[HDU-CT] Embedded Systems Design (Summer 2022)

Lab2

University : HDU

Institute: ITMO joint institute

Name: Zhan Peng

ID: 212320017

Specialty: Computer Science

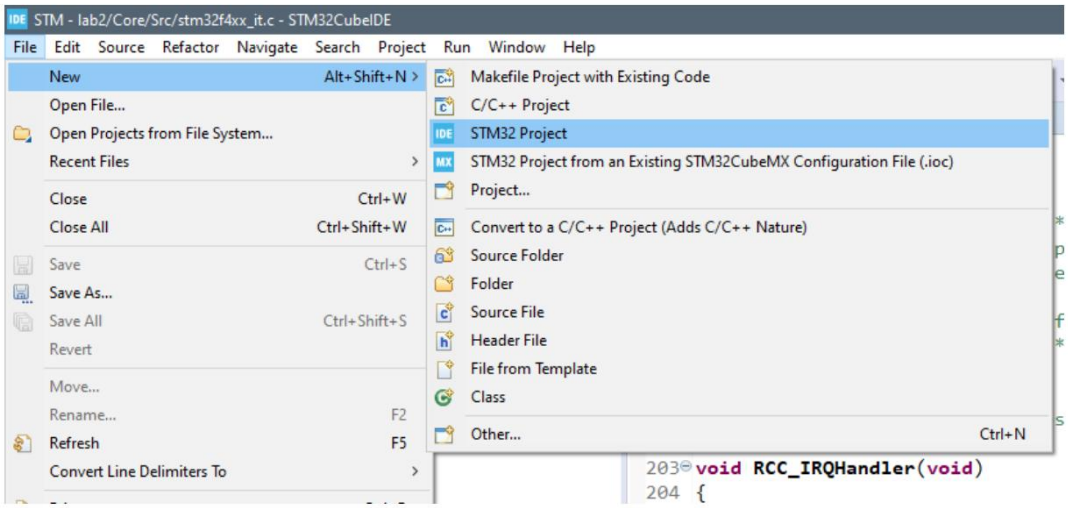
Year of studying: 2022

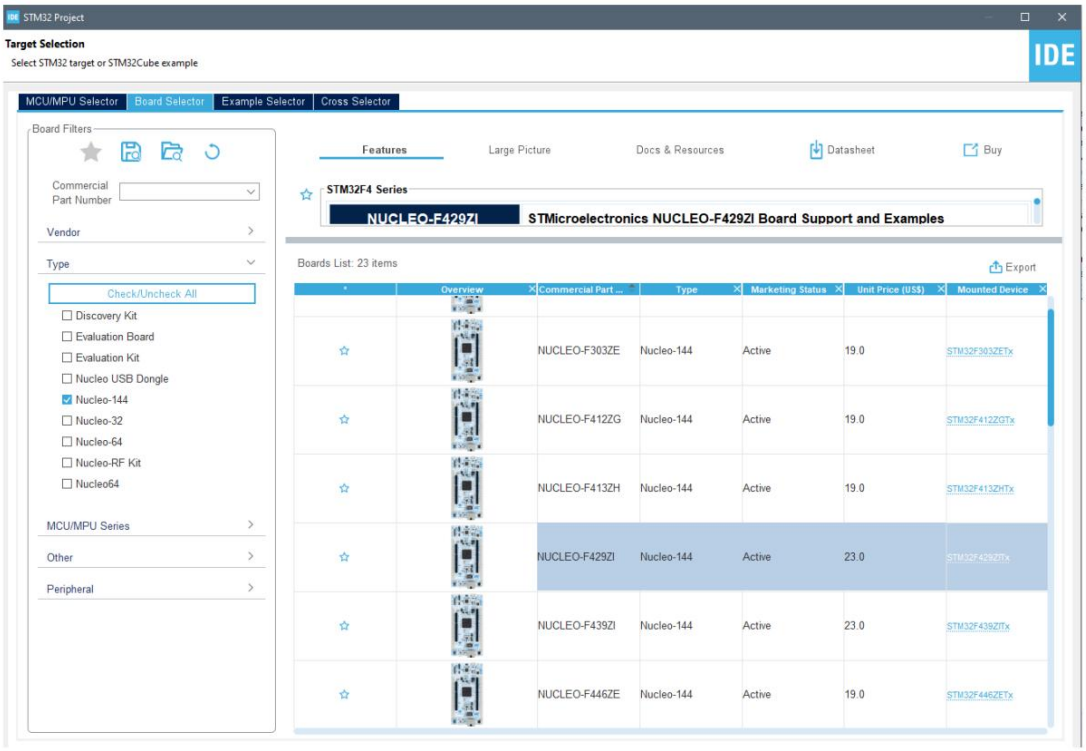
Number of the work: 2

Date: Sep 23rd

Description of project creation and programming process

1. Create new STM32 project .

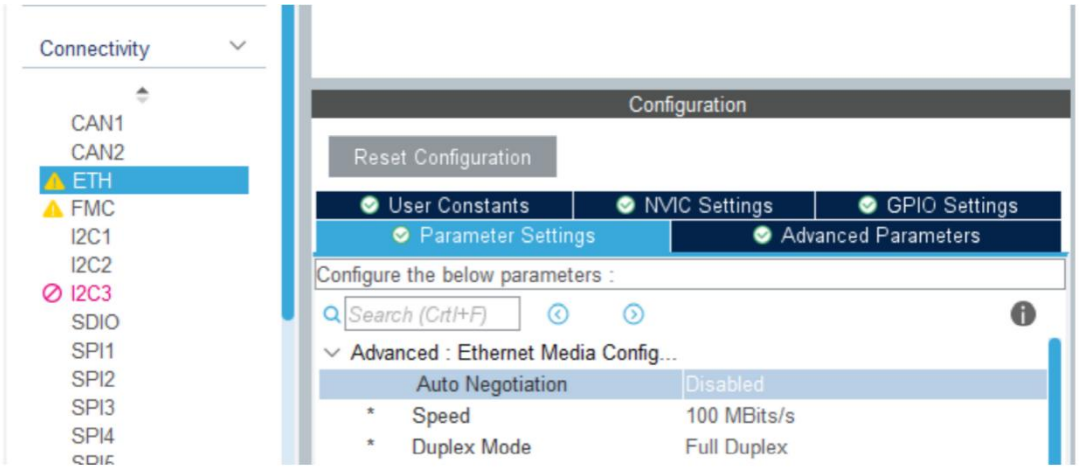




My variant is follow Variant 7:

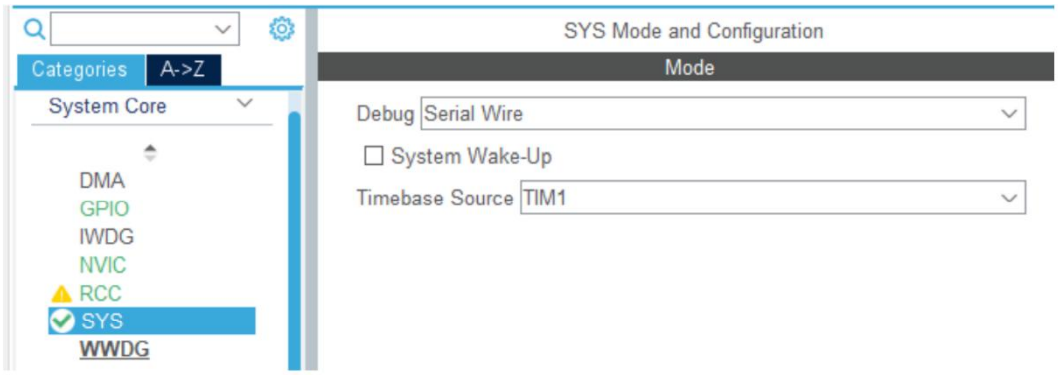
R high G Normal B 5B-1G-5R

- As first step of configuration, disable Ethernet auto negotiation.

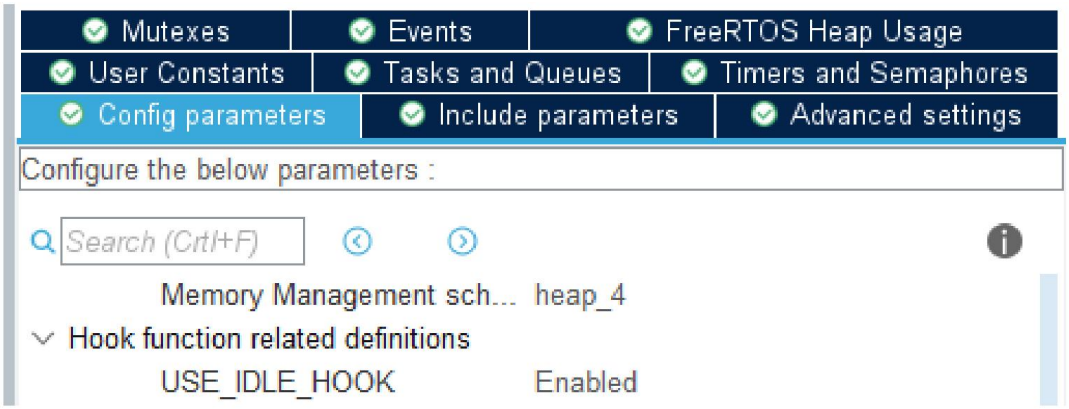


- In the Configuration tool configure FreeRTOS in the Middleware and choose CMSIS_V1.

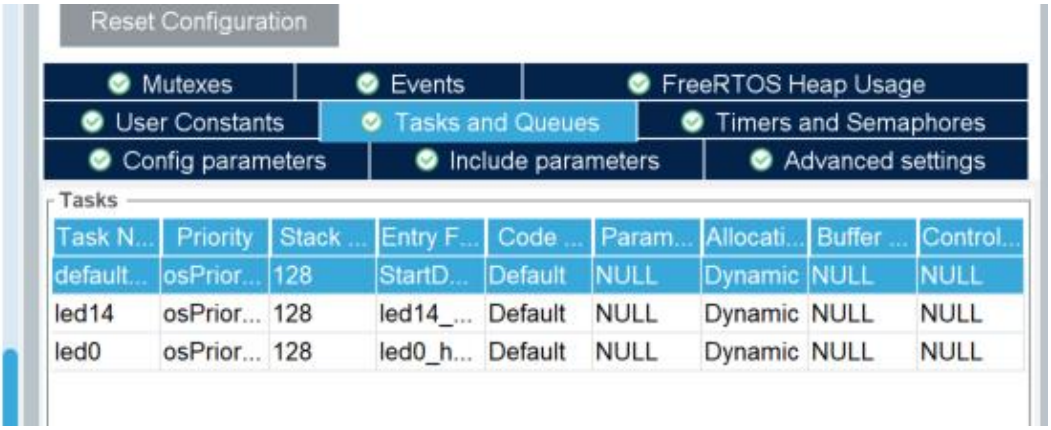
- Select Timer 1 to be the Timebase Source in System Core → SYS setting.



- Enable Idle Hook in order to implement an idle function.



- Create two simple tasks with different priorities to control green and blue LEDs.



- Click the Save button and generate the code.

2. Main.c

I can see two tasks already created in the main function. The kernel starts after the tasks creation.

```
/* definition and creation of led14 */
osThreadDef(led14, led14_handler, osPriorityHigh, 0, 128);
led14Handle = osThreadCreate(osThread(led14), NULL);

/* definition and creation of led0 */
osThreadDef(led0, led0_handler, osPriorityNormal, 0, 128);
led0Handle = osThreadCreate(osThread(led0), NULL);
```

Add an idle hook using the function vApplicationIdleHook(). In this case it will blink the red LED.

```
/* USER CODE BEGIN 4 */
void vApplicationIdleHook(void)
{
  /* USER CODE BEGIN 5 */
    int i = 0;
    for(;;)
    {
      HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_7);
      HAL_Delay(500);
      if(i == 9){
          i = 0;
          HAL_GPIO_WritePin(GPIOB, GPIO_PIN_7, 0);
          osThreadResume(led0Handle);
      }
      i ++;
    }
  /* USER CODE END 5 */
}
```

Write the task function to control the green LED. We suspend the function by using function vTaskDelay() (same as osDelay()). You may change it to HAL_Delay() and see what's the difference.

```
/* USER CODE END Header_led14_handler */
void led14_handler(void const * argument)
{
/* USER CODE BEGIN led14_handler */
  /* Infinite loop */
    osThreadSuspend(NULL);
    int i = 0;
    for(;;)
    {
      HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_14);
      HAL_Delay(500);
      if(i == 9){
          i = 0;
          HAL_GPIO_WritePin(GPIOB, GPIO_PIN_14, 0);
          osThreadSuspend(NULL);
      }
      i ++;
    }
  /* USER CODE END led14_handler */
}
```

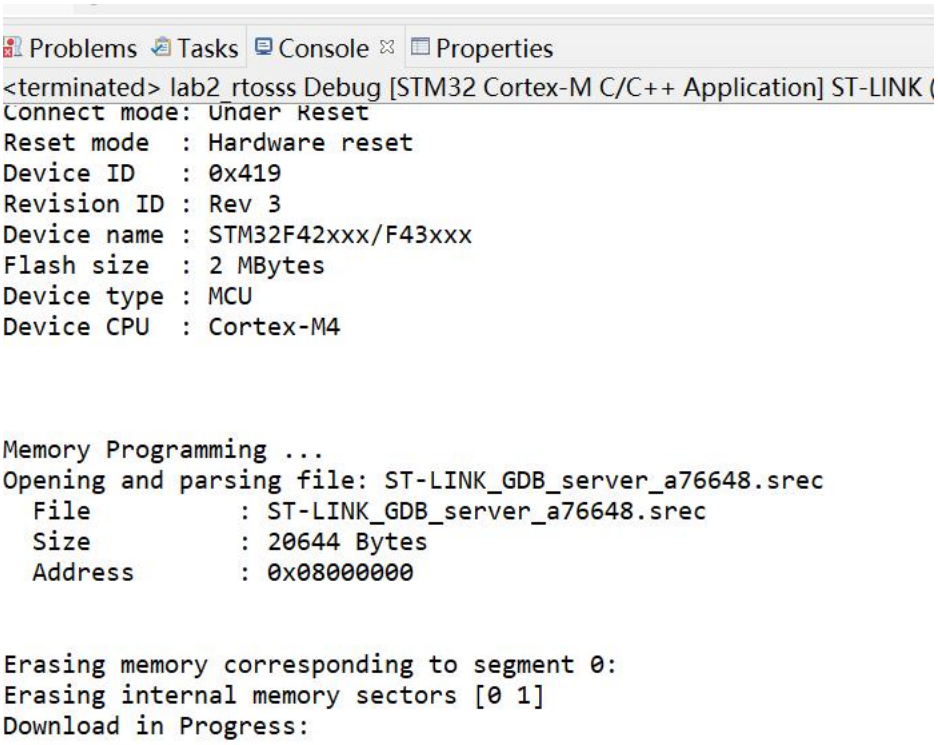
In the same way we write the function to control the blue LED. This task was declared to have higher priority. It will work for 10 ticks and then delete itself.

```
/* USER CODE END Header_led0_handler */
void led0_handler(void const * argument)
{
  /* USER CODE BEGIN led0_handler */
  /* Infinite loop */
    osThreadSuspend(NULL);
    int i = 0;
    for(;;)
    {
      HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_0);
      HAL_Delay(500);
      if(i == 1){
          i = 0;
          HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, 0);
          osThreadResume(led14Handle);
          osThreadSuspend(NULL);
      }
      i ++;
    }
  /* USER CODE END led0_handler */
}
```

3. Save and build the project.

```
18:07:31 **** Incremental Build of configuration Debug for project lab2_rtosss ****
make -j4 all
arm-none-eabi-size   lab2_rtosss.elf
   text    data     bss     dec     hex filename
  20616      24   19120   39760    9b50 lab2_rtosss.elf
Finished building: default.size.stdout


18:07:32 Build Finished. 0 errors, 0 warnings. (took 1s.54ms)
```


Problems
Tasks
Console
Properties
<terminated> lab2_rtosss Debug [STM32 Cortex-M C/C++ Application] ST-LINK (
Connect mode: Under Reset
Reset mode : Hardware reset
Device ID : 0x419
Revision ID : Rev 3
Device name : STM32F42xxx/F43xxx
Flash size : 2 MBytes
Device type : MCU
Device CPU : Cortex-M4
Memory Programming ...
Opening and parsing file: ST-LINK_GDB_server_a76648.srec
File : ST-LINK_GDB_server_a76648.srec
Size : 20644 Bytes
Address : 0x08000000
Erasing memory corresponding to segment 0:
Erasing internal memory sectors [0 1]
Download in Progress: