

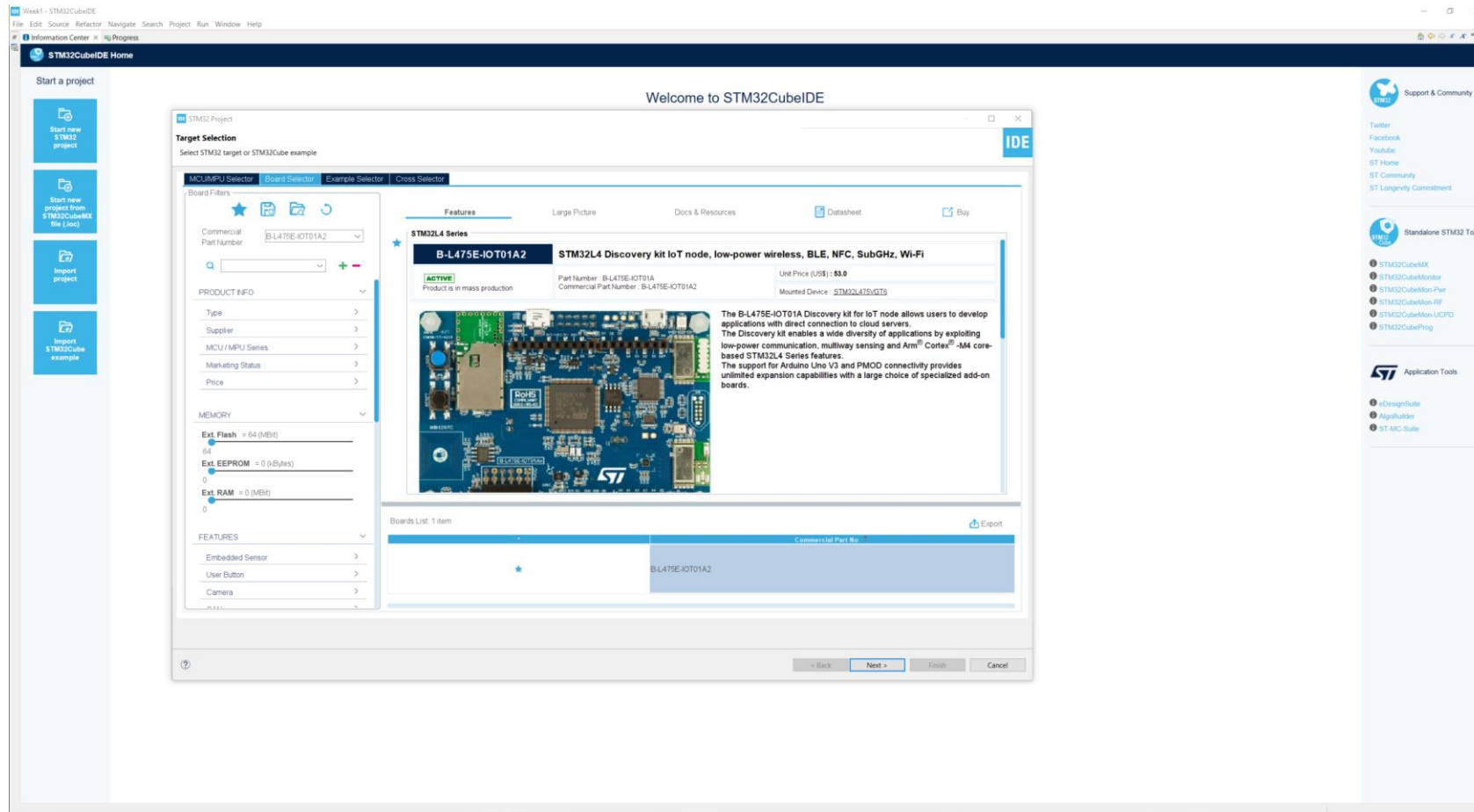
UCSD Embedded RTOS Assignment 4

By

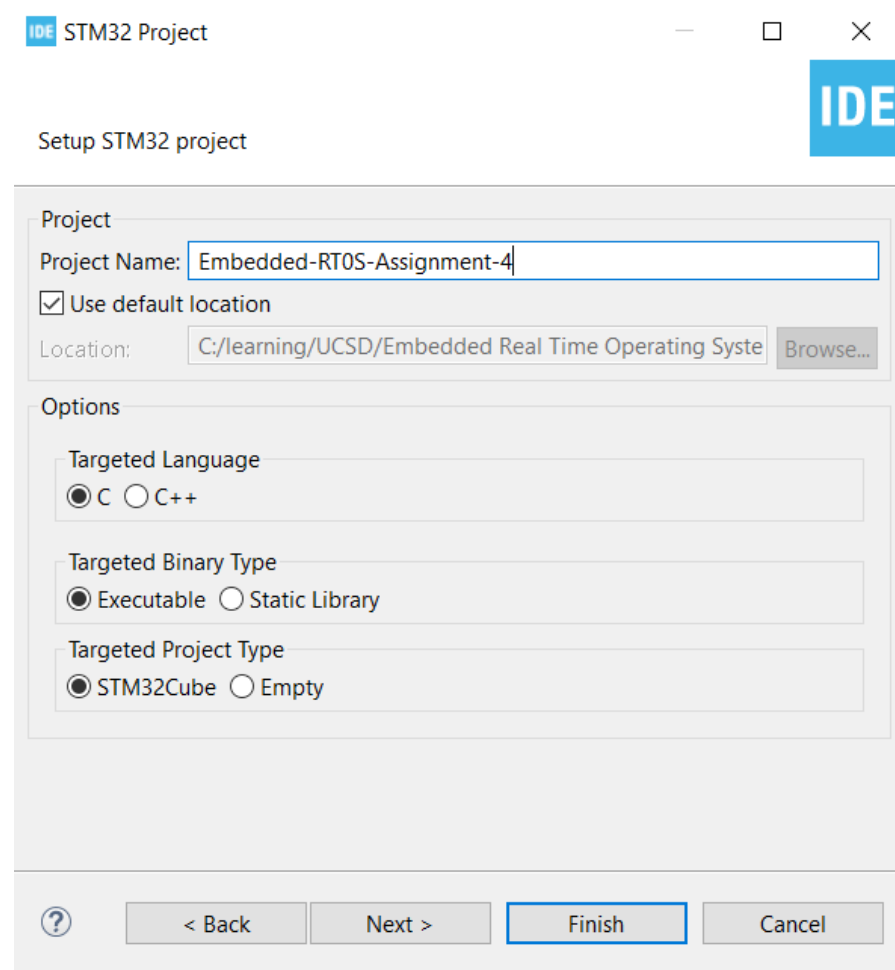
Hsuankai Chang

hsuankac@umich.edu

Step 2. Access board selector and type in the board you use, click Next



Step 3. Enter the project name then click Next



The image shows a 'Setup STM32 project' dialog box from the IDE. The window title is 'IDE STM32 Project'. The dialog is titled 'Setup STM32 project'. It has two main sections: 'Project' and 'Options'. In the 'Project' section, the 'Project Name' field contains 'Embedded-RTOS-Assignment-4'. The 'Use default location' checkbox is checked. The 'Location' field shows 'C:/learning/UCSD/Embedded Real Time Operating Syste' with a 'Browse...' button. The 'Options' section has three sub-sections: 'Targeted Language' with 'C' selected, 'Targeted Binary Type' with 'Executable' selected, and 'Targeted Project Type' with 'STM32Cube' selected. At the bottom, there are buttons for '?', '< Back', 'Next >', 'Finish' (which is highlighted with a blue border), and 'Cancel'.

IDE STM32 Project

Setup STM32 project

Project

Project Name: Embedded-RTOS-Assignment-4

☒ Use default location

Location: C:/learning/UCSD/Embedded Real Time Operating Syste Browse...

Options

Targeted Language

☒ C ☐ C++

Targeted Binary Type

☒ Executable ☐ Static Library

Targeted Project Type

☒ STM32Cube ☐ Empty

? < Back Next > Finish Cancel

Step 4. See the firmware package name and version



The image shows a screenshot of the 'Firmware Library Package Setup' dialog box in the STM32 Project IDE. The dialog box has a title bar with the IDE logo and the text 'STM32 Project'. The main title is 'Firmware Library Package Setup' and the subtitle is 'Setup STM32 target's firmware'. The dialog is divided into three sections: 'Target and Firmware Package', 'Firmware and Software Package Repository', and 'Code Generator Options'. In the 'Target and Firmware Package' section, the 'Target Reference' is 'B-L475E-IOT01A2' and the 'Firmware Package Name and Version' is 'STM32Cube FW_L4 V1.17.2'. In the 'Firmware and Software Package Repository' section, the 'Location' is 'C:\Users\hsuankai.chang\STM32Cube\Repository' and there is a link to 'Firmware Updater'. In the 'Code Generator Options' section, there are three radio buttons: 'Add necessary library files as reference in the toolchain project configuration file', 'Copy all used libraries into the project folder', and 'Copy only the necessary library files'. The 'Finish' button is highlighted with a blue border.

IDE STM32 Project

Firmware Library Package Setup

Setup STM32 target's firmware

Target and Firmware Package

Target Reference: B-L475E-IOT01A2

Firmware Package Name and Version: STM32Cube FW_L4 V1.17.2

Firmware and Software Package Repository

Location:
C:\Users\hsuankai.chang\STM32Cube\Repository

See ['Firmware Updater'](#) for settings related to package installation

Code Generator Options

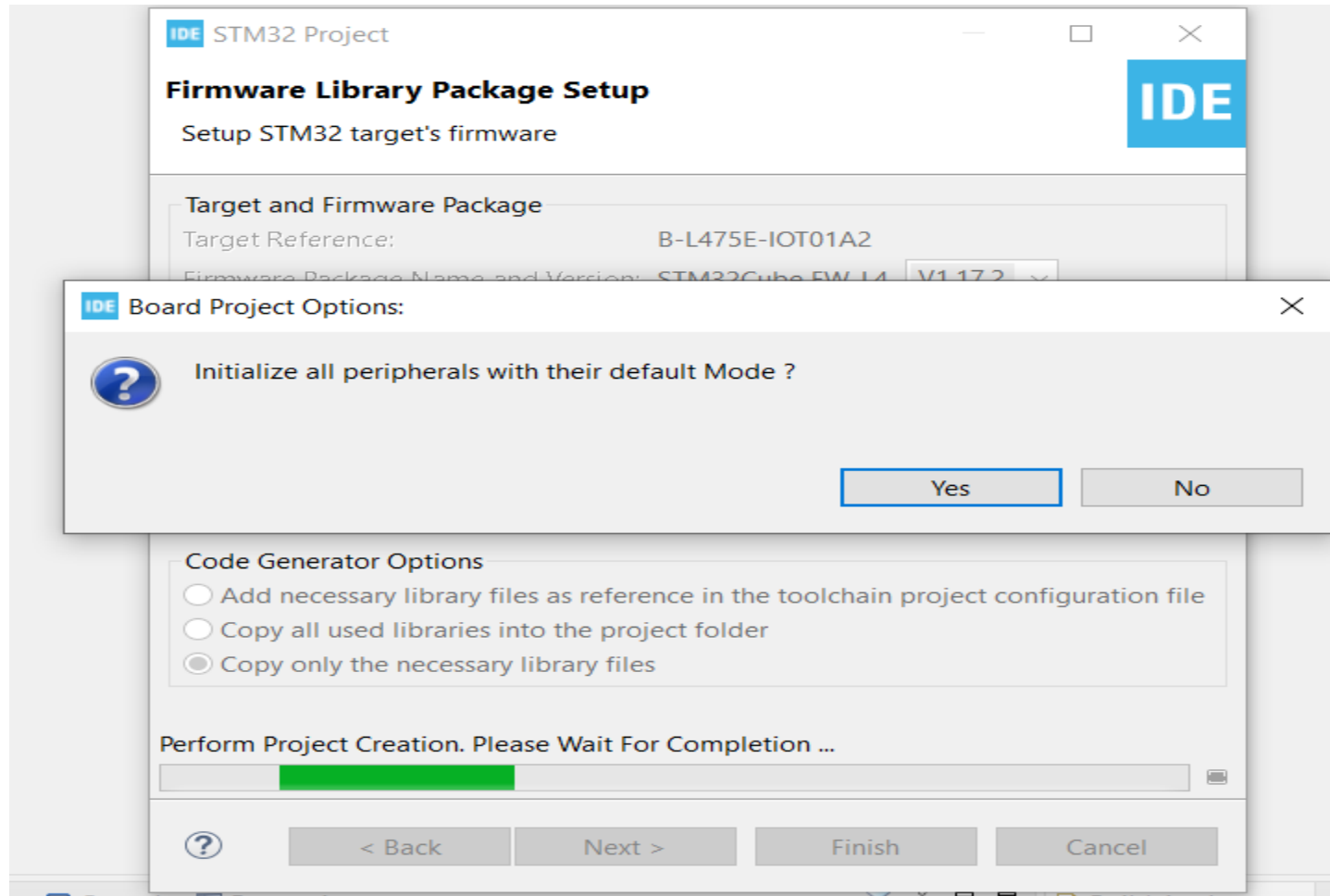
☐ Add necessary library files as reference in the toolchain project configuration file

☐ Copy all used libraries into the project folder

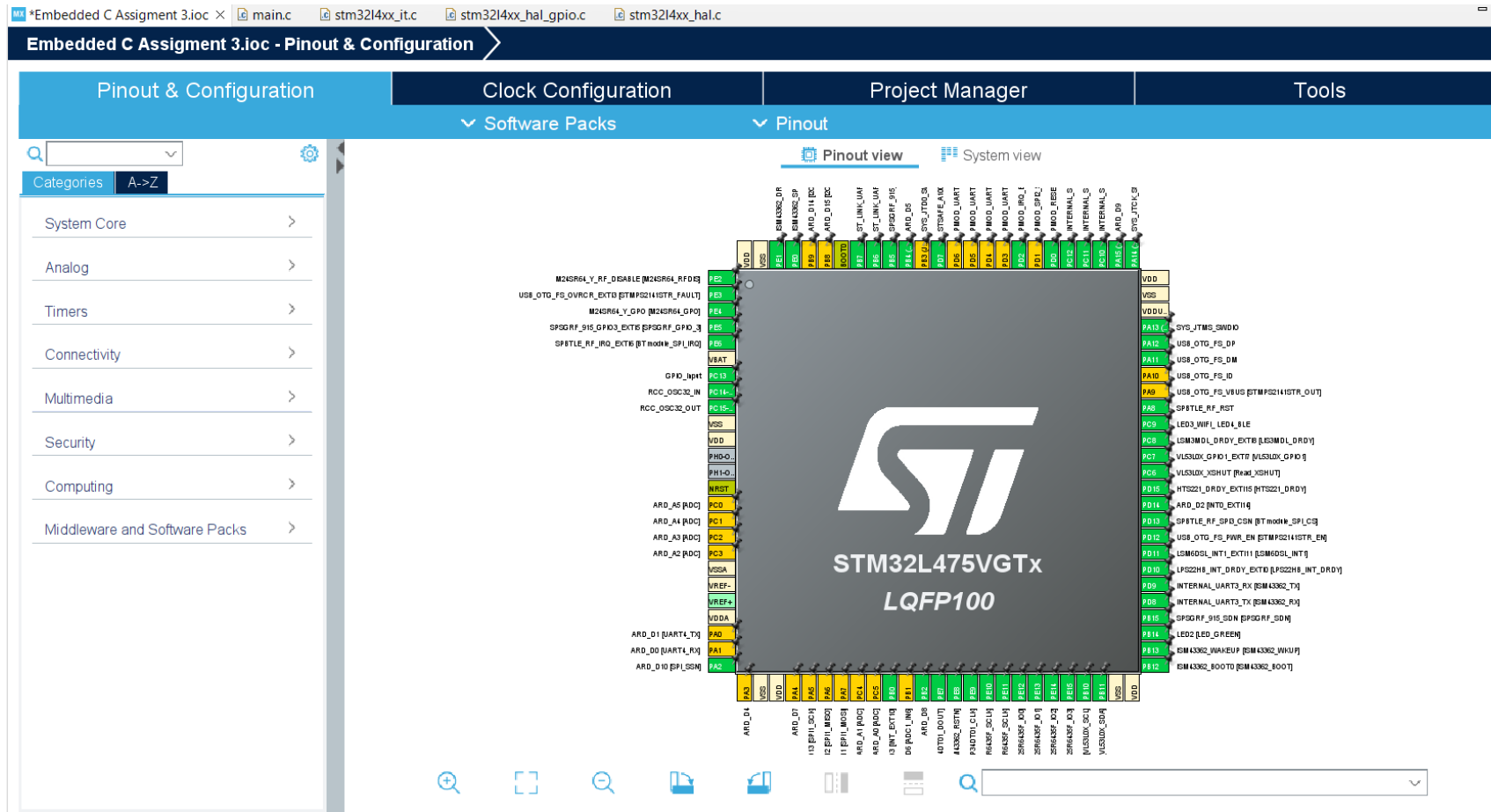
☒ Copy only the necessary library files

? < Back Next > Finish Cancel

Step 5. Click yes to initialize all peripherals to default



Step 6. When in .ioc file, click Pinout & Configurations



Step 7. Enable the CMSIS_V1 RTOS, and add one more tasks.

Embedded-RTOS-Assignment-4.ioc × main.c

Embedded-RTOS-Assignment-4.ioc - Pinout & Configuration

Pinout & Configuration

Categories A->Z

- Timers >
- Connectivity >
- Multimedia >
- Security >
- Computing >
- Middleware and ... >

AIROC-Wi-Fi-Blue

FATFS

FP-ATR-SIGFOX1

FP-SNS-SMARTA

☒ FREERTOS

I-CUBE-Cesium

I-CUBE-ITTIADB

I-CUBE-embOS

I-CUBE-wolfSSL

I-Cube-SoM-uGO/

TOUCHSENSING

USB_DEVICE

USB_HOST

X-CUBE-AI

X-CUBE-ALGOBL

X-CUBE-ALS

Clock Configuration

Project Manager

Software Packs

Pinout

FREERTOS Mode and Configuration

Configuration

Reset Configuration

☒ Tasks and Queues

☒ Timers and Semaphores

☒ Mutexes

☒ Events

☒ FreeRTOS Heap Usage

☒ Config parameters

☒ Include parameters

☒ Advanced settings

☒ User Constants

Tasks

Task Name	Priority	Stack Size...	Entry Function	Code Generati...	Parameter	Allocation	Buffer Name	Control Block N...
defaultTask	osPriorityNormal	128	StartDefaultTask	Default	NULL	Dynamic	NULL	NULL
myTask02	osPriorityLow	128	StartTask02	Default	NULL	Dynamic	NULL	NULL

Add Delete

Queues

Queue Name	Queue Size	Item Size	Allocation	Buffer Name	Control Block Name
------------	------------	-----------	------------	-------------	--------------------

Add Delete

Step 8. Change Timebase from systick to TIM1

*Embedded-RTOS-Assignment-1.ioc X

Embedded-RTOS-Assignment-1.ioc - Pinout & Configuration

Pinout & Configuration

Categories A-Z

System Core

- DMA
- GPIO
- IWDG
- NVIC
- ⚠ RCC
- ⚠ **SYS**
- ⚠ TSC
- WWDG

Analog >

Timers >

Connectivity >

Multimedia >

Security >

Computing >

Middleware and... >

Software Packs

SYS Mode and Configuration

Mode

Debug Serial Wire

- ☐ System Wake-Up 1
- ☐ System Wake-Up 2
- ☐ System Wake-Up 3
- ☐ System Wake-Up 4
- ☐ System Wake-Up 5

Power Voltage Detector In Disable

VREBUF Mode Disable

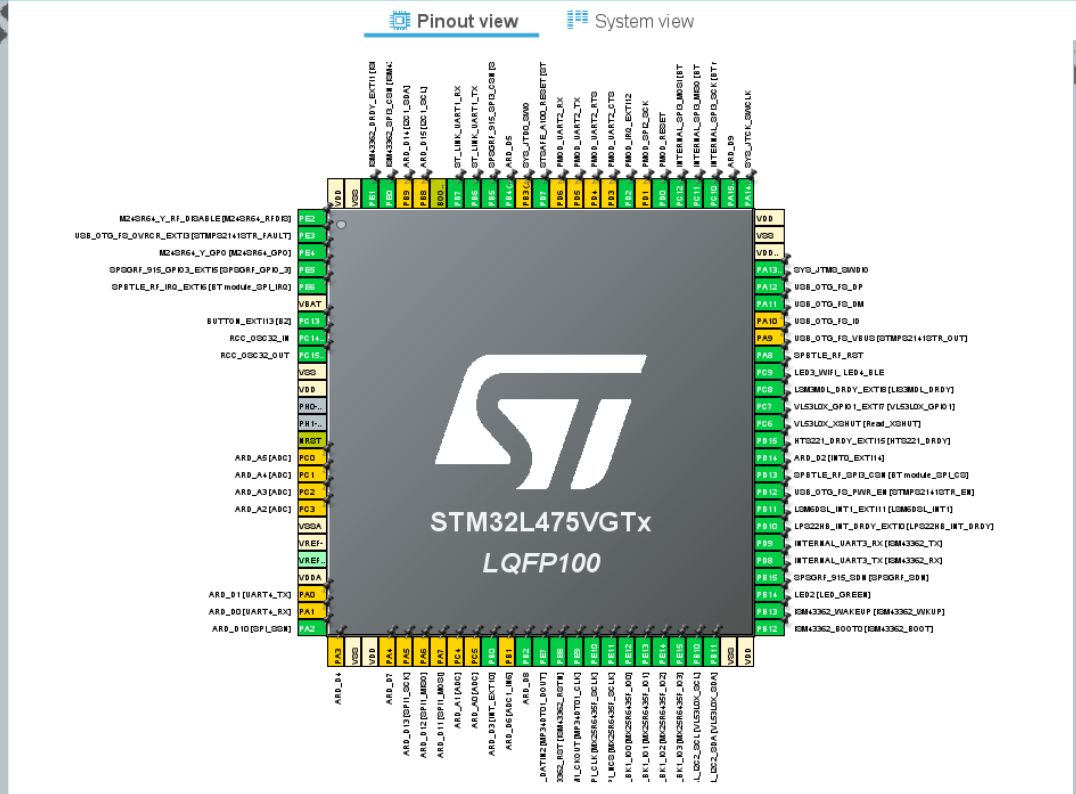
Timebase Source TIM1

Configuration

⚠ Warning: This peripheral has no parameters to be configured.

Pinout

Pinout view System view



Step 9. Enable the software timer in FreeRTOS settings and add two more timers. One one-shot timer and one auto reload timer

Tasks and Queues Timers and Semaphores Mutexes Events FreeRTOS Heap Usage

Config parameters Include parameters Advanced settings User Constants

Configure the below parameters :

Search (Ctrl+F)

USE_DAEMON_TASK_STARTUP_HOOK Disabled

CHECK_FOR_STACK_OVERFLOW Disabled

Run time and task stats gathering related definitions

GENERATE_RUN_TIME_STATS Disabled

USE_TRACE_FACILITY Disabled

USE_STATS_FORMATTING_FUNCTIONS Disabled

Co-routine related definitions

USE_CO_ROUTINES Disabled

MAX_CO_ROUTINE_PRIORITIES 2

Software timer definitions

USE_TIMERS Enabled

TIMER_TASK_PRIORITY 2

TIMER_QUEUE_LENGTH 10

TIMER_TASK_STACK_DEPTH 256 Words

Interrupt nesting behaviour configuration

LIBRARY_LOWEST_INTERRUPT_PRIORITY 15

LIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY 5

Added with 10.2.1 support

MESSAGE_BUFFER_LENGTH_TYPE size_t

USE_POSIX_ERRNO Disabled

Embedded-RTOS-Assignment-4.ioc x main.c

Embedded-RTOS-Assignment-4.ioc - Pinout & Configuration

Pinout & Configuration Clock Configuration Project Manager

Software Packs Pinout

FREERTOS Mode and Configuration

Configuration

Reset Configuration

Tasks and Queues Timers and Semaphores Mutexes Events FreeRTOS Heap Usage

Config parameters Include parameters Advanced settings User Constants

Timers

Timer Name	Callback	Type	Code Generation O...	Parameter	Allocation	Control Block Name
myTimer01	prvMyTimerOneShot	osTimerOnce	Default	NULL	Dynamic	NULL
myTimer02	prvMyTimerAutoRel...	osTimerPeriodic	Default	NULL	Dynamic	NULL

Add Delete

Step 10. Create "Task 1" that blinks the LED2 at a rate of 1 second. Create "Task 2" that blinks the "Wifi/BLE" LED at a rate of 2 seconds

```
Embedded-RTOS-Assignment-4.ioc × main.c ×
709 void StartDefaultTask(void const * argument)
710 {
711     /* USER CODE BEGIN 5 */
712     /* Infinite loop */
713     for(;;)
714     {
715         HAL_GPIO_TogglePin(LED2_GPIO_Port, LED2_Pin);
716         osDelay(1000);
717     }
718     /* USER CODE END 5 */
719 }
720
721 /* USER CODE BEGIN Header_StartTask02 */
722 /**
723  * @brief Function implementing the myTask02 thread.
724  * @param argument: Not used
725  * @retval None
726  */
727 /* USER CODE END Header_StartTask02 */
728 void StartTask02(void const * argument)
729 {
730     /* USER CODE BEGIN StartTask02 */
731     /* Infinite loop */
732     for(;;)
733     {
734         HAL_GPIO_TogglePin(LED3_WIFI__LED4_BLE_GPIO_Port, LED3_WIFI__LED4_BLE_Pin);
735         osDelay(2000);
736     }
737     /* USER CODE END StartTask02 */
738 }
---
```

Step 11. Create "Timer 1" that is a One-Shot timer function named prvMyTimerOneShot(). The timer should fire 15 seconds after startup and display a message "prvMyTimerOneShot" on the console.

```
MX Embedded-RTOS-Assignment-4.ioc × main.c ×
736     }
737     /* USER CODE END StartTask02 */
738 }
739
740 /* prvMyTimerOneShot function */
741 void prvMyTimerOneShot(void const * argument)
742 {
743     /* USER CODE BEGIN prvMyTimerOneShot */
744     char buf[100];
745     snprintf(buf, sizeof(buf), "prvMyTimerOneShot\r\n");
746     HAL_UART_Transmit(&huart1, (uint8_t*)buf, strlen(buf), 1000);
747     /* USER CODE END prvMyTimerOneShot */
748 }
749
```

Step 12. Create "Timer 2" that is an auto-reload timer function named `prvMyTimerAutoReload()`. The timer should fire every 5 seconds and display a count and a message on the console, for example, "prvMyTimerAutoReload: 1".

```
MX Embedded-RTOS-Assignment-4.ioc  main.c ×
745     snprintf(buf, sizeof(buf), "prvMyTimerOneShot\r\n");
746     HAL_UART_Transmit(&huart1, (uint8_t*)buf, strlen(buf), 1000);
747     /* USER CODE END prvMyTimerOneShot */
748 }
749
750 /* prvMyTimerAutoReload function */
751 void prvMyTimerAutoReload(void const * argument)
752 {
753     /* USER CODE BEGIN prvMyTimerAutoReload */
754     static int count = 1;
755     char buf[100];
756     snprintf(buf, sizeof(buf), "prvMyTimerAutoReload: %d\r\n", count);
757     HAL_UART_Transmit(&huart1, (uint8_t*)buf, strlen(buf), 1000);
758     count ++;
759     /* USER CODE END prvMyTimerAutoReload */
760 }
761
```

Step 13. Don't forget to start the timer and set the timing parameter

```
Embedded-RTOS-Assignment-4.ioc  main.c ×
123  MX_USART1_UART_Init();
124  MX_USB_OTG_FS_PCD_Init();
125  /* USER CODE BEGIN 2 */
126
127  /* USER CODE END 2 */
128
129  /* USER CODE BEGIN RTOS_MUTEX */
130  /* add mutexes, ... */
131  /* USER CODE END RTOS_MUTEX */
132
133  /* USER CODE BEGIN RTOS_SEMAPHORES */
134  /* add semaphores, ... */
135  /* USER CODE END RTOS_SEMAPHORES */
136
137  /* Create the timer(s) */
138  /* definition and creation of myTimer01 */
139  osTimerDef(myTimer01, prvMyTimerOneShot);
140  myTimer01Handle = osTimerCreate(osTimer(myTimer01), osTimerOnce, NULL);
141
142  /* definition and creation of myTimer02 */
143  osTimerDef(myTimer02, prvMyTimerAutoReload);
144  myTimer02Handle = osTimerCreate(osTimer(myTimer02), osTimerPeriodic, NULL);
145
146  /* USER CODE BEGIN RTOS_TIMERS */
147  /* start timers, add new ones, ... */
148  osTimerStart(myTimer01Handle, 15000);
149  osTimerStart(myTimer02Handle, 5000);
150  /* USER CODE END RTOS_TIMERS */
151
152  /* USER CODE BEGIN RTOS_QUEUES */
153  /* add queues, ... */
```

Step 14. Debug and run the code, test is successful

