

Cs/053

Application Note

Cesium and STM32CubeMX



Weston Embedded Solutions
2200 N. Commerce Parkway, Suite 200
Weston, FL 33326
United States of America

www.weston-embedded.com

The programs and code examples in this manual are presented for instructional value. The programs and examples have been carefully tested but are not guaranteed to any particular purpose. The publisher does not offer any warranties and does not guarantee the accuracy, adequacy, or completeness of any information herein and is not responsible for any errors and omissions. The publisher assumes no liability for damages resulting from the use of the information in this manual or for any infringement of the intellectual property rights of third parties that would result from the use of this information.



Table of Contents

- Chapter 1 - Abstract 4**
 - 1.1. Required Software and Hardware 4
- Chapter 2 - Getting Started 5**
 - 2.1. Prerequisites 5
 - 2.2. Obtaining the Cesium STM32Cube Expansion Package 5
 - 2.3. Installing the Expansion Package 5
- Chapter 3 - Using “WES.I-Cube-Cesium” 8**
 - 3.1. Create STM32CubeMX Project 8
 - 3.2. Adding the Cesium Expansion Package 9
 - 3.3. Selecting Timebase Source 11
 - 3.4. Generating Code 12
 - 3.5. Toggle GPIO 14
- Chapter 4 - Revision History 16**
- List of Figures 18**
- List of Tables 19**
- Code Listings 20**

Chapter 1 - Abstract

This application note describes how to create an STM32CubeMX project with basic configurations to run Cesium Cs/OS3 RTOS using either system-tick or dynamic-tick configurations. It will guide the user through the process of installing the Cesium package into STM32CubeMX, selecting/configuring the Software Package within the ST tool, and generate an EWARM, MDK-ARM, or STM32CubeIDE project. The process will be explained using an STM32U575I-EV evaluation board; however, the same process can be applied to any MCU or board using the ARMv8-M architecture, such as ARM Cortex-M23, Cortex-M33 or Cortex-M55; as well as the ARMv7-M architecture, such as ARM Cortex-M7, Cortex-M4, or Cortex-M3.

1.1. Required Software and Hardware

- STM32CubeMX from STMicroelectronics
- IAR Embedded Workbench for ARM (EWARM)
- MDK-ARM(μVision)
- STM32CubeIDE
- STM32U575I-EV Evaluation Board
- “WES.I-CUBE-Cesium.1.4.0.pack” expansion package from Weston Embedded Solutions

Chapter 2 - Getting Started

This chapter gives the user an insight into how to install and use the “WES.I-Cube-Cesium.1.4.0.pack” expansion package. The following topics are covered in this chapter:

- Prerequisites
- Obtaining the Cesium STM32Cube expansion package
- Installing the expansion package

2.1. Prerequisites

Before creating and running the first STM32CubeMX EWARM application, the minimal set of required tools and components must be verified:

- Toolchain for the specific microcontroller
- Development board
- Cesium STM32Cube Expansion Package

2.2. Obtaining the Cesium STM32Cube Expansion Package

The expansion package is available to customers via the following link: https://www.weston-embedded.com/stm_packages/

2.3. Installing the Expansion Package

Once the Cesium STM32Cube Expansion Package has been downloaded to the host machine, the package can be installed directly into the STM32CubeMX tool by clicking on the “**Install/Remove**” button under the “**Manage software installations**” section as shown in Figure 2.1.

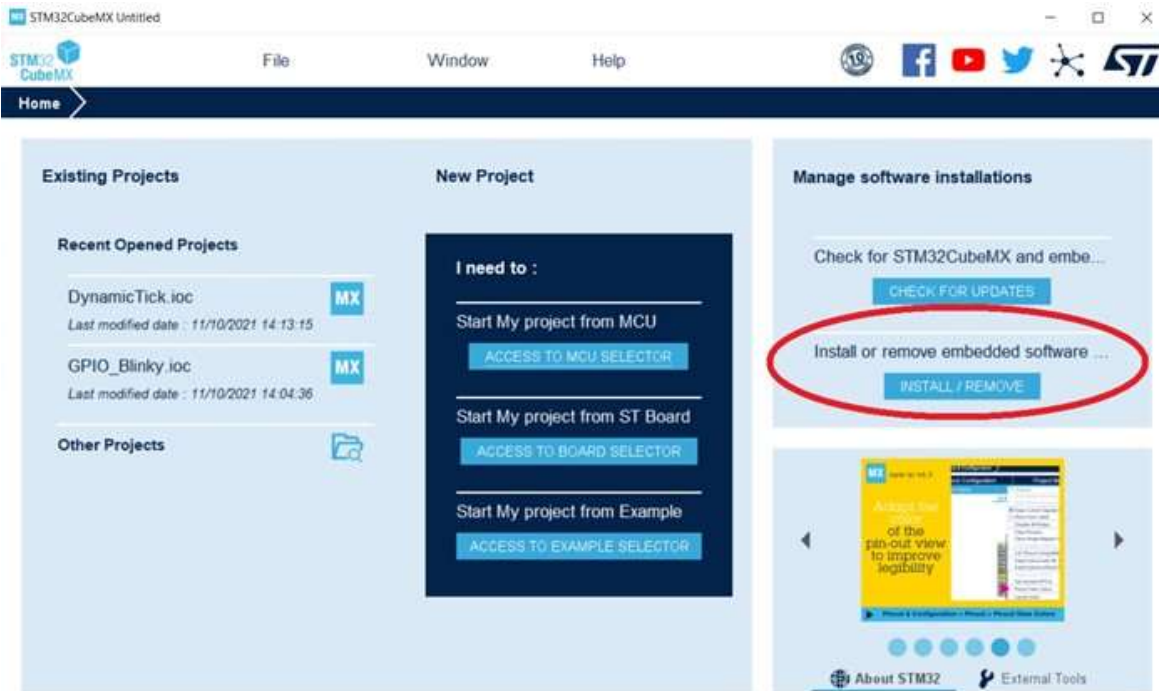


Figure 2.1 – Manage Software Installations

After clicking the “**INSTALL/REMOVE**” button, the “WES.I-Cube-Cesium.1.4.0.pack” expansion package can be dragged and dropped into the “**Embedded Software Packages Manager**” pop-up window. This will install the package into the STM32CubeMX tool as shown in Figure 2.2.

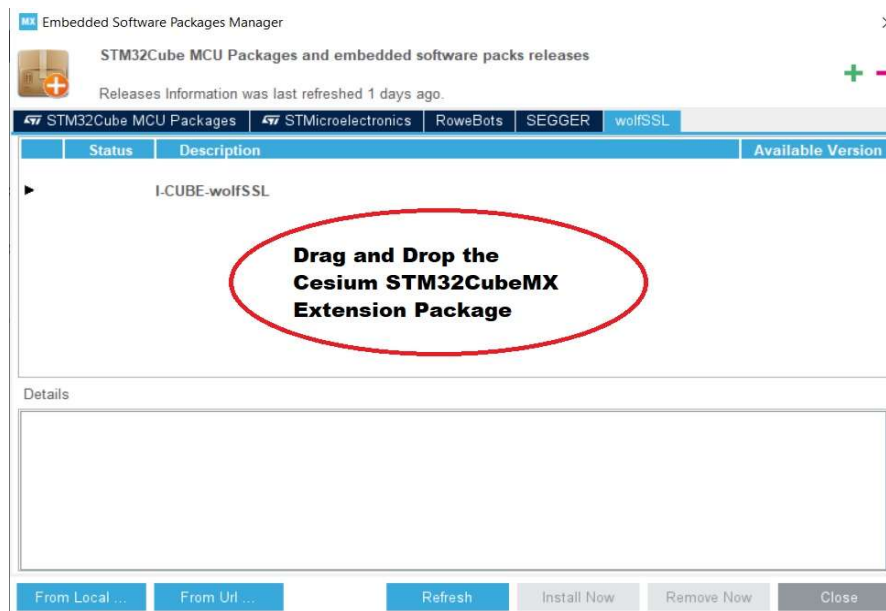


Figure 2.2 – Embedded Software Package Manager

Figure 2.3 shows the complete result after ST tool finishes the installation. At this point, the “Embedded Software Package Manager” window can be closed, and the STM32CubeMX should be ready to use the new Cesium Expansion Package.

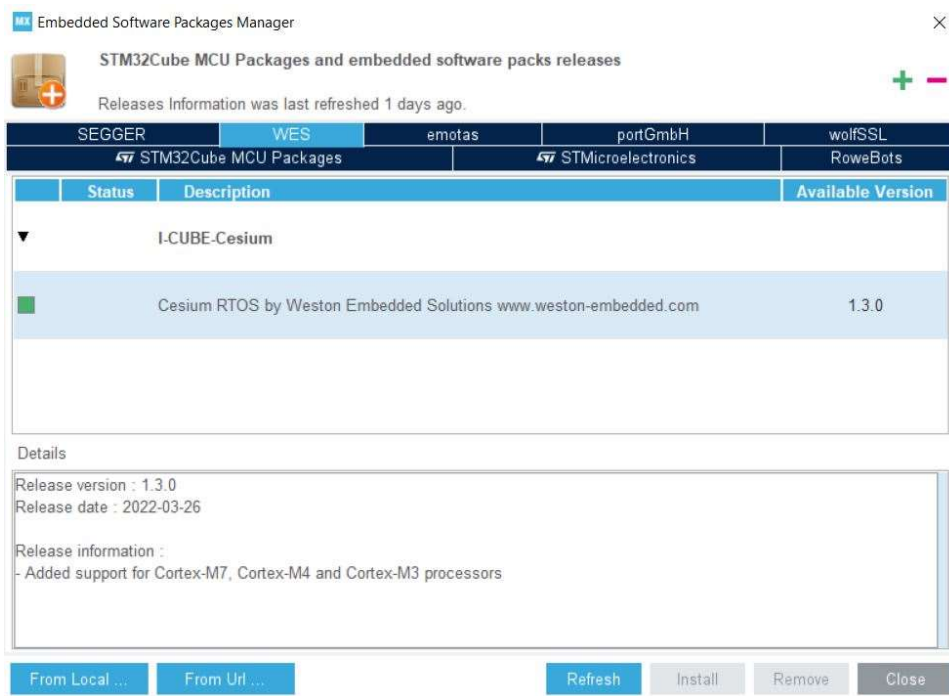


Figure 2.3 – WES “I-Cube-Cesium” Expansion Package

Chapter 3 - Using “WES.I-Cube-Cesium”

3.1. Create STM32CubeMX Project

Before going into more detail, this section assumes the user is familiar with STM32CubeMX project creation; therefore, a few steps might be missing from the process. From the user’s point of view, STM32CubeMX will be opened and an example project using the STM32U575I-EV evaluation board will be created. The STM32U575I-EV evaluation board was chosen since it utilizes an MCU with the ARMv8-M(Cortex-M55, Cortex-M33 or Cortex-M23) architecture; however, the same procedure can be applied to other boards with the same architecture or ARMv7-M(Cortex-M7, Cortex-M4 or Cortex-M3) since the expansion package is generic enough to do so.

Start STM32CubeMX tool and choose either “File → New Project...” from the menu or click on the “ACCESS TO BOARD SELECTOR” button. After the “New Project from a Board” window pops-up make sure to look for and select the **STM32U575I-EV** board and click the “Start Project” button as shown in Figure 3.1.

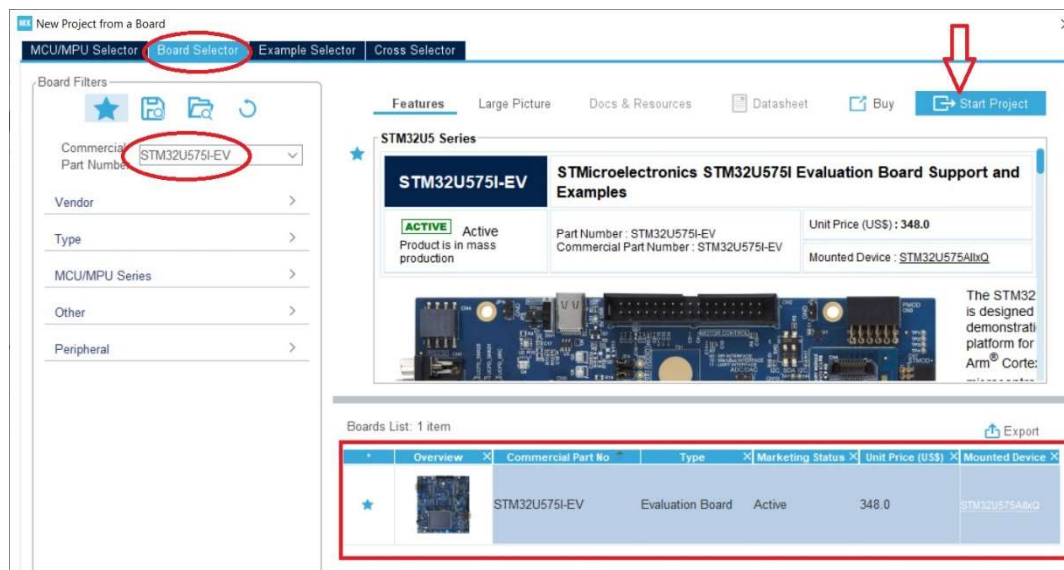


Figure 3.1 – New Project from a Board

If a new message appears asking to initialize all peripheral with their default mode, then select “NO” and ensure that the “No TrustZone” option is selected as well. The end-result will be an STM32CubeMX workspace where the user will be able to configure clocks, add/configure peripherals, and add the Cesium Expansion Package.

3.2. Adding the Cesium Expansion Package

Once a project is started with the STM32CubeMX, the user can make several configurations to peripherals according to the application’s needs; however, this section only focuses on how to add the Cesium Expansion Package library code and a “Blinky” application code to the workspace by using the following procedure:

- a) Open the “**Software Packs Component Selector**” window using the keyboard shortcut “**ALT+O**” or by clicking on “**Software Packs**” tab and choosing “**Software Components**” as shown in Figure 3.2.

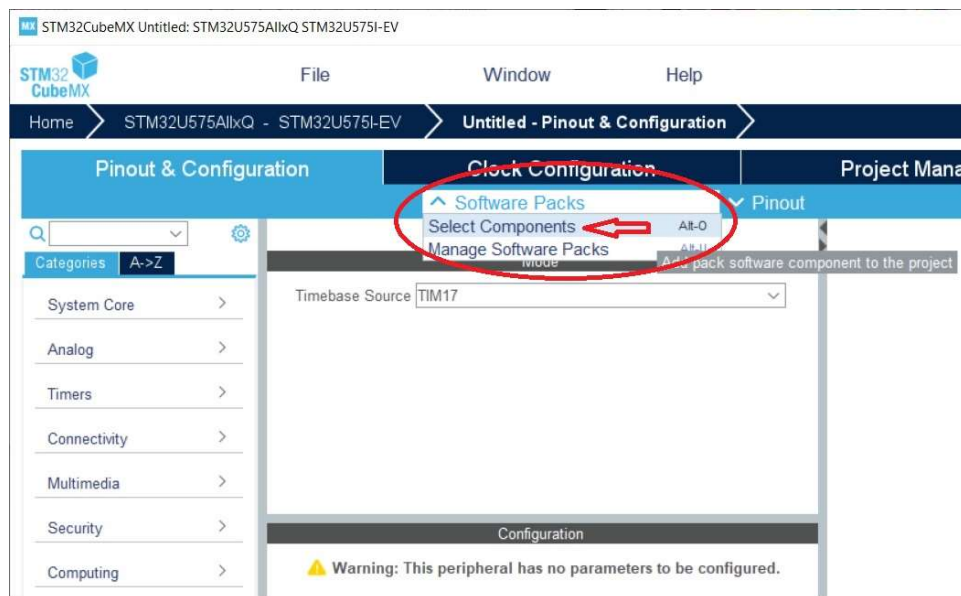


Figure 3.2 – Select “Software Packs”

- b) Once the “**Software Packs Component Selector**” window appears, search for the “**WES.I-Cube-Cesium**” component and select which RTOS Cesium library you would like to use as shown in Figure 3.3. The user can choose between SysTick or Dynamic Tick configurations. If Dynamic Tick is selected, then additional steps will be required on the STM32CubeMX workspace, which will be explained in Section 3.3. In addition to selecting the Cesium RTOS library, the user needs to add the “**GPIO_Blinky**” application code. The application code contains everything the RTOS needs to initialize, start, and create the first task. If the “**GPIO_Blinky**” application code is not selected, then the user will need to manually initialize Cesium RTOS.

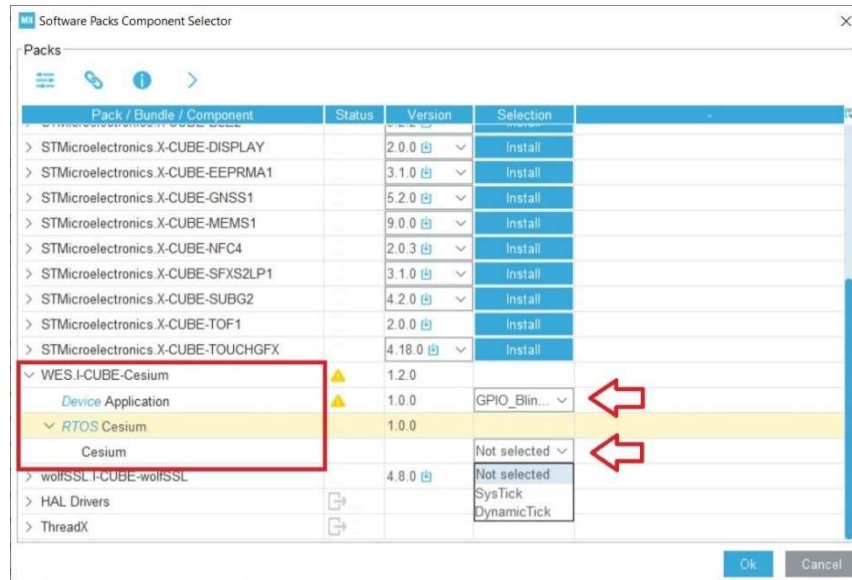


Figure 3.3 – Software Packs Component Selector

- c) After completing the previous step, a new category called “**Software Packs**” will be created in the “**Pinout & Configuration**” tab. Click on the “**WES.I-CUBE-Cesium**” option as shown in Figure 3.4 and select the check boxes for “**Device application**” and “**RTOS Cesium**”. Selecting the “**Device Application**” option will allow the user to add the required API to toggle an LED on the board.

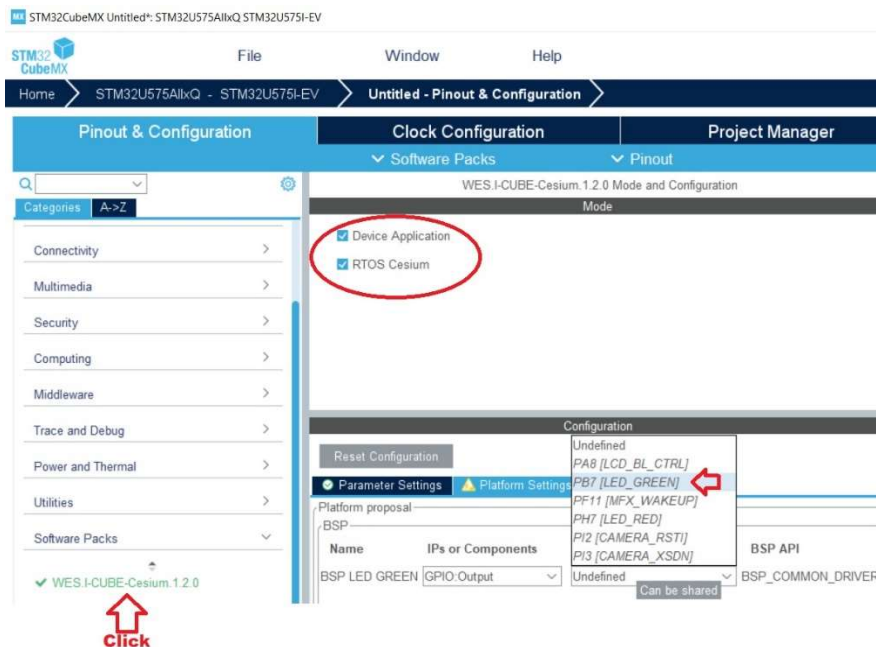


Figure 3.4 – Pinout & Clock Configuration

3.3. Selecting Timebase Source

The Cesium Expansion Package comes with two configurations to choose from. One for SysTick, which uses the internal ARMv8-M timer to generate a tick, and the second one is the Dynamic Tick, which uses one of the MCU peripheral timers to generate the tick for the RTOS to keep track of time. The only requirement from the user is to set the Timebase source in the STM32CubeMX workspace, which will be shown in the following steps:

- a) Under the “**Pinout & Configuration**” tab, locate the “**System Core → Sys**” category to open the “**SYS Mode and Configuration**” window. This will allow to choose the SysTick for the “**Timebase Source**” dropdown list as shown in Figure 3.5.

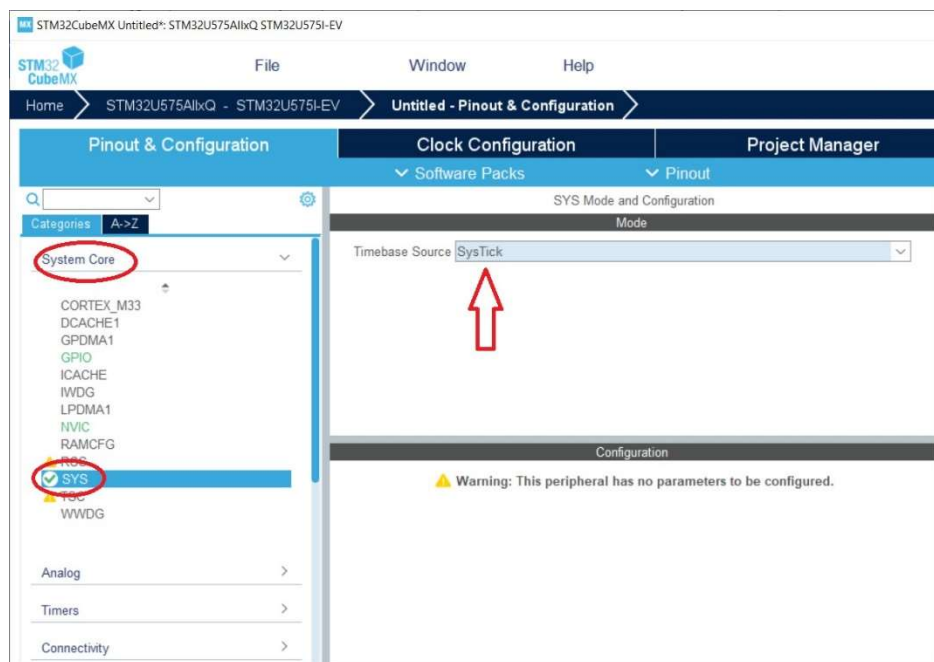


Figure 3.5 – SysTick Chosen as Timebase Source

- b) This step can be skipped if the user decides to use the SysTick as time source for Cesium RTOS. However, if Dynamic Tick is chosen, locate the **TIM2** peripheral under the “**Timers**” category as shown in Figure 3.6. Only the “**One pulse mode**” checkbox needs to be checked to enable the TIM2 API functions when the final code is generated. The generated code will work with TIM2; however, if the timer is not available with your MCU then it is possible to choose a different TIMx and modify ‘**bsp_os.c**’ accordingly.

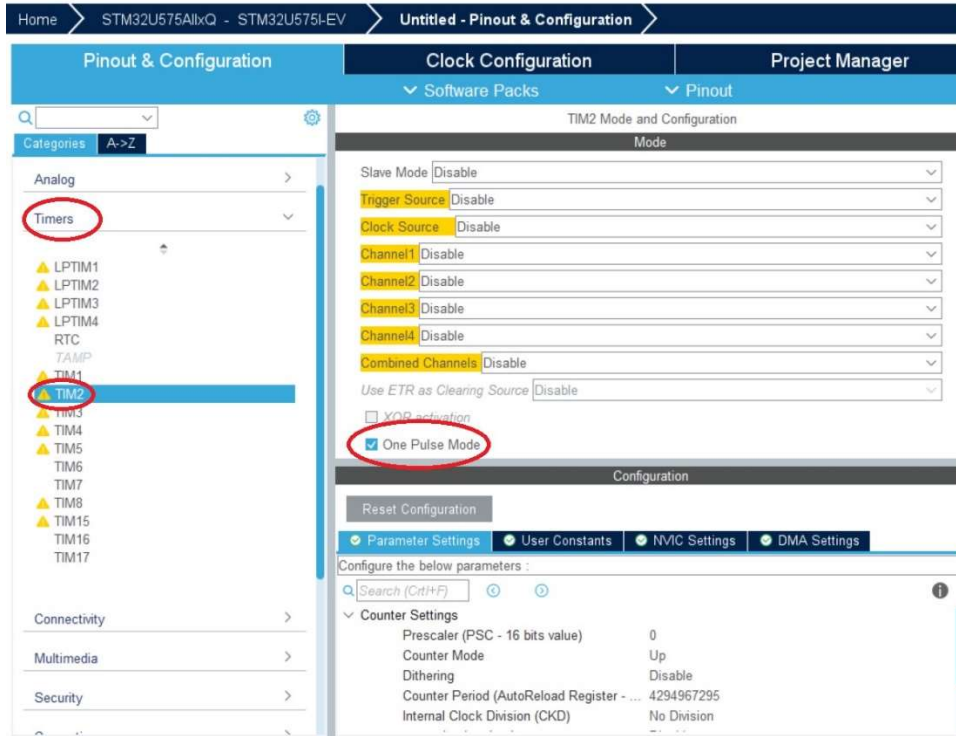


Figure 3.6 – Enabling TIM2 for Dynamic Tick Configuration

3.4. Generating Code

At this point, the user is ready to name the project and generate the code using STM32CubeMX tool. The generated code can be chosen between EWARM, MDK-ARM or STM32CubeIDE workspace with all the necessary files required to build and run Cesium RTOS. The following steps show how to generate the final workspace:

- Select the “**Project Manager**” tab in the STM32CubeMX workspace, and provide a project name, location, and toolchain to be used as shown in Figure 3.7.

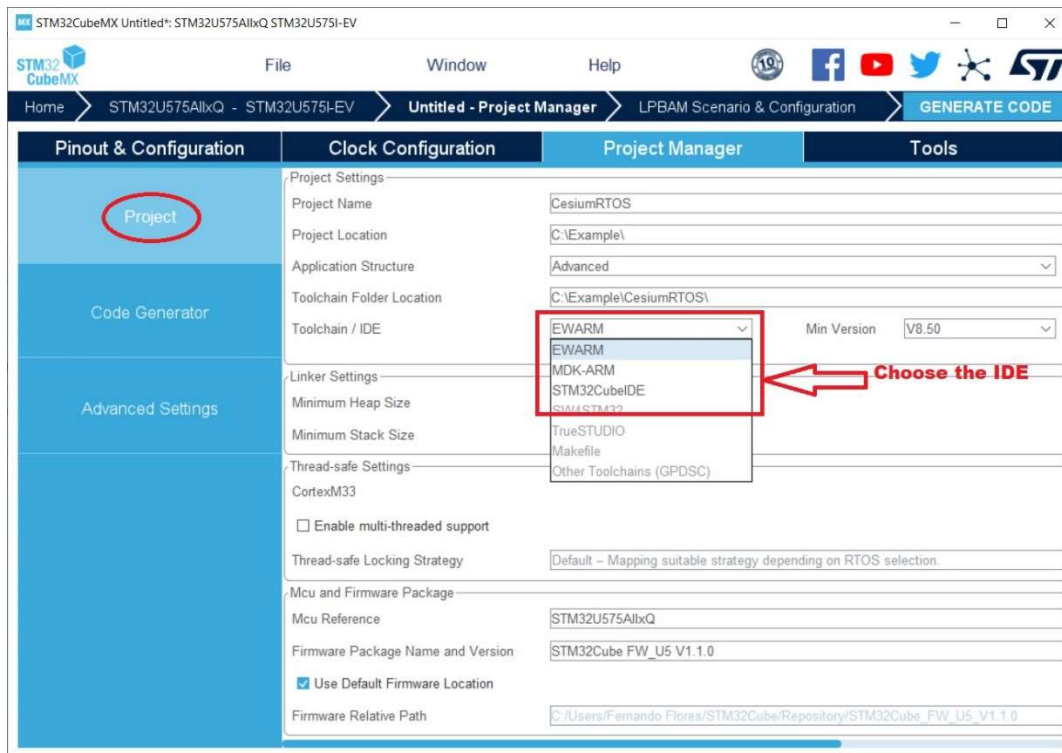


Figure 3.7 – Project Manager’s Project Information

- b) Choose the “**Code Generator**” option and make sure to select “**Copy all used libraries into the project folder**” as shown in Figure 3.8. Once this is done, click on the “**GENERATE CODE**” button. STM32CubeMX will generate all the necessary files and create the appropriate workspace, which the user will be able to compile without errors.
- **Note:** In the case of STM32CubeIDE, make sure to unselect “Generate Under Root” box before generating.

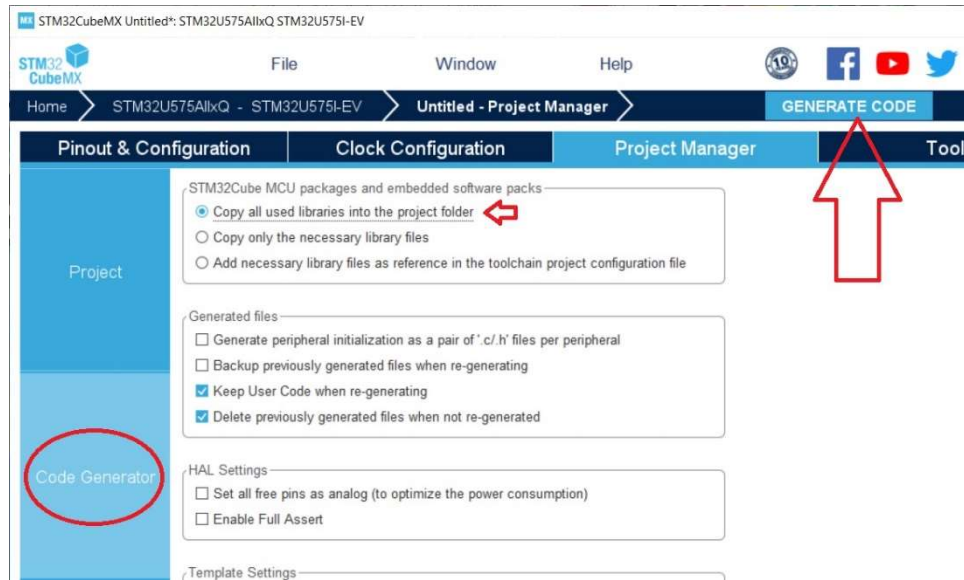


Figure 3.8 – Code Generator

- c) Once the code generation is finished, open the new created project and build it.

3.5. Toggle GPIO

The final piece is to enable the GPIO that will control the LED on board. The code snippet provided is only an example for the STM32U575I-EV board. The user can adapt it based on the LED selection shown in Figure 3.4.

- a) Locate the “**StartupTask**” task, which should have been generated in the “**app_gpio_blinky.c**” file and modify the “**USER CODE XXXX**” sections as shown in bold in Listing 3.1. Once this is done, the user can compile and download the code into the board, and the LED should blink at 1 second intervals.

```

static void StartupTask (void *p_arg)
{
    OS_ERR os_err;
    /* USER CODE BEGIN 1 */
    GPIO_InitTypeDef GPIO_InitStruct;
    /* USER CODE END 1 */

    (void)p_arg;

    BSP_OS_TickEnable();

    #if OS_CFG_STAT_TASK_EN > 0u
        OSStatTaskCPUUsageInit(&os_err);
    #endif

    #ifndef CPU_CFG_INT_DIS_MEAS_EN
        CPU_IntDisMeasMaxCurReset();
    #endif

    /* USER CODE BEGIN 2 */
    /* ----- LED_GREEN CONFIGURATION ----- */
    BSP_LED_Init(LED_GREEN);
    /* USER CODE END 2 */

    while (DEF_TRUE) { /* Task body, always written as an infinite loop.*/
    /* USER CODE BEGIN 3 */
        OSTimeDlyHMSM(0u, 0u, 1u, 0u,
                     OS_OPT_TIME_HMSM_STRICT,
                     &os_err);

        BSP_LED_Toggle(LED_GREEN);
    /* USER CODE END 3 */
    }
}

```

Listing 3.1 – LED Toggle Example

Chapter 4 - Revision History

Should the user find any errors in this document, please inform Weston Embedded Solutions in order to apply the appropriate corrections for future updates.

REVISION	DATE	DESCRIPTION
1	04-Nov-2021	- Initial document with version control - Document has all the required steps to install and use the Cesium expansion package with the STM32CubeMX using EWARM
2	20-Mar-2022	Added: Chapter 1 - Abstract : Added reference to Keil-MDK and STM32CubeIDE 1.1 Required Software and Hardware : Added bullet points for Keil-MDK and STM32CubeIDE. 3.4 Generating Code : Added Keil-MDK and STM32CubeIDE mentioned in the paragraph. Updated: Figure 3.7 – Project Manager’s Project Information : Updated the image for other IDE’s
3	27-Mar-2022	Updated: Figure 2.3 – WES “I-Cube-Cesium” Expansion Package : Updated the image with newest release version. Added: Chapter 1 - Abstract : Added text in the paragraph for ARMv7-M architecture, as well as ARM Cortex-M7, M4 and M3 processors. Changed: - Document name to “AN-Cesium-STM32CubeMX” since the package can support ARMv7-M(Cortex-M7, Cortex-M4 and Cortex-M3) and ARMv8-M(Cortex-M33 and Cortex-M23).
4	01-Mar-2023	Updated: Obtaining the Cesium STM32Cube Expansion Package : Update the hyperlink where to get the expansion package from WES website

5	08-Aug-2025	Updated: • Update reference to “WES.I-CUBE-Cesium.1.4.0.pack” • Added references to Cortex-M55 support. • Update IDE names such EWARM and MDK-ARM.
---	-------------	---

Table 4.1 – Revision History

List of Figures

Figure 2.1 – Manage Software Installations	6
Figure 2.2 – Embedded Software Package Manager	6
Figure 2.3 – WES “I-Cube-Cesium” Expansion Package	7
Figure 3.1 – New Project from a Board	8
Figure 3.2 – Select “Software Packs”	9
Figure 3.3 – Software Packs Component Selector.....	10
Figure 3.4 – Pinout & Clock Configuration.....	10
Figure 3.5 – SysTick Chosen as Timebase Source	11
Figure 3.6 – Enabling TIM2 for Dynamic Tick Configuration	12
Figure 3.7 – Project Manager’s Project Information	13
Figure 3.8 – Code Generator.....	14

List of Tables

Table 4.1 – Revision History	17
------------------------------------	----

Code Listings

Listing 3.1 – LED Toggle Example	15
--	----