**■ USB Interface 정의 및 개발 배경**

**1. USB Interface 정의**

1) Universal Serial Bus (범용 직렬 버스)

2) INTEL, MS, COMPAQ, DEC, IBM, NOTEL, NEC 등 7개사가 공동으로 제안

3) 주변 기기 등을 개인용 컴퓨터(PC)에 접속하기 위한 인터페이스의 공동화를 목적으로 개발

**2. USB Interface 개발 배경**

1) 제한된 IRQ 해결 방안 필요

2) 고속의 디바이스 필요 (Parallel의 단점 극복 목적)

3) 손쉬운 디바이스 접속 및 설치

4) 다양한 하드웨어가 접속 가능한 버스 필요

5) 작은 크기의 커넥터 필요

**■ USB Interface 특징**

**1. 넓은 확장 성 제공**

 ▪ 최대 127개의 디바이스를 Daisy-Chain 방식으로 연결 가능

 ▪ 최대 5m 까지의 케이블 사용 가능

**2. PnP 지원**

 ▪ Jumper Setting 없이 하드웨어설치가능

 ▪ 포트에 꽂으면 자동으로 감지 (동시에 적절한 드라이버를 호출)

 ▪ 사용이 끝나면 IRQ, Port Address등을 염려 않고 빼도, 자동으로 인식 및 종료 (드라이버 해제)

**3. Hot Plug In 지원**

 ▪ 동작 도중 하드웨어 탈.부착 가능

**4. 버스 자체 전원 지원**

 ▪ 버스 내에서 전원 공급 (+5V, 총500mA 지원)

**5. 규격화 된 소형 커넥터**

 ▪ 휴대장치에 적합한 소형 크기

 ▪ 휴대 성 증대

 ▪ 규격화 된 커넥터 사용으로 인한 포트 수의 획기적 감소

**■ USB Interface 버전 별 특징**

**▪ USB 1.1**

\* 12Mbits/s의 Full-Speed Mode, 1.5Mbits/s의 Low-Speed Mode 지원

\* Low-Speed 모드는 Full-Speed에서 지원해야 하는 질높은 케이블의 재질을 값싼 재질로 대체하기 위한 규격

\* EMI 전자파 내성에 강함 (Low-Speed 지원 장치 예 : USB 마우스, USB 키보드 등)

**▪ USB 2.0**

\* 데스크 탑 컴퓨터들에서 480Mbits/s의 High Speed로 사용 됨

\* IEEE 1394에 대응하기 위한 인터페이스

**▪ USB 3.0**

\* 5.0Gbit/s의 Super-Speed Mode 지원 (27Gbyte의 데이터를 1분만에 전송)

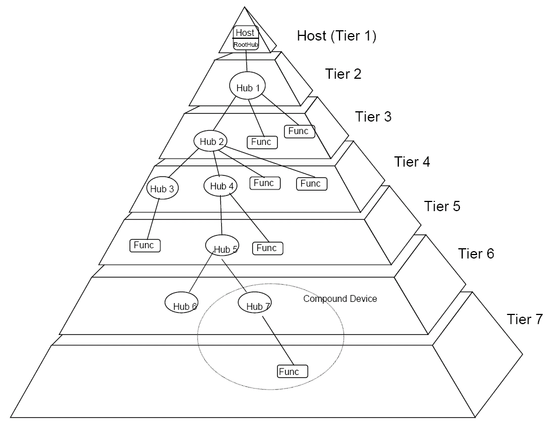
\* 한정 된 속도의 USB 2.0 차기 버전

\* USB1.1, 2.0 규격과는 다른 방식의 커넥터 구조

**■ USB Interface Vendor LOGO**

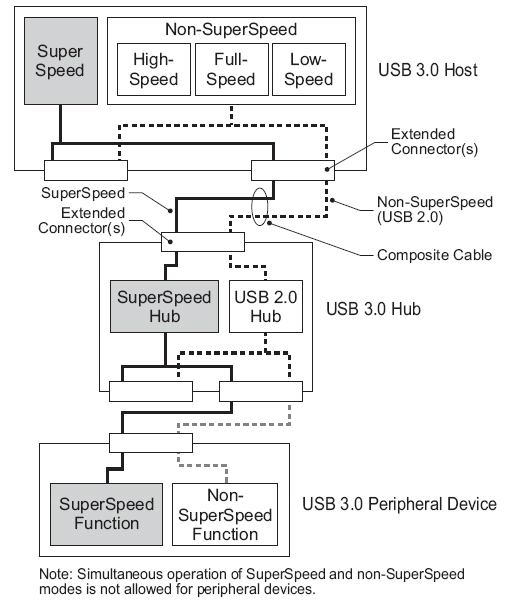


**■ USB 2.0 버스 인터페이스 구조**



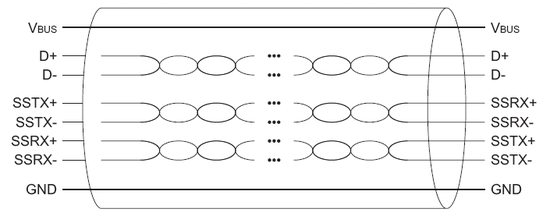
**- USB 2.0 Bus 형태의 구조도 -**

**■ USB 3.0 버스 인터페이스 구조**

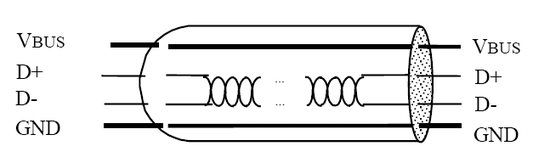
****

**- USB 3.0 Bus 형태의 구조도 -**

**■ USB 2.0 / 3.0 간 케이블 구조 및 인터페이스 신호 비교**



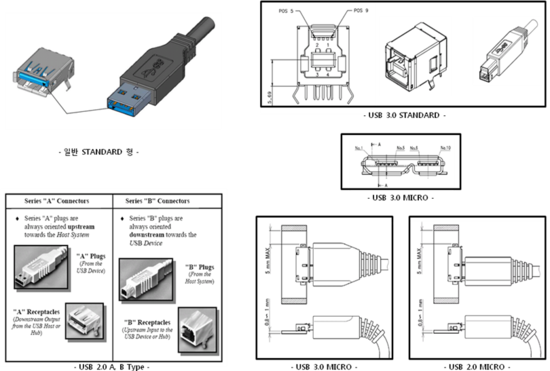
**- USB 3.0 케이블 구조 -**



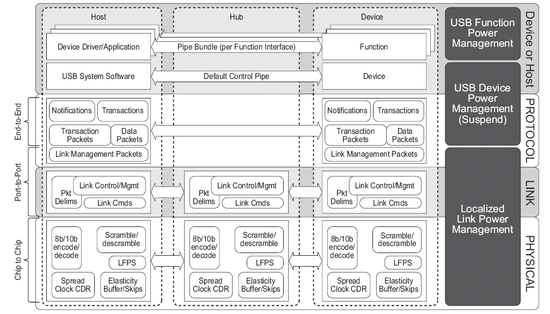
**- USB 2.0 케이블 구조 -**

|  |  |  |
| --- | --- | --- |
| **항목** | **USB 3.0** | **USB 2.0** |
| **속도** | ▪ Super-Speed (5.0 Gbps) | ▪ Low-Speed (1.5 Mbps)  ▪ Full-Speed (12 Mbps)  ▪ High-Speed (480 Mbps) |
| **데이터**  **인터페이스** | ▪ 2개의 단일 신호  ▪ 4개의 Super-Speed  Differential 신호  ▪ 양 방향의 데이터 흐름 | ▪ 2개의 Half-Duplex  Differential 신호  ▪ 단 방향의 데이터 흐름 |
| **케이블** | ▪ 4개의 Super-Speed 데이터 라인  ▪ 2개의 기존 USB 2.0 데이터 라인 | ▪ 2개의 데이터 라인 |

**■ USB 커넥터 구조**



**■ USB 시스템 LAYER 구조**

****

**- USB 시스템 LAYER 구조도-**

**■ USB 통신 방식**

**- USB 통신 방식**

▪ 모든 Data 통신은 Host의 Data IN/OUT 명령으로 이루어 짐

(Client 측에서 IN/OUT 명령은 불가)

(1) Host의 요청에 따라서 Client 측에서 전달 - IN    Transaction (Client => Host)

(2) Host가 Client 측으로 전달                       - OUT  Transaction (Host => Client)

**■ End-Point**

**1. End-Point 정의**

▪ USB 버스를 통해서 Client로 전달 될 Data / Host 에 전달 될 Data를 저장하는 공간

▪ Data의 사용목적에 따라서 전송방향과 형태가 결정 (Control, Bulk, Interrupt, Isochronous)

**2. End-Point Zero ?**

▪ USB Client 내 End-Point들은 번호가 부여

▪ USB Client는 반드시 End-Point Zero(0번) - Default End-Point 혹은 Control End-Point라 부름

▪ 장치 Configuration 및 Control 하기 위한 용도로 사용

▪ End-Point 중 유일한 양방향(IN / OUT) Data 전송을 지원

**[출처]** [USB 기술 자료 - (1) USB Interface](http://blog.naver.com/insopack77/20060730469)|**작성자** [인소팩](http://blog.naver.com/insopack77)

**I. USB Descriptor 정의**

  **DEVICE DESCRIPTOR 정의 및 종류**

**1. USB Descriptor 정의**

\* USB는 대표적인 PnP (Plug & Play)를 지원하는 인터페이스

   => 디바이스에 대한 정보 및 설정 사항을 알기 위해 Descriptor를 읽어 와야 함

   => Host가 Device에게 Device에 대한 정보를 요구하고, Device가 자신의 정보를 전달 이 때 사용하는 정보 의미

\* 연결된 디바이스의 종류를 알게 되고, 디바이스의 특성에 맞게 데이터 전송 양을 조절 할 수 있음

\* USB Enumeration (열거) 과정에서 중요하게 사용

**2. USB Descriptor 종류**

**(1) Device Descriptor**: 디바이스에 대한 일반정보, 단 하나의 descriptor존재

**(2) Configuration Descriptor**      : 하나 또는 이상의 Descriptor 기술 가능

                                                    : 여러 가지의 인터페이스 기술 가능

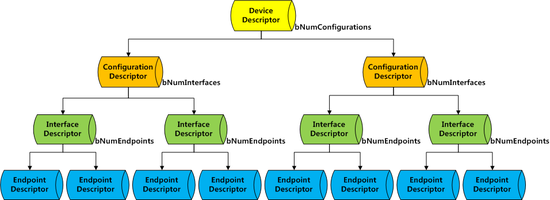
                                                    : 인터페이스 또한 여러 개의 End-Point 정의 기술 가능

**(3) Interface Descriptor**: 인터페이스 내용기술, Alternate 셋팅을  가질 수 있음

**(4) Endpoint Descriptor**: pipe, endpoint0은 descriptor가 존재하지 않음

**(5) String Descriptor**: unicode format, vender이름, 디바이스 이름, serial number

**3. USB Descriptor 구조**



**Ⅱ. USB Descriptor 분류**

**1. Device Descriptor**

▪ Device의 모든 것을 표현

▪ USB Device는 단지 하나의 Device Descriptor를 가질 수 있음

▪ Device Descriptor는 Device에서 지원하는 USB Revision 명시

▪ 적절한 USB 드라이버를 읽어올 시에 사용되는 Product ID와 Vendor ID 명시

▪ Device가 가질 수 있는 가능한 Configuration 수 명시

▪ Configuration의 값은 얼마나 많은 Configuration Descriptor로 분기할 수 있는지 나타냄

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Offset** | **Field** | **Size** | **Symbol** | **Description** |
| **0** | bLength | 1 | Number | 바이트 단위 Descriptor 크기 |
| **1** | bDescriptorType | 1 | Constant | Device를 나타내는 상수 |
| **2** | bcdUSB | 2 | BCD | USB Version Release 번호 |
| **4** | bDeviceClass | 1 | Class | 클래스 코드 |
| **5** | bDeviceSubClass | 1 | SubClass | 하위 클래스 코드 |
| **6** | bDeviceProtocol | 1 | Protocol | 프로토콜 코드 |
| **7** | bMaxPacketSize0 | 1 | Number | Endpoint 0의 최대 Packet 크기 |
| **8** | idVendor | 2 | ID | 제조사 ID (USB-IF 할당) |
| **10** | idProduct | 2 | ID | 제품 ID (제조사 할당) |
| **12** | bcdDevice | 2 | BCD | Device Release 번호 |
| **14** | iManufacturer | 1 | Index | 제조사 문자열 Descriptor Index |
| **15** | iProduct | 1 | Index | 제품 문자열 Descriptor Index |
| **16** | iSerialNumber | 1 | Index | 시리얼 번호를 포함한 문자열 Descriptor Index |
| **17** | bNumConfigurations | 1 | Number | 설정 가능한 수 |

**2. Configuration Descriptor**

▪ 여러 개의 다른 Configurations을 가질 수 있음 ex) 외장 HDD PWR

▪ Power 소모량 및 Device가 Self-Powered인지 Bus Powered인지, 가지고 있는 Interface 수 등의 정보를 포함

▪ Device가 Enumerate될 때 Host는 Device Descriptor를 읽고 어떤 Configuration이 활성화 될 것인지를 결정

▪ 한번에 한가지 Configuration만 가능

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Offset** | **Field** | **Size** | **Symbol** | **Description** |
| **0** | bLength | 1 | Number | 바이트 단위 Descriptor 크기 |
| **1** | bDescriptorType | 1 | Constant | Configuration을 나타내는 상수 |
| **2** | wTotalLength | 2 | Number | Configuration Descriptor와 전체 종속  Descriptor의 총 바이트 수 |
| **4** | bNumInterfaces | 1 | Number | Configuration의 Interface 수 |
| **5** | bConfigurationValue | 1 | Number | Set Configuration과 Get Configuration  Request용 식별 자 |
| **6** | iConfiguration | 1 | Index | Configuration에 대한 문자열 Descriptor  Index 문자열 |
| **7** | bmAttributes | 1 | Bitmap | 자체 / 버스 전력과 원격 깨움 설정 |
| **8** | bMaxPower | 1 | mA | Device가 필요한 버스 전력 |

**3. Interface Descriptor**

▪ Device의 한가지 Feature를 나타내는 기능적 그룹화 과정 또는 Header로 보일 수 있음

Ex) Fax / Scanner / Printer가 되는 Multi-Function Device 시 Fax, Scanner, Printer각 기능별로Interface Descriptor 존재

▪ Configuration Descriptor와는 달리 한번에 하나의 Interface Descriptor만 활성화 해야한다는 등의 제한 없음

▪ Interface의 수를 나타내는 bInterfaceNumber Field, Interface에 Setting을 변화시킬 수 있는bAlternateSetting Field가 존재

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Offset** | **Field** | **Size** | **Symbol** | **Description** |
| **0** | bLength | 1 | Number | 바이트 단위 Descriptor 크기 |
| **1** | bDescriptorType | 1 | Constant | Interface를 나타내는 상수 |
| **2** | bInterfaceNumber | 1 | Number | Interface 식별 번호 |
| **3** | bAlternateSetting | 1 | Number | - |
| **4** | bNumEndpoints | 1 | Number | 지원되는 Endpoint 수 (Endpoint 0 제외) |
| **5** | bInterfaceClass | 1 | Class | 클래스 코드 |
| **6** | bInterfaceSubClass | 1 | SubClass | 서브 클래스 코드 |
| **7** | bInterfaceProtocol | 1 | Protocol | 프로토콜 코드 |
| **8** | iInterface | 1 | Index | Interface를 위한 문자열 Descriptor Index |

**4. Endpoint Descriptor**

▪ End-Point (Control Endpoint를 제외)를 설명하는데  사용

▪ End-Point 0는 항상 Control Endpoint로 사용되며 어떤 Descriptor의 Request이전에 이미 설정

▪ Host는 이들 Descriptor로부터 받은 정보를 Bus의 Band Width를 정하는 데 사용

▪ Device가 어떠한 Endpoint를 지원하여도 Endpoint Description 상에 명시 되어 있지 않을 시Endpoint 사용 불가

   => Endpoint를 이용한 Data 전송 통로인 Pipe를 Host가 생성해주지 않음

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Offset** | **Field** | **Size** | **Symbol** | **Description** |
| **0** | bLength | 1 | Number | 바이트 단위 Descriptor 크기 |
| **1** | bDescriptorType | 1 | Constant | Endpoint를 나타내는 상수 |
| **2** | bEndpointAddress | 1 | Endpoint | Endpoint 번호와 방향 |
| **3** | bmAttributes | 1 | Bitmap | 전송 타입 |
| **4** | wMaxPacketSize | 2 | Number | 최대 Packet 크기 |
| **6** | bInterval | 1 | Number | High-Speed BULK OUT Endpoint일 경우 최대NAK 속도 의미 , 다른 BULK Endpoint일 경우 무시 |

**Ⅲ. USB Enumeration**

**1. USB Enumeration 정의**

**■ USB Enumeration**

▪ USB host의 역할 중 하나인 Bus의 상태를 주기적으로 계속 확인

▪ 새로운 Device (Client device)가 Bus 에 연결되었는지 또는 Device가 Bus에서 제거되었는지 확인

▪ 새로운 Device가 Bus에 연결되면 host는 Device에 Control Packet을 보내는 것으로 Enumeration 시작

▪ Control Packet 전송은 Device의 Endpoint 0을 통해서 이루어 짐

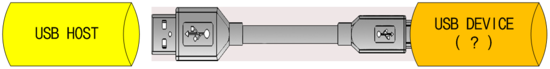
▪ Enumeration과정 중 Device & Host의 주 역할

             - Device :  Host의 요구에 따라 Device descriptor를 제공

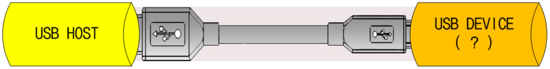
             - Host     :  Descriptor를 기반으로 적절한 Device Driver를 로드 하여 USB Device를 설정

**2. USB Enumeration 순서**

**(1) USB DEVICE 장치 연결 안 됨**



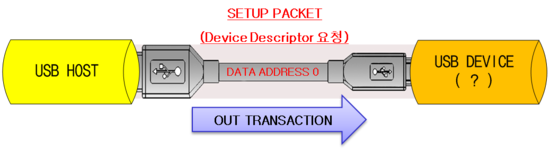
**(2) USB DEVICE 장치 Cable을 통한 연결 상태**



**(3) USB HOST 측 Data Address 임시 할당 상태**



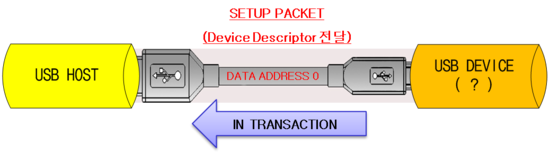
**(4) 임시 Data Address를 통한 Device Descriptor 요청**



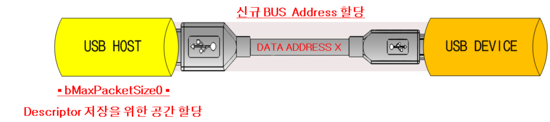
**(5) Host 측에서 송신 된 Device Descriptor 요청 수신**



**(6) 임시 Data Address를 통한 Device Descriptor 전송**



**(7) Host 측에서 송신 된 Device Descriptor 요청 수신**

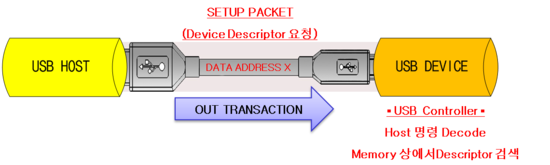


|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Size** | **Value** | **Description** |
| bMaxPacketSize0 | 1 | Number | Endpoint 0의 최대 Packet 크기 |

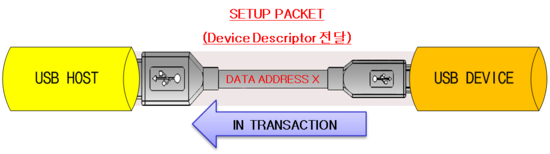
▪ Device Descriptor 내부 8번째 byte 값을 통하여 Descriptor 저장을 위한 16진수 byte 공간 할당

▪ 할당 이후부터는 해당 주소로 Device 호출 (Device 할당 주소 외의 명령은 무시)

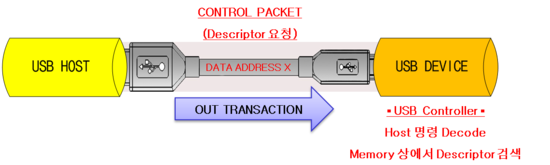
**(8) Data Address를 통한 Device Descriptor 요청 및 수신**



**(9) Data Address를 통한 Device Descriptor 전송**



**(10) Device Descriptor를 제외한 나머지 Descriptor 요청 및 수신**



**(11) Data Address를 통한 나머지 Descriptor 전송 및 Enumeration 종료**

**[출처]** [USB 기술 자료 - (2) USB Descriptor, Enumeration](http://blog.naver.com/insopack77/20061074373)|**작성자** [인소팩](http://blog.naver.com/insopack77)

**I. End-Point 및 전송 방식 이해**

**1. End-Point 세부 개념 이해**

**(1) End-Point 개념**

1) 데이터를 저장할 수 있는 버퍼

2) 데이터의 메모리 블록 또는 USB 칩의 레지스터

3) End-Point에 저장 된 데이터 : 송신 또는 수신하기 위하여 대기중인 데이터 의미

**(2) End-Point 특성**

1) 단 방향으로 동작 (Control End-Point 제외)

2) End-Point 주소는 End-Point 번호와 전송 방향으로 결정

3) 전송 방향은 Host 관점에서 해석

4) 모든 디바이스에서 End-Point 0은 Control End-Point로 사용

5) 표현 방식 : End-Point 1 IN / End-Point 1 OUT

6) Full-Speed, High-Speed : 30개 / Low-Speed : 2개

**(3) End-Point 0**

1) Control End-Point / Default End-Point로 명칭

2) Default Pipe를 사용하여 HOST와 Configuration을 수행

**2. 전송 인터페이스 방식**

**(1) Bulk 전송 방식**

1) 대량의 데이터를 고 신뢰성으로 전송하기 위한 모드

2) USB BUS가 다른 전송으로 바쁘지 않는 상황에서는 벌크 전송이 가장 빠른 전송 방식

3) 최대 Packet 크기

**(2) Interrupt 전송 방식**

1) 데이터를 주기적으로 전송해야 할 경우 유용

2) 일반적으로 키보드, 마우스, 장치 상태 보고 등에 사용

3) 디바이스가 데이터를 전송하면 Host 측에서 즉각적 인터럽트 발생 (Polling 필수)

4) 모든 전송 방식에서 사용 가능 (HID 디바이스는 반드시 Interrupt IN 지원)

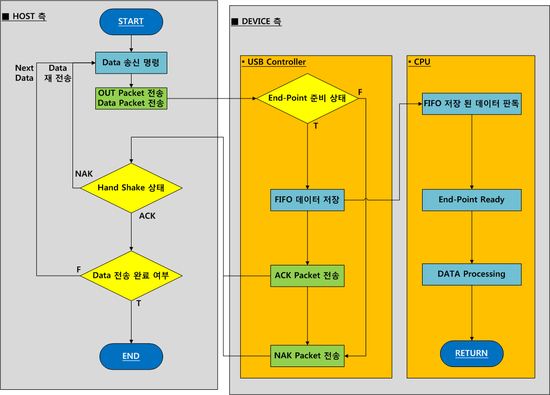
5) 최대 Packet 크기

**(3) Isochronous  전송 방식**

1) Streaming 및 실시간 전송에 적합

2) 데이터 전송 시 일정한 속도, 시간을 보장하기 때문에 Host에서 전송 완료 시간 예측이 가능

**Ⅱ. HOST-CLIENT 간 통신 흐름 분석**

****

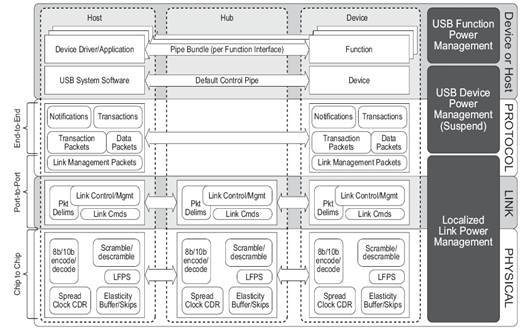
                                    -      **HOST-CLIENT 간 USB 통신 흐름도 -**

**▪ USB Controller**: Host에서 전송 된 Data의 수신 완료 상태를 인터럽트 신호를 통해 CPU 측으로 전달

**▪ CPU**: 인터럽트 수신 시, FIFO 데이터를 메모리에 저장 후 End-Point를 준비 상태로 전환 및 인터럽트에 해당하는 작업을 수행

**Ⅲ. TRANSACTION 개념 이해**

**1. USB SYSTEM LAYER**

****

**- USB 시스템 LAYER 구조도-**

**2. TRANSACTION 개념 정의**

**■ Transaction 정의**

(1) 서비스를 End-Point로 전달

(2) 서비스 정의 : Host가 Device 측으로 정보를 보내거나, 요청해서 받는 것

(3) 각 Transaction은 데이터, 식별, 에러확인, 상태, 제어정보 포함

(4) Data가 적을 시 : 1 Transaction에 모두 전송

    Data가 많을 시 : 여러 Transaction에 나누어 전송

**3. 전송 방식에 따른 TRANSACTION 분류**

**■ Transaction Packet 분류**

**▪ Token        Packet**: Device Address / Transaction이 향할 End-Point 번호를 담고 있음

**▪ Data        Packet**: Transaction 과정에서 Host 혹은 Device가 전송하는 데이터를 포함

**▪ Handshake Packet**: Data Packet을 받은 쪽에서 보내는 데이터 / Data를 오류 없이 받았는지 확인 가능

|  |  |  |  |
| --- | --- | --- | --- |
| **전송 방식** | **전송 스테이지** | **Transaction** | **Packet 종류** |
| **CONTROL** | Setup Stage | 1개 | Token  Data  Handshake |
| Data Stage | NONE 또는 1개 이상  (IN / OUT) |
| Status Stage | 1개 |
| **BULK** | - | 1개 또는 그 이상  (IN / OUT) |
| **INTERRUPT** | - | 1개 또는 그 이상  (IN / OUT) |
| **ISOCHRONOUS** | - | 1개 또는 그 이상  (IN / OUT) | Token  Data |

**- 전송 방식에 따른 TRANSACTION 분류-**

**4. 세부 USB PACKET 전송 리스트**

|  |  |  |  |
| --- | --- | --- | --- |
| **PID 타입** | **PID 명** | **PID <3:0>** | **설명** |
| **Token** | **OUT** | 0001B | End-Point로 전송 |
| **IN** | 1001B | Host로 전송 |
| **SOF** | 0101B | Frame 시작 |
| **SETUP** | 1101B | Setup |
| **Data** | **DATA0** | 0011B | Data Packet PID 짝수 |
| **DATA1** | 1011B | Data Packet PID 홀수 |
| **DATA2** | 0111B | Data Packet PID High-Speed |
| **MDATA** | 1111B | Data Packet PID High-Speed |
| **Handshake** | **ACK** | 0010B | Data 수신 성공 |
| **NAK** | 1010B | Data 전송 불가 |
| **STALL** | 1110B | End-Point 정지 상태  혹은 Control Pipe의 요청을 지원하지 않음 |
| **NYET** | 0110B | 현재 이후 데이터 처리 준비 안됨 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PID**  **타입** | **PID 명** | **PID**  **<3:0>** | **Source** | **설명** |
| **Special** | **PRE** | 1100B | Host | Hub에게 다음 Packet이 LS Packet이라고 알림 |
| **ERR** | 1100B | Hub | Hub가 분할 Transaction에서 LS / FS 에러 리포트 |
| **SPLIT** | 1000B | Host | 분할 Transaction을 타나내는 토큰 Packet 앞에 옴 |
| **PING** | 0100B | Host | NYET 후 Busy Check로 HS에서만 사용 |
| **Reserved** | 000B | - | Reserved |

**5. HANDSHAKE PACKET 세부 분석**

**■ Handshake Packet 정의**

(1) RS232 통신 상 RTS, CTS, XON, XOFF와 같은 역할

(2) Isochronous 방식을 제외 한 나머지 방식에서 모두 사용 (Control 전송 방식에서는 Status Stage를 사용)

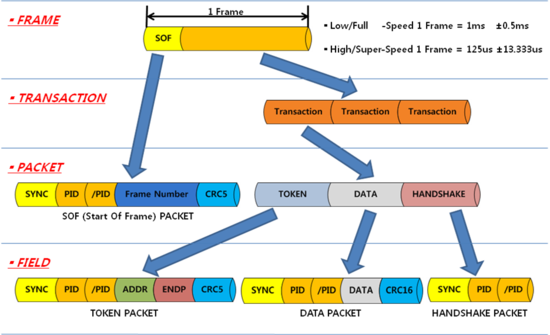
(3) 각 Transaction은 데이터, 식별, 에러확인, 상태, 제어정보 포함

(4) 위치, 소스, Handshake 신호의 내용은 Transaction의 종류에 따라 다름

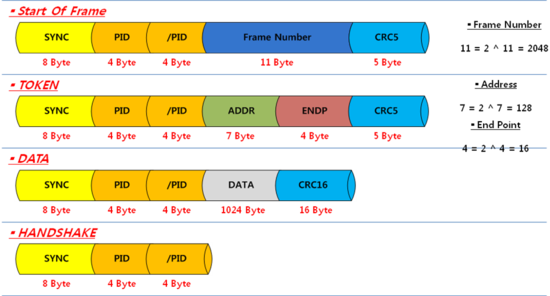
|  |  |
| --- | --- |
| **전송 방식** | **Transaction** |
| **ACK** | 데이터 전송 성공 |
| **NAK** | 다른 일로 바쁨 / Host는 나중에 다시 시도 |
| **STALL** | 제어 전송에 담긴 요청을 지원하지 않음 / End-Point 상 Halt 설정 됨 |
| **NYET** | End-Point가 현재 Transaction 데이터는 수락하지만  그 이상의 데이터는 처리준비가 되지 않음 |

**Ⅳ. USB PACKET 분석**

**1. PACKET 별 구조 흐름도**



**2. PACKET 별 세부 통신 사이즈 분석**

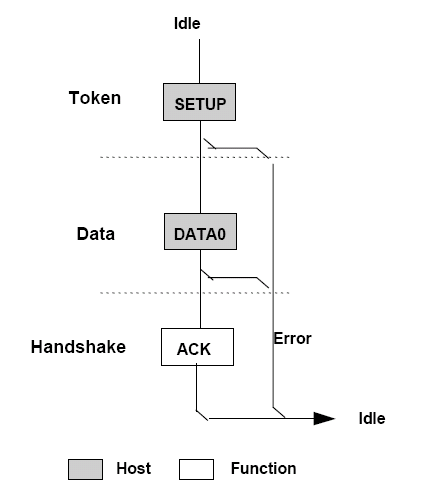


▪ 0부터 2048 까지 Frame 진행 뒤, 0으로 초기화하여 Frame Number 재 시작

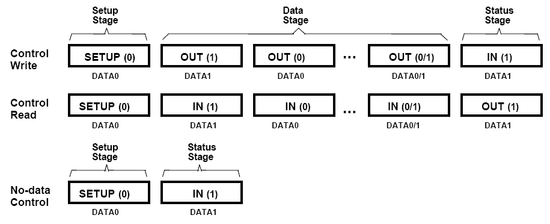
▪ TOKEN Packet은 오직 Host에서만 전송이 가능

**Ⅴ. 전송방식에 따른 TRANSACTION TYPE**

**1. CONTROL TRANSACTION FORMAT**

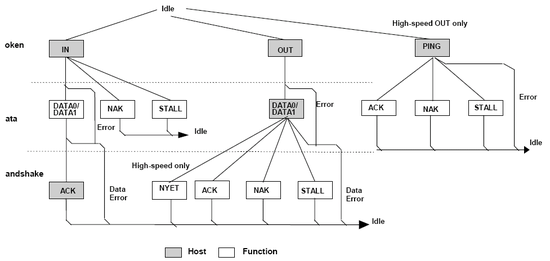
****

     - **Control SETUP Transaction Format Flow -**



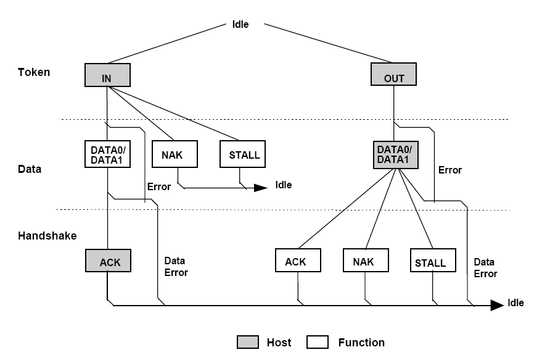
**- Control Read, Write Sequence -**

**2. BULK TRANSACTION FORMAT**

****

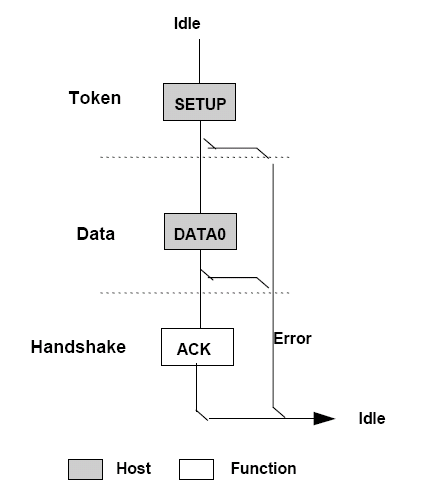
**- Bulk Transaction Format Flow -**

**3. INTERRUPT TRANSACTION FORMAT**

****

**- Interrupt Transaction Format Flow -**

**4. ISOCHRONOUS TRANSACTION FORMAT**

****

**- Isochronous Transaction Format Flow -**

**[출처]** [USB 기술 자료 - (3) USB Protocol](http://blog.naver.com/insopack77/20061102959)|**작성자** [인소팩](http://blog.naver.com/insopack77)