第5讲使用STM32固件库操纵GPIO

内容提要

5.1 STM32固件库

5.2 固件库中与GPIO相关的api

5.3 建立工程-STM32库方式

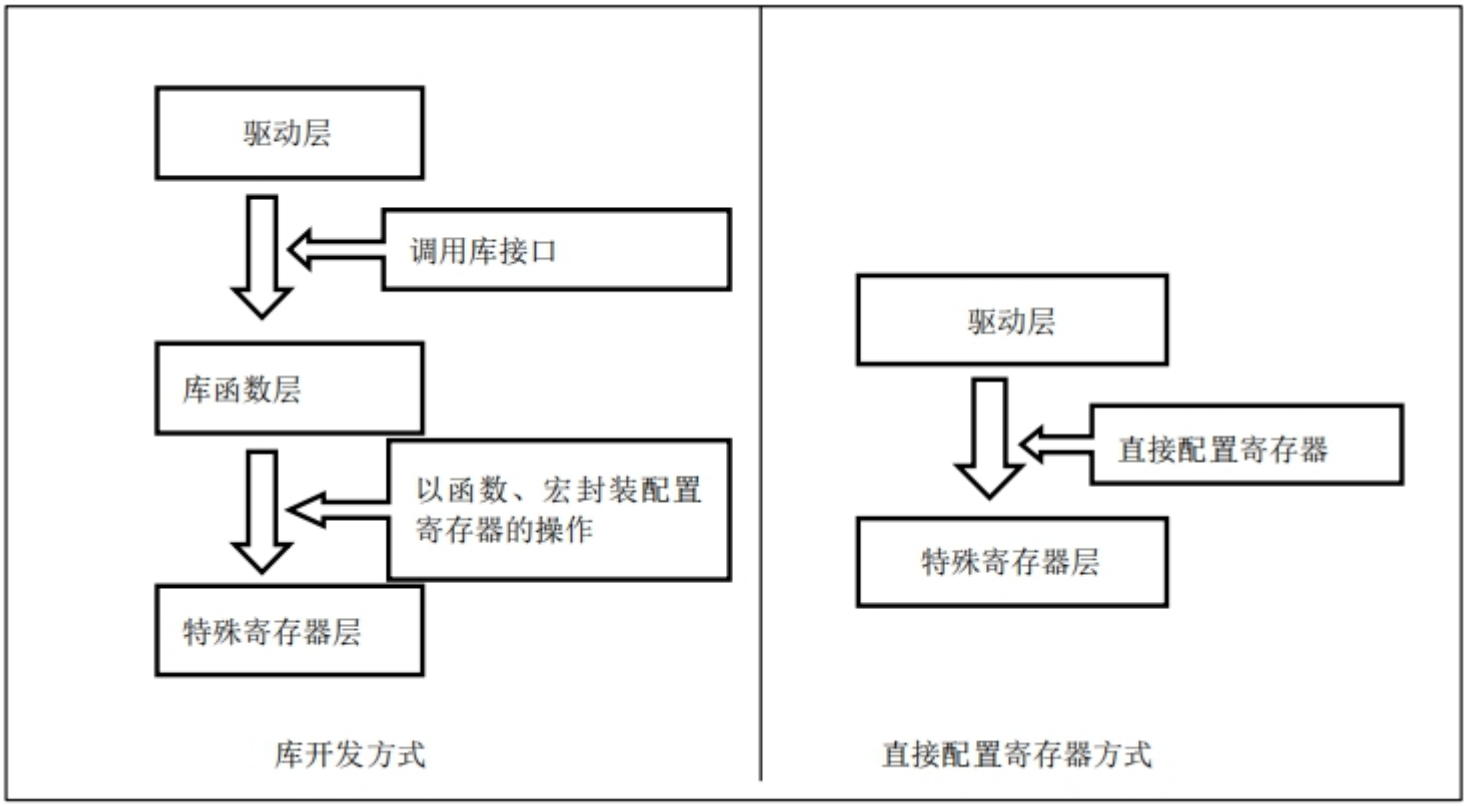
5.4 使用stm32库函数操纵gpio举例

5.1 STM32固件库

什么是STM32固件库（函数库）

- “STM32 标准函数库”它是由 ST 公司针对 STM32 提供的函数接口，即 API (Application Program Interface)，开发者可调用这些函数接口来配置 STM32的寄存器，使开发人员得以脱离最底层的寄存器操作，有开发快速、易于阅读、维护成本低等优点。
- 当我们调用库 API 的时候不需要去了解库底层的寄存器操作，就像开始学习 C 语言的时候，用 printf()函数时只是学习它的使用格式，并没有去研究它的源码实现，但需要深入研究的时候，经过千锤百炼的库源码就是最佳学习范例。
- 实际上，库是架设在寄存器与用户驱动层之间的代码，向下处理与寄存器直接相关的配置，向上为用户提供配置寄存器的接口。

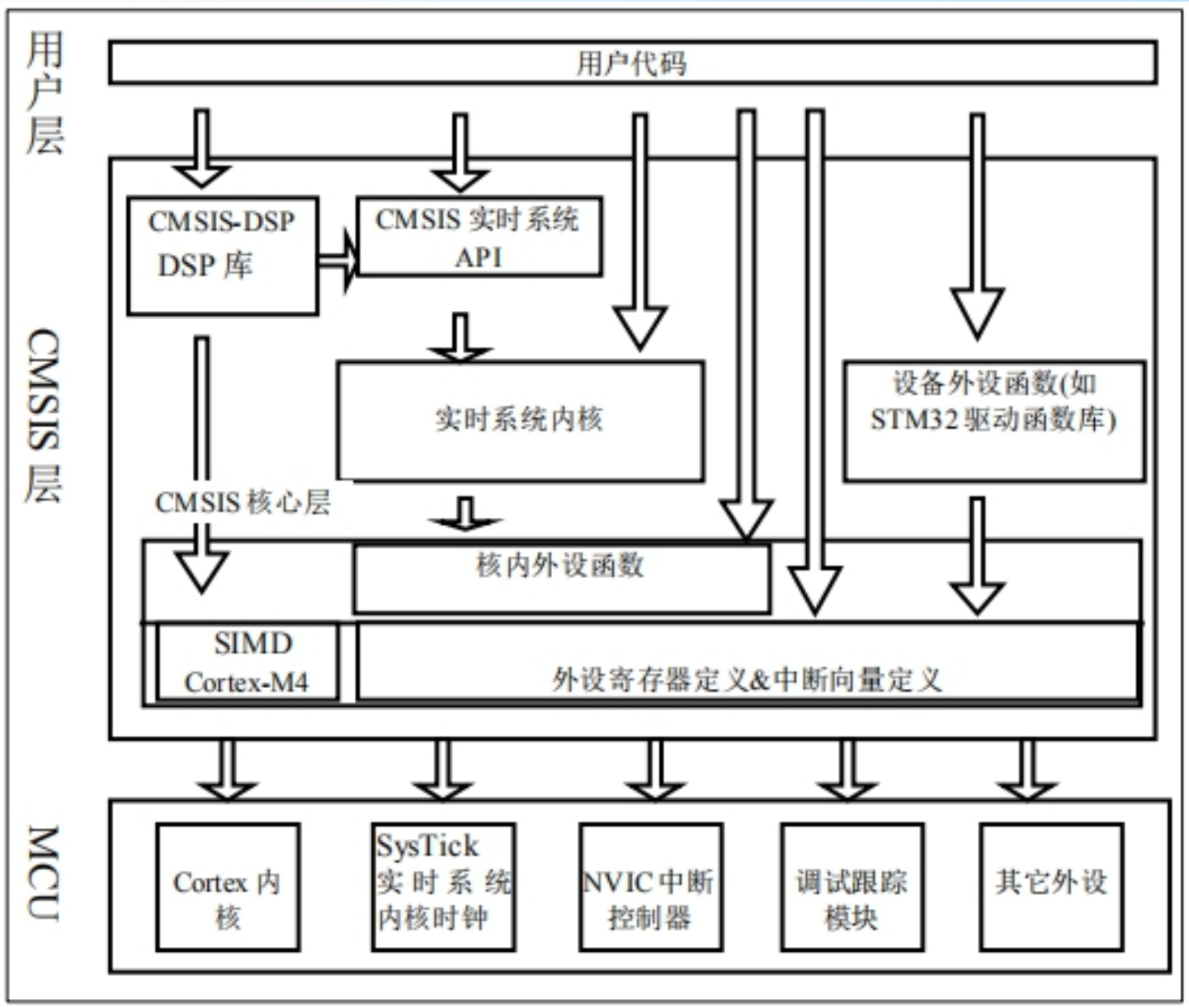
库开发方式与直接配置寄存器方式的区别如图所示



CMSIS 标准及库层次关系

- 基于Cortex核系列芯片采用的内核都是相同的，区别主要表现为核外的片上外设的差异，但这些差异却导致软件在相同内核、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(Cortex MicroController 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。

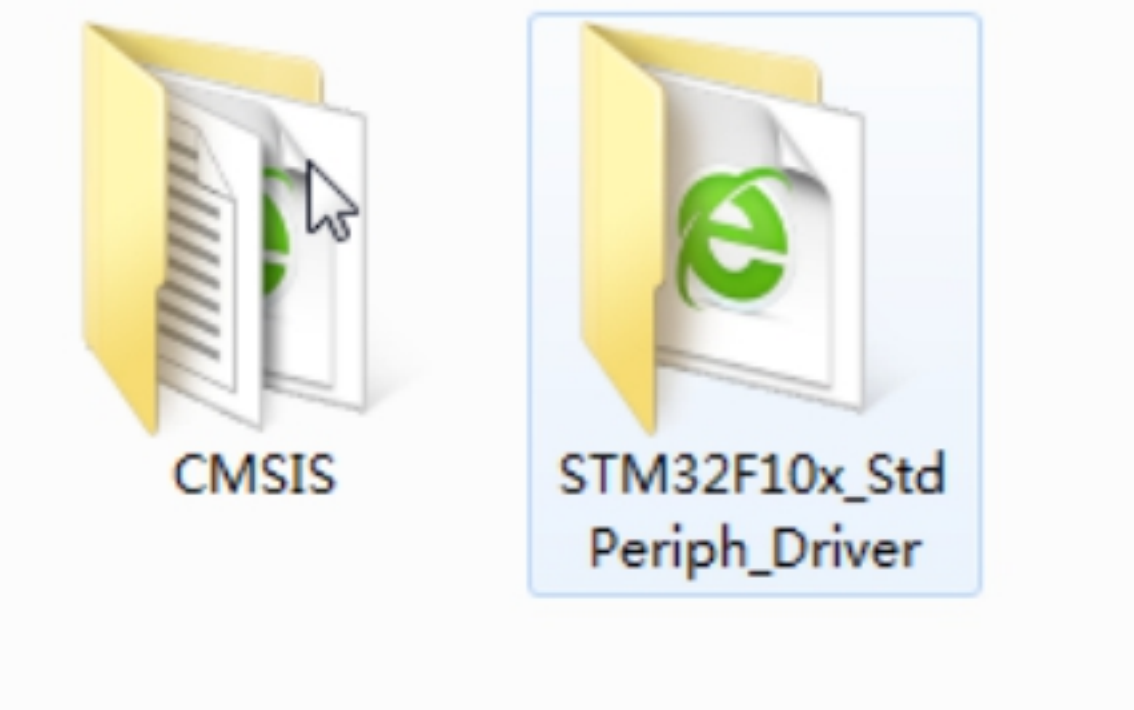
库目录、文件简介

- STM32 标准库可以从官网获得，解压库文件后进入其目录：“STM32F10x_StdPeriph_Lib_V3.5.0\”
- 软件库各文件夹的内容说明见图

- Libraries文件夹：驱动库（固件库）的源代码及启动文件
- Project文件夹：用驱动库写的例子和工程模板
- Utilities：包含了基于 ST 官方实验板的例程（一般用不到）
- stm32f10x_stdperiph_lib_um.chm： 库帮助文档

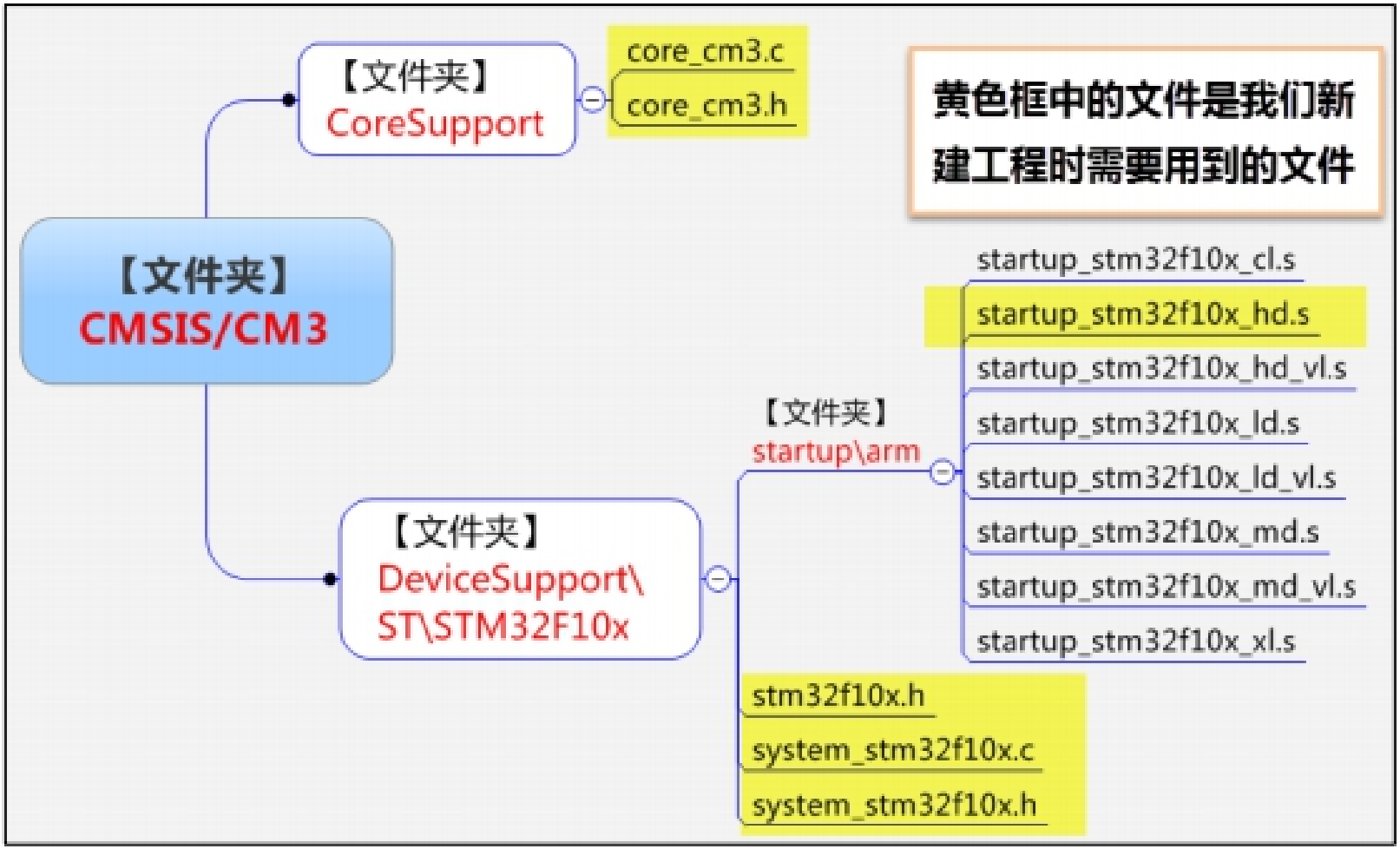
在使用库开发时，我们需要把 libraries 目录下的库函数文件添加到工程中，并查阅库帮助文档来了解每一个库函数的使用方法。

- 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。



CMSIS 文件夹

- STM32F10x_StdPeriph_Lib_V3.5.0\Libraries\CMSIS\文件夹展开内容



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 ：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 ：实现了操作内核外设寄存器的函数
- core_cm3.h中包含了stdint.h，其中提供一些类型定义。

```
/* exact-width signed integer types */
typedef   signed          char int8_t;
typedef   signed short     int int16_t;
typedef   signed           int int32_t;
typedef   signed       __int64 int64_t;
```

启动文件

- 启动文件放在 startup\arm 文件夹下面，针对不同型号的单片机用的启动文件不一样

启动文件	区别
startup_stm32f10x_ld.s	ld: low-density 小容量， FLASH 容量在 16-32K 之间
startup_stm32f10x_md.s	md: medium-density 中容量，FLASH 容量在 64-128K 之间
startup_stm32f10x_hd.s	hd: high-density 中容量， FLASH 容量在 256-512K 之间
startup_stm32f10x_xl.s	xl: 超大容量， FLASH 容量在 512-1024K 之间
以上四种都属于基本型，包括 STM32F101xx、STM32F102xx、STM32F103xx 系列	
startup_stm32f10x_cl.s	cl:connectivity line devices 互联型，特指 STM32F105xx 和 STM32F107xx 系列
startup_stm32f10x_ld_vl.s startup_stm32f10x_md_vl.s startup_stm32f10x_hd_vl.s	vl:value line devices 超值型系列，特指 STM32F100xx 系列

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

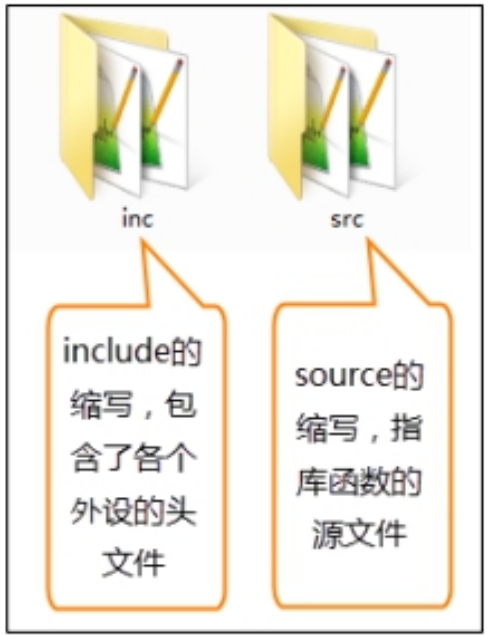
- 这个头文件实现了片上外设的所有寄存器的映射，在内核中与之相对应的头文件是 core_cm3.h。

system_stm32f10x.c

- system_stm32f10x.c 文件实现了 STM32 的时钟配置，操作的是RCC（Reset and Clock Control）这个片上外设。
- 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。

STM32F10x_StdPeriph_Driver 文件夹

- Libraries\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 及 inc 文件夹是 ST 标准库的主要内容,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。



Stm32f10x_ppp.h和stm32f10x_ppp.c

- 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:针对GPIO，在 src 文件夹下有一个 stm32f10x_gpio.c 源文件，在 inc 文件夹下有一个 stm32f10x_gpio.h 头文件。

misc.h和misc.c

- 这两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

stm32f10x_it.c、 stm32f10x_conf.h 和 system_stm32f10x.c

- 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。

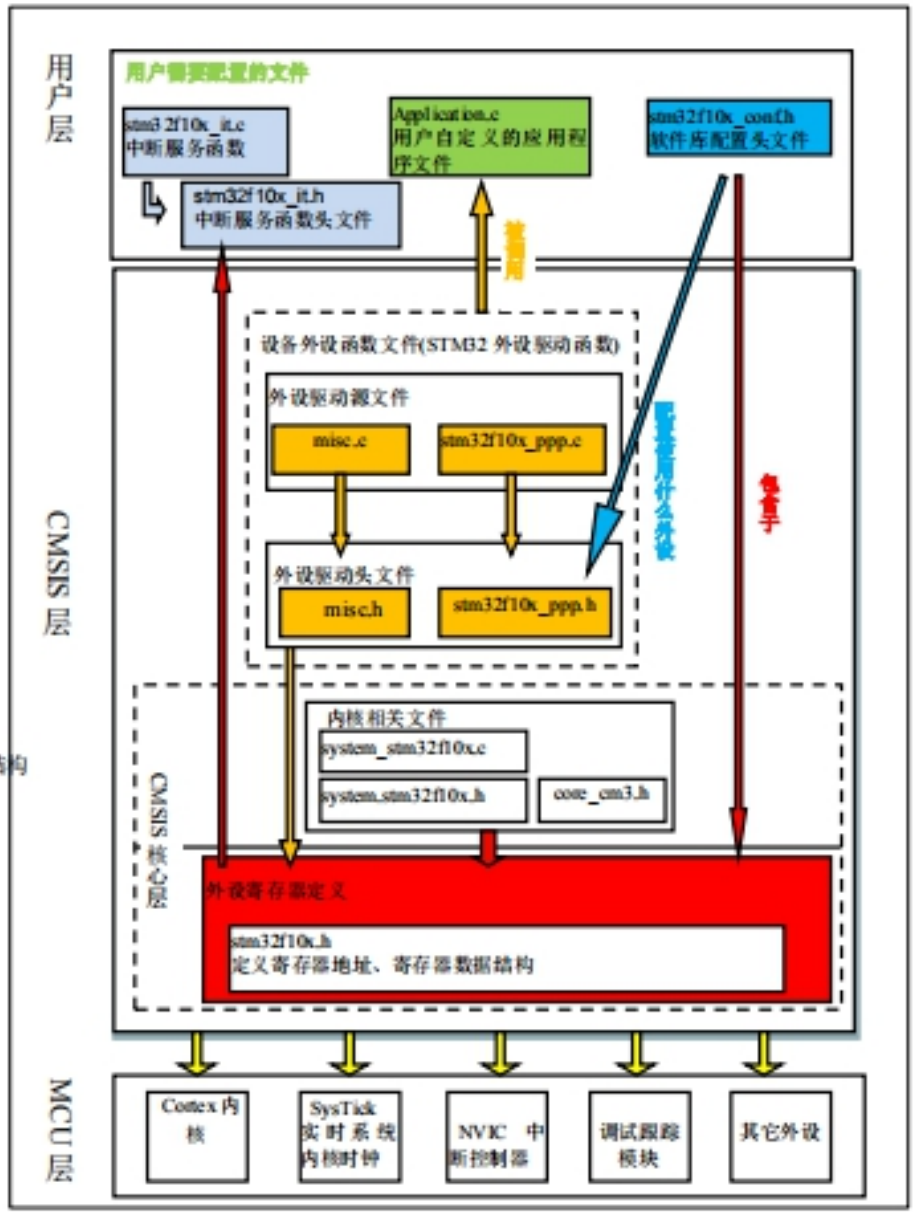
- stm32f10x_it.c：这个文件是专门用来编写中断服务函数的。文件中已经定义了一些系统异常(特殊中断)的接口，其它普通中断服务函数由我们自己添加（其接口参见启动代码）。
- system_stm32f10x.c：这个文件包含了 STM32 芯片上电后初始化系统时钟、扩展外部存储器用的函数，如供启动文件调用的“SystemInit”函数，用于上电后初始化时钟，该函数的定义就存储在 system_stm32f10x.c 文件。STM32F103 系列的芯片，调用库的这个 SystemInit 函数后，系统时钟被初始化为 72MHz，如有需要可以修改这个文件的内容，设置成自己所需的时钟频率，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， 我们可以用一个头文件 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。

```
 1 #include "stm32f10x_adc.h"
 2 #include "stm32f10x_bkp.h"
 3 #include "stm32f10x_can.h"
 4 #include "stm32f10x_cec.h"
 5 #include "stm32f10x_crc.h"
 6 #include "stm32f10x_dac.h"
 7 #include "stm32f10x_dbgmcu.h"
 8 #include "stm32f10x_dma.h"
 9 #include "stm32f10x_exti.h"
10 #include "stm32f10x_flash.h"
11 #include "stm32f10x_fsmc.h"
12 #include "stm32f10x_gpio.h"
13 #include "stm32f10x_i2c.h"
14 #include "stm32f10x_iwdg.h"
15 #include "stm32f10x_pwr.h"
16 #include "stm32f10x_rcc.h"
17 #include "stm32f10x_rtc.h"
18 #include "stm32f10x_sdio.h"
19 #include "stm32f10x_spi.h"
20 #include "stm32f10x_tim.h"
21 #include "stm32f10x_usart.h"
22 #include "stm32f10x_wwdg.h"
23 #include "misc.h"
```

默认情况下是所有头文件都被包含，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。

库中各文件间的关系



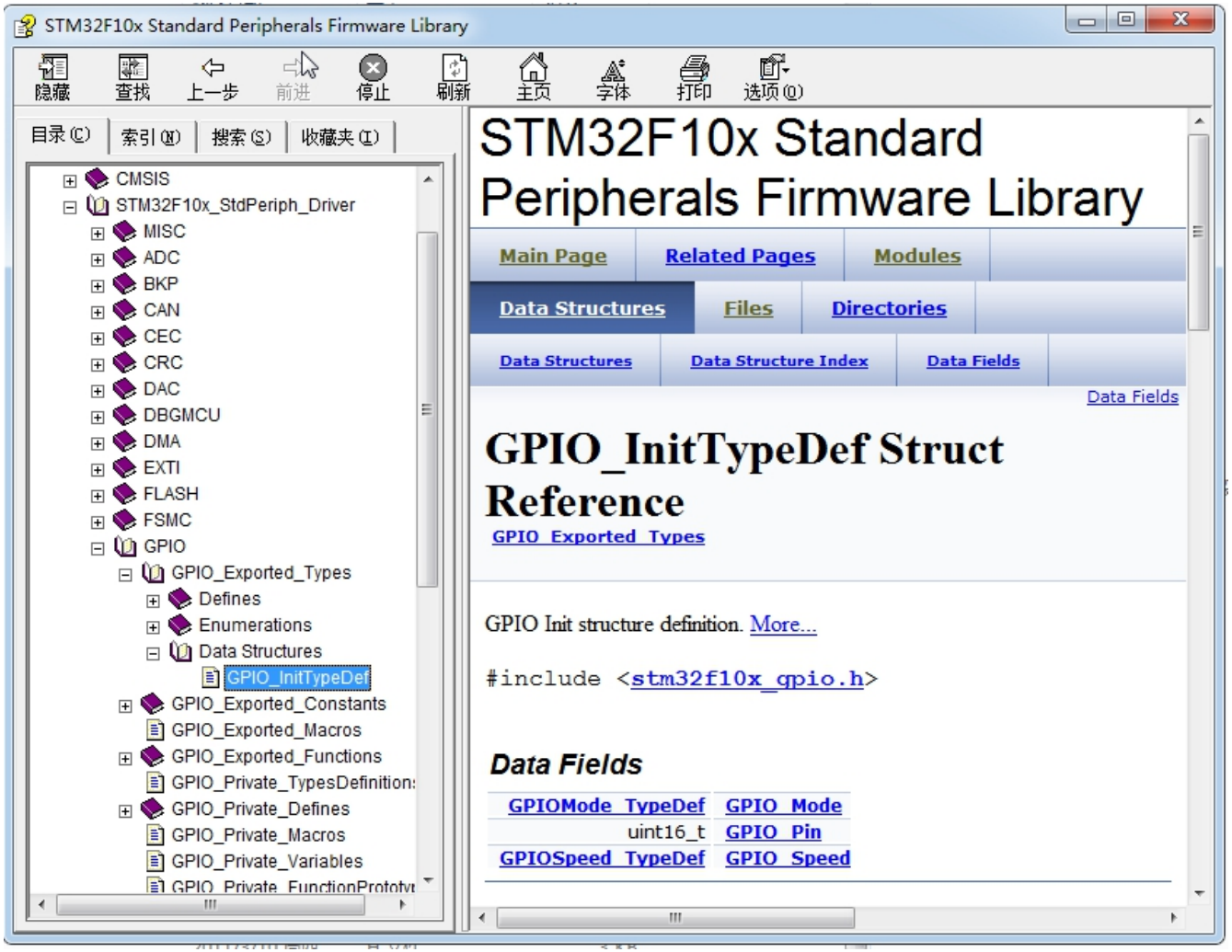
STM32库文件结构

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外，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和改动（用条件编译的方法或加注释的方法）。

库帮助文档

- 在使用库函数时，我们要通过查阅此文件来了解标准库提供了哪些外设、函数原型或库函数的调用的方法。



5.2 固件库中与GPIO相关的api

（1）开启外设时钟

（2）初始化外设

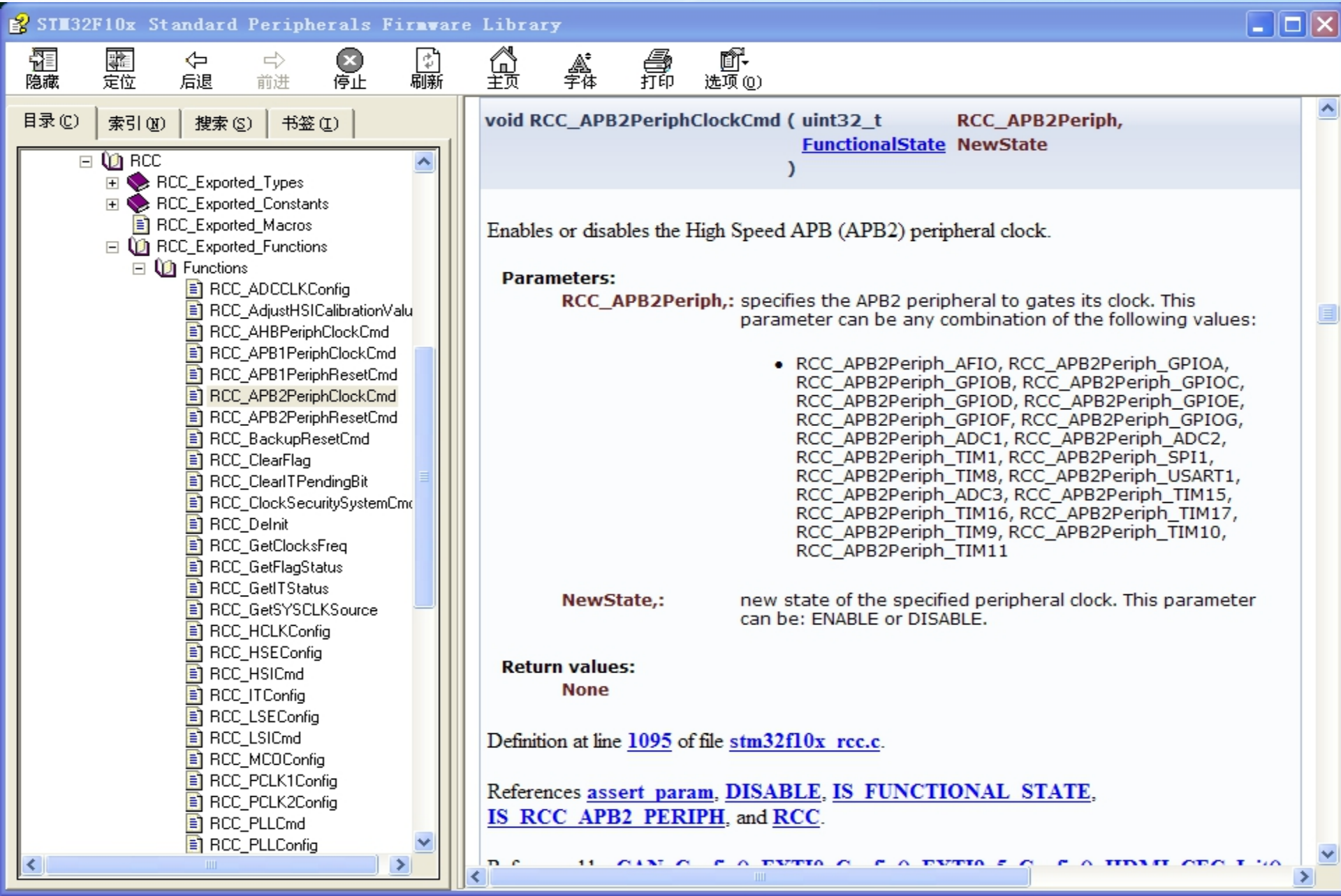
（3）控制外设工作

1. 开启外设时钟

- 使用复位和时钟控制RCC驱动程序（stm32f10x_rcc.c）
- 有3个针对不同总线连接的外设时钟命令函数
  - RCC_AHBPeriphClockCmd
  - RCC_APB1PeriphClockCmd
  - RCC_APB2PeriphClockCmd
- GPIO通过APB2总线连接系统
- 开启GPIO外设时钟的函数

  RCC_APB2PeriphClockCmd

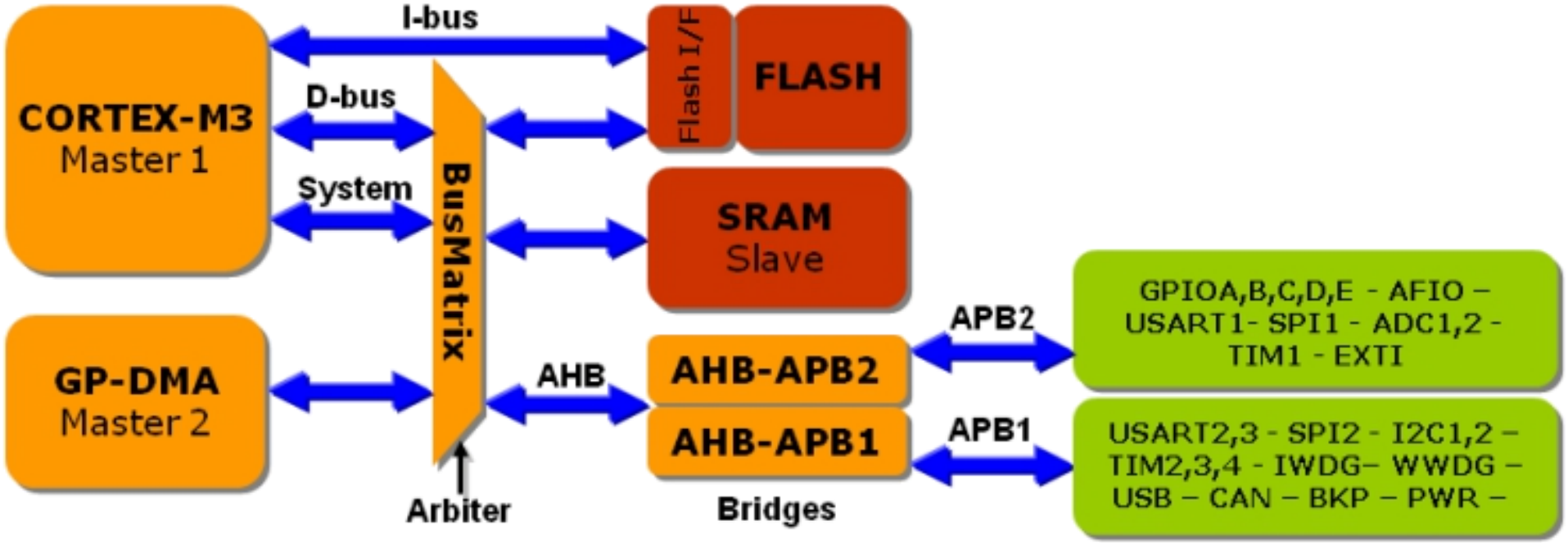
RCC_APB2PeriphClockCmd函数



GPIO通过APB2总线连接系统

- 如：开启GPIOA外设时钟

```
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
```



2. 初始化外设

1. 定义外设初始化结构变量(PPP是外设名称)

```
PPP_InitTypeDef PPP_InitStructure;
```

2. 用允许的成员值填充外设初始化结构成员

```
PPP_InitStructure.member1 = val1;
    …
```

3. 调用PPP_Init函数初始化外设

```
PPP_Init(PPP, &PPP_InitStructure);
```

5. 允许外设开始工作(不是所有的外设都需要)

```
PPP_Cmd(PPP, ENABLE);
```

STM32库给出的外设初始化过程

GPIO的初始化函数GPIO_Init

void GPIO_Init (GPIO_TypeDef * GPIOx,
GPIO_InitTypeDef * GPIO_InitStruct)

Initializes the GPIOx peripheral according to the specified parameters in the GPIO_InitStruct.

Parameters:

GPIOx,: where x can be (A..G) to select the GPIO peripheral.

GPIO_InitStruct,: pointer to a GPIO_InitTypeDef structure that contains the configuration information for the specified GPIO peripheral.

Return values:

None

Definition at line 173 of file stm32f10x_gpio.c.

GPIO的初始化结构类型GPIO_InitTypeDef

```
typedef struct
{  uint16_t GPIO_Pin;
/* 指定配置的GPIO引脚 */
   GPIOSpeed_TypeDef GPIO_Speed;
/* 指定GPIO引脚输出的最高频率 */
   GPIOMode_TypeDef GPIO_Mode;
/* 指定GPIO引脚配置的工作模式 */
}GPIO_InitTypeDef;
```

定义在GPIO头文件（stm32f10x_gpio.h）

GPIO_InitTypeDef成员1：GPIO_Pin

- 要进行配置的GPIO引脚编号
- 其值是常量GPIO_Pin_y（y是0…15和ALL）

```
#define GPIO_Pin_0 ((uint16_t)0x0001)
  /*!< Pin 0 selected */
#define GPIO_Pin_1 ((uint16_t)0x0002)
  /*!< Pin 1 selected */
  ……
#define GPIO_Pin_15     ((uint16_t)0x8000)
  /*!< Pin 15 selected */
#define GPIO_Pin_All    ((uint16_t)0xFFFF)
  /*!< All pins selected */
```

main.c | startup_stm32f10x_hd.s | delay.c | misc.c | led.c | stm32f10x_gpio.h

```
123 /** @defgroup GPIO_pins_define
124   * @{
125   */
126
127 #define GPIO_Pin_0                 ((uint16_t)0x0001)  /*!< Pin 0 selected */
128 #define GPIO_Pin_1                 ((uint16_t)0x0002)  /*!< Pin 1 selected */
129 #define GPIO_Pin_2                 ((uint16_t)0x0004)  /*!< Pin 2 selected */
130 #define GPIO_Pin_3                 ((uint16_t)0x0008)  /*!< Pin 3 selected */
131 #define GPIO_Pin_4                 ((uint16_t)0x0010)  /*!< Pin 4 selected */
132 #define GPIO_Pin_5                 ((uint16_t)0x0020)  /*!< Pin 5 selected */
133 #define GPIO_Pin_6                 ((uint16_t)0x0040)  /*!< Pin 6 selected */
134 #define GPIO_Pin_7                 ((uint16_t)0x0080)  /*!< Pin 7 selected */
135 #define GPIO_Pin_8                 ((uint16_t)0x0100)  /*!< Pin 8 selected */
136 #define GPIO_Pin_9                 ((uint16_t)0x0200)  /*!< Pin 9 selected */
137 #define GPIO_Pin_10                ((uint16_t)0x0400)  /*!< Pin 10 selected */
138 #define GPIO_Pin_11                ((uint16_t)0x0800)  /*!< Pin 11 selected */
139 #define GPIO_Pin_12                ((uint16_t)0x1000)  /*!< Pin 12 selected */
140 #define GPIO_Pin_13                ((uint16_t)0x2000)  /*!< Pin 13 selected */
141 #define GPIO_Pin_14                ((uint16_t)0x4000)  /*!< Pin 14 selected */
142 #define GPIO_Pin_15                ((uint16_t)0x8000)  /*!< Pin 15 selected */
143 #define GPIO_Pin_All               ((uint16_t)0xFFFF)  /*!< All pins selected */
```

- 如：PA0—PA7均使用

```
GPIO_InitStructure.GPIO_Pin=
GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2|GPIO_Pin_3|GP
IO_Pin_4|GPIO_Pin_5|GPIO_Pin_6|GPIO_Pin_7;
```

GPIO_InitTypeDef成员2：GPIO_Speed

- 定义在枚举类型GPIOSpeed_TypeDef 中

```
typedef enum
{   GPIO_Speed_10MHz = 1,
    GPIO_Speed_2MHz,
  // 不赋值的枚举变量自动加1，故此常量值为2
    GPIO_Speed_50MHz     // 常量值为3
}GPIOSpeed_TypeDef;
```

- 选择最高输出频率

```
GPIO_InitStructure.GPIO_Speed=
GPIO_Speed_50MHz;
```

GPIO_InitTypeDef成员3：GPIO_Mode

- 定义在枚举类型GPIOMode_TypeDef 中

```
typedef enum
{ GPIO_Mode_AIN = 0x0,          // 模拟输入模式
  GPIO_Mode_IN_FLOATING = 0x04,    // 浮空输入模式
  GPIO_Mode_IPD = 0x28,         // 下拉输入模式
  GPIO_Mode_IPU = 0x48,         // 上拉输入模式
  GPIO_Mode_Out_OD = 0x14,      // 通用开漏输出模式
  GPIO_Mode_Out_PP = 0x10,      // 通用推挽输出模式
  GPIO_Mode_AF_OD = 0x1C,       // 复用开漏输出模式
  GPIO_Mode_AF_PP = 0x18        // 复用推挽输出模式
}GPIOMode_TypeDef;
```

- 选择工作模式为推挽输出

```
GPIO_InitStructure.GPIO_Mode= GPIO_Mode_Out_PP;
```

根据设定参数初始化GPIOA

```
void LED_Config(void)
{

    GPIO_InitTypeDef  GPIO_InitStructure;

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2|
   GPIO_Pin_3|GPIO_Pin_4|GPIO_Pin_5|GPIO_Pin_6|GPIO_Pin_7;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

}
```

3. 控制外设工作

- 外设驱动库提供控制外设工作的有关函数
- 对GPIO主要是输入和输出数据
- 本例中只需要输出函数，例如
  - 字输出GPIO_Write函数
  - 位输出GPIO_WriteBit函数
  - 置位GPIO_SetBits函数
  - 复位GPIO_ResetBits函数


```
void GPIO_SetBits (GPIO_TypeDef *GPIOx,
    uint16_t GPIO_Pin)
void GPIO_ResetBits (GPIO_TypeDef *GPIOx,
    uint16_t GPIO_Pin)
```

```
void GPIO_SetBits ( GPIO_TypeDef *  GPIOx,
                    uint16_t        GPIO_Pin
                  )
```

Sets the selected data port bits.

Parameters:

GPIOx,: where x can be (A..G) to select the GPIO peripheral.

GPIO_Pin,: specifies the port bits to be written. This parameter can be any combination of GPIO_Pin_x where x can be (0..15).

Return values:

None

Definition at line 358 of file stm32f10x_gpio.c.

References assert_param, GPIO_TypeDef::BSRR, IS_GPIO_ALL_PERIPH, and IS_GPIO_PIN.

Referenced by EXTI15_10_IRQHandler(), EXTI9_5_IRQHandler(), and main().

```
void GPIO_ResetBits ( GPIO_TypeDef *  GPIOx,
                      uint16_t        GPIO_Pin
                    )
```

Clears the selected data port bits.

Parameters:

GPIOx,: where x can be (A..G) to select the GPIO peripheral.

GPIO_Pin,: specifies the port bits to be written. This parameter can be any combination of GPIO_Pin_x where x can be (0..15).

Return values:

None

Definition at line 374 of file stm32f10x_gpio.c.

References assert_param, GPIO_TypeDef::BRR, IS_GPIO_ALL_PERIPH, and IS_GPIO_PIN.

Referenced by EXTI15_10_IRQHandler(), EXTI9_5_IRQHandler(), and TIM2_IRQHandler().

```
void GPIO_Write ( GPIO_TypeDef *  GPIOx,
                  uint16_t        PortVal
                )
```

Writes data to the specified GPIO data port.

Parameters:

GPIOx,: where x can be (A..G) to select the GPIO peripheral.

PortVal,: specifies the value to be written to the port output data register.

Return values:

None

Definition at line 417 of file stm32f10x_gpio.c.

References assert_param, IS_GPIO_ALL_PERIPH, and GPIO_TypeDef::ODR.

```
void GPIO_WriteBit ( GPIO_TypeDef *  GPIOx,
                     uint16_t        GPIO_Pin,
                     BitAction       BitVal
                   )
```

Sets or clears the selected data port bit.

Parameters:

GPIOx,: where x can be (A..G) to select the GPIO peripheral.

GPIO_Pin,: specifies the port bit to be written. This parameter can be one of GPIO_Pin_x where x can be (0..15).

BitVal,: specifies the value to be written to the selected bit. This parameter can be one of the BitAction enum values:

- Bit_RESET: to clear the port pin
- Bit_SET: to set the port pin

Return values:

None

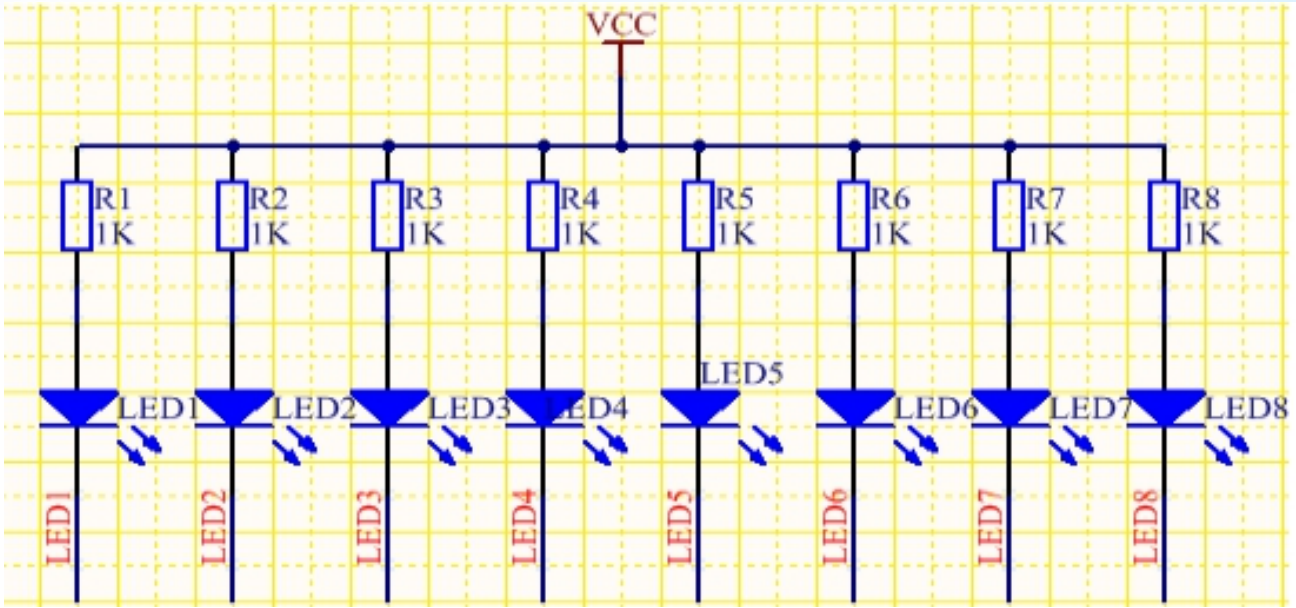
Definition at line 394 of file stm32f10x_gpio.c.

5.3 建立工程-STM32库方式

- 新建工程
- 配置J-LINK硬件调试
- 编译和下载程序

5.4 使用stm32库函数操纵GPIO

- 硬件连接
  - pc3--led1
  - pc2--led2
  - pc1--led3



8个LED为共阳极接法，故：

GPIO引脚输出低电平、LED亮

GPIO引脚输出高电平、LED灭

配置工程环境--修改stm32f10x_conf.h

```
27 /* Uncomment/Comment the line be
28 //#include "stm32f10x_adc.h"
29 //#include "stm32f10x_bkp.h"
30 //#include "stm32f10x_can.h"
31 //#include "stm32f10x_cec.h"
32 //#include "stm32f10x_crc.h"
33 //#include "stm32f10x_dac.h"
34 //#include "stm32f10x_dbgmcu.h"
35 //#include "stm32f10x_dma.h"
36 //#include "stm32f10x_exti.h"
37 //#include "stm32f10x_flash.h"
38 //#include "stm32f10x_fsmc.h"
39 #include "stm32f10x_gpio.h"
40 //#include "stm32f10x_i2c.h"
41 //#include "stm32f10x_iwdg.h"
42 //#include "stm32f10x_pwr.h"
43 #include "stm32f10x_rcc.h"
44 //#include "stm32f10x_rtc.h"
45 //#include "stm32f10x_sdio.h"
46 //#include "stm32f10x_spi.h"
47 //#include "stm32f10x_tim.h"
48 //#include "stm32f10x_usart.h"
49 //#include "stm32f10x_wwdg.h"
50 //#include "misc.h" /* High leve
```

编写用户文件

- 初始化结构体GPIO_InitTypeDef类型--led.c
- 初始化库函数GPIO_Init()--led.c
- 开启外设时钟--led.c
- 控制IO输出高、低电平-led.c,led.h,main.c

Led. c

```
void LED_GPIO_Config(void)
{
    /*定义一个GPIO_InitTypeDef类型的结构体*/
    GPIO_InitTypeDef GPIO_InitStructure;

    /*开启GPIOC的外设时钟*/
    RCC_APB2PeriphClockCmd( RCC_APB2Periph_GPIOC, ENABLE);

    /*选择要控制的GPIOC引脚*/
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_3 | GPIO_Pin_4 | GPIO_Pin_5;

    /*设置引脚模式为通用推挽输出*/
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;

    /*设置引脚速率为50MHz */
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;

    /*调用库函数，初始化GPIOC*/
    GPIO_Init(GPIOC, &GPIO_InitStructure);

    /* 关闭所有led灯     */
    GPIO_SetBits(GPIOC, GPIO_Pin_3 | GPIO_Pin_4 | GPIO_Pin_5);
}
```

Led. h

```
#define ON  0
#define OFF 1

//带参宏，可以像内联函数一样使用
#define LED1(a) if (a)  \
                    GPIO_SetBits(GPIOC,GPIO_Pin_3);\
                    else        \
                    GPIO_ResetBits(GPIOC,GPIO_Pin_3)

#define LED2(a) if (a)  \
                    GPIO_SetBits(GPIOC,GPIO_Pin_4);\
                    else        \
                    GPIO_ResetBits(GPIOC,GPIO_Pin_4)

#define LED3(a) if (a)  \
                    GPIO_SetBits(GPIOC,GPIO_Pin_5);\
                    else        \
                    GPIO_ResetBits(GPIOC,GPIO_Pin_5)
```

main.c

```
int main(void)
{
    /* LED 端口初始化 */
    LED_GPIO_Config();

    while (1)
    {
        LED1( ON );             // 亮
        Delay(0x0FFFEF);
        LED1( OFF );            // 灭

        LED2( ON );
        Delay(0x0FFFEF);
        LED2( OFF );

        LED3( ON );
        Delay(0x0FFFEF);
        LED3( OFF );
    }
}

void Delay(__IO u32 nCount)  //简单的延时函数
{
    for(; nCount != 0; nCount--);
}
```