

---

# Programmation en C des microcontrôleurs,

## Exercices Sujet0\_\_c\_\_micro

---

Licence EPOCS \*(ECUE23)

Pierre Pardo

2020-2021

Ce sujet contient 2 exercice(s).

### Comment répondre ?

Ce sujet comporte des exercices, les réponses seront données sous formes de fichier sources (\*.c et/ou \*.h) ou de fichier texte (\*.txt, \*.odt) quand il y a des réponses associées soit:

- Sur un support numérique (clef USB) à la fin du cours ou du TP.
- Par email à [pierre.pardo@univ-amu.fr](mailto:pierre.pardo@univ-amu.fr), au maximum le lendemain du cours ou du TP.

Il y aura souvent plusieurs fichiers aux noms identiques, il faudra transférer les arborescences complètes des fichiers. Le plus simple est de compresser les dossiers.

---

\*Licence professionnelle Électronique Pour Objets Connectés et Smart-Grids



## Carte Nucleo / USB

I Nous allons utiliser la carte Nucleo seule.

Nous allons utiliser une tâche **FreeRTOS** dont le rôle va être :

- Allumer la led
- Attendre un délai
- Eteindre la led
- Attendre un délai
- ...et recommencer ...

(a) Dans un premier temps vous allez copier un dossier (**AImporter**) depuis ma clef **USB** sur votre PC. Dans un dossier qui ne doit PAS être celui du workspace.

- Importer le projet `_basicProjectFreeRTOSLedReg` dans **STM32CubeIDE** (voir la méthode ci dessous)
- Renommer le immédiatement en `basicProjectFreeRTOSLedReg_imp` depuis l'environnement (voir la méthode ci dessous)

Pour importer un projet :

- Ouvrir le menu : **File/Import**
- Sélectionner la source d'importation : **General/Existing Projects into Workspace**
- Bouton **Next>**
- Bouton **Browse...** (en face de **Select root directory**) et choisissez le chemin ou vous avez copié le dossier **AImporter**.
- Sélectionner en cochant la case en face du projet `_basicProjectFreeRTOSLedReg`
- Vérifier que la case à cocher **Copy projects into workspace** est sélectionnée
- Bouton **Finish**

Pour renommer un projet depuis l'environnement :

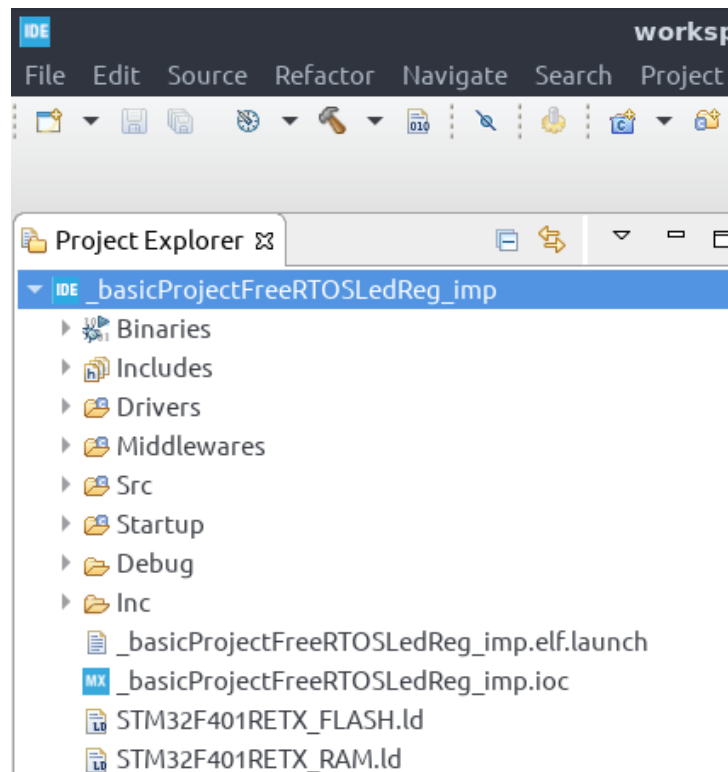
- Click droit sur le nom du projet
- Menu renommer
- Choisir le nouveau nom



## Erreur possible lors du renommage de projet

Il peut arriver que le fichier `launch` ne soit pas renommé. Il faut alors le faire manuellement.

Ce fichier se trouve directement dans la liste des fichiers du projet. Il a une extension `.elf.launch`. Vous devez le renommer comme le nom du projet (mais avec l'extension `.elf.launch`).



- (b) Compiler et transférer ce programme sur la carte Nucleo. Normalement la led doit clignoter.
- (c) Utilisation de STM-Studio.  
Nous allons utiliser STM-Studio pour visualiser le comportement de notre programme.  
Pour cela, vous allez :
  - Configurer STM-Studio pour visualiser une variable à l'adresse 0x40020014 (GPIOA->ODR).
  - Ajouter la variable au VarViewer
- (d) `vTaskDelay()` n'est pas recommandé pour des tâches périodiques, mais plutôt pour avoir un délai relatif. Modifier le programme pour utiliser `vTaskDelayUntil`.
- (e) Vérifier le fonctionnement avec STM-studio.



## Carte Nucleo / USB

I Nous allons utiliser la carte Nucleo seule.

Nous allons utiliser deux tâches **FreeRTOS** dont le rôle va être :

- Tâche 1 **vTaskDisplay1**
    - Afficher un message sur la liaison série
    - Attendre un peu (1s)
  - Tâche 2 **vTaskDisplay2**
    - Afficher un message sur la liaison série
    - Attendre un peu (300ms au début)
- (a) — Importer le projet **basicProjectFreeRTOSwDebugShareSerial** dans **STM32CubeIDE**  
 — Renommer le immédiatement en **\_basicProjectFreeRTOSwDebugShareSerial\_imp**.
- (b) Analyser le code source (le **main.c**) pour voir comment est organisé ce programme.
- (c) Compiler et transférer ce programme sur la carte **Nucleo**.  
 Lancer un terminal (**Putty** ou **Termite**) (115200,8,N,1 sans contrôle).  
 Cela a l'air de bien se passer, avec une suite d'affichage de "XXXXXXXXXXXX...XXX"  
 et de "---....--".
- (d) Modifier la **vTaskDisplay2** pour avoir un délai de 30 ms.

```
const portTickType DELAY = pdMS_TO_TICKS(30);
```

Relancer le programme.

Vous devez normalement voir des soucis d'accès à la ressource partagée **printf** qui utilise l'**USART2**.

Nous avons un **problème d'accès concurrent**.

Vous pouvez essayer de changer les priorités des tâches dans le **xTaskCreate**.  
 Mais cela ne résoudra pas le problème. Cela peut juste le cacher (un peu).

- (e) Rajouter des **Mutex** pour protéger la section critique **printf**.  
 Tester, en modifiant les délais et la priorité.
- (f) Il existe une tâche **vTaskDisplayDebugs** qui n'est pas lancé actuellement car la fonction **xTaskCreate** est commentée

```
// BaseType_t t3 = xTaskCreate(vTaskDisplayDebugs, "Dbg", 1000, (void *)
↪ NULL, tskIDLE_PRIORITY + 1, NULL);
```

Vous pouvez la décommenter, cete tâche affiche des informations sur l'état des tâches. Vous devez ajouter dans cette fonction la protection par **Mutex** du **printf**.

### Rappel API **Mutex**

Vous pouvez trouver les définitions complètes des fonctions liées aux **Mutex** dans la documentation. Mais les fonctions les plus utiles sont définies ci-dessous.

```
SemaphoreHandle_t xSemaphoreCreateMutex( void );  
BaseType_t xSemaphoreTake( SemaphoreHandle_t xSemaphore, TickType_t  
↪ xTicksToWait );  
BaseType_t xSemaphoreGive( SemaphoreHandle_t xSemaphore );
```

Une attente infinie peut être choisie avec `portMAX_DELAY` comme `xTicksToWait`.  
Sinon il faut tester la valeur de retour de la fonction `xSemaphoreTake`.