STM32F030使用手册

**作者: 向仔州**

目录

[开发环境搭建 2](#_Toc179553501)

[GPIO输入输出功能 7](#_Toc179553502)

[GPIO输出功能 7](#_Toc179553503)

[延时函数实现 7](#_Toc179553504)

[GPIO输入功能 8](#_Toc179553505)

[GPIO外部中断触发功能 8](#_Toc179553506)

[串口1发送接收实现 10](#_Toc179553507)

[串口1发送 10](#_Toc179553508)

[串口1 打印输出实现 11](#_Toc179553509)

[串口1 接收中断实现 11](#_Toc179553510)

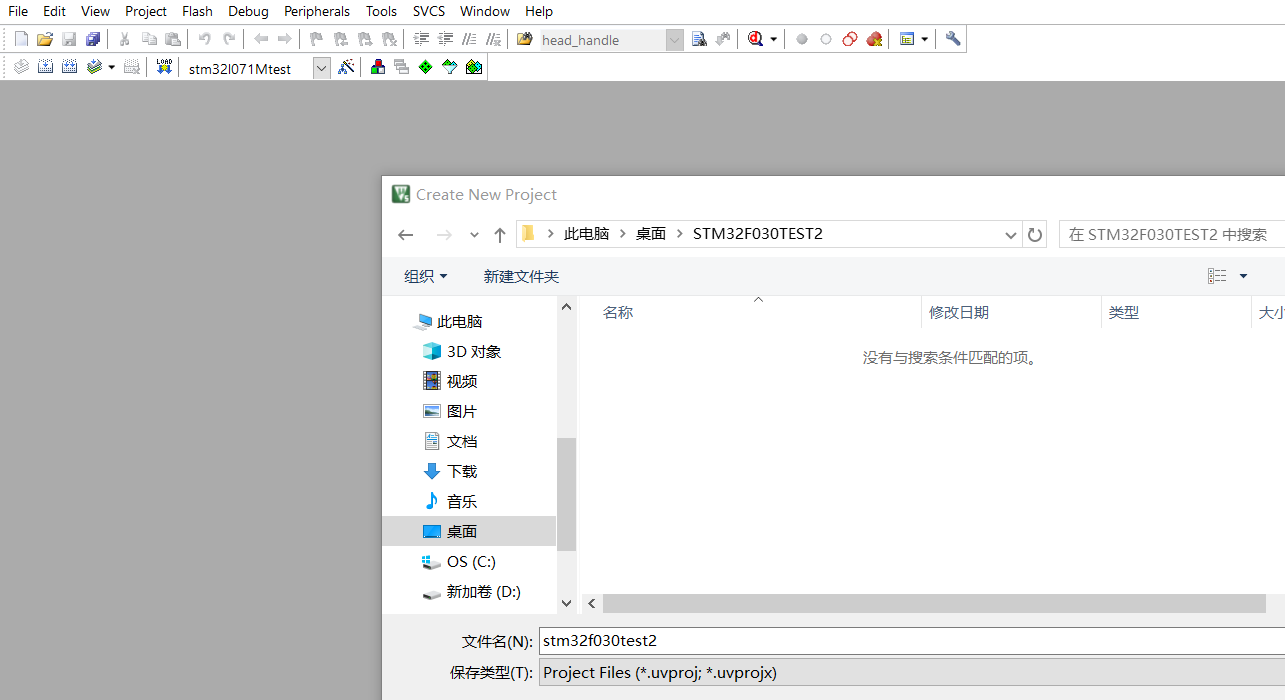
[串口1 DMA空闲中断接收数据 13](#_Toc179553511)

[定时器TIME3使用 15](#_Toc179553512)

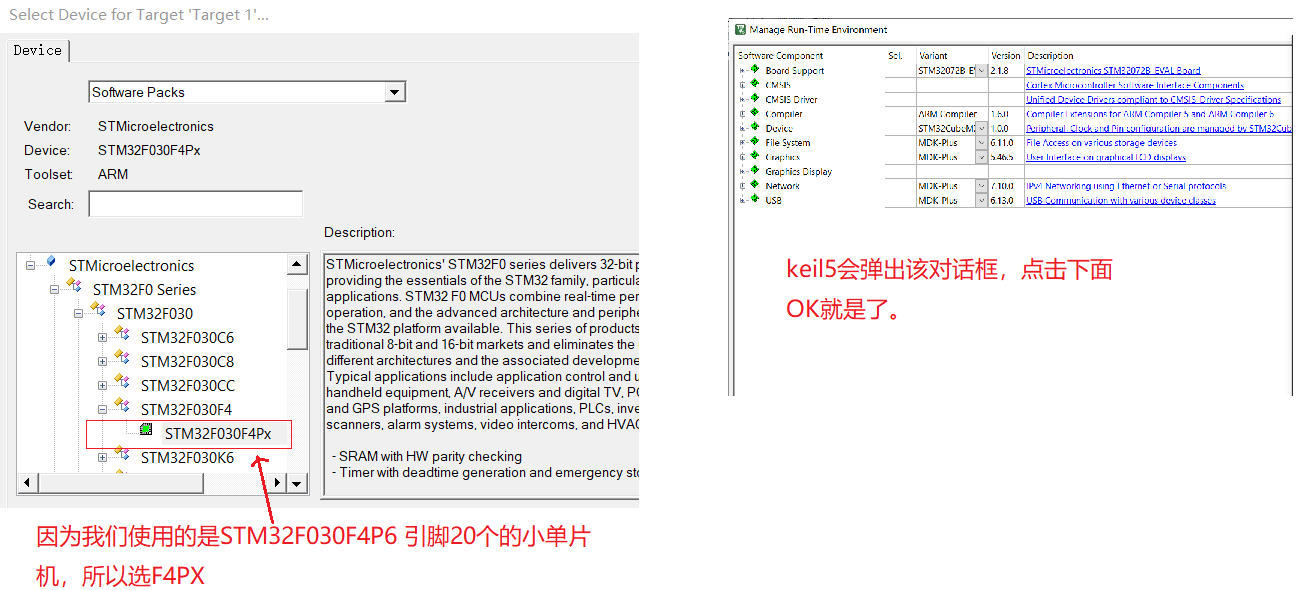
## 开发环境搭建

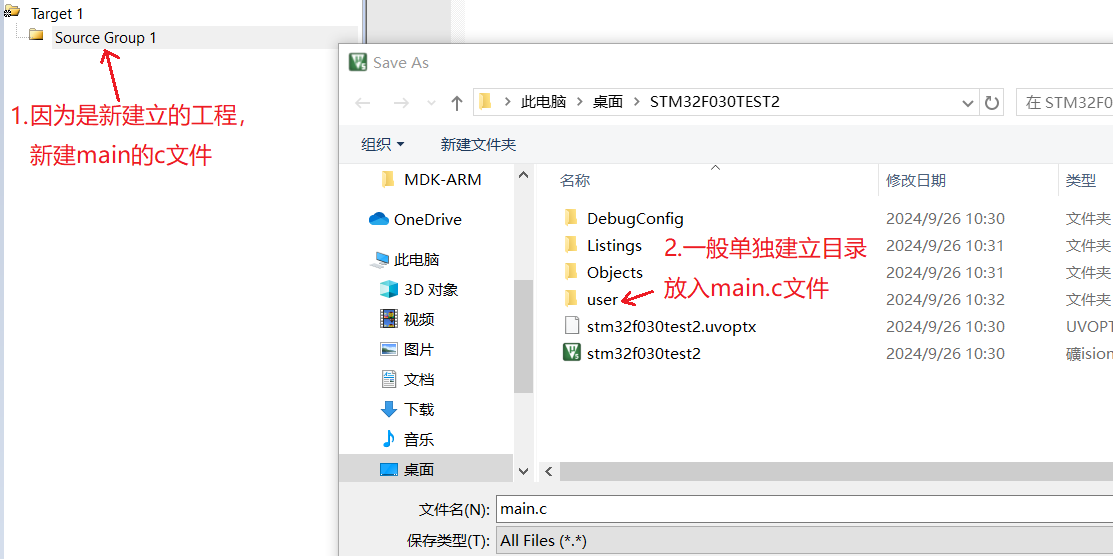
MDK5.0 keil开发安装环境按照网上的安装

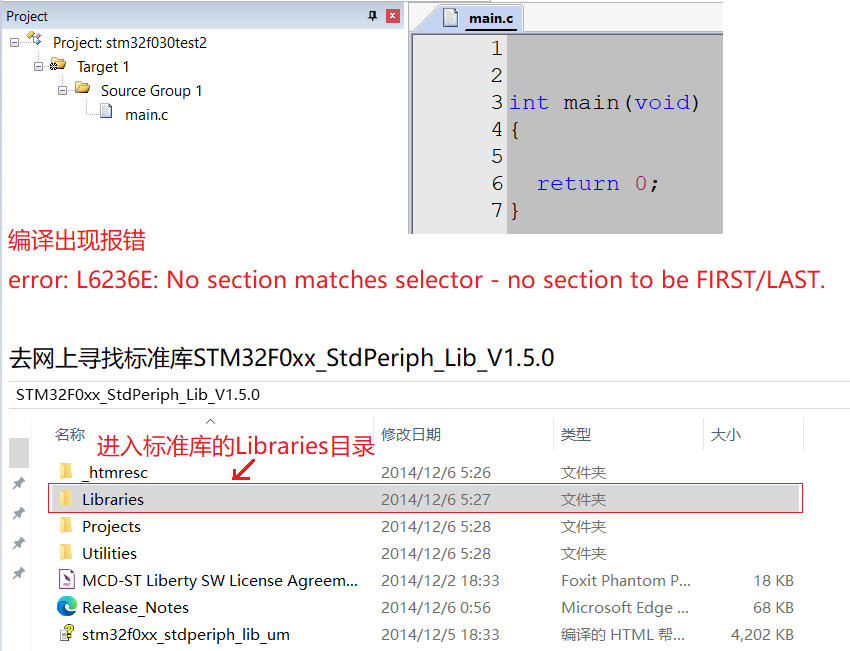
C:\Users\ASUS\Documents\WeChat Files\wxid_np6asaq3n7f322\FileStorage\Temp\1727317433155.pngMDK安装完成之后，安装STM32F0补丁包。

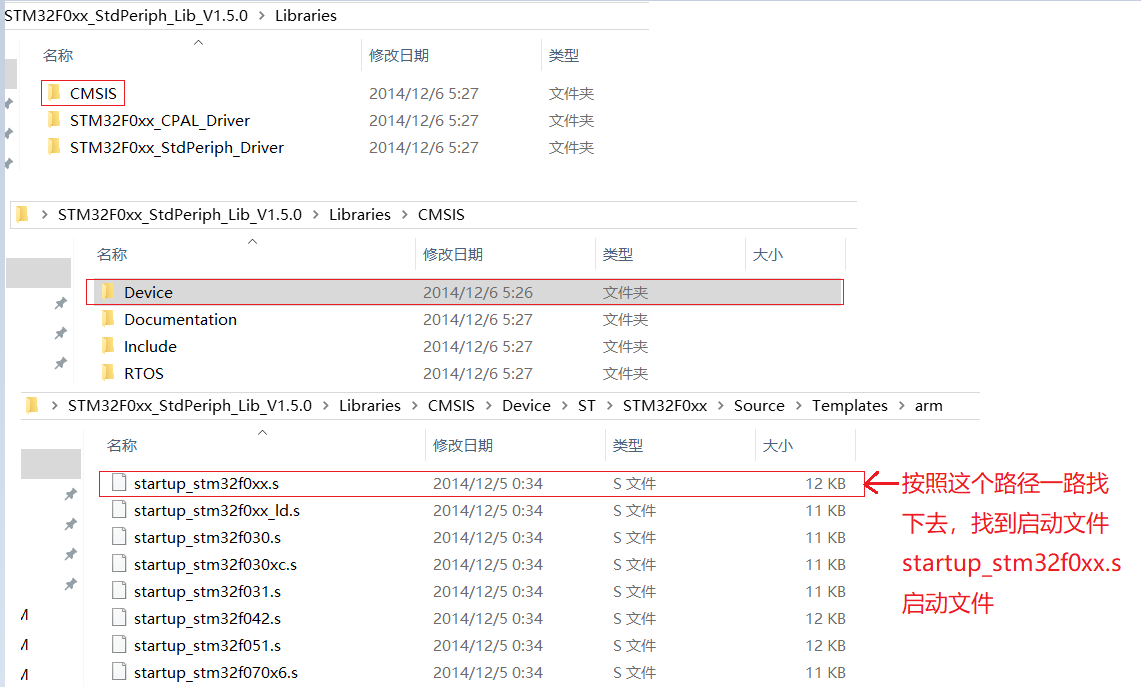


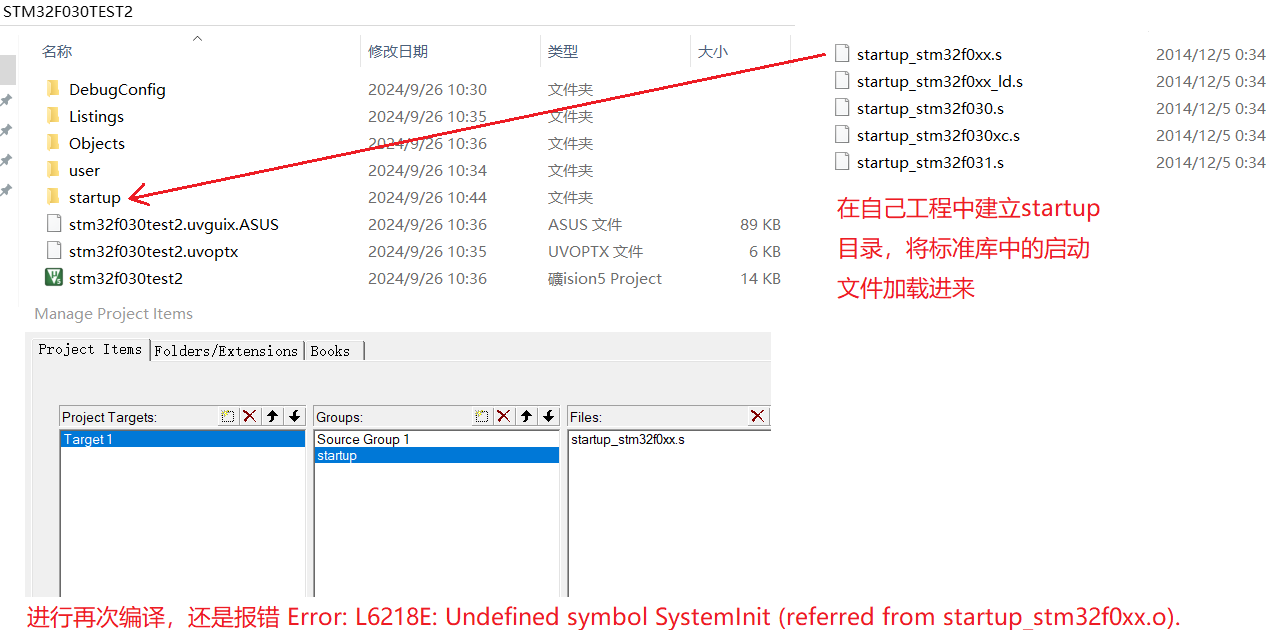
新建STM32F0工程

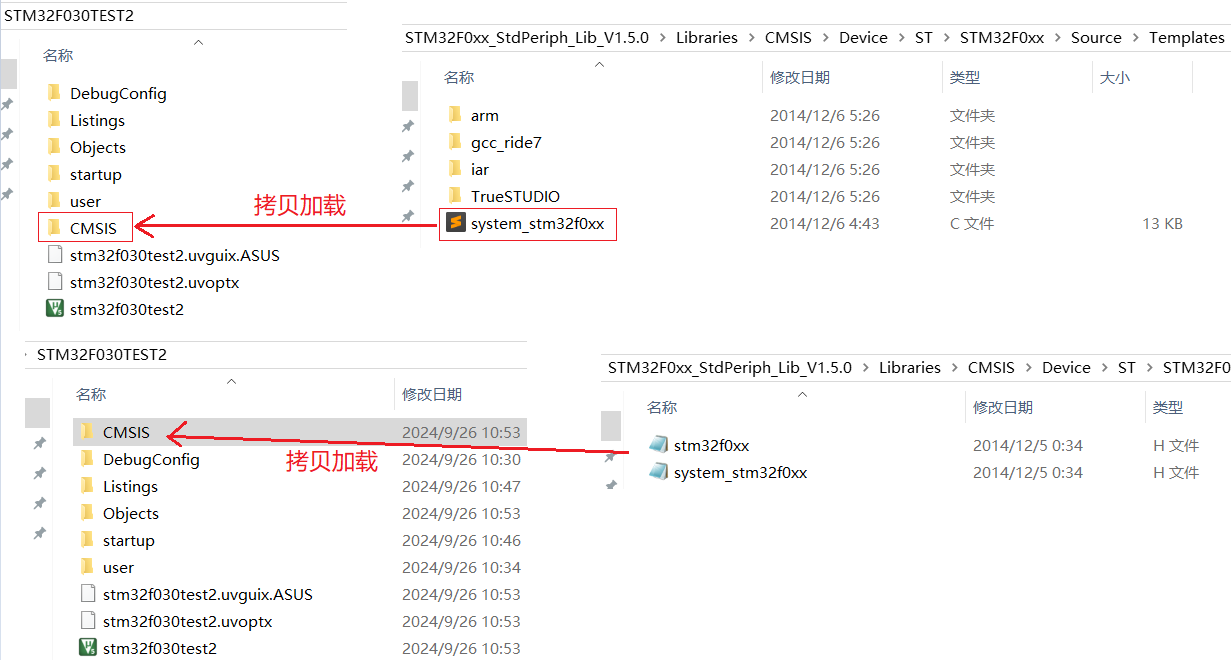


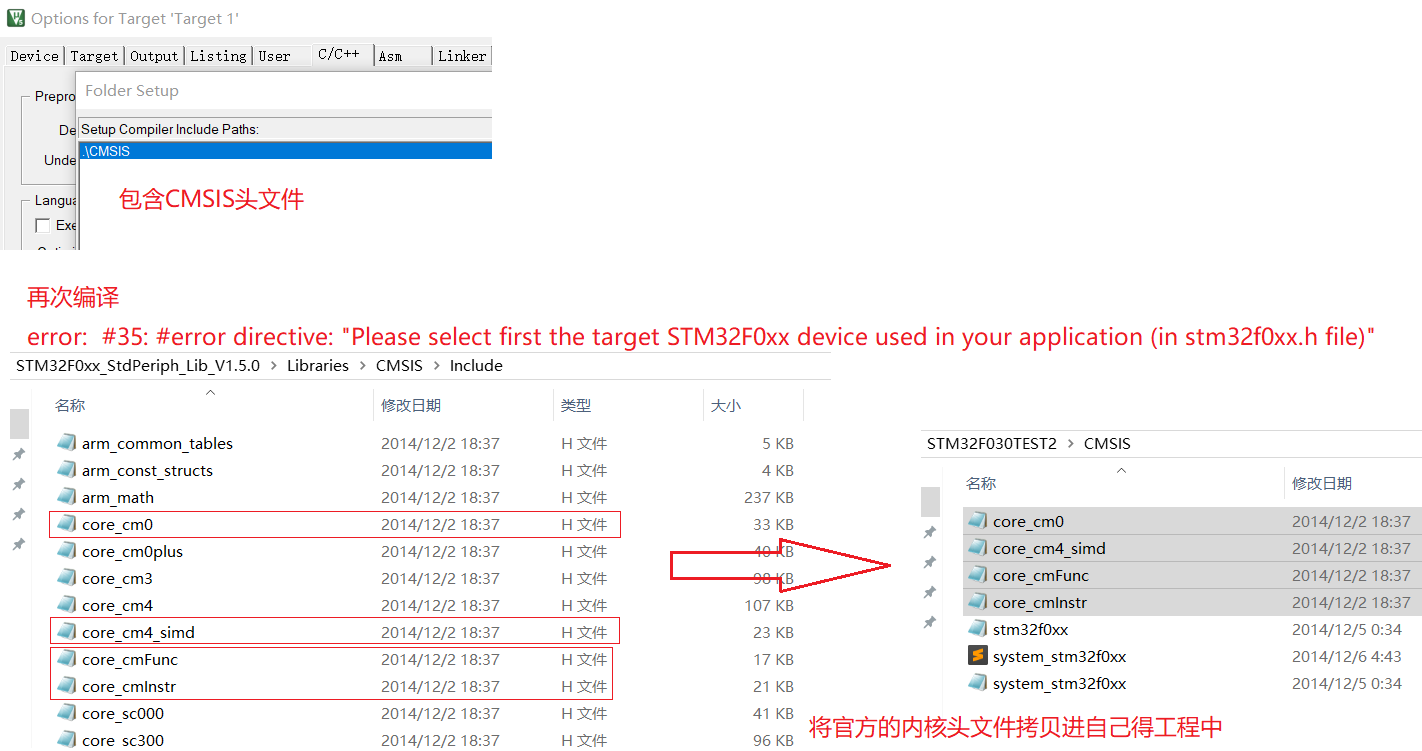


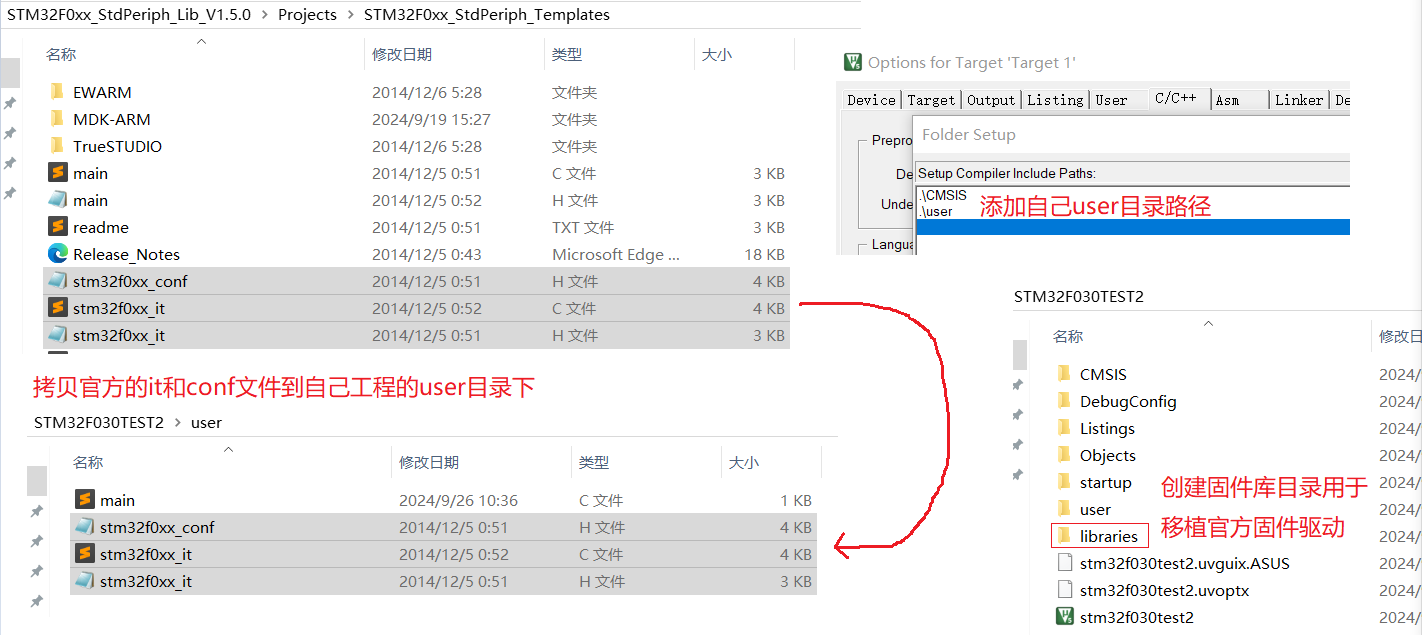




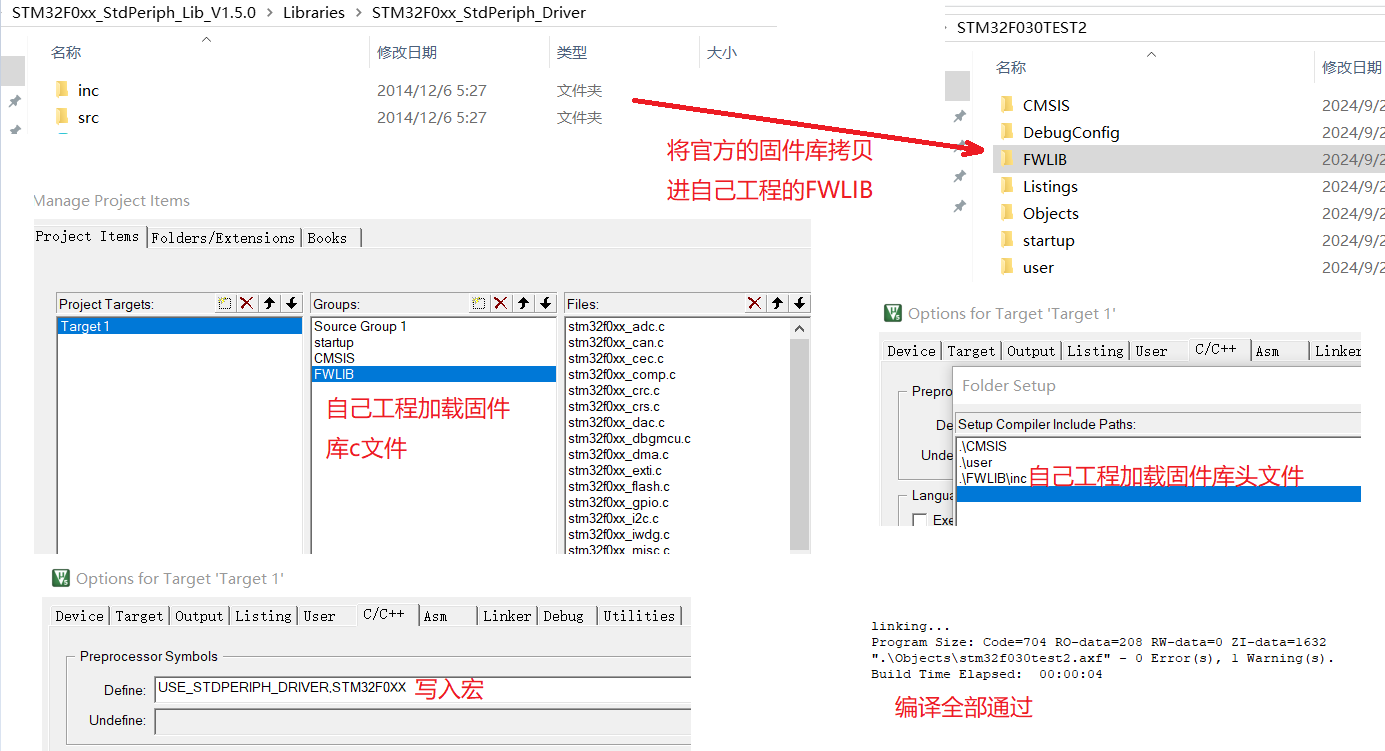








我将libraries目录改名炜FWLIB



USE\_STDPERIPH\_DRIVER,STM32F0XX

在SystemInit函数中，系统默认使用的是外部时钟起振配置。

void SystemInit (void){

…………………

SetSysClock();

}

以下就是外部时钟起振代码

static void SetSysClock(void)

{

\_\_IO uint32\_t StartUpCounter = 0, HSEStatus = 0;

/\* SYSCLK, HCLK, PCLK configuration ----------------------------------------\*/

/\* Enable HSE \*/

RCC->CR |= ((uint32\_t)RCC\_CR\_HSEON);

/\* Wait till HSE is ready and if Time out is reached exit \*/

do

{

HSEStatus = RCC->CR & RCC\_CR\_HSERDY;

StartUpCounter++;

} while((HSEStatus == 0) && (StartUpCounter != HSE\_STARTUP\_TIMEOUT));

if ((RCC->CR & RCC\_CR\_HSERDY) != RESET)

{

HSEStatus = (uint32\_t)0x01;

}

else

{

HSEStatus = (uint32\_t)0x00;

}

if (HSEStatus == (uint32\_t)0x01)

{

/\* Enable Prefetch Buffer and set Flash Latency \*/

FLASH->ACR = FLASH\_ACR\_PRFTBE | FLASH\_ACR\_LATENCY;

/\* HCLK = SYSCLK \*/

RCC->CFGR |= (uint32\_t)RCC\_CFGR\_HPRE\_DIV1;

/\* PCLK = HCLK \*/

RCC->CFGR |= (uint32\_t)RCC\_CFGR\_PPRE\_DIV1;

/\* PLL configuration = HSE \* 6 = 48 MHz \*/

RCC->CFGR &= (uint32\_t)((uint32\_t)~(RCC\_CFGR\_PLLSRC | RCC\_CFGR\_PLLXTPRE | RCC\_CFGR\_PLLMULL));

RCC->CFGR |= (uint32\_t)(RCC\_CFGR\_PLLSRC\_PREDIV1 | RCC\_CFGR\_PLLXTPRE\_PREDIV1 | RCC\_CFGR\_PLLMULL6);

/\* Enable PLL \*/

RCC->CR |= RCC\_CR\_PLLON;

/\* Wait till PLL is ready \*/

while((RCC->CR & RCC\_CR\_PLLRDY) == 0)

{

}

/\* Select PLL as system clock source \*/

RCC->CFGR &= (uint32\_t)((uint32\_t)~(RCC\_CFGR\_SW));

RCC->CFGR |= (uint32\_t)RCC\_CFGR\_SW\_PLL;

/\* Wait till PLL is used as system clock source \*/

while ((RCC->CFGR & (uint32\_t)RCC\_CFGR\_SWS) != (uint32\_t)RCC\_CFGR\_SWS\_PLL)

{

}

}

else

{ /\* If HSE fails to start-up, the application will have wrong clock

configuration. User can add here some code to deal with this error \*/

}

}

现在改为内部时钟起振代码

static void SetSysClock2(void)

{

\_\_IO uint32\_t StartUpCounter = 0, HSIStatus = 0;

/\* SYSCLK, HCLK, PCLK configuration ----------------------------------------\*/

/\* Enable HSI\*/ //使能内部时钟

RCC->CR |= ((uint32\_t)RCC\_CR\_HSION);

//等待内部时钟起振

do

{

HSIStatus = RCC->CR & RCC\_CR\_HSIRDY;

StartUpCounter++;

} while((HSIStatus== 0) && (StartUpCounter != HSI\_STARTUP\_TIMEOUT));

if ((RCC->CR & RCC\_CR\_HSIRDY) != RESET)

{

HSIStatus = (uint32\_t)0x01;

}

else

{

HSIStatus = (uint32\_t)0x00;

}

if (HSIStatus == (uint32\_t)0x01)

{

/\* Enable Prefetch Buffer and set Flash Latency \*/ //flash总线时钟使能

FLASH->ACR = FLASH\_ACR\_PRFTBE | FLASH\_ACR\_LATENCY;

/\* HCLK = SYSCLK \*///外设AHB总线时钟等于系统时钟

RCC->CFGR |= (uint32\_t)RCC\_CFGR\_HPRE\_DIV1;

/\* PCLK = HCLK \*///外设APB总线时钟等于系统时钟

RCC->CFGR |= (uint32\_t)RCC\_CFGR\_PPRE\_DIV1;

/\* PLL configuration = HSI/2 \* 12= 48 MHz \*/

RCC->CFGR &= (uint32\_t)((uint32\_t)~(RCC\_CFGR\_PLLSRC | RCC\_CFGR\_PLLSRC | RCC\_CFGR\_PLLMULL));

RCC->CFGR |= (uint32\_t)(RCC\_CFGR\_PLLSRC\_HSI\_Div2 | RCC\_CFGR\_PLLMULL12); //RC时钟2分频后 进行12倍频

/\* Enable PLL \*///使能锁相环倍频开关

RCC->CR |= RCC\_CR\_PLLON;

/\* Wait till PLL is ready \*///等待锁相环就绪

while((RCC->CR & RCC\_CR\_PLLRDY) == 0);

/\* Select PLL as system clock source \*///选择锁相环输出时钟作为系统时钟

RCC->CFGR &= (uint32\_t)((uint32\_t)~(RCC\_CFGR\_SW));

RCC->CFGR |= (uint32\_t)RCC\_CFGR\_SW\_PLL;

/\* Wait till PLL is used as system clock source \*///等待锁相环输出时钟已经成为系统时钟

while ((RCC->CFGR & (uint32\_t)RCC\_CFGR\_SWS) != (uint32\_t)RCC\_CFGR\_SWS\_PLL);

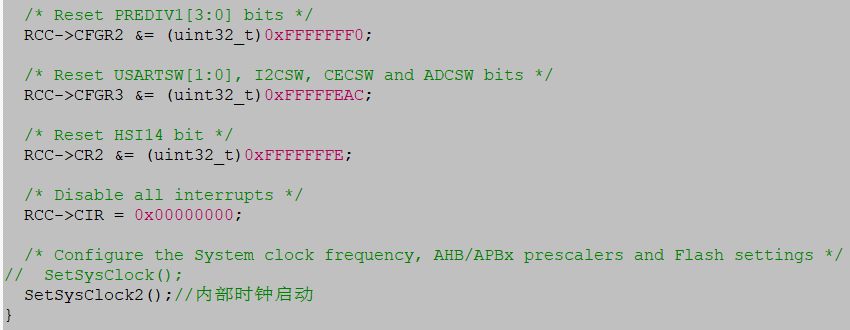
}

else

{

}

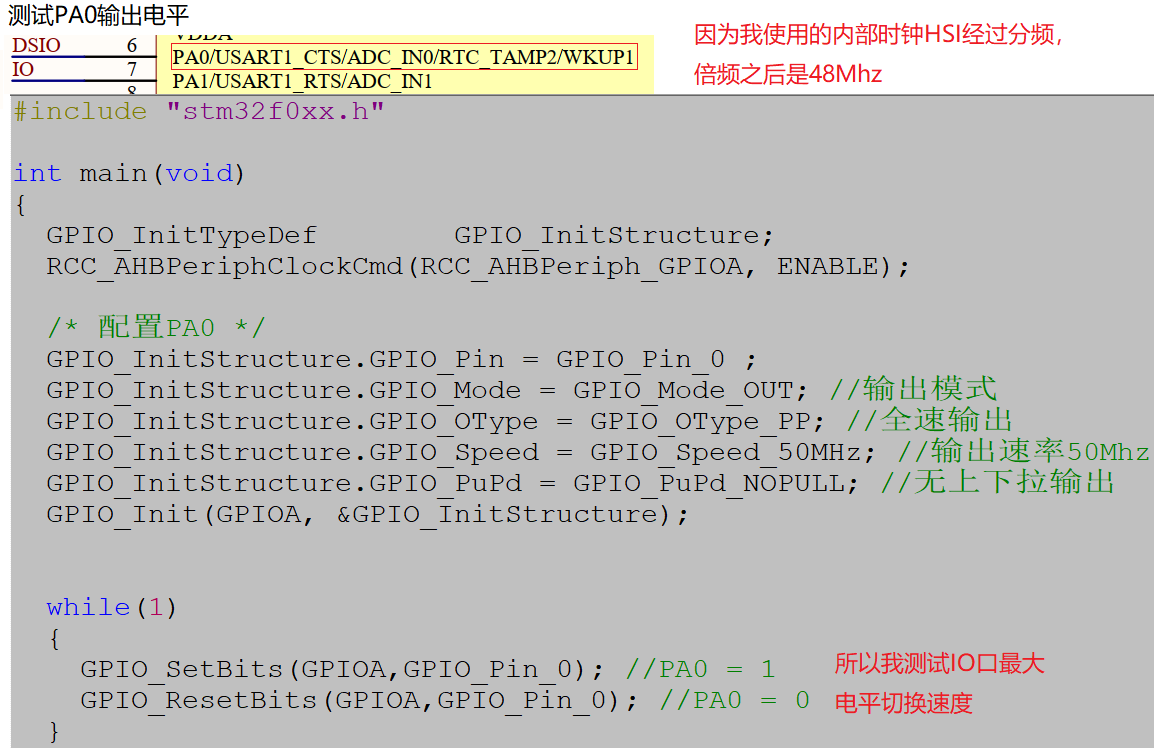
}

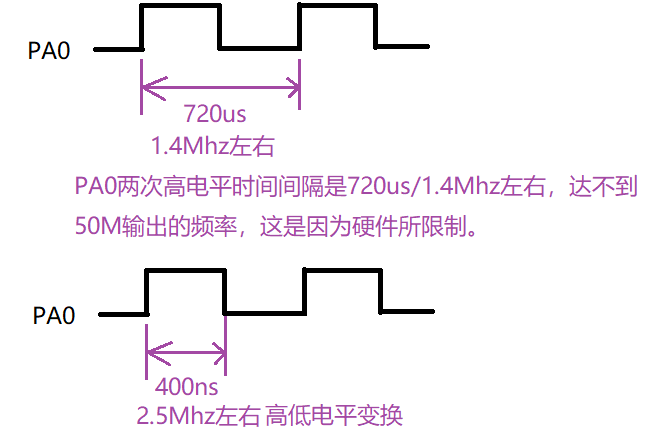


KEIL 配置成SWD模式，下载测试下。

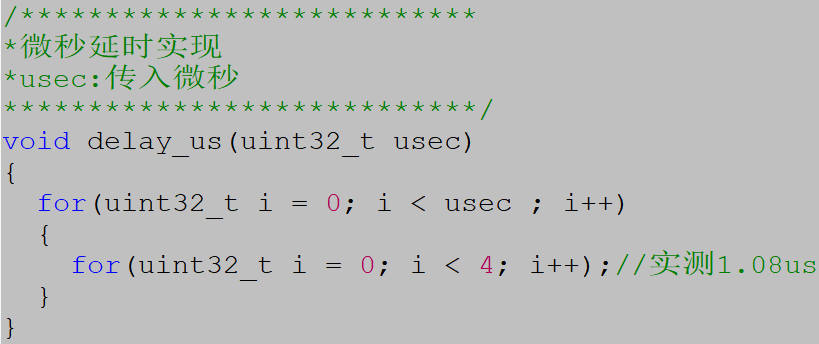
## GPIO输入输出功能

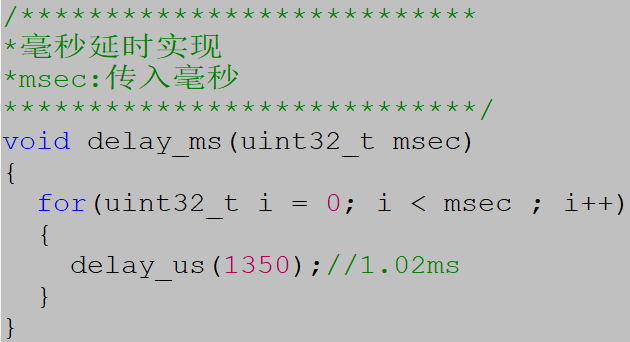
### GPIO输出功能





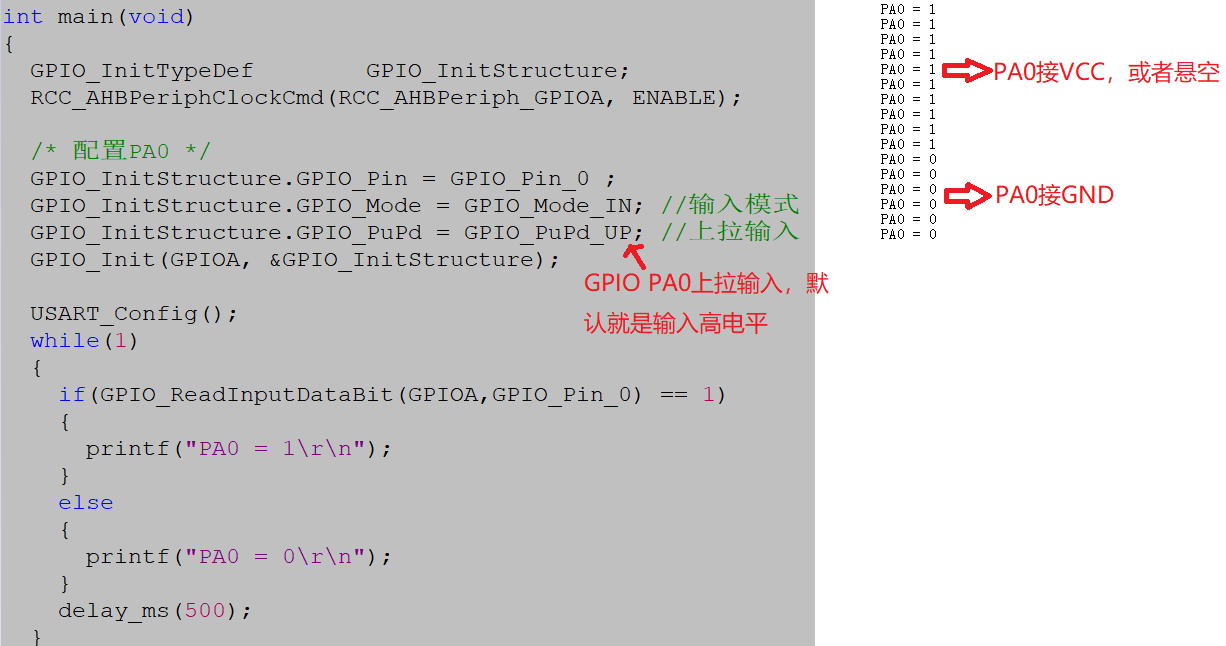
### 延时函数实现

内部晶振时钟实现的

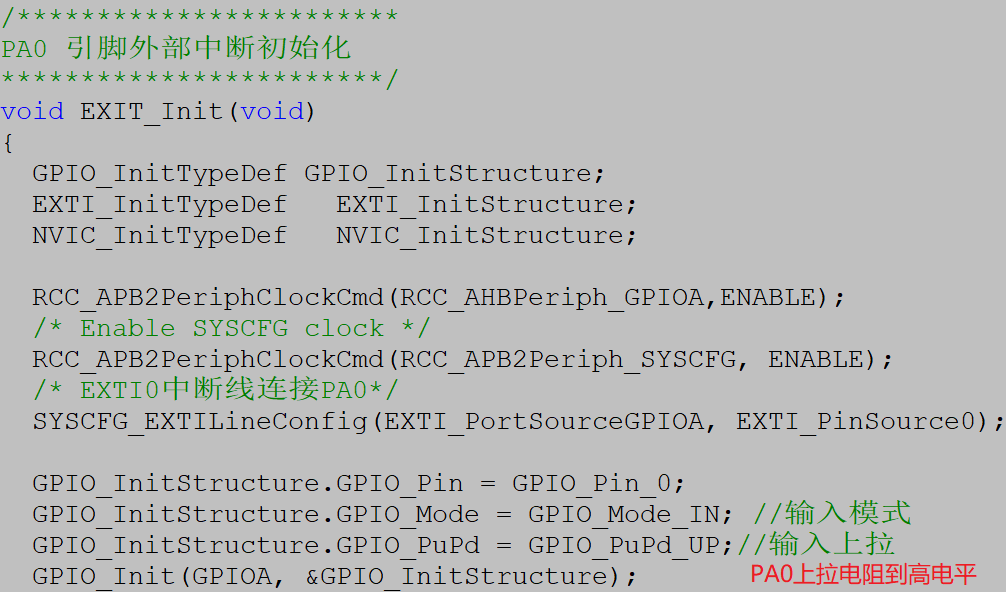


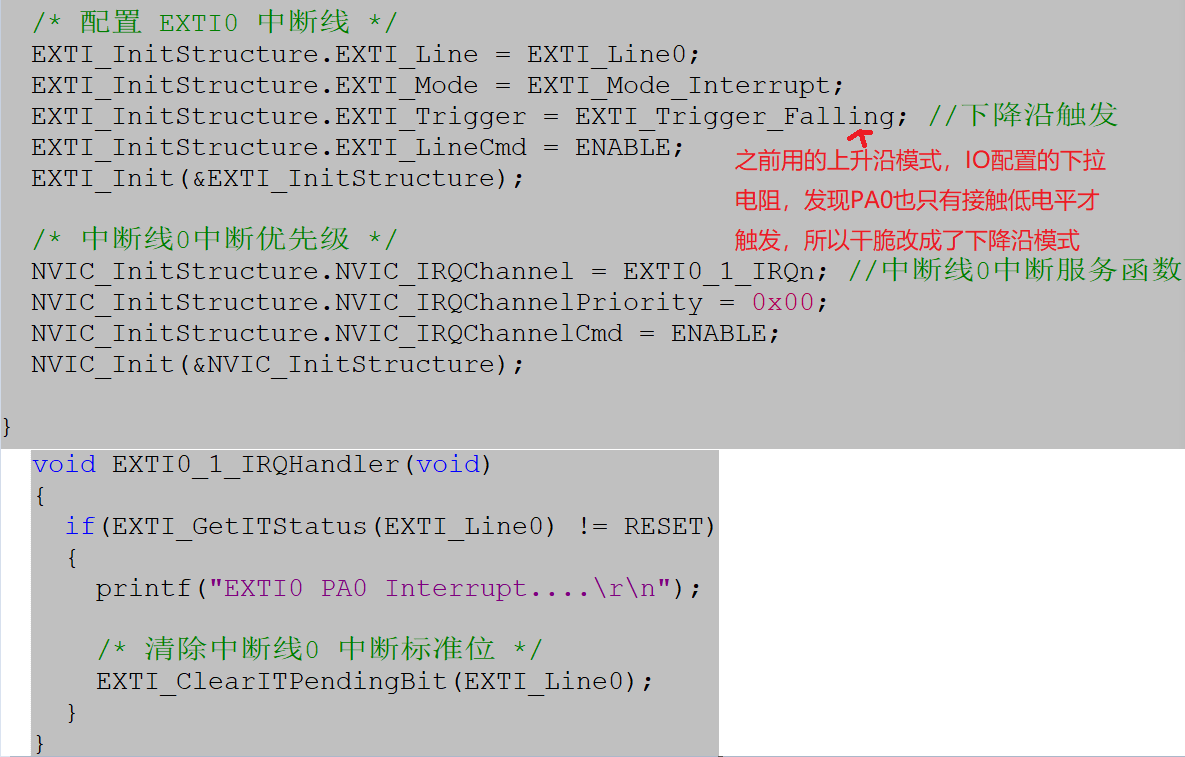
以上是48Mhz内部HSI时钟下实现的延时函数

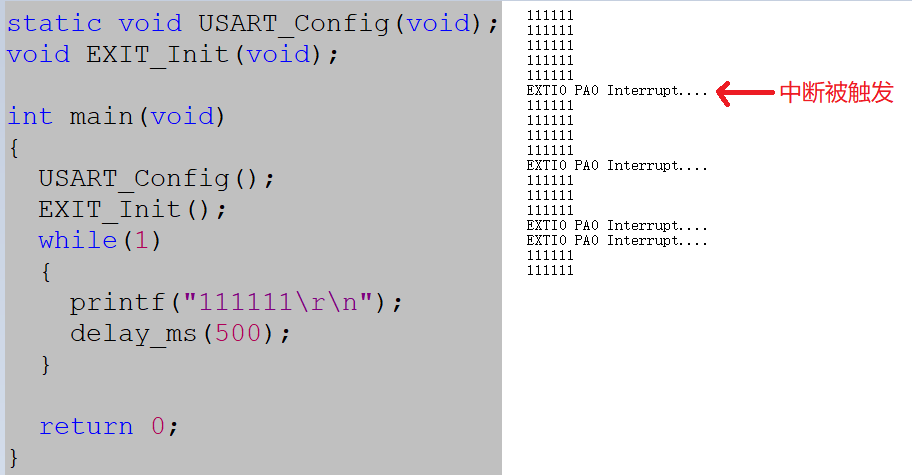
### GPIO输入功能



### GPIO外部中断触发功能

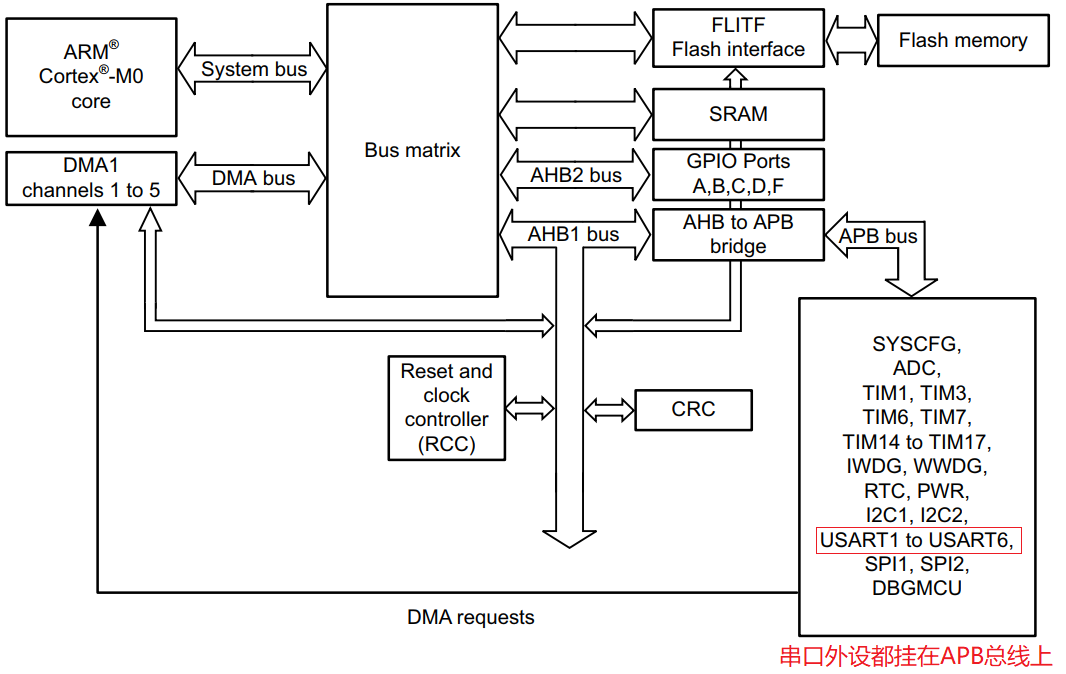


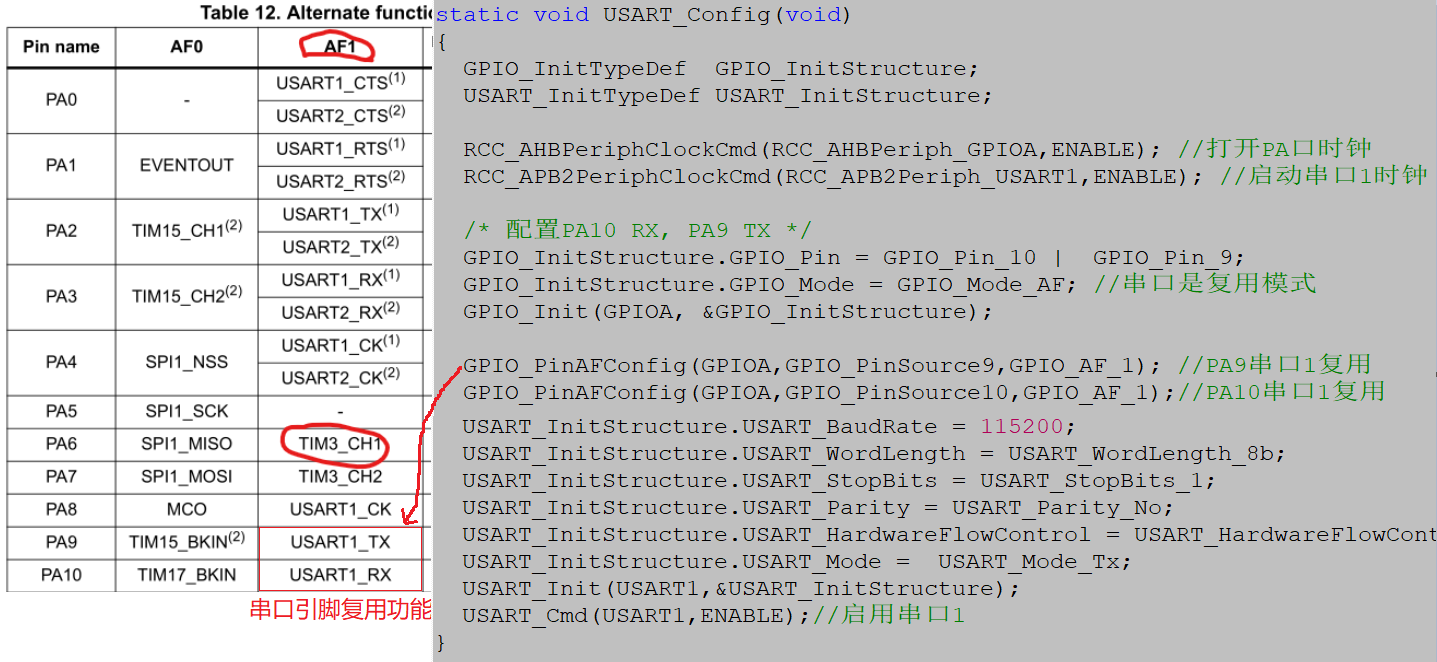


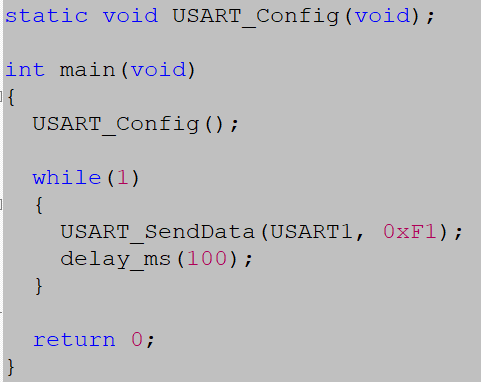
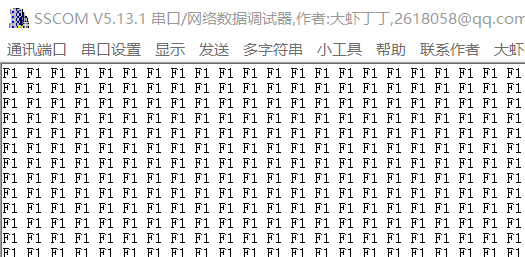


## 串口1发送接收实现

### 串口1发送

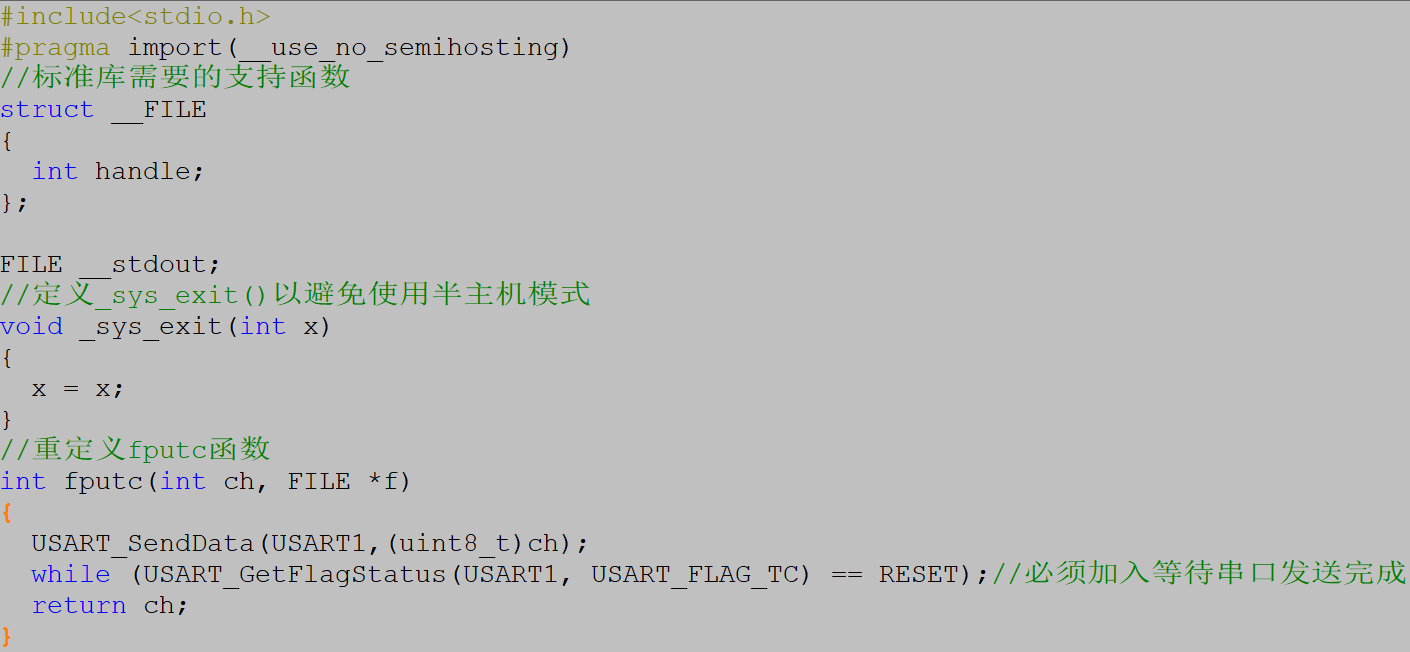


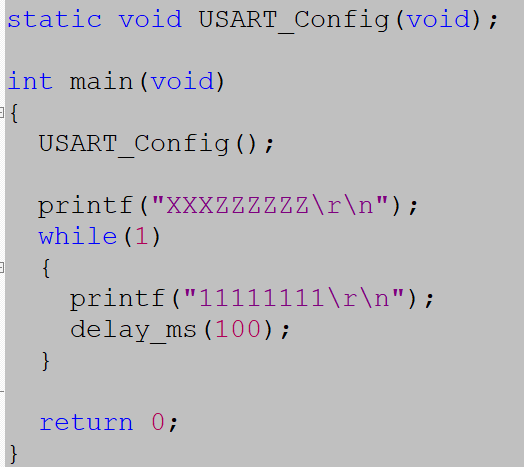
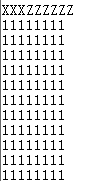


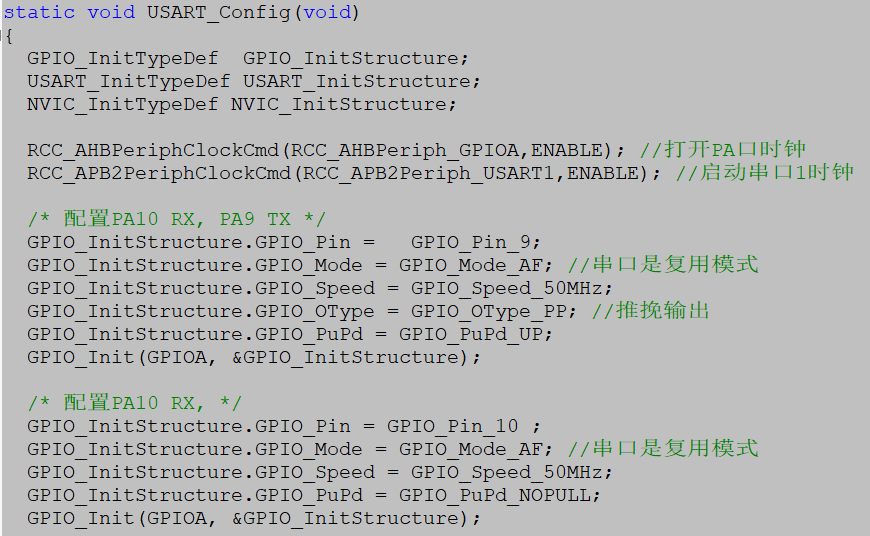
数据发送测试成功。

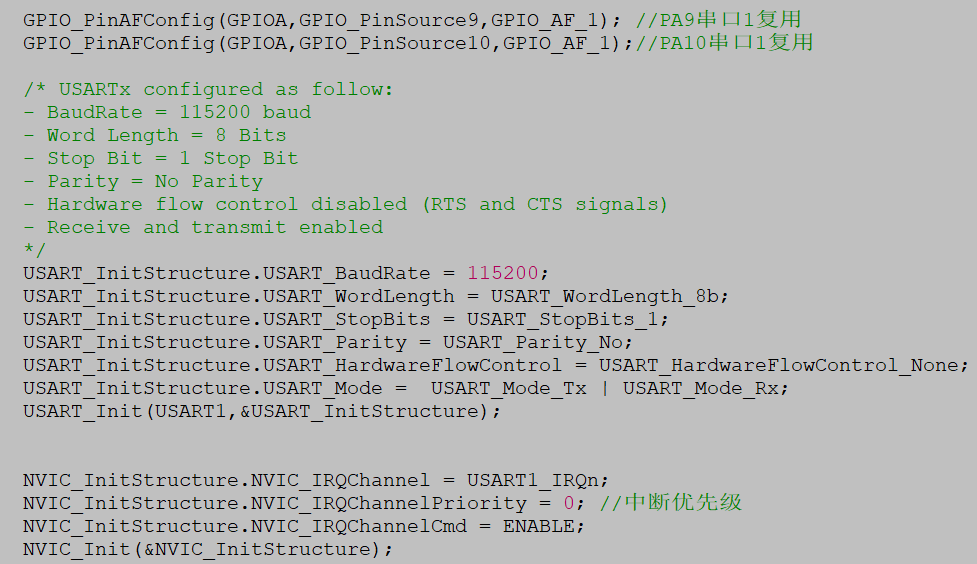
### 串口1 打印输出实现

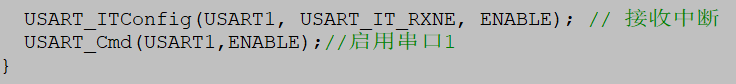


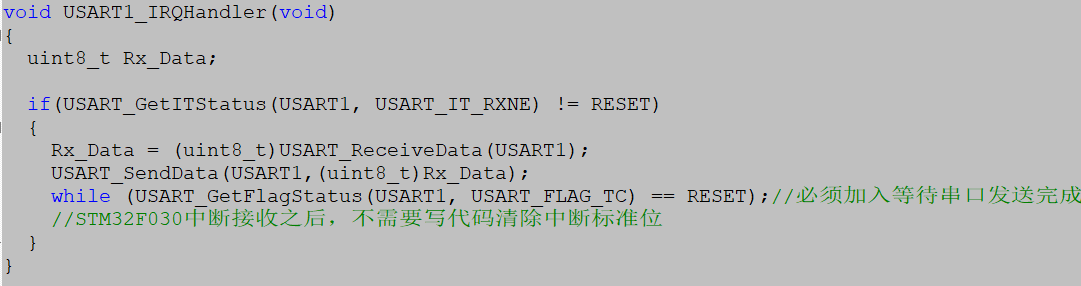
 实现完成

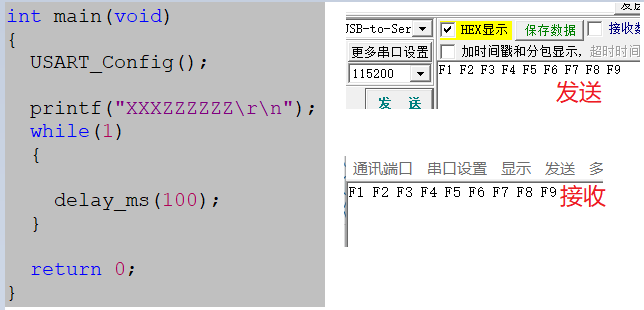
### 串口1 接收中断实现



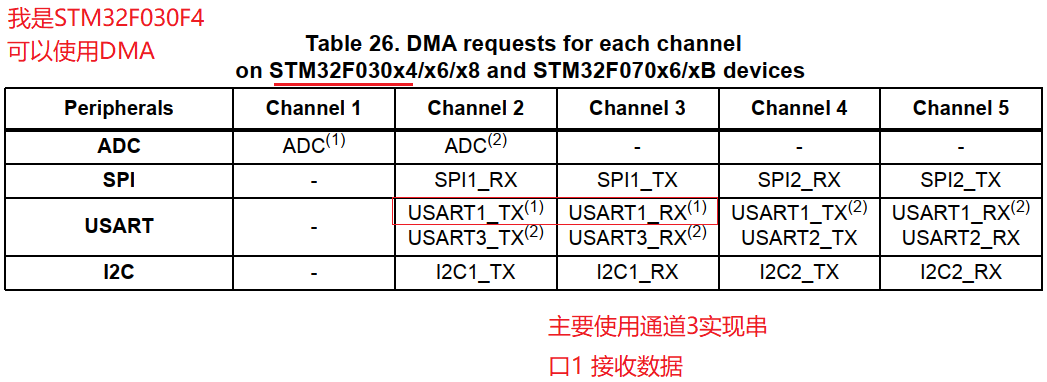


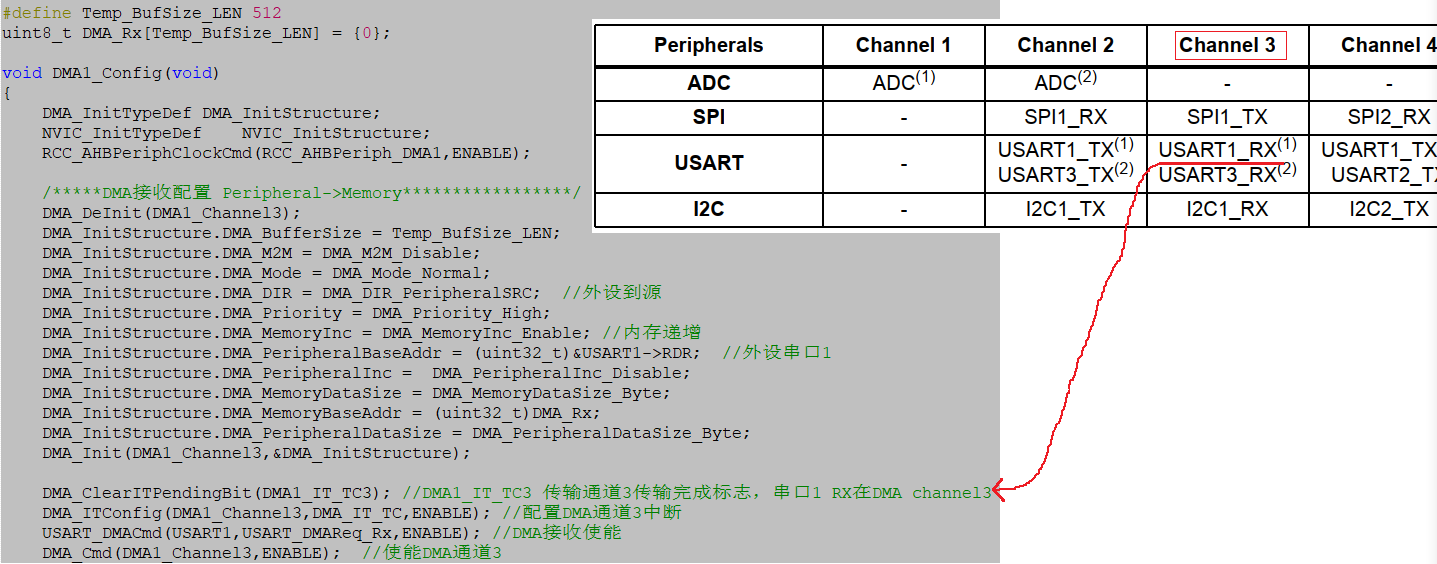


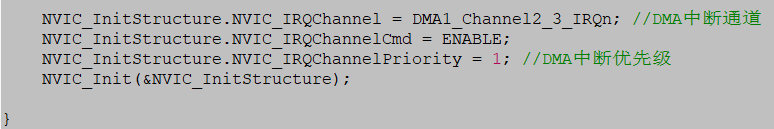


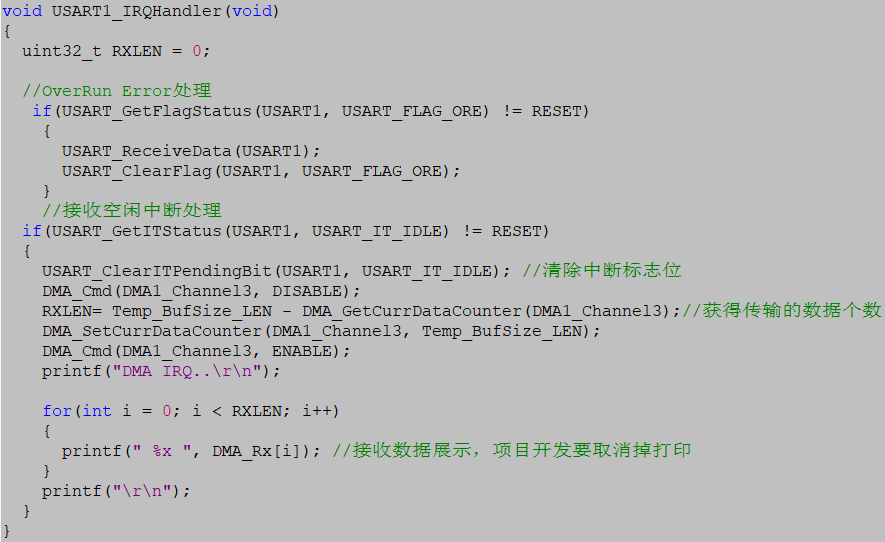


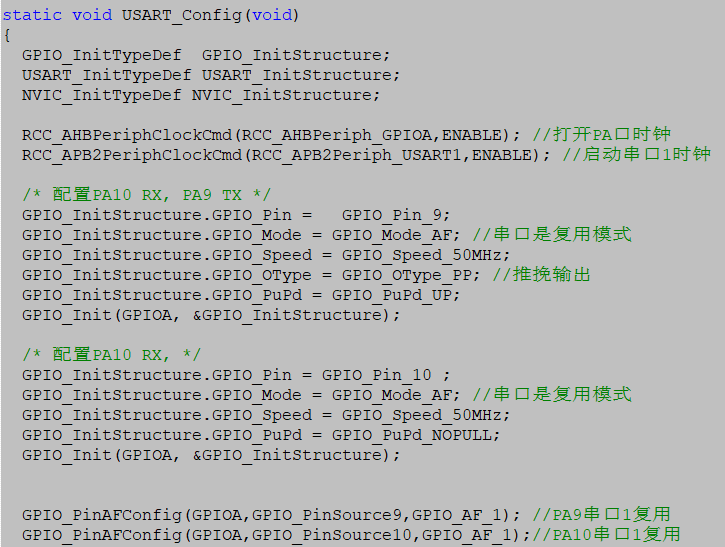
### 串口1 DMA空闲中断接收数据

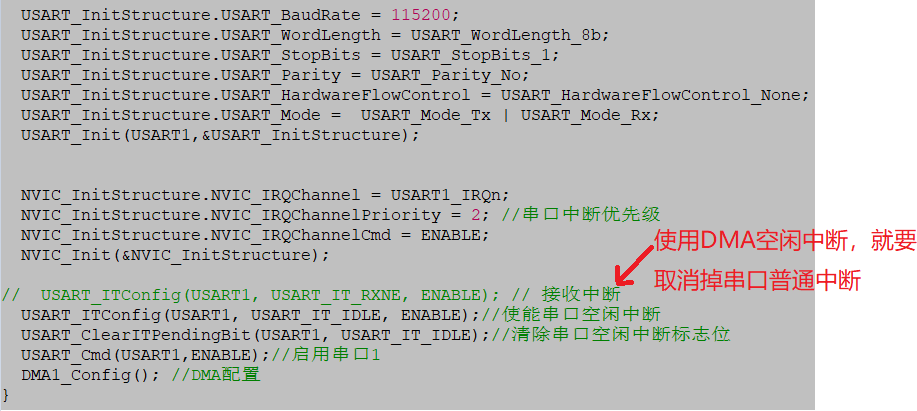


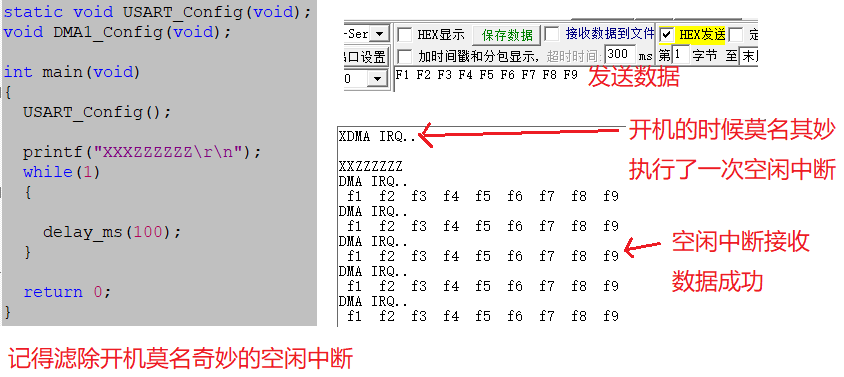












## 定时器TIME3使用

