

CHAPITRE1

LES PROTOCOLES USB

1.1 INTRODUCTION :

Contrairement à la RS232 et des interfaces sérieelles similaires où le format des données envoyées n'est pas défini, l'USB est composé de plusieurs couches de protocoles.

En fait la plupart des circuits intégrés contrôleur d'USB s'occupent 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).

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 (handShaking), signalant que les données ou le jeton ont été reçus correctement ou encore la terminaison est bloquée, ou n'est pas disponible pour accepter de données.

1.2 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 où 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.

PID0	PID1	PID2	PID3	nPID0	nPID1	nPID2	nPID3
------	------	------	------	-------	-------	-------	-------

- **ADDR :**

Le champ adresse détermine à quel appareil le paquet est destiné. Sa longueur est de 7 bits, permettant de supporter 127 appareils. L'adresse 0 n'est pas valide, Un appareil qu'on ne lui a pas encore attribué d'adresse, doit répondre aux paquets envoyés à l'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 est suivie par un " J " d'une durée de 1 bit.

1.3 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 demain " 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 :

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

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.

- o Data0
- o 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
------	-----	------	-------	-----

- o La taille maximale de données " charge utile " pour les appareils basse vitesse est de 8 octets.
- o La taille maximale de données " charge utile " pour les appareils pleine vitesse est de 1023 octets.
- o La taille maximale de données " charge utile " pour les appareils haute vitesse est de 1024 octets.

- Les paquets " poignée de mains " :
Il y a 3 sortes de paquets " poignée de mains " qui font simplement partie du PID :
 - o ACK - validant que le paquet a été reçu correctement.
 - o 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.
 - o 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 ± 500 ns sur un bus pleine vitesse ou bien toutes les $125\mu s \pm 0,0625\mu s$ sur un bus haute vitesse.

Sync	PID	Frame Number	CRC5	EOP
------	-----	--------------	------	-----

1.4 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 communication sur la fonction USB.

Au niveau de la couche logicielle, le pilote (driver) logiciel envoie, par exemple, un paquet EP1. A la sortie de l'hôte, la donnée aboutit au tampon EP1 OUT. Le microprogramme pourra alors lire 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 qui s'exécute sur ce même appareil.

1.5 Les types de terminaisons :

La spécification du Bus Série Universel définit 4 types de transferts ou de terminaisons :

- Transferts de commande
- Transferts d'interruption
- Transferts Isochrone
- Transferts en Bloc (BULK)

1.5.1 Transferts en Bloc(BULK)

Les Transferts en Bloc peuvent être utilisés pour de grandes quantités de données sporadiques.

De tels exemples pourraient inclure un travail d'impression envoyé à une imprimante ou une image provenant d'un scanner. Les Transferts en Bloc se prémunissent de correction d'erreurs sous la forme d'un champ CRC16 sur les données " charge utile " et sur les mécanismes de détection et de retransmission d'erreurs qui assure la transmission et la réception de données de manière infallible.

Les Transferts en Bloc utiliseront une bande passante de réserve non attribuée sur le bus après que toutes les autres transactions aient été allouées. Si le bus est bloqué par une interruption, les blocs de données peuvent alors s'écouler doucement sur le Bus. En conséquence, les transferts en bloc devraient seulement être utilisés pour des communications insensibles au temps du fait du non garanti du temps d'attente.

a) Les Transferts en Bloc :

- Utilisés pour de grandes quantités de données sporadiques.
- Détection d'erreurs via le CRC, avec la garantie de livraison.
- Pas de garantie de bande passante ou du temps d'attente minimum.
- Des flux de données - Unidirectionnel.
- Seulement des modes pleine et haute vitesse.

Les Transferts en Bloc sont seulement supportés par des appareils pleine et haute vitesse. Pour des terminaisons pleine vitesse, la longueur maximale du paquet en Bloc est soit 8, 16, 32 ou 64 octets.

Pour des terminaisons haute vitesse, la longueur maximale du paquet peut aller jusqu'à 512 octets. Si la charge utile des données est inférieure à la taille maximale du paquet, elle n'a pas besoin d'être remplie avec des zéros.

Un transfert en Bloc est considéré comme complet quand il a transféré la quantité exacte de données demandées, ou un paquet inférieur à la taille maximale de la terminaison, ou un paquet de longueur zéro.

IN: Quand l'hôte est prêt à recevoir des données en Bloc, il émet un jeton IN. Si la fonction reçoit le jeton IN avec une erreur, il ignore le paquet.

Si le jeton est reçu correctement, la fonction peut soit répondre avec un paquet DATA contenant les données en Bloc à envoyer ou bien un paquet Stall signalant que la terminaison a eu une erreur ou un paquet NACK signalant à l'hôte que la terminaison est opérationnelle, mais elle n'a pas de données à envoyer provisoirement.

OUT: Quand l'hôte veut envoyer à la fonction un paquet de données en Bloc, il émet un jeton OUT suivi par un paquet DATA contenant les données en Bloc. Si une partie du jeton OUT ou du paquet DATA est altérée, alors la fonction ignore le paquet.

Si le tampon de terminaison de la fonction est vide et qu'il a cadencé les données dans le tampon de terminaison, il émet un ACK prévenant l'hôte qu'il a reçu correctement les données.

Si le tampon de terminaison n'est pas vide à cause du traitement d'un paquet précédent, alors la fonction retourne un NAK. Toutefois si la terminaison a eu une erreur et que son bit d'arrêt a été positionné, elle retourne un Stall (Bloqué).

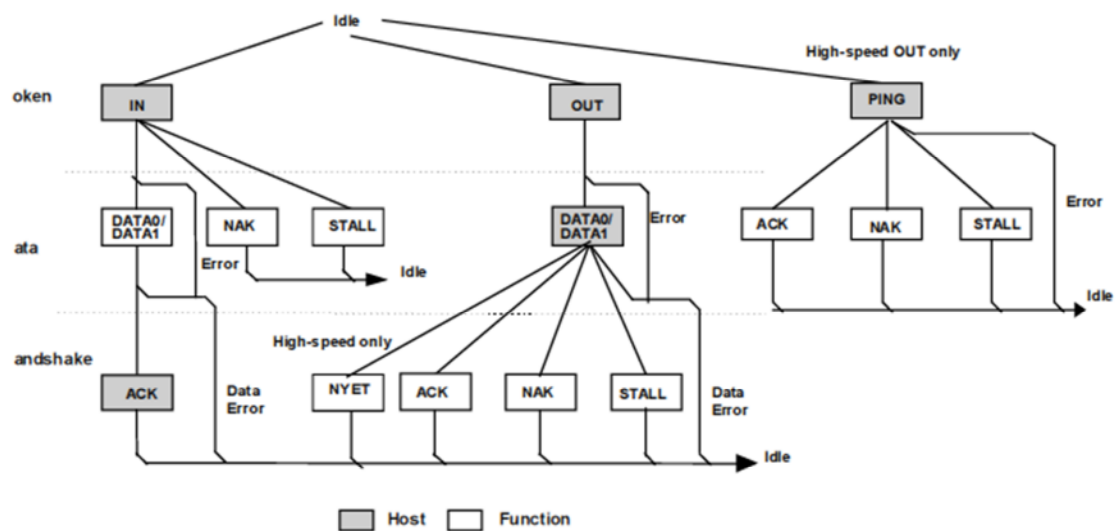


Figure 1.1 : le format d'une transaction en bloc IN et OUT et PING

1.5.2 Transferts de commande :

Les Transferts de commande sont régulièrement utilisés pour les opérations de commande et d'état. Ils sont essentiels pour installer un appareil USB avec toutes les fonctions d'énumération qui seront exécutés en utilisant les Transferts de commande. Ils surviennent généralement en paquets directs et par rafales qui sont initiés par l'hôte et utilisent le meilleur rendement de livraison.

La longueur du paquet du Transfert de commande pour appareil basse vitesse doit être de 8 octets, les appareils pleine vitesse autorisent une taille de paquet de 8, 16, 32 ou 64 octets et les appareils haute vitesse doivent avoir une taille de paquet de 64 octets.

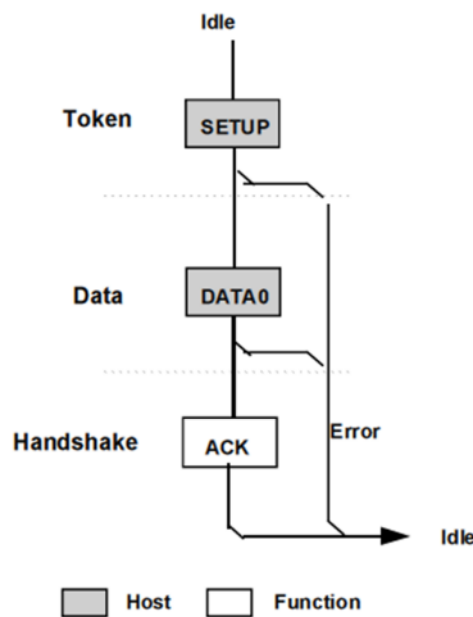


Figure 1.2 : le format d'une transaction de transferts de commande

1.5.3 Transferts d'Interruption :

Celui qui a une expérience sur les demandes d'interruption microcontrôleurs saura que les interruptions sont générées par l'appareil. Toutefois sous USB, si un appareil demande l'attention de l'hôte, il doit attendre que l'hôte l'interroge avant de signaler qu'il a besoin d'une attention urgente!

Les Transferts d'Interruption fournissent :

- Temps de retard (ou Latence) garanti.
- Ligne de flux – Unidirectionnel.
- Détection d'erreur et nouvel essai sur période suivante

Les transferts d'interruptions sont communément non périodiques, les petits appareils qui ont "initié" la communication ont besoin de temps de retard limité.

- La taille maximale de "charge utile" (payload) pour des appareils basse vitesse est de 8 octets.
- La taille maximale de "charge utile" (payload) pour des appareils pleine vitesse est de 64 octets.

- La taille maximale de "charge utile" (payload) pour des appareils haute vitesse est de 1024 octets.

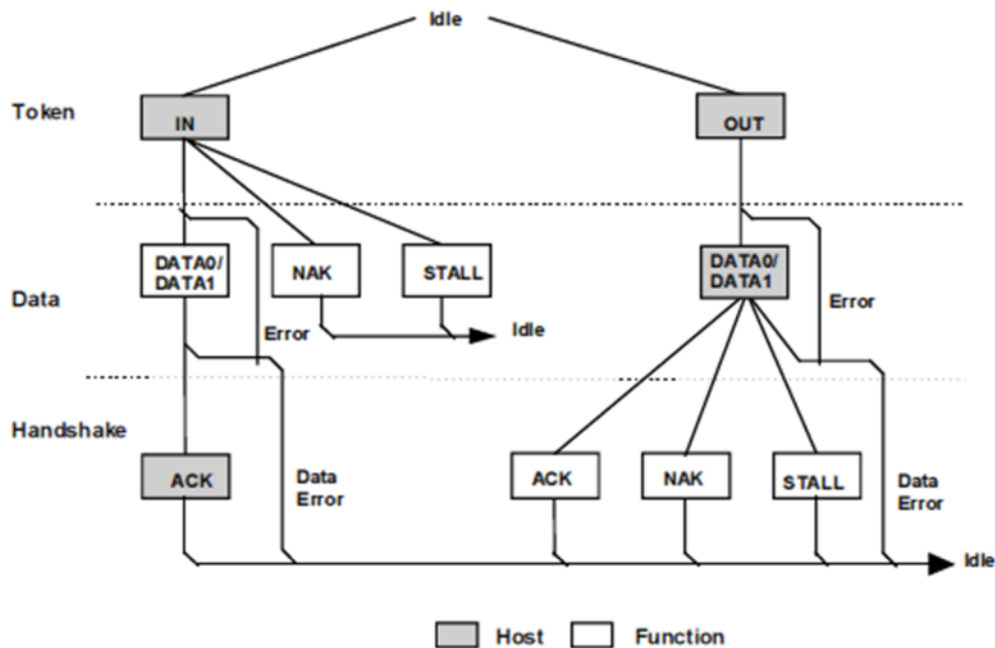


Figure 1.3 format d'une transaction d'interruption d'entrée IN et d'interruption de sortie OUT.

IN:

L'hôte interrogera périodiquement la terminaison d'interruption. Le taux d'interrogation est spécifié dans le descripteur de terminaison qui sera examiné plus tard. Chaque interrogation obligera l'hôte à envoyer un jeton IN.

Si le jeton IN est altéré, la fonction ignore le paquet et continue la surveillance du Bus pour de nouveaux jetons.

Si une interruption a été mise en attente par l'appareil, la fonction enverra un paquet Data contenant des données ayant rapport à l'interruption quand il recevra le jeton IN. Sur des réceptions au niveau de l'hôte, celui-ci retournera un ACK. Toutefois si les données sont altérées, l'hôte n'activera aucun état.

Si, d'autre part, une condition d'interruption n'était pas présente quand l'hôte interrogeait la terminaison d'interruption avec un jeton IN, alors la fonction signale cet état en envoyant un NAK. Si une erreur se produisait sur cette terminaison, un STALL (Bloqué) sera envoyé en réponse à un jeton IN.

OUT: Quand l'hôte veut envoyer à l'appareil les données d'interruptions, il émet un jeton OUT suivi par un paquet Data contenant les données d'interruption. Si une partie du jeton OUT ou du paquet Data est altéré alors la fonction ignore le paquet. Si le tampon de terminaison de la fonction est vide et qu'il ait cadencé les données dans le tampon de terminaison il émettrait un ACK indiquant à l'hôte qu'il a reçu correctement les données. Si le tampon de terminaison n'est pas vide à cause du traitement d'un paquet précédent, alors la fonction retourne un NAK.

Toutefois si une erreur se produisait à cause de la terminaison et que son bit d'arrêt (Halt) ait été positionné, elle renverrait un STALL (Bloqué).

1.5.4 Transferts Isochrones :

Les Transferts Isochrones se produisent continuellement et périodiquement. Ils contiennent généralement des informations à temps réel, tel des trains de données audio ou vidéo.

Les transferts Isochrones fournissent :

- Un accès garanti à la bande passante USB.
- Un temps d'attente limité.
- Des flux de données - Unidirectionnel.
- La détection d'erreur via le CRC, mais sans reprise ou garantie de livraison.
- Seulement des modes pleine et haute vitesse.
- Pas de données de basculement.

La taille maximale de données en " charge utile " (payload) est spécifiée dans le descripteur de terminaison d'une terminaison Isochrone. Cela peut être un maximum de 1023 octets pour un appareil pleine vitesse et 1024 octets pour un appareil haute vitesse. Comme la taille maximale de données en " charge utile " va effectuer des exigences de bande passante du Bus, il est prudent de spécifier une taille de " charge utile " modérée. Si on utilise des grandes " charges utiles " il peut être aussi plus intéressant de spécifier une série d'interfaces alternatives avec des tailles de charges utiles Isochrones variables. Si pendant l'énumération, l'hôte ne peut pas valider une terminaison Isochrone nommée à cause des restrictions de bande passante, il reste une solution de repli préférable à un échec complet. Les données qui ont été envoyées sur une terminaison Isochrone peuvent être inférieures à la taille pré-négociée et peuvent varier en longueur de transaction en transaction.

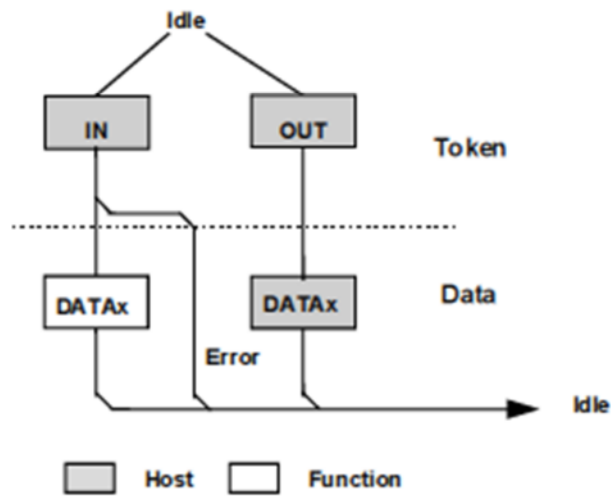


Figure 1.4 Le format d'une transaction Isochrone IN et OUT

1.6l'USB : l'utilisation des microcontrôleur PIC

Bus série universel (USB) fournit une interface commune qui simplifie énormément comment un utilisateur connecte plusieurs types de périphériques à un ordinateur individuel (le PC). Au-delà du PC, beaucoup de systèmes incorporés peuvent profiter de la norme USB et ce pour interagir avec une large variété de périphériques fonctionnant avec cette norme.

Contrairement à un PC, un hôte incorporé supporte seulement un jeu prédéfini de périphériques. La société Microchip fournit à ses produits dédiés à la norme USB un 'firmware' qui active et configure le module USB intégré dans ses microcontrôleurs.

L'utilisation de framework simplifie la mise en œuvre du module USB en mode hôte, il est plus facile de contrôler presque n'importe quel type de dispositif périphérique désiré.

1.6.1 Architecture d'application :

l'architecture d'application est en réalité un ensemble de plusieurs couches (la Figure 1.5), avec les différents composants de Microchip USB Hôte incorporé

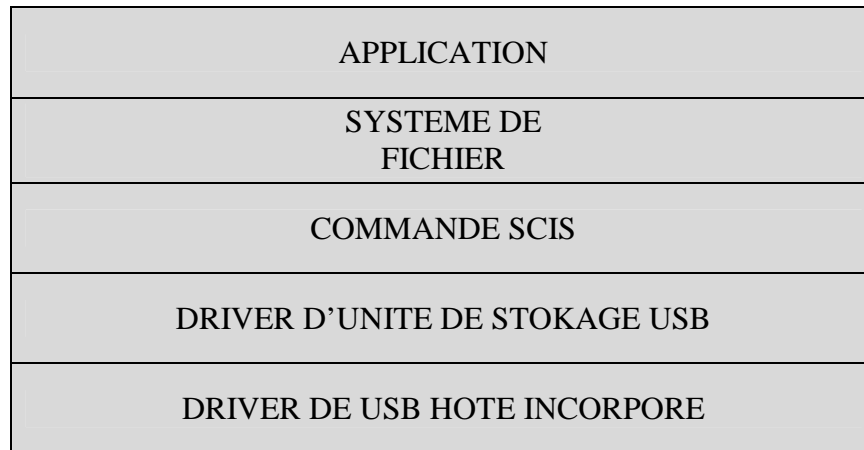


Figure 1.5 Architecture d'application

Tableau1.2 nous montre les source des fichier utilisés dans cette application

Driver de USB hôte incorporé :

Le driver de USB hôte incorporé fournit le support (l'assistance) générique pour des USB hôte incorporé . On fournit l'interface à cette couche automatiquement dans le driver d'unité de stockage USB.

- Driver d'unité de stockage USB :

La couche suivante fournit le Driver pour unité de stockage , qui est exigée pour interface avec des dispositifs de stockage massifs(comme un Disque flash USB).

Couche	Nom de fichier
DRIVER DE USB HOTE INCORPORE	usb_host.c
	usb_host.h
	USBCore.h
DRIVER D'UNITE DE STOKAGE USB	usb_host_msd.c
	usb_host_msd.h
COMMANDE SCIS	usb_host_msd_scsi.c
	usb_host_msd_scsi.h
SYSTEME DE FICHER	FSIO.c
	FSIO.h
	FSDefs.h

Tableau 1.2 source de fichiers

- Support de système de fichiers et commande de SCSI :

La bibliothèque de système de fichiers fournit la couche support (d'assistance) de système de fichiers décrite dans la AN1045 de Microchip.

Cette note d'application utilise cinq fonctions (voir Tableau 1.3) pour interfacer avec le matériel. En remplaçant ces fonctions à bas niveau avec SCSI commande qui utilise le driver d'unité de stockage USB pour la communication, nous pouvons utiliser cette note(1045) d'application pour fournir l'interface de système de fichiers au Disque flash USB.

Nom de fonction de Library	Description
USBHostMSDSCSIInit()	Appelé quand un dispositif est connecté.
USBHostMSDSCSIMediaDetect()	Indique si vraiment les médias sont Actuellement attaché
USBHostMSDSCSISectorRead()	Lit le secteur indiqué
USBHostMSDSCSISectorWrite()	Ecrit le secteur indiqué
USBHostMSDSCSIWriteProtectState()	Indique si les médias sont protégé a l'écriture

Tableau 1.3

1.6 Conclusion :

La programmation du Protocol USB est très complexe et demande des connaissances approfondie, avec un code correspondant très long.

Pour cela l'utilisation de Framework s'avère nécessaire. Le Framework sera expliqué dans le CHAPITRE4.