# 补充配置RCC寄存器部分

RCC部分由许多寄存器组成，这里参考HIT\_HT的代码，来补充我的代码。RCC整体配置如下流图所示：

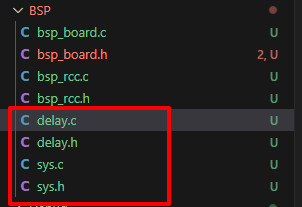


具体配置如下代码所示：

1. void BSP\_RCC\_Periph\_Clock\_Init(void)
2. {
3. *//ADC CLK Prescaler, ADC CLK = 72 / 6 = 12 Mhz*
4. RCC\_ADCCLKConfig(RCC\_PCLK2\_Div6);
5. *//USB OTG CLK FS Prescaler, when PLL = 72Mhz, OTGFSPRE = 0*
6. RCC\_USBCLKConfig(RCC\_USBCLKSource\_PLLCLK\_1Div5);
7. *//MCO*
8. RCC\_MCOConfig(RCC\_MCO\_NoClock);
9. *//Clear Clock interrupt register*
10. RCC->CIR = 0;
11. *//Reset APB2 and APB1 Peripheral*
12. RCC->APB2RSTR = 0;
13. RCC->APB1RSTR = 0;
14. *//AHB Peripheral CLK Enable*
15. RCC\_AHBPeriphClockCmd(RCC\_AHBPeriph\_SRAM,ENABLE);
16. RCC\_AHBPeriphClockCmd(RCC\_AHBPeriph\_FLITF,ENABLE);
17. *//APB2 Peripheral CLK Enable*
18. RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_AFIO,ENABLE);
19. RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA,ENABLE);
20. RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOB,ENABLE);
21. RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOC,ENABLE);
22. *//APB1 Peripheral CLK Enable*
23. RCC->APB1ENR = 0;
24. *//RTC CLK*
25. RCC\_LSEConfig(RCC\_LSE\_ON);
26. RCC\_RTCCLKConfig(RCC\_RTCCLKSource\_LSI);
27. RCC\_RTCCLKCmd(ENABLE);
28. }

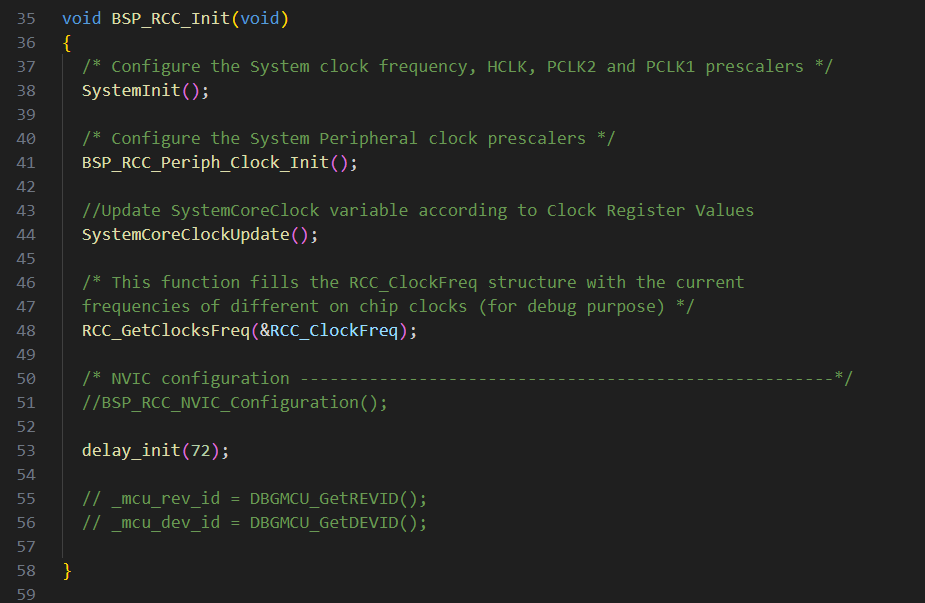
# 添加延时函数

添加文件到BSP中，如下图所示：



在“bsp\_rcc.h”中包含此头文件，添加 #include "delay.h"。

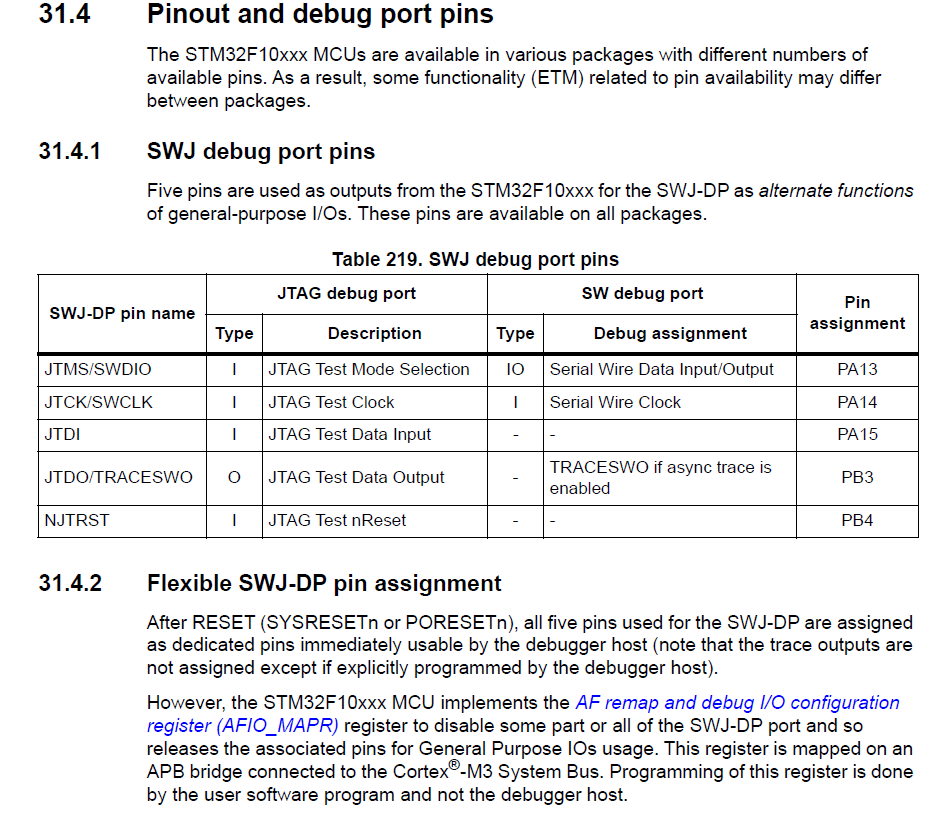
然后，在RCC初始化函数中初始化。代码如下图所示：



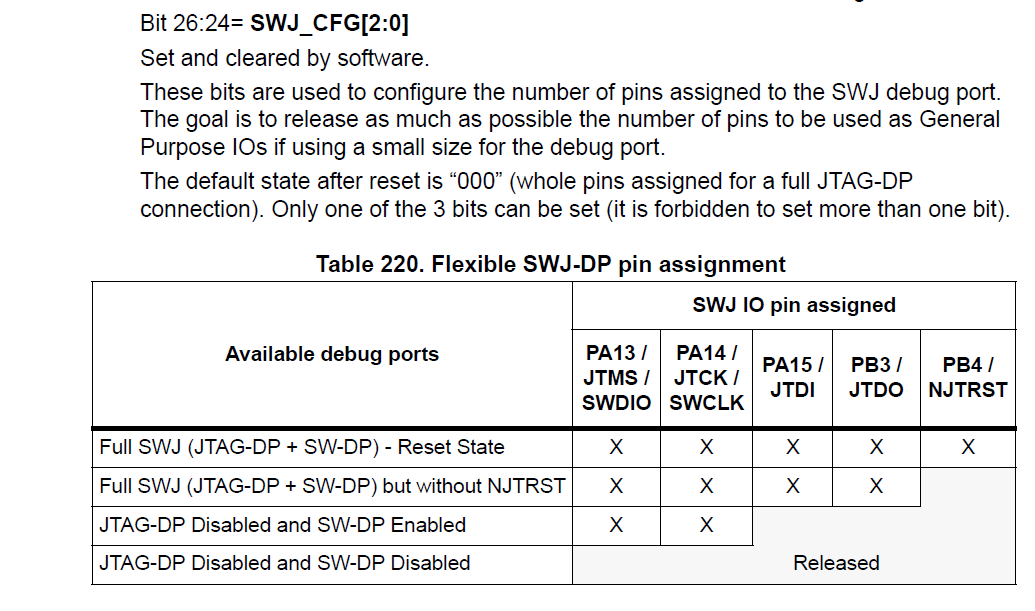
延时函数就配置好了，之后可以在函数中使用。

# 添加SWJ配置函数

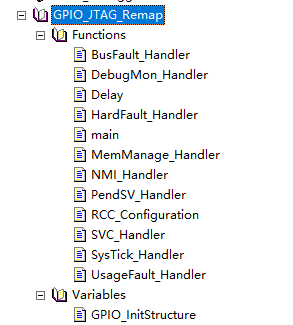
为了能够适应引脚资源紧张，不得不使用SWJ引脚时，需要重定义SWJ的引脚。在ST的官方手册RM0008中31.4节对此进行了描述。



通过修改SWJ\_CFG寄存器，可以对SWJ的引脚进行配置，使之成为通用IO。



参考标准外设库的示例，GPIO中的JTAG\_Remap工程。



我们来写自己的SWJ配置函数。

在“sys.c”中，我们编写以下函数：

1. */\*\**
2. \* @brief  Configures the SWJ Remap.
3. \* @param  SWJ\_REMAP: defines the SWJ Reamp Function.
4. \*   This parameter can be one of the following values:
5. \*     @arg SWJ\_ENABLE: Full SWJ Enable (JTAG-DP + SW-DP)
6. \*     @arg SWJ\_NONJTRST: Full SWJ Enabled (JTAG-DP + SW-DP) but without JTRST
7. \*     @arg SWJ\_NOJTAG: JTAG-DP Disabled and SW-DP Enabled
8. \*     @arg SWJ\_DISABLE: Full SWJ Disabled (JTAG-DP + SW-DP)
9. \* @retval None
10. \*/
11. void sys\_SWJ\_REMAP\_Config(uint8\_t SWJ\_REMAP)
12. {
13. switch (SWJ\_REMAP)
14. {
15. case SWJ\_ENABLE:
16. */\* Full SWJ Enable (JTAG-DP + SW-DP) \*/*
17. break;
18. case SWJ\_NONJTRST:
19. */\* Full SWJ Enabled (JTAG-DP + SW-DP) but without JTRST \*/*
20. GPIO\_PinRemapConfig(GPIO\_Remap\_SWJ\_NoJTRST, ENABLE);
21. break;
22. case SWJ\_NOJTAG:
23. */\* JTAG-DP Disabled and SW-DP Enabled \*/*
24. GPIO\_PinRemapConfig(GPIO\_Remap\_SWJ\_JTAGDisable, ENABLE);
25. break;
26. case SWJ\_DISABLE:
27. */\* Full SWJ Disabled (JTAG-DP + SW-DP) \*/*
28. GPIO\_PinRemapConfig(GPIO\_Remap\_SWJ\_Disable, ENABLE);
29. break;
30. default:
31. break;
32. }
33. }

# SWJ的配置使用

在主函数中，我们使用该配置，给出如下代码示例：

1. int main(void)
2. {
3. *// SYS CLK Init*
4. BSP\_RCC\_Init();
5. */\*\* Config SWJ Reamp*
6. \*   If using the PA15、PA14、PA13、PB4、PB3 for Remapping Pin, these Clock Must be Config before SWJ REMAP config Function!
7. \*/
8. sys\_SWJ\_REMAP\_Config(SWJ\_DISABLE);
9. GPIO\_InitTypeDef GPIO\_InitStructure;
10. */\* Configure PA.13 (JTMS/SWDAT), PA.14 (JTCK/SWCLK) and PA.15 (JTDI) as*
11. output push-pull \*/
12. GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_13 | GPIO\_Pin\_14 | GPIO\_Pin\_15;
13. GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;
14. GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_PP;
15. GPIO\_Init(GPIOA, &GPIO\_InitStructure);
16. */\* Configure PB.03 (JTDO) and PB.04 (JTRST) as output push-pull \*/*
17. GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_3 | GPIO\_Pin\_4;
18. GPIO\_Init(GPIOB, &GPIO\_InitStructure);
19. while (1)
20. {
21. */\* code \*/*
22. }
24. }

# 参考资料：

1. RM0008文档
2. STM32F10x\_StdPeriph\_Lib\_V3.6.0