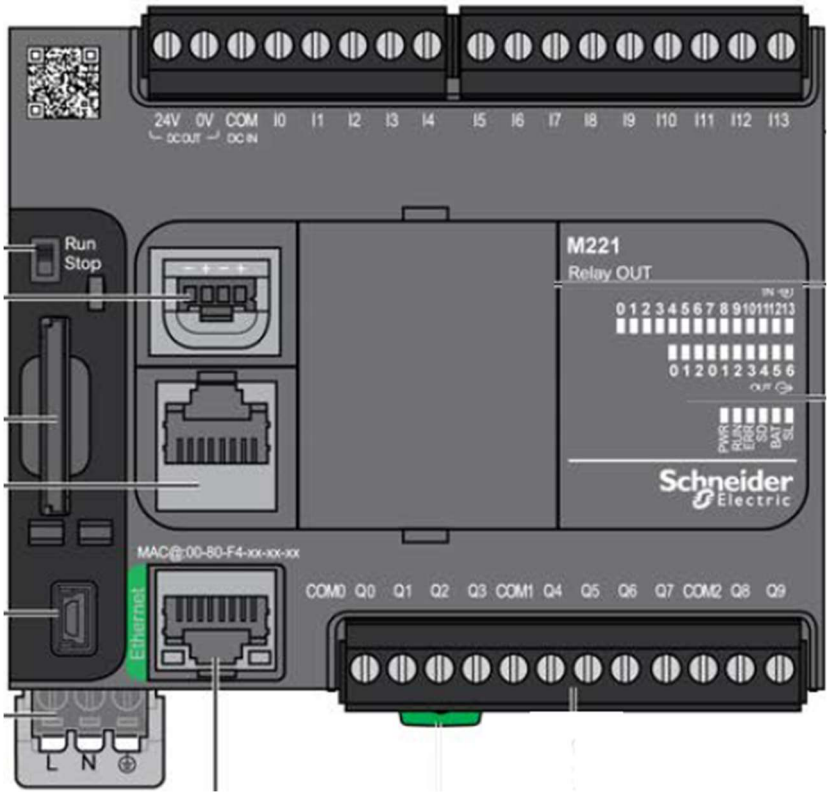
# Découverte du protocole MODBUS

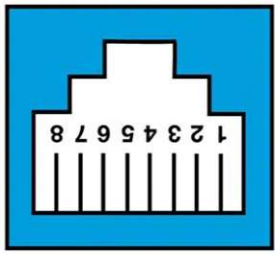
## 1 – Modbus sur liaison série (RTU ou ASCII)

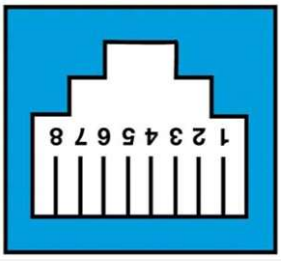
La documentation de l'automate est donnée en annexe.

⇨ Lancer le logiciel Machine Expert – Basic de Schneider
⇨ Créer un nouveau projet
⇨ Désactiver les protections en lecture et écriture
⇨ Enregistrer le projet
⇨ Choisir un automate M221 CE24R
⇨ Préciser l'emplacement des entrées numériques, les sorties numériques, les entrées analogiques, le port série et le port Ethernet.



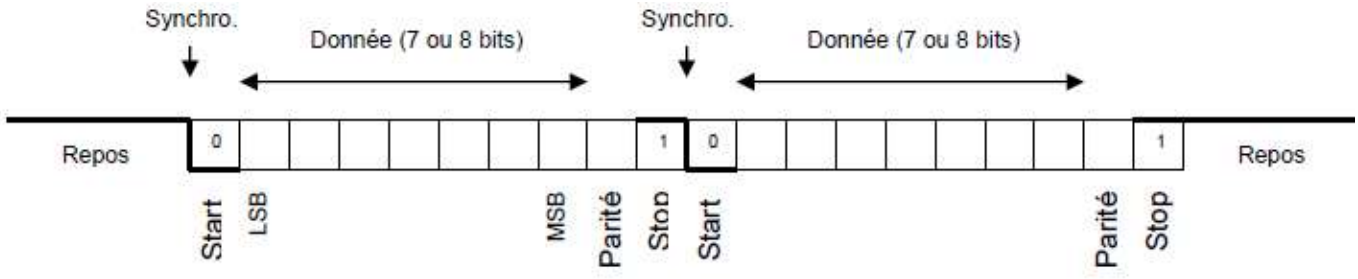

⇨ Pour la ligne série SL1, préciser les broches à utiliser pour la liaison RS232
⇨ Pour la ligne série SL1, préciser les broches à utiliser pour la liaison RS485



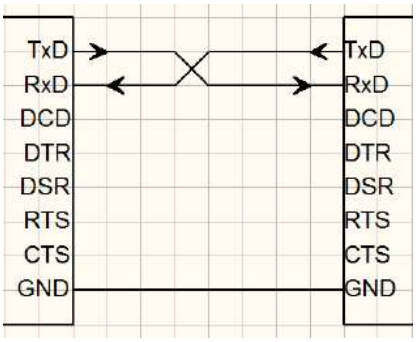
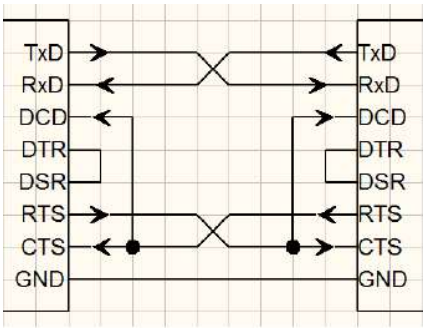
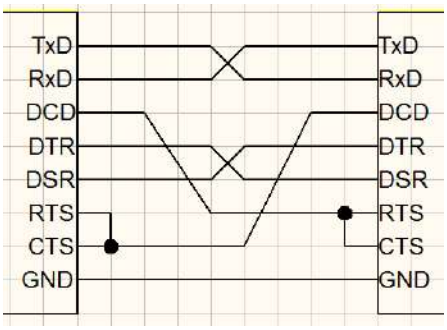



**Remarques sur les liaisons RS232 et RS485**

La transmission des données binaires se fait en série et utilise un codage appelé NRZ. Chaque donnée transmise démarre avec un bit de start à 0 et termine par au moins un bit de stop à 1. La ligne au repos est à 1.
Un bit de parité, utilisé pour contrôler des erreurs de transmission, peut être ajouté à la donnée.



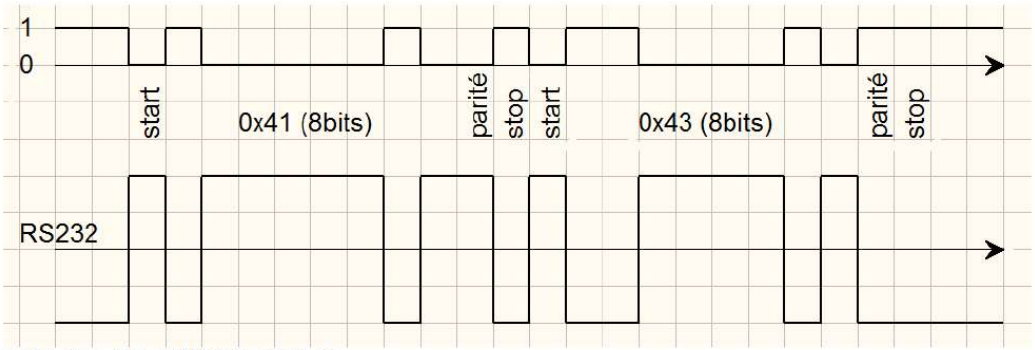

**RS232** : Une RS232 relie 2 terminaux avec un câble croisé, avec des lignes de contrôle ou pas.

| Liaison 3 fils | Liaison 5 fils | Câble « nul modem |
|---|---|---|
|  |  |  |
| Protocole :<br>• logiciel Xon / Xoff uniquement | Protocoles :<br>• Logiciel Xon / Xof<br>• Matériel RTS / CTS | Protocoles :<br>• Logiciel Xon / Xof<br>• Matériel DTR / DSR |



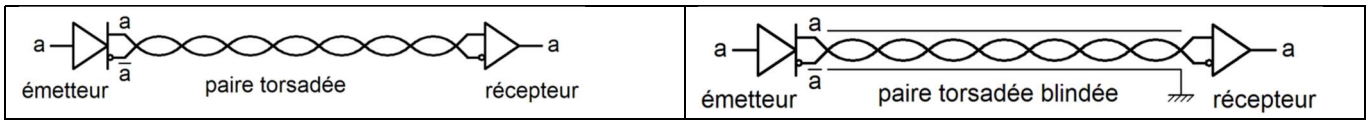
Les tensions sur une ligne RS232 sont inversées (avec tension négative) par rapport au codage

Transmission des codes ASCII de 'A' (41h) et 'C' (43h) à la vitesse de 4800 bauds, 8 bits de donnée, parité paire, 2 bits de stop.

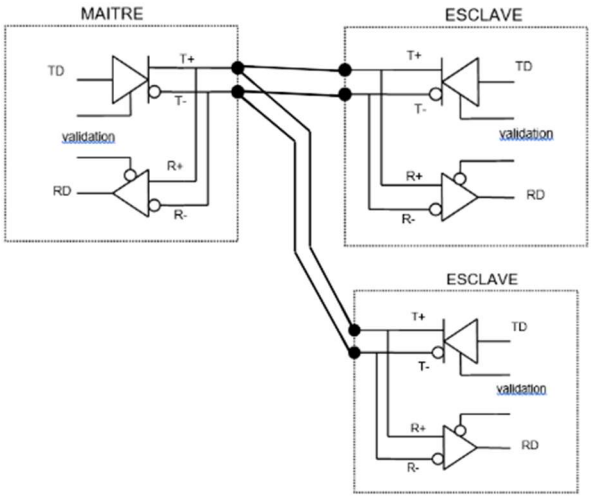



Durée d'un bit = 1/4800 = 208 µs

**RS485** : Une liaison RS485 utilise une ligne différentielle (paire torsadée) sur laquelle on peut connecter 32 terminaux. Un terminal a alors le rôle de maître, et les autres le rôle d'esclave.




Connexion d'un maître à 2 esclaves




## Comparatif RS232 / RS485

| | RS232 | RS485 |
|---|---|---|
| Mode | Asymétrique | Différentiel |
| Nombre d'émetteur | 1 | 32 |
| Nombre de récepteur | 1 | 32 |
| Longueur de câble | 15m | 1200m |
| Débit maxi | 20kb/s | 10Mb/s<br>100kb/s (1200m) |
| Tension de sortie (chargée) | ±5V min.<br>±15V max. | Tension différentielle<br>±1,5V min.<br>±6V max. |

Dans le menu Configuration/SL1 (ligne série) du logiciel Machine Expert – Basic :
⇨ Préciser si le protocole Modbus est disponible sur la ligne série

⇨ Indiquer les débits disponibles

⇨ Préciser les choix de la parité

⇨ Indiquer les choix de la taille en bits des données

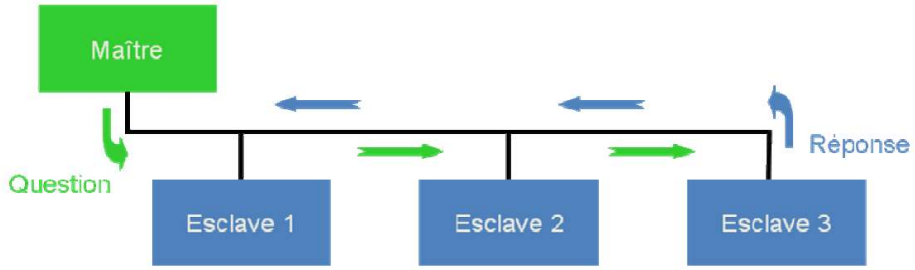
⇨ Indiquer les choix possibles en nombre de bits d'arrêt (stop)

⇨ Les supports physiques disponibles (RS232, RS485…)

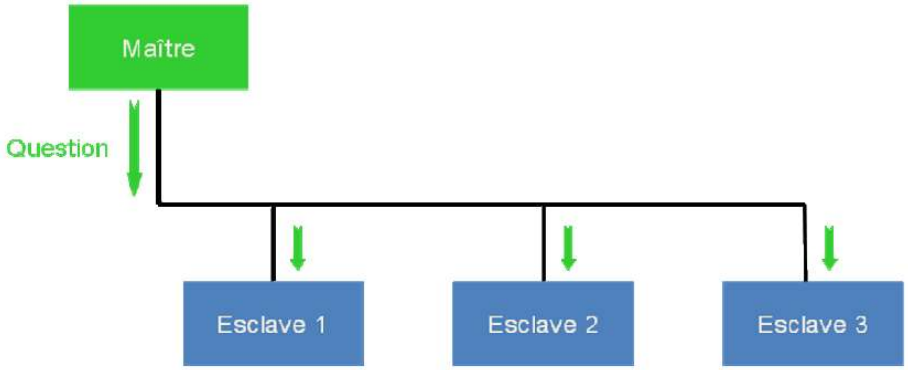
Dans le cas d'une liaison RS485, il est possible de relier 32 dispositifs sur la même ligne.

Un module est maître, et les autres modules sont esclaves. Chaque esclave est identifié par une adresse, comprise entre 1 et 247.

Interrogation d'un esclave par le maître

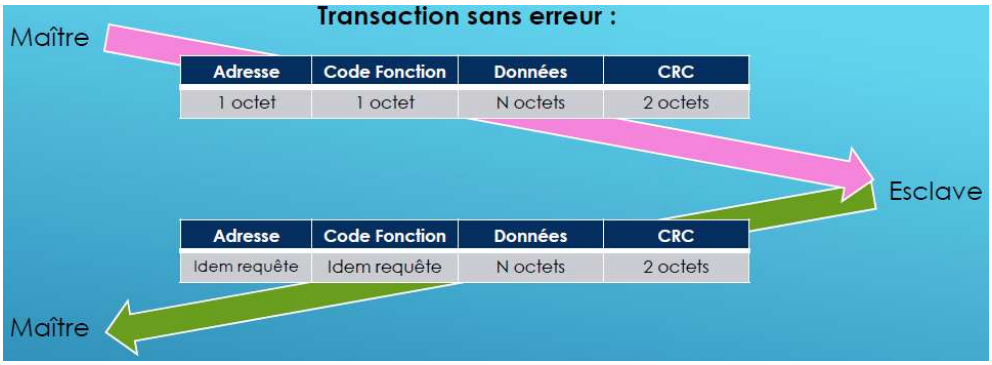



Diffusion vers tous les esclaves




⇨ A partir du menu Configuration/SL1/Modbus du logiciel, préciser si l'automate accepte un mode Maître ou un mode Esclave.

Lors d'une transmission entre un maître et un esclave, les trames échangées (cas où il n'y a pas d'erreur) ont le format suivant :



On retrouve :

- L'adresse de l'esclave
- Un code fonction
- N octets de données
- Un CRC pour le contrôle de la transmission

Les codes fonctions acceptés par le M221 sont les suivants :

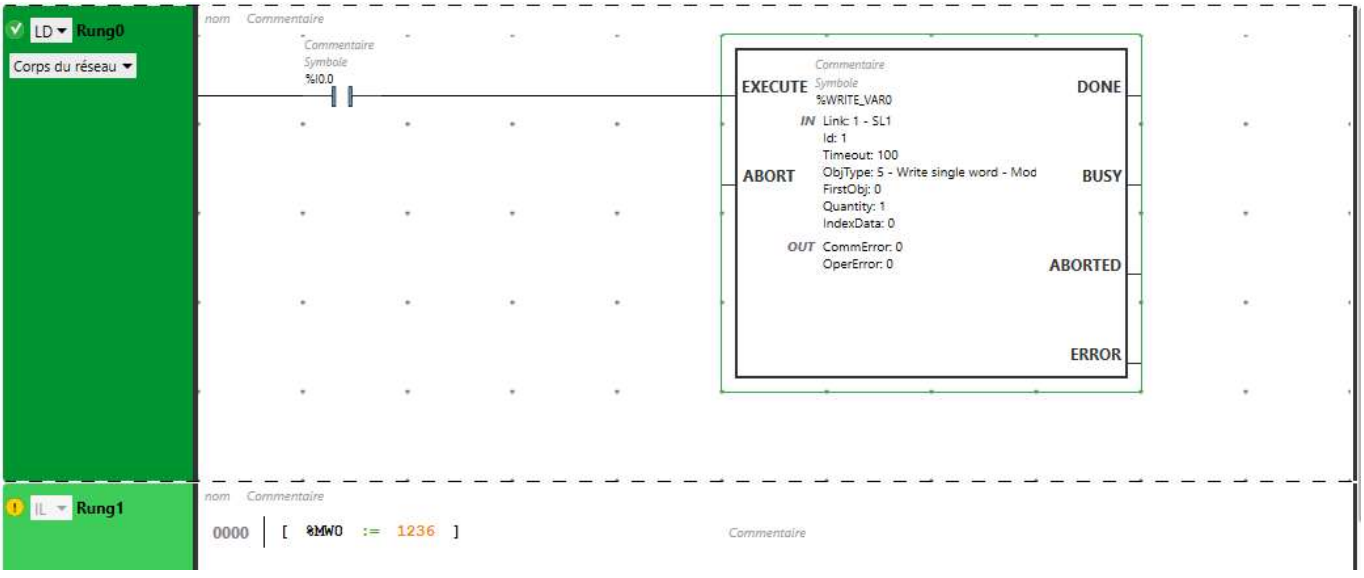
## What are the supported Modbus Function Codes for a M221 controller?

This table lists the function codes supported by both serial Modbus and Modbus TCP and their effect on controller mem variables:

| Supported Modbus Function Code | Supported Sub-Function Code | Description |
|---|---|---|
| 1 (0x01) | - | Read multiple internal bits %M |
| 2 (0x02) | - | Read multiple internal bits %M |
| 3 (0x03) | - | Read multiple internal registers %MW |
| 4 (0x04) | - | Read multiple internal registers %MW |
| 5 (0x05) | - | Force single internal bit %M |
| 6 (0x06) | - | Write single internal register %MW |
| 8 (0x08) | 0 (0x00), 10 (0x0A)...18 (0x12) | Diagnostics |
| 15 (0x0F) | - | Write multiple internal bits %M |
| 16 (0x10) | - | Write multiple internal registers %MW |
| 23 (0x17) | - | Read/write multiple internal registers %MW |
| 43 (0x2B) | 14 (0x0E) | Read device identification (regular service) |

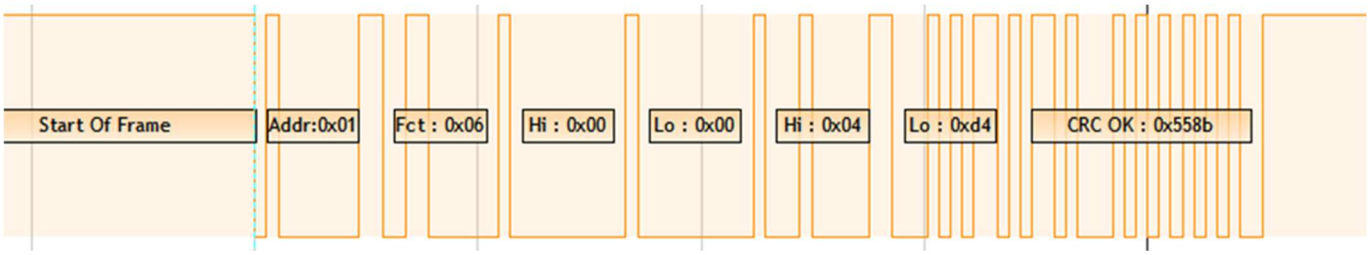
Note: For function code 5 and 6 you must use the EXCH function. These two function codes are not available with the

⇨ Sous le logiciel Machine Expert – Basic, réaliser le programme suivant (il ne sera pas testé), on utilise ici le bloc « Write Var ».



Dans cet exemple le mot mémoire de l'adresse 0 est envoyé à l'esclave d'adresse 1, sur la ligne SL1 configurée en RS485 maître. L'envoi est déclenché sur un front montant de I0.0. Le mot mémoire (16 bits) à l'adresse 0 est initialisé à la valeur 1236 (4D4 en hexa).

Avec un analyseur de trame, on relève les informations envoyées :




⇨ Retrouver l'adresse de l'esclave, le code fonction, l'adresse de destination, la donnée transmise (voir ci-dessous l'extrait de la norme)

### 6.6 06 (0x06) Write Single Register

This function code is used to write a single holding register in a remote device.

The Request PDU specifies the address of the register to be written. Registers are addressed starting at zero. Therefore register numbered 1 is addressed as 0.

The normal response is an echo of the request, returned after the register contents have been written.

**Request**

| | | |
|---|---|---|
| Function code | 1 Byte | **0x06** |
| Register Address | 2 Bytes | 0x0000 to 0xFFFF |
| Register Value | 2 Bytes | 0x0000 to 0xFFFF |

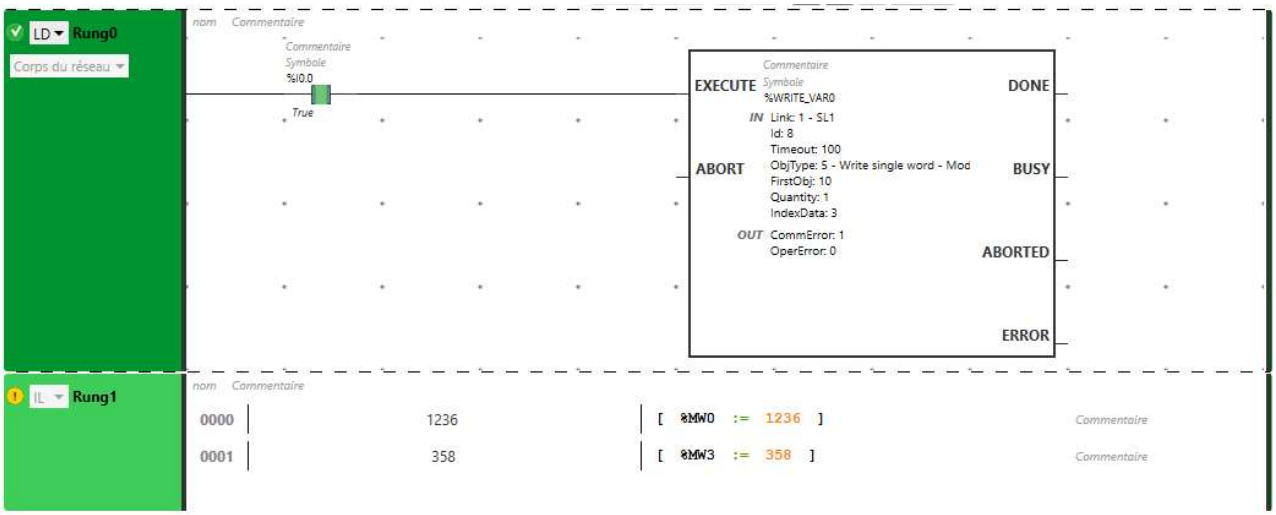
**Response**

| | | |
|---|---|---|
| Function code | 1 Byte | **0x06** |
| Register Address | 2 Bytes | 0x0000 to 0xFFFF |
| Register Value | 2 Bytes | 0x0000 to 0xFFFF |

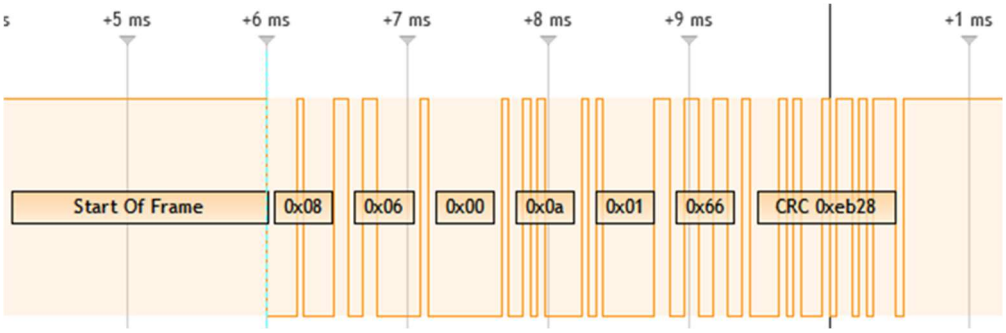
**Error**

| | | |
|---|---|---|
| Error code | 1 Byte | **0x86** |
| Exception code | 1 Byte | 01 or 02 or 03 or 04 |

On modifie la programmation de la manière suivante :



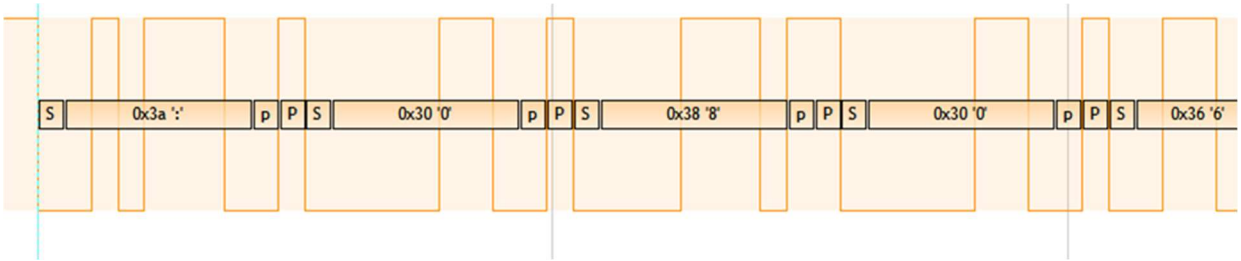

Et on obtient la trame suivante :

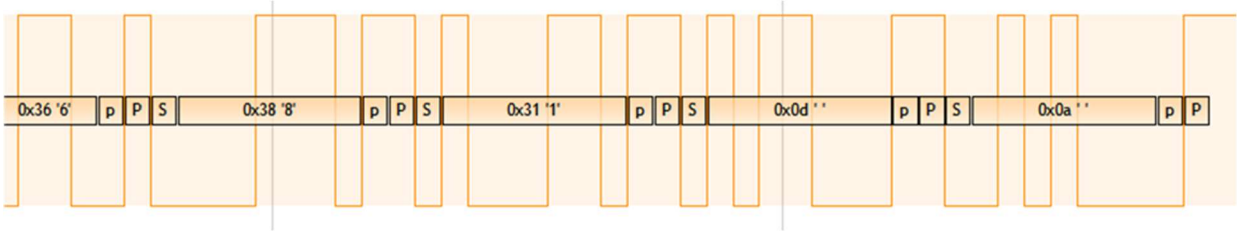



⇨ Justifier les codes relevés sur cette trame, en faisant la correspondance avec la nouvelle programmation.

En mode ASCII, les données sont remplacées par les caractères représentant une notation hexadécimale des valeurs.

Exemples : La valeur 0x01 0x66 (notation hexa décimale du nombre binaire) sera remplacée par
0x30 (‘0’) 0x31 (‘1’) 0x36 (‘6’) 0x36 (‘6’)
La trame commence par ‘ :’ et finie par un retour chariot (\r = 0x0d) et un saut de ligne (\n =0x0A).

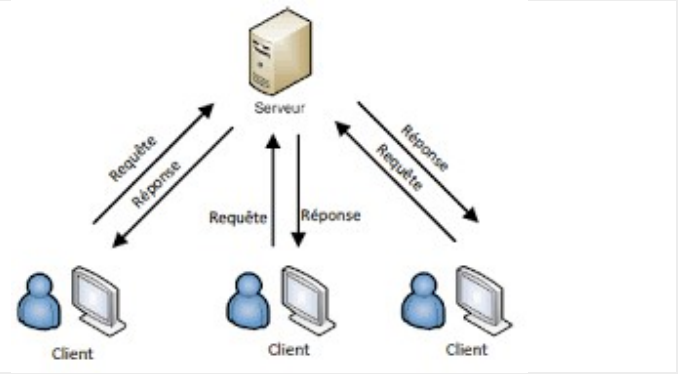


## 2 – Modbus TCP

Les trames MODBUS sont encapsulées dans des trames Ethernet. Le mode de transport est TCP, qui utilise l'architecture Client / Serveur. Le port d'application est le 502.

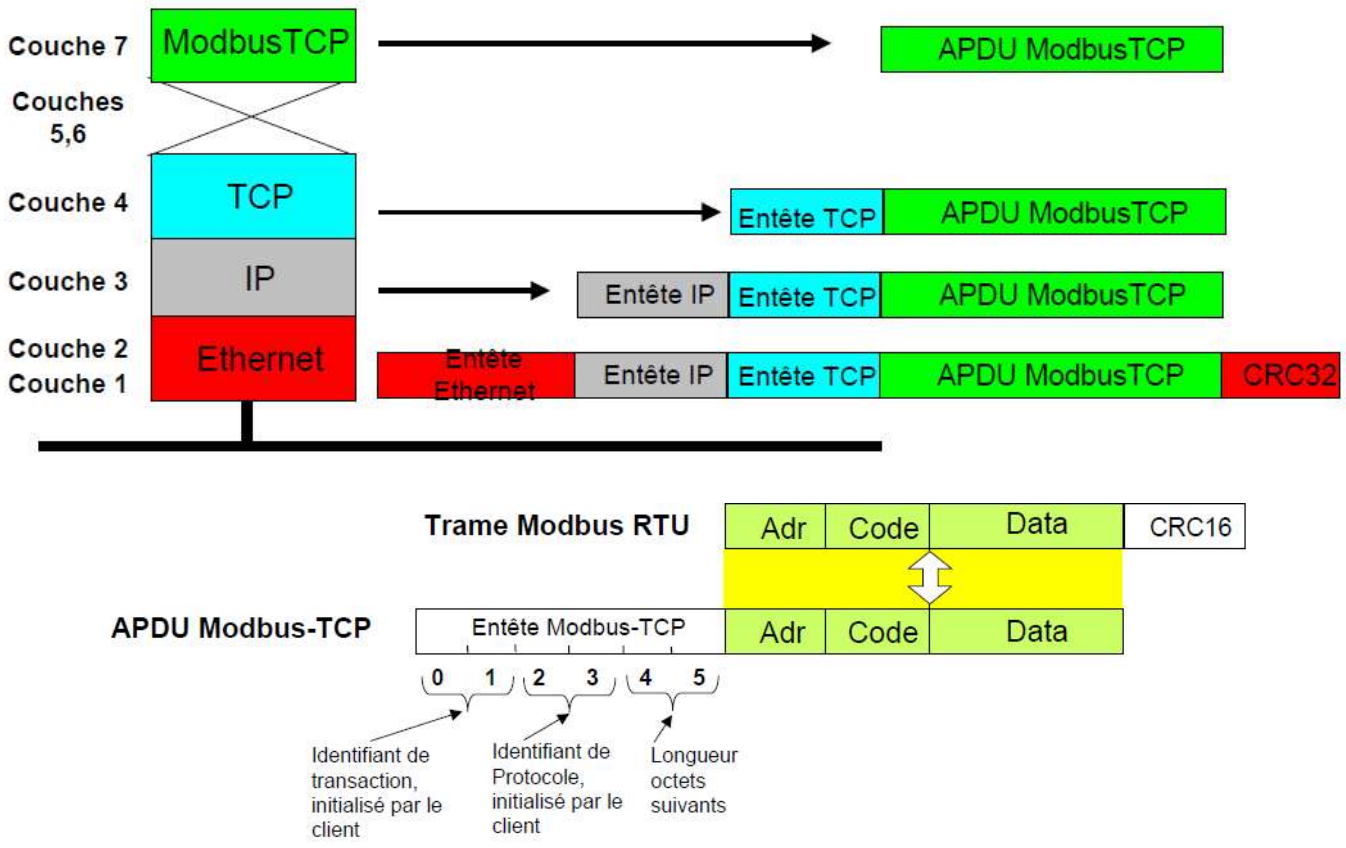
Les clients se connectent à un serveur. Ils font des demandes au serveur. Le serveur répond aux demandes.




⇨ Dans le menu Configuration/ETH1 du logiciel, préciser s'il est possible d'activer l'automate en mode serveur Modbus TCP

⇨ Dans le menu Configuration/ETH1/Modbus TCP, préciser s'il est possible d'activer l'automate en mode client Modbus TCP

Les trames Modbus sont encapsulées dans des trames Ethernet, au niveau application.



On a enregistré l'échange des trames entre le logiciel NodeRed (installé sur Raspberry, en mode client Modbus) et l'automate programmable (en mode serveur Modbus).
La première trame correspond à la demande du client, et la deuxième à la réponse du serveur.
La programmation de l'automate reste la même que précédemment.

Sur ces relevés, en entourant les informations demandées :
⇨ Identifier les adresses IP de la raspberry et de l'automate
⇨ Identifier le port de destination utilisé par Modbus
⇨ Déterminer le code fonction Modbus
⇨ Retrouver les données renvoyées par l'automate.

### 03 (0x03) Read Holding Registers

**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 Byte | 2 x **N*** |
| Register value | **N*** x 2 Bytes | |

***N** = Quantity of Registers