第4章 ドライバ不要＆接続すると自動的にプログラムも起動できる

USB仮想CD-ROMドライブの設計事例

桑野 雅彦

USB標準クラスに準拠したUSBターゲットの設計事例として，最後はマスストレージ・クラスを取り上げる．USBのストレージでは，実際のドライブ制御コマンドとしてSCSIコマンドを採用しているものが多い．ここでもUSBストレージとして一般的なバルク・オンリ方式でSCSIコマンド・セットを採用した仮想CD-ROMドライブを実現する．
（編集部）

● USB仮想CD-ROMドライブを作ろう！

ここではマスストレージ・クラスUSB機器の製作例として，Cypress Semiconductor社（以降，Cypress社）のEZ-USB/FX2の先に1Mビット（128Kバイト）のフラッシュROMを接続し，これをCD-ROMに見せかける，「USB接続の仮想CD-ROMドライブ」を作ってみることにしました．

マスストレージ・クラスとは，ハード・ディスク，MO，CD-ROMなどの大容量外部記憶装置の接続を想定したものです．ホストから送られてきたコマンドをファームウェアで処理し，ハード・ディスクなどがつながっているときと同じように応答すれば，フラッシュROMやSRAMなどをハード・ディスクのように見せかけることができます．市販のUSBメモリなどはこの方法でフラッシュROMをディスクに見せかけています．最近では携帯電話などもマスストレージ・クラス対応となり，パソコンのUSBポートと直結するだけで画像データのコピーなどが簡単に行えるようになっています．

ハード・ディスクに見せかけた場合，512バイト単位のセクタごとにリード/ライトできれば，基本的にファイル・システムなどは全部OS側で面倒をみてくれるので非常に楽です．しかし，それではあまり面白くありません．そこで同じマスストレージ・デバイスであるCD-ROMに見せかけてみることにしました．

CD-ROMなので，読み出しのみで書き込みはできません．またCD-ROMのフォーマットであるISO9660形式であらかじめデータを用意しておく必要があります．しかし，オートRUN機能を使えば，ドライブを接続しただけで指定したプログラムを自動実行させるなど，ハード・ディスクにはないユニークな動作も期待できそうです．

なお，今回はあくまでも実験ということで，処理しているコマンドなどは，筆者の実験環境（OSはWindows 2000）における必要最小限の実装になっています．マスストレージ・クラスで定義されているすべてのコマンドを実装しているわけではないので，ほかのOS環境などではこのままでは動作しないこともあります．

1. USBマスストレージ・クラスの概要

● マスストレージ・クラスの資料

マスストレージ・クラスに準拠したものを作るためには，まず仕様書を入手しておくことが必要でしょう．マスストレージ・クラスの資料としては，本稿執筆時点で次の五つが`http://www.usb.org/developers/devclass_docs#approved`にて公開されています．

（1）Mass Storage Overview 1.2
（2）Mass Storage Bulk Only 1.0
（3）Mass Storage Control/Bulk/Interrupt（CBI）Specification 1.1
（4）Mass Storage UFI Command Specification 1.0
（5）Mass Storage Bootability Specification 1.0

（1）は単なる概要，（4）はUFI（USB Floppy Interface），つまりUSBフロッピ・ディスクの説明なので，今回は関係ありません．（5）はUSBマスストレージ・デバイスからの起動方法について書いたもので，これも今回は関係ありません．

従って今回のファームウェア作成で関係あるのは，（2）

KEYWORD ―USB仮想CD-ROMドライブ，マスストレージ・クラス，バルク・オンリ方式，SCSIコマンド・セット，EZ-USB/FX2

と(3)ということになります.

● CBI とバルク・オンリの 2 種類

USB 機器の場合，すべての USB 機器が持つコントロール・エンドポイント(エンドポイント 0)に加えて，ユーザが自由に使えるデータ用のエンドポイントを用意するのが一般的です．機器への制御コマンドなどはコントロール・エンドポイントを使ったベンダ・リクエスト(内容は自由に設定可能)で行い，実データのやりとりはユーザ用のエンドポイントを使うという具合に使用するわけです．

マスストレージ・クラスでは，パソコン(PC)とターゲット機器の間でコマンドやデータのやりとりを行うために，次の二つの方法を提供しています．

(1) CBI(コントロール/バルク/インタラプト)
(2) バルク・オンリ

● CBI で使うエンドポイント

CBI は，次の四つのエンドポイントを使用します．

(1) コントロール・エンドポイント
(2) バルク IN エンドポイント
(3) バルク OUT エンドポイント
(4) インタラプト IN エンドポイント

クラス定義コマンドをコントロール・エンドポイント経由で送り，バルク IN/OUT エンドポイントで実際のデータ転送を，さらにインタラプト・エンドポイントでコマンド実行完了などのステータスを転送します．送られるコマンドは ADSC(Accept Device-Specific Command)と呼ばれ，コントロール転送のデータ OUT ステージで送られるという仕様です．

USB としては比較的素直な方法といえますが，実際には次に説明するバルク・オンリによる実装の方が多いようです．

● バルク・オンリで使うエンドポイント

バルク・オンリはその名のとおりバルク IN/OUT エンドポイントだけで，コマンドやステータス，実際のデータの転送まですべて処理します．コントロール・エンドポイントはすべての USB 機器が持つ必要があるので，実装する必要のあるエンドポイントは，

(1) コントロール・エンドポイント
(2) バルク IN エンドポイント
(3) バルク OUT エンドポイント

の三つとなります．CBI ではコマンドをコントロール・エンドポイントで送っていたのに対して，バルク・オンリではバルク OUT エンドポイントをコマンドとデータで共用します．

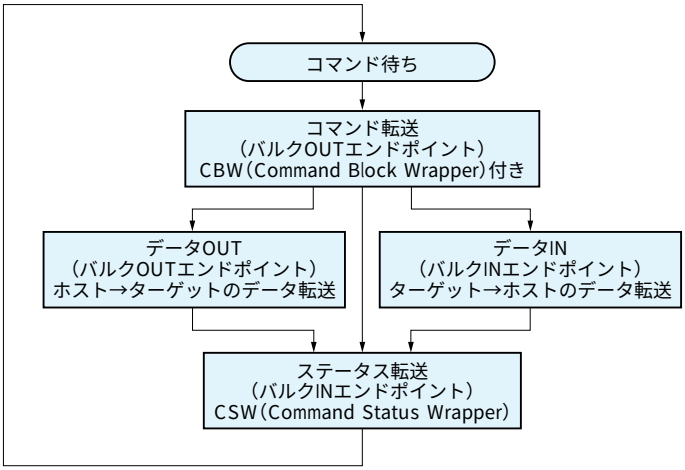

図 1　バルク・オンリにおけるコマンド/データ/ステータス伝送の流れ

実際の製品をいくつか調べたところ，バルク・オンリを使う USB ストレージ機器が多いようなので，今回もバルク・オンリを使うことにしました．

● バルク・オンリでのコマンド/データ転送の考え方

バルク・オンリの場合，コントロール・エンドポイントは，各種の USB 機器に共通する標準リクエスト(プラグ&プレイに関係する情報のやりとりや，デバイスの始動/停止要求など)以外のコマンドや情報，データのやりとりを，バルク OUT エンドポイントとバルク IN エンドポイントを使って行います．

図 1 にバルク・オンリにおける大まかな動作フローを示します．通常は一番上のコマンド待ち状態で，バルク OUT エンドポイントへのコマンド到達を待ち，以下のような動作を繰り返します．

(1) コマンド到達

コマンドはホストからバルク OUT エンドポイント経由で送られてきます．コマンドは後で説明するように，SCSI(ATAPI)コマンドに 14 バイトの CBW(Command Block Wrapper)と名付けられたヘッダの付いたものになっています．

(2) データ入出力

コマンドがリードやライトなどのデータ転送を伴うものであった場合には，引き続きバルク IN またはバルク OUT エンドポイントを使ったデータ入出力動作が行われます．

New Products ── ライフボート，USB 鍵付きハード・ディスク・データ管理ソフトを発売
ライフボートは，ハード・ディスク装置の中にセキュアな領域を作成し，その領域のデータを管理するソフトウェアと，USB 鍵からなる「カチャッと USB 秘密のドライブ 2 USB 鍵付き」を発売した．USB 鍵とパスワードによってデータを保護する．価格は 8,190 円．