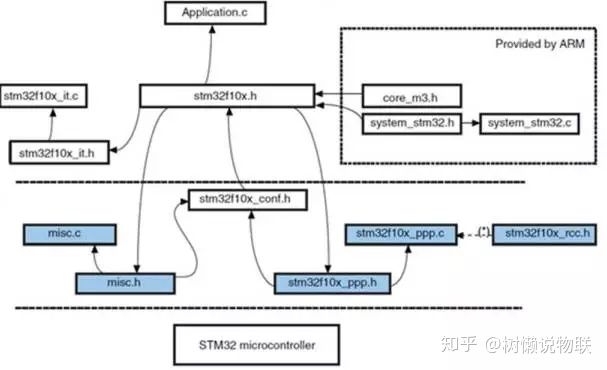
下面简单介绍各个文件件及文件的作用。

* \_htmresc 文件夹：存放ST公司的LOGO图标，这个文件夹不用管。
* **Libraries 文件夹**：在这个文件夹内有两个子目录，CMSIS文件夹用于存放符合CMSIS标准的文件，包括STM32启动文件、ARM Cortex内核文件和对应外设头文件stm32f10x.h。STM32F10x\_StdPeriph\_Driver文件夹用于存放STM32外设驱动文件，inc目录用于存放外设的头文件，src目录用于存放外设的源文件。从这些源文件的命名就可以知道对应文件的功能，比如stm32f10x\_gpio.c文件，包含了对STM32的GPIO寄存器的操作函数等，如果要对STM32 GPIO操作可以调用这个文件内的函数，但是要记得添加对应的头文件，如stm32f10x\_gpio.h。此文件夹内文件在后面创建库函数模板的时候会重点使用。
* **Project文件夹**：此文件夹下有2个目录，STM32F10x\_StdPeriph\_Examples文件夹存放的是ST公司提供的外设驱动例程，在开发过程中可以借鉴这些例程快速构建自己的外设驱动。有时间大家可以看看，里面的编程思路对我们还是很有帮助的。STM32F10x\_StdPeriph\_Templates文件夹存放的是官方的固件库工程模板， 我们后面创建自己工程模板的时候就需要复制此文件夹内的几个文件。
* Utilities文件夹：此文件夹是ST官方评估板的一些源文件，这里不需要管。
* 固件库文件夹根目录下有一个stm32f10x\_stdperiph\_lib\_um.chm文件，这个文件是固件库的**帮助文档**，可以直接双击打开。这个文档对于后面学习库函数是非常有帮助的， 只不过这里只有英文版的， 但是里面的英文都很好理解，如果学习过程中有不明白的单词，都可以通过百度翻译等软件查询。

**文件介绍**  
在介绍库文件之前先看下它们文件之间的关系，如下图所示。

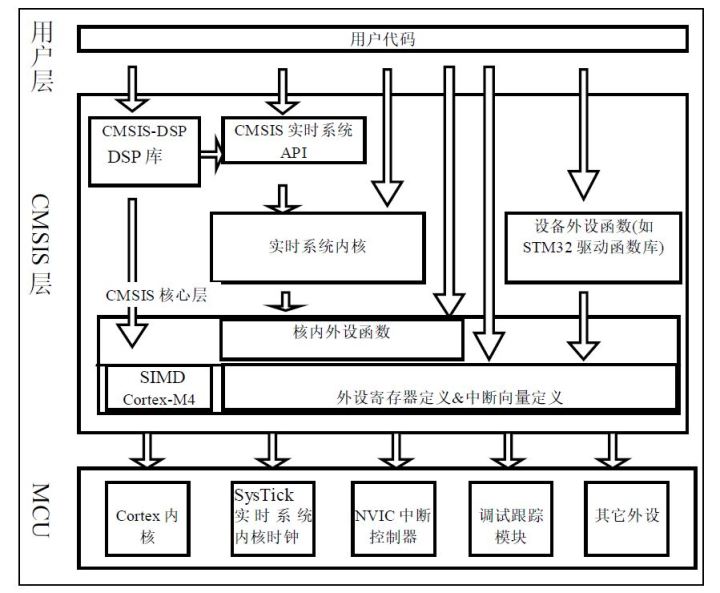


core\_cm3.h 文件位于“\STM32 最新固件库v3.5\Libraries\CMSIS\CM3\CoreSupport”目录下，说明此文件属于CMSIS标准文件，是用来提供进入M3内核的接口文件，属于CMSIS的核心文件，由ARM提供。对于所有M3内核的芯片来说这个文件都是相同的，不需要我们修改。  
stm32f10x.h 、system\_stm32f10x.h和system\_stm32f10x.c文件存放在“\STM32最新固件库v3.5\Libraries\CMSIS\CM3\DeviceSupport\ST\STM32F10x”目录下，**system\_stm32f10x.h是片上外设接入层系统头文件**。主要是申明设置系统及总线时钟相关的函数。与其对应的源文件是system\_stm32f10x.c。这个文件里面  
有一个非常重要的SystemInit()函数申明，这个函数在我们系统启动的时候都会调用，用来设置系统的整个系统和总线时钟。而stm32f10x.h是STM32F10x的头文件，类似于51单片机的reg.51，在开发STM32F10x程序的时候基本上都会调用这个头文件，可见其重要性。此文件内部封装了STM32的总线、内存和外设寄存器等，同时该文件还包含了一些时钟相关的定义和中断相关定义等。  
**stm32f10x\_ppp.c文件是STM32外设的驱动源文件**， 比如stm32f10x\_gpio.c文件。 里面已经封装好操作GPIO外设底层的内容， 提供给我们使用的是一些API函数。stm32f10x\_ppp.h就是对应的头文件。还有stm32f10x\_rcc.c、misc.c和misc.h文件他们都是存放在“\STM32最新固件库v3.5\Libraries\STM32F10x\_StdPeriph\_Driver”内。  
stm32f10x\_it.c文件用于存放中断函数，不过中断函数也可以放在其他工程文件内，所以这个文件很少操作，对应的stm32f10x\_it.h文件是它的头文件。 **stm32f10x\_conf.h文件是配置文件**，用于删减我们使用的外设头文件，比如使用GPIO外设，那么就需要调用stm32f10x\_gpio.h头文件，如果不使用GPIO外设，可以将此头文件注释掉，一般情况下我们不会对这个配置文件操作，因为如果不使用一个外设，可以在工程内不调用即可。这几个文件存在“\STM32 最新固件库v3.5\Project\STM32F10x\_StdPeriph\_Template”内。  
Application.c文件用于存放用户编写的应用程序，文件名可以根据个人爱好命名。我们通常会命名为main.c，表示存放我们的主函数代码。  
在后面创建工程模板时，添加这些文件还不够，还要将STM32的启动文件添加进来，否则系统不能启动。ST固件库提供的启动文件有很多，需根据使用的STM32芯片来选择，若开发板上使用的是高容量的STM32F1 芯片，选择startup\_stm32f10x\_hd.s。启动文件startup\_stm32f10x\_hd.s存放在“\STM32最新固件库v3.5\Libraries\CMSIS\CM3\DeviceSupport\ST\STM32F10x\startup\arm”内。

**CMSIS 标准及库层次关系**

因为基于Cortex 系列芯片采用的内核都是相同的，区别主要为核外的片上外设的差异，这些差异却导致软件在同内核，不同外设的芯片上移植困难。为了解决不同的芯片厂商生产的Cortex 微控制器软件 的兼容性问题，ARM 与芯片厂商建立了CMSIS 标准(CortexMicroController Software Interface Standard)。

所谓CMSIS 标准，实际是新建了一个软件抽象层。



CMSIS 标准中最主要的为CMSIS 核心层，它包括了：

* 内核函数层：其中包含用于访问内核寄存器的名称、地址定义，主要由ARM 公司提供。
* 设备外设访问层：提供了片上的核外外设的地址和中断定义，主要由芯片生产商提供。

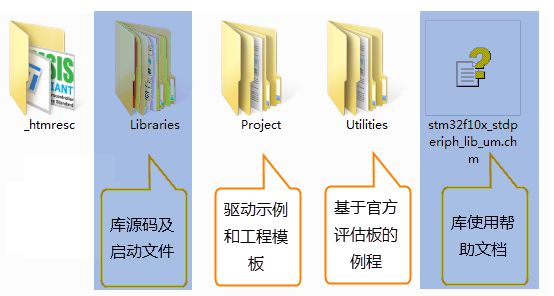
可见CMSIS 层位于硬件层与操作系统或用户层之间，提供了与芯片生产商无关的硬件抽象层，可以为接口外设、实时操作系统提供简单的处理器软件接口，屏蔽了硬件差异，这对软件的移植是有极大的好处的。STM32 的库，就是按照CMSIS 标准建立的。

**库目录、文件简介**

STM32 标准库可以从官网获得，也可以直接从本书的配套资料得到。本书讲解的例程全部采用3.5.0 库文件。以下内容请大家打开STM32 标准库文件配合阅读。

解压库文件后进入其目录：

“STM32F10x\_StdPeriph\_Lib\_V3.5.0\”

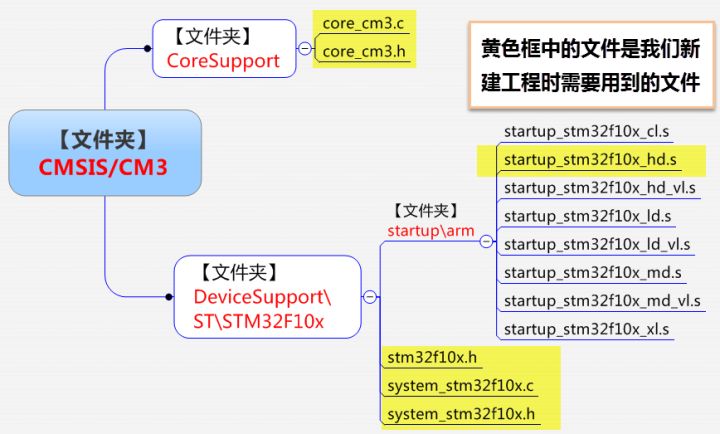


* Libraries：文件夹下是驱动库的源代码及启动文件，这个非常重要，我们要使用的固件库就在这个文件夹里面。。
* Project ：文件夹下是用驱动库写的例子和工程模板，其中那些为每个外设写好的例程对我们非常有用，我们在学习的时候就可以参考这里面的例程，非常全面，简直就是穷尽了外设的所有功能。
* Utilities：包含了基于ST 官方实验板的例程，不需要用到，略过即可。
* stm32f10x\_stdperiph\_lib\_um.chm： 库帮助文档，这个很有用，不喜欢直接看源码的可以在合理查询每个外设的函数说明，非常详细。这是一个已经编译好的HTML 文件，主要讲述如何使用驱动库来编写自己的应用程序。说得形象一点，这个HTML 就是告诉我们：ST 公司已经为你写好了每个外设的驱动了，想知道如何运用这些例子就来向我求救吧。不幸的是，这个帮助文档是英文的，这对很多英文不好的朋友来说是一个很大的障碍。但这里要告诉大家，英文仅仅是一种工具，绝对不能让它成为我们学习的障碍。其实这些英文还是很简单的，我们需要的是拿下它的勇气。

在使用库开发时，我们需要把libraries 目录下的库函数文件添加到工程中，并查阅库帮助文档来了解ST 提供的库函数，这个文档说明了每一个库函数的使用方法。

进入Libraries 文件夹看到， 关于内核与外设的库文件分别存放在CMSIS 和STM32F10x\_StdPeriph\_Driver 文件夹中。

**1. CMSIS 文件夹。**



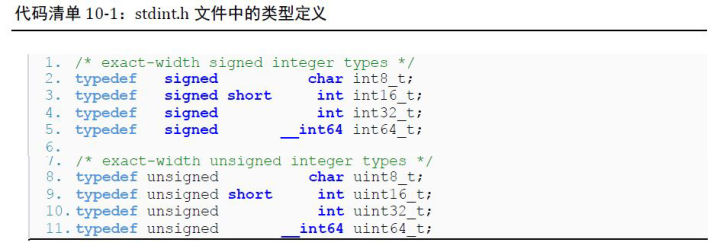
其中黄色框框住的是我们需要用到的内容，下面我们一一讲解下这几个文件的作用。

**内核相关文件**

在CoreSupport 文件夹中有core\_cm3.c 和core\_cm3.h 两个文件。Core\_cm3.h 头文件里

面实现了内核的寄存器映射，对应外设头文件stm32f10x.h，区别就是一个针对内核的外设，一个针对片上（内核之外）的外设。core\_cm3.c 文件实现了一下操作内核外设寄存器的函数，用的比较少。

我们还需要了解的是core\_cm3.h 头文件中包含了“stdint.h” 这个头文件，这是一个ANSI C 文件，是独立于处理器之外的，就像我们熟知的C 语言头文件 “stdio.h” 文件一样。位于RVMDK 这个软件的安装目录下，主要作用是提供一些类型定义。



这些新类型定义屏蔽了在不同芯片平台时，出现的诸如int 的大小是16 位，还是32 位的差异。所以在我们以后的程序中，都将使用新类型如uint8\_t 、uint16\_t 等。在稍旧版的程序中还经常会出现如u8、u16、u32 这样的类型，分别表示的无符号的8位、16 位、32 位整型。初学者碰到这样的旧类型感觉一头雾水，它们定义的位置在STM32f10x.h 文件中。建议在以后的新程序中尽量使用uint8\_t 、uint16\_t 类型的定义。

**启动文件**

启动文件放在startup/arm 这个文件夹下面，这里面启动文件有很多个，不同型号的单片机用的启动文件不一样，有关每个启动文件的详细说明见表



我们开发板中用的STM32F103VET6 或者STM32F103ZET6 的FLASH 都是512K，属于基本型的大容量产品，启动文件统一选择startup\_stm32f10x\_hd.s。

**Stm32f10x.h**

这个头文件实现了片上外设的所以寄存器的映射，是一个非常重要的头文件，在内核中与之想对应的头文件是core\_cm3.h。

**system\_stm32f10x.c**

system\_stm32f10x.c 文件实现了STM32 的时钟配置，操作的是片上的RCC 这个外设。系统在上电之后，首选会执行由汇编编写的启动文件，启动文件中的复位函数中调用的SystemInit 函数就在这个文件里面定义。**调用完之后，系统的时钟就被初始化成72M**。如果后面我们需要重新配置系统时钟，我们就可以参考这个函数重写。为了维持库的完整性，我们不会直接在这个文件里面修改时钟配置函数。

**2. STM32F10x\_StdPeriph\_Driver 文件夹**

文件目录：Libraries\STM32F10x\_StdPeriph\_Driver

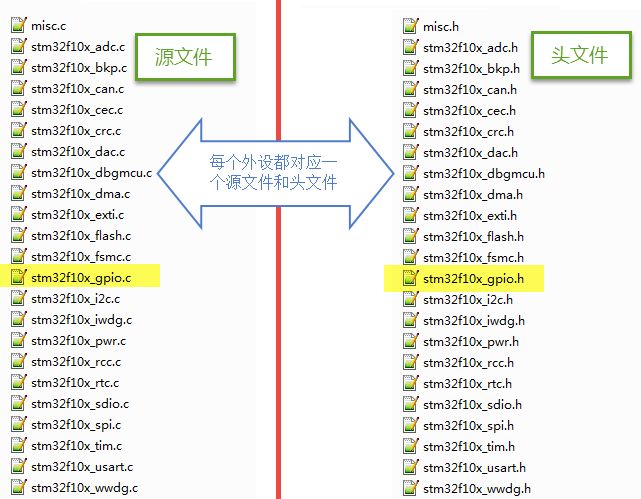
进入libraries 目录下的STM32F10x\_StdPeriph\_Driver 文件夹。



STM32F10x\_StdPeriph\_Driver 文件夹下有inc（include 的缩写）跟src（source 的简写）这两个文件夹，这里的文件属于CMSIS 之外的的、芯片片上外设部分。src 里面是每个设备外设的驱动源程序，inc 则是相对应的外设头文件。src 及inc 文件夹是ST 标准库的主要内容，甚至不少人直接认为ST 标准库就是指这些文件，可见其重要性。

在src 和inc 文件夹里的就是ST 公司针对每个STM32 外设而编写的库函数文件，每个外设对应一个 .c 和 .h 后缀的文件。我们把这类外设文件统称为： stm32f10x\_ppp.c 或stm32f10x\_ppp.h 文件，PPP 表示外设名称。如在上一章中我们自建的stm32f10x\_gpio.c 及stm32f10x\_gpio.h 文件，就属于这一类。

如针对模数转换(ADC)外设，在src 文件夹下有一个stm32f10x\_adc.c 源文件，在inc 文件夹下有一个stm32f10x\_adc.h 头文件，若我们开发的工程中用到了STM32 内部的ADC，则至少要把这两个文件包含到工程里。



这两个文件夹中，还有一个很特别的misc.c 文件，这个文件提供了**外设对内核中的NVIC(中断向量控制器)的访问函数**，在配置中断时，我们必须把这个文件添加到工程中。

**3. stm32f10x\_it.c、 stm32f10x\_conf.h 和system\_stm32f10x.c 文件**

文件目录：STM32F10x\_StdPeriph\_Lib\_V3.5.0\Project\STM32F10x\_StdPeriph\_Template在这个文件目录下，存放了官方的一个库工程模板，我们在用库建立一个完整的工程时， 还需要添加这个目录下的stm32f10x\_it.c 、stm32f10x\_it.h 、stm32f10x\_conf.h 和system\_stm32f10x.c 这四个文件。

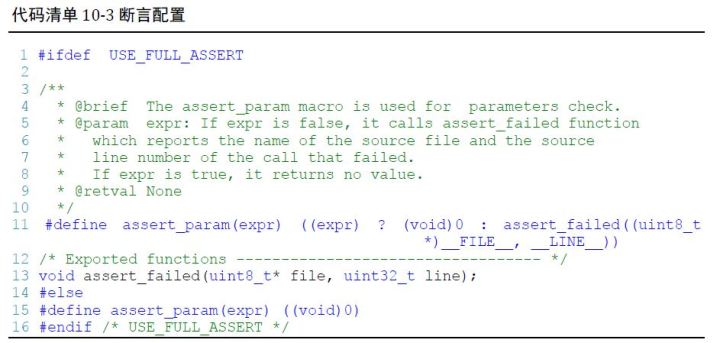
**stm32f10x\_it.c：这个文件是专门用来编写中断服务函数的**，在我们修改前，这个文件已经定义了一些系统异常(特殊中断)的接口，其它普通中断服务函数由我们自己添加。但是我们怎么知道这些中断服务函数的接口如何写？是不是可以自定义呢？答案当然不是，这些都可以在汇编启动文件中找到，在学习中断和启动文件的时候我们会详细介绍

**system\_stm32f10x.c：这个文件包含了STM32 芯片上电后初始化系统时钟、扩展外部存储器用的函数**，例如我们前两章提到供启动文件调用的“SystemInit”函数，用于上电后初始化时钟，该函数的定义就存储在system\_stm32f10x.c 文件。STM32F103 系列的芯片，调用库的这个SystemInit 函数后，系统时钟被初始化为72MHz，如有需要可以修改这个文件的内容，设置成自己所需的时钟频率，但鉴于保持库的完整性，我们在做系统时钟配置的时候会另外重写时钟配置函数。

**stm32f10x\_conf.h：这个文件被包含进stm32f10x.h 文件**。当我们使用固件库编程的时候，如果需要某个外设的驱动库，就需要包含该外设的头文件：stm32f10x\_ppp.h，包含一个还好，如果是用了多外设，就需要包含多个头文件，这不仅影响代码美观也不好管理，现我们用一个头文件stm32f10x\_conf.h 把这些外设的头文件都包含在里面，让这个配置头文件统一管理这些外设的头文件，我们在应用程序中只需要包含这个配置头文件即可，我们又知道这个头文件在stm32f10x.h 的最后被包含，所以最终我们只需要包含stm32f10x.h这个头文件即可，非常方便。Stm32f10x\_conf.h 见代码清单10-2。默认情况下是所以头文件都被包含，没有被注释掉。我们也可以把不要的都注释掉，只留下需要使用的即可。



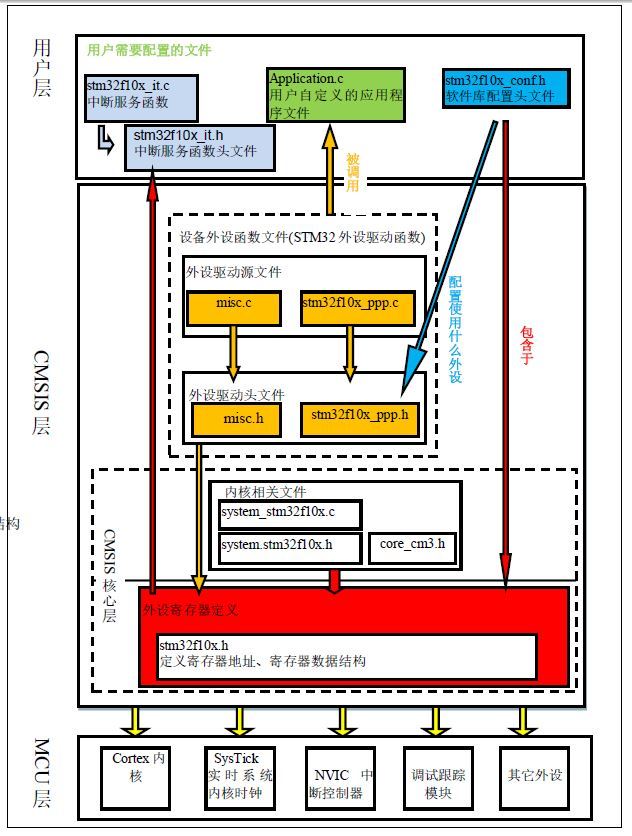
stm32f10x\_conf.h 这个文件还可配置是否使用“断言”编译选项，见代码清单 10-3。



在ST 标准库的函数中，一般会包含输入参数检查，即上述代码中的“assert\_param”宏，当参数不符合要求时，会调用“assert\_failed”函数，这个函数默认是空的。实际开发中使用断言时，先通过定义USE\_FULL\_ASSERT 宏来使能断言，然后定义“assert\_failed”函数，通常我们会让它调用printf 函数输出错误说明。使能断言后，程序运行时会检查函数的输入参数，当软件经过测试，可发布时，会取消USE\_FULL\_ASSERT宏来去掉断言功能，使程序全速运行。

**10.1.2 库各文件间的关系**

前面向大家简单介绍了各个库文件的作用，库文件是直接包含进工程即可，丝毫不用修改，而有的文件就要我们在使用的时候根据具体的需要进行配置。接下来从整体上把握一下各个文件在库工程中的层次或关系，这些文件对应到CMSIS 标准架构上。



上图描述了STM32 库各文件之间的调用关系，在实际的使用库开发工程的过程中，我们把位于CMSIS 层的文件包含进工程，除了特殊系统时钟需要修改system\_stm32f10x.c，其它文件丝毫不用修改，也不建议修改。

对于位于用户层的几个文件，就是我们在使用库的时候，针对不同的应用对库文件进行增删（用条件编译的方法增删）和改动的文件。