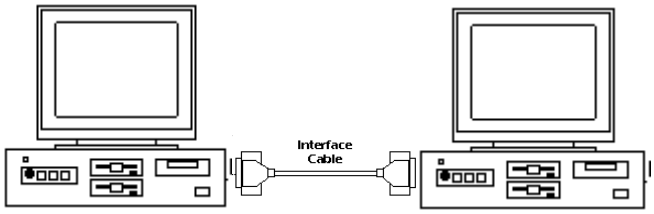
1 – Présentation : Transmission série via port RS232

On a intercepté un message transmis entre 2 PC via le port série.



2 – Données

Le paramétrage de ce port série est le suivant :

- 1 bit de start ;
- Transmission des données sur 8 bits (codage ASCII) ;
- 1 bit de parité paire ;
- 1 bit de stop.

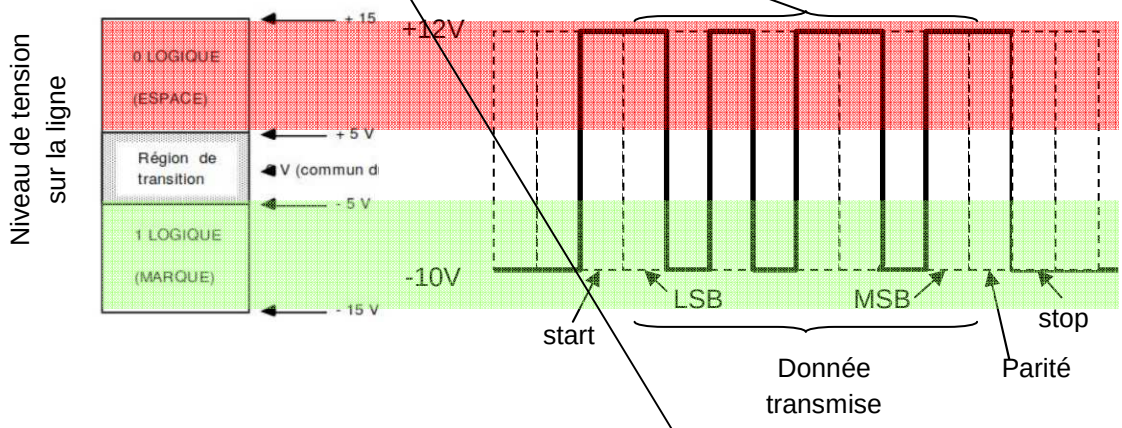
(voir les pages 3 & 4 du sujet pour comprendre ce charabia).

Le code hexadécimal du message transmis entre les 2 PC est le suivant :

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00000000	?	?	61	69	6D	65	20	6C	27	61	75	74	6F	6D	61	74
00000010	69	73	6D	65												

Message transmis (en hexadécimal)

Le premier octet transmis (octet 0) a donné le signal électrique sur la ligne RS232 suivant :



Le second octet transmis (octet1) a donné la valeur décimale $39_{(10)}$

3 – Problématique

Objectif global : décoder le message transmis

4 – Travail demandé

Objectif N°1 : Décoder l'octet 0 transmis

1- Transformer le signal électrique obtenu pour le premier octet en un code binaire (donner des explications)

2- Convertir ce code binaire en hexadécimal (expliquer la méthode)

3- Trouver la lettre qui se cache derrière ce premier octet (expliquer la méthode)

Objectif N°2 : décoder l'octet 1 transmis

4- Convertir en hexadécimal le 2ème octet (expliquer la méthode)

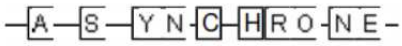
5- Décoder la lettre qui se cache derrière ce 2ème octet

Objectif N°3 : décoder le reste des octets transmis

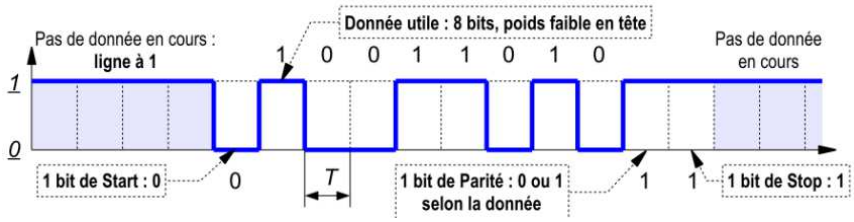
6- Décoder chaque lettre qui se cache derrière les octets 2 à 20 et donner le message transmis

Fonctionnement d'une liaison série asynchrone :

Dans ce type de transmission, la source de données produit des caractères à des instants aléatoires. Chaque caractère est transmis au moment où il est produit sans tenir compte des caractères précédents ou suivants.

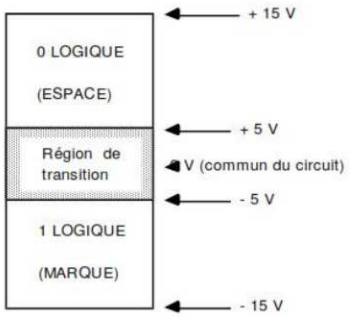


La série de bits qui représentent l'envoi d'un caractère, une dizaine ou une douzaine de bits, doit respecter des temps précis et rigoureusement calibrés. Chaque bit se voit imparti un temps donné, sur lequel le récepteur est synchronisé, faute de quoi rien d'exact ne pourra être détaché de la réception. C'est le front descendant du bit start qui déclenche, à la réception, l'horloge de lecture.



- Le bit de start indique qu'un caractère va être transmis ;
- Il est suivi du mot de code du caractère à transmettre (LSB➔Poids faible, MSB➔Poids fort) ;
- Le bit de parité (facultatif) permet un contrôle de la transmission (exemple de paramétrage pair : ce bit vaut 1 si le nombre de 1 contenu dans le mot transmis est pair) ;
- Un ou deux bits de stop terminent la transmission ;
- Le temps de transmission de chacun des bits (start, bits utiles, parité, stop) est arbitrairement fixé à une valeur constante connue de l'émetteur et du récepteur.

Les niveaux logiques d'une liaison RS232.



Un niveau de tension haut correspond à un niveau logique 0 et un niveau de tension bas représente un niveau logique 1.

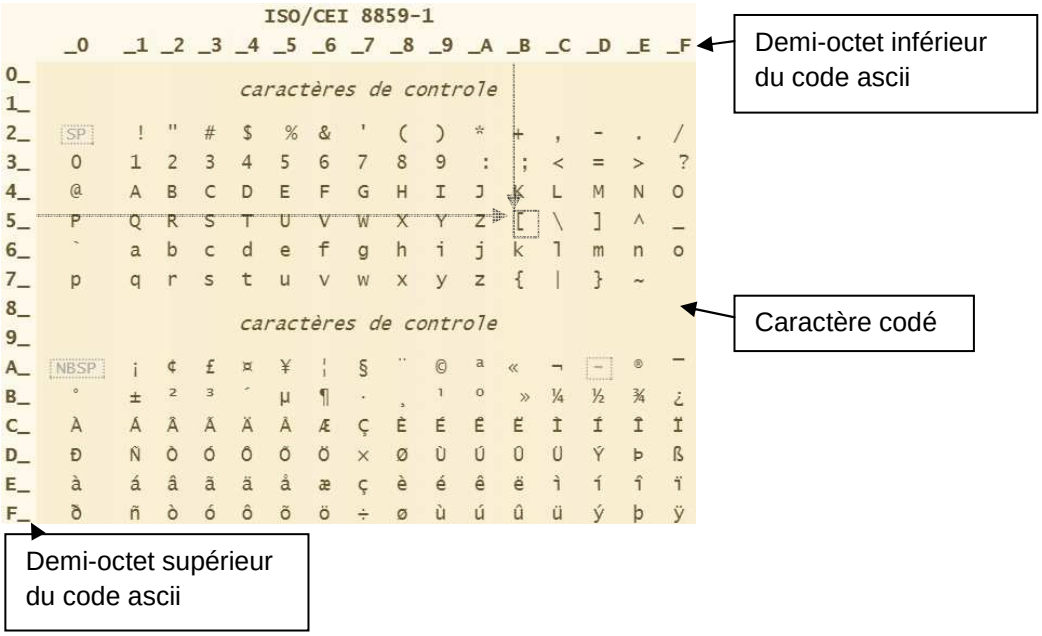
Codage ASCII étendu ISO/CEI 8859-1

Un message revient à rédiger un ensemble de caractères standardisés. Le message est donc formé par une chaine de caractères. Chaque caractère est codé en binaire selon un code prédéfini.

La norme **ASCII (American Standard Code for Information Interchange**) permet à toutes sortes de machines de stocker, analyser et communiquer de l'information textuelle. En particulier, la quasi-totalité des ordinateurs personnels et des stations de travail utilisent l'encodage **ASCII**.

Le codage **ASCII** est souvent complété par des correspondances supplémentaires afin de permettre l'encodage informatique d'autres caractères, comme les caractères accentués par exemple. Cette norme s'appelle **ISO-8859** et se décline par exemple en **ISO-8859-1** lorsqu'elle étend l'**ASCII** avec les caractères accentués d'Europe occidentale.

Le tableau ci-dessous donna la correspondance entre le caractère transmis et le code hexa décimal (formé sur deux demi-octets) qui lui correspond.



Exemple : le code hexadécimal transmis est 5B et le caractère qui se cache derrière est [