Noyau Temps Réel

TP FreeRTOS

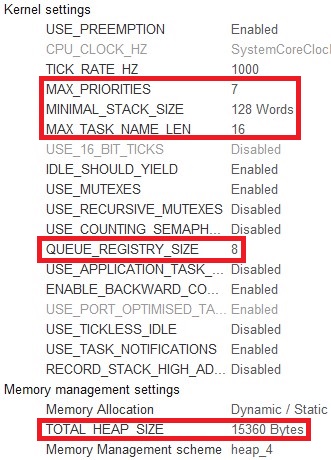
BOUDET Anthony

MANOLOGLOU Jeremy

1 FreeRTOS, tâches et sémaphores

* 1. Tâche simple

1. En quoi le paramètre TOTAL\_HEAP\_SIZE a-t-il de l’importance ?

****

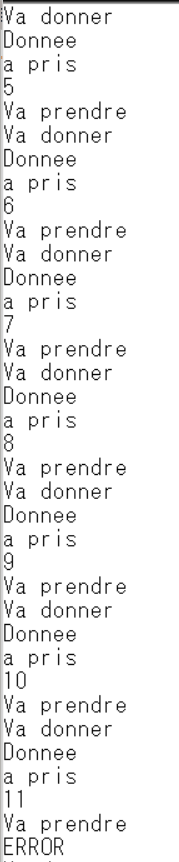
*Paramètres pertinents*

Le paramètre **TOTAL\_HEAP\_SIZE** est important car il donne la quantité totale d’espace disponible du tas. Il est défini dans <FreeRTOSConfig.h>.

1. Quel est le rôle de la macro portTICK\_PERIOD\_MS ?

**portTICK\_PERIOD\_MS** est la macro qui gère l’intervalle de temps minimale entre chaque mise à jour du système. Sa valeur est définie à partir du registre **TICK\_RATE\_HZ**.

* 1. Sémaphores pour la synchronisation

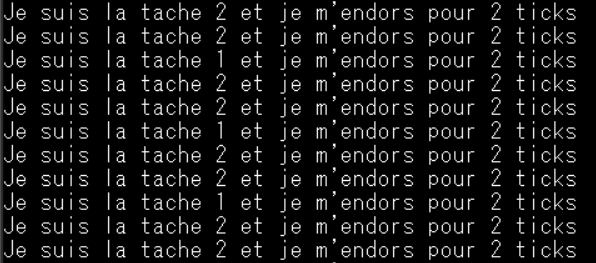


*Console UART*

1. Changez les priorités. Expliquez les changements dans l’affichage.

Le blocage ne s’effectue pas au même endroit suivant les priorités. Quand la **TaskTake** est prioritaire, elle devient active quand le sémaphore est donné, sinon elle devient active quand **TaskGive** est bloquée par le délai.

1.5 Réentrance et exclusion mutuelle



*Problème !*

1. Expliquez d’où vient le problème.

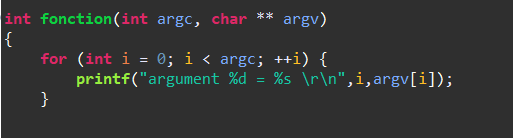
La fonction **printf()** se télescope entre la **task1** et la **task2** car la fonction est rappelée avec une plus grande priorité dans la **task2** alors qu’elle n’a pas fini d’émettre dans la **task1**

1. Proposez une solution en utilisant un sémaphore Mutex.

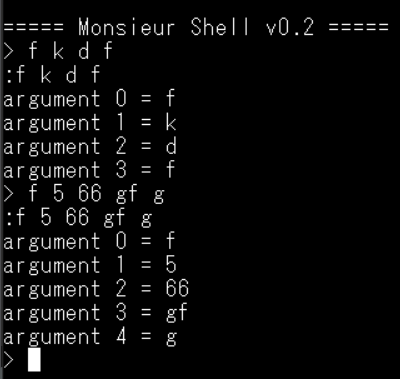
Le sémaphore Mutex va permettre à la tâche en cour de se dérouler tant qu’e celle-ci ne renvoie pas le sémaphore Mutex. Ainsi la tâche ne sera pas préemptée et la fonction **printf()** se fera en entière avant de passer la main.

1. On joue avec le Shell
   1. Terminer l’intégration du Shell commencé en TD. Quel est le problème et comment le résoudre.

Le système n’est pas capable de prendre tous les arguments en entrée et affiche des arguments vides si trop peu ont été fournis. Il faut alors lire les éléments un par un pour en connaitre la quantité. Cela nécessite une gestion de l’interruption UART lors de la réception de chaque caractère. Le démarrage du Shell et du Scheduler sont aussi en conflit, le lancement de l’un doit automatiquement permettre le lancement de l’autre.



*Solution !*



*Vue de la console UART*

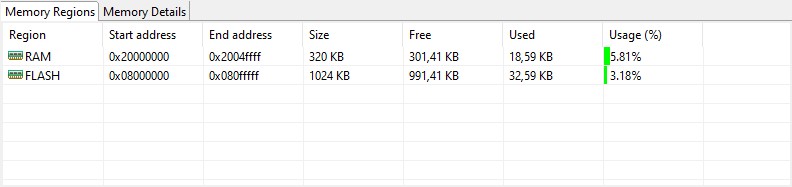
1. Debug, gestion d’erreur et statistiques
   1. Gestion du tas
2. Quel est le nom de la zone réservée à l’allocation dynamique ?

La zone dédiée à l’allocation dynamique est le tas, lui-même dans la RAM.

1. Est-ce géré par FreeRTOS ou la HAL ?

La quantité allouée au démarrage est fixée par la quantité demandée dans FreeRTOS (Total\_Heap\_Size).

1. Notez la mémoire RAM et Flash utilisée, comme dans l’exemple ci-dessous



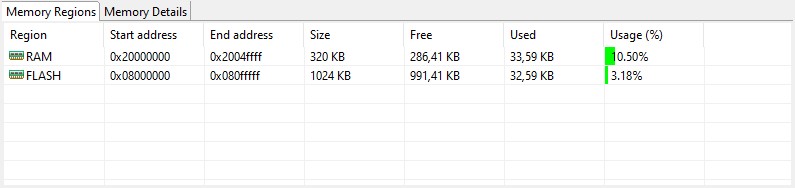
*Mémoire par défaut*

1. Notez la nouvelle utilisation mémoire.

On a pu créer 16 tâches bidons, mais la mémoire RAM n’a pas changé.

1. Notez la nouvelle utilisation mémoire. Expliquez les trois relevés.

On double la taille du tas :



*Mémoire modifiée*

La mémoire RAM à presque doublée !

La taille du tas est définie au démarrage, et alloue des emplacements vides dans la RAM. C’est-à-dire qu’ils sont existants même s’ils sont vides. Le fait de remplir le tas ne change donc rien à la quantité de RAM consommée car seule la partie déjà allouée et donc déjà contée sera écrite.