Guide d'utilisation du microcontrôleur STM32

Table des matières

INTRODUCTION

Ce tutoriel est basé sur la carte NUCLEO 32F401RE et le logiciel TrueStudio d'Atollic. Avec ce tutoriel vous pourrez prendre en main les microcontrôleurs de la famille STM. Le dernier STM32 donne une grande flexibilité pour créer des systèmes performance incluant les appareils temps réels sans compromettre les fonctionnalités.

1) FONCTIONALITES

Description des noms de codes pour NUCLEO-TXXXRY
TXX : gamme de produits STM32
R: nombre de broches du paquet STM32

- ➔ R pour 64

Y: taille de la mémoire FLASH

- ➔ 8 pour 64 Ko, B pour 128 Ko, C pour 256 Ko, E pour 512 Ko, G pour 1 Mo et Z pour 192 Ko

Interface SWD : Serial Wire Debug est une interface électrique à 2 broches, c'est une alternative au JTAG qui utilise le même protocole. SWD utilise un protocole bidirectionnel filaire standard de CPU ARM, défini dans l'interface de débogage ARM v5. Cela permet au débuggeur de devenir un autre maître de bus AMBA pour accéder à la mémoire système et aux registres de périphériques.

JTAG : Join Test Action Group est une norme industrielle pour vérifier les designs et tester les circuits imprimés, intitulée "Standart Test Access Port and Boundary-Scan Architecture". Le terme est abusivement et largement utilisé au lieu de Boundary Scan.

Boundary Scan: la scrutation de frontières est une technique conçue pour faciliter et automatiser le test des cartes électroniques numériques.

CPU

ARM

AMBA

2) LE PROGRAMMER ST-LINKER2-1

La carte dispose du programmeur/débugger ST-LINKV2-1 qui permet de programmer les microcontrôleurs STM 8 bits et 32 bits. Nous nous concentrerons sur ceux de 32 bits. Il est paramétré par le connecteur 2 (CN2) Soit :

- par défaut, s'il y a deux jumpers, le programmeur est connecté au STM32F401RE de la carte
- si les deux jumpers sont enlevés, le programmeur s'utilise avec le connecteur 4 (CN4) par interface SWD. Vous trouverez ci-dessous les pins et les descriptions respectives du CN4

Pins	**Descriptions**
pin 1 (pastille carrée)	VDD_TARGET, VDD du microcontrôleur cible
pin 2	SWCLK, horloge du SWD
pin 3	GND, la masse
pin 4	SWDIO, entrée/sortie de données SWD
pin 5	NRST, reset de la cible STM32
pin 6	SWO, réservé

A noter que si le pin 5 (NRST) du connecteur 4 (CN4) est utilisé, le pont de soudure (SB12) de NRST doit être à OFF.

3) Connecteurs STM32F401

Vous trouverez dans le tableau suivant le rôle de chaque pin du connecteur STM32F401 ainsi que leurs descriptions respectives

Pins	Rôles	Descriptions
JP1	permet de contrôler le courant max provenant du port USB (CN1).	- Si le jumper est placé, le courant max est de 300 mA - S'il n'y a pas de jumper, le courant max est de 100 mA
JP5	permet de définir la source d'alimentation de la carte.	- Si le jumper est du côté de U5V (ST-LINK VBUS), l'alimentation provient du port USB - Si le jumper est du côté de E5V, l'alimentation est une source externe venant de Vin ou de E5V
VIN , E5V	permettent d'alimenter la carte par une source externe	- Le pin 6 du connecteur 7 (CN7) doit recevoir 5 V +-0,25V avec 500 mA max lorsque E5V est choisi - Le pin 8 du connecteur 6 (CN6) et le pin 24 du connecteur 7 (CN7) peuvent recevoir de 7 à 12 V. Pour 7 V, le courant est limité à 800 mA, pour 9 V à 450 mA et pour 12 V à 250 mA
LD3 (Led PWR)	indique si le STM32 est alimenté est que le pin +5V est fonctionnel.	Elle est connectée à PA5 (D13 sur la carte)
LD2	C'est une led utilisateur	elle est connectée à PA5 (D13 sur la carte)
B1	Bouton utilisateur	Il est connecté à PC13
B2	Il permet de réinitialiser le microcontrôleur	Il est connecté à NRST

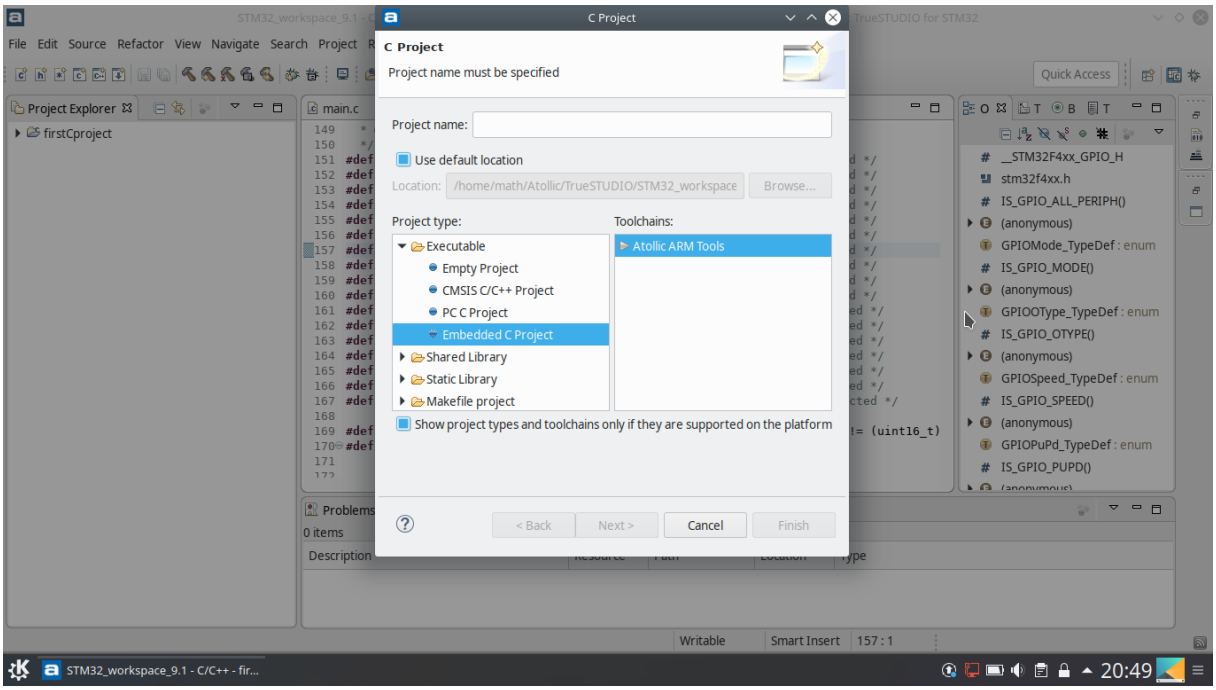
.

4) PROGRAMMATION GPIO

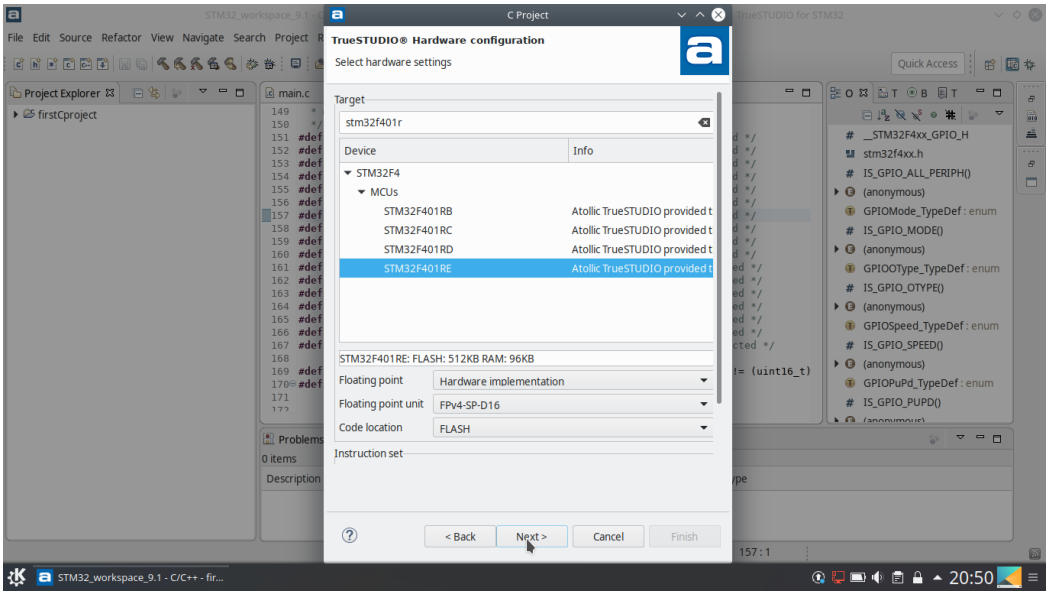
GPIO veut dire : General Purpose Input/Output, ou Entrées/Sorties pour un usage générique.
Ce paragraphe explique comment utiliser d'un pin.

Lancer le logiciel TrueSTUDIO

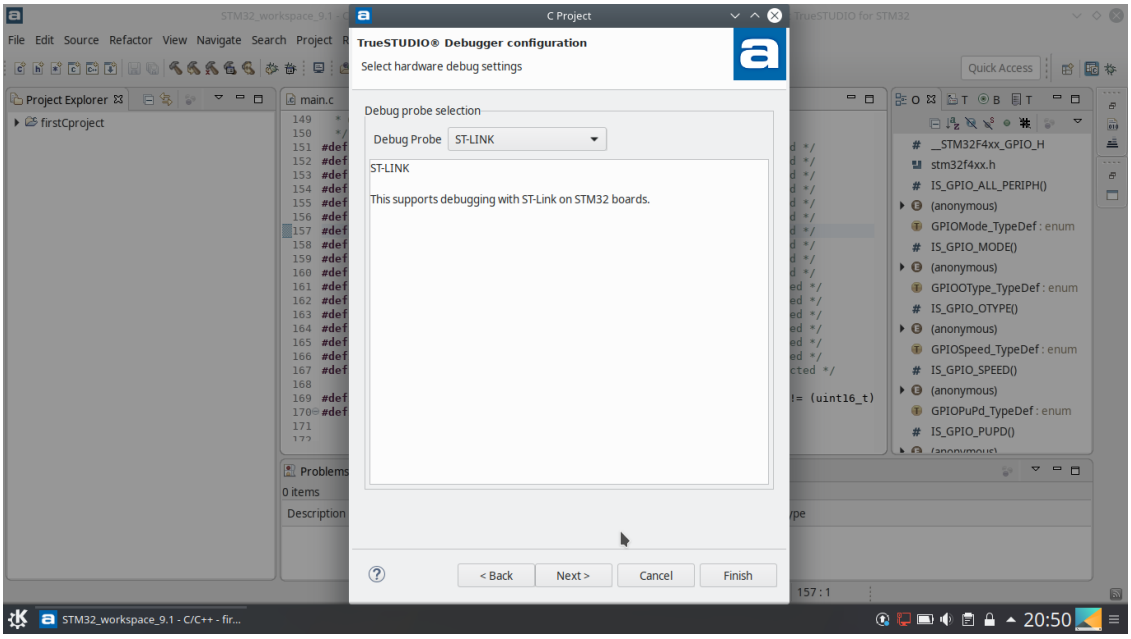
- la première étape est de créer un projet (file⍰new⍰ cproject).
  NB: Ce tutoriel se base sur un projet en C, mais il est possible d'utiliser un projet en C++. Cependant, un projet en c++ a moins d'aide qu'un projet en C pour définir les registres.
- Créer un C project en lui donnant un nom, et en choisissant embedded C project dans executable avec la toolchains Atollic ARM tools.



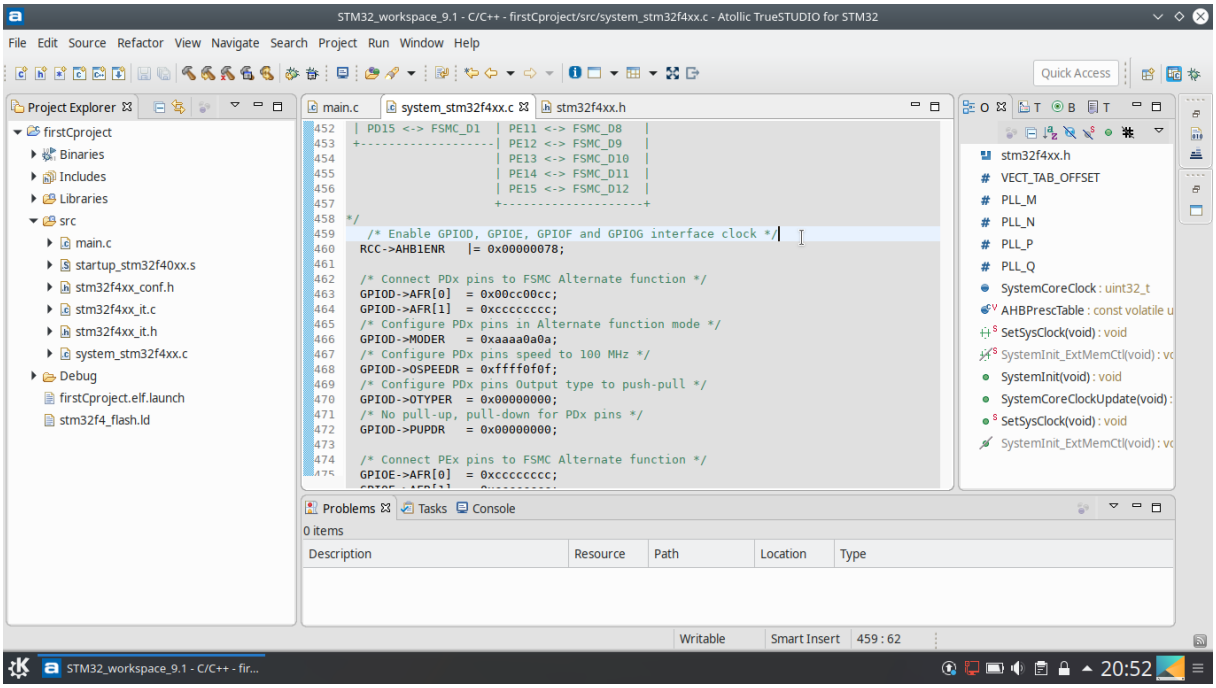
- Ensuite, il faut choisir sa cible



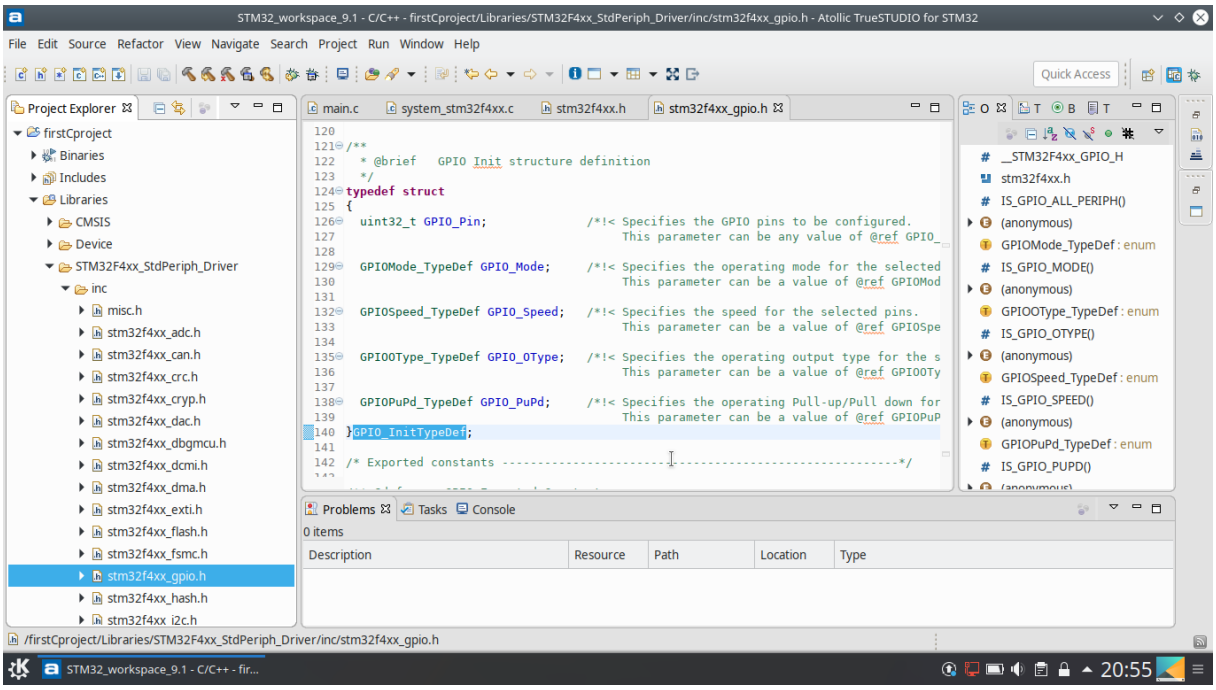
- puis utiliser le debuggeur ST-LINK (il n'est donc pas utile d'installer les autres débuggeur lors de l'installation du logiciel).

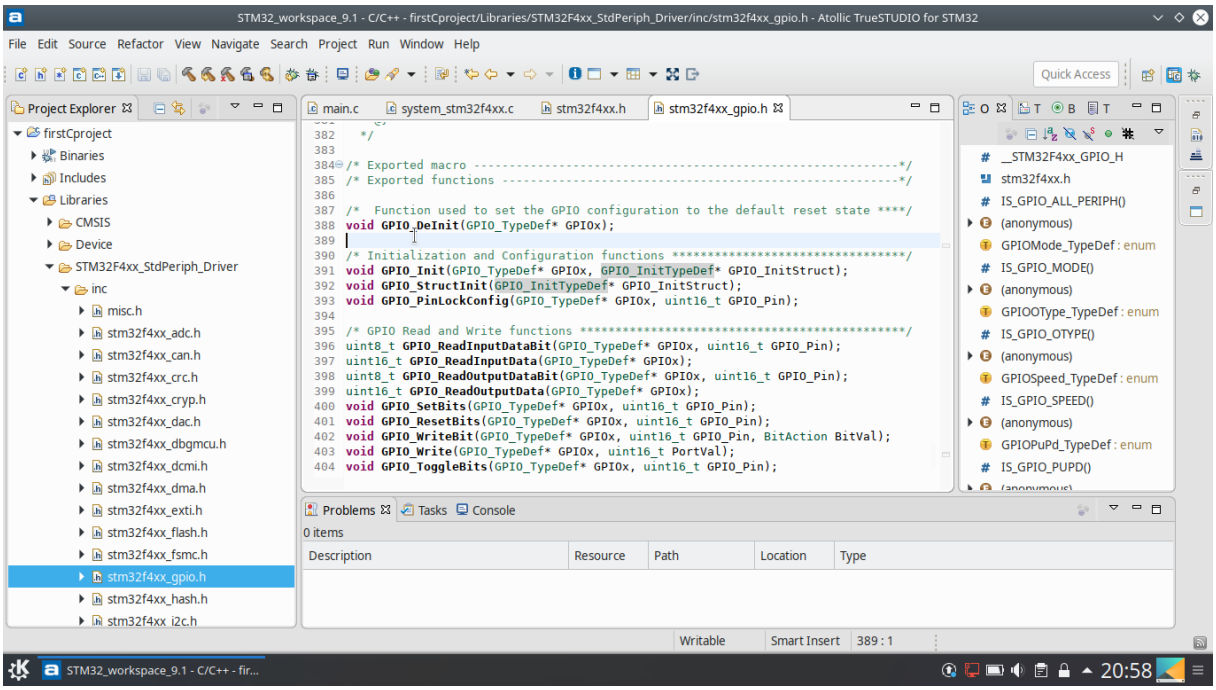


Maintenant que le projet est créé, il convient de regarder les fichiers contenus dans le répertoire src. On apprend dans le fichier system_stm32f4xx.c que les pins sont divisés en ports que l'on sélectionne avec GPIOX (X doit être remplacé par la lettre correspondant au port) qui est une structure avec divers paramètres.
On apprend surtout qu'il faut activer l'horloge de chaque port pour l'utiliser dans le registre RCC->AHB1ENR, pour connaître le bit correspondant au port voulu, on peut utiliser la documentation ou chercher dans les fichiers de projets avec un clic droit et open declaration.

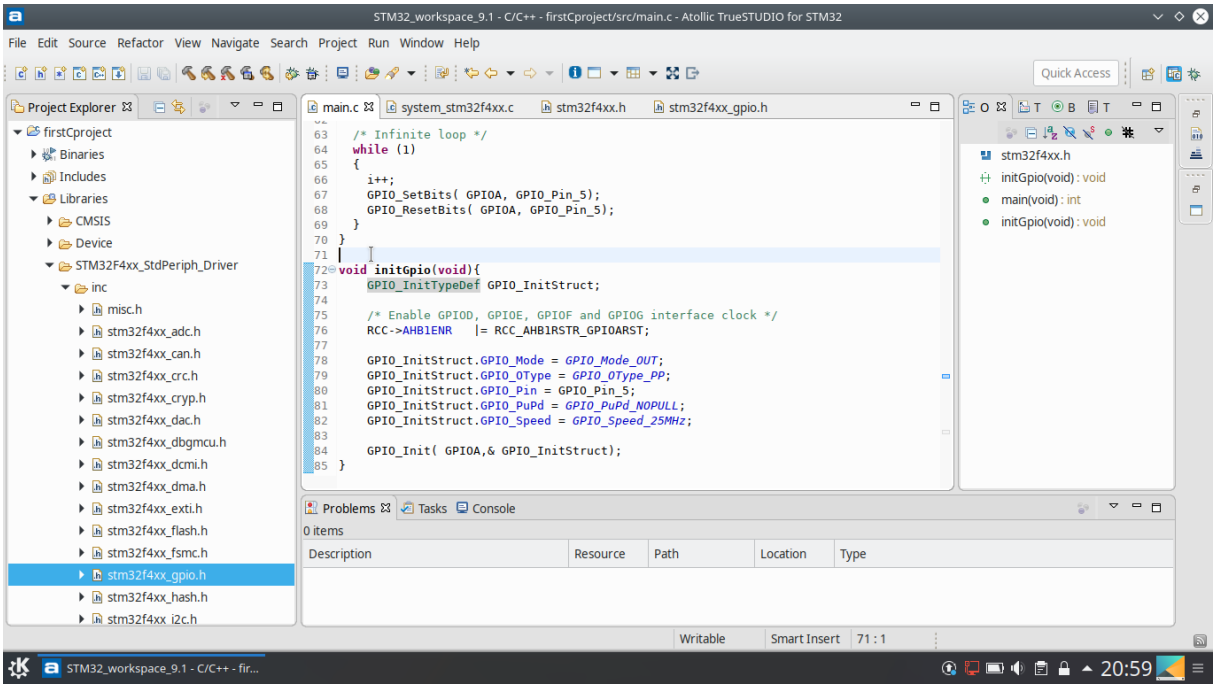

STM32_workspace_9.1 - C/C++ - firstCproject/src/system_stm32f4xx.c - Atollic TrueSTUDIO for STM32
File Edit Source Refactor View Navigate Search Project Run Window Help
Quick Access
Project Explorer
firstCproject
Binaries
Includes
Libraries
src
main.c
startup_stm32f40xx.s
stm32f4xx_conf.h
stm32f4xx_it.c
stm32f4xx_it.h
system_stm32f4xx.c
Debug
firstCproject.elf.launch
stm32f4_flash.ld
main.c
system_stm32f4xx.c
stm32f4xx.h
| PD15 <-> FSMC_D1 | PE11 <-> FSMC_D8 |
+------------------| PE12 <-> FSMC_D9 |
| PE13 <-> FSMC_D10 |
| PE14 <-> FSMC_D11 |
| PE15 <-> FSMC_D12 |
+-------------------+
*/
/* Enable GPIOD, GPIOE, GPIOF and GPIOG interface clock */
RCC->AHB1ENR |= 0x00000078;
/* Connect PDx pins to FSMC Alternate function */
GPIOD->AFR[0] = 0x00cc00cc;
GPIOD->AFR[1] = 0xcccccccc;
/* Configure PDx pins in Alternate function mode */
GPIOD->MODER = 0xaaaa0a0a;
/* Configure PDx pins speed to 100 MHz */
GPIOD->OSPEEDR = 0xffff0f0f;
/* Configure PDx pins Output type to push-pull */
GPIOD->OTYPER = 0x00000000;
/* No pull-up, pull-down for PDx pins */
GPIOD->PUPDR = 0x00000000;
/* Connect PEx pins to FSMC Alternate function */
GPIOE->AFR[0] = 0xcccccccc;
stm32f4xx.h
VECT_TAB_OFFSET
PLL_M
PLL_N
PLL_P
PLL_Q
SystemCoreClock : uint32_t
AHBPrescTable : const volatile u
SetSysClock(void) : void
SystemInit_ExtMemCtl(void) : vc
SystemInit(void) : void
SystemCoreClockUpdate(void) :
SetSysClock(void) : void
SystemInit_ExtMemCtl(void) : vc
Problems
Tasks
Console
0 items
Description
Resource
Path
Location
Type
Writable
Smart Insert
459 : 62
STM32_workspace_9.1 - C/C++ - fir...
20:52

Ensuite, il faut définir les paramètres du pin choisi, une recherche nous amène au fichier stm32f4xx_gpio.h qui contient une structure GPIO_InitTypeDef et des fonctions comme GPIO_Init.





Sur le fichier main.c, il ne nous manque plus qu'à implémenter une fonction destinée à initialiser les pins avec les bons paramètres. Il suffit ensuite de rajouter dans la fonction main les appels des fonctions permettant de contrôler l'état du pin.



Il est maintenant temps de compiler le projet et de brancher le STLINK pour programmer la cible et passer au débuggage.