Informatique embarquée
TD — Bus de communication

UART et I²C

Objectifs
- Comprendre comment fonctionnent les bus de communication série (UART et I²C)
- Comprendre la différence entre communication synchrone et asynchrone
- Savoir mettre en œuvre ces protocoles dans le contexte de MBED (programmation d'un bus terrain).

Matériel requis.
- Une plateforme nucleo-board STM32F446RE
- Une carte fille avec des LEDS
- Un capteur de température

1 Connexion série asynchrone – UART

On souhaite réaliser une communication série entre la carte STM32F et un émulateur de terminal UART sur l'ordinateur.

1. Aller sur le site : https ://www.mbed.com/en/ puis **créer un nouveau projet**.
2. Écrire un programme qui permet de tester le port série en émission et en réception. Attention de vérifier que les paramètres de transmission sont identiques entre l'hôte et la carte STM32F (baud rate, bit de parité, bit de stop, format, *etc.*).
3. Créer une fonction qui permet d'incrémenter un compteur à partir de l'appui sur le bouton utilisateur.
   - Utilisez des interruptions pour le bouton !
4. On souhaite que la valeur du compteur soit envoyée uniquement lorsque l'ordinateur envoie à la carte STM32F, le caractère « C ». Modifier le programme en conséquence.

2 Communication entre deux cartes.

On souhaite réaliser une communication série entre deux cartes STM32F.

1. Aller sur le site : https ://www.mbed.com/en/ puis **créer un nouveau projet**.
2. Écrire un programme qui permet de configurer le port série « serial 2 ».
   - Indice : il faut sélectionner des broches qui possèdent la fonction de communication RT série ou TX série (regardez le *pinout* !)
3. Créer une fonction qui permet d'incrémenter un compteur à partir de l'appui sur le bouton utilisateur.
4. Une carte doit compter le nombre d'appuis sur le bouton, et enverra le compte *uniquement* si on a saisi le caractère « C ». L'autre carte recevra ce compte, et l'affichera à l'écran. La 2è carte saura qu'elle a reçu des données grâce à une interruption. Quelques indices :

— On peut associer une interruption à la réception de données sur une connexion série grâce à la fonction `attach`.
— Connecter les deux cartes entre elles, mais aussi les deux sur le même PC : utilisez 2 instances de Tera Term pour visualiser/saisir les caractères.

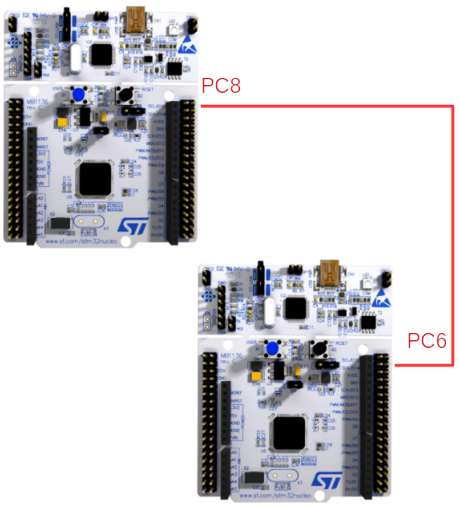

FIGURE 1 – Schéma de connexion entre 2 cartes STM32 pour l'exercice 1.

3 Connexion série synchrone – I²C

On souhaite réaliser un *datalogger* (un enregistreur de données) permettant d'enregistrer les mesures d'un capteur de température sur 1h, à raison d'une mesure toutes les minutes. Pour cet exercice, on utilisera le capteur MCP9808 de la carte Adafruit (https://learn.adafruit.com/adafruit-mcp9808-precision-i2c-temperature-sensor-guide/overview).

1. Cloner un projet précédent, et supprimer tout le code de main.cpp à l'exception de `#include "mbed.h"` et `int main() { }`.
2. Lire le datasheet de la carte
3. Proposer un synoptique pour le programme
4. Proposer une fonction permettant de réaliser l'initialisation des paramètres du capteur.
5. Ecrire une fonction permettant de récupérer une valeur de température
6. A l'aide du timer, proposer une fonction permettant de générer une interruption toutes les minutes.
7. Finaliser le programme de manière à enregistrer une mesure de la température toutes les minutes pendant 1h. On vérifiera les valeurs acquises sur la console du PC via la communication UART de l'exercice 1.

Annexe — Liste de fonctions utiles

- `void wait(float delay);` force le micro-processeur à attendre `delay` micro-secondes avant d'exécuter l'instruction suivante dans le programme.
- Déclarer une connexion série UART : `Serial nom_connexion(BROCHE_TX, BROCHE_RX);`
- Lire un caractère sur une connexion UART :

```
char c;
/* . . . */
c = nom_connexion.getc();
```

- Écrire le caractère `'a'` sur une connexion UART :

```
nom_connexion.putc('a');
```

- `fall(`*`nom_fonction`*`)`, `rise(`*`nom_fonction`*`)` : fonctions rattachés à la variable qui « capture » les interruptions, et qui indiquent si on capture sur un front montant ou descendant.
  - Utilisation : `variable.rise(&nom_fonction);`
  - Utilisation : `variable.fall(&nom_fonction);`
- `attach(`*`nom_fonction`*`)` : fonction associée à certains types de variables, par exemple, `Ticker` et `Serial`, et qui permettent de lever une interruption lorsqu'un événement survient, et d'appeler `nom_fonction`
  - Utilisation : `variable.attach(&nom_fonction);`

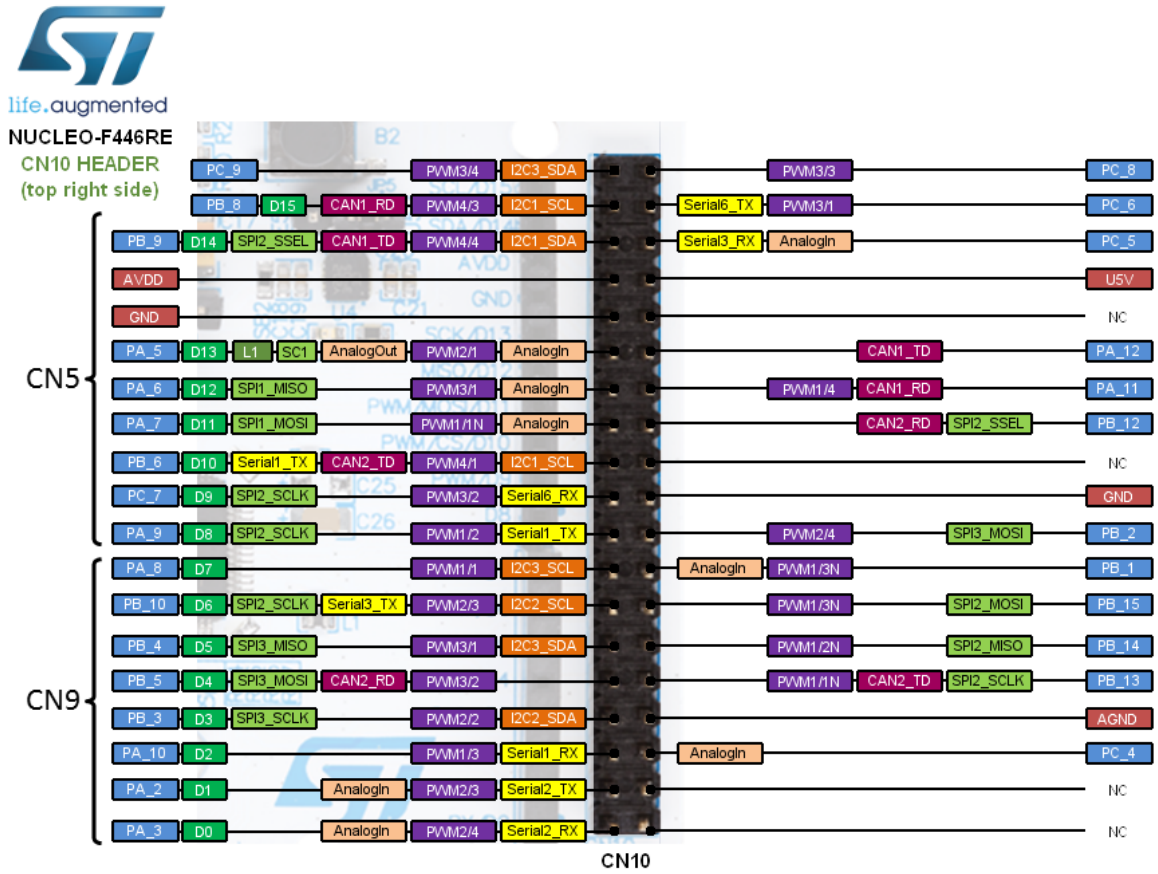

FIGURE 2 – Schéma des entrées-sorties pour la carte Nucleo-64.