Versione N° 1.1

LaurTec

Il protocollo USB

Autore : *Mauro Laurenti*

ID: AN4008-IT

INFORMATIVA

Come prescritto dall'art. 1, comma 1, della legge 21 maggio 2004 n.128, l'autore avvisa di aver assolto, per la seguente opera dell'ingegno, a tutti gli obblighi della legge 22 Aprile del 1941 n. 633, sulla tutela del diritto d'autore.

AVVERTENZE

I progetti presentati non hanno la marcatura CE, quindi non possono essere utilizzati per scopi commerciali nella Comunità Economica Europea.

Chiunque decida di far uso delle nozioni riportate nella seguente opera o decida di realizzare i circuiti proposti, è tenuto pertanto a prestare la massima attenzione in osservanza alle normative in vigore sulla sicurezza.

L'autore declina ogni responsabilità per eventuali danni causati a persone, animali o cose derivante dall'utilizzo diretto o indiretto del materiale, dei dispositivi o del software presentati nella seguente opera.

Si fa inoltre presente che quanto riportato viene fornito così com'è, a solo scopo didattico e formativo, senza garanzia alcuna della sua correttezza.

L'autore ringrazia anticipatamente per la segnalazione di ogni errore.

Indice

Introduzione

Collegare un sistema Embedded al PC è una di quelle cose che entusiasma tutti gli appassionati di elettronica ed informatica. Negli anni 90 il tutto era facilmente ottenibile con la porta parallela utilizzata per collegare le stampanti al PC. Grazie alle istruzioni IN e OUT, offerte dai linguaggi di programmazione come il Basic, gestire la porta parallela era questione di pochi minuti. Con gli anni, la porta parallela è scomparsa dai PC come anche la porta RS232 utilizzata dai Modem. La scomparsa dei due bus dai Personal Computer è legata all'introduzione del protocollo USB (Universal Serial Bus) introdotto appunto per poter supportare molteplici applicazioni utilizzando un solo bus. L'intento è effettivamente riuscito e dopo oltre dieci anni siamo arrivati alla versione delle specifiche USB 3.1. In questo Tutorial vedremo i dettagli delle specifiche USB fornendo le conoscenze base per poter utilizzare senza problemi uno Stack USB per qual si voglia microcontrollore.

Il Tutorial descrive con maggiori dettagli le specifiche USB 2.0 visto che i microcontrollori con architettura ad 8, 16 e 32bit supportano le modalità Low Speed e Full Speed. Le specifiche USB 3.0 e 3.1 sono descritte principalmente per fornire un confronto con le attuali esigenze e sviluppi tecnologici, ma non sono forniti i dettagli delle specifiche.

Storia del protocollo USB

I primi Personal Computer (PC) introdotti sul mercato erano caratterizzati da molte interfacce e connettori dedicati a periferiche esterne. Il numero di connettori e difficoltà per l'utilizzatore erano molteplici. Basti per esempio ricordare il connettore per la tastiera e il mouse, che non dovevano essere invertiti altrimenti la tastiera e il mouse non funzionavano correttamente. Per capire se i cavi erano collegati correttamente bisognava vedere il disegno posto dietro il *case* del computer che riportava in maniera poco leggibile il simbolo della tastiera e mouse. Alcune marche di PC avevano inoltre connettori proprietari per cui la tastiera del PC non poteva essere utilizzata in altri. In maniera analoga l'interfaccia dei modem utilizzava un connettore DB25 maschio mentre le stampanti utilizzavano un DB25 femmina. Per limitare i problemi di inversione dei cavi le stampanti avevano frequentemente un connettore diverso da un lato al fine di evitare di collegare il cavo alla porta seriale. I modem hanno poi cominciato ad utilizzare il connettore DB9 piuttosto che DB25 al fine di ridurre lo spazio e problemi di errori con la porta parallela utilizzata dalle stampanti. Molti dei connettori appena descritti avevano inoltre le viti di fissaggio al fine di rendere tutto più stabile ma anche perché non era possibile un *hot-plug* ovvero non era possibile attaccare o staccare il connettore con il computer acceso. Per complicare la giungla di cavi dietro al PC ogni periferica oltre al cavo di connessione al PC, aveva un cavo di alimentazione. Ogni produttore di periferiche non avendo regole, cercava solo di soddisfare le proprie esigenze. Nel 1996 l'USB Implementers Forum (USB-IF) presentò la prima versione del protocollo USB. I grandi dell'industria si erano seduti assieme e avevano consolidato molte regole dettate dal buon senso, limiti meccanici, elettrici e di compatibilità. Stava nascendo il protocollo USB. Tra i cardini del protocollo USB sono state seguite le seguenti linee guida:

- Universalità
- Semplicità d'uso

Universalità

Il nome stesso del protocollo USB ovvero *Universal Serial Bus*, mette in evidenza uno degli scopi principali del nuovo standard, ovvero poter essere universale e indipendente dalla periferica: mouse, tastiera, memoria di massa, webcam o altro. Una flessibilità fornita dal fatto che lo standard supporta diversi bit rate a seconda delle esigenze.

Semplicità d'uso

Per la semplicità d'uso concorrono vari elementi sia hardware che software. In particolare i connettori sono diversi da un lato e dall'altro al fine di evitare l'inversione dei cavi (fa eccezione il nuovo standard meccanico USB Type C). L'alimentazione delle periferiche collegate al bus può essere fornita dal bus stesso fino ad un massimo di 500mA o 900mA a seconda della versione USB. Una periferica può essere collegata e staccata dal PC anche quando è acceso (*hot-plug* e *unplug*). Ogni periferica collegata al bus USB contiene delle informazioni base su se stessa che permettono al sistema operativo di decidere quale driver utilizzare, qualora esso sia già disponibile sul PC.

A partire dall'introduzione della prima versione USB 1.0 si sono succedute diverse versioni al fine di poter seguire le esigenze di mercato e le nuove periferiche introdotte. Nonostante l'attuale versione USB 3.1, ogni specifica USB introdotta dall'USB IF ha mantenuto la compatibilità con le precedenti versioni. Questa decisione ha favorito ulteriormente l'utilizzo della porta USB come standard di fatto, favorendo la sua adozione in molte periferiche collegabili al PC.

Le specifiche USB rilasciate dall'USB IF descrivono il protocollo da un punto di vista del software, elettrico e meccanico, fornendo test di verifica per garantire che siano propriamente implementate e poter certificare un prodotto come conforme alle specifiche USB facendo uso degli opportuni loghi da associare al prodotto.

Vediamo brevemente le varie versioni delle specifiche USB e le principali caratteristiche tecniche che contraddistinguono le stesse. Sebbene la storia inizi con la versione 1.0, l'USB IF ha rilasciato le seguenti versioni intermedie utilizzate nei primi prototipi di chip non rilasciati sul mercato: ver. 0.7, 0.8, 0.9, 0,99, 1.0 FDR.

Specifiche USB 1.0

Le specifiche USB 1.0 vennero rilasciate dall'USB IF nel 1996. Le specifiche vennero pensate per poter supportare periferiche lente come la tastiera e mouse, ma anche periferiche con esigenze di trasferimento dati maggiori, come per esempio le memorie di massa. Il protocollo definisce due modalità:

- Low Speed, con bit rate fino a 1.5Mbits/s
- Full Speed, con bit rate fino a 12MBits/s

Il protocollo USB, alla sua introduzione non era direttamente supportato dal sistema operativo Windows 95, il quale richiedeva driver specifici aggiuntivi.

Specifiche USB 1.1

La versione USB 1.1 venne introdotta nel 1998 al fine di poter correggere alcuni problemi che non erano stati messi in evidenza fino all'utilizzo più esteso del protocollo avvenuto con l'adozione da parte di Apple sui propri PC iMac. In particolare Apple ebbe l'idea semplice ma efficace di utilizzare la porta USB come porta universale per le varie periferiche esterne, ovvero sfruttare proprio una delle ragioni principali per cui si erano definite le specifiche USB. La versione USB 1.1 non introdusse alcun cambiamento dal punto di vista del bit rate, in particolare le modalità supportare rimasero Low Speed e Full Speed.

Specifiche USB 2.0

Il numero di periferiche che faceva uso della porta USB crebbe, ed in particolare l'esigenza della velocità di trasferimento dati andava oltre alle specifiche USB 1.x. Nel 2000, l'USB IF introdusse le nuove specifiche USB 2.0. Tra le principali caratteristiche venne introdotta la nuova modalità:

- High Speed, con bit rate fino a 480Mbits/s

E' importante mettere subito in evidenza che periferiche compatibili con le specifiche USB 2.0 non devono necessariamente supportare 480Mbits/s. Per esempio i microcontrollori con architettura ad 8, 16 e 32 bit, pur supportando le specifiche USB 2.0 supportano frequentemente solo la modalità Low Speed e Full Speed. Questo discende semplicemente dal fatto che per supportare la modalità High Speed sono richieste frequenze di clock e risorse che eccedono quelle normalmente presenti in un microcontrollore. Le specifiche USB 2.0 hanno reso obsolete le specifiche USB 1.x.

Specifiche USB 3.0

Nel 2008 l'USB IF ha introdotto le nuove specifiche USB 3.0 con lo scopo di poter supportare un bit rate fino a 5Gbits/s, introducendo la nuova modalità:

- Super Speed, con bit rate fino a 5Gbits/s

Per poter raggiungere le nuove specifiche tecniche sono stati necessari diversi cambiamenti da un punto di vista elettrico e meccanico, in particolare il nuovo standard ha introdotto un nuovo connettore mantenendo però la compatibilità con le vecchie specifiche. Le nuove specifiche hanno introdotto anche ulteriori cambiamenti al fine di supportare un'esigenza collaterale nata con il tempo, ovvero la ricarica del cellulare. In particolare fino alle specifiche USB 2.0 un dispositivo USB poteva richiedere fino ad un massimo di 500mA, mentre le periferiche USB 3.0 ne possono richiedere fino a 900mA.
Una periferica USB 2.0 collegata ad una porta USB 3.0 può richiedere una corrente massima di 500mA in accordo alle specifiche USB 2.0 e non 900mA.

Specifiche USB 3.1

Sebbene il protocollo USB 3.0 abbia esteso il bit rate a 5Gbits/s, nel 2013 l'USB IF ha introdotto la nuova modalità:

- Super Speed+, con bit rate fino a 10Gbits/s

Le nuove specifiche hanno mantenuto la compatibilità con le specifiche USB 2.0 e USB 3.0. Le specifiche USB 3.x non hanno reso obsolete le specifiche USB 2.0 ma le hanno piuttosto estese. Nuovi dispositivi possono infatti essere rilasciati sul mercato soddisfacendo solo le specifiche USB 2.0.

Licenza di utilizzo del protocollo USB

La comodità della porta USB è stata pensata prevalentemente per gli utilizzatori finali, che effettivamente rappresentano la maggioranza dell'utenza se paragonata ai progettisti. I progettisti o produttori che vogliano mettere sul mercato un prodotto USB devono iscriversi al gruppo USB IF e richiedere un proprio *Vendor ID* (VID) da associare ai propri prodotti affinché siano propriamente riconosciuti dal sistema operativo che dovrà associare il Driver al dispositivo. In realtà la scelta del driver viene a dipendere non solo dal VID ma anche dal *Product ID* (PID), ma i dettagli li vedremo a breve. In particolare ad ogni VID vengono assegnati anche 65536 combinazioni di PID. Il prezzo del VID dipende dal tipo di contratto che viene fatto ma è a 4 cifre, senza contare i decimali.
Questi costi sono effettivamente elevati per piccole società, ma visto che ad ogni VID ogni produttore riceve molti PID disponibili, le case produttrici di bridge USB e MCU hanno pensato il *sublicensing* dei PID, ovvero si ha la possibilità di usare il VID del produttore del chip e il PID assegnato con il *sublicensing*. Questa tipologia di contratto di tipo gratuita richiede che il VID e PID siano utilizzati solo con i chip del produttore associato al VID. Un altro limite vien posto anche sul numero massimo di prodotti che possono essere introdotti sul mercato con la combinazione VID-PID. Normalmente questo limite è associato a volumi di vendita abbastanza elevati da non porre più grossi problemi ad un acquisto diretto di un proprio VID e PID.

Oltre alla licenza associata al VID e PID è importante mettere in evidenza che il logo USB può essere utilizzato solo sui prodotti che soddisfano determinati test imposti dall'USB Implementers Forum e per i quali sia stata pagata apposita licenza. Per i prodotti di piccola produzione per i quali siano stati dati in sublicensing VID e PID non si ha in automatico il diritto di usare il logo USB.

Sebbene il VID e PID aiutino ad identificare il dispositivo e decidere che driver associare allo stesso in base ad informazioni interne al dispositivo, realizzare un prodotto USB che venga riconosciuto senza avvisi all'armanti per l'utilizzatore finale, richiede altri test ovvero soldi. Affinché il proprio dispositivo sia riconosciuto dal sistema operativo di Windows senza avere messaggi di avviso per l'utente finale riguardo al fatto che il driver o dispositivo non sono registrati e potrebbero causare problemi, bisogna fare ulteriori test software e pagare una licenza opportuna. Anche qui sono disponibili diverse licenze, ma la cosa buona è che tolto il fatto che non potete usare il logo di Windows o il logo USB senza pagare delle licenze, nulla vi vieta di poter mettere in commercio il vostro prodotto facendo uso della porta USB.

Specifiche di sistema del protocollo USB

Il protocollo USB permette la comunicazione tra il PC e un dispositivo per mezzo di un collegamento a stella. Ovvero il PC è al centro della rete mentre i vari dispositivi sono collegati al PC. In maniera più precisa le specifiche USB definiscono i seguenti elementi:

- Host
- Device (Dispositivi)
- Hub

collegati tra loro per mezzo di cavi. In particolare l'Host è rappresentato spesso dal PC, e rappresenta il dispositivo Master, ovvero colui che inizia ogni comunicazione, almeno per le specifiche USB 2.0. In particolare l'Host controlla tutti i dispositivi ad esso collegati, che in una rete USB raggiunge fino a 127 periferiche, includendo tutti i dispositivi e Hub. Normalmente l'Host ha un numero limitato di porte USB dove è possibile fisicamente collegare dei dispositivi esterni. Per estendere il numero di porte si fa uso di Hub, i quali si collegano ad una porta dell'Host o altro Hub ed estendono la stessa con un numero tipico di 2, 4 e 7 porte aggiuntive. Lo stesso Host integra al suo interno un Hub al fine di poter fornire le porte di collegamento presenti sul PC. Tale Hub prende il nome di *root Hub*, ovvero Hub radice o principale. Le specifiche USB permettono di collegare in una rete fino a 5 livelli di Hub. Sebbene sia possibile collegare 127 dispositivi su un solo Host, le risorse che sarebbero disponibili per dispositivo potrebbero risultare piuttosto limitate. Infatti il bit rate supportato dalle specifiche USB deve intendersi per Host e non per porta. Il disagio di risorse limitate si potrebbe far sentire in realtà molto prima di raggiungere 127 dispositivi, per questo spesso i PC possiedono in realtà due Host e due *root Hub* al fine di mantenere una maggior flessibilità e un bit rate adeguato per ogni periferica. Da quanto appena detto, una rete tipica basata sul protocollo USB potrebbe essere simile a quella riportata in Figura 1.

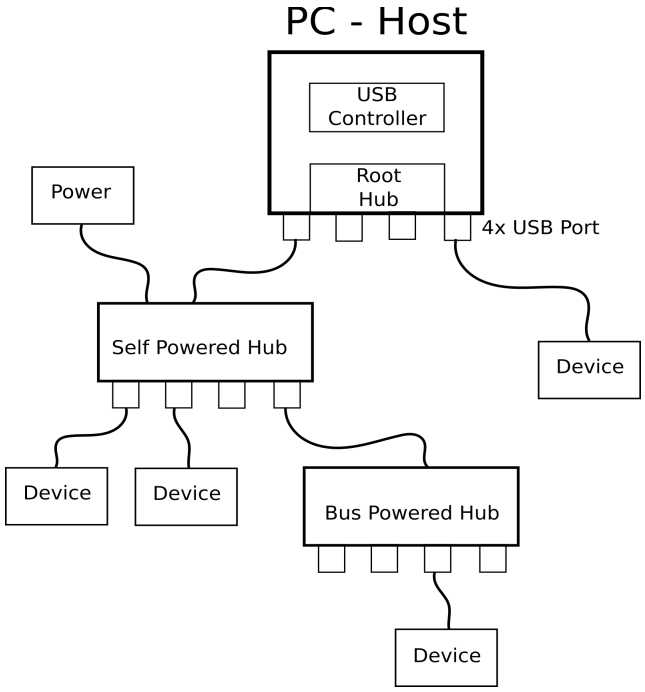

Figura 1: *Rete tipica disponibile con il protocollo USB.*

Enumerazione di un dispositivo

Ogni volta che un dispositivo viene collegato ad un Host, vengono eseguite varie operazioni al fine di poter utilizzare il dispositivo in maniera opportuna. Come prima cosa l'Host deve riconoscere che il dispositivo è stato collegato (*Attached*). Questo avviene controllando le linee del bus USB e il relativo resistore di pull-up che viene inserito dal dispositivo (maggiori dettagli sono mostrati in seguito). Questo determina una variazione di tensione sul bus che appunto identifica sia il collegamento del dispositivo che un eventuale scollegamento (*Detached*) dello stesso. Per i dispositivi High Speed e nel protocollo USB 3.x, l'enumerazione e controllo di tensione sul bus avviene in maniera differente.

Rilevato il collegamento di un dispositivo alla porta USB, l'Host lo inizializza associandogli un indirizzo unico che verrà poi utilizzato per identificare in maniera univoca la periferica sul bus. Tale valore può variare da connessione a connessione e l'utilizzatore finale non deve compiere alcuna operazione affinché ciò avvenga. Con l'avvenuta inizializzazione viene formato un canale di comunicazione tra l'Host e il dispositivo che prende il nome di *Pipe*. Tramite tale canale avviene la comunicazione effettiva tra l'Host e il dispositivo. Ogni dispositivo potrebbe avere più canali di comunicazione a seconda del tipo di dispositivo ma comunque un solo indirizzo. In particolare per ogni verso di comunicazione viene a crearsi un *Pipe*. La sola formazione logica del canale di comunicazione non è ancora sufficiente per utilizzare il dispositivo, il quale su richiesta deve fornire altre informazioni al fine di classificare il dispositivo stesso e permettere al sistema operativo di poter caricare il relativo driver. Per i dispositivi USB 3.x, durante l'enumerazione viene creato un dispositivo composito ovvero un doppio dispositivo per il supporto USB 2.0 e USB3.0.

Endpoints

Ogni dispositivo possiede degli Endpoint ovvero dei buffer di memoria per mezzo dei quali è possibile scambiare informazioni con l'Host. L'Endpoint 0 è quello base di configurazione, che viene utilizzato dall'Host al fine di configurare il dispositivo durante la fase di enumerazione. Gli Endpoint possono avere un numero compreso tra 0 e 15 (USB Full Speed a High Speed) e ognuno ha un verso di trasmissione IN e OUT, oltre ad una dimensione che identifica il buffer stesso. In particolare tale verso viene definito con riferimento all'Host, per cui un Endpoint IN rappresenta un buffer di memoria presente sul dispositivo che fornisce informazioni all'Host. Un Endpoint OUT rappresenta un buffer nel dispositivo che riceve dati in uscita dall'Host. L'Endpoint rappresenta un elemento della Pipe precedentemente definita. L'Host non possiede degli Endpoint ma semplici buffer di ingresso e uscita.

Un dispositivo che voglia inviare dei dati all'Host non deve attendere il suo turno per inviare i dati prima di caricare gli stessi nell'Endpoint, bensì li deve caricare nell'Endpoint prima che avvenga la richiesta da parte dell'Host dei dati. Se così non si facesse l'Host non troverebbe mai dei dati pronti per la lettura e la periferica non riuscirebbe mai ad inviarli.

Tipologie di trasmissione

Il protocollo USB, al fine di supportare un'ampia tipologia di applicazioni, fornisce diverse modalità di trasmissione dati. In particolare lo standard USB 2.0 supporta le seguenti tipologie di trasmissione:

- Control
- Interrupt
- Bulk
- Isochronous

Tutte le tipologie di trasmissione dati sono caratterizzate dal non avere una reale formattazione dei dati, eccetto per il Control quando deve inizializzare il dispositivo. Il formato dei dati viene infatti a dipendere dal dispositivo stesso, il Firmware e relativo driver. Vediamo qualche dettaglio delle varie modalità.

Control

La modalità Control è utilizzata durante la fase di inizializzazione del dispositivo. In particolare fa uso dell'Endpoint 0 al fine di scambiare i dati con il dispositivo. La trasmissione dati per mezzo della modalità Control permette in particolare di leggere i diversi Descriptor presenti nel dispositivo, ovvero le informazioni sullo stesso.
In Tabella 1 sono riportati i dettagli delle dimensioni dei pacchetti dati che è possibile inviare in tale modalità.

Standard	Dimensione Pacchetto dati
Low Speed	8
Full Speed	8, 16, 32, 64
High Speed	64

Tabella 1: *Dimensioni pacchetti dati.*

Sebbene la modalità Control venga generalmente utilizzata per la fase di inizializzazione del dispositivo, nulla vieta di utilizzare tale modalità anche per la comunicazione dati vera e propria. Al fine di supportare la fase di inizializzazione ogni dispositivo deve avere l'Endpoint 0.

Interrupt

La modalità di trasmissione Interrupt viene utilizzata da quei dispositivi che devono fornire dati in maniera piuttosto rapida, come per esempio mouse e tastiere. Sebbene la modalità si chiami Interrupt, in realtà i dispositivi USB non generano nessun Interrupt che permette di richiamare l'attenzione dell'Host al fine di poter fornire i dati. La comunicazione tra Host e Dispositivo è sempre guidata dall'Host che interroga periodicamente il Dispositivo al fine di determinare se sono presenti o meno dei dati. Quanto appena detto non è vero nel caso di Dispositivi USB 3.x i quali possono effettivamente inviare una richiesta all'Host al fine di poter richiedere una lettura dei dati eventualmente disponibili. La modalità Interrupt garantisce che la lettura avvenga in un determinato lasso di tempo impostabile tra le configurazioni del dispositivo ovvero nel

rispettivo Descriptor. In Tabella 2 sono riportati i dettagli delle dimensioni dei pacchetti dati che è possibile inviare in tale modalità.

Standard	Dimensione Pacchetto dati	Intervallo
Low Speed	1-8	10-255ms
Full Speed	1-64	1-255ms
High Speed	1-1024	125us-4096s

Tabella 2: *Dimensioni pacchetti dati.*

Quando un dispositivo è configurato in modalità Low Speed, supporta solo le modalità Control e Interrupt.

Bulk

La modalità Bulk, sebbene non supporti dei tempi prestabiliti per lo scambio delle informazioni, qualora il bus USB sia per lunghi periodi in stato di Idle, permette di raggiungere il miglior bit rate, ovvero trasferimento dati possibili. Ciononostante qualora l'Host debba condividere il bus con più dispositivi, la modalità Bulk potrebbe non essere molto efficiente. In particolare fanno uso della modalità di trasmissione Bulk i dispositivi di memoria di massa e anche la classe CDC (*Communication Device Class*). In Tabella 3 sono riportati i dettagli delle dimensioni dei pacchetti dati che è possibile inviare in tale modalità.

Standard	Dimensione Pacchetto dati
Full Speed	8, 16, 32, 64
High Speed	512

Tabella 3: *Dimensioni pacchetti dati.*

Isochronous

La modalità Isochronous è stata pensata per raggiungere un'elevata trasmissione dei dati ma senza segnali di *Acknowledge* ovvero di segnalazione di correttezza dei dati. Il flusso di dati viene semplicemente inviato come se vengano sempre ricevuti correttamente. Tale modalità può tornare utile nel caso di sistemi video e audio dove eventuali interruzioni o disturbi sono in generale accettabili. Al tempo stesso non essendo presenti segnali di correzione dei dati si capisce che questa modalità, sebbene veloce, non potrebbe essere utilizzata per la trasmissione di file, a meno di non implementare sistemi di correzione a livello del dispositivo, ma se si hanno queste esigenze è bene usare un'altra modalità. In Tabella 4 sono riportati i dettagli delle dimensioni dei pacchetti dati che è possibile inviare in tale modalità.

Standard	Dimensione Pacchetto dati
Full Speed	0-1024
High Speed	0-1023

Tabella 4: *Dimensioni pacchetti dati.*

Classe dei Dispositivi

Il protocollo USB sebbene non definisca un formato dati predefinito, lasciando libertà al singolo Hardware e driver di definire il modo con cui vengono formattati e inviati i dati, definisce delle classi di dispositivi, associando ad esse particolari caratteristiche da un punto di vista della modalità di trasmissione che deve essere utilizzata. Questo comporta che volendo utilizzare una determinata modalità di trasmissione ci si ritrova piuttosto a scegliere la particolare classe di dispositivi che si avvicina al particolare sistema che si sta progettando. Tra le classi di dispositivi più note che ci si trova ad utilizzare con microcontrollori ad 8 bit si ricordano:

- HID : Human Interface Device
- Mass Storage Device
- CDC : Communication Device Class

Oltre a queste classi di dispositivi, qualora si stia utilizzando un microcontrollore a 32bit o Microprocessori a 32-64 bit, si può avere a che fare anche con le classi di dispositivi riportati in Tabella 5.

ID	Descriptor	Device Class
0x00	Device	L'informazione della classe è fornita nell'Interface Descriptor
0x01	Interface	Classe Audio
0x02	Entrambi	Comunicazione (CDC)
0x03	Interface	HID (Human Interface Device)
0x05	Interface	Physical
0x06	Interface	Image
0x07	Interface	Stampanti
0x08	Interface	Memoria di massa (Mass Storage)
0x09	Device	Hub
0x0A	Interface	Comunicazione (CDC)
0x0B	Interface	Smart Card
0x0D	Interface	Content Security
0x0E	Interface	Video
0x0F	Interface	Personal Healthcare
0x10	Interface	Audio/Video
0x11	Device	Billboard Device Class
0x12	Interface	USB Type C Bridge Class
0xDC	Entrambi	Diagnostic Device
0xE0	Interface	Wireless Controller
0xEF	Entrambi	Miscellaneus
0xFE	Interface	Application Specific
0xFF	Entrambi	Vendor Specific

Tabella 5: *Classe di dispositivi USB.*

In particolare la Tabella riporta il codice identificativo della classe e anche il tipo di Descriptor a cui fa riferimento, ovvero Device, Interface o entrambi. Maggiori dettagli sui Descriptor sono forniti a breve.

L'identificazione della Classe avviene scrivendo il relativo ID nel parametro *bDeviceClass* contenuto nei rispettivi Descriptor del dispositivo. Oltre ai valori standard l'utente finale è comunque libero di definire una propria classe, semplicemente scrivendo il codice esadecimale 0xFE e 0xFF.

USB Descriptor

Abbiamo visto che un dispositivo, una volta che lo si collega ad un Host o Hub, viene inizializzato al fine di assegnare un indirizzo univoco sul bus. Oltre a fornire queste informazioni al dispositivo, l'Host richiede anche informazioni al dispositivo per poterlo identificare e determinare quale driver debba essere caricato al fine di permettere il corretto funzionamento della periferica. Tutte le informazioni che permettono all'Host di comunicare correttamente con il dispositivo, inizializzarlo e determinare il driver da caricare, sono contenute all'interno di vari *Descriptor* (descrittori) contenuti nel dispositivo. Un Descriptor non è altro che un insieme di registri che contengono tutte le informazioni necessarie e vengono letti in successione. Da un punto di vista della programmazione un Descriptor è generalmente una struttura dati i cui campi sono organizzati al fine di soddisfare le specifiche USB in termini di ordine dei campi e byte assegnati agli stessi. In particolare le varie informazioni sono distribuite in diversi Descriptor e ogni Descriptor è identificato da un codice. L'Host in base al Descriptor che vuole leggere invia al Dispositivo il relativo comando (*Get Descriptor*) che contiene il codice del Descriptor che vuole leggere e il Dispositivo deve rispondere con il contenuto della struttura dati richiesta. I Descriptor definiti dalle specifiche USB sono i seguenti:

- Device Descriptor
- Configuration Descriptor
- Interface Descriptor
- Endpoint Descriptor
- String Descriptor

In particolare i dispositivi che supportano le specifiche USB High Speed, ovvero devono cambiare la velocità di comunicazione, o possono cambiarla, possiedono dei Descriptor aggiuntivi che permettono di fornire le informazioni del dispositivo e relative configurazioni nella modalità High Speed.

I vari Descriptor sono in generale definiti all'interno dello Stack Software che viene fornito dal produttore del microcontrollore utilizzato. In particolare molti produttori, oltre a fornire lo Stack USB forniscono anche molti esempi per le diverse classi di dispositivi per cui ci si trova spesso a dover semplicemente prendere l'esempio che più si avvicina alla propria applicazione. Spesso l'esempio non può però essere utilizzato senza dover cambiare alcuni parametri di configurazione. In particolare i Descriptor che ci si trova maggiormente a dover cambiare e adattarli alle proprie esigenze sono:

- Device Descriptor
- Configuration Descriptor

In particolare il Device Descriptor è utile per i seguenti parametri:

- Può definire la classe del dispositivo, anche se spesso questa viene definita nell'Interface Descriptor.
- Contiene i parametri VID (idVendor) e PID (idProduct) che permettono di identificare il proprio dispositivo. Come detto il VID e PID sono univoci per ogni prodotto e ci sono diversi modi per poterne ottenere uno.
- E' possibile definire il numero seriale del dispositivo. Questo può tornare utile qualora sia possibile attaccare all'Host più dispositivi dello stesso tipo e si abbia l'esigenza di doverli identificare in maniera univoca. Infatti in questo caso il VID e PID sarebbero gli stessi mentre il numero di serie sarebbe diverso.
- E' possibile impostare del testo informativo per il dispositivo, ovvero puntatori a stringhe che contengono una breve descrizione del dispositivo.

Il Configuration Descriptor torna utile per i seguenti parametri:

- Permette di definire se il dispositivo è alimentato da USB o tramite alimentazione esterna.
- Permette di impostare la massima corrente che il dispositivo assorbirà dopo l'enumerazione. In particolare la corrente può essere al massimo di 500mA per un dispositivo USB 2.0 mentre 900mA per un dispositivo USB 3.x. Per le specifiche USB 2.0 si deve scrivere il valore come multiplo di 2mA. Per cui per scrivere 200mA si deve scrivere 100 (da convertire in esadecimale a seconda delle esigenze).

Dati sul bus

Normalmente la conoscenza di quanto spiegato fino ad ora è più che sufficiente per per coloro che devono sviluppare un'applicazione Embedded. Infatti i dettagli della comunicazione a livello dei pacchetti dati e controllo della loro correttezza avviene a livello Hardware e il Software non deve in generale preoccuparsi dei dettagli. A scopo di completezza è comunque bene avere una conoscenza di base di ciò che avviene sotto il cofano e sapere le basi dell'organizzazione dei dati.
La comunicazione, come visto è controllata sempre dall'Host, almeno secondo le specifiche USB 2.0 ed avviene per mezzo della lettura e scrittura degli Endpoint. Il trasferimento di dati avviene per mezzo di transazioni multiple di dati (*Transaction*). Ogni transazione, a seconda della tipologia della stessa, può essere divisa in:

- Token Packet
- Data Packet
- Handshake Packet

A sua volta il Token Packet permette di trasmettere le seguenti informazioni che permettono di identificare il dispositivo e l'Endpoint a cui fare riferimento

- PID: Packed Identifier
- Address: Indirizzo del dispositivo

- Endpoint: Numero dell'Endpoint
- CRC: Cyclic Redundancy Check per l'identificazione di eventuali errori

Il Data Packet contiene i seguenti campi:

- PID: Packed Identifier
- Data: Byte dati in numero dipendente dal tipo di trasferimento
- CRC: Cyclic Redundancy Check per l'identificazione di eventuali errori

L'*Handshake* Packet contiene il solo campo PID.

Il modo con cui una transazione viene effettivamente suddivisa, viene a dipendere dal tipo di transazione. In particolare le transazioni Control, Bulk, Interrupt possiedono tutte e tre le componenti Token Packet, Data Packet e Handshake Packet, mentre la modalità Isochronous possiede solo il Token Packet e Data Packet. Questo discende dal fatto che seppure si dovessero presentare degli errori i dati non verrebbero ritrasmessi, per cui non è importante avere la conferma della correttezza dei dati. L'*Handshake Packet* non serve solo per segnalare eventuali errori e richiedere di spedire nuovamente i dati, ma anche per permettere di avvertire l'Host se il Device non è pronto, inviando in particolare un NAK (*Not Acknowledged*). Si capisce che dal momento che un eventuale segnalazione da parte del Dispositivo relativa al non essere pronto avviene solo alla fine della trasmissione dei dati, qualora il dispositivo non dovesse essere pronto frequentemente, si potrebbe perdere molto tempo, ovvero banda sul bus. Per questa ragione l'USB 2.0 High Speed ha introdotto anche la funzione PING che permette all'Host, qualora abbia ricevuto una richiesta di attesa da parte del dispositivo, di riprovare con il comando PING per verificare se il dispositivo è nuovamente pronto, piuttosto che inviare nuovamente tutti i dati e ricevere nuovamente una richiesta di attesa. La versione USB 3.0 ha inoltre aggiunto la possibilità di permettere al Dispositivo di richiedere all'Host di leggere i dati, perché pronti nel relativo Endpoint, non richiedendo dunque nemmeno l'invio del PING.

Il campo PID è di un byte ma solo 4 bit vengono utilizzati e i restanti 4 bit rappresentano il complemento (valore invertito) del valore reale. Questo permette di avere un piccolo controllo sul valore del PID stesso. I vari valori che può assumere il PID sono raggruppati in quattro diverse categorie:

- Categoria Token: specifica se si ha a che fare con un tipo di transazione IN (1001), OUT (0001), *Start of Frame* SOF (0101) o SETUP (1101).

- Categoria Data: assume i valori, 0011, 1011, 0111, 1111 a seconda del pacchetto dati che si sta inviando.

- Categoria *Handshake*: assume i valori ACK (0010), NAK (1010), STALL (1110) NYET (0110).

- Categoria *Special*: assume i valori PRE (1100 da Host), ERR (1100 da Hub), SPLIT (1000), PING (0100) ed EXT (0000).

Per i dettagli di ogni voce si rimanda alle specifiche USB 2.0, ma come detto i dettagli di

una trasmissione a livello dei campi sopra citati viene gestita tutta a livello Hardware, per cui se si deve progettare un sistema a microcontrollore non ci si deve preoccupare di questi dettagli a meno di non dover risolvere problemi e dover fare un Debug a basso livello analizzando i dati sul bus.

Da quanto appena detto si capisce che sul bus oltre ai dati veri e propri vengono inviati molti altri byte. In particolare in gergo si dice che è presente un certo *over head* derivante dalla trasmissione delle informazioni aggiuntive oltre ai dati stessi. Per questa ragione le velocità effettive che si possono raggiungere nelle varie modalità supportate dalle specifiche USB sono in realtà più basse. I valori effettivi che si possono raggiungere, considerando il bus libero e la periferica in modalità Bulk, sono:

- SuperSpeed :400MB/s
- High Speed : 53MB/s
- Full Speed : 1.2MB/s
- Low Speed: 800B/s

Specifiche meccaniche del protocollo USB

Le specifiche USB sono state realizzate sin dall'inizio con l'idea di permettere dal lato dell'utilizzatore un'esperienza d'uso semplice. Come abbiamo visto, molte delle periferiche utilizzate con i primi PC avevano connettori simili che potevano creare anche errori di connessione invertendo gli stessi ed inserendoli su connettori dedicati per applicazioni diverse. Per tale ragione l'USB IF ha progettato per il protocollo USB un nuovo connettore che evitasse di poter essere erroneamente collegato ad altri connettori esistenti sul mercato e che non permettesse errori di inversione. I connettori realizzati per il supporto delle specifiche USB1.x sono stati nominati Type A (Standard A) e Type B (Standard B). In particolare dal lato Host (PC) è presente il Type A mentre dal lato del dispositivo (periferica) è presente il Type B. Il Type A come il Type B si dividono a loro volta in connettore maschio e femmina, o come li definisce l'USB IF *plugs* e *receptacles*. Allo stesso modo delle specifiche USB anche le specifiche relative ai connettori si sono evolute con il tempo, in particolare l'USB IF, mantenendo il resto delle specifiche invariate, ha introdotto altri connettori ufficiali al fine di poterne permettere l'utilizzo anche su periferiche più piccole. Questo ha portato alla definizione dei nuovi connettori di tipo Mini A, Micro A, Mini B e Micro B. In Figura 2[1] sono riportati i dettagli dei connettori USB tipo *Plug*, per ognuno di essi è presente il rispettivo *Receptacle* (non mostrato in Figura)

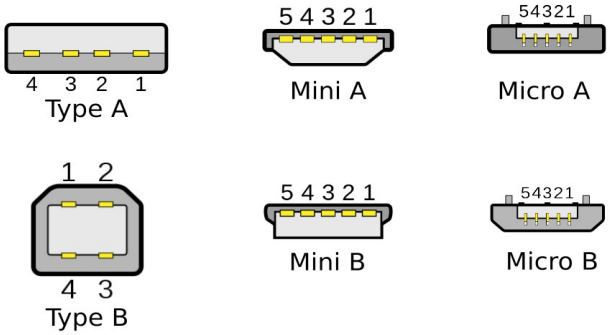

Figura 2: *Famiglia di connettori usati nei cavi USB 2.0.*

[1] L'immagine è una rielaborazione delle immagini disponibili al sito https://en.wikipedia.org/wiki/USB.

L'esperienza fornita con il tempo e l'utilizzo dei rispettivi connettori da parte degli utilizzatori finali, ha portato alcuni di essi ad non essere più ufficialmente supportati, sebbene continuino ad essere utilizzati. La presenza dei vari connettori fornisce un'ampia gamma di combinazioni che possono essere utilizzate a seconda delle proprie esigenze progettali. Naturalmente decidendo di utilizzare una combinazione tipica permette di acquistare cavi a basso costo. In particolare i cavi USB fanno quasi sempre uso del connettore Type A dal lato Host al fine di potersi collegare ad un PC, che rappresenta appunto il collegamento tipico. Dal momento che l'USB fornisce anche l'alimentazione alla periferica collegata alla porta, l'USB IF ha deciso di rendere non conformi i cavi con connettori Type A su entrambe le estremità, al fine di evitare possibili cortocircuiti tra due Host.

L'introduzione delle specifiche USB 3.0 ha portato qualche rompicapo ai progettisti USB IF che hanno dovuto specificare un nuovo connettore che fosse ancora compatibile con le precedenti specifiche USB 2.0. Questo ha portato ai nuovi connettori per il supporto delle specifiche SuperSpeed (SS) e SuperSpeed+ (SS+) riportati in Figura 3[2]. In particolare la Figura mette in evidenza i connettori tipo *Plug*, ma per ognuno di essi è presente il rispettivo *Receptacle*.

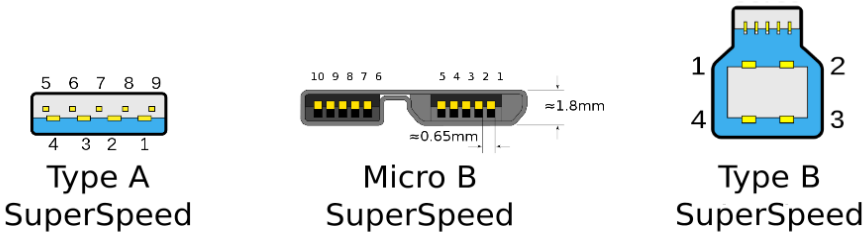

Figura 3: *Famiglia di connettori Plug usati nei cavi USB 3.0.*

In particolare un connettore Micro B per il supporto USB 2.0 può collegarsi al nuovo connettore USB 3.x continuando ad operare secondo le specifiche USB 2.0.
Allo stesso modo il connettore USB 3.x Type A può collegarsi ad un connettore USB 2.0 Type A receptacle, operando secondo le specifiche USB 2.0.

Lo standard USB specifica oltre alla dimensione dei singoli connettori anche la posizione del logo USB al fine di mostrare la conformità del cavo e facilitare la connessione del cavo stesso. Onestamente la posizione del logo sebbene sia stata posizionata per facilitare la connessione, rimane di uso poco pratico visto che ogni connettore se non allineato correttamente non riesce comunque a collegarsi.

Pinout dei connettori USB

Sebbene non abbiamo ancora trattato gli aspetti elettrici relativi al protocollo USB abbiamo già detto che lo standard prevede che l'alimentazione sia fornita dalla porta USB dell'Host, per cui due pin di ogni connettore USB sono dedicati al + e il – dell'alimentazione. Altre linee devono essere dedicate anche per la trasmissione dei dati per cui anche qui alcuni pin devono essere dedicati alle linee dati. Questi pin variano però in base alle specifiche, infatti l'USB 1.x e USB 2.0 fanno uso di una trasmissione Half

[2] L'immagine è una rielaborazione delle immagini disponibili al sito https://en.wikipedia.org/wiki/USB.

Duplex per cui per trasmettere e ricevere i dati si ha un solo canale di comunicazione che deve essere condiviso in maniera opportuna al fine di poter sia trasmettere che ricevere dati. Il canale in questione è di tipo bilanciato, ovvero di tipo differenziale, per cui sono presenti due linee dati nominate D+ e D-.
Lo standard USB ha introdotto come estensione alle specifiche USB 2.0 lo standard USB On The Go che permette ad un dispositivo con risorse limitate di poter operare come Host oltre che come Device, senza dover supportare tutte le specifiche richieste normalmente ad un Host. Questo standard ha posto l'esigenza del pin aggiuntivo nominato ID. In particolare a seconda che un dispositivo stia operando come Host o come Device deve essere o meno collegato a massa. Maggiori dettagli sull'USB On The Go sono riportati di seguito. In Tabella 6 è riportato un riassunto dei relativi connettori e pinout.

Pin	Type A	Type B	Mini A e B	Micro A e B	Micro B SS	Type C	
1	Vbus	Vbus	Vbus	Vbus	Vbus	GND	GND
2	D-	D-	D-	D-	D-	SSTX+1	SSRX+1
3	D+	D+	D+	D+	D+	SSTX-1	SSRX-1
4	GND	GND	ID	ID	ID	Vbus	Vbus
5			GND	GND	GND	CC1	SBU2
6					SSTX-	D+1	D+2
7					SSTX+	D-1	D-2
8					GND	SBU1	CC2
9					SSRX-	Vbus	Vbus
10					SSRX+	SSRX-2	SSTX-2
11						SSRX+2	SSTX+2
12						GND	GND

Tabella 6: *Pinout dei connettori USB.*

Lo standard USB 3.x, al fine di raggiungere un maggior bit rate ha richiesto di avere linee dedicate per la trasmissione e ricezione al fine di ottenere una trasmissione Full Duplex. Le specifiche richiedono in particolare che ogni coppia di linea dati abbia una linea di massa per questo si parla anche non di Full Duplex ma anche di Dual Simplex. Le due linee di massa sono poi collegate assieme nel punto comune di massa (GND, pin 8)
Con l'introduzione dello standard USB 3.1 l'USB IF ha anche introdotto le specifiche di un nuovo connettore, nominato Type C, che effettivamente racchiude i principi che hanno spinto a definire le specifiche USB, ovvero la semplicità d'utilizzo. Il nuovo cavo Type C, come riportato in Figura 4[3], è simmetrico da ambo i lati, ovvero permette di essere collegato indifferentemente da un lato o dall'altro senza alcun problema. Il connettore Type C ricorda molto il cavo Apple Lightning. I dettagli sul pinout sono riportati in Tabella 6.

[3] L'immagine è una rielaborazione dell'originale disponibile al sito https://en.wikipedia.org/wiki/USB.

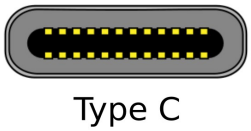

Figura 4: *Connettore Type C.*

I cavi che fanno uso del connettore Type C possono essere anche di tipo attivo, ovvero il produttore può inserire una memoria interna per contenere determinate informazioni di autenticità del prodotto e proprietà dello stesso.

Oltre ai connettori introdotti fin ad ora, supportatati da una specifica ufficiale da parte dell'USB IF, molti costruttori hanno introdotto connettori proprietari. I più comuni sono stati quelli da parte di produttori di cellulari che hanno creato cavi con USB plug Type A da un lato e connettore proprietario dall'altro. Questi cavi sono andati scomparendo solo dopo l'introduzione del connettore USB Micro B e la normativa Europea EN 62684:2010 che ha proposto che nel mercato Europeo i cellulari fossero ricaricabili per mezzo di un unico connettore, in particolare facendo uso del connettore USB. Questo ha permesso di limitare la generazione di rifiuti speciali, che richiederebbero secondo la normativa 2011/65/UE un trattamento speciale per la raccolta e lo smaltimento. La normativa prevede anche la possibilità di bypassare la regola qualora sia fornito un cavo adattatore tra USB e un connettore proprietario dal lato del cellulare, il quale permetterebbe comunque di utilizzare caricatori USB standard. Questa soluzione è stata in particolare adottata dalla Apple, la quale per i propri iPhone fornisce il connettore Apple Lightning.

Cavi USB

Oltre ai connettori appena descritti lo standard USB ha previsto delle specifiche anche per i cavi usati per collegare i vari connettori. Nelle specifiche USB 1.0 le specifiche dei cavi erano in termini di lunghezza degli stessi, in particolare:

- 3m max. per Low Speed
- 5m max. per Full Speed

Sebbene l'attenuazione offerta da un cavo sia proporzionale alla frequenza del segnale che lo percorre, in modalità Full Speed il cavo può essere più lungo. Le ragioni sono semplicemente legate al fatto che in realtà il cavo USB 1.x Low Speed è specificato in maniera da poter essere più economico, in particolare non richiede lo schermaggio né della coppia di cavi di dati né del cavo totale. Nel caso bisogna supportare la modalità Full Speed, il cavo che include la coppia dati e alimentazione, deve essere schermato.

La coppia di cavi utilizzata per la linea dati è di tipo *Twisted Pair*, ovvero un cavo intrecciato, come riportato in Figura 5.

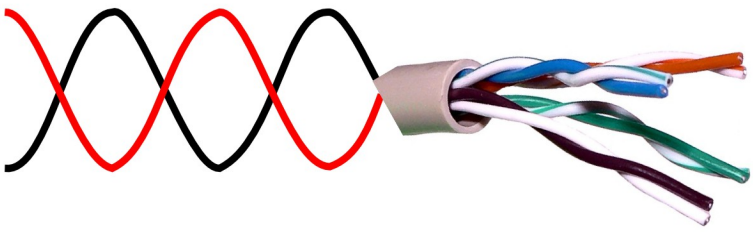

Figura 5: *Esempio di cavo Twisted Pair.*

I cavi intrecciati in maniera simmetrica offrono particolare immunità al rumore esterno, in particolare alle radiazioni elettromagnetiche a bassa frequenza, mentre lo schermo conduttivo presente sul cavo USB offre maggiore schermaggio ad alte frequenze. La ragione per cui un cavo schermato offre una buona immunità al rumore è dovuta al fatto che espone una ridotta superficie al campo elettromagnetico che si sta accoppiando con i cavi. Se si pensa alla differenza tra cavi intrecciati e non intrecciati, può risultare complicato pensare al fatto che il cavo intrecciato offra una superficie ridotta rispetto a cavi non intrecciati, infatti mentalmente si possono pensare le due tipologie di cavi piuttosto simili e con cavi affiancati l'un l'altro in maniera analoga. Guardando meglio i dettagli di Figura 6 è possibile notare che un campo elettromagnetico che investe i cavi si trova ad accoppiarsi con tante piccole aree piuttosto che una singola area come nei cavi non intrecciati. Ogni singola area permette di generare per induzione elettromagnetica una corrette indotta sul cavo di segno che viene a dipendere dall'orientamento dell'area. Dal momento che ogni anello composto da cavo intrecciato ha un'area di tipo alternata, la corrente ad ogni anello ha segno opposto, per cui si cancella con quello dell'area adiacente. Per questa ragione ai fini del rumore totale, ovvero area totale con cui si accoppia il campo elettromagnetico, si parla di area equivalente ridotta, visto che il rumore totale viene a dipendere solo dalle piccole differenze di aree che generano un rumore che non si cancella.

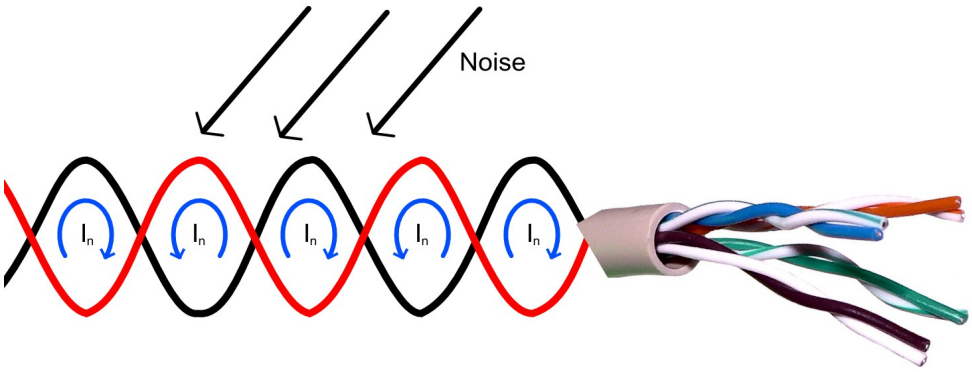

Figura 6: *Cancellazione del rumore nei cavi Twisted.*

Da quanto appena detto si capisce che nel caso di un cavo non intrecciato si ha che l'area equivalante risulta molto più ampia, visto che si ha un solo anello. L'esigenza di rendere una trasmissione immune al rumore è legata semplicemente al fatto che il rumore potrebbe generare un errore nell'interpretazione del valore di un bit. Come descritto in precedenza, l'USB IF ha previsto altre caratteristiche al fine di rendere la trasmissione USB affidabile, alcune delle quali sono a livello di scelte Hardware mentre altre a livello Software, ovvero per mezzo del riconoscimento di eventuali dati errati per i quali viene richiesta una nuova trasmissione dei dati.

Le specifiche dei cavi si è andata generalizzando a partire dalle specifiche USB 2.0. Infatti l'USB IF non ha più specificato una lunghezza massima dei cavi ma ho posto dei limiti sull'attenuazione offerta dai cavi alle diverse frequenze operative dei vari standard. In questo modo in base alla qualità del cavo e alla velocità di propagazione dei segnali elettrici, è eventualmente possibile usare cavi più o meno lunghi. Ciononostante la lunghezza di un massimo di 5m è consigliata anche per la modalità High Speed. In particolare i cavi USB 1.x che supportano la modalità Full Speed, possono essere anche

utilizzati per la modalità High Speed.

Per la modalità SuperSpeed e SuperSpeed+ i cavi sono normalmente di 3m sebbene aumentando la qualità dei degli stessi sia possibile anche avere lunghezze maggiori, normalmente a scapito del diametro dei cavi e della rigidità del cavo. I cavi per il supporto delle specifiche USB 3.x hanno aggiunto alcuni vincoli di schermaggio, in particolare ogni coppia di linea dati deve essere schermata e avere la propria linea di massa. Oltre a questo il cavo deve avere una schermatura globale come nel caso delle specifiche Full Speed. Dal momento che il bus USB supporta fino a 5 Hub, si possono raggiungere lunghezze superiori a 5m, infatti tra ogni Hub si può far uso di un cavo della lunghezza massima consentita.

Sebbene l'USB IF abbia cercato di portare ordine nel mondo dei connettori, tra le combinazioni possibili ci sono alcune vietate. Oltre al cavo con connettore plug Type A-plug Type A precedentemente visto, l'USB IF vieta anche il connettore plug Type A da un lato e Type A receptacle dall'altro. Questa tipologia di cavi permetterebbe infatti di estendere la lunghezza totale del bus oltre specifica. Sebbene l'USB IF non abbia definito alcuna specifica per tale tipologia di cavi, in commercio è possibili trovarli, specialmente di breve lunghezza, frequentemente utilizzati qualora la propria periferica non si colleghi facilmente al PC, soprattutto nel caso dei *Notebook*, che stanno diventando sempre più piccoli. Oltre ai cavi brevi con plug Type A e Type A receptacle sono presenti anche cavi da 5m. In questo caso bisogna fare particolare attenzione e vedere se sono di tipo attivo o meno. Quelli di tipo attivo contengono un Hub ad una porta, per cui collegarli al proprio PC permette effettivamente di allungare, secondo specifica USB la lunghezza massima del cavo. Altri sono semplici cavi e costano molto meno, per cui si è frequentemente attratti da quest'ultimi. La differenza pratica tra i due tipi di cavi è che nel caso di quelli attivi si possono collegare in cascata fino a 5 senza problemi, mentre nel caso di quelli non attivi se si collega un qualunque altro cavo che allunga oltre 5m la lunghezza totale, si potrebbero avere problemi, in particolare collegare due cavi non attivi da 5m è sconsigliato oltre che non supportato da specifica.
Se non si è certi di avere un cavo attivo o meno, lo si può scoprire facilmente collegandolo al PC. Cavi attivi contengono come detto un Hub, per cui quando vengono collegati al PC richiedono l'inizializzazione dello stesso (enumerazione) e l'installazione del relativo driver, che in generale è contenuto nel sistema operativo.

Sebbene molti dei cavi e connettori USB 3.x siano di colore blue, questo è solo un colore consigliato. Cavi che rispettano le specifiche USB 3.x possono essere anche di altro colore. Non è insolito trovare connettori sui *Notebook* che abbiano colore nero come i classici connettori USB, ma l'icona vicino al connettore è comunque secondo specifica con l'icona USB e la SS per denotare il supporto SuperSpeed.

USB On The GO

Abbiamo visto che secondo le specifiche USB 2.0 l'arbitraggio del bus spetta all'Host, il quale è responsabile dell'inizializzazione di ogni Device e Hub eventualmente collegati ad una porta USB. In particolare ogni Device fornisce dati all'Host solo su richiesta dello stesso. In questo contesto le specifiche USB 2.0 sono state estese per mezzo delle specifiche USB OTG (*On The Go*). Le nuove specifiche sono state introdotte dall'USB IF nel 2001 e si sono estese con gli anni all'attuale versione 2.0-1.1 rilasciata nel 2012.
Le specifiche USB OTG permettono ad un dispositivo di comportarsi come un semplice Host al quale possono essere collegati dei dispositivi. I microcontrollori che supportano le specifiche OTG permettono non solo di essere collegati ad un Host come dispositivi, come fin ora spiegato, ma permettono anche di poter collegare un dispositivo al microcontrollore stesso, come per esempio una memoria USB. Un esempio tipico di dispositivo OTG è il cellulare. Infatti collegando questo al PC si comporta come Device, ma al tempo stesso è possibile collegare al cellulare anche dei dispositivi esterni come per esempio una penna USB, una tastiera, una telecamera e altro, in qual caso il cellulare si comporta come Host.

Un dispositivo che supporti la modalità OTG non può operare in contemporanea come Host e come Dispositivo, ma eventualmente se collegato con un altro dispositivo OTG, può cambiare ruolo a seconda delle esigenze, facendo uso del protocollo HNP (*Host Negotiation Protocol*). Nel caso in cui un dispositivo OTG non debba supportare il collegamento con un altro dispositivo OTG, non è richiesto che sia supportato il protocollo HNP. Il riconoscimento dell'Host e Device avviene per mezzo della linea ID presente nei cavi e connettori Mini/Micro A e B. In particolare l'Host (nominato A) ha il pin ID collegato a massa mentre il Dispositivo (nominato B) lo ha aperto o collegato a massa per mezzo di un resistore maggiore di 1MΩ. Le specifiche OTG sebbene permettano ad un dispositivo di operare come Host, consente che le sue funzioni siano più limitate al fine di avere un sistema meno complesso e costoso. In particolare un dispositivo OTG non deve supportare gli Hub e la modalità High Speed e Low Speed. Il loro supporto è una scelta progettuale ma non è obbligatorio. Diversamente da un Host USB un dispositivo OTG deve poter supportare un minimo di corrente di soli 8mA contro i 100mA di un Host normale. Sebbene questo possa alleggerire le esigenze relative ai convertitori DC-DC utilizzati, dal momento che molti dei dispositivi USB possono usufruire di un minimo di 100mA durante la fase di enumerazione, non è insolito che anche i dispositivi USB OTG supportino questa corrente minima. Al fine di permettere ad un dispositivo OTG di poter risparmiare potenza è concesso disattivare Vbus qualora si voglia disattivare un dispositivo esterno. Ciononostante è presente il protocollo SRP (*Session Request Protocol*) che permette ad un dispositivo di richiedere nuovamente tensione.

Specifiche Power del protocollo USB

Gli aggiornamenti delle specifiche USB hanno coinvolto diversi aspetti. Per quanto riguarda gli aspetti relativi al Power è bene trattare le varie versioni separatamente visto che diverse versioni hanno specifiche diverse.

Specifiche Power USB 1.x e 2.0

Le specifiche USB definisce due famiglie di dispositivi nominate *High Power Device* e *Low Power Device*. In particolare i primi possono assorbire fino a 500mA mentre i secondi fino a 100mA. In entrambi i casi è comunque richiesto che alla connessione del dispositivo la corrente assorbita dal dispositivo non sia maggiore di 100mA. Affinché un dispositivo possa assorbire maggiore corrente è necessario che avvenga l'enumerazione dello stesso e che sia letto il *Configuration Descriptor*. In particolare il parametro bMaxPower contiene il valore della massima corrente. Tale parametro è composto di un solo byte e il valore di corrente è espresso in multipli di 2mA. In commercio sono però presenti diversi dispositivi che permettono di assorbire o cercano di assorbire direttamente 500mA dalla porta USB senza supportare alcuna enumerazione del dispositivo. Basti per esempio pensare agli hard-disk che forniscono cavi ad Y con doppio connettore plug Type A per poter essere collegato a due porte e poter usufruire di maggior corrente, o anche ai ventilatori o luci che possono essere collegati direttamente alla porta USB, e possono facilmente erogare correnti maggiori di 100mA.
I livelli di corrente supportati pongono dei vincoli importanti ai dispositivi USB, in particolare impone agli stessi un eventuale alimentazione esterna nel caso in cui si vogliano supportare corrente maggiori di 500mA. Naturalmente rimanere nei limiti dei 500mA permette di risparmiare notevoli costi dal lato del sistema, visto che si possono risparmiare trasformatori o convertitori AC-DC per convertire 220V a 5V con la necessaria corrente per alimentare il dispositivo. In particolare un dispositivo potrebbe anche prelevare parte della corrente dalla porta USB e una parte da un alimentatore esterno. In questo contesto si definiscono due classi di dispositivi:

- Bus Powered
- Self Powered

I dispositivi *Bus Powered* sono quelli che rientrano nelle specifiche USB e possono prelevare la corrente direttamente dal bus mentre i *Self Powered* sono quelli con alimentazione propria ma ai quali non è vietato prelevare anche una parte della corrente dal Bus, in accordo con le specifiche USB

La tensione disponibile sul bus ha un valore nominale di 5V, ma a causa della lunghezza dei cavi e della corrente che deve essere fornita dal dispositivo Host o Hub, ha una tolleranza piuttosto ampia e varia da un minimo di 4.4V a un massimo 5.25V. A supporto delle specifiche Type C e delle massime correnti, le specifiche sono state aggiornate con un valore di tensione pari a 5.50V, per cui Host e Hub precedenti alla specifica *"ECN USB 2.0 VBUS Max Limit"*, hanno una tensione massima pari a 5.25V mentre quelle successive, potrebbero avere una tensione massima pari a 5.50V. Per quanto riguarda il valore minimo, sebbene sia 4.4V, bisogna distinguere due casi, ovvero:

- Low Power Port
- High Power Port

Una porta USB si definisce *Low Power* qualora fornisca correnti massime da poter alimentare solo dispositivi *Low Power* (100mA max.), mentre una porta si definisce *High Power* qualora possa fornire correnti a dispositivi *High Power* (500mA max.). I limiti di tensione minimi per questi dispositivi sono rispettivamente:

- Low Power Port: 4.75V
- High Power Port: 4.40V

Il valore differisce per tenere conto della diversa caduta di tensione in funzione della corrente del dispositivo. Dai valori di tensioni ora descritti, si capisce che nel caso in cui si debba alimentare il proprio dispositivo con un'alimentazione pari a 3.3-3.6V, tipici per molti microcontrollori a 16bit, non si hanno problemi, infatti basta usare un LDO per convertire i livelli di tensione da Vbus al valore necessario. Nel caso in cui si necessiti però di alimentare un dispositivo a 5V la cui tolleranza sulla tensione sia del 5%-10%, ovvero compresa tra 4.5V e 5.5V, si hanno problemi sul valore minimo. Sebbene 4.4V è quasi 4.5V, volendo realizzare un dispositivo in specifica è necessario far uso di convertitore DC-DC complesso, come per esempio basati su architetture buck-boost o SEPIC. Tali configurazioni permetterebbero infatti di ottenere 5V in uscita partendo anche da tensioni d'ingresso inferiori a 5V. Sistemi di alimentazione troppo complessi potrebbero avere qualche problema da un punto di vista dei picchi di corrente in ingresso visto che secondo le specifiche USB è necessario avere una capacità di carico sul dispositivo compresa tra 1-10µF. In particolare il dispositivo elettronico, dal punto di vista dei consumi deve poter essere modellizabile con un resistore da 44Ω in parallelo con una capacità compresa tra 1-10µF. Sebbene sia presente un limite di 10µF questa capacità è quella che viene vista dall'Host o Hub direttamente sulle linee Vbus. Valori maggiori possono essere utilizzati, se necessari per l'applicazione, qualora il sistema fornisca la possibilità di limitare la corrente di picco entro le specifiche USB e garantendo una caduta di tensione su Vbus limitata (per maggiori dettagli si faccia riferimento alle specifiche USB ufficiali).

I dispositivi USB collegati ad una porta USB devono supportare la modalità *Suspend State* durante la quale è richiesto che vengano ridotti i consumi a 2.5mA come media in 1s. Tale corrente include la corrente dell'intero sistema, inclusa quella necessaria per i resistori di pull-up da 1.5KΩ, che come vedremo a breve sono necessari per determinare il bit rate della periferica (Low Speed o Full Speed).

La versione originale delle specifiche USB imponeva che la corrente in *Suspend State* fosse di soli 500µA. Dal momento che molti dispositivi avevano difficoltà a raggiungere tali livelli, l'USB IF ha aumentato tali valori a 2.5mA tramite l'ECN *"Suspend Current Limit Changes"*. In particolare gli Hub possono anche erogare 12.5mA permettendo di supportare fino a 4 porte in Suspend State.

Un dispositivo deve entrare in *Suspend Mode* nel caso in cui non ci sia attività sul bus per oltre 3ms. Nel caso si faccia uso di microcontrollori e Stack forniti dal produttore, lo stato *Suspend State* viene normalmente identificato e reso disponibile dallo Stack stesso, ma

viene normalmente richiesto l'intervento dell'applicazione al fine di poter effettivamente ridurre i consumi del sistema nei limiti dei 2.5mA. Infatti ogni sistema Hardware potrebbe avere particolari esigenze e limitare i consumi potrebbe richiedere il disattivare particolari periferiche interne o esterne al microcontrollore.

I sistemi collegati ai *Notebook* alimentati a batteria potrebbero essere costretti ad entrare in *Suspend State* al fine di ridurre i consumi. Particolari periferiche come Mouse e Tastiere, potendo svegliare il PC da una modalità *Sleep*, dovrebbero essere impostati in maniera tale da avere sempre disponibile la tensione di alimentazione sul bus.

Oltre alla modalità *Suspend*, le specifiche USB supportano anche la modalità *Sleep*, che permette al dispositivo finale di ridurre i consumi. In questa modalità non è comunque obbligatorio ridurre i consumi e non necessariamente un dispositivo deve supportarla.

Specifiche Power USB 3.x

Le specifiche USB 3.x arricchiscono le possibilità offerte dalla porta USB per quanto riguarda la massima corrente che può essere fornita. Pur mantenendo la divisione di *High Power Device* e *Low Power Device* la prima categoria supporta fino a 900mA mentre la seconda fino a 150mA.

Un dispositivo conforme alle specifiche USB 2.0 collegato ad una porta USB 3.x deve continuare a rispettare le specifiche USB 2.0 ovvero rispettivamente 100mA (*Low Power Device*) e 500mA (*High Power Device*).

Per poter utilizzare correnti superiori a 150mA è necessario che il dispositivo venga propriamente inizializzato durante la fase di enumerazione e che il valore di corrente richiesto dal dispositivo sia scritto all'interno del *Configuration Descriptor*. In particolare il parametro di un byte bMaxPower contiene il valore della corrente espressa in multipli di 8mA. Le specifiche USB 3.x allo stesso modo delle specifiche USB 2.0 richiedono che un dispositivo entri in *Suspend State* qualora non ci siano attività sul bus. Oltre a tale modalità le specifiche USB 3.x introducono altre possibilità per permettere il risparmio di corrente, particolarmente utili per dispositivi alimentati a batteria. Senza entrare nei dettagli, tali modalità sono:

- **U0**: Modalità normale in cui avviene la comunicazione tra Host e i dispositivi.
- **U1**: Modalità a basso consumo con rapida transizione alla modalità operativa U0.
- **U2**: Modalità con ulteriore riduzione dei consumi in cui il dispositivo elettronico può anche disattivare il clock di sistema a scapito di un più lungo tempo di transizione per tornare in modalità operativa U0.
- **U3**: Tale modalità è l'equivalente della modalità *Suspend State*.

Protezione e limiti di corrente

Correnti operative massime pari a 500mA e 900mA, se si considera che ogni Hub può avere più connettori, tipicamente 4 o più, può richiedere da parte dell'Host il dover fornire potenze superiori a 20W. Come vedremo a breve la porta USB, tramite alcune estensioni delle specifiche può essere utilizzata anche per caricare le batterie dei sistemi esterni, estendendo le correnti fino a 5A. In queste situazioni le potenze in gioco sono certamente tali per cui avere delle precauzioni risulta di particolare importanza.

Sebbene le correnti in gioco possano essere elevate è bene mettere in evidenza che il fatto che un dispositivo richieda correnti maggiori di 100mA o 150mA, non necessariamente il sistema Host deve soddisfare tale richiesta. In particolare se si prende un *Notebook* la cui batteria si sta scaricando e si collega un sistema alla porta USB che richiede 500mA, probabilmente la richiesta non verrà accordata. Allo stesso modo se prendiamo un caso di un Hub collegato ad una porta USB per estenderla a 4 porte, se tale Hub è *Bus Powered* vuol dire che le 4 porte hanno a disposizione al massimo 500mA da condividere, in particolare la somma di tutte le correnti deve essere inferiore alla corrente massima della singola porta USB alla quale è collegato l'Hub. Il controllo della corrente viene fatta durante la fase di enumerazione e verifica delle esigenze delle singole periferiche.

Il controllo delle correnti appena descritto è di natura Software e fa uso semplicemente delle informazioni contenute nel *Configuration Descriptor*. Le specifiche USB richiedono che sia presente anche un controllo Hardware della corrente, limitando la stessa qualora si eccedano i limiti. In particolare l'*Host Controller* deve essere notificato di eventuali anomalie e il sistema operativo può prendere decisioni di conseguenza. Il limite di corrente a seconda del dispositivo di protezione utilizzato potrebbe mantenere la tensione costante e limitare la corrente o mantenere la corrente al valore massimo consentito e limitare la tensione. Altre tecniche fanno uso della modalità *Hichup* (singhiozzo) in cui tensione e corrente sono azzerate per un certo periodo di tempo e ciclicamente viene fatto un tentativo di verifica dello stato di anomalia, ripristinando la tensione di uscita e verificando il limite di corrente.

Al verificarsi di un evento di corrente eccessiva ed entrata in protezione dell'uscita, molti chip di protezione forniscono un pin di FAULT per segnalare l'anomalia all'Host e permettere di soddisfare le specifiche USB. Oltre al limite della corrente massima operativa, l'USB IF impone anche dei limiti di corrente all'inserimento del dispositivo ovvero di picco.

Oltre alle considerazioni sul limiti di corrente, qualora si progetti un sistema USB che sia *Self Powered*, ovvero con alimentazione esterna, è necessario tenere a mente che questo non deve fornire alimentazione alla porta USB qualora questa non sia attiva. Questo in particolare è imposto anche ai resistori di pull-up utilizzati per determinare la modalità attiva. Questo significa che per evitare di attivare i resistori di pull-up qualora la porta USB non sia alimenta e il sistema sia alimentato esternamente, è sempre necessario prevedere un controllo della tensione sull'USB Vbus.

La porta USB per la ricarica delle batterie

Sebbene al tempo della stesura delle specifiche del protocollo USB non fosse stato preso in considerazione che la porta USB potesse essere utilizzata per ricaricare le batterie di dispositivi portatili come i cellulari, tale pratica è divenuta piuttosto comune. Questo ha portato l'USB IF ad estendere le specifiche USB al fine di utilizzare la porta USB anche per ricaricare le batterie, permettendo, qualora le risorse dell'Host lo permettano, di fornire una corrente superiore a 500mA per l'USB 2.0 o 900mA per l'USB 3.x. In particolare l'estensione va sotto il nome *Battery Charging Specification* (BC). Attualmente sono presenti diverse versioni che permettono di estendere la corrente di ricarica fino a 1.5A (BC 1.1) e 5A (BC 1.2). E' bene mettere in evidenza che anche in questo caso sebbene un sistema Host che supporti lo standard *Battery Charging*, non necessariamente fornirà la corrente massima delle specifiche. Le condizioni operative del sistema in cui risiede l'Host potrebbero determinare dei vincoli sul processo di carica, limitando la corrente massima che viene effettivamente resa disponibile. Il protocollo su cui si basa lo standard BC è piuttosto semplice e consiste nell'avere le linee D+ e D- collegate tra loro per mezzo di un resistore di 200Ω e si pongono le linee D+ e D- a valori di tensioni specifiche al fine di permettere sia all'Host che al dispositivo di impostare la porta USB secondo lo standard *Battery Charging*. Le specifiche BC sono state introdotte dall'USB IF piuttosto in ritardo, tanto da spingere sia Apple che Samsung a definire un proprio standard per permettere ad un dispositivo di richiedere una corrente di carica maggiore delle specifiche USB, senza dover necessariamente avviare l'enumerazione del dispositivo sotto carica. In particolare questo a portato allo sviluppo di circuiti integrati come il TPS2511 (USB *Dedicated Charging Port Controller*) che permette di supportare l'identificazione del dispositivo sotto carica e di limitare la corrente proteggendo l'Host.

Specifiche elettriche del protocollo USB

Il protocollo USB definisce per la trasmissione dei dati un bus seriale con linee differenziali. In particolare in base alla versione del protocollo si ha una trasmissione seriale *Half Duplex* (USB 1.x e 2.0) o *Dual Simplex* (USB 3.x) asincrona, ovvero senza trasmissione diretta del segnale di clock. Il *bit rate* come abbiamo già visto, varia in base alla modalità utilizzata. Vediamo i dettagli delle specifiche elettriche in base alla versione e del bit rate supportato.

Specifiche elettriche Low Speed e Full Speed

Gran parte dei microcontrollori, avendo risorse limitate supportano prevalentemente lo standard Low Speed e Full Speed. In particolare il *bit rate* viene a variare in base allo standard come anche la tolleranza richiesta alla frequenza del clock. I dettagli sono riportati nella Tabella 7.

Standard	Bit Rate	Clock Accuracy
Low Speed	1.5Mbit/s	1,50%
Full Speed	12Mbit/s	0,25%

Tabella 7: *Specifiche del clock di sistema.*

Utilizzando la modalità Low Speed, grazie al fatto che l'accuratezza del clock non è troppo elevata, è possibile utilizzare anche un oscillatore RC, spesso integrato anche all'interno dei microcontrollori. Per supportare la modalità Full Speed è invece necessario un cristallo, il quale, sebbene abbia un costo di poche decine di centesimi, rappresenta un costo aggiuntivo del sistema. In base alla modalità utilizzata, ogni bit inviato sulla linea, differisce dal punto di vista dello *Slew Rate* (rapidità dei fronti di salita e discesa). In particolare la modalità Low Speed ha dei fronti rallentati rispetto alla modalità Full Speed. Il fatto di avere dei fronti di salita e discesa meno rapidi permette di ridurre le variazioni del flusso di corrente ovvero le radiazioni elettromagnetiche, che sono appunto legate ad all'accelerazione di cariche elettriche. Questa scelta permette di utilizzare dei cavi non schermati, ovvero più economici, nel caso della modalità Low Speed.

I bit trasmessi sul bus USB sono di tipo differenziale, in particolare il bus è composto da due linee elettriche nominate D+ e D-, oltre a massa e Vbus. Un bit 1 è trasmesso quando D+ è positivo rispetto a D-, mentre un bit 0 è trasmesso quando D- è positivo rispetto a D+. I bit ora descritti non rappresentano in realtà il bit reale che si vuole trasmettere. Per giungere a questo bisogna introdurre un livello di astrazione aggiuntivo, ovvero il simbolo J e il simbolo K. La corrispondenza tra il bit sulla linea del bus e il simbolo J e K viene a dipendere dallo standard, come riportato in Tabella 8.

Bit sul Bus	Low Speed	Full Speed
0	J	K
1	K	J

Tabella 8: *Equivalenza tra bit sul bus e simboli J e K.*

Questa astrazione serve poiché un bus nello stato di *Idle*, viene a sua volta a dipendere dalla modalità di trasmissione. In particolare l'Host deve avere sulle linee D+ e D- dei resistori di *pull-down* da 15KΩ ±5% verso massa mentre il dispositivo USB deve avere rispettivamente o un resistore da 1.5KΩ ±5% collegato tra D- e 3.3V per indicare la modalità Low Speed, o tra D+ e 3.3V per indicare la modalità Full Speed. Uno *Start of Packet* (SOP), ovvero inizio di un pacchetto, si verifica al passaggio dallo Stato di *Idle* al livello K. Avendo definito i simboli J e K e averli invertiti dal punto di vista del significato dei bit sul bus, permette di non dover pensare al valore elettrico del bus stesso quando si ha a che fare con le rispettive modalità Low Speed o Full Speed.
Introdotto il concetto del simbolo J e K si può giungere all'effettivo valore del bit trasmesso sul bus, tenendo conto della codifica NRZI (*Non Return to Zero Inverted*) utilizzata dallo standard USB 2.0. In particolare nella codifica NRZI si definisce come 0 una variazione di tensione sul bus, mentre come 1 un livello costante di tensione sul bus. Per cui ragionando in simboli quando si ha una variazione di simbolo si ha uno 0 mentre se il simbolo rimane lo stesso si ha un 1. Questa volta 0 e 1 sono effettivamente il valore del bit che vogliamo trasmettere, in particolare i bit sono inviati a partire dall'LSB (*Less Significant Bit*) all'MSB (*Most Significant Bit*). Oltre alla codifica NRZI il protocollo USB fa uso del *Bit Stuffing*, ovvero dell'inversione automatica del bit qualora ci sia un numero eccessivo di 1. Nel caso delle specifiche USB viene introdotto un bit di *Stuffing* (0) ogni 6 bit pari 1. Questo permette di inserire un numero di variazioni di bit sufficienti sul bus e garantire che ogni sistema possa rimanere sincronizzato in maniera opportuna. Infatti, il bus USB, non trasmettendo direttamente il clock, richiede che ogni dispositivo rimanga sincronizzato sul bit rate richiesto per la comunicazione. Per facilitare la sincronizzazione dei dispositivi, ogni pacchetto dati inizia anche con la trasmissione del campo SYNC, che nel caso delle modalità Low Speed e Full Speed è rappresentato dai simboli KJKJKJKK.
A scopo riassuntivo è bene riportare in una Tabella i vari segnali che è possibile trovare sul bus USB.

Segnale	Linea D+	Linea D-	Low Speed	Full Speed
1 Differenziale	Alto	Basso		
0 Differenziale	Basso	Alto		
Stato di Idle			D- connesso a +3.3V con resistore 1.5KΩ ±5%	D+ connesso a +3.3V con resistore 1.5KΩ ±5%
Simbolo J			0 Differenziale	1 Differenziale
Simbolo K			1 Differenziale	0 Differenziale
SE0 (Single Ended 0)	Basso	Basso		
SE1 (Single Ended 1)	Alto	Alto		
SOP (Start of Packet)			Da Idle a K	Da Idle a K
EOP (End of Packet)			SE0 per un bit, seguito da J	SE0 per un bit, seguito da J
SYNC			KJKJKJKK	KJKJKJKK
Reset			SE0 per almeno 10ms	SE0 per almeno 10ms

Tabella 9: *Tabella riassuntiva dei segnali sul bus USB.*

Modulo TX e RX per la modalità Low Speed e Full Speed

Le specifiche USB descrivono in maniera piuttosto chiara come devono essere realizzati i dispositivi d'interfaccia sul bus USB, sia dal lato Host che dal lato del dispositivo. In particolare per mezzo di diagramma ad occhi e specifiche temporali descrivono le specifiche a cui devono soddisfare. Dal lato del dispositivo l'hardware d'interfaccia sul bus (PHY), si presenta come in Figura 7 (schema a blocchi semplificato). Si può notare che bisogna supportare i resistori di pull-up sulle linee D+ e D-, che devono essere opportunamente attivati in base alle caratteristiche del sistema. In particolare un dispositivo che opererà sempre in una modalità potrebbe avere un solo resistore collegato in maniera fissa a 3.3V. Molti microcontrollori possiedono tali resistori anche integrati nella MCU stessa per cui bisogna effettivamente attivare il resistore d'interesse.

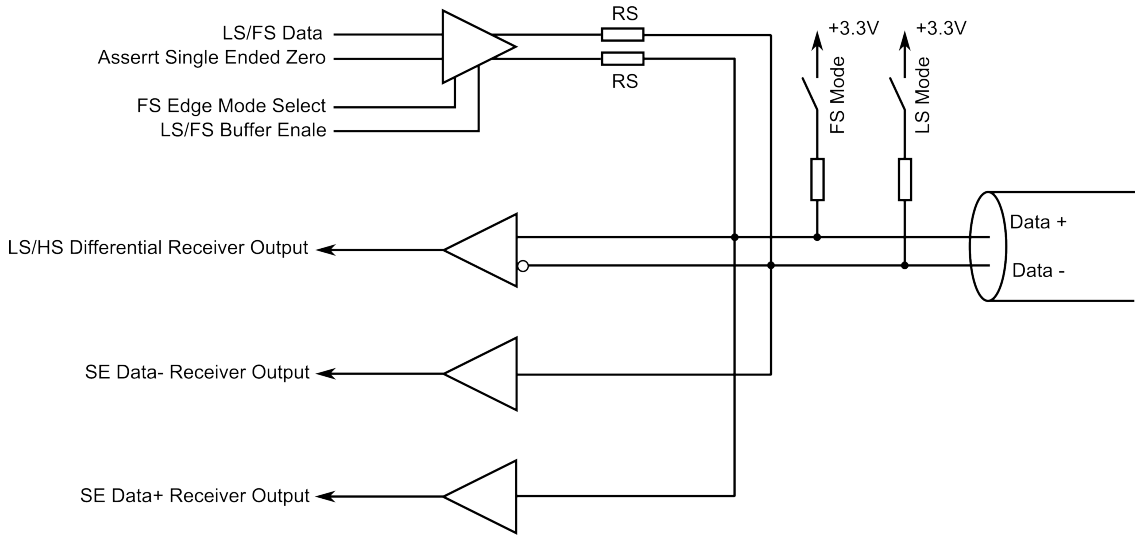

Figura 7: *PHY dal lato del dispositivo (Upstream).*

La presenza del resistore, oltre a definire lo stato di Idle, permette all'Host di rilevare se il dispositivo è collegato o meno. Il buffer di uscita presenta un'impedenza di uscita in serie (RS) pari a 36Ω, in aggiunta a quella interna delle uscite push-pull (piuttosto bassa). Tali resistori sono aggiunti appositamente all'impedenza del buffer e adattarla alla linea di trasmissione, limitando le riflessioni. Ciononostante la linea non è terminata dal buffer in recezione. Dal lato Host lo schema elettrico si presenta in maniera piuttosto speculare al dispositivo come riportato in Figura 8. I buffer *Single Ended* presenti sia dal lato del dispositivo che dal lato Host permettono di rilevare la tensione di linea ovvero le condizioni di Idle, SE0, SE1, Reset, Connessione, Disconnessione.

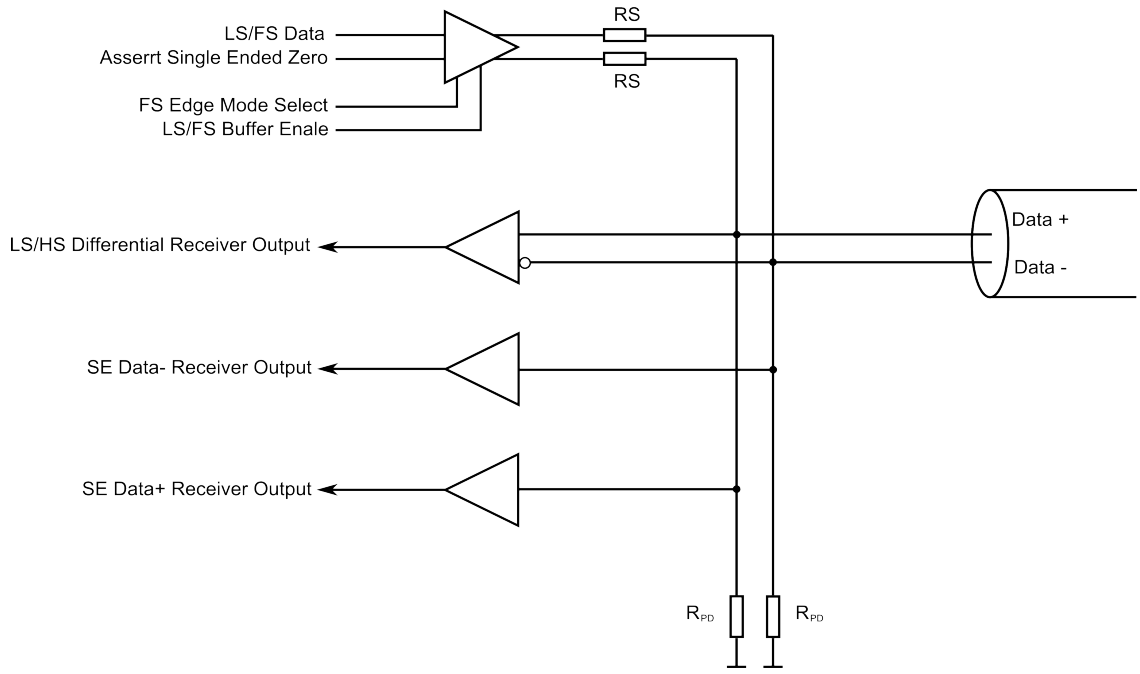

Figura 8: *PHY dal lato dell'Host (Downstream).*

Specifiche elettriche High Speed

Dal momento che i microcontrollori, per la natura delle risorse disponibili, non supportano le specifiche High Speed, sebbene supportino lo standard USB 2.0, non entrerò nei dettagli dello stesso. Per maggiori informazioni, qualora ci si trovi a gestire un microprocessore con prestazioni idonee a supportare la modalità High Speed, si faccia riferimento alla bibliografia.

Lo standard High Speed supporta un bit rate di 480Mbit/s. Tale frequenza è multipla di 12, per cui facendo uso di un PLL si può spesso usare uno stesso quarzo da 12MHz sia per supportare la modalità Full Speed che High Speed. L'aumento della frequenza, sebbene sia supportata dagli stessi cavi utilizzabili per la modalità Full Speed, richiede che la linea sia terminata al fine di evitare riflessioni sul bus. In particolare un dispositivo USB High Speed, deve presentarsi alla connessione come dispositivo Full Speed, ovvero con un resistore di pull-up sulla linea D+. Dopo la sequenza di *Handshake* che permette di impostare la modalità High Speed, il resistore di pull-up deve essere eliminato dalla linea e la trasmissione dei dati può avvenire al massimo bit rate. Al fine di facilitare la sincronizzazione del PLL interno, il campo SYNC possiede 15 sequenze KJ piuttosto che 3 seguite dai simboli KK. Il fatto che le linee non hanno più i resistori di pull-up, per determinare la disconnessione del cavo viene effettuato un controllo della tensione che per la presenza della terminazione viene ad essere la metà di quella effettivamente in uscita al buffer di trasmissione. Se il dispositivo dovesse essere rimosso, ovvero la terminazione venisse a mancare, la tensione raddoppierebbe, e questo indicherebbe la disconnessione del dispositivo stesso. La modalità High Speed, aggiunge altri buffer dedicati a funzioni speciali ed in particolare aggiunge ulteriore complessità all'Host che deve supportare tutte le modalità USB 2.0.

Specifiche elettriche USB 3.x

Sebbene il numero di dispositivi che supportano il protocollo USB 3.x stia crescendo, questi sono tutti caratterizzati dall'avere performance di calcolo piuttosto elevate. Allo stato attuale le MCU sono troppo semplici per poter supportare un bit rate di 5Gb/s e 10Gb/s, per cui non entrerò nei dettagli dello stesso. Lo standard USB 3.x ha aumentato notevolmente il bit rate del bus permettendo un trasferimento dati più veloce. Al fine di supportare un bit rate più elevato non si è aumentata solo la frequenza di clock ma i *transceiver* sono stati migliorati fornendo il supporto di una linea dati differenziale dedicata per ogni senso di trasmissione come anche una massa. Un Host USB 3.x dovendo anche supportare un Host 2.0 possiede non solo le linee TX e RX dedicate alla comunicazione USB 3.x ma anche le linee D+ e D-.

Strumenti di Debug per il protocollo USB

Realizzare un prodotto con supporto USB richiede la progettazione di un sistema sia da un punto di vista Hardware che Software. Sistemi complessi richiedono spesso una fase di Debug al fine di poter rendere il sistema funzionante secondo specifica. Diversamente dai protocolli seriali come RS232, SPI e I2C per i quali sono presenti strumenti di sviluppo piuttosto economici e per i quali un oscilloscopio economico può essere sufficiente per il Debug del sistema, per il protocollo USB le cose non sono cosi semplici. Per la maggior parte dei progettisti la trasmissione sul canale USB rimane una scatola chiusa e non può essere visualizzata se non acquistando strumenti di Debug costosi come il *Protocol Analyzer* (Analizzatore di Protocollo) da collegare direttamente sul bus USB. I costi di questi strumenti possono variare da poche centinaia di euro a qualche migliaia. Normalmente i più economici non supportano il protocollo USB ma solo protocolli seriale base. Quelli di fascia bassa permettono generalmente di analizzare i dati sul bus USB facendo uso di ingressi digitali o meglio comparatori, mentre gli strumenti più costosi hanno ingressi analogici e permettono analisi più dettagliate del bus che vanno oltre alla semplice visualizzazione dei dati o segnali, infatti permettono di visualizzare la qualità del segnale da un punto di vista elettrico e verificare le specifiche elettriche USB relative al *timing*. Questa seconda fascia di strumenti sono più costosi e generalmente presenti solo nei laboratori o centri di ricerca e sviluppo che fanno uso frequente del protocollo USB. Spesso sono abbinati a oscilloscopi MSO (*Mixed Signal Oscilloscope*) di fascia alta.

Oltre agli strumenti Hardware per la verifica della qualità del segnale e analisi dei dati che transitano sul bus, sono presenti anche degli strumenti software per l'analisi ad alto livello dei dati che transitano. Infatti alcuni software possono intercettare i pacchetti dati inviati e ricevuti, come degli sniffer. Ciononostante al livello del driver non è possibile verificare i dettagli dei segnali visibili da un *Protocol Analyzer*, visto che i segnali di *Handshake* sono gestiti a livello hardware prima dell'intervento del driver e solo un collegamento diretto sul bus permette di visualizzarli. Alcuni di questi software sono forniti dall'USB IF al sito USB.org. In rete sono anche disponibili software gratuiti come USB sniffer per l'analisi dei dati USB e l'analisi della configurazione del dispositivo stesso. Per poter leggere le impostazioni del dispositivo, si capisce che l'hardware deve aver completato correttamente l'enumerazione altrimenti non è possibile fare alcun Debug. Per

cui tali strumenti sono da considerarsi di secondo livello visto che presumono che l'hardware e lo stack stiano funzionando già piuttosto bene. Alcuni di questi software richiedono l'installazione di driver dedicati al fine di intercettare quante più informazioni possibili dal dispositivo USB in analisi. Se si facesse uso dei driver standard non si avrebbe infatti accesso ai segnali visibili solo a livello interno del driver.

Una volta sviluppata la propria applicazione, a seconda della classe USB supportata, possono essere utilizzate altre applicazioni di supporto. Per esempio per la classe CDC (emulazione porta seriale) si può far uso dell'applicazione *RS232 Terminal* scaricabile dal sito LaurTec.it, la quale permette di inviare e ricevere dati per mezzo di una porta seriale.

Analisi delle linee dati

Gran parte degli oscilloscopi digitali, sebbene non forniscano la decodifica del protocollo USB, permettono, se con banda e frequenze di campionamento sufficienti, di verificare direttamente l'attività sul bus USB. In particolare le Figure sotto riportate sono state effettuate con un oscilloscopio con 200MHz di banda e un campionamento per canale pari a 1GS/s. In particolare CH1=D+ mentre CH2=D- e massa comune. Dall'analisi delle misure, sebbene non siano state fatte con il *Protocol Analyzer*, è comunque possibile determinare alcune delle impostazioni del dispositivo e verificare la qualità della trasmissione.

Nel caso di stream dati complessi come il protocollo USB, al fine di poter vedere i dati, avendo a disposizione una memoria sufficiente sull'oscilloscopio, può tornare utile impostare una base dei tempi larga e campionare una successione di pacchetti dati, per poi bloccare l'immagine ed effettuare semplicemente lo zoom nella parte dei dati d'interesse. Cercare infatti di prendere un SOF potrebbe risultare piuttosto noioso, sebbene possibile viste alcune simmetrie dei pacchetti dati. Una volta campionato un buon intervallo di tempo, avendo tanta pazienza potrebbe essere anche possibile decodificare le informazioni dei singoli pacchetti o esportare le immagini in formato CSV (*Comma Separated Value*) e creare un'applicazione ad hoc per l'analisi dei dati.

In Figura 9 è riportato un dettaglio di tre pacchetti dati inviati sul bus. In particolare il PC era collegato ad una scheda di sviluppo "Freedom III", ovvero ad un PIC18F14K50 programmato con classe CDC. Dai cursori verticali è possibile vedere che la frequenza con cui vengono inviati i pacchetti è di 1KHz ovvero ogni ms. In particolare, sebbene non sia possibile vedere i dettagli, ogni pacchetto contiene 64 Byte. Nel caso in cui si fosse avuto un dispositivo High Speed il *Data Field* avrebbe potuto avere fino a 1024 byte.

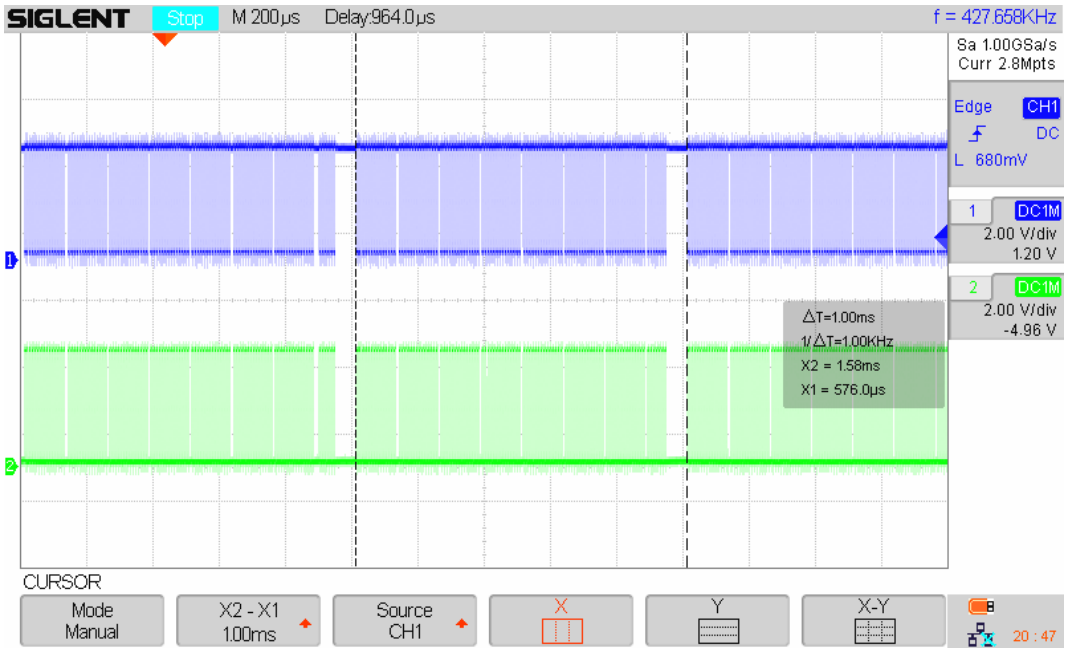

Figura 9: *Transizioni multiple sul bus USB.*

Facendo uno zoom sul singolo bit è possibile notare la frequenza di trasmissione. In particolare la Figura 10 mostra che l'ampiezza del bit è di 83ns ovvero 12MHz. Da questo si capisce, che il PIC18F14K50 è impostato in modalità Full Speed, ovvero con bit rate pari a 12MHz.

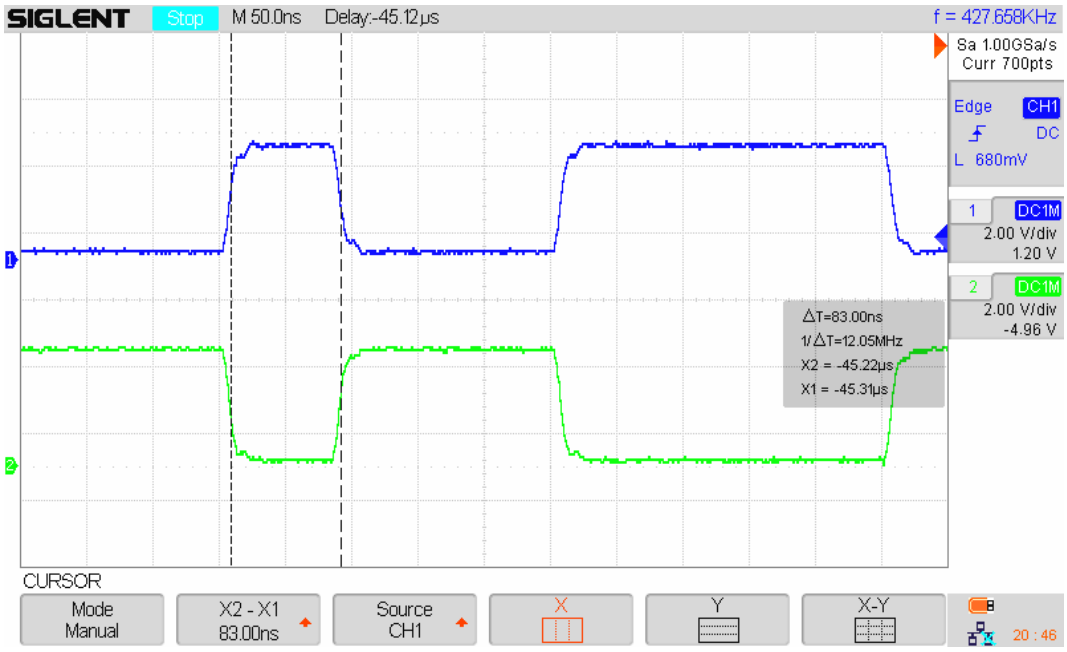

Figura 10: *Larghezza di un bit in una trasmissione Full Speed.*

Facendo lo zoom all'inizio del pacchetto è possibile notare altre cose interessanti. In particolare lo stato di IDLE è con CH1 alto e CH2 basso. Questa caratteristica, sapendo che CH1=D+, mostra che effettivamente il resistore di pull-up è propriamente attivato. In caso di problemi è certamente una cosa da verificare.

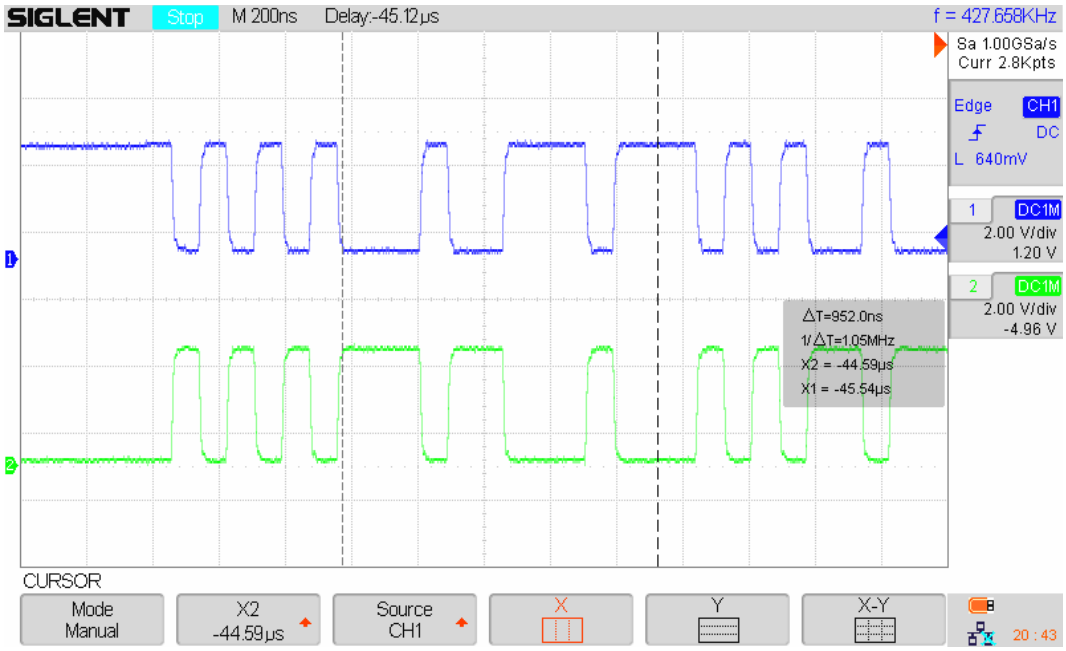

Figura 11: *USB IDLE e data SYNC a inizio pacchetto dati.*

Se si sa che il dispositivo sta lavorando in Full Speed, vedendo che CH1 è a livello alto, può anche aiutare per determinare che effettivamente CH1=D+. Oltre al livello di IDLE è possibile vedere il segnale di SYNC di inizio pacchetto, ovvero KJKJKJKK. Dopo il sincronismo è presente poi il PID (*Packet Identifier*) ovvero KJKK JJJK. Per decodificare il valore bisogna tener conto che il sincronismo termina con K. Dal momento che il PID inizia anche con K si ha che la sequenza decodificata è: 1001 0110, ovvero un pacchetto TOKEN IN (1001).
Analizzando la fine del pacchetto dati mostrato in Figura 12, è possibile notare il segnale EOP ovvero *End of Packet* che delimita la fine del pacchetto dati. Questo è composto dal livello SE0, ovvero D+ e D- a livello basso, seguito da un simbolo J, ovvero D+ alto. Dopo il livello J il bus va in stato di IDLE. Nel caso Full Speed, sebbene il livello J e IDLE siano di pari tensione, è comunque possibile vedere che l'inizio del livello alto è effettivamente un J visto che è leggermente più alto e di ampiezza un bit.

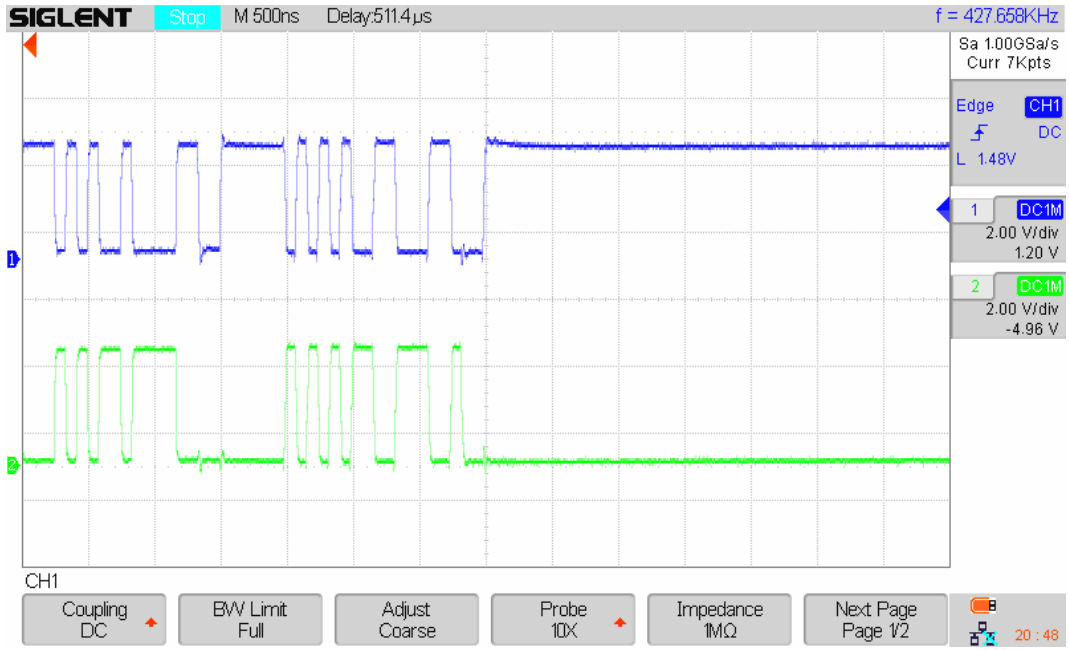

Figura 12: *Sequenza EOP sul bus USB.*

Quanto appena mostrato mette in evidenza come l'oscilloscopio possa comunque essere utilizzato per effettuare un Debug più approfondito sul sistema Embedded, sebbene possa non essere presente alcuna opzione di decodifica USB. In particolare è possibile verificare alcune delle impostazioni base del dispositivo.

Integrati con supporto USB

Sul mercato sono presenti diverse soluzioni che permettono un facile utilizzo del protocollo USB permettendo spesso di dimenticarsi totalmente dei dettagli. Come visto facendo uso della classe CDC si potrebbe anche scrivere un software come se si facesse uso di una porta RS232.
Spesso, nello sviluppo di piccoli volumi, la decisione sull'utilizzo di un integrato o MCU come soluzione ottimale è guidata da precedenti esperienze di utilizzo e/o tempo che si vuole investire. In particolare volendo sviluppare dal lato Host un semplice programma che utilizzi l'interfaccia seriale RS232, può portare alla scelta dell'utilizzo della classe CDC. Questa sarebbe la soluzione più semplice dal lato Host ma richiede l'installazione del file .inf con le relative informazioni del driver da utilizzare. Sebbene questo sia un passo non complicato la classe HID può semplificare la fase d'installazione ma complica lo sviluppo del software dal lato Host, sebbene siano presenti esempi ad hoc sviluppati da molti fornitori. La scelta sul come sviluppare il software dal lato Host non è il solo peso sulla bilancia. In particolare anche la scelta sul come sviluppare il sistema Embedded dal lato hardware sono dei vincoli importanti. Chi non ha esperienza con il protocollo USB ma sa utilizzare già dei microcontrollori spesso preferisce utilizzare la sola porta UART e collegarsi con degli integrati indipendenti che supportano la classe CDC senza richiedere alcuna programmazione. Di questi integrati i più famosi sono senza dubbio quelli forniti dalle seguenti società:

- FTDI (Es. FT232H, FT230X, FT231X)
- Texas Instruments (Es. TUSB3410)
- Cypress (Es. CY7C64225)

Recentemente anche la Microchip ha rilasciato un bridge USB-UART basato probabilmente su una versione programmata del PCI18F14Kxx ma che effettivamente dal lato hardware è un bridge USB a tutti gli effetti. Questo approccio è il più semplice e non richiede una reale conoscenza del protocollo USB.
Qualora dal lato Hardware si voglia utilizzare un microcontrollore le cose si complicano ma al tempo stesso si ha maggiore flessibilità visto che con la stessa MCU si possono sviluppare applicazioni per il supporto della classe CDC, HID, Mass Storage Device o altre classi USB, oltre a sistemi compositi. Microcontrollori con PHY e USB Engine sono disponibili in MCU sia ad 8, 16 che 32 bit. Tra i fornitori più noti che forniscono soluzioni complete Hardware e Software (fornendo stack USB gratuiti) si ricordano:

- Microchip (Es. PIC18F4550, PIC18F14K50, ecc.)
- ST (ST32Fxx,ecc.)
- Texas Instruments (MSP430F5xx, TM4C12x, ecc.)

Indice Alfabetico

Bibliografia

[1] www.LaurTec.it: sito ufficiale delle schede di sviluppo presentate nel Tutorial dove poter scaricare i relativi manuali utente ed esempi presentati.
[2] www.usb.org: Specifiche USB scaricabili gratuitamente.
[3] USB Complete: Autore Jan Axelson (Lakeview Research)

History

Data	Versione	Autore	Revisione	Descrizione Cambiamento
19.05.17	1.1	Mauro Laurenti	Mauro Laurenti	Correzione di errori di battitura e nuova lettura del testo che ha portato ad alcuni cambiamenti nell'esposizione ma non nei contenuti.
20.12.16	1.0	Mauro Laurenti	Mauro Laurenti	Versione Originale.