基于 STM32CubeMX 开发 U 盘访问应用

前言

一些应用中，涉及到对存储设备的数据访问，例如 uSD 卡、U 盘。具备 USB OTG 控制器的 STM32，可以实现对 U 盘访问的支持。本文介绍 STM32 对于 U 盘访问的硬件/软件实现。介绍如何利用 STM32CubeMX，一步一步实现 STM32 访问 U 盘。仅需要简单的几个步骤， 实现 U 盘访问的应用开发。

一 MSC 类简介

MSC (Mass Storage Class) 是 USB 规范提供的一种 USB 大容量存储设备类，允许一个 USB 接口的设备与 USB 主机相连接，以便在两者之间传输文件。USB MSC 传输协议分为 CBI (Control / Bulk / Interrupt) 和 BOT (Bulk-only Transfer)。BOT 协议在不影响功能的情况下省去了对 Interrupt 端点的需求，被存储设备广泛支持。STM32 提供的 USB 库支持 USB MSC BOT 协议。

MSC 设备包含很多种，例如 U 盘、读卡器、移动硬盘等。STM32 通过内含的 USB OTG 控制器（支持主机模式）和 USB 主机库，实现对 MSC 类的支持，进而实现对 U 盘访问的支持。本文中不涉及到 USB MSC (Mass Storage Class) 的详细介绍，更多 MSC 类介绍，请参考《USB 进阶培训_Part1_USB 类的介绍》。

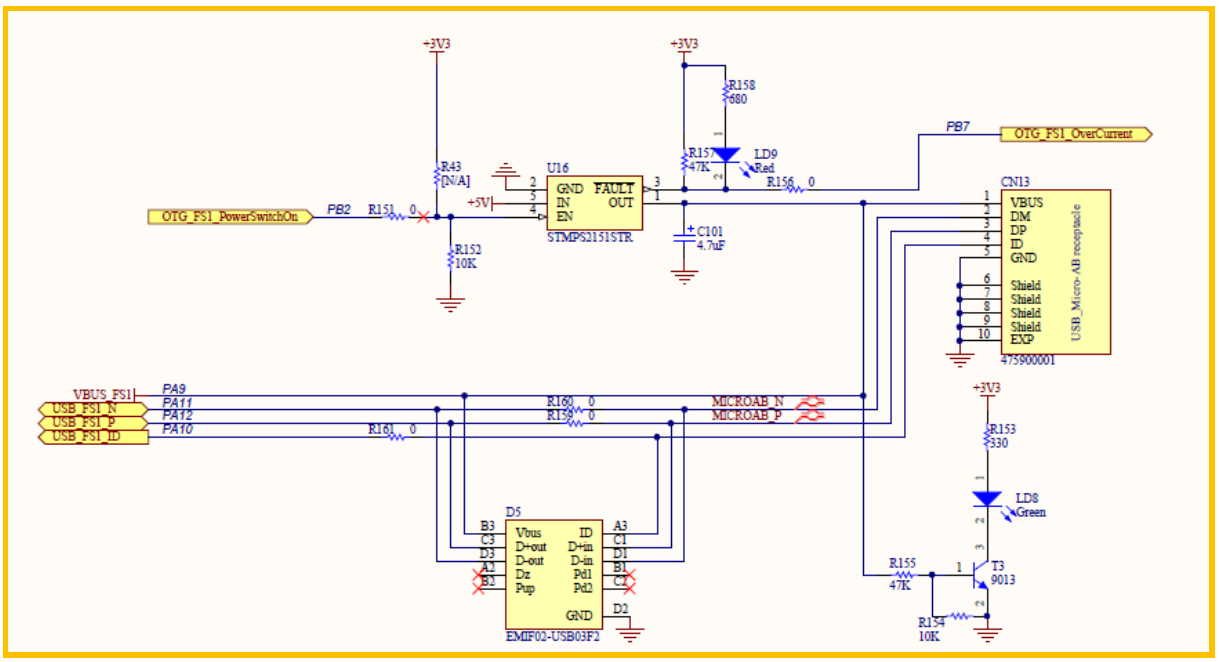
二 MSC 在 STM32 上的实现

2.1 硬件支持

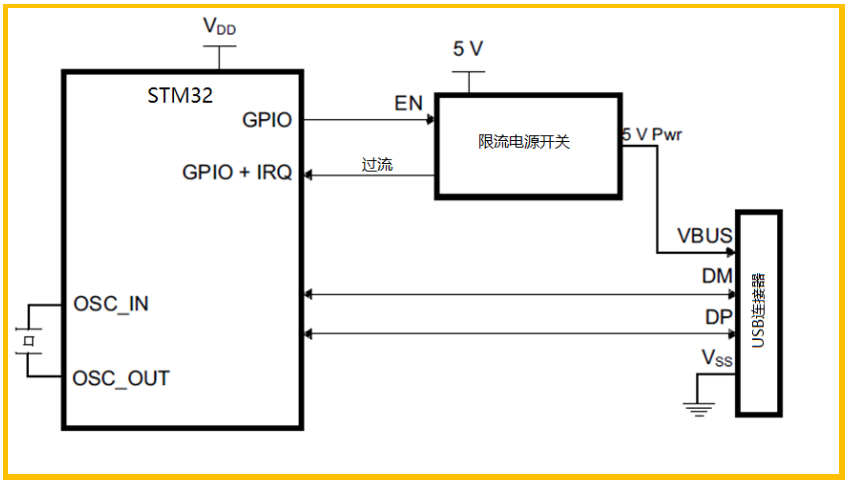
不同的 STM32 系列，对 USB 的支持情况如下。对于 STM32F105/107、STM32F2、STM32F4、STM32F7 和 STM32H7 系列，部分型号支持 USB OTG，能够实现 USB 主机和从机。U 盘作为 USB MSC 设备，需要 STM32 作为 USB MSC 主机，实现两者间的访问。（注：系列中不是所有型号都支持 OTG，以具体型号手册为准。）

	USB	OTG	
	FS	FS	HS
STM32F0	Y		
STM32F102/103	Y		
STM32F105/107		Y	
STM32F2/F4/F7/H7		Y	Y
STM32F3	Y		
STM32FL0/L1	Y		
STM32L4		Y	

STM32 OTG 硬件电路图如下所示。CN3 为 USB 连接器。



对于 STM32 OTG，配置为仅主机模式时，不需使用 OTG_ID，可将其引脚用于其他功能。并且在不使用 SRP (Session Request Protocol) 和 HNP (Host Negotiation Protocol) 时，可不连接 VBUS 至 PA9。电路简化如下图。



更多关于 STM32 USB 硬件设计，请参考《AN4879 USB hardware and PCB guidelines using STM32 MCUs》。

2.2 软件支持

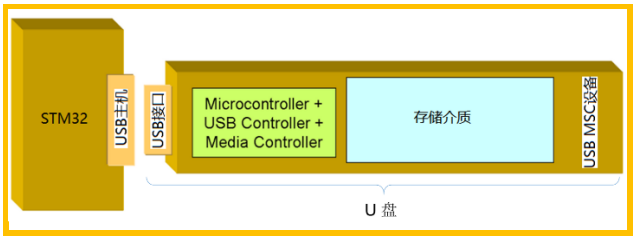
STM32 拥有丰富的软件资源，推荐如下两个软件包：

- STM32Cube 软件包（包含 USB/OTG 库，支持 MSC 协议）
- STM32CubeMX 辅助开发工具（辅助开发自定义板上应用）

上述软件都是免费对外开发，可在意法半导体官网 www.st.com 下载。

三 U 盘访问实现例

一步一步呈现访问 U 盘的 STM32 开发过程，实现对 U 盘的读写等操作。在下面的步骤详解中，会介绍一些主机库和应用机制的内容。如果希望快速开发，可以直接按照步骤开发，略过讲解性的内容。



3.1 前期准备

出于遵循完全一致的实验步骤考虑，实例基于 ST 发布的 STM32F469I-DISCO 板。除此之外，开发者也可以根据自己的目标板的具体情况，参考后面介绍的实例进行配置。

STM32 板	USB 线	STM32CubeMX	Cube 软件包	IDE
STM32F469I-DISCO **链接**(2)： https://www.st.com/en/evaluation-tools/32f469idiscovery.html	2 * USB 线 Type A ←→ Mini B Type A ←→ Micro B	**链接**： https://www.st.com/en/development-tools/stm32cubemx.html **说明**：本文实现例中采用版本 v4.28.0	STM32CubeF4(1) **说明**：在安装 STM32CubeMX 后，在其“菜单栏\Help\Install New Libraries”中安装 STM32CubeF4. 本文实现例中采用的是 V1.21.0	IAR (EWARM) **链接**： https://www.iar.com/ **说明 1**：例中以 IAR 为例。除 IAR 外，CubeMX 还支持 MDK、TrueStudio 和 SW4STM32 等。 文本实现例中采用 IAR v8.30.1

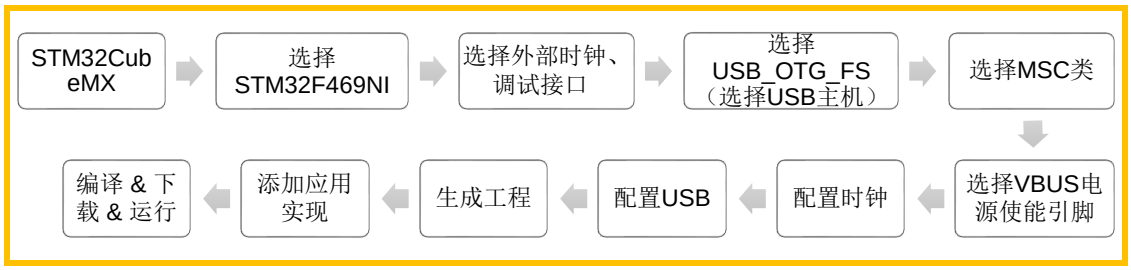
注 1：STM32F469I-DISCO 板的原理图、用户手册等资源可通过链接获取。

注 2：STM32CubeF4 可以按照介绍方式获取，或者直接在官网下载，然后通过 STM32CubeMX 的本地导入软件包功能完成导入。

3.2 应用实现

3. 2. 1 开发流程

结合 STM32CubeMX 的软件开发流程如下图。



3. 2. 2 开发步骤详解

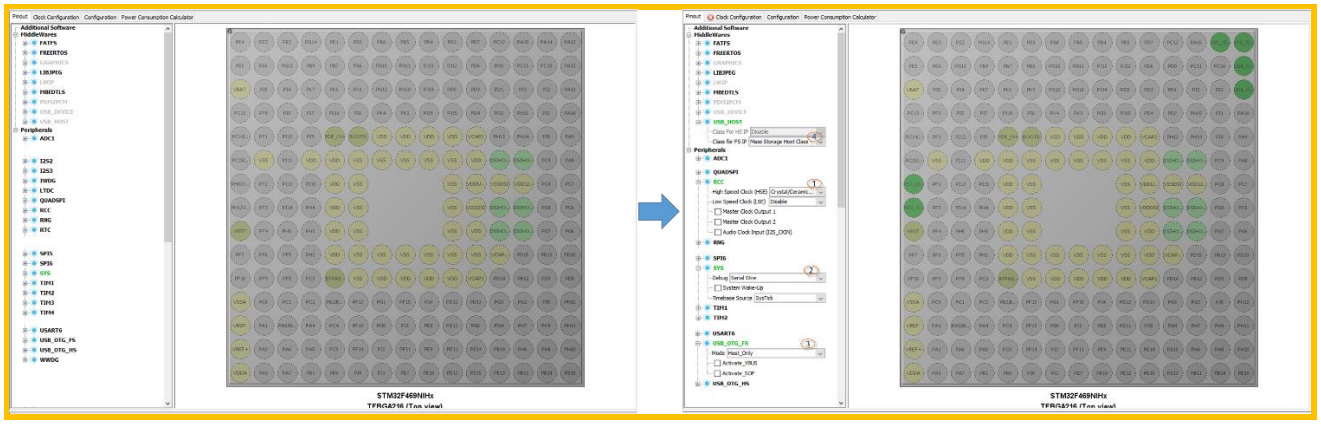
接下来一步一步呈现实现过程。

1. 打开 STM32CubeMX，点击 File\New Project，选择 STM32F469NI（STM32F469I-DISCO 上微控制器型号）。
2. 外设和中间件的选择。首先结合 STM32F469I-DISCO 板的原理图，了解功能实现必备的接口的使用情况，如下表所示。

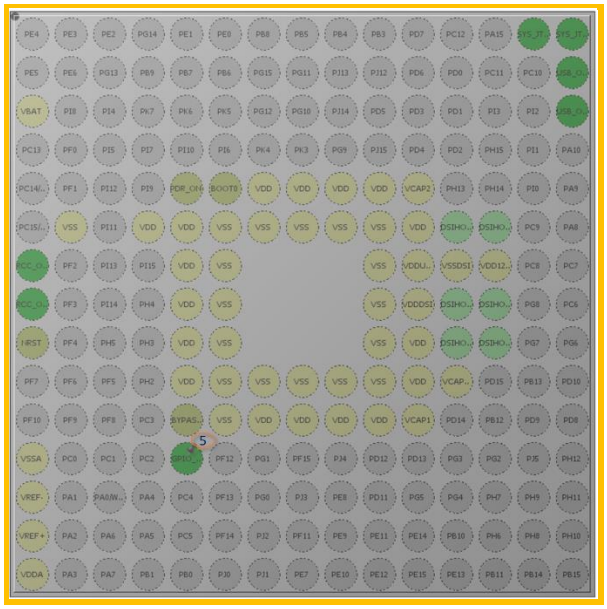
功能	接口	IO	说明
时钟源 （8MHz）	HSE	PH0	OSC_IN
		PH1	OSC_OUT
调试接口	SWD	PA13	SWDIO 数据线
		PA14	SWDCLK 时钟线
VBUS 使能	GPIO	PB2	输出引脚，控制 VBUS 使能
USB OTG FS 接口(1)	OTG	PA11	USB 数据负线
		PA12	USB 数据正线

注1. 在不考虑 SRP 和 HNP 时，访问 U 盘，只需要 STM32 的 USB OTG 的 USB 数据正负线即可。

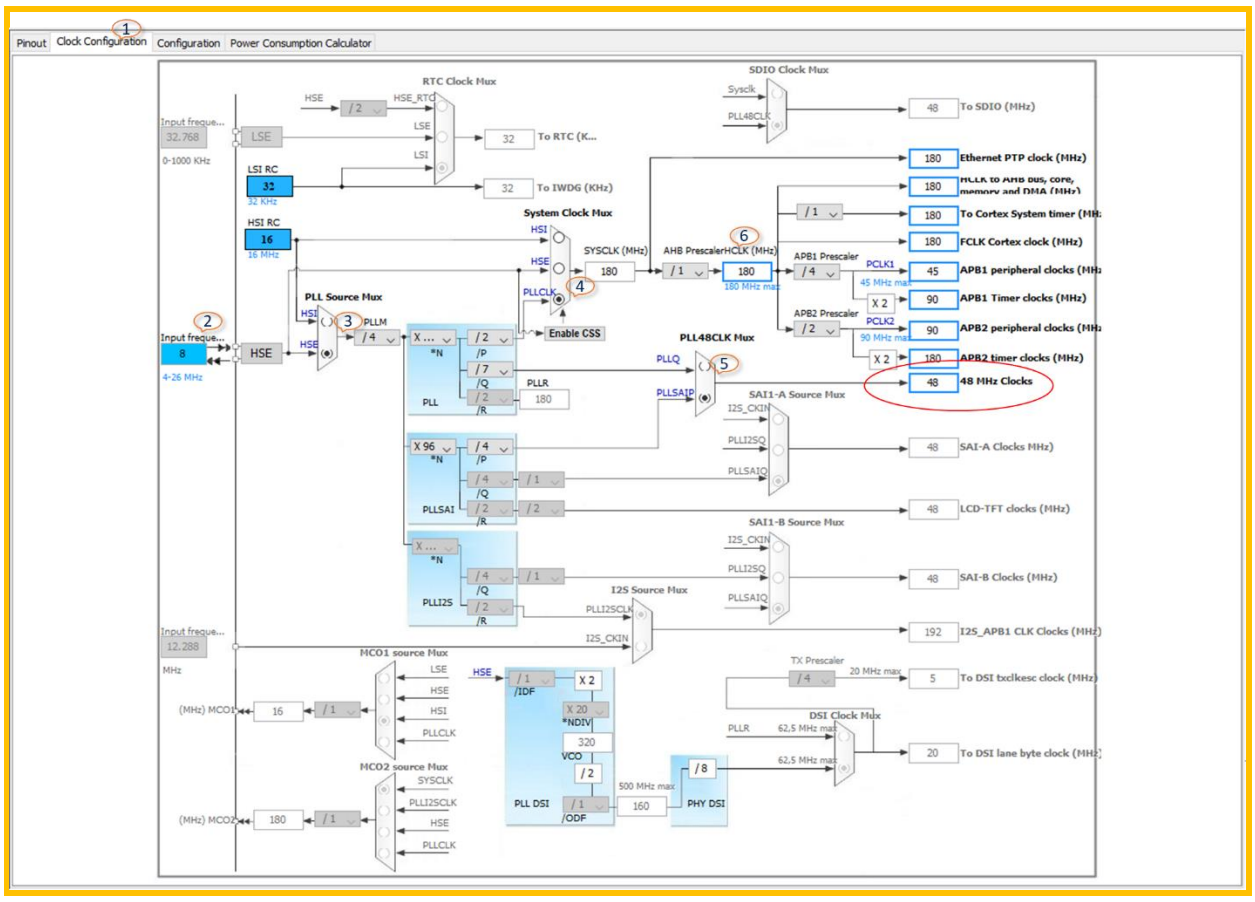
根据硬件情况，选择调试接口、外部时钟、USB OTG FS 和 USB 中间件，如下图所示。



在引脚分布图，PB2 引脚上单击鼠标左键，选择 ‘GPIO_Output’，如下图所示。

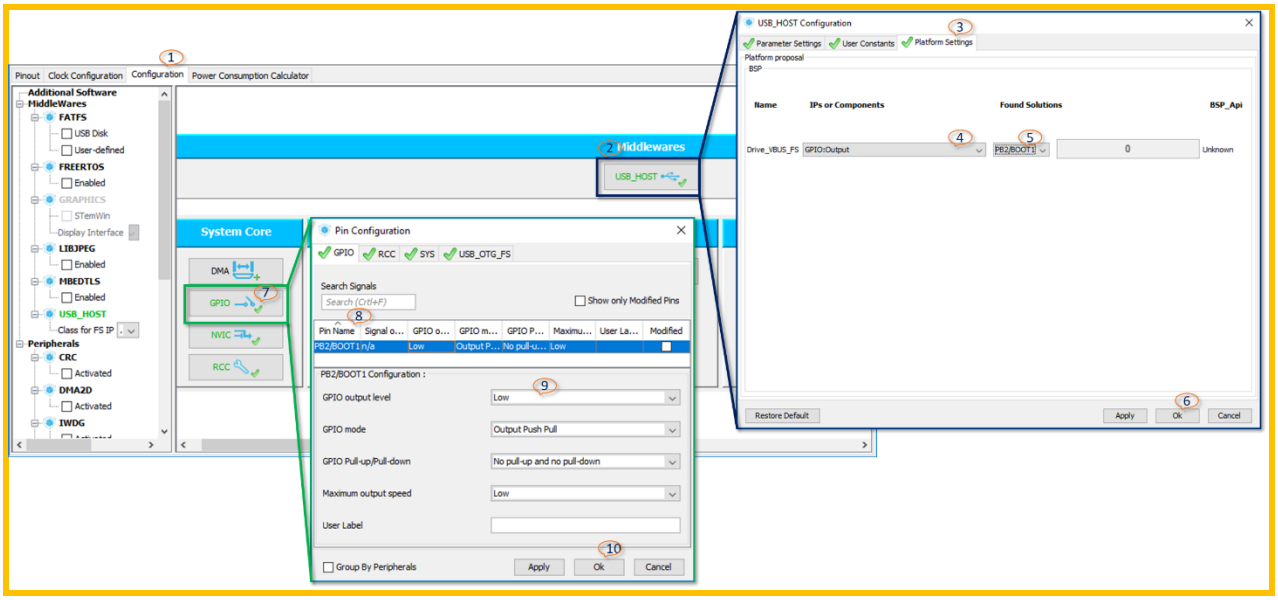


3. 配置时钟，使 USB 时钟为 48MHz，如下图。其中，步骤 2 中外部高速晶振值要和硬件上匹配。由于 USB 时钟精度要求高，STM32F469 内部时钟无法达到要求，必须选用外部高速时钟，如步骤 3 所示。对于 HCLK（处理器时钟，步骤 6 配置），根据性能需要进行设置，或者简单的设置为最大值。

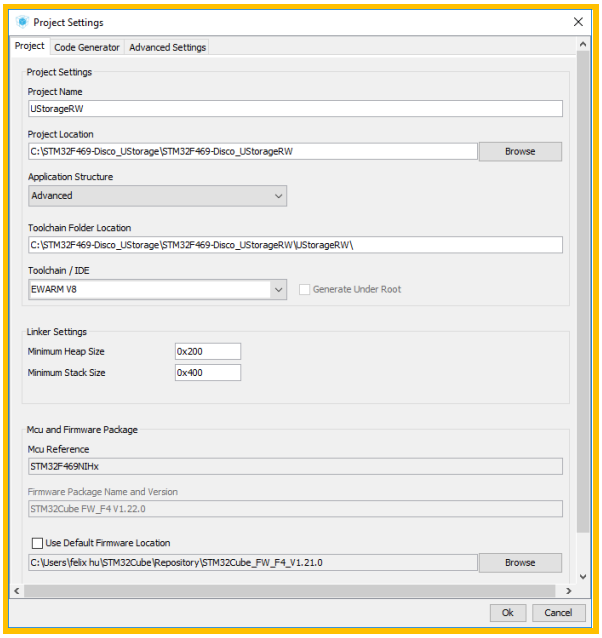


4. USB 和关联的引脚配置。主要对 VBUS 使能引脚进行配置及关联。

   在 USB_HOST 界面，配置 Drive_VBUS_FS 关联引脚为 PB2（与硬件连接对应）。VBUS 电源开关器件为 STMPS2151STR，高电平使能。所以配置如下，在初始化后为低电平（图中步骤 7~10）。其他保持默认。

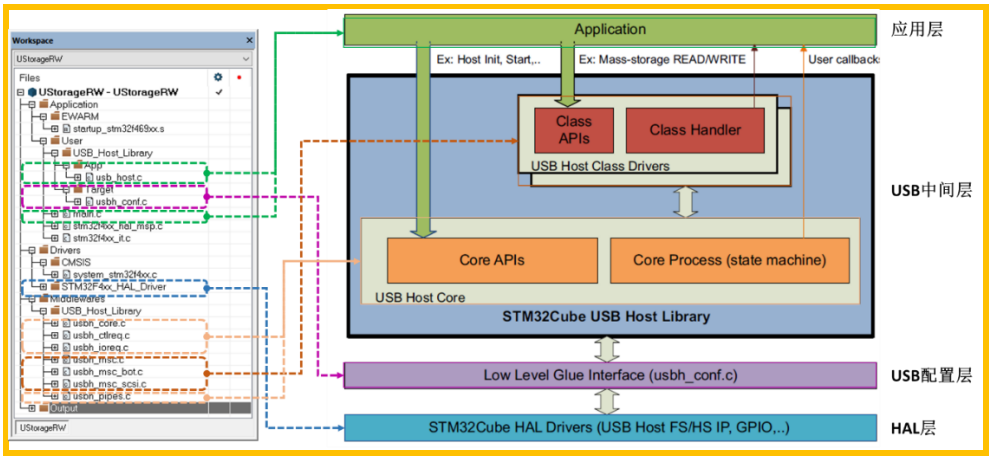


5. 设置工程，如下图所示（点击菜单栏\Project\Settings 打开）。选择对应的 IDE 和希望基于的 STM32Cube 软件包位置。实例简单，堆栈占用小，堆栈配置保持默认即可。除 EWARM 外，STM32CubeMX 还支持 MDK-ARM、TrueStudio、SW4STM32 等。



6. 点击菜单栏\Project\Generate Code 生成工程。

工程生成后，会出现提示框，点击 ‘Open Project’打开工程。工程架构和文件结构如下图（右侧为 USB 主机应用架构）。生成工程包含全部层的实现，开发者在生成工程的基础上，可直接调用写读等 API，实现对 U 盘的访问。



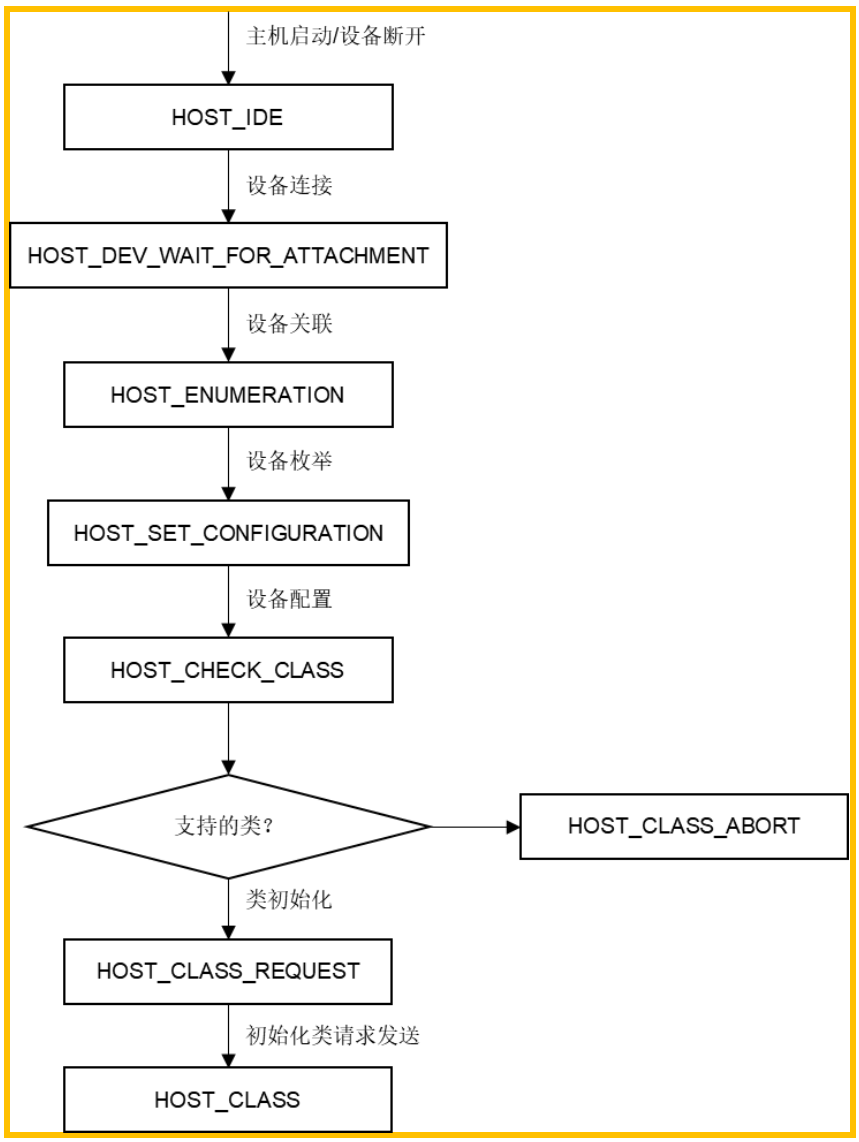
注： USB 主机库和各文件功能介绍，请参考《UM1720 STM32Cube USB host library》。

7. 添加 U 盘访问实现。

STM32CubeMX 生成的工程，调用 STM32Cube USB 主机库。在库中，软核将 USB 事件传输至用户层，并执行回调用户函数。方便在用户层的回调函数中添加应用实现。用户回调事件如下表。

USB 软核 用户回调事件	描述
HOST_USER_CONNECT	通知应用有设备连接
HOST_USER_DISCONNECT	通知应用设备断开
HOST_USER_CLASS_ACTIVE	通知应用类初始化过程完成
HOST_USER_SET_CONFIGURATION	通知应用设备标准枚举完成
HOST_USER_CLASS_SELECTED	通知发现超过两个设备类

USB 软核状态机如下图所示。



在 HOST_CLASS 状态中，软核会通知应用类初始化完成。对 U 盘的访问，应放置在应用层接收到类初始化完成事件后，即 HOST_USER_CLASS_ACTIVE 事件后，否则无法正常工作。

在工程中添加/修改对 U 盘的写读访问操作和相关源码，如下表所示。

文件名	函数名	操作	源码
usb_host.c		新增（定义状态变量和读写缓存）	… (1) /* USER CODE BEGIN PV */ /* Private variables ---------- ----------*/ HAL_StatusTypeDef status; MSC_LUNTypeDef uStorageInf; uint8_t buffer_tx[0x100] ; uint8_t buffer_rx[0x100] ; /* USER CODE END PV */ … (1)

	USBH_UserProcess	新增（U 盘信息读取、读写 API 调用）	… (1)
			case HOST_USER_CLASS_ACTIVE:
			  Appli_state = APPLICATION_READY;
			  /* Get USB storage information */
			  USBH_MSC_GetLUNInfo(&hUsbHostFS,0,&uStorageInf);
			
			  for(uint32_t i = 0; i< sizeof(buffer_tx); i++)
			  buffer_tx[i] = i;
			  /* Write one sector data from buffer_tx to tenth sector */
			  status = USBH_MSC_Write(&hUsbHostFS, 0, 10, (uint8_t *)buffer_tx, 1) ;
			  if(status)
			  {
			  while(1); //Program USB storage card unsuccessfully!
			  }
			  /* Read one sector data from buffer_tx to tenth sector */
			  status = USBH_MSC_Read(&hUsbHostFS, 0, 10, (uint8_t *)buffer_rx, 1) ;
			  if(status)
			  {
			  while(1); //Read USB storage card unsuccessfully!
			  }
			… (1)

usbh_conf.c		新增(2)（VBUS 电源开关驱动 API）	… (1)
			/* USER CODE BEGIN 1 */
			void MX_DriverVbusFS(uint8_t state);
			/**
			  * @brief Drive VBUS.
			  * @param state : VBUS state
			  * This parameter can be one of the these values:
			  * 1 : VBUS Active
			  * 0 : VBUS Inactive
			  */
			void MX_DriverVbusFS(uint8_t state)
			{
			  if(state == 0)
			  {
			  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_2, GPIO_PIN_RESET);
			  }
			  else
			  {
			  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_2, GPIO_PIN_SET);
			  }
			}
			/* USER CODE END 1 */
			… (1)

	USB H_LL _Init	修改	… (1)
			if (HAL_HCD_Init(&hhcd_USB_OTG_FS) != HAL_OK)
			{
			//_Error_Handler(__FILE__, __LINE__);
			while(1); //Modify because there is no _Error_Handler() definition in the generated project
			}
			… (1)

注1. 省略号为了表示还有其他没有改动部分的源码，不用添加进源码中。展现的源码为了更好的表现出对应工程中位置，包含一些生成工程时原有的源码。

注2. MX_DriverVbusFS()中，需要根据实际情况进行配置。实验板上采用的电源开关 STMPS2151STR 高电平有效。在一些系列 STM32 的生成工程中，MX_DriverVbusFS 函数原型已经生成，此时无需再添加这个函数的声明，但要确认函数中引脚设置对应关系（形参为 1 时表示要激活）。

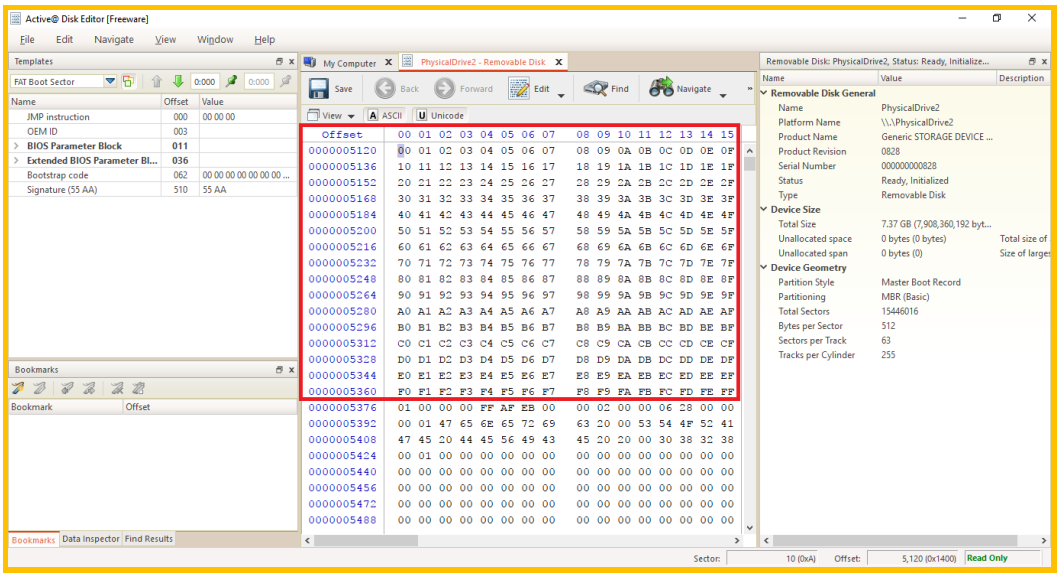
8. 编译生成的应用固件。

利用 IDE 进行编译、链接、下载到目标板，实现 U 盘信息获取和读写操作。

四 测试&验证

连接 U 盘至 STM32F469I-DISCO 板的 USB_User 连接口。IAR 进入在线调试模式，利用在线调试，查看获取到的 U 盘信息和写读缓存中的数据情况。

运行完毕后，连接 U 盘至 PC，利用 PC 上安装的 Active@ Disk Editor（外部链接）查看 U 盘对应扇区数据（10 扇区对应起始位置为 5120），从而验证 U 盘信息获取和写读功能正常。如下图所示。



五 小结

STM32CubeMX 加速了 STM32 的开发过程。即使类似 USB 这种复杂的外设使用，也可以如上述实现例，只需要几个步骤即可以实现 USB 应用。并且提供了除 MSC 类之外，HID、UAC、CDC 等类支持。

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。