Spécialité SI

LYCÉE de l'ÉLORN
le choix de votre réussite

Les liaisons de données numériques

1- LA TRANSMISSION DE DONNEES - CODAGE DES INFORMATIONS

Les réseaux de communication permettent la transmission d'informations codées sous la forme d'une suite d'éléments binaires entre des équipements terminaux. Plusieurs encodages existent pour transmettre des caractères.

L'encodage ASCII

Le code normalisé le plus utilisé est le code ASCII (American Standard Code for Information Interchange). À chaque caractère (lettres, chiffres, symboles, codes de contrôle) correspond un nombre de 7 bits, ce qui permet de coder 128 (2^7) caractères différents représentés et codés en décimal dans le tableau suivant.

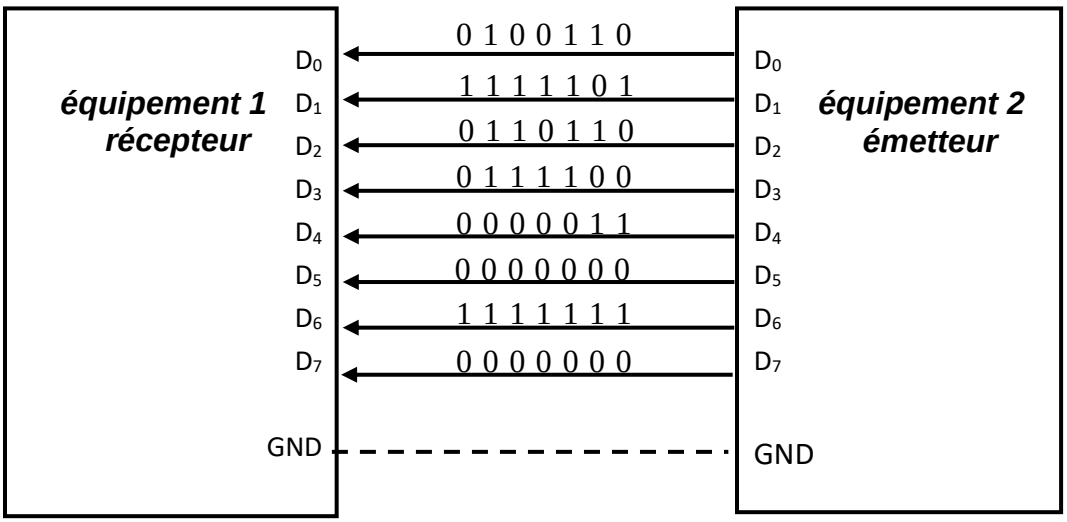
000	(nul)	016 ▶	(dle)	032	sp	048	0	064	@	080	P	096	`	112	p
001 ☺	(soh)	017 ◀	(dc1)	033	!	049	1	065	A	081	Q	097	a	113	q
002 ☻	(stx)	018 ↕	(dc2)	034	"	050	2	066	B	082	R	098	b	114	r
003 ♥	(etx)	019 ‼	(dc3)	035	#	051	3	067	C	083	S	099	c	115	s
004 ♦	(eot)	020 ¶	(dc4)	036	$	052	4	068	D	084	T	100	d	116	t
005 ♣	(enq)	021 §	(nak)	037	%	053	5	069	E	085	U	101	e	117	u
006 ♠	(ack)	022 ▬	(syn)	038	&	054	6	070	F	086	V	102	f	118	v
007 •	(bel)	023 ↨	(etb)	039	'	055	7	071	G	087	W	103	g	119	w
008 ◘	(bs)	024 ↑	(can)	040	(	056	8	072	H	088	X	104	h	120	x
009	(tab)	025 ↓	(em)	041	)	057	9	073	I	089	Y	105	i	121	y
010	(lf)	026	(eof)	042	*	058	:	074	J	090	Z	106	j	122	z
011 ♂	(vt)	027 ←	(esc)	043	+	059	;	075	K	091	[	107	k	123	{
012 ♀	(np)	028 ∟	(fs)	044	,	060	<	076	L	092	\	108	l	124	|
013	(cr)	029 ↔	(gs)	045	-	061	=	077	M	093	]	109	m	125	}
014 ♫	(so)	030 ▲	(rs)	046	.	062	>	078	N	094	^	110	n	126	~
015 ☼	(si)	031 ▼	(us)	047	/	063	?	079	O	095	_	111	o	127	⌂

L'encodage UTF-8

L'encodage UTF-8 utilise les mêmes codes qu'ASCII pour ses premiers caractères. Il se sert d'octets additionnels pour représenter des caractères spéciaux comme 'É', 'ç' ou '€' et tous les autres symboles, y compris les émoticônes, utilisés dans le monde.

2 - TRANSMISSION PARALLELE

Les bits sont envoyés simultanément sur des fils distincts pour arriver ensemble à destination. La ligne de transmission comporte n fils d'information plus un fil commun (GND) et plusieurs fils de service suivant le protocole. Ce type de câblage est devenu obsolète pour les imprimantes ou les disques durs.

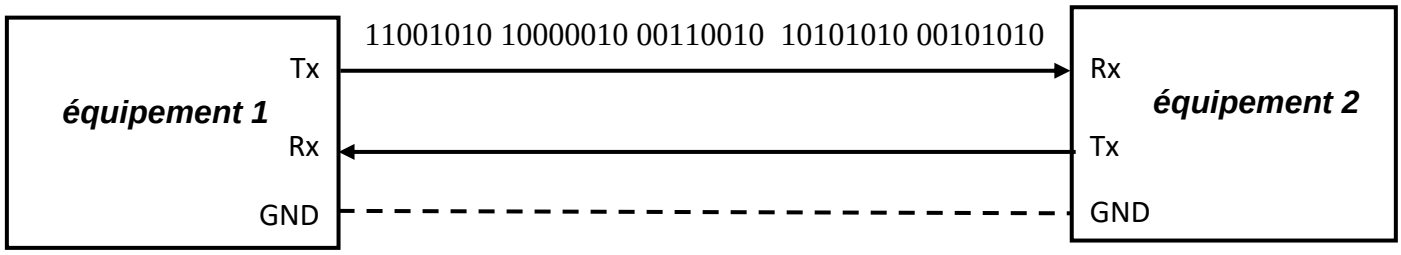


→ Décoder le mot envoyé de 2 vers 1

3 - TRANSMISSION SÉRIE

3-1 – Présentation de la transmission série asynchrone

Les bits sont envoyés les uns derrière les autres sur le même fil. La ligne comporte au moins un fil d'information plus un fil commun. Il est convenu de commencer par envoyer le bit de poids faible (mode little endian) de chaque octet.



→ Décoder le mot envoyé de 1 vers 1

Ce mode de transmission est dit **asynchrone** car l'intervalle de temps entre 2 bits transmis est paramétré sur les 2 équipements. Il existe un mode d'échange **synchrone** qui utilise un signal supplémentaire d'acquittement des données. La vitesse de transmission s'exprime en bauds, bits/s ou bps.

3.2- Protocole de transmission série, la norme RS 232

Il est nécessaire d'établir un protocole de transmission afin que les éléments communicants puissent se comprendre. Bien entendu le protocole devra être commun aux deux éléments communicants.

Paramètres rentrant en jeu :

- **Longueur des mots :** 6, 7 ou 8 bits
- **La vitesse de transmission :** les différentes vitesses de transmission sont réglables à partir de 110 bits par seconde (bps) de la façon suivante : 110 bps, 150 bps, 300 bps, 600 bps, 1200 bps, 2400 bps, 4800 bps, 9600 bps …18,2 kbps …56 kbps … x 2 …
- **Parité** : le mot transmis peut-être complété ou non d'un bit de parité qui sert à détecter les erreurs éventuelles de transmission. Il existe deux types de parité.
    - ***parité paire*** : le bit ajouté à la donnée est positionné de telle façon que le nombre des états 1 soit paire sur l'ensemble données + bit de parité.

      ex : soit la donnée 11001011 contenant 5 bit à 1, le bit de parité paire est positionné à 1, ramenant ainsi le nombre de 1 à 6 (nb paire).

    - ***parité impaire*** : le bit ajouté à la donnée est positionné de telle façon que le nombre des états 1 soit impaire sur l'ensemble données + bit de parité

      ex : soit la donnée 11001011 contenant 5 bits à 1, le bit de parité paire est positionné à 0, laissant ainsi un nombre de 1 impaire.
- **Bit de start :** la ligne au repos est à **l'état logique 1**. Pour indiquer qu'un mot va être transmis la ligne passe à l'état bas avant de commencer le transfert.
- **Bit de stop :** après la transmission, la ligne est positionnée au repos pendant 1, 2 ou 1,5 périodes selon le nombre de bits de stop.

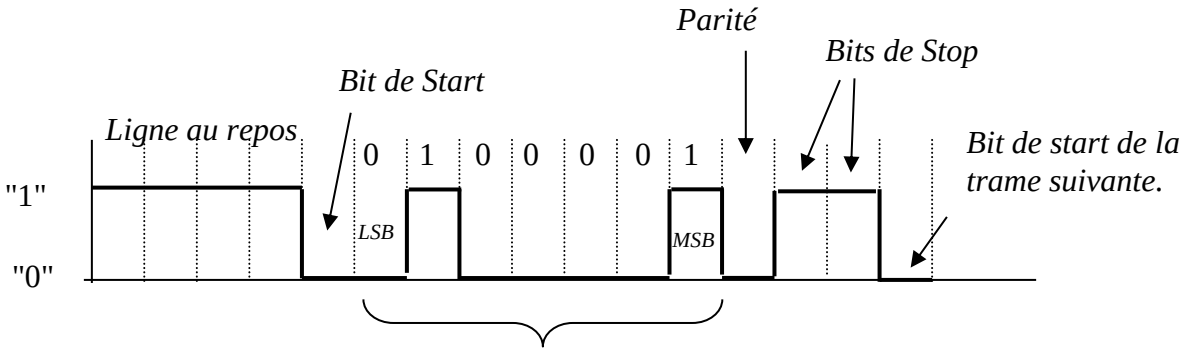
Format (chronogramme, oscillogramme) des trames

Le bit de Start apparaît en premier dans la trame puis les données (poids faible en premier), la parité éventuelle et le (les) bit(s) de stop.

Exemple :

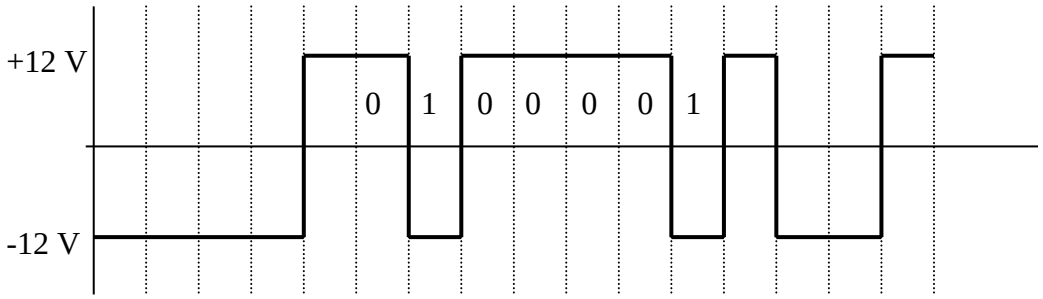
Soit à transmettre en parité paire, 7 bits de données, 2 bits de stop, le caractère "B" dont le codage ASCII est $(42)_{16}$ ou $(1000010)_2$; La trame série sera la suivante :

a) D'un point de vue logique (TTL)

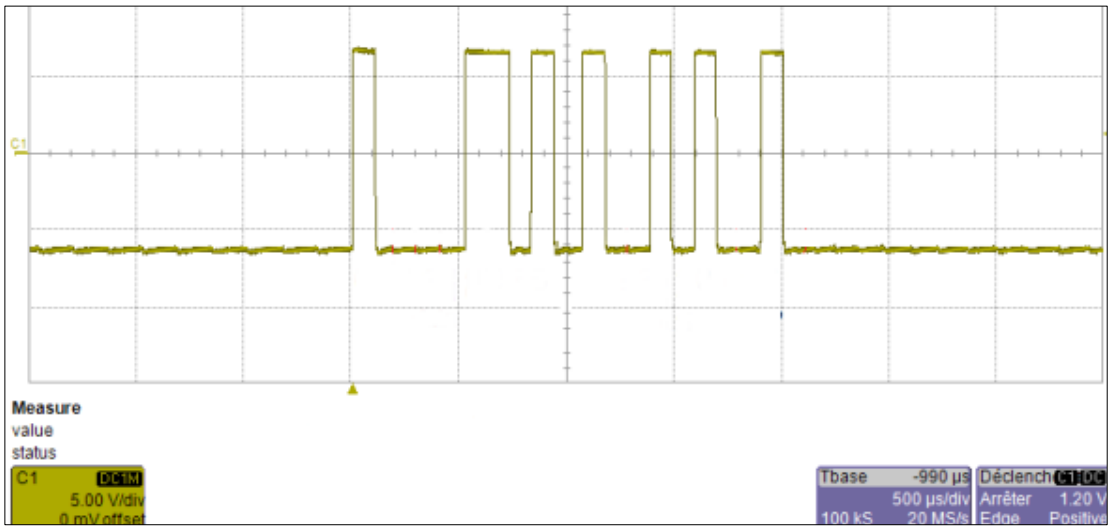


Donnée sur 7 bits (bit de poids faible en premier)

b) D'un point de vue RS 232



c) Déterminer le débit (en bit/s) ainsi que le message envoyé par la ligne RS232 ci-dessous :



Le protocole est : 1 start, 7 bits de données, parité impaire, 1 stop

3 – Manipulations

Sur une carte Micro:bit, toutes les actions de print() programmées en Python ont pour effet de générer une chaîne de caractère sur la liaison série (via l'USB) avec les paramètres par défaut uivants : 115200 bps, 8 bits de données, 1 bit de stop, pas de parité.

3.1 Observation d'une trame de communication à l'oscilloscope.

Faites fonctionner ce programme puis relever et justifier les trames vues sur la broche 0 avec le paramètre uart.EVEN puis avec uart.ODD.

```
from microbit import *
sleep(2000)

uart.init(baudrate = 115200, bits = 8, parity = uart.EVEN, stop = 1, tx = pin0, rx = pin1)

while True:
    uart.write("e")
    sleep(10)
```

3.2 Communication PC ↔ Micro:bit

Le script suivant est capable d'envoyer ou de recevoir des données via une liaison série.

```
from microbit import *
sleep(2000)

uart.init(baudrate=14400, bits=8, parity=None, stop=2)

while True:
    if button_a.is_pressed():
        uart.write("je suis la carte microbit \n")

    if uart.any():
        msg=uart.read()
        display.scroll(msg)
    sleep(200)
```

a) **Tester** son fonctionnement en mettant en œuvre un terminal série Termite (ou Putty) dont les paramètres seront à définir pour assurer la compatibilité.

b) Un capteur de température est intégré dans un des composants de la carte. Une fonction python permet de retourner la valeur de la température de la carte, vous la repérerez sur cette page : *https://microbit-challenges.readthedocs.io/en/latest/tutorials/thermometer.html#*

Faites-en sorte que la température de la carte soit envoyée 10 fois par seconde sur le terminal série du PC à un débit de 9600 bps. La chaine de caractère envoyée sera par exemple de la forme « 21 °C » .

Au besoin tentez d'étalonner ce thermomètre.

c) Mise en œuvre d'une interface graphique sous Tkinter.

Tkinter (Tk interface) est un module intégré à la bibliothèque standard de Python qui permet de créer des interfaces graphiques. Vous avez à votre disposition le script python « *aff_tkinter.py* » qui sera capable d'afficher les caractères reçus depuis un port série de l'ordinateur.

Ouvrez *aff_tkinter.py* sur Edupython ou sur Thonny pour réaliser un affichage graphique de la donnée transmise par la carte Microbit. Ajouter du texte ou des graphiques sur cette interface afin de la personnaliser.