

# Les clés USB

## Table des matières

<b>1. HISTORIQUE</b> .....	<b>2</b>
<b>2. DÉFINITION</b> .....	<b>2</b>
<b>3. UTILISATION</b> .....	<b>3</b>
<b>4. LA MEMOIRE FLASH</b> .....	<b>4</b>
A/ L' EEPROM .....	4
B/ LA MEMOIRE NOR .....	4
C/ LA MEMOIRE NAND .....	5
<b>5. INTRODUCTION TECHNIQUE ET NORME DU BUS</b> .....	<b>6</b>
<b>6. SPECIFICATIONS PHYSIQUE</b> .....	<b>8</b>
A/LA CLE .....	8
B/LES CONNECTEURS .....	9
C/LE BUS .....	10
• La topologie de bus .....	10
• L'hôte USB .....	10
• Caractéristiques électriques .....	10
<b>7. LE MODELE DE FLUX DE DONNEES DE L'USB</b> .....	<b>13</b>
A/LES DIFFERENTES COUCHES .....	13
B/LA TOPOLOGIE DU BUS .....	14
C/LE POINT FINAL .....	15
<b>8. LES TYPES DE TRANSFERTS</b> .....	<b>17</b>
A/INTRODUCTION .....	17
B/LES DIFFERENTS TYPES DE TRANSFERT .....	18
• Transfert de contrôle .....	18
• Transfert en rafale (en groupe) .....	19
• Les transferts d'interruption .....	20
• Les transferts isochrones .....	21
• Les transactions SPLITS. (uniquement pour la norme USB 2) .....	22
<b>9. LA COUCHE PROTOCOLE</b> .....	<b>23</b>
A/CHAMP DE SYNCHRONISATION .....	23
B/CHAMP D'IDENTIFICATION DE PAQUET .....	23
C/LES CHAMPS D'ADRESSE .....	24
D/LE CHAMP DE NUMERO DE TRAME .....	24
F/LES CRC (CODES CYCLIQUES REDONDANTS) .....	25
G/LES PAQUETS "TOKEN" .....	25
H/LES PAQUETS DE DEBUT DE TRAME .....	25
I/LES PAQUETS DE DONNEES .....	26
J/LES PAQUETS DE POIGNEES DE MAINS .....	26
<b>10. EN CE MOMENT...</b> .....	<b>27</b>
<b>11. DÉRIVÉS</b> .....	<b>28</b>
<b>12. CRITÈRES DE CHOIX</b> .....	<b>29</b>
<b>13. CONCLUSION</b> .....	<b>29</b>
<b>14. BIBLIOGRAPHIE</b> .....	<b>30</b>

## 1. Historique

Inventée en 1998 par IBM, la clé USB fut créée pour remplacer la disquette qui ne répondait plus au demande au niveau de la capacité et du CD-ROM qui n'est pas assez pratique, puisque pour y écrire des données, il faut impérativement un graveur sur l'ordinateur où se trouvent les données.

Leurs capacités peuvent varier de quelques mégaoctets à quelques Gigaoctets et se base sur de la mémoire EEPROM. Les premières clés avaient des capacités mémoires de 8MB à 32MB. Rapidement après, on trouvait des clés avec un minimum de 64 Mo, toutefois la taille standard était plutôt aux alentours de 256 Mo.

On trouve maintenant des clés USB basées sur des minis disques-durs, qui affichent des capacités encore plus importantes pour des prix plus raisonnables que les mémoires flash équivalentes. Ces périphériques sont toujours alimentés par le bus USB. Par rapport aux clés USB traditionnelles, les débits sont généralement meilleurs mais les temps d'accès sont plus importants. Ils sont aussi un peu plus fragiles et peuvent avoir tendance à chauffer en cas d'utilisation intensive, de plus leur taille est légèrement plus importante, toutefois ils tiennent toujours facilement dans la poche.

## 2. Définition

Une **clé USB** est un petit périphérique de stockage de données qui utilise une mémoire Flash et un connecteur USB.

Les clés USB sont alimentées par la connexion USB d'un ordinateur, sur lequel, celles-ci sont branchées. Une clé USB ne contient donc pas de batterie. Elles sont insensibles à la poussière et aux rayures, contrairement a d'autres supports telle la disquette ou le cd-rom, ce qui leur donne un indéniable avantage au niveau de la fiabilité.

De plus, c'est un périphérique qui supporte le protocole plug-and-play, c'est-à-dire que, dès la connexion du périphérique, l'utilisateur peut y accéder sans devoir installer divers pilotes supplémentaires.

Enfin, un des gros avantage des périphériques USB comme les clés, c'est la possibilité de brancher ceux-ci à chaud. Ainsi on ne doit pas éteindre l'ordinateur pour pouvoir les utiliser.

En théorie, la mémoire flash est censée conserver les données plus de dix ans, ce qui est plus que suffisant tant que le but n'est pas l'archivage de l'information.



### 3. Utilisation

L'accès à l'utilisation de ce support est relativement aisé, en effet, les systèmes d'exploitation couvrent une bonne partie de la configuration USB, et très peu d'ordinateur n'ont pas encore de support USB sur la carte mère (et si cela est le cas, il existe des cartes d'extensions PCI vers USB).

A partir de Windows 2000, il n'est même plus nécessaire de fournir un driver au système d'exploitation, les pilotes étant déjà intégrés dans celui-ci et la configuration se faisant automatiquement.

L'utilisation de la clé est très simple aussi : il suffit de l'insérer dans un port USB pour que l'ordinateur reconnaisse la clé comme un nouveau disque amovible, sans nécessiter de redémarrage (d'où le terme « HotPlug »). Après cela, la clé est accessible exactement comme pour tout disque dur.

Certains constructeurs utilisent maintenant le système de clé USB pour leurs lecteurs MP3. Ceux-la possèdent alors une batterie externe, mais elle n'est en rien nécessaire à la conservation des données dans la clé.

Pour le retrait de la clé, il est conseillé de suivre les instructions de retrait, car certains systèmes d'exploitation, pour gagner de la vitesse, utilisent la cache d'écriture sur le disque. C'est-à-dire que le système d'exploitation n'écrit pas tout de suite les données sur le disque mais dans une cache. Le problème, c'est que si une coupure de courant ou un retrait sans précaution a lieu, les données ne seront peut être pas écrites sur le disque et donc perdues. C'est pour cela que pour un périphérique amovible, comme les clés, il vaut mieux désactiver ce paramètre. Pour se faire, il suffit d'aller dans les propriétés de la clé et de cocher la case, « optimiser pour une suppression rapide ».

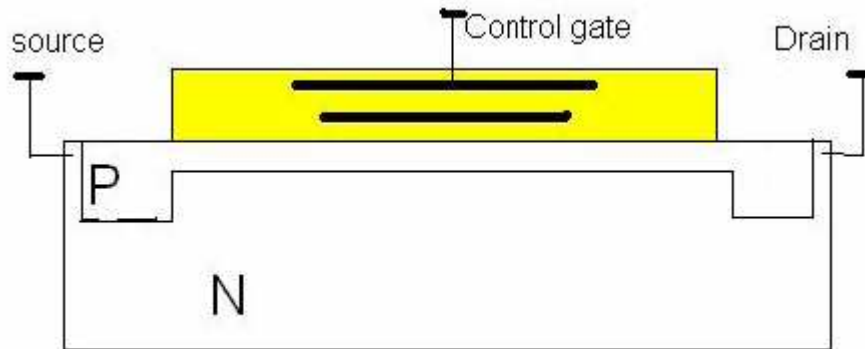
De plus en plus de carte mère supporte aussi maintenant le démarrage sur USB, ce qui permettra de remplacer à terme, les disquettes de boot. Cependant, peu d'outils proposent déjà cette possibilité dans leurs systèmes de sauvegarde. (Outils de récupération de données, de sauvegardes ou d'antivirus).

## 4. La mémoire Flash

### a/ l'EEPROM

La partie principale de la clef USB est bien sur sa mémoire. Cette mémoire est de type MOSFET EEPROM, quelque soit le type de flash (NAND ou NOR).

Voici un petit schéma et des explications sur le fonctionnement d'un MOSFET classique :



La barre se trouvant sous la « control gate » est appelée « floating gate ». C'est elle qui enregistre l'information. Le principe de fonctionnement est assez simple. Tant que l'EEPROM n'est pas programmée le courant passe entre la source et le drain. Bloquer le transistor est alors l'autre état logique. Pour faire cela, on met la source et le drain à la masse, puis on place une tension positive sur le « control gate ». Le « control gate » maintenant positif va attirer les électrons de la source au travers de la mince couche de substrat (zone P). Les électrons qui passent se retrouvent alors sur le « floating gate ». A la lecture, la faible tension entre la source et le drain ne permet plus de passer le champ électrique créé par les électrons bloqués dans le « floating gate ».

Les détails exact d'écriture sur la cellule dépendent de la structure (NAND ou NOR). Par contre, le problème d'effacement est commun à toutes les EEPROM. Pour effacer la mémoire, le « control gate » et la source sont mis à la masse, et on applique une tension au drain. Les électrons qui étaient sur le « floating gate » sont alors repoussés à la masse et la tension à l'intérieur du transistor redevient normale. Cet effacement se fait par blocs, ce qui veut dire que la réécriture de données sur un octet nécessite l'effacement d'une série d'octet autour avant de réécrire la totalité.

### b/ La mémoire NOR

Le premier type de mémoire flash à avoir été inventé, par Intel en 1988, était basé sur des portes NON-OU. Celle-ci avait l'avantage d'être à accès direct, mais les temps d'écriture et d'effacement étaient particulièrement longs. De plus, ces mémoires étaient assez coûteuses. Cette mémoire ne supporte que 10 à 100 milles effacements, après quoi le pont P est trop faible pour laisser passer le flux de courant. Il a été utilisé dans des PDA et des appareils à photos digitales. Les premières mémoires « Compact Flash » utilisaient aussi cette structure.

### **c/ La mémoire NAND**

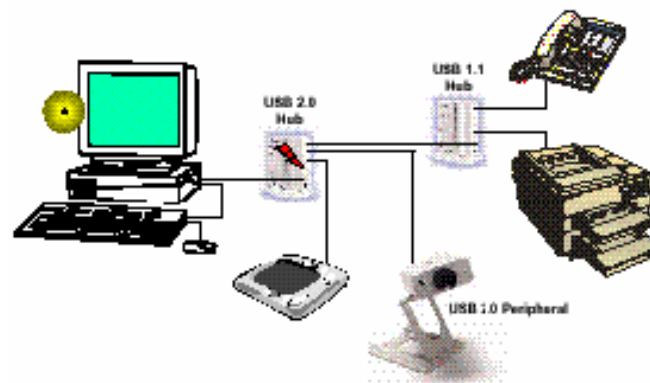
Sorti par Samsung et Toshiba en 1989, ces mémoires gagnent en vitesse en prix et en densité possible grâce à une taille réduite par rapport à la NOR. De plus, elles supportent 10 fois plus d'effacements et de reprogrammations que celle-ci. Cependant leur interface d'entrées sortie ne permet plus que l'accès séquentiel aux cellules.

La mémoire NAND, vu sa vitesse bien plus grande et sa bien meilleure intégration que la NOR est plus répandue dans l'utilisation des mémoires amovibles.

De nouvelle mémoire flash permette maintenant de stocker plusieurs bits sur 1 seul transistor en contrôlant la tension placée sur le « floating gate ». Une partie sensitive s'occupe de la reconversion depuis du champs électriques jusqu'aux bits. Ce système est appelé une cellule multi-level pour les différents niveaux de champs résultant du « floating gate ».

## 5. Introduction technique et norme du bus

Une équipe composée de Compaq, Hewlett Packard, Intel, Lucent, Microsoft, Nec et Philips a conduit le développement des spécifications de la norme USB version 2. La même équipe, à quelques exceptions près, avait déjà participé à l'élaboration de la version 1.1 en 1995. Cette nouvelle version augmente les capacités de transfert des données par un facteur de 4 tout en gardant une compatibilité descendante en conservant entre autre les mêmes câbles, les mêmes connecteurs et la même interface logicielle. L'utilisateur de périphériques USB ne verra aucun changement dans leur usage mais pourra bénéficier de l'ajout d'une gamme étendue de périphériques hautes performances tels que les caméras de vidéo conférence, les scanners et imprimantes, les périphériques de stockage rapides. La version 2 de la norme USB propose donc une amélioration de la bande passante par rapport à la version 1.1. Les périphériques à la norme 1.1 pourront se connecter sans problème sur tout PC équipé à la norme 2.



Le but principal de la norme USB était de définir une extension externe du bus rendant possible l'ajout de périphériques de façon simple tel que raccorder un poste téléphonique à une prise murale. Ceci fut possible par l'utilisation d'une architecture externe étendue comme renseignée dans la figure ci dessus. On y retrouve:

- Un PC hôte comprenant le contrôleur matériel et logiciel,
- des connecteurs et des câbles d'assemblage robustes,
- un bus extensible au travers de concentrateurs multi ports,
- des protocoles maître/esclave.

**Le rôle du système logiciel** est d'apporter une vue uniforme du système d'entrées/sorties pour toutes les applications logicielles. Cette couche va cacher les détails de l'implémentation matérielle de façon à rendre les applications logicielles plus portables. Pour le sous système d'entrées/sorties USB en particulier, il gère l'attachement et le détachement dynamique des périphériques. Cette phase appelée énumération, implique une communication avec les périphériques pour découvrir l'identité des pilotes qui doivent être chargés s'ils ne le sont pas encore. Une adresse unique est assignée à chaque périphérique durant l'énumération pour permettre des transactions de données. On retrouvera également pour ces périphériques, la possibilité de gestion de l'énergie sans intervention externe de l'utilisateur.

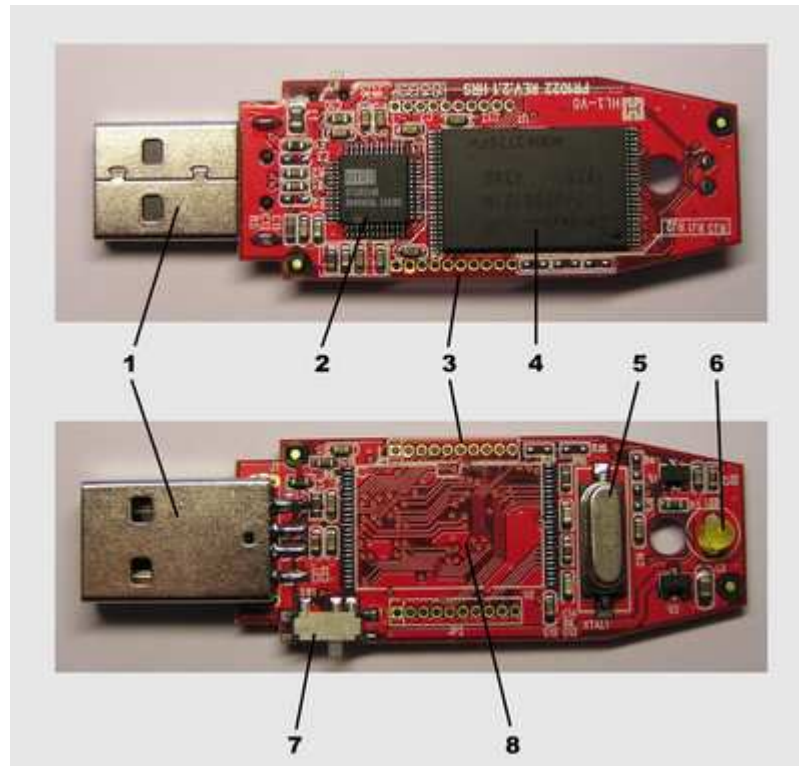
En plus d'apporter une connectivité additionnelle pour les périphériques USB, **le rôle du HUB** est d'assurer la gestion de l'alimentation électrique des périphériques attachés. Il reconnaît dynamiquement l'attachement de périphériques et leur fournit

au moins 0.5W de puissance par périphérique durant l'initialisation. Sous contrôle du logiciel hôte, le HUB peut procurer plus de puissance par périphérique jusqu'à un maximum de 2.5W. Un HUB nouvellement attaché va se voir assigner une adresse unique et *les HUB peuvent être mis en cascade jusqu'à 5 niveaux*. Durant le fonctionnement, un HUB agit comme un répéteur bidirectionnel et répétera les signaux de et vers l'hôte. Le HUB va également analyser ces signaux et détecter les transactions qui lui sont adressées. Un HUB supporte les périphériques à 12Mb/s (pleine vitesse) et 1.5Mb/s (basse vitesse). Ces vitesses correspondent à la norme 1.1.

**Tous les périphériques USB** sont des esclaves qui doivent obéir à un protocole défini. Ils doivent réagir à des demandes de transactions qui sont envoyées par l'hôte. Le périphérique envoie et reçoit des données vers/de l'hôte en utilisant un format standard de données USB. Les périphériques USB à la norme 1.1 peuvent travailler à 12Mb/s ou 1.5Mb/sec. USB 2.0 est une évolution des spécifications de USB 1.1. La plupart des câbles et connecteurs de la norme 1.1 ont été testés et confirment la compatibilité descendante de la nouvelle norme. Une vitesse de transfert de 480Mb/sec peut être atteinte par la norme. L'utilisation de trames plus petites permet l'utilisation de buffers petits malgré la vitesse de transfert importante. L'utilisation d'un HUB à la norme 2.0 permet, si des périphériques de la même norme sont utilisés, une vitesse de transfert de 480Mb/sec. Si un périphérique de la norme 1.1 est connecté sur ce même HUB, celui-ci permettra des transferts à 12 ou 1.5 Mb/sec en s'adaptant au périphérique.

## 6. Spécifications physique

### a/La clé

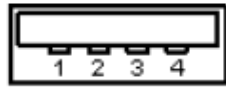


Après ouverture et retrait de la coque en plastique, l'électronique de la clé apparaît (ici un modèle de 64 MB USB 2.0).

1. Connecteur USB mâle (type A)
2. Contrôleur « *Ours Technology Inc. OTi-2168 USB 2.0* ». Ce circuit implémente le contrôleur pour l'USB 2.0 et assure une interface entre des données transmises linéairement et la structure en blocs de la mémoire flash. Il permet d'éviter la gestion bas-niveau de la mémoire et contient un petit microprocesseur RISC ainsi qu'un peu de RAM et de ROM. Les données sont transmises au « *Hynix* » (n°4) via un bus de données/adresses sur 8 lignes.
3. JP1 et JP2 : deux connecteurs avec 10 pins, principalement utilisés pour les tests et les débogages.
4. « *Hynix Semiconductor HY27USxx121M* », mémoire flash qui contient 4096 blocs indépendants (chacun avec 16 KBytes), soit 64 MB au total.
5. Un oscillateur cristal « *SKC Shin Chang Electronics* » cadencé à 12 MHz. En faite c'est l'horloge du microprocesseur RISC.
6. Une LED jaune pour indiquer l'activité de la clé.
7. Un interrupteur à deux positions pour protéger la clé en écriture.
8. Zone vierge mais prête à recevoir une autre mémoire flash pour offrir un modèle de 128 MB sans avoir à créer un autre schéma.

## b/ Les connecteurs

Tous les appareils ont une connexion amont vers l'hôte et tous les hôtes ont une connexion aval vers l'appareil. Les connecteurs amont et aval ne sont pas interchangeables mécaniquement, éliminant ainsi les connexions de rebouclage interdite aux Hubs comme pour un port aval connecté à un port aval. Il y a généralement 2 types de connecteurs, appelé type A et type B présenté ci-dessous.



Connecteur USB type A



Connecteur USB type B

Les prises mâles de type A sont toujours tournés vers l'Amont. Les prises femelles de type A se trouveront généralement sur les hôtes et les Compact Flash Par exemple, les prises femelles de type A sont courantes sur les cartes mères des ordinateurs et les Compact Flash

Les prises mâles de type B sont toujours connectées vers l'aval et par conséquent les prises femelles de type B se trouvent sur les appareils.

Il peut-être intéressant de trouver des câbles de type A vers type A avec un câblage direct et une matrice de changeur de genre USB dans certains magasins d'ordinateurs. C'est en contradiction avec la spécification USB. Les seuls appareils avec prise type A vers prise type A sont des ponts que l'on utilise pour brancher 2 ordinateurs entre eux.

D'autres câbles prohibés sont les extensions USB qui ont une prise mâle à une extrémité (soit de type A ou de type B) et une prise femelle à l'autre. Ces câbles ne respectent pas les exigences de longueur de câble de l'USB.

L'USB 2.0 comprenait un errata qui présente les minis connecteurs USB de type B. On peut trouver les détails de ces connecteurs dans la notice changement technique des minis connecteurs B. La raison d'être des minis connecteurs est issue de la gamme d'appareils électroniques miniatures comme les téléphones mobiles et les agendas électroniques. Le connecteur ordinaire de type B est trop grand pour être facilement intégré à ces appareils.

Il vient de paraître une spécificité On-the-Go qui ajoute la fonctionnalité paire à pair (peer to peer) à l'USB. D'où l'introduction des hôtes USB dans les téléphones mobiles et les agendas électroniques, de même qu'une particularité pour les prises mâles mini A, les prises femelles mini A et les prises femelles mini A-B. Tout porte à croire que nous devrions être inondés de câbles mini USB et d'une gamme de câbles convertisseurs de la taille mini à standard.



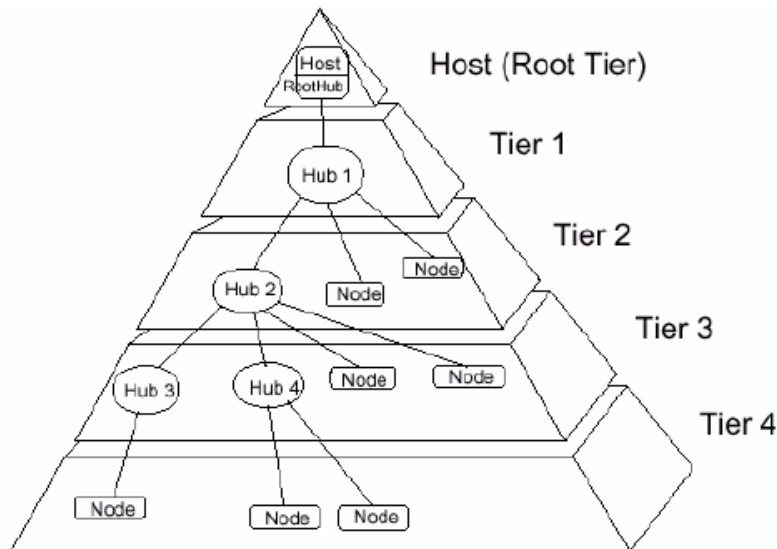
Numéro de broches	Couleurs des câbles	Fonction
1	Rouge	V <sub>BUS</sub> (5 volts)
2	Blanc	D-
3	Vert	D+
4	Noir	Masse

On utilise des couleurs standard pour les fils intérieurs des câbles USB de façon à faciliter l'identification des fils d'un constructeur à un autre. La normalisation précise les différents paramètres électriques pour les câbles.

## c/Le Bus

- La topologie de bus

**L'Universal Serial Bus (USB)** connecte les composants USB avec l'hôte USB. L'interconnexion physique USB se base sur une topologie en étoile. Un HUB est le centre de chaque étoile. Chaque segment est une connexion point à point entre l'hôte et un HUB ou un HUB connecté sur un autre HUB. La topologie est illustrée par la figure ci après :



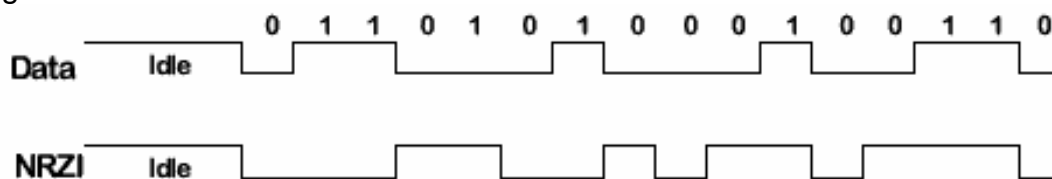
Comme indiqué plus haut, ainsi que sur le schéma, on ne pourra pas mettre de HUB sur plus de cinq niveaux.

- L'hôte USB

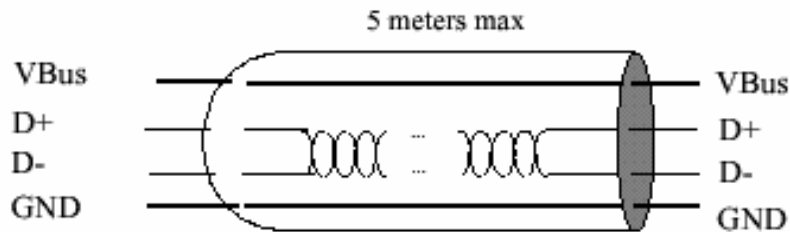
Il y a seulement un hôte dans un système USB. Le contrôleur hôte peut être implémenté dans une combinaison matérielle, logicielle ou firmware. Un HUB racine est intégré dans le système hôte pour permettre un ou plusieurs points d'attache.

- Caractéristiques électriques

Les signaux USB et la puissance sont transférés via un câble comprenant 4 fils. Les signaux sont différentiels et le câble a une impédance caractéristique de 90 Ohms. L'horloge est transmise de façon codée en même temps que le signal différentiel. Le schéma d'encodage de l'horloge est du NRZI. Un champ de synchronisation (SYNC) précède chaque paquet pour permettre au récepteur(s) de synchroniser leurs horloges.



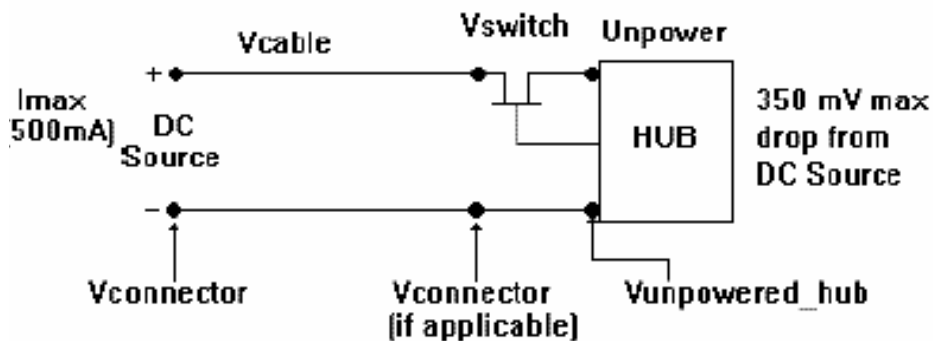
Dans le code NRZI, un 1 est représenté par aucun changement dans le niveau tandis qu'un zéro est représenté par un changement dans le niveau. Le câble va également transporter VBus et GND sur des fils distincts pour délivrer la puissance aux composants. VBus est de +5V à la source. USB autorise des segments de câble de longueur variable pouvant aller jusque plusieurs mètres (5m maximum).



Les spécifications électriques de la couche physique de USB imposent que le maximum de variation de la tension d'alimentation entre deux HUB ou entre les HUB et les fonctions(périphériques) soient de 350 mV.

Nous retrouverons la formule suivante:

$$V_{\text{unpowered\_hub}} = V_{\text{switch}} + 4 \cdot V_{\text{connector}} + 2 \cdot V_{\text{cable}}$$



$$V_{\text{switch}} = I_{\text{max}} \cdot \text{résistance FET} = 100\text{mV}$$

$$V_{\text{connector}} = I_{\text{max}} \cdot 30\text{m}\Omega \text{ (résistance du connecteur)} = 15\text{mV}$$

$$V_{\text{cable}} = I_{\text{max}} \cdot \text{résistance du câble (donnée en } \Omega/\text{m)}$$

$$I_{\text{max}} = 500\text{mA}$$

$$\text{Nous avons donc: } 350\text{mV} = 100\text{mV} + 4 \cdot 15\text{mV} + 2 \cdot V_{\text{cable}}$$

$$V_{\text{cable}} = (350 - 100 - 60) / 2 = 95\text{mV}$$

$$95\text{mV} / 500\text{mA} = 0.19\Omega$$

Resistance	Longueur
0.232Ω/m	.81m
0.145Ω/m	1.31m
0.091Ω/m	2.08m
0.057Ω/m	3.33m
0.036Ω/m	5.00m

Chaque segment USB apporte une quantité limitée de puissance sur le câble. L'hôte fournit la puissance aux composants qui lui sont directement connectés. En plus, chaque composant USB peut avoir sa propre alimentation.

Les composants USB qui sont totalement dépendant de la puissance fournie par le câble sont appelés composants alimentés par le bus.

Inversement, ceux qui ont leur propre source d'alimentation sont appelés composants auto alimentés.

Pour simplifier le concept de puissance fournie ou consommée, on utilisera le concept de charge unitaire (unit load). Une charge unitaire est définie comme étant de 100mA. Un HUB alimenté par le bus tire toute la puissance pour ses fonctions internes et ses ports en aval des bornes de puissance du connecteur USB. Il peut tirer 1 charge dès sa mise sous tension avec un total de 5 charges qui sont réparties entre les fonctions internes et les ports externes. Chaque port externe peut fournir uniquement 1 charge par port indifféremment du courant tiré sur les autres ports du HUB. Un HUB auto alimenté peut tirer jusqu'à une charge à partir de sa connexion en amont pour permettre à l'interface de fonctionner même si le reste du HUB est hors tension. Le HUB peut fournir 5 charges unitaires sur tout ses ports externes en aval. Le HUB sur un hôte dans un ordinateur de type « desktop » est un HUB auto alimenté. Le même HUB dans un portable peut être soit auto alimenté ou alimenté par le bus.

Tous les composants, qu'ils soient alimentés par le bus ou autonomes peuvent seulement tirer du courant sur le bus. Ils ne peuvent pas fournir de courant en amont vers un hôte ou vers un HUB. Lors de leur mise sous tension, tous les composants doivent s'assurer que leur port en amont n'est pas désactivé de manière à permettre la réception du signal de reset et que le maximum de courant de fonctionnement tiré soit d'une charge unitaire.

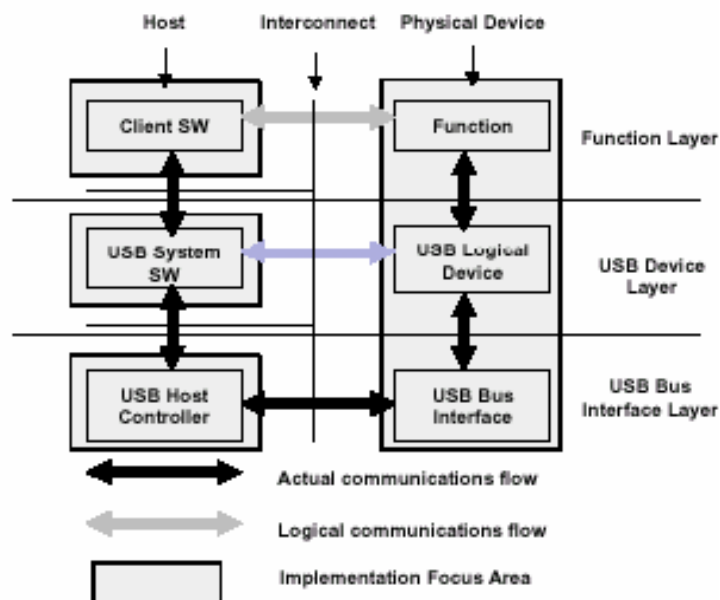
Du fait que les fonctions internes intégrées du HUB alimenté par le bus nécessitent 1 charge et que le HUB ne puisse tirer que 5 charges sur le bus, on ne pourra donc retrouver sur ces HUB que 4 ports externes.

Pour le HUB auto alimenté, le nombre de ports en aval est uniquement limité par la puissance pouvant être fournie en local par le HUB et par des raisons de sécurité la normalisation prévoit l'obligation d'une protection contre les surcharges à 5A.

## 7. Le modèle de flux de données de l'USB

### a/Les différentes couches

L'USB se présente sous une structure en couche comme représentée dans la figure ci-dessous ;



On retrouve notamment 4 zones de concentration:

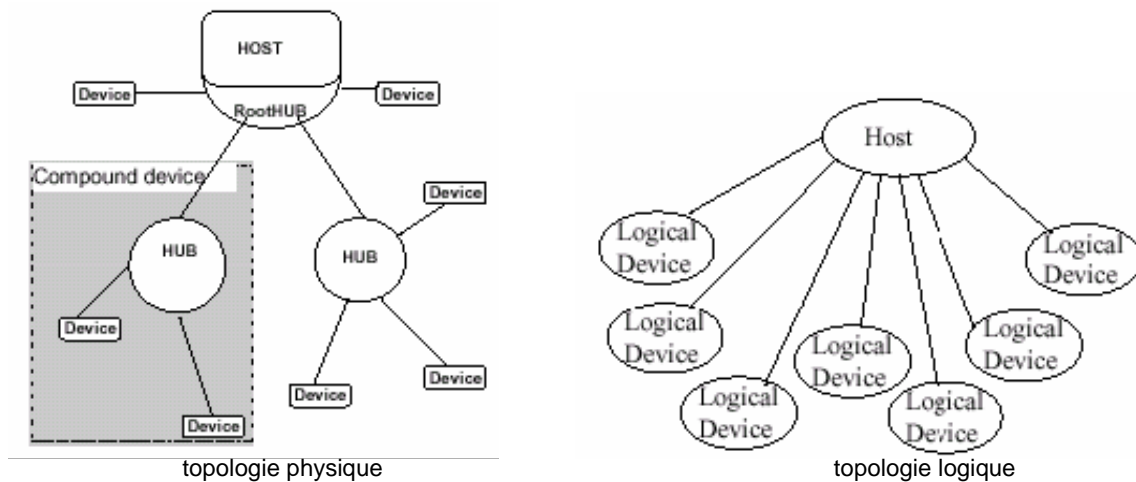
- Le composant physique USB: c'est une partie matérielle se situant à la fin d'un câble USB qui effectue certaines fonctions utilisatrices utiles.
- Le logiciel client: c'est le logiciel qui est exécuté sur l'hôte en correspondance avec un composant USB. Typiquement, ce logiciel est fourni avec le système d'exploitation ou avec le composant USB.
- Le logiciel système USB: c'est le logiciel qui supporte l'USB dans un système d'exploitation particulier, indépendamment du logiciel client ou d'un composant USB.
- Le contrôleur hôte USB (Hôte côté interface du bus): ce sont le logiciel et le matériel qui permettent à un composant USB d'être attaché à l'hôte.

La figure montre une connexion simple d'un hôte et d'un composant au travers d'un certain nombre de couches et d'entité. On retrouve les couches suivantes:

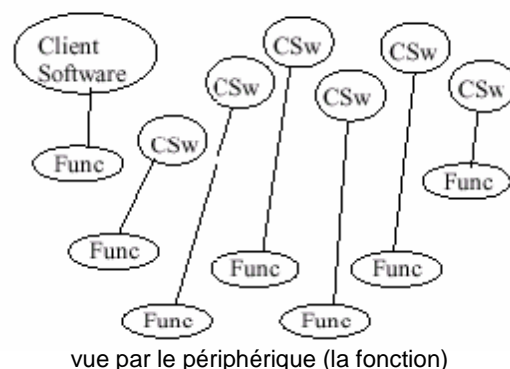
- La couche des fonctions: cette couche apporte des possibilités additionnelles à l'hôte par l'intermédiaire d'une couche logicielle client associée appropriée.
- La couche des composants USB: cette couche est la vue qu'a le logiciel système USB pour effectuer des opérations de base USB avec un composant.
- La couche d'interface de bus USB: la couche d'interface de bus USB va apporter la connexion aux niveaux physiques, signaux et paquets entre un hôte et un composant.

## **b/La topologie du bus**

Au niveau de la topologie du bus, on peut distinguer la topologie de bus physique qui a été décrite dans un paragraphe précédent et la topologie logique. Tandis que les composants sont physiquement attachés à l'USB via une topologie en étoile câblée, l'hôte communique avec chaque composant logique comme si ils étaient directement connectés sur les ports du concentrateur racine.



Les concentrateurs sont des composants logiques mais ils ne sont pas représentés pour simplifier le schéma. Même si le plupart des activités hôtes/composants utilisent cette perspective logique, l'hôte mémorise la topologie physique pour prendre en charge la déconnexion des concentrateurs. En effet, lorsque un concentrateur est déconnecté, tous les composants lui étant connectés par ses ports doivent être supprimés de la vue de la topologie logique. Même si les topologies physique et logique de l'USB reflètent la nature partagée du bus, le logiciel client manipulant l'interface client USB est présenté avec la vue qu'il a uniquement des relations avec son interface. Le logiciel client ne doit pas manipuler ses fonctions via des zones mémoire spécifiques ou des commandes d'entrées/sorties comme pour les autres bus mais uniquement passer par une interface de programmation USB.

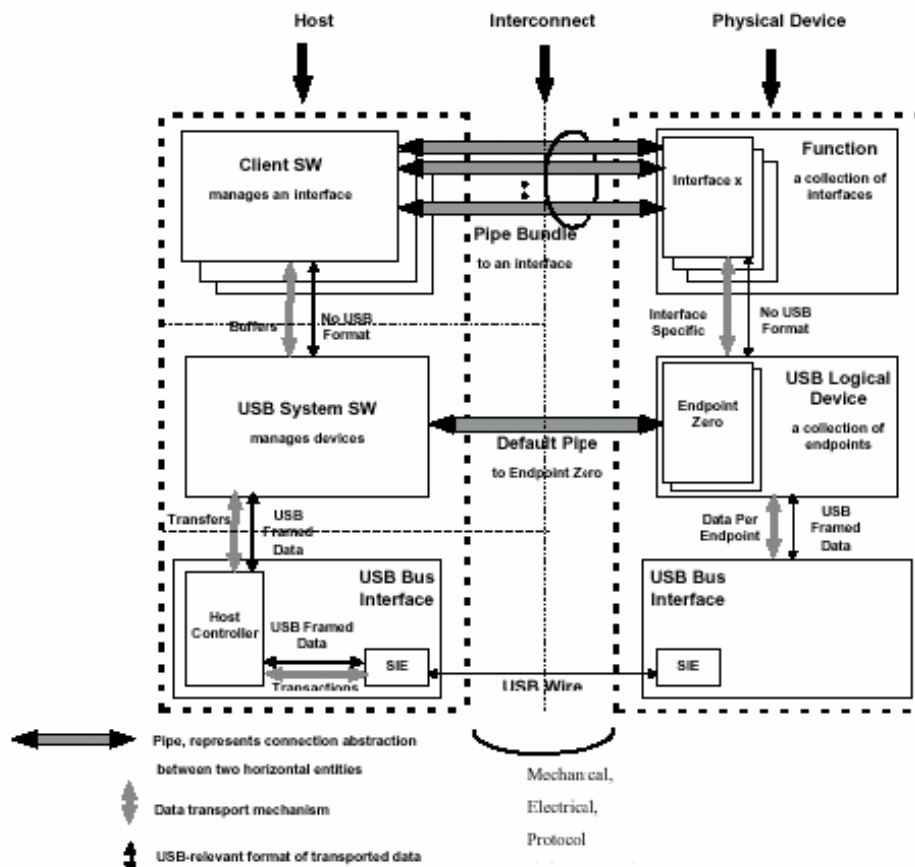


Le logiciel client communique directement avec l'interface de son composant USB sans se soucier de la structure physique de l'USB. Chaque couche de l'hôte apparaît comme communiquant directement avec la couche correspondante du composant USB.

## c/Le point final

Un point final (end point) est une partie unique identifiable d'un composant USB qui est l'extrémité d'un flux de communication entre l'hôte et un composant. Chaque composant logique USB est composé d'une collection de points finaux indépendants. Le logiciel peut seulement communiquer avec un composant au moyen d'un ou plusieurs points finaux. Chaque composant logique a une adresse unique qui lui a été assignée par le système durant l'attachement. Chaque point final sur un composant a un identificateur unique, c'est le numéro de point final (endpoint number). La combinaison de l'adresse du composant et du numéro de point final permet à chacun de ces points d'être référencé de façon unique. Un point final a des caractéristiques qui déterminent le type de transfert nécessaire entre lui et le logiciel client. Les points finaux se décrivent deux mêmes par:

- Leur fréquence d'accès au bus et leur temps d'attente (latency time)
- Leur bande passante nécessaire
- Leur numéro de point final
- Leur exigence sur la gestion des erreurs
- La taille maximale des paquets qu'un point final est capable d'envoyer ou de recevoir.
- Le type de transfert



Tous les composants USB doivent avoir un point final ayant le numéro 0 qui est utilisé pour initialiser et manipuler au départ le composant logique USB. Il permet d'avoir accès aux informations de configuration du composant.

Le point final 0 est toujours configuré dès que le composant USB est attaché et mis sous tension.

Un canal (pipe) est une association entre un point final sur un composant et l'hôte. Ces canaux représentent la possibilité de faire passer des données entre le logiciel sur l'hôte via un buffer mémoire et un point final sur un composant.

Il y a deux modes de communications par canaux:

- Stream (flot)- Les données à travers les canaux n'ont pas une structure USB définie. Elles sont transmises dans la portion de paquet de données des transactions sur le bus. Les données arrivent dans l'ordre dans lequel elles ont été transmises. Un tel canal vers un composant est lié à un simple numéro de point final dans une direction appropriée. Tout échange avec le même numéro mais dans la direction opposée doit se faire via un autre canal.
- Message- Les données à travers les canaux ont quelques structures USB définies.

USB ne va pas interpréter le contenu de la donnée qui est délivrée par le canal. Même si un canal de type message nécessite que la donnée soit structurée en accord avec la norme USB, le contenu même de celle-ci n'est pas interprétée par l'USB. Un canal de message interagit avec un point final d'une manière tout à fait différente que pour un canal de flot. Une requête est dans un premier temps envoyée de l'hôte vers le composant USB. Cette requête est suivie par les transferts de données dans la direction appropriée. Finalement une phase d'état suit un peu plus tard par l'envoi d'une réponse du point final.

Les canaux de messages permettent un flot de communication bidirectionnelle bien que ce flot soit de façon prédominante dans un seul sens. Le canal vers le numéro de point final 0 est toujours un canal de message. Des logiciels clients multiples sur l'hôte peuvent envoyer des requêtes via le canal par défaut mais elles sont envoyées vers le point final dans l'ordre premier envoyé, premier traité. Les canaux seront également utilisés en relation avec:

- Une demande d'accès sur le bus USB avec une utilisation de bande passante
- Un type de transfert donné
- Les caractéristiques associées avec un point final.

Puisqu'il existe au moins un point final portant le numéro 0, on retrouvera au moins un canal qui sera appelé canal par défaut. Le logiciel client demande un transfert de données via un IRP (I/O Request Packets) transmis à un canal et ensuite soit il attend, soit il est averti quand la transaction sur le bus est terminée avec succès ou par des erreurs (trois erreurs successives interrompent la transaction).

Si il n'y a pas d'IRP en attente ou en cours de traitement pour un canal, celui ci est en inactivité et le contrôleur hôte ne prendra aucune action vis à vis de ce canal; le point final pour un tel canal ne verra aucune transaction vers lui.

## 8. Les types de transferts

### a/introduction

USB transporte les données au travers un canal entre le buffer mémoire associé à un logiciel client sur l'hôte et un point final sur un composant USB. La donnée transportée par des canaux de messages sont transportées dans une structure USB définie mais USB permet à des données structurées spécifiques à des composants d'être transportées dans un message de données défini.

USB définit aussi que la donnée transportée sur le bus doit être découpée en paquets pour n'importe quel canal, mais le formatage et l'interprétation finales de la donnée transportée dans cette zone de donnée de la transaction sur le bus reste la responsabilité du logiciel client et des fonctions utilisant le canal.

Cependant, USB apporte différents types de transfert qui sont optimisés pour convenir le mieux au service demandé. Un IRP utilise une ou plusieurs transactions de bus pour déplacer des informations entre le logiciel client et les fonctions associées.

Chaque type de transfert détermine des caractéristiques variées du flot de communication incluant:

- Le format de données imposé par USB
- La direction de la communication
- La contrainte de la taille des paquets
- Les contraintes d'accès sur le bus
- Les séquences de données nécessaires

Le concepteur d'un composant USB choisit les possibilités de chacun des points finaux du composant.

Quand un canal est établi pour un point final, la plupart des caractéristiques de transfert sont déterminées et restent inchangées durant la durée de vie du canal.

USB définit quatre types de transfert:

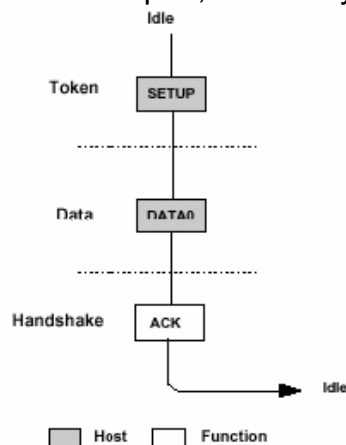
- Transfert de contrôle: de façon non périodique, le logiciel hôte va initialiser une communication demande/réponse utilisée typiquement pour les opérations de commandes et d'états.
- Transfert isochrone: communication périodique et continue entre l'hôte et le composant typiquement utilisée pour des informations relatives au temps.
- Transfert d'interruption: petites données, non périodique, faible fréquence. Le composant initialise une communication typiquement utilisée pour avvertir l'hôte que le composant a besoin d'un service.
- Transfert en groupe: communication avec des rafales très larges utilisée pour des données qui peuvent utiliser la moindre bande passante disponible et qui peut être retardée jusqu'à ce que toute la bande passante soit disponible.

**Dans les paragraphes suivants, nous abordons les types de transferts. Nous parlons aussi dans ces paragraphes des différents paquets de données qui eux, sont abordés dans le chapitre traitant de la couche de protocole qui se trouve un peu plus loin dans le dossier.**

## b/Les différents types de transfert

- Transfert de contrôle

Les transferts de contrôle ont au minimum deux étapes de transaction: le setup et le status. Un transfert de contrôle peut contenir de façon optionnelle une étape de données entre l'étape de setup et l'étape de status. Durant l'étape de setup, une transaction de type setup est utilisée pour transmettre des informations vers un point final d'une fonction. Les transactions de setup sont similaires au format d'un OUT mais un PID de setup sera utilisé au lieu d'un PID OUT. Un setup utilise toujours un PID DATA0 pour le champ de données de la transaction de setup. La fonction recevant un SETUP doit accepter la donnée de setup et répondre avec une poignée de main ACK ou si la donnée est corrompue, ne renvoyer aucune poignée de main.



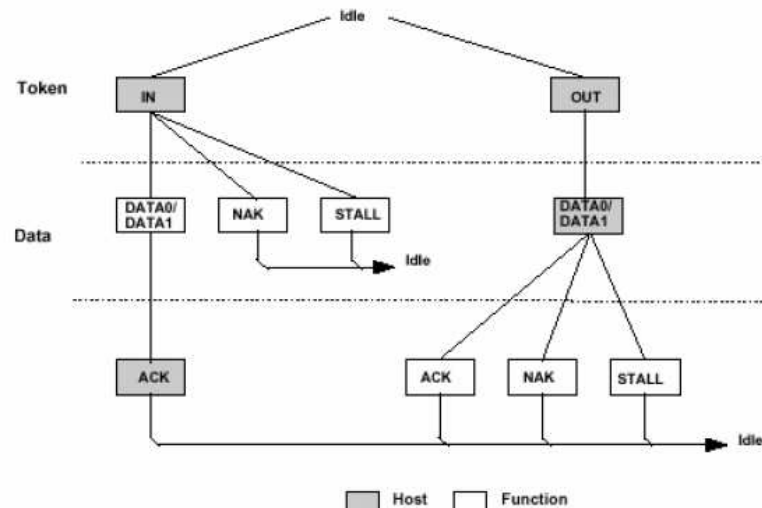
L'étape de données, si elle est présente dans le transfert de contrôle, consiste en une ou plusieurs transactions IN ou OUT et suivent les mêmes règles de protocole qu'un transfert en rafales. Toutes les transactions dans l'étape des données ont la même direction, toutes IN ou toutes OUT. Le nombre de données qui vont être transmises durant cette phase et leur direction sont spécifiés durant l'étape de setup. Si la quantité de données excède la taille des paquets prénégociée, la donnée est envoyée dans de multiples transactions IN ou OUT. L'étape de status dans un transfert de contrôle est la dernière opération de la séquence. Une étape de status est représentée par un changement de direction dans le flot de données et utilise toujours un PID DATA1. La figure suivante montre l'ordre des transactions, la valeur du bit de séquence et le type de données PID. Les différents bits de séquence sont représentés entre parenthèse.

L'étape de status va permettre le rapport vers l'hôte du résultat du transfert précédent de la donnée et du setup. Trois résultats sont possibles:

- La séquence de commande a été complétée avec succès
- La séquence de commande a échoué
- La fonction est encore occupée à compléter la commande

- Transfert en rafale (en groupe)

Les transactions en rafale sont caractérisées par la capacité de garantir une transmission de données sans erreur entre l'hôte et la fonction au moyen d'une détection d'erreur avec renvois éventuels. Une transaction en rafale consiste en trois étapes: transfert d'un paquet token, d'un paquet de données et d'un paquet de poignée de main. Dans certaines conditions de flux et de calage, la phase de données peut être remplacée avec une poignée de main résultant d'une transaction en deux étapes avec aucune donnée transmise.



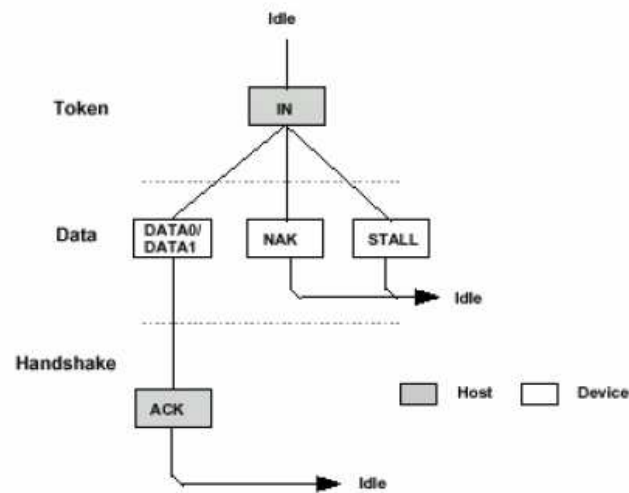
**Quand un hôte désire recevoir** des données en rafale, il envoie un token IN. Le point final de la fonction répond en envoyant soit un paquet de données ou si il est incapable de retourner la donnée, une poignée de main NAK ou STALL. Un NAK indique que la fonction est temporairement incapable de retourner la donnée tandis qu'un STALL indique que le point final est calé de façon permanente et nécessite de ce fait une intervention du software de l'hôte. Si l'hôte reçoit une donnée valide, il répond avec une poignée de main ACK. Si l'hôte détecte une erreur durant la réception des données, il ne retourne aucun paquet vers la fonction.

**Quand l'hôte souhaite transmettre des données en rafale**, il envoie en premier un paquet token de type OUT suivi par un paquet de données. La fonction va alors retourner une des trois poignées de mains. ACK indique que le paquet de données a été reçu sans erreur et informe l'hôte qu'il peut envoyer le paquet suivant dans la séquence. NAK indique que la donnée a été reçu sans erreur mais que l'hôte devrait renvoyer la donnée parce que la fonction était dans une condition temporaire ne lui permettant pas d'accepter cette donnée à ce moment (ex: buffer plein). Si le point final était calé, STALL est retourné pour indiquer que l'hôte ne doit pas réessayer de transmettre la donnée parce que il y a une condition d'erreur sur la fonction. Si le paquet de données a été reçu avec une erreur de CRC ou de bits de remplissage, aucune poignée de main n'est renvoyée.

L'hôte initialise toujours la première transaction du transfert sur le bus avec un PID DATA0. La seconde transaction utilise un PID DATA1 et les transferts successifs de données alternent les séquences pour le reste du transfert en rafale.

- Les transferts d'interruption

Les transactions d'interruption consistent en un seul paquet IN. Dès réception d'un token IN, une fonction peut envoyer des données, NAK ou STALL. Si le point final n'a pas de nouvelles informations d'interruption à retourner (ex: il n'y a aucune interruption en attente), la fonction retourne un NAK durant la phase de données. Si une interruption est en attente, la fonction retourne l'information d'interruption comme un paquet de données. L'hôte, en réponse à ce paquet de données, envoie soit un ACK si la donnée a été reçue sans erreur où ne retourne rien si le paquet était corrompu.



- Les transferts isochrones.

Dans n'importe quel système de communication, le transmetteur et le récepteur doivent être suffisamment synchronisés pour délivrer les données de façon robuste. Dans un système de communication asynchrone, les données peuvent être délivrées de façon robuste en permettant au transmetteur de détecter que le récepteur n'a pas reçu les données correctement et de simplement les retransmettre.

Dans un système de communication isochrone, le transmetteur et le récepteur reste synchronisés au niveau de l'horloge et des données pour délivrer la donnée de façon robuste.

USB ne supporte pas les retransmissions de paquets erronés de façon à ce que un minimum de bande passante puisse être alloué pour le transfert sans perdre le moindre temps de synchronisation qui serait du à des retransmissions de paquets erronés. Il est critique que la paire transmetteur/récepteur dans une transmission isochrone reste synchronisée que ce soit dans une transmission normale ou dans le cas ou des erreurs se produisent sur le bus.

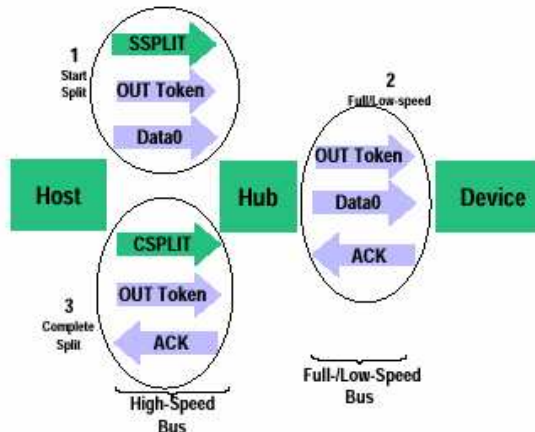
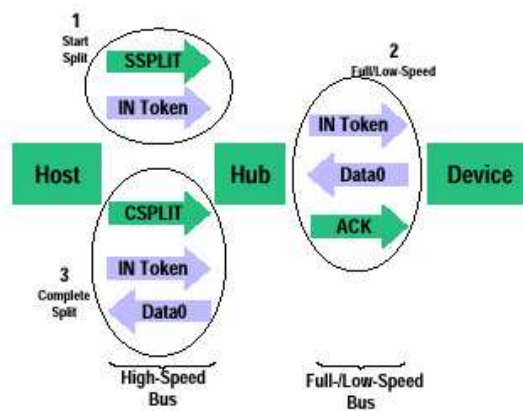
Dans la plupart des systèmes qui échangent des données en isochrone, une simple horloge globale est utilisée et elle permet à toutes les entités dans le système de se synchroniser. Du fait qu'une large variété de périphériques avec des fréquences naturelles différentes puissent être attachés à l'USB, aucune horloge unique ne peut satisfaire les nécessités de synchronisation de tous les périphériques et logiciels. USB va donc définir un modèle d'horloge qui va permettre à une gamme étendue de périphériques de coexister sur le bus tout en ayant un coût de mise en oeuvre raisonnable.

- Les transactions SPLITS. (uniquement pour la norme USB 2)

La transaction SPLIT à très haute vitesse est utilisée seulement entre un contrôleur hôte et un HUB quand le HUB a des périphériques lents et rapides qui lui sont connectés. Une transaction SPLIT à haute vitesse est utilisée pour initialiser une transaction à basse vitesse ou vitesse élevée via le HUB vers un point final d'un périphérique lent ou rapide. Cette transaction permet de récupérer du HUB l'état d'une transaction basse vitesse ou vitesse élevée.

On retrouve deux parties dans cette transaction: un start-split (début de transaction SSPLIT) et un complete-split (transaction terminée CSPLIT).

On va retrouver dans les deux schémas suivants, les successions des différents paquets permettant à l'hôte de transmettre à très haute vitesse vers le HUB tandis que celui-ci transmet les données de/vers le périphérique à basse vitesse ou vitesse élevée.



Cette approche permet à un contrôleur hôte de démarrer une transaction à basse/grande vitesse via une transaction à vitesse élevée et de pouvoir continuer avec d'autres transactions à vitesse élevée sans avoir à attendre que la transaction à basse/grande vitesse soit terminée et ce à vitesse plus basse.

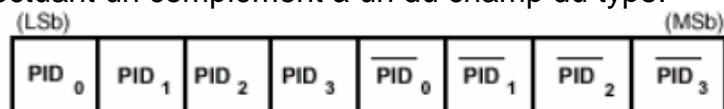
## 9. La couche protocole

### a/Champ de synchronisation

Tous les paquets commencent avec un champ de synchronisation (SYNC) qui est une séquence codée qui génère une densité de flancs de transition importante. Ce champ apparaît sur le bus comme un temps d'inactivité suivi d'une chaîne de caractères 'KJKJKJKK' dans son encodage NRZ1. C'est utilisé par le circuit d'entrée pour aligner les données entrantes avec l'horloge locale. SYNXC sert uniquement de mécanisme de synchronisation.

### b/Champ d'identification de paquet

Un paquet d'identification (PID) suit immédiatement un champ de synchronisation sur chaque paquet USB. Un PID est constitué d'un champ de type composé de 4 bits suivi par un autre de vérification composé lui aussi de 4 bits. Le PID indique le type du paquet et de ce fait son format et le type de détection d'erreurs qui lui est appliqué. Le champ de vérification du PID assure un codage fiable du PID de sorte que le reste des paquets soit interprété correctement. Le champ de vérification PID est généré en effectuant un complément à un du champ du type.



L'hôte et toutes les fonctions doivent effectuer un décodage complet de tous les champs PID reçus. Tout PID reçu avec un champ de contrôle mauvais ou avec des codes ayant une valeur non définie seront considérés comme corrompus et de ce fait, le paquet PID mais aussi les suivants sont ignorés. Si la fonction reçoit un autre PID valide pour un type de transaction ou une direction qui n'est pas supportée par la fonction, celle ne répondra pas à la demande.

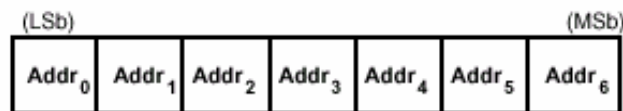
PID Type	PID Name	PID[3:0]	Description
Token	OUT	b0001	Address + endpoint number in host -> function transaction
	IN	b1001	Address + endpoint number in function -> host transaction
	SOF	b0101	Start of frame marker and frame number
	SETUP	b1101	Address + endpoint number in host -> function transaction for setup to a control endpoint
Data	DATA0	b0011	Data packet PID even
	DATA1	b1011	Data packet PID odd
Handshake	ACK	b0010	Receiver accepts error free data packet
	NAK	b1010	Rx device cannot accept data or Tx device cannot send data
	STALL	b1110	Endpoint is stalled
Special	PRE	b1100	Host-issued preamble. Enables downstream bus traffic to LS devices.

Les PID sont divisés en quatre groupes de codes: token, data, handshake ou spécial avec les deux premiers bits indiquant le groupe.

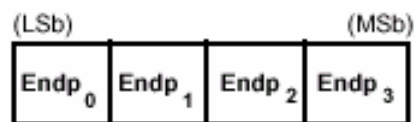
### c/ Les champs d'adresse

Les points d'entrées d'une fonction sont adressés en utilisant deux champs: le champ d'adresse de la fonction et le champ du point final. Une fonction a besoin de décoder complètement à la fois le champ d'adresse et le champ du point final.

Le champ d'adresse de la fonction (ADDR) spécifie la fonction via son adresse qui est soit la source ou la destination des paquets de données, dépendant de la valeur du "token" PID. Comme indiqué dans la figure ci après, on retrouve un total de 128 adresses différentes (adresse codée sur 7 bits). Un champ d'adresse est spécifié pour les "token" IN, OUT et SETUP. Dès leur mise sous tension et leur initialisation, l'adresse par défaut d'une fonction est 0 et doit être programmée par l'hôte durant le procédé d'énumération. L'adresse 0 est réservée et ne peut être assignée pour un mode normal de travail.



Le champ du point final est codé sur quatre bits (ENDP). Ce champ est défini pour les "token" PID de type IN, OUT et SETUP. Toute fonction doit supporter un point final qui porte le numéro 0. Les périphériques à basse vitesse supportent un maximum de deux points finaux par fonction. Les fonctions à grande vitesse peuvent supporter jusqu'à 16 points finaux.

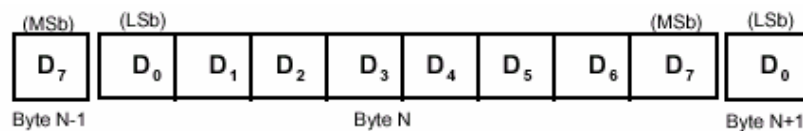


### d/ Le champ de numéro de trame

Le champ de numéro de trame est un codé sur 11 bits. Ce numéro est incrémenté par l'hôte lors de chaque trame transmise. Ce champ est transmis uniquement pour les "floating gate" SOF.

### e/ Le champ de données

Le champ de données peut comprendre de 0 à 1023 bytes et doit être un nombre entier de byte. La figure suivante montre le format lorsque de multiples octets sont transmis.



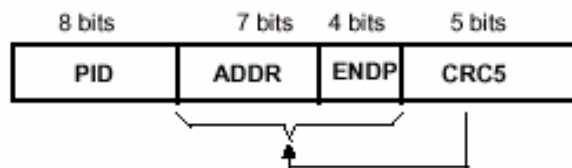
La taille des paquets varie en fonction du type de transfert effectué.

### f/ Les CRC (Codes cycliques redondants)

Les CRC sont utilisés pour protéger tous les champs non PID dans un paquet "floating gate" ou dans un paquet de données. Le CRC des paquets "floating gate" couvrent les champs ADDR, ENDP des types IN, SETUP et OUT. Le polynôme générateur est le suivant:  $G(x)=X^5+X^2+1$

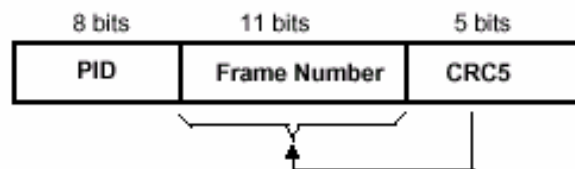
Le CRC couvrant les données repose sur un polynôme de 16 bits:  $G(x)=X^{16}+X^{15}+X^2+1$

### g/ Les paquets "floating gate"



Un paquet "floating gate" est composé d'un PID, spécifiant un paquet IN, OUT ou SETUP et un champ ADDR et ENDP. Pour les transactions OUT et SETUP, les champs d'adresse et de point final identifient le point final qui va recevoir les paquets suivants. Pour les transactions IN, ces champs identifient quel point final transmettra un paquet de données. L'hôte est le seul élément dans la topologie USB responsable de l'envoi d'un tel paquet.

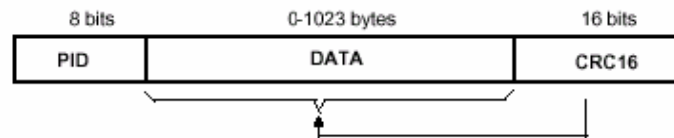
### h/ Les paquets de début de trame



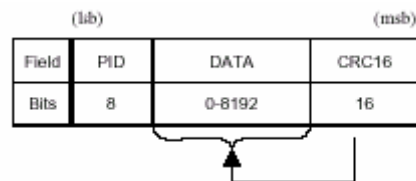
Un paquet SOF (Start Of Frame) est constitué d'un champ PID identifiant le paquet suivi d'un champ de numéro de trame comme illustré ci-dessus.

## i/ Les paquets de données

Un paquet de données est constitué d'un champ PID, d'un champ de données et d'un champ CRC. Il y a deux types de paquets de données, identifiés par un PID différent: DATA0 et DATA1



Les données doivent toujours être envoyées dans un nombre complet de bytes. Pour la norme USB 2, le nombre de données transmises peut être plus important.



## j/ Les paquets de poignées de mains

Un paquet de poignée de main est uniquement constitué d'un PID. Ce paquet est utilisé pour uniquement reporter l'état d'une transaction de données et peut retourner des valeurs indiquant la réception correcte des données. Seulement des types de transactions qui supportent le contrôle de flux peuvent supporter un retour de poignée de mains.

Il y a trois types de paquets de poignée de mains:

**ACK:** indique que le paquet de données est reçu sans bits de remplissage ou sans erreur de CRC sur la partie de données et que le PID a été reçu correctement. ACK peut également être transmis lorsque les bits de séquence coïncident et que le récepteur peut accepter la donnée ACK peut être transmis par l'hôte pour une transaction de type IN ou par une fonction dans une transaction de type OUT.

**NACK:** indique qu'une fonction est incapable d'accepter la donnée de l'hôte ou que la fonction n'a pas de données à transmettre vers l'hôte (IN). NAK ne peut être retourné que par des fonctions dans une phase de données d'une transaction IN ou dans une phase de transaction OUT et un hôte ne peut transmettre de tel paquet. NAK est utilisé pour le contrôle de flux destiné pour indiquer que la fonction est incapable temporairement de transmettre ou de recevoir des données mais pourra éventuellement le faire sans l'intervention de l'hôte.

**STALL:** est retourné par une fonction en réponse à un paquet "floating gate" IN ou après une phase de données de type OUT. STALL indique qu'une fonction est incapable de transmettre ou de recevoir des données et que cette condition nécessite une intervention de l'hôte pour retirer la cale(stall). Aussitôt que le point final d'une fonction est calé, la fonction doit continuer à retourner STALL jusqu'à ce que la condition causant le calage soit effacée par une intervention de l'hôte. L'hôte n'est pas autorisé à retourner un STALL quelle que soit la condition.

## 10. En ce moment...

### **Clé USB LaCie (orange) 8Go** USB 2.0



Prix : 99€

Pas plus grande qu'une carte de crédit, la très conviviale clé USB LaCie est le dispositif de stockage le plus petit que LaCie et jamais créé. Et pourtant, ses capacités sont réellement impressionnantes (pour un prix par Go inférieur aux clés Flash). Grâce au branchement Plug & Play Hi-Speed USB 2.0, vous pouvez l'utiliser partout sans alimentation électrique ni pilote. En métal robuste, compacte et ultraplate (6 mm d'épaisseur), vous pouvez la glisser discrètement dans votre poche et l'emporter partout avec vous. Pour le stockage de fichiers, photos, titres audio ou clips vidéo et le transfert entre ordinateurs afin d'en faire profiter vos amis, votre famille et vos collègues.


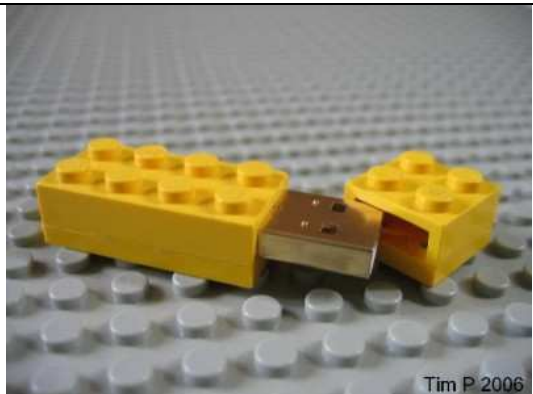

### **Sony Micro Vault PRO**

Sony améliore sa gamme de clé USB en annonçant sa nouvelle Micro Vault PRO de 8 Go. Comme son nom l'indique cette clé possède une capacité de stockage de 8 Go. Contrairement aux autres modèles Micro Vault, la Micro Vault PRO n'exploite pas de la mémoire Flash mais bien un disque dur miniature d'un pouce.



Prix : 150€

## 11. Dérivés

<p><b>Clé USB Lampe</b></p> <p>Ici la source lumineuse est fournie par une LED dont la durée de vie est supérieure à celle d'une ampoule classique...</p> <p>prix :60€</p>	
<p><b>Toutes les formes...</b></p>	 <p>Tim P 2006</p>
<p><b>Une Radio FM cochon</b></p> <p>Un haut-parleur dans le groin, la queue pour régler le volume, une oreille pour scanner les fréquences et l'autre pour revenir à la fréquence la plus basse.</p> <p>Prix : 20€</p>	

## 12. Critères de choix

Il n'y a pas vraiment de critères définis pour bien choisir ça clé USB. Le choix d'une clé va se faire sur vos propres critères. La taille par exemple, bien que certaines clés aient la même capacité mémoire, elle peuvent avoir une taille différentes et peut être aussi des caractéristiques propres à leurs constructeurs (capteur biométrique, fonction mp3, etc.). C'est donc à vous de choisir en fonction des utilisations que vous allez pouvoir en faire.

Il est peut être aussi important, si vous travaillé encore sur un système d'exploitation précédent Windows 2000, de regarder si la clé est fournie avec des drivers. Car avant Windows 2000, les clés USB n'étaient pas reconnue automatiquement par le système.

## 13. Conclusion

Les clés USB sont aujourd'hui très bon marché et devraient encore diminuer dans l'avenir. Si le besoin d'un stockage amovible se fait donc ressentir, se serait dommage de sans privé. De plus, le passage à la norme USB2.0 étant maintenant effectué depuis un bon moment et celui-ci offrant des débits très rapide, les clés remplaceront vos anciens périphériques de stockage amovible avec brio.

## 14. Bibliographie

Wikipédia :

[http://fr.wikipedia.org/wiki/Clef\\_USB](http://fr.wikipedia.org/wiki/Clef_USB)

Cours Mr Wilfart :

<http://hesit.be/files/info/2/1093871060-micro%20ordi%20cours%202004.pdf>

Beyond logic:

<http://www.beyondlogic.org/usb/usbdevdvs.htm>

LaCie et Sony sur les sites respectifs :

[www.lacie.com](http://www.lacie.com)

[www.sony.com](http://www.sony.com)