L'USB en bref

Donner un sens au standard USB

Les protocoles USB

Contrairement à la RS232 et des interfaces sérielles similaires où le format des données envoyées n'est pas défini, l'USB est composé de plusieurs couches de protocoles. Bien que cela semble compliqué, n'abandonnez pas pour l'instant. Une fois que vous aurez compris ce qui se passe, vous devrez uniquement vous inquiéter des couches supérieures. En fait la plupart des Circuits Intégrés (C.Int.) contrôleur d'USB s'occuperont de la couche inférieure, la rendant ainsi presque invisible au regard du concepteur final.

Chaque transaction USB consiste d'un

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

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

Les champs de paquet USB ordinaires

Les données sur le BUS USB sont transmises avec le bit LSB en premier. Les paquets USB se composent des champs suivants,

- **Sync**

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

- **PID**

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

Groupe	Valeur PID	Identificateur Paquet
Token **Jeton**	0001	OUT Token
	1001	IN Token
	0101	SOF Token
	1101	SETUP Token
Data **Données**	0011	DATA0
	1011	DATA1
	0111	DATA2
	1111	MDATA
Handshake **Poignée de Mains**	0010	ACK Handshake
	1010	NAK Handshake
	1110	STALL Handshake
	0110	NYET (No Response Yet)
Special	1100	PREamble
	1100	ERR
	1000	Split
	0100	Ping

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

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

PID_0	PID_1	PID_2	PID_3	$nPID_0$	$nPID_1$	$nPID_2$	$nPID_3$

- **ADDR**

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

- **ENDP**

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

- **CRC**

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

- **EOP**

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

Les types de paquet USB

L'USB a quatre types différents de paquet. Les paquets jetons indiquent le type de la transaction qui va suivre, les paquets de données contiennent la charge utile, les paquets " poignée de mains " sont utilisés pour valider les données ou rapporter les erreurs et les paquets début de trame (SOF) indiquent le commencement d'une nouvelle trame.

- **Les paquets jetons.**

  Il y a 3 sortes de paquets Jetons,

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

Les paquets jetons doivent se conformer au format suivant,

Sync	PID	ADDR	ENDP	CRC5	EOP

- **Les paquets de données**

  Il y a 2 sortes de paquets de données, chacun étant capable de transmettre plus de 1024 octets de données.

  - Data0
  - Data1

Le mode haute vitesse définit 2 autres PIDs de données, DATA2 et MDATA. Les paquets de données ont le format suivant :

Sync	PID	Data	CRC16	EOP

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

- **Les paquets " poignée de mains "**

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

  - **ACK** - validant que le paquet a été reçu correctement.
  - **NAK** - rapporte que temporairement l'appareil ne peut ni envoyer ou recevoir des données. Aussi utilisé pendant les transactions d'interruptions pour avertir l'hôte qu'il n'a pas de données à envoyer.
  - **STALL** (Bloqué) - L'appareil se retrouve dans un état qui va exiger l'intervention de l'hôte.

Les paquets " poignée de mains " ont le format suivant,

Sync	PID	EOP

- **Les paquets début de trame (SOF).**

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

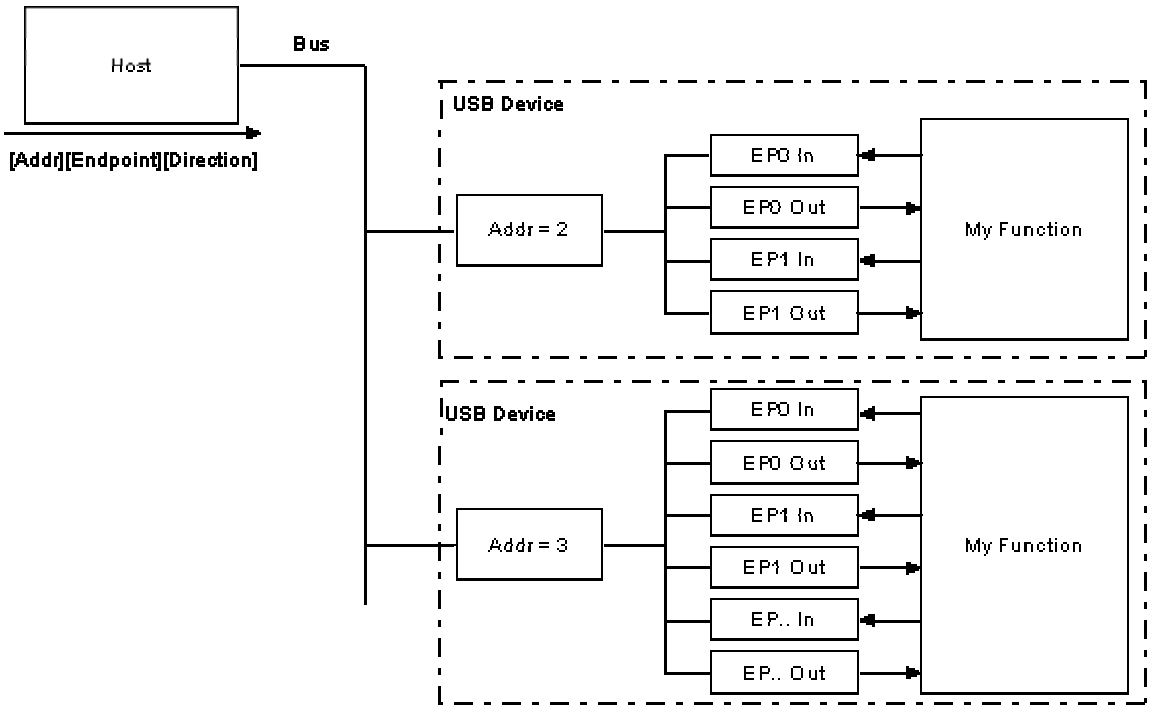
Sync	PID	Frame Number	CRC5	EOP

Les fonctions USB

Quand nous pensons à un appareil USB, nous pensons à un périphérique USB, mais un appareil USB peut signifier un appareil émetteur / récepteur USB sur l'hôte ou périphérique, un HUB USB ou un circuit intégré contrôleur d'hôte ou un appareil périphérique USB. Par conséquent le standard fait références aux fonctions USB qui peuvent être considérées comme appareil USB qui fournissent une possibilité ou une fonction comme une imprimante, in lecteur Zip, un scanner, un modem ou un autre périphérique.

Donc à l'heure qu'il est nous devrions savoir de quoi est composé un paquet USB. Ce n'est pas le cas ? Vous avez déjà oublié combien de bits composent un champ PID ? Bon, ne vous alarmez pas trop. Heureusement la plupart des fonctions USB manipulent les protocoles USB bas niveau jusqu'à la couche transaction (que nous traiterons au chapitre prochain) dans le silicium. La raison pour laquelle nous traiterons cette information est que la plupart des contrôleurs de fonction USB signaleront des erreurs comme l'Erreur d'Encodage PID. Sans traiter rapidement ceci, l'on pourrait se demander ce qu'est l'erreur d'encodage PID ?

Si vous avez proposé que les 4 derniers bits du PID ne correspondent pas au complément des 4 premiers bits alors vous auriez raison.



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

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

Les terminaisons

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

Tous les appareils doivent prendre en charge la terminaison zéro. C'est la terminaison qui reçoit la totalité de la commande des appareils et des demandes d'états pendant l'énumération et tant que l'appareil est opérationnel sur le bus.

Canaux de communications (Pipes)

Tandis que l'appareil envoie et reçoit des données sur une succession de terminaisons, le logiciel client transfère des données à travers des canaux de communications. Un canal de communication (Pipe) est une connexion logique entre l'hôte et les terminaisons. Les canaux de communications auront aussi un ensemble de paramètres qui leur seront associés tels que : combien de bande passante leur est allouée, quel type de transfert (Commande, Bloc, Iso ou Interruption) ils utilisent, la direction du flux de données et les tailles maximales du paquet / tampons. Par exemple le canal de communication par défaut est un canal bidirectionnel composé d'une terminaison zéro IN et d'une terminaison zéro OUT avec un type de transfert de commande.

L'USB définit 2 types de canaux de communications

- **Les flux de données (Stream Pipes)** n'ont pas de format USB défini. Cela veut dire que vous pouvez envoyer n'importe quelle sorte de données par un flux de données et que vous pouvez rapporter les données de sorties à l'autre extrémité. Les données circulent séquentiellement et ont une direction prédéfinie, soit en entrée soit en sortie. Les flux de données supporteront les types de transferts en Bloc, Isochrone et d'Interruptions. Les flux de données peuvent soit être contrôlés par l'hôte ou l'appareil.
- **Les canaux de messages (Message Pipes)** ont un format USB défini. Ils sont contrôlés par l'hôte et sont initiés par une demande émanant de l'hôte. Les données sont ensuite transférées dans la direction voulue, dictées par la demande. Par conséquent les canaux de messages permettent aux données de circuler dans les deux directions mais ne prendront seulement en charge que les transferts de commande.