

これからは USBでいこう!

桑野 雅彦

1. USB化の勧め

● MS-DOSからWindowsに

自作などでちょっとした周辺機器をつなごうとするとき、従来はパラレル・ポートやシリアル(RS-232-C)ポートなどのいわゆるレガシ・インターフェースを使うのが一般的でした。簡単なデジタル信号の入出力ならば、プリンタ・ポートを直接制御することで入出力が可能ですし、速度をそれほど必要としないのならばシリアル・ポート経由でコマンドやデータをやりとりすればよいわけです。

MS-DOSなどがメインの環境であった時代には、確かにこのような方法が非常に扱いやすかったのですが、パソコンのOSがWindowsに移り、プリンタ・ポートやシリアル・ポートがOSの管理下に置かれるようになるに従って、これらのポートはだんだんと扱いにくいものになってしまいました。ポートを勝手にアクセスすることも不可能ではありませんが、既存の周辺機器との衝突も考慮すると、いろいろと問題が出てきそうですし、機器をつなぐたびに本体の裏に回ってケーブルをつなぎかえるというのも、意外と煩わしいものです。

また、プリンタ・ポートやシリアル・ポートには電源ピンがないので、微小電流で動作するような回路を除けば、基本的にバッテリーやACアダプタなどから、

別途電源を供給する必要があります。単純なことなのですが、これが案外めんどうだというのは、実際に使ったことのある方ならばよくご存じのことでしょう。

これらの古典的なポートに代わる新たな拡張ポートとして注目したいのが、Windows98以降に標準でサポートされるようになったUSBです。表1にUSBとレガシ・インターフェースの比較を示します。

● USBのメリット

USBと聞くと思わず後ずさりしてしまいそうな方も多いことと思いますが、ここで改めてUSB化によるメリットを考えてみましょう。USBは電源供給ピンが用意されており、シリアル・ポートよりはるかに速い伝送速度を実現できます。また、最初から複数のデバイスを接続することを前提として考えられています。さらにプラグ&プレイが基本なので、必要なときにつないで、用が済めば切り離すということができる。さらにプラグ&プレイが基本なので、必要なときにつないで、用が済めば切り離すということができます。

仕様に基づいて作ればOSの管理下のデバイスとしてきちんと扱われるというのは、単に気分が良いというだけでなく、ほかの機器に悪影響を与えないで動作できるということにつながり、なによりも、より安心して利用することができます。

さらに、Windows98以降はマウスやジョイスティックなどのHID(ヒューマン・インターフェース・デバイス)ドライバなどが最初から組み込まれています。

表1
USBとレガシ・
インターフェースの比較

	USB	レガシ・インターフェース(シリアル/パラレルなど)
接続	1対N	1対1(ケーブル差し替え)
活線挿抜	可能	とくに配慮されていない
電源供給	あり(500mA)	なし(ACアダプタやバッテリーなど)
転送速度	480Mbps(ハイ・スピード) 12Mbps(フル・スピード) 1.5Mbps(ロー・スピード)	数kbps ~ 数百kbps程度
複数機器接続	可能	不可
プラグ&プレイ	可能	不可

そのため、これらのエミュレーションを行うようなアダプタを作ろうとした場合、ホスト側でプログラムを用意する必要が一切ありません。標準デバイスをエミュレートするようなUSB機器であるならば、ホスト側については何も考えずにターゲット側を作り込むだけで使えるようになるというわけです。

● USB機器を作るには

USBはこのように自作派にとっても多くのメリットがあります。しかし、いざUSB機器を作ろうとした場合、プリンタ・ポートやシリアル・ポートを使う場合に比べると、かなり敷居が高くなります。

まず第一に、RS-232-Cコントローラなどに比べると、USBコントローラが入手しやすいとは言いがたいものであることが挙げられます。シリアル・ポートを内蔵した1チップ・マイコンは数多くありますが、USBコントローラを内蔵したものはまだごく一部に限られます。

また、USBインターフェースを使った製作事例などがあまり発表されていないために、具体的な方法がなかなかわからないということも、敷居を高くしている要因といえるでしょう。

これからUSB機器を自作したいと考えている人に、ぜひ本書を読んでUSBの全体像を理解したり、USB機器設計の手がかりをつかんでいただければと思います。

● 実はUSBについて懐疑的だった!?

正直に言ってしまうと、筆者はUSBインターフェースについては非常に懐疑的でした。すでに接続トポロジとしてはUSBと大差のない100Base-TXのLANカードが1000円あまりで購入できるうえ、使い勝手に大きな差がないと思われるIEEE1394がすでに上に控えている状態で、わずかに12Mbpsでデバイス間での通信さえ自由に行えないようなインターフェースの存在意義がよくわからなかったのです。

USBターゲットのファームウェアなどに手を出さずことになってまじめに調べ始めたものの、雑誌などでの説明はたいがいコンセプト的なところか、あるいは物理層やデータ・リンク・レベルの話が大部分で、具体的に何かを動かしたという例はほとんどない状態でした。

それなら自力で何とかするしかない、仕様書をダウンロードしてはみたものの、あまりに膨大な量に圧倒されてどこから手をつければよいのかさっぱりわからず、デバイスのユーザズ・マニュアルやレジスタ・セット・マニュアルと突き合わせながらつたない

英語力を頼りに関係のありそうなところを片っ端から訳していたという始末でした(おかげで変な日本語を書き連ねたノートが数冊できてしまった)。

こうして実際にターゲットが動くようになってみると、複雑だと思ったことの大部分はUSBコントローラやホスト側で処理されてしまうため、実際にバルク転送やインタラプト転送を行いたいというレベルなら、その取り扱いは思っていたよりずっと単純であることがわかってきました。しかも、ちょっとつないで実験して、切り離して加工して、またつなぐということがごくあたりまえのこととしてできるのです。

たとえば、ROMライターにしても、コンセントを探したり、マシンの後に回ってケーブルを着脱したりせず、ハブにちょっとケーブルを差し込んでダウンロード、ソフトウェアを立ち上げて、使い終わったらまた抜いて……こんなことがあたりまえのようにできるのがUSBならではのところでしょう。

これまで個人レベルではちょっと敷居が高く、近寄りたがいイメージだったUSBが、本書のような解説書でぐっと身近なものとなるはずです。あなたも自分のアイデアを生かしてオリジナルのUSB機器を作ってみましょう。

2. 本書の読み方

● USBのシステム構成

図1にUSBのシステム構成を示します。USBの周辺機器というのは、上司(ホスト)からの指示がなければ

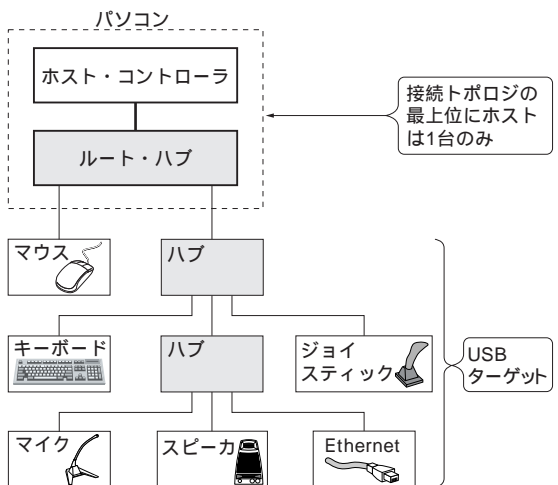


図1 USBシステムの構成

ば動かさず、言われたことしかやらない怠け者の部下のようなものです。上司はそれぞれの部下がどのような仕事をしているかを考えて、各ターゲットのエンドポイントをアクセスするスケジュールを立てて処理しなくてはならないというわけです。

ホスト側はそれぞれのターゲット機器へのコマンド送出や状態チェック、データの読み取りなどのスケジューリング、着脱のチェックなどをすべて行わなくてはならないので非常にたいへんなのですが、ホストのCPUはとても高性能ですし、コントローラ・チップのゲート数もマザーボード上のコントローラの総ゲート数からすればたいしたことはなく、コスト上の問題はありません。

逆にターゲットの側はハードウェアもソフトウェアもかなり単純な構造となり、安価な機器にも使用できるということになります。おおざっぱな計算ですが、ホスト・コントローラは1万ゲート程度、ターゲット側は1500ゲート程度で実現できるでしょう。

このように、ホストとターゲットをあえて非対称とすることで、ターゲット側を極力単純で低価格なものにしながら、さまざまな周辺機器を一つのバスに接続するという目標を達成したのがUSBと言えます。実際にキーボードやマウスといった、低価格な周辺機器にもUSBインターフェースを搭載することができたのは、この設計思想のおかげと言えるでしょう。

● USBと一口にいても…

さて、一口で「USB機器を設計する」といっても、それは具体的にはどのようなUSB機器なのでしょう。図1からもわかるように、USBシステムはホストとターゲットが明確に分かれています。開発するのはUSBホスト機器なのでしょう。それともUSBターゲット機器なのでしょう。

● WindowsやLinuxなどのOSを搭載しているパソコンに接続して使用するUSB周辺機器を開発する

この場合はパソコンがホストとなるので、USBターゲット機器の開発となります。そしてこの場合でも、より具体的にどの部分の開発を担当するかで、必要な知識は異なってきます。大きく分けると次の三つになるでしょうか。

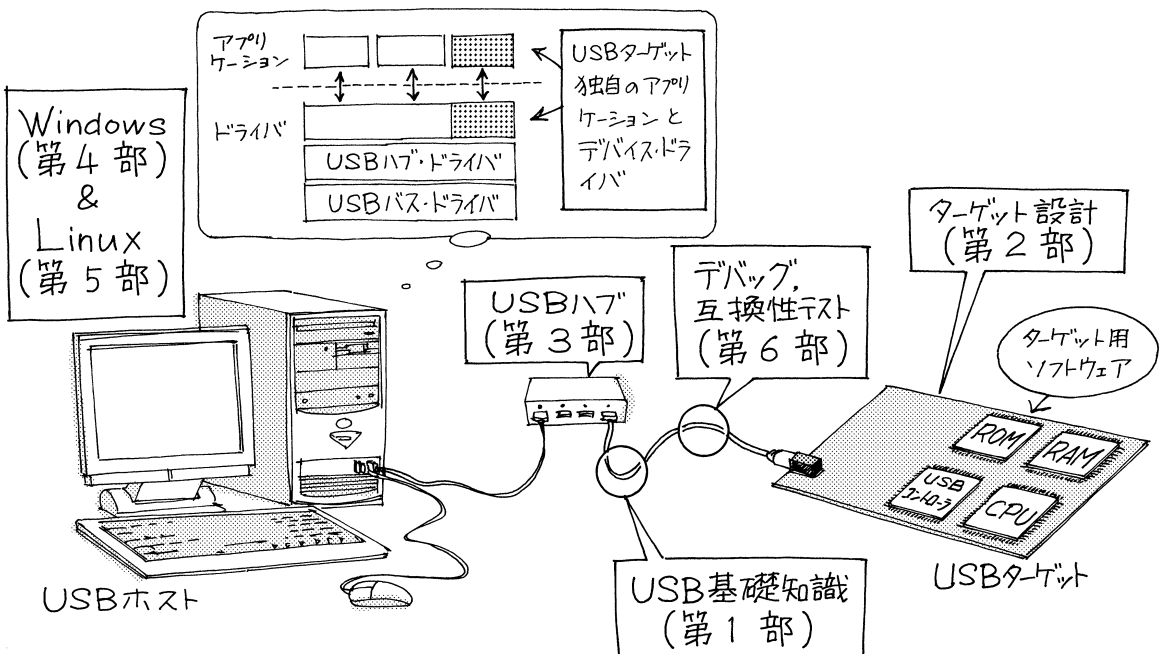
▶ ターゲット機器ハードウェア開発

ハードウェアは目に見えるので、もっともわかりやすい部分でしょうか。どんなターゲット・コントローラを使って、どのような回路でターゲット機器を構成するのかを、具体的な事例を示しながら解説します。

これについて詳しく知りたい場合は、第2部の第3章やAppendix2を参照してください。また参考文献(1)を手元に置いておくことも強くお勧めします。

▶ ターゲット機器ソフトウェア開発

USBターゲット・コントローラの中には、プログラム不要のもの、つまりUSB変換チップのようなもの



のも存在します。そのようなターゲット・コントローラを使う場合は、ターゲット機器用のソフトウェアを開発する必要はありません。しかし一般的にこのようなターゲット・コントローラは機能や自由度が少ないため、たいていの場合は、ターゲット・コントローラと組み込みCPUを組み合わせることになります。CPUが存在するということは、そのCPUを動かすためのソフトウェアが必要だということです。

これについて詳しく知りたい場合は、第2部の第2章やAppendix1を参照してください。

▶ ホスト側のドライバおよびアプリケーション開発

WindowsやLinuxはUSBにも対応しているので、ホスト側のUSBシステムを管理する基本的なUSBバス・ドライバはすでに完成しています。USBバス・ドライバとは、パソコンにUSB機器が接続された直後に動作する初期化ルーチンや、基本的なデータ転送を管理する部分で、そのパソコンに接続されているUSB周辺機器すべてに共通する基本ドライバです。

WindowsやLinuxのような大規模OSは、ドライバも階層構造になっています。開発するターゲット機器用のドライバは、USBバス・ドライバの上に乗る形で動作します。USBとして共通の処理はUSBバス・ドライバが処理してくれるので、ターゲット用デバイス・ドライバとして処理が必要な部分は、そのターゲット固有の機能を使う部分だけです。

また、そのデバイス・ドライバを呼び出して実際にターゲット機器を制御するアプリケーションも必要でしょう。

これらについて詳しく知りたい場合は、OSがWindowsの場合は第4部を、Linuxの場合は第5部を参照してください。

▶ USBターゲット機器のデバッグ技法について

試作したターゲット機器が、そのまますんなりと動作することはまずありません。正常に動作するようデバッグ作業が必要でしょう。USBはホストとターゲットの通信なので、まずはUSBバス上でパケットが正常に流れているかどうかを確認する必要があります。

これらUSB機器のデバッグや互換性テストについては、第6部を参照してください。

● 本書で解説される範囲

本書では以上のような開発場面を想定して、USBターゲットのファームウェアの一般論から、代表的なUSBターゲット・コントローラを使った具体的なハードウェア設計事例、そしてWindowsおよびLinux用

のドライバおよびサンプル・アプリケーションの作成までを解説します。

● USB周辺機器を接続できる組み込み機器を開発する

最近では、パソコンで使われている周辺機器を組み込み機器にも接続したいという要求が増えてきています。デジタル・カメラとプリンタを直接接続してパソコンを使わずに印刷したり、AV機器にUSBキーボードを接続して曲のタイトル名を入力可能にするなどの機能を実現するには、組み込み機器にUSBホスト機能が必要です。

これら組み込み機器にUSBホスト機能を実装する方法については、参考文献(2)や(3)などを参照してください。

● 基本的にはUSBターゲットだが、状況に応じてホスト機能をもつ機器を開発する

最近では、普段はUSBターゲット機器でありながら、必要に応じてUSBホストに化けるUSB機器も登場しています。たとえばPDAは、自宅や会社に戻ってパソコンに接続するときはターゲットとして動作しますが、PDAに外付けのUSBキーボードやUSB通信機器を接続して使う場合はホストとして動作します。

このように、ターゲットとホストの両方の機能をもつ機器の設計手法には、大きく分けて次の二つがあります。

- ホスト機能とターゲット機能の両方をそれぞれ実装する
- USB On-The-Go仕様に準拠する

前者はホスト用のUSBコネクタとターゲット用のUSBコネクタをそれぞれもっており、ユーザがどちらから接続するかで動作を変えるものです。後者はUSBコネクタは一つで、使用状況によってホスト機能とターゲット機能が切り替わるものです。詳細は参考文献(2)などを参照してください。

参考文献

- (1) USBターゲット機器設計のすべて、TECH I Vol.27.
- (2) USBホスト・システム実践の実装、Interface 2005年12月号.
- (3) USBホスト&ターゲット・システム設計技法、Interface 2004年10月号.