



Le Port USB

USB : Universal Serial Bus

L'Universal Serial Bus (USB) est un bus informatique (série synchrone), il sert à brancher des périphériques informatiques à un hôte (ordinateur).

1)Historique

L'USB a été conçu au milieu des années 1990 afin de remplacer les nombreux ports externes d'ordinateur lents et incompatibles. Les différentes normes existantes sont :

- **USB 1.0** en 1994
- **USB 1.1** en 1998 (Low speed 1.5 Mbits/s, Full speed 12 Mbits/s).
- **USB 2.0** en 2000 (High speed 480 Mbits/s).
- **USB 3.0** en projet, pour 2009 ou 2010(Super speed 4.8 Gbit/s soit 600 Mo/s).

2)Usage

Le port USB est le moyen de communication le plus utilisé pour faire communiquer un appareil avec l'ordinateur. La norme USB permet tous types de périphériques :

Clés USB, lecteurs multimédias (mp3, mp4), clés TNT, WIFI, BLUETOOTH et infrarouge, appareils photo numériques, Webcam, imprimante, scanner, souris, clavier…

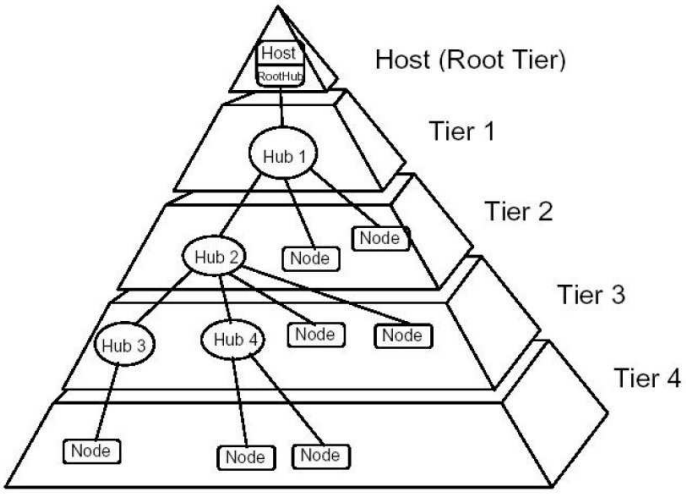
3)Spécifications techniques

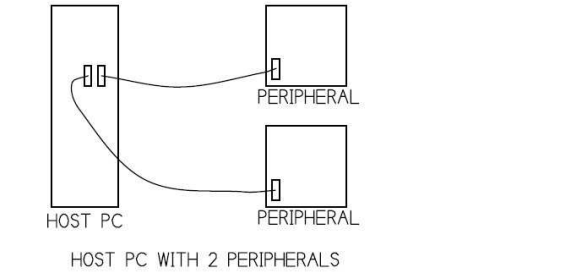
- Auto détection & configuration (Plug&Play) par le système d'exploitation.
- *Le bus supporte les branchements et débranchements à chaud ("hot plug", sans avoir besoin de redémarrer l'ordinateur)*
- Fournit l'alimentation électrique des périphériques, dans la limite des 500 mA et 5V.
- *L'USB supporte le branchement simultané de 127 périphériques, en série.*
- *Topologie en arbre dont la racine est l'hôte (le PC ou le Mac).*
- *Les périphériques inutilisés sont automatiquement mis en veille.*
- Supporte 3 vitesses de transmissions qui peuvent cohabiter.

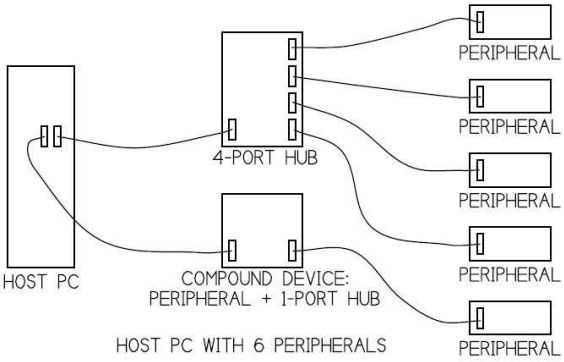
- **Basse vitesse** 1,5 Mb/s *pour les appareils interactifs tels que clavier, souris, joystick,...*
- **Pleine vitesse** 12 Mb/s pour des appareils à gros débits tels que imprimantes, scanners, disques durs,modem, téléphone, ...
- **Haute vitesse** 480 Mb/s pour des appareils multimédia à très gros débits tels que caméscope disques durs, graveurs. Cette vitesse concurrence le bus FireWire.

Half duplex – la transmission de données ne peut se faire seulement que dans une direction a la fois.

La topologie est une topologie « étoile série » ou encore « tiered star » qui tolère jusqu'à 5 niveaux de concentrateurs. La spécification limite le nombre de périphérique à 127.
Un dispositif conforme à la norme 2.0 peut-être relié à un HUB 1.1 mais dans ce cas le trafic sera celui défini par la norme 1.1.



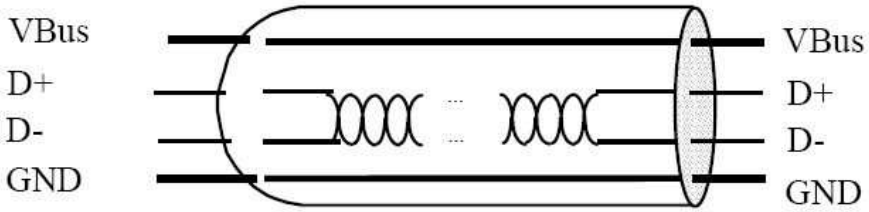




4) Câblage

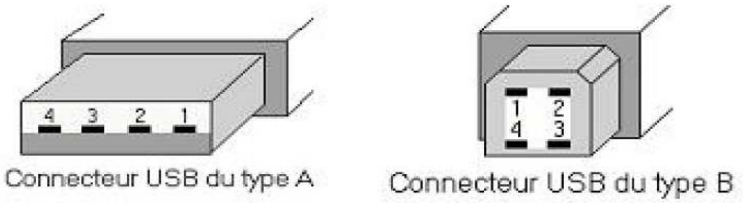
4.1) Le câble USB

Le câble transporte deux paires de fils : La paire de signal destinée au transfert de données D+ et D- et une seconde paire qui peut être utilisée pour la télé alimentation GND et Vcc.





4.2)Couleurs des conducteurs et brochage prise male :

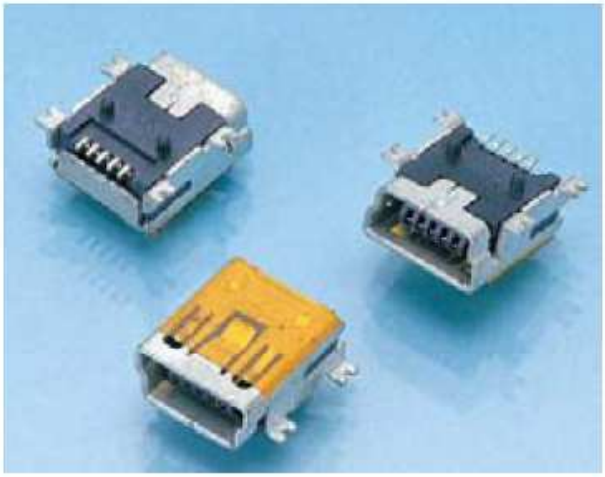


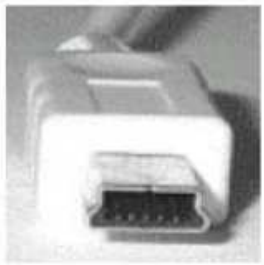
Brochage des connecteurs USB de type A et B

1. Alimentation +5V (VBUS) 500 mA maximum : rouge
2. Données (D-) : blanc
3. Données (D+) : vert
4. Masse (GND) : noir

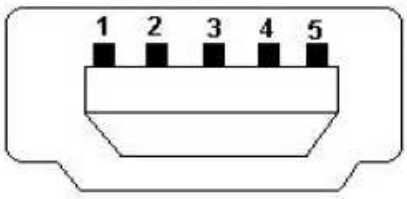
4.3) Connecteur USB type mini

Il existe une autre série de connecteurs appelée USB "mini A" et "mini B". Ce sont les connecteurs que l'on retrouve sur les appareils photo et les lecteurs MP3.
Si on regarde en détail, on s'aperçoit que ces connecteurs disposent non pas de 4 connexions mais de 5. Soit celles du bus USB, plus une connexion supplémentaire pour la configuration du périphérique, utile lors de l'utilisation OTG (On The Go).
L'OTG est une évolution de la norme USB, qui permet la connexion entre deux appareils **sans ordinateur**, par exemple entre un appareil photo et une imprimante. L'USB ayant besoin d'un hôte (host) et un host unique, la fonction de l'OTG est de donner ce rôle d'hôte à un des deux périphériques. Le rôle host/device est fixé par le sens du câble, en positionnant cette broche supplémentaire à +Vusb ou à la masse (GND) suivant les extrémités. Cet échange est normalisé par la norme HNP (Host Negociation Protocol).











4.4) Les normes électrique :

4.4.1) L'alimentation :

Un des avantages de l'USB réside dans ces appareils alimentés par le Bus.

La tension d'alimentation pour chaque appareil est entre 4.4V et 5.25V.

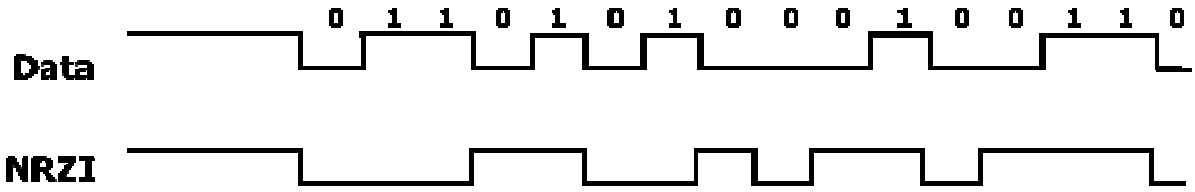
On définit **une unité** de puissance correspondant à **100mA** (500mW plus exactement). **Par défaut** un appareil est en basse puissance et consomme **une unité** au maximum jusqu'à sa configuration; **par programmation** il peut ensuite monter jusqu'à **5 unités(**500mA).
Les appareils autoalimentés peuvent consommer une unité permettant à l'interface de continuer à fonctionner sans alimentation externe. Le contrôleur de hub informe l'hôte des conditions d'alimentation.
Un appareil en état de veille ne peux pas consommer plus de 0,5mA., s'il est configuré comme une source de réveil il peut aller jusqu'à 2,5mA.

4.4.2) Transmission de données

L'USB utilise une paire de transmission différentielle pour les données (lignes de données D+ et D-).

Celle-ci est codée en utilisant le **NRZI** (*Non Return to Zero Inverted*) et est garnie de bits pour assurer les transitions adéquates dans le flot de données.

Un "1" est représenté par l'absence de changement d'état et un "0" par un changement d'état.



La période est la durée de transmission d'un bit selon la vitesse utilisée.

Si le HUB détecte qu'une ligne de donnée avoisine les 90% de Vcc il en conclura qu'un composant USB est connecté ainsi que des informations sur la vitesse du composant, Low ou Full Speed.

4.4.3) Identification de la vitesse

Un appareil USB doit indiquer sa vitesse en mettant soit D+ ou D- à 3,3V. Pour faire le choix entre la version USB Low Speed et Full Speed, il suffit de placer une résistance de tirage sur l'interface d'entrée (Fonction ou Hub). Cette résistance de tirage (Pull-Up de 1.5Kohm) est placée soit sur D- dans le cas du Low Speed ou sur D+ dans le cas du Full Speed. Sans résistance de rappel, l'USB suppose qu'il n'y a rien de connecté au BUS.

Par contre côté sortie du HUB il y a une résistance sur chacune des lignes D+ et D- (Pull Down de 15KOhm). Les résistances Pull down étant de 15kOhm et les résistances de Pull up de 1.5kOhm (en fonction de la vitesse choisie) il y aura toujours une ligne de donnée qui sera à 90% de Vcc.

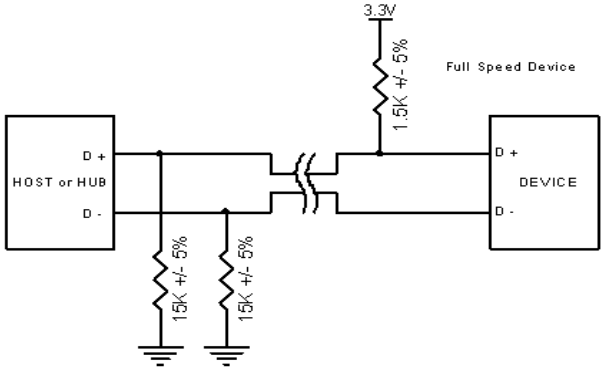


Figure 2 : Appareil **pleine vitesse** avec résistance de rappel état haut branché sur D+

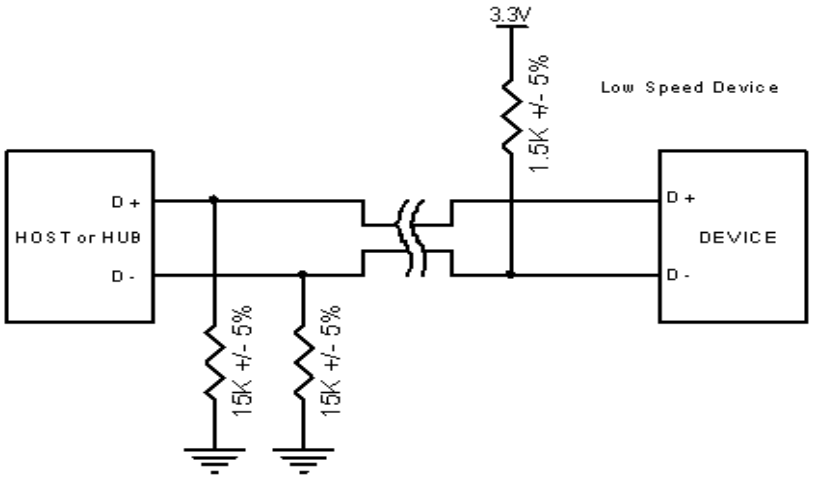


Figure 3 : Appareil **basse vitesse** avec résistance de rappel état haut branché sur D-

Vous noterez que nous n'avons pas inclus d'identification de vitesse pour le mode haute vitesse. Les appareils haute vitesse démarreront dès qu'ils seront connectés en tant qu'appareils pleine vitesse (1,5 kOhms à 3,3V). Une fois qu'il sera attaché il émettra un Chirp à haute vitesse pendant la réinitialisation et établira une connexion à grande vitesse si le Hub le supporte. Si l'appareil fonctionne en mode haute vitesse, alors la résistance de rappel est retirée pour équilibrer la ligne.

Après détection du périphérique et de sa vitesse, l'énumération peut commencer, c'est parti pour la grande aventure de l'USB.

5) L'Énumération

L'énumération est une étape qui permet d'identifier et de configurer le périphérique qui vient juste d'être branché sur le bus USB. Pendant l'énumération le périphérique fournit à l'hôte (un ordinateur en principe) une suite de descripteurs qui permettent son identification.

L'hôte attribuera à l'appareil une adresse et validera une configuration lui permettant de transférer des données sur le bus.

Une énumération ordinaire sous Windows implique les étapes suivantes :

- L'hôte ou Hub détecte la connexion d'un nouvel appareil via les résistances de rappel de l'appareil reliées sur les 2 fils de données. L'hôte attend au moins 100ms, le temps que la prise soit insérée complètement et que l'alimentation de l'appareil soit stabilisée.
- L'hôte émet un " reset " mettant l'appareil dans l'état par défaut. L'appareil peut maintenant répondre à l'adresse zéro par défaut.
- L'hôte (sous MS Windows) demande les 64 premiers octets du **descripteur d'appareil**.
- Après avoir reçu les 8 premiers octets du **descripteur d'appareil**, l'hôte émet immédiatement un autre reset sur le bus.
- L'hôte émet maintenant une commande *SetAdress*, mettant l'appareil dans l'état adressable.
- L'hôte demande la totalité des 18 octets du **descripteur d'appareil**.
- Puis il demande les 9 octets du **descripteur de configuration** pour déterminer la taille totale.
- L'hôte demande les 255 octets du **descripteur de configuration.**
- L'hôte demande l'un des **descripteurs de chaînes** s'ils étaient indiqués.

Ensuite " Windows " demandera un driver (pilote logiciel) pour votre appareil ou chargera directement les drivers si le périphérique utilise une classe générique

5.1) Les descripteurs USB

En effet lors de la connexion (ou déconnexion) il y a une phase de détection et une phase d'identification effectuée par l'hôte qu'on appelle énumération. Lors de cette phase, le périphérique fournit à l'hôte une suite de descripteurs qui permettent son identification complète.
Lors de cette phase d'énumération, on assigne une adresse unique (Unique ID) au périphérique, on charge le driver correspondant et on positionne le composant dans la configuration qui lui à été donné par les descripteurs.

Il n'est pas indispensable de connaître parfaitement le processus d'énumération et le système de descripteurs pour pourvoir faire fonctionner un composant USB mais il est bon d'en connaître les grandes lignes pour pouvoir, au besoin, changer les descripteurs ou bien par simple culture générale.

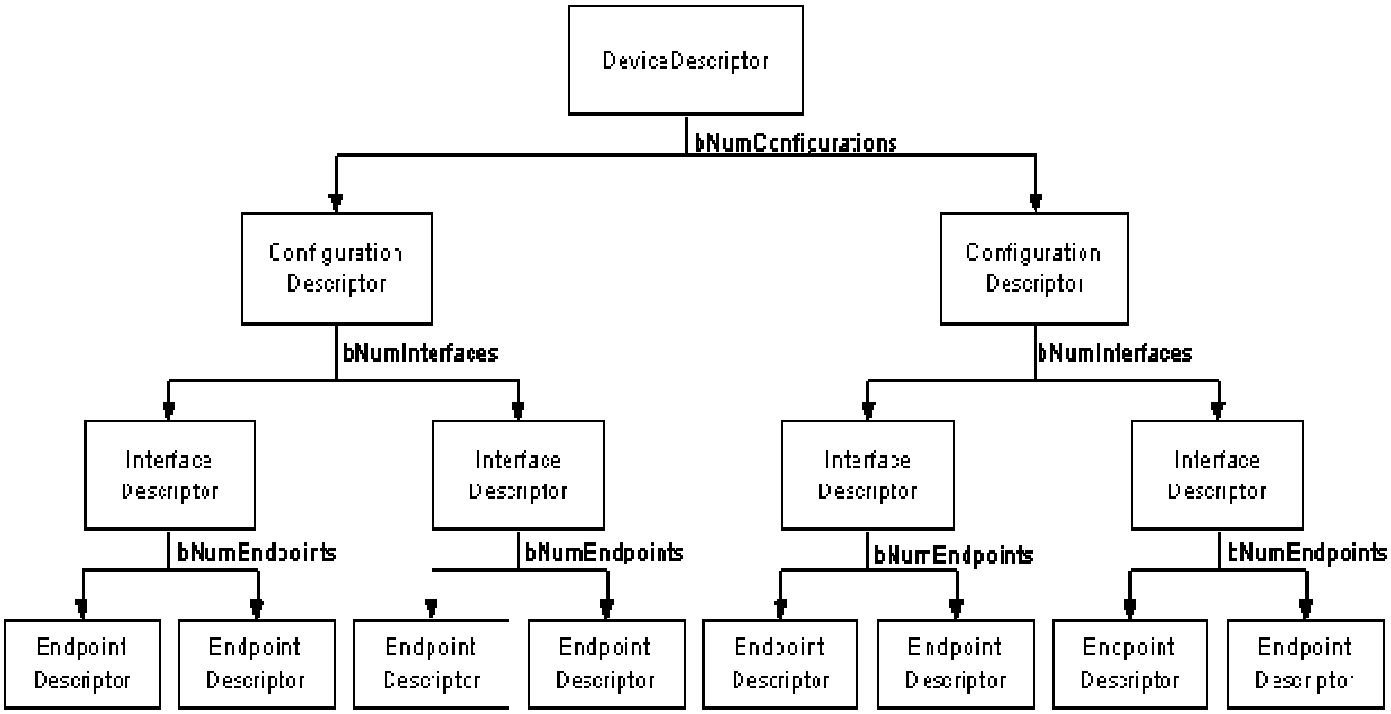
Notons également pour ceux qui ne le savent pas encore, que cette phase d'énumération est totalement transparente et automatique pour l'utilisateur.

Les descripteurs " standards " sont regroupés en 4 catégories :

- Descripteur d'appareil
- Descripteur de configuration
- Descripteurs d'Interfaces
- Descripteurs de Terminaisons

Voici une liste des autres descripteurs possibles :

- Les Descripteurs HID (Human Interface Device) qui sont les plus utilisés. En fait, ils regroupent tous les appareils qu'utilisent directement les personnes c'est à dire les souris, les claviers, les Gamepads, les écrans, etc.
- Les descripteurs MSD (périphérique de stockage de masse).



5.1.1) Descripteur d'appareil

Ce type de descripteur donne les informations générales. C'est le premier descripteur que vient lire le host. Comme dit précédemment il y a une très grande diversité de composants USB, il y a des propriétés communes à chaque dispositif d'USB comme par exemple le numéro de spécification USB qui est présent dans toutes les configurations, de même pour les numéros d'identifiant de produit et de vendeur. (Product ID et Vendor ID).

Un Ordinateur peut, uniquement avec ces informations contenues dans le « Device Descriptors », reconnaître un composant USB. il donne également des renseignements sur le *pipe* de communication par défaut qui est utilisé pour la configuration. Voici un exemple des différents descripteurs faisant partie de la catégorie « Device Descripteur». Dans le champ description il y a la signification de chaque descripteur, cela

Offset	Nom	Taille	Description
0	Blength	1	Taille du descripteur, toujours égale à 18 pour un device descriptor.
1	BdescriptorType	1	Type du descripteur, toujours 01h pour un device descriptor.
2	BcdUSB	2	C'est la version de la spécification USB (codée en DCB) exemple : Version 1.0 -> 0100h Version 1.1 -> 0110h Version 2.0 -> 0200h
4	BDeviceClass	1	Si le composant appartient a une classe, ce champ renseigne sur cette classe. Si il n'appartient à aucune classe le champ doit être mis à 0. L'adresse FFh est réservée et signifie que la classe est spécifique au vendeur
5	BdeviceSubClass	1	Si un composant appartient déjà à une classe, ce champ permettra de restreindre encore une fois en définissant une sous classe. Si bDeviceClass = 0, BdeviceSubClass doit forcément être égal à 0 aussi.
6	BdeviceProtocol	1	Ce champ renseigne sur le protocole choisi par la classe ou la sous classe. C'est dans ce champ qu'un Hub USB 2.0 dira s'il supporte à la fois le Full et le High Speed.
7	BmaxPacketSize0	1	Taille maximum des paquets de l'Endpoint 0. Low Speed : 8 Full Speed : 8 – 16 – 32 – 64 High Speed : 64
8	IdVendor	2	Ce champ renseigne sur le Vendor ID. Un chapitre spécial à été réservé pour expliquer son utilité.
10	IdProduct	2	Ce champ renseigne sur le Product ID. Un chapitre spécial à été réservé pour expliquer son utilité.
12	BcdDevice	2	(Device release number) Codé en BCD, ce champ, optionnel, renseigne sur le driver à charger par exemple.
14	IManufacturer	1	C'est un index qui pointe sur un String qui donne le nom du Constructeur. Ce champ est optionnel. 0 si inutilisé.
15	Iproduct	1	C'est un index qui pointe sur un String qui donne des informations sur le produit. Ce champ est optionnel. 0 si inutilisé.
16	ISerialNumber	1	C'est un index qui pointe sur un String qui donne des informations sur le numéro de série du composant. Ce champ est optionnel. 0 si inutilisé. Ce champ peut être intéressant si l'on utilise plusieurs périphériques identique possédant les même PID et PID.
17	BnumConfiguration	1	Nombre de configuration que supporte le composant.

5.1.2) Descripteur de configuration

Un descripteur de configuration renseigne sur les différents états dans lequel peut se trouver le composant USB.
Ces descripteurs de configuration définissent par exemple l'origine de l'alimentation, elle peut soit provenir d'une alimentation extérieure, soit directement du bus USB. Il se peut qu'il y ait deux configurations différentes selon le type d'alimentation. En effet,si le système USB est directement alimenté par le bus USB il se peut en raison des conditions de puissance d'énergie, que le dispositif pourrait invalider quelques paramètres.
La numérotation des configurations commence à 1, la configuration 0 est une configuration réservée. Si le périphérique est dans cette configuration là, on dit qu'il est pas configuré « unconfigurated » et ne peut pas communiquer avec la host tant qu'il n'est pas sorti de cet état.
Voici un exemple des différents descripteurs faisant partie de la catégorie « Configurationdescriptor »

Offset	Name	Size	Description
0	BLengt	1	Taille du descripteur, toujours égale à 9 pour un configuration descriptor.
1	BDescriptorType	1	Type du descripteur, toujours 02h pour un configuration descriptor.
2	WTotalLength	2	Longueur total en bytes de toutes les données dans cette configuration. Cette longueur inclus les descripteurs d'interfaces, HIDs, reports et les EndPoint.
4	BNumInterfaces	1	Nombre d'interfaces supportée dans cette configuration.
5	BconfigurationValue	1	Identifiant de Set_Configuration et de Get_Configuration.
6	Iconfiguration	1	C'est un index qui pointe sur un String qui décrit cette configuration. Ce champs est optionnel. 0 si
			inutilisé.
7	BmAttributes	1	Précise si le bus comment est alimenté le périphérique. Self Power ou Bus. Précise également si le périphérique supporte le remote wakeup.
8	MaxPower	1	Consommation du composant USB (en mA /2)

5.1.3) Descripteur d'interface

Une interface peut être considérée comme un ensemble d' «Endpoint ». Ce mot n'a pas de traduction très réaliste mis à part « point final ».

Un Endpoint est en quelque sorte l'extrémité d'un « pipe ». Un « pipe » est une sorte de tuyau par lequel transitent les données via le host. Un descripteur d'interface communique une information unique à tous ses Endpoints.

Si une configuration est choisie, toutes ses interfaces sont activent et par conséquent, aucun Endpoint ne peut être relié à des interfaces différentes sous la même configuration.

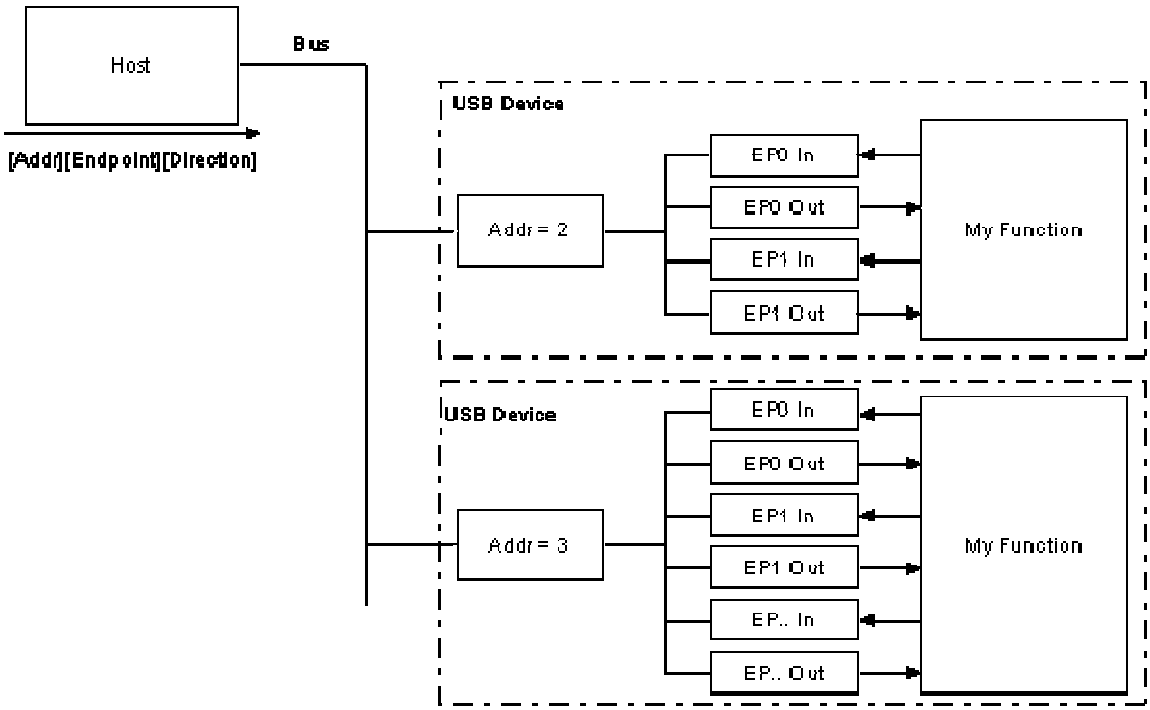
Les interfaces sous différentes configurations peuvent partager des Endpoints.

Voici un exemple des différents descripteurs faisant partie de la catégorie « Interface

Offset	Name	Size	Description
0	Blength	**1**	Taille du descripteur, toujours égale à 9 pour un interface descriptor.
1	BdescriptorType	**1**	Type du descripteur, toujours 04h pour un interface descriptor.
2	BinterfaceNumber	**1**	Dans un device, une configuration possède plusieurs interfaces qui peuvent être actives en même temps. Ce champs spécifie pour pouvoir reconnaître les interfaces de la même configuration.
3	BAlternateSetting	**1**	Cette valeur sélectionne par contre une interface bien précise.
4	BnumEndpoints	**1**	Nombre d'EndPoint supporté par l'interface. L'EndPoint 0 n'est pas compter. Si c'est le seul supporté BnumEndpoints = 0.
5	BinterfaceClass	**1**	Si le composant appartient à une classe, ce champs renseigne sur cette classe. L'adresse FFh est réservée et signifie que la classe est spécifique au vendeur La class HID a le code 03h. L'adresse 00h est réservée.
6	BInterfaceSubClass	**1**	Si un composant appartient déjà à une classe, ce champs permettra de restreindre encore une fois en définissant une sous classe. Si BInterfaceClass = 0, bInterfaceSubClass doit forcément être égal à 0 aussi. Si BinterfaceClass à une adresse entre 01h et FEh, bInterfaceSubClass dois être un code
			définit par la spécification USB. Une valeur FFh signifie que c'est une classe spécifique à un vendeur bien précis.
7	BinterfaceProtocol	**1**	Ce champs renseigne sur le protocole choisi par la classe ou la sous classe.
8	Interface	**1**	C'est un index qui pointe sur un String qui décrit cette interface.

5.1.4) Descripteur de terminaison (Endpoints)

Un descripteur d'Endpoint indique la direction du transfert (IN ou OUT), ses types de transfert (ISOCHRONOUS, BULK, INTERRUPTION ou CONTROL), ainsi que d'autres informations qui sont regroupé dans le tableau suivant.
En fait, l'ordinateur « le host » communique uniquement avec ces Endpoints. Tous les transferts de paquet de données transitant sur le bus proviennent d'un Endpoint ou sont envoyés à un Endpoint. Généralement les Endpoints correspondent aux Entrées-Sorties ou au registre du dispositif USB.
Le nombre maximum d'Endpoint est différent selon que l'on utilise de l'USB Low Speed ou de l'USB High speed. Un dispositif USB High Speed peut supporter jusqu'à 15 Endpoints tandis qu'un dispositif USB Low Speed ne peut que supporter 3 Endpoints. Nous verrons par la suite les différentes contraintes sur l'utilisation des ces Endpoints pour un dispositif USB Low Speed.
Par contre, il est possible que deux Endpoints partagent le même numéro
Un Endpoint transitant les données dans un sens (IN par exemple) et l'autre pour les données transitant dans l'autre sens (OUT). Dans ce cas, il est nécessaire de définir deux descripteurs différents. L'Endpoint 0 est un Endpoint particulier, c'est le seul qui est bi-directionnel et présent dans tous les dispositifs. Il est utilisé par le host pour contrôler le système.
Il n'est cependant pas nécessaire de lui affecter un descripteur particulier. L'EndPoint 0 fonctionne toujours en mode *Control*. Voici un exemple des différents descripteurs faisant partie de la catégorie « Endpoint descriptor»

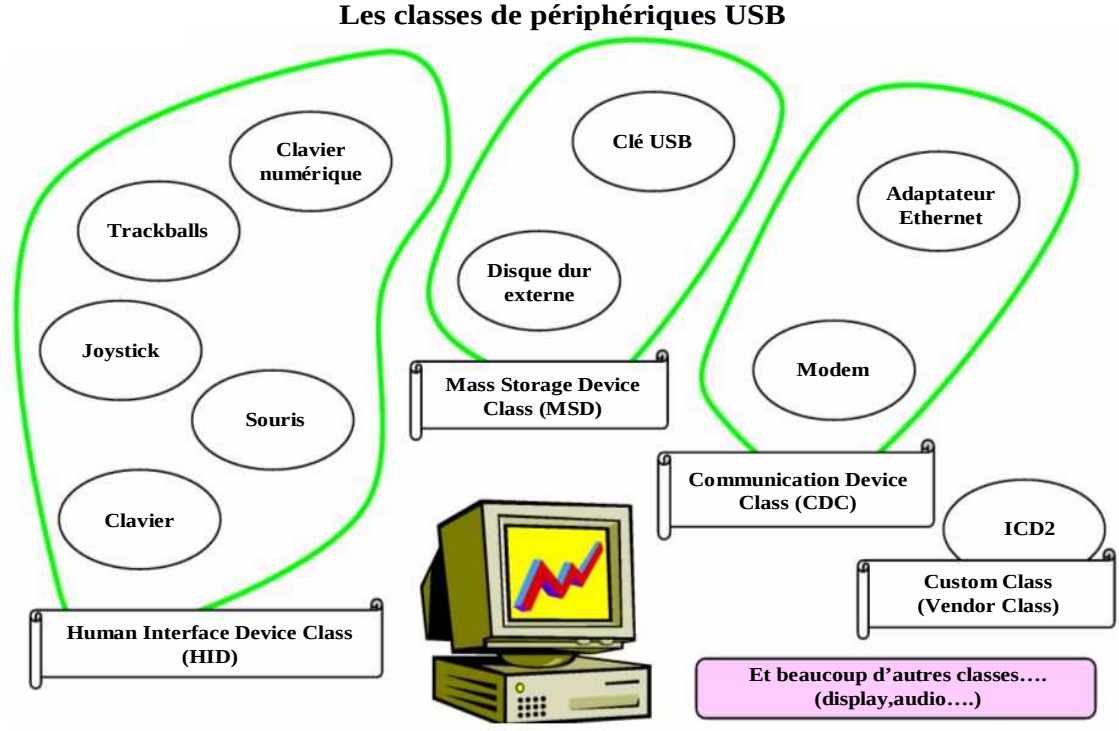


Offset	Name	Size	Description
0	BLength	1	Taille du descripteur, toujours égale à 7 pour un Endpoint descriptor.
1	BdescriptorType	1	Type du descripteur, toujours 05h pour un EndPoint descriptor.
2	BendpointAddress	1	Renseigne sur le numéro de l'EndPoint et la direction.
3	BmAttributes	1	Renseigne sur le type de transport utilisés. 00 = Contrôle 01 = Isochrone 10 = Bulk 11 = Interrupt
4	WMaxPacketSize	2	Spécifie la taille maximale des paquets Bits 0 à 9 : 0 à 1023 Bits 10 et 11 : indique le nombre de transactions supplémentaire par microframe (uniquement pour le High Speed) Bits 12 à 15 : réservé.
6	Binterval	1	Indique le temps maximal (en ms) entre le polling des EndPoints.

6) Classes de périphériques

Les périphériques sont divisés en plusieurs classes. Chaque classe définit un comportement commun et des protocoles pour les périphériques qui utilisent les mêmes fonctionnalités.

Les classes de périphériques USB



7) Le Protocole USB

Le bus USB est un bus fonctionnant sur la hiérarchie, commandé par un hôte unique. L'hôte utilise un protocole **Maître/Esclave à jeton** (le jeton matérialisant le droit de transmettre) pour communiquer avec les périphériques USB. Cela signifie que c'est l'hôte qui décide du transfert des données et que les différents périphériques ne peuvent pas établir de connexion entre eux tant que le maître n'a pas donné l'autorisation.
Le protocole USB est, comme tous les autres protocoles un protocole à encapsulation. Mais avant de s'intéresser à la trame proprement dite, voyant ce qu'il en est du protocole. Le client driver communique les demandes de transfert des applications via des IRP (I/O Packet). Puis, l'USB driver traduit chaque transfert en une suite de transactions. Ensuite l'USB Host Controller driver regroupe les transactions en trames et finalement l'USB Host Controller traduit les transactions en paquets et enchaîne les trames.

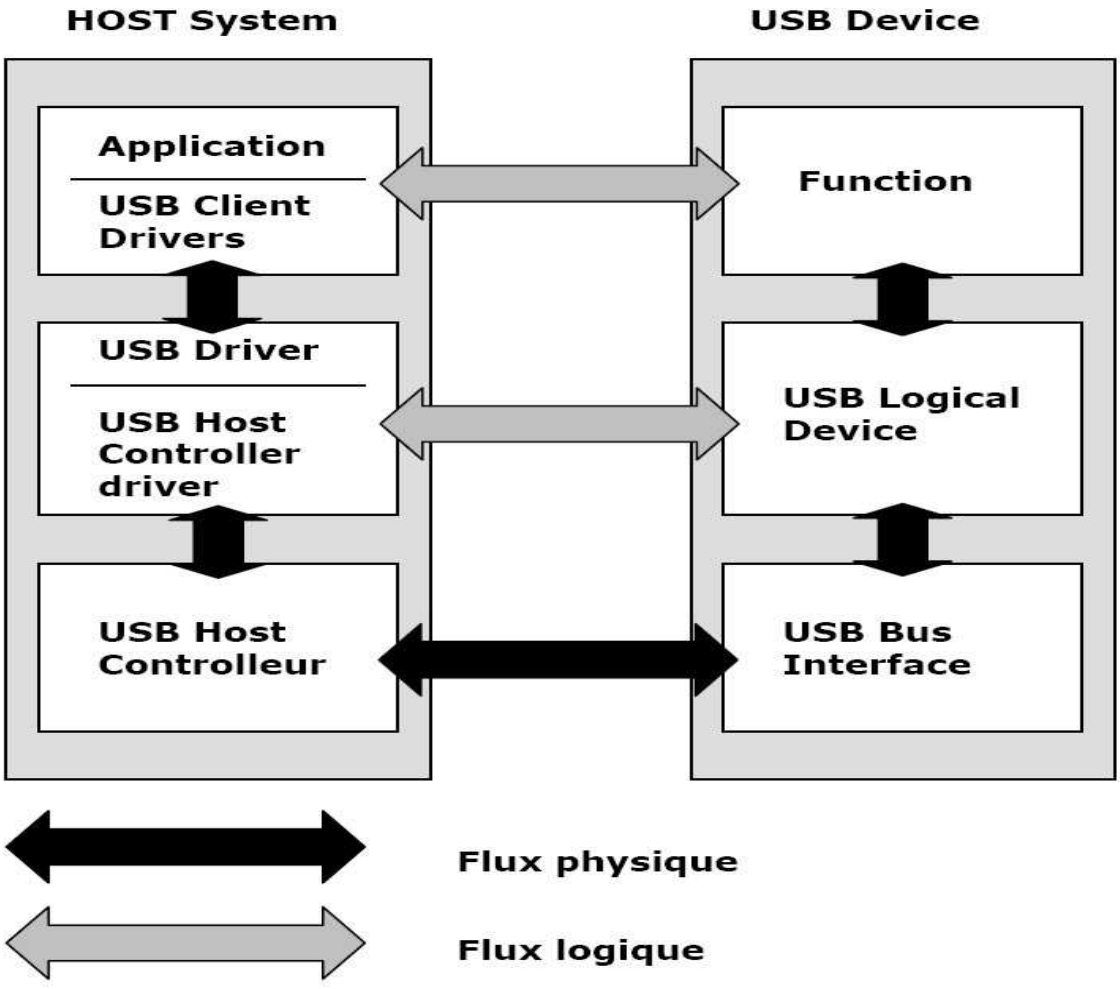
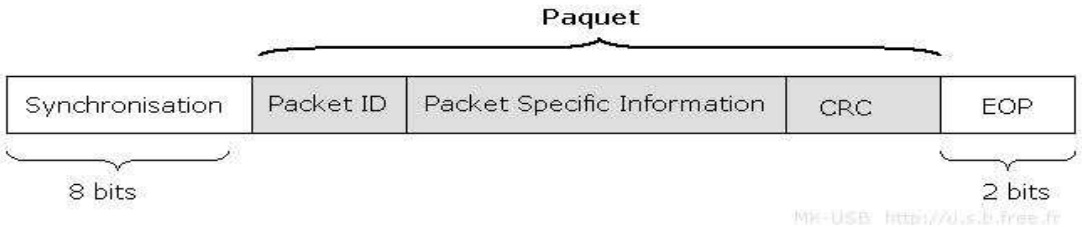


Figure 6 : *Protocole USB*

7.1) Type de paquet USB

Contrairement à la liaison série RS232 et des interfaces séries similaires où le format des données envoyées n'est pas défini, l'USB lui est composé de plusieurs couches de protocoles bien définis. La plupart des circuits intégrés USB s'occuperont de la couche inférieure, la rendant ainsi presque invisible au regard du concepteur final. Mais il est tout de même intéressant d'en connaître les grandes lignes. L'USB a quatre types différents de paquet :
- Token (En-tête)
- SOF (Start of frame)
- Data (Optionnel)
- Acknowledge (Handshake)

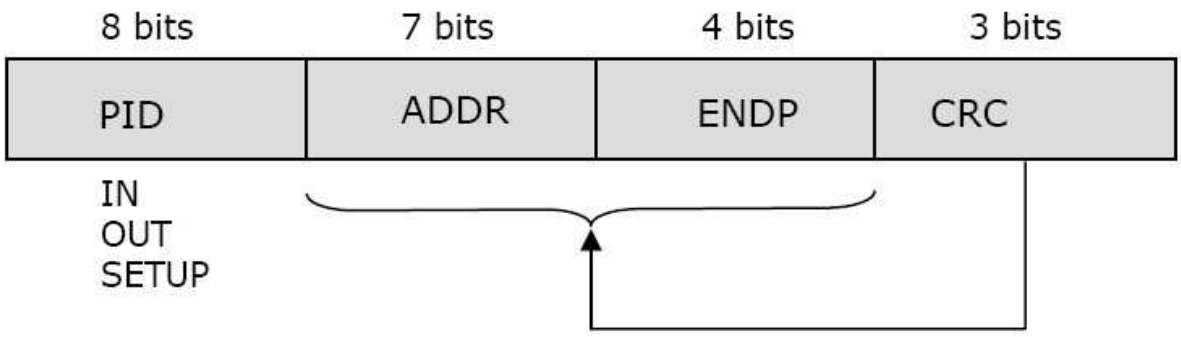
Les **paquets jetons** indiquent le type de la transaction qui va suivre et a pour but de transporter l'adresse USB et le sens du transfert. Les **paquets de données** contiennent les données utiles. Les **paquets "Handshake"** sont utilisés pour valider les données ou rapporter les erreurs. Et les **paquets début de trame (SOF)** indiquent le commencement d'une nouvelle trame.
Les différents paquets ont une structure bien définie que voici:



Packet ID : identification du paquet.
Packet Specific information : information spécifique du paquet.
CRC : contrôle à redondance cyclique.
EOP : end of packet.
La structure des paquets étant identique il faut préciser que le format est différent selon la nature du paquet. Voici ci dessous les différents formats de paquet.

7.1.1) Les paquets jetons

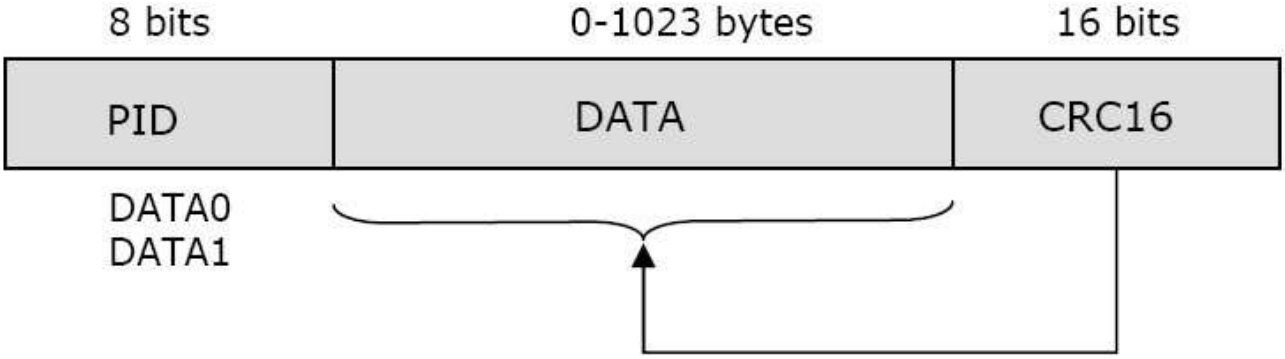
• Il y a 3 sortes de paquets Jetons,
o **In** : Informe l'appareil USB que l'hôte veut lire des informations.
o **Out** : Informe l'appareil USB.que l'hôte veut envoyer des informations.
o **Setup** : Utilisé pour commencer les transferts de commande.



Format du paquet Token

7.1.2) Les paquets Data

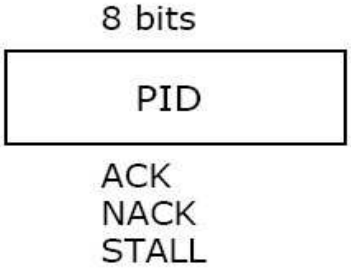
Les paquets de *Data* ont le format suivant :



Format du paquet Data

7.1.3) Les paquets « Handshake »

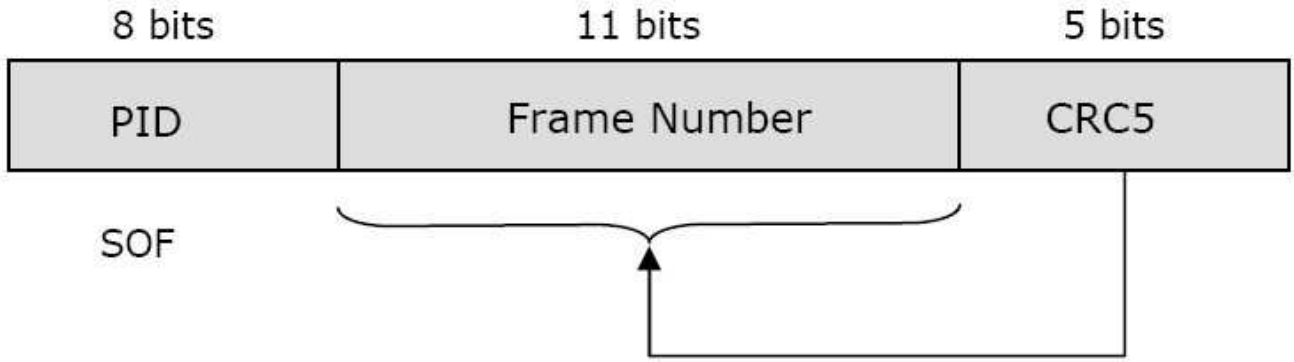
• Il y a 3 sortes de paquets *Handshake* qui font simplement partie du PID.
ACK - validant que le paquet a été reçu correctement.
NAK - Indique que l'appareil ne peut temporairement ni envoyer ou recevoir des données. Aussi utilisé pendant les transactions d'interruptions pour avertir l'hôte qu'il n'a pas de données à envoyer.
STALL (Bloqué) - L'appareil se retrouve dans un état qui va exiger l'intervention de l'hôte.



Format du paquet Handshake

7.1.4) Les paquets SOF (start of frame)

Les paquets *SOF* ont le format suivant :



Format du paquet SOF

PID

PID signifie Paquet ID. Ce champ est utilisé pour identifier le type de paquet qui est envoyé. Le tableau suivant montre les valeurs possibles.

Groupe	Valeur PID	Identificateur Paquet
Token	0001	OUT Token
	1001	IN Token
	0101	SOF Token
	1101	SETUP Token
Data	0011	DATA0
	1011	DATA1
	0111	DATA2
	1111	MDATA
Handshake	0010	ACK Handshake
	1010	NAK Handshake
	1110	STALL Handshake
	0110	NYET (No response Yet)
Special	0000	PREambule
	1100	ERR
	1000	Split
	0100	Ping

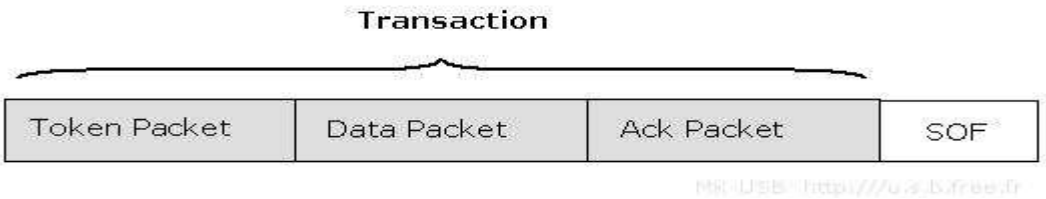
Tableau 1 : *Désignation des PIDs*

7.2) Les transactions

Les transactions USB se font par l'intermédiaire de l'émission de plusieurs paquets dont le format obéit à un standard.
Chaque transaction consiste dans sa version la plus maximaliste en la succession des paquets suivants :

- un paquet Jeton (Token) (en tête avec le type de la transaction qui va suivre et qui a pour but de transporter l'adresse USB et le sens du transfert.)
- un paquet de données (DATA) (contenant la " charge utile ")
- un paquet d'état (HandShake) (utilisé pour valider les transactions et pour fournir des moyens de corrections d'erreurs).
- Les paquets début de trame (SOF) indiquent le commencement d'une nouvelle trame.

Voici ci dessous le détail d'une transaction :



Le type de transaction va dépendre du mode de transfert

7.3) Les modes de transfert

Le système USB crée des canaux virtuels pour les différents flux de données.

Un canal aboutit dans l'appareil sur une terminaison.

Un appareil possède plusieurs terminaisons et est toujours reliés à l'hôte au moins par le "canal par défaut" aboutissant à la terminaison numéro zéro; cette terminaison est bidirectionnelle.

Pour les autres terminaisons il peut y avoir 2 canaux, l'un montant et l'autre descendant; la combinaison de l'adresse de l'appareil, du numéro de la terminaison et de la direction est unique. Un canal est caractérisé par sa terminaison, sa bande passante et son **mode de transfert**.

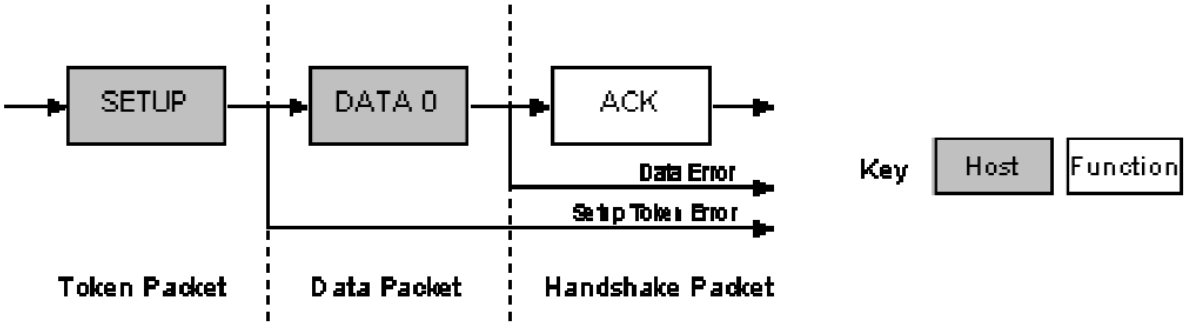
La spécification de l'USB définit 4 modes de transferts entre l'hôte et les périphériques.

7.3.1) Transfert en mode contrôle

Ce sont les transferts qui sont généralement utilisés pour les opérations de commande et d'état. Ce mode de transfert est, par exemple, utilisé pendant l'énumération du périphérique.

Un Transfert de contrôle ou commande peut avoir plus de 3 étapes.

L'**étape d'installation** se passe lorsqu'une demande est envoyée. Elle est composée de 3 paquets. Le jeton (token) d'installation envoyé le premier est celui qui contient l'adresse et le numéro de la terminaison. Le paquet de données est envoyé après et a toujours un type PID de Data0 et inclut un paquet d'installation qui détaille le type de la demande.
. Le dernier paquet est une poignée de mains utilisé pour valider la bonne réception ou pour indiquer une erreur. Si la fonction reçoit correctement la donnée d'installation (CRC et PID etc...Ok) elle répond avec ACK, sinon elle ignore la donnée et n'envoie pas un paquet de poignée de mains.
Les fonctions ne peuvent pas émettre un paquet STALL ou NAK en réponse à un paquet d'installation.

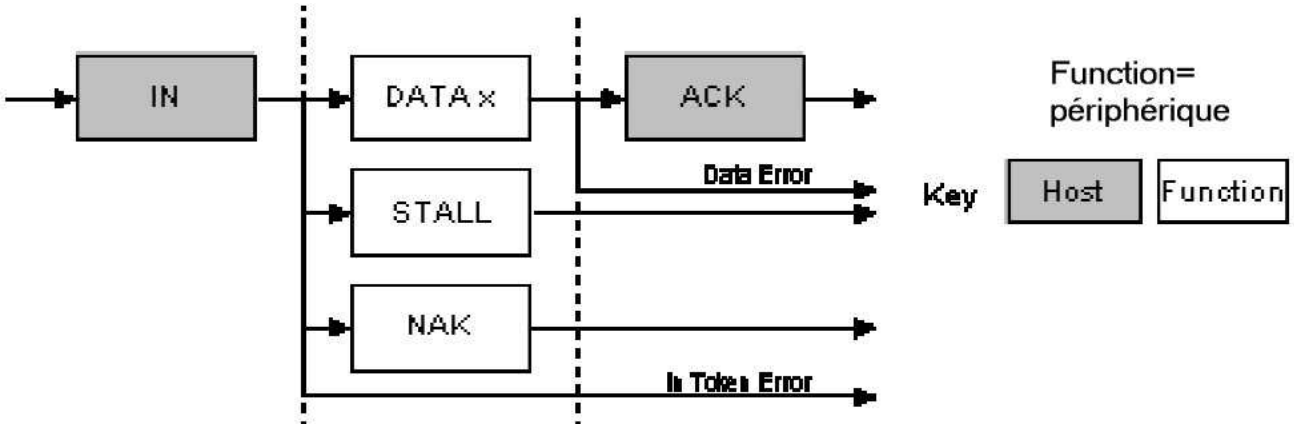


Function = périphérique

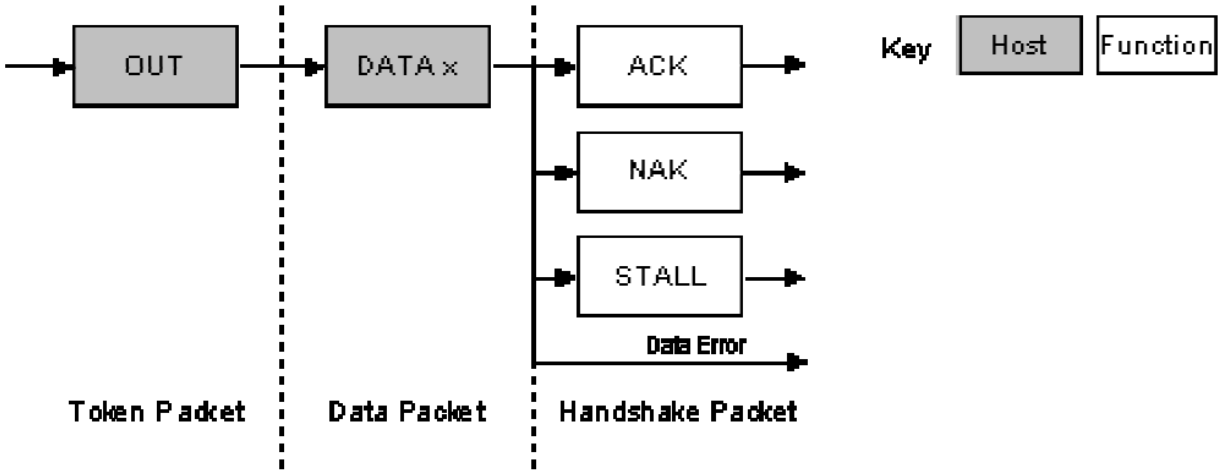
L'**étape de données** facultative consiste en un ou plusieurs transferts IN (Entrée) ou OUT (sorties).
La demande d'installation indique la quantité de données qui doit être envoyée dans cette étape. Si elle dépasse la taille maximale du paquet, les données seront envoyées en plusieurs transferts, chacune ayant la longueur maximale du paquet à l'exception du dernier paquet.

L'étape de données comporte 2 scénarios différents selon la direction du transfert de données.

IN (Entrée): Quand l'hôte est prêt à recevoir les données de commande, il émet un jeton (token) IN. Si la fonction reçoit le jeton IN avec une erreur, c'est-à-dire que le PID ne correspond pas avec les bits inversés du PID, il ignore donc le paquet. Si le jeton est reçu correctement, l'appareil peut soit répondre avec un paquet de données contenant les données de commande à envoyer, soit un paquet "d'arrêt" signalant que la terminaison a eu une erreur soit un paquet NAK signalant à l'hôte que la terminaison fonctionne, mais provisoirement n'a pas de données à envoyer.



OUT (Sortie): Quand l'hôte a besoin d'envoyer à l'appareil un paquet de données de commande, il émet un jeton OUT suivi par un paquet de données contenant les données de commande comme "charge utile" (payload). Si une partie du jeton OUT ou du paquet de données est altéré alors la fonction ignore le paquet. Si le buffer de terminaison de la fonction était vide et qu'il a cadencé les données dans le buffer de terminaison, il produit un ACK avisant l'hôte qu'il a bien reçu les données. Si le buffer de terminaison n'est pas vide à cause du traitement du paquet précédent, alors la fonction retourne un NAK. Toutefois si la terminaison
comporte une erreur et que son bit "halt" ai été positionné, elle retourne un STALL (Bloqué).



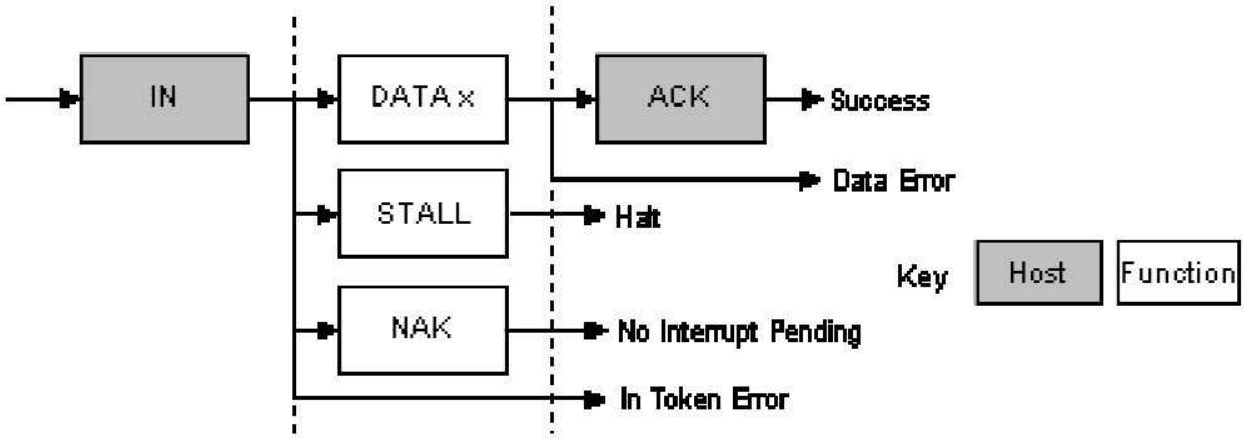
L'**étape d'état** rend compte des états de l'ensemble de demandes et cette fois encore change selon la direction du transfert.

7.3.2) Transfert en mode d'interruption

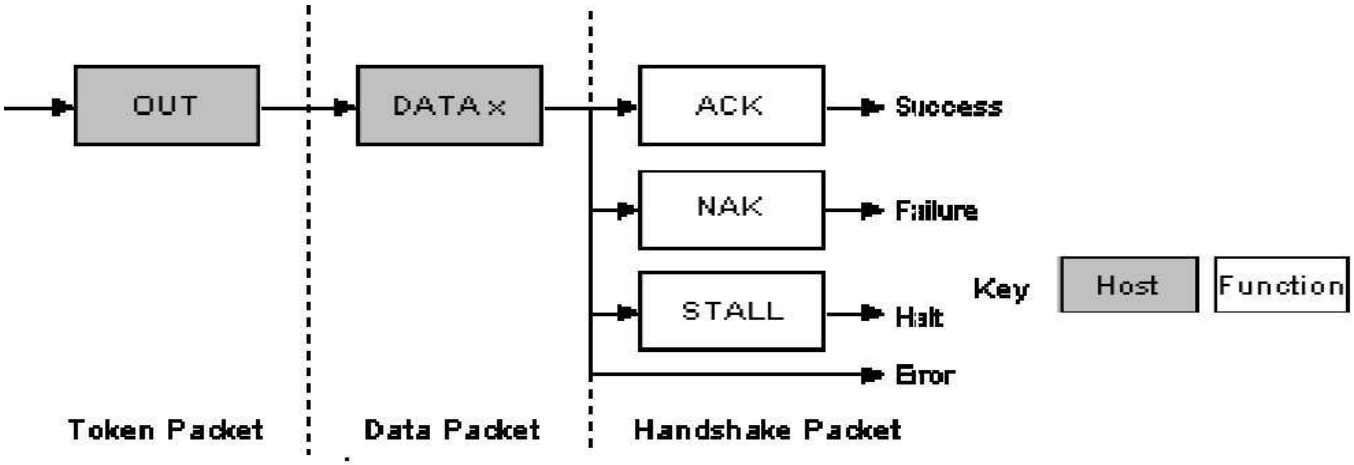
Il est destiné aux appareils transmettant peu de données mais dans un délai garanti, il garantit la fréquence d'interrogations de l'appareil par l'hôte à une cadence fixe de scrutation ainsi que la reprise sur les erreurs.

Il est utilisé pour des transferts à l'initiative du périphérique (asynchrones) et pour des transferts périodiques ou permanents comme les claviers, les souris….

IN: L'hôte interrogera périodiquement la terminaison d'interruption. Le taux d'interrogation est spécifié dans le descripteur de terminaison qui sera examiné plus tard. Chaque interrogation obligera l'hôte à envoyer un jeton IN. Si le jeton IN est altéré, la fonction ignore le paquet et continue la surveillance du Bus pour de nouveaux jetons.



OUT: Quand l'hôte veut envoyer à l'appareil les données d'interruptions, il émet un jeton OUT suivi par un paquet Data contenant les données d'interruption. Si une partie du jeton OUT ou du paquet Data est altéré alors la fonction ignore le paquet. Si le tampon de terminaison de la fonction était vide et qu'il ait cadencé les données dans le tampon de terminaison il émettrait un ACK prévenant l'hôte qu'il a reçu correctement les données. Si le tampon de terminaison n'est pas vide à cause du traitement d'un paquet précédent, alors la fonction retourne un NAK. Toutefois si une erreur se produisait à cause de la terminaison et que son bit d'arrêt (Halt) ait été positionné, elle renverrait un STALL (Bloqué).

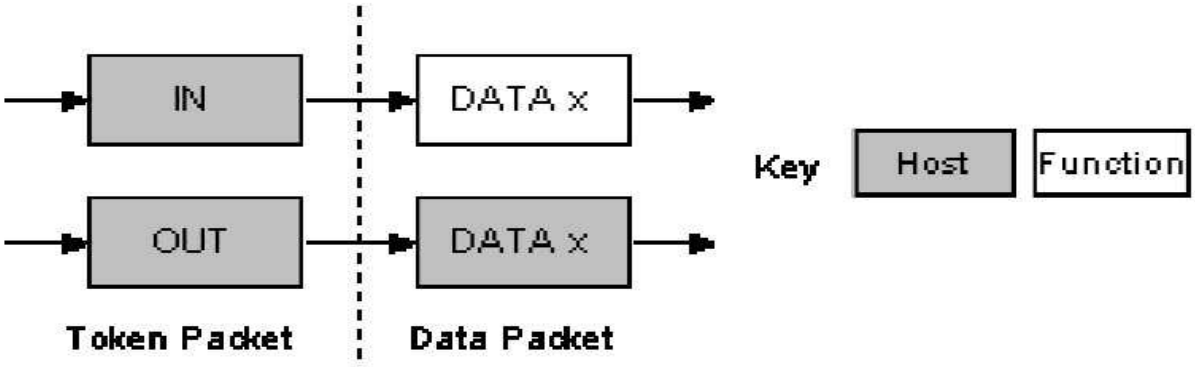


7.3.3)Transfert en mode isochrone

C'est un transfert à débit constant tolérant aux erreurs, destiné aux flux importants de données tels l'audio ou la vidéo. Le débit dans le canal est garanti, aucun format n'est imposé aux données.

C'est certainement le mode de transfert le plus efficace en matière de débit, de disponibilité de la bande passante et du délai d'attente. Mais c'est aussi le plus complexe.

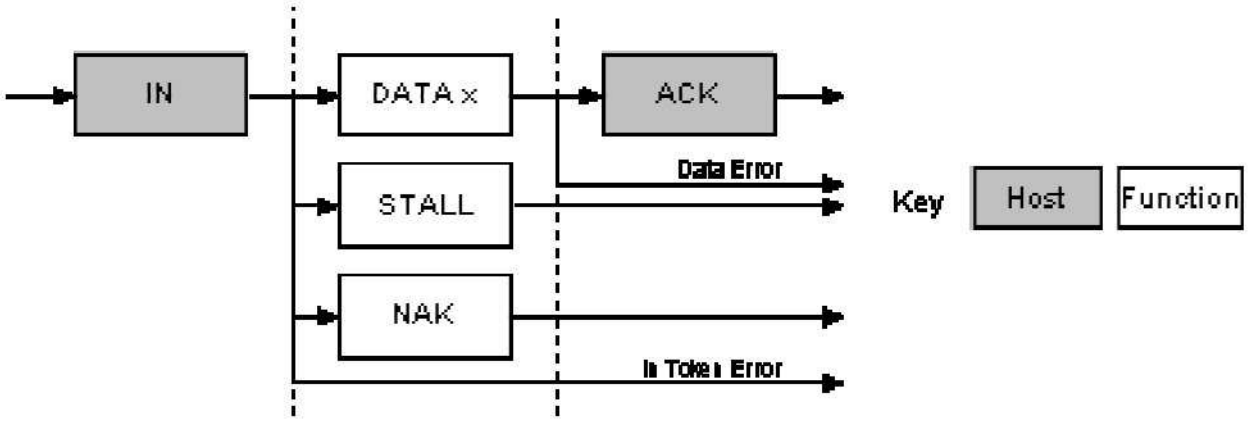
Le diagramme ci-dessus montre le format d'une transaction Isochrone IN et OUT. Les transactions Isochrones n'ont pas d'étape de poignée de mains et ne peuvent pas rendre compte des erreurs ou des conditions STALL / HALT ((Bloqué) / Arrêt).



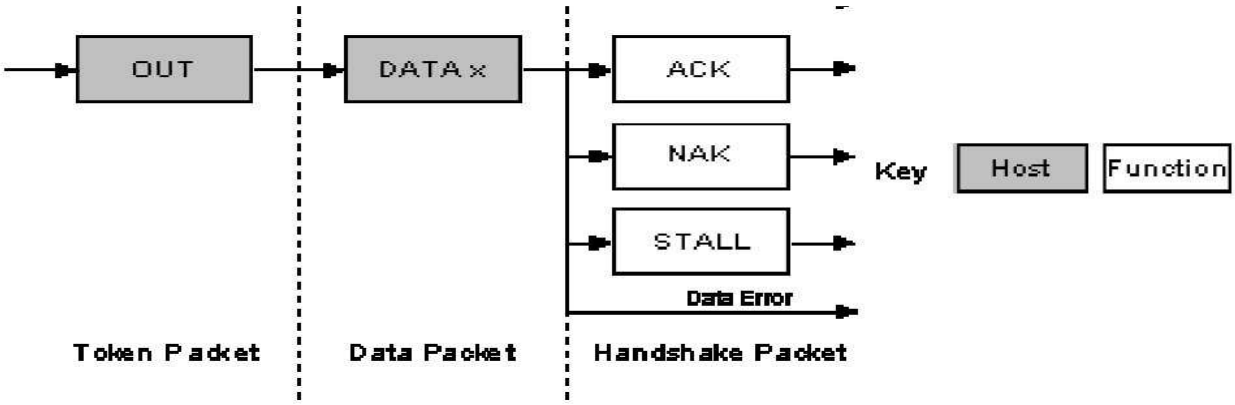
7.3.4)Transfert en mode en Bloc ou bulk

Ce mode est réservé pour les gros transferts de données. Le débit est variable et dépend de la disponibilité. Ce mode assure la reprise sur les erreurs. Utilisé pour les périphériques de stockage de masse comme une clé usb, un disque dur externe, lecteur de cartes mémoires…

IN: Quand l'hôte est prêt à recevoir des données en Bloc, il émet un jeton IN. Si la fonction reçoit le jeton IN avec une erreur, il ignore le paquet. Si le jeton est reçu correctement, la fonction peut soit répondre avec un paquet DATA contenant les données en Bloc à envoyer ou bien un paquet Stall signalant que la terminaison a eu une erreur ou un paquet NACK signalant àl'hôte que la terminaison travaille, mais provisoirement n'a pas de données à envoyer.



OUT: Quand l'hôte veut envoyer à la fonction un paquet de données en Bloc, il émet un jeton OUT suivi par un paquet DATA contenant les données en Bloc. Si une partie du jeton OUT ou du paquet DATA est altérée, alors la fonction ignore le paquet. Si le tampon de terminaison de la fonction est vide et qu'il a cadencé les données dans le tampon de terminaison, il émet un ACK prévenant l'hôte qu'il a reçu correctement les données. Si le tampon de terminaison n'est pas vide à cause du traitement d'un paquet précédent, alors la fonction retourne un NAK. Toutefois si la terminaison a eu une erreur et que son bit d'arrêt a été positionné, elle retourne un Stall (Bloqué).



Voici un tableau récapitulatif des caractéristiques des 4 modes de transfert :

	Control autorisé aux périphériques low speed	**Isochrone** Transfert interdit aux périphériques low speed	**Interruption** autorisé aux périphériques low speed	**Bulk** interdit aux périphériques low speed
Direction	Bidirectionnelle	Unidirectionnelle	Unidirectionnelle	Unidirectionnelle
Contrainte sur la taille maximale du bloc de données	Low speed : 8 octets pas trame Full speed : 8, 16, 32, ou 64 octets par trame high speed : 64 octets par trame	full speed : 1023 octets par trame high speed : 1024 octets par trame	Low speed : 8 octets pas trame Full speed : 64 octets par trame High speed : jusqu'à 1024 octets par trame	Full speed : 8, 16, 32, ou 64 octets par trame High speed : jusqu'à 1024 octets par trame
Accusé de réception, reprise sur erreur	Oui	Non	Oui	Oui
Utilisation	Configuration, commande, état…	Audio, vidéo…	Clavier, souris…	Clé usb, disque dur externe