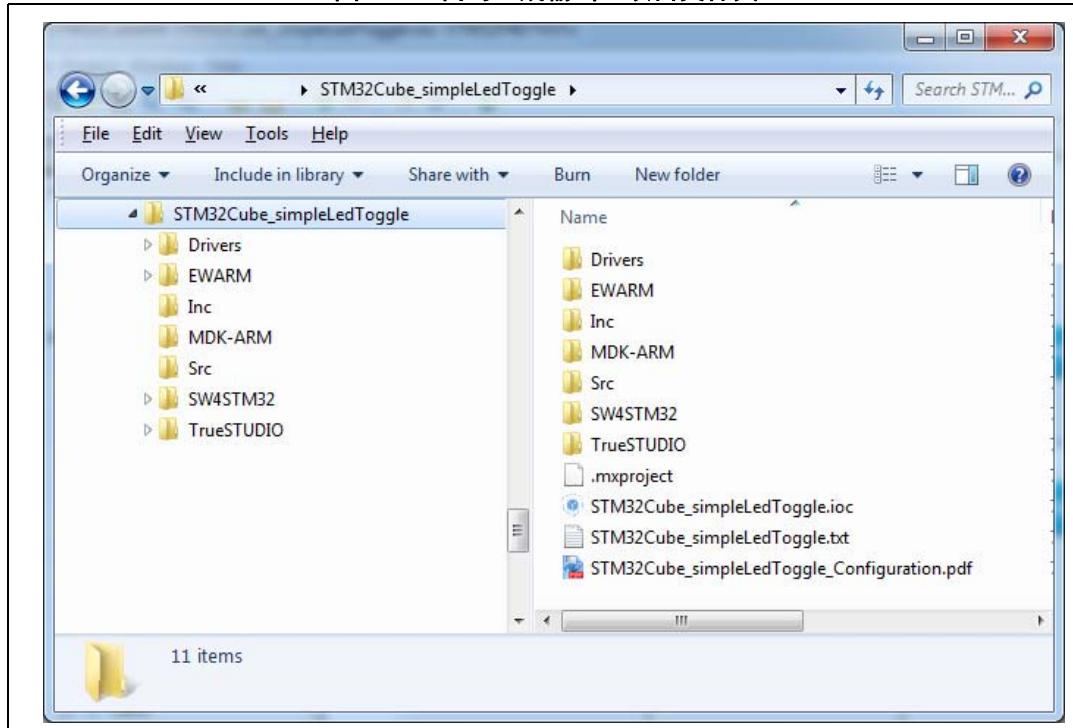
7.8 构建和更新C代码项目

本例介绍了如何在IAR™WARM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“项目设置”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。

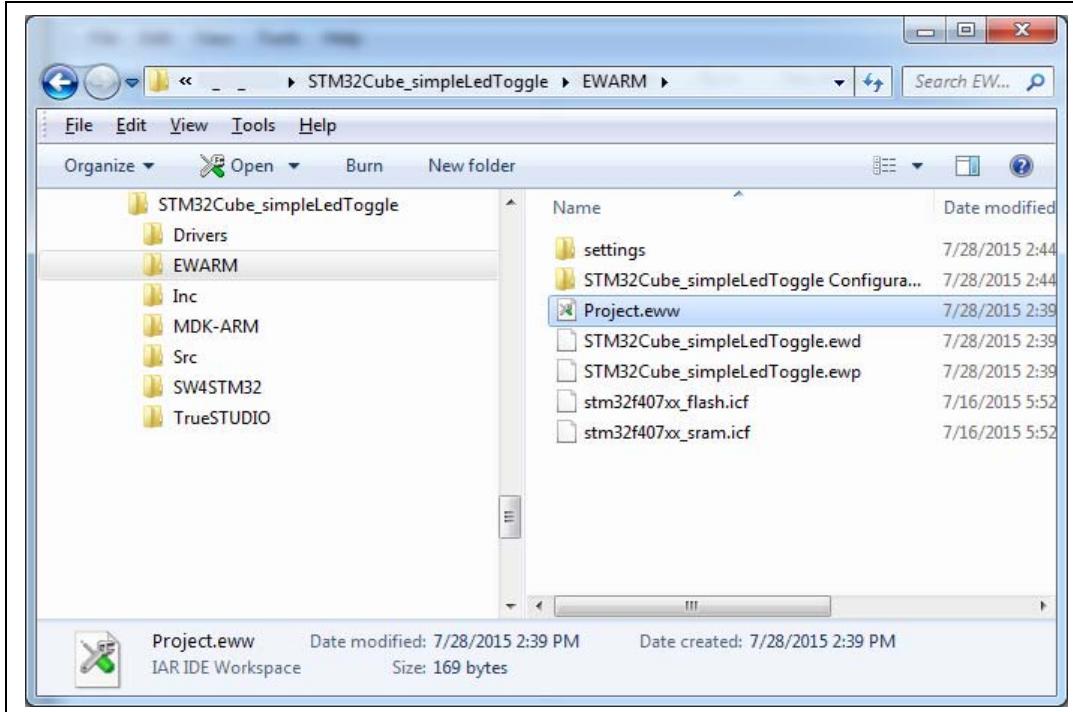
1. 点击对话框窗口中的打开项目，或双击STM32CubeMX下的工具链文件夹中提供的相关IDE文件，直接在IDE工具链中打开项目（参见图 195）。

图197. C代码生成输出：项目文件夹



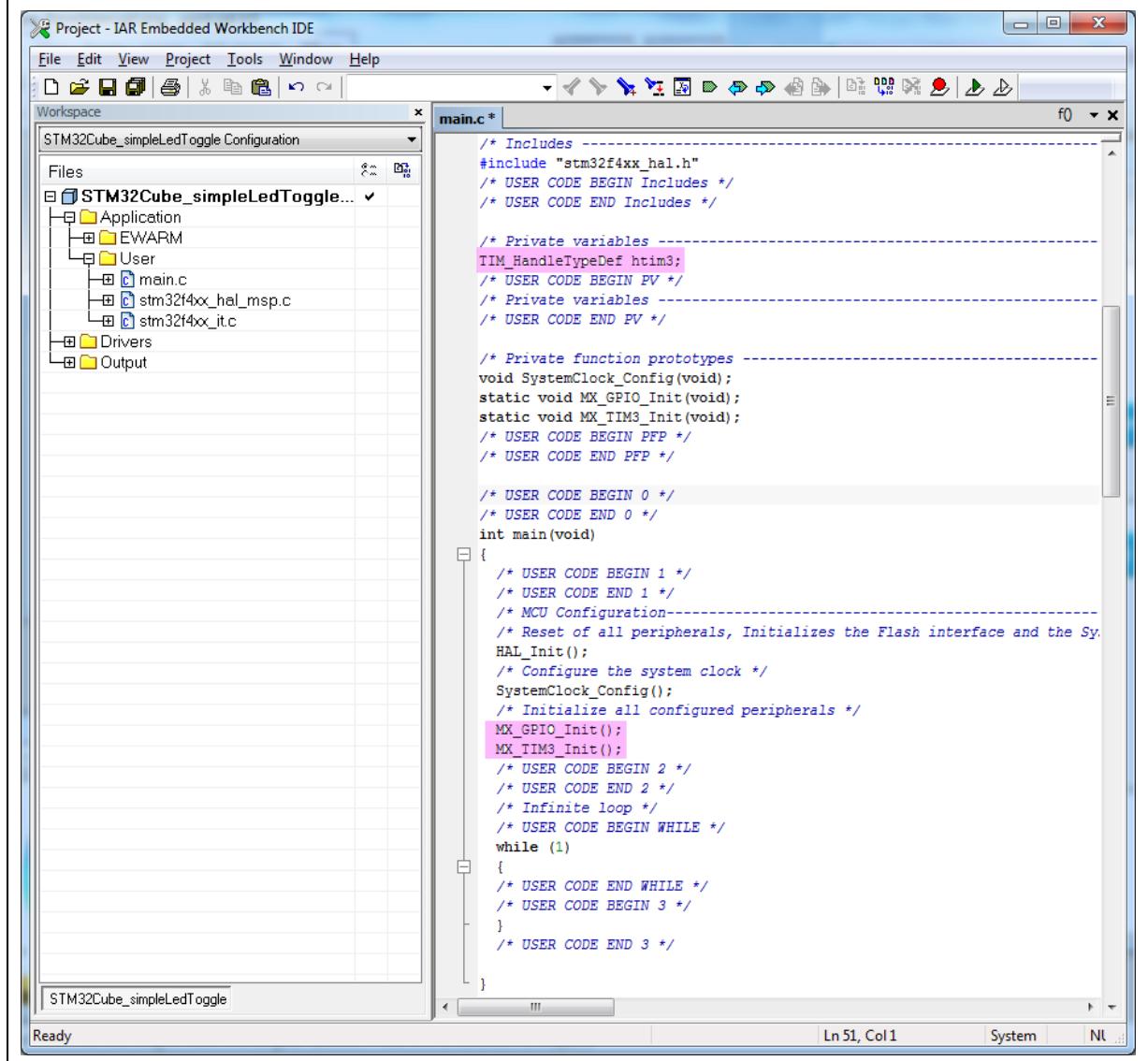
2. 例如，选择.eww文件可加载IAR™ EWARM IDE中的项目。

图198. 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。

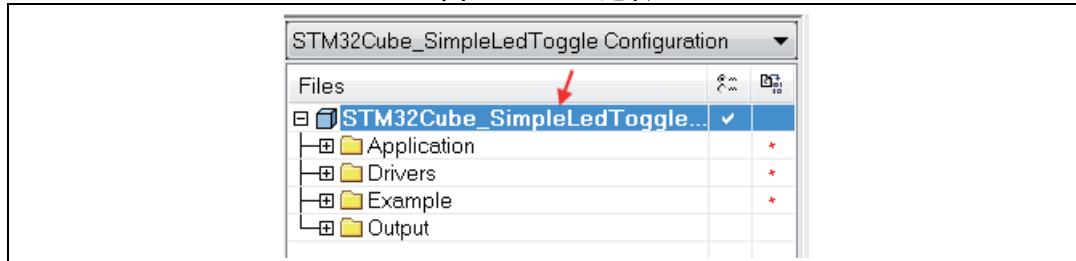
图199. 在IAR™ IDE中打开STM32CubeMX生成的项目



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。

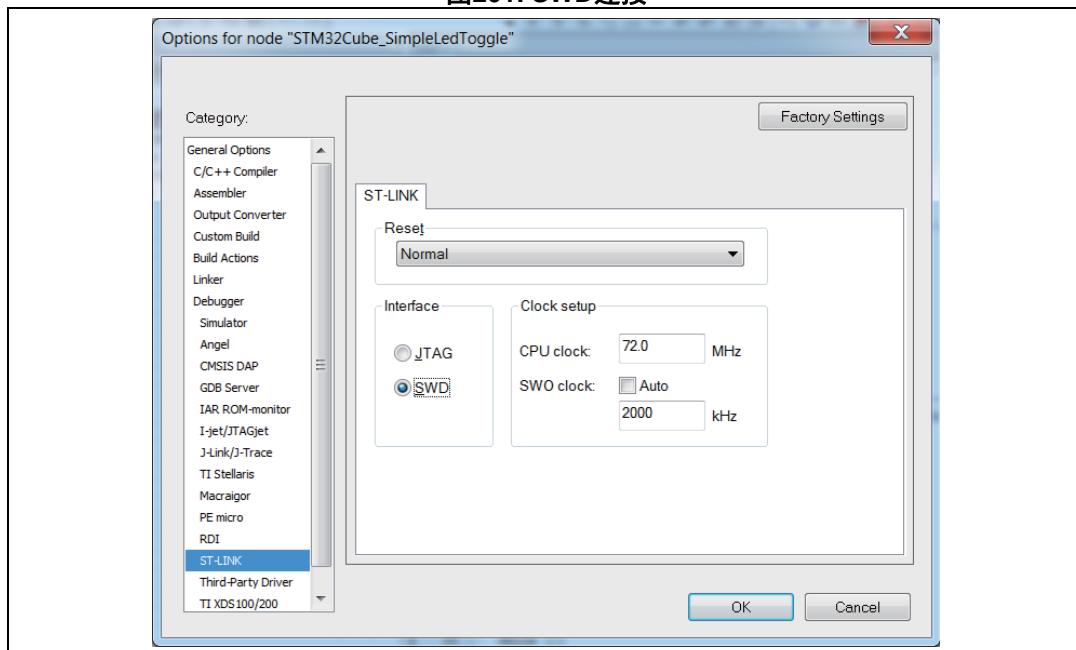
4. 在IAR™ IDE中，右键单击项目名称并选择选项。

图200. 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201. SWD连接



6. 选择项目>全部重建。检查项目创建是否成功。

图202. 项目创建日志

The screenshot shows the 'Messages' tab of a project log window. It lists several source files: 'stm32f4xx_hal_tim.c', 'stm32f4xx_hal_tim_ex.c', 'stm32f4xx_it.c', 'stm32f4xx_ll_sdmmc.c', and 'system_stm32f4xx.c'. Below this, the word 'Linking' is shown. At the bottom, it displays 'Total number of errors: 0' and 'Total number of warnings: 0'.

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203. 用户部分2

The screenshot shows a portion of the main.c file. It includes standard initialization code like HAL_Init() and SystemClock_Config(). A specific section of code is highlighted in pink:

```
/* 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204. 用户部分4

```
/* USER CODE BEGIN 4 */

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    if ( htim->Instance == htim3.Instance )
    {
        HAL_GPIO_TogglePin(GPIOB, 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翻转驱动绿色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。如果已启用“项目设置”中的 Keep Current Signals Placement 选项，项目将更新，保留用户部分中的C代码。

7.9 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 205）。
4. 点击尝试导入按钮并检查导入状态，以确认导入是否成功（参见图 206）。
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. 选择项目 > 设置配置新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205. “导入项目”菜单

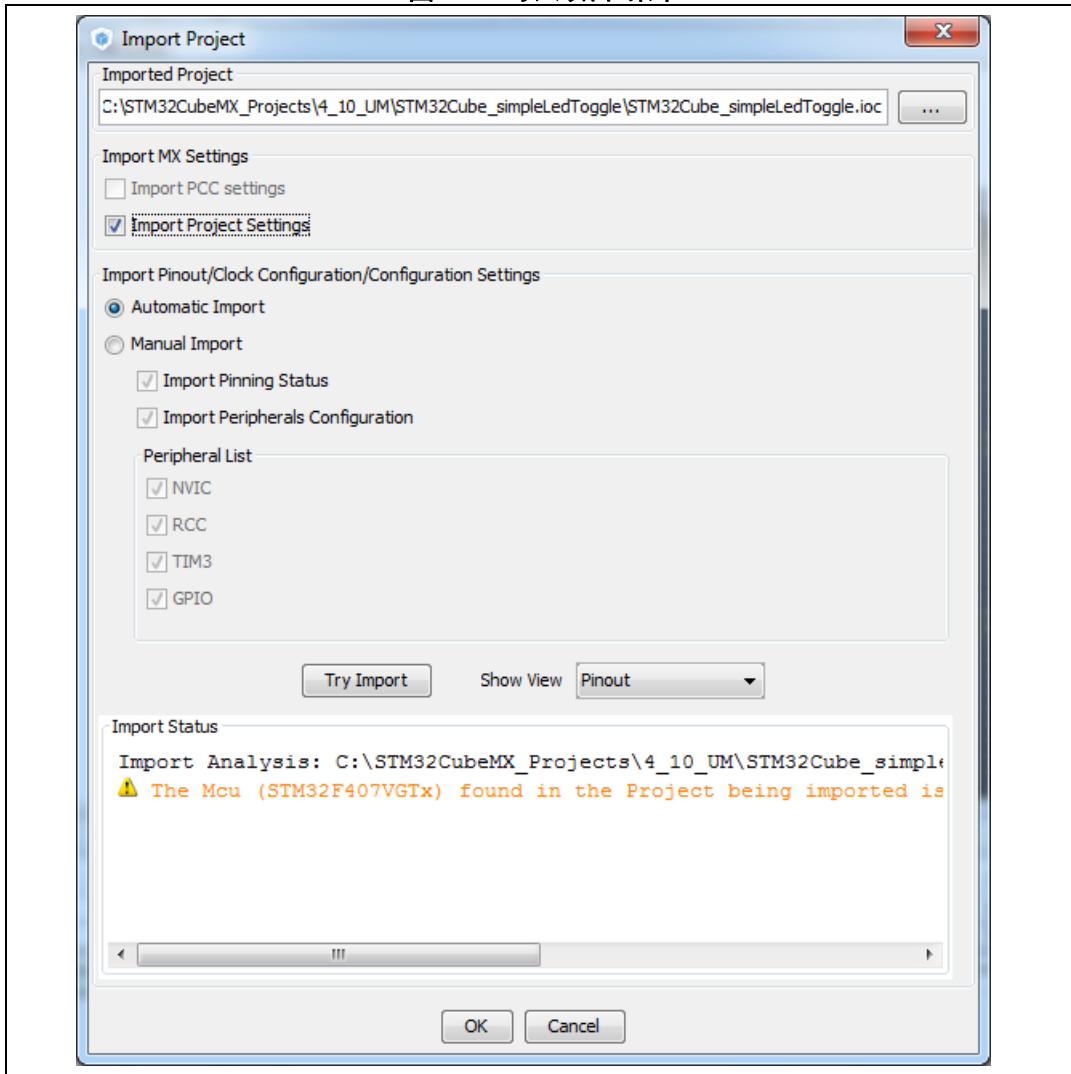
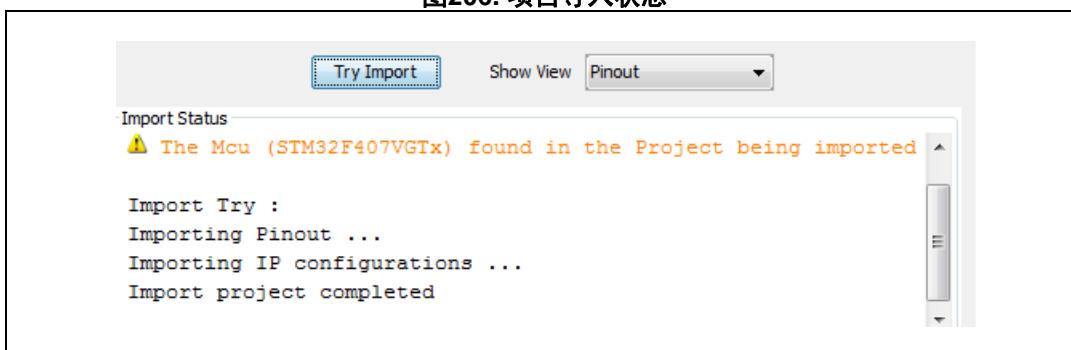


图206. 项目导入状态



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

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. 使选项将所有外设初始化为其默认模式保持未选中状态，以便仅为应用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 207）。载入引脚布局视图，该视图与评估板上的MCU引脚布局配置匹配（参见图 208）。

图207. 板子外设初始化对话框

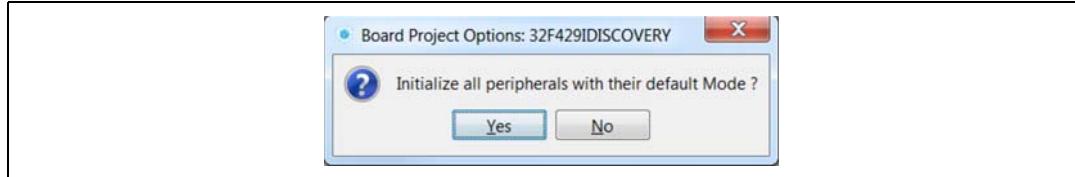
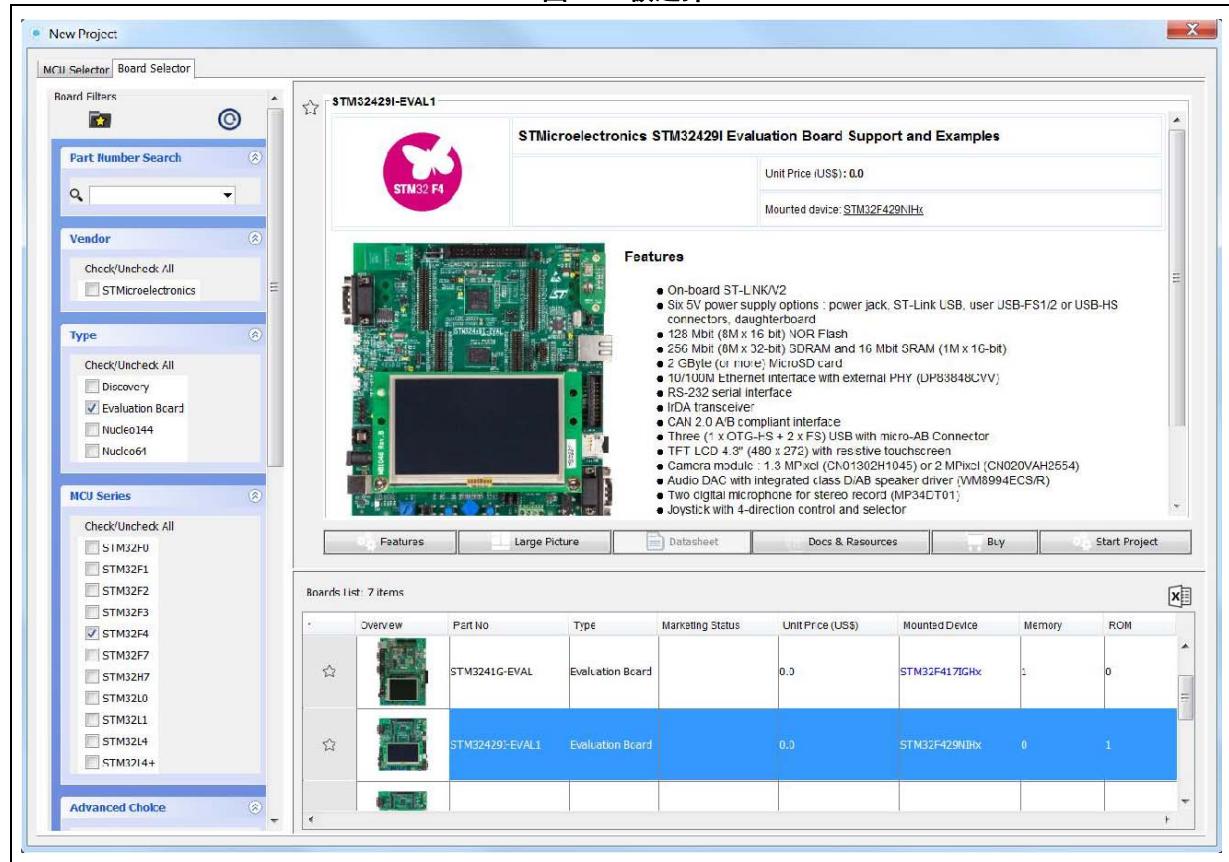
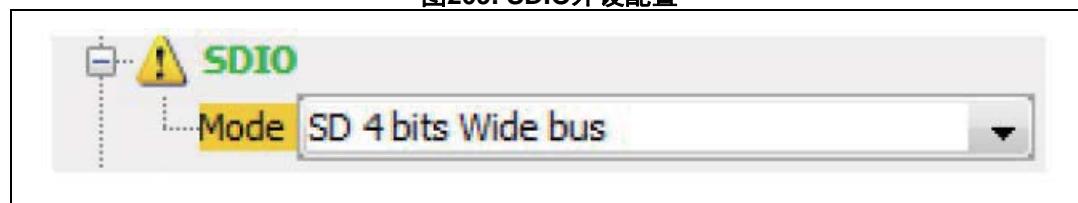


图208. 板选择



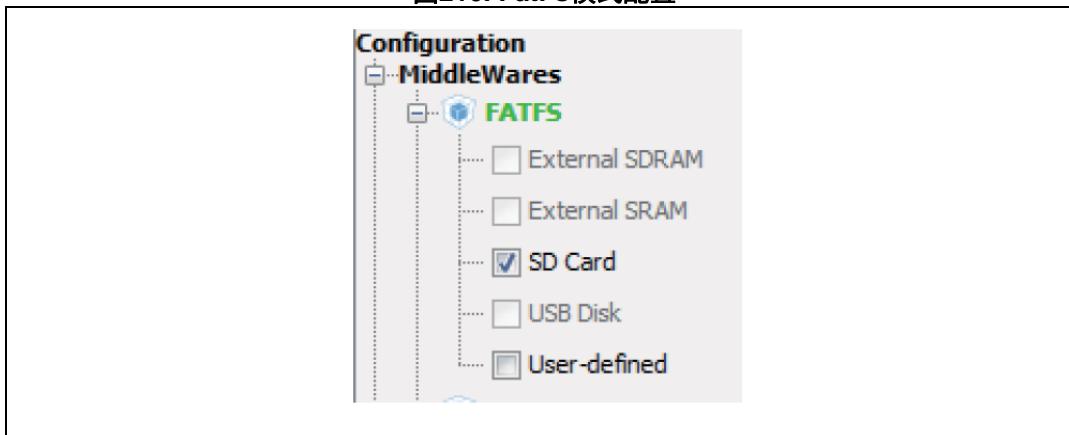
7. 在左侧的“外设”树中，展开SDIO外设并选择SD 4位宽总线（参见图 209）。

图209. SDIO外设配置



8. 在“中间件”类别下，FatFs模式选择“SD卡”（参见图 210）。

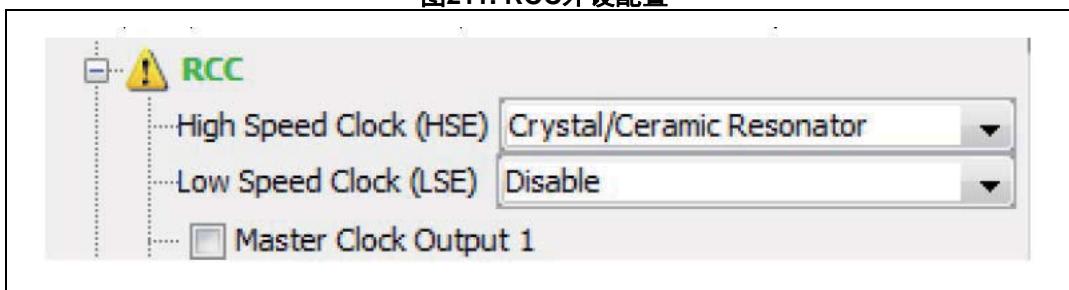
图210. FatFs模式配置



9. 按以下方法配置时钟：

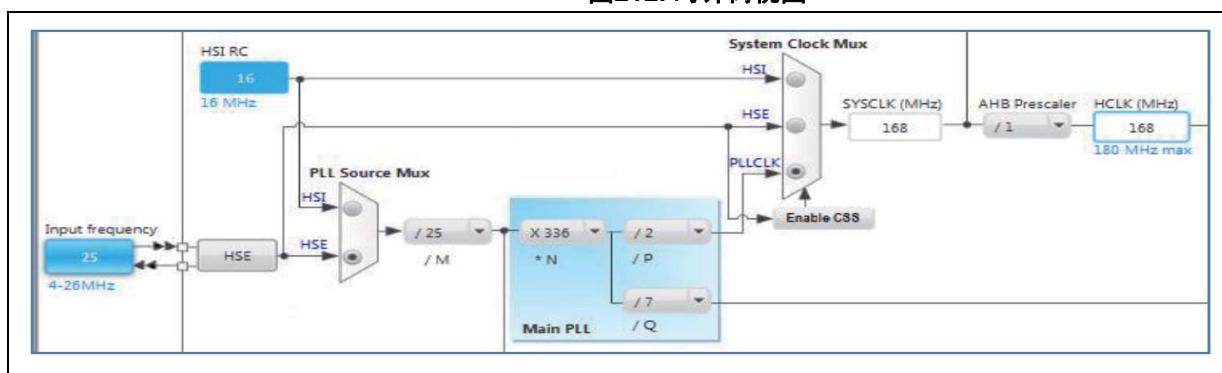
a) 从引脚布局视图中选择RCC外设（参见图 211）。

图211. RCC外设配置



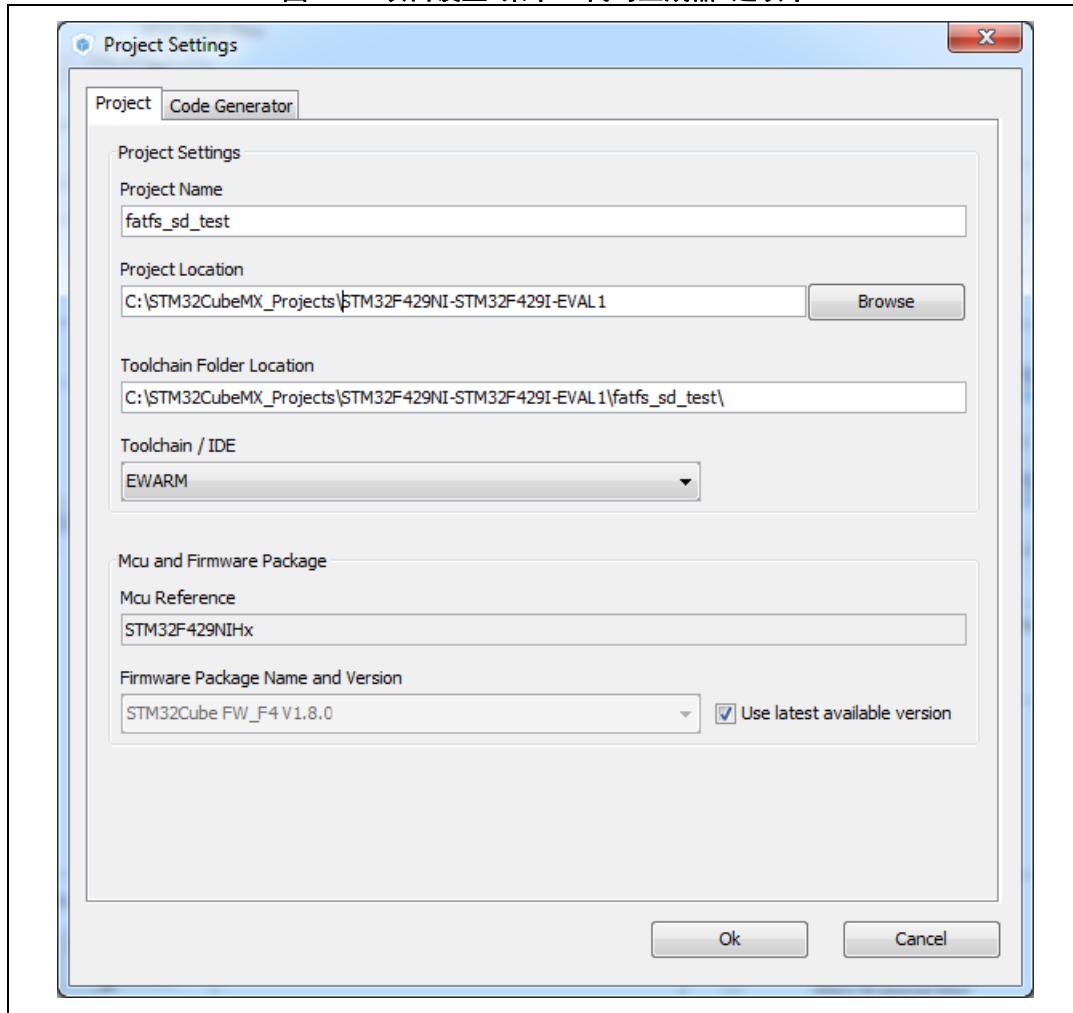
b) 在时钟选项卡中配置时钟树（参见图 212）。

图212. 时钟树视图



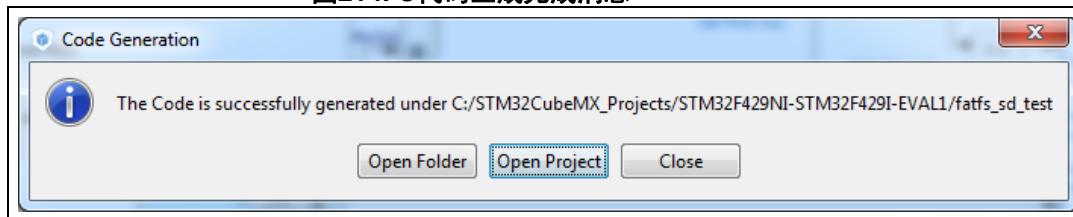
10. 在项目设置菜单中，指定项目名称和目标文件夹。然后选择EWARM IDE工具链。

图213. “项目设置”菜单 -“代码生成器”选项卡



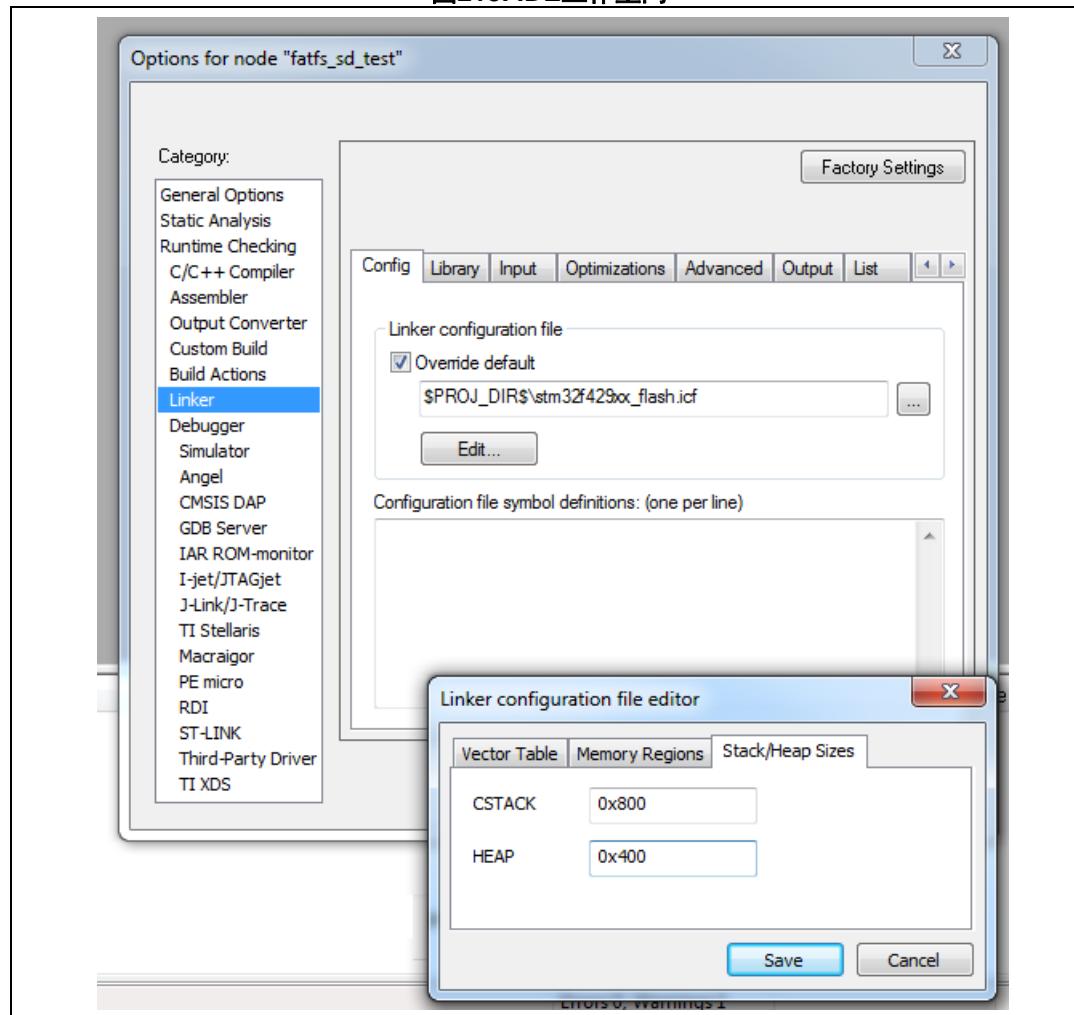
11. 点击确定。然后在工具栏菜单中点击 生成项目。
12. 代码生成完毕后，点击代码生成对话框窗口中的打开项目（参见图 214）。此操作会直接在IDE中打开项目。

图214. 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。调整堆和堆栈大小（参见图 215）。

图215. 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,  
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x29,0x74,  
0x51,0x0e,0x63,0x30,0x04,0x4c,0x1d,0x0f,0x56,0x25,0x11,0x79,0x41,0x1f,  
0x85,0x6f,0x25,0x79,0x7e,0x27,0x72,0x72,0xb0,0x50,0x43,0x00,0x44,0x15,  
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) 插入主函数局部变量：

```
int main(void)  
{  
  
/* 用户代码开始 1 */  
HRESULT 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- 将文件系统对象注册到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)
```

9 教程 3 - 使用功耗计算器优化嵌入式应用功耗等

9.1 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程序的影响。

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跳变次数
- 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Flash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。

9.2 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，然后逐步引入STM32L476RG微控制器提供的低功耗特性。

9.3 使用功耗计算器

9.3.1 创建功耗系列

请按照以下步骤创建序列（参见[图 216](#)）：

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 217](#)）。
 - 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 217](#)）。

图216. 功耗计算示例

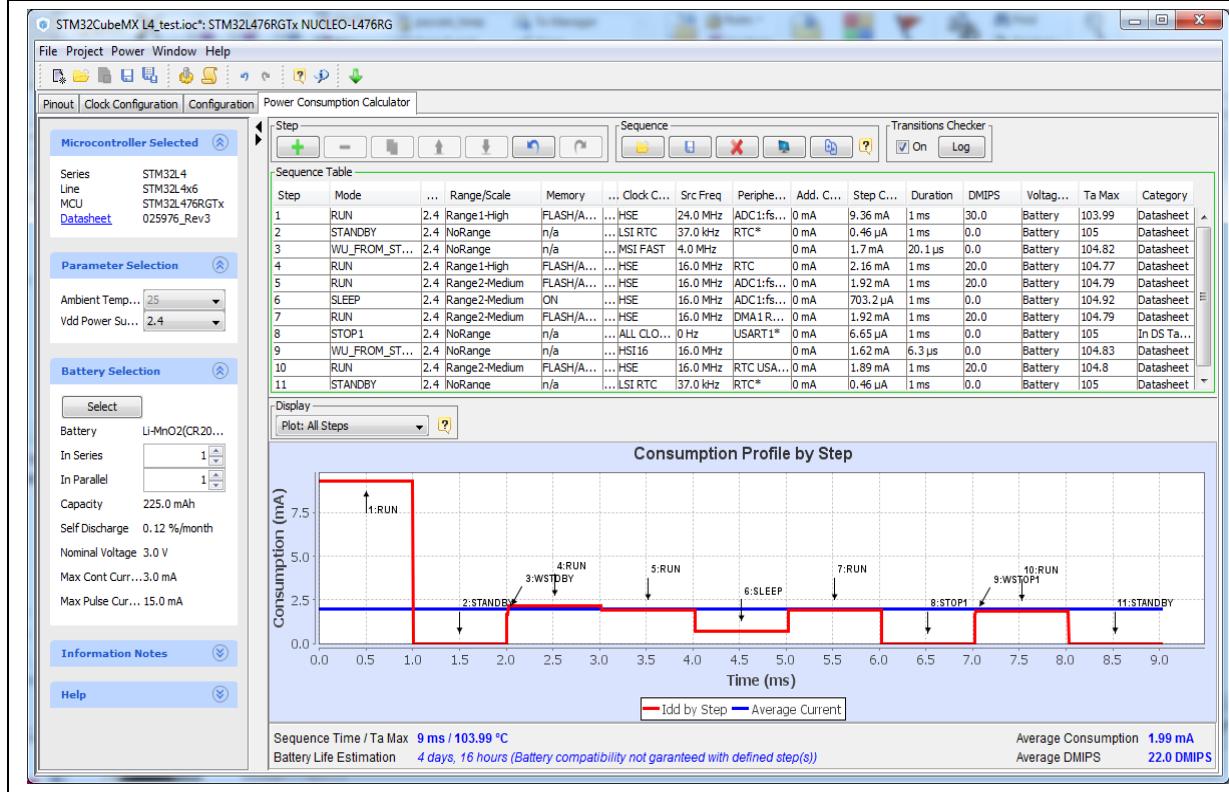
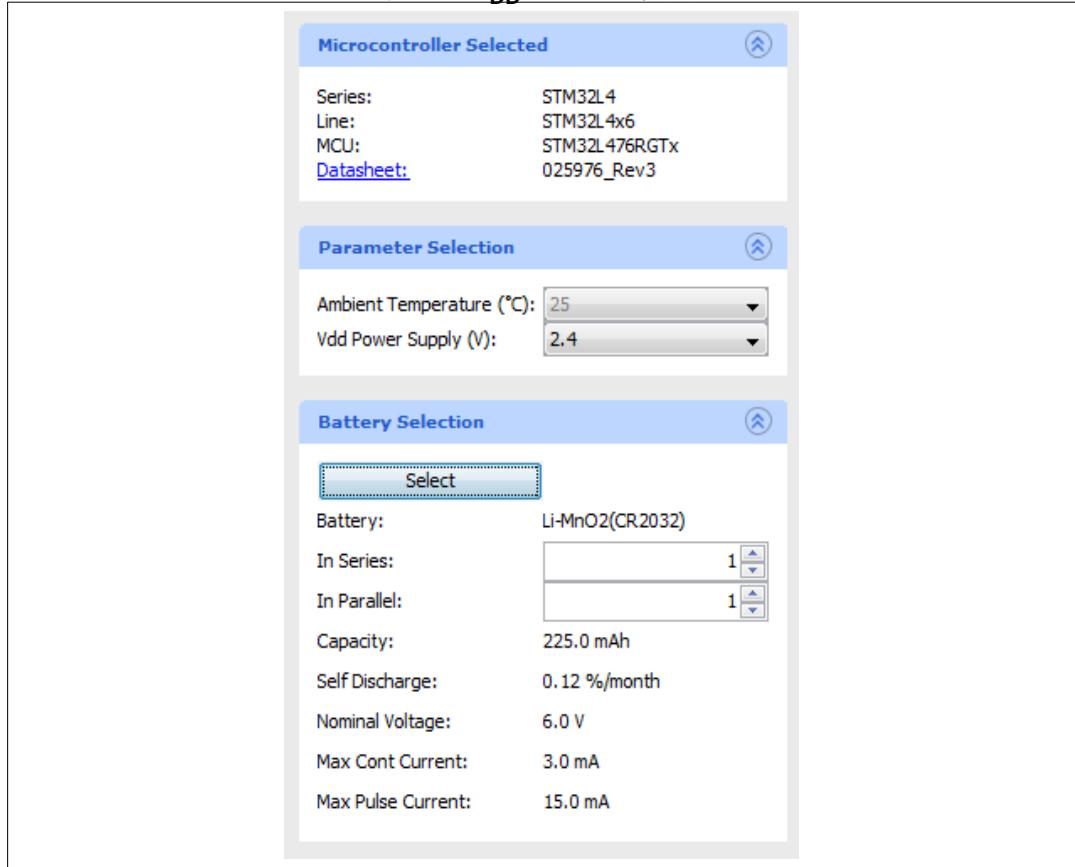


图217. V_{DD}和电池选择菜单

5. 使能跳变检查器确保序列有效（参见图 217）。此选项可用于验证序列是否符合允许在 STM32L476RG 中实现的跳变。
6. 点击添加按钮添加与图 217 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跳变时间预设的唤醒跳变除外（参见图 218）。
 - 一些功耗未知或可忽略不计的外设将以“*”突出显示（参见图 218）。

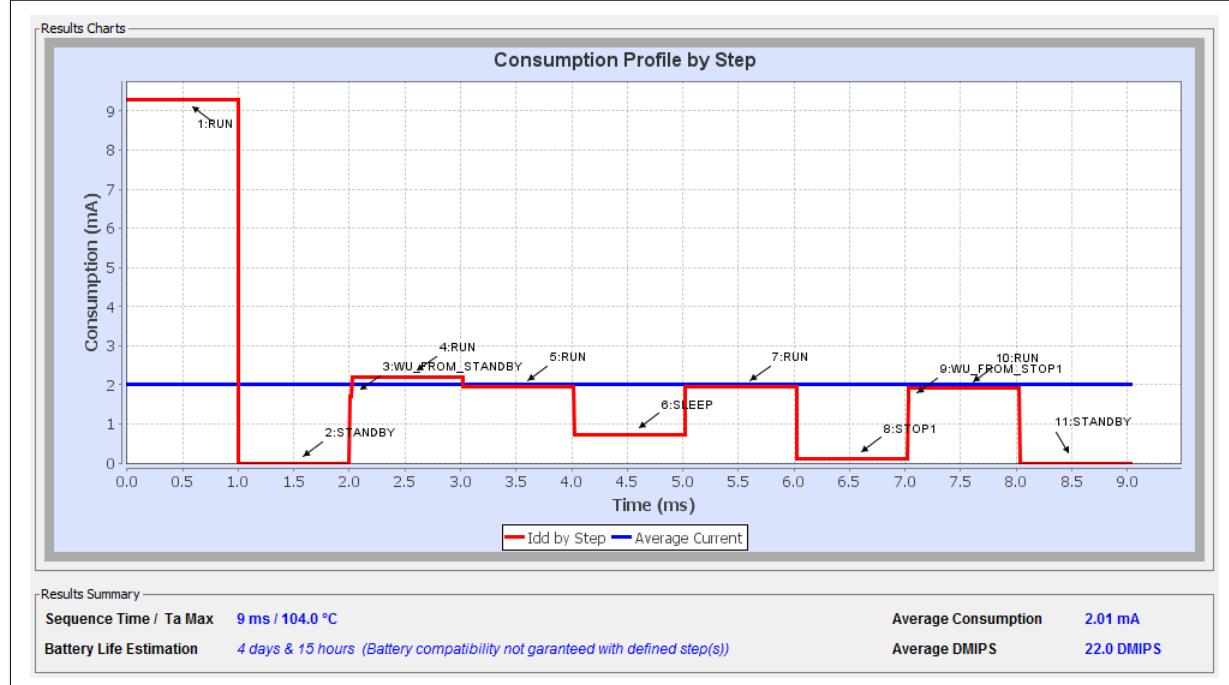
图218. 序列表

Sequence Table															
Step	Mode	...	Range/Scale	Memory	...	Clock C...	Src Freq	Periphe...	Add. C...	Step C...	Duration	DMIPS	Voltag...	Ta Max	Category
1	RUN	2.4	Range1-High	FLASH/A...	...	HSE	24.0 MHz	ADC1:fs...	0 mA	9.36 mA	1 ms	30.0	Battery	103.99	Datasheet
2	STANDBY	2.4	NoRange	n/a	...	LSI RTC	37.0 kHz	RTC*	0 mA	0.46 μA	1 ms	0.0	Battery	105	Datasheet
3	WU_FROM_ST...	2.4	NoRange	n/a	...	MSI FAST	4.0 MHz		0 mA	1.7 mA	20.1 μs	0.0	Battery	104.82	Datasheet
4	RUN	2.4	Range1-High	FLASH/A...	...	HSE	16.0 MHz	RTC	0 mA	2.16 mA	1 ms	20.0	Battery	104.77	Datasheet
5	RUN	2.4	Range2-Medium	FLASH/A...	...	HSE	16.0 MHz	ADC1:fs...	0 mA	1.92 mA	1 ms	20.0	Battery	104.79	Datasheet
6	SLEEP	2.4	Range2-Medium	ON	...	HSE	16.0 MHz	ADC1:fs...	0 mA	703.2 μA	1 ms	0.0	Battery	104.92	Datasheet
7	RUN	2.4	Range2-Medium	FLASH/A...	...	HSE	16.0 MHz	DMA1 R...	0 mA	1.92 mA	1 ms	20.0	Battery	104.79	Datasheet
8	STOP1	2.4	NoRange	n/a	...	ALL CLO...	0 Hz	USART1*	0 mA	6.65 μA	1 ms	0.0	Battery	105	In DS Ta...
9	WU_FROM_ST...	2.4	NoRange	n/a	...	HSI16	16.0 MHz		0 mA	1.62 mA	6.3 μs	0.0	Battery	104.83	Datasheet
10	RUN	2.4	Range2-Medium	FLASH/A...	...	HSE	16.0 MHz	RTC USA...	0 mA	1.89 mA	1 ms	20.0	Battery	104.8	Datasheet
11	STANDBY	2.4	NoRange	n/a	...	LSI RTC	37.0 kHz	RTC*	0 mA	0.46 μA	1 ms	0.0	Battery	105	Datasheet

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 219）。

图219. 优化前的序列结果



9.3.2 优化应用功耗

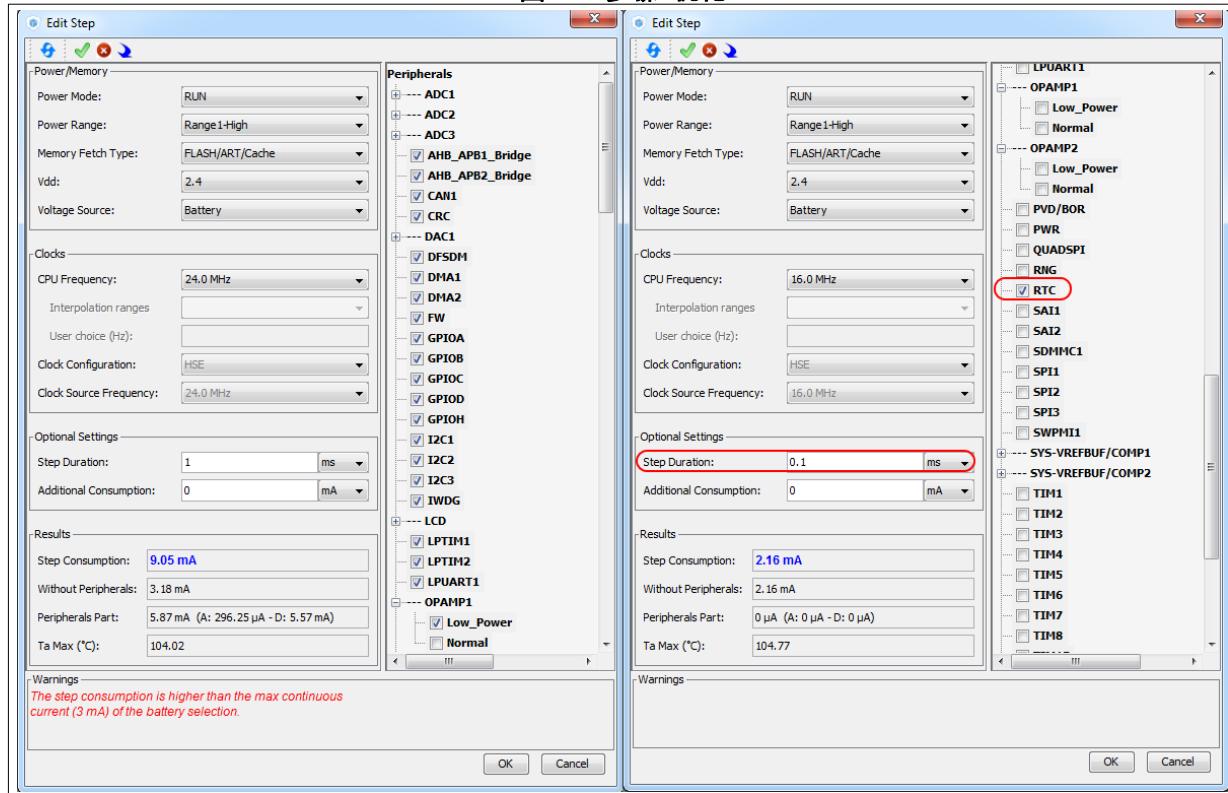
我们现在通过一些操作优化总功耗和电池使用寿命。这些操作在步骤1、4、5、6、7、8和10中执行。

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 220）。

图220. 步骤1优化



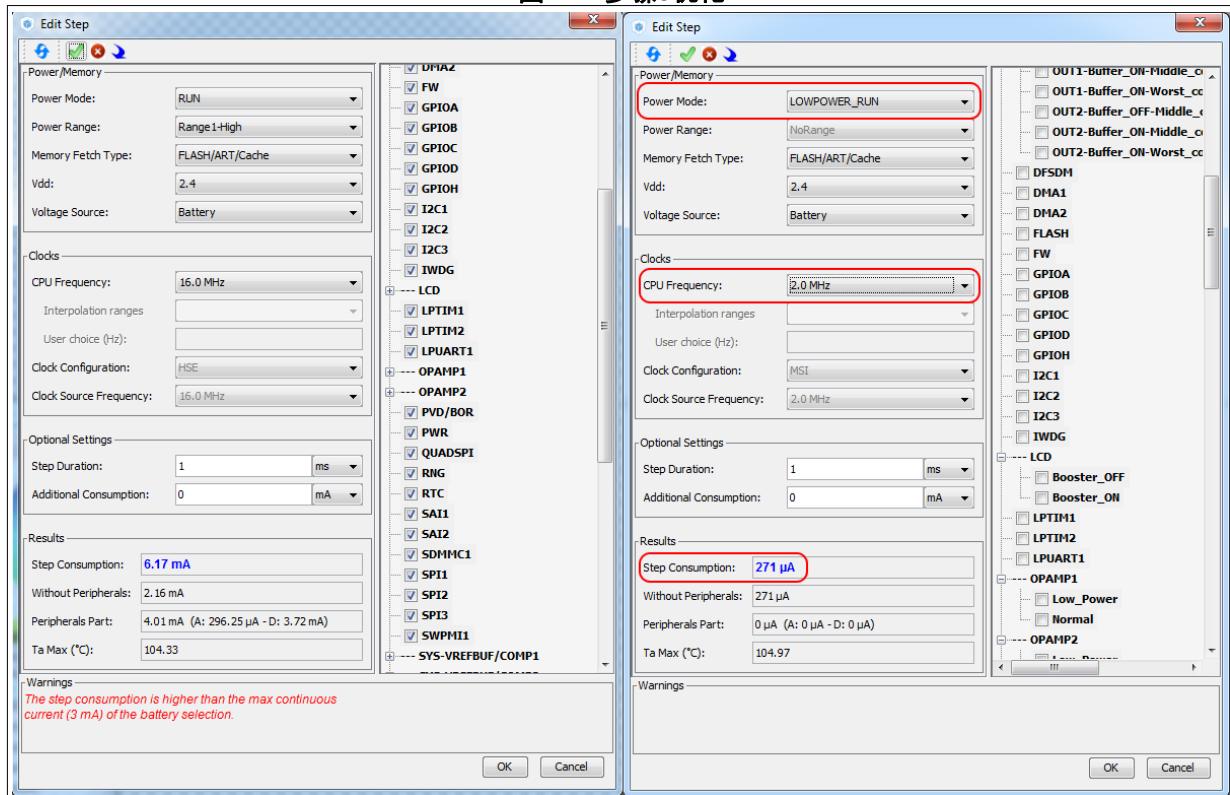
步骤4（运行， RTC）

- 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 221）。

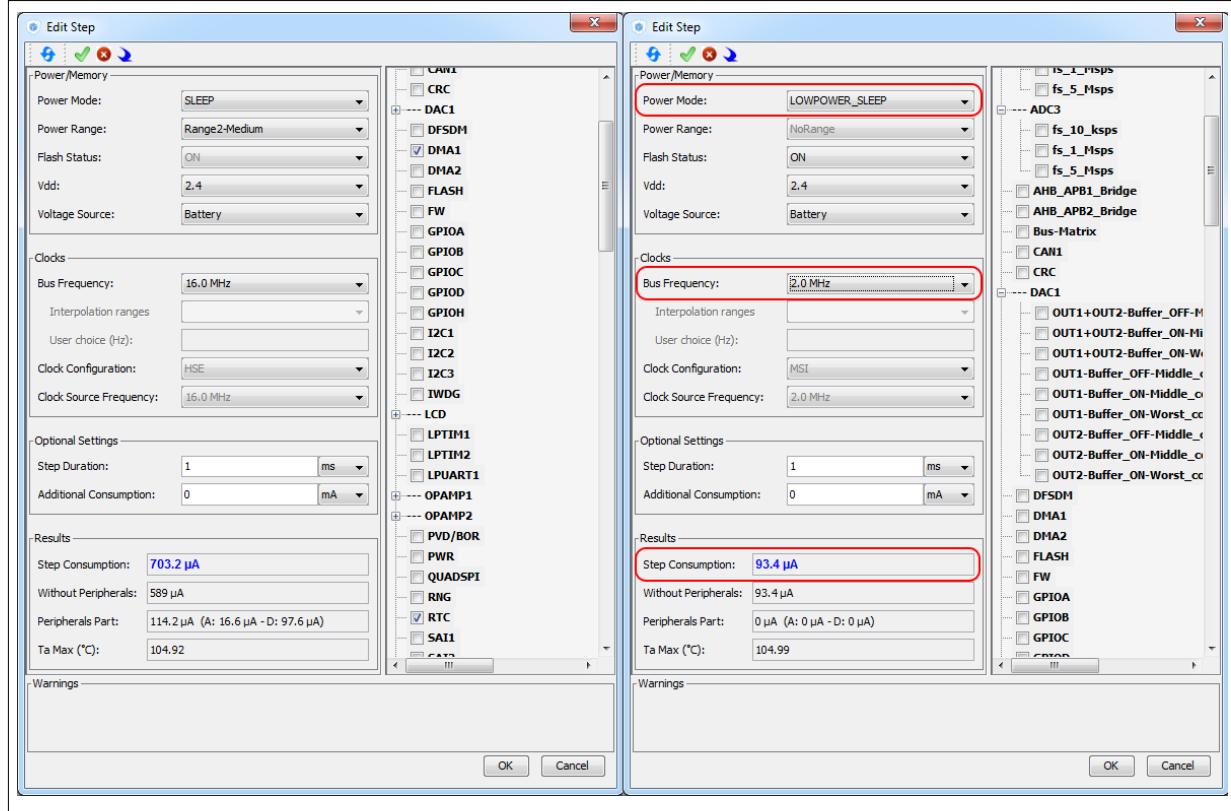
图221. 步骤5优化



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 μ A降低至93 μ A（参见图 222）。

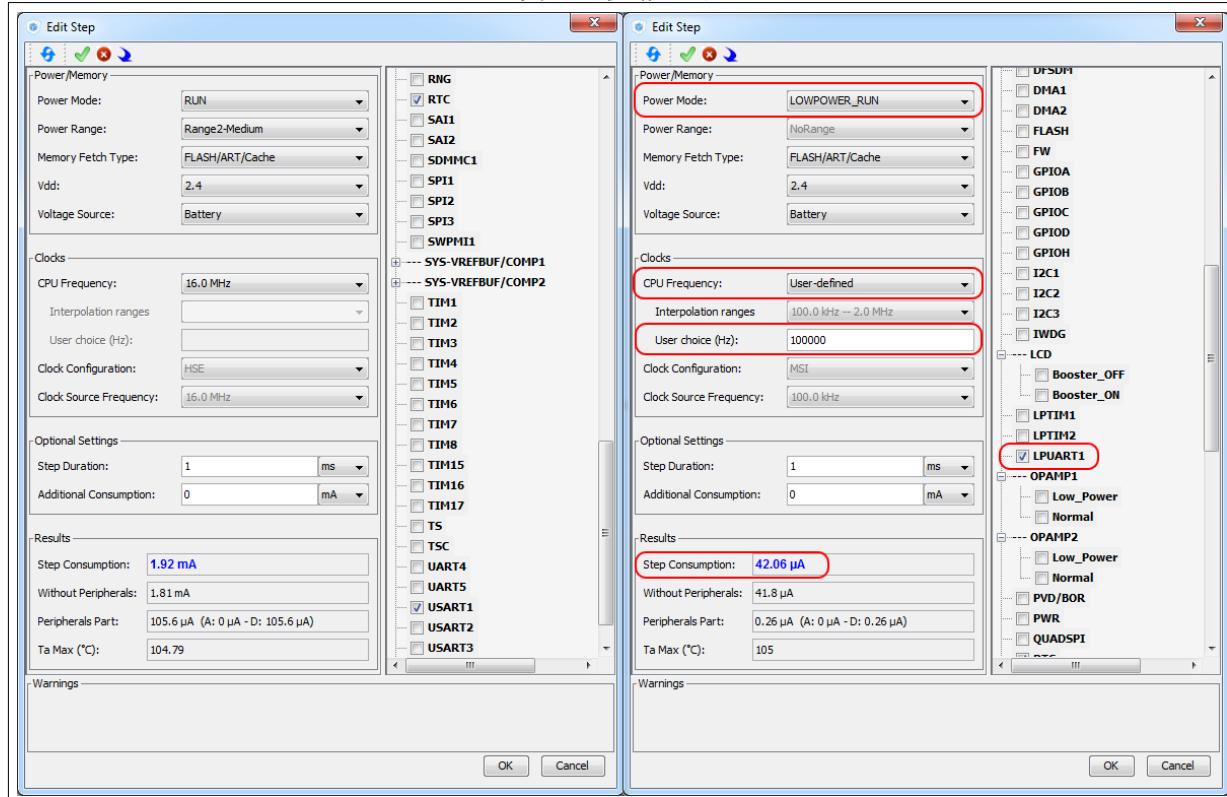
图222. 步骤6优化



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 μ A降低至42 μ A（参见图 223）。

图223. 步骤7优化

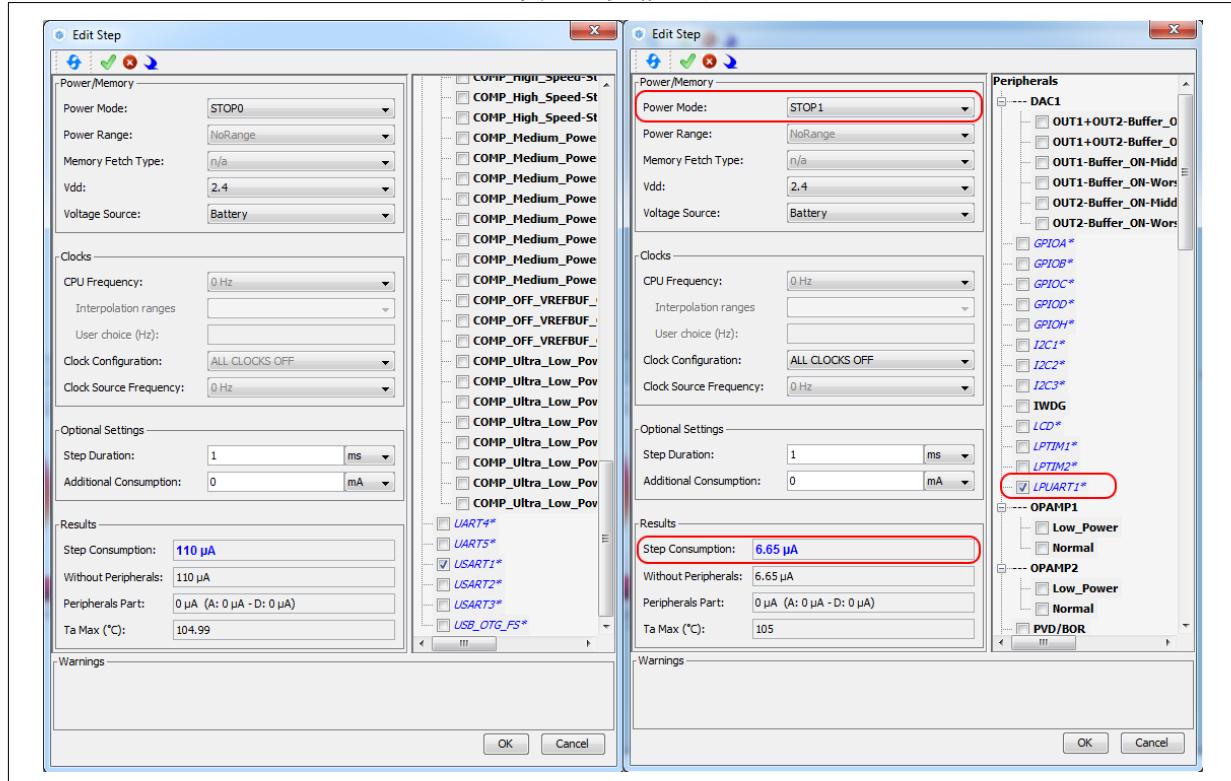


步骤8（停机0，USART）

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

电流消耗从 $110 \mu\text{A}$ 降低至 $6.65 \mu\text{A}$ （参见图 224）。

图224. 步骤8优化



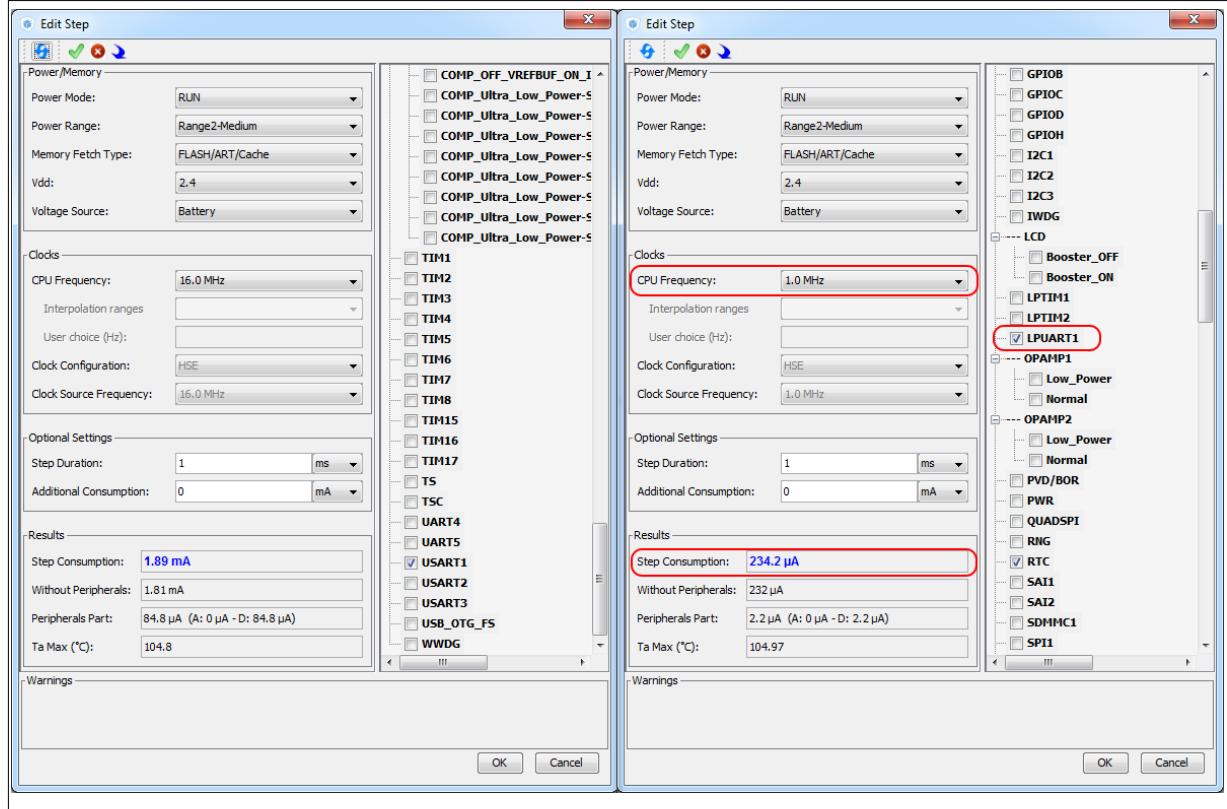
步骤10 (RTC, USART)

- 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 μ A（参见图 225）。

图 226中提供的示例显示平均电流消耗降低155 μ A。.

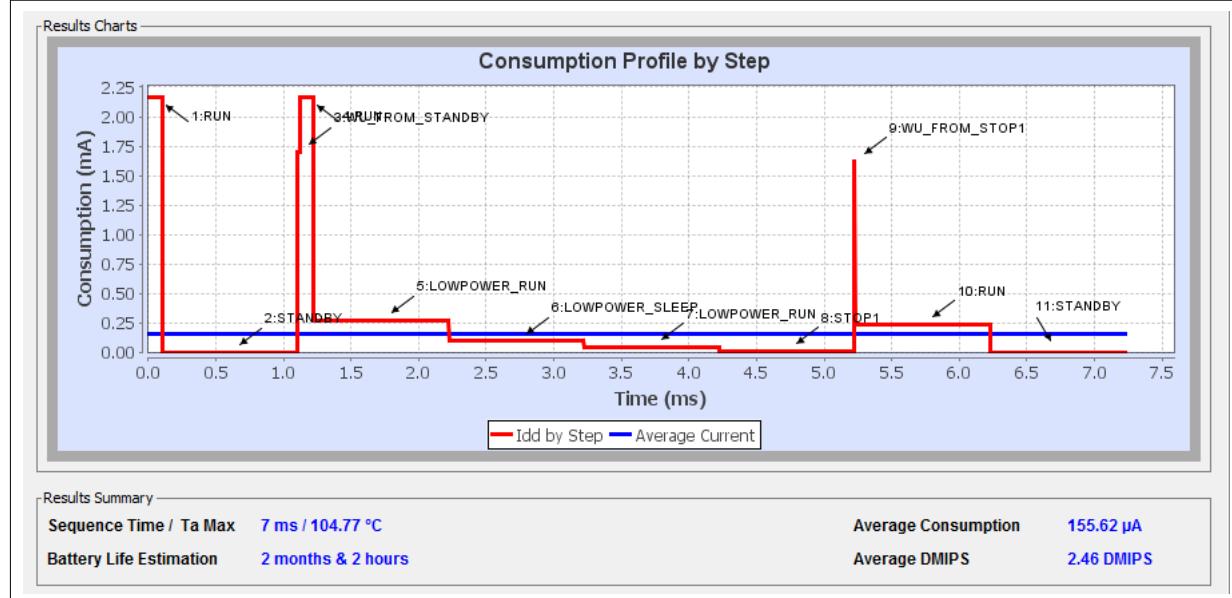
图225. 步骤10优化



有关序列整体结果，请参见图 226：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 μ A。

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226. 优化后的功耗系列结果



10 教程4 - 通过串口与STM32L053xx Nucleo板通信示例

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。

10.1 教程概述

教程4将介绍以下步骤：

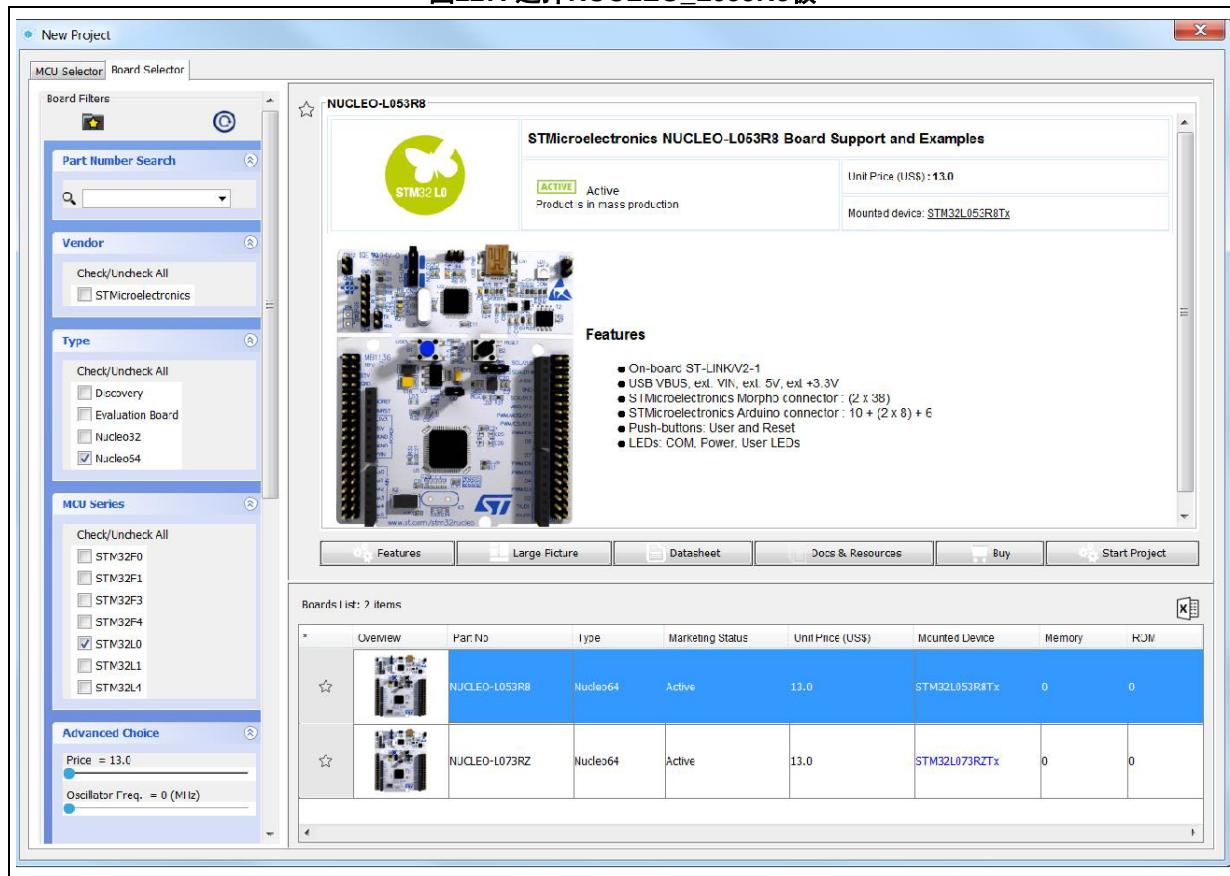
1. 从新项目菜单中选择NUCLEO-L053R8板。
2. 从引脚布局视图中选择所需功能（调试、USART、定时器）：外设工作模式以及引脚上的相关信号分配。
3. 在时钟配置视图中配置MCU时钟树。
4. 在配置视图中配置外设参数
5. 在“项目设置”菜单中配置项目设置并生成项目（仅限初始化代码）。
6. 为项目更新对应于UART通信的用户应用代码示例。
7. 在板子上编译和执行项目。
8. 将Tera Term软件配置为PC上的串行通信客户端。
9. 结果显示在PC屏上。

10.2 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 227](#)）。

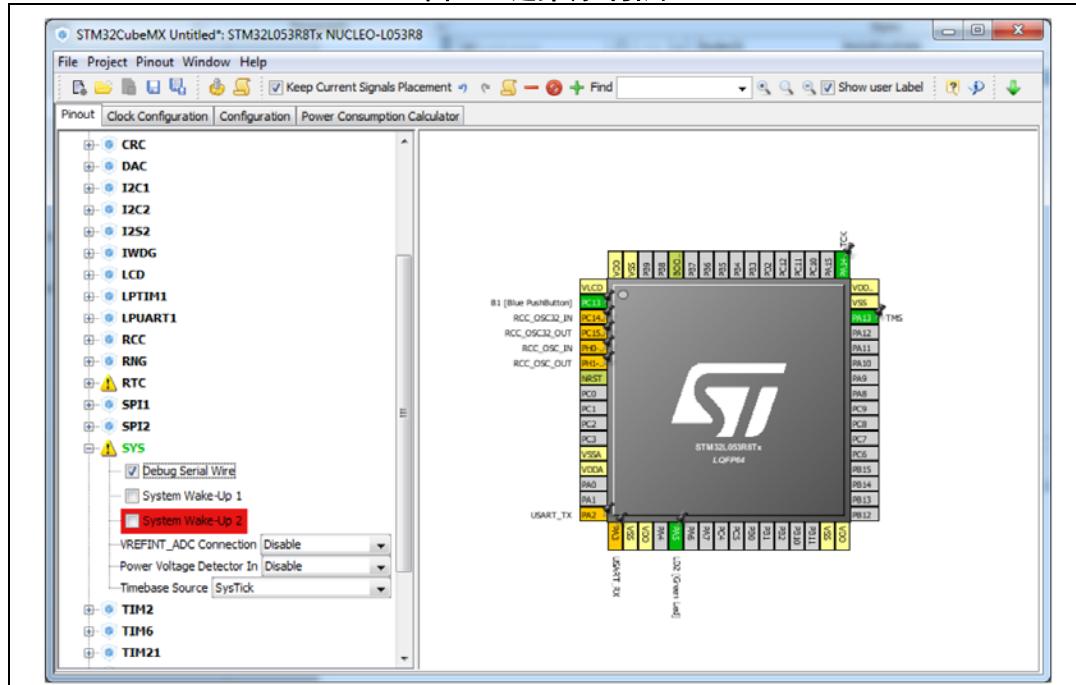
图227. 选择NUCLEO_L053R8板



10.3 从“引脚布局”视图中选择功能

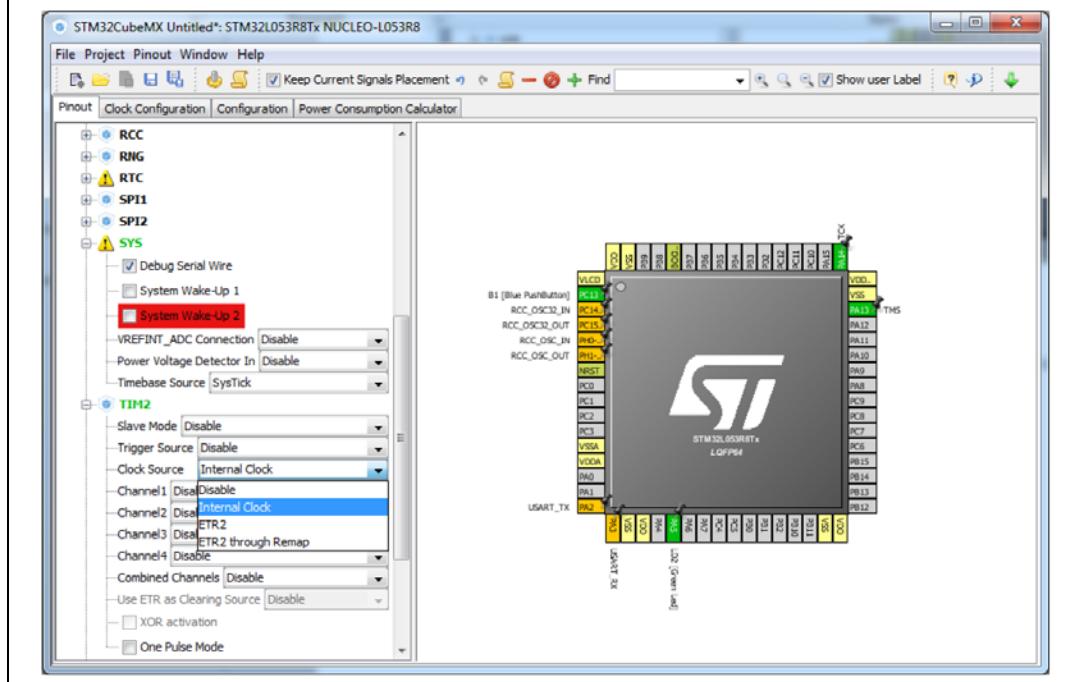
- 在SYS下选择“调试串行线”（参见图 228）。

图228. 选择调试引脚



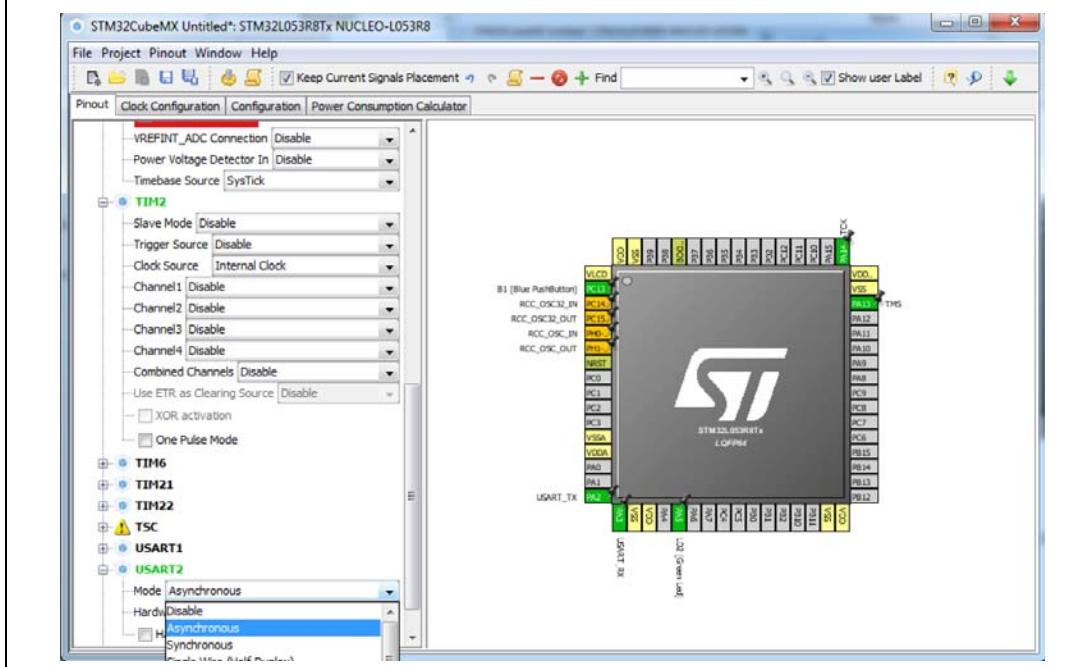
2. 在TIM2外设下选择“内部时钟”作为时钟源（参见图 229）。

图229. 选择TIM2时钟源



3. 为USART2外设选择异步模式（参见图 230）。

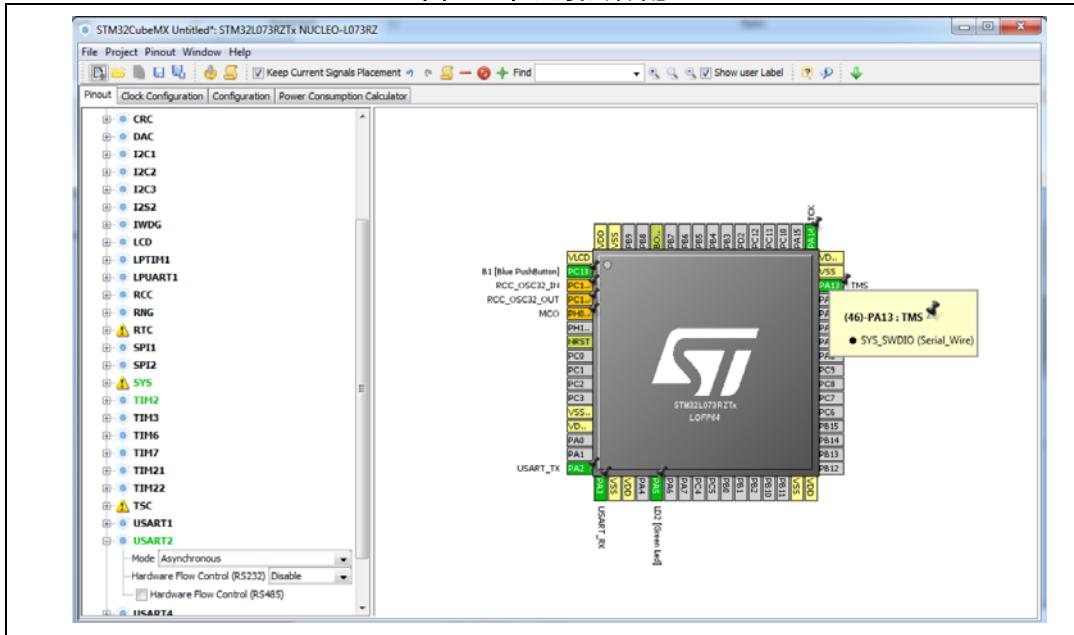
图230. 为USART2选择异步模式



4. 检查为各引脚分配的信号是否正确（参见图 231）：

- 为PA13分配SYS_SWDIO
- 为PA14分配TCK
- 为PA2分配USART_TX
- 为PA3分配USART_RX

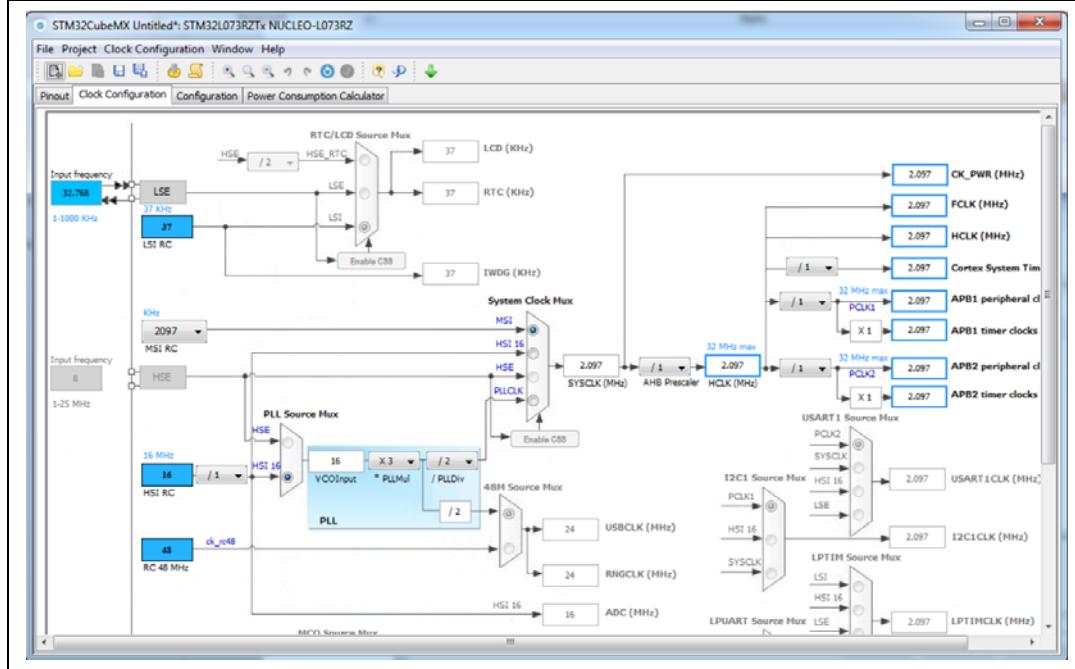
图231. 检查引脚分配



10.4 在“时钟配置”视图中配置MCU时钟树

- 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 232）。

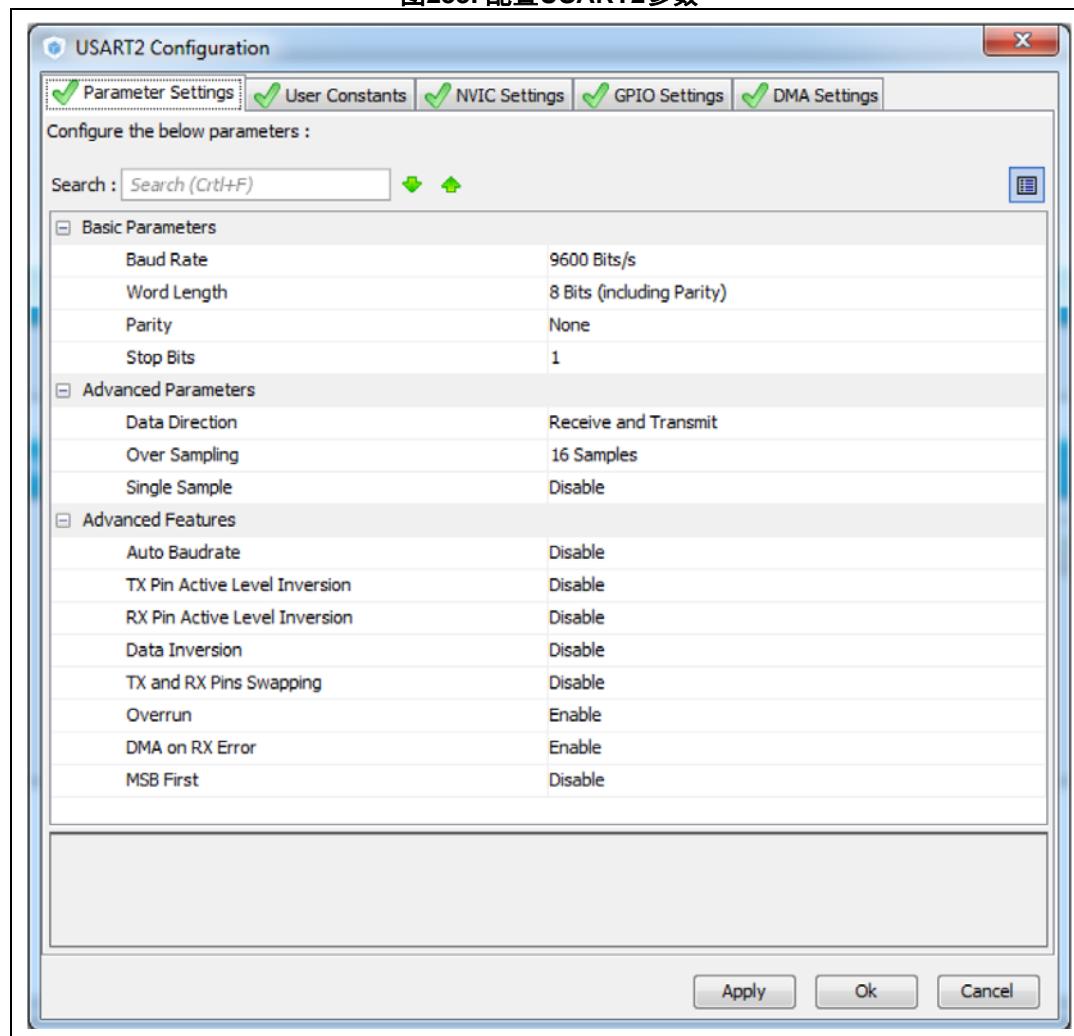
图232. 配置MCU时钟树



10.5 在“配置”视图中配置外设参数

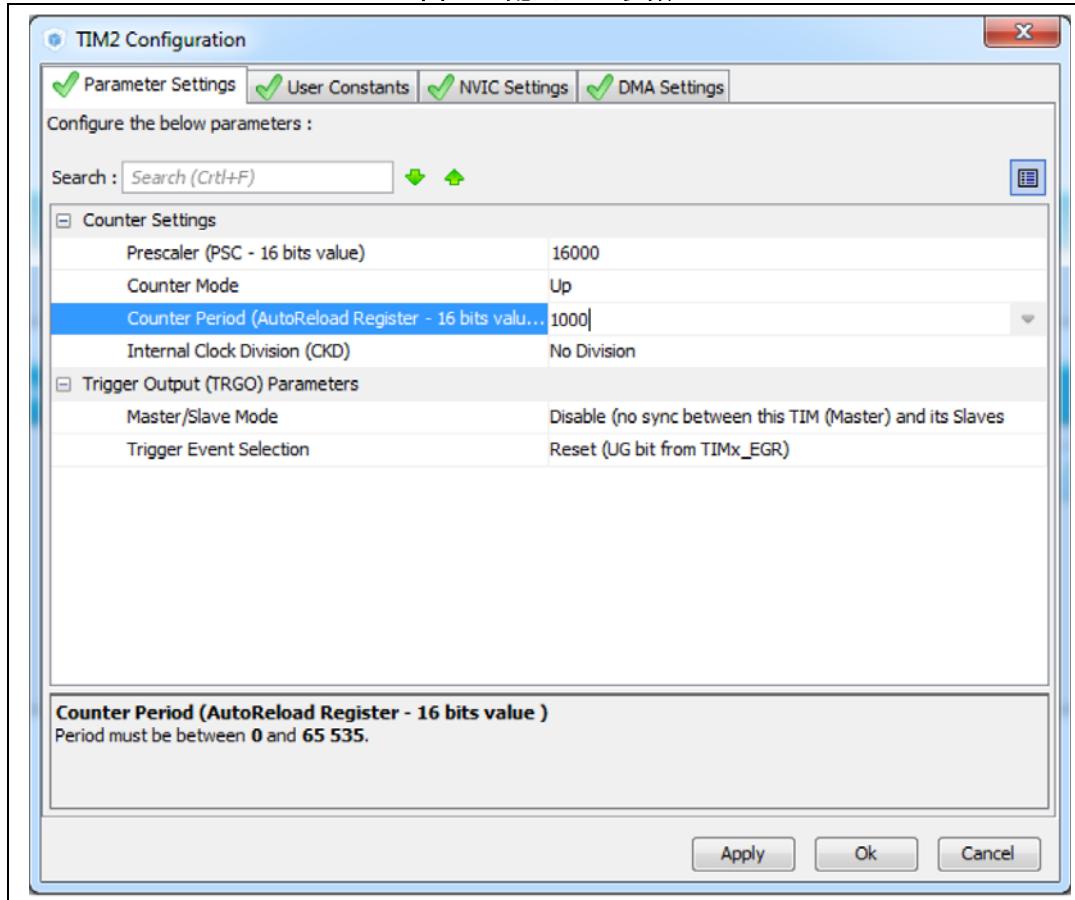
1. 在配置选项卡中，点击USART2打开外设参数设置窗口，并将波特率设为9600。确保“数据方向”设为“接收和发送”(参见图 233)。
2. 点击“确定”应用更改并关闭窗口。

图233. 配置USART2参数



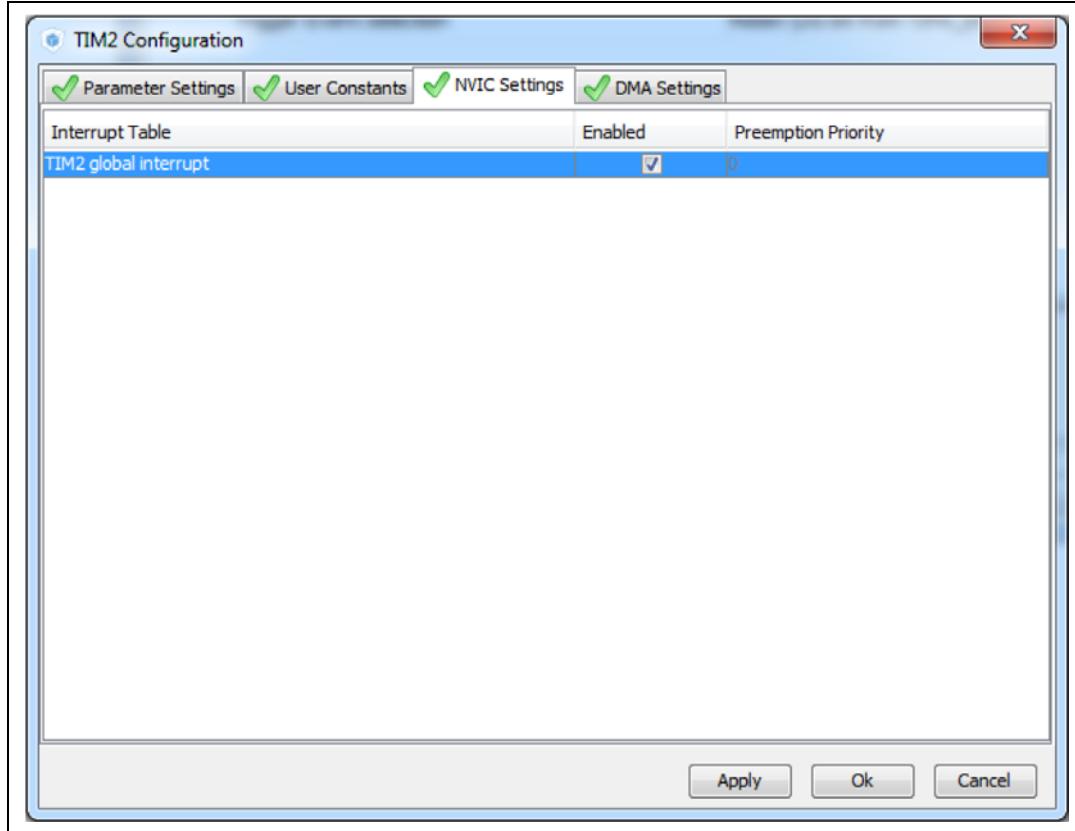
3. 点击TIM2并将预分频器改为16000，将“字长”改为8位，将“计数器周期”改为1000（参见图 234）。

图234. 配置TIM2参数



4. 在NVIC设置选项卡中启用TIM2全局中断（参见图 235）。

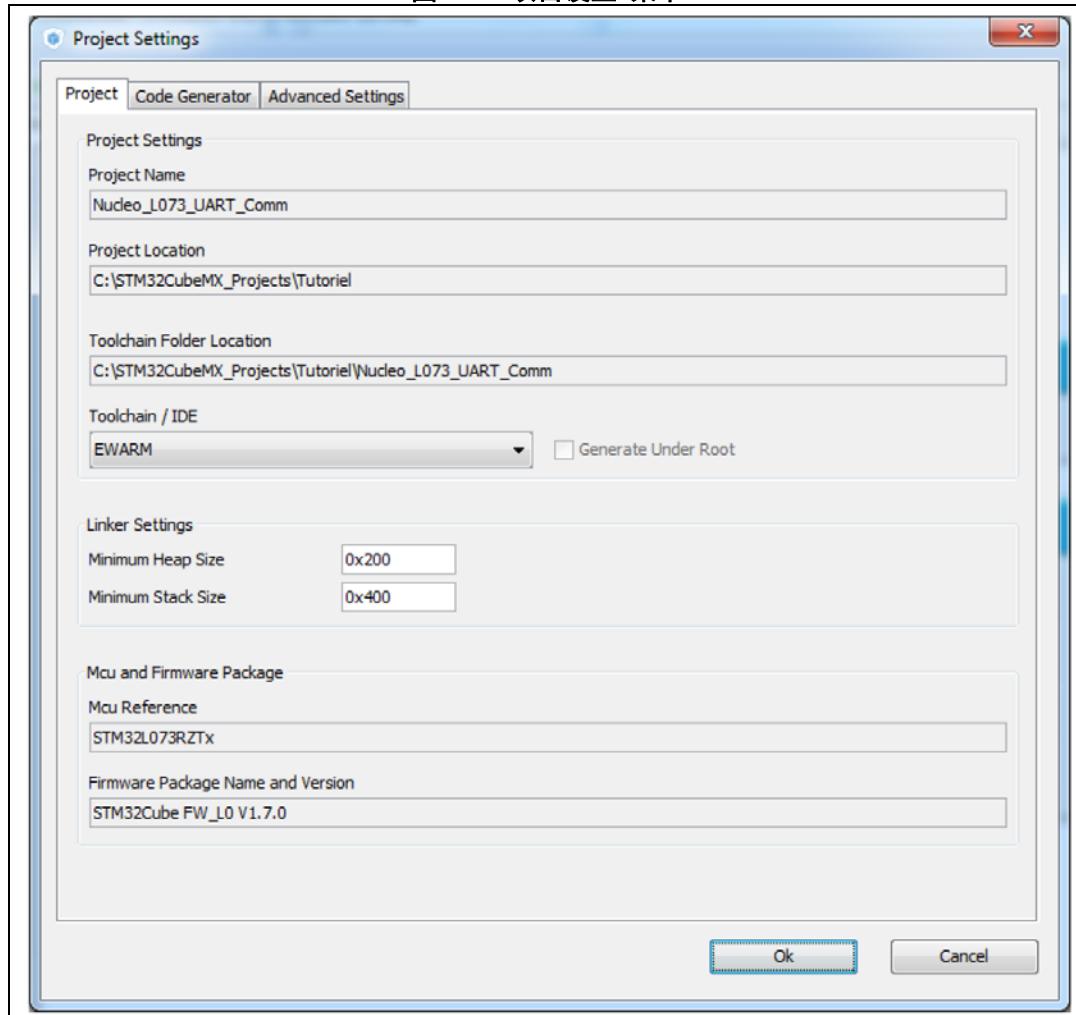
图235. 启用TIM2中断



10.6 配置项目设置并生成项目

- 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 236）。

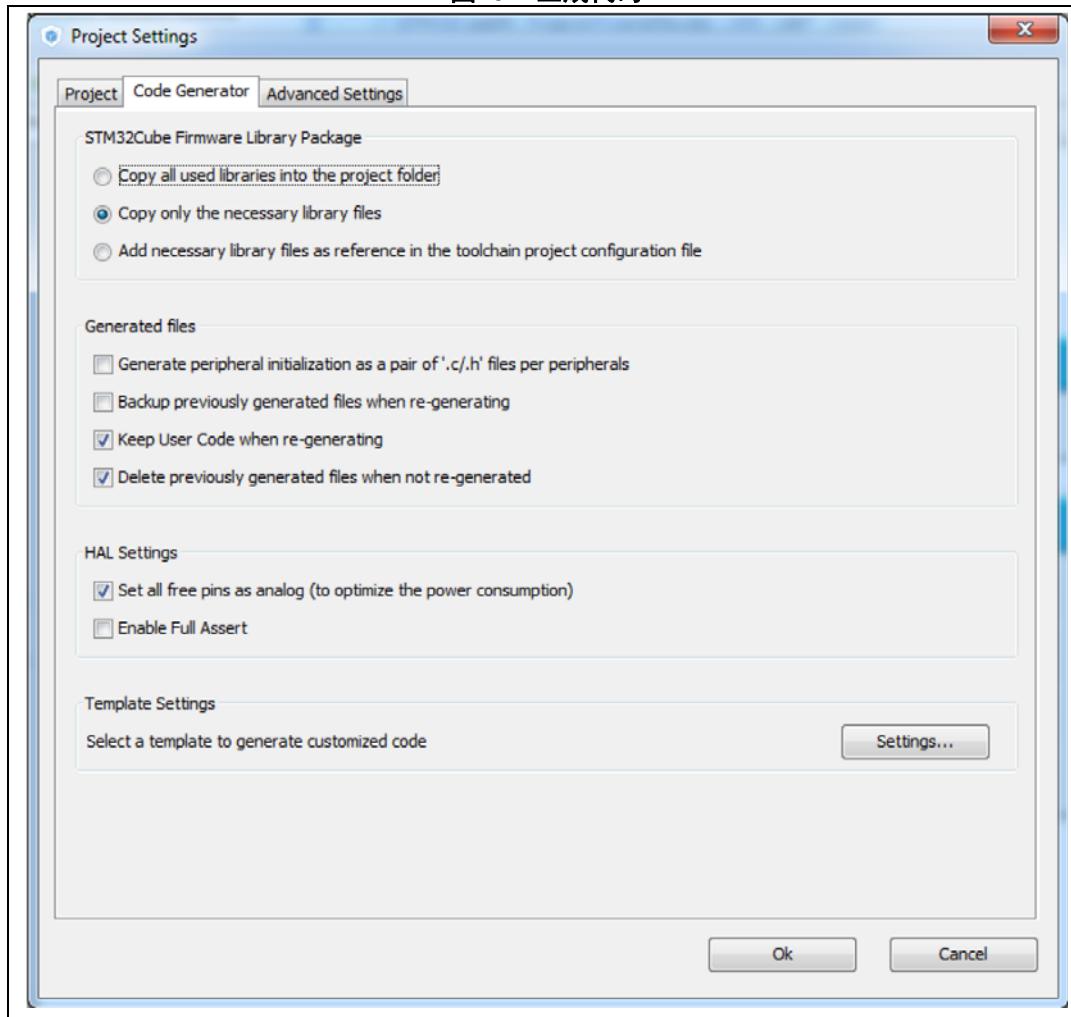
图236. “项目设置”菜单



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 237 所示配置要生成的代码，并点击确定以生成代码。

图237. 生成代码



10.7 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 */
```

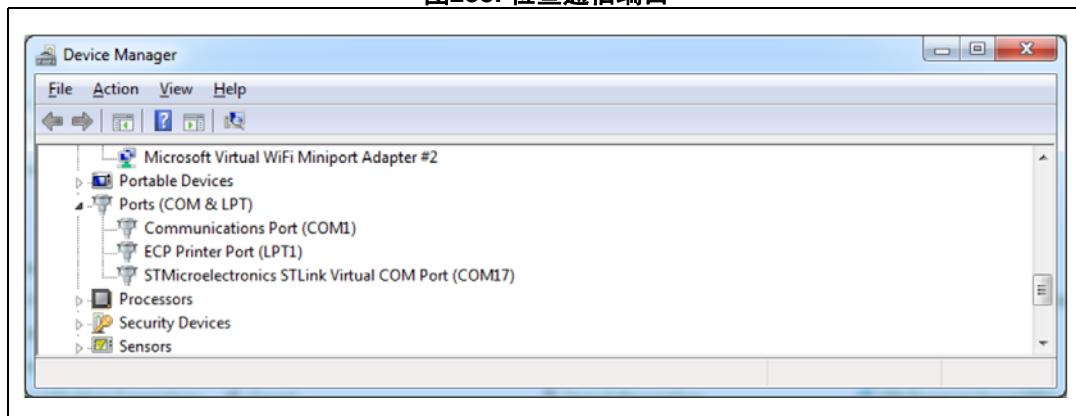
10.8 编译并运行项目

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。

10.9 将Tera Term软件配置为PC上的串行通信客户端

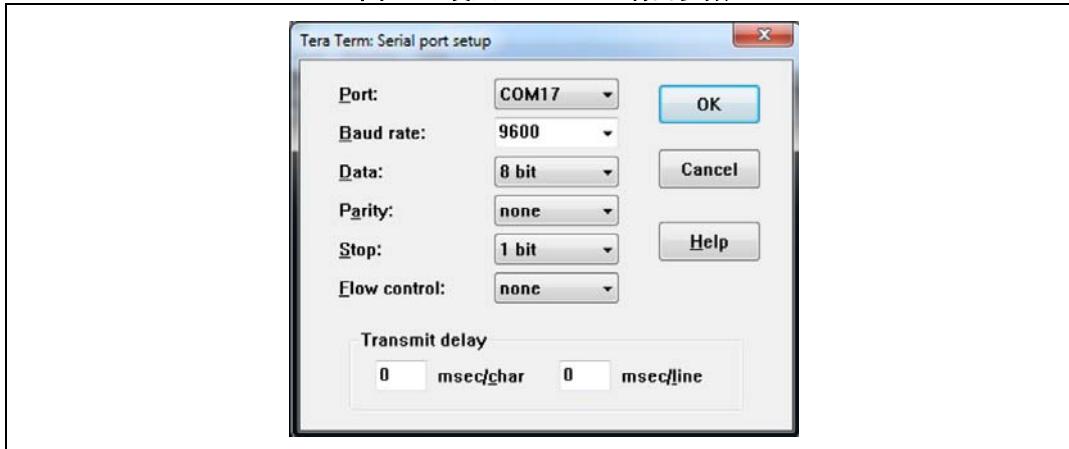
1. 在计算机上的“设备管理器”窗口中检查ST Microelectronics使用的虚拟通信端口（参见图 238）。

图238. 检查通信端口



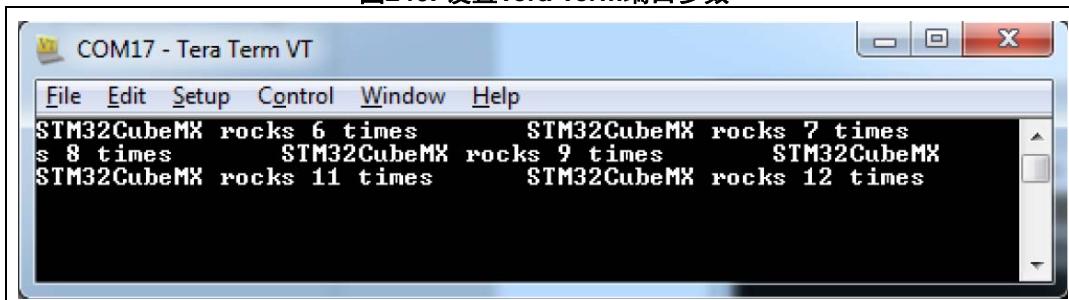
2. 要将Tera Term配置为侦听相关虚拟通信端口，请对参数进行调整，以使其与MCU上的USART2参数设置相匹配（参见图 239）。

图239. 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 240）。

图240. 设置Tera Term端口参数



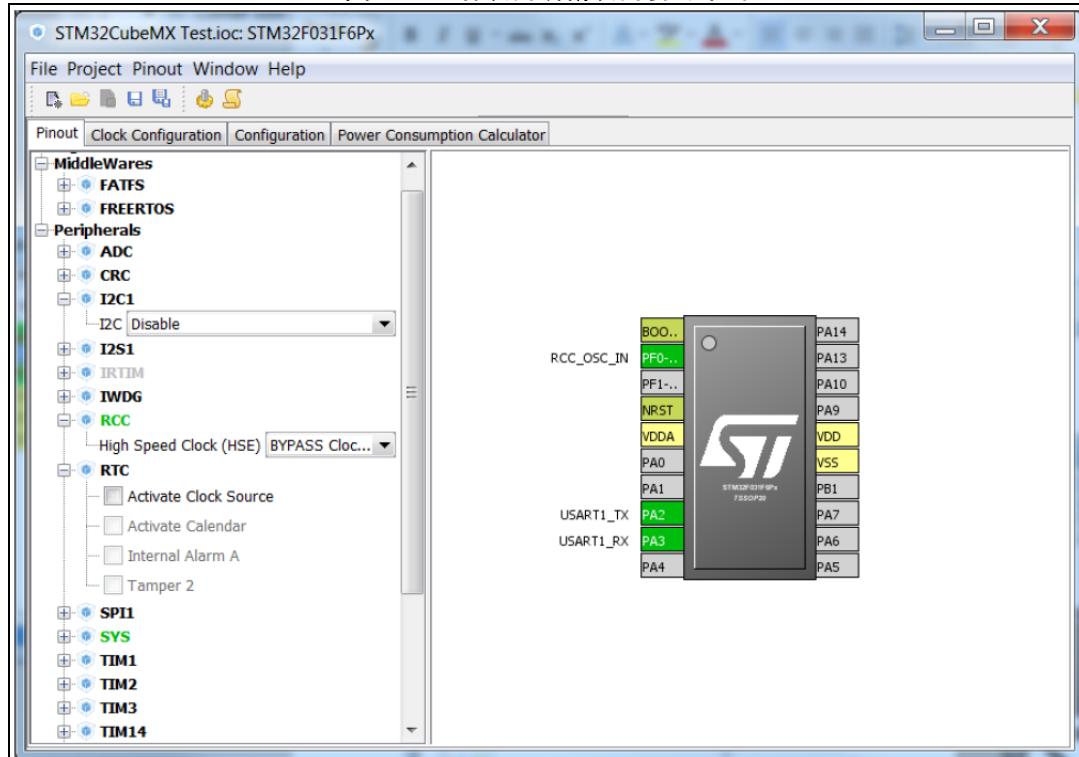
11 教程5：将当前项目配置导出到兼容MCU

如果在引脚布局菜单中选择列出与引脚布局兼容的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241. 已有项目或新项目引脚布局



2. 进入引脚布局菜单并选择列出与引脚布局兼容的MCU。弹出引脚布局兼容窗口（参见图 242 和图 243）。

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。

行颜色阴影和注释列表表示匹配度：

- 亮绿色表示完全匹配：MCU与当前项目完全匹配（相关示例，请参见图 243）。
- 浅绿色表示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 242）。

- 黄色表示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 243）。

图242. 与引脚布局兼容的MCU列表 - 部分匹配且硬件兼容

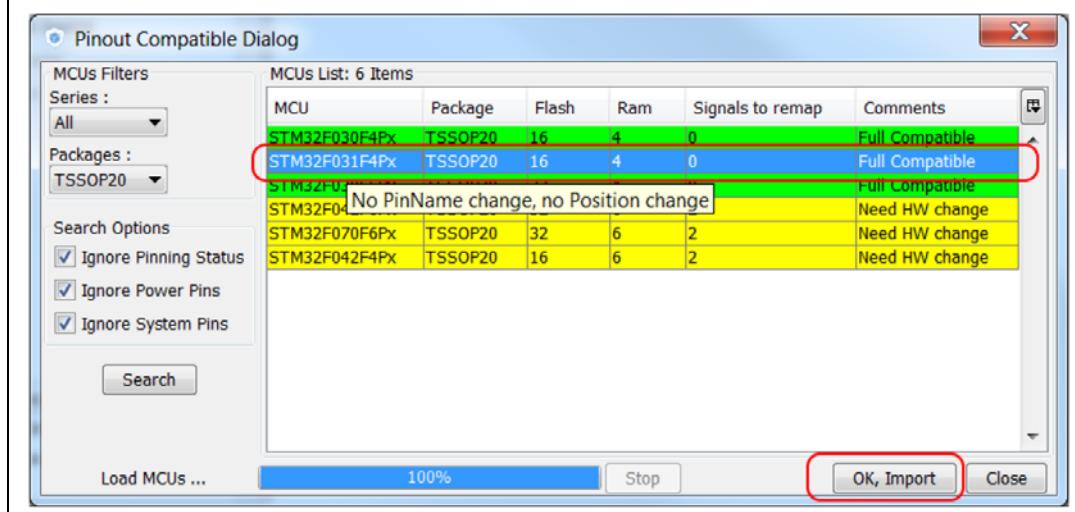
The screenshot shows the 'Pinout Compatible Dialog' window. On the left, there are filters for 'Series' (set to 'STM32F0') and 'Packages' (set to 'All'). Under 'Search Options', three checkboxes are checked: 'Ignore Pinning Status', 'Ignore Power Pins', and 'Ignore System Pins'. A 'Search' button is below these options. To the right is a table titled 'MCUs List: 105 Items' with columns: MCU, Package, Flash, Ram, Signals to remap, and Comments. The table lists various STM32F0 series MCUs with their respective package types (e.g., UFQFPN48, LQFP48) and memory sizes. A specific row for STM32F072CBUx is highlighted in yellow, with a tooltip indicating 'USART1_RX at the same Position pin(43), PinName changes from PA10 to PB7'.

图243. 与引脚布局兼容的MCU列表 - 完全匹配和部分匹配

The screenshot shows the 'Pinout Compatible Dialog' window with different filter settings. The 'Series' dropdown is set to 'All' and the 'Packages' dropdown is set to 'TSSOP20'. The 'Search Options' checkboxes are all checked. The table on the right is titled 'MCUs List: 6 Items' and contains six rows, each with a different STM32F0 series MCU and its TSSOP20 package. The last three rows (STM32F038F6Px, STM32F042F6Px, STM32F042F4Px) have a yellow background, indicating they require hardware changes ('Need HW change').

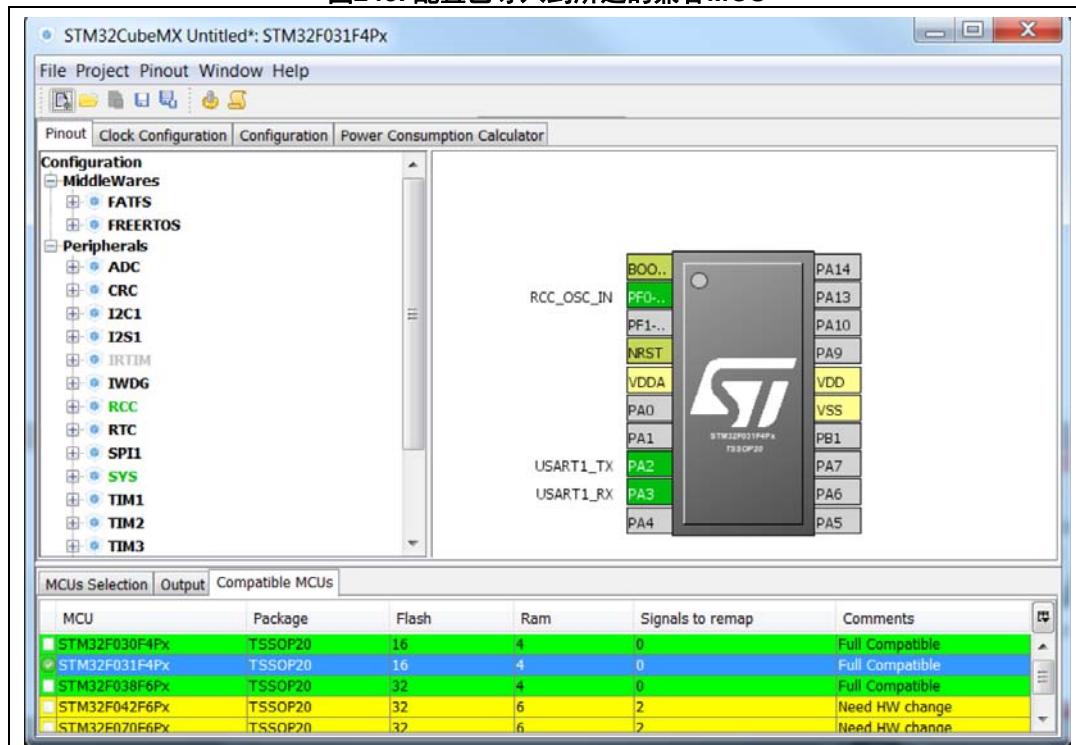
3. 然后选择要导入当前配置的MCU，并点击确定，导入：

图244. 选择兼容MCU，并导入配置



配置现在可用于所选MCU：

图245. 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。

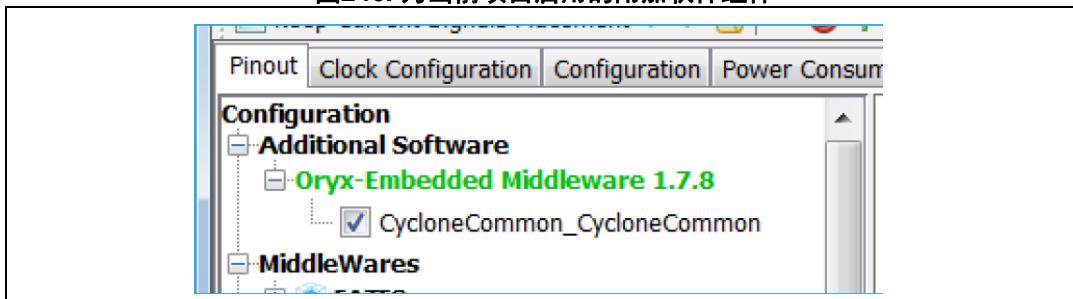
12 教程6 – 将嵌入式软件包添加到用户项目

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[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. 选择添加软件包组件菜单  打开附加软件组件窗口，并选择以下软件组件：CycloneCommon软件包中的Compiler Support、RTOS Port/None和Date Time Helper Routines（参见[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 246](#)）。

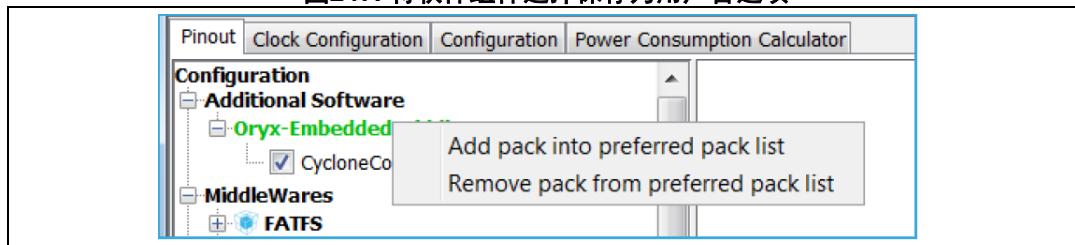
图246. 为当前项目启用的附加软件组件



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。

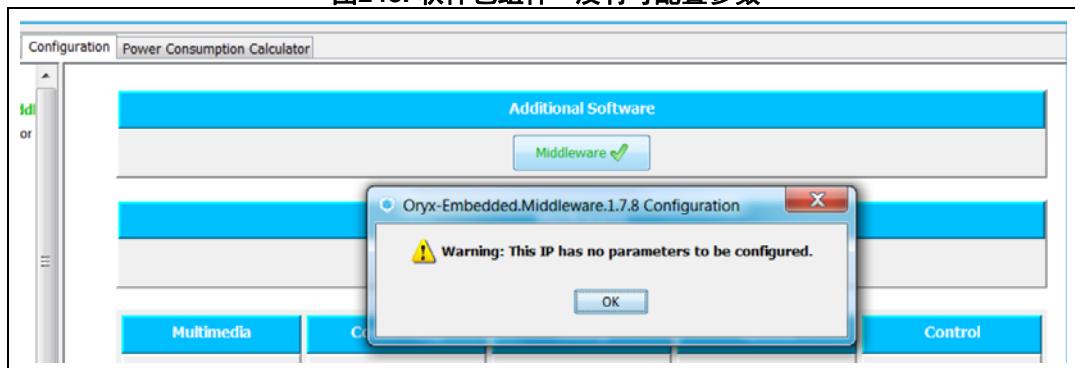
6. 也可保留当前软件组件选择，供后续项目使用：
 - a) 右键单击软件包名称并选择将软件包添加到首选软件包列表。稍后可点击将软件包从首选软件包列表中删除以将其删除（参见[图 247](#)）。
- 通过这种方式，下次创建项目时，将自动在附加软件下推荐对应于所选软件的软件包。

图247. 将软件组件选择保存为用户首选项



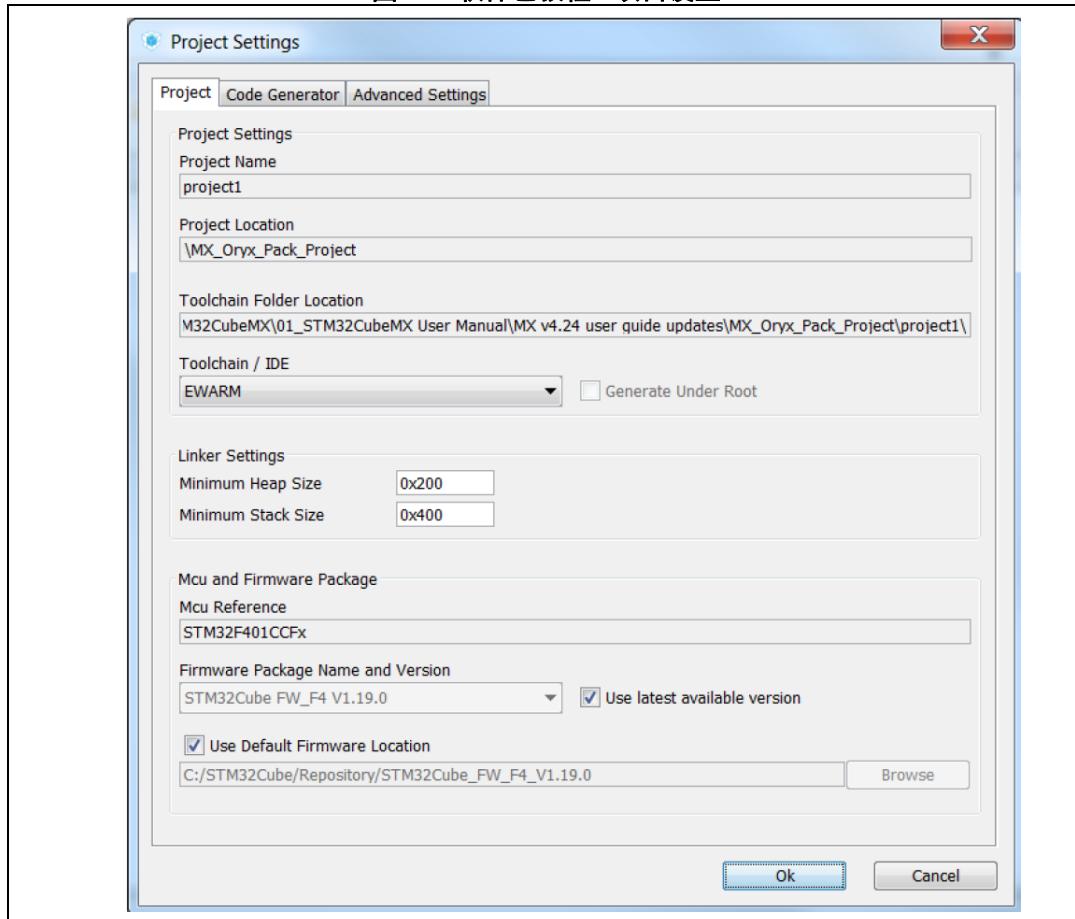
7. 检查是否没有参数可在配置选项卡中进行配置（参见图 248）。

图248. 软件包组件 - 没有可配置参数



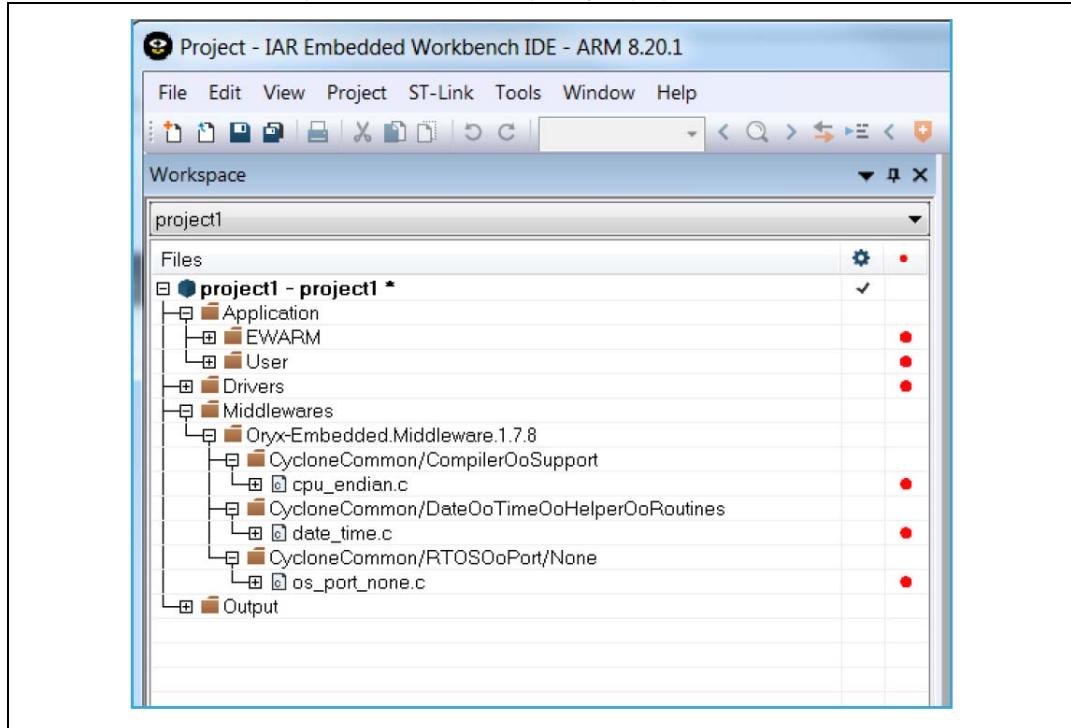
8. 选择项目，然后选择项目设置指定项目参数（参见图 249），并选择IAR™ EWARM 作为IDE。

图249. 软件包教程 - 项目设置



9. 点击 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。
10. 点击打开项目。Oryx软件组件会显示在生成的项目中（参见 [图 250](#)）。

图250. 生成的项目以及第三方软件包组件



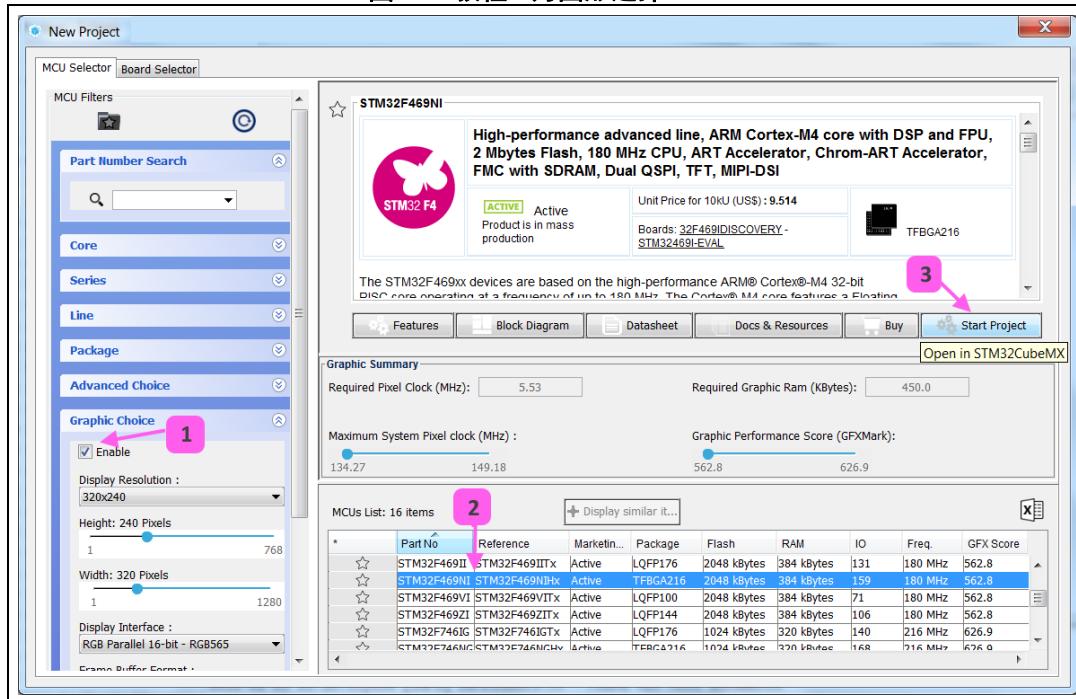
13 教程 7 – 使用STemWin图形框架

13.1 步骤1：为图形选择MCU

首先启动STM32CubeMX，并点击“新项目”打开“新项目”窗口（参见图 251）

- 在左侧面板上，选中“图形”选择下的“启用”复选框可显示与图形相关的参数、图形总结面板以及合适的MCU列表。
- 从MCU列表中选择STM32F469NIHx。
- 点击“开始项目”打开STM32CubeMX项目主视图。

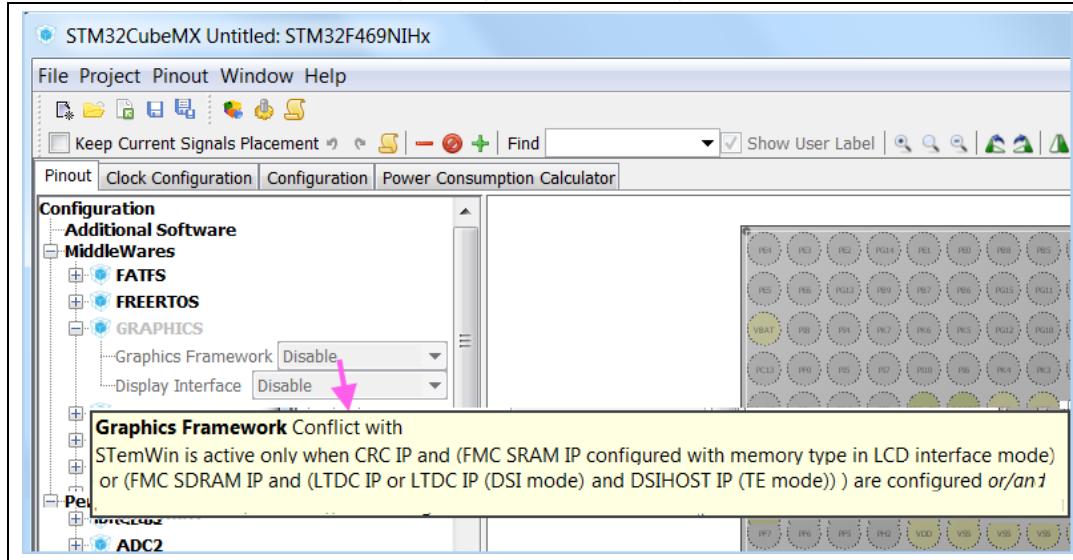
图251. 教程 - 为图形选择MCU



13.2 步骤2：在引脚布局视图中启用STemWin

要启用STemWin，必须解决冲突。将鼠标悬停在“图形框架选择”字段可在工具提示中显示冲突详情（参见图 252）。

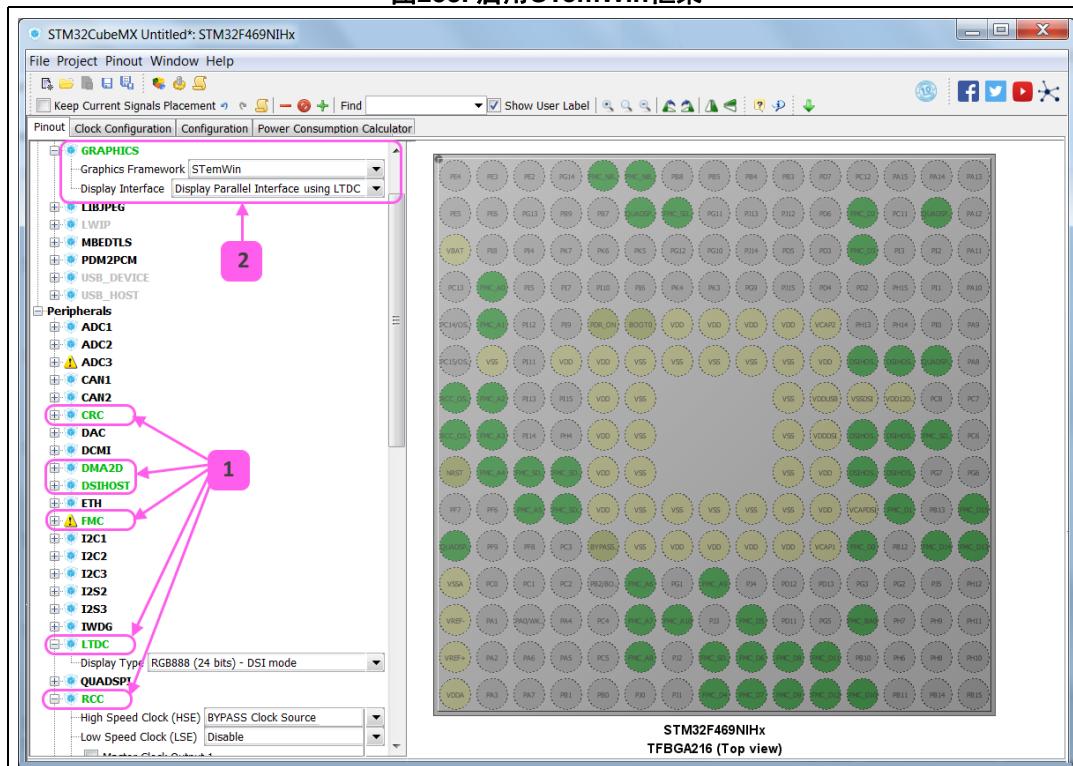
图252. 图形框架工具提示



在引脚布局视图左侧面板中（参见图 253）

- 启用必要的外设：CRC、DMA2D（可选，用于图形加速）、FMC SDRAM模式、RCC HSE、LTDC（DSI模式）和DSI主机（适配指令模式）及TE引脚（用于连接“显示”界面）。在“图形”下，STemWin框架此时可供选择。
- 选择“STemWin”作为图形框架，选择“使用LTDC显示并行界面”作为显示界面。

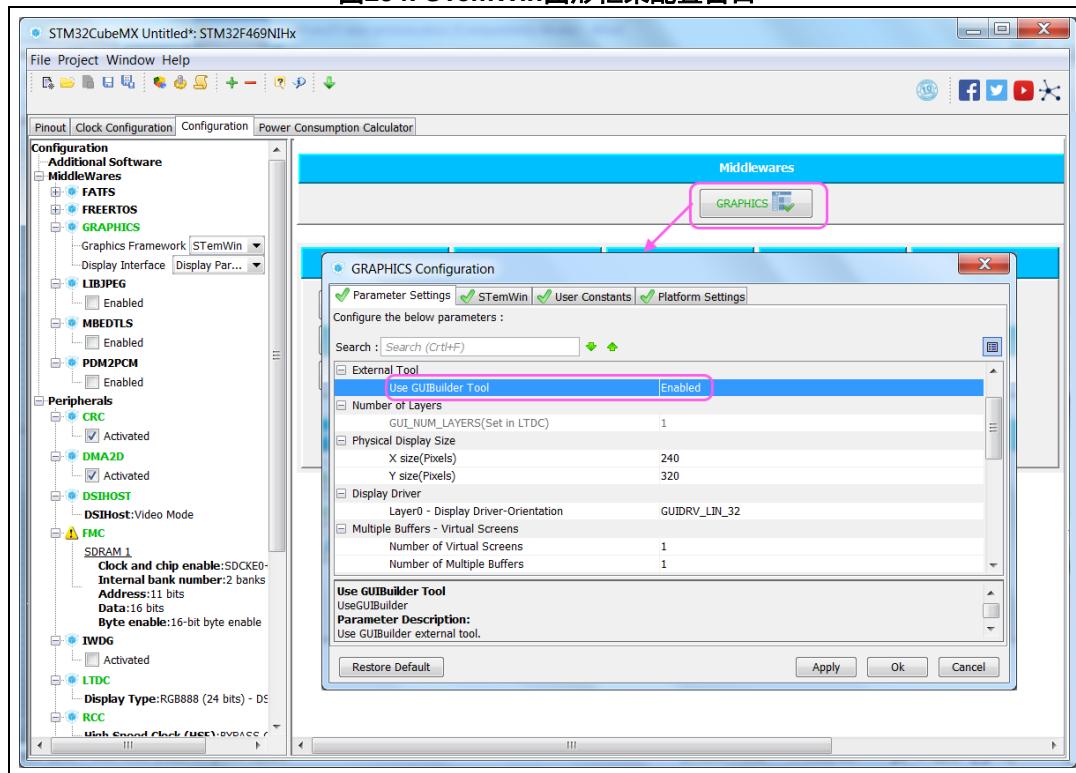
图253. 启用STemWin框架



13.3 步骤3：从配置窗口中配置STemWin参数

1. 选择配置视图并点击“图形”按钮可显示STemWin图形框架配置（参见图 254）
2. 使用“参数设置”面板调整“图形”参数：在“外部工具”组下，先启用 GUIBuilder 工具，然后再继续执行下一步。

图254. STemWin图形框架配置窗口



13.4 步骤4：在配置窗口中使用STemWin GUIBuilder工具

1. 请务必在参数设置面板中启用GUIBuilder工具。
2. 选择STemWin面板，并将“应用类别”配置为Window或WindowFrame（参见图 255）
3. 点击“执行”打开GUIBuilder用户界面（参见图 256）。

注：如果未指定项目设置，将弹出项目设置窗口。STemWin不支持TrueStudio®、Makefile以及
其他（GPDSC）工具链，这些工具链在用户界面中以灰色显示。

注：STM32CubeMX启动STM32CubeF4嵌入式软件包中的GUIBuilder可执行文件时，该包将下载
(若不可用) 到用户的STM32Cube资源库中。

注： GUIBuilder工具包打开时，STM32CubeMX用户界面不可访问。

4. 使用GUIBuilder用户界面完成GUI设计工作，并点击“文件 > 保存”生成相应的C文件，为Window类型生成的是WindowDLG.c，为WindowFrame类型生成的是FrameWindowDLG.c。
5. 关闭GUIBuilder会返回到STM32CubeMX用户界面。

图255. STemWin GUIBuilder配置面板

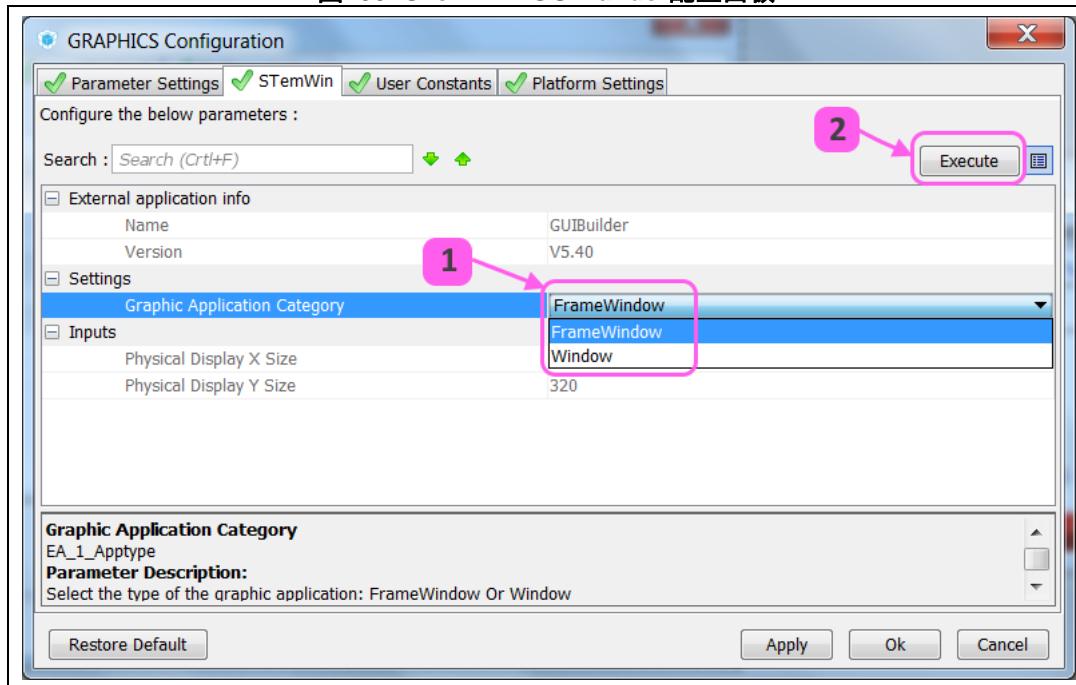
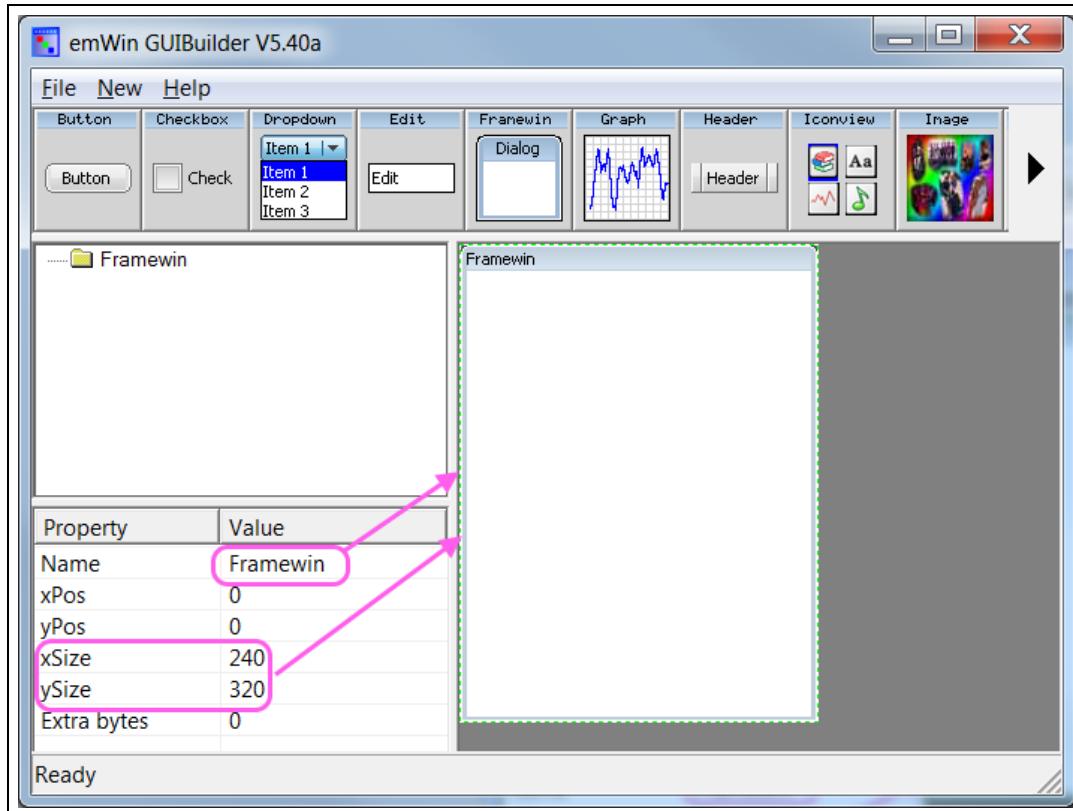


图256. STemWin GUIBuilder



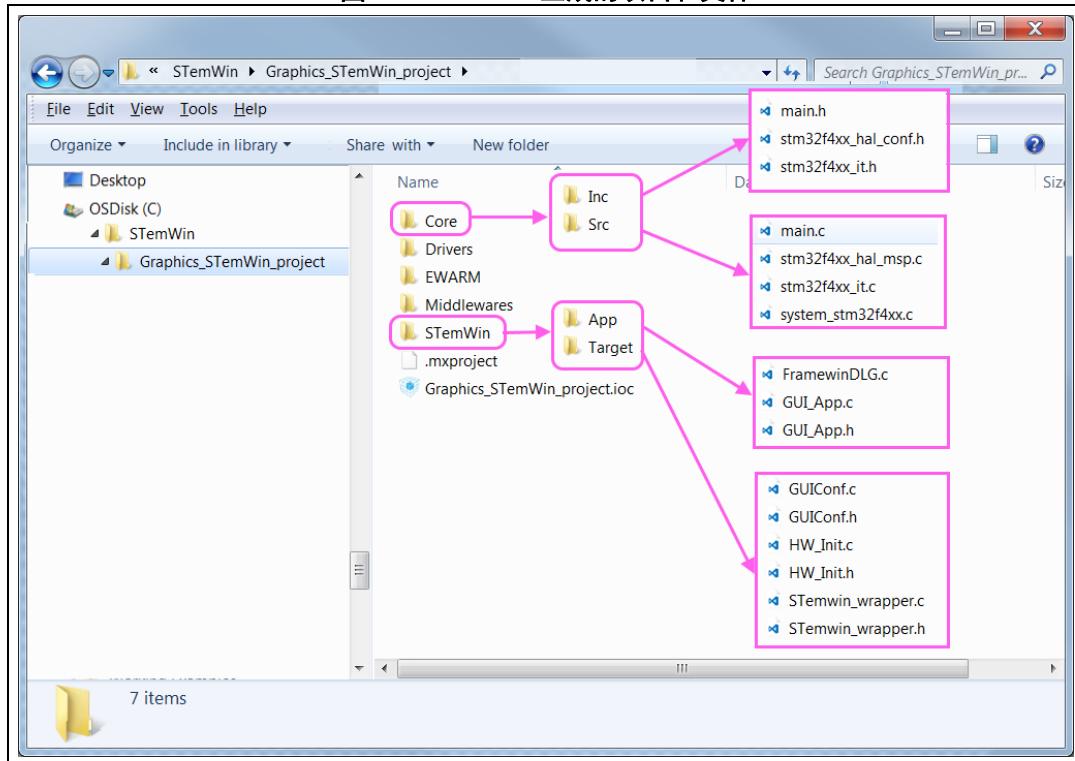
13.5 步骤5：生成嵌入式C项目和更新

点击 可生成项目（参见图 257）。

- 应用主文件会在Core/Inc和Src文件夹下生成。
- STemWin库配置文件会在STemWin/App文件夹下生成。
- 硬件初始化和封装文件会在STemWin/Target文件夹下生成。

将用户代码放在专用部分（即//USER START与//USER END标签之间）可更新生成的代码。

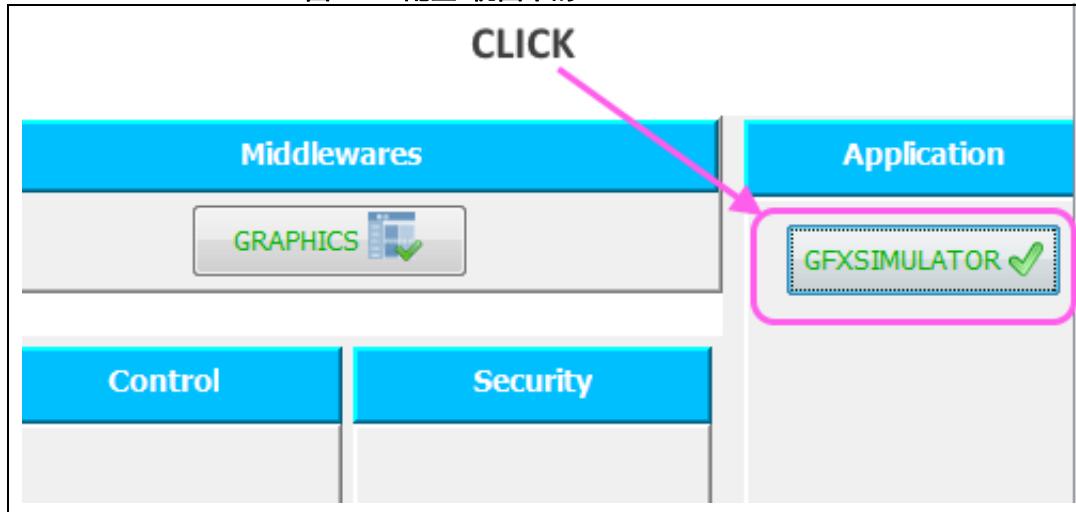
图257. StemWin生成的项目和文件



14 教程8：使用STM32CubeMX图形仿真器

点击配置视图中的GFXSIMULATOR可打开图形仿真器用户界面（参见[图 258](#)）。

图258. “配置”视图中的GFXSIMULATOR



1. 在顶部面板中检查仿真器的当前工作假设和强制配置设置，例如LTDC是否未启用、一些结果是否将显示为NA（不可用）。请参见[图 259](#)。
2. 在中间偏左的面板中调整仿真器参数值，与右侧面板中的当前配置设置进行比较（参见[图 260](#)）。
3. 在底部面板中检查一组受支持用例的结果。
4. 或者点击“导入设置”将当前项目配置更新为仿真器设置，以获得相同的性能结果。
 - 弹出一条确认消息。
 - 在适当的情况下，会弹出一条警告消息，其中包含无法导入的设置（未使用的外设、时钟频率值超出范围...）。
5. 关闭“图形仿真器”窗口会返回到STM32CubeMX配置视图。

图259. “图形仿真器”用户界面

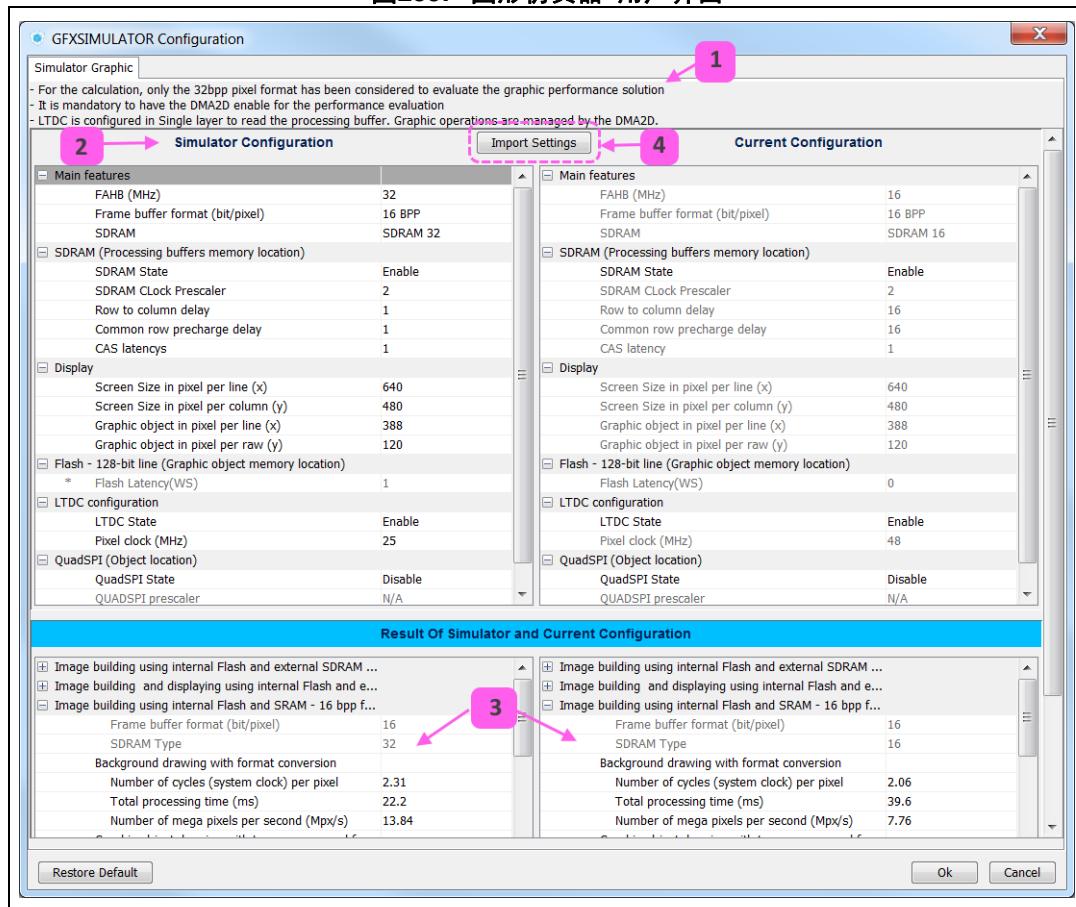
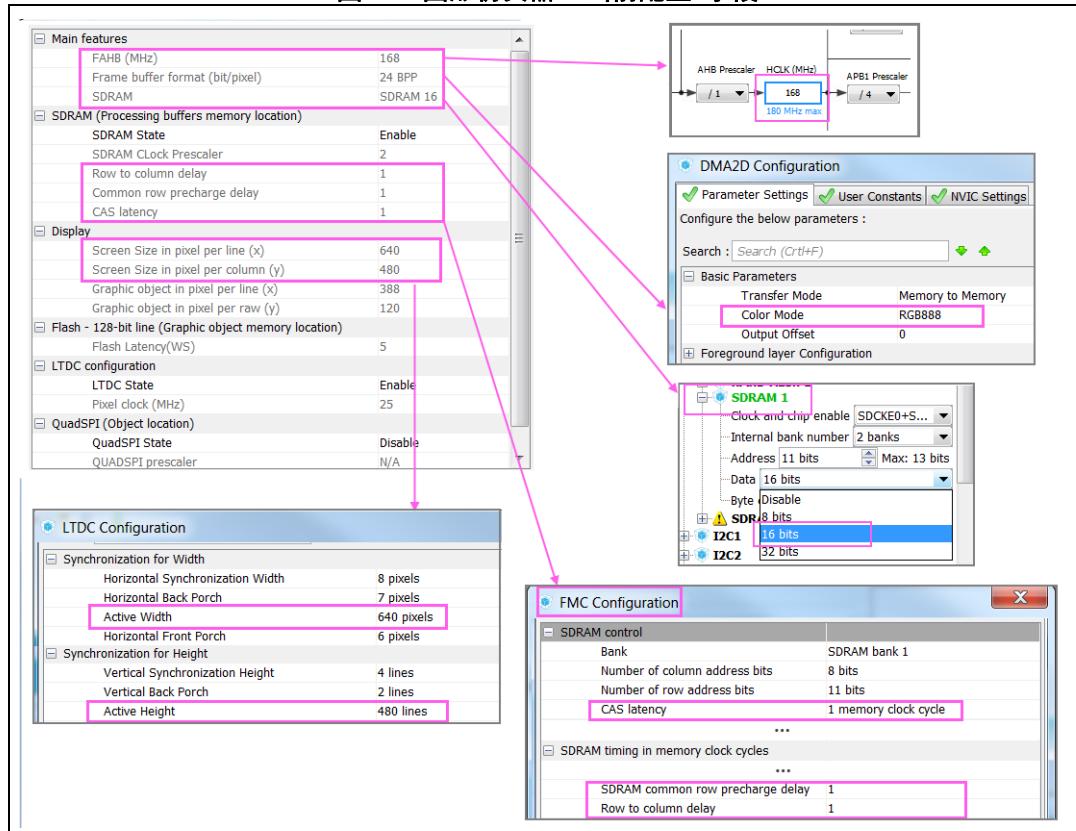


图260. 图形仿真器 -“当前配置”字段



15 FAQ

15.1 在“引脚布局配置”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。

15.2 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。

15.3 为什么芯片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。

15.4 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：

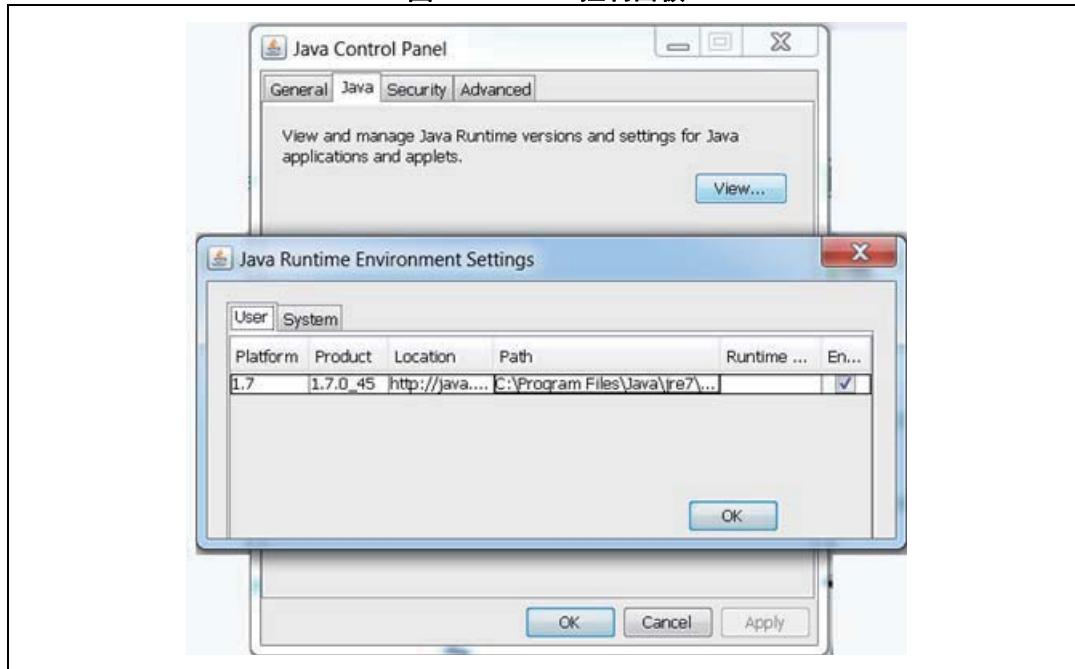
- 删除所有Java™安装版本，然后只重新安装一个版本（32位或64位）（Java 7 update 45或更新版本）。
- 保留32位和64位安装版本，但确保64位版本至少为Java 7 update 45。

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 261）。

图261. Java™控制面板



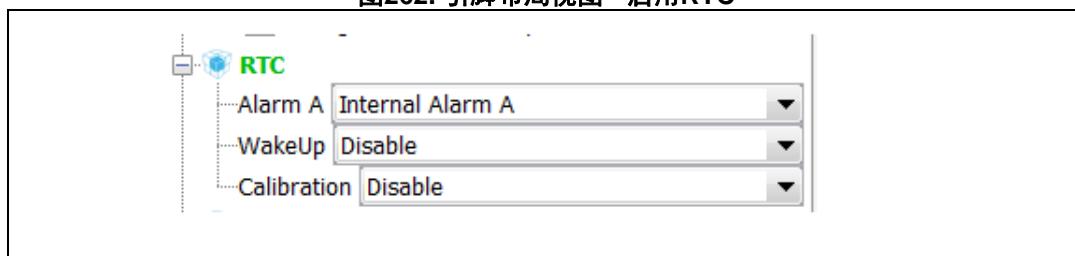
还可输入MS-DOS命令“`java -version`”以检查最新安装的Java版本（在此调用的Java程序是在C:\Windows\System32下安装的程序的副本）：

```
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)
```

15.5 为何RTC复用器在时钟树视图中仍无效？

要启用RTC复用器，用户应在引脚布局视图中启用RTC外设，如下图所示。

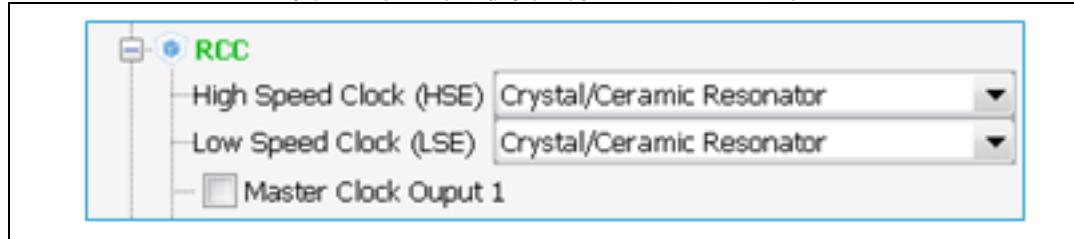
图262. 引脚布局视图 - 启用RTC



15.6 如何选择LSE和HSE作为时钟源并更改频率？

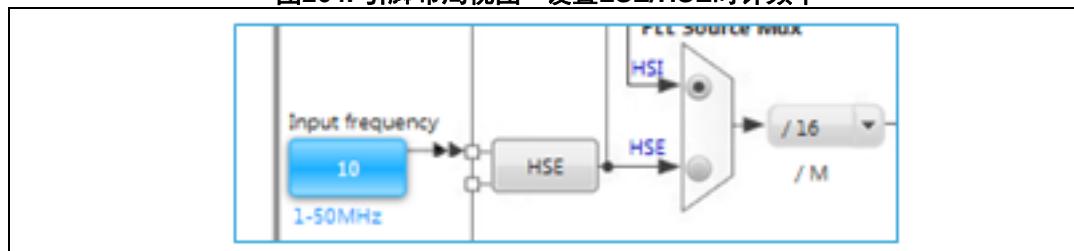
在引脚布局视图中依此对RCC进行配置后，LSE和HSE时钟会立即生效。相关示例，请参见[图 263](#)。

图263. 引脚布局视图 - 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 264](#)。

图264. 引脚布局视图 - 设置LSE/HSE时钟频率



15.7 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）。”

15.8 以太网配置：为什么有时候我不能指定DP83848或LAN8742A？

对于大部分系列，STM32CubeMX将根据已选的以太网模式调整可行的PHY组件驱动程序列表：

- 如果选择以太网MII模式，用户将能够在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，用户将能够在LAN8742A组件驱动程序与“用户Phy”之间选择。

如果选择“用户Phy”，用户需要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：块间依赖性
- 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）。

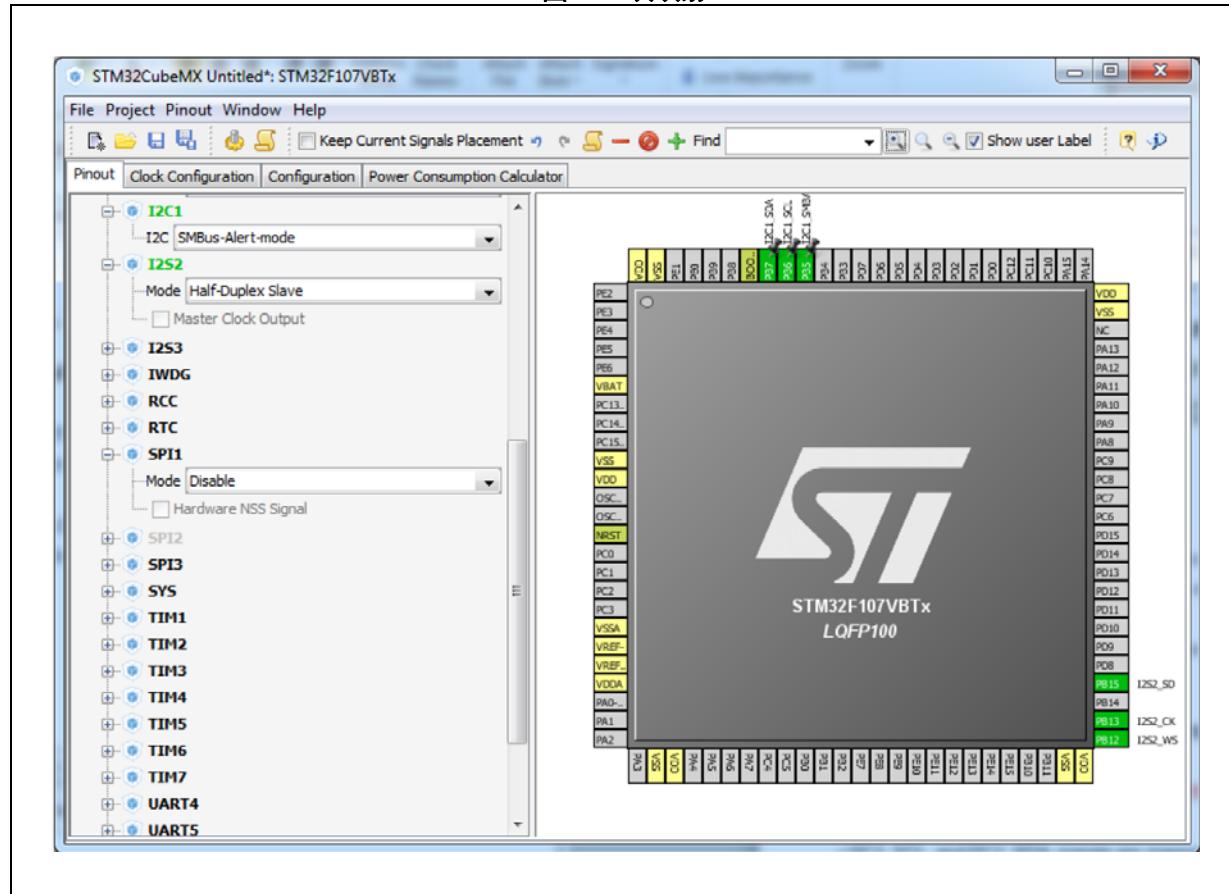
清除引脚信号时，会同时取消该模式需要使用的所有引脚/信号的映射，引脚变回灰色。

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 265）。
- I2C1外设模式设为SMBus-Alert模式。

图265. 块映射

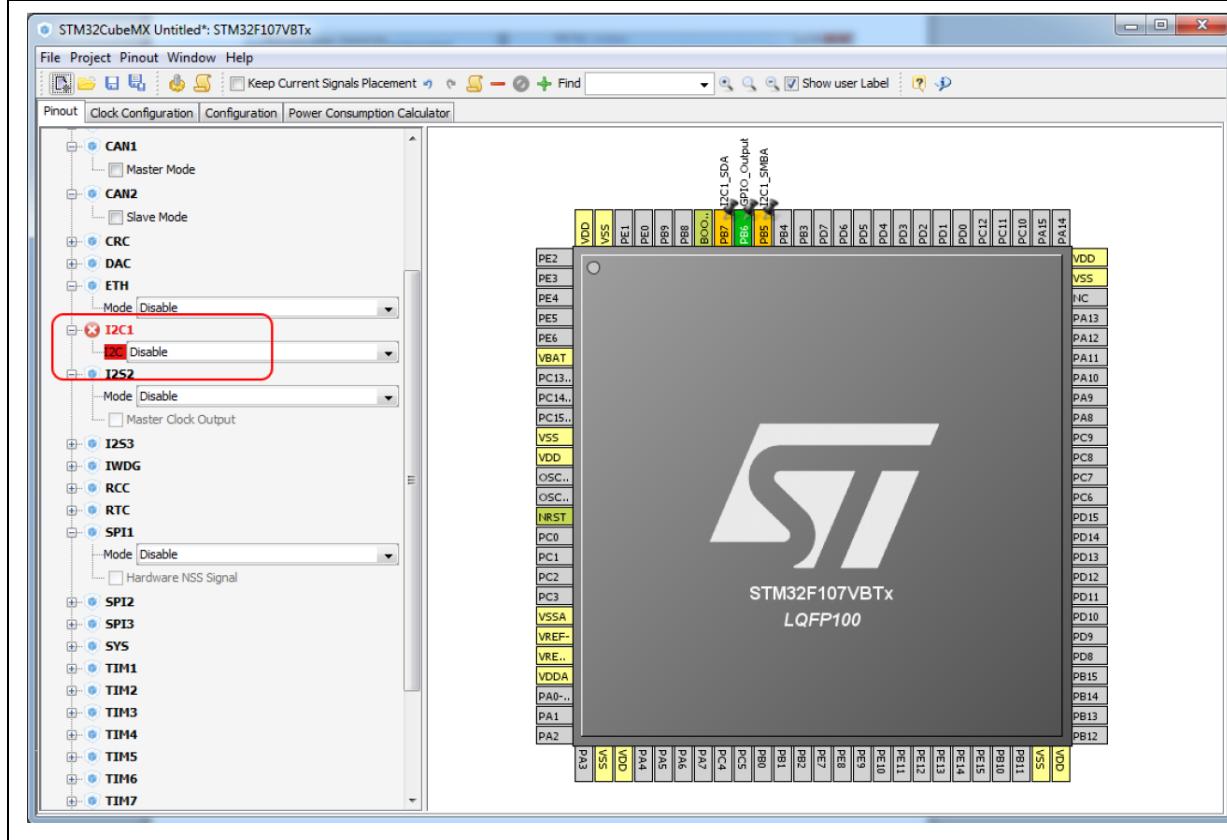


STM32F107x MCU块重新映射示例

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 266）。

图266. 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 267和图 268）。

图267. 块重新映射 - 示例1

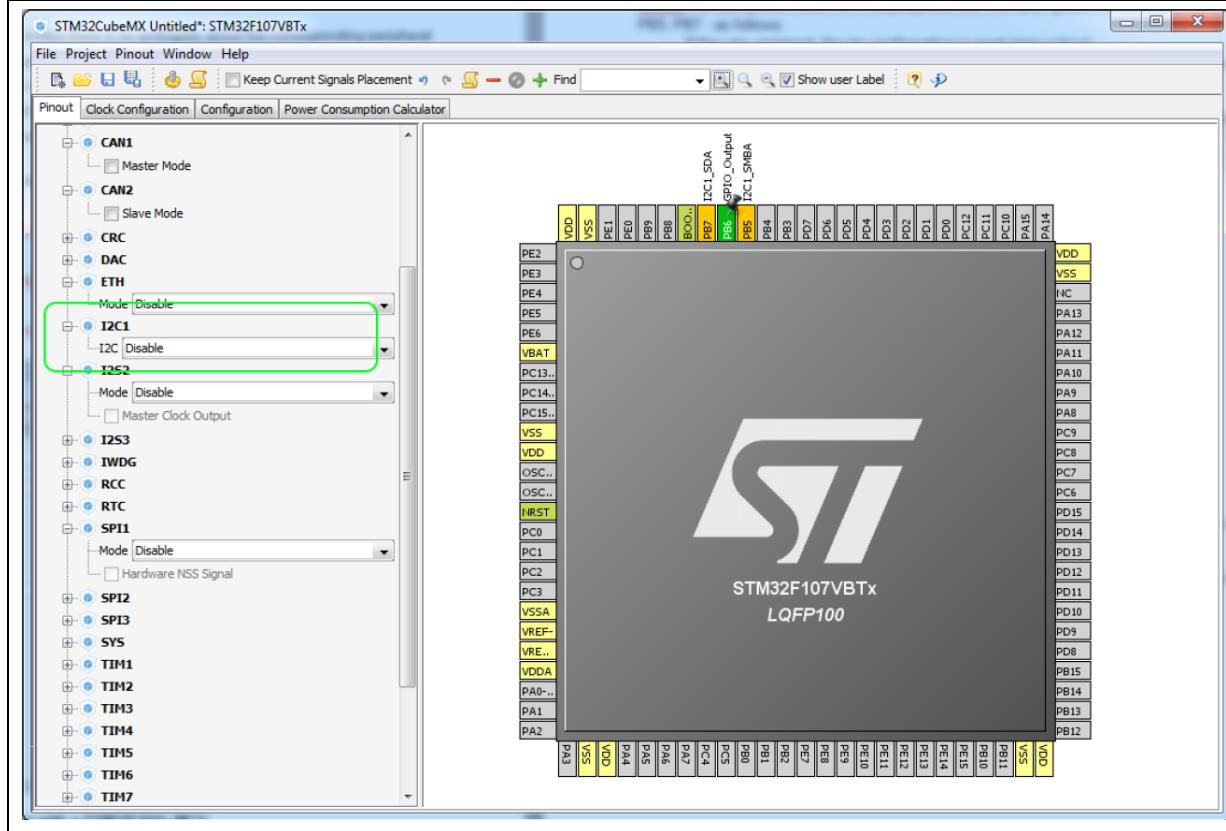
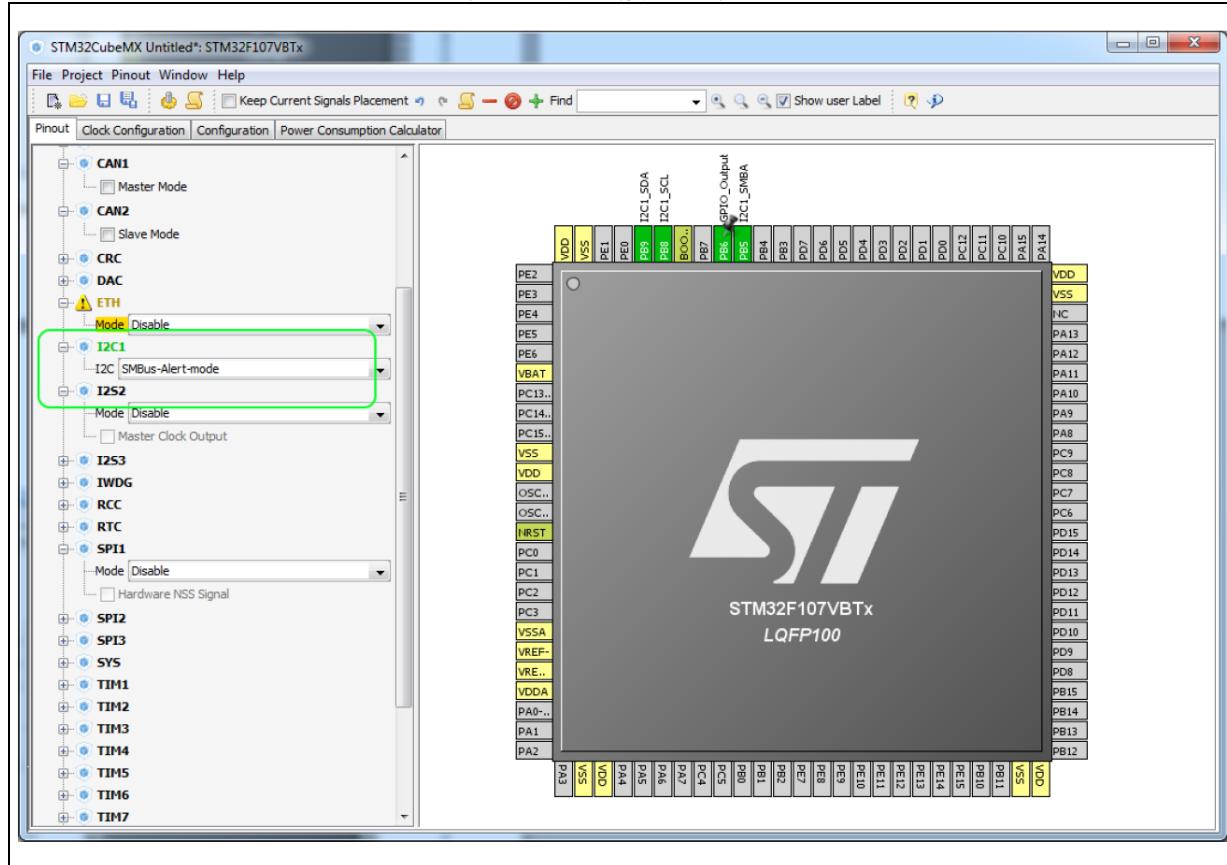


图268. 块重新映射 - 示例2



A.2 块间依赖性

在芯片视图中，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 269）。

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 270）。

（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269. 块相关性 - SPI信号分配给PB3/4/5

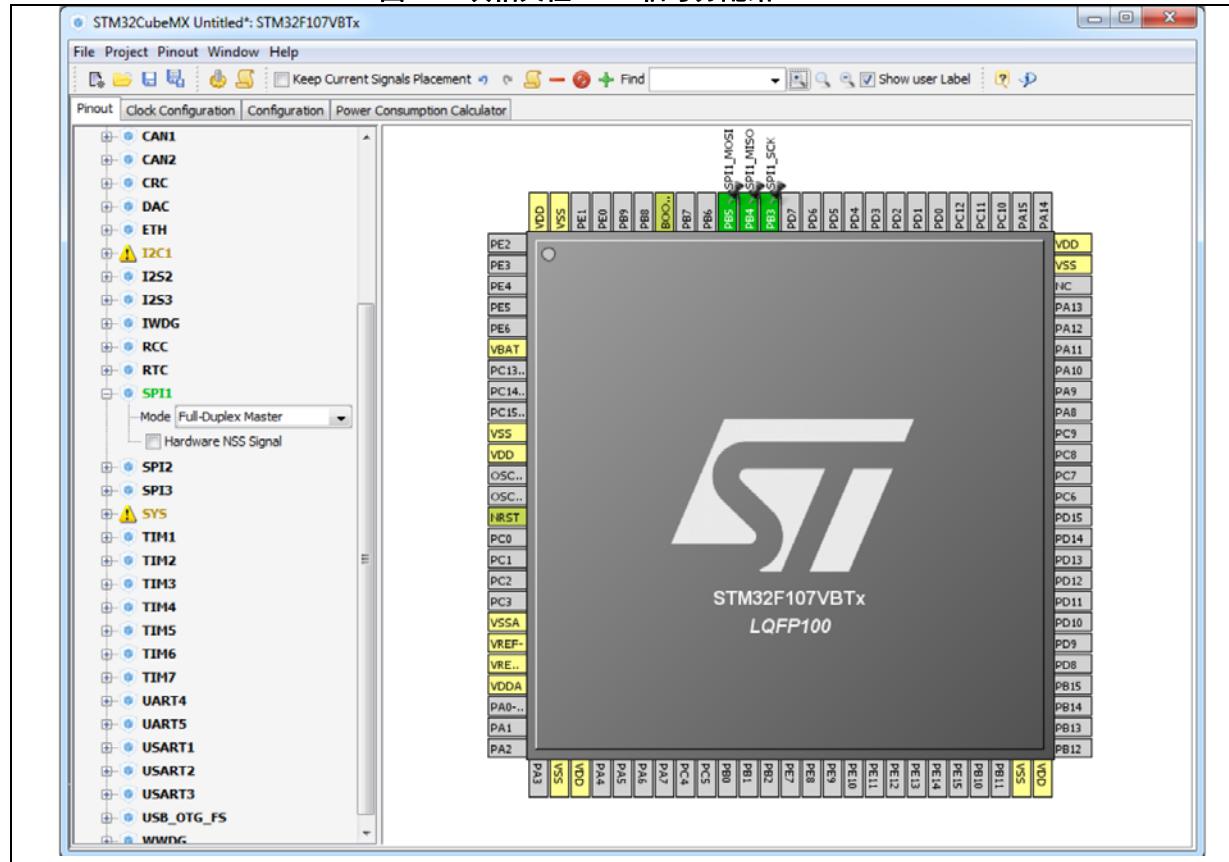
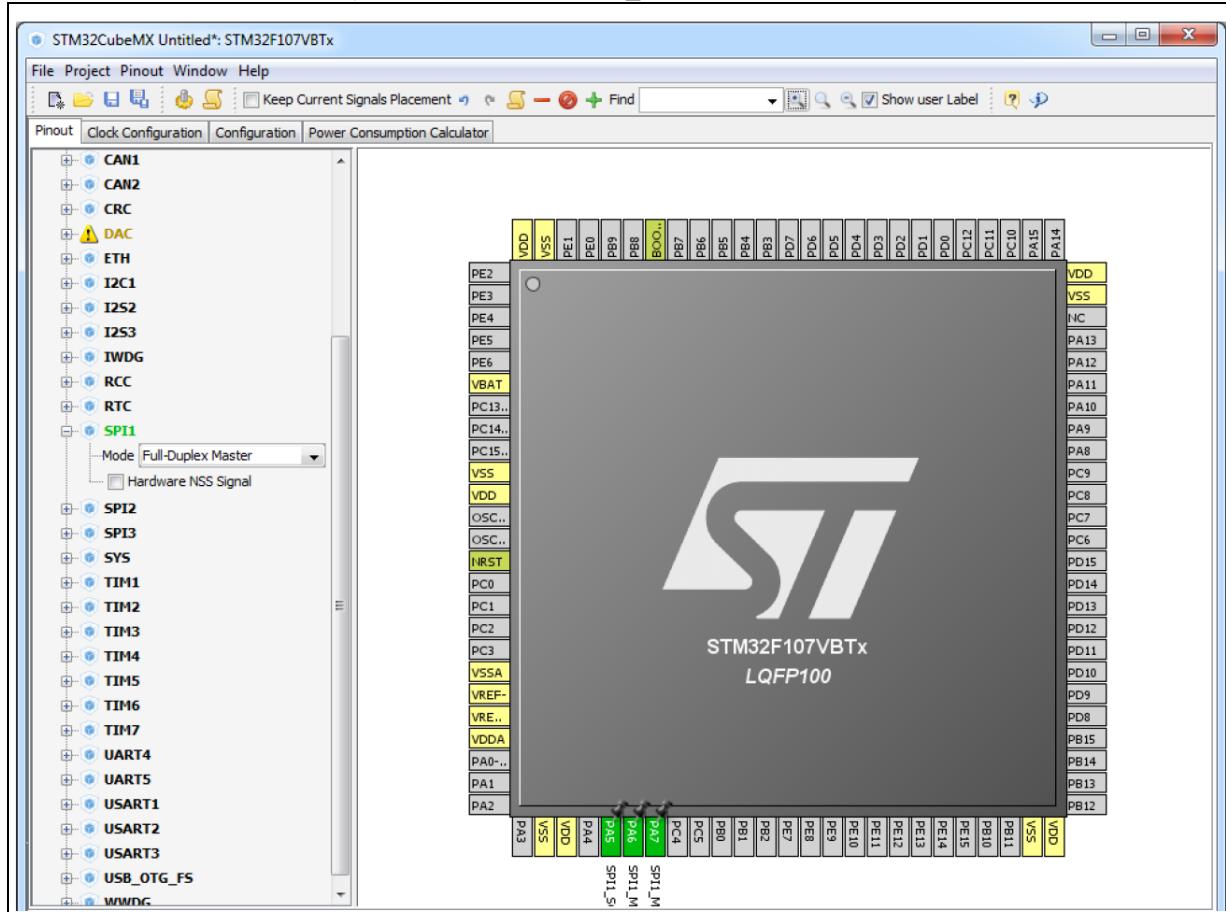


图270. 块相关性 - SPI1_MISO功能分配给PA6



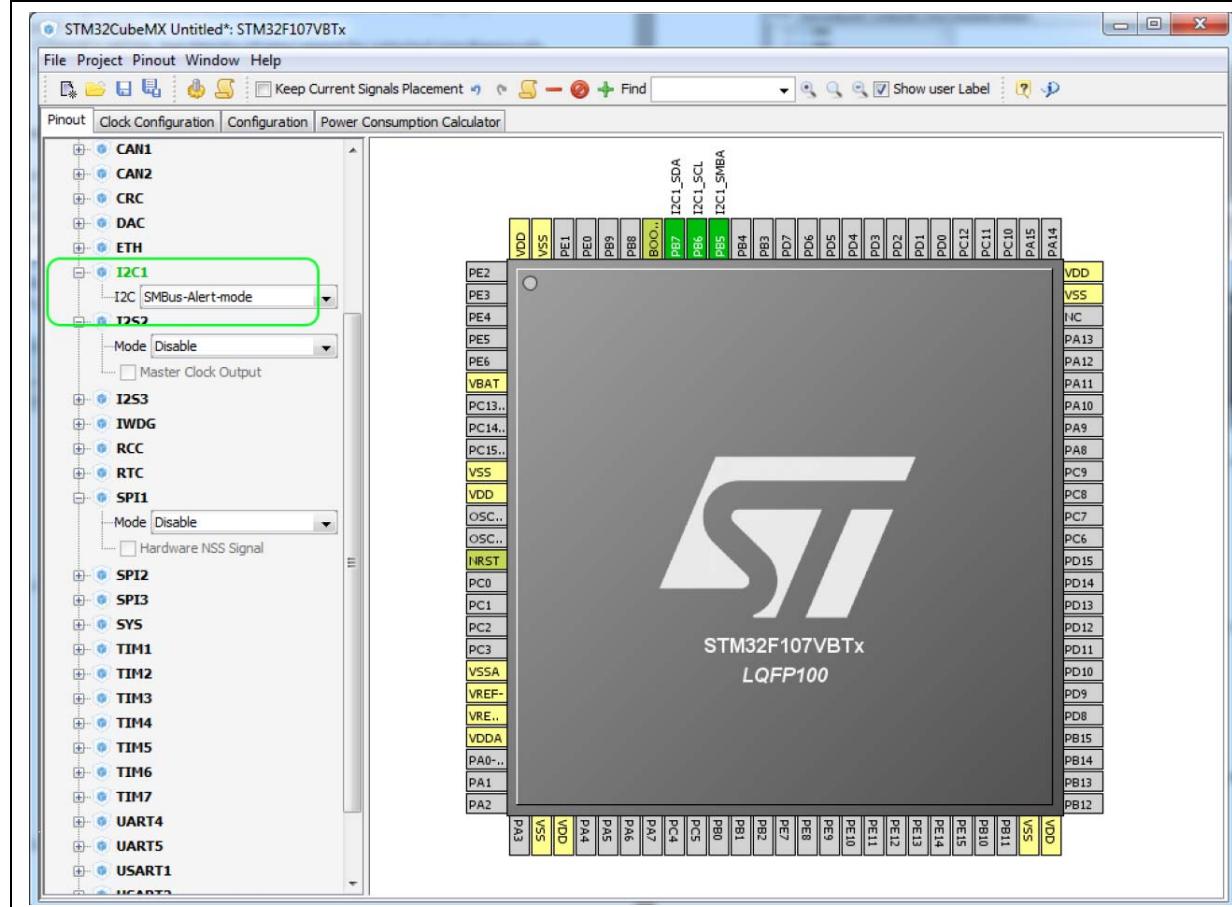
A.3 一个块 = 一种外设模式

如果已在芯片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 271中的外设树）。

图271. 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）

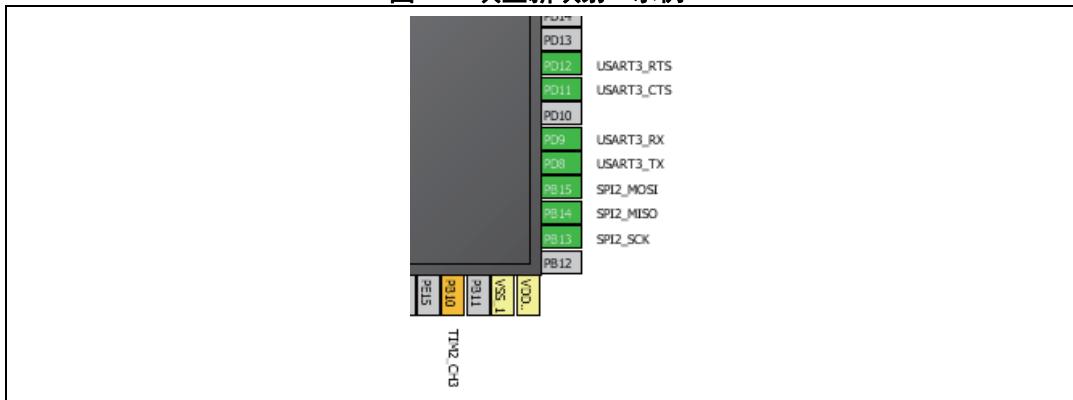
要配置外设模式，STM32CubeMX会选择引脚块，并将各模式信号分配给此块中的一个引脚。这样它便会搜索模式可映射到的第一个空闲块。

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。如果未找到替代块，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 272）。

图272. 块重新映射 - 示例2



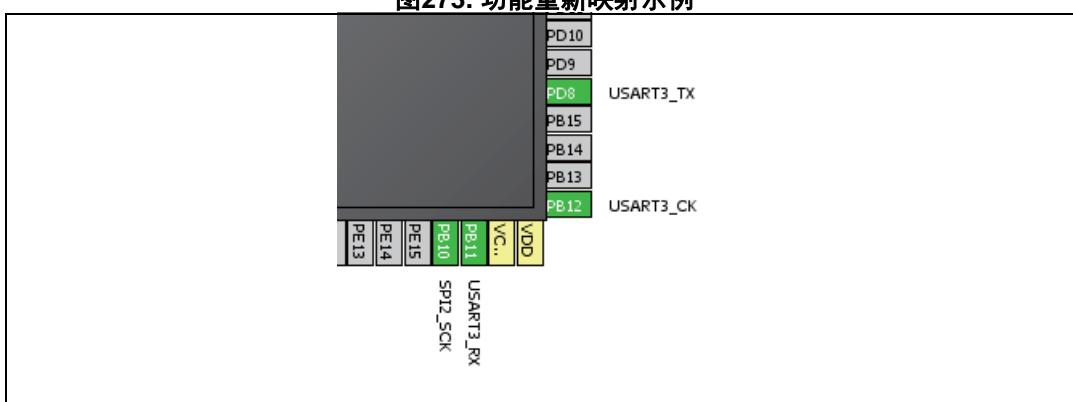
A.5 功能重新映射

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 273）。

图273. 功能重新映射示例



A.6 块转移（仅适用于STM32F10x，且“保留当前信号布置”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 274）。

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 275）。

图274. 未应用块转移

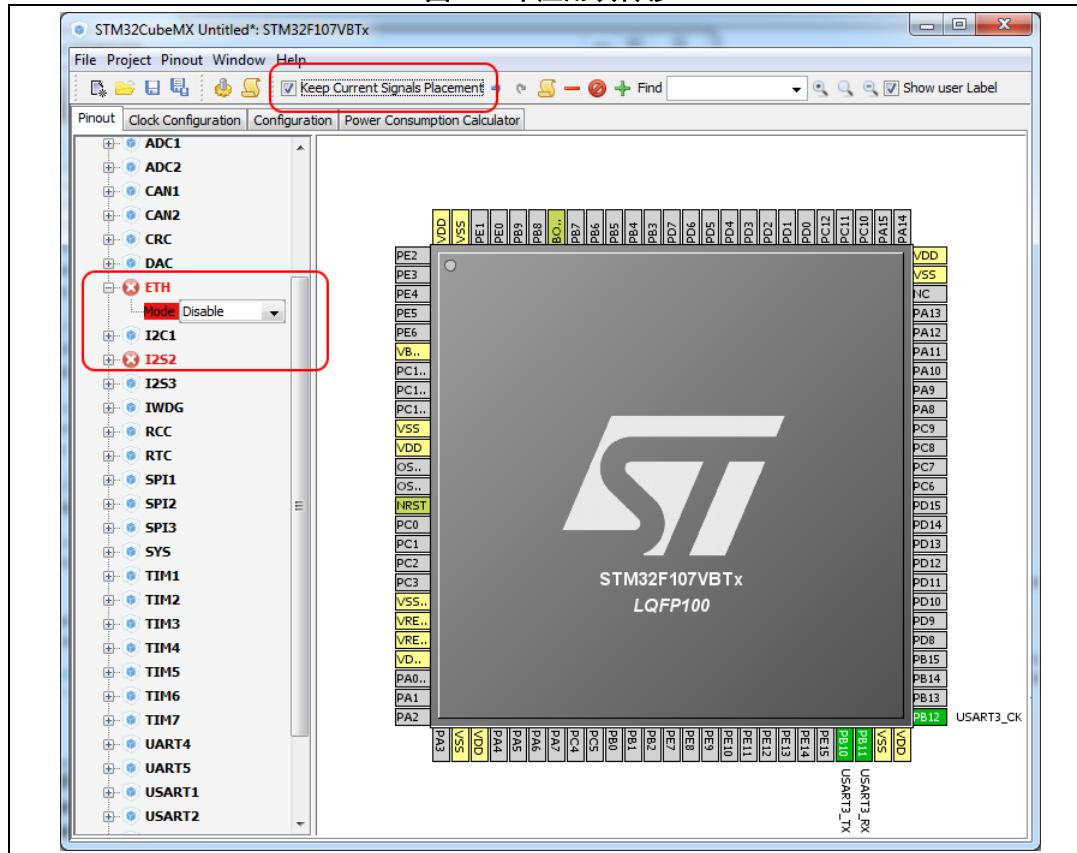
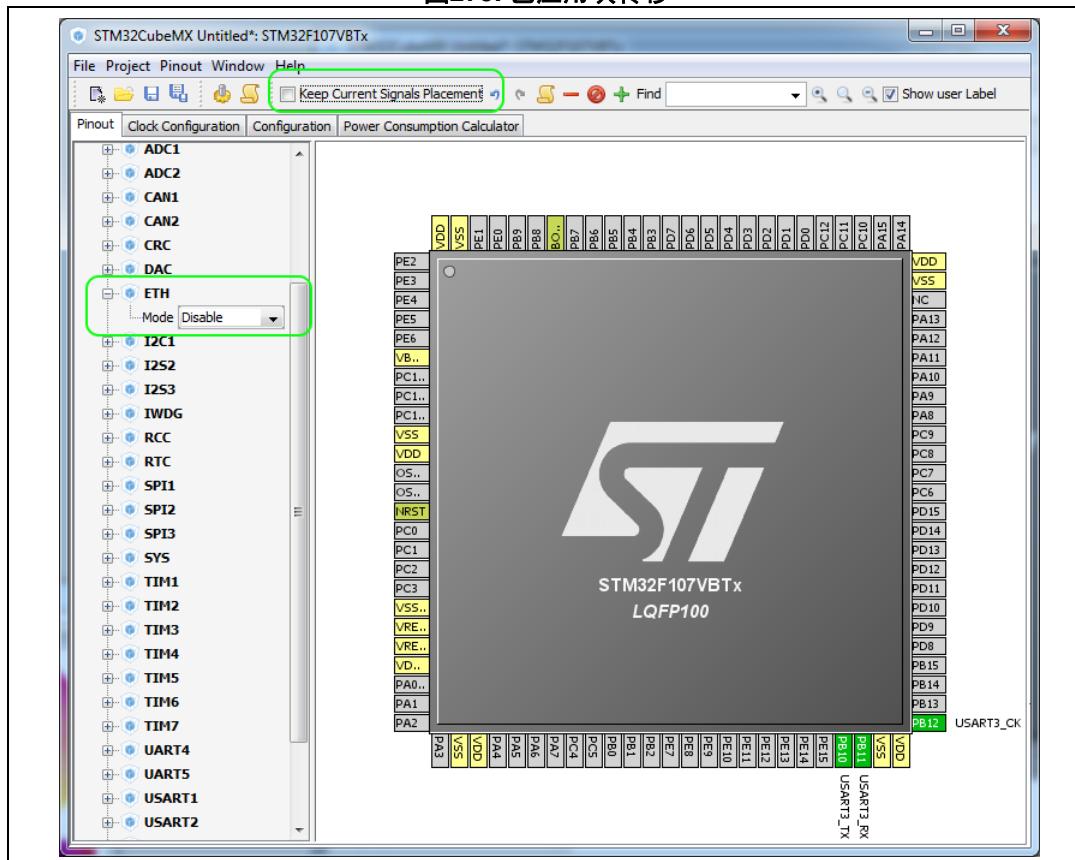


图275. 已应用块转移



A.7 设置或清除外设模式

“外设”面板和芯片视图相互关联：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）可通过芯片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

STM32CubeMX代码生成设计选择和限制

本节总结了STM32CubeMX的设计选择和限制。

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

B.3.1 概述

STM32CubeMX不支持在中间件协议栈文件中插入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 */
}

```

```
for(;;)
{
    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。

USBID、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FatFsSD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

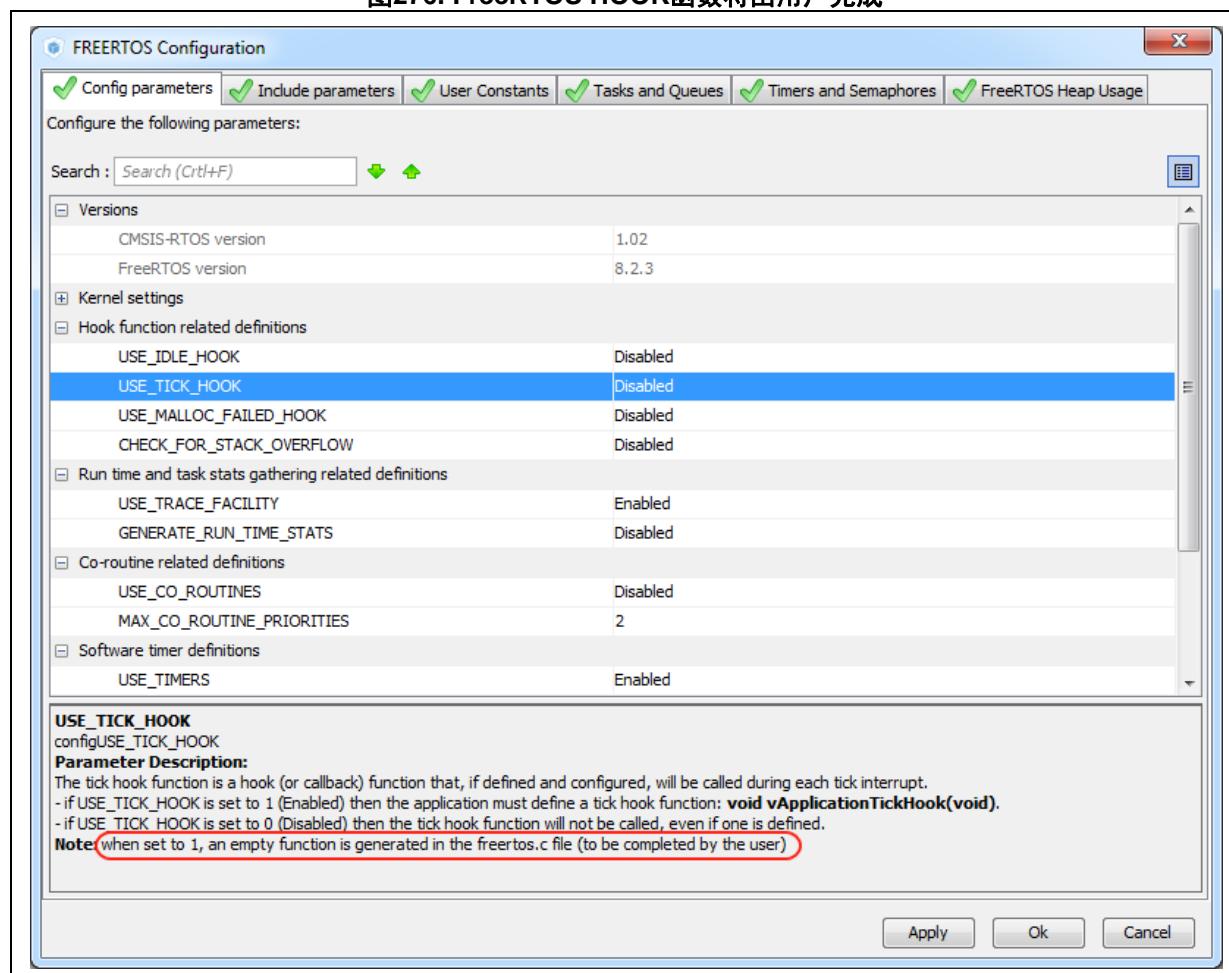
FreeRTOS是适用于微控制器的免费实时嵌入式操作系统。

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 276）。

图276. 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*wip.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不支持在协议栈文件中插入用户C代码。但一些LwIP用例需要修改协议栈文件（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 278）。

必须禁用DHCP才能配置静态IP地址。

图277. LwIP 1.4.1配置

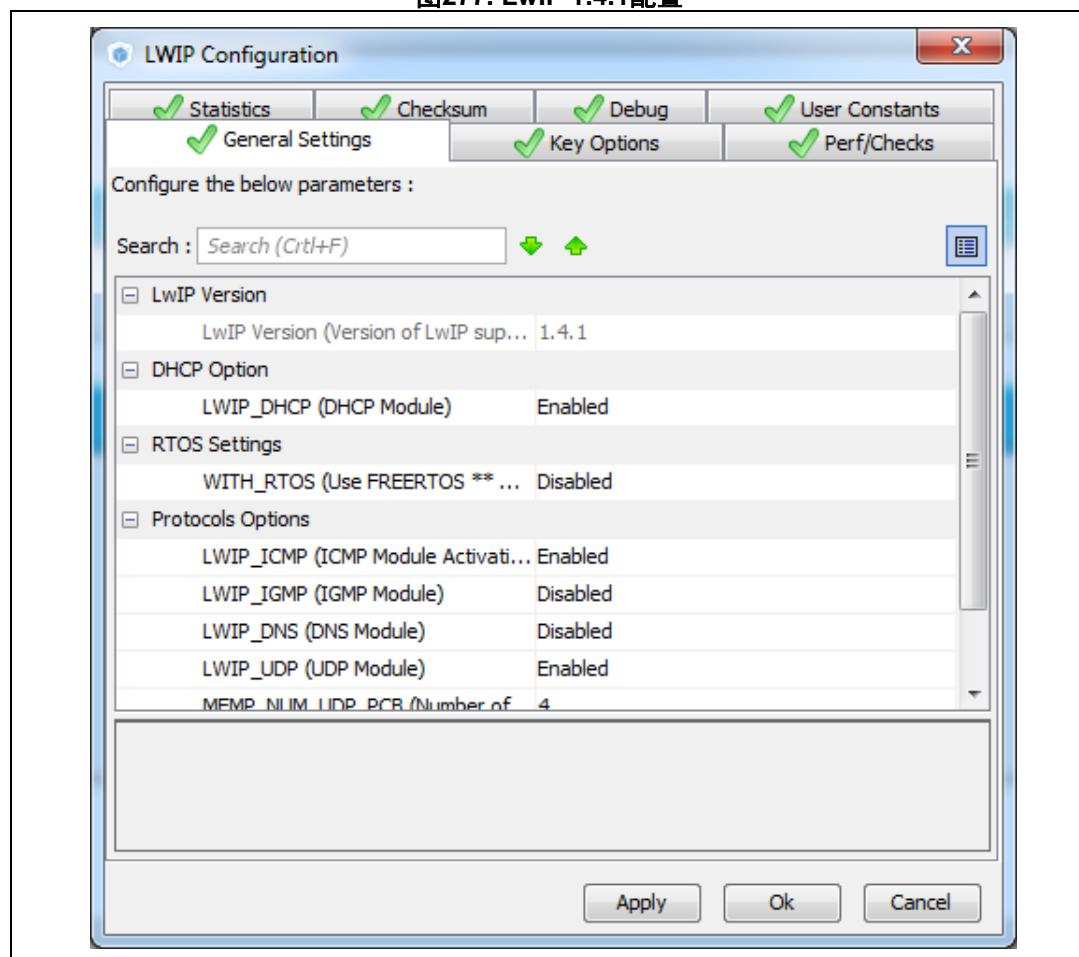
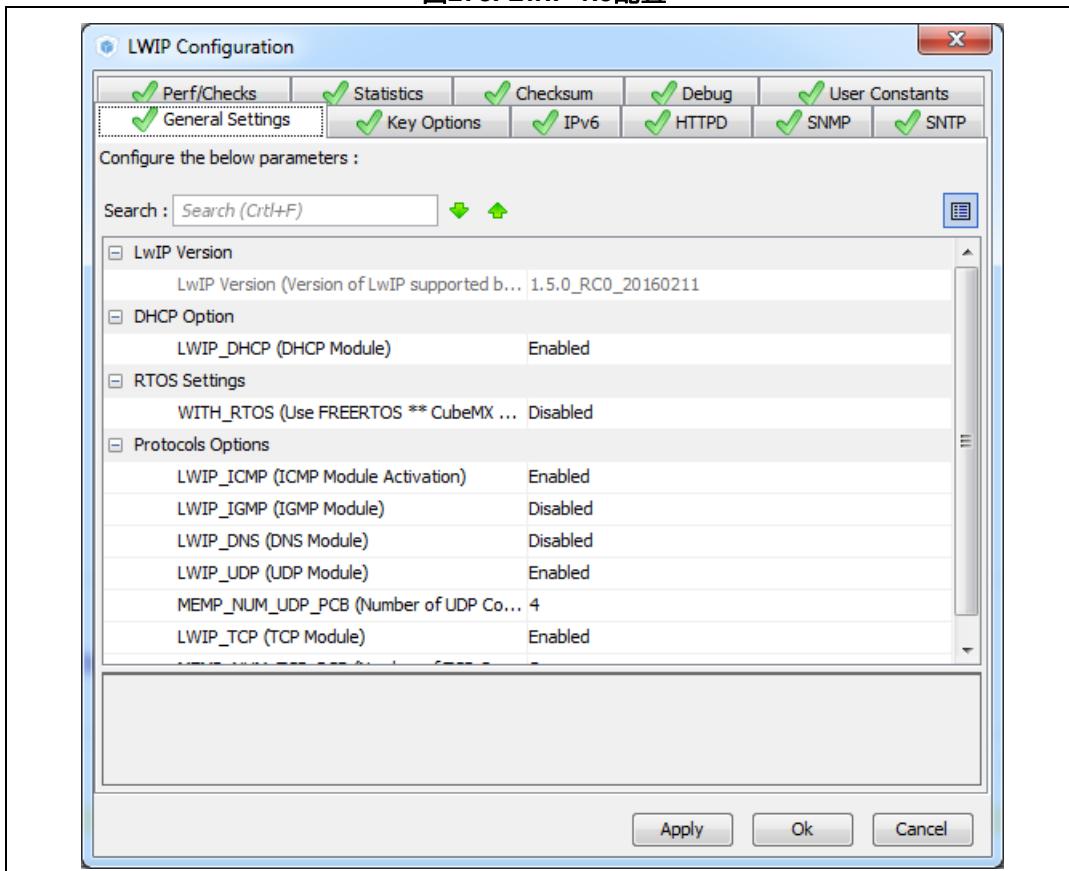


图278. LwIP 1.5配置



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。

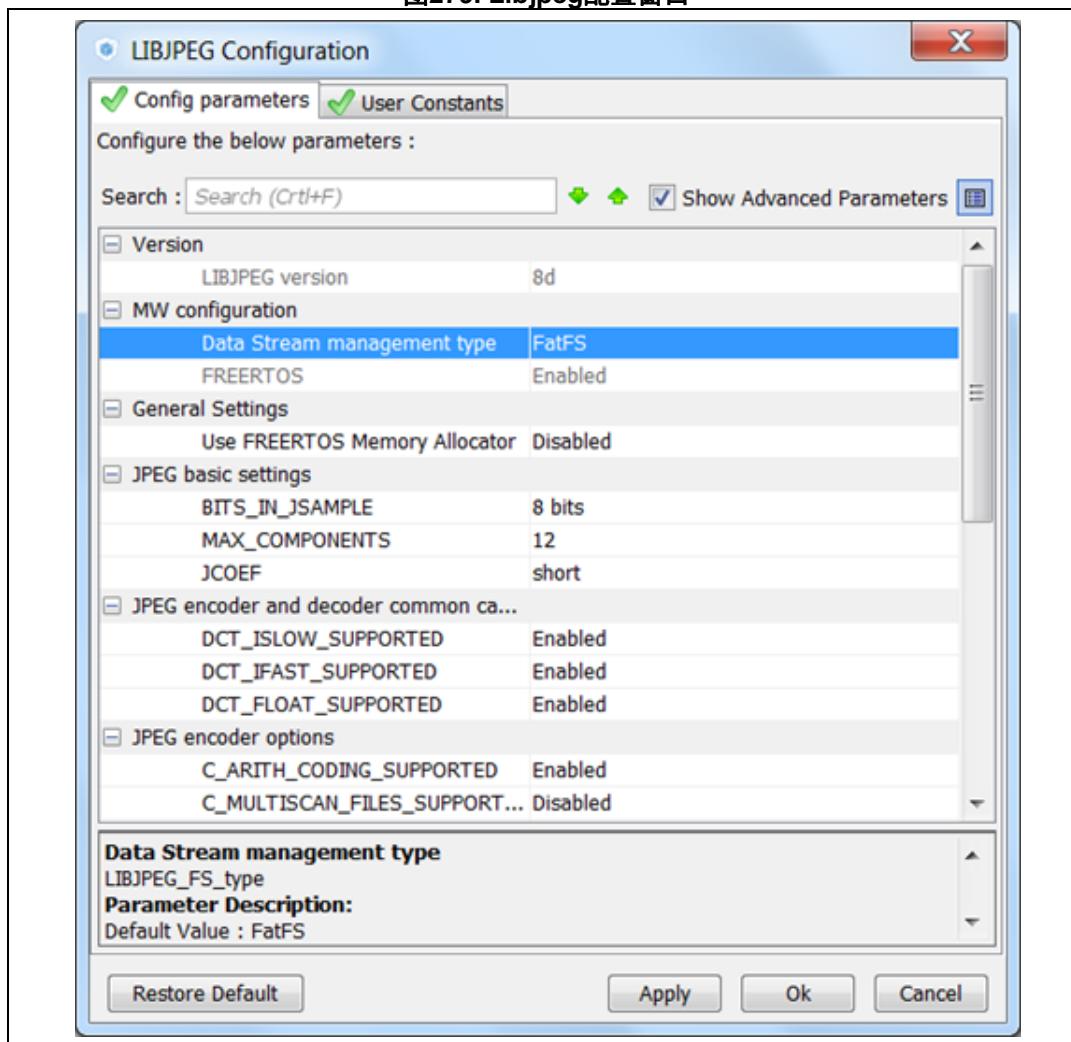
B.3.7 Libjpeg

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279. 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代码尺寸。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 280: Mbed TLS \(无LwIP\)](#)）。

如果使用LwIP堆栈，还必须启用FreeRTOS（参见 [图 281: 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 282：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280. Mbed TLS（无LwIP）

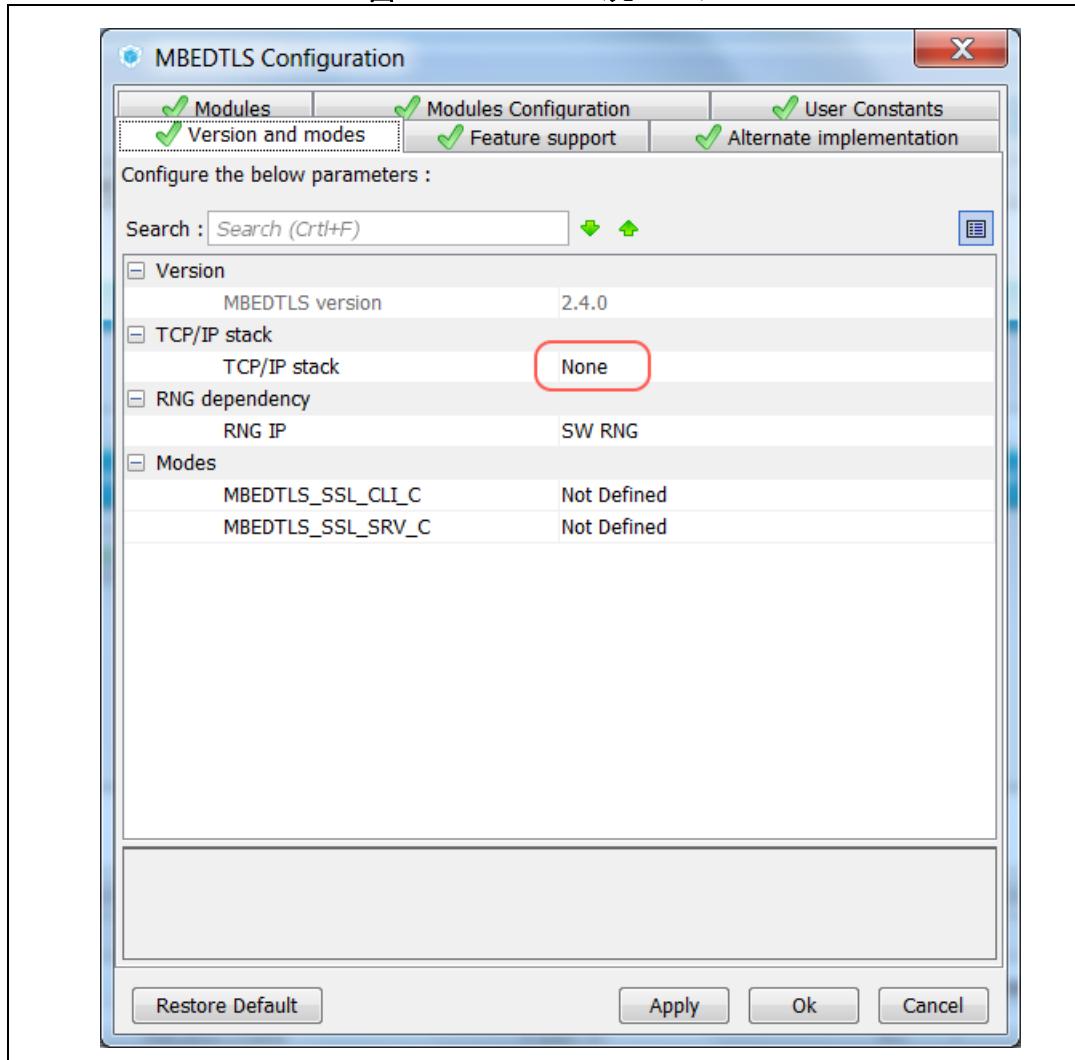


图281. Mbed TLS (有LwIP和FreeRTOS)

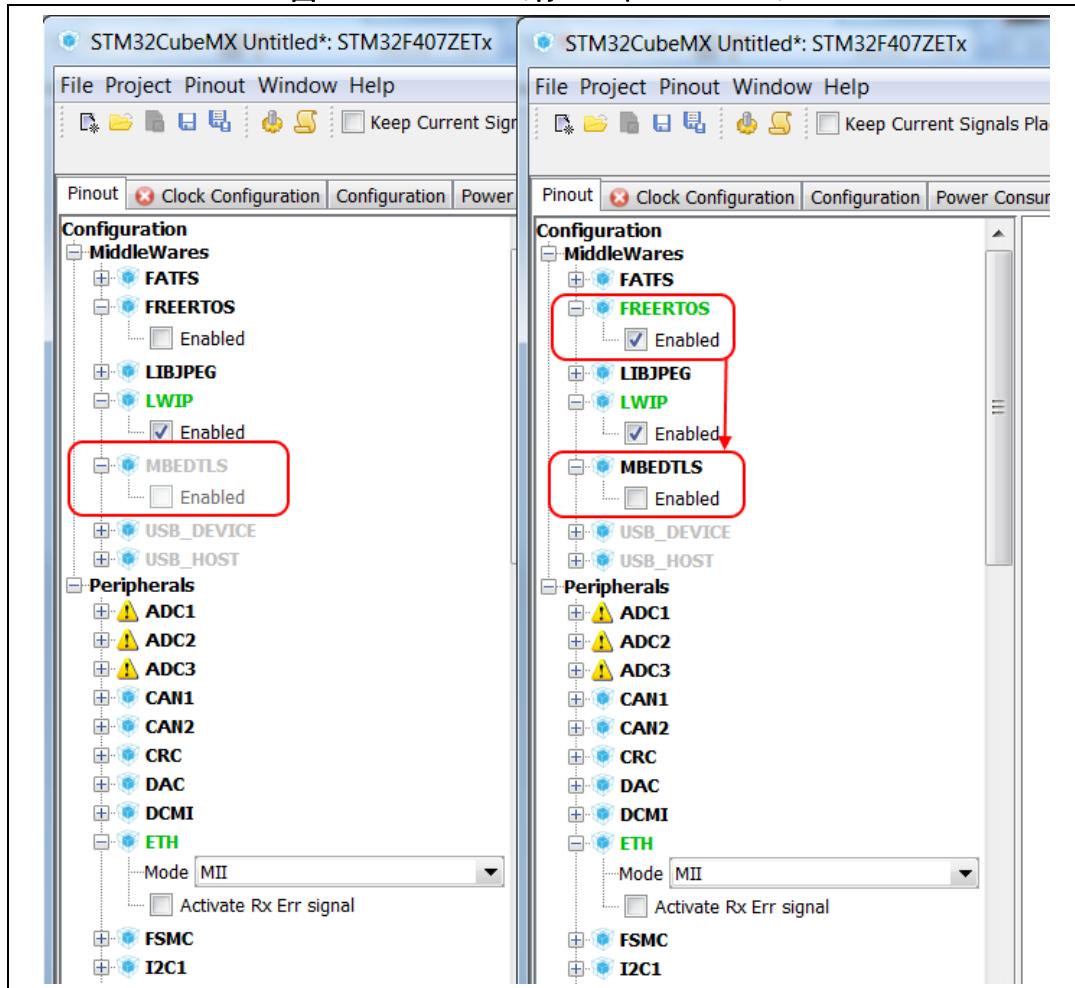
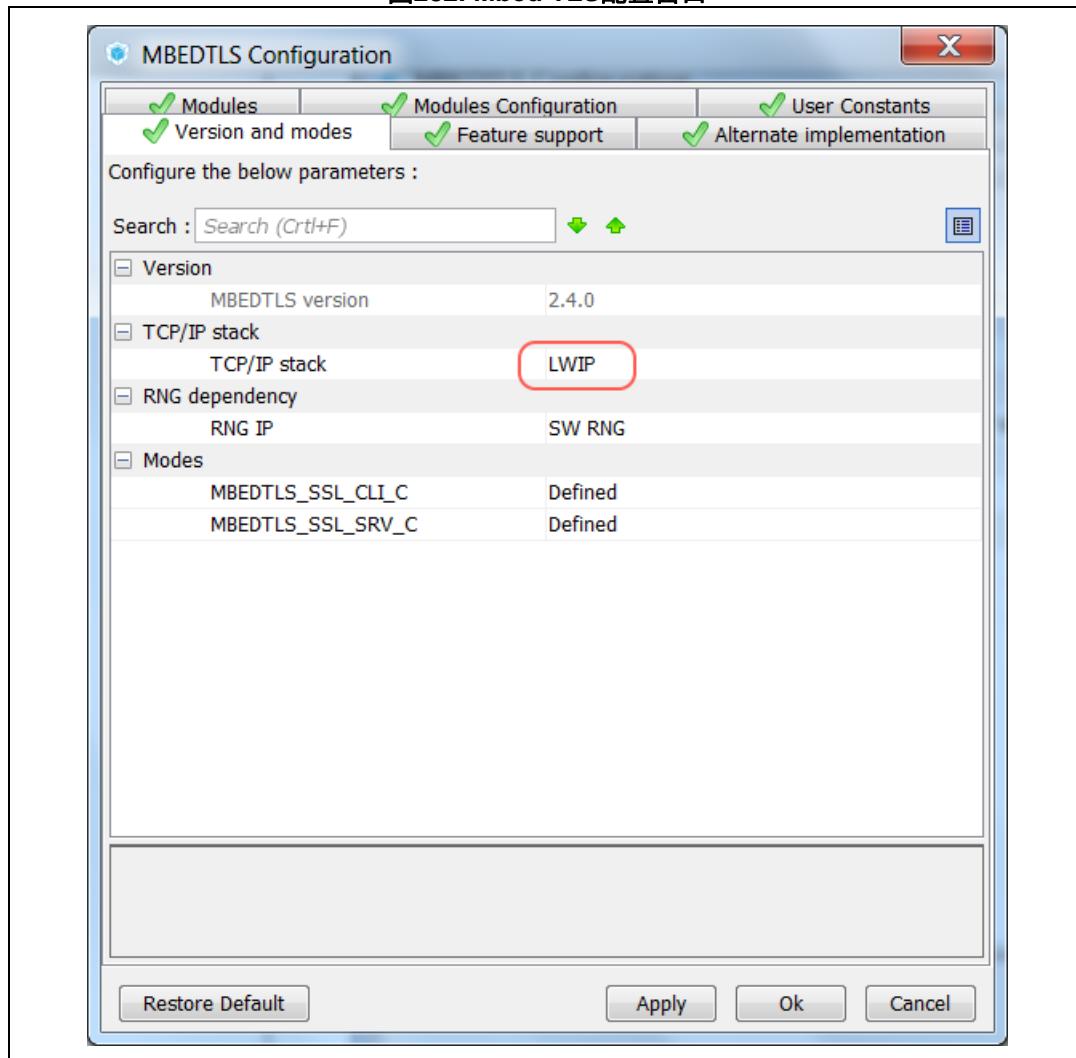


图282. Mbed TLS配置窗口



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 283：启用 TouchSensing 外设](#)、[图 284：触摸感应传感器选择面板](#)和[图 285：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283. 启用TouchSensing外设

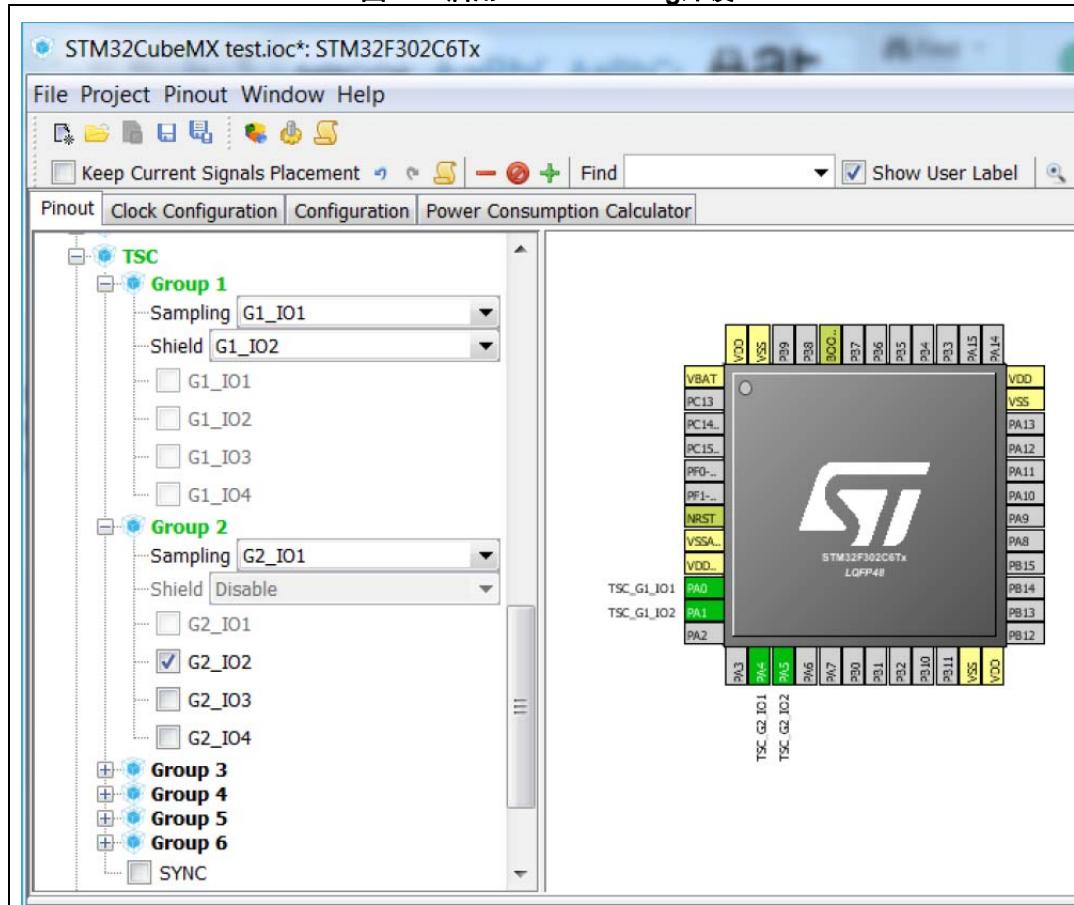


图284. 触摸感应传感器选择面板

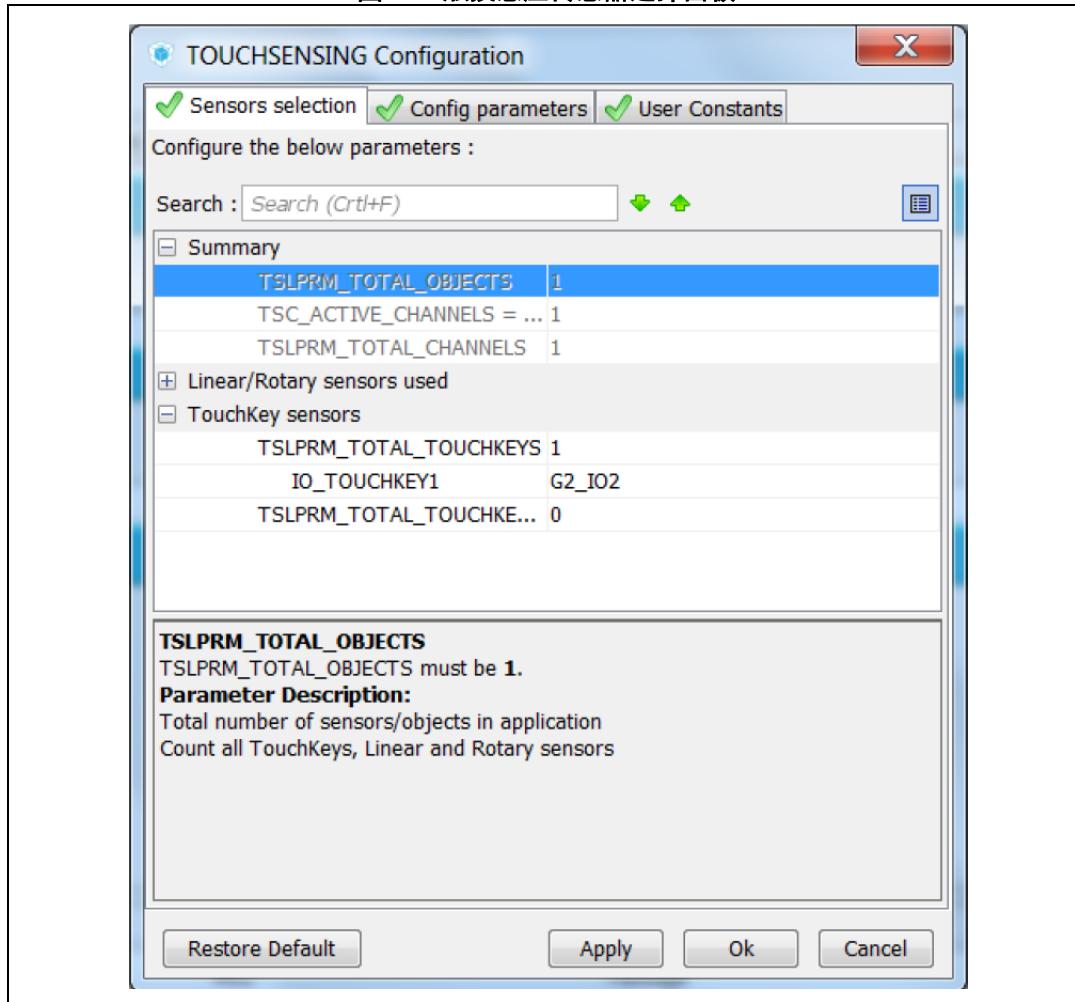
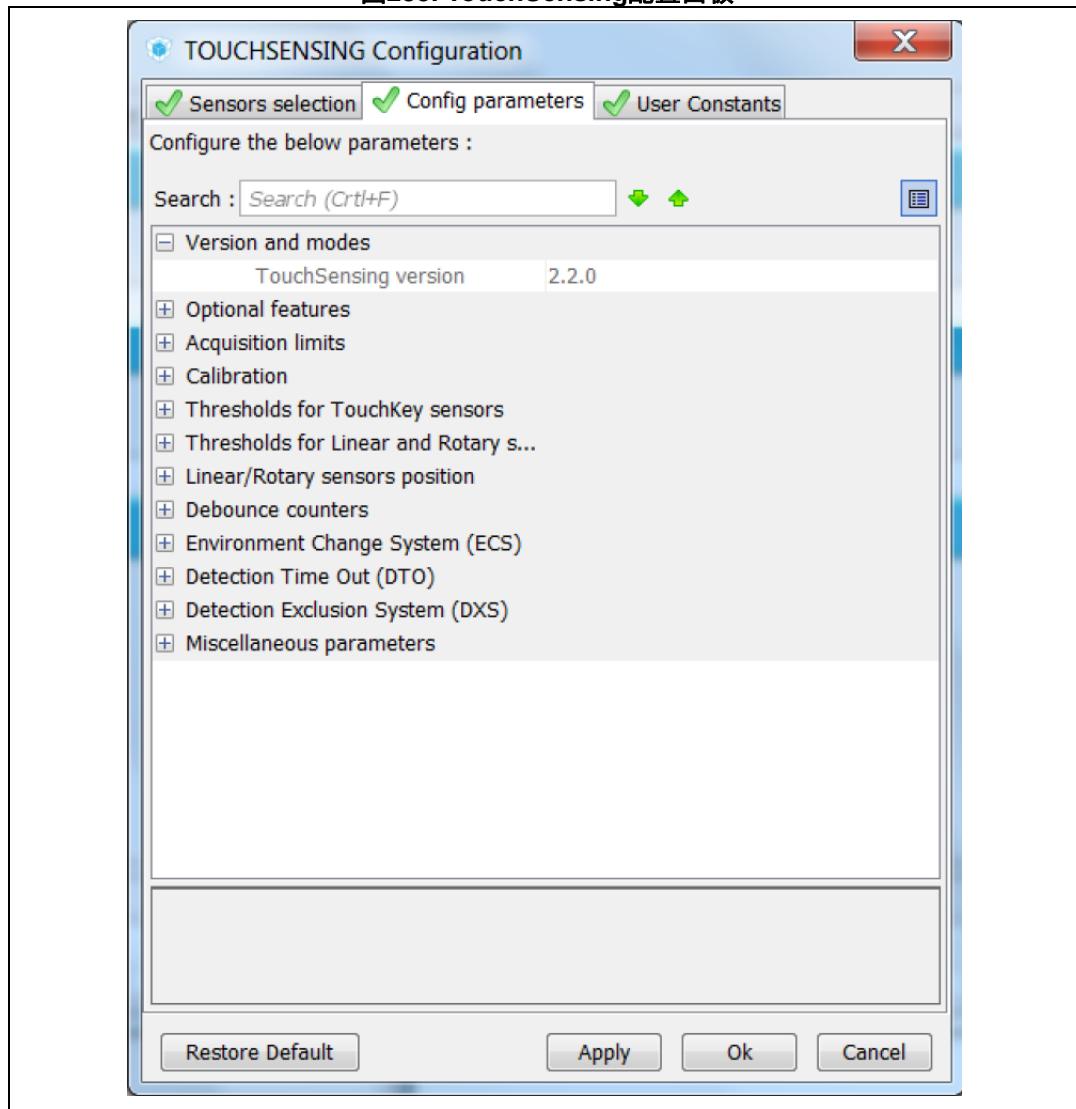


图285. TouchSensing配置面板



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 图形

意法半导体已为STM32产品选择STemWin框架，请参见专门介绍如何使用STM32CubeMX为该框架创建项目的教程部分。

生成的项目沿用[图 286](#)中显示的嵌入式软件架构，包含的硬件初始化文件和封装文件可适应图形协议栈的特殊性。

图286. 图形应用架构



图形堆栈配有特定的UI设计工具，可通过STM32CubeMX用户界面调用此工具执行高级设计工作：

- **GUIBuilder**（用于STemWin框架）

所有通过工具生成的C、C++或其他文件都保存在STM32CubeMX项目文件夹中。

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 STM32L162QDH6）

- Z = 144 个引脚

- I = 176 (+25) 个引脚

- B = 208个引脚（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存储器

- B = 128 KB Flash存储器

- C = 256 KB Flash存储器

- D = 384 KB Flash存储器

- E = 512 KB Flash存储器

- F = 768 KB Flash存储器

- G = 1024 KB Flash存储器

- I = 2048 KB Flash存储器

- 封装

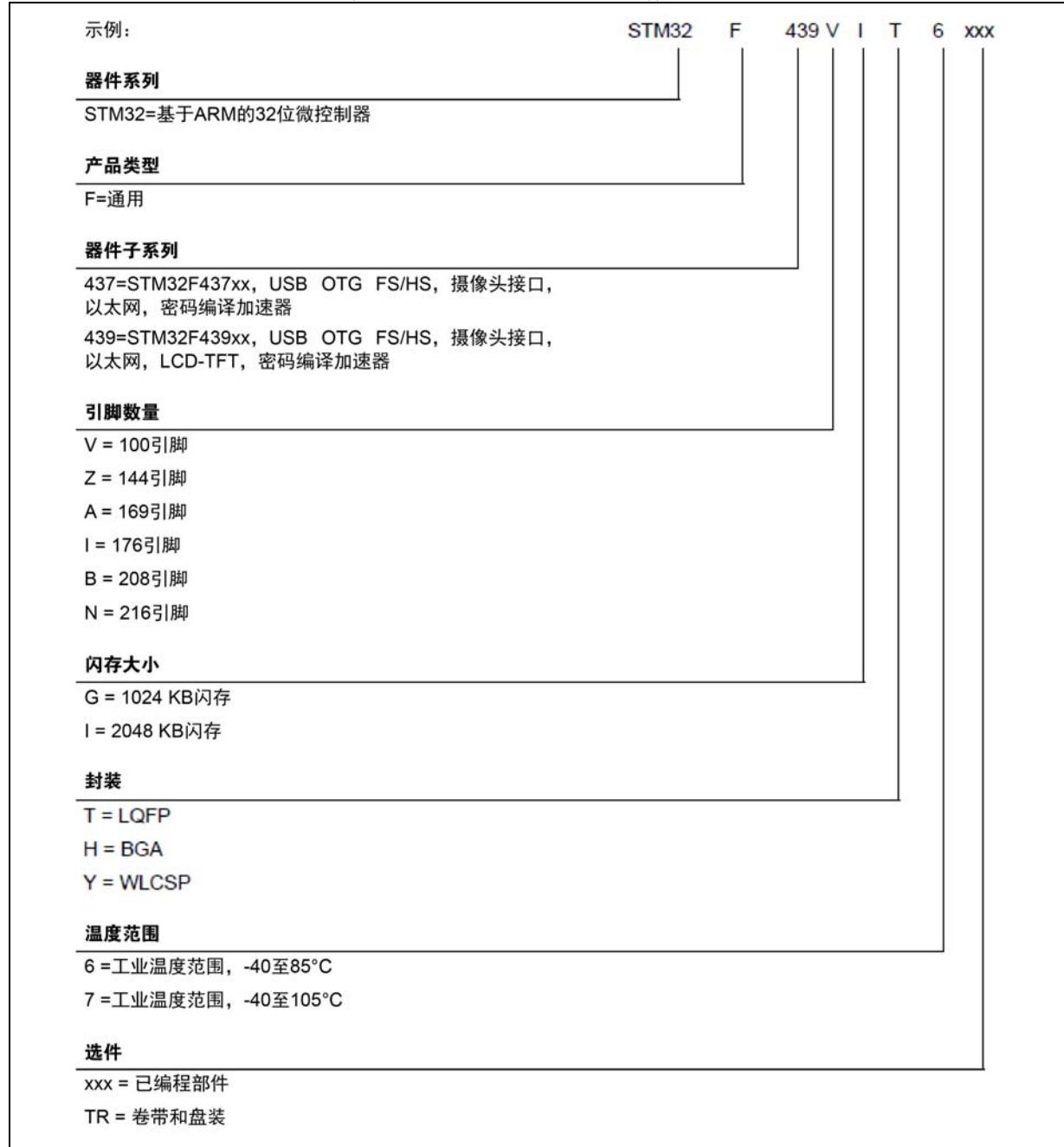
- B = SDIP

- H = BGA

- M = SO
- P = TSSOP
- T = LQFP
- U = VFQFPN
- Y = WLCSP

[图 287](#) 举例介绍了STM32微控制器产品的编号模式。

图287. 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。该跳变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、HSRC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、HSRC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，V_{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HSIRC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、MSIRC、HSE晶振和HSIRC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 μ s内退出待机模式。

- **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 μ s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** (V_{DD}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** (V_{DD}电压范围限制为1.20到1.32 V)
HCLK频率范围最大为144 MHz (168 MHz时过载)
- **级别3** (V_{DD}电压范围限制为1.08到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

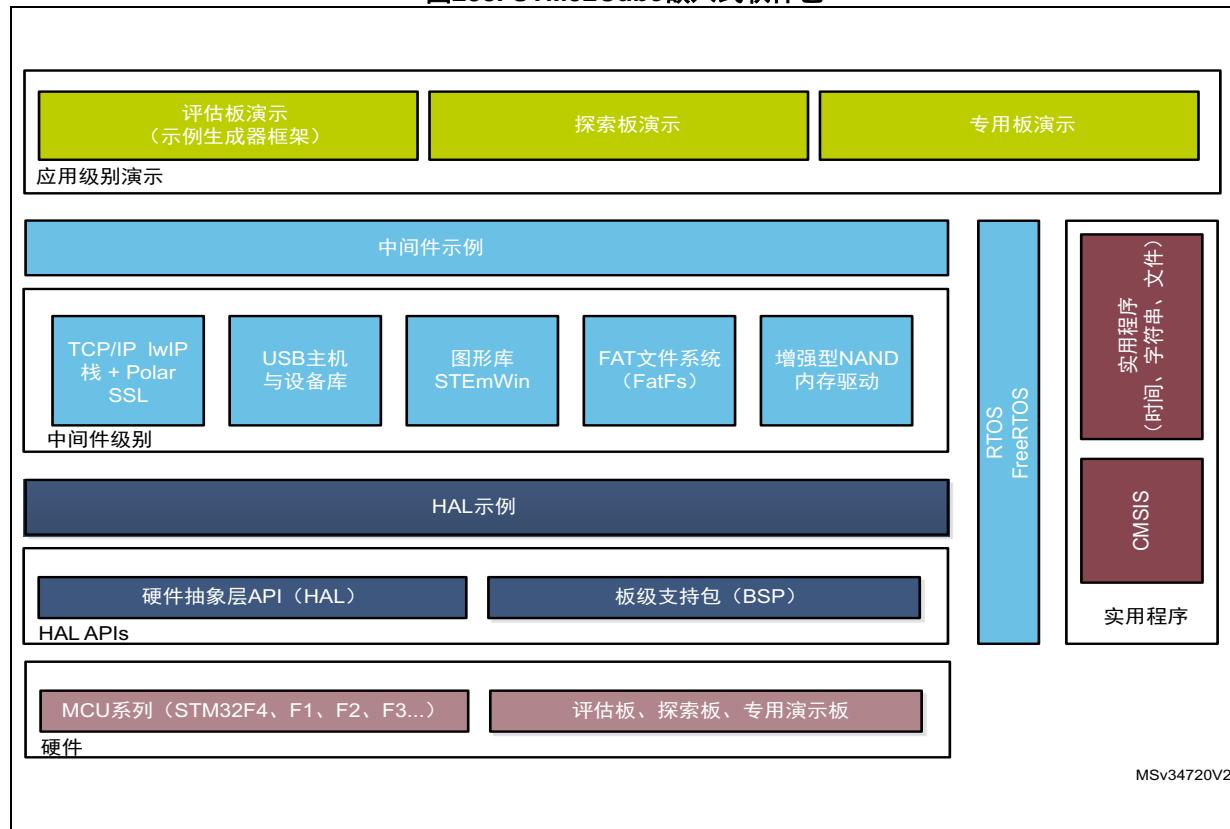
- 范围1 (V_{DD}范围限制为1.71到3.6 V)，CPU最高运行频率为32 MHz
- 范围2 (整个V_{DD}范围)，CPU最大频率为16 MHz
- 范围3 (整个V_{DD}范围)，CPU最大频率限制为4.2 MHz。

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 288）。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288. 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。

16 版本历史

表22. 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	<p>在封面、主要特性、“外设和中间件配置”窗口和附录E: STM32Cube嵌入式软件包中增加了对STM32CubeF2和STM32F2系列的支持。</p> <p>更新了创建一个新STM32CubeMX项目、配置MCU引脚布局、配置MCU初始化参数。</p> <p>“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。</p> <p>增加了安装“Java 7更新45”或更新版的JRE时, 为何会出现“Java 7更新45”错误?。</p>
2014年4月 24日	3	4.3	<p>在封面、主要特性、规则和限制和“外设和中间件配置”窗口中增加了对STM32CubeL0和STM32L0系列的支持</p> <p>在表 3: 文件菜单功能、引脚布局菜单和新项目窗口中增加了板子选择。更新了表 5: 引脚布局菜单。</p> <p>更新了图 120: 功耗计算器默认视图, 并在建立功耗系列中增加了电池选择。</p> <p>更新了功耗计算器视图中的注释</p> <p>更新了创建一个新STM32CubeMX项目。</p> <p>增加了为何RTC复用器在时钟树视图中仍无效? , 如何选择LSE和HSE作为时钟源并更改频率? , 以及在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? 。</p>

表22. 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	<p>在封面、主要特性、规则和限制中增加了对STM32CubeF0、STM32CubeF3、STM32F0和STM32F3系列的支持。</p> <p>在主要特性、表 2：欢迎页面快捷方式、新项目窗口、工具栏和菜单、“设置未使用 / 重置已使用GPIO”窗口、“项目设置”窗口和“引脚布局”视图中增加了板子选择功能和引脚锁定功能。增加了在引脚上锁定和标记信号。</p> <p>更新了配置视图、时钟树配置视图和功耗计算器视图。</p> <p>更新了图 34：选择MCU时显示的STM32CubeMX主窗口、图 50：“项目设置”窗口、图 68：“关于”窗口、图 69：STM32CubeMX“引脚布局”视图、图 70：“芯片”视图、图 120：功耗计算器默认视图、图 121：电池选择、图 122：建立功耗系列、图 124：功耗系列：新步骤默认视图、图 132：构建序列后的功耗计算器视图、图 133：序列表管理功能、图88：PCC编辑步骤窗口、图83：功耗系列：配置新步骤（STM32F4示例）、图 130：在引脚布局视图中选择的ADC、图 131：功耗计算器步骤配置窗口：使用导入引脚布局使能ADC、图 135：结果区域描述、图 136：外设功耗工具提示、图 216：功耗计算示例、图155：序列表和图156：功耗计算结果。</p> <p>更新了图 82：STM32CubeMX“配置”视图和图39：STM32CubeMX配置视图 - STM32F1系列标题。</p> <p>在功耗计算器视图中增加了STM32L1。</p> <p>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配置功耗系列中的步骤中删除。</p> <p>更新了第8.2节：检查结果。</p> <p>更新了附录B.3.4：FatFs和附录D：STM32微控制器功耗参数。增加了附录D.1.3：STM32L0系列和D.2.3：STM32L0系列有三种VCORE范围。</p>

表22. 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	<p>在封面 主要特性、规则和限制中增加了对STM32CubeL1的支持。</p> <p>更新了 卸载STM32CubeMX独立版本。</p> <p>在 获取STM32Cube和第三方软件发布和更新中增加了离线更新，修改了 图 16：“嵌入式软件包管理器”窗口和安装STM32 MCU软件包。</p> <p>更新了 STM32CubeMX用户界面前言、表 2：欢迎页面快捷方式和新项目窗口。</p> <p>增加了 图 33：新项目窗口 - 板选择器。</p> <p>更新了 图 55：项目设置代码生成器。</p> <p>修改了“项目设置”窗口中的步骤3。</p> <p>更新了 图39：STM32CubeMX配置视图 - STM32F1系列。</p> <p>在“外设和中间件配置”窗口中增加了 STM32L1。</p> <p>更新了 图 94：“GPIO配置”窗口 - GPIO选择、 “GPIO配置”窗口和图 100：DMA MemToMem配置。</p> <p>更新了 时钟树配置视图的前言。更新了 时钟树配置功能和建议、功耗计算器视图、图 124：功耗系列：新步骤默认视图、图 132：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 131：功耗计算器步骤配置窗口：使用导入引脚布局使能ADC。增加了 图 134：功耗：外设功耗图，并更新了 图 136：外设功耗工具提示。更新了 功耗系列步骤参数词汇表。</p> <p>更新了 STM32CubeMXC代码生成概述。</p> <p>更新了 创建一个新STM32CubeMX项目和配置MCU引脚布局。</p> <p>增加了 教程2 - 使用STM32429I-EVAL评估板的SD卡上的FatFs示例，并更新了 第8节：教程3- 生成GPIO初始化C代码（仅限STM32F1系列）。</p> <p>更新了 配置功耗系列中的步骤。</p>

表22. 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	<p>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。</p> <p>更新了主要特性和规则和限制。</p> <p>更新了软件要求中的Eclipse IDE。</p> <p>更新了图 12: “更新程序设置”窗口、图 16: “嵌入式软件包管理器”窗口和图 33: 新项目窗口 - 板选择器，更新了“项目设置”窗口和“更新管理器”窗口。</p> <p>更新了图 68: “关于”窗口。</p> <p>删除了图 STM32CubeMX配置视图 - STM32F1系列。</p> <p>更新了表 11: STM32CubeMX “芯片”视图 - 图标和颜色方案。</p> <p>更新了“外设和中间件配置”窗口。</p> <p>更新了图 98: 添加新的DMA请求和图 100: DMA MemToMem配置。</p> <p>更新了时钟树配置功能。</p> <p>更新了图 121: 电池选择、图 122: 建立功耗系列、图88: PCC编辑步骤窗口。</p> <p>增加了自定义代码生成。</p> <p>更新了图 168: 时钟树视图和图 173: 配置视图。</p> <p>更新了配置外设中的外设配置序列和图 175: 定时器3配置窗口。</p> <p>删除了教程3: 生成GPIO初始化C代码（仅限STM32F1系列）。</p> <p>更新了图 179: GPIO模式配置。</p> <p>更新了图 216: 功耗计算示例和图155: 序列表。</p> <p>更新了附录A.1: 块一致性、A.2: 块间依赖性和A.3: 一个块 = 一种外设模式。</p> <p>附录A.4: 块重新映射（仅限STM32F10x）：更新了示例。</p> <p>附录A.6: 块转移（仅适用于STM32F10x，且“保留当前信号布置”已取消选中）：更新了示例</p> <p>更新了附录A.8: 分别映射功能。</p> <p>更新了附录B.3.1: 概述。</p> <p>更新了附录D.1.3: STM32L0系列。</p>

表22. 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	<p>主要特性: 删除了STM32F1系列的引脚布局初始化C代码生成; 更新了完整的项目生成。</p> <p>更新了图 16: “嵌入式软件包管理器”窗口和图 33: 新项目窗口 - 板选择器。</p> <p>更新了“项目设置”窗口中的IDE列表, 并修改了图 50: “项目设置”窗口。</p> <p>更新了时钟树配置功能。更新了图 116: STM32F429xx时钟树配置视图。</p> <p>功耗计算器视图: 增加了转换检查器选项。更新了图 120: 功耗计算器默认视图、图 121: 电池选择和图 122: 建立功耗系列。增加了图 126: 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 127: 在已配置的序列中使能跳变检查器选项 - 至少有一个跳变无效和图 128: 跳变检查器选项 - 显示日志。更新了图 132: 构建序列后的功耗计算器视图。更新了管理序列步骤和管理整个序列 (加载、保存和比较)。更新了图88: PCC编辑步骤窗口和图 135: 结果区域描述。</p> <p>更新了图 216: 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。</p> <p>更新了附录B.3.1: 概述和B.3.5: FreeRTOS。</p>
2015年5月 28日	8	4.8	增加了 从命令行安装STM32CubeMX 和 在命令行模式下运行STM32CubeMX 。
2015年7月 9日	9	4.9	<p>增加了STLM32F7和STM32L4微控制器系列。</p> <p>增加了导入项目功能。在表 3: 文件菜单功能中增加了导入功能。</p> <p>增加了“导入项目”窗口。更新了图 124: 功耗系列: 新步骤默认视图、图88: PCC编辑步骤窗口、图83: 功耗系列: 配置新步骤 (STM32F4示例)、图 131: 功耗计算器步骤配置窗口: 使用导入引脚布局使能ADC和图 136: 外设功耗工具提示。</p> <p>更新了在命令行模式下运行STM32CubeMX中运行STM32CubeMX的命令行。</p> <p>更新了配置视图中的注释。</p> <p>在第 5.14.1节中增加了新时钟树配置功能。</p> <p>更新了图 181: FatFs禁用。</p> <p>修改了附录B.1: STM32CubeMX生成的C代码和用户部分中的代码示例。</p> <p>更新了附录B.3.1: 概述。</p> <p>更新了附录B.3.4: FatFs中生成的.h文件。</p>

表22. 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	<p>将引言中的UM1742替换为UM1940。</p> <p>更新了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。</p> <p>更新了新项目窗口中的板子选择。</p> <p>更新了配置视图概述。更新了“外设和中间件配置”窗口、“GPIO 配置”窗口和“DMA配置”窗口。增加了“用户常量”配置窗口。</p> <p>更新了时钟树配置视图，并添加了存储路径。</p> <p>更新了创建一个新STM32CubeMX项目、配置MCU时钟树、配置MCU初始化参数、下载固件包和生成C代码和构建和更新C代码项目。增加了切换到另一MCU。</p> <p>更新了教程2 - 使用STM32429I-EVAL评估板的SD卡上的FatFs示例，并将STM32F429I-EVAL替换为STM32429I-EVAL。</p>
2015年10月 16日	11	4.11	<p>更新了图 16：“嵌入式软件包管理器”窗口和检查更新。</p> <p>“用户常量”配置窗口中支持字符串常量。</p> <p>更新了时钟树配置视图。</p> <p>更新了功耗计算器视图。</p> <p>修改了图 216：功耗计算示例。</p> <p>更新了教程3 - 使用功耗计算器优化嵌入式应用功耗等。</p> <p>在软件要求中增加了Eclipse Mars</p>
2015年12月 3日	12	4.12	<p>代码生成选项现在受“项目设置”菜单支持。</p> <p>更新了软件要求。</p> <p>在“导入项目”窗口中增加了项目设置。更新了图 42：自动项目导入；修改了手动项目导入步骤，并更新了图 43：手动项目导入和图 44：“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。</p> <p>更新了图 117：有错误的时钟树配置视图。</p> <p>在仅使用HAL驱动程序生成STM32Cube代码（默认模式）中增加了mxconstants.h。</p> <p>更新了图 216：功耗计算示例至图 225：步骤10优化。</p> <p>更新了图 226：优化后的功耗系列结果。</p>

表22. 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	<p>更新了主要特性:</p> <ul style="list-style-type: none"> - 与.ioc文件相关的信息。 - 时钟树配置 - 自动更新STM32CubeMX和STM32Cube。 <p>STM32CubeMX更新了规则和限制中与C代码生成相关的限制。在支持的操作系统和架构中增加了Linux。更新了软件要求中的Java Run Time Environment版本号。</p> <p>更新了安装STM32CubeMX独立版本、卸载STM32CubeMX独立版本和下载STM32CubeMX 插件安装包。</p> <p>更新了STM32CubeMX 作为独立应用程序运行。</p> <p>更新了“项目设置”窗口和“更新管理器”窗口。</p> <p>更新了在引脚上锁定和标记信号。</p> <p>增加了设置HAL时基源</p> <p>更新了图 83: 面向GPIO、DMA和NVIC设置 (STM32F4系列) 的配置窗口。</p> <p>在“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 94: “GPIO配置”窗口 - GPIO选择。</p> <p>修改了图 120: 功耗计算器默认视图、图 122: 建立功耗系列、图 123: 步骤管理功能、图 126: 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 127: 在已配置的序列中使能跳变检查器选项 - 至少有一个跳变无效。</p> <p>在导入引脚布局中增加了导入引脚布局按钮图标。</p> <p>增加了选择/取消选择所有外设。修改了图 132: 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 135: 结果区域描述和图 136: 外设功耗工具提示。</p> <p>更新了图 216: 功耗计算示例和图 218: 序列表。</p> <p>更新了自定义代码生成。</p> <p>更新了创建一个新STM32CubeMX项目中的图 160: 具有MCU选择的引脚布局视图和图 161: 无MCU选择窗口的引脚布局视图。</p> <p>更新了配置外设。</p> <p>更新了设置项目选项中的图 187: 项目设置和工具链选择和图 188: “项目设置”菜单 - “代码生成器”选项卡，并更新了下载固件包和生成C代码中的图 189: 缺少固件包警告消息。</p>

表22. 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	<p>将STM32CubeMX版本号升级为4.14.0。</p> <p>在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。</p> <p>在支持的操作系统和架构、安装STM32CubeMX 独立版本、卸载STM32CubeMX独立版本和从Eclipse IDE运行STM32CubeMX 插件中增加了MacOS。</p> <p>在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。</p> <p>更新了更新程序配置中的新库安装序列。</p> <p>更新了引脚布局菜单中的图 37：引脚布局菜单（已选择引脚布局选项卡）和图 38：引脚布局菜单（未选择引脚布局选项卡）。</p> <p>修改了表 6：窗口菜单。</p> <p>更新了输出窗口。</p> <p>更新了图 50：“项目设置”窗口和“项目”选项卡。</p> <p>更新了设置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。</p> <p>更新了“用户常量”配置窗口中的图 85：“用户常量”窗口和图 86：摘录生成的main.h文件。</p> <p>“GPIO配置”窗口：更新了图 95：“GPIO配置”窗口 - 显示GPIO设置、图 96：按外设分组的GPIO配置和图 97：多引脚配置。</p> <p>更新了“NVIC配置”窗口。</p>
2016年5月 18日	15	4.15	<p>导入项目功能不再限于同系列MCU（参见主要特性、文件菜单和“导入项目”窗口）。</p> <p>更新了在命令行模式下运行STM32CubeMX中的命令行。</p> <p>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。</p> <p>在表 3：文件菜单功能中增加了加载项目菜单的注意。</p> <p>更新了表 4：项目菜单中生成代码菜单的描述。</p> <p>更新了表 5：引脚布局菜单中的设置未使用GPIO菜单。</p> <p>在使用NVIC选项卡视图启用中断中增加了启用FreeRTOS的实例。</p> <p>增加了FreeRTOS中间件配置视图。</p> <p>更新了附录B.3.5：FreeRTOS和B.3.6：LwIP。</p>

表22. 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	<p>将整个文档中的 <code>mxconstants.h</code> 替换为 <code>main.h</code>。</p> <p>更新了 引言、支持的操作系统和架构 和 软件要求。</p> <p>增加了 安装STM32 MCU软件包补丁。</p> <p>更新了 表 2: 欢迎页面快捷方式 中的加载项目描述。</p> <p>更新了 表 5: 引脚布局菜单 中的清除引脚布局功能。</p> <p>更新了“高级设置”选项卡，增加底层驱动程序。</p> <p>在 表 13: “外设和中间件配置”窗口按钮与工具提示 中增加了不检查和十进制和十六进制检查选项。</p> <p>更新了 任务和队列选项卡 和 图 111: FreeRTOS堆用量。</p> <p>更新了 图 95: “GPIO配置”窗口 - 显示GPIO设置。</p> <p>在整个文档中将 PPC 替换为 功耗计算器。</p> <p>增加了 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。</p> <p>更新了 图 262: 引脚布局视图 - 启用RTC。</p> <p>增加了 教程4 - 通过串口与STM32L053xx Nucleo板通信示例。</p> <p>增加了 STM32CubeMX 版本号与文档修订版本之间的对应关系。</p>
2016年11月 21日	17	4.18	<p>删除了 软件要求 中的 Windows XP 并增加了 Windows 10。</p> <p>更新了 卸载STM32CubeMX独立版本。</p> <p>在 表 1: 命令行摘要 中增加了 <code>setDriver</code> 命令行。</p> <p>增加了列出引脚布局兼容 MCU 的功能：</p> <ul style="list-style-type: none"> - 更新了 表 5: 引脚布局菜单。 - 增加了 教程5: 将当前项目配置导出到兼容MCU <p>在 “项目”选项卡 和 图 50: “项目设置”窗口 中增加了固件位置选择选项。</p> <p>增加了恢复默认值功能：</p> <ul style="list-style-type: none"> - 更新了 表 13: “外设和中间件配置”窗口按钮与工具提示 - 更新了 图 87: 使用常量进行外设参数设置。
2017年1月 12日	18	4.19	<p>项目导入不再限于同系列微控制器：更新了 引言、图 42: 自动项目导入、图 43: 手动项目导入、图 44: “导入项目”菜单 - 尝试导入时出现错误 和 图 45: “导入项目”菜单 - 调整之后导入成功。</p> <p>修改了附录 B.3.4: FatFs、B.3.5: FreeRTOS 和 B.3.6: LwIP。增加了附录 B.3.7: Libjpeg。</p>
2017年3月 2日	19	4.20	<p>表 11: STM32CubeMX “芯片”视图 - 图标和颜色方案：</p> <ul style="list-style-type: none"> - 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。 <p>更新了 图 84: “外设配置”窗口 (STM32F4系列)、图 85: “用户常量”窗口、图 91: 删除用于外设配置的用户常量 - 对外设配置的影响、图 92: 在用户常量列表中搜索常量名称 和 图 93: 在用户常量列表中搜索常量值。</p> <p>增加了 SMPS特性。</p> <p>增加了 其他C项目生成设置。</p> <p>在附录 B.3.7: Libjpeg 中向包含 Libjpeg 的软件包列表中添加了 STM32CubeF4。</p>

表22. 文档版本历史

日期	版本	STM32CubeMX 版本号	变更
2017年5月 5日	20	4.21	<p>在STM32Cube 概述中做了少量改动。</p> <p>更新了图 28: 新项目窗口 - MCU选择器和图 50: “项目设置”窗口。</p> <p>更新了“项目”选项卡中项目设置的描述。</p> <p>更新了图 58: “高级设置”窗口。</p> <p>在附录B.3.7: Libjpeg中嵌入了Libjpeg的软件包列表中增加了STM32CubeF2和STM32CubeH7。</p> <p>修改了图 288: STM32Cube嵌入式软件包的外观。</p>
2017年7月 6日	21	4.22	<p>将STM32H7增加到受支持STM32系列列表中。</p> <p>在获取STM32Cube和第三方软件发布和更新中增加了MCU数据和文档刷新功能，并更新了图 12: “更新程序设置”窗口。</p> <p>在新项目窗口中增加了识别靠近MCU的功能，更新了图 28: 新项目窗口 - MCU选择器、增加了图 31: 新项目窗口 - 具有相近MCU特性的MCU列表和图 32: 新项目窗口 - 显示相近MCU的MCU列表，更新了图 159: MCU选择。</p> <p>更新了图 34: 选择MCU时显示的STM32CubeMX主窗口。</p> <p>在表 5: 引脚布局菜单中增加了顺时针/逆时针旋转以及顶部/底部视图。</p> <p>增加了社交链接。</p> <p>更新了图 143: 为每个步骤配置SMPS模式。</p> <p>更新了使用底层驱动程序生成STM32Cube代码。</p> <p>更新了图 187: 项目设置和工具链选择。</p>
2017年9月 5日	22	4.22.1	<p>在引言、功耗计算器视图和使用底层驱动程序生成STM32Cube代码中增加了STM32L4+系列。</p> <p>在STM32CubeMX 作为独立应用程序运行中增加了在MacOS上运行STM32CubeMX的指南。</p> <p>删除了从Eclipse IDE运行STM32CubeMX 插件中的MacOS。</p> <p>增加了以太网配置：为什么有时候我不能指定DP83848或LAN8742A？</p>
2017年10月 18日	23	4.23	<p>增加了概述。</p> <p>将新项目窗口中的显示相近按钮重命名为显示类似项目。</p> <p>在帮助菜单中增加了刷新数据和文档和资源菜单。</p> <p>在使用底层驱动程序生成STM32Cube代码中增加了STM32F2、STM32F4和STM32F7系列。</p> <p>增加了附录B.3.8: Mbed TLS。</p> <p>更新了用户手册修订版本22对应的STM32CubeMX版本号。</p>

表22. 文档版本历史

日期	版本	STM32CubeMX 版本号	变更
2018年1月 16日	24	4.24	<p>用“STM32Cube MCU封装”替换了“STM32Cube固件封装”。 更新了STM32Cube 概述。</p> <p>更新了支持的操作系统和架构中的MacOS。更新了软件要求中的Eclipse要求。 获取STM32Cube和第三方软件发布和更新:</p> <ul style="list-style-type: none"> - 更新了“前言”一节 - 更新了图 13: “连接参数”选项卡 - 无代理 - 第 4.5.2 节重命名为“安装STM32 MCU软件包”并进行了更新。 - 第 4.5.3 节重命名为“安装STM32 MCU软件包补丁” - 增加了安装嵌入式软件包 - 更新了检查更新 <p>更新了图 33: 新项目窗口 - 板选择器。</p> <p>更新了图 35: 选择板时显示的STM32CubeMX主窗口 (未选中外设默认选项)和介绍性文字。</p> <p>更新了图 36: 选择板时显示的STM32CubeMX主窗口 (选中外设默认选项)和介绍性文字。</p> <p>在表 4: 项目菜单中增加了“选择附加软件组件”菜单。 将“安装新库”菜单重命名为“管理嵌入式软件包”，并在表 7: 帮助菜单中更新了相应描述。</p> <p>更新了删除已安装的嵌入式软件包。</p> <p>更新了“更新管理器”窗口</p> <p>更新了“更新管理器”窗口。增加了附加软件组件选择窗口。 在表 11: STM32CubeMX “芯片”视图 - 图标和颜色方案中增加了引脚堆叠功能。</p> <p>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。</p> <p>教程2 - 使用STM32429I-EVAL评估板的SD卡上的FatFs示例: 更新了图 208: 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。</p> <p>教程4 - 通过串口与STM32L053xx Nucleo板通信示例: 更新了图 227: 选择NUCLEO_L053R8板。</p> <p>增加了教程6 - 将嵌入式软件包添加到用户项目。</p>

表22. 文档版本历史

日期	版本	STM32CubeMX 版本号	变更
2018年1月 16日	24 (续)	4.24	<p>增加了附录B.3.9: TouchSensing和B.3.10: PDM2PCM。 “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。</p>
2018 年3月 7日	25	4.25	<p>更新了引言、STM32Cube 概述、规则和限制、 安装STM32CubeMX 独立版本、STM32CubeMX用户界面、“项目”选 项卡和“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 3: 文件菜单功能和表 16: 功率超载模式与HCLK频率之间 的关系。 更新了图 28: 新项目窗口-MCU选择器、图 29: 在MCU选择器中使 能图形选择、图 50: “项目设置”窗口、图 52: 选择不同的固件 位置、图 112: 使能STemWin框架、图 113: 图形配置视图、 图 263: 引脚布局视图-启用LSE和HSE时钟和图 264: 引脚布局视 图 - 设置LSE/HSE时钟频率。 增加了导出至Excel功能、显示收藏的MCU特性和图形框架和仿真 器。 增加了教程7-使用STemWin图形框架、教程8: 使用STM32CubeMX 图形仿真器及其包含的小节。 增加了图形。</p>

表23. 中文文档版本历史

日期	版本	STM32CubeMX 版本号	变更
2019年3月 25日	1	4.1	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。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。本文档的中文版本为英文版本的翻译件，仅供参考之用；若中文版本与英文版本有任何冲突或不一致，则以英文版本为准。

© 2019 STMicroelectronics - 保留所有权利