Support de cours de Transmission et Acquisition de Données

Responsable du cours :
Eric Magarotto

Pour toute remarque ou question relative au cours :
Laboratoire d'Automatique & de Procédés (LAP-ISMRA)
e-mail : eric.magarotto@greyc.ismra.fr
web : www.greyc.ismra.fr/EquipeAuto/EricM/EMWelcome.html
tel : 02-31-45-27-09

Objectifs du Cours

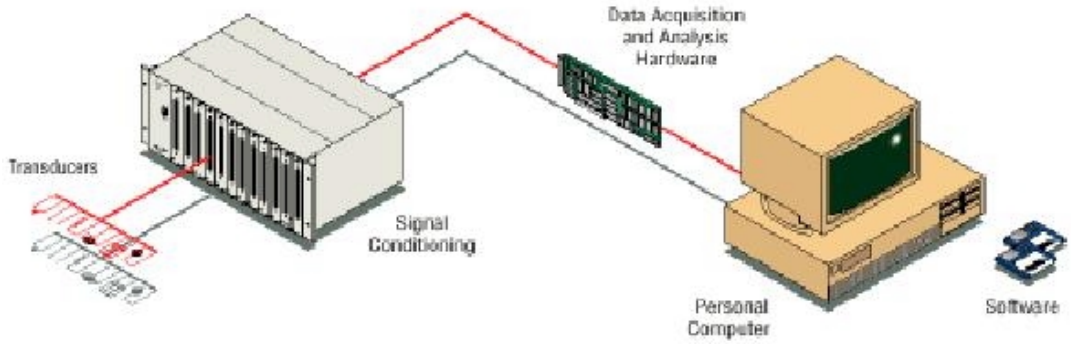
Le but est d'étudier les dispositifs d'échanges de données. Cela passe par une compréhension des différents composants responsables des transmissions de données. On étudiera les communications entre les ordinateurs et leurs composants ou bien entre un ordinateur et des périphériques. Après un bref rappel d'architecture générale (carte mère et bus), nous préciserons les divers types de liaisons, les contrôleurs d'interfaces associés (UART, SCSI, USB et IEEE1394) et les protocoles qui régissent ces communications. Nous terminerons ce cours par un exemple d'étude de carte d'acquisition des données (DAQ), celle-ci étant utilisée pour les séances de TP d'Informatique et d'Automatique.

Version provisoire du 15 décembre 2003

Table des Matières

I. ARCHITECTURE ET NOTIONS DE TRANSMISSION DE DONNEES

Un système d'acquisition de données peut être représenté par la figure suivante :

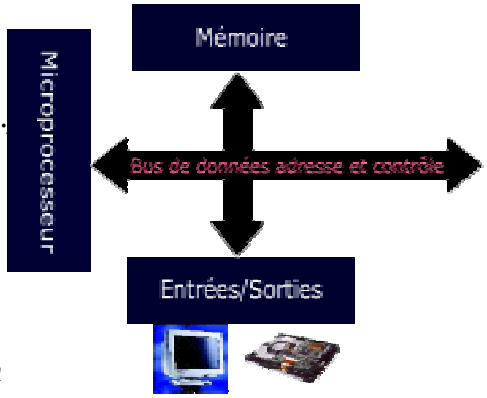


A une extrémité de la chaîne, on trouve les capteurs, puis le conditionnement (mise en forme) du signal puis la carte d'acquisition des données proprement dite et enfin l'ordinateur et la partie logicielle. Chaque étape nécessite une connaissance des parties hardware et software ainsi que de bonnes notions des règles de communications. C'est la raison pour laquelle nous débuterons ce cours par un bref rappel d'architecture, en particulier sur le rôle des divers éléments d'une carte mère ainsi que les relations entre le micro-processeur et les interface d'Entrées/Sorties par le biais des bus. Nous étudierons ensuite les divers protocoles de communication pour les différents types de liaisons, de communications et de bus. Enfin, nous terminerons par un exemple d'étude d'une carte d'acquisition temps réel des données.

I.1 - Généralités sur les Entrées / Sorties

Ordinateurs : Traitement et stockage de l'information
*E/S : **échange** d'informations avec son environnement*

- *extérieur vers PC : acquisition (clavier, réseau, disque dur, etc…*
- *PC vers extérieur : écran, disque, réseau, imprimante, etc…*



*Important pour les performances du PC : même si le processeur est très rapide, les performances peuvent se dégrader **considérablement** lors des opérations de lecture/écriture : dépendent qualité des divers composants « fond de panier », carte-mère et bus (entre autres).*
Les échanges d'informations se font entre la mémoire principale et un périphérique relié à l'ordinateur.
*Cet échange nécessite une **interface** ou contrôleur, circuit électrique gérant la connection. L'interface réalise des fonctions plus ou moins complexe selon le type de périphérique. Son but est **surtout** de décharger le processeur pour améliorer les performances du système.*

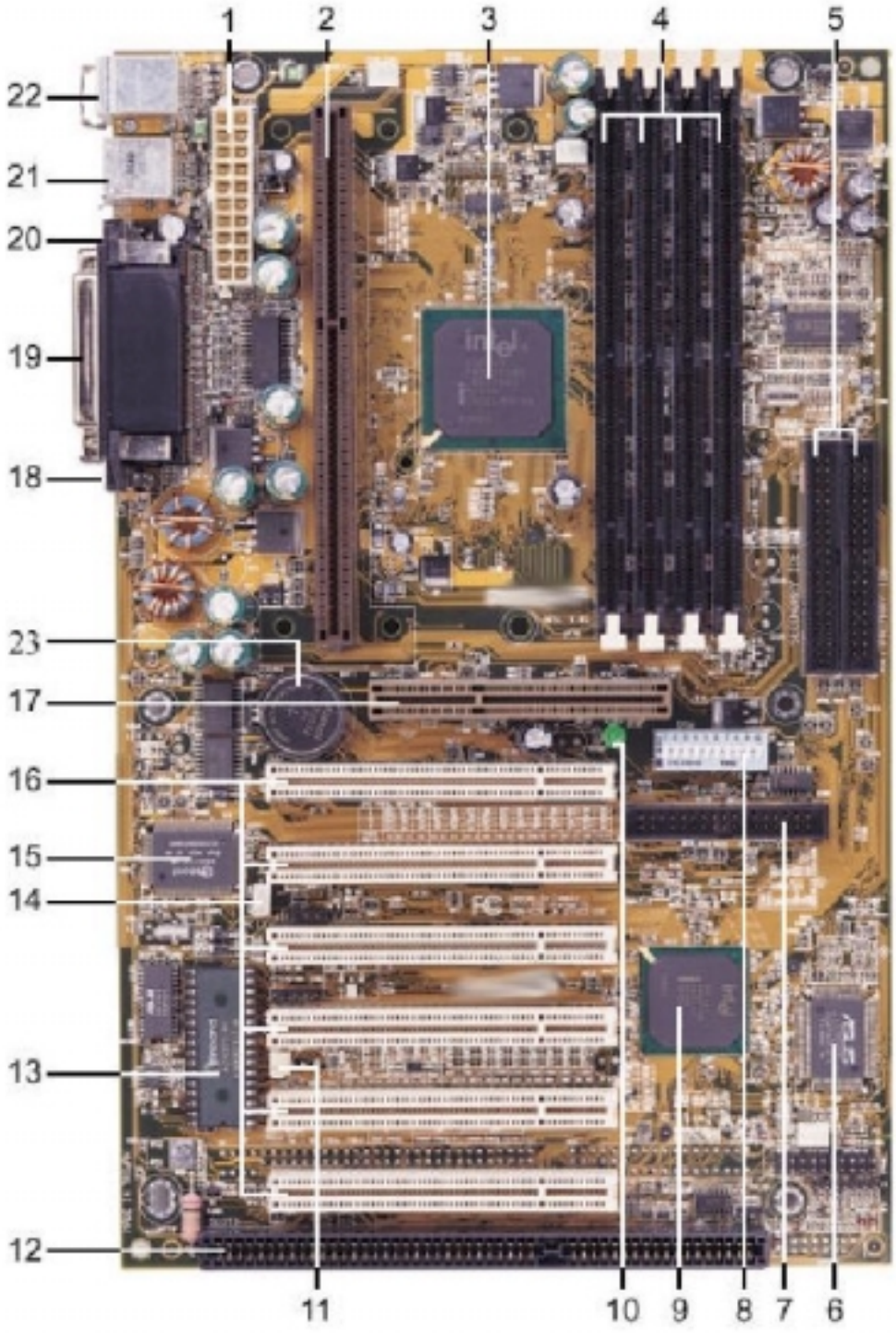
I.2 - Un peu d'Architecture des µ-ordinateurs…Carte-Mère

☞ **: performances d'un ordinateur**

*Elles sont liées à **l'ensemble** des composants, de leur compatibilité et de leur niveau d'aptitude à fonctionner correctement **ensemble**. Les 3 éléments essentiels sont le calcul (processeur-coprocesseur arithmétique), les mémoires (RAM, graphique, etc...), les éléments de communication (interfaces E/S, ports, bus internes-externes). Il est donc préférable d'avoir une bonne carte mère avec un bus rapide et large, un Disque Dur rapide et une bonne carte graphique avec un processeur moyen, plutôt que le contraire…*

☞ **: carte mère**

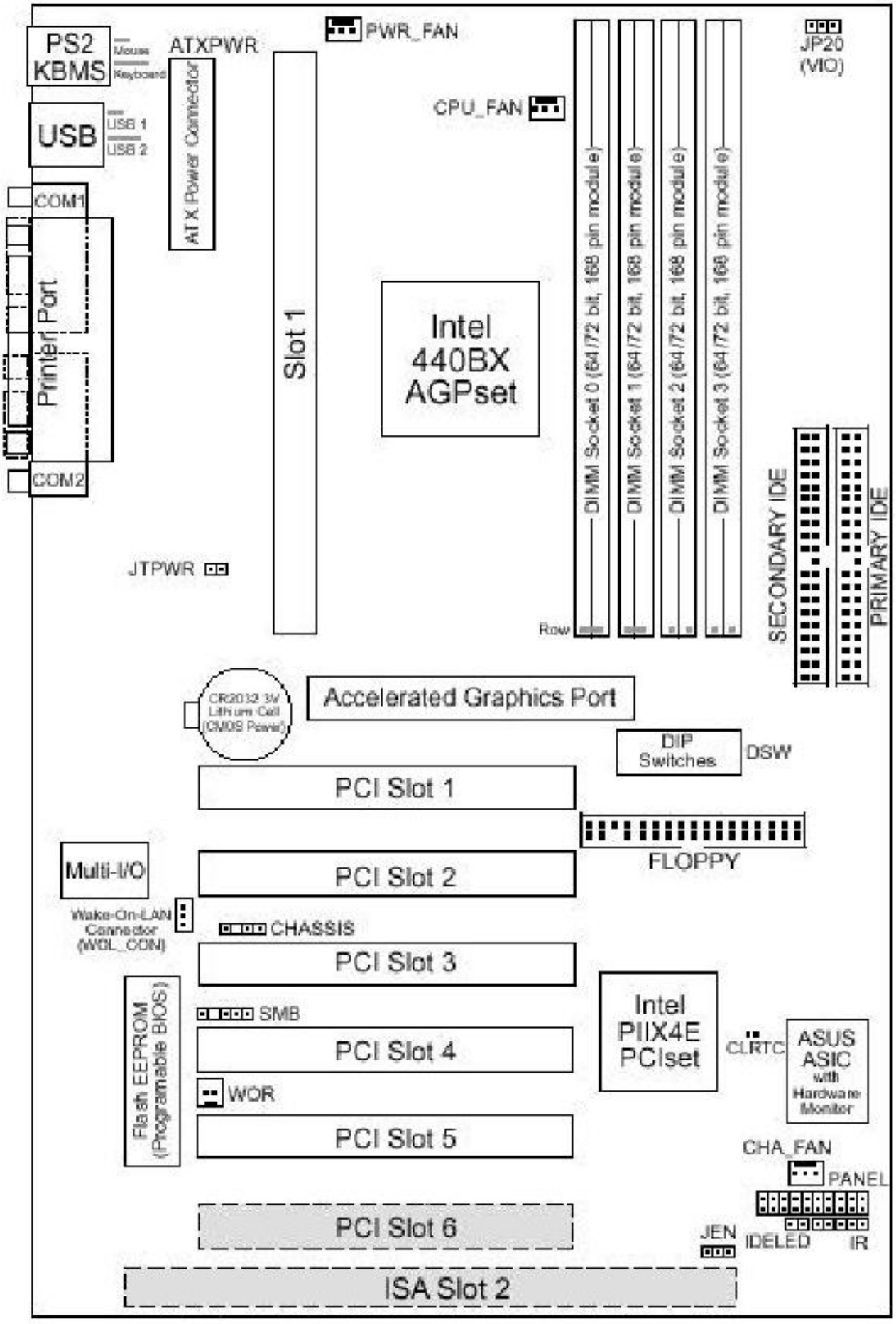
La carte mère est le composant de l'ordinateur qui dirige et organise le fonctionnement de tous les autres composants. C'est le donneur d'ordre pour le processeur.Elle est composée de nombreux bus ISA, PCI, AGP, des ports série, parallèle, USB, IEEE 1394, IDE, clavier et souris, de la mémoire (principale et cache niveau 2), du BIOS, du processeur et des contrôleurs de bus (chipset) ainsi que de divers connecteurs (voir exemple carte ATX ci-dessous).



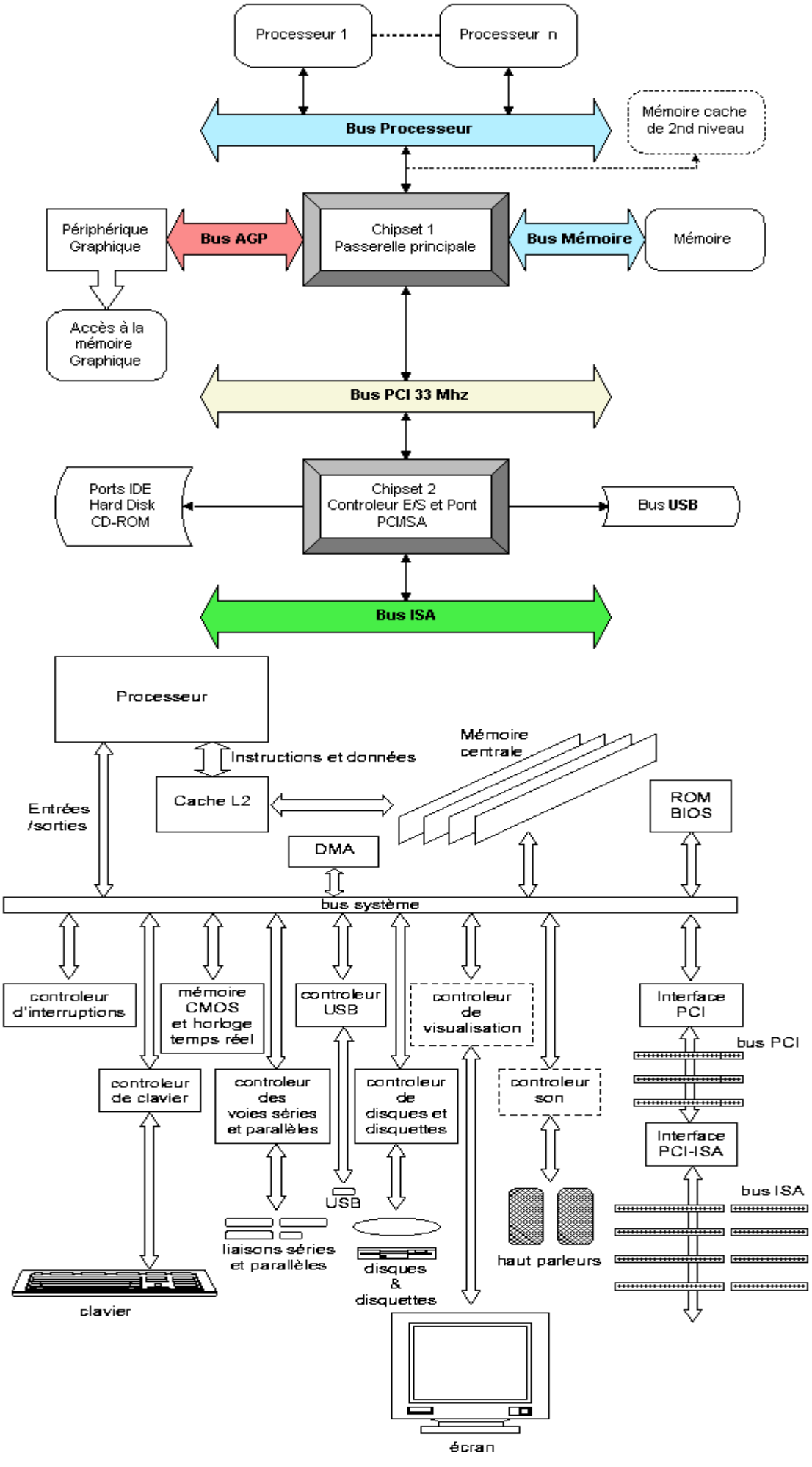
Carte ATX

1. connecteur d'alimentation
2. support processeur « slot 1 »
3. Intel 440 BX AGP set
4. 4 connecteurs DIMM (RAM)
5. 2 connecteurs IDE
6. Puce de contrôle tensions/temp.
7. Connecteur disquette
8. DIP switches
9. Intel PIIx4e PCI set
10.LED carte mère
11.Wake On Ring (reveil par modem)
12.port ISA
13.ROM Flash (BIOS)
14.Wake On LAN (réveil par réseau)
15.Puce I/O
16.6 ports PCI
17.Port AGP
18.connecteur Port série (COM2)
19.connecteur Port //
20.connecteur Port série (COM1)
21.2 connecteurs USB
22.Port PS/2 souris & clavier
23.Pile

Sous forme de schéma :



Structure d'un PC – relations intérieur/exterieur :



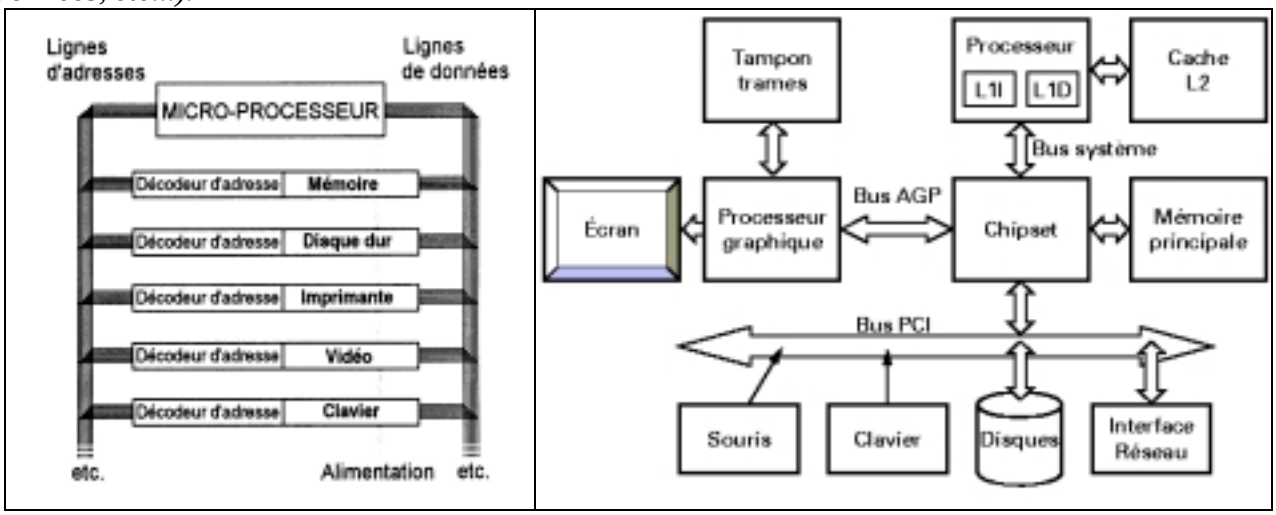
I.3 - Les Bus du PC

Tout système informatique est constitué d'une carte mère. Sur ces cartes mères, outre les composants principaux tels que le processeur, le BIOS, la RAM ou la mémoire cache, il faut absolument que ceux ci puissent communiquer entre eux. C'est le rôle des ***BUS****.*

Définition [BUS] *: on désigne par BUS un conducteur servant de canal de transmission commun entre plusieurs circuits qui peuvent s'y connecter à la demande, soit en tant qu'émetteurs, soit en tant que récepteur ou les deux. Dans le cas ou plusieurs organes communiquent, il faudra gérer (avec des règles) cette communication. En plus d'avoir une architecture et une connectique particulière, il est nécessaire d'y associer un protocole d'échange.*

Un BUS est constitué de plusieurs lignes :

- *ligne de données : liaison bidirectionnelle qui assure le transfert des informations entre un élément et un autre.*
- *ligne d'adresses : liaison bidirectionnelle qui permet la sélection des informations à traiter dans un emplacement mémoire qui peut avoir 2^n emplacements.*
- *lignes de commandes : liaison pour assurer la synchronisation des flux d'informations sur les bus de données et d'adresses. Les signaux de commandes que l'on peut rencontrer sont l'horloge (« clock »), les signaux de demandes d'interruption et d'accord (« acknowledge »), les signaux d'arbitrage des échanges, le contrôle des échanges (read/write, type de transfert, types des données, etc...).*



Un BUS est caractérisé par sa largeur (nombre de bits) et sa fréquence (nombre de cycles/s) qui permettent de déterminer sa bande passante et son taux de transfert maximal théorique (en Mo) de la manière suivante.

exemple *: si fréquence BUS_{PCI} = 66 Mhz avec largeur 32 bits alors : TxT = ((66*32)/8) ≈ 264 Mo/s*

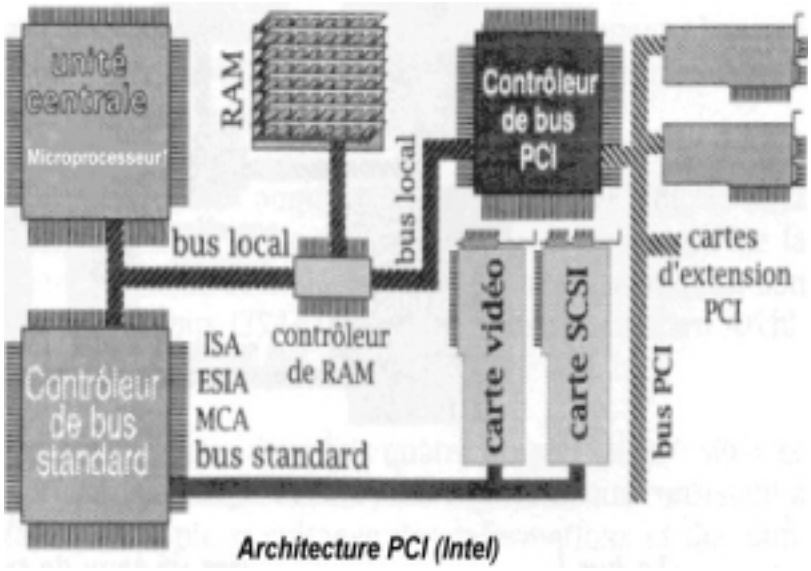
Il existe un grand nombre de bus. Une classification possible est de différencier les bus en fonction des équipements qui s'y connectent (« system », « backplane », « peripheral »). Nous n'étudierons pas les bus processeur et mémoire. Dans le cadre de la compréhension des échanges, nous nous concentrerons sur les bus d'E/S (ou connecteurs d'extensions).

☞ : bus local (« system », « backplane ») : ISA, PCI

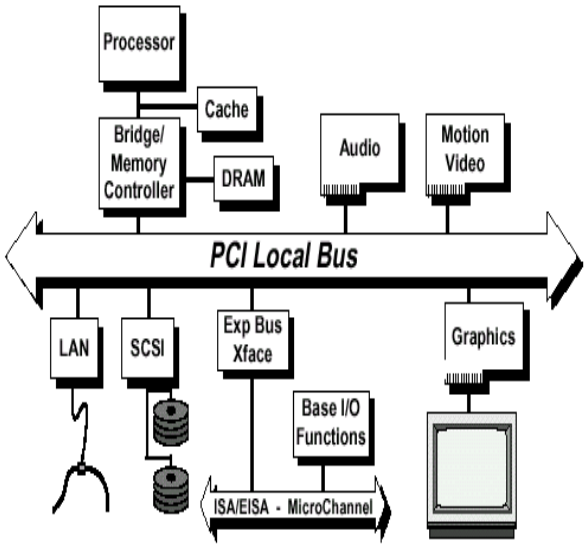
La transmission s'effectue en ***parallèle****. C'est le plus rapide. En effet, c'est celui sur lequel sont directement connectés le processeur et la mémoire (exemple VME, MultiBus). Il regroupe un bus d'adresses, un bus de données et un bus de commande. Il est aussi reliés aux contrôleurs de bus d'extensions. Standardisation récente : connexion d'extensions sur la carte mère. De nombreux bus sont utilisés sur un groupe de connecteurs (slots) pouvant recevoir des cartes : ISA, PCI*

- **Bus ISA** *(Industry Standard Architecture) : tombé en désuétude (jusqu'au P3). de fréquence relativement basse et de caractéristique peu puissante, il était utilisé pour connecter des cartes "lentes" (modems, carte sons).*

- **Bus PCI** *(Peripheral Component Interface) : bus parallèle à TxT plus important que ISA (132 Mo/s). La nouvelle génération (PCI Express pour 2003) est un bus série capable de transférer des données sur 32 voies, chacune d'elle assurant un débit de 200 Mo/s (au total : 6 Go/s). Avec le PCI, on dissocie le processeur des bus. Fréquence de bus différente et décharge du processeur. A l'origine, couplé avec l'ISA, il était plutôt utilisé pour les cartes hauts débits (carte vidéo, SCSI, réseaux hauts débits). C'est un bus* ***synchrone*** *supportant un multiplexage des signaux d'adressages et de données. Il est avant tout prévu pour travailler avec des systèmes 32 bits, dans un fonctionnement en échange entre un* ***maître*** *et un* ***esclave*** *sous le contrôle d'un* ***arbitre****. L'interface PCI prévoit l'utilisation d'un minimum de signaux avec un protocole décorélé des protocoles de transferts des microprocesseurs existants. D'après la spécification du bus, un minimum de 49 signaux est nécessaire pour un module capable d'assurer des fonctions de maître, tandis que les autres signaux normalisés peuvent n'être que des options.*



Architecture PCI (Intel)



☞ : bus système série : I2C, SPI

Les transmissions système de type série sont plus rare mais elles permettent de relier entre eux un certain nombre de circuits de façon simple (circuit imprimé simplifié : appareil économique). Très utilisés dans les systèmes industriels pour lesquels le temps de traitement n'est pas critique.

☞ : bus de périphériques (internes) : IDE, EIDE, AGP, SCSI

Ces bus permettent de relier une interface (contrôleur) de l'ordinateur à des périphériques. Les liaisons sont essentiellement parallèles et permettent l'échange d'information avec les disques durs conformes aux spécifications ATAPI (AT Attachment Paquet Interface). Le standard SCSI (Small Computer System Interface) est une interface qui nécessite une carte (adaptateur SCSI). Il existe deux types de bus SCSI : // asymétrique et // différentiel (attention aux signaux électriques différents). Aujourd'hui on trouve le Wide SCSI-2 (10 Mbit/s en standard et 20 en Fast SCSI, voire 80 en Fast 40). Le SCSI-3 permet le chaînage de 32 périphériques. Nous ferons une étude plus détaillée dans un prochain cours

☞ : bus de périphériques externes et ports d' E/S

Ils permettent les échanges entre un système et les périphériques externes par le biais de connecteurs (ports). Ces échanges peuvent prendre la forme d'une liaison par câble avec connecteurs appropriés. Il existe aussi des liaisons sans câble (infrarouge,protocole RC5)

- **Port // (Centronics)** *: Conçu à l'origine pour liaisons en sortie vers une imprimante, il est utilisé pour bien d'autre choses, aussi bien en sortie (usage standard) qu'en entrée-sortie (les données d'entrée transitent via les lignes de contrôle).*
- **Port Série** ***(RS 232 et UART 8250-16550)*** *: Utilisé pour connecter une grande variété de périphériques (imprimante série, souris, appareils industriels, etc…). Il peut être configuré de différentes manières, tant en ce qui concerne la rapidité des échanges, qu'en ce qui concerne le format des mots transmis ou le contrôle de la parité. Ceci nécessite de la part de l'utilisateur une bonne connaissance des réglages de l'appareil connecté. Le Port RS 232 est destiné aux liaisons entre 2 appareils. Ce n'est donc pas exactement un bus (bus sous entend une information distribuée à un*

certain nombre d'hôtes). Les informations sont transmises en références commune, ce qui en limite la portée et la sécurité. On y remédie avec un interfaçage particulier (RS485, réseaux locaux)

- **USB 1.1 et 2.0** *: Il est amené à remplacer toutes les différentes sortes de connexions d'entrée-sortie sur les PC. On peut y connecter tous les périphériques et en grand nombre (127) en utilisant un "hub". Il est même amené à transmettre les signaux vidéo. Il possède un certain confort d'utilisation (branchement à chaud : hot Plug & Play). Le système reconnaît le périphérique, adapte son mode de transmission, lui attribue une adresse (énumération) et l'alimente si sa puissance nécessaire est faible. La liaison repose essentiellement sur une paire différentielle mode série (câble 4 fils). Deux vitesses permettent de concilier le nombre d'hôtes et la rapidité de transfert pour les seuls périphériques qui en ont besoin (détection comme lent ou rapide). L'USB 1.1 est limité (1.5 Mbits/s en lent, 12 Mbit/s en rapide) tandis que l'USB 2 annonce un débit max de 480 Mbits/s. Simple mécaniquement, il ne l'est pas du point de vue électronique et nécessite une configuration précise. La communication se fait selon un protocole "token ring" (principe de l'anneau à jeton) plus complexe. La tendance est fortement orienté réseau.*
- **IEEE1394 (a et b)** *:* **Firewire et I-Link** *: A l'origine , la solution Apple. Il possède à peu près la même structure que l'USB bien que plus ancien. Il utilise un câble de 6 fils (2 paires pour les données et l'horloge, et deux fils pour l'alimentation) lui permettant d'obtenir un débit de 400 Mbit/s ou 800 Mbits/s (voire bientôt 1.6 et 3.2 Gbit/s vers fin 2003). Grâce à ses 2 fils d'horloge, le Firewire peut fonctionner en mode* ***asynchrone*** *ou bien* ***isochrone****. Possibilité d'utiliser des ponts (plusieurs bus entre eux, adressage par identificateur de nœud sur 16 bits) : 65 535 périphériques au max ! La bande passante est plus élevée que pour l'USB : acquisition vidéo.*

☞ : bus d'instrumentation GPIB, HPIB : (IEEE 488)

Ce bus permet la liaison d'équipements de mesure. Le GPIB (General Purpose Interface Bus) ou HPIB (Hewlet Packard Interface Bus) est formé de lignes de transmission de données en parallèle et de lignes de contrôle des échanges (transmissions sur des distances réduites : une pièce). En utilisant le protocole de norme IEEE488, ces bus devait équiper tous les appareils connectable sur réseau d'instrumentation. On leur préférera une interface standard série, voire une carte réseau.

☞ : autres bus : PC Card, CAN

Le bus PCCard (anciennement PCMCIA pour Personal Computer Memory Card Association) est un bus d'extension utilisé sur des ordinateurs portables. Il permet la connexion de périphériques de taille très réduite (format carte bancaire) : très utile pour les systèmes embarqués.
Initialement VAN (développé par Renault PSA-SAGEM-Valéo) et CAN (développé par Bosch) ces bus était destiné à l'équipement automobile. Aujourd'hui, le bus CAN (Controller Area Network) se situe entre bus et réseau (transmission par câble, infrarouge ou fibre optique) et représente avant tout un protocole de transmission.

I.4 - Notions principales sur les E / S et les liaisons

☞ : principes

Les communications se font par signaux

- *via un* ***port*** *: point de connexion*
- *sur un* ***bus*** *: contient plusieurs cables et se définit par le protocole (règles) de comunication.*
- *grâce à un* ***contrôleur*** *(4 registres : états, commande, data in & out)*

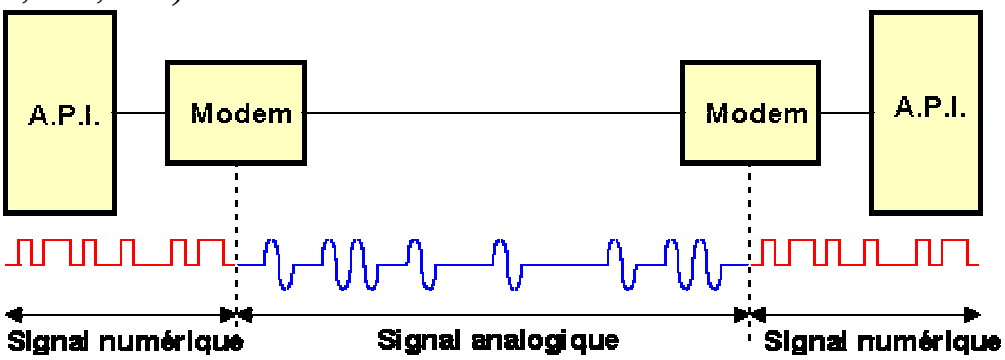
Les données échangés entre un périphérique et le processeur transitent par l'interface (contrôleur) associé à ce périphérique. L'interface possède de la mémoire tampon pour stocker les données échangées. Le contrôleur stocke aussi les informations nécessaire à la gestion de la communication :

- ***informations de commande*** *: définition du mode de fonctionnement (sens de transfert : entrée ou sortie), mode de transfert (scrutation ou interruption), etc…Ces informations sont transmises à l'interface avant le début du transfert (phase d'initialisation)*
- ***informations d'état*** *: mémorisation de la manière dont le transfert s'est effectué (erreur de transmission, réception d'information, etc…). ces informations sont destinées au processeur.*

On accède aux données des interfaces par le biais d'un espace d'adresse d'E/S.

☞ : généralités sur la transmission

Dans le cadre d'échange entre plusieurs ordinateurs (réseau) ou bien entre ordinateurs et périphériques externes, il faut tout d'abord choisir le codage de l'information a transmettre. Le type de données peut être de nature sonore, texte, graphique etc..
La représentation adoptée peut être analogique (variation d'un grandeur physique) ou numérique (codage binaire, NRZ, etc...).



*La transmission est cactérisé par le **sens** des échanges, le **mode** et la **synchronisation** (entre émetteur et récepteur)*

☞ : rapidité, taux des transferts

Il existe 2 unités pour qualifier la rapidité des échanges :

- *Bauds : nombre de bits de données transmis par seconde.*
- *Bits/sec : nombres de bits (quelconques) transmis par seconde.*

La vitesse de transfert effective est calculée sur les données (on ne tient pas compte des bits de start et de stop pour une communication asynchrone, et des bits de synchronisation pour une communication synchrone).
***Exemple** : si on utilise 4 niveaux d'amplitude électrique un évenement (00=-15 V, 01=-5 V, 10=+5 V, 11=+15 V), dans ce cas le transferten bit/s est double de celui en bauds.*

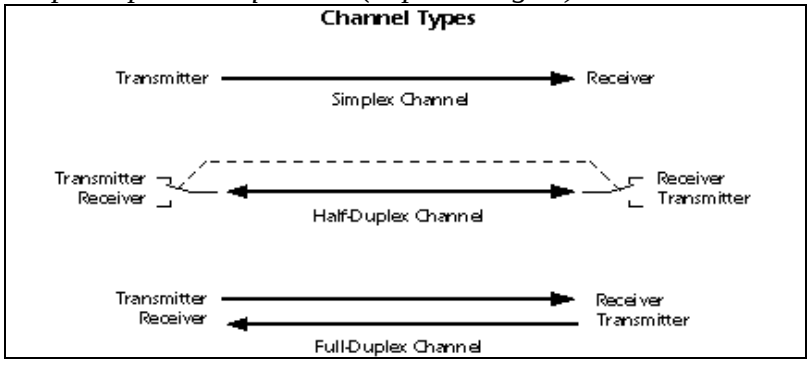
Les techniques pour augmenter la rapidité d'un bus consistent (principalement) à :

- *augmenter la largeur de bus (+ de place physique, + de bruit, + de courant)*
- *augmenter la vitesse d'horloge (tous les périphériques ne tiennent pas forcément cette cadence)*
- *découper les transactions en paquets de données (« split transaction »)*
- *modifier les amplitudes des signaux électriques*

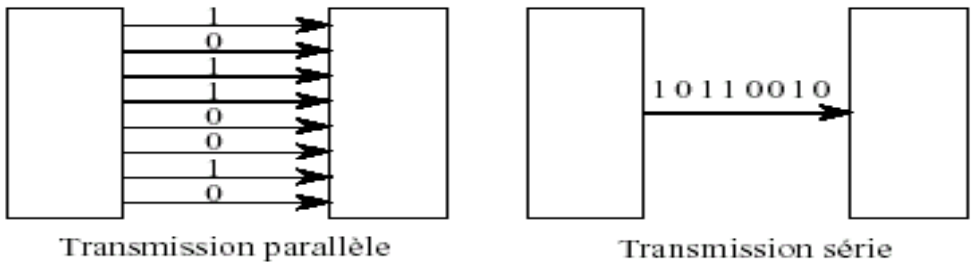
I.5 - Modes de transmissions

☞ : simplex / duplex

- **simplex** : *Les données circulent dans <u>un seul sens</u> : émetteur vers récepteur (ex : ordinateur → imprimante, souris → ordinateur, radio).*
- **half-duplex** : *les données circulent dans les 2 sens mais <u>pas simultanément</u> : la bande passante est utilisée en intégralité (aussi appelé alternat ou semi-duplex). Exemple : talkie/walkies, êtres humains (on ne coupe pas la parole).*
- **full-duplex** *: les données circulent de manière <u>bidirectionnelle</u> et <u>simultanément</u> : la bande passante est divisée par 2 pour chaque sens (duplex intégral).*



☞ : liaisons série / parallèle

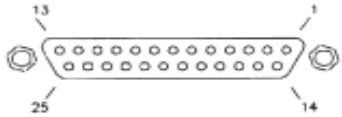


• Transmission simultanée de 8 bits sur 8 voies différentes (rapide) • fils proches → perturbations importantes à haut débit • pb de place (beaucoup fils)	• données envoyées bit par bit (+ lente) • données en série pour émission / réception (UART registres à décalage) • avantage : grande distance, universelle (beaucoup applications)

☞ : ports série / parallèle

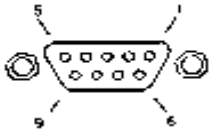
- **<u>port parallèle</u>** *: L'usage normal consiste à sortir 8 lignes (1 octet) de données sous forme parallèle et de contrôler les échanges au moyen de 5 lignes en entrée (Busy, _Ack, PE, Slct, Error) et 4 en sortie (_Slctin, _init, _afdx, _strobe). Le port // à subi des évolutions qui lui ont permis de considérer (sous sa forme actuelle) les lignes de données bidirectionnelles (réunion du port EPP et ECP).*

connecteur DB25 (port // et/ou série) :



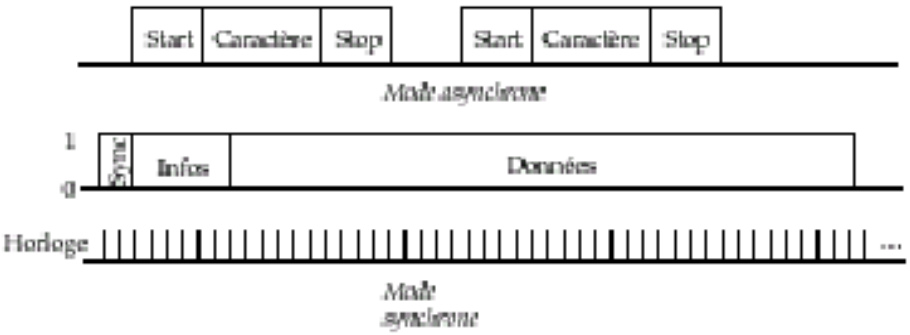
- **<u>port série</u> :** *L'échange de donnée se fait par ligne unique. Les bits sont donc envoyés à la suite. Les ports séries actuels sont bidirectionnels (2 lignes, une par sens de communication). Très utilisé : téléphone.*

connecteur DB9 (port série) :



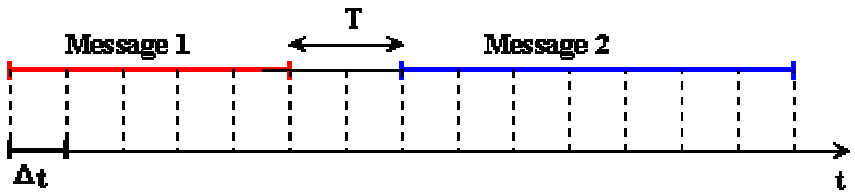
☞ : synchrone / asynchrone

La liaison série est la plus utilisée. Un seul fil transporte l'information → problème de synchronisation entre émetteur et récepteur (distinguer et reconnaître les séquences de bits utiles).



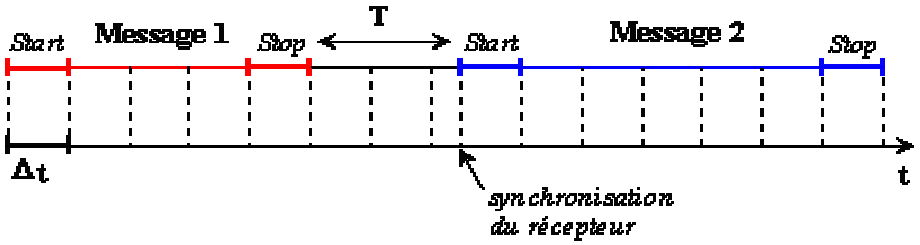
Deux types de transmission remédient à ce problème :

- **synchrone :** *l'émetteur et le récepteur sont cadencés à la même fréquence d'horloge (incorporé au bus ou bien aux données). L'horloge de reception et d'émission doivent émettre le même signal d'horloge (pour la synchronisation). Le matériel le plus lent impose donc le rythme des communications. On parle alors de transmission orienté* ***<u>message</u>****. Le temps qui sépare l'envoi de 2 messages doit être un multiple de Δt d'un bit. Un ou plusieurs caractères de synchronisation puis la totalité des données. Pas de contrôle d'erreurs (overrun ou underrun). Ex : mode BSC ou HDLC (High level Data Link Control).*



Δt : temps nécessaire pour l'envoi d'un bit
T: temps entre l'envoi du message 1 et du message 2, multiple entier de Δt

- **asynchrone :** *Transmission série entre terminaux et/ou ordinateurs ayant lieu caractère par caractère, le temps entre deux caractères n'étant pas défini. Chaque caractère est précédé d'un STARTbit et suivi d'un ou deux STOPbits, ce qui signifie que, en ASCII (étendu), il faudra 10 à 11 bits par caractère. L'avantage des transmissions asynchrones réside dans la simplicité de la méthode (le caractère est envoyé dès que la touche est appuyée). La synchronisation est donc imposée par le protocole. On parle de transmission orientée* ***caractère****.*



Δt : temps nécessaire pour l'envoi d'un bit
T: temps quelconque entre l'envoi du message 1 et du message 2

☞ **: protocoles :**

Même notion que la langue dans le language humain. Ce sont des règles strictes, définissant les questions et les réponses devant avoir lieu lorsque deux équipements sont en communication. Ces règles prévoient des procédures de récupération en cas d'erreur de transmission ou de « timeout » (réponse non parvenue dans les délais).

☞ **: polling / interruption**

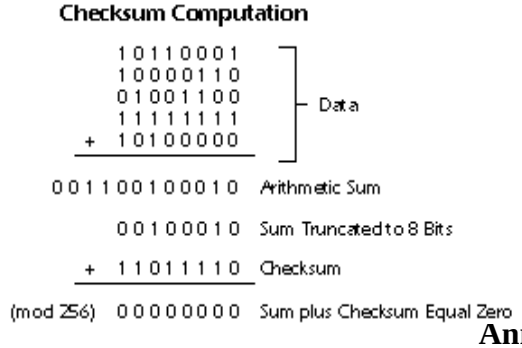
- **polling (scrutation)** *: méthode qui consiste a scruter en permanence l'état de la ligne (du bus) pour savoir si une information est arrivée. Cela ralentit énormément la vitesse des échanges et monopolise une grande part des ressources CPU..*
- **Interruption** *: Evénement extérieur au programme ou à l'unité centrale qui arrête (déroute) l'exécution du travail en cours. Afin de pouvoir continuer ce travail par la suite, il est important de sauver toute l'information critique de ce travail. Surtout utilisée pour traiter des périphériques à vitesse max (sans scruter l'état "prêt" du périphérique).*

☞ **: techniques de correction / erreurs**

- **Parité** *: Bit(s) ajouté(s) à l'information pour en vérifier l'intégrité. Lorsque l'on parle de caractères, la parité (horizontale : LRC) peut être paire (toujours un nombre pair de bits positionnés à 1) ou impaire, ou forcée à un, ou nulle. La vérification de la parité peut être utile lorsque l'on transmet de l'information à haute vitesse ou sur des lignes de qualité médiocre.*

Elle peut aussi être verticale → Checksum (à la fin de chaque message).

- **CRC** *: Cyclic Redundancy Check. Code (polynômial) de redondance cyclique. Le CRC est utilisé pour se prémunir contre les pertes d'informations ou leur altération en créant une redondance limitée permettant de garantir la non-altération du message. Les CRC comprennent 16 ou 24 bits.*



I.6 - Liaison physique (support) de transmissions

Pour que la transmission puisse s'établir on a besoin d'un support physique (voie ou canal de transmission). Les données circulent sous forme d'ondes (acoustiques, electromagnétiques, électriques ou lumineuses). Les supports peuvent être filaires (câble) aérien (onde hertzienne) ou optique (fibre, laser).

☞ **: canal de transmission**

Il s'agit d'une liaison (ligne de transmission) entre un émetteur et un récepteur. Ce canal n'est pas forcément composé d'un seul support physique. En effet, chaque « machine » en extrémité de ligne (DTE pour Data Terminal Equipment) possède un équipement relatif au support physique auquel elles sont reliés (DCE pour Data Communication Equipment).

☞ **: perturbations :** *la transmission ne se fait pas sans pertes. On remarque divers phénomènes dans les communications :*

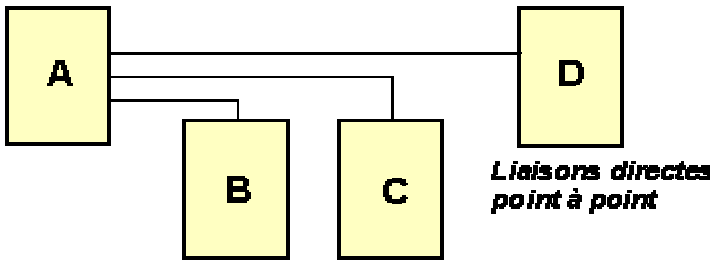
- *parasites (bruit),*
- *affaiblissement (perte en énergie dissipée dans la ligne) proportionnel à la longueur et la fréquence. Bande passante : intervalle de fréquence sur lequel le signal ne subit pas un affaiblissement trop grand (supérieur à 3dB en général)*
- *distorsion (déphasage).*

☞ **: multiplexage**

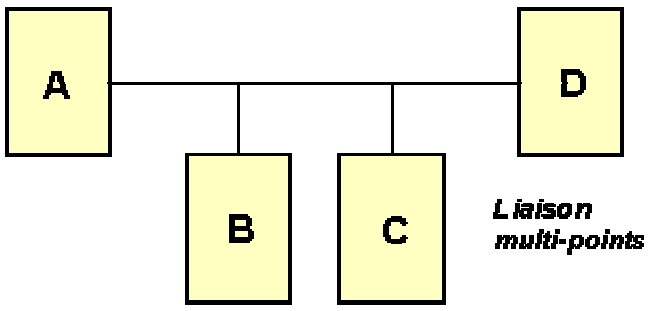
On appelle multiplexage la capacité à transmettre sur un seul support physique des données provenant de plusieurs equipement (émetteurs et récepteurs). Il existe le multiplexage fréquentiel (FDM), temporel (TDM) et statistique.

☞ **: liaisons p2p, multipoints, boucles**

- **p2p** *: point à point. liaison la plus simple. elle relie directement 2 ordinateurs. Multiplication des connexions → solution onéreuse.*



- **multipoints** *: chaque équipement est relié aux autres. Nécessite le respect absolu d'un protocole de communication.*



☞ **: liaisons tensions : modes asymétrique & symétrique (différentiel)**

- **mode asymétrique** : *les états logiques sont transmis sur la ligne par 2 niveaux de tension, l'un positif, l'autre négatif. Le plus utilisé travaille en logique négative pour le 1 logique (exemple : RS 232). Les systèmes basés sur ce mode sont sensibles aux bruits (→ 20Kbit/s, 15 mètres max).*
- **mode symétrique (différentiel)** : *Il s'agit d'un ampli-différentiel. Il n'est concerné que par la différence de tension (insensible aux bruits) → grande distance (1200 mètres) , vitesses élevées (10 Mbit/s)*

II. E/S SERIE ASYNCHRONE : RS 232

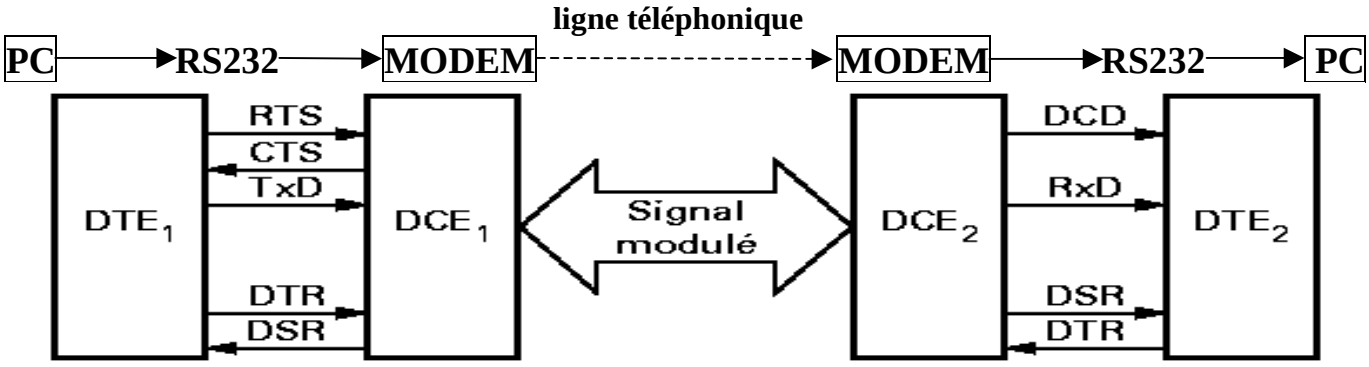
II.1 - Généralités

*La liaison série **asynchrone** est couramment utilisé en informatique et en automatique (de manière plus générale en pilotage de procédés) pour traiter tout problème de transmission. Pour transporter l'information, on utilise la tension (RS 232, RS422-liaison multipoint, RS485-liaison multipoint bidirectionnelle) ou le courant (boucle de courant 20mA) selon la norme EIA 232 (Electronic Industry Associate) depuis 1969 puis réévaluée en 1991.*

La liaison RS232 est simple, universelle, parfaitement connue et supportée par un grand nombre de périphériques. Contrairement à la liaison parallèle, elle autorise de grandes distances mais possède des réglages plus compliqués. Ceci justifie une étude détaillée.

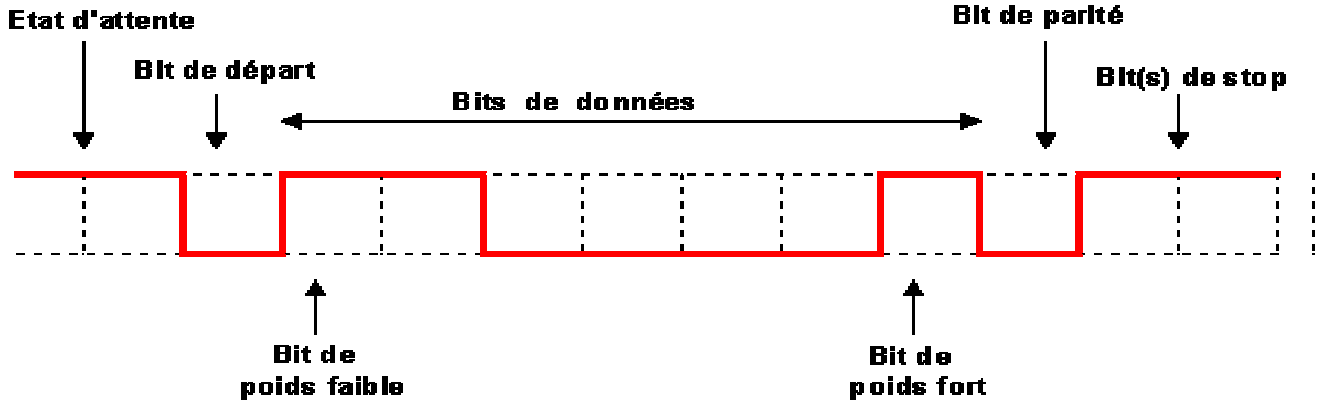
II.2 - Format et principe de communication

Schéma de principe :



*Pour que deux ordinateurs puissent communiquer entre eux, un protocole de transmission bien précis doit être respecté pour que l'émetteur et le récepteur se comprennent. Il s'agit avant tout de reconnaître les données utiles (caractère, mot, message) des données de contrôle (autres bits). Le format d'une **trame** est le suivant.*

attente	1 Bit Start	5,6,7 ou8 Bits de données(100 0011 = 43H)	parité **(facultatif)**	1-1.5-2 Bit Stop



Exemple: transmission de la lettre "C" (code ASCII: H '43')

*La transmission des caractères ne peut fonctionner correctement qu'à condition que les différents paramètres variables de cette trame soient connus aussi bien de l'émetteur que du récepteur. Il est alors nécessaire d'ajuster les paramètres suivants : la **vitesse** de transmission, le **nombre de bits** du caractère à transmettre, la **parité**, le nombre de bits **stop** et le **protocole** (handshaking matériel ou logiciel).*

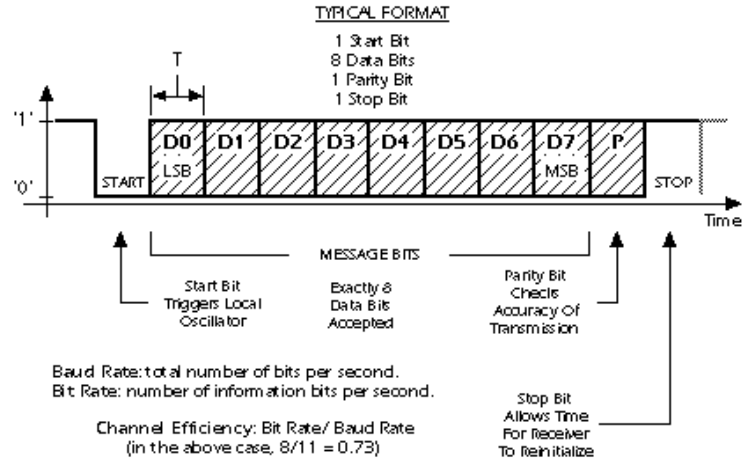
*Lorsqu'aucun caractère n'est transmis, le niveau de la ligne de transmission est à l'état haut (1 logique) : ceci permet de dissocier le bruit d'un changement d'état. La synchronisation de l'horloge d'échantillonnage des bits à la réception est assurée à l'aide du bit de Start. C'est à partir du Start que le récepteur se synchronise et échantillonne les autres bits du caractère. Donc si la ligne passe à l'état bas / 0 logique (rôle du **Start**), le récepteur sait que des bits (utiles : données caractère) vont être transmis (début de communication).*

*Le récepteur recevra ensuite les bits de **données** (codé sur 5,6,7 ou 8 bits) en commencant par le LSB suivis d'un bit de **parité** (facultatif) et d'un (ou plusieurs) bit de S**top** qui indique au récepteur la fin de transmission de la donnée. La transmission d'un caractère se termine toujours de cette façon et le niveau logique de la ligne revient à l'état haut. En ce qui concerne le bit de parité, il sert à détecter un éventuel problème lors de la transmission des données. On distingue la parité paire et impaire :*

- ***parité paire** : mettre le bit de parité à 1 ou 0 pour assurer un **nombre total pair de bit à 1**.*
- ***parité impaire** : mettre le bit de parité à 1 ou 0 pour assurer un **nombre total impair de bit à 1**.*

Les fréquences de transmission autorisées sont précisées par la norme RS232. On utilise habituellement des liaisons à 300, 1200, 2400, 4800, 9600 et 19200 Bauds.

En résumé :

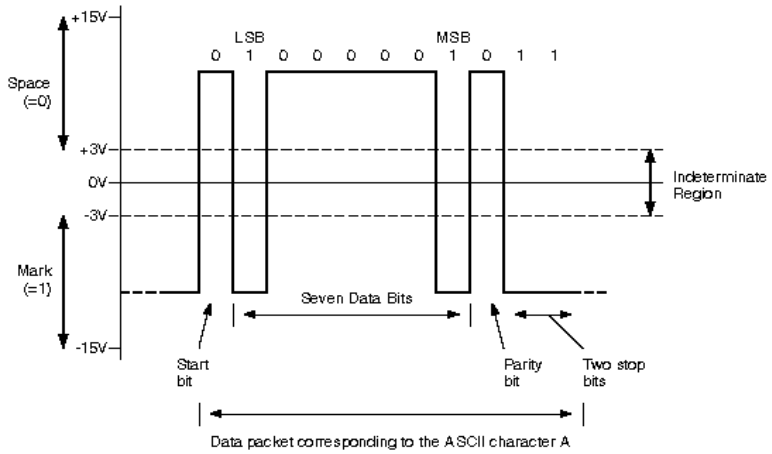


II.3 - Norme RS232

La norme RS232 définit des caractéristiques électriques (niveaux des signaux), mécaniques (connecteurs) et fonctionnelles (nature des signaux). Les valeurs des tensions que les interfaces séries doivent fournir (et reconnaître) aux matériels connectés sont les suivantes (logique négative) :

- *Un 0 logique est reconnu pour une tension allant de +3 à +25V.*
- *Un 1 logique est reconnu pour une tension allant de -3 à -25V.*

Généralement, les signaux envoyés sont compris entre -15 et + 15 V (tensions d'alimentation des A-Op). Sur une liaison série au repos, on doit observer un 1 logique (voir schéma ci dessous) :



II.4 - Brochage du port série standard

Le port série permet de connecter un grand nombre de périphériques tels que :

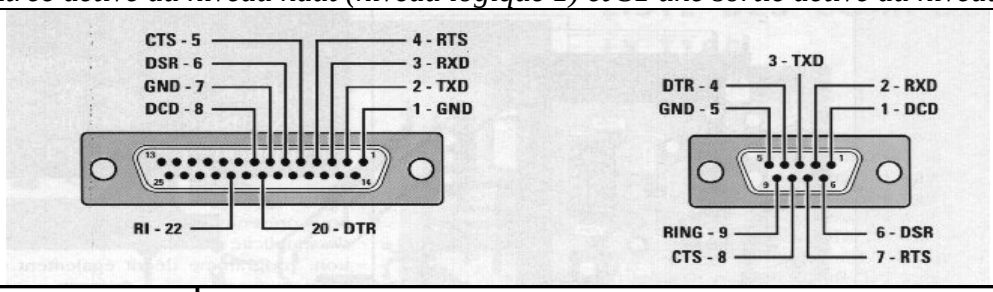
- *un modem*
- *une imprimante*
- *un autre ordinateur*
- *un lecteur de codes barres, etc…*

La plupart des périphériques nécessitant une connexion bilatérale pour communiquer avec un PC utilisent un port série standard RS232.

DB 9	*DB 25*	*Signal*	*Fonction*	*Sens*
1	8	DCD	Détection de porteuse/tonalité (Data Carrier Detected)	⇦ E1
2	3	RxD	Réception des données (Received Data)	⇦ E
3	2	TxD	Transmission des données (Transmitted Data)	⇨ S
4	20	DTR	Terminal prêt (Data Terminal Ready)	⇨ S1
5	7	SG	Masse du signal (Signal Ground)	
6	6	DSR	Données prêtes (Data Set Ready)	⇦ E1
7	4	RTS	Demande d'émission (Request To Send)	⇨ S1
8	5	CTS	Prêt pour l'émission (Clear To Send)	⇦ E1
9	22	RI	Indicateur de sonnerie (Ring Indicator)	⇨ E1

II.5 - Description des signaux

E1 signifie entrée active au niveau haut (niveau logique 1) et S1 une sortie active au niveau haut .



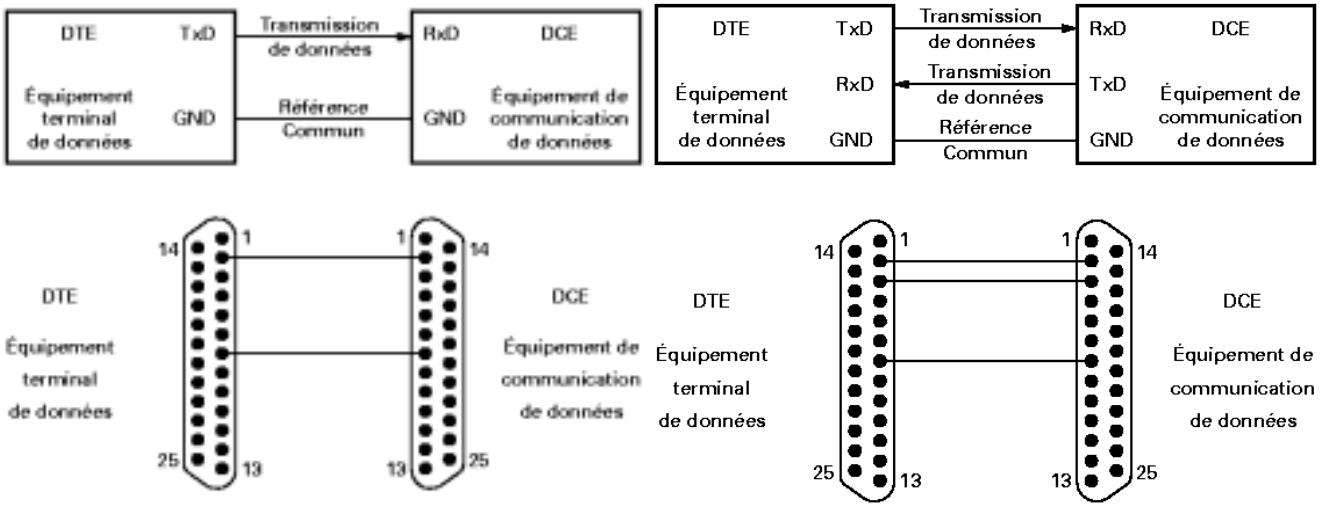
Connecteur	Fonctions attribuées aux broches								
	DCD	RxD	TxD	DTR	GND	DSR	RTS	CTS	RI
DB-9	1	2	3	4	5	6	7	8	9
DB-25 (1)	8	3	2	20	7	6	4	5	22

- ***SG** est la liaison de masse électrique qui peut être confondue avec la masse mécanique*
- ***TxD** est la ligne d'envoi des données.*
- ***RxD** est la ligne de réception des données.*
- ***RTS** est une ligne de demande d'émission. La ligne est positionnée à l'état haut lorsque le terminal veut envoyer des données.* **(DTE → DCE)**
- ***DTR** est positionné par l'émetteur (terminal) à l'état haut pour signaler au récepteur (ordinateur) qu'il est en ligne et qu'il est prêt à émettre des données.*
- ***DSR** indique (si niveau haut) que l'ordinateur est prêt à recevoir les données du terminal.*
- ***CTS** est une ligne d'invitation à émettre. Elle passe au niveau haut lorsque le DTE attend des données du DCE. Il signale qu'il peut recevoir les données du DCE et qu'il peut donc les lui envoyer.*
- ***DCD** n'est utilisé en principe que sur les modems. C'est la ligne de détection de la porteuse qui passe à l'état haut lorsque le modem reçoit une porteuse valide (tonalité). Il permet à un modem d'avertir le terminal qu'il est en relation avec un autre.*
- ***RI** est encore une ligne utilisée seulement par les modems. C'est l'indication de sonnerie. Par cette ligne, le modem avertit l'ordinateur que le téléphone sonne. (en général, intégré).*

II.6 - Modalités de transmission (simplex-duplex)

Pour réaliser une interface, il faut 2 éléments : un DTE (Data Terminal Equipment, exemple PC, écrans) et un DCE (Data Communication Equipment, exemple modem). La transmission entre le DTE et le DCE peut se faire de plusieurs manières :

- *dans une seule direction (DTE vers DCE ou DCE vers DTE). On parle de liaison simplex (voir figure ci-dessous). Dans ce cas, 2 fils suffisent : l'un pour transporter les données, l'autre comme référence (la masse en général).*
- *en mode bidirectionnel : les dispositifs DCE et DTE doivent pouvoir émettre et recevoir des données. On parle de liaison duplex (voir figure ci-dessous).Si la communication est alternée : half-duplex, simultanée : full-duplex.*



Liaison simplex liaison duplex

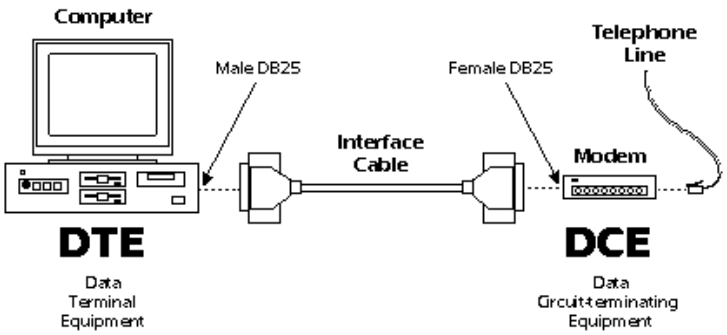
II.7 - Différents protocoles

Lorsqu'un DTE émet plus de données que ne peut accepter un DCE ou pour déterminer si le DTE ou si le DCE est en service, on utilise les signaux de protocole d'accord (« Handshake »). Les signaux correspondants permettent de surveiller l'état d'un autre et de répondre en conséquence. Ils indiquent la façon par laquelle le flot des données passant dans l'interface est régulé et commandé.

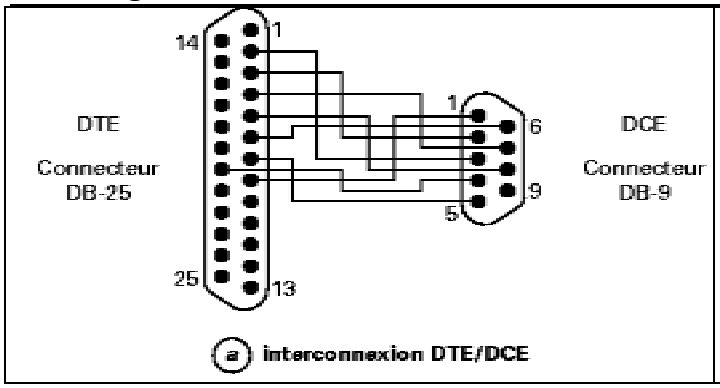
Selon la nature (DTE ou DCE) des appareils connectés, le type de connexion et le protocole d'échange pourra être différent. Il existe 2 grandes familles de protocole d'accord :

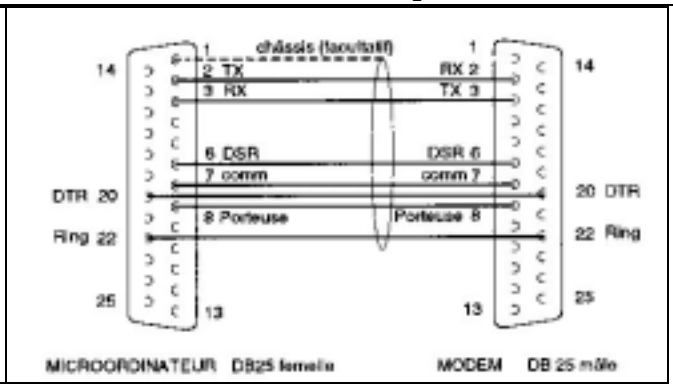
- *Matériel : au niveau physique (fils, tensions). Utilisable seulement si les appareils concernés peuvent être connectés par câbles. DTR-DSR et RTS-CTS.*
- *Logiciel : au niveau du contenu des données. Ces dernières contiennent des caractères spéciaux (de contrôle). Xon-Xoff.*

II.7.1 Liaison DTE-DCE (via modem)



Aux extrémités du câble on trouve un connecteur DB25 femelle (coté ordinateur) et un connecteur DB25 mâle (coté modem). Les connections sont directes (« câbles droits »).





***DCE -> DTE** : le modem DCE (émetteur), avant toute action, vérifie le niveau de son fil 20 (DTR = 1 ? : est-ce que le DTE est prêt ?). Si le terminal est éteint (niveau bas), rien ne se passe. Lorsque le DTE est en ligne (DTR = 1), alors le modem teste le fil 4 (RTS). S'il est activé, le modem accepte d'envoyer ses données au terminal. Ainsi, le modem ne transmet rien au terminal tant que DTR et RTS ne sont pas tous les deux mis à 1 par le terminal DTE.*

***DTE -> DCE** : le terminal possède un clavier et peut donc envoyer des caractères au modem à condition que celui-ci soit en relation avec l'ordinateur distant ce qui est détecté par la présence d'une fréquence porteuse (carrier). Si c'est le cas, le DCE met le fil 8 (DCD) au niveau haut. Il peut mettre aussi DSR au niveau haut pour "dire" au terminal que des données arrivent et qu'il faut les lire, et le fil 5 (CTS) pour lui signaler qu'il peut écrire...*

Le terminal manoeuvre les lignes DTR et RTS, les plaçant au niveau haut lorsqu'il est prêt à un dialogue, et il scrute les lignes DCD, DSR et CTS pour vérifier que son correspondant l'écoute.

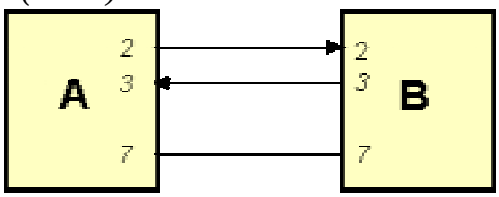
On voit que ces lignes font partiellement double emploi dans une communication full-duplex. C'est pourquoi fréquemment un DTE pourra ne commander que DTR ou RTS et ne scruter que l'une ou l'autre des lignes DSR, CTS et/ou CD. Un DCE pourra ne scruter que DTR ou RTS et ne commander que l'une ou l'autre des lignes DSR, CTS et/ou CD.

II.7.2 Liaison DTE-DTE sans contrôle de flux

Pour connecter 2 DTE entre eux par une liaison série RS232, il faut au minimum 3 fils :

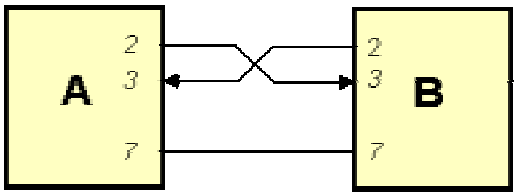
- *Un pour les données qui circulent dans un sens.*
- *Un pour les données qui circulent dans l'autre sens.*
- *Un pour la masse électrique des signaux.*

On réalise donc le câblage suivant (DB25) :



C'est une liaison half-duplex. En effet, B doit attendre de recevoir la donnée en 2 avant de pouvoir émettre sur A en 3.

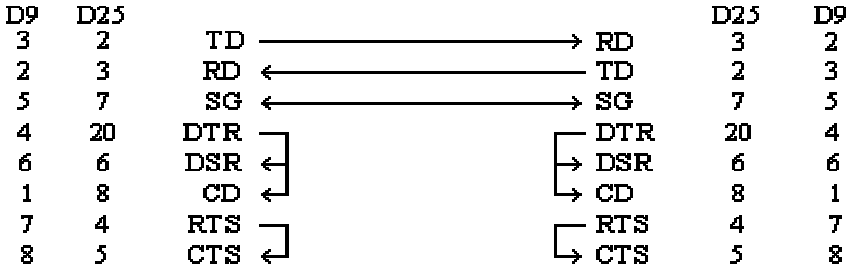
*Si on veut initier une liaison full-duplex, l'émission de l'un doit correspondre à la réception de l'autre. On doit donc **croiser** les connexions (soit directement sur l'équipement, soit sur le câble). Cette liaison à 3 fils est minimum.*



En effet personne ne vérifie si son correspondant est prêt à émettre, prêt à recevoir ou tout simplement sous tension !

☞ **: null modem**

Ici, le but est de s'affranchir d'un grand nombre de fils (économie, place, simplicité, etc…).

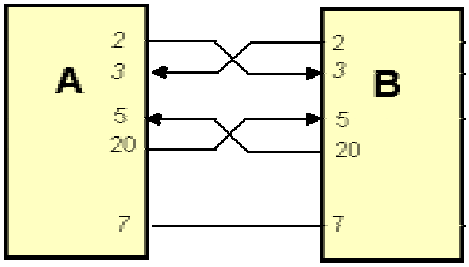


L'idée est de faire croire que l'ordinateur A parle avec un modem B plutôt qu'avec un ordinateur B. La ligne DTR est rebouclée sur DSR et DCD. Ainsi, lorsque DTR est mis à 1, le DSR et DCD deviennent immédiatement actifs. L'ordinateur A pense alors que le modem (virtuel) B auquel il est connecté est prêt et a détecté la porteuse d'un autre modem. Comme les ordinateurs communiquent à la même vitesse (pas de contrôle de flux) RTS et CTS sont bouclées ensemble. Ainsi, lorsque lordinateur veut émettre, il positionne RTS à 1 et reçoit immédiatement 1 sur CTS (autorisation d'émettre) et commence donc son émission.

II.7.3 Liaison DTE-DTE + contrôle de flux matériel

☞ : liaison complète

D'autres signaux ont été ajoutés à ces trois lignes de base afin de permettre un contrôle du déroulement de la liaison par l'un ou l'autre des équipements et éviter par exemple qu'un équipement envoie des informations à un autre qui n'est pas prêt à les recevoir parce qu'il n'est pas connecté, pas sous tension ou parce qu'il est trop lent.

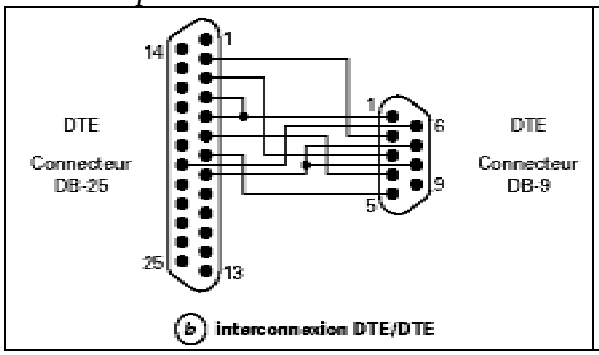


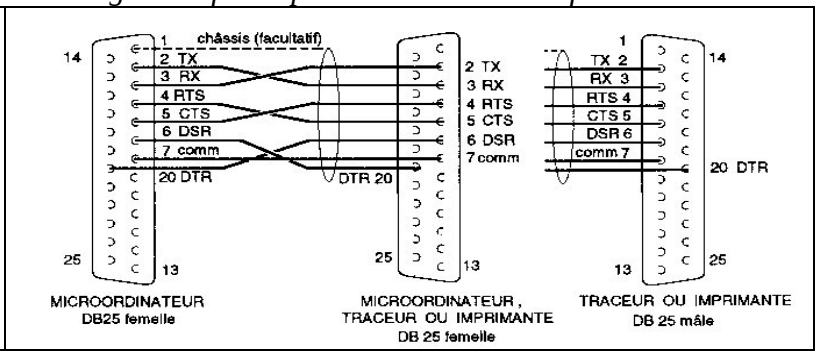
***Problème** : éviter que chaque DTE attend une autorisation sur l'une ou l'autre de ses lignes CTS, DSR et CD car on attendra indéfiniment que la conversation commence. Cependant, on remarquera que chacun des correspondants manipule ses lignes RTS et DTR dès qu'il est prêt et conclura que l'on peut servir les CTS, DSR et CD de chacun par des lignes RTS et DTR de l'autre.*

***Problème de régulation de flux de données** : il serait extraordinaire que l'émetteur envoie des données exactement au rythme optimal pour le récepteur. S'il est plus lent, cela n'est pas grave. Par contre, s'il est plus rapide que le récepteur, il est possible de perdre des données. Pour éviter ce problème, il faut :*

- *soit que le récepteur traite les données plus vite qu'elles n'arrivent*
- *soit qu'il fasse stopper le flux de données lorsqu'il risque d'être momentanément dépassé. pour ce faire, il faut que le récepteur avertisse l'émetteur.*

L'idée la plus naturelle est de cabler la totalité des signaux afin de pouvoir contrôler les flux





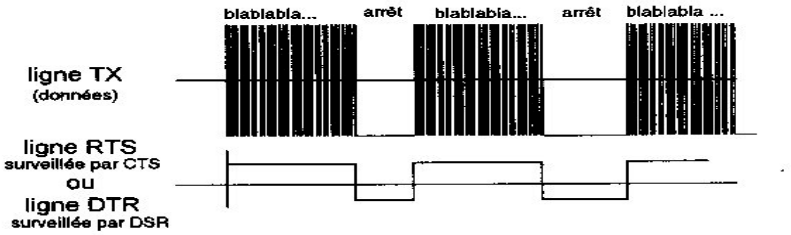
*Le récepteur indique à l'émetteur qu'il est prêt à recevoir une donnée. L'émetteur met alors la donnée sur la ligne de transmission. C'est le **Handshaking**.*

*Dans le cas où des signaux physiques sont chargés de contrôler le flux des échanges de données, on parlera de **handshaking matériel** caractérisé par le terme de **protocole RTS/CTS**.*

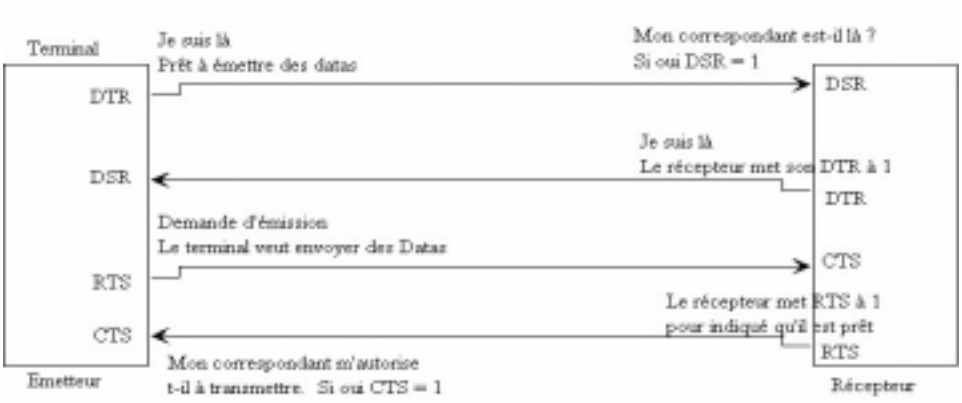
☞ : protocole RTS / CTS

*Lorsque l'émetteur veut émettre ses données, il doit positionner la ligne **RTS** à l'état haut pour demander au récepteur s'il est prêt à accepter ces données. Le récepteur, lorsqu'il est prêt à recevoir les données, va positionner le signal **CTS** de l'émetteur à l'état haut pour lui indiquer qu'il est prêt.*

L'émetteur n'envoie ses données que si la ligne DTR est mise à l'état haut par le récepteur. Le récepteur le constate en la surveillant par son entrée DSR.



Résumé :



- ***Pour l'émission d'un caractère :***
  - *met à 1 la ligne DTR (20) : "je suis là"*
  - *met à 1 la ligne RTS (4) : "je demande à émettre"*
  - *vérifie si DSR (6) vaut 1 : "mon correspondant est-il toujours là"*
  - *attend le signal CTS (5) : "suis-je autorisé à transmettre ?"*
  - *envoie alors le caractère*

Pour la réception d'un caractère :

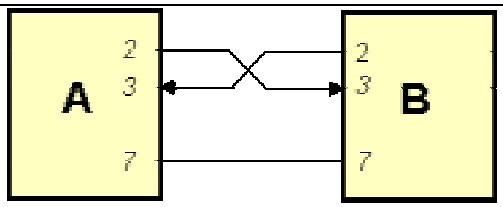
- *met DTR (20) à 1 : "je suis là"*
- *vérifie si DSR (6) vaut 1 : "mon correspondant est-il toujours présent?"*
- *attend la réception d'un caractère : si le caractère est présent dans le registre de réception de données, on le lit.*

II.7.4 Liaison DTE-DTE + contrôle de flux logiciel

☞ : protocole ETX / ACK *:*

*Le plus simple des protocoles logiciels est le protocole ETX-ACK. Les données sont envoyées par blocs de **longueur fixe**. Après émission d'un bloc, le caractère ASCII ETX (0x03, envoyé) est envoyé. Le récepteur répond par le caractère ASCII ACK (0x06, reçu) si la transmission s'est correctement déroulée ou par le caractère NAK (0x015, non reçu) dans le cas contraire.*

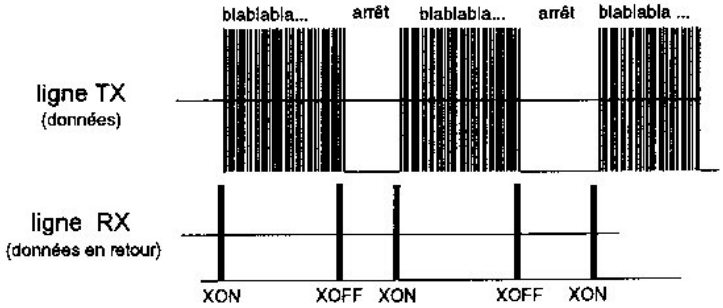
Ce protocole ne nécessite qu'une liaison sur 3 fils. Le reste de la négociation entre l'émetteur et le récepteur pour échanger des données se fait par logiciel.



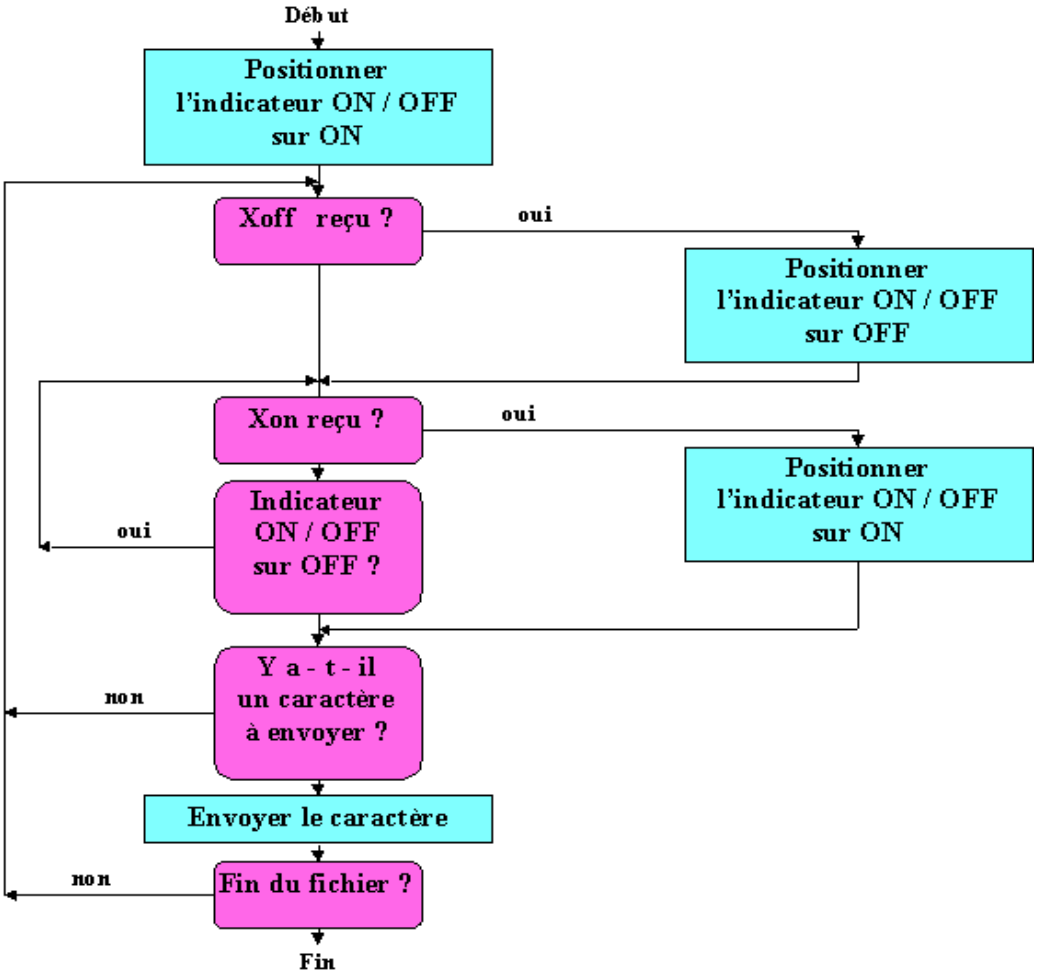
Dans le cas où des caractères particuliers sont chargés de contrôler le flux (la vitesse) des échanges de données, on parlera de ***handshaking logiciel****, caractérisé par le terme de* ***protocole Xon/Xoff****.*

Ce protocole est basé sur les caractères XON (DC1,code ASCII 0x11) et XOFF (DC3,code ASCII 0x13) :

Le récepteur gère un buffer. Les données sont traités par blocs de ***longueur variable****. Lorsque son buffer est plein à 80 %, le récepteur envoie le caractère XOFF. L'émetteur lorsqu'il reçoit le caractère XOFF doit immédiatement suspendre son émission. Lorsque le récepteur a vidé son buffer à 50%, il envoie un caractère XON à l'émetteur. A la réception de XON l'émetteur peut reprendre son émission.*



Il est possible que l'émetteur ne reçoive pas ou perde les caractères XON - XOFF. Pour pallier à ces problèmes, lorsque l'émetteur n'a pas reçu de caractères depuis un certain temps, ce dernier peut reprendre de sa propre initiative le transfert. Si le récepteur n'est pas d'accord, ce dernier pourra toujours réémettre un XOFF.



II.8 - Annexes

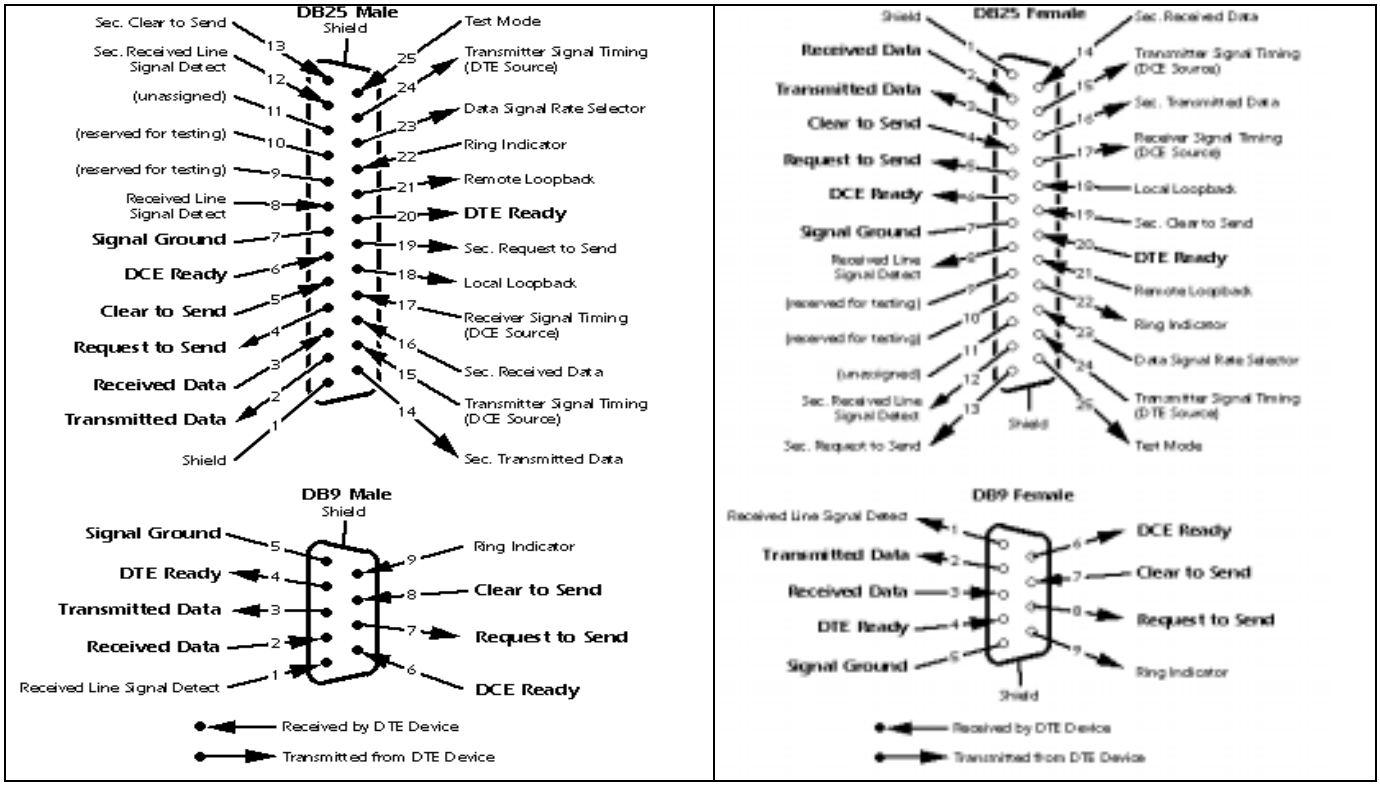
II.8.1 Procédure de test : rebouclage entrées / sorties

Pour détecter (procédure de LOOPBACK) si une panne dans la transmission est matérielle ou logicielle, il suffit de connecter TxD à RxD, DTR étant rebouclé sur DSR et DCD et RTS est bouclé sur CTS.

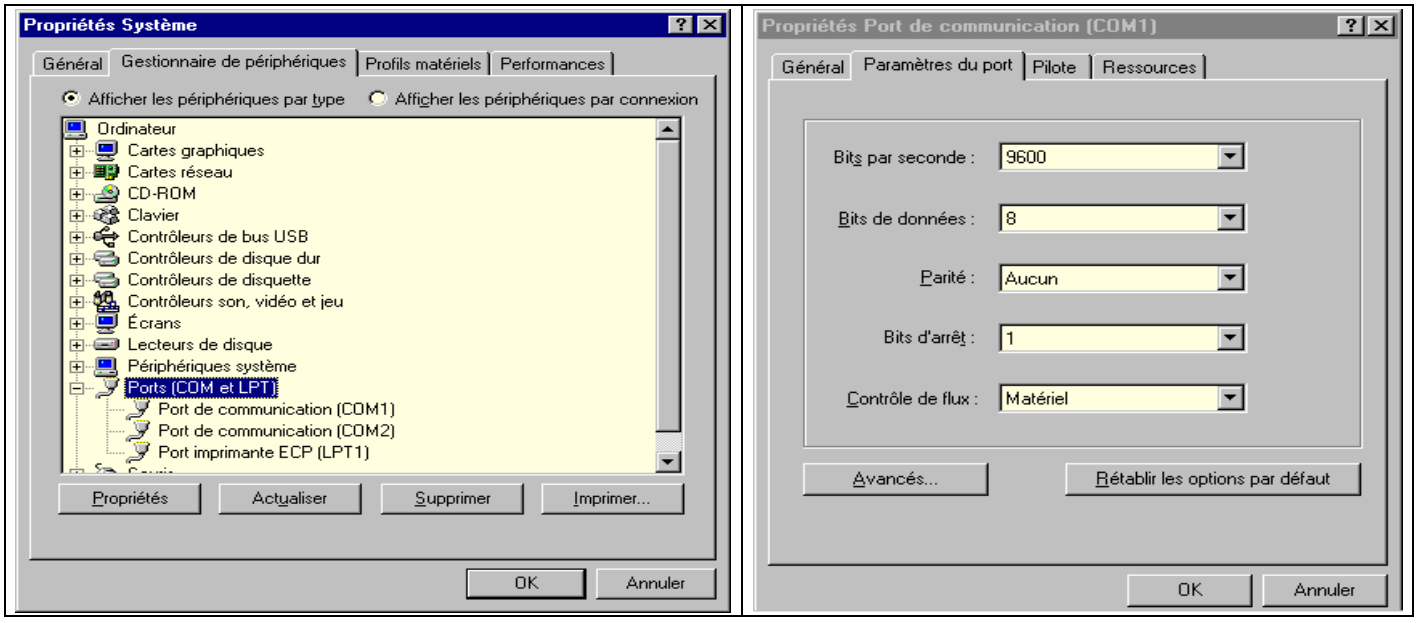
II.8.2 Comment reconnaître un DTE d'un DCE ?

Sur une DB25, un DTE délivre une tension non nulle sur les signaux TD (mesurer 2-7), RTS et DTR tandis qu'un DCE délivre une tension non nulle sur les signaux RD (mesurer 3-7), CTS, DSR et DCD.

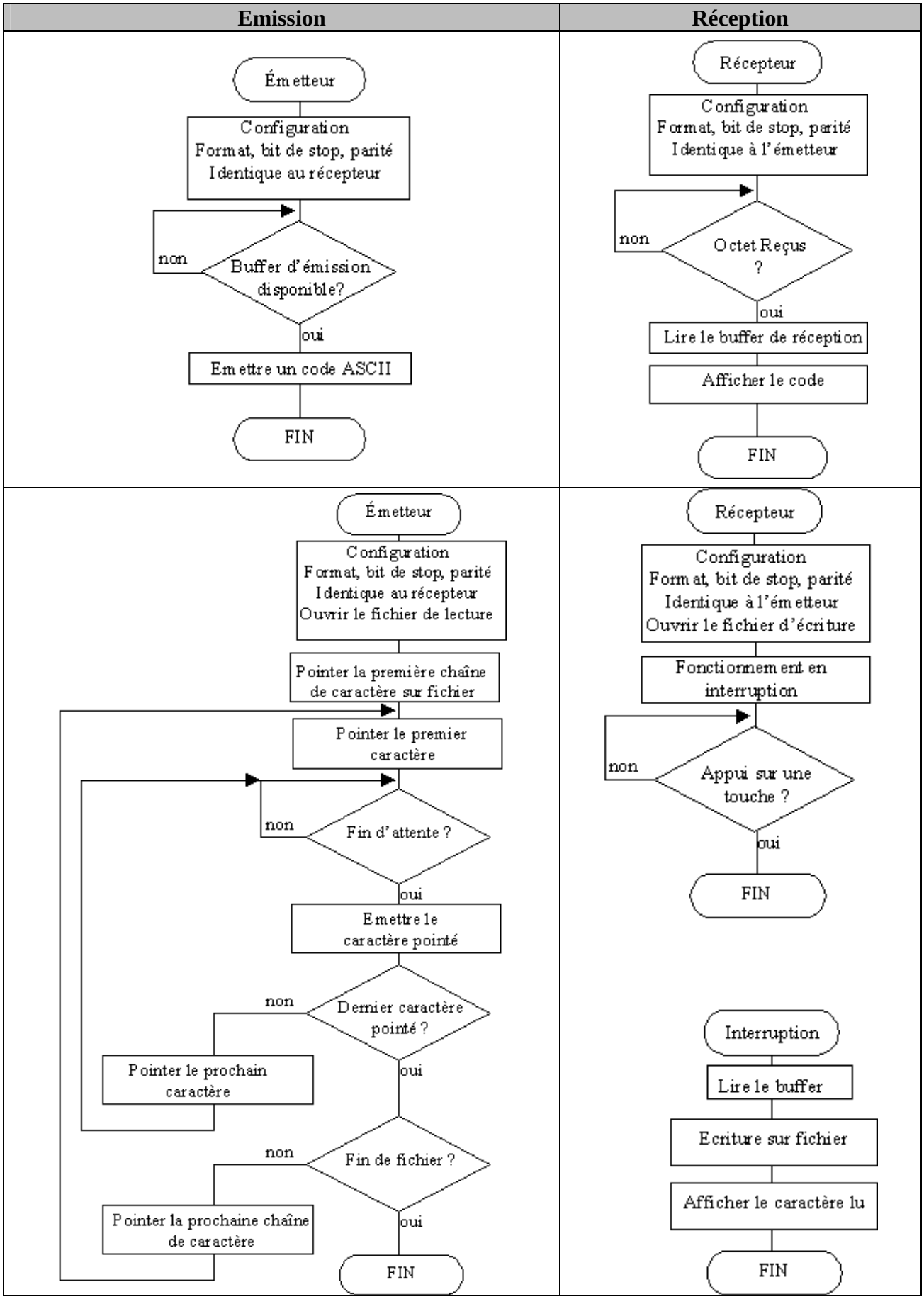
II.8.3 DTE-DCE : connections sur DB25



II.8.4 Protocole d'échange sur PC



II.8.5 Algorithmes Emission-Réception

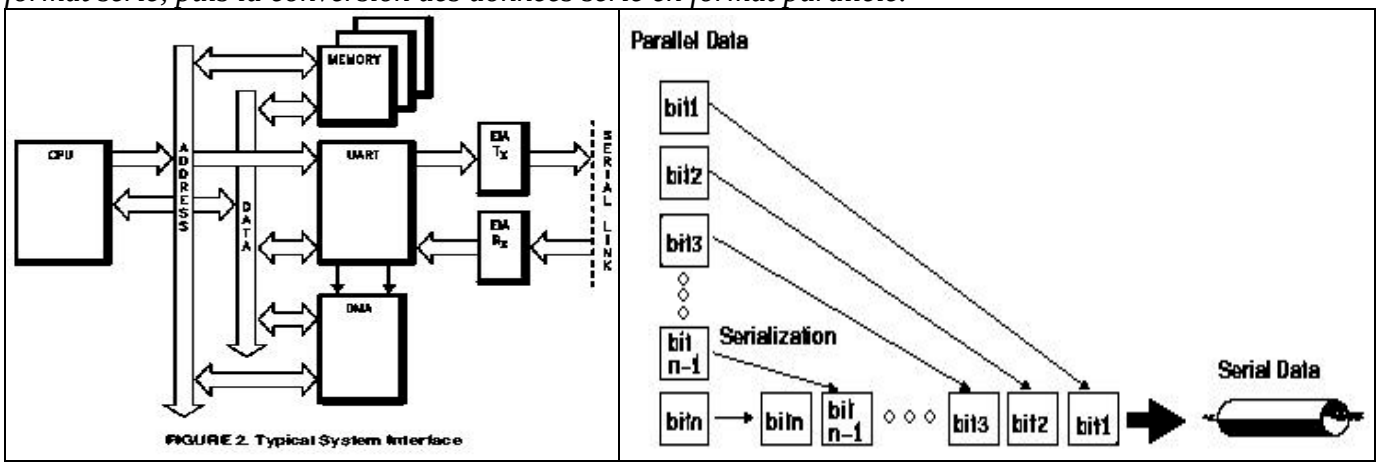


III. *UART 8250-16550*

III.1 - Introduction

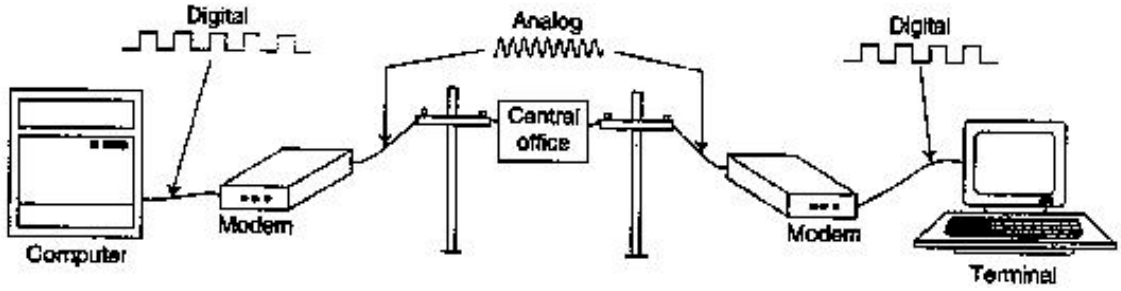
Après avoir étudié le mode de fonctionnement d'une liaison série RS232, nous allons étudier comment programmer le contrôleur d'interface UART chargé de piloter cette liaison.

L'UART 8250-16550 *(**U**niversal **A**synchronous **R**eceiver **T**ransmitter) constitue le cœur de tout port série. Il contrôle complètement la conversion des données natives parallèles de l'ordinateur (CPU) en format série, puis la conversion des données série en format parallèle.*

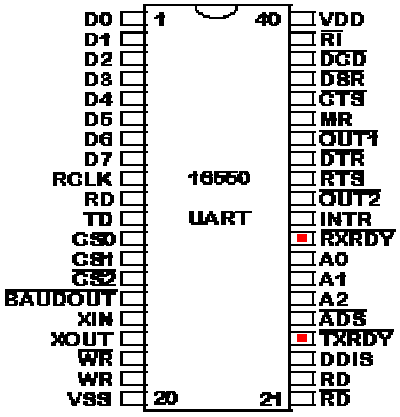


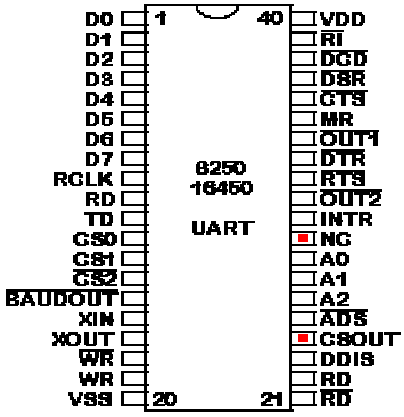
*La RS232 est l'interface standard, le modem réalise la CNA / CAN, et les échanges se font par une liaison téléphonique moyennant une modulation FSK (**F**réquency **S**hift **K**ey) selon le schéma de principe suivant : .*

CPU → **UART** → **Port RS232** → **MODEM (FSK)** ligne téléphonique



*Les PC originaux utilisaient l'**UART 8250**, qui est toujours utilisée par un grand nombre de cartes adaptateurs série du marché. L'évolution a conduit à l'**UART16550** (tampon 16 octets, débit 115.2 KBps → communications rapides). Son architecture interne est donnée en annexe. Les différences se situent dans l'apparition de 2 FIFO au niveau des registres d'émission / réception pour contrôler les échanges.*





III.2 - Configuration du port série

*L'UART peut être utilisée selon 2 modes : **interruption** et **scrutation**.*

- *En mode « scrutation » (« polling »), le CPU lit en permanence les lignes pour déterminer l'état de la transmission (prêt à émettre TxRDY, donnée reçue RxRDY).*
- *En mode « interruption », à chaque fois qu'un caractère est reçu par un port série, il doit attirer l'attention de l'ordinateur en activant un canal de requête d'interruption. Les ordinateurs à bus ISA 8 bits comportent 8 canaux de ce type. Les ordinateurs à bus ISA 16 bits en comportent 16. C'est généralement la puce contrôleur d'interruption 8259 qui traite ces requêtes d'attention. Sur une configuration standard, le port série COM1 utilise l'IRQ 4 et le port série COM2 utilise l'IRQ3.*

Lorsqu'un port série est installé sur un PC, il doit être configuré pour utiliser des adresses d'E/S spécifiques et des interruptions (appelées IRQ). Il est préférable d'utiliser les standards existants. COM1 et COM2 qui sont respectivement aux adresses de base 0x3F8 et 0x2F8 :

Connaissant l'adresse mémoire du port série concerné, comment dialoguer avec l'UART 16550 afin de configurer la communication ?

① - en utilisant les fonctions de l'interruption logicielle 0x14

② - en programmant les registres du 16550

III.3 - Les Registres de l'UART 8250

Le 16550 comporte 11 registres de 8 bits classés en trois catégories :

- *les registres de contrôle : DLL, DLM, IER, LCR, MCR, FCR.*
- *les registres d'état : LSR, MSR et IIR*
- *les registres de données : RBR et THR*

TABLE DES REGISTRES

adresse port+$A_2A_1A_0$	000			001		010		011	100	101	110	111	
DLAB	0	0	1	1	0	x	x	x	x	x	x	0	1
R/W	R	W	R/W	R/W	R/W	R	W	R/W	R/W	R	R	R/W	
Name	**RBR**	**THR**	**DLL**	**DLM**	**IER**	**IIR**	**FCR**	**LCR**	**MCR**	**LSR**	**MSR**	**SCR**	
Bit 0	D0	D0	B0	B8	ERDA	IP	FE	WLS0	DTR	DR	DCTS	x	P0
Bit 1	D1	D1	B1	B9	ETD	ID1	RFR	WLS1	RTS	OE	DDSR	x	P1
Bit 2	D2	D2	B2	B10	ERLS	ID2	TFR	STB	OUT1	PE	TERI	x	P2
Bit 3	D3	D3	B3	B11	EMS	0	DMS	PEN	OUT2	FE	DDCD	x	P3
Bit 4	D4	D4	B4	B12	0	0	TTL	EPS	LOOP	BI	CTS	x	P4
Bit 5	D5	D5	B5	B13	0	0	TTM	SP	AFE	THRE	DSR	x	P5
Bit 6	D6	D6	B6	B14	0	FE1	RTL	BC	0	TSRE	RI	x	ILS / OS
Bit 7	D7	D7	B7	B15	0	FE2	RTM	DLAB	0	RFE	DCD	x	0

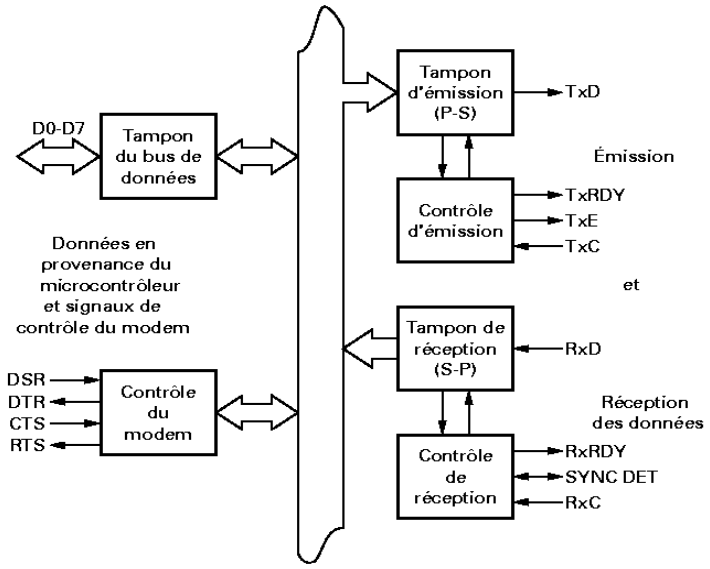
- *L'adresse physique de chaque registre est calculée relativement à l'adresse de base. Si l'adresse de base de l'interface est 3xF8 (COM1), l'adresse du LCR sera 0x3F8+0x003 = 0x3FB.*
- *L'accès aux registres se fait par l'utilisation des fonctions **inp(adresse)** et **outp(adresse, val)**.*
  - *char **inp**(int adresse_port**)** : lit un caractère(ASCII) sur le port situé à l'adresse indiquée et le retourne*
  - ***outp** (int adresse_port, char val) : écrit la valeur « val » sur le port situé à l'adresse indiqué.*

- *Certains registres possèdent la même adresse physique. Toutefois il ne peuvent être accédés simultanément (lecture ou écriture). Cela ne pose donc pas de problème (économie de registre physique).*
- *Nous allons étudier progressivement (selon leur adresse dans la table) chaque registre.*

III.3.1 RBR/THR : émission/réception

RBR (Receiver Buffer Register) : registre tampon de réception, couplé au RSR (registre à décalage)
THR (Transmitter Holding Register) : registre d'attente d'émission, couplé au TSR (registre à décalage)

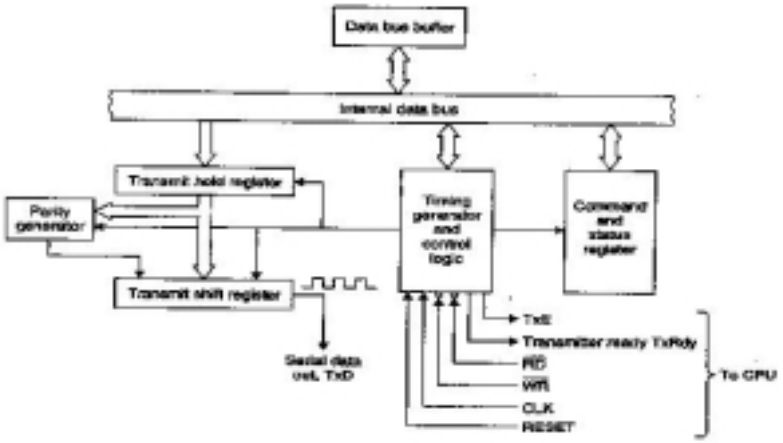
Le chargement et la lecture s'effectuent par le CPU au travers des registres d'émission THR et de réception RBR situés tous deux à l'adresse relative +0 et possédant 8 bits de données (D0 à D7). Le bit D0 est le LSB. Il est le premier à être émis ou bien reçu. On écrit (W) la donnée dans THR tandis qu'on la lit (R) dans RBR.



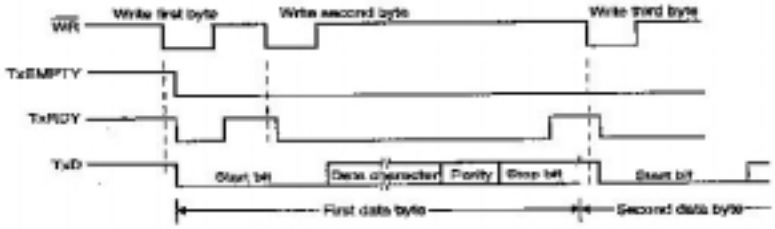
Sync Det : signal utilisé pour synchroniser le récepteur
TxE : signal permettant d'inhiber la transmission

<u>en émission (sans FIFO) :</u>

La donnée à transmettre transite sur le bus interne et arrive dans le registre THR controlé par d'autres registres. La donnée est ensuite transférée dans le registre à décalage d'émission (TSR) afin de sérialiser la donnée sur la ligne d'émission TxD. Lorsque le registre d'émission est vide , l'état est communiqué au système via THRE.

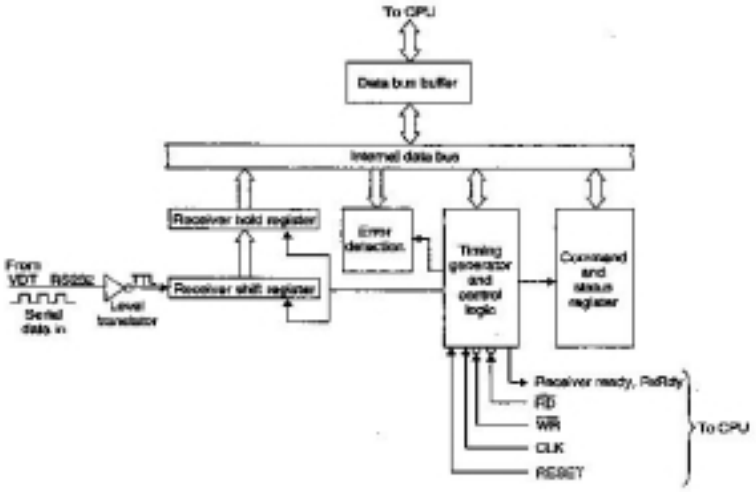


Exemple de séquence temporelle en émission :



☞ ***en réception (sans FIFO) :***

La donnée provenant de la ligne RS232 est convertie en TTL (exploitable par le registre) puis est chargée dans le registre à décalage de réception (RSR) où l'on désérialise la donnée. Elle est ensuite transférée dans le registre de réception (RBR) sous contrôle d'autres registres avant de transiter sur le bus interne en direction du CPU.



III.3.2 DLL / DLM : vitesse de transmission

***DLL : D**ivisor **L**atch **L**SB – registre LSB de division d'horloge*
***DLM : D**ivisor **L**atch **M**SB – registre MSB de division d'horloge*
*L'accès se fait aux adresses relatives +0 (DLL) et +1 (DLM) mais nécessite que le bit **DLAB** soit positionné à 1 (pour ne pas accéder par défaut aux RBR-THR en +0 et à IER en +1). Le chargement se fait sur 2 registres car on a besoin de coder les valeurs sur 2 octets. Ces registres permettent de diviser la fréquence d'entrée (déjà pré-divisée par 16) par la valeur chargée. Sur les PC, les fréquences d'horloges sont de 1.8432 et 3.072 Mhz.*

Le mode de calcul est le suivant : vitesse transmission = $\dfrac{\textbf{\textit{fréquence horloge}}}{(16 * [DLM \times 256 + DLL])}$

Ci dessous, un tableau résumé des valeurs à charger pour les principales vitesses de transmission :

Vitesse transmission	***50***	***300***	***1200***	***2400***	***4800***	***9600***	***19200***	***38400***	***56200***
Nb à charger pour 1.8432 Mhz	*2304 0x900*	*384 0x180*	*96 0x60*	*48 0x30*	*24 0x18*	*12 0x0C*	*6 0x06*		*2 0x02*
Nb à charger pour 3.072 Mhz	*3840 0xF00*	*640 0x280*	*160 0xA0*	*80 0x50*	*40 0x28*	*20 0x14*	*10 0x0A*	*5 0x05*	

III.3.3 IER : autorisation d'interruption

***IER : I**nterrupt **E**nable **R**egister – registre d'autorisation des interruptions*
L'accès se fait à l'adresse relative +1 avec DLAB à 0. Il permet de spécifier les conditions de génération des interruptions (autorise les interruptions en réception ou en transmission et l'accès à LSR). Lorsqu'un bit est à zéro, cela signifie qu'il n'y aura pas de signal d'interruption émis pour l'événement désigné. Si tous les bits sont à 0, cela signifie qu'il n'y aura aucun cas d'interruption produite par le circuit vers le gestionnaire d'interruption du PC (8259).

- *bit 0 (ERDA : **E**nable **R**eceive **D**ata **A**valaible int) : interruption quand données reçues dans RBR (0 = non et 1 = oui)*
- *bit 1 (ETD : **E**nable **T**ransmit holding register **D**ata available interrupt) : interruption produite lorsque le registre d'attente d'émission THR devient vide (fin d'émission d'un caractère)*
- *bit 2 (ERLS) : **E**nable **R**eceiver **L**ine **S**tatus interrupt) : interruption autorisée lors d'un changement d'état de la ligne de réception.*
- *bit 3 (EMS : **E**nable **M**odem **S**tatus int) : interruption autorisée lors changement d'état du modem.*
- *bits 4 à 7 inutilisés et toujours à zéro*

III.3.4 IIR : nature des interruptions

***IIR** : **I**nterrupt **I**dentification **R**egister – registre d'identification des interruptions*

L'accès se fait à l'adresse relative +2 en lecture seule. Il permet de traiter une interruption selon 4 niveaux de priorité (interface avec un microprocesseur).

- *bit 0 (IP : **I**nterrupt **P**ending) : 0 : interruption demandée et 1 dans le cas contraire*
- *bits 1 à 3 (ID1-ID2-ID3 : **ID**entificator interrupt bit 1-2 et 3) indiquent la nature, la source et la conséquence de l'interruption selon le tableau :*

Bit 3 ID3	Bit 2 ID2	Bit 1 ID1	Bit 0 IP	Niveau priorité	Type d'interruption	Source d'interruption	Arrêt d'interruption
0	0	0	1	aucun	aucune	aucune	aucune
0	1	1	0	1	Modif état ligne réception	erreur de réception (OE, FE, PE, BI)	Lecture LSR
0	1	0	0	2	Donnée reçue disponible	Réception RBR et prêt à être lu ou seuil atteint dans FIFO RBR	Lire RBR
1	1	0	0	2	Temps écoulé	1 caractère min dans RBR FIFO et aucune action	Lire RBR
0	0	1	0	3	THR vide	THRE→ RxRDY	R-IIR ou W-THR
0	0	0	0	4	Modif état modem	CTS, DSR, RI, DCD ont changé	Lecture MSR

- *bit 4-5 : 0 (réservés)*
- *bit 6-7 : FE1 et FE2 (**F**IFO **E**nable ? bit 1 et 2) : Validation FIFO. FE2 mis à 1 si bit 0 de FCR (FE)= 1. FE2 = 0→ FIFOs inactivées. FE2 = 1 → FIFOs activée. FE1 = 0 → FIFO activée mais inutilisable*

III.3.5 FCR : réglages des FIFO

***FCR** : **F**IFO **C**ontrol **R**egister – registre de contrôle des FIFO*

L'accès se fait à l'adresse relative +2 en écriture seule. Ce registre valide, efface les FIFO, règle le niveau du trigger (en octets) des tampons et sélectionne le valeur du signal en DMA.

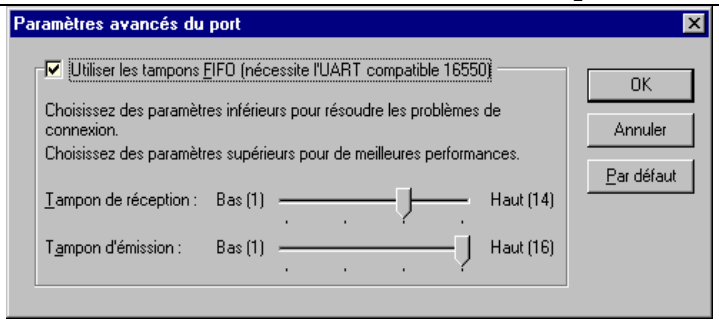
- *bit 0 (FE : **F**ifo **E**nable) :1 pour valider les FIFO RBR et THR. 0 efface la totalité des FIFO et empêche l'écriture des autres bits du registre.*
- *bit 1 (RFR : **R**eceiver **F**ifo **R**eset) : efface contenu FIFO RBR (et pas le RSR lui-même). bistable (revient à 0 automatiquement après usage).*
- *bit 2 (TFR : **T**ransmit **F**ifo **R**eset) : efface contenu FIFO THR (et pas le TSR lui-même). bistable (revient à 0 automatiquement après usage).*
- *bit 3 (DMS : **D**ma **M**ode **S**elect) :RxRDY et TxRDY passent de 1 à 0 (deviennent inactifs).*
- *bit 4-5 (TTL-TTM : **T**ransmit **T**rigger level **L**SB-**M**SB) : définissent le seuil max à atteindre (en octets)sur FIFO THR (=TFR) avant de déclencher une interruption FIFO.*

Bit 5	Bit 4	seuil
0	0	1 octet
0	1	4 octets
1	0	8 octets
1	1	16 octets

- *bit 6-7 (RTL-RTM : **R**eceiver **T**rigger level **L**SB-**M**SB) : mêmes niveaux mais pour FIFO RBR (RFR)*

Bit 7	Bit 6	seuil
0	0	1 octet
0	1	4 octets
1	0	8 octets
1	1	16 octets

exemple de réglage des seuils sur PC:



Fonctionnement des FIFOs :

- *En réception, lorsqu'un UART reçoit un octet par son entrée série, elle le "désérialise" et le stocke dans un registre interne (RBR) et prévient (par interruption), qu'il est à disposition du programme de traitement. Les données successives peuvent continuer à arriver et sont stockées dans le FIFO. Le processeur doit lire le RBR (et ainsi le vider) avant que le FIFO RBR soit plein. Ceci exige un traitement des interruptions rapide.*
- *En émission, lorsqu'un UART veut émettre, il reçoit en parallèle les données destinées au registre d'émission (THR). Il doit attendre que la ligne soit libre (handshaking d'autorisation d'émettre), puis il "sérialise" et prévient par une interruption que le registre de sortie (THR) est libre. Avantage de FIFO THR : pas besoin d'attendre que le registre à décalage ait fini avant d'envoyer les autres données vers THR puis FIFO THR. Attente seulement lorsque FIFO THR plein.*
- *Dans un sens comme dans l'autre,on voit que les registres RBR et THR ont gagné à être remplacé par des FIFO pouvant emmagasiner plusieurs octets et dont les mécanismes d'enregistrement (push) et de récupération (pop) sont nettement plus rapides que les accès classiques (aléatoires).*
- *Les piles du type FIFO font partie de ces mécanismes. En contrepartie de leur rapidité, elles ne peuvent récupérer qu'un seul octet à la fois (le dernier entré). On comprendra que l'effet tampon est d'autant plus efficace que le nombre d'octets pouvant être stocké est grand. Ce nombre est réglable (figure, bits 4-7). Si le nombre d'octets est suffisant, le processeur ne sera pas interrompu pendant ses activités et aura le temps de vider les FIFO entre des écritures (des lectures) dans ce buffer.*

III.3.6 LCR : réglages de la transmission

***LCR** : **L**ine **C**ontrol **R**egister – registre de contrôle de la ligne*
L'accès se fait à l'adresse relative +3. Ce registre permet de spécifier le format de la trame de la liaison série (vitesse de transmission, nombre de bit de données pour le codage du caractère, nombre de bits stop et type de parité). Il permet l'accès aux registres de réception (RBR) et d'émission (THR).

- *bits 0 et 1 (WLS0 et 1 : **W**orld **L**ength **S**elect bit 0 et 1) : nombre de bits du caractère (codage sur 5, 6, 7 ou 8 bits)*
- *bit 2 (STB : **ST**op **B**its) : nombre de bits Stop*

Bit 2	*Bit 1*	*Bit 0*	*codage*	*nbre STOP bits*
0	*x*	*x*	*5-6-7 ou 8*	*1*
1	*0*	*0*	*5*	*1.5*
1	*0*	*1*	*6*	*2*
1	*1*	*0*	*7*	*2*
1	*1*	*1*	*8*	*2*

- *bit 3 (PEN : **P**arity **EN**able) : validation de parité.*
- *bit 4 (EPS : **E**ven **P**arity **S**elect) : parité paire ou impaire.*
- *bit 5 (SP : **S**ticky **P**arity) : forcage de parité (inverse de EPS).*

Bit 5	*Bit 4*	*Bit 3*	*type parité*
x	*x*	*0*	*(NP : No Parity) → aucune parité*
0	*0*	*1*	*(OP : Odd Parity) → parité impaire*
0	*1*	*1*	*(EP : Even Parity) → parité paire*
1	*0*	*1*	*(HP : High Parity) → parité haute (1)*
1	*1*	*1*	*(LP : Low Parity) → parité basse (0)*

- *bit 6 (BC : **B**reak **C**ontrol) : contrôle de break (0 = hors service, 1 = mise à zéro de la sortie TXD tant que BC=1).*
- *bit 7 (DLAB : **D**ivisor **L**atch **A**ccess **B**it) : DLAB=1, autorise l'accès aux registres de division DLL et DLM. DLAB = 0, permet l'accès aux registres RBR, THR et IER*

III.3.7 MCR : réglages du modem

***MCR** : **M**odem **C**ontrol **R**egister – registre de contrôle du modem*
L'accès se fait à l'adresse relative +4.

- *bit 0 (DTR : **D**ata **T**erminal **R**eady) : force la ligne DTR dans l'état indiqué*
- *bit 1 (RTS : **R**equest **T**o **S**end) : force la ligne RTS dans l'état indiqué*
- *bit 2 (OUT1) : valide (1) ou invalide (0) la sortie OUT1 (en la forçant à 1)*
- *bit 3 (OUT 2) : idem pour OUT2*
- *bit 4 (LOOP) : Si 1, simule un **LOOP**-back, permet de tester l'UART en détectant si la défaillance est matérielle (rupture,pb de ligne) ou logicielle (pb de configuration).*
- *bit 5 (AFE : **A**uto**F**low control **E**nable) : si 1, mode diagnostic : la donnée émise est immédiatement reçue. Cela permet de vérifier le bon fonctionnement du transfert de donnée (bus interne).*
- *bits 6-7 : 0*

III.3.8 LSR : état de la transmission

***LSR** : **L**ine **S**tatus **R**egister – registre d'état de la ligne*
L'accès se fait à l'adresse relative +5 en lecture seule. Ce registre rend compte de l'état de la transmission. Il permet de connaître l'état des buffers de réception (bit 0 à 4) et de transmission (5 et 6) et de détecter une éventuelle erreur de transmission. Un lecture du LSR remet à 0 les bits 1 à 4.

- *bit 0 (DR : **D**ata **R**eady) : 1 si donnée entièrement reçue et transférée dans le RBR (ou FIFO RBR). Remis à 0 par opération de lecture du RBR (ou FIFO RBR).*
- *bit 1 (OE : **O**verrun **E**rror) : 1 si survitesse, un caractère nouveau vient d'arriver alors que le précédent n'a pas encore été lu ou bien remplissage de FIFO au delà du seuil préconisé*
- *bit 2 (PE : **P**arity **E**rror) : 1 si une erreur de parité est détectée. Avec les FIFO, l'erreur s'applique au dernier caractère sortant (first out).*
- *bit 3 (FE : **F**raming **E**rror) :1 si une erreur de format (nombre de bit de stop incorrect) est détectée. Avec les FIFO, l'erreur s'applique au dernier caractère sortant (first out). Après une erreur, l'UART prend 2 bits pour « resynchroniser ».*
- *bit 4 (BI : **B**reak **I**ndicator) : 1 si une erreur de break est détectée (idle time trop long, vitesses de transmission différentes entre ordinateurs). Avec les FIFO, l'erreur s'applique au dernier caractère sortant (first out). Après une erreur, l'UART attend 2 instants d'horloge au niveau sur SIN pour autoriser le prochain caractère à traiter.*
- *bit 5 (THRE : **T**ransmit **H**olding **R**egister **E**mpty) :1 si THR est vide, prêt à attendre un prochaine donnée (mis à 1 lorsque le transfert de THR vers TSR est effectué). Il est remis à zéro par le rechargement de THR. Avec les FIFO, mis à 1 lorsque FIFO THR est vide.*
- *bit 6 (TSRE : **T**ransmit **S**hift **R**egister **E**mpty) : 1 si THR <u>et</u> TSR sont vides. Il est remis à zéro par le chargement de THR. avec les FIFO, 1 lorsque FIFO <u>et</u> TSR vides.*
- *bit 7 (RFE : **R**BR **F**IFO **E**rror) : 1 si une erreur (PE, FE, BI) est survenue dans le FIFO RBR.*

III.3.9 MSR : état du modem

***MSR** : **M**odem **S**tatus **R**egister – registre d'état du modem*
L'accès se fait à l'adresse relative +6 en lecture seule. Ce registre donne l'état des lignes de contrôle du modem.

- *bit 0 (DCTS : **D**elta **C**lear **T**o **S**end) : 1 si CTS a changé d'état depuis la dernière lecture.*
- *bit 1 (DDSR : **D**elta **D**ata **S**et **R**eady) : 1 si DSR a changé d'état depuis la dernière lecture.*
- *bit 2 (TERI : **T**railing **E**dge **I**ndicator **R**ing) : 1 si RI est passée de l'état haut à bas depuis la dernière lecture.*
- *bit 3 (DDCD : **D**elta **D**ata **C**arrier **D**etect) : 1 si DCD a changé d'état depuis la dernière lecture.*

- *bit 4 (CTS : **C**lear **T**o **S**end) :1 si CTS=0. En mode Diagnostic (MCR4=LOOP=1) égal à RTS.*
- *bit 5 (DTR : **D**ata **T**erminal **R**eady) : 1 si DTR=0. En Diagnostic (MCR4=LOOP=1) égal à DTR.*
- *bit 6 (RI : **R**ing **I**ndicator) : 1 si RI=0. En mode Diagnostic (MCR4=LOOP=1) égal à OUT1.*
- *bit 7 : (DCD : **D**ata **C**arrier **D**etect) 1 si DCD=0. En Diagnostic (MCR4=LOOP=1) égal à OUT2.*

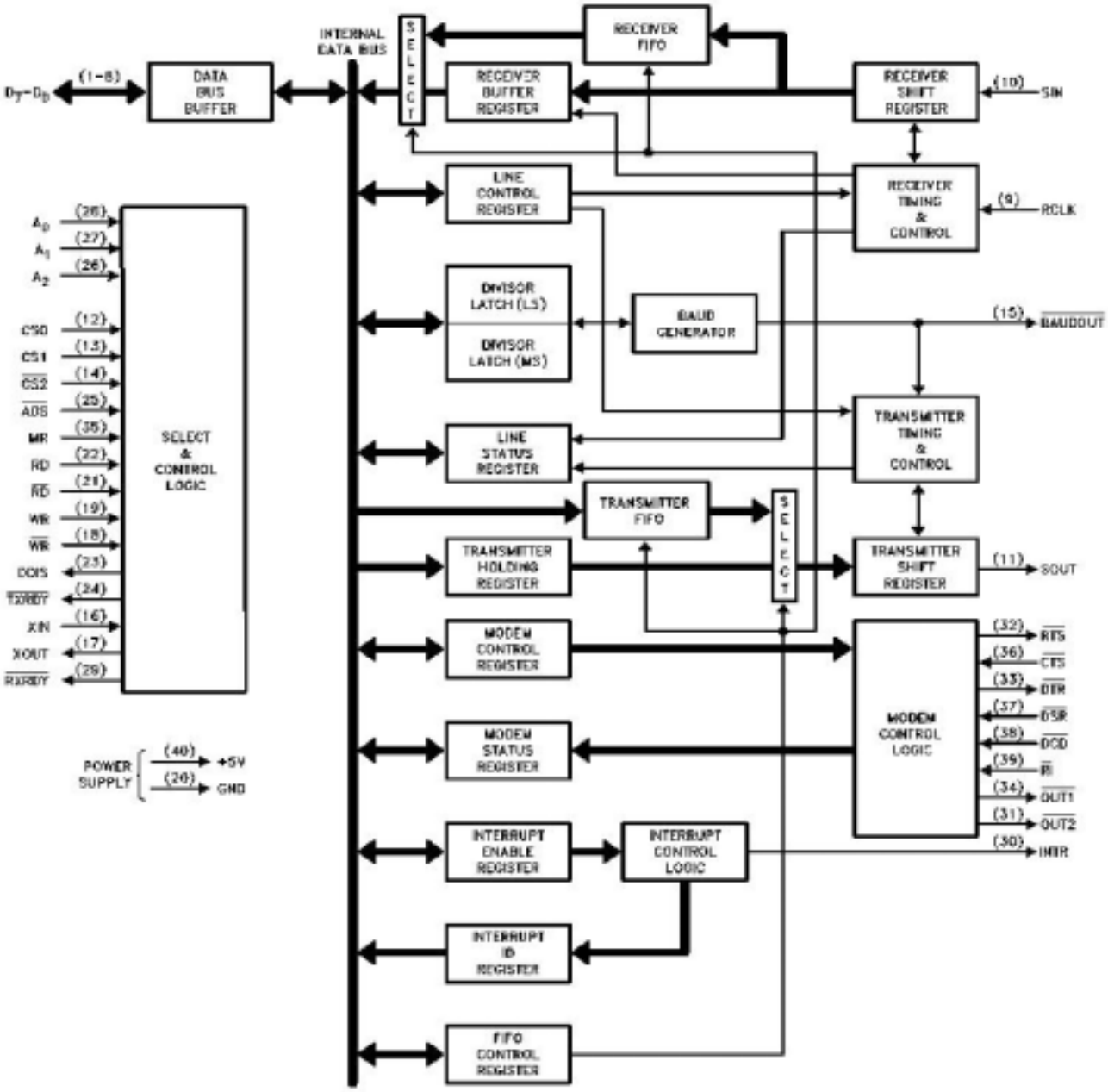
III.3.10 SCR : réglage horloge externe

***SCR** : **SC**ratch **R**egister – registre de débuggage logiciel.*

L'accès se fait à l'adresse relative +7. Ce registre n'affecte pas les autres opérations de l'UART. Il permet de spécifier une vitesse particulière de transmission via une horloge externe (connectée sur Xin) plutôt que d'utiliser les registres programmables de vitesse (DLL-DLM). Dépend de DLAB.

- *Si DLAB = 1 : bit P0 à P5 : valeur du diviseur (jusqu'à 31.5). Exemple : Si Fréquence externe = 22.118 Mhz sur XIN, diviseur (bits P0 à P5)= 5.5, alors clock UART = 4.02 Mhz. Si UART Diviseur (DLL+DLM)=1, vitesse = 250 Kbauds. Si P6=1, sélection des interruptions logiques (ILS : **I**nterrupt **L**ogic **S**elect), sinon sélection de la sortie de l'UART (OS : uart Output Select)*
- *Si DLAB = 0 : rien*

III.4 - Annexe : architecture d'un UART 16550



IV. LE BUS I²C

IV.1 - Introduction

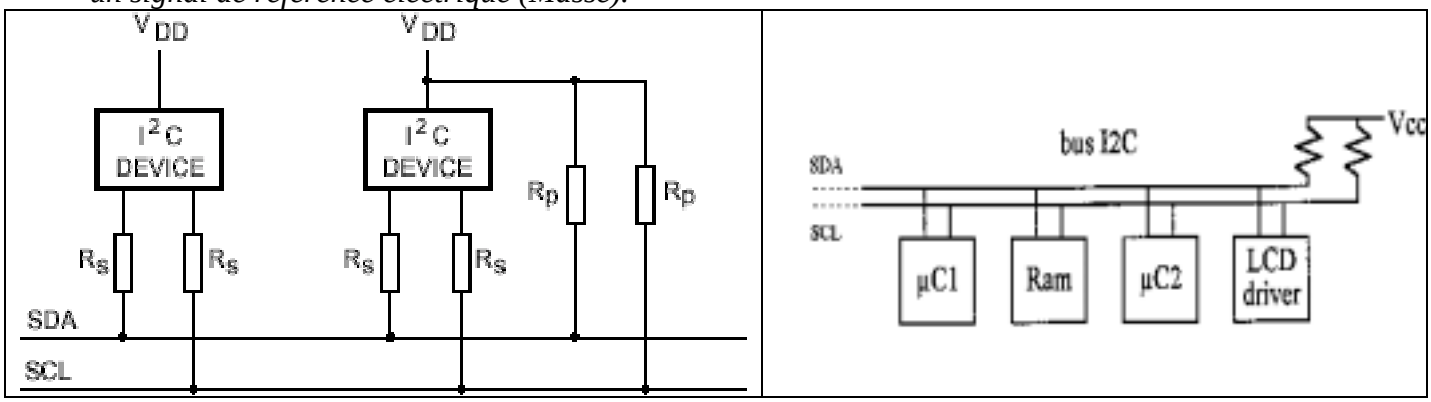
IV.1.1 Présentation:

*Le bus I2C (**I**nter **I**ntegrated **C**ircuit), externe au PC, a été développé au début des années 80 par Philips semiconductors pour permettre de relier facilement à un microprocesseur les différents circuits d'un téléviseur moderne sans augmenter les coûts. Il permet de réaliser au sein d'un appareil un micro-réseau trifilaire de type série, synchrone, multimaître à détection de collision avec hot-plugging. Un système utilisant ce bus peut représenter une alternative à bas prix à une liaison RS232 : ACCESS Bus.*

IV.1.2 Caractéristiques

Le bus I2C permet de faire communiquer entre eux des composants électroniques très divers grâce à seulement trois fils :

- *2 signaux synchrones : un signal de donnée (SDA), un signal d'horloge (SCL).*
- *un signal de référence électrique (Masse).*



Les lignes SDA, SCL sont toutes les deux reliées à l'alimentation positive Vcc (qui dépend de la technologie des composants : TTL ou CMOS) des circuits par l'intermédiaire de résistances de pull-up (R_P), l'interface I2C des circuits étant toujours à collecteurs ouverts. Le niveau résultant est alors une fonction logique ET de toutes les sorties connectées sur les lignes (Les lignes sont à l'état haut lorsque le bus est libre). R_P limite la charge capacitive et R_S les pics de tension par induction.

De nombreux fabricants ayant adopté ce système, la variété des circuits disponibles disposant d'un port I2C est important : Ports d'E/S bidirectionnels, Convertisseurs A/N et N/A, Mémoires (RAM, EPROM, EEPROM, etc...), Circuits Audio (Egaliseur, Contrôle de volume, ...) et autre LED, LCD...Le nombre important de circuits intégrés comprenant l'interface I2C offre de multiples avantages aux concepteurs de systèmes électroniques. En effet, l'atout majeur de cette liaison est de pouvoir facilement connecter en parallèle sur le bus de nouveaux éléments ou d'échanger un élément contre un autre sans restructurer le système.

Le nombre de composants qu'il est ainsi possible de relier est essentiellement limité par la charge capacitive des lignes SDA et SCL : 400 pF.

Les données sont transmises en série à l00 Kbits/s (12.5 ko/s) en mode standard et jusqu'à 400 Kbits/s (50 ko/s) en mode rapide, ce qui ouvre la porte de cette technologie à toutes les applications où la vitesse n'est pas primordiales. La version la plus récente intègre une grande vitesse allant jusqu'à 3,2 Mbits/s (425 ko/s).

Pour éviter la prise de communication simultanée de plusieurs composants, on utilise un protocole qui permet de définir les règles de communication.

IV.2 - Le protocole I2C

Le protocole I2C définit la succession des états logiques possibles sur SDA et SCL, et la façon dont doivent réagir les circuits en cas de conflits.
Il existe deux modes d'utilisation du bus :

- *le mode Maître-Esclave (1 seul circuit dirige la communication)*
- *le mode Multi-Maîtres (plusieurs circuits peuvent prendre la main sur le bus)*

Le maître de la liaison est le circuit qui génère l'horloge SCL, les données sur SDA étant générées par le maître ou par l'esclave suivant le sens de transfert réclamé par le maître. Un maître peut donc être récepteur ou émetteur.

Le protocole I2C gère la liaison et les conflits éventuels grâce à la fonction logique ET des lignes et à une procédure d'arbitrage du bus.
Le protocole est représenté par une trame du type :

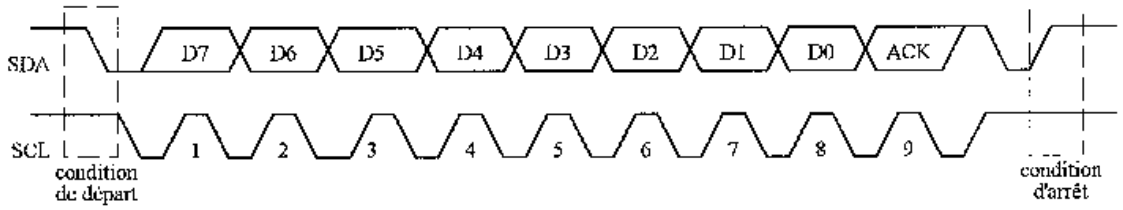
***Prise** de Parole (de Contrôle)*	***START***	***ADR** : envoi adresse périph (MSB en premier)*	***R/W***	***ACK$_{ADR}$** : accusé de réception pour réception adresse*	***DATA*** *R/W*	***ACK$_{DATA}$** : accusé de réception pour réception données*	***STOP***

IV.2.1 La prise de contrôle du bus

Un seul bit est transféré par coup d'horloge. Le bit transmis doit rester fixe pendant l'état haut de l'horloge. Si cette condition n'est pas respectée alors les changements sur la ligne sont interprétés comme des conditions de départ ou d'arrêt de transmission.

En effet, pour prendre le contrôle du bus, il faut tout d'abord que celui-ci soit au repos (SDA et SCL à '1') pendant un intervalle suffisant. Pour transmettre des données sur le bus, il faut donc surveiller deux conditions particulières :

- *La condition de départ : SDA passe à '0' alors que SCL reste à '1'*
- *La condition d'arrêt : SDA passe à '1' alors que SCL reste à '1'*



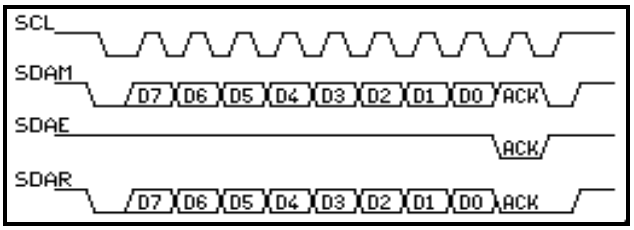
Lorsqu'un circuit, après avoir vérifié que le bus est libre, prend le contrôle de celui-ci, il en devient le maître. C'est lui qui génère le signal d'horloge.
Les autres composants doivent continuer de scruter le bus en attendant la condition de Stop

IV.2.2 La transmission d'un octet

*Les conditions de Start et Stop sont donc imposées par le maître (quel que soit le sens du transfert). Un nombre indéfini d'octets peuvent être transmis entre un Start et un Stop. Chaque octet est transmis sur 9 bits, le 9ème bit servant d'accusé de réception (**ACK**nowledge). C'est l'esclave récepteur qui génère le bit ACK après chaque octet reçu du maître émetteur ou bien c'est le maître récepteur qui le génère après chaque octet de l'esclave émetteur.*

Après avoir imposé la condition de départ, le maître applique sur SDA le bit de poids fort D7 (c'est lui qui est transmis en premier). Il valide ensuite la donnée en appliquant pendant un instant un niveau '1' sur la ligne SCL. Lorsque SCL revient à '0', il recommence l'opération jusqu'à ce que l'octet complet soit transmis. Il envoie alors un bit ACK à '1' tout en scrutant l'état réel de SDA.

L'esclave doit alors imposer un niveau '0' pour signaler au maître que la transmission s'est effectuée correctement. Les sorties de chacun étant à collecteur ouvert, le maître voit le '0' et peut alors passer à la suite.



- *SCL : Horloge imposée par le maître.*
- *SDAM : SDA imposés par le maître.*
- *SDAE : SDA imposés par l'esclave*
- *SDAR : SDA réels résultants*

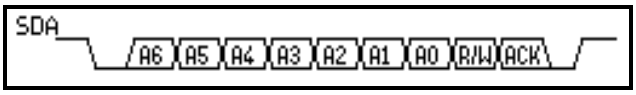
IV.2.3 La transmission d'une adresse

Le nombre de composants qu'il est possible de connecter sur un bus I2C étant largement supérieur à deux, il est nécessaire de définir pour chacun une adresse unique.

Au cours d'une transmission, le maître commence toujours à sélectionner un esclave à l'aide d'une adresse et par indiquer à l'esclave le sens de transmission.

L'adresse d'un circuit est codée sur 7 bits complétée par un bit R/W qui permet au maître de signaler s'il veut lire ou écrire une donnée. Le bit d'acquittement ACK fonctionne comme pour une donnée, ceci permet au maître de vérifier si l'esclave est disponible.

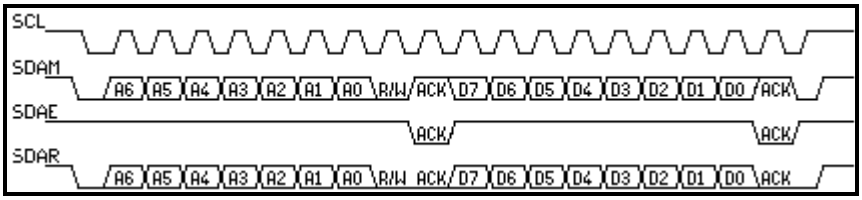
Le 1er octet d'une trame de dialogue comprend ainsi une adresse d'esclave sur 7 bit et un 8ème bit R/W précisant si le maître veut émettre ou recevoir les données.



- **Cas particulier des mémoires :** *L'espace adressable d'un circuit de mémoire étant sensiblement plus grand que la plupart des autres types de circuits, l'adresse d'une information y est codée sur deux octets ou plus. Le premier représente toujours l'adresse(physique) du circuit et les suivants l'adresse interne de la mémoire.*
- **Les adresses réservées :** *certaines adresses sont réservées (modes de fonctionnement particuliers).*

IV.2.4 Ecriture d'une donnée

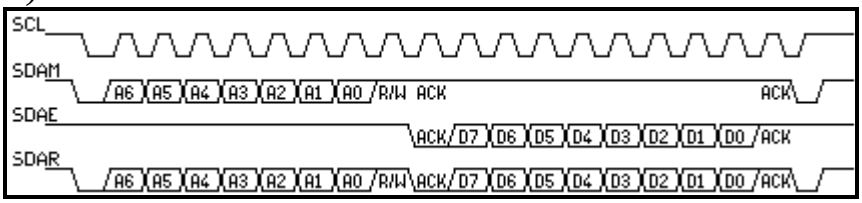
L'écriture d'une donnée par le maître ne pose pas de problème particulier. Tous les circuits connectés sur le bus détectent la condition Start (par interruption). Seul le circuit reconnaissant son adresse communique ensuite avec le maître.



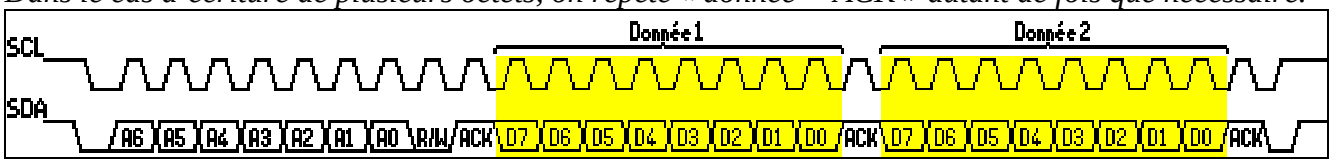
L'écriture d'un octet dans certains composants (mémoires, microcontrôleur,...) peut prendre un certain temps. Il est donc possible que le maître soit obligé d'attendre l'acquittement ACK avant de passer à la suite.

IV.2.5 Lecture d'une donnée

La lecture d'une donnée par le maître se caractérise par l'utilisation spéciale qui est faite du bit ACK. Après la lecture d'un octet, le maître positionne ACK à '0' s'il veut lire la donnée suivante (cas d'une mémoire par exemple) ou à '1' le cas échéant. Il envoie alors la condition d'arrêt.



Dans le cas d'écriture de plusieurs octets, on répète « donnée + ACK » autant de fois que nécessaire.



IV.2.6 Récapitulatif

IV.3 - La gestion des conflits

IV.3.1 Mise en situation

La structure même du bus I2C a été conçue pour pouvoir y accueillir plusieurs maîtres. Se pose alors le problème commun à tout les réseaux utilisant un canal de communication unique : le contrôle du bus. En effet, chaque maître pouvant prendre possession du bus dès que celui-ci est libre, il existe la possibilité que deux maîtres désirent prendre la main en même temps. Si cela ne pose pas de problème sur le plan électrique grâce à l'utilisation de collecteurs ouverts, il faut pouvoir détecter cet état pour éviter la corruption des données transmises.

IV.3.2 Principe

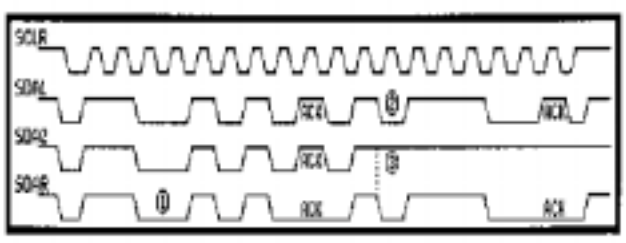
Comme nous l'avons vu précédemment, pour prendre le contrôle du bus, un maître potentiel doit d'abord vérifier que celui-ci soit libre et qu'une condition d'arrêt a bien été envoyée depuis au moins 4,7 µs. Mais il reste la possibilité que plusieurs maîtres prennent le contrôle du bus simultanément.

Chaque circuit vérifie en permanence l'état des lignes SDA et SCL y compris lorsqu'ils sont eux même en train d'envoyer des données. On distingue alors plusieurs cas :

- *Les différents maîtres envoient les mêmes données au même moment : les données ne sont pas corrompues, la transmission s'effectue normalement, comme si un seul maître avait parlé. Ce cas est très rare.*
- *Un maître impose un '0' sur le bus : il relira forcément '0' et continuera à transmettre. Il ne peut pas alors détecter un éventuel conflit.*
- *Un maître cherche à appliquer un '1' sur le bus : s'il ne relit pas un niveau '1', c'est qu'un autre maître a pris la main en même temps. Le premier perd alors immédiatement le contrôle du bus, pour ne pas perturber la transmission du second. Il continue néanmoins à lire les données au cas où celles-ci lui auraient été destinées.*

IV.3.3 Exemple de conflit

Soit le chronogramme suivant :



- SCLR: Horloge résultante.
- SDA 1: Niveaux de SDA imposés par le maître n° 1.
- SDA 2 : Niveaux de SDA imposés par le maître n°2.
- SDA R: Niveaux de SDA réels résultants lus par les deux maîtres.

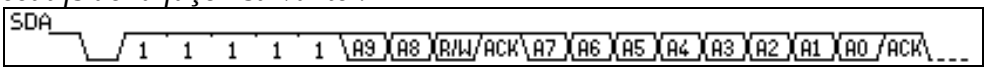
Analyse

- *Le premier octet est transmis normalement car les deux maîtres imposent les même données. (Cas n°1). Le bit ACK est mis à '0' par l'esclave.*
- *Lors du deuxième octet, le maître n°2 cherche à imposer un '1' (SDA2), mais relit un '0' (SDAR), il perd alors le contrôle du bus et devient esclave (Cas n°3). I1 reprendra le contrôle du bus lorsque celui-ci sera de nouveau libre. Le maître n°1 ne voit pas le conflit et continue à transmettre normalement (Cas n°2).*
- *En conclusion, l'esclave à reçu les données du maître n°1 sans erreurs et le conflit est passé inaperçu.*

IV.3.4 Nouvelles caractéristiques

Afin de compenser quelques lacunes des premières spécifications du bus I2C, quelques nouvelles améliorations ont été apportées :

- ***Le mode rapide** : le bus à désormais la capacité de transmettre des données jusqu'à une vitesse de 3.4 Mbits/s (0.425 Mo/s).*
- ***Des entrées à Trigger de Schmitt** : afin de limiter la sensibilité au bruit.*
- *La mise en **haute impédance** d'un circuit non alimenté : ceci évite de bloquer le bus si un périphérique n'est pas alimenté.*
- *Extension à **10 bits de l'adressage** des circuits : l'adressage d'un circuit se fait maintenant dans deux octets consécutifs de la façon suivante :*



Attention, dans ce cas, la plage d'adressage totale (adresses réservées comprises) commence à 0x7B00 !!

IV.3.5 Les adresses réservées

Certaines adresses ne sont pas utilisées pour l'adressage de composants → adresses réservées.
Attention : 2 cas possibles selon adressage 8 bits ou bien 10 bits.

adresse	fonction	description
0000 0000	appel général	*Après l'émission d'un appel général, les circuits ayant la capacité de traiter ce genre d'appel émettent un acquittement. Le deuxième octet permet de définir le contenu de l'appel*
0000 0001	octet de Start	*Utilisé pour synchroniser les périphériques lents avec les rapides*
0000 001x	protocole Cbus	*Tous les circuits I2C deviennent « sourds ».* *On peut transmettre ce que l'on veut sur le bus.* *Retour à la normale dès détection d'une condition d'arrêt.*
0000 010x	autres protocoles	
0000 0110	Reset	*Remet tout les registres des circuits connectés dans leur état initial (équivalent à celui lors de la mise sous tension).* *Les circuits qui en sont capables rechargent leur adresse d'esclave*
0000 0111		*Réservé pour usage futur (ex : débuggage réseau)*
0000 1xxx	High Speed	*Passage en mode High Speed (3.4 Mbits/s)*
1111 0xxx	pour adressage 10 bits	*Réservées en adressage 8 bits* ***mais** utilisées en adressage 10 bits*
1111 1xxx		*Réservées pour usage futur (ex : débuggage réseau)*

V. BUS USB

V.1 - Introduction

La démocratisation d'appareils numériques grand public nécessitant des taux de transfert soutenus (camescope, scanner, appareils photos numériques, etc…), a obligé les constructeurs à développer des interfaces plus rapides, plus universelles tout en pouvant y raccorder un maximum de ces périphériques hétéroclites.

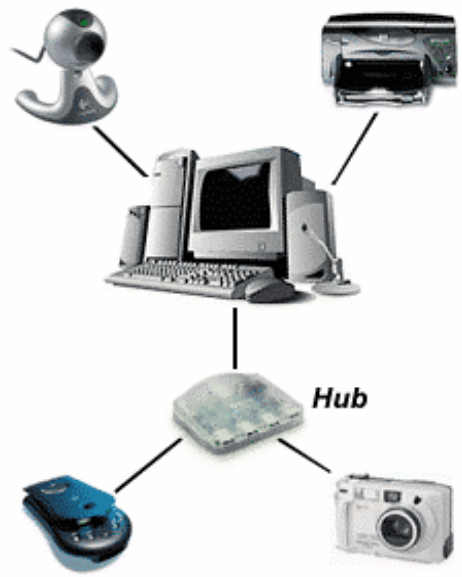
*Les ports série / parallèle, avec leurs caractéristiques désuètes, voient leur utilité décroître au profit de récentes interfaces comme l'USB (**U**niversal **S**erial **B**us), le FireWire (domaine grand public) ou d'autres plus anciennes mais qui ne cessent d'être améliorée comme le SCSI (domaine professionnel). Nous allons étudier l'interface USB qui représente une nouvelle liaison série **synchrone**.*

☞ USB 1.1 & 2.0

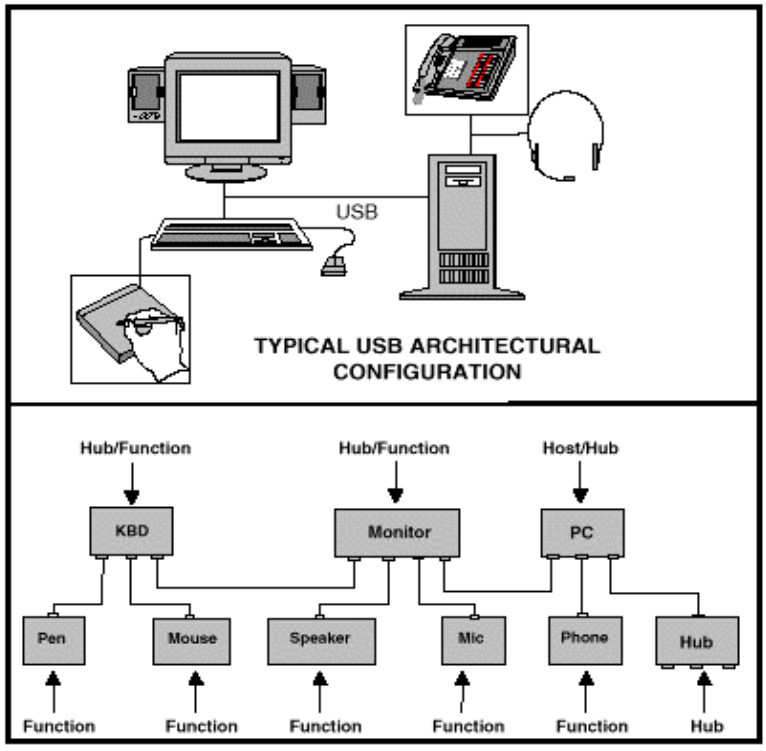
La première version de l´Universal Serial Bus, l´interface USB 1.1 permet de connecter à un micro-ordinateur jusqu´à 127 (adressage sur 7 bits) périphériques pour un débit maximal de 12 Mbits/s, (soit 1.5 Mo/s), le tout sur un bus série comme son nom l´indique. Principalement destiné aux périphériques lents, l´interface a été développée par Compaq, HP, Intel, Lucent, Microsoft, NEC et Philips depuis 1994 pour aboutir sur le marché grand public en 1998.

L´USB 2.0 (12/2000) s´enrichit d´une troisième vitesse de connexion entre le Host (le « maître ») et les périphériques USB 2.0 : il est maintenant possible de relier des disques durs externes, imprimantes, scanners et autres lecteurs à des vitesses frôlant les 480 Mbits/s (soit 60 Mo/s). Le gain en vitesse a été notamment possible grâce à la réduction du voltage des signaux transmis dans les câbles, ceux-ci passant de 3.3V à 0.4V. Les périphériques répondant à la norme 2.0 ne peuvent pas bénéficier de la vitesse haute s´ils sont raccordés derrière un HUB USB 1.1. Il y a un gros "boum" des chipsets récents qui intègrent tous la norme USB 2.0 (depuis milieu 2002-2003).

Parmi les périphériques pouvant être raccordés au micro-ordinateur, nous pouvons citer claviers, souris, joysticks, scanners, imprimantes, téléphones, assistants personnels, baladeurs mp3, lecteurs/graveurs de cd-rom, disques optiques et magnétiques, webcams, modems, etc…. Autant dire que les constructeurs pensent que cette interface est vouée à remplacer à court terme toutes les autres interfaces d´entrées/sorties de nos micro-ordinateurs.



Structure
étoile série
(topologie logique)



topologie physique

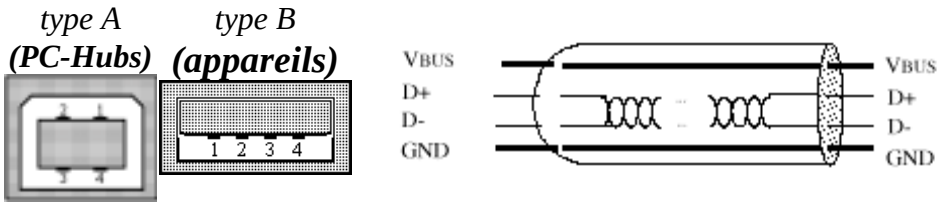
Les principaux critères du développement de ce bus ont été les suivants :

- *3 débits cohabitent faible, pleine et haute vitesse (Low Speed 1.5 Mbit/s, Full (Medium) Speed 12 Mbit/s et High Speed 480 Mbit/s pour USB 2.0)*
- *Facilité d'emploi et d'utilisation : hot plug & play (détection du changement de la tension entre D+ et D-), hot swapping*
- *Une seule IRQ pour tous les périphériques connectés au port (fin aux pb de conflits)*
- *Energie (faible : 100 mA) distribuée par le câble (moins de prises)*
- *Bas-prix pour les débits « faibles » (12 Mbit/s = 1.5Mo/s)*
- *Transmission en temp réel (voix, audio, vidéo comprimée)*
- *Protocole souple pour différents modes :*
  - *Transmission de paquets*
  - *Mode isochrone*
  - *Messages courts*
  - *Séparation des commandes et des données*
- *Adaptation à différentes configuration sur le terrain*
- *Adaptation aisée à de nouveaux types de périphériques (adressage dynamique)*
- *Support de l´architecture CTI (Computer Telephony Integration)*
- *Topologie en étoile (via des Hubs) ou en bus (chainés les uns aux autres)*
- *Nombre maximum de périphériques élevé (127). Récemment, carte multi-ports gérée par 2 ou + controleurs.*
- *principalement half-duplex*

☞ Câbles

Le bus USB nécessite une connectivité propre composée de connecteurs de deux types, les connecteurs de type A disposés aux entrées du contrôleur maître et des HUB (« upstream »), et des connecteurs de type B sur les périphériques (« downstream »), ainsi un seul type de câble est nécessaire pour relier tous périphériques au micro-ordinateur.

connecteurs A et B
1 (rouge) : alim (Vbus)
2 (blanc) : D-
3 (vert) : D+
4 (noir) : masse (GND)



Ce câble se compose de 4 fils, une paire torsadée pour le transfert des données, un fil au potentiel de +5V qui permet d´alimenter les périphériques USB si nécessaire et enfin la masse (téléalimentation). Il peut être blindé ou non, le mode basse vitesse de 1.5 Mbits/s ayant une tolérance supérieure aux perturbations électromagnétiques. Un blindage est fortement recommandé pour une utilisation pleine vitesse à 12 Mbits/s (longueur max de 5 mètres de câble entre 2 éléments).
Enfin, un autre atout de ce bus est qu´il peut transporter l´alimentation des périphériques s´y raccordant, dans la limite de 500 mA pour un appareil relié à un port le permettant.

☞ Topologie

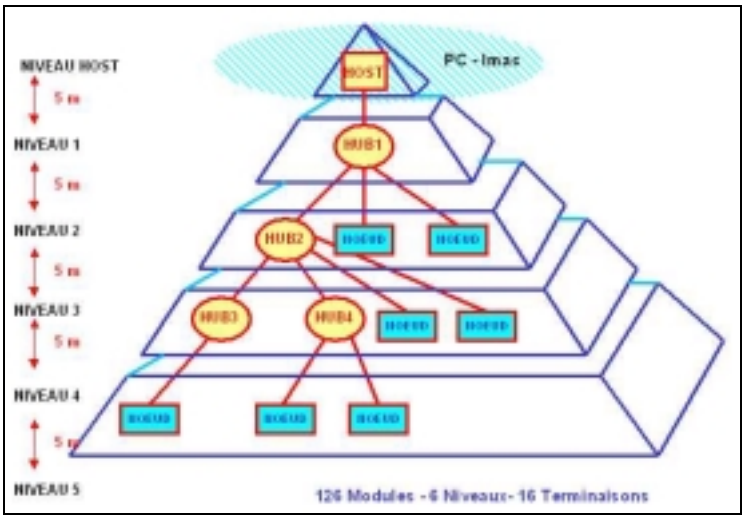
*L´USB propose une topologie physique **peer-to-peer**, il établit donc des connexions point à point entre le contrôleur maître (host) et chaque périphérique branché (device).*

Même si théoriquement, la topologie du bus USB permet de connecter dans n´importe quel ordre des périphériques, certaines règles doivent être respectées. Ainsi, une architecture USB se construit à base de 3 éléments :

- *Un contrôleur maître USB ("Host") + Root Hub*
- *Des périphériques USB ("Devices") nommés de manière plus générale « fonctions »*
- *Des concentrateurs : "Hubs" qui sont des périphériques particuliers disposant de plusieurs ports de connexion au bus USB. Port unique côté « upstream » et multiple côté « downstream ».*

*La structure de connexion des périphériques USB au contrôleur maître se fait sous forme d´**étoile à étage** ("tiered star") , où le centre est le contrôleur maître et les extrémités de celle-ci les périphériques (voir figure architecture).*

Un élément supplémentaire s´avère indispensable à la connexion d´un certain nombre de périphériques USB au contrôleur principal (situé dans le South Bridge du chipset de la carte mère ou sur une carte additionnelle). en générale, on ne dispose pas de plus de 5 connecteurs USB. Il devient alors nécessaire d´introduire dans l´arborescence des ***HUB USB*** *:*



La topologie physique du bus USB devient ainsi un ***arbre****, où les noeuds sont alors les HUB et les feuilles les périphériques.*

Remarques *:*

- *En étudiant la topologie logique de l'USB, on met en évidence la nature partagée du bus.Chaque logiciel client (côté « host ») qui va manipuler une fonction (côté « device ») la verra comme si elle était la seule sur le bus. Les concepteurs de logiciel n'ont plus à se préoccuper des problèmes liés au partage du bus.*
- *Pas d'interconnexion possible entre 2 PC (ou de manière équivalente pas de partage de périphériques par 2 PC). Il ne peut y avoir qu'un host par bus (pas d'arrangement multi-maître). La nouvelle spécification USB 2.0 (2002) « on-the-go » introduit une possibilité d'arbitrage (négociation) pour le choix du rôle de host (reservé à une connexion simple p2p entre 2 « hosts »)).*
- *2 types de HUB USB : ceux possédant un circuit d´alimentation individuel pour chaque port (hub « classique ») et d´autres dépourvus de cette caractéristique (hub interne au clavier).*
- *3 contrôleurs d'hôte :*
  - *contrôleur UHCI (Intel) : aspect logiciel > matériel.*
  - *contrôleur OHCI (Compaq, Microsoft, NSC) : aspect matériel>logiciel.*
  - *contrôleur EHCI (Standard USB 2.0) : 1 seul driver.*

V.2 - Transactions & Transferts

La possibilité d´une telle topologie entraîne des conséquences quant aux protocoles d´échanges de données entre le PC et les périphériques :

☞ *2 modes de* ***transmissions*** *:*

- *mode* ***asynchrone*** *: comme un port série classique*
- *mode* ***isochrone*** *: permet une communication périodique et continue entre le contrôleur maître et les périphériques (à chaque milliseconde précisément, le contrôleur maître transmet un paquet pour maintenir tous les périphériques synchronisés).*

☞ *préliminaire à la communication : "****PIPE****"*

- *réservation d'un canal entre logiciel du Host et End-Point du périphérique*

☞ *3 phases de* ***transactions*** *(protocoles) :*

- *Paquet Jeton (TOKEN) : Un jeton est émis par le host (obligatoire), puis un PID (Packet ID) identifie le type de transaction*
- *Paquet Données (DATA) : donnée utile (payload), optionnel*
- *Paquet d'Etat : (HANDSHAKE) : validation transactions, erreurs.*

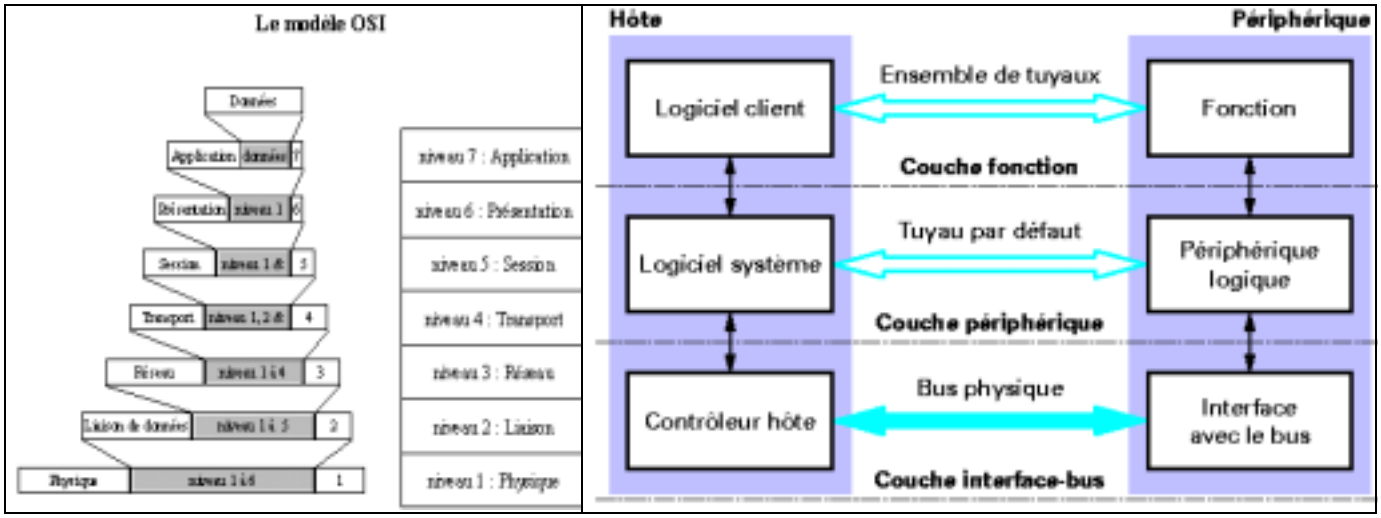
☞ *4 modes de **transferts** des paquets (terminaisons):*

- *Control Transfers : (paquets de contrôle : Message) : configuration, états, initialisation des périphériques.*
- *Bulk Data Transfers : (paquets en bloc, en rafale de données → imprimantes, scanners.*
- *Interrupts Transfers : (paquets d´interruption) : message court émis de temps en temps (l'USB ne supporte pas les interruptions)*
- *Isochronous Transfers : (paquets isochrones) : transmission à intervalles de temps réguliers et à bande passante garantie.*

☞ ***Séquencement** : bus synchrone : 1 ms (= bus cadencé à 1 KHz)*

V.3 - Protocole USB

Le fonctionnement de l'USB est présenté sous forme d'un système séparé en 3 couches (les 3 couches basses du modèle standard OSI). Cette représentation permet à chaque intervenant dans la réalisation d'un élément USB de bien délimiter son travail en respectant les spécifications attendues par les autres intervenants.



*La communication se fait entre le Host (ordinateur) et les Device (périphériques) selon le principe de l'anneau à jeton (**Token Ring**) : la bande passante est partagée temporellement entre tous les périphériques connectés et l'hôte émet un signal de début de séquence (« Frame ») chaque milliseconde, intervalle de temps pendant lequel il va donner simultanément la parole à chacun d'entre-eux (sensation de connexion privilégiée).*

*Le protocole de communication est qualifié de **« host scheduled token based protocol »** : le host initie toutes les transactions. Le premier paquet (le jeton) est produit par le host pour décrire ce qui va suivre, si la transaction de données sera en lecture ou en écriture, ce que sera l'adresse de l'appareil et la terminaison (mode de transfert) désignée. Le jeton contient donc une grande partie de l'information sur la nature de la communication.*

*Lors de la connection d'un nouveau périphérique, le host procède à la phase d'**énumération** ce qui consiste en. Le jeton contient l'adresse du périphérique codée sur 7 bits=2^7=128 périphériques-1 (car adresse '0' est réservée, adresse par défaut avant affectation) désigne le périphérique concerné. Si ce dernier reconnaît son adresse dans le jeton, il envoie un paquet de données en réponse, sinon, il fait suivre aux autres périphériques qui lui sont chaînés. En détail, un client envoie au système USB une requête transformée en IRP (**I**nput-output **R**equest **P**acket) par le driver USB. Le host gère la liste des IRP (énumération). Chacune se concrétise par la succession d'une ou de plusieurs transactions (s'il faut fragmenter les données) sur le bus. Chaque transaction se décompose alors en la succession d'au plus 3 paquets dont les rôles sont bien définis (token, data, handshake) dans les protocoles.*

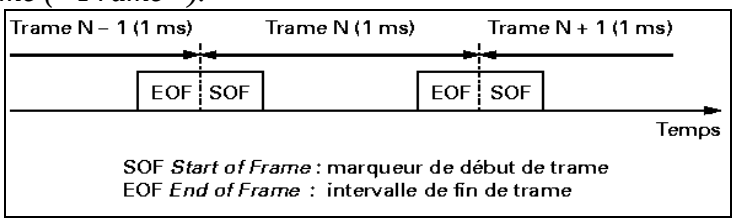
V.3.1 Pipes

*Définit au niveau de la couche fonction, un « **Pipe** » est un préliminaire à la communication :*

- *réservation d'un canal entre logiciel du Host et la terminaison (« End-Point ») du périphérique*
- *à chaque canal est associé :*
  - *la bande passante requise*
  - *les caractéristiques du End-Point (direction, taille max données, etc…)*
  - *le type de transfert de données*
- *2 type de PIPE :*
  - *STREAM : les flux de données brutes n'ont pas de format USB prédéfini. Prise en charge des transferts en Bloc, Isochrone et Interruption.*
  - *MESSAGE : les canaux ont un format USB défini. Prise en charge uniquement des transferts de commande*

V.3.2 Trame de communication

*La communication entre périphériques et hôtes sur le bus physique se présente sous la forme d'un flot de bits émis en série. C'est le SIE (**S**erial **I**nterface **E**ngine) qui s'occupe de transformer le flot de bits arrivant en parallèle dans les buffers du contrôleur hôte, en un flux de données série conforme à la spécification USB. Ce flux est constitué d'un ensemble de bits interprétables dans une fenêtre de temps déterminée. Cette taille de fenêtre est fixée à une milliseconde et l'ensemble des bits contenus dans cette fenêtre constitue une trame (« Frame »).*



Une trame est divisées en 3 parties :

- *un marqueur de début de trame (SOF : **S**tart **O**f **F**rame) qui est un paquet spécial émis par l'hôte toute les millisecondes.*
- *un ensemble de transactions de différents types entre hôte et périphériques*
- *un intervalle de fin de trame (EOF : **E**nd **O**f **F**rame) qui est une période comprise entre la dernière transaction émise dans la trame et le jeton SOF de la trame suivante.*

*Un bit isolé ne signifie rien. Un périphérique doit d'abord isoler une trame à l'intérieur de laquelle on retrouve des ensembles de paquets réalisant des fonctions particulières appelées **transactions**. C'est toujours l'hôte qui initie la transaction et c'est le périphérique qui a été sélectionné par le host qui peut alors utiliser le bus. Il reçoit la permission de parler au moment où il reçoit le jeton initiateur (SOF). La manière de distribuer le jeton et l'ordre dans lequel les paquets vont circuler sur le bus sont définis dans le protocole régissant le fonctionnement du bus.*

V.3.3 Format des paquets

Le LSB est envoyé en premier. Un paquet est composé de plusieurs champs :

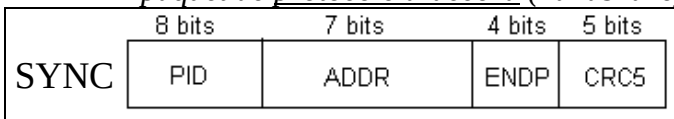
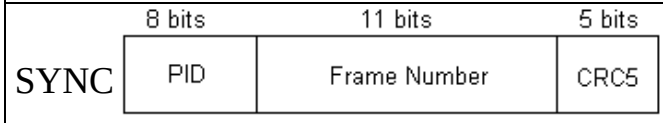
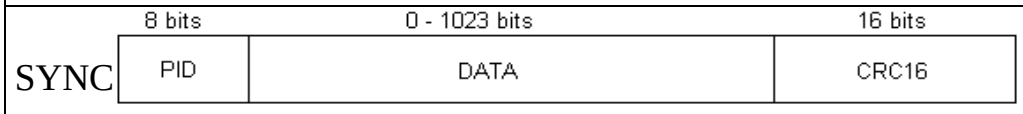
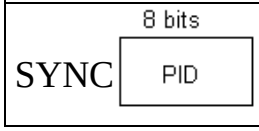
SYNC	PID	ADDR	DATA	ENDP	CRC	EOP
8-32 bits	8 bits	7 bits	0-1024 octets	4 bits	5-16 bits	3 bits

V.3.4 Types de paquets

Le format d'un paquet et les différents champs qui le composent varient d'u type de paquet à un autre. Nous allons décrire sommairement les différents types de paquets possibles :

- *paquet <u>jeton</u> : contient un champ PID, un champ ADDResse, un champ terminaison (ENDP) et un champ CRC 5. **<u>Seul</u>** le host peut envoyer ce type de paquet.*
- *paquet de <u>début de trame</u> (SOF) : composé de 3 champs : PID-Frame-CRC 5.*
- *paquet de <u>données</u> : composé de 3 champs : PID-DATA-CRC 16.*

- *paquet de protocole d'accord (handshake) : un seul champ : PID.*

	*Token (IN/OUT/SETUP)*
	*Token (SOF)*
	*Data*
	*Handshake PREamble*

V.3.5 Champs des paquets

- *SYNC : SYNChronisation de l'horloge du récepteur avec celle de l'émetteur. 8 bits (0000 0001) en low et full speed, 32 en high speed.*
- *PID : Paquet Identificateur. Un PID consiste en 4 bits suivis des compléments à un de ces 4 bits afin d'effectuer une vérification de ce PID. Le PID indique le type du paquet, son format et le type d'erreur de détection appliquée au paquet. Si la vérification du PID a échouée ou s'il n'est pas décodé comme la bonne valeur, le reste du paquet sera ignoré par le récepteur*

groupe	valeur PID	type paquet
Token (Jeton)	0001	OUT : informe le device que le host veut émettre
	1001	IN : informe le device que le host veut recevoir
	0101	SOF (Start Of Frame) : début de tableau
	1101	SETUP : configuration, début des transferts commande
Data (Données)	0011	DATA 0
	1011	DATA 1
	0111	DATA 2 : full speed,1024 max
	1111	MDATA : high speed,1024 max
Handshake (état de la transaction)	0010	ACK (acknowledge) : validation, accusé recpetion
	1010	NAK (No AcKnowledge) : le périphérique ne peut accepter (envoyer) les données sur Rx (Tx).
	1110	STALL : bloqué
	0110	NYET (No response YET) : pas encore de réponse
Special	0000	PREamble : synchroniseur initial
	1100	ERR : erreur
	1000	Split : partage, fractionnement des données
	0100	Ping : bonne connexion ?

- *ADDR : le champ adresse indique pour quelle fonction est destinée le paquet. Suivant le PID, la fonction est l'émetteur ou le récepteur. Chaque fonction USB doit répondre à une adresse unique. Les périphériques répondent à l'adresse 0 lorsqu'ils viennent d'être connectés sur le bus. Ensuite, c'est l'hôte qui assigne une adresse unique à chacun. Sa longueur de 7 bits lui permet d'adresser 127 appareils (2^7=128) puisque l'adresse 0 est réservée lors de l'énumération (interrogation de tous les périphériques par l'envoi d'un jeton par le host afin de connaître l'ensemble des adresses pour pouvoir attribuer une nouvelle adresse au nouveau périphérique qui, en retour, s'identifie).*
- *FRAME : le champ de trame permet d'identifier le numéro de la trame dans laquelle les paquets sont envoyés. Ce numéro, codé sur 11 bits, est incrémenté à chaque nouvelle création de trame, c'est à dire toutes les millisecondes. Ce champ, n'existe qu'une fois par trame dans le jeton SOF.*
- *DATA : Le champ données peut aller de 0 à 1024 bits(128 octets). 8 octets en low speed, 1023 en full speed, 1024 en high speed.*

- *ENDP : un ajout de 4 bits d'ENDP permet un adressage des fonctions plus flexible (dans le cas où plus d'une sous-voie est utilisée). Les numéros de point d'arrêt sont des fonctions spécifiques. Ce champ est défini uniquement pour les PID : IN, SETUP et OUT. Les périphériques lents supportent au maximum 2 adresses de points d'arrêt (terminaisons) par fonction. Les périphériques « Full Speed » acceptent jusqu'à 16 points d'arrêt (codage sur 4 bits).*
- *CRC : Vérification de la redondance des cycles. Ce champ est utilisé pour vérifier l'intégrité des données des paquets. Le PID n'est pas inclus dans la vérification. Tous les CRC sont générés avant l'insertion des bits de bourrage et sont décodés par les récepteurs après avoir enlevé ces bits de bourrage. Un CRC défectueux indique que un ou plusieurs des champs protégés est faux, le récepteur ignore ainsi ce(s) champ(s), voire le paquet entier.*
  - *Pour un PID jeton, le CRC est de 5 bits, il est calculé à partir des champs ADDR et ENDP. Le polynôme générateur est : $G(X) = X^5 + X^2 + 1$. Si tous les bits sont reçus sans erreur, le CRC au niveau du récepteur est : 01100.*
  - *Pour un PID FRAME, le CRC est de 5 bits, calculé à partir des 11 bits du champ FRAME.*
  - *Pour un PID DATA, le CRC est composé de 16 bits, calculé à partir du champ données. Le polynôme générateur est : $G(X) = X^{16} + X^{15} + X^2 + 1$. Si tous les bits de données et de CRC sont reçus sans erreur, le CRC est 1000000000001101.*

V.3.6 Détection d'erreurs

Le bus USB étant bien plus évolué que les bus précédemment cités, la détection d´erreur ne se fait pas par un système de contrôle de parité mais utilise le CRC (Code de Redondance Cyclique, qui permet de corriger parfaitement 100% de mots contenant 1 ou 2 erreurs). De plus, le contrôleur peut réinitialiser jusqu´à 3 fois de manière hardware la liaison avec un autre élément avant d´en avertir le logiciel client.
Il y a 3 catégories d'erreur générale pour les paquets USB :

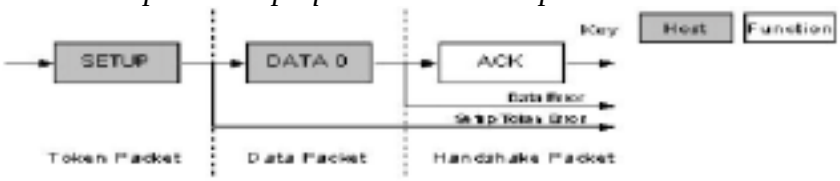
Violations du bit de bourrage / Bits de vérification du PID / CRC

Champ	*Type d'erreur*	*Action requise*
PID	*Vérification du PID, Bit de bourrage*	*Paquet ignoré*
ADDR	*Bit de Bourrage, CRC d'adresse*	*Jeton (Token) ignoré*
FRAME NUMBER	*Bit de Bourrage,CRC de frame*	*Champ du n° de tableau ignoré*
DATA	*Bit de Bourrage, CRC d'adresse*	*Données ignorées*

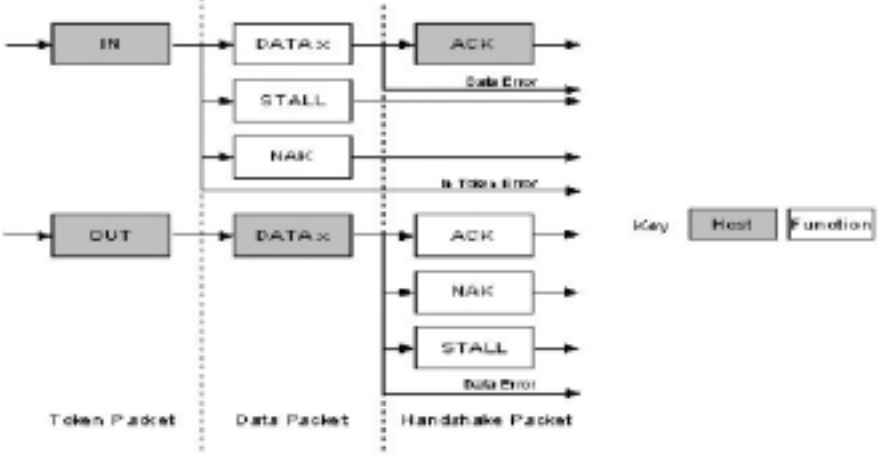
V.4 - Types de transferts (terminaisons)

V.4.1 Control Transfers

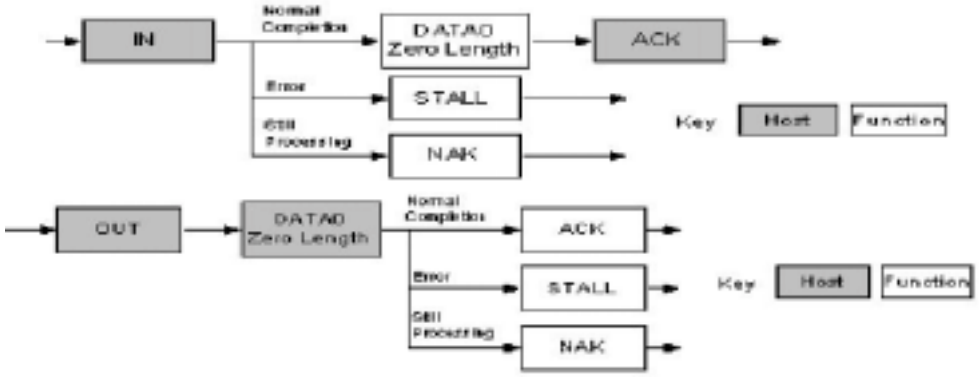
- *étape d'installation : composé de 3 paquets : Token Setup-Data 0-Handshake Ack.*



- *étape de données (facultative) :*



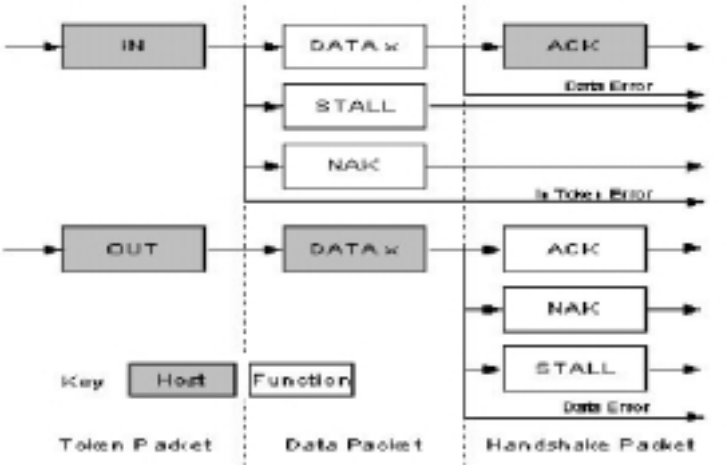
- *étape d'état*



- *Message Stream*
- *Configuration, status, set up des périphériques*
- *Contrainte sur la taille des paquets : 8, 16, 32, 64 octets ("Full Speed") et 8 octets ("Low Speed")*
- *Protocole overhead : 45 octets*

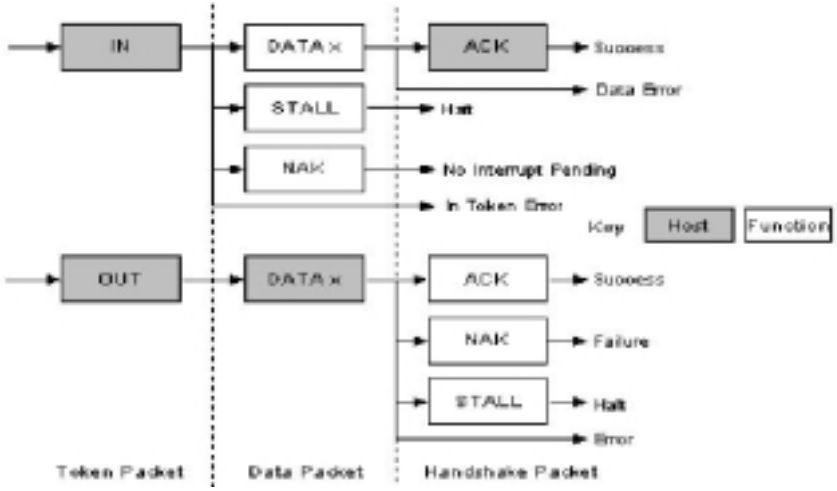
Le caractère Plug & Play est assuré par une phase de reconnaissance et de configuration automatique des périphériques connectés au bus. En fait, le host détecte tout périphérique nouvellement branché, l'interroge à l'aide de transferts de contrôle pour connaître ses caractéristiques et choisit en conséquence le driver adéquat pour le contrôler. De même, toute déconnexion est automatiquement détectée (en hard) par le hub concerné qui la signale alors au host afin que celui-ci en tienne compte.

V.4.2 Bulk Data Transfers :



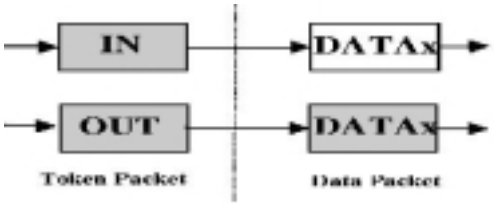
- *Pipe Stream*
- *Paquets en rafale de données (ex : Scanners, imprimantes (sans exigence de temps réel)*
- *Taille des paquets : 8, 16, 32, 64 octets (uniquement "Full Speed")*
- *Utilise la bande passante quand elle est libre*
- *Protocole overhead : 13 octets, data 1023 octets max*

V.4.3 Interrupts Transfers



- *Pipe Stream*
- *Message court (petite quantité de données) émis de temps en temps (pas d'interruptions)*
- *Taille des paquets : < 64 octets ("Full Speed") et < 8 octets ("Low Speed")*
- *Transmission en "Best Effort"*
- *Protocole overhead : 13 octets*

V.4.4 Isochronous Transfers

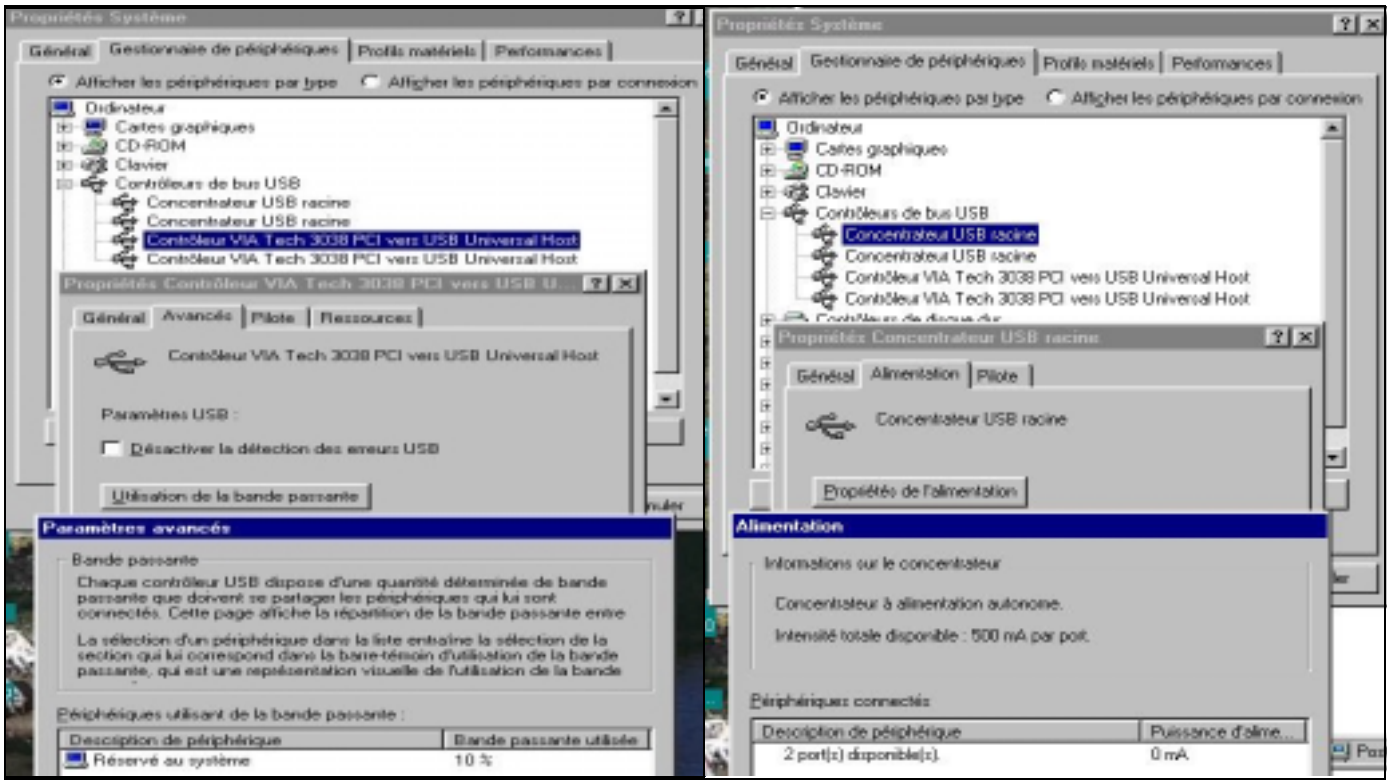


- *Pipe Stream*
- *Transmission isochrone (intervalles de temps réguliers)*
- *Bande passante garantie*
- *Contrainte sur la taille des paquets : < 1023 octets/frame*
- *Applications temps réel (exemple : Phonie, voix, video, ISDN ...)*
- *Protocole overhead : 9 octets*

Les échanges isochrones sont les plus privilégiées dans le sens où le host leur réserve une bande passante garantie. Ainsi, celui-ci peut refuser l'accès au bus à un périphérique s'il juge que les ressources qu'il requiert ne sont pas disponibles.

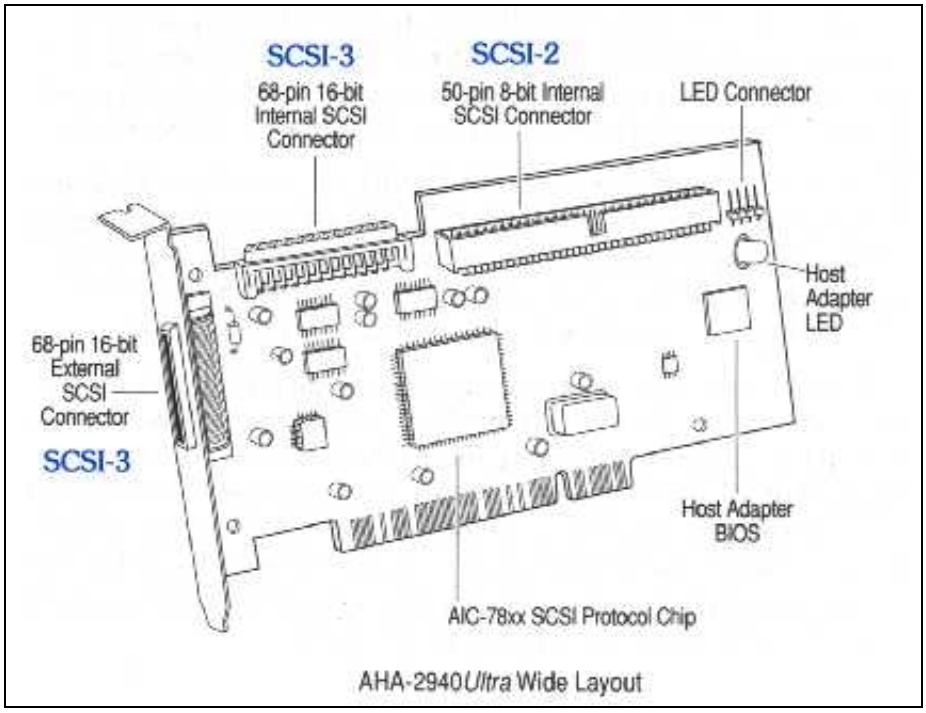
	Control	***Isochronous***	***Interrupt***	***Bulk***
Mode (format data)	*Message (prédéfini)*	*Stream (pas prédéfini)*	*Stream (pas prédéfini)*	*Stream (pas prédéfini)*
Direction	*Bidirectionnel*	*Unidirectionnel*	*Unidirectionnel*	*Unidirectionnel*
Taille Max Données	*8,16,32,64 o (Full) 8 octets (Low)*	*1023 o (par trame) Full Speed*	*64 o (Full) 8 o (Low)*	*8,16,32,64 o (Full)*
ACK-Reprise/Erreur	*oui*	*non*	*oui*	*oui*
Bande réservée	*10% de la trame "Best Effort"*	*90% de la trame "guaranteed"*		*Non "Good Effort"*

V.5 - Sur votre PC…



VI. SCSI

VI.1 - SCSI : généralité & norme



*Le SCSI (1986), ou **S**mall **C**omputer **S**ystem **I**nterface se distingue des autres ports d´entrées/sorties externes, d´une part car il permet la connexion de périphériques internes, et d´autre part il nécessite un contrôleur externe non-intégré à un chipset. Une carte additionnelle SCSI ISA ou PCI est donc nécessaire pour pouvoir connecter des périphériques de ce type (il existe maintenant des contrôleurs SCSI intégrés sur les cartes mère).*

Les normes SCSI définissent les paramètres électriques des interfaces d'E/S. Le SCSI-1 (1986) évolua rapidement vers le SCSI-2 (1994) pour permettre la connexion d´une multitude de périphériques rapides selon une architecture en chaîne. Elle définit 18 commandes CCS (Common Command Set)

Le SCSI-3 permet une fréquence de bus (Ultra, Ultra2, Ultra3) bien plus importante que le SCSI-2 et se démarque par l´adoption d´une interface série (Fibre Channel, FireWire, SSA) reconnue par de nombreux périphériques : Hard Disks, lecteurs/graveurs CD/DVD, scanners, caméscopes, ...

Bien que l´équipement nécessaire à mettre en place une solution SCSI dans son micro-ordinateur soit onéreux, ce qui le rend surtout accessible aux professionnels, la norme SCSI dispose de réels avantages par rapport aux autres normes : suivant la largeur du bus, un nombre différents de périphériques peut se raccorder au contrôleur, 7 dans le cas d´une largeur de 8 bits et 14 dans le cas d´une largeur de 16 bits. La sérialisation du SCSI permet d´augmenter ce nombre à plus de 100.

VI.2 - Principe de fonctionnement sommaire

L'adressage des périphériques se fait grâce à des numéros d'identification. La premier numéro est l'ID qui permet de désigner le contrôleur intégré au périphérique. En effet, le contrôleur peut avoir jusqu'à 8 unités Logiques qui sont repérées par un identificateur appelé LUN (Logical Unit Number). Un ordinateur peut comporter plusieurs cartes SCSI, repérée chacune par un numéro de carte. Pour établir une communication, l'ordinateur doit lui donner une adresse de la forme : <n° de carte><ID><LUN>

2 types de bus existent :

- *bus asymétrique : basé sur une architecture parallèle dans laquelle chaque canal circule sur un fil, ce qui le rend sensible aux interférences.*
- *bus différentiel : Transport des signaux sur une paire de fils, les perturbations sont compensées (l'information est codée par différence entre les 2 fils)*

L´usage d´un contrôleur, permet une économie de ressources processeur central, la gestion des accès et des transferts étant réalisée par des processeurs spécialisés.

VI.3 - SCSI parallèle

Il s'agit des interfaces Ultra 3, 160, 320 et 640. Les commandes sont placées en file d'attente, les périphériques acceptent ainsi plusieurs commandes et le bus les traite dans l'ordre le plus judicieux (meilleure utilisation de la bande passante, intéressant pour les systèmes multitâches).

Par souci de compatibilité entre les périphériques, on a modifié la gestion du transfert de données de la manière suivante (Ultra 160):

- *Mise en œuvre du CRC : Pour éviter le phénomène de diaphonie (perte de données lors d'échanges rapides), le protocole gère maintenant Le CRC qui permet de vérifier l'intégrité des données.*
- *Domaine validation : La carte contrôleur déterminait le taux de transfert à utiliser avec un périphérique (INQUIRY), ce process consommait beaucoup (5 Mo/s !) et pouvait rendre ce périphérique inacessible en cas d'erreur de communication. Dorénavant, la carte négocie le meilleur taux de transfert.*
- *Double transition clocking : Utilisation des fronts montant et descendant du signal pour transférer les données. Les commandes ne sont transmises que sur front montant (transfert de 2 fois plus de données sur un même signal).*

Puis il y a eu intégration des extensions suivantes (Ultra 320) :

- *Packetized SCSI : gestion des paquets de données. Réduction du nombre de commandes transférées en même temps que les données.*
- *Quick Arbitration & Selection (QAS): Réduction du nombre de commandes.*
- *R/W Data Streaming : Nouvelles gestion du flux de données. Envoi de plusieurs paquets sans intercalage de commandes (accroissement du débit).*
- *Flow Control : Pré-traitement des données avec gestion d'une mémoire cache. envoi des données en mode rafale sur le bus*
- *Pre-compensation : L'augmentation de la fréquence du Bus provoque une diminution de l'amplitude et une augmentation du phénomène de réflexion du signal (bruit HF), phénomènes que l'on régule par la pré-compensation.*

VI.4 - Interfaces SCSI Série

*Il s'agit des interfaces SSA (**S**erial **S**torage **A**rchitecture), FC (**F**iber **C**hannel) et IEEE1394. La transmission parallèle entraîne des restrictions en termes de longueur de câble (plus le taux de transfert est grand, plus le câble doit être court à cause des parasites dus aux courants induits et des interférences).*

Ces interfaces permettent aux signaux de données de transiter par une seule voie plutôt que d'être transmis en parallèle via des conducteurs multiples. Les signaux de commandes, d'états et de données sont encapsulés en paquets pour la transmission. Elles possèdent plusieurs avantages :

- *interconnexions peer-to-peer ("point à point") : augmentation de la fiabilité et réduction de la complexité du câblage.*
- *double accès : transmission des données par 2 voies indépendantes*
- *connectivité accrue et terminaisons simplifiées.*
- *possibilités de topologies en Bus, Anneaux et Arbres*
- *connexions réseau et liaisons WAN.*

VII. IEEE1394

VII.1 - Introduction

Le FireWire (IEEE 1394-1995) est une partie intégrante de la norme SCSI-3. Il utilise un bus série et facilite la connexion de périphériques externes. On s'oriente de plus en plus vers une gestion bus réseaux

☞ ***FireWire ? IEEE 1394 ? i-Link ?***

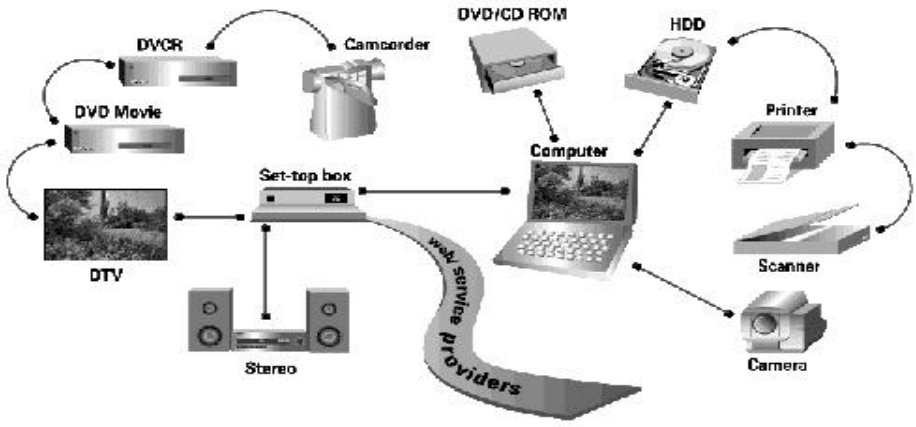
- *IEEE 1394-1995 Standard for a High Performance Serial Bus est le nom de la norme originale (création 1995).*
- *FireWire est l'implantation Mac (Apple Computer, Inc.) de l'IEEE 1394.*
- *P1394.x désigne un câble à 4 conducteurs (6 conducteurs composent le câble normalement, les 2 conducteurs de l´alimentation sont supprimés)*
- *DigitalLink, ou i-Link (Sony Corporation) : produits Digital Handycam*

L'IEEE 1394 possède les avantages suivants :

- *Interface numérique : pas de CAN dégradante pour les données.*
- *Taille réduite : câble fin et peu coûteux.*
- *Utilisation très facile, Hot Plug and Play.*
- *Taux de transfert importants, bus supportant des périphériques mixtes ayant une vitesse de transmission de 100, 200 ou 400 Mbits/s.*
- *Topologie flexible : support du Daisy Chaining (« marguerite »).*
- *Garantie de bande passante pour tous les périphériques raccordés.*

Grâce à ses performances, l´IEEE-1394 est destinée particulièrement à relier :

- *Les ordinateurs.*
- *Les produits manipulant de l´audio, des images et de la vidéo.*
- *Les imprimantes et les scanners.*
- *Les disques durs.*
- *Les caméscopes.*



D´une topologie très proche de celle du l´USB, le FireWire se démarque par sa bande passante bien supérieure. Il ne nécessite pas de HUBs FireWire (en plus de l´utilisation d´une transmission isochrone, le FireWire supporte le transport de deux flux vidéo temps réel avec une qualité broadcast en simultanée) et possède également des ports pour une utilisation interne.

VII.2 - Architecture

VII.2.1 Chipset VIA Fire xx

VIA propose des chipsets intégrant le FireWire. Prévu pour concurrencer le FireWire, l´USB2 propose des caractéristiques attrayantes mais est destiné à un marché un peu différent. Dans sa nouvelle

déclinaison, l´IEEE 1394b 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 puis à 1.6 Gbits/s et 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 ! (câbles UTP-5 en 100 Mbits/s).*
- *Nouveau protocole visant à améliorer la disponibilité de la bande passante grâce à une implémentation BIOS du protocole d´adressage.*
- *Compatibilité avec les périphériques **IEEE 1394-1995** par un mode bilingue.*
- *Nouvelles applications (automobile, acquisition audio-vidéo).*

VII.2.2 Un bus standard ?

En 1995, L'IEEE adopte finalement le Serial Bus comme standard (IEEE1394-1995, Standard for a High Performance Serial Bus, interface numérique de référence sur les équipements audio-vidéo grand public). De nombreux produits IEEE 1394 sont disponibles sur le marché (caméras, câbles & connecteurs Molex, boîtiers TI, cartes d'interface Adaptec, imprimantes, scanners). La version appelée 1394b "bilingual" (bilingue) garantit la compatibilité avec « l´ancienne » norme IEEE 1394-1995.

VII.2.3 Câbles

Deux paires de câbles (TPA et TPB) servent pour le transport des informations (données + horloge) et une paire (VP-VG) est utilisée pour la téléalimentation (1,5 A sous 8 à 40V). Cette alimentation permet d'alimenter les cartes réseau afin que l'arrêt d'un équipement ne provoque pas de coupure sur le réseau. Le support (moins rapide que la fibre optique) offre des gammes de vitesse amplement suffisantes pour les applications du multimedia : 100 Mbit/s (12.5 Mo/s) en half-duplex, 200 Mbit/s (25 Mo/s) et 400 Mbit/s (50 Mo/s) en full-duplex (×2, ×4 : possibilité dûe aux câbles) et 800 Mbit/s (100 Mo/s).
Pour IEEE 1394-b les vitesses sont de 1.6 (200 Mo/s) et 3.2 Gbits/s (400 Mo/s).

VII.2.4 Norme

La norme définit deux environnements distincts :

- *châssis (**backplane**) : il permet d'interconnecter les cartes d'un micro-ordinateur, et vient en quelque sorte suppléer les bus de fond de panier.*
- *câble (**cable**). il est défini pour l'interconnexion d'équipements externes. Ses caractéristiques principales sont :*
  - *63 équipements au plus sur un réseau, reliés point à point,*
  - *27 ports au plus sur chaque équipement*
  - *topologie acyclique (pas d'anneau) daisy chain en arbre (cf. figure 1) ,*
  - *distances maximales autorisées : 4,5 m entre deux noeuds voisins, 72 m entre deux noeuds quelconques (c'est-à-dire 16 « sauts »),*
  - *adressage sur 64 bits (norme IEEE 1212, systèmes multibus). Spécification des registres mémoires de chaque nœud et des commandes (transactions read, write et lock) qui permettent d'y accéder.*

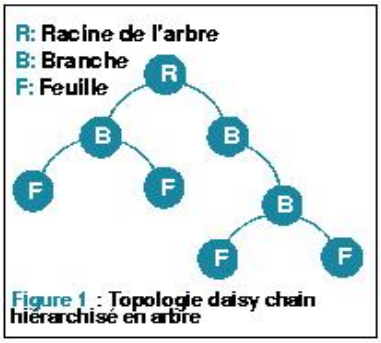


Figure 1 : Topologie daisy chain hiérarchisé en arbre

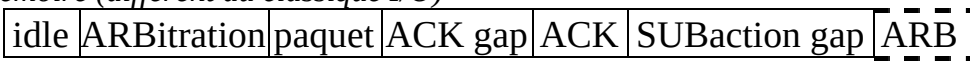
Contrairement au SCSI « classique » qui possède un cablâge supervisé (daisy chain, adresse en manuel ou fixée, terminaison obligatoirement en bout de chaîne), avec IEEE 1394, on se retrouve en non-supervisé (daisy chain hiérarchisé, réseau acyclic, sélection des adresse automatiques, pas de terminaison, situation physique des périphériques quelconque au plug).

VII.3 - Fonctionnement du bus FireWire?

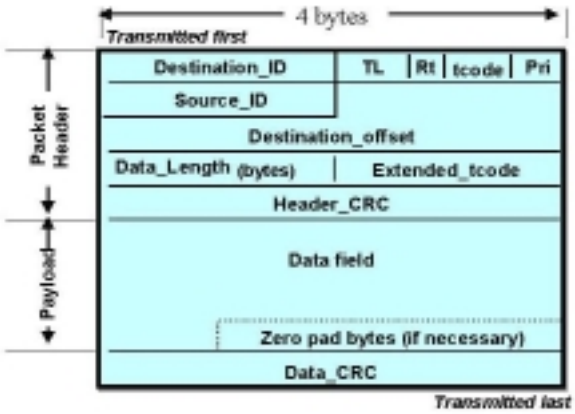
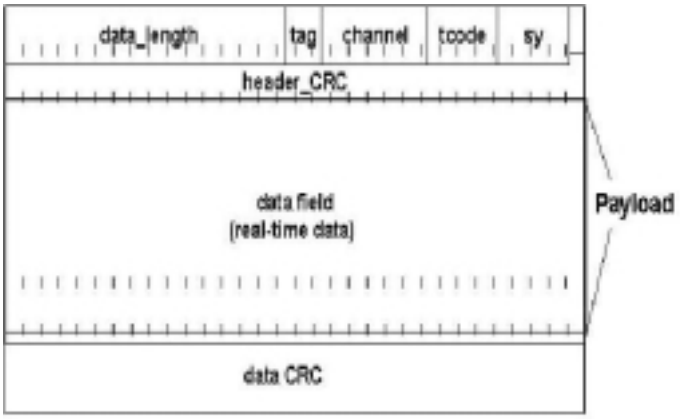
Le réseau est flexible et simple d'utilisation. Il est facile d'ajouter ou de retirer à tout moment (Hot Plug & Play) un équipement à n'importe quel endroit du réseau : le bus se (re)configure en lançant automatiquement des phases d'initialisation.

Caractéristiques principales :

- *architecture R/W en mémoire (différent du classique I/O)*
- *trame standard :*



- *émission / réception par **paquets** de données de 280 octets maximum (Header = 20 octets +Payload = 256 octets + CRC = 4 octets).*

paquet asynchrone	paquet isochrone
	

- *cycle nominal = 125 µs (8000 cycles/s)*
- *détection automatique de la vitesse des échanges.*
- *envoi des données avec une vitesse compatible au récepteur.*
- *nœuds : réception sur un port, émission sur plusieurs ports possibles*
- *pendant la transmission chaque nœud peut accèder au bus → half-duplex*
- *2 modes :*
  - ***différentiel** : pour reset, arbitrage, configuration, transmission paquets.*
  - ***commun** : pour détection d'attachement/détachement de périphérique, commande du signal de vitesse, et du signal de suspension / reprise de transmission sur le bus.*
- *séquence **émission :** ARBitrage / Prefixe-Data-End / Liberation Bus*
- *séquence **réception :** Data reçue (PHY) puis envoyée à couche liaison*

VII.3.1 Un protocole, trois couches

*La norme IEEE 1394 s'appuie sur le modèle OSI et spécifie trois couches : les deux couches basses **physique** et **liaison** qui correspondent aux couches 1 et 2 du modèle OSI et la couche transaction qui fait office d'interface entre la couche application et la couche liaison pour le mode asynchrone :*

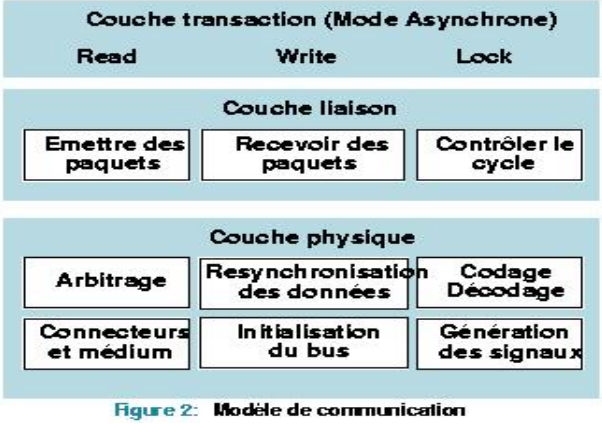
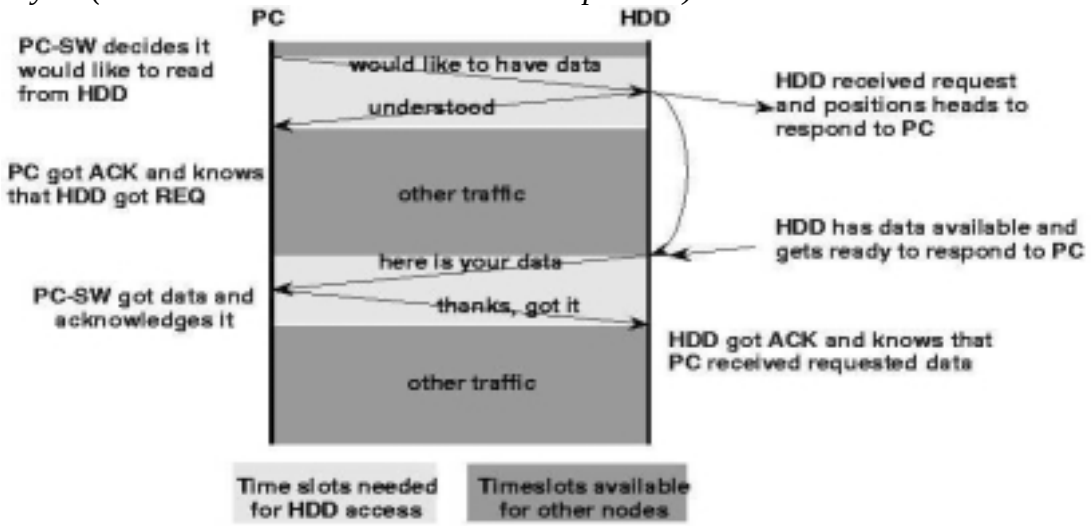


Figure 2: Modèle de communication

La couche liaison spécifie deux modes de transfert (car 2 fils d'horloge) :

- *Mode **asynchrone** : transmision de paquets à intervalles de temps variables avec garantie de la bonne réception des données par accusé de réception. Ce temps de latence ne peut pas être quantifié car il dépend de taux d´utilisation du bus par d´autres transmissions pour d´autres appareils communicants entre eux.*
- *Mode **isochrone** : envoi de paquets (données) de taille fixe à intervalle de temps régulier (cadencé par les 2 fils d'horloge) sans accusé de réception (débit fixe, bande passante garantie et adressage simplifié). Les communications isochrones sont prioritaires aux asynchrones de sorte que la bande passante est assurée. La communication isochrone entre deux appareils ou plus est assimilable à un canal. Une fois un canal établit, l´appareil demandeur est garanti d´avoir l'espace-temps demandé à chaque cycle (données vidéo ou autres besoins "temps réel").*



La co-existence de ces modes est assurée par une méthode déterministe d'accès au support. La racine du réseau est particularisée au cours des phases d'initialisation et distribue le droit de parole au cours de la phase d'arbitrage qui précède systématiquement tout échange de données en cadençant les échanges isochrones (rôle de cycle master).

La gestion de réseau est distribuée dans les noeuds qui se répartissent en six profils, selon leurs performances : Les « noeuds » sont

- *« repeater » : simples couches physique actives*
- *« transaction capable » : transfert de données en asynchrone*
- *« isochronous capable » : supportant le transfert isochrone*
- *« cycle master capable » : chef d'orchestre pour l'arbitrage*
- *« isochronous ressource manager capable » : capables de gérer les numéros de canaux isochrones et leur taille*
- *« bus manager capable » gestion de la téléalimentation, optimisation de la vitesse et de la topologie*

VII.3.2 Une couche physique optimisée

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. Les signaux sont mesurés en mode différentiel et les niveaux de tension sont de faibles amplitudes (quelques centaines de millivolts). Une particularité du bus IEEE 1394 réside dans l'utilisation de deux paires pour transporter les données. En effet, en plus d'un signal data classique (RxD en réception ou TxD en émission), les noeuds émettent aussi un signal appelé strobe (RxStrb ou TxStrb) qui change d'état lorsque deux bits égaux consécutifs sont émis sur la paire data. Une transition a donc systématiquement lieu à chaque top d'horloge, soit sur la paire data, soit sur la paire strobe. Cette technique optimise ainsi la bande passante et explique en partie la valeur élevée des vitesses atteintes.

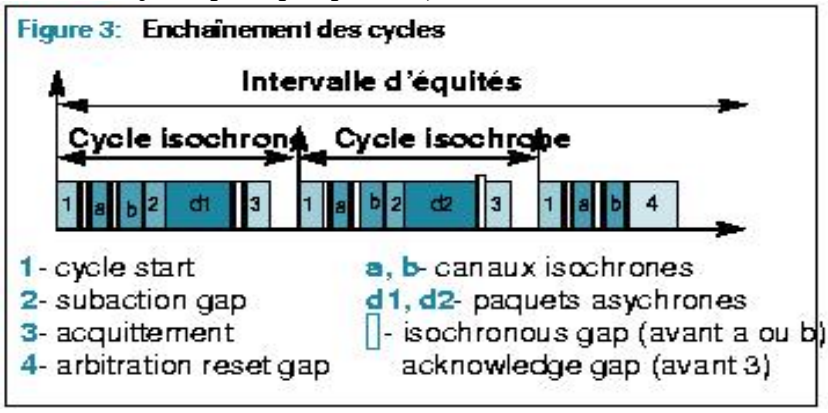
VII.3.3 Un arbitrage efficace et juste

La phase d'arbitrage permet de gérer l'accès au support. Chaque noeud qui souhaite envoyer des données demande obligatoirement l'autorisation d'émettre à la racine. La topologie crée donc une priorité naturelle entre les noeuds : + un noeud est loin de la racine, - il est prioritaire (arbitrage des câbles).

Pour pallier cette inégalité le protocole introduit 2 notions :

- ***fairness interval** (mode asynchrone) : les noeuds ne peuvent émettre qu'un seul paquet asynchrone pendant un intervalle d'équité. Celui-ci se termine quand tous les noeuds asynchrones qui ont demandé la parole l'ont prise une fois. Chaque nœud a donc une possibilité d'accès pendant cet intervalle.*
- ***cycle master** : (mode isochrone) : la racine déclenche le début de cycle isochrone à l'aide d'un paquet asynchrone particulier (le cycle start) qui est le plus prioritaire puisque généré par la racine. Chaque noeud isochrone qui prend alors la parole a pris soin au préalable de se réserver une bande passante auprès du noeud isochronous ressource manager. Le cycle isochrone prend fin lorsque tous les noeuds isochrones qui ont demandé la parole l'ont prise une fois.*

*Les moyens mis en oeuvre pour implémenter ces principes reposent sur les notions de signaux de signalisation et d'intervalles d'attente . A **chaque type de données** (asynchrone, acquittement, isochrone) est associée une **classe d'arbitrage** (fair, immediate, isochronous) qui est caractérisée par un **intervalle de temps d'attente** (subaction gap, acknowledge gap, isochronous gap) pendant lequel le bus doit être libre. Avant d'envoyer sa demande d'autorisation de prise de parole à la racine, un noeud doit détecter un de ces intervalles. Le subaction gap est plus large que l'acknowledge gap ou l'isochronous gap. Ceci garantit que le transfert asynchrone ne vient jamais perturber le transfert isochrone et qu'une nouvelle requête ne précède pas l'émission d'un acquittement attendu. La demande d'autorisation et sa réponse sont véhiculées point à point à l'aide de signaux de signalisation. Pendant cette phase, les noeuds tentent (au temps t) de forcer l'état des lignes dans un état logique précis. Un noeud qui souhaite par exemple prendre la parole force ses lignes TPA et TPB dans les états Z et 0. Ensuite, au temps t + dt (dt vaut une dizaine de nanosecondes), les noeuds observent l'état résultant des lignes et en déduisent le leur (par exemple : «je peux parler» ou «je ne peux pas parler»).*



VII.3.4 Plug & Play et le Hot Plugging ?

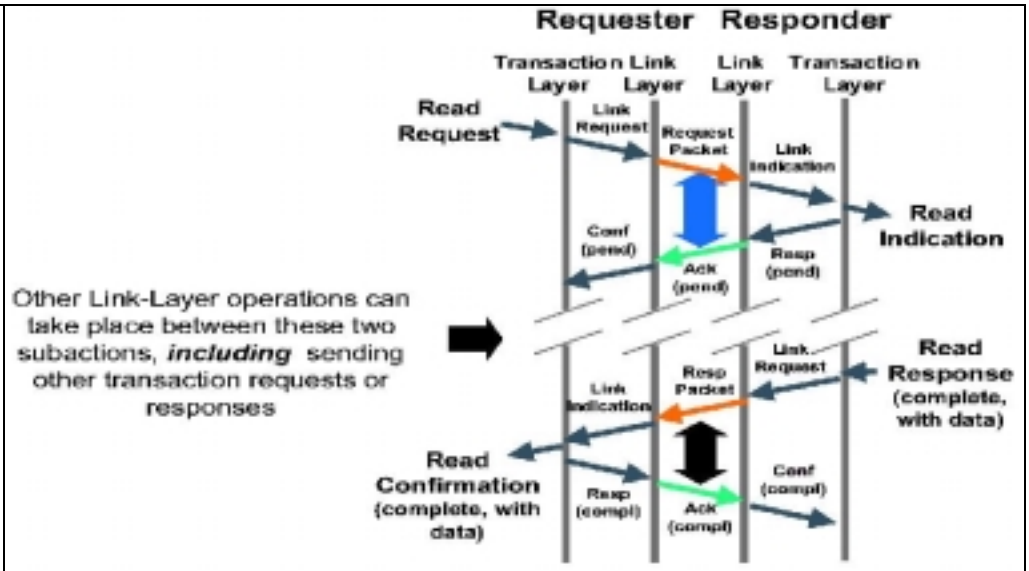
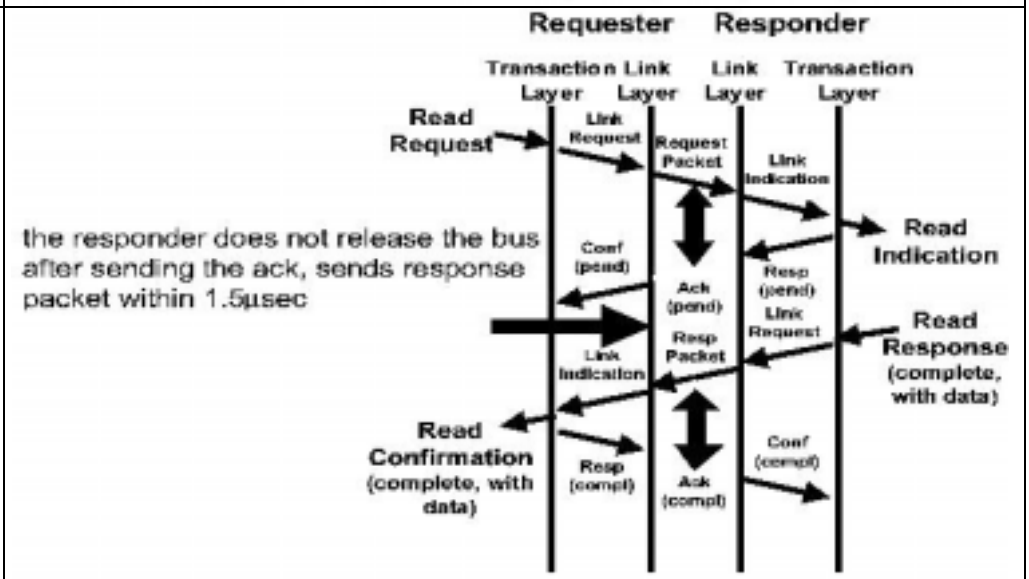
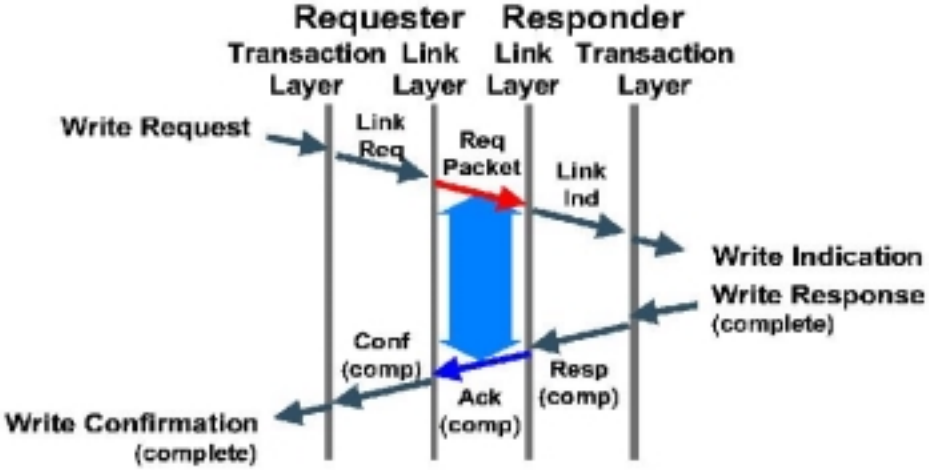
*Au démarrage ou à la suite d'un changement de configuration, le réseau se (re)configure de façon automatique en lançant trois phases successives : la phase de **bus initialize**, la phase de **tree identity**, et la phase de **self identity**. Lors de la première étape, le premier noeud qui détecte un changement génère le signal de signalisation reset qui force tous les noeuds du réseau à oublier la topologie courante et à identifier leur nature (feuille ou branche) . Lors de la seconde étape, chaque noeud se numérote et se positionne dans la hiérarchie arborescente (son unique port parent pointe vers la racine, ses autres ports sont des ports enfants) à l'aide de signaux de signalisation. A l'issue de cette étape, le noeud racine se reconnaît car il est le seul (par construction) à ne posséder que des ports enfants. Au cours de la self identity, les noeuds choisissent leur adresse réseau, diffusent les débits qu'ils supportent, proposent éventuellement leur candidature à tel ou tel rôle de manager. A l'issue de cette phase, l'isochronous ressource manager (s'il existe) a été identifié et le cycle master (la racine) a été activé. La communication peut alors débuter.*

Bus INITialize	**T**ree **ID**entification	**S**elf **ID**entification	**N**ormal **ARB**itration
Live Insertion New Cycle Master	Topologie réseau en arbre	n° nœud→physique voisins→même vitesse	Root prioritaire

VII.3.5 IEEE 1394 b, des améliorations…

- *Vitesses : 1.6-3.2 Gbit/s, grandes distances : 50-70-100 mètres (UTP5) ou 800 m (Fibre Optique)*
- *Low Power : possibilité faible puissance (1.5 A = maximum) → économies d'énergie.*
- *Arbitrage amélioré :*
  - *suppression du « subaction gap » une fois un cycle démarré (transmission + rapide)*
  - *arbitration « Fly-By » pour éviter colision DATA/ACK suite à la modification précédente.*
  - *Priority arbitration : pipelining → le premier demandeur est le maître.*

VII.4 - Annexes

Split Transaction	
Concatenation	
Unification	

VIII. *ACQUISITION DE DONNEES*

VIII.1 - Introduction

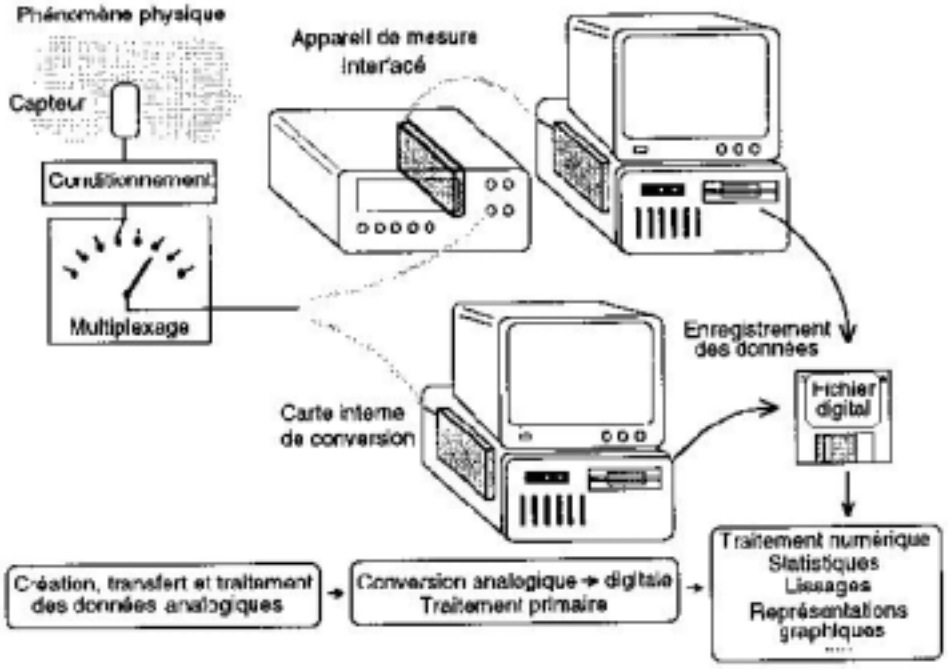
Dans le domaine des sciences pour l'ingénieur, le système de mesure le plus simple et le plus répandu consiste à utiliser un micro ordinateur interfacé avec le système sur lequel on travaille. Etablir un dialogue entre un micro ordinateur et un système extérieur au PC est possible par l'intermédiaire de cartes d'interface. Ces ***cartes dites d'Entrées/Sorties*** *sont directement implantées à l'intérieur du PC et connectées sur le bus PC via des Slots. Elles sont généralement équipées de compteurs (timers), de circuits convertisseurs offrant des entrées et des sorties analogiques et des circuits à base de portes logiques offrant des entrées et des sorties logiques à l'utilisateur. Cet ensemble PC et carte d'interface est complété par un logiciel (généralement appelé driver) permettant la gestion de l'ensemble des fonctions de la carte Entrées/Sorties par l'intermédiaire d'un programme.*

VIII.2 - structure générale d'un système de mesure

VIII.2.1 Les Capteurs

Les capteurs sont un des éléments de fin de chaîne d'acquisition de données. Ils sont sensibles aux phénomènes physiques ou chimiques. Leur rôle consiste à produire un signal électrique, le plus souvent une tension analogique, mais aussi parfois une intensité, une fréquence ou une série de pulsations. Ces signaux sont liés au milieu dans lequel ils sont placés ou au phénomène qu'ils doivent détecter. Leurs caractéristiques importantes sont le domaine d'utilisation, la sélectivité, la sensibilité, la dérive et la reproductibilité, l'encombrement, la fiabilité, le coût, etc...

<u>Problèmes d'utilisation</u> : perturbation du phénomène mesuré, niveau de signal, problèmes dus aux parasites lors de la transmission de leurs signaux, non-linéarité de la réponse...



Exemple d'un système d'acquisition

VIII.2.2 Le conditionnement du signal

Les signaux issus du capteur ne sont pas toujours directement utilisables par le dispositif de CNA. Par exemple, les cartes de conversion les plus répandues ont un calibre d'entrée de 0 - 10 volts, avec une résistance de quelques centaines de kΩ.. Souvent, il faudra :

- *Amplifier les signaux*
- *Adapter leur impédance*
- *Décaler leur origine de manière à exploiter au mieux l'amplitude acceptée par l'organe de mesure électrique.*
- *Transformer des courants ou des fréquences en tensions*
- *Filtrer pour éliminer des interférences, peut-être aussi linéariser de manière à obtenir un signal proportionnel à la grandeur étudiée.*

Lorsque la distance entre le capteur et le système convertisseur analogique-numérique est grande, on a intérêt à transmettre un signal amplifié, à relativement basse impédance, et éventuellement symétrique, pour diminuer l'importance relative des tensions parasites ajoutées le long des câbles de liaison. C'est un autre rôle du conditionnement du signal.

Nous engloberons aussi sous ce terme les techniques nécessaires pour isoler électriquement et amplifier les tensions disponibles cette fois en sortie, de manière à les rendre aptes à commander les actionneurs des processus externes comme le déclenchement d'un chauffage, le lancement d'un moteur ou l'ouverture d'une électrovanne.

VIII.2.3 Le multiplexage

Le multiplexage est une méthode permettant de scruter successivement plusieurs voies d'entrée dans un système d'acquisition. Bien entendu, cela allonge d'autant l'intervalle de temps qui s'écoule entre deux mesures sur un même canal.

Le multiplexage peut consister simplement en une commutation sur l'une ou l'autre voie par l'intermédiaire de relais électromagnétiques pilotés par le PC lui-même, mais beaucoup de dispositifs de mesure comportent en entrée un système de multiplexage réalisé entièrement en semi-conducteurs, et bien entendu programmable.

Lorsque l'on désire une acquisition plus rapide, ou une mesure quasi-simultanée sur plusieurs voies, on préfère consacrer un convertisseur analogique-numérique à chaque voie.

VIII.2.4 La Conversion Numérique Analogique (CNA)

La conversion analogique-numérique transforme les valeurs de signaux électriques continûment variables (signaux analogiques) en nombres exploitables par l'ordinateur de manière numérique.

Cette conversion est parfois effectuée dans des appareils externes (l'exemple le plus classique étant le multimètre). Ces appareils externes envoient à l'ordinateur le résultat numérique obtenu par divers systèmes de communication, les plus répandus étant la liaison RS 232 ou le système de bus IEEE 488. On peut aussi dans certains cas utiliser la liaison parallèle prévue pour une imprimante.

Les liaisons RS 232 et les connexions d'imprimante sont très généralement prévues d'origine dans les PC, mais par contre les liaisons IEEE doivent être effectuées à travers d'une carte d'interface ajoutée. D'autres systèmes seraient possibles, par exemple au moyen de cartes d'interfaces SCSI, Ethernet ou USB.

De plus en plus, on incorpore le dispositif de conversion à l'ordinateur lui-même, sous forme d'une carte d'extension qui comporte un ensemble d'entrées-sorties numériques, de compteurs, etc.. On évite cependant d'incorporer dans l'ordinateur des cartes de mesure lorsqu'on cherche une très grande immunité vis-à-vis du bruit de fond, car le PC est un puissant générateur de parasites...

VIII.2.5 Le Logiciel d'Acquisition et de Traitement

Le logiciel d'acquisition gère le multiplexage éventuel, le protocole de communication avec l'appareil de mesure, les changements de calibre s'il y a lieu, ou les diverses étapes de la conversion si l'on a recours à une carte incorporée.

Les données récupérées sous forme de chaîne de caractères ou d'octets sont ensuite mises en forme et transformées en valeurs numériques, éventuellement traitées de manière élémentaire. Le logiciel peut contrôler un dispositif expérimental, au moyen de sorties numériques ou même analogiques.

VIII.2.6 Timer (Compteurs-Décompteurs)

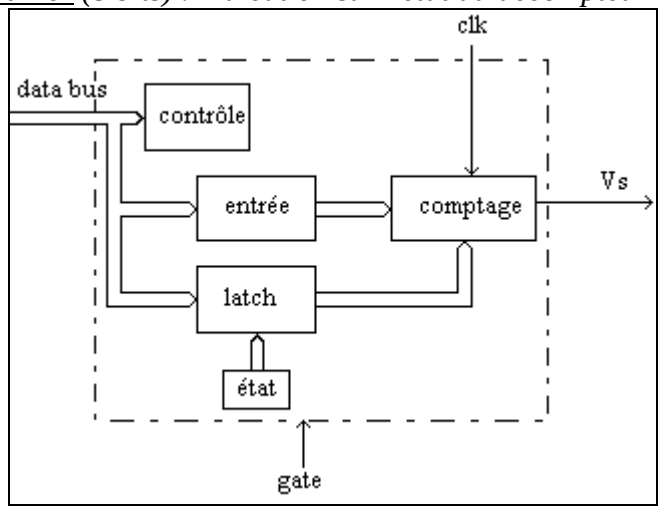
Objectif : prise en compte d'une manière précise du facteur temps.

Les timers possèdent en général 4 registres :

1. *le registre d'entrée (précompte) : chargement de la donnée en 2 fois (LSB + MSB) → écriture*
2. *le registre de décomptage : chargement à partir du registre d'entrée et décompte à chaque front descendant de l'horloge du quartz et autorisation du décompte par la gâchette "gate".*

   durée de décomptage max avec un compteur 16 bits et un quartz de 4 MHz : $2^{16}.\frac{1}{F_{osc}} = 16.4 \text{ ms}$

   2 timers en cascade → durée de décompte plus importante
3. *le registre de contrôle (8 bits) : définit le mode de fonctionnement*
4. *le registre d'état du timer (8 bits) : indication sur l'état du décompteur → opération de lecture*



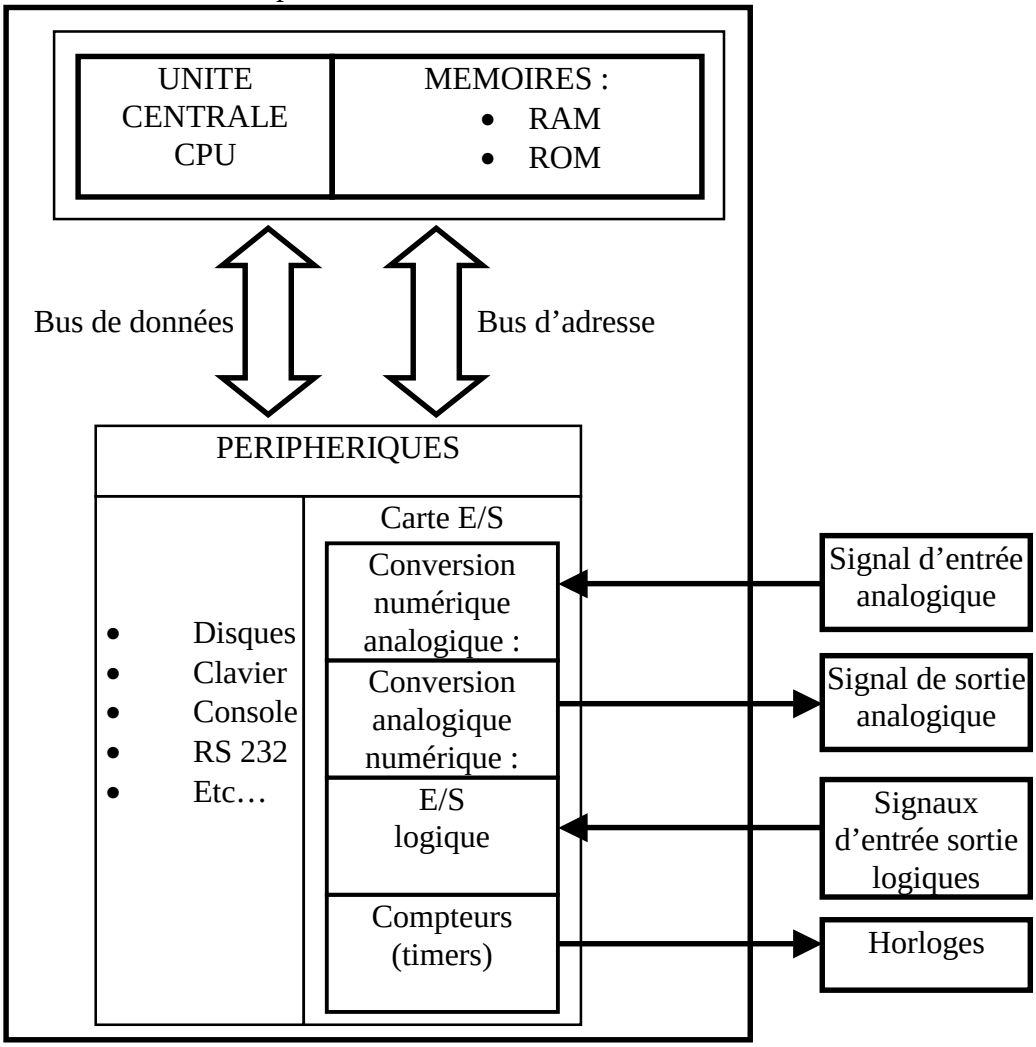
VIII.3 - Généralités sur les cartes E/S

- *Une carte E/S est considérée comme un périphérique externe*
- *Une carte E/S permet un dialogue entre le PC et l'extérieur*
  - *réalise l'acquisition de signaux analogiques*
  - *élabore des signaux analogiques à partir du PC*
  - *échange bidirectionnel avec l'extérieur de signaux logiques*
  - *dispose d'une base de temps (compteur - timer)*
- *Une carte E/S est constituée de :*
  - *Convertisseur Analogique - Numérique (CAN) : extérieur ⇨ PC*
  - *Convertisseur Numérique - Analogique (CNA) : PC ⇨ extérieur*
  - *Circuits d'Entrée-Sortie logiques*
  - *Circuits qui gèrent la base de temps ⇨ Timer*
- *Lecture et écriture dans les mémoires des composants (registres)*
- *Chacun de ces registres doit avoir sa propre adresse*
- *Détermination de ces adresses à partie de l'adresse de base de la carte E/S*

VIII.4 - Etude d'une carte d'acquisition : la carte « Impulsion »

VIII.4.1 Introduction

La carte d'interface est considérée par le PC comme un périphérique supplémentaire. Cette carte est dotée de composants électroniques possédant des mémoires appelées dans ce cas registres dans lesquels on ira lire ou écrire des données. Comme toute mémoire, on accède à un registres en le sélectionnant par le bus d'adresse et son contenu est modifié (écriture) ou lu par l'intermédiaire du bus de données.
La carte proposée occupe deux groupes d'adresses 16 bits sélectionnables par l'utilisateur au moyen de cavaliers. Le constructeur permet de choisir au préalable deux adresses dites de base à partir desquelles sont définies l'ensemble des adresses des différents registres de la carte. Ces adresses de base sont fixées à des valeurs non encore utilisées par le PC.



VIII.4.2 Description

La carte possède 2 groupes d'adresses sélectionnables par l'utilisateur :

groupe 1	*0x330*	*CAN, CNA, Timer*
groupe 2	*0x340*	*E/S logiques*

***CNA** : 1 convertisseur analogique - numérique 12 bits (8+4) :*
- *8 entrées analogiques*
- *gamme 0/+10V ou -5/+5V (par soft)*
- *temps de conversion environ 25µs*

***CAN** : 1 convertisseur numérique - analogique 12 bits (8+4)*
- *2 sorties analogiques indépendantes*

***TIMER** : 3 timers programmables (horloge externe ou interne à 4 MHz)*
***IO** : 8 entrées (TTL) et 8 sorties logiques (collecteur ouvert).*

VIII.4.3 La Conversion Numérique-Analogique

L'obtention d'une tension analogique est possible par l'intermédiaire d'un convertisseur numérique-analogique (CNA) qui convertit une valeur codée avec N bits en une tension analogique.
Un CNA peut être considéré comme un boîtier comportant :

- *1 registre N bits*
- *1 tension analogique de référence*
- *1 sortie analogique*

Tension analogique délivrée en sortie : $v_s = v_{ref} \cdot \frac{Nb}{2^N - 1}$ *N=taille registre,Nb=nbre à convertir*

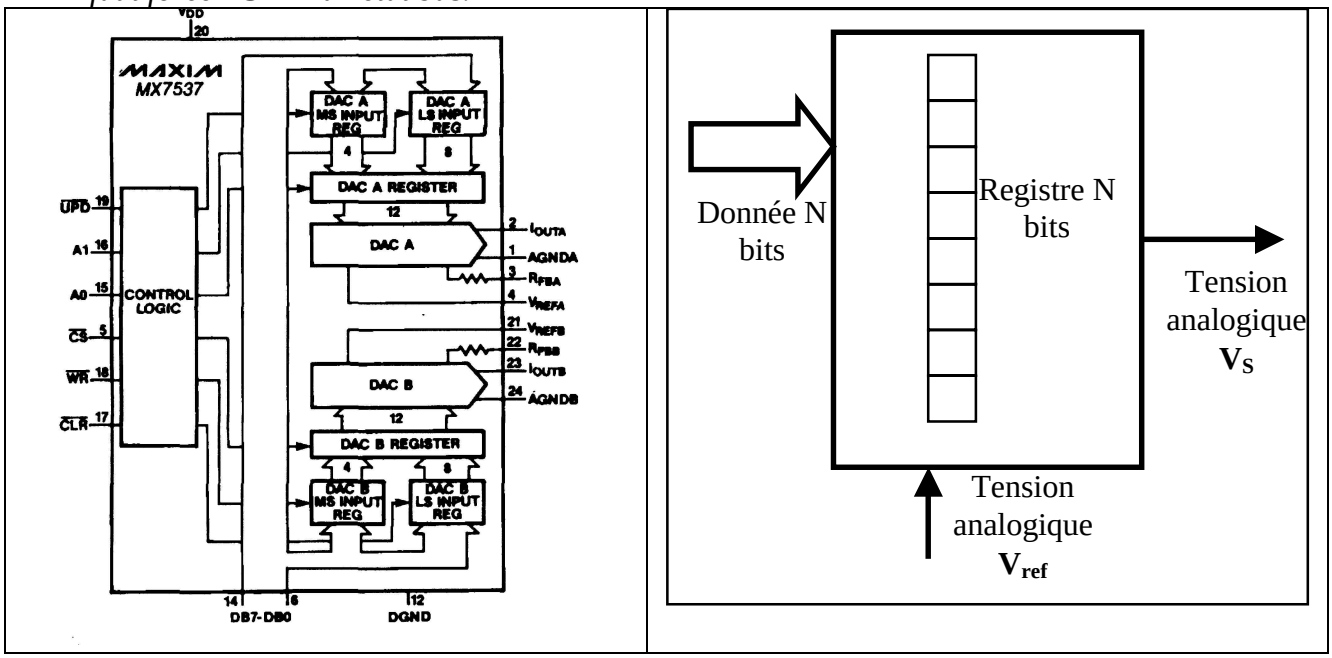
☞ *Quantification : 2.44 mV. Détermination du LSB (Less Significant Bit) : chaque convertisseur du boîtier est un convertisseur 12 bits (N=12). Pour une tension de référence de 10 volts, le code décimal allant de 0 à 4095, la précision du bit de poids faible (noté LSB) est de 2.44 mV.*

VIII.4.4 Etude du Convertisseur AD7537

Le circuit réalisant cette fonction et présent sur la carte est l'AD7537 qui contient deux convertisseurs dans son boîtier. Par conséquent, l'utilisateur dispose de deux sorties analogiques totalement indépendantes. En étudiant le diagramme de fonctionnement de ce circuit, on remarque qu'il est prévu pour se connecter sur un bus de données 8 bits. Le registre interne d'un CNA est scindé en deux parties (structure 8+4) : 8 bits de LSB et 4 bits de MSB (poids forts) ce qui oblige à charger une donnée en deux fois dans deux registres d'entrée temporaires (DAC AouB LS et DAC AouB MS) placés à deux adresses différentes. En fait, la donnée ne sera présente dans le registre 12 bits (DAC A ou DAC B register) que si on agit sur la broche (19) $\overline{\text{UPD}}$ *du convertisseur. Un signal logique 0 sur cette broche transfère simultanément les données des registres d'entrée vers les registres des deux CNA.*

Toute conversion se fait en 2 étapes :

1. *chargement dans 2 registres d'entrée A ou B (2 adresses ≠)*
2. *lancement de la conversion (= charger le registre du DAC) en agissant sur la broche 19 (*$\overline{\text{UPD}}$*). Il faut forcer* $\overline{\text{UPD}}$ *à l'état bas.*

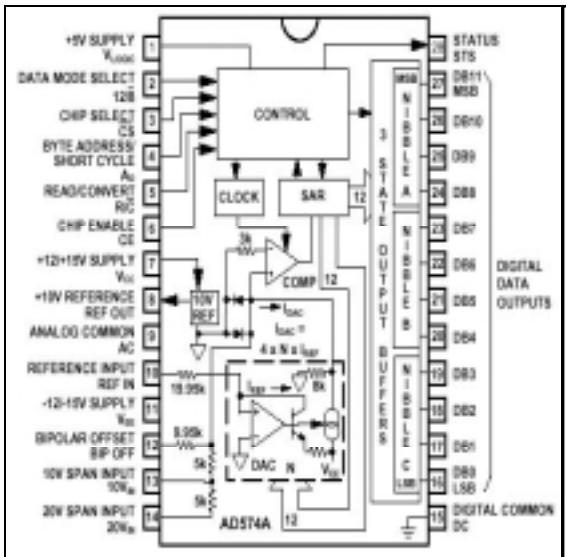


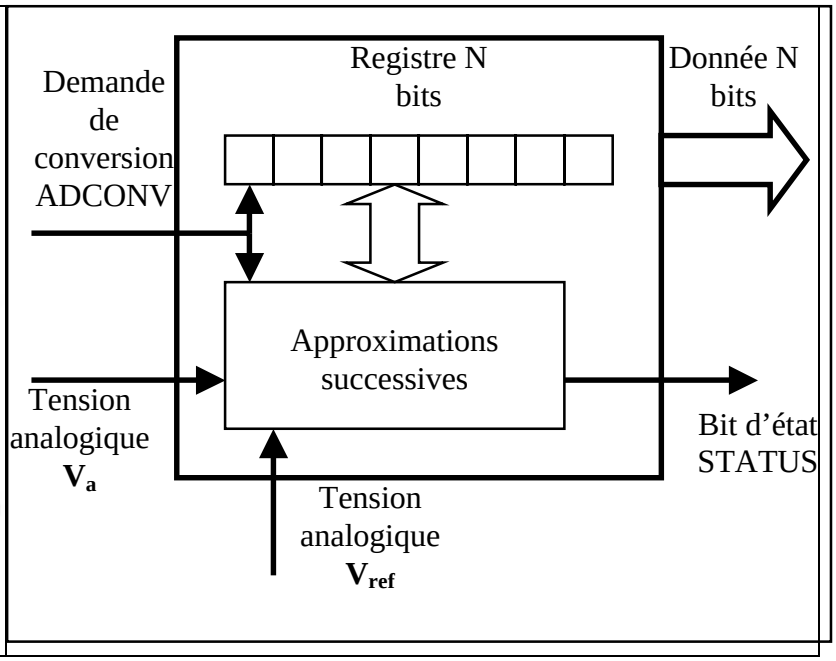
- *5 adresses suffisantes pour gérer le DAC :*
  - *2 adresses pour les registres du DAC A*
  - *2 adresses pour les registres du DAC B*
  - *1 adresse pour activer* $\overline{\text{UPD}}$

Les 2 registres d'entrée ne sont validés qu'en écriture et $\overline{\text{UPD}}$ *est activé par l'intermédiaire d'un décodeur d'adresse. C'est donc l'adresse (et non la donnée) et le mode d'adressage qui importent.*

DA1L=ADR1+4	D7	D6	D5	D4	D3	D2	D1	D0	W
	8 LSB du DAC A								
DA1H=ADR1+5					D11	D10	D9	D8	W
	4 MSB du DAC A								
CONVDA=ADR1+7	lancement de la conversion								R

VIII.4.5 Le Convertisseur Analogique-Numérique AD574





Opération inverse du CNA : L'acquisition d'un signal analogique est possible grâce au convertisseur analogique-numérique qui convertit une tension analogique en un mot de N bits. Le circuit réalisant cette fonction est l'AD574. Il contient un seul convertisseur dans son boîtier. La conversion analogique-numérique se fera en trois étapes :

*① - déclenchement de conversion par une opération de lecture à l'adresse **adconv** d'un registre logique*

*② - attente fin de conversion (≈25µs) ⇨ test du bit **busy**. Tant que conversion en cours, busy 'état haut*

③ - lecture du résultat en 2 fois. On utilisera 2 registres en lecture aux adresses ADH et ADL

Conversion :

- *La technique de conversion se fait par approximations successives (voir cours), la conversion n'est pas immédiate et nécessite un temps de conversion d'environ 25 µs pour l'AD574. Pendant que la conversion est en cours, le boitier le signale par le bit d'état **BUSY** à l'état haut. L'attente de fin de conversion peut être réalisée soit par une boucle d'attente de longueur supérieure ou égale au temps de conversion soit par lecture dans un registre logique (**adh**) du signal **BUSY** qui passe à l'état bas lorsque la conversion est finie et que le résultat peut être lu.*

Multiplexage :

- *Le convertisseur analogique-numérique est en général le circuit le plus coûteux. Par conséquent, lorsque l'on souhaite plusieurs voies d'entrées analogiques, on préfère adjoindre au CAN un multiplexeur analogique plutôt que de multiplier ce circuit. Ce multiplexeur permet de choisir par programmation entre X voies d'entrées celle qui sera connectée au CAN. Un tel circuit 8 voies est utilisé sur la carte étudiée. La voie est choisie par écriture de son numéro dans un registre 8 bits d'adresse **mux**.*

Polarité :

- *L'un des bits non utilisés dans ce registre **mux** a été utilisé pour commander le mode de polarité du CAN ce qui évite un registre supplémentaire. Choix de 2 modes de conversion : le mode unipolaire (0 - V_{ref} volts) ou bipolaire (-V_{ref}/2 - +V_{ref}/2) par simple action d'un signal logique ce qui consiste en une écriture dans le registre d'adresse **mux**. Si **pol** = 0, mode bipolaire sinon mode unipolaire.*

ADL=ADR1+5	D7	D6	D5	D4	D3	D2	D1	D0	R
	8 LSB du CAN								
ADH=ADR1+4	busy				D11	D10	D9	D8	R
	4 MSB du CAN + bit busy								
ADCONV=ADR1+6	lancement de la conversion								R
MUX=ADR2+1					NX_2	NX_1	NX_0	POL	W

→ pol=1:0/10V pol=0 :-5/+5V

Protocole pour le lancement d'une acquisition

① - choix de la polarité et du numéro de la voie d'acquisition

*② - action sur **adconv** par une opération de lecture et test de busy*

③ - lecture de la conversion en 2 fois (ADH et ADL)

Pour la carte utilisée la valeur de V_{ref} est de 10,24 Volts ce qui assure un LSB à exactement 25 mV.

VIII.4.6 Etude du Timer 8254

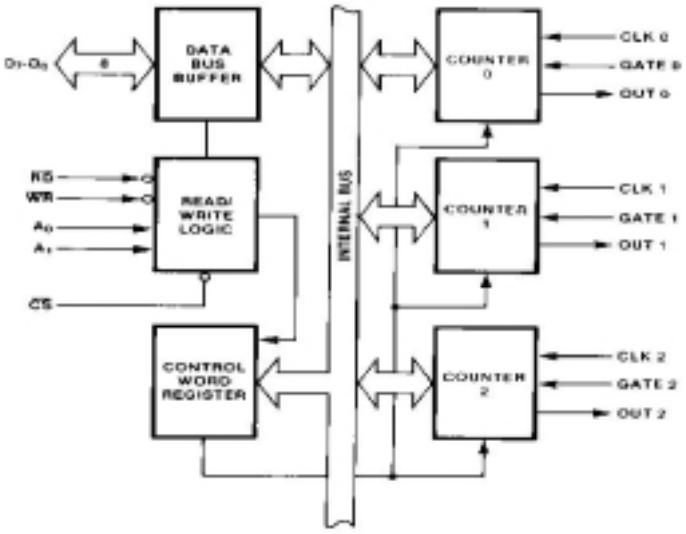


Figure 1. 82C54 Block Diagram

*Le timer 8254 est inséré sur une carte d'acquisition avec un bus de donnée de **8 bits**. Le registre d'entrée est un registre **16 bits** ce qui oblige à écrire la donnée en 2 fois (poids faible d'abord). Le registre de comptage est aussi un registre **16 bits**. Les registres de contrôle et d'état sont des registres **8 bits**.*

VIII.4.7 Fonctionnement du timer

*Pour commencer, il faut charger le registre d'entrée avec un nombre de son choix (0 à 65535). Le front de l'horloge suivant provoque le transfert de ce nombre dans le registre de comptage où il sera décrémenté à chaque front descendant de l'horloge. Lorsque le compte est arrivé à 0, soit le compteur attend une nouvelle initialisation (mode dit monocoup), soit il se réinitialise avec le nombre toujours présent dans le registre d'entrée et on recommence le décomptage (mode cyclique ou périodique). On peut aussi demander un signal de sortie symétrique (signaux carrés) ou impulsionnel. Une tension d'entrée logique "**gate**" autorise ou non le décompte. L'ensemble des caractéristiques de fonctionnement (monocoup, périodique, forme du signal de sortie et rôle du gate) définit le mode de fonctionnement.*

Le 8254 possède 3 compteurs 16 bits indépendants : timer 0, timer 1 et timer 2.

*① - <u>le registre d'entrée</u> : chaque compteur a son propre registre d'entrée 16 bits séparé en deux octets (MSByte-LSByte). Chacun de ces registres est accessible en écriture aux adresses **count0** (resp. **count1**, **count2**) pour le timer0 (resp. timer1, timer2). Il y a donc 3 registres d'entrée. Bus de données 8 bits, écrire dans ce registre en 2 fois : $1^{ère}$ écriture → LSB, $2^{ème}$ écriture → MSB (logique interne).*

*② - <u>le registre de contrôle</u> : on devrait avoir trois registres de contrôle. En réalité, les trois registres sont multiplexés à l'intérieur du boîtier et l'aiguillage est assuré par les deux bits de poids fort de ce registre noté **timcw (timer control word)**. <u>Avantage</u> : on n'a besoin que d'une seule adresse pour les 3 timers. Les autres bits du mot de contrôle servent à définir le mode de fonctionnement du compteur sélectionné. Pour ce boîtier, il y a cinq modes disponibles :*

- *Mode 0 : interruption à la fin du comptage (monocoup).*
- *Mode 1 : monostable réarmable mono coup.*
- *Mode 2 : oscillateur diviseur par N (périodique).*
- *Mode 3 : générateur de signaux carrés (périodique).*
- *Mode 4 : générateur d'impulsions synchronisé par programme (soft).*
- *Mode 5 : générateur d'impulsions synchronisé par gate (hard).*

*③ - le registre d'état : outre l'accès aux registres de contrôle des trois compteurs (codes 00, 01 ou 10), il y a une 4ème possibilité : le code 11 (bits D7-D6) qui est la commande de **readback**. Le 8254 possède trois registres 16 bits supplémentaires appelés latchs, mémoires ou tampons. La commande **readback** permet de recevoir des informations du compteur (lecture des comptes ou lecture d'un mot d'état). Dans cette commande, on peut soit "latcher" simultanément le contenu de plusieurs compteurs ce qui donne le temps de lire le contenu en temps différé, soit "latcher" l'état des compteurs (le mot d'état de 8 bits contient en particulier un flag reflétant l'état de sortie du compteur). Il faut dans ce cas tester le bit **Output** du registre d'état. Les adresses de ces trois latchs sont celles déjà utilisées pour les registres d'entrée mais cette fois avec un accès en lecture. La détection fin de comptage est réalisée en scrutant le bit **D7 Output** du **Status Byte** : la sortie est à 1 pendant le décompte et passe à 0 en fin de comptage durant une période d'horloge (lire count0 tant que ([count0] & 0x80) = 0x80 puis lire count0 tant que ([count0]&0x80) = 0).*

***Résumé** : **adr1** étant l'adresse de base de la carte (soit **0x330** ou **0x250**), on a besoin de 4 adresses seulement pour gérer les 3 timers : **count0** = adr1|**count1** = adr1+1|**count2** = adr1+2|**timcw** = adr1+3.*

VIII.5 - Annexes

VIII.5.1 Les Fonctions du Langage C

*La communication avec les registres de la carte s'effectue avec les fonctions **inp** et **outp** :*

- *char **inp**(int adresse_port) : lit un octet sur le port situé à l'adresse indiquée et le retourne*
- ***outp** (int adresse_port, char val) : écrit la valeur val sur le port situé à l'adresse indiqué.*

***Programmation** : on veut envoyer la tension Vout sur le DAC A.*

① - convertir la tension Vout en un nombre de bits Nb $2^{12} - 1 = 4095 \rightarrow 10V$; $Nb \rightarrow Vout$

② - convertir ce résultat réel en entier

③ - récupérer les 8 bits de poids faible et les placer dans DA1L

④ - récupérer les 4 bits de poids fort et les placer dans DA1H

⑤ - déclencher la conversion - action sur $\overline{\text{UPD}}$ *en lecture*

↳ *pour accéder aux 8 bits de poids faible : → reste de la division entière du nombre par 256*

↳ *pour accéder aux 8 bits de poids fort : → quotient de la division entière du nombre par 256*

VIII.5.2 Tableau récapitulatif des adresses de la carte « Impulsion »

FONCTION	ADRESSE	Lect R	Ecri.W	DONNEE
Convertisseur Numérique Analogique CNA	DA1L = ADR1+4		W	8 LSB du CNA A
	DA1H = ADR1+5		W	4 MSB du CNA A
	DA2L = ADR1+6		W	8 LSB du CNA B
	DA2H = ADR1+7		W	4 LSB du CNA B
	CONVDA = ADR1+4	R		UPD Charg. registres
Convertisseur Analogique Numérique CAN	ADH = ADR1+4	R		4 MSB du CAN + STATUS
	ADL = ADR1+5	R		8 LSB du CAN
	ADCONV = ADR1+6	R		Lancement conversion
	MUX = ADR2+1		W	Multiplexeur+polarité
Compteurs (timers)	COUNT0 = ADR1	R	W	Compte du compteur 0
	COUNT1 = ADR1+1	R	W	Compte du compteur 1
	COUNT2 = ADR1+2	R	W	Compte du compteur 2
	TIMCW = ADR1+3		W	Registre de contrôle
Fonctions logiques	*IOIN = ADR2*	*R*		*Entrées logiques*
	IOOUT = ADR2		*W*	*Sorties logiques*

VIII.5.3 Registres du Timer

Control Word Format

A_1, A_0 = 11 $\overline{CS}$ = 0 $\overline{RD}$ = 1 $\overline{WR}$ = 0

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC1	SC0	RW1	RW0	M2	M1	M0	BCD

SC — Select Counter:

SC1	SC0	
0	0	Select Counter 0
0	1	Select Counter 1
1	0	Select Counter 2
1	1	Read-Back Command (See Read Operations)

M — MODE:

M2	M1	M0	
0	0	0	Mode 0
0	0	1	Mode 1
X	1	0	Mode 2
X	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

RW — Read/Write:

RW1	RW0	
0	0	Counter Latch Command (see Read Operations)
0	1	Read/Write least significant byte only.
1	0	Read/Write most significant byte only.
1	1	Read/Write least significant byte first, then most significant byte.

BCD:

0	Binary Counter 16-bits
1	Binary Coded Decimal (BCD) Counter (4 Decades)

NOTE: Don't care bits (X) should be 0 to insure compatibility with future Intel products.

Figure 7. Control Word Format

A_1, A_0 = 11; $\overline{CS}$ = 0; $\overline{RD}$ = 1; $\overline{WR}$ = 0

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC1	SC0	0	0	X	X	X	X

SC1, SC0 - specify counter to be latched

SC1	SC0	Counter
0	0	0
0	1	1
1	0	2
1	1	Read-Back Command

D5,D4 - 00 designates Counter Latch Command

X - don't care

NOTE:
Don't care bits (X) should be 0 to insure compatibility with future Intel products.

Figure 9. Counter Latching Command Format

A0, A1 – 11 $\overline{CS}$ – 0 $\overline{RD}$ – 1 $\overline{WR}$ – 0

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	1	$\overline{COUNT}$	$\overline{STATUS}$	CNT 2	CNT 1	CNT 0	0

D_5: 0 – Latch count of selected counter(s)
D_4: 0 – Latch status of selected counter(s)
D_3: 1 – Select counter 2
D_2: 1 – Select counter 1
D_1: 1 – Select counter 0
D_0: Reserved for future expansion; must be 0

Figure 10. Read-Back Command Format

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
OUTPUT	NULL COUNT	RW1	RW0	M2	M1	M0	BCD

D_7 1 – Out Pin is 1
0 = Out Pin is 0
D_6 1 – Null count
0 = Count available for reading
D_5-D_0 Counter Programmed Mode (See Figure 7)

Figure 11. Status Byte

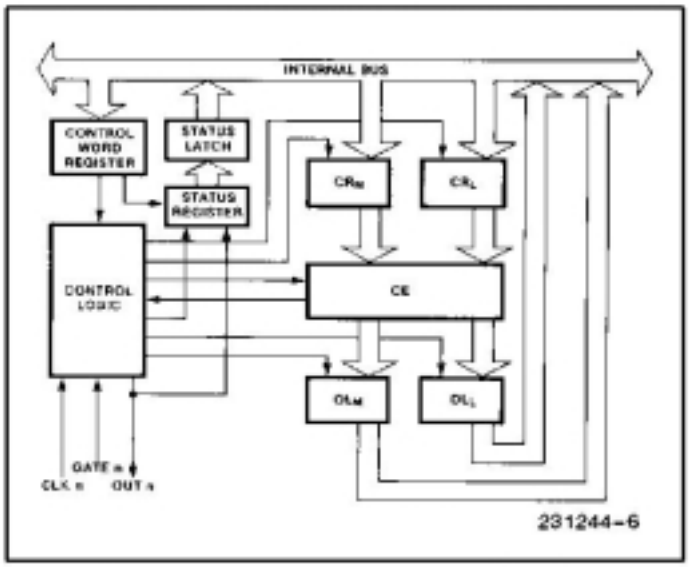


Figure 5. Internal Block Diagram of a Counter

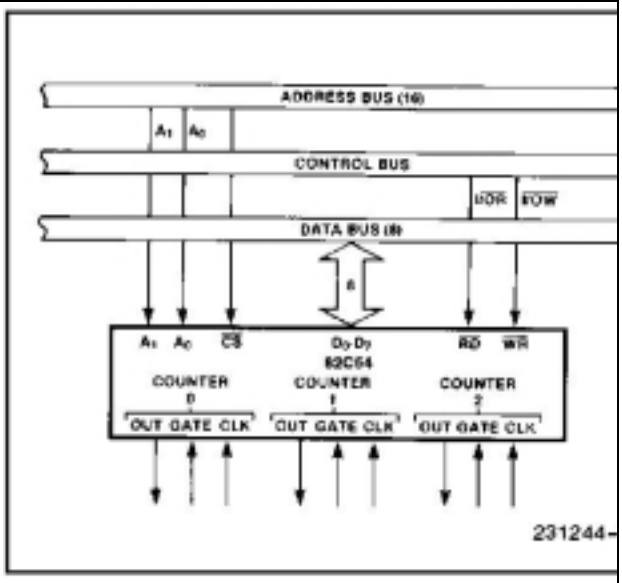


Figure 6. 82C54 System Interface

IX. CONCLUSIONS

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 intéressante (gestion de type réseau).

Dans sa première version, l´USB était jusqu´alors limité à des périphériques lents (claviers, imprimantes, souris, webcam). Avec sa version 2.0, il investit des domaines qui étaient jusqu´alors réservés à l´IEEE 1394 ou au SCSI (périphériques de stockage externe ou 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.

Type	Débit (Mbit/s)	Débit (Mo/s)
Port série	0.115	0,015
I2C (rapide)	0.400	0,050
Port parallèle (standard)	0.920	0,115
I2C (new)	3,2	0,425
USB 1.0	12	1,5
Port parallèle (ECP/EPP)	24	3
IDE	26,4 - 133,6	3,3 - 16,7
SCSI-1	40	5
IEEE 488.2 (new GPIB)	64	8
SCSI-2 (fast SCSI, fast narrow SCSI)	80	10
Fast wide SCSI	160	20
Ultra SCSI (SCSI-3, Fast-20, Ultra narrow)	160	20
Ultra IDE	264	33
Wide Ultra SCSI (Fast wide 20)	320	40
Ultra-2 SCSI	320	40
IEEE-1394	100 - 400	12,5 - 50
USB 2.0	480	60
Wide Ultra-2 SCSI	640	80
Ultra-3 SCSI	640	80
Wide Ultra-3 SCSI	1280	160
PCI Express	(6x) 1600	(6x) 200
IEEE 1394b, 1394.1, Fiber Chanel AL	800 - 3200	100 - 400