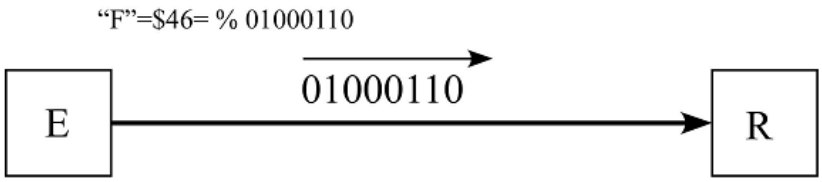
2.1 Définition

C'est la transmission d'une donnée bit après bit sur une même ligne, elle est surtout adaptée aux grandes distances, elle est également utilisée sur courtes distances pour des transferts assez lents (petits blocs de données).

Illustration :



On transmet toujours le bit de poids le plus faible le premier.

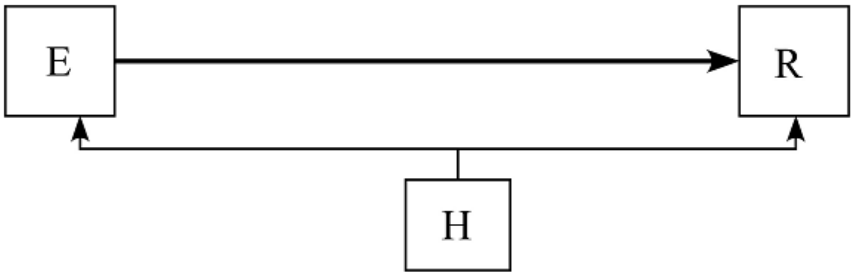
Avantage : diminution du nombre de connexions (1 fil pour l'émission, 1 fil pour la réception).

Inconvénient : vitesse de transmission plus faible que pour une interface parallèle.

Il existe deux types de transmissions séries :

2.2 Transmission synchrone :

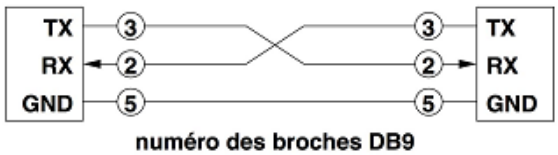
Les octets successifs sont transmis par blocs séparés par des octets de synchronisation. Dans ce mode de transmission, l'émission des données ainsi que leur réception se fait en synchronisation sur la même horloge.

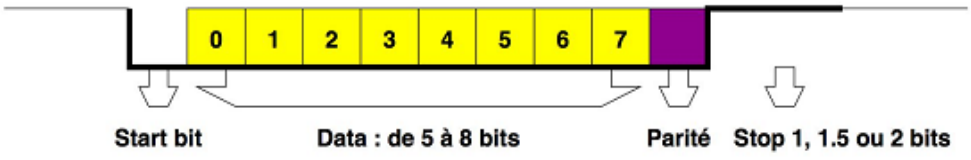


2.3 Transmission asynchrone

Dans ce mode de transmission, le transfert des données n'est pas cadencé par une même horloge (**H**), ce mode est le plus utilisé car dans la plus part des applications l'émetteur et le récepteur sont cadencés (rythmés) chacun par sa propre horloge.

RS232





Protocole de transmission :

Afin que les éléments communicants puissent se comprendre, il est nécessaire d'établir un protocole de transmission. Ce protocole devra être le même pour les deux éléments afin que la transmission fonctionne correctement.
Paramètres rentrant en jeu :

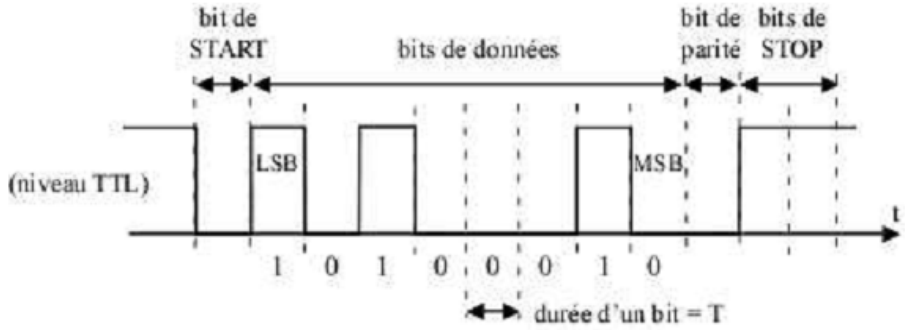
- **Longueur des mots :** 7 bits (ex : caractère ascii) ou 8 bits

- **La vitesse de transmission :** les différentes vitesses de transmission son réglables à partir de 110 bauds (bits par seconde) de la façon suivante : 110 bds, 150 bds, 300 bds, 600 bds, 1200 bds, 2400 bds, 4800 bds, 9600 bds.

- **Parité** : le mot transmis peut être suivi ou non d'un bit de parité qui sert à détecter les erreurs éventuelles de transmission. Il existe deux types de parité.
  parité paire : le bit ajouté à la donnée est positionné de telle façon que le nombre des états 1 soit paire sur l'ensemble donné + bit de parité
  ex : soit la donnée 11001011 contenant 5 état 1, le bit de parité paire est positionné à 1, ramenant ainsi le nombre de 1 à 6.
  parité impaire : le bit ajouté à la donnée est positionné de telle façon que le nombre des états 1 soit impaire sur l'ensemble donné + bit de parité
  ex : soit la donnée 11001001 contenant 5 état 1, le bit de parité paire est positionné à 0, laissant ainsi un nombre de 1 impaire..
- **Bit de start :** la ligne au repos est à l'état logique 1 pour indiquer qu'un mot va être transmis la ligne passe à l'état bas avant de commencer le transfert. Ce bit permet de synchroniser l'horloge du récepteur.

- **Bit de stop :** après la transmission, la ligne est positionnée au repos pendant 1, 2 ou 1,5 périodes d'horloge selon le nombre de bits de stop.

Exemple : transmission du caractère '***E***' (code ASCII $45 = %01000101) sous forme série :



- Niveau TTL : 0 L → 0v et 1 L → +5v ; la ligne est au début au repos (niveau haut (1))
- le bit ***START*** marque le début de la transmission du caractère ;
- les bits de données sont transmis l'un après l'autre en commençant par le bit de poids faible. Ils peuvent être au nombre de 5, 6, 7 ou 8. Chaque bit est maintenu sur la ligne pendant une durée déterminée T. L'inverse de cette durée définit la fréquence de bit = nombre de bits par secondes = vitesse de transmission ;
- le bit de parité (***facultatif***) est un bit supplémentaire dont la valeur dépend du nombre de bits de données égaux à 1. Il est utilisé pour la détection d'erreurs de transmission ;
- les bits ***STOP*** (1, 1.5 ou 2) marquent la fin de la transmission du caractère.

I2C caractéristiques principales

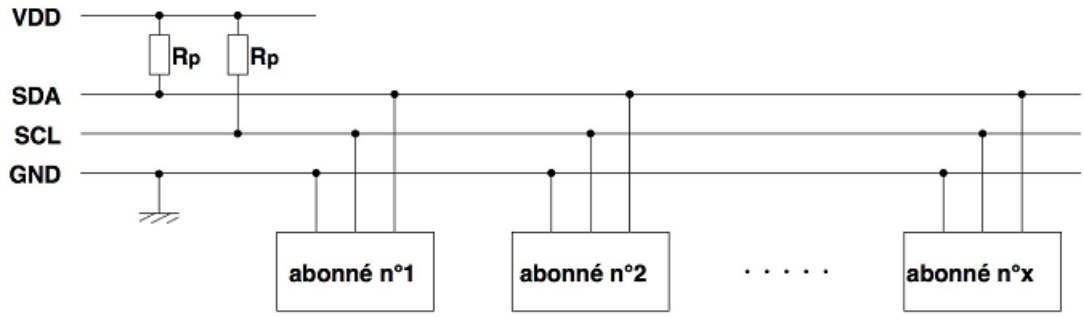
- I2C = IIC = Inter Integrated Circuit
- Protocole défini dans les années 80 par Philips.
- Protocole simple et très diffusé.
- Jusqu'à 128 abonnés (version de base) communiquent sur 3 fils :
  - SCL (horloge) .
  - SDA (data) .
  - GND (tension de référence).

I2C Glossaire

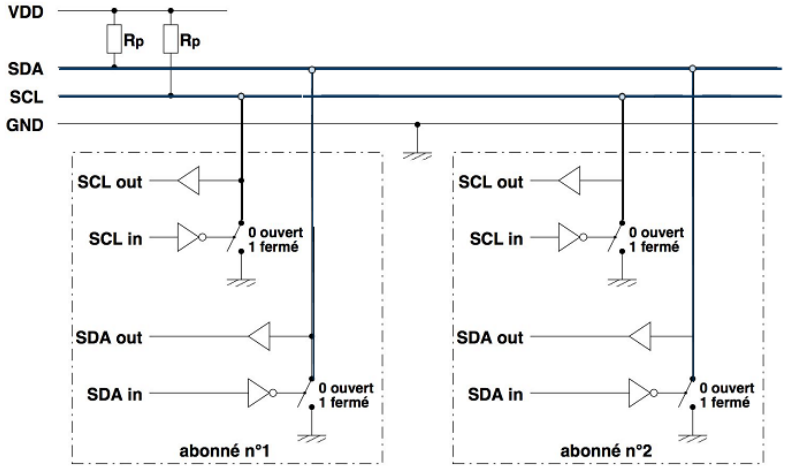
abonné	tout élément connecté sur le bus.
émetteur	tout abonné qui envoie des données sur SDA.
récepteur	tout abonné qui reçoit des données de SDA.
maître	tout abonné qui démarre et termine un échange. Le maître place l'horloge sur SCL.
esclave	tout abonné adressé par un maître. Un esclave à la possibilité de ralentir l'horloge du maitre.
adresse	numéro attribué à un esclave. Sur le bus tous les esclaves ont une adresse unique.
échange	diablogue entre un maitre et un esclave. Il commence par une adresse émise par le maitre, suivie d'une ou plusieurs données émises par le maitre ou l'esclave. Un maitre peut chainer plusieurs échanges d'affilé.
arbitrage	résolution du conflit d'un accès simultané par 2 maîtres.

I2C cablage

- Les lignes SCL et SDA sont à VDD si aucun abonné ne parle.
- Pour mettre 1 sur SCL ou SDA, un abonné programme le port en entrée. la résitance Rp se charge de tirer la ligne à 1
- Pour mettre 0 sur SCL ou SDA, un abonné doit écrire un 0. c.-à-d. relier la ligne à la masse.
- Il ne peut jamais y avoir de conflit électrique (court-circuit VDD-GND).



I2C schéma de principe



I2C principe d'un échange

Le maître :

- émet une condition de démarrage
- envoie une adresse sur 7 bits
- envoie la commande r/w
- lit l'accué et stoppe si NACK
- pour une écriture, il boucle sur
  - envoie les 8 bits de donnée
  - lit l'accusé et stoppe si NACK
- pour une lecture, il boucle sur
  - lit les 8 bits de donnée
  - émet ACK, ou NACK pour stopper
- émet une condition de stop

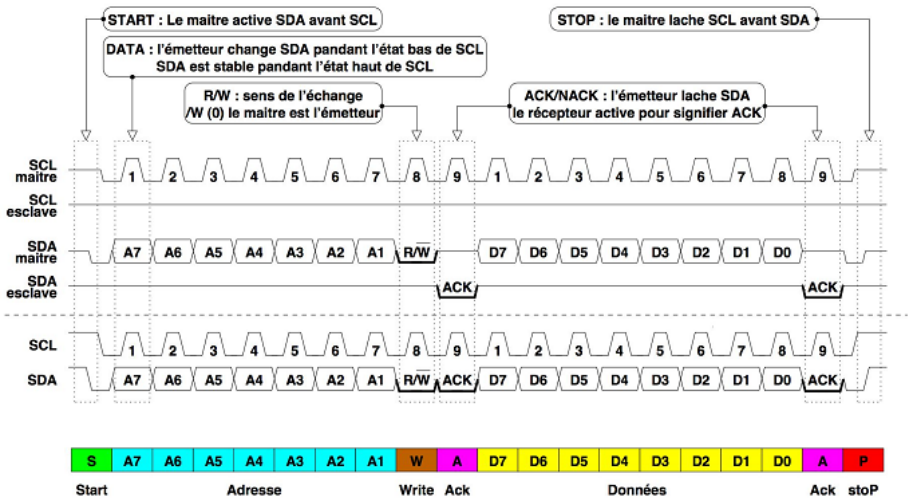
L'esclave :

- attend une condition de démarrage
- lit l'adresse sur 7 bits
- lit la commande r/w
- émet ACK si concerné
- pour une écriture, il boucle sur
  - lit les 8 bits de donnée
  - met ACK ou NACK pour arrêter
- pour une lecture, il boucle sur
  - écrit les 8 bits de donnée
  - lit l'accusé et stoppe si NACK

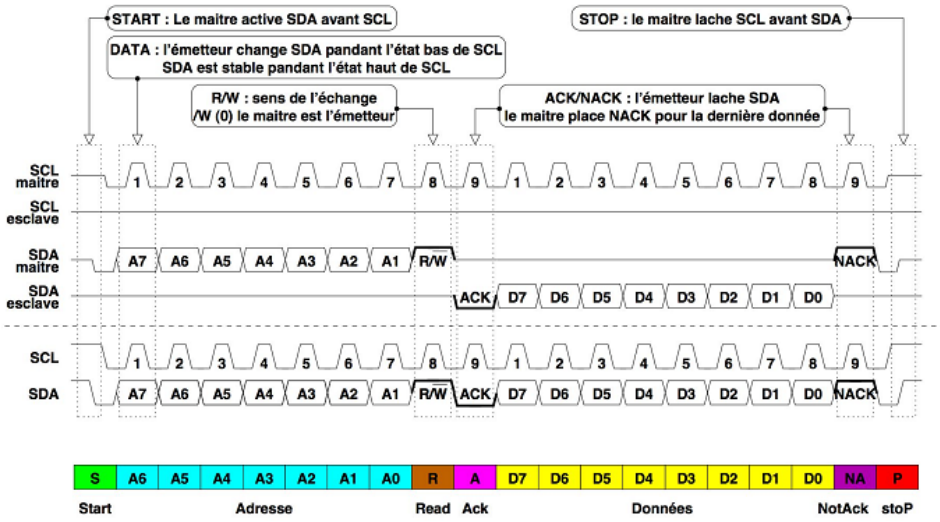
le maitre et l'esclave peuvent ralentir l'échange en jouant sur SCL

- C'est le maître qui pilote l'horloge SCL en la mettant à 0 et à 1
- Si l'esclave force l'horloge à 0, le maitre ne peut plus la mettre à 1
  Le maître comprend qu'il doit attendre que l'esclave "relâche" l'horloge

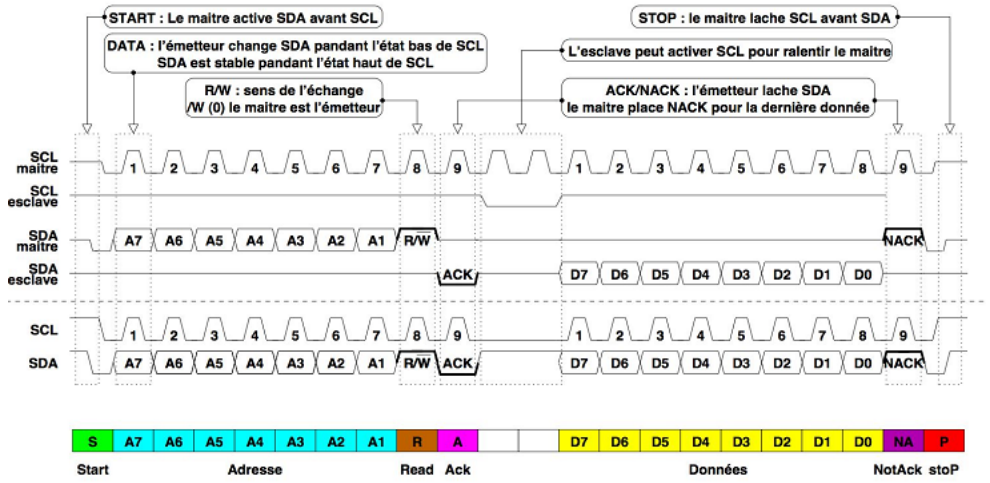
I2C trame de base : écriture d'un octet



I2C trame de base : lecture d'un octet

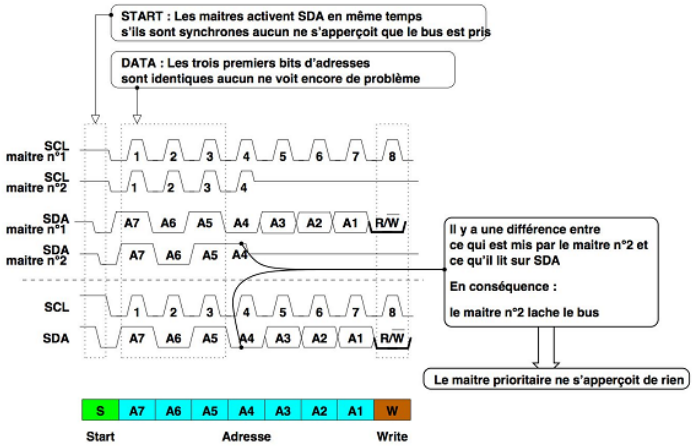


I2C trame de base : état d'attente



Arbitrage entre maîtres

Si deux maitres tentent de démarrer un échange simultanément :
le premier qui dit 1 sur SDA a perdu.



LIAISON SERIE MODBUS RS485

1.1. Généralités :

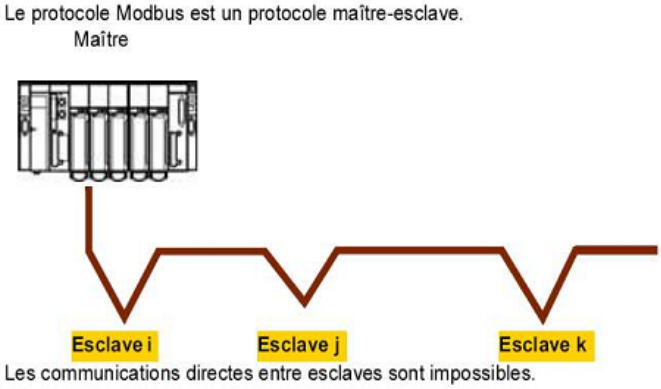
Le bus Modbus répond aux architectures Maître/Esclave. Il a été créé par la société Modicon pour interconnecter les automates programmables. Ce protocole a rencontré beaucoup de succès depuis sa création du fait de sa simplicité et de sa bonne fiabilité.

1.2. Principe général :

Le bus est composé d'une station Maître et de stations esclaves. Seule la station Maître peut être à l'initiative de l'échange (la communication directe entre stations Esclaves n'est pas réalisable). Le maître peut s'adresser aux esclaves individuellement ou envoyer un message de diffusion générale à tous les esclaves. Les esclaves renvoient un message (réponse) aux requêtes qui leur sont adressées individuellement. Les requêtes de diffusion générale n'attendent pas de réponses en retour.

Principe

Le protocole Modbus est un protocole maître-esclave.



Les communications directes entre esclaves sont impossibles.

Un seul équipement peut émettre sur la ligne à tout moment.
Le maître gère les échanges et lui seul peut prendre l'initiative.
Il interroge chacun des esclaves successivement.
Aucun esclave ne peut envoyer de message à moins qu'il ne soit invité à le faire.
Le maître répète la question lorsqu'un échange est incorrect et déclare l'esclave interrogé absent si aucune réponse n'est reçue dans un délai donné.
Si un esclave ne comprend pas un message, il envoie une réponse d'exception au maître. Le maître peut réitérer ou non la requête.

Le bus Modbus RS485 peut accueillir 32 nœuds : 1 maître et jusqu'à 31 esclaves.

Différences majeures avec la norme RS232 :

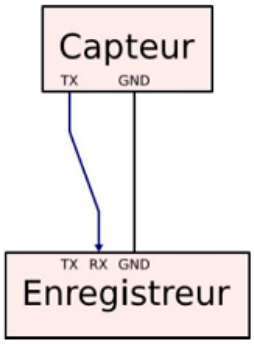
La norme RS485, contrairement à la norme RS232, ne définit que les caractéristiques électriques de la couche physique. Les principales différences sont le medium de communication (une paire torsadée), un mode de tensions différentielles, et la possibilité de travailler en réseau (et non en mode point à point).

	RS232	**RS485**
Connexion	Point à point	Bus
Emetteurs / récepteurs	1 / 1	32 / 32
Couplage électrique	Mode asymétrique	Mode symétrique (différentiel)
Support physique	2 fils de données + masse	1 paire torsadée
Type de liaison	Full duplex	Half duplex
Débit maximum	20 kb/s	10 Mb/s
Portée typique	10 m	1 km

LES TYPES DE TRANSMISSIONS :

3.1. Transmission simplex : mono-directionnel

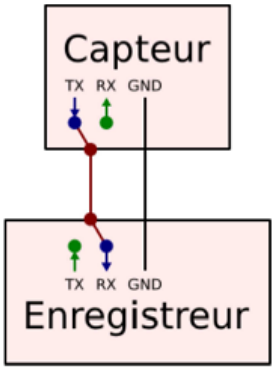
- Unidirectionnelle.
- Les données sont transmises dans un seul sens.
- Ce mode de communication est utilisé quand il n'est pas nécessaire pour l'émetteur d'obtenir une réponse de la part du récepteur. Un circuit électronique comme un capteur qui envoie régulièrement et de manière autonome des données pourra utiliser une liaison simplex.



Si l'on prend l'exemple d'un capteur et d'un système d'enregistrement, la communication **simplex** permettrait au capteur de transmettre ses données de manière autonome, sans être en mesure de recevoir des commandes ou des accusés de réception de la part de l'enregistreur.

3.2. Transmission half-duplex : bi-directionnel alterné

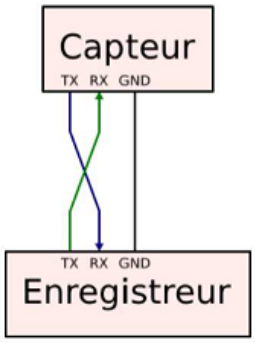
- Bidirectionnelle.
- La transmission est possible dans les 2 sens, mais pas simultanément.
- Il ne peut y avoir sur la ligne qu'un seul équipement en train d'émettre.
- Dans la communication half-duplex, deux systèmes interconnectés sont capables d'émettre et de recevoir chacun leur tour.
- Il faut que les systèmes communicants soient en mesure de déterminer qui a le droit de parler. Dans le cas contraire, on risque d'avoir une collision (quand les deux systèmes tentent de parler simultanément).



Si l'on prend l'exemple d'un capteur et d'un système d'enregistrement, la communication **half-duplex** permettrait par exemple au capteur de se mettre en attente d'une requête de l'enregistreur, puis, à la demande de celui-ci de transférer les données mesurées.

3.3. Transmission full-duplex : bi-directionnel simultané

- Bidirectionnelle.
- Les données sont reçues ou transmises simultanément dans les 2 sens. Deux systèmes interconnectés sont capables d'émettre et de recevoir simultanément.



Si l'on prend l'exemple d'un capteur et d'un système d'enregistrement, la communication **full-duplex** permettrait au capteur de transmettre ses données quand bon lui semble, tout en autorisant le système d'enregistrement à lui envoyer des commandes à tout moment.

Pour des raisons liées au coût et à la robustesse, la plupart des réseaux de communication industriels utilisent :

⇒ Une transmission numérique série asynchrone **half-duplex**.

L'avantage de ce système de communication par rapport au mode full-duplex est qu'il réduit par deux le nombre de canaux de communication nécessaires.

5. TRAME MODBUS :

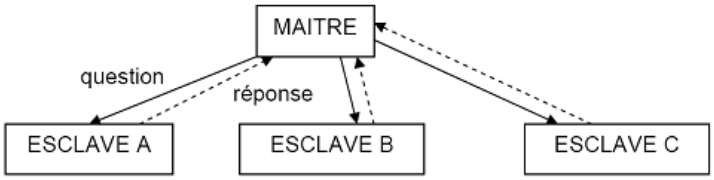
Les trames sont de 2 types :

- Mode RTU (Remote Terminal Unit) : les données sont sur 8 bits.
- Mode ASCII : les données sont sur 7 bits (les trames sont donc visibles en hexadécimal et il faut deux caractères pour représenter un octet).

Ce dernier mode est quasiment tombé en désuétude.

5.1. Principe général :

Le protocole Modbus consiste en la définition de trames d'échange.



Le maître envoie une **demande** et attend une **réponse**.

Le maître peut aussi diffuser un message à tous les esclaves présents sur le réseau. Ceux-ci exécutent l'ordre du message sans émettre une réponse.

5.2. Trame MODBUS RTU (Remote Terminal Unit ⇔ Unité Terminale Distante) :

Le mode de transmission utilisé est le mode RTU. La trame ne contient ni octet d'en-tête de message, ni octet de fin de message. Elle est définie de la manière suivante :

START	Adresse	Fonction	Données	CRC	END
Silence	1 octet	1 octet	n octets	2 octets	Silence

N° esclave	Code fonction	1er paramètre		Autres paramètres	CRC16	
1 octet	1 octet	PF : 1 octet	Pf : 1 octet	N octets	PF : 1 octet	Pf : 1 octet

N° esclave : de 1 à 247.

N° fonction :

- 01 : Lecture de n bits de sortie consécutifs,
- 02 : Lecture de n bits de sortie consécutifs,
- 03 : Lecture de n mots de sortie consécutifs,
- 04 : Lecture de n mots d'entrées consécutifs,
- 05 : Ecriture d'un bit interne ou de sortie,
- 06 : Ecriture d'un mot interne ou registre…

Il existe 19 fonctions possibles sur Modbus. Ces fonctions sont codées sur 1 octet en hexadécimal, tous les équipements ne supportent pas toutes les fonctions.

1re paramètre : Adresse du bit ou du mot adressé.

2ème paramètre : Quantité de mots adressés ou valeur du bit ou du mot écrit selon la fonction utilisée.

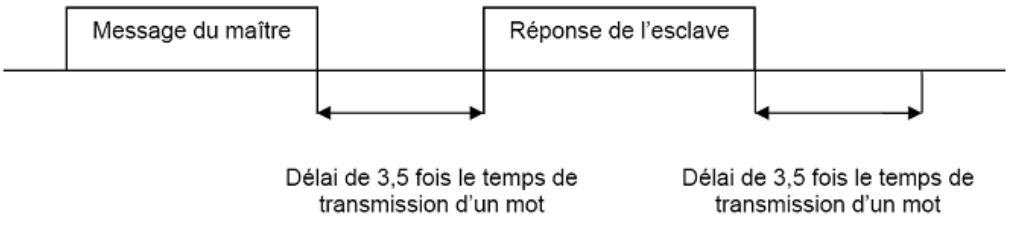
Autres paramètres : Données écrites dans plusieurs mots consécutifs.

CRC16 : Contrôle par redondance cyclique pour détecter les erreurs de transmission.

La détection de fin de trame est réalisée sur un silence supérieur ou égal à 3 caractères.

5.3. Transmission d'un message :

Avant et après chaque message, il doit y avoir un silence équivalent à 3,5 fois le temps de transmission d'un mot.



L'ensemble du message doit être transmis de manière continue. Si un silence de plus de 1,5 fois le temps de transmission d'un mot intervient en cours de transmission, le destinataire du message considérera que la prochaine information qu'il recevra sera l'adresse du début d'un nouveau message.

5.4. Trame MODBUS ASCII :

Chaque octet composant une trame est codé avec 2 caractères ASCII (2 fois 8 bits).

START	Adresse	Fonction	Données	LRC	END
1 caractère	2 caractères	2 caractères	n caractères	2 caractères	2 caractères

Le mode ASCII permet d'avoir des intervalles de plus d'une seconde entre les différents caractères sans que cela ne génère d'erreurs, alors que le mode RTU permet un débit plus élevé pour une même vitesse de transmission.

PARAMETRAGE DE LA COMMUNICATION MODBUS :

Il faut ajuster les paramètres de communication entre le maître et l'esclave :

- Vitesse de communication : **9600** ou **19200** bits/seconde,
- Données (trame) : **8** bits,
- Parité : **Paire** (even), **impaire** (odd) ou **sans parité**,
- Arrêt : **1** ou **2** bits de stop.

Exemple : **8E1** (8 bits de données, parité paire, 1 bit de stop)

Avant l'émission du message, le signal est au niveau logique « 1 » tant qu'aucune transmission n'est en cours.