Responsable pédagogique	AF	AM	PB	**PM**
Période	Sem1	Sem2	Sem3	**Sem 4**
Volume horaire	Cours		TP	
			8	

TER
TP2 Embarque Modbus over Serial
Pour Module BB

Indicateur temporel (hors rédaction du compte-rendu) :

questions	1h	2-8h
1		
2		

Documents à rendre : Compte-rendu contenant à minima les sources (avec entête standard) et les résultats obtenus.

BUT : On souhaite faire communiquer un client, avec plusieurs modules esclaves BBE/S en **Modbus sur liaison RS485** . Cf Fig1 . synoptique matériel et Annexe3.

Vous coderez les appels de fonctions modbus dans le main.cpp ; de sorte que seul le client est à coder.

Note : Pour ce TP, il est indispensable d'utiliser les fonctions fournies pour la communication jbus, la création par Makefile. Quand vous compilerez, vous obtiendrez le binaire nommé *progjbus*. (vérifiez par ***file progjbus*** le type de processeur sur lequel il est exécutable : normalement ELF 32-bit LSB executable, ARM, sur le raspberry).

Si vous avez besoin de compiler votre main.cpp , faites un make all.

TEST : Vous disposez dans l'archive du programme de test ***jbus2 (à essayer en root donc sudo ./jbus2)***

Remarque importante :

Vous devrez travailler que sur le raspberry en étant en ssh. Tous les fichiers issus du sujet .tar devront etre mis par sftp sur votre repertoire du raspberry.

Le synoptique (fig 1), vous montre que vous serez plusieurs connectés simultanément sur la machine raspberry (adresse ip 192.168.20.203 ou 192.168.20.207) ; login : pi ; mdp : raspberry .

Chacun d'entre vous aura à développer et tester son propre programme client.

Etant donné qu'il n'y a qu'une liaison série ; il n'y aura donc, à un instant T , qu'un seul programme client qui disposera de la liaison série modbus !!

Ne pas retirer le gestionnaire de signal (CTRL-C <=> SIGINT) proposé dans main.cpp. Autrement vous risquez d'avoir à redémarrer le client pour " récupérer " la liaison série.

1. Repérage des adresses de modules et registres internes

Etudier la documentation des modules BBES (Annexe 1) ; afin de connaître les adresses à gérer en écriture / lecture sur chacune des cartes. Ce sont ces adresses que vous mettrez dans vos trames.

Vous ferez attention au réglage (SETUP) de la communication rs485, qui doit être en 8 bits de données, 2 bits de stop, Sans parite.

ATTENTION : un des 3 modules disponibles est un BBS16 (16 sorties relais) et ne dispose pas d'entrées.(par contre il possède 2 fois plus de sorties disponibles qu'avec un BBES).

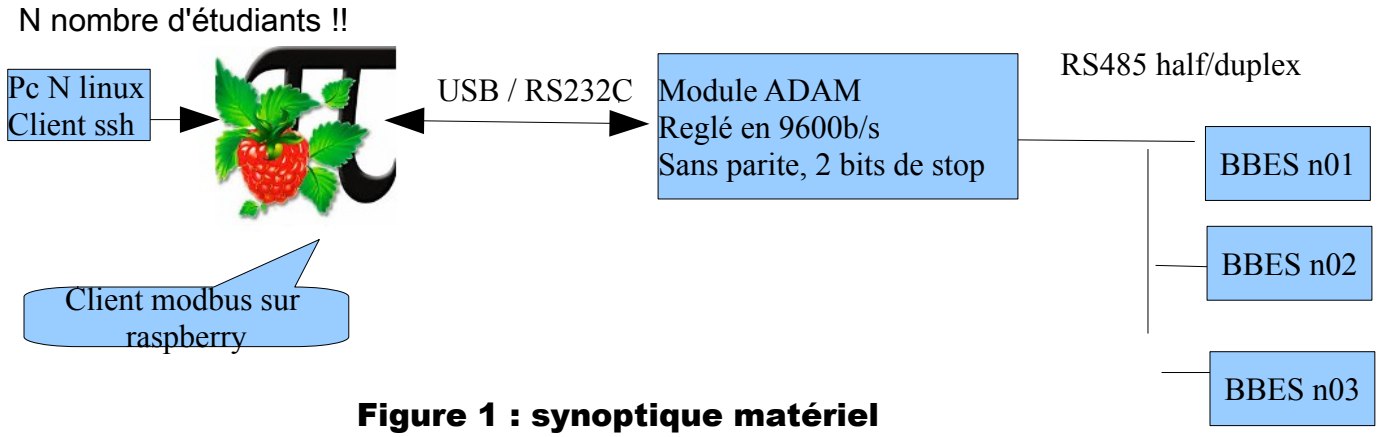


Figure 1 : synoptique matériel

Les adresses des modules sont à repérer selon la documentation (Cf annexe 1)

Vous pouvez vous concentrez sur le client désormais. Le(s) serveur(s) modbus (boitiers BBES) est(sont) actif(s) dès la mise sous tension de la platine maquette.

2. Codage de l'interface cliente

Etudier le paramétrage de l'IHM texte cliente afin de pouvoir envoyer des trames au format MODBUS pour écrire/lire des données sur différents esclaves.

Les trames générées le seront sous une forme standard modbus RTU.

Cf en Annexe 2 et 3 les exemples d'exécution, selon l'usage de la librairie Modbus fournie Annexe 4.

IMPORTANT : Créez-vous un répertoire personnel de test, sur cette machine (i.e cliente modbus), afin ne pas confondre par la suite, votre binaire avec celui d'autres étudiants.

Travail demandé

A partir du main.cpp

Codez l'instanciation d'un objet de classe jbus en omplétant l'appel correct au constructeur de classe jbus. Exemple : j1 = new jbus(…) ; (Cf extrait de documentation doxygen Annexe 4 ou lancer un navigateur web sur le fichier index.html du dossier html de votre répertoire, pour avoir la doc en ligne)

La liaison série est en **9600 bauds, 8 bits de données, 2 bit de stop, Sans parité.**

Créer ensuite un menu IHM permettant, au sein d'une boucle infinie de choisir le type de carte et le type d'opération à faire. Utiliser dans les bons cas, les méthodes de la classe jbus.

Attention, à bien savoir si vous lisez(écrivez) un (ou plusieurs) mot, un bit etc..., sur quel module (N° module CF roue codeuse), et en quelle adresse de registre interne.

A vous de lire par programmation et d'afficher le résultat par fonction modbus appropriée.

Attention : Pensez à utiliser les accesseurs fournis dans la classe pour pouvoir afficher vos trames reçues.

Annexe 1 : Documentation BBES

NOTICE D'UTILISATION DES TERMINAUX ESCLAVE BB16

sous protocole MODBUS® JBUS®

Caractéristiques des entrées sorties T.O.R

Entrées contact sec

Les entrées contact sec sont actives et isolées galvaniquement de la CPU à 3000Vcc. Le BB16 délivre un courant de 6mA par entrée à partir d'une tension commune de 12V.

Sorties relais

Sorties relais libre de tout potentiel et isolées entre elles à 3000Vcc. Pouvoir de coupure de chaque sortie est de 5A sous 250Vac

Port de communication

La jonction de communication est isolée Galvaniquement à 3000Vcc

Characteristics of inputs and outputs

Dynamic inputs

Dynamic inputs are active and galvanically isolated from the CPU with 3000Vcc.
The BB16 deliver a current of 6mA by input from a common tension of 12V.

Outputs

All outputs are isolated with 3000Vcc and free of any potentiel. Max current by output : 5A at 250Vac

Serial Port

The communication junction is galvanically isolated at 3000Vcc

Configuration

Utilisation de la roue codeuse SETUP du BB16

SETUP vous permet de configurer le port de communication de BB16

POSITION du curseur	VITESSE (Bds)	PARITE	POSITION du curseur	VITESSE (Bds)	PARITE
0	**300**	**SANS**	**8**	**4800**	**PAIRE**
1	**1200**	**SANS**	**9**	**4800**	**IMPAIRE**
2	1200	PAIRE	A	9600	SANS
3	1200	IMPAIRE	B	9600	PAIRE
4	2400	SANS	C	9600	IMPAIRE
5	2400	PAIRE	D	19200	SANS
6	2400	IMPAIRE	E	19200	PAIRE
7	4800	PAIRE	F	19200	IMPAIRE

Utilisation des roues codeuses A0 et A1

Accessible sur la face avant, A0 et A1 vous permettent de sélecter le n° abonné allant de 0x01 à 0xFF (1 à 255). A0 correspond au 4 bits de poids faible et A1 au 4 bits de poids fort. Prise en compte du set up après reset (marche/arrêt du BB16)

Configuration

Use of the rotary switch SETUP

SETUP let you configuring the serial port of BB16

Use of the rotary switches A0 and A1

Accessible on the front face, A0 and A1 allow you to select the n° slave of 0x01 with 0xFF (1 to 255). A0 corresponds to the LSB and A1 with the MSB. Taking into account of the set up after reset (Stop and go of the BB16)

Utilisation

Les adresses du BB16 accessibles par l'utilisateur sont les suivantes :

Adresse 0	Etat des entrées sorties. Bit de poids faible correspond à l'I/O 0. Les entrées sont placées côté poids faible et les sorties poids fort.	State of input output. Weak weight bit corresponds to I/O 0. Input are placed by weak weight and output by strong weight
Adresse 1	Etat des entrées sorties après filtrage numérique.	State of input output after numerical filtering
Adresse 2 à 9	Compteurs 16 bits associés aux entrées sorties. Compteur à l'adresse 1 associé à I/O n° 0	Reader 16 bits associated to input output. Reader of adress 1 associated to I/O n°0
Adresse 10	L'octet de poids fort permet de configurer le filtre numérique des entrées TOR.	8 bits strong weight enable to configure the numerical filter of TOR input
	Fréquence de comptage = 1/((256-Filtre)x 10^{-3})	Frequency of reader = = 1/((256-Filter)x 10^{-3})
	L'octet de poids faible permet de configurer le front de comptage pour chaque I/O	8 bits of weak weight enable to configure the front of reader for every I/O
	Bit à 1 correspond à front montant	Bit to 1 correspond to rising front
	Bit à 0 correspond à front descendant	Bit to 0 correspond to downward front
Adresse 11	Mémorisation du dernier événement après une lecture de n Mots. Le passage à 1 de l'un des bits correspond à un changement d'état de l'un des I/O après une lecture de n mots	Memorizing of last event after reading n words. The passage to 1 of one of the bit correspond to a change of state of one of the I/O after reading n words
Adresse 12	Temps de repli des sorties relais sur 16 bits	Time of fold of output on 16 bits

Use

Adresses of BB16 :

	(0x0000< Repli<0xFFFF)	(0x0000< Repli<0xFFFF)
	Repli = 0x0000 temps de repli 50 mS	Fold = 0x0000 time of fold 50 mS
	Repli = 0xFFFF temps de repli de 3276S	Fold = 0xFFFF time of fold of 3276S
	Temps de repli (S)= Repli x 0.05	Time of fold (S) = Fold x 0.05
Adresse 13	Activation de la position de replis pour chaque sorties TOR. Un bit à 1 active la position de repli sur l'IO correspondant. Bit de poids faible correspond à l'I/O 0.	Activation of fallback position for every TOR output. A bit to 1 activate fallback position on the I/O corresponding. Weak weight bit correspond to I/O 0
Adresse 14	Configuration de l'état de la position de repli. Un bit à 1 active l'IO correspondant (relais femé) et à 0 inactive l'IO correspondant (relais ouvert). Bit de poids faible correspond à l'I/O 0.	Configuration of the state of fallback position. A bit to 1 activate the I/O corresponding (relay closed) and to 0 inactive the I/O corresponding (relay open). Weak weight bit correspond to IO 0
Adresse 15	L'octet de poids forts permet de fixer le nombre d'impulsion sur les sorties. 0 pas d'impulsion, 0xFF impulsion à l'infinie.	8 bits of strong weight enable to fixe the number of pulse on output. 0 any impulse, 0xFF impulse to the infinite
	L'octet de poids faible permet de fixer la durées de l'impulsion par pas de 50mS. 1 durées de l'impulsion de 50mS, 0xFF durées de l'impulsion de 12,5S.	8 bits of weak weight enable to fix the durations of the impulse by step of 50mS. 1 duration of impulse of 50mS, 0xFF duration of impulse of 12.5S.
Adresse 16	Activation des impulsions sur les sorties. Bit à 1 lance le cycle d'impulsion.	Activation of the impulses on the output. Bit to 1 give the cycle of impulse
Adresse 17	Etat de la première impulsion. Bit de poids faible correspond à l'I/O 0.	State of the first impulse. Weak weight bit correspond to I/O 0
Adresse 18	L'octet de poids faible permet de fixer le temps de retournement entre question et réponse (pas de 50mS).	Time of reversal between question and answer.

Les fonctions implémentées dans le BB16 sont les suivantes :

The functions implemented in the BB16 are as follows:

Adresse mot	Lecture n bits (1 et 2)	Lecture n Mots (3 et 4)	Ecriture 1 bit (fct 5)	Ecriture 1 mot (fct 6)	Lecutre rapide (fct 7)	Diagno (fct 8)	Compt (fct 11)	Ecriture n bits (fct 15)	Ecirture n mots (fct 16)
0	√	√	√	√	√			√	√
1 à 18	√	√	√	√				√	√
Comp						√	√		

Adaptation de ligne

Résistances de terminaison

Pour une transmission longue distance (>100m) une adaptation de ligne est impérative. Pour cela, le câblage à chaque extrémité de ligne d'une résistance de 120 Ω est nécessaire.

Résistance de polarisation

Cette polarisation est constituée par 2 résistances (de 1KΩ à 2KΩ) placées pour la résistance de PULL UP entre la ligne +RT et le +5V, et pour la résistance de PULL DOWN

Adaptation of line

Terminal resistor

If the distance of transmission is more than 100 meters long, it is necessary to set a terminal resistance of 120 Ω .

Polarisation resistor

Consist on 2 resistors (from 1KΩ to 2KΩ). The PULL UP resistance is between +RT and +5V and the PULL DOWN resistance between –RT and Gnd. Polarisation resistor must be used at the begining

entre la ligne -RT et le 0V. Cette polarisation est à placer soit en début de ligne, côté maître soit en fin de ligne côté esclave. **Attention : ne pas mettre de résistance de polarisation ou de terminaison sur les dérivations.**

Communication

Jonction RS422/485

En standard, le BB16 est équipé de jonction de dialogue RS422 (4fils) et RS485 (2 fils)
Jonction RS422/RS485
Le fonctionnement bipaire ou monopaire (2/4 fils) est déterminé par le câblage du bornier de jonctions

Câblage Bipaire

Ligne Tx représentée par la borne +/–RT sera à connecter sur la ligne +/- RX du maître
Ligne RX représentée par la borne +R et –R sera à connecter sur la ligne +/- Tx

Câblage Monopaire

1 seule paire est utilisée pour les jonctions RX et TX. +/- RT du maître sera connecté sur le +/- RT de l'esclave

Caractéristiques mécaniques

Robuste boîtier aluminium, anodisé, incolore
Dim : 105x55x95mm (HxlxP) - Rail din
Poids total: alimentation secteur 320g
alimentation continu 270g
Conditions de fonctionnement
Temp. fonctionnement de –10°C à +60°C
Temp. stockage de -20°C à +70°C
Humidité de 0 à 90° sans condensation

Caractéristiques générales de la transmission

RS485 EIA Ve+ > Ve- de 0,2 mV →état logique 1
RS485: nb d'abonnés maximal sans répéteur = 32.
Impédance de ligne de 100 Ω à 150 Ω.
Isolement liaison série / terre de 2000 Véff
Liaison: 800 m à 38400Bds et 2 Km à 19200Bds

with master or at the end with slave of twisted pair cable

Warning: don't used terminal and polarization resistance on wiring derivation.

Communication

RS422/485 junction

In general, the BB16 is equipped of RS422 dialog junction and RS485
The half and full duplex operation is possible without any commutation determined by wiring of junction terminal

In Fullduplex

TX line represented by +/–RT will be connected on +/-RX line of master
RX line represented by +R and –R will be connected on +/- TX line of the master

In Halfduplex

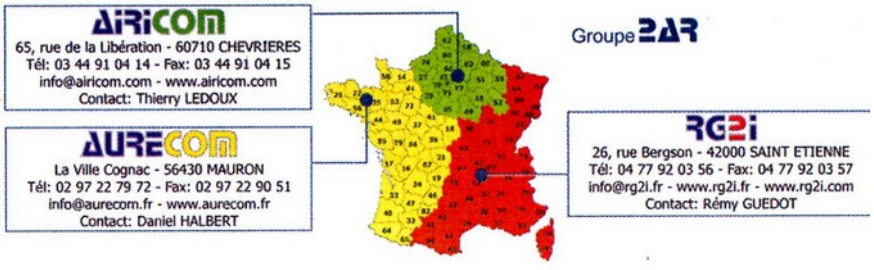
Only 1 pair is used for RX and TX junctions. +/-RT of master will be connected on +/-RT of slave

Mechanical characteristics

Aluminium, colourless, anodized casing
Symetrical, asymetrical pods
Total weight : 320g mains supply
270g DC supply
Operating conditions
Operating temp : -10 to +60°C
Stockage temp : -20 to +70 °C
Humidity : 0 to 90° without condensation

Transmission characteristics

RS485 EIA Ve+ > Ve - de 0,2 mV → hight logic level
RS485 : nb max of receivers without repetor = 32
Impedance of connexion from 100 Ω to 150 Ω.
Serial line and ground isolated at 2000 VRMS
Connexion : 800m at 38400Bds and 2Km at 19200Bds



Annexe 1 bis : visuel des connexions

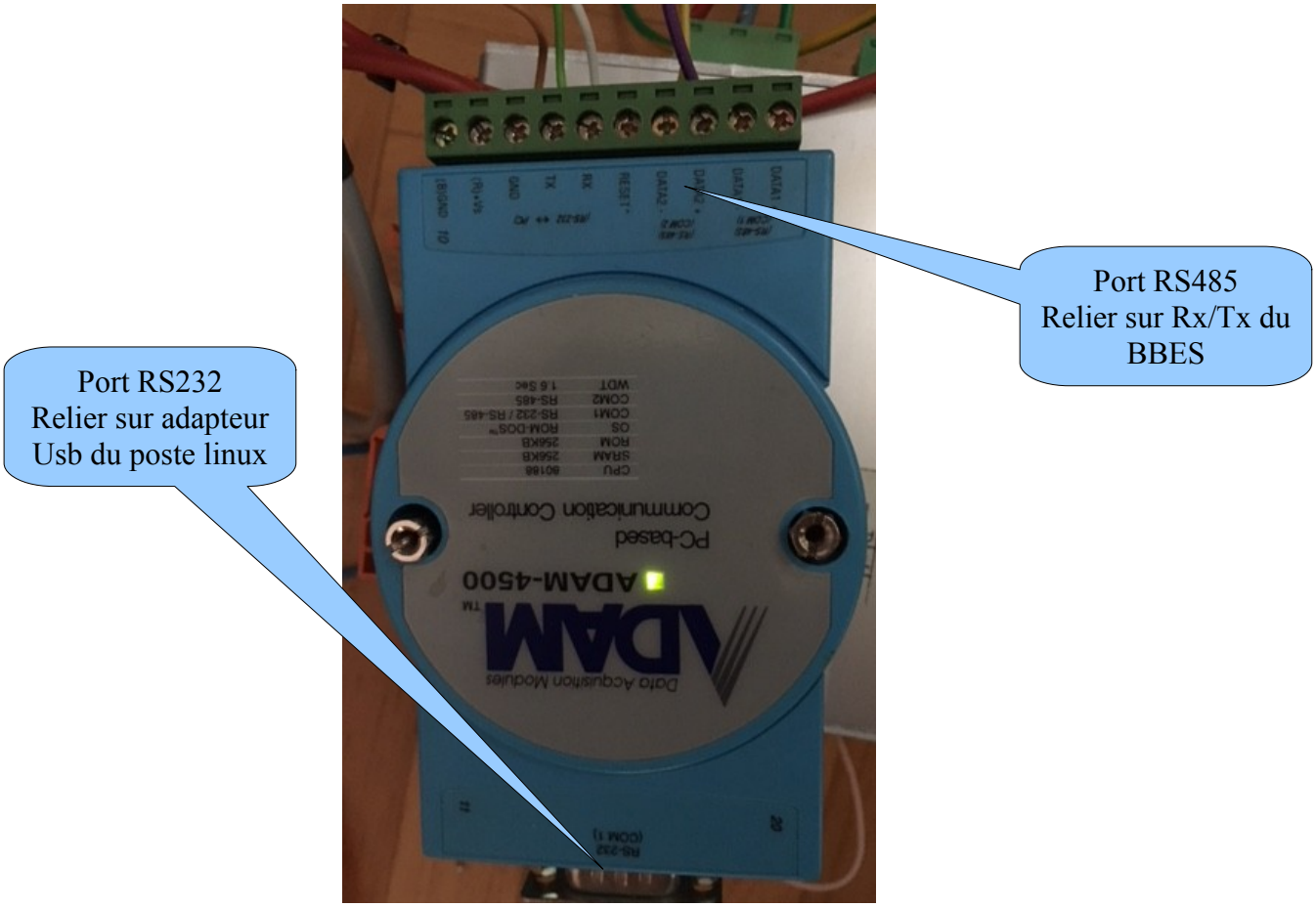


Figure 2: module ADAM4500

Annexe 2 : Exemples d'exécution

```
pm@patrickMAC:~/TS2/TER/correction/v1$ sudo ./jbus2
[sudo] Mot de passe de pm :
 port/dev/ttyUSB0
Ouverture port USB1
8 bits
parite noOK
Faites un choix opération 1 : Ecriture  2 : Lecture
1
Faites un choix de carte : 0: BBES ; 1 DAC ; 2 : CNT ; 3 : VDOUT ; 4 : APAL ; 5
: VDAD ; 6 : VDIN
0
adresse Module BBES ?
2
adresse registre?
0
valeur a ecrire
256
valeur trame reponse ecriture 2 6 0 0 1 0 88 69
Faites un choix opération 1 : Ecriture  2 : Lecture
```

Figure 3 Exemple coté client MODBUS : écriture d'un mot de valeur 256 (sortie S0) sur le module BBES n° 2

Octet Poids faible Entrées= E7E6...E0
Octet Poids fort Sorties= S7S6...S0
Où E0 = 1 et S0 = 1

```
Faites un choix de carte : 0: BBES ; 1 DAC ; 2 : CNT ; 3 : VDOUT ; 4 : APAL ; 5
: VDAD ; 6 : VDIN
0
adresse Module BBES ?
2
adresse registre?
0
valeur trame reponse lecture  2 1 2 1 1 3d ac
```

2 octets lus

Figure 4 Exemple coté client : lecture de 16 bits (fonction 1) sur le module BBES n° 2

Annexe 3 : visuel du module BBES

Figure 5 : visuel de la BBES d'adresse 3, avec l'entrée E0 = 1 et la sortie S0 = 1

Annexe 4 : extrait de la documentation Web de l'interface fournie

Accessibilité par navigateur sur le fichier index.html du répertoire local d'installation ./html/index.htm

Fonctions membres publiques

	jbus (char *line, int b_rate, int nb_bit_data, int n_bit_stop, int parity_yes_no, int odd_even_ms) constructeur permettant de creer des objets avec les reglages serie adaptés Plus de détails...
int	**API_Module_Read_Nbits** (unsigned char slave_adress, short adresse_variable_tor, short N) methode de lecture de N bits interne sur module esclave Plus de détails...
int	**API_Module_Read_Nwords** (unsigned char slave_adress, short adresse_mot, short N) methode de lecture de N mots internes sur module esclave Plus de détails...
int	**API_Module_Write_1word** (unsigned char slave_adress, short adresse_variable_tor, short valeur) methode d'ecriture de 1 mot sur un module de l'API Plus de détails...
void	**read_value_C** (unsigned char *res, int n) methode pour lire trame reponse en ecriture Plus de détails...
void	**read_value_R** (unsigned char *res, int n) methode pour lire trame reponse en lecture Plus de détails...

Documentation des constructeurs et destructeur

```
jbus::jbus ( std::string  line,
             int          b_rate,
             int          nb_bit_data,
             int          n_bit_stop,
             int          parity_yes_no,
             int          odd_even_ms
           )
```

constructeur permettant de creer des objets avec les reglages serie adaptés

Paramètres

[in] **stinrg** line : nom port serie "/dev/ttyS0" <=> COM1 , "/dev/ttyUSB0" <=> USB1

[in] **int** b_rate : baud rate 9600,19200 etc

[in] **int** nb_bit_data : nombre de bit de donnees 5,6,7,ou 8

[in] **int** n_bit_stop : nb bit stop 1, 1,5 , ou 2

[in] **int** parity_yes_no : parite 'O',1 <=> oui , 'N', 0 <=> non

[in] **int** odd_even_ms : type parite : 'o','O','0', 0 <=> IMPAIRE; 'e','E','1',1 <=> PAIRE

```
)
```

methode de lecture de N bits interne sur module esclave

Paramètres

[in] **unsigned** char slave_adress : Numero du module

[in] **short** adresse_variable_tor : adresse variable ds API

[in] **short** N : Nombre bits à lire

Renvoie

code d'erreur eventuel -1 NOK O OK

```
int jbus::API_Module_Read_Nwords ( unsigned char  slave_adress,
                                   short          adresse_mot,
                                   short          N
                                 )
```

methode de lecture de N mots internes sur module esclave

Paramètres

[in] **unsigned** char slave_adress : Numero du module

[in] **short** adresse_mot : adresse variable ds API

[in] **short** N : Nombre mots à lire

Renvoie

code d'erreur eventuel -1 NOK 0 OK

int jbus::API_Module_Read_Nwords (unsigned char slave_adress,
short adresse_mot,
short N
)

methode de lecture de N mots internes sur module esclave

Paramètres

[in] **unsigned** char slave_adress : Numero du module
[in] **short** adresse_mot : adresse variable ds API
[in] **short** N : Nombre mots à lire

Renvoie

code d'erreur eventuel -1 NOK 0 OK

int jbus::API_Module_Write_1word (unsigned char slave_adress,
short adresse_variable_tor,
short valeur
)

methode d'ecriture de 1 mot sur un module de l'API

Paramètres

[in] **unsigned** char slave_adress : Numero du module
[in] **short** adresse_variable_tor : adresse variable ds API
[in] **short** valeur : valeur bits a ecrire

Renvoie

sortie : code d'erreur eventuel -1 NOK 0 OK