1 – GPIO / LEDs

Objectifs: *introduction à la programmation de périphériques (GPIO/LEDs), programmation directe par écriture dans les registres, programmation utilisant la librairie ARM CMSIS.*
Préparation: lire la parie **1.1** Description du sujet, ainsi que STM32F4 Series Reference Manual p. 187-189.
A quelle horloge est connecté le port `GPIOD`?

Réalisation : les parties **1.2** et **1.3** doivent être validées par l'encadrant au cours de la séance.

La préparation est obligatoire et vérifiée en début de séance.

1.1 Description

Les séances de Travaux Pratiques se déroulent sur des cartes STM32F4-Discovery développées par STMicroelectronics et équipées d'un processeur ARM Cortex-M4. L'environnement de compilation et de debug intégré utilisé est le logiciel Keil µVision.

L'objectif de cette séance est de faire clignoter une LED. Les LEDs de la carte STM32F4-Discovery sont reliées aux sorties d'un périphérique appelé GPIO (General Purpose Input/Output). Le GPIO est un composant très utilisé qui permet de mettre à 0 ou à 1 les sorties qu'il contrôle.

Configuration du projet en C:

- Ouvrir Keil µVision.
- Project → Create new µVision project
  Sélectionner le répertoire `C:\users\elec3\TP_microprocesseurs\TP1` (à créer s'il n'existe pas). **/!\ ne jamais travailler dans Mes Documents, ni sur le Bureau Windows.**
  Nom du fichier : TP1_GPIO
- Dans la fenêtre Select Device for Target 'Target 1', cliquer dans STMicroelectronics et sélectionner STM32F4 Series / STM32F407 / STM32F407VG.
- Dans la fenêtre Manage Run-Time Environment, sélectionner: Board Support → STM32F4-Discovery (à sélectionner dans le menu déroulant), CMSIS → CORE, Device → Startup
- Pour pouvoir utiliser les librairies CMSIS pour les LEDs et le GPIO, cocher aussi Board Support → STM32F4-Discovery → LED et Device → GPIO dans le menu, puis faites OK.
- Dans le menu projet à gauche, clic droit sur Source Group 1, puis Add New Item to Group 'Source Group 1' / C File (.c) / Name: `main_led.c`, puis Add.
- Essayez de compiler, si vous avez l'erreur « function assert param declared implicitely », clic droit dans Target1 → Options for Target 'Target 1'… → C/C++ → Preprocessor Symbols → Define : `USE_STDPERIPH_DRIVER`

Configuration du debugger ST LINK:

- Dans le menu Options → Debug → ST Link Debugger → Settings → Port: SW.

Premier test:
Utilisez le code ci-dessous pour compiler (Build) et charger (Load)

```
#include "stm32f4xx.h" //librairie
int main(){        //programme principal
     while(1){ } //boucle principale
}
```

- Si vous avez l'erreur « no ST-LINK Device found », allez dans C:\Keil\ARM\STLink\USBDriver et lancez « dpinst_amd64 »

On procèdera de la même façon à chaque fois qu'il faudra créer un projet.

Programmation des registres en C:

- Toute la documentation est accessible depuis l'onglet Books dans l'onglet de gauche. En particulier la description de tous les registres se trouve dans STM32F4 Series Reference Manual. La documentation plus spécifique à la carte STM32F4-Discovery se trouve dans User Manual (STM32F4-Discovery).
- Les adresses des registres sont toutes prédéfinies dans le fichier `stm32f4xx.h` par l'intermédiaire de `#define`, de structures et de définitions de type. Par exemple, pour écrire la configuration `0x1` dans le registre `GPIOx_MODER` du `GPIOG`, on utilisera l'affectation: `GPIOG->MODER = 0x1;`

L'objectif est d'écrire un programme qui fait clignoter la LED `LD4` (verte). Afin de comprendre en détail le fonctionnement et l'utilisation des périphériques (ici dans le cas du GPIO), nous utiliserons deux approches: La programmation directe des registres et l'utilisation de la librairie standard développée par le constructeur (CMSIS, ARM Cortex Microcontroller Software Interface Standard).

1.2 Programmation directe des registres

La LED `LD4` est connectée au port `PD.12` du GPIO (cf User Manual STM32F4-Discovery). En s'aidant de la documentation STM32F4 Series Reference Manual décrivant le fonctionnement et les registres des périphériques (à partir de la page 198), l'objectif consiste à écrire les trois fonctions suivantes (dans le fichier `main_led.c`) qui permettront de manipuler la LED `LD4`:

- `void LD4_Initialize(void)` qui permet d'initialiser le GPIO pour utiliser la LED `LD4`. Les registres à configurer sont:
  - `RCC_AHB1ENR` pour activer l'horloge du port `GPIOD`.
  - `GPIOx_MODER` pour configurer la broche `PD.12` en General purpose output mode.
  - `GPIOx_OTYPER` pour configurer la broche `PD.12` en Output push-pull.
  - `GPIOx_OSPEEDR` pour configurer la broche `PD.12` en 50 MHz Fast speed.
  - `GPIOx_PUPDR` pour configurer la broche `PD.12` en Pull-down.
- `void LD4_On(void)` qui permet d'allumer la LED `LD4`. Les registres à configurer sont:
  - `GPIOx_BSRRL` pour mettre la broche `PD.12` à 1.
- `void LD4_Off(void)` qui permet d'éteindre la LED `LD4`. Les registres à configurer sont:
  - `GPIOx_BSRRH` pour mettre la broche `PD.12` à 0.

Ecrire un programme principal qui fait clignoter la LED `LD4` en utilisant ces 3 fonctions. Pour que le clignotement soit visible, il faudra ralentir l'exécution du programme en utilisant par exemple une boucle for entre deux clignotements: `for (i=0; i<10000000; i++);`

1.3 Utilisation de la librairie standard CMSIS

L'ARM Cortex Microcontroller Software Interface Standard (CMSIS) est une librairie de pilotes prédéfinis pour la série des processeurs ARM Cortex-M. L'utilisation de ces librairies simplifie le développement d'applications. Par exemple, plusieurs fonctions déjà écrites pour la manipulation des LEDs sont présentes dans le fichier `LED.c` (onglet Project / Board Support).

Ecrire un deuxième programme (dans un fichier `main_gpio_cmsis.c`) qui fait clignoter les quatre LEDs `LD3`, `LD4`, `LD5`, `LD6`, numérotées de 0 à 3 en utilisant les fonctions CMSIS fournies dans `LED.c`.

Modifier le programme précédent pour allumer successivement chaque LED précédente en effectuant une rotation dans le sens horaire.