**1.**

* **Broches en rouge** : Ces broches sont déjà utilisées pour une fonction spécifique (comme des ports d'entrée/sortie ou des fonctions internes du microcontrôleur). Cela signifie que ces broches ne peuvent pas être modifiées ou réassignées.
* **Broches en orange** : Ces broches sont des broches configurables (par exemple, elles peuvent être utilisées pour des entrées/sorties ou pour des fonctions alternatives). Elles sont disponibles pour être configurées selon les besoins du projet.

### 2. Où se situe le fichier main.c ?

Le fichier main.c se situe dans le répertoire **Core/Src/** du projet généré par STM32CubeIDE. Ce fichier contient le point d'entrée du programme et le code initialisé par STM32CubeMX.

### 3. ****À quoi servent les commentaires**** BEGIN ****et**** END ****?****

Les commentaires BEGIN et END sont des marqueurs utilisés par **STM32CubeMX** pour indiquer les sections où l'utilisateur peut ajouter du code sans risquer de le perdre lors d'une régénération du code.

### 4. ****Quels sont les paramètres à passer à**** HAL\_Delay ****et**** BSP\_LED\_XXX ****?****

* **HAL\_Delay(ms)** : Prend un paramètre en millisecondes pour spécifier le délai. Exemple : HAL\_Delay(1000); pour attendre 1 seconde.
* **BSP\_LED\_On(LED)**, **BSP\_LED\_Off(LED)**, **BSP\_LED\_Toggle(LED)** : Prend en paramètre la LED que tu veux contrôler (généralement définie comme LEDx dans les fichiers de la carte). Exemple : BSP\_LED\_On(LED2);.

### 5. ****Dans quel fichier les ports d’entrée/sorties sont-ils définis ?****

Les définitions des **ports GPIO** sont généralement situées dans le fichier **stm32g4xx\_hal\_gpio.h** ou dans les fichiers générés comme **main.h** et **gpio.h**, où les **broches (pins)** et **ports** sont mappés en fonction de la configuration définie dans STM32CubeMX.

**6.**

#include "main.h"

int main(void) {

HAL\_Init();

SystemClock\_Config(); // Configurer l'horloge du système

MX\_GPIO\_Init(); // Initialiser les GPIO

while (1) {

BSP\_LED\_Toggle(LED2); // Inverser l'état de la LED2

HAL\_Delay(1000); // Délai de 1 seconde

}

}

**7.**

int main(void) {

HAL\_Init();

SystemClock\_Config(); // Configurer l'horloge du système

MX\_GPIO\_Init(); // Initialiser les GPIO

while (1) {

if (HAL\_GPIO\_ReadPin(GPIOC, GPIO\_PIN\_13) == GPIO\_PIN\_RESET) { // Bouton pressé

BSP\_LED\_On(LED2); // Allumer la LED

} else {

BSP\_LED\_Off(LED2); // Éteindre la LED

}

HAL\_Delay(100); // Petit délai pour éviter les rebonds

}

}

**2 Avec des interruptions**

**1.** Pour configurer un timer comme TIM2, voici les étapes principales (comme détaillé précédemment) :

1. **Activer l'horloge du Timer** :  
   Utilise la fonction \_\_HAL\_RCC\_TIM2\_CLK\_ENABLE(); pour activer l'horloge TIM2.
2. **Configurer le Prescaler et la Période** :
   * Le **Prescaler** divise la fréquence d'horloge pour ajuster la fréquence du timer.
   * La **Période** détermine quand le timer génère un événement (mise à jour ou interruption).
3. **Configurer le mode d’interruption** (si nécessaire) :
   * Assurer que l’interruption est activée dans le NVIC :

c

Copy code

HAL\_NVIC\_SetPriority(TIM2\_IRQn, 0, 0);

HAL\_NVIC\_EnableIRQ(TIM2\_IRQn);

1. **Démarrer le Timer avec ou sans interruption** :
   * Sans interruption : HAL\_TIM\_Base\_Start(&htim2);
   * Avec interruption : HAL\_TIM\_Base\_Start\_IT(&htim2);