CPE LYON

LIVE AND DISCOVER

4ETI Bases des systèmes embarqués

A05 – Généralités - UART

Version 2022 – Ver 25/11/2022 13:59

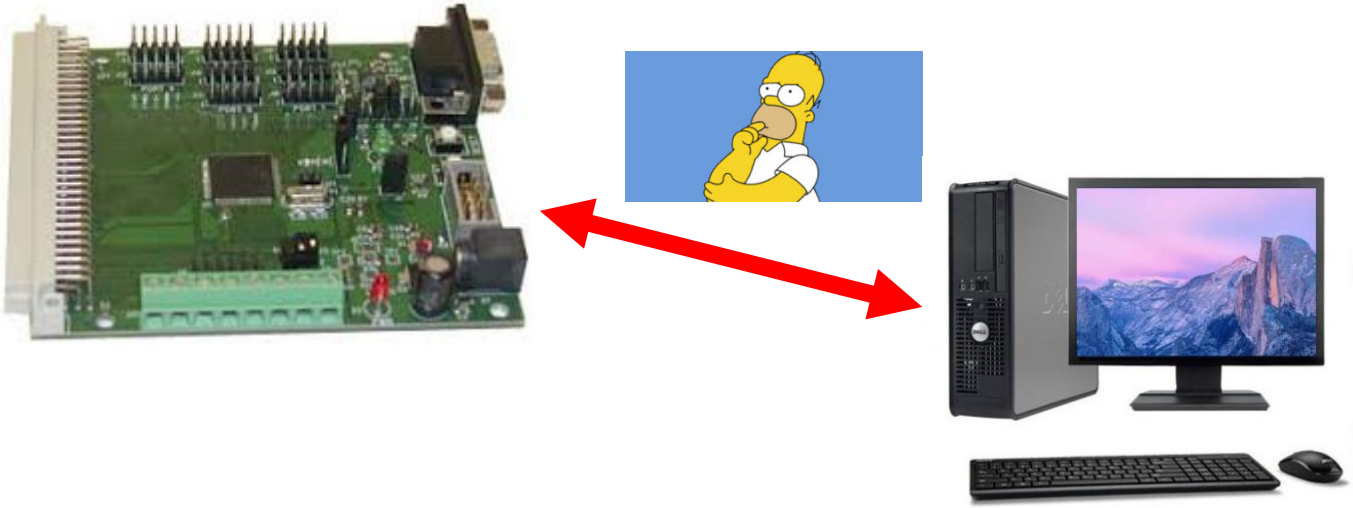
UNIVERSITE DE LYON
établissement associé

UART – USART Quel usage?

Comment établir une communication **simple** entre un microcontrôleur et un micro-ordinateur?
Simple veut dire:

1. Un nombre de signaux à échanger le plus faible possible
2. Un protocole d'échange d'information réduit à sa plus simple expression
3. Utilisation d'interfaces matérielles déjà disponibles sur le micro-ordinateur
4. Coté micro ordinateur, possibilité d'utiliser des applications élémentaires de type terminal de commande pour communiquer

Pourquoi une liaison simple: pour quelle soit gérable pour un micro contrôleur de faible puissance. Ce type de liaison est très aussi pratique dans des phases de débogage.

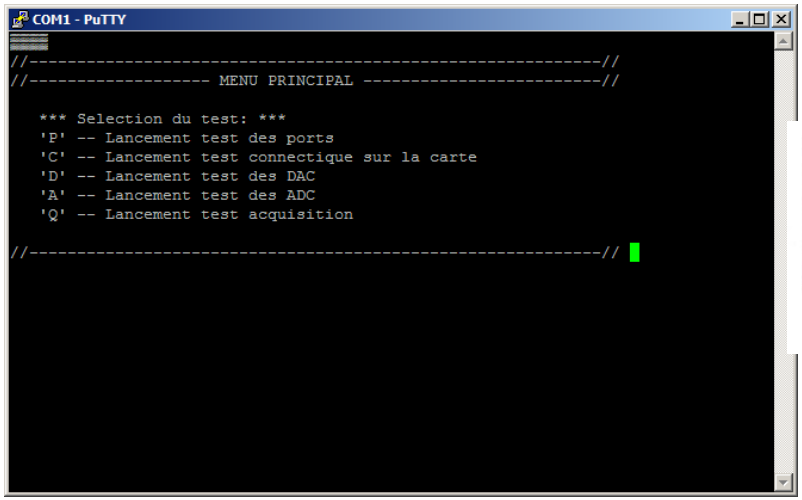


UART – USART Communication UC <-> PC

Exemple de communication simple au niveau du PC: **la liaison série RS232:**

- Condition 1: respectée – dans sa version simplifiée la communication s'établit sur 2 fils, un signal de transmission, et un signal de réception, (sans oublier la liaison équipotentielle 0V sur une « troisième » fil)
- Condition 2: respectée – aucun protocole, tout octet échangé est un octet de données
- Condition3: respectée (avec une réserve). Les liaisons RS232 équipaient tous les PC, il y a encore quelques années. Aujourd'hui, des adaptateurs USB existent pour émuler côté PC cette liaison série
- Condition 4: respectée - Coté micro ordinateur, la communication peut se faire via un logiciel terminal de commande

NB: la liaison série RS232
est une liaison asynchrone



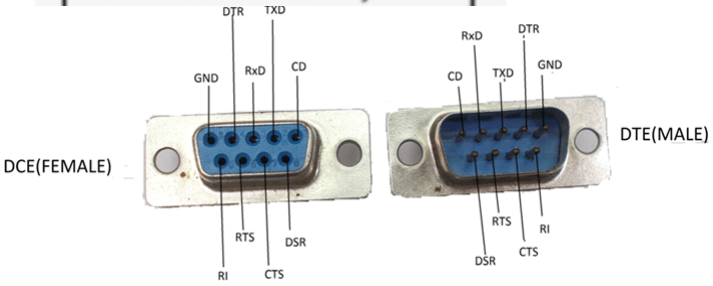
Un terminal de commande sur PC – Exemple: Putty

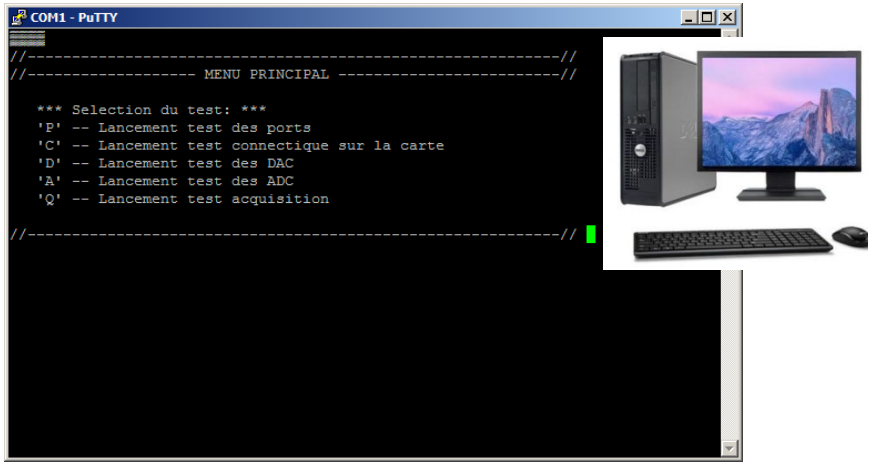
Une application Terminal de commande sur PC, telle que Putty peut assurer la communication sur un port série RS232.
Lorsqu'elle est configurée en mode « Serial », alors l'application fonctionne de la manière suivante:

- A chaque appui sur une touche du clavier, le code ASCII (1 octet) de la touche appuyée est envoyé sur la liaison série. Ex: lors d'un appui sur la touche 'A' (majuscules activées), l'octet de valeur 0x41 est transmis sur la liaison série.
- Inversement, à chaque fois qu'un octet est reçu via la liaison série, on affiche sur l'écran le caractère équivalent au code ASCII. Ex: si on reçoit un octet de valeur 0x48, alors sur l'écran s'affiche le caractère 'H'

Dec	Hex	Oct	Char
64	40	100	@
65	41	101	A
66	42	102	B
67	43	103	C
68	44	104	D
69	45	105	E
70	46	106	F
71	47	107	G
72	48	110	H
73	49	111	I
74	4A	112	J

Extrait de la table ASCII





UART – USART - Définition

Côté micro contrôleur: c'est le périphérique UART/USART qui sera chargé de gérer ces communications en mode série.

UART = Universal Asynchronous Receiver-Transmitter

USART = Universal Synchronous Asynchronous Receiver-Transmitter

Rôle de ce périphérique: Recevoir et transmettre des informations sous forme série.

Intérêt: minimiser le nombre de fils (1 fil au lieu de 8)

Inconvénient: débit limité (par rapport à une transmission en parallèle)

Usage: échange point à point entre 2 « systèmes » de signaux de commande et/ou d'informations

Universal: divers modes possibles – nombre de bits, parité, vitesse….

Synchronous: l'horloge est transmise, soit au travers d'un signal supplémentaire, soit elle est associée à l'information transmise.

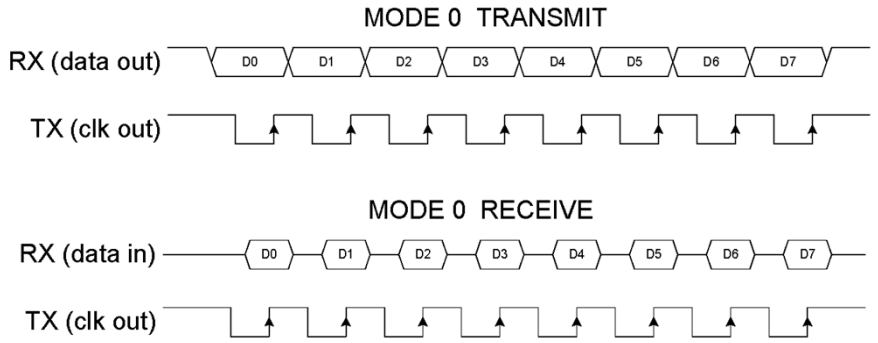
Asynchronous: transmission asynchrone, il n'y a pas de transmission de l'horloge aussi bien à l'aide d'un signal spécifique, ou associée à l'information transmise

Receiver: peut assurer la réception d'informations

Transmitter: peut assurer la transmission d'informations

Transmission synchrone

Transmission synchrone – Horloge transmise sur une ligne supplémentaire – Le récepteur peut récupérer le signal d'information en utilisant les fronts d'horloge

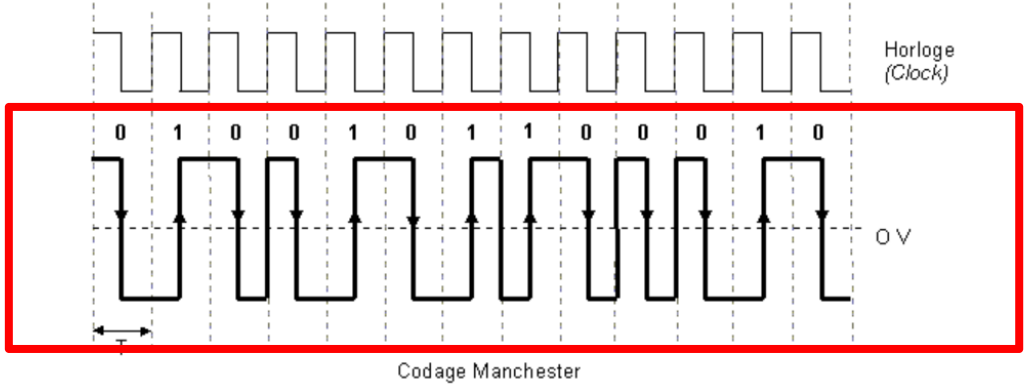


Signal

Horloge

SILICON LABS

Transmission synchrone – Horloge transmise à l'intérieur du signal – Le récepteur peut régénérer l'horloge et n'a pas besoin d'avoir d'informations sur la vitesse de transmission – Ex du codage Manchester (Bi-phase)



Codage Manchester

Signal + Horloge

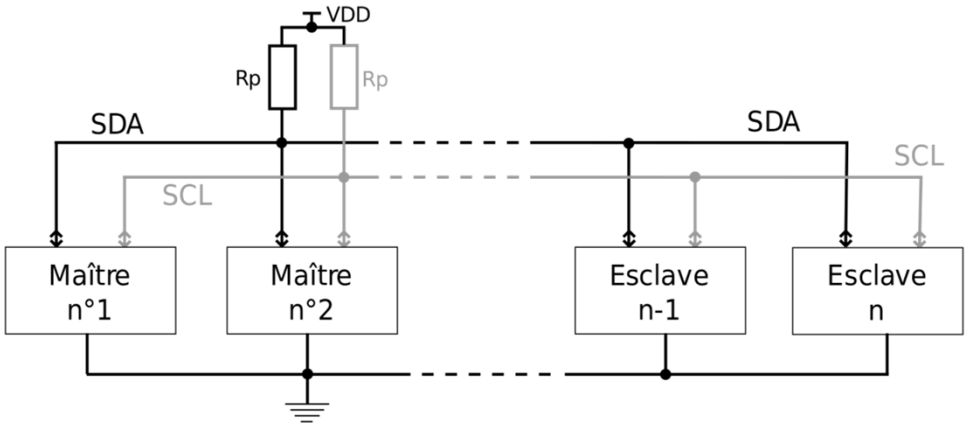
CPE LYON LIVE AND DISCOVER

Exemple de communication à bus synchrone: I2C (SMbus-TWI)

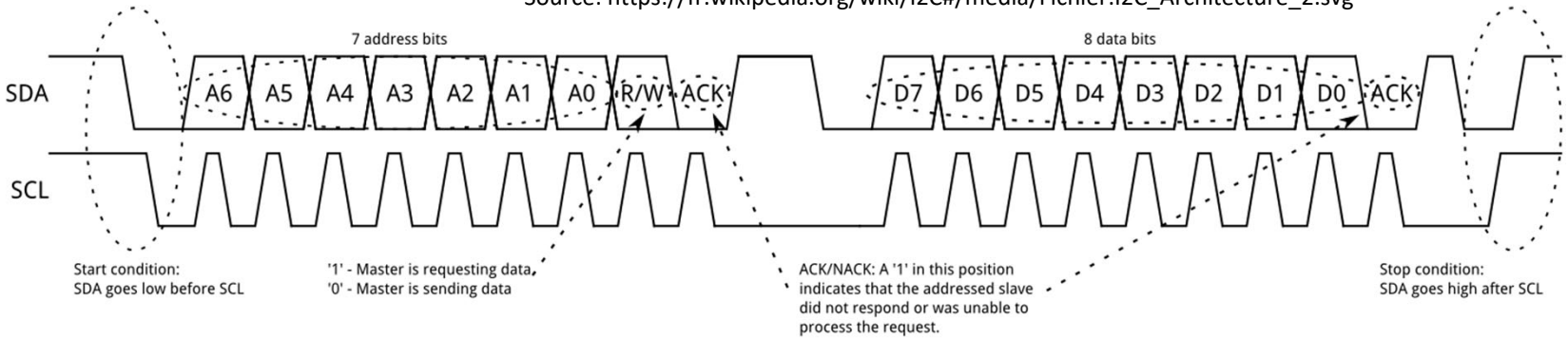
I2C comme Inter Integrated Circuit (Philips 1982)

C'est bus série synchrone bidirectionnel half-duplex à 2 fils: 1 signal d'horloge SCL et 1 signal de données SCL

Ce bus de communication est en général géré par un périphérique dédié.



Source: https://fr.wikipedia.org/wiki/I2C#/media/Fichier:I2C_Architecture_2.svg

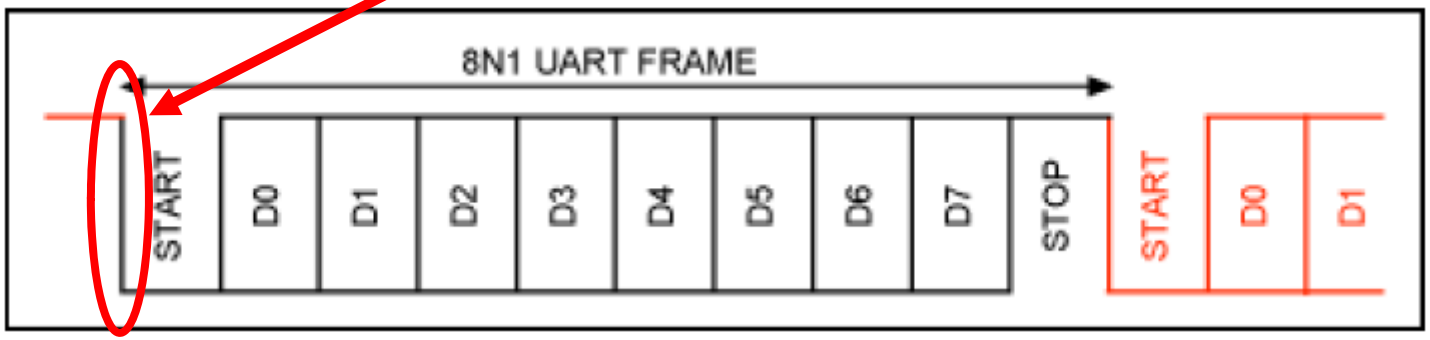


Source: https://learn.sparkfun.com/tutorials/i2c/all

Transmission Asynchrone

Dans ce mode, l'horloge n'est pas transmise entre émetteur et récepteur, le récepteur doit donc connaitre **précisément** la vitesse de transmission et doit être en mesure de se resynchroniser sur le signal reçu.

Transmission asynchrone: Premier Front descendant (bit start): le récepteur se « cale » dessus pour décoder -

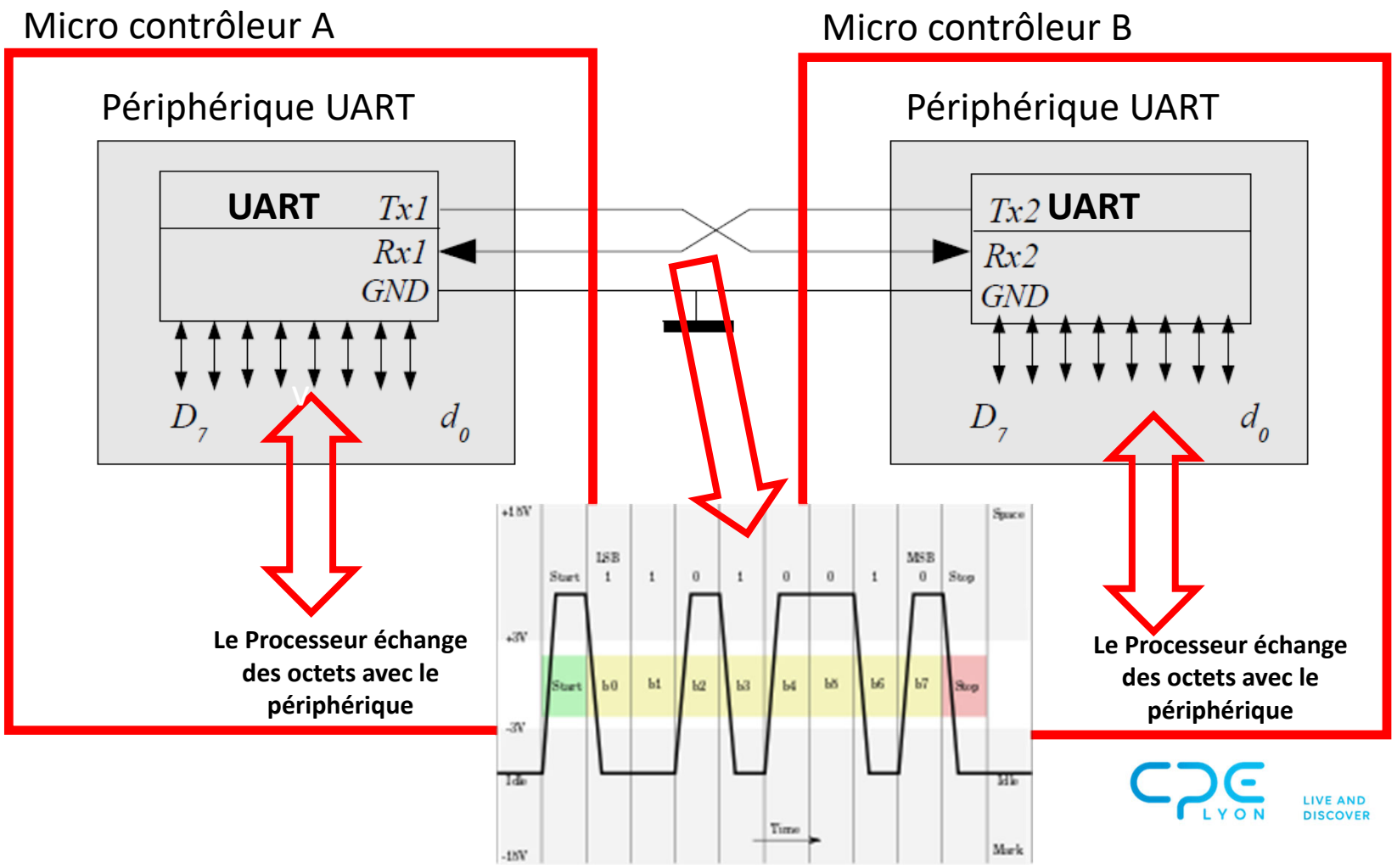


Source: Maxim Integrated – Tutorial 2141

Exemple de transmission typique d'un octet dans une UART
La trame est constitué d'un bit de start, 8 bits de donnée et un bit de Stop. 1 + 8 + 1 = 10 bits

Liaison UART et Processeur

Le processeur échange les informations avec le périphérique UART au travers de registres interfacés dans l'espace mémoire entrées-sorties du processeur (espace SFR pour le 8051)



La norme RS232 – Communication asynchrone

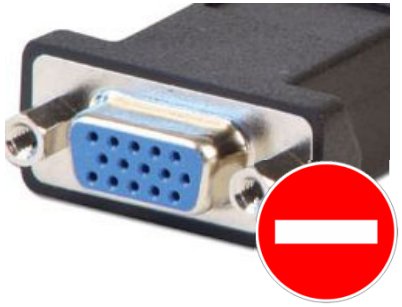
La norme RS232 (1962) est une norme de communication série développée à l'origine pour la télégraphie.

Elle s'appuie sur les liaisons séries « UART » asynchrones

Jusqu'à un passé récent, chaque PC était équipé d'au moins une interface RS232 (Port Série COMn)

Même si elle est obsolète de nos jours, elle est encore couramment utilisée en mode Debug (Mise au point) à cause de sa simplicité et sa rusticité.

Ne pas confondre:
Connecteur Série COMn
Et connecteur VGA!!



Même sans port série, il est possible de connecter une liaison UART à un PC, au travers d'un port USB émulant un port COM
Utilisation d'un câble d'adaptation USB/RS232 ou USB/UART

La norme RS232 - Spécifications

RS-232 Specifications -

- **Communication Asynchrone à 3 fils minimum: TX – RX - GND**
- **TRANSMITTED SIGNAL VOLTAGE LEVELS:**
  - **Binary 0: +5 to +15 Vdc** (called a “space” or “on”)
  - **Binary 1: -5 to -15 Vdc** (called a “mark” or “off”)
- **RECEIVED SIGNAL VOLTAGE LEVELS:**
  - Binary 0: +3 to +13 Vdc
  - Binary 1: -3 to -13 Vdc
- **DATA FORMAT:**
  - **Start bit**: **Binary 0**
  - Data: 5, 6, 7 or 8 bits
  - Parity: Odd, even, mark or space (not used with 8-bit data)
  - **Stop bit**: **Binary 1**, one or two bits

A l’opposé des signaux logiques habituels:

- Valeur binaire 0: 0V
- Valeur binaire 1: VCC

Vitesse de transmission et RS232

Pour exprimer la vitesse de transmission en RS232, on utilise souvent le **Baud** comme unité.

Le Baud est une unité qui exprime la rapidité d'une modulation, et est égal au nombre de « motifs élémentaires » transmis par seconde.

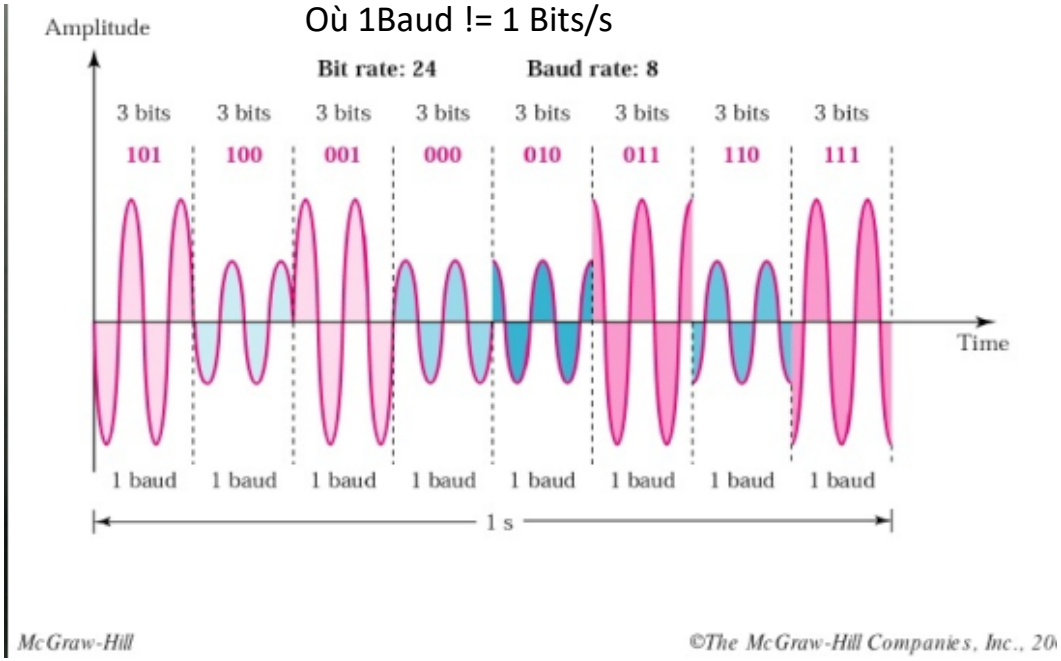
Selon les modulations, un motif élémentaire peut encoder de 1 à plusieurs bits.

Cas général: 1 Baud != 1 Bit/s

Or, en RS232, le motif élémentaire ne peut avoir que la valeur 0 ou 1, on n'encode donc qu'un seul bit dans un motif élémentaire

Ainsi, en RS232, on peut écrire qu'un 1 Baud correspond à un débit de 1 Bit/s (Ex: une transmission à 9600 Bd, correspond à un débit de 9600 Bits/s

Exemple d'une modulation complexe
Où 1Baud != 1 Bits/s

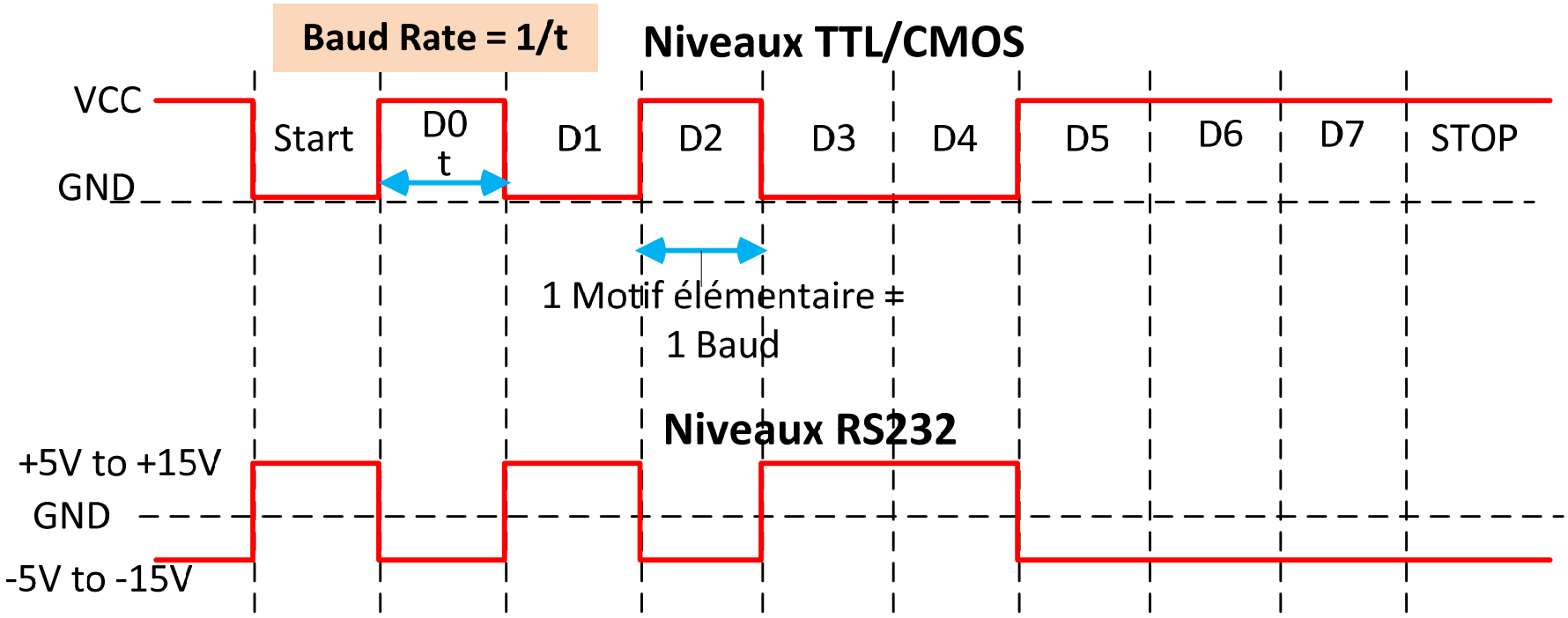


Les vitesses de transmission normalisées en RS232 sont:
300,1200,2400,4800,9600,19200, 28800, 38400, 57600, 115200 Bauds

Transmission RS232 - Chronogrammes

Niveaux CMOS/TTL ou RS232

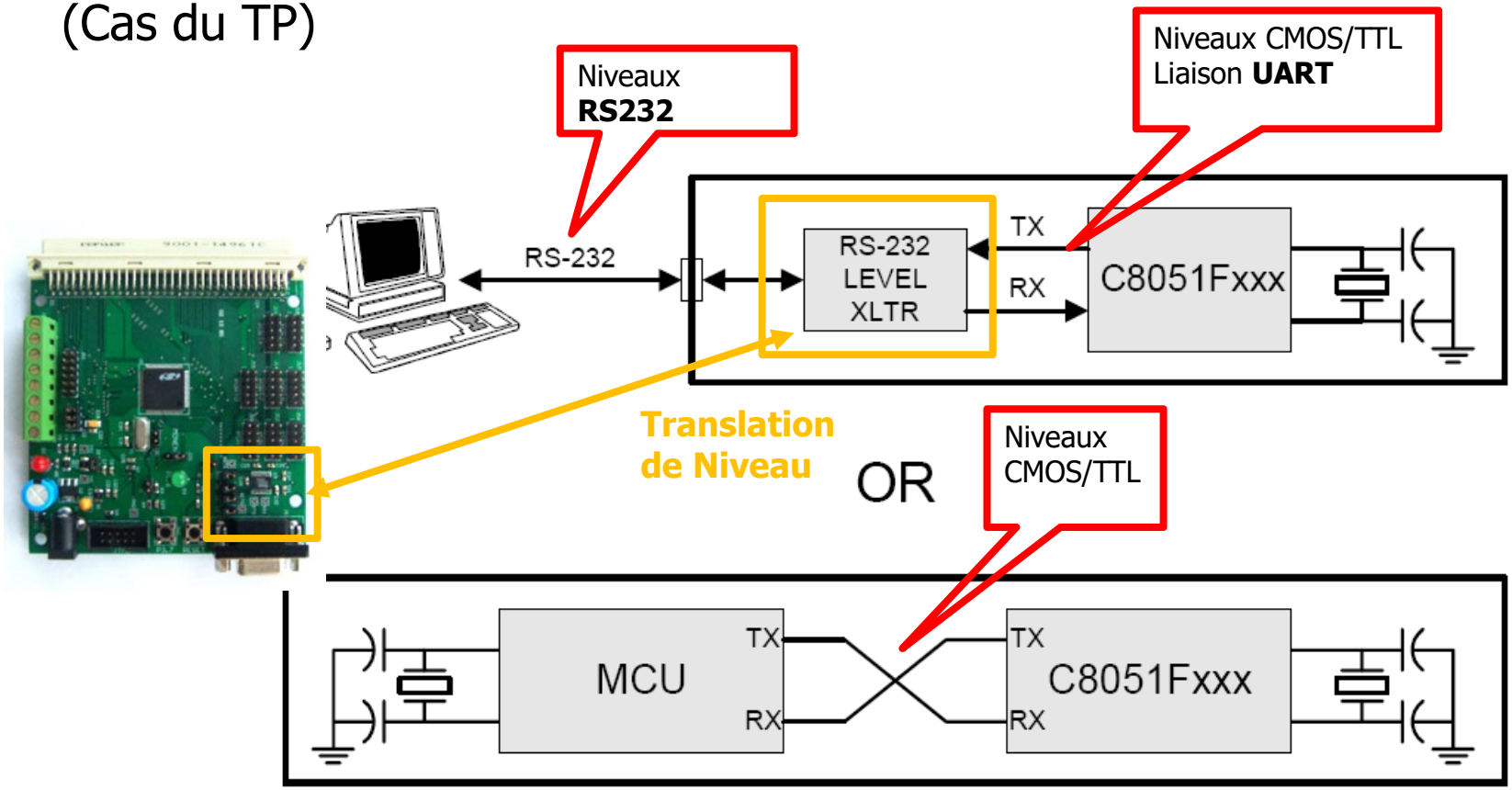
Exemple: Transmission de 0xE5 = 1110 0101



On pourra remarquer que l'octet de valeur 0XE5 ne correspond pas à un code ASCII (les codes ASCII vont de 0 à 0x7F). Cependant rien n'interdit d'échanger des codes supérieurs à 0x7F. Ils ne seront simplement pas affichés sur l'écran du terminal Putty

Applications UARTs en asynchrone

Connexion Microcontrôleur <-> Terminal
(Cas du TP)



Connexion Microcontrôleur <-> Microcontrôleur

La translation de niveau permet d'assurer une adaptation entre les niveaux CMOS/TTL (0-3V3 ou 0-5V)
et les niveaux RS232 (+/- 10V)

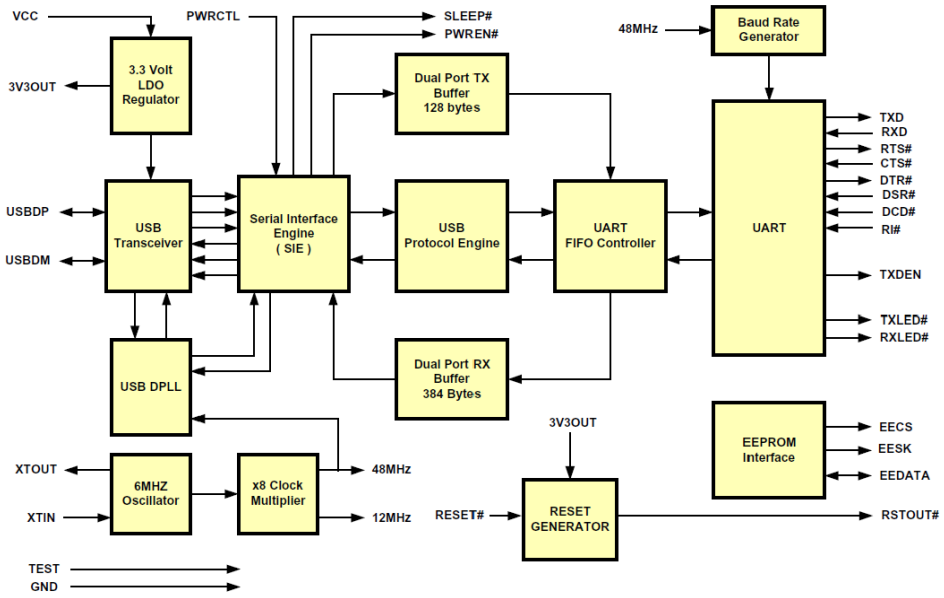
Liaison Microcontrôleur <-> Circuit spécifique

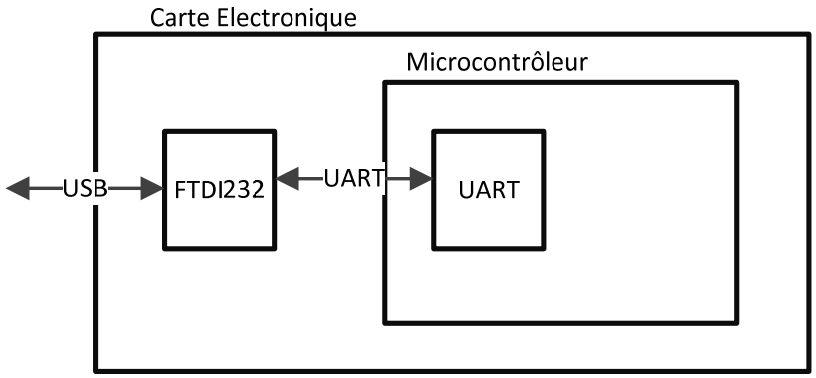
Exemple du FTDI232 – Interface USB – UART

Ce type de circuit permet d'implémenter **facilement** une interface USB sur une carte électronique. Il se charge de gérer l'USB.

Le microcontrôleur communique avec ce dernier via une « simple » liaison UART

2 FT232B Block Diagram





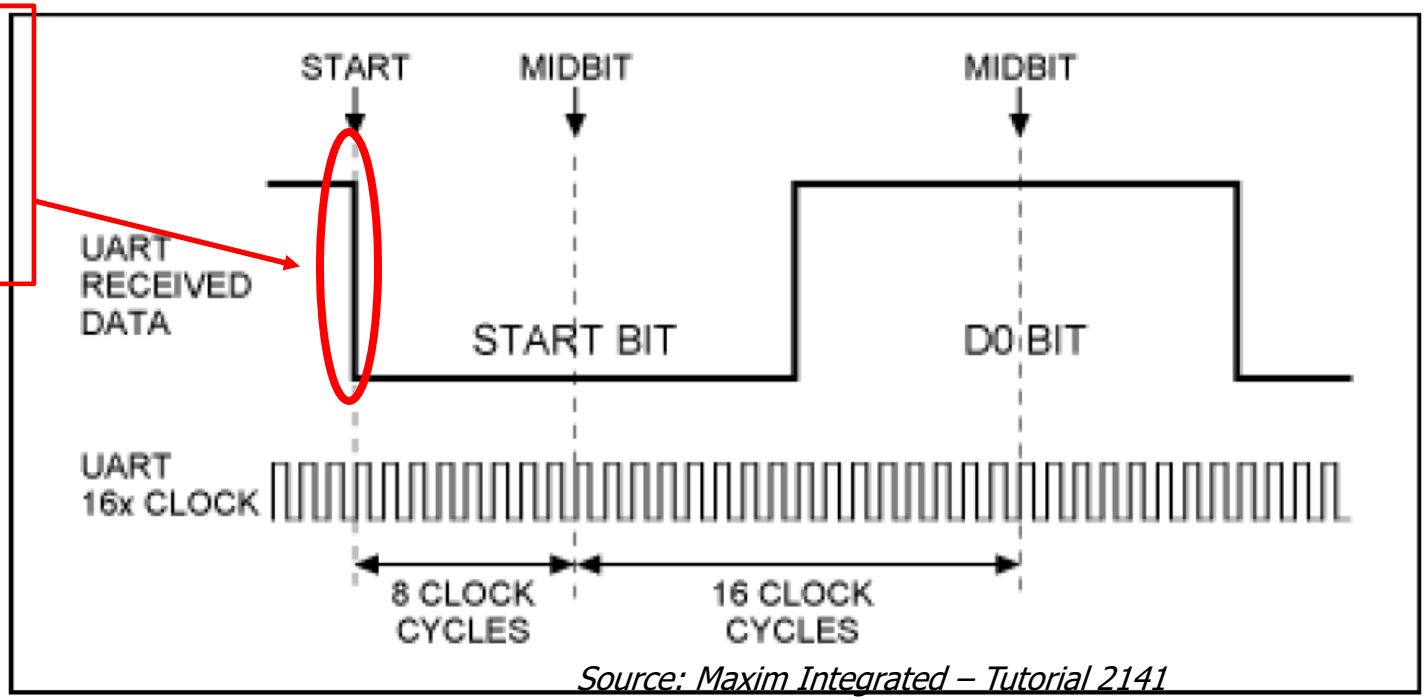
Une Horloge pour l'UART

Pour fonctionner l'UART a besoin d'un **signal d'horloge**

- **Pour cadencer la transmission**, un registre à décalage entrée parallèle, sortie série, cadencée avec une horloge de fréquence égale à la fréquence de transmission peut suffire.
- Pour **assurer la réception du signal**, le séquencement interne sera à une **fréquence bien supérieure** à la fréquence de réception pour une réception fiable.

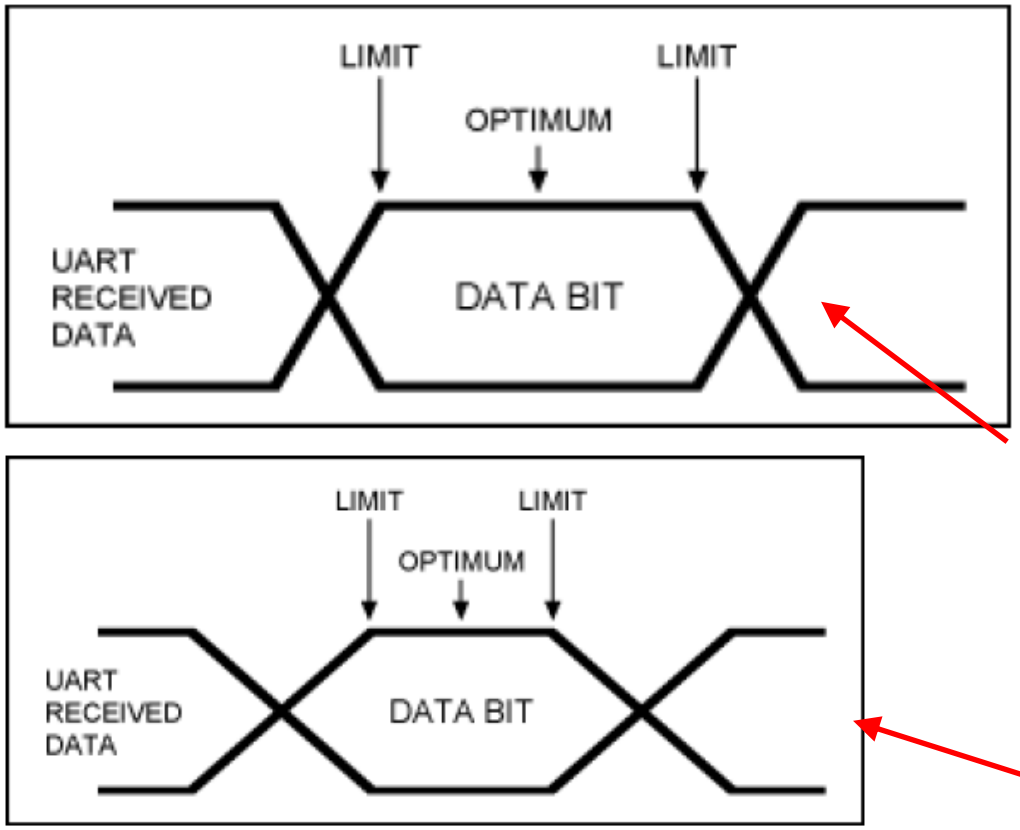
Mécanisme de synchronisation à la réception

Imprécision sur l'instant d'échantillonnage **à +/- 1** coup de CLK



Source: Maxim Integrated – Tutorial 2141

Echantillonnage des bits à la réception



Exemples:

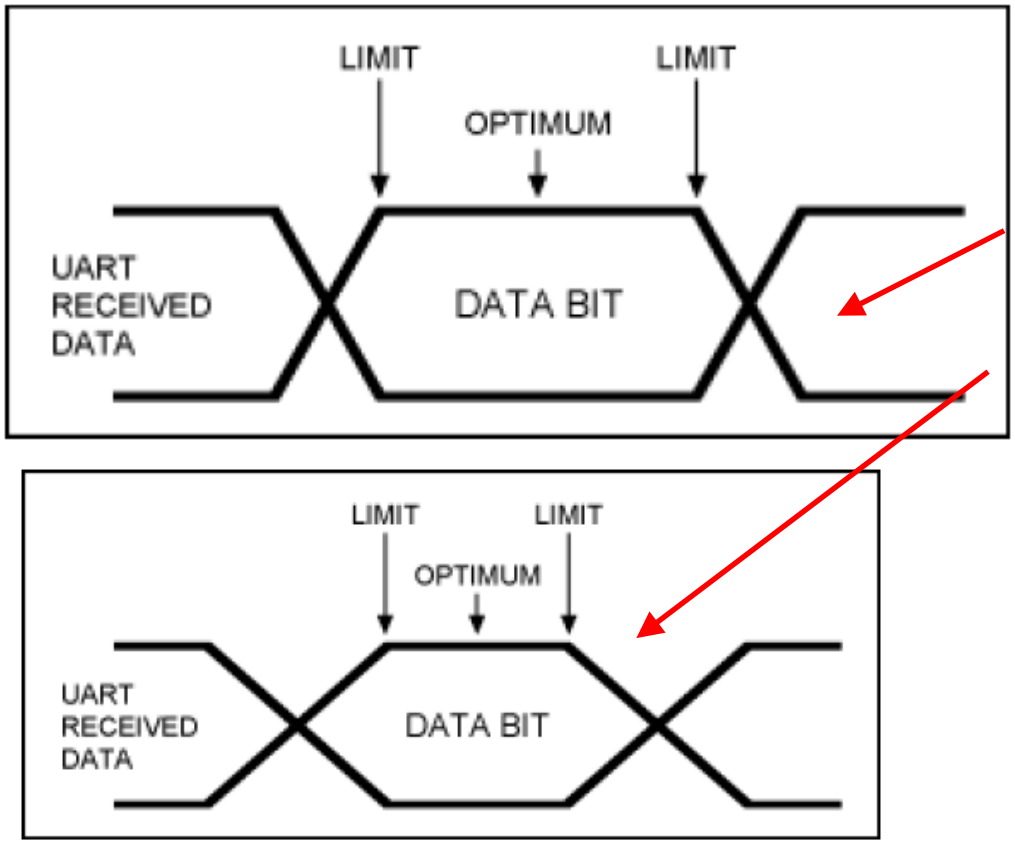
Pour un échantillonnage à une fréquence égale à 16 fois la fréquence du signal

- Transmission peu dégradée (liaison courte) – Information disponible sur 75% du cycle (+/-37,5%) - Information valide sur +/- 6 cycles d'horloge → +/-5 en tenant compte de la synchro du bit Start

- Transmission dégradée (liaison longue) – Information disponible sur 50% du cycle (+/-25%) – Information valide sur +/- 4 cycles d'horloge → +/- 3 avec la synchro bit de start.

Source: Maxim Integrated – Tutorial 2141

Précision requise de l'horloge



Source: Maxim Integrated – Tutorial 2141

Pour une transmission 8 bits, cela requiert 16X (1 + 8 + 0,5) = 152 cycles d'horloge.
(pour envoyer un octet, il y a 1 bit de start, 8 bits de données, et 0,5 bits de stop)

- Transmission peu dégradée: Sur le dernier bit la précision de l'horloge devra être meilleure que +/-5 / 152 = +/-3,3%
- Transmission dégradée: Sur le dernier bit la précision de l'horloge devra être meilleure que +/-3 / 152 = +/-2%

Attention à la précision de l'horloge système et du calcul des diviseurs d'horloge!!

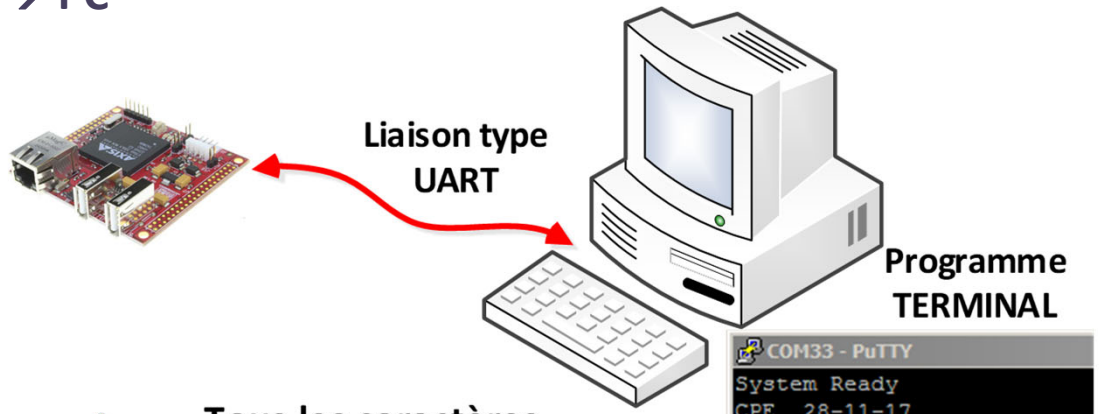
Cette imprécision se manifeste systématiquement sur le dernier bit, donc sur le bit de poids fort

Pour info: la précision de l'horloge interne du 8051F020 est de +/-20%!!

Utilisation de l'UART – Cas typique

Echange entre le microcontrôleur et un PC pour:

- **La transmission de commandes du PC vers le microcontrôleur PC → UC**
- **La remontée d'informations de mesures UC → PC**
- **Le debug UC → PC -**



Tous les caractères affichés respectent un encodage ASCII
On parle alors de Transmission en mode « Texte »

Table ASCII

ASCII TABLE

Remarque: encodage sur 7 bits

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Caractères non imprimables

Source: commons.wikimedia.org

Utilisation du codage ASCII

Cas d'un échange d'informations entre un UC et un terminal (couple écran – clavier)

Le terminal sur le PC impose l'encodage en ASCII pour afficher les caractères sur l'écran.

Côté UC, on se devra de respecter cette convention.

Ainsi:

- Pour afficher le caractère « K » sur l'écran du terminal, il faudra transmettre l'octet 0x4B sur la liaison série (Code ASCII de K: 0x4B)
- Si je presse la touche « Entrée » (Carriage Return) le terminal va envoyer l'octet 0x0D sur la liaison série. (Code ASCII de la commande « Carriage Return » : 0x0D)

Exemple: Transmission de la chaine de caractères: « 1 2 3! » avec retour à la ligne

On transmettra sur la ligne les octets: 0x31 0x20 0x32 0x20 0x33 0x21 0x0A 0x0D

'1' Space '2' Space '3' '!' LineFeed CarriageReturn

Cas d'un échange UC <- -> UC

Le codage ASCII (mode transmission « Texte »n'est pas indispensable. Il est possible d'échanger en mode « binaire ». L'essentiel est que les 2 systèmes partagent les mêmes conventions.

Exemple: transmission de la valeur 24 (par ex, valeur entière d'une température)

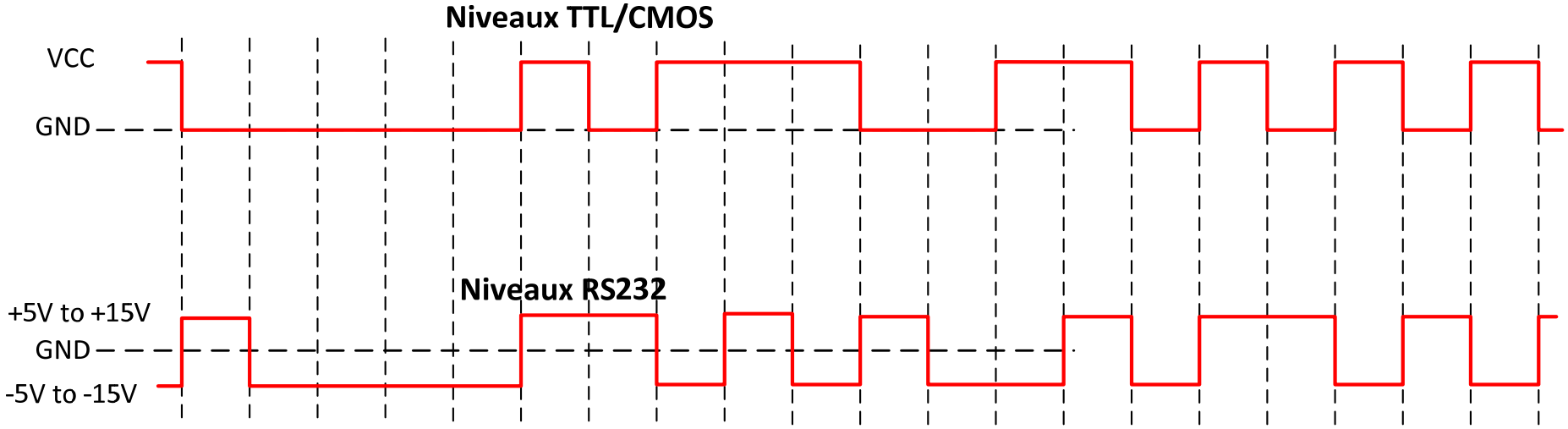
En mode binaire: transmission de 0x18 (24dec = 18hexa)

En mode texte: transmission de 0x32 0x34 («24») c'est-à-dire 0x32, code ASCII du chiffre 2, suivi de 0x34, code ASCII du chiffre 4

Pause Chronogramme RS232

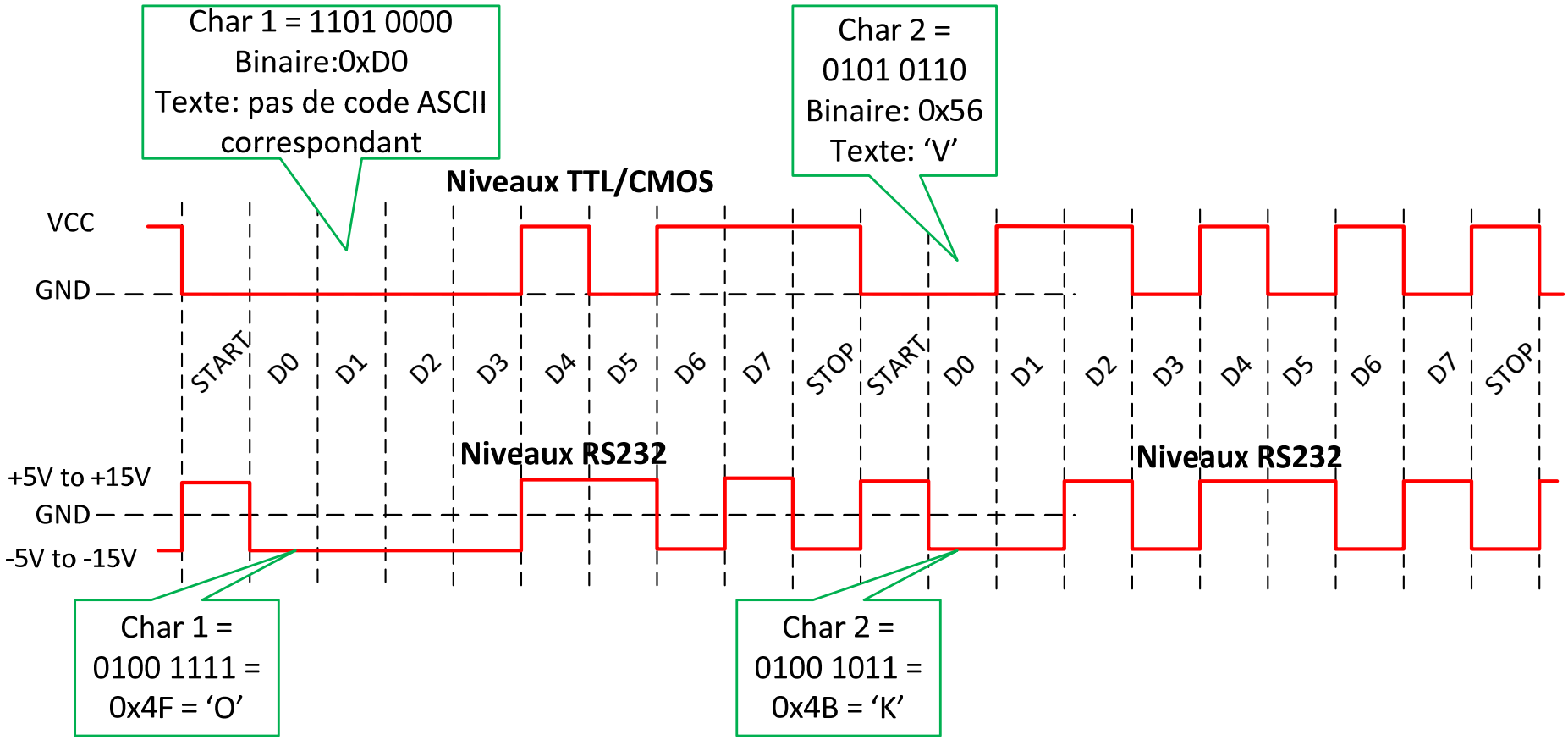
Niveaux CMOS/TTL sur les broches du 8051F020
ou RS232 sur le connecteur DB9

Décoder les signaux Série TTL/CMOS et série RS232



CORRECTION

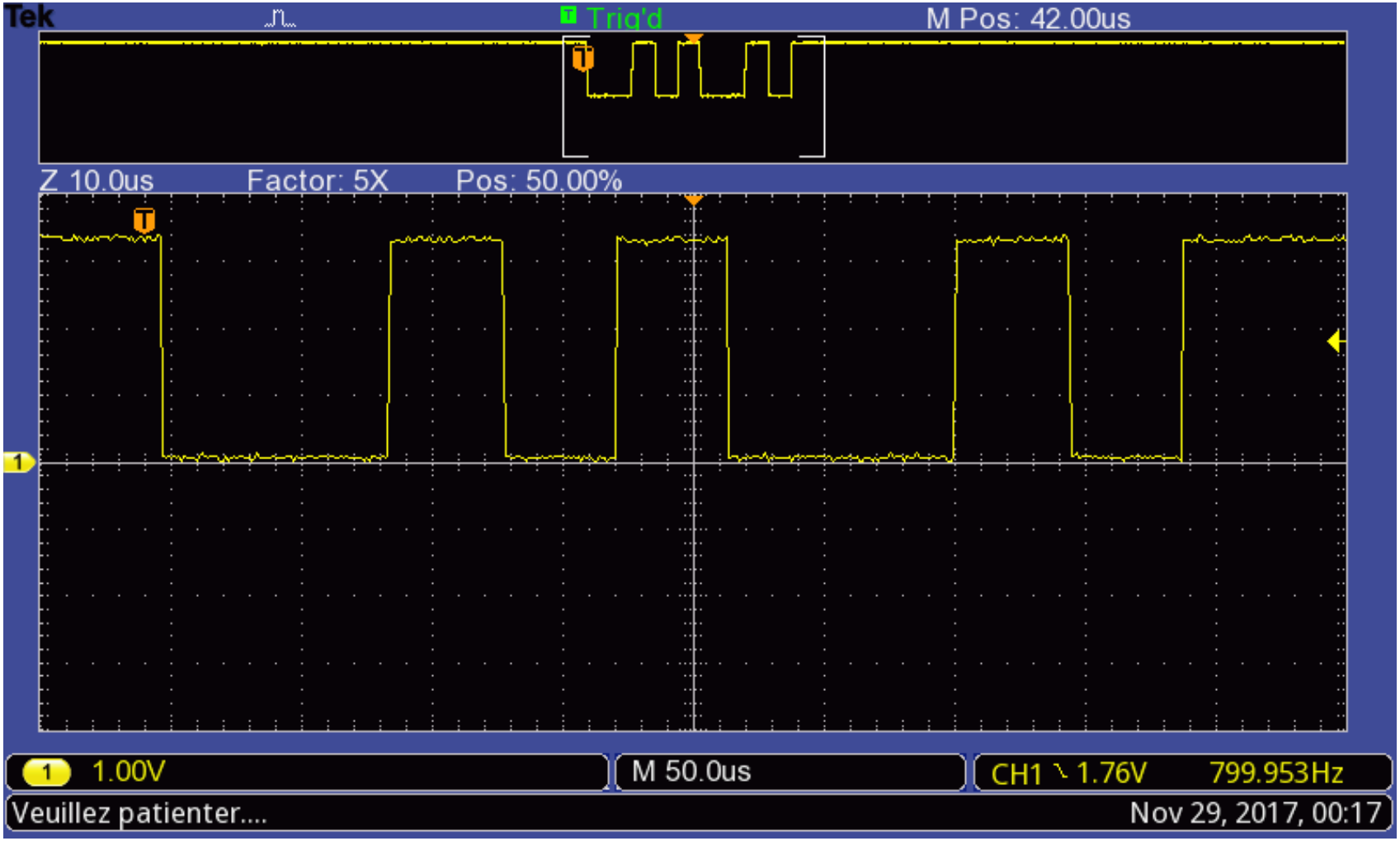
Pause Chronogramme RS232



Pause Oscillogrammes RS232

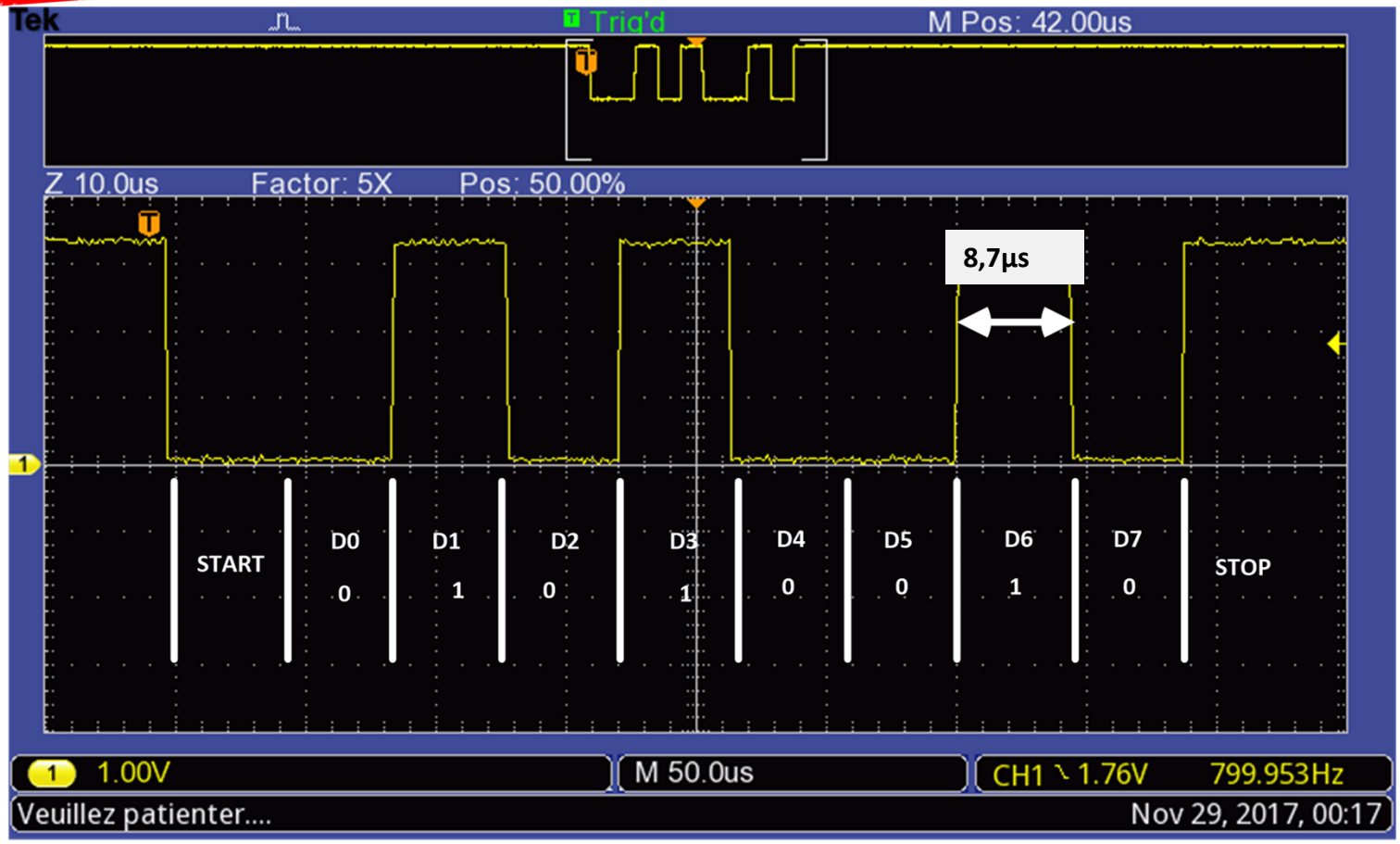
Ce chronogramme devrait vous permettre de répondre aux questions suivantes:

- Les niveaux échangés sont-ils des niveaux UART ou RS232?
- Quelle est la vitesse de transmission?
- Quelle est la valeur de l'octet transmis?



CORRECTION

Corrigé - Pause Oscillogrammes RS232



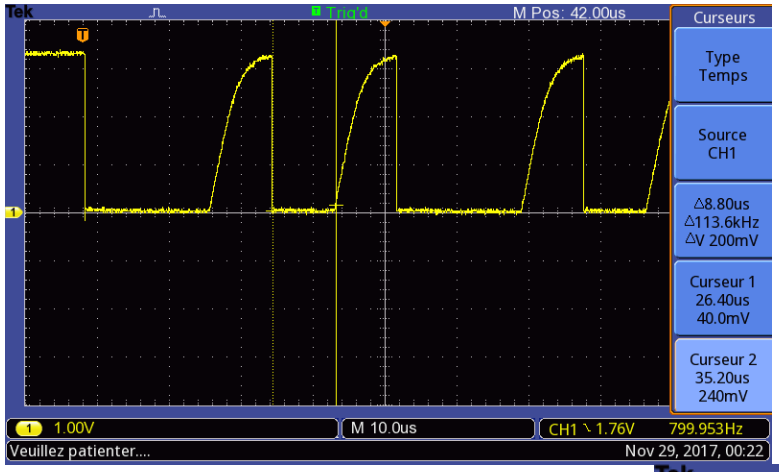
Niveaux UART (0 – 3,3V) - Valeur transmise: 01001010B = 0x4A = 'J'

Vitesse de transmission approximative : $1/8{,}7X10^{-6}$ = 115200 Bit/s (115200 baud)

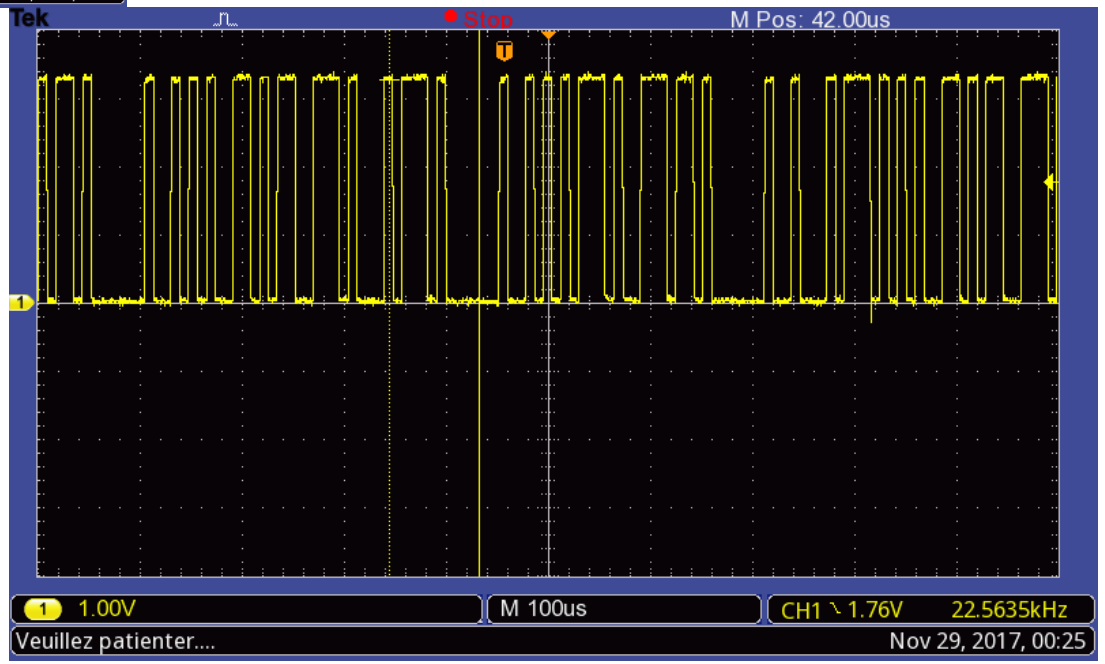
Temps de transmission d'un octet: 10 X 8,7µS = 87µs

CORRECTION

Oscillogrammes RS232



Mais ce n'est pas toujours aussi simple….



Cas typique Transmission d'un entier sur une liaison série – Interface M to M (Machine to Machine)

Attention, ce raisonnement est valable si on **maitrise les 2 côtés** (Emetteur et Récepteur) de la liaison. Bien souvent, on ne maitrise pas une extrémité de la liaison; on doit alors s'adapter…

Par ailleurs cette approche est destinée à assurer la sécurisation de la transmission avec le minimum de code possible.

Soit un entier défini et initialisé ainsi:

```
Int int_value = 36656; // = 0x8F30
```

Problème: On souhaite le transmettre sur la liaison série

Remarque: la liaison série travaille toujours en transmettant les octets un par un (registre SBUF dans les 8051)

Solution bas niveau : Transmission « directe » d'un entier sur une liaison série

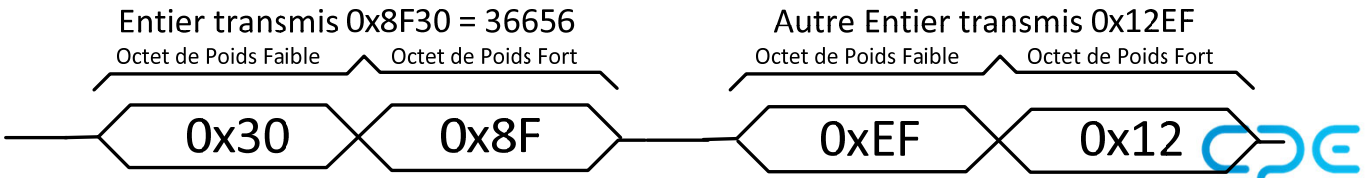
Pour comprendre, raisonnons en hexadécimal: 36656 (base10) = 8F30 (base16)

- Décomposer l'entier en 2 octets
  - Solution 1: le microcontrôleur le fait par calcul

    Exemple: `octet_H = int_value/256;`

    `octet_L = int_value%256;`
  - Solution 2: à la compilation, en utilisant des unions
- Transmettre les octets
  - Ordre de transmission? Se fixer une convention entre émetteur et récepteur
  - Dans notre exemple, si on choisit un ordre de transmission octetL – octetH

  On va alors envoyer: octet1: 0x30 suivi de 0x8F
- Inconvénients de cette transmission **en « binaire »? (par opposition au mode « texte »)**
  - Impossible de distinguer un octet de poids faible et un octet de poids fort – Synchronisation impossible en cas de séquence de Déconnexion/Reconnexion
  - Chaque octet constituant l'entier peut prendre toutes les valeurs possibles de 0 à FF(base16), il est donc impossible de rajouter des octets supplémentaires de valeur différentes pour synchroniser les échanges



Solution bas niveau - Codage ASCII

- Une Etape supplémentaire: Transmettre des caractères de contrôle et de synchronisation
- Ceci implique de coder chaque octet de données transmis en code ASCII (pour avoir à disposition des valeurs pour des caractères de contrôle et de synchro)
- Intérêt: pour transmettre une valeur numérique (un entier) on utilise seulement 16 (voire 22) valeurs d'octet différentes: les codes ASCII des chiffres '0' à '9' et les codes ASCII des chiffres hexadécimaux 'a' à 'f' (ou 'A' à 'F').
- Toutes les autres valeurs sont disponibles pour transmettre des signaux de contrôle et de synchronisation.
- Inconvénient: le nombre d'octets à transférer est augmenté.
- Mais beaucoup plus de souplesse pour l'échange d'informations (de surcroit, vérification immédiate sur une console).
- Cette solution est robuste, car associée à une gestion de timeout, elle passe sans problème l'épreuve de câble déconnecté/reconnecté en pleine transmission.
- Coté réception, le traitement est très simple, essentiellement car la taille de la trame est constante.

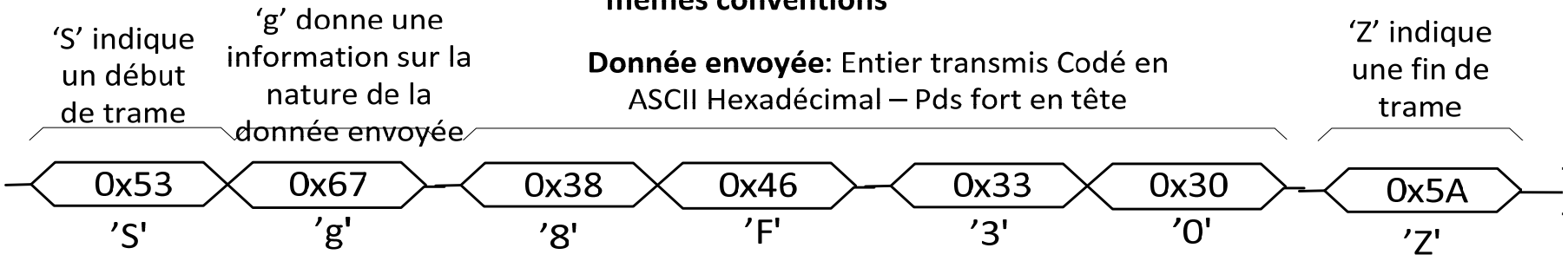
Exemple de transmission en Codage ASCII

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

Transmission orientée « M to M » (machine to machine)
Sécurisation de la transmission
Resynchronisation possible
Code de traitement simple

Exemple d'une trame de données codée en ASCII

Les choix de codage sont purement arbitraires. **Il faut simplement qu'émetteur et récepteur partagent les mêmes conventions**



Utilisation de fonctions de Librairies

Utilisation d'une fonction de librairie "printf" – Transmission en mode texte – Codage ASCII – envoi des codes '3' '6' '6' '5' '6' et possiblement des caractères LINE FEED et CARRIAGE RETURN (ils permettent de mettre en oeuvre la synchronisation)

Avantages:

- Simplicité du codage
- Le printf permet de prendre en charge de nombreux formats

Mais:

- Code volumineux (1Ko environ!).
- Temps d'exécution important
- Côté récepteur, l'analyse de la chaine transmise requiert un peu de traitement

LIVE AND
DISCOVER

Contact

François JOLY
Tél. : 04 72 43 13 36
francois.joly@cpe.fr

www.cpe.fr