


5. FatFs and USB using default settings are already marked as configured . Click FatFs and USB buttons to display default configuration settings. You can also change them by following the guidelines provided at the bottom of the window.

Figure 143. FatFs IP instances

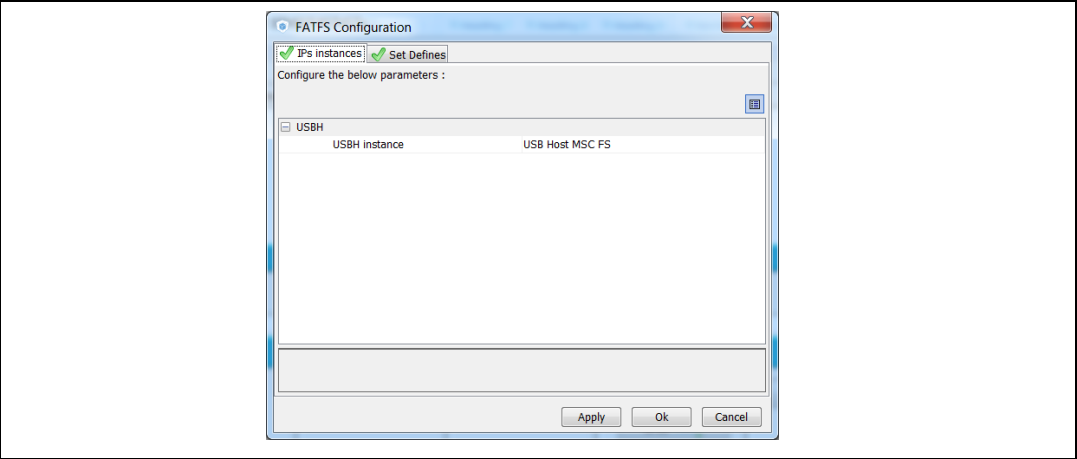
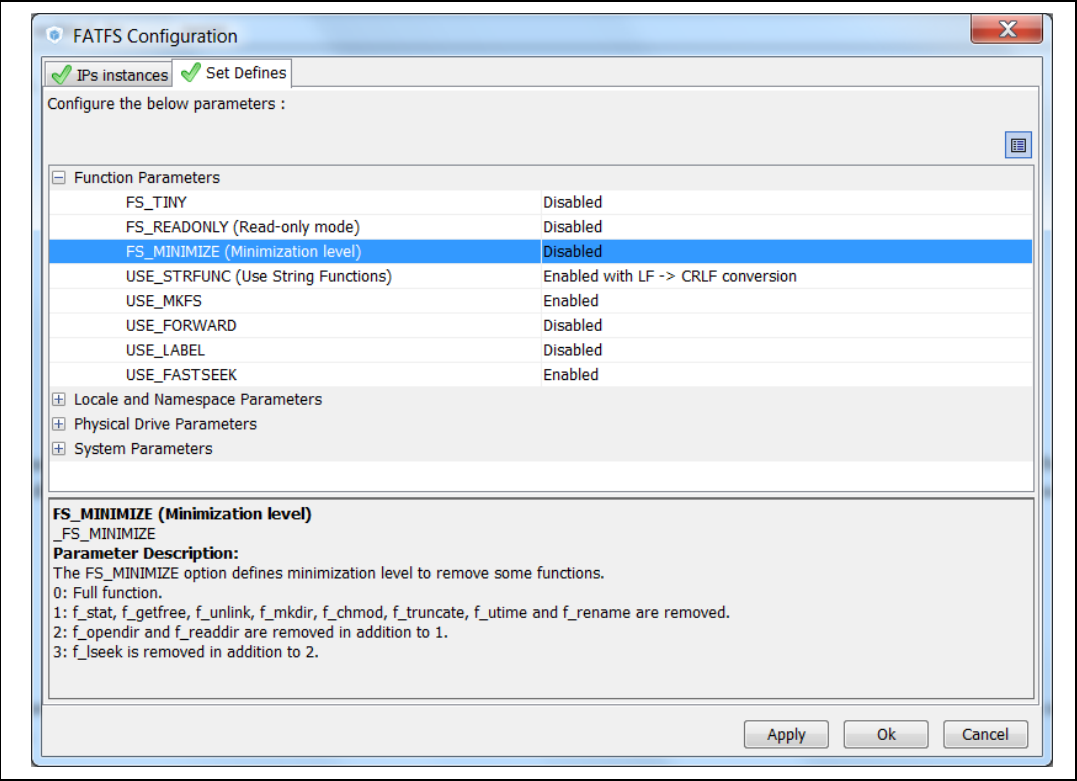


Figure 144. FatFs define statements



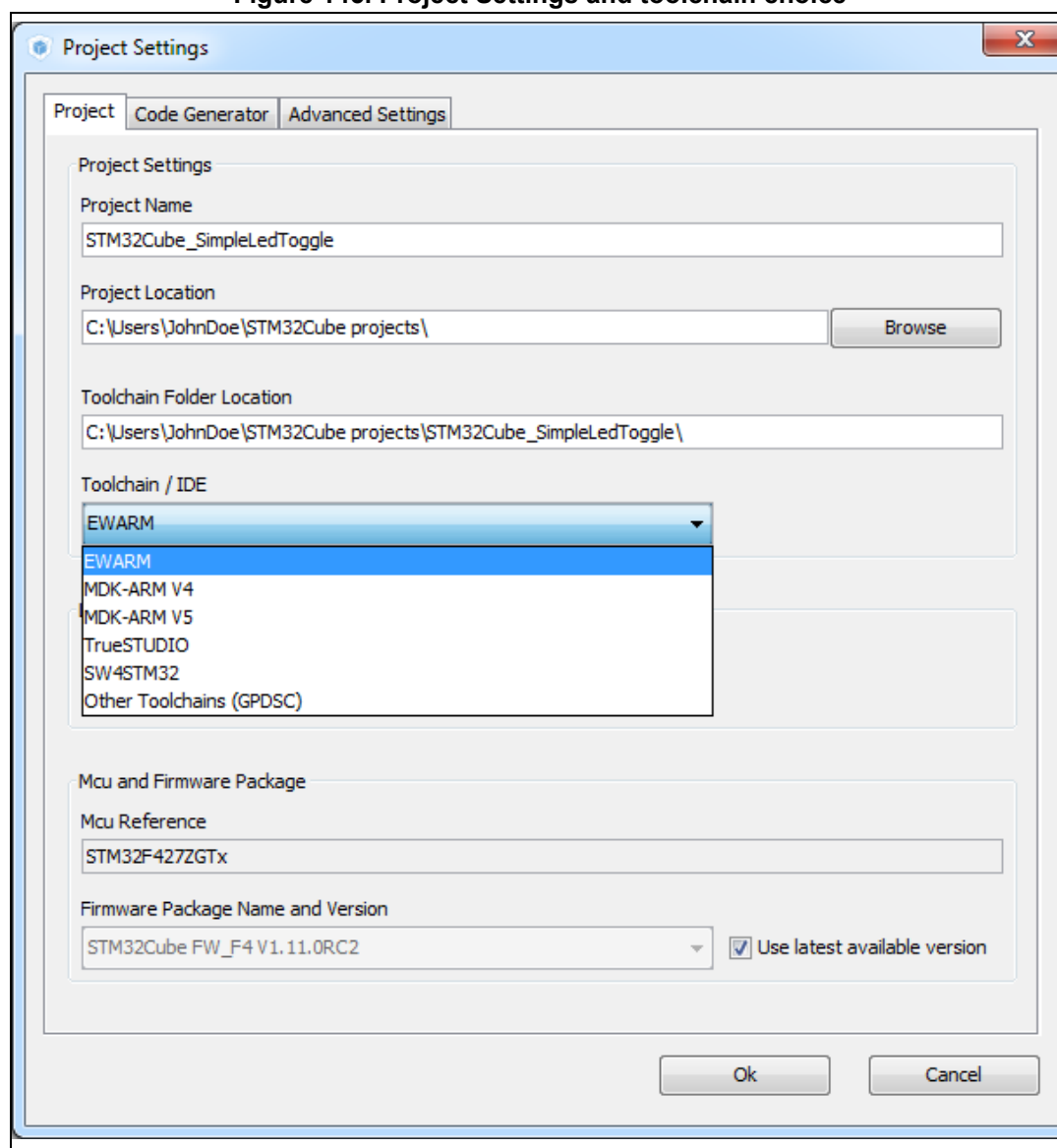
6.7 Generating a complete C project

6.7.1 Setting project options

Default project settings can be adjusted prior to C code generation as described in [Figure 145](#).

1. Select **Settings** from the **Project** menu to open the Project settings window.
2. Select the **Project Tab** and choose a Project **name**, **location** and a **toolchain** to generate the project (see [Figure 145](#)).

Figure 145. Project Settings and toolchain choice

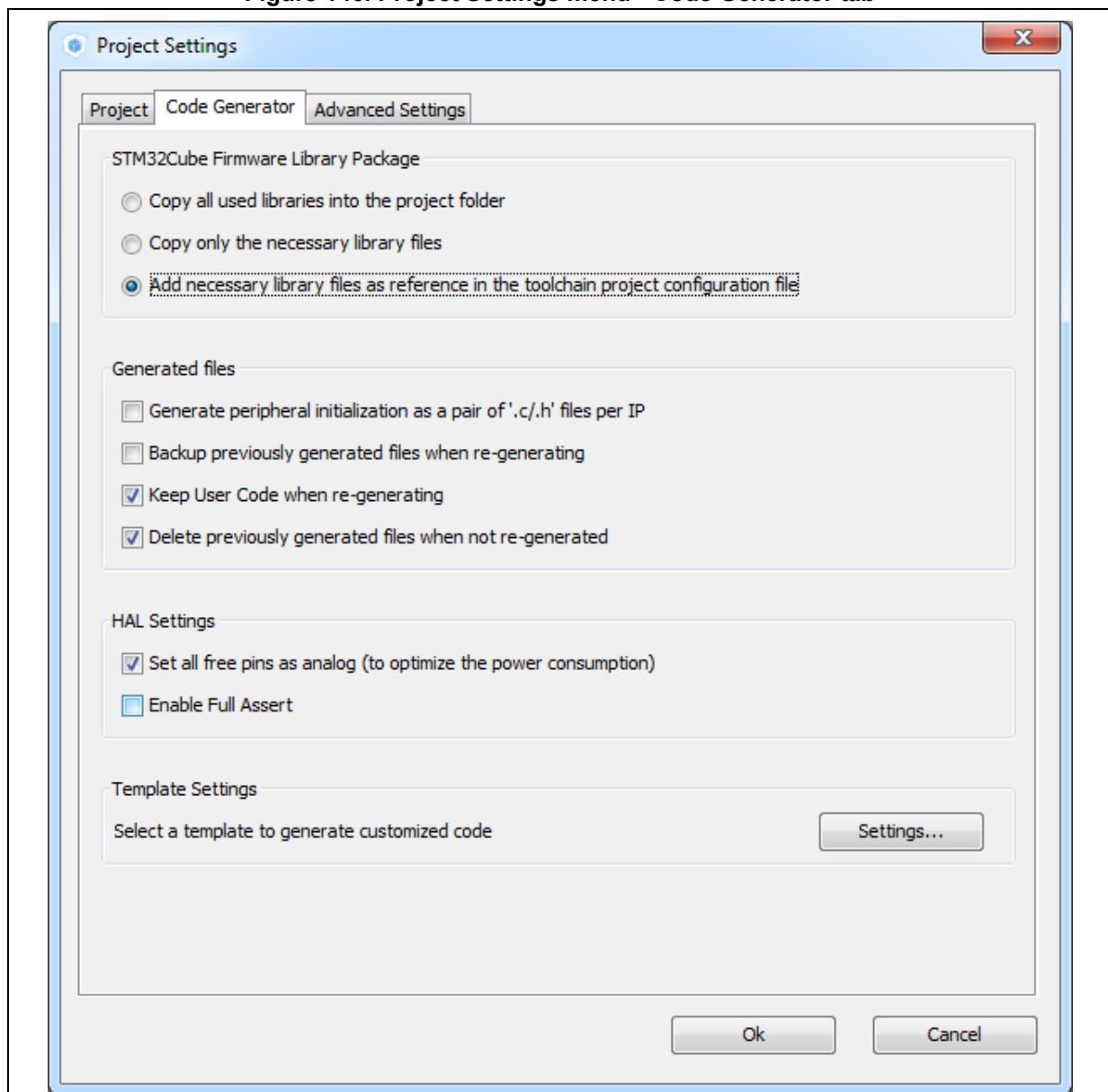


3. Select the **Code Generator** tab to choose various C code generation options:
 - The library files copied to *Projects* folder.
 - C code regeneration (e.g. what is kept or backed up during C code regeneration).
 - HAL specific action (e.g. set all free pins as analog I/Os to reduce MCU power consumption).

In the tutorial example, select the settings as displayed in the figure below and click OK.

Note: A dialog window appears when the firmware package is missing. Go to next section for explanation on how to download the firmware package.

Figure 146. Project Settings menu - Code Generator tab

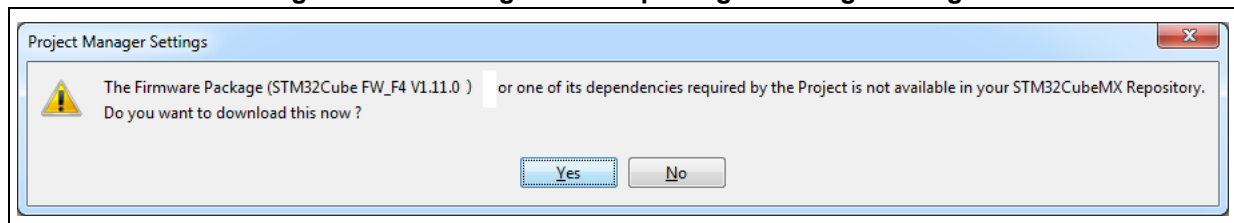


6.7.2 Downloading firmware package and generating the C code

1. Click  to generate the C code.

During C code generation, STM32CubeMX copies files from the relevant STM32Cube firmware package into the project folder so that the project can be compiled. When generating a project for the first time, the firmware package is not available on the user PC and a warning message is displayed:

Figure 147. Missing firmware package warning message

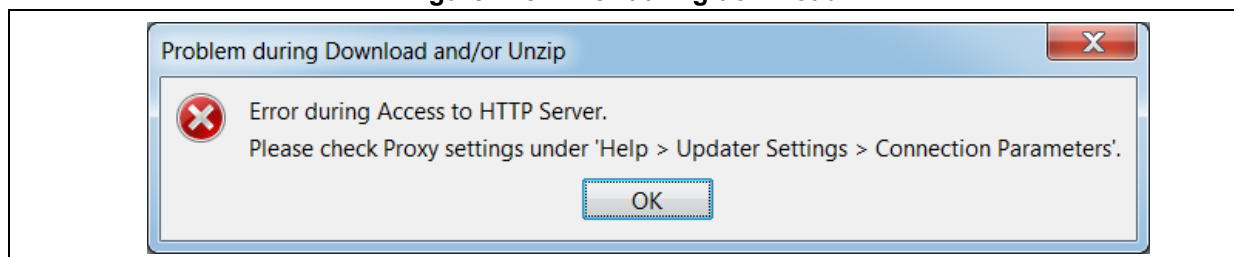


2. STM32CubeMX offers to download the relevant firmware package or to go on. Click **Download** to obtain a complete project, that is a project ready to be used in the selected IDE.

By clicking **Continue**, only *Inc* and *Src* folders will be created, holding STM32CubeMX generated initialization files. The necessary firmware and middleware libraries will have to be copied manually to obtain a complete project.

If the download fails, the below error message is displayed:

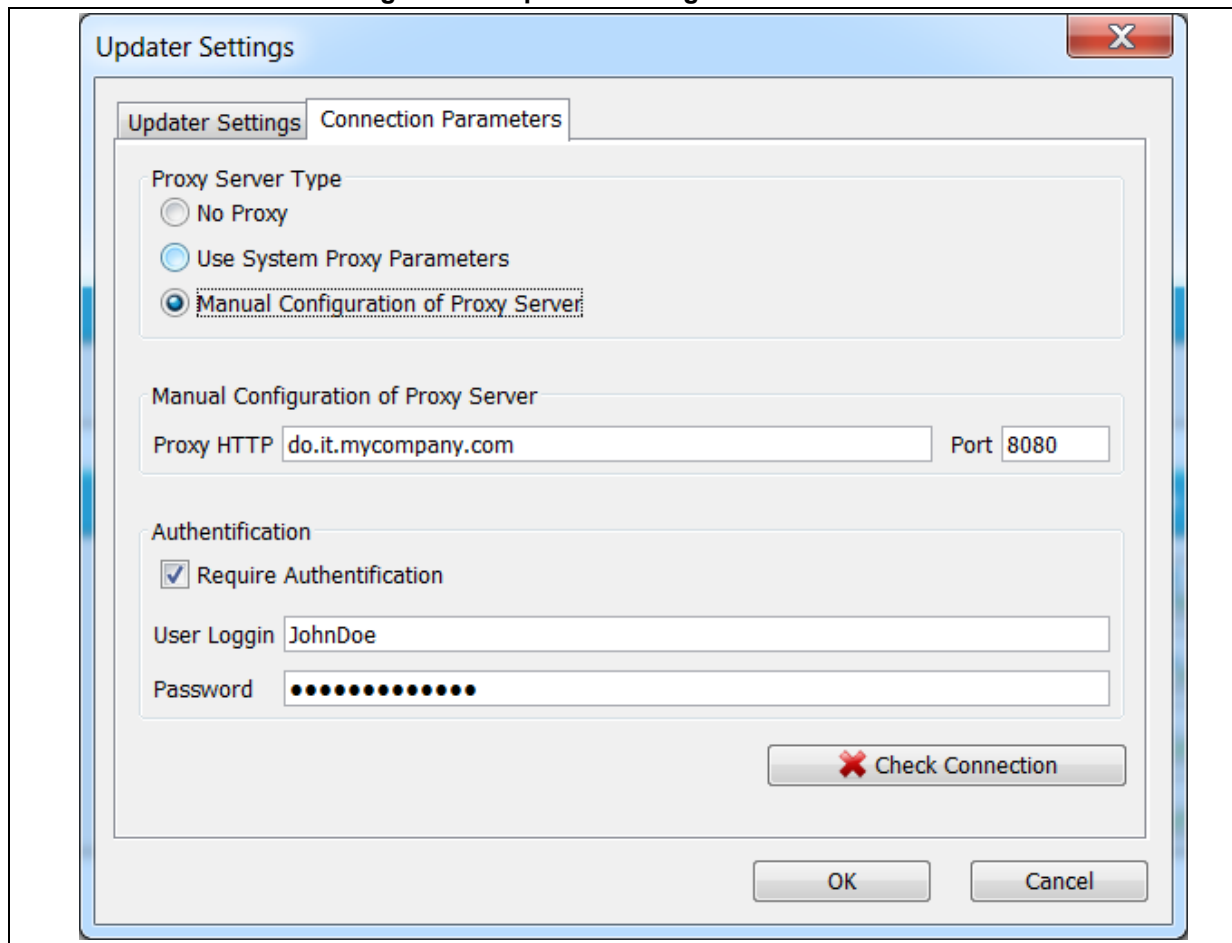
Figure 148. Error during download



To solve this issue, execute the next two steps. Skip them otherwise.

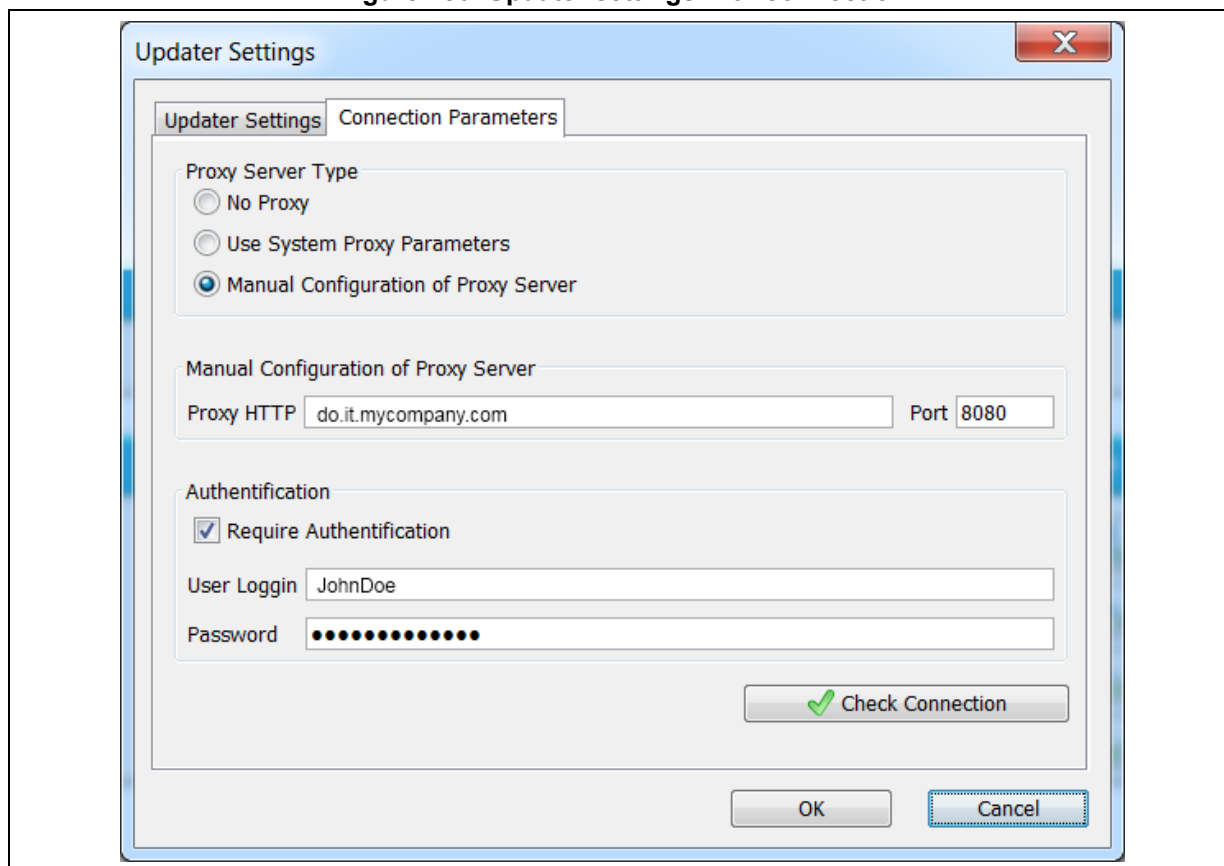
3. Select **Help > Updater settings menu** and adjust the connection parameters to match your network configuration.

Figure 149. Updater settings for download



4. Click **Check connection**. The check mark turns green once the connection is established.

Figure 150. Updater settings with connection




5. Once the connection is functional, click  to generate the C code. The C code generation process starts and progress is displayed as illustrated in the next figures.

Figure 151. Downloading the firmware package

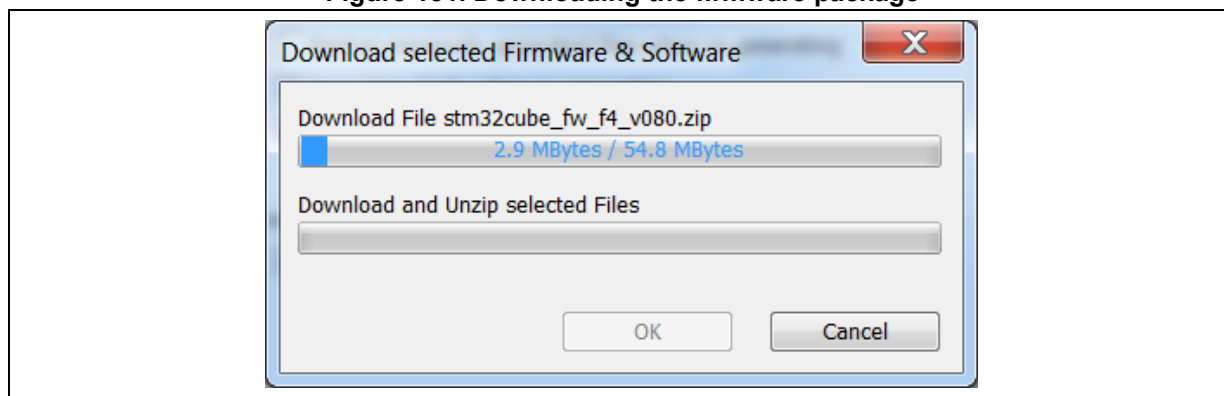
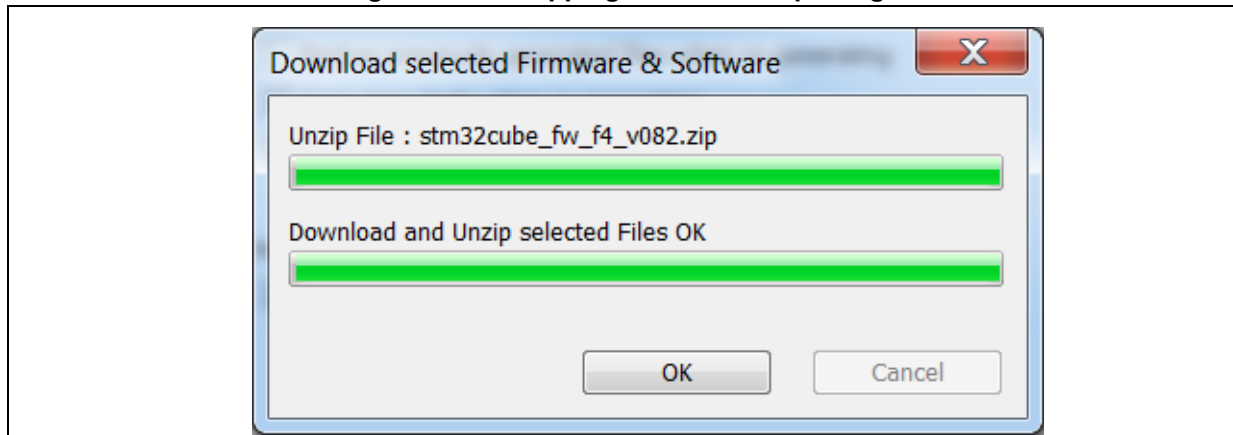
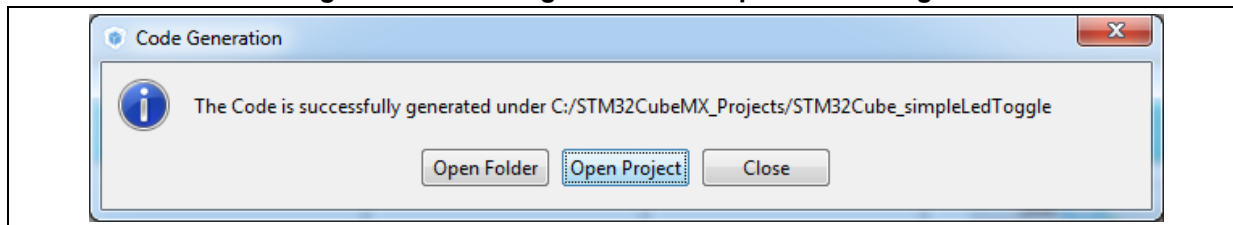


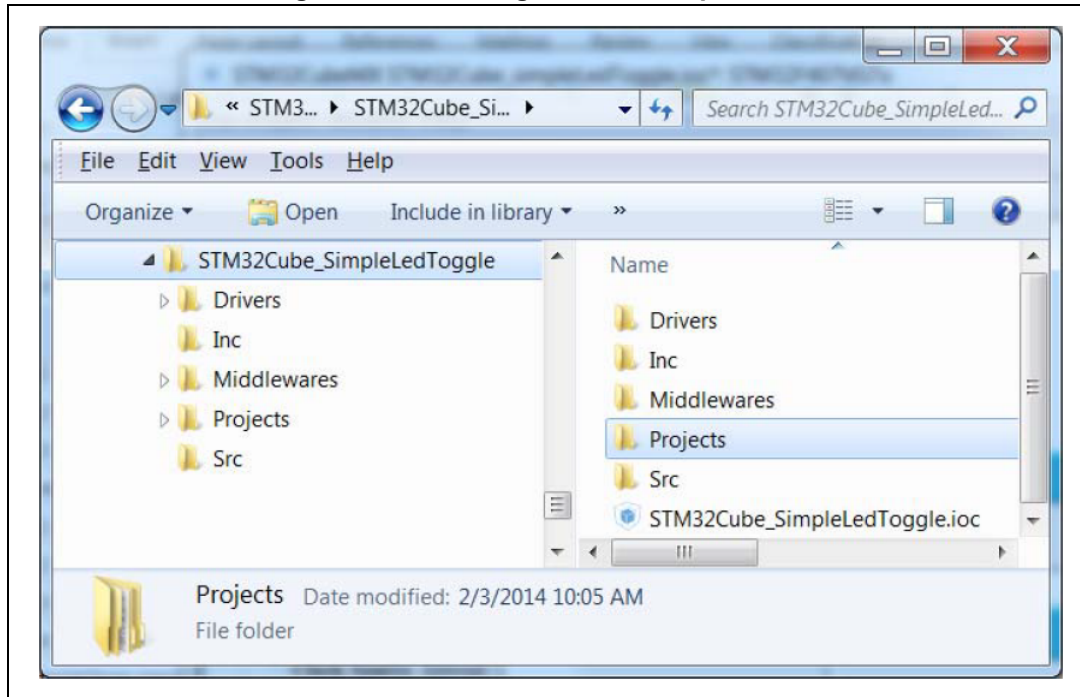
Figure 152. Unzipping the firmware package

6. Finally, a confirmation message is displayed to indicate that the C code generation has been successful.

Figure 153. C code generation completion message

7. Click **Open Folder** to display the generated project contents or click **Open Project** to open the project directly in your IDE. Then proceed with [Section 6.8](#).

Figure 154. C code generation output folder



The generated project contains:

- The STM32CubeMX .ioc project file located in the root folder. It contains the project user configuration and settings generated through STM32CubeMX user interface.
- The *Drivers* and *Middlewares* folders hold copies of the firmware package files relevant for the user configuration.
- The *Projects* folder contains IDE specific folders with all the files required for the project development and debug within the IDE.
- The *Inc* and *Src* folders contain STM32CubeMX generated files for middleware, peripheral and GPIO initialization, including the main.c file. The STM32CubeMX generated files contain user-dedicated sections allowing to insert user-defined C code.

Caution: C code written within the user sections is preserved at next C code generation, while C code written outside these sections is overwritten.

User C code will be lost if user sections are moved or if user sections delimiters are renamed.

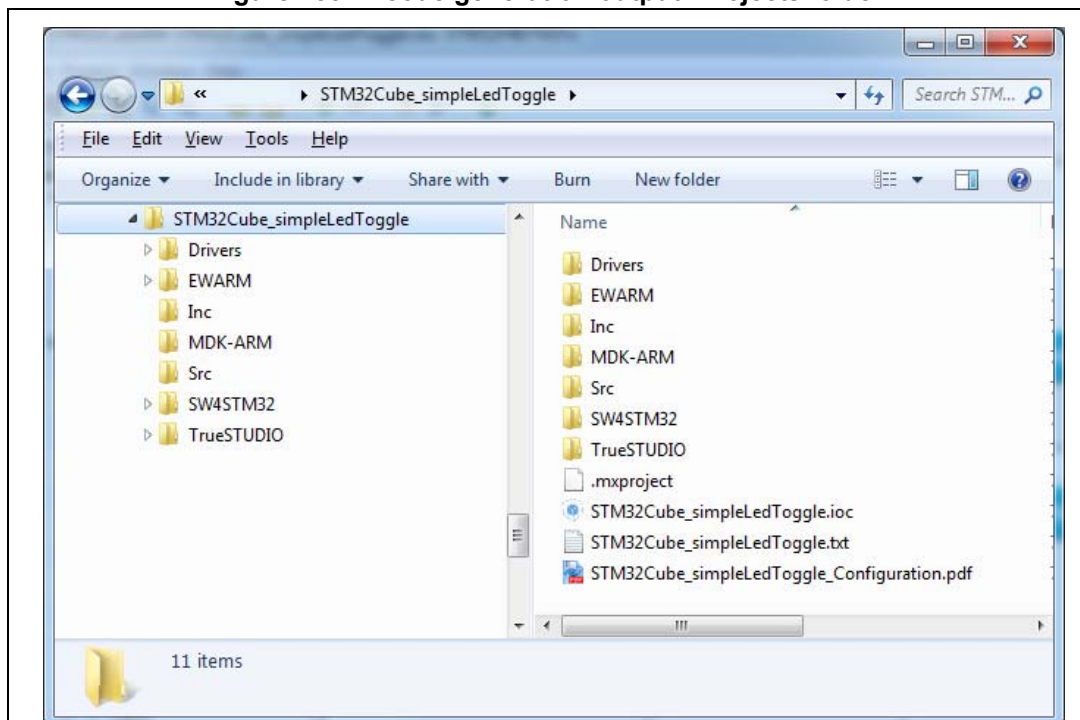
6.8 Building and updating the C code project

This example explains how to use the generated initialization C code and complete the project, within IAR EWARM toolchain, to have the LED blink according to the TIM3 frequency.

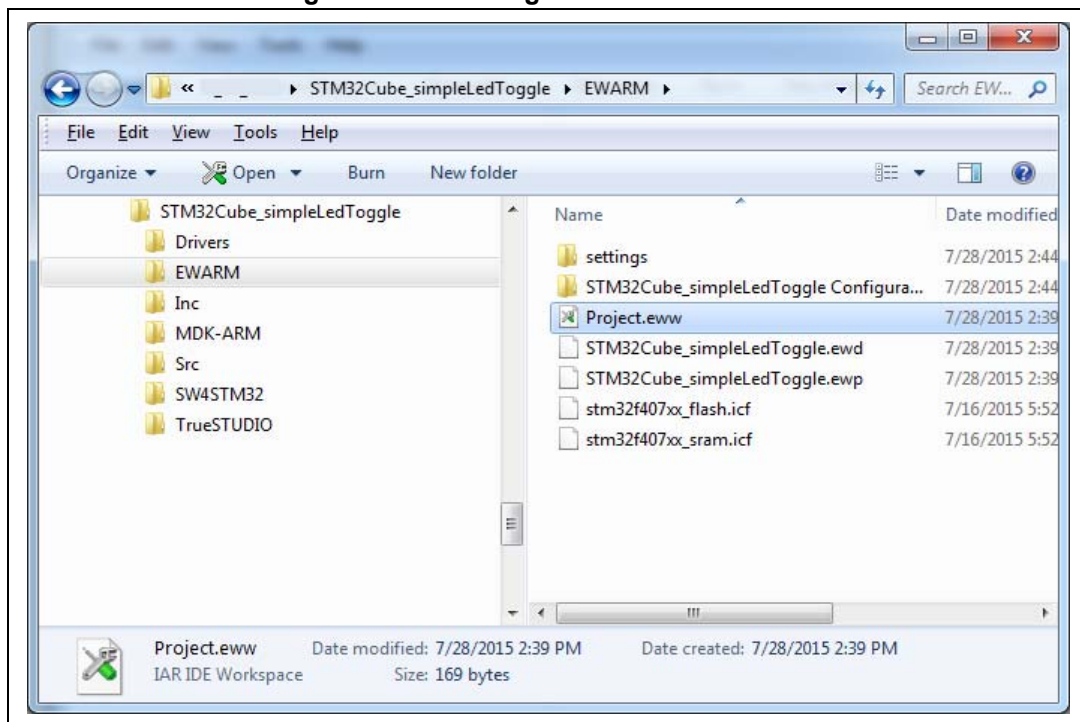
A folder is available for the toolchains selected for C code generation: the project can be generated for more than one toolchain by choosing a different toolchain from the Project Settings menu and clicking Generate code once again.

1. Open the project directly in the IDE toolchain by clicking **Open Project** from the dialog window or by double-clicking the relevant IDE file available in the toolchain folder under STM32CubeMX generated project directory (see [Figure 153](#)).

Figure 155. C code generation output: Projects folder

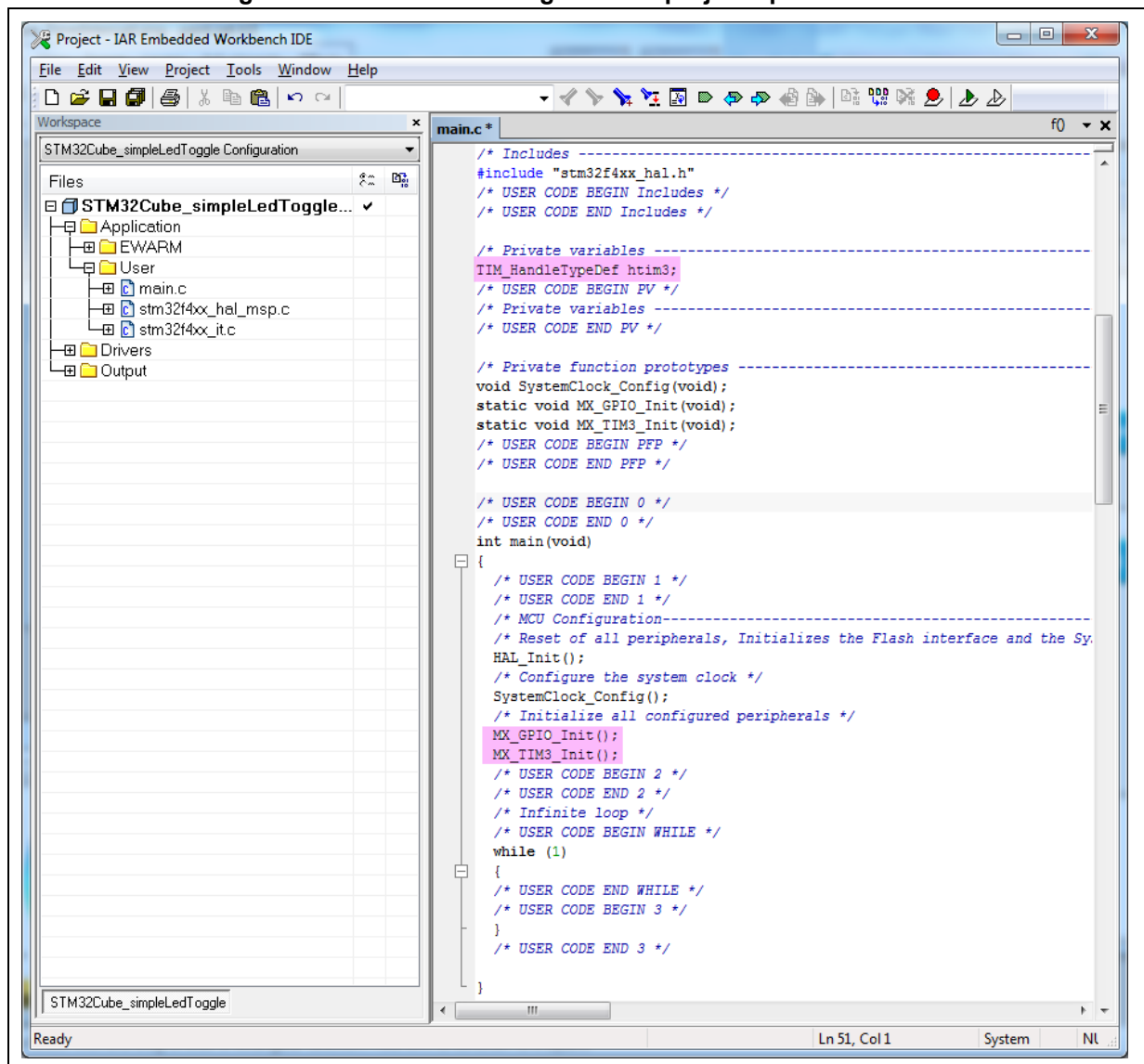


2. As an example, select .eww file to load the project in the IAR EWARM IDE.

Figure 156. C code generation for EWARM

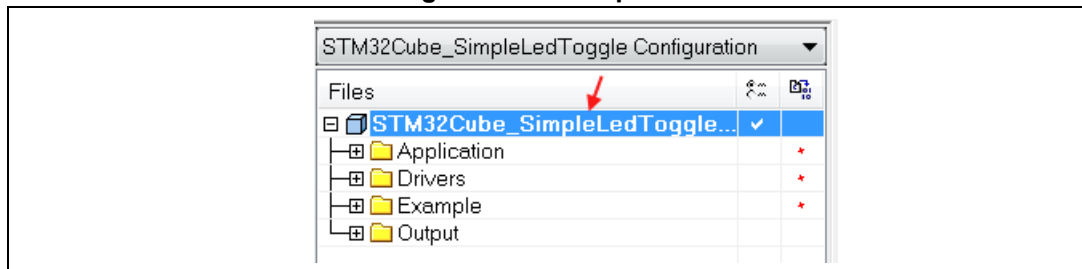
3. Select the main.c file to open in editor.

Figure 157. STM32CubeMX generated project open in IAR IDE

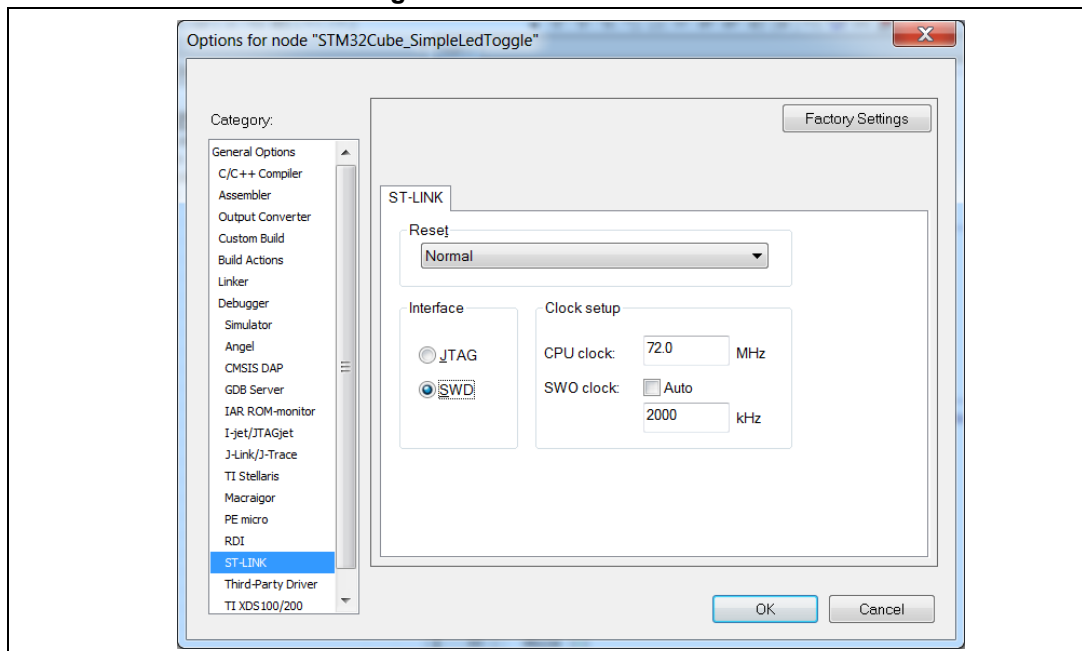


The htim3 structure handler, system clock, GPIO and TIM3 initialization functions are defined. The initialization functions are called in the main.c. For now the user C code sections are empty.

4. In the IAR IDE, right-click the project name and select **Options**.

Figure 158. IAR options

5. Click the ST-LINK category and make sure SWD is selected to communicate with the STM32F4DISCOVERY board. Click OK.

Figure 159. SWD connection

6. Select **Project > Rebuild all**. Check if the project building has succeeded.

Figure 160. Project building log

```
Messages
stm32f4xx_hal_tim.c
stm32f4xx_hal_tim_ex.c
stm32f4xx_it.c
stm32f4xx_ll_sdmmc.c
system_stm32f4xx.c
Linking

Total number of errors: 0
Total number of warnings: 0
```

7. Add user C code in the dedicated user sections **only**.

Note: The main `while(1)` loop is placed in a user section.

For example:

- Edit the `main.c` file.
- To start timer 3, update User Section 2 with the following C code:

Figure 161. User Section 2

```
HAL_Init();
/* Configure the system clock */
SystemClock_Config();
/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_TIM3_Init();

/* USER CODE BEGIN 2 */
HAL_TIM_Base_Start_IT(&htim3);
/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
```

- Then, add the following C code in User Section 4:



Figure 162. User Section 4

```
/* USER CODE BEGIN 4 */

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    if ( htim->Instance == htim3.Instance )
    {
        HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_12);
    }
}

/* USER CODE END 4 */
```

This C code implements the weak callback function defined in the HAL timer driver (stm32f4xx_hal_tim.h) to toggle the GPIO pin driving the green LED when the timer counter period has elapsed.

8. Rebuild and program your board using . Make sure the SWD ST-LINK option is checked as a Project options otherwise board programming will fail.
9. Launch the program using . The green LED on the STM32F4DISCOVERY board will blink every second.
10. To change the MCU configuration, go back to STM32CubeMX user interface, implement the changes and regenerate the C code. The project will be updated, preserving the C code in the user sections if ☐ **Keep Current Signals Placement** option in Project Settings is enabled.

6.9 Switching to another MCU

STM32CubeMX allows loading a project configuration on an MCU of the same series.

Proceed as follows:

1. Select **File > New Project**.
2. Select an MCU belonging to the same series. As an example, you can select the STM32F429ZITx that is the core MCU of the 32F429IDISCOVERY board.
3. Select **File > Import project**. In the **Import project** window, browse to the .ioc file to load. A message warns you that the currently selected MCU (STM32F429ZITx) differs from the one specified in the .ioc file (STM32F407VGTx). Several import options are proposed (see [Figure 163](#)).
4. Click the **Try Import** button and check the import status to verify if the import succeeded (see [Figure 164](#)).
5. Click OK to really import the project. An output tab is then displayed to report the import results.
6. The green LED on 32F429IDISCOVERY board is connected to PG13: CTRL+ right click PD12 and drag and drop it on PG13.
7. Select **Project > Settings** to configure the new project name and folder location. Click **Generate icon** to save the project and generate the code.
8. Select **Open the project** from the dialog window, update the user sections with the user code, making sure to update the GPIO settings for PG13. Build the project and flash the board. Launch the program and check that LED blinks once per second.

Figure 163. Import Project menu

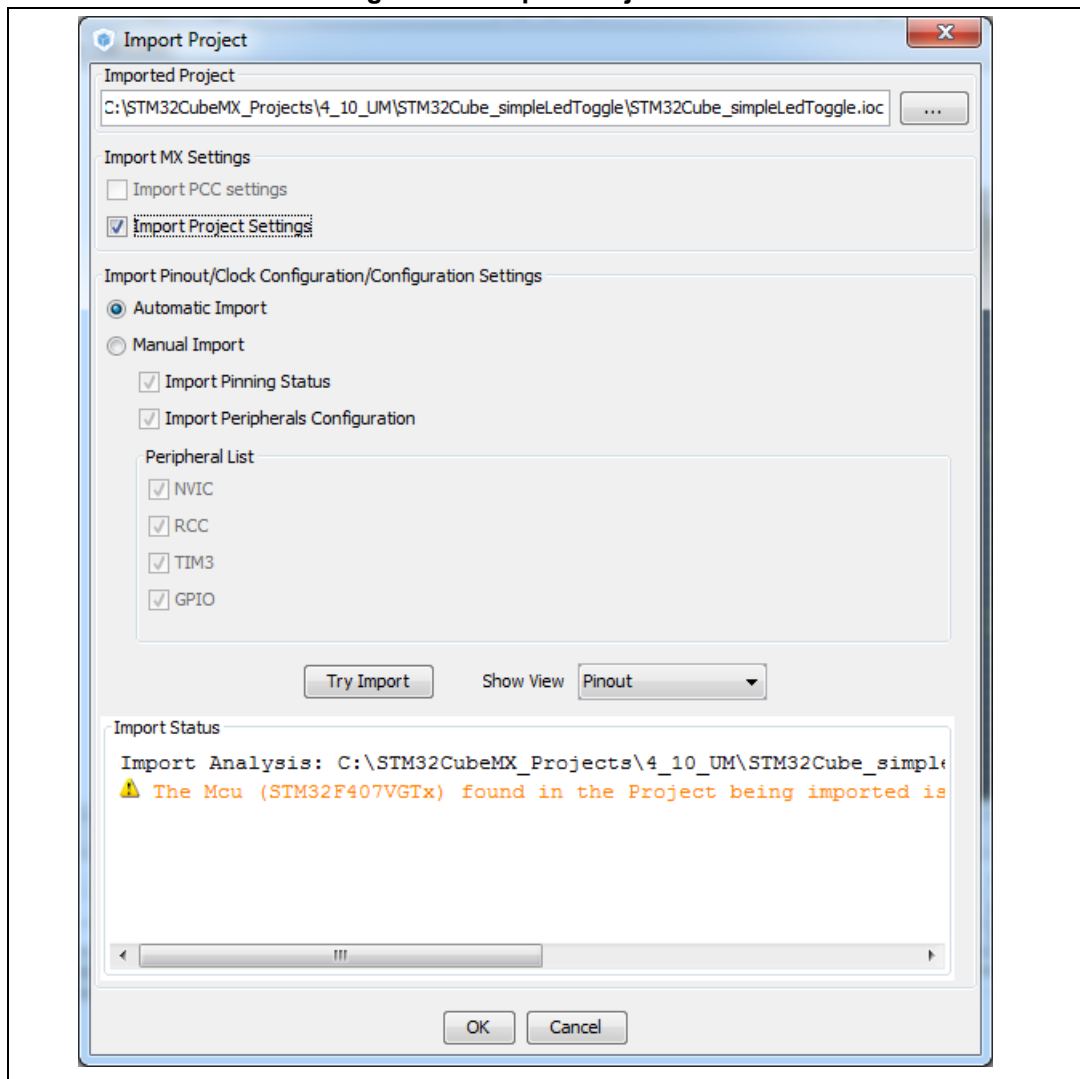
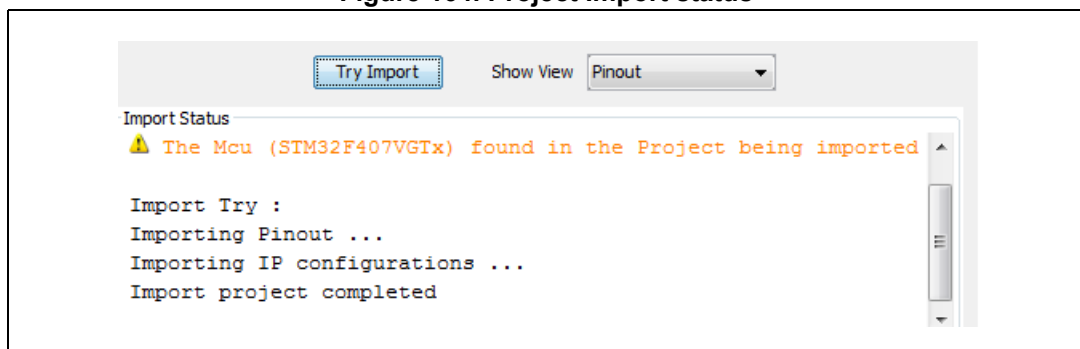


Figure 164. Project Import status



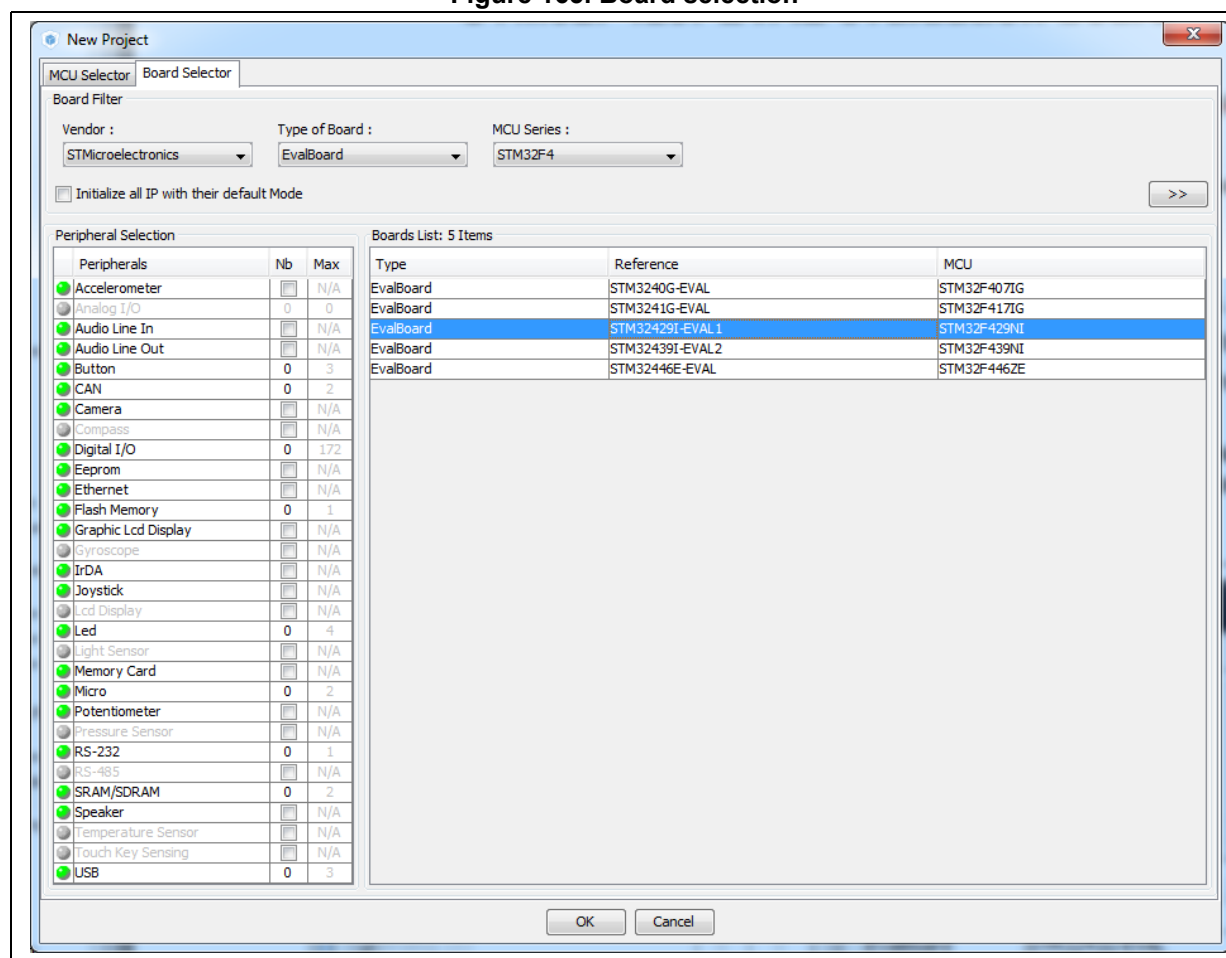
7 Tutorial 2 - Example of FatFs on an SD card using STM32429I-EVAL evaluation board

The tutorial consists in creating and writing to a file on the STM32429I-EVAL1 SD card using the FatFs file system middleware.

To generate a project and run tutorial 2, follow the sequence below:

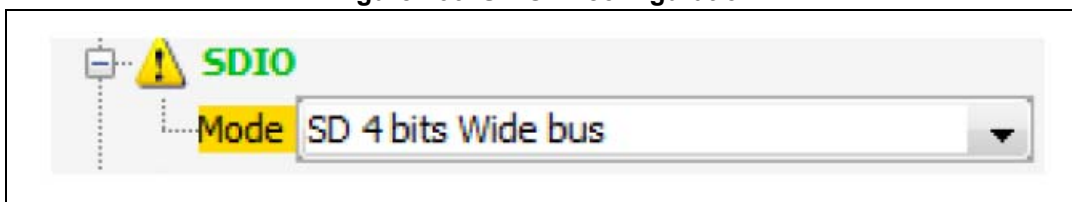
1. Launch STM32CubeMX.
2. Select **File > New Project**. The Project window opens.
3. Click the **Board Selector** Tab to display the list of ST boards.
4. Select **EvalBoard** as type of Board and **STM32F4** as series to filter down the list.
5. Leave the option **Initialize all IPs with their default mode** unchecked so that the code is generated only for the IPs used by the application.
6. Select the STM32429I-EVAL board and click OK. The **Pinout** view is loaded, matching the MCU pinout configuration on the evaluation board (see [Figure 165](#)).

Figure 165. Board selection



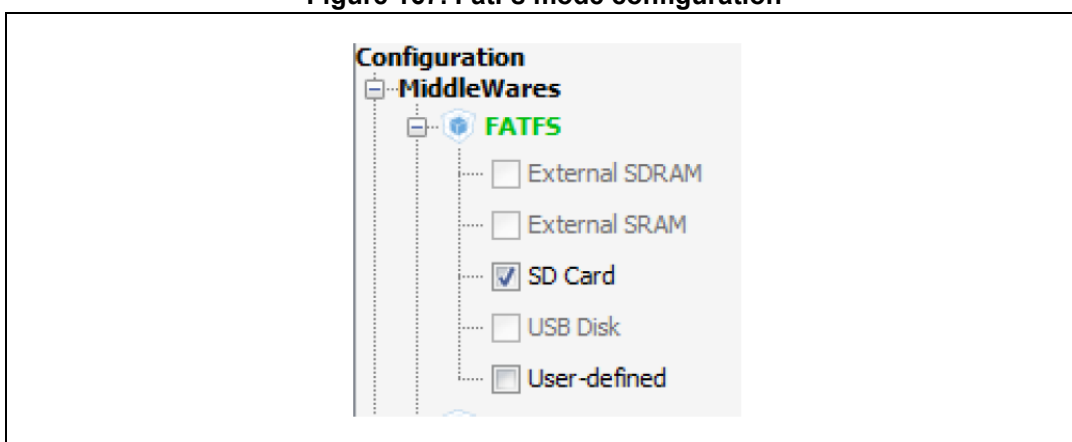
7. From the Peripheral tree on the left, expand the SDIO IP and select the SD 4 bits wide bus (see [Figure 166](#)).

Figure 166. SDIO IP configuration



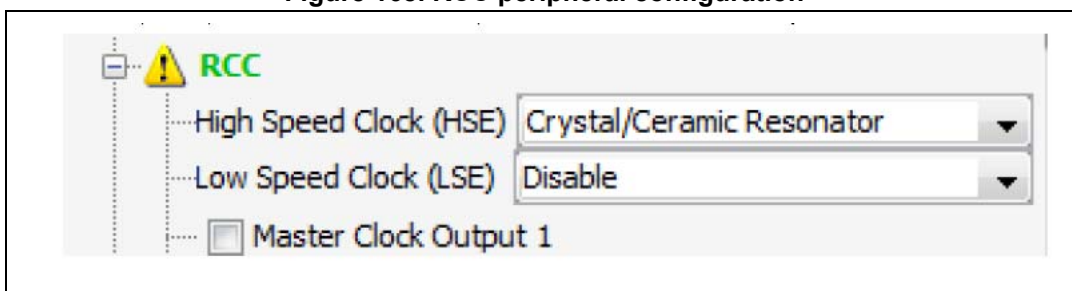
8. Under the Middlewares category, check “SD Card” as FatFs mode (see [Figure 167](#)).

Figure 167. FatFs mode configuration



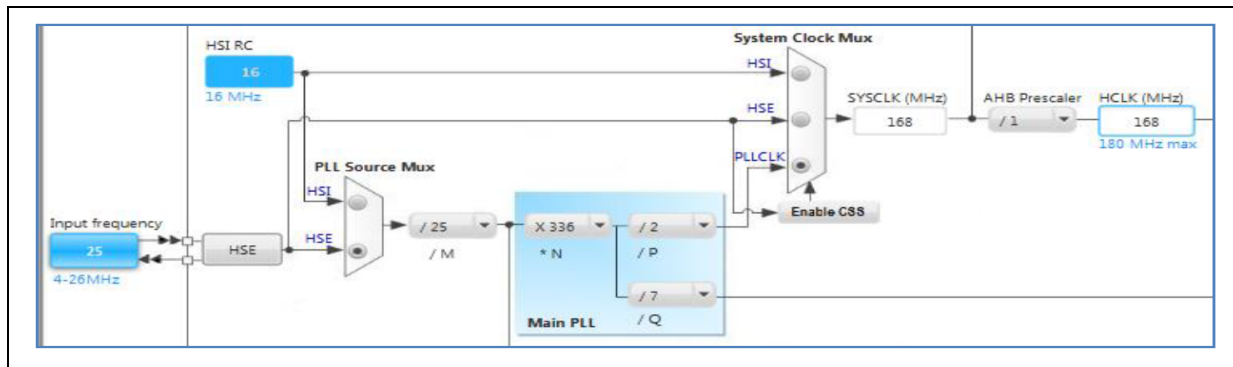
9. Configure the clocks as follows:
 - a) Select the RCC peripheral from the **Pinout** view (see [Figure 168](#)).

Figure 168. RCC peripheral configuration



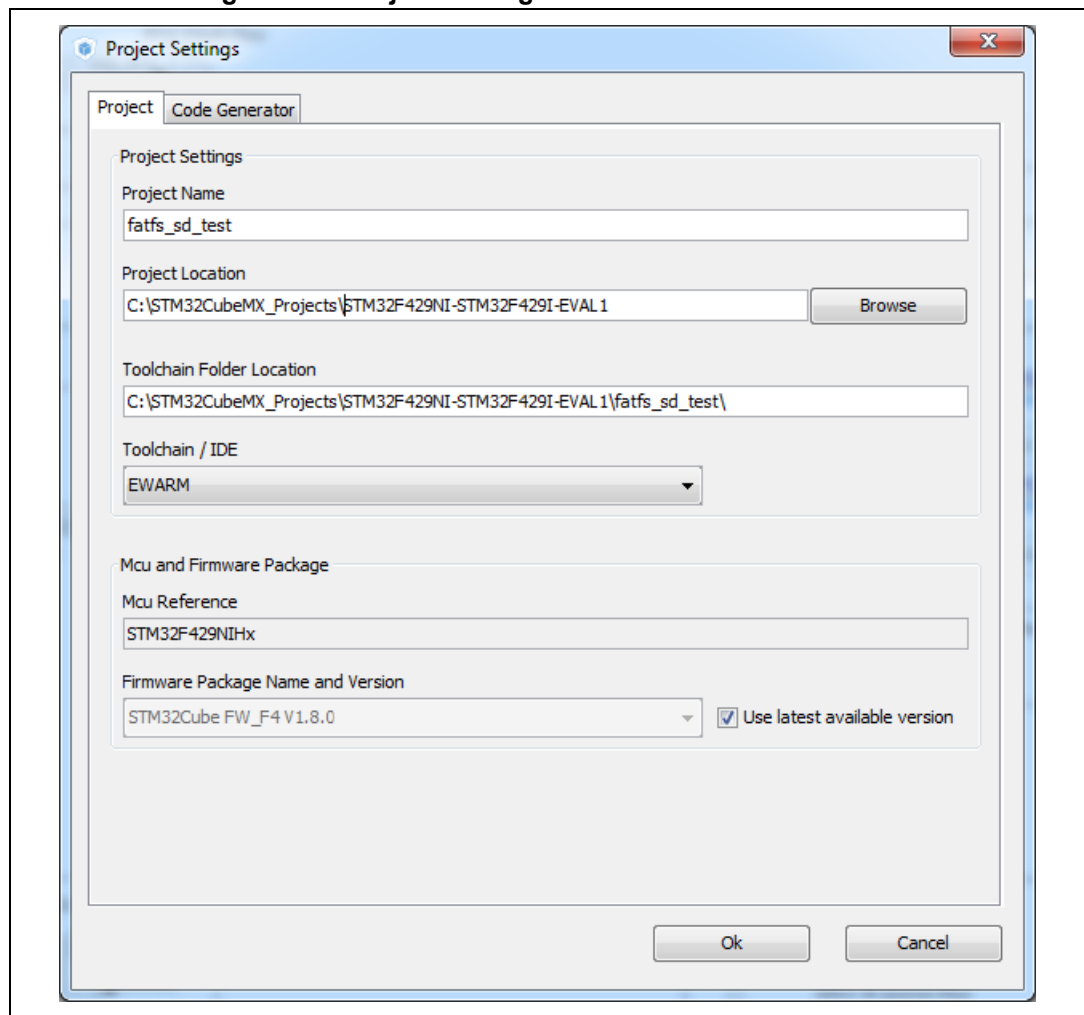
- b) Configure the clock tree from the clock tab (see [Figure 169](#)).

Figure 169. Clock tree view



10. In the **Project Settings** menu, specify the project name and destination folder. Then, select the EWARM IDE toolchain.

Figure 170. Project Settings menu - Code Generator tab




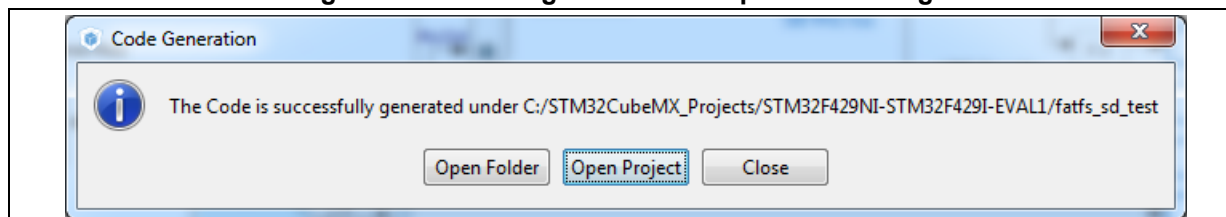
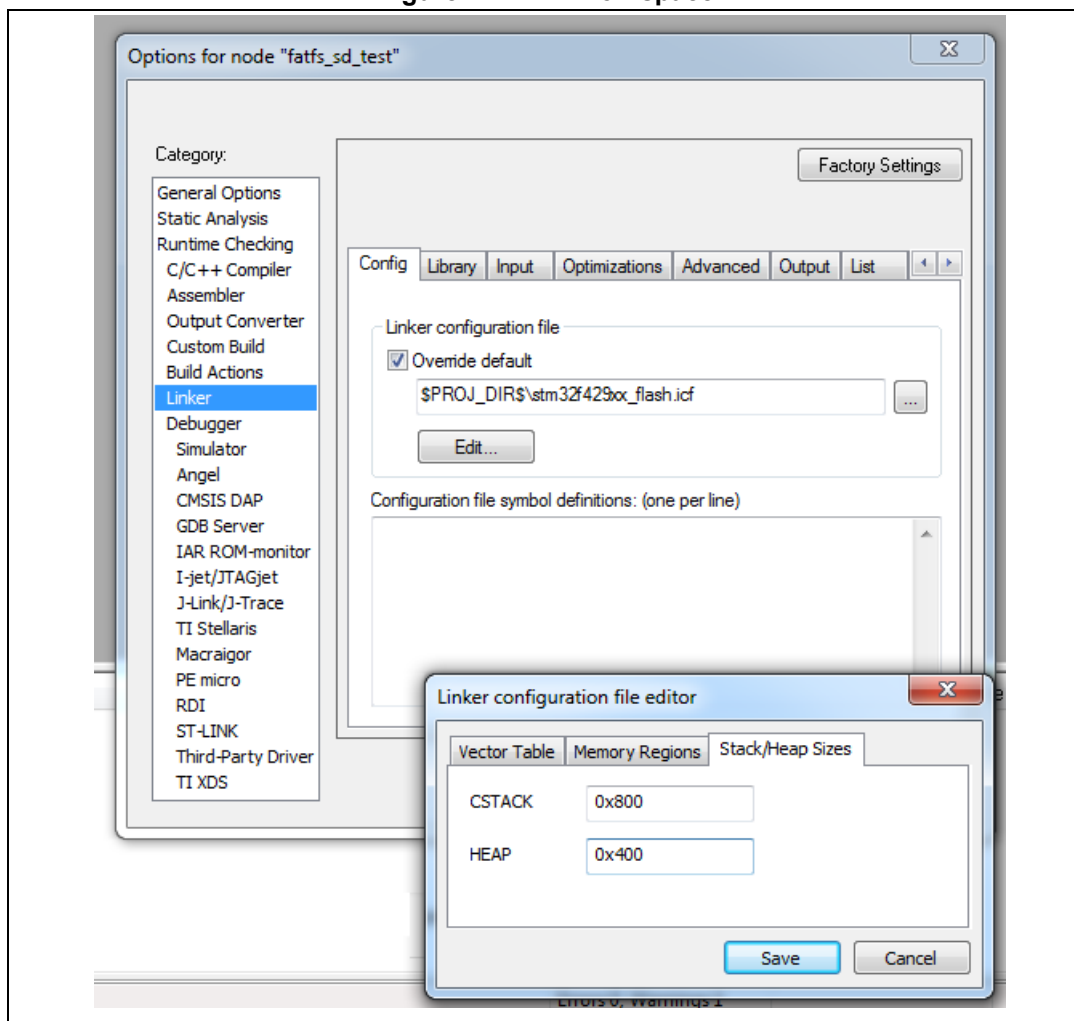
11. Click **Ok**. Then, in the toolbar menu, click  to generate the project.
12. Upon code generation completion, click **Open Project** in the **Code Generation** dialog window (see [Figure 171](#)). This opens the project directly in the IDE.

Figure 171. C code generation completion message

13. In the IDE, check that heap and stack sizes are sufficient: right click the project name and select Options, then select Linker. Check **Override default** to use the icf file from STM32CubeMX generated project folder. Adjust the heap and stack sizes (see [Figure 172](#)).

Figure 172. IDE workspace

Note: When using the MDK-ARM toolchain, go to the Application/MDK-ARM folder and double click the startup_xx.s file to edit and adjust the heap and stack sizes there.

14. Go to the Application/User folder. Double click the main.c file and edit it.
15. The tutorial consists in creating and writing to a file on the evaluation board SD card using the FatFs file system middleware:
 - a) At startup all LEDs are OFF.
 - b) The red LED is turned ON to indicate that an error occurred (FatFs initialization, file read/write access errors..).
 - c) The orange LED is turned ON to indicate that the FatFs link has been successfully mounted on the SD driver.
 - d) The blue LED is turned ON to indicate that the file has been successfully written to the SD Card.
 - e) The green LED is turned ON to indicate that the file has been successfully read from file the SD Card.
16. For use case implementation, update main.c with the following code:
 - a) Insert main.c private variables in a dedicated user code section:

```
/* USER CODE BEGIN PV */
/* Private variables -----*/
FATFS SDFatFs; /* File system object for SD card logical drive */
FIL MyFile;    /* File object */
const char wtext[] = "Hello World!";
const uint8_t image1_bmp[] = {
0x42,0x4d,0x36,0x84,0x03,0x00,0x00,0x00,0x00,0x00,0x36,0x00,0x00,0x00,
0x28,0x00,0x00,0x00,0x40,0x01,0x00,0x00,0xf0,0x00,0x00,0x00,0x01,0x00,
0x18,0x00,0x00,0x00,0x00,0x00,0x00,0x84,0x03,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x29,0x74,
0x51,0x0e,0x63,0x30,0x04,0x4c,0x1d,0x0f,0x56,0x25,0x11,0x79,0x41,0x1f,
0x85,0x6f,0x25,0x79,0x7e,0x27,0x72,0x72,0x0b,0x50,0x43,0x00,0x44,0x15,
0x00,0x4b,0x0f,0x00,0x4a,0x15,0x07,0x50,0x16,0x03,0x54,0x22,0x23,0x70,
0x65,0x30,0x82,0x6d,0x0f,0x6c,0x3e,0x22,0x80,0x5d,0x23,0x8b,0x5b,0x26};
/* USER CODE END PV */
```

- b) Insert main functional local variables:

```
int main(void)
{

/* USER CODE BEGIN 1 */
FRESULT res; /* FatFs function common result code */
uint32_t byteswritten, bytesread; /* File write/read counts */
char rtext[256]; /* File read buffer */
/* USER CODE END 1 */

/* MCU Configuration-----*/

/* Reset of all peripherals, Initializes the Flash interface and the
Systick. */
HAL_Init();
```