BREVET DE TECHNICIEN SUPÉRIEUR
INFORMATIQUE ET RÉSEAUX
POUR L'INDUSTRIE ET LES SERVICES TECHNIQUES

ÉTUDE D'UN SYSTÈME INFORMATISÉ

#

Session 2014

_____

DUREE : 6 HEURES

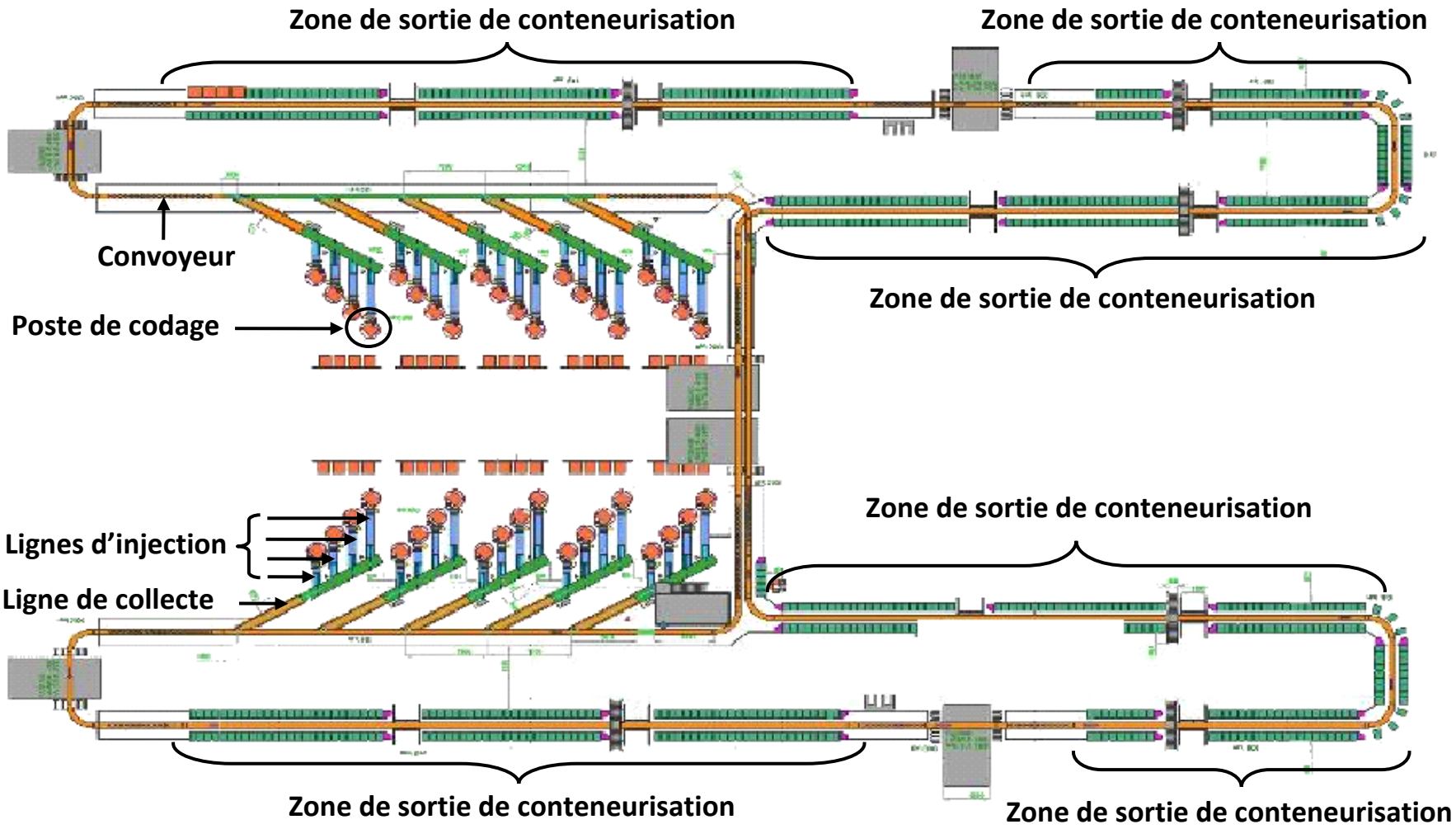
Coefficient 5

_____

ANNEXES

(19 PAGES)

Annexe 1 : Plan du « système de tri »



Annexe 2 : Code EAN 13

1 Présentation

Les codes EAN 13 (European Article Numbering à 13 chiffres) sont les codes à barres les plus utilisés dans le monde entier pour l'ensemble de produits de grande consommation. Ils comportent 13 chiffres dont la signification varie suivant le type du produit.

2 Composition d'un code-barres

Le code EAN 13 est constitué d'une suite de 13 caractères. Chaque caractère étant un chiffre compris entre 0 et 9.

Pour être traité par ordinateur (accès aux bases de données…), le code EAN doit être numérisé. Il est alors décomposé en une suite de bits (0 ou 1).

Les bits sont représentés par des barres qui peuvent être lues par des lecteurs optiques. Les 1 sont représentés par des barres noires, les 0 par des barres blanches. Toutes les barres ont la même épaisseur.

Le code barre est constitué (de gauche à droite) par :

- Le premier caractère (chiffre) du préfixe qui n'est pas codé sur les barres.
- Le Début, codé 101 : il permet d'identifier les limites du code-barres, et de donner une référence pour les largeurs des barres.
- Le second caractère du Préfixe + les cinq premiers caractères du Numéro d'Article.
- Le Séparateur Central, codé 01010.
- Les cinq caractères suivants du Numéro d'Article.
- La clé de contrôle.
- Le caractère de Fin, codé 101 (même fonction que Début et Séparateur central)

Chacun des 12 caractères est codé par 7 bits, ce qui fait un total de 95 barres noires ou blanches par article (3 + 6 x 7 + 5 + 6 x 7 + 3 = 95).

3 Codage d'un caractère

Chaque caractère peut prendre 10 valeurs différentes (entre 0 et 9).

- Les caractères situés à gauche du Séparateur Central du symbole EAN-13 utilisent deux jeux de codification nommés Set A et Set B. Ceux situés à droite utilisent le jeu de codification nommé Set C (table de codage des caractères ci-dessous).
- Le premier caractère du Préfixe (qui n'est pas codé) détermine l'alternance des Set A et B à utiliser pour le codage des 6 caractères situés à gauche du séparateur central (table de parité ci-dessous).

Table des caractères

Caractère	Set A	Set B	Set C
	Partie gauche		Partie droite
0	0001101	0100111	1110010
1	0011001	0110011	1100110
2	0010011	0011011	1101100
3	0111101	0100001	1000010
4	0100011	0011101	1011100
5	0110001	0111001	1001110
6	0101111	0000101	1010000
7	0111011	0010001	1000100
8	0110111	0001001	1001000
9	0001011	0010111	1110100

Table de parité

1° caractère	Succession des jeux
0	AAAAAA
1	AABABB
2	AABBAB
3	AABBBA
4	ABAABB
5	ABBAAB
6	ABBBAA
7	ABABAB
8	ABABBA
9	ABBABA

Exemple : le code du paragraphe précédent

- Premier caractère = 7 donc le caractère suivant est codé avec le Set A, le suivant le Set B, puis le Set A et ainsi de suite ;
- Second caractère = 6 donc en Set A donne 0101111 soit un espace, un trait fin noir, un espace et 4 traits fins noirs accolés ;
- Troisième caractère = 1 soit en Set B 0110011 donc un espace, un double trait noir, un double espace et un double trait noir ;
- Huitième caractère = 6 en Set C car à droite du séparateur, cela donne 1010000 soit un trait noir, un espace, un trait noir et un quadruple espace.

4 Calcul de la clé de contrôle EAN 13

Prenons l'exemple du calcul de la clé de contrôle du code EAN 13 dont les 12 premiers chiffres sont 471-9-5120-0288-x (où x est la clé de contrôle que l'on cherche). Il faut établir le tableau suivant :

Chiffres du code à barres	4	7	1	9	5	1	2	0	0	2	8	8
Poids	1	3	1	3	1	3	1	3	1	3	1	3
Produit	4	21	1	27	5	3	2	0	0	6	8	24

1. Pour les poids, on alterne les valeurs 1 et 3.
2. On multiplie le poids par le chiffre du code à barres
3. On calcule ensuite la somme des résultats,
4. On calcule le reste de la division par 10 de la somme précédemment calculée :
   - Si le reste de la division est égal à 0, alors la clé est 0,
   - Sinon, on ôte à 10, le reste ainsi trouvé : Clé = 10 - Reste.

La somme vaut dans cet exemple 4+21+1+27+5+3+2+0+0+6+8+24 = 101, le reste de la division par 10 est 1 ; la clé vaut donc 10-1 = 9.

Le code EAN 13 complet est 4719-5120-0288-9.

Annexe 3 : Chaînes de caractères en C/C++

3-1 Classe "string"

La classe string fait partie de la bibliothèque STL. Elle nécessite l'inclusion du fichier string :

```
#include <string>
```

Les strings sont des chaînes de caractères avec une gestion de la mémoire et des méthodes de gestion intégrées.

Voici quelques méthodes de cette classe :

- `string(); // constructeur d'une chaine vide`
- `string(const char *s); //constructeur d'une chaine copie du char *.`
- `const char *c_str() const; // conversion du string vers un char*.`
- `size_t length() const; // renvoie la longueur de la chaine`
- `bool empty() const; // indique si la chaine est vide (Vrai = vide).`
- `char &operator[](size_t n); // retourne le caractère à la position n`
- `char &at(size_t n); // idem`

Exemples :

```
string s1 ;
string s2 = "Bonjour " ;
string s3 = "monsieur" ;
cout << s3.length() << endl ; // 8
s1= s2 + s3 ;
cout << s1.c_str() << endl ; // Bonjour monsieur
cout << s2[2] << endl ; // Affiche n
```

3-2 Fonction `sprintf`

La fonction `sprintf` copie des données dans une variable de la même façon que printf les affiche à l'écran.

Syntaxe :

```
#include <stdio.h>
int sprintf(char *str, const char *format, ...);
```

Exemple :

Ci-dessous un programme qui contient deux chaînes de caractères : `nom` et `prenom` et une variable entière qui contient un `age`. On va placer dans une autre chaîne de caractères une phrase disant : `nom prenom` a `age` ans :

```
#include <stdio.h>
int main()
{
      char chaine[256], prenom[]="Jacques", nom[]="Dupond";
      int  age = 30;
      sprintf(chaine, "%s %s a %d ans", prenom, nom, age);
      printf("Le contenu de la variable : %s\n", chaine);
      return 0;
}
```

Annexe 4 : Bus AS-i

Le bus AS-i (*Actuator Sensor Interface*) a été développé en 1993 par le consortium AS-i (*Siemens*, *Schneider*, *Festo*, *Eurotherm*,…).
Son objectif est de faciliter la connexion des éléments d'entrées (capteurs) et de sorties (actionneurs) de type tout ou rien (TOR) des systèmes automatisés.

1 Couche physique :

La topologie physique du réseau AS-i est de type «bus». La longueur maximale du câble d'un bus AS-i est de 100m.
Le câble du bus AS-i est composé d'une paire de fils de couleur bleue et marron gainés d'un isolant électrique jaune. Il fournit la tension alimentation U_{bus} = 30V à des stations esclaves tout en véhiculant des données numériques.
Les capteurs et les actionneurs sont reliés au bus via les stations esclaves. Celles-ci se connectent au bus par une prise vampire.
Chaque esclave peut contenir soit 4 entrées soit 4 sorties soit 2 entrées et 2 sorties.

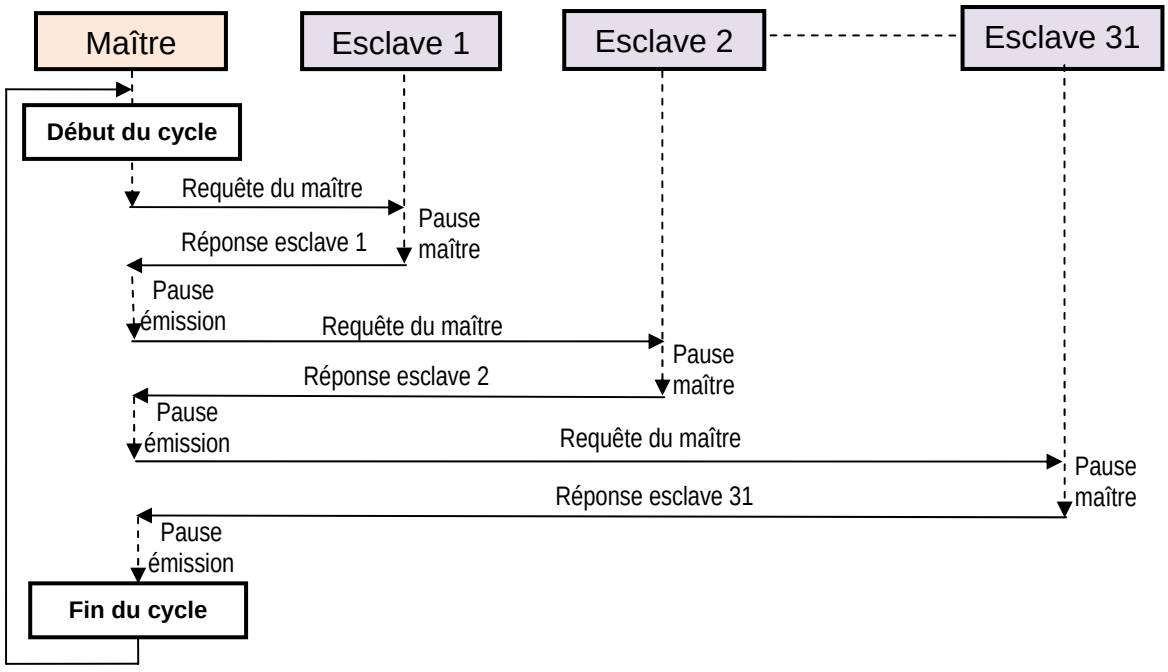
2 Couche liaison de données :

La méthode d'accès du bus AS-i est de type « `maître-esclave` ». Chaque bus est composé d'un seul maître et de 31 esclaves maximum. Le maître fournit à chaque esclave une adresse unique comprise entre 1 et 31.
Il existe deux types de maître :

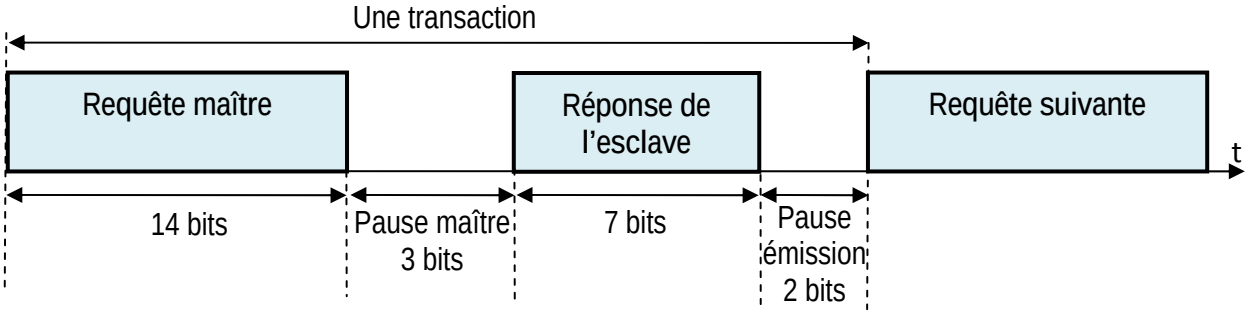
- Maître coupleur "automate". Ce type de maître est directement intégré à un automate programmable industriel ;
- Maître "passerelle". Ce type de maître est utilisé pour interfacer le bus AS-i avec un autre bus de terrain.

3 Protocole de communication AS-i :

Le maître interroge cycliquement, l'un après l'autre, chacune des 31 stations esclaves potentielles sur le bus. En un cyclique, le maître met à jour les sorties et fait acquisition de l'état des entrées de l'ensemble des esclaves. Le temps de cycle est garanti.



Le maître émet la requête (interroge l'esclave) et attend la réponse de l'esclave interrogé durant le temps dit « pause maître » de trois intervalles bits.
Après avoir reçu une réponse correcte, le maître respecte une pause dite « pause émission » ou « pause esclave » de durée de 2 temps de bits avant d'entamer une nouvelle transaction (interrogation d'une nouvelle esclave).
La durée d'un temps de bit est 6 µs.
Une transaction est composée de la requête du maître, le temps de « pause maître », la réponse de l'esclave et le temps de « pause émission ».

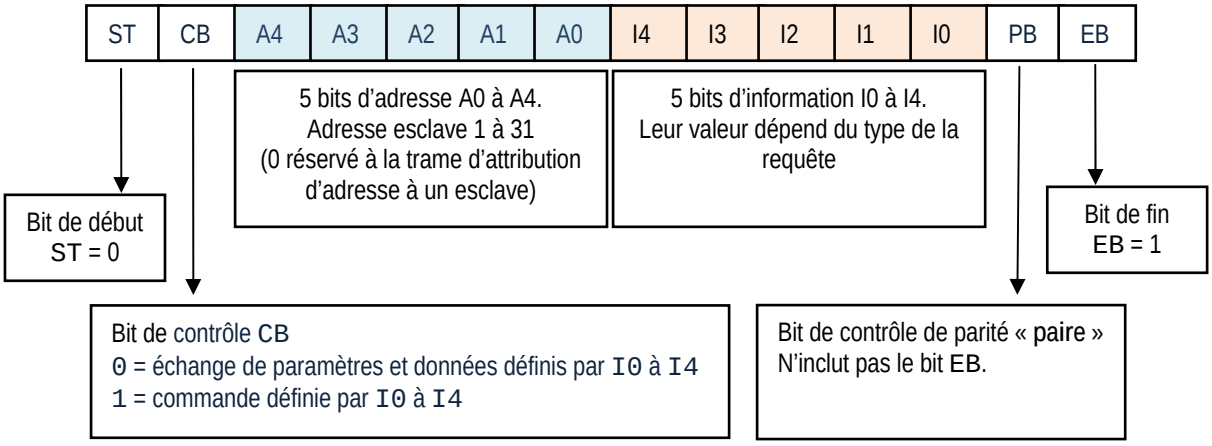


4 Structures des trames AS-i :

Une transaction AS-i est constituée par une trame de requête de la station maître et une trame de réponse de l'esclave.
Les trames émises par le maître et l'esclave commencent par l'émission du « bit de début ST » de niveau logique 0 et se terminent par le bit « fin de bit EB » de niveau logique 1.

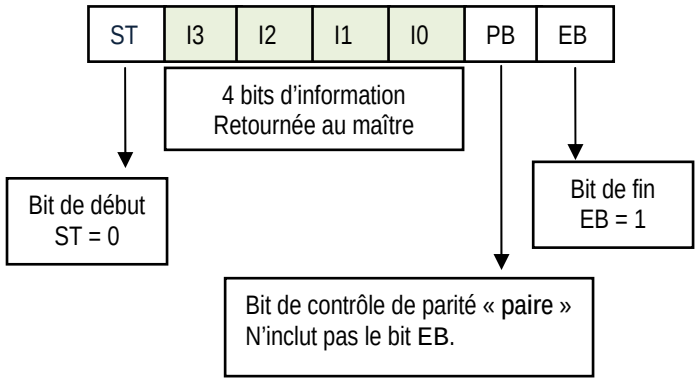
4-1 Structure des trames de requête :

Les trames de requête du maître contiennent toujours 14 bits et sont constituées de la façon suivante :



4-2 Structure des trames de réponse :

Les trames de réponse des stations esclaves contiennent toujours 7 bits et sont constituées de la façon suivante :



5 Trames pour réaliser les transactions :

Les différents types de trames de requête du maître et de réponse de l'esclave sont donnés dans le tableau ci-dessous.

Requêtes du maître	CB	5 bits d'adresse esclave				
Échanger des données	0	A4	A3	A2	A1	A0
Écrire des paramètres	0	A4	A3	A2	A1	A0
Attribuer l'adresse à un esclave	0	0	0	0	0	0
Mettre à zéro l'esclave	1	A4	A3	A2	A1	A0
Réinitialiser l'adresse d'un esclave	1	A4	A3	A2	A1	A0
Lire I/O configuré	1	A4	A3	A2	A1	A0
Lire code ID d'un esclave	1	A4	A3	A2	A1	A0
Lire le statut d'un esclave	1	A4	A3	A2	A1	A0
Lire et remettre à zéro le statut d'un esclave	1	A4	A3	A2	A1	A0

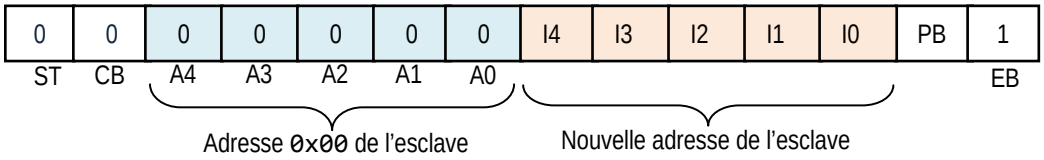
5-1 Trames pour réaliser une transaction de type « attribuer l'adresse à un esclave » :

À la première mise sous tension, l'adresse par défaut d'une station esclave est à `0x00`.

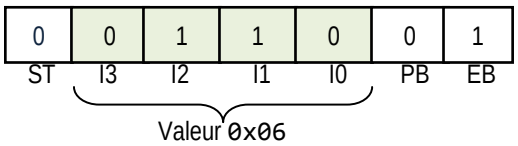
Le maître utilise l'adresse `0x00` pour attribuer une adresse à une station esclave nouvellement installée sur le bus.

Après avoir envoyé la trame d'acquittement, la station esclave peut recevoir de nouvelles requêtes du maître basées sur sa nouvelle adresse.

Trame de requête du maître :



Trame de réponse de l'esclave (acquittement) :



C'est la trame d'acquittement qui à pour valeur 6 quelle que soit l'adresse de l'esclave.

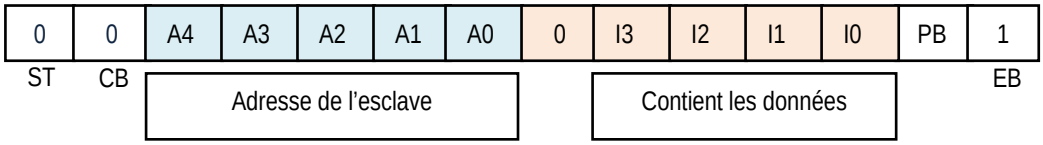
5-2 Trames pour réaliser une transaction de type « échanger des données » :

Le maître utilise deux types de transaction :

- Pour demander l'état des capteurs reliés à un nœud esclave où sont reliés les capteurs tout ou rien. Dans ce cas, les bits `I0` à `I3` de la trame réponse de l'esclave contiendra l'état des capteurs.
- Pour modifier l'état des actionneurs reliés à un esclave de sortie. Le maître indique l'état des actionneurs à l'aide des bits `I0` à `I3` de sa trame de requête.

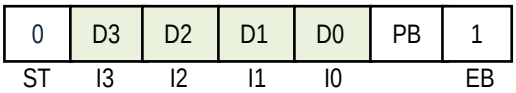
Trame de requête du maître :

La trame de requête pour transférer les données a la structure suivante :



Trame de réponse de l'esclave :

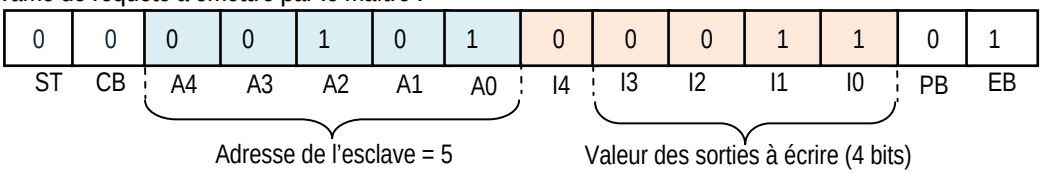
La trame de réponse de l'esclave doit avoir la structure suivante :



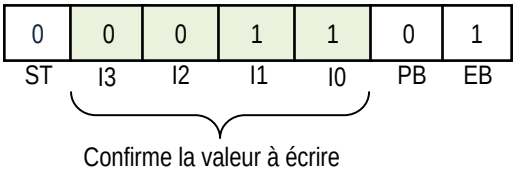
Exemple :

Exemple de requête du maître pour forcer les quatre sorties d'un nœud esclave contenant 4 sorties « Tout Ou Rien » d'adresse 5 :

Trame de requête à émettre par le maître :



Trame de réponse émise par l'esclave 5 :



6 Caractéristiques des signaux électriques véhiculées dans le support physique :

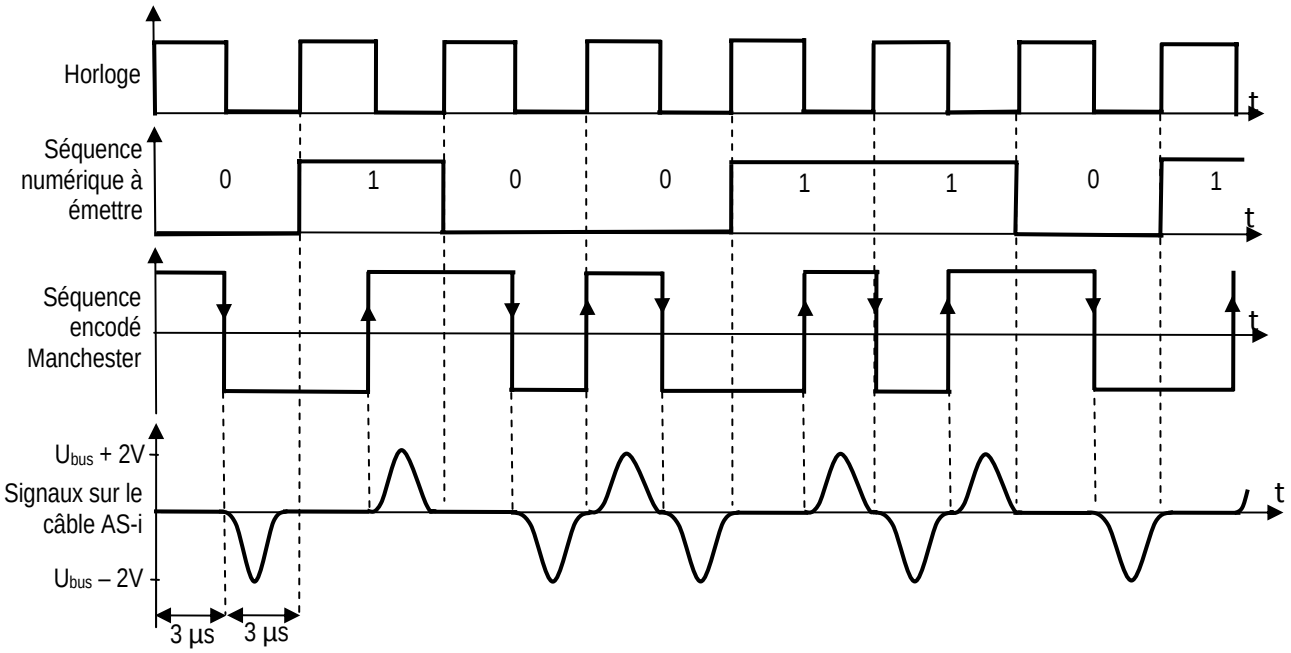
Le bus AS-i établit la communication entre la station maître et les stations esclaves. C'est une communication de données numérique en série et en multipoint.

Les signaux numériques à véhiculer sur le support physique sont codés en code «*Manchester*» puis modulés par des impulsions alternatives (APM).

- un «0» logique est codé en code Manchester par un intervalle de 3 µs au niveau haut suivi d'un intervalle de 3 µs au niveau bas ;
- un «1» logique est codé en code Manchester par un intervalle de 3 µs au niveau bas suivi d'un intervalle de 3 µs au niveau haut.

Puis les signaux du code Manchester sont transformés en impulsions de tension.

- un front descendant sur le signal du code Manchester se traduit par une impulsion de tension « négative » sur le câble de transmission ;
- un front montant sur le signal du code Manchester se traduit par une impulsion de tension « positive » sur le câble de transmission.



Annexe 5 : Passerelles Profibus-DP/AS-i

Extrait du catalogue Siemens :

Passerelles entre AS-Interface et PROFIBUS ou PROFINET selon spéc. 3.0 AS-i

DP/AS-i LINK Advanced	Désignation	Caractéristiques techniques	N° de référence
	1 maître AS-i	Passerelle compacte entre réseau PROFIBUS DP et AS-Interface, écran entièrement graphique et touches de commande, interface Web pour un diagnostic, une mise en service et des mises à jour du firmware interactives	**6GK1 415-2BA10**
	2 maîtres AS-i		**6GK1 415-2BA20**
IE/AS-i LINK PN IO			
	1 maître AS-i	Passerelle compacte entre réseau PROFINET et AS-Interface, écran entièrement graphique et touches de commande, interface Web pour un diagnostic, une mise en service et des mises à jour du firmware interactives	**6GK1 411-2AB10**
	2 maîtres AS-i		**6GK1 411-2AB20**

Extrait d'une offre de prix :

Poste	*Quantité*	*code article, référence , désignation*		*Prix HT en EUR*	*Unité*	*Délai*
1	1	6GK14152BA10 DP/AS-I Link Advanced mono maitre AS-I V3	SIEMENS-ABT	596.11/	U	10 jours
2	1	6GK14152BA20 DP/AS-I Link Advanced double maitre AS-I V3	SIEMENS-ABT	775.66/	U	10 jours
3	1	6GK14112AB10 IE/ASI Link PNIO, Passerelle PNIO-ASI mono maitre	SIEMENS-ABT	730.51/	U	2 SEM
4	1	6GK14112AB20 IE/ASI Link PNIO, Passerelle PNIO-ASI double maitre	SIEMENS-ABT	955.21/	U	2 SEM

Annexe 6 : Esclave AS-i

Product data sheet **3RK1200-0CQ20-0AA3**

AS-INTERFACE COMPACT MOD. K45, IP67,
DIGITAL, 4 INPUTS 4 X 1 INPUT,
200MA MAX., PNP 4 X M12 SOCKET

General technical data:		
Design of the product		digital I/O modules for operation in the field, IP67 - K45
Type		4 inputs
Design of the slave type		standard slave
I/O configuration		0
ID/ID2 code		0/F
Number / I/O sockets		4
Design of the electrical connection / of the inputs and outputs		M12 screw-type terminals
AS interface / total current input / max	mA	270
operating voltage		
• according to AS-Interface spezification	V	26.5 ... 31.6
Ground terminal		Using PIN5 on the M12 sockets. Outgoing via flat tab sleeve (2.8 x 0.8 mm form A)
Addressing		front addressing socket
AS interface / Connection / Polarity reversal protection		built-in
Delivery note		the modules are delivered without mounting plate
Note 1		All K45 compact modules are delivered with stainless steel screws/sockets

Annexe 7 : Requêtes « SQL »

S.Q.L. (Structured Query Language) est un langage normalisé de requêtes structurées et un standard d'accès aux bases de données relationnelles.

1 La requête SELECT

Elle permet :

- De sélectionner tous ou certains champs (ou colonnes) d'une ou plusieurs tables en fonction de critères ;
- D'extraire certaines occurrences ou tuples ou enregistrements et de les trier en fonction de critères ;
- D'utiliser des fonctions arithmétiques et de groupements pour des calculs.

Syntaxe générale de la requête SELECT :

```
SELECT [DISTINCT] Liste des champs séparés par une virgule
FROM Liste des tables concernées, séparées par une virgule
WHERE Liste des critères de choix ;
```

L'option DISTINCT permet de ne pas prendre en compte les doublons.

Exemples :

```
SELECT n_dep , nom_dep FROM DEPOSITAIRE ;
Extrait la liste des numéros et des noms des dépositaires de la table DEPOSITAIRE.
SELECT * FROM DEPOSITAIRE ;
Extrait tous les enregistrements et tous les champs de la table DEPOSITAIRE.
SELECT DISTINCT nom_dep FROM DEPOSITAIRE ;
Extrait la liste des noms des dépositaires sans doublons de la table DEPOSITAIRE.
```

La clause WHERE :

La clause WHERE permet de sélectionner dans la table les tuples correspondants aux critères précisés dans cette clause.

Les conditions sont une expression logique pouvant contenir :

- les champs ou colonnes des tables citées dans FROM ;
- les opérateurs de comparaison : >, <, =, >=, <= ;
- les opérateurs NOT, OR, AND ;
- les opérateurs d'ensemble BETWEEN, IS NULL, IS NOT NULL, LIKE, IN.

Exemple :

```
SELECT date FROM commande WHERE (numero > 25 AND numero < 50) ;
Extrait la date des commandes dont le numéro est compris entre 26 et 49.
```

La jointure ou sélection sur plusieurs tables permet d'utiliser les relations entre les tables :

```
SELECT nom_table1.nom_col1 , nom_table2.nom_col1 [, ….]
FROM nom_table1, nom_table2
WHERE nom_table1.nom_col = nom_table2.nom_col ;
```

Pour coupler deux tables ou plus, il faut d'abord préciser les tables concernées dans la clause FROM, ainsi que le ou les critère(s) qui permettront d'associer les lignes des différentes tables dans une clause WHERE pour former un résultat grâce à une clause SELECT.

Exemple :

```
SELECT * FROM LIVRAISON, EDITION
WHERE LIVRAISON.n_edit=EDITION.n_edit AND Lib_edit=''La Provence'';
Extrait toutes les livraisons de l'édition ''La Provence''.
```

2 La requête INSERT

INSERT permet d'ajouter un ou plusieurs enregistrements dans une table.

Syntaxe générale de la requête INSERT :

```
INSERT INTO nom_table [ (nom_col1 [, nom_col2,…])]
VALUES (constante1 [, constantes2,…]) ;
```

Exemple :

```
INSERT INTO DEPOSITAIRE (n_dep , nom_dep , adr_dep)
VALUES (68,'Quentin','Marseille') ;
Insère un nouvel enregistrement dans la table dépositaire avec les valeurs :
N_dep=68 ; nom_dep='Quentin' ; adr_dep='Marseille'
```

3 La requête UPDATE

Elle permet de mettre à jour les données d'un enregistrement.

Syntaxe générale de la requête UPDATE :

```
UPDATE nom_table SET nom_col1= constante1|NULL [, nom_col2= constante2|NULL, …]
WHERE conditions… ;
```

Exemple :

```
UPDATE DEPOSITAIRE SET adr_dep='Toulon' WHERE n_dep=68 ;
Met à jour l'adresse du dépositaire de numéro 68 avec la valeur 'Toulon'.
```

4 La requête DELETE

Elle permet de supprimer un enregistrement d'une table.

Syntaxe générale de la requête DELETE :

```
DELETE FROM nom_table WHERE conditions… ;
```

Exemples :

```
DELETE FROM LIVRAISON WHERE jour=7 ;
Supprime toutes les livraisons effectuées le jour 7.
DELETE FROM DEPOSITAIRE ;
Supprime tous les dépositaires de la base
```

Annexe 8 : Tableau HTML

Un tableau HTML est encadré par les balises `<table>` et `</table>`. Il comprend :

- Des balises `<tr> </tr>` (Table Row) qui permet d'ajouter une ligne dans le tableau.
- Des balises `<th> </th>` (Table Header) dans les balises tr pour ajouter des cellules d'entête.
- Des balises `<td> </td>` (Table Data) dans les balises tr pour ajouter des cellules de valeur.

Exemple :

```
<table>
   <tr>
      <th>Évènement</th>
      <th>Date</th>
   </tr>
   <tr>
      <td>Mariage</td>
      <td>22/6/2014</td>
   </tr>
   <tr>
      <td>Anniversaire</td>
      <td>3/7/2014</td>
   </tr>
</table>
```

Résultat :

Évènement	Date
Mariage	22/6/2014
Anniversaire	3/7/2014

Annexe 9 : Langage « PHP »

1 Les bases du langage

Le code source PHP se trouve sur le serveur qui l'interprète et l'exécute. Il ne transmet au navigateur que le code HTML obtenu.

1.1 Structure d'un script PHP

Le script PHP doit avoir comme extension .php et le code doit être compris entre `<?php et ?>`. Chaque instruction se termine par un point virgule « `;` » comme en C et les commentaires sont précédés par « `//` » ou compris entre « `/*` » et « `*/` »comme en C++. Les blocs d'instructions sont délimités par des accolades `{}`.

Pour afficher une ligne dans la page HTML, il faut utiliser la structure `echo`. Par exemple, l'instruction « `echo` "la variable `foo` est égal à `$foo`"; » permet d'afficher « la variable `foo` est égal à `bonjour` » si la variable `$foo` contient « `bonjour` ».

1.2 Les variables

Les variables peuvent être du type booléen, entier, chaîne de caractères … mais elles ne sont pas déclarées et commencent toutes par le caractère $. Le type est déterminé lors de l'exécution par l'interpréteur PHP. Par exemple, « `$foo="essai" ;` » crée une variable de type chaîne de caractères appelée `$foo`.

Les types PHP possibles sont :

- booléen TRUE FALSE (i.e. 0,"","0") ;
- entier ;
- flottant ;
- chaîne de caractères (entre " ou ') ;
- tableau.

1.3 Les structures conditionnelles

La structure si-sinon s'écrit comme suit :

```
if (condition)
{
  // code si condition vraie;
}
else
{
  // code si condition fausse;
}
```

`else if` permet d'enchaîner une série d'instructions et évite d'avoir à imbriquer des instructions `if`.

La structure switch s'écrit comme suit :

```
switch ($variable)
{
  case valeur_1 :
     // Premier bloc d'instructions...
     break;
  case valeur_2 :
     // Second bloc d'instructions...
     break;
  ...
  case valeur_N :
     // Nième bloc d'instructions...
     break;
  default :
     // bloc d'instructions par défaut...
     break;
}
```

1.4 Les structures itératives

Le langage PHP propose quatre types de boucles :

La boucle for permet d'exécuter une série d'instructions un nombre déterminé de fois :

```
for (initialisation; condition; incrémentation) {
  // bloc d'instructions...
}
```

La boucle while permet d'exécuter une série d'instructions tant que la condition est vérifiée :

```
while (condition) {
  // bloc d'instructions...
}
```

La boucle do while s'écrit comme suit :

```
do {
  // bloc d'instructions...
}
while (condition);
```

La boucle foreach est utilisée exclusivement pour les tableaux ou toutes autres structures tabulaires. Elle s'écrit comme suit :

```
foreach ($tableau as $valeur) {
  // bloc d'instructions...
}
```

1.5 Les tableaux

Le langage PHP dispose de deux types de tableaux : les tableaux indicés et associatifs.
Les tableaux indicés peuvent se créer avec la fonction `array`.

Exemple :

```
$prenoms = array ('Pierre', 'Michel', 'Nicole', 'Véronique', 'Albert');
echo $prenom[1] ; // affiche dans la page HTML « Michel »
```

Les tableaux associatifs fonctionnent sur le même principe, sauf qu'au lieu de numéroter les cases, on va les étiqueter en leur donnant à chacune un nom différent.

Exemple :

```
$coordonnees = array (
'prenom' => 'Pierre',
'nom' => 'Dupont',
'adresse' => '25 rue nouvelle',
'ville' => 'Paris');
$coordonnees['nom'] = 'Martin'; // le nom est maintenant Martin
echo $coordonnees['ville']; // affiche dans la page HTML « Paris »
```

2 Accès à la base de données MySQL

L'accès s'effectue en 5 étapes :

1. Se connecter ;
2. Sélectionner une base ;
3. Exécuter des requêtes ;
4. Exploiter les résultats et traiter les erreurs ;
5. Fermer la connexion.

2.1 Connexion à un serveur MySQL

`resource mysql_connect ( string server , string username , string password , bool new_link , int client_flags ) ;`

Cette fonction établit une connexion avec le serveur (MySQL), pour un compte utilisateur et un mot de passe secret. Elle renvoie l'identifiant de connexion qui sera utilisé ensuite pour dialoguer avec la base de données en cas de succès et FALSE sinon.

Exemple :

```
$con=mysql_connect("192.168.109.10","user1","gigngtrrl");
```

2.2 Sélection d'une Bdd

`bool mysql_select_db ( string database_name , resource link_identifier ) ;`

Cette fonction permet de sélectionner une base de données. Les requêtes s'appliqueront donc à cette base.

Exemple :

```
mysql_select_db("mabase",$con);
```

2.3 Exécution d'une requête

`resource mysql_query ( string query , resource link_identifier ) ;`

Cette fonction adresse une requête SQL au serveur MySQL. Le code doit contenir la requête SQL.

Pour les requêtes du type SELECT , SHOW , DESCRIBE ou EXPLAIN , mysql_query retournera une ressource en cas de succès, ou FALSE en cas d'erreur. Cette ressource sera à utiliser ultérieurement dans les opérations de consultation.

Pour les autres types de requêtes, UPDATE , DELETE , DROP , etc., mysql_query retourne TRUE en cas de succès ou FALSE en cas d'erreur.

2.4 Exploitation des résultats

`array mysql_fetch_row ( resource result );`

Cette fonction récupère une des lignes du résultat de la function `mysql-query` et positionne le curseur sur la ligne suivante.

La ligne est représentée sous forme d'un tableau (une liste de valeurs).

`array mysql_fetch_array ( resource result , int result_type );`

Cette fonction transforme la ligne courante en un tableau associatif (clé/valeur) à indices numériques ou nommés. L'accès aux données se fait grâce à `$Tableau[Indice]` où les indices correspondent aux champs utilisés dans la requête.

Pour récupérer toutes les lignes du résultat, il faut les lire une par une grâce à une des deux fonctions ci-dessus. Le premier appel retournera la première du résultat de la requête, le second appel, la seconde ligne et ainsi de suite.

Exemple :

```
$resut= mysql_query("SELECT NumAvion,Type,Nom FROM avion");
while ($ligne = mysql_fetch_row($resultat)) {
   echo "NumAvion : $ligne[0]<br>";
   echo "Type : $ligne[1]<br>";
   echo "Nom : $ligne[2]<br>";
}
```

2.5 Fermeture la connexion

`bool mysql_close([ressource $link_identifier);`

Cette fonction ferme la connexion au serveur MySQL associée à l'identifiant spécifié. Si `$link_identifier)` n'est pas spécifié, cette commande s'applique à la dernière connexion ouverte.

Exemple :

```
mysql_close($con);
```

Annexe 10 : Code ASCII

	Hexadécimale.	Caractère	Décimale	Hexadécimale	Caractère
0	0x00	NUL Null	64	0x40	@
1	0x01	SOH Start of Header)	65	0x41	A
2	0x02	STX Start Of Text	66	0x42	B
3	0x03	ETH End Of Text	67	0x43	C
4	0x04	EOT End Of Transmit	68	0x44	D
5	0x05	ENQ Enquiry	69	0x45	E
6	0x06	ACK Acknowledgment	70	0x46	F
7	0x07	BEL Bell	71	0x47	G
8	0x08	BS Backspace)	72	0x48	H
9	0x09	HT Horizontal Tab	73	0x49	I
10	0x0A	LF Line Feed	74	0x4A	J
11	0x0B	VT Vertical Tab	75	0x4B	K
12	0x0C	FF Form Feed	76	0x4C	L
13	0x0D	CR Carriage Return)	77	0x4D	M
14	0x0E	SO Shift Out	78	0x4E	N
15	0x0F	SI Shift In	79	0x4F	O
16	0x10	DLE Data Link Escape	80	0x50	P
17	0x11	DC1 Device Control 1	81	0x51	Q
18	0x12	DC2 Device Control 2	82	0x52	R
19	0x13	DC3 Device Control 3	83	0x53	S
20	0x14	DC4 Device control 4	84	0x54	T
21	0x15	NAK	85	0x55	U
22	0x16	SYN Synchronous Idle	86	0x56	V
23	0x17	ETB End Transm. Block	87	0x57	W
24	0x18	Cancel	88	0x58	X
25	0x19	EM End Of Medium	89	0x59	Y
26	0x1A	SUB Substitute	90	0x5A	Z
27	0x1B	ESC Escape	91	0x5B	[
28	0x1C	FS File Separator	92	0x5C	\
29	0x1D	GS Group Separator	93	0x5D	]
30	0x1E	RS Request to Send	94	0x5E	^
31	0x1F	US Unit Separator	95	0x5F	_
32	0x20	SP Space	96	0x60	`
33	0x21	!	97	0x61	a
34	0x22	" double quote	98	0x62	b
35	0x23	# number sign	99	0x63	c
36	0x24	$ dollar sign	100	0x64	d
37	0x25	% percent	101	0x65	e
38	0x26	& ampersand	102	0x66	f
39	0x27	' single quote	103	0x67	g
40	0x28	(	104	0x68	h
41	0x29	)	105	0x69	i
42	0x2A	* asterisk	106	0x6A	j
43	0x2B	+ plus	107	0x6B	k
44	0x2C	, comma	108	0x6C	l
45	0x2D	- minus	109	0x6D	m
46	0x2E	. dot	110	0x6E	n
47	0x2F	/	111	0x6F	o
48	0x30	0	112	0x70	p
49	0x31	1	113	0x71	q
50	0x32	2	114	0x72	r
51	0x33	3	115	0x73	s
52	0x34	4	116	0x74	t
53	0x35	5	117	0x75	u
54	0x36	6	118	0x76	v
55	0x37	7	119	0x77	w
56	0x38	8	120	0x78	x
57	0x39	9	121	0x79	y
58	0x3A	:	122	0x7A	z
59	0x3B	;	123	0x7B	{
60	0x3C	<	124	0x7C	|
61	0x3D	=	125	0x7D	}
62	0x3E	>	126	0x7E	~
63	0x3F	?	127	0x7F	DEL