面向 STM32WL 系列的 STM32CubeWL 入门

引言

意法半导体提供的 STM32Cube，旨在通过减少开发工作量、时间和成本，来显著提高开发人员的工作效率。STM32Cube 涵盖整个 STM32 产品系列。

STM32Cube 包括：

- 一套用户友好的软件开发工具，覆盖从设计到生产的整个工程开发过程，其中包括：
  - 图形软件配置工具 STM32CubeMX，可通过图形向导自动生成初始化 C 代码。
  - STM32CubeProgrammer（STM32CubeProg），支持图形接口和命令行接口的编程工具。
  - STM32CubeMonitor-Power（STM32CubeMonPwr），测量并帮助优化 MCU 功耗的监控工具。
  - STM32CubeMonitor，配有专用附加组件、利用 STM32WL 执行射频测试（动态数据包传输/接收、PER 测量）并以图形表示射频性能的多功能监控工具。
- 针对每个系列提供综合的嵌入式软件平台，（比如用于 STM32WL 系列的 STM32CubeWL）：
  - STM32 抽象层嵌入式软件 STM32Cube HAL，确保用户应用在 STM32 各个产品之间实现最大限度的可移植性。
  - 底层 API（LL）提供快速、轻量且面向专业人士的层，比 HAL 更接近于硬件。LL API 仅可用于一组外设。
  - 一组统一的中间件组件，如 FatFS、FreeRTOS™、LoRaWAN®、SubGHz_Phy、Sigfox™、KMS、SE 和 mbed-crypto。
  - 所有嵌入式软件实用工具均配备一套完整的示例。

1 STM32CubeWL 主要特性

STM32CubeWL 在基于 Arm® Cortex®-M 处理器的 STM32WL 系列微控制器上运行。

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

STM32CubeWL 将开发 STM32WL 微控制器应用所需的所有通用内置软件组件聚集在单一软件包中。根据 STM32Cube 计划，这套组件具有高度可移植性，不仅在 STM32WL 系列范围内，也适用于其他 STM32 系列。

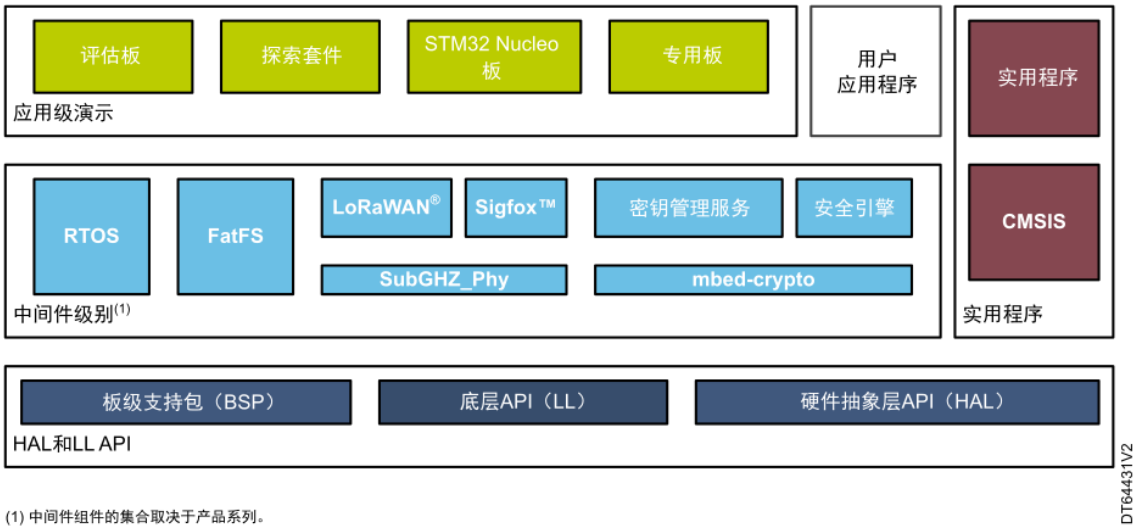
STM32CubeWL 与可以生成初始化代码的 STM32CubeMX 代码生成器完全兼容。软件包包括底层（LL）和硬件抽象层（HAL）API。这些 API 涵盖了微控制器硬件，以及在意法半导体板上运行的大量示例。HAL 和 LL API 提供开源 BSD 许可证，以便用户使用。

STM32CubeWL MCU 软件包还包含一组中间件组件以及相应的示例。它们具有免费的用户友好许可条款：

- 使用 FreeRTOS™开源解决方案实现 CMSIS-RTOS
- 基于开源 FatFS 解决方案的 FAT 文件系统
- LoRaWAN®，Lora 广域网
- SubGHz_Phy，根据 OSI 模型适于所有以上 MAC 层的常用物理层
- Sigfox™，Sigfox 协议库
- KMS，安全密钥管理服务
- SE，安全引擎
- mbed-crypto，mbed 加密库

STM32CubeWL MCU 软件包中还提供实现所有这些中间件组件的一些应用程序。

图 1. STM32CubeWL 软件组件

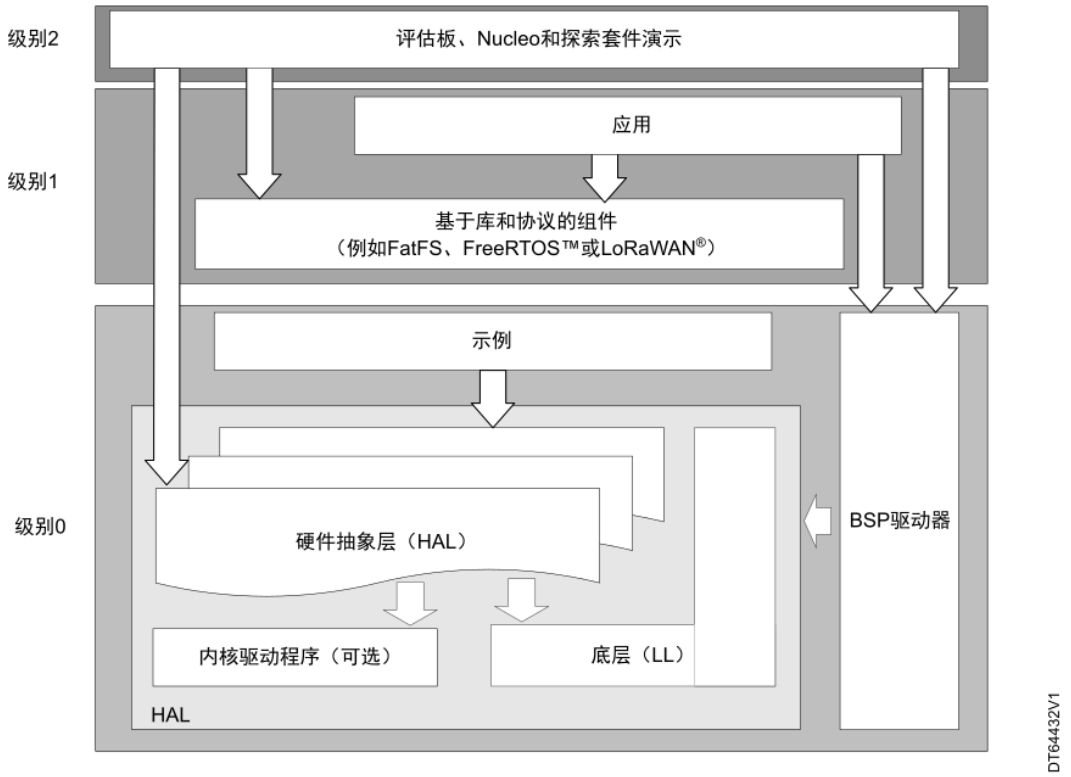


(1) 中间件组件的集合取决于产品系列。

2 STM32CubeWL 架构概述

STM32CubeWL 软件解决方案围绕三个独立的级别构建，可以轻松交互，如图 2 中所述。

图 2. STM32CubeWL 软件架构



2.1 级别 0

此层级分为三个子层：

- 板级支持包（BSP）
- 硬件抽象层（HAL）
  - HAL 外设驱动程序
  - 底层驱动
- 基本外设用例

2.1.1 板级支持包（BSP）

该层提供了对应于板载硬件组件的一系列 API（如 LCD、Audio、microSD™和 MEMS 驱动程序）。它包含两部分：

- 组件
  该驱动程序与板件上的外部器件（而不是 STM32）有关。组件驱动程序为 BSP 驱动程序的外部组件提供专用 API，并且可以移植到任何其他板件上。
- BSP 驱动程序
  允许将组件驱动程序链接到专用板件上，并提供一组易于使用的 API。API 命名规则是 BSP_FUNCT_Action()。
  示例：BSP_LED_Init(), BSP_LED_On()

BSP 基于模块化架构，只需执行低层级例程，便可轻松移植到任何硬件上。

2.1.2 硬件抽象层（HAL）和底层（LL）

STM32CubeWL HAL 和 LL 是互补的，可满足广泛的应用要求:

- HAL 驱动程序提供面向功能的高可移植的顶层 API。它们向最终用户隐藏了 MCU 和外设的复杂性。
  HAL 驱动程序提供通用多实例且面向功能的 API，通过提供可用的步骤来帮助用户简化应用程序的实现。例如，对于通信外设（I²C、UART 等），它提供了 API，用于外设初始化和配置，以及基于轮询、中断或 DMA 处理的数据传输管理和处理通信过程中可能出现的通信错误。HAL 驱动程序 API 分为两类：
  - 为所有 STM32 系列提供通用功能的通用 API
  - 以及为特定系列或特定产品编号的器件提供特殊定制功能的扩展 API。
- 底层 API 提供寄存器级别的底层 API，带有更好的优化，但可移植性较差。需要对 MCU 和外设技术参数有深入的了解。
  LL 驱动程序旨在提供一个快速、轻量且面向专业人士的层，比 HAL 更接近于硬件。与 HAL 相反，LL API 不用于优化访问并非关键特性的外设，或者需要大量软件配置和/或复杂上层栈的外设。
  底层（LL）驱动程序具有：
  - 一组函数，用于根据数据结构中指定的参数，对外设主要特性进行初始化
  - 一组函数，用于使用每个字段相应的复位值填充初始化数据结构
  - 函数，用于外设去初始化（外设寄存器恢复为默认值）
  - 一组内联函数，用于直接和原子寄存器访问
  - 完全独立于 HAL，可在独立模式（无 HAL 驱动程序）下使用
  - 涵盖全部支持的外设特性
- 实现双核：
  - 相同的 HAL/LL 驱动程序支持单核及双核 STM32WL 系列
    - 在 STM32WL 双核器件中，两个内核（Cortex®-M4 和 Cortex®-M0+）可以采用相同的方式访问所有外设。这意味着，Cortex®-M4 与 Cortex®-M0+之间不存在外设分割或默认分配。为此，两个内核之间共享相同的外设 HAL 和 LL 驱动程序。
    - 此外，有些外设（主要是：RCC、GPIO、PWR、HSEM、IPCC、GTZC...）还具有附加的双核特性：
      - “DUAL_CORE”定义用于界定仅双核系列上可用的代码（定义、函数、宏...）。
      - “CORE_CM0PLUS”定义用于界定双核系列上适于 Cortex®-M0+的特定配置/代码部分的代码。使用反转或“else”语句时，此定义用于界定双核系列上适于 Cortex®-M4+的特定配置/代码部分的代码。

2.1.3 基本外设用例

该层包含有围绕 STM32 外设构建（仅使用 HAL 和 BSP 资源）的示例。

可提供单核与双核示例：

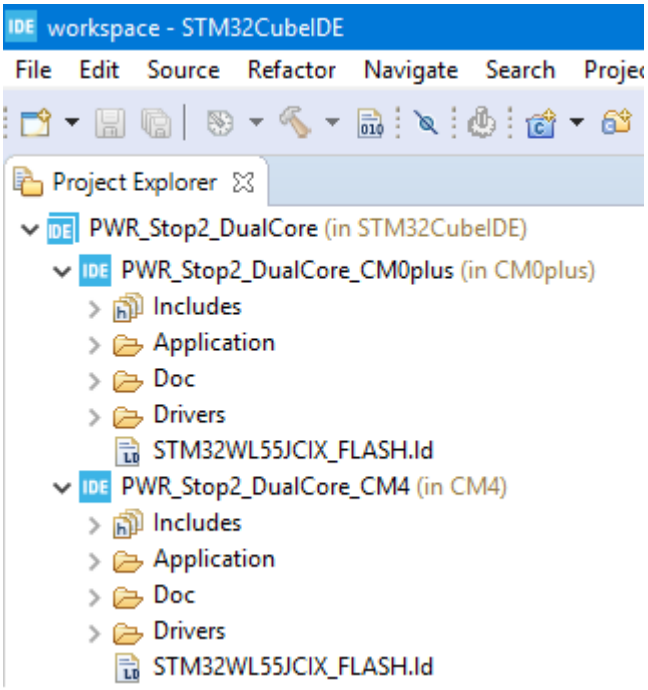
- 能够在单核上演示的外设或特性的单核示例
- 需要两个内核上同时运行程序的外设或特性的双核示例（IPCC、HSEM、PWR 双核特性、GTZC...）

双核示例和应用采用专用架构：

- 在双核系列中，仅为每个示例/应用提供一个工程（一个工作区）。这一点将与传统（单核系列）架构兼容。
- 名称后缀为“CM4”和“CM0PLUS”的每个工作区（每个内核各 1 个）配置两个目标工程。

- 每个目标配置具有各自的选项设置：目标设备、链接器选项、RO、RW 区、预处理器符号（CORE_CM0PLUS），因此可以单独针对每个内核进行用户代码的编译、链接和编程。编译会产生两种二进制：CM4 二进制文件和 CM0+二进制文件。

图 3. 双核工程架构



2.2 级别 1

此层级分为两个子层：

- 中间件组件
- 基于中间件组件的示例

2.2.1 中间件组件

中间件是涵盖 FatFS、FreeRTOS™、LoRaWAN®、SubGHz_Phy、Sigfox™、KMS、SE 和 mbed-crypto 的一组库。该层组件之间的水平交互通过调用 API 功能直接完成。与底层驱动程序的垂直交互则通过在库系统调用接口中实现的特定回调和静态宏完成。例如，FatFS 实现了磁盘 I/O 驱动程序，以访问 microSD™驱动。

每个中间件组件的主要特性如下：

- FAT 文件系统
  - FatFS FAT 开源库
  - 支持长文件名
  - 动态多驱动器支持
  - RTOS 和独立操作
  - microSD™的示例
- FreeRTOS™
  - 开源标准
  - CMSIS 兼容性层
  - 低功耗模式下的无时间片操作
  - 与所有 STM32Cube 中间件模块集成

- LoRaWAN®
  - 提供非常引人注目的远距离、低功耗和安全数据传输组合。使用该技术的公共和私有网络的覆盖范围超过现有的蜂窝网络。易于加入到现有基础设施，并为电池供电的物联网应用提供解决方案。
- SubGHz_Phy
  - 实现了适合 Sub-GHz 协议的 PHY 层。尽管专门用于 LoRaWAN® MAC，但也可连接大多数 Sub-GHz 协议。提供管理传输、射频接收处理程序和超时的抽象层。利用独特的射频 API，还实现了适合其他 I-CUBE-LRAWAN 的多个射频驱动程序
- Sigfox™
  - 实现了兼容 Sigfox™协议网络的 Sigfox™协议库。还囊括 RF 测试协议库，从而对 RF Sigfox™工具进行测试。
- SE
  - 安全引擎中间件（SE）提供了用于管理所有关键数据和操作（例如访问软件密钥的操作，等等）的受保护环境。受保护的代码和数据可通过一个单入口点（调用门机制）进行访问，因此不能在不经过单入口点的情况下运行或访问任何 SE 代码或数据，否则会产生系统复位。
- KMS
  - 密钥管理服务中间件（KMS）通过 PKCS #11 API（基于密钥 ID 的 API）为用户应用提供加密服务。通过将 KMS 置于安全区域中确保安全性。用户应用密钥存储在安全区域中，可进行安全更新（更新前进行真实性检查、解密和完整性检查）。
- mbed-crypto
  - mbed-crypto 中间件以开源代码形式提供。此中间件提供 PSA 密码 API 实现。

2.2.2 基于中间件组件的示例

每个中间件组件都附带一个或多个示例（又称应用程序），来演示如何使用它。还提供了使用多个中间件组件的集成示例。

3 STM32CubeWL 软件包概述

3.1 支持的 STM32WL 型号和硬件

STM32Cube 提供围绕通用架构构建的高度可移植的硬件抽象层（HAL）。它允许使用构建层，例如使用中间件层来实现其功能，而无需深入了解所使用的 MCU。这可提高库代码的可复用性，并确保可向其他设备轻松移植。

此外，由于其分层架构，STM32CubeWL 可为所有 STM32WL 系列提供全面支持。用户只需在 stm32wlxx.h 中定义正确的宏即可。

表 1 显示要根据所使用的 STM32WL 型号定义的宏。该宏也应在编译器预处理器中定义。

表 1. STM32WL 系列的宏

在 stm32wlxx.h 中定义的宏	STM32WL 型号
STM32WL54xx	STM32WL54CC, STM32WL54JC
STM32WL55xx	STM32WL55CC, STM32WL55JC
STM32WLE4xx	STM32WLE4C8, STM32WLE4CB, STM32WLE4CC, STM32WLE4J8, STM32WLE4JB, STM32WLE4JC
STM32WLE5xx	STM32WLE5C8, STM32WLE5CB, STM32WLE5CC, STM32WLE5J8, STM32WLE5JB, STM32WLE5JC
STM32WL5Mxx	STM32WL5MOC

STM32CubeWL 为所有级别提供了丰富的示例和应用程序集，简单易懂，便于使用任何 HAL 驱动程序和/或中间件组件。这些示例在表 1 中列出的 STMicroelectronics 板上运行。

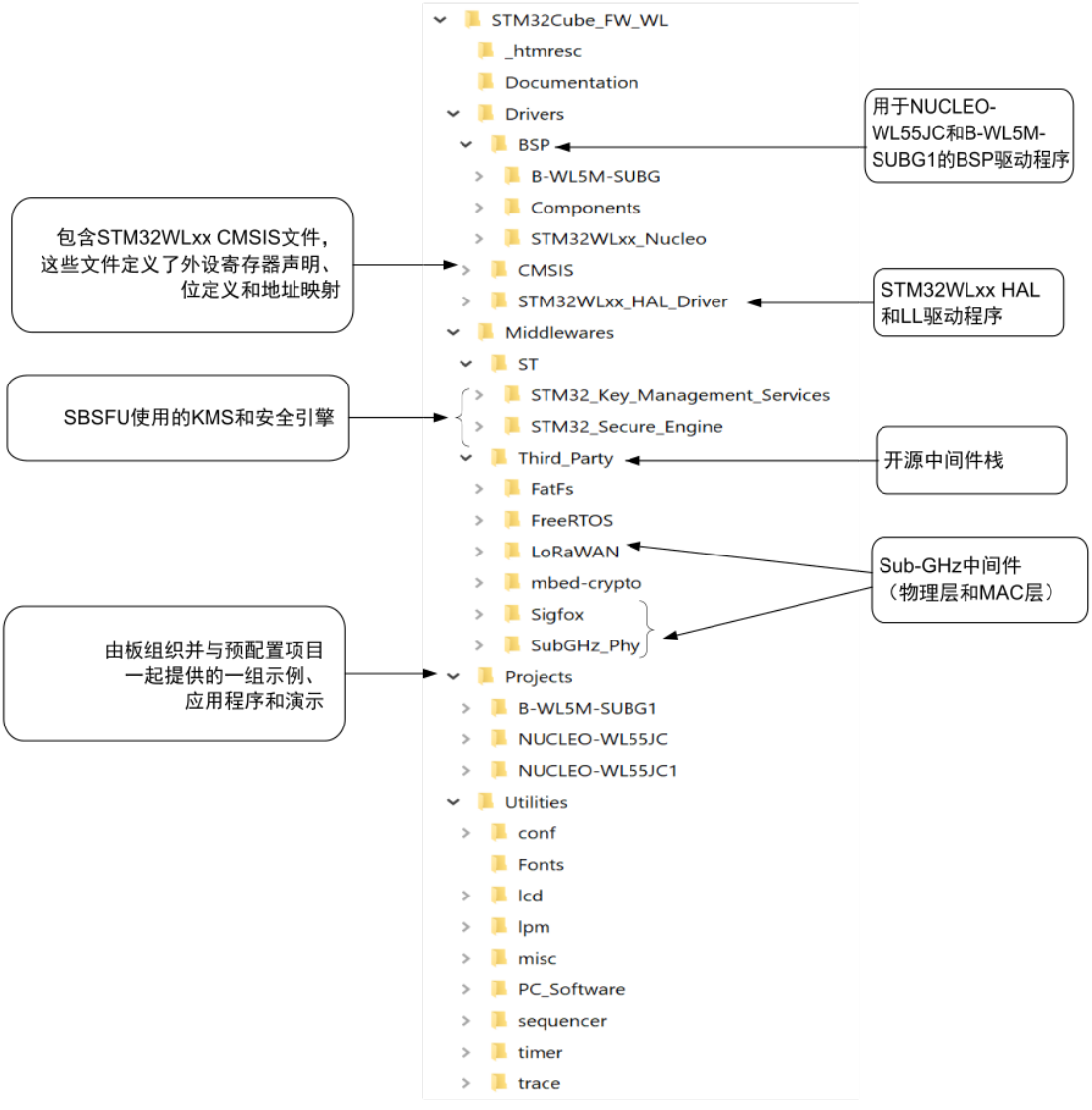
表 2. STM32WL 系列的开发板

支持的板
NUCLEO-WL55JC1, NUCLEO-WL55JC2, B-WL5M-SUBG1

3.2 软件包概述

STM32CubeWL 软件解决方案以一个单一的 zip 软件包提供，其结构如图 4 中所示。

图 4. STM32CubeWL 软件包结构

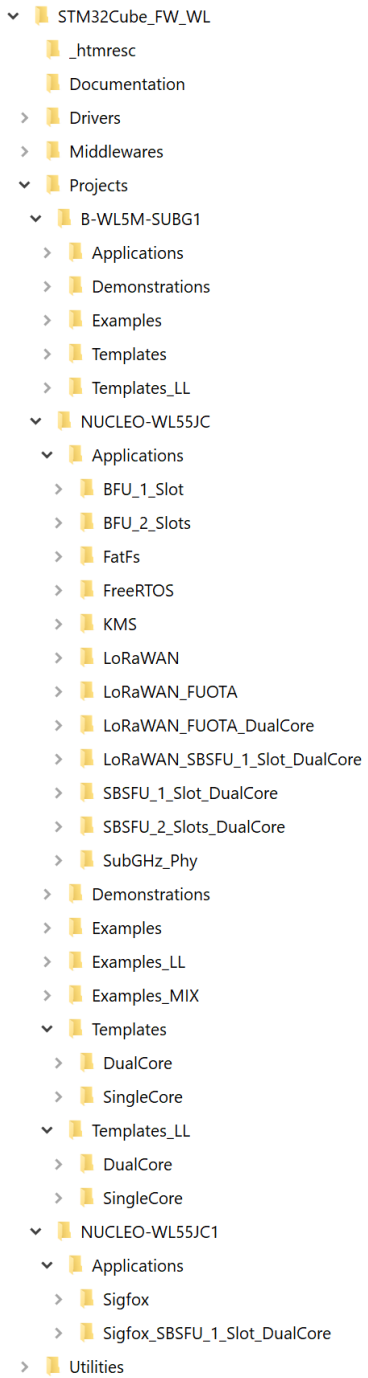


注意： *用户不得修改组件文件。用户只可更改\Project 下的源文件。*

对于 NUCLEO-WL55JC1、NUCLEO-WL55JC2 和 B-WL5M-SUBG1 板，将在一组示例中提供 EWARM、MDK-ARM 和 STM32CubeIDE 工具链的预配置工程。

图 5 展示了 STM32CubeWL MCU 软件包的工程结构。

图 5. STM32CubeWL 示例概述



这些示例根据其适用的 STM32Cube 级别进行分类，并按以下说明进行命名：

- 级别 0 的示例称为 Examples、Examples_LL 和 Examples_MIX。这些示例分别使用 HAL 驱动程序、LL 驱动程序以及无任何中间件组件的 HAL 和 LL 驱动程序组合。
- 级别 1 的示例被称为“Applications”。它们提供了每个中间件组件的典型用例。

通过 Templates 和 Templates_LL 目录中提供的模板工程，可在给定的板上快速构建任何软件应用。每个模板目录中包含两个工程：用于所有 STM32WL 系列的单核工程以及用于 STM32WL5x（双核）系列的双核工程。

单核示例都具有相同的结构：

- 包含所有头文件的*Inc* 文件夹。

- 源代码的*\Src* 文件夹。
- *\EWARM*、*\MDK-ARM 和*\STM32CubeIDE* 文件夹包含每个工具链的预配置工程。
- *readme.txt* 描述了示例行为和使其运行所需的环境
- *.ioc* 文件，允许用户打开 STM32CubeMX 中的大多数软件示例（从 STM32CubeMX 6.1.0 开始）

所有双核示例都具有相同的结构：

- 两个单独的文件夹 *CM4* 和 *CM0PLUS* 分别用于 Cortex®-M4 和 Cortex®-M0+
- 每个文件夹（CM4 和 CM0PLUS）提供：
  - 包含用于 Cortex®-M4/M0+的所有头文件的\Inc 文件夹
  - 包含用于 Cortex®-M4/M0+的源代码文件的\Src 文件夹
- 包含两个内核使用的所有头文件和源文件的*\Common* 文件夹。默认情况下，Common\System 设置为共享 system_stm32wlxx.c 全局文件。但也可以根据示例定义其他特定目录。
- *\EWARM*、*\MDK-ARM*、*\STM32CubeIDE* 文件夹包含用于每个工具链的预配置工程（Cortex®-M4 和 Cortex®-M0+目标配置）
- *readme.txt* 描述了示例行为和使其运行所需的环境
- *.ioc* 文件，允许用户打开 STM32CubeMX 中的大多数软件示例（从 STM32CubeMX 6.1.0 开始）

4 STM32CubeWL 入门

4.1 运行第一个示例

本节介绍了在 STM32CubeWL 中运行第一个示例的简便性。举例说明了如何生成在 NUCLEO-WL55JC 板上运行的简单 LED 切换：

1. 下载 STM32CubeWL 软件包。将其解压到所选目录中。确保不要修改图 5 所示的软件包结构。请注意，还建议将软件包复制到尽可能靠近根卷的位置（C\Eval 或 G:\Tests），因为某些 IDE 在路径长度过长时会遇到问题。
2. 浏览到*\Projects\NUCLEO-WL55JC\Examples*。
3. 打开*\GPIO*，然后打开*\GPIO_EXTI* 文件夹。
4. 使用首选工具链打开工程。下面简单概述了使用所支持的工具链打开、构建和运行示例的具体方式。
5. 重新编译所有文件，并将二进制文件加载到目标内存中。
6. 运行示例：每次按下用户按钮（SW1），LED2 就会切换（有关更多详细信息，请参见示例自述文件）。

要使用支持的工具链打开、构建和运行某个示例，请按照以下步骤操作：

EWARM

1. 在示例文件夹下，打开\EWARM 子文件夹
2. 启动 Project.eww 工作区(*)
3. 重建所有文件：工程→重建全部
4. 加载工程镜像：工程→调试
5. 运行程序：调试→运行（F5）

MDK-ARM

1. 在示例文件夹下方，打开\MDK-ARM 子文件夹。
2. 打开 Project.uvproj 工作区（*）
3. 重建所有文件：工程→重建全部目标文件。
4. 加载工程镜像：调试→开始/停止调试会话。
5. 运行程序：调试→运行（F5）。

• STM32CubeIDE

1. 打开 STM32CubeIDE 工具链。
2. 点击文件系统中的“文件”→“打开工程”
3. 浏览到 STM32CubeIDE 工作区目录并选择工程。
4. 重建所有工程文件：在“Project explorer”窗口中选择工程，然后单击“Project→build project”菜单。

(*)：工作区名称可能因示例而变化。

4.2 开发自己的应用

4.2.1 使用 STM32CubeMX 开发或更新应用

在 STM32CubeWL MCU 软件包中，生成几乎所有的示例工程时都使用 STM32CubeMX 工具用于初始化系统、外设和中间件。

直接从 STM32CubeMX 工具中使用现有示例工程需要 STM32CubeMX 6.1.0 或更高版本：

- 安装 STM32CubeMX 后，打开并根据需要更新建议的工程。打开现有工程的最简单方法是双击*.ioc 文件，以便 STM32CubeMX 自动打开工程及其源文件。
- 此类工程的初始化源代码由 STM32CubeMX 生成；主应用程序源代码包含在注释“USER CODE BEGIN”和“USER CODE END”中。如果 IP 选择和设置已被修改，则 STM32CubeMX 更新代码的初始化部分，但保留主应用程序源代码。

要在 STM32CubeMX 中开发自定义工程，请按照以下流程逐步操作：

1. 选择与所需外设组相匹配的 STM32 微控制器。
2. 使用引脚排列冲突解决器、时钟树设置助手、功耗计算器以及执行 MCU 外设配置（例如 GPIO 或 USART）和中间件栈（例如 USB）的实用程序，配置所有必需的内置软件。
3. 基于所选配置，生成初始化 C 代码。该代码在多种开发环境中即时使用。在下一次代码生成中，用户代码将会被保留。

有关 STM32CubeMX 的更多信息，请参阅 UM1718“STM32CubeMX 用于 STM32 配置和初始化 C 代码生成”。

有关 STM32CubeWL 的可用示例工程的列表，请参阅应用笔记 AN5409“STM32WL 系列 STM32Cube 软件示例”。

4.2.2 驱动程序应用

4.2.2.1 HAL 应用程序

本节介绍了使用 STM32CubeWL 创建自定义 HAL 应用程序所需的步骤：

1. **创建用户工程**
   要新建一个工程，请从\Projects\<STM32xxx_yyy>\Templates 下为每个板提供的模板工程开始，或者从\Projects\<STM32xxy_yyy>\Examples 或\Projects\<STM32xx_yyy>\Applications（其中<STM32xxx_yyy>是指板名称，例如）下的任何可用工程开始。
   Template 工程会提供空的主循环函数，但首先最好熟悉 STM32CubeWL 的工程设置。模板具有下列特性：
   - 它包含 HAL、CMSIS 和 BSP 驱动程序的源代码，这些驱动程序是在给定板上开发代码所需的最少组件。
   - 它包含所有软件组件的包含路径。
   - 它定义支持的 STM32WL 器件，因此可以相应地配置 CMSIS 和 HAL 驱动程序。
   - 它提供了预配置的即用型用户文件，如下所示：
     HAL 使用 ARM Core SysTick 初始化为默认时基。
     为 HAL_Delay()实现 SysTick ISR。

注意： *将现有工程复制到其他位置时，请确保更新包含路径。*

2. **将所需的中间件添加到用户工程中（可选）**
   可用的中间件栈包括：FatFS、FreeRTOS™和 LoRaWAN®。要了解要添加到工程文件列表中的源文件，请参阅每个中间件随附的文档。可参阅\Projects\STM32xxx_yyy\Applications\<MW_Stack>下的应用程序（其中<MW_Stack>是指中间件栈，例如 FreeRTOS™），以了解应添加哪些源文件和哪些包含路径。
3. **配置软件组件**
   HAL 和中间件组件使用头文件中声明的#define 宏提供了一组构建时间配置选项。每个组件中都设有一个模板配置文件，必须将其复制到工程文件夹中（该配置文件的名称通常为 xxx_conf_template.h，在将其复制到工程文件夹中时，需要删除“_template”一词）。配置文件提供了用于了解每个配置选项的影响的充足信息。每个组件随附文档中提供了更多详细信息。
4. **启动 HAL 库**
   跳转到主程序后，应用程序代码需调用 HAL_Init()API 来初始化 HAL 库，该库执行以下任务：
   a. 配置 Flash 预取和 SysTick 中断优先级（通过 sstm32wlxx_hal_conf.h 中定义的宏）。
   b. 在 stm32wlxx_hal_conf.h 中定义的 SysTick 中断优先级 TICK_INT_PRIO 处配置 SysTick，以每毫秒生成一个中断。
   c. 设置 NVIC 组优先级到 0。
   d. 调用 stm32wlxx_hal_msp.c 用户文件中定义的 HAL_MspInit()回调函数，以执行全局低层硬件初始化。

5. **配置系统时钟**
通过调用下述两个 API 完成系统时钟配置：
   a. HAL_RCC_OscConfig()：该 API 用于配置内部和/或外部振荡器以及 PLL 源和分频系数。用户选择配置一个或所有振荡器。如果不需要以高频率运行系统，则可以跳过 PLL 配置。
   b. HAL_RCC_ClockConfig()：该 API 用于配置系统时钟源、Flash 存储器延迟以及 AHB 和 APB 预分频器。
6. **初始化外设**
   a. 首先编写外设 HAL_PPP_MspInit 函数。请按如下步骤操作：
      - 启用外设钟。
      - 配置外设 GPIO。
      - 配置 DMA 通道并启用 DMA 中断（若需要）。
      - 启用外设中断（若需要）。
   b. 如果需要，编辑 *stm32xxx_it.c* 以调用所需的中断处理程序（外设和 DMA）。
   c. 如果需要外设中断或 DMA，则写入进程完成回调函数。
   d. 在用户 *main.c* 文件中，初始化外设句柄结构，然后调用函数 HAL_PPP_Init()以初始化用户外设。
7. **开发用户应用程序**
在这个阶段，系统已准备就绪，可以开始开发用户应用程序代码。
   a. HAL 具有直观且易用的 API 用于配置外设。它支持轮询、中断和 DMA 编程模型，可适应任何应用程序的需求。有关如何使用每个外设的更多详细信息，请参阅 STM32CubeWL MCU 软件包中提供的丰富示例集。
   b. 如果用户应用程序存在一些实时要求，将在 STM32CubeWL 内提供大量示例，说明如何使用 FreeRTOS™以及如何将其与所有中间件栈进行集成。这为开发用户应用程序提供了良好的开端。

注意： 在默认的 HAL 实现中，使用 SysTick 计时器作为时基：它会以固定的时间间隔生成中断。如果从外设 ISR 进程调用 HAL_Delay()，应确保 SysTick 中断的优先级高于外设中断的优先级（在数值上较低）。否则，ISR 中断服务程序将被阻塞。影响时基配置的函数被声明为 weak，以允许在用户文件中需要其他实现的情况下进行重写（例如，使用通用计时器或其他时间源）。若需更多详细信息，请参考 HAL_TimeBase 示例。

4.2.2.2 *LL 应用程序*

本节介绍了使用 STM32CubeWL 创建自有 LL 应用程序所需的步骤。

1. **创建工程**
   要新建一个工程，应从为*\Projects\<STM32xxx_yyy>\Templates_LL* 下为每个板提供的 Templates_LL 工程开始，或者从*\Projects\<STM32xxy_yyy>\Examples_LL* 下的任何可用工程开始（<STM32xxx_yyy> 是指板的名称，例如 NUCLEO-WL55JC）。
   Template 工程会提供空的主循环函数，但首先最好熟悉 STM32CubeWL 的工程设置。
   模板主要特性如下：
   – 它包含 LL 和 CMSIS 驱动程序的源代码，这些驱动程序是在给定板上开发代码所需的最少的组件。
   – 它包含所有必需的软件组件的包含路径。
   – 它选择支持的 STM32WL 器件并允许相应地配置 CMSIS 和 LL 驱动程序。
   – 它提供了即用型用户文件，这些文件按以下方式预先配置：
   main.h:LED 和 USER_BUTTON 定义抽象层。
   main.c：最大频率的系统时钟配置。
2. **将现有工程移植到另一个板上**
   要将现有工程移植到另一个目标板上，应从为每个板提供的 Templates_LL 工程开始，该工程位于*\Projects\<STM32xxx_yyy>\Templates_LL* 下：
   a. 选择一个 LL 示例
   要查找已部署 LL 示例的板，请参考 LL 示例 *STM32CubeProjectsList.html* 的列表。
   b. 移植 LL 示例
   ○ 复制/粘贴 *Templates_LL* 文件夹-以保留初始源文件-或直接更新现有 *Templates_LL* 工程。
   ○ 然后，移植主要是用 *Examples_LL* 目标工程替换 *Templates_LL* 文件。
   ○ 保留所有主板特定部件。为清楚起见，板特定部件已标记有特定的标签：

```
/* ==============   板特定配置代码开始   ============== */
/* ==============   板特定配置代码结束    ============== */
```

   因此主要的移植步骤如下：
   ○ 替换 *stm32wlxx_it.h* 文件
   ○ 替换 *stm32wlxx_it.c* 文件
   ○ 替换 main.h 文件并进行更新：将 LL 模板的 LED 和用户按钮的定义保留在“BOARD SPECIFIC CONFIGURATION”标签下。
   ○ 替换 main.c 文件并进行更新：
   将 SystemClock_Config()LL 模板函数的时钟配置保留在“BOARD SPECIFIC CONFIGURATION”标签下。
   根据 LED 的定义，将每个出现的 LEDx 替换为 *main.h* 中可用的另一个 LEDy。

由于这些调整，此示例可在目标板上运行。

4.2.2.3 *如何将单核示例从 CPU1 迁移到 CPU2*

STM32WL 单核示例仅可在 CPU1（CM4）上运行，而在 CPU2（CM0+）上禁用。按照以下步骤，也可以在 CPU2 上运行任何单核示例：

1. 从双核工程开始。为了简化工程创建，STM32WL 软件包中会提供一些最简单的双核工程：
   – 使用 HAL 驱动程序的双核模板：
   ...\Firmware\Projects\NUCLEO-WL55JC\Templates\DualCore
   – 使用 LL 驱动程序的双核模板：
   ...\Firmware\Projects\NUCLEO-WL55JC\Templates_LL\DualCore
2. 选择任意单核示例，以迁移到 CPU2

3. 将示例中的源文件复制到模板。
以 HAL 示例 GPIO_EXTI 迁移到双核模板为例：
   – 将 *GPIO_EXTI/Inc* 的内容复制到 *Templates\DualCore\CM0PLUS\Inc*
   – 将 *GPIO_EXTI/Src* 的内容复制到 *Templates\DualCore\CM0PLUS\Src*

注意： *CPU1 的程序与 CPU2 兼容，因此 CPU2 也可以执行，但有一些例外：*
- *IRQ 处理程序和中断特定于所选内核：在 stm32wlxx_it.h 和 stm32wlxx_it.c 文件中，必须根据所选内核自定义 IRQ 处理程序。在源文件中，还必须相应地自定义包含 NVIC 函数的中断配置。例如，IRQ 处理程序 MemManage_Handler 仅可用于 CPU1（CM4）。请参考所选 CPU 的启动文件中的 IRQ 处理程序列表。*
- *很少的外设特性特定于所选内核：请参考 HAL 和 LL 驱动程序中的编译开关 CORE_CM0PLUS。*

4. 删除 CPU1 程序中的系统时钟配置：双核模板通过 CPU1 执行系统时钟配置（通用配置），然后再执行 CPU2 启动。通过复制 CPU2 文件夹中的示例程序，CPU2 也可执行系统时钟配置。此系统时钟配置与示例要求匹配，必须保留。

注意： *另一种解决方案是在配置与示例要求匹配的情况下仅通过 CPU1 执行系统时钟配置。但是，如果从待机或关断模式中唤醒，CPU2 程序必须具有恢复系统时钟的例程。*

注意： *CPU1 的程序现在仅执行一个动作：启动 CPU2。*

5. 或者，可在 CPU2 启动后使 CPU1 进入低功耗模式。建议采用最深的低功耗模式（关断模式），以便 CPU2 根据需要管理所有系统低功耗模式（停止、待机、关断）。

使用 HAL 驱动程序使 CPU1 进入关断模式的代码示例：

```
/* Request to enter in Shutdown mode */
HAL_PWREx_EnterSHUTDOWNMode();
```

使用 LL 驱动程序使 CPU1 进入关断模式的代码示例：

```
/* Request to enter in Shutdown mode */
LL_PWR_SetPowerMode(LL_PWR_MODE_SHUTDOWN);

/* Set SLEEPDEEP bit of Cortex System Control Register */
LL_LPM_EnableDeepSleep();

/* This option is used to ensure that store operations are completed */
#if defined ( __CC_ARM)
  __force_stores();
#endif
/* Request Wait For Interrupt */
__WFI();
```

注意： 使用 CubeMX（file.ioc）重新生成双核模板将覆盖先前迁移的代码。或者，也可以更新双核模板（file.ioc）的 CubeMX 配置，以便与迁移到 CPU2 的示例匹配。

4.2.3 安全应用

此软件包随附安全应用。

4.2.3.1 KMS 应用

KMS 应用演示了密钥管理服务的使用：

- 使用 AES 密钥（嵌入式密钥）进行数据加密和解密
- 使用 RSA 密钥（嵌入式密钥）进行数据签名和验证
- 基于密钥派生（嵌入式密钥）进行数据加密和解密
- 在 PC 端生成 KMS Blob 二进制（用于导入设备上的密钥）
- 使用 AES 密钥进行加密和解密（使用 KMS Blob 导入密钥）

4.2.3.2 *SBSFU 应用*

4.2.3.2.1 STM32WL5x（双核）系列

STM32WL5x 双核器件具有类似于 TZ 的双核隔离和其他安全特性（如 HDP、WRP、RDP），因此支持安全解决方案。

SBSFU 应用演示了安全启动以及 2 个镜像的安全启动和安全固件更新（M4 用户应用以及 M0+ LPWAN 协议栈）。

SBSFU 应用附带 2 个固件更新配置：

- 2 个插槽（1 个下载插槽用于更新与所下载镜像匹配的执行插槽）
- 1 个插槽（无下载插槽，直接将所下载镜像写入到匹配的执行插槽中）。

注意： *本示例的下载部分采用 UART。因此，提供了另一个示例，以演示使用 LORA 下载的无线固件更新。*

4.2.3.2.2 STM32WLEx（单核）系列

STM32WLEx 单核器件不支持全部安全特性。

尽管如此，仍可进行固件更新，并提供以下 BFU 应用：使用 1 个插槽完成启动以及 1 个镜像的固件更新。下载通过 UART 完成。

STM32WLEx 单核器件不支持全部安全特性。

尽管如此，仍可进行固件更新，并提供 2 种配置的 BFU 应用：

- 2 个插槽（1 个下载插槽与 1 个执行插槽相关）
- 1 个插槽（无下载插槽，镜像直接写入到执行插槽中）。

4.2.4 RF 应用

软件包中提供三种 RF 应用。详情如下：

LoRaWAN®示例是使用 LoRaWAN®设备练习 LoRaWAN®协议栈和 SubGHz_Phy RF 驱动程序的 LoRaWAN®示例。这些应用位于 *Projects\NUCLEO-WL55JC\Applications*和 *Projects\B-WL5M-SUBG1\Applications*中。

- LoRaWAN\LoRaWAN_AT_Slave 实现了 LoRaWAN®调制解调器，它由外部主机利用 AT 指令接口通过 UART 进行控制。此应用可提供带和不带 KMS（密钥管理系统）的单核及双核形式。更多信息请参阅 AN5406 和 AN5481。LoRaWAN_AT_Slave 设备可连接至 STM32CubeMonitor，从而将 AT 指令发送到设备，执行数据包错误率测量等。
- LoRaWAN\LoRaWAN_End_Node 实现了将传感器数据发送到 LoRaWAN®网络服务器的 LoRaWAN®应用。此应用可提供具有可选 FreeRTOS 和 KMS 的单核及双核形式。更多信息请参阅应用笔记 AN5406：如何使用 STM32CubeWL 构建 LoRa®应用。
- LoRaWAN_FUOTA_DualCore 实现了将传感器数据发送到 LoRaWAN®网络服务器的双核 LoRaWAN®应用。另外，LoRaWAN®应用和协议栈还可以通过 LoRaWAN®网络服务器将 SBSFU 用作安全启动和安全固件更新框架，从而进行无线更新。而且，LoRaWAN_FUOTA 示例采用了 STM32WL55 提供的隔离特性，因此完全安全。更多信息请参阅应用笔记 AN5554：使用 STM32CubeWL 进行 LoRaWAN®无线固件更新。
- LoRaWAN_FUOTA 实现了将传感器数据发送到 LoRaWAN®网络服务器的单核 LoRaWAN®应用设备。另外，LoRaWAN®应用和协议栈还可以通过 LoRaWAN®网络服务器将 BFU 用作启动和固件更新框架，从而进行无线更新。
- LoRaWAN_SBSFU_1_Slot_DualCore 实现了将传感器数据发送到 LoRaWAN®网络服务器的双核安全 LoRaWAN®应用。此应用由安全启动（SB）和安全引擎提供的安全环境提供安全保护。此应用还可以利用本地加载程序 Y-modem（SFU）进行更新。

- LoRaWAN_FUOTA_DualCore_ExtFlash 实现了将传感器数据发送到 LoRaWAN®网络服务器的双核 LoRaWAN®应用。另外，LoRaWAN®应用和协议栈还可以通过 LoRaWAN®网络服务器将 SBSFU 用作安全启动和安全固件更新框架，从而进行无线更新。而且，LoRaWAN_FUOTA 示例采用了 STM32WL55 提供的隔离特性，因此完全安全。此外，用于存储所接收固件二进制的下载插槽使用外部 Flash 增加 MCU 中的可用 Flash 存储器。更多信息请参阅应用笔记 AN5554：使用 STM32CubeWL 进行 LoRaWAN®无线固件更新。

SubGHz_Phy 示例：这些应用采用了 SubGHz_Phy 无线中间件。这些应用位于 Projects\NUCLEO-WL55JC\Applications\SubGHz_Phy 和 Projects\B-WL5M- SUBG1\Applications\SubGHz_Phy 中。更多信息请参阅应用笔记 AN5406。

- SubGHz_Phy_PingPong 应用可提供单核及双核形式。PingPong 应用在两个 PingPong 设备之间实现了无线链路连接。
- SubGHz_Phy_Per 应用仅可提供单核形式。此应用在传输设备和接收设备之间实现了无线链路连接，并可以统计数据包错误率（PER）。
- SubGHz_Phy_AT_Slave 应用仅可提供单核形式。此应用的所有 RF 测试均由外部主机利用 AT 指令接口通过 UART 进行控制。
- SubGHz_Phy_LrFhss 应用仅可提供单核形式。此应用通过生成一些上行链路，实现了基于包内跳频的 LR-FHSS 调制测试应用。此应用需要与这种新物理层兼容的专用基础架构。

Sigfox™示例：这些应用实现了 Sigfox™启动以及 Sigfox™设备运行。这些应用位于工程路径 Projects\NUCLEO-WL55JC1\Applications\和 Projects\B-WL5M-SUBG1\Applications\中

- Sigfox™应用可提供单核及双核形式。双核工程可在使用或不使用 KMS 情况下运行。更多信息请参阅应用笔记 AN5480：如何使用 STM32CubeWL 构建 Sigfox™应用。
- Sigfox_AT_Slave 实现了 Sigfox™应用调制解调器，它由执行终端的计算机等外部主机利用 AT 指令接口通过 UART 进行控制。
- Sigfox_PushButton 是 Sigfox™设备在按下用户按钮时将温度和电池电量发送到 Sigfox™网络的示例。
- Sigfox_SBSFU_1_Slot_DualCore 实现了双核安全 Sigfox®应用设备将传感器数据发送到 Sigfox®网络。此应用由安全启动（SB）和安全引擎提供的安全环境提供安全保护。此应用还可以利用本地加载程序 Y-modem（SFU）进行更新。

4.2.5 RF 演示

本地网络示例采用具有一个集中器和多达 14 个传感器的非 LoRaWAN®本地网络，这些传感器可以连接至集中器并将传感器数据发送到集中器。更多信息请参阅 UM2786。本地网络示例工程位于 *Projects\NUCLEO-WL55JC\Demonstrations\LocalNetwork* 和 *Projects\B-WL5M-SUBG1\Demonstrations\LocalNetwork_Sensor* 中。

- STM32WLxx_Nucleo 板烧写一个集中器示例实现集中器每 16 秒发送一个信标帧和一个同步帧，以管理多达 14 个传感器组成的网络，并接收每个连接传感器的数据。集中器可以连接至 STM32CubeMonitor，以配置地理区域，并显示检测到的传感器列表和连接传感器的数据。
- 两个**传感器**工程实现了：
  - 将传感器数据（基于内部温度传感器和 NUCLEO-WL55JC 板上的 VBat 信息）发送到演示集中器
  - 将传感器数据（基于 B-WL5M-SUBG1 板上可用的所有传感器）发送到演示集中器。

4.2.6 获取 STM32CubeWL 版本更新

最新 STM32CubeWL MCU 软件包版本和补丁可从 www.st.com/stm32wl 获取。使用 STM32CubeMX 中的“CHECK FOR UPDATE”（检查更新）按钮可检索这些更新。关于更多详情，请参考 STM32CubeMX 的第 3 节中有关 STM32 配置和初始化 C 代码生成信息（UM1718）。

5 FAQ

5.1 STM32CubeWL 软件包的许可方案是什么？

HAL 是根据非限制性 BSD（伯克利软件发行）许可证进行发行。

Semtech 和意法半导体推出的中间件栈 LoRaWAN®和 SubGHz_Phy 根据非限制性 BSD（伯克利软件发行）许可证发行。

中间件栈 Sigfox™根据 SLA0044 和 Sigfox™具体条款发行。

中间件安全性安全引擎和密钥管理服务根据 SLA0044 发行。

基于众所周知的开源解决方案（FreeRTOS™和 FatFS）的中间件具有用户友好的许可条款。

更多详细内容，请参阅每个中间件的许可协议。

5.2 STM32CubeWL 软件包支持哪些板？

STM32CubeWL 软件包支持 NUCLEO-WL55JC1、NUCLEO-WL55JC2 和 B-WL5M-SUBG1 板。

5.3 现成的工具集工程是否随附任何示例？

是的。STM32CubeWL 提供丰富的示例和应用程序集。另外还随附适合基于 IAR™的工具链的预配置工程。

5.4 是否有与标准外设库的链接？

STM32Cube HAL 和 LL 驱动程序是标准外设库的替代程序：

- 与标准外设 API 相比，HAL 驱动程序具有更高的抽象级别。而且，更注重外设通用特性，而不是硬件。更高的抽象级别允许定义易于从一个产品移植到另一产品的一系列用户友好型 API。
- LL 驱动程序提供寄存器级别的底层 API。它们以更加简单、清晰的方式进行组织，避免了直接访问寄存器。LL 驱动程序还包括外设初始化 API，与 SPL 提供的外设 API 相比，这些 API 在功能上相似，但更加优化。与 HAL 驱动程序相比，这些 LL 初始化 API 可以更轻松地实现从 SPL 到 STM32Cube LL 驱动程序的迁移，因为每个 SPL API 都有其等效的 LL API。

5.5 HAL 层是否利用中断或 DMA？如何对此进行控制？

是的。HAL 层支持三种 API 编程模型：轮询、中断和 DMA（带有或不带有中断生成）。

5.6 如何管理产品/外设的特定功能？

HAL 驱动程序可提供扩展的 API，即作为通用 API 的附加组件的特定函数，用于仅支持某些产品/系列上可用的功能。

5.7 何时应使用 HAL 而不是 LL 驱动程序？

HAL 驱动程序可提供面向功能的高级 API，并具有高度可移植性。产品/外设的复杂性对最终用户而言是隐藏的。

LL 驱动程序提供寄存器级别的底层 API，更够更好地优化，但可移植性较差。需要深入了解产品/外设技术参数。

5.8 如何将 LL 驱动程序纳入我的环境？是否有 HAL 的 LL 配置文件？

没有配置文件。源代码必须直接包含必要的 *stm32wlxx_ll_ppp.h* 文件。

5.9 是否可以同时使用 HAL 和 LL 驱动程序？如果是，约束条件是什么？

可以同时使用 HAL 和 LL 驱动程序。利用 HAL 处理外设初始化阶段，然后利用 LL 驱动程序管理 I/O 操作。

HAL 和 LL 之间的主要区别在于 HAL 驱动程序需要创建并使用句柄进行操作管理，而 LL 驱动程序直接在外设寄存器上进行操作。HAL 和 LL 的混合使用在 *Examples_MIX* 示例中进行说明。

5.10 LL 是否具有 HAL 不具有的 API？

是的，有。

在 stm32wlxx_ll_cortex.h 中添加了一些 Cortex® API，例如用于访问 SCB 或 SysTick 寄存器。

5.11 为何未在 LL 驱动程序上启用 SysTick 中断？

在独立模式下使用 LL 驱动程序时，无需启用 SysTick 中断，因为 LL API 中没有使用这些中断，而 HAL 函数需要使用 SysTick 中断来管理超时。

5.12 如何启用 LL 初始化 API？

LL 初始化 API 和相关资源（结构、文字和原型）的定义根据 USE_FULL_LL_DRIVER 编译开关而定。

为了能够使用 LL API，请将此开关添加到工具链编译器预处理器中。

5.13 STM32CubeMX 如何基于内置软件生成代码？

STM32CubeMX 内置有 STM32 微控制器（包括其外设和软件）相关知识，可以为用户提供图形表示并根据用户配置生成*.h/*.c 文件。

5.14 如何获取有关最新 STM32CubeWL MCU 软件包版本的定期更新？

请参考第 4.2.6 节。

版本历史

表 3. 文档版本历史

日期	版本	变更
2019 年 12 月 11 日	1	初始版本。
2020 年 10 月 12 日	2	更新了“前言”一节。 更新了第 1 节“STM32CubeWL 主要特性”。 更新了图 1. STM32CubeWL 软件组件。 更新了第 2.1.2 节“硬件抽象层（HAL）和底层（LL）”。 更新了第 2.1.3 节“基本外设用例”。 更新了第 3 节“STM32CubeWL 软件包概述”。 更新了第 4.1 节“运行您的第一个示例”。 增加了第 4.2.1 节“使用 STM32CubeMX 开发或更新应用”。 重新组织了第 4.2.2 节“驱动程序应用”。 增加了第 4.2.2.3 节“如何将单核示例从 CPU1 迁移到 CPU2”。 更新了第 5.1 节“STM32CubeWL 软件的许可方案是什么？”。 更新了第 5.13 节 STM32CubeMX 如何基于内置软件生成代码？ 增加了第 5.14 节“如何获取有关最新 STM32CubeWL MCU 软件包版本的定期更新？”。
2021 年 3 月 16 日	3	将“引言”部分中的 STM32CubeMonitor-RF 替换为 STM32CubeMonitor。
2021 年 6 月 7 日	4	更新了图 5. STM32CubeWL 示例概述。 更新了第 4.2.3.2 节“SBSFU 应用” 更新了第 4.2.4 节“RF 应用”
2022 年 1 月 27 日	5	更新了表 2. STM32WL 系列的板。 更新了第 3.2 节“软件包概述”，包括图 4. STM32CubeWL 软件包结构和图 5.STM32CubeWL 示例概述。 更新了第 4.2.4 节“RF 应用”。
2022 年 11 月 03 日	6	更新了第 1 节“STM32CubeWL 主要特性”，以简化产品名称。 更新了第 2 节“STM32CubeWL 架构概述”，以简化产品名称。 更新了第 3.1 节“支持的 STM32WL 设备和硬件”，在表 1 中增加了 STM32WL5MO 设备；在表 2 中增加了 B-WL5M-SUBG1 板。 更新了第 3.2 节“软件包概述”：编辑了图 4，以增加 B-WL5M-SUBG1 板。编辑了图 5，以增加 STM32CubeWL MCU 软件包信息。增加了 B-WL5M- SUBG1 信息以及 EWARM、MDK-ARM 和 STM32CubeIDE 工具链。 更新了第 4.2.4 节“RF 应用”，以简化产品名称；增加 B-WL5M-SUBG1 信息；以及增加 LoRaWAN_FUOTA_DualCore_ExtFlash 应用信息； 更新了第 4.2.5 节“RF 演示”，以增加 B-WL5M-SUBG1 工程路径；以及增加两个传感器工程信息。 更新了第 5.2 节“STM32CubeWL 软件包支持哪些板？”中有关 NUCLEO-WL55JC1、NUCLEO-WL55JC2 和 B-WL5M-SUBG1 板的信息。

目录

表格索引

图片目录

重要通知 - 请仔细阅读

意法半导体公司及其子公司（“意法半导体”）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意法半导体产品的最新信息。意法半导体产品的销售依照订单确认时的相关意法半导体销售条款。

买方自行负责对意法半导体产品的选择和使用，意法半导体概不承担与应用协助或买方产品设计相关的任何责任。

意法半导体不对任何知识产权进行任何明示或默示的授权或许可。

转售的意法半导体产品如有不同于此处提供的信息的规定，将导致意法半导体针对该产品授予的任何保证失效。

ST 和 ST 标志是意法半导体的商标。关于意法半导体商标的其他信息，请访问 www.st.com/trademarks。其他所有产品或服务名称是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。