

# Une approche synchrone à la conception de systèmes embarqués temps réel

Dumitru Potop-Butucaru

dumitru.potop@inria.fr

cours EPITA, 2022, 5<sup>ème</sup> séance

# Contenu

- ARINC 653 (2<sup>ème</sup> partie)
  - RPi653 – une implémentation de ARINC 653
- Préparation du TP

# Implantation des systèmes embarqués

Graphes  
flot de données  
déterministes

Spécification fonctionnelle

Spécification non-fonctionnelle

**Mapping**  
Dans l'espace (allocation)  
Dans le temps (ordonnancement)  
Calculs et communications

Correction  
fonctionnelle

Implantation  
exécutable

Exigences

- Mentionnées avant

Architecture-cible (**fixée**)

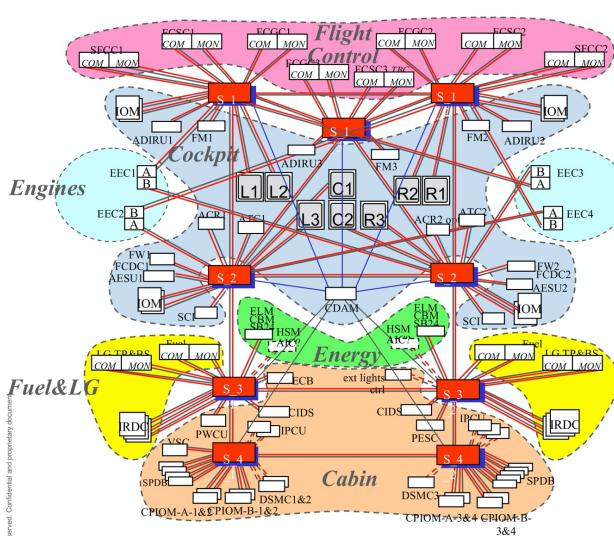
- Processeurs (CPUs, accélérateurs, etc.)
- Interconnect (bus, DMAs, NoCs, etc.)
- Stockage (RAM)
- Système d'exploitation**
- Hypothèses sur l'utilisation de la plate-forme**

Satisfaction des  
exigences

# Systèmes avioniques

- Les systèmes embarqués avioniques :
  - Complexes, répartis, temps réel, soumis à la certification

# Systèmes avioniques



## AFDX Network:

- 100 Mbits
- Redundant Network (A&B) with independent alimentation
- AFDX switches = 2 x 8
- NB of ports (connections) possible on each switch (20-24)
- MTBF of the switch is very high (100 000 hours expected)
- Up to 80 AFDX subscriber

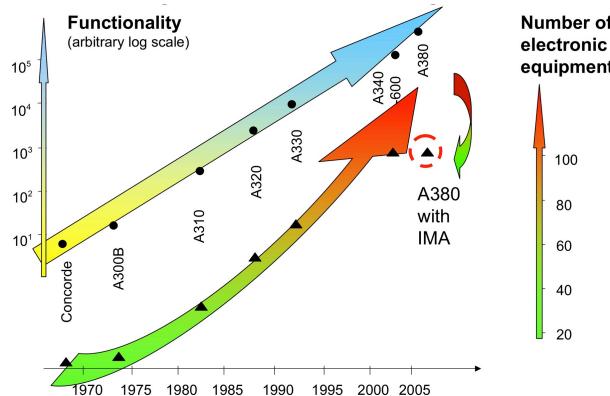
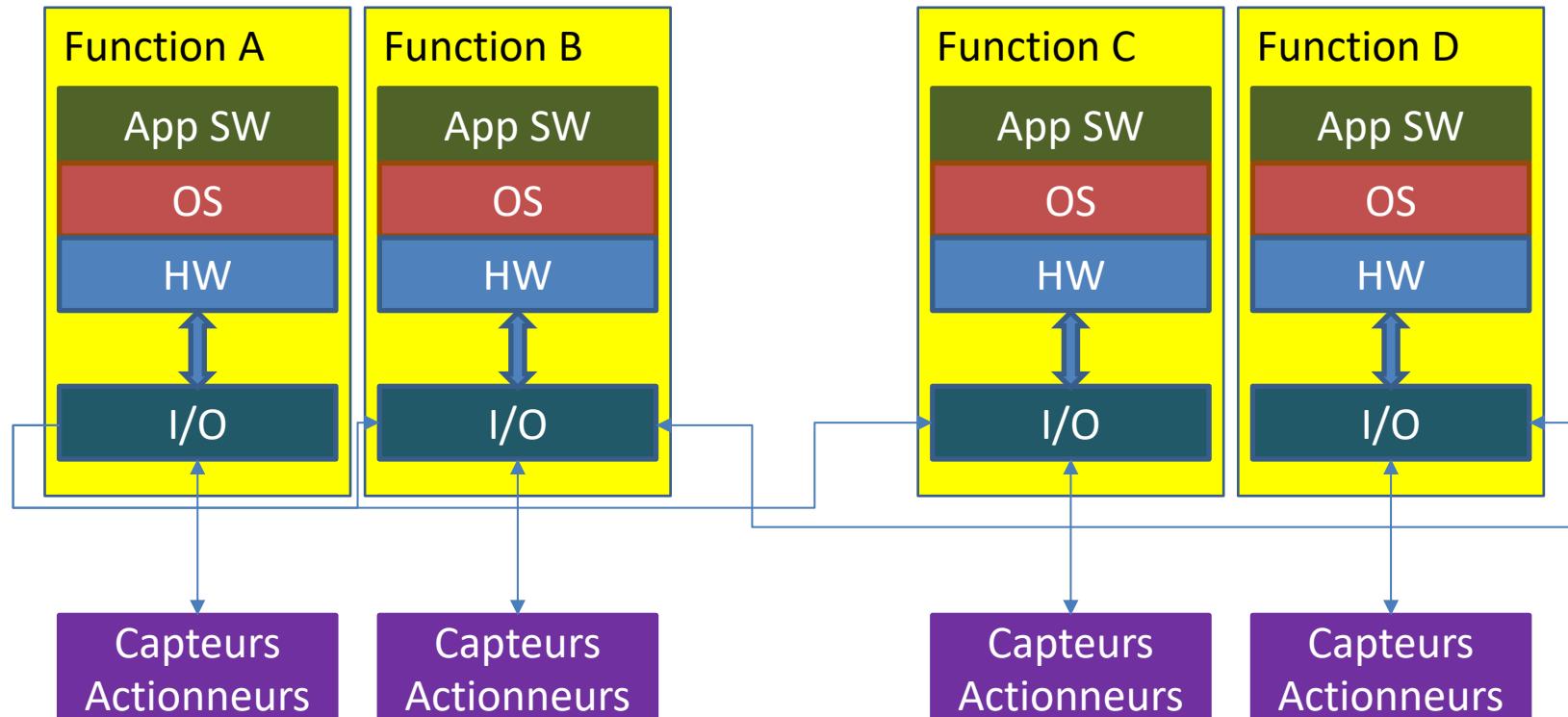


Image Copyright © balleba

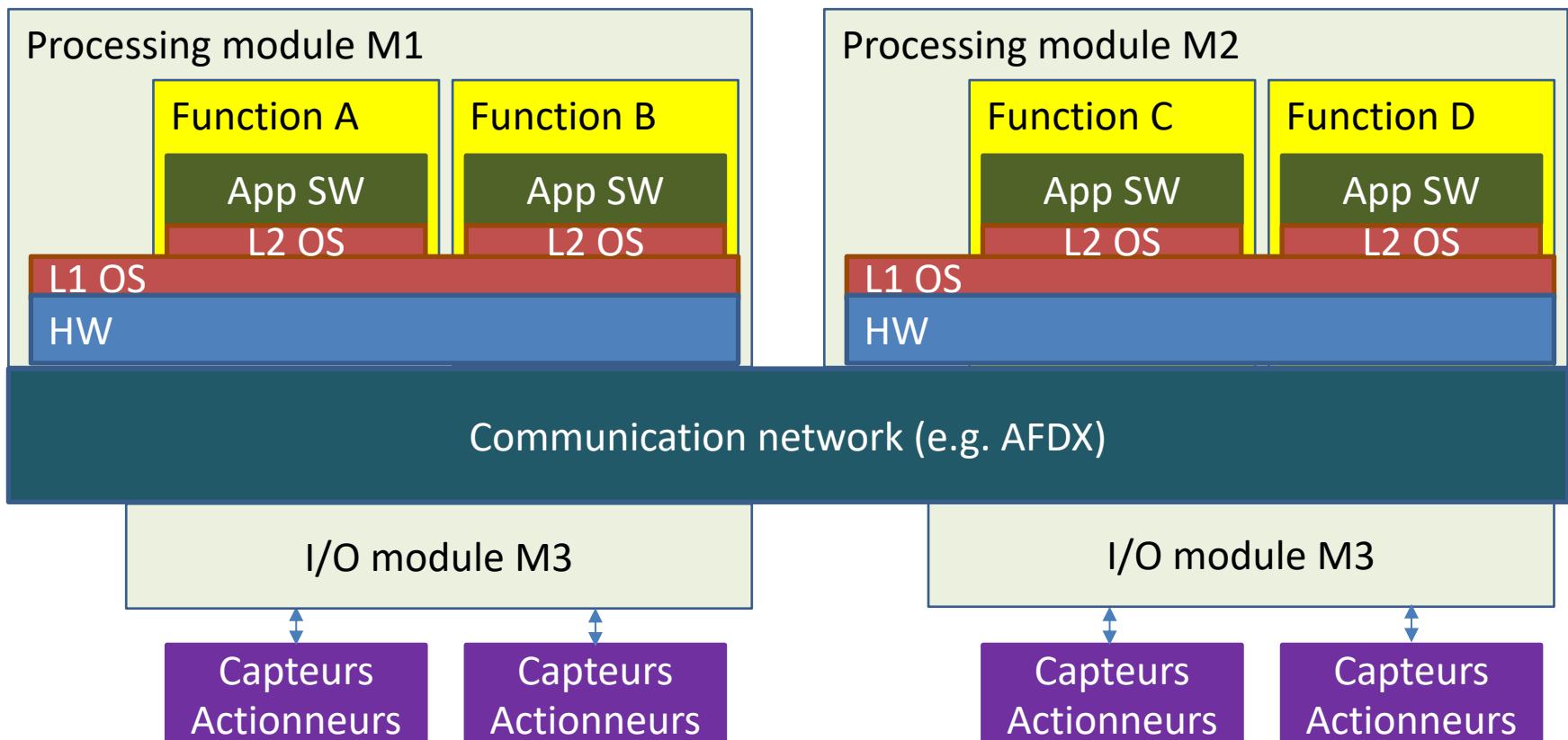
# Federated vs. IMA avionics

- Federated avionics



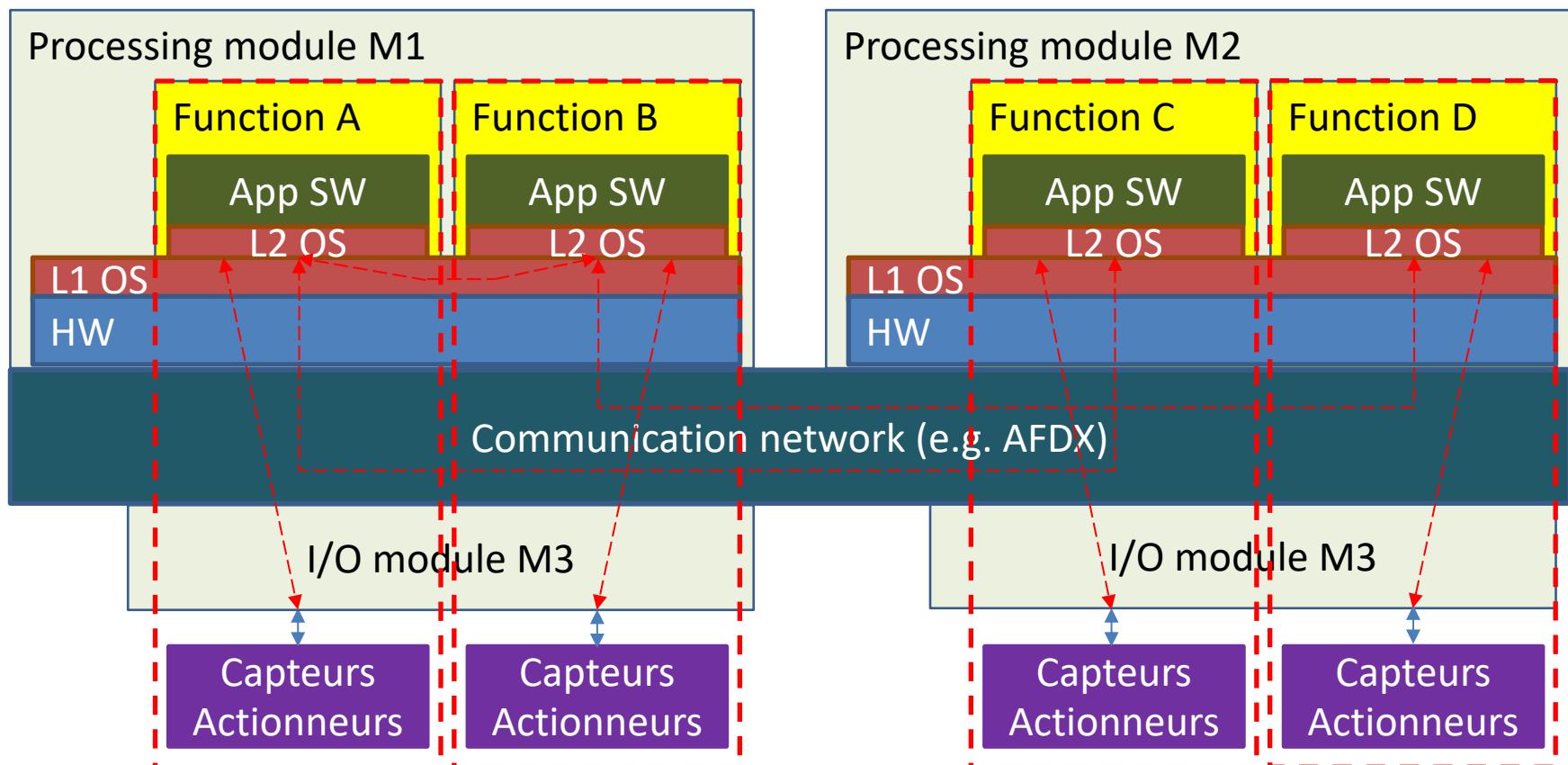
# Federated vs. IMA avionics

- IMA avionics



# Federated vs. IMA avionics

- IMA avionics – isolation spatiale et temporelle



# Concepts fondamentaux

- Organisation à 4 niveaux
  - Système IMA
  - Module = calculateur avionique IMA
  - Partition = application
  - Process = tâche séquentielle
- Transversal : Health monitoring & error recovery

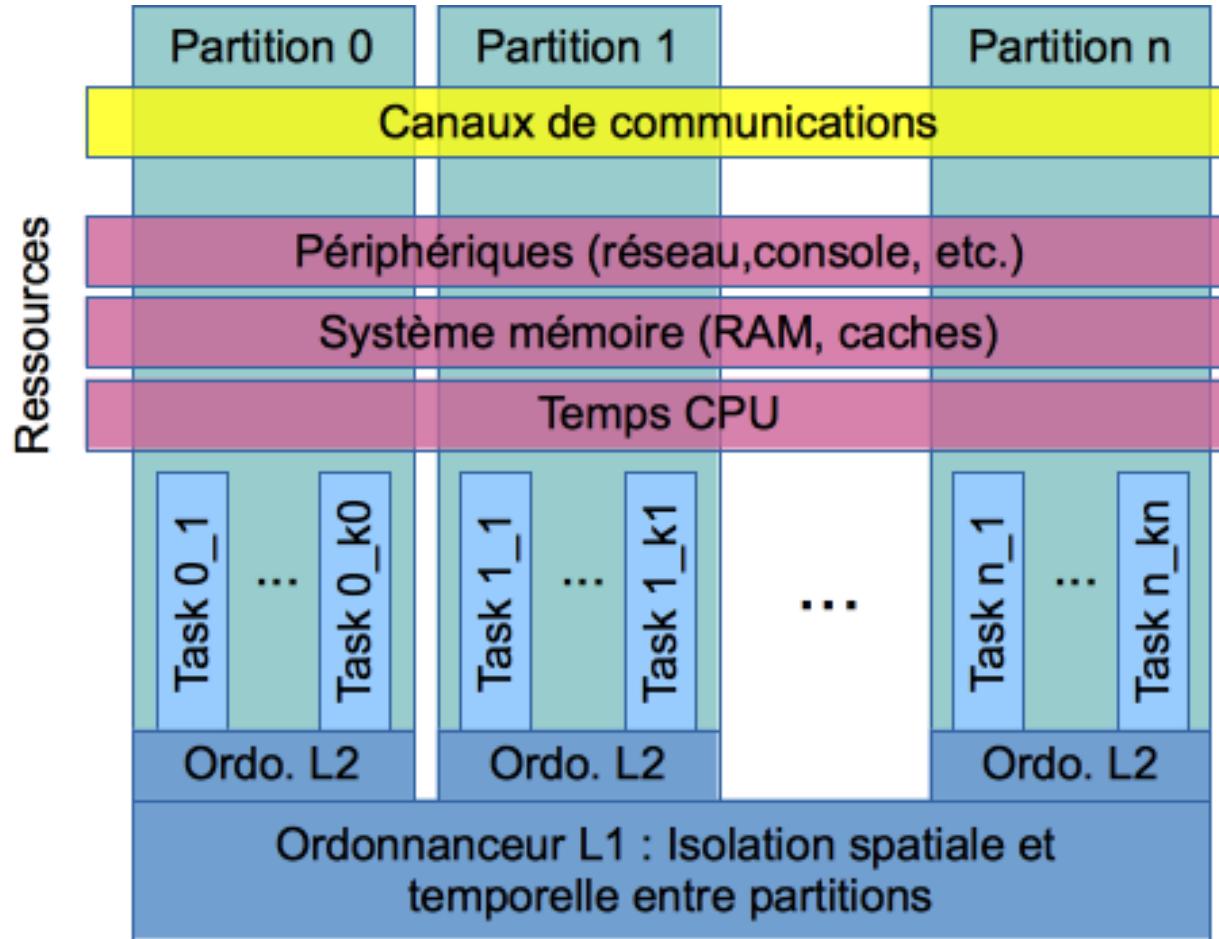
# Concepts fondamentaux

- TSP = Partitionnement spatial et temporel
  - Ordonnanceur hiérarchique à 2 niveaux
    - L1 = Static TDM au niveau module, entre partitions
    - L2 = Préemptif à base de priorités au niveau partition, entre tâches
  - Isolation mémoire et temporelle totale entre partitions
    - Communications inter-partitions par canaux/ports gérés par le niveau module
    - Attention aux canaux cachés
      - État des périphériques, des caches
      - Canaux cachés temporels

# Concepts fondamentaux

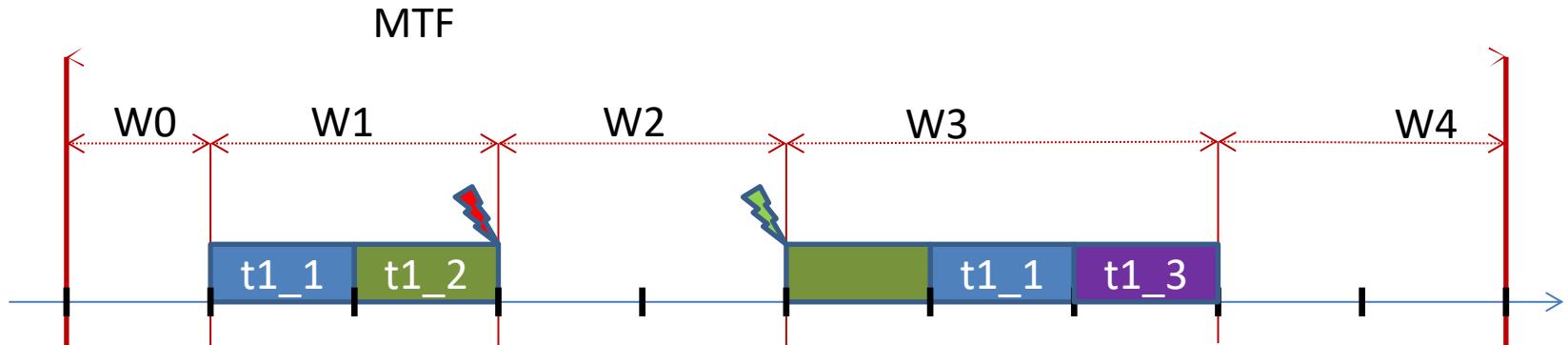
- Configuration
  - Un fichier XML
    - Configuration (statique) du module
    - Configuration initiale des partitions
    - HM niveau module
  - API APEX (C/Ada) pour le niveau partition
    - Evolution de la configuration des partitions
    - Configuration des tâches
    - HM niveau partition
    - Deux parties :
      - Point d'entrée - création de tâches et de ports
      - Tâches elles-mêmes
  - Tous les objets sont nommés, en XML et APEX

# Organisation d'un système Arinc 653



# Rappel : Niveau module (L1)

- Ordonnancement TDM de type multiplexage temporel statique
  - Souvent le temps est divisé en ticks (quantes de temps de taille fixe).
  - Exemple : tick = 500usec, MTF = 5ms, 5 fenêtres (W0-W4), dont W1, W3 associées à la partition 1

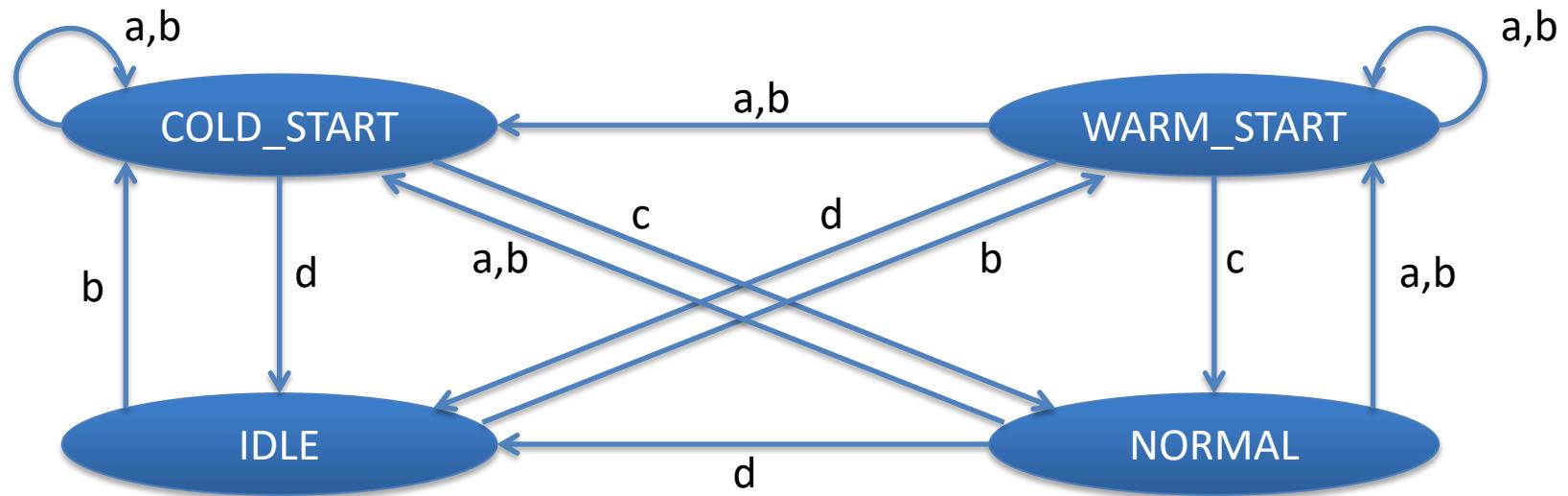


# Module

- Exécution
  - Phase d'initialisation
    - Dont la durée, non-négligeable, est à prendre en compte dans les systèmes formés de plusieurs modules
  - Exécution périodique TDM des partitions
    - Potentiellement interrompue par des erreurs

# Partition

- Modes d'exécution



a = force restart by APEX call, e.g. SET\_PARTITION\_MODE(COLD\_START)

b = force (re)start by HM, boot

c = enter preemptive mode – SET\_PARTITION\_MODE(NORMAL)

d = enter IDLE through either HM or APEX call, e.g. SET\_PARTITION\_MODE(IDLE)

# Partition

- Modes d'exécution
  - COLD\_START, WARM\_START
    - (Re-)initialisation de la partition
    - Point d'entrée appelé
    - Exécution préemptive du point d'entrée seulement !
  - NORMAL
    - Exécution multi-tâche
    - Le plus souvent, le point d'entrée n'est plus exécuté
  - IDLE
    - Aucun code n'est exécuté, les ressources ne sont pas utilisées

# Partition

- Changements de mode
  - APEX
    - SET\_PARTITION\_MODE
    - GET\_PARTITION\_STATUS
  - HM
    - XML config

# Partition

- Espace d'adressage unique
  - Complètement inaccessible à partir d'autres partitions
  - Partagé par toutes les tâches et par le point d'entrée
    - En général, il est possible de communiquer par variables partagées
    - Configuration des canaux, des processus, accessible à toutes les tâches
- Exécution de code en mode utilisateur seulement (non-privilégié)

# Point d'entrée d'une partition

- Fonction séquentielle démarrée lors du passage en mode COLD\_START ou WARM\_START
  - Initialisation de tous les canaux I/O
  - Création de tous les processus (tâches)
  - Démarrage de **certains** processus
  - Le plus souvent, passage en mode NORMAL
  - Dans la plupart des implémentations d'ARINC653, ce code s'exécute seulement en modes \*\_START
  - Peut être préempté par l'ordonnanceur L1

# Processus

- Unité d'exécution concurrente lors de l'exécution en mode NORMAL
- Execution en espace utilisateur
- Configuration
  - Espace mémoire, allouée par l'ordonnanceur L2 – code, données, pile
  - Temps réel – période, capacité, priorité
- État
  - Contexte matériel – état des registres (PC, SP, ...), potentiellement état des périphériques
  - État d'exécution L2
    - Attentes éventuelles (e.g. de sémaphores, événements)
    - Priorité courante
    - Échéance courante

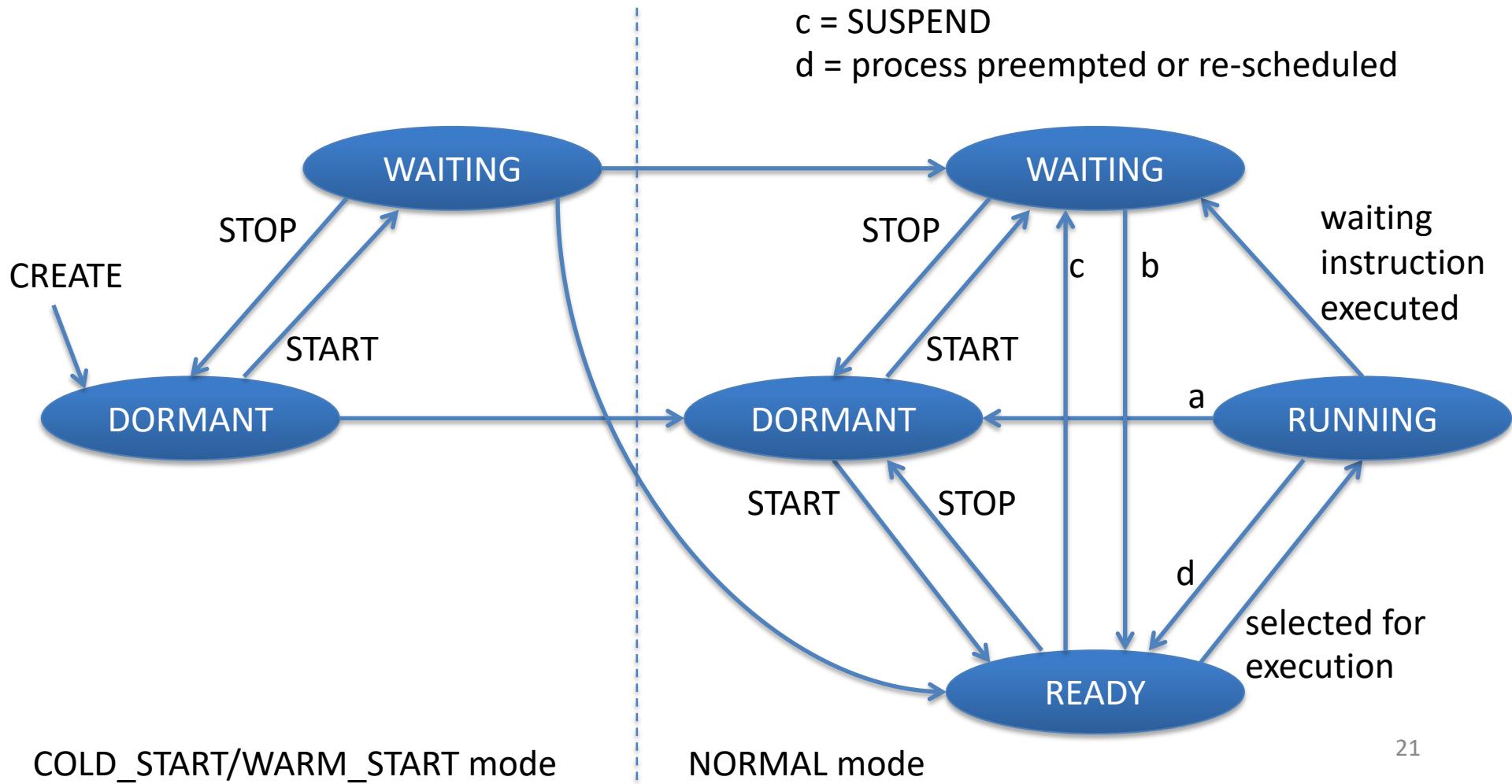
# État d'exécution L2 d'un processus

a = STOP\_SELF

b = RESUME, resource becomes available, or  
timeout expires

c = SUSPEND

d = process preempted or re-scheduled



# Ordonnancement L2 des processus

- Actif en mode NORMAL
- Préemptif à base de priorités
  - À chaque instant, un processus est choisi et rendu RUNNING parmi ceux de priorité la plus haute parmi ceux qui sont READY
  - Au maximum un processus RUNNING à chaque instant
- Les erreurs (e.g. échéance manquée) génèrent des événements HM (RPi653 – messages)
- Attention :
  - Si la plate-forme permet l'utilisation d'opérations non-interruptibles (e.g. appels systèmes longs), possibles canaux cachés
  - Si une tâche ne doit pas être interrompue, il faut s'assurer qu'elle est exécutée en début de fenêtre

# Ordonnancement L2 des processus

- API
  - Attributs (statiques) et état (dyn.) des processus

```
typedef struct {  
    SYSTEM_TIME_TYPE PERIOD;  
    SYSTEM_TIME_TYPE TIME_CAPACITY;  
    SYSTEM_ADDRESS_TYPE ENTRY_POINT;  
    STACK_SIZE_TYPE STACK_SIZE;  
    PRIORITY_TYPE BASE_PRIORITY;  
    DEADLINE_TYPE DEADLINE;  
    PROCESS_NAME_TYPE NAME;  
} PROCESS_ATTRIBUTE_TYPE;
```

```
typedef struct {  
    SYSTEM_TIME_TYPE DEADLINE_TIME;  
    PRIORITY_TYPE CURRENT_PRIORITY;  
    PROCESS_STATE_TYPE PROCESS_STATE;  
    PROCESS_ATTRIBUTE_TYPE ATTRIBUTES;  
} PROCESS_STATUS_TYPE;
```

# Ordonnancement L2 des processus

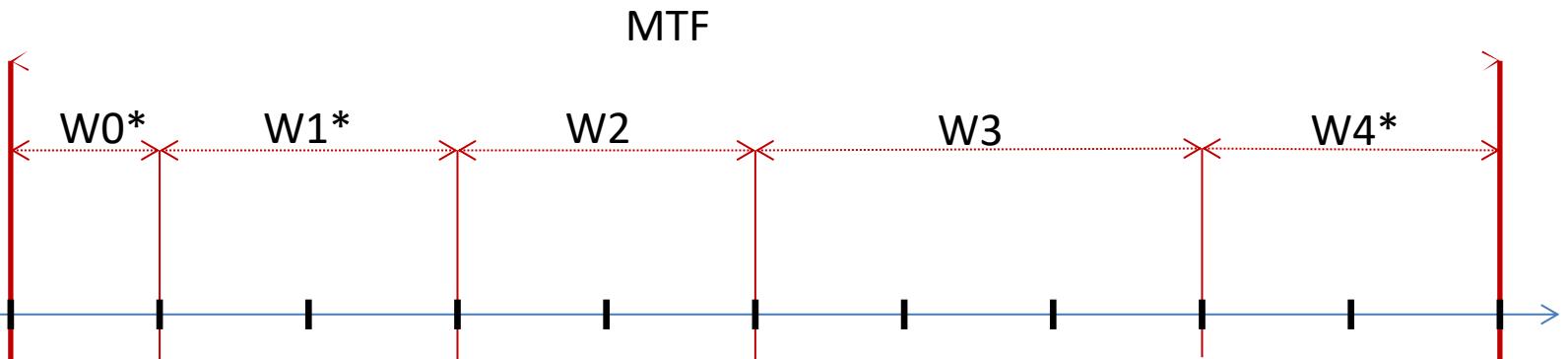
- Appels APEX (1/2)
  - CREATE\_PROCESS
  - START
  - DELAYED\_START
  - PERIODIC\_WAIT – attente de la prochaine date d'arrivée périodique
    - Seulement pour processus périodiques
  - GET\_PROCESS\_STATUS
  - GET\_PROCESS\_ID
  - GET\_MY\_ID
  - SET\_PRIORITY
  - SUSPEND\_SELF – place le processus en état WAITING jusqu'à RESUME
  - SUSPEND – place un processus en état WAITING jusqu'à RESUME
  - RESUME – passe en état READY ou RUNNING
  - STOP\_SELF – mise en état DORMANT
  - STOP – mise en état DORMANT d'un autre process

# Ordonnancement L2 des processus

- Appels APEX (2/2)
  - LOCK\_PREEMPTION – incremente le “lock level”-> ordo non-préemptif
  - UNLOCK\_PREEMPTION – décrémente le “lock level”. À 0 -> préemptif
  - TIMED\_WAIT – suspension de l’exécution pour une durée minimale
    - Parametre=0 -> demande de re-scheduling
    - Parfois utilisée pour l’implémentation d’ordonnanceurs périodiques
      - Dépendant de l’implantation
  - GET\_TIME – lecture de l’horloge système (gérée par le module, commune à toutes les partitions)
  - REPLENISH – augmentation de la deadline courante d’un processus en cours d’exécution

# Comment cela fonctionne ?

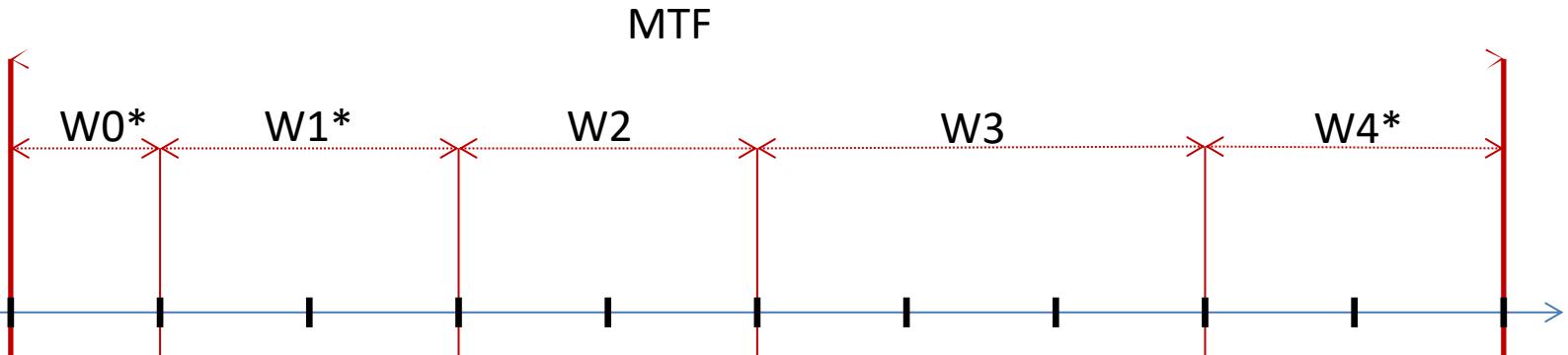
- Prenons un système avec 3 partitions
  - $P_0$  = partition système (gestion I/O)
  - $P_1, P_2$  = partitions applicatives
- $MTF = 10\text{ms}$ , tick = 1ms
  - Fenêtrage :  $W_0 \rightarrow P_0; W_1, W_3 \rightarrow P_1; W_2, W_4 \rightarrow P_2$
  - PPS :  $W_0, W_1, W_4$  (marquées avec \*)



# Comment cela fonctionne ?

- Point d'entrée de la partition 1:

```
...
T1_attr.PERIOD = MTF ;
CREATE_PROCESS(&T1_attr,&pid1,&rc) ;
START(pid1,&rc);
SET_PARTITION_MODE(NORMAL,&rc) ;
...
...
```

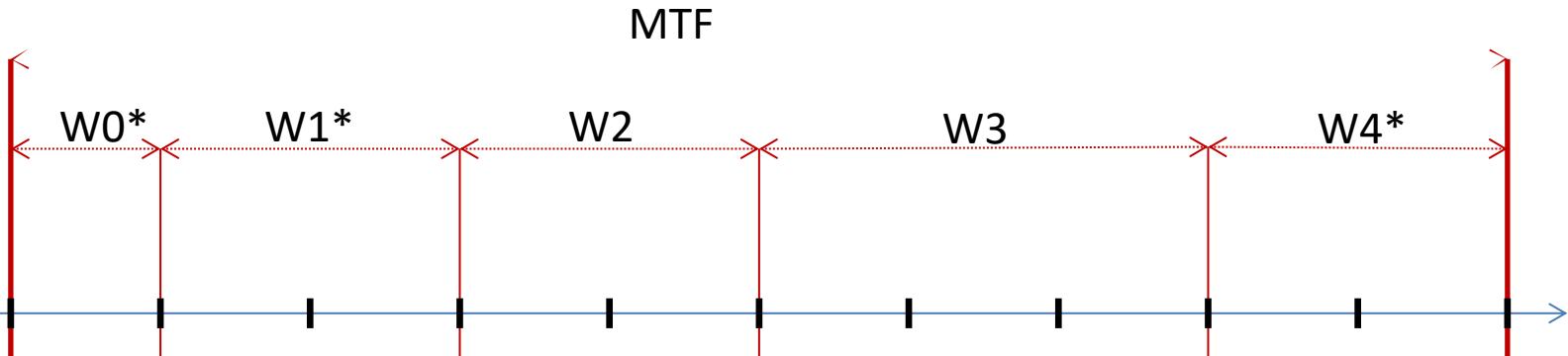


# Comment cela fonctionne ?

- Point d'entrée de la partition 1:

```
...
T1_attr.PERIOD = MTF ;
CREATE_PROCESS(&T1_attr,&pid1,&rc) ;
START(pid1,&rc);
SET_PARTITION_MODE(NORMAL,&rc) ;
...
```

- Exécution :

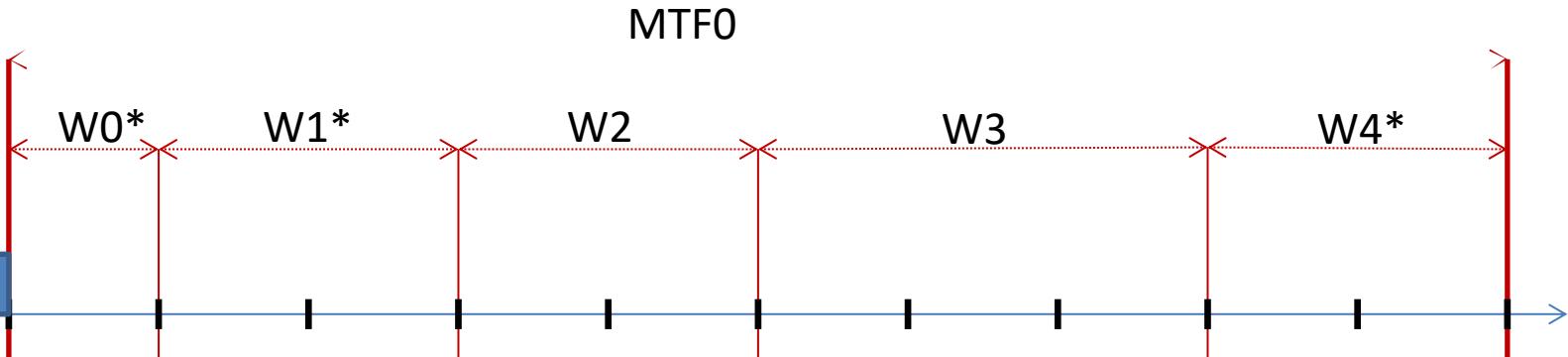


# Comment cela fonctionne ?

- Point d'entrée de la partition 1:

```
...
T1_attr.PERIOD = MTF ;
CREATE_PROCESS(&T1_attr,&pid1,&rc) ;
START(pid1,&rc);
SET_PARTITION_MODE(NORMAL,&rc) ;
...
```

- Exécution :

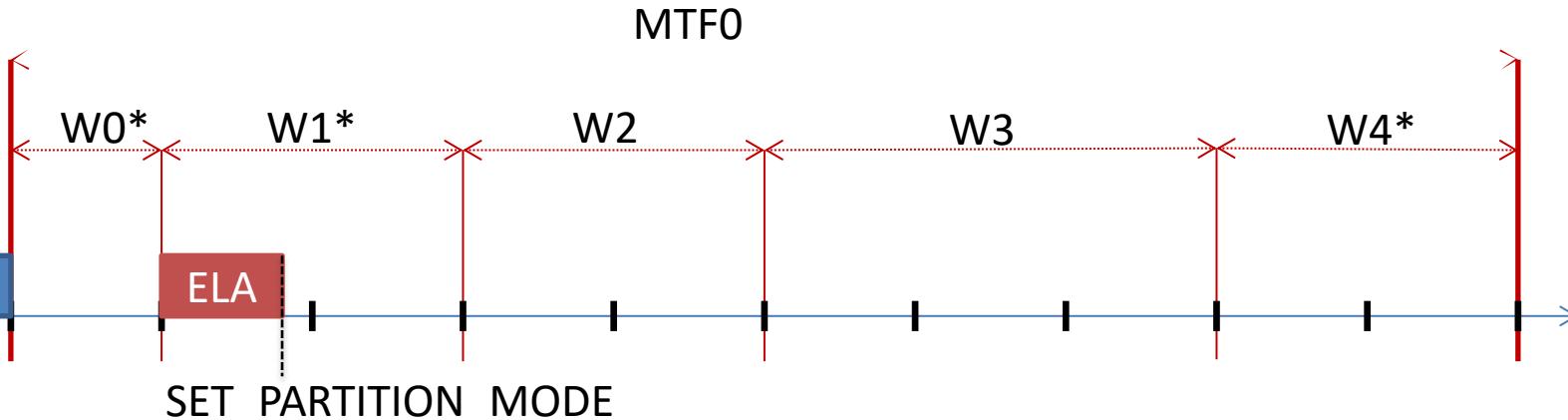


# Comment cela fonctionne ?

- Point d'entrée de la partition 1:

```
...
T1_attr.PERIOD = MTF ;
CREATE_PROCESS(&T1_attr,&pid1,&rc) ;
START(pid1,&rc);
SET_PARTITION_MODE(NORMAL,&rc) ;
...
```

- Exécution :

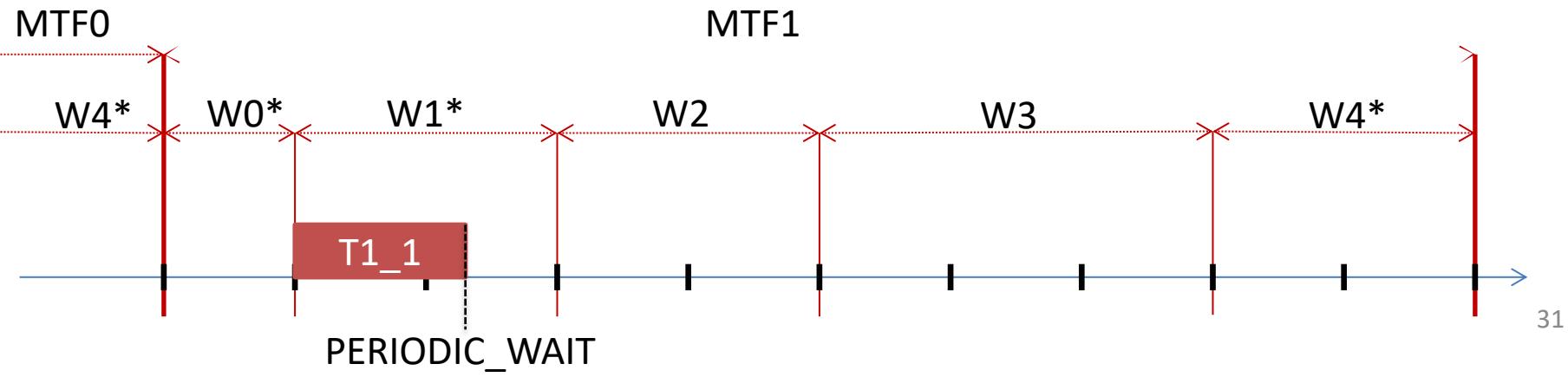


# Comment cela fonctionne ?

- Point d'entrée de la partition 1:

```
...
T1_attr.PERIOD = MTF ;
CREATE_PROCESS(&T1_attr,&pid1,&rc) ;
START(pid1,&rc);
SET_PARTITION_MODE(NORMAL,&rc) ;
...
```

- Exécution :

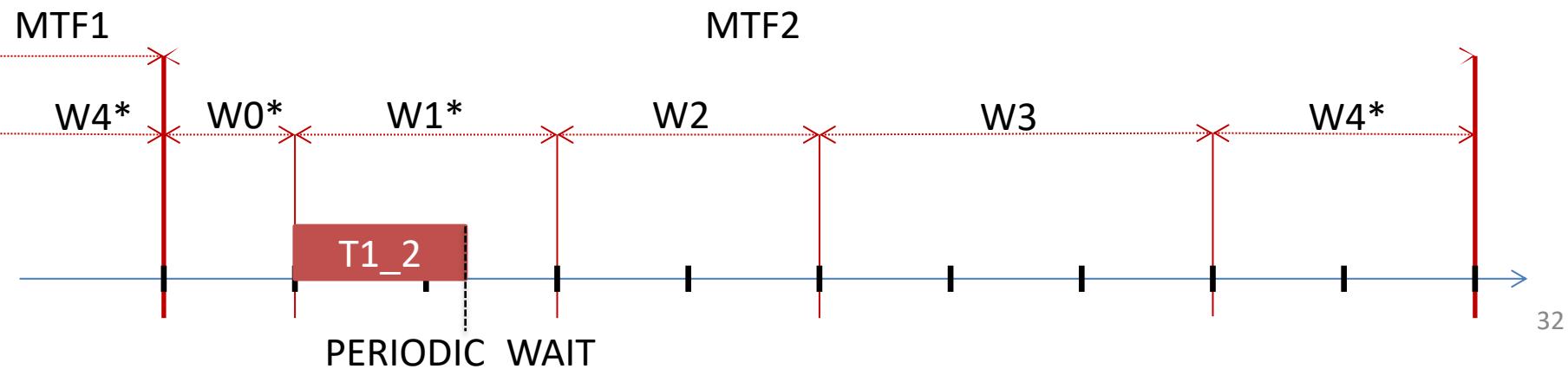


# Comment cela fonctionne ?

- Point d'entrée de la partition 1:

```
...
T1_attr.PERIOD = MTF ;
CREATE_PROCESS(&T1_attr,&pid1,&rc) ;
START(pid1,&rc);
SET_PARTITION_MODE(NORMAL,&rc) ;
...
```

- Exécution :

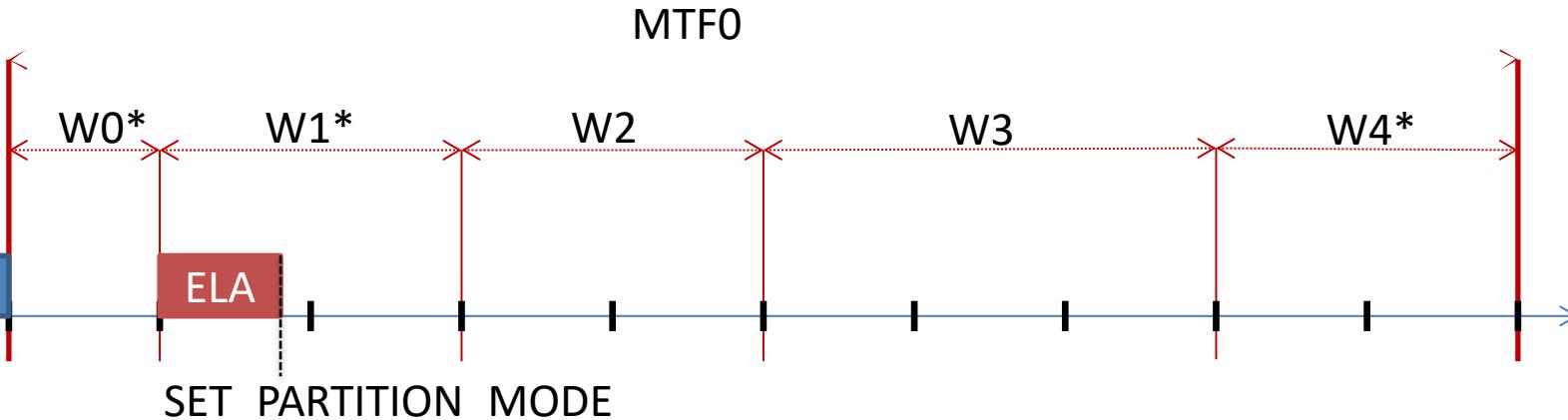


# Comment cela fonctionne ?

- Point d'entrée de la partition 1:

```
...
T1_attr.PERIOD = MTF ;
CREATE_PROCESS(&T1_attr,&pid1,&rc) ;
DELAYED_START(pid1,5MS,&rc);
SET_PARTITION_MODE(NORMAL,&rc) ;
...
...
```

- Exécution :

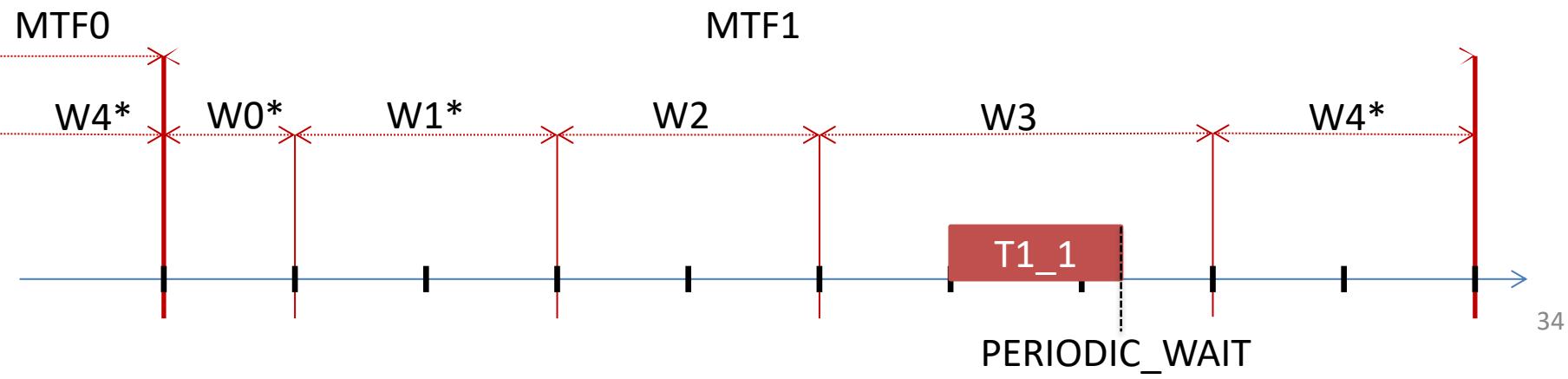


# Comment cela fonctionne ?

- Point d'entrée de la partition 1:

```
...
T1_attr.PERIOD = MTF ;
CREATE_PROCESS(&T1_attr,&pid1,&rc) ;
DELAYED_START(pid1,5MS,&rc);
SET_PARTITION_MODE(NORMAL,&rc) ;
...
...
```

- Exécution :

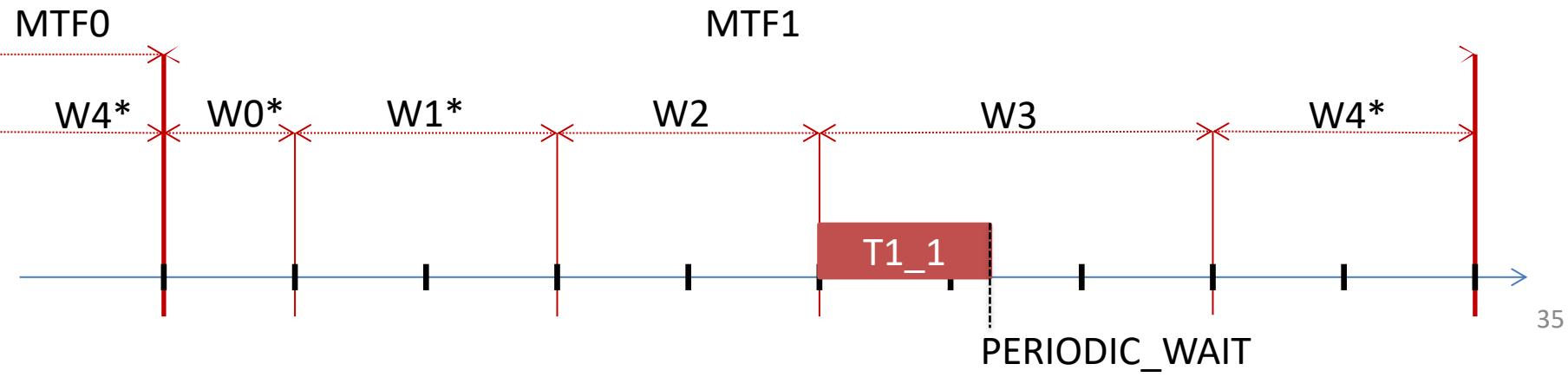


# Comment cela fonctionne ?

- Point d'entrée de la partition 1:

```
...
T1_attr.PERIOD = MTF ;
CREATE_PROCESS(&T1_attr,&pid1,&rc) ;
DELAYED_START(pid1,3MS,&rc);
SET_PARTITION_MODE(NORMAL,&rc) ;
...
...
```

- Exécution :

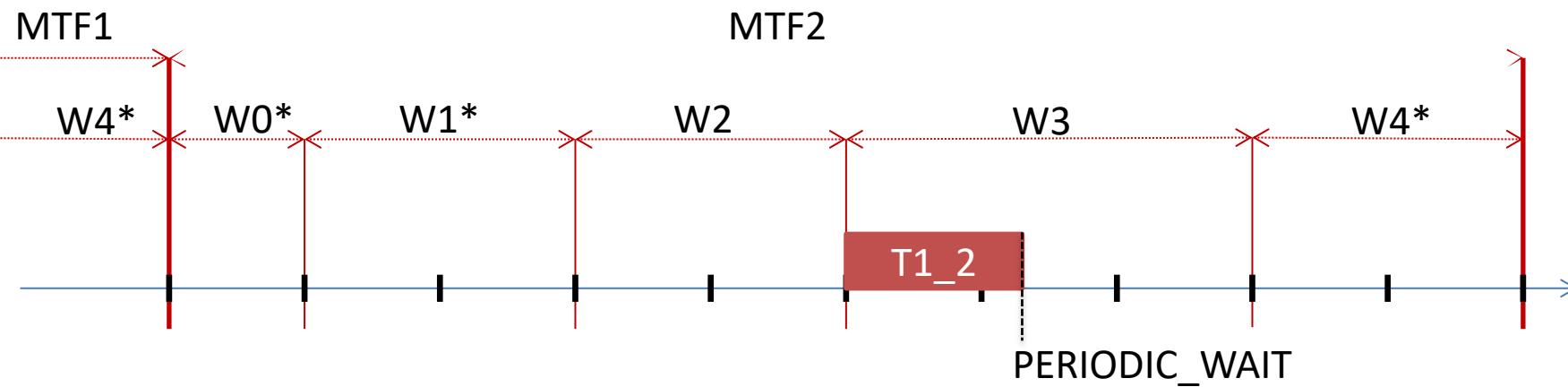


# Comment cela fonctionne ?

- Point d'entrée de la partition 1:

```
...
T1_attr.PERIOD = MTF ;
CREATE_PROCESS(&T1_attr,&pid1,&rc) ;
DELAYED_START(pid1,3MS,&rc);
SET_PARTITION_MODE(NORMAL,&rc) ;
...
...
```

- Exécution :

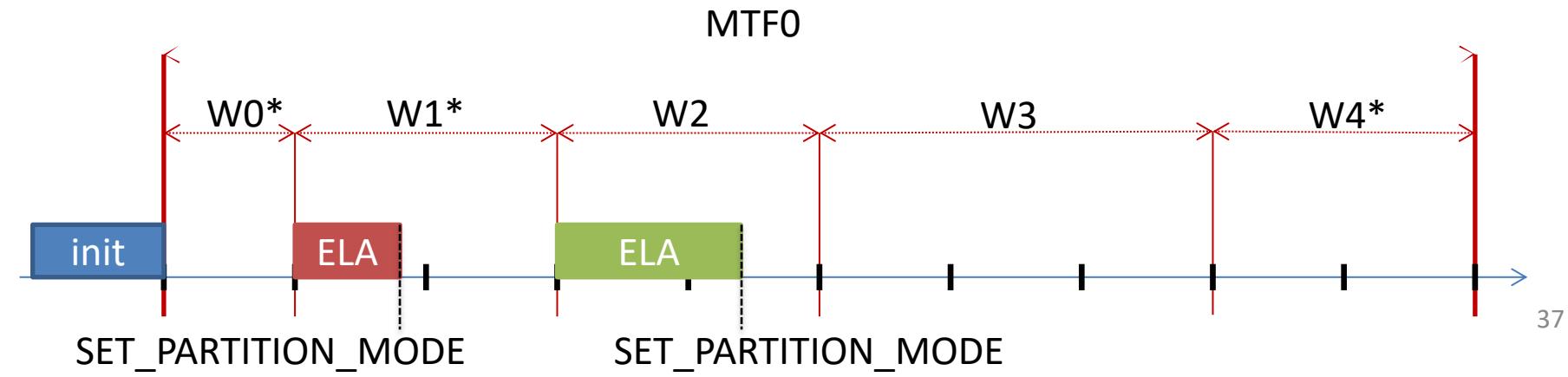


# Comment cela fonctionne ?

- Point d'entrée de la partition 2:

```
...
T1_attr.PERIOD = 5MS ; // MTF/2
CREATE_PROCESS(&T1_attr,&pid1,&rc) ;
START(pid1,&rc);
SET_PARTITION_MODE(NORMAL,&rc) ;
...
...
```

- Exécution :

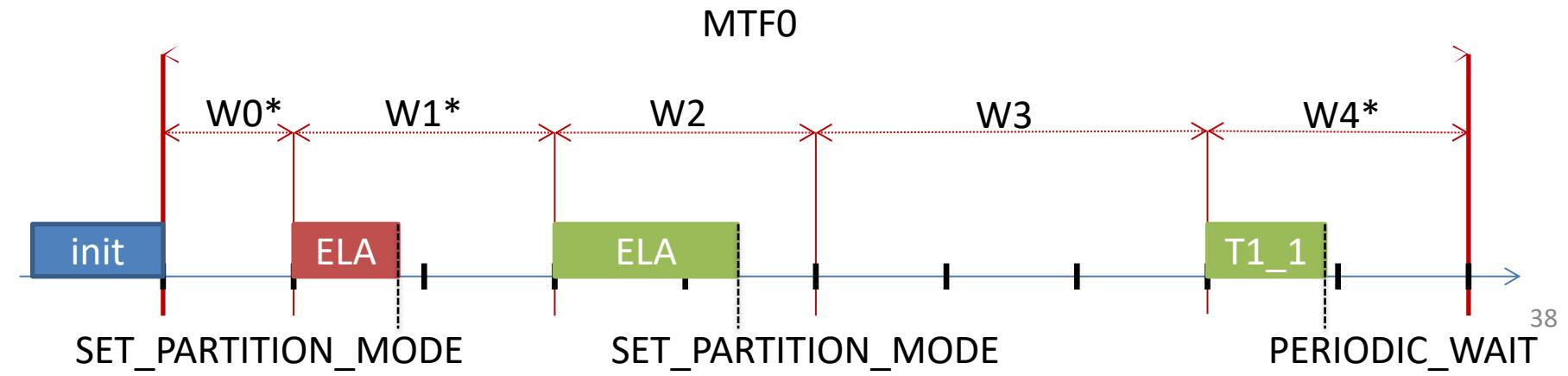


# Comment cela fonctionne ?

- Point d'entrée de la partition 2:

```
...
T1_attr.PERIOD = 5MS ;
CREATE_PROCESS(&T1_attr,&pid1,&rc) ;
START(pid1,&rc);
SET_PARTITION_MODE(NORMAL,&rc) ;
...
...
```

- Exécution :

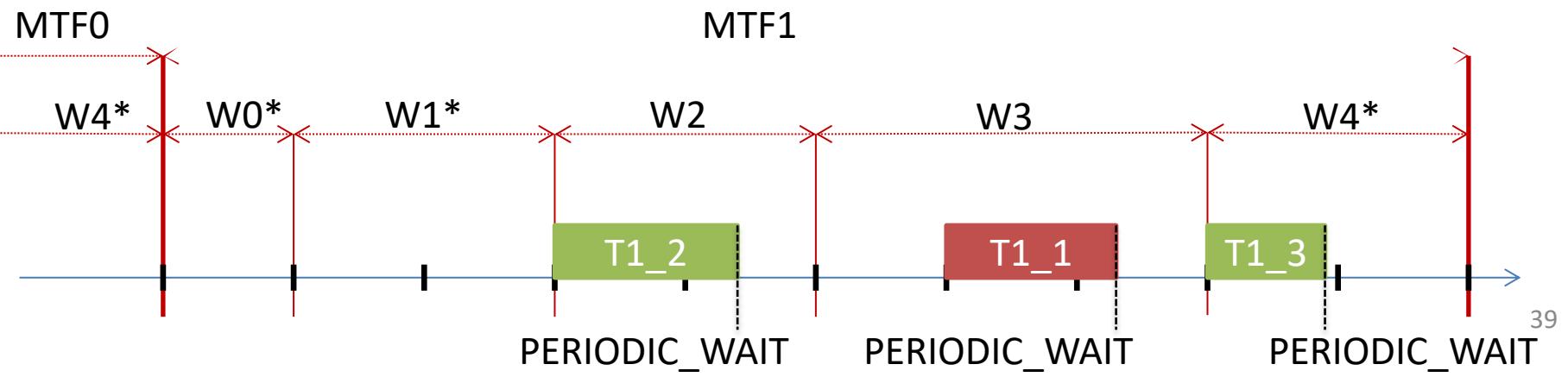


# Comment cela fonctionne ?

- Point d'entrée de la partition 2:

```
...
T1_attr.PERIOD = 5MS ;
CREATE_PROCESS(&T1_attr,&pid1,&rc) ;
START(pid1,&rc);
SET_PARTITION_MODE(NORMAL,&rc) ;
...
```

- Exécution :



# Préparation du TP

- RPi653 – implémentation d'ARINC 653 sur la Raspberry Pi
  - Détails d'implantation : Quand la Raspberry Pi se met à l'avionique. Open Silicium (“preprint” fourni uniquement pour le cours, ne pas diffuser)
  - Open Source (GPL v3)
    - Vous pouvez l'utiliser par la suite dans des applications open-source

# RPi653

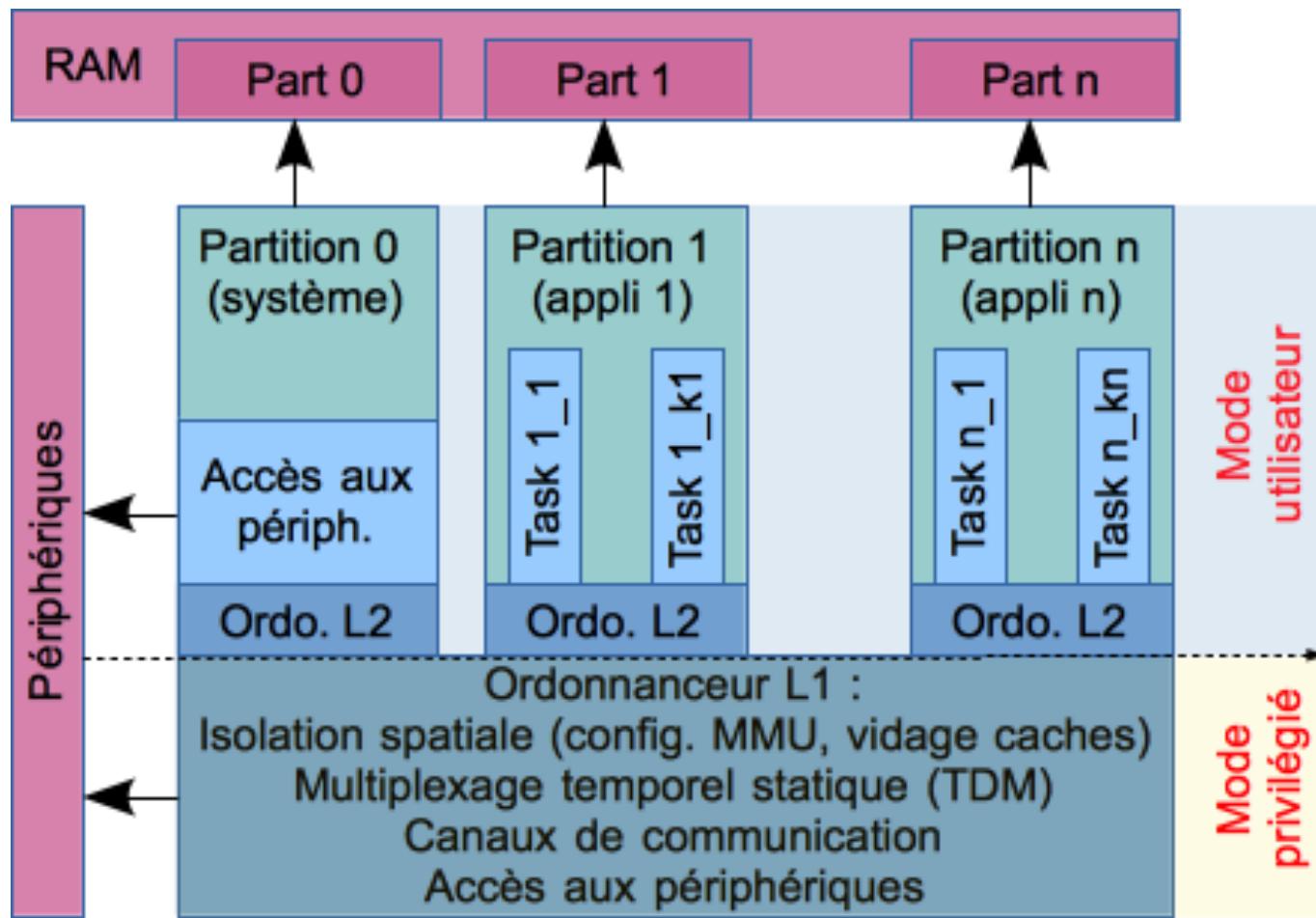
- Implantation partielle
  - Pas de Health Monitoring
  - Couverture partielle d'APEX
    - Seulement des ports “queuing” + simplifications
    - Communications point-à-point seulement
      - seul type de communication qui ne soit pas dépendant de l'implémentation
    - Simplification du fichier de configuration
- Choix particuliers d'implémentation (1/2)
  - Niveau L2 en mode utilisateur
    - Isolation totale entre niveau L1 et L2 (possibilité de changer l'ordonnanceur L2)
  - Simplifications
    - Toutes les périodes et les durées doivent être données en multiples du tick
    - Configuration des ports réalisée par défaut
    - Constantes de temps en msec, pas en nsec

# RPi653

- Choix particuliers d'implantation (2/2)
  - Partition système 0
    - Partie des pilotes de périphériques qui peut être exécutée en mode utilisateur
      - Pilote de console
      - Accès aux périphériques
    - Des fenêtres doivent lui être dédiées
    - Isolée par rapport aux autres partitions
  - Extension d'APEX par un ensemble de services système
    - Console printing (facilite le débogage)

# RPi653

- Choix particuliers d'implantation

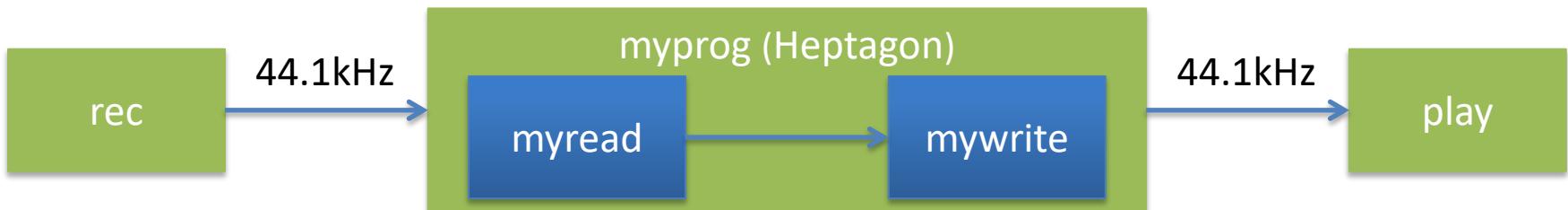


# Préparation du TP

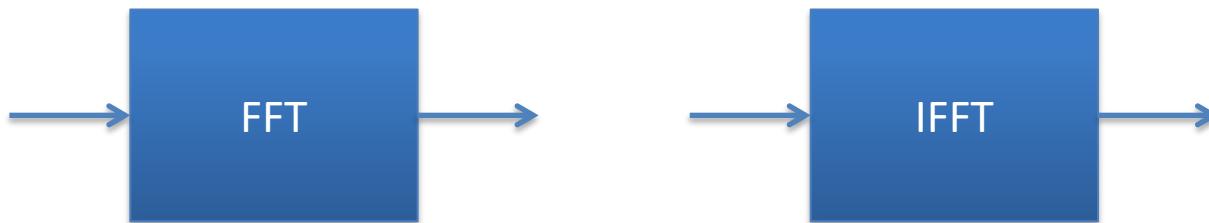
- Objectif : application de "pitch shifting"
  - Extension de la FFT avec du code fourni

# Application de pitch shifting

- TP précédents (1/2)
  - Lien entre code Heptagon et le système audio de l'ordinateur

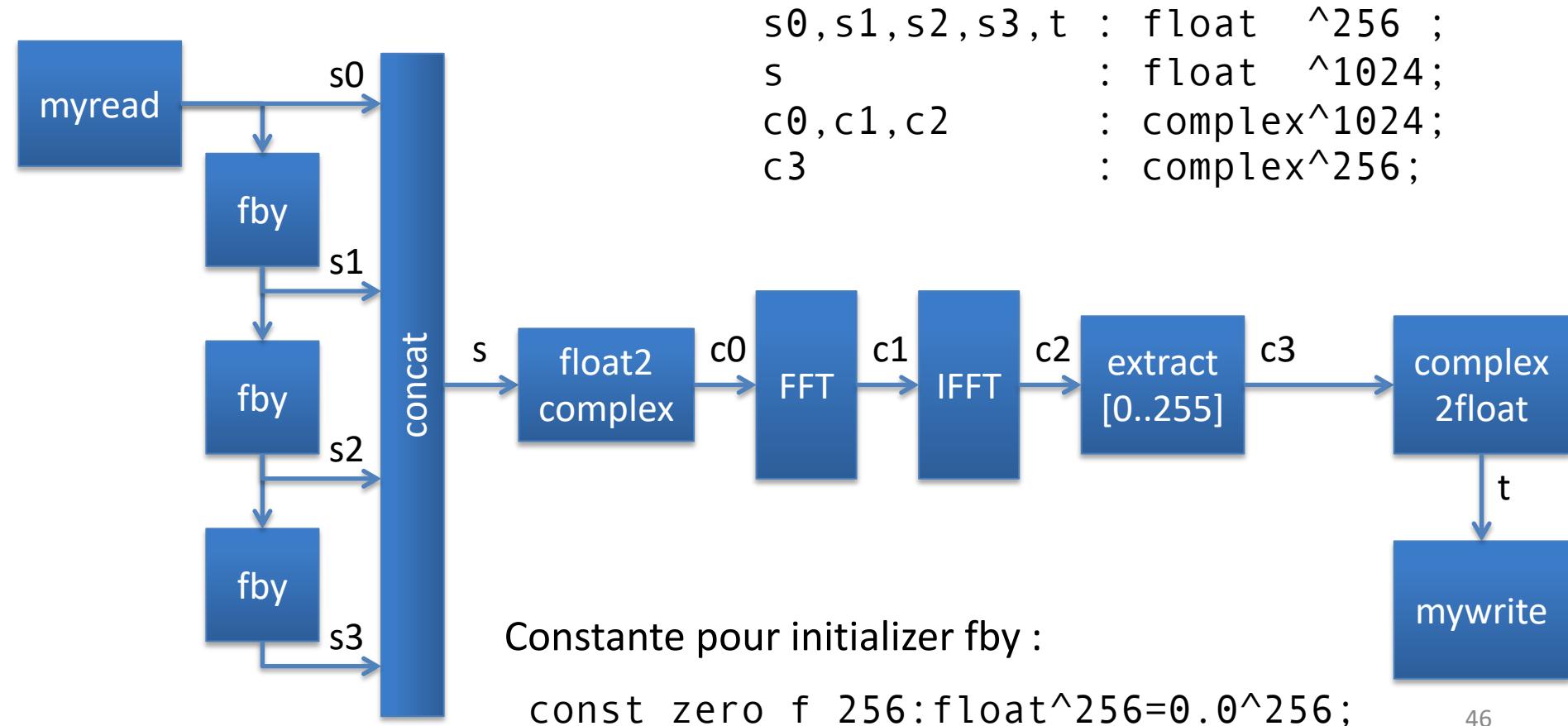


- Programmation de la FFT et de la IFFT



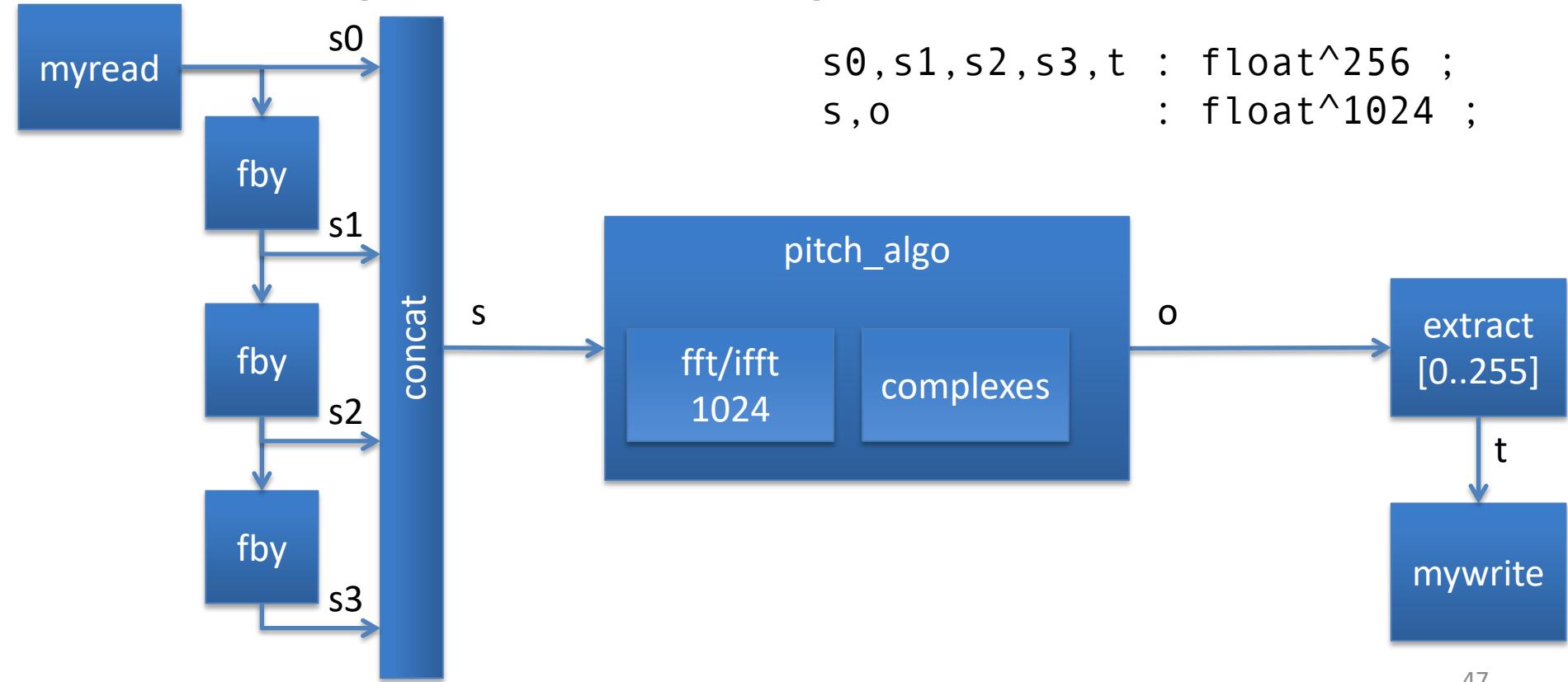
# Application de pitch shifting

- TPs précédents (2/2)



# Application de pitch shifting

- Aujourd'hui :
  - Changer le timbre du signal d'entrée



# Application de pitch shifting

- Le noeud pitch\_algo est fourni
  - Vous devrez l'interfacer avec :
    - la FFT/IFFT à 1024 points que vous avez écrite
    - votre bibliothèque sndio
    - le noeud principal qui implémente la fenêtre glissante
  - Attention : le Makefile fourni fonctionne si :
    - la fft est contenue dans fft.ept
    - l'interface sndio est contenue dans sndio.epi
    - le code C de sndio est contenu dans sndio\_c/sndio.[ch]
    - le noeud principal s'appelle "main"
    - Divers ajustements peuvent être nécessaires