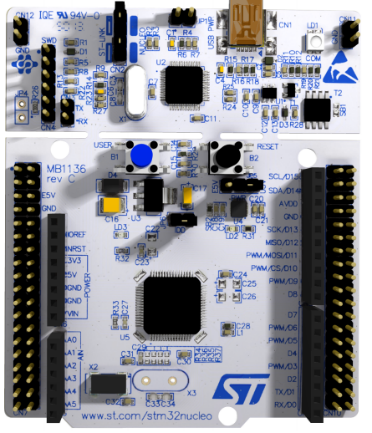
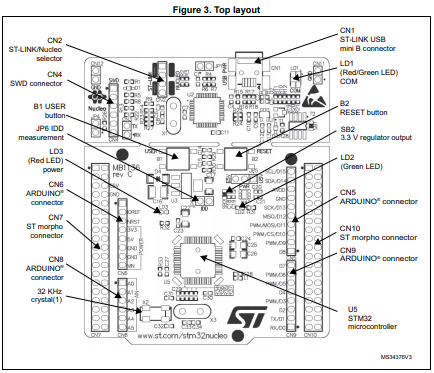
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

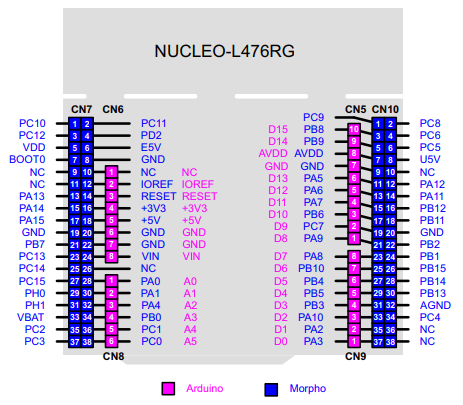
PREMIER PAS VERS LE DEVELOPPEMENT AUTOUR DU STM32

Guetting\_Started

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Présentation de la carte Nucleo stm32L476rg**





**LEDS**

La LED LD1 tricolore (vert, orange, rouge) LD1 (COM) renseigne sur ST-LINK état des communications. La couleur par défaut de LD1 est le rouge. LD1 devient vert pour indiquer que communication est en cours entre le PC et le ST-LINK/V2 :

• Clignotement lent rouge/éteint : à la mise sous tension avant l'initialisation USB

• Clignotement rapide rouge/éteint : après la première communication correcte entre le PC et ST-LINK/V2-1 (énumération)

• LED rouge allumée : lorsque l'initialisation entre le PC et ST-LINK/V2-1 est terminée

• LED verte allumée : après une initialisation réussie de la communication cible

• Clignotant Rouge/Vert : pendant la communication avec la cible

• Vert allumé : communication terminée et réussie

• Orange allumé : échec de la communication

La LED Utilisateur LD2 : la LED verte est une LED utilisateur connectée au signal ARDUINO® D13 correspondant vers STM32 I/O PA5 (broche 21) ou PB13 (broche 34) selon la cible STM32. Faire référence à Tableau 11 à Tableau 23 lorsque :

• l'E/S est à la valeur HAUTE, la LED est allumée

• l'I/O est LOW, la LED est éteinte

LD3 PWR : la LED rouge indique que la partie STM32 est alimentée et que l'alimentation +5V est disponible

**PUSH-BUTTON**

B1 USER : le bouton utilisateur est connecté à l'I/O PC13 (broche 2) du STM32 microcontrôleur.

B2 RESET : ce bouton poussoir est connecté à NRST, et est utilisé pour le RESET le STM32 microcontrôleur.

Avant de commencer à développer il est primordial d'aller chercher nous même les informations.

Pour se repérer dans le STM32, il nous faut 2 documentations essentielles :

* Le datasheet (organisatin interne du micro, spécifications électrique, etc)

<https://www.st.com/resource/en/datasheet/stm32l476rg.pdf>

* Le reference manual (fonctionnement détaillé de chaque périphérique et registres associés)

<https://www.st.com/resource/en/reference_manual/dm00083560-stm32l4x5-and-stm32l4x6-advanced-arm-based-32-bit-mcus-stmicroelectronics.pdf>

* Bien sur si l’on développe sur carte nucleo pour nos tests il faut la petite documentation

<https://www.st.com/en/evaluation-tools/nucleo-l476rg.html#documentation>

La connexion : Nous avons plusieurs choix possibles :

Pour une première connexion il nous faut le driver STLink/V2: stsw-link009\_v2.0.2.

<https://www.st.com/en/development-tools/stsw-link009.html>

La programmation : Nous avons plusieurs choix possibles :

Nous allons choisir Atollic de TrueStudio car il permet de prendre en compte la compilation arm et la possibilité de programmer en C/C++ et d'accéder au registre et d'intégrer une architecture personnalisée.

Nous allons aussi utiliser CubeMX pour comprendre en cas de difficulté de fonctionnement puisqu'il utilise les dernières bibliothèques et permet de débuguer certaines fois, de plus cela permet de compenser les faibles tutos sur Atollic.

Mais pour débuter facilement sans prise de tête nous allons utiliser Keil : MDK-ARM

[https://www.keil.com/demo/eval/arm.htm#!#DOWNLOAD](https://www.keil.com/demo/eval/arm.htm#!)

Mais pour débuter et gagner en compréhension et se familiariser avec les bibliothèques :

STM32CUBEMX et CUBEIDE

<https://www.st.com/en/development-tools/stm32cubemx.html>

<https://www.st.com/en/development-tools/stm32cubeide.html>

**Premier Test pour la mise en place de l’environnement de travail**

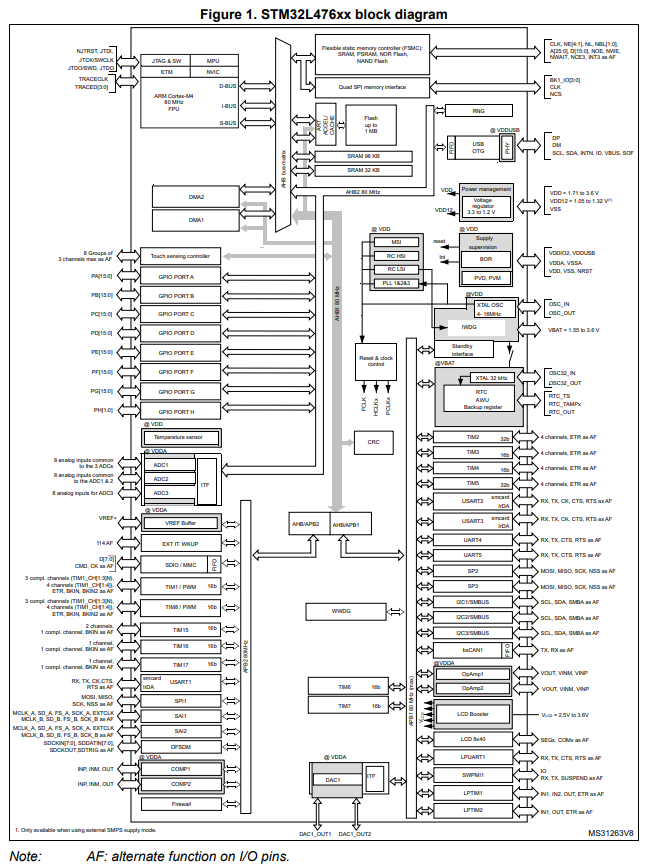
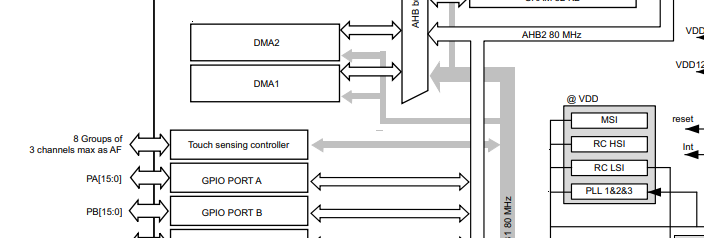
Faire clignoter la Led (LD2) reliée sur le port PA5 de la carte Nucleo.

Il nous faut la documentation

* datasheet [stm32l476rg.pdf]
* reference\_manual (dm00083560-stm32l4x5-and-stm32l4x6-advanced-arm-based-32-bit-mcus-stmicroelectronics.pdf) [RM0351]
* Manual-Board carte nucleo-34 board [UM1724]

*Sur les microcontrôleurs 8 et 16 bits (MSP430, ATMEL, PIC, etc) on configure les périphériques et c’est parti, Mais sur ARM il faut : Activer (AHB2ENR) et régler au besoin l’horloge du périphérique + Activer le registre GPIOA (MODER)+ Activer le port pin (1 ou 0) (ODR).*

1. il faut savoir sur quel bus d’horloge est connecté le dit périphérique GPIO\_PORT\_A du port PA5

[datasheet]

Le PortA PA[15:0] est sur le bus AHB2 80MHZ.

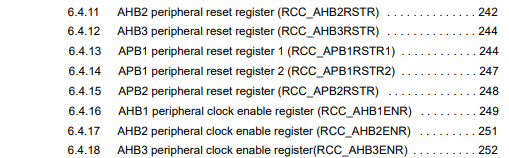
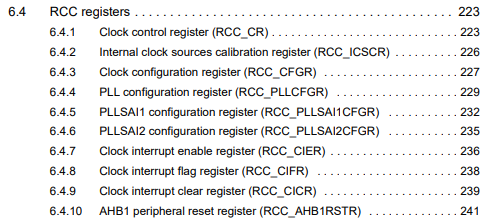
Maintenant il faut aller voir dans [reference\_manual] ce que raconte le bus AHB2 ?

C’est un registre classé dans la fonctionnalité RCC (Reset and Clock Control).

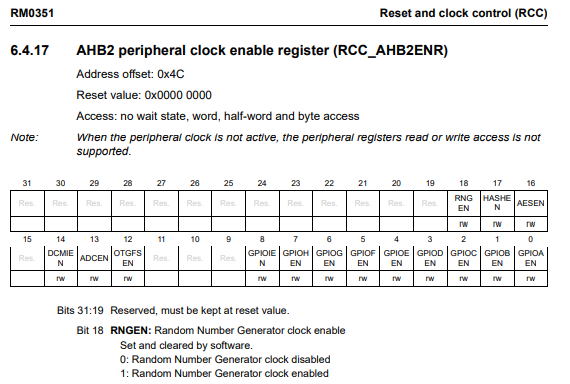
D’où le pointeur RCC->AHB2xxx et ce registre AHB2 doit être enable (AHB2ENR) ou reset (AHB2RSTR).

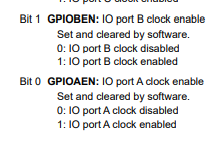
Il est toujours intéressant d’aller lire ce que fait la fonctionnalité RCC.

[reference\_manual]

1.  La bus AHB2 est rattaché au registre : manual\_register\_stm32L476xx.pdf

AHB2 peripheral clock enable register (RCC\_AHB2ENR) p251

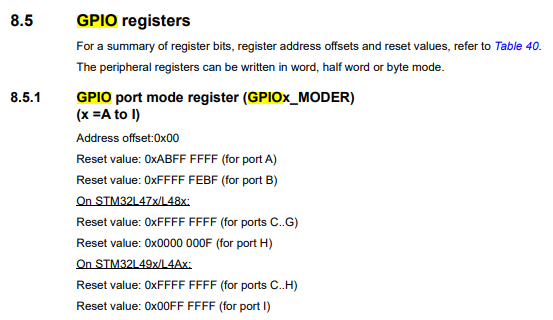


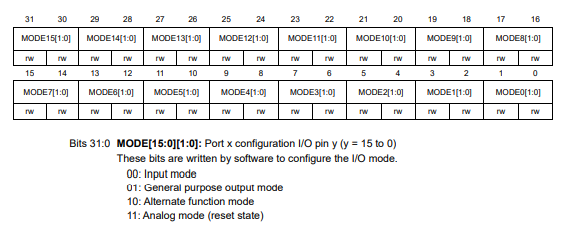


On voit qu’il faut mettre le bit 0 du registre à 1.

Pour activer l’horloge du port A (on a l’utilité de chaque bit p252 GPIOAEN).

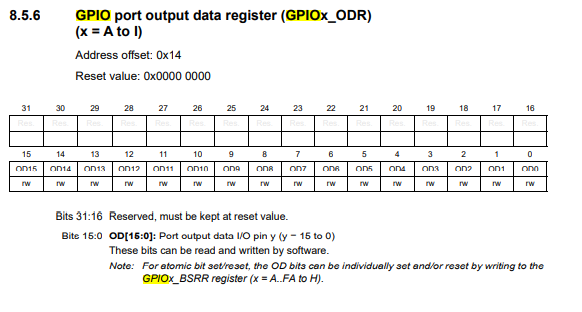
1. Concernant le registre GPIOA (MODER) il faut l’activer





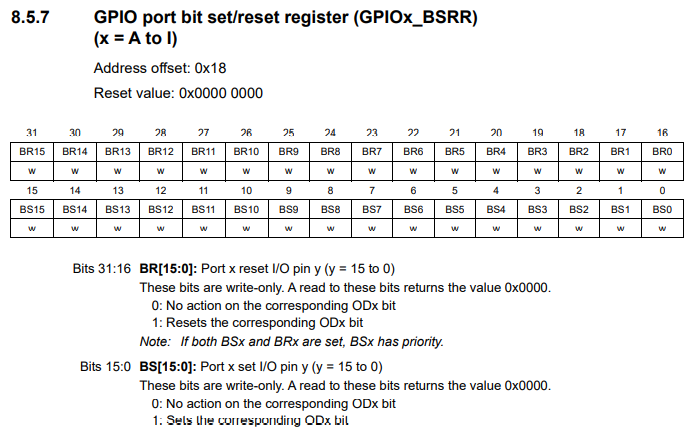
Concernant le registre 32 bits MODER de la fonctionnalité GPIOA il s’agit de mettre le bit 10 à 1.

1. Activer la sortie PA5 (*Activer le port pin (1 ou 0) (ODR).*



Et enfin on met à 1 ou 0 le bit de pA5 via le registre ODR de la fonctionnalité GPIOA

Si tu te demandes comment connaitre les registres nécessaires à telle ou telle fonctionnalités, c'est écrit dans le Reference Manual, regarde le chapitre **8 General-purpose I/Os (GPIO)** tout est écrit.



Donc le programme est

#include "stm32l4xx.h"

int main(void)

{

volatile unsigned int i =0;

RCC -> AHB2ENR|= 1<<0;

//GPIOA->MODER |= 1 << 10;//erreur

GPIOA->MODER &= 0xFFFFF7FF;

while(1)

{for (i= 0; i < 50000; i++)

GPIOA -> ODR= (1<<5); //on affecte 1 au bit du rang 5

//GPIOA -> BSRR= (1<<5);

for (i = 0; i < 50000; i++)

GPIOA -> ODR= (0<<5); //PA5 à 0 ////GPIOA -> BSRR= (1<<21);}

}

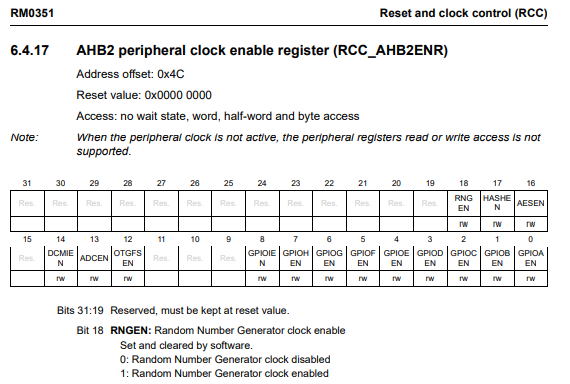
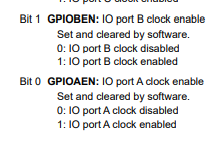
}

Nous avons compris que :

Il faut Activer l’horloge et si besoin la régler.

Le PortA PA[15 :0] est sur le bus AHB2 80MHZ. [datasheet] organisation interne du stm32.

AHB2 est un registre classé dans la fonctionnalité RCC (Reset and Clock Control), d’où le pointeur RCC->AHB2xxx et ce registre AHB2 doit être enable (AHB2ENR)



RCC -> AHB2ENR|= 1<<0;

…

On met à 1 le bit 0 du registre AHB2ENR

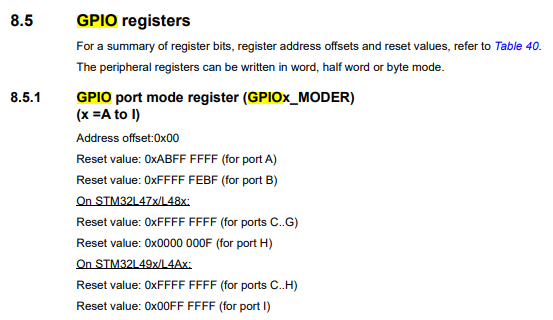
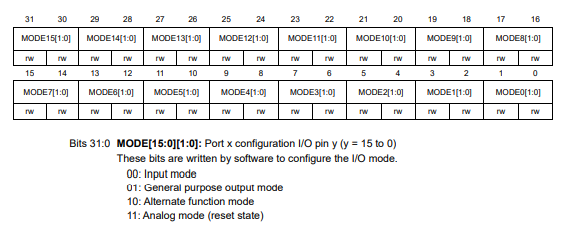
0000 0000 0000 0000 0000 0000 0000 0000

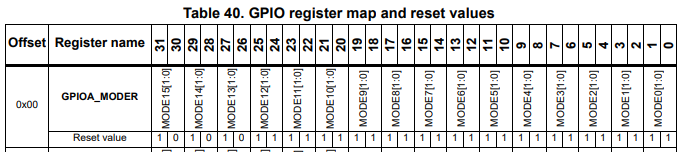
OU 0000 0000 0000 0000 0000 0000 0000 0001

--------------------------------------------------------------

= 0000 0000 0000 0000 0000 0000 0000 0001

Il faut activer le port GPIOA avec MODER





Concernant le registre 32 bits MODER de la fonctionnalité GPIOA 5 il s’agit de mettre MODE5[01].

GPIOA->MODER &= 0xFFFFF7FF; //GPIOA->MODER |= 1 << 10;//erreur

Reset Value : 0XABFF FFFF

1010 1011 1111 1111 1111 1111 1111 1111 : 0XABFF FFFF

OU 0000 0000 0000 0000 0000 **01**00 0000 0000

-----------------------------------------------------------------------------

1010 1011 1111 1111 1111 1111 1111 1111 = GPIOA->MODER |= 1 << 10 **ERREUR BIT 10 à 1**

Ici tout est à 1 en entrée, nous on veut MODE5[0 :1] ouput mode5 [01 : General purpose output mode]

Soit le bit 10 à 1 et le bit 11 à 0 pour que PA5 soit une sortie

1010 1011 1111 1111 1111 1111 1111 1111 : 0XABFF FFFF

& 1111 1111 1111 1111 1111 **01**11 1111 1111

-----------------------------------------------------------------------------

1010 1011 1111 1111 1111 **01**11 1111 1111 = GPIOA->MODER &= 0xFFFFF7FF

GPIOA->MODER &= 0<<11

Voici pourquoi il y a l’erreur :

Pour une question d’écriture et de préférence :

GPIOA->ODR |= (1 << 5); /\* mise à 1 du bit n°5 \*/ moi je préfère

GPIOA->ODR |= 0x20; /\* mise à 1 du bit n°5 \*/ ce qui revient exactement au même.  
 OU 0000 0000

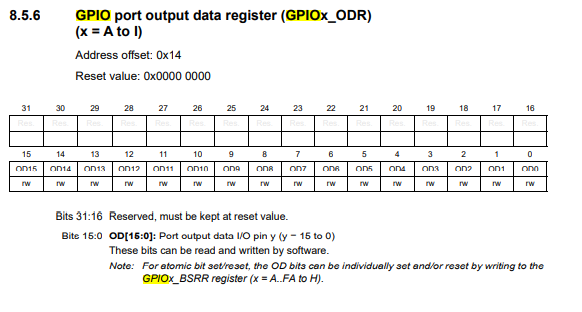
0x20 = 0010 0000 = 0010 0000

GPIOA->ODR &= ~(1 << 5); /\* mise à 0 du bit n°5 \*/ moi je préfère

GPIOA->ODR &= 0xDF; /\* mise à 0 du bit n°5 \*/ ce qui revient exactement au même.  
 ET 1111 1111

0xDF = 1101 1111 = 1101 1111

Il faut actionner la sortie

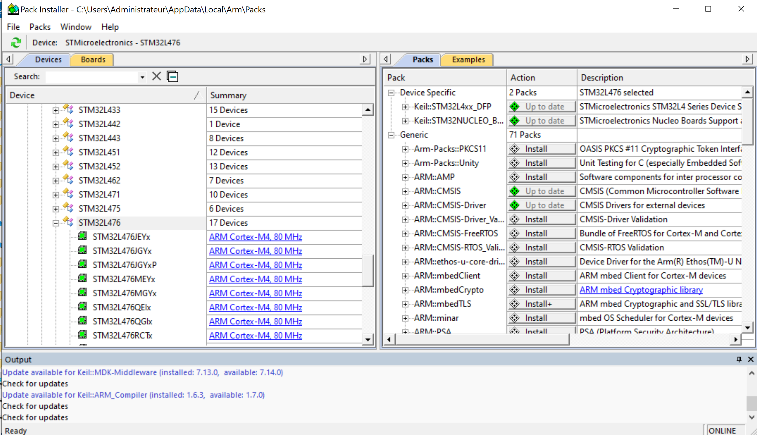


Et enfin on met à 1 ou 0 le bit de pA5 via le registre ODR de la fonctionnalité GPIOA

GPIOA -> ODR= (0<<5) GPIOA -> ODR= (1<<5)

Pour directement rentrer dans la programmation et tester, nous allons utiliser keil de MDK-ARM

Téléchargement du logiciel : [https://www.keil.com/demo/eval/arm.htm#!#DOWNLOAD](https://www.keil.com/demo/eval/arm.htm#!)

choisir STMicroelectronic/STM32L476

pack :

STMicroelectronic STM32L4

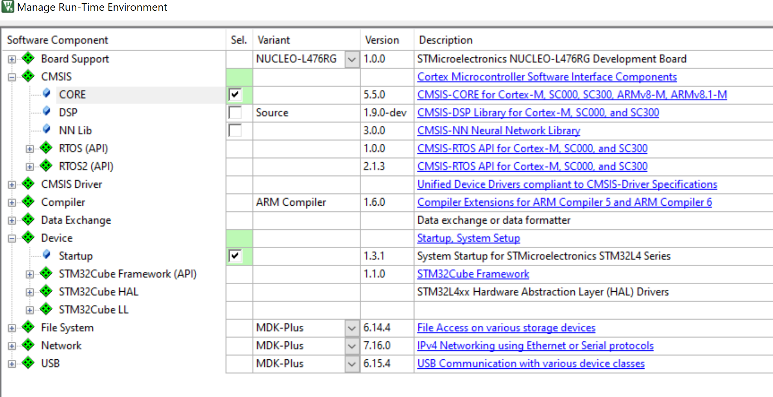
STMicroelectronic Nucleo

CMSIS (Cortex Micro Soft Interface Standard)

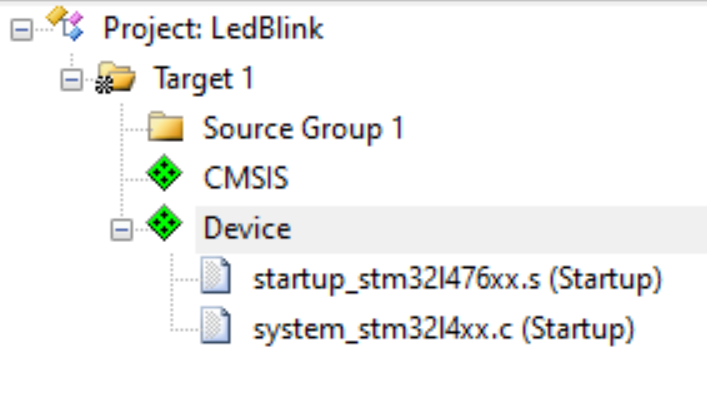
Middleware for keil MDK-Pro et MDK-Plus

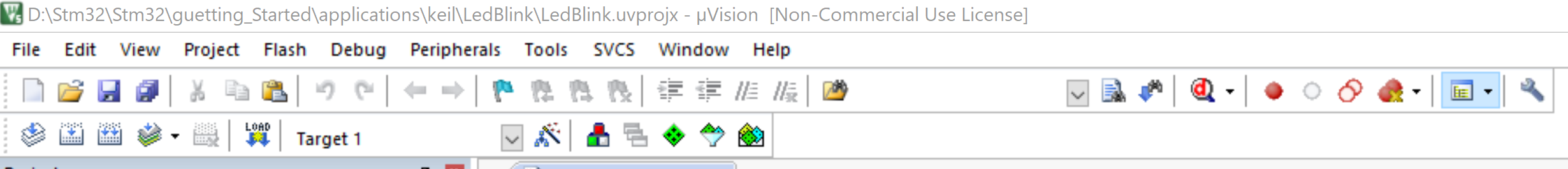
Fermer

Ouvrir Keil>Projet>New Projet>choisir sont stm32L476RG et son dossier de travail.

choisir, nous avons besoin du fichier startup et il nous demande de rajouter aussi CMSIS/CORE

Cela crée l’arborescence suivante





Ajoute des packages

Ajouter un nouveau stm32

Arborescence

Configuration de la cible et choix driver STLink

Charger le programme dans la cible

Créer un fichier et copier le code enregistrer, mais il faut ajouter se fichier dans notre arborescence on double-click sur le répertoire et on ajoute notre fichier.

Nous pouvons tester notre LED clignotante (Super)

C’est bien Keil, mais travailler avec les registres c’est un peu fou fou.

En plus on a besoin parfois de rentrer dans les détails.

Que faut-il appeler comme registre pour que ça fonctionne et oui on n’est pas des dieux du

Microcontroller

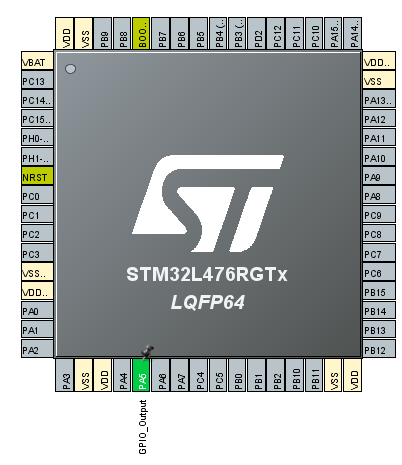
Alors on installe stm32CubeMX avec ses nombreux tutos du net on voit dans les détailles des registres en mode debug mais aussi cela nous approche d’un peu plus des bibliothèques pour prendre de la hauteur.

Pour se familiariser avec le stm32 avec une interface : stm32CUBEMX

Téléchargement : <https://www.st.com/en/development-tools/stm32cubemx.html>

<https://www.st.com/en/development-tools/stm32cubeide.html>

Il manque plus que choisir sa carte et c’est partie, nous pouvons générer pour plusieurs plateformes.



Application Led

Mode debug STLINK

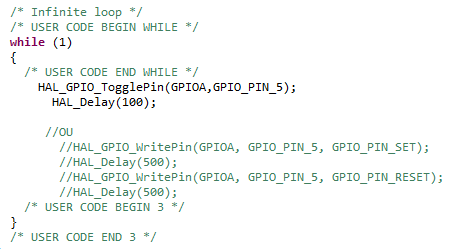
HAL\_GPIO\_WritePin(GPIOA,GPIO\_PIN\_3, *GPIO\_PIN\_SET*);

HAL\_Delay(1000);

HAL\_GPIO\_WritePin(GPIOA,GPIO\_PIN\_3, *GPIO\_PIN\_RESET*);

HAL\_Delay(1000);

Dans le main



GPIO NUCLEO PIN

