

TP1 : Programmation du STM32F4 dans l'environnement Keil uVision

A. Objectifs


- Se familiariser avec l'environnement de développement intégré (IDE) Keil uVision et la plateforme matérielle STM32F4 discovery.
- Programmation directe des entrées/sorties GPIO (sans l'utilisation des bibliothèques spécifiques fournies par le constructeur)

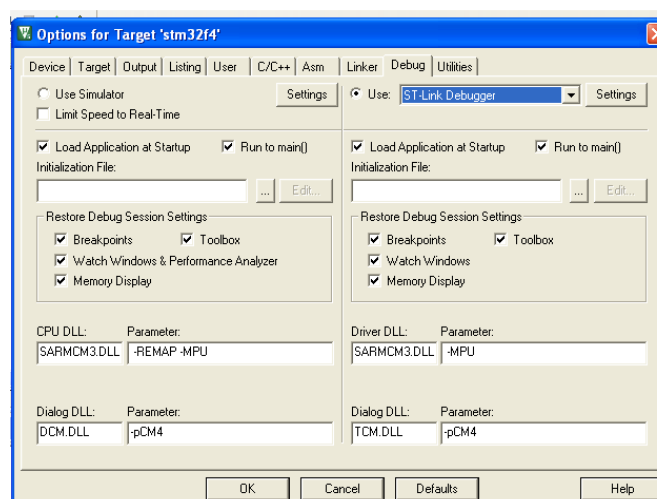
B. Premier projet

1. Créer un nouveau projet







- Menu : Project → new uvision project
 - choisir un dossier et un nom pour votre projet
 - Choisir STM32F4 series → STM32F407 → STM32F407VGTx ;
 - Dans la boîte de dialogue « Manage Runtime Environment »
 - Cocher Device → startup; Noter la couleur orange, et examiner le message qui s'affiche en bas. Ce message indique une dépendance non satisfaite
 - Satisfaire la dépendance en sélectionnant CMSIS → Core

2. Configuration du projet

- Un projet peut être configuré pour être exécuté sur le matériel réel ou sur un simulateur (avec des limitations)
- Ouvrir la boîte de dialogue options du projet  (ou menu : Project → options of target ...)
 - Cas matériel réel :
 - Dans l'onglet « debug », choisir l'option « use ST link debugger ». Choisir « settings ». S'assurer que la carte est bien détectée (dans la partie SW devices) et qu'un algorithme de chargement de flash est bien présent (dans l'onglet Flash Downloads)
 - Cas simulateur
 - Dans l'onglet « debug », choisir l'option « Use simulator »



3. Compilation et exécution

- La compilation du projet (appelée aussi construction ou *build*) est déclenchée par le menu Project / Build Target (ou F7 ou )
- Si un ou plusieurs fichiers source (*.c) sont modifiés, il faut relancer la construction (F7). Seuls les fichiers modifiés sont alors recompilés automatiquement. Cette règle ne s'applique pas pour les fichiers d'entêtes (*.h). Si un fichier d'entête est modifié, il faut forcer la reconstruction du projet entier avec l'outil *rebuild* 
- Noter le message d'erreur qui s'affiche en bas
- Ajouter un fichier main.c au groupe «source group 1» (menu contextuel → add new item to group ...)
 1. choisir le type C File (.c) de la liste ; entrer *main* dans le champ « Name »
- Éditer *main.c* en ajoutant une fonction *main* qui implémente une boucle infinie dans laquelle on incrémente une variable globale *x*
- Lancer l'exécution interactive (Debug) du projet en cliquant sur l'outil icône 
 1. L'exécution s'arrête par défaut au niveau de la fonction *main*
 2. Appliquer un reset au processeur via le bouton 
 3. Suivre l'exécution pas-à-pas via le bouton 
 4. Placer un point d'arrêt (break point) dans le corps de la boucle *while* de la fonction *main* en cliquant sur la bande grise à gauche du code source. Un point rouge devra s'afficher.
 5. Continuer l'exécution normalement via le bouton . Le processeur devra s'arrêter au niveau du point d'arrêt. Vous pouvez alors reprendre pas-à-pas, etc.
 6. Ajouter la variable globale *x* à l'inspection (watch) via le menu contextuel en cliquant sur cette variable dans le code source (Add x to ... ⇨ watch 1)
 7. Examiner l'évolution de la variable *x*

C. Programmation des GPIO

Dans cette partie, on se propose d'implémenter une animation simple qui consiste à faire défiler les 4 diodes LED (*LD3 ... LD6*) présents sur la carte dans un sens, puis dans un autre lorsqu'on appuie sur le bouton poussoir *User*.

Le défilement des diodes consiste à allumer une seule diode à la fois pendant un cycle (250 ms), puis l'éteindre et passer à la diode suivante (cycle suivant), ainsi de suite.

Si l'utilisateur appuie sur le bouton *User*, le sens de défilement s'inverse à compter du prochain cycle.

Examiner le fichier `system_stm32f4xx.c`

- ouvrir le fichier d'entête `stm32f4xx.h` (clic-droit sur `stm32f4xx.h` → « open document ... »)
- Dans `stm32f4xx.h`, repérer la ligne contenant « `#include "stm32f407xx.h"` » et ouvrir le fichier d'entête `stm32f407xx.h`. Ce fichier contient la définitions des différents types de données relatifs aux périphériques du micro-contrôleur STM32F407

Pour optimiser la consommation d'énergie, la majorité des périphériques du STM32F407 ont leurs horloges désactivées par défaut. Activer/désactiver l'horloge d'un périphérique passe par un autre périphérique spécifique appelé

RCC (Reset and Clock Control). Le registre contenant les bits d'activation des ports GPIO s'appelle AHB1ENR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									DMA2EN	DMA1EN	Reserved				
									rw	rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CRCEN	Reserved				GPIOH EN	Reserved		GPIOEEN	GIOD EN	GPIOC EN	GPIOB EN	GPIOA EN
			rw					rw			rw	rw	rw	rw	rw

1. Écrire une fonction void RCC_GPIO_Enable(int mask) qui permet d'activer les horloges des ports GPIO correspondant à 1 dans le masque
Exemple RCC_GPIO_Enable(3) permet d'activer GPIOA et GPIOB
2. Écrire la fonction void GPIO_Configure(GPIO_TypeDef *GPIOx, uint32_t num, uint32_t mode, uint32_t otype, uint32_t ospeed, uint32_t updown) qui permet de configurer le pin numéro num de GPIOx
3. Écrire la fonction uint32_t GPIO_PinRead (GPIO_TypeDef *GPIOx, uint32_t num) qui permet de retourner la valeur du pin numéro num du GPIOx
4. Écrire la fonction void GPIO_PinWrite (GPIO_TypeDef *GPIOx, uint32_t num, uint32_t val) qui permet d'écrire la valeur val dans le pin numéro num du GPIOx
5. Écrire la fonction void GPIO_PinsClear (GPIO_TypeDef *GPIOx, uint32_t mask) qui permet de remettre à 0 les pins correspondant à 1 dans le mask
Exemple GPIO_PinsClear(GPIOD, 5) permet de mettre à 0 les pins numéro 0 et 2
6. Écrire le programme principal dans le fichier main.c
 - Dans la fonction main, configurer le périphérique SysTick et les pins GPIO utilisés.
 - Définir la routine de gestion d'interruption du SysTick : void SysTick_Handler(){...} qui permet d'implémenter l'animation souhaitée

Remarque importante : le programme doit pouvoir détecter même des clics brefs sur le bouton User. Il doit aussi pouvoir gérer les appuis longs sur ce bouton en les traitant comme un seul clic

