Exercices FreeRTOS

EXERCICE 10-1 : DECOUVERTE FREERTOS

OBJECTIFS

Cet exercice a pour but de découvrir **FreeRTOS** en se basant sur l'exemple "basic" de Microchip.

DESCRIPTION DE L'APPLICATION SOUHAITEE

Il s'agit de mettre en place 3 tâches indépendantes faisant chacune clignoter une LED à une fréquence différente.

DETAILS DE LA REALISATION

COPIE DU PROJET HARMONY

Création d'un projet en copiant le projet C:\microchip\harmony\v<n>\apps\rtos\freertos\basic et en le renommant Mc32Ex10_1.
Choisir la configuration pic32mx_sk :

- Supprimer les autres configurations (pi32mz).
- "Remove from project" des fichiers qui correspondent aux autres configurations.

AJOUT DU BSP PIC32MX_SKES

Dans le MHC, sélectionner le BSP du kit SKES.

AJOUT AFFICHAGE INVITE

On ajoute l'init. du lcd et l'affichage d'un message sur les 2 premières lignes dans l'initialisation de app.c.
Exemple d'affichage :

```
Ex_10_1 FreeRTOS
Nom
```

ESSAIS ET DECOUVERTE

Si tout s'est bien passé à la compilation, on charge le projet sur le kit. Le message doit s'afficher et les LED 1,2 et 3 doivent clignoter :

- LED1 (utilisé dans app1.c), toggle toutes les 200 ms,
- LED2 (utilisé dans app2.c), toggle toutes les 200 ms,
- LED3 (utilisé dans app3.c), toggle toutes les 50 ms.

MODIFICATIONS ET ESSAIS

MODIFICATION DES PRIORITES DES TACHES

Dans le MHC, établissez à 4 la priorité des applications app1, app2 et app3.

MODIFICATION DU CONTENU DES TACHES

- Dans APP_Tasks de App.c mettre en commentaire les :
  // xQueueSend(xQueue, &ulValueToSend1, 0U);

- Dans APP1_Tasks de App1.c, on met tout en commentaire et on remplace par :

```
while(1) {
        vTaskDelay((TickType_t)1);
        BSP_LEDToggle( BSP_LED_1 );
}
```

- Dans APP2_Tasks de App2.c, on met tout en commentaire et on remplace par :

```
while(1) {
        vTaskDelay((TickType_t)2);
        BSP_LEDToggle( BSP_LED_2 );
}
```

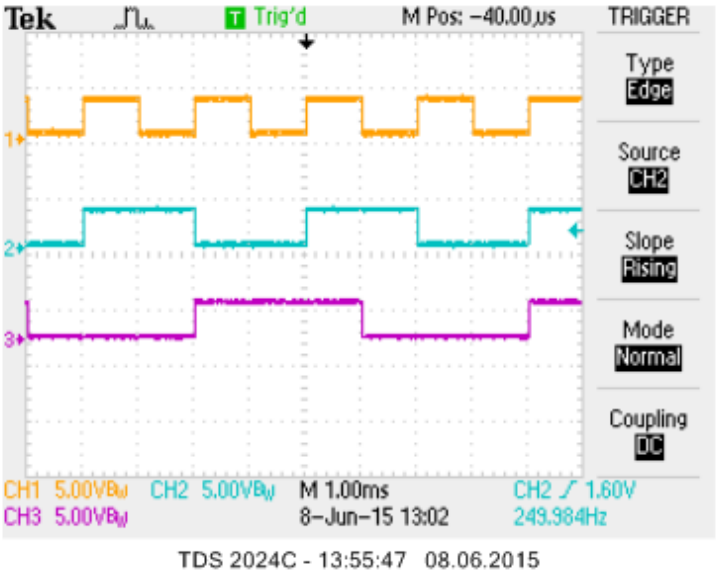
- Dans APP3_Tasks de App3.c, on introduit :

```
while(1) {
        vTaskDelay((TickType_t)3);
        BSP_LEDToggle( BSP_LED_3 );
}
```

CONTROLE SIGNAUX SUR LES LEDS

On doit obtenir :

- une inversion toutes les 1 ms sur led_1 (canal1),
- une inversion toutes les 2 ms sur led_2 (canal2),
- une inversion toutes les 3 ms sur led_3 (canal3).



EXERCICE 10-2

OBJECTIFS

Cet exercice a pour but de mettre en oeuvre une communication inter-tâches dans FreeRTOS à l'aide de queues.

DESCRIPTION DE L'APPLICATION SOUHAITEE

En vous inspirant de l'exercice précédent, réalisez une application qui utilise les 4 boutons-poussoirs S6 à S9.
A chaque appui, le code de la touche '1', '2', '3' ou '4' est :

- affiché sur le lcd,
- envoyé via le port série.

DETAILS DE LA REALISATION

AFFICHAGE

Lignes 1 et 2 utilisées pour l'invite. La ligne 4 affiche la touche appuyée.

Exemple d'affichage :

```
Ex_10_2 FreeRTOS
Nom

3
```

TACHES

Génération d'un projet Harmony qui comporte 3 tâches :

- **appclav** pour la gestion des boutons-poussoirs.
  Cette tâche gère les antirebonds. Elle fait un envoi du caractère à afficher dans une queue destinée au LCD et dans une autre queue destinée au port série.
  Priorité 3. Tâche réveillée toutes les 1 ms. Pour une période constante, utilisation de vTaskDelayUntil().

- **applcd** pour la gestion du LCD.
  Priorité 2. Affiche le caractère sur le LCD dès qu'il est disponible dans la queue.
  Cette tâche est la seule à utiliser le LCD.

- **appcomm** pour la gestion du port série.
  Priorité 2. Envoie le caractère via UART dès qu'il est disponible dans la queue.
  Cette tâche est la seule à utiliser l'UART.