



## MC4 - Programmation Multicoeur avec OSEK/VDX et AutoSAR

*Eviter les pièges de la programmation temps réel multicoeur*

### Objectifs

- Découvrez les notions de multitâche temps réel
- Maîtriser la programmation concurrente
  - Sur le même processeur
  - Sur un système multiprocesseur
- Comprendre les interactions avec l'architecture du processeur
  - Cache
  - Pipeline
  - Optimisations
  - Multi-core et Hyperthreading
- Comprendre la structure d'un noyau temps réel
  - OSEK/VDX
  - Autosar multicoeur

Ce cours apprend à maîtriser la programmation temps réel et multi-tâches en comprenant comment résoudre efficacement les problèmes rencontrés en fonction des primitives disponibles sur l'OS utilisé.

### Matériel

- Un PC Linux par binôme
- Une carte cible sous Linux
- Compilateur et débogueur croisés
- Support de cours imprimé
- Présentation et solutions des exercices

### Pré-requis

- Bonne connaissance de la programmation C, si possible en environnement embarqué
- Compréhension de base de l'architecture des processeurs

### Démarche pédagogique

- Les exercices s'attachent à mettre en œuvre les mécanismes disponibles pour résoudre des problèmes classiques: Readers-writers, producteurs-consommateurs, le repas des philosophes...
- Chaque exercice comprend une explication détaillée et un schéma, ce qui aide à comprendre le fonctionnement des algorithmes.
- Pour chaque exercice il est fourni un code quasiment complet avec des parties à compléter, ce qui permet, après une phase de compréhension du code fourni, de maîtriser des fonctionnalités qui habituellement prennent des heures à concevoir.
- Le cours comprend des exercices facultatifs destinés à approfondir la compréhension des sujets traités.

### Environnement du cours

- Cours théorique
  - Support de cours imprimé et au format PDF (en anglais).
  - Le formateur répond aux questions des stagiaires en direct pendant la formation et fournit une assistance technique et pédagogique.
- Activités pratiques
  - Les activités pratiques représentent de 40% à 50% de la durée du cours.
  - Elles permettent de valider ou compléter les connaissances acquises pendant le cours théorique.

- Exemples de code, exercices et solutions
- Un PC (Linux ou Windows) par binôme de stagiaires (si plus de 6 stagiaires) pour les activités pratiques avec, si approprié, une carte cible embarquée.
- Le formateur accède aux PC des stagiaires pour l'assistance technique et pédagogique.
- Une machine virtuelle préconfigurée téléchargeable pour refaire les activités pratiques après le cours
- Au début de chaque demi-journée une période est réservée à une interaction avec les stagiaires pour s'assurer que le cours répond à leurs attentes et l'adapter si nécessaire

## Audience visée

- Tout ingénieur ou technicien en systèmes embarqués possédant les prérequis ci-dessus.

# Plan du cours

## Premier jour

### Tâches et ordonnancement des systèmes embarqués

- Tâche et descripteurs de tâches
- Changement de contexte

**Exercice** : Écriture d'une routine de changement de contexte

- Mécanismes d'ordonnancement et de préemption
  - Ordonnancement avec ou sans tic d'horloge
- Modes d'ordonnancement et preuves d'ordonnancement
  - Ordonnancement à priorités fixes
  - Ordonnancement RMA et EDF
  - Ordonnements adaptatifs

**Exercice** : Écriture d'un ordonnanceur simple, à priorités fixes

### Gestion d'interruptions dans les systèmes temps-réel

- Besoin d'interruptions dans un système temps réel
  - Interruptions de timer
  - Interruptions de périphériques
- Notion d'interruption sur niveau ou sur front
- Acquiescement matériel et logiciel
- Vectorisation des interruptions

**Exercice** : Écriture d'un gestionnaire d'interruption simple

- Interruption et ordonnancement

**Exercice** : Extension de l'ordonnanceur pour supporter un ordonnancement en ronde (round-robin)

### Interactions entre cœurs en multicoeurs

- Cohérence de cache
  - Snooping
  - Snoop Control Unit: transferts cache à cache
  - Machine d'état MOESI
- Ordonnancement et cohérence mémoire
  - Ordre des accès
  - Ordonnancement mémoire
  - Barrières mémoire
  - Cohérence de données et DMA
- Accès aux données et multicoeurs
  - Instructions Read-Modify-Write
  - Linked-Read/Conditional-Write

- Synchronisation en multicoeurs
  - Spinlocks
  - Interruptions inter processeurs

**Exercice :** Écriture d'un spinlock

## Second day

### Ordonnancement multicoeurs

- Ordonnancement multicoeurs
  - Affectation d'interruptions à un coeur
  - Ordonnanceur multicoeurs
- Optimisations multicoeurs
  - Optimisation de l'utilisation des caches
  - Éviter les "faux" partages
  - Éviter la saturation des caches

**Exercice :** Étude d'un ordonnanceur multicoeurs

### Les primitives de synchronisation

- Mise en attente et réveil des tâches
- Sémaphores

**Exercice :** Mise en œuvre des sémaphores par interaction directe avec l'ordonnanceur

- Exclusion mutuelle
  - Spinlocks et masquage d'interruptions
  - Mutex ou sémaphores

**Exercice :** Mise en œuvre du mécanisme de mutex

- Mutex récursifs et non récursifs

**Exercice :** Vérifier la bonne imbrication des mutex récursifs et l'utilisation de mutex non récursifs

- Le problème de l'inversion de priorité
- L'héritage de priorité (le mécanisme automagique)
- Le plafond de priorité (la réponse centrée sur la conception)

**Exercice :** Mise en place du mécanisme de plafond de priorité

- Mutexes et variables condition

**Exercice :** Ajout du support des variables condition aux mutex

- Les boites aux lettres

## Third day

### Solutions aux problèmes de parallélisme

- Les divers problèmes de la programmation parallèle
  - Accès concurrent non maîtrisé

**Exercice :** Le problème "producteur-consommateur", ou une illustration d'accès concurrents et sa solution

- Deadlocks (étreinte fatale)
- Livelocks (blocage vivant)
- Starvation (famine)

**Exercice :** Le problème du "dîner des philosophes", illustration des risques de deadlocks, livelocks et de famine

### Architecture multi-tâches de Osek/VDX

- Gestion des tâches
  - Tâches de base
  - Tâches étendues
  - Politiques d'ordonnancement
  - Activation et terminaison des tâches
- Gestion des interruptions

- Evènements de synchronisation
- Ressources

## **Autosar et la programmation Multicoeur**

- L'architecture multicoeur d'Autosar
- Notion d'entités localisables
- Améliorations de l'ordonnancement OSEK/VDX
- Spinlocks Autosar
- Le communicateur inter OS-Application
- Migrer une applications Autosar vers une plateforme multicoeur