1. **Hardware**

Xác định tốc độ

Được xác định bằng các kiểm tra Pull D+ hoặc D- đến 3.3V.

Nếu không có điện trở pull up này, thì phía Host/Hub sẽ coi như không có gì được kết nối vào bus. Một số thiết bị tích hợp điện trở này bên trong thành phần mạch của nó, và cho phép bật hay không bằng xử lý trong firmware, mốt số khác thì sử dụng điện trở gắn ngoài chip.

Một số thiết bị cho phép khởi động trước cả khi bật tắt điện trở pull up để host xác định tốc độ của nó. A diagram of a circuit

Description automatically generated

Figure 1: Thiết bị Full Speed có một điện trở pull up nối với D+

A diagram of a low speed circuit

Description automatically generated

Figure 2: Thiết bị Low Speed có một điện trở Pull Up nối với D-

Thiết bị High Speed sẽ bắt đầu kết nối như là một thiết bị Full Speed. Khi đã được đã gắn vào Bus, nó sẽ thiết lập chế độ High Speed ở bước Reset nếu Hub/Host support tốc độ này. Khi chế độ High Speed được bật, thì điện trở Pull Up sẽ được loại bỏ để giữ cân bằng line.

Thiết bị tuân theo USB 2.0 không yêu cầu bắt buộc support High Speed. Điều này cho phép thiết bị có thể được sản xuất với giá rẻ hơn nếu không có yêu cầu khắt khe về tốc độ. Một thiết bị low speed cũng thế, không yêu cầu support Full Speed nếu nó không cần thiết cho thiết bị.

Một thiết bị High Speed không cần support low speed, nó chỉ cần support Full Speed vì sẽ sử dụng ở bước connect ban đầu. Tuân thủ USB 2.0, downstreamxuống thiết bị phải support cả 3 mode về tốc độ.

1. **Nguồn**

Một thiết bị không thể nhận một dòng cao hơn cái được thiết trong suốt quá trình emuration được kể cả nó không có nguồn cấp ngoài đi nữa. Có 3 lớp cung cấp nguồn chu USB:

Cấp nguồn thấp qua bus: Low-power bus powered functions  
Cấp nguồn cao bus: High-power bus powered functions  
Tự cấp nguồn : Self-powered functions

Chức năng cung cấp nguồn thấp qua bus sẽ cung cấp nguồn cho thiết bị từ Vbus, nó không thể có dòng lớn hơn 1 unit được. Đặc tả USB định nghĩa 1 unit tải tương đương với 100mA. Chức năng cung cấp nguồn thấp qua bus này phải được thiết kế để làm việc điện áp từ 4.40V đến 5.25V được đo ở phía UpStream (host/hub). Với những thiết bị 3.3V thì cần phải có LCD Regulators.

Chức năng cung cấp nguồn cao qua bus sẽ cung cấp nguồn cho thiết bị từ Vbú, nó không thể có dòng lớn hơn 1 unit cho đến khi được configured xong.

Chức năng tự cấp nguồn, có thể nhận được nguồn tối đa 1 unit từ bus, còn lại nhận từ nguồn bên ngoài. Nếu nguồn bên ngoài bị ngắt, thì ngay lúc đó phải đảm bảo không lấy nhiều hơn 1 unit từ bus

Không có thiết bị USB nào được phép điều chỉnh VBus từ upstream. Nếu VBus hỏng, tín hiệu qua các chân D+/D- sẽ bị ngắt trong 10 s.

Một vấn đề khác với VBus là dòng ban đầu phải được giới hạn. Điều này được đề cập đến trong phần 7.2.4.1 của đặc tả. Dòng sẽ hóa giải bởi một điện dung vừa đủ giữa VBus và ground. Trong đặc tả chỉ giá trị lớn nhất của điện dung có thể sử dụng là 10uF. Một điều nữa, khi rút thiết bị khỏi kết nối, dòng hiện tại sẽ bị ném qua dây USB một sự tăng áp lớn có thể xảy trên các đầu mở dây. Để ngăn điều này, một điện dung nhỏ nhất 1uF được chỉ định.

Với những thiết bị sử dụng nguồn từ bus, nó không thể có dòng lớn hơn 500mA, thường là không thể. Bạn có tự hỏi đây là giá trị gì không? Có lẽ là Suspend Mode.

1. **Protocol**

USB được tạo bởi nhiều lớp protocol

Mỗi USB Transaction bao gồm

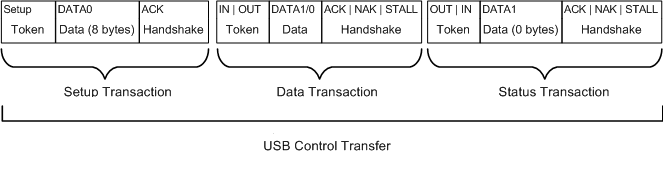
Một packet chứa Token (Giống như Header cho biết cái gì sẽ đến tiếp sau đó) – luôn có

Một packet chứa dữ liệu (có thể không có) , (chứa payload)

Một packet chứa trạng thái (được sử dụng để thông báo trạng thái truyền và cung cấp cơ chế kiểm soát lỗi)

USB là dạng bus tập trung vào host. Host sẽ bắt đầu mọi transactions

Khi gửi gói đầu tiên, một token được sinh ra bởi host. Từ token đó, sẽ xác định cái gì dữ liệu nào được gửi tiếp sau đó. Dữ liệu đó có thể là để đọc, ghi và địa chỉ thiết bị là gì, cho endpoint nào. Packet sau token nói chung sẽ mang dữ liệu, cuối cùng là một gói để kết thúc quá trình. Gói cuối sẽ cho biết dữ liệu hoặc token được gửi thành công hay chưa.



Sau đây sẽ là các trường phổ biến trong một Packet USB. Chú ý rằng, dữ liệu truyền trên USB Bus là dạng LSBit First.

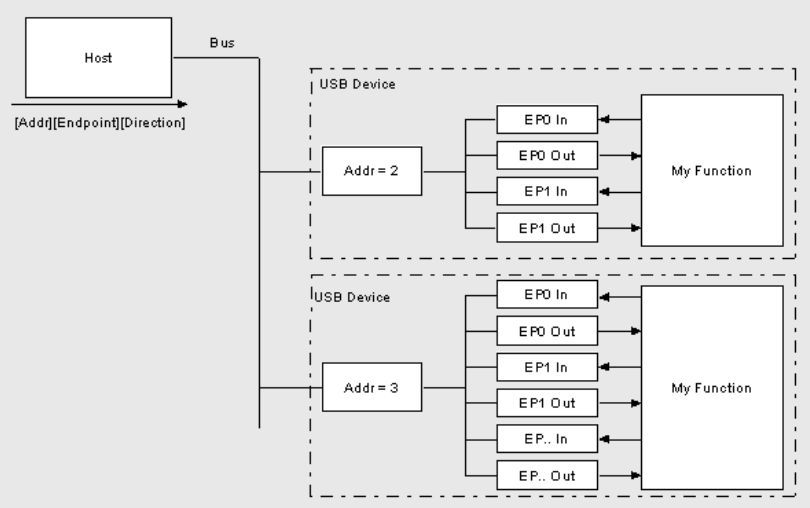
|  |  |
| --- | --- |
| Sync | Tất cả các  Packet đều bắt đầu bằng trường này. Ở Low Speed và Full Speed thì trường này có độ dài 8 bit, còn ở High Speed là 32 bit. Trường này được sử dụng để đồng bộ clock giữa bên gửi và bên nhận. 2 bit cuối của Sync báo rằng sau đó là điểm bắt đầu của trường PID. |
| PID | PID nghĩa là Packet ID. Trường này được sử dụng để xác định loại packet được gửi. Bảng sau mô tả loại packet và giá trị tương ứng.   |  |  |  | | --- | --- | --- | | Nhóm | Giá trị PID (Loại Packet) | Tên Packet | | Token | 0001 | OUT Token | | 1001 | IN Token | | 0101 | SOF Token | | 1101 | SETUP Token | | Data | 0011 | DATA0 | | 1011 | DATA1 | | 0111 | DATA2 | | 1111 | MDATA | | Handshake | 0010 | ACK Handshake | | 1010 | NAK Handshake | | 1110 | STALL Handshake | | 0110 | NYET (No Response Yet) | | Special | 1100 | PREamble | | 1100 | ERR | | 1000 | Split | | 0100 | Ping |   Như bảng trên ta cũng thấy PID chiếm 4 bit. Nhưng để đảm bảo nhận chính xác, giá trị đảo của 4bit này được thêm vào đuổi, thành ra có 8 bit cho trường PID.   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | PID0 | PID1 | PID2 | PID3 | nPID0 | nPID1 | nPID2 | nPID3 | |
| ADDR | Trường địa chỉ, dùng để chỉ ra Packet này sẽ cho thiết bị nào. Có 7bit cho trường này, vì thế có tối đa 127 địa chỉ hợp (như ta đã nói từ đầu số thiết bị tối đa là 127) + 1 địa chỉ ADDR=0 là không hợp lệ. Địa chỉ Zero này không được gán cho bất cứ thiết bị nào, chẳng những thế những thiết bị khi chưa được cấp địa chỉ phải trả lời (bằng Packet) tới địa chỉ Zero. |
| ENDP | EndPoint, Độ dài 4 bit vì thế hỗ trợ tối đa 16 Endpoint.  Có 1 trường hợp đặc biệt cho thiết bị Low Speed là chỉ có 2 Endpoint ở mỗi đầu gửi nhận (tức là tối đa 4 Endpoin thôi |
| CRC | Giá trị này được sinh ra trên dữ liệu chính của Packet (phần Payload). Trường CRC của Packet Data có độ dài 16 bit,các Packet khác có độ dài cố định 5 bit. |
| EOP | End Of Packet |

* Các loại USB Packet

Như đã trình bày trong bảng về trường PID, chúng ta có 4 nhóm (loại lớn) Packet khác nhau. Packet Token sẽ xác định loại transaction, Packet data sẽ chứa Payload. Packet Handshake được sử dụng cho báo cáo tình trạng gửi nhận, thông báo lỗi. Cuối cùng, Packet Start Of Frame để báo hiệu bắt đầu 1 frame mới.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Packet Token | Có 3 loại Packet Token  IN – Báo cho USB Device biết host muốn đọc thông tin từ nó  OUT – Báo cho USB Device biết host gửi thông tin cho nó  SETUP – Báo cho USB Device biết host sẽ truyền thông tin điều khiển  Mỗi Packet Token phải tuân theo định dạng sau:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Sync** | **PID** | **ADDR** | **ENDP** | **CRC5** | **EOP** | |
| Packet Data | Có 2 loại Packet data, mỗi loại đều có thể truyền tối đa 1024 byte dữ liệu.  Data0  Data1  Ở High Speed mode, có thể sử dụng thêm 2 PID khác nữa là DATA2 và MDATA.  Packet data sẽ có định dạng sau:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Sync** | **PID** | **Data** | **CRC16** | **EOP** | |  |  |  |  |  |   Ở Low Speed, cho phép tối đa 8 bytes payload.(phần Data ở định dạng trên)  Ở Full Speed,  cho phép tối đa 1023 byte payload.  Ở High Speed, cho phép tối đa 1024 bytes.  Data phải được gửi thành nhiều byte. |
| Packet Handshake | Có 3 loại Packet Handshake là :  ACK – Cho biết Packet đã được gửi nhận thành công chưa  NAK – Cho biết Device không tạm thời không thể gửi hoặc nhận dữ liệu. Ngoài ra, gói này cũng được sử dụng trong Transaction dạng Interrupt để báo cho host biêt rằng device chẳng ó gì để gửi.  STALL – Device báo rằng trạng thái hiện tại cần cần thiệp từ phía Host.  Packet Handshake sẽ có dạng sau:   |  |  |  | | --- | --- | --- | | **Sync** | **PID** | **EOP** | |
| Packet Start Of Frame | Hay Packet SOF, chứa dữ liệu là một giá trị 11 bit. Được gửi bởi Host theo chu kì 1ms+- 500ns với thiết bị Full Speed, và chu kì 1us – 0.0625us trên High Speed.  Định dạng của SOF sẽ như sau:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Sync** | **PID** | **Frame Number** | **CRC5** | **EOP** | |

Hấu hết USB Functions thực hiện từ dưới cho đến tầng transaction (chúng ta sẽ nói ở những Chapter sau) trong phần cứng. Lý do, chúng ta phải biết những thông tin chi tiết phía trên là vì USB Functions sẽ thông báo các lỗi.



Phần mềm sẽ phải xử lý ngắt, đọc dữ liệu từ buffer của EndPoint, rồi parse Device Descriptor Request.

* Endpoints

Có thể hiểu là nguồn hay bể chứa dữ liệu. Như ta đã biết USB là xoanh quanh Host, tức là mọi giao tiếp được bắt đầu từ phía Host, EndPoints là điểm cuối của kênh giao tiếp và nằm phía USB Functions. Giả sử driver gửi một Packet đến EP1 của thiết bị. Dữ liệu này được ném từ Host, và trôi xuống EP1 OUT Buffer. Firmware phía USB Functions cứ thế đọc dữ liệu này. Rồi xử lý tóe loe gì đó. Sau đó nếu nó muốn trả dữ liệu nào đó về Host, USB Functions lại không thể tống dữ liệu đó vào cái Channel lúc trước được (vì ngược chiều, đâm nhau thì toi). Vì thế, nó ghi dữ liệu vào EP1 IN Bufer. Dữ liệu được ghi vào cứ nằm ở đấy đến khi nào Host gửi Packet IN (yêu cầu lấy dữ liệu) thì mới được chuyển đi. EndPoint cũng được xem như điểm trung chuyển giữa phần cứng của Function Device và Firmware chạy trên Function device.

Tất cả các Device (của Host và Function) đều có EndPoint Zero. Đây là EndPoint duy nhất nhận toàn bộ request Control/Status trong suốt quá trình Enumeration, cũng như lúc thiết bị giao tiếp trên bus.

* **Pipes**

Đã nói ở trên, EndPoint là điểm trung giữa phần cứng và phần mềm trong USB Function Device. Tuy nhiên, nâng cao lên 1 chút ở mức phần mềm (dù vẫn thuộc Firmware – Driver Level) thì người ta thường sử dụng một khái niệm khác nữa. Đó là các Pipes. Một Pipe là một kết nối logical giữa Host và 1 hoặc nhiều Endpoint. Pipe có một tập các tham số kèm theo chúng như bandwidth là bao nhiêu, loại truyền  (transfer type) (Control, Bulk, Iso, Interrupt), hướng của luồng dữ liệu chạy qua, kích thước Buffer của Pipe nữa. Lấy ví dụ về Default Pipe là một Pipe 2 chiều nối Host với EndPoint Zero IN và EndPoint Zero OUT, loại truyền là Control Transfer.

Về cơ bản, có 2 loại Pipe lớn:

|  |  |
| --- | --- |
| Stream Pipe (Pipe dạng dòng | truyền dữ liệu không định dạng trước, khi sử dụng loại này bạn có đơn giản là có thể gửi bất cứ dữ liệu gì ở 1 đầu, và lấy dữ liệu ra ở đầu còn lại. Luồng dữ liệu sử dụng Pipe này thường được định nghĩa trước, hoặc là IN hoặc là OUT. Pipe dạng dòng hỗ trợ phương thức truyền Bulk, Isochronous và Interrupt. Stream có thể được điều khiển (controlled về phần mềm) bởi Host hoặc Device |
| Message Pipi (Pipe truyền messsage | Dùng để truyền dữ liệu đã được định nghĩa theo USB Format. Được điều khiển cũng như xuất phát từ Host. Dữ liệu được truyền đi theo hướng mong muốn dựa trên request từ Host. Nó hỗ trợ truyền dữ liệu cả 2 hướng và chỉ hỗ trợ Control Transfer thôi. |

1. **Endpoints**

**Đặc tả USB định nghĩa 4 loại Transfer/EndPoint**

Control Transfer

Interrupt Transfer

Isochronous Transfer

Bulk Transfer

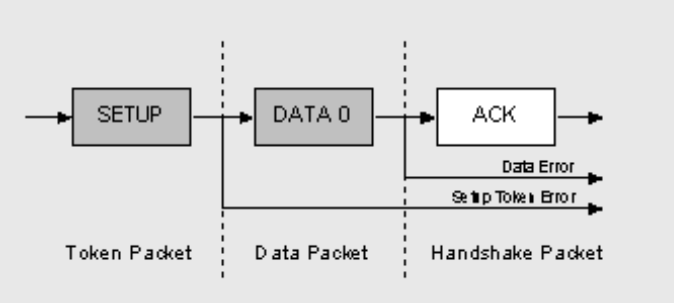
1. **Control Transfer (Truyền điều khiển)**

Control Transfer thường được sử dụng cho Commands và Status. Đây là loại truyền chủ yếu được sử dụng trong suốt quá trình Enumeration. Ở dạng truyền này, các gói tin được gửi theo phương trâm best effort delivery (gửi đến khi nào nhận được thì tính tiếp). Độ dài mỗi packet trong dạng truyền control ở mỗi speed lại khác nhau. Ở low speed, nó là 8 byte; Ở High Speed cho phép độ dài 8, 16, 32 và 64 nữa; Ở Full Speed thì phải là 64 byte.

Trong dạng truyền Control, mỗi lần truyền sẽ có 3 stage, mỗi Stage là một nhóm packet và được thực hiện liên tiếp nhau.

* **Đầu tiên là Setup Stage**

Ở Stage này, request sẽ được chuyển đến phía Function Device. Quá trình này được thực hiện trong 3 packet. Packet đầu tiên là Setup Token, chứa Address và Endpoint Number. Tiếp theo là Data Packet, luôn chứa PID Type ở Data0 và một Setup Packet chỉ rõ loại request (Setup Packet sẽ được giải thích chương 6). Packet cuối cùng là một handshaked, dùng để cho biết việc nhận đã xảy thành công hay chưa. Nếu phía Function Device nhận thành công Setup Data (gồm CRC và PID đều OK), nó sẽ trả lời phía Host bằng ACK. Ngược lại, nó có thẻ bỏ qua, không cần gửi lại gì hết. Phía Function không thể gửi một gói STALL hoặc NAK để trả lời Setup Packet được.



* **Data Stage**
* Stage này không phải lúc nào cũng có. Nó chứa một hoặc nhiều transfer IN hoặc OUT. Request Setup sẽ cho biết lượng dữ liệu được truyền. Nếu lượng dữ liệu phải truyền vượt quá kích thước tối đa của một Packet, thì dữ liệu sẽ được gửi lần lượt theo từng gói.

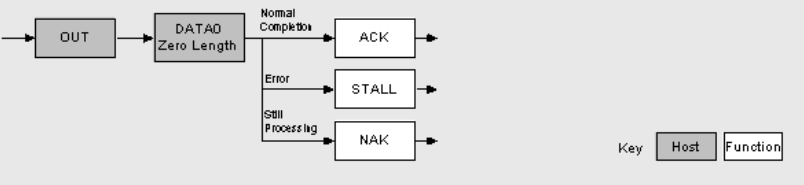
Có 2 kịch bản gửi dữ liệu, đó là IN và OUT.

|  |  |
| --- | --- |
| IN | Khi Host muốn nhận dữ liệu, nó sẽ gửi một IN Token. Phía Function nhận được IN Token sẽ kiểm tra xem trường PID (bao gồm phần đảo bit) có hợp lệ hay không. Nếu hợp lệ, device Function sẽ trả lại bằng một packet DATA chứa dữ liệu cần gửi hoặc packet STALL nếu EndPoint có lỗi hoặc NAK nếu EndPoint đang có xử lý khác (tất nhiên các trường hợp lỗi sẽ không có dữ liệu được gửi đi). |
| OUT | Khi host cần gửi một gói dữ liệu điều khiển, nó sẽ gửi đi một token OUT với phần payload là dữ liệu cần gửi. Nếu có bất cứ phần nào của token OUT hoặc packet Data bị lỗi, thì phía Function sẽ bỏ qua các packet này. Ngược lại, nếu dữ liệu hợp lệ và buffer Endpoint chỉ định đang rỗng nữa thì phía Functions sẽ đẩy dữ liệu vào buffer đó. Rồi Function sẽ gửi lại một ACK cho Host khi nó nhận đầy đủ dữ liệu. Còn nếu buffer Endpoint không rỗng (tức đang có dữ liệu chờ xử lý), nó sẽ gửi lại phía Host một NAK. Còn lại, khi endpoint gặp bất cứ lỗi nào khác, Function sẽ trả lại STALL cho Host. |

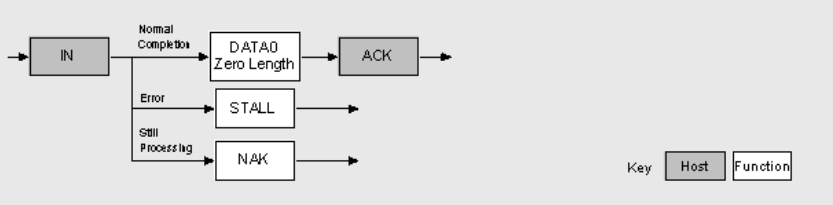
* **Status Stage**

Sẽ thông báo trạng thái của toàn bộ quá trình, nội dung của stage này phụ thuộc vào hướng truyền. Thông báo về Status luôn được thực hiện bởi Functions.

**IN :** Nếu host gửi một Token IN để nhận dữ liệu, thì phía Host phải xác nhận dữ liệu đã được nhận thành công. Việc xác nhận này được thực hiện khi Host gửi một token OUT với độ dài payload là zero. Việc nhận được token này sẽ coi như là ACK. Nếu không nhận được hoặc nhận lỗi, phía Host sẽ coi là STALL. Trường hợp còn lại sẽ coi là NACK, để báo rằng Host sẽ lặp lại Status Stage sau đó.



**OUT**: Nếu host gửi OUT token để gửi dữ liệu ra thì phía Function sẽ xác nhận việc nhận dữ liệu thành công bằng cách gửi lại một Packet có độ dài là Zero cho 1 Token IN mà Host sẽ gửi sau cùng. Tuy nhiên, nếu có lỗi xảy ra khi trả lại Zero Packet thì nó sẽ coi là STALL. Nếu Function ko thể trả lời IN Token do bận, nó sẽ báo lại là NACK để Host nhận lại Status trong Phase tiếp theo.



1. Truyền control

Ví dụ : Host muốn request Device Descriptor trong quá trình Enumeration thì những packet nào sẽ được gửi/nhận

Đầu tiên, Host sẽ gửi Set up Token để báo cho function biết rằng nó sẽ gửi Setup packet tiếp sau. Trường địa chỉ ADDR sẽ chứa địa chỉ của device mà Host muốn request đến. EndPoint number là Zero, tức là Default Pipe.

Tiếp đó, Host sẽ gửi Data Packet (loại Setup) chưa 8 byte payload (chính là Device Descriptor Request được miêu tả trong Chapter 09 của USB Specification). Phía USB Function sau đó gửi ACK (để xác nhận) sau khi đã nhận Setup Packet đầy đủ và không lỗi. Nếu nội dung của Packet bị lỗi, Device sẽ bỏ qua Packet đó và không gửi ACK gì hết. Phía Host sẽ tự nhận ra và gửi lại Packet đó sau đó ít lâu.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **1. Setup Token** | **Sync** | **PID** | **ADDR** | **ENDP** | **CRC5** | **EOP** | Address & Endpoint Number |
|  | | | | | | | |
| **2. Data0 Packet** | **Sync** | **PID** | **Data0** | | **CRC16** | **EOP** | Device Descriptor Request |
|  | | | | | | | |
| **3. Ack Handshake** | **Sync** | **PID** | **EOP** |  | | | Device Ack. Setup Pack |

Giờ phía Device sẽ decode Packet và có được 8 byte. Sau khi xác định được 8 byte này là Device Descriptor Request, nó sẽ gửi Device Descriptor trong các packet bên dưới đây.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **1. In Token** | **Sync** | **PID** | **ADDR** | **ENDP** | **CRC5** | **EOP** | Address & Endpoint Number |
|  | | | | | | | |
| **2. Data1 Packet** | **Sync** | **PID** | **Data1** | | **CRC16** | **EOP** | First 8 Bytes of Device Descriptor |
|  | | | | | | | |
| **3. Ack Handshake** | **Sync** | **PID** | **EOP** |  | | | Host Acknowledges Packet |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **1. In Token** | **Sync** | **PID** | **ADDR** | **ENDP** | **CRC5** | **EOP** | Address & Endpoint Number |
|  | | | | | | | |
| **2. Data0 Packet** | **Sync** | **PID** | **Data0** | | **CRC16** | **EOP** | Last 4 bytes + Padding |
|  | | | | | | | |
| **3. Ack Handshake** | **Sync** | **PID** | **EOP** |  | | | Host Acknowledges Packet |

Ở hình trên, payload tối đa của 1 packet được giả định là có độ dài 8 byte. Vì thế, ta thấy Device Descriptor (độ dài 12 byte) được chia ra 2 lần gửi tương ứng với 2 lần nhận IN Token.

Sau khi Device Descriptor được gửi hết, thì Host sẽ yêu cầu phía Function xác nhận dữ liệu gửi bằng cách gửi OUT và mong muốn nhận được response với length là ZERO.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **1. Out Token** | **Sync** | **PID** | **ADDR** | **ENDP** | **CRC5** | **EOP** | Address & Endpoint Number |
|  | | | | | | | |
| **2. Data1 Packet** | **Sync** | **PID** | **Data1** | | **CRC16** | **EOP** | Zero Length Packet |
|  | | | | | | | |
| **3. Ack Handshake** | **Sync** | **PID** | **EOP** |  | | | Device Ack. Entire Transaction |

### **Interrupt Transfers (Dạng truyền ngắt)**

Ngắt được sinh ra từ thiết bị ở bất cứ thời điểm cho phép nào. Tuy nhiên, trong USB thì phía thiết bị (Function) vẫn phải đợi Host hỏi rồi mới báo là có ngắt hay không.

**Đặc điểm của Interrupt Transfers:**

* Có độ trễ được đảm bảo
* 1 hướng
* Có phát hiện lỗi và retry

Interrupt Transfer thông thường là không có chu kì cố định, thiết bị nhỏ sẽ bắt đầu truyển trong một độ trễ đảm bảo. Các Interrupt request sinh ra từ thiết bị sẽ được giữ đến khi host hỏi .

**Đặc trưng về payload của data:**

* Kích thước tối đa của payload cho low-speed là 8 bytes.
* Kich thước tối đa của payload cho Full-Speed là 64 bytes.
* Kích thước tối đa của payload cho High-Speed là 1024 bytes.

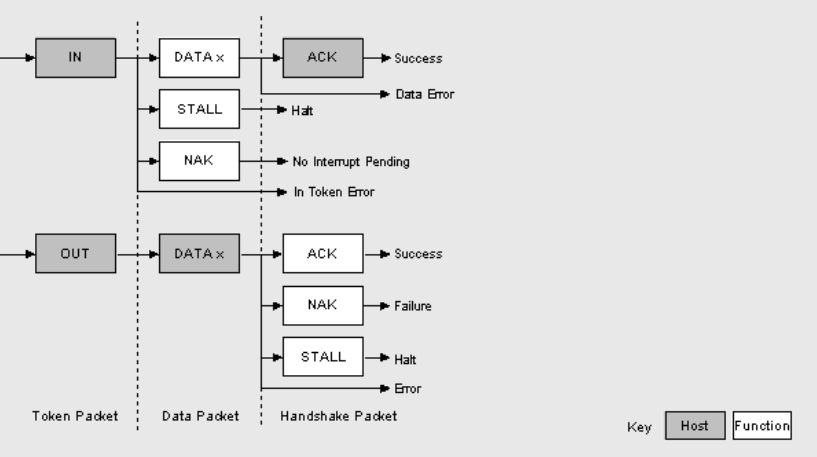


Figure 3: Interrupt phổ biến IN và OUT

**IN** : Phía Host sẽ định kì poll (hỏi) phía Function xem có ngắt hay chưa. Tần số polling này được quy định trong phần EndPoint Description của thiết bị đó. Mỗi lần hỏi host sẽ gửi một token IN. Nếu token này bị hỏng, phía Function sẽ bỏ qua packet và tiếp tục theo dõi bus để đợi token mới.

Nếu có ngắt đang được xếp bên trong thiết bị, thì Function sẽ gửi một packet trả loại phía Host và kèm theo dữ liệu tương ứng với ngắt đó. Host sau khi nhận được trả lời, nó sẽ báo lại bằng 1 ACK. Nếu dữ liệu trả lại bị hỏng, thì Host sẽ không trả lại gì hết. Nếu không có ngắt nào xảy ra khi Function nhận được poll từ Host, nó sẽ báo lại cho Host bằng NAK. Nếu có lỗi xảy ra trên Function thì nó sẽ gửi lại STALL cho Host.

**OUT**: Khi một host muốn gửi dữ liệu ngắt đến thiết bị bằng ngắt, nó sẽ đây một OUT token, tiếp theo đó là dữ liệu. Nếu bất kì phần nào của OUT token hoặc dữ liệu bị hỏng, phía Function sẽ bỏ qua packet. Nếu buffer của EndPoint bị rỗng, Function sẽ copy dữ liệu vào, rồi báo lại cho Host biết bằng 1 ACK. Nếu buffer của EndPoint không rỗng, tức là dữ liệu trước đó chưa được xử lý, thì Function sẽ gửi trả lại Host 1 gói NACK. Tuy nhiên, nếu lỗi xảy ra liên tục, và thấy rằng nên gửi lại thì nó sẽ gửi cho Host 1 STALL.

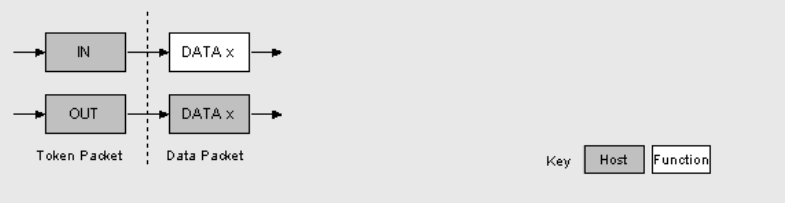
### **Isochronous Transfer (Dạng truyền đồng bộ)**

Là dạng truyền xảy ra liên tục và có chu kì ổn định. Được sử dụng khi truyền những thông tin đặc trưng như âm thanh, hình ảnh. Nếu delay hoặc gửi lại xảy ra khi truyền âm thanh, hình ảnh, thì chúng ta sẽ thấy một số âm thanh có vẻ kì cục, kiểu như xước đĩa etc. Hoặc các nhạc không đồng bộ với hình ảnh, ta có thể dễ dàng nhận thấy chúng. Tuy nhiên, nếu 1 packet hoặc 1 frame bị drop một vài lần thì nó ít khi bị chú ý bởi người xem, nghe.

**Dạng truyền đồng bộ sẽ cung cấp:**

* Một bandwidth đảm bảo
* Độ trẽ được giới hạn
* Một đường ống (stream pipe) vô hướng
* Phát hiện lỗi thông qua CRC, nhưng không yêu cầu gửi lại
* Chỉ sử dụng ở Full hoặc High Speed mode
* Không data toggling

Kích thước lớn nhất của dữ liệu truyền (payload) được chỉ rõ trong phần mô tả EndPoint của một Endpoint truyền đồng bộ. Nó có thể truyền tới 1023 bytes với Full Speed và 1024 bytes với High Speed. Kích thước lớn nhất của payload sẽ ảnh hướng đến bandwitdh yêu cầu đến bus. Vì thế, phải thận trọng để chọn payload size phù hợp. Nếu sử dụng một payload lớn, thì nên sử dụng các định nghĩa cho alternative interface (interface thay thế khi cần) để thực hiện nhiều payload size. Nếu trong quá trình enumeration, host không thể cho phép bandwidth mà Endpoint yêu cầu do việc hạn chế sử dụng bandwidth, thỉnh thoảng sẽ phải hạn bandwidth xuống chứ không hẳn là không thể sử dụng được. Dữ liệu được gửi trong Endpoint có thể ít hơn con số đã được chấp nhận tại bước enumeration. Có thể thay đổi trong mỗi transaction.



Hình trên mô tả định dạng truyền dồng bộ với IN và OUT token. Transaction truyền động bộ không có bước handshake và cũng không thể thông báo lỗi dạng STALL, HALT được.

### **Bulk Transfer (Dạng truyền hàng loạt)**

Có thể được sử dụng cho truyền những dữ liệu rất lớn. Ví dụ, một dữ liệu để print hoặc một ảnh được scan từ scanner. Dạng truyền hàng loạt cung cấp cơ chế phát hiện lỗi thông qua CRC16 và có cơ chế để yêu cầu gửi lại nếu lỗi để đảm bảo dữ liệu được truyền/nhận không lỗi.

Dạng truyền này sử dụng số bandwidth chưa được sử dụng còn lại sau khi tất cả các dạng truyền khác đã được cấp. Nếu bus quá bận với các dạng truyền khác thì dạng truyền hàng loạt sẽ rất nhỏ giọt. Vì thế, dạng truyền này nên sử dụng với các dữ liệu không yêu cầu đáp ứng về mặt thời gian, cũng như đảm bảo về độ trễ.

**Dạng truyền hàng loạt:**

* Sử dụng để truyền dữ liệu lớn
* Phát hiện lỗi bằng CRC, với cơ chế đảm bảo truyền nhận
* Không đảm bảo bandwidth tối thiểu
* Truyền theo đường ống (stream pipe) vô hướng
* Chỉ sử dụng ở mode Full và High Speed.

Dạng truyền hàng loạt chỉ được hỗ trợ ở Full và High Speed. Kích thước tối đa của payload mỗi packet có thể là 8,16,32,64 byte với Full Speed, và lên đến 512 bytes với High Speed. Dù dữ liệu truyền thực tế nhỏ hơn kích thước lớn nhất, nó cũng không cần padding zero. Một lần truyền hàng loạt được xem là hoàn thành khi nó truyền chính xác số dữ liệu được yêu cầu. Packet truyền đi có thể nhỏ hơn kích thước lớn nhất của EndPoint hoặc là Zero-length packet.

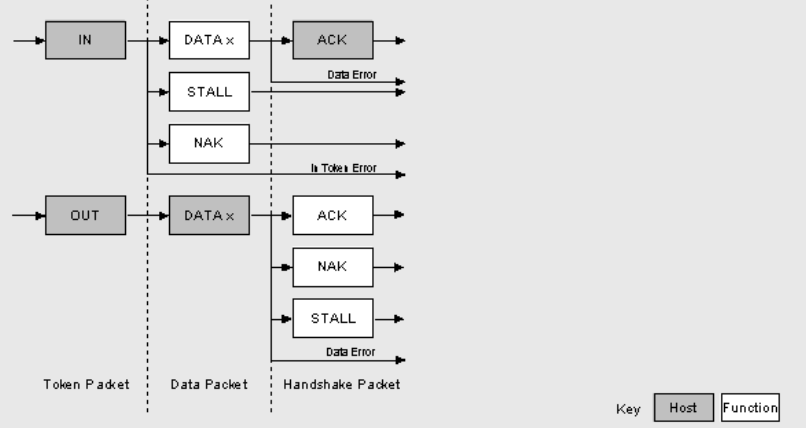


Figure 4: định dạng của 1 bulk IN và bulk OUT transaction:

**IN** : Khi Host sẵn sàng nhận dữ liệu hàng loạt, nó sẽ gửi một IN token. Gói này sẽ bị Function bỏ qua nếu có lỗi. Function sẽ gửi trả lại dữ liệu bằng gói DATA nếu nhận đúng gói IN. STALL được gửi khi phía Function có lỗi, NACK được gửi nếu phía Function chưa có dữ liệu hoặc EndPoint đang bận.

**OUT**: Khi host muốn gửi đến Function dữ liệu bulk, nó sẽ gửi một OUT Token, sau đó là gói dữ liệu. Nếu OUT token hoặc Data Packet bị hỏng, thì Function sẽ bỏ qua Packet này. Nếu buffer của EndPoint là Empty, thì Function sẽ copy dữ liệu vào buffer rồi trả lại Host gói ACK. Nếu endpoint buffer chưa rỗng do dữ liệu trước chưa được xử lý, thì Function sẽ trả lại NAK. Nếu có lỗi xảy ra, Function sẽ gửi lại STALL.

### **Quản lý BandWidth**

Host chịu trách nhiệm quản lý bandwidth của bus. Việc này được thực hiện trong quá trình enumeration khi cấu hình các Endpoint đồng bộ, End Point ngắt.

Đặc tả của USB nêu chỉ rõ giới hạn để đảm bảo rằng không cho phép quá 90% số frame được cấp phương thức truyền có tính chu kì (Interrupt và Isochronous) khi ở Full-Speed. Ở High-Speed, giới hạn trên được giảm xuống còn 80% microframe có thể cấp cho các phương thức truyền có tính chu kì.

Vì thế, bạn sẽ thấy rằng, sẽ có một đủ bandwidth đế mức không dùng hết với các phương thức truyền có tính chu kì. Chỉ còn khoảng 10% dành cho control nếu nó được sử dụng, còn lại mới đến lượt bulk transfer.

1. **Đặc tả thiết bị**

Các đặc tả phổ biết nhất bao gồm:

* Miêu tả thiết bị (Device Descriptors)
* Miêu tả các cấu hình (Configuration Descriptors)
* Miêu tả giao diện (Interface Descriptors)
* Miêu tả điểm đầu cuối (EndPoint Descriptors)
* Các chuỗi sử dụng trong các miêu tả trên.

1. Mỗi thiết bị chỉ có 1 miêu tả thiết bị (Device Descriptors).

Thông in chứa trong mô tả thiết bị bao gồm phiên bản nào của USB mà thiết bị tuân theo, mã sản phẩm (Product IDs), mã nhà sản xuất (Vendor IDs), chính các thông tin này được sử dụng để xác định driver nào cần thiết để có thể giao tiếp được với thiết bị, cũng như số lượng cấu hình mà thiết bị có thể có. Số lượng cấu hình của thiết bị liên quan đến việc cấu hình mà thiết bị có thể theo (tất nhiên mỗi thời điểm chỉ 1).

Miêu tả cấu hình chỉ rõ các giá trị như điện năng sử dụng trong mỗi cấu hình, thiết bị là tự cấp nguồn (self-powered) hay lấy nguồn từ bus (bus-powered), rồi thì số lượng giao diện (interface) của mỗi cấu hình. Trong quá trình enumurated, Host sẽ đọc miêu tả thiết bị, và đưa ra quyết định cấu hình nào được sử dụng. Chỉ được phép sử dụng 1 cấu hình.

Ví dụ, một thiết bị có 2 cấu hình là sử dụng nguồn bus và sử dụng nguồn tự có. nếu thiết bị được cắm vào Host có nguồn ổn định nó có thể chọn cấu hình sử dụng bus, khi đó nó có thể hoạt động mà không cần một nguồn cắm ngoài nào khác. Ngược lại, nếu nó kết nối với một laptop chẳng hạn, nó có thể bật cấu hình sử dụng nguồn tự có, khi đó ta cần kết nối nó với một nguồn khác để có thể giao tiếp được.

Không nhất thiết 2 cấu hình phải có cách sử dụng nguồn khác nhau. Có thể 2 cấu hình sử dụng nguồn giống hệt nhau những khác nhau về các thông số khác như các giao diện, kết hợp các Endpoint. Chú rằng, khi thay đổi cấu hình, Host phải dừng toàn bộ các EndPoint đang sử dụng để chuyển sang cấu hình mới. Dù có thể chưa nhiều cấu hình, nhưng rất ít thiết bị có nhiều hơn 1 cấu hình.

Miêu tả giao diện có thể được xem miêu tả của một nhóm Endpoint phối hợp với nhau để thực hiện một tính năng của thiết bị. Ví dụ: bạn có một thiết bị đa chức năng như fax/scanner/printer chẳng hạn. Miêu tả giao diện 1 có thể là các Endpoint của chức năng fax; giao diện thứ 2 có thể là của chức năng scanner và giao diện thứ 3 dành cho chức năng printer. Không giống miêu tả cấu hình, không có giới về số lượng được phép thực hiện. Tức là một thiết bị có thể có 1 hoặc nhiều miêu tả giao diện cùng chạy tại 1 thời điểm.

Mỗi miêu tả giao diện có 1 trường là **bInterfaceNumber** để chỉ số của Giao diện và **bAlternateSetting**, sẽ cho phép một giao diện thay đổi bất cứ lúc nào. Ví dụ, nếu một thiết bị có 2 giao diện, giao diện 1 và giao diện 2 chẳng hạn. Giao diện thứ nhất sẽ giá trị **bInterfaceNumber** được gán là zero (tức là giao diện đầu tiên), và **bAlternateSetting** cũng là 0 (tức là cấu hình mặc định). Giao diện thứ 2 sẽ có **bInterfaceNumber** là 1, tức là chỉ giao diện thứ 2 và cũng **bAlternateSetting** được gán là 0 (tức là cấu hình mặc định). Ta cũng có thể đưa ra một miêu tả giao diện khác với  **bInterfaceNumber** là 1 (giao diện 2), nhưng **bAlternateSetting** tức là chỉ một setting khác của giao diện 2. Khi cấu hình trên được sử dụng, tức là 2 miêu tả giao diện trên sẽ được sử dụng với **bAlternativeSettings** là zero. Tuy nhiên, trong quá trình sử dụng, Host có thể gửi một yêu cầu **SetInterface** đến interface 1 yêu cầu nó chuyển sang setting thay thế. Khi đó nó sẽ chạy trên một miêu tả giao diện khác.

Mỗi miêu tả Endpoint được sử dụng để chỉ rõ loại truyền, hướng truyền, tần số polling (hỏi thiết bị) và kích thước lớn nhất của packet cho mỗi Endpoint. Endpoint Zero mặc định được sủ dụng là Endpoint điều khiển và nó không bao giờ có miêu tả.

1. Sự kết hợp các loại đặc tả (Composition of USB Descriptors)

Tất cả các miêu tả từ thiết bị, cấu hình, giao diện etc, đều có một định dạng chung nhất. Byte đầu tiên chỉ ra độ dài của miêu tả, trong khi byte thứ 2 chỉ loại đặc tả. Nếu độ dài của một đặc tả nhỏ hơn kích thước mà đặc trả USB đã định nghĩa thì Host sẽ bỏ qua luôn. Tuy nhiên, nếu kích thước lớn hơn thì Host sẽ bỏ phần thừa ra và tìm miêu tả kế thiếp từ vị trí cuối cùng độ dài thực tế.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Offset | Field | Size | Value | Ý nghĩa |
| 0 | bLength | 1 | Number | Kích thước Descriptor tính theo Bytes |
| 1 | bDescriptionType | 1 | Constant | Loại đặc tả |
| 2 | … | n |  | Các tham số của đặc tả |

1. Miêu tả thiết bị (Device Descriptors)

Sẽ miêu tả toàn bộ thông tin về thiết bị. Tất nhiên, vì vậy nó chỉ có 1 thôi. Nó diễn tả một vài thông tin cơ bản, trong đó có một vài thông tin quan trọng như phiên bản USB nó hỗ trợ, kích thước tối đa của packet, nhà sản xuất (Vendor), mã sản phầm. Số lượng cấu hình mà thiết bị có. Định dạng được miêu tả dưới bảng sau:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Offset | Field | Size | Value | Description |
| 0 | bLength | 1 | Number | Kích thước của Descriptor (18 bytes) |
| 1 | bDescriptorType | 1 | Constant | Là loại Device Descriptors (0x01) |
| 2 | bcdUSB | 2 | BCD | Phiển bản đặc tả USB mà thiết bị tuân theo. |
| 4 | bDeviceClass | 1 | Class | Max class (Được gán bởi USB Org)  Nếu nó bằng 0 thì mỗi interface nên có class riêng của nó.  Nếu là 0xFF thì class này do Vendor tự quy định.  Ngoài ra thì mọi giá trị đều hợp lệ. |
| 5 | bDeviceSubClass | 1 | SubClass | Mã của Subclass  (Được gán bởi USB Org) |
| 6 | bDeviceProtocol | 1 | Protocol | Mã protocol (Assigned by USB Org) |
| 7 | bMaxPacketSize | 1 | Number | Kích thước lớn nhất của Packet khi sử dụng Endpoin Zero (mặc định phải có). Các giá trị có thể là 8, 16, 32, 64 |
| 8 | idVendor | 2 | ID | Vendor ID (Assigned by USB Org) |
| 10 | idProduct | 2 | ID | Product ID (Assigned by Manufacturer) |
| 12 | bcdDevice | 2 | BCD | Device Release Number |
| 14 | iManufacturer | 1 | Index | Chỉ số của chuỗi mô tả nhà sản xuất |
| 15 | iProduct | 1 | Index | Chĩ số của chuỗi mô tả sản phẩm |
| 16 | iSerialNumber | 1 | Index | Chỉ số của mã Serial của sản phẩm |
| 17 | bNumConfigurations | 1 | Integer | Số cấu hình hiện có của thiết bị |

* bcdUSB : phiên bản cao nhất của USB mà thiết bị hỗ trợ. Giá trị ở dạng 0xJJMN, JJ là phiên bản major, M là miên bản minor, N là sub minor. Ví dụ USB 2.0 sẽ là 0x0200, USB1.1 sẽ là 0x1100, và USB1.0 sẽ là 0x1000.
* bDeviceClass, bDeviceSubClass và bDeviceProtocol : Là thông số chủ yếu để hệ điều hành phía Host xác định driver cho thiết bị. Thông thường, chỉ bDeviceClass được set ở mức miêu tả thiết bị, còn 2 thông số còn lại được xác định tại các mức miêu tả interface. Điều này cho phép một thiết bị hỗ trợ nhiều class.
* bMaxPacketSize : Kích thước packet của Endpoint Zero. Tất cả các thiết bị bắt buộc phải support EndPoint Zero.
* idVendor và idProduct : được hệ điều hành phía Host sử dụng để tìm driver cho thiết bị, VendorID được gán bởi [*USB-IF*](http://www.usb.org/)*.*
* bcdDevice: có cùng định dạng với bcdUSB và được sử dụng để cung cấp phiên bản của thiết bị. Giá trị này sẽ được gán bởi developer người phát triển thiết bị.
* 3 chuỗi miêu tả sẽ tồn tại để cung cấp những thông tin chi tiết về nhà sản xuất, sản phẩm, số serial. Không bắt buộc phải có 3 chuỗi này, nếu không có 2 chuỗi này thì index zero sẽ được sử dụng.
* bNumConfigurations : định nghĩa số lượng cấu hình mà thiết bị hỗ trợ tại tốc độ hiện tại của nó.