Protocollo USB

Elementi di un sistema USB

HOST: dispositivo master (PC), l'unico autorizzato ad avviare una comunicazione sul bus

HUB: come nelle reti di computer, l'hub fornisce un punto di connessione a più periferiche partendo da una sola porta USB.

DEVICE: ogni dispositivo nel sistema USB che non è un host è necessariamente una periferica (hub inclusi). Esistono periferiche autoalimentate, o alimentate dal bus e periferiche full-speed (12Mbps) o low-speed (1.5Mbps).

Comunicazione USB

L'host (unico dispositivo in grado di iniziare una comunicazione) interroga ciclicamente le periferiche seconda una certa schedulazione, inviando un pacchetto dati contenente i parametri:

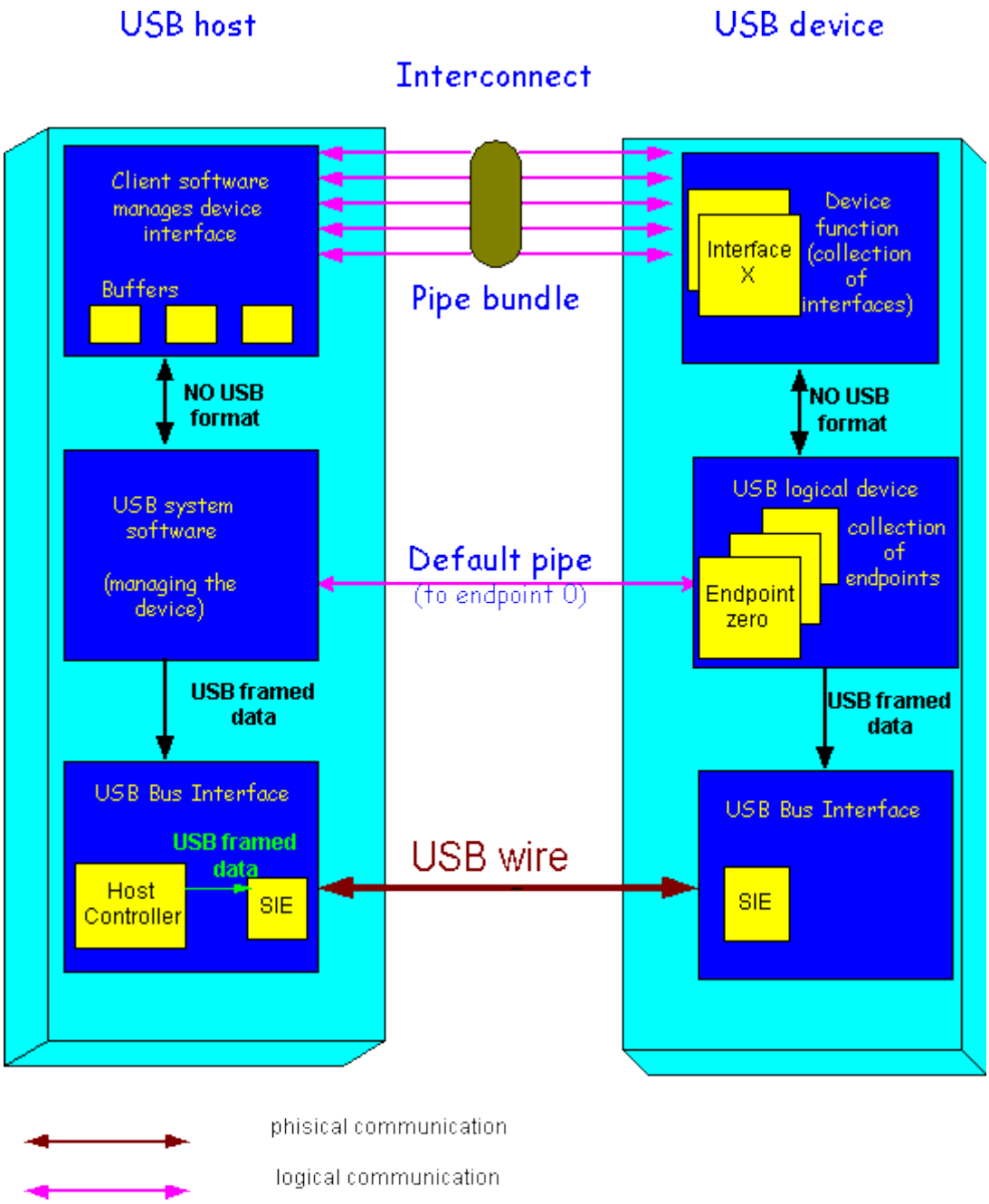
- Tipo di transazione
- Direzione della transazione
- Indirizzo della periferica USB interessata alla transazione
- Endpoint number

Ogni dispositivo che riceve il pacchetto dati USB (token packet):

- decodifica il campo indirizzo stabilendo se è l'interessato
- definisce il verso della transazione

L'entità che invia il pacchetto dati può rispondere con dati significativi oppure specificare che non ha nessun dato da trasmettere.

Il ricevitore da parte sua risponde con un pacchetto dati di handshake per notificare al trasmittente l'avvenuta ricezione e quindi il successo della transazione.



Flusso di comunicazione USB

La comunicazione logica tra client software (host) e funzione della periferica è effettuata tramite pipes. Un pipe è un'associazione tra uno specifico endpoint sulla periferica e l'appropriato software sull'host Un endpoint è la sorgente o la destinazione dei dati trasmessi sul cavo USB. Un'interfaccia è composta da endpoints raggruppati. Il client software trasmette dati tra il buffer dell'host e l'endpoint nella periferica, gestendo la specifica interfaccia.

La comunicazione sul bus può essere in due direzioni:

- OUT: flusso di dati da host a device
- IN: flusso di dati da device a host

Strati di comunicazione USB - Physical Layer

Lo strato fisico è l'interfaccia fisica ovvero il cavo USB. Il suo compito è di trasmettere e ricevere "0" ed "1" in modo corretto. La trasmissione di uno "0" avviene portando il potenziale del terminale D+ del cavo USB a livello basso e quello del terminale D- a livello alto, e viceversa per trasmettere un "1". I bit sono inviati partendo dal LSB. I dati vengono codificati e decodificati usando il metodo NRZI (Non Return to Zero Inverted – rappresenta i cambiamenti di stato logico con "1" e la permanenza di uno stato logico con "0").

Esistono alcuni tipi di segnali sul bus, individuati come casi speciali:

- Reset signaling: l'host può resettare la periferica inviando un SE0 (single ended zero) cioè portando entrambe i potenziali D+ e D- a livello basso per più di 2,5µs
- Suspend signaling: l'host può forzare la periferica in suspend mode, in cui il dispositivo non risponderà al traffico USB. Si attiva il suspend mode quando il bus rimane inattivo per più di 3ms e per un tempo non superiore ai 10ms
- Resume signaling: una periferica riprende le sue operazioni quando riconosce un segnale K ("0" differenziale per periferiche full speed e "1" differenziale per periferiche low speed) per un minimo di 20ms
- EOP (End Of Packet) signaling: corrisponde alla trasmissione di SE0 per 2 cicli d i clock, seguito da un segnale J ("1" differenziale per periferiche full speed e "0" differenziale per periferiche low speed) per un ciclo di clock.

Physical Layer - Serial Interface Engine (SIE)

La SIE è il modulo fisico presente nell'host e nei devices responsabile della serializzazione e deserializzazione (conversione dello stream di dati seriale in uno parallelo) delle trasmissioni USB.
Codifica i dati in uscita secondo la logica NRZI e decodifica i dati in ingresso.
Genera il codice di controllo CRC (Cyclic Redundancy Check) per gli stream in uscita e verifica il CRC degli stream in ingresso.
Rivela il PID (packet's id – è un campo nel pacchetto USB che individua il tipo di dato) così come i segnali SOP, EOP, RESET e RESUME sul bus.

Physical Layer - Host Controller (HC)

L'HC è l'elemento hardware più importante del sistema USB, colui che inizia tutte le transazioni, controlla gli accessi ed è il motore principale di controllo del flusso. Le sue funzioni fondamentali sono:

- Frame generation: l'HC è responsabile della partizione del tempo in unità di tempo di 1ms chiamate frame mediante l'emissione di uno SOF (Start Of Frame)

- Data Processing: gestione delle richieste di dati da e per l'host
- Protocol Engine: gestione del protocollo USB a livello interfaccia
- Error handling: gestione degli errori (Timeout, CRC, Overflow, …)
- Remote wakeup: possibilità di iniziare l'USB in suspend mode e rilevare un segnale di wakeup sul bus.

Protocol Engine Layer

È lo strato intermedio nel modello di comunicazione. Si occupa della traduzione dei dati tra applicazioni (software dell'host e funzioni della periferica) e il protocollo USB. Assume denominazione diversa a seconda che sia riferito all'host USB (USB System Software) o alla periferica USB (USB Logical Driver).

Protocol Engine Layer - USB System SW

Il sistema software USB è responsabile dell'allocazione della larghezza di banda, della gestione dell'energia fornita al bus oltre all'abilitazione delle periferiche per l'accesso al bus. Si compone di:

- Host Controller Driver (HCD): è un'interfaccia all'host controller.
- USB driver (USBD): il client software richiede dati dall'USBD nella forma di IRPs (I/O Request Packets) che consiste nella richiesta di invio/ricezione di dati attraverso un certo pipe. Fornisce al client una descrizione generale della periferica che il software sta gestendo. È necessario per creare un enumeration process (un processo che si attiva nel momento in cui la periferica è collegata al bus e alla fine del quale la stessa risulta totalmente configurata e diventa parte del sistema USB).

Protocol Engine Layer - USB Logical Device

È composto da un insieme di endpoints indipendenti: Ad ogni endpoint è dato un indirizzo unico (endpoint number) in fase di progetto e anche l'USB logical device è unicamente indirizzato alla fine dell'enumeration process. La combinazione di questi indirizzi e della direzione dei dati (da o per l'endpoint), caratterizza completamente l'endpoint.
Tutte le periferiche USB devono supportare la comunicazione con il default pipe, il quale gioca un ruolo importante nel processo di enumerazione ed è l'unico canale di comunicazione al device nel momento del collegamento. Il default pipe è associato all'endpoint zero, che è l'unico in grado di performare comunicazione bidirezionale.

Application Layer

Lo strato applicazioni è costituito dal client software per quanto riguarda l'host e dalle funzioni dal lato device. Il client software gestisce le interfacce trasferendo dati dal suo buffer agli endpoint associati alle interfacce stesse. Il client software lavora con una specifica funzione di una periferica, indipendentemente dalle altre funzioni della periferica nel sistema.

Il protocollo USB

Le transizioni USB avvengono tramite pacchetti. Ogni transizione è composta da tre fasi:

- Token phase: l'host inizia il pacchetto indicando il tipo di transizione
- Data phase: i dati attuali sono trasmessi tramite pacchetti. La direzione dei dati coincide con la direzione indicata nella fase di token trasmessa precedentemente
- Handshake phase (opzionale): è il pacchetto inviato indicante il successo o il fallimento della transizione

USB usa un polling protocol: ogni volta che l'host vuole ricevere dati da una periferica le invia un token;
- se la periferica ha dati da inviare li invia e l'host risponde con un pacchetto di handshake.
- se la periferica non ha dati da inviare, l'host emette un token verso la periferica successiva.
- se l'host vuole inviare dati ad una periferica, le invia prima un token appropriato e il successivo pacchetto di dati. La periferica risponde con un pacchetto di handshake.

Ci sono quattro tipi fondamentali di trasferimento USB:

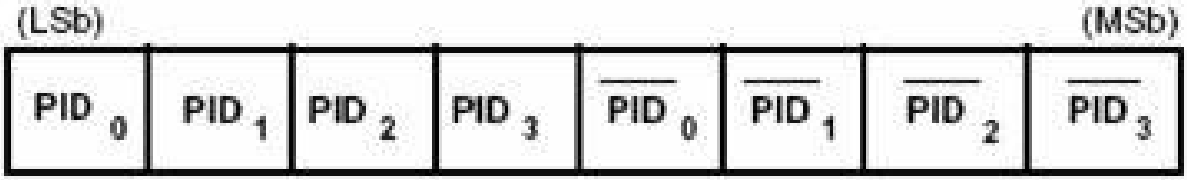
- Isochronous transfer: trasferimento isocrono, usato per periferiche multimediali (audio, video, …). È garantita la larghezza di banda richiesta. Questo tipo di trasferimento è meno attento al successo del trasferimento dati e permette più errori
- Bulk transfer: consiste nell'invio massiccio di dati (stampanti, scanner, …). La larghezza di banda allocata in ogni transizione varia in accordo con le disponibilità del bus nel momento del trasferimento. Garantisce pochi errori nella comunicazione
- Interrupt transfer: usato per periferiche che hanno bisogno di riportare brevi eventi (mouse, joystick, …)
- Control transfer: usato per configurare le periferiche. La configurazione avviene durante il processo di enumerazione ma si può effettuare in ogni istante della comunicazione

Il protocollo USB - Formato dei campi del pacchetto

Differenti campi dei pacchetti sono:

- Sync Field: si colloca all'inizio del pacchetto. Permette alle periferiche riceventi di sincronizzare il clock interno con i dati in ingresso
- PID – Packet Identifier Field: è il campo contenente l'identità del pacchetto. È composto da 8 bit: i primi 4 sono usati per notificare l'id del pacchetto e i 4 successivi sono usati come check bits

IMM8



Il protocollo USB - Formato dei campi del pacchetto – PID1

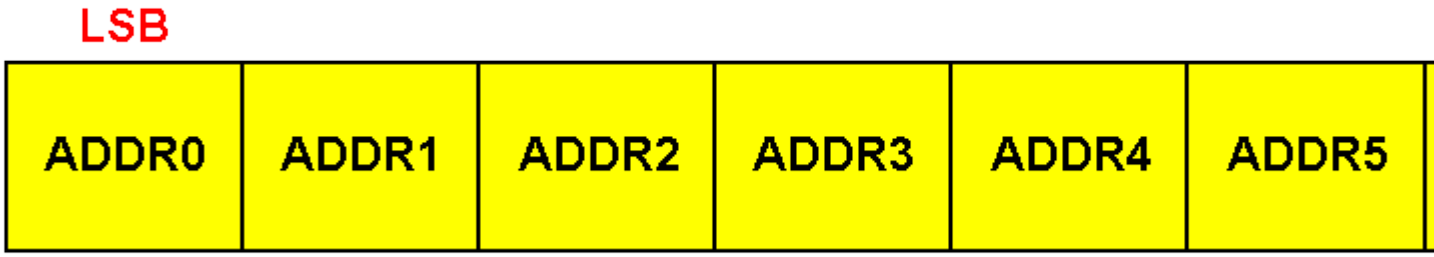
Il PID si divide in tre gruppi principali:

- Tokens:
  - Out token: indica che i dati successivi saranno trasmessi dall'host alla periferica
  - In token: indica che i dati successivi saranno trasmessi dalla periferica all'host
  - SOF token: indica lo start del frame
  - SETUP token: indica che i dati successivi saranno inviati dall'host al device e contengono comandi di setup usati per la configurazione
- Data: descrive il tipo di dato (pari, dispari, ad alta o bassa velocita, isocrono, …)
- Handshake: usato nei pacchetti di handshale prima di indicare il fallimento o la riuscita del trasferimento. Può essere:
  - ACK: il ricevitore riceve pacchetti liberi da errore
  - NAK: il ricevitore non è in grado di ricevere dati (overflow, …) o il trasmettitore non è in grado di inviare dati (underflow, …)
  - STALL: lo specifico endpoint si è arrestato o il comando di SETUP non è supportato

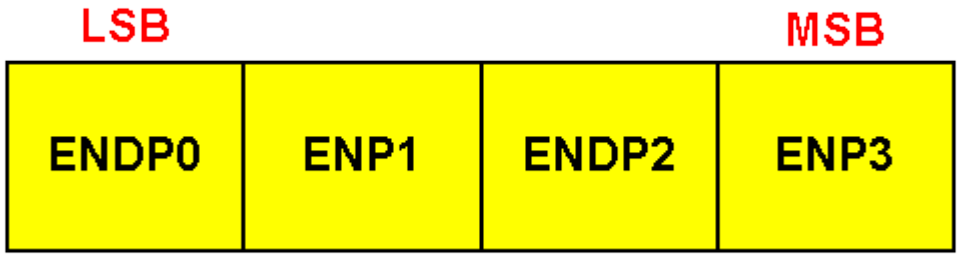
Il protocollo USB -Formato dei campi del pacchetto - Address Field

Il campo di indirizzo è diviso in due campi:

- Address field (ADDR): contiene l'attuale indirizzo della funzione (in genere è la periferica stessa), assegnato durante l'enumeration process. L'indirizzo della funzione è univoco e possono esserci 127 diversi indirizzi nel sistema



- Endpoint Field (ENDP): contiene il numero di endpoint assegnato ad una funzione.



Il protocollo USB - Formato dei campi del pacchetto -Frame Number Field

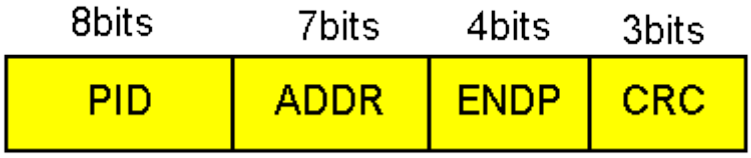
E' composto da 11 bit che indicano il numero del frame corrente. Il campo è contenuto solo nei token SOF che indicano l'inizio del frame Data field: contiene il dato trasmetto nella transizione. Può contenere un massimo di 1023 bytes.

CRC field: è usato per proteggere tutti i campi e i dati di un pacchetto (tranne il campo PID). È composto da 5 bit in un token packet e da 16 bit in un data packet.

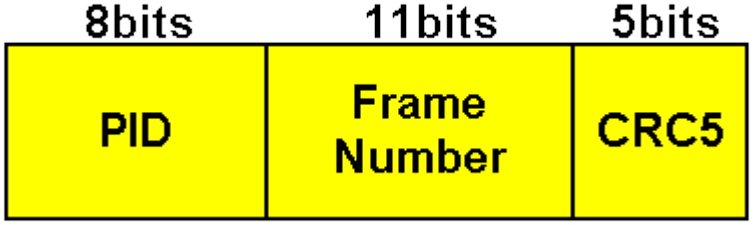
Formato dei pacchetti

I vari formati dei pacchetti sono:

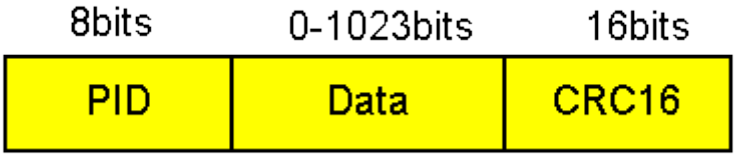
- Token packet: ogni transizione inizia con l'emissione di un token da parte dell'host. I campi ADDR e ENDP definiscono univocamente l'endpoint che deve ricevere i dati di SETUP o OUT o l'endpoint che deve trasmettere dati negli spostamenti IN



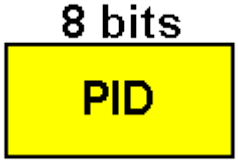
- Start of Frame Packet: l'host emette un SOF ogni 1msec±0.0005msec. Il pacchetto contiene il campo del numero del frame. SOF può essere usato come trigger per processi di OUT isocrono



- Data Packets: sono composti da PID (che indica che il pacchetto contiene dati), campo dei dati, e il codice CRC16 per proteggere i dati



- Handshake Packets: composto solo da PID indicante il risultato dello operazioni precedenti



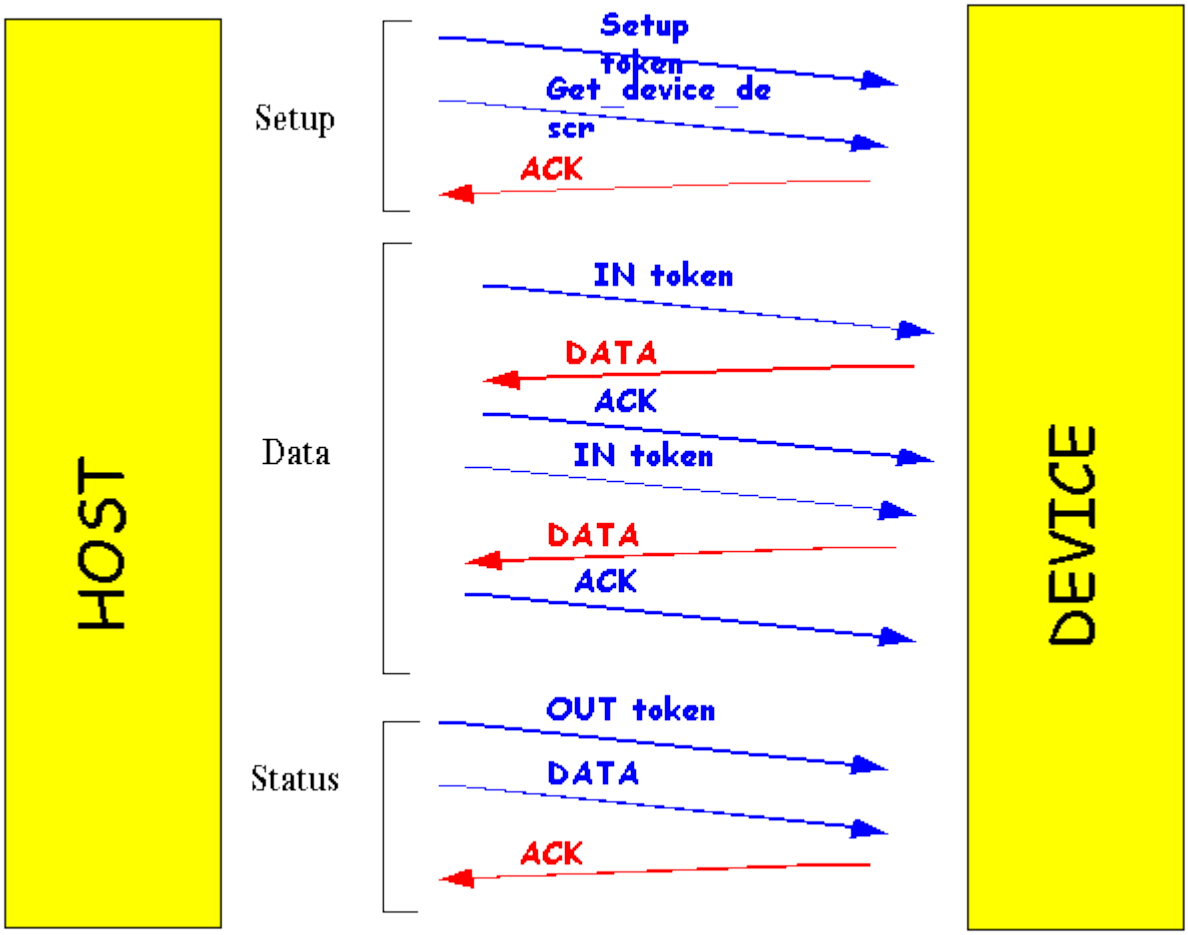
Formato delle transazioni - Control transfer

Il controllo dei trasferimenti si compone di due o tre fasi: setup, data (opzionale) e status, ognuna delle quali è composta da token, data, handshake.
Comandi di setup:

- SET_ADDRESS: impone un indirizzo permanente ad una funzione
- GET_DEVICE_DESCRIPTOR: comando per la ricezione di informazioni riguardo alla periferica
- GET_CONFIGURATION_DESCRIPTOR: comando per la ricezione di informazioni riguardo alla configurazione di una periferica
- GET_CONFIGURATION: l'host rileva quale configurazione è attiva al momento nella periferica
- SET_CONFIGURATION: l'host impone una configurazione specifica alla periferica

All'inizio di un operazione di setup, l'host emette un SETUP token, seguito da un pacchetto di comando setup. La periferica risponde con un pacchetto ACK. Successivamente, il flusso dei dati viene emesso nella direzione indicata nel stage precedente. Possono essere eseguite più transazioni (IN o OUT, ma tutte nella stessa direzione), ognuna delle quali inizia con un IN/OUT token emesso dall'host e termina con l'emissione dell'eventuale handshake. Lo status stage riporta i risultati dei precedenti stage all'host.

Esempio di control Transfer



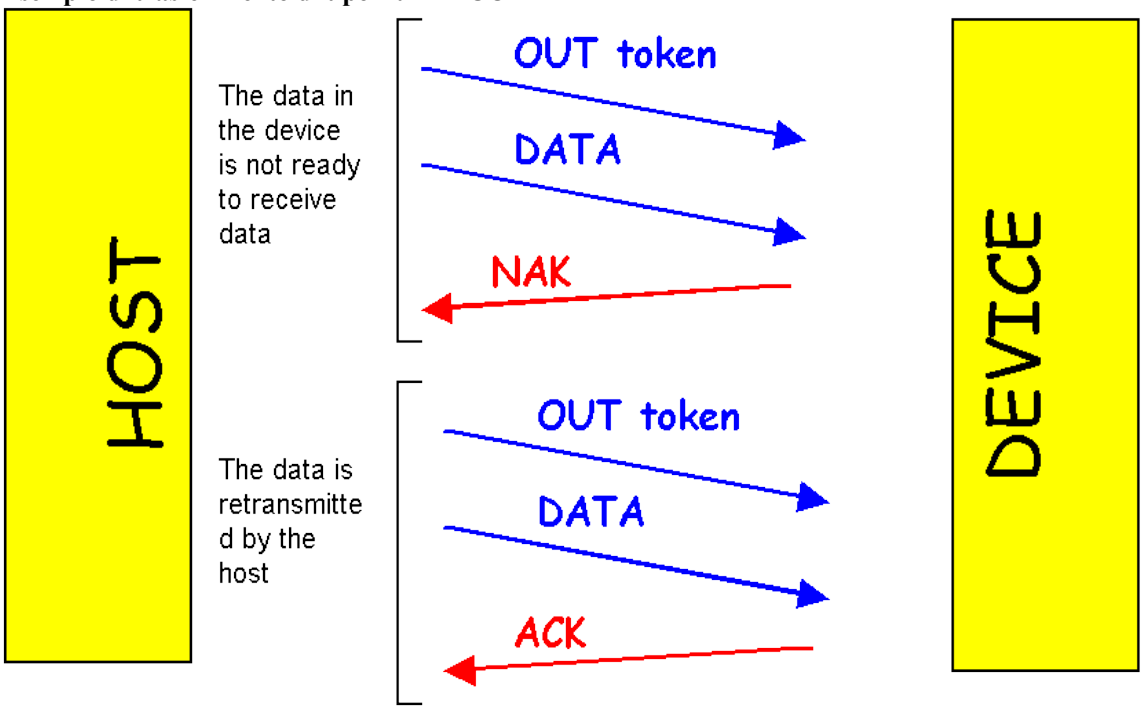
Formato delle transazioni - Bulk Transfer

La transazione si compone di tre fasi:

- Invio di un token dall'host indicante la direzione della comunicazione
- Trasmissione dei dati
- Se non si rivelano errori, si invia l'handshake, altrimenti nessun pacchetto di handshake è inviato

I trasferimenti di massa sono altamenti affidabili grazie ai meccanismi di handshake e timeout Se ci sono problemi nel sistema USB, l'host li individua e previene il blocco del sistema.

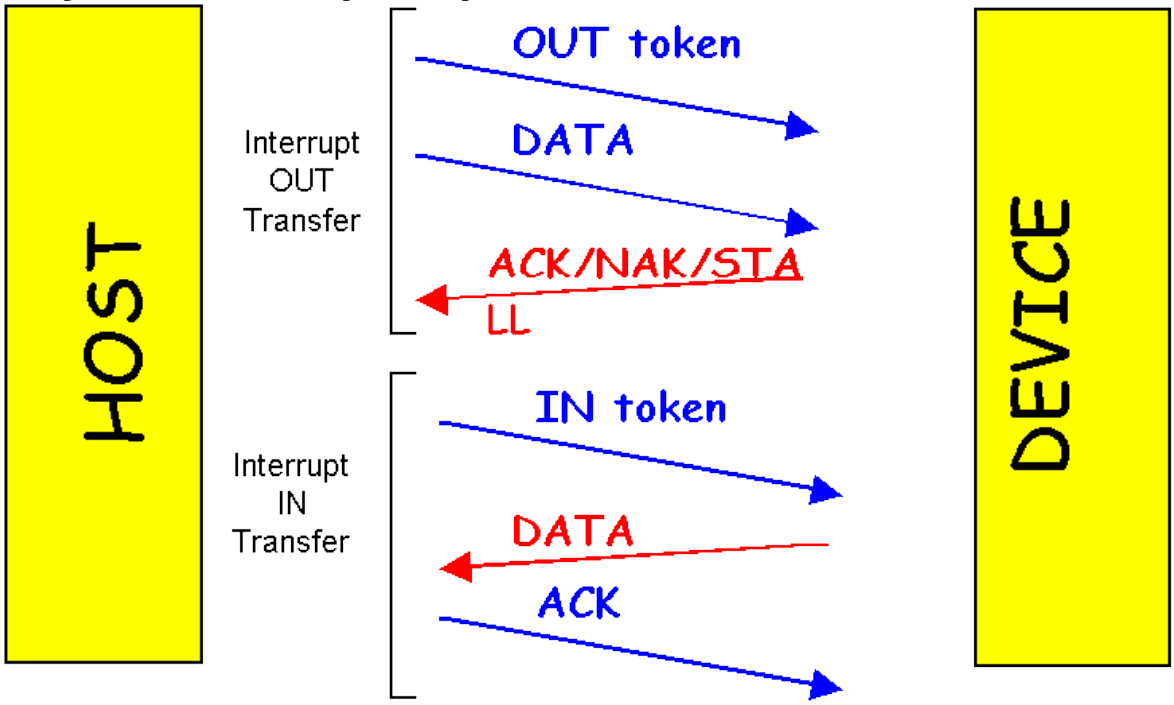
Esempio di trasferimento di tipo Bulk in OUT



Formato delle transazioni - Interrupt Transfer

È un metodo di trasferimento simile al bulk transfer.
L'host che vuole conoscere lo stato di un interrupt, invia un IN token all'appropriato endpoint.
Se ci sono interrupt in attesa, la funzione invia dettagli a riguardo. L'host risponde con un segnale di
ACK se il trasferimento ha avuto esito positivo. Se non ci sono interrupt in attesa, la funzione restituisce un NAK packet. Nel caso di errore nella funzione sarà inviato uno STALL packet. L'host invia un OUT token nel caso in cui vuole trasmettere dati alla periferica, seguito dal pacchetto di dati. Se il trasferimento è corretto l'host riceve un ACK o NAK o STALL packet. Se ci sono errori, non viene inviato handshake.

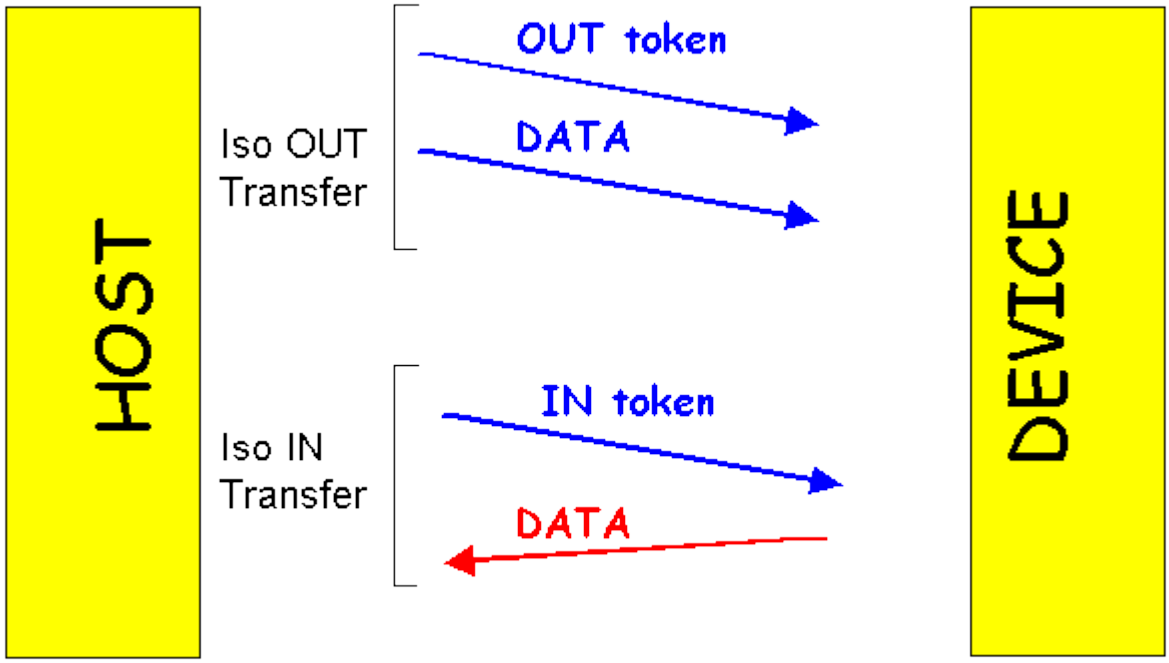
Esempio di trasferimento di tipo Interrupt



Formato delle transizioni - Isochronous Transfer

I trasferimenti isocroni sono composti da una o più fasi di transizione. L'host emette un IN token prima di ricevere dati dalla periferica, o un OUT token prima di inviare dati. Avviene l'emissione o la ricezione dei dati in base a quanto specificato precedentemente. Non esiste fase di handshake in questo tipo di trasferimento.

Esempio di trasferimento di tipo Isocrono



I contenuti di questa pagina sono in parte una traduzione del seguente sito: http://www2.rad.com/networks/2000/usb/maintxt.htm#USB_Communication_Layers