Unités d'entrée/sortie UE

On utilise ces CI d'entrée/sortie en écrivant et lisant des valeurs dans des registres internes:

- ➔ Registre de données Ecriture et lecture
- ➔ Registre de commande ou contrôle (en écriture)
- ➔ Registre d'état (Status) (en lecture)

Caractéristiques

* Synchrone / Asynchrone:

- Synchrone: Une horloge est échangée => Pas d'écart, de dérives
  (plus compliqué mais meilleur débit)
- Asynchrone: Pas d'horloge commune => Dérives => Messages courts

*Série: Renvoi des bits les uns après les autres

Intel: UART (Universal Asynchronous Receiver Transmitter)
Motorola: ACIA (Asynchronous Communication Interface Adapter)

* Parallèle: Envoi des bits ensemble en parallèle => Plus rapide
=> Plus compliqué car diaphonie, beaucoup de fils

Intel: PIO (Parallel Input/Output)
Motorola: PIA (Peripheral Interface Adapter)

* Spécialisées:

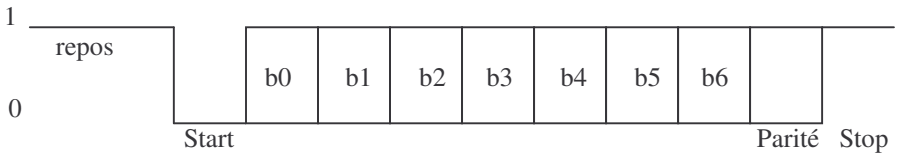
Contrôleurs USB, Ethernet, HDLC,...

Interface Série V24 (RS232)

Signaux: 0: +12 V 1: -12 V
Start = 0 Stop et repos =1 Parité: Nombre de 1 de (data et parité) du type indiqué

Trame:

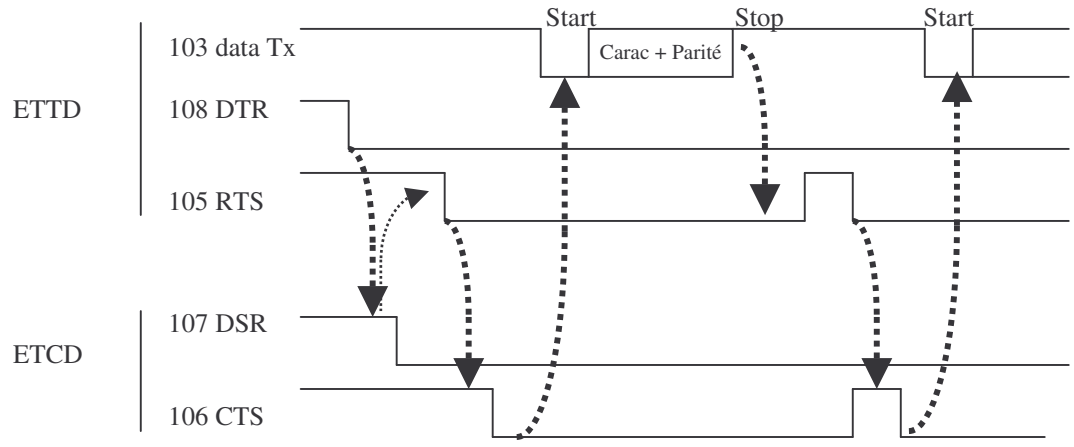
Ex: 7-P-1



Connecteur DB9:

	DP (CD) 109	RD (RX) 104	ED (TX) 103	TDP (DTR) 108	TS (SG) 102	PDP (DSR) 107	DPE (RTS) 105	PAE (CTS) 106	IA (RI) 125
Broches sur DB25	8	3	2	20	7	6	4	5	22
Broches sur DB9	1	2	3	4	5	6	7	8	9

Gestion des contrôles:



- ① Le logiciel sur l'ETTD active le 108 DTR pour interroger s'il y a un ETCD.

- ② L'ETCD répond par le 107 DSR qu'il est prêt à échanger avec l'ETTD.

- ③ Si ensuite, l'ETTD veut émettre une donnée => Il active RTS = 105.

- ④ L'ETCD indique qu'il est prêt à recevoir la donnée de l'ETTD et à l'émettre en ligne => Il active CTS = 106

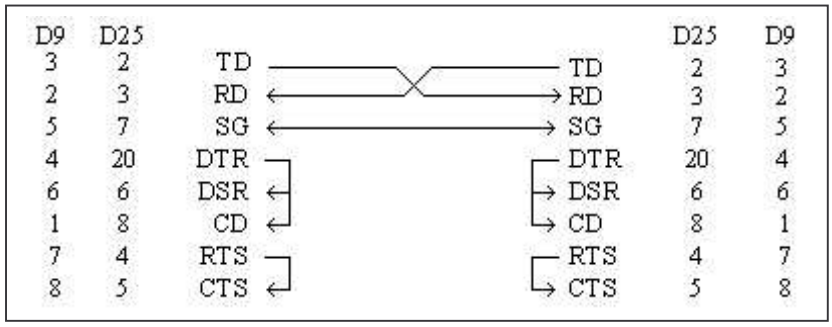
- ⑤ L'ETTD envoie sa donnée sur le 103 Tx.

- ⑥ L'ETTD désactive 105 RTS une fois la donnée émise.

Remarque: on peut éliminer les contrôles en les rebouclant sur la même machine.

Câble croisé null modem (PC à PC):

Contrairement au câble droit ETTD/ETCD (PC à modem) où on relie les mêmes broches sur les 2 connecteurs,
pour le câble null modem
- on croise Tx et Rx
- on annule les contrôles



UART Intel 8250

I) Présentation du circuit

sur PC par défaut: COM1 : 0x3F8 COM2 : 0x2F8

Il y a plus de 8 registres différents et on n'a sur le circuit que 3 bits d'adressage interne.
=> Utilisation du bit DLAB qui va lever les ambiguités. Suivant DLAB, on sélectionne tel ou tel registres.

Adresse			Registre	
A2	A1	A0	si DLAB = 0	si DLAB = 1
0	0	0	RBR : Receiver Buffer (registre de réception) en lecture THR : Transmitter Holding Register (registre d'émission) en écriture	DLL : Divisor Latch LSB (poids faible diviseur horloge)
0	0	1	IER : Interrupt Enable Register	DLM : Divisor Latch MSB (poids fort diviseur horloge)
0	1	0	IIR : Interrupt Identification Register	
0	1	1	LCR : Line Control Register	
1	0	0	MCR : Modem Control Register	
1	0	1	LSR : Line Status Register	
1	1	0	MSR : Modem Status Register	
1	1	1	Non utilisé	

Exemple : si l'adresse de base de l'interface est par exemple 3F8H, l'adresse du registre RBR sera 3F8H + 000 = 3F8H et celle de IIR 3F8H + 010b = 3F8H + 2H = 3FAH.
On voit, comme nous l'avons dit plus haut, que les registres DLM et IER (par exemple) possèdent la même adresse (001b). Si le bit DLAB est 0, cette adresse permet d'accéder à DLM, et si DLAB=1 à IER.
Le bit DLAB est le bit de poids fort du registre LCR.

Choix de la rapidité de modulation = Débit 115200 / vitesse = 256 * DLM + DLL

Registres de données: Registre THR : **Registre d'émission.** (broche WR active)
Registre RBR : **Registre de réception.** (broche RD active)

Registre LCR (Line Control Register) : Registre de commande pour définir certains paramètres de la transmission.

B7	B6	B5	B4	B3	B2	B1	B0
DLAB	0	0	0 : parité impaire 1 : Parité paire	0: Sans parité 1: Avec parité	0 : 1 bit stop 1 : 2 bits stop	B1B0=00 : 5 bits de data B1B0=01 : 6 bits de data B1B0=10 : 7 bits de data B1B0=11 : 8 bits de data	

Registre IER : Utilisé pour les entrées/sorties par interruption.

Bit 0 : interruption lorsque donnée reçue dans RBR;
Bit 1 : interruption lorsque registre THR vide;
Bit 2 : interruption lorsque l'un des 4 bits de poids faible de LSR passe à 1;
Bit 3 : interruption sur état du modem.

Registre LSR : Registre d'état indiquant le fonctionnement en réception (bits 0 à 4) et en émission (bits 5 et 6).

Bit 0	passe à 1 lorsqu'une donnée a été reçue et chargée dans RBR.
Bit 1	signale erreur d'écrasement. 1 si donnée reçue et chargée dans RBR alors que la précédente n'avait pas été lue. Remis automatiquement à 0 par la lecture du registre LSR.
Bit 2	passe à 1 à la suite d'une erreur de parité. Remis à 0 par la lecture de LSR.
Bit 3	passe à 1 lorsque le niveau du bit STOP n'est pas valide (erreur de format). Remis à 0 par la lecture de LSR.
Bit 4	passe à 1 lorsque le niveau de la liaison est resté à 0 pendant la durée d'émission de la donnée (problème de l'emetteur). Remis à 0 par la lecture de LSR.
Bit 5	passe à 1 lorsqu'une donnée est transférée de THR vers le registre à décalage d'émission (THR est alors libre pour le caractère suivant).
Bit 6	passe à 1 lorsque le registre à décalage d'émission est vide.
Bit 7	Toujours à 0

Registre IIR : IIR est utilisé pour les interruptions. Il permet d'identifier la cause de l'interruption émise par l'UART.

Registre MCR : MCR est le registre de commande du modem.

Bit 0 : commande le niveau de DTR qui informe le modem que le port série du PC est prêt à communiquer;
Bit 1 : commande RTS: Le port série du PC demande au modem s'il est prêt à recevoir une donnée.
…

Registre MSR : MSR est le registre d'état du fonctionnement du modem.

Bit 0 : Changement de CTS
Bit 1 : Changement de DSR
Bit 2 : Changement de RING
…

II) Programmation de l'interface en langage C

Init : DLAB à 1 : ouptortb(LCR,0x80)
DLL en adbase
DLM en adbase+1
Config en LCR

Lecture_bit_registre :
Lire val, Masque=1, masque=masque<<n°bit, val = val&masque (& = ET binaire pas logique)
Si val nul alors return 0 sinon return 1

Réception carac par scrutation (polling) :
Tant que bit 0 de LSR = 0 : attendre
Lire la registre de données

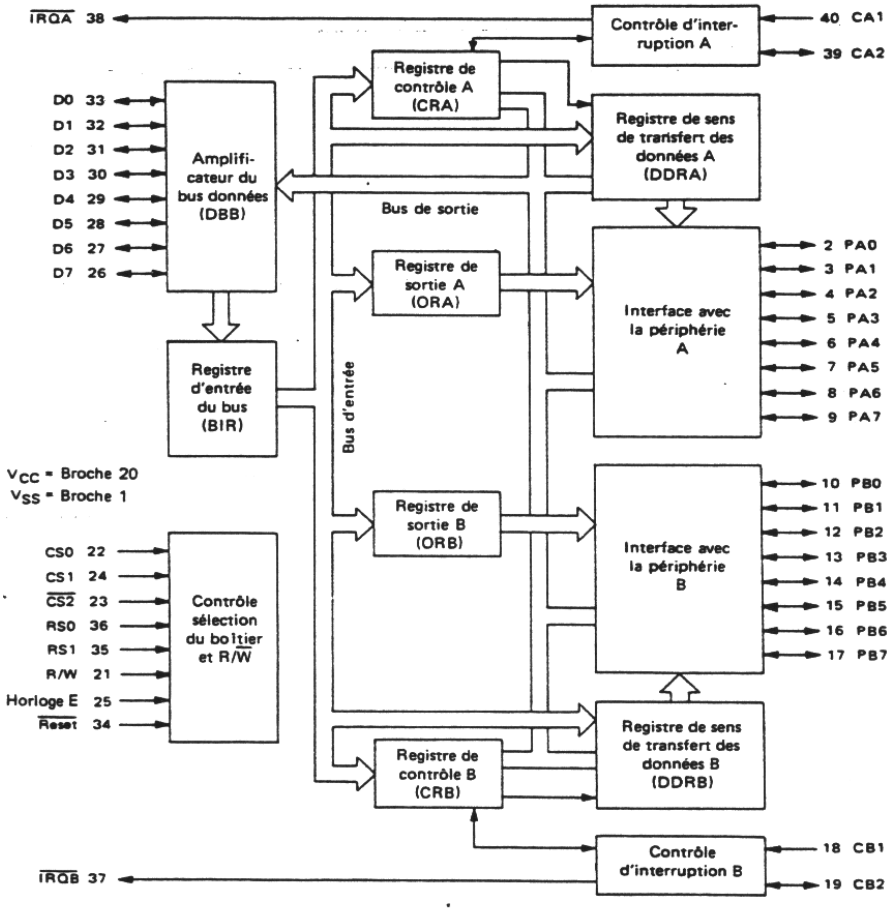
Emission carac :
Ecrire le registre de données
Tant que bit 6 de LSR = 0 : attendre

Interruptions :
Il faut détourner les interruptions
(void interrupt (*ancien_sousprg), void interrupt nouveau_sousprog(); ,
getvect , setvect avec 0x0C pour COM1, 0x0B pour COM2)
Il faut autoriser les interruptions sur le PIC (mettre à 0 le bit 4 pour COM1 ou 3 pour COM2 de 0x21)
Remarque : Sur certaines cartes, la sortie OUT2 de l'UART intervient dans le contrôle d'interruption.
Il est parfois nécessaire de la mettre à 1 (bit 3 mis à 1 dans MCR)
Il faut acquitter les interruptions dans le sous prog : outportb(0x20,0x20)

Port parallèle Motorola PIA 6821

Le PIA 6821 (Peripherical Interface Adapter) donne 2 ports parallèles bidirectionnels 8 bits et 4 lignes de contrôle. Chacune des lignes de données vers la périphérie (Port A et B) peut être programmée pour être utilisée soit en entrée, soit en sortie à partir des registres internes et du bus de données.

- bus de données (D0 à D7) bidirectionnel vers le MPU
- 2 bus de données (ou port A-B) bidirectionnels vers la périphérie
- 2 registres de contrôle programmables
- 4 lignes d'entrée d'interruption contrôlables individuellement



Lien µP / UE:

Bus de données (D0 à D7) : 8 lignes bidirectionnelles directement reliées au bus du µP.

Bus d'adresses :

- CS0, CS1, /CS2 (Chip Select) : Sélectionnent le boîtier PIA (CS0·CS1·/CS2 = 1·1·0)
- RS0, RS1 (Register Select) : Sélectionnent les registres internes (=> 4 adresses de registres).

Bus de contrôle :

- E : Signal d'activation des échanges
- /RESET : Initialisation du PIA. Les registres internes sont mis à zéro.
- R/W : Lecture - Ecriture
- IRQA, IRQB : Lignes d'interruptions

Lignes UE/extérieur: PA0 à PA7, PB0 à PB7 : Suivant la programmation du DDRA/DDRB, ces broches seront utilisées en entrée ou en sortie.

FONCTIONNEMENT :

De la périphérie vers le processeur :

La donnée disponible sur le port A en réception (port A configuré en entrée) est directement transmise à l'amplificateur de bus de données par l'intermédiaire du bus de sortie. Elle ne transite pas par l'ORA => pas mémorisation des données en entrée.

Du processeur vers la périphérie :

La donnée disponible sur le bus du MPU est chargée dans le registre de sortie B par l'intermédiaire du bus d'entrée, elle est donc mémorisée. Le port B est en sortie, la donnée est disponible tant qu'une nouvelle écriture n'est pas intervenue.

Sélection des registres :

Bus d'adresses		MPU	A15 ..... A2			A1	A0			
			Logique de décodage							
		PIA	CS0	CS1	$\overline{CS2}$	RS1	RS0	CRA2	CRB2	Adresses
REGISTRES	A	CRA	1	1	0	0	1	—	—	ADR + 1
		DDRA	1	1	0	0	0	0	—	ADR
		ORA	1	1	0	0	0	1	—	ADR
	B	CRB	1	1	0	1	1	—	—	ADR + 3
		DDRB	1	1	0	1	0	—	0	ADR + 2
		ORB	1	1	0	1	0	—	1	ADR + 2

* An = ligne adresse du MPU.
* CSn = ligne sélection de boîtier.
* RSn = ligne sélection de registre.
* CRX = registre de contrôle.
* ADR = adresse de base résultante de la logique de décodage.

Pour accéder à un registre, l'adresse
- sélectionne le boîtier PIA (décodage d'adresse commandant CSX)
- sélectionne le registre interne par RS1 et RS0.

Ex :
ADR = 0xE000 (adresse de base)
ADR+1 = 0xE001
ADR+2 = 0xE002
ADR+3 = 0xE003

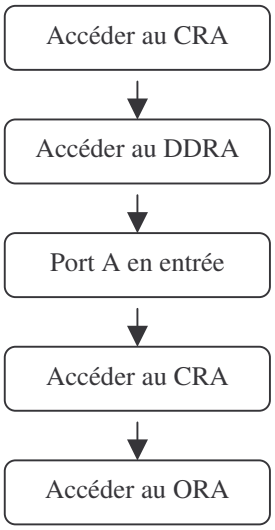
Registre de contrôle CRA - CRB :
Ce registre contrôle les ports de sortie A et B.
On y accéde en ADR+1 (états sur RS1 et RS0).
Ce registre va permettre d'accéder aux autres registres : DDRA-DDRB et ORA-ORB à partir du bit 2 du registre de contrôle : 0 -> Accès à DDR,
1-> Accès à OR

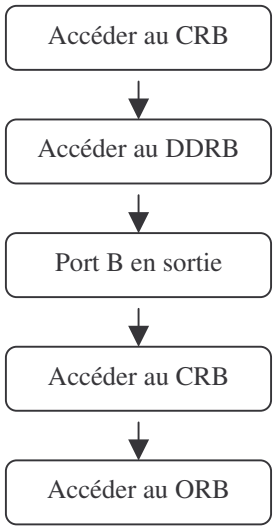
	7	6	5	4	3	2	1	0
CRA	IRQA1	IRQA2	Contrôle de CA2			Accès à DDRA	Contrôle de CA1	
CRB	IRQB1	IRQB2	Contrôle de CB2			Accès à DDRB	Contrôle de CB1	

Registres de sortie ORA – ORB :
Registres dans lesquels vont transiter les données. Pour lire une donnée reçue par un port, il suffira de lire le contenu de ce registre.

Registres de direction DDRA – DDRB :
Ils définissent le sens de transfert de chacune des lignes des ports A et B. 0 -> ligne en entrée.
1 -> ligne en sortie,
Ex : pour mettre la ligne PB4 en entrée et toutes les autres en sortie, il faut écrire la valeur 0xEF dans le registre DDRB.

PROGRAMMATION D'INITIALISATION DU PIA :
Exemple port A en entrée, port B en sortie.





Port parallèle de PC: Intel

La liaison série sur un PC s'accompagne de limitations de débit et de distance.

=> Pour améliorer la distance: Modems avec réseau d'un opérateur de Télécoms.

=> Pour améliorer le débit: Faire passer plusieurs bits en parallèle.

Mais la diaphonie (interférences électromagnétiques entre fils) limitent encore plus la distance.

Et il faut plus de fils => coût plus élevé.

Pour les systèmes à microprocesseur, on dispose de circuit simple pour la liaison parallèle comme le PIO 8255 similaire au PIA 6821. Pour les PC, on utilise des systèmes plus performants.

I) Historique

En 1981, IBM inclut un port parallèle sur le PC pour des transmissions plus rapides que les liaisons séries:
Mode standard : SPP unidirectionnel 8 bits(écrire vers une imprimante).

Mode quartet (Nibble mode): **"Bidirectionnel" 4 bits** en utilisant 4 lignes de contrôle entrantes.

Mode octet (Byte mode): **Bidirectionnel 8 bits:** Les circuits électroniques donnent des lignes bidirectionnelles, plus besoin de mettre des données sur les lignes de contrôle.

Mais la transmission est **limitée en débit** (150 ko/s sortant) car demandant beaucoup de traitements logiciels (pas de DMA), n'est pas définie par un standard électrique => problèmes d'interopérabilité, est limitée en distance par la puissance des drivers de ligne électroniques.

Mode EPP = IEEE1284: En 1991 **bidirectionnel "haut débit"** (plus de 1 Mo/s), compatible SPP, Quartet, Octet.
De débit inférieur à celui du bus ISA (8Mo/s), il permet l'émission/réception avec des périphériques amovibles avant USB (lecteurs de CD-ROM ou disques durs).

Encore plus récent, le **port ECP** (Extended Capacity Port): **Plug and Play** (branchez et utilisez) et **transferts améliorés** (traitements logiciels, compression, canaux logiques, ...)

Les modes Standard et quartet peuvent être utilisés sur tout circuit parallèle.
Les autres modes demandent que le circuit électronique soit prévu pour ces fonctionnements.
Le choix du mode de fonctionnement SPP, EPP ou ECP se fait dans le BIOS.

II) Description du port parallèle

Le port parallèle sur PC est fait de 17 signaux et 8 broches à la masse. Les signaux sont de 3 catégories :

+ Signaux de contrôle (4) : Ecriture pour contrôle de liaison (Handshake = enchaînement de requêtes/réponses pour le déroulement de la transmission).

+ Signaux d'état (5) : Lecture des indication de l'état des appareils, de cause de problème.

+ Signaux de données (8) : Envoi de données du PC vers le périphérique seulement à l'origine.

Dans la définition originale (SPP : Standard Parallel Mode), le port parallèle du PC est dédié aux imprimantes c'est pourquoi on trouve des noms de broches liés à l'impression :

Groupe	SPP Signal	In/Out	Signal Description
Contrôle	/STROBE	Out broche 1	Actif bas. Le PC indique la présence certaine de données valides sur la ligne (elles le sont avant).
	/AUTOFEED	Out broche 14	Actif bas. Indique à l'imprimante d'insérer automatiquement un saut de ligne en plus du retour chariot à chaque return.
	/SELECT IN	Out broche 17	Actif bas. Indique à l'imprimante qu'elle est sélectionnée.
	/INIT	Out broche 16	Actif bas. Utilisé pour envoyer un reset à l'imprimante.

Etat	/ACK	In broche 10	L'imprimante envoie une impulsion à 0 sur cette ligne pour indiquer à l'ordinateur qu'elle a bien reçu le caractère transmis et qu'il peut continuer la transmission.
	BUSY	In broche 11	Cette ligne est mise à 1 par l'imprimante lorsque son buffer de réception est plein (contrôle de flux matériel). Le PC doit attendre que cette ligne revienne à 0 pour ré-émettre.
	PE	In broche 12	L'imprimante indique qu'il n'y a plus de papier (Paper Error)
	SELECT	In broche 13	Cette ligne indique à l'ordinateur si l'imprimante est "on line" ou "off line".
	/ERROR	In broche 15	L'imprimante indique une erreur au PC.
Données	DATA[8:1]	Out	8 lignes de données- seulement en sortie en SPP. broche 2 = D0, 3 = D1, 4 = D2, 5 = D3, ..., 9 = D7

Broches 18 à 25 = Masse

Remarque: /A ou nA indique « Complément de A ». Ici cela indique des variables actives sur niveau bas.

Le lien entre le matériel et le logiciel se fait par des registres : Le port parallèle est installé à une adresse de base ADR (378h ou 278h habituellement). Les registres du port parallèles sont au nombre de 3 :

Registre de données en ADR est :

7	6	5	4	3	2	1	0
D7 = MSB	D6	D5	D4	D3	D2	D1	D0 = LSB

Il n'est accessible qu'en écriture dans les premiers modes SPP.

Registre d'état en ADR+1 est :

7	6	5	4	3	2	1	0
BUSY	/ACK	PE	SELECT	/ERROR	Non défini	Non défini	Non défini

Il n'est accessible qu'en lecture.

Registre de contrôle en ADR+2 est :

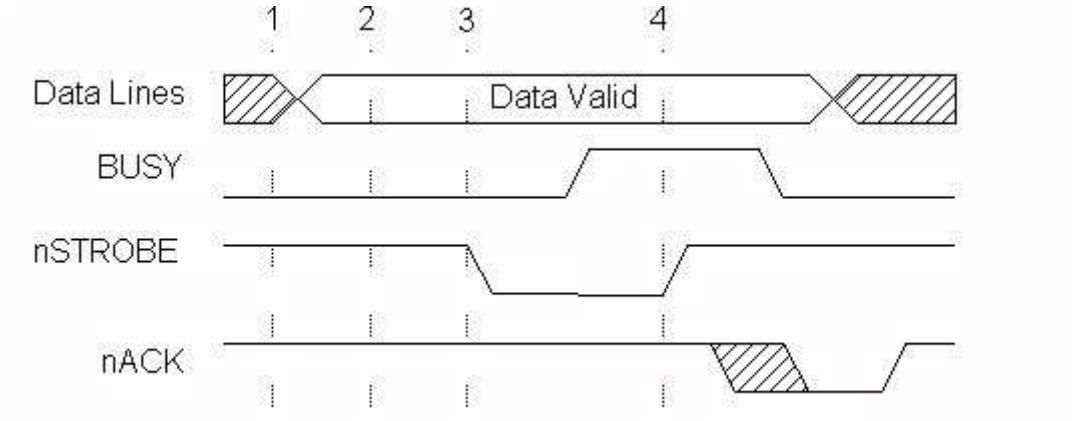
7	6	5	4	3	2	1	0
Non défini	Non défini	DIR	IRQ ENABLE	SELECT IN	/INIT	AUTOFEED	/STROBE

Il est accessible en lecture et en écriture.
Le bit 4 autorise ou non une interruption lorsque ACK passe à 0 (l'imprimante valide la Réception d'un caractère).
Le bit 5 ne sert que pour les modes bidirectionnels : Ce bit à 1 rend le port accessible en écriture.

III) Mode normaux : SPP (standard parallel port)

① Mode standard (Mode Centronics ou Compatibility mode)

C'est le mode utilisé pour le fonctionnement normal du port SPP : Les données sont placées sur les lignes de données.



1. Ecrire les données dans le registre de données (=> Elles sont sur les fils de sortie)
2. Lire le registre d'état pour vérifier que l'imprimante n'est pas occupée (BUSY)
3. Si elle ne l'est pas, écrire le registre de contrôle pour mettre /STROBE = 0.
4. Ecrire le registre de contrôle pour changer /STROBE et pouvoir passer à la donnée suivante.

② Mode Quartet (Nibble mode, 4 bits) pour ajouter des entrées de données

- ☞ bidirectionnel
- ☞ entrées de données = entrées de status.
- ☞ Un périphérique enverra un octet de données en envoyant 2 quartets (nibbles) à la suite.
- ☞ /ACK utilisé pour les interruptions => on utilise les 4 autres entrées restantes (BUSY, PE, SELECT, /ERROR).

Remarque: Les noms de broche changent (/Autofeed -> HostBusy, /SelectIn -> 1284 Active, /ACK -> PtrClk, ...)

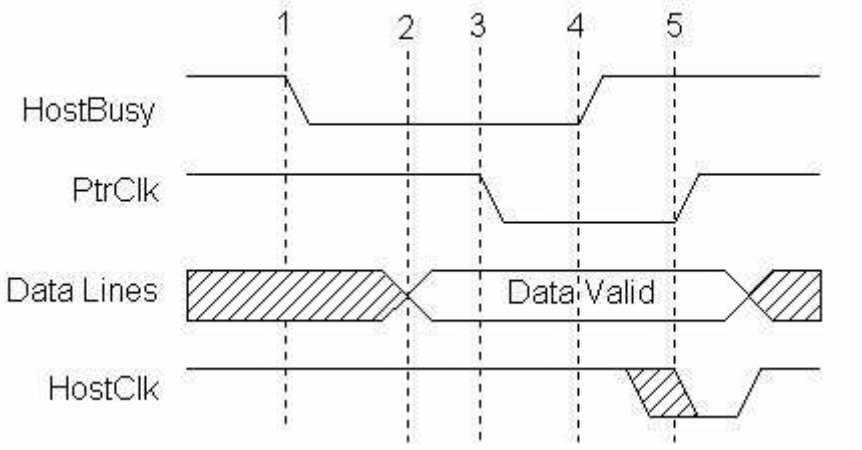
=> Liaison bidirectionnelle de données malgré un circuit électronique unidirectionnel pour ses lignes de données. Mais le débit est limité à 50 ko/s.

③ Mode Octet (Byte mode): Bidirectionnel 8 bits

On a un port parallèle bidirectionnel électroniquement. Le sens est commandé par le bit 5 DIR du registre de contrôle Mais la gestion reste la même qu'auparavant.

Signal SPP	Nom en mode Octet	In/Out	Description – Utilisation lors d'un transfert ***entrant*** en mode octet
/STROBE	HostClk	Out	Signal d'acquittement. Mis bas pour indiquer que l'octet a été reçu.
/AUTOFEED	HostBusy	Out	Signal de Handshake. Mis bas pour indiquer que le PC est prêt pour un octet. Mis haut pour indiquer que l'octet a été reçu.
/SELECTIN	1284Active	Out	Mis à l'état haut pour indiquer un transfert entrant (1284).
/INIT	/INIT	Out	Pas utilisé, laissé haut.
/ACK	PtrClk	In	Mis bas pour indiquer des données valides sur les lignes de données, Mis haut en réponse à HostBusy passant au niveau haut.
BUSY	PtrBusy	In	Signal Busy du PC.
PE	AckDataReq	In	Suit /DataAvail
SELECT	Xflag	In	Drapeau d'extensibilité. Pas utilisé en mode octet.
/ERROR	/DataAvail	In	Mis bas par le périphérique pour indiquer qu'une donnée entrante est disponible.
DATA[8:1]	DATA[8:1]	Bi-Di	Envoi des données du périphérique au PC.

Ex: Réception



1. Le PC signale qu'il est prêt à recevoir en mettant HostBusy bas
2. Le périphérique répond en plaçant les données sur les lignes de données
3. Le périphérique indique que l'octet est valide en passant PtrClk à bas
4. Le PC met HostBusy haut pour indiquer qu'il a reçu et qu'il n'est pas encore prêt pour un suivant
5. Le périphérique met PtrClk haut pour acquitter cette information.
   Le PC applique une impulsion HostClk pour acquitter à son tour. Le périphérique peut changer les données.

On recommence les étapes 1 à 5 pour un second octet.

On a encore beaucoup de contrôles via logiciel=> gestion lourde et pénible =>Débit encore limité.

=> En 1991, mode étendu EPP => Autre type de transfert
=> Le circuit gère les échanges (génère automatiquement les signaux de contrôle)
Il est contrôleur de l'échange parallèle ce qui décharge le µP.

IV) Modes étendus

① Mode EPP (Enhanced Parallel Port)

Le Handshake est géré par le contrôleur, pas par le logiciel (chip d'I/O 82360).

Le protocole EPP fournit 4 types de cycles de transfert de données :
- Ecriture/Lecture de données (Data_Write cycle, Data_Read cycle)
- Ecriture/Lecture d'adresse (Address_Write cycle, Address_Read cycle)

On ajoute des registres en ADR+3 et ADR+4 pour les transferts EPP.

En générant une simple instruction d'écriture en ADR+4, le contrôleur EPP génère automatiquement les signaux de Handshake et de Strobe nécessaires pour que le transfert suive le cycle EPP Data_Write.

En revanche l'utilisation des registres SPP (en ADR, ADR+1, ADR+2) amènent un comportement compatible SPP.

=> On arrive à 2Mo/s (proche du fonctionnement d'une carte interne ISA = 6 Mo/s). Avec les handshakes imbriqués, la vitesse sera fixée par le terminal le moins rapide.

Remarque : Sur les PC actuels on a un contrôleur AIP (Advanced Integrated Peripherical = Intel) qui intègre contrôleur de port série, port parallèle, de disquette,…

② Mode ECP (Extended Capability Port) = "The IEEE 1284 Extended Capabilities Port Protocol and ISA Interface Standard"

ECP proposé par HP et Microsoft permet en plus des performances du mode EPP de définir un protocole d'échange d'informations de gestion du matériel.

=> Autres cycles d'échanges
=> Plug and Play
=> Gestion FIFO
=> Compression temps réel RLE (taux de 64 pour image vers imprimante ou scanner).,
=> Utilisation de transfert DMA (Direct Memory Access).
=> Adressage de canal (plusieurs appareils logiques sur une seule liaison physique = fax/modem/imprimante ou scanner/imprimante).

Il y a 2 types de cycles (bidirectionnels): - Cycle de données
- Cycle de commandes -> Comptage RLE (Run Lenght Encoding).
-> Adressage de canal.

FIFO: Découpler les transferts internes au PC (bus ISA, DMA,…) avec les transferts sur la liaison parallèle.

Modification de transfert = Dialogue logiciel => ECP est fait pour gros transferts:
Contrairement à EPP où l'on peut mixer entrées et sorties sans dialogue ou ajout de données, en ECP, un changement de direction doit être négocié. C'est une complication du logiciel de gestion surtout si on interrompt une liaison DMA.

On ajoute des registres en ADR+400 h, ADR+ 401 h et ADR + 402 h.

Le mode ECP est défini par le BIOS. Puis un mode de fonctionnement est défini dans le registre ECR (Extended Control Register). Il est à ADR+ 402 h

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Mode (selon tableau précédent)			Interruption ECP	Autorisation mode DMA	Bit de service ECP	FIFO pleine	FIFO Vide

Mode = 000 pour fonctionnement SPP, 001 pour mode octet, 011 pour ECP, 110 pour mode test,...

Bus USB (Universal Serial Bus)

Développement de l'informatique grand public => besoin de plus de liaisons,
=> besoin de plug and play,
=> besoin de multimédia = acheminer différents flux

USB grâce aux progrès de l'électronique répond à ces besoins avec des contrôleurs plus performants. Il utilise une liaison série semi-duplex mais haut débit et suivant un protocole très évolué.
=> La complexité diminue pour l'usager avec le plug and play, donc elle augmente pour le protocole.

On distingue les versions USB1.0 : débit maxi de 1,5 Mbit/s (Low Speed)
USB1.1 : Débit maxi de 12 Mbit/s (Full Speed)
USB2.0 : Débit maxi 480 Mbit/s (High Speed)
USB3.0 : Débit maxi 5 Gbit/s (Super Speed)

I) Présentation

Un système de communication USB est constitué des éléments suivants :

Un hôte	Il n'y a qu'un seul hôte qui doit gérer les échanges de tout le bus. Il arbitre les accès au bus.

Un hub	Comme les hubs de réseau, il fournit des points de connexion, la topologie logique étant en étoile. Tout appareil USB est connecté directement à l'hôte sur le plan logique même si on a plusieurs hubs en cascade (un hub est transparent sur le plan logique). Les principales fonctions du hub hormis la connexion physique sont : - la responsabilité de détecter l'attachement ou le détachement d'un appareil. - la gestion de l'alimentation des appareils auto-alimentés par le bus USB. - la détection d'erreur sur le bus et la correction. - la détection de la vitesse d'un appareil

Un périphérique USB	Tout ce qui n'est pas hôte est un périphérique USB (hub compris). On distingue les périphériques autoalimentés ou non par le bus USB

Remarque : L'objectif étant la simplicité d'utilisation, on a 2 types de prises permettant de ne pas se tromper dans le chaînage des appareils.

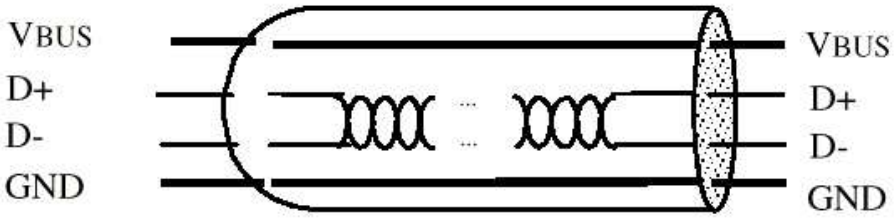
Dans tout périphérique USB, on trouve le SIE (Serial Interface Engine) :
- Sérialisation et désérialisation des données : Les données sont transmises en série.
- Codage NRZI des données et l'insertion de 0 sans signification pour faire varier le signal électrique (voir plus loin).
- Ajout de CRC (contrôle d'erreur) pour les données sortantes et vérification du CRC reçu en réception
- Il détecte enfin l'identité d'un paquet et l'état du bus.

Dans l'hôte USB, en plus du SIE on a aussi le HC (Host Controller). Ce contrôleur est l'élément important qui initie les échanges, contrôle l'accès au bus, contrôle que tout se déroule selon le protocole USB.

Les principales fonctions du HC sont :
- Génération de trame : Le HC découpe la transmission en intervalles de 1 ms : Trame
  Il envoie régulièrement un paquet SOF (start of frame)
- Gestion d'échange de données : Le HC gère les requêtes de transfert de données dans les 2 sens.
- Gestion du protocole USB : Le HC enchaîne les échanges selon le protocole USB.
- Contrôle d'erreur : Le HC gère les erreurs telles que : + Non réponse par un timeout
  + Bit erroné par système de CRC
  + Entête erronée avant des données
- Reprise à distance : Le HC peut mettre en veille un périphérique, détecter une demande de sortie de cet état par le périphérique, le sortir de cet état.

II) Description des échanges physiques USB

Le câble USB est fait de 4 fils : - 2 fils pour d'alimentation Vbus (+5V) et GND (0V)
- 2 fils pour les données : Paire différentielle D_+ et D_- .



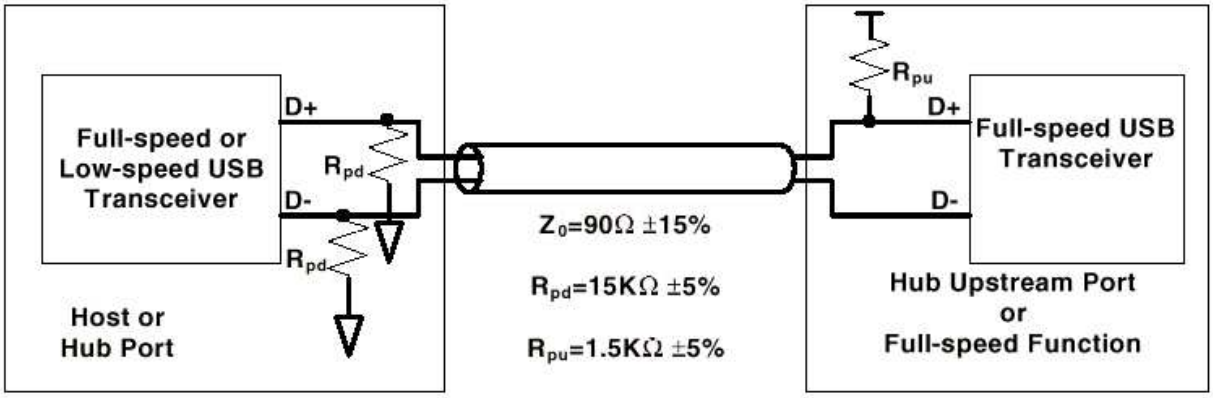
La transmission est bidirectionnelle mais en semi-duplex sur le plan électronique. On verra les types de transferts qui permettent de s'approcher d'une liaison duplex grâce à la vitesse des échanges.

Le niveau physique définit les **données** suivantes : 0 différentiel : D_+ bas, D_- haut
1 différentiel : D_+ haut, D_- bas
SEO (Single ended zero) : D_+ bas, D_- bas

On définit différents **états du bus** :

état J : "1" différentiel pour les appareils haut débit, "0" différentiel pour les appareils bas débits
état K : "0" différentiel pour les appareils haut débit, "1" différentiel pour les appareils bas débits
état de veille (idle state) : Electriquement c'est l'état J mais dans le déroulement des opérations ce n'est pas la même chose car l'état J est imposé par une source alors qu'en état de veille la source est inactive.

Les résistances de terminaison de bus le place en état de veille (Idle) au repos. Cela permet aussi en début de connexion d'identifier le débit (résistance de tirage à +5V sur D_+ pour le haut débit , D_- pour le bas débit).



Pour un appareil USB2, il commence un dialogue en USB1 full speed et dialogue avec le contrôleur pour passer en USB2.

On rencontre d'autres utilisations liées aux niveaux électriques que la transmission d'informations :

Reset	L'hôte peut réinitialiser un périphérique : Il envoie SEO plus de 2.5µs (normalement 10 ms !).
Suspension	L'hôte peut mettre le périphérique dans un mode veille (pas de trafic). Il passe en veille quand il reconnaîtra l'état veille (idle state) sur le bus plus de 3 ms = La source n'impose plus de tension. Le périphérique reste en écoute du bus.
Reprise	Un périphérique en veille reprend le trafic en reconnaissant l'état K sur le bus (l'état de veille étant électriquement identique à l'état J). L'hôte envoie cet état K pendant au moins 20 ms. Un périphérique peut parfois se réactiver lui-même au bout d'un certain temps.
SOP (Start of packet)	Pour indiquer le début d'un paquet Au passage de l'état de veille (identique électriquement à J) à l'état K
EOP (end of packet)	Pour indiquer la fin d'un paquet EOP est fait de SE0 sur 2 bits puis de l'état J sur 1 bit.

- Le LSB est transmis en premier dans les échanges.
- Le codage est NRZI : 1 = mémoire, 0 basculement (1 ici = info à envoyer = 1 logique à ne pas confondre avec 1 électrique).

Remarque : Une longue suite de 1 donne un niveau électrique constant ce qui est mauvais pour la récupération d'horloge du récepteur => USB utilise l'insertion de bit (bit staffing) : Après 6 bits à 1 successifs, il y a insertion d'un 0. Le récepteur saura qu'il faut supprimer un 0 après 6 bits à 1, que ce n'est pas de l'information.

III) Le protocole USB

① Déroulement d'un échange USB

L'hôte USB a en charge la plupart de la complexité du protocole => Conception périphérique simple et peu coûteuse.

Les données sont échangées dans les 2 sens sous forme de paquets. Un échange se déroulera selon 3 phases :

Phase du jeton (Token phase) : L'hôte « lance » le jeton en indiquant le type de la transaction à venir.

Phase de transfert de données : Les données sont émises sous forme de paquets. La direction des données suit celle indiquée par le jeton transmis auparavant.

Phase de Handshake : Un paquet de Handshake est envoyé indiquant le succès ou l'échec de l'échange.

USB est un protocole à scrutation (polling) :

L'hôte veut recevoir des données d'un périphérique => il envoie un jeton avec l'adresse de celui-ci. Si le périphérique a des données à envoyer, il les envoie après réception du jeton. S'il n'a rien, l'hôte envoie un jeton adressé au périphérique suivant.

Si l'hôte veut envoyer des données vers un périphérique, il envoie un jeton avec l'adresse de celui-ci puis les données.

Une **transaction** a eu lieu chaque fois que l'on réalise le cycle Jeton-Données-Handshake.

Le protocole inclut des systèmes de Handshake, des chiens de garde, des systèmes de correction d'erreurs afin d'avoir un système robuste.

Le branchement à chaud d'un périphérique modifie la valeur électrique sur le bus ce qui permet à l'hôte de détecter cette présence. Ensuite l'hôte pourra s'informer et configurer le périphérique (pas besoin de redémarrer le PC pour détecter le périphérique).

② Différents transferts possibles

USB traite des flux différents => Qualité de service. On définit différents types d'échanges sur le bus:

	Types de transfert			
	Contrôle	**Bulk**	**Interruption**	**Isochrone**
Utilisable en bas débit	Oui	Non	Oui	Non
Intègre une correction d'erreur	Oui	Oui	Oui	Non
Garantit le débit	Non	Non	Non	Oui
Garantit le non dépassement d'une latence maximale(délai de transmission)	Non	Non	Oui	Oui
Application typique	Configuration d'un appareil	Imprimante	Souris	Son ou image

Les transferts se faisant en 3 phases (jeton, données, handshake), la différence entre transferts se fera donc sur la gestion de la communication (périodicité, protection contre les erreurs,…).

③ Différents paquets échangés

a) Champs intervenants dans USB

Champ SYNC :	Début de chaque paquet = Etat de veille (Idle équivalent électrique à l'état J) suivi de "KJKJKJKK" = en NRZI à 00000001. Ainsi le récepteur se synchronise.

Champ PID :	Type du paquet. (8 bits = 4 premiers pour l'identité du paquet puis le complément des 4 premiers envoyés par sécurité). - Paquet jeton (token) : JETON OUT : transfert hôte vers périphérique à venir. JETON IN : transfert périphérique à l'hôte à venir. JETON SETUP : Paquet à venir de l'hôte au périphérique pour la configuration. - Paquet SOF : Le PID indique un paquet SOF (Start Of Frame). - Paquet de Données : Le PID de données peut être DATA0 ou DATA1. => Cela ajoute un test de synchronisation des transferts : Les data doivent alterner entre DATA0 et DATA1. - Paquet de Handshake : Un paquet handshake indique le succès ou l'échec d'une procédure. Le PID indiquera si le paquet Handshake est ACK, NAK ou STALL : ACK : Acquittement = tout s'est bien déroulé. NAK : Le récepteur ne peut plus recevoir de données ou l'émetteur ne peut Plus en émettre => Contrôle de flux. STALL : La commande de SETUP n'est pas comprise.

Champ Adresse :	Il est divisé en 2 parties : - Adresse (ADDR) : 7 bits pour définir le périphérique. L'adresse 0 est réservée comme une voie de broadcast. - Endpoint (ENDP): 4 bits pour définir des fonctions internes différentes à un même périphérique (scanner et imprimante par exemple). Le endpoint 0 est réservée pour le broadcast (va avec adresse 0).

Champ Numéro de trame :	Il indique sur 11 bits le numéro de trame dans les paquets SOF (start of frame). La trame est l'ensemble des transmissions sur 1ms.

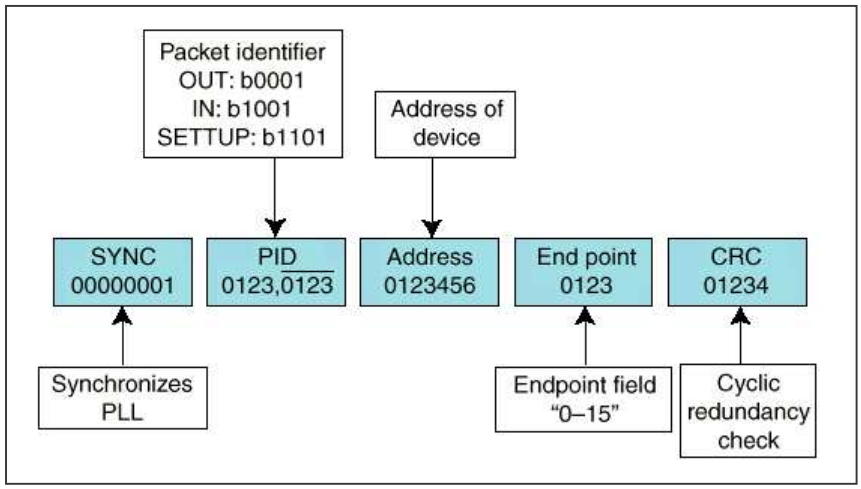
Champ Donnée :	Il contient les données échangées dans les paquets de données. Il peut aller jusqu'à 1023 octets.

Champ CRC :	Détection des erreurs : cyclic redundancy check. Sa longueur varie selon le type de paquet.

b) Paquet Jeton (TOKEN)

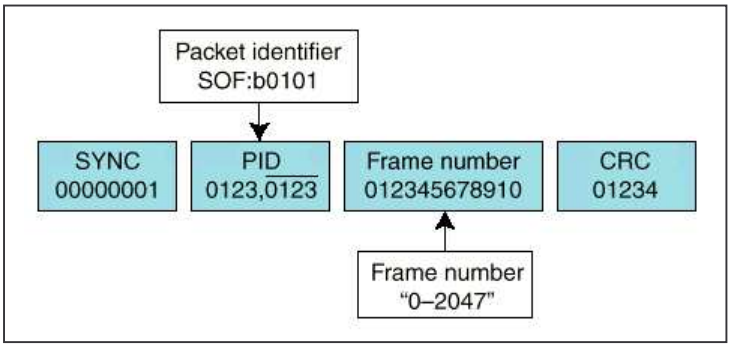
Paquet de 24 bits

Si le PID est 1000b => Jeton OUT
(MSB à gauche mais LSB émis en premier)
Si le PID est 1001b => Jeton IN
Si le PID est 1011b => Jeton SETUP



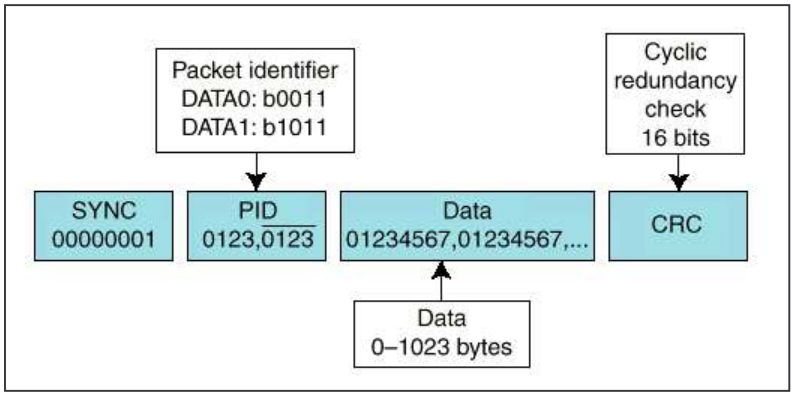
c) Paquet SOF (Start Of Frame)

Paquet de 24 bits
le PID est 1010b



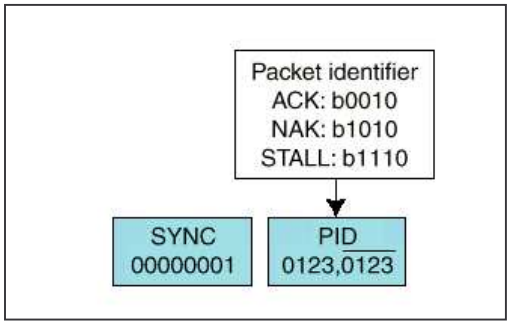
d) Paquet de donnée

Paquet de 24 bits plus les données (jusqu'à 1024 octets)
Si le PID est 1100 en binaire : DATA0
Si le PID est 1101 en binaire : DATA1



e) Paquet de Handshake

Paquet de 8 bits
Si le PID est 0100 en binaire : ACK : Succès de la procédure
Si le PID est 0101 en binaire : NACK : Echec de la procédure
Si le PID est 0111 en binaire : STALL : Commande non comprise



④ Transfert de contrôle

Ce transfert sert à la configuration. Elle est faite par un processus d'énumération. Quand un appareil se connecte, l'hôte a besoin d'apprendre ses caractéristiques et de le configurer.

La transaction dans un transfert de contrôle se déroule selon les phases :

1) Phase de SETUP	a) Phase de Jeton b) Phase de données c) Phase de Handshake
2) Phase de données (optionnelle)	a) Phase de Jeton b) Phase de données c) Phase de Handshake
3) Phase d'état (status)	a) Phase de Jeton b) Phase de données c) Phase de Handshake

La phase de SETUP sert à indiquer au périphérique quelle commande l'hôte veut envoyer :

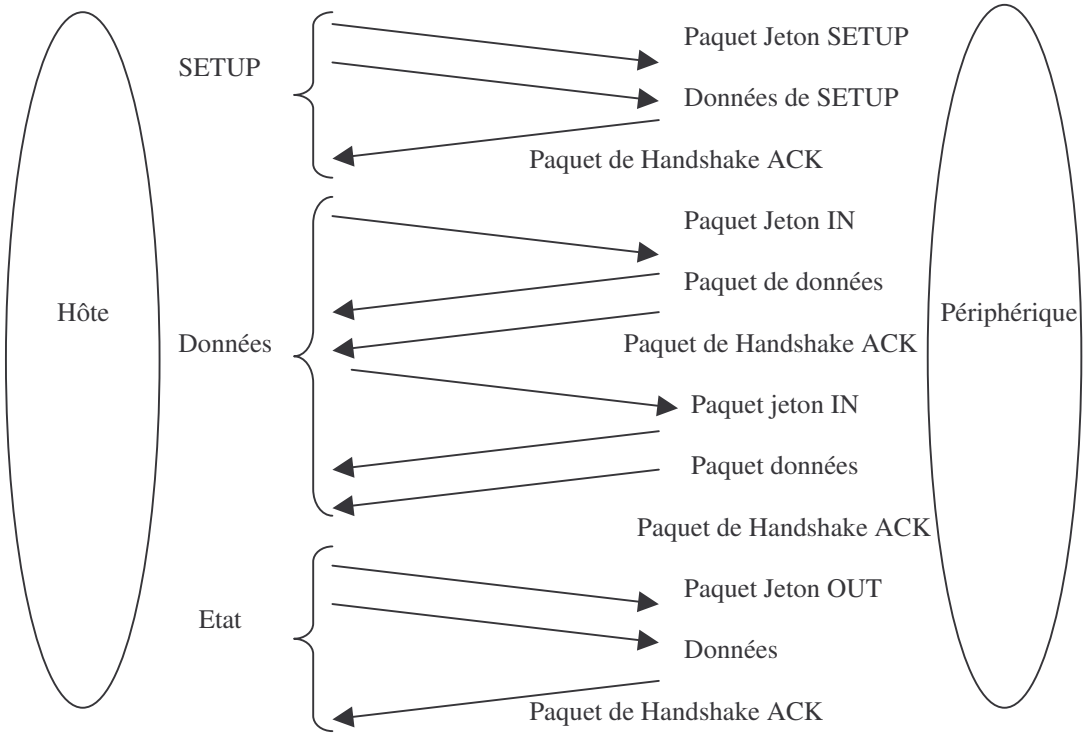
- Commande SET_ADRESS : Affectation d'une adresse
- Commande GET_DEVICE_DESCRIPTOR : L'hôte veut les caractéristiques du périphérique.
- Commande GET_CONFIGURATION_DESCRIPTOR : L'hôte souhaite en savoir plus sur un point particulier.
- Commande GET_CONFIGURATION : L'hôte détecte la configuration active à ce moment.
- Commande SET_CONFIGURATION : L'hôte impose une configuration.

=> L'hôte envoie un paquet jeton SETUP, suivi par le paquet décrivant la commande, le périphérique doit répondre par un paquet de Handshake ACK.

La phase de données sert à échanger des données. On peut avoir plusieurs transactions consécutives, soit entrantes, soit sortantes, mais toutes de même direction.

=> L'Hôte répète les transactions : Envoi d'un paquet jeton IN ou OUT, échange des données, d'un paquet de Handshake pour terminer la transaction.

La phase d'état consiste en un rapport du périphérique à l'hôte concernant les 2 phases précédentes.



⑤ Transfert Bulk

Transfert Bulk = transfert massif de données.
=> La bande passante allouée (par le temps alloué) dépend de l'occupation du bus.
=> Le transfert doit être fiable => Utilisation d'acquittement.

La transaction pour un tel type de transfert se déroule selon les 3 phases Jeton-Données-Handshake.

En cas d'erreur détectée, il n'y a pas d'acquittement par l'émission d'un paquet de Handshake.

Si un périphérique ne peut répondre à une demande de transfert de l'hôte par un jeton IN, au lieu du paquet de donnée il envoie un paquet de Handshake NAK (retransmission demandée) ou STALL (abandon).

⑥ Transfert d'interruption

Le transfert d'interruption = faible délai d'attente (souris, joystick) ET faible volume de données.
=> Un périphérique dans ce mode définira à la config l'intervalle de temps entre l'envoi de 2 informations.
L'hôte devra contacter le périphérique périodiquement selon cet intervalle de temps.

Le déroulement est le même que pour le transfert Bulk.

⑦ Transfert isochrone

Le transfert isochrone = échanges temps réel continu (audio, la vidéo).
=> Garantie de bande passante pour ce transfert qui est prioritaire = L'hôte garantit un accès périodique.

En revanche la fiabilité est moins contrôlée que pour les autres transferts donc on se passe d'acquittement.
=> La transaction en transfert isochrone = répétition de transactions à 2 phases Jeton-Données (pas de Handshake).

L'hôte envoie un paquet jeton IN pour recevoir des données ou OUT pour en émettre. Puis les données sont échangées.