

Sommaire

1.Préambule.....	1
2.Présentation.....	2
3.Technologie des communications asynchrones.....	3
3.1.Liaison série « logique ».....	3
3.2.Communications normalisées.....	4
3.3.Liaison RS232C.....	5
3.4.Liaisons RS422, RS485.....	7
4.Le code ASCII.....	8
4.1.Présentation.....	8
4.2.Table de caractères.....	8
4.3.Caractères spécifiques de contrôle.....	9
5.Application : gestion d'une imprimante au fil de l'eau.....	10
5.1.Objectif.....	10
5.2.Le matériel nécessaire.....	10
5.3.Raccordements.....	10
5.4.Configuration.....	10
5.5.Programmation.....	11
6.Application : gestion d'un API en esclave Modbus.....	12
6.1.Objectifs.....	12
6.2.ModBus.....	12
6.3.Le matériel nécessaire.....	14
6.4.Configuration.....	16
6.5.Manipulations.....	18
6.6.Synthèse du dialogue.....	21
7.Des liens.....	21

1.Préambule

Ce document-ressource est destiné à fournir toutes les informations nécessaires à l'enseignement des communications industrielles asynchrones en Section de Technicien Supérieur en Électrotechnique : théorie des liaisons de communication, standards industriels, mise en œuvre de matériels dédiés.

Il présente aussi un cadre de travaux pratiques qui peuvent être mis en place afin d'illustrer l'enseignement : communication avec des automates industriels récents, réalisation de cordons de mesure, utilisation d'une application logicielle pédagogique développée spécifiquement, relevés à l'oscilloscope.

2.Présentation

Les « liaisons séries » sont des moyens de transport d'informations (communication) entre divers systèmes numériques. On les oppose aux liaisons parallèles par le fait que les différents bits d'une donnée ne sont pas envoyés en même temps mais les uns après les autres, ce qui limite le nombre de fils de transmission. Elles sont appelées asynchrones car aucune horloge n'est transportée avec le signal de données. Les liaisons séries asynchrones sont rencontrées sous différentes normes dans tous les domaines du traitement de l'information :

Domaine	Dispositif	Type de communication
Bureautique	connexion de souris, modem, ...	RS232 sur COM1 à COM4
	connexion de souris et clavier à l'unité centrale	PS/2
Mesure	connexion oscilloscope numérique à traceur ou imprimante	
Automatismes	connexion API à console ou PC de programmation	RS232 ou RS485
	réseau Modbus, Fip-IO, FipWay	RS485

3. Technologie des communications asynchrones

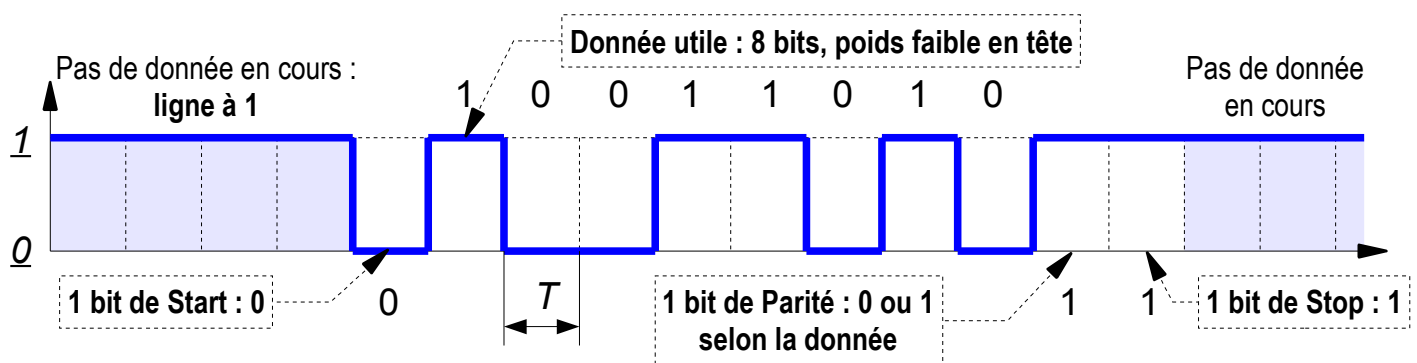
3.1. Liaison série « logique »

3.1.1. Protocole

Le protocole d'échange asynchrone est défini par l'envoi, pour chaque caractère émis, de :

- un bit de Start,
- les 5 à 8 bits de données, poids faible en tête,
- éventuellement, un bit de vérification de Parité qui permet de détecter des erreurs de transmission des 8 bits de donnée sur la ligne,
- 1, 1½, ou 2 bits de Stop après.

Lorsque aucun caractère ne circule sur la ligne, celle-ci reste à l'état logique haut (« 1 »).



Note : Ce chronogramme représente l'état logique AVANT la mise en forme par l'adaptation de ligne, c'est à dire indépendamment du standard RS232, 422 ou 485.

3.1.2. Vitesse de transmission

La vitesse de transmission représente la quantité d'informations qui peuvent être transportées pendant un certain temps. Elle est exprimée en bits par seconde (bps).

L'unité de BAUDS, parfois rencontrée, est une caractéristique du signal logique modulé (donc converti en analogique), et représente le nombre de variations de fréquence (ou de phase) par seconde. Si chaque niveau logique (bit) est associé à une fréquence, les unités Bauds et bps sont équivalentes.

Les vitesses de transmission peuvent être entre autres :

Vitesse en bits par seconde (bps)	Application
75	Émission Clavier Minitel → Serveur Télétex
110, 300, 600	
1200	Réception Serveur Télétex → Écran Minitel
2400	
4800	API
9600	API, Modem-Fax
14400	API, Modem-Fax
19200	API
56000, 115200, 128000, 256000	

3.2. Communications normalisées

Les différents types d'interface sont couramment désignés par le numéro de l'avis ou de la norme qui les définissent :

"RS..." correspond aux normes américaines définies par l'EIA (Electronics Industries Association).

"V..." ou "X..." correspond aux avis internationaux définis par le CCITT (Comité Consultatif International pour le Téléphone et les Télécommunications).

Boucle de courant particulièrement utilisée dans l'industrie, ne correspond pas à une norme.

EIA CCITT	RS 232 V24 / V28	RS 423	RS 422 V11 / X27	RS 485 V11 / X27	Boucle de courant
Type d'interface	Unipolaire	Unipolaire	Différentiel	Différentiel	0-20 mA
Sensibilité					
Distance (m)	15	1200	1200	1200	1000 à 2000
Débit max. (bps)	19200	100 K	10 Mbps sur 12 m 100 Kbps sur 12 m	10 Mbps sur 12 m 100 Kbps sur 12 m	19200
Multipoint	non	oui	oui	oui	oui
Nombre d'émetteurs	1	1	1	32	
Nombre récepteurs	1	10	10	32	
Niveau de sortie non chargé (V)	± 25	± 6	± 6	± 6	
Niveau de sortie chargé (V)	± 5 à ± 15	± 3,6	± 2	± 1,5	
Impédance d'entrée	3 à 7 kΩ	≥ 4kΩ	≥ 4 kΩ	≥ 12 kΩ	
Charge émetteur	3 à 7 kΩ	≥ 450 Ω	100 Ω	54 Ω	

3.2.1. Glossaire

Unipolaire	chaque signal électrique est référencé par rapport à une masse unique. Pour 7 signaux échangés, on aurait besoin de $7 + 1 = 8$ conducteurs.
Différentiel	chaque signal électrique est transporté entre 2 conducteurs, chacun a donc sa référence. Pour 4 signaux échangés, on aurait besoin de $2 \times 4 = 8$ conducteurs.
DTE	« Data Terminal Equipment » : un équipement terminal de données est typiquement un ordinateur qui peut envoyer des données (depuis une application ou un clavier) et recevoir des données (vers une application ou l'écran).
DCE	« Data Communication Equipment » : un équipement de communication de données ne génère aucune donnée mais convertit leur niveau électrique, typiquement c'est un Modem.
Mark	Niveau logique haut = « 1 »
Space	Niveau logique bas = « 0 »

3.3. Liaison RS232C

L'évolution temporelle des signaux RS232 est conforme aux signaux de liaison asynchrone décrits précédemment. La spécificité de RS232 tient dans l'adaptation en tension des signaux afin d'être transmis sur une distance supérieure (15m).

3.3.1. Désignation des signaux

3.3.1.1 Signaux de données

Sur une liaison bidirectionnelle minimale sans contrôle de flux, il faudra 3 conducteurs :

Tx	Transmitted Data	Conducteur d'émission des données
Rx	Received Data	Conducteur de réception des données
Gnd	Ground	Conducteur de masse du signal

3.3.1.2 Contrôle de flux

Les équipements connectés pour un échange de données (communication) peuvent ne pas fonctionner à la même vitesse. Si le récepteur est plus rapide que l'émetteur, aucun problème n'apparaît.

Si l'émetteur travaille plus vite que le récepteur, des données peuvent être perdues. Il faut donc mettre en place un contrôle de flux par des signaux appropriés.

Contrôle de flux matériel

Le contrôle de flux est assuré par la présence et la connexion de conducteurs supplémentaires entre le DTE et le DCE.

RTS	Request To Send	Ce signal est abaissé (« 0 ») pour préparer le DCE à accepter les données transmises. La préparation consiste à activer les circuits de réception, ou activer le canal dans les applications demi-duplex. Lorsque le DCE est prêt, il acquitte en abaissant CTS.
CTS	Clear To Send	Le signal est abaissé par le DCE pour informer le DTE que la transmission peut débuter.

Contrôle de flux logiciel

Le récepteur stoppe le flux de données en envoyant sur la ligne de données un caractère dédié nommé XOFF, et le relance en envoyant le caractère XON. D'où le nom du protocole XON/XOFF. Le caractère XON est DC1, XOFF est DC3.

3.3.1.3 Signaux de contrôle et d'état de modem

DSR	Data Set Ready	Si connecté à un modem : <ul style="list-style-type: none"> ✓ Le modem est connecté sur une ligne téléphonique saine ✓ Le modem est en mode Data, et non pas en mode voix ou numérotation ✓ Le modem est en train de générer une tonalité de réponse Si connecté à un autre dispositif : <ul style="list-style-type: none"> ✓ Le DCE est actif. Si non utilisé, doit être forcé à « 0 ».
DCD	Data Carrier Detect	Détection de porteuse de données sur la ligne
DTR	Data Terminal Ready	
RI	Ring Indicator	Signale une sonnerie d'appel téléphonique sur le modem

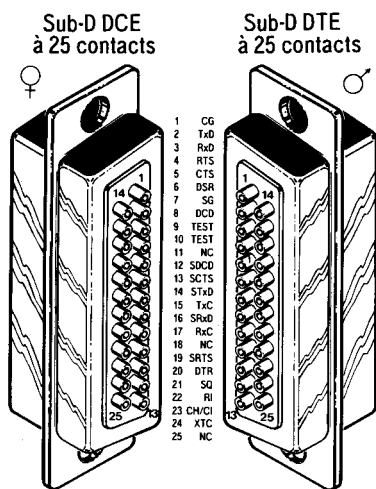
3.3.2. Niveaux des signaux

Niveau logique	Polarité	Intervalle de niveau électrique	Typique
'1'	Basse	entre -3V et -15 V	-12V
'0'	Haute	entre +3V et +15 V	+ 12V

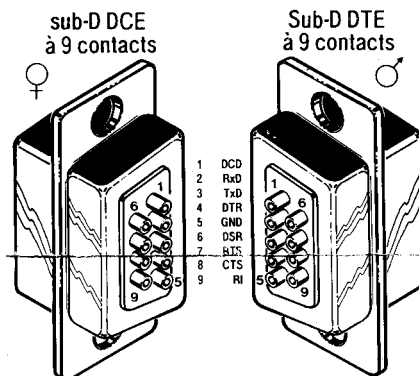
On dit donc que l'on travaille en logique négative.

3.3.3. Connectique

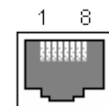
Canon Sub-D 25 broches



Canon Sub-D 9 broches



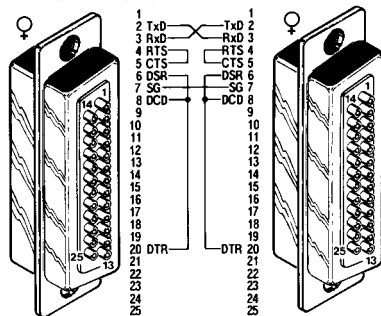
RJ45 (RS 232 D)



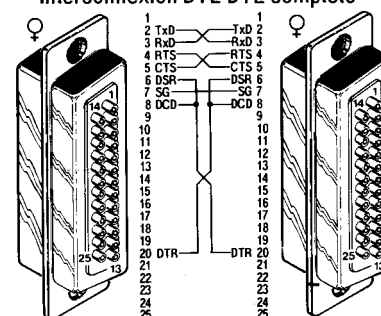
- 1 DSR/RI
- 2 CD
- 3 DTR
- 4 Gnd
- 5 RxD
- 6 TxD
- 7 CTS
- 8 RTS

3.3.4. Raccordements

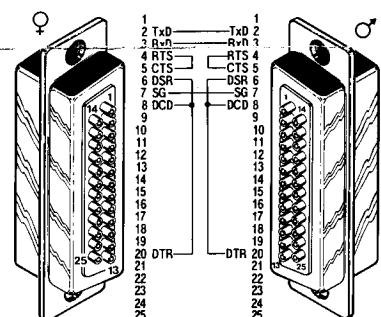
Interconnexion trifilaire DTE-DTE



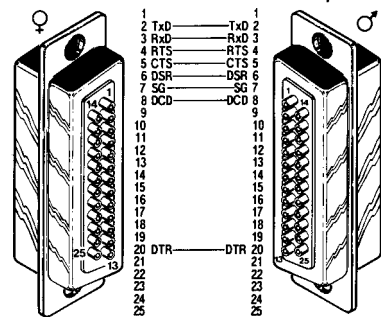
Interconnexion DTE-DTE complète



Interconnexion trifilaire DTE-DCE



Interconnexion DTE-DCE complète



3.4. Liaisons RS422, RS485

L'évolution temporelle des signaux RS422 et RS485 est identique aux signaux RS232 et liaison asynchrone décrits précédemment. La spécificité de RS422/485 tient dans l'adaptation en tension différentielle des signaux afin d'être transmis sur une distance supérieure (jusqu'à 1200 m).

3.4.1. Nature des signaux

Sur une liaison bidirectionnelle (sans contrôle de flux) de type 4 fils, il faudra :

- les 2 conducteurs d'émission des données Tx+, Tx-
- les 2 conducteurs de réception des données Rx+, Rx-
- le blindage

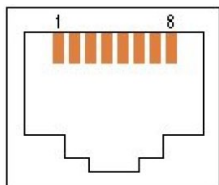
Sur une liaison bidirectionnelle (sans contrôle de flux) de type 2 fils, il faudra :

- le conducteur d'émission/ réception des données Tx/Rx+ polarité positive
- le conducteur d'émission/ réception des données Tx/Rx- polarité négative
- le blindage

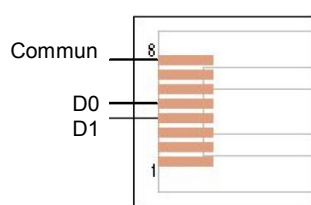
3.4.2. Connectique

Connecteur RJ45

Vue de face

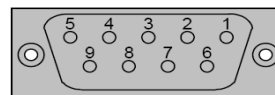


Vue de dessus

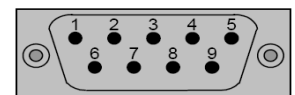


Connecteur Sub-D 9 broches

Femelle



Mâle



RJ45	Sub-DB9	Exigence	2 fils		4 fils		Description
			IDv	EIA/TIA 485	IDv	EIA/TIA 485	
1	8	requis			RxD0	A'	Receiver terminal 0, potentiel Va'
2	4	requis			RxD1	B'	Receiver terminal 1, potentiel Vb'
3	3	optionnel	PMC		PMC		Port Mode Control
4	5	requis	D1	B/B'	TxD1	B	Transceiver terminal 1, potentiel V1/Vb
5	9	requis	D0	A/A'	TxD0	A	Transceiver terminal 0, potentiel V0/Va
7	2	recommandé	VP				Alimentation positive 5..24VDC
8	1	requis	Common	C/C'	Common	C/C'	Commun d'alimentation et de signal

4. Le code ASCII

La connaissance du code ASCII est nécessaire afin de décoder les trames échangées sur le bus de communications.

4.1. Présentation

A.S.C.I.I. est l'abréviation de American Standard Code for Information Exchange. Ce codage consiste à associer une valeur numérique binaire (interprétable en hexadécimal, décimal, ...) à chacun des caractères utilisables dans l'échange de données informatique : caractères alphabétiques et numériques (alphanumérique), ponctuation, codes de contrôles divers.

Différentes variantes du code ASCII sont disponibles pour différentes langues. Il existe même une version Extended de ASCII où le 8ème bit de données est utilisé, ce qui permet de distinguer 2 fois plus de caractères, notamment les caractères accentués pour le français. On exprime phonétiquement ce sigle ASCII par le son « aski ».

4.2. Table de caractères

4.2.1. Codes hexadécimaux

F f	. 0	. 1	. 2	. 3	. 4	. 5	. 6	. 7	. 8	. 9	. A	. B	. C	. D	. E	. F
0 .	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1 .	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2 .		!	"	#	\$	%	&	'	()	*	+	,	-	.	/
3 .	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4 .	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5 .	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
6 .	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7 .	p	q	r	s	t	u	v	w	x	y	z	{		}	~	Del

4.2.2. Utilisation et conversions

Le caractère « A » a comme premier digit hexadécimal un « 4 » (colonne de gauche) et comme second digit hexadécimal un « 1 » (ligne du haut). Son code ASCII hexadécimal est donc la valeur hexa 41(h).

Ce même code ASCII en décimal donne :

Conversion en décimal :

Les puissances de 16		16^1		16^0	
valent		16		1	
associées à	×	4		1	
donnent	=	64	+	1	= 65

Conversion en binaire :

Chaque digit		4		1
est converti en binaire		0100		0001

4.2.3. Interprétation

Lorsqu'une donnée est visualisée, comme dans une table d'animation d'automate programmable, celle-ci peut être affichée et interprétée selon différents codages, indépendamment de son utilisation : en décimal, en hexadécimal, en binaire ou en caractère ASCII.

4.3. Caractères spécifiques de contrôle

Nom	Commande	Action	
Commandes de format			
CR	Carriage return	Retour chariot : retour en début de ligne	
LF	Line feed	Avancer d'une ligne : passage à la ligne suivante	
BS	Backspace	Espace arrière : suppression du caractère précédent	
HT	Horizontal tabulation	Tabulation horizontale : déplacement dans la ligne pour aligner le texte qui suit	
VT	Vertical tabulation	Tabulation verticale	
SP	Space	Espace	
FF	Form feed	Avancer d'une feuille : passer à la page suivante (NP = Nouvelle Page en français)	
Extension de code			
SO	Shift out		
SI	Shift in		
ESC	Escape	Début de séquence d'échappement	
Commande de séparation			
FS	File separator		
GS	Group separator		
RS	Record separator		
US	Unit separator		
EM	End of medium		
Commandes de communication synchrone			
SOH	Start of header	ACK	Positive acknowledge
STX	Start of text	NAK	Negative acknowledge
ETX	End of text	SYN	Synchronisation
EOT	End of transmission	DLE	Data link escape
ETB	End of transmission block	NUL	Null
ENQ	Enquiry		
Commandes de périphérique			
DC1	Device control 1 = X-ON	DC3	Device control 3 = X-OFF
DC2	Device control 2	DC4	Device control 4
Commandes diverses			
CAN	Cancel		
SUB	Substitute		
DEL	Delete	Supprime le caractère qui suit	
BEL	Bell	Emet un « bip » ou un autre avertissement sonore	

5.Application : gestion d'une imprimante au fil de l'eau

5.1.Objectif

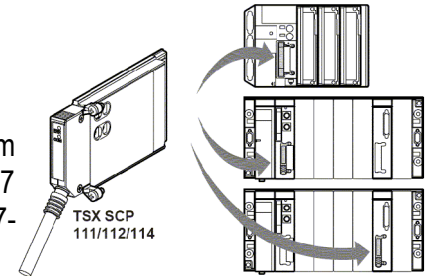
Générer une trace, un historique d'événements sur support papier, par l'envoi de messages « au fil de l'eau » vers une imprimante série ou munie d'un adaptateur parallèle / série.

5.2.Le matériel nécessaire

5.2.1.L'automate

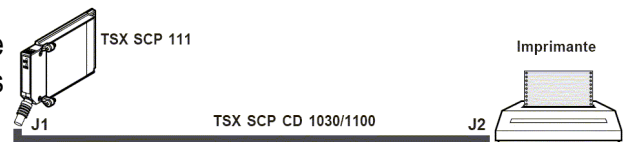
L'expérience est ici basée sur un API Schneider-Electric Telemecanique Premium TSX-57-3623, mais peut être tout API Schneider Premium TSX-57 ou Micro TSX-37 disposant d'un emplacement PC-Card (ex-PCMCIA) et programmable sous PL7-Junior/Pro, de préférence en langage structuré.

Les TSX-57 disposent pour la plupart d'un emplacement PC-Card sur le module CPU, mais on utilisera ici un module de communication additionnel TSX-SCY-21601, pouvant recevoir des cartes PCMCIA sur la voie 1.



5.2.2.L'imprimante

La manipulation est basée ici sur une (vieuse) imprimante matricielle à impact, à port parallèle, qui fait suffisamment de bruit pour intriguer les étudiants et les inciter à envoyer des données dessus.



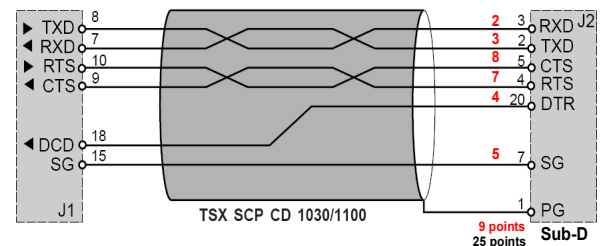
5.2.3.Les adaptateurs de communications

La communication entre l'API et l'imprimante série est assurée par une carte de communication RS-232 de référence TSX-SCP-111 associée à un câble TSX-SCP-CD-1030

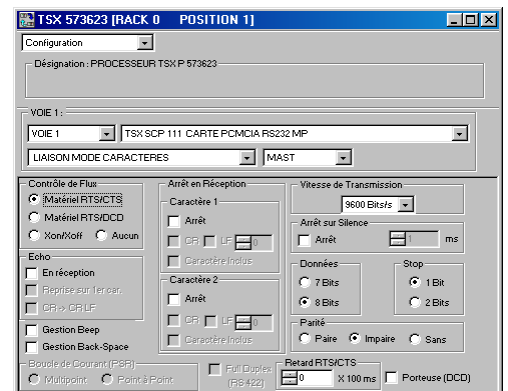
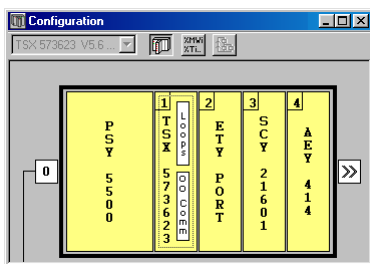
Un adaptateur série/ parallèle, auto-configurable, alimenté par pile 9V, fourni chez Radiospares sous la référence 215-167, est connecté sur le port Centronics de l'imprimante parallèle, celle-ci étant considérée ainsi comme une imprimante série RS232.



5.3.Raccordements



5.4.Configuration



5.5. Programmation

Instructions de base :

```
(*Remise a zero du tampon d'impression
%MW50 : Début du tampon d'impression
%MW98 : Index de pointage dans la table de mots
0..39 : longueur du buffer : 40 mots = 80 caractères
32 : Code ASCII du caractère [Espace] *)

FOR %MW98:=0 TO 39 DO
    %MW50[%MW98]:=32;
END_FOR;

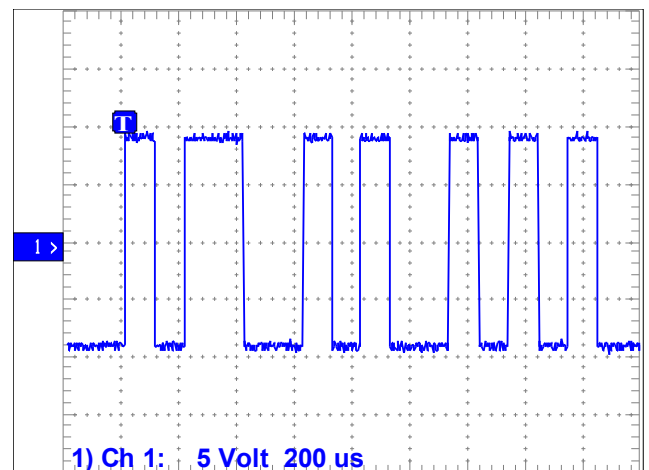
(*procedure envoi à l imprimante
%MW250 : Contrôle de communication (4 mots)
%MW253 : Dernier mot de contrôle = longueur de chaîne à émettre *)
(*)
%M250 : Copie du bit d'état de communication
%M11 : Bit de commande de l'envoi, doit être mis à 1
%MB100 : Début du tampon d'impression = %MW50
*)
%M250:=NOT %MW250:X0;
IF %M250 AND %M11 THEN
    %MW90:=16#0D0A;
    %MW253:=82; (* nbre de caractères à émettre*)
    PRINT_CHAR(ADR#{0}1.1.SYS,%MB100:82,%MW250:4);
    RESET %M11;
END_IF;
```

5.5.1. Mesures de signaux

Ici le caractère Y, poids faible en tête après le Start à 0 :

0100110101 → 01011001₍₂₎ → 59₍₁₆₎ → « Y »

suivi d'un autre caractère (partiel).



6.Application : gestion d'un API en esclave Modbus

6.1.Objectifs

- Utiliser un Automate Programmable Industriel d'entrée de gamme comme support d'entrées/ sorties déportées pilotées à partir d'un automate maître ou bien à partir d'un micro-ordinateur de type PC.
- Mettre en évidence les trames de communication afin d'être capable de diagnostiquer des défauts de communication lors d'opérations de mise en œuvre ou de maintenance

6.2.ModBus

6.2.1.Présentation

Le protocole Modbus est un protocole Maître-Esclave qui permet à un seul et unique maître de demander des réponses à des esclaves ou d'agir en fonction de la requête. 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 du maître n'attendent pas de réponses en retour.

Jusqu'à 32 nœuds peuvent résider sur un réseau RS485 (1 maître et jusqu'à 31 esclaves).

6.2.2.Protocole

- Le nombre de bits de la liaison asynchrone doit être de 8.
- La parité utilisée par défaut doit être paire (EVEN). Si l'on n'utilise aucune parité, on doit passer à 2 bits de STOP.
 - Toutes les spécifications de ModBus sont disponibles sur <http://www.modbus.org>
 - Pour voir les requêtes ModBus supportées par le TWIDO, voir l'aide en ligne de TwidoSoft et rechercher l'expression « Requêtes Modbus standard »

6.2.2.1Requête en mode RTU

- Trame :

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 sorties consécutifs : Param1 = Adresse, Param2 = Quantité
 - 02 : Lecture de n bits d'entrées consécutifs : Param1 = Adresse, Param2 = Quantité
 - **01 / 02 : Sur Schneider TSX-Nano et Twido : lecture de n bits internes %Mi consécutifs**
 - 03 : Lecture de n mots internes consécutifs : Param1 = Adresse, Param2 = Quantité
 - 04 : Lecture de n mots registres d'entrées consécutifs : Param1 = Adresse, Param2 = Quantité
 - **03 / 04 : Sur Schneider TSX-Nano et Twido : lecture de n mots internes %MWi consécutifs**
 - 05 : Écriture d'un bit interne ou de sortie : Param1 = Adresse, Param2 = 0000 ('0') ou FF00 ('1')
 - 06 : Écriture d'un mot interne ou registre : Param1 = Adresse, Param2 = Valeur
 - 15 : Écriture de n bits internes ou de sortie consécutifs :
Param1 = Adresse, Param2 = Nbre de bits, Param3 = Nbre d'octets, Params[j] = valeurs
 - 16 : Écriture de n mots internes ou registres consécutifs :
Param1 = Adresse, Param2 = Nbre de mots, Param3 = Nbre d'octets, Params[j] = valeurs
- 1er paramètre : adresse du bit ou mot adressé
- 2ème paramètre : 'quantité de mots adressés' ou 'valeur du bit ou mot écrit' selon la fonction utilisée
- Autres paramètres : données écrites dans plusieurs mots consécutifs
- CRC16 : code de redondance cyclique pour détecter les erreurs de transmission
- La fin de trame est détectée par une **absence d'émission** pendant une durée de **3 caractères** minimum

6.2.2.2 Réponse en mode RTU

➤ Fonctions 01, 02, 03, 04

N° esclave	Code fonction	Nbre de caractères	Données reçues		CRC16	
1 octet	1 octet	PF : 1 octet	N octets		PF : 1 octet	Pf : 1 octet

➤ Fonctions 05, 06,

N° esclave	Code fonction	Adresse affectée		Donnée écrite		CRC16	
1 octet	1 octet	PF : 1 octet	Pf : 1 octet	PF : 1 octet	Pf : 1 octet	PF : 1 octet	Pf : 1 octet

➤ Fonctions 15, 16

N° esclave	Code fonction	Adresse affectée		Nbre de données écrites		CRC16	
1 octet	1 octet	PF : 1 octet	Pf : 1 octet	PF : 1 octet	Pf : 1 octet	PF : 1 octet	Pf : 1 octet

6.2.2.3 Requête en mode ASCII

- Le nombre de bits de la liaison asynchrone doit être de 7.
- La parité utilisée par défaut doit être paire (EVEN). Si l'on n'utilise aucune parité, on doit passer à 2 bits de STOP.
- Le temps par défaut entre 2 caractères peut aller jusqu'à 1 seconde, sinon configurer pour des intervalles plus longs.

Début	N° esclave	Code fonction	1er paramètre	Autres paramètres	LRC	Fin
1 caractère '!' : \$3A	2 caractères '01' à 'F7'	2 caractères '01', '02', '03', '04', ...	4 caractères		2 caractères	2 caractères 'CR' : \$0D, 'LF' : \$0A

- Les fonctions utilisées sont identiques au mode RTU

6.2.2.4 Calcul du LRC

Somme en hexadécimal, modulo FF, du contenu de la trame, hors entêtes; complémentée à 2 et codée en ASCII.

Exemple : Écriture de la valeur \$1968 à l'adresse \$00A8 sur l'esclave N° 11

➤ Trame RTU

N° esclave	Code fonction	Adresse de la donnée		Valeur à écrire		CRC16	
0B	06	PF : 00	Pf : A8	PF : 19	Pf : 68	PF :	Pf :

➤ Trame ASCII équivalente

	Dép.	Esclave		Fonction		Adresse				Valeur				LRC		Fin de ligne	
Caractère	:	0	B	0	6	0	0	A	8	1	9	6	8	C	6	CR	LF
Code ASCII	3A	30	42	30	36	30	30	41	38	31	39	36	38	43	36	0D	0A

➤ Calcul du LRC

- $0B + 06 + 00 + A8 + 19 + 68 = 13A$ modulo FF = 3A = 0011 1010 b
- Complément à 1 : 1100 0101 b
- Complément à 2 : additionner 1 : 1100 0110 b
- Conversion en Hexadécimal : **C** **6**
- Codage en ASCII : 43 36


6.2.2.5 Réponse en mode ASCII

Les types de réponses sont les mêmes qu'en mode RTU, sur des trames ASCII

6.3. Le matériel nécessaire

6.3.1. L'automate

L'interface d'entrées/ sorties ModBus utilisée ici est un automate Schneider-Electric TWIDO Compact TWDLCAA24DRF. Cet API alimenté en 230V~ dispose de 14 entrées 24VDC et de 10 sorties relais.

 Le port RS485 intégré (N°1) sera utilisé pour le paramétrage et le développement du programme. Le TWIDO a alors été équipé d'un module d'adaptation RS485 supplémentaire sur le port N°2 de référence TWDNAC485D à sortie Mini-DIN 8 broches. Ce module n'est disponible que sur les automates 16 & 24 E/S compacts et pour le module d'expansion Afficheur.



Pour d'autres types de raccordements, on pourra utiliser le module TWDNAC485T sur bornier à vis ou TWDNAC232D en RS232 sur Mini-DIN 8 broches.

6.3.2. Le système maître

Le maître ModBus utilisé sera un **micro-ordinateur** de type **PC** équipé d'un port série RS232 en connecteur Sub-D 9 broches, sous un système d'exploitation **Windows** de la génération NT : NT, 2000, **XP** et supérieurs.

Les résultats ne sont pas garantis sous les versions précédentes Windows 9x et Millenium car le driver logiciel de communication série intégré est buggé dans la gestion des signaux de contrôle de flux. Tout autre système d'exploitation comme LINUX peut être utilisé pour cette expérimentation en utilisant les logiciels de communication adaptés.

Les nouveaux **micro-ordinateurs portables** ne sont plus équipés de ports série RS232. Des **adaptateurs USB->Série** sont alors proposés. Malheureusement, si ceux-ci remplissent à peu près normalement leur fonction pour des communications standard RS232, ils sont quasiment **inutilisables pour** des communications **RS485 sur 2 fils**. En effet, le RS485 sur 2 fils est un mode semi-duplex, dans lequel l'émetteur et le récepteur prennent la ligne à tour de rôle. La prise de ligne est assurée par l'activation du signal de demande d'émission RTS. Les adaptateurs USB->Série laissent le RTS en permanence à l'état actif, ce qui induit un court-circuit entre l'émetteur maître (PC) et le récepteur esclave (TWIDO) lorsque ce dernier répond.

6.3.3. Connexion d'un PC à un automate TWIDO

Le raccordement entre le port RS232 du PC et le port RS485 du TWIDO est du même type que pour la programmation, à l'aide du câble adaptateur TSXPCX1031, dont les différentes fonctions du mode série sont sélectionnées selon la position du commutateur sur le convertisseur :

<i>Position de l'interrupteur Rotary</i>	<i>Fonction</i>	<i>Signal /DTP</i>	<i>Signal RTS</i>
0	TER MULTI – Connexion en mode point à point. Force le port du terminal en mode maître, protocole par défaut (remplace la câble réf. TSX PCU 1031)	1	oui
1	OTHER MULTI – Connexion en mode multipoint. Autres types de communication (remplace le câble réf. TSX PCD 1030)	0	oui
2	TER DIRECT – Connexion en mode point à point. Force le port du terminal en mode maître, protocole par défaut (remplace la câble réf. TSX PCX 1030, basculé en position maître et TSX PCU 1030)	1	non
3	OTHER DIRECT – Connexion en mode point à point. Autres types de communication définis par la configuration de l'automate (remplace le câble réf. TSX OXC 1030, basculé en position esclave)	0	non

La longueur maximale des connexions RS485 non isolées sur les automates TWIDO est de 200m.

Attention : Dommages électriques potentiels de l'automate :

Ne connectez pas le câble de communication à l'automate avant de la connecter au PC.

Connectez toujours le câble au PC en premier.

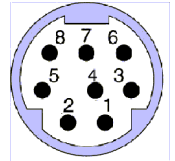
6.3.4.Limitations de la couche Modbus du Twido

- Adresses des esclaves : 1 à 247
- Bits : 128 bits sur demande
- Mots : 64 mots de 16 bits sur demande

6.3.5.Réalisation d'un cordon de mesure des signaux RS485

➤ Signaux utiles

La documentation en ligne de TwidoSoft fournit le brochage des connecteurs mini-DIN à 8 broches :



Pour assurer la liaison Modbus, il suffit de câbler les signaux RS485 A(+) et B(-). Cependant, le cordon convertisseur TSXPCX1031 est alimenté par le Twido, il est alors nécessaire de permettre le passage de l'alimentation par le 0V et le 5V. D'ailleurs, les seules broches câblées sur le module option RS485 sont ces 4 broches de N° 1, 2, 7 et 8.

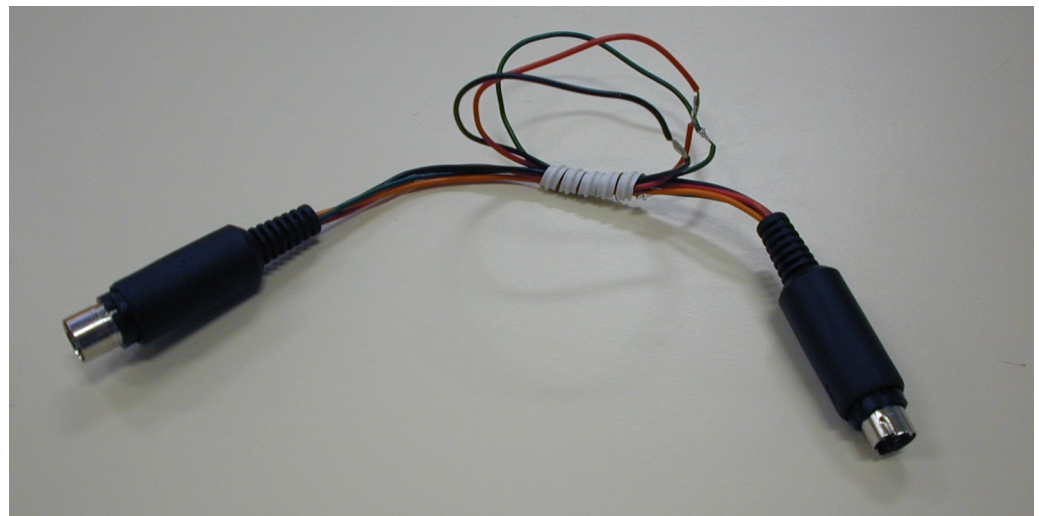
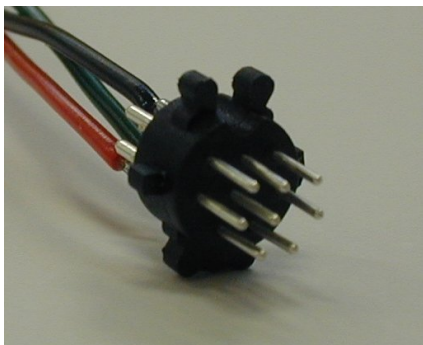
Broches	Base RS485	Option RS485	Option RS232-C
1	A (+)	A (+)	RTS
2	B (-)	B (-)	DTR
3	nc	nc	TxD
4	/DE	nc	RxD
5	/DPT	nc	DSR
6	nc	nc	Gnd
7	0 V	0 V	Gnd
8	5 V	5 V	5 V

➤ Réalisation du cordon de mesure

Il est très difficile de trouver un fournisseur de cordons prolongateurs mini-Din 8 broches (Mâle / Femelle), desquels on aurait extrait les conducteurs 1 et 2. Seul Apple aurait utilisé ce type de câbles pour des liaisons d'imprimante série.

Il faudra donc se procurer un connecteur mâle et 1 connecteur femelle sur lesquels on soudera les 4 conducteurs. Fort heureusement, la broche N° 4 n'est pas à souder. Les broches 1, 2, 7 et 8 sont bien accessibles.

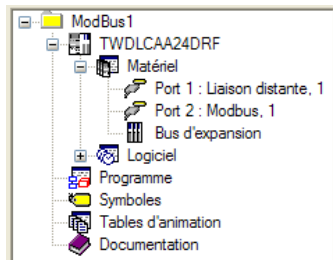
➤ Quelques photos de la réalisation



Les conducteurs ont été dénudés en leur milieu afin d'y attacher les pinces Grip-fil de l'oscilloscope, et d'observer le signal RS485.

6.4. Configuration

6.4.1. L'automate



Il existe deux types de périphériques ModBus compatibles avec TwidoSoft :

Maître : transmet une requête ModBus et demande des réponses aux périphériques esclaves. Compatible avec l'instruction EXCH.

Esclave : répond aux requêtes ModBus d'un maître ModBus.

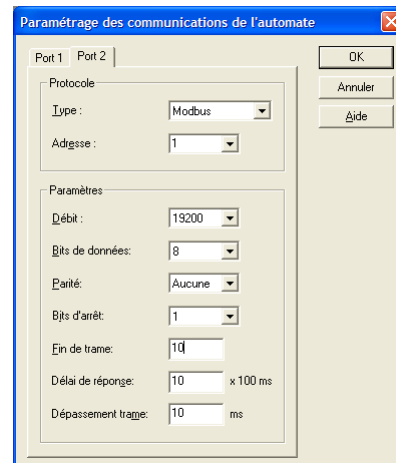
Le TWIDO ne nécessite aucune programmation pour devenir esclave ModBus, simplement une configuration.

Les automates Twido gèrent l'utilisation des modes ModBus ASCII et ModBus RTU. Le nombre de bits de données sélectionnés dans la boîte de dialogue **Paramétrage des communications de l'automate** détermine le mode activé :

8 bits de données : protocole **ModBus RTU**

7 bits de données : protocole **ModBus ASCII**

- Connecter le PC sur le port de communication TER (N°1) ;
- Lancer TwidoSoft et créer une nouvelle application ;
- Configurer le port N°2 en Modbus RTU (8 bits) ou ASCII (7 bits) à la vitesse de votre choix : ex. 9600 ou 19200 bps ;
- Créer un programme de base qui permet d'agir sur les variables internes et sur les E/S :
 - Recopier les 14 entrées %I0.0 à %I0.13 dans les bits 14 internes %M0 à %M13
 - Recopier les 10 bits internes %M16 à %M25 sur les 10 sorties %Q0.0 à %Q0.9
 - Placer les 4 constantes suivantes dans les mots %MW0 à %MW3 : 0x1234, 0x5678, 0x1000, 0xABCD
 - Incrémenter %MW4 chaque 1/10e de seconde
 - Incrémenter %MW6 sur chaque front montant de %I0.0



```
(* IMAGES ENTRÉES / SORTIES DANS BITS INTERNES *)
LD 1
[ %M0:14 := %I0.0:14 ]
[ %Q0.0:10 := %M16:10 ]
(* AFFECTATION CONSTANTES DANS 4 MOTS *)
LD 1
[ %MW0 := 16#1234 ]
[ %MW1 := 16#5678 ]
[ %MW2 := 16#1000 ]
[ %MW3 := 16#ABCD ]
(* DÉTECTION FRONT MONTANT %S5 (CHAQUE 1/10E SECONDE) DANS %M31 *)
LD %S5
ANDN %M30
ST %M31
LDN %S5
ST %M30
(* INCRÉMENT %MW4 CHAQUE 1/10E SECONDE *)
LD %M31
[ %MW4 := %MW4 + 1 ]
(* INCRÉMENT %MW6 SUR CHAQUE MISE À 1 DE L'ENTRÉE 0 *)
LDR %I0.0
[ %MW6 := %MW6 + 1 ]
```

- Enregistrer l'application
- Connecter logiquement l'API et y transférer le programme
- Placer l'automate en mode RUN
- Quitter TwidoSoft

6.4.2. Le cordon de communications

Une connexion multipoint est mise en oeuvre entre le port COM RS232 du PC et le RS-485 du Twido. Le commutateur du TSXPCX1031 sera configuré en position OTHER-MULTI = 1. La connexion multipoint au lieu de point-à-point permet au PC de ne pas attendre les signaux CTS et DSR de la part du Twido Esclave.

6.4.3. L'utilitaire Modbus Maître

Pour les liaisons RTU, télécharger sur Internet un logiciel d'interrogation Modbus maître. On peut trouver :

- **ModbusView** sur <http://www.saphir.fr> ou <http://www.zdnet.fr>
- **ModbusView** sur <http://www.oceancontrols.com.au>
- **ModScan32** sur <http://www.win-tech.com> , en évaluation 30 jours
- **ModBus Request Pro** sur <http://www.soft-control.fr/articles.php?id=11>
- ...

Configurer la communication avec le même protocole que celui défini dans le Twido :

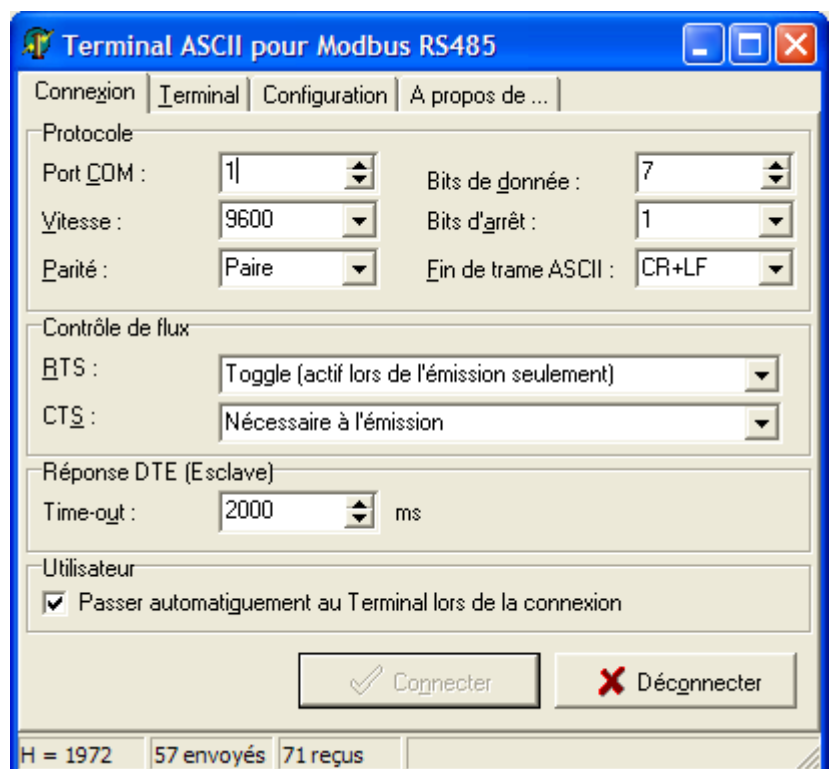
- ModBus RTU (8 bits) ou ASCII (7 bits), parité paire, 1 bit de stop, contrôle de flux matériel gérant le RTS, ...

Les utilitaires spécialisés Modbus sont programmés de manière à gérer le RTS du port série en mode Toggle, c'est-à-dire d'activer le RTS automatiquement à l'émission. A ce moment-là, la ligne RS485 quitte le mode réception (haute impédance) et est mise en charge sous +/- 5V différentiels.

Il aurait été intéressant pour nos manipulations de trouver un utilitaire de Terminal ASCII permettant de construire et d'envoyer caractère par caractère les trames Modbus. Le logiciel intégré à Windows HyperTerminal paraît au premier abord intéressant, mais celui-ci ne permet pas la configuration du RTS en mode Toggle comme décrit au § 6.3.2. Il laisse le RTS en permanence à l'état actif.

J'ai donc entrepris d'écrire mon propre utilitaire de Terminal Modbus ASCII à l'aide de l'outil de développement en Pascal sous Windows : « Borland Delphi ». Celui-ci est fourni en accompagnement de ce document, et peut être utilisé librement dans tout établissement d'enseignement public.

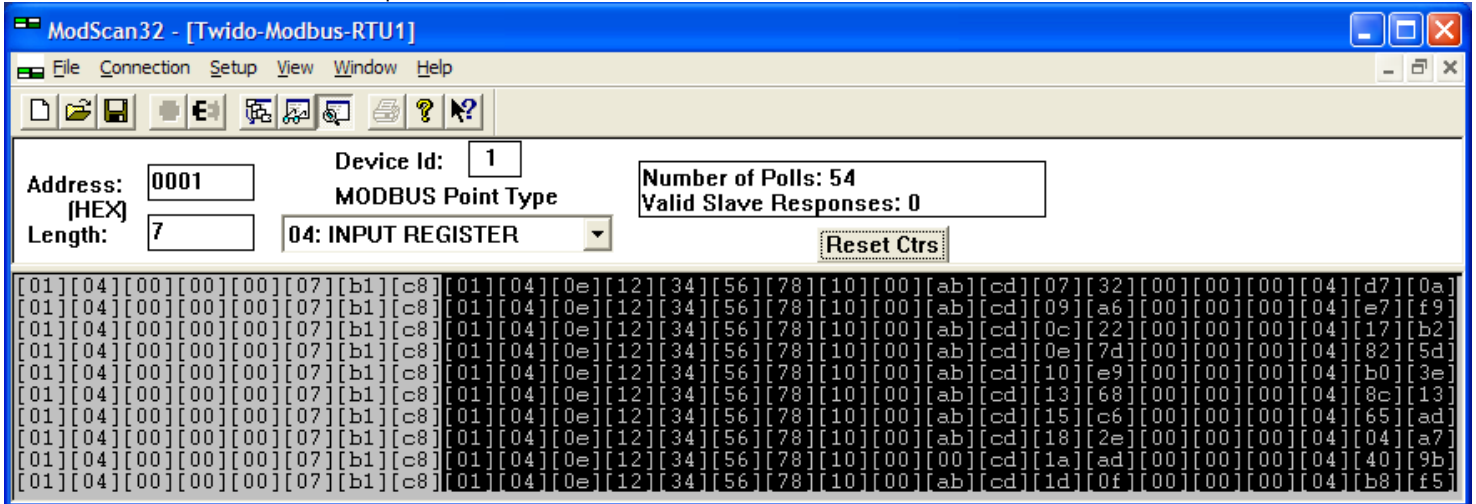
L'écran de configuration ci-contre montre les options de configuration par défaut et adaptés pour les manipulations.



6.5.Manipulations

6.5.1.Communications RTU ou ASCII avec un utilitaire spécialisé Modbus

- Sélectionner une trame d'après sa fonction d'interrogation :
 - Fct 01 / 02 : lire 14 bits aux adresses %M0 à %M13 renvoie l'état des entrées %I0.0 à %I0.13
 - Fct 03 / 04 : écrire 3 bits '101' aux adresses %M34 à %M36 affecte l'état des sorties %Q0.2 à %Q0.4
 - Fct 05 / 06 : lire la valeur des mots %MW0 à %MW6
 - Fct 15 / 16 : écrire d'autres valeurs dans les mots %MW0 à %MW3, puis vérifier l'écriture par une lecture
- Afficher la trame émise, noter les résultats obtenus et analyser les données.
 - Ci-dessous un exemple de lecture de %MW0 à %MW6 avec l'utilitaire ModScan32 :

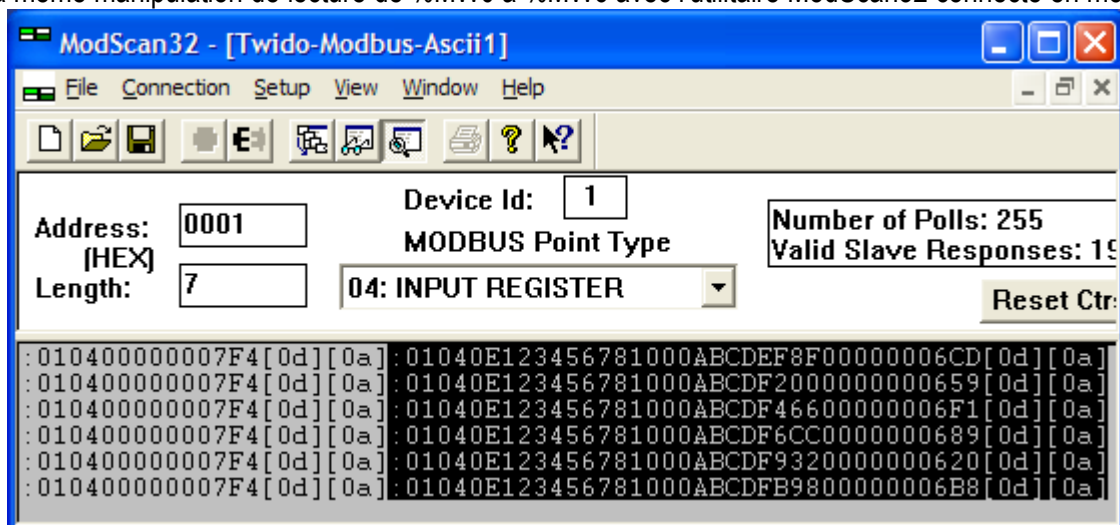


On peut reconnaître les données du Twido :

- Les constantes écrites dans le programme : 1234h, 5678h, 1000h, ABCDh.
- Le mot suivant %MW4 change à chaque requête, puisqu'il est incrémenté chaque 1/10e de seconde.
- Le dernier mot correspondant à %MW6 vaut 4 car il y a eu 4 manoeuvres de l'entrée %I0.0.

A noter que ModScan32 décale l'adresse mémoire : son adresse locale 1 correspond à interroger l'adresse distante 0.

- Basculer le TWIDO en mode Modbus ASCII (par TwidoSoft) ; effectuer la même manipulation de lecture de %MW0 à %MW6 avec l'utilitaire ModScan32 connecté en mode ASCII :

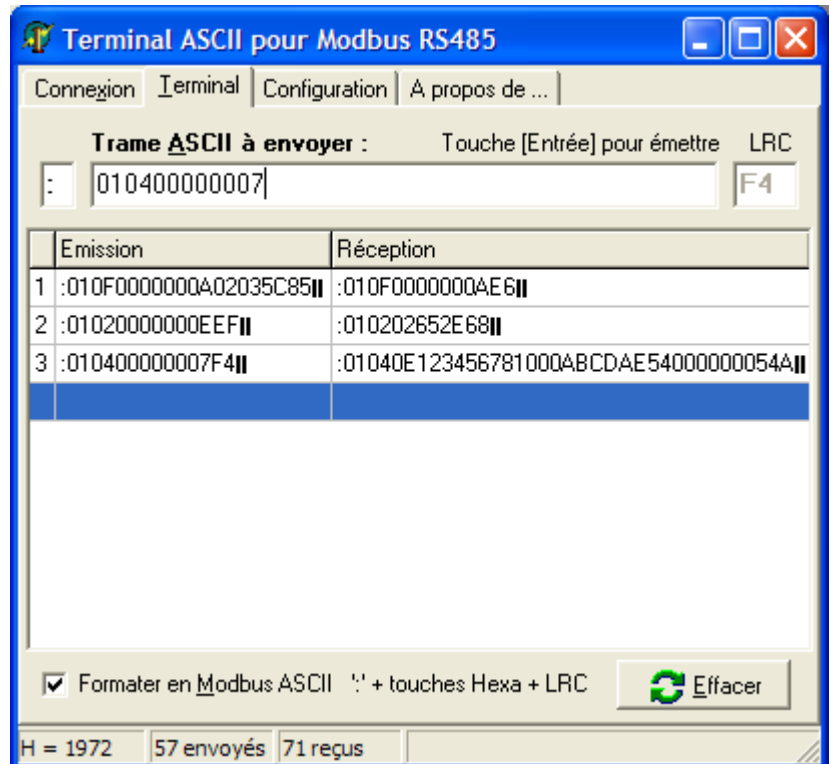


On peut noter :

- Le caractère « : » de début de trame
- les 2 caractères de fin de trame : fin de ligne [0D] et de saut de ligne [0A]
-

6.5.2. Communication ASCII avec le Terminal dédié

- Saisir des trames Modbus dans la zone d'édition, et appuyer sur [Entrée] pour l'émettre.
 - Lorsque la case « Formater en Modbus ASCII » n'est pas cochée, l'étudiant doit calculer lui-même le LRC et le saisir en fin de chaîne.
 - Lorsque cette case est cochée, seuls les caractères Modbus autorisés sont édités, et le LRC est calculé et envoyé automatiquement.
- Les 3 trames présentées ci-contre sont :
 - écriture de 10 bits à partir de %M10
 - Erratum : selon le programme Twido fourni, il faudrait écrire en %M32 afin d'activer les sorties.
 - lecture de 14 bits à partir de %M0
 - lecture de 7 mots à partir de %MW0



6.5.3. Mise en oeuvre d'un analyseur de réseau RS232

Un analyseur de réseau asynchrone permet de mettre en évidence les signaux de contrôle de flux échangés, mais aussi d'afficher en clair les trames de données. Nous avons choisi ce modèle :

Thurlby DA100 + DA101 + DA102 : <http://www.mbelectronique.fr/html/uploads/prodfr1529.htm>

de Thurlby Thandar Instruments : <http://www.ttinst.co.uk/languages/home-french.htm>

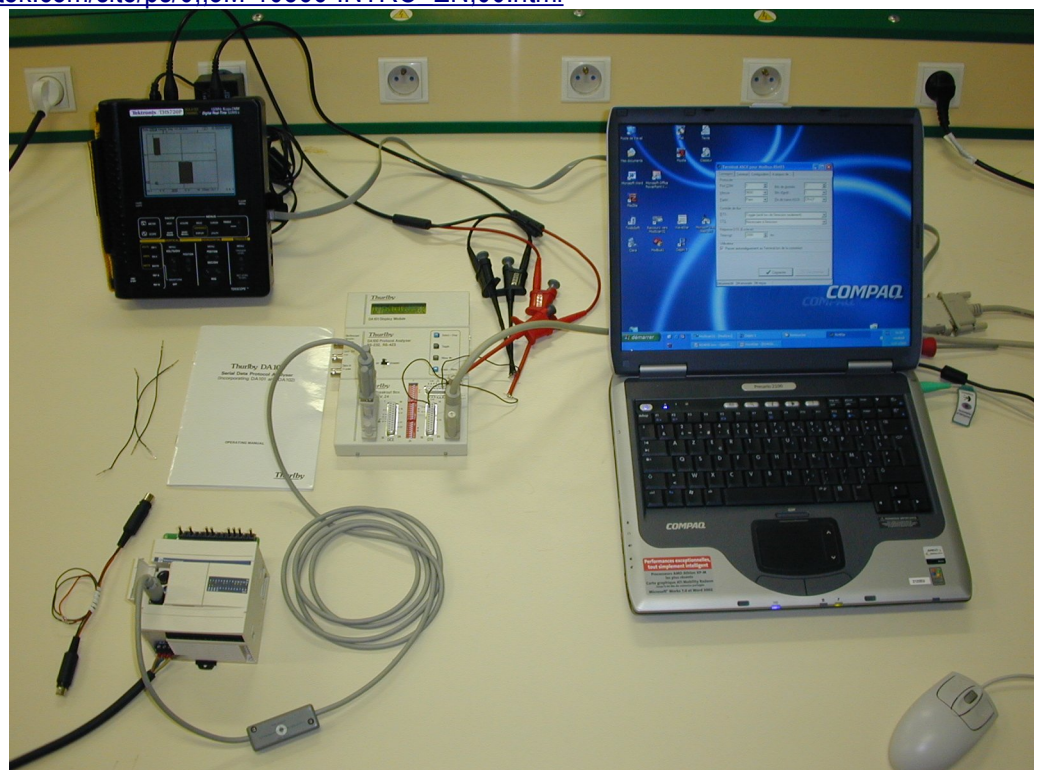
On peut le trouver chez le distributeur Radiospares, qui propose l'ensemble DA100/101/102 pour environ 350 à 400 €.

Cet analyseur met à disposition l'ensemble des signaux de contrôle RS232 qui peuvent alors être visualisés à l'oscilloscope. Celui-ci doit être un modèle à entrées isolées afin d'avoir des masses indépendantes entre les signaux. Ainsi une voie pourra afficher la ligne RS232, tandis que la seconde affichera les données RS485. Nous disposons du modèle

Tektronix THS720P : http://www.tek.com/site/ps/0_3M-10566-INTRO_EN.00.html

L'ensemble du banc de manipulations peut alors ressembler à l'illustration ci-contre :

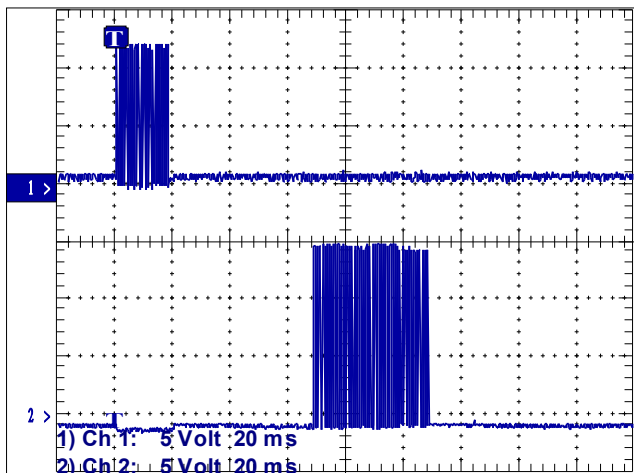
- Sortie RS232 du PC
 - ➡ Analyseur de réseau
 - ➡ Adaptateur RS485
 - ➡ API Twido
- Analyseur de réseau
 - ➡ Oscilloscope



6.5.4. Relevé de signaux RS232 à l'oscilloscope

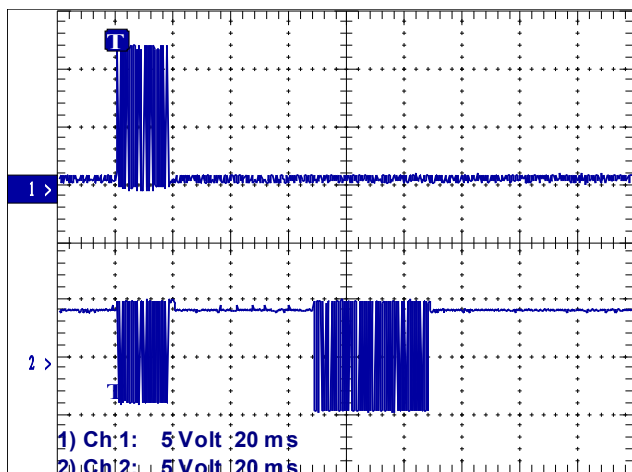
Sur ce premier relevé, on trouve le signal **RTS** issu du PC, qui permet à l'adaptateur RS485 de prendre la ligne en charge, et d'émettre les données ici représentées par le signal **TxD**.

On peut noter les niveaux de tension RS232 en logique négative qui sont d'environ 0,6V pour le niveau bas (1 logique) et environ 12V pour le niveau haut (0 logique).

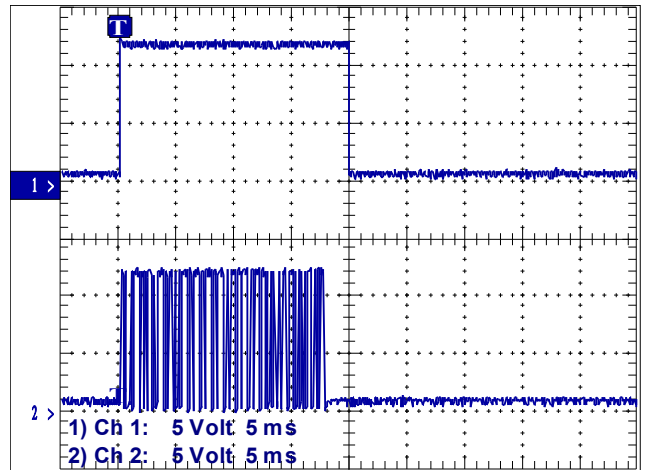


Sur ce troisième relevé, on trouve une suite d'échanges question/réponse, à un intervalle d'environ 1 seconde entre une réponse et la nouvelle requête.

6.5.5. Relevé de signaux RS485 à l'oscilloscope

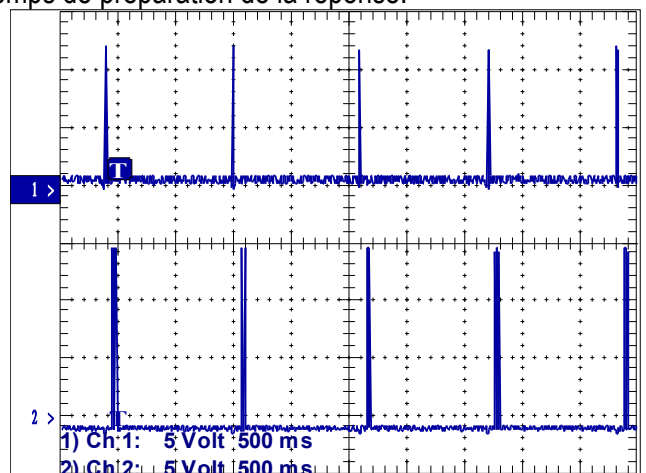


Enfin une dernière association RS232/ RS485 pour le premier caractère « : » de la trame, de code ASCII « 3A ». Le caractère a une durée d'environ 1,1 ms à compter du bit de Start qui apparaît à la première division de l'écran.



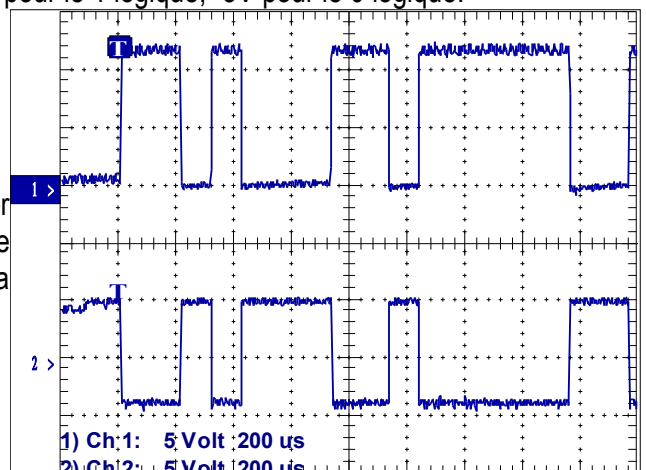
Le deuxième relevé montre l'échange question/ réponse avec en haut la trame de question du PC (**TxD**), en bas la trame de réponse de l'API (**RxD**).

On peut noter le temps écoulé entre la question et la réponse (environ 50ms), qui représente le temps de pause signifiant la fin de trame (3 caractères ~ 3 ms à 9600bps), + le temps de décodage de la question, + le temps de préparation de la réponse.

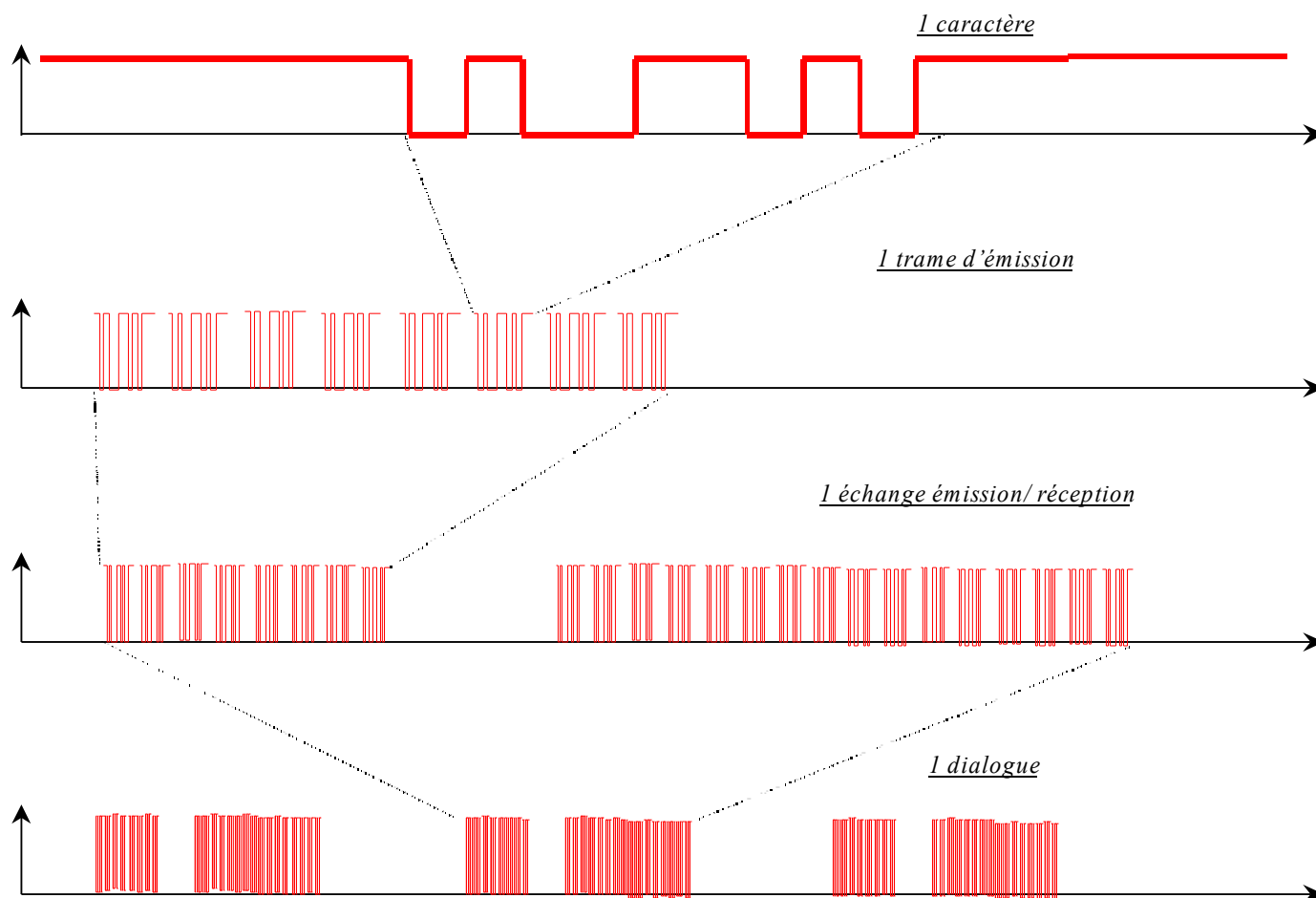


Ici la voie 1 représente la même émission TxD en RS232.

La voie 2 montre la ligne RS485 qui supporte l'émission TxD et la réception RxD en mode différentiel, c'est à dire en niveaux de tension +5V pour le 1 logique, -5V pour le 0 logique.



6.6.Synthèse du dialogue



7.Des liens

7.1.Normes

http://www.camiresearch.com/Data_Com_Basics/RS232_standard.html

http://bb-elec.com/technical_library.asp

7.2.Schémas de convertisseurs RS232 / RS485

http://sjeffroy.free.fr/Divers/Convertisseur_RS232-RS485/convertisseur_rs232-rs485.html