CIS 890 -- High-Assurance Systems Spring 2019 Homework: CubeMX and STM32F Discovery Board Due Monday, January 28 at 11:59pm

[Note: These instructions are for students using Windows. If you need to use Mac OSX or Linux, you need to talk to John or Venkat about how to customize the assignment. In general, only Step 2 is different for Mac or Linux.]

Purpose

The purpose of the assignment is to

- Understand the basic concepts of CubeMX, perform the installation of the tool and generate code for STM32F Discovery board.
- Setup a development environment to execute the code generated by CubeMX on the STM32F Discovery board.
- Execute a simple program to blink an LED on the STM32F Discovery board.
- Perform a debug operation on the code to ensure that the code behaves as expected.

Objectives/ Deliverables

At the end of the assignment, you will have generated application code for the STM32F Discovery board in FreeRTOS using CubeMX and written custom code inside of the application to be able to blink an LED on the STM32F Discovery board. You will also have executed the code on the STM32F Discovery board using an Integrated Development Environment (IDE) and performed debugging on the code to ensure appropriate results.

1. CubeMX

STM32CubeMX is a graphical tool that allows a very easy configuration of STM32 microcontrollers and the generation of the corresponding initialization C code through a step-by-step process.

Download

 Go to <u>https://www.st.com/en/development-tools/stm32cubemx.html#getsoftware-scroll</u> and download version 5.0.1 version of CubeMX.
 Note: Version 5.0.1 of CubeMX is the latest version of CubeMX at the time of creating this assignment. All instructions and screenshots that follow are geared towards version 5.0.1.

Creation of a New Project

- After the tool is downloaded, launch the tool and click on File -> New Project.
- Upon selecting **New Project**, the **MCU Selector** window opens up. From amongst the list of MCUs listed, select the **STM32F407VG** board. In order to navigate through the overwhelming list of MCUs shown on the window, check the STM32F4 in the series tab to narrow down the list. It is illustrated in the screenshot below.

Check/Uncheck All	The STM32F	105vv and STM32E	107vv fs	amily incorne	arates high_sneed embedded r	nemories /Flash n	nemory un	to 1 Mhyte I	n to 10	,
STM32F0										_
STM32F1	MCUs List: 201 item	s	+	Display simi	lar items					X
STM32F2								_		
STM32F3	* Part No	Reference	Market.	Unit Price for	. Board	Package	Flash	RAM IO	Freq.	GFX S
STM225	\$1M32F405ZG	STM32F405ZGTX	Active	6.662		LQFP144	1024 KBy	192 KBytes 114	1 168 M.	0.0
STWJ2F4	STM32F407IE	STM32F407IEHx	Active	6.338		UFBGA1	512 kBytes	192 kBytes 140) 168 M	0.0
STM32F7	会	STM32F407IETx	Active	6.338		LQFP176	512 kBytes	192 kBytes 140) 168 M	. 0.0
STM32G0	STM32E407IG	STM32F407IGHx	Active	7.264	STM3240G-EVAL	UFBGA1	1024 kBy	192 kBytes 14/) 168 M	0.0
_		STM32F407IGTx	Active	7.264		LQFP176	1024 kBy	192 kBytes 14) 168 M	. 0.0
STM32H7	☆ STM32F407VE	STM32F407VETx	Active	5.644		LQFP100	512 kBytes	192 kBytes 82	168 M.	0.0
STM32L0	☆ STM32F407VG	STM32F407VGTx	Active	6.57		LQFP100	1024 kBy	192 kBytes 82	168 M.,	0.0
STM3211	☆ STM32F407ZE	STM32F407ZETx	Active	6.107		LQFP144	512 kBytes	192 kBytes 11	168 M.	0.0
	1 STM32F407ZG	STM32F407ZGTx	Active	7.033		LQFP144	1024 kBy	192 kBytes 11-	4 168 M	. 0.0
SIM32L4	STM32F410C8	STM32F410C8Ux	Active	1.502		UFQFPN.	64 kBytes	32 kBytes 36	100 M.	0.0
STM32L4+	\$	STM32F410CBTx	Active	171		LOFP48	128 kBytes	32 kBytes 35	100 M	0.0
CTM22LE	STM32F410CB	OTMOOD 440 CDL IV	Activo	4 74		LICOEDNI	100 kDites	22 kDites 26	100 14	0.0

Figure: MCU Selector on CubeMX

• The board selector window is displayed next. Click on the board displayed as shown in the picture below.

*	Overview	Part No	Туре	Marketing Status	Unit Price (US\$)	Mounted Device
Y		STM32F4DISCOVERY	Discovery	Active	19.89	

Figure: Board Selector on CubeMX

- Click Yes on the dialog box that shows up upon the selection of the board.
- If all the above instructions have been executed correctly, a window that looks like the picture shown below should be displayed.

V

тмз2 🍞	File			
CubeMX Home /	STM32F407VGTx - STM32F4	ADISCOVERY / Untitled - Pino	ut & Configuration	GENERATE CODE
Pinol	ut & Configuration	Clock Configuration	Project Manager	Tools
Options Q Categories A>> System Core Analog Timers Connectivity Multimedia Security Computing Middleware			Pinout view Pinout view Pino	VOD VOD VAD VAD
		• [] Q		× [1873/2]

Figure: Board Configuration for STM32F407VG Board

Generating Code in FreeRTOS for STM32F Discovery Board

CubeMX generates boiler plate code in FreeRTOS which can later be viewed in an IDE to be executed on the STM32F Discovery board. The generation of code is simple and can be achieved by following the instructions given below. There are primarily four things that need to be done.

- A. Enabling FreeRTOS as the middleware and task creation for code generation.
- B. Changing the Timebase Source in the System Core tab.
- C. Assigning a name and selecting the appropriate toolchain for the generated code in the Project Manager tab.
- D. Generating code.

A. Enabling FreeRTOS as the Middleware and Task Creation for Code Generation

- Click on the **Middleware** dropdown on the Pinout & Configuration tab.
- Select **FREERTOS** from the list of available options in the drop-down.
- Check the **Enabled** checkbox that shows up to the right upon the selection of the FREERTOS option from the drop-down list.

Pinout & Configuration	Clock Configuration	
	Additional Softwares 🗸 🗸	Pinout
Options Q ~	FREERTOS Mode and Configuration	
Categories A->Z	Mode	
System Core >	Enabled	
Analog >		
Timers >		
Connectivity >		
Multimedia >		
Security >		
Computing >	Configuration	
Middleware	Keset Configuration O Mutexes O FreeRTOS Heap Usage	
\$	Tasks and Queues Timers and Semaph	ores
FATES	Config parameters Oliciude parameters Olice User t	Jonstants
	Q Search (Crt/+F) ③ ④	0
PDM2PCM USB_DEVICE	 ✓ Versions FreeRTOS version 9.0.0 CMSIS-RTOS version 1.02 	1

Figure: Enabling FreeRTOS in Middleware Tab

- Click on the **Tasks and Queues** option from the Configuration parameters.
- Under **Tasks**, click on the default task and change the name of the task from Default Task to **LEDBlinkingTask** from and the name of the entry function from StartDefaultTask to **StartLEDBlinkingTask**.

Ec	lit Task		\times
	Task Name	LEDBlinkingTask	
	Priority	osPriorityNormal	~
	Stack Size (Words)	128	
Enab	Entry Function	StartLEDBlinkingTask	1
	Code Generation Option	Default	\sim
	Parameter	NULL	
	Allocation	Dynamic	~
	Buffer Name	NULL	
-	Control Block Name	NULL	
eset Ci	exes 📀	FreeRTOS Heap Usage	
⊘ Task Config pa ks	s and Queues rameters 📀 Include p		nores Consta
N Prior BliosPrio	ity Stack Entry F Code . b 128 StartL Default	Param. Allocati Buffe NULL DynamicNULI	erCo L NU

Figure: FreeRTOS Task Creation using CubeMX

B. Changing Timebase Source in the System Core Tab

The default timebase source for the STM32Cube-HAL is SysTick. The reason behind changing it from SysTick to a different timer is that most of the RTOSs (FreeRTOS in our case) force SysTick priority to be the lowest which can be an issue when doing scheduling. While we don't perform complex scheduling tasks in this assignment, it is a good practice to do so to ensure that you don't miss out on doing so for any complex assignments that you might work in future.

- Click on the System Core drop-down in the Pinout & Configuration tab.
- Select the **SYS** option from the list of options in the drop-down list.
- Change the **Timebase Source** from SysTick to TIM1 by selecting it from the drop-down list (TIM1 is used for this demonstration. However, any of the timebase sources other than SysTick can be used).

Pinout & Conf	iguration	Clock Configuration	
		Additional Softwares	✓ Pinout
Options Q	~	SYS Mode and Configuration	
Categories A->Z		Mode	
System Core	~	Debug Serial Wire	~
<u>^</u>		System Wake-Up	
DMA		Timebase Source TIM1	~
GPIO			
NVIC			
A SYS			
WWDG			
Analog	>		
Timers	>	Configuration	
		No configuration available	
Connectivity	>		
Multimedia	>		
Security			
Computing	>		
Middleware	~		
÷			

Figure: Timebase Source Selection from System Core Tab

C. Name Assignment and Toolchain Selection

- Click on the **Project Manager** tab.
- Assign the name of the project to be **FreeRTOS_Exercise** in the project name field.
- Select a toolchain/ IDE from the **Toolchain/ IDE** dropdown. While you could select any toolchain of your choice, the recommendation is to select select **TrueSTUDIO** since that will be the IDE used as explained later in the assignment.

Pinout &	Configuration	Clock Configuration	Project Manager
Project	Project Settings Project Name FreeRTOS_Exercise Project Location C:\Users\margs\OneDrive\Dow	cuments	
Code Generator	Basic Toolchain Folder Location C:\Users\margs\OneDrive\Doo Toolchain / IDE TrueSTUDIO		
Advanced Settings	Linker Settings Minimum Heap Size Minimum Stack Size	0×200 0×400	

Figure: Project Settings on Project Manager Tab

D. Code Generation

In order to generate code in FreeRTOS, click on the **GENERATE CODE** option to the top-right of the window.

File	Window	Help		🥺 F 🖻 🎽 🔆
/GTx - STM32F4	IDISCOVERY / Fre	eRTOS_Exercise.ioc -	Project Manager	GENERATE CODE
ation	Clock Con	figuration	Project Manager	Tools
ttings ime §_Exercise cation margs\OneDrive\Doc	cuments			
n Structure	~ D	o not generate the main()		
Folder Location margs\OneDrive\Doc	cuments\FreeRTOS_Ex	erating user source code		
/ IDE NO	✓ V G	enerate Under Root		
tings	0~200			

Figure: FreeRTOS Code Generation

The code generation step concludes the usage of CubeMX tool for this assignment.

2. Integrated Development Environment (IDE)

In order to execute the code generated by CubeMX, an IDE is required. Depending on the Toolchain/ IDE that you selected in the Project Manager tab while on CubeMX, the corresponding IDE needs to be installed. If you have followed the recommendation from the previous section and selected TrueStudio, install the **Atollic TrueSTUDIO for STM32** IDE.

Download

The Atollic TrueStudio for STM32 IDE can be downloaded from <u>https://atollic.com/resources/download/windows/</u>.

Once the installer is downloaded, go ahead and install the IDE on your machine.

Importing Projects into Atollic TrueSTUDIO IDE

After the install of the IDE is complete, the project that was generated by CubeMX i.e. FreeRTOS_Ex needs to be imported into TrueStudio. The application can then be modified/ executed in the IDE. Atollic TrueSTUDIO is similar to Eclipse IDE and you can use your Eclipse IDE experience to navigate through the IDE if you have any experience with Eclipse prior to this. In order to import the project into TrueStudio IDE,

- Click on File -> Open Projects From File System.
- Click on Directory and select the project FreeRTOS_Ex from the directory on your machine.

Import Projects from File System or Archive System analyzes the content of your fold Import source:	ve :hive ler or archive file to find projects and import them in the IDE.	 Directory
type filter text		Select All
Folder	Import as	Deselect All
Use <u>installed project configurators</u> to: Search for nested projects Detect and configure project natures Working Sets	Browse for Folder × Select the folder to find projects to import • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • •	0 of 0 selected
Add project to working sets Working sets:	Middlewares Src startup Folder: FreeRTOS_Exercise Make New Folder OK Cancel	New Select
	< Back Next >	Finish Cancel

Figure: Project Import into Atollic TrueStudio

• The project will then be imported into your IDE workspace and you will be able to see the folder structure and the code on the IDE as shown below.

🎦 Project Explorer 💥 📃 🚍 🧏 😨 🗢 🗖	c main.c c queue.c c tasks.c c port.c h portmacro.h c ma	in.c 🔀 main.c 🔀	», 🗆 🗖	<u>₽</u> 0 ⊠	🚡 т 🛞 В	T -	E
 DemoProject Exercise 2.2 FreeRTOS_Exercise Officient of the second of the seco	<pre>1 * USER CODE BEGIN Header */ 20/** 20/** 20/** 20/** 20/** 20/** 20/** 20/** 20/** 20/** 20/** 20/** 20/** 20/** 20/*** 20/*** 20/*** 20/**********</pre>	ithout ditions are met: ight notice, opyright notice, he documentation ther romote products ssion. works of this troller or	×	에 mm 에 mm 에 bit 에 hit 에 hit 어 hit	El Jag Responses El Jag Response El J	via via	d d d
KIOS book preliminary exercise EXTOS_Ex.4 EXTOS_Ex.5	Problems 🧔 Tasks 🕒 Console 🛛 📄 Properties 🖓 🗖 🔛 👘 🖓	Build Analyzer S	3 <u>==</u> Static Stack Analy	yzer	PM		E
> 25 RTOS_Ex.7	CDT Build Console [FreeRTOS_Exercise]	Memory Regions	Memory Details				
> 2 RTOS_EX.9		Region RAM	Start address End 0x20000000 0x20	address S 0020000 1	ize 28 KB	Free 109.07 KB	< >
	< · · · · · · · · · · · · · · · · · · ·	<				>	

Figure: Imported Project Code on Atollic TrueSTUDIO for STM32 IDE

This concludes the install and import of projects into the Atollic TrueStudio IDE.

3. LED Blinking Program for STM32F Discovery Board on Atollic TrueSTUDIO IDE

From the project code that was imported into the TrueSTUDIO IDE, open the main.c file that you will find under the Src directory of the project. All the code for the assignment is written in the main.c file and it is a good idea to take a few minutes at this moment and read through the code in the file and get a brief understanding.

For the purposes of the LED blinking program, we will modify the code in the **StartLEDBlinkingTask** task that was created in CubeMX. You can find the code for the StartLEDBlinking task in the main.c file. It will look like this:

void StartLEDBlinkingTask(void const * argument)
{
 /* init code for USB_HOST */
 MX_USB_HOST_Init();
 /* USER CODE BEGIN 5 */
 /* Infinite loop */
 for(;;)
 {
 osDelay(1);
 }
 /* USER CODE END 5 */
}

Provided below are the functions that can be used to perform Read and Write operations to the GPIO on the STM32F Discovery board. Utilize the functions to turn ON and turn OFF an LED.

• HAL_GPIO_WritePin(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin, GPIO_PinState PinState): Set or clear the selected data port bit.

GPIOx,:	where x can be (AH) to select the GPIO peripheral for STM32L4 family
GPIO_Pin,:	specifies the port bit to be written. This parameter can be one of GPIO_PIN_x where x can be (015).

- PinState,:
 specifies the value to be written to the selected bit. This parameter can be one of the GPIO_PinState enum values:

 GPIO_PIN_RESET: to clear the port pin

 GPIO_PIN_SET: to set the port pin

 GPIO_PIN_SET: to set the port pin
 - osDelay(uint32_t millisec): Wait for a time specified in milliseconds before the next statement executes.

Using the functions specified above, write code in the infinite for loop in the **StartLEDBlinkingTask** function to

- Turn ON the Green LED on the board.
- Keep the LED ON for a period of 1000 ms.
- Turn OFF the Green LED on the board.
- Keep the LED OFF for a period of 1000 ms.

The Green LED on the board is a part of the **D** peripheral of the STM32L4 family, the GPIO pin is **12** and the PinState is **GPIO_PIN_SET** to turn ON the LED and **GPIO_PIN_RESET** to turn OFF the LED.

4. Execute and Debug the Code on STM32F Discovery Board

Execute the code on the STM32F Discovery board following the instructions below and perform the task.

- After the code for the LED is written, connect the STM32F Discovery board to your computer.
- Click on the **Run** button on the Atollic TrueSTUDIO IDE and click on **Debug** button. Doing so starts the execution of the program on the STM32F Discovery board in Debug mode.

Note: Debug mode is suitable for the assignment since it is easy to watch the execution of the code that is written.

Place a breakpoint on the line of code after the LED is turned ON and observe the LED in the ON state. Take a picture of the STM32F board with the LED turned ON to include it as part of your deliverables for this assignment.

Deliverables

- The main.c file from your FreeRTOS_Ex project that contains the code in the **StartLEDBlinkingTask function** that makes the green LED blink.
- Pictures of the Green LED turned ON upon hitting the breakpoint placed in code.