UM2656
用户手册面

面向STM32L5系列的STM32CubeL5入门

引言

STM32Cube源自意法半导体，旨在通过减少开发工作量、时间和成本，明显提高设计人员的生产率。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项目开发过程，其中包括：
  - STM32CubeMX，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），图形接口和命令行接口中可用的编程工具。
  - STM32CubeMonitor-Power（STM32CubeMonPwr），测量并帮助优化MCU功耗的监控工具。
- STM32Cube MCU包，针对于每个微控制器系列的综合嵌入式软件平台（例如，STM32L5系列的STM32CubeL5），它包括：
  - STM32Cube硬件抽象层（HAL），确保在STM32各个产品之间实现最大限度的可移植性。
  - STM32Cube 底层API，通过硬件提供高度用户控制，确保最佳性能和内存开销。
  - 一套一致的中间件组件，比如RTOS、USB器件、USB PD、FatFs、STMTouch™、TrustedFirmware（TF-M）、mbed-TLS和mbed加密。
  - 配备完整外设和应用示例的全部嵌入式软件实用工具。

本用户手册介绍了如何开始使用 STM32CubeL5MCU软件包。

STM32CubeL5介绍了STM32CubeL5MCU软件包的主要功能。

*第 2 节*和*第 3 节*概述了STM32CubeL5架构和MCU软件包结构。

目录

表格索引

图片目录

1 STM32CubeL5 主要特性

STM32CubeL5MCU软件包在基于具有TrustZone® 和FPU的Arm®(a) Cortex®-M33处理器的STM32 32位微控制器上运行。

STM32CubeL5将开发STM32L5系列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倡议，该组组件具有高度可移植性，不仅在STM32L5系列微控制器内，而且在其他STM32系列中也是如此。

STM32CubeL5与用于生成初始化代码的STM32CubeMX代码生成器完全兼容。该软件包包括涵盖微控制器硬件的底层（LL）和硬件抽象层（HAL）API，以及在STMicroelectronics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。

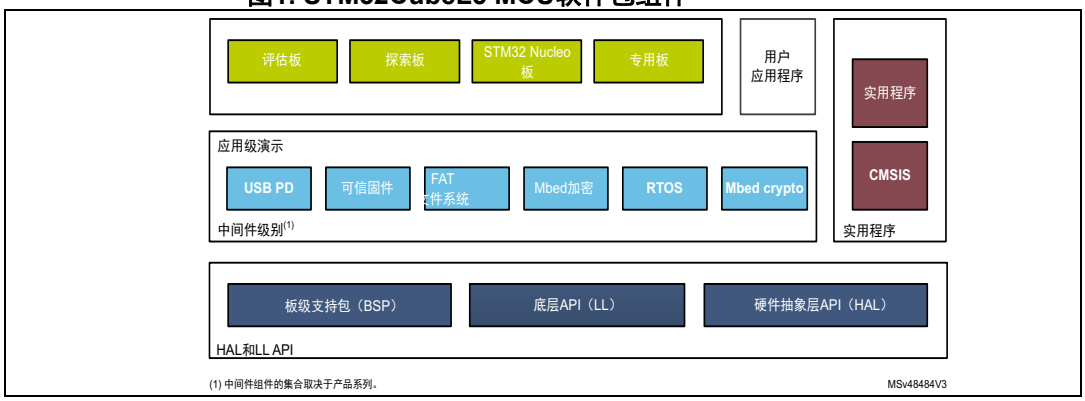
STM32CubeL5MCU软件包还包含一组中间件组件以及相应的示例。它们的许可条款免费、易用。

- 使用FreeRTOS™开源解决方案实现CMSIS-RTOS
- 完整的USB器件堆栈支持以下器件类别：HID，MSC，CDC，音频，DFU，LPM，BCD。
- USB PD库
- Arm可信固件-M（TF-M）集成解决方案
- Mbed TLS和Mbed加密库
- 基于开源FatFS解决方案的FAT文件系统
- STMTouch触摸感应库解决方案。

STM32CubeL5MCU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和演示。

STM32CubeL5 MCU软件包组件布局如*图 1*中所示。

图1. STM32CubeL5 MCU软件包组件

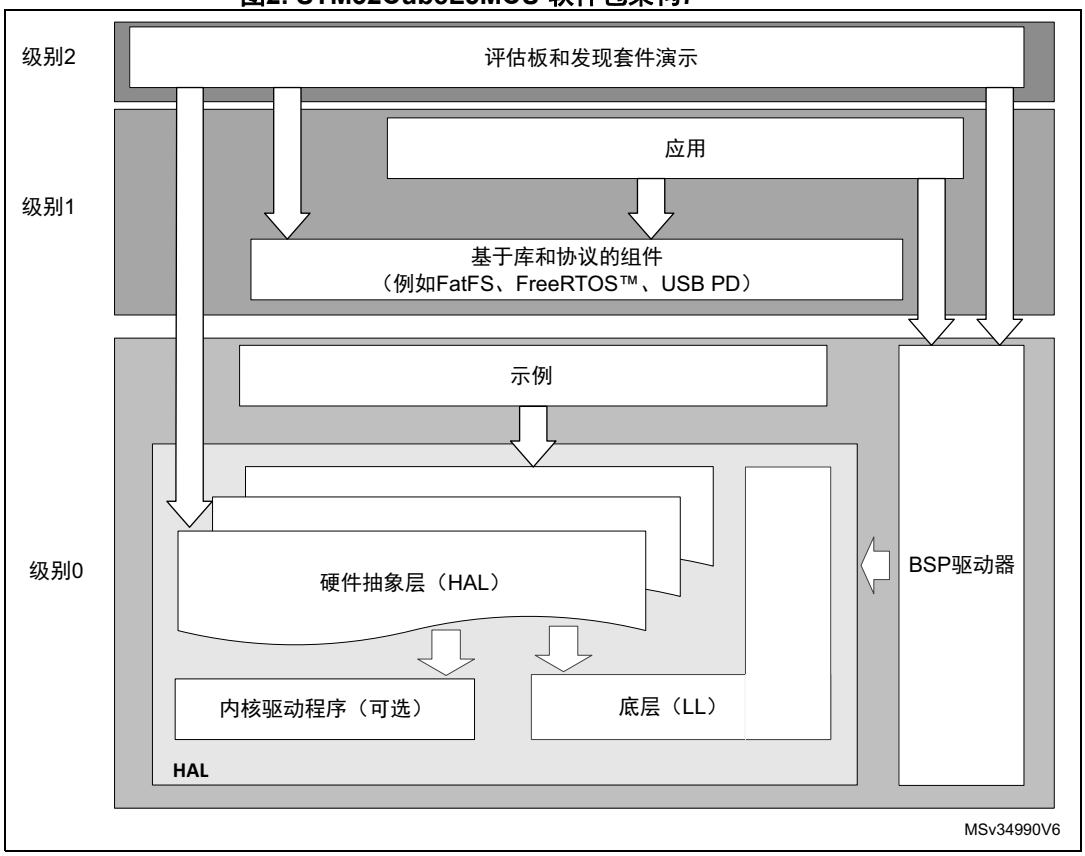


a. Arm是Arm Limited（或其子公司）在美国和或其他地区的注册商标。

2 STM32CubeL5架构概述

STM32CubeL5MCU软件包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. STM32CubeL5MCU 软件包架构7



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）

该层提供了一组特定于板上实现的每个硬件组件的API（例如LCD、音频、microSD™和MEMS驱动程序）。不支持音频和MEMS驱动程序。它包含两部分：

- 组件驱动程序；

  该驱动程序与板上的外部器件（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上。

- BSP驱动器；

  这是组件驱动程序链接到特定板的位置，以提供一组用户友好型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）

STM32CubeL5 WL HAL和LL是互补的，可满足广泛的应用要求:

- HAL驱动程序可提供高度可移植的面向高级功能的API。它们向最终用户隐藏了MCU和外设的复杂性。

  HAL驱动程序可提供面向特征的通用多实例API，这些API可提供即用型流程来简化用户应用程序的实现。例如，对于通信外设（I^2S、UART和其他外设），提供了对API进行初始化和配置的API，可基于轮询、中断或DMA进程来管理数据传输，并处理在通信过程中可能发生的通信错误。HAL驱动程序API分为两类：

  – 为所有STM32系列微控制器提供通用功能的通用API
  – 以及为特定系列或特定编号的部件提供特殊定制功能的扩展API。

- 底层API是在寄存器级别提供的底层API，这些API经过进一步优化，可移植性下降。需要对MCU和外设技术参数有更深入的了解。

  底层（LL）驱动程序旨在提供面向专家的快速轻量级层，与HAL相比，更接近硬件。与HAL相反的是，对于不关注优化访问的外设或需要大量软件配置和/或复杂上层堆栈的外设而言，LL API并不适用。

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，用于使用每个字段对应的复位值填充初始化数据结构
– 一组函数，用于复位外设（外设寄存器恢复为其默认值）
– 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。

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。

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

中间件是一组涵盖USB设备库、USB PD库、FreeRTOS™、FatFS、Arm可信固件-M（TF-M）、Mbed TLS、Mbed Crypto和STMTouch触摸感应的库。该层组件之间的水平交互是通过调用特征API来直接执行的，而与底层驱动程序的垂直交互是通过库系统调用接口中实现的特定回调函数和静态宏来管理的。例如，FatFs实现磁盘I/O驱动程序，用来访问microSD驱动器或实现USB大容量存储类。USB PD可提供最新USB C型供电服务。在USB.org规范的这一演变过程中，实现用于电源管理的专用协议。更多详细信息，请参见http://www.usb.org/developers/powerdelivery/。

每个中间件组件的主要特性如下：

- **USB设备库：**
  - 支持多种USB类别（大容量存储、HID、CDC、DFU、LPM和BCD）。
  - 支持多数据包传输功能，通过该功能可发送大量数据而无需将其拆分为最大数据包大小的传输件。
  - 无需更改库代码（可保存为只读），使用配置文件即可更改内核和库配置。
  - 32位对齐数据结构体以处理高速模式中基于DMA的传输。
  - RTOS和独立操作。
  - 使用配置文件通过抽象层与低级驱动程序链接，从而避免库和低级驱动程序之间存在任何依赖关系。
- **USB PD设备和内核库**
  - PD2和PD3规范（支持供电/受电/双重角色）
  - 快速角色交换
  - 电池电量耗尽
  - 无需更改库代码（只读），使用配置文件更改内核和库配置
  - RTOS和独立操作。
  - 使用配置文件通过抽象层与低级驱动程序链接，从而避免库和低级驱动程序之间存在任何依赖关系。
- **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
  - 支持TrustZone。
- **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
- **Arm可信固件-M（TF-M）**
  - TrustZone的Arm平台安全架构（PSA）的实现参考
  - 安全服务包括：

    安全存储服务

    认证

    加密服务

    TF-M审核日志

    平台服务

TF-M应用的示例可在*\Projects\STM32L562E-DK\Applications\TFM*下在STM32CubeL5固件包中找到。

- **Mbed TLS**
  - 基于开源的SSL/TLS安全层
- **Mbed加密**
  - 支持多种加密操作的开源密码库，包括：

    密钥管理

    哈希算法

    对称加密算法

    非对称加密算法

    消息认证（MAC）

    密钥生成和衍生

    关联数据的认证加密（AEAD）。
- **STM32触摸感应库：**
  强大的STMTouch电容式触摸感应解决方案，支持接近、触摸键、线性和旋转触摸传感器。它基于成熟的表面电荷转移采集原理。

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。

2.3 级别 2

该层级是一个全局实时图形演示单层，基于中间件服务层、低层级抽象层和用于板基功能的基本外设使用应用程序。

3 STM32CubeL5MCU软件包概述

3.1 支持的STM32L5系列器件和硬件

STM32Cube提供围绕通用架构构建的高度可移植的硬件抽象层（HAL）。它允许使用构建层原理，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，STM32CubeL5可为所有STM32L5系列提供全面支持。用户只需在*stm32l5xx.h*中定义正确的宏即可。

*表 1*显示要根据所使用的STM32L5器件定义的宏。该宏也应在编译器预处理器中定义。

表1. STM32L5系列 的宏

在*stm32l5xx.h*中定义的宏	STM32L5器件
STM32L552xx	STM32L552CC、STM32L552CE、STM32L552ME、STM32L552QC、STM32L552QE、STM32L552RC、STM32L552RE、STM32L552VC、STM32L552VE、STM32L552ZC、STM32L552ZE。
STM32L562xx	STM32L562CE、STM32L562ME、STM32L562QE、STM32L562RE、STM32L562VE、STM32L562ZE。

STM32CubeL5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 2*中列出的STMicroelectronics板上运行。

表2. STM32L5系列 的板

板	板级STM32L5支持的器件
NUCLEO-L552ZE-Q	STM32L552xx
STM32L552E-EV	STM32L552xx
STM32L562E-DK	STM32L562xx

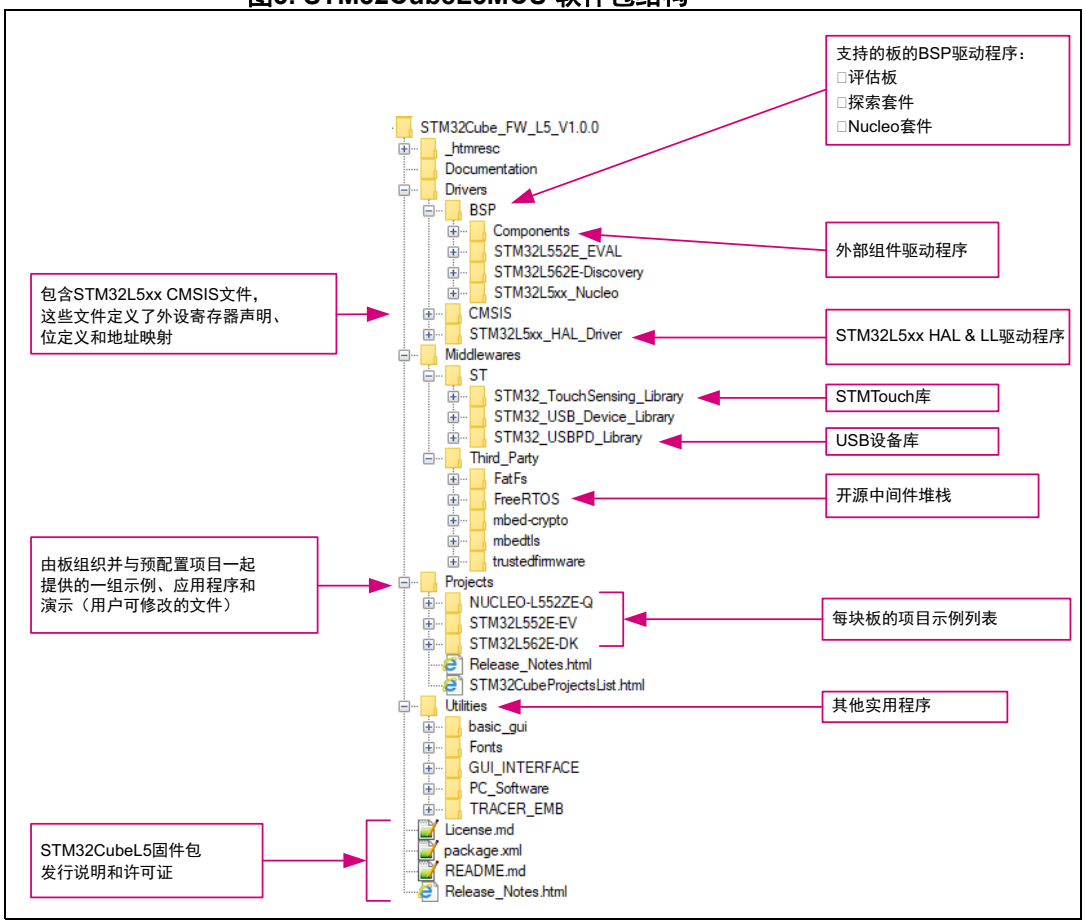
STM32CubeL5支持在以上*表 2*中列出的Nucleo-144板。

STM32CubeL5MCU软件包可以在任何兼容的硬件上运行。如果用户自己的板具有相同的硬件功能（LED、LCD显示屏、按钮...），用户只需更新BSP驱动程序，即可将所提供的示例移植到自己的板上。

3.2 MCUMCU软件包概述

STM32CubeL5MCU软件包解决方案以一个单一的zip软件包提供，其结构如*图 3*中所示。

图3. STM32CubeL5MCU 软件包结构

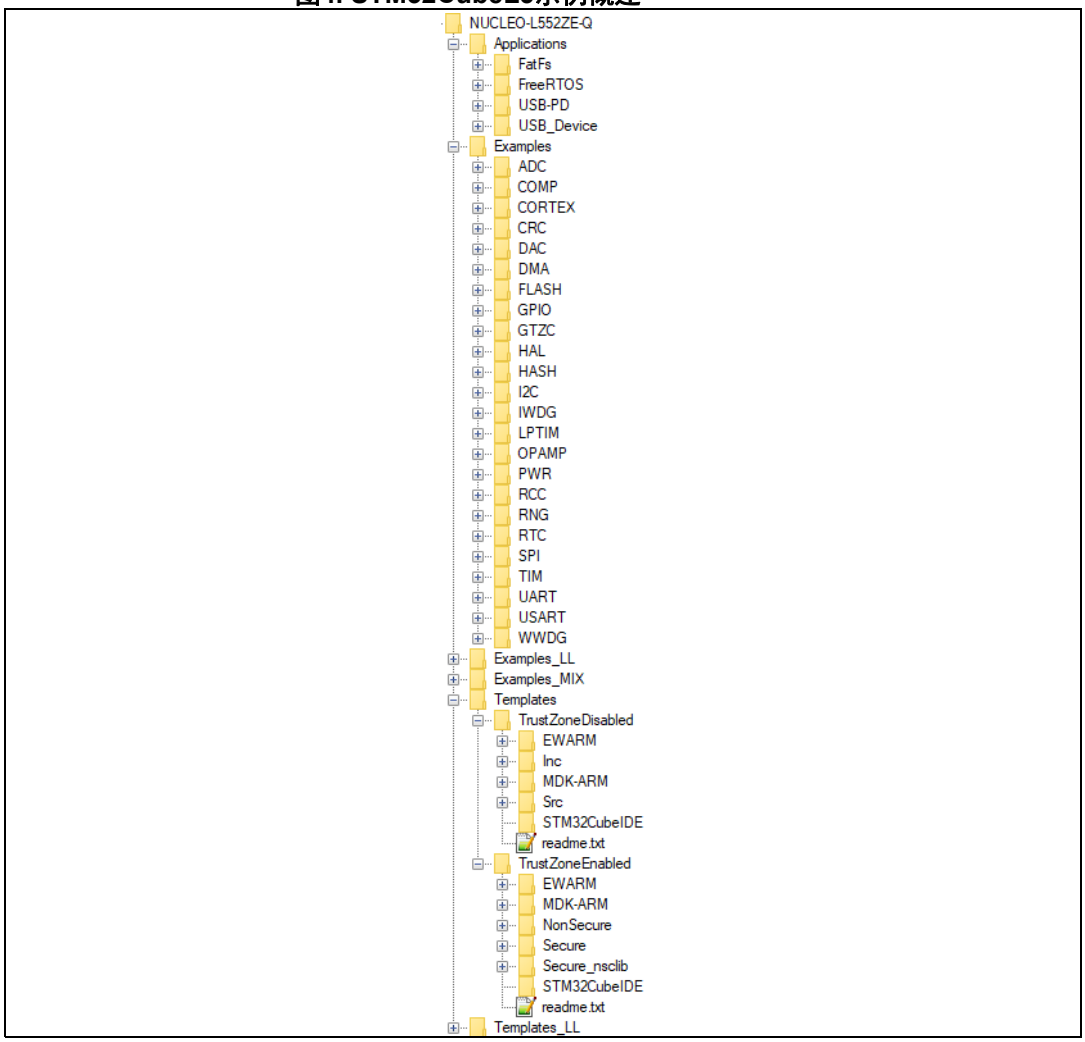


1. 用户不得修改组件文件。用户只可编辑*Projects*源。

对于每个板，提供了一组示例，其中包含用于EWARM、MDK-ARM和STM32CubeIDE工具链的预配置项目。

*图 4*显示NUCLEO-G071RB板的项目结构。

图4. STM32CubeL5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的示例被称为“*Examples*”、“*Examples_LL*”和“*Examples_MIX*”。它们分别使用HAL驱动程序、LL驱动程序以及HAL和LL驱动程序的组合，没有任何中间件组件。
- 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目录中提供的模板项目，可在给定的板上快速构建任何固件应用。

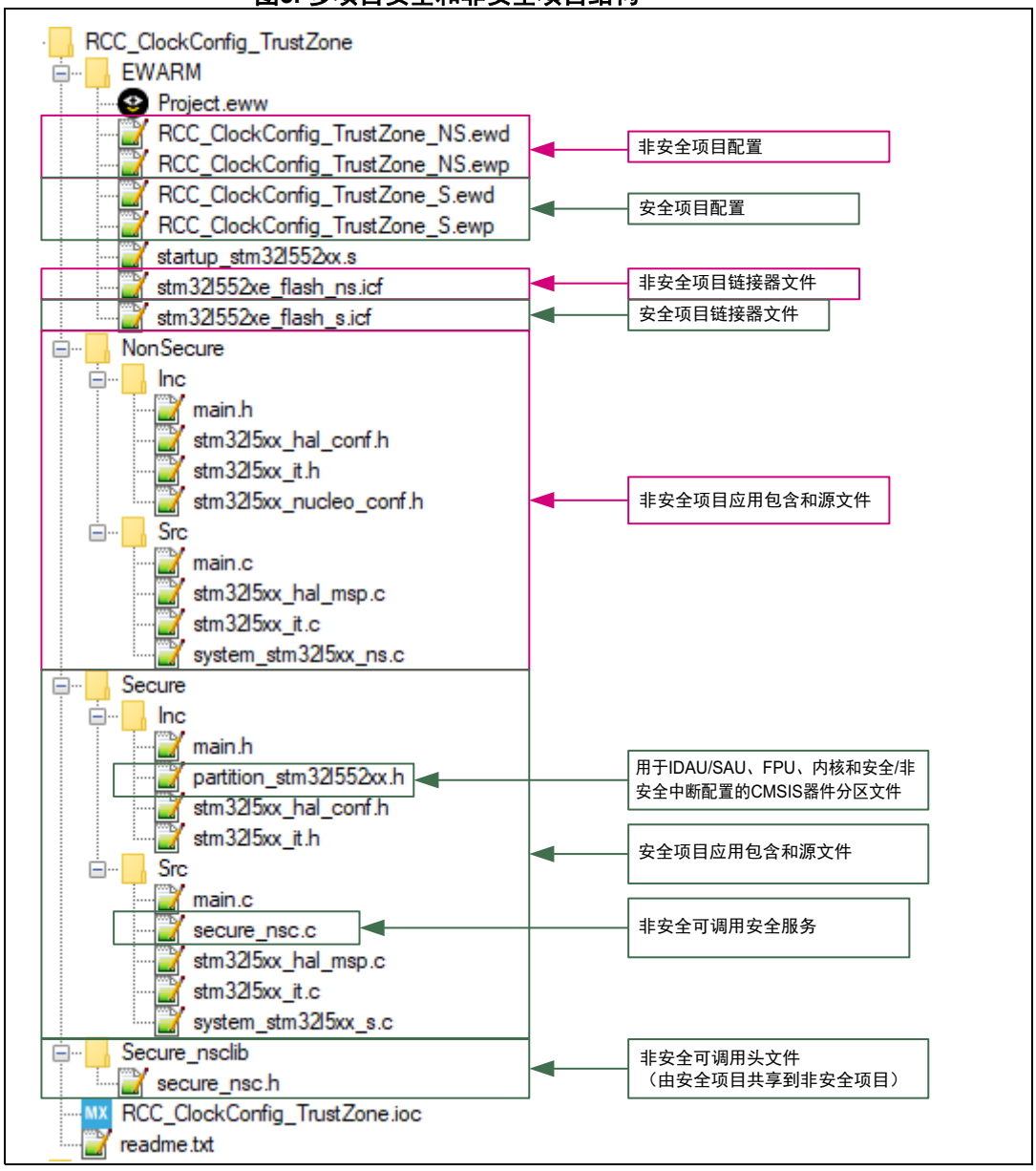
3.2.1 启用TrustZone的项目

启用TrustZone的“*Examples*”名称以“_TrustZone”为前缀。该规则也适用于“*Applications*”（TFM除外，其本身用于TrustZone）。

启用TrustZone的“Examples”和“Applications”具有多项目结构，该结构由安全子项目和非安全子项目组成，如以下*图 5*中所示。

启用TrustZone的项目是根据CMSIS-5器件模板开发的，该模板已扩展为包括系统分区可听文件partition_<device>.h，主要负责安全执行状态下安全属性单元（SAU）、FPU和安全/非安全中断分配的设置。该设置是在进入安全应用程序main()函数之前，在启动时调用的安全CMSIS SystemInit()函数中执行的（请参阅软件指南“Arm TrustZone-M”文档）。

图5. 多项目安全和非安全项目结构



STM32CubeL5软件包固件包在*partition_<device>.h*文件中提供了默认的存储器分区，这些文件位于以下位置：
\Drivers\CMSIS\Device\STM32L5\Include\Templates。
默认SAU区域定义如下：

- SAU区域0：0x0C03E000-0x0C03FFFF（安全、非安全可调用）
- SAU区域1：0x08040000-0x0807FFFF（非安全FLASH Bank2（256 KB））
- SAU区域2：0x20018000-0x2003FFFF（非安全RAM（下半个SRAM1 + SRAM2（160 KB））
- SAU区域3：0x40000000-0x4FFFFFFF（非安全外设映射的存储器）
- SAU区域4：0x60000000-0x9FFFFFFF（非安全外部存储器）
- SAU区域5：0x0BF90000-0x0BFA8FFF（非安全系统存储器）

为了匹配默认分区，STM32L5xx系列器件应设置以下用户选项字节：

- TZEN=1 （启用TrustZone的器件）
- DBANK=1 （双组Flash配置）
- SECWM1_PSTRT=0x0 SECWM1_PEND=0x7F（将Flash Bank1的全部128页设置为安全页面）
- SECWM2_PSTRT=0x1 SECWM2_PEND=0x0（未将Flash Bank2的页面设置为安全页面，因此Bank2不安全）。

注： *在默认情况下，内部Flash在TZEN=1时是完全安全的，并且应根据应用存储器配置设置用户选项字节SECWM1_PSTRT/SECWM1_PEND和SECWM2_PSTRT/SECWM2_PEND（SAU区域和安全/非安全应用程序项目链接器文件也需要对齐）。*

所有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项目。
- *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中的大多数固件示例（从STM32CubeMX 5.4.0开始）

*表 3*提供了每个板可用的项目数。

表3. 每个板的示例数

级别	STM32L562E-DK	STM32L552E-EV	NUCLEO-L552ZE-Q	总计
Templates_LL	1	1	1	3
模板	2	2	2	6
Examples_MIX	0	0	11	11
Examples_LL	2	0	67	69
示例	61	50	93	204
演示	0	1	0	1
应用	13	21	14	48
总计	79	75	188	**342**

4 入门STM32CubeL5

4.1 运行第一示例

本节介绍了在STM32L5板上运行第一个示例的简便性。该程序只需切换NUCLEO-G071RB板上的LED：

下载STM32CubeL5MCU软件包。将其解压缩到适当的目录中。确保*图 3*中所示的软件包结构未被修改。请注意，还建议将软件包复制到尽可能靠近根卷的位置（例如C\ST或G:\Tests），因为某些IDE在路径长度过长时会遇到问题。

4.1.1 运行首次启用TrustZone的示例

在加载和运行启用TrustZone的示例之前，应阅读示例自述文件以了解任何特定配置，以确保如3.2.1节中所述启用安全性（TZEN=1（用户选项字节））。

1. 浏览到 *\Projects\NUCLEO-L552ZE-Q\Examples*。
2. 打开*\GPIO*，然后打开*\GPIO_IoToggle_TrustZone*文件夹。
3. 使用首选工具链打开项目。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。
4. 依次重建所有安全和非安全项目文件，并将安全和非安全映像加载到目标存储器中。
5. 运行示例：安全应用程序定期每秒切换一次LED1，非安全应用程序以同样的速度定期切换两次LED2（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
  a) 在示例文件夹下，打开*EWARM*子文件夹
  b) 启动Project.eww工作区
  c) 重建xxxxx_S安全项目文件：**Project-> Rebuild all**
  d) 将xxxxx_NS非安全项目设置为“Active”应用程序（右键单击“xxxxx_NS”项目 **Set as Active**）
  e) 重建xxxxx_NS非安全项目文件：**Project-> Rebuild all**
  f) 使用Project-> Download-> Download active application刷新非安全二进制文件
  g) 将xxxxx_S设置为活动应用程序（右键单击xxxxx_S项目 **Set as Active**）
  h) 使用“**Download”和“Debug**”按钮（Ctrl+D）刷新安全二进制文件
  i) 运行程序：**Debug->Go(F5)**
- MDK-ARM
  a) 打开您的MDK-ARM工具链
  b) 打开多项目工作区文件Project.uvmpw
  c) 选择xxxxx_s项目作为活动项目（**Set as Active Project**）
  d) 构建xxxxx_s项目
  e) 选择xxxxx_ns项目作为活动项目（**Set as Active Project**）
  f) 构建xxxxx_ns项目
  g) 加载非安全二进制代码（**F8**）
  （这应将\MDK-ARM\xxxxx_ns\Exe\xxxxx_ns.axf下载到闪存）
  h) 选择Project_s项目作为活动项目（**Set as Active Project**）
  i) 加载安全二进制代码（**F8**）
  （这应将\MDK-ARM \xxxxx_s \Exe \xxxxx_s.axf下载到闪存）
  j) 运行示例

- STM32CubeIDE
  a) 打开您的STM32CubeIDE工具链
  b) 打开多项目工作区文件.project
  c) 重建xxxxx_安全项目
  d) 重建xxxxx_非安全项目。
  e) 为安全项目启动“Debug as STM32 Cortex-M C/C++ Application”。
  在“Edit configuration”窗口中，选择“startup”面板，并添加非安全项目符号的加载图像。
  请注意，非安全项目应在安全项目之前加载。
  然后单击“OK”。
  f) 在调试透视图中运行示例。

4.1.2 运行首次禁用TrustZone的示例

在加载和运行禁用TrustZone的示例之前，应阅读示例自述文件以了解任何特定配置，或者如果未提及任何内容，应确保板器件已禁用安全性（TZEN=0（用户选项字节））。有关对TZEN=0进行可选回归的信息，请参阅FAQ。

1. 浏览到 *\Projects\NUCLEO-L552ZE-Q\Examples*。
2. 打开*\GPIO*，然后打开*\GPIO_EXTI*文件夹。
3. 使用首选工具链打开项目。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。
4. 重新编译所有文件，并将二进制文件加载到目标内存中。
5. 运行示例：每次按下USER按钮，LED1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
  a) 在示例文件夹下，打开*\EWARM*子文件夹
  b) 启动Project.eww工作区[a]
  c) 重建所有文件：**Project-> Rebuild all**
  d) 加载项目图像：**Project-> Debug**
  e) 运行程序：**Debug->Go(F5)**
- MDK-ARM
  a) 在示例文件夹下，打开*\MDK-ARM*子文件夹
  b) 启动Project.uvprojx工作区[a]
  c) 重建所有文件：**Project->Rebuild all target file**
  d) 加载项目图像：**Debug->Start/Stop Debug Session**
  e) 运行程序：**Debug->Run （F5）**。
- STM32CubeIDE
  a) 打开STM32CubeIDE工具链
  b) 单击“**File->Switch Workspace->Other**”，浏览到STM32CubeIDE工作区目录
  c) 单击“**File->Import**”，选择“**General->Existing Projects into Workspace**”，然后单击“**Next**”
  d) 浏览到STM32CubeIDE工作区目录并选择项目
  e) 重建所有项目文件：在“**Project explorer**”窗口中选择项目，然后单击“**Project->build project**”菜单
  f) 运行程序：Run->Debug（F11）

a. 工作区名称可能因示例而变化。

4.2 开发自定义应用

4.2.1 使用STM32CubeMX开发或更新应用

在STM32CubeL5MCU软件包中，生成几乎所有的示例项目时都使用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项目需要STM32CubeMX5.5.0或更高版本：

- 安装STM32CubeMX后，打开并根据需要更新建议的项目。打开现有项目的最简单方法是双击*.*ioc*文件，以便STM32CubeMX自动打开项目及其源文件。
- 此类项目的初始化源代码由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，STM32CubeMX则更新代码的初始化部分，但保留主应用程序源代码。

要在STM32CubeMX中开发自定义项目，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. 使用管脚冲突解决器、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）和中间件堆栈（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。用户代码将保留在下一代代码中。

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STM32CubeL5的可用示例项目的列表，请参阅应用笔记AN5424“STM32L5*系列STM32Cube固件示例*”。

4.2.2 HAL应用程序

本节介绍了使用STM32CubeL5创建自定义HAL应用程序所需的步骤：

1. **创建一个项目**

   要新建一个项目，请从*\Projects\<STM32xxx_yyy>\Templates*下为每个板提供的*模板*项目开始，或者从*\Projects\<STM32xxy_yyy>\Examples*或*\Projects\<STM32xx_yyy>\Applications*（其中*<STM32xxx_yyy>*是指板名称，例如STM32G081B-EVAL）下的任何可用项目开始。

   *模板*项目具有清空主循环功能，而这也是了解STM32CubeL5项目设置的一个很好的起点。模板具有下列特性：

   – 它包含HAL源代码；CMSIS和BSP驱动程序是在给定板上开发代码所需的最少组件集。
   – 它包含所有固件组件的包含路径。
   – 它定义了所支持的STM32L5器件，从而可以正确配置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： *将现有项目复制到另一个位置时，应确保所有包含路径均已更新。*

2. **将所需的中间件添加到项目中（可选）**

   可用的中间件堆栈包括：USB设备库，USB PD库，FreeRTOS™，FatFS和STM触摸感应库。要确定要添加到项目文件列表中的源文件，请参阅每个中间件随附的文档。请参阅*\Projects\STM32xxx_yyy\Applications\<MW_Stack>*下的应用程序（其中*<MW_Stack>*是指中间件堆栈，例如USB_Device），以了解应添加哪些源文件和哪些包含路径。

3. **配置固件组件**

   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，必须将其复制到项目文件夹中（该配置文件的名称通常为*xxx_conf_template.h*，在将其复制到项目文件夹中时，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 s*tm32l5xx_hal_conf.h*中定义的宏）。

b) 通过配置SysTick，以*stm32l5xx_hal_conf.h*中限定的SysTick中断优先级TICK_INT_PRIO产生一毫秒的中断，该中断由MSI计时（在该阶段，时钟尚未配置，因此系统通过内部MSI以4 MHz的频率运行）。

c) 设置NVIC组优先级到0。

d) 调用*stm32l5xx_hal_msp.c*用户文件中定义的HAL_MspInit()回调函数，以执行全局低级硬件初始化。

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源和因素。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、闪存延迟以及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) 如果计划使用外设中断或DMA，则写入进程完成回调函数。

d) 在*main.c*文件中，初始化外设句柄结构，然后调用函数HAL_PPP_Init()以初始化外设。

7. **开发一个应用程序**

在这个阶段，系统已准备就绪，可以开始开发应用程序代码。

– 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STM32CubeL5MCU软件包中提供的丰富示例集。

– 如果应用程序存在实时约束，会有大量示例说明如何使用FreeRTOS™以及如何将其与STM32CubeL5中提供的所有中间件堆栈进行集成。这为开发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覆盖（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。

4.2.3 LL应用程序

本节介绍了使用STM32CubeL5创建自定义LL应用程序所需的步骤。

1. 创建一个项目

要新建一个项目，应从为*\Projects\<STM32xxx_yyy>\Templates_LL*下为每个板提供的*Templates_LL*项目开始，或者从*\Projects\<STM32xxy_yyy>\Examples_LL*下的任何可用项目开始（*<STM32xxx_yyy>*是指板的名称，例如）。

*模板*项目具有清空主循环功能，这是了解STM32CubeL5的项目设置的良好起点。

*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。

– 它包含所有必需的固件组件的包含路径。

– 它选择所支持的STM32L5器件，并可以正确配置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. 将现有项目移植到另一个板上

要将现有项目移植到另一个目标板上，应从为每个板提供的*Templates_LL*项目开始，该项目位于*\Projects\<STM32xxx_yyy>\Templates_LL*下：

a) 选择一个LL示例

要查找在其上部署LL示例的板，请参阅；要查找LL示例*STM32CubeProjectsList.html*列表，请参阅*表 3：每个板的示例数*，或参阅应用笔记AN5424“*STM32L5系列*的STM32Cube固件示例”。

b) 移植LL示例：

– 复制/粘贴*Templates_LL*文件夹-以保留原始源-或直接更新现有*Templates_LL*项目。

– 然后，移植主要是用*Examples_LL*目标项目替换*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*stm32l5xx_it.h*文件

– 替换*stm32l5xx_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。

通过这些修改，该示例现在可以在目标板上运行。

4.3 获取STM32CubeL5版本更新

最新STM32CubeL5MCU软件包版本和补丁可从www.st.com/stm32l5获取。这些版本和补丁可以从STM32CubeMX中的“CHECKFORUPDATE”按钮中检索。有关更多详细信息，请参阅UM1718 第3节“*STM32CubeMX用于STM32配置和初始化C代码生成*”。

5 FAQ

5.1 STM32CubeL5MCU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。

意法半导体制造的中间件堆栈（例如：USB设备库）带有许可模型，只要该模型在意法半导体的器件上运行，就可以轻松地重复使用。

基于知名开源解决方案（FreeRTOS™和FatFS）的中间件的许可条款对用户友好。有关更多详细信息，请参阅相应的中间件许可协议。

5.2 STM32CubeL5MCU软件包支持哪些板？

STM32CubeL5MCU软件包为以下STM32L5板提供BSP驱动程序和即用型示例：

- NUCLEO-G071RB
- STM32G071B-DISCO
- STM32G0316-DISCO

5.3 现成的工具集项目是否随附任何示例？

是的。STM32CubeL5提供丰富的示例和应用程序集。它们具有用于IAR™、Keil®和基于GCC的工具链的预配置项目。

5.4 如何在STM32L5系列器件上启用TrustZone？

所有STM32L5器件均支持TrustZone。出厂默认状态为禁用TrustZone。通过FLASH_OPTR寄存器中的TZEN选项位激活TrustZone安全性。用户选项字节配置可以通过STM32CubeProgrammer完成。

5.5 如何在STM32L5系列器件上禁用TrustZone？

除非将读出保护（RDP）级别设置为2，否则可以将TZEN选项位还原为0，从而禁用所有STM32L5器件上的TrustZone选项。第一步是设置RDP级别1保护，然后将TZEN位复位为0并将RDP级别复位为0，然后应用选项字节更新（所有这些操作都可以通过STM32CubeProgrammer完成）。

5.6 如何更新安全/非安全存储器映射

对于安全应用程序和非安全应用程序的存储器隔离，安全应用程序和非安全应用程序共享相同的内部闪存和内置SRAM。

STM32CubeL5固件包在partition_<device>.h文件中提供了默认的存储器分区，这些文件位于以下位置：

\Drivers\CMSIS\Device\STM32L5\Include\Templates（请参阅*第 3.2.1 节*）。对安全应用程序和非安全应用程序之间的任何存储器映射分区进行更改时，都需要进行以下更新和调整（安全和非安全存储器空间之间无重叠，并且使用安全和非安全存储器地址别名）：

- SAU非安全区域更新（内部Flash和SRAM）（请参阅partition_stm32l5xx.h文件）。
- 更新安全和非安全链接器文件，以正确定位安全和非安全代码和数据。
- 更新非安全地址，跳转到（在非安全链接器文件中的安全main.c和非安全复位处理程序中）
- 更新Flash水印选项字节（SEC_WMx_PSTRT/SEC_WMx_PEND），以定义安全/非安全Flash区域（使用STM32CubeProgrammer）。

5.7 为何要输入SecureFault_Handler()？

如果通过安全代码使用SCB->SHCSR |= SCB_SHCSR_SECUREFAULTENA_Msk启用SecureFault处理程序，则可以访问SecureFault_Handler()。

在应用程序执行期间，任何到SecureFault_Handler()的跳转都是因为在IDAU/SAU级别检测到了安全冲突，例如将非安全应用程序提取到安全地址。

如果未启用SecureFault处理程序，安全冲突将升级到HardFault处理程序。

5.8 是否有与标准外设库的链接？

STM32CubeHAL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。这些驱动程序的重点在于外设（而不是硬件）的通用功能。一组用户友好型API可以达到更高的抽象级别，从而使其易于从一个产品移植到另一个产品。
- LL驱动程序提供了底层寄存器级别的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STM32CubeLL驱动程序的迁移，因为每个SPL API都有其等效的LL API。

5.9 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）。

5.10 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。

5.11 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。

5.12 如何定期获取最新STM32CubeL5的更新MCU软件包版本？

请参见*第 4.3 节*。

5.13 何时应使用HAL与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/IP的复杂性对最终用户而言是隐藏的。

LL驱动程序提供了底层寄存器级别的API，具有更好的优化功能但移植性较差。它们需要深入了解产品/IP规范。

5.14 如何将LL驱动程序纳入现有环境中？是否有HAL的LL配置文件？

没有配置文件。源代码应直接包含必要的*stm32l5xx_ll_ppp.h*文件。

5.15 HAL和LL驱动程序是否可以一起使用？如果是，约束条件是什么？

可以同时使用HAL和LL驱动程序。将HAL用于IP初始化阶段，然后使用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。

5.16 LL是否具有HAL不具有的API？

是的，有。在stm32l5xx_ll_cortex.h中添加了一些Cortex®API，例如用于访问SCB或SysTick寄存器。

5.17 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。

5.18 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。

6 版本历史

表4. 文档版本历史

日期	版本	变更
2019年12月12日	1	初始版本

表5. 中文文档版本历史

日期	版本	变更
2021年7月26日	1	中文初始版本

重要通知 - 请仔细阅读

意法半导体公司及其子公司 （“ST”）保留随时对 ST 产品和 / 或本文档进行变更、更正、增强、修改和改进的权利，恕不另行通知。买方在订货之前应获取关于 ST 产品的最新信息。 ST 产品的销售依照订单确认时的相关 ST 销售条款。

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

ST 不对任何知识产权进行任何明示或默示的授权或许可。

转售的 ST 产品如有不同于此处提供的信息的规定，将导致 ST 针对该产品授予的任何保证失效。

ST 和 ST 徽标是 ST 的商标。若需 ST 商标的更多信息，请参考 *www.st.com/trademarks*。所有其他产品或服务名称均为其各自所有者的财产。

本文档中的信息取代本文档所有早期版本中提供的信息。