# TP1 –Conception des systèmes temps réels

## I. Objectif :

- Comprendre et introduire la notion du noyau temps réel OSA.
- Concevoir et tester une application temps réel.

## II. Notion de noyau temps réel OSA

Une application est dite temps réelle si elle a la capacité de traiter en un temps déterminé un flux d'évènements issus d'un processus, sans perdre un seul de ces évènements.

Une application temps réel multitâche monoprocesseur définit qu'un seul processeur est capable d'exécuter plusieurs tâches en même temps. Il s'occupe de la coordination et de cohabitation de ces différents élements.

OSA est un système d'exploitation coopératif multitâche en temps réel ce type de système appelé (RTOS : REAL TIME OPERATING SYSTEM), OSA est dédié pour les microcontrôleurs PIC10, PIC12, PIC16, PIC18, PIC24, dsPIC, pour Atmel AVR contrôleurs 8 bits, et pour STMicroelectronics STM8.

OSA permet au programmeur de se concentrer sur des tâches axées sur les problèmes (algorithmique, mathématiques, etc.) et ne pas avoir à se soucier des tâches secondaires. Toutes les tâches secondaires sont effectuées par le noyau de l'OSA.

Une tâche dans OSA est une fonction C. Cette fonction doit contenir une boucle infinie qui présente à l'intérieur au moins un service qui commute le contexte de la tâche. Une tâche peut être créée comme suit :

```
void SimpleTask
(void)
                // Infinite
      OS_Yield(); // Unconditional
   }
}
```

Chaque tâche a une priorité. Il existe huit niveaux de priorité de 0 (le plus élevé) à 7 (le plus bas). La priorité de la tâche peut être modifiée lors de l'exécution.

Dans cet exemple, nous avons créé une tâche à exécuter la fonction Task1() avec la priorité la plus basse. Une fois le service est terminé, le système d'exploitation reconnait que la fonction Task1() est une tâche créée et la donner le contrôle lorsqu'elle est prête.

```
#include <osa.h>

void Task1 (void)

{

    for (;;)

    {
        OS_Yield();
    }
}

void main (void)

{

    OS_Init(); OS_Task_Create(7,Task1);

    ...

    for (;;) OS_Sched();

}
```

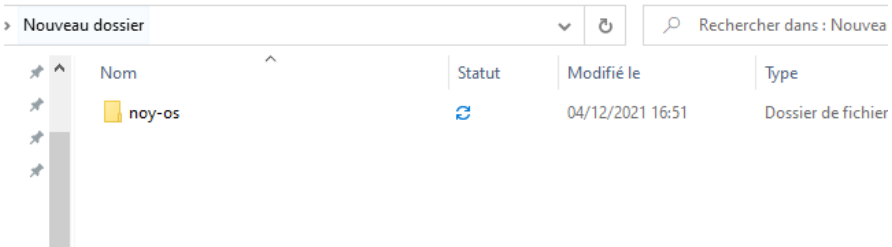
## III. Travail demandé :

Nous allons concevoir dans cette partie une application multitâche à base du noyau OSA sur un microcontrôleur 18F4550. Le compilateur utilisé sera MikroC Pro pour PIC.

**1)** Suivez les étapes suivantes pour réaliser une application temps réel.

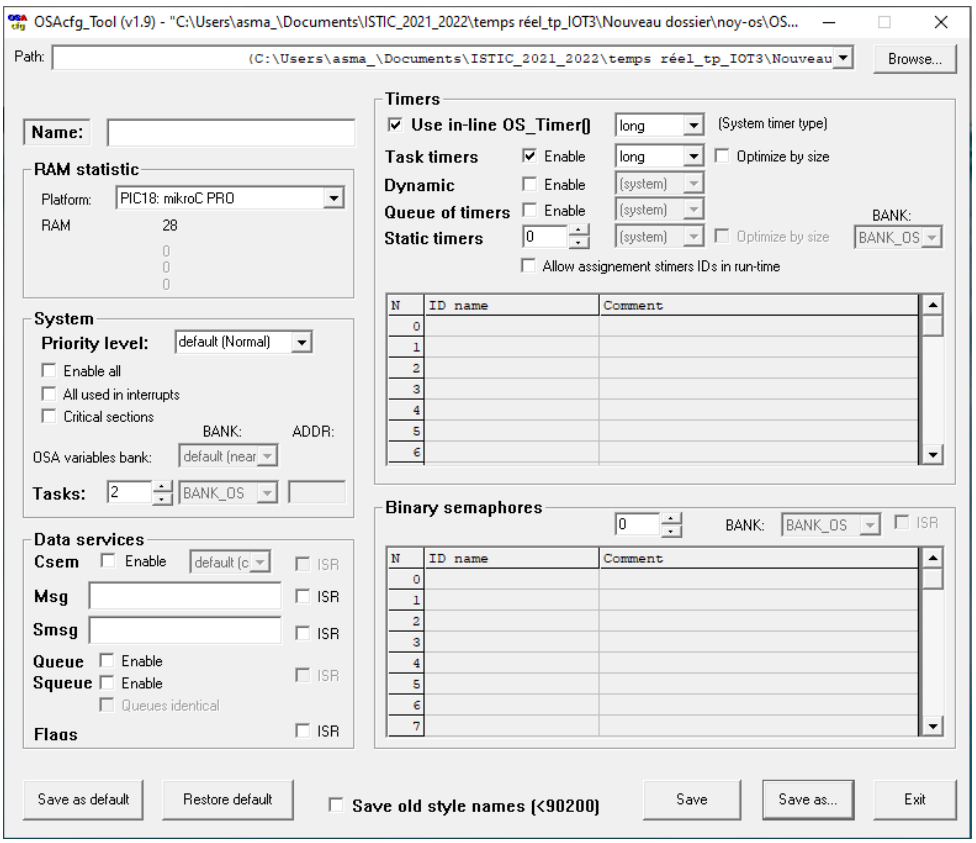
**Etape 1 :** création d'un répertoire de projet OSA

- Créer un nouveau dossier sur le bureau et le renommer sous le nom de votre projet.
- Importer le dossier noyau_OSA dans ce nouveau dossier




- Lancer OSA Configuration OSAcfg_Tool afin de saisir les paramètres de configuration de votre application OSA (définir le nombre des tâches nécessaires, les sémaphores, gérer les interruptions...)

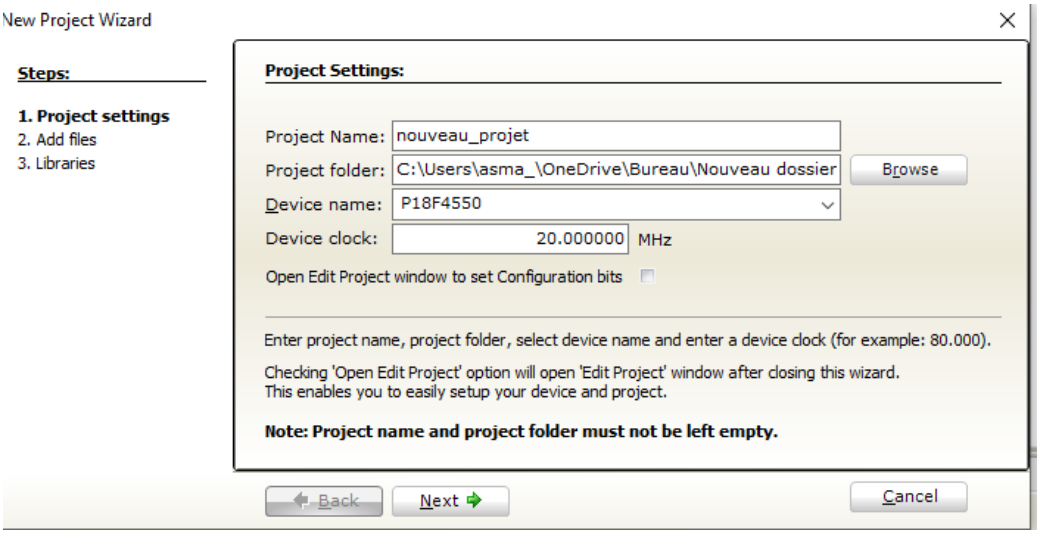
Exemple : La configuration ci-après permet de saisir 2 tâches.



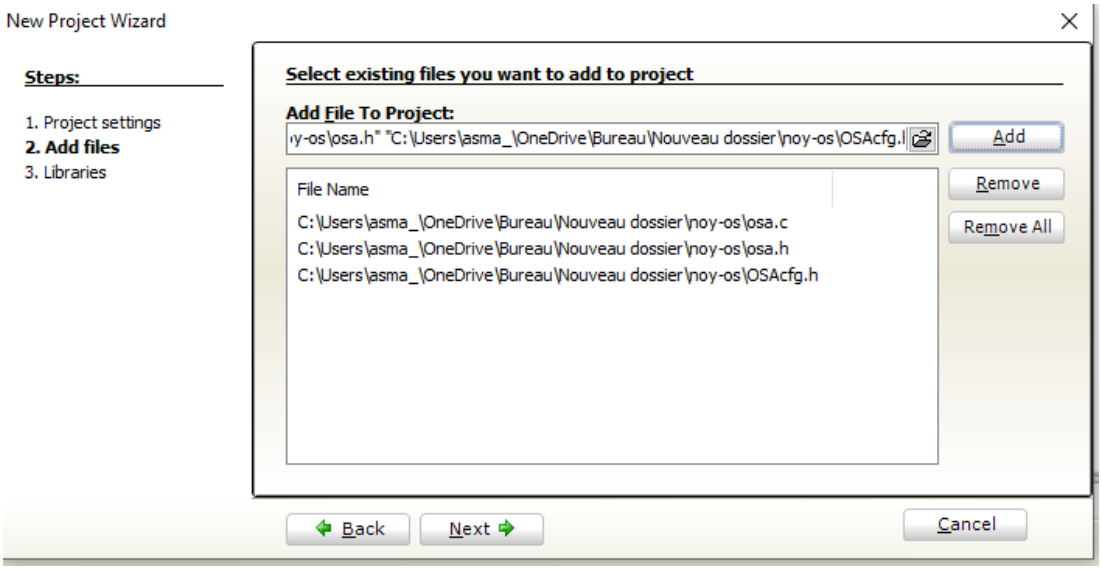

- Sauvegarder le fichier sous le dossier noyau osa

**Etape 2 :** création d'une application temps réel

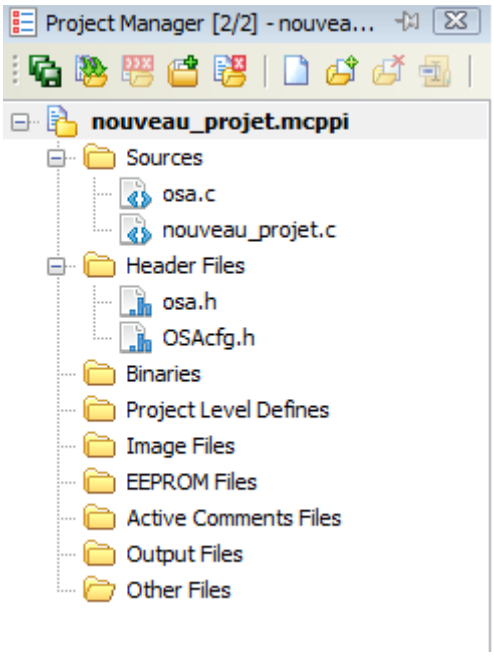
- Lancer MikroC et créer un nouveau projet



- Ajouter les fichiers osa.c, osa.h, oscfg.h




- Vérifier les fichiers ajoutés dans l'arborescence du projet




**Etape 3 :** saisir le code de l'application temps réel

```
#include <OSA.h>
#include <OSAcfg.h>
void InitTimer0(){
  T0CON       = 0x88;
  TMR0H       = 0xD1;
```

```
  TMR0L        = 0x21;
  GIE_bit      = 1;
  TMR0IE_bit   = 1;
}

void Interrupt(){
  if (TMR0IF_bit){
    TMR0IF_bit   = 0;
    TMR0H   = 0xD1;
    TMR0L   = 0x21;

    OS_Timer();      //Initialisation du timer pour le comptage des ticks
  }
}

void Thread1(void)
{
     while(1)
     {
// saisir le code

     }
}

void main() {
// Configuration du PORTD comme sortie
//mise à zéro du PORTD

    OS_Init();    //Initialisation de OS services
  OS_Task_Create(0,Thread1);

  InitTimer0();
```

```
    OS_Run();                    // Appel à l'ordonnanceur
}
```

2) Ajouter une ligne de code à la première tâche **Thread 1** pour changer l'état du bit 1 du PORTD.
3) Créer trois tâches **Thread 2, Thread 3** et **Thread 4** identiques au Thread 1 en changeant respectivement l'état des bits 2, 3, 4 du PORTD.
4) Compléter la fonction **main** en ajoutant la ligne de code qui permet de configurer le PORTD comme étant sortie et son initialisation.
5) Compiler et tester votre projet avec le schéma ISIS suivant.