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

E4 - ÉTUDE D'UN SYSTÈME INFORMATISÉ U4

SESSION 2015

———

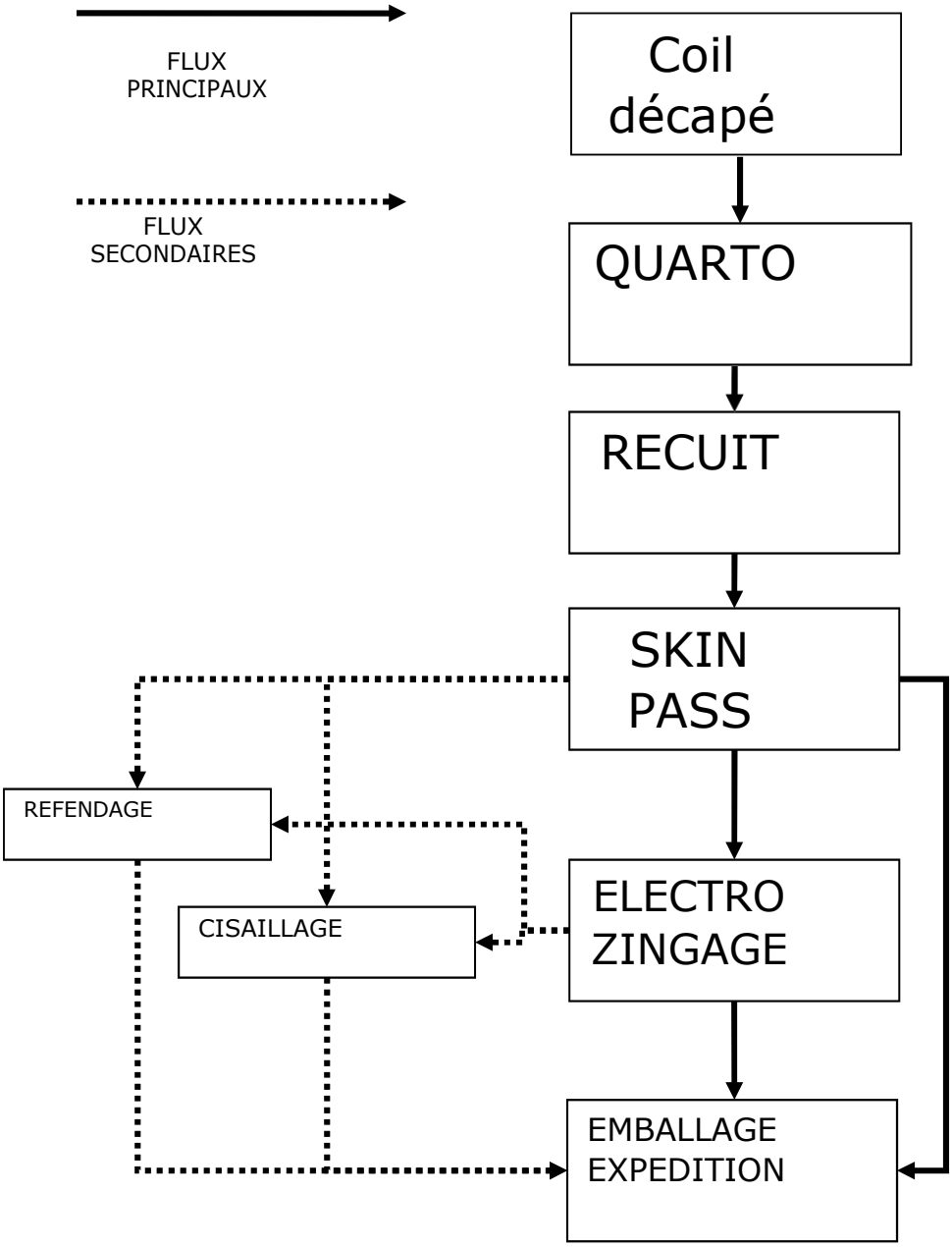
Durée : 6 heures
Coefficient 5

———

DOCUMENT ANNEXES

(12 PAGES)

Annexe 1 Flux de matière



Glossaire :

COÏL : bobine de métal enroulé

QUARTO : cage de laminage permettant de réduire l'épaisseur de la tôle

RECUIT : Traitement thermique permettant de redonner les caractéristiques de dureté au métal

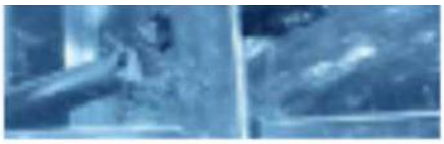
SKIN PASS : Traitement de surface

ELECTRO-ZINGAGE : Dépôt d'une pellicule de zinc sur la tôle

REFENDAGE : Découpe de la feuille de tôle d'une bobine dans le sens de la longueur (de la tôle déroulée) pour obtenir en bout de chaîne deux ou plusieurs bobines de largeur plus faible.

CISAILLAGE : Découpe de la feuille de tôle dans le sens de la largeur (de la tôle déroulée) pour obtenir des plaques.

Annexe 2 Capteurs inductifs



Capteurs standards

Détecteurs de proximité inductifs IME 18

Avantages

- Indice de protection IP 67
- Portée de 2 à 10 mm
- Fréquence de commutation jusqu'à 1 kHz
- Fonction NO ou NC
- Sortie NPN ou PNP
- Câble ou connecteur
- Version courte ou longue

Caractéristiques techniques	
Diamètre du filetage x pas [mm]	M18 x 1
Matériau du boîtier	Laiton plaqué nickel
Tension d'alimentation	CC 10 ... 30 V
Sorties	PNP/NPN
Fréq. de commutation max.	1 000/s
Indice de protection	IP 67/IP 68
Raccordement	Câble / connecteur

Référence	Désignation	Description	Portée	Boîtier court 50 mm	Boîtier std 69 mm	Sortie PNP	Raccordement
1040937	IME18-05BPOZC0K	Gamme standard	5 mm noyable	✔		normalement fermée	connecteur M12, 4 pôles
1040938	IME18-05BPOZC0S				✔	normalement fermée	
1040933	IME18-05BPSZC0K			✔		normalement ouverte	
1040934	IME18-05BPSZC0S				✔	normalement ouverte	
1040953	IME18-08NPOZC0K		8 mm non noyable	✔		normalement fermée	
1040954	IME18-08NPOZC0S				✔	normalement fermée	
1040949	IME18-08NPSZC0K			✔		normalement ouverte	
1040950	IME18-08NPSZC0S				✔	normalement ouverte	
1040969	IME18-08BPOZC0K	Gamme advanced	8 mm noyable	✔		normalement fermée	
1040970	IME18-08BPOZC0S				✔	normalement fermée	
1040965	IME18-08BPSZC0K			✔		normalement ouverte	
1040966	IME18-08BPSZC0S				✔	normalement ouverte	
1040985	IME18-12NPOZC0K		12 mm non noyable	✔		normalement fermée	
1040986	IME18-12NPOZC0S				✔	normalement fermée	
1040981	IME18-12NPSZC0K			✔		normalement ouverte	
1040982	IME18-12NPSZC0S				✔	normalement ouverte	

Capteurs standards

Détecteurs de proximité inductifs IME 30

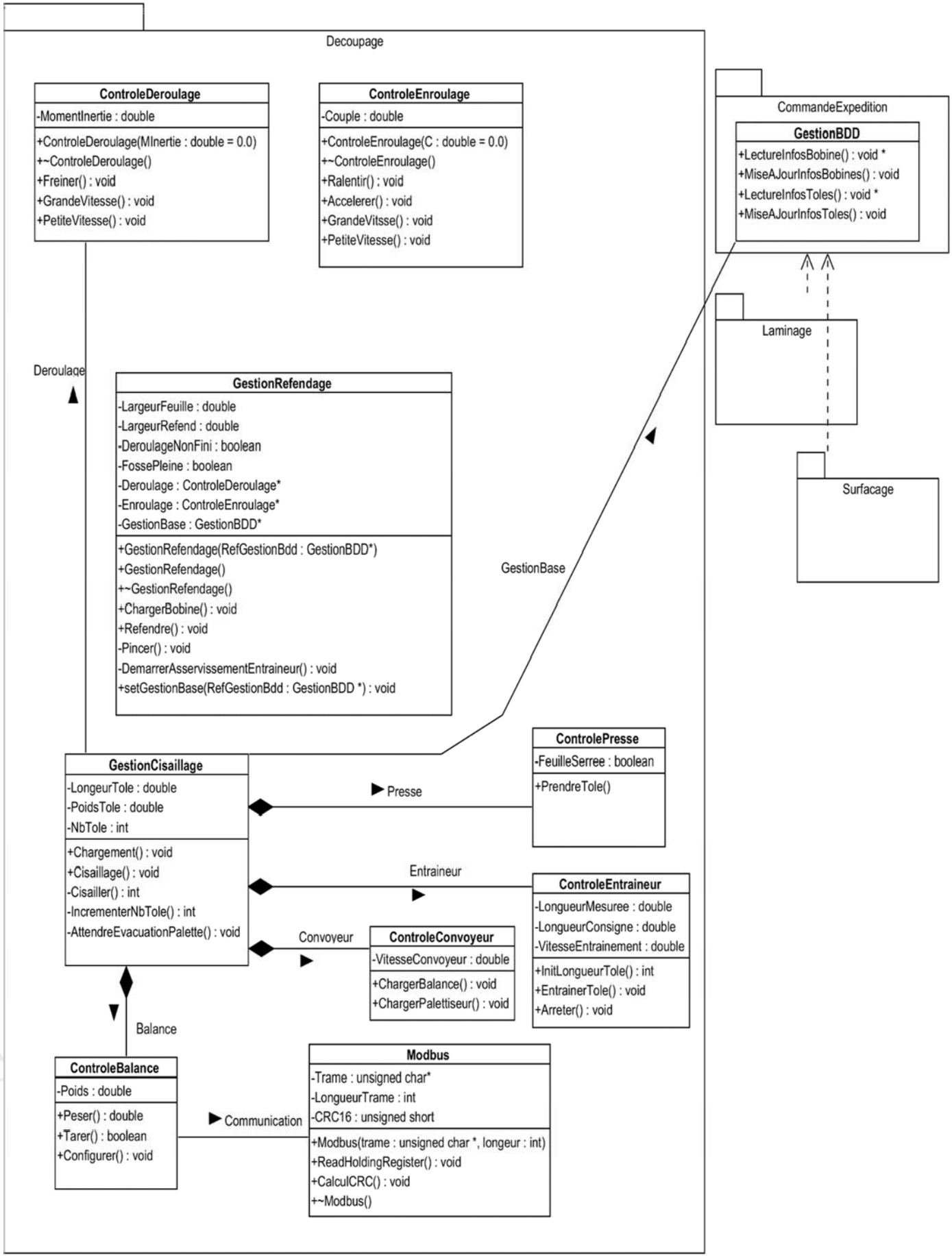
Avantages

- Indice de protection IP 67
- Portée de 2 à 10 mm
- Fréquence de commutation jusqu'à 500 kHz
- Fonction NO ou NC
- Sortie NPN ou PNP
- Câble ou connecteur
- Version courte ou longue

Caractéristiques techniques	
Diamètre du filetage x pas [mm]	M30 x 1
Matériau du boîtier	Laiton plaqué nickel
Tension d'alimentation	CC 10 ... 30 V
Sorties	PNP/NPN
Fréq. de commutation max.	500/s
Indice de protection	IP 67/IP 68
Raccordement	Câble / connecteur
Température d'utilisation	–20 °C ... +65 °C

Référence	Désignation	Description	Portée	Boîtier court 52 mm	Boîtier std 71 mm	Sortie PNP	Raccordement
1041001	IME30-10BPOZC0K	Gamme standard	10 mm noyable	✔		normalement fermée	connecteur M12 4 pôles
1041002	IME30-10BPOZC0S		10 mm noyable		✔	normalement fermée	
1040997	IME30-10BPSZC0K		10 mm noyable	✔		normalement ouverte	
1040998	IME30-10BPSZC0S		10 mm noyable		✔	normalement ouverte	
1041017	IME30-15NPOZC0K		15 mm non noyable	✔		normalement fermée	
1041018	IME30-15NPOZC0S		15 mm non noyable		✔	normalement fermée	
1041013	IME30-15NPSZC0K		15 mm non noyable	✔		normalement ouverte	
1041014	IME30-15NPSZC0S		15 mm non noyable		✔	normalement ouverte	
1041033	IME30-15BPOZC0K	Gamme advanced	15 mm noyable	✔		normalement fermée	
1041034	IME30-15BPOZC0S		15 mm noyable		✔	normalement fermée	
1041029	IME30-15BPSZC0K		15 mm noyable	✔		normalement ouverte	
1041030	IME30-15BPSZC0S		15 mm noyable		✔	normalement ouverte	
1041049	IME30-20NPOZC0K		20 mm non noyable	✔		normalement fermée	
1041050	IME30-20NPOZC0S		20 mm non noyable		✔	normalement fermée	
1041045	IME30-20NPSZC0K		20 mm non noyable	✔		normalement ouverte	
1041046	IME30-20NPSZC0S		20 mm non noyable		✔	normalement ouverte	

Annexe 3 : Diagramme de classes partiel du système



Annexe 4 : Extrait documentation MODBUS

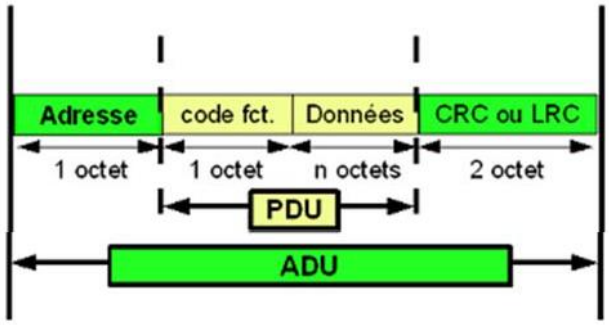
Présentation

MODBUS est un protocole de communication couramment utilisé dans l'industrie pour faire communiquer des automates programmables.
Le protocole MODBUS est un protocole de dialogue basé sur une structure hiérarchisée entre un maître et plusieurs esclaves. Les esclaves possèdent une adresse comprise entre 1 et 64.

Échange Maître/ Esclave sur Modbus

Le maître interroge un esclave de numéro unique sur le réseau et attend de la part de cet esclave une réponse. Lorsque l'esclave envoie sa réponse, il place sa propre adresse dans le champ adresse afin que le maître puisse l'identifier.

Format d'une trame Modbus



Une trame Modbus est composée de deux parties :

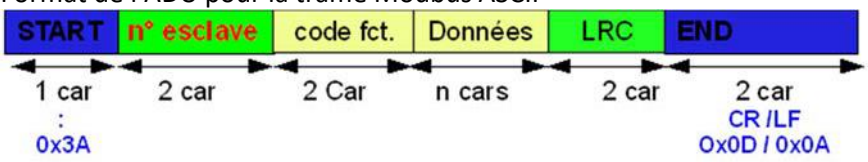
- "ADU" Application Data Unit : Cette partie est fonction de la couche de communication sur laquelle s'appuie Modbus.
  L'exemple ci-dessus correspond à un *ADU Modbus sur liaison série*.
- "PDU" Protocol Data Unit : Est composé des champs :
  - *"Code fonction"* : Ce champ est prédéfini par le protocole Modbus (voir paragraphe 2 de cette annexe). Ce code fonction définie l'action à exécuter sur l'esclave.
    Par exemple le code fonction 0x0F (write Multiple Coils) permet de modifier les sorties numériques sur un esclave.
  - *"Données"* : Dans le champ "Données" sont présentes des informations relatives au code fonction (exemple : adresse de registre pour l'écriture sur les sorties) et des données à échanger entre le maître et l'esclave (exemple : les informations relatives aux sorties à mettre à l'état haut).

Codage des trames Modbus sur liaison série :

Deux types de codage peuvent être utilisés pour communiquer sur un réseau Modbus sur liaison série : ASCII ou RTU

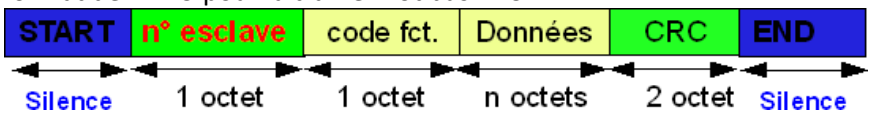
- Mode *ASCII* :
  - Chaque octet composant une trame est codé avec 2 caractères ASCII (2 fois 8 bits).
    Par exemple, la valeur 01 est codée avec les codes ASCII des symboles '0' et '1' (soit 0x30 et 0x31).

- Format de l'ADU pour la trame Modbus ASCII

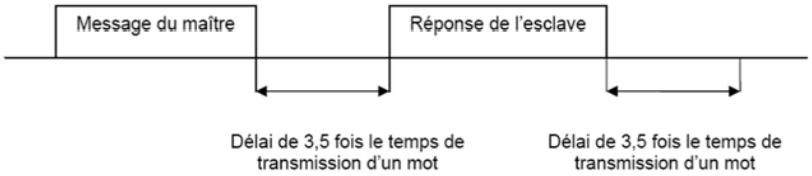


LRC : C'est la somme en hexadécimal modulo 256 du contenu de la trame hors délimiteurs (START et END), complémentée à 2. Le LRC est transmise en ASCII.

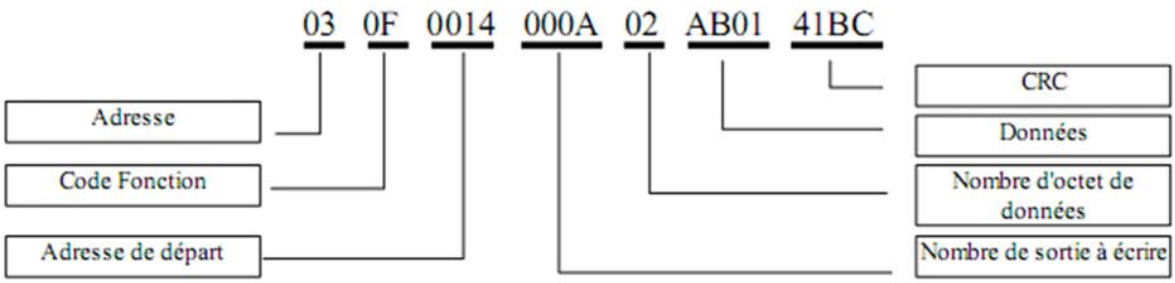
- Mode *RTU* :
  - Les octets composants la trame ne sont pas codés.
    Par exemple, la valeur $(15)_{10}$ aura pour octet 0x0F ou $(00001111)_2$.
  - Format de l'ADU pour la trame Modbus RTU



Silence : Les trames RTU ne comportent ni entête, ni délimiteur de fin.
La station réceptrice contrôle le temps séparant deux caractères consécutifs, s'il est supérieur à 3 caractères et demi, elle considère que le prochain caractère est un début de trame.



Exemple d'une requête modbus RTU pour écrire une série de 10 sorties commençant à l'adresse 20 sur l'esclave à l'adresse 3:



15 (0x0F) Write Multiple Coils

This function code is used to force each coil in a sequence of coils to either ON or OFF in a remote device. The Request PDU specifies the coil references to be forced. Coils are addressed starting at zero. Therefore coil numbered 1 is addressed as 0.

The requested ON/OFF states are specified by contents of the request data field. A logical '1' in a bit position of the field requests the corresponding output to be ON. A logical '0' requests it to be OFF.

The normal response returns the function code, starting address, and quantity of coils forced.

Request PDU

Function code	1 Byte	**0x0F**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Outputs	2 Bytes	0x0001 to 0x07B0
Byte Count	1 Byte	**N***
Outputs Value	**N*** x 1 Byte	

***N** = Quantity of Outputs / 8, if the remainder is different of 0 ⇒ N = N+1

Response PDU

Function code	1 Byte	**0x0F**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Outputs	2 Bytes	0x0001 to 0x07B0

Error

Error code	1 Byte	**0x8F**
Exception code	1 Byte	01 or 02 or 03 or 04

03 (0x03) Read Holding Registers

This function code is used to read the contents of a contiguous block of holding registers in a remote device. The Request PDU specifies the starting register address and the number of registers. In the PDU Registers are addressed starting at zero. Therefore registers numbered 1-16 are addressed as 0-15.

The register data in the response message are packed as two bytes per register, with the binary contents right justified within each byte. For each register, the first byte contains the high order bits and the second contains the low order bits.

Request :

Function code	1 Byte	**0x03**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Registers	2 Bytes	1 to 125 (0x7D)

Response :

Function code	1 Byte	**0x03**
Byte Count	1 Bytes	2 x **N***
Register Value	**N*** x 2 Bytes	1 to 125 (0x7D)

***N** = Quantity of Register

Error :

Function code	1 Byte	**0x83**
Exception code	1 Bytes	01 or 02 or 03 or 04
Register Value	N* x 2 Bytes	1 to 125 (0x7D)

Algorithme pour le calcul du CRC 16 bits d'une Trame:

```
Debut

    CRC = FFFFh
    OctetPointe = Premier octet de la trame
    Pour (chaque octet de la trame)
        CRC = CRC OU_Exclusif OctetPointe
        Pour Cpt variant de 1 à 8
            Si ((CRC AND 0001h) non NUL) alors
                CRC = CRC décalé d'un bit à droite
                CRC = CRC OU_Exclusif A001h
            sinon
                CRC = CRC décalé d'un bit à droite
            finSi
        finPour
        OctetPointe = Octet suivant de la trame
    FinPour

Fin.
```

Annexe 5 Aide-mémoire sur le langage S.Q.L.

1 DEFINITION

S.Q.L. (Structured Query Language) est un Langage normalisé de Requêtes Structurées et un standard d'accès aux bases de données relationnelles.

2 LES INSTRUCTIONS DE SQL

2.1 L'instruction 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 fonctions de critères.
- D'utiliser des fonctions arithmétiques et de groupements pour des calculs.

Syntaxe générale de l'instruction **SELECT** :

SELECT *Liste des champs séparés par une virgule*
FROM *Liste des tables concernées, séparées par une virgule* **WHERE** *Liste des critères de choix*

2.2 La projection

SELECT [DISTINCT] nomcol1 [, nomcol2 , …] **FROM** nomtable1 [, nomtable2 , ….] ;

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.

2.3 La restriction (ou sélection)

SELECT nom_col1 [, nom_col2, ….] **FROM** nom_table1 [, nom_Table2,….] **WHERE** conditions ;

La clause WHERE permet de sélectionner dans la table obtenue par SELECT … FROM … les tuples correspondants à des critères précis.

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 n_dep FROM LIVRAISON WHERE (prise > 25 AND prise < 50) ;
```

Extrait les numéros de dépositaires avec la restriction des quantités livrées comprises entre 26 et 49.

2.4 La jointure (ou sélection sur plusieurs tables)

En SQL, il est possible d'enchaîner plusieurs jointures dans la même instruction SELECT.

- En SQL de base

SELECT * FROM table1**,** table2**,** table3**, ...**

WHERE table1.attribut1**=**table2.attribut1 **AND** table2.attribut2**=**table3.attribut2 **AND** ...**;**

Exemple :

SELECT * FROM Produit, Détail_Commande

WHERE Produit.CodePrd=Détail_Commande.CodePrd ;

ou en utilisant des alias pour les noms des tables :

SELECT * FROM Produit A, Détail_Commande B

WHERE A.CodePrd=B.CodePrd ;

- Avec la clause **INNER JOIN** (jointure dite interne) à partir du SQL2, supportée aujourd'hui par tous les SGBDR :

SELECT *

FROM table1 **INNER JOIN** table2 **ON** table1.attribut1=table2.attribut1 **INNER JOIN** table3 **ON** table2.attribut2=table3.attribut3... ;

Le mot clé INNER est facultatif sur la plupart des SGBDR (sauf MS Access).

Cette notation rend plus lisible la requête en distinguant clairement les conditions de jointures, derrière ON, et les éventuelles conditions de sélection ou restriction, derrière WHERE.

De plus, l'oubli d'un ON (et donc de la condition de jointure) empêchera l'exécution de la requête, alors qu'avec l'ancienne notation, l'oubli d'une condition de jointure derrière WHERE, n'empêche pas l'exécution de la requête, produisant alors un bien coûteux produit cartésien entre les tables !

Le même exemple que précédemment en utilisant aussi les alias :

SELECT *

FROM Produit A INNER JOIN Détail_Commande B ON A.CodePrd=B.CodePrd ;

La norme SQL2 définit aussi l' équi-jointure naturelle, joignant les 2 tables sur l'ensemble des attributs qu'elles ont en commun, mais en ne gardant qu'une seule colonne pour chaque attribut joint, contrairement aux 2 expressions précédentes :

SELECT *

FROM table1 **NATURAL JOIN** table2 ;

Il est aussi possible de restreindre (ou préciser) le ou les attributs de jointure avec USING :

SELECT *

FROM table1 **INNER JOIN** table2 **USING (**attribut1**)** ;

NATURAL JOIN et USING ne sont pas supportés par tous les SGBDR.

2.5 L'instruction INSERT

INSERT permet d'ajouter un ou plusieurs enregistrements dans une table. *Insertion d'un enregistrement :*

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') ;
```

Insert un nouvel enregistrement dans la table dépositaire avec les valeurs :

N_dep=68 ; nom_dep='Quentin' ; adr_dep='Marseille'

2.6 L'instruction UPDATE

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

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'.

2.7 L'instruction DELETE

Elle permet de supprimer un enregistrement d'une table.

DELETE FROM nom_table **WHERE** conditions… ;

Annexe 6 : Architecture informatique.