
Programmation en C des microcontrôleurs, Exercices Sujet0bis__c__micro

Licence EPOCS *(ECUE23)

Pierre Pardo

2020-2021

Ce sujet contient 1 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, 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

TP 1: Synchronisation Tâches

Nous allons utiliser des tâches qui devront se synchroniser entre elles, et éventuellement recevoir une synchronisation d'une interruption :

- Tâche 1 `vTaskReceiver`
 - Est bloquée, en attente d'un événement de synchronisation
 - Quand elle se débloque, elle affiche un message
- Tâche 2 `vTaskSender`
 - Attend un temps aléatoire
 - Transmet un message de synchronisation
 - Et recommence
- Une interruption matérielle peut se produire (connectée au bouton bleu)
 - Elle envoie un message de synchronisation
- (a) — Importer le projet `basicProjectFreeRTOSvDebugSynchroTask` dans STM32CubeIDE
 - Renommer le immédiatement en `basicProjectFreeRTOSvDebugSynchroTask_imp`.
- (b) Analyser le code source (le `main.c`) pour voir comment est organisé ce programme.

Compléter les zones qui ressemblent à :

```
////////////////////////////////////////  
//////// Attente d'un événement de synchronisation  
////////////////////////////////////////
```

Rappel API Sémaphore

```
SemaphoreHandle_t xSemaphoreCreateBinary( void );  
BaseType_t xSemaphoreTake( SemaphoreHandle_t xSemaphore, TickType_t  
    ↪ xTicksToWait );  
BaseType_t xSemaphoreTakeFromISR( SemaphoreHandle_t xSemaphore, signed  
    ↪ BaseType_t *pxHigherPriorityTaskWoken );  
BaseType_t xSemaphoreGive( SemaphoreHandle_t xSemaphore );  
BaseType_t xSemaphoreGiveFromISR( SemaphoreHandle_t xSemaphore, signed  
    ↪ BaseType_t *pxHigherPriorityTaskWoken );
```

- (c) Compiler et transférer votre programme sur la carte Nucleo.
Lancer un terminal (Putty ou Termite) (115200,8,N,1 sans contrôle).
Vous devriez voir apparaître des messages `Sync received` et la led doit clignoter (avec un SOS).
- (d) Essayer de changer le texte affiché, même en mettant un texte long, il n'y a pas de soucis d'affichage. L'affichage est synchronisé avec le sémaphore.
- (e) Dans l'interruption, il y a 2 endroits possibles pour l'acquitter.
L'acquiescement est fait avec la ligne :

```
EXTI_ClearITPendingBit(EXTI_Line13); /* Clear the EXTI line 13 pending bit  
    ↪ */
```

- Quel est le meilleur endroit pour l'acquiescement ?
- Y a-t-il un meilleur endroit ?