Conception d'interfaces pour système à processeur.

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

Etienne Messerli

Février 2019

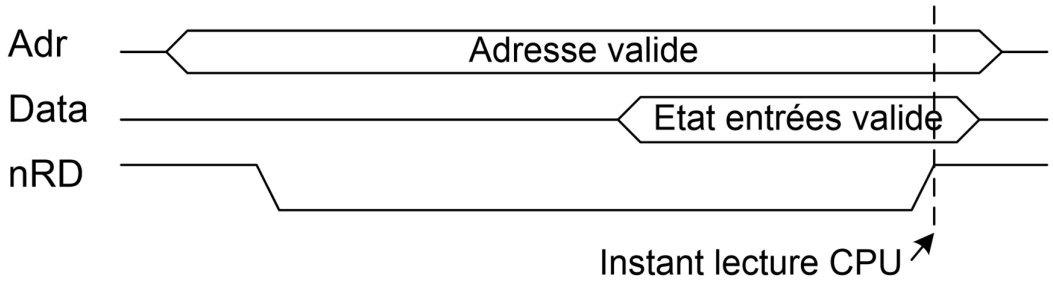
Les entrées/sorties, I/Os

- Actions du CPU sur les I/Os réalisées via des instructions assembleurs, soit:
  - Lecture d'entrée (in): instruction LDR
  - Ecriture de l'état de sortie (out): instruction STR
- Accès aux I/Os via le bus du système à processeur
- Accès multiplexé aux I/Os !
- Permet d'adresser un grand nombre de I/O

Cycle du CPU: lecture In

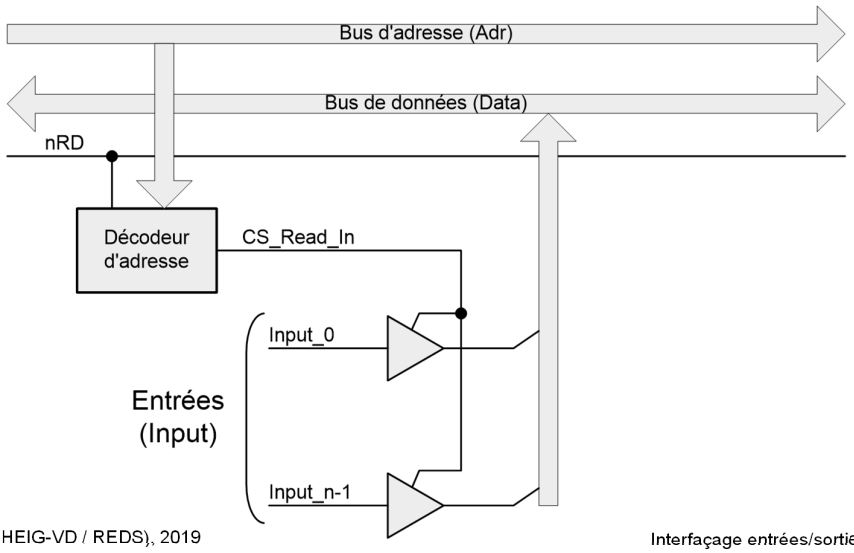
Acquisition des entrées via un cycle de lecture CPU:

- ARM: LDR Rd, [Rn]
  Rd registre destination
  Rn registre avec adresse entrées
- Chaque adresse permet de lire au maximum N entrées
  N = nombre de bits du bus de data



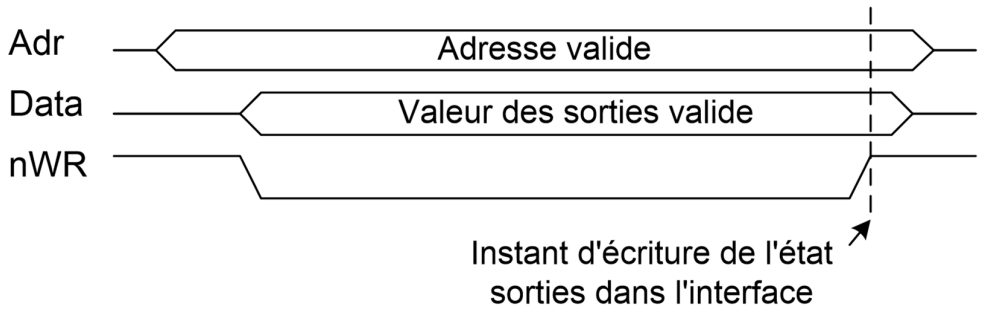
Interface pour entrées (In)

- Schéma de base pour accéder aux entrées
  - Entrées connectées sur le bus de données via des portes 3 états
  - Portes 3 états activées lors d'un accès en lecture à l'adresse correspondante aux entrées



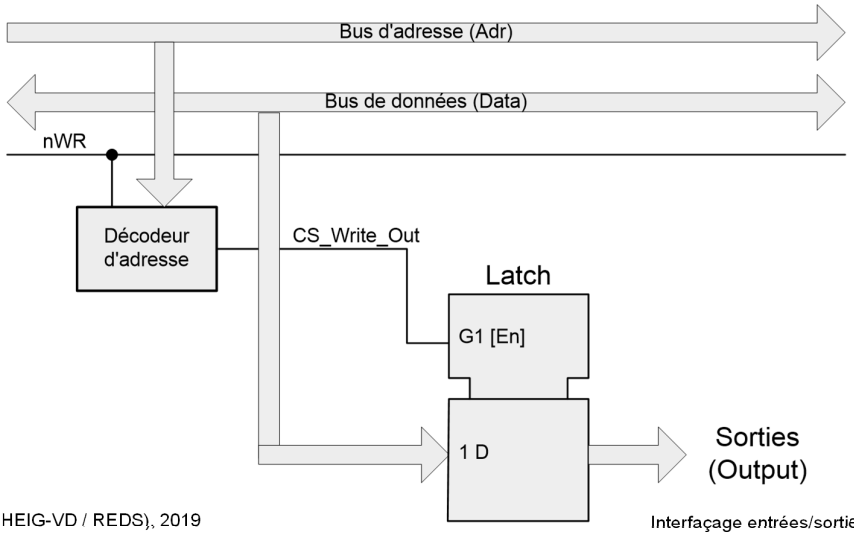
Cycle du CPU: écriture Out

- Ecriture des sorties via un cycle d'écriture CPU
  - ARM: STR Rd, [Rn] Rd registre source
    Rn registre avec adresse sorties
  - Chaque adresse permet l'écriture d'au maximum N sorties
    N = nombre de bits du bus de data



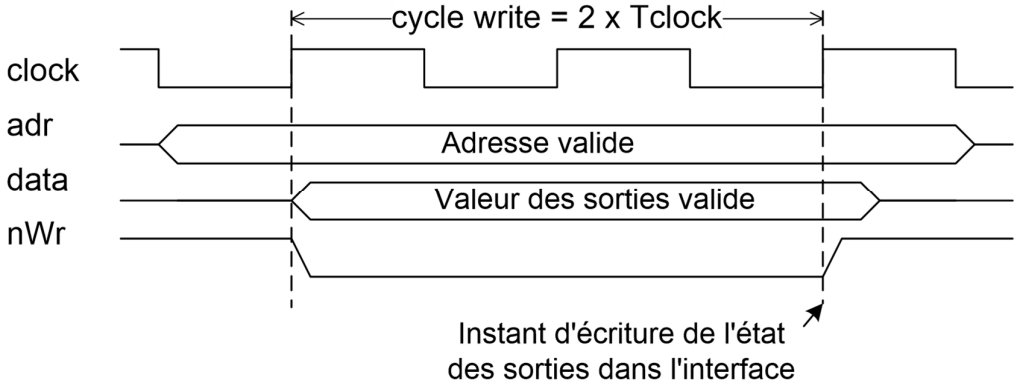
Interface pour sorties (Out)

- Schéma de base pour accéder aux sorties
  - information valide sur le bus de données **uniquement** durant le cycle d'écriture (write)
  - nécessaire de **mémoriser** l'état des sorties



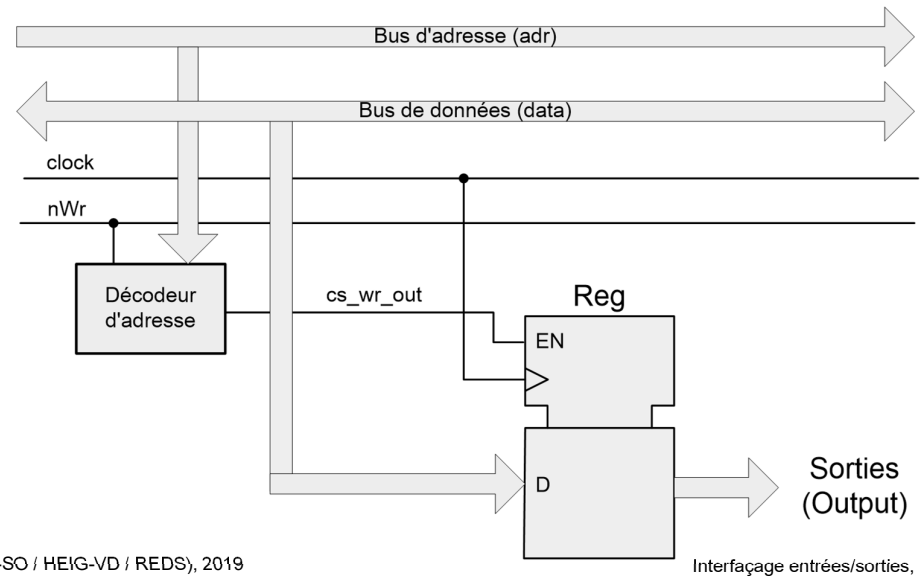
Interface synchrone

- Fréquemment le bus du système à processeur dispose d'une horloge.
- On parle de bus "synchrone"
- Exemple de cycle d'écriture synchrone sur une interface de sorties



Interface pour sorties (Out)

- Schéma de base pour l'écrire des sorties sur un bus synchrone
  - **mémorisation** de l'état des sorties au flanc actif de l'horloge (exemple: flanc montant)



Exemples d'interface

Voir donnée séparée: Exemple 1

- Réaliser une interface pour 16 entrées et 16 sorties

puis traiter les variantes:

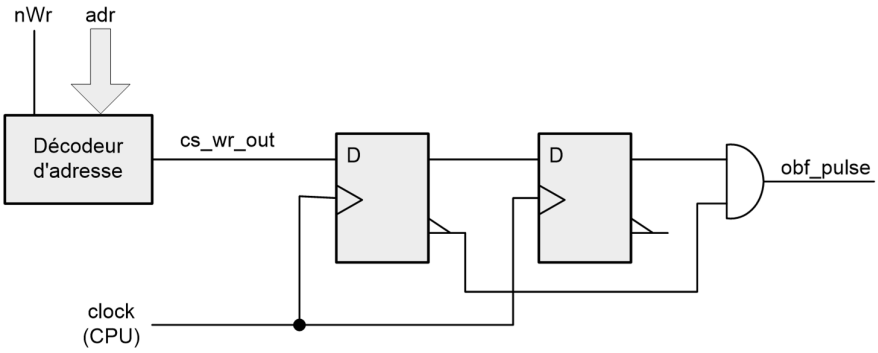
a) Relecture de l'états des sorties
b) Adapter l'interface pour 32 sorties
c) Adapter l'interface pour 24 entrées

Indication changement état I/O

- Les interfaces décrites précédemment n'ont pas de fil d'asservissement
  - il n'y a aucune information sur l'évolution de l'état des entrées ou des sorties
- Il est souvent utile, voir nécessaire, d'indiquer au CPU (programmeur) le changement d'état des entrées/sorties
  - on parle d'interface avec fil d'asservissement ou gestion de flux

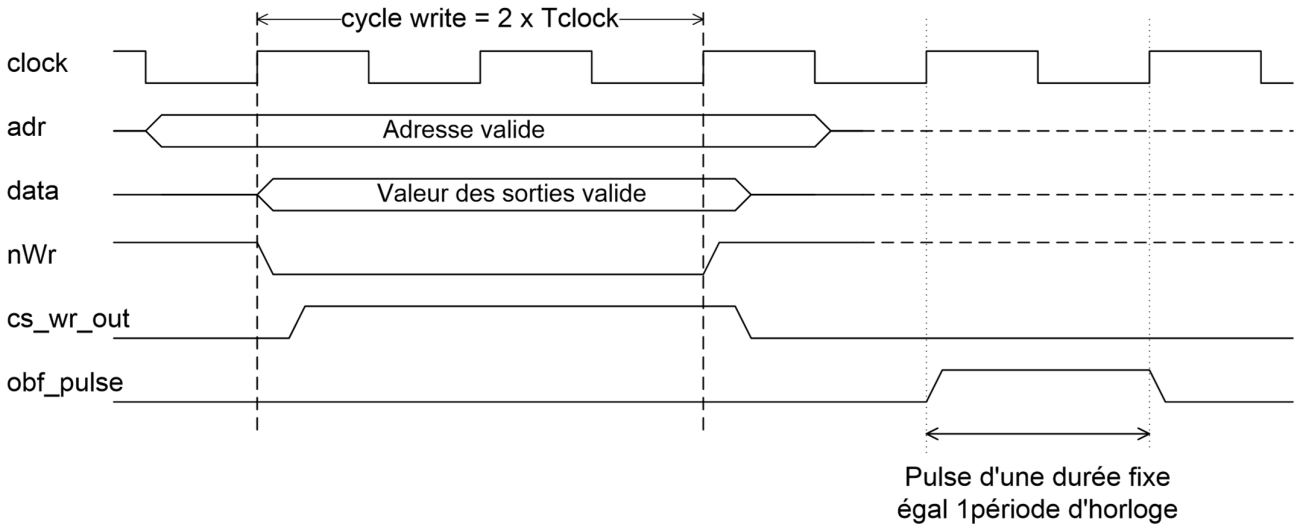
I/O avec un fil d'asservissement

- Nous souhaitons informer le périphérique que l'état de sortie a été mis à jour
- Génération d'une impulsion sur une sortie OBF (Output Buffer Full)



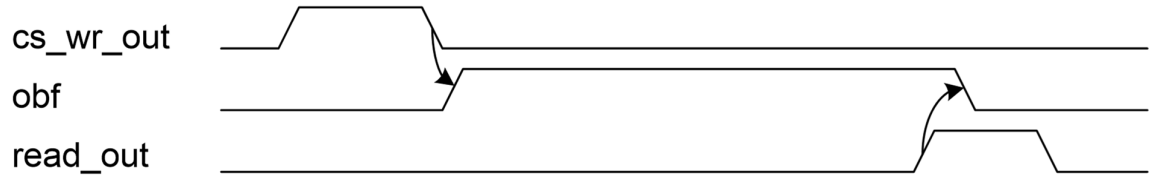
I/O avec un fil d'asservissement

- Chronogramme du déroulement:



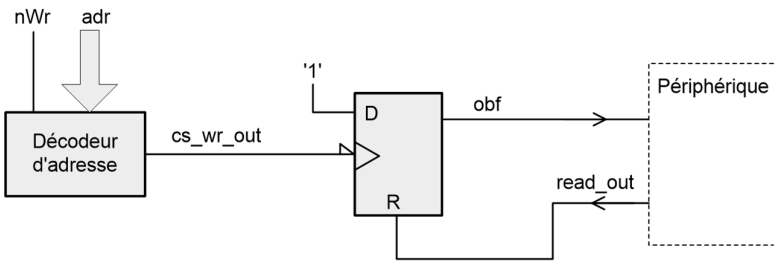
I/O avec un fil d'asservissement

- Gestion de flux à un seul fil pas optimum
  - Risque de ne pas voir l'impulsion
- Solution: gestion de flux avec deux fils d'asservissement
  - plus fiable, car permet des timings différents pour chaque élément (acteur)



I/O avec deux fils d'asservissement

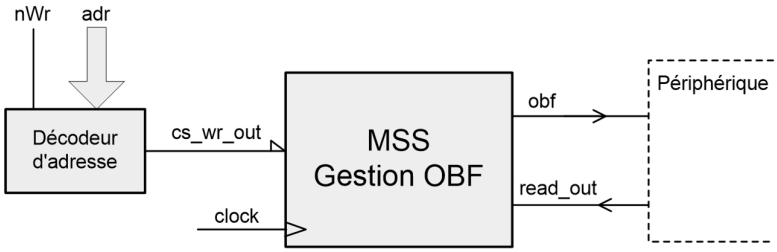
- Voici une proposition de schéma pour une sortie avec deux fils d'asservissement:



Solution asynchrone!
Le signal read_out **doit être sans aléas** !
Il est préférable d'utiliser un solution synchrone basée sur une machine d'états

I/O avec deux fils d'asservissement

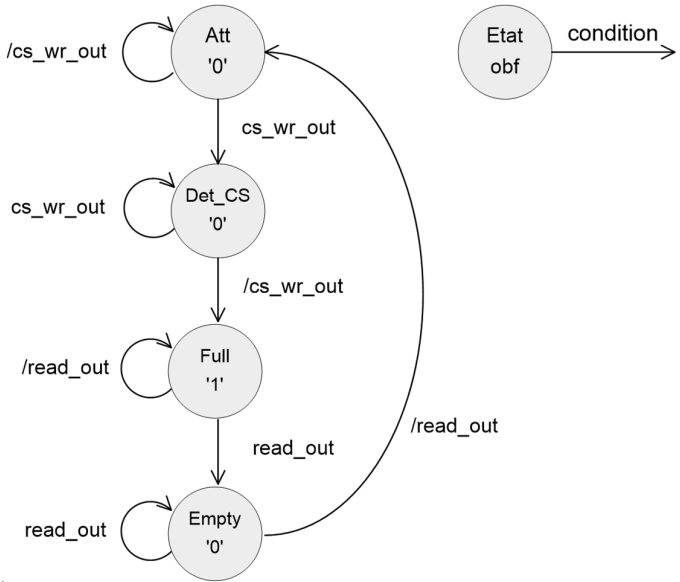
- Solution synchrone avec une machine d'états recommandée (MSS simple). Voici le schéma bloc:



La solution par machine d'état permet une grande souplesse dans la gestion de flux. Les modifications sont facilement réalisable

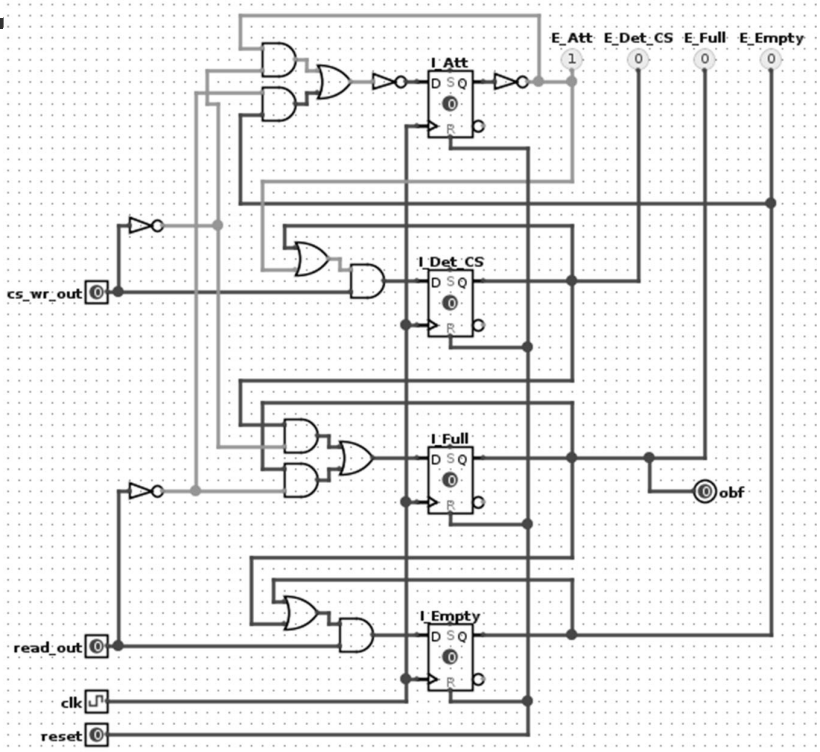
I/O avec deux fils d'asservissement

- Graphe des états de la solution synchrone avec une machine d'états (MSS simple)



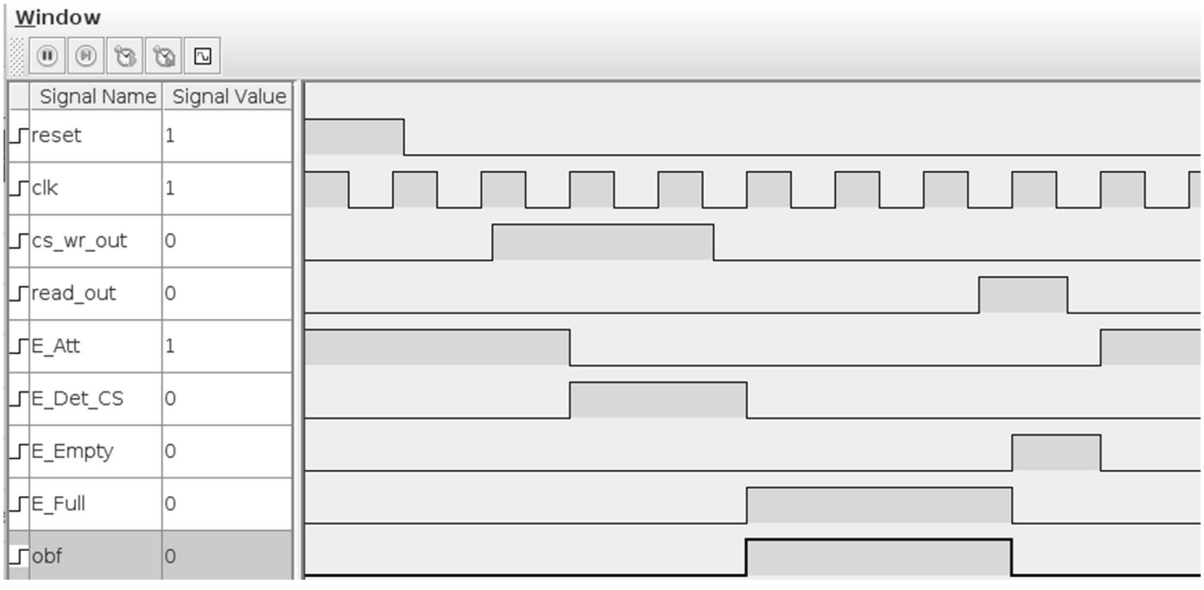
I/O avec deux fils d'asservissement

- Schéma de la MSS



I/O avec deux fils d'asservissement

- Simulation de la MSS

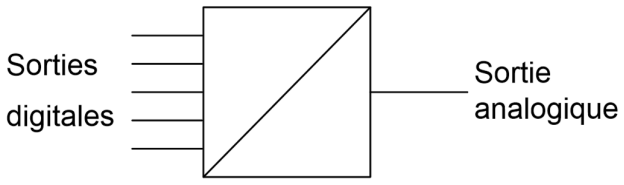


Autres types entrées/sorties …

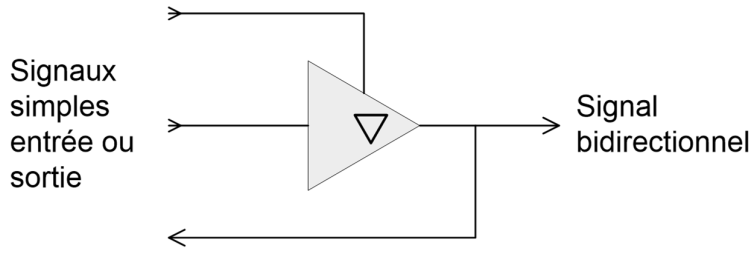
- Il existe de nombreux types d'entrées/sorties
  - ES simple digitales, écran, clavier, liaison série, capteurs délivrant des signaux analogiques (tension, courant, …), signaux bidirectionnels, …
- Un système à processeur peut uniquement lire ou écrire des entrées/sorties digitales
  - Il sera nécessaire de réaliser une interface entre les périphériques et le système à processeur.
  - Pour les signaux analogique l'utilisation de convertisseurs sera nécessaire pour transformer les signaux analogiques et signaux digitaux.

… autres types entrées/sorties …

- Signaux analogiques:



- Signal bidirectionnel



… autres types entrées/sorties

- Tous les périphériques seront vu comme un ensemble d'entrées/sorties simples ou vecteurs.
- La fonctionnalité sera définie par les interactions et les séquences entre ces E/S
- Nécessite souvent une gestion de flux (fils d'asservissement)

Exemples d'interface

Voir donnée séparée: Exemple 1

suite de variantes:

d) Implémenter un contrôle de flux pour les entrées avec un registre d'état dans l'interface.
un signal new indique la mise à jour des entrées

Timing du local bus REPTAR

Cycle de lecture (read)

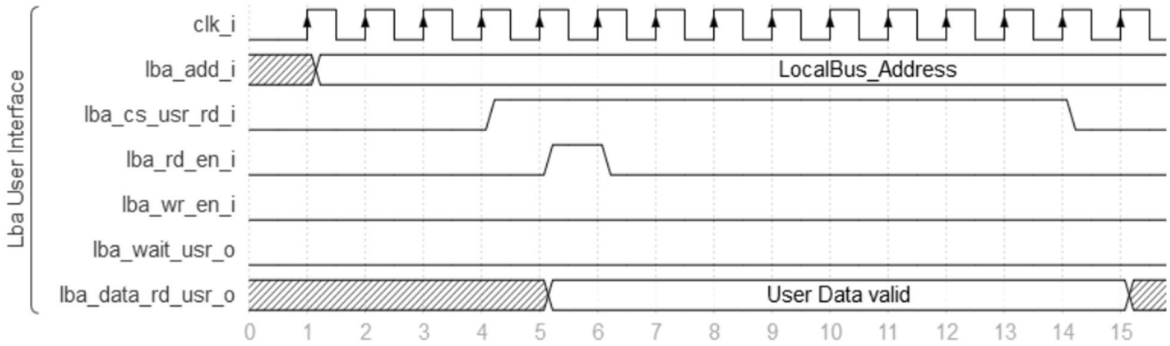


Figure 24 – Local Bus Asynchronous User Interface timing for read access without wait

Expliquez la fonction du signal lba_rd_en_i ?

Timing du local bus REPTAR

Cycle d'écriture (write)

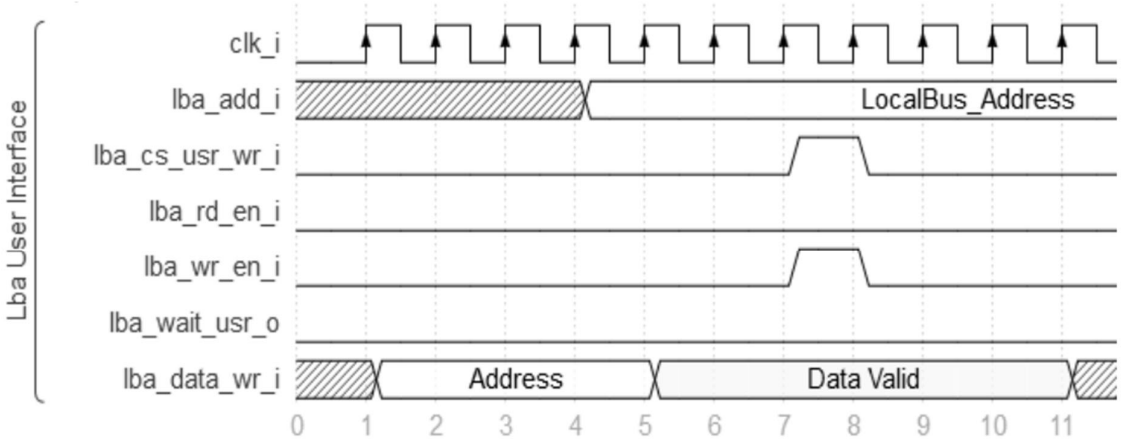


Figure 25 – Local Bus Asynchronous User Interface timing for write access

Expliquez pourquoi le signal lba_cs_usr_wr_i dure une seule période?

Gestion des entrées/sorties (I/O)

- Il y plusieurs méthodes pour gérer les I/O
  - Scrutation
  - Interruption
  - Accès direct en mémoire DMA
    en anglais: Direct Memory Access

- Critères de choix de la méthode de gestion:
  - quantité de matériel
  - temps de réaction (latence)
  - taux d'occupation du CPU

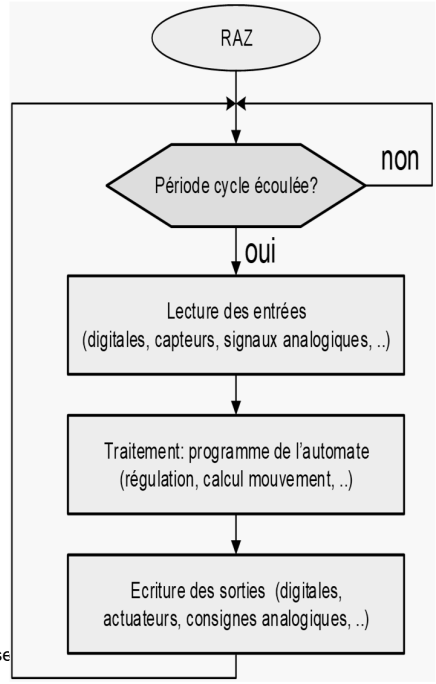
Gestion I/O par scrutation

- Avantages et inconvénients de la scrutation:
  - Pas besoin de matériel spécifique +
  - Utilise du temps CPU (ralenti progr.) -
  - Temps de réponse variable -

- Cas d'applications:
  - Evolution régulière des IO
    - scrutation répond parfaitement
  - Processeur peu chargé, ou peu de tâches en //

Gestion par scrutation I/O

Exemple: Automates programmable industriel



Temps du cycle de régulation:

- Pour relancer la séquence à temps constant, il y a un générateur de cycle.
- Il faut garantir que la boucle prend toujours un temps < que celui du cycle de régulation!

Gestion I/O par interruption

- Avantages et inconvénients des interruptions:
  - Nécessite du matériel spécifique -
  - Implique un changement de contexte du CPU (sous-progr.) -
  - Permet au CPU d'exécuter d'autres taches +
- Cas d'applications:
  - Changement peu fréquente IO, mais nécessitant une réaction rapide (timing prédictible)
  - Processeur ayant de multiple tâches
    - processeur fortement chargé

Gestion I/O par DMA

- DMA pour "Direct Memory Acces"
- Avantages et inconvénients DMA:
  - Nécessite beaucoup de matériel spécifique (contrôleur DMA => maitre sur le bus) -
  - N'utilise pas de temps CPU +
  - Le CPU doit libérer le bus -
- Cas d'application:
  - Transfert de volume important de donnée
    - Chargement des programmes en mémoire
  - Transfert avec un débit élevé

dia volontairement laissé vide

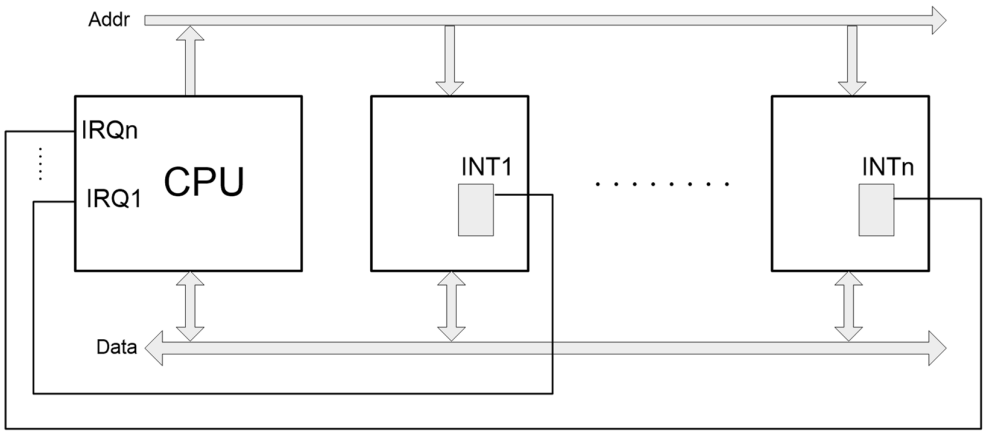
Gestion par interruptions

Contenu:

- Lignes d'interruption séparées jusqu'au CPU
- Une seul ligne d'interruption au CPU
  - Gestion des demande d'interruptions: scrutation, vectorisation, diasy chain (priorité)
- Contrôleur d'interruption

Gestion des interruptions …

- Lignes d'interruption séparées :



… gestion des interruptions …

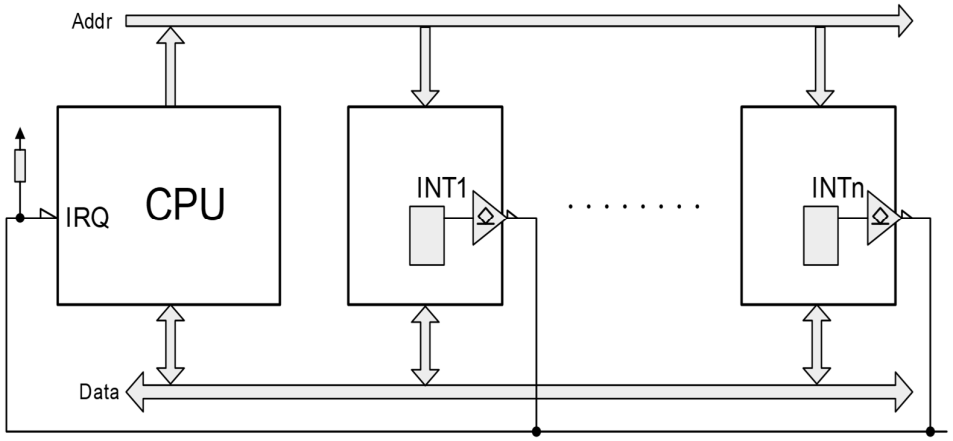
Lignes d'interruption séparées :

- Avantages
  - Efficace, chaque interface a sa propre IRQ sur le CPU
- Inconvénients
  - Beaucoup de lignes d'interruption sur le CPU
    - nécessite de nombreuses pins sur le circuit CPU
  - Beaucoup de logique dans le CPU
  - Extension impossible, solution non modulaire

- Cette solution n'est jamais utilisée en pratique

… gestion des interruptions …

- Lignes d'interruption groupées via des portes à collecteurs ouverts :



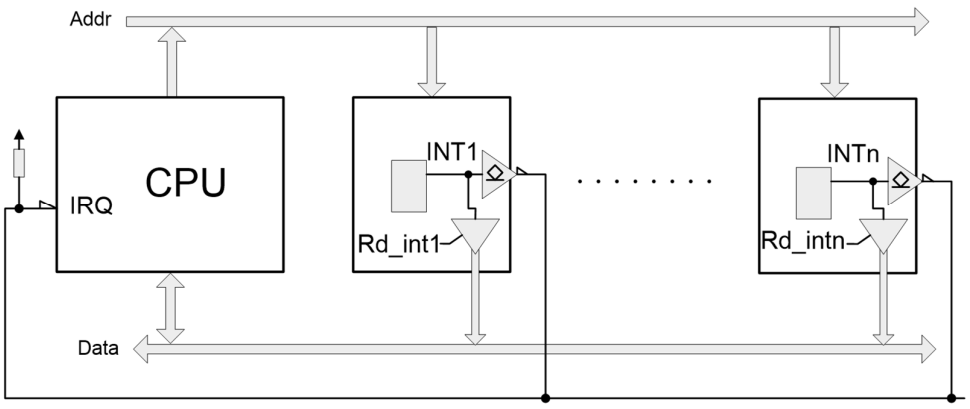
… gestion des interruptions …

Lignes d'interruption groupées via des portes à collecteurs ouverts :

- Avantages
  - câblage simple
  - facilement extensible (pas de limite)
- Inconvénients
  - Il faut déterminer QUI a fait la demande d'interruption.
  - Définir un ordre de priorité
- Plusieurs solutions pour déterminer la source:
  - Scrutation ou vectorisation

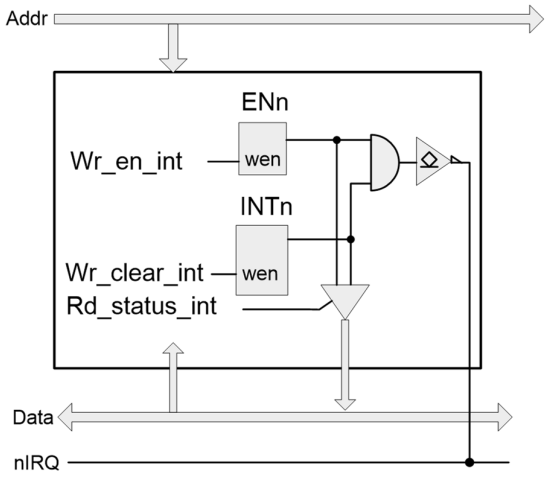
Interruption par scrutation

- Un flag indique s'il y a une demande d'interruption
- Celui-ci peut être lu par le CPU
- L'ordre de lecture détermine la priorité



Interruption par scrutation

- Fonctionnalités supplémentaires permettant la gestion des phases de l'interruption:
  - Lecture statut
    - état demande d'interruption
    - état du masque d'interruption
  - Masque d'interruption
    - permet de bloquer la demande d'interruption
  - Quittance de l'interruption
    - efface la demande d'interruption



Exemples d'interface

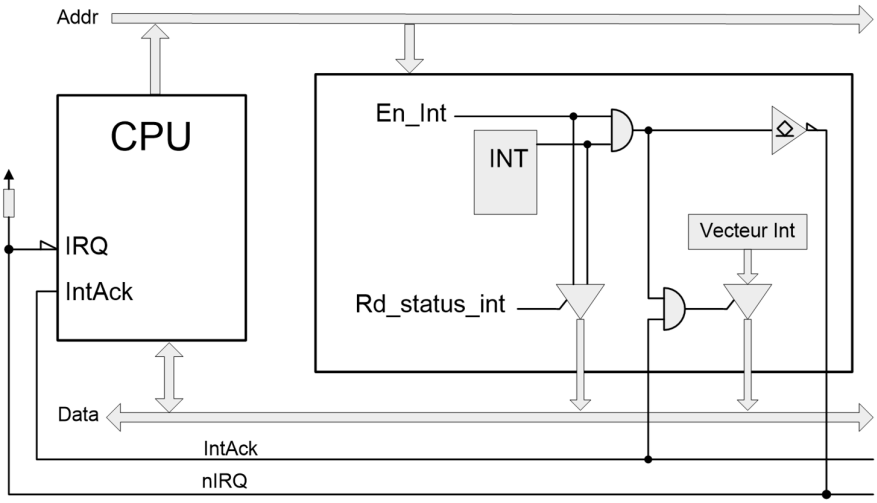
Voir donnée séparée: Exemple 1

suite de variantes:

e) Générer une interruption lorsqu'une nouvelle information est présente sur les entrées (activation du signal New)

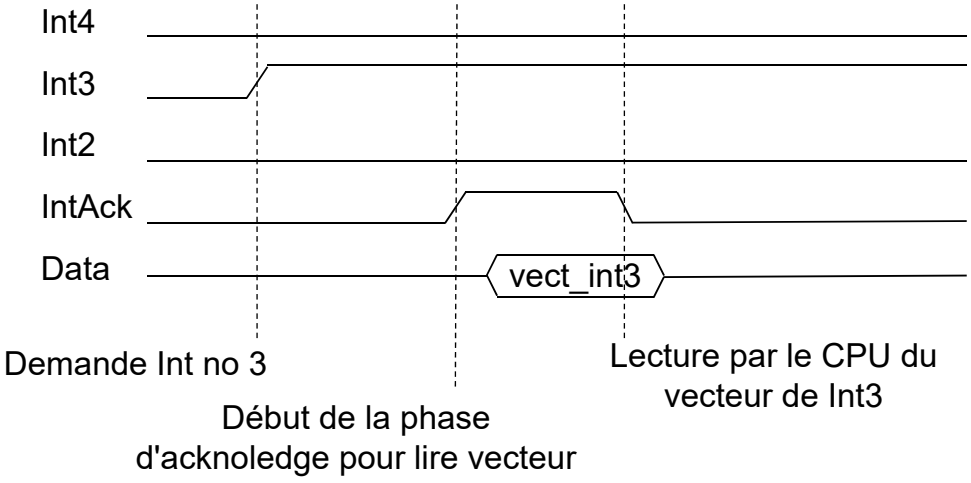
Interruption par vectorisation …

- La source de l'interruption est indiquée par un vecteur placé sur le bus de donnée lors de la phase d'acquittement de l'interruption (IntAck)



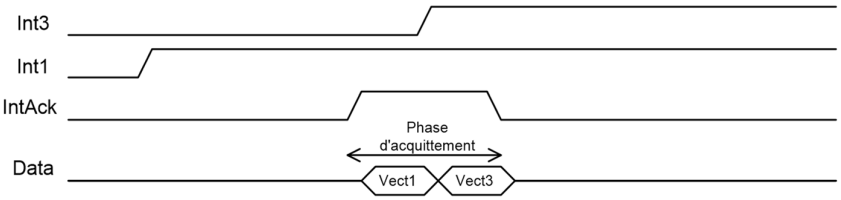
… interruption par vectorisation …

- Exemple de chronogramme du traitement d'une interruption avec vectorisation



… interruption par vectorisation…

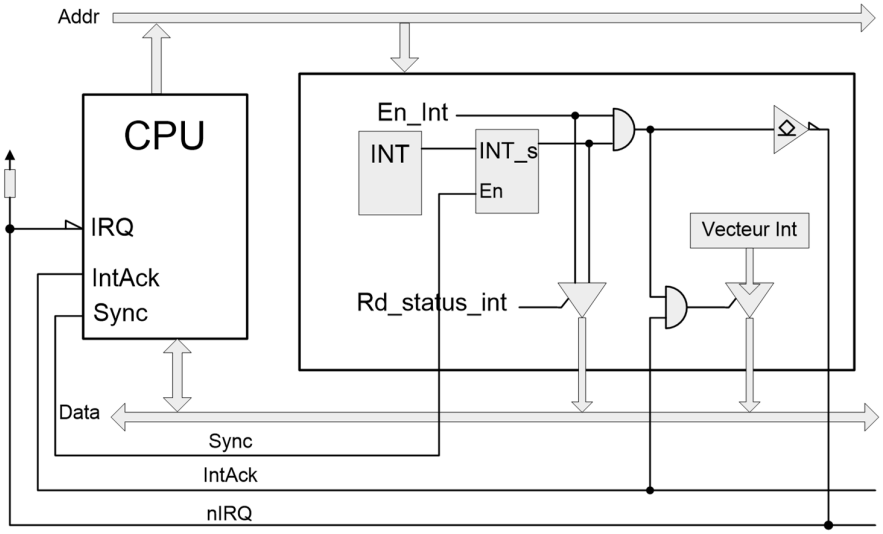
- Attention à l'activation asynchrone des demandes d'interruption
- Possible qu'une interruption plus prioritaire s'active pendant la phase d'acknowledge (IntAck: acquittement de la demande)



Dès lors, il y a un risque de lire le mauvais numéro de vecteur, voir solution page suivante.

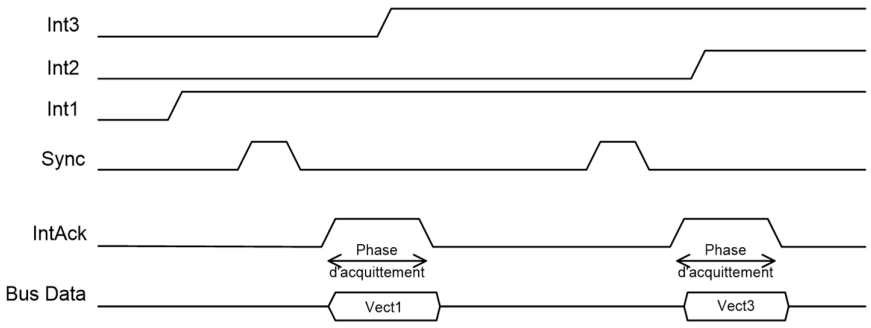
… interruption par vectorisation…

- Synchronisation des demandes d'interruption
  - Le signal Sync autorise l'acceptation de la demande d'interruption dans l'interface.



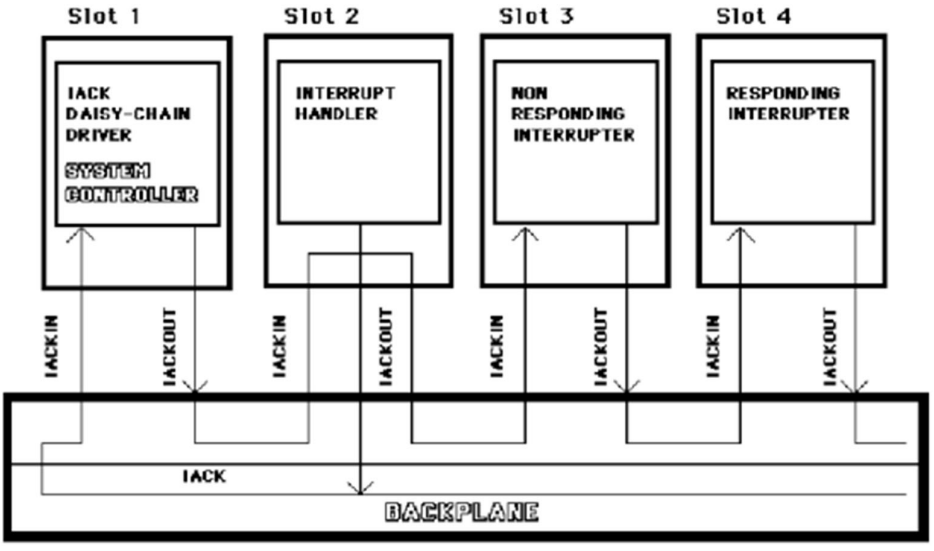
… interruption par vectorisation…

- Timing de la synchronisation des demandes d'interruption



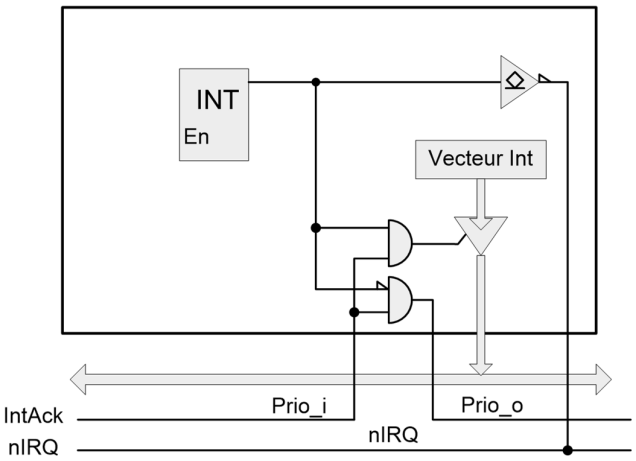
… interruption par vectorisation…

- Vectorisation avec chaine de priorité (daisy chain)



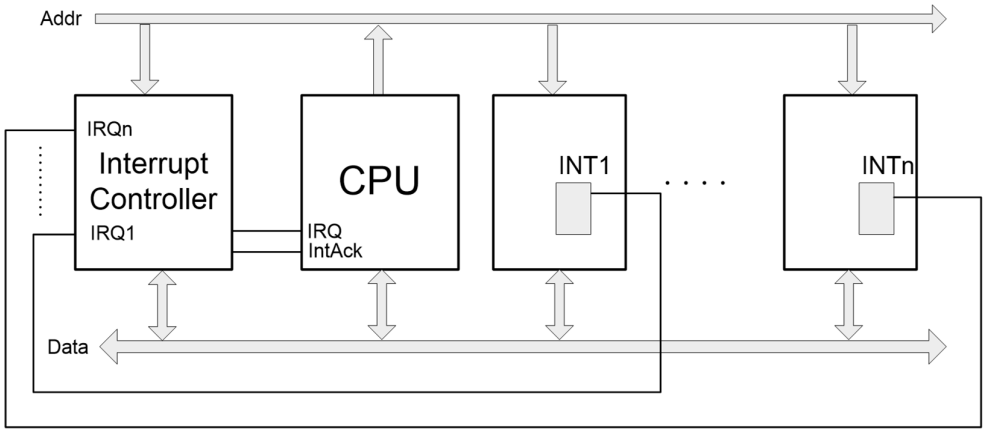
… interruption par vectorisation

- Priorité définie par une chaine de priorité:



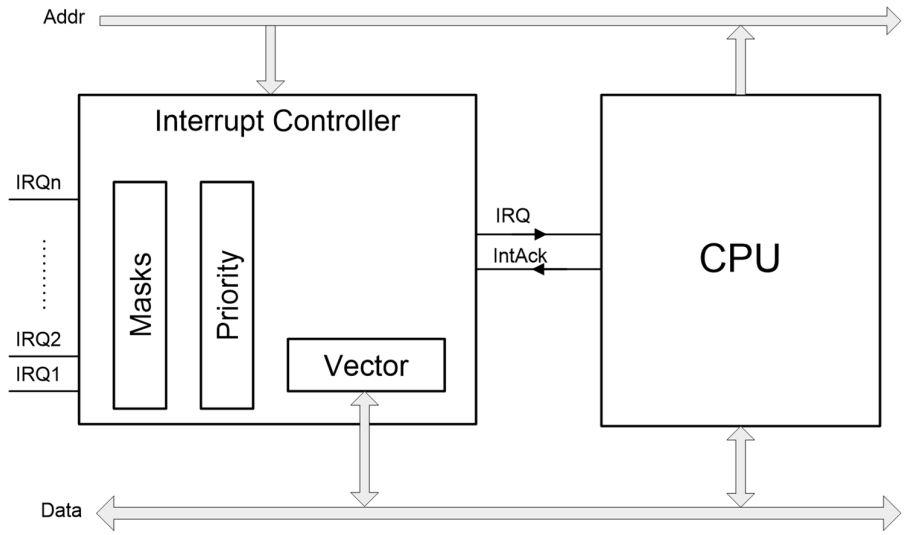
Contrôleur d'interruptions …

- Utilisation d'un contrôleur d'interruption qui gère les masques, les priorités et l'état des demandes pour le CPU



… contrôleur d'interruptions …

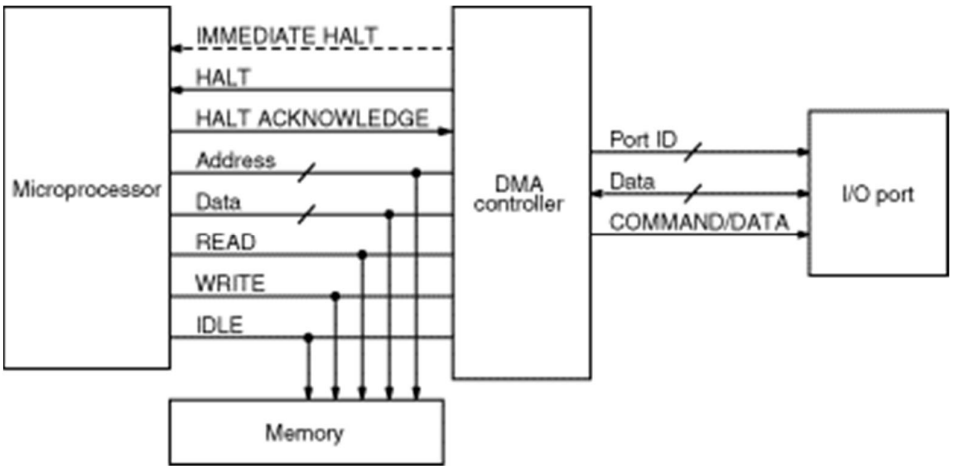
- Structure d'un contrôleur d'interruption:



dia volontairement laissé vide

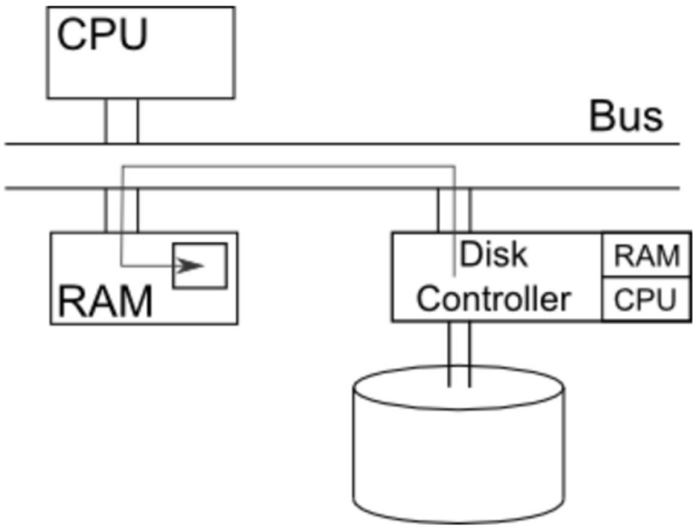
Contrôleur DMA …

- DMA: Direct memory access
  - Transfert direct entre de donnée d'un périphérique vers/depuis la mémoire
- Contrôleur attaché à un port IO



… contrôleur DMA …

- Contrôleur pour un disque dur



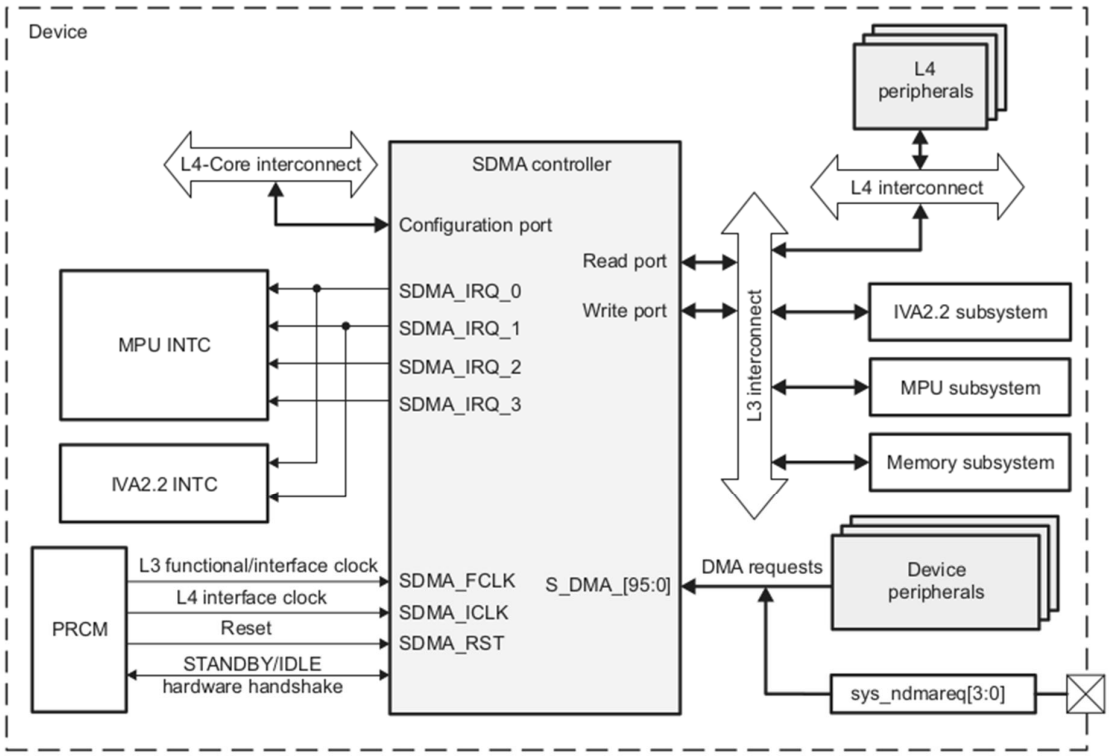
src: http://www.cs.ucla.edu/classes/fall10/cs111/srcibe/3a/

Contrôleur DMA REPTAR

- Exemple contrôleur DMA module DM3730
  - SDMA: system direct memory access
  - Caractéristiques :
    - Data transfer support in either direction between:
      - Memory and memory
      - Memory and peripheral device
    - 32 logical DMA channels supporting:
      - 8-bit, 16-bit, or 32-bit data element transfer size
    - First-come, first-serve DMA scheduling with fixed priority
    - Up to 96 DMA requests
    - Four programmable interrupt request output lines
    - FIFO depth: 256 x 32-bits

SDMA module DM3730 …

Figure 11-1. SDMA Overview

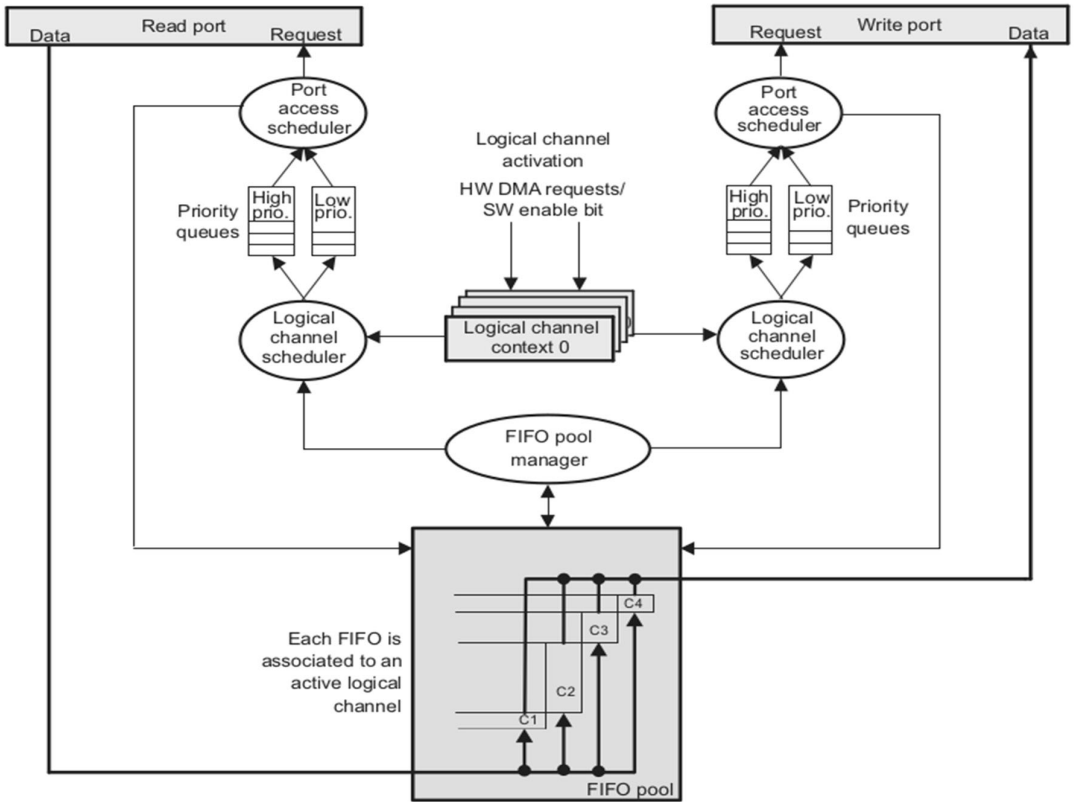


… SDMA module DM3730 …

- Description de la structure:
  - The SDMA module has three ports
    - one read, one write for data transfer
    - one configuration port and provides
  - Provides multiple logical channel support.
  - Dynamically allocated FIFO queue memory pool provides buffering between the read and write ports.
  - Read and write ports are multithreaded

… SDMA module DM3730 …

Figure 11-6. SDMA Controller Top-Level Block Diagram



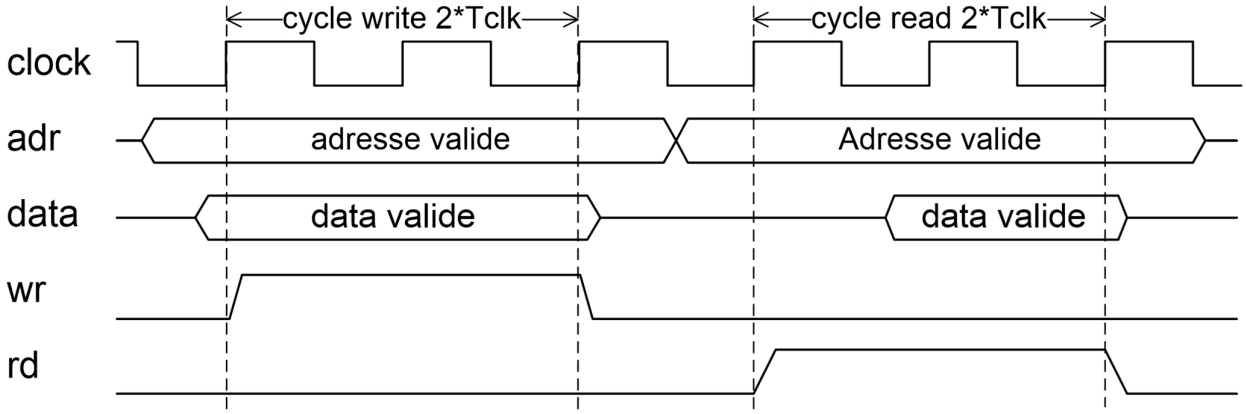
Bus et types de transfert

- Transfert synchrone
  - rythme imposé par le maître
    - fréquemment timing référencé sur le Clock du CPU
  - l'esclave doit suivre
  - pas de ligne de synchronisation
- Transfert asynchrone
  - lignes de synchronisation des échanges
    - une ligne de synchronisation pour chaque côté
  - vitesse de transfert adaptée à l'interlocuteur

dia volontairement laissé vide

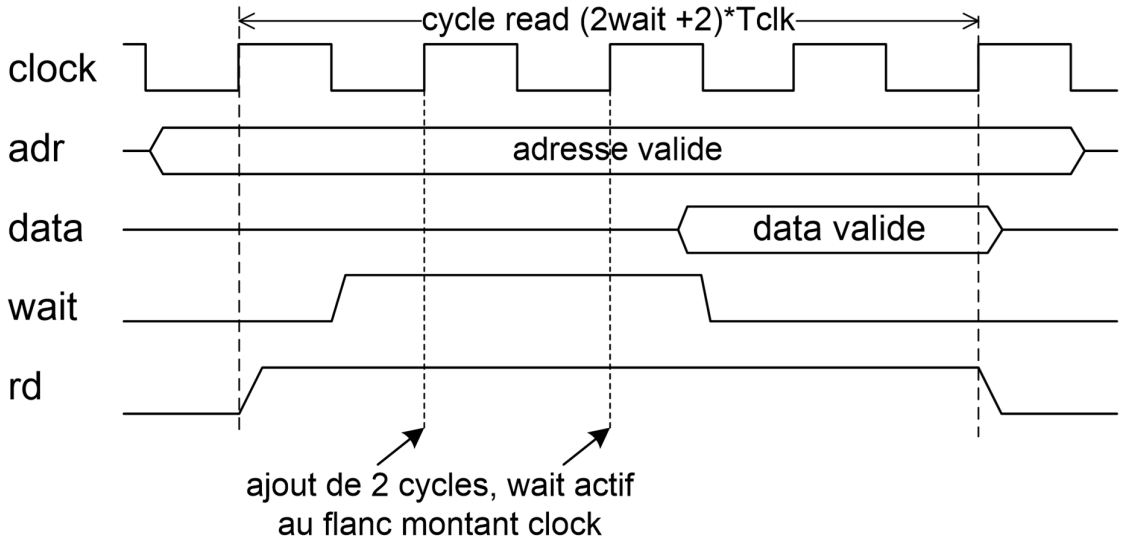
Bus synchrone

- Cycles de durée fixe imposé par le maitre
  - Tous les interfaces, esclaves, doivent respecter les timing du maitre (CPU)



Bus semi-synchrone

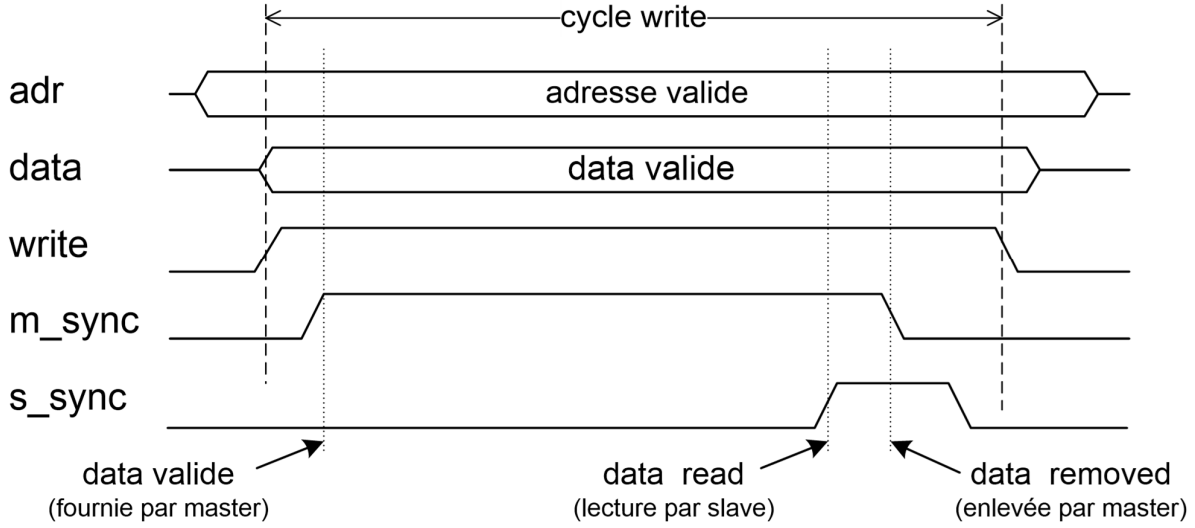
- Adaptation du timing avec l'ajout d'un signal pour prolonger le cycle de N périodes



Bus asynchrone

- Cycle d'écriture

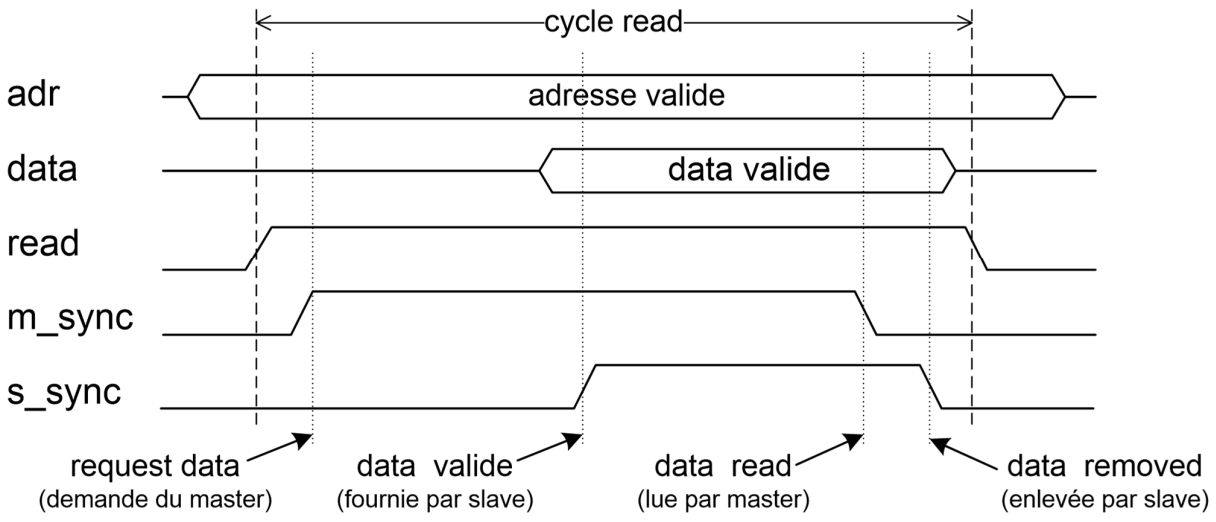
m_sync: master synchronisation
s_sync: slave synchronisation



Bus asynchrone

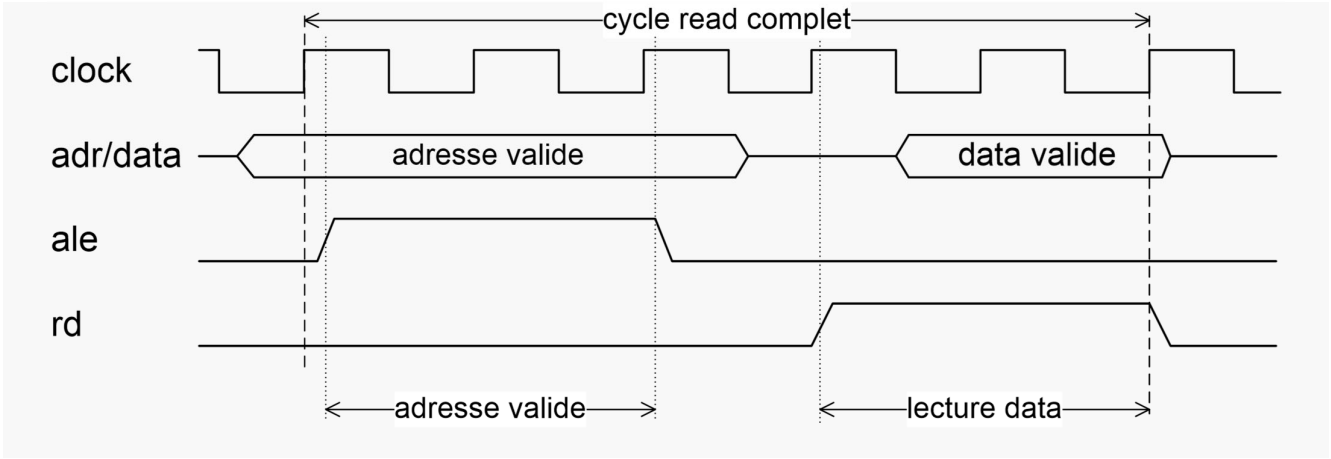
- Cycle de lecture

m_sync: master synchronisation
s_sync: slave synchronisation



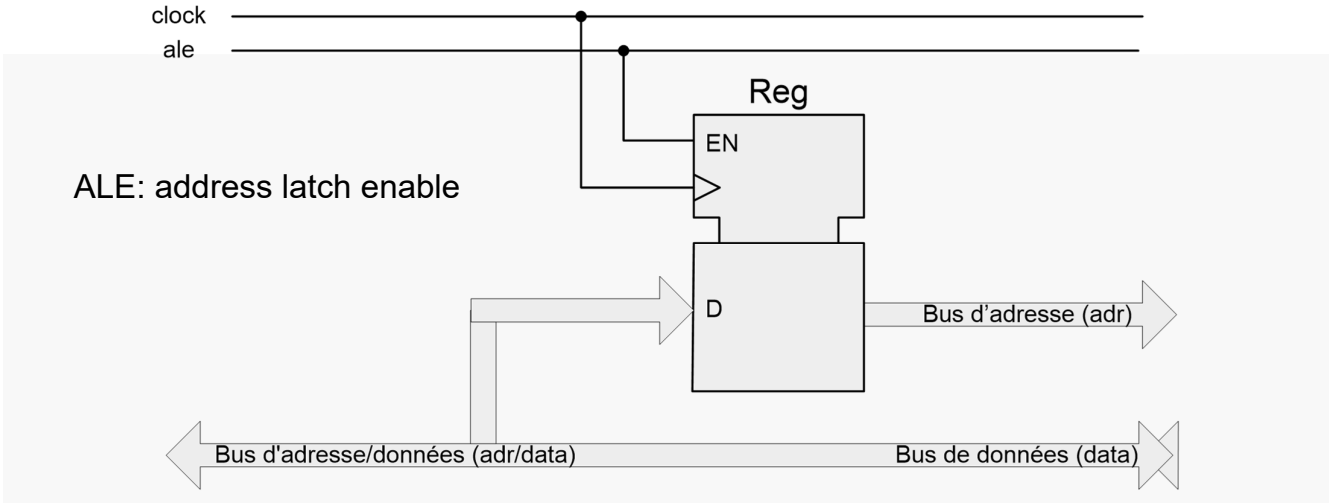
Bus multiplexé, timing

- Multiplexages sur un même bus des adresses et des données
  - économie du nombre de ligne
  - une phase adresse, puis une phase donnée



Bus multiplexé, schéma

- Nécessite de démultiplexer le bus *adr/data*
  - attention: démultiplexage temporel !
  - mémoriser les adresses lorsque ALE actif pour la phase de transfert des données



Accès par byte sur un bus

- Convention dans un système à processeur:
  - l'adresse numérote des bytes
  - disposition des bytes: Little endian ou Big endian
- Bus avec indication des bytes valides:
  - des signaux indiquent les bytes valides lors d'un transfert, abrégé **BE** pour **B**yte **E**nable
  - indication fixe selon convention du bus
  - bus 16 bits: BE1 (MSB) et BE0 (LSB)
  - bus 32 bits: BE3 (MSB), BE2, BE1 et BE0 (LSB)

Accès par byte sur un bus

- Exemple de bus 32 bits avec signaux de byte valide (BE : Byte Enable)

BE3 2 1 0	Byte3 D31…24	Byte2 D23…16	Byte1 D15…8	Byte0 D7...0	Explication
1 1 1 1	valide	valide	valide	valide	transfert de mots 32 bits
1 1 0 0	valide	valide	-	-	transfert de mots 16 bits
0 1 0 0	-	valide	-	-	transfert d'un byte
0 0 0 1	-	-	-	valide	transfert d'un byte