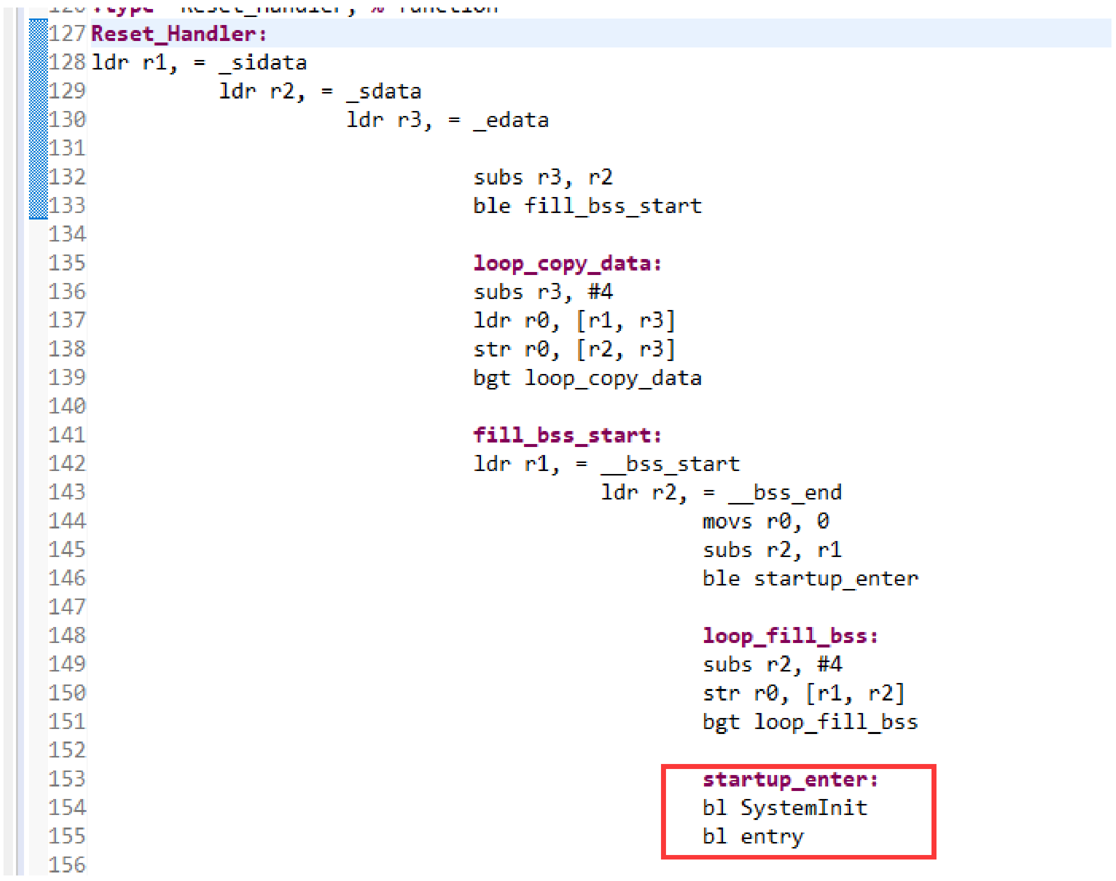
复位以后可以在 startup_apm32f10x_hd.s 中看到

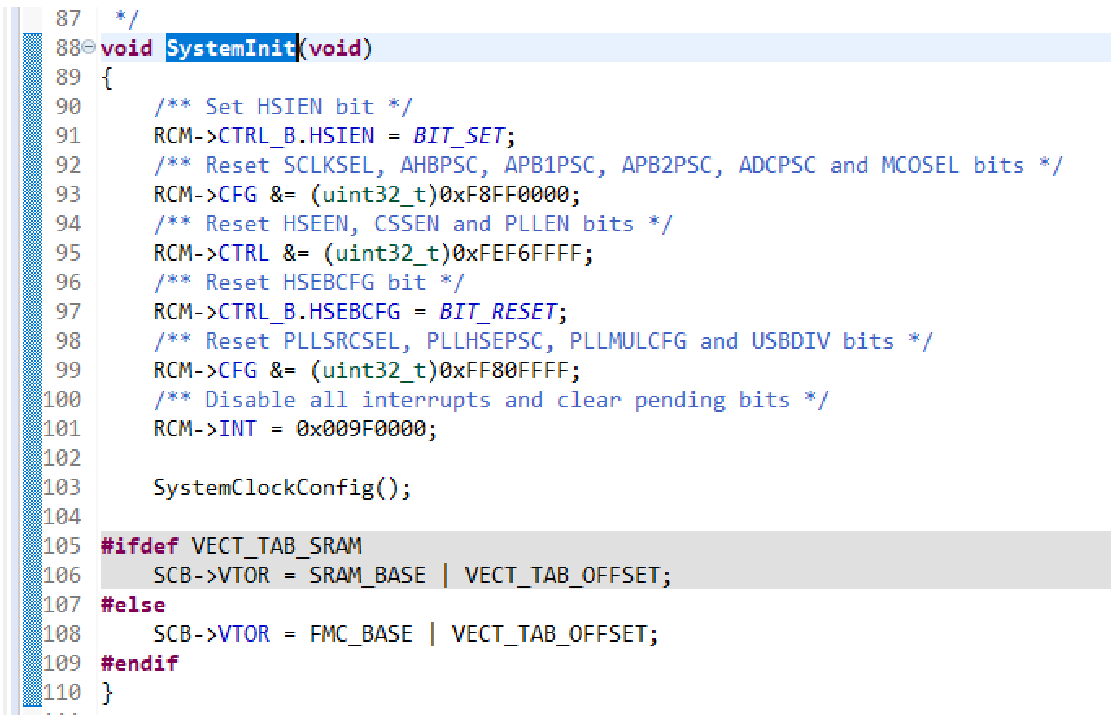


```
Reset_Handler:
ldr r1, = _sidata
          ldr r2, = _sdata
                    ldr r3, = _edata

                              subs r3, r2
                              ble fill_bss_start

                              loop_copy_data:
                              subs r3, #4
                              ldr r0, [r1, r3]
                              str r0, [r2, r3]
                              bgt loop_copy_data

                              fill_bss_start:
                              ldr r1, = __bss_start
                                        ldr r2, = __bss_end
                                                movs r0, 0
                                                subs r2, r1
                                                ble startup_enter

                                                loop_fill_bss:
                                                subs r2, #4
                                                str r0, [r1, r2]
                                                bgt loop_fill_bss

                                                startup_enter:
                                                bl SystemInit
                                                bl entry
```

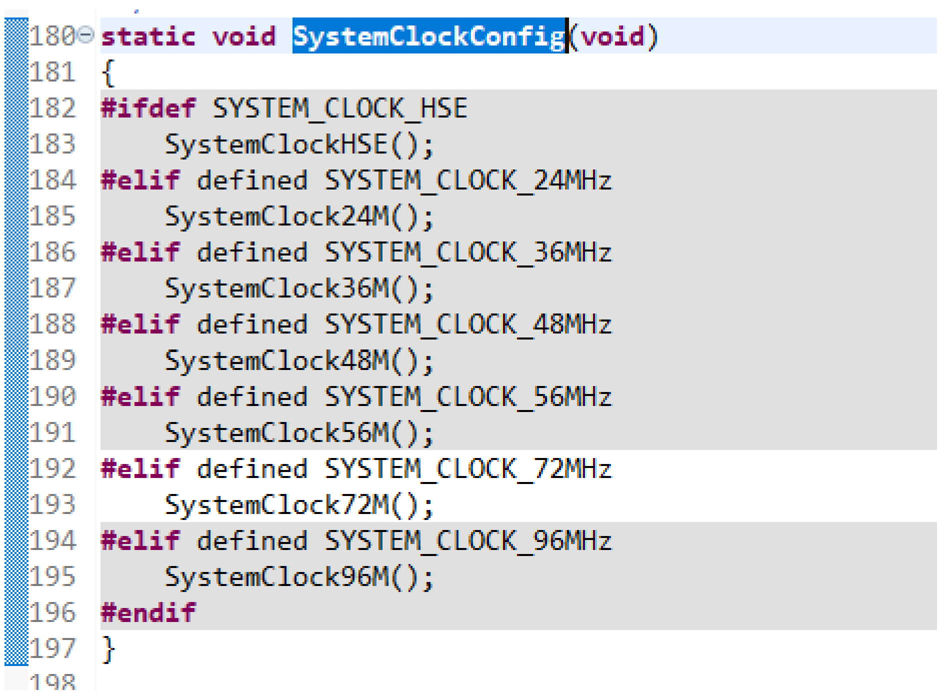
可以看见

初始化 Bss 段后

先执行系统时钟 72MHZ 再进入 **entry**

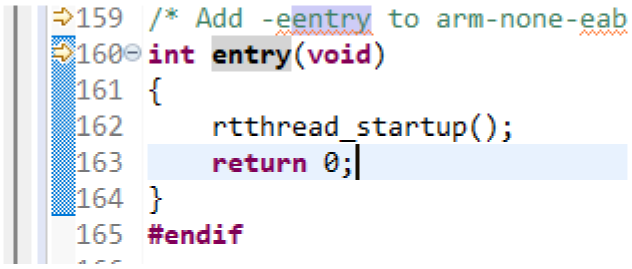


```
 */
void SystemInit(void)
{
    /** Set HSIEN bit */
    RCM->CTRL_B.HSIEN = BIT_SET;
    /** Reset SCLKSEL, AHBPSC, APB1PSC, APB2PSC, ADCPSC and MCOSEL bits */
    RCM->CFG &= (uint32_t)0xF8FF0000;
    /** Reset HSEEN, CSSEN and PLLEN bits */
    RCM->CTRL &= (uint32_t)0xFEF6FFFF;
    /** Reset HSEBCFG bit */
    RCM->CTRL_B.HSEBCFG = BIT_RESET;
    /** Reset PLLSRCSEL, PLLHSEPSC, PLLMULCFG and USBDIV bits */
    RCM->CFG &= (uint32_t)0xFF80FFFF;
    /** Disable all interrupts and clear pending bits */
    RCM->INT = 0x009F0000;

    SystemClockConfig();

#ifdef VECT_TAB_SRAM
    SCB->VTOR = SRAM_BASE | VECT_TAB_OFFSET;
#else
    SCB->VTOR = FMC_BASE | VECT_TAB_OFFSET;
#endif
}
```

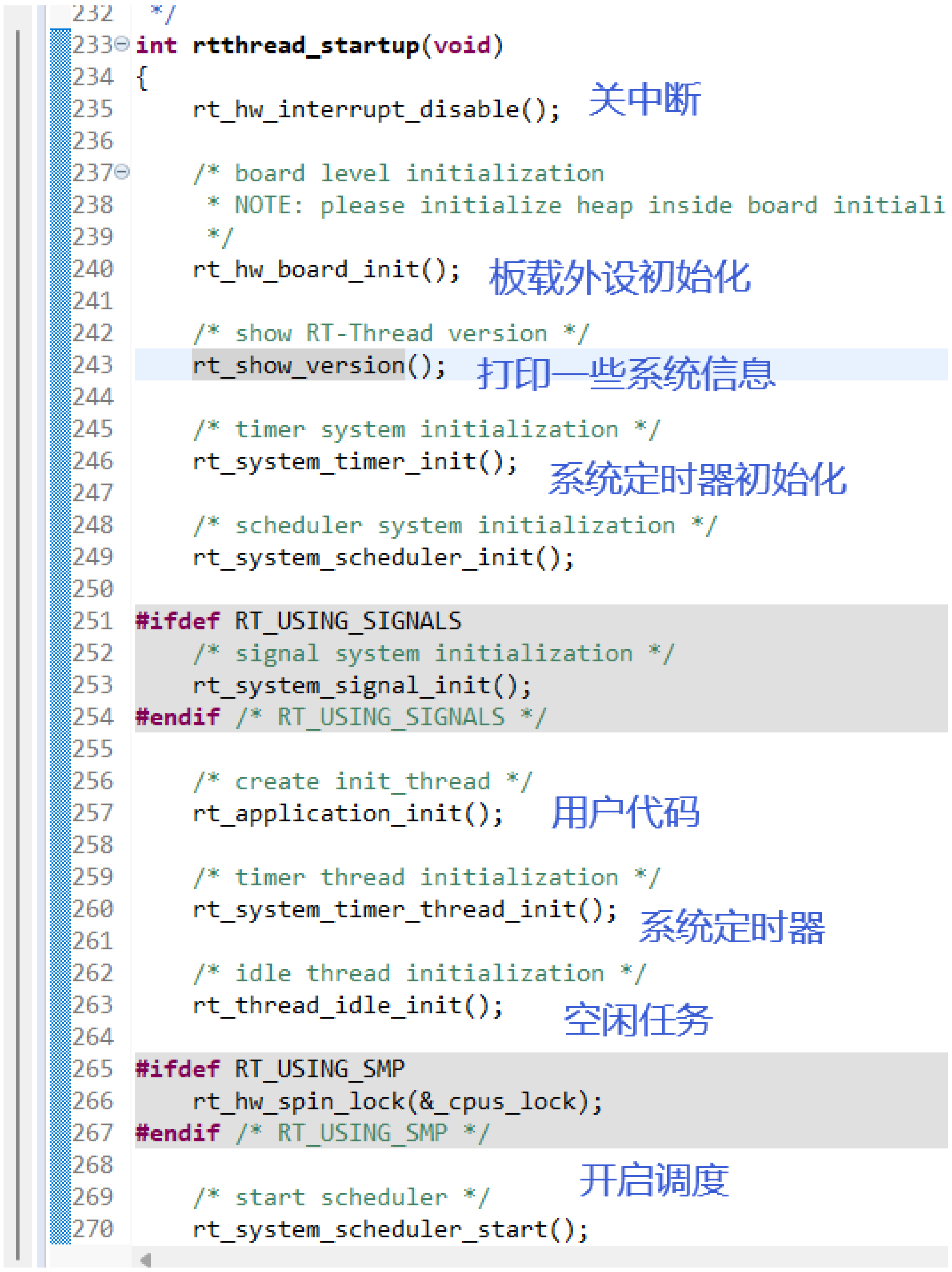


```
static void SystemClockConfig(void)
{
#ifdef SYSTEM_CLOCK_HSE
    SystemClockHSE();
#elif defined SYSTEM_CLOCK_24MHz
    SystemClock24M();
#elif defined SYSTEM_CLOCK_36MHz
    SystemClock36M();
#elif defined SYSTEM_CLOCK_48MHz
    SystemClock48M();
#elif defined SYSTEM_CLOCK_56MHz
    SystemClock56M();
#elif defined SYSTEM_CLOCK_72MHz
    SystemClock72M();
#elif defined SYSTEM_CLOCK_96MHz
    SystemClock96M();
#endif
}
```

进入以后，调用 rtthread_startup（）函数

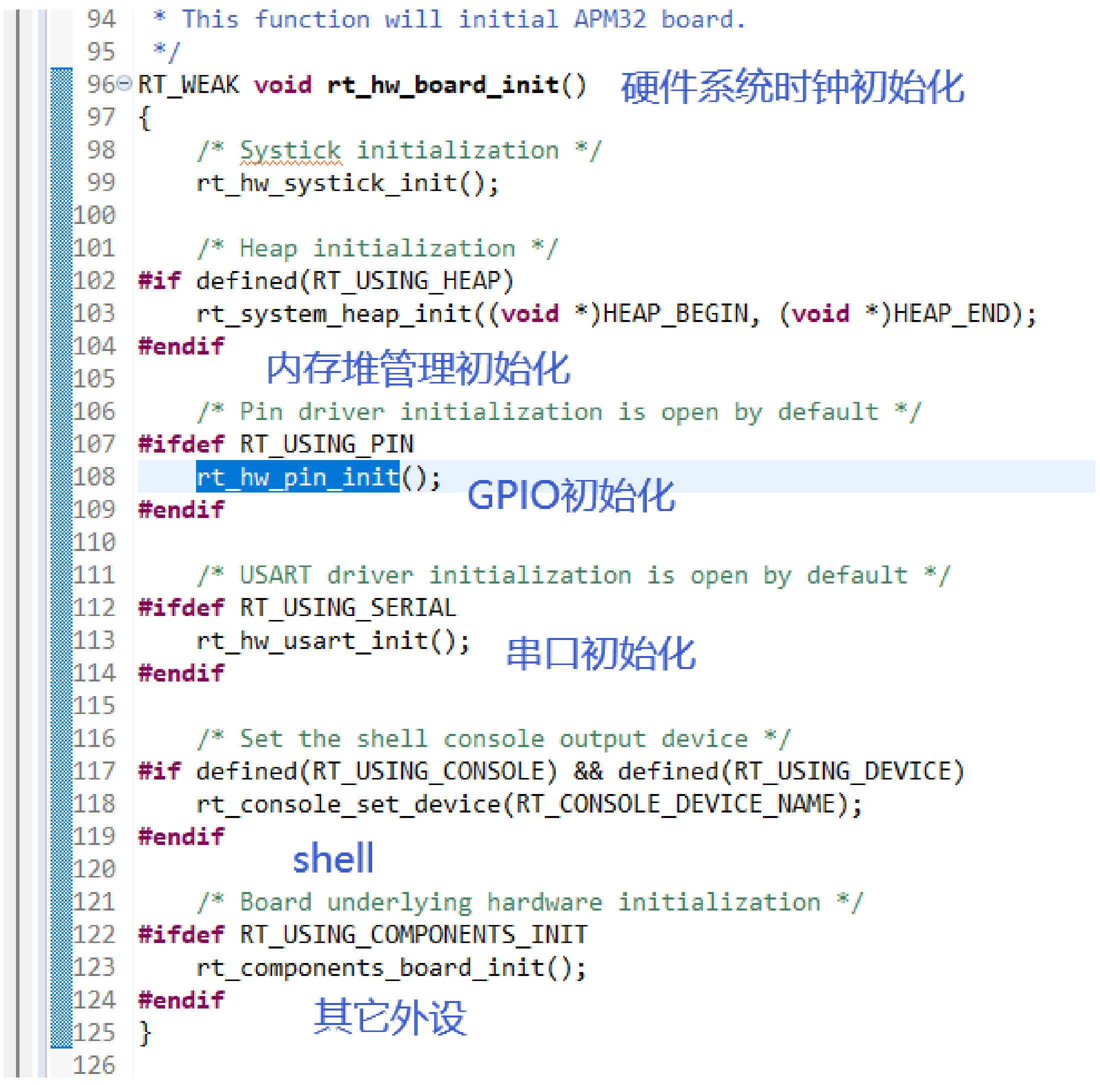


```
/* Add -eentry to arm-none-eab
int entry(void)
{
    rtthread_startup();
    return 0;
}
#endif
```



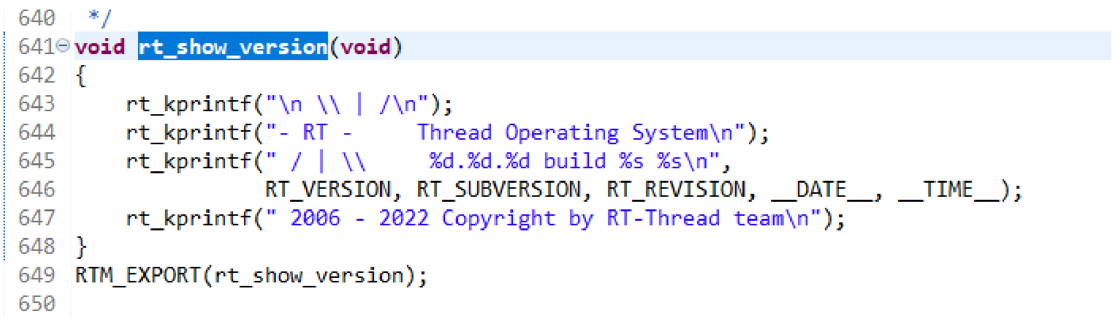
```c
 */
int rtthread_startup(void)
{
    rt_hw_interrupt_disable();

    /* board level initialization
     * NOTE: please initialize heap inside board initiali
     */
    rt_hw_board_init();

    /* show RT-Thread version */
    rt_show_version();

    /* timer system initialization */
    rt_system_timer_init();

    /* scheduler system initialization */
    rt_system_scheduler_init();

#ifdef RT_USING_SIGNALS
    /* signal system initialization */
    rt_system_signal_init();
#endif /* RT_USING_SIGNALS */

    /* create init_thread */
    rt_application_init();

    /* timer thread initialization */
    rt_system_timer_thread_init();

    /* idle thread initialization */
    rt_thread_idle_init();

#ifdef RT_USING_SMP
    rt_hw_spin_lock(&_cpus_lock);
#endif /* RT_USING_SMP */

    /* start scheduler */
    rt_system_scheduler_start();
```

1、其中板载外设初始化包括



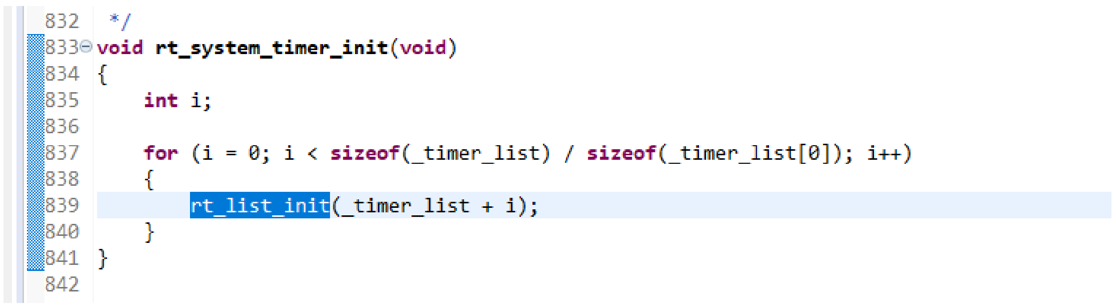
```c
 * This function will initial APM32 board.
 */
RT_WEAK void rt_hw_board_init()
{
    /* Systick initialization */
    rt_hw_systick_init();

    /* Heap initialization */
#if defined(RT_USING_HEAP)
    rt_system_heap_init((void *)HEAP_BEGIN, (void *)HEAP_END);
#endif

    /* Pin driver initialization is open by default */
#ifdef RT_USING_PIN
    rt_hw_pin_init();
#endif

    /* USART driver initialization is open by default */
#ifdef RT_USING_SERIAL
    rt_hw_usart_init();
#endif

    /* Set the shell console output device */
#if defined(RT_USING_CONSOLE) && defined(RT_USING_DEVICE)
    rt_console_set_device(RT_CONSOLE_DEVICE_NAME);
#endif

    /* Board underlying hardware initialization */
#ifdef RT_USING_COMPONENTS_INIT
    rt_components_board_init();
#endif
}
```

2、**rt_show_version 打印一些信息**



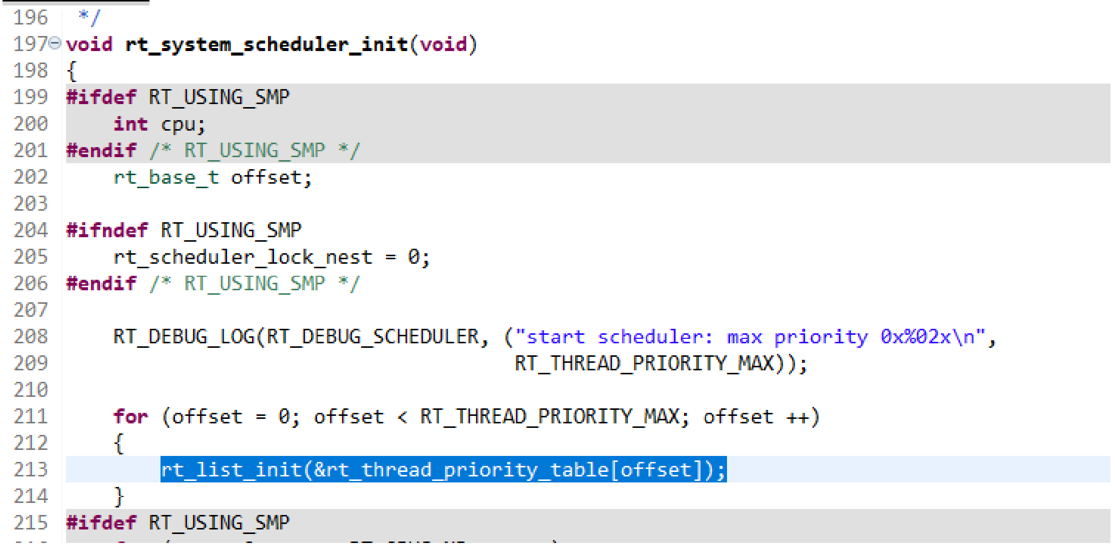
```c
 */
void rt_show_version(void)
{
    rt_kprintf("\n \\ | /\n");
    rt_kprintf("- RT -     Thread Operating System\n");
    rt_kprintf(" / | \\     %d.%d.%d build %s %s\n",
               RT_VERSION, RT_SUBVERSION, RT_REVISION, __DATE__, __TIME__);
    rt_kprintf(" 2006 - 2022 Copyright by RT-Thread team\n");
}
RTM_EXPORT(rt_show_version);
```

3、**rt_system_timer_init 链表初始化**



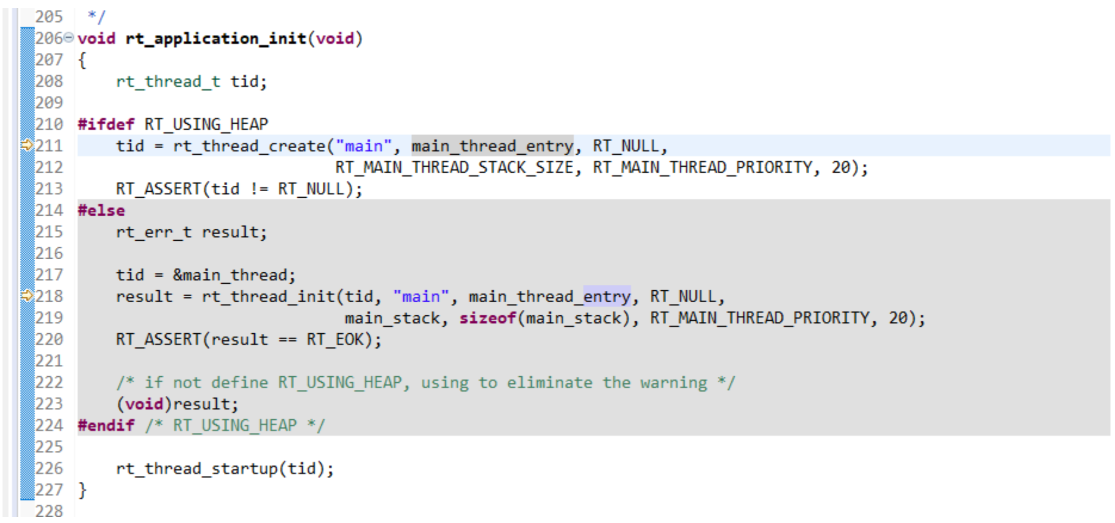
```c
 */
void rt_system_timer_init(void)
{
    int i;

    for (i = 0; i < sizeof(_timer_list) / sizeof(_timer_list[0]); i++)
    {
        rt_list_init(_timer_list + i);
    }
}
```

4、rt_system_scheduler_init 系统调度初始化，主要是排列线程优先级，放入线程优先级链表



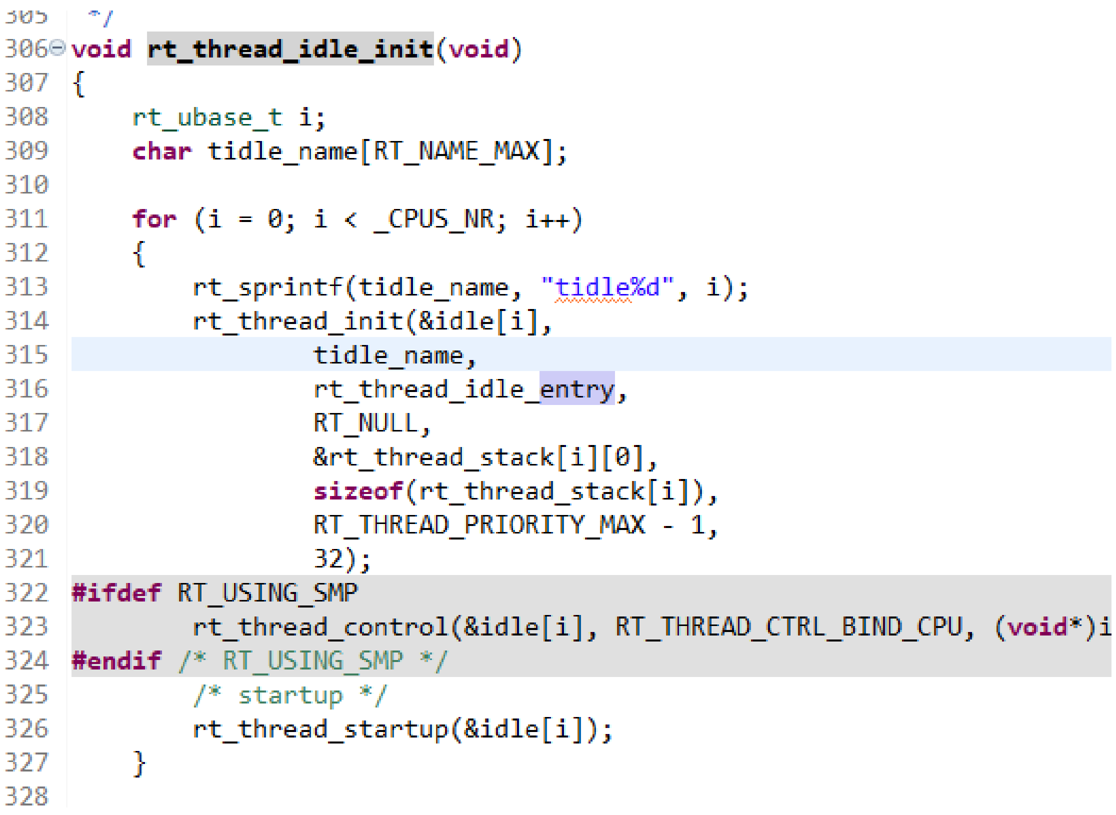
```
 */
void rt_system_scheduler_init(void)
{
#ifdef RT_USING_SMP
    int cpu;
#endif /* RT_USING_SMP */
    rt_base_t offset;

#ifndef RT_USING_SMP
    rt_scheduler_lock_nest = 0;
#endif /* RT_USING_SMP */

    RT_DEBUG_LOG(RT_DEBUG_SCHEDULER, ("start scheduler: max priority 0x%02x\n",
                                      RT_THREAD_PRIORITY_MAX));

    for (offset = 0; offset < RT_THREAD_PRIORITY_MAX; offset ++)
    {
        rt_list_init(&rt_thread_priority_table[offset]);
    }
#ifdef RT_USING_SMP
```

5、**rt_application_init 创建主线程**



```
 */
void rt_application_init(void)
{
    rt_thread_t tid;

#ifdef RT_USING_HEAP
    tid = rt_thread_create("main", main_thread_entry, RT_NULL,
                           RT_MAIN_THREAD_STACK_SIZE, RT_MAIN_THREAD_PRIORITY, 20);
    RT_ASSERT(tid != RT_NULL);
#else
    rt_err_t result;

    tid = &main_thread;
    result = rt_thread_init(tid, "main", main_thread_entry, RT_NULL,
                            main_stack, sizeof(main_stack), RT_MAIN_THREAD_PRIORITY, 20);
    RT_ASSERT(result == RT_EOK);

    /* if not define RT_USING_HEAP, using to eliminate the warning */
    (void)result;
#endif /* RT_USING_HEAP */

    rt_thread_startup(tid);
}
```

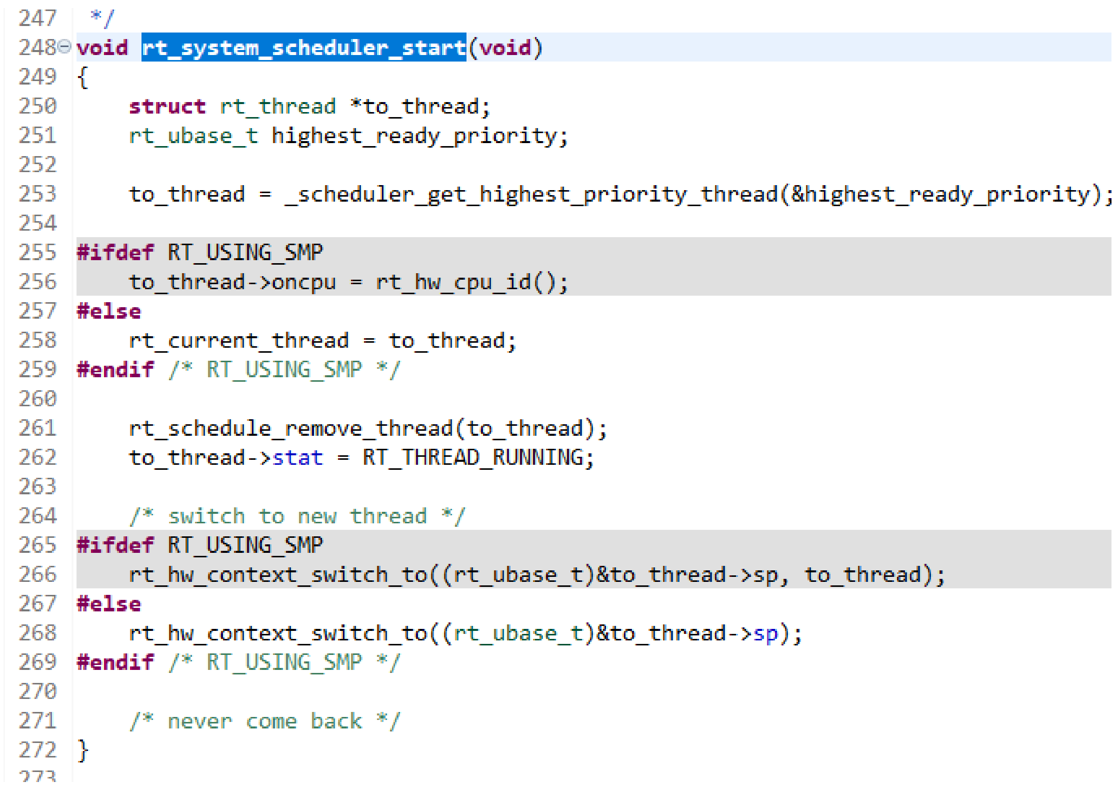
6、**rt_thread_idle_init 空闲线程初始化**



```c
 */
void rt_thread_idle_init(void)
{
    rt_ubase_t i;
    char tidle_name[RT_NAME_MAX];

    for (i = 0; i < _CPUS_NR; i++)
    {
        rt_sprintf(tidle_name, "tidle%d", i);
        rt_thread_init(&idle[i],
                tidle_name,
                rt_thread_idle_entry,
                RT_NULL,
                &rt_thread_stack[i][0],
                sizeof(rt_thread_stack[i]),
                RT_THREAD_PRIORITY_MAX - 1,
                32);
#ifdef RT_USING_SMP
        rt_thread_control(&idle[i], RT_THREAD_CTRL_BIND_CPU, (void*)i
#endif /* RT_USING_SMP */
        /* startup */
        rt_thread_startup(&idle[i]);
    }

```

7、rt_system_scheduler_start 开启系统任务调度

主要是获取最高优先级线程，将 sp 指针指向它，当前面执行完成后系统会自动执行从 sp 指针这里执行



```c
 */
void rt_system_scheduler_start(void)
{
    struct rt_thread *to_thread;
    rt_ubase_t highest_ready_priority;

    to_thread = _scheduler_get_highest_priority_thread(&highest_ready_priority);

#ifdef RT_USING_SMP
    to_thread->oncpu = rt_hw_cpu_id();
#else
    rt_current_thread = to_thread;
#endif /* RT_USING_SMP */

    rt_schedule_remove_thread(to_thread);
    to_thread->stat = RT_THREAD_RUNNING;

    /* switch to new thread */
#ifdef RT_USING_SMP
    rt_hw_context_switch_to((rt_ubase_t)&to_thread->sp, to_thread);
#else
    rt_hw_context_switch_to((rt_ubase_t)&to_thread->sp);
#endif /* RT_USING_SMP */

    /* never come back */
}
```