TP STR – FREERTOS –

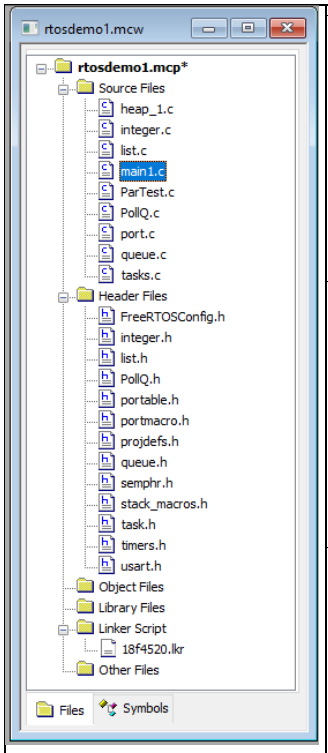
1 OBJECTIF

Mettre en pratique les différents concepts de la programmation concurrente :

- Synchronisation des tâches (sémaphores).
- La protection des ressources (Mutex).
- La communication inter-tâches (Message Queue).

2 TRAVAIL DEMANDE

Ouvrez le projet « rtosdemo1.mcp » dans le répertoire D:\FreeRtosPjt\PIC18_MPLAB\. Ouvrez ensuite la fichier « main1.c »



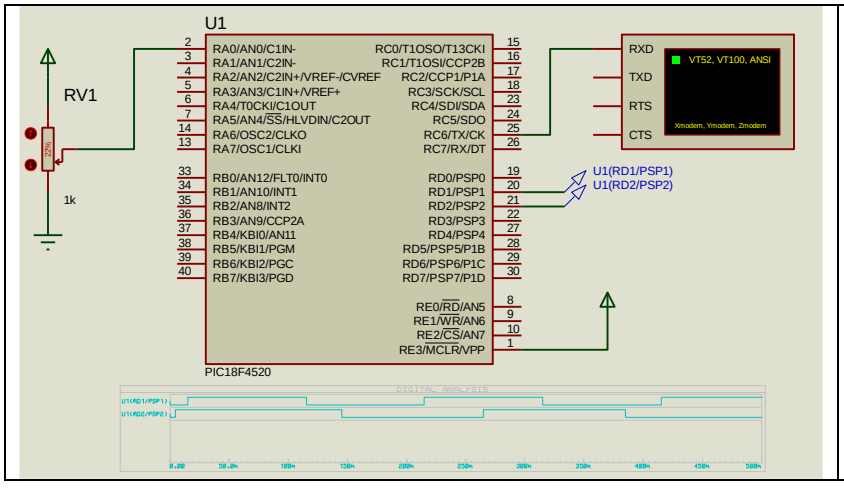

```
void Tache1 (void *p)
{  portTickType xLastWakeTime;
   xLastWakeTime = xTaskGetTickCount();

   for(;;)
   {
            LED1 = !LED1;
            putrsUSART(" -- Tache 1 Active--\n\r") ;
            vTaskDelayUntil(&xLastWakeTime,100/portTICK_RATE_MS);
   }
}
void Tache2 (void *p)
{

   for(;;)
   {
            LED2 = !LED2;
            putrsUSART(" -- Tache 2 Active--\n\r") ;
            vTaskDelay(100/portTICK_RATE_MS);
   }
}
/*-----------------------------------------------------------*/
void main( void )
{
   TRISD = 0;
   uartconfig();

   xTaskCreate( Tache1, "LED1", configMINIMAL_STACK_SIZE, NULL, 1, NULL );
   xTaskCreate( Tache2, "LED2", configMINIMAL_STACK_SIZE, NULL, 1, NULL );
   vTaskStartScheduler();
   while (1);
}
```

Le programme comporte deux tâches périodiques ; la périodicité des tâches est générée par les primitives `vTaskDelayUntil` et `vTaskDelay`. Compiler le programme et faites la simulation sous ISIS.



- PIC18F4520 - 8 Mhz
- Virtual Terminal
- pot-hg :

1. En utilisant le « Graphe Mode », mesurer la périodicité de chaque tâche. Conclure.
2. Remplacer dans la tâche2, `vTaskDelay` par `vTaskDelayUntil.` Mesurer à nouveau la période de ta tâche2 et déduire le quantum de temps « Time Slicing ».
3. Que peut-on dire des messages affichés dans la fenêtre du « Virtual Terminal »

2.1 Protection des ressources

La fonction d'affichage est une ressource partagée par les deux tâches. Faites la protection par des Mutex afin d'afficher correctement les deux messages.

```
SemaphoreHandle_t  Mutex ;
Mutex = xSemaphoreCreateMutex( void ) ;     // création du Mutex

xSemaphoreTake( SemaphoreHandle_t xSemaphore, TickType_t xTicksToWait );
xSemaphore  :  l'identificateur de la sémaphore (handle)
xTicksToWait   : temps d'attente de la disponibilité de la ressource. La valeur
portMAX_DELAY permet d'avoir une attente infinie.

xSemaphoreGive( SemaphoreHandle_t xSemaphore );
```

2.2 Synchronisation

Le programme ci-dessous, comporte deux tâches :

- une tâche Acquisition, qui convertie une grandeur analogique (simulé par un potentiomètre) en équivalent numérique (adc_val).
- Une tâche Transmission, qui envoie la valeur convertie sur le port série (fonction printf()).

1. Tapez, compilez et exécutez le programme sous ISIS.
2. Faites la synchronisation des tâches en utilisant le sémaphore binaire.

```
SemaphoreHandle_t  Sem ;
Sem = xSemaphoreCreateBinary( void );
```

Rq : le sémaphore est créé à l'état « vide », ressource indisponible.

```
/***********************************Programme *******************************/

#include <stdio.h>
#include "FreeRTOS.h"
#include "task.h"
#include "queue.h"
#include "semphr.h"
#include "usart.h"
#include <stdlib.h>

SemaphoreHandle_t xSemaphore = NULL;
unsigned char adc_val ;
/*****************************************************/
void ADC_config(void)
{
   ADCON0 = 0x01;
   ADCON1 = 0x0E;
   ADCON2 = 0x11;
}
char Read_ADC(void)
{
      ADCON0bits.GO = 1;
      while(ADCON0bits.GO == 1);
      return(ADRESH);
}
```

```
void uartconfig(void)
{
   TRISCbits.RC6 = 0;        //TX pin set as output
   TRISCbits.RC7 = 1;        //RX pin set as input
   SPBRG = 12;              //Writing SPBRG Register
   TXSTA = 0b00100000;
   RCSTA = 0b10010000;
}
//*****************************************************************************************
void Acquisition (void *p)
{  portTickType xLastWakeTime;
   xLastWakeTime = xTaskGetTickCount();
   for(;;)
   {
          adc_val = Read_ADC();
          putrsUSART(" -- Tache 1 Active--\n\r") ;

          vTaskDelayUntil(&xLastWakeTime,800/portTICK_RATE_MS) ;
   }
}
void Transmission1 (void *p)
{
   for(;;)
   {
          printf("adc_val = %u\n\r",adc_val) ;
   }
}
/*-----------------------------------------------------------------------------------------*/
void main( void )
{
   ADC_config();
   uartconfig();
   xTaskCreate( Acquisition, "Acq", configMINIMAL_STACK_SIZE, NULL, 1, NULL );
   xTaskCreate( Transmission, "Trans", configMINIMAL_STACK_SIZE, NULL, 1, NULL );
   vTaskStartScheduler();
   while (1);
}
```

2.3 Les Boîtes aux lettres (Message Queue)

La tâche Acquisition va maintenant envoyer dans une ***boîte aux lettres***, la valeur convertie à la tâche Transmission. Apportez les modifications nécessaires au programme précédent afin d'implémenter le concept de communication inter-tâches en utilisant le « message queue ».

```
/*************************************************************************************************************************/
QueueHandle_t   Queue ;
QueueHandle_t  xQueueCreate(UBaseType_t uxQueueLength,  UBaseType_t uxItemSize);
uxQueueLength : nombre maximum d'élément contenu dans la queue
uxItemSize    : nombre d'octets pour chaque élément.

BaseType_t  xQueueSend( QueueHandle_t  xQueue,    const void *  pvItemToQueue,
TickType_t xTicksToWait  );
xQueue : l'indicateur retourné par la fonction xQueueCreate
pvItemToQueue : pointeur sur l'élément placé dans la file d'attente
xTicksToWait : nombre de Ticks maximal pendant lequel la tâche reste bloquer en
attendant un espace disponible dans la file d'attente.

BaseType_t  xQueueReceive(QueueHandle_t  xQueue,     void  *pvBuffer,TickType_t
xTicksToWait );
xQueue :  l'indicateur retourné par la fonction xQueueCreate
pvBuffer : pointeur sur le buffer dans lequel sera copié l'élément reçu.
xTicksToWait : nombre de Ticks maximal pendant lequel la tâche reste bloquer en
attendant la présence d'un élément dans la file d'attente. Si xTicksToWait = 0,
la fonction retourne immédiatement si la file est vide. La valeur portMAX_DELAY
permet une attente infinie.

UBaseType_t uxQueueMessagesWaiting( QueueHandle_t xQueue ) : retourne le nombre
d'éléments stockés dans la file d'attente.
```