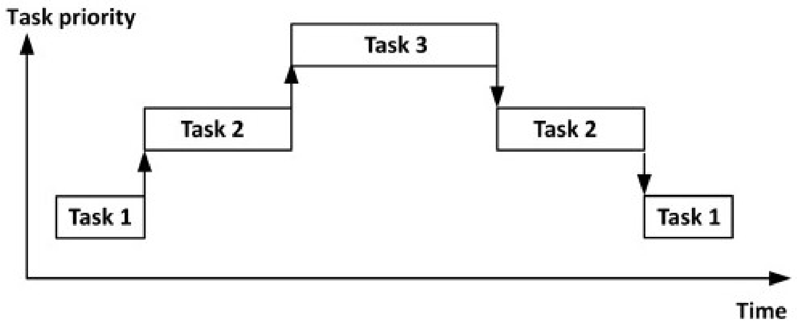
Lab 2 Preemptive Scheduling



Real Time Operating Systems - Dogan Ibrahim, in Arm-Based Microcontroller Multitasking Projects, 2021

Section 1 Lab Objectives

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

1. Understand HAL blocking commands and RTOS commands putting a state in WAITING.
2. Understand preemptive scheduling.

Section 2 PreLaboratory Preparation

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

Equipment

1. Bring your breadboard and wire kit.
2. Bring your STM32L476 board.

On Line Learning

1. Watch at least the first ten minutes of *LIVE Shop Talk 35: Understanding Blocking and Non-Blocking Operations and Functions* `https://www.youtube.com/watch?v=Vf7URLWvUWw&t=562s`

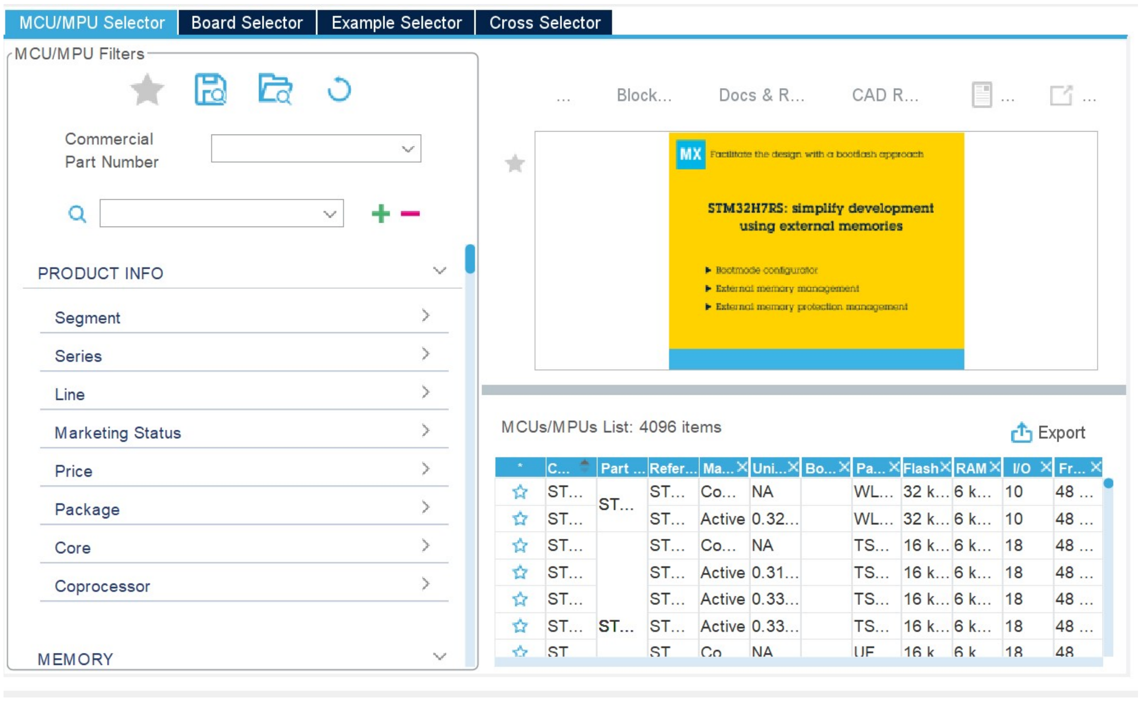
Preparation for the Prelab Quiz

The quiz will be available in lab for the first ten (10) minutes of your laboratory session. You may use any of your prelab preparation as a reference while taking the quiz.

Section 3 Blocking Commands

Section 3.1 Project Creation - Repeated from Lab 1

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


MCU/MPU Selector
Board Selector
Example Selector
Cross Selector
MCU/MPU Filters
Commercial Part Number
PRODUCT INFO
Segment
Series
Line
Marketing Status
Price
Package
Core
Coprocessor
MEMORY
... Block... Docs & R... CAD R... ... ...
MX Facilitate the design with a bootflash approach
STM32H7RS: simplify development using external memories
Bootmode configurator
External memory management
External memory protection management
Export

MCUs/MPUs List: 4096 items

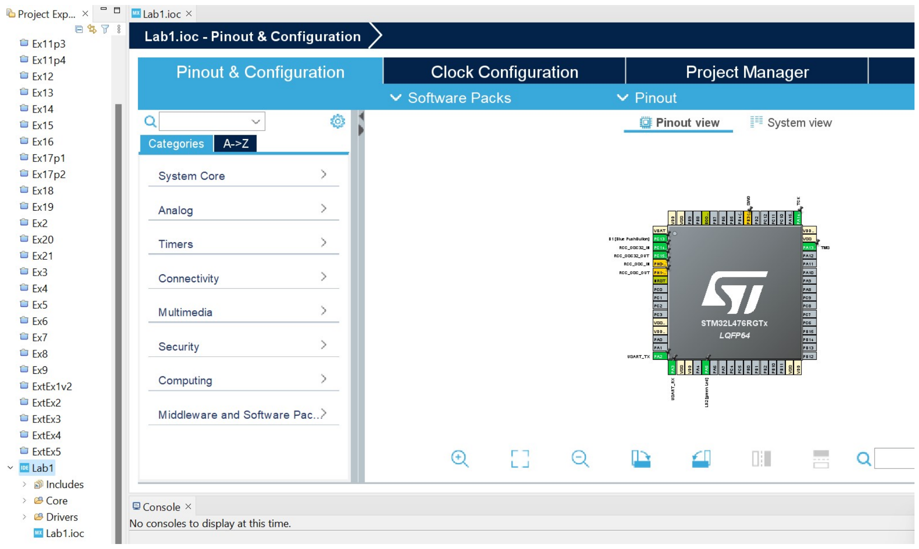
*	C...	Part ...	Refer...	Ma...	Uni...	Bo...	Pa...	Flash	RAM	I/O	Fr...
☆	ST...	ST...	ST...	Co...	NA		WL...	32 k...	6 k...	10	48 ...
☆	ST...		ST...	Active	0.32...		WL...	32 k...	6 k...	10	48 ...
☆	ST...		ST...	Co...	NA		TS...	16 k...	6 k...	18	48 ...
☆	ST...		ST...	Active	0.31...		TS...	16 k...	6 k...	18	48 ...
☆	ST...		ST...	Active	0.33...		TS...	16 k...	6 k...	18	48 ...
☆	ST...	ST...	ST...	Active	0.33...		TS...	16 k...	6 k...	18	48 ...
☆	ST		ST	Co	NA		UF	16 k	6 k	18	48

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.



No consoles to display at this time.

Figure 2: Resulting Project Screen

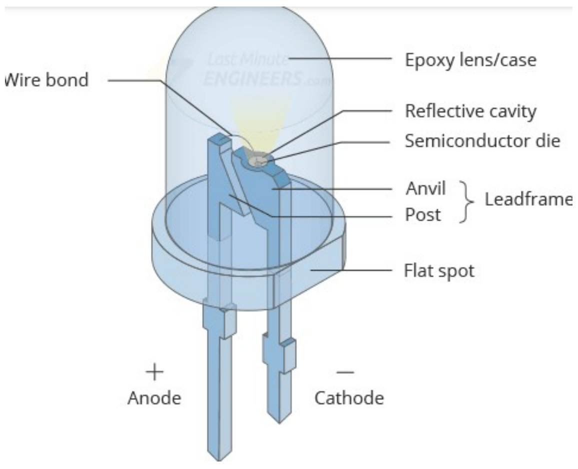


Figure 3: LED showing flat spot

5. Find PA10 on the chip diagram. Click and select GPIO_Output. Right click on the pin and change the label to LED2.

> Most people find the cathode of an LED by finding the short leg which can be a problem if the LED is trimmed. I find it better to find the flat spot, shown in figure 3 on the cathode side which reminds me of the vertical line on the cathode side of an LED symbol.

6. Rebuild (or use) the resistor/LED circuit in Figure 5 from last lab. The cathode of the LED is connected to ground (GND in connector CN6) and the resistor is connected to PA10 on CN9.

LEDs	Port	Pin	CN9 Label
1	A	5	on board
2	A	10	D2
other	options		
3	B	5	D4
4	B	10	D6
5	A	8	D7

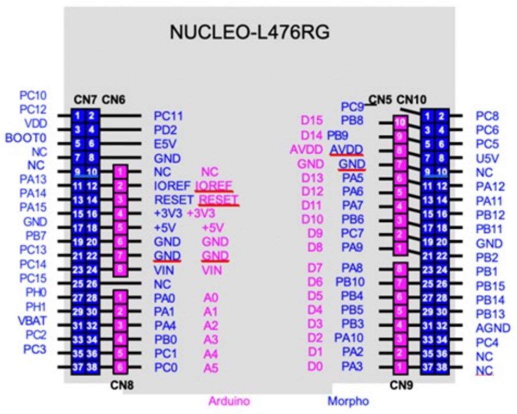

NUCLEO-L476RG
CN7 CN6
CN5 CN10
PC10 PC12 VDD BOOT0 NC NC PA13 PA14 PA15 GND PB7 PC13 PC14 PC15 PH0 PH1 VBAT PC2 PC3
PC11 PD2 E5V GND NC IOREF RESET +3V3 +5V GND GND VIN NC PA0 PA1 PA4 PB0 PC1 PC0
NC IOREF RESET +3V3 +5V GND GND VIN A0 A1 A2 A3 A4 A5
PC9 PB8 PB9 AVDD GND PA5 PA6 PA7 PB6 PC7 PA9 PA8 PB10 PB4 PB5 PB3 PA10 PA2 PA3
D15 D14 AVDD GND D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0
PC8 PC6 PC5 U5V NC PA12 PA11 PB12 PB11 GND PB2 PB1 PB15 PB14 PB13 AGND PC4 NC NC
CN8
CN9
Arduino
Morpho

Figure 4: NucleoL476RG Pinout

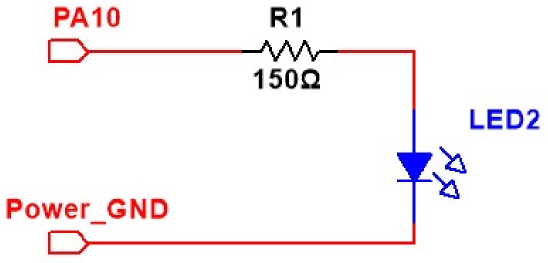


Figure 5: LED2 Circuit

Section 3.2 Blink two LEDs in main while(1) loop

1. *The end of Section 3 has a summary table you must complete for a sign-off.*
2. Click the yellow gear icon in the center of the task bar.
3. Search for while (1) in main.c. Add the HAL commands as shown below. The on board and off board LEDs will blink as shown in figure 6

```c
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
  /* turn on-board LED on/off, every 4 s. */
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
  HAL_Delay(2000); /* HAL_Delay is a blocking command */
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
  HAL_Delay(2000);
  /* turn the bread board LED on/off, every 2 s. */
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, GPIO_PIN_SET);
  HAL_Delay(1000);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, GPIO_PIN_RESET);
  HAL_Delay(1000);
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
}

```

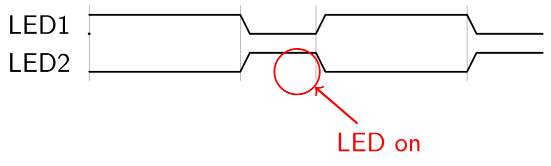


Figure 6: On main loop

4. Connect the computer to the board using a USB cord and run the program by clicking the green bug in the center of the task bar.
5. Resume the program by clicking the play button on the left side of the task bar.
6. Verify that the on-board LED blinks on and then off in four seconds *and then* the off-board LED blinks on and off for 2 seconds. Sequential, as you would suspect.

Think about the blocking delay (HAL_DELAY()). Think about how the delay has to complete before the next command can start in the main while loop. The two LEDs are not able to flash simultanously.

Section 3.3 Blink two LEDs using RTOS using one task

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

Commenting the code in the main while loop is optional because once the RTOS is turned on (below), the scheduler will be called and take full control. The while loop in main will never execute.

2. Turn on FreeRTOS and create two tasks by:

(a) Expanding the project and double clicking on the ioc file.

(b) Changing the Timebase SYS to TIM1. Timebase can be found in: System Core → SYS → Timebase Source (bottom of Mode window).

(c) Turning on FreeRTOS (found in Middleware and Software Packs), select CMSIS_V1.

(d) Selecting the Advanced RTOS settings and enabling USE_NEWLIB_REENTRANT.

(e) Go to FREERTOS → Tasks and Queues. Double click on the defaultTask and change defaultTask to LEDTask1 and change StartDefaultTask to StartLEDTask1 . Then click OK.

(f) Use Add to create LEDTask2 like last lab.

(g) Click the gear to generate code.

(h) In main.c, find StartLEDTask1 and use it to flash the two LEDs just like you did in the while loop (the code is the same).

```c
/* USER CODE END Header_StartLEDTask1 */
void StartLEDTask1(void const * argument)
{
/* USER CODE BEGIN 5 */
/* Infinite loop */
for(;;)
{
/* turn on-board LED on/off, every 4 s. */
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
HAL_Delay(2000); /* HAL_Delay is a blocking command */
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
HAL_Delay(2000);
/* turn the bread board LED on/off, every 2 s. */
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, GPIO_PIN_SET);
HAL_Delay(1000);
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, GPIO_PIN_RESET);
HAL_Delay(1000);
}
/* USER CODE END 5 */
}

```

(i) To run the program, click the green bug in the center of the task bar.

(j) To resume the program, click the play button on the left side of the task bar.

(k) Verify that the results are the same.

(l) Change the delays to osDelay (instead of HAL_Delay). Verify that the results are still the same.

We must use both tasks to get parallel operation.

Section 3.4 Blink two LEDs using RTOS using two tasks

(a) Change StartLEDTask1 so it flashes one LED as shown below.

```c
      /* USER CODE END Header_StartLEDTask1 */
      void StartLEDTask1(void const * argument)
      {
        /* USER CODE BEGIN 5 */
        /* Infinite loop */
        for(;;)
        {
          HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
          osDelay(2000); /* osDelay is also a blocking command */
          HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
```

```
            osDelay(2000);
        }
        /* USER CODE END 5 */
    }

```

(b) Find StartLEDTask2 and write similar code to make PA10 blink on/off every 2 sec.

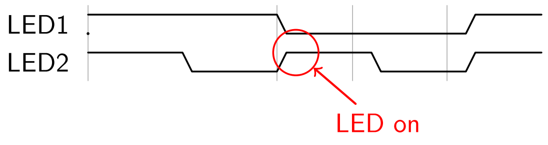


Figure 7: Two RTOS tasks running in parallel

3. To run the program, click the green bug in the center of the task bar.

4. To resume the program, click the play button on the left side of the task bar.

5. Verify both LEDs are blinking in parallel, like the timing diagram in figure 7.

6. Change *ALL* the osDelay commands to HAL_Delay.

7. Fill out the table below for a sign-off.

Description	Were the LEDs ever on at the same time?	Did both LEDs blink?	Quick description of LED operation
Section 3.2: Two LEDs in while loop	Yes/No	Yes/No	
Section 3.3: Two LEDs in one RTOS Task	Yes/No	Yes/No	
Section 3.4: Two LEDs in two RTOS Tasks using os-Delay	Yes/No	Yes/No	
Section 3.4: Two LEDs in two RTOS Tasks using ALL HAL_Delay	Yes/No	Yes/No	

Section 4 Preemptive Scheduling

Section 4.1 Two blinking LED again

1. Go to FREERTOS → Tasks and Queues and verify that LEDTask1 priority is osPriorityNormal (if it is not, set it to osPriorityNormal). Change LEDTask2 priority to osPriorityAboveNormal. Then click OK.

2. In main.c make LEDTask2 use osDelays. Leave LEDTask1 with HAL_Delays.

3. Click the gear to generate code.

4. Predict the operation and run the code.

5. Enter your observations and comments in the table at the end of this section.

Section 4.2 One blinking LED

6. Go to FREERTOS → Tasks and Queues and change LEDTask1 priority to osPriorityAboveNormal and change LEDTask2 priority to osPriorityNormal. Then click OK.
7. Click the gear to generate code.
8. Predict the operation and run the code.
9. Enter your observations and comments in the table at the end of this section.
10. Fill out the table below for a sign-off.

Description	Were the LEDs ever on at the same time?	Did both LEDs blink?	Quick description of LED operation
Section 4.1: Task1 Normal Priority with HAL_Delay; Task2 AboveNormal Priority with osDelay	Yes/No	Yes/No	
Section 4.2: Task1 AboveNormal Priority with HAL_Delay; Task2 Normal Priority with osDelay	Yes/No	Yes/No	

Section 5 Sign-offs

Name:________________________________

Section 3.4: LED blinking summary table completed.	Date .
Section 4.2: Priority allows two LEDs to blink, even with HAL_Delay.	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. The completed tables.

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%.