Module Complémentaire

Systèmes multitâches, systèmes temps réel

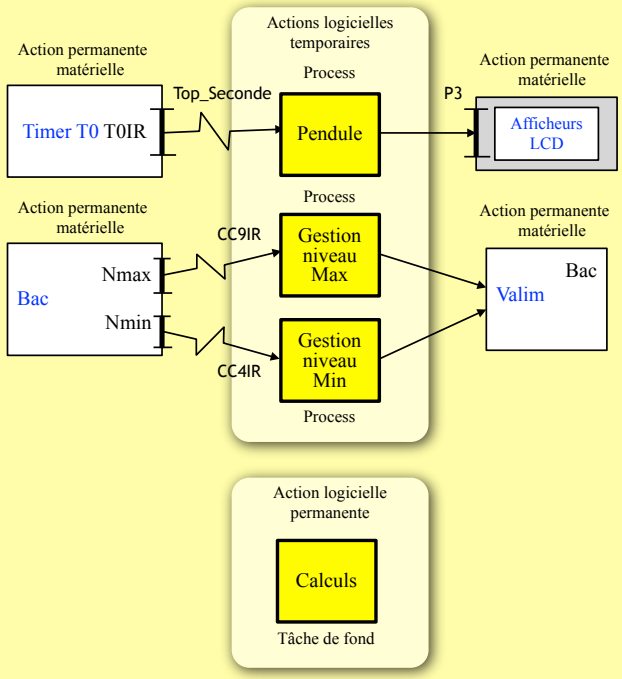
MC II2

Yvon TRINQUET

Institut Universitaire de Technologie de Nantes - Département G.E.I.I.

Chapitre 1 – Généralités

➡ **Exemple** : gestion d'un bac et affichage de l'heure (gestion par interruptions de programme)



Système = ensemble d'actions

- **Actions matérielles** (timer, le bac, l'afficheur LCD)
- **Actions logicielles** :
  - ➡ **permanente** : la tâche de fond
  - ➡ **temporaires** : les process
- Les actions communiquent via les **variables partagées** (variable en mémoire, port d'E/S …)
- Les **interruptions matérielles** sont le moyen de provoquer l'exécution d'une procédure (process)

➡ **Modèle d'exécution** d'un tel système (cas du C167, vu en 1e année)

- La tâche de fond s'exécute en permanence
- Elle est interrompue (suspendue) chaque fois qu'une demande d'interruption est prise en compte :
  - ➡ toutes les secondes, par l'IT engendrée par T0IR (Timer 0 Interrupt Request)
  - ➡ chaque fois que le niveau dépasse le niveau maximum (détecteur de transition sur CC9IO, qui engendre CC9IR)
  - ➡ chaque fois que le niveau devient inférieur au niveau minimum (détecteur de transition sur CC4IO, qui engendre CC4IR)
- le process (procédure « dynamique ») est le relais de l'interruption : il a une activité correspondant à la requête
- Si plusieurs requêtes d'interruptions sont présentes en même temps c'est le système de priorité matérielle qui gère cela (chaque IT a une priorité différente) : un seul process est actif à un instant donné
- Attention aux conflits d'accès sur les variables partagées !

➡ Structure logicielle d'un tel système (cas du C167)

```
void main (void)
{
   T0IC=0x10;              // Timer 0 sur le niveau 4
   T0IE=1;                 // Timer 0 Interrupt Enable
   CC4IE=1 ;               // Capture canal 4 Enable
   CC9IE=1 ;// Capture canal 9 Enable
   IEN=1;                  // Global Interrupt Enable

   while (1)   Calculs() ;

}
```

- la validation des interruptions est ici très détaillée,
- les validations des masques locaux T0IE, CC4IE, CC9IE (IE Interrupt Enable) auraient pu être faits avec l'initialisation des registres de contrôle des interruptions : T0IC, CC4IC, CC9IC (IC Interrupt Control). Le détail est traité dans le chapitre 5.

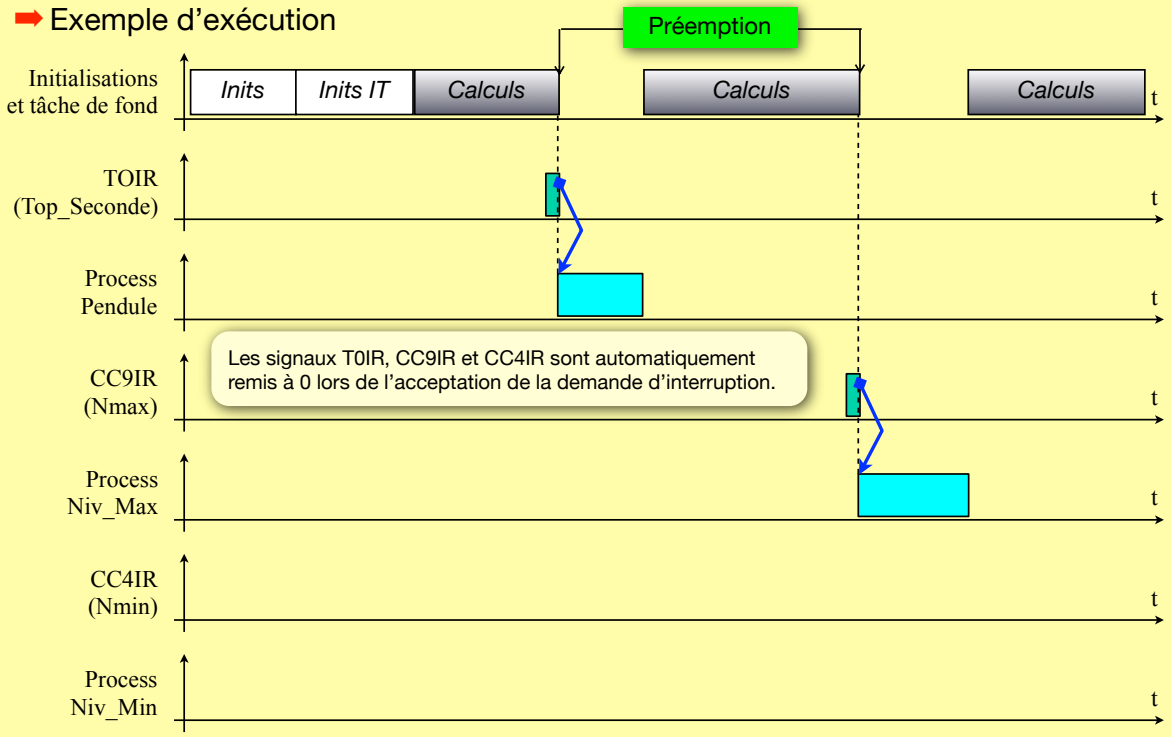
Le code de la tâche de fond

Initialisations des interruptions

```
void Init_ES (void)
{
   - - -
}

void Init_Timer (void)
{
   - - -
}

void Pendule (void) interrupt 32
{
   - - -
}

void Niv_Max (void) interrupt 25
{
   - - -
}

void Niv_Min (void) interrupt 20
{
   - - -
}
```

Les interruptions attachées aux process

➡ Exemple d'exécution



Modèle d'exécution précédent = modèle basique de systèmes multitâches

- une « activité » de fond (la tâche de fond)
- des procédures activées par interruption

Nouveau modèle :

- la « **tâche** » : activité logicielle indépendante ou non des autres tâches du système
- un support logiciel d'exécution pour la gestion des tâches :

  le **système d'exploitation** (ou **OS = Operating System**)

On conçoit une application sous forme de tâches coopérantes, à exécution pseudo-parallèle, et on l'implémente comme cela.

Le rôle du système d'exploitation (basique dans notre cas) est d'offrir des services aux tâches :

- services permettant la **coopération entre tâches** et **tâches/environnement**
- gérer le pseudo-parallélisme d'exécution **des tâches en gérant le processeur**

Coopération entre tâches = un ensemble de services permettant

➡ **l'activation** entre tâches :
une tâche peut en activer une autre

➡ **la synchronisation** entre tâches (notion d'événement) :
2 ou plusieurs tâches peuvent se synchroniser via un mécanisme

➡ **la communication** entre tâches (notion de message) :
2 ou plusieurs tâches peuvent s'échanger des messages

➡ **le partage de données** communes en exclusion mutuelle d'accès :
2 ou plusieurs tâches peuvent accéder en exclusion mutuelle à des données, via un protocole

Gestion du processeur par le système d'exploitation =

- allocation du processeur (ressource unique) à une tâche parmi celles qui le demandent

  ↳ **besoin d'un critère de choix** pour décider

- exemples de critères :
  - temps partagé (« time-sharing ») équitablement
  - critère multi-aspects (temps alloué, date de la demande …)
  - **critère d'importance ou de priorité**
- allouer le processeur à une tâche en fonction d'un critère

  = **ordonnancer** les tâches du système (de l'application)
- ce choix est fait par un module spécifique de l'OS :

  l'ordonnanceur qui exécute un algorithme d'ordonnancement

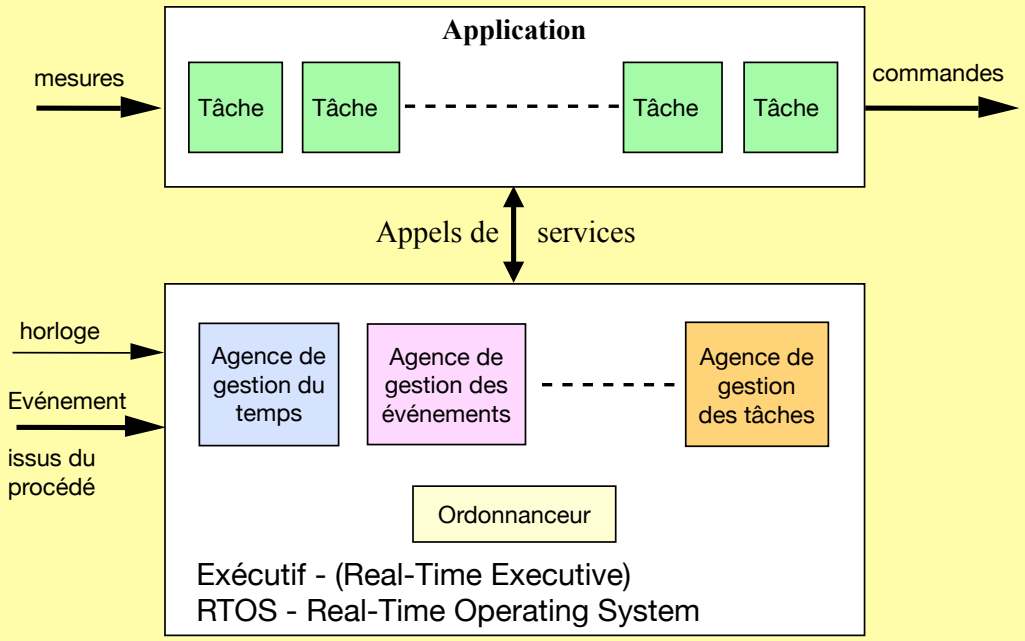
Système d'exploitation temps réel

= **RTOS** (Real-Time OS) = **Exécutif Temps Réel**

- « Temps Réel » : on se réfère au **temps physique**
- le système de commande doit réagir en tenant compte des **temps de réponse** imposés par le procédé contrôlé :
  ex. : dès que le niveau dépasse Nmax je dois fermer la vanne avant un certain temps
- ces contraintes (deadline = échéance temporelle) sont associées aux tâches en leur donnant une **priorité** (valeur entière*)
- la priorité d'une tâche est utilisée par l'ordonnanceur pour décider de l'allocation du processeur, lorsque plusieurs tâches sont en compétition pour s'exécuter
- si le système est bien conçu, toutes les tâches doivent respecter leurs échéances (analyse d'ordonnançabilité)

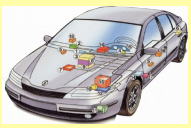
* La priorité peut être **statique** (le cas usuel, que nous traiterons) ou dynamique

➡ L'exécutif vu comme une machine virtuelle



Classification sommaire des RTOS

- exécutifs « généralistes » : VxWorks, OSE …
  - ➡ applications générales civiles et militaires
- exécutifs Unix Temps Réel (POSIX) ou Linux (RTAI, RT-Linux)
  - ➡ évolution des applications du monde Unix vers le temps réel
- le **standard OSEK/VDX**
  - ➡ étudié pour les systèmes embarqués dans l'automobile

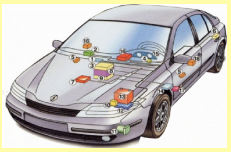


OSEK/VDX est une proposition de standard, pas un produit !

- ➡ spécification ISO (produits commerciaux ou Open Source**)

Nous utiliserons « Trampoline » l'implémentation open source de l'IRCCyN*

* IRCCyN : Institut de Recherche en Communications et Cybernétique de Nantes, UMR CNRS 6597, Équipe « Systèmes Temps Réel » (** http://Trampoline.rts-software.org)



OSEK/VDX en quelques mots

Son contexte :

- celui de l'électronique embarquée dans les véhicules avec des contraintes temps réel dures et souples (Powertrain, Chassis, Body, Telematics)
- des « petits » calculateurs (peu de Ram, 16 bits …)
- des systèmes en réseau (CAN, LIN, FlexRay …)
- une sûreté de fonctionnement élevée

Les travaux :

- ceux d'un consortium européen (constructeurs, équipementiers, universitaires), depuis les années 1990
- OSEK/VDX : Offene Systeme und deren Schnittstellen für dir Elektronik im Kraftfahrzeug / Vehicle Distributed eXecutive
- fusion des travaux des projets OSEK (Allemand) et VDX (Français)

➡ OSEK/VDX : état actuel

OSEK/VDX OS	version 2.2.3	février 2005
	noyau du système d'exploitation	
OSEK/VDX COM	version 3.0.3	juillet 2004
	services pour la communication	
OSEK/VDX NM	version 2.5.3	juillet 2004
	gestion et surveillance du réseau	
OSEK/VDX OIL	version 2.5	juillet 2004
	langage de description des applications	

Le site Web de référence : **http://www.osek-vdx.org**

Ces parties d 'OSEK/VDX sont également un **standard ISO** : ISO 17356

Principaux services d'OSEK/VDX OS

- ⟶ - Services pour les **tâches**
- ⟶ - Services de synchronisation (**événements**)
- ⟶ - Services d **'exclusion mutuelle**
- ⟶ - Services pour les phénomènes récurrents (**compteurs et alarmes**)
- - Service pour la **communication** (dans OSEK/VDX COM)
- - Services pour la gestion des **interruptions**
- - Services **systèmes** et gestion des **erreurs**

Tous les objets sont statiques :

- pas de création dynamique d'objets
- pas de suppression dynamique d'objets

Fin de chapitre