Une approche synchrone à la conception de systèmes embarqués temps réel

Dumitru Potop-Butucaru

dumitru.potop@inria.fr

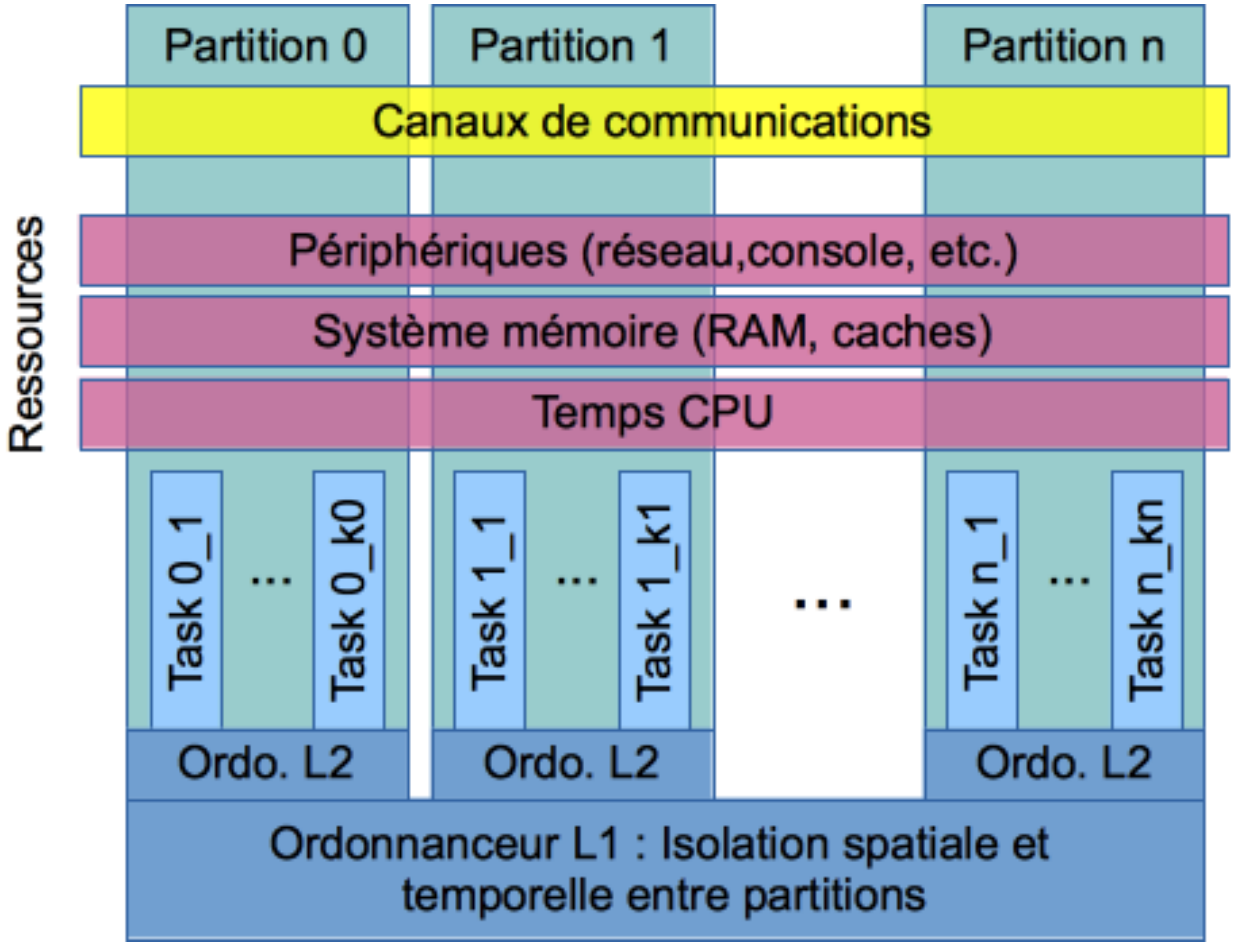
cours EPITA, 2022, 6ème séance

Contenu de ce cours

- Standard ARINC 653
- Système d'exploitation RPi653

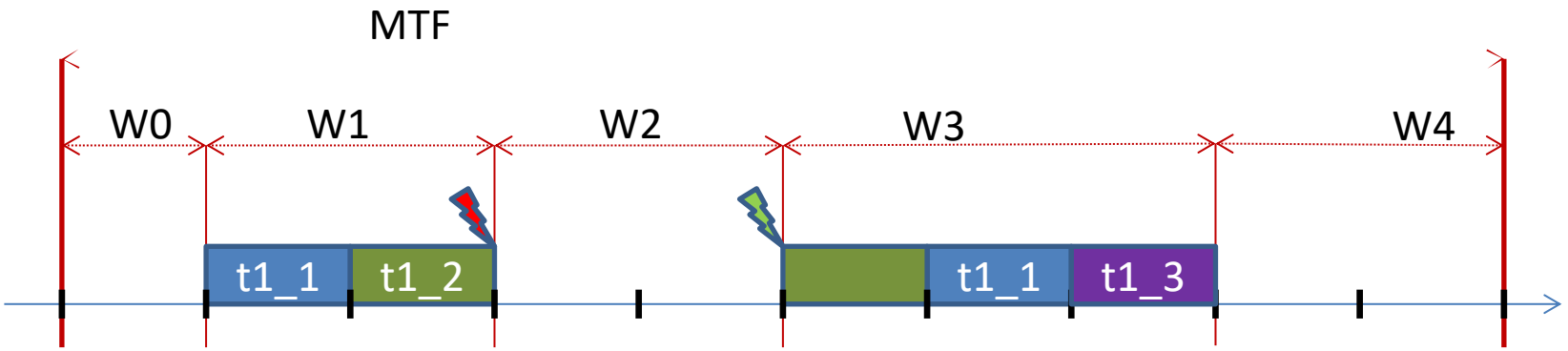
- Préparation du TP
  - Utilisation de RPi653
  - Démo

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

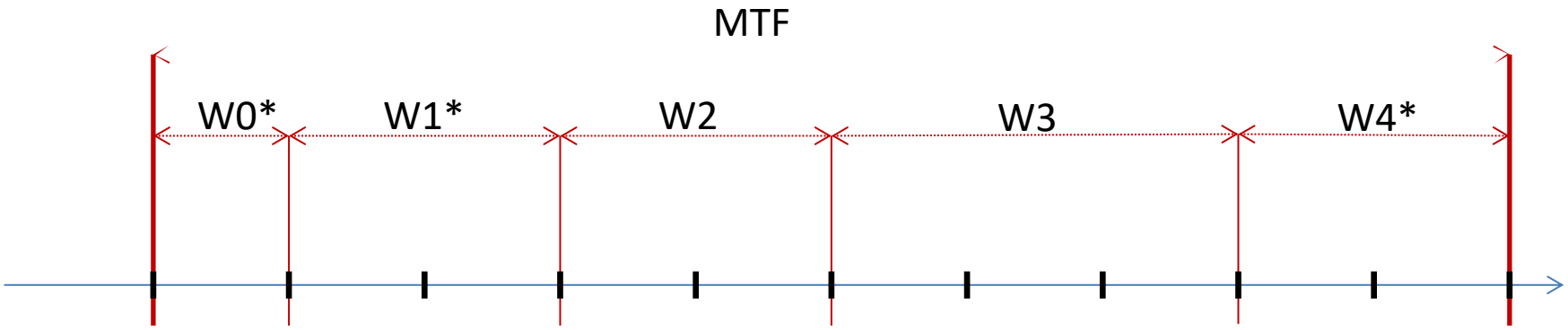


Rappel : Niveau partition (L2)

- Fonctions et API APEX en C
  - Gestion de la vie d'une application/partition
    - SET_PARTITION_MODE
  - Ordonnancement des tâches (juste le sous-ensemble périodique)
    - `CREATE_PROCESS`
    - `DELAYED_START et START`
      - Attention à la date de 1ère arrivée qui dépend du point de référence temporelle
    - `PERIODIC_WAIT`
  - Communications inter-partitions
    - Configuration des ports, envoi, réception
  - Communications intra-partition
  - Actions HM

Comment cela fonctionne ?

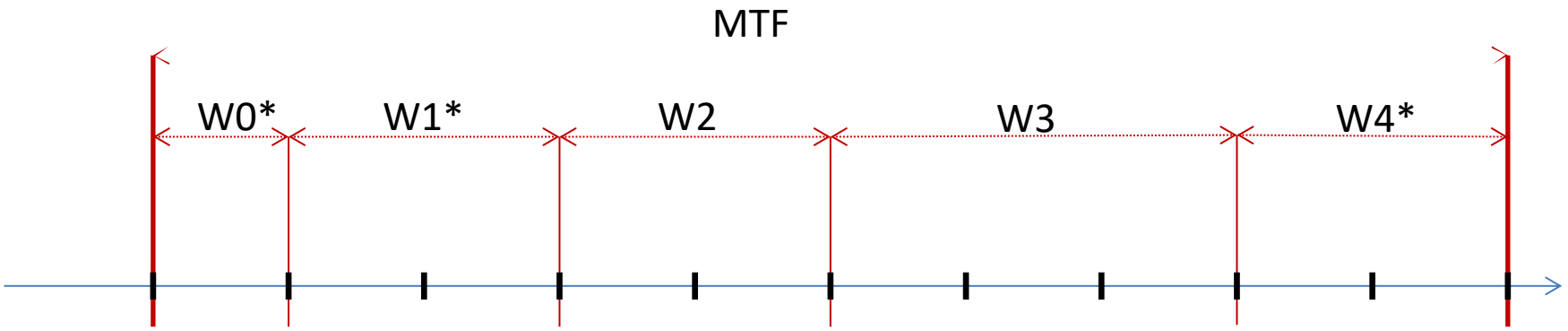
- Prenons un système avec 3 partitions
  - P0 = partition système (gestion I/O)
  - P1, P2 = partitions applicatives
- MTF = 10ms, tick = 1ms
  - Fenêtrage : W0-> P0; W1,W3->P1; W2,W4->P2
  - PPS : W0, W1, W4 (marquées avec *)



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) ;
...
```

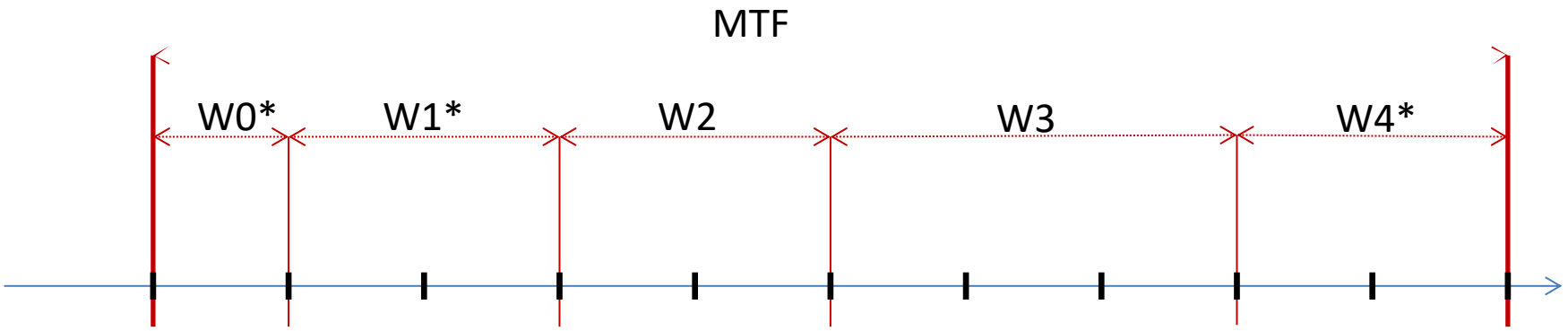


Comment cela fonctionne ?

- 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 :

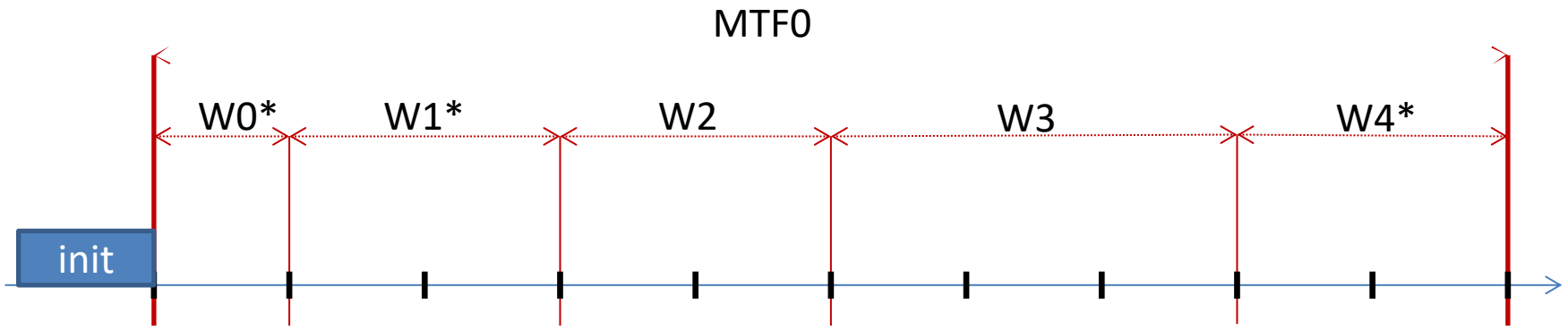


Comment cela fonctionne ?

- 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 :

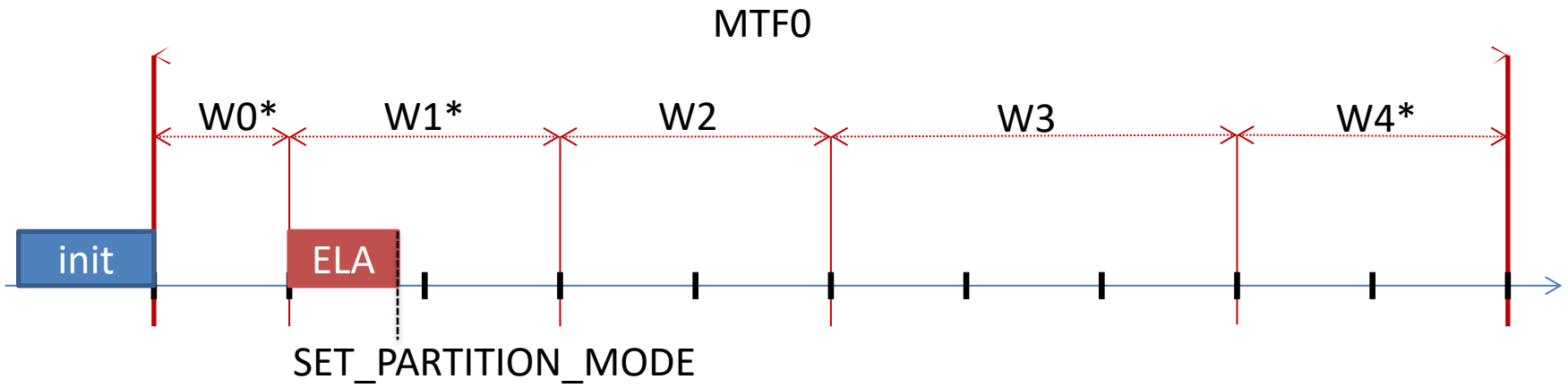


Comment cela fonctionne ?

- 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 :

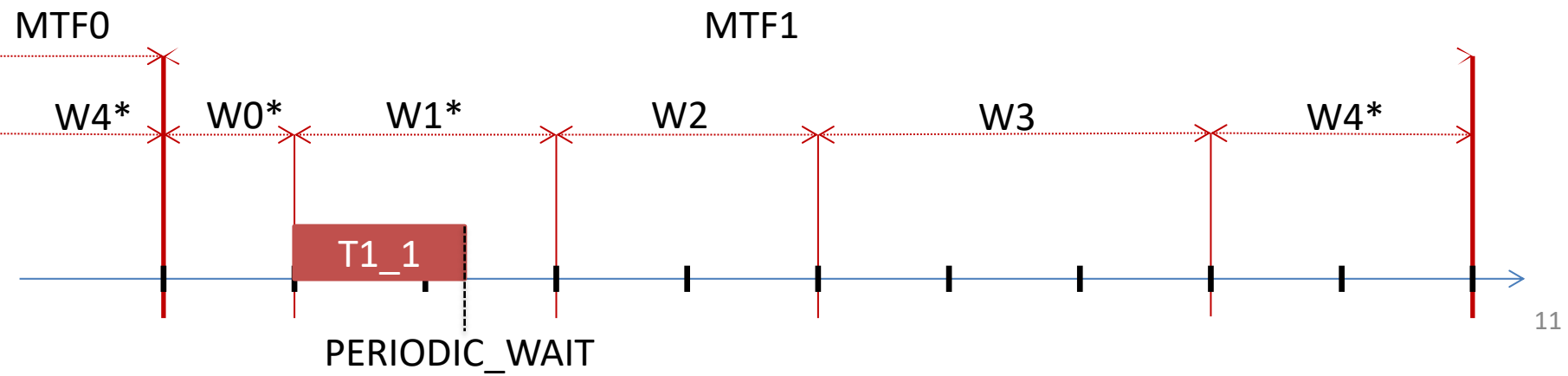


Comment cela fonctionne ?

- 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 :

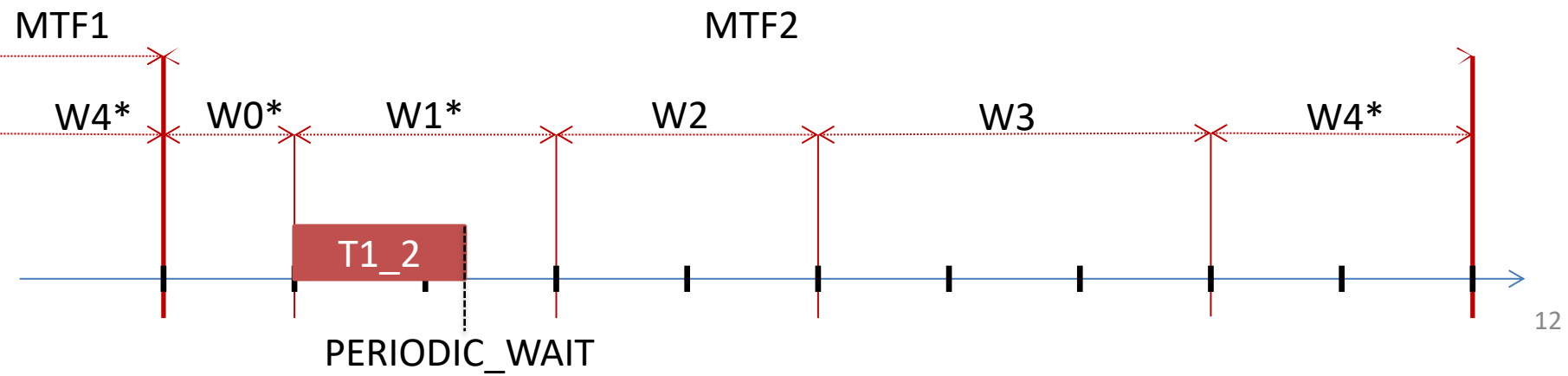


Comment cela fonctionne ?

- 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 :

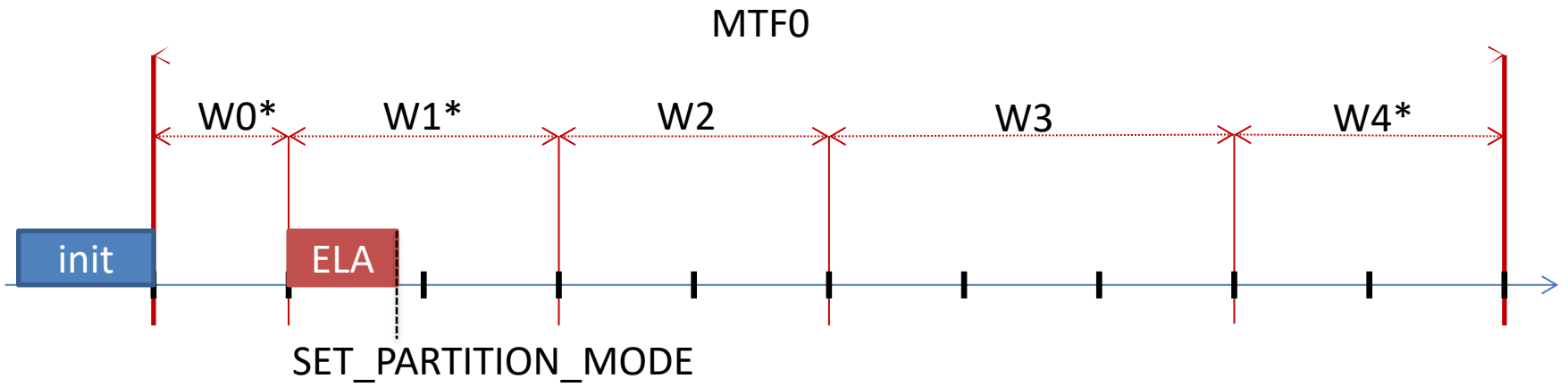


Comment cela fonctionne ?

- 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 :

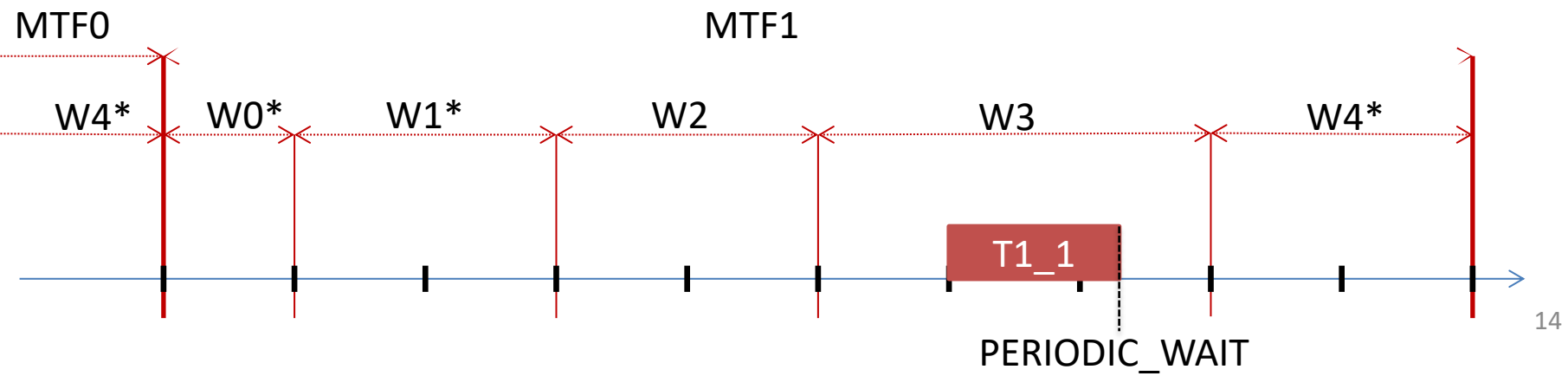


Comment cela fonctionne ?

- 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 :

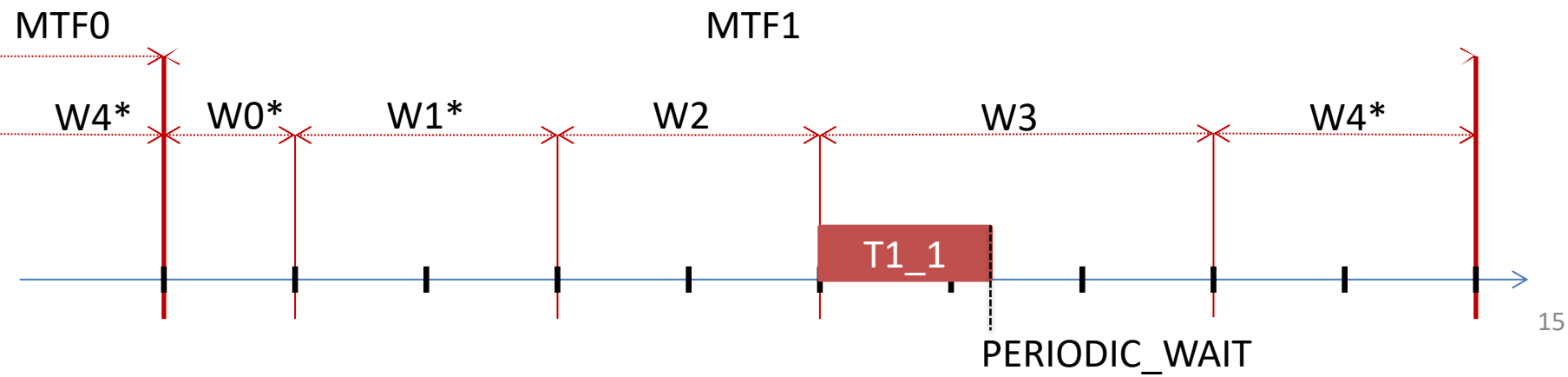


Comment cela fonctionne ?

- 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 :

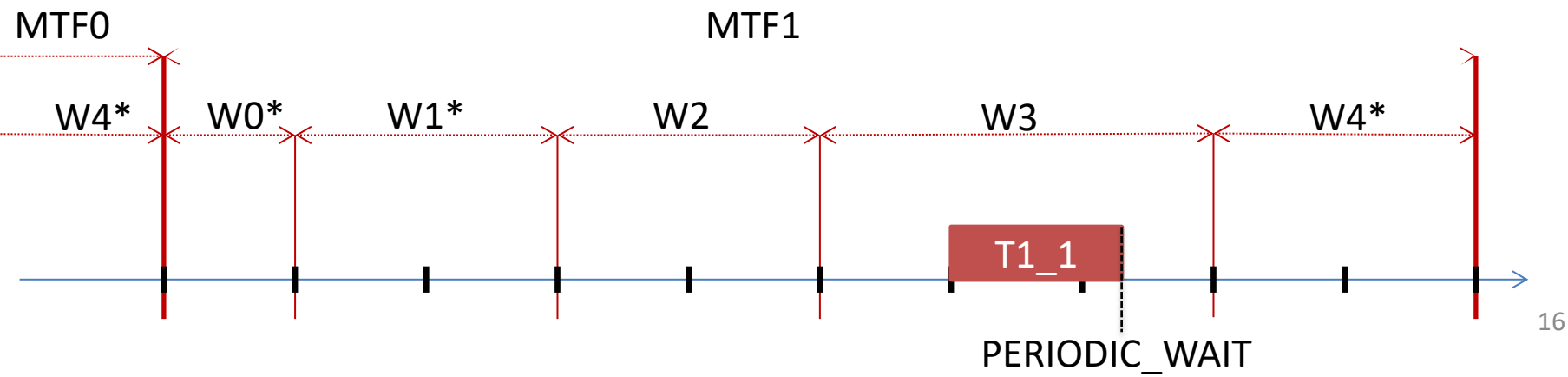


Comment cela fonctionne ?

- Point d'entrée de la partition 1:

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

- Exécution :

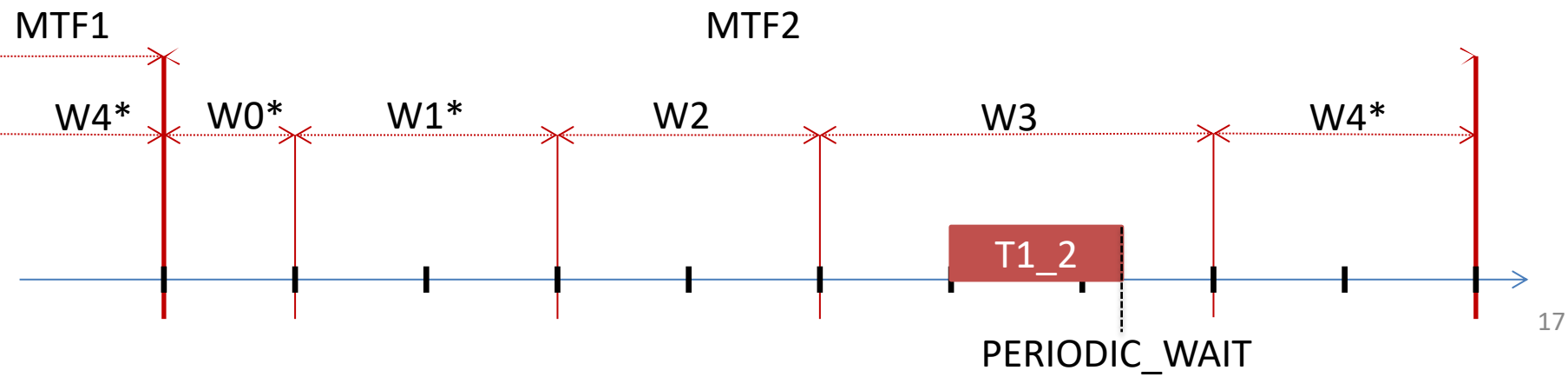


Comment cela fonctionne ?

- 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 :

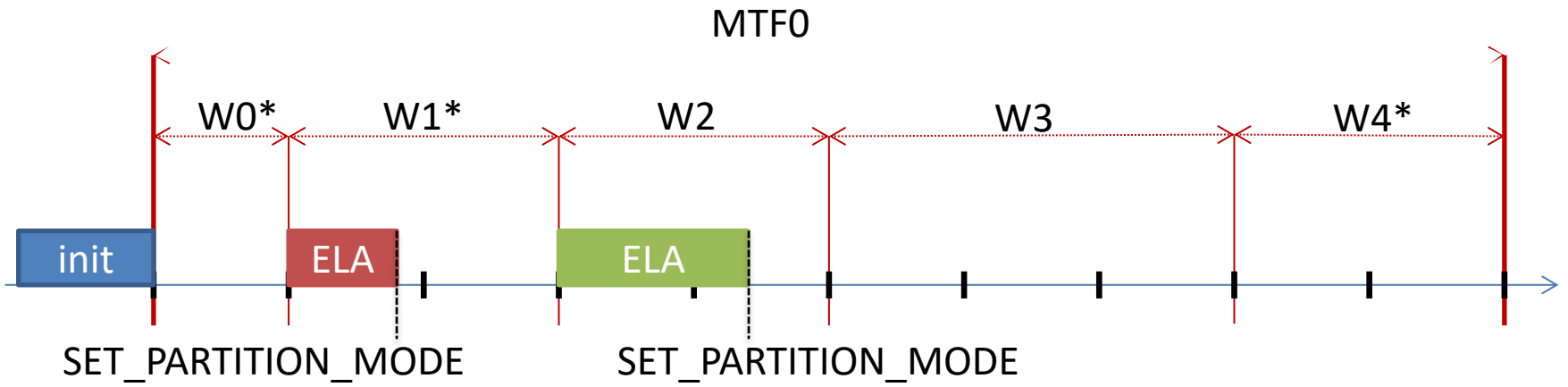


Comment cela fonctionne ?

- Point d'entrée de la **partition 2**:

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

- Exécution :

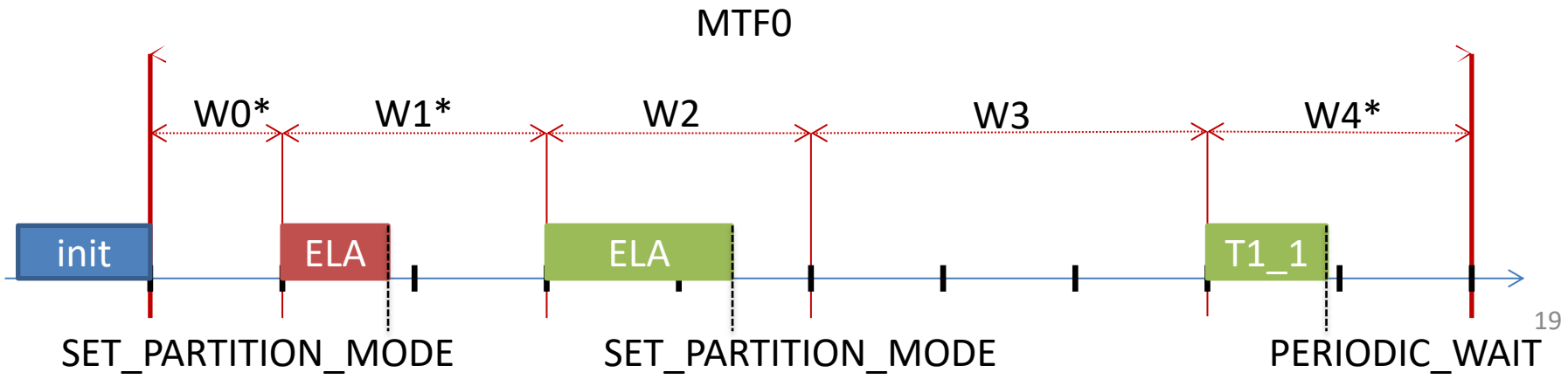


Comment cela fonctionne ?

- 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 :

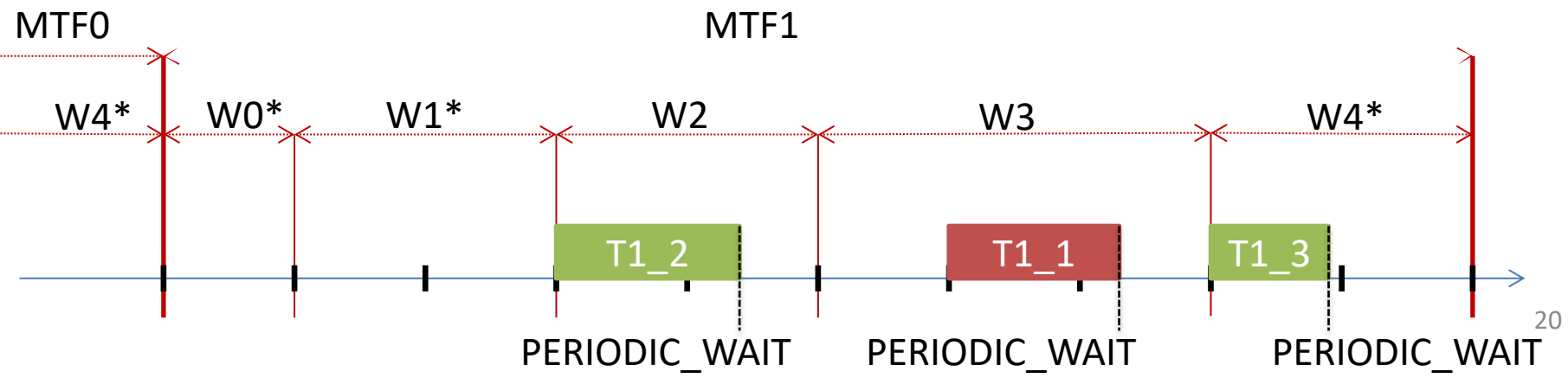


Comment cela fonctionne ?

- 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 :



Codage de politiques d'ordo classiques

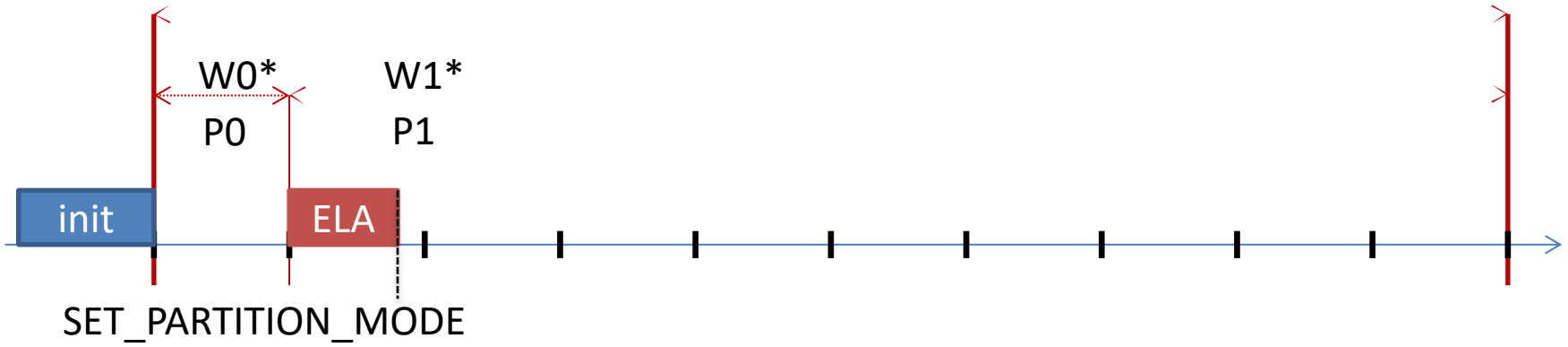
- RM: n=2, T_1 =5, C_1 =2, T_2 =7, C_2 =3

```
...
T1_attr.PERIOD = 5MS ;
T1_attr.BASE_PRIORITY = 3 ; /* greater priority */
CREATE_PROCESS(&T1_attr,&pid1,&rc) ;
START(pid1,&rc);
T2_attr.PERIOD = 7MS ;
T2_attr.BASE_PRIORITY = 2 ; /* lower priority */
CREATE_PROCESS(&T2_attr,&pid2,&rc) ;
START(pid2,&rc);
SET_PARTITION_MODE(NORMAL,&rc) ;
...
```

Codage de politiques d'ordo classiques

- RM: n=2, T_1 =5, C_1 =2, T_2 =7, C_2 =3

```
...
T1_attr.PERIOD = 5MS ;
T1_attr.BASE_PRIORITY = 3 ; /* greater priority */
CREATE_PROCESS(&T1_attr,&pid1,&rc) ;
START(pid1,&rc);
T2_attr.PERIOD = 7MS ;
T2_attr.BASE_PRIORITY = 2 ; /* lower priority */
CREATE_PROCESS(&T2_attr,&pid2,&rc) ;
START(pid2,&rc);
SET_PARTITION_MODE(NORMAL,&rc) ;
...
```

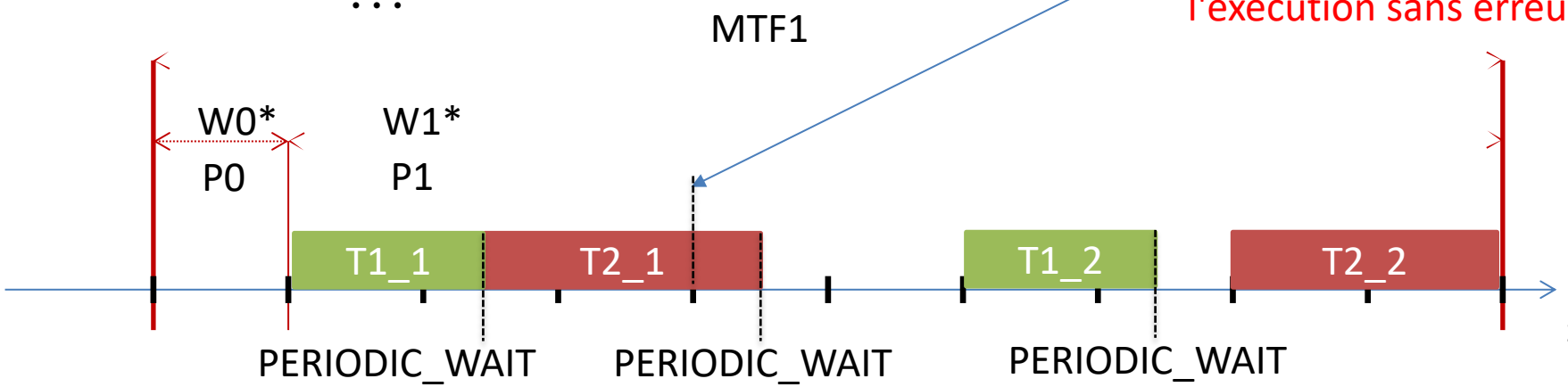


Codage de politiques d'ordo classiques

- RM: n=2, T_1 =5, C_1 =2, T_2 =7, C_2 =3

```
...
T1_attr.PERIOD = 5MS ;
T1_attr.BASE_PRIORITY = 3 ; /* greater priority */
CREATE_PROCESS(&T1_attr,&pid1,&rc) ;
START(pid1,&rc);
T2_attr.PERIOD = 7MS ;
T2_attr.BASE_PRIORITY = 2 ; /* lower priority */
CREATE_PROCESS(&T2_attr,&pid2,&rc) ;
START(pid2,&rc);
SET_PARTITION_MODE(NORMAL,&rc) ;
...
```

Deadline miss!
C_2 doit être 6 pour permettre l'exécution sans erreur.



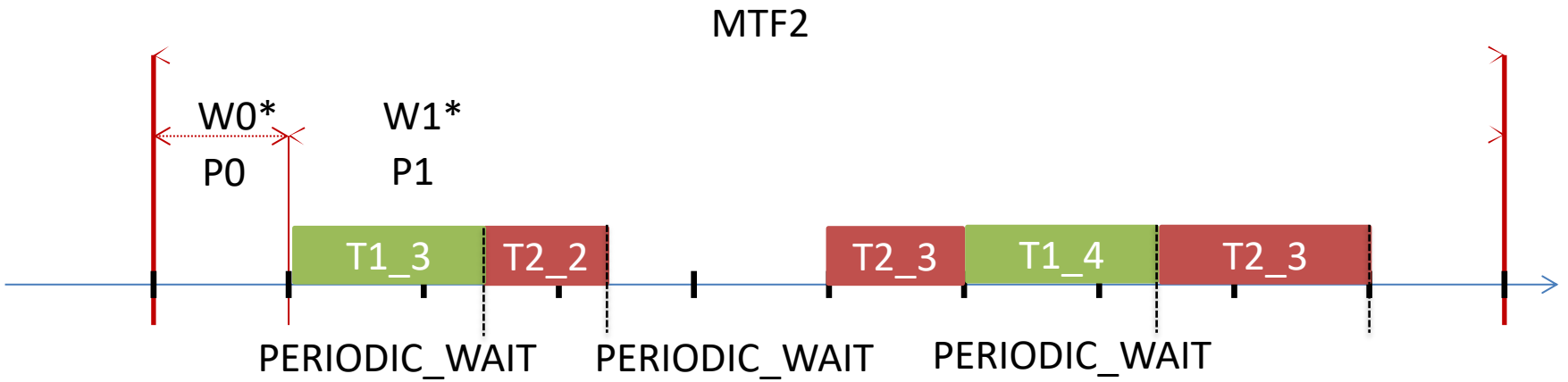
Codage de politiques d'ordo classiques

- Configuration L1 – fichier config.pok (RPi653)

```
tick        0x3e8
mtf         0x2710

windows 2
0           0x0      0x3e8   0 1
1           0x3e8    0x2710  1 1
```

MTF = 10ms = 10 ticks de 1ms chacun



Codage de politiques d'ordo classiques

- Mais EDF ?
  - En changeant dynamiquement la priorité, en fonction des deadlines, à chaque arrivée de tâche

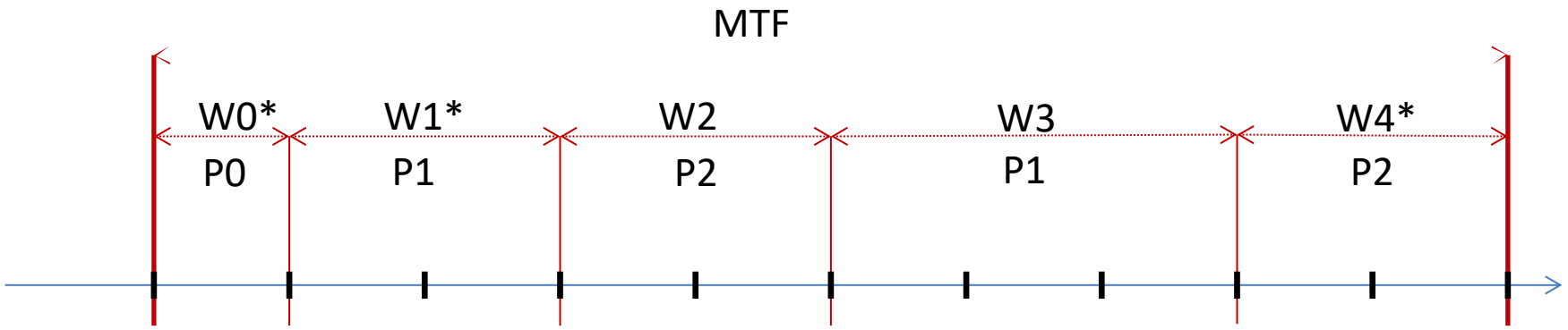
Configuration L1 du MTF

- Fichier config.pok (RPi653)

```
tick      0x8000
mtf      0x50000
```

MTF ~320ms = 10 ticks de ~32ms chacun

```
windows 5
0         0x0      0x8000   0 1
1         0x8000   0x18000  1 1
2         0x18000  0x28000  2 0
3         0x28000  0x40000  1 0
4         0x40000  0x50000  2 1
```



RPi653

- RPi653 – implémentation de 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

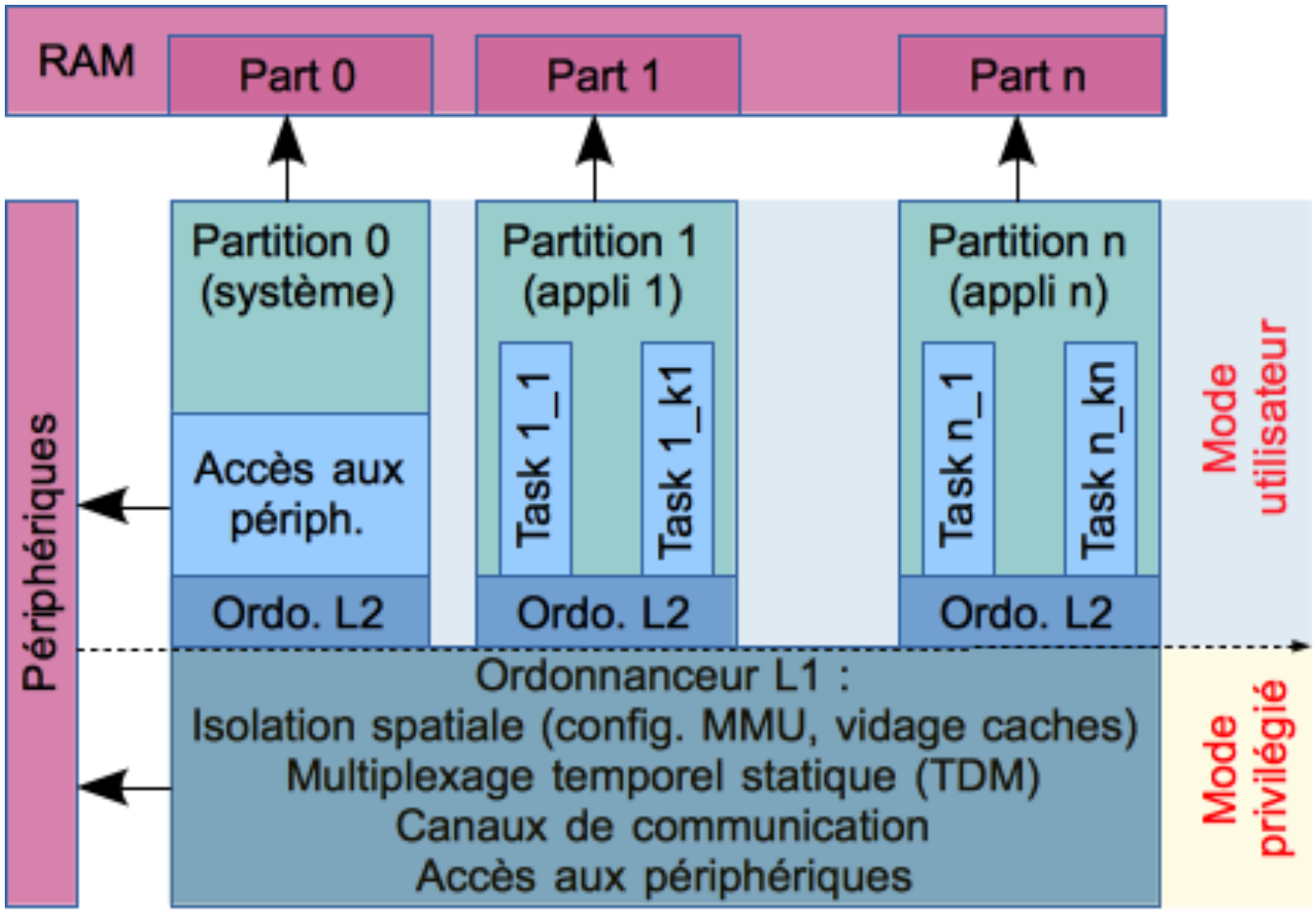
- Implantation partielle
  - Pas de Health Monitoring
  - Couverture partielle d'APEX
    - Seulement des ports "queuing" + simplifications
    - Communications point-à-point seulement
      - seul type de communication qui ne soit pas dépendant de l'implémentation
    - Simplification du fichier de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

RPi653

- Choix particuliers d'implantation
  - Partition système 0
    - Partie des pilotes de périphériques qui peut être exécutée en mode utilisateur
      - Pilote de console
    - Des fenêtres doivent lui être dédiées
    - Isolée par rapport au noyau L2
  - Extension d'APEX par un ensemble de services système
    - Console printing (facilite le débogage)

RPi653

- Choix particuliers d'implantation



Préparation du TP

- Sources rpi653-onepart.tar.gz
  - Une seule partition avec une seule tâche, code minimaliste, facile à comprendre et à modifier

- Objectif 1 : deux tâches dans une partition, ordonnées sous politique RM
  - Similaire à l'exemple des transparents 21-24
- Objectif 2 : deux tâches dans une partition, ordonnées sous politique RM, une des tâches démarée avec DELAYED_START
- Objectif 3 : deux partitions, chacune ayant une fenêtre dans le MTF et une tâche

- Démo

Objectif 1 : deux tâches

- Créer une copie de l'arborescence rpi653-onepart
  - Il contient une seule partition, nommée mypart
  - Le point d'entrée de la partition est la fonction main_process
    - Identifier les appels APEX qui créent et démarrent la tâche (processus APEX) nommée "t1"
    - Identifier la configuration de "t1" (struct de type PROCESS_ATTRIBUTE_TYPE)
      - Durées fournies en microsecondes => la période de t1 est de 0xf4240=1000000ms=1s
      - MTF de 10 ticks to 100ms chacun (1 tick = 100000=0x186A0)
  - Identifier la fonction-process de "t1"

Objectif 1 : deux tâches

- Créer et démarrer une nouvelle tâche t2
  - Création d'une fonction-tâche (par copie et modification des messages de la fonction-tâche t1)
  - Création de variables globales de types PROCESS_ATTRIBUTE_TYPE et PROCESS_ID_TYPE pour stocker attributs et PID
  - Attributs à utiliser
    - période : 500ms=0x7A120
    - capacité : égale à sa période
    - nom : t2
    - priorité : 2 (plus grande que celle de t1)
    - deadline et pile comme pour t1
  - Création de la tâche et vérification du fait que la création n'a pas donné d'erreur
  - Démarrage au point de référence temporelle (avec START)
- Seul fichier à modifier : arinc_main.c

Objectif 1 : deux tâches

- Compilation et traçage
  - Exemple

```
K 1:0system_partition_entry_point: init completed.
1 1:1   L2 L2_start: init completed, entering elaboration.
1 2:1   T1 Task t2. Instance 0
1 2:1   T0 Task t1. Instance 0
1 2:7   T1 Task t2. Instance 1
```

  - Organisation d'un message de console:
    - K (kernel ou partition système) ou identifiant de la partition applicative (ici, seulement 1)
    - #MTF:#tick (2:1 = le tick 1 du MTF 2)
    - L2 = ordonnanceur L2, T0 = tâche de PID 0
  - Pourquoi la tâche t2 s'exécute avant t1 à la date 2:1 ?

Objectif 2 : démarrage retardé

- Créer une copie de la hiérarchie de l'objectif 1
- Modification
  - Démarrage avec délai de 100ms=0x186A0 par rapport au point de référence temporelle (DELAYED_START)
- Seul fichier à modifier : arinc_main.c
- Expliquer le changement dans la trace produite
  - Pourquoi t1 s'exécute en premier ?

Objectif 3 : deux partitions

- Créer une copie de l'arborescence rpi653-onepart
- Ajouter une partition <nom>
  - Le nom doit avoir <= 8 caractères
- Modification de la configuration de l'ordonnanceur L1
  - Fichier output/config.pok
  - Modifier :
    - Augmenter de 1 le nombre de partitions
    - Rajout d'une partition dans la liste
      - Numéro = 2 (après les partitions existantes)
      - Fichier <nom>.elf
      - Taille mémoire allouée (0x100000, comme pour les autres partitions)
      - Période (égale au MTF, comme pour les autres partitions)
      - Sans ports (i.e. nombre de ports = 0)
    - Augmenter de 1 le nombre de fenêtres
    - Diviser la fenêtre 1 en 2 fenêtres plus petites
      - Attention, les barrières entre fenêtres doivent être des multiples du tick
      - Attribut PPS 1
      - Format d'une description de fenêtre :
        - » Identifiant de fenêtre, date de début, date de fin, partition, pps

Objectif 3 : deux partitions

- Créer un répertoire <nom> par copie de mypart
  - Modification du script de compilation de la partition
    - Changer le nom de partition de "mypart" en <nom>
  - Modification du fichier arinc_main.c
    - Changement de "mypart" en <nom> dans tous les messages.
    - Aucune autre modification (une seule tâche de la même période que celle de mypart)
- Modifier le script de compilation global
  - Pour chaque règle impliquant mypart, créer l'action équivalente pour la partition <nom>
- Recompiler (make clean; make)
- Exécuter