XHSC 小华半导体
XIAOHUA SEMICONDUCTOR

FreeRTOS 培训（一）
–系统介绍及系统移植

2025年3月28日

XHSC 小华半导体
XIAOHUA SEMICONDUCTOR

FreeRTOS系统介绍及系统移植

主讲人：欧阳健

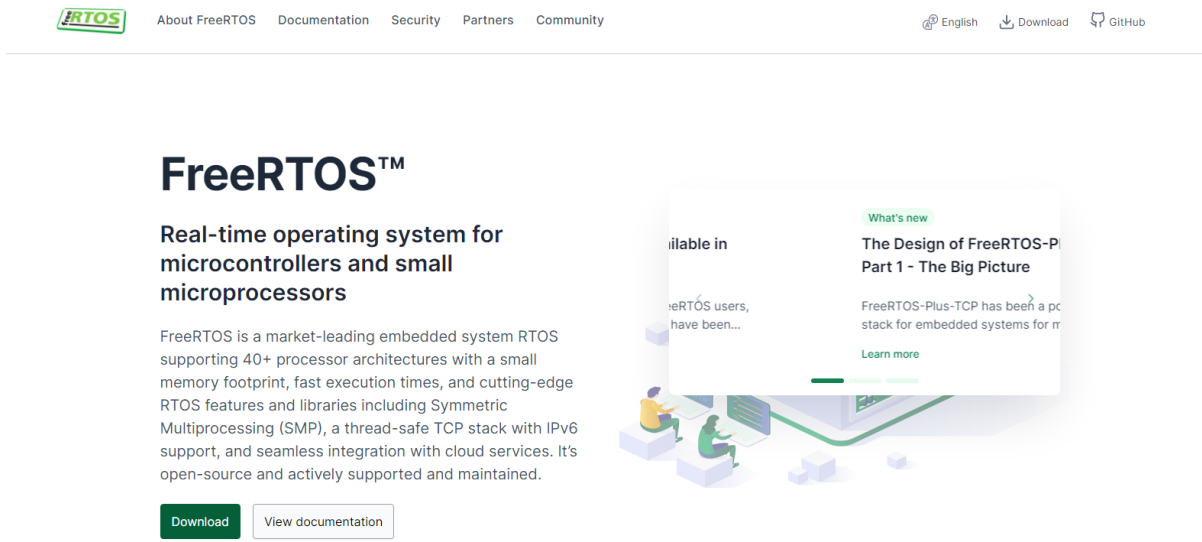
CEC 中国电子

内容目录

CONTENTS

FreeRTOS概述

- FreeRTOS是小型实时操作系统内核。作为一个轻量级的操作系统，其功能包括：任务管理、时间管理、信号量、消息队列、内存管理、记录功能、软件定时器、协程等，可基本满足较小系统的需要。
- 由于RTOS需占用一定的系统资源(尤其是RAM资源)，只有μC/OS-II、embOS、salvo、FreeRTOS等少数实时操作系统能在小RAM单片机上运行。相对μC/OS-II、embOS等商业操作系统，FreeRTOS操作系统是完全免费的操作系统，具有源码公开、可移植、可裁减、调度策略灵活的特点，可以方便地移植到各种单片机上运行，其最新版本为11.1.0版。
- 在嵌入式领域中，嵌入式实时操作系统正得到越来越广泛的应用。采用嵌入式实时操作系统(RTOS)可以更合理、更有效地利用CPU的资源，简化应用软件的设计，缩短系统开发时间，更好地保证系统的实时性和可靠性。
- https://www.freertos.org/



什么是实时操作系统

- 实时操作系统的核心特征
- **确定性（Determinism）**
  - 任务的执行时间可预测，无论是最坏情况执行时间（WCET）还是平均时间，均能通过调度算法保证。
- **优先级驱动的调度**
  - 支持基于优先级的抢占式调度（Preemptive Scheduling），高优先级任务可中断低优先级任务。
  - 调度算法需确保**高优先级任务始终优先执行**，如固定优先级调度（RMS）或动态优先级调度（EDF）。
- **低延迟中断处理**
  - 中断响应时间极短（通常为微秒级），确保外部事件能及时触发任务。
  - 例如：汽车安全气囊系统需要在碰撞检测后立即触发充气装置。
- **资源管理与同步**
  - 提供高效的资源共享机制（如信号量、互斥锁、消息队列），避免优先级反转（Priority Inversion）。
  - 通过优先级继承（Priority Inheritance）或天花板协议（Ceiling Protocol）解决资源竞争问题。
- **小型化与低开销**
  - 内核代码精简，内存占用小（通常为 KB 级），适用于资源受限的嵌入式设备。

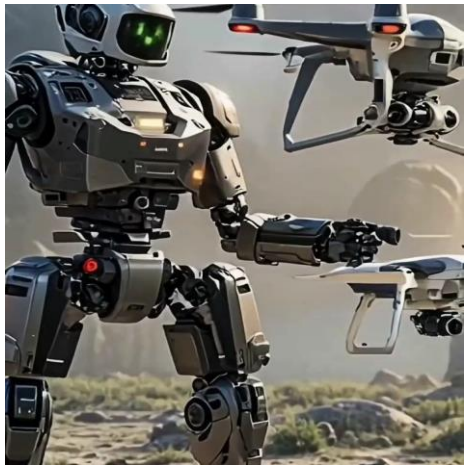
FreeRTOS的特点与优势

- 用户可配置内核功能
- 多平台的支持
- 提供一个高层次的信任代码的完整性
- 目标代码小，简单易用
- 遵循MISRA-C标准的编程规范
- 强大的执行跟踪功能
- 堆栈溢出检测
- 没有限制的任务数量
- 没有限制的任务优先级
- 多个任务可以分配相同的优先权队列，二进制信号量，计数信号灯和递归通信和同步的任务优先级继承
- 免费开源的源代码

FreeRTOS的应用场景

智能家居

工业自动化

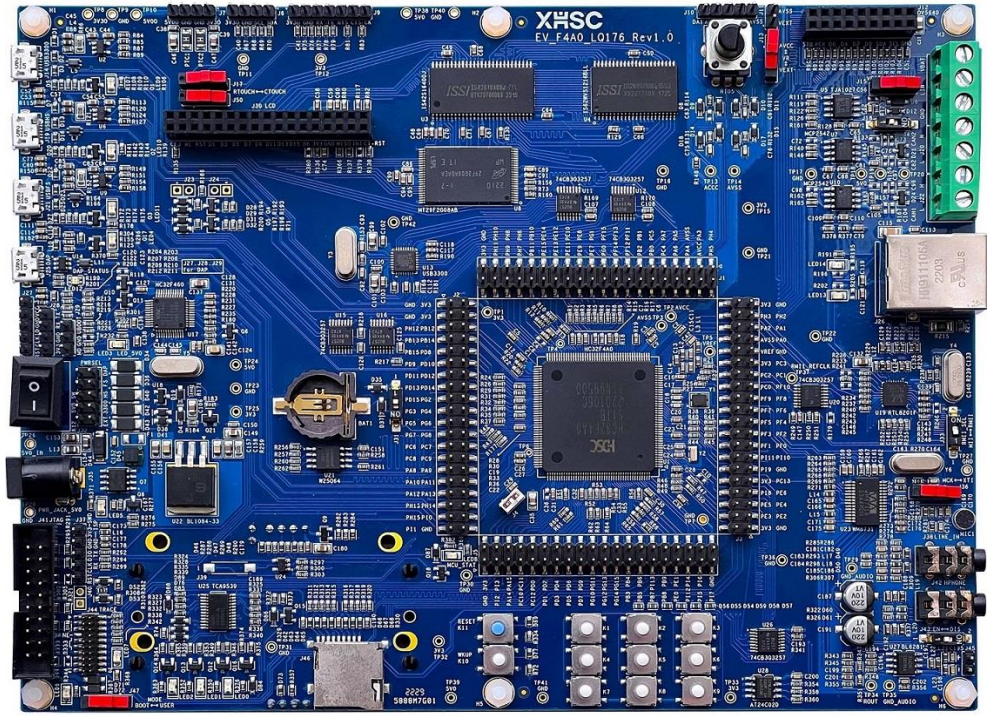


无人机和机器人

物联网

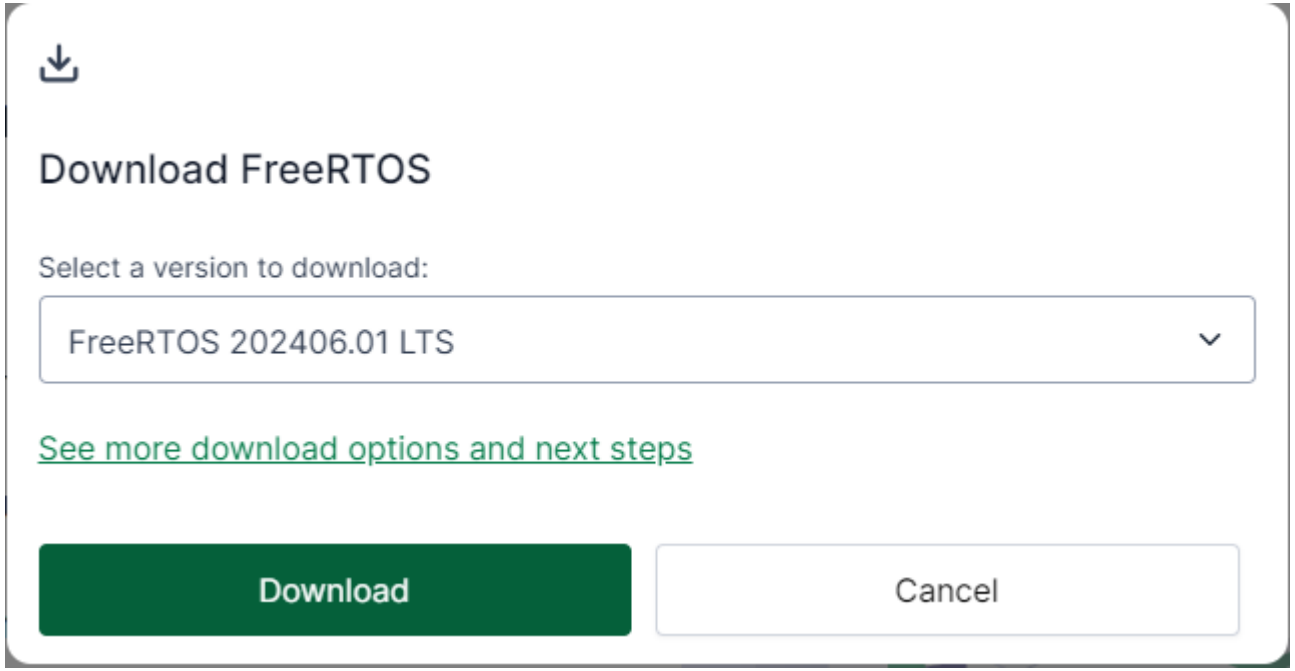
硬件平台和开发环境

- 以HC32F4A0 EVB为硬件平台
- 使用KEIL作为IDE工具，版本5.29.0, complier 6编译器
- 基于HC32F4A0_DDL_Rev2.3.0 驱动库
- FreeRTOS内核版本11.1.0



源码获取与移植

- https://www.freertos.org/

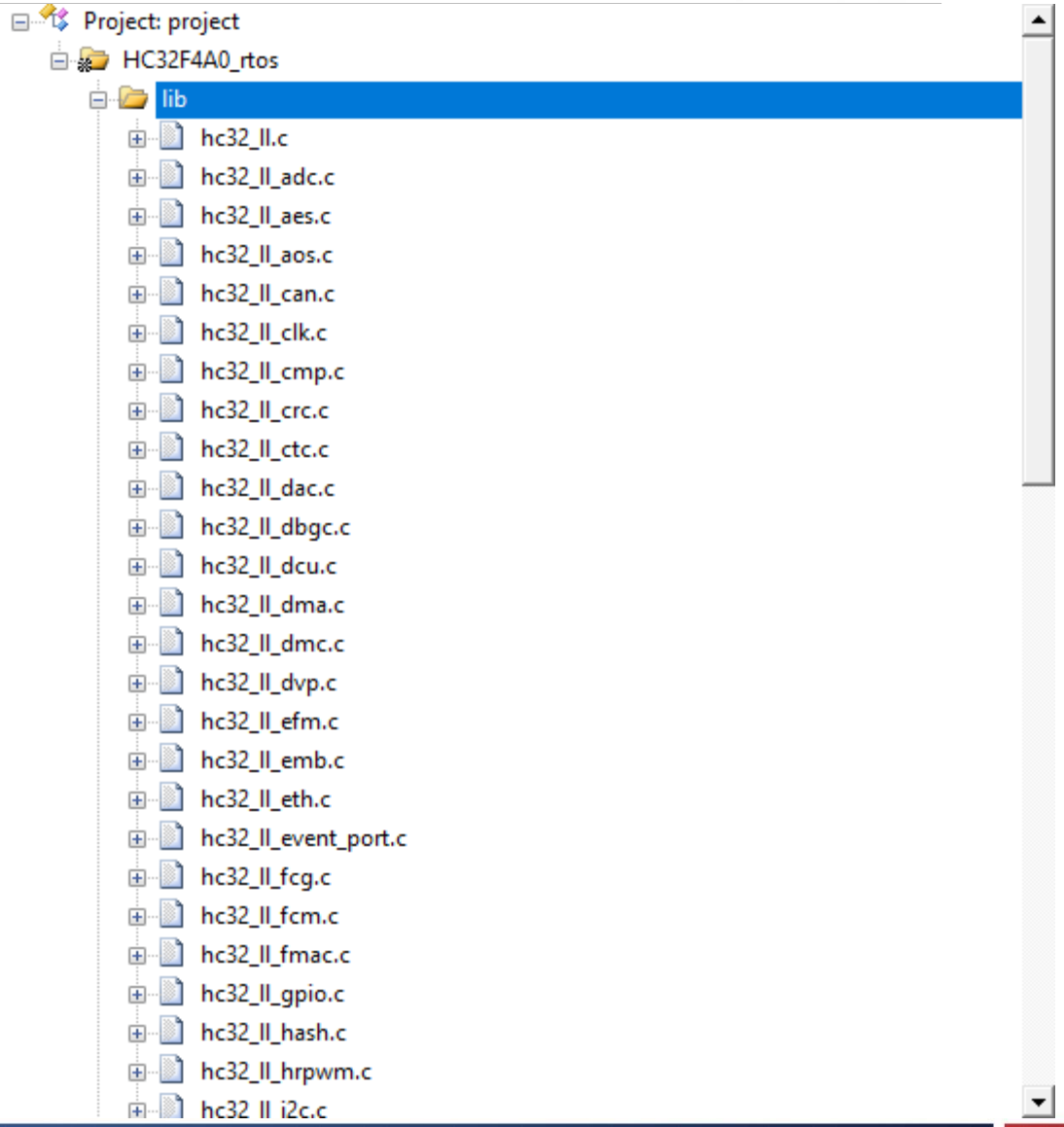


源码获取与移植

名称	修改日期	类型	大小
FreeRTOSv202406.01-LTS	2025/3/3 10:52	文件夹	
HC32F4A0_DDL_Rev2.3.0	2024/12/25 12:36	文件夹	
RTOS_project	2025/3/3 13:26	文件夹	
.gitignore	2025/3/3 11:06	GITIGNORE 文件	1 KB
LICENSE	2025/3/3 11:06	文件	10 KB
README.en.md	2025/3/3 11:06	Markdown 源文件	1 KB
README.md	2025/3/3 11:06	Markdown 源文件	2 KB

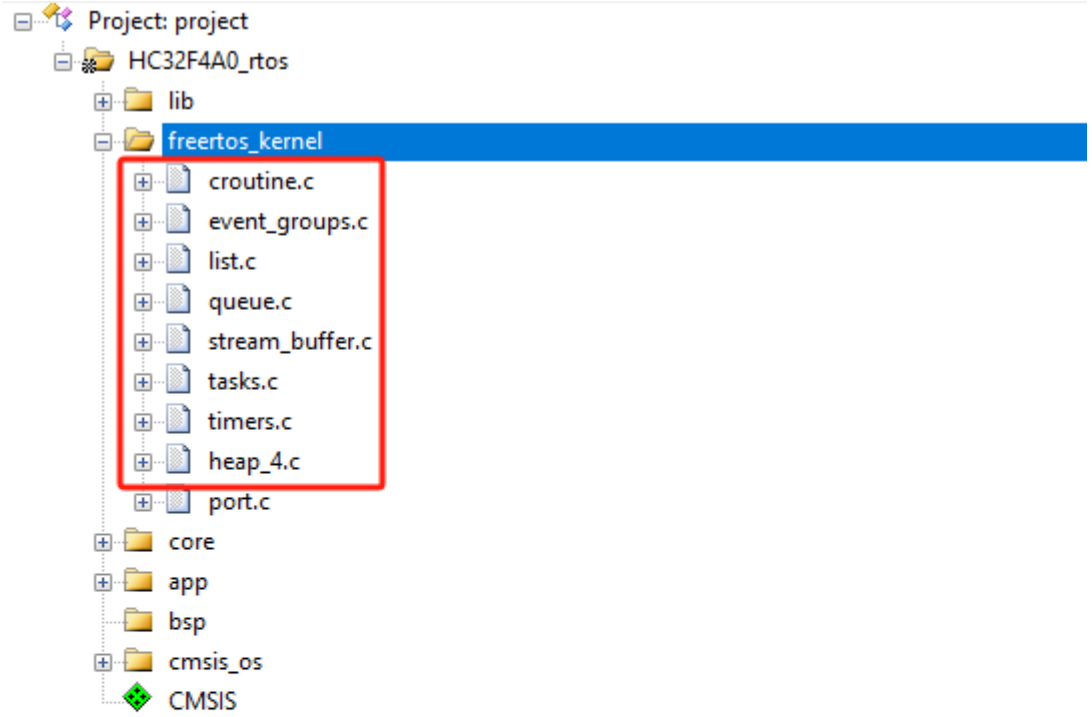
源码获取与移植

- 添加所需要的驱动库hc32_ll_xxx.c



源码获取与移植

- 添加freertos内核代码

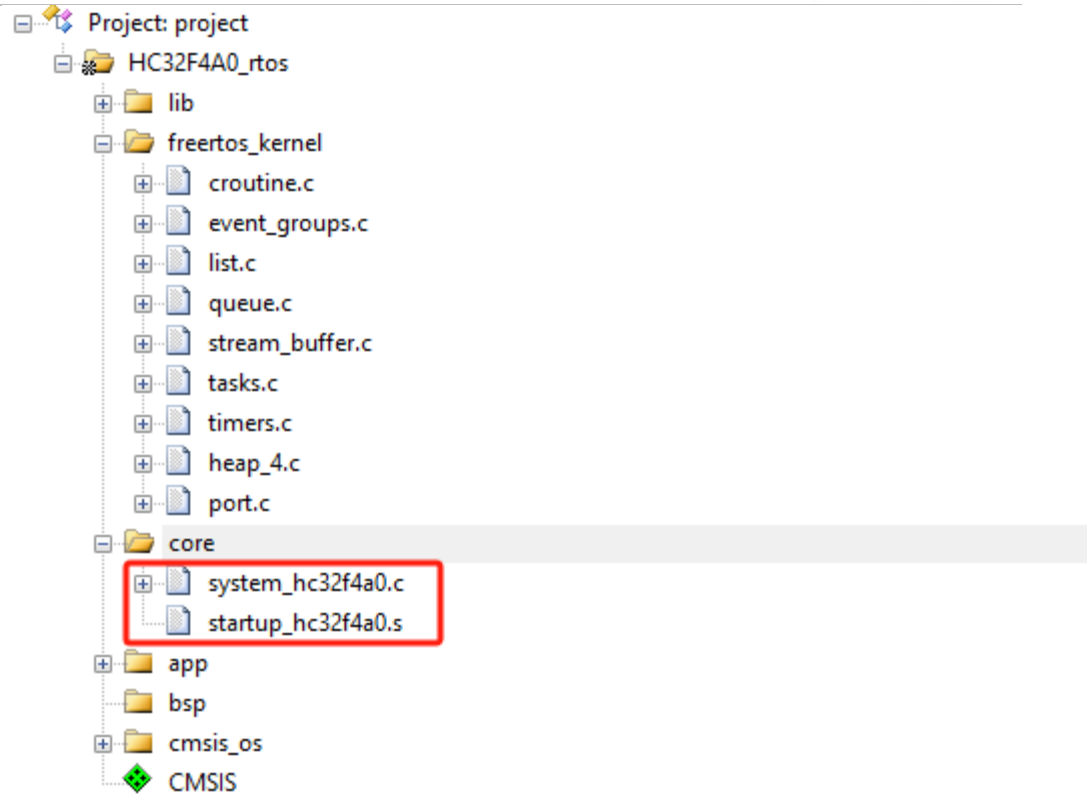


源码获取与移植

- 添加PORT
- Complier 6选择FreeRTOS-LTS\FreeRTOS\FreeRTOS-Kernel\portable\GCC\ARM_CM4F里的port.c

源码获取与移植

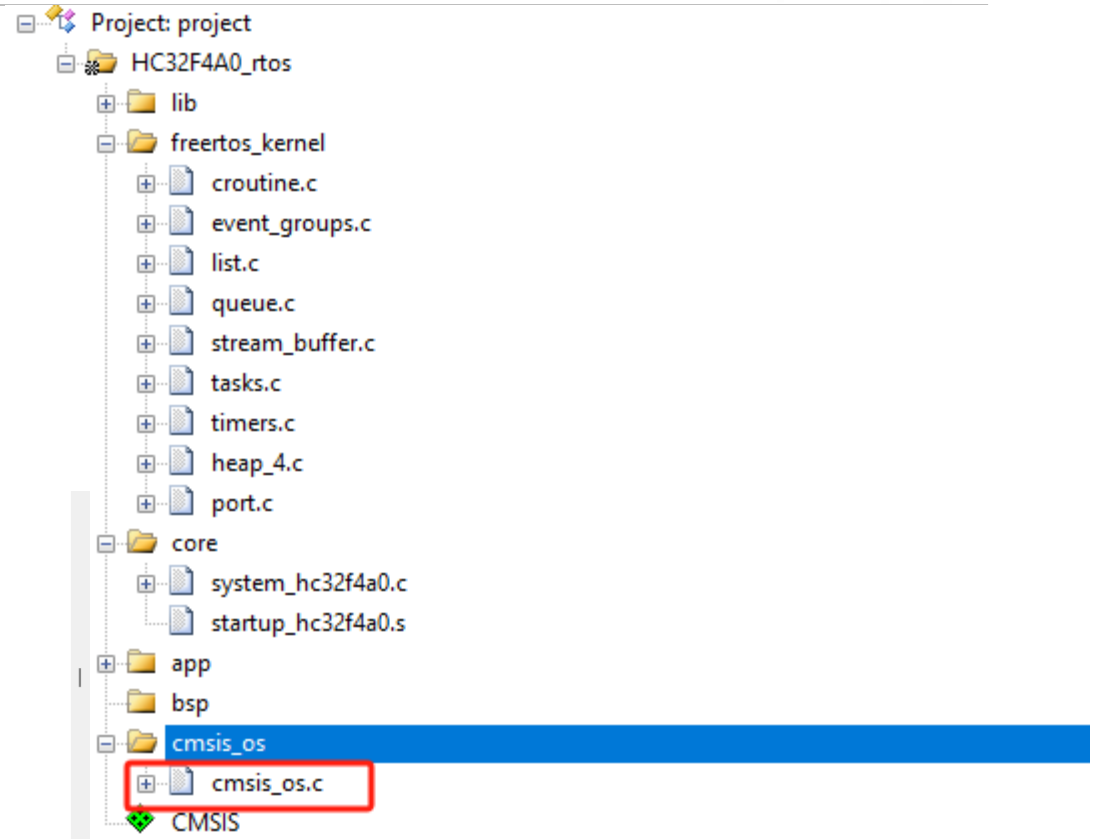
- 添加MCU启动代码



源码获取与移植

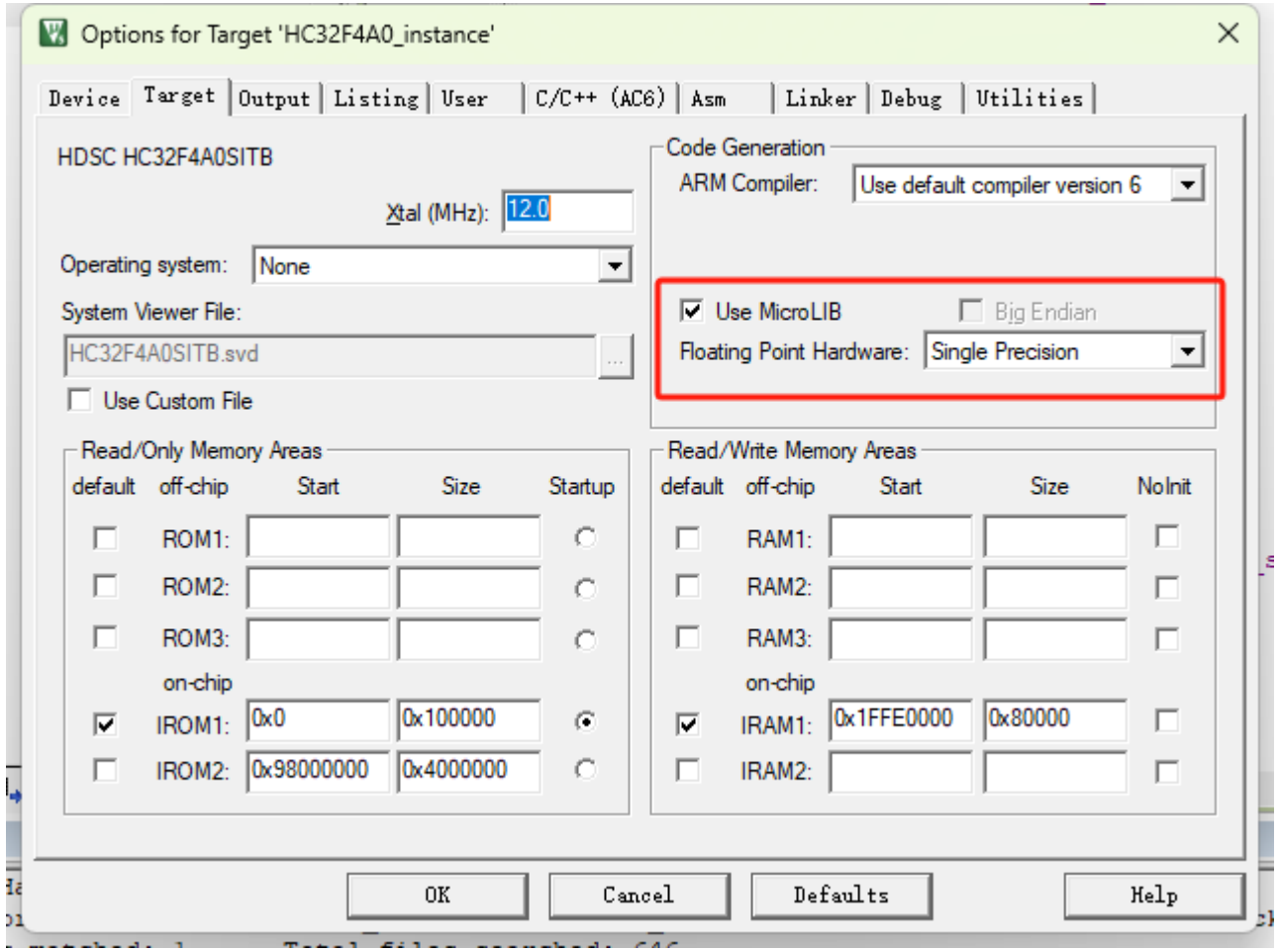
- 添加cmsis_os，并修改
- 新版本的freertos任务通知函数增加了索引参数，调用时需要增加这一入参，默认写0。

```
446  */
447  int32_t osSignalSet (osThreadId thread_id, int32_t signal)
448  {
449  #if( configUSE_TASK_NOTIFICATIONS == 1 )
450    BaseType_t xHigherPriorityTaskWoken = pdFALSE;
451    uint32_t ulPreviousNotificationValue = 0;
452
453    if (inHandlerMode())
454    {
455      if(xTaskGenericNotifyFromISR( thread_id , 0, (uint32_t)signal, eSetBits, &ulPreviousNotificationValue, &xHigherPriorityTaskWoken ) != pdPASS )
456        return 0x80000000;
457
458      portYIELD_FROM_ISR( xHigherPriorityTaskWoken );
459    }
460    else if(xTaskGenericNotify( thread_id , 0, (uint32_t)signal, eSetBits, &ulPreviousNotificationValue) != pdPASS )
461      return 0x80000000;
462
463    return ulPreviousNotificationValue;
464  #else
465    (void) thread_id;
466    (void) signal;
467
468    return 0x80000000; /* Task Notification not supported */
469  #endif
470  }
471
472  /**
473   * @brief  Clear the specified Signal Flags of an active thread.
474   * @param  thread_id  thread ID obtained by \ref osThreadCreate or \ref osThreadGetId.
475   * @param  signals    specifies the signal flags of the thread that shall be cleared.
476   * @retval  previous signal flags of the specified thread or 0x80000000 in case of incorrect parameters.
477   * @note   MUST REMAIN UNCHANGED: \b osSignalClear shall be consistent in every CMSIS-RTOS.
478   */
479  int32_t osSignalClear (osThreadId thread_id, int32_t signal);
480
481  /**
```



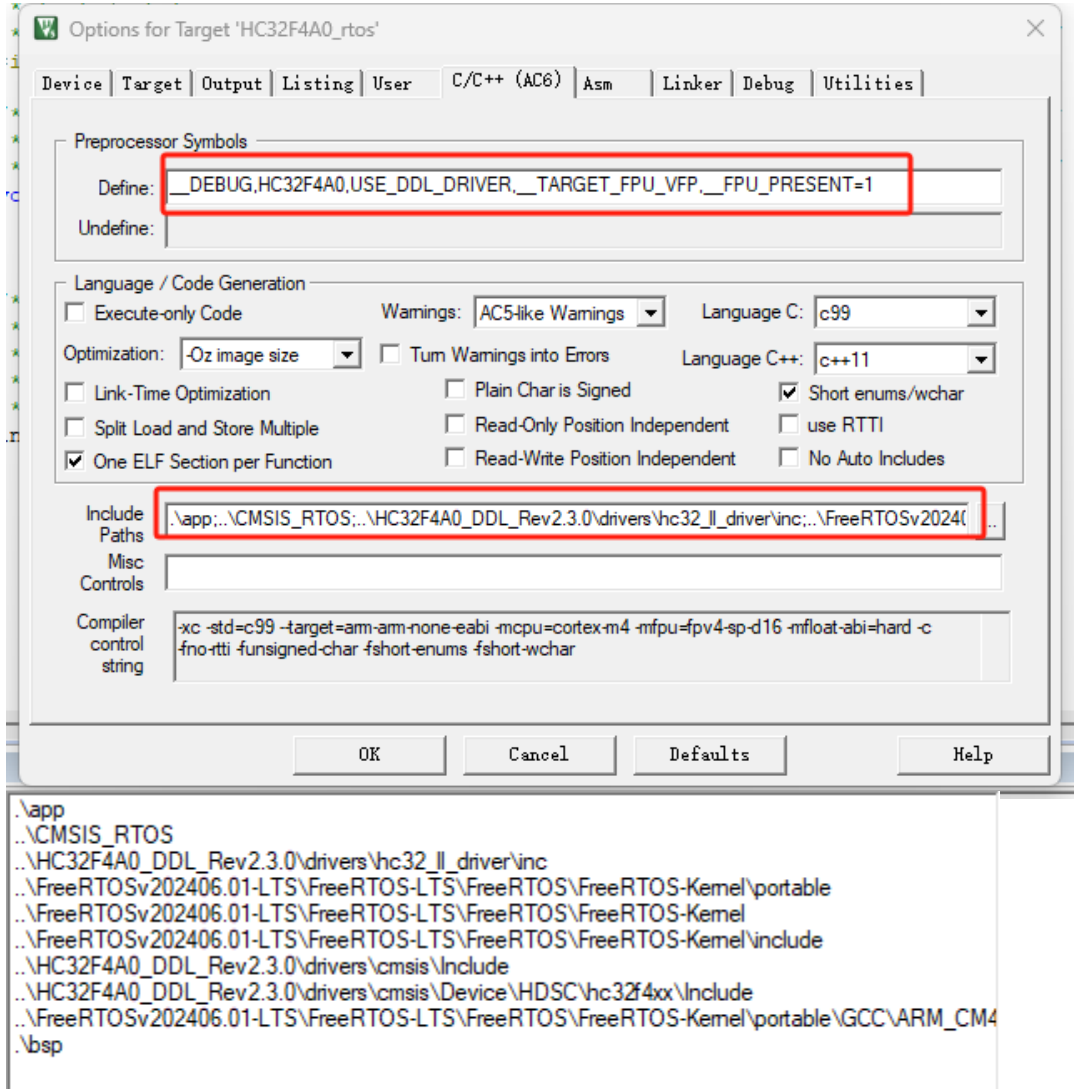
源码获取与移植

- 设置编译选项
  - 注意如果要使用printf等微库内的函数，要把MicroLIB勾选上。
  - 本实例打开了FPU，提升性能。



源码获取与移植

- 设置编译选项
- 全局宏:
  __DEBUG,HC32F4A0,USE_DDL_DRIVER,__TARGET_FPU_VFP,__FPU_PRESENT=1
- 添加路径
  - .\app;..\CMSIS_RTOS;
  - ..\HC32F4A0_DDL_Rev2.3.0\drivers\hc32_ll_driver\inc;
  - ..\FreeRTOSv202406.01-LTS\FreeRTOS-LTS\FreeRTOS\FreeRTOS-Kernel\portable;
  - ..\FreeRTOSv202406.01-LTS\FreeRTOS-LTS\FreeRTOS\FreeRTOS-Kernel;
  - ..\FreeRTOSv202406.01-LTS\FreeRTOS-LTS\FreeRTOS\FreeRTOS-Kernel\include;
  - ..\HC32F4A0_DDL_Rev2.3.0\drivers\cmsis\Include;
  - ..\HC32F4A0_DDL_Rev2.3.0\drivers\cmsis\Device\HDSC\hc32f4xx\Include;
  - ..\FreeRTOSv202406.01-LTS\FreeRTOS-LTS\FreeRTOS\FreeRTOS-Kernel\portable\GCC\ARM_CM4F;
  - .\bsp



源码获取与移植

- 初始化系统时钟

portmacro.h | bsp_clk.c 2 | main.c

RTOS_project > bsp > bsp_clk.c > bsp_system_clk_init(void)

```c
#include "bsp_clk.h"


void bsp_system_clk_init(void)
{
    stc_clock_xtal_init_t stcXtalInit;
    stc_clock_pll_init_t stcPLLHInit;

    /* PCLK0, HCLK  Max 240MHz */
    /* PCLK1, PCLK4 Max 120MHz */
    /* PCLK2, PCLK3 Max 60MHz  */
    /* EX BUS Max 120MHz */
    CLK_SetClockDiv(CLK_BUS_CLK_ALL, \
                    (CLK_PCLK0_DIV1 | CLK_PCLK1_DIV2 | CLK_PCLK2_DIV4 | \
                     CLK_PCLK3_DIV4 | CLK_PCLK4_DIV2 | CLK_EXCLK_DIV2 | \
                     CLK_HCLK_DIV1));

    GPIO_AnalogCmd(GPIO_PORT_H, GPIO_PIN_01 | GPIO_PIN_00, ENABLE);
    (void)CLK_XtalStructInit(&stcXtalInit);
    /* Config Xtal and enable Xtal */
    stcXtalInit.u8Mode   = CLK_XTAL_MD_OSC;
    stcXtalInit.u8Drv    = CLK_XTAL_DRV_ULOW;
    stcXtalInit.u8State  = CLK_XTAL_ON;
    stcXtalInit.u8StableTime = CLK_XTAL_STB_2MS;
    (void)CLK_XtalInit(&stcXtalInit);

    (void)CLK_PLLStructInit(&stcPLLHInit);
    /* VCO = (8/1)*120 = 960MHz*/
    stcPLLHInit.u8PLLState = CLK_PLL_ON;
    stcPLLHInit.PLLCFGR = 0UL;
    stcPLLHInit.PLLCFGR_f.PLLM = 1UL - 1UL;
    stcPLLHInit.PLLCFGR_f.PLLN = 120UL - 1UL;
    stcPLLHInit.PLLCFGR_f.PLLP = 4UL - 1UL;
    stcPLLHInit.PLLCFGR_f.PLLQ = 4UL - 1UL;
    stcPLLHInit.PLLCFGR_f.PLLR = 4UL - 1UL;
    stcPLLHInit.PLLCFGR_f.PLLSRC = CLK_PLL_SRC_XTAL;
    (void)CLK_PLLInit(&stcPLLHInit);

    /* Highspeed SRAM set to 0 Read/Write wait cycle */
    SRAM_SetWaitCycle(SRAM_SRAMH, SRAM_WAIT_CYCLE0, SRAM_WAIT_CYCLE0);

    /* SRAM1_2_3_4_backup set to 1 Read/Write wait cycle */
    SRAM_SetWaitCycle((SRAM_SRAM123 | SRAM_SRAM4 | SRAM_SRAMB), SRAM_WAIT_CYCLE1

    /* 0-wait @ 40MHz */
    (void)EFM_SetWaitCycle(EFM_WAIT_CYCLE5);
    EFM_ICacheCmd(ENABLE);
    EFM_DCacheCmd(ENABLE);
```

源码获取与移植

- 添加Tick中断处理，将osSystickHandler();
  加到中断处理中

```
32  ********************************************************************
33  /*MCU Peripheral Initial*/
34  void core_init(void)
35  {
36      bsp_system_clk_init();
37      bsp_gpio_init();
38  }
39  /*systick handler*/
40  void SysTick_Handler(void)
41  {
42      SysTick_IncTick();
43      osSystickHandler();
44  }
45  /**
46   * @brief  Main function of ADC analog watchdog project
47   * @param  None
48   * @retval int32_t return value, if needed
49   */
50  int32_t main(void)
51  {
52      /* MCU Peripheral registers write unprotected. */
53      LL_PERIPH_WE(LL_PERIPH_ALL);
```

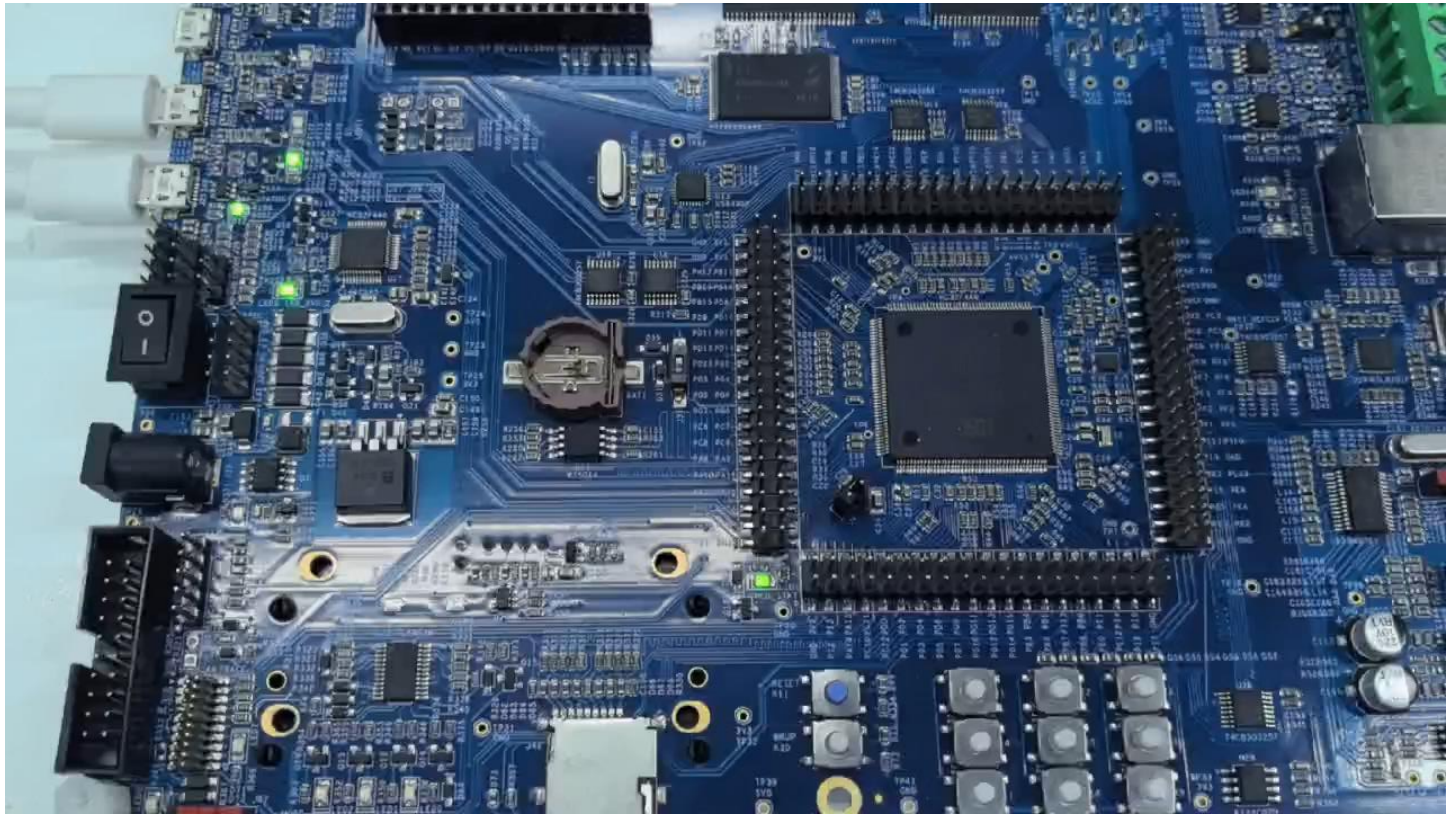
第一个FreeRTOS示例程序

- 建立第一个任务，点亮一个LED灯。每秒闪烁一次

```c
#include "task_led.h"
TaskHandle_t Hd_Task_led;

void task_led(void *arg)
{
    while(1)
    {
        bsp_led_toggle();
        vTaskDelay(1000/portTICK_PERIOD_MS);
    }
}

void led_task_create(void)
{
    xTaskCreate(task_led,(const char *)"LED_Task", configMINIMAL_STACK_SIZE*2, NULL, tskIDLE_PRIORITY+3, &Hd_Task_led);
}

void OS_Start(void)
{
    led_task_create();
    vTaskStartScheduler();
}
```

第一个FreeRTOS示例程序

- 运行效果

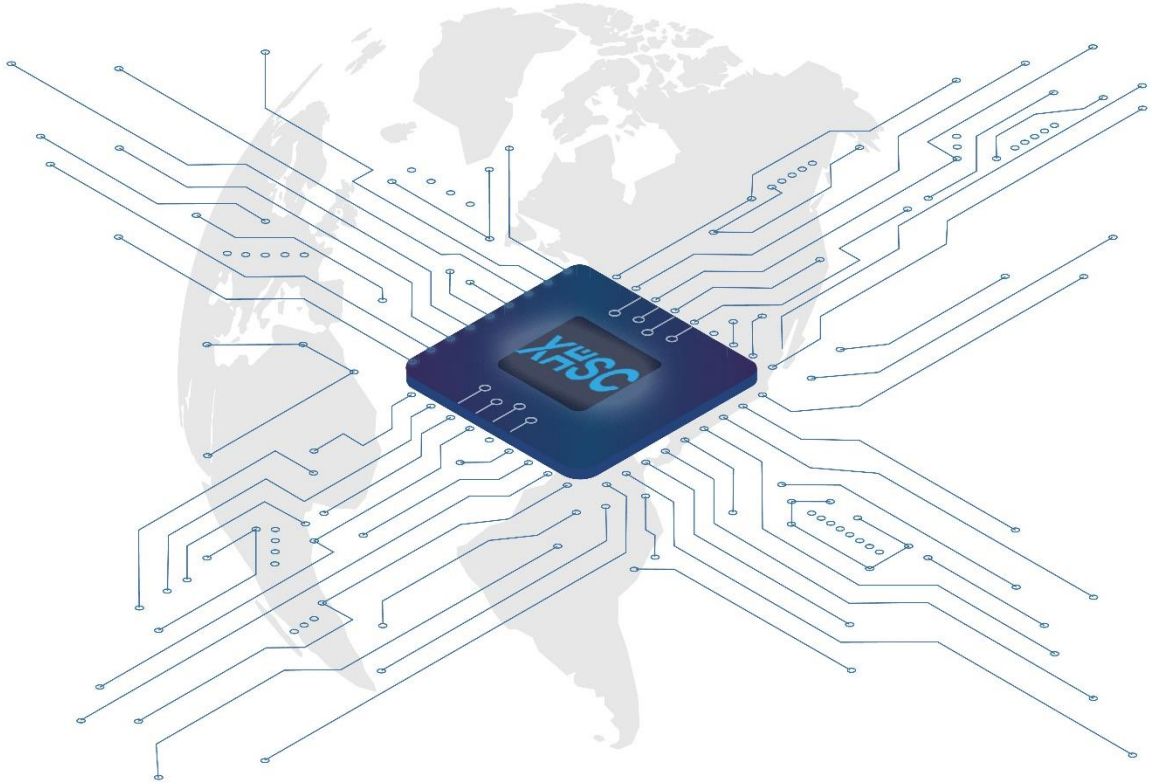


XHSC 小华半导体
XIAOHUA SEMICONDUCTOR

Q&A

铸就芯程 智造未来

地址：上海市浦东新区中科路1867号A座10层
电话：021-38880888
官网：www.xhsc.com.cn



XHSC 小华半导体 XIAOHUA SEMICONDUCTOR