



Institut Supérieur des Etudes Technologiques
en Communications de Tunis

Guide de l'élève

Module

ATELIER BUS DE COMMUNICATION

Promotion

Master Professionnel Smart'Com

1^{ère} année

Enseignant responsable

Mohamed Chaker Bali

SOMMAIRE

TP 1 :

Initiation à l'environnement de développement μ Vision de Kiel

TP 2 :

Programmation de l'interface de communication UART

TP 3 :

Programmation de l'interface de communication SPI/I2C

TP 1

Initiation à l'environnement de développement μ Vision de Kiel

1. Objectifs

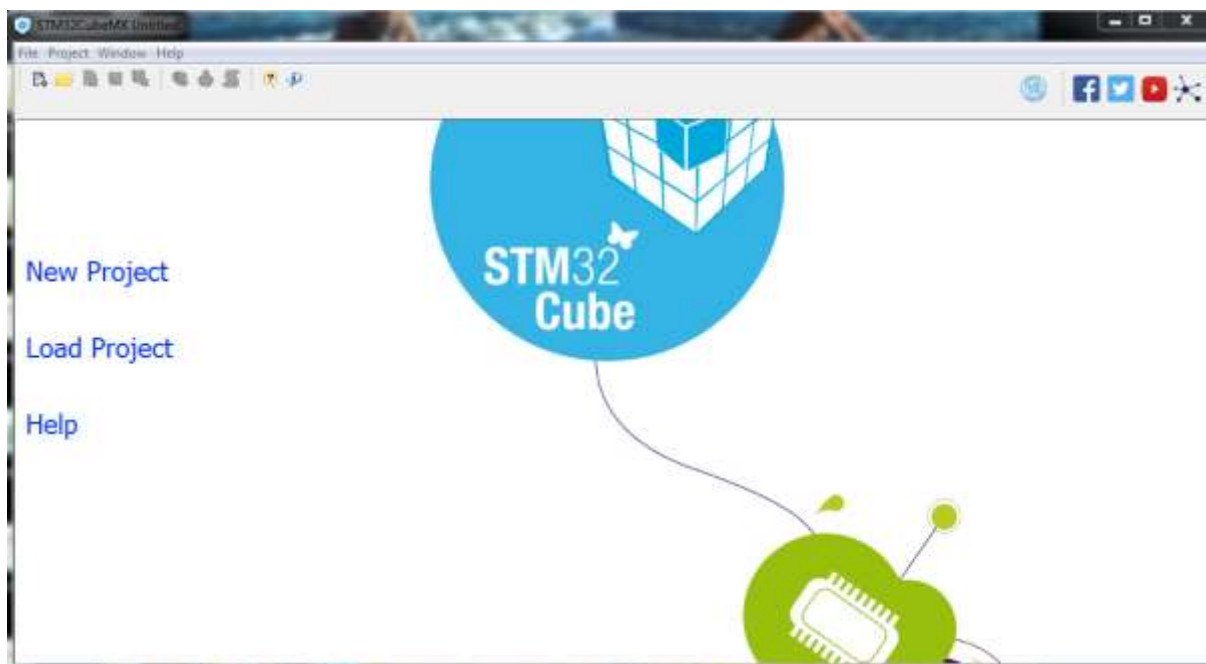
- Prise en main de l'environnement de développement μ Vision de Kiel.
- Compréhension des fichiers sources nécessaires à la programmation du microcontrôleur.
- Utilisation du debugger, visualisation des registres et des mémoires.

2. Travail demandé

2.1. Outil STM32CubeMX

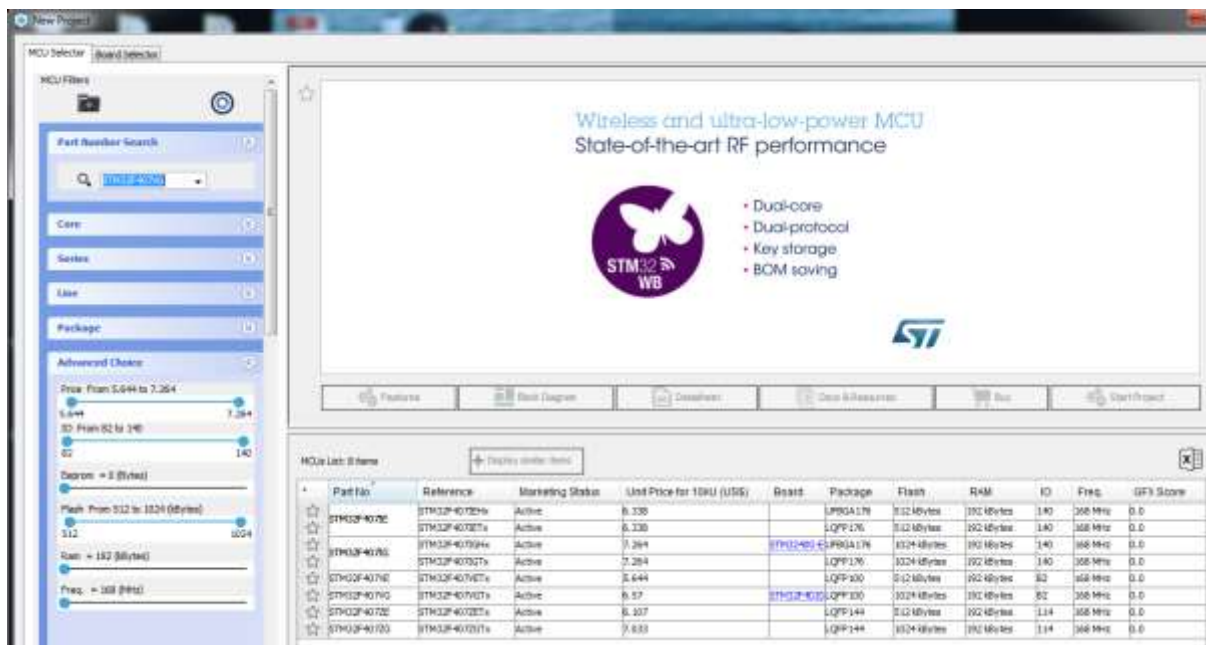
On se propose d'utiliser l'outil STM32CubeMX. Cet utilitaire permet de configurer graphiquement le microcontrôleur. On y spécifie notamment les sources d'horloge, les périphériques que l'on souhaite utiliser, les sources interruptions, etc.

1. Lancer STM32CubeMX

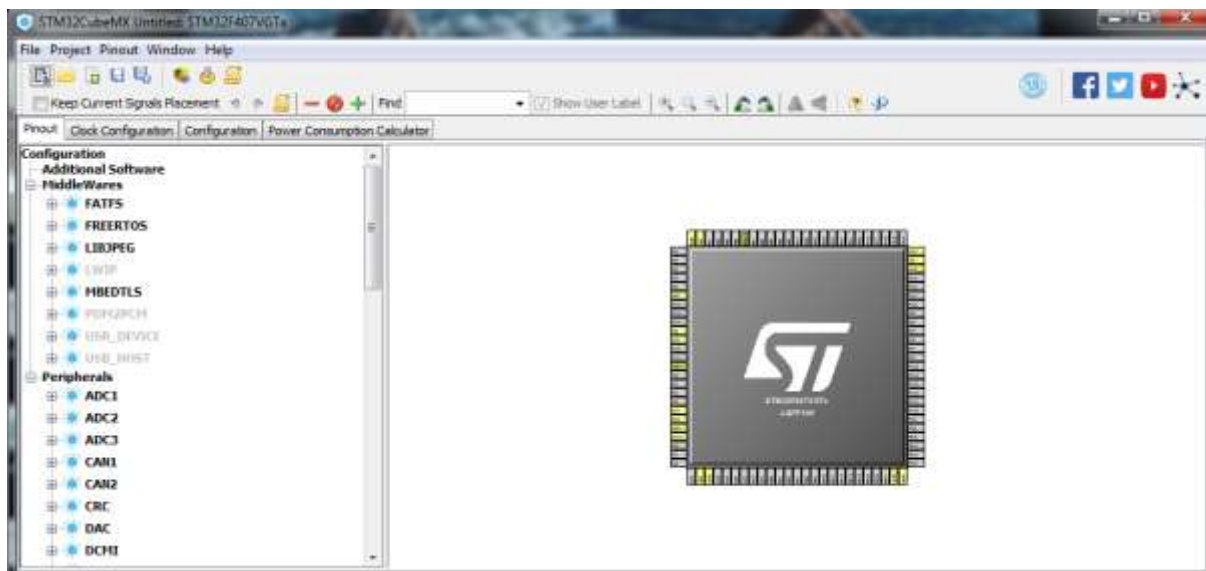


2. Créer un nouveau projet.

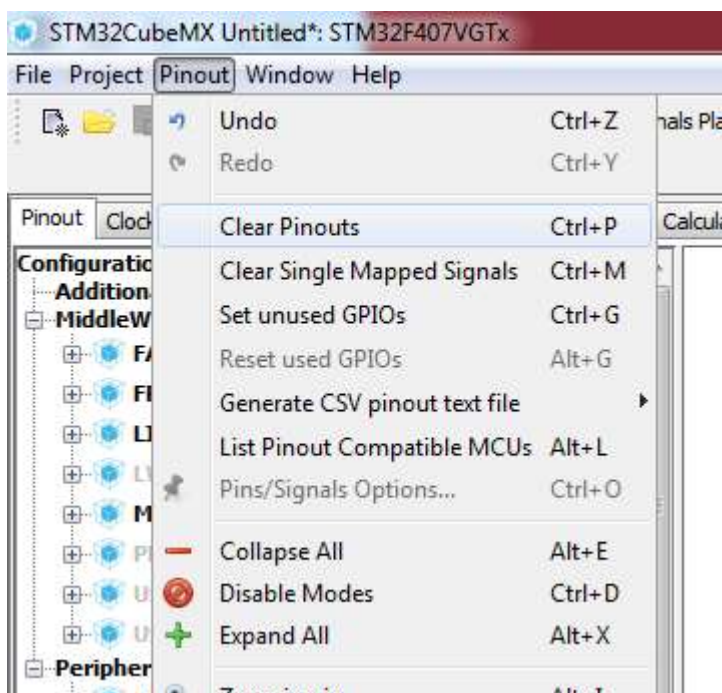
Chercher et sélectionner ensuite la plateforme que l'on souhaite utiliser (STM32F407VGT6 Discovery Board) puis OK.



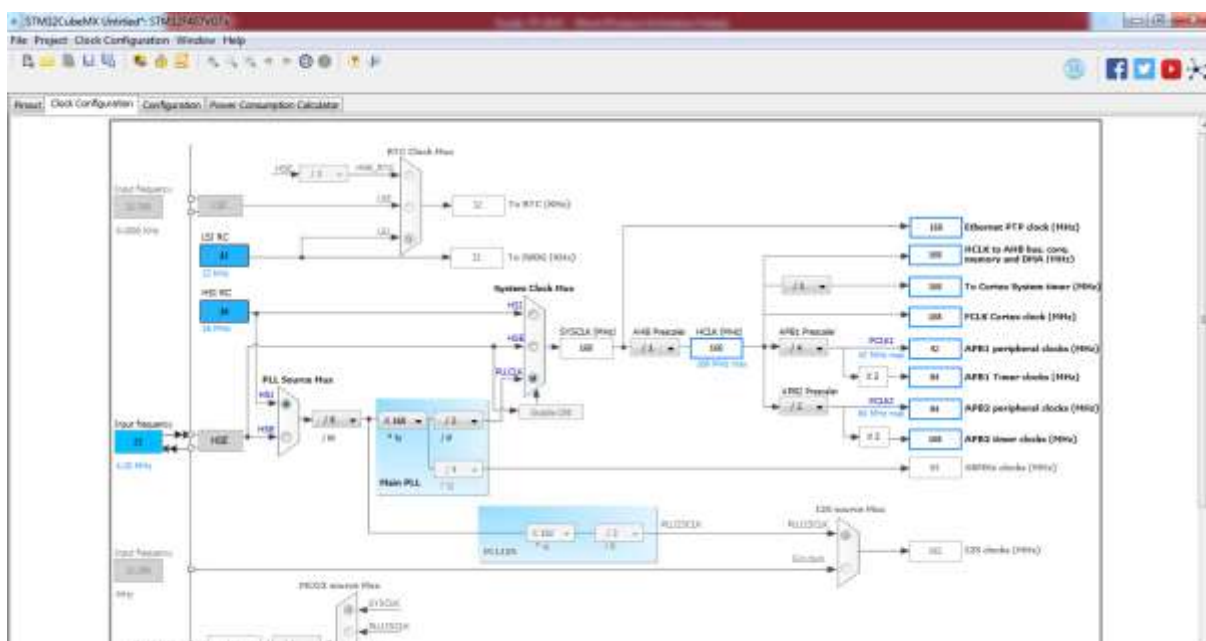
Vous arrivez sur l'écran principal de configuration du MCU.



3. Faire une remise à zéro des broches via l'onglet Pinout/Clear Pinouts



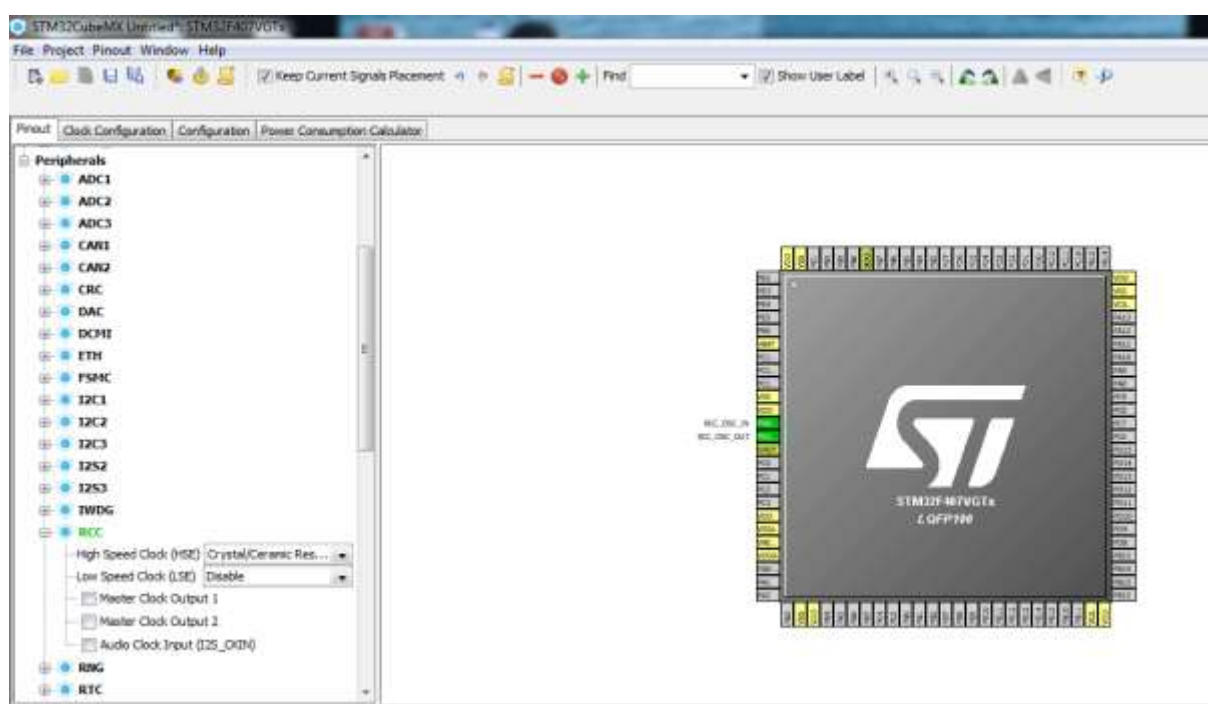
4. Vous pouvez observer dans la fenêtre Pinout, l'ensemble des périphériques qui peuvent être utilisés ainsi que les OS temps-réels et autres fonctionnalités logicielles.
5. A côté de Pinout se trouve clock configuration.



Dans cette fenêtre, vous pouvez graphiquement définir les sources d'horloge pour configurer l'horloge principale (SYSCLK) du microcontrôleur. Vous noterez qu'il existe plusieurs entrées

possibles (HSI/HSE et PLLCLK). En d'autres termes, vous pouvez choisir entre une clock interne (HSI - 16MHz), une clock externe (HSE - 4 à 26MHz) et une horloge générée par une PLL (PLLCLK). De plus, vous remarquerez que l'horloge des bus et des périphériques dépendent de SYSCLK, hormis l'horloge pour l'USB ainsi que pour l'audio (I2S clock). Nous allons choisir une source d'horloge externe et générer une horloge système SYSCLK à 168MHz. Nous avons vu aussi qu'il existe un registre spécifique de contrôle au sein du microcontrôleur permettant de configurer la source d'horloge principale. C'est le registre RCC (Reset and Clock Control).

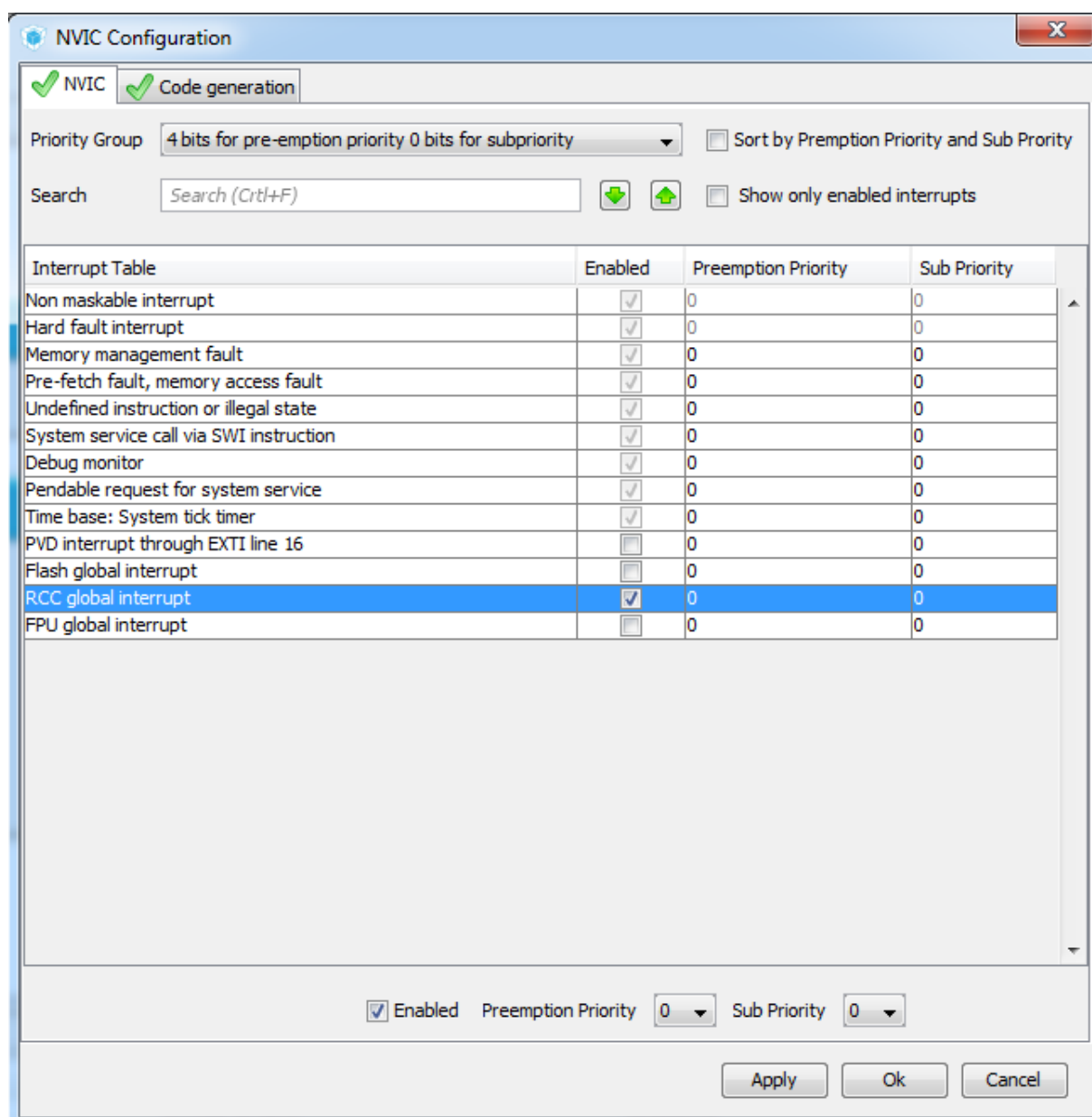
Dans la fenêtre Pinout, indiquer comme source d'horloge HSE (High Speed Clock) que la source d'horloge provient du Crystal/Ceramic Resonator.



6. Le contrôleur de reset et d'horloge (RCC) permet aussi de gérer les différentes sources de reset :

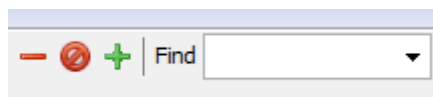
- System reset (remise à zéro des registres via NRST pin)
- Power reset,
- Backup domain reset.

Il est nécessaire d'autoriser les interruptions au niveau du contrôleur d'interruption NVIC comme indiqué sur la figure suivante, accessible via la fenêtre configuration :

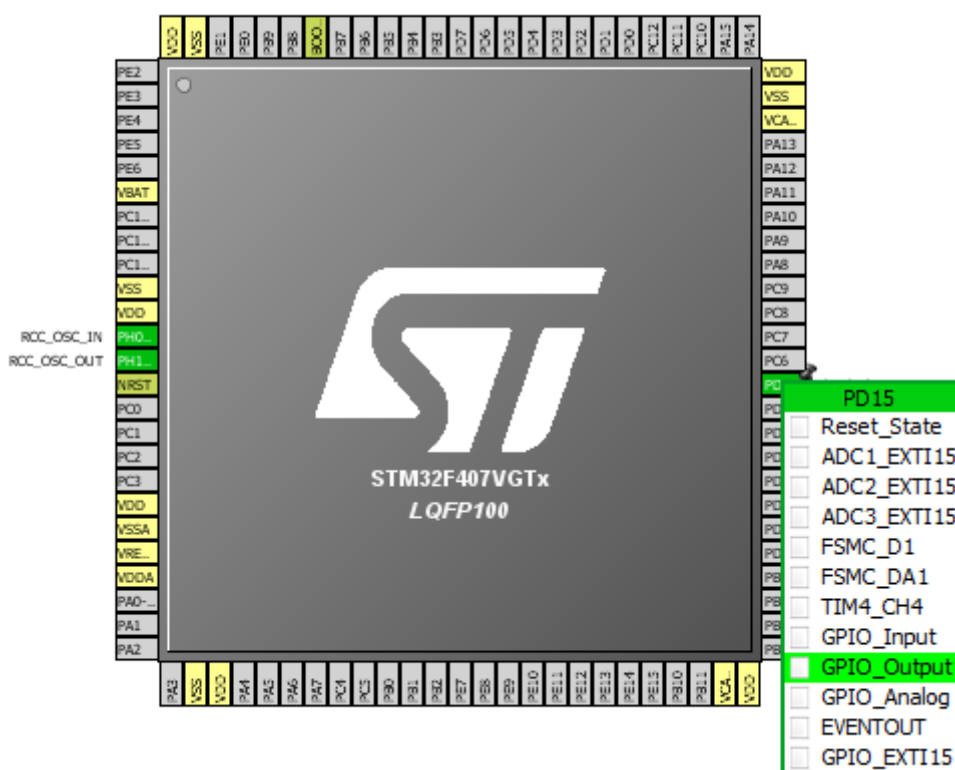


Apply et OK.

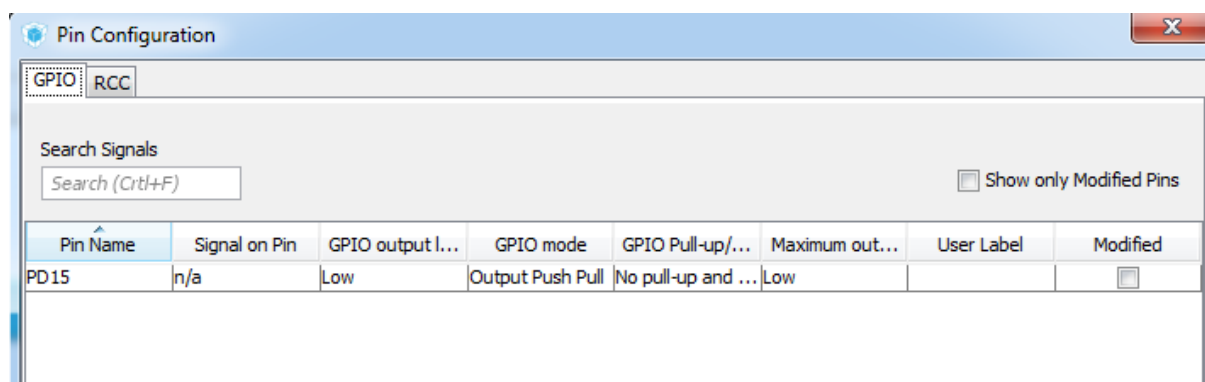
- On souhaite aussi pouvoir contrôler la LED. Celle-ci est connectée au Port D d'E/S (GPIO_D) et à la broche 15 (PD15). Vous pouvez la chercher via le Find :



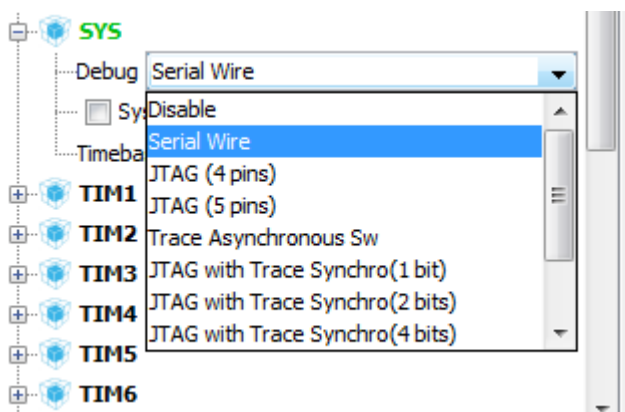
Il faut donc la définir en tant que GPIO_Output comme suit :



Vous noterez que la broche PD15 possède plusieurs fonctionnalités (pouvant servir à d'autres périphériques comme le TIMER, l'ADC, la liaison SPI ...). Une fois configuré, retournez dans la fenêtre configuration, puis GPIO. Vous noterez comment a été configurée la broche (push-pull...)

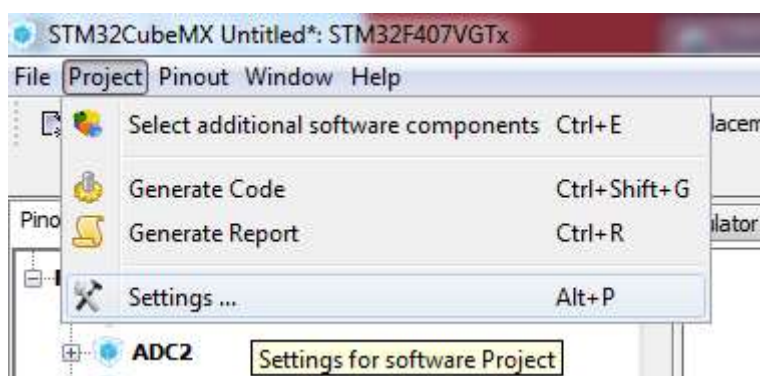


- Dans la fenêtre Pinout, indiquer comme méthode de debug dans le registre SYS la méthode Serial Wire.

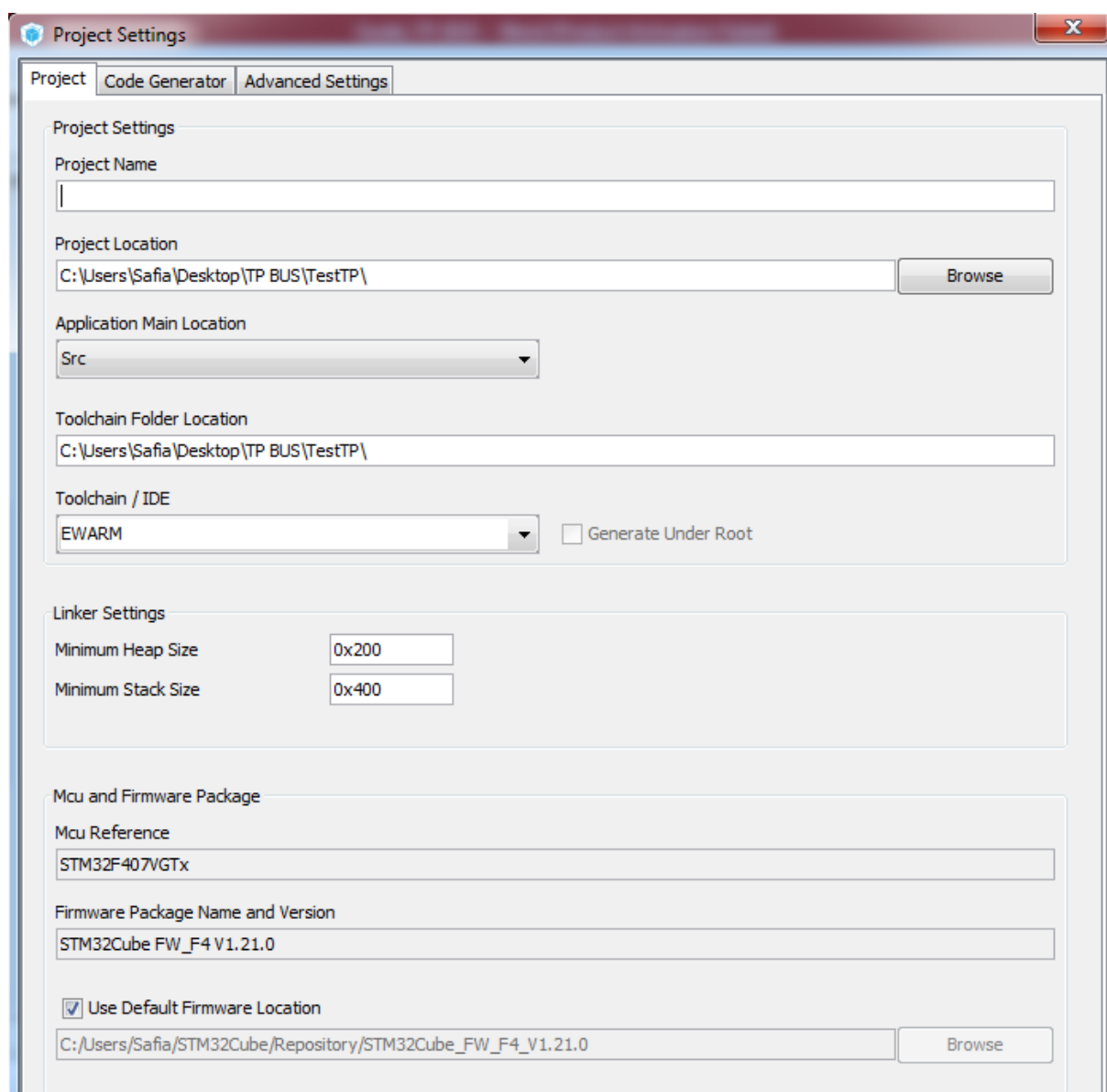


9. Génération du code et du projet pour Keil uVision

C'est la dernière étape de configuration du projet. Aller dans Project/Settings.



- Dans Project Name indiquer le nom de votre projet.
- Dans Project location, indiquer le répertoire de votre projet où sera générer le code.
- Dans Toolchain/IDE, indiquer MDK-ARM v5.
- Dans Linker Settings, vous pouvez observer la taille des zones de Heap et de Stack (Pile).
La zone 'Heap' est une mémoire allouée pour stocker des variables créées dynamiquement au cours de l'exécution. La Pile est une mémoire spécifique permettant la sauvegarde d'information (contenu de registre, adresse de sous-programme, ...).
Typiquement, si l'on définit une taille minimale de 0x200, cela revient à réserver 512 octets de mémoire RAM pour la zone de 'heap'.
Pour la Pile, la zone minimale est deux fois plus grande (environ 1Ko).



Project Settings

Project Name

Project Location

C:\Users\Safia\Desktop\TP BUS\TestTP\

Browse

Application Main Location

Src

Toolchain Folder Location

C:\Users\Safia\Desktop\TP BUS\TestTP\

Toolchain / IDE

EWARM

☐ Generate Under Root

Linker Settings

Minimum Heap Size

0x200

Minimum Stack Size

0x400

Mcu and Firmware Package

Mcu Reference

STM32F407VGTx

Firmware Package Name and Version

STM32Cube FW_F4 V1.21.0

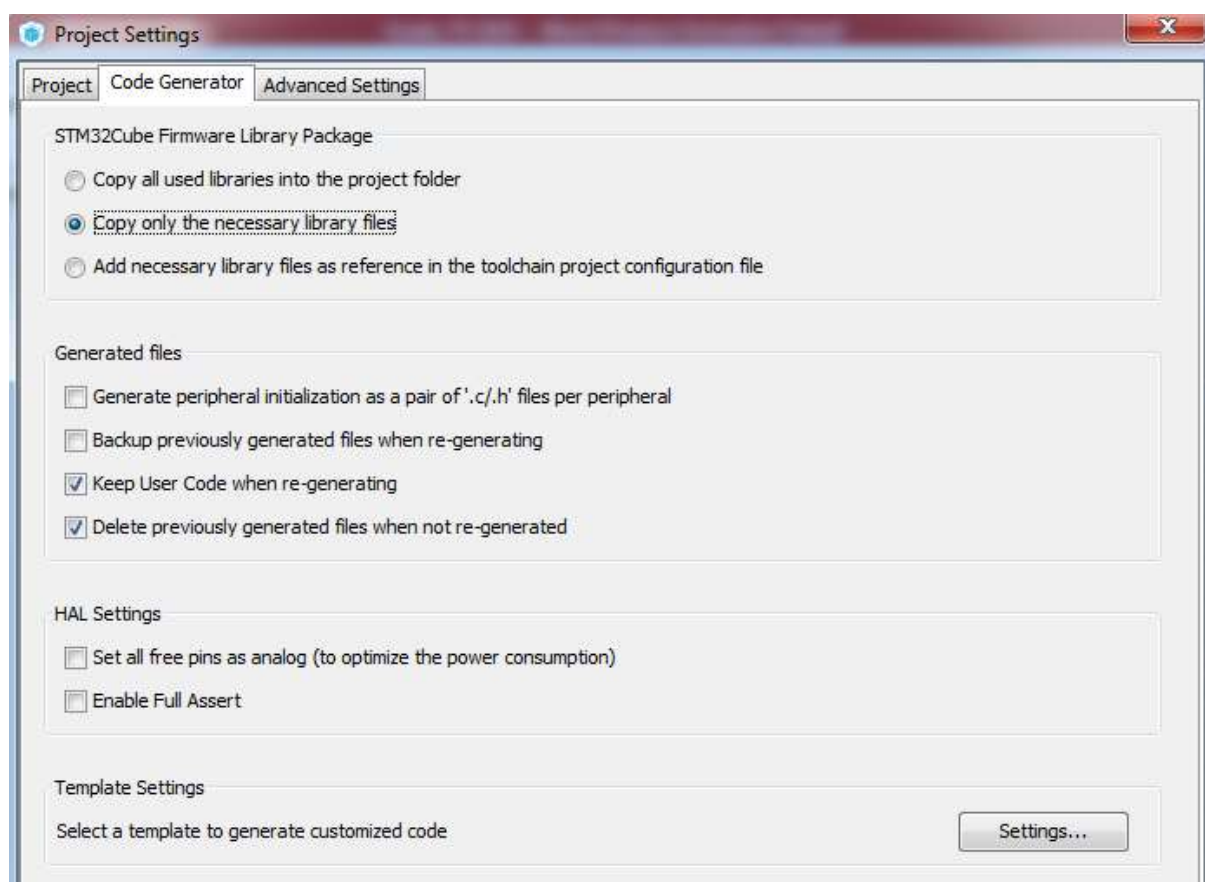
☒ Use Default Firmware Location

C:/Users/Safia/STM32Cube/Repository/STM32Cube_FW_F4_V1.21.0

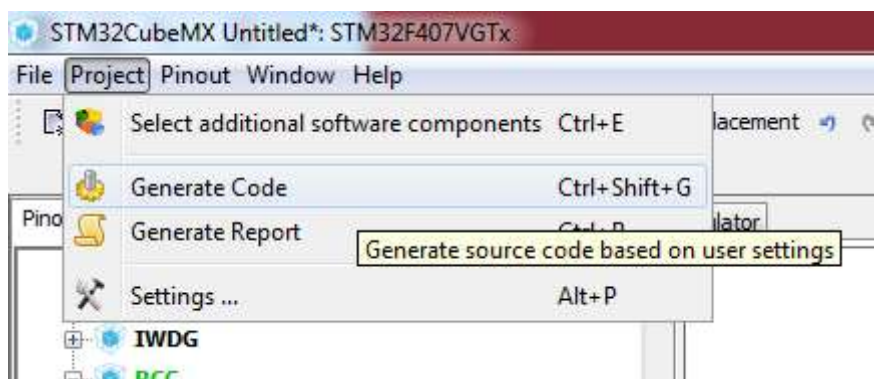
Browse

Dans l'onglet suivant, Code Generator, sélectionner "Copy only the necessary library files".

Puis OK



Enfin, dans l'onglet Project, Generate Code.



Cliquer ensuite sur Open Project pour ouvrir le projet généré sous µVision.

2.2. Analyse des fichiers générés dans µVision de Kiel

Avant de s'intéresser à la programmation du microcontrôleur, on se propose d'étudier les fichiers sources qui ont été générés afin de bien comprendre leurs rôles respectifs.

Dans la fenêtre Project, vous pouvez observer plusieurs répertoires :

- Application/MDK-ARM
- Drivers/CMSIS
- Drivers/STM32F4xx_HAL_Driver

- Application/User

Le répertoire Application/MDK –ARM contient un fichier nommé startup_stm32F401xe.s.

Question : Selon vous à quoi sert ce fichier ? Retrouver les informations suivantes :

- Taille de la Pile ?
- Contenu à l'adresse 0 de la table des vecteurs d'exceptions/interruptions ?
- Quelle action est réalisée lors du reset par le Reset_Handler ?

Dans le répertoire Drivers/CMSIS, regarder le fichier nommé system_stm32f4xx.c. Vous pouvez observer que ce fichier contient plusieurs fonctions réalisant notamment l'initialisation du CPU, la configuration de l'horloge système (SYSCLK).

Vous aurez noté, que la fonction SystemInit est la fonction appelée directement à la suite du reset. Dans le répertoire Drivers/STM32F4xx_HAL_Drivers sont présents les drivers permettant le contrôle de nombreux périphériques du microprocesseur (DMA, Timer, GPIO, ...).

Dans le dernier dossier appelé Application/User, se trouve 3 fichiers.

- main.c
- stm32f4xx_hal_msp.c
- stm32f4xx_it.c

Question : Que contiennent les fichiers stm32f4xx_it.c et stm32f4xx_hal_msp.c ?

Ouvrir le fichier stm32f401xe.h (étendez le fichier main.c)

Question : Que contient ce fichier ? Retrouver le mapping mémoire.

A quelles adresses commence et termine la FLASH ? la SRAM ? la mémoire allouée pour les périphériques ?

A quelle adresse commence la zone mémoire pour l'ADC ?

2.3. Ajout de fonctionnalités

Comme vous pouvez le remarquer, le code n'effectue aucun traitement spécial après les étapes d'initialisation et de configuration. Nous allons donc rajouter quelques traitements.

1. Ecriture de la fonction

Modifier votre code pour mettre à 1 la led pour une période de 250 ms.

Utiliser la ligne suivante pour allumer la led :

```
HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_15);
```

Pour le delay :

```
HAL_Delay(250);
```

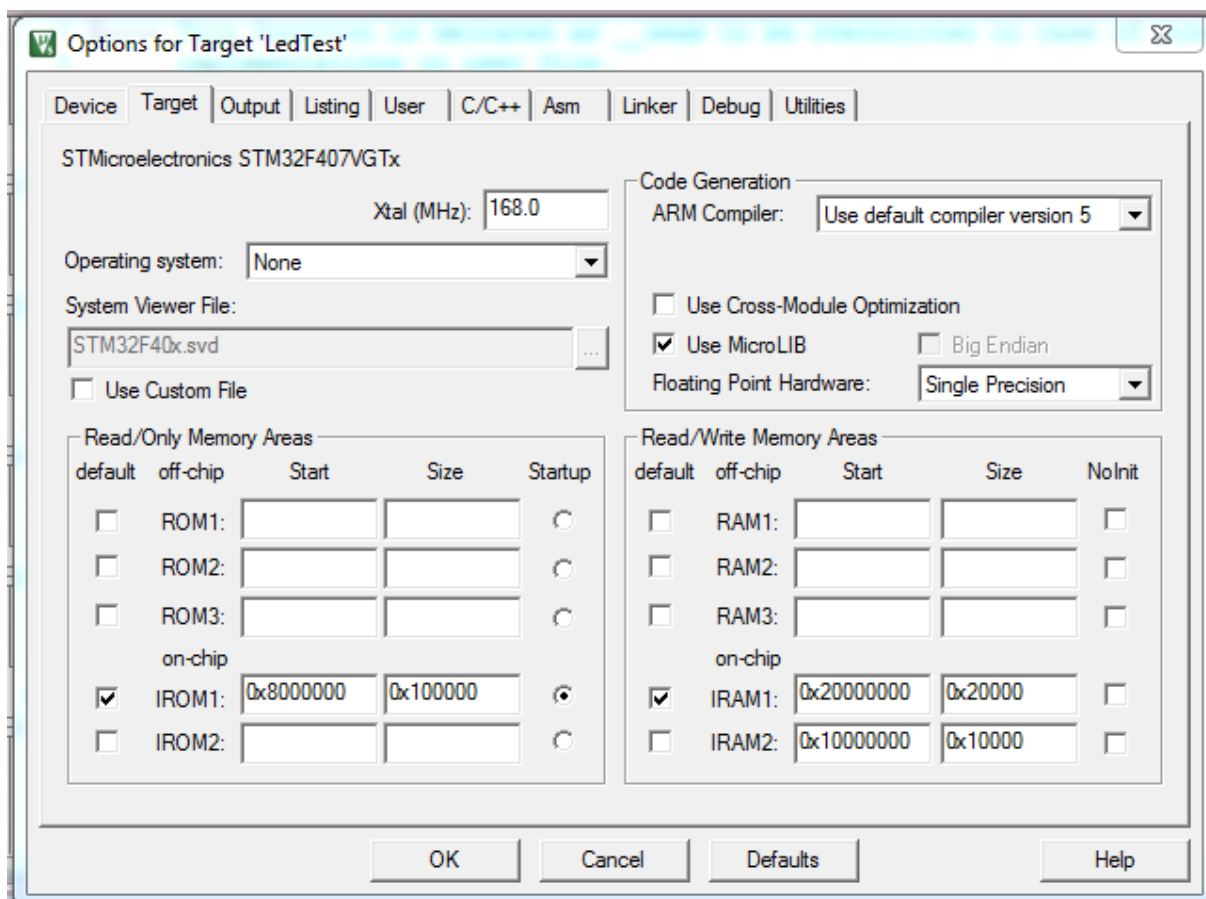
Une fois le code complété, compiler via la touche F7 ou l'icône  :

S'il y a une erreur, consulter les fenêtres Build Output et Error list pour savoir d'où elle provient et la corriger.

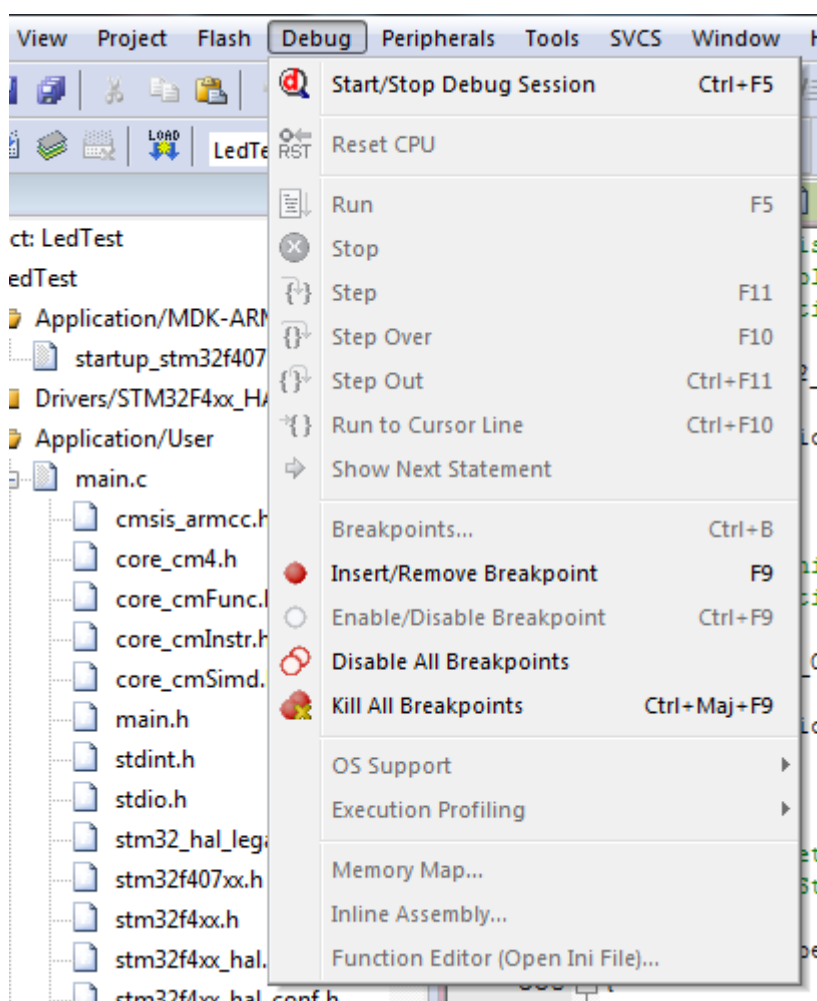
Recompiler jusqu'à ce qu'il n'y ait plus d'erreur. Si c'est le cas, le fichier exécutable a été créé. On peut passer à l'étape de debug.

2. Debug

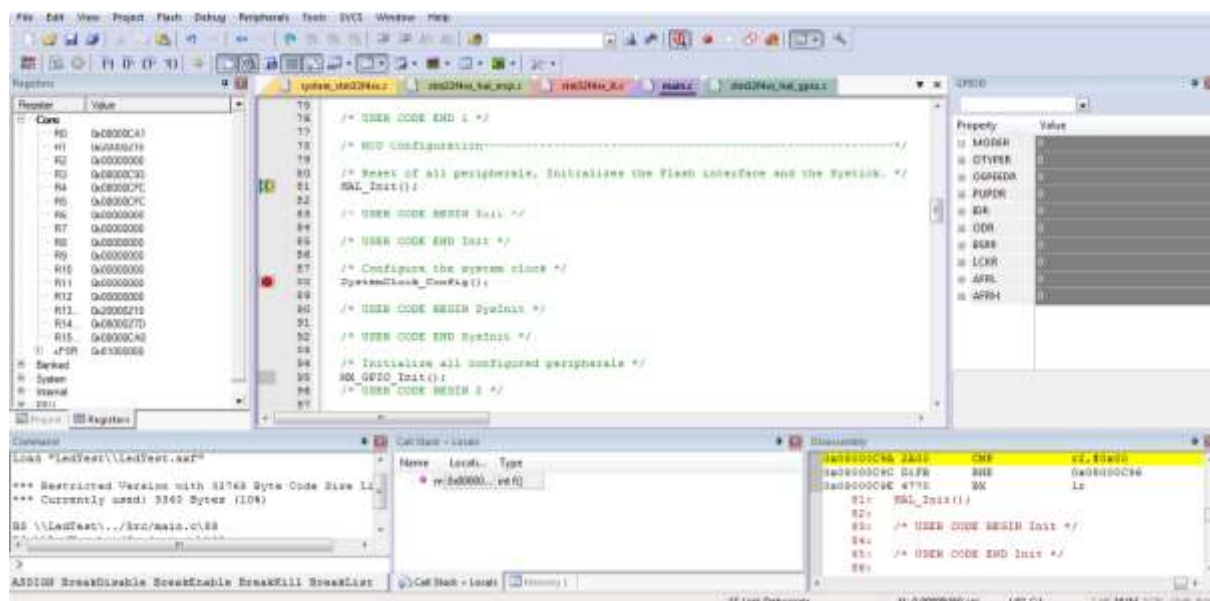
Connecter la plateforme via le lien USB. Il est possible que le driver de debug ne soit pas spécifié. Vérifier dans l'onglet Flash/Configure Flash Tools, puis dans Debug, vérifier que STLink Debugger est bien choisi comme indiqué par la figure ci-dessous, puis OK :



On va maintenant s'intéresser au debugger. Lancer le via l'onglet Debug/Start-Stop Debug session.



Plusieurs fenêtres sont alors visibles comme le montre la figure suivante :



- Registers -> affiche le contenu courant des registres du microcontrôleur.
- Disassembly -> affiche le code assembleur généré à partir de votre code C.

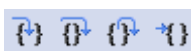
- Command -> permet à l'utilisateur de rentrer des commandes lors du debug
- Call Stack + locals -> permet d'observer l'adresse de la fonction courante exécutée ainsi que les variables locales utilisées (adresses et contenus)

Mettre un point d'arrêt sur la ligne d'appel à la fonction `HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_15)` dans le fichier `main.c` (en cliquant sur la zone grise à gauche du numéro de ligne).

On lance ensuite l'exécution du code jusqu'à cette ligne en cliquant sur F5 ou via l'icône suivante



On dispose aussi de plusieurs commandes pour effectuer du pas-à-pas :



1) 2) 3) 4)

1-Step -> permet de rentrer dans la prochaine étape d'exécution de la commande C

2-Step over -> permet de continuer l'exécution jusqu'à la prochaine ligne

3-Step out -> permet de sortie de la fonction courante

4-Run cursor to line -> permet de lancer l'exécution du code jusqu'à la ligne indiquée

3. Bilan et conclusion

Lors de ce TP, vous avez pu vous analyser les différents fichiers nécessaires à l'initialisation et la configuration du microcontrôleur.

Le debugger est un élément indispensable lors de tout développement de code pour un processeur ou un microcontrôleur. μ Vision dispose d'un environnement très riche dont vous avez vu quelques possibilités. Ainsi, pouvoir accéder directement aux registres, visualiser le contenu des mémoires, (etc.) sont autant d'éléments indispensables pour bien comprendre, tester et valider une implémentation.