认识HAL库

1，初识HAL库（了解）

1.1，CMSIS简介

CMSIS (微控制器软件接口标准)：Cortex Microcontroller Software Interface Standard，是由ARM和与其合作的芯片厂商、软件工具厂商，共同制定的标准

1.2，HAL库简介

直接操作寄存器
- 执行效率高
- 时间成本高

标准库
- F0/F1/F3/F2/F4/L1
- 目前已停止维护

HAL库
- 全系列兼容
- ST目前主推的库
- 兼容性、易移植性
- 效率低

LL库
- 全系列兼容
- 与HAL库捆绑发布
- 轻量级、效率高
- 不匹配部分复杂外设

2，STM32Cube固件包浅析（了解）

2.1，如何获取STM32Cube固件包？

方式一：ST官网搜索STM32Cube

方式二：开发板A盘资料：A 盘→8，STM32 参考资料→1，STM32CubeXX固件包

2.2，STM32Cube固件包文件夹简介

- Drivers
  - BSP
  - CMSIS
    - Device
      - stm32f1xx.h
      - stm32f103xe.h
      - system_stm32f1xx.h
      - system_stm32f1xx.c
      - startup_stm32f103xe.s
    - Include
      - cmsis_armcc.h
      - cmsis_armclang.h
      - cmsis_compiler.h
      - cmsis_version.h
      - core_cm3.h
      - mpu_armv7.h
  - STM32F1xx_HAL_Driver
- Middlewares
  - ST
    - STemWin
    - STM32_USB_Device_Library
    - STM32_USB_Host_Library
  - Third_Party
    - FatFs
    - FreeRTOS
    - LwIP
- Projects

2.3，CMSIS文件夹关键文件

3，HAL库框架结构（了解）

3.1，HAL库文件夹结构

- Inc
- Src

包含HAL库和LL库驱动源码

用户手册

3.2，HAL库文件介绍

文件	描述
sm32f1xx_hal.c stm32f1xx_hal.h	HAL库初始化、系统滴答、HAL库延时等相关函数
stm32f1xx_hal_conf.h	HAL库的用户配置文件，用于裁剪HAL库、配置晶振参数等
stm32hxx_hal_def.h	包含HAL库通用的枚举类型数据和宏定义
stm32f1xx_hal_cortex.h stm32f1xx_hal_cortex.c	内核通用函数定义和声明，如NVIC、MPU、系统软复位、Systick等，其实主要是对core_cm3.h 文件的相关函数再次封装。
stm32f1xx_hal_ppp.c stm32f1xx_hal_ppp.h	某任意外设驱动源码，PPP表示任意外设
stm32f1xx_hal_ppp_ex.c stm32f1xx_hal_ppp_ex.h	主要是存放外设的扩展（特殊）功能的驱动源码，PPP表示任意外设
stm32f1xx_ll_ppp.c stm32f1xx_ll_ppp.h	LL库驱动源码，在部分STM32F1xx_hal_ppp.c 或stm32f1xx_hal_ppp_ex.c 中会被调用

3.3，HAL库API函数和变量命名规则

文件名称	stm32f1xx_hal_ppp .(c/h)	stm32f1xx_hal_ppp_ex .(c/h)
函数名	HAL_PPP_Function	HAL_PPPEx_Function
外设句柄	PPP_HandleTypeDef	无
外设工作参数初始化结构体	PPP_InitTypeDef	PPP_InitTypeDef
配置参数结构体	PPP_YyyyConfTypeDef	PPP_YyyyConfTypeDef

初始化/反初始化函数:HAL_PPP_Init(), HAL_PPP_DeInit()
外设读写函数:HAL_PPP_Read(),HAL_PPP_Write(),HAL_PPP_Transmit(), HAL_PPP_Receive()
控制函数:HAL_PPP_Set (),HAL_PPP_Get ()
状态和错误:HAL_PPP_GetState (), HAL_PPP_GetError ()

HAL库对寄存器位操作的相关宏定义

宏定义结构	用途
__HAL_PPP_ENABLE_IT(__HANDLE__, __INTERRUPT__)	使能外设中断
__HAL_PPP_DISABLE_IT(__HANDLE__, __INTERRUPT__)	禁用外设中断
__HAL_PPP_GET_IT (__HANDLE__, __INTERRUPT__)	获取外设某一中断源
__HAL_PPP_CLEAR_IT (__HANDLE__, __INTERRUPT__)	清除外设中断
__HAL_PPP_GET_FLAG (__HANDLE__, __FLAG__)	获取外设的状态标记
__HAL_PPP_CLEAR_FLAG (__HANDLE__, __FLAG__)	清除外设的状态标记
__HAL_PPP_ENABLE(__HANDLE__)	使能某一外设
__HAL_PPP_DISABLE(__HANDLE__)	禁用某一外设
__HAL_PPP_XXXX (__HANDLE__, __PARAM__)	针对外设的特殊操作
__HAL_PPP_GET_IT_SOURCE (__HANDLE__, __INTERRUPT__)	检查外设的中断源

HAL库回调函数

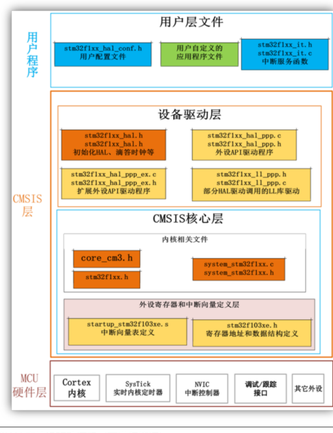
回调函数	举例
HAL_PPP_MspInit() / _MspDeInit ()	举例: HAL_USART_MspInit() 会被HAL_PPP_Init() 函数调用，该函数主要用于存放外设使用到的GPIO、CLOCK、NVIC、DMA等初始化代码
HAL_PPP_ProcessCpltCallback	举例: HAL_USART_TxCpltCallback 由外设中断或DMA中断调用，HAL库中断公共处理函数已经实现对中断标记位读取、判断和清除操作，用户只需要专注于中断逻辑功能的实现即可
HAL_PPP_ErrorCallback	举例: HAL_USART_ErrorCallback 外设或DMA中断中发生的错误，用于发生错误处理

此类函数通常被_weak修饰（弱函数），允许用户重新定义该函数

4，如何使用HAL库（熟悉）

4.1，基于CMSIS应用程序文件描述

类别	文件名	描述	是否必须
用户程序文件	main.c	存放 main 函数，不一定要在这个文件	否
	main.h	包含头文件、声明等作用，已删除	否
	stm32f1xx_it.c	用户中断服务函数存放文件，不一定放到这个文件，可删除	否
	stm32f1xx_it.h		否
	stm32f1xx_hal_conf.h	用户配置文件	是
	stm32f1xx_hal_msp.c	回调函数存放文件，已删除	否
设备驱动层	stm32f1xx_hal.c	HAL 库的初始化，系统滴答，HAL 库延时函数等功能	是
	stm32f1xx_hal.h		是
	stm32hxx_hal_def.h	通用 HAL 库资源定义	是
	stm32f1xx_hal_ppp.c	外设的操作 API 函数文件	是
	stm32f1xx_hal_ppp.h		是
	stm32f1xx_hal_ppp_ex.c	拓展外设特性的 API 函数文件	是
	stm32f1xx_hal_ppp_ex.h		是
	stm32f1xx_ll_ppp.c	LL 库文件，在一些复杂外设中实现底层功能	是
	stm32f1xx_ll_ppp.h		是
CMSIS 核心层	stm32f1xx.h	STM32F1 系列的顶层头文件	是
	stm32f103xe.h	STM32F103系列片上外设头文件	是
	system_stm32f1xx.c	主要存放系统初始化函数 SystemInit	是
	system_stm32f1xx.h		是
	startup_stm32f1xx.s	启动文件，运行到 main 函数前的准备	是
	core_cm3.h	内核寄存器定义，如 Systick、SCB 等	是
	cmsis_armcc.h	编译器相关头文件，一般都不需要去了解，只需加入工程即可	是
	cmsis_armclang.h		
	cmsis_compiler.h		
	cmsis_version.h		
	mpu_armv7.h		



4.2，HAL 库的用户配置文件

1，裁剪HAL库外设驱动源码（不进行编译）
2，设置外部高速晶振频率（根据开发板实际情况设置）
3，设置外部低速晶振频率（根据开发板实际情况设置）

4.3，stm32h7xx_hal.c 文件

```c
HAL_StatusTypeDef HAL_Init(void)
{
    __HAL_FLASH_PREFETCH_BUFFER_ENABLE();       /* 使能FLASH预取缓冲 */

    HAL_NVIC_SetPriorityGrouping(NVIC_PRIORITYGROUP_2); /* 配置中断优先级分组 */

    /* 使用滴答定时器作为时钟基准，配置 1ms 滴答（重置后默认的时钟源为 HSI） */
    HAL_InitTick(TICK_INT_PRIORITY);

    HAL_MspInit();      /* 初始化其它底层硬件（如果必要） */

    return HAL_OK;      /* 返回函数状态 */
}
```

5，HAL库使用注意事项（了解）

1，使用HAL库出现问题，还是得通过参考手册检查是否硬件操作是否有问题
2，尽量不通过修改库源码实现功能，这样不方便库更新
3，HAL库可能会存在错误，要有质疑精神
4，有些HAL库API函数执行效率偏低，我们可能会直接通过操作寄存器的方式代替

6，课堂总结（了解）