ECE 5780/6780 Lab 1 – Spring 2024

Due Date: Monday, January 22 (20 points)

Objectives

The purpose of this lab is to become familiar with the process of creating a multi-task microcontroller application using FreeRTOS, Keil µVision, and the STM32L476 Nucleo-64 Board.

Overview

In this lab you will create, download, execute, and debug a multi-task C program that controls the LED on the STM32L476 Nucleo-64 Board using the on-board user push button.

Preparation

You will need your STM32L476 Nucleo-64 Board.

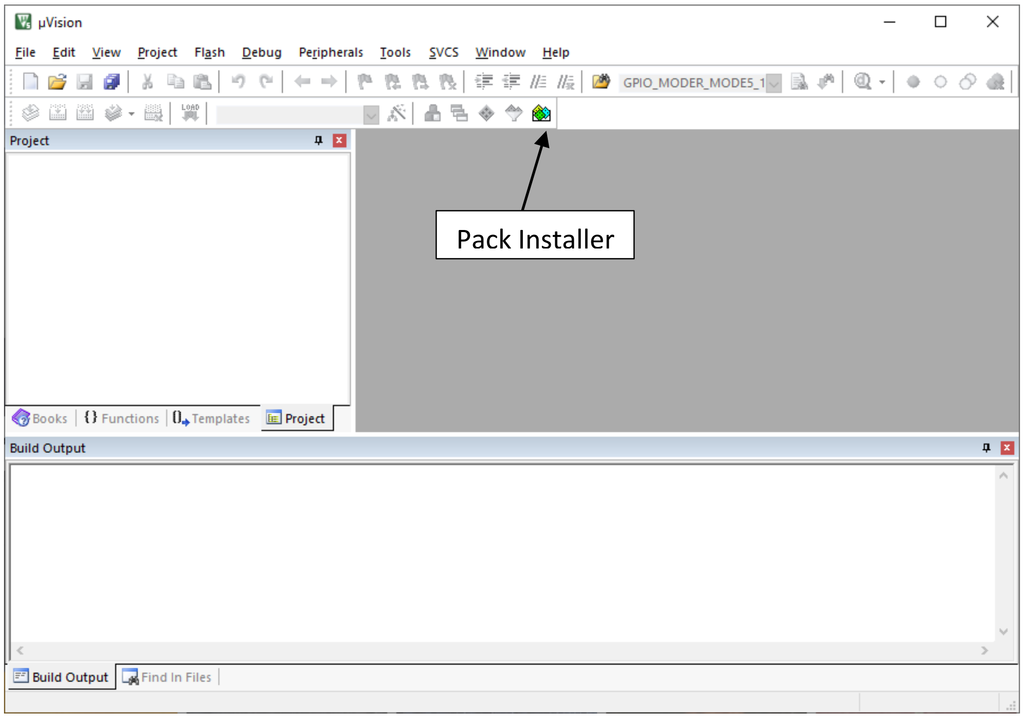
Procedure

1. You may need to install the ARM::CMSIS-FreeRTOS pack in Keil. If so, follow these steps.

   a. Start Keil µVision.

   b. Make sure there are no projects open.

   c. Start the **Keil Pack Installer**. See figure below.


µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
GPIO_MODER_MODES_1
Project
Pack Installer
Books {} Functions Templates Project
Build Output
Build Output Find In Files

   d. When the installer opens, it will run package updates. This could take several minutes, depending on the number of updates needed. Wait until updates are complete (See status bar at bottom of window).

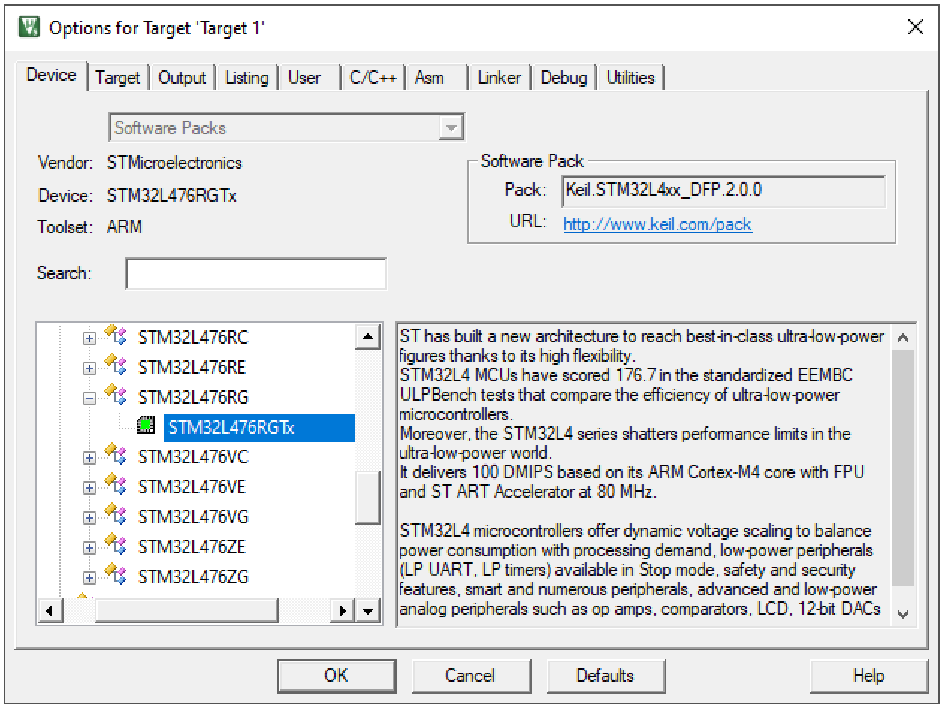
   e. In the right-side window of the Pack Installer, expand the Generic menu and look for **ARM::CMSIS-FreeRTOS**. If it is marked "Up to data" nothing else needs to be done. If it is marked "Install+", click on the button and wait for installation to complete.

2. Create a new Keil µVision project that uses FreeRTOS and targets the STM32L476 Nucleo-64 Board.

   a. Start Keil µVision.

   b. Select **Project -> New µVision Project** and give your project a name and a path that make sense to you.

c. Select the appropriate microcontroller. Our Nucleo boards use the STMicroelectronics STM32L476RGTx. See the figure below.


Options for Target 'Target 1'
Device | Target | Output | Listing | User | C/C++ | Asm | Linker | Debug | Utilities
Software Packs
Search:
STM32L476RC
STM32L476RE
STM32L476RG
STM32L476VC
STM32L476VE
STM32L476VG
STM32L476ZE
STM32L476ZG
OK | Cancel | Defaults | Help

Vendor: STMicroelectronics
Device: STM32L476RGTx
Toolset: ARM

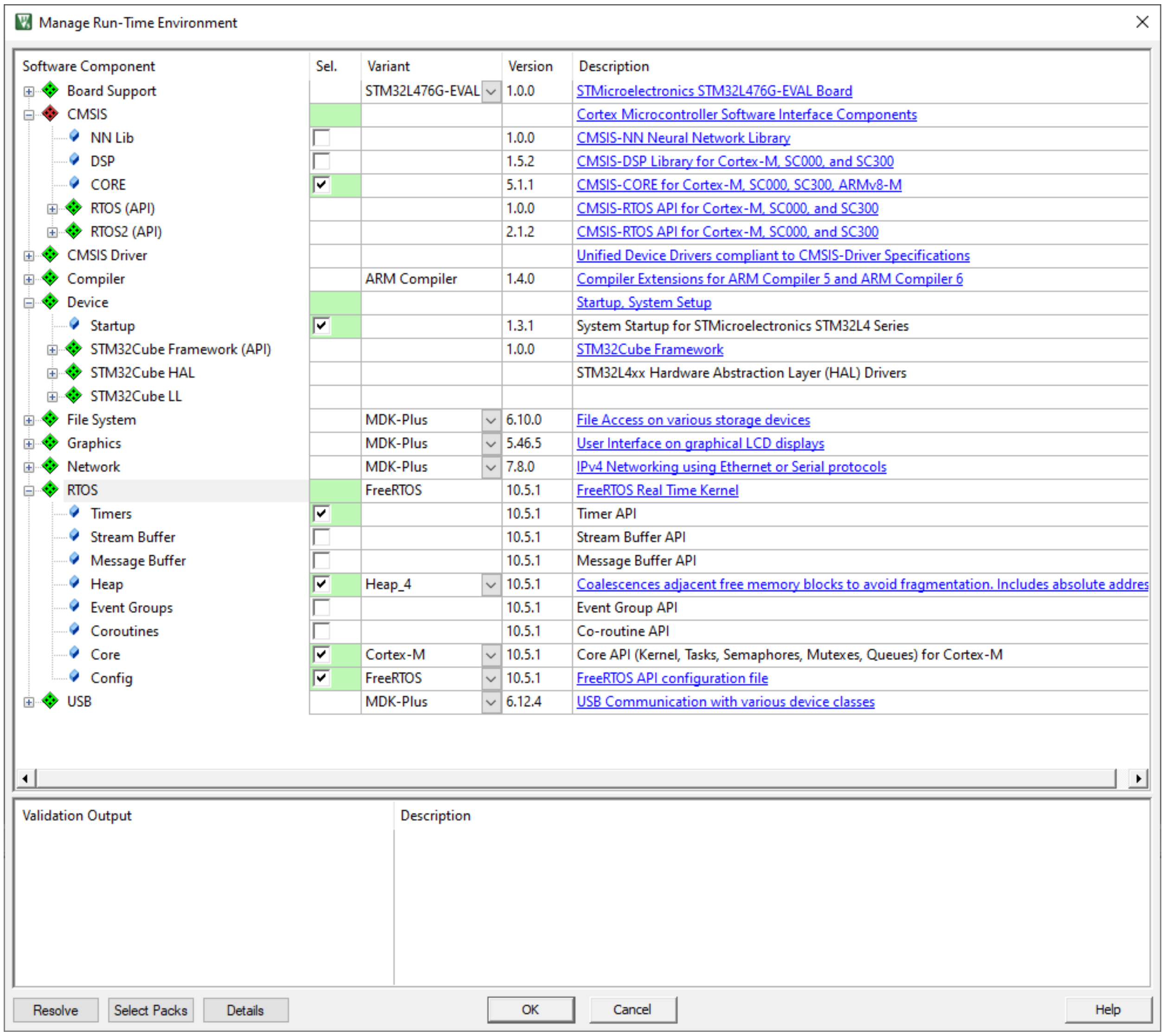
Software Pack
Pack: Keil.STM32L4xx_DFP.2.0.0
URL: http://www.keil.com/pack

STM32L476RGTx

ST has built a new architecture to reach best-in-class ultra-low-power figures thanks to its high flexibility.
STM32L4 MCUs have scored 176.7 in the standardized EEMBC ULPBench tests that compare the efficiency of ultra-low-power microcontrollers.
Moreover, the STM32L4 series shatters performance limits in the ultra-low-power world.
It delivers 100 DMIPS based on its ARM Cortex-M4 core with FPU and ST ART Accelerator at 80 MHz.

STM32L4 microcontrollers offer dynamic voltage scaling to balance power consumption with processing demand, low-power peripherals (LP UART, LP timers) available in Stop mode, safety and security features, smart and numerous peripherals, advanced and low-power analog peripherals such as op amps, comparators, LCD, 12-bit DACs

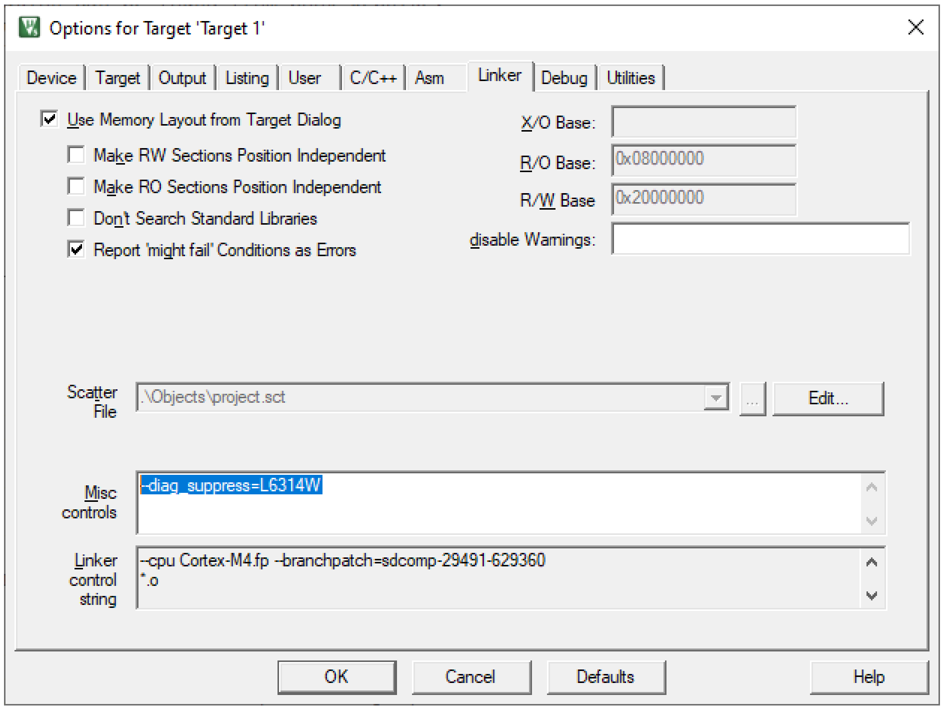
d. After clicking “OK” the **Manage Run-Time Enivronment** page appears. This is where we specify that the project will use the FreeRTOS operating system. Make the selections shown on the screenshot below. Notice that we are specifying the use of FreeRTOS, enabling timers, setting up the heap, and instructing the OS to begin on system startup. Click “OK” when you are done.



Manage Run-Time Environment

Software Component	Sel.	Variant	Version	Description
Board Support		STM32L476G-EVAL	1.0.0	STMicroelectronics STM32L476G-EVAL Board
CMSIS				Cortex Microcontroller Software Interface Components
NN Lib	☐		1.0.0	CMSIS-NN Neural Network Library
DSP	☐		1.5.2	CMSIS-DSP Library for Cortex-M, SC000, and SC300
CORE	✓		5.1.1	CMSIS-CORE for Cortex-M, SC000, SC300, ARMv8-M
RTOS (API)			1.0.0	CMSIS-RTOS API for Cortex-M, SC000, and SC300
RTOS2 (API)			2.1.2	CMSIS-RTOS API for Cortex-M, SC000, and SC300
CMSIS Driver				Unified Device Drivers compliant to CMSIS-Driver Specifications
Compiler		ARM Compiler	1.4.0	Compiler Extensions for ARM Compiler 5 and ARM Compiler 6
Device				Startup, System Setup
Startup	✓		1.3.1	System Startup for STMicroelectronics STM32L4 Series
STM32Cube Framework (API)			1.0.0	STM32Cube Framework
STM32Cube HAL				STM32L4xx Hardware Abstraction Layer (HAL) Drivers
STM32Cube LL				
File System		MDK-Plus	6.10.0	File Access on various storage devices
Graphics		MDK-Plus	5.46.5	User Interface on graphical LCD displays
Network		MDK-Plus	7.8.0	IPv4 Networking using Ethernet or Serial protocols
RTOS		FreeRTOS	10.5.1	FreeRTOS Real Time Kernel
Timers	✓		10.5.1	Timer API
Stream Buffer	☐		10.5.1	Stream Buffer API
Message Buffer	☐		10.5.1	Message Buffer API
Heap	✓	Heap_4	10.5.1	Coalescences adjacent free memory blocks to avoid fragmentation. Includes absolute addres
Event Groups	☐		10.5.1	Event Group API
Coroutines	☐		10.5.1	Co-routine API
Core	✓	Cortex-M	10.5.1	Core API (Kernel, Tasks, Semaphores, Mutexes, Queues) for Cortex-M
Config	✓	FreeRTOS	10.5.1	FreeRTOS API configuration file
USB		MDK-Plus	6.12.4	USB Communication with various device classes

e. Click **Project->Options for Target 'Target 1'**, select the **Linker** tab, and make sure **Use Memory Layout from Target Dialog** is selected. The linker will fail otherwise. Also, paste the string "--diag_suppress=L6314W" into the **Misc controls** text box as shown.



Options for Target 'Target 1'

Device | Target | Output | Listing | User | C/C++ | Asm | Linker | Debug | Utilities

[✓] Use Memory Layout from Target Dialog
[ ] Make RW Sections Position Independent
[ ] Make RO Sections Position Independent
[ ] Don't Search Standard Libraries
[✓] Report 'might fail' Conditions as Errors

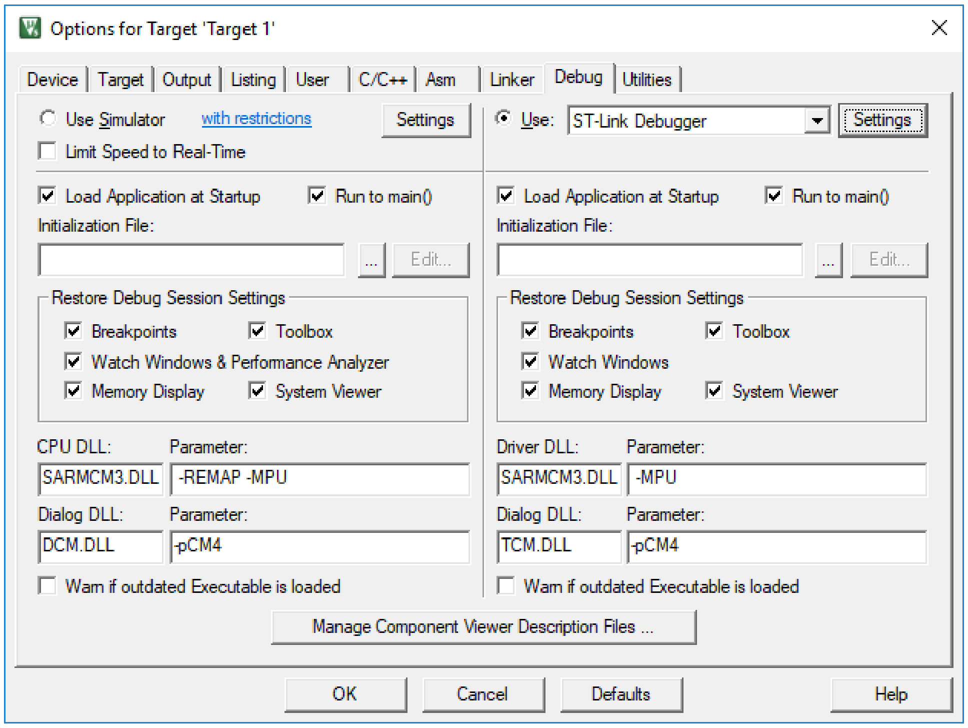
X/O Base:
R/O Base: 0x08000000
R/W Base 0x20000000
disable Warnings:

Scatter File: .\Objects\project.sct

Misc controls: --diag_suppress=L6314W

Linker control string: --cpu Cortex-M4.fp --branchpatch=sdcomp-29491-629360 *.o

f. Still in the Options dialog box, click on the **Debug** tab, select **Use** in the right-hand column, then select **ST-Link Debugger** from the drop-down menu.



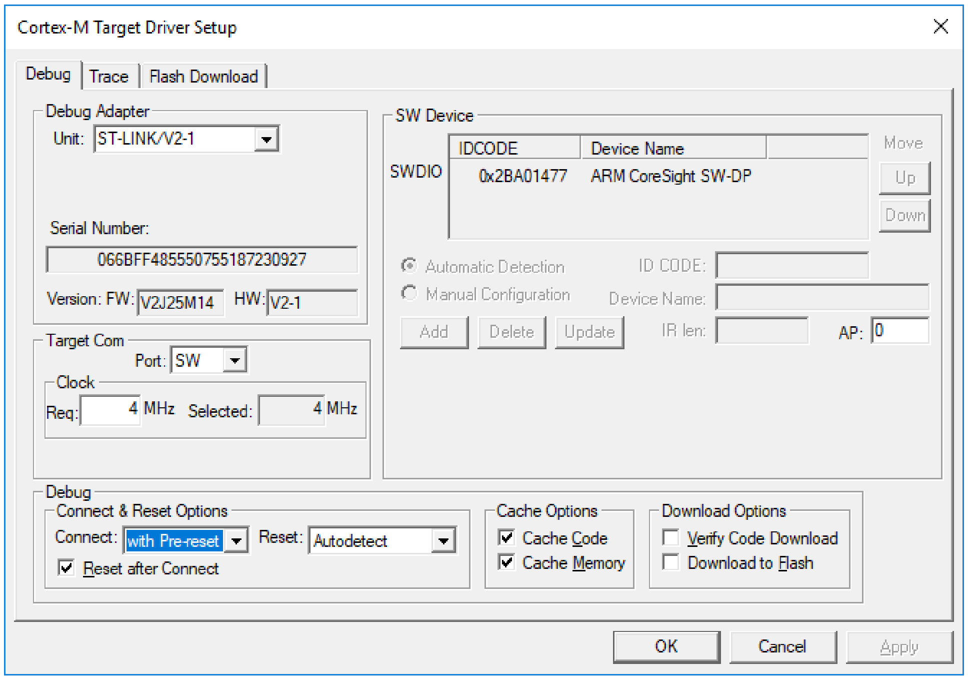
Options for Target 'Target 1'

Device | Target | Output | Listing | User | C/C++ | Asm | Linker | Debug | Utilities

( ) Use Simulator with restrictions
[ ] Limit Speed to Real-Time
[✓] Load Application at Startup [✓] Run to main()
Initialization File:
Restore Debug Session Settings
[✓] Breakpoints [✓] Toolbox
[✓] Watch Windows & Performance Analyzer
[✓] Memory Display [✓] System Viewer
CPU DLL: SARMCM3.DLL Parameter: -REMAP -MPU
Dialog DLL: DCM.DLL Parameter: -pCM4
[ ] Warn if outdated Executable is loaded

(•) Use: ST-Link Debugger
[✓] Load Application at Startup [✓] Run to main()
Initialization File:
Restore Debug Session Settings
[✓] Breakpoints [✓] Toolbox
[✓] Watch Windows
[✓] Memory Display [✓] System Viewer
Driver DLL: SARMCM3.DLL Parameter: -MPU
Dialog DLL: TCM.DLL Parameter: -pCM4
[ ] Warn if outdated Executable is loaded

g. Click on **Settings** in the top-right corner, select the **Debug** tab, and select **with Pre-reset** in the drop-down menu of the **Connect & Reset Options** section (bottom left). Then click **OK** until all pop-ups are closed.



Cortex-M Target Driver Setup

Debug | Trace | Flash Download

Debug Adapter
Unit: ST-LINK/V2-1
Serial Number: 066BFF485550755187230927
Version: FW: V2J25M14 HW: V2-1

Target Com
Port: SW
Clock
Req: 4 MHz Selected: 4 MHz

SW Device

	IDCODE	Device Name
SWDIO	0x2BA01477	ARM CoreSight SW-DP

Move Up Down

Automatic Detection ID CODE:
Manual Configuration Device Name:
Add Delete Update IR len: AP: 0

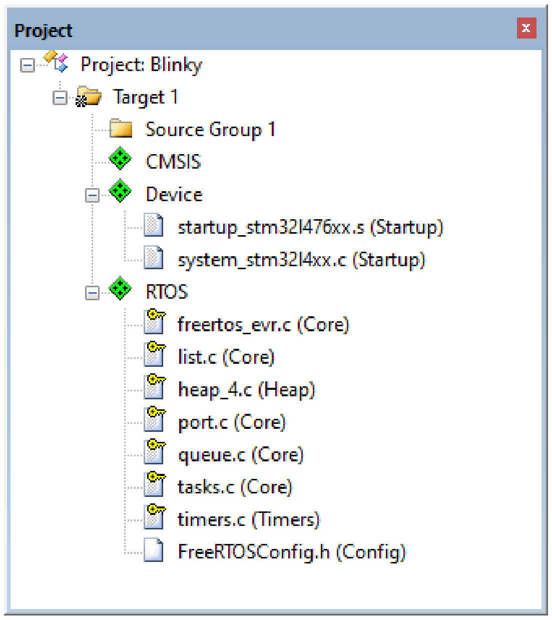
Debug
Connect & Reset Options
Connect: with Pre-reset Reset: Autodetect
☑ Reset after Connect

Cache Options
☑ Cache Code
☑ Cache Memory

Download Options
☐ Verify Code Download
☐ Download to Flash

OK Cancel Apply

h. Notice the Project pane on the left side of **µVision**. Pay special attention to the included files under Device and RTOS. Look through these files and familiarize yourself the contents.



```
Project
Project: Blinky
  Target 1
    Source Group 1
    CMSIS
    Device
      startup_stm32l476xx.s (Startup)
      system_stm32l4xx.c (Startup)
    RTOS
      freertos_evr.c (Core)
      list.c (Core)
      heap_4.c (Heap)
      port.c (Core)
      queue.c (Core)
      tasks.c (Core)
      timers.c (Timers)
      FreeRTOSConfig.h (Config)
```

   i. Right-click on “Source Group 1”, select “Add new item to group…”, and create a new C File named main.c.

3. Using the concepts discussed in class and found in the FreeRTOS documentation (specifically Chapter 2 of the FreeRTOS Reference Manual), add appropriate functionality to the newly-created main.c file to do the following:
   a. Configure appropriate GPIO pins.
   b. Create a task to control the LED. This task should simply turn the LED on or off based on input from the push button.
   c. Create a second task to monitor the push button. Each time the button is pressed, the LED should toggle between on and off. Use a global variable to share information between the two tasks.
   d. Start the FreeRTOS scheduler.

4. Pass-off
   a. Demonstrate the working system to the instructor, either in-person, via Zoom, or via a recorded video (emailed to the instructor). Make sure your pass-off shows all of the requirements listed in Procedure 3.