第 3 章
无人机飞控系统的底层开发

本章在介绍 STM32 系列微控制器 GPIO、时钟配置、ADC、DMA、TIM、USART 和模拟 I2C 总线等模块基础上，实现了对无人机状态指示灯的控制、无人机系统时钟的配置、无人机电池电压的读取、无人机控制信号的输出、无人机遥控信号的接收、无人机数据的收发、无人机 MAVLink 消息的收发，以及通过 I2C 总线读 EEPROM。本章将逐一对上述模块的相关原理进行详细介绍，并以实例开发的形式帮助读者掌握无人机状态、输入/输出控制信号的实现原理与过程，完成无人机飞控系统的底层开发。

3.1 无人机状态指示灯的控制

学习目标

了解 ARM Cortex-M 系列芯片的 GPIO 分类，通过配置 STM32F407 芯片的 GPIO 相关寄存器控制 LED 指示灯的亮灭，根据事先约定好的亮灭规则判断无人机的当前状态。

3.1.1 开发原理

LED 指示灯的控制

实时了解飞控系统的 LED 指示灯，可以直观地了解无人机当前存在的问题。“光标”飞控系统是通过 4 个 LED 指示灯的状态来表示无人机当前运行情况的，如表 3-1 所示，4 个 LED 指示灯分别为 LED1、LED2、LED3、LED4。

表 3-1　LED 状态指示

LED 指示灯	LED 指示灯的状态	对应的无人机的当前运行情况
LED1	闪烁	陀螺仪、加速计运行异常
	常亮	陀螺仪、加速计运行正常
LED2	闪烁	磁力计运行异常
	常亮	磁力计运行正常
LED3	闪烁	气压计或光流模块运行异常
	常亮	气压计或光流模块运行正常
LED4	闪烁	系统处于待机状态
	常亮	系统处于启动状态

“光标”飞控系统是通过 STM32F407 的 GPIO 来控制 LED 的亮灭及闪烁的。STM32F407 共有 7 组 GPIO 端口，分别为 GPIOA、GPIOB、…、GPIOG，每组 GPIO 端口均有 16 个 GPIO 端口，共 112 个 GPIO 端口，另外再加上 PH0 和 PH1，因此 STM32F407 有 114 个 GPIO 端口。

GPIO 端口模式包括输入模式、输出模式、复用模式、模拟模式，端口类型包括推挽和开漏两种类型，端口状态分为上拉、下拉和浮空三种状态。STM32F407 的每个 GPIO 端口的输出频率均可设置为 2 MHz、25 MHz、50 MHz 或 100 MHz。推挽类型的 GPIO 端口可以输出高电平和低电平，用于连接数字器件；开漏类型的 GPIO 端口只能输出低电平，适用于电流型的驱动，其吸收电流的能力相对较强，当需要高电平时可以通过外部上拉电阻将低电平拉高。

STM32F407 的 GPIO 端口的结构如图 3-1 所示。

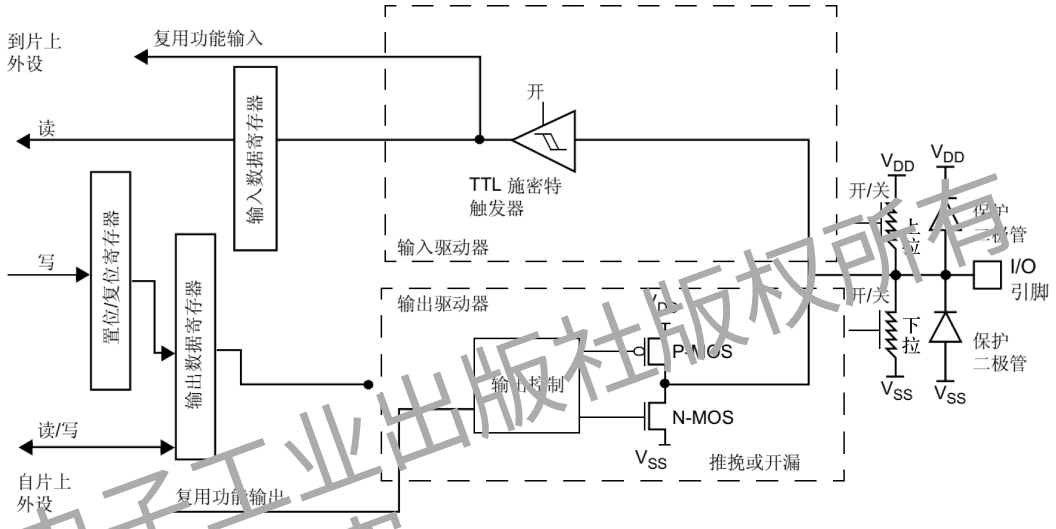

图 3-1　STM32F407 的 GPIO 端口的结构

每组 GPIO 端口寄存器包括模式寄存器（GPIOx_MODER）、输出类型寄存器（GPIOx_OTYPER）、输出速度寄存器（GPIOx_OSPEEDR）、上拉/下拉寄存器（GPIOx_PUPDR）、输入数据寄存器（GPIOx_IDR）、输出数据寄存器（GPIOx_ODR）、置位/复位寄存器（GPIOx_BSRR）、配置锁存寄存器（GPIOx_LCKR）、两个复位功能寄存器（低位 GPIOx_AFRL 和高位 GPIOx_AFRH）。

（1）模式寄存器（GPIOx_MODER）如图 3-2 所示，其中位 2*y*:2*y*+1（*y*=0～15）是 MODER*y*[1:0]，表示端口配置位，通过软件向这些位写入数据可以配置 GPIO 的模式，00 表示输入模式，01 表示输出模式，10 表示复用模式，11 表示模拟模式。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

图 3-2　模式寄存器

（2）输出类型寄存器（GPIOx_OTYPER）如图 3-3 所示，其中位 31:16 为保留位，必须保持复位值；位 15:0 是 OT*y*（*y*=0～15），是端口配置位，通过软件向这些位写入数据可以

配置 GPIO 端口的类型，0 表示推挽类型，1 表示开漏类型。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

图 3-3　输出类型寄存器

（3）输出速度寄存器（GPIOx_OSPEEDR）如图 3-4 所示，其中位 2*y*:2*y*+1（*y*=0～15）是 OSPEEDR*y*[1:0]，是端口配置位，通过软件向这些位写入数据可以配置 GPIO 的输出频率（速率），00 表示 2 MHz（低速），01 表示 25 MHz（中速），10 表示 50 MHz（快速），当负载为 30 pF 时 11 表示 100 MHz（高速），当负载为 15 pF 时 11 表示 80 MHz（最大速度）。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEEDR15[1:0]		OSPEEDR14[1:0]		OSPEEDR13[1:0]		OSPEEDR12[1:0]		OSPEEDR11[1:0]		OSPEEDR10[1:0]		OSPEEDR9[1:0]		OSPEEDR8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1[1:0]		OSPEEDR0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

图 3-4　输出速度寄存器

（4）上拉/下拉寄存器（GPIOx_PUPDR）如图 3-5 所示，其中位 2*y*:2*y*+1（*y*=0～15）是 PUPDR*y*[1:0]，是端口配置位，通过软件向这些位写入数据可以配置 GPIO 的上拉或下拉，00 表示无上拉或下拉，01 表示上拉，10 表示下拉，11 为保留。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

图 3-5　上拉/下拉寄存器

（5）输入数据寄存器（GPIOx_IDR）如图 3-6 所示，其中位 31:16 为保留位，必须保持复位值；位 15:0 是 IDR*y*（*y*=0～15），用于保存输入数据，这些位都是只读的，只能在字模式下访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

图 3-6　输入数据寄存器

（6）输出数据寄存器（GPIOx_ODR）如图 3-7 所示，其中位 31:16 为保留位，必须保持复位值；位 15:0 是 ODR*y*（*y*=0～15），用于保存输出数据，通过软件对这些位进行读写操作。注意：对于原子置位/复位，通过写入 GPIOx_BSRR，可以分别对 ODR 位进行置位和复位操作。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

图 3-7　输出数据寄存器

（7）置位/复位寄存器（GPIOx_BSRR）如图 3-8 所示。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

图 3-8　置位/复位寄存器

（8）配置锁定寄存器（GPIOx_LCKR）如图 3-9 所示。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															LCKK
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

图 3-9　配置锁定寄存器

（9）复用功能低位寄存器（GPIOx_AFRL）如图 3-10 所示。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFRL7[3:0]				AFRL6[3:0]				AFRL5[3:0]				AFRL4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFRL3[3:0]				AFRL2[3:0]				AFRL1[3:0]				AFRL0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

图 3-10　复用功能低位寄存器

（10）复用功能高位寄存器（GPIOx_AFRH）如图 3-11 所示。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFRH15[3:0]				AFRH14[3:0]				AFRH13[3:0]				AFRH12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFRH11[3:0]				AFRH10[3:0]				AFRH9[3:0]				AFRH8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

图 3-11　复用功能高位寄存器

关于 GPIO 寄存器的详细资料，请参考 STM32F407 的芯片手册。

串口+中断　串口通信

3.1.2　开发步骤

（1）查看“光标”飞控系统的电路原理图，了解 LED 连接的芯片引脚，以便配置 GPIO。LED 的电路原理图如图 3-12 所示。

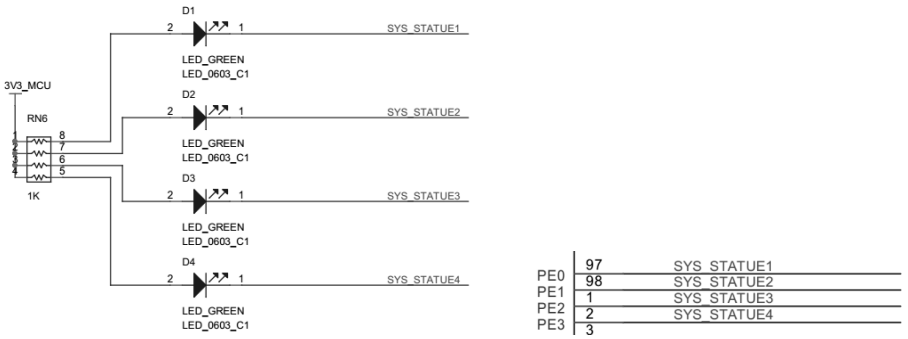

图 3-12　LED 的电路原理图

（2）由 STM32F407 的方框图（见图 3-13）可知，GPIO 连接在 AHB1 时钟线上。

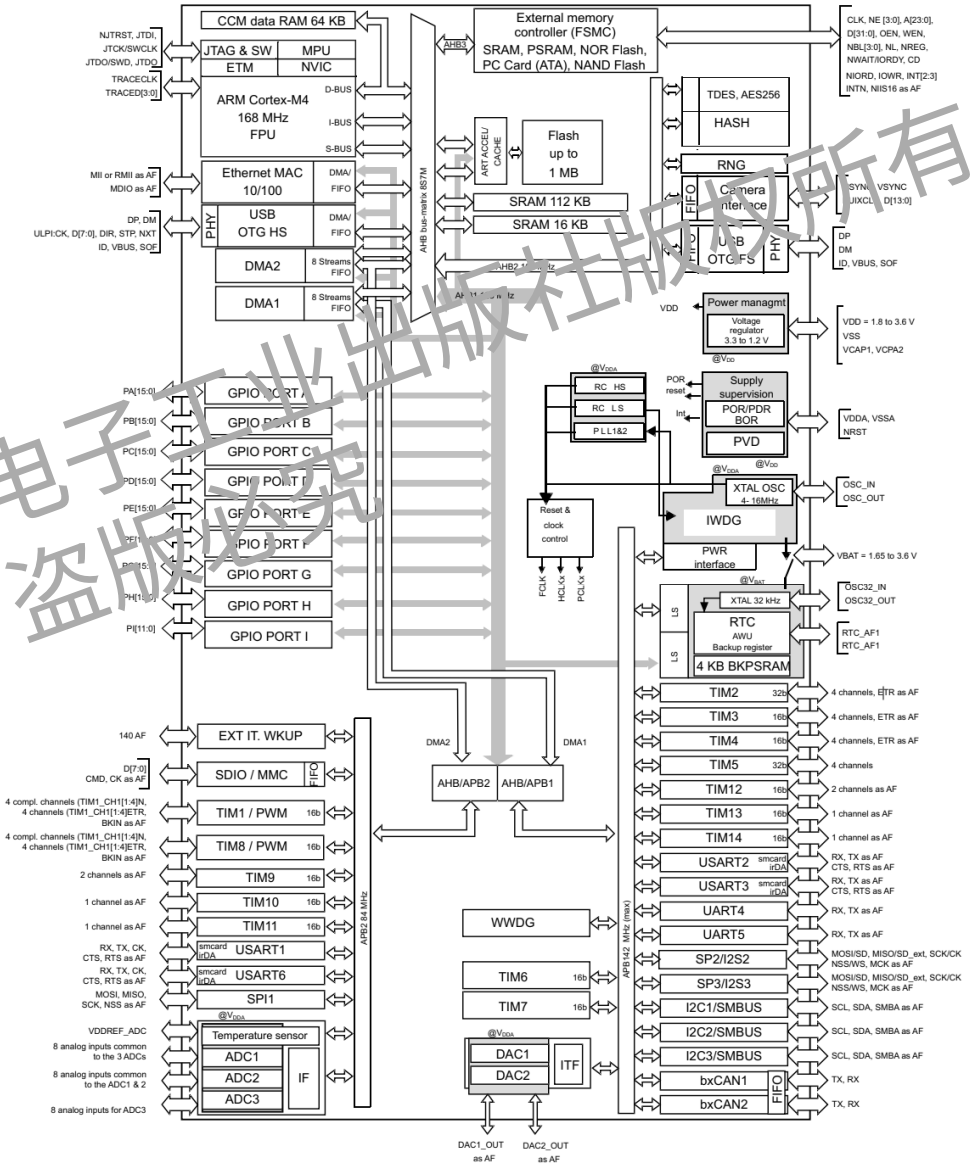

图 3-13　STM32F4 方框图

（3）新建一个工程。在工程文件夹 User\bsp_stm32f4xx\src 中新建一个文件，并命名为 bsp_led.c 文件，如图 3-14 所示。

图 3-14　新建 bsp_led.c 文件

（4）在工程文件夹 User\bsp_stm32f4xx\inc 中新建一个文件，并命名为 bsp_led.h 文件，如图 3-15 所示。

图 3-15　新建 bsp_led.h 文件

（5）打开工程后新建 BSP 目录，如图 3-16 所示，并将 bsp_led.c 文件添加到工程中。

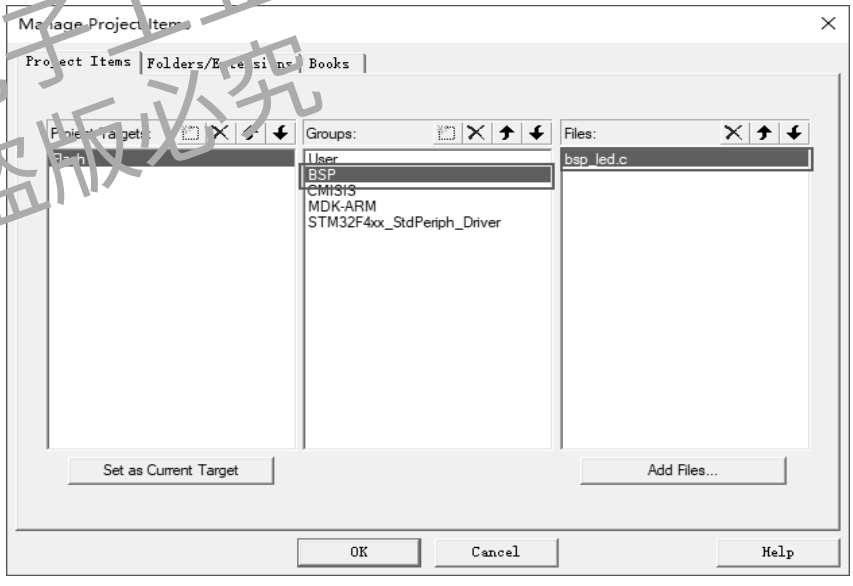

图 3-16　打开工程后新建 BSP 目录

（6）将 bsp_led.h 文件的路径添加到工程中，如图 3-17 所示。

（7）在 bsp_led.c 文件中包含 bsp_led.h 文件，代码如下：

```
#include "bsp_led.h"
```

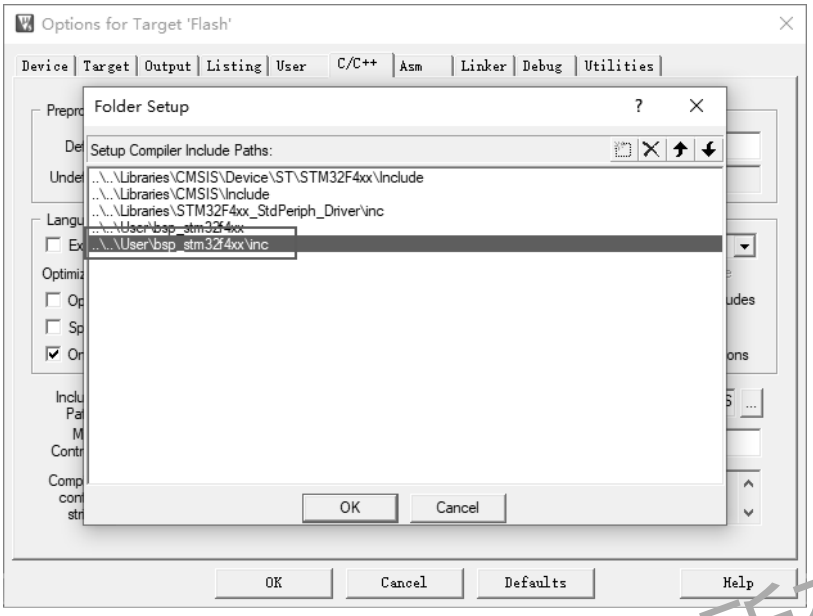

图 3-17　将 bsp_led.h 文件的路径添加到工程中

（8）在 bsp_led.c 文件中定义 LED_Init()函数，代码如下：

```
//LED 的 GPIO 初始化函数
void LED_Init(void)
{
}
```

（9）在 LED_Init()函数中配置 GPIO。调用 RCC_AHB1PeriphClockCmd()函数开启 GPIO 时钟，打开 STM32F4xx_StdPeriph_Driver 下的 stm32f4xx_rcc.c 文件，跳转到 stm32f4xx_rcc.h 文件，如图 3-18 所示。

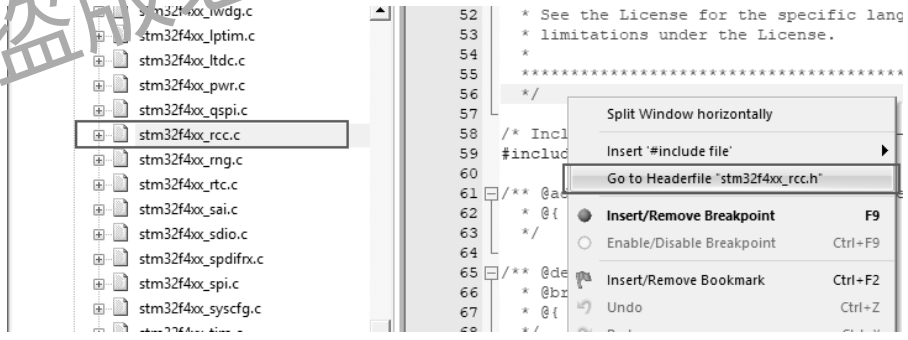

图 3-18　打开 stm32f4xx_rcc.c 文件并跳转到 stm32f4xx_rcc.h

（10）在 stm32f4xx_rcc.h 文件底部有多个时钟使能函数，单击鼠标右键可跳转到 RCC_AHB1PeriphClockCmd()函数，如图 3-19 所示。

（11）在 RCC_AHB1PeriphClockCmd()函数中可以发现连接 LED 的 GPIOE 就是在这个函数中被使能的，所以需要将 RCC_AHB1PeriphClockCmd()函数复制到我们创建的 LED_Init()函数中。RCC_AHB1PeriphClockCmd()函数如图 3-20 所示。

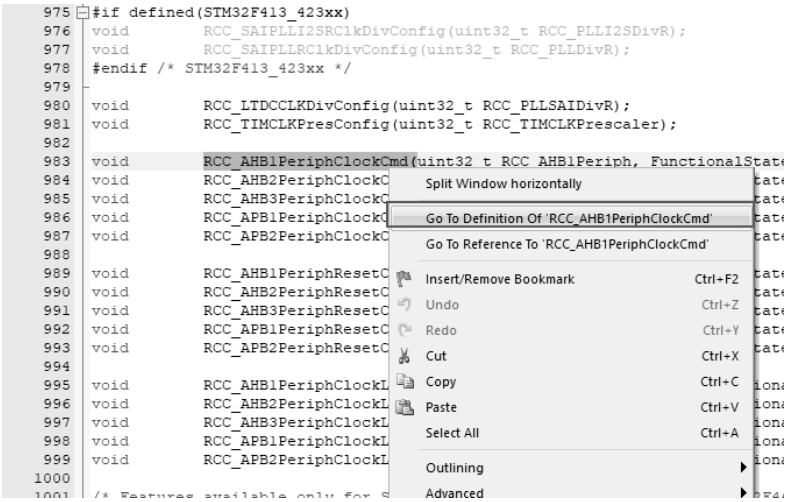

图 3-19　跳转到 RCC_AHB1PeriphClockCmd()函数

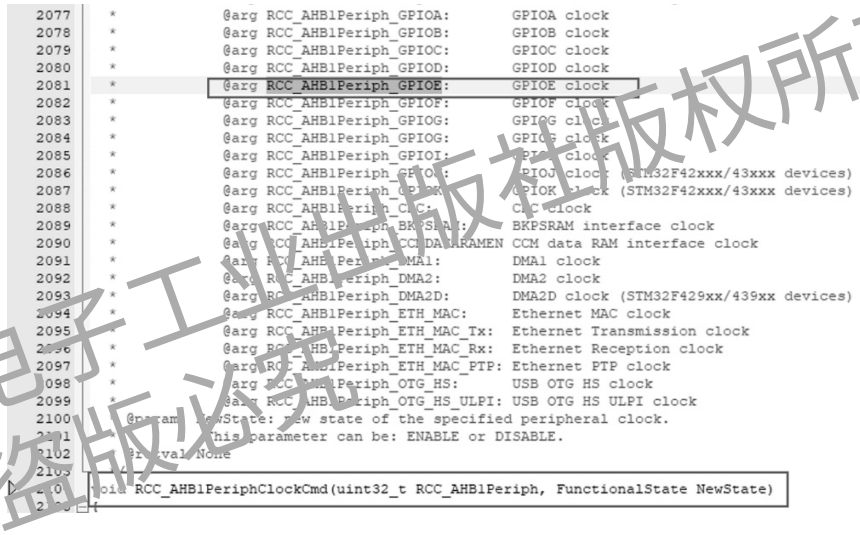

图 3-20　RCC_AHB1PeriphClockCmd()函数

```
//LED 的 GPIO 初始化函数
void LED_Init(void)
{
    RCC_AHB1PeriphClockCmd();
}
```

（12）通过鼠标右键跳转到 RCC_AHB1PeriphClockCmd()函数的两个参数列表，找到参数 RCC_AHBPeriph_GPIOE 和 ENABLE，将这两个参数填入 RCC_AHB1PeriphClockCmd()函数中。参数 RCC_AHBPeriph_GPIOE 和 ENABLE 如图 3-21 所示。

```
//LED 的 GPIO 初始化函数
void LED_Init(void)
{
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOE , ENABLE); //开启 GPIOE 的时钟
}
```