Utiliser les interruptions

- Précautions à prendre pour écrire une IT:
  - Boucles limitées
    - *for* : OK pour un faible nombre d'iterrations
    - *while* : Dangereux (on ne maitrise pas le nombre d'iterrations)
  - Fonctionnement par succession de tests unitaires
  - Toujours identifier la cause de l'interruption (pas de suppositions)
  - Pas d'attente (ralentissement système)
    - Pas de sous-fonction bloquante non plus
    - Compensé par la plannification d'un autre déclenchement IT au moment nécessaire...

Interruptions / Priorités

- Si plusieurs vecteurs d'IT, possibilité d'accorder différentes priorités.
  - Hiérarchie donnée à la configuration
    - Une interruption ne peut se déclencher que si le processeur éxécute une interruption de plus faible priorité

- Selon la série, la configuration des priorités est variable
  - Un vecteur/fonction par priorité, (low cost)
  - Priorité configurable par vecteur...

UART en interruption (réception)

- Il est possible de déclencher une IT à la réception d'un caractère…
  - Bien définir le rôle de l'interruption
    - Gérer le flux de données ?
    - Filtrer les données ?
    - Interpréter les données ?
  - Le plus propre est de rester neutre, on se contente de gérer le flux de données
    - l'interuption place le caractère reçu en file d'attente
    - Et c'est tout !

      (c'est précisément ce qui se passe dans un système informatique)
    - l'application utilisatrice des données va lire la file d'attente
      - La structure de la file doit rester cohérente

UART en interruption (envoi)

- Le déclenchement de l'IT se fait quand le canal d'envoi est libre
  - l'IT peut lire le prochain caractère à envoyer dans une FIFO
  - Le fait d'envoyer un nouveau caractère efface la demande d'IT
- s'il n'y a plus de caractères à envoyer :
  - La fifo d'envoi est vide
  - Pas d'envoi => pas de mise à zero de la demande d'IT
  - Il faut désactiver le déclenchement des ITs en envoi
  - On les réactivera lorsque l'utilisateur aura écrit de nouvelles données dans la fifo en envoi

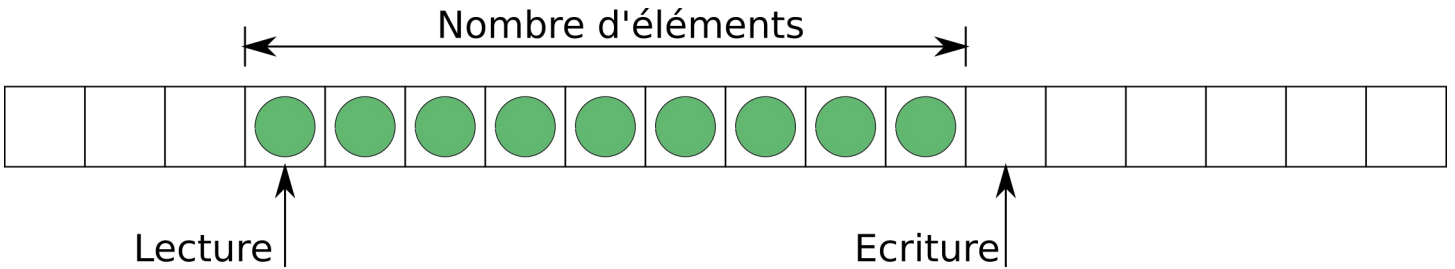
Bilan de l’UART en IT

- Les interruption combinées aux FIFOs permettent un fonctionnement plus fluide de l’application
- La fourniture de fonctions de configuration améliore encore la facilité d’usage pour l’utilisateur

=> on obtient un pilote de périphérique

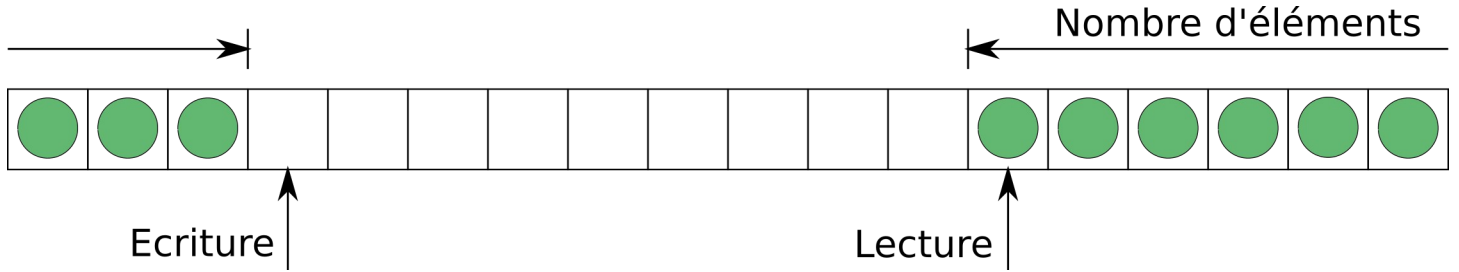
File d’attente

- Utilisation d’un tableau et :
  au choix (deux parmis trois)
  - d’un indice de lecture
  - d’un indice d’écriture
  - Du nombre d’éléments stockés



File d’attente : Warnings

- Problèmes de cohérence éventuels
- Gérer le côté circulaire



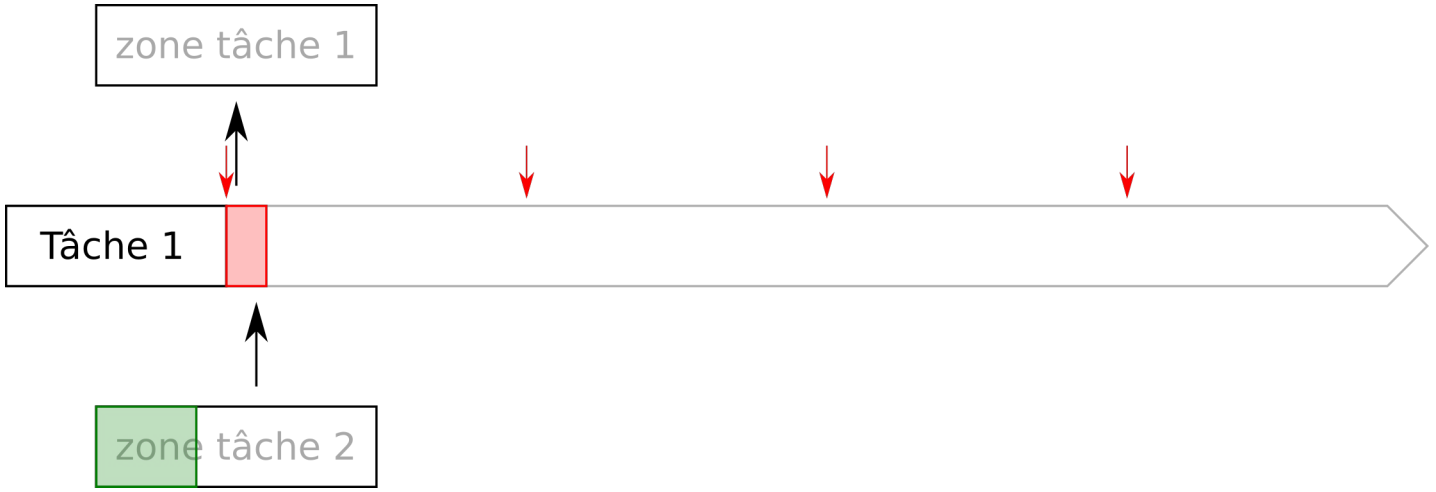
- ! ! ! Faire des schémas ! ! !

Réseau d’interruptions

- Système d’interruptions auto-entretenu:
  - Une fois le système configuré, on ne fait plus que réagir aux demandes d’interruption
  - Selon la configuration, les IT modifient leur propre niveau de fonctionnement.

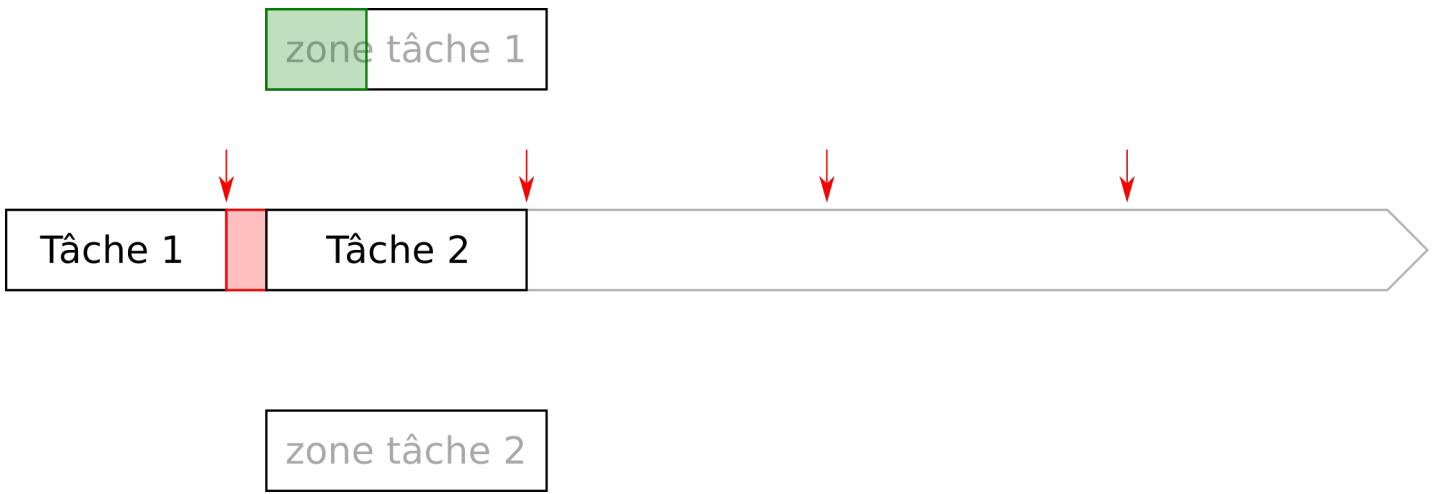
Réalisation d’un ordonnanceur

- Le principe est de ne pas restaurer le contexte qui a été sauvegardé (tâche 1), mais celui d’un autre programme (tâche 2)



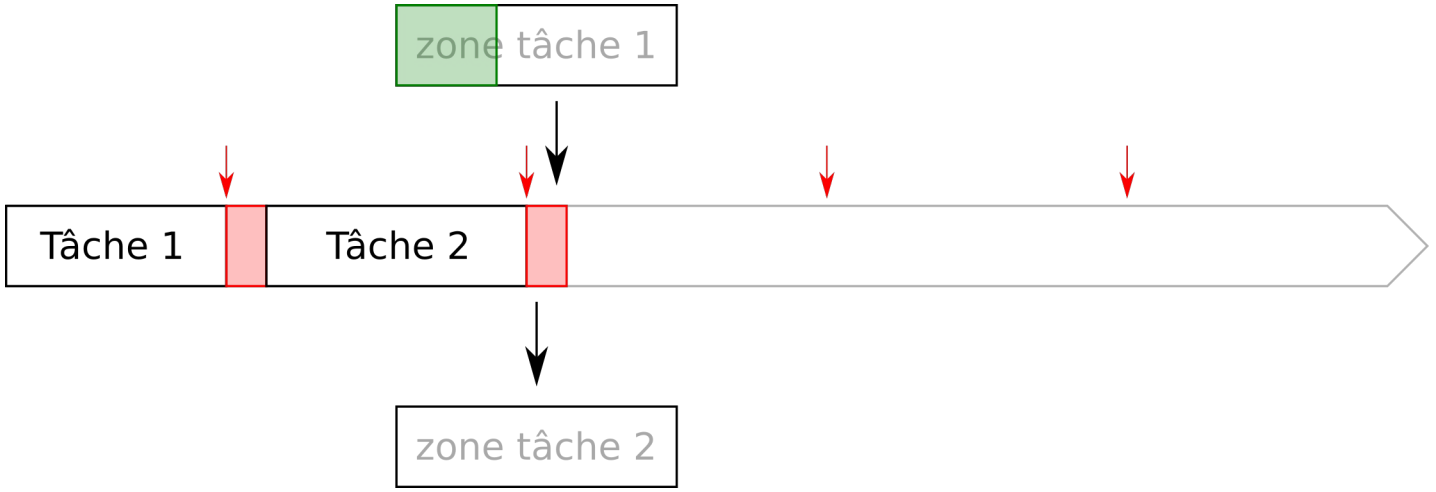
Réalisation d’un ordonnanceur

- Le principe est de ne pas restaurer le contexte qui a été sauvegardé (tâche 1), mais celui d’un autre programme (tâche 2)



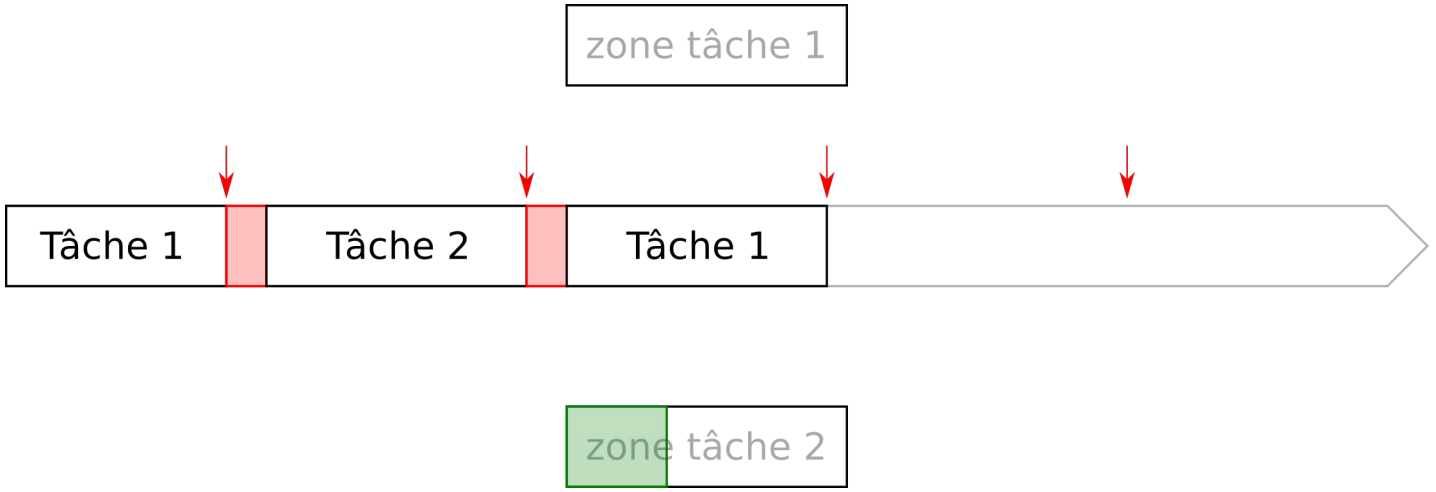
Réalisation d’un ordonnanceur

- Le principe est de ne pas restaurer le contexte qui a été sauvegardé (tâche 1), mais celui d’un autre programme (tâche 2)



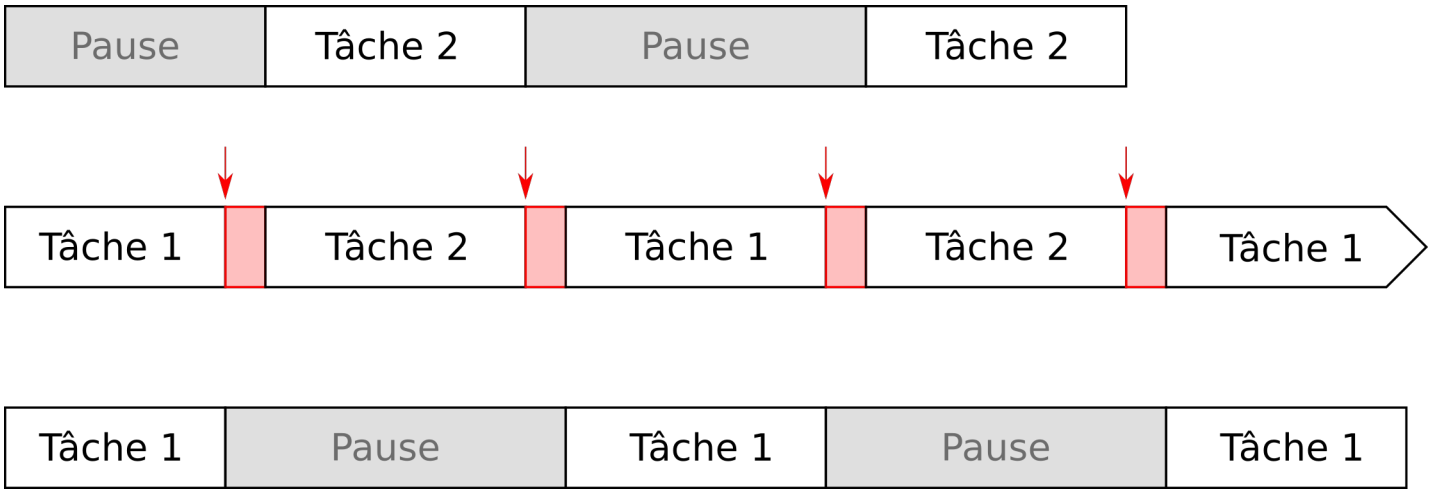
Réalisation d’un ordonnanceur

- Le principe est de ne pas restaurer le contexte qui a été sauvegardé (tâche 1), mais celui d’un autre programme (tâche 2)



Réalisation d’un ordonnanceur

- Chaque tâche est exécutée de façon linéaire avec des *pauses* régulières



Échanges d’informations :

- Une interruption
  - ne reçoit pas d’argument (déclenchement suite à un évènement matériel),
  - ne renvoie pas de résultat (déclenchement asynchrone)
- Les entrées d’une interruption sont :
  - l’état matériel du système
  - Les variables globales (état logiciel du système)
- Les sorties d’une IT sont:
  - Changement d’état materiel (traitement de l’évènement)
  - Changement d’état logiciel (mise à jour des variables globales)

=> L’usage des interruption ne se fait que par des variables globales

PB : intégrité des structures de données

- Une structure de données complexe ne peut pas être cohérente en permanence.
  - Ex théorique:

    On ne peut pas modifier en même temps *A* et les cases représentant les totaux…

A	B	C	A+B+C
D	E	F	D+E+F
G	H	I	G+H+I
A+D+G	B+E+H	C+F+I	Total

La structure est dite non cohérente entre la première modification et la fin des mises à jour qui en découlent...

PB : intégrité des structures de données

- Que se passe-t-il si une IT se déclenche à un moment ou la structure de données n'est pas cohérente ?
  - Si l'IT ne veut que lire la structure selon la structure et les algorithmes :
    - Aucun problème
    - Problème ponctuel (arctéfact / anomalie)
    - Problème cumulatif (perte de fonctionalité)
      - Éventuellement récupérable si inclus dans une régulation
    - Boucle infinie / exception (erreur fatale (grave))
      - Système instable / inutilisable
      - Besoin d'un chien de garde
  - Dans tous les cas : Débogage extrèmement complexe (difficile de reproduire les conditions à l'identique)

PB : intégrité des structures de données

- Si l'IT modifie la structure de données:

A'	B	C	A+B+C
D	E	F	D+E+F
G	H	I	G+H+I
A'+D+G	B+E+H	C+F+I	Total

A'	B	C'	A+B+C'
D	E	F	D+E+F
G	H	I	G+H+I
A'+D+G	B+E+H	C'+F+I	**Total**

Selon la méthode de calcul du total :

- Par la dernière ligne : résultat correct
- Par la dernière colonne : résultat faux
- Par ajout de C'-C : résultat faux

PB : intégrité des structures de données

- Si l'IT modifie la structure de données:

A'	B	C	A+B+C
D	E	F	D+E+F
G	H	I	G+H+I
A'+D+G	B+E+H	C+F+I	Total

A'	B	C'	A+B+C'
D	E	F	D+E+F
G	H	I	G+H+I
A'+D+G	B+E+H	C'+F+I	**Total**

A'	B	C'	A'+B+C
D	E	F	D+E+F
G	H	I	G+H+I
A'+D+G	B+E+H	C'+F+I	**Total**

Selon la méthode de calcul du total :

- Par la dernière ligne : résultat correct
- Par la dernière colonne : résultat faux
- Par ajout de C'-C : résultat faux

Au retour de l'IT :

- Le tableau continuera à être mis à jour…
  - Des bonnes valeurs seront écrasées
  - Encore moins de chances que le total soit juste

Résoudre les problèmes de cohérence :

- Si présence d'un système d'exploitation :
  - Usage de mutex (verrous de protection)
- Si pas de système d'exploitation :
  - Interdire temporairement l'usage des interruptions
  - Ne doit être réalisé
    - QUE le temps de mise à jour de la structure
    - QUE sur les interruptions succeptibles d'accéder à la structure
  - La structure de données doit être pensée pour être mise a jour RAPIDEMENT !

Où rencontre-t-on les problèmes de cohérence ?

- Structures triées
- Structures chainées
  - Si des opérations complexes sont effectuées
  - Si les chainages sont multiples
- Systèmes de fichiers

- Toute structure de données pour laquelle les modifications ne sont pas atomiques
  - Au hasard : file d'attente

Petit calcul statistique

Temps de mise à jour la structure de données : environ 100ns

Fréquence d'interruption : 500Hz (période 2 ms)

- Probabilité que tout se passe bien au déclenchement d'une IT:

Petit calcul de probabilités

Temps de mise à jour la structure de données : environ 100ns

Fréquence d'interruption : 500Hz (période 2 ms)

- Probabilité que tout se passe bien au déclenchement d'une IT:

  (2 ms – 100 ns) / 2ms = 99,995%
- Probabilité que tout se passe bien pendant 1s:

  99,995% ^ 500 = 97,53%
- Pendant 1 mn:

  97,53% ^ 60 = 22,31 %
- Pendant 5 mn:

  22,31% ^ 5 = 0,06 %

PB : partage de fonctions / réentrance

- Si une fonction est appelée par une Interruption, elle doit obéir aux même règles que la fonction d'IT

- Si une fonction est appelée par plusieurs interruptions (ou par une IT et le prog principal), elle doit être réentrante:

  - Pas d'usage de variables statiques/globales non protégées

    → facile sur un processeur avec pile intégrée

    → sur processeurs plus petits : le compilateur galère !!!

    - La fonction est souvent dupliquée pour éviter les problèmes
    - Augmentation de l'empreinte mémoire et de la taille du code