ETUDE ET MISE EN ŒUVRE DU MODBUS ASCII ET RTU

OBJECTIF DE LA FORMATION

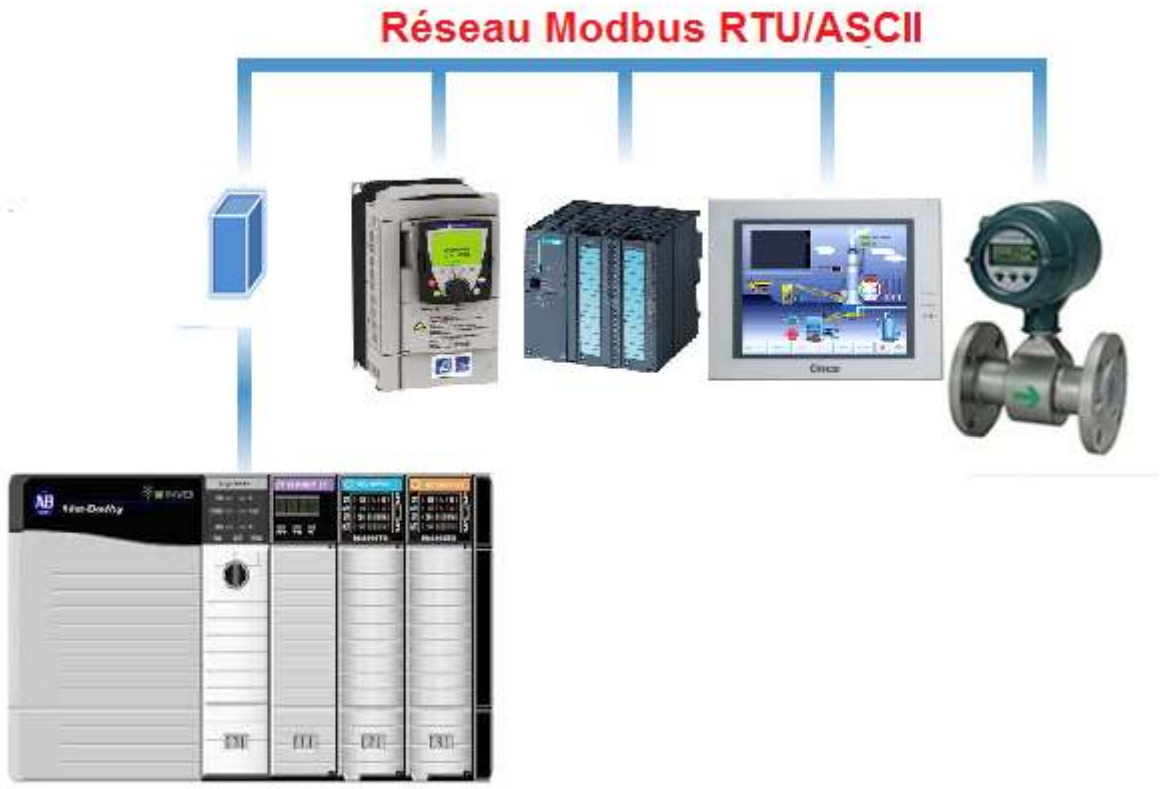
Cette formation a été conçue pour démystifier le bus industriel Modbus qui est très utilisé en automatisme et en informatique industrielle. En effet,il peut être difficile pour un débutant de comprendre le protocole modbus,qui pourtant , est relativement simple à mettre en œuvre si on connait les tenants et les aboutissants de celui-ci.

Cette formation exposera toute la théorie à propos du modbus ASCII et RTU,ce qui vous permettra de mieux les appréhender et de pouvoir par la suite établir des communications basées sur ces protocoles.

A la fin de la formation, vous serez donc en mesure de faire communiquer différents équipements industriels via le protocole modbus ASCII/RTU,de comprendre la constitution des trames modbus ASCII/RTU et de créer des applications en C# permettant de communiquer via modbus.

INTRODUCTION

Le Modbus est un protocole de communication industriel introduit par Modicon en 1979.Il est généralement utilisé avec les automates programmables ou les équipements de types industriels. Il est maintenant devenu une norme "open protocol" dans le domaine de l'automatisme et de la communication industrielle, et est le moyen le plus couramment utilisé pour faire communiquer des équipements industriels.IL existe des versions avec des modifications mineures ou adaptées à d'autres environnements (comme par exemple JBUS ou MODBUS II).



Un des avantages du protocole Modbus est sa flexibilité, mais aussi sa facilité de mise en œuvre. La plupart des appareils et dispositifs embarqués comme les microcontrôleurs, les automates, les capteurs intelligents etc…sont équipés d'interface Modbus et sont capables de communiquer en Modbus. Au début,le Modbus a été initialement conçu pour fonctionner avec les lignes de communication filaires série mais il existe aujourd'hui des extensions à la norme pour les communications sans fil et les réseaux TCP / IP.

Le protocole Modbus permet la communication entre plusieurs équipements connectés sur un même réseau, par exemple un système qui mesure la

température et l'humidité d'un four peut communiquer ses résultats à un ordinateur de traitement via Modbus.

QUELQUES ELEMENTS DE VOCABULAIRES

Les canaux de transmission

Un canal de transmission ou ligne de transmission est une liaison entre deux machines. On désigne généralement le terme émetteur la machine qui envoie les données et récepteur celle qui les reçoit.

Les modes de transmission

Selon le sens des échanges, on distingue 3 modes de transmission :

- **Mode simplex ou unidirectionnel** : il caractérise une liaison dans laquelle les données circulent dans un seul sens,c'est-à-dire de l'émétteur vers le récepteur.
- **Mode half duplex ou bi-directionnel alterné** : caractérise une liaison dans laquelle les données circulent dans un sens ou dans l'autre mais pas les deux en même temps. Ce type de liaison permet d'avoir une liaison bidirectionnelle utilisant la capacité totale de la ligne.
- **Mode full duplex ou duplex intégral** : caractérise une liaison dans laquelle les données circulent de façon directionnelle et simultanée. Chaque extrémité de la ligne peut émettre et recevoir en même temps, ce qui signifie que la bande passante est divisée par deux pour chaque sens d'émission des données si un même support de transmission est utilisé pour les deux transmissions.

Les unités de mesure des vitesses de transmission

Il existe 2 unités pour qualifier la rapidité des échanges :

- Bauds : nombre de bits de données transmis par seconde
- Bits/sec : nombres de bits (quelconques) transmis par seconde

La vitesse de transfert effective est calculée sur les données (on ne tient pas compte des bits de start et de stop pour une communication asynchrone, et des bits de synchronisation pour une communication synchrone).

Le port série

L'échange de données se fait par ligne unique. Les bits sont donc envoyés à la suite. Les ports séries actuels sont bidirectionnels (2 lignes, une par sens de communication).

Notions de communication maître/esclave

Le maître-esclave est un modèle utilisé en technologie, notamment en informatique. Un périphérique, un processus ou un serveur est le maître, l'autre (ou plusieurs autres) est/sont le(s) esclave(s). Le maître donne des ordres à l'esclave qui les exécute.

La notion de protocole

Un protocole est un ensemble de règles strictes, définissant les questions et les réponses devant avoir lieu lorsque deux équipements sont en communication. Ces règles prévoient des procédures de récupération en cas d'erreur de transmission ou de « timeout » (réponse non parvenue dans les délais).

LES MESSAGES DE BROADCAST

Aussi appelé message de diffusion est une communication unidirectionnelle initiée par le maitre et envoyé à tous les esclaves. Ce type de message n’obtient pas de réponse de la part des esclaves, il est utilisé pour envoyer des commandes communes à tous les esclaves par exemple les commandes de configuration ou de réinitialisation.

LES VARIATIONS DU PROTOCOLE MODBUS

Il existe 3 variations du protocole Modbus:

- Le Modbus RTU (8bits)
- Le Modbus ASCII (7 bits)
- Le Modbus TCP/IP (ethernet)

Cette formation s'axera essentiellement sur les variations ASCII et RTU

Modbus RTU VS Modbus ASCII

Une des principales différences entre le modbus ASCII et le modbus RTU est que la communication via modbus RTU est plus rapide.Aussi,le modbus RTU est beaucoup plus populaire que le modbus ASCII qui tend à disparaitre.

LES SUPPORTS PHYSIQUES DE TRANSMISSION DU PROTOCOLE MODBUS

- **Paire torsadée**
- **Radio**
- **Micro-onde**
- **Fibre optique**

LES STANDARDS ELECTRIQUES RS-232/RS-422/RS-485

Les communications Modbus RTU et ASCII peuvent s'effectuer via les standards électriques suivants :

- **RS-232**
- **RS-485**
- **RS-422**

Le **RS232,RS422 et RS485** sont des standards de transmission de données série. Chacune de ces interfaces a des avantages et des inconvénients.

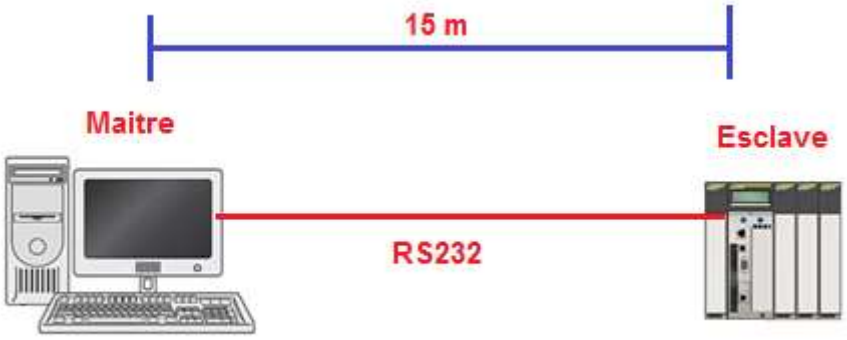
- Le RS232

C'est le plus connu des standards de communication série. Les ports série RS232 sont présents sur la plupart des PCs standards. Il est de type point to point et est composé des lignes **Rx,Tx et GND**.

Le RS232 permet de faire communiquer uniquement un maitre et un esclave sur chaque ligne. Il fonctionne en **full duplex** et sa vitesse de communication peut aller **jusqu'à 115 kbits/s**.

En RS232, la distance séparant les deux équipements ne dépasse pas généralement **15 m**. Si on n'a besoin d'ajouter plusieurs esclaves sur la même ligne, il faudra utiliser les liaisons RS422 ou RS485 qui sont plus adéquates.

Le RS232 a comme inconvénients d'être inadapté dans les environnements où il y'a beaucoup de bruits ou parasites (risque perturbation transmission).

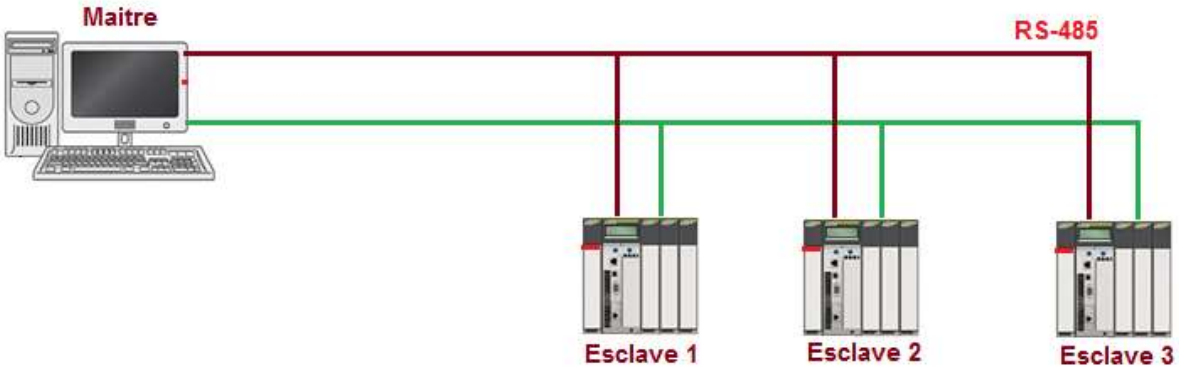


- Le RS422

Il est **full duplex** et est utilisé sur les ordinateurs Apple, sa vitesse de transmission peut aller **jusqu'à 10 Mbits/s**. Les signaux sont envoyés sur 2 fils afin d'augmenter la fréquence de transmission. Il peut supporter **jusqu'à 10 récepteurs par ligne** (on dit alors qu'il est multidrop ou multi-points).

- RS485

Les médias de type RS485 sont souvent en **half duplex** c'est-à-dire la transmission s'effectue via **2 fils**.



Ils permettent de faire communiquer **jusqu'à 32 périphériques sur la même ligne de données** et sur une distance pouvant aller **jusqu'à 1200 m sans répéteurs**.

A noter que l'on peut obtenir du full duplex en utilisant 4 fils de transmission au lieu de 2.Cela permet d'avoir un débit de transmission plus rapide.

Chaque périphérique esclave peut aussi communiquer avec les 32 autres périphériques. Les protocoles de communication RS422 et RS485 sont **multi-drop** c'est à dire plusieurs périphériques peuvent communiquer sur la même ligne de données. Le RS485 a comme avantages d'être immunisé contre les bruits ou parasites.

Configuration des ports série

Lorsque l'on veut établir une communication modbus RTU ou ASCII via les liaisons série RS-232 ou RS-485,il est obligatoire de configurer les ports série entrant en jeu lors de la communication.Un équipement industriel disposant d'un port série est fourni la plupart du temps avec un logiciel permettant de configurer le port afin d'établir une communication.

Les ports séries disposent de 4 paramètres de configuration :

- Baud rate : exprimé en bits/s,désigne la vitesse de la transmission(1200 bits/s,4800 bits/s,9600 bits/s etc..).La valeur 9600 bits/s reste cependant la plus utilisée lors des configurations.

- Nombre de bits de données : désigne le nombre de bits que comporte la trame (7 ou 8 bits)

- Nombre de bits de stop : désigne le nombre de bit de stop que comporte la trame.On peut avoir 1 ou 2 bits de stop.

- Parité : Ce sont des bits ajoutés à une trame pour en vérifier l'intégrité. La vérification de la parité peut être utile lorsque l'on transmet de l'information à haute vitesse ou sur des lignes de qualité médiocre.
  En communication modbus RTU ou ASCII,on a pas besoin d'utilisation les bits de parité car le protocole modbus dispose de son propre système de détection d'erreur.Ainsi,on pourra mettre pour le paramètre parité du port série « None ».Les autres valeurs possibles sont : **Paire** (even), **impaire** (odd).

NB : Pour que des équipements d'un même réseau série puissent communiquer ensemble,ils doivent avoir les mêmes paramètres.

LES ZONES MEMOIRES MODBUS

Au sein de chaque appareil compatible Modbus(automates,variateurs,compteurs etc..), il y'a une partie de la mémoire qui est dédiée au Modbus. Cette partie est appelée **« zone de mémoire Modbus »**

En modbus,la plupart des types de données traditionnels ont une nouvelle dénomination:

- Une sortie physique sur un seul bit est appelé **coils ou bobine**
- Une entrée physique sur un seul bit est appelé **« discrete inputs » ou Inputs contact**
- Les input registers ou analog input
- Les holdings registers

Les coils et « discrete inputs » sont de type booléen alors que les inputs registers(entrées analogiques) et holding registers peuvent comporter des nombres codés sur 2 octets(16 bits) c'est à dire qu'ils sont capables de stocker des nombres compris entre 0 et 65535 .

Les « discrete input » et « input register » sont en lecture seule donc on pourra pas écrire sur ces plages de données.

Nom de la table de données	Adresse utilisée dans la trame	Adresse Modbus
Output coils (R/W)	0000 – 9998	00001 - 09999
Inputs contact (R)	0000 – 9998	10001 - 19999
Input registers (R)	0000 – 9998	30001 - 39999
Holding registers (R/W)	0000 – 9998	40001 - 49999

Si on fait l'analogie par rapport aux automates :

Les coils correspondent aux sorties numériques (ex : vérins)

Les Inputs contact correspondent entrées numériques(ex : boutons poussoirs)

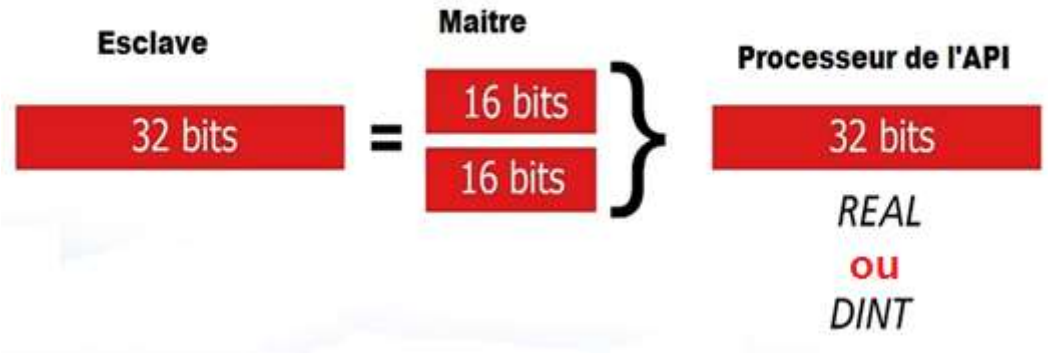
Les inputs registers correspondent aux entrées analogiques (ex :capteur température)

Les holding registers correspondent aux sorties analogiques(ex : variateur de vitesse)

Comme la montre l'image ci-dessus, la plage des « holdings registers » est comprise entre **40001-49999**, cela implique qu'il ne peut y avoir plus de **9999 registres**.

Les registres 40001-49999 correspondent aux adresses de données allant de **0000 à 270E (9998)**.

En modbus, les types float(32 bits) et DInt(32 bits) n'existent pas.Pour envoyer ou recevoir un type float ou Dint,on découpe les données en deux paquées de 16 bits.



LES CODES FONCTION EN MODDUS

Les codes de fonction définissent la commande que le dispositif esclave doit exécuter : **lecture, écriture, diagnostic** etc.. Les codes de fonction vont de **1 à 255**. Certains codes fonction ont des sous code fonction. Cependant, les codes fonction 01,02 , 03 et 04 sont les plus utilisés.

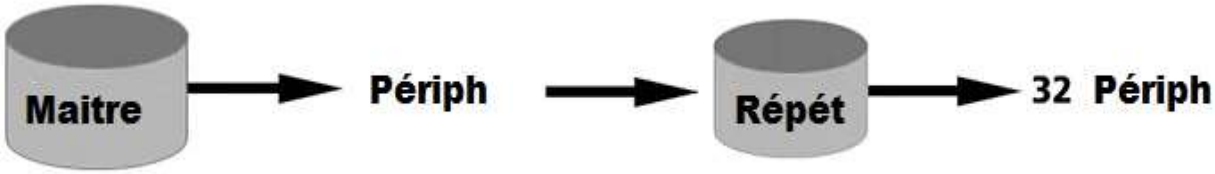
Code d'opération	Action	Description
01 (01 HEX)	Lecture d'un ou de n bits de sortie consécutifs (coils)	Permet de lire la valeur du coil (o ou 1)
02 (02 HEX)	Lecture de n bits d'entrée consécutifs (input contact)	Permet de lire les valeurs qui sont stockées dans les inputs contact
03 (03 HEX)	Lecture de n holding registers	Permet de lire la valeur stockée dans une ou

		plusieurs holding registers
04 (04 HEX)	Lecture de n input registers	Permet de lire les valeurs stockées dans les inputs registers
05 (05 HEX)	Ecriture d'un seul bit de sortie (coils)	Permet d'envoyer 0 ou 1 dans le coil choisi
06 (06 HEX)	Ecriture d'un seul holding register	Permet d'écrire une valeur comprise entre 0 et 65 535 dans un seul holding register
16 (10 HEX)	Ecriture de plusieurs holding registers	Permet d'écrire une valeur comprise entre 0 et 65 535 dans plusieurs holding registers

Le Modbus Unit ID

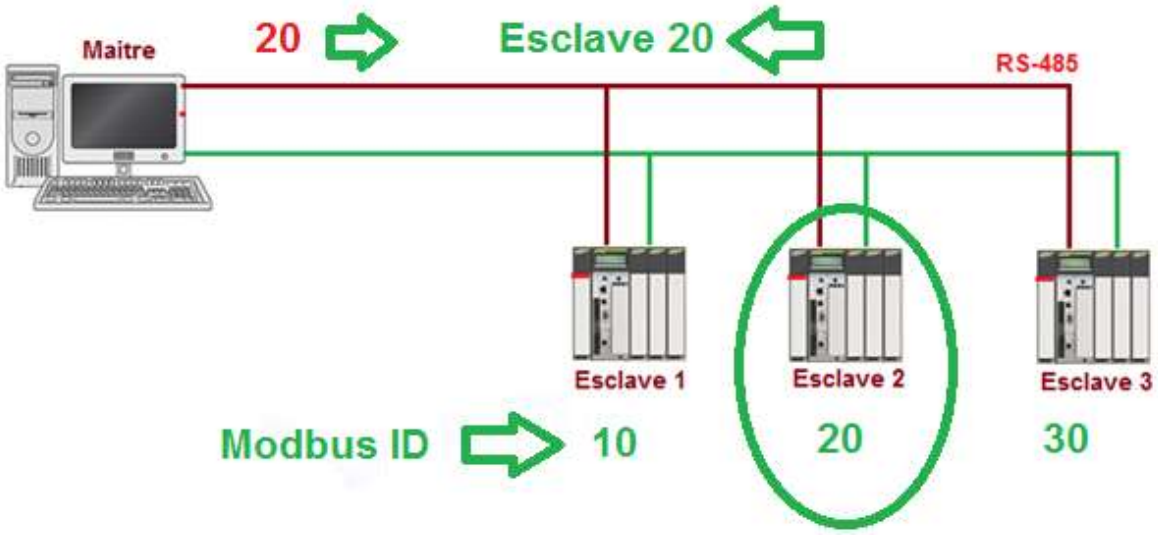
En modbus série,**seul le maître est actif, les esclaves sont complètement passifs.** C'est le maître qui doit lire et écrire dans chaque esclave. Le maitre peut communiquer avec un nombre d'esclaves allant jusqu'à 247 (cas du modbus via RS-485 avec l'utilisation de répéteurs) sur le même réseau. Les adresses allant de **248 à 255 sont des adressées réservées**.

Le RS485 ne peut pas comporter plus de 32 périphériques sur le même nœud, on utilise alors des répéteurs afin de pouvoir ajouter d'autres périphériques sur la ligne.



Le modbus Unit ID permet, dans le cas où on a plusieurs esclaves sur un même réseau de désigner l'esclave que l'on veut interroger.

Généralement,l'équipement compatible modbus est fourni avec un logiciel qui permet de configurer le modbus unit ID de celui.Ainsi,l'Unit ID de l'équipement en question est stockée dans la mémoire de celui-ci.

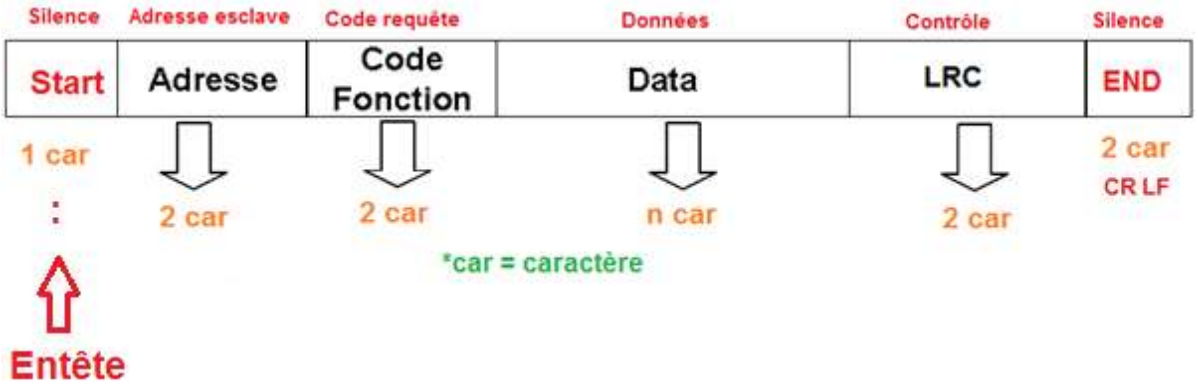

Maitre
20
Esclave 20
RS-485
Esclave 1
Esclave 2
Esclave 3
Modbus ID
10
20
30

LE MODBUS ASCII

La variation Modbus ASCII a été la première variation du protocole Modbus à être implémenté, son encodage est réalisé en hexadécimal et utilise les caractères ASCII.Chaque octet de la trame est représenté sous la forme des codes ASCII des caractères qui les représentent. Par exemple,l'octet 01000010 soit 0x42 sera transmis sous la forme de la chaine '42' correspondant aux codes des chiffres 4 et 2.

En modbus ASCII,une trame est la concaténation des éléments suivants :

- **Entête** : ' **:** ' (3AH)
- **Adresse de l'esclave**
- **Code fonction**
- **Données**
- **LRC**
- **Délimiteur de fin** constitué des caractères Carrier Return(0x0D) et Line Feed (0x0A)



Dans l'exemple ci-dessous, le maitre demande à l'esclave 04 de mettre sa sortie d'adresse 0 à l'état logique « 1 ».

Entête	Adresse esclave	Code fonction	Données	LRC	Délimiteur	
1 carac	2 carac	2 caracr	Nx2 carac	2 carac	Retour chariot	Line feed
:	04	05	0000FF00	F8	CR	LF

En modbus ASCII, on utilise le LRC ou Longitudinal Redundancy Check pour effectuer le contrôle d'intégrité des messages. Le LRC est la somme en

hexadécimal modulo 256 du contenu de la trame hors délimiteurs, complémentée à 2 et transmise en ASCII.

Le calcul pratique du LRC se fait par :

S = (n° esclave + fonction + data) & 256

S = ~ S

S = S + 1

Le contrôle de validité se fait par :

(N° esclave + fonction + data + LRC reçu) = 0x00

Supposons qu'on veuille lire la valeur du mot 800 d'un registre via Modbus ASCII, on aura :

Trame de lecture

Esclave	Fonction	Adresse premier mot	Nombre de mots	LRC
01H	03H	PF/pf	01H (PF/pf)	PF/pf

Trame de réponse de l'esclave

Esclave	Fonction	Nombre octets	Valeur 1er mot	…….	Valeur dernier mot	LRC
01H	03H	1 octet	PF/pf	…….	PF/pf	PF/pf

On aura donc la trame suivante pour la lecture :

- **Numéro d'esclave** : 1
- **Fonction** : 3 (fonction pour lire un holding register)
- **Adresse du mot à lire** : 800 (PF : 03 pf : 20)
- **Nombre de mots à lire** : 1 (PF : 00 pf : 01)
- **LRC** : -31356 (PF : 85 pf : 84)

Dans cet exemple, 800 décimal doit être converti en HEXA sur 2 octets Poids Fort puis poids faible. Il faut savoir qu'un PC parle intuitivement en HEXA. C'est à dire que si on prend le décimal 65 qui a pour équivalent caractère 'A', quand on va passer chr(65) sur la ligne, on transmettra 'A' qui est en fait 41H. De la même manière, quand on passera le décimal 0 qui a pour équivalent caractère NUL, on transmettra NUL qui est en fait 00H.

La trame qui sera envoyée est la suivante :

01 03 0320 0001 8584 (se munir d'une table de conversion ASCII/DEC/HEX).

01 fait 2 caractères on enverra chr(01) qui est le caractère SOH donc l'HEXA 01H et ainsi de suite...

Par conséquent, la trame définitive qui sera transmise est la suivante

:chr(01)+chr(03)+chr(03)+chr(20)+chr(00)+chr(01)+chr(85)+chr(84)

LE MODBUS RTU (Remote Terminal Unit)

La communication Modbus RTU est de type série et se fait via les interfaces série RS232, RS485 ou RS422.Le codage des informations s'effectue en binaire.Le modbus RTU fait partie des protocoles industriels les plus utilisés.

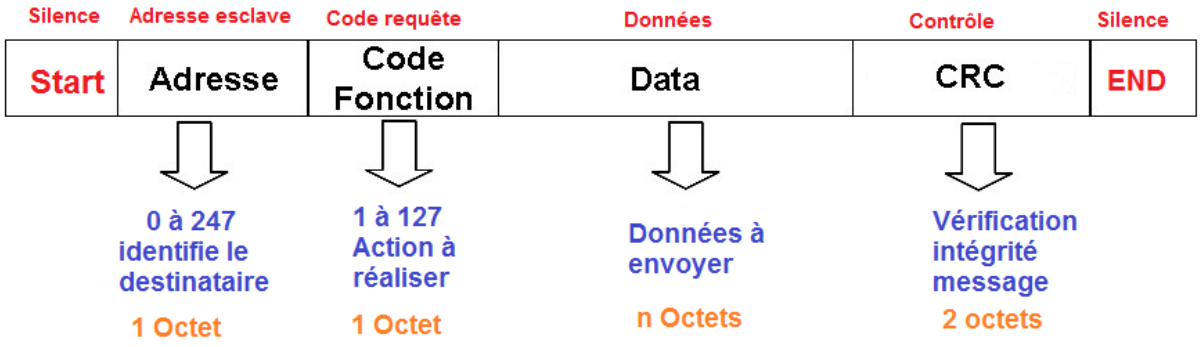
Si la communication s'effectue via le RS232, il ne peut y avoir dans ce cas qu'un seul maitre et qu'un seul esclave. Par contre si la communication s'effectue via le RS485 ou le RS422, on peut avoir plusieurs esclaves.

NB : En modbus RTU on peut pas avoir plusieurs maitres.Le mode de fonctionnement multi-maitre n'est possible qu'avec le modus TCP/IP

L'avantage du mode RTU est que les données à transmettre prennent moins de place donc moins de temps pendant les transmissions. En effet, on adresse plus de données en 8 qu'en 7 bits.

La trame du MODBUS RTU est constituée d'une suite de caractères hexadécimaux et contient les informations suivantes :

- Numéro d'esclave (1 octet) (le numéro 00 est réservé aux messages de diffusion)
- Code fonction (1 octet)
- Données (n octets)
- CRC (2 octets)



Chaque octet composant une trame RTU est codé sur 2 caractères hexadécimaux (2 fois 4 bits)

La taille maximale des données est de 256 octets. L'ensemble des informations contenues dans le message est exprimé en hexadécimal.

Chaque octet composant un message est transmis en mode RTU de la manière suivante :

Sans contrôle de la parité :

Start	B0	B1	B2	B3	B4	B5	B6	B7	Stop	Stop

Avec contrôle de la parité

Start	B0	B1	B2	B3	B4	B5	B6	B7	Parité	Stop

Dans le cas d'un contrôle de parité, il vous est demandé de confirmer l'état du contrôle : paire (even) ou impaire(odd).

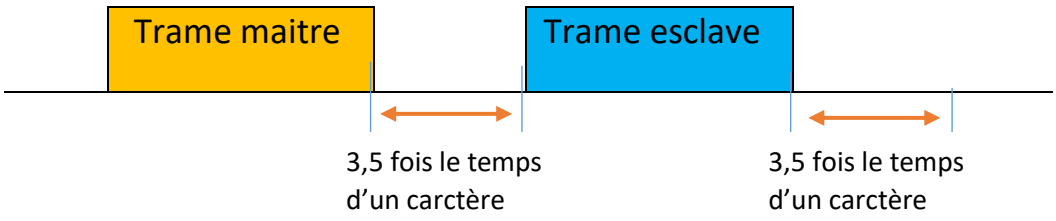
Avant et après chaque message(trame), il doit y avoir un silence minimum de 3,5 fois le temps de transmission d'un caractère. L'ensemble du message doit être transmis de manière continue.

Ainsi,l'équipement détecte le début d'un message quand il reçoit un caractère valide (contenant son adresse ou l'adresse 00) dans un intervalle de temps d'au moins 3,5 fois la longueur d'un caractère.

Si le débit de transmission est 9600 bits/s, on aura : 3,5 caractère (3,5 * 11 * (1/9600))

Le temps maximum entre 2 caractères doit être inférieur à 1,5 fois le temps de transmission d'un caractère. Dans le cas contraire, il y a une erreur de transmission.

NB : 1 caractère est un format de 11 bits constitué de : 1 bit de start, 8 bits de données et 2 bit de stop **(ou 1 bit parité + 1 bit stop)**



La nature des informations de la trame peut varier selon que l'on fera de la lecture/écriture, de mots, de bits ....

Calcul du CRC d'une trame modbus RTU

Le CRC est une technique utilisée pour assurer une fiabilité proche de 100%. Il est donc superflu d'utiliser les contrôles de flux et de parité. CRC signifie (cyclical Redundancy check) ou test de redondance cyclique et permet de vérifier s'il n'y en a pas d'erreur au niveau de la trame.Il est généré par le maitre.

Le CRC calculé sur 16 bits fait partie intégrante du message et il est vérifié par le destinataire. Il est calculé sur tous les octets de la trame à part lui-même bien-entendu.Lors d'une transmission,si le CRC est incorrecte, le périphérique esclave redemande au maitre de renvoyer le message.

Calcul de CRC en En C#:

```
"…
uint crc16(byte [ ] buf , uint len)
{
    uint POLY=0xA001;
    int crc;
    uint i;
    uint j;

     j = 0;
     for (crc = 0xFFFF; len != 0; len --){
          crc ^= buf[j];
          j++;
          for (i = 0; i < 8; i ++){
             if ((crc & 0x0001)>0){
                crc = (crc >> 1) ^ POLY;
              }else{
                crc >>= 1;
              }
           }
       }
       return (crc);
}
…"
```

Différence entre trames modbus RTU et ASCII

	Modbus ASCII	Modbus RTU
Codage	ASCII '0'….'9', 'A'…..'F'	Binaire 0…..255
Gestion erreur	LRC	CRC

Début trame	Caractère ‘ :’	3,5 fois le tps de transmission d’un caractère
Fin trame	CR/LF	3,5 fois le tps de transmission d’un caractère
Temps max entre caractère	1 s	1,5 fois le tps de transmission d’un caractère
Bit de start	1	1
Bits de données	7	8
Parité	Paire/impaire/aucun	Paire/impaire/aucun
Bits de parité	1 si y’a parité , 2 si y’a pas	1 si y’a parité , 2 si y’a pas

NB : 1 caractère est un format de 11 bits constitué de : 1 bit de start, 8 bits de données et 2 bit de stop (ou 1 bit de parité et 1 bit de stop)

Ci-dessous, un exemple de trames de requête en ASCII et RTU demandant la lecture de 3 registres d’un esclave avec leur trame réponse correspondante.

Trame de requête de 3 registres :

	Exemple (HEX)	Caractères ASCII	8 bits mode RTU
Entête		‘ :’	Aucun
Adresse de l’esclave	06	‘06’	0000 0110
Code fonction	03	‘03’	0000 0011
Adresse de début PF	00	‘00’	0000 0000
Adresse de début pf	6B	‘6B’	0110 1011
Nombre de de registres PF	00	‘00’	0000 0000
Nombre de registre pf	03	‘03’	0000 0011
Contrôle erreur		LRC (2 carac)	CRC (16 bits)
Fin trame		CR LF	Aucun
Total		17 octets	8 octets

Trame de réponse de l'esclave

	Exemple (HEX)	Caractères ASCII	8 bits mode RTU
Entête		' :'	Aucun
Adresse de l'esclave	06	'06'	0000 0110
Code fonction	03	'03'	0000 0011
Nombre d'octets de données	06	'06'	0000 0110
Données 0 PF	02	'02'	0000 0010
Données 0 pf	2B	'2B'	0010 1011
Données 1 PF	00	'00'	0000 0000
Données 1 pf	00	00	0000 0000
Données 2 PF	00	00	0000 0000
Données 2 pf	63	63	0110 0011
Contrôle erreur		LRC (2 carac)	CRC (16 bits)
Fin trame		CR LF	Aucun
Total		23 octets	11 octets

Exemple de trames de communications modbus RTU

Exemple # 1. Lecture d'un coil d'un esclave

Supposons que nous voulions savoir quelle trame d'information doit envoyer le maitre à l'esclave afin de connaitre la valeur du **coil n ° 5** d'un esclave dont l'adresse est **#240**.

Constitution de la trame requête :

- **Premier élément : L'adresse de l'esclave**

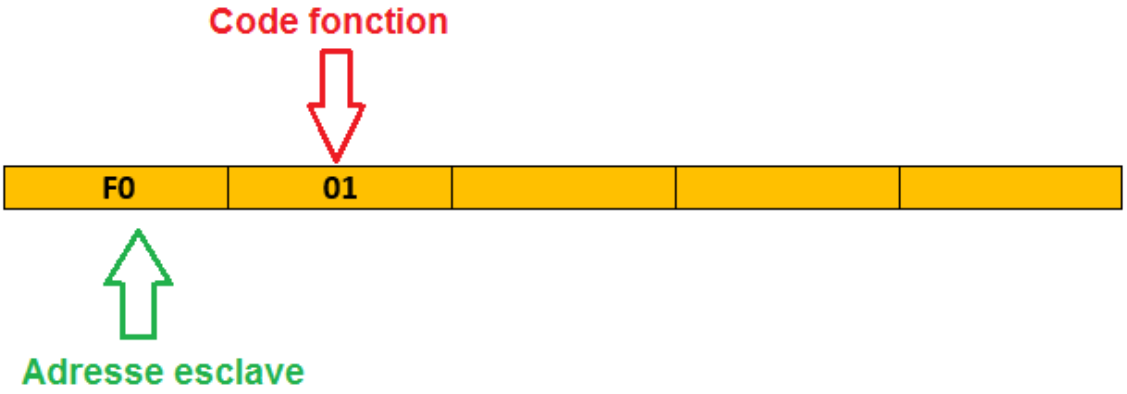
La première donnée constituant la trame MODBUS est l'adresse ou le numéro de l'esclave. En effet la requête est envoyée à tous les esclaves mais seulement l'un d'eux va répondre. L'esclave qui répond c'est tout simplement celui dont l'adresse est précisée dans la trame. Le numéro d'esclave est un nombre compris entre 1 et 247 (La valeur 0 étant réservée aux messages de broadcast ou de diffusion), de sorte qu'il ne nécessite qu'un seul octet pour stocker ces données.

Pour notre exemple, si vous voulez envoyer des données au numéro d'esclave 240, la trame d'information va donc commencer par le numéro de l'esclave qui dans notre cas est 240 (ou F0 en hexadécimal).

Deuxième élément: Le code d'opération ou code fonction modbus

Il est codé sur un seul octet et permet de dire à l'esclave quelle fonction ou opération il doit effectuer.

Pour notre exemple, nous voulons connaître la valeur du coil n ° 5, l'action à effectuer est donc **une lecture d'une seule bobine ou coil**, le code fonction est donc 01 HEX, comme on peut le voir dans les tableaux précédents. Ainsi, le paquet de trames d'informations ou de données MODBUS est formé comme indiqué dans la figure suivante:



Troisième élément : Adresse du coil ou bobine à lire

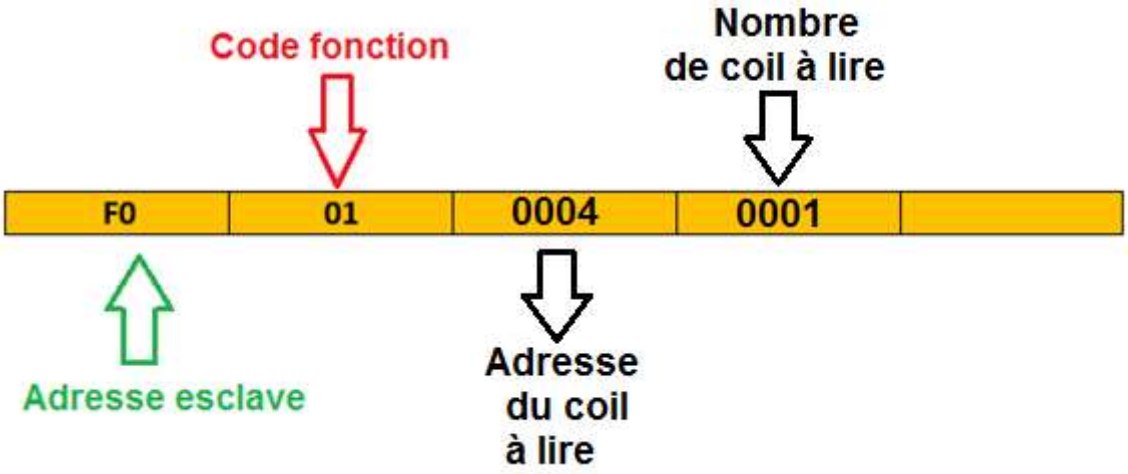
Le code fonction 01 a besoin des éléments adresse à lire et nombre de données à lire.Ainsi, on doit préciser l'adresse du coil ou de la bobine à lire, il doit être un nombre de 2 octets, car l'adresse est un nombre compris entre 0 et 9998, pour notre exemple, on veut lire la bobine numéro 5 cependant les adresses de début des bobines démarrent à 0 comme représenté dans l'image ci-dessous.

Adresse Modbus	Adresse utilisée dans la trame	Nom table données
1 - 9999	0000 – 9998	Output Coils
[illegible]	[illegible]	Inputs Contac
30000 - 39999	0000 – 9998	Inputs Regist
40001 - 49999	0000 – 9998	Holding Regis

L'adresse MODBUS est celle de la bobine que l'on veut lire, pour notre exemple c'est la bobine n ° 5, mais l'adresse réelle utilisée est celle de la colonne suivante(colonne intitulée **« adresse utilisée dans la trame »**) qui comme mentionné sur l'image ci-dessus est un nombre compris entre 0 et 9998.En regardant l'image,on peut remarquer que l'adresse modbus du coil numéro 1 correspond à l'adresse trame **0000** , on peut donc en déduire que l'adresse trame correspondant à l'adresse modbus du coil numéro 5 est 0004.On soustrait tout simplement 1 unité.0004 se code sur deux octets en hexadécimal.

Quatrième élément: Nombre de bobines à lire

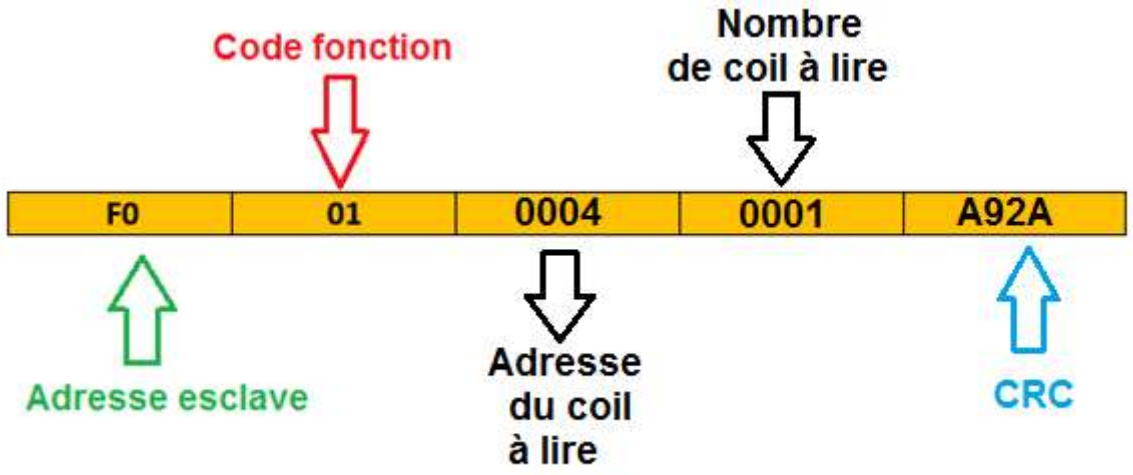
On doit aussi préciser à la fonction 01 HEX le nombre de données (ici coils) que l'on veut lire.La fonction 01 permet de lire de 1 à un total de 9998 bobines comme on peut le voir dans les tableaux précédents. Pour notre exemple, seule la valeur de la bobine n ° 5 est demandée, ainsi le nombre de bobine à lire est 1, converti au format deux octets on obtient : 00001. Ainsi on aura la trame suivante :



Cinquième élément : code de détection d'erreur: CRC

Le cinquième et dernier élément à ajouter à la trame est le CRC (Cyclic Redundancy Check, Cyclic Redundancy Check).C'est un nombre à deux octets qui est utilisé pour détecter les erreurs.Le maître génère le CRC correspondant à la bonne information,l'esclave calcule à son tour son CRC en fonction des données qui lui parviennent.Si les données lui parviennent correctement, le CRC calculé par celui-ci doit être égal au CRC envoyé par le maître.Si ces deux CRC ne sont pas égaux, cela entraine une erreur d'intégrité des données,prouvant que l'esclave n'a pas reçu les données de manière correcte.Ceci est un des avantages du MODBUS, car cette détection d'erreur permet de vérifier que les données reçues sont bien correctes.

Pour notre exemple,le CRC est égal à A92A (hex).Pour calculer le CRC,vous pouvez utiliser le logiciel **modbus poll ou d'autres logiciels**.



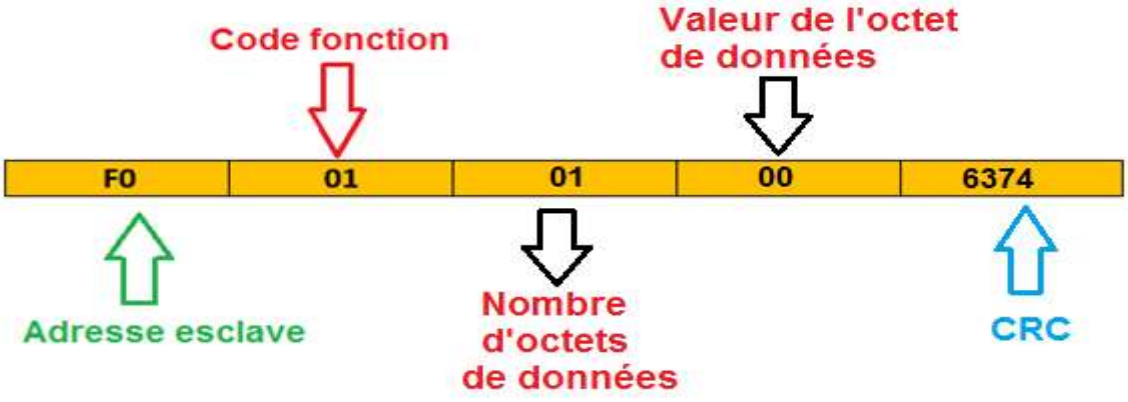
Constituions de la trame réponse de l'esclave

Une fois que l'esclave ait reçu la requête du maitre,il va envoyer à son tour une trame réponse au maitre.

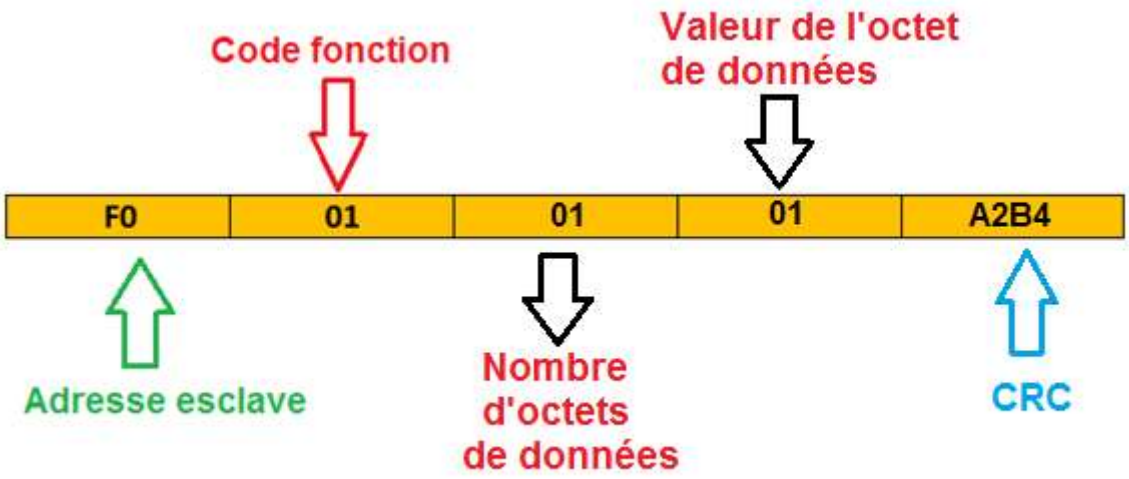
Lorsque le maitre effectue sa requête, dans notre cas une requête avec fonction de lecture 01 HEX, l'esclave se doit de répondre en envoyant la valeur contenu dans la bobine numéro 5.Etant donné qu'on a une bobine (un bit),la valeur qu'elle contient ne peut qu'être 0 ou 1.

En effet,en fonction de la valeur(0 ou 1) du contenu de la bobine numéro 5,on aura des trames différentes. Mais quoi qu'il en soit, la trame renvoyée par l'esclave commence, comme pour la trame maitre avec un octet désignant le numéro de l'esclave, un autre pour désigner le code fonction, des octets pour les données et enfin un code CRC.

Réponse de l'esclave au maître en supposant que la valeur contenu dans la bobine numéro 5 est 0



Réponse de l'esclave au maître en supposant que la valeur contenu dans la bobine numéro 5 est 1



Dans les deux cas, la réponse est à peu près la même, sauf pour la valeur de l'octet de données et pour le CRC.

Remarques 1

Si une erreur apparaît sur la requête du maitre, l'esclave envoie via sa trame de réponse un code d'exception en mettant à 1 le bit de poids fort du code de fonction (0x80).Avec ce bit le maitre sait qu'il y'a eu une erreur, mais pour avoir plus de détails sur le type d'erreur il doit vérifier le champ de données envoyé par l'esclave:

Code	Nom	Signification
01	Illegal Function	Le code de fonction reçu ne correspond pas à une commande disponible sur l'esclave
02	Illegal Data Adress	L'adresse indiquée dans la trame est invalide
03	Illegal Data Value	La valeur envoyée à l'esclave est invalide
04	Slave Device Failure	L'esclave a reçu la trame ,a commencé le traitement, mais

		une erreur est survenue,il ne peut pas terminer la tâche
05	Acknowlege	L'esclave a reçu la trame ,a commencé le traitement, mais cela va prendre un peu de temps. Cette réponse empêche le maitre de considérer cela comme un timeout. Le maître peut envoyer une trame de polling plus tard pour vérifier si la tâche est complète
06	Slave Device Busy	L'esclave est en train d'effectuer une autre tâche et ne peut pas répondre à cette demande alors, le maître devra réessayer plus tard

Remarques 2

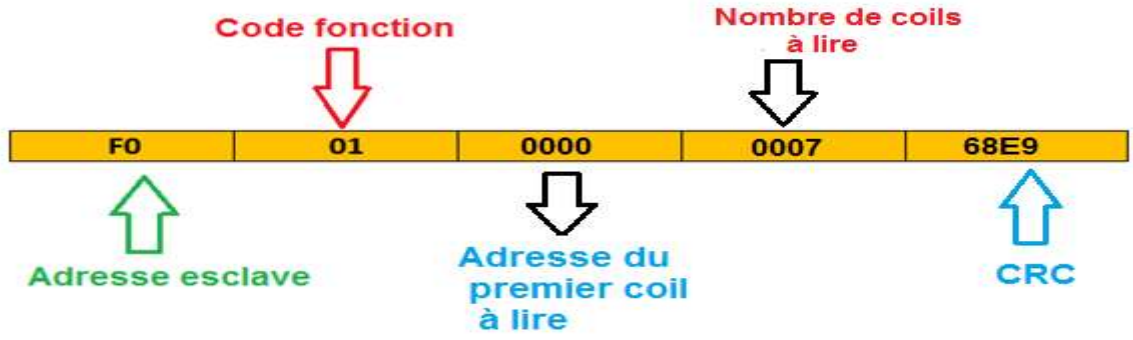
Mis à part les messages de broadcast ou diffusion, toutes les requêtes initiées par le maitre reçoivent une réponse de la part de l'esclave.

Exemple # 2. La lecture de plusieurs bobines d'esclaves

Dans ce cas ci, la même fonction "01" servira à lire l'état de plusieurs bobines ou coils.

Supposons que nous voulions savoir quelle trame doit envoyer le maitre afin de connaître la valeur des bobines # 1, # 2, # 3, # 4, # 5, # 6, et # 7 d'un esclave dont l'adresse est # 240.Ci-dessous la requête que doit envoyer le maitre :

Trame requête maitre

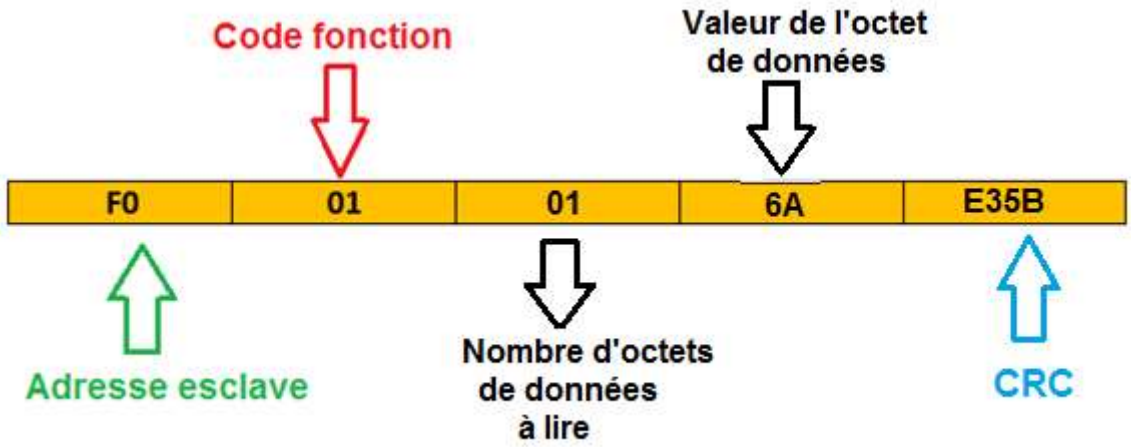


La première bobine à lire est la bobine 1 c'est la raison pour laquelle l'adresse de trame est **0000**, le nombre total de bobines à lire est de 7 c'est la raison pour laquelle on a **0007**.

Trame réponse de l'esclave

On peut avoir plusieurs réponses du fait que les coils peuvent prendre 0 ou 1 comme valeur. On va prendre comme exemple le cas de figure suivant :

Supposons que les bobines # 1 = 0, # 2 = 1, n ° 3 = 0, # 4 = 1, # 5 = 0, # 6 = 1, n ° 7 = 1, dans ce cas, la réponse de l'esclave est telle qu'indiquée dans l'image suivante:



Deux points à noter, le nombre d'octets de données est 1, car un octet peut représenter 7 bobines, si nous avions par exemple demandé l'état de 15

bobines, l'octet de données serait 02 puisque on a besoin de deux octets pour coder 15 bobines (15 bits).

D'autre part on a 6A comme valeur de l'octet de données. Cette valeur est obtenue comme dans l'image ci-dessous.

Coil 7	Coil 6	Coil 5	Coil 4	Coil 3	Coil 2	Coil 1
1	1	0	1	0	1	0

6 **A**

Quelques outils logiciels utiles

Modbus PLC Simulator : il permet de simuler une liaison Modbus pour des fins de conception ou de tests. Il supporte le modbus série et le modbus TCP/IP.

Free serial port monitor : il permet de superviser les données transitant sur port série

Com0Com : il permet de créer des ports com virtuels

Wireshark : il permet de disséquer un réseau TCP/IP

Nmodbus4 : c'est une bibliothèque dotnet permettant d'établir des communications modbus RTU et TCP/IP en utilisant les langages comme visual Basic et C#.

Virtual serial port :

CONCLUSION

Le protocole Modbus offre l'avantage d'être disponible en natif sur la plupart des équipements industriels. En choisissant le Modbus comme protocole de communication, vous avez une bonne chance d'éviter des problèmes de compatibilité liées aux interfaces de communication.

En communication Modbus, la liaison RS232 est utilisée pour les courtes distances alors que le RS485 permet de couvrir des distances plus longues.

Le RS422 a les mêmes spécificités que le RS485, mais il fonctionne en full duplex c'est-à-dire que vous pouvez envoyer et transmettre des données en même temps.

Pour optimiser les transmissions ou éliminer la nécessité d'utilisation de câble, d'autres supports physiques de transmission peuvent être utilisé : fibre optique, radio, micro-onde etc…