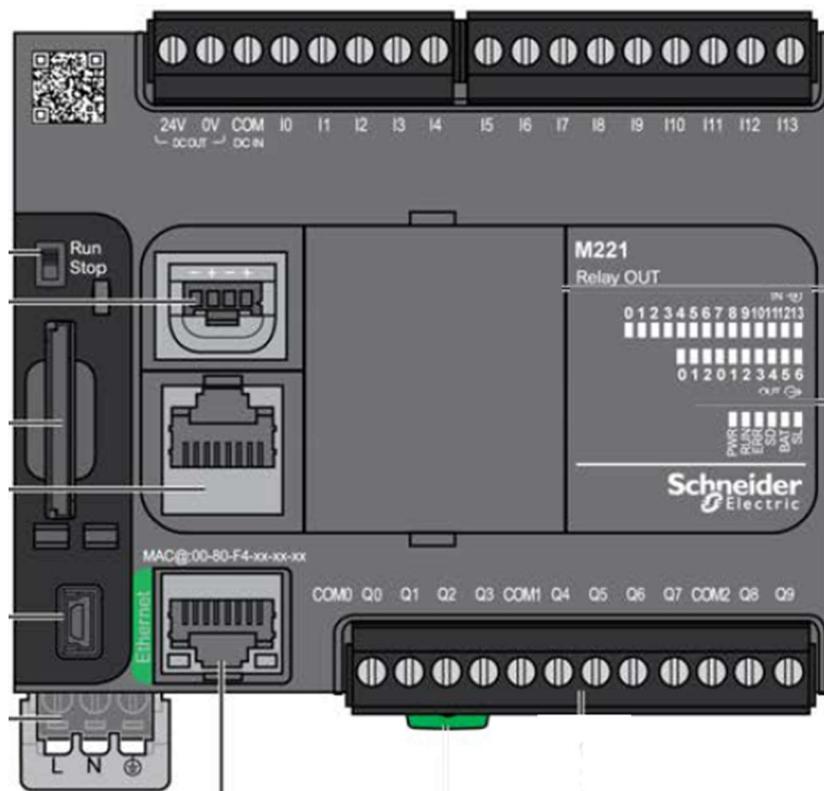


Découverte du protocole MODBUS

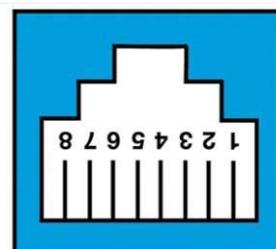
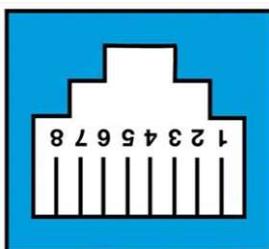
1 – Modbus sur liaison série (RTU ou ASCII)

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.

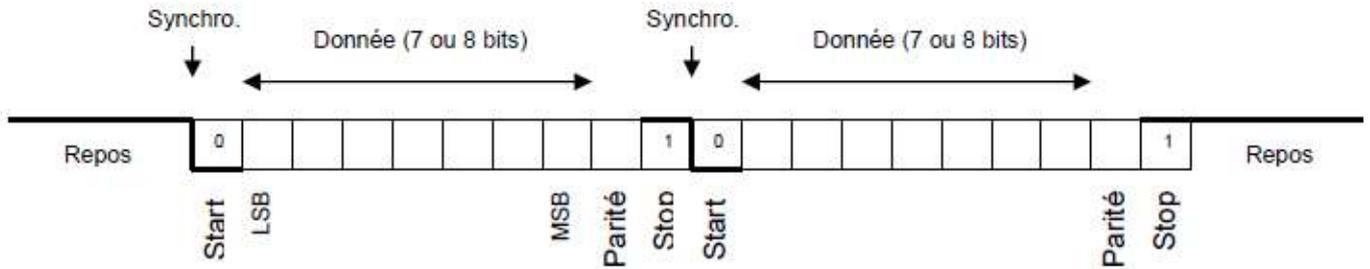


- ⇒ Pour la ligne série SL1, préciser les broches à utiliser pour la liaison RS232
- ⇒ Pour la ligne série SL1, préciser les broches à utiliser pour la liaison RS485

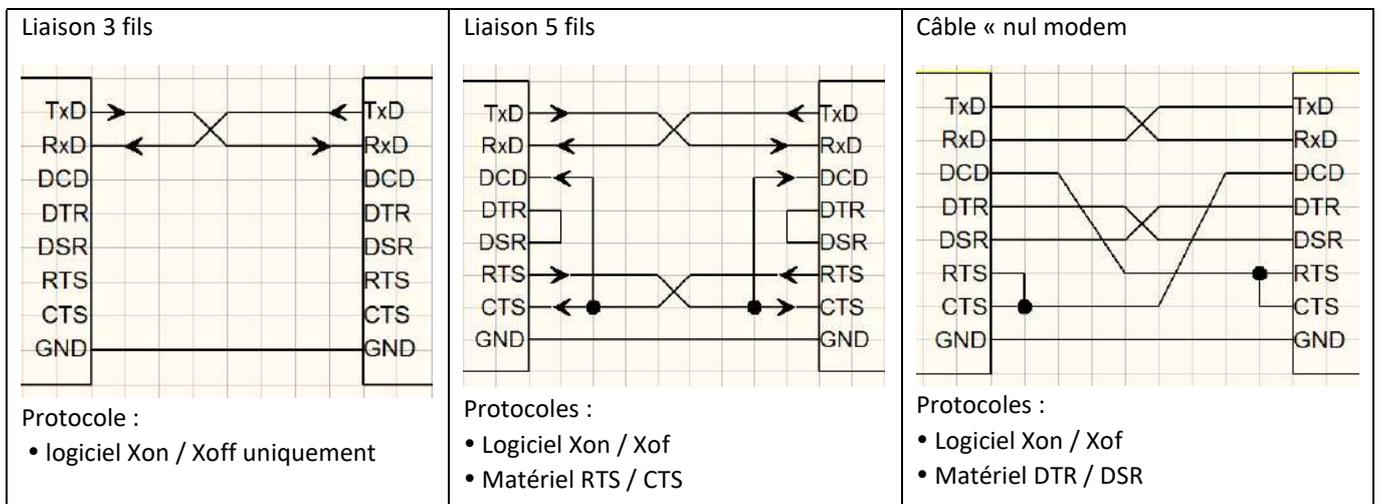


Remarques sur les liaisons RS232 et RS485

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.

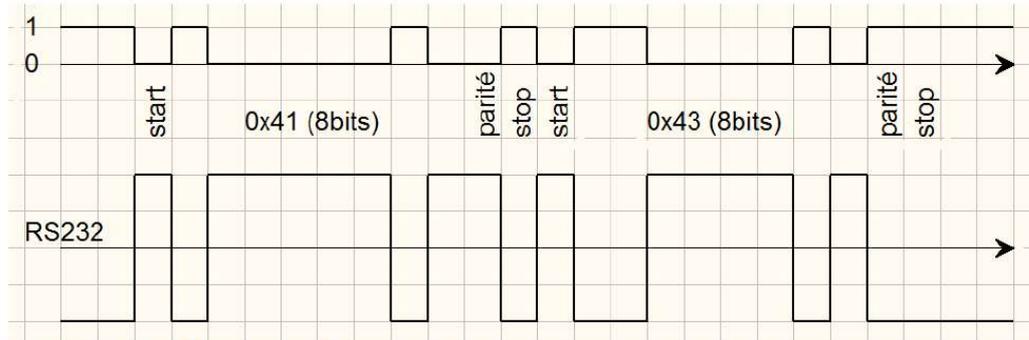


RS232 : Une RS232 relie 2 terminaux avec un câble croisé, avec des lignes de contrôle ou pas.



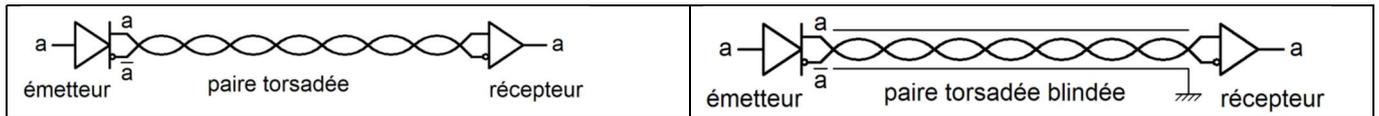
Les tensions sur une ligne RS232 sont inversées (avec tension négative) par rapport au codage

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.

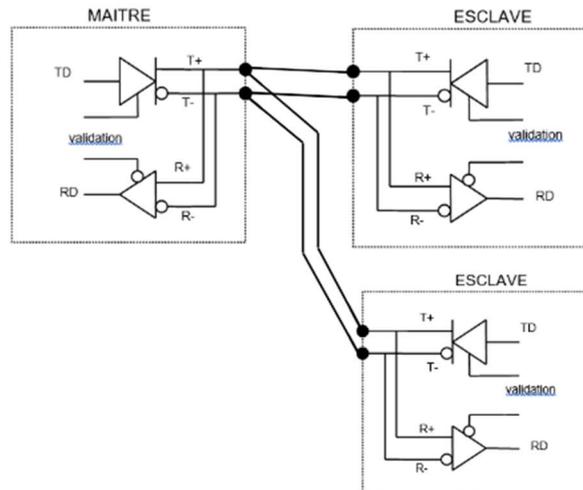


Durée d'un bit = $1/4800 = 208 \mu\text{s}$

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 100kb/s (1200m)
Tension de sortie (chargée)	±5V min. ±15V max.	Tension différentielle ±1,5V min. ±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

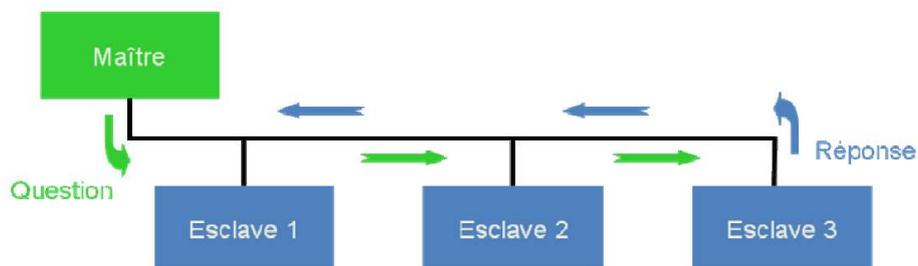
⇒ Indiquer les choix possibles en nombre de bits d'arrêt (stop)

⇒ Les supports physiques disponibles (RS232, RS485...)

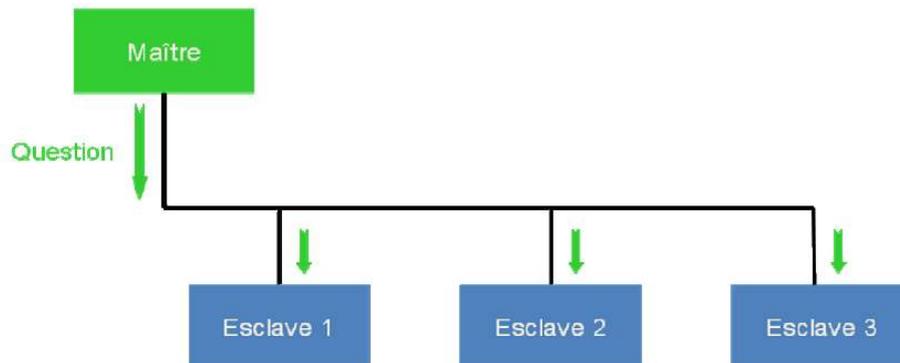
Dans le cas d'une liaison RS485, il est possible de relier 32 dispositifs sur la même ligne.

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

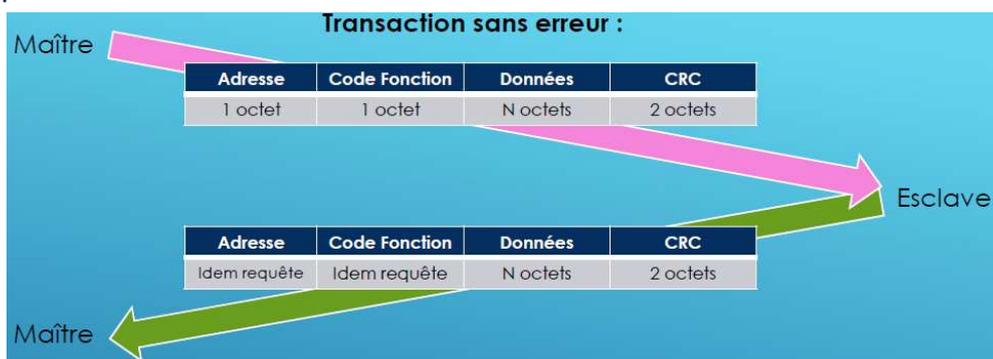


Diffusion vers tous les esclaves



⇒ 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 memory 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 ».

The screenshot shows a PLC programming environment with two rungs. Rung 0 (Rung0) is selected and contains a normally open contact labeled '%I0.0'. A line from this contact connects to the 'EXECUTE' input of a 'WRITE_VAR' function block. The function block has several parameters: 'Symbol' is '%WRITE_VAR', 'Link' is '1 - SL1', 'Id' is '1', 'Timeout' is '100', 'ObjType' is '5 - Write single word - Mod', 'FirstObj' is '0', 'Quantity' is '1', 'IndexData' is '0', 'CommError' is '0', and 'OperError' is '0'. The function block has four outputs: 'DONE', 'BUSY', 'ABORTED', and 'ERROR'. Rung 1 (Rung1) is partially visible below and contains a coil labeled '0000' with the assignment '[%MW0 := 1236]'.

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 IO.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

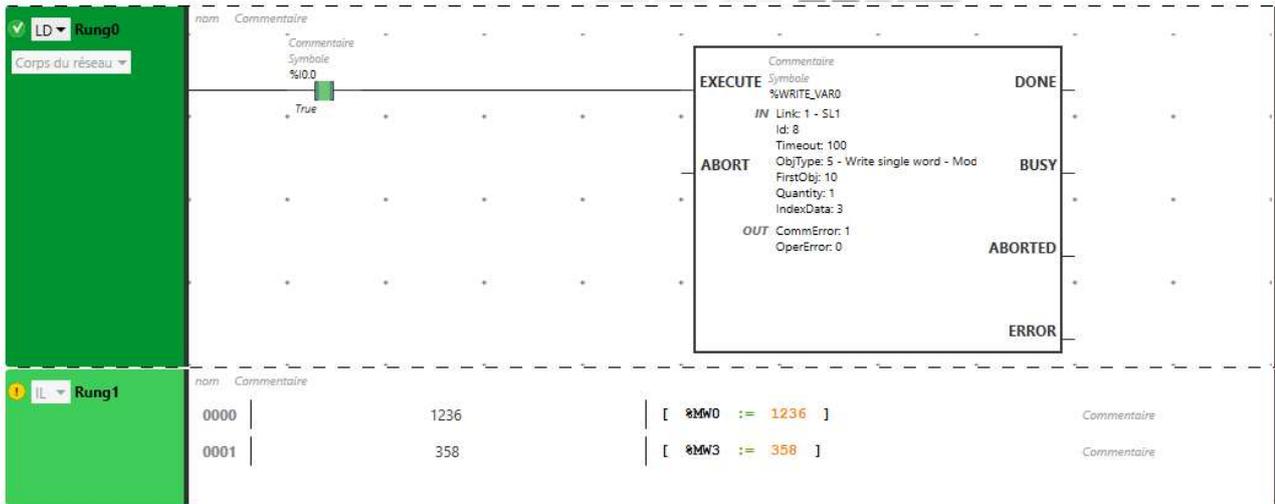
Response

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

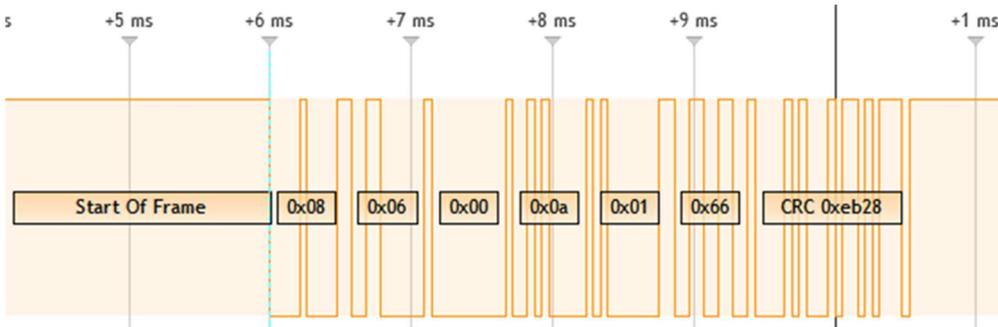
Error

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 :

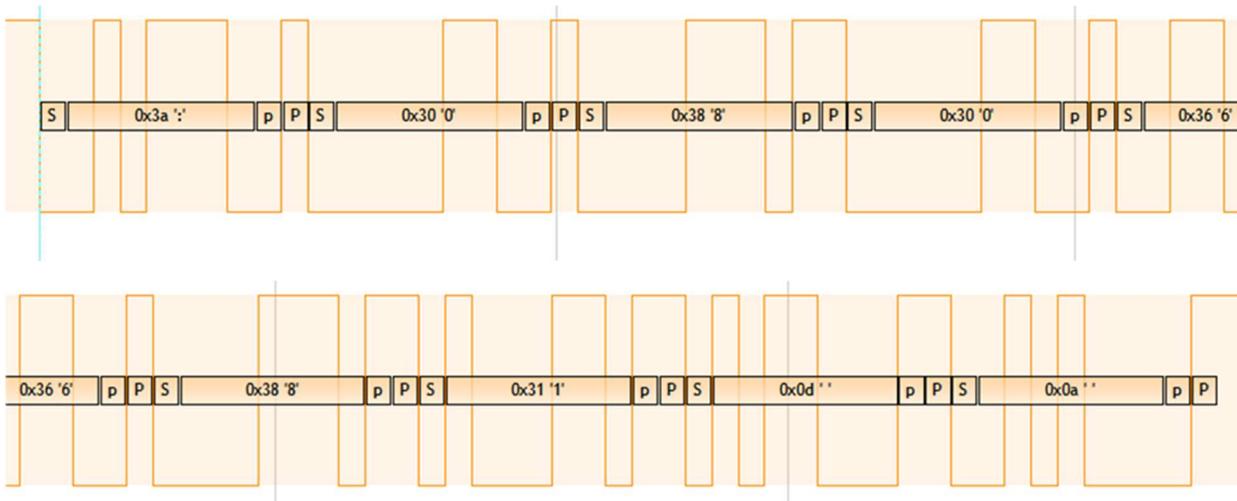


⇒ Justifier les codes relevés sur cette trame, en faisant la correspondance avec la nouvelle programmation.

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')

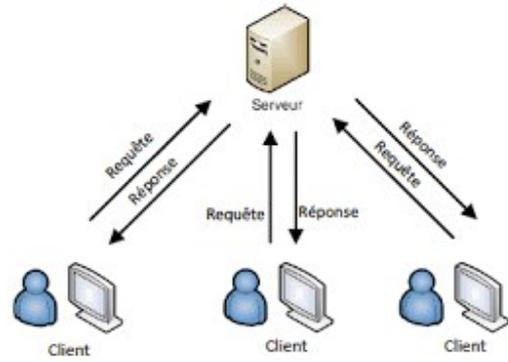
La trame commence par ':' et finie par un retour chariot (\r = 0x0d) et un saut de ligne (\n = 0x0a).



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.

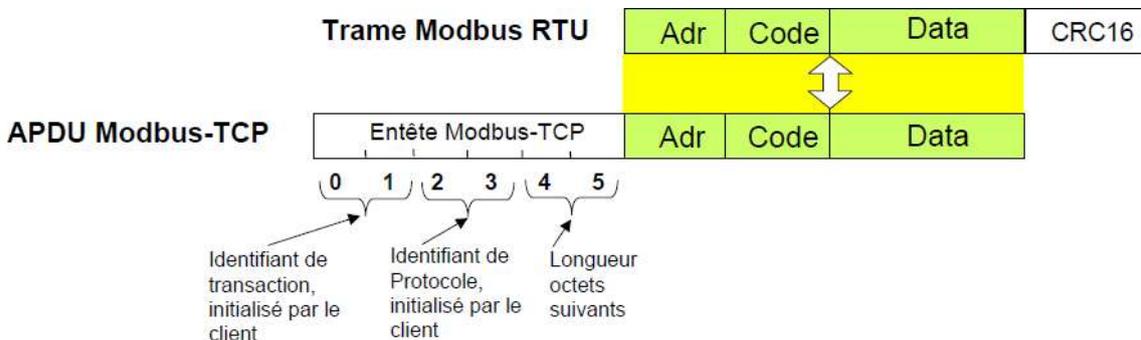
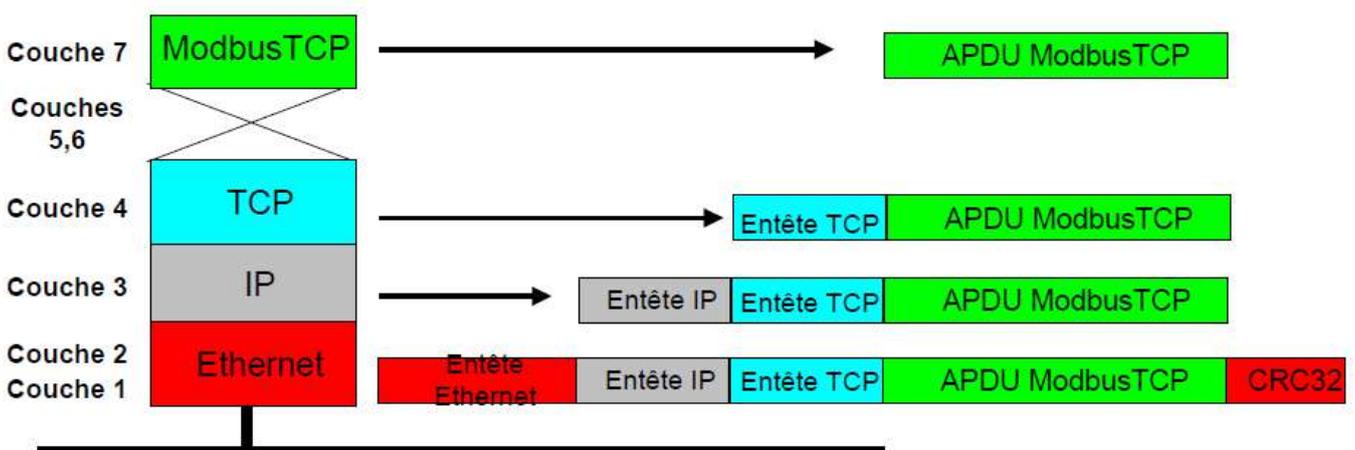
Les clients se connectent à un serveur. Ils font des demandes au serveur. Le serveur répond aux demandes.



⇒ 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

tcp.port==502

No.	Time	Source	Destination	Protocol	Length	Info
98	3.394070560	172.16.6.167	172.16.18.161	Modbus...	66	Query: Trans: 3; Unit: 255, Func: 3: Read Holding Registers
99	3.395289983	172.16.18.161	172.16.6.167	Modbus...	71	Response: Trans: 3; Unit: 255, Func: 3: Read Holding Registers
100	3.395384826	172.16.6.167	172.16.18.161	TCP	54	57638 → 502 [ACK] Seq=13 Ack=18 Win=64189 Len=0

Frame 98: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
 Ethernet II, Src: Raspberr_9d:a2:cc (b8:27:eb:9d:a2:cc), Dst: Telemch_0e:61:71 (00:80:f4:0e:61:71)
 Destination: Telemch_0e:61:71 (00:80:f4:0e:61:71)
 Source: Raspberr_9d:a2:cc (b8:27:eb:9d:a2:cc)
 Type: IPv4 (0x0800)
 Internet Protocol Version 4, Src: 172.16.6.167, Dst: 172.16.18.161
 Transmission Control Protocol, Src Port: 57638, Dst Port: 502, Seq: 1, Ack: 1, Len: 12
 Modbus/TCP
 Transaction Identifier: 3
 Protocol Identifier: 0
 Length: 6
 Unit Identifier: 255
 Modbus
 .000 0011 = Function Code: Read Holding Registers (3)
 Reference Number: 0
 Word Count: 4

```

0000 00 80 f4 0e 61 71 b8 27 eb 9d a2 cc 00 00 45 00  ....aq'.....E
0010 00 34 63 44 40 00 40 06 66 17 ac 10 06 a7 ac 10  4cD@0f.....
0020 12 a1 e1 26 01 f6 6e 8f 32 aa a2 f4 aa 6b 50 18  ..&.n 2....kP
0030 fa ce 71 8f 00 00 00 03 00 00 00 06 ff 03 00 00  ..q.....f
0040 00 04
  
```

tcp.port==502

No.	Time	Source	Destination	Protocol	Length	Info
98	3.394070560	172.16.6.167	172.16.18.161	Modbus...	66	Query: Trans: 3; Unit: 255, Func: 3: Read Holding Registers
99	3.395289983	172.16.18.161	172.16.6.167	Modbus...	71	Response: Trans: 3; Unit: 255, Func: 3: Read Holding Registers
100	3.395384826	172.16.6.167	172.16.18.161	TCP	54	57638 → 502 [ACK] Seq=13 Ack=18 Win=64189 Len=0

Frame 99: 71 bytes on wire (568 bits), 71 bytes captured (568 bits) on interface 0
 Ethernet II, Src: Telemch_0e:61:71 (00:80:f4:0e:61:71), Dst: Raspberr_9d:a2:cc (b8:27:eb:9d:a2:cc)
 Destination: Raspberr_9d:a2:cc (b8:27:eb:9d:a2:cc)
 Source: Telemch_0e:61:71 (00:80:f4:0e:61:71)
 Type: IPv4 (0x0800)
 Internet Protocol Version 4, Src: 172.16.18.161, Dst: 172.16.6.167
 Transmission Control Protocol, Src Port: 502, Dst Port: 57638, Seq: 1, Ack: 13, Len: 17
 Modbus/TCP
 Transaction Identifier: 3
 Protocol Identifier: 0
 Length: 11
 Unit Identifier: 255
 Modbus
 .000 0011 = Function Code: Read Holding Registers (3)
 [Request Frame: 98]
 Byte Count: 8
 Register 0 (UINT16): 1236
 Register 1 (UINT16): 0
 Register 2 (UINT16): 0
 Register 3 (UINT16): 358

```

0000 b8 27 eb 9d a2 cc 00 00 f4 0e 61 71 08 00 45 00  ..'.....aq..E
0010 00 39 61 b3 00 00 40 06 a7 a3 ac 10 12 a1 ac 10  .9a...@f.....
0020 06 a7 01 f6 e1 26 a2 f4 aa 6b 6e 8f 32 b6 50 18  ..&.n 2....kP
0030 11 1c 1a 5c 00 00 00 03 00 00 00 0b ff 03 08 04  ..\.....f
0040 04 00 00 00 00 01 66
  
```