

Bus de communication

introduction

- bus de communications avec des systèmes extérieurs à la machine
 - ▶ on ne traitera pas les bus PCI, ISA, PCMCIA ou SCSI (périphériques)
 - ▶ ni les bus série comme RS232 ou USB
 - ▶ PC104 et PC104+ sont deux bus que l'on rencontre beaucoup dans les applications TR. Ils sont équivalents à ISA et PCI
- VME
- bus de terrain (CAN)

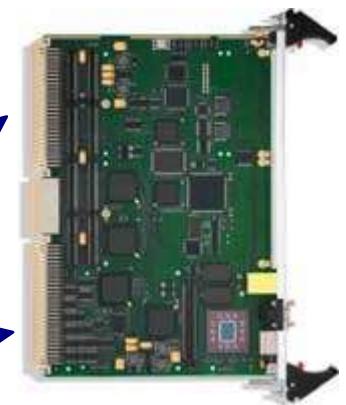
Bus VME

le bus VME

- Versa Module Eurocard
- standard industriel né au début des années 1980
 - au début pour les cartes basées sur des processeurs Motorola 68000
- très utilisé dans les applications mettant en jeu l'acquisition et le traitement des données
- norme passée dans le domaine public (pas de licence pour un bus propriétaire)
- VME International Trade Union <http://www.vita.com>

le standard VME

- défini dans la norme IEEE 1014-87 ou IEC 821-297
- bus multiprocesseur asynchrone:
 - ▶ plusieurs cartes maîtresses peuvent accéder à des ressources communes (cartes mémoire, modules d'E/S, etc...)
 - ▶ cadencé par un mécanisme de handshake
- cartes au format "double Europe" (norme DIN 41612 et DIN 41494) sur 2 connecteurs P1 et P2 de 96 broches



le standard VME

- Connecteur P1

- ▶ un bus de données de 16 bits
- ▶ un bus d'adresses de 24 bits
- ▶ 6 lignes de modification d'adresses
- ▶ plusieurs lignes d'alimentation et de contrôle

le standard VME

<i>Pin Assignment for the VMEbus P1/J1 Connector</i>			
Pin	Row a	Row b	Row c
1	D00	BBSY*	D08
2	D01	BCLR*	D09
3	D02	ACFAIL*	D10
4	D03	BG0IN*	D11
5	D04	BG0OUT*	D12
6	D05	BG1IN*	D13
7	D06	BG1OUT*	D14
8	D07	BG2IN*	D15
9	GND	BG2OUT*	GND
10	SYSCLK	BG3IN*	SYSFAIL*
11	GND	BG3OUT*	BERR*
12	DS1*	BR0*	SYSRESET*
13	DS0*	BR1*	LWORD*
14	WRITE*	BR2*	AM5
15	GND	BR3*	A23
16	DTACK*	AM0	A22

<i>Pin Assignment for the VMEbus P1/J1 Connector</i>			
Pin	Row a	Row b	Row c
17	GND	AM1	A21
18	AS*	AM2	A20
19	GND	AM3	A19
20	IACK*	GND	A18
21	IACKIN*	GND	A17
22	IACKOUT*	GND	A16
23	AM4	GND	A15
24	A07	IRQ7*	A14
25	A06	IRQ6*	A13
26	A05	IRQ5*	A12
27	A04	IRQ4*	A11
28	A03	IRQ3*	A10
29	A02	IRQ2*	A09
30	A01	IRQ1*	A08
31	-12 VDC	+5 VSTBY	+12 VDC
32	+5 VDC	+5 VDC	+5 VDC

* indique que le signal est activé quand le niveau est bas

● voir <http://www.vita.com/vmefaq/vmepins.html>

le standard VME

- Connecteur P2
 - ▶ extensions du bus de données (D15-D31) et du bus d'adresses (A24-A31)
 - ▶ extensions définies par l'utilisateur
 - ✓ VSB
 - ✓ PCI
 - ✓ autres...
- une extension du bus pour les processeurs 64 bits a été standardisée
 - ▶ utilise des connecteurs P1 et P2 à 160 broches
 - ▶ plus un connecteur optionnel P0 dédié aux E/S

organisation du bus VME

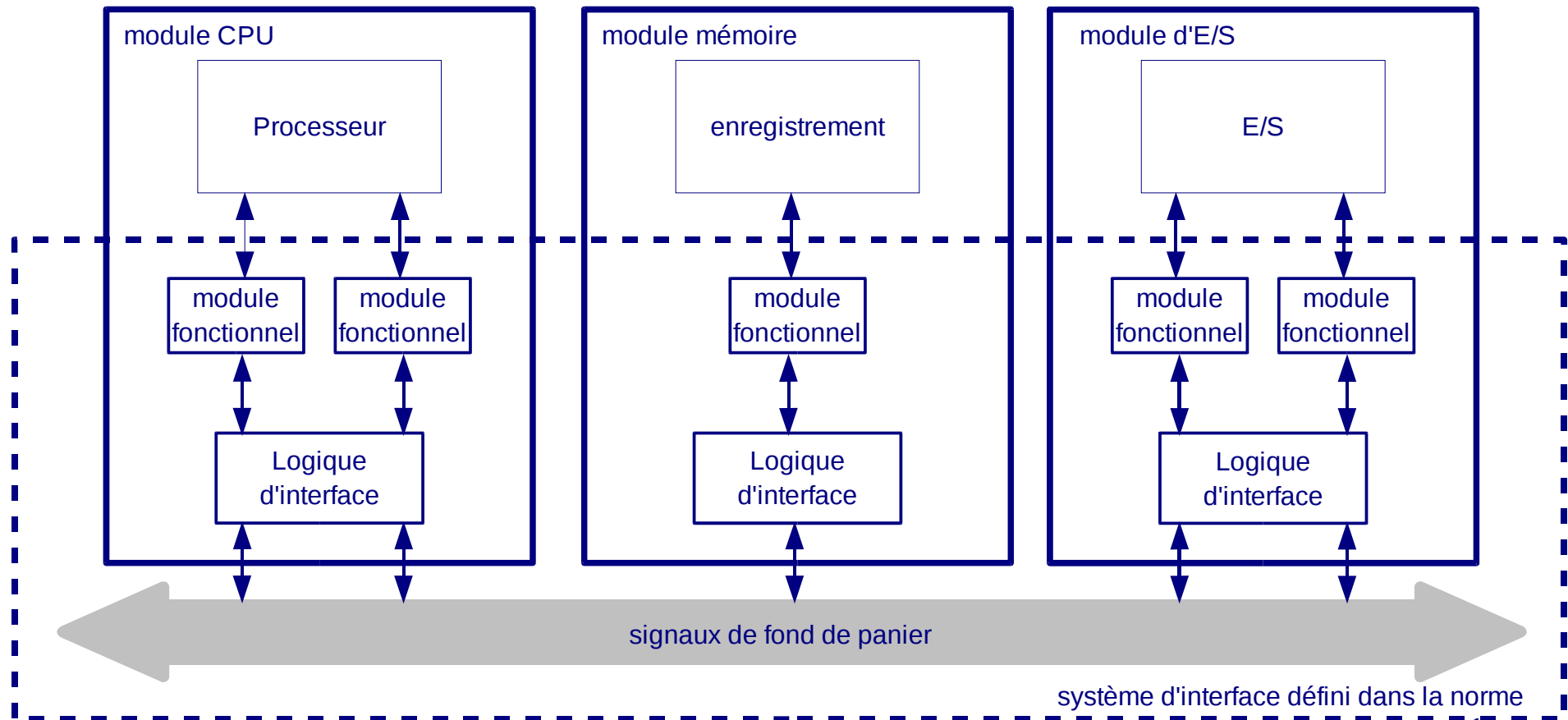
- transfert de données en mode maître-esclave
- asynchrone
- non multiplexé
- transferts sur des largeurs de mots de 8, 16 ou 32 bits
- 7 niveaux de priorité d'interruption
- 4 niveaux d'arbitrage du bus de données
- lignes spécialisées pour
 - ▶ détection rapide de défauts
 - ▶ contrôle de bus
 - ▶ contrôle de l'alimentation électrique

organisation du bus VME

- bus VME logiquement organisé en 5 sous-bus
 - ▶ Data Transfer Bus
 - ▶ Data Transfer Arbitration Bus
 - ▶ Priority interrupt Bus
 - ▶ Utility bus
 - ▶ Serial Bus

organisation du bus VME

- décrit en termes de modules fonctionnels qui servent à gérer l'utilisation du bus de transfert de données (DTB)
 - conceptuels, parfois associés à du hardware



organisation du bus VME

- fonctions d'interfaces

- ▶ transfert de données

- ✓ les modules maîtres peuvent initier des transferts de données sur le DTB
 - ✓ les modules esclaves peuvent détecter les transferts et y participer

- ▶ arbitrage du DTB

- ✓ plusieurs maîtres peuvent coexister dans un système VME, un seul à la fois peut avoir le contrôle du bus
 - ✓ module pour les fonctions de demande (requester)
 - ✓ module pour les fonctions d'arbitrage (arbiter)

organisation du bus VME

- ▶ gestion des interruptions
 - ✓ une ressource peut demander une interruption de l'activité normale
 - ✓ module pour la génération de l'interruption (interrupter)
 - ✓ module pour le traitement (interrupt handler)
- ▶ utilitaires
 - ✓ horloge
 - ✓ (ré)initialisation
 - ✓ défauts du système
 - ✓ défaut d'alimentation
 - ✓ lignes série de communication

transfert des données

- par l'intermédiaire d'un bus spécialisé (DTB)
 - ▶ bus d'adresses
 - ✓ bus de 32 bits A0 à A31
 - ✓ adressage sur 16, 24 ou 32 bits
 - ✓ espace mémoire adressable minimum : 1 octet
 - ▶ bus de données
 - ✓ 3 largeurs de mots : 8/16/32 bits (D8/D16/D32)
 - ▶ bus de modification d'adresse
 - ✓ AM0 à AM5
 - ▶ lignes de contrôle de transfert par le maître
 - ▶ lignes d'état contrôlées par l'esclave

transfert des données

- lignes de modification d'adresse
 - ▶ pour permettre au maître d'envoyer des informations supplémentaires lors d'un transfert :
 - ✓ configuration dynamique du système, en imposant à l'esclave de ne réagir qu'à un seul code
 - ✓ sélection dynamique de l'emplacement de l'esclave dans l'espace d'adressage du maître
 - ✓ modification dynamique des privilèges nécessaires pour accéder à certains niveaux d'exécution de l'esclave
 - ✓ modification du type de transfert (par blocs)

transfert des données

- lignes de contrôle d'état
 - ▶ 4 lignes de contrôle pour le module maître
 - ✓ $\overline{\text{AS}}$: address strobe
 - ✓ $\overline{\text{DS0}}$: data strobe 0 : sélection l'octet de poids faible
 - ✓ $\overline{\text{DS1}}$: data strobe 1 : sélection l'octet de poids fort
 - ✓ $\overline{\text{LWORD}}$: accès à un mot de 32 bits
 - ✓ $\overline{\text{WRITE}}$: direction du transfert (niveau bas :
 maître $\hat{=}$ esclave
 - ▶ 2 lignes d'état contrôlées par le module esclave
 - ✓ $\overline{\text{DTACK}}$: data acknowledge (transfert réussi)
 - ✓ $\overline{\text{BERR}}$: erreur dans le transfert en cours

transfert des données

- bus d'adresses

- ▶ pour accéder à un octet de mémoire par le module maître
 - ✓ valeur de l'adresse sur le bus d'adresse A01-A32
 - ✓ utilisation des lignes DS0, DS1, LWORD et A01 pour le mode d'accès

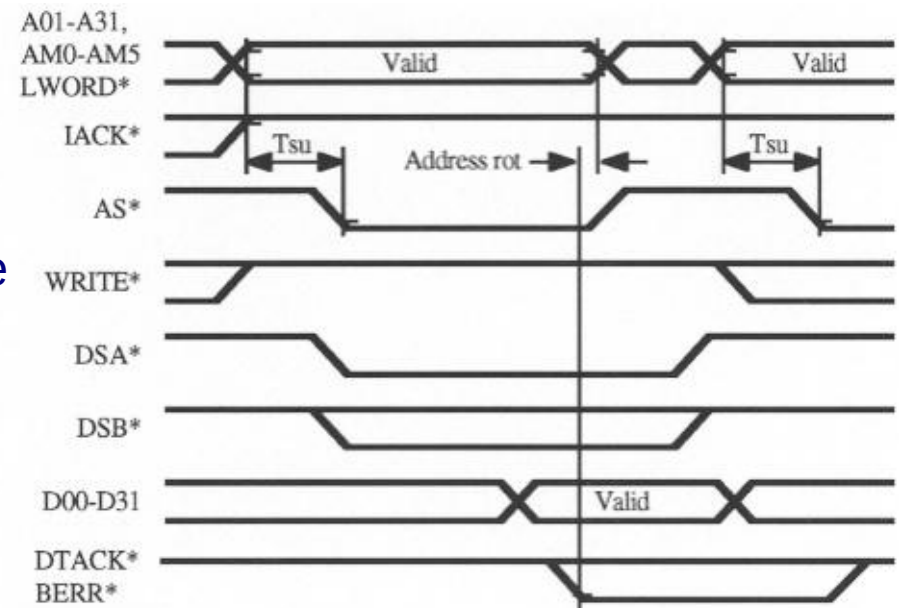
- lignes de données

- ▶ sur les lignes D0 à D15
- ▶ les lignes DS0, DS1, A01 et LWORD permettent de choisir l'octet, la longueur du transfert et l'alignement

transfert des données

exemple de lecture

- 1) le module maître utilise les lignes A01-31, AM0-AM5 et LWORD pour choisir l'esclave et le mode
- 2) le maître désactive IACK pour indiquer que ce n'est pas un cycle en réponse à un interrupt
- 3) le maître active AS pour valider l'adresse
- 4) le maître désactive WRITE pour indiquer un cycle de lecture
- 5) le maître active les lignes DS0 et/ou DS1 pour indiquer où (sur D0 à D31) il veut lire les données
- 6) après un certain temps, l'esclave active les lignes de données D0 à D31
- 7) quand les lignes D0-D31 sont stables, l'esclave active DTACK (ou BERR) pour signaler le succès (ou l'échec) du transfert
- 8) après un certain temps, le maître lit les lignes de données
- 9) le maître désactive les lignes DS0 et DS1 pour indiquer que les données ont été lues
- 10) l'esclave désactive DTACK pour indiquer qu'il a terminé avec le cycle



arbitrage pour l'accès au bus

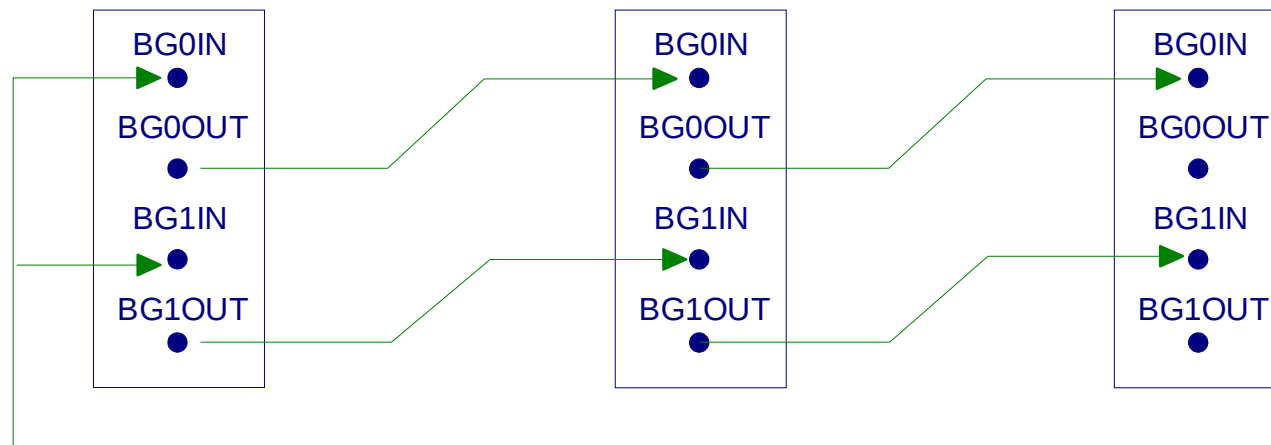
- plusieurs modules maîtres peuvent coexister dans le même chassis
- un seul maître peut posséder le contrôle du DTB
- la logique d'accès est implantée au niveau du hardware
 - ▶ accès exclusif d'un maître au DTB
 - ▶ gestion des demandes d'accès (BUS REQUEST)
 - ✓ 3 modes :
 - × PRI-ARBITER : priorité décroissante de BR3 à BR0
 - × RRS-ARBITER : Round robin de BR(n) à BR(n-1)
 - × ONE-ARBITER : requêtes de BR3 seulement acceptées

arbitrage pour l'accès au bus

- lignes d'arbitrage du bus DTB
 - ✓ 4 lignes de requêtes BR0 à BR3
 - ✓ 1 ligne BBSY (Bus Busy) pour indiquer que le bus est contrôlé par un maître
 - ✓ 1 ligne BCLR (Bus Clear) pour indiquer qu'une requête de priorité supérieure est présente en mode PRI-ARBITER
 - ✓ des lignes d'allocation de requête BG[0-3]IN et BG[0-3]OUT qui doivent être transmises d'un module à l'autre par une daisy chain (ou par des jumpers si il n'y a pas de module présent)
- un module demandant le contrôle du bus doit donc gérer 1 ligne de demande (BR0-3), 1 ligne d'allocation (BG[0-3]OUT) et la ligne de bus busy BBSY

arbitrage pour l'accès au bus

- le module assurant le rôle d'arbitre DOIT occuper le slot 1 dans le chassis VME
- le positionnement géographique des modules procure implicitement un niveau de priorité supplémentaire



arbitrage pour l'accès au bus

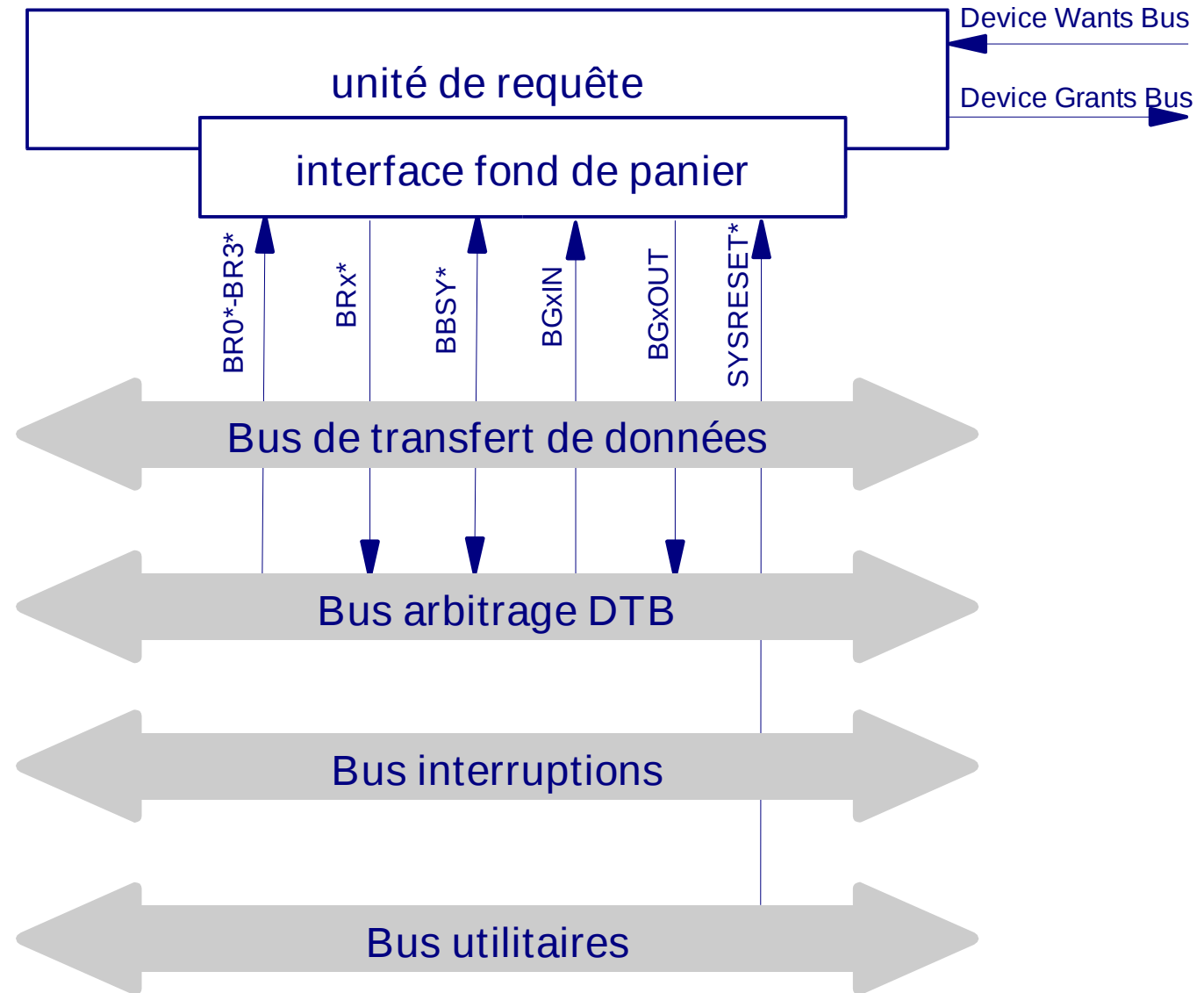
- générateur de requête

- ▶ gère le signal `DEVICE_WANT_BUS` du maître de son module VME, ou l'unité de gestion des interrupts de ce même module
- ▶ génère un Bus Request (`BR[0-3]`)
- ▶ détecte le signal `BG[0-3]IN` et le transmet à `BG[0-3]OUT` si le bus n'est pas demandé par le maître de son module
- ▶ si l'unité maître demande le DTB et que la ligne `BG[0-3]IN` est en niveau bas, le générateur indique la disponibilité du bus par un signal `DEVICE_GRANTED_BUS` et met la ligne `BBSY` en niveau bas

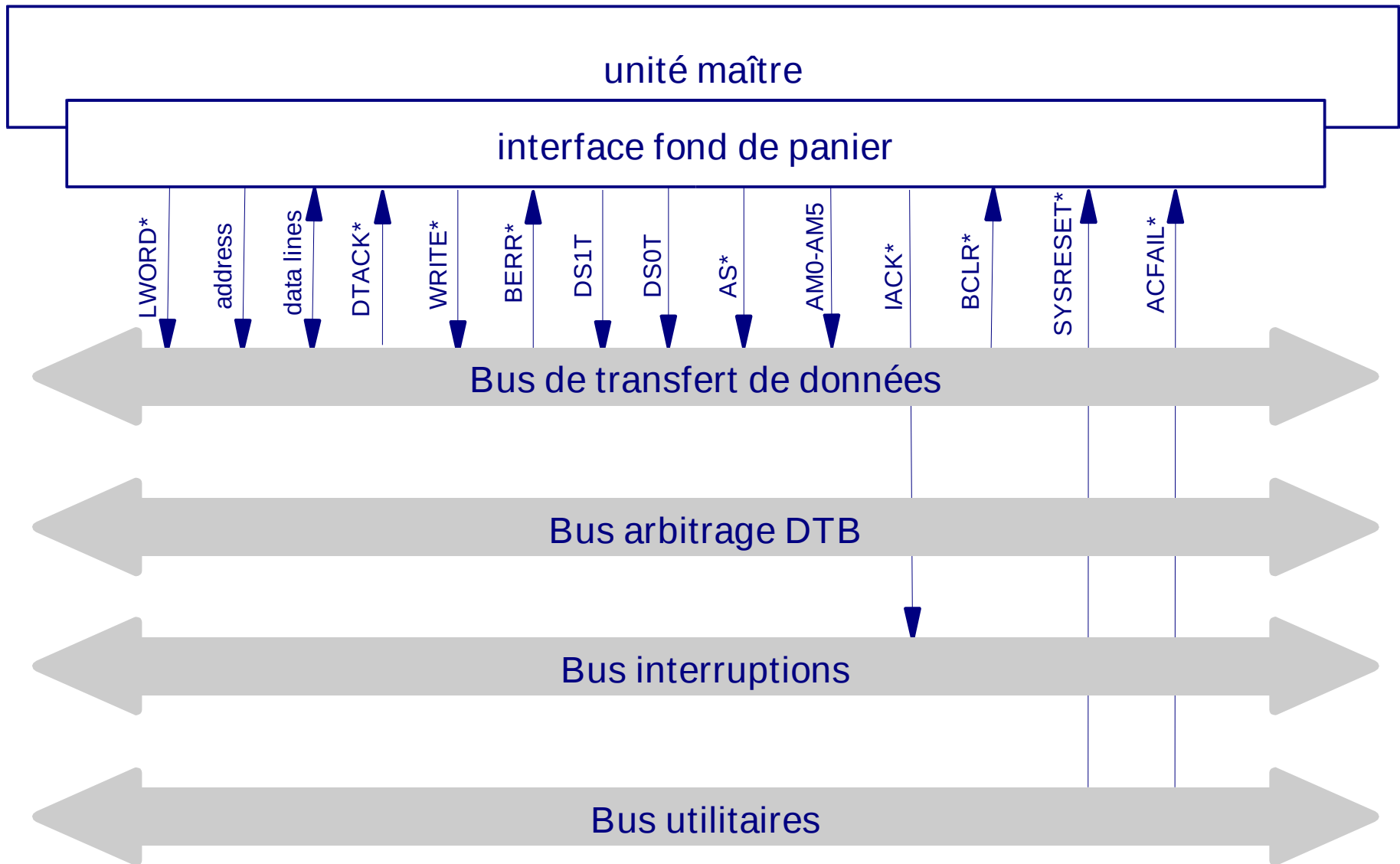
arbitrage pour l'accès au bus

- ▶ 3 modes de fonctionnement pour l'unité de requête de bus
 - ✓ **RWD (Release When Done)**
BSSY est remise au niveau haut dès que la demande de contrôle est annulée
 - ✓ **ROR (Release On Request)**
BSSY n'est remise au niveau haut que quand la demande de contrôle est annulée et qu'une autre requête BR[0-3] est présente (évite la surcharge du bus)
 - ✓ **FAIR**
pas de requête de bus tant que des requêtes sont encore en cours de traitement

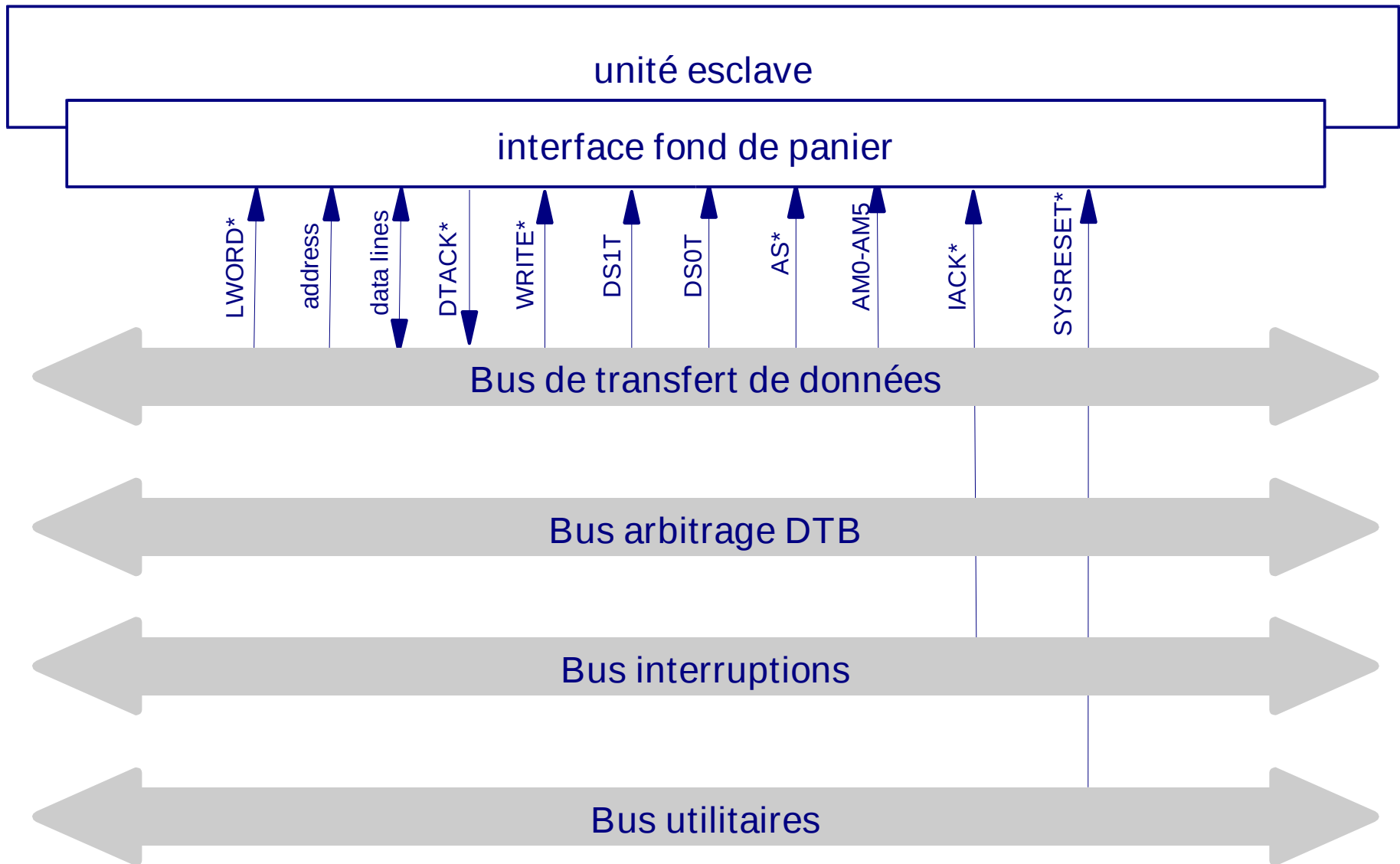
bloc diagramme du module de requête



bloc diagramme du module maître

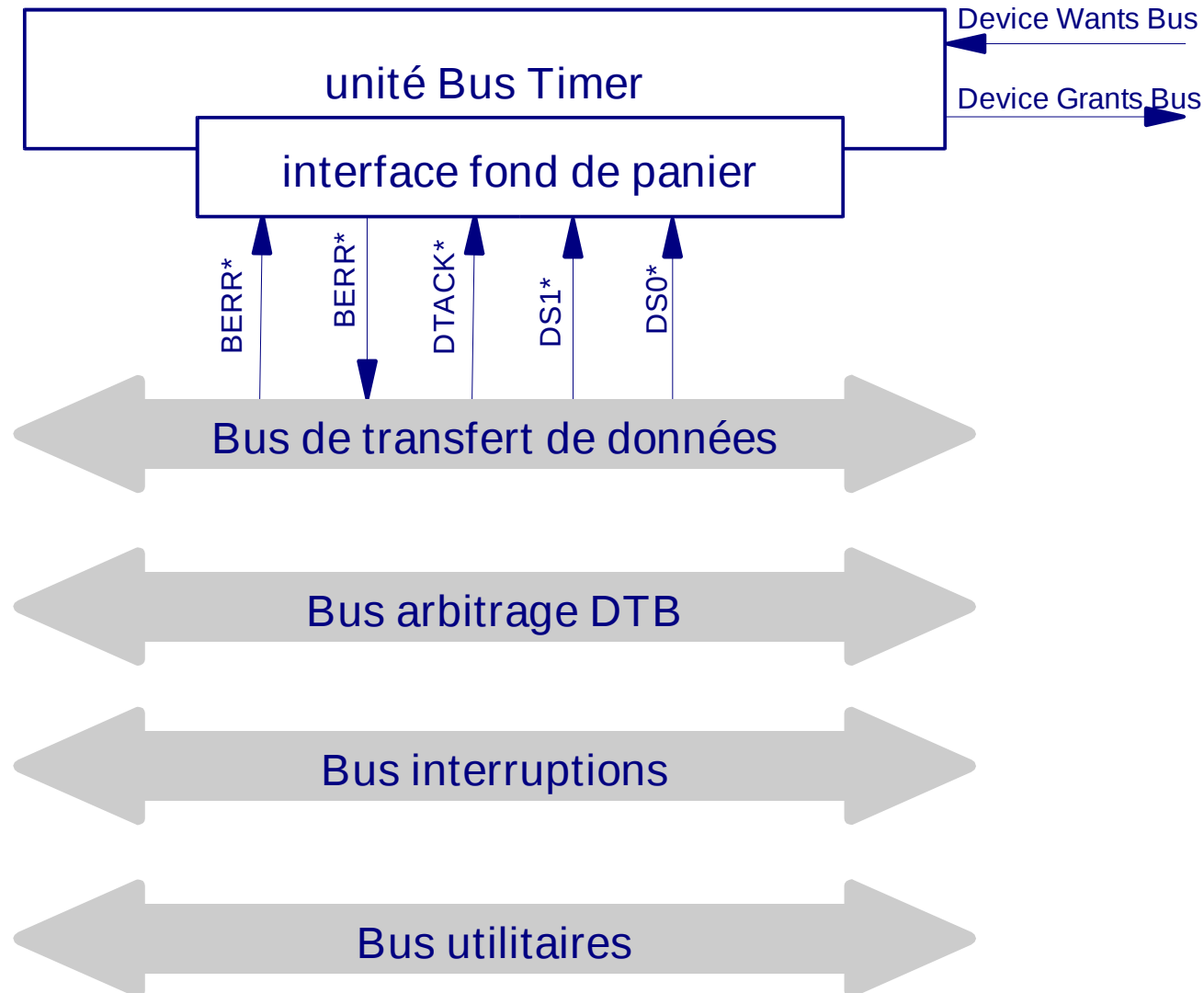


bloc diagramme du module esclave



bloc diagramme du module bus timer

- mesure du temps mis par les cycles sur le DTB



les modes de transfert de données

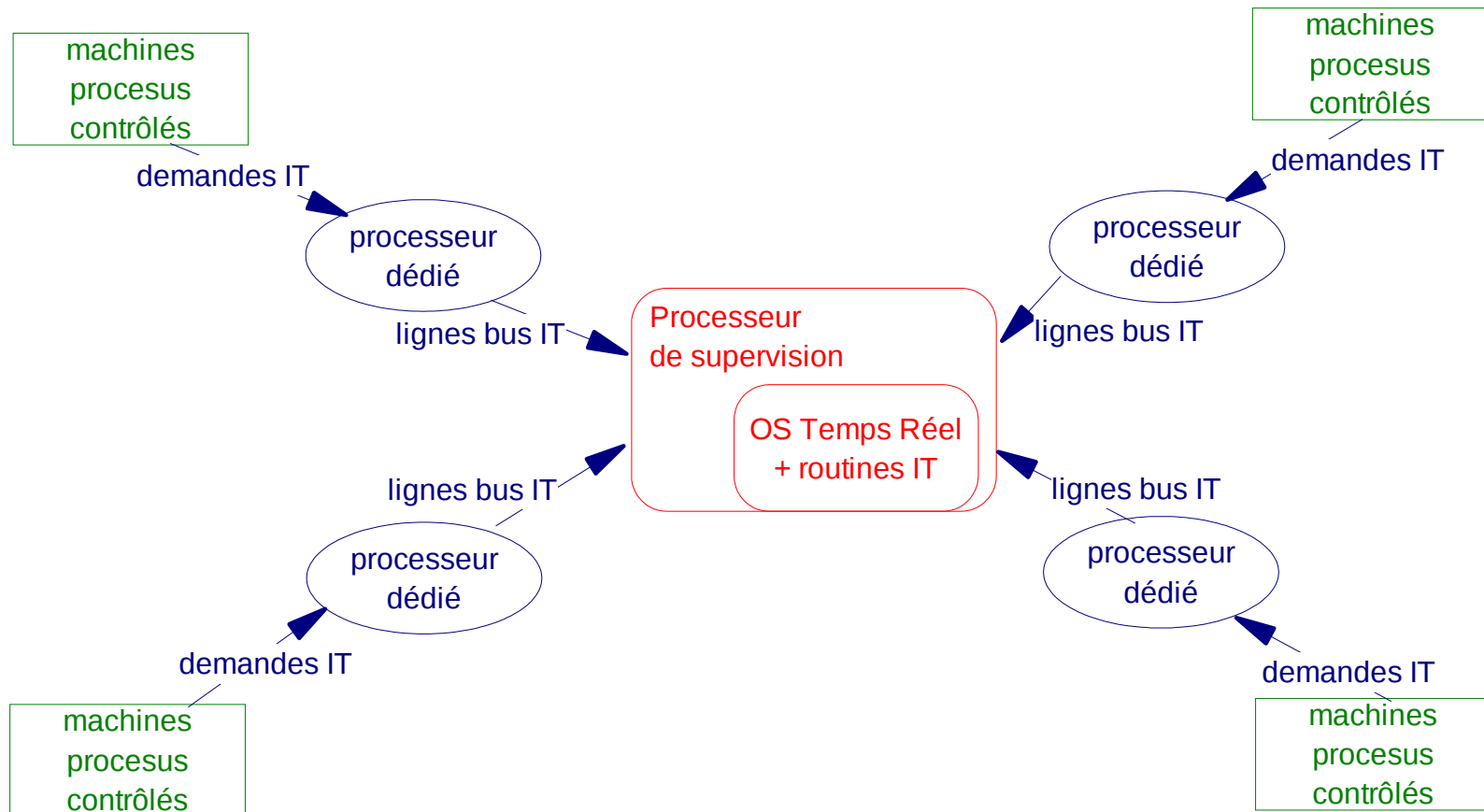
- cycle VME : séquence d'opérations qui permet la communication entre deux modules sur le bus VME, en suivant les protocoles définis dans la norme
- une fois qu'un cycle est démarré, et jusqu'à ce qu'il se termine, les autres cartes du chassis n'ont plus accès au contrôle
- les transferts de données peuvent se faire en modes 8 bits (octet et octet étendu), 16 bits et 32 bits

les modes de transfert de données

- 7 types de cycles :
 - ▶ **lecture** : transfert esclave $\hat{I}$ maître
 - ▶ **écriture** : transfert maître $\hat{I}$ esclave
 - ▶ **lecture par blocs** : seule l'adresse de départ est fournie et le maître ne rend le bus qu'à la fin
256 blocs maximum
 - ▶ **écriture par blocs** : analogue à la lecture par blocs
 - ▶ **lecture-modification-écriture**, sans rendre le bus pour émuler l'atomicité de l'accès à une variable
 - ▶ **adressage** pour positionner simplement les adresses sur le bus
 - ▶ **acquittement d'interruption** pour confirmer la réception d'un interrupt et le transfert du vecteur

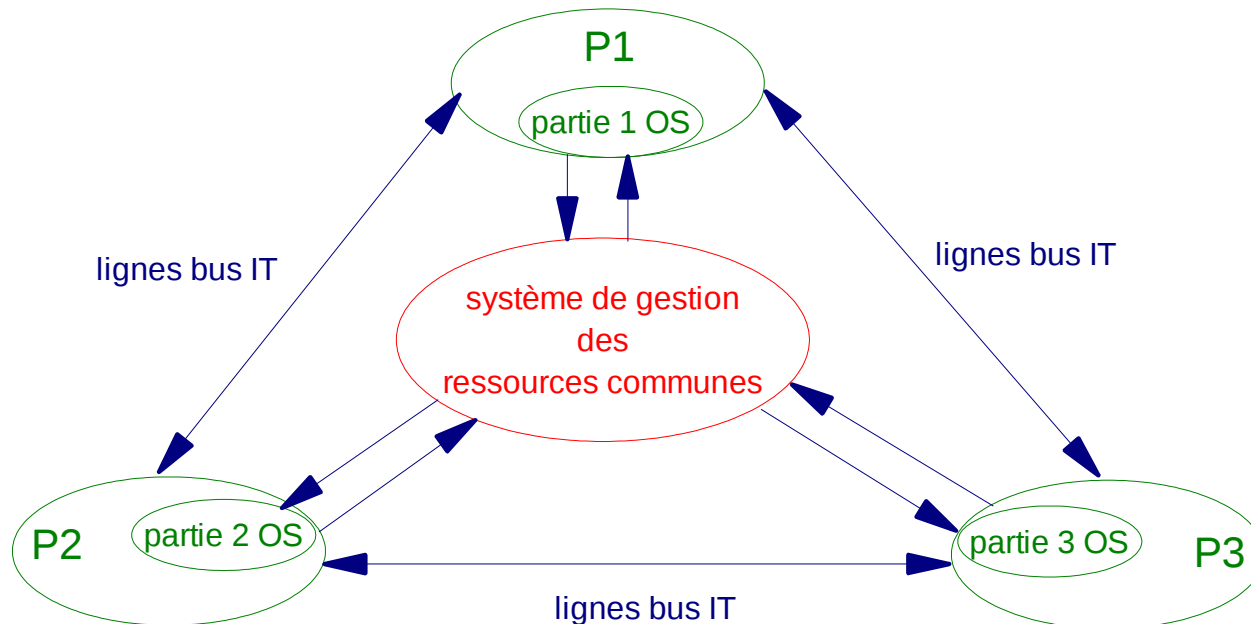
bus hiérarchisé d'interruptions

- pour générer et traiter les interruptions
- 2 catégories de systèmes de gestion :
 - ▶ système centralisé où un seul processeur reçoit et traite les interruptions



bus hiérarchisé d'interruptions

- ▶ système décentralisé où chaque processeur ne traite que les interruptions qui lui sont destinées. L'allocation de la prochaine tâche se fera sur le premier processeur qui se libère. Chaque processeur exécute une partie du système d'exploitation et a accès aux ressources globales

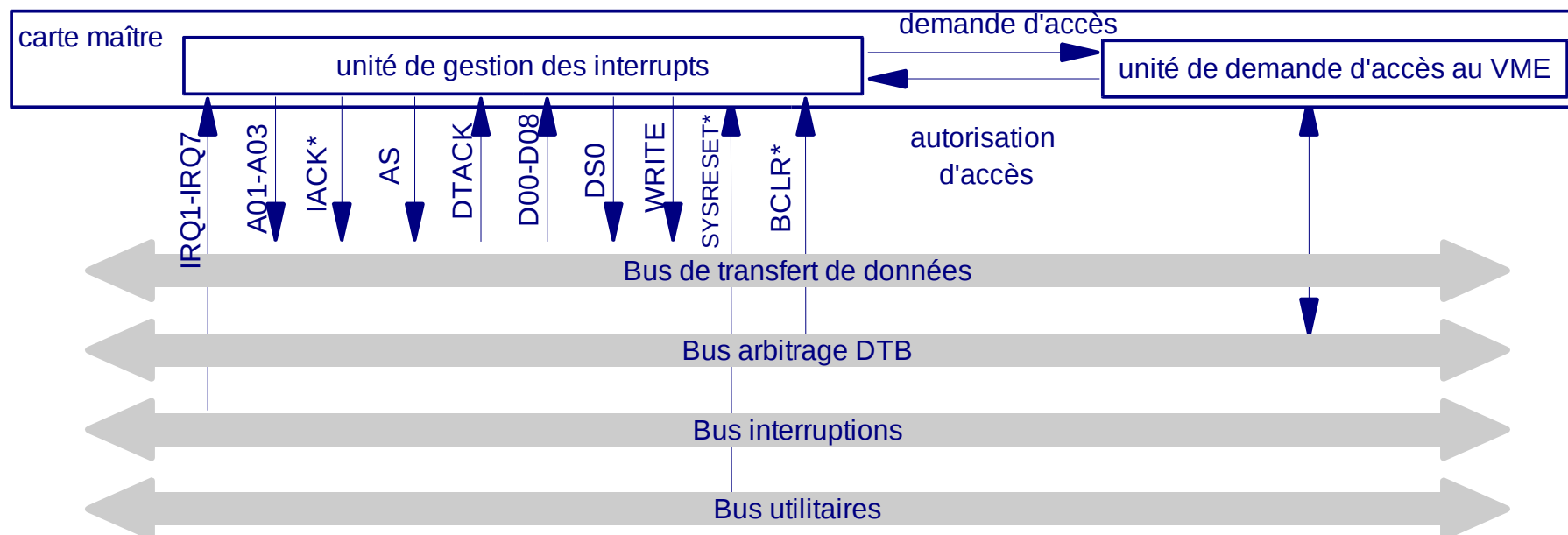


bus hiérarchisé d'interruptions

- lignes de gestion des interruptions
 - ▶ 7 lignes de demande IRQ1 à IRQ7
 - ✓ chaque ligne peut être activée par un niveau bas de l'unité de demande d'interruption (interrupter)
 - ✓ IRQ7 est la plus prioritaire dans un système centralisé
 - ▶ 1 ligne de prise en compte ACK
 - ▶ 2 lignes IACKIN et IACKOUT pour faire un chaînage des interruptions
 - ✓ la continuité des lignes doit être assurée par des cavaliers si un slot n'est pas occupé

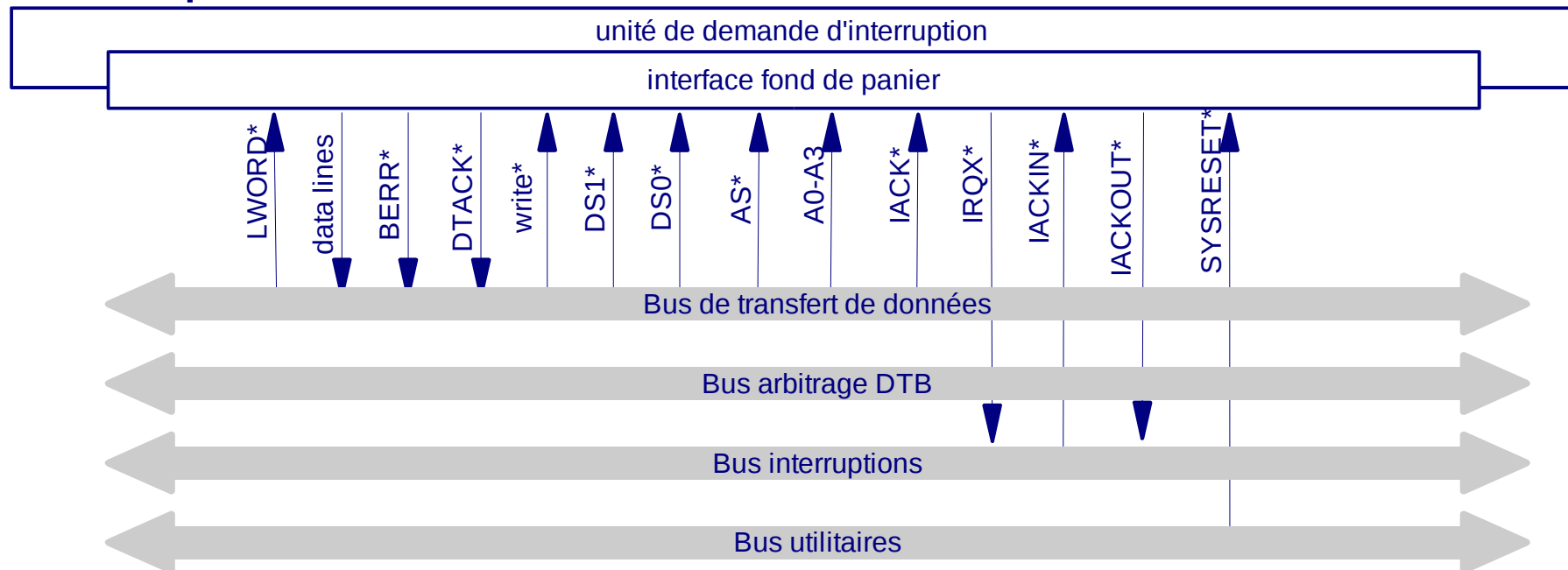
bus hiérarchisé d'interruptions

- unité de gestion des interruptions, pour
 - ▶ déterminer parmi les demandes celle qui a la priorité la plus élevée
 - ▶ demander le contrôle du bus via l'unité de demande de contrôle du bus et générer alors le signal IACK de prise en compte de l'interrupt sur le bus
 - ▶ lire le vecteur d'interruption et initialiser la séquence de traitement



bus hiérarchisé d'interruptions

- unité de demande des interruptions
 - ▶ adresse une demande à l'unité de gestion associée à la ligne de demande utilisée
 - ▶ fournit un vecteur d'interruption quand elle reçoit le signal de prise en compte
 - ▶ transmet le signal de prise en compte si elle n'est pas la source



bus utilitaires

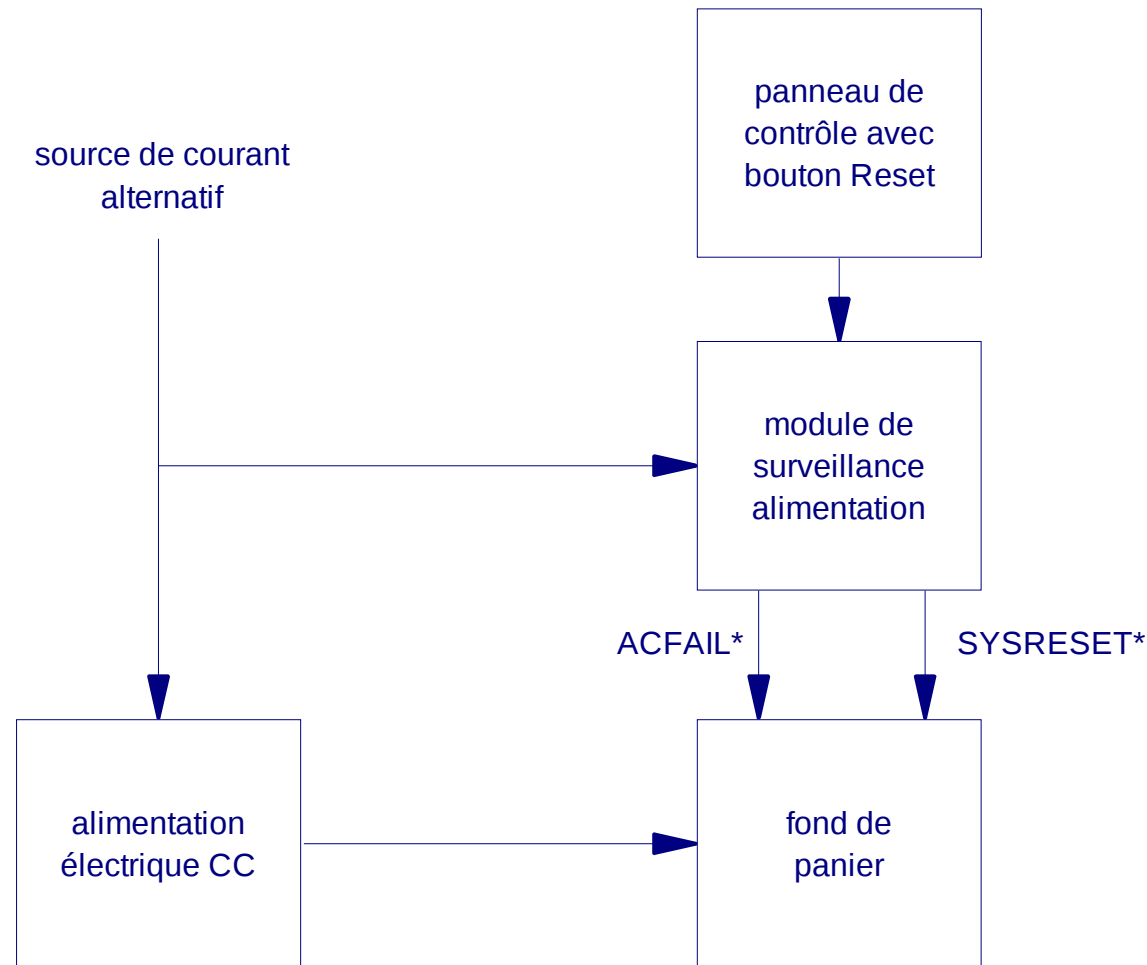
- lignes utilitaires :
 - ▶ **SYSCLK** horloge système
 - ▶ **SERCLK** horloge série
 - ▶ **SERDAT** données série
 - ▶ **ACFAIL*** panne d'alimentation
 - ▶ **SYSRESET** initialisation du système (prise en compte optionnelle)
 - ▶ **SYSFAIL** panne système

bus utilitaires

- modules utilitaires :
 - ▶ générateur de l'horloge système
 - ✓ autonome, fréquence 16 MHz, dans le module dans le slot #1
 - ✓ base de temps pour certains systèmes
 - ▶ horloge série
 - ✓ forme suivant IEEE P1132
 - ✓ pour des transferts série, en utilisant également les 2 lignes de transfert série

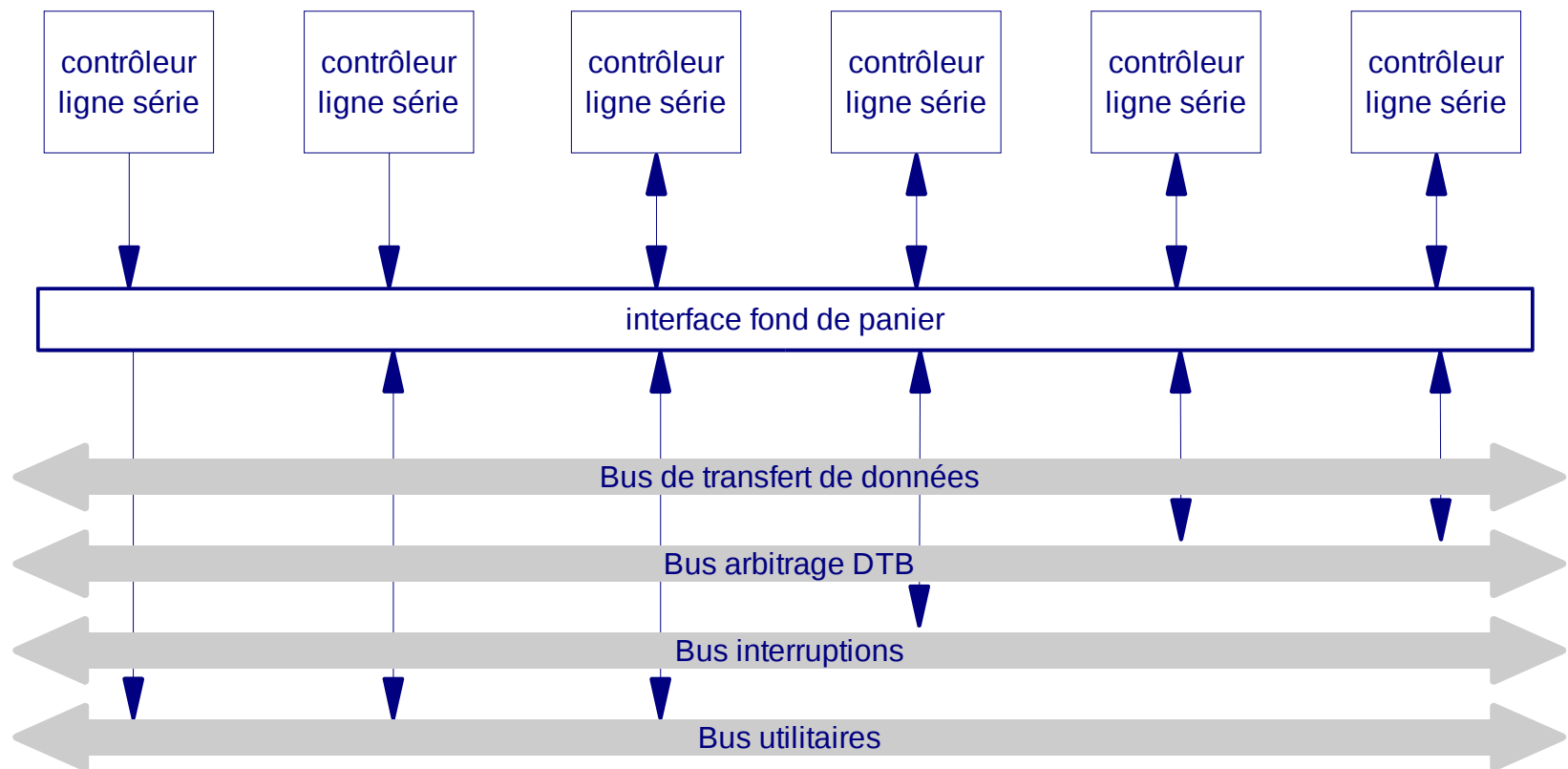
bus utilitaires

► surveillance de l'alimentation électrique



contrôle du système

- fonction assurée par le premier module dans le chassis



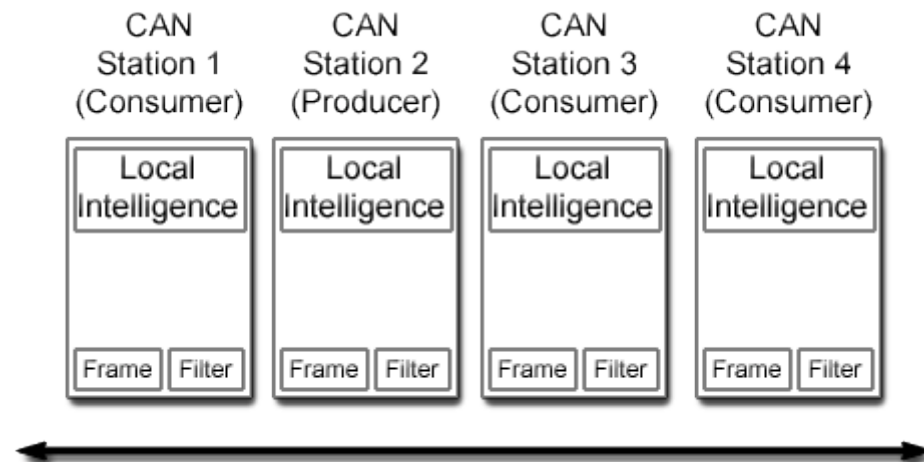
Bus CAN

introduction

- Controller Area Network présenté en 1986 par la société Bosch (équipementier pour l'automobile)
 - ▶ <http://www.can-cia.org>
- système basé sur un bus série
 - ▶ mécanisme d'arbitration non destructive garantissant l'accès immédiat au bus pour la requête de plus haute priorité
 - ▶ mécanisme de contrôle décentralisé (pas de maître)
 - ▶ détection automatique des erreurs et déconnexion des modules fautifs
 - ▶ identification des messages non à partir des adresses (émetteur/récepteur) mais à partir du contenu
 - ▶ lié à OSEK

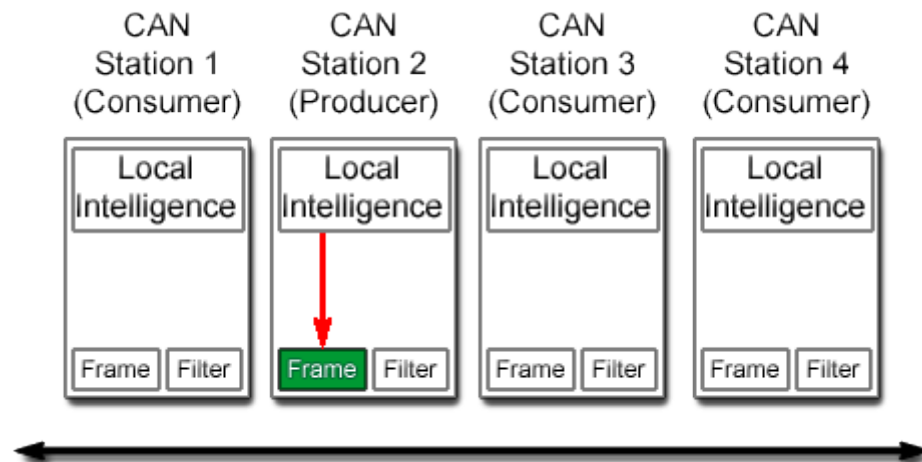
protocole de communication

- ISO 11898
- mécanisme de communication par diffusion (broadcast)



protocole de communication

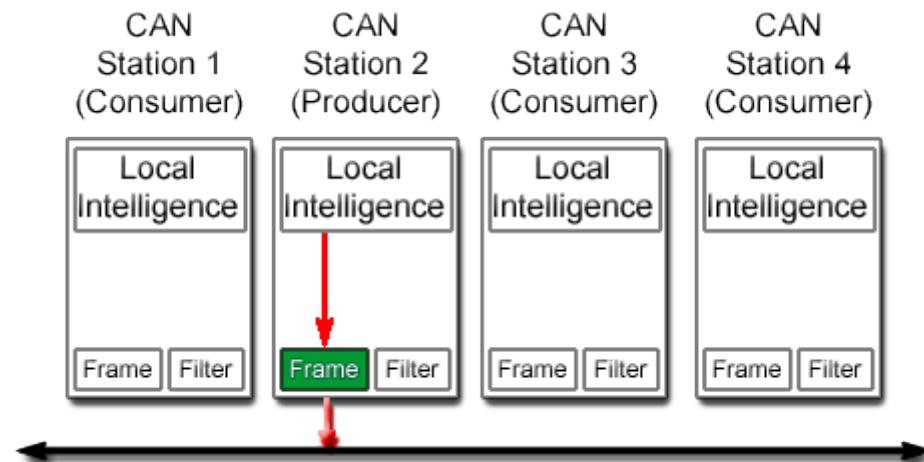
- ISO 11898
- mécanisme de communication par diffusion (broadcast)



© 2002. CAN in Automation - TS

protocole de communication

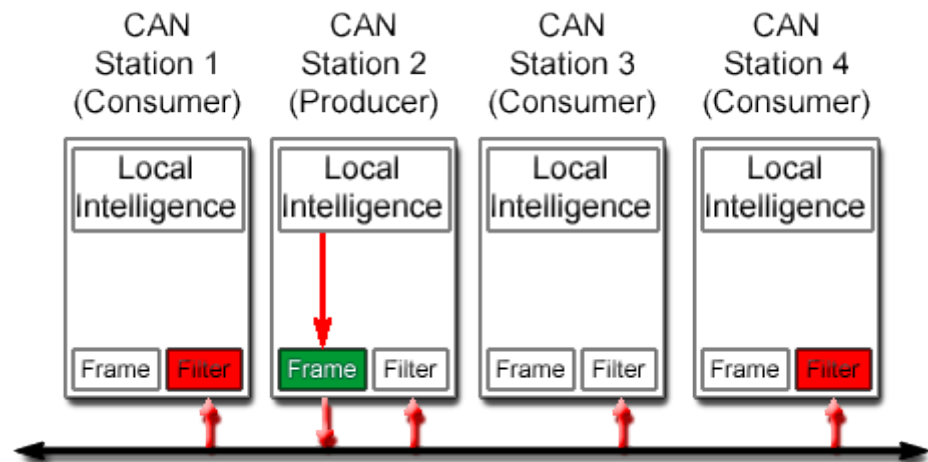
- ISO 11898
- mécanisme de communication par diffusion (broadcast)



© 2002. CAN in Automation - TS

protocole de communication

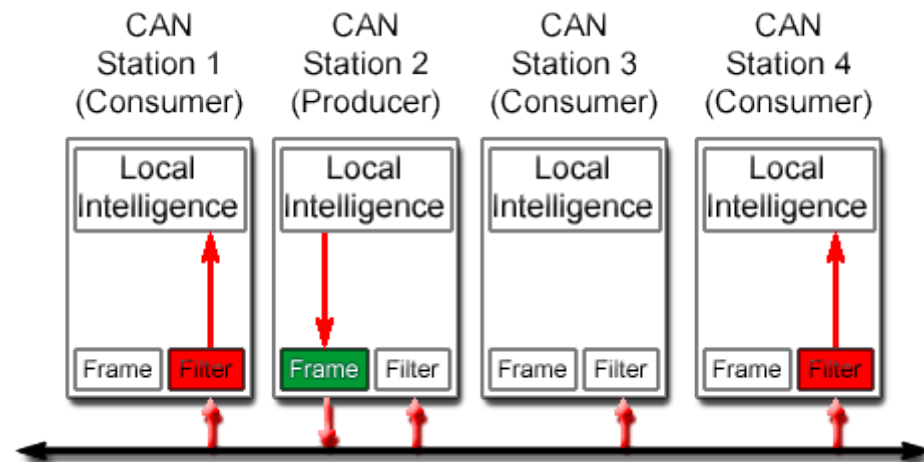
- ISO 11898
- mécanisme de communication par diffusion (broadcast)



© 2002. CAN in Automation - TS

protocole de communication

- ISO 11898
- mécanisme de communication par diffusion (broadcast)



© 2002. CAN in Automation - TS

protocole de communication

- procure une grande flexibilité (ajout de stations de réception)
- arbitrage à partir de l'identificateur du message
 - ▶ comparaison bit à bit
 - ▶ l'identificateur de plus petite valeur est le plus prioritaire
 - ▶ défini au moment du design et ne peut pas être changé dynamiquement
- format des messages standardisé

protocole de communication

- détection des erreurs directement au niveau du message
 - ▶ utilisation d'un mécanisme de CRC (Cyclic Redundancy Check)
 - ▶ vérification de la structure globale du frame
 - ▶ l'envoi d'un bit ACK par tous les receveurs qui fait qu'une erreur est détectée par l'émetteur si aucun ACK n'est reçu
- ou au niveau du bit
 - ▶ en monitorant au niveau des émetteurs les niveaux sur le bus et en comparant les bits émis et les bits reçus

protocole de communication

- dès qu'une erreur est détectée
 - ▶ la transmission en cours est interrompue par l'envoi d'un flag d'erreur
 - ▶ après la terminaison de la transmission interrompue, l'émetteur essaie de re-transmettre (avec éventuellement compétition pour accéder au bus)
 - ▶ une analyse statistique intrinsèque au protocole permet de distinguer les erreurs sporadiques et les erreurs permanentes. Un mode spécial peut être déclenché pour isoler la station défectueuse

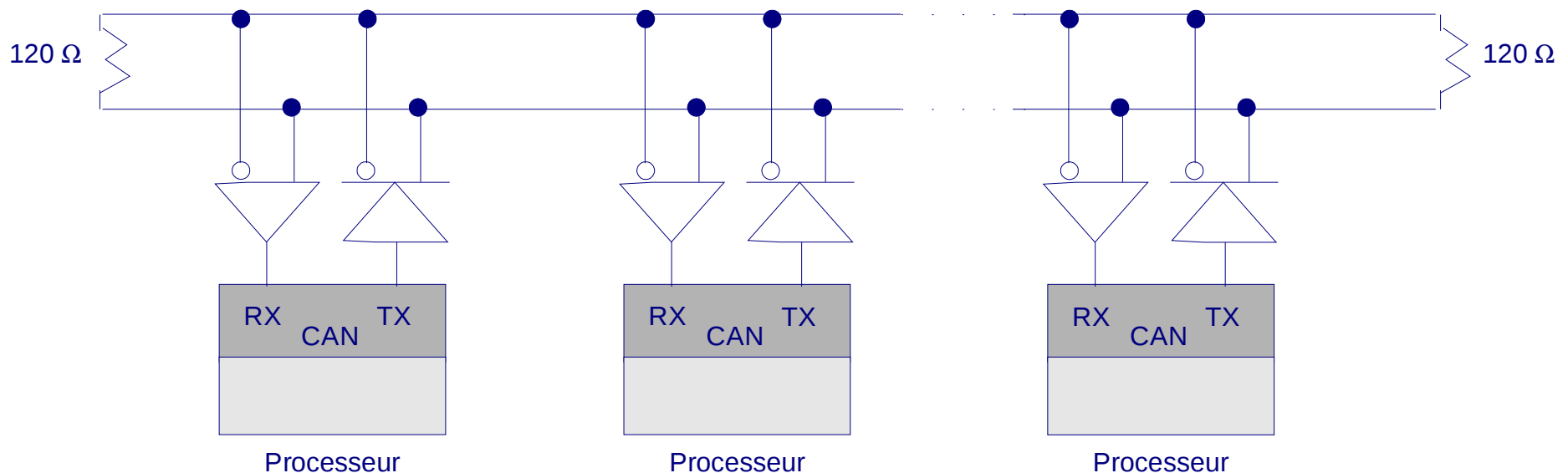
couche physique de communication

- CAN suit le modèle OSI et implémente directement (au niveau du matériel) les 2 couches inférieures
 - ▶ couche physique
 - ▶ couche de liens
- 2 options pour la couche physique
 - ▶ haute vitesse (ISO 11898-2)
 - ▶ basse vitesse, tolérante aux fautes (ISO 11898-3)

couche physique de communication

- option à haute vitesse

- ▶ de 1 Mb/s pour une longueur maximale de 40 m
- ▶ à ~ 50 kb/s pour une longueur maximale de 1 km
- ▶ sur un bus en paire torsadée terminée par des résistances de charge de 120 Ω



couche physique de communication

- option à basse vitesse, tolérante aux fautes
 - ▶ plusieurs versions
 - ✓ ISO 11519-2
 - ✓ ISO 11992
 - ✓ **ISO 11898-3**
 - ▶ résistance de terminaison $\sim 100 \Omega$ (pas moins)
 - ▶ entre 20 et 32 stations sur un bus de 40 m maximum à 125 kb/s maximum

extensions de CAN

- **TTCAN** : time triggered CAN
 - ▶ décrit dans la norme ISO 11898-4
 - ▶ pour gérer l'ordonnancement de messages dont l'envoi sur un bus CAN est déclenché par le temps ou par des événements
 - ✓ applications de contrôle
 - ✓ amélioration des performances temps réel (diminution de la gigue dans la latence de transmission des messages, déterminisme du schéma de communication, tolérance aux fautes)

extensions de CAN : TTCAN

- ▶ basé sur l'utilisation d'un message de référence envoyé périodiquement par un maître du temps
- ▶ l'envoi du message de référence démarre un "cycle de base" qui comporte plusieurs fenêtres temporelles de tailles différentes durant lesquelles des messages peuvent être envoyés
 - ✓ fenêtres "exclusives" pour les messages périodiques, au début desquelles un message prédéfini est envoyé
 - ✓ fenêtres "arbitrées" pour les messages non périodiques (spontanés)
 - ✓ fenêtres "libres" pour éventuellement étendre le réseau
 - ✓ le pattern des fenêtres est défini au moment du design

extensions de CAN : TTCAN

- ▶ tolérance aux fautes grâce à la présence de plusieurs "maîtres du temps"
 - ✓ 1 seul maître effectif
 - ✓ plusieurs maîtres potentiels
 - ✓ accès à la maîtrise suivant la priorité du message de référence
 - ✓ tentative de prise de la maîtrise à l'occurrence d'un timeout sur la réception du message de référence
- ▶ possibilité de changer dynamiquement la base de temps, ou même de passer temporairement à une structure apériodique (event driven) grâce à un message de référence spécial

extensions de CAN : CANopen

- couche de haut niveau construite au dessus de CAN
- initialement un projet ESPRIT, contrôlé depuis 1995 par le consortium CiA
- facilite le travail des concepteurs en proposant
 - ▶ des objets pour la communication (Process Data Objects)
 - ▶ des objets pour la configuration (Service Data Objects)
 - ▶ des objets pour la gestion du temps (réel)
 - ▶ des objets pour la gestion du réseau
 - ▶ des couches applicatives
- permettant de ré-utiliser le software et le hardware

autres extensions de CAN

- développements spécifiques
 - ▶ CAN Kingdom (contrôle de machines)
<http://www.cankingdom.org/>
 - ▶ DeviceNet (automatisation d'usines)
<http://www.odva.org/>