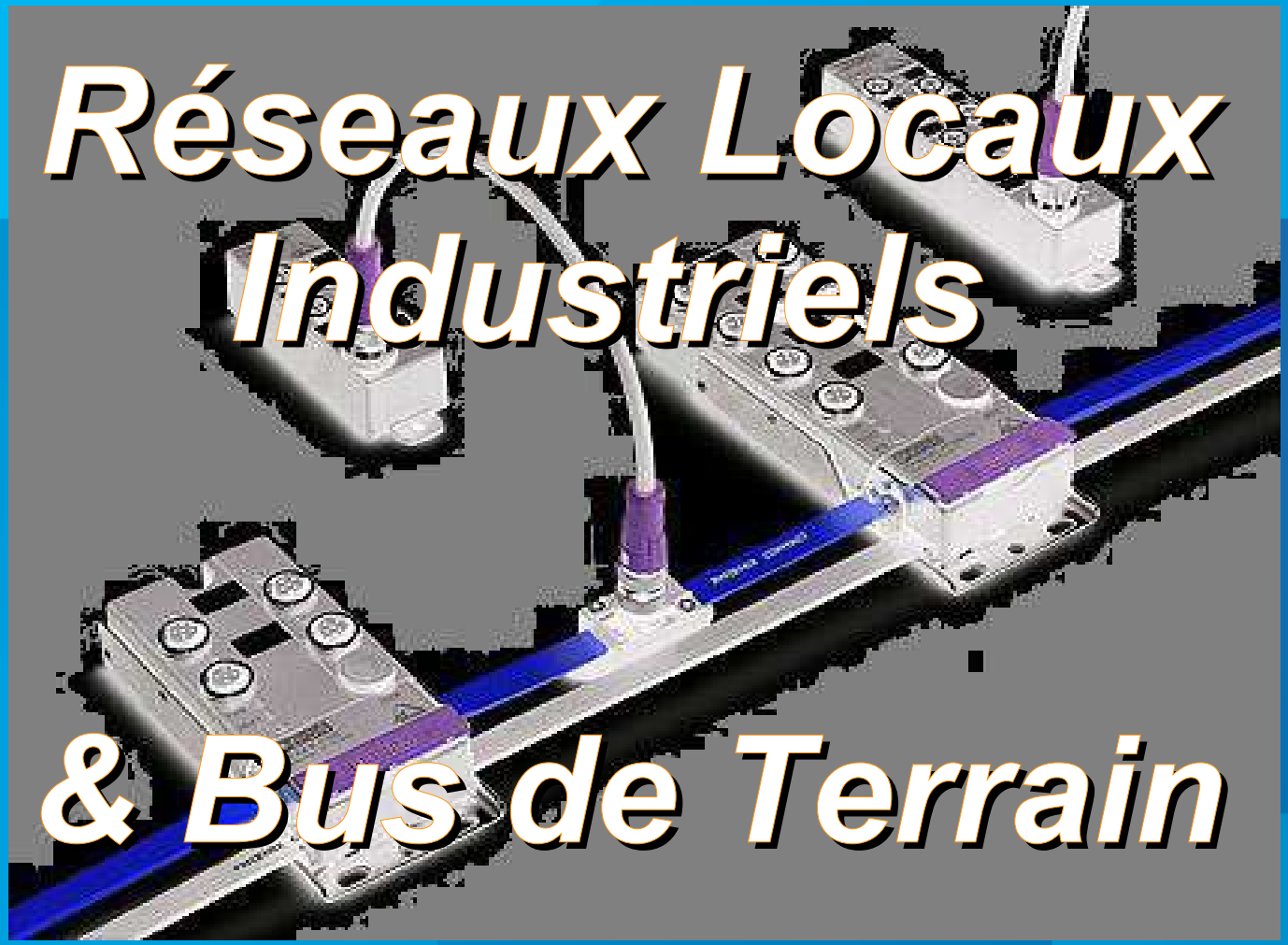
Réseaux Locaux Industriels & Bus de Terrain



neanne@univ-tln.fr

Plan du Cours :

1 / Introduction aux RLI -

2 / Asi : Actuator Sensor interface

3 / Modbus (RTU & TCP)

4 / ProfiBUS / ProfiNET

5 / CANopen

Ch 1 : Introduction aux RLI

1 / Place des RLI dans l'industrie

a - Définitions
b - Les différents niveaux de communication
c - Quelques Protocoles Standards
d - Intérêts des RLI

2 / Contraintes "temps réel"

3 / Modèle de Référence des RLI

a - Rappel : Modèle OSI
b - Adaptation au cas des RLI

4 / Critères de Comparaison

5 / Compétences liées aux RLI

6 / Conclusion / Perspectives

1- Place des RLI dans l'industrie

a - Définitions

• **Bus de terrain =** Systeme de Communication Numérique, reliant différents types d ’équipements d ’automatismes.

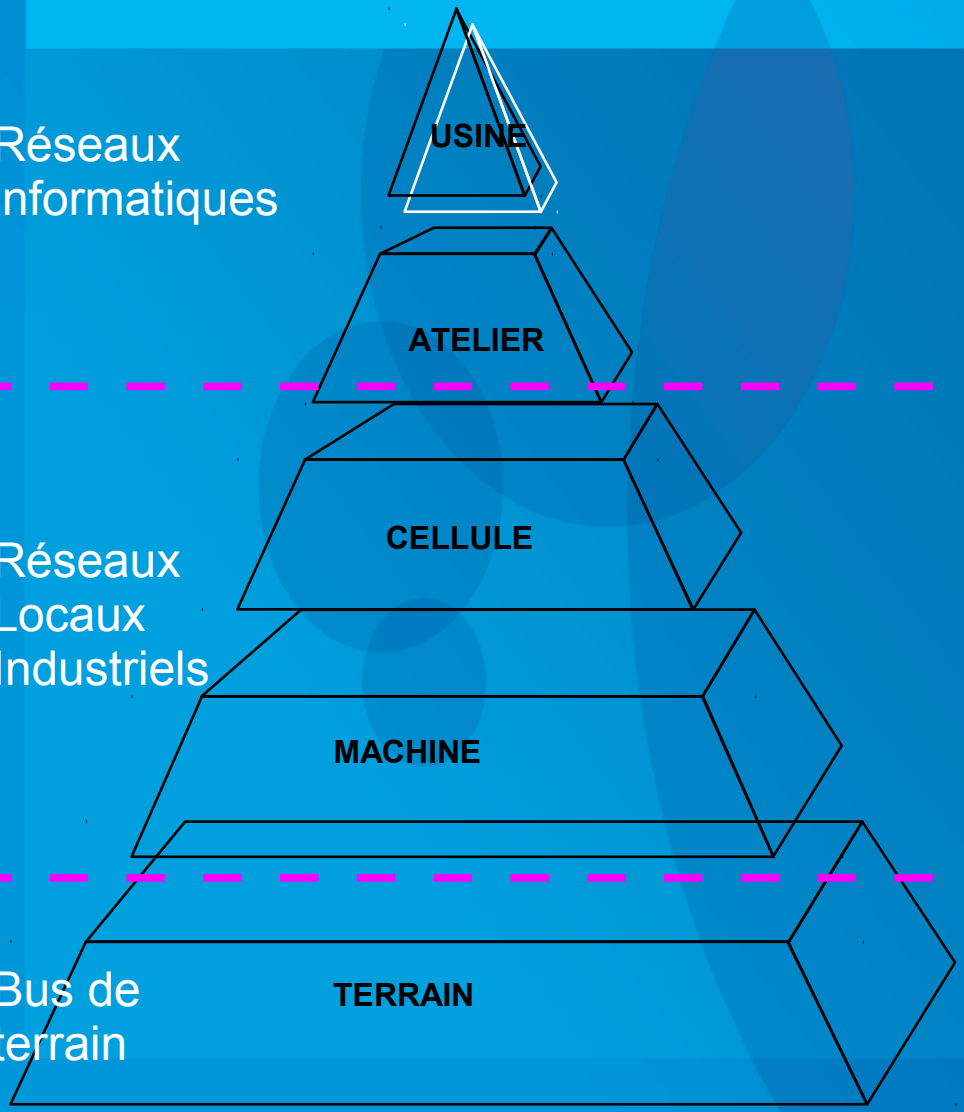
• **Réseau Local Industriel =** Réseau Informatique de type LAN, dont la topologie, les performances et la robustesse sont adaptées aux contraintes industrielles :

✔ **Electromagnétiques** → Robustesse du support de transmission et/ou des signaux électriques mis en jeu vis à vis des perturbations électromagnétiques (Couche 1 du modèle OSI)

✔ **Physiques** (poussières, chocs, températures, atmosphères explosives, vibrations) → Robustesse du support de transmission (Couche 1 du modèle OSI)

✔ **Temporelles** → Modes d'accès au médium favorisant des temps de réponse déterministes (Couche 2 du modèle OSI)

b - Les Différents Niveaux de Communication



Réseaux informatiques

Réseaux Locaux Industriels

Bus de terrain

Gestion, administration, organisation

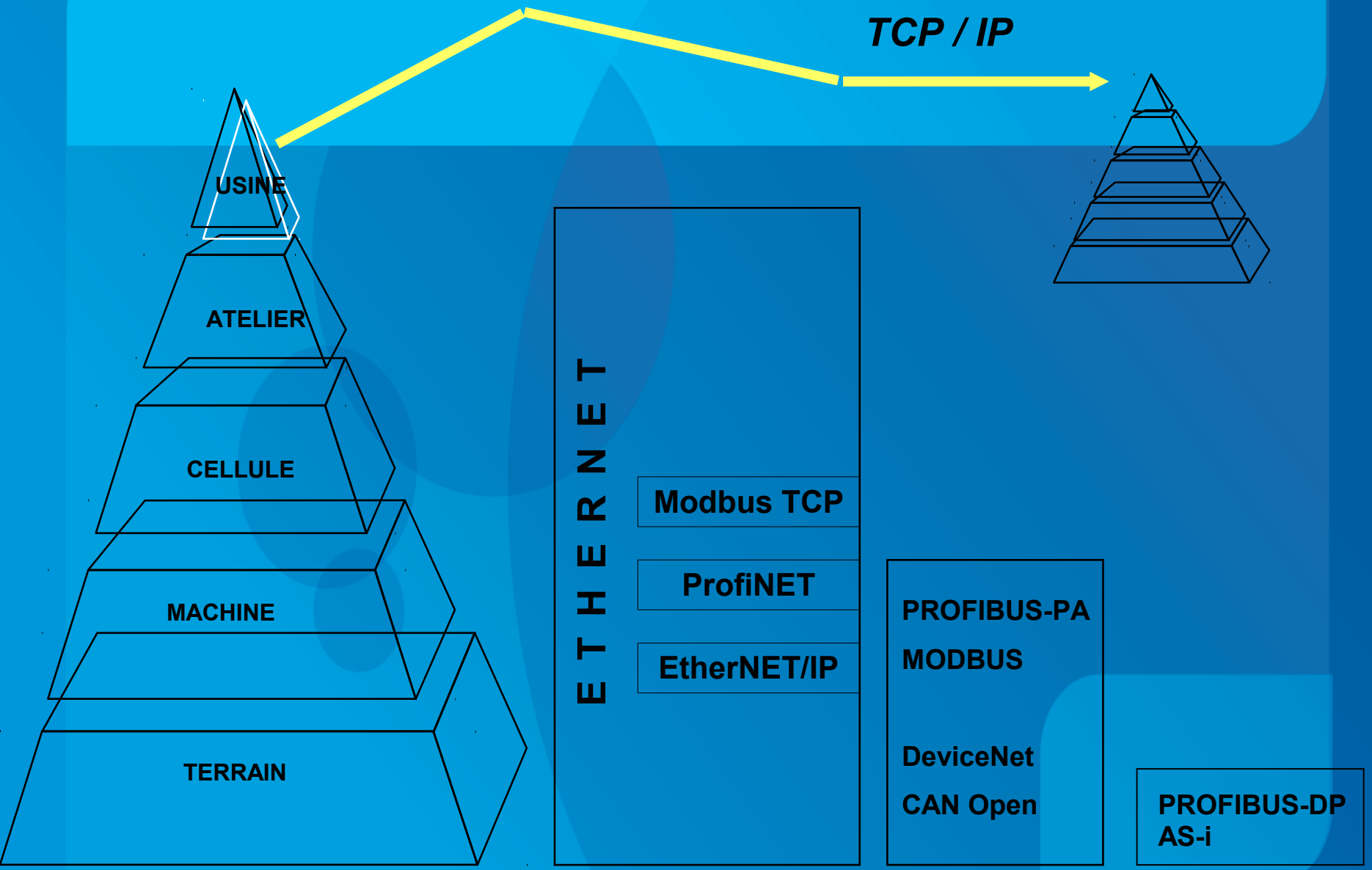
Gestion de production

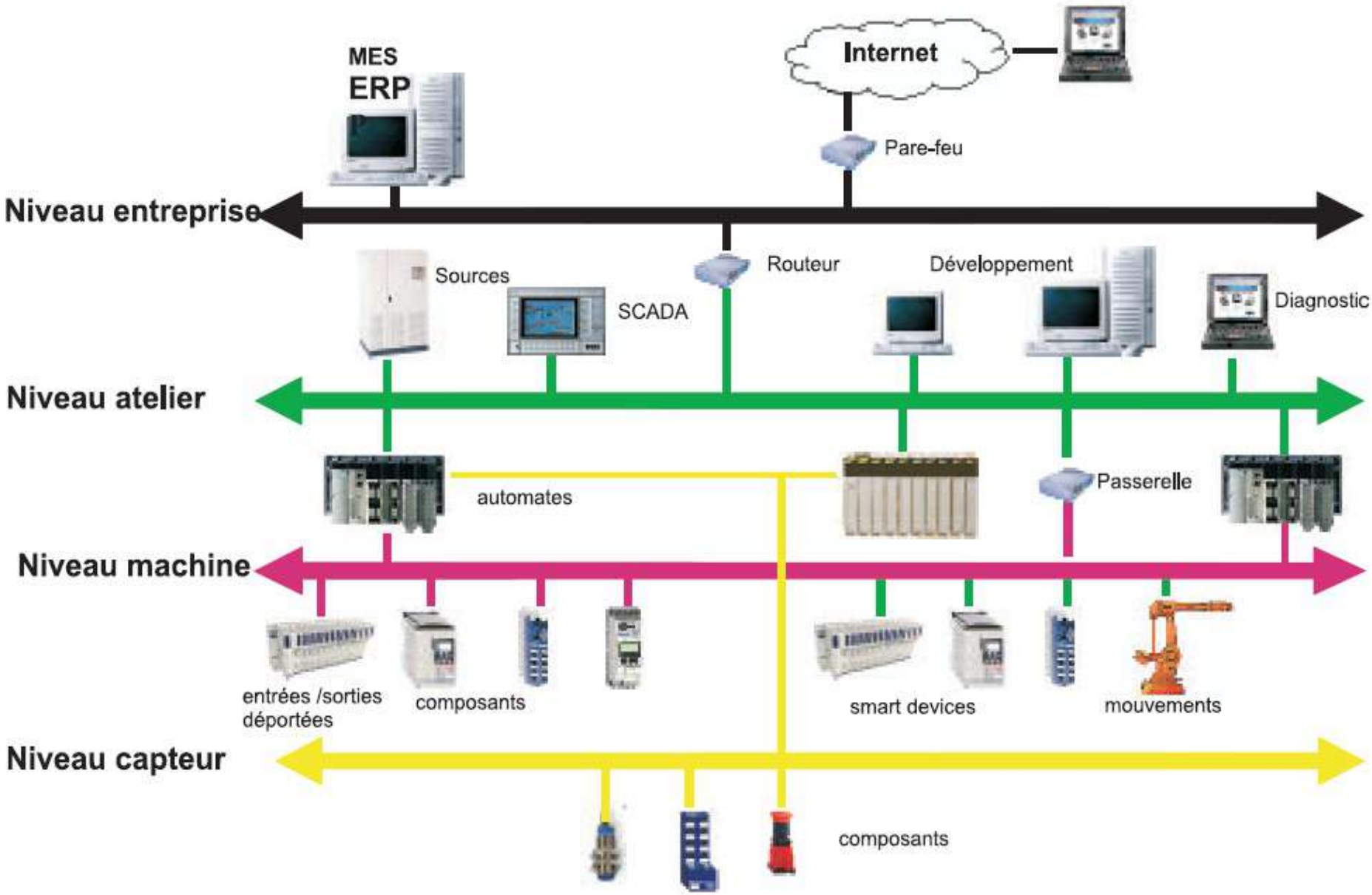
Supervision

Synchronisation

Acquisition / Action

c - Quelques Protocoles Standards :


TCP / IP
USINE
ATELIER
CELLULE
MACHINE
TERRAIN
ETHERNET
Modbus TCP
ProfiNET
EtherNET/IP
PROFIBUS-PA
MODBUS
DeviceNet
CAN Open
PROFIBUS-DP
AS-i


MES
ERP
Internet
Pare-feu
Niveau entreprise
Sources
SCADA
Routeur
Développement
Diagnostic
Niveau atelier
automates
Passerelle
Niveau machine
entrées /sorties déportées
composants
smart devices
mouvements
Niveau capteur
composants

d - Intérêts des RLI & BT

✔ **INTEROPERABILITE** : plusieurs systèmes, identiques ou radicalement différents, peuvent communiquer

✔ **REPARTITION de l'INTELLIGENCE** : L'algorithme est éclaté à travers les différents équipements communicants de l'automatisme : il s'en suit une fiabilité est des performances accrues. (processeurs « moins surchargés »)

✔ **DISTRIBUTION des Informations :**

- Remontée des informations vers la supervision, Traitement des alarmes centralisé...
- Mise à jour des programmes, firmwares depuis un poste centrale ou via internet
- Télécontrôle, Télémaintenance etc...

➢ Technologiques :

✔ **PRODUCTIVITE ACCRUE** : Grâce aux gains de performances de l'automatisation

✔ **REDUCTION DES COUTS DE MAINTENANCE et des DUREES d'ARRÊT de PRODUCTION** : Grâce aux outils de maintenance développés sur les réseau, la localisation des dysfonctionnements devient plus aisée.

✔ **REDUCTION des COUTS de CABLAGE :** Les bus de terrain de bas niveau permettent le raccord des capteurs & actionneurs du système sur un bus → Dépenses moindres en câblage et temps d'installation réduit.

2- Contraintes "temps réel"

Les différents niveaux de communication entre les composants d'un système automatisé ***impliquent des contraintes temporelles*** différentes, mais d'une façon générale ***plus sévères*** que pour un réseau informatique “tertiaire”.

Le diagramme suivant nous montre que les contraintes de temps sont ***d'autant plus critiques que l'on se rapproche du niveau « terrain »*** de la pyramide.

Niveau supervision

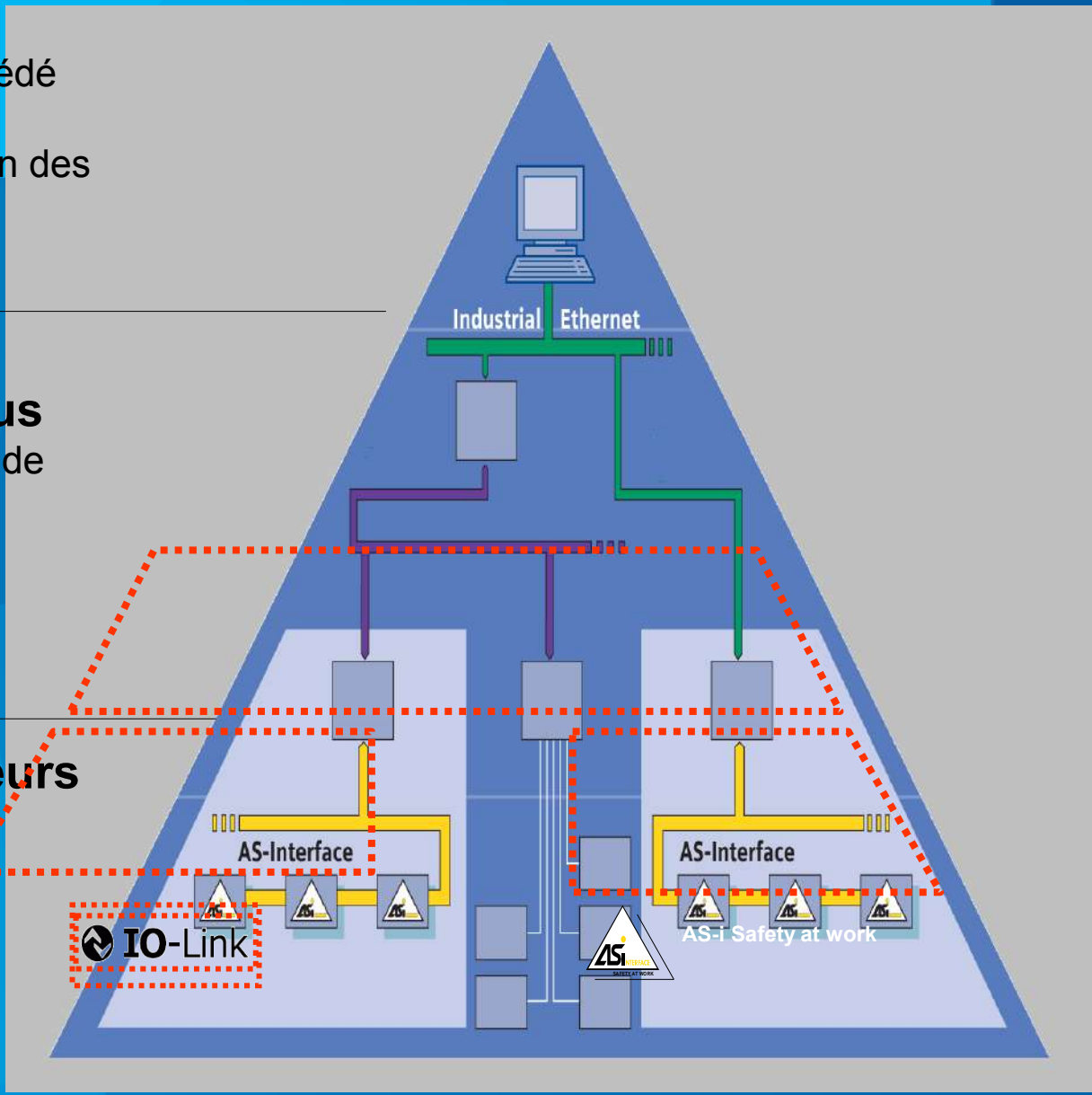
- Conduite et surveillance de procédé
- Gestion de production
- Gestion d'entreprise (Planification des Ressources)

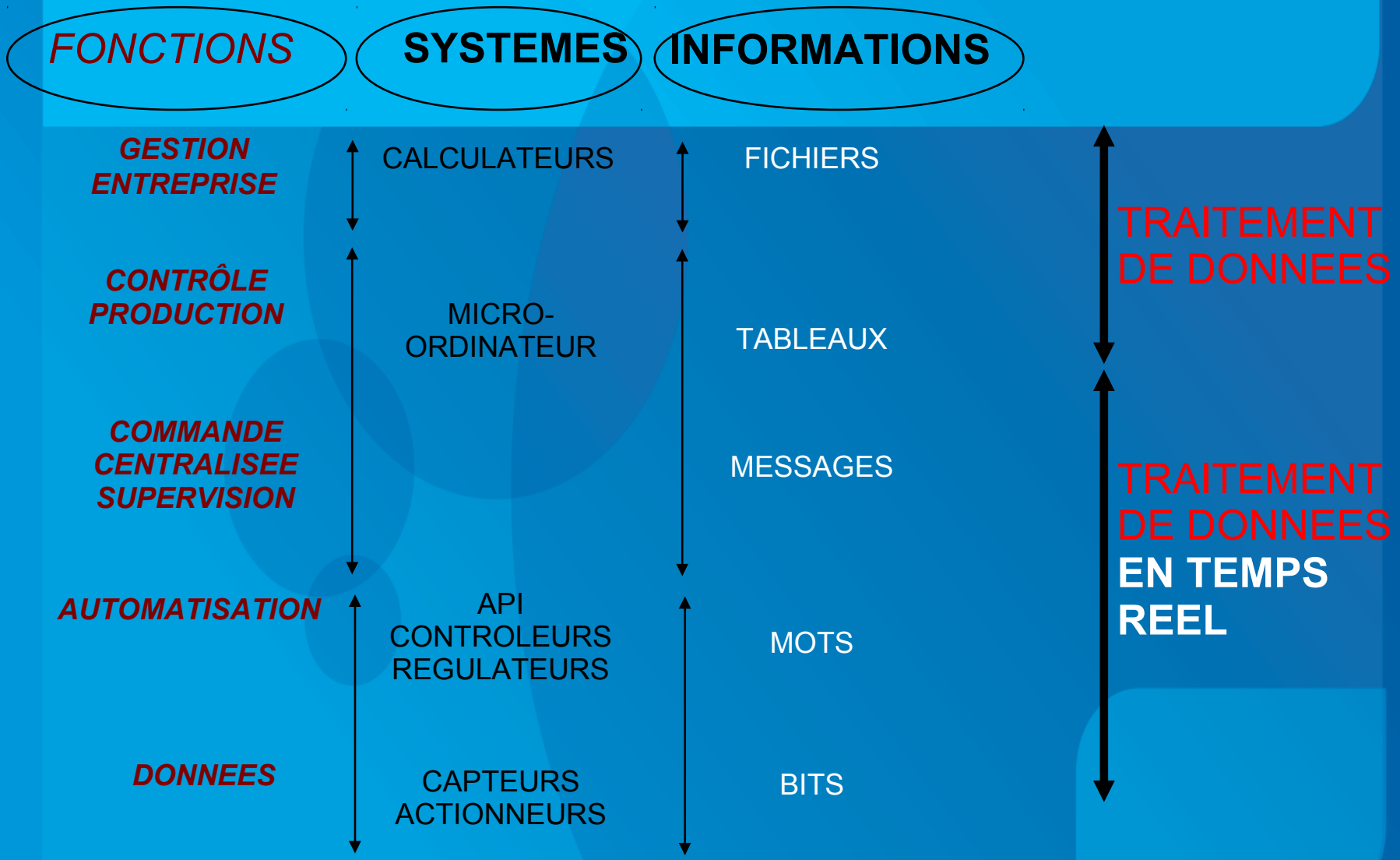
Niveau terrain et processus

- Contrôle des procédés, réseaux de terrain, API, Interface Homme-Machine, commande numérique

Niveau actionneurs / capteurs

- Faible volume de données, avec de nombreux appareils connectés et une exigence de temps réel




FONCTIONS
SYSTEMES
INFORMATIONS
GESTION ENTREPRISE
CONTRÔLE PRODUCTION
COMMANDE CENTRALISEE SUPERVISION
AUTOMATISATION
DONNEES
CALCULATEURS
MICRO-ORDINATEUR
API CONTROLEURS REGULATEURS
CAPTEURS ACTIONNEURS
FICHIERS
TABLEAUX
MESSAGES
MOTS
BITS
TRAITEMENT DE DONNEES
TRAITEMENT DE DONNEES EN TEMPS REEL

3- Modèle de Référence des RLI

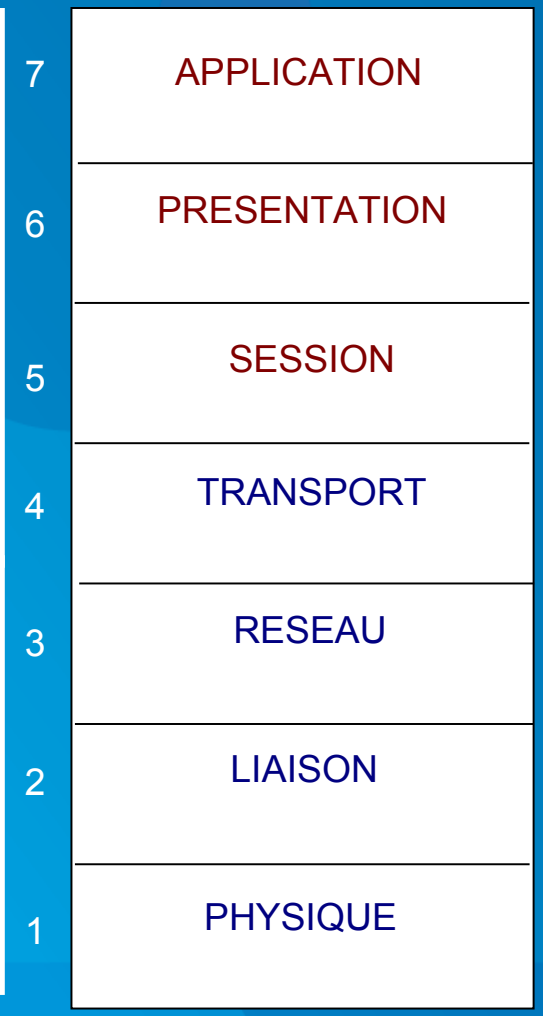
a – Rappel : Modèle OSI :

Couches hautes :

Traitement des informations relatives à la gestion des informations entres les systèmes informatiques

Couches basses :

Gestion des échanges entre les entités concernées + spécifications physiques



Interface avec l 'application

Représentation des données

Synchronisation du dialogue

Connexion entre les 2 hôtes distants

Routage=Acheminement des paquets

Construction des trames
+ Détection des erreurs

Codage des bits
+ Caractéristiques électriques

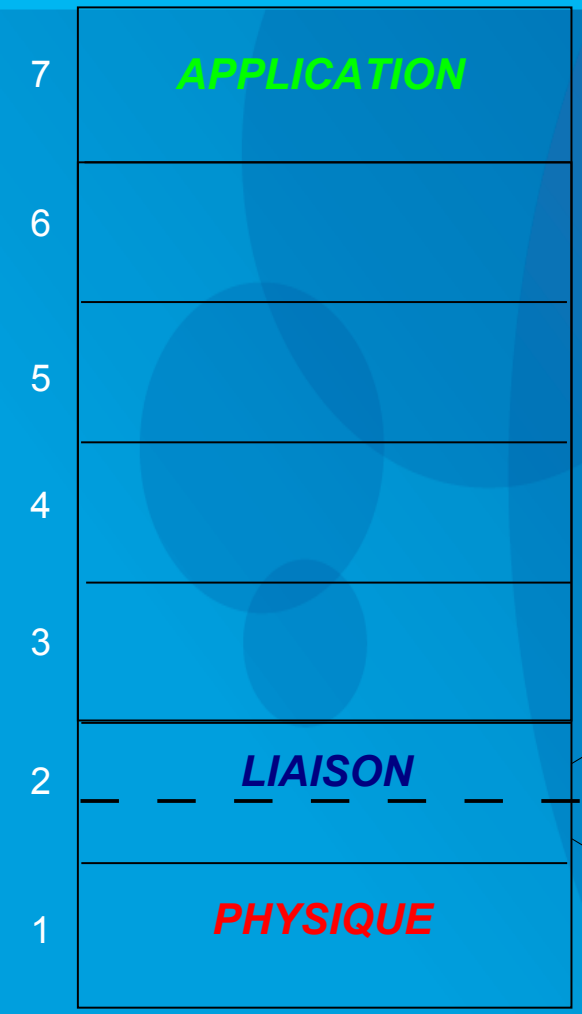
b – Adaptation au cas des RLI :

Le modèle OSI gère les grands réseaux à commutation de paquets. Le temps n'a pas été pris en compte.
Pour les réseaux locaux, la notion de temps réel est un point très important.
La **couche physique** est indispensable à la communication. La couche **liaison de données** aussi pour l'***accès au média*** et la ***détection des erreurs***.

Les couches réseaux et transport ont été définies pour gérer les problèmes des paquets qui transitent par des stations intermédiaires : elles n'ont plus lieu d'être pour les RLI car toutes les stations sont interconnectées.
La couche session permet l'échange d'une grande quantité d'information, ce qui n'est pas le cas pour les RLI.
La couche présentation peut être figée et non dynamique ce qui la rend transparente.

La couche application reste évidemment nécessaire.

Le bus de Terrain est donc basé sur une restriction du modèle I.S.O. à 3 couches :



Services offerts : données échangées, profils d'équipements...

La couche 2 est subdivisée en 2 sous-couches :

L.L.C : Logical Link Control → *détection des erreurs, mise en trame*

M.A.C : Medium Access Control → accès au réseau/bus, adressage

4- Critères de Comparaison

Voici quelques critères qui permettront de guider le choix d'un protocole de communication pour une application donnée :

- **Longueur maximale :** Longueur maxi du réseau en fonction du nombre de répéteurs et du type de médium utilisé.

- **Topologie :** Architecture physique et implantation des nœuds connectés au réseau, structure de câblage de toutes les stations.

- **Temps de réaction maximal** : Délai maximal possible qui peut survenir lors de l'envoi d'informations.
Ce temps dépend du temps de cycle, du nombre d'abonnés, de la longueur du réseau, du médium et de la vitesse physique de transmission

- **Nombre maximum d'équipements** : Nombre maximum de noeuds pouvant être connecté au réseau.

5- Compétences liées aux RLI

a – pour l'informaticien :

➔*Concevoir des programmes d'interfaçage entre un PC (Liaison série, ethernet) et un équipement communicant selon un protocole industriel.*

b – pour le concepteur de SE :

➔*Adapter les caractéristiques électroniques de la carte à celles des signaux véhiculés par le bus de terrain (couche physique : niveaux des signaux, connectique etc...)*
➔*Gestion logicielle du protocole (couche liaison + application)*

c – pour l'automaticien :

➔*Choisir un protocole de communication en fonction des contraintes imposées par le système automatisé.*

➔*Paramétrer des équipements communicants.*

➔*Gérer les bus et réseaux de terrain dans le programme de l'API.*

Ch 2 : Bus de Terrain ASi : Actuator Sensor interface

1 / Présentation

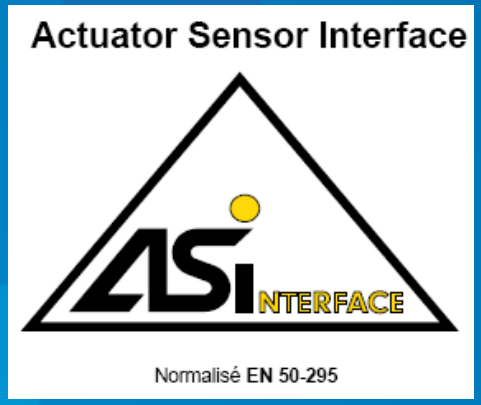
a - Un bus de niveau 0
b - Historique
c - Modèle OSI

2 / Couche Physique

a - Le câble ASi
b - Topologies
c – Signaux électriques &Alimentation du bus

3 / Couche Liaison

a – Système “maître-esclave”
b – Polling des esclaves
c – Mise en trame



4 / Couche Application

a – Catalogue des Requêtes
b – Catalogue des Réponses
c – Exemples de composants
d - Profils ASi
e - Plan mémoire du coupleur

5 / Evolutions

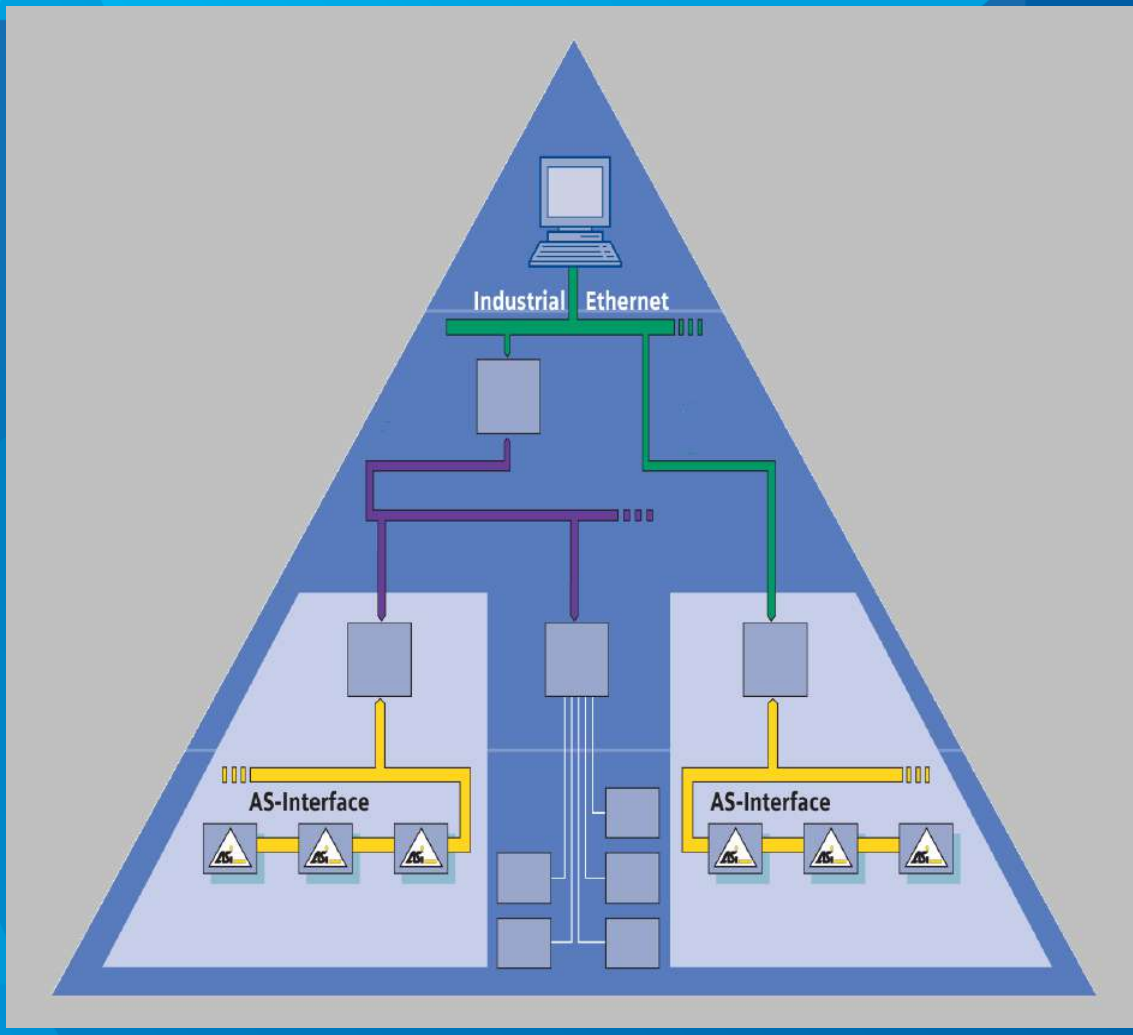
a – Asi v2,0 - v3,0
b - ASi-safe

1- Présentation du bus Asi :

a / Un Bus de niveau 0 :

Asi est un bus présent au Niveau 0 de la pyramide CIM, c'est à dire qu'il est chargé de faire communiquer entre Eux des nœuds à intelligence Limitée ou nulle (ex : capteurs TOR).

La taille des informations est Réduite au maximum (bits) En revanche, celles-ci doivent Être transmises avec certitude Dans des délais très courts (qq ms).



➔ Points clés :

... simplicité :

- transmission données et puissance sur 2 fils non blindés, non torsadés.

... flexiblilité :

- topologie libre et évolutive.

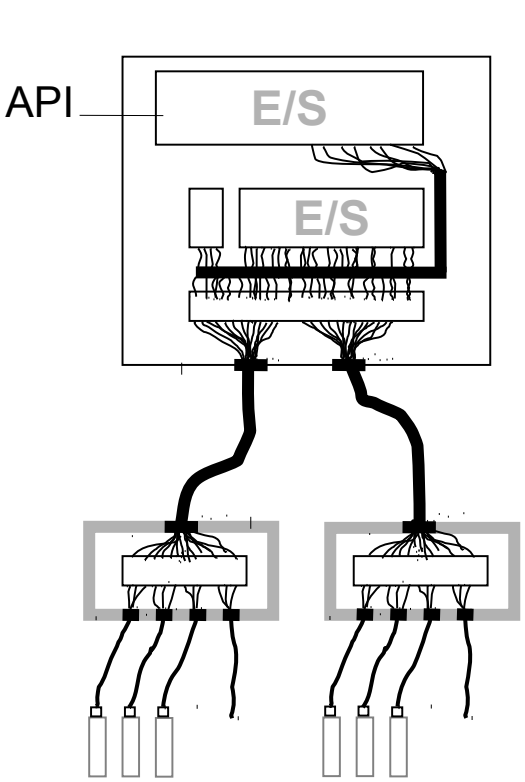
... sûreté :

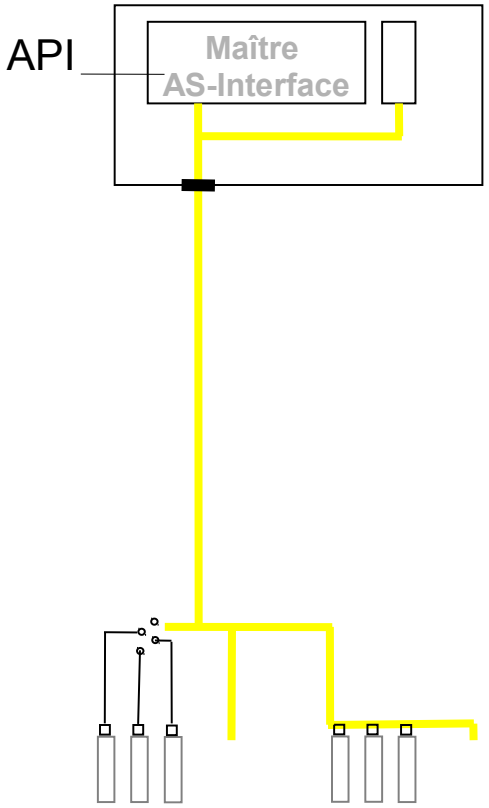
- Concept de transmission robuste et efficace.

... standardisation :

- standardisation électrique ; existence de “profils” (interchangeabilité)
- Composant unique pour tous les constructeurs.

Comparaison Câblage traditionnel / Câblage Asi :





Critères de Comparaison :

✔ Jusqu'à 31 esclaves (62 en version v2.0), pouvant comporter chacun jusqu'à 4 capteurs/actionneurs peuvent être raccordés à 1 seul maître (soit jusqu'à 496 E/S TOR par maître ASi)

✔ Temps de réponse déterministe : 5 ms (10 ms en v2.0) maximum

✔ Débit : environ 170 kB/s

✔ Longueur maximum du bus 100m (300m avec répéteurs)

b / Historique :

✔ Création en 1990 par un consortium de fabricants (Schneider Siemens...) de capteurs/actionneurs.

✔ Adopté par plus de 60 fabricants.

✔ Une large gamme de produits (E/S déportées, composants de distribution électrique BT, distributeurs pneumatiques, composants de sécurité etc...).

c / Modèle OSI :

Couche Application	Lecture / Ecriture simultanée de bits Lecture d'état Gestion d'adresse esclave Initialisation esclave Paramétrage et configuration esclave
Couches Présentation	VIDE
Couches Session	VIDE
Couches Transport	VIDE
Couches Réseau	VIDE
Couche Liaison de Données	Gestion Maître / Esclave : Question - Réponse, Diffusion. Service périodique de polling des esclaves (max 31) Service apériodique de configuration Contrôle d'erreur par bit de parité
Couche Physique	Modulation APM (Alternative Pulse Modulation) Télé-alimentation par le bus Câbles bifilaires spécifiques jaune, noir, rouge

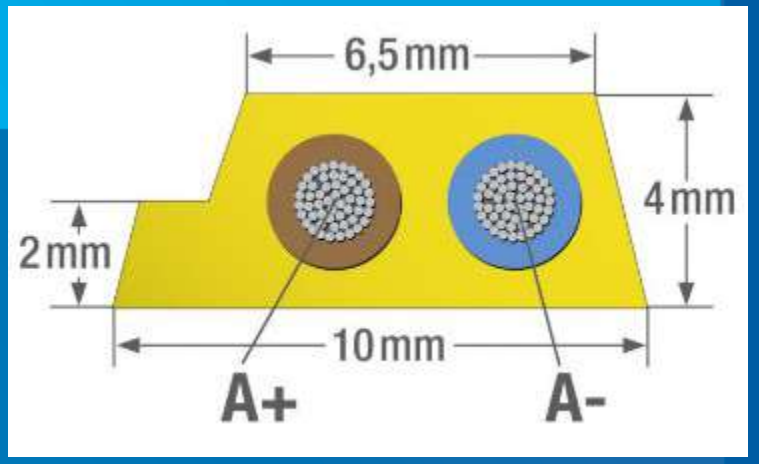
Réseau AS-i

2- Couche Physique :

a / Le Câble ASi :

Technologie prise vampire :

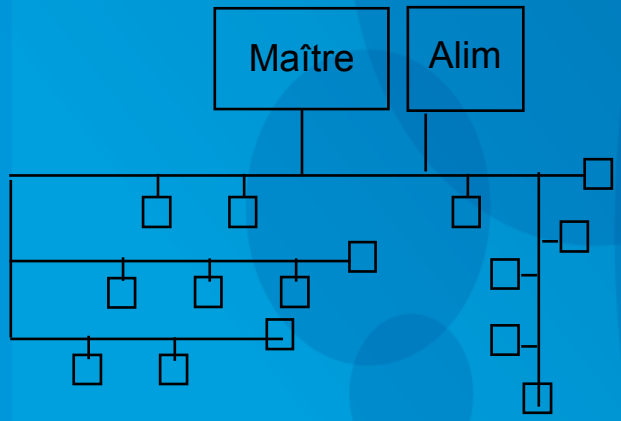
- connexion rapide et aisée des capteurs/actionneurs ou modules
- câble plat ***codé mécaniquement***
- deux fils pour les ***données et la puissance***
- Gaine isolante ***IP67 autocicatrisante***
- ***inutile de dénuder*** pour raccorder un nouveau composant sur le bus
- montage dans toutes les positions



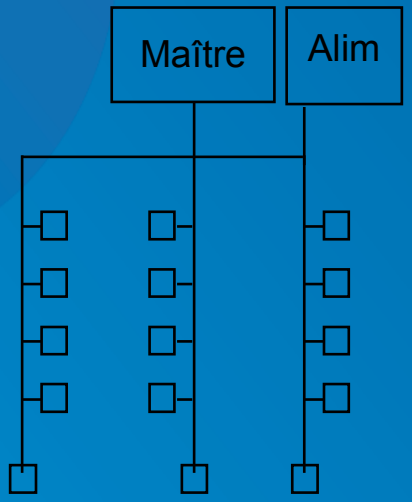
b / Topologies supportées :

Le câblage peut suivre toutes les topologies.

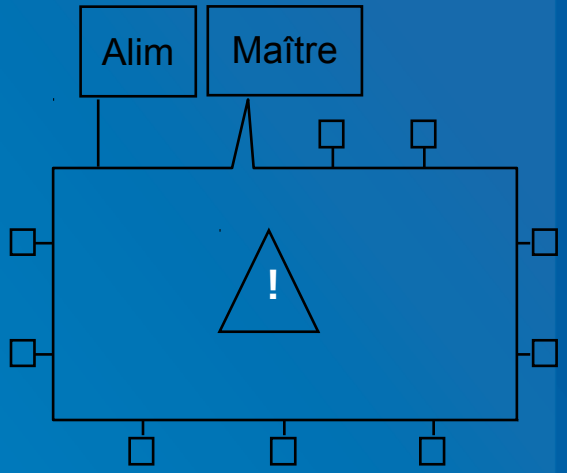
Pas de contrainte à l'installation ou lors des évolutions



Topologie arborescente



Topologie étoile



Topologie anneau

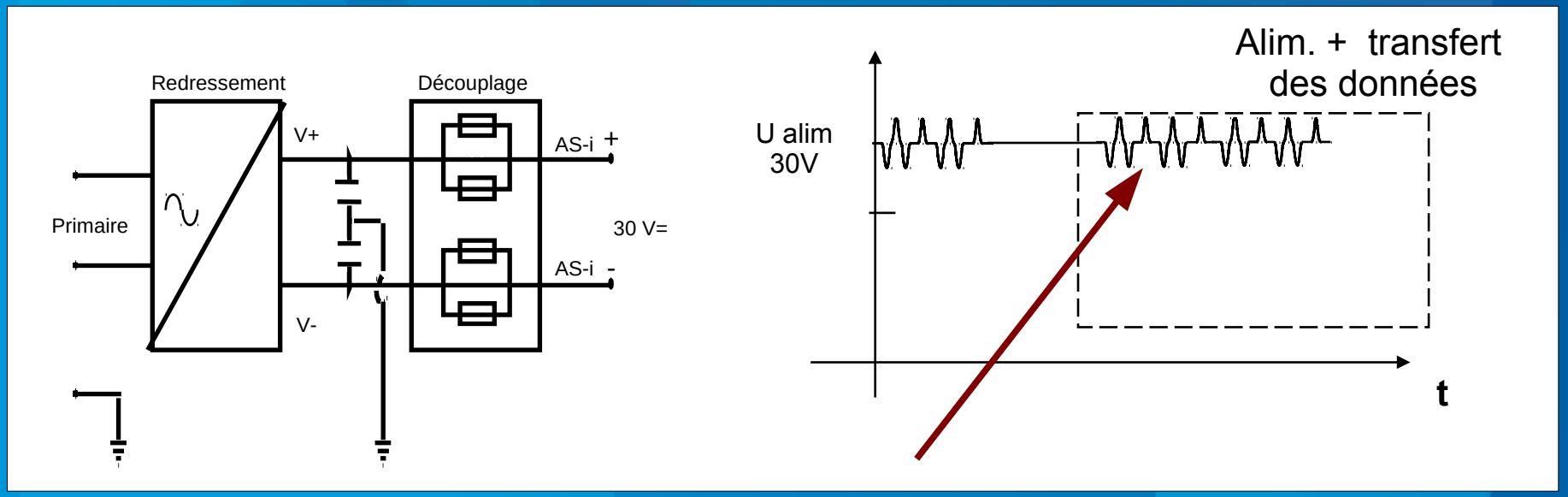
c / Alimentation et Signaux électriques sur le bus:

- Transmission par courants porteurs AS :

  Un seul câble pour alimenter les capteurs & actionneurs (jusqu'à 8A) connectés au bus, et le transfert des données.

- Alimentation en mode différentiel (TBTS):

  Bonne immunité aux perturbations (CEI 1000-4).



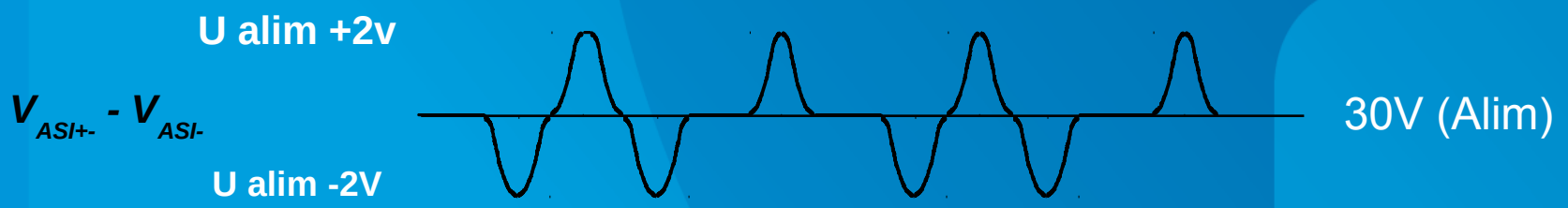
Codage Manchester par courant porteur

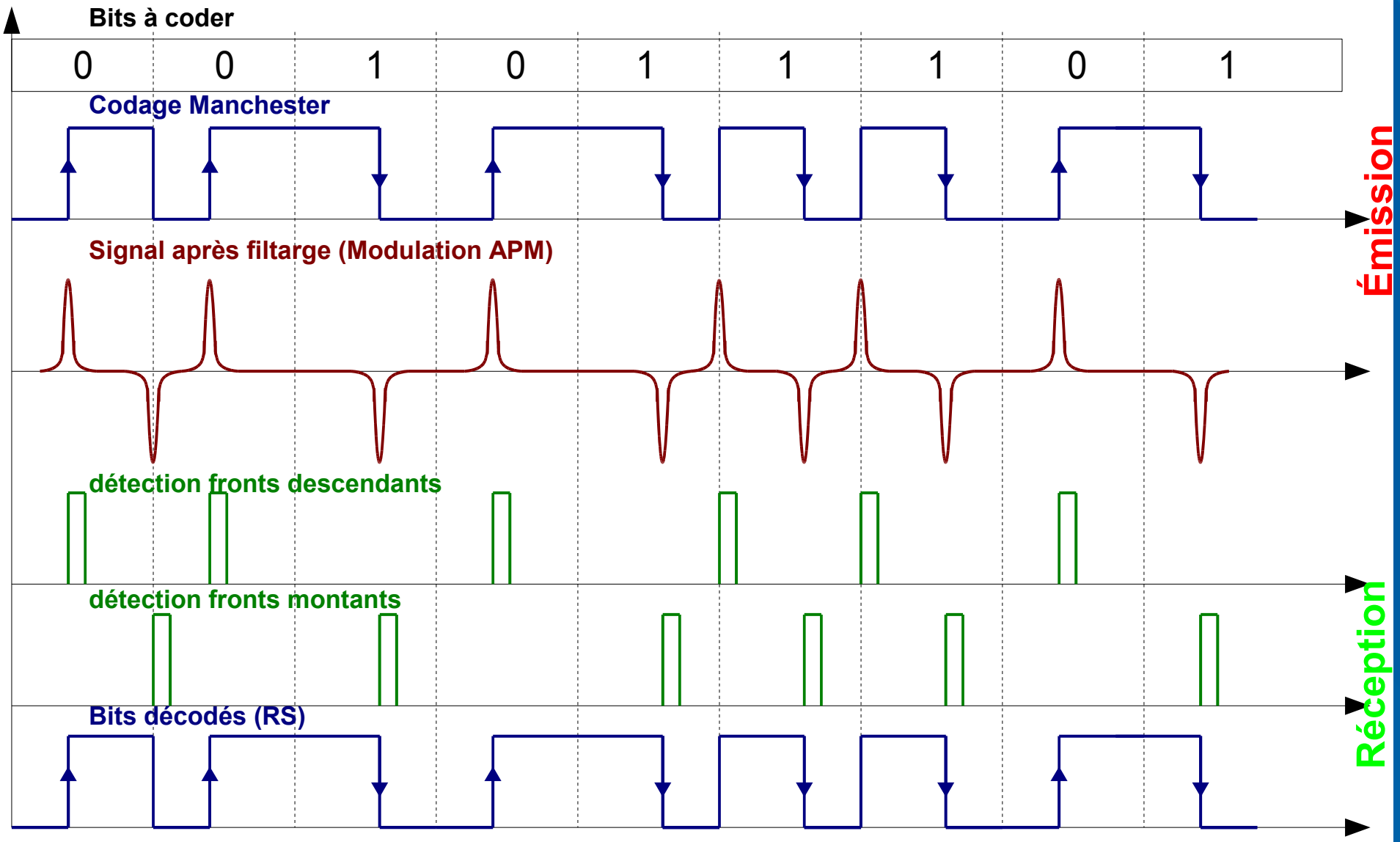
Codage des trames :

- Traitement par codage efficaces (MII, NRZ, APM et sin²).
- Forte redondance intrinsèque des signaux.

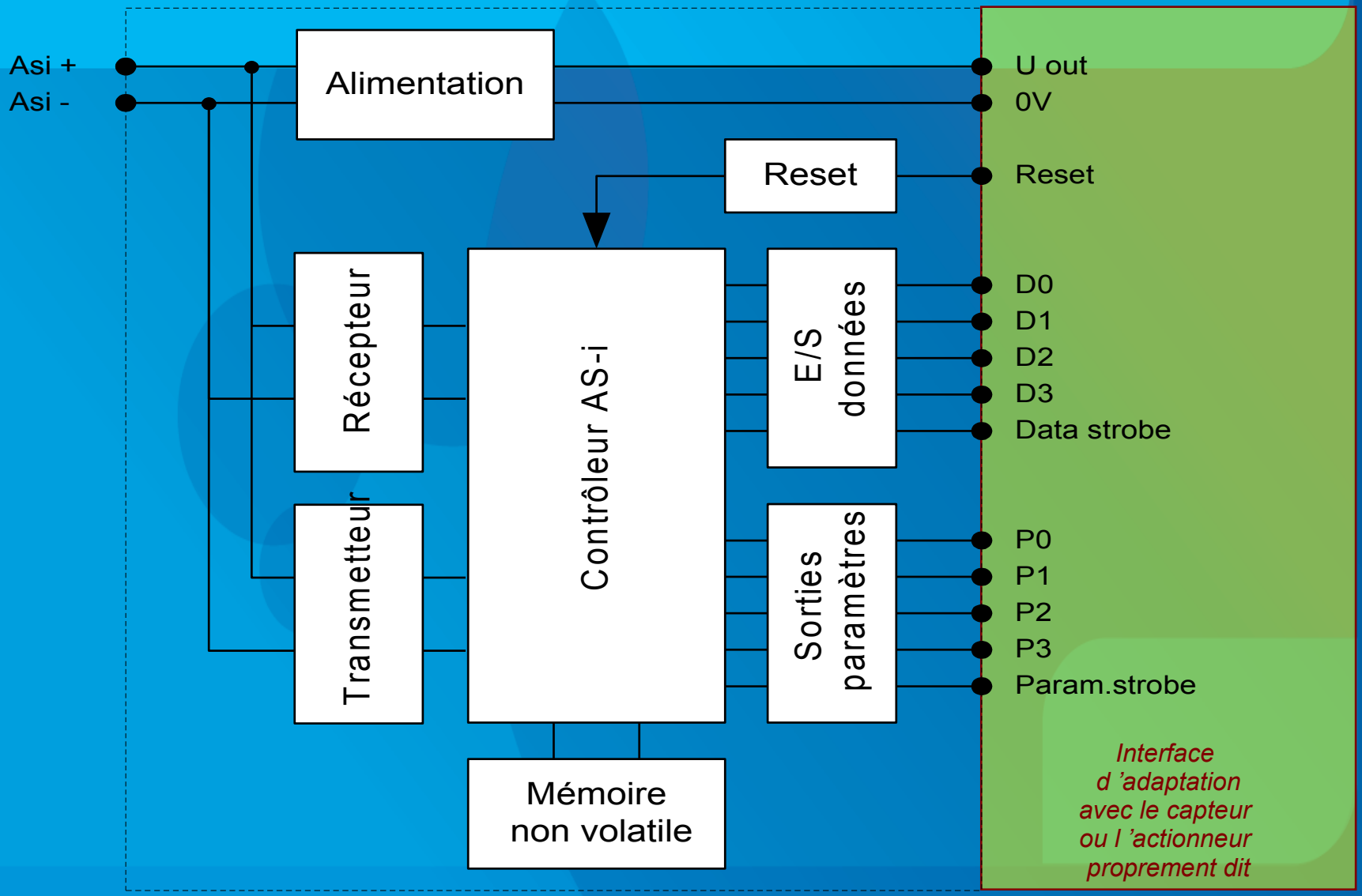
M II : Manchester II
NRZ : No return to zero
APM: Alternate pulse modulation

- **transmission différentielle → Bonne immunité aux perturbations**
- **MII + NRZ → Intégrité des données garantie.**
- **Filtrage sin² → Faible rayonnement**




Bits à coder
0 0 1 0 1 1 1 0 1
Codage Manchester
Signal après filtarge (Modulation APM)
détection fronts descendants
détection fronts montants
Bits décodés (RS)
Émission
Réception

Structure d'un esclave ASi :

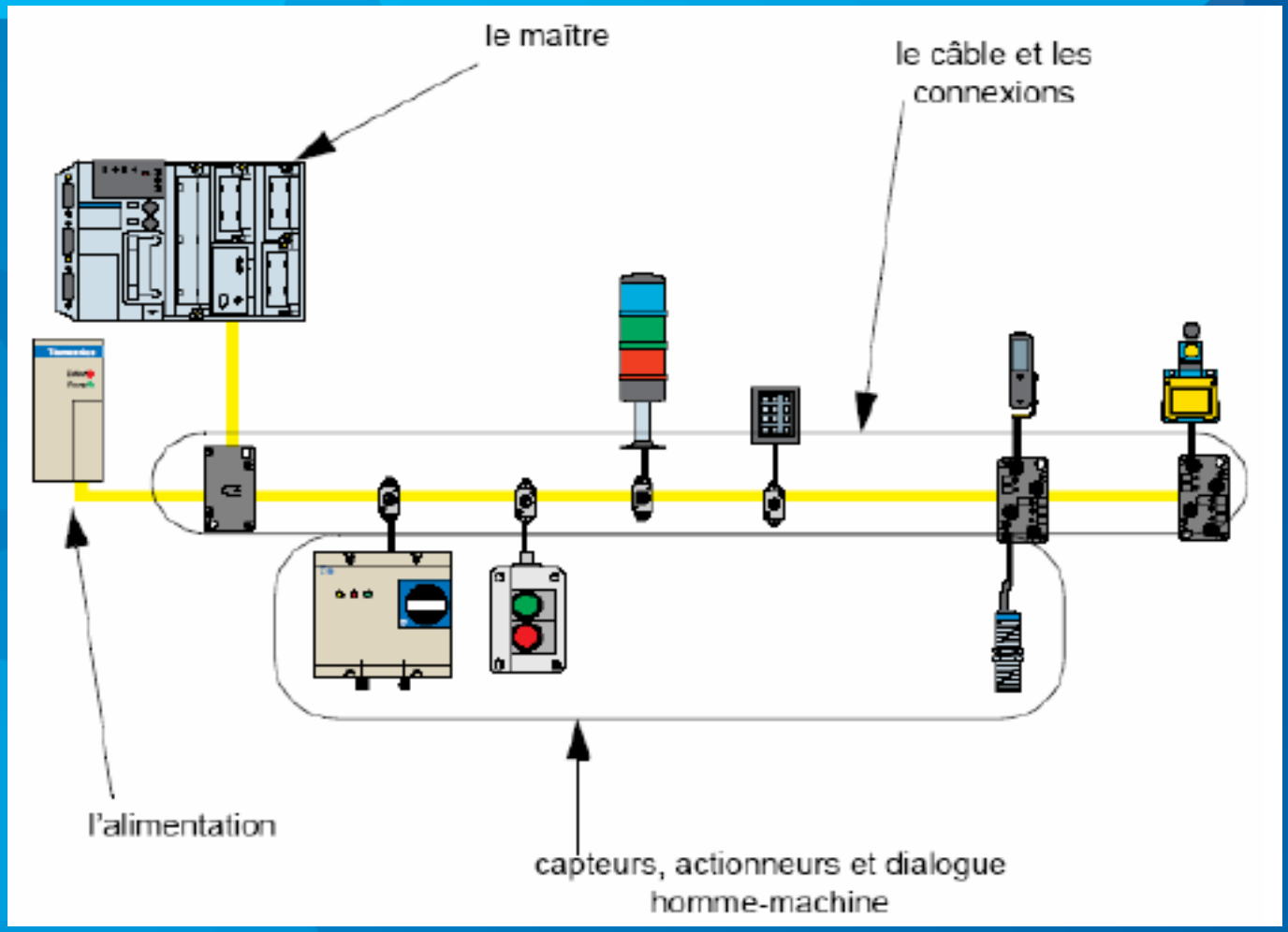


3- Couche Liaison

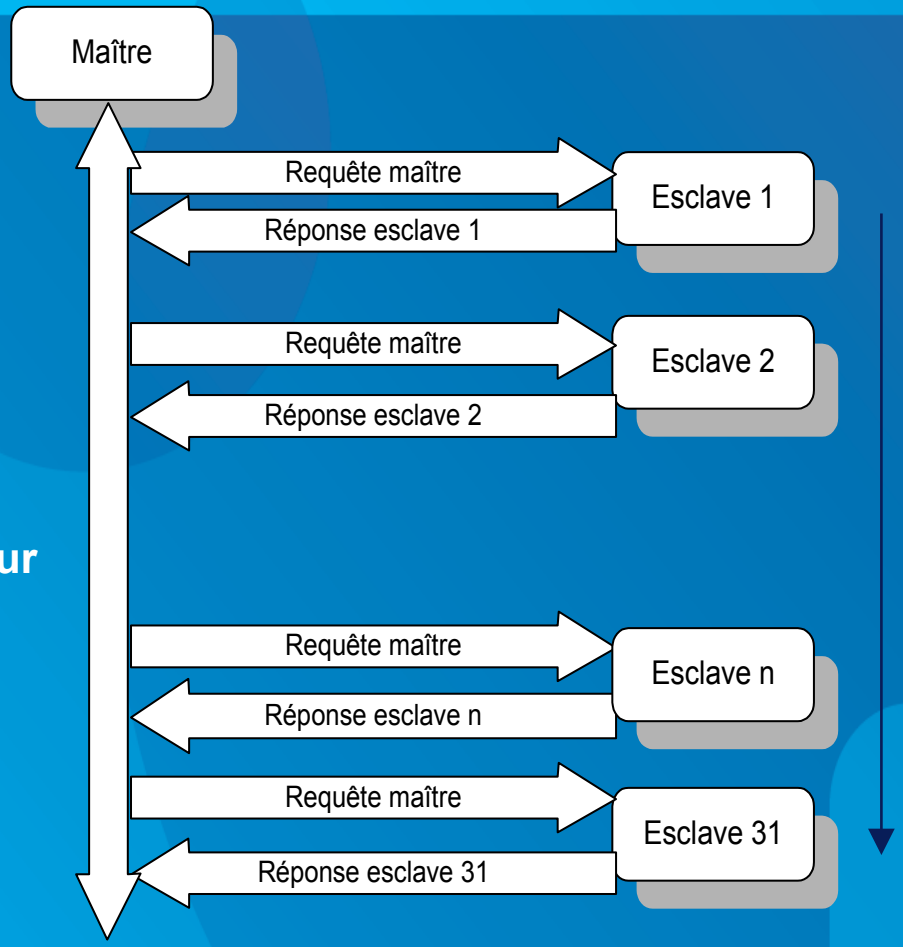
a / Système Maître / Esclave :

Asi est basé sur une architecture ***maître/esclave,*** avec 1 seul maître pâr bus qui peut initier une communication.

Cette technique permet de ***garantir la durée du temps de cycle*** (temps nécessaire à l'interrogation de tous les esclaves connectés au bus).



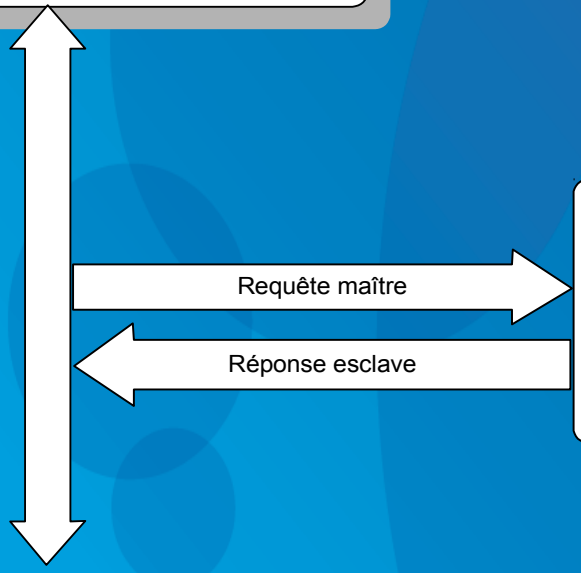
b / Polling des esclaves :



Temps de cycle caractéristique: 5 ms pour 31 esclaves

31 esclaves maximum

Maître:
polling cyclique des esclaves



Esclave AS-i: *adresse de **1** à **31***

mise à jour des sorties

acquisition de l'état des entrées

1 Esclave AS-i supporte:

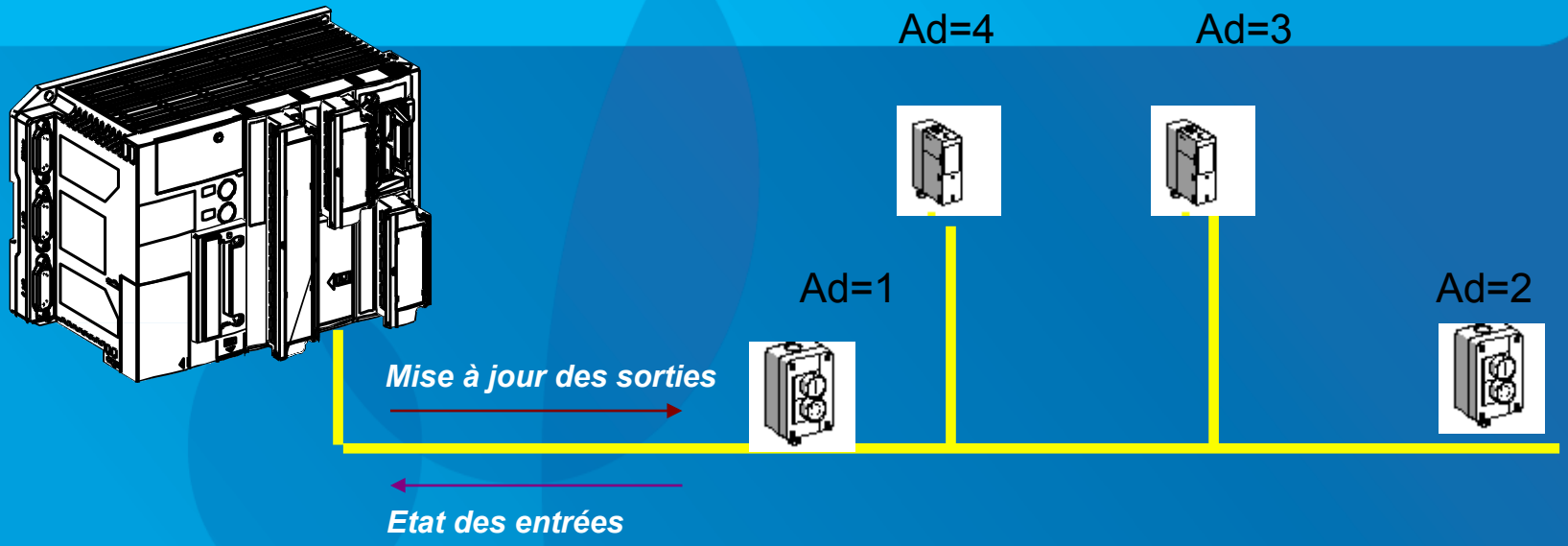
4 interfaces logiques
(entrées, sorties ou bidirectionnelles)

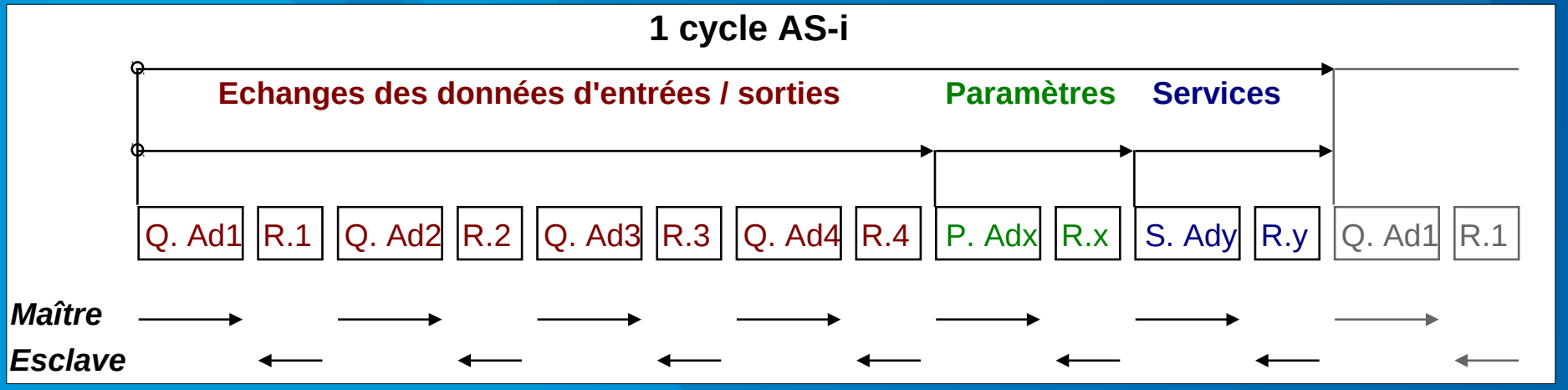
et au besoin,
4 paramètres
pour sélectionner un état particulier

- **Un seul mâitre Asi peut donc gérer jusqu'à 62 x 8 = 496 E/S T.O.R**

(en Asi v2.0 ou supérieur – 248 pour Asi v1.0)

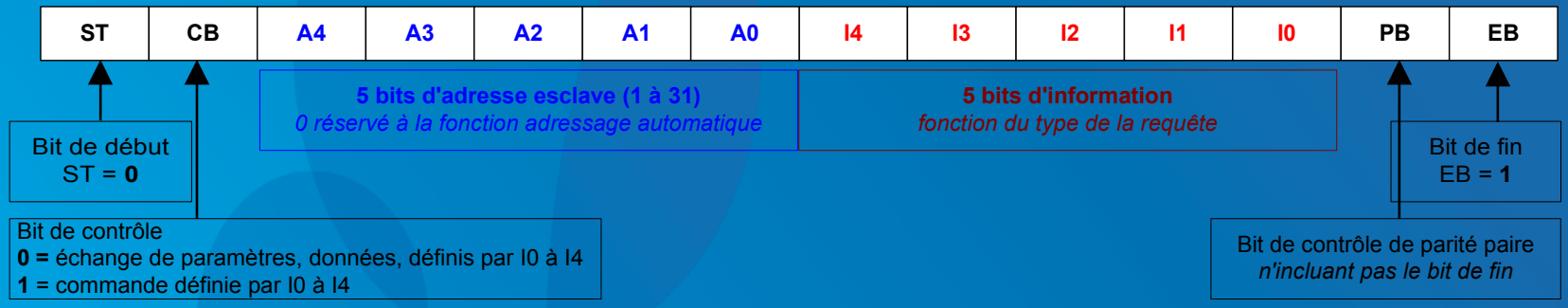
Les échanges Maître / Esclave :



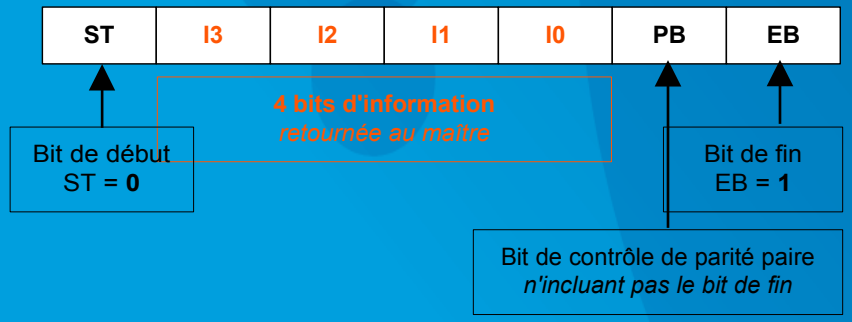


C / Mise en trame :

Requête du maître :



Réponse de l ’esclave :



Exercice :

Calculer la durée minimale d'un cycle de traitement AS-i lorsque le nombre maximum d'esclaves est connecté au maître et vérifier la durée annoncée par les caractéristiques du bus AS-i.

Le débit sur la ligne est maximum (170 kBps)

Réponse :

$T_{cycle} = (31 + 1 + 1) * (14 + 7) / 170.10^3 = 4{,}07\ ms < 5ms$

[Nombre de trames] * [Nombre de bits/trame] / [débit]

4- Couche Appplication.

a- Catalogue des requêtes du maître :

Requêtes du maître

	CB	5 bits d'adresse esclave					5 bits d'information				
Echange de données	0	A4	A3	A2	A1	A0	0	S3	S2	S1	S0
Ecriture de paramètres	0	A4	A3	A2	A1	A0	0	P3	P2	P1	P0
Ecriture d'adresse	0	0	0	0	0	0	A4	A3	A2	A1	A0
Reset esclave	1	A4	A3	A2	A1	A0	1	1	1	0	0
Reset adresse	1	A4	A3	A2	A1	A0	0	0	0	0	0
Lecture I/O Configuré	1	A4	A3	A2	A1	A0	1	0	0	0	0
Lecture code ID	1	A4	A3	A2	A1	A0	1	0	0	0	1
Lecture Status esclave	1	A4	A3	A2	A1	A0	1	1	1	1	0
Lecture et reset Status esclave	1	A4	A3	A2	A1	A0	1	1	1	1	1

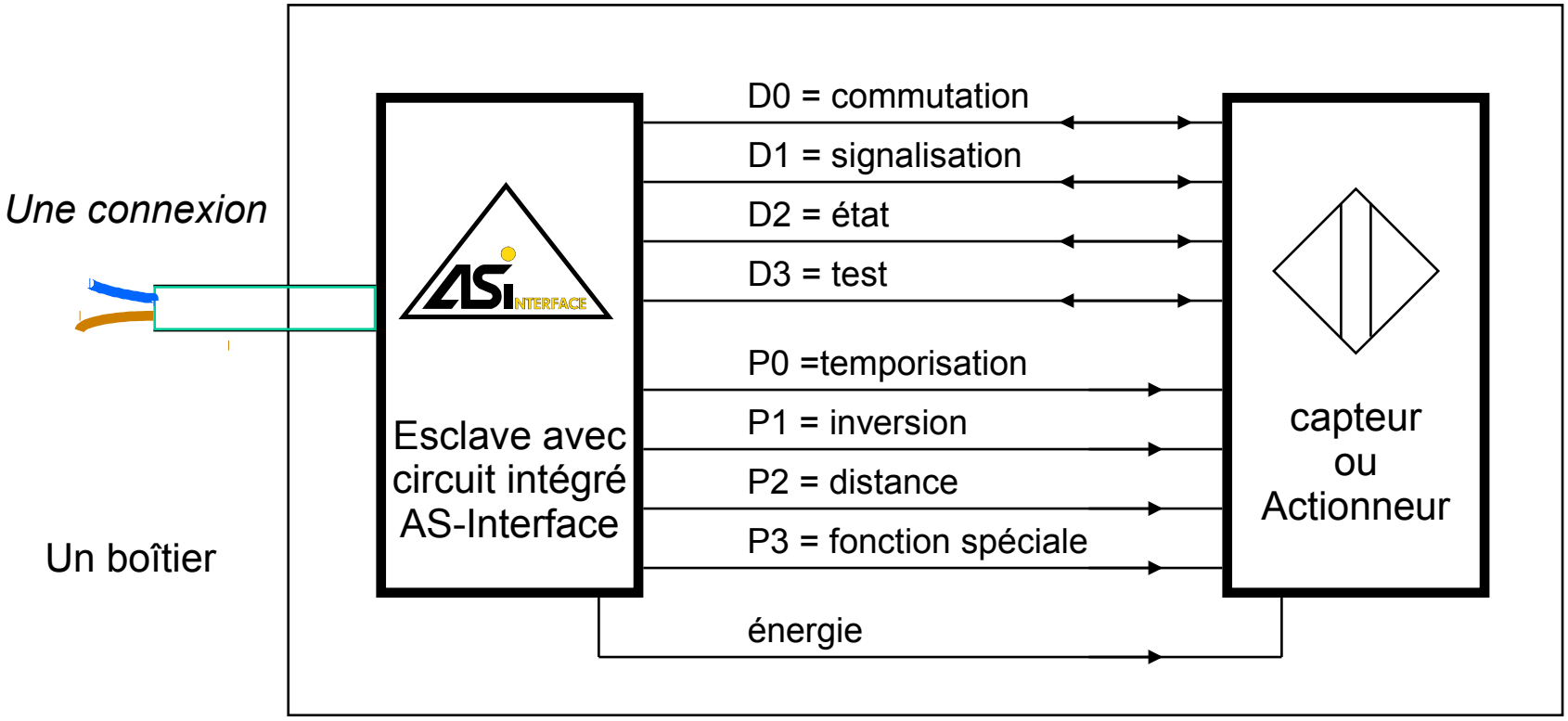
b- Catalogue des réponses des esclaves:

Réponse esclave

4 bits d'information				
E3	E2	E1	E0	Ei = entrées esclave
P3	P2	P1	P0	Pi = paramètres renvoyés en écho
0	1	1	0	Ack de l'esclave '6 Transaction 15 ms max
0	1	1	0	Ack de l'esclave '6 Transaction 2 ms max
0	1	1	0	Ack de l'esclave '6
C3	C2	C1	C0	I/O code esclave de '0 à 'F
ID3	ID2	ID1	ID0	ID code esclave de '0 à 'F
St3	St2	St1	St0	Sti = 4 bits d'états de l'esclave
St3	St2	St1	St0	Sti = 4 bits d'états de l'esclave avant RAZ

c- Exemples de composant ASi :

Structure :



Exemple de capteur :

Cellule Photoélectrique XUJK... (Schneider)

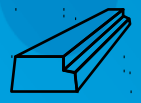
Paramètres :

→ Distance de détection (P1)
→ Dark-on / Dark-off (P2)

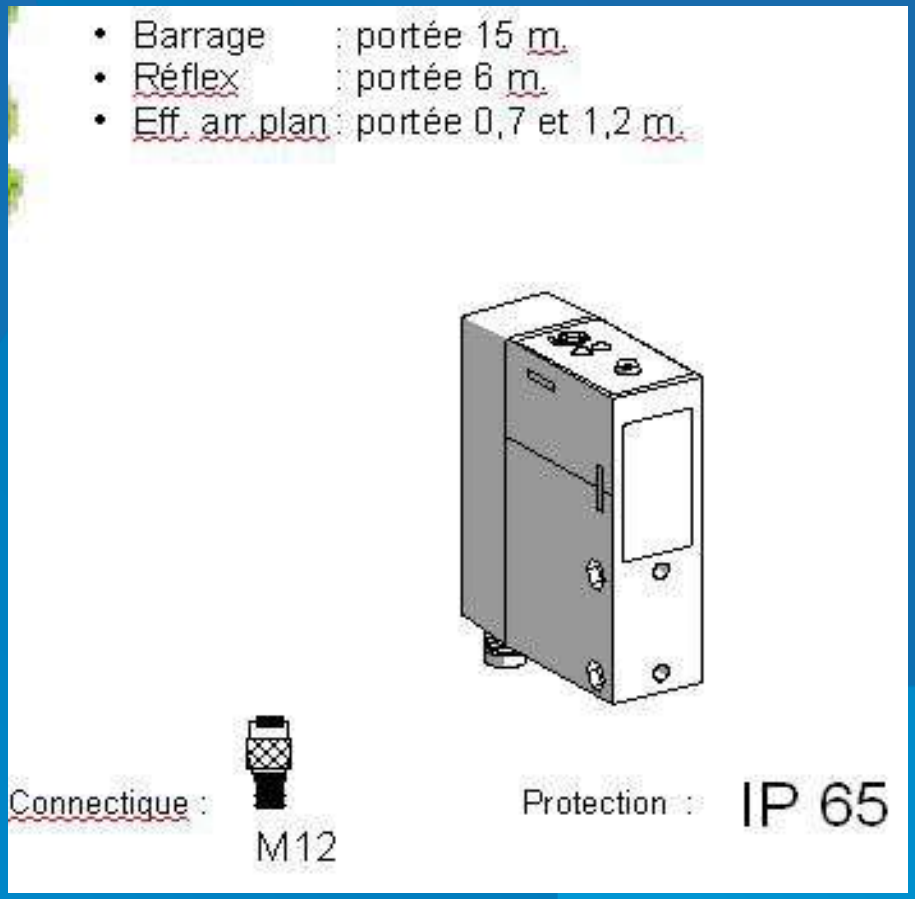
Données :

→ Détection (entrée I1)
→ Encrassement lentille (entrée I2)

Alimentation par le bus :

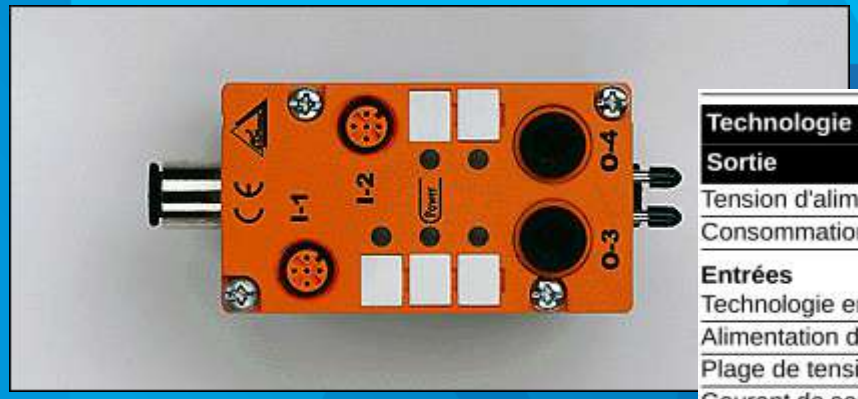


AS-i



Exemple d'actionneur (pré) :

Distriubteur électropneumatique AC2027 (ecomot)



Technologie	2 entrées / 2 sorties
Sortie	**échappement direct avec raccordement pour tube pne**
Tension d'alimentation [V]	26,5...31,6 DC
Consommation totale via AS-i [mA]	< 150
Entrées	
Technologie entrée	PNP
Alimentation des capteurs	AS-i
Plage de tension [V]	20...30 DC
Courant de sortie total pour toutes les entrées et sorties [mA]	100
Résistant aux courts-circuits	oui
Niveau du signal logique "haut" 1 [V]	> 10
Courant d'entrée, niveau haut/bas[mA]	> 3 / < 1,5
Sorties	
Système pneumatique	2 distributeurs 3/2 monostables

ecomat300

Système bus AS-interface

AC2027

AS-i AirBox
1

Interface AS-i pour les embases
câblage pour câble plat ou
presse-étoupe
Prises M12 x 1

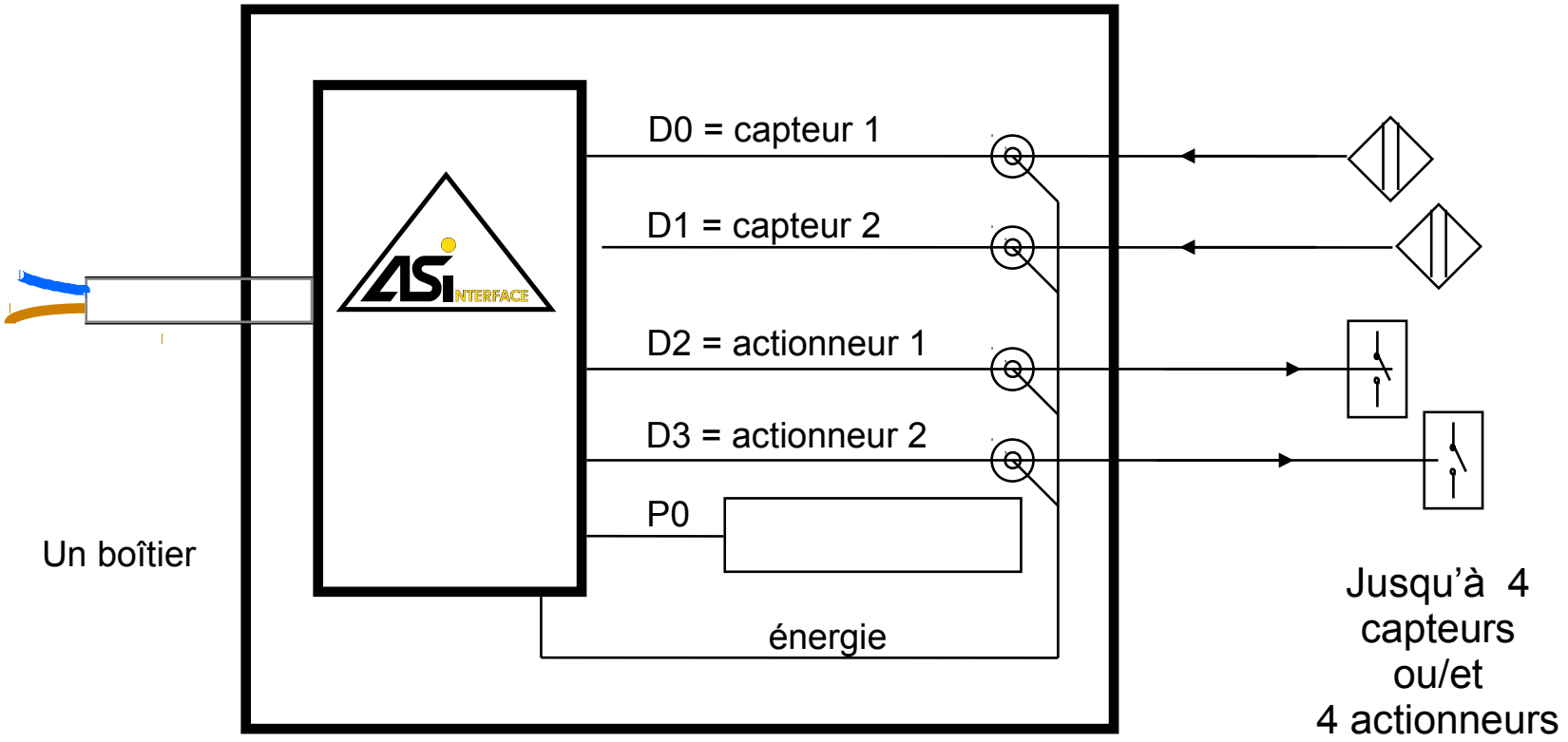
Raccord pneumatique via systèn
Legris
2 Entrées digitales

2 sorties pneumatiques

AS-i classification

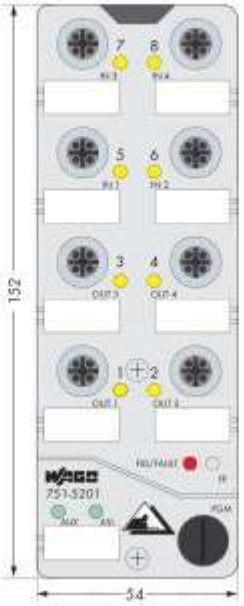
AS-i profile	S-3.F				
I/O configuration [hex]	3				
ID code [Hex]	F				
AS-i certificate	10501				
Data bits	**Data bit**	**D0**	**D1**	**D2**	**D3**
	In-/Outputs	I-1	I-2	O-3	O-4

interface AS-i pour capteurs/actionneurs conventionnels :



Esclaves AS-Interface avec indice de protection IP 67

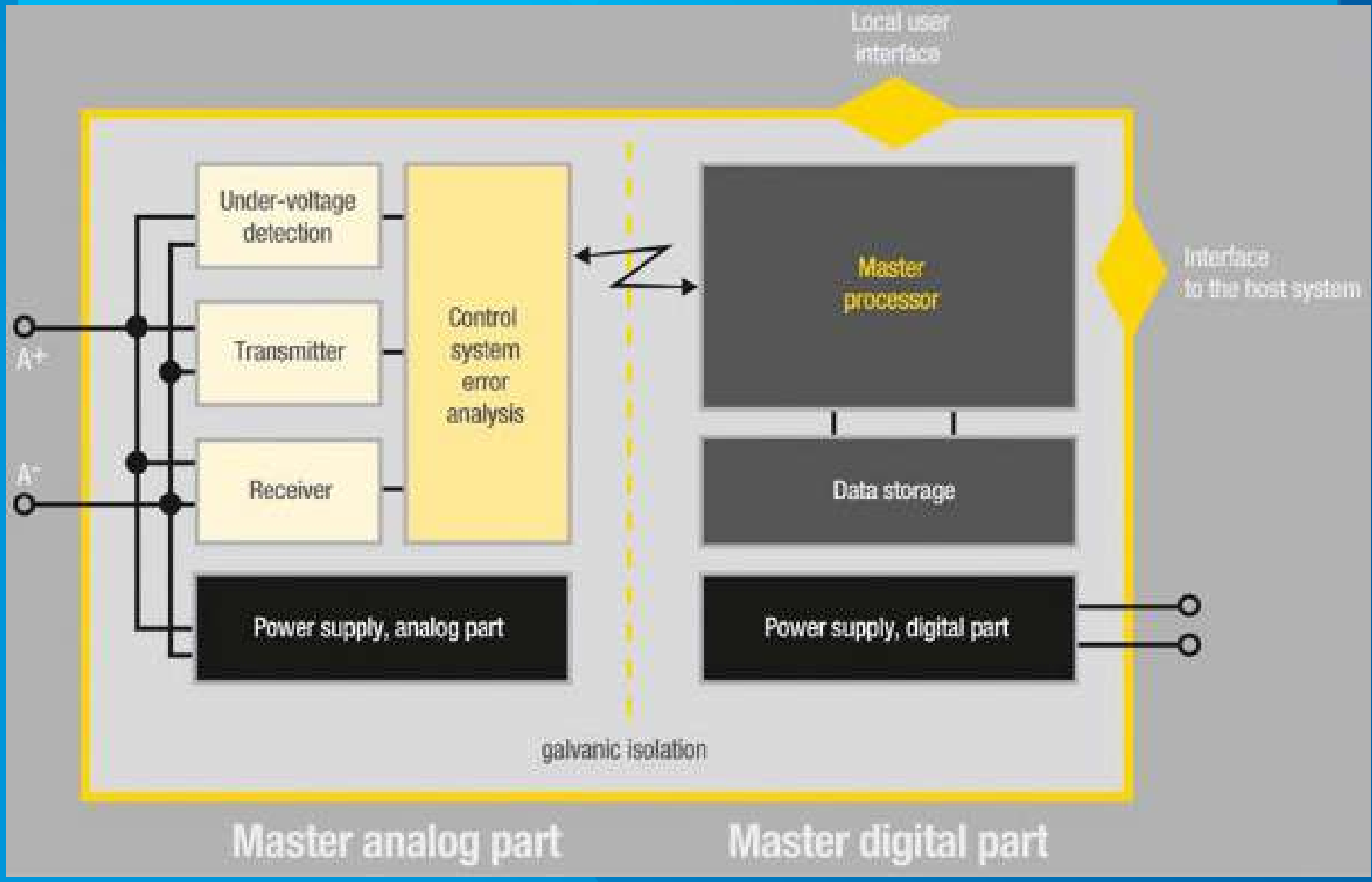
Module d'E/S digitales 4 E/4 S 24 V/2 A Single



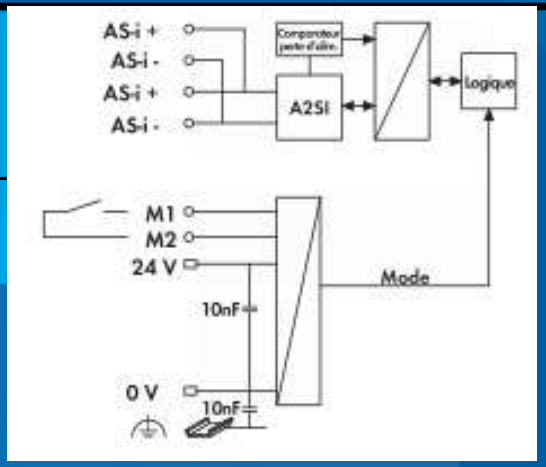
Données techniques

Tension de fonctionnement (AS-I)	26,5 V ... 31,6 V
Consommation de courant typ.	35 mA
Consommation de courant max.	50 mA (au démarrage), 280 mA y compris alimentation des capteurs
Version AS-Interface	3.0
Profil AS-Interface	S-7.F.E
Possibilité de mode d'adressage étendu	non
Code d'identification	7.F.F.E
Indications de la fonction	
LED indiquant la fonction « Marche » / « Arrêt »	jaune
LED indiquant une erreur	rouge
LED indiquant la fonction (ASI /AUX)	vert
Affectation des bits de données	
D0	bidirectionnel I-1/O-1
D1	bidirectionnel I-2/O-2
D2	bidirectionnel I-3/O-3
D3	bidirectionnel I-4/O-4

Structure d'un maître Asi :

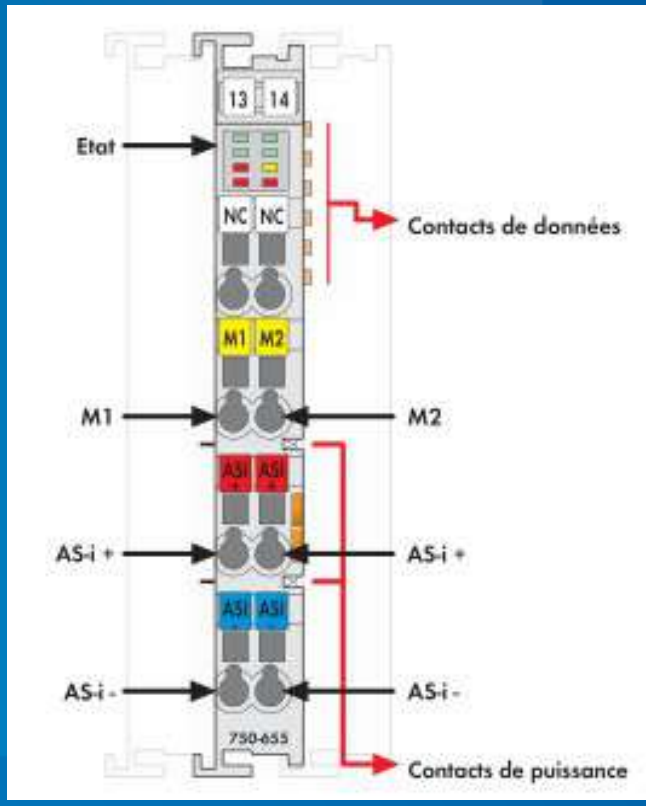


Exemple de maître Asi pour API : Borne WAGO 750-655 pour contrôleur 750-xxx



Données techniques

Spécification (AS-I)	2.1
Nombre d'esclaves	jusqu'à 62
Consommation de courant (AS-I)	40 mA
Alimentation	26,5 V ... 31,6 V
Longueur max. du bus (AS-I)	100 m, avec répéteur 300 m
Temps de cycle (AS-I)	0,3 ms ... 10 ms, en fonction du nombre d'esclaves
Configuration	via les données, WAGO I/O-Check (à partir de version 2.1)
Canal de communication	1
Consommation de courant max. (interne)	55 mA
Alimentation	par système interne DC/DC
Séparation galvanique	500 V (système / alimentation / AS-I)
Unité d'adressage	12 ... 48 octets max., configurable y compris 1 octet commande/ état



Console d'adressage ASi:

- Adressage des esclaves Asi (Modes A/B),

- Fonctions de Diagnostique (Tension Bus etc...),

- Lecture des profils ASi,

- etc...

d - Profils ASi.

Chaque équipement ASi dispose d'un profil, qui va permettre de garantir l'***interopérabilité*** des produits ASi entre fabricants.

Ce profil prend la forme d'une valeur (1 octet), notée en hexadécimal, et définit le comportement de l'esclave sur le bus :

- Données d'entrées et de sorties échangées
- Paramètres accessibles
- Type de capteur / actionneur

Ci-après, deux exemples de profils éxistants.

- Profil 1.1 : capteurs inductifs :

E/S		Niv.	Définition	Hôte
D0 = IN	Signal	0	Signal ouvert	Entrée
		1	Signal fermé	
D1 = IN	Warning	0	Alarme ON	Entrée
		1	Alarme OFF	
D2 = IN	Disponibilité	0	Indisponible	Entrée
		1	Disponible	
D3 = OUT	Test	0	Test inactif	Sortie
		1	Test activé	

E/S		Niv.	Définition
P0	Tempo	0	Tempo ON
		1	Tempo OFF
P1	Inversion D0	0	D0 inversé
		1	D0 normal
P2	Fréquence	0	Fréquence basse
		1	Fréquence haute
P3	Fonction spéciale	0	Fonction spéciale
		1	Fonction normale

- Profil B.1 : 2 actionneurs avec retour :
Ce profil sera par exemple utilisé pour une commande de vérin.

E/S		Niv.	Définition	Hôte
D0 = OUT	Actionneur 1 signal	0	Actionneur inactivé	Sortie
		1	Actionneur activé	
D1 = OUT	Actionneur 2 signal	0	Actionneur inactivé	Sortie
		1	Actionneur activé	
D2 = IN	Capteur 1 signal	0	Signal ouvert	Entrée
		1	Signal fermé	
D3 = IN	Capteur 2 signal	0	Signal ouvert	Entrée
		1	Signal fermé	

Paramètres		Niv.	Définition
P0	Chien de garde	0	Chien de garde actif
		1	Chien de garde inactif
P1	Verrouillage D0 et D1	0	D0 et D1 verrouillés
		1	D0 et D1 normaux
P2	RAZ distante	0	RAZ activé
		1	Fonctionnement normal
P3	Fonction spéciale	0	Fonction spéciale
		1	Fonction de base

e / Plan mémoire du coupleur.

Quel que soit l'API utilisé, le coupleur (Maître sur le réseau) ASi établi un plan mémoire image des entrées et des sorties, qui servira d'interface entre le bus ASi et le programme de l'API.
Le travail de l'automaticien consistera alors - outre l'adressage des esclaves - à déterminer les adresses CEI de chaque E/S connectée au bus, afin de pouvoir l'utiliser dans son programme.

Exemple de plan d'adressage pour une station Wago (borne ASi en 1° position) :

Bit :	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
%IW4	@2					@3			XXX				@1			
%QW4																
%IW5	@6					@7			@4				@5			
%QW5																
%IW6	@10					@11			@8				@9			
%QW6																
%IW7	@14					@15			@12				@13			
%QW7																
%IW8	.....					.....			.....				@16			
%QW8																

Dans le plan-mémoire précédent, considérons qu'un actionneur de profil B1 soit relié au bus, et affecté à l'adresse 4.

L'actionneur 1 de cet esclave pourra être accédé par le programme de l'API à l'adresse CEI suivante :

……

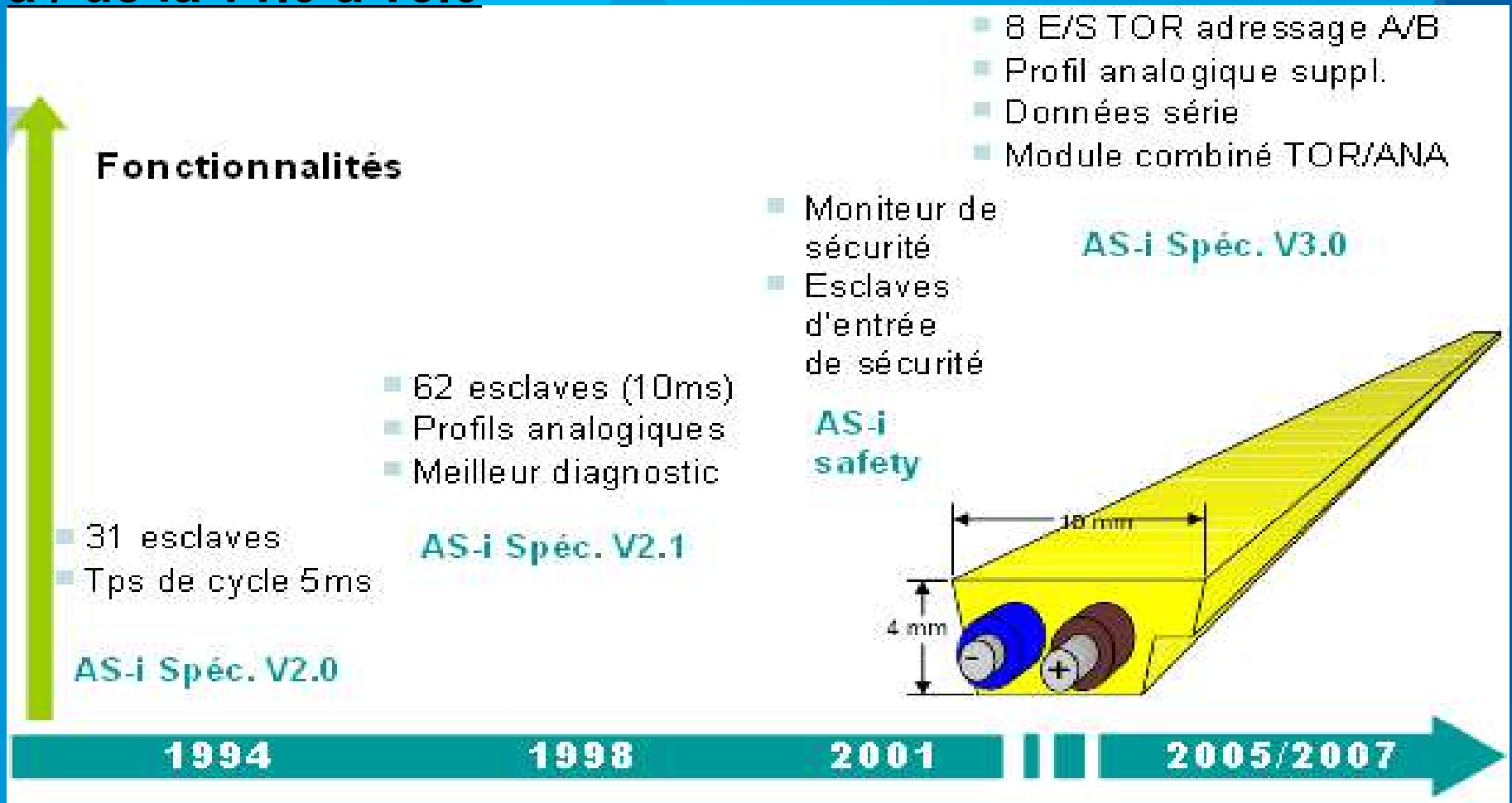
Le capteur 1 de cet esclave pourra être accédé par le programme de l'API à l'adresse CEI suivante :

……

Donnez l'adresse CEI d'un capteur inductif de profil 1.1 relié au bus ASi du coupleur précédent, et dont l'adresse ASi est 8.

5- Évolutions du bus ASi

a / de la v1.0 à v3.0



- AS-i rime avec simplicité !
- Compatibilité ascendante

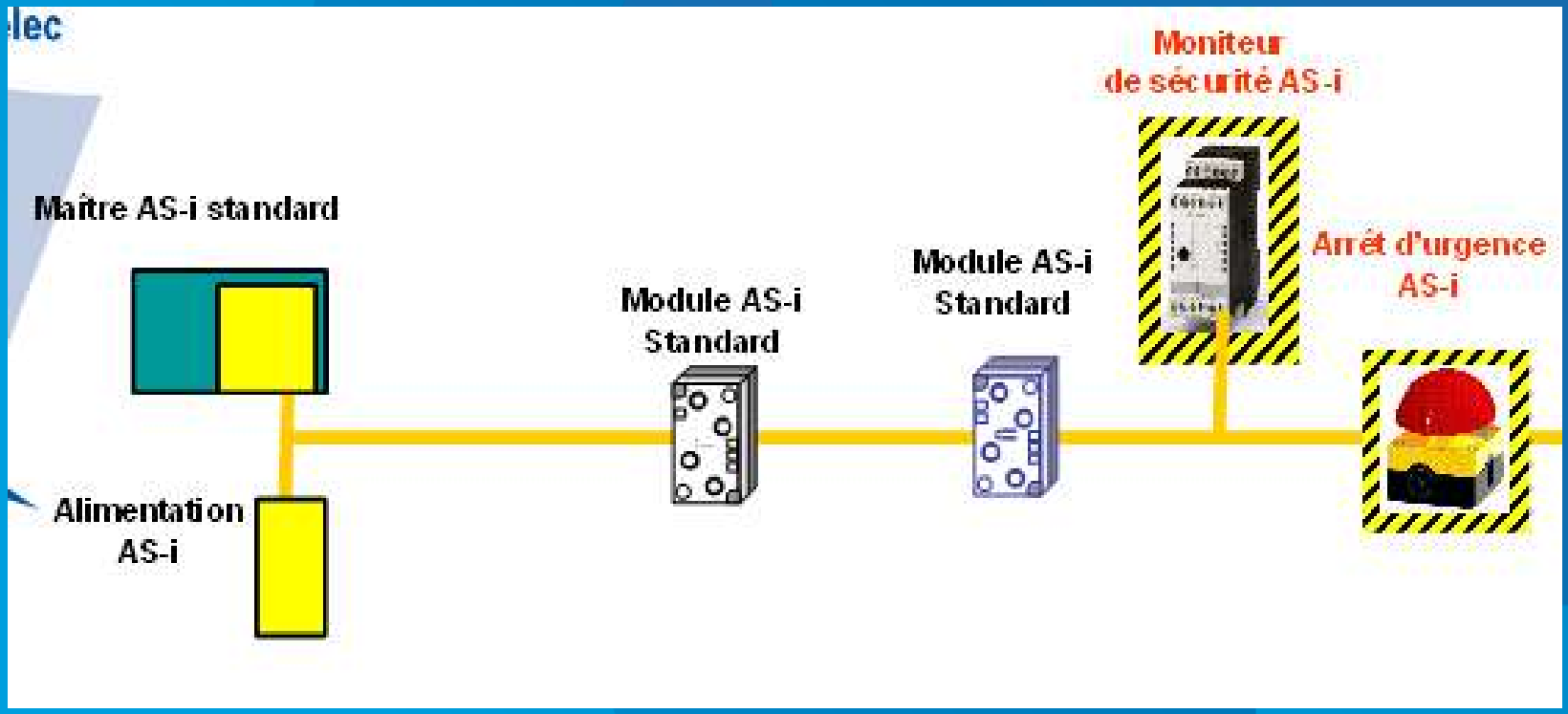
b / ASi-safe :

Les composants liés à la ***sécurité des personnes*** au sein des systèmes automatisés sont régis par des ***normes spécifiques***.***(EN 954-1 , IEC61508)***

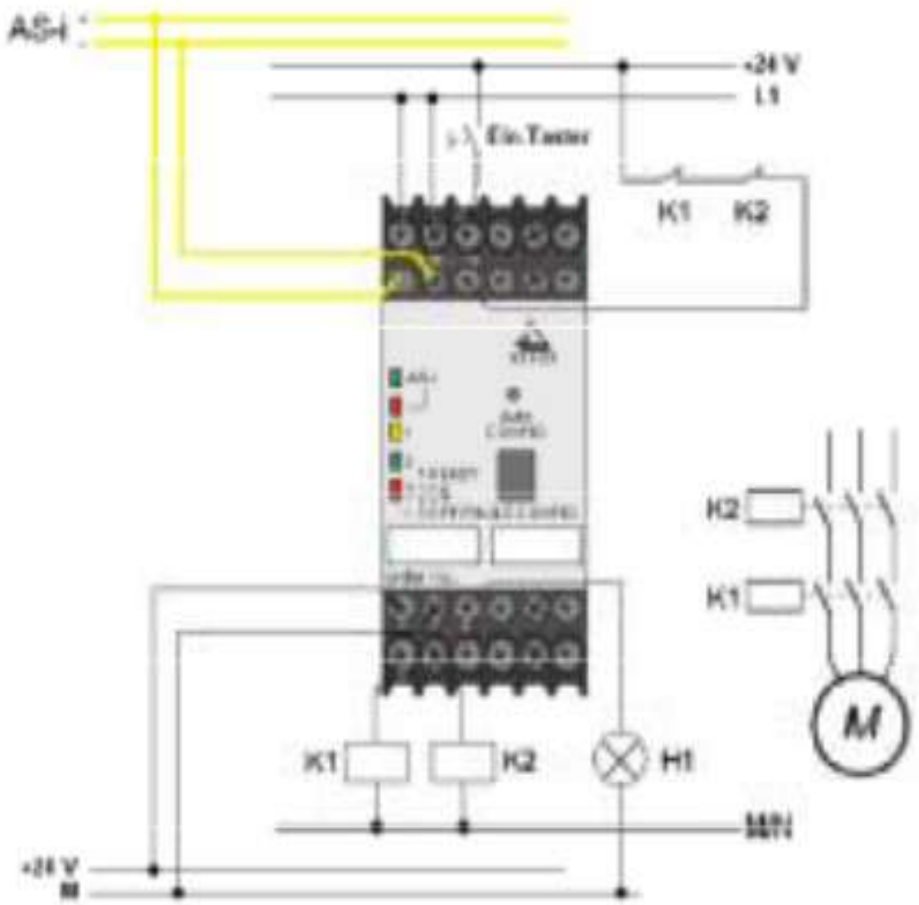
Lorsque la commande de ces composant (arrêt d'urgence, barrières immatérielles, relai de coupure etc...) est gérée électriquement, elle doit être ***commandée par un moniteur de sécurité dédié***, et non par le programme de l'API.

Depuis 2001, le moniteur de sécurité et les composants de sécurité peuvent être ***connectés au bus Asi*** afin d'utiliser le bus comme support de communication.

Les composants de sécurité Asi-safe et les composants Asi standards ***cohabitent sur le même bus*** :



Moniteur de sécurité :



1.13/1.14	
1.23/1.24	Enable ciruits
1.32	Signal circuit
1.Y1	On button optional
1.Y2	Feedback circuit

Configuration logicielle du moniteur

- Plusieurs sorties relais sécurité
- Paramétrage : Stop catégorie 0 et 1, Inhibition redémarrage, Acquittement local, ...

✔ *Principe des échanges avec le moniteur de sécurité :*

- Chaque esclave dispose d'une *table spécifique 8 x 4 bits*.
- Une fonction d'apprentissage du moniteur *mémorise les tables* de tous les esclaves Asi-SAFE
- En fonctionnement, le moniteur observe les réponses émises par les esclaves Asi-safe, et les *compare à sa table mémorisée*
- Le moniteur *déclenche les relais de sécurité* lorsque la réponse d'un esclave Asi-safe est différente de la ligne de la table attendue, c'est à dire :
-

  - Si STOP demandé (0000)
  - Communication interrompue
  - Esclave en défaut
  - Autre code différent du code mémorisé

Surveillance des esclaves de sécurité par le moniteur :

Le moniteur de sécurité doit être programmé, afin de *disposer en mémoire des différentes tables* des esclaves de sécurité connectés au bus.

A chaque cycle de polling, les moniteur et les esclaves de sécurité seront interrogés au même titre que les autres esclaves raccordés au bus.

Les réponse des esclaves de sécurité seront analysées par le moniteur de sécurité (et parviendront également au maître ASi qui pourra les traiter).

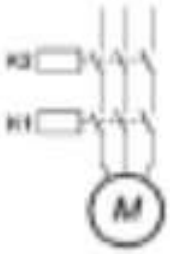
Si une réponse diffère de la séquence attendue :

- Le moniteur *attend le prochain cycle*. Si la réponse diffère de nouveau de la table, les *sorties* pilotées par le moniteur *sont coupées* (ou mises en position de *repli*).

- Si la réponse est « normale » au cycle suivant, c'est qu'il s'agissait d'une erreur de transmission et le message précédent n'est pas pris en compte. Cela permet d'éviter les *coupures intempestives*.

Temps de réponse ASi-safe :

As-I assure une réaction réflexe de sécurité en moins de 35 ms (ouverture d'un relais pour coupure de puissance)



35 ms

Signal sécurité Généré par esclave Safe

5 ms | 5 ms | 5 ms | 5 ms | 15 ms

Commande de sécurité Générée sur le moniteur Safe

Incertitude Max. pour démarrage cycle ASi

Temps cycle ASi

Répétition du cycle (code invalide permis une fois seulement)

Temps traitement Moniteur Safe

Temps commutation Relais moniteur Safe

Ch 3 : Modbus (RTU & TCP)

1 - Historique :

Modbus est la contraction de « ***Modicon*** » et de « ***Bus*** ». Il s'agit du **premier bus de terrain normalisé**, et a été conçu dans les années 80 par le fabriquant d'automates Modicon (aujourd'hui Schneider-Télémécanique).

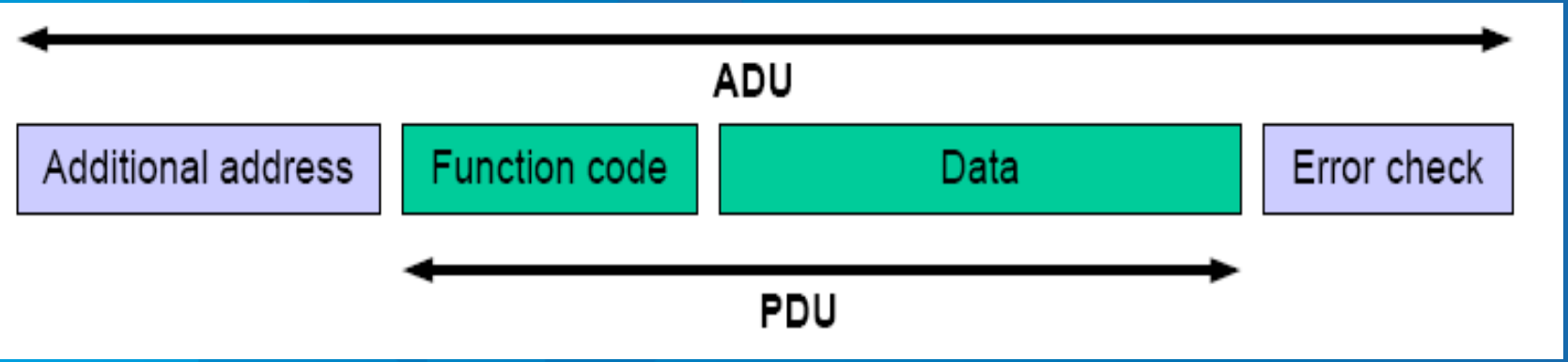
→ Historiquement, Modbus a été développé pour le milieu industriel sur des ***lignes séries*** (RS232, RS485),

→ puis avec l'essor des réseaux informatiques sur ***Ethernet***.

Modbus représentent actuellement **un des principaux standards** de communication dans le domaine de l'automatisation industrielle, il est supporté par l'association ***Modbus-IDA*** chargée de définir les spécifications nécessaires au développement de composants « Modbus compliant ».

2 – Description des échanges Modbus:

Modbus définit une trame de base baptisée « **PDU** » (« Protocol Data Unit ») indépendante du type de protocole considéré (série ou TCP).



Elle comporte 2 champs :

- **Code de la Fonction Modbus :** Spécifie le type d'opération initié par la communication (lecture de mots, écriture de bits etc...)
- **Données :** Données nécessaires à l'éxécution de la fonction ou renvoyées par celle-ci.

La trame Modbus complète, « **ADU** » (Application Data Unit ») comportera en plus les informations d'adressage et de détection d'erreur propres au média de communication envisagé (série ou TCP).

✔ les types de données échangées :

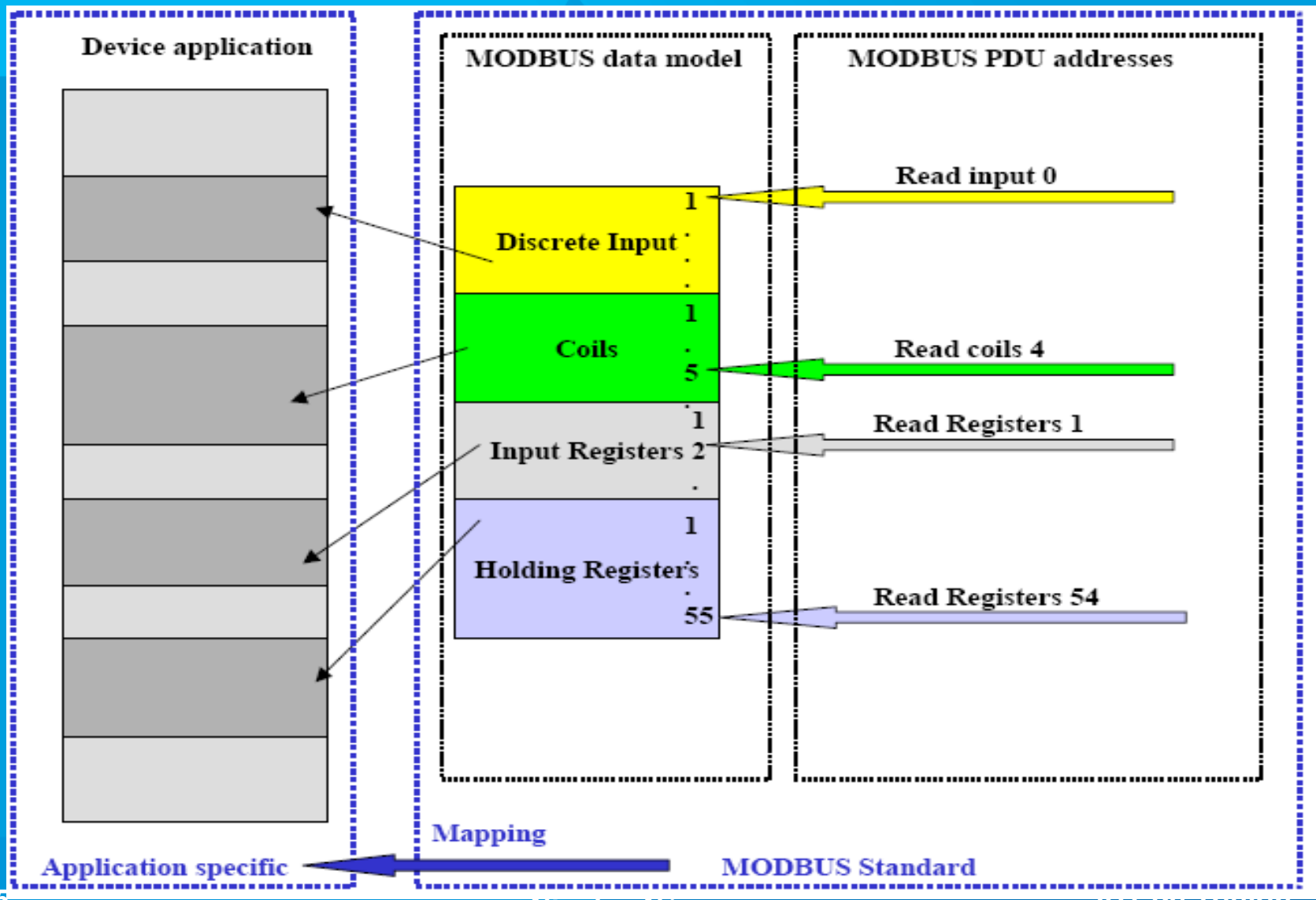
Modbus manipule 4 types de données :

	Type d'objet	*Accès*	*Exemple:*
Bits d'entrée ("Discrets Inputs")	Bits	Lecture	Interface d'entrées TOR
Bits de sortie ('Coils")	Bits	Lecture/Ecriture	Interface de sorties TOR, bits internes
Mots d'entrées ("Input Registers")	Mots	Lecture	Interface d'entrée analogique
Mots de sortie ("Holding Registers")	Mots	Lecture/Ecriture	Registres internes, sorties analogiques

Les adresses des objets Modbus sont codées sur 16 bits, ce qui autorise 65536 objets de chaque type possible par équipement accessibles via Modbus.

C'est à chaque fabriquant de matériel de définir les plages d'adresses pour chacun de ces segments de données, ceux-ci pouvant tout aussi bien être disjoints que confondus :

Les adresses Modbus PDU commencent à 0, chaque fabriquant spécifie via une table de correspondances les adresses correspondantes des objets accessibles de son équipement :



Les fonctions Modbus sont codées sur un octet :

- Les valeurs ***1 à 127*** correspondent à des ***fonctions Modbus***;

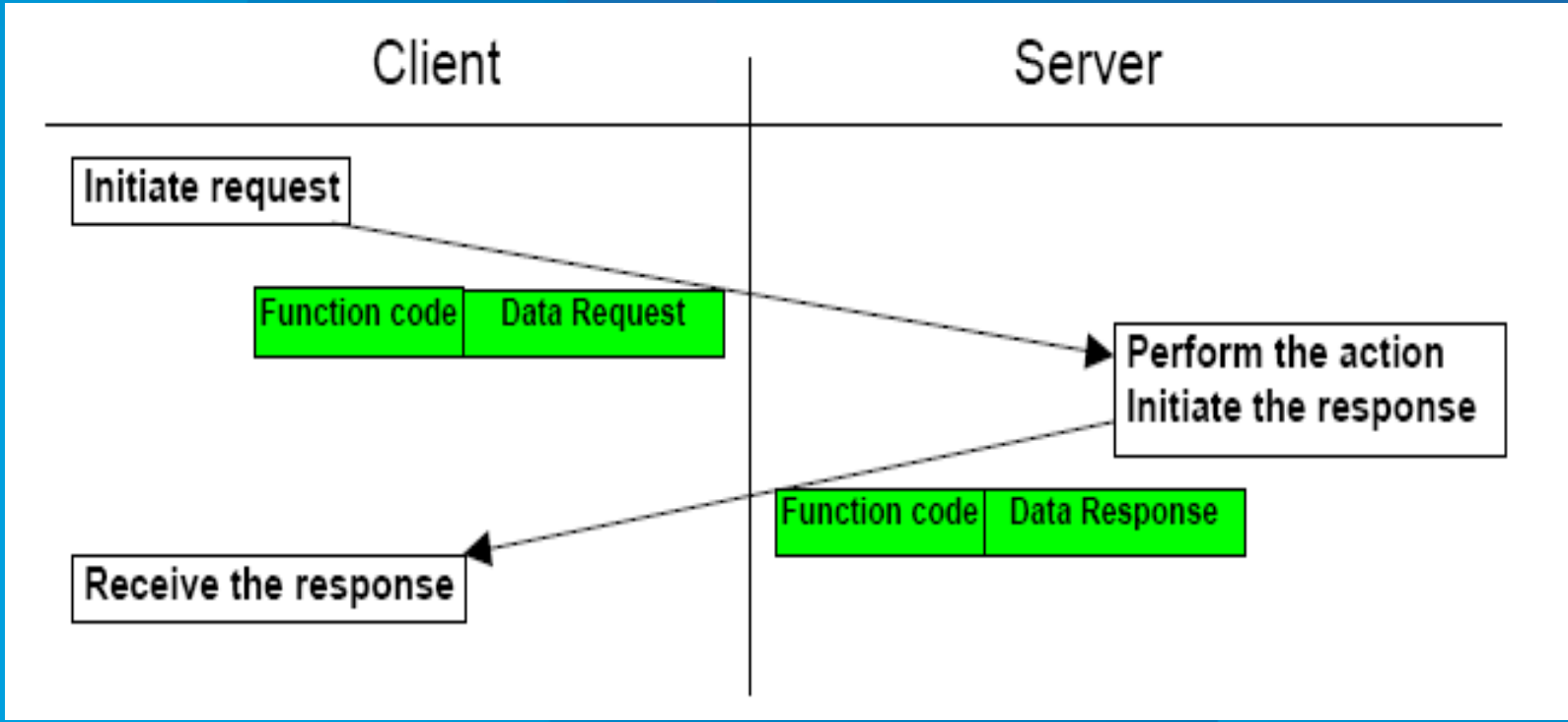
- Les valeurs ***128 à 255*** correspondent aux ***codes d'exceptions*** indiquant qu'une erreur s'est produite au cours d'un échange;

- Le code 0 n'est pas valide.

La longueur totale de la trame PDU ne peut excéder 253 octets (pour des raisons de compatibilité des échanges entre les modes série et TCP).

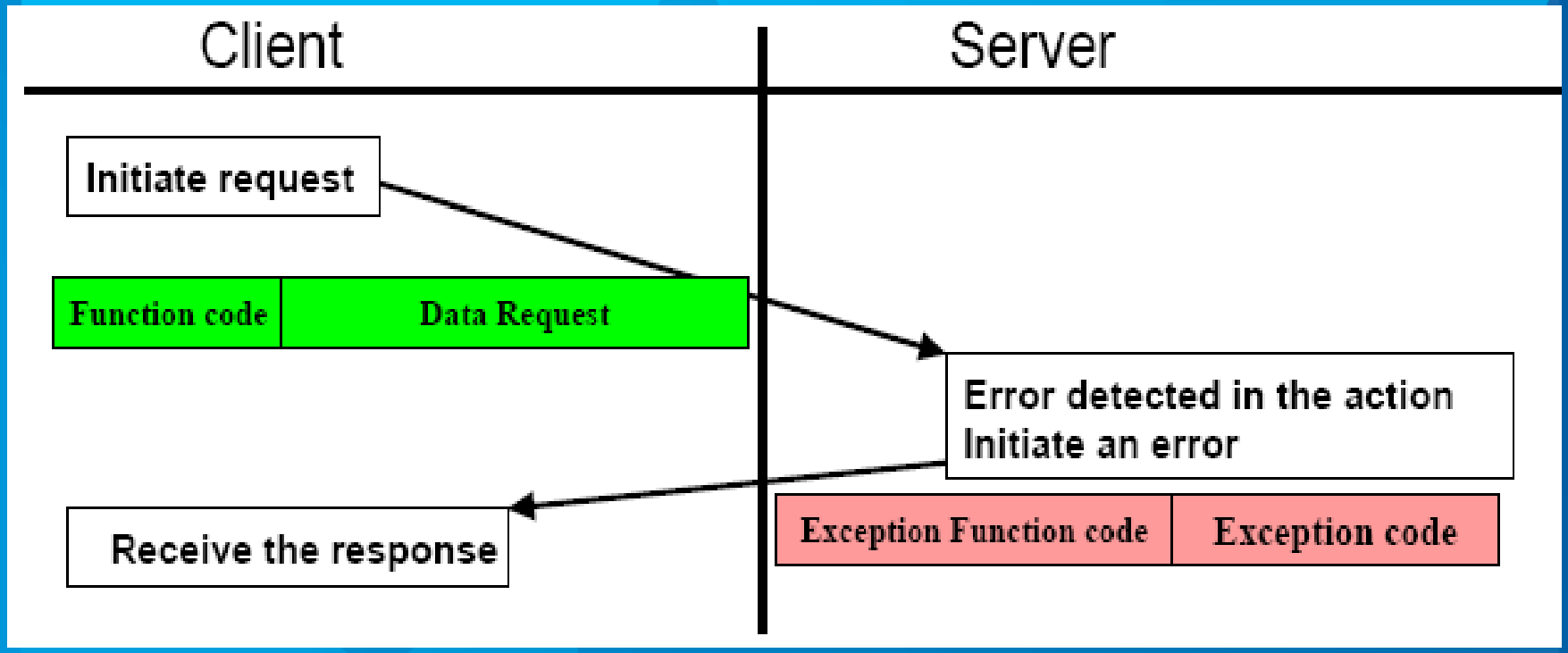
La **longueur champ de données** peut donc être comprise entre ***0*** et ***252*** octets.

Les échanges Modbus sont basés sur un modèle ***Client / Serveur*** (maître/esclave dans le cas série).
Le Client (maître) est toujours à l'initiative d'un échange. Il envoie une ***requête*** au serveur qui lui retourne, après analyse de cette requête une ***réponse*** :



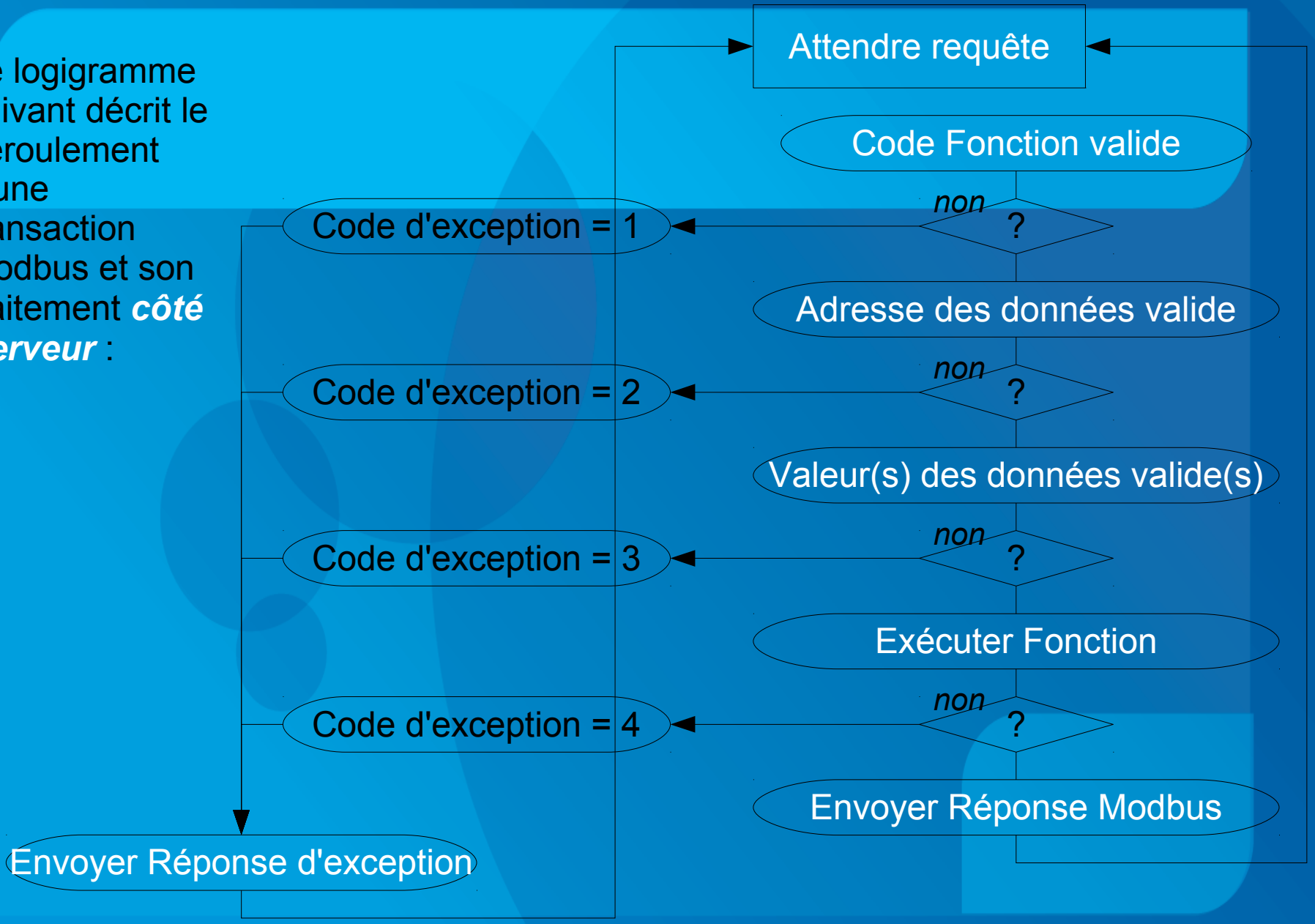
La trame de réponse contient les éventuelles données demandées par le client dans le champ de données et le code de la fonction exécutée en ***echo*** (**Accusé réception :** Signifie que la fonction a été correctement exécutée).

Si une erreur est détectée à la réception grâce au champ de contrôle (error check), ou si une erreur survient lors de l'éxécution de la requête par le serveur, la réponse renvoyée par celui-ci est alors dite « d'***exception*** » :



Le champ « fonction » de la réponse contient alors la valeur ***[128 + code fonction]*** indiquant au client que la fonction n'a pu être éxécutée; le champ « données » contient quant à lui un ***code d'exception*** indiquant le type d'erreur qui s'est produite.

Le logigramme suivant décrit le déroulement d'une transaction Modbus et son traitement ***côté Serveur*** :



✔ Principales fonctions offertes par Modbus :

Code	Nature des fonctions MODBUS	Code	Nature des fonctions MODBUS
H'01'	Lecture de n bits de sortie	H'0B'	Lecture du compteur d'événements
H'02'	Lecture de n bits d'entrée	H'0C'	Lecture des événements de connexion
H'03'	Lecture de n mots de sortie	H'0D'	Téléchargement, télé déchargement et MM
H'04'	Lecture de n mots d'entrée	H'0E'	Demande de CR de fonctionnement
H'05'	Ecriture de 1 bit de sortie	H'0F'	Ecriture de n bits de sortie
H'06'	Ecriture de 1 mot de sortie	H'10'	Ecriture de n mots de sortie
H'07'	Lecture du statut d'exception		
H'08'	Accès aux compteurs de diagnostic	H'11'	Lecture d'identification
H'09	Téléchargement, télé déchargement et mode de marche	H'12'	Téléchargement, télé déchargement et MM
H'0A'	Demande de CR de fonctionnement	H'13'	Reset de l'esclave après erreur

✔ Exemple de système industriel automatisé mettant en œuvre des échanges ModBus :

Le système suivant concerne la partie commande (P.C) d'un procédé industriel destiné à la fabrication de colle.

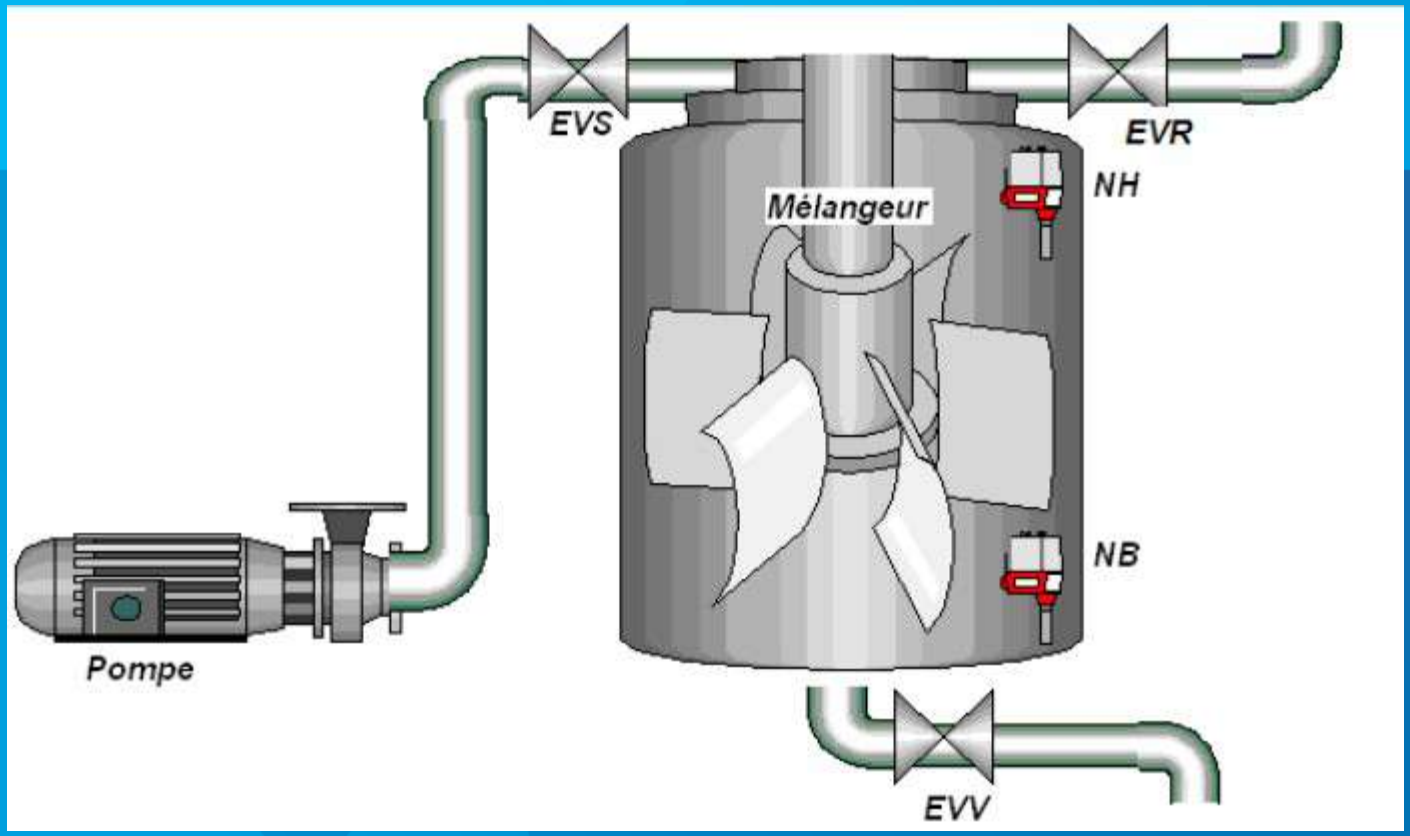
Ce procédé nécessite le mélange d'un réactif en poudre avec un solvant. Le solvant est puisé au moyen d'une pompe P1 vers la cuve de mélange. P1 est actionné via un moteur asynchrone piloté par un variateur (1) de vitesse.

Le mélange est effectué par une pâle, également mue par un ensemble “moteur asynchrone + variateur” (2).

Un API gère le pilotage du procédé (acquisition des capteurs : niveaux, températures, commande des vannes, commande des variateurs).

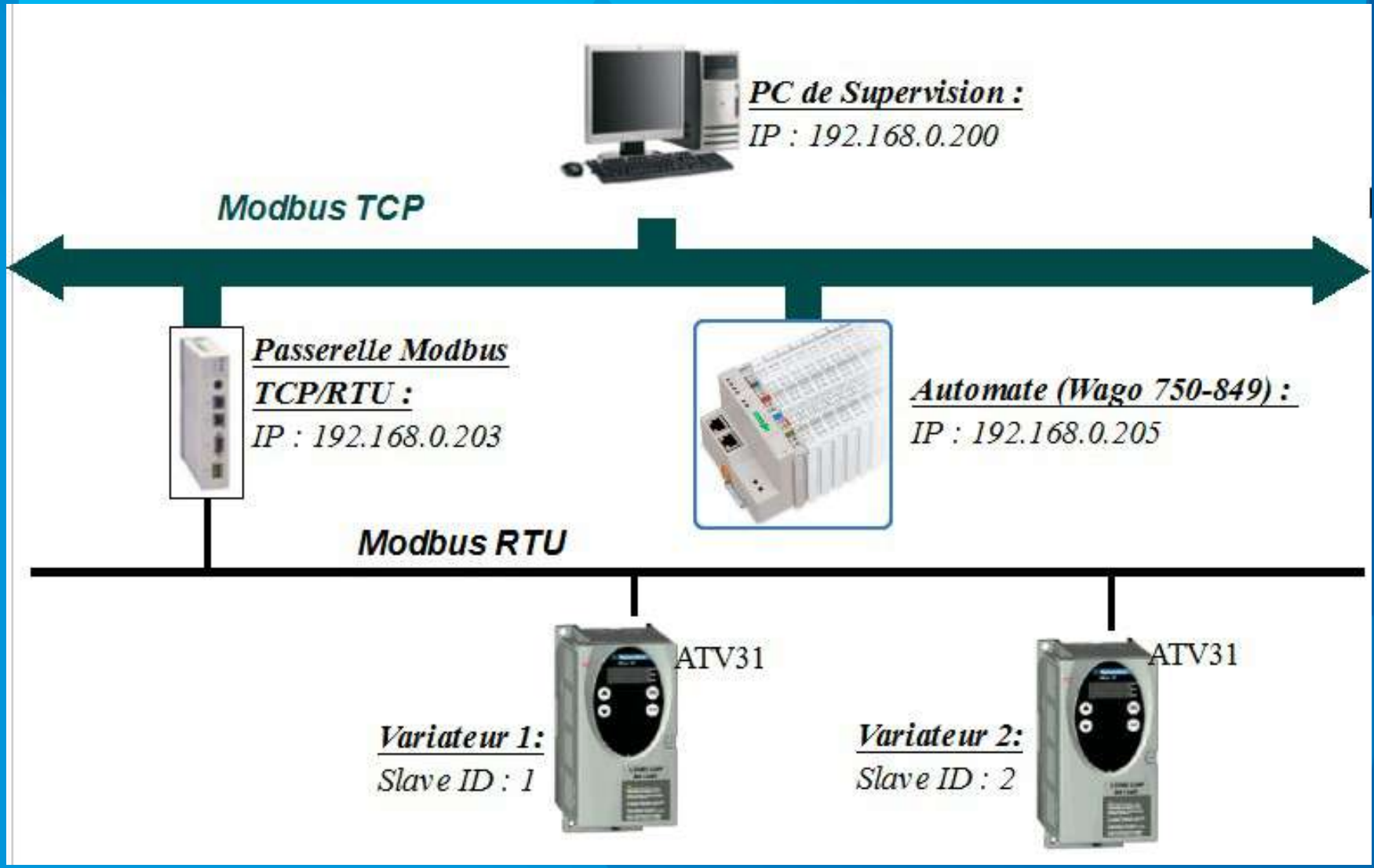
Un PC assure la supervision du système (visualisation du cycle de production, alarmes/défauts, choix des modes de marche ...)

✗ Synoptique de la partie opérative du système :



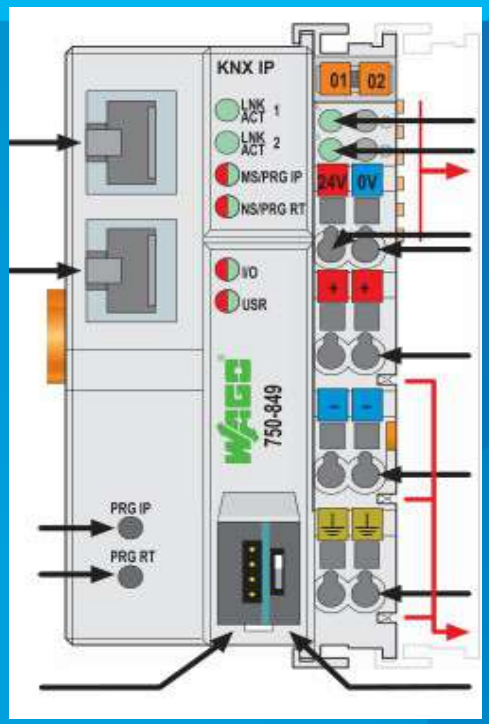
EVS	Electrovanne “Solvant”
EVR	Electrovanne “Réactif”
EVV	Electrovanne “Vidange”
NH, NB	Cpateurs “Niveau Haut” et “Niveau Bas”
fcso, fcro, fcvo	fins de course des elctrovannes

✗ Synoptique de la partie commande du système :



✗ Équipements mis en jeu : API Wago 750-849

Ports Ethernet pour les échanges Modbus TCP (entre autres)



+ carte 8 E/TOR pour les capteurs de niveau et fin de course des electrovannes

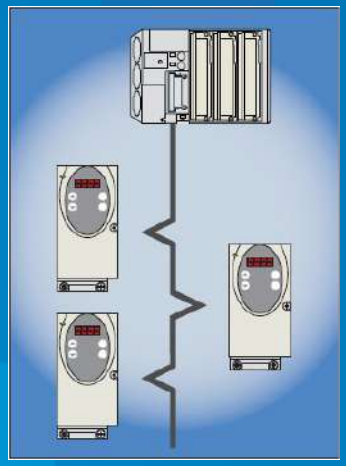
+ carte 2 E Analogiques pour mesures de températures dans l'enceinte

+ carte 4 S TOR pour pilotage des électrovannes (monostables).

Dans notre application, l'API est chargé de la commande du process. Toutes ses données d'E/S sont accessibles via Ethernet au superviseur en Modbus TCP.
Il transmet également ses ordres aux variateurs via le réseau, en Modbus TCP.

✗ Équipements mis en jeu : variateur communicant ATV31

Variateur de Vitesse Altivar 31 :



Plusieurs centaines de variables accessibles via Modbus :

- En lecture : Courant dans les phases (mot), état thermique du moteur (mot), fins de course (bits)...

- En lecture/écriture : réglages PID (mots), consigne de vitesse (mot), M/A (bits), rampes d'accélération et de décélération (mots) etc...

Connecteur RJ45 pour réseau Modbus

Dans notre application, deux ATV31 sont utilisés pour piloter les variateurs. Ils pourront, grâce à leur communication Modbus transmettre leur état (courants, défauts thermiques etc...) à la supervision, et recevoir les ordres de commande des API.

✗ Équipements mis en jeu : passerelle TCP/RTU

Elle permet de convertir les couches physiques Ethernet / RS485, et d'encapsuler / Désencapsuler les trames entre les protocoles Modbus RTU et Modbus TCP.

✔ Fonction 0x01 : Lecture de n bits de sorties.

Cette fonction permet de lire 1 à 2000 bits consécutifs dans le segments mémoire du serveur correspondants aux sorties discrètes (« coils »).

La ***trame de requête*** contient les champs suivants :

Code Fonction	1 Octet	0x01
Adresse de départ	2 Ocets	De 0x0000 à 0xFFFF
Nombre de bits à lire :	2 Octets	De 1 à 2000 (0x7D0)

La ***trame de réponse***, si aucune erreur ne survient :

Champ :	***Taille :***	***Valeur :***
Code Fonction :	1 octet	0x01
Nombre d'octets de la trame de réponse :	1 octets	N*
Etat des bits lus :	N octets	Etat des bits lus

** Rq : N = (nb_de_bits) / 8 si nb_de_bits est un multiple de 8*
N = (nb_de_bits) / 8 + 1 dans le cas contraire

Si une ***erreur*** survient au cours de l'échange, la trame de réponse devient :

Champ :	Taille :	Valeur :
Code d'erreur :	1 octet	0x81
Code d'exception :	1 octet	Code de l'erreur

✔ Fonction 0x02 : Lecture de n bits d'entrées.

Cette fonction permet de lire 1 à 2000 bits consécutifs dans le segments mémoire du serveur correspondants aux entrées discrètes (« discrete inputs »).

La ***trame de requête*** contient les champs suivants :

Champ :	Taille :	Valeur :
Code Fonction :	1 octet	0x02
Adresse du 1er bit à lire :	2 octets	0x0000 à 0xFFFF
Nombre de bits à lire :	2 octets	1 à 2000 (0x7D0)

La ***trame de réponse***, si aucune erreur ne survient :

Champ :	Taille :	Valeur :
Code Fonction :	1 octet	0x02
Nombre d'octets de la trame de réponse :	1 octet	N
Etat des bits lus :	N x 1 octet	Valeurs lues

Si une ***erreur*** survient au cours de l'échange, la trame de réponse devient :

Champ :	Taille :	Valeur :
Code d'erreur :	1 octet	0x82
Code d'exception :	1 octet	Code de l'erreur

➢ *Exemple :*

Le superviseur émet une requête permettant de lire l'état des capteurs de niveau (entrées TOR 1 & 2 de l'API). La cuve est à moitié pleine.

Requête :

Nom du champ :	Valeur :

Réponse :

Nom du champ :	Valeur :

✔ Fonction 0x03 : Lecture de n registres.

Cette fonction permet de lire 1 à 125 mots (16 bits) consécutifs dans le segments mémoire du serveur correspondants aux registres accessibles en lecture/écriture (« Holding register »).

La ***trame de requête*** contient les champs suivants :

Champ :	Taille :	Valeur :
Code Fonction :	1 octet	0x03
Adresse du 1° registre à lire :	2 octets	0 à 0xFFFF
Nombre de registres à lire (n) :	2 octets	1 à 125 (0x7D)

La ***trame de réponse***, si aucune erreur ne survient :

Champ :	Taille :	Valeur :
Code Fonction :	1 octet	0x03
Nombre d'octets du champ suivant :	1 octet	2 x n
Mots lus :	n x 2 octets	Valeurs lues

Si une ***erreur*** survient au cours de l'échange, la trame de réponse devient :

Champ :	Taille :	Valeur :
Code d'erreur :	1 octet	0x83
Code d'exception :	1 octet	Code de l'erreur

➢ *Exemple :*

Le superviseur émet une requête permettant de lire l'état de la température dans la cuve (stockée en 1/100° de degrés dans le mot interne n°1000 de l'API). Celle-ci est de 42,3°C.

Requête :

Nom du champ :	Valeur :

Réponse :

Nom du champ :	Valeur :

✔ Fonction 0x04 : Lecture de n registres d'entrée.

Cette fonction permet de lire 1 à 125 mots (16 bits) consécutifs dans le segments mémoire du serveur correspondants aux registres accessibles en lecture seule (« Input register »).

La ***trame de requête*** contient les champs suivants :

Champ :	Taille :	Valeur :
Code Fonction :	1 octet	0x04
Adresse de départ :	2 octets	0x0000 à 0xFFFF
Nombre de registres à lire (n) :	2 octets	1 à 125 (0x7D)

La ***trame de réponse***, si aucune erreur ne survient :

Champ :	Taille :	Valeur :
Code Fonction :	1 octet	0x04
Nombre d'octets de la trame de réponse :	1 octet	2 x n
Valeurs de registres lus :	nx2 octets	valeurs

✔ Fonction 0x05 : Ecriture d'un bit de sortie.

Cette fonction permet de forcer une sortie à '1' ou à '0' sur l'équipement distant. Le sous-champ « valeur » du champ « données » contient 2 octets : la valeur 0xFF00 permet le forçage à '1', la valeur 0x0000 le forçage à '0'; toutes les autres valeurs sont interdites. La réponse à cette requête est un écho de la requête.

La ***trame de requête*** contient les champs suivants :

Champ :	***Taille :***	***Valeur :***
Code Fonction :	1 octet	0x05
Adresse du bit à écrire :	2 octets	0x0000 à 0xFFFF
Valeur à écrire ('0' ou '1') :	2 octets	***0x0000 ou 0xFF00***

Si aucune erreur ne survient, la trame de réponse est identique à la trame de requête (écho)

La ***trame de réponse***, si une erreur survient :

Champ :	***Taille :***	***Valeur :***
Code d'erreur :	1 octet	0x85
Code d'exception :	1 octet	01, 02, 03, ou 04

✔ Fonction 0x06 : Ecriture d'un registre de sortie.

Cette fonction permet l'écriture d'une variable sur un mot accessible en lecture/écriture de l'équipement distant. La réponse à cette requête est un écho de la requête.

La ***trame de requête*** contient les champs suivants :

Champ :	Taille :	Valeur :
Code Fonction :	1 octet	0x06
Adresse du registre à écrire :	2 octets	0x0000 à 0xFFFF
Valeur à écrire (big endian) :	2 octets	*0x0000 à 0xFFFF*

Si aucune erreur ne survient, la trame de réponse est identique à la trame de requête (écho)

La ***trame de réponse***, si une erreur survient :

Champ :	Taille :	Valeur :
Code d'erreur :	1 octet	0x86
Code d'exception :	1 octet	01, 02, 03, ou 04

✔ Fonction 0x0F : Ecriture de n bits de sortie.

Cette fonction permet d'écrire 1 à 1968 bits consécutifs d'un équipement distant. La réponse à cette requête renvoie le nombre de bits écrits ainsi que l'adresse de départ en écho.

La ***trame de requête*** contient les champs suivants :

Champ :	Taille :	Valeur :
Code Fonction :	1 octet	0x0F
Adresse de départ :	2 octets	0x0000 à 0xFFFF
Nombre de bits à écrire :	2 octets	*0x0001 à 0x07B0*
Nombre d'octets du champ suivant	1 octet	0 à 255
Valeurs à écrire	N octets	.....

Adresse de départ	b_{15} b_{14} b_{13} ..... b_3 b_2 b_1 b_0
Adresse de départ + 1	b_{31} b_{30} b_{29} ..... b_{18} b_{17} b_{16}
Adresse de départ + 2	*etc...*

✔ Fonction 0x10 : Ecriture de n registres.

Cette fonction permet d'écrire 1 à 123 registres consécutifs d'un équipement distant. La réponse à cette requête renvoie le nombre de registres écrits ainsi que l'adresse de départ.

La ***trame de requête*** contient les champs suivants :

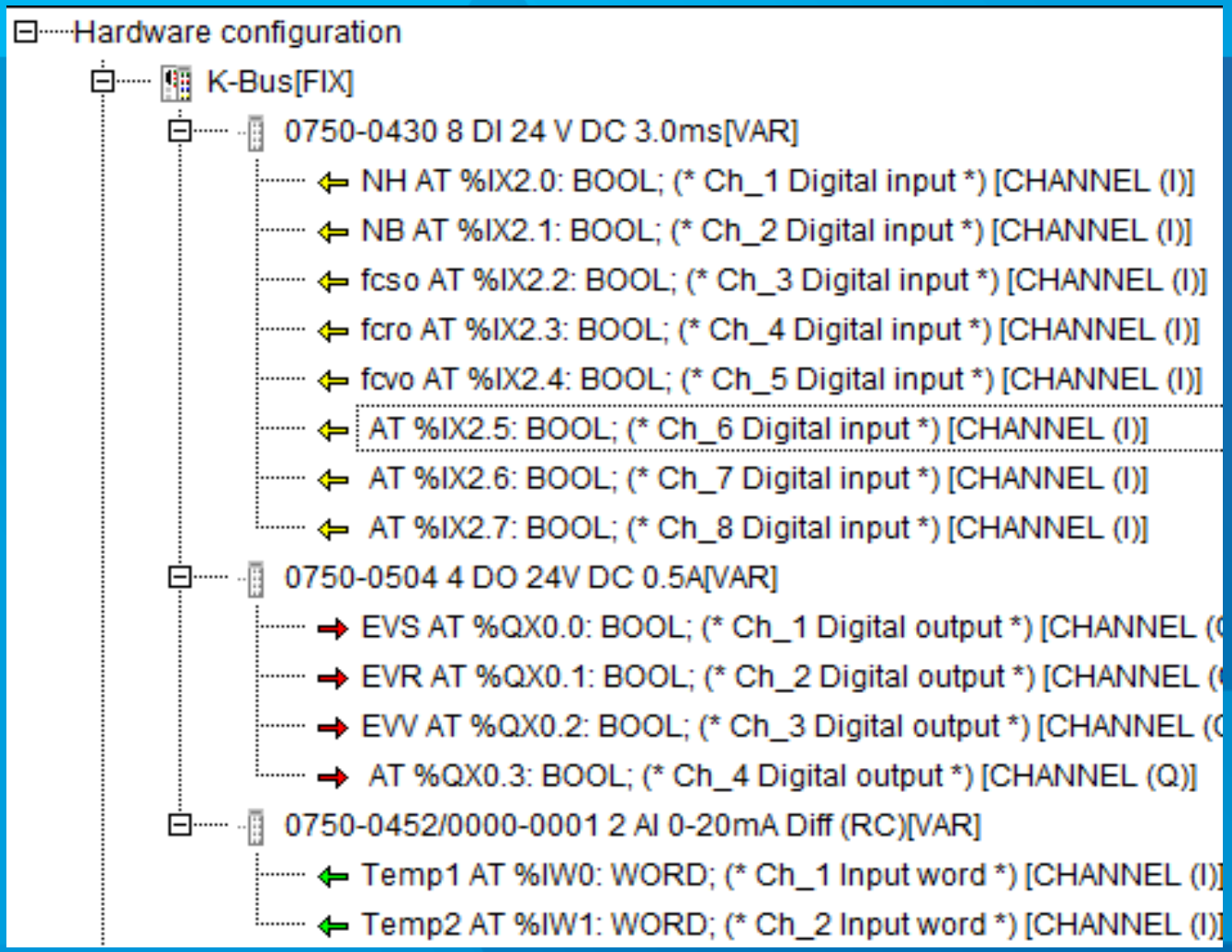
Champ :	***Taille :***	***Valeur :***
Code Fonction :	1 octet	0x10
Adresse du 1° mot à écrire :	2 octets	0x0000 à 0xFFFF
Nombre de mots à écrire (n) :	2 octets	0 à 0x7B
Nombre d'octets du champ suivant	1 octets	0 à 255
Valeurs à écrire (big endian) :	n octets	***0x0000 à 0xFFFF***

La ***trame de réponse (sans erreur)*** :

Champ :	***Taille :***	***Valeur :***
Code Fonction :	1 octet	0x10
Adresse du 1° mot écrit :	2 octet	0 à 0xFFFF
Nombre de mots écrits	2 octets	n

✔ Exercice.

Voici la déclaration des E/S du système présenté précédemment :



Donnez les trames Modbus (PDU) permettant à un équipement distant :

- La lecture de toutes les entrées TOR utilisées :
- D'ouvrir la vanne de vidange
- De connaître l'état des températures

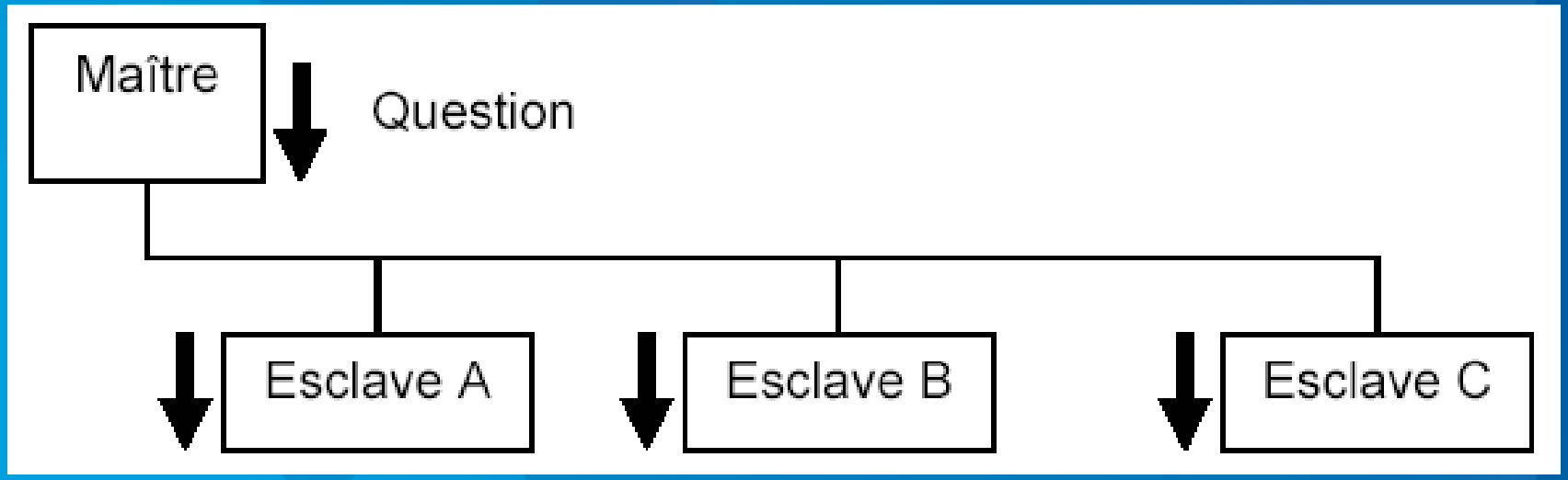
3 - Modbus Série :

Les premières implantations du protocole Modbus sont apparues sur des lignes séries.

Layer	ISO/OSI Model	
7	Application	MODBUS Application Protocol
6	Presentation	Empty
5	Session	Empty
4	Transport	Empty
3	Network	Empty
2	Data Link	MODBUS Serial Line Protocol
1	Physical	EIA/TIA-485 (or EIA/TIA-232)

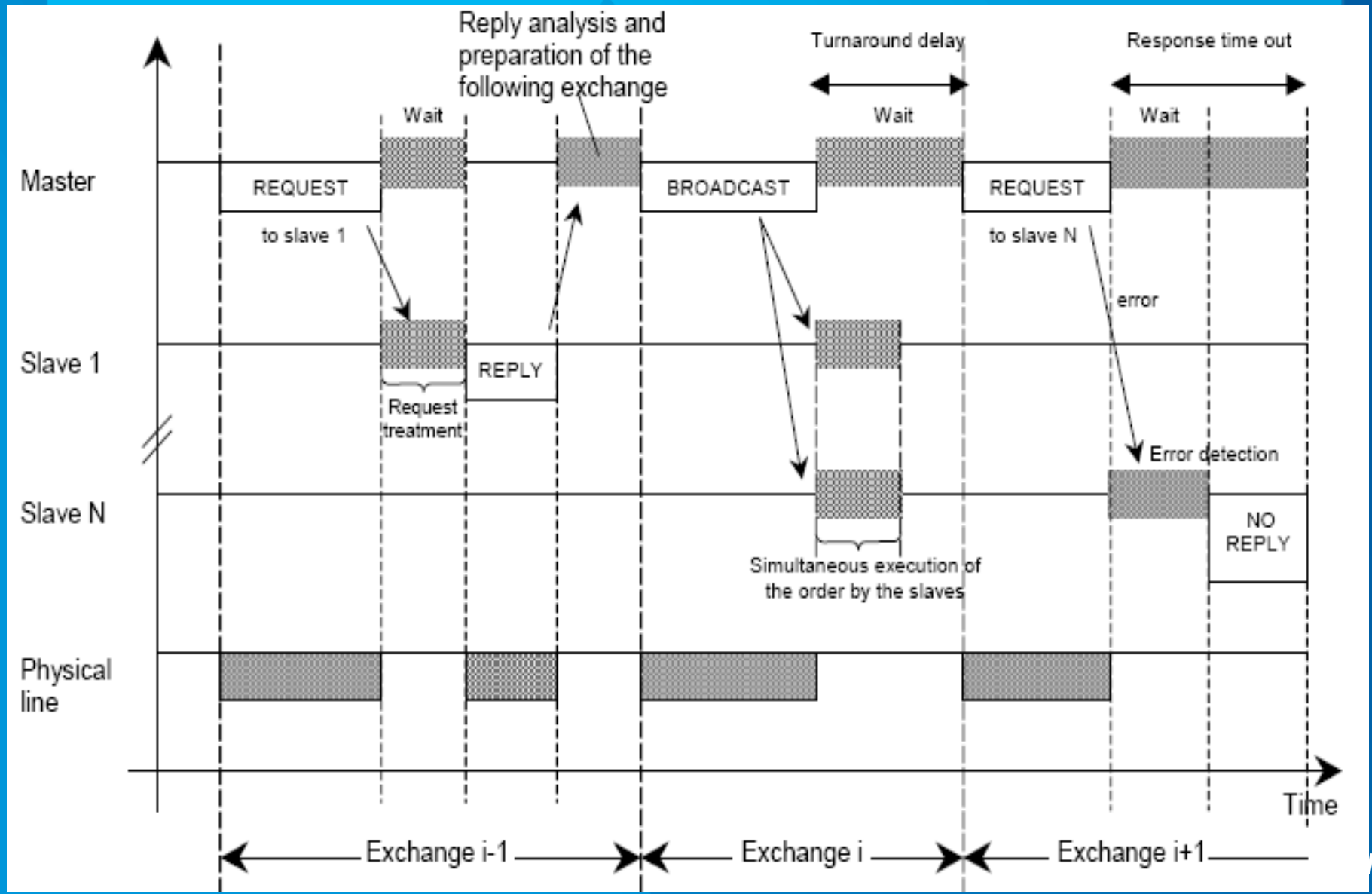
Les services offerts par ces protocoles sont ceux définies par la couche application commune aux protocoles Modbus, et quelques services supplémentaires viennent se greffer.

Tout d'abord, un mode appelé « ***diffusion*** » (« Broadcast ») permet au ***maître (client)*** de s'adresser à l'ensemble des ***esclaves (serveurs)*** présents sur le bus en envoyant une requête à l'***adresse 0***.



D'autre part, un certain nombre de *fonctions de diagnostique* ne sont implantées que sur Modbus série.

Timing des communications Modbus-Série :

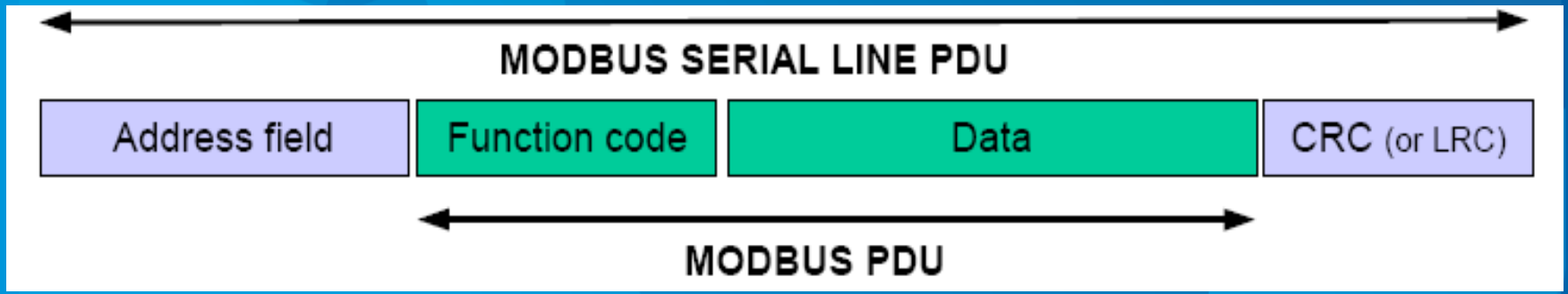


En plus des champs définis par la couche applicative de Modbus (code fonction et données), Les trames Modbus série incluent :

✔ **l'adresse de l'esclave en en-tête** ***(1 octet)*** **:**

- Adr = 0 pour la diffusion,
- 1<Adr<255 pour un échange entre maître et esclave)

✔ **Champ de contrôle de la validité de l'échange :** ***(1 ou 2 octets)***

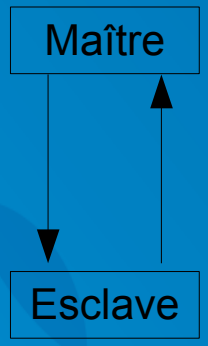


✔ Couche Physique :

La couche physique de Modbus série obéit à l'une des 2 normes RS232 ou RS485 :

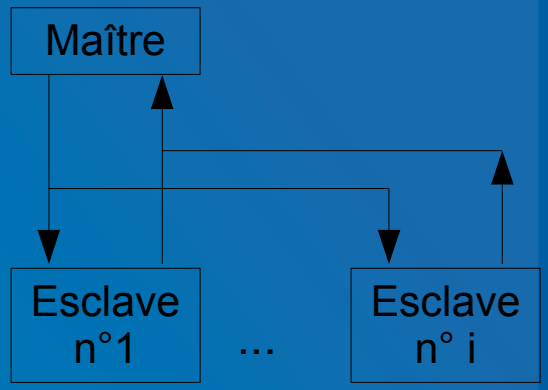
RS232 :

Echanges entre 1 Maître & 1 esclave : POINT à POINT & Courtes Distances (<20m)



RS485 :

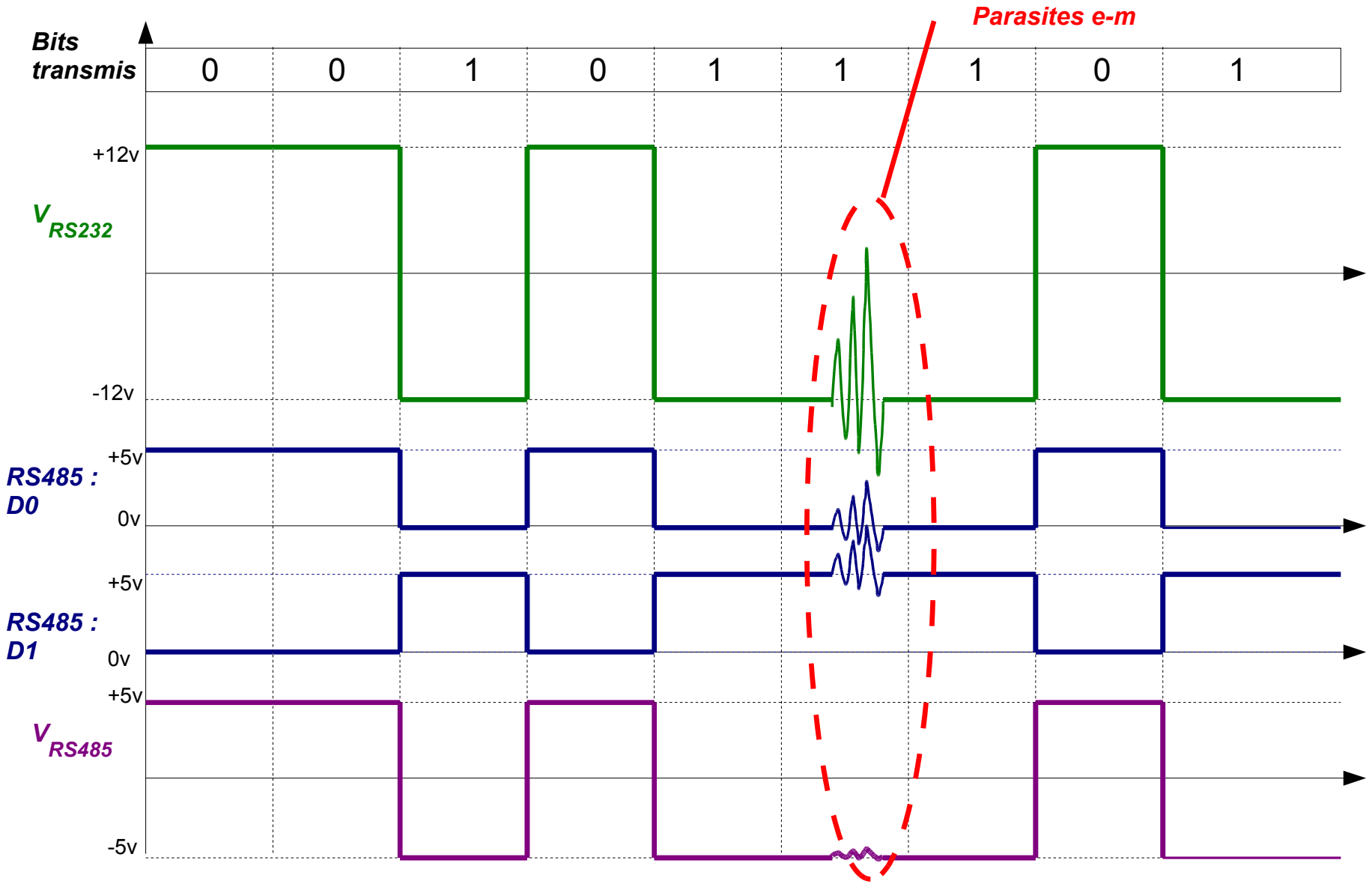
Echanges entre 1 Maître & plsrs esclave : MULTIPOINT Jusqu'à 120m de distance



✗ Rappels :

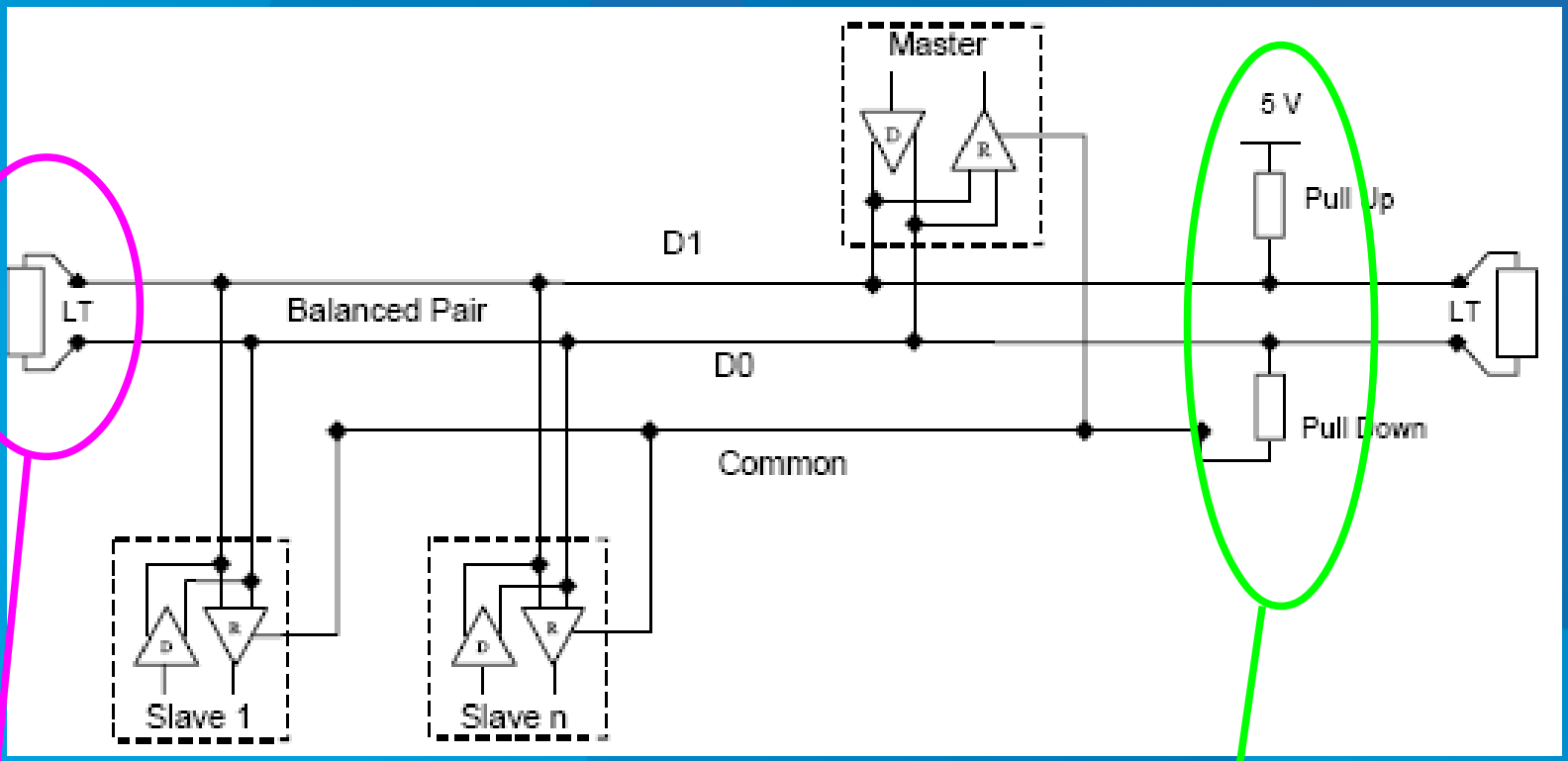
➔***RS232 :*** Transmission NRZ, niveaux de tension +/-12V référencés par rapport à la masse.

➔***RS485 :*** Transmission différentielle sur paire torsadée, niveaux de tension 0/5v.


Parasites e-m
Bits transmis
0 0 1 0 1 1 1 0 1
+12v
V RS232
-12v
+5v
RS485 : D0
0v
+5v
RS485 : D1
0v
+5v
V RS485
-5v

✔ RS485 2 fils.

Il s'agit du mode le plus répandu, tout équipement Modbus série doit permettre ce type de liaison.

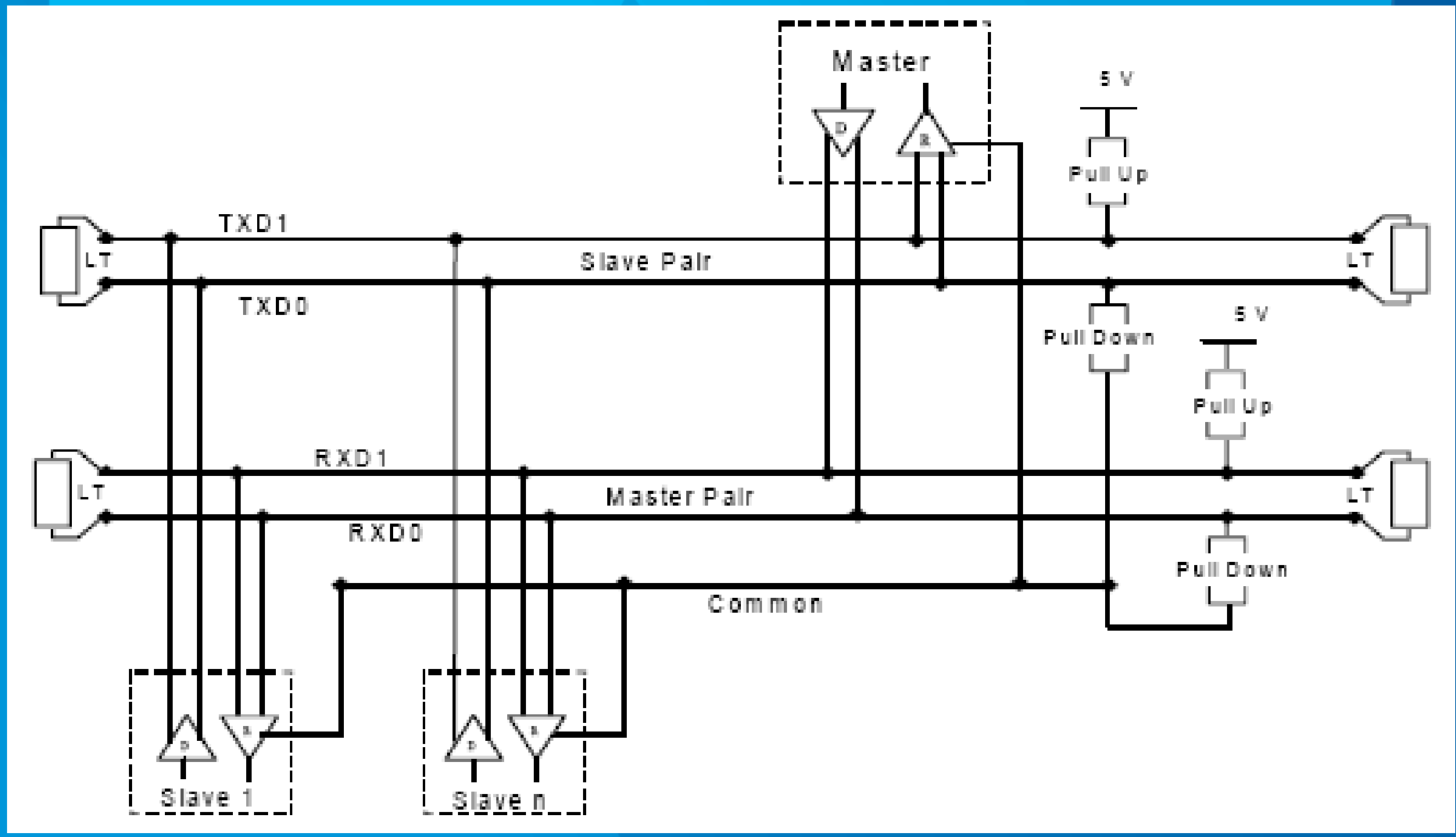


Terminaisons de bus

Résistances de polarisation

Désignation Modbus	Nom norme RS485	Direction	Description
D0	A/A'	E/S	D0=V+ pour bit = '0' D0=V- pour bit = '1'
D1	B/B'	E/S	D1=V- pour bit = '0' D1=V+ pour bit = '1'
Common	C/C'	x	Masse commune (optionnelle)

✔ RS485 4 fils. (optionnel)



Désignation Modbus	Nom norme RS485	Direction	Description
TXD0	A	S	Emission : TXD0=V+ pour bit = '0' TXD0=V- pour bit = '1'
TXD1	B	S	Emission : TXD1=V- pour bit = '0' TXD1=V+ pour bit = '1'
RXD0	A'	E	Réception : RXD0=V+ pour bit = '0' RXD0=V- pour bit = '1'
RXD1	B'	E	Réception : RXD1=V- pour bit = '0' RXD1=V+ pour bit = '1'
Common	C/C'	x	Masse commune (optionnelle)

✔ RS232.

Ce mode est réservé aux liaisons point à point pour des distances <20m.

Signal	For DCE	Required on DCE (1)	Required on DTE (1)	Description
Common	--	X	X	Signal Common
CTS	In			Clear to Send
DCD	--			Data Carrier Detected (from DCE to DTE)
DSR	In			Data Set Ready
DTR	Out			Data Terminal Ready
RTS	Out			Request to Send
RXD	In	X	X	Received Data
TXD	Out	X	X	Transmitted Data

✔ Débit binaire :

Pour l'ensemble des modes de transmissions précédemment décrits, le débit binaire par défaut est de 19 200 Bauds.

✔ Connecteurs :

➢ Modbus RS485 2 fils :

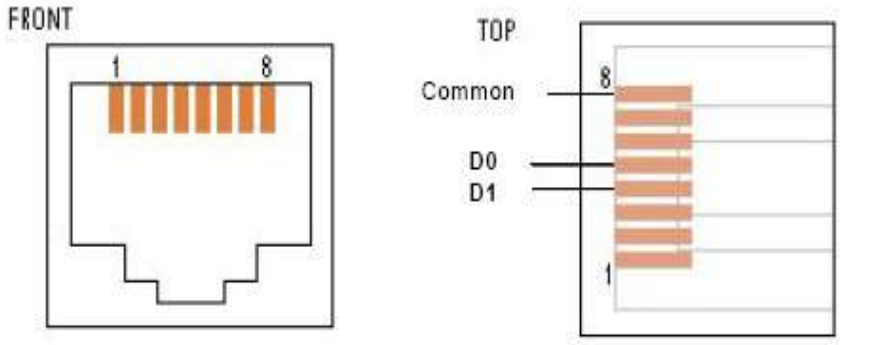
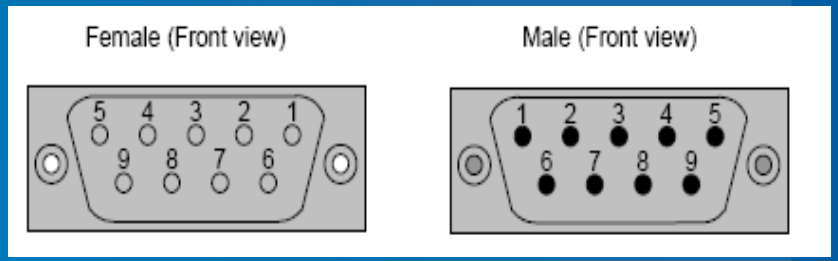
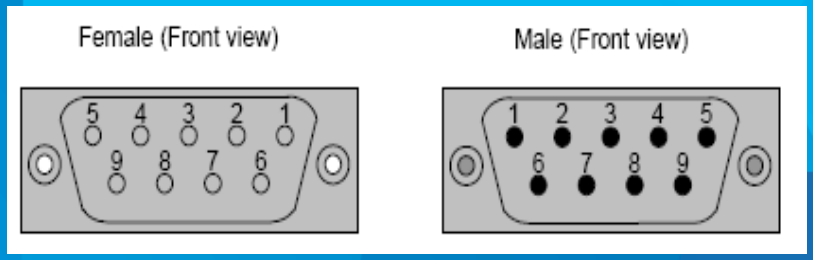


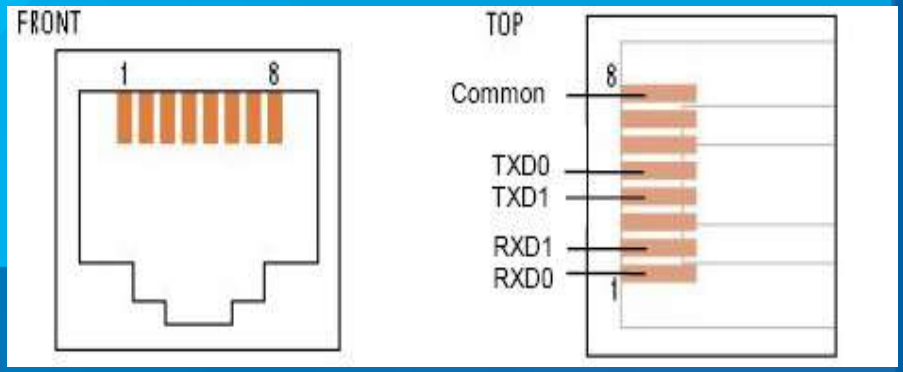
Figure 24: 2W- MODBUS on RJ45 connector (required pin-out)



Pin on RJ45	Pin on D9-shell	Level of requirement	IDv Circuit	ITr Circuit	EIA/TIA-485 name	Description for IDv
3	3	optional	PMC	--	--	Port Mode Control
4	5	**required**	D1	D1	B/B'	**Transceiver terminal 1, V1 Voltage** (V1 > V0 for binary 1 [OFF] state)
5	9	**required**	D0	D0	A/A'	**Transceiver terminal 0, V0 Voltage** (V0 > V1 for binary 0 [ON] state)
7	2	recommended	VP	--	--	Positive 5...24 V D.C. Power Supply
8	1	**required**	Common	Common	C/C'	**Signal and Power Supply Common**

➢ Modbus RS485 4 fils :





Pin on RJ45	Pin on D9-shell	Level of requirement	IDv Signal	ITr Signal	EIA/TIA-485 name	Description for IDv
1	8	**required**	RXD0	RXD0	A'	**Receiver terminal 0, Va' Voltage** (Va' > Vb' for binary 0 [ON] state)
2	4	**required**	RXD1	RXD1	B'	**Receiver terminal 1, Vb' Voltage** (Vb' > Va' for binary 1 [OFF] state)
3	3	optional	PMC	--	--	Port Mode Control
4	5	**required**	TXD1	TXD1	B	**Generator terminal 1, Vb Voltage** (Vb > Va for binary 1 [OFF] state)
5	9	**required**	TXD0	TXD0	A	**Generator terminal 0, Va Voltage** (Va > Vb for binary 0 [ON] state)
7	2	recommended	VP	--	--	Positive 5...24 V DC Power Supply
8	1	**required**	Common	Common	C/C'	**Signal and Power Supply Common**

➢ Modbus RS232 :

DCE Underlined pins can be output			Circuit			DTE Underlined pins can be output		
Pin on RJ45	Pin on D9-shell	Level of requirement	Name	Description	RS232 Source	Level of requirement	Pin on RJ45	Pin on D9-shell
1	2	required	TXD	Transmitted Data	DTE	required	2	3
2	3	required	RXD	Received Data	DCE	required	1	2
3	7	optional	CTS	Clear to Send	DCE	optional	6	8
6	8	optional	RTS	Request to Send	DTE	optional	3	7
8	5	required	Common	Signal Common	--	required	8	5

✔ Couche Liaison de données :

2 variantes du protocole Modbus série co-existent, qui se différencient au niveau de la couche liaison de données :

➔***Modbus RTU*** et ***Modbus ASCII***.

L'implantation Modbus RTU est obligatoire sur un composant Modbus série; celle de Modbus ASCII étant facultative.

Modbus RTU présente une meilleure efficacité en terme de débit binaire.

Nous limiterons donc notre étude à Modbus RTU, version plus répandue.

✔ Format d'une trame Modbus RTU :

Les octets transportés par les trames Modbus RTU comportent par défaut 11 bits :

- ✔ 1 bit de START
- ✔ 8 bits de données
- ✔ 1 bit de parité
- ✔ 1 bit de STOP

Start	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Parité	Stop

La mise en trame des différents champs est la suivante :

Slave Address	Function Code	Data	CRC
1 byte	1 byte	0 up to 252 byte(s)	2 bytes CRC Low | CRC Hi

Les trames sont séparées par des intervalles de « silence » de durée au moins égale à 3,5 caractères hexadécimaux (1 caractère hexa = 4 bits) :

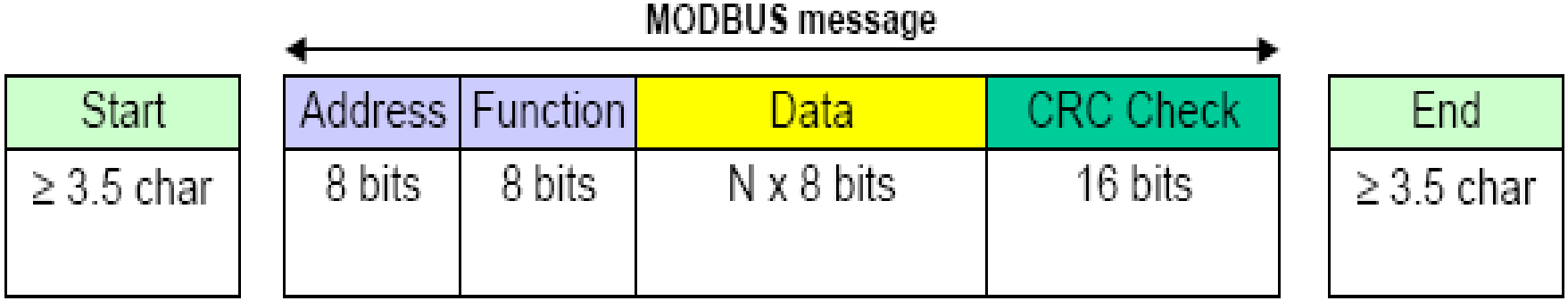


Figure 13: RTU Message Frame

✔ Calcul du CRC :

Le dernier champ de la trame permet un contrôle des données émise. Le calcul CRC (« Cyclic Redondant Code ») est une méthode très efficace et permet de détecter jusqu'à erreurs situées n'importe où dans la trame.

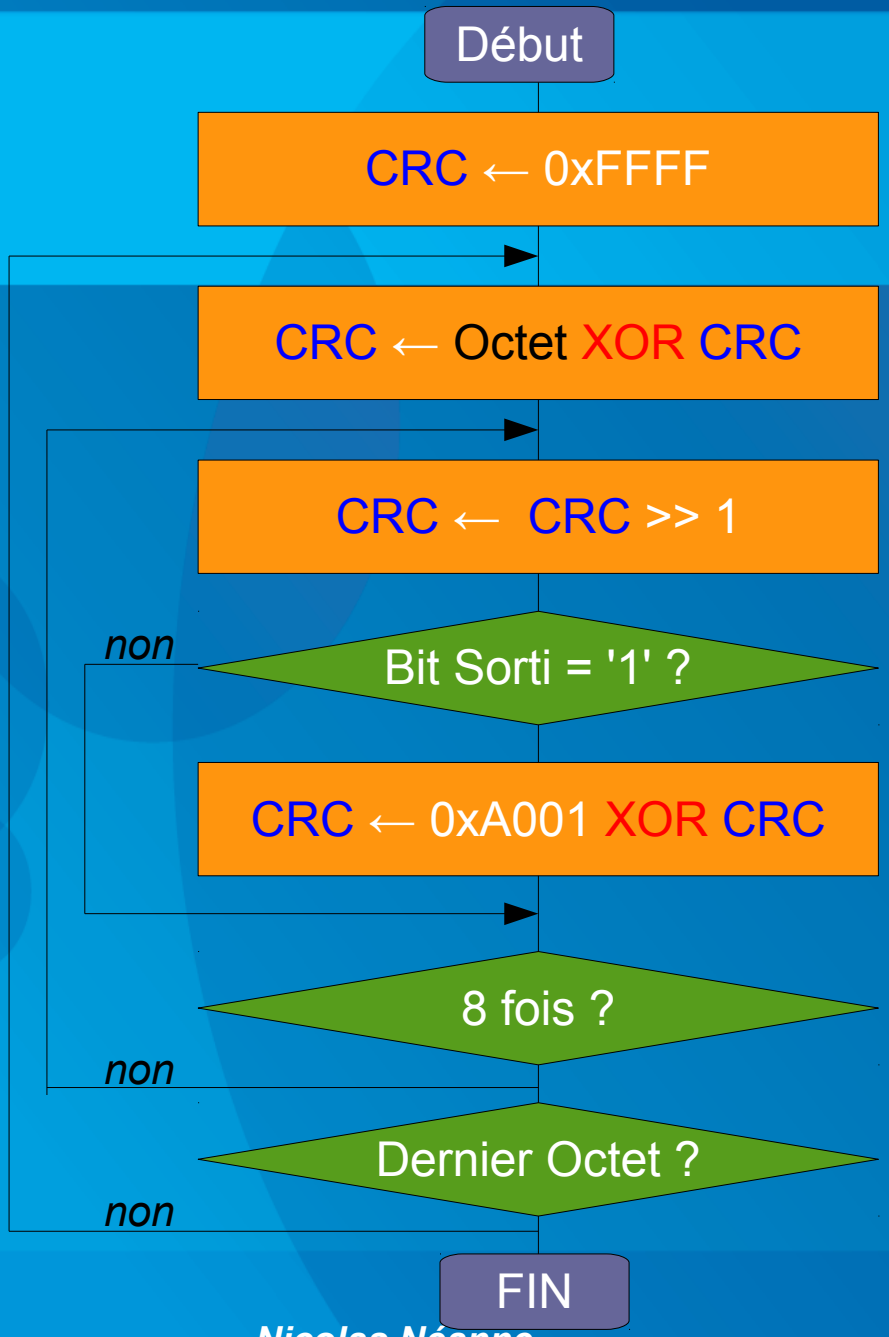
La théorie sur laquelle est basée le calcul de ce code fait appel aux fonctions polynomiales et à leur divisibilité par un polynôme particulier dit « générateur ».

Les données utiles de la trame (c'est à dire les octets de données, sans bit de START STOP et Parité) sont utilisées pour générer un polynôme dont la divisibilité est vérifiée à l'émission et à la réception.

Si les résultats ne concordent pas, c'est qu'une erreur est survenue durant la transmission.

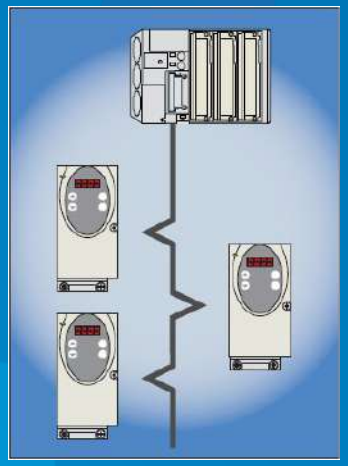
Le format du CRC utilisé dans Modbus RTU est 16 bits; l'octet de poids faible est transmis en 1° dans la trame (!).

✔ Algorithme de Calcul du CRC :



✔ Exemples d'équipements Mosbus RTU :

Variateur de Vitesse Altivar 31 :



Plusieurs centaines de variables accessibles via Modbus :

- En lecture : Courant dans les phases (mot), état thermique du moteur (mot), fins de course (bits)...

- En lecture/écriture : réglages PID (mots), consigne de vitesse (mot), M/A (bits), rampes d'accélération et de décélération (mots) etc...

Connecteur RJ45 pour réseau Modbus

API Schneider M340 :

- Communication Modbus RTU (RS485) intégrée à la CPU.

- Paramétrage de la communication via le logiciel de programmation Unity.

- Programmation des requêtes Modbus via une bibliothèque logicielle disponible sous Unity.

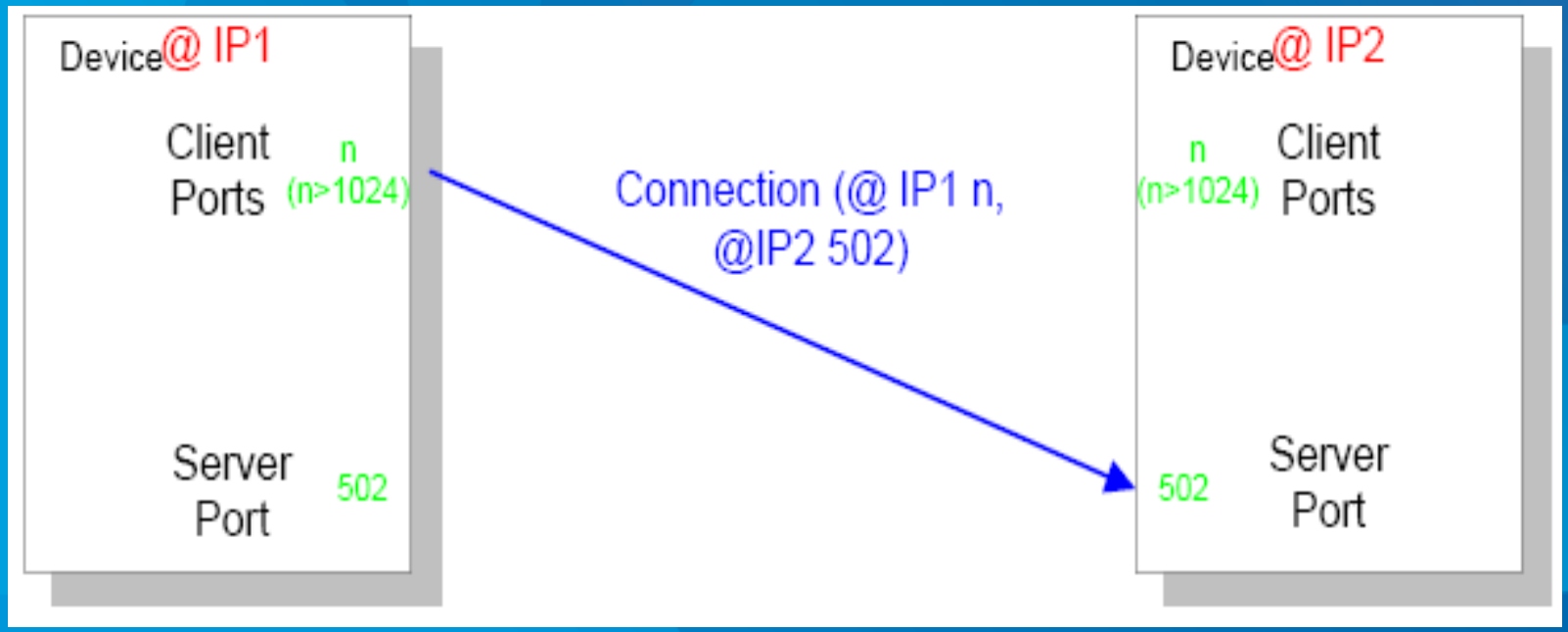
Connecteur RJ45 pour réseau Modbus

4 – Modbus TCP.

• Port Modbus TCP :

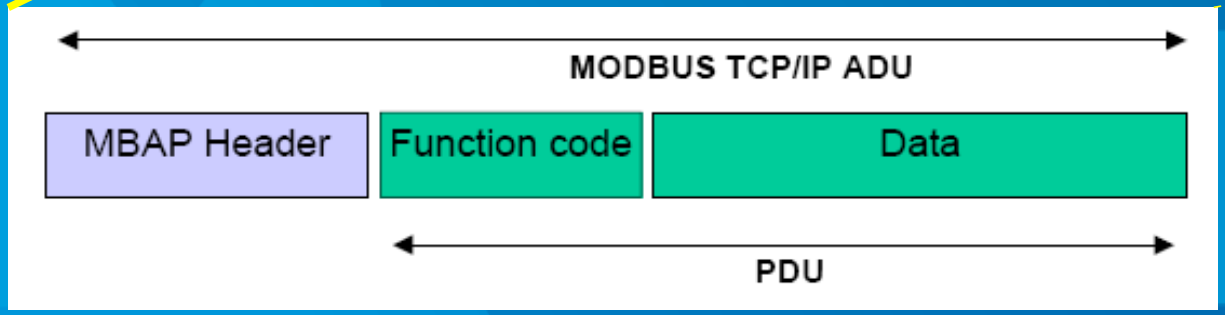
Le ***port 502*** est réservé aux communications Modbus. L'équipement serveur « écoute » donc sur ce port pour recevoir les requêtes émises par le client.

Celles-ci sont émises par le client sur un port >1024 :

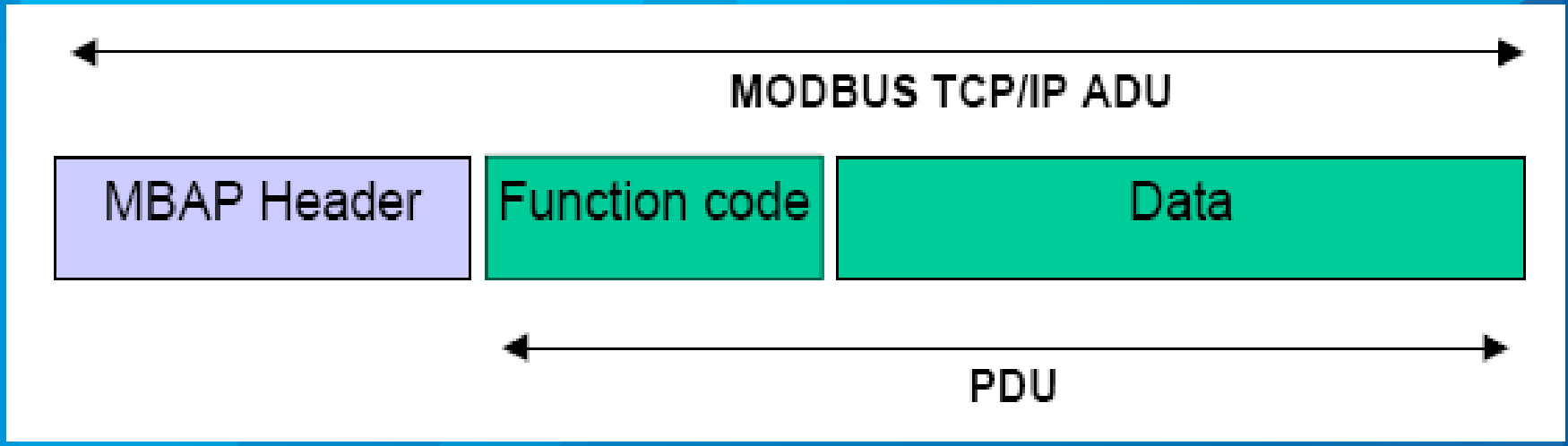


• Encapsulation des trames :

Le protocole Modbus TCP permet d'***encapsuler*** des trames Modbus PDU dans des ***trames Ethernet***, et ainsi offrir les ***services Modbus*** ce type de réseau :



La trame Modbus TCP est constituée d'une trame Modbus-TCP précédée d'une ***en-tête*** baptisée « ***MBAP Header*** » :



L'absence de champ de contrôle propre à la trame Modbus-TCP s'explique par le fait que celui-ci est déjà réalisé par la couche 2 d'Ethernet (***CRC 32 bits***).

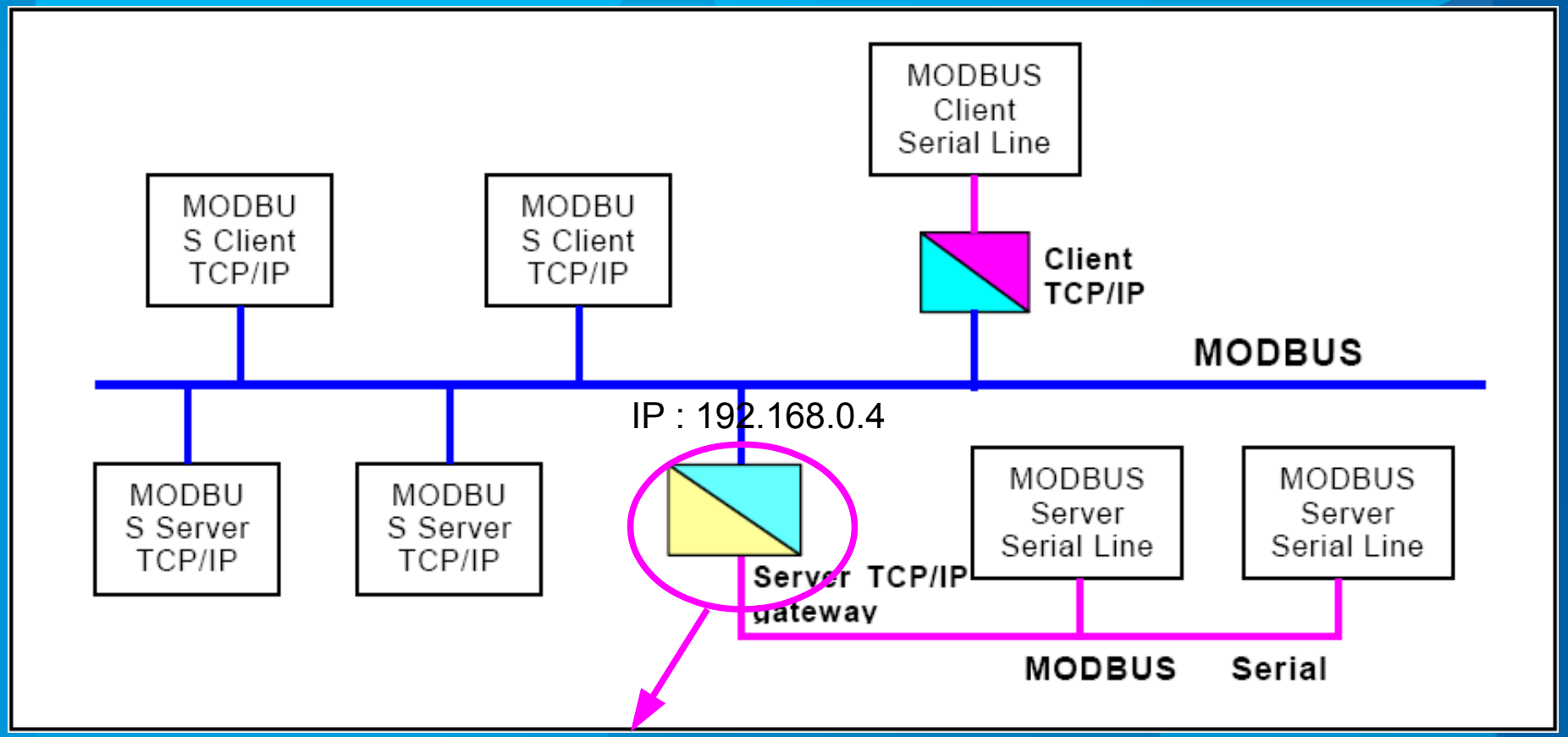
L' en-tête ***MBAP Header*** comporte 4 champs :

✔ **Transaction Identifier (2 octets) :** Ce champ est utilisé pour identifier les transaction circulant sur le réseau, afin que le client puisse faire le lien entre une ***requête*** qu'il a émise et une ***confirmation*** qu'il reçoit.

✔ **Protocol Identifier (2 octets) :** Ce champ permet d'utiliser plusieurs variantes de protocoles et de les distinguer; pour Modbus, ce champ est à ***0x00***.

✔ **Length (2 octets) :** Ce champ indique la taille (en octets) de la trame Modbus à (partir du champ suivant).

✔ **Unit Identifier (1 octet) :** Ce champ est utilisé lorsque la trame est adressée à une passerelle Modbus-TCP / Modbus-série, afin d'identifier l'adresse Modbus de l'esclave sur cette ligne :



Passerelle Modbus-TCP / Modbus série *Unit ID =1* *Unit ID =2*

Le contrôle sur chacun de ces champs pour un client et un serveur est récapitulé dans le tableau suivant :

	Client	*Serveur*
Transact. ID	*Initialise la valeur*	*Retourne au client la valeur qu'il a initialisée*
Protocole ID	*Initialise la valeur*	*Retourne au client la valeur qu'il a initialisée*
Length	*Initialise la valeur*	*Initialise la valeur*
Unit. ID	*Initialise la valeur*	*Retourne au client la valeur qu'il a initialisée*

➢ *Exemple :*

Ecrire la trame MB-ADU complète émise par l'API permettant de mettre en route la pompe (48° bit de sortie du variateur correspondant).

✔ Modbus-TCP & Les contrôleurs Wago 750-xxx :

Le contrôleur 750-849 est communique naturellement en Modbus-TCP à travers ses ports Ethernet :

→ *Toutes ses données d'entrée/sortie sont accessible via des requêtes Modbus sur le réseau Ethernet.*

Pour cela, une ***table mémoire image des E/S*** est automatiquement créée par le contrôleur en fonction des cartes présentes sur le rack.

- Les entrées pourront ainsi être accédées par les requêtes de lecture (lecture de bits d'entreée pour les E TOR, lecture de mots d'entreée pour les E Analogiques)

- Les sorties sont quant à elles accessibles en lecture ou en écriture (bits de sortie/bobines ou registres)

Accès Modbus des données de type "mot" en Modbus sur les contrôleurs Wago 750 :

Lecture Mots : FC 03 / 04		
DEC	HEX	CEI 61131-3
0	0x0000	%IW0
255	0x00FF	%IW255
256	0x0100	%QW256
511	0x01FF	%QW511
512	0x0200	%QW0
767	0x02FF	%QW255
768	0x0300	%IW256
1023	0x03FF	%IW511
12288	0x3000	%MW0
24575	0x5FFF	%MW12287
24576	0x6000	%IW512
25340	0X62FC	%IW1275
28672	0x7000	%QW512
29346	0X72FC	%QW1275

Ecriture Mots : FC 06 / 16		
DEC	HEX	CEI 61131-3
0	0x0000	%QW0
255	0x00FF	%QW255
256	0x0100	%IW256
511	0x01FF	%IW511
512	0x0200	%QW0
767	0x02FF	%QW255
768	0x0300	%IW256
1023	0x03FF	%IW511
12288	0x3000	%MW0
24575	0x5FFF	%MW12287
24576	0x6000	%QW512
25340	0X62FC	%QW1275
28672	0x7000	%QW512
29346	0X72FC	%QW1275

L'adressage CEI est celui utilisé dans les programmes des API. Les adresses de mot ont la forme suivante :

% I W 01 → 2° Mot d'entrée
% Q W 20 → 21° mot de sortie
% M W 31 → 32° mot interne

Accès Modbus des données de type "bit" en Modbus sur les contrôleurs Wago 750 :

Lecture Bits : FC 01 / 02		
DEC	HEX	CEI 61131-3
0	0x0000	%IX0.0
255	0x00FF	%IX15.15
256	0x0100	%IX16.0
511	0x01FF	%IX31.15
512	0x0200	%QX0.0
767	0x02FF	%QX15.15
768	0x0300	%QX16.0
1023	0x03FF	%QX31.15
4096	0x1000	%QX256.0
8191	0x1FFF	%QX511.15
8192	0x2000	%IX256.0
12287	0x2FFF	%IX511.15

Ecriture Bits : FC 05 / 15		
DEC	HEX	CEI 61131-3
0	0x0000	%QX0.0
255	0x00FF	%QX15.15
256	0x0100	%QX16.0
511	0x01FF	%QX31.15
512	0x0200	%QX0.0
767	0x02FF	%QX15.15
768	0x0300	%QX16.0
1023	0x03FF	%QX31.15
4096	0x1000	%IX256.0
8191	0x1FFF	%IX511.15
8192	0x2000	%IX256.0
12287	0x2FFF	%IX511.15

Les adresses CEI des variables de type bit ont la forme suivante :

% I X 10.2 → bit d'entrée (3°bit du 11° mot)
% Q X 0.3 → bit de sortie
% MX0.1 → bit interne

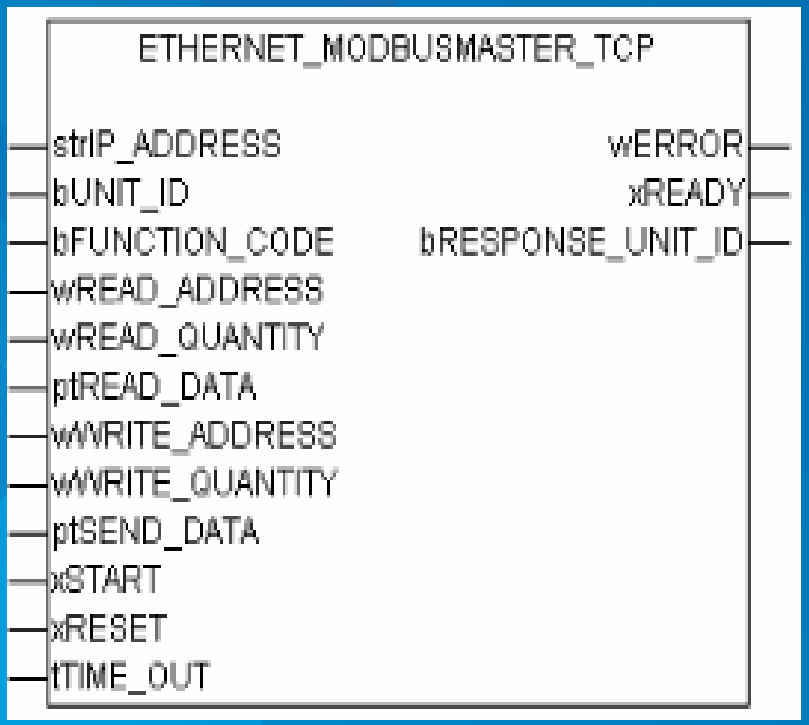
Rq :

→ les bits internes ne sont pas accessibles en Modbus TCP sur ces contrôleurs

→ Le n° d'adresse CEI pour les bits est composé d'un n°de mot et d'un n° de bit à l'intérieur de ce mot (poids fort = 15, poids faible=0)

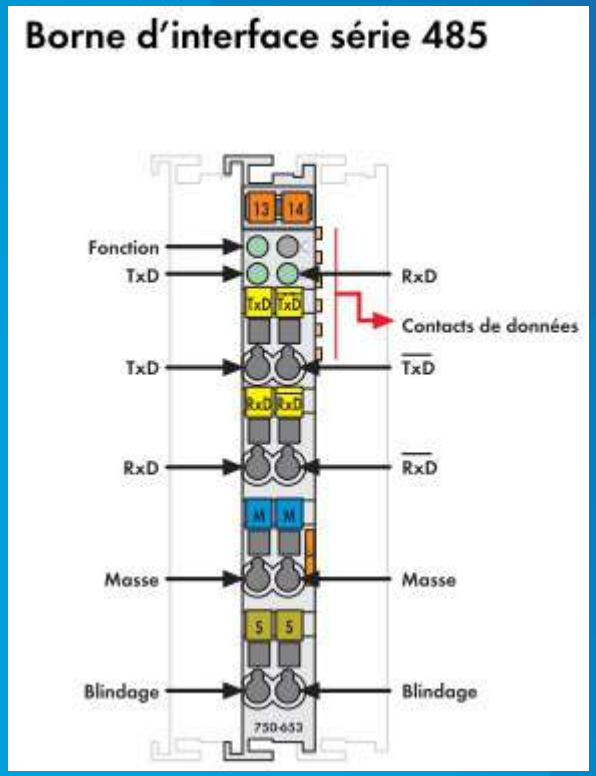
→ Par ailleurs, des blocs fonctionnels de communication permettent au programme du contrôleur d'initier une requête Modbus-TCP :

Bloc fonctionnel « Ethernet_ModbusMaster-TCP » de la bibliothèque « Modbus_Ethernet_03.lib » :

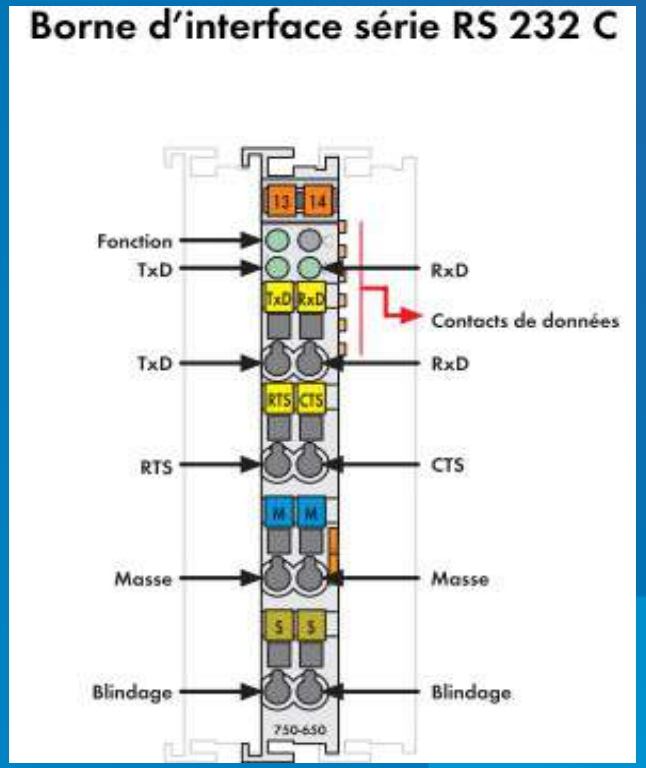


✔ Modbus RTU & Les contrôleurs Wago 750-xxx :

Des bornes spécifiques permettent d'assumer les couches physiques RS232 et RS485 utilisées en Modbus RTU (par ailleurs, les contrôleurs disposent également d'un port RD232 sur l'unité centrale) :

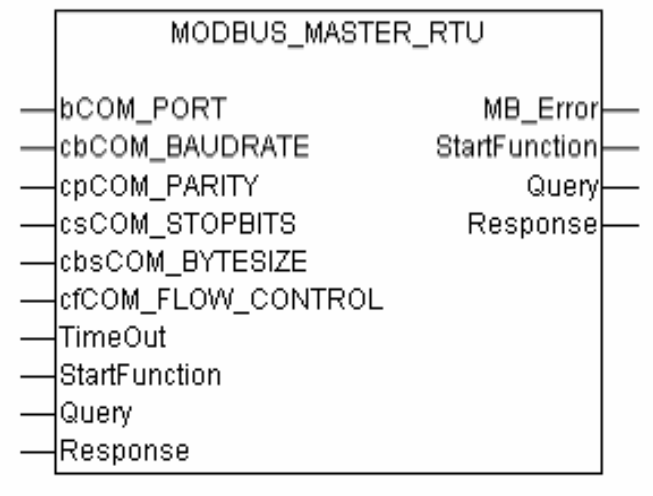


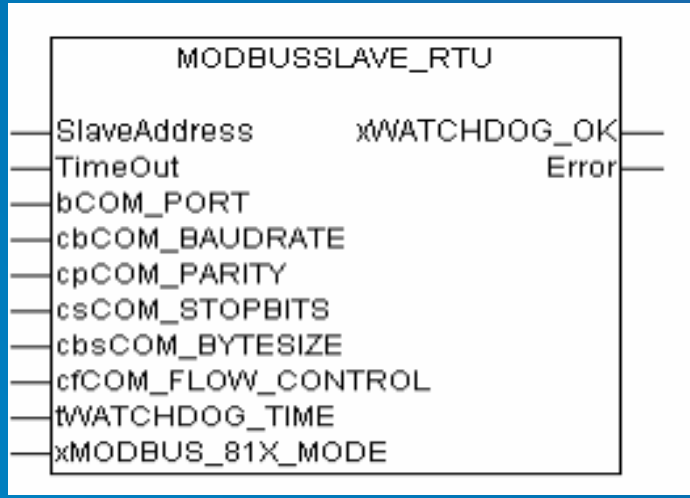
Borne 750-653



Borne 750-650

La bibliothèque « Modb_l05.lib » contient les blocs fonctionnels permettant de rendre une borne série (RS232 ou RS485) maître ou esclave :





4 / ProfiBus DP & ProfiNet

a / Présentation générale de profibus

b / Profibus-DP : Couche Physique

c / Profibus-DP : Couche liaison de données

d / Profibus-DP : Couche application

e / Profinet

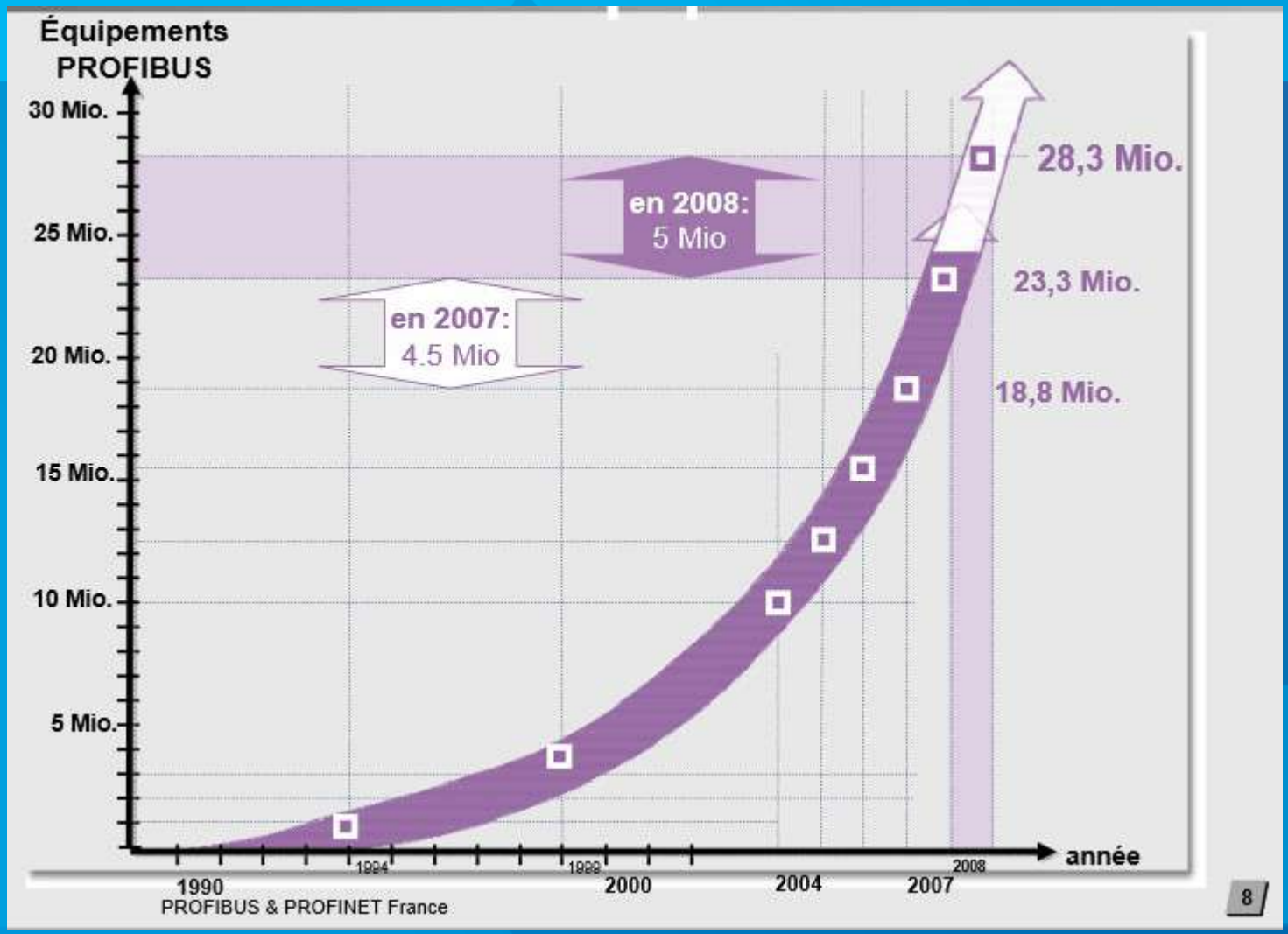
✔ Présentation :

Profibus est un protocole de communication basé sur les profils d'équipements. A son origine, Siemens, un des leaders mondiaux de l'automatisme - Profibus demeure néanmoins un réseau ouvert regroupant un grand nombre de fabricants membre de l'association *Profibus.org* :

Plusieurs déclinaisons de ce protocoles co-éxistent (Profibus DP, profibus PA, Profibus FMS, Profinet) sur différents supports (RS485, FO, Ethernet...)

Plusieurs déclinaisons de ce protocoles co-éxistent (Profibus DP, profibus PA, Profibus FMS, Profinet) sur différents supports (RS485, FO, Ethernet...)

Profibus fait partie des standards incontournables de l'automatisme et connaît depuis 20 ans un accroissement spectaculaire :



✗ Exemple d'architecture Profibus :

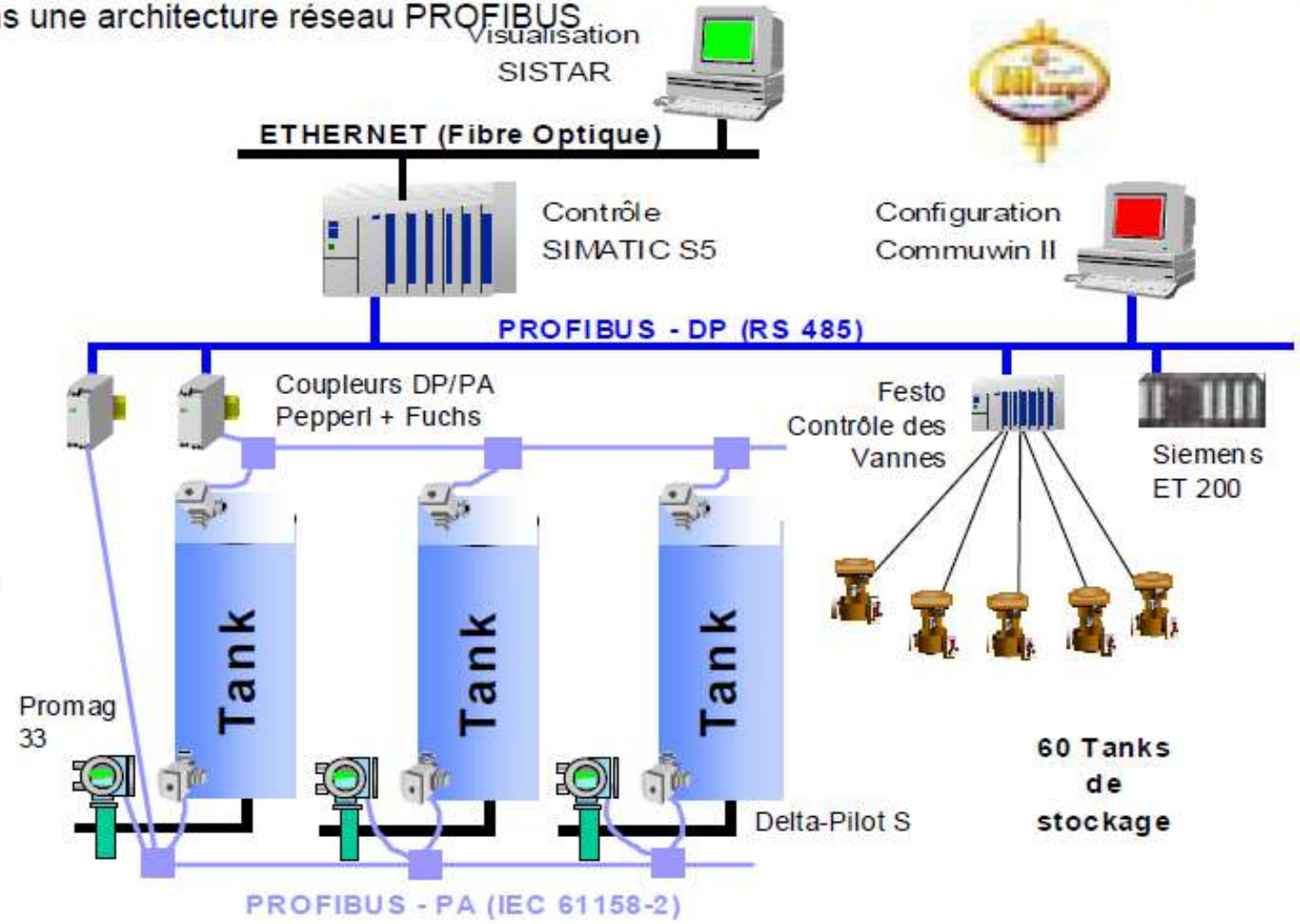
La société Bitburger, brasserie installée à Bitburg, Allemagne, gère une installation de stockage constituée de plus de 60 réservoirs dans une architecture réseau PROFIBUS.

Cette application s'appuie sur une architecture réseau à trois niveaux :

Profibus-PA pour l'instrumentation de terrain ;

Profibus-DP pour le niveau contrôle/commande de process ;

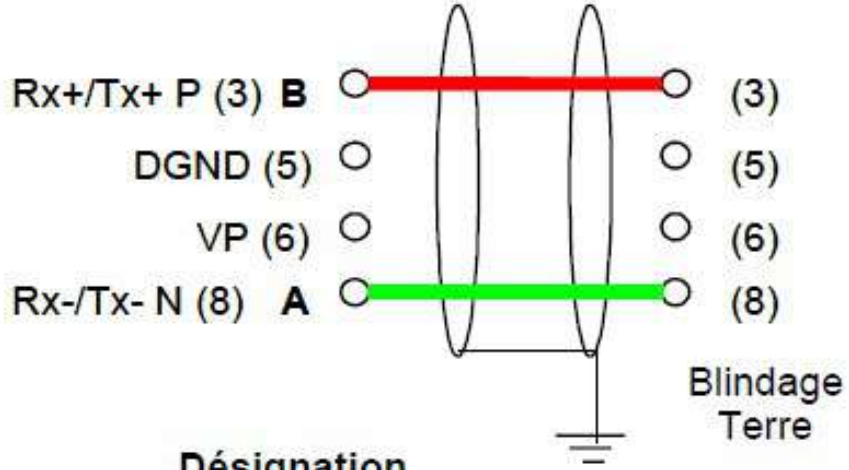
Ethernet pour la supervision de l'installation.



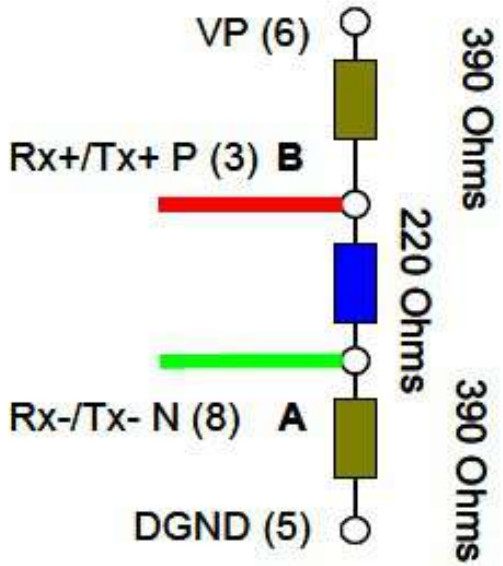
b – Profibus-DP : Couche Physique

Connecteur SubD 9

Brochage Normalisé RS 485 - 2 fils



Terminaison de Bus Polarisée



Broche	Signal	Désignation
1	**Terre**	**Terre de protection blindage**
2	M24	Masse 24 V
3 Rouge	**TX+/RX+ (B)**	**Ligne émission/Réception positif**
4	CNTR-P	Signal de contrôle pour répéteur
5	DGND	Masse Données 5V
6	VP	Signal sortie 5V pour polarisation
7	P24	Signal sortie 24 V
8 Vert	**TX- /RX- (A)**	**Ligne émission/Réception négatif**
9	CNTR-N	Signal de contrôle pour répéteur

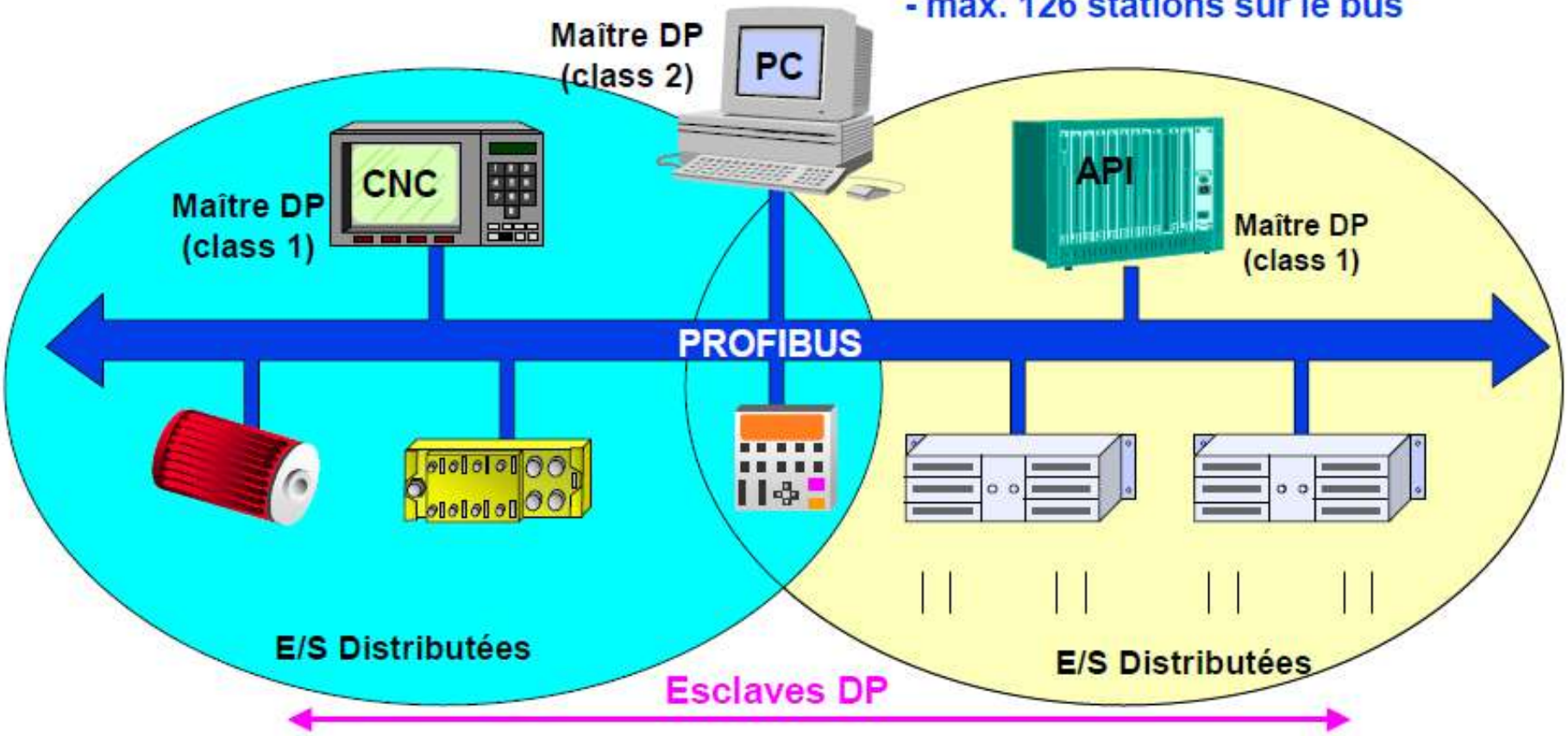
Vert A (TX-/RX-)

Rouge B (TX+/RX+)

c – Profibus-DP : Couche Liaison

Plusieurs Maître DP peuvent accéder en lecture à un esclave

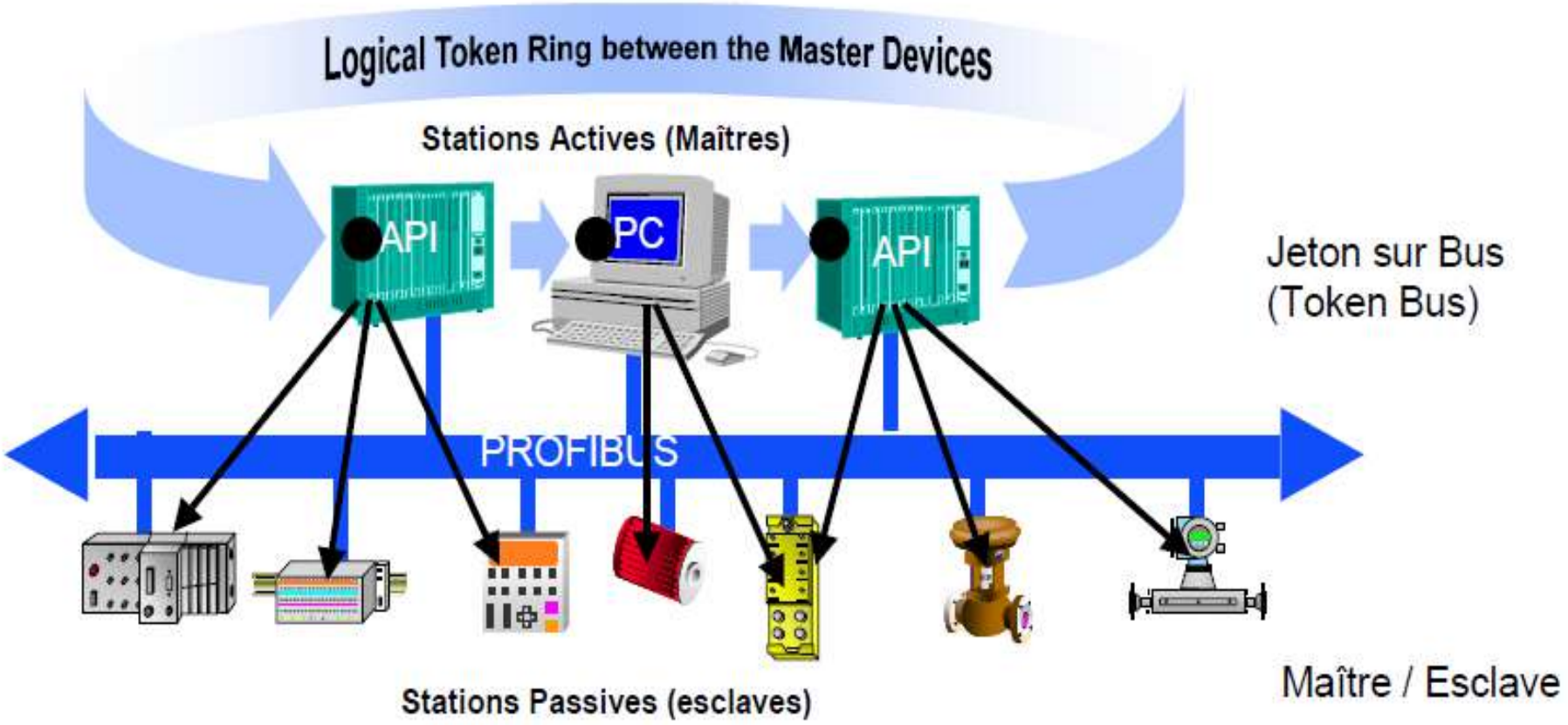
Structure MultiMaître DP consiste en
- multiple maître (classe 1 ou 2)
- 1 à max. 124 Esclaves DP
- max. 126 stations sur le bus



✔ Sous-couche MAC :

Bus à Jeton entre Stations Maîtres + Maître-Esclave avec stations passives

API, PC — Bloc E/S, IHM, Variateur, …



✔ Gestion du jeton :

Pour la gestion de l'anneau logique, chaque station active, mémorise trois paramètres :

- **TS (This Station)** : @ station locale active
- **PS (Previous Station)** : @ station active précédente dans l'anneau logique
- **NS (Next Station)** : @ station active suivante dans l'anneau logique

Et gère trois tables internes :

LAS : List of Active Station

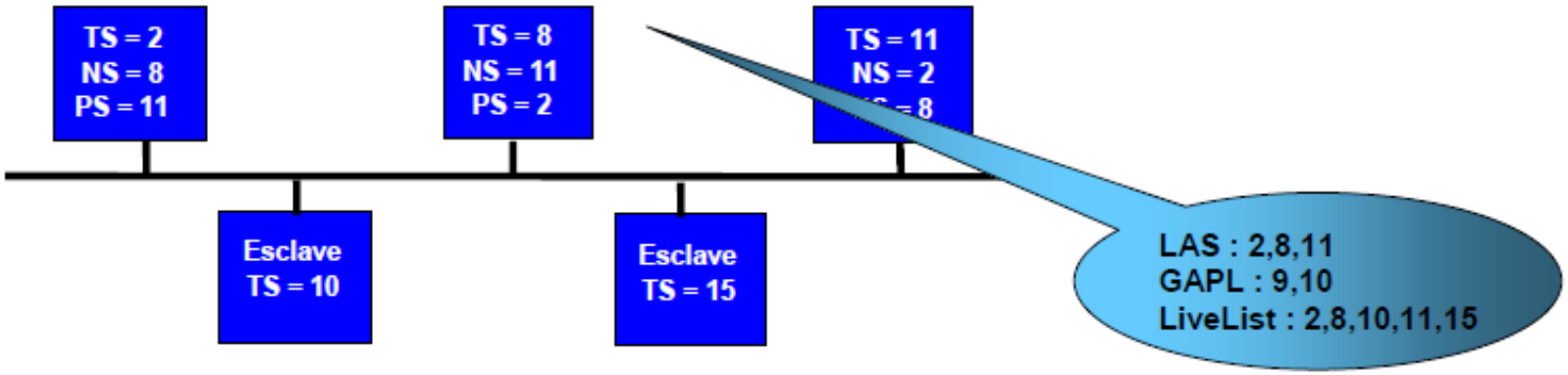
Liste (commune) de toutes les adresses des stations actives sur le réseau, constituée pendant l 'état d 'écoute du jeton (« Listen Token ») après mise sous tension

GAPL : Gap List

Liste spécifique à chaque station active, contenant les adresses manquantes entre cette station et la prochaine dans l 'anneau.

Live List

Liste (commune) des adresses de toutes les stations présentes sur le réseau actives et passives.

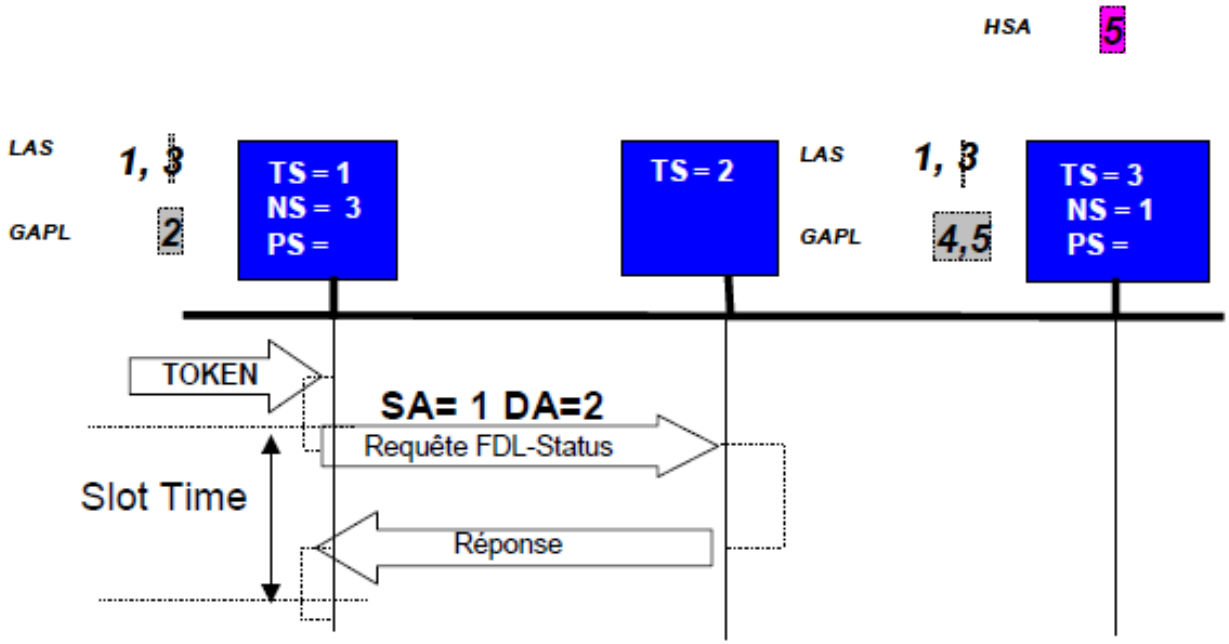


✔ Mise à jour de l'anneau :

Requête FDL-Status

3 Réponses possibles :

- Station **active Prête** à entrer dans l 'anneau
- Station **active Non Prête**
- Station **Passive**



« **G** » ou « **GapFactor** » Facteur d'actualisation de la GAPLIST

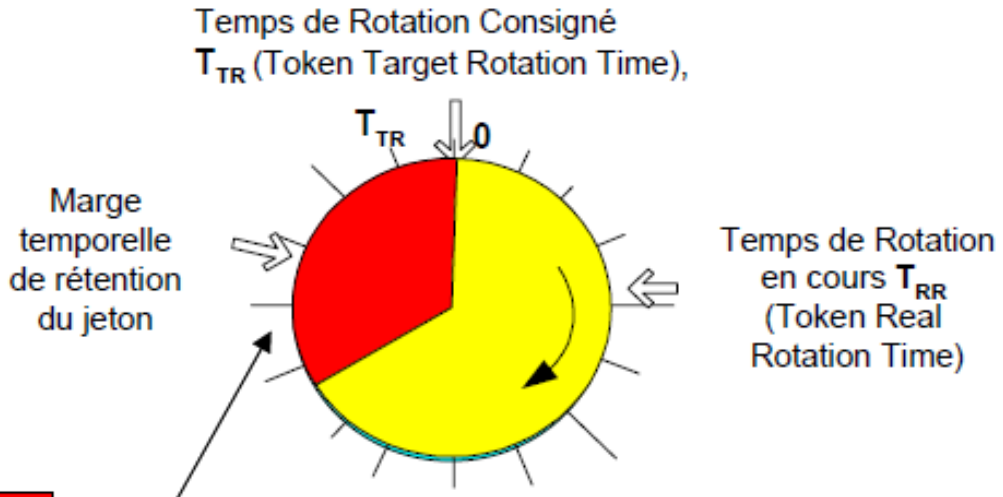
Cycliquement à chaque jeton, une station active met à jour sa **GAPLIST** par une FDL-Request d 'un numéro prélevé dans la GAPLIST. Le facteur d'actualisation du GAP est un multiple du Temps de rotation du jeton (TTR). Le laps de temps entre deux actualisations est donné par la formule : TTR * G (de 1à 100)

✔ Un réseau déterministe :

Méthode hybride : Token Bus + Maître / Esclave ⇨ Fonctionnement Déterministe

Fonctionnement rapide et sécurisé : temps de rotation du jeton dans l'anneau logique est surveillé en permanence par chaque station active.

Chaque station maître surveille le temps de rotation du jeton (intervalle entre deux réceptions de jeton consécutives) T_{RR} **(Token Real Rotation Time)** et le compare à un temps de référence à la configuration du réseau T_{TR} (Token Target Rotation Time)



Le temps restant est utilisable par la station active pour émettre des télégrammes

$$T_{RR} < T_{TR}$$

✔ Un réseau déterministe (suite) :

Le calcul du T_{TR} **(Token Target Rotation Time)** est effectué à la configuration du réseau sur les stations maître, en fonction :

- du **nombre de stations actives**,
- de la **taille des messages** de priorité haute,
- en intégrant **une marge de temps** nécessaire à l'envoi de message de priorité basse
- en intégrant **d'éventuelles tentatives de ré-émission** de télégrammes.

La formule de calcul de la valeur minimale de T_{TR} est donnée ci-dessous, elle prend en compte l'envoi d'un message de priorité haute par jeton :

$$\min T_{TR} = na * (T_{TC} + high\ T_{MC}) + k * low\ T_{MC} + mt * RET\ T_{MC}$$

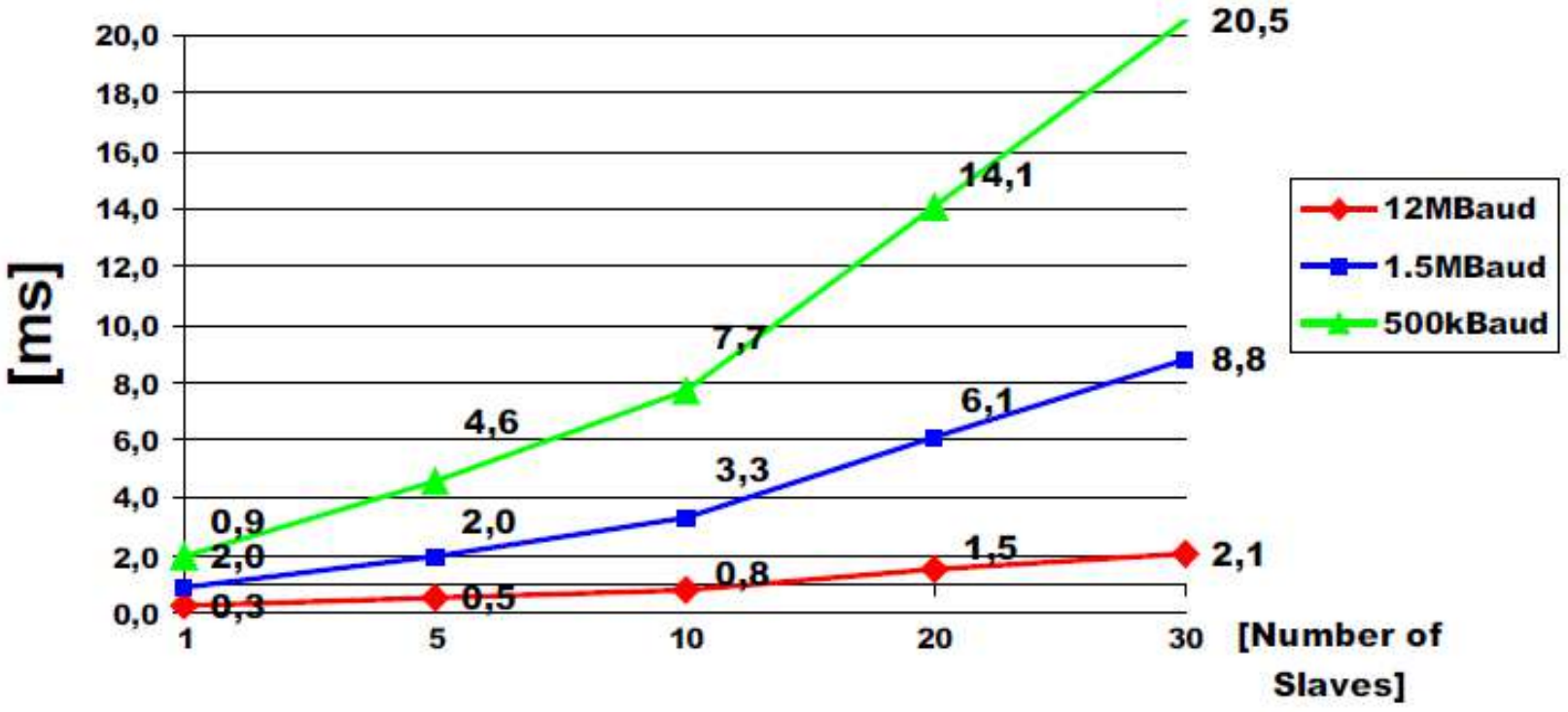
avec :

na	: nombre de stations maîtres
k	: nombre estimé de messages de priorité basse par jeton
T_{TC}	: temps de cycle du jeton correspondant au temps d'envoi d'une trame jeton
T_{MC}	: temps de cycle transaction réseau : requête + réponse ou acquittement
mt	: nombre maximum de répétitions de télégramme
RET T_{MC}	: temps de cycle de répétition d'un télégramme

✔ Temps de bus :

Le maître lit 2 octets d 'entrées et écrit 2 octets de sorties par un service SRD (Send and Request Data) dans chaque esclave :

- ➔ à 0,5 MBps = 18 ms pour 30 esclaves
- ➔ à 1,5 MBps = 6 ms pour 30 esclaves
- ➔ à 12 MBps = 1 ms pour 30 esclaves



✔ Paramètres de CONFIGURATION :

HSA = Highest Station Address : Adresse de la station active la plus élevée. Les stations passives peuvent avoir une adresse supérieure à HSA. (2 à 126)

TS = Adresse physique de cette station : 0 à 126.

RETRY_CTR = Compteur de répétitions : Nb de répétitions du télégramme en absence de réponse. (1 à 8)

TSL =Time Slot : Temps d'attente de réception : Ne peut pas être inférieur à 2 ms.

TSET = Temps mort : Temps qui peut s'écouler entre un événement (par ex. la réception de caractères ou l'écoulement d'un temps interne de surveillance) et la réaction à cet événement. Plage des valeurs : 1 à 255 b/s.

MIN_TSDR = Délai minimum entre la réception du dernier bit d'un télégramme et l'émission du premier bit du télégramme suivant. Plage des valeurs : 10 à 65535 b/s.

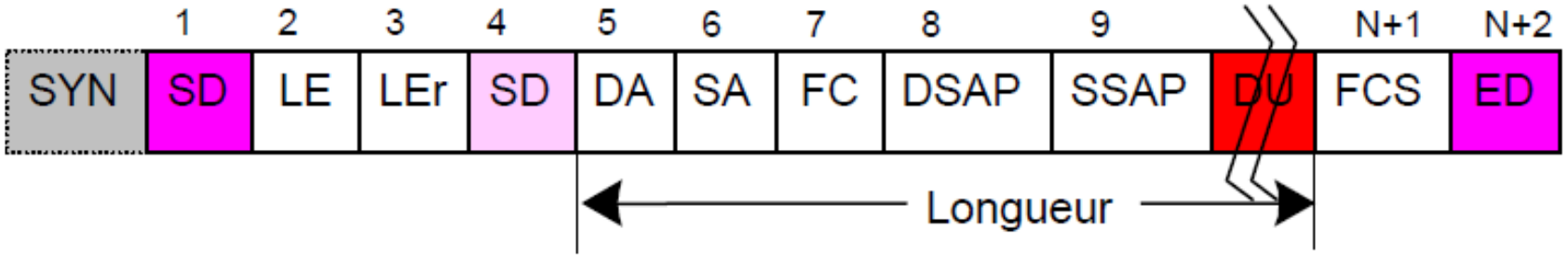
MAX_TSDR = Délai maximum entre la réception du dernier bit d'un télégramme et l'émission du premier bit du télégramme suivant.

TTR = Target- Rotation-Time : Temps de Rotation du jeton : Ce paramètre a une influence critique sur les performances du système. Cette valeur détermine le temps maximum de rotation du jeton pour toutes les stations actives présentes sur l'anneau logique. Lors de la réception du jeton, ce temps est comparé avec le temps de cycle effectivement écoulé. Selon le résultat de cette comparaison, le système détermine si la station considérée peut expédier des télégrammes (les messages de type "prioritaires" prédominent).

GAP Factor = Facteur d'actualisation de liste : Période d'émission d'une invitation pour une nouvelle station à se connecter sur le bus

✔ Format général d'un télégramme Profibus :

Trame **maximum 255 Octets**, transmission **asynchrone**, caractère sur 11 bits : 1 **Start**, 1 **Stop**, 8 **data Bits**, **parité paire.**



Sync Time : La transmission de chaque trame est précédée d'une attente de synchronisation égal à 33 Temps Bits à ' 1 '. (1 Tbit = 83 ns pour 12 MBit/s)

SYN	Temps de synchronisation (33 Tbits)
SD	Délimiteur de début de trame
LE	Longueur de la trame de DA à FCS
LEr	Répétition de la longueur de la trame
DA	Adresse Destination
SA	Adresse Source
DSAP	Point d'accès du service de destination
SSAP	Point d'accès du service de la source
DU	**Données (244 octets Maximum)**
FC	Contrôle (Requête/Réponse/Acquittement)
FCS	Clef de contrôle (Checksum)
ED	Délimiteur de fin (16H)

⇨Le **télégramme de réponse** est du même format que la trame émise ou d'un format court sur un octet d'acquittement selon le type de télégramme.

✔ Exemples de télégrammes particuliers :

1) Passage du Jeton

DCH	DA	SA

2) Longueur fixe sans Données (FDL Status Req)

10H	DA	SA	FC	FCS	16H

3) Longueur fixe avec Données (8 Octets)

A2H	DA	SA	FC	DU	FCS	16H

DA = Adresse Destination
SA = Adresse source
FC = Code Fonction
DU = Données
DSAP = SAP Destination
SSAP = SAP Source
LE = Longueur
LEr = Longueur répétée
FCS = Checksum (Mod 256 hors délimiteurs)
DCH, 10H, A2H, 68H = Délimiteur début
16H = Délimiteur de fin

4) Longueur variable avec Données (4 à 249 octets)

68H	LE	LEr	68H	DA	SA	FC	DSAP	SSAP	DU	FCS	16H

5) Acquittement court

E5H

ACK : pour des requêtes en SDA

NACK : pour des requêtes en SRD (No data available)

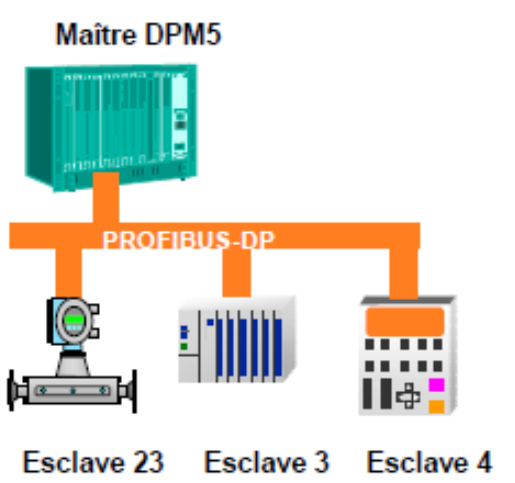
c – Profibus-DP : Couche application

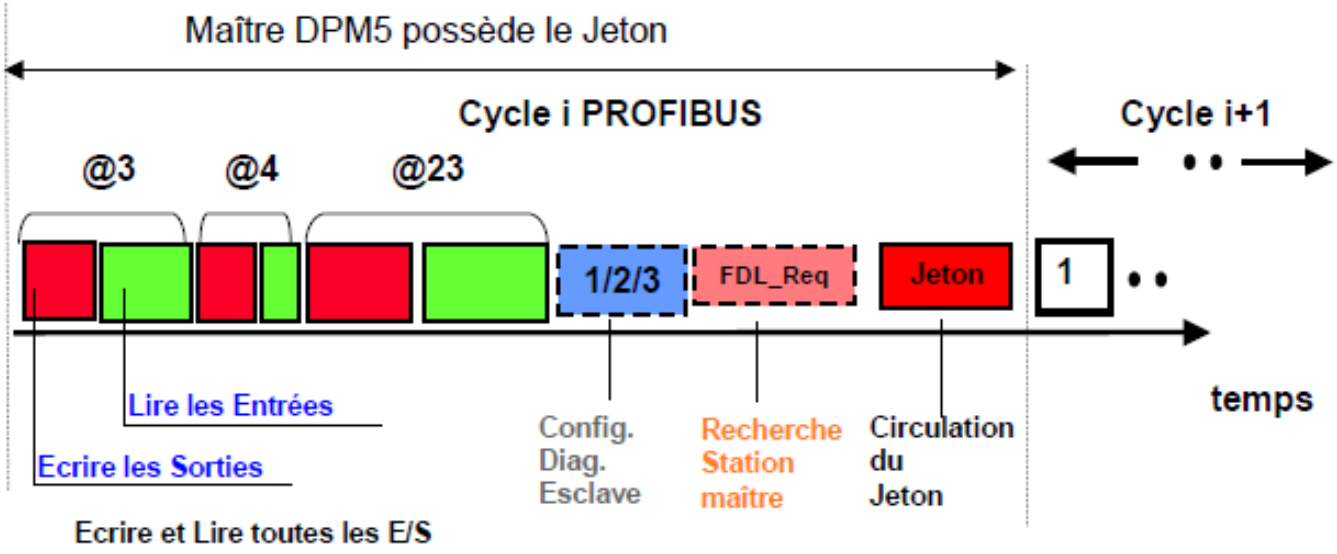
✔ Profibus DP v0 :

Service couche 2 de base Profibus : Echange cyclique des E/S de chaque esclave

Lorsqu'un maître DP possède le Jeton, il effectue :

- ⇨ Accès cyclique (polling) en lecture/écriture de toutes les E/S des esclaves DP gérés
- ⇨ Accès acyclique à tout esclave en Configuration/Diagnostic selon son état remonté précédemment
- ⇨ Recherche d'une autre station maître (FDL_Request)
- ⇨ Circulation du jeton dans l'anneau logique

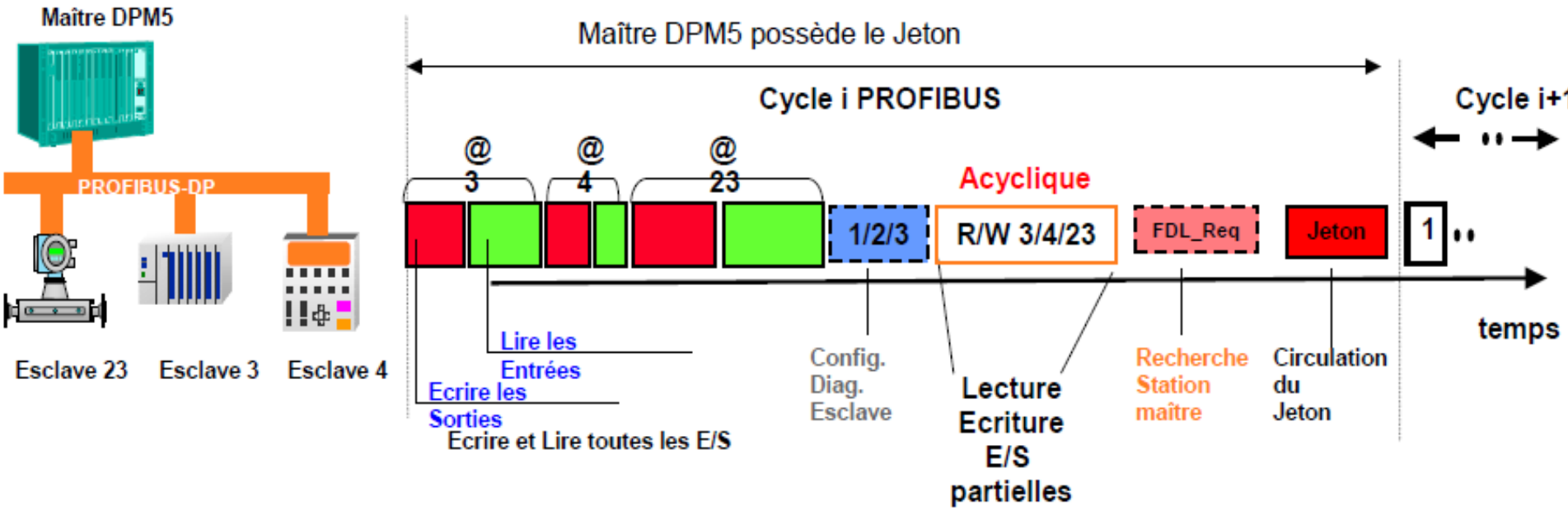




✔ Profibus DP v1 :

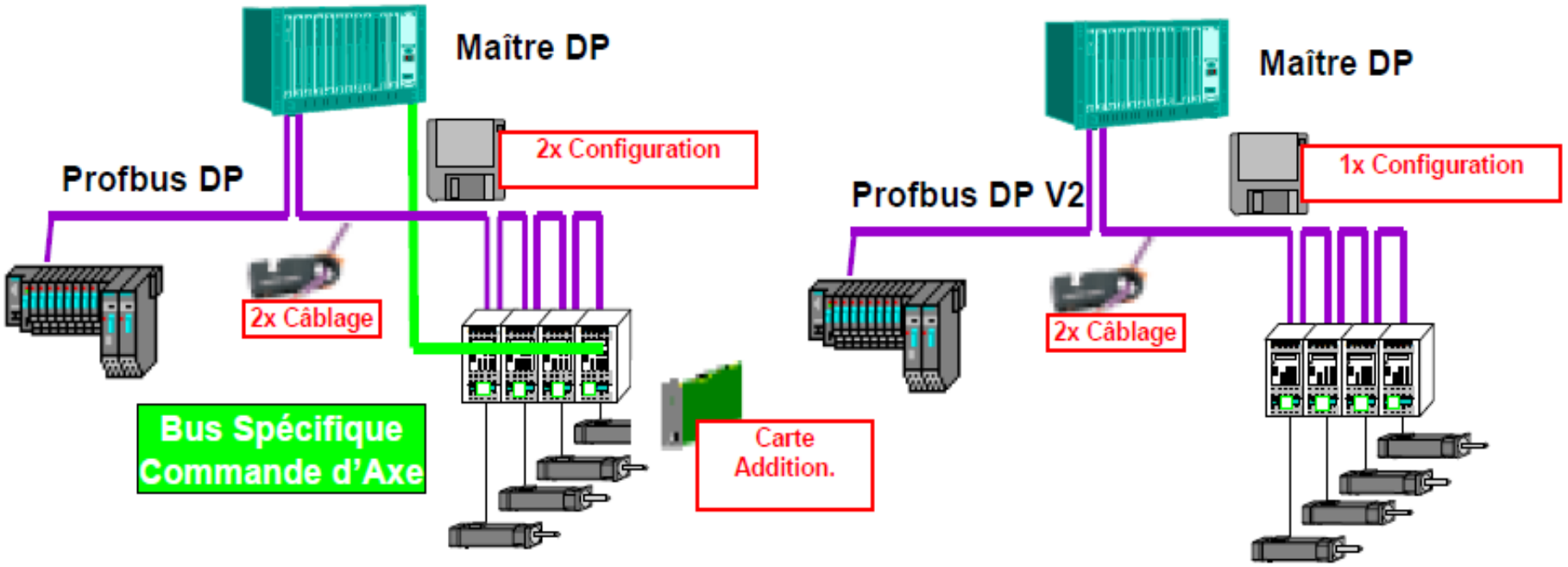
Service couche 7 étendu = DP V0 + Echange acyclique utilisateur

- ⇨ Cycle DP-V0
- ⇨ Accès acyclique sur demande à tout module d'E/S adressable individuellement
- ⇨ Paramétrage, exploitation, visualisation des alarmes (alarme d'état, de mise à jour, de constructeur) des appareils de terrain intelligents



✔ Profibus DP v2 :

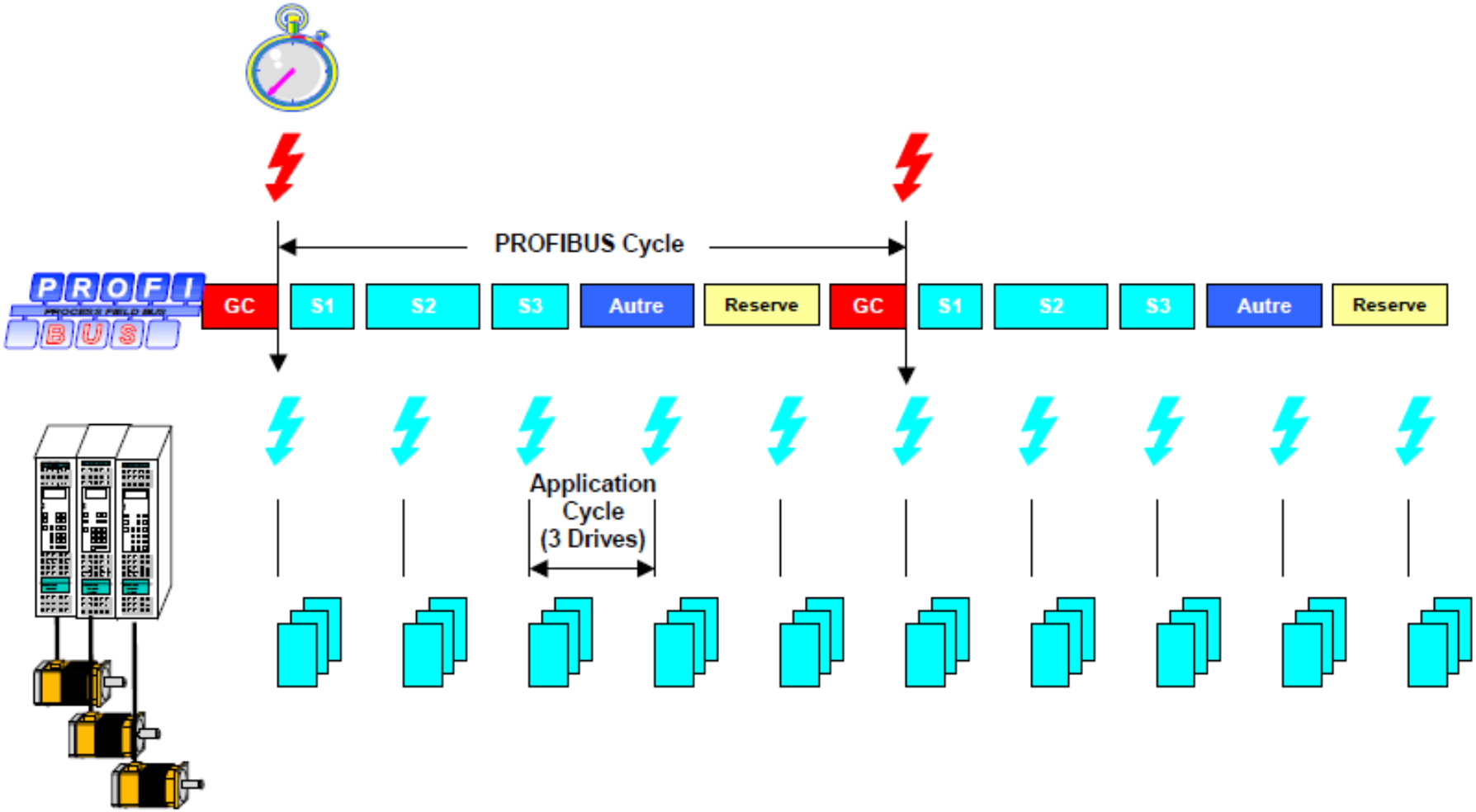
Service couche 7 étendu destiné à la commande d'axes par le réseau : **Motion Control**



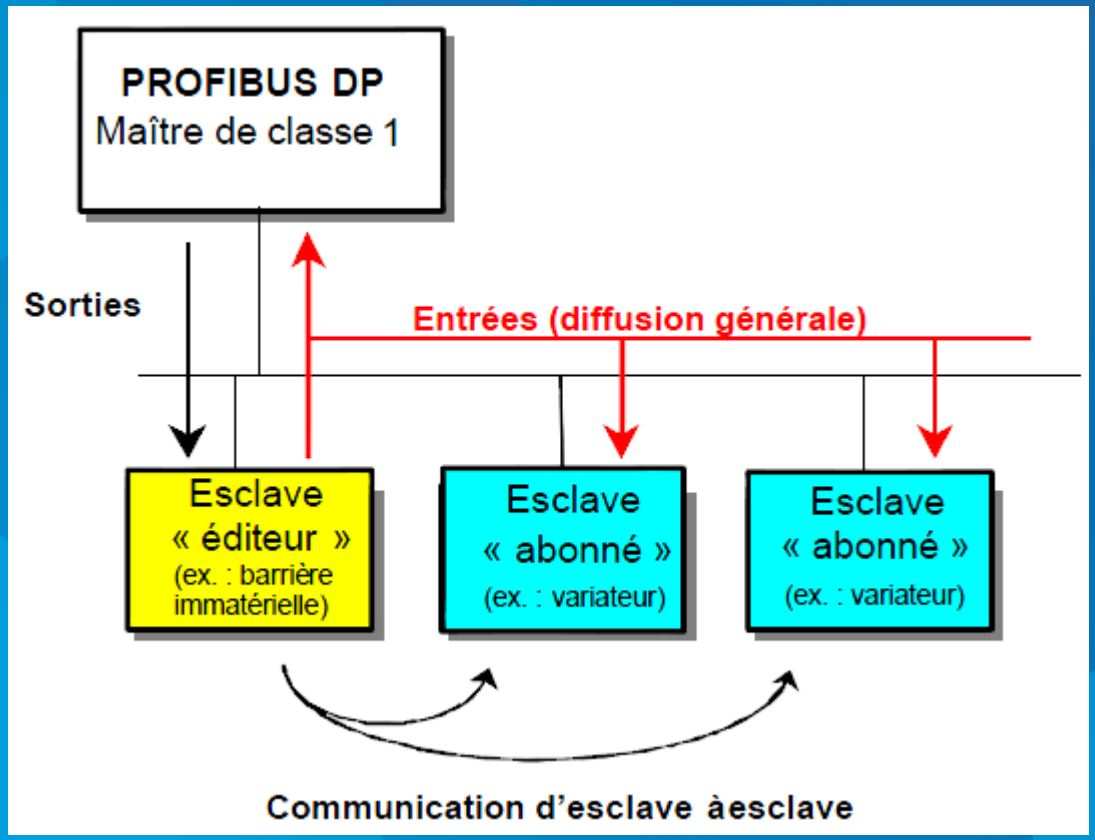
Les extensions par rapport à DP-V1 sont :

- ⇨Isochronous Mode (IsoM) : Cycles Bus DP équidistants
- ⇨Data Exchange Broadcast (DxB) : Modèle Producteur / Consommateur entre esclave
- ⇨Time Synchronization (Time Stamp) : Horodatage des données
- ⇨Redundancy concept (Redundancy) : Redondance active Maître et esclave

✔ Profibus DP v2 : Synchronisation



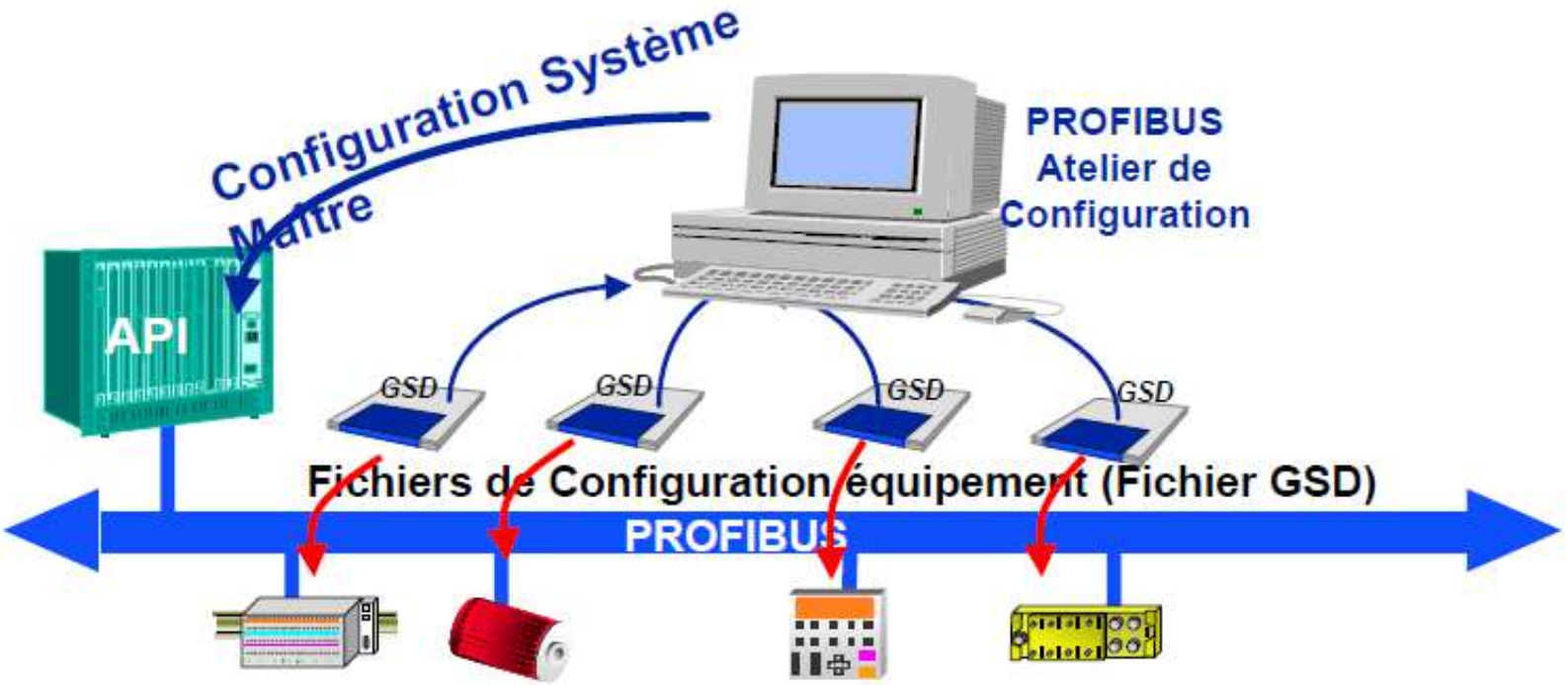
✔ Profibus DP v2 : échanges producteur/consommateur



✔ Configuration par fichiers GSD :

- Tout équipement Profibus-DP est caractérisé par un fichier de configuration «**fichier GSD**»
  - ⇨ Configuration Software : *Generic Station Description*

- **Fichier GSD** :
  - Spécifie les caractéristiques réseau d'un équipement maître ou esclave (E/S, vitesses, …)
  - Fichier ASCII, à structure normalisée, fournit par le constructeur de l'équipement.
  - ⇨ Editeur de GSD sur www.Profibus.com



✔ Exemple de fichier GSD :

```
;=============================================================
=====
;  GSD-Datei für das Produkt BIS C-6002
;  Fa. Balluff GmbH
;  Stand: 1.0 - M.Speidel 07158/173-187
;=============================================================
=====

#Profibus_DP

Vendor_Name             = "Gebhard Balluff GmbH & Co."
GSD_Revision            = 2
Model_Name              = "BIS C-6002"
Revision                = "Ausgabestand 1.0"
Ident_Number            = 0x051F
Protocol_Ident          = 0
Station_Type            = 0
FMS_supp                = 0
Hardware_Release        = "V1.0"
Software_Release        = "V1.0"
9.6_supp                = 1
19.2_supp               = 1
93.75_supp              = 1
187.5_supp              = 1
500_supp                = 1
1.5M_supp               = 1
3M_supp                 = 1
6M_supp                 = 1
12M_supp                = 1
MaxTsdr_9.6             = 60
MaxTsdr_19.2            = 60
MaxTsdr_93.75           = 60
MaxTsdr_187.5           = 60
MaxTsdr_500             = 100
MaxTsdr_1.5M            = 150
MaxTsdr_3M              = 250
MaxTsdr_6M              = 450
MaxTsdr_12M             = 800
Redundancy              = 0
Repeater_Ctrl_Sig       = 0
24V_Pins                = 0
;
;---Slave spezifische Werte---
;
Freeze_Mode_supp        = 1
Sync_Mode_supp          = 1
Auto_Baud_supp          = 1
Set_Slave_Add_supp      = 0
User_Prm_Data_Len       = 0x06
User_Prm_Data           = 0x00,0x80,0x00,0x02,0x00,0x02
Min_Slave_Intervall     = 0x0005
Modular_Station         = 1
Max_Module              = 0x01
Max_Input_Len           = 0x80
Max_Output_Len          = 0x80
Max_Data_Len            = 0x0100
Max_Diag_Data_Len       = 6
Bitmap_Device           ="BI860x2n"
Family_Ident_Systems    = 11

Module = "consistent, 2 words I and O"  0x40,0xC1,0x80,0xC1
EndModule
Module = "2 words I and O"  0x40,0x41,0x80,0x41
EndModule

Module = "consistent, 4 words I and O"  0x40,0xC3,0x80,0xC3
EndModule
Module = "4 words I and O"  0x40,0x43,0x80,0x43
EndModule

Module = "consistent, 5 words I and O"  0x40,0xC4,0x80,0xC4
EndModule
Module = "5 words I and O"  0x40,0x44,0x80,0x44
EndModule

; idem 6, 7, 8, 9 words

Module = "consistent, 10 words I and O"
0x40,0xC9,0x80,0xC9
EndModule
Module = "10 words I and O"  0x40,0x49,0x80,0x49
EndModule

Module = "consistent, 11 words I and O"
0x40,0xCA,0x80,0xCA
EndModule
Module = "11 words I and O"  0x40,0x4A,0x80,0x4A
EndModule

Module = "consistent, 12 words I and O"
0x40,0xCB,0x80,0xCB
EndModule
Module = "12 words I and O"  0x40,0x4B,0x80,0x4B
EndModule

Module = "consistent, 16 words I and O"
0x40,0xCF,0x80,0xCF
EndModule
Module = "16 words I and O"  0x40,0x4F,0x80,0x4F
EndModule

Module = "consistent, 32 words I and O"
0x40,0xDF,0x80,0xDF
EndModule
Module = "32 words I and O"  0x40,0x5F,0x80,0x5F
EndModule

Module = "consistent, 64 words I and O"
0x40,0xFF,0x80,0xFF
EndModule
Module = "64 words I and O"  0x40,0x7F,0x80,0x7F
EndModule
```

Equipt BIS C-6002

Identification Profibus 0x051F

Profibus DP

Esclave DP

Vitesse Bus

Délai max réponse esclave
Unité Temp bit

Esclave Supporte mode synchro E/S

Vitesse auto config

Adr esclave non config par maître

Nb octets param. et valeurs défaut

Fréquence Maxi polling (*100 µs)

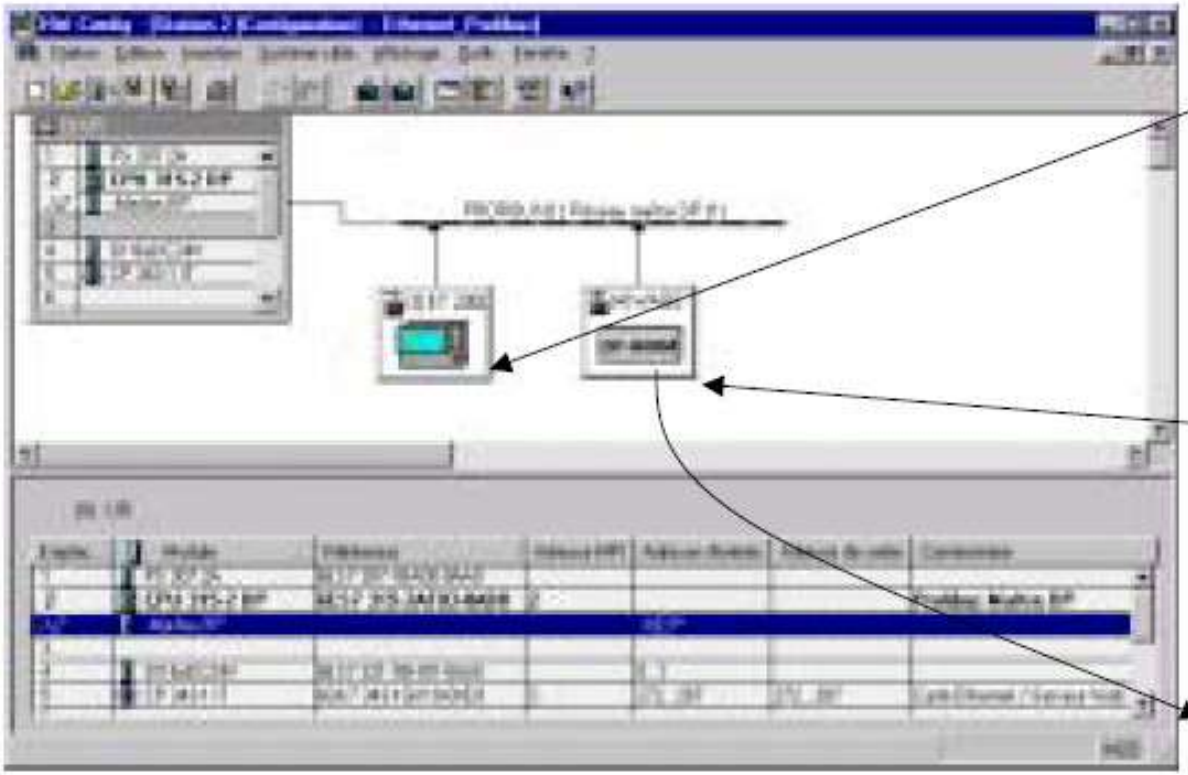
Station modulaire :
1 seul module
128 octets E et S maxi
Cumul maxi 256 octets

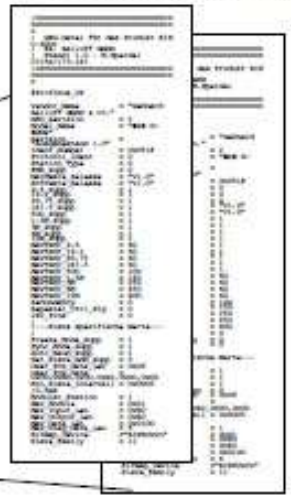
6 octets de diag esclave

Choix de module possible sur l'esclave
Et
Taille d'E/S

Configuration Logique du Réseau par "Drag & Drop" des Fichiers GSD

Catalogue de Fichiers GSD

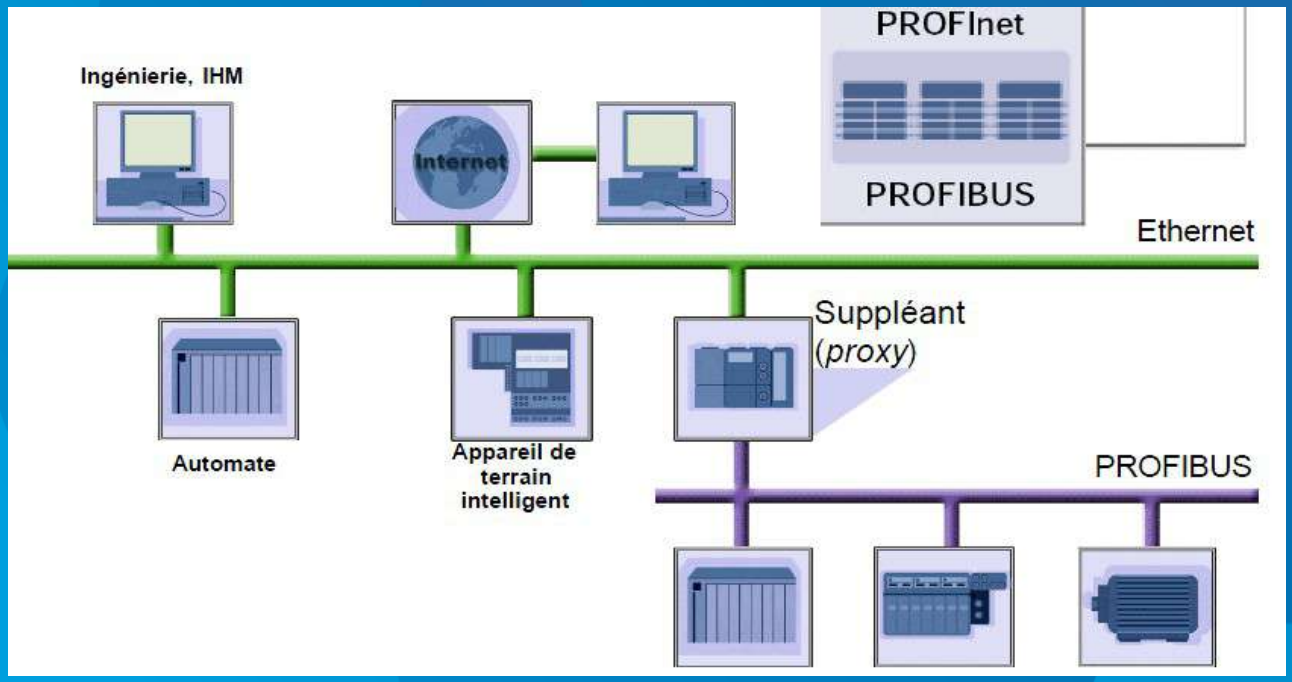




Instanciation de la configuration de l'esclave (1 Carte E , 1 Carte S)

d – Profinet :

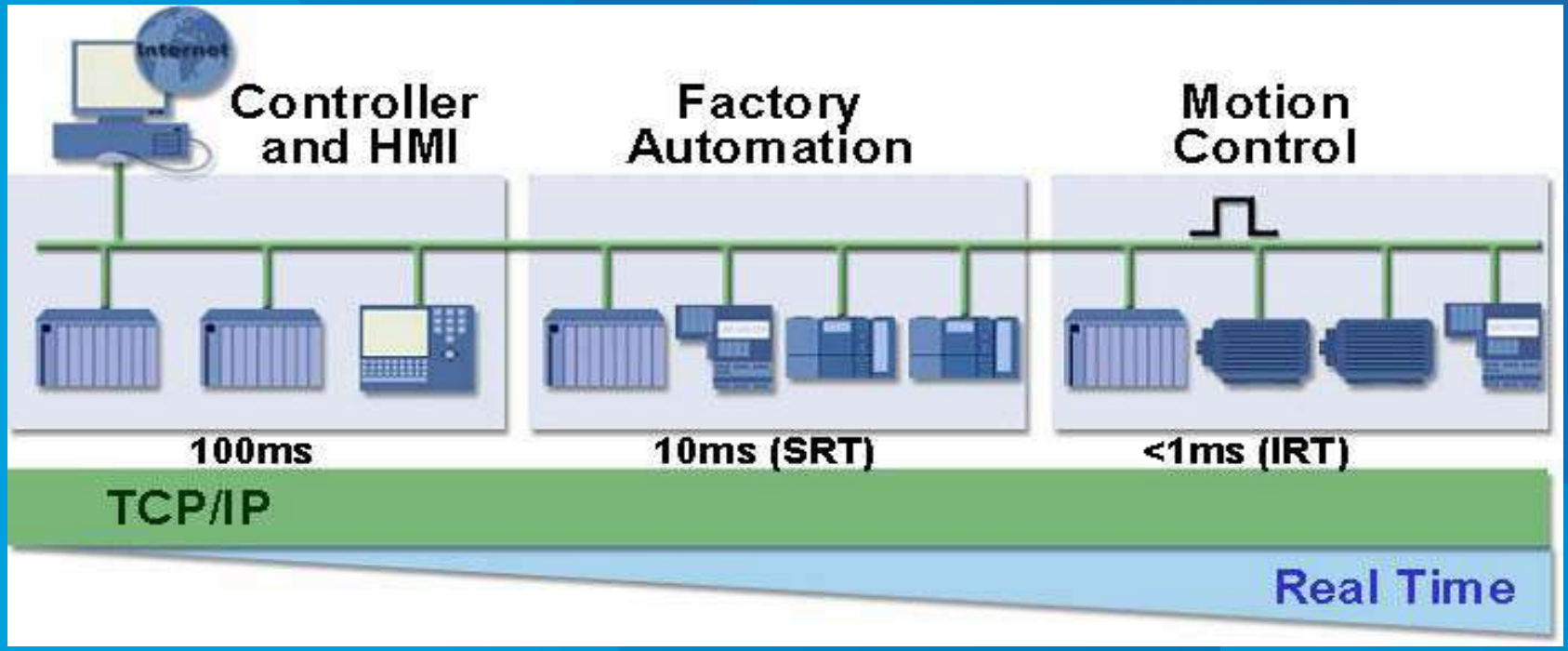
De la même façon que le protocole Modbus a été porté sur Ethernet avec Modbus-TCP, le protocole Profinet reprend les principaux services de Profibus, sur ethernet.



Cependant, à la différence de Modbus-TCP, la contrainte « temps réel fort » a été transposée à Profinet, ce qui implique un portage plus élaboré qu'une simple encapsulation des trames.

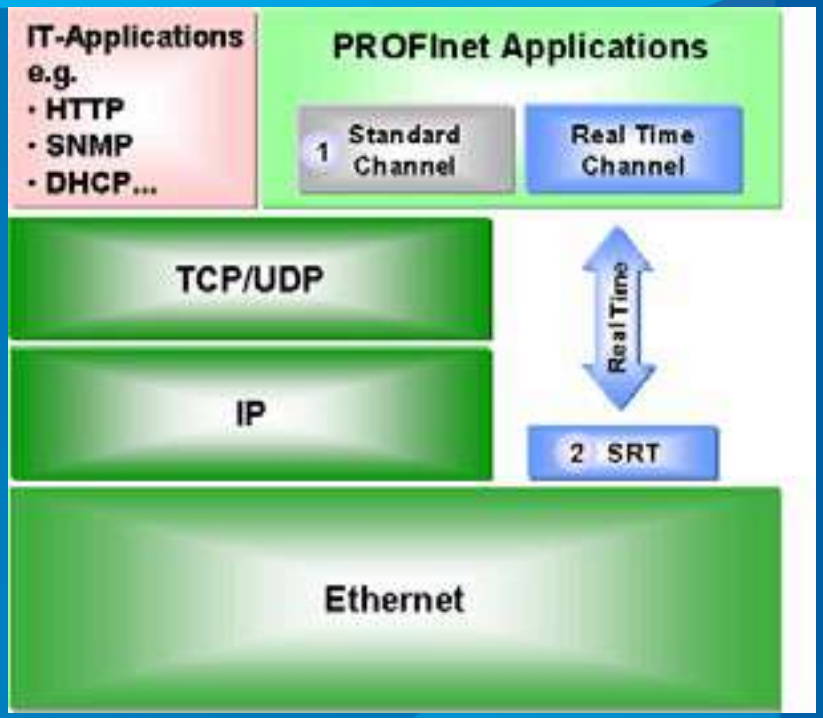
Profinet définit 3 niveaux de temps réel :

- non déterministe
- temps réel « soft » (SRT)
- temps réel isochrone (IRT)



Temps réel logiciel SRT :

Pour satisfaire les contraintes temps réel de l'automatisation, PROFInet possède un canal de transmission optimisé, dénommé *Soft Real Time*.
Basé sur Ethernet (couche 2), il raccourcit considérablement le temps de traitement dans la pile de communication et accroît la vitesse **Ethernet** de rafraîchissement des données.

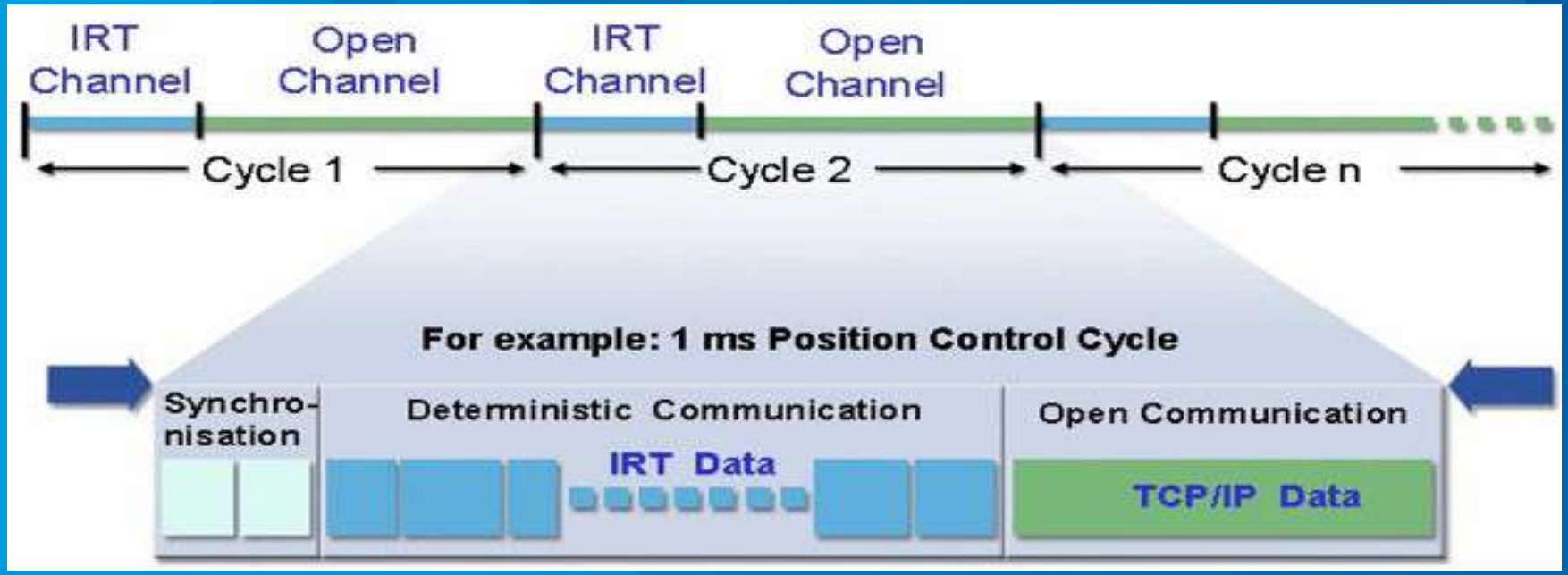


Tout d'abord, la *suppression de plusieurs niveaux de protocole* réduit la longueur du message; ensuite, la durée de préparation des données à la transmission et au traitement par l'application est écourtée.

Parallèlement, la puissance de calcul réservée dans l'appareil à la communication
est nettement allégée.

PROFInet ne se contente pas de minimiser la pile de communication des automatismes programmables :
il optimise aussi la transmission en attribuant à chaque paquet de données PROFInet une priorité

Les échanges entre appareils sont ensuite contrôlés par les constituants du réseau, en fonction de ces priorités : la priorité 6, accordée d'office aux données temps réel, l'emporte sur le traitement des applications .



Temps réel matériel IRT :

La solution IRT ne suffit pas aux applications de *positionnement* et de *synchronisme* du *Motion Control*.
Celles-ci exigent des temps de rafraîchissement de l'ordre de 1 ms avec une incertitude sur les tops de synchronisation (*jitter*) entre deux cycles consécutifs de 1 µs, pour synchroniser un maximum de 100 noeuds.

Pour satisfaire ces contraintes déterministes, PROFInet a défini, au niveau de la couche 2 , *une méthode de transmission contrôlée par tranche de temps*.

Grâce à la *synchronisation d'horloge des participants* du bus, il est possible de réserver sur le réseau une *tranche pour la transmission des données critiques* de la tâche d'automatisation.

Le cycle de transmission est donc segmenté en parties «déterministe» et «non déterministe» : les télégrammes cycliques temps réel sollicitent la tranche déterministe tandis que les télégrammes TCP/IP occupent la plage non déterministe.

Cette transmission «isochrone» est matérielle : un circuit ASIC se charge de la synchronisation du cycle et de la réservation du canal temporel pour les données temps réel. Cela implique l'utilisation de *cartes réseau spécifiques* pour connecter un PC à un réseau profinet.

5 / CANopen :

Control Area Network

a / Présentation

b / Couche Physique

c / Couche liaison

d / Couche applicative

a – Présentation :

CAN signifie « Control Area Network », il s'agit d'un bus de terrain développé au départ par Bosch pour l'industrie automobile.

C'est un standard ouvert qui possède des caractéristiques intéressantes au niveau ***immunité*** et surtout ***accès au médium*** permettant une utilisation de la bande passante plus rationnelle qu'un système “maître-esclave”.

Ce bus a depuis été adapté aux ***besoins des automatismes industriels***, à travers plusieurs implémentations très voisines dont les plus célèbres sont :

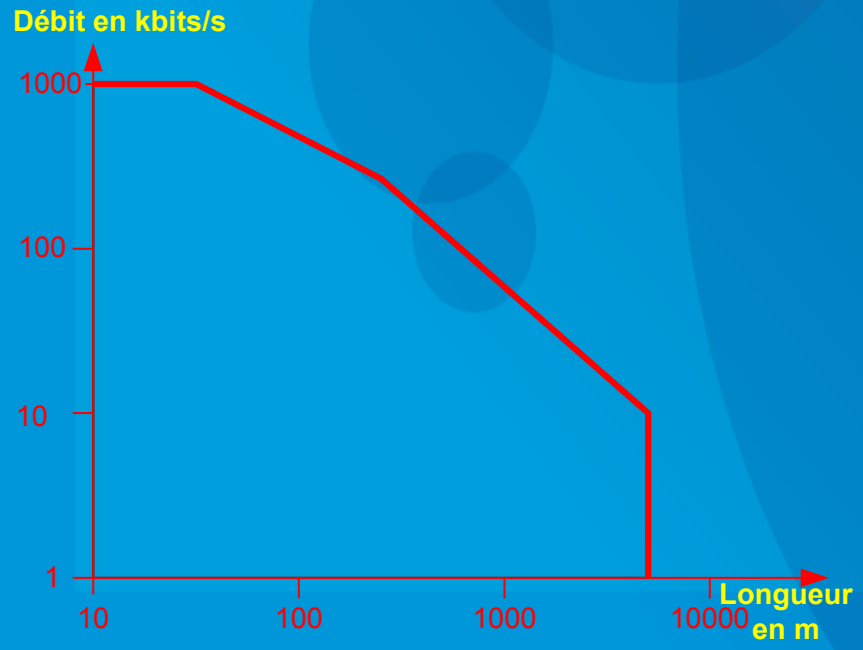
- ***DeviceNET***, porté notamment par Rockwell (U.S)

- ***CANopen***, supporté notamment par Schneider (Eur.)

➢ Critères de Comparaison :

➔ ***Débit / Longueur de bus :*** 3 débits sont possibles sur CANopen ; plus le débit est élevé, et plus la longueur de bus tolérée est faible :

Cable Type	125 kbps	250 kbps	500 kbps
Thick Round Cable	500 m (1,640 ft)	250 m (820 ft)	100 m (328 ft)
Thin Round Cable	100 m (328 ft)	100 m (328 ft)	100 m (328 ft)
Flat Cable	420 m (1,378 ft)	200 m (656 ft)	75 m (246 ft)
Maximum Drop Length	6 m (20 ft)	6 m (20 ft)	6 m (20 ft)
Cumulative Drop Length	156 m (512 ft)	78 m (256 ft)	39 m (128 ft)



➔ **Nombre de Nœuds :** jusqu'à 64

b – Couche Physique :

➢ **Connectique :** le bus DeviceNET utilise 4 fils :

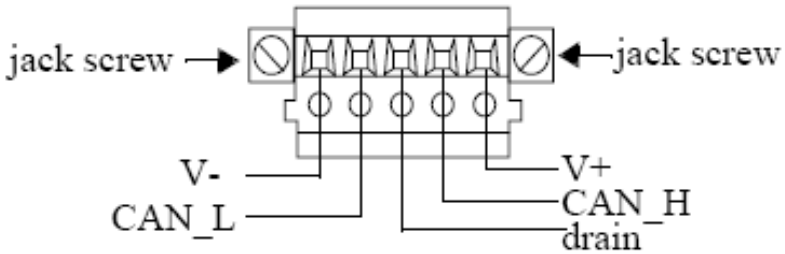
- Alimentation (V+ et V-)

- Données (CAN_L et CAN_H)

Wire Color	Wire Identity	Usage Round	Usage Flat
white	CAN_H	signal	signal
blue	CAN_L	signal	signal
bare	drain	shield	n/a
black	V-	power	power
red	V+	power	power

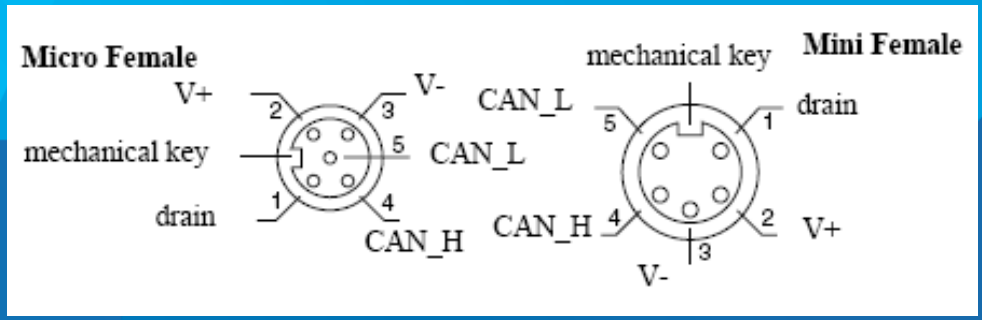
➢ **Connectique :** Les 3 types de connecteurs suivants sont utilisables :

Figure 3
Screw-terminal and
Hard-wired Connectors (IP20)



5-pin linear plug

Figure 4
Mini and Micro-style
Pluggable Connectors (IP67)



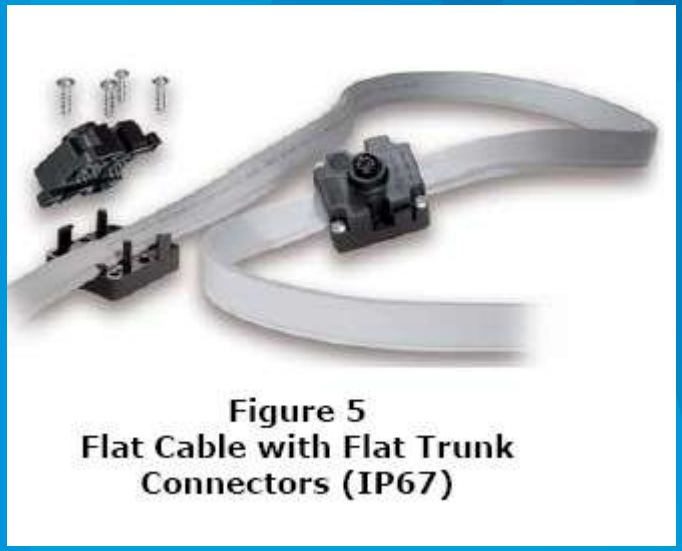
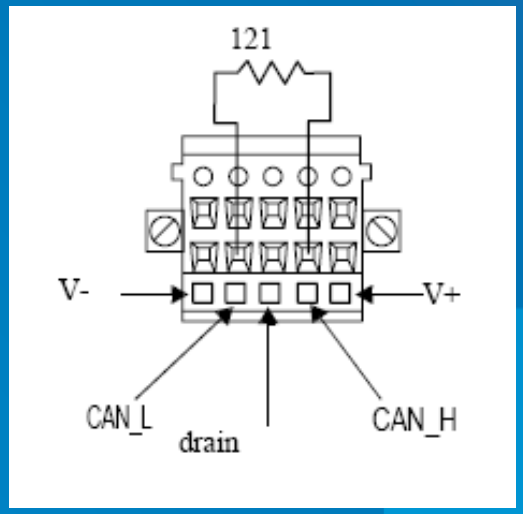
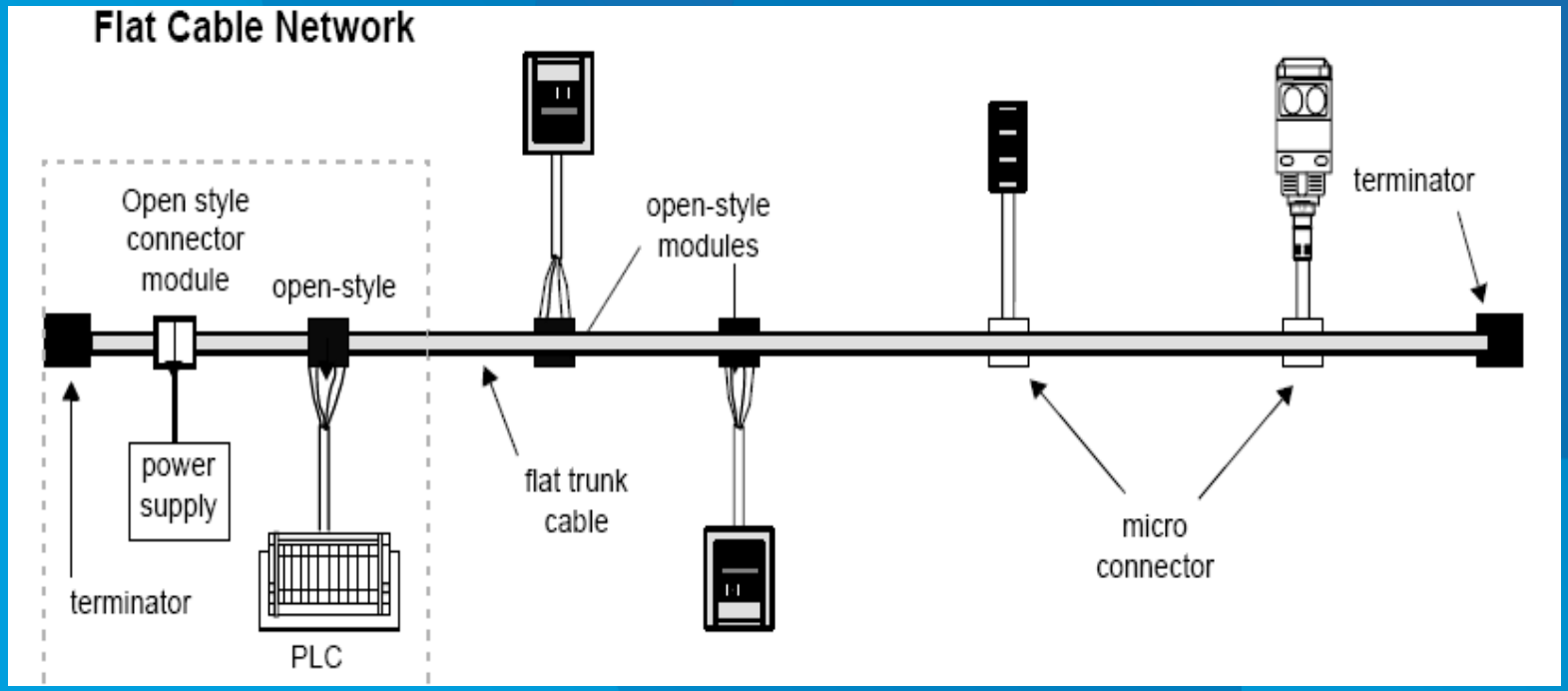


Figure 5
Flat Cable with Flat Trunk
Connectors (IP67)

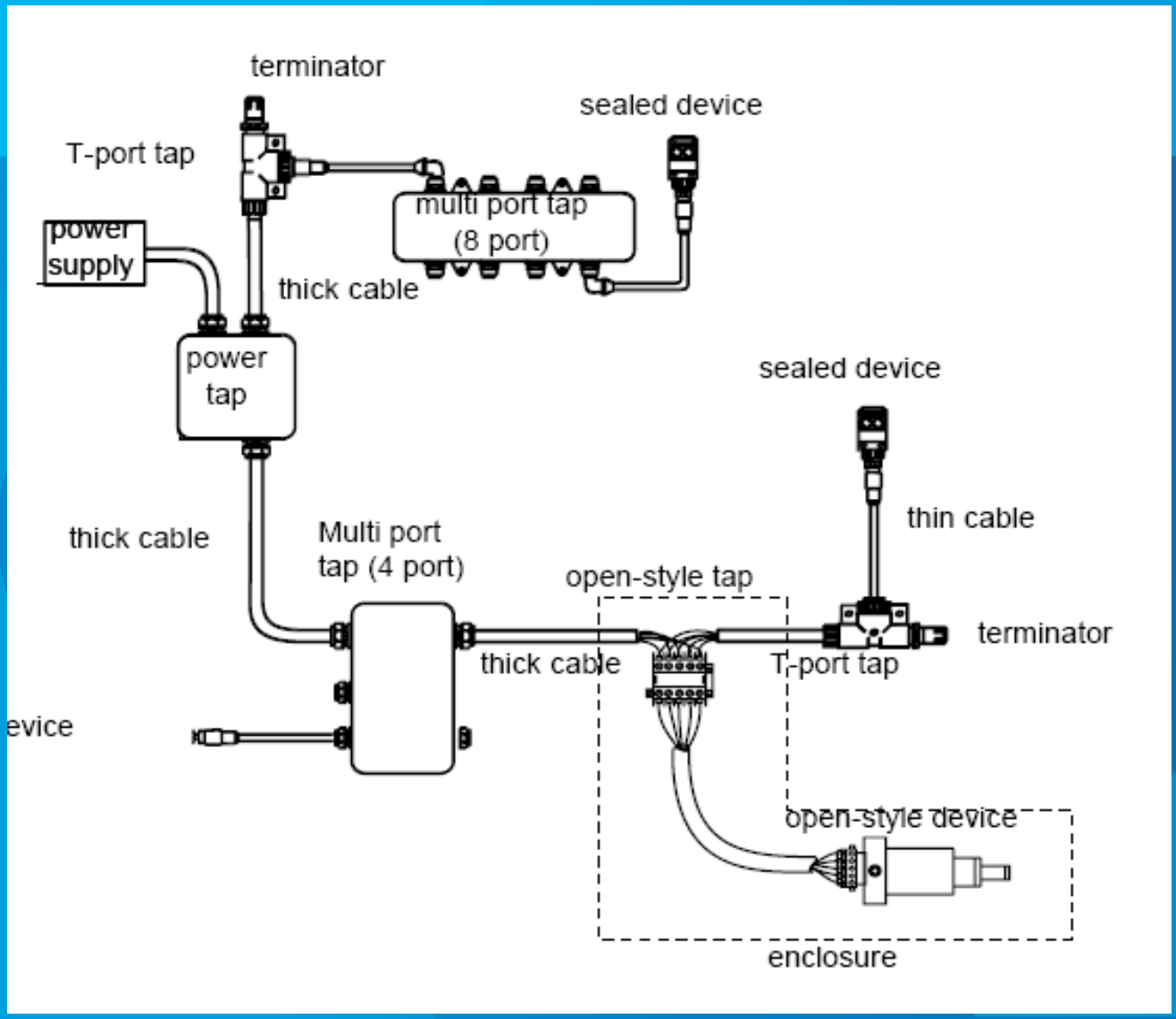


➢ **Topologie :** Toutes les topologies sont envisageables pour CANopen, attention cependant à respecter les longueurs de segments admissibles (cf norme), et à ne pas oublier les résistances de terminaison (120 Ω) :

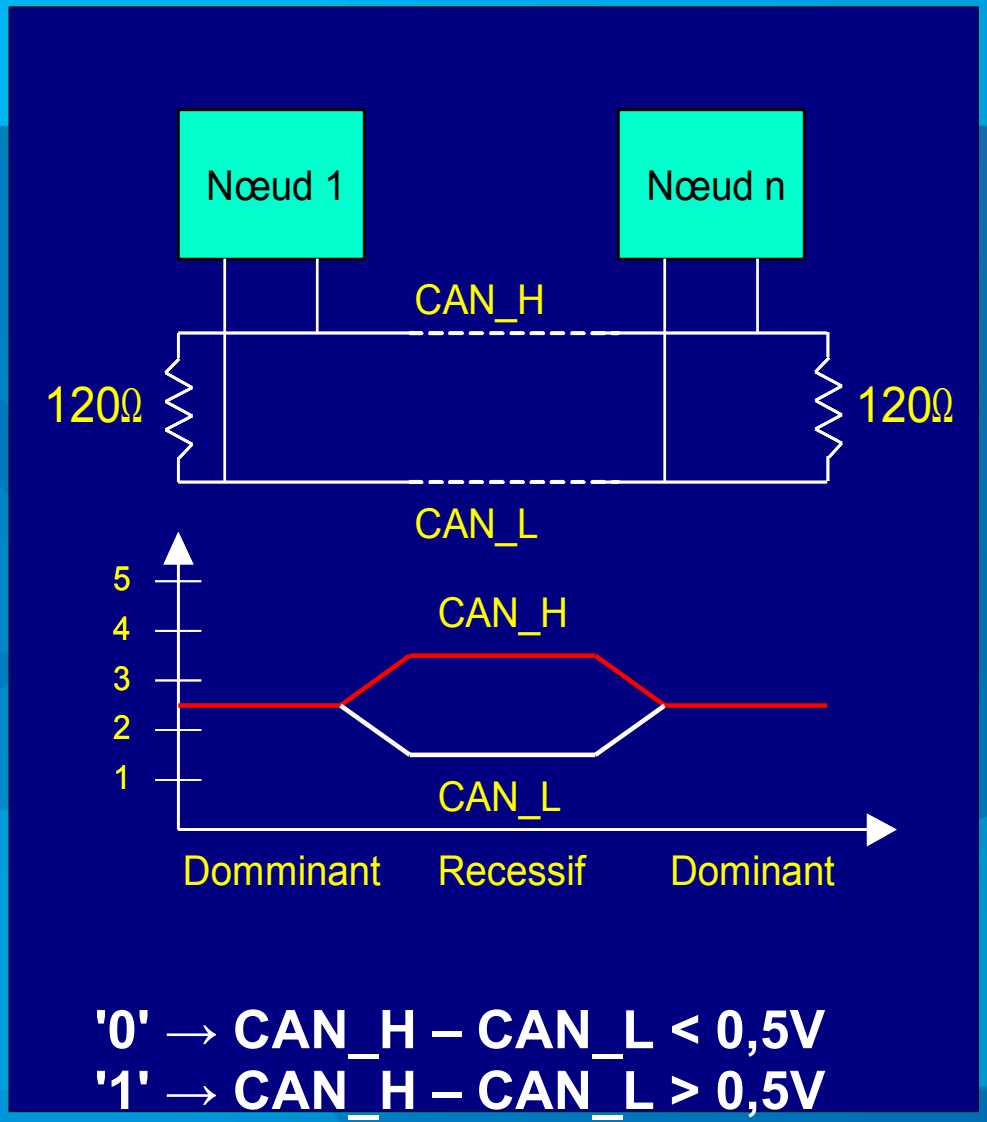
Topologie Bus sur câble plat :



Topologie Arborescente sur câble blindé :



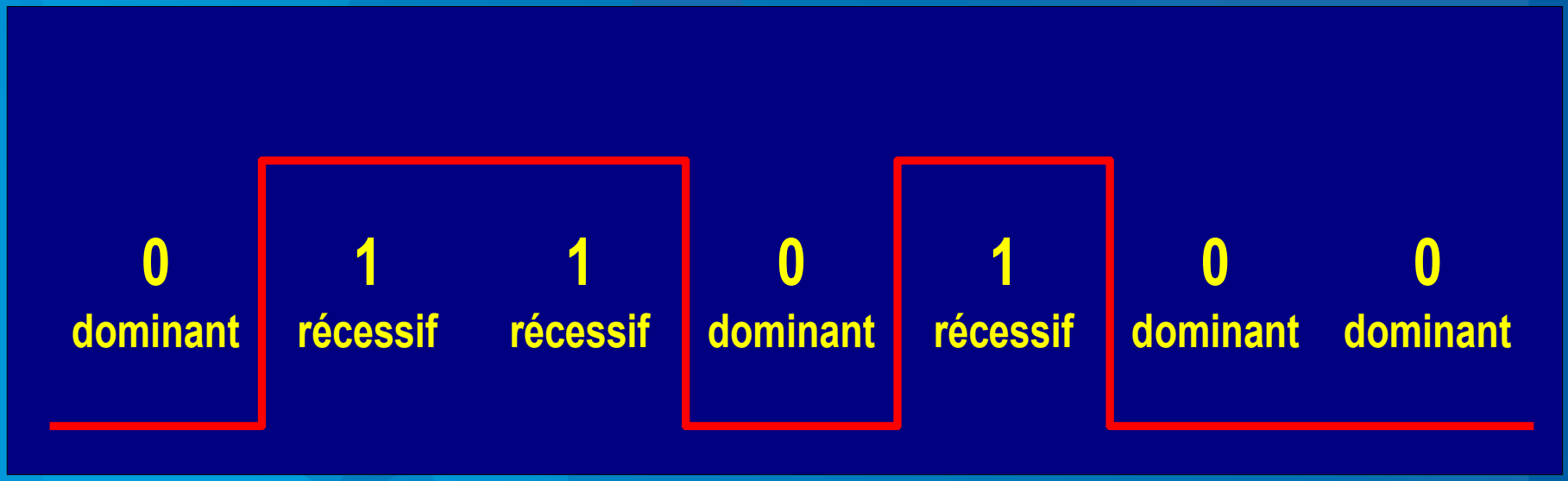
Codage : Le codage des bits est de type ***NRZ*** avec un système de bits ***dominants*** ('0') et ***récessifs*** ('1') :



'0' → CAN_H – CAN_L < 0,5V
'1' → CAN_H – CAN_L > 0,5V

Note : La transmission des données est différentielle.

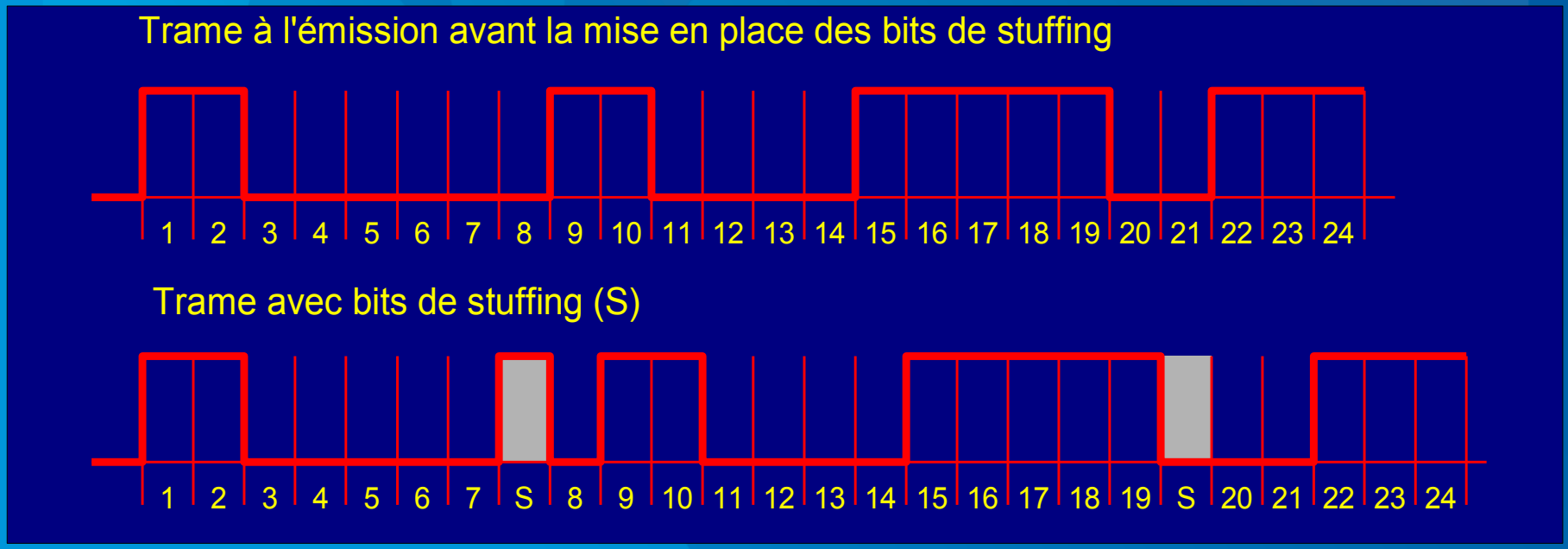
Cela signifie que si plusieurs nœuds ***émettent simultanément*** sur le bus, ce sont les ***états dominants*** qui sont ***effectivement appliqués*** sur le bus.



Cette particularité va être exploitée par la couche liaison de données pour ***éviter les collisions***.

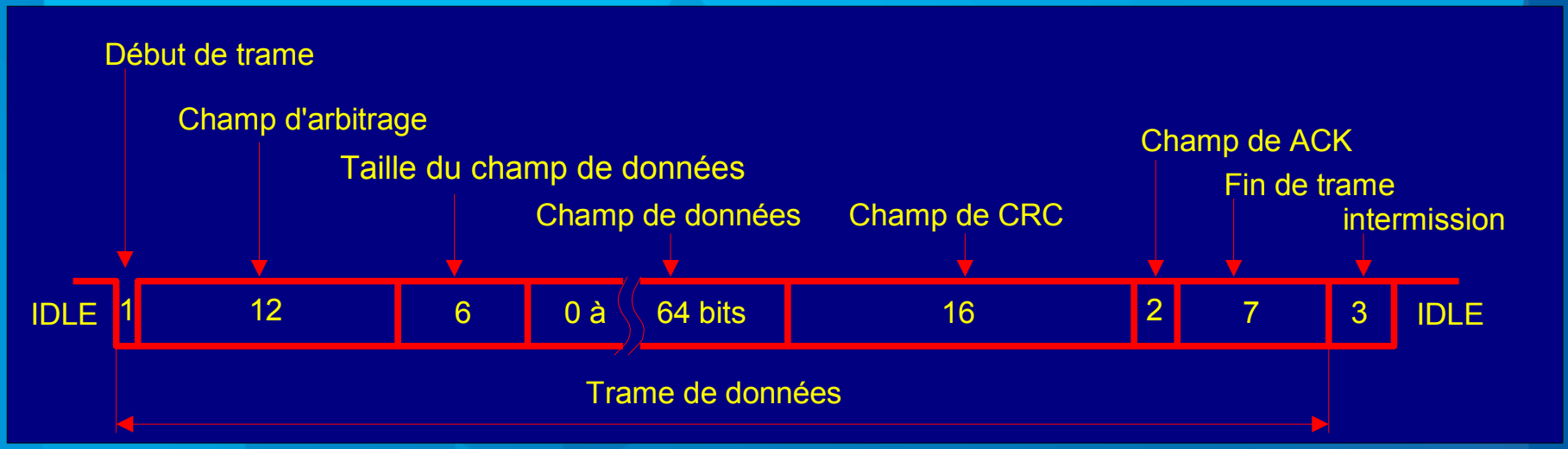
➢ **Bit Stuffing :** Le codage NRZ pose problème dans le cas d'une ***longue suite de bits identiques***.
En effet, contrairement au codage Manchester où l'on a 1 changement d'état par bit transmis, ce qui permet au nœud récepteur de se resynchroniser à chaque bit, ici, on peut facilement se ***désynchroniser***.

➔**Parade :** Le ***bit stuffing***. Cette technique consiste à introduire un bit d'état opposé si 5 bits identiques sont transmis. Evidemment, à la réception, il faudra distinguer les bits de stuffing des bits d'information.



c – Couche Liaison de données :

➢ **Format d'une trame de données :**

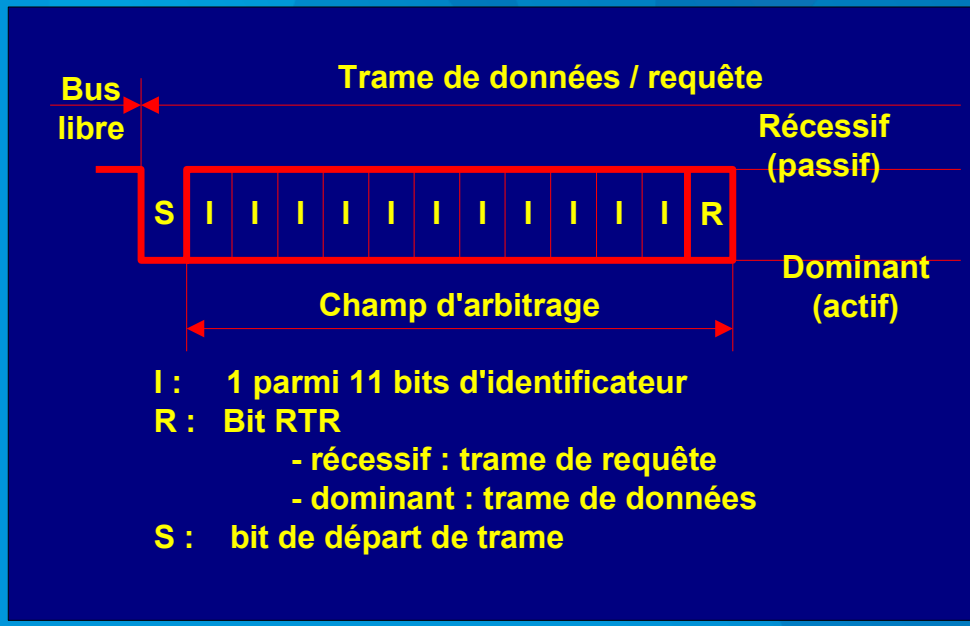


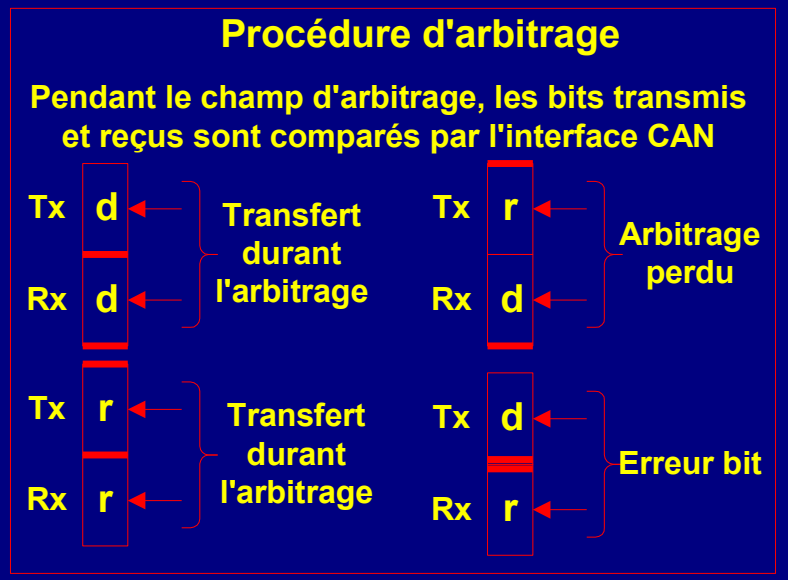
➢ **Principe du CSMA/CA :** La topologie du réseau CAN fait que par nature, les information émises par un nœud sont ***physiquement diffusées*** vers l'ensemble des autres nœuds

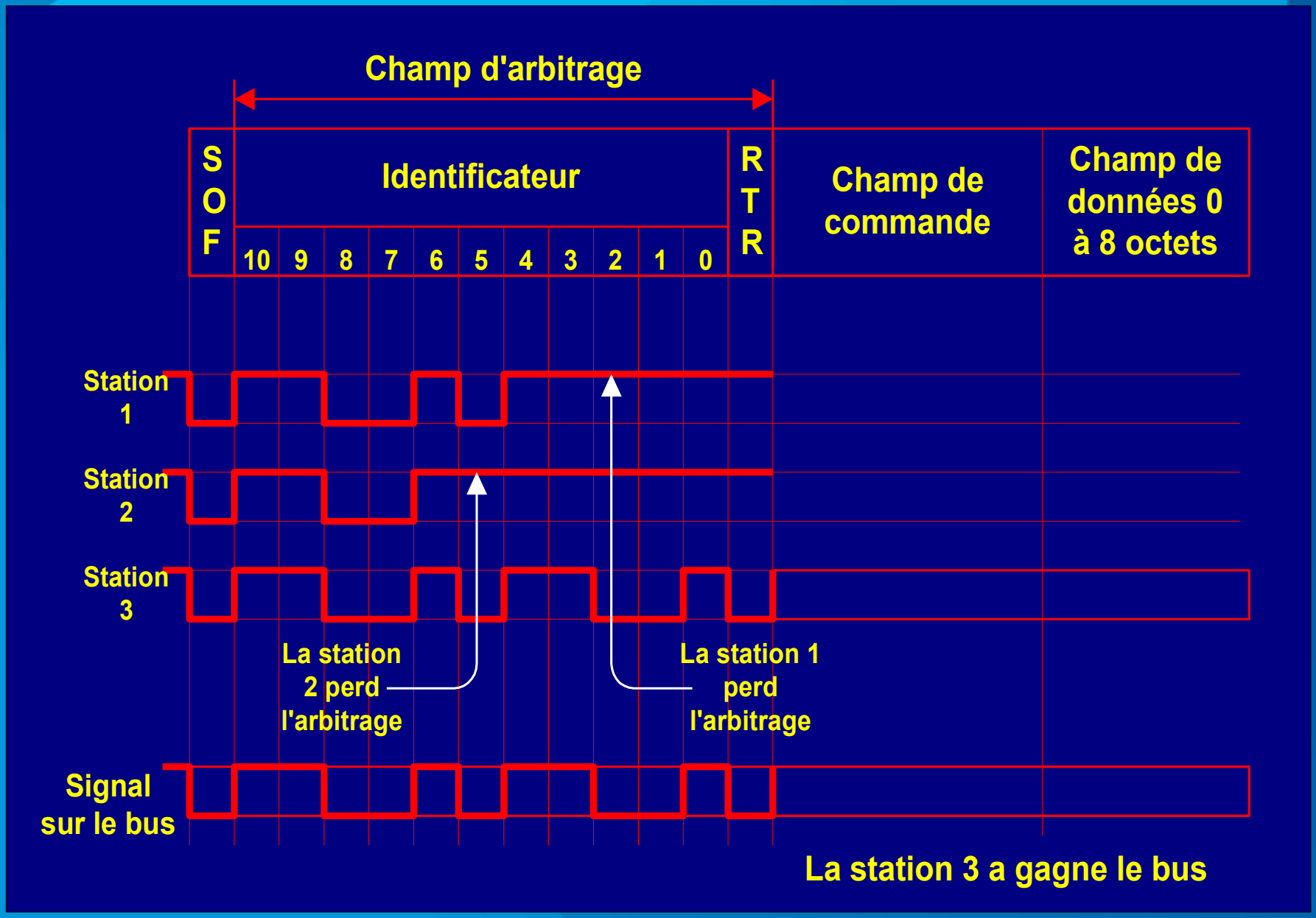
Or, CANopen n'est pas un réseau de type maître esclave, donc ***plusieurs nœuds*** peuvent prendre la parole ***simultanément***. Un mécanisme d'***arbitrage*** est donc nécessaire. Cette fonction va être remplie par une méthode d'accès au médium de type ***CSMA/CA*** : « ***Carrier Sense Multiple Access / Collision Avoided*** ».

➢ **Méthode d'arbitrage** : Les échanges sur le réseau ne peuvent débuter qu'à des ***instants précis*** matérialisés par le bit ***SOF*** qui débute la trame. La ***gestion du temps*** est une fonction primordiale implantée dans les équipements CAN).

Le premier champ de la trame est un ***champ d'arbitrage*** dans lequel tous les nœuds qui souhaitent prendre la parole vont émettre un ***identificateur*** (unique sur le réseau).
Chacun va alors ***« écouter »*** sur le bus et ***comparer*** ce qu'il émet et ce qu'il écoute :






Champ d'arbitrage
S O F
Identificateur
10 9 8 7 6 5 4 3 2 1 0
R T R
Champ de commande
Champ de données 0 à 8 octets
Station 1
Station 2
Station 3
La station 2 perd l'arbitrage
La station 1 perd l'arbitrage
Signal sur le bus
La station 3 a gagne le bus

➢ Rôle des bits dans le champ d'arbitrage:

- ✔ *Le bit SOF ("Start Of Frame")* est dominant il signale à toutes les stations le début d'un échange. Cet échange ne peut démarrer que si le bus était précédemment au repos. *Toutes les stations doivent se synchroniser* sur le front montant de ce bit.

- ✔ *Identificateur :* La longueur de l'identificateur est de *11 bits*, les bits sont transmis MSB en tête. Les 7 bits les plus significatifs (de ID_10 à ID_4) ne doivent pas être tous récessifs.

  (donc les valeurs interdites sont : ID = 1111111XXXX soit $2^{11} - 2^4$ = 2032 combinaisons.)

  Il *détermine le niveau de priorité* du noeud qui émet.

- ✔ *Le bit RTR :* il permet de *distinguer les échanges implicites des échanges explicites* (trames de requêtes).

➢ Groupes de champs d'arbitrage :

Les identifiants de connexion sont répartis en 4 groupes, en fonction des niveaux de priorité des messages. Un identifiant peut comporter :

- ✔ ***Un « message ID »*** représentant le niveau de priorité du message porté sur le bus;
- ✔ ***Un « MAC ID »*** indiquant le numéro du noeud initiant la connexion.

Connection ID = CAN Identifier (bits 10:0)											Used for
10	9	8	7	6	5	4	3	2	1	0	
0	Message ID				Source MAC ID						Message Group 1
1	0	MAC ID						Message ID			Message Group 2
1	1	Message ID			Source MAC ID						Message Group 3
1	1	1	1	1	Message ID						Message Group 4
1	1	1	1	1	1	1	x	x	x	x	Invalid CAN Identifiers

➢ Fragmentation des trames :

CANopen prévoit une fragmentation des trames lorsque le champ de données dépasse les 8 octets.

Le premier octet du champ de données vaut 0 dans le cas d'une trame non fragmentée, et indique le numéro du paquet dans le cas contraire.

c – Couche Applicative :

➢ Les différents types d'objets échangés :

✔ *Objets données process (PDO) :*

Les objets données process (PDO) sont utilisés pour leur ***rapidité de transmission*** des données process.

Un PDO peut transporter des données utiles jusqu'à 8 octets, ce qui est le maximum pour une trame CAN non fragmentée

La transmission d'un PDO utilise le modèle "***producteur-consommateur***" de CAN étendu par ***transferts synchronisés***.
Le transfert synchronisé des PDOs s'appuie sur le transfert des messages SYNC sur le bus CAN. Un PDO est envoyé en ***mode cyclique*** après un nombre configurable (de 1 à 240) de messages SYNC reçus.

Il est aussi possible d'attendre la disponibilité des variables du processus d'application et d'envoyer un PDO après la prochaine réception d'un message SYNC.
Ceci est appelé le ***transfert synchronisé acyclique***.

✔ Objets gestion réseau (NMT) :

Les objets gestion réseau (NMT) changent les états, ou contrôlent les états d'un équipement Can Open.
Un message NMT est un message avec l'identifiant CAN '0', ce qui fournit aux messages NMT le plus haut niveau de priorité.

Le message NMT consiste toujours en 2 octets de données utiles dans la trame CAN :

- Le premier octet contient la ***commande NMT encodée***;
- Le second octet contient l'***ID du noeud adressé***.

Un équipement CANopen démarre dans l'état « initialisation » une fois le bouton de mise en service appuyé. Lorsque l'équipement a terminé son initialisation, il fournit un objet NMP de ***démarrage*** afin de prévenir le maître.

Le protocole de détection de collision pour la surveillance de l'état de l'équipement est implémenté avec les objets NMT.

✔ Objets fonctions spéciales (SYNC, EMCY, TIME) :

- Can Open doit avoir un ***producteur SYNC*** pour synchroniser les actions des noeuds Can Open. Un producteur SYNC émet périodiquement l'objet SYNC. L'objet SYNC possède l'identifiant 128. Ceci peut entraîner l'apparition d'un retard due à la priorité de ce message.

- L'erreur interne d'un équipement peut déclencher un ***objet urgence (EMCY)***. La réaction des clients EMCY dépend des applications. Le standard Can Open définit plusieurs codes d'urgence. L'objet urgence est transmis dans une trame CAN unique de 8 octets.

- Une trame CAN avec l'ID CAN 256 et 6 octets de données utiles peut être utilisée pour transmettre l'heure du jour à plusieurs noeuds Can Open.
Cet ***objet temps (TIME)*** contient la valeur de la date et de l'heure dans l'objet de type Time-Of-Day.

✔ ***Mécanismes de surveillance (Watchdog)*** *:*

Can Open possède 2 méthodes de surveillance de l'état des équipements :

- Un ***gestionnaire réseau*** peut scruter régulièrement chaque équipement à des intervalles de temps configurables.

Cette méthode est appelée "***Node guarding***". Cette technique est cependant consommatrice de bande Passante.

- Un autre mécanisme est l'envoi régulier d'un message de la part de chaque équipement.

Cette méthode permet d'économiser la bande passante par rapport à la méthode dite "Node guarding".

d – Exemple de matériel CANopen :

- ***PC industriel Wago :***

758-876

WAGO-I/O-IPC-P14 Linux 2.6

High-performance industrial PC with Pentium®M 1,4 GHz

PC pour environnement industriel, OS Linux embarqué.

Maître CANopen intégré avec connectique.

• Codeur de Position (absolu) CANopen :

Codeur antidéflagrants homologué ATEX
Directive 94/9/CE

Codeur à enveloppe antidéflagrante pour atmosphères explosibles, environnement gaz et poussières.
Conception robuste pour application à fortes sollicitations
Axe traversant jusqu'à 30mm.
Domaines d'application : atmosphères explosibles sauf mines grisouteuses

Paramètres programmables

Résolution: définit le nombre de point par tour (0 à 8 192)
Résolution globale (Total Measuring Range) : définit le nombre de codes total du codeur (2 à 536 870 912)
Vitesse de transmission : configurable de 10kBaud (1 000) à 1 Mbaud (40 m) ; valeur par défaut : 20 Kbaud
Adresse: définit la position logicielle du codeur sur le bus (1 à 127, valeur par défaut : id = 1)
Sens : Permet de définir le sens de comptage du codeur
2 butées programmables: une butée haute et une butée basse
RAX : définit la valeur de sa position actuelle (axe immobile)

Les modes de communication

L'interrogation du codeur peut se faire suivant 3 modes :
Mode POOLING: le codeur répond aux demandes du maître. Ce mode permet de programmer et d'interroger les paramètres du codeur et sa position
Mode CYCLIQUE: le codeur transmet sa position de manière asynchrone. La fréquence d'émission est définie par le registre Cyclique Timer programmable de 0 à 65 535 ms
Mode SYNCHRO: Le codeur transmet sa position sur une demande de manière synchro au maître

CONNECTIQUE CANOPEN

Bleu	Noir	Blanc	Rouge
CAN LOW	0V	CAN HIGH	11/30Vdc

✔ *Exercice :*

Soient 3 noeuds connectés à un réseau CANOpen possédant les identifiants 2, 3 et 5.

Ces 3 noeuds “prennent la parole” simultanément sur le bus pour émettre un message de type “groupe 2”.

1 / Tracez en concordance des temps les bits émis par chacun des noeuds (indiquez si ces bits sont dominants 'D' ou récessifs 'R').

2 / Quel noeud gagne l'arbitrage ?

3/ Tracez, en concordance avec le chronogramme précédent, les tensions appliquées sur les 2 lignes du bus V_{CAN_L} et V_{CAN_H}.