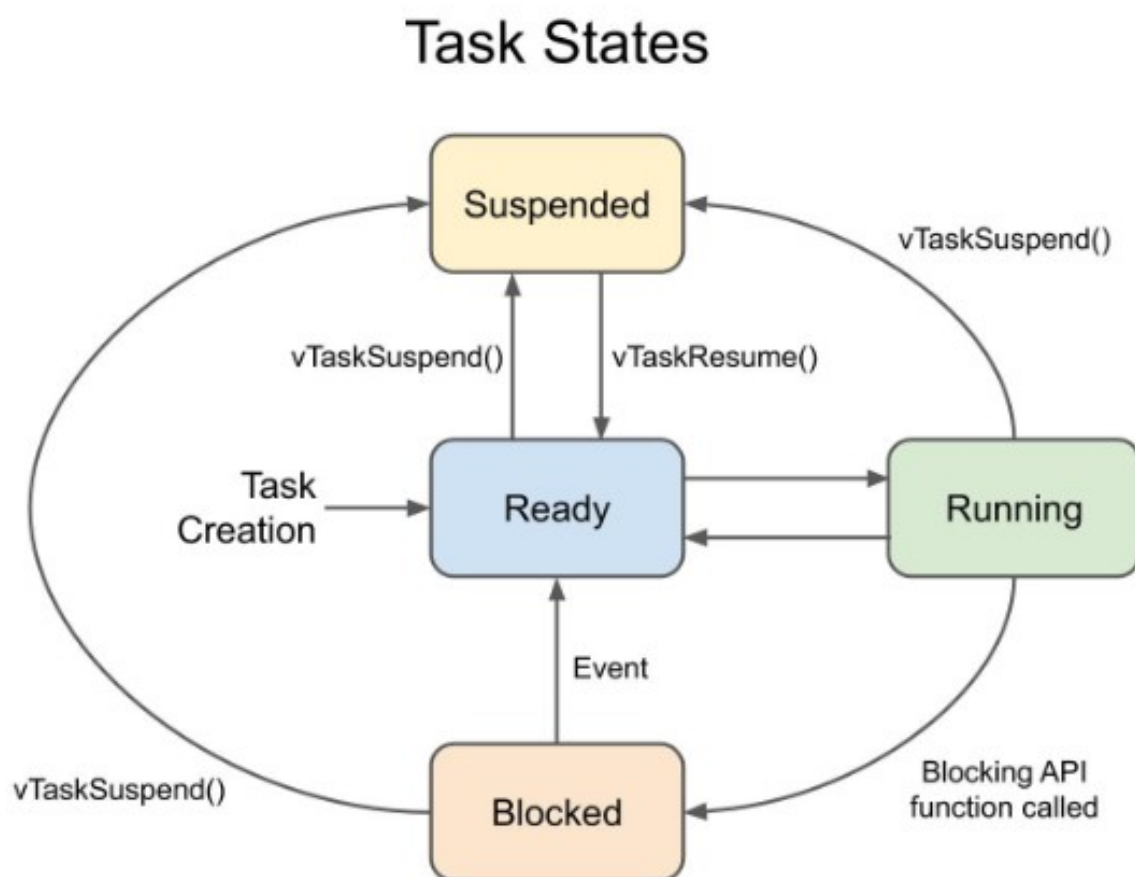


Lab 1 Hello Real World



Section 1 Lab Objectives

When this lab exercise is completed, the student should be able to:

1. Create a task in FreeRTOS, a widely used real time operating system.
2. Setup two tasks in a RTOS.
3. Change a task's priority.

Section 2 PreLaboratory Preparation

Prior to your scheduled laboratory meeting time the following items need to be completed.

Research

1. Read chapter 1 of the course text, Real-Time Operating Systems Book 1 - The Theory. Sections 6 and 8 will be covered on the quiz.

Video 2.5 ST Login <https://www.youtube.com/watch?v=MVs3vykxxM4> First three minutes of the video provide a lot of background that can be skipped.

On Line Learning

1. Watch *Introduction to RTOS* <https://www.youtube.com/watch?v=F321087yYy4&list=PLEBQazBOHUYQ4hAPU1cJED6t3DU0h34bz>
2. Watch *STM32 CubeIDE Getting Started* -

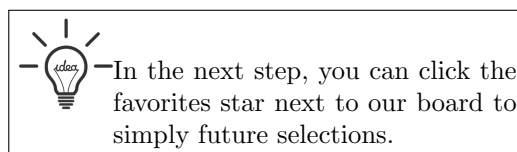
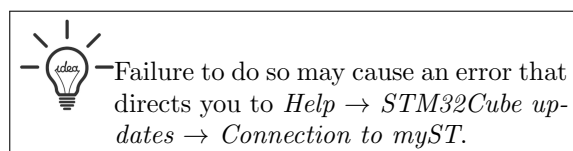
Preparation for NEXT WEEK'S Prelab Quiz

Starting next week, a paper quiz will be available in lab for the first ten (10) minutes of the lab session. Next week's quiz will cover the prelab preparation of both this week and next week's lab.

Section 3 Simple Output

Section 3.1 Login to STM32Cube

You are required to login to STM32Cube. The on line learning section of the pre-lab preparation provides a video with login instructions.



Section 3.2 Project Creation

1. Start STM32CubeIDE, select File → New → STM32 Project. You will now be presented with figure 1.

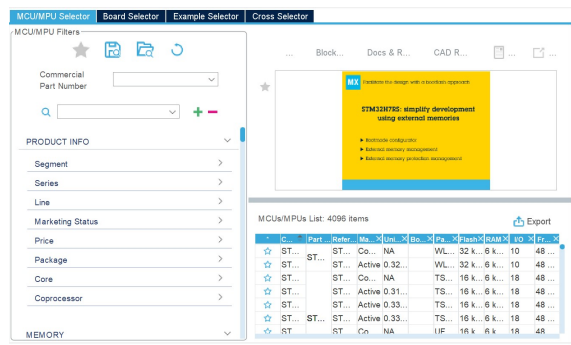


Figure 1: Target Selection

2. Click the Board Selector Tab and enter Nucleo-L476RG in the Commercial Part Number box. Select the Nucleo-l476RG and click Next. Click the favorites star to simply future selection.

3. Enter a project name click Finish.

4. Click yes to initializing all peripherals with default values.

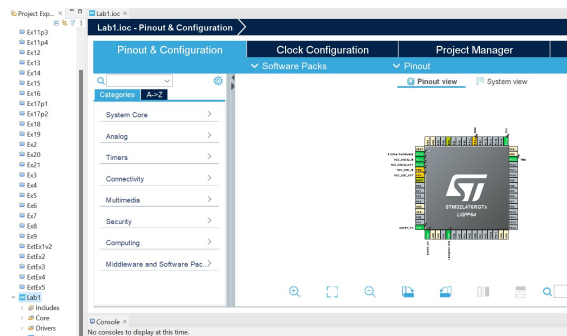





Figure 2: Resulting Project Screen

 The outcome of doing this is that the pinout view of the microcontroller. The LED and push button connections are shown in green. Locate them on the chip diagram. Your Project Explorer window may only have one project. This list will grow as the semester progresses.

 The LED is at pin 5 of GPIO group A. This can be determined by looking at the microcontroller in the IOC file. The LED pin is labeled PA5.

 When you add code to main.c, keep it between any USER CODE BEGIN and END comments. Code outside these areas will be deleted each time main is updated by pressing the gear or changing/saving the IOC file.

Section 3.3 Blink the on board LED

1. Click the yellow gear icon in the center of the task bar, shown in figure 3.

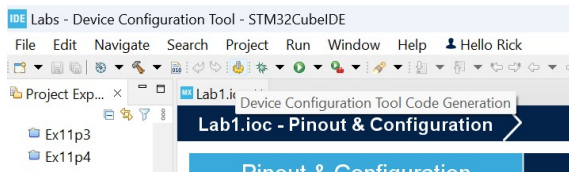



Figure 3: Device Configuration Tool Code Creation


 You will now find that additional folders and files are automatically generated. The folder contains all files need to program your chip. At the stage the automatically-generated code doesn't do anything apart from initializing the micro. You will add to main.c to blink the LED.

2. Search for while (1) in main.c. Add the HAL command as shown below.

```

1  /* Infinite loop */
2  /* USER CODE BEGIN WHILE */
3  while (1)
4  {
5      HAL_GPIO_TogglePin(GPIOA,
6                          GPIO_PIN_5);
7      HAL_Delay(500);
8      /* USER CODE END WHILE */
9
10     /* USER CODE BEGIN 3 */
11 }

```

 To use the autocomplete feature when entering long commands, start typing hal_gp and then press CTRL-space. Find the command in the list.

3. Connect the computer to the board using a USB cord.
4. To run the program, click the green bug in the center of the task bar, shown in figure 4.

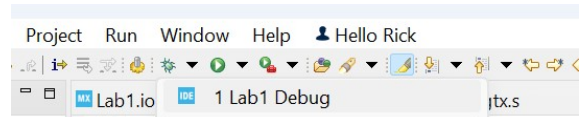



Figure 4: Start Debug

 The program will stop at HAL_INIT(). You can set break points after this point to stop the program.

5. To resume the program, click the play button on the left side of the task bar, shown in figure 5.

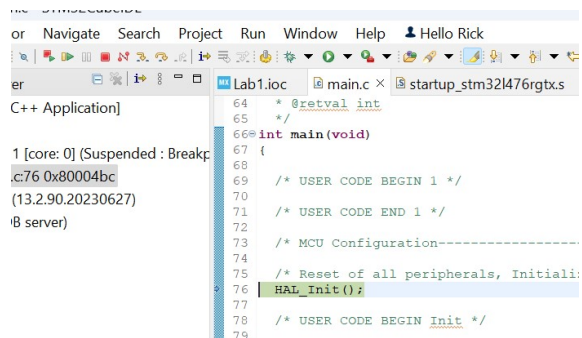



Figure 5: Resume Debug

6. Verify a green LED is blinking on and then off every second. The on board user LED is labeled LD2.

 You can also turn the LED on using HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET); and turn it off using HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);

Section 4 RTOS Output

Section 4.1 Blink the LED using a RTOS Task

1. Comment out code you entered in main's while loop in the last section, as shown below.

```

1  /* Infinite loop */
2  /* USER CODE BEGIN WHILE */
3  while (1)
4  {
5      /* HAL_GPIO_TogglePin(GPIOA,
6       GPIO_PIN_5); */
7      /* HAL_Delay(500); */
8      /* USER CODE END WHILE */
9
10     /* USER CODE BEGIN 3 */
11 }

```

2. Select Window → Show View → Project Explorer as shown in Figure 6 to turn on the Project Explorer window.

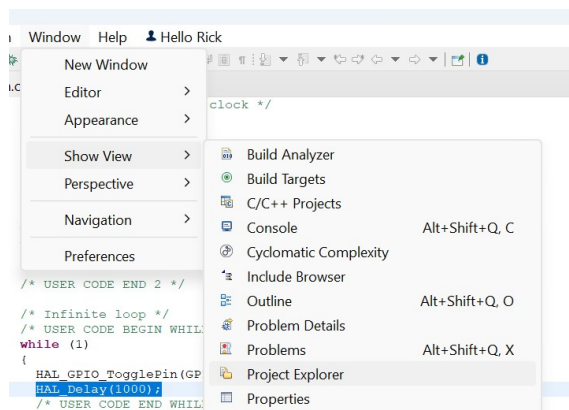
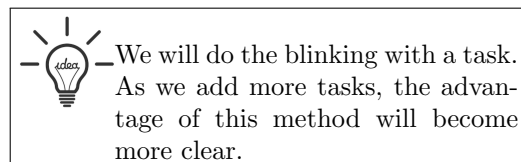
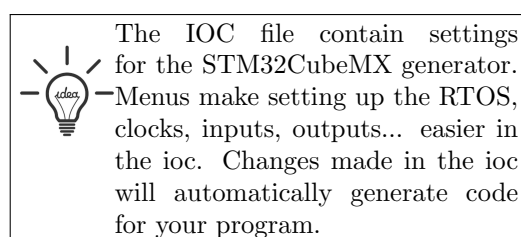


Figure 6: Turning on Project Explorer



3. Expand the project and double click on the ioc file.



4. Configure the RTOS in the .ioc using the menu setting shown below. FREERTOS can be found in the Middleware and Software Packs group. Select CMSIS_V1.

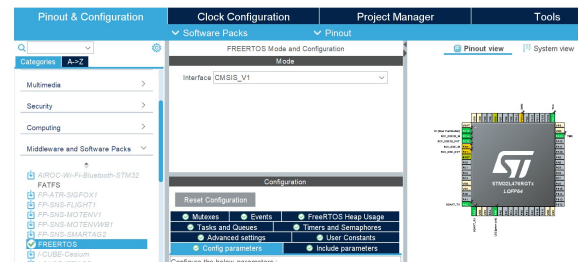


Figure 7: Turning on FreeRTOS

5. Select the Advanced RTOS settings using the menu setting shown below. Enable USE_NEWLIB_REENTRANT.

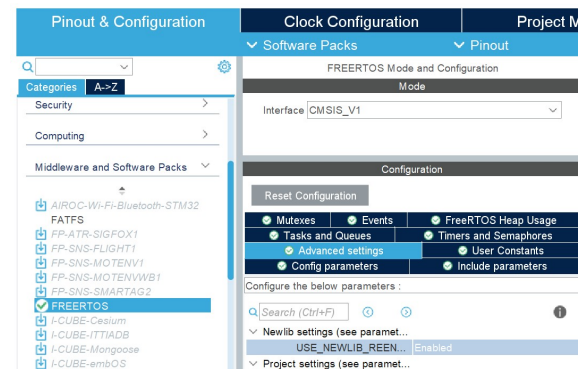


Figure 8: System Core, Reentrant

6. Change the Time Base SYS, found in the System Core group, in the .ioc using the menu setting shown below. Change the Time Base to TIM1.

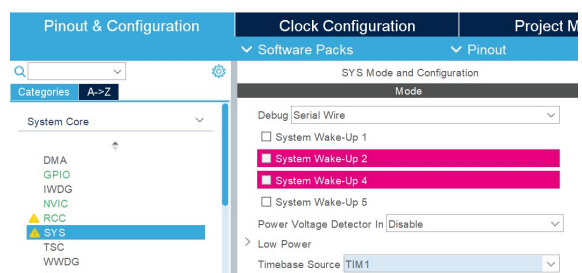



Figure 9: System Core, change Timebase to TIM1



A default task will be created for you. Reentrant functions are said to be 'thread safe' because more than one thread can call the function without risk of data corrupted. Again, with only one task, not that important yet. The SysTick interrupt will be used by FreeRTOS and run at the slow rate of 1ms. HAL (the micro hardware) is moved to a new timer, TIM1 in this lab.

7. Click the yellow gear icon (again) in the center of the task bar.
8. This time, search for StartDefaultTask in main.c. Add the HAL commands as shown below.

```

1  /* USER CODE END
   Header_StartDefaultTask */
2  void StartDefaultTask(void const *
   argument)
3  {
4      /* USER CODE BEGIN 5 */
5      /* Infinite loop */
6      for (;;)
7      {
8          HAL_GPIO_TogglePin(GPIOA,
9          GPIO_PIN_5);
10         osDelay(500);
11     }
12     /* USER CODE END 5 */
13 }

```

9. To run the program, click the green bug in the center of the task bar, shown in figure 4.
10. To resume the program, click the play button on the left side of the task bar, shown in figure 5.
11. Verify a green LED is blinking on and then off every second, as shown in figure 10. The on board user LED is labeled LD2.

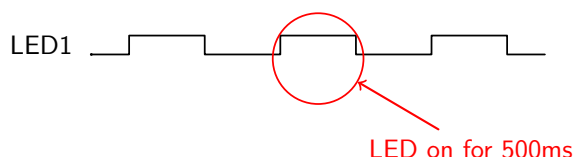


Figure 10: LED on and off, 1 second period


12. Find the following commands in main and take screen shots: 1) osThreadId defaultTaskHandle, 2) osThreadDef(defaultTask, StartDefaultTask, osPriorityNormal, 0, 128), 3) osKernelStart() (including comment "We should never get here..."). Copy screen-shots of the

commands into Word.


13. Change the task so the LED stays on for 1 second and turns off for a half a second.
14. For a sign-off, show your flashing LED and the screen-shots.




HINT: Use the HAL_GPIO_WritePin command and two HAL_DELAY commands.




osThreadId creates a thread (task) identifier. osThreadDef creates the default thread and osKernelStart() starts the scheduler.



IMPORTANT: FreeRTOS isn't a background task; once osKernelStart() is called, FreeRTOS runs its tasks. Hence, after the scheduler is started, all subsequent operations are controlled by the RTOS. The while (1) in main will never run.



The osDelay(1000); function can be replaced with the FreeRTOS native function, vTaskDelay(1000);. The commands will do the same 1 second delay.



Test Fact: The delay function puts the task into a timed *suspension* state. The transition from *suspended* to *ready* occurs when the designated delay time elapses. As this is the only task in the system, it is immediately set *running*.

Section 4.2 Blink two LEDs, two RTOS Tasks

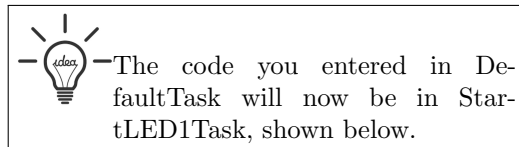
1. Expand the project and double click on the ioc file.
2. Go to Middleware and Software Packs → FREERTOS → and select Tasks and Queues.
3. Double click on the defaultTask and change defaultTask to LED1Task and change StartDefaultTask to StartLED1Task as shown in Fig-

Figure 11. Then click OK.

Edit Task	
Task Name	LED1Task
Priority	osPriorityNormal
Stack Size (Words)	128
Entry Function	StartLED1Task
Code Generation Option	Default
Parameter	NULL
Allocation	Dynamic
Buffer Name	NULL
Control Block Name	NULL
<div>OK Cancel</div>	

Figure 11: LED1 Task settings

- Click on Add and make a second task and change the Task Name and Entry Function to LED2Task and StartLED2Task, respectively. The other defaults are fine, click OK.
- Find PA10 on the chip diagram. Click and select GPIO_Output. Right click on the pin and change the label to LED2.
- Click the gear to generate code.



```

1  /* USER CODE END Header_StartLED1Task
2  */
3  void StartLED1Task(void const *
4    argument)
5  {
6    /* USER CODE BEGIN 5 */
7    /* Infinite loop */
8    for (;;)
9    {
10     HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5)
11     ;
12     osDelay(500);
13   }
14   /* USER CODE END 5 */

```

- Build the resistor/LED circuit in Figure 12. The cathode of the LED is connected to ground (GND in connector CN6) and the resistor is connected to PA10 on CN9.

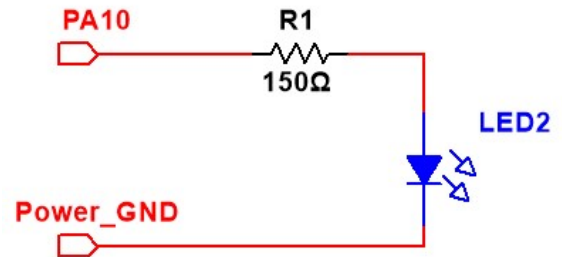


Figure 12: LED2 Circuit

- Find StartLED2Task and make it blink at a different rate of your choosing. The LED is connected to PA10 which is GPIOA, pin 10, also labeled D2 on the board (third from bottom).

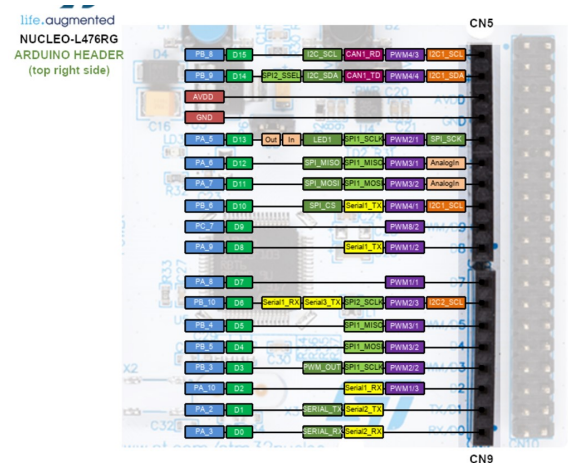


Figure 13: CN9 Connector showing PA10, also called D2

- Verify both LEDs are blinking and get a sign off.



Common problems that cause the LED not to light: 1) LED in backwards. The LED flat side with the shorter lead should be connected ground. 2) Wrong pin selected in the ioc. The board will send Target is not responding, retrying... if it is unable to control the output. 3) Using HAL_Delay instead of osDelay. The tasks are moved to Pending when they execute an osDelay. HAL_Delay leaved the task Active, preventing the second task from moving to Active.

Section 5 Sign-offs

Name: _____

Section 4.1: Task Blink with command screen shots.

Date

Section 4.2: Two tasks, each blink an LED.

Date

Section 6 Report

By the start of next lab, upload to MyCourses Dropbox in a single document containing.

1. A scan or picture of this sign-off sheet.
2. Screen shots of task commands.

Reports submitted **one week late** will receive a maximum grade of 70%. Reports submitted more than three weeks late will receive a maximum grade of 40%.