MANUAL FREERTOS Addition to Project

Here’s a clear and simplified guide to setting up an STM32 project with FreeRTOS integration:

1. Starting a New STM32 Project

1. Open STM32CubeIDE and navigate to the **Board Selector** tab.
2. Search for your board model and select it, then click **Next**.
3. Enter a project name and choose your preferred programming language.
4. Select **STM32Cube** for the project type (this avoids manually writing peripheral drivers).
5. When prompted, click **No** to initialize all peripherals with their default settings.

---

2. Adding FreeRTOS to Your Project

Step 1: Set up Folder Structure

1. Right-click the root directory of your project and create a folder called ThirdParty.
2. Your project folder structure should now look like this:

   ```
   Core/
   Drivers/
   ThirdParty/
   ```

3. Inside ThirdParty, create a folder called FreeRTOS.

Step 2: Copy FreeRTOS Files

1. In the downloaded FreeRTOS kernel source, you should have the following structure:

   ```
   FreeRTOS/
   FreeRTOS-plus/
   ```

2. Copy the **LICENSE** folder from the FreeRTOS/ directory into ThirdParty/FreeRTOS/.
3. Copy **everything** from FreeRTOS/Source/, except the portable folder, into ThirdParty/FreeRTOS/.
4. Copy the **portable** folder from FreeRTOS/Source/ into ThirdParty/FreeRTOS/.

Step 3: Clean Up the portable Folder

1. Inside the portable folder, delete everything except:
   - GCC/
   - MemMang/
   - ReadMe (if it exists)
   - MSVC-MingW/ **(exclude from the build if it is unused)**
   - CMakeLists.txt (if it’s present)

2. Inside ThirdParty/FreeRTOS/portable/GCC/, keep only the ARM_CM4F folder, which is used for ARM Cortex-M4 with floating-point support. Delete other architecture-specific folders. (Keep your target device)

---

3. Configuring Include Paths

1. Right-click the project and go to **Properties > C/C++ Build > Settings > Tool Settings**.
2. Under **MCU GCC Compiler > Include paths**, add the following paths:
   - ThirdParty/FreeRTOS
   - ThirdParty/FreeRTOS/include
   - ThirdParty/FreeRTOS/portable/GCC/ARM_CM4F

(Note: Path to portable may vary based on board and compiler type.)

---

4. Adjust Project Configuration

1. Right-click on the ThirdParty folder, select **Properties > C/C++ Build**.
   - Make sure **Exclude resource from build** is **unchecked**.
   - Click **Apply & Close**.
2. In Core/Src/sysmem.c, FreeRTOS manages memory heap:
   - Right-click sysmem.c and go to **C/C++ Build**.
   - **Check Exclude resource from build**.
   - Click **Apply & Close**.

---

5. Heap Management

1. In ThirdParty/FreeRTOS/portable/MemMang, delete all files except heap_4.c (this will manage heap memory).

---

6. Add FreeRTOS Configuration

1. Download the FreeRTOSConfig.h file:
   - Search for the target microcontroller's FreeRTOSConfig.h in the kernel source code directory under **Demo**.
2. Paste the FreeRTOSConfig.h file into ThirdParty/FreeRTOS/.

---

7. Fixing SystemCoreClock Error

If you encounter a SystemCoreClock error, add this code to FreeRTOSConfig.h:

```
#if defined(__ICCARM__) || defined(__GNUC__) || defined(__CC_ARM)
  #include <stdint.h>
  extern uint32_t SystemCoreClock;
#endif
```

---

8. Removing Redundant Interrupt Handlers

1. You may get errors for the following interrupt handlers, which are already defined in the FreeRTOS libraries:
   - SVC_Handler
   - PendSV_Handler
   - SysTick_Handler

To fix this:

- Open Core/Src/system_stm32f4xx_it.c and remove these handlers.

---

9. Resolve Undefined References

1. If you encounter the undefined reference to 'vApplicationTickHook' error, open FreeRTOSConfig.h.
2. Add or modify the following macros to disable unnecessary hooks:

```
#define configUSE_TICK_HOOK 0
#define configUSE_MALLOC_FAILED_HOOK 0
#define configCHECK_FOR_STACK_OVERFLOW 0
```

These steps should allow you to successfully set up an STM32 project and integrate FreeRTOS.

Stm32CubeIDE Based FREERTOS Addition to Project

1. Starting a New STM32 Project

1. Open STM32CubeIDE and navigate to the **Board Selector** tab.
2. Search for your board model and select it, then click **Next**.
3. Enter a project name and choose your preferred programming language.
4. Select **STM32Cube** for the project type (this avoids manually writing peripheral drivers).
5. When prompted, click **No** to initialize all peripherals with their default settings.

2. Configuring FreeRTOS

1. In STM32CubeIDE, go to the **Pinout & Configuration** tab.
2. Under the **Middleware & Software Packs** section, click on FreeRTOS.
3. In the **Mode Menu**, select **CMSIS_V2.**
4. Under **Configuration Menu**, you can configure various FreeRTOS parameters, such as tick rate, heap management, and other settings.
5. Under **Configuration Menu,** click **Advanced Settings**
    - **Newlib settings** :
      USE_NEWLIB_REENTRANT : ENABLED
6. Once you're done configuring, click **Yes to generate the code** with FreeRTOS integrated.

When RTOS is used it is recommended to use Hardware Abstraction Layer timebase source.

1. Under the **System Core** section, go to the **SYS.**
2. **Timebase Source** could be settled to **TIM4** or **TIM6**.