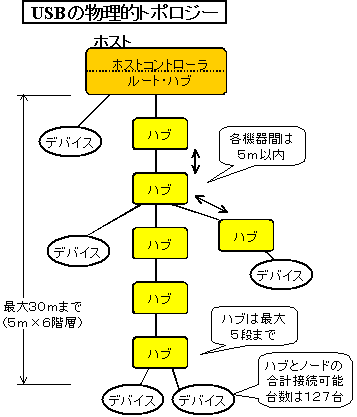
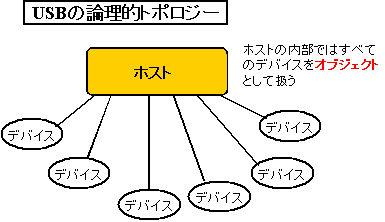
**USBの基本アーキテクチャ**

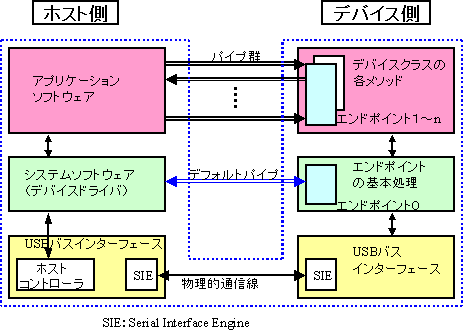
**【物理的なアーキテクチャ】**

　USBでの物理的な構成は下図のようになっていて、１台のパソコンが親機となって、  
ポーリング（問い合わせ）によって子機側と通信をします。  
　すべての通信の制御を親機が制御し、子機側から勝手に通信を開始することは出来  
ないようになっています。  
　このような構成図をネットワークとして見て、全体構成を**トポロジー**（Topology)と呼び、  
親機を**ホスト**（Host)、子機を**ノード**(Node)と呼んでいます。  
実際の使用状態では、ノードには「**デバイス**」と「**ハブデバイス**」（ハブと呼ぶ）とがあり  
ます。  
  
下図のように、ホストと呼ばれる中身は、処理装置となるホストコントローラと、最初の  
USBバスの接続部となるルートハブが一体化されたものとなっています。  
そして実際のモノとしては、これがパソコン本体になります。  
　このルートハブに直接ノードとなるデバイスを接続するこも出来ますし、さらにデバイス  
の台数が多いときには、ハブを挿入して台数を増やすことが出来ます。  
市販されているハブは、４台から８台程度のデバイスを接続することが可能になってい  
ます。しかし、ハブの従属接続は５段までと制限されています。  
　ホストの内部ではハブとデバイスすべてにアドレスが付与されて管理されます。この  
アドレスが最大１２７個なので、USBに接続可能なデバイスは最大１２７台ということに  
なります。（ルートハブはアドレスが不要なので含まれません）  
  


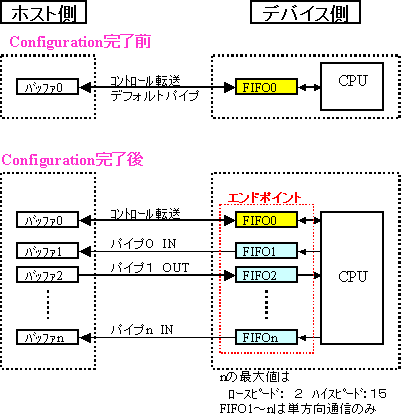
**【論理構成とデバイスクラス】**

上記はUSBの物理的な構成ですが、ホストの内部でソフトウェアから見た論理的な  
構成は下図のような単純な構成になっています。  
  
  
  
  
  
　ホストの中では、各デバイス（ノードとハブ両方を含む）を、**オブジェクト**として扱って  
います。従って、オブジェクトとなる各デバイスは、「**クラスの概念**」を持っており、  
それぞれに「**デバイスクラス**」を持っています。　基本のデバイスクラスはひとつで、  
いろいろな種類のデバイスはその基本のデバイスクラスの派生クラスとなっています。  
  
このオブジェクト指向の概念を使っているため、見かけ上、ホストからデバイスへの  
アクセスはオブジェクトへのアクセスという形となります。つまり、「**メッセージ**」のやり  
取りでデータの送受が実行されます。  
具体的には、get・・・( )　と set･･･( ) コマンドだけでデータの送受を行うことになります。  
  
そしてそのメッセージの処理を実行する「**メソッド**」の部分がデバイスの中のファーム  
ウェアの処理で実現されることになるわけです。  
　従って、デバイス側はメッセージに対応したメソッド部分だけを実装すれば、全体の  
制御はホストが実行してくれるので、デバイスは簡単な構成で処理を実現することが  
可能となります。

**【通信のアーキテクチャ】**

USBでの通信の全体構成は下図で表すことが出来ます。  
  
(1) **物理レベルの通信**  
　まず、物理的な通信はWireつまり実際のケーブル接続で行われます。この時に  
　実際に物理レベルでの通信を実行するのは、USBコントローラのICの役割となって  
　いますので、使う側からはまったく物理的な通信については知らなくともUSBを使う  
　ことが可能になるようになっています。  
　この１本のケーブル上で通信されるので、瞬間的には１個のデータの送受信しか  
　出来ない訳ですが、タイムシェアをして見かけ上、いくつかの通信線でつながって  
　いるような使い方をします。  
  
(2) **システムレベルの通信**  
　次のレベルではホストのシステムソフトウェア（Windowsのデバイスドライバレベル）での  
　通信で、これは必須の通信となりますので、ホストとデバイス間には「**デフォルトパイプ**」  
　という通信の論理接続を用意します。  
　そしてこのデフォルトパイプを使って行う通信を、「**コントロール転送**」と呼んでいます。  
　このコントロール転送により、リセットで初期化されたときに、システム設定を初期化し  
　たり、USBケーブルが接続された時に、各種の設定制御をするために通信ができるよう  
　にします。  
　つまり、このコントロール転送を使って、「**コンフィギュレーション**」を行い、デバイスの  
　使い方の設定情報をホストとデバイス間でやりとりします。  
　これで使用するパイプの本数や、転送モードなどの設定情報をホストがデバイスに要求し、  
　それを元にしてホストがデバイスの使用条件を設定します。  
  
(3) **アプリケーションレベルの通信**  
　次のレベルはアプリケーションのレベルで、この間の論理的な通信線を「**パイプ**」と  
　呼び、必要に応じて複数のパイプを設定することが出来ます。  
　このパイプはあくまでも論理的な通信線ですから、実際の通信は１本のUSBの線上で  
　時分割で行われることになります。  
　このパイプを使った通信は、ホストから転送する「**OUT**」と、デバイス側から転送する  
　「**IN**」のどちらか片方向だけの通信となっています。  
　これに対し、コントロール転送を行うデフォルトパイプは双方向通信となっています。  
  
  


**【エンドポイントとパイプ】**

USBでの論理的な通信はエンドポイントとパイプの概念で表現されます。  
これを図で表すと下図のようになっています。  
つまり、デバイス側には「**エンドポイント**」と呼ばれる「**FIFOバッファ**」が通信のための  
実体となり、ホスト側にも同様のバッファが用意されて、このバッファ同士が「**パイプ**」で  
接続されて、データを送受することになります。  
  
まず、デバイスをUSBに接続した直後の状態は、コンフィギュレーション前の状態と  
なっていて、デフォルトパイプであるコントロール転送だけが通信のできる状態となって  
います。これに対応するのが「**エンドポイント０（FIFO0)**」となります  
  
このコントロール転送を使って、コンフィギュレーションを実行し、デバイスの使い方が  
設定されたあとは、下側のように、コントロール転送以外に、あらたなエンドポイントが  
追加され、パイプが構成されます。このパイプは、片方向の通信しか出来ないように  
なっています。このエンドポイントの番号はホストが指定することが出来ます。  
IN/OUTの方向はホストを中心に考え、ホストへの入力をIN、ホストからの出力をOUT  
と定義しています。  
またパイプの番号とエンドポイントの番号とは独立で、ホストが決めることが出来ます。  
  


**【USBの転送モード】**

上記のパイプを使ったUSBの通信の仕方には、「バルク転送」、「インタラプト転送」  
「アイソクロナス転送」があります。これにデフォルトパイプの「コントロール転送」を加えて  
全部で４つの転送方法があり、それぞれの特徴は下記のようになっています。  
（USB　Ver1.1仕様）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 項　目 | コントロール転送 | バルク転送 | インタラプト転送 | アイソクロナス転送 |
| 特　徴 | 少ないデータ量の 半二重通信 | 大容量データの 一括高信頼転送 | 小容量データの 定周期転送 | 一定時間内のデータ 量が保証された転送 |
| 用　途 | セットアップ、設定 パラメータ転送用 | 記憶装置。スキャナ などの大容量高速 データの転送 | 計測やマンマシン 機器のデータ転送 | 音声などのリアル タイムな転送 |
| 転送速度 | 1.5Mbps/12Mbps | 12Mbs | 1.5Mbps/12Mbps | 12Mbps |
| 転送周期 | 不定 | 不定 | Nmsec(N=1～255) | 1ms/フレーム |
| データ量 /ﾊﾟｹｯﾄ | 1～64バイト(フル） 1～8バイト(ロー) | 8/16/32/64バイト | 1～64バイト(フル) 1～8バイト（ロー) | 1～1023バイト |
| 信頼性 転送速度 遅延時間 | －(再送あり） － － | ◎(再送あり) ◎ △ | ◎(再送あり) ○ ◎ | △(再送なし) ◎ ◎ |

この転送モードのどれを使うかは、デバイス側がコンフィギュレーション用のデータと  
して持っていて、USBを接続した時にコントロール転送を使ってホストにその情報を渡し、  
ホストで確認したあと、ホスト側から設定されてからモードが確定します。  
この時ホスト側から、デバイスのアドレス付けと、パイプ番号とエンドポイント番号の  
割り付けが行われます。