Introduction à Modbus

Protocole Modbus

logo Modbus
Image not found or type unknown

Introduction

Modbus est un protocole de communication **non propriétaire** créé par Modicon en 1979. Les spécifications du protocole sont données librement sur le site de la Modbus Organization. Ce consortium a été créé par Schneider suite au rachat de Modicon en 1997 pour promouvoir Mobdus auprès des fabricants et utilisateurs.

Modbus est très populaire dans les environnement industriels car c'est un protocole simple, facile à intégrer, efficace, fiable, **ouvert** et royalty-free ! Vous pouvez très facilement intégrer Modbus dans vos projets à base d'ESP32, Raspberry, STM32 ...

Le protocole Modbus était à l'origine un protocole sur bus série (Modbus RTU). Il a évolué pour s'intégrer aux technologies TCP/IP quand Ethernet est monté en puissance. On le retrouve dans les domaines de:

- gestion technique des bâtiments
- systèmes de management de l'énergie
- processus complexes d'automatisation industrielle

C'est une couche applicative (niveau 7 OSI) qui se base sur les liaison séries ou sur les trames Ethernet et les couches TCP/IP.

Stack de communication Modbus :

Modbus Stack
Image not found or type unknown

On distingue les différents modes de communication :

- **Modbus TCP** : communication TCP/IP basée sur le modèle client/serveur
- **Modbus RTU** : transmission **série** asynchrone via **RS-485**, RS-232 ou RS-422.
- Modbus ASCII : similaire au protocole RTU, format sur 7 bit (utilisation très rare)

Nous débuterons l’analyse du protocole suivant la chronologie avec l’étude du **Mobdus RTU** (**R**emote **T**erminal **U**nit) sur liaison série.

Modbus RTU

Principe du protocole Master / Slave utilisé en Modbus Serial

La terminologie Master / Slave est remise en cause ces dernières années dans la communauté des développeurs et l’on évite de l’utiliser sur de nouveaux projets. Comme ces termes sont utilisés dans les spécifications officielles “Modbus Serial”, je continuerai des les employer sur cet exemple par cohérence avec les documentations.

Principe de fonctionnement :

- Seul un Master (au même moment) est connecté au bus, et un ou plusieur (247 maxi) Slaves sont également connecté sur le bus.
- Une communication Modbus est toujours initié par le Master. Les Slaves ne vont jamais transmettre de données sans requête du Master.
- Les Slaves ne peuvent pas communiquer entre eux.

Le Master peut initier une transaction avec le Slave suivant deux modes :

- **mode unicast** le Master s’adresse à un Slave individuel. Après réception de la requête et traitement de celle-ci, le Slave renvoie la réponse au Master. Dans ce mode, une transaction Modbus consiste en deux messages: la requête du Master (request) et la réponse du Slave (reply). Chaque Slave doit posseder une adresse unique (de 1 à 247) de manière à ce qu’il puisse être interrogé indépendamment des autres Slaves.

Modbus RTU unicast
Image not found or type unknown

Le fait d’interroger les Slaves les uns à la suite des autres consiste à effectuer du “Polling”.

- **mode broadcast** le Master envoie **un message à l’ensemble des Slaves**. Les messages de broadcast sont forcément de type écriture. L’adresse 0 est reservée pour identifier un échange de type broadcast.

Description du protocole

Le protocole Modbus définie un Protocol Data Unit (**PDU**) indépendant des couches de communication. Il s’agit de la structure du message de base :

Modbus PDU
Image not found or type unknown

Function Code représente le type d’ordre (lire, écrire) et les datas sont les paramètres de l’ordre (lire 4 registres mémoire depuis l’adresse 0x3214 par exemple).

Empaqueter le protocole Modbus sur un bus série ou Ethernet nécessite des champs additionels au PDU.

Modbus RTU PDU
Image not found or type unknown

Sur une liaison Modbus série, l’Address field contient uniquement l’adresse du Slave.

Le champ CRC contient un code de contrôle d’intégrité de message pour détecter les erreurs de transmission.

Les règles d’adressage Modbus

Les adresses des appareils (devices) Modbus sont codés sur 1 octet (8 bits). Il y a donc 256 adresses possibles.

0	From 1 to 247	From 248 to 255
Broadcast address	Slave individual addresses	Reserved

- L’adresse 0 est réservée comme adresse de broadcast.
- Le Master Modbus n’a pas d’adresse spécifique. Seuls les Slaves doivent posséder une adresse qui doît être unique sur le bus série.
- Le fait que les spécifications Modbus indiquent qu’il est possible d’affecter des adresses comprises entre 1 et 247 ne veut pas forcément dire que tous les fabricants permettent cet interval (certains fabricants limitent les adresses de 1 à 100).

Les types de données

Il y a deux types de données en Modbus, le bit et le Word (16 bits).

	Type d’objet	Accès	Exemple
Discrete Input	bit	Read-Only	Entrée TOR, fin de course, contact auxilliaire de disjoncteurs, …
Coil	bit	Read-Write	Sortie TOR, bit interne, RAZ d’un compteur d’énergie, …
Input Register	Word (16 bits)	Read-Only	Entrée analogique, lecture d’un capteur, …

	Type d’objet	Accès	Exemple
Holding Register	Word (16 bits)	Read-Write	Sortie analogique, variable de programme (ex : temporisation, opérande d’un calcul,...) Valeur de paramétrage d’un équipement (ex: consigne de vitesse d’un variateur de fréquence,...)

- Input et Input Register correspondent à des entrées. Ce sont des variables que l’on peut uniquement accéder en lecture (Read-Only).
- Coil (bobine) et Holding Register correspondent à des sorties que l’on peut forcer (write) mais également lire (read).

Un registre est codé sur 16 bits. Holding Register correspond ainsi à 16 Coil en mode Read-Write tandis que Input Register correspond à 16 entrées que l’on peut seulement acceder en lecture (Read-only).

Rappel :

- 1 Word = 2 bytes = 16 bits
- 1 Register est codé sur un Word soit 16 bits

Les fonctions Modbus

Les “Function Code” correspondent aux types d’ordres, lire ou écrire par exemple, ainsi que le type d’accès (accès au niveau bit ou au niveau registre de 16 bits). Les fonctions sont identifiées par un code sur 8 bits qui peut être représenté en décimal ou en hexa.

Bit access

Code	Hex	Nom de fonction	Commentaire
02	0x02	Read Discrete Inputs	Physical Discrete Inputs
01	0x01	Read Coils	Internal Bits or Physical coils
05	0x05	Write Single Coil	Internal Bits or Physical coils
15	0x0F	Write Multiple Coils	Internal Bits or Physical coils

16-bit access (register)

Code	Hex	Nom de fonction	Commentaire
04	0x04	Read Input Register	Physical Input Registers

Code	Hex	Nom de fonction	Commentaire
03	0x03	Read Holding Registers	Internal Registers or Physical Output Registers
06	0x06	Write Single Register	Internal Registers or Physical Output Registers
16	0x10	Write Multiple Registers	Internal Registers or Physical Output Registers

Les tableaux ci-dessus ne sont pas exhaustif, il y a également des Function Code pour réaliser du diagnostique. Il faut savoir que les fabricants de matériel Modbus n'intègre pas forcément toutes les fonctions possibles. Les fonctions Modbus disponibles sont données dans la documentation technique du constructeur.

Description d'une trame Modbus série

Une communication Modbus série est définie par

- vitesse en bit/s (9600, 19200, 115200, autre)
- 1 bit de start
- 8 bits de données (LSB envoyé en premier)
- 1 bit de parité
- 1 ou 2 bit de stop

Classiquement, en Modbus RTU, c'est la parité paire (**Even**) qui est utilisée. Si l'on choisit de ne pas implémenter le contrôle de parité (**None**) il faut placer 2 bits de stop.

Une trame Modbus RTU est composée *a minima* de 4 octets et au maximun de 256 octets. Chaque octet (byte) qui compose une trame Modbus est codé de la manière suivante :

Modbus RTU frame
Image not found or type unknown

Une trame Modbus RTU

Une trame Modbus RTU est ainsi composée :

- 1 byte pour Slave Address
- 1 byte pour Function Code
- 0 à 252 byte pour Data
- 2 bytes pour le CRC

Modbus RTU Frame
Image not found or type unknown

La taille maximale d'une trame Modbus RTU est de 256 bytes.

Le CRC est calculé avec l'algo CRC-16-MODBUS.

Acquisition d’une trame Modbus de type request



Le décodage de trame Modbus intégré donne au format hexa la trame suivante :

`01 03 00 01 00 02 95 CB`

On en déduit :

- Slave Address : `01`
- Function Code : `03` -> Read Holding Register
- Data : `00 01 00 02`
- CRC : `95 CB`

Pour Data, suivant les caractéristiques de la fonction `03` Read Holding Register, les deux premiers bytes `00 01` corresponde à l’adresse de registre de départ et les deux suivants `00 02` correspondent aux nombre de registres que l’on souhaite lire à partir du registre de départ.

En résumé: la trame Modbus RTU suivante effectue la requête suivante -> Au Slave `01`, donne la valeur des `00 02` premiers registres à partir de l’adresse mémoire `00 01`.

Branchement Modbus RTU en configuration 2 Wire

Le branchement Modbus RTU classique est le “2 Wire” en conformité avec le standard RS-485. Sur un “2W-Bus”, seul un driver à la fois a la possibilité de transmettre un message.

- LT : Line Terminator, c’est les résistance de terminaison (polarisation) du Bus. Elles font classiquement 120Ω120Ω ou 150Ω150Ω
- Les résistances de terminaison sont placées au début du bus et à la fin du bus.
- Balanced Pair : Paire de fils torsadés qui constituent le support de transmission.



On parle de topologie 2 fils (2-Wire), mais on se rend compte sur le schéma, que finalement, 3 fils sont utilisés avec la masse (Common).

Modbus Name	RS-485 Name	Autre Nom	Description
D1	B	D+ ou Data+	Transceiver Terminal 1 (V1>V0 for binary 1 [OFF] state)
D0	A	D- ou Data-	Transceiver Terminal 0 (V0>V1 for binary 0 [ON] state)

Modbus Name	RS-485 Name	Autre Nom	Description
Common	C	0v ou GND	Commun, Masse (0V)

En RS-485, à 9600 bit/s sur une paire torsadée en AWG26, on arrive à une longueur de bus maximale de 1000 m!

Les résistances de polarisation (RPull−UpRPull−Up et RPull−DownRPull−Down) permettent de limiter le bruit sur le bus quand il n’y a pas de communication. Les valeurs de ces résistances sont comprises entre 450Ω450Ω et 650Ω650Ω.

Remarques : Il existe également des configurations de branchement en 4 fils (4-Wire) mais c’est rare.

Connectique Modbus RTU

En Modbus RTU RS-485, trois types de connecteurs connecteurs sont souvent utilisés :

- bornier à visser (ou borne automatique)
- connecteur DB9
- connecteur RJ45

Bornier à visser :

Sur le Wago Controller 100, la connexion se fait par un bornier automatique et utilise les abréviations D+ (D1 ou B) et D- (D0 ou A). L’abréviation GND est utilisée pour le commun (0V) et SH (Shield) pour une connexion au blindage.



Connecteur DB9 :

L’automate PFC200 de chez Wago utilise une connectique DB9 qui permet de réaliser des liaisons RS-485 ou RS232. Pour le Modbus RTU, c’est la RS-485 qui est classiquement utilisée.

PFC200 WAGO	Connecteur DB9
	

La documentation constructeur donne les informations suivantes pour la connectique DB9 du PFC200 en mode RS485.

Contact	Signal RS-485	Description
1	NC	Not assigned

Contact	Signal RS-485	Description
2	NC	Not assigned
3	A (Tx/Rx+)	Transmitt/receive Data+
4	NC	Not assigned
5	FB_GND	Ground
6	FB_5V	Power Supply
7	NC	Not assigned
8	B (Tx/Rx-)	Transmitt/receive Data-
9	NC	Not assigned
Housing	Shield	Shield

On se rend compte que Wago ne respecte pas la norme Modbus dans ce produit ! Ils appellent A -> Data + et B -> Data - qui correspond à la dénomination Profibus de Siemens ! Si votre communication ne fonctionne pas, il suffit parfois d'inverser les fils A-B car le fabricant a mélangé la norme.

Connecteur RJ45

Les fabricants utilisent aussi parfois un connecteur RJ45 pour les liaison RS-485 ! L'erreur est de croire que l'on peut connecter ce type d'appareils sur un switch ou sur le port RJ45 de votre PC. **NE LE FAITES PAS !**

Bien qu'il s'agisse d'un connecteur RJ45, il s'agit d'une liaison série qui est transportée et il faut donc l'associer à une interface série et non au port RJ45 de votre PC ou de votre switch ! Les fabricants adoptent parfois la connectique RJ45 car les câbles sont peu chers avec un branchement qui est facile et rapide.

Wago Current Sensor Modbus RTU

La documentation Wago donne l'association des broches du connecteur RJ45 :

Pin	Function
1	Ub
2	Ub
3	n.c.
4	A (Data+)
5	B (Data-)
6	n.c.

Pin	Function
7	GND
8	GND

Toujours la même erreur chez Wago. Ils appellent A -> Data+ et B -> Data- qui correspond à la dénomination Profibus de Siemens.

Connexion RJ45 et DB9 selon spécifications Modbus

Modbus DB9 RJ45

Image not found or type unknown

Pin on RJ45	Pin on DB9	Level of requirement	Modbus	RS-485	Description
3	3	optional	PMC	-	Port Mode Control
4	5	**required**	D1	B	Transceiver terminal 1, V1 Voltage (V1>V0 for binary 1 [OFF] state)
5	9	**required**	D0	A	Transceiver terminal 0, V0 Voltage (V0>V1 for binary 0 [ON] state)
7	2	recommended	VP	-	Positive 5..24 Vdc Power Supply
8	1	**required**	Common	C	Signal and Power Supply Common

On se rend compte que Wago n’a pas suivi les recommandations de câblage fixées par la Modbus Organization, de nombreux fabricants font de même. Quand il s’agit d’appareillages d’un même constructeur, cela ne pose pas de soucis, par contre, il faut parfois inverser les signaux A et B quand on mélange les appareillages de fabricants différents sur un même bus Modbus RTU. En Modbus TCP, comme c’est sur du câble Ethernet, on n’a pas ce problème.

Exemple : Modbus RTU avec un capteur de Température et

Humidité

Dans cet exemple, je vais connecter un capteur de température et d’humidité PKTH100B-CZ1 qui communique en Modbus RTU avec mon ordinateur portable.

Pour que le PC portable puisse communiquer en RS-485, je lui ajoute un convertisseur FTDI USB-RS485, ainsi qu’un Oscilloscope pour visualiser les trames Modbus-RTU (côté didactique)

Capteur PKTH100B-CZ1	FTDI USB-RS485	Oscilloscope
PKTH100-1 Sensor Image not found or type unknown	FTDI USB-RS485 Image not found or type unknown	SigLent SDS1202X-E Image not found or type unknown

L’analyse de la documentation du câble FTDI USB-RS485 nous donne les informations suivantes :

Les câble USB-RS485

FTDI USB-RS485 colors
Image not found or type unknown

Signal	Couleur de fil
GND	Noir
(A) Data -	Jaune
+5V	Rouge
R de 120Ω120Ω pin 1	Brun
(B) Data +	Orange
R de 120Ω120Ω pin 2	Vert

Le capteur

PKTH100-1 Sensor
Image not found or type unknown

Terminals number	1	2	3	4
Identifying	GND	VCC	B	A
Description	Power-	Power+	RS485-	RS485+

On remarque que sur la documentation du capteur, le signal A est nommé RS485+ tandis que sur la document du convertisseur USB-RS485, le signal A est nommé Data - …

On va rester pragramatique et brancher le fil A (jaune) sur le bornier A (4) du capteur et le fil B (Orange) du convertisseur vers la borne B (3) du capteur. Si jamais cela ne fonctionne pas, il suffira d'inverser ;)

Les masses devant être communes, on branchera le fil GND (noir) du convertisseur à la borne GND (1) du capteur.

Pour alimenter le capteur, j'utilise une alimentation de laboratoire de 24Vdc. Pareil, je brancherai le +24Vdc de l'alimentation à la borne VCC (2) du capteur et le 0V de l'alimentation à la borne GND (1) du capteur.

Pour les résistances de terminaison de 120Ω120Ω, je fais le choix de ne pas les placer dans un premier temps car la longueur de bus est très faible.

La manipulation



Manipulation Modbus Capteur Temperature

La documentation (en chinois) indique les paramètres suivants :

- Vitesse de transmission : 9600 bit/s
- 8 bits de données
- Parity : None
- 1 Stop bit (non respect de la norme)
- Slave Address (factory) : 1

La document indique également que la requête à envoyer est une fonction de type `03` Read Holding Resgister à l'adresse de Slave `1` et que l'on lit à partir du registre mémoire `0` un nombre de 2 registres.

La trame à envoyer avec le CRC est la suivante : `01 03 00 00 00 02 C4 0B`

J'utilise le logiciel QModMaster pour générer facilement la trame et et bien sur, cela ne fonctionne pas :(

> On va essayer d'inverser les fils A et B -> boum, ça fonctionne...bref

Les différentes étapes de la configuration de qModMaster

On le numéro du port Com utilisé par le convertisseur USB-RS485 avec le gestionnaire de périphériques Windows. On remarque que dans mon cas, c'est le COM5 qui lui a été attribué. Cela nous permet de paramétrer la liaison série RTU dans QModMaster avec le bon numéro de Com et

l’on saisie également les paramètres de liaison du capteur de température (9600 bit/s 8bits de données 1 bit de stop et parity None)

Gestionnaire de périphériques	Config Serial dans QModMaster
Image not found or type unknown	Image not found or type unknown

Dans QModMaster, je choisis le Mode RTU, le Slave Address à 1, le Function Code à `0x03` pour Read Holding Register, le Start Address à 0 et Number of Coils (Registers) à 2.

Dans le Bus Monitor, on remarque que la trame de request vaut : `01 03 00 00 00 02 C4 0B` -> ce qui était demandé par la doc, donc on est OK !

La trame de réponse (reply) du capteurs vaut : `01 03 04 01 28 02 22 FA BE`

QModMaster	Bus Monitor
Image not found or type unknown	Image not found or type unknown

Le décodage du résultat est donné directement par QModMaster:

- Le premier registre vaut : `296` en décimal

- Le second registre vaut : `546` en décimal

La documentation du capteur indique que la valeur du premier registre correspond à la température multipliée par 10. On en déduit qu'il fait 29,6°C en cette journée d'août -> c'est OK

L'humidité multipliée par 10 est dans le second registre. On en déduit que l'humidité relative Hr=54.6%Hr=54.6% ce qui est conforme.

Méthode de décodage à partir de la trame de réponse

La trame de réponse (reply) du capteurs vaut : `01 03 04 01 28 02 22 FA BE` . On peut décoder le contenu de la manière suivante :

- `01` : correspond à l'adresse du capteuur qui donne la réponse
- `03` : indique qu'il répond à une réquête de type 03 Read Holding Register
- `04` : c'est la valeur de la fonction 03 + 1 pour dire que tout c'est bien passé !
- `01 28` : c'est la valeur en hexa du contenu du premier registre avec `01` l'octet de poids fort et `28` l'octet de poids faible. Converti en décimal, on obtient `296`
- `02 22` : correspond à la valeur en hexa du second registre. Converti en décimal, on obtient `546` .
- `FA BE` : correspond au CRC de la trame de réponse.

Capture des trames Modbus RTU avec l'oscilloscope

On peut observer la trame de request générée par QModMaster qui vaut `01 03 00 00 00 02 C4 0B`

Modbus Oscilloscope Frame
Image not found or type unknown

Et la trame de réponse du capteur qui vaut `01 03 04 01 28 02 22 FA BE`

Modbus Oscilloscope Frame
Image not found or type unknown

Source https://celka.fr/ocw/plc-control/modbus/intro-modbus/intro/

Philippe Celka Copyright © 2025 CC Attribution-Non Commercial-Share Alike 4.0 International

---

Revision #1
Created 21 May 2025 08:04:44 by admin_idf
Updated 13 April 2026 14:18:47 by admin_idf