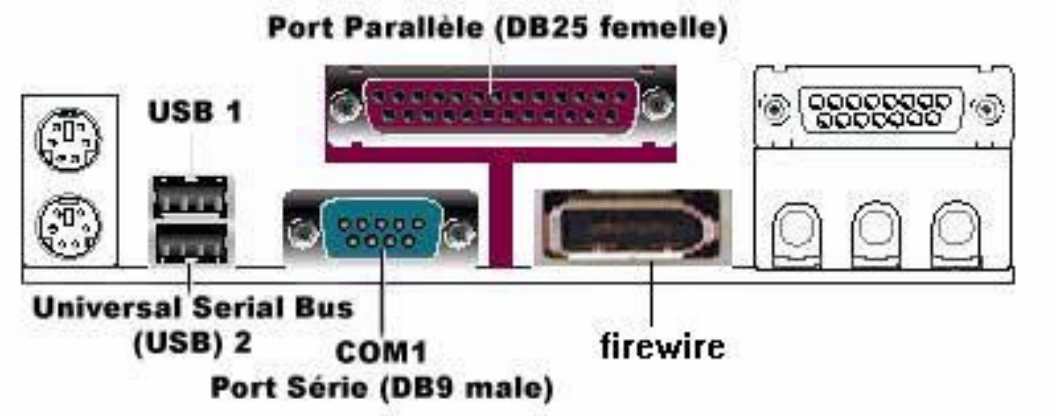
LES PRINCIPAUX PORTS PC



SOMMAIRE

1.Port Série

L'interface série est une interface asynchrone, ce qui signifie que le signal de cette interface n'est pas synchronisé avec celui d'un bus quelconque. C'est à dire que les bits des données sont envoyés les uns après les autres.

Pilotée par l'UART (Universal Asynchronous Receiving Transmitter), cette interface, même si elle est en voie de disparition, possède des caractéristiques très peu enviables aux interfaces récentes tout en ayant l'avantage d'être flexible et très facile d'utilisation.

Le port série, comme son nom l'indique, permet de véhiculer des informations en série, à l'opposé d'une interface parallèle qui peut véhiculer un mot entier en une période donnée. Les mots à transmettre sont donc auparavant codés puis ensuite décodés à leur arrivée pour pouvoir interprétés, les bus internes des ordinateurs étant de type parallèle. Une autre caractéristique intéressante est que la liaison série est totalement asynchrone. Aucune horloge n 'est transmise. Il suffit donc de se mettre d'accord sur la vitesse de transfert des bits et rajouter des bits de synchronisation entre les deux appareils de communicants.

1.1.Géométrie

A l'origine, tous les compatibles PC possèdent 2 ports séries: COM1 et COM2. L'un d'entre eux se présente sous la forme d'une prise DB9 mâle et le deuxième, sous la forme d'une DB25 mâle.

```
DB9 Mâle (vue de devant)          DB25 Mâle (vue de devant)
-------------                     ------------------------------
\ 1 2 3 4 5 /                     \ 1  2  3  4  5  7  8 ... 13 /
 \ 6 7 8 9 /                       \ 14 15 16 17 18 .....25 /
  ---------                         --------------------------
```

Description et attribution des signaux :

Broche DB9	Broche DB25	Nom
1	8	DCD
2	3	RX
3	2	TX
4	20	DTR
5	7	GND
6	6	DSR
7	4	RTS
8	5	CTS
9	22	RI

- **DCD (Data Carrier Detect)** :cette ligne est une entrée active haute. Elle signale à l'ordinateur qu'une liaison a été établie avec un correspondant.
- **RX (Receive Data)** :cette ligne est une entrée. C'est ici que transitent les informations du correspondant vers l'ordinateur.
- **TX (Transmit Data)** :cette ligne est une sortie. Les données de l'ordinateur vers le correspondant sont véhiculées par son intermédiaire.
- **DTR (Data Terminal Ready)** :cette ligne est une sortie active haute. Elle permet à l'ordinateur de signaler au correspondant que le port série a été libéré et qu'il peut être utilisé s'il le souhaite.
- **GND (GrouND)** :c'est la masse.
- **DSR (Data Set Ready)** :Cette ligne est une entrée active haute. Elle permet au correspondant de signaler qu'une donnée est prête.
- **RTS (Request To Send)** :cette ligne est une sortie active haute. Elle indique au correspondant que l'ordinateur veut lui transmettre des données.
- **CTS (Clear To Send)** :cette ligne est une entrée active haute. Elle indique à l'ordinateur que le correspondant est prêt à recevoir des données.
- **RI (Ring Indicator)** :cette ligne est une entrée active haute. Elle permet à l'ordinateur de savoir qu'un correspondant veut initier une communication avec lui.

D'un point de vue électronique, les signaux TX et RX en sortie des prises répondent aux normes RS232. Pour cela on utilise des tensions positives et négatives avec une logique négative.

- En émission la tension est comprise entre 5 et 25V est un 0 logique.
- En émission la tension est comprise entre -5 et -25V est un 1 logique.
- En réception la tension est comprise entre 3 et 25V est un 0 logique.
- En réception la tension est comprise entre -3 et -25V est un 1 logique.

C'est à dire que le « 1 » logique est compris entre -3 et -25V et que le « 0 » logique est compris entre +3 et +25V

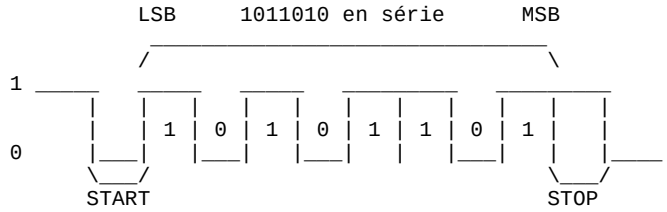
1.2.Programmation

La programmation du port série passe par la description de son fonctionnement et par une petite explication des protocoles de transmission. Les PC possèdent en général deux ports série: COM1, généralement réservé à l'indispensable souris, et COM2 utilisé parfois en conjonction avec un modem externe.

1.3.Fonctionnement d'une liaison série

La communication série nécessite trois fils au minimum: une masse pour référencer les signaux, un fil émetteur et un fil récepteur. Notre liaison série est en effet full-duplex, c'est à dire que l'on peut émettre et recevoir en même temps (comme le téléphone par exemple).
La différence principale entre le port parallèle et le port série est que les informations ne sont pas transmises simultanément sur des fils séparés (D0 à D7) mais les unes après les autres sur un même fil. Cela amène une économie de câble (un fil au lieu de 8) mais un montage décodeur devient nécessaire pour retransformer les données sérialisées.

La figure ci-dessous montre comment l'octet 10110101 est transformé pour être transmis sur un seul fil. Vous voyez qu'en plus de l'information utile (10110101) se greffent d'autres bits comme le bit de start. Ces bits sont utiles pour la synchronisation de l'émetteur et du récepteur.



En effet, la liaison série est totalement asynchrone, aucune horloge n'est transmise, il faut donc se mettre d'accord sur la vitesse de transfert des bits et rajouter des bits de synchronisation.
Voici un petit résumé des différents paramètres rentrant en jeu lors d'une communication série:

- **longueur de mot:** sur le PC, le BIOS ne permet une longueur de mot que de 7 ou 8 bits.
- **Parité:** le mot transmis peut être suivi d'un bit de parité qui sert à détecter les erreurs éventuelles de transmission. Il existe deux parités: la parité paire et la parité impaire. Dans le cas de la parité paire, et pour le mot 10110101 contenant 5 états à 1, le bit de parité sera 1 amenant ainsi le nombre total de 1 à un nombre pair (6). Dans le cas de la parité impaire, le bit de parité aurait été 0 car le nombre total de 1 est déjà impair. L'intérêt de ce rajout est le suivant: si jamais lors de la transmission un état 1 est transformé en état 0 (perturbation du canal par des parasites par exemple) le nombre total de 1 change et donc le bit de parité recalculé par le récepteur ne correspond plus à celui reçu. L'erreur est donc détectée. Evidemment, si deux états à 1 passent à 0, l'erreur ne sera pas détectée mais la probabilité pour que cela arrive est très faible.
- **Bit de start:** lorsque rien ne circule sur la ligne, celle-ci est à l'état haut. Pour indiquer qu'un mot va être transmis, la ligne passe à bas avant de commencer le transfert. Cette précaution permet de resynchroniser le récepteur.
- **Bits de stop:** ces bits signalent la fin de la transmission. Selon le protocole utilisé, il peut y avoir 1, 1.5, ou 2 bits de stop (ces bits sont toujours à 1).
- **Vitesse de transmission:** la plupart des cartes série permettent de choisir une vitesse entre 300 et 9600 bauds (par exemple à 300 bauds, un bit est transmis tout les un trois-centième de seconde). Les cartes récentes proposent des vitesses jusqu'à 115200 bauds. Ces vitesses ne vous paraissent peut-être pas énormes mais il faut garder à l'esprit que la liaison série est avant tout pensée pour les liaisons téléphoniques par modems, dont la bande passante est très limitée.

1.4.Protocoles de transmission

On ne peut réussir une transmission qu'à partir du moment où l'émetteur et le récepteur
Se sont entendu sur la vitesse, le nombre de bit de stop, la longueur du mot et la parité. A ce niveau là, savoir à quel voltage correspond un état haut n'a aucune importance.
D'une manière générale, la parité est toujours présente car elle permet de détecter la plus grande partie des erreurs de transmission.

1.5.Programmation port série

L'accès aux registres contrôlant les ports série se fait par l'intermédiaire de l'interruption DOS 14h. A cette IT correspond 4 fonctions permettant de configurer et de contrôler l'interface série, dont voici les descriptions complètes.

Fonction 0x00: Initialisation de l'interface série
Permet de fixer le protocole de transmission.
Entrée: AH = 0x00
DX = Numéro de l'interface série
0x00 = COM1
0x01 = COM2
AL = Paramètres de configuration
Bits 0-1: longueur du mot
10 = 7 bits
11 = 8 bits
Bit 2: nombre de bits de stop
0 = 1 bit de stop
1 = 2 bits de stop
Bit 3-4: contrôle de parité
00 = aucun
01 = impair
11 = pair
Bit 5-7: vitesse de transmission
000 = 110 bauds
001 = 150 bauds
010 = 300 bauds
011 = 600 bauds
100 = 1200 bauds
101 = 2400 bauds
110 = 4800 bauds
111 = 9600 bauds
Sortie: AH = Etat de l'interface série
Bit 0: données prêtes
Bit 1: données effacées
Bit 2: erreur de parité
Bit 3: violation de protocole
Bit 4: interruption détectée
Bit 5: transmission Hold Register vide
Bit 6: transmisison Shift Register vide
Bit 7: time out (le périphérique ne répond pas)
AL = Etat du modem
Bit 0: (delta) modem prêt à émettre
Bit 1: (delta) modem activé
Bit 2: (delta) sonnerie
Bit 3: (delta) liaison établie
Bit 4: modem prêt à émettre
Bit 5: modem activé
Bit 6: sonnerie
Bit 7: liaison établie
(Les bits delta montrent une modification par rapport au dernier appel de la fonction)

Fonction 0x01: Emission de caractères
Entrée: AH = 0x01
DX = Numéro de l'interface série (voir précédemment)
AL = Code du caractère à sortir
Sortie: AH = Bit 7: 0 caractère transmis
1 erreur, d'où:
Bit 0-6 = Etat de l'interface série
Bit 0: données prêtes
Bit 1: données effacées
Bit 2: erreur de parité
Bit 3: violation de protocole
Bit 4: interruption détectée
Bit 5: transmission Hold Register vide
Bit 6: transmisison Shift Register vide

Fonction 0x02: Réception de caractères
Entrée: AH = 0x02
DX = Numéro de l'interface série (voir précédemment) Sortie: AH = Bit 7: 0 caractère reçu, d'où:
AL = Caractère reçu
Bit 7: 1 erreur, d'où:
Bit 0-6: Etat de l'interface série
Bit 0: données prêtes
Bit 1: données effacées
Bit 2: erreur de parité
Bit 3: violation de protocole
Bit 4: interruption détectée
Bit 5: transmission Hold Register vide
Bit 6: transmisison Shift Register vide

Fonction 0x03: Tester état
Entrée: AH = 0x03
DX = Numéro de l'interface série (voir précédemment) Sortie: AH = Etat de l'interface série
Bit 0: données prêtes
Bit 1: données effacées
Bit 2: erreur de parité
Bit 3: violation de protocole
Bit 4: interruption détectée
Bit 5: transmission Hold Register vide
Bit 6: transmisison Shift Register vide
Bit 7: time out (le périphérique ne répond pas)
AL = Etat du modem
Bit 0: (delta) modem prêt à émettre
Bit 1: (delta) modem activé
Bit 2: (delta) sonnerie
Bit 3: (delta) liaison établie
Bit 4: modem prêt à émettre
Bit 5: modem activé
Bit 6: sonnerie
Bit 7: liaison établie
Remarque :cette fonction doit être appelée avant 0x02 afin de déterminer si un caractère a été reçu. Dans ce cas le bit 0 du registre AH vaut 1.

Afin de mieux saisir l'utilisation de ces fonctions, voici un petit exemple en C effectuant le réglage du protocole à 1200 bauds, 7 bits et parité paire.

```
pregs.h.ah = 0x00;              /* fonction 0: réglage du protocole */
pregs.h.al = 0x9A;              /* 7 bits, parité paire, 1200 bauds */
pregs.x.dx = COM;
int86(0x14,&pregs,&pregs);      /* IT DOS 14 */
```

2.Port Parallèle

Tout comme le port série, le port parallèle est très répandu et son utilisation est encore très large de par sa vitesse, son coût et sa large intégration. Il se présente sous la forme d'un connecteur DB25 femelle et a été conçue plus spécialement pour pouvoir relier une imprimante au PC. D'ailleurs, il n'est pas difficile de s'apercevoir que la plupart des caractéristiques des pattes d'un connecteur de cette interface ont une fonction en relation étroite avec ce type de périphérique. Le port parallèle a subi plusieurs améliorations aux cours des évolutions et des besoins croissants des PC.

2.1.Mode SPP

Mode de base, le mode SPP (Standard Parallel Port) est le protocole de base permettant l'envoi de données vers une imprimante. Il peut être appelé dans ce cas « Printer Mode » ou « unidirectionnel mode » ou encore « Centronics » (nom également attribué au cordon reliant le PC aux imprimantes). Depuis, le SPP st devenu capable d'envoyer et de recevoir. Il peut donc être bidirectionnel. On le retrouve alors sous des appellations « Bidirectionnal mode » ou simplement « SPP ». C'est le protocole le plus simple, mais la vitesse de transmission maximale que l'on peut espérer obtenir avec un tel port est de l'ordre de 150 ko/s.

2.2.Mode EPP

En 1991, Xircom, Zenith et Intel ont développé un port plus rapide appelé EPP (Enhanced Parallel Port). Il permet d'attendre un débit théorique de 2 Mo/s, soit un débit environ treize fois supérieur au SPP. Si ce débit reste inférieur à celui des bus ISA (8 Mo/s), il permet néanmoins l'échange de données avec des périphériques tels que les lecteurs de CD-ROM ou les disques durs. Le protocole EPP est lui explicitement bidirectionnel. Il constitue un vrai protocole de communication bidirectionnel alors que SPP reste un protocole « bricolé » lorsqu'il est bidirectionnel.

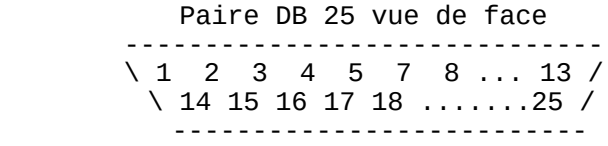
2.3.Mode ECP

Le plus récent de ces modes est l'ECP (Extended Capacity Port). Il a été conçu par Hewlett Packard et Microsfot. Il dérive fortement d'ailleurs du mode EPP et en possède les mêmes caractéristiques avec toutefois des fonctionnalités supplémentaires, comme la gestion des périphériques « Plug and Play », l'identification de périphériques auprès de la machine dès le début du boot, le support du DMA (Direct Memory Access). Ainsi, il est possible d'envoyer ou de recevoir des données sans avoir besoin du processeur. Une autre fonction intéressante :ce protocole comprime les données, selon une compression RLE (Run Lenght Encoding - compression des répétitions d'octets) au niveau matériel. Le taux de compression peut attendre 64:1. Ceci est utile avec des périphériques comme des scanners et des imprimantes où une grande partie des données est constituée de longues chaînes répétitives.

Il utilise aussi un « buffer » de type FIFO pour envoyer et recevoir des données. Enfin, l'adressage des périphériques se fait par un numéro de canal. Par exemple, un fax muni d'une liaison parallèle peut être utilisé comme scanner, modem/fax et imprimante, chaque fonction pouvant être adressée séparément. Ce système d'adressage ne permet pas d'utiliser des périphériques différents. On est limité à l'utilisation d'un périphérique multifonctions, chaque fonction ayant une adresse.

2.4.Géométrie du port parallèle

Le port parallèle des PC se présente sous la forme d'une prise DB25 femelle (femelle parce qu'il est composé de *trous...)* dont voici la géométrie:



Le brochage est le suivant :

Broche	Nom	Sortie	Entrée
1	/STROBE	X	X
2	D0	X	X si EPP
3	D1	X	X si EPP
4	D2	X	X si EPP
5	D3	X	X si EPP
6	D4	X	X si EPP
7	D5	X	X si EPP
8	D6	X	X si EPP
9	D7	X	X si EPP
10	/ACK	-	X
11	BUSY	-	X
12	PE	-	X
13	SELECT	-	X
14	/AUTOFEED	X	X
15	/ERROR	-	X
16	/INIT	X	X
17	/SELECT	X	X
18-25	MASSE	X	X

2.5.Description des signaux

- **STROBE** : cette ligne active basse (donc a 0) indique à l'imprimante que des données sont présentes sur les lignes D0 à D7 et qu'il faut les prendre en compte.
- **D0 à D7** : c'est le bus de données sur lequel véhicule la valeur du caractère à imprimer. On ne peut écrire sur ce port, à moins d'avoir un port parallèle étendu (c'est le cas pour les ports de type ECP/EPP).
- **ACK** : l'imprimante met à 0 cette ligne pour indiquer à l'ordinateur qu'elle a bien reçu le caractère transmit et qu'il peut continuer la transmission.
- **BUSY** : cette ligne est mise à 0 par l'imprimante lorsque son buffer de réception est plein. L'ordinateur est ainsi averti que celle-ci ne peut plus recevoir de données. Il doit attendre que cette ligne revienne à 1 pour recommencer à émettre.
- **PE** :signifie " paper error ". L'imprimante indique par cette ligne à l'ordinateur que l'alimentation en papier a été interrompue.
- **SELECT** : cette ligne indique à l'ordinateur si l'imprimante est "on line" ou "off line".
- **AUTOFEED** : lorsque ce signal est à 1, l'imprimante doit effectuer un saut de ligne à chaque caractère "return" reçu. En effet, certaines imprimantes se contentent d'effectuer un simple retour du chariot en présence de ce caractère.
- **ERROR** : indique à l'ordinateur que l'imprimante a détecté une erreur.
- **INIT** : l'ordinateur peut effectuer une initialisation de l'imprimante par l'intermédiaire de cette ligne.
- **SELECT IN** : l'ordinateur peut mettre l'imprimante hors ligne par l'intermédiaire de ce signal.
- **MASSE** : c'est la masse du PC.

2.6.Programmer avec le port parallèle

Il est très facile de programmer cette interface. Trois registres seulement sont nécessaires au contrôle total des signaux, que l'on contrôle par les ports d'entrée/sortie du PC (Rappel : un registre est un endroit où sont stockées des valeurs). Chaque port parallèle possède ses propres registres (il n'y a pas de multiplexage ici) :

Port parallèle	Port du registre de données	Port du registre d'état	Port du registre de commande
n°1	378h	379h	37Ah
n°2	278h	279h	27Ah
n°3	3BCh	3BDh	3BEh

Les lignes de données

Ce registre ne peut qu'être écrit.

Détail du registre

Bit	7	6	5	4	3	2	1	0
Nom	D7	D6	D5	D4	D3	D2	D1	D0

Les lignes d'état de l'imprimante

Ce registre, accessible uniquement en lecture au contraire du précédent.... Il ne faut surtout pas écrire dessus... sinon on grille le PC.

Bit	7	6	5	4	3	2	1	0
Nom	/BUSY	ACK	PE	SELECT	/ERROR	X	X	X

Le "/" indique "Logique Inversée"

X : Indéfini

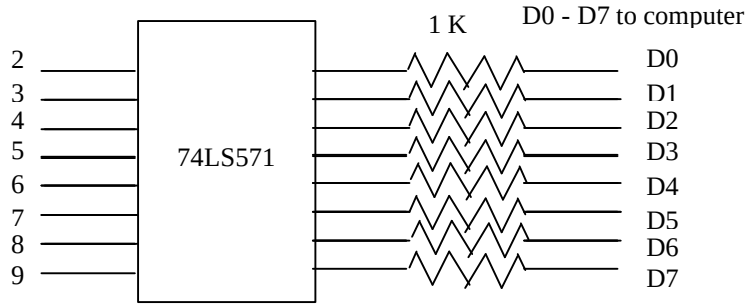
Les lignes de commandes de l'imprimante

Ce dernier registre est accessible à la fois en lecture et en écriture.

Bit	7	6	5	4	3	2	1	0
Nom	X	X	BIT CTRL	IRQ ENABLE	SELECT IN	/INIT	AUTOFEED	/STROBE

Programmation du port

Il est très facile de griller (Sens propre et figuré !) un PC simplement en jouant avec le port parallèle ! Pour ceux qui connaissent un peu l'électronique, il vous suffit de mettre un montage suiveur avec des portes ET correctement câblées. Comme ça, ce sera la porte qui grillera et non le PC ! Pour cela, vous pouvez mettre un 74LS571 qui est un CI suiveur et amplificateur. Ainsi, il protégera correctement votre PC et il vous permettra de disposer d'une tension de sortie correcte.



3.Port USB

L'USB (Universal Serial Bus) version 1.1 comprend 2 vitesses :

- Mode vitesse rapide de 12Mbits/s
- Mode vitesse lente de 1,5Mbits/s

Alors que l'USB 2.0 fait monter les enchères jusqu'à 480Mbits/s, il a été crée pour entrer en compétition avec le Bus Série Firewire.

L'USB est un Bus Série. Il utilise 4 fils isolés dont 2 sont l'alimentation (+5V et GND). Les 2 restants forment une paire torsadée qui véhicule les signaux de données différentiels. Il utilise un schéma d'encodage NRZI (No Return To zero).

L'hôte USB a la charge de mener bien toutes les transactions et de programmer la bande passante. Les données peuvent être envoyées par différentes méthodes de transactions en utilisant un protocole basé sur un système de jeton.

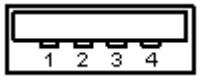
L'USB supporte le système " plug'n plug " branchement à chaud avec des drivers qui sont directement chargeable et déchargeable. L'utilisateur branche simplement l'appareil sur le Bus. L'Hôte détectera cet ajout, interrogera l'appareil nouvellement inséré et chargera le driver approprié. Une fois que l'utilisateur a terminé, on peut simplement retirer le câble, l'Hôte détectera cette absence et déchargera automatiquement le driver.

Le chargement du driver approprié sera réalisé en utilisant une combinaison PID/VID (Interface Produit Machine/ Vendeur Machine).

3.1.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.

Il y a généralement 2 types de connecteurs USB, appelé type A et type B présenté ci-dessous.



Connecteur USB type A

Connecteur USB type B

	Mâle	Femelle
Type A	Appareil	Hôte
Type B	Hôte	Appareil

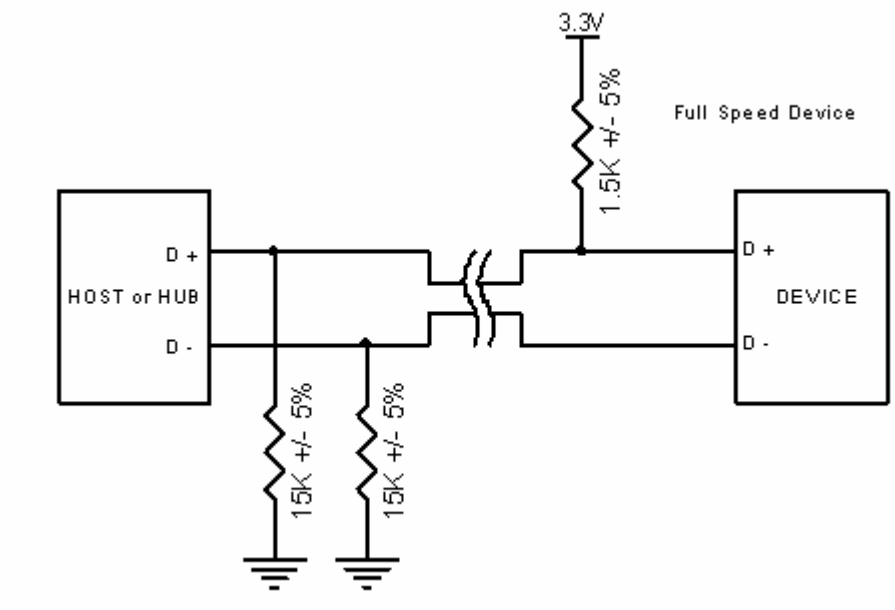
On utilise des couleurs standard pour les fils intérieurs des câbles USB de façon à faciliter l'identification des fils d'un constructeur à un autre.

Numéro de broches	Couleurs des câbles	Fonction
1	Rouge	V_{BUS} (5 volts)
2	Blanc	D-
3	Vert	D+
4	Noir	Masse

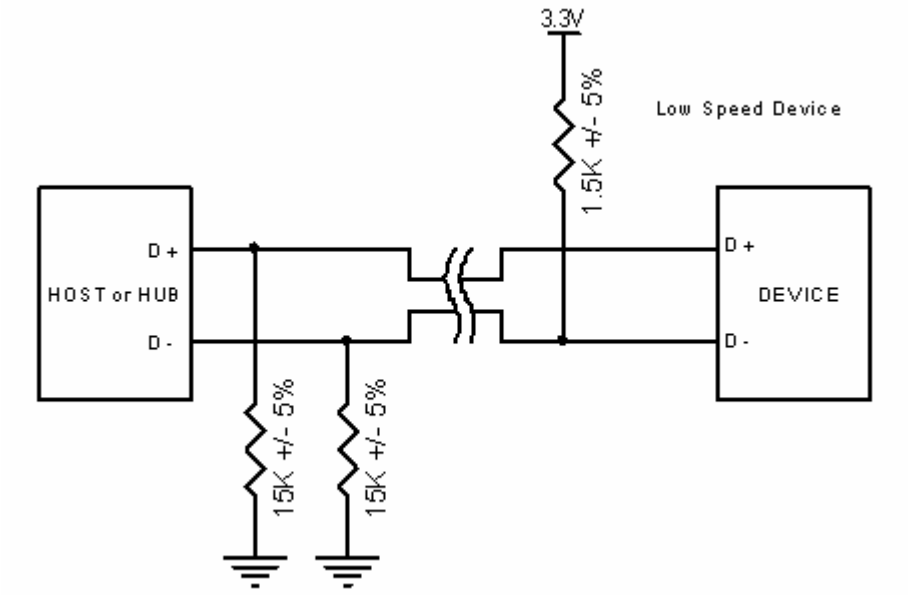
3.2.Electrique

3.2.1.Identification Vitesse

Un appareil USB doit indiquer sa vitesse en mettant soit D+ ou D- à 3,3V. Un appareil pleine vitesse, représenté plus bas utilisera une résistance de rappel rattaché à D+ pour se signaler comme tel. Ces résistances de rappel à l'extrémité de l'appareil seraient aussi utilisées par l'hôte pour détecter la présence d'un appareil connecté à son port. Sans résistance de rappel, l'USB suppose qu'il n'y a rien de connecté au BUS.



Appareil pleine vitesse avec résistance de rappel état haut branché sur D+



Appareil basse vitesse avec résistance de rappel état haut branché sur D-

3.2.2.Alimentaion :Vbus

Un des avantages de l'USB réside dans ces appareils alimentés par le Bus. Ceux-ci obtiennent leur alimentation à partir du Bus et ne demande aucune prise externe et câble additionnel.

Il existe 3 classes de fonctions USB :

- Les fonctions alimentés par le Bus à basse puissance (Low-Power)
- Les fonctions alimentés par le Bus à haute puissance (High-Power)
- Les fonctions auto alimentées (Self-powered)

Les fonctions alimentées par le Bus à basse puissance tirent toutes leurs puissances de VBus et ne peuvent en tirer que la charge d'une unité. La spécification USB défini une charge d'unité à 100mA. Les fonctions alimentées par le Bus à basse puissance peuvent être conçues pour travailler à une tension de V_{BUS} tombant à 4,4V et montant à un maximum de 5,25V mesuré à la prise amont de l'appareil.

Les fonctions alimentés par le Bus à haute puissance tirent toutes leurs puissances du Bus et ne peuvent tirer plus d'une unité de charge jusqu'à ce qu'elles aient été configurées, après quoi elles peuvent tirer 5 unités de charge (500mA max) pourvu que cela soit demandé dans son descripteur. Les fonctions du Bus à Haute puissance doivent être capable d'être détectées et énumérées à un minimum de 4,40V.

Les fonctions auto alimentées peuvent tirer jusqu'à une unité et faire dériver le reste de leur alimentation d'une source extérieure. Si cette source extérieure venait à manquer, il doit y avoir des réserves en place de manière à ne pas tirer plus d'une unité de charge du Bus.

Aucun appareil USB, qu'il soit alimenté par le Bus ou bien auto alimenté ne peut piloter V_{Bus} sur son port face amont. Si V_{Bus} est perdu, l'appareil a une durée de 10 secondes pour retirer l'alimentation des résistances de rappel de D+/D- utilisées pour l'identification de la vitesse.

3.4.Protocoles USB

Chaque transaction USB consiste d'un

- Paquet Jeton (Token) (en tête définissant ce qu'il attend par la suite)
- Paquet DATA optionnel (contenant la charge utile « payload »)
- Paquet d'état (utilisé pour valider les transactions et pour fournir des moyens de corrections d'erreurs).

L'USB est un Bus géré par l'hôte donc c'est lui qui initie toutes les transactions. Le premier paquet, aussi appelé Jeton est produit par l'hôte pour décrire ce qui va suivre et si la transaction de données sera en lecture ou écriture et ce que sera l'adresse de l'appareil et la terminaison désignée. Le paquet suivant est généralement un paquet de données transportant la « charge utile » et est suivi par un paquet « poignée de mains » (handShaking), signalant si les données ou le jeton ont été reçus correctement ou si la terminaison est bloquée, ou n'est pas disponible pour accepter les données.

3.4.1.Champs de paquets ordinaires

Les données sur le Bus USB sont transmises avec le bit LSB (Least Significant Bit) en premier. Les paquets USB se composent des champs suivants :

Sync :Tous les paquets doivent commencer avec un champ Sync. Le champ Sync fait de 8 bits de long pour la basse et pleine vitesse ou 32 bits pour la haute vitesse, il est utilisé pour synchroniser l'horloge du récepteur avec celle de l'émetteur. Les 2 derniers bits indiquent l'endroit ou le champ PID commence.

PID :PID signifie Paquet ID. Ce champ est utilisé pour identifier le type de paquet qui est envoyé. Il y a 4 bits pour le PID, toutefois pour s'assurer qu'il a été reçu correctement, les 4 bits sont complémentés et répétés faisant un PID de 8 bits au total.

Remarque :
SOF = Start Of Frame ; Début de Trame
SETUP = Installation, configuration
ACK = ACKnowledge ; Validation
NAK = No AcKnowledge ; Pas de validation
STALL = Bloqué
PREamble = Synchroniseur initial
Split = Partager, Fractionner
Ping = S'assurer d'une bonne connexion

Il y a 4 bits pour le PID, toutefois pour s'assurer qu'il a été reçu correctement, les 4 bits sont **complémentés** et répétés faisant un PID de 8 bits au total. Le format résultant figure ci-dessous.

Le tableau suivant montre les valeurs possibles.

GROUPE	VALEUR PID	IDENTIFICATEUR PAQUET
Token Jeton	0001	OUT Token
	1001	IN Token
	0101	SOF Token
	1101	Setup Token
Data Données	0011	DATA0
	1011	DATA1
	0111	DATA2
	1111	MDATA
Handshake Poignée de Mains	0010	ACK Handshake
	1010	NAK Handshake
	1110	STALL Handshake
	0110	NYET (No Reponse Yet)
Special	1100	PREamble
	1100	ERR
	1000	Split
	0100	Ping

ADDR :Le champ adresse détermine à quel appareil le paquet est destiné. Sa longueur de 7 bits, lui permet de supporter 127 appareils. L'adresse 0 n'est pas valide, tant qu'un appareil qui n'a pas encore d'adresse attribuée, doit répondre aux paquets envoyés d'adresse 0.

ENDP :Le champ de terminaison est composé de 4 bits, autorisant 16 terminaisons possibles. Les appareils basse vitesse, toutefois peuvent seulement avoir 2 terminaisons additionnelles au dessus du canal de communication par défaut.

CRC :Les Contrôles à Redondance Cyclique sont exécutés sur les données à l'intérieur du paquet de charge utile. Tous les paquets jetons ont un CRC de 5 bits tandis que les paquets de données ont un CRC de 16 bits.

EOP :Fin de Paquet. Signalé par une sortie unique zéro (SE0) pendant une durée approximative de 2 bits suivie par un " J " d'une durée de 1 bit.

3.4.2.Types de paquets USB

L'USB a quatre types différents de paquets :

- Les paquets jetons indiquent le type de la transaction qui va suivre
- Les paquets de données contiennent la charge utile
- Les paquets « poignée de mains » sont utilisés pour valider les données ou rapporter les erreurs
- Les paquets début de trame (SOF) indiquent le commencement d'une nouvelle trame.

Les paquets jetons.

Il y a 3 sortes de paquets Jetons,

- **In** :Informe l'appareil USB que l'hôte veut lire des informations.
- **Out** :Informe l'appareil USB que l'hôte veut envoyer des informations.
- **Setup** **:**Utilisé pour commencer les transferts de commande.

Le format de la trame est le suivant :

Sync	PID	ADDR	ENDP	CRC5	EOP

Les paquets de données

Il y a 2 sortes de paquets de données, chacun étant capable de transmettre plus de 1024 octets de données, les paquets de données ont le format suivant :

Sync	PID	Data	CRC16	EOP

- La taille maximale de données « charge utile » pour les appareils basse vitesse est de 8 octets.
- La taille maximale de données « charge utile » pour les appareils pleine vitesse est de 1023 octets.
- La taille maximale de données « charge utile » pour les appareils haute vitesse est de 1024 octets.
- Les données doivent être envoyées en multiple d'octets.

Les paquets " poignée de mains "

Il y a 3 sortes de paquets " poignée de mains " qui font simplement partie du PID :

- ACK :validant que le paquet a été reçu correctement.
- NAK :rapporte que temporairement l'appareil ne peut 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.

Les paquets " poignée de mains " ont le format suivant

Sync	PID	EOP

Les paquets début de trame (SOF).

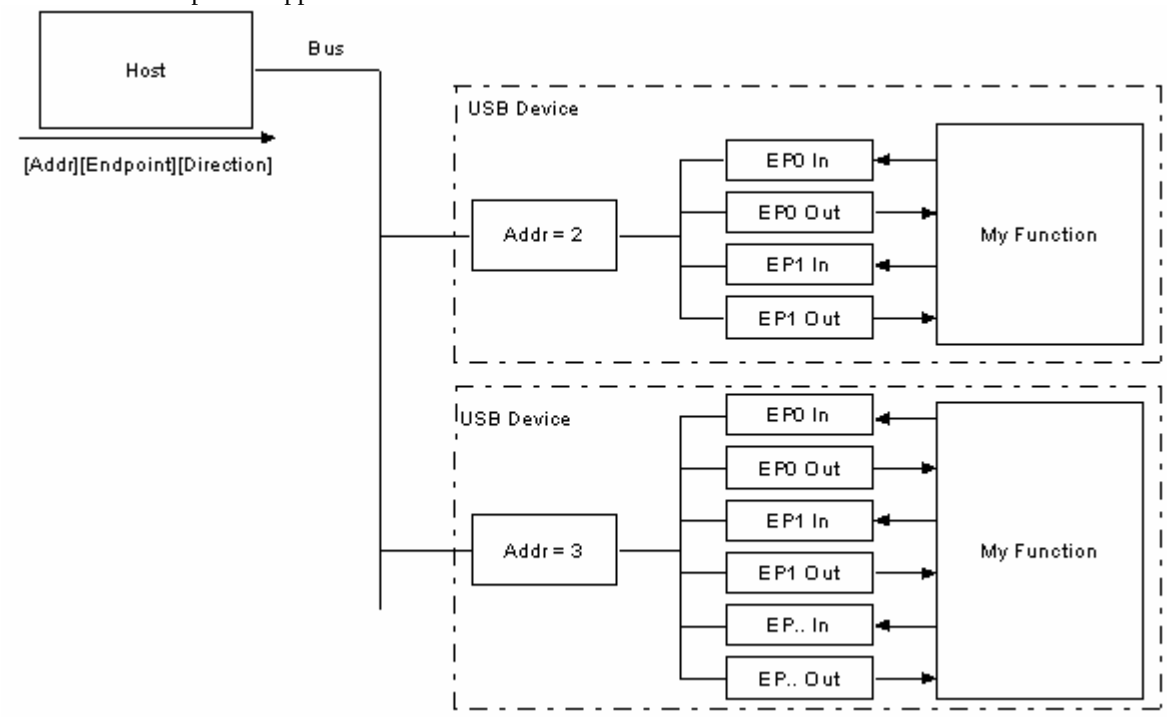
Le paquet SOF composé d'une trame de 11 bits est envoyé par l'hôte toutes les 1ms ± 500ns sur un bus pleine vitesse ou bien toutes les 125µs ± 0,0625µs sur un bus haute vitesse.

Sync	PID	Frame Number	CRC5	EOP

3.5.Fonctions USB

La plupart des fonctions USB manipulent les protocoles USB. Ils ont des séries de tampons (buffers), généralement de 8 octets de long. Chaque tampon appartiendra à une terminaison EP0 IN, EP0 OUT etc... Supposons par exemple, que l'hôte envoie une demande de descripteur d'appareil. La fonction matérielle lira le paquet d'installation et déterminera à partir du champ adresse si le paquet est pour lui-même, et si c'est le cas, il copiera la « charge utile » du paquet de données suivant au tampon de la terminaison appropriée, dictée par la valeur dans le champ de la terminaison du jeton d'installation. Il enverra ensuite un paquet « poignée de mains » pour valider la réception de l'octet et générera une interruption interne à l'intérieur du semi-conducteur / micro contrôleur pour la terminaison appropriée signalant qu'il a reçu un paquet. C'est en principe déjà intégré dans la matière (silicium).

Le logiciel a maintenant une interruption, et doit lire le contenu du tampon de terminaison et analyser la demande de descripteur d'appareil.



3.6.Terminaisons

Les terminaisons peuvent être décrites comme émetteurs ou récepteurs de données. Du fait que le Bus est régi par l'hôte, les terminaisons se présentent à la fin de la chaîne de communications sur la fonction USB. Au niveau de la couche logicielle, le pilote (driver) logiciel de votre appareil va envoyer, par exemple, un paquet à vos appareils EP1. A la sortie de l'hôte, la donnée aboutira au tampon EP1 OUT. Votre microprogramme pourra alors lire à loisir cette donnée. S'il veut retourner la donnée, la fonction ne peut pas simplement écrire sur le BUS comme celui-ci est contrôlé par l'hôte. Par conséquent il écrit la donnée dans EP1 IN qui s'installe dans le tampon jusqu'à ce que l'hôte envoie un paquet IN à cette terminaison demandant la donnée. Les terminaisons peuvent être aussi considérées comme l'interface entre le matériel de l'appareil de fonction et le microprogramme s'exécutant sur ce même appareil.

4.Bus Firewire

FireWire est le nom marketing donné par Apple au bus série IEEE P 1394 dont il est à l'origine. Le nom « grand public » est IEEE 1394. Cette interface série a été mise au point pour gérer les connexions des ordinateurs multimédias et des périphériques associés. L'interface IEEE 1394 (ou IEEE 1394 A) permet, par exemple, de transmettre des séquences vidéo numériques (issues d'une caméra vidéo numérique (DV), d'un magnétoscope ou encore d'une télévision numérique HDTV) directement à un ordinateur.

Adaptée pour le transferts de données isochrones (en temps réel), l'interface IEEE 1394 pourra aussi être mise en œuvre dans les imprimantes, les scanners et les unités de stockage (disque durs notamment). Les transferts isochrones garantissent un taux minimum de transfert constant, ce qui est particulièrement intéressant pour les applications multimédias. Un grand nombre de constructeurs c'est rallié à cette norme.

Le bus IEEE 1394 est un bus destiné aux périphériques nécessitant des débits soutenus très élevés, comme le traitement de la vidéo, par exemple. I1 est actuellement décliné en quatre versions qui atteignent respectivement 100 Mbits/s, 200 Mbits/s et 400 Mbits/s. C'est à dire 12.5, 25 et 50 Mo/s (Million d'octets/s). Remarque 400 Mbits/s = 400 000 000 /8 ~ 50 Mo/s, soit un peu moins que la meilleur interface SCSI actuelle (Ultra 2 SCSI avec 80 Mo/s). Le bus FireWire offre également la possibilité de relier entre eux jusqu'à 63 périphériques, sans hub, et d'une façon réellement Plug and Play, un peu à la manière de l 'USB. Ainsi, il est possible d'ajouter ou de retirer un périphérique à chaud (notion de hot plugging), sans qu'il soit nécessaire de redémarrer le PC ou de configurer quoi que ce soit.

De nouvelles spécifications du bus IEEE 1394 B et IEEE 1394.1 sorties en février 2000, on permis de porter les débits à 800 Mbits/s, 1.6 Gbits/s et 3.2 Gbits/s soit 100 Mo/s, 200 Mo/s et 400 Mo/s. Ces nouvelles spécifications intègres la communication avec le bus IEEE-488 et la gestion des cartes d'Entrées/Sorties pour l'instrumentation industrielle. La version IEEE 1394.1 correspond à une évolution vers le réseau. C'est à dire que le bus se transforme et est géré comme un réseau.

4.1.Connecteurs

Le bus IEEE 1394 suit à peu près la même structure que le bus USB. Si ce n'est qu'il utilise un câble composé de six fils (deux paires pour les données et pour l'horloge, et deux fils pour l'alimentation électrique) lui permettant d'obtenir un débit de 400 Mbps (il devrait atteindre prochainement 1 Gbps). Ainsi, les deux fils dédiés à une horloge montrent la différence majeure qui existe entre le bus USB et le bus IEEE 1394 :ce dernier peut fonctionner selon deux modes de transfert :

- le mode de transfert asynchrone
- le mode isochrone

4.2.Protocoles

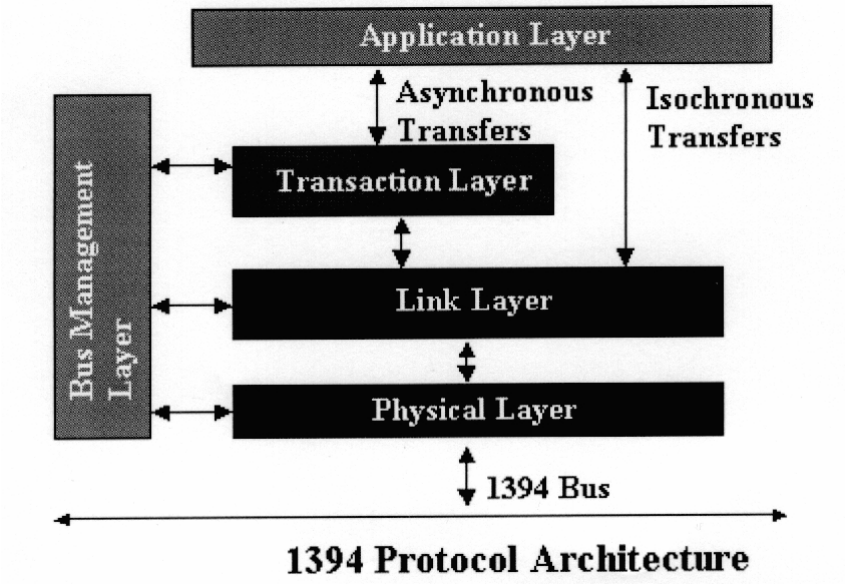
IEEE 1394 utilise une technologie de transmission de donnée par paquets qui est organisée comme un espace mémoire interconnecté entre plusieurs appareils. La structure d'un réseau 1394 est constitué de deux couches : une couche de transaction et une couche de gestion du bus série. Ces couches logicielles peuvent être directement intégrées en « hard ». La couche physique doit gérer le signal de mise sous tension à distance, la reconnaissance du signal de sélection de l'appareil, le signal d'initialisation du bus et la réception/émission des données. La couche de lien formate les données en paquets pour la transmission via le câble 1394 et supporte les modes de communications asynchrone et isochrone.

La couche physique supporte les différentes vitesse de transmission suivant le matériel la composant. La couche de lien définit deux modes de communication :asynchrone et isochrone.

Le mode de transmission asynchrone garanti la bonne réception des données par accusé de réception. Ce temps de latence ne peut être quantifier car il dépend de taux d'utilisation du bus 1394 par d'autres transmissions pour d'autres appareils communicants entre eux. Ce paquet de données peut être envoyé à une adresse d'un appareil connecté au réseau ou à toutes les adresses.

Le mode de transmission isochrone est différent. Il réserve, pour la transmission, un espace-temps de dimension particulière et cyclique, toutes les 125 µs. Depuis un appareil, un espace-temps isochrone est garanti.

Les communications isochrones sont prioritaires aux asynchrones de sorte que la bande passante pour les communications isochrones est assurée. Ainsi, la communication isochrone entre deux appareils ou plus est assimilable à un canal. Une fois qu'un canal a été établit, l'appareil demandeur est garanti d'avoir l'espace-temps demandé à chaque cycle. C'est le mode de transmission que l'on choisit pour le transport de données vidéo ou toutes autres données qui ont besoin d'avoir une transmission garantie en « temps réel ».



1394 Protocol Architecture

VIA fournit à l'heure actuelle une solution single chip pour l'IEEE 1394, le VIA Fire II. VIA proposera d'ici quelques mois des chipsets intégrant le FireWire.

Une nouvelle version de l'IEEE 1394-1995 vient de voir le jour :l'IEEE 1394b, un petit « b » pas si négligeable que cela. Explications : prévu pour concurrencer le FireWire, l´USB2 proposent des caractéristiques attrayantes. Destiné pourtant à un marché un peu différent, l'USB 2.0 pourrait faire de l'ombre au FireWire, qui dispose d'une implantation inférieure à l'USB dans le monde de la micro-informatique. Dans sa nouvelle déclinaison, l'IEEE 1394 propose des caractéristiques et fonctionnalités encore plus avancées que son concurrent et son prédécesseur :

- Bande passante améliorée, passant ainsi de 400 Mbits/s à 800 Mbits/s. Cette vitesse devrait être amenée rapidement à 1.6 Gbits/s puis 3.2 Gbits/s grâce à l'utilisation de fibre optique plastique multimodes. Dans cette configuration, les câbles reliant deux appareils pourront dépasser les 100 mètres ! Il sera néanmoins possible d'atteindre des raccordements de 100 mètres avec des câbles UTP catégorie 5 en 100 Mbits/s.
- Nouvelle implémentation du protocole visant à améliorer la disponibilité de la bande passante grâce à une implémentation BOSS du protocole d'adressage.
- Compatibilité avec les périphériques IEEE 1394-1995 par un mode bilingue.
- Nouvelles applications dans le monde de l'automobile et de l'équipement audio-visuel.
- Réduction des coûts de fabrication des chips et de leur intégration dans les circuits analogiques

5.Conclusion

Les anciennes entrées/sorties de nos micro-ordinateurs ont largement profité des développements rapide de ces dernières années. Les ports série et parallèle déjà présent sur les premiers PC commencent à être à bout de souffle et ne répondent plus vraiment aux besoins des utilisateurs.

Deux principaux bus se disputent désormais le marché, l´USB et l´IEEE-1394. Dans leur dernière déclinaison, ces deux interfaces proposent toutes les deux des caractéristiques très intéressantes.

Relativement lent dans sa première version, l'USB était jusqu'alors limité à des périphériques lents tels que les claviers, imprimantes, souris ou webcam. Avec sa version 2.0, il pourra investir de nouveaux domaines qui étaient jusqu'alors réservés à l'IEEE 1394 ou au SCSI, tels que les périphériques de stockage externe ou encore les caméscopes DV.

Reste que l'IEEE 1394 restera certainement l'interface privilégiée des périphériques très gourmand, notamment en ce qui concerne l'audio vidéo (caméscopes, TV numériques, DVD). En effet, il offre d'une part une bande passante plus importante dans sa dernière évolution, et est plus apte à gérer plusieurs périphériques à haut débit.

D'autres périphériques auront le choix entre les deux interfaces, permettant ainsi une orientation privilégiant tantôt l'agressivité du prix, tantôt les performances. Les constructeurs voient donc dans l'avenir la coexistence de ces deux bus, même s'il y' a fort à parier que l'argument prix pèse le plus dans la balance, comme ce fut le cas entre IDE et SCSI. Wait & See !