# 嵌入式系统实验报告



|  |  |
| --- | --- |
| 实验名称： | 嵌入式 C 程序生命周期观察与分析 |
| 姓 名： | 王小龙 |
| 学 号： | 2020211502 |
| 学 院(系)： | 计算机学院 |
| 专 业： | 网络工程 |
| 指导教师： | 戴志涛、刘健培 |

2022年 11 月 29 日

# 实验目的

本实验从程序的生命周期角度，进一步分析嵌入式系统下一个程序从创建到运行的过程，以及该过程中涉及到的一些工具与概念。

# 实验环境

MDK开发平台

ST-Link 仿真器

RealView MDK5.23集成开发软件

PC机Window10 (64bit)

串口调试工具

# 实验要求

1.按照以下实验过程的说明阅读相关材料，按照步骤观察程序的编译、链接、 加载、运行过程，并练习使用 MDK 配置/控制该过程的方法。

2.完成实验步骤 2.3，将程序的启动与运行的流程图贴在作业答卷里。

# 实验原理

MDK 编译系统的主要流程有：

1. 编译，MDK 软件使用的编译器是 armcc 和 armasm，它们根据每个 c/c++和汇编源文件编译成对应的以".o"为后缀名的对象文件(Object Code，也称目标文件)，其内容主要是从源文件编译得到的机器码，包含了代码、数据以及调试使用的信息。

C源码的编译过程可细分为 3 个阶段：预处理、编译、汇编。

预处理（CPP）阶段根据以字符#开头的命令，修改原始 C 程序（文本格式）。结果是生成另一个 C 程序（文本格式）。

编译阶段编译预处理阶段产生的 C 程序，翻译为汇编程序（文本格式）。

汇编阶段将汇编程序翻译为机器语言指令，将指令和数据打包为以".o"为后缀名的目标文件（二进制格式）。

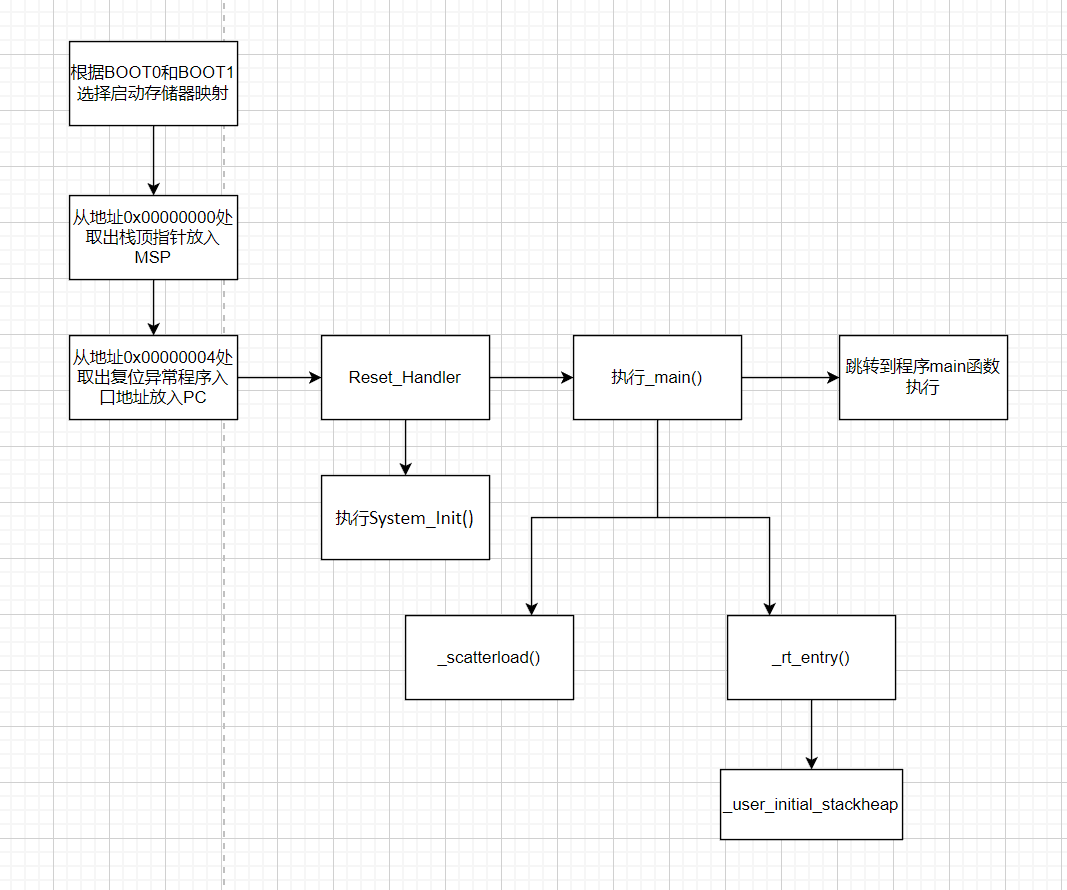
嵌入式系统因为目标系统资源限制，一般使用交叉编译的方式，即编译工具所在的处理器和编译目标的处理器不是同一类。

(2)链接，链接器 armlink 把各个.o 文件及库文件链接成一个可执行的映像文件，".axf"或".elf"后缀，二进制格式；

(3)格式转换，一般来说 Windows/Linux 系统使用链接器直接生成可执行映像文件 exe/elf 后，操作系统根据该文件的信息加载后，就可以运行程序了，但在嵌入式平台上，需要把该文件的内容加载到芯片上，所以还需要对链接器生成的 elf 映像文件利用格式转换器 fromelf 转换成".bin"或".hex"文件（将映像文件中的调试信息和注释等过滤掉），交给下载器下载到芯片的 FLASH 或 ROM 中，在上电后由处理器启动运行。

# 实验步骤

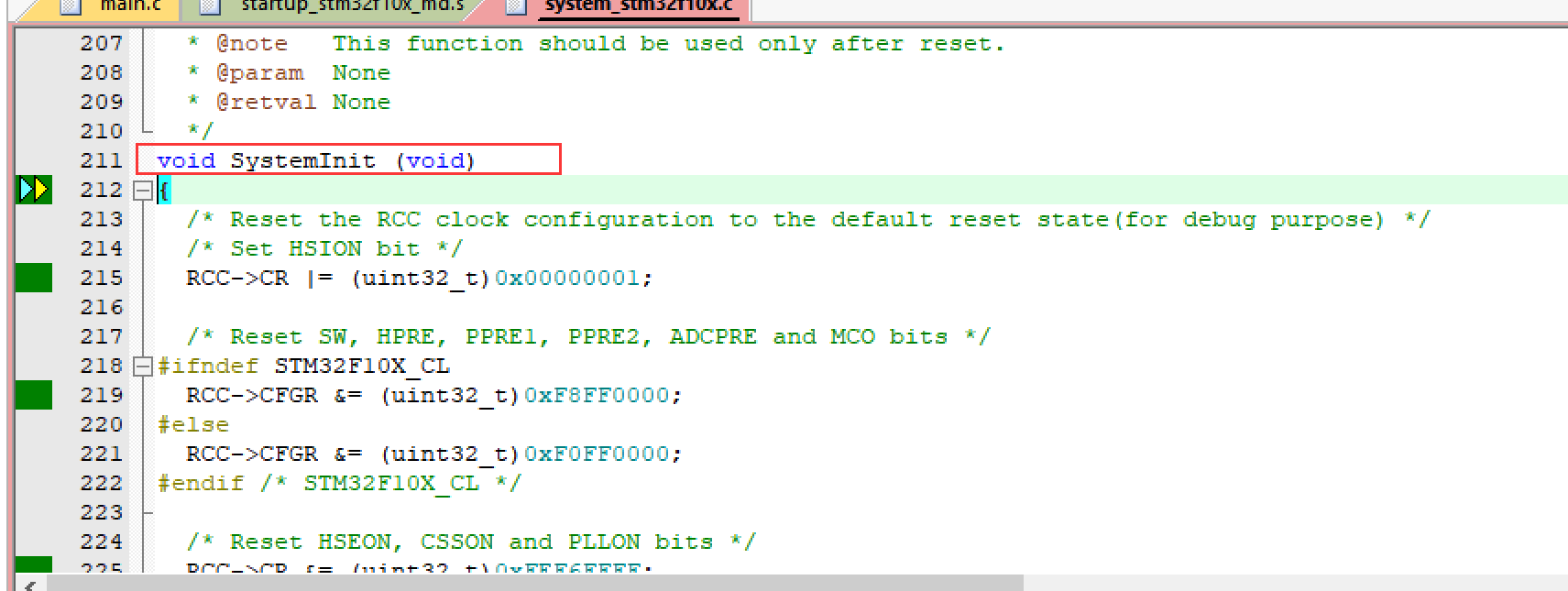
分析 lab2 的程序从处理器上电执行第一行指令到程序结束所做的工作流程图如下：



下面分析程序的启动：

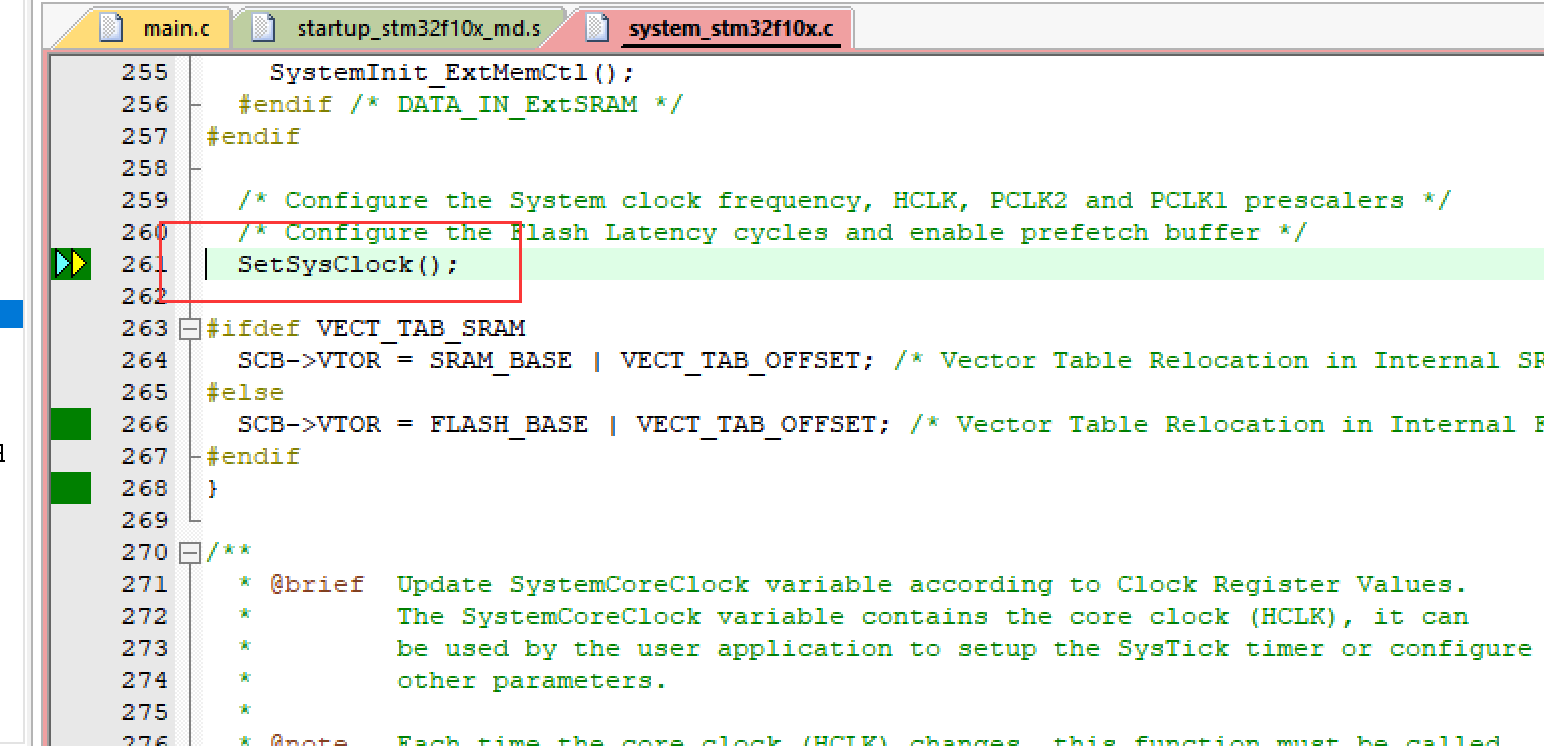
1.首先在SystemInit处加上断点，开始调试程序。

2.程序进入了SystemInit函数，开始初始化时间系统：

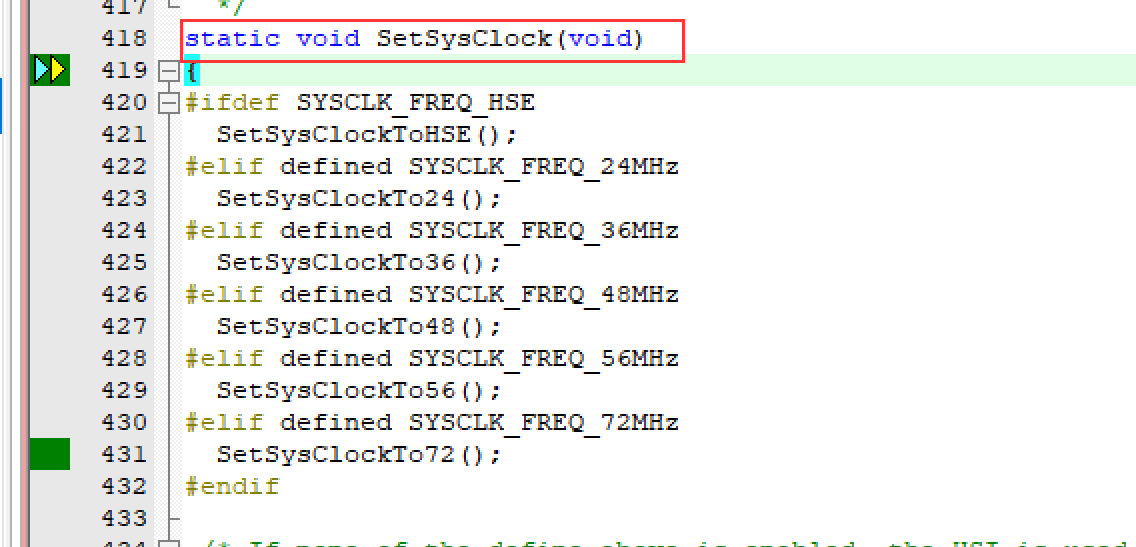


单步执行，可以看出其进行了一些初始化，

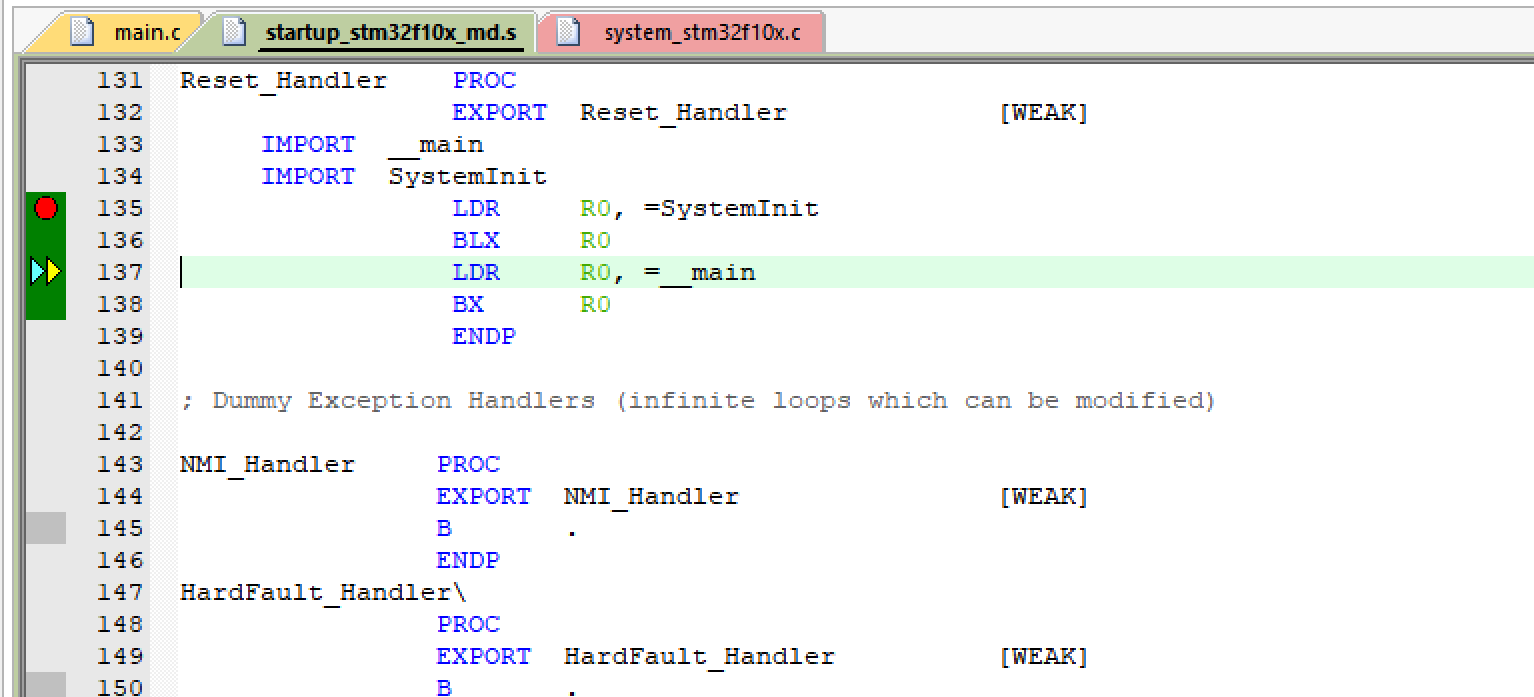
然后进入SetSysClock函数，初始化时钟系统：



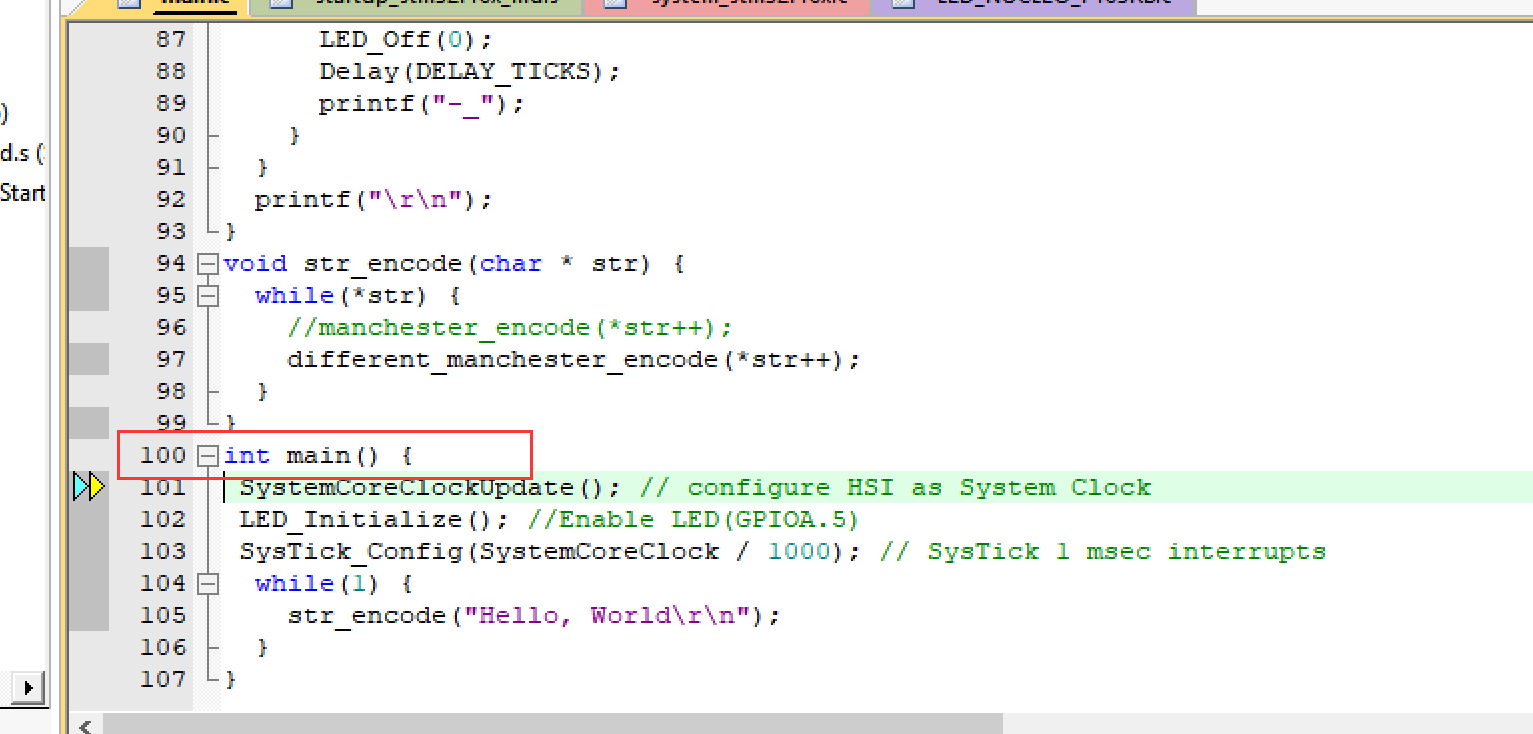
static void SetSysClock(void)函数如下:



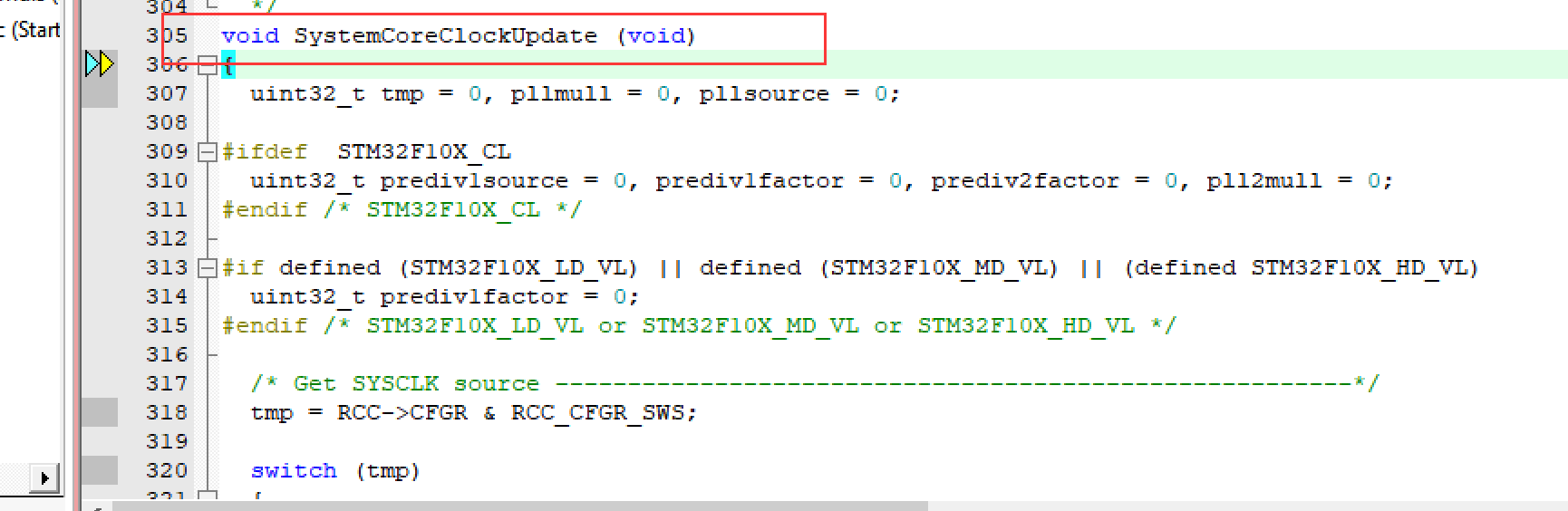
3.继续单步执行，程序会调用\_main()函数，完成应用程序运行环境的初始化:



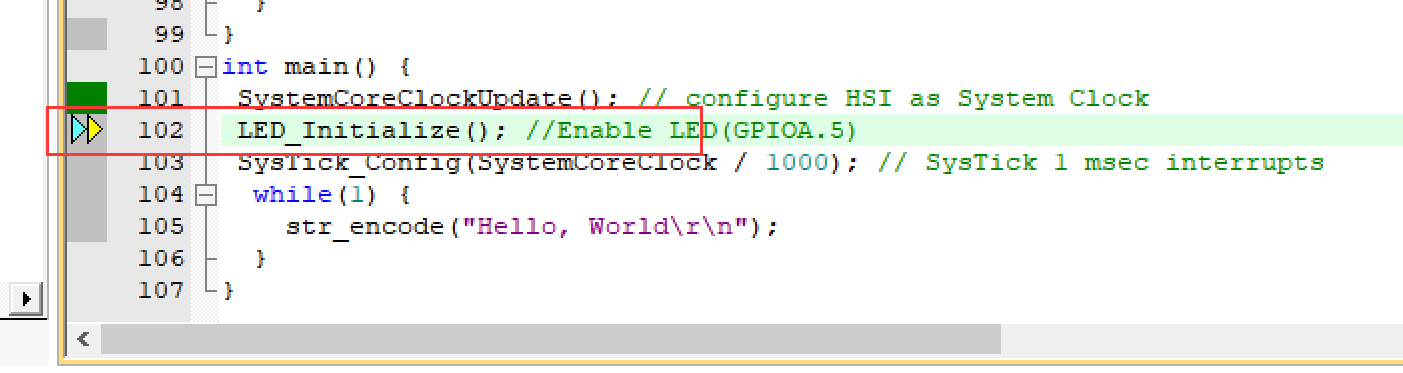
4.继续执行，跳转至main()函数

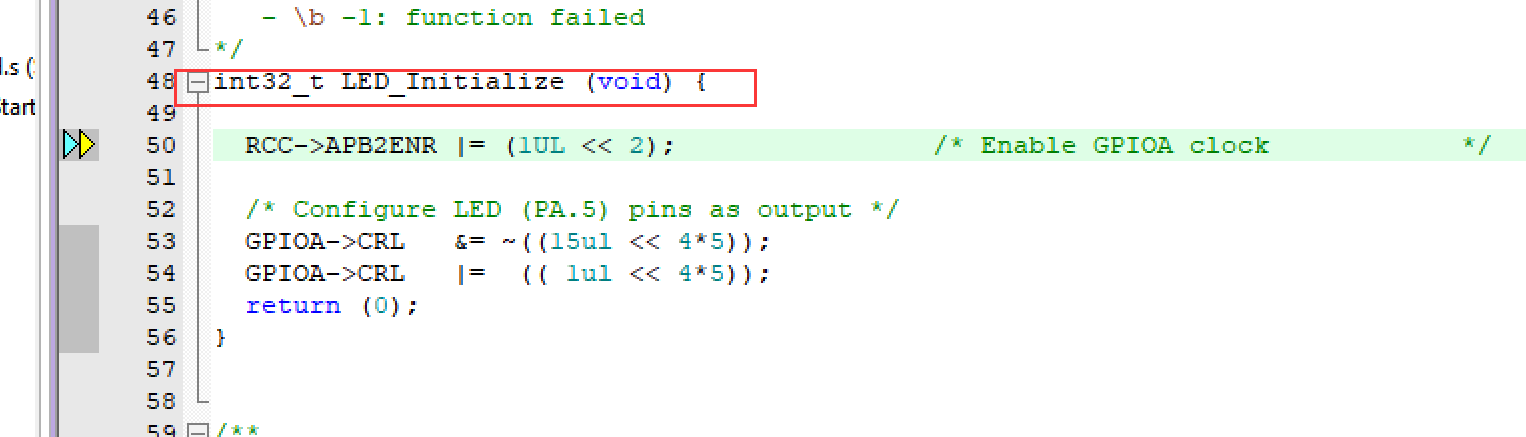


5.继续执行，会进入system\_stm32f10x.c执行里面的SystemCoreClockUpdate();函数，完成更新系统核时间

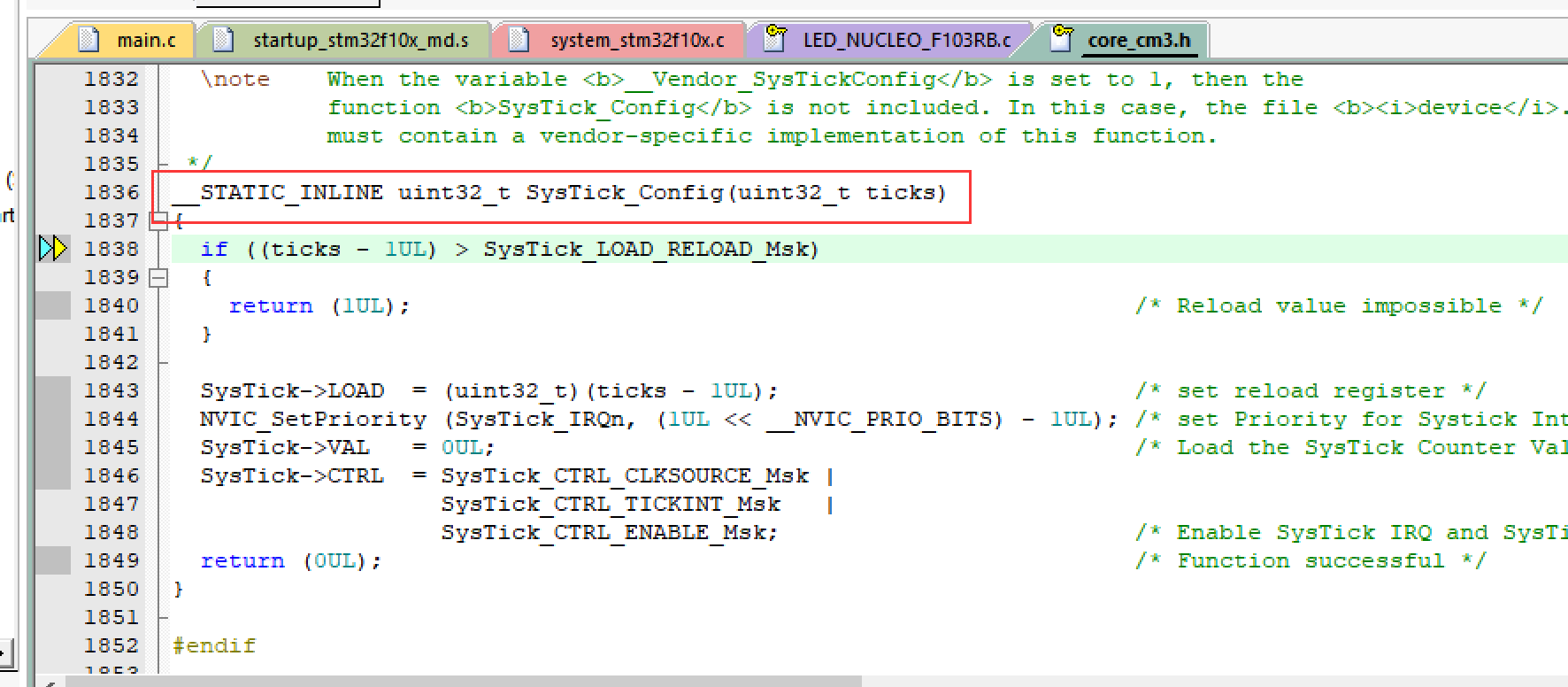


6.继续执行，会进入LED\_NUCLEO\_F103RB.c文件执行 LED\_Initialize();函数，完成初始化LED灯

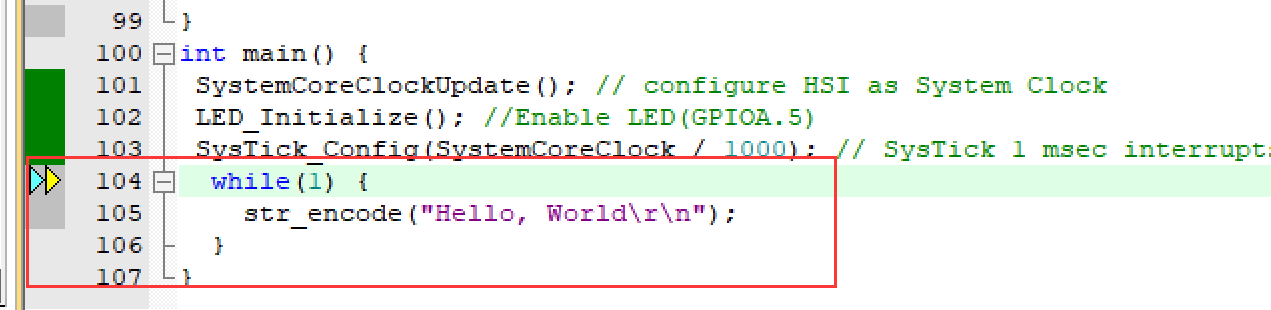




7.继续执行，会进入core\_cm3.h执行 SysTick\_Config(SystemCoreClock / 1000);函数：



8.继续执行，会进入while循坏执行我们自己编写的代码



**程序最后也不会结束，因为是while死循环。**

# 实验总结

通过本次实验，学习了嵌入式C程序生命周期观察与分析的流程，并对程序的编译、链接、加载、运行过程有了更深的理解，为后面的学习奠定了基础，收获颇多。