USB マスストレージデバイスの研究

0aet1132　　志水 光朝

東海大学工学部通信工学科

指導教員

清水 尚彦 助教授

目 次

第 I 部　研究の目的 **5**

第 II 部　USB **6**

1　USB とは **6**

2　バス速度 **6**
2.1　ロースピード (LS) 6
2.2　フルスピード (FS) 6
2.3　ハイスピード (HS) 6

3　物理的バストポロジ **7**

4　論理的バスロポロジ **7**

5　通信プロトコル **8**
5.1　４つの転送 (トランスファ) 10
5.1.1　コントロール転送 10
5.1.2　バルク転送 10
5.1.3　インタラプト (割り込み) 転送 10
5.1.4　アイソクロナス転送 10
5.2　パケット 11
5.2.1　同期 (SYNC) パターン 11
5.2.2　PID(パケット識別子) 11
5.3　USB1.X 仕様でのトランザクション (クラシックトランザクション) 11
5.4　USB2.0 仕様でのトランザクション 11
5.4.1　μフレーム 12
5.4.2　アウトトランザクションでの PING フロー制御 . . 13

6　デバイスフレームワーク **15**
6.1　アドレスとエンドポイント 15
6.2　デバイス構成インターフェース・エンドポイント 15
6.3　ディスクリプタ 15
6.3.1　デバイスディスクリプタ 15
6.3.2　コンフィグレーションディスクリプタ 16
6.3.3　インターフェースディスクリプタ 16

6.3.4 エンドポイントディスクリプタ 16
6.3.5 ストリングディスクリプタ 16
6.4 デバイスリクエスト 16

7 プラグ&プレイの概要 16

第III部 USB Msss Storage Class 17

8 USB Mass Storage Class 17

9 サブクラスコードについて 17

10 バルクオンリー転送 18

11 CBWとCSW 18
11.1 CBWの詳細 20
11.1.1 CBWSignature 20
11.1.2 CBWTag 20
11.1.3 CBWDataTransferLength 21
11.1.4 CBWFlags 21
11.1.5 CBWLUN 21
11.1.6 CBWCBLength 21
11.1.7 CBWCB 21
11.2 CSWの詳細 21
11.2.1 CSWSignature 21
11.2.2 CSWTag 21
11.2.3 CSWDataResidue 22
11.2.4 CSWStatus 22
11.3 CBW、CSWが有功、正常である条件 22
11.3.1 CBWであると有功とされる条件 22
11.3.2 CBWが正常である条件 22
11.3.3 CSWであると有功とされる条件 23
11.3.4 CSWが正常である条件 23
11.4 ステータストランスポートフローチャート 23
11.4.1 フェーズエラー 23
11.4.2 リセットリカバリー (Reset Recovery) 23

12 Host/Device Data Transfers 24
12.1 概要 24
12.2 デバイスエラーの処理 24

12.3 ホストエラーの処理 25
12.4 CBW が有功でなかった場合 25
12.5 デバイス内部エラー 25
12.6 コマンドフェイル 25
12.7 １３ Cases 26
12.7.1 ホストがデータ転送を望んでいない場合 27
12.7.2 ホストがデバイスに対してデータを要求する場合 28
12.7.3 ホストがデバイスに対してデータを送る場合 29

第 IV 部　製作にむけて **31**

13 実機ログの検証 **31**
13.1 サブクラスコード 31
13.2 CBW 31
13.3 USB の仕様 31

14 実機のログにて使用されていたコマンド の解説 **32**
14.1 Inquiry(オペコード =12h) 32
14.2 Read Capacity(オペコード =25h) 32
14.3 Read(10)(オペコード =28h) 32
14.4 Mode Sense(6)(オペコード =1A) 32
14.5 Test Uniti Ready(オペコード =00h) 32
14.6 Vendor-Specific(オペコード =23h) 33

15 まとめ **33**

第 V 部　感想 **35**

第 I 部
研究の目的

USB マスストレージデバイスを構築するためにはどうすればいいかを調べます。主には外付けハードディスクをターゲットに考えました。

第 II 部
USB

1　USB とは

USB(Universal Serial Bus) は 3 種類のバス速度をもち、低価格なシステムで多くの分野で普及しています。ホスト主導で、プラグ＆プレイに対応しており、多種多様なマルチメディアに対応している等これからもますます拡大していくことが予想されます。現在 USB では 1.0、1.1、2.0 の 3 種類の仕様がリリースされています。

2　バス速度

USB2.0 では 3 種類のバス速度が定義されています。

2.1　ロースピード (LS)

1.5Mbps のバス速度です。マウスなどの非常にコストを重視する機器に利用されています。マウス、キーボード、ゲームパッドなどの入力装置に採用されています。

2.2　フルスピード (FS)

12Mbps のバス速度です。USB1.1 までの仕様策定時において、低価格で、使い勝手が良く、多くの PC 周辺機器で必要十分な速度で、さらに新しいタイプの周辺機器開発を促進するための新しいバスとして定義されました。USB2.0 仕様で追加されたハイスピードが定義されても、１２Mbps のバス速度で十分な PC 周辺機器が多く存在するので、今後も多くの USB デバイスで採用されるでしょう。

2.3　ハイスピード (HS)

480Mbps のバス速度で、USB2.0 仕様で追加された仕様です。ハードディスクドライブ、DVD ドライブ等の高速なバスを必要とするマスストレージ機器、ハイエンドのカラーイメージプリンタなど、現時点からしばらくの期間は、ほとんどの周辺機器で必要十分なバス速度を誇ります。

3　物理的バストポロジ

これは、エンドユーザーも意識する必要のあるバストポロジになります。ホストは唯一のバスマスタとして、バス分岐点にハブを配置して末端に USB デバイスがぶらさがらという階段状のスター型のバスを構成します。(図 1) 一つの USB システムの中では、１台のホストと最大１２７台までの USB 機器 (ハブ/デバイス) を接続できます。

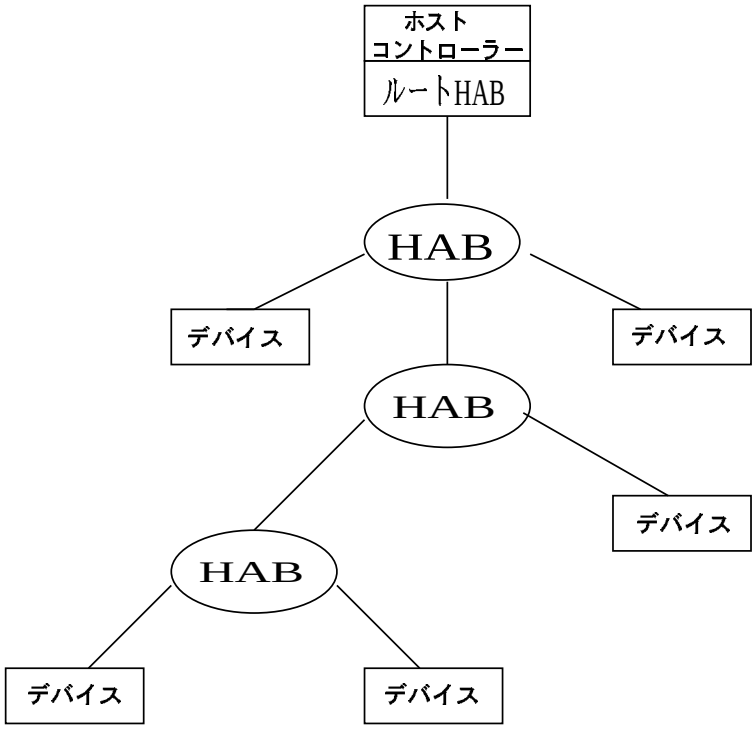

図 1: 物理的バストポロジ

4　論理的バスロポロジ

これは、デバイス開発社が機能を実現するにあたって、ホスト側アプリケーションソフトウェアも含めたデバイス側システム設計時に、意識する必要があるバストポロジになります。物理的なバストポロジは階段状のスター型をとりますが、論理的なバストポロジにおいて、ハブの存在が削除され、一つのホストと複数のデバイスを接続している 1 対多の通信のように、抽象化されます。(図 2) この抽象化は、ホストとハブの処理によっ

て、実現されています。USB プロトコル仕様を理解するうえでは、この観点にバストポロジは理解する必要があります。

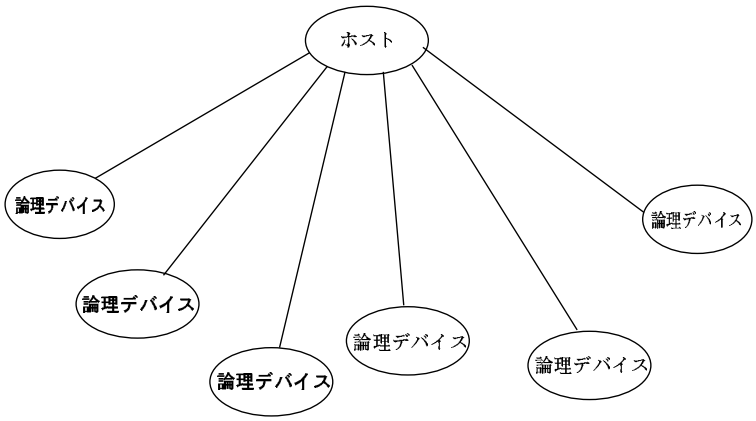

図 2: 論理的バストポロジ

5 通信プロトコル

USB システムにおいて、通信はホストの USB プロトコルスタック・スタックが主導権を握っています。すなわち、たとえデバイス側からデータを送信する場合であっても、ホストがデバイスに対してバスの使用権を与えてからデバイスはデータを送信します。USB デバイス上では、パケットという単位でデータを送受信します。いくつかのパケットによって、トランザクションという通信単位を構成します。またトランザクションがいくつか集まって、一つの転送 (トランスファ) を構成します。図 3 にその様子に示します。

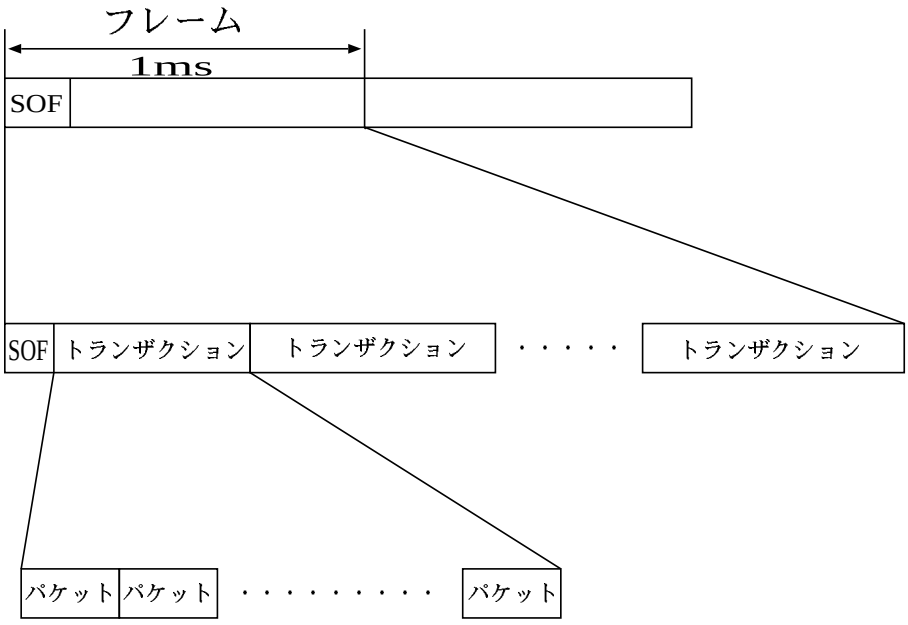

図 3: ３つの通信単位

5.1 ４つの転送 (トランスファ)

USB システムでは特徴をもった４種類の転送があります。

5.1.1 コントロール転送

勃発的で非周期的な通信のうち、リクエスト-レスポンス形態の双方向通信です。標準デバイスリクエストと呼ぶコントロール転送を使用したやり取りが定義されています。これは、主にホストがデバイスをプラグ＆プレイするために使用され、全ての USB デバイスは、このリクエストに対応する必要があります。主にデバイスの制御を目的にした転送です。

5.1.2 バルク転送

勃発的で非周期的な通信のうち、遅延が問題にならない大量のデータを転送する用途で使います。たとえばスキャナのイメージデータやストレージ機器のデータで使います。他の転送タイプよりスケジューリングの優先度が低くなりますが、他の転送の空き時間を全て使用することができます。

5.1.3 インタラプト (割り込み) 転送

非周期的で低頻度のイベントを、ホスト-デバイス間で通知するために使用します。たとえば、マウスやキーボードの入力データがあります。この転送タイプの名前から連想されるイメージとは異なり、インタラプト (割り込み) 転送は周期的なホストのポーリングによって処理され、そのポーリング間隔はデバイス側からホストに申告します。

5.1.4 アイソクロナス転送

連続的で周期的な通信に使用されます。通信経路を確立した後は、限定的なレイテンシ (遅延) で一定の転送レートが保証されます。動画や音声データのようなリアルタイム性を必要とするストリーミングデータで使用されます。

5.2 パケット

ホストやデバイスがバスの時間を確保する単位です。同期 (SYNC) パターンで始まり、EOP までの時間的に連続したビット列で構成される片方向の通信単位です。

5.2.1 同期 (SYNC) パターン

先頭の同期 (SYNC) パターンは、80h の 1 バイトのデータパターンです。この同期パターンの間に、受信側のロジック回路が内部のビットクロックを調整します。

5.2.2 PID(パケット識別子)

同期パターンのすぐ後に、PIDが続き、パケットの種類を示しますPIDのデータ値は、PID 番号 4 ビットと、それを反転した 4 ビットを続けることにより PID 値の保護をしています。PID の一覧を表 1 に示します。

5.3 USB1.X 仕様でのトランザクション (クラシックトランザクション)

トランザクションは、ホストがスケジューリングする単位で、また、転送を構成する単位にもなります。なお、USB の通信方向の IN/OUT というフレーズは、ホストを中心にしたものです。トランザクションのフォーマットを図 4 に示します。

5.4 USB2.0 仕様でのトランザクション

USB2.0 のハイスピード (HS) で動作しているバス上を流れるトランザクションフォーマットは、USB1.X 仕様に対して大きな仕様追加が行なわれています。ただし、従来までの USB1.X 仕様のデバイスにおいては、これらの仕様追加を意識する必要はありません。

ハードディスクドライブは大量のデータを扱うため、ハイスピードのバス速度を実現することを目標したほうが良いでしょう。

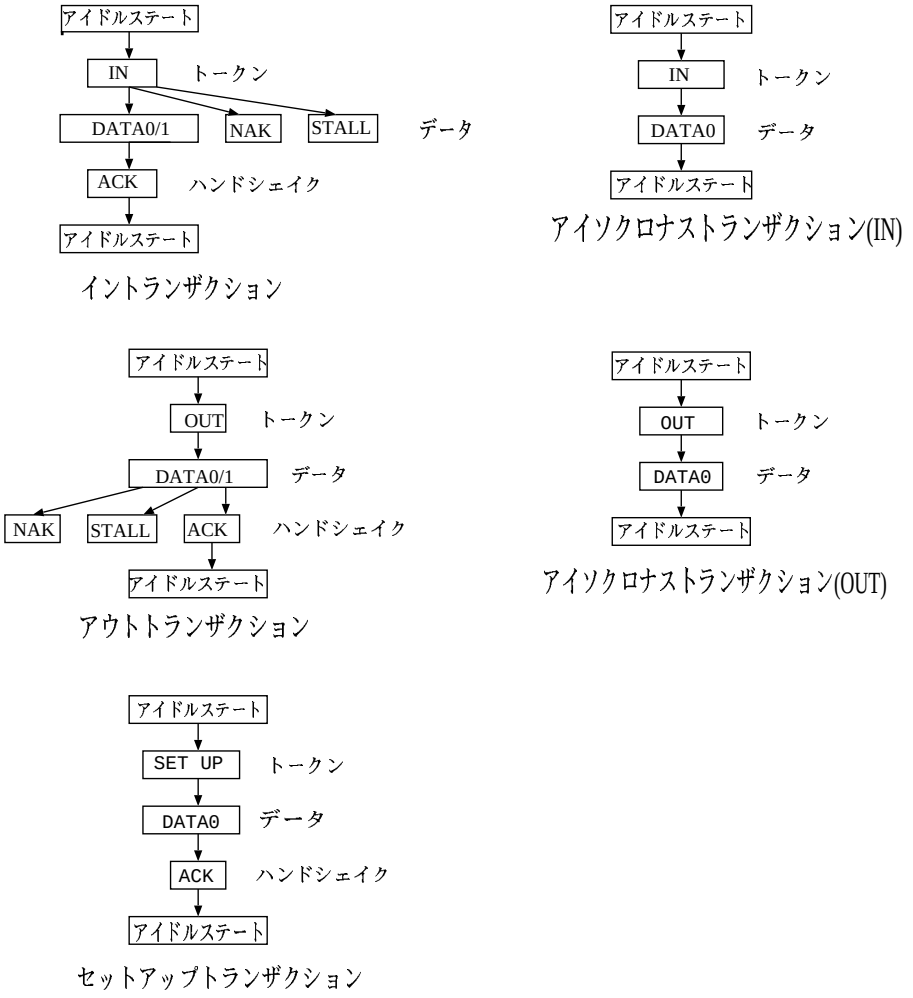

図 4: USB1.X 仕様のトランザクション

5.4.1 μフレーム

フルスピードバスにおけるフレームは１ msごとですが、ハイスピードのバンド幅を有効に活用するため、ハイスピード上のフレームの周期は、従来の１/８の長さである 125 μ sになりました。(μフレームといいます) これにより、アイソクロナス転送を行なうハイスピード (HS) デバイスにとって、１ msのフレームより、要求されるバッファサイズを減らすことができます。μフレームにおいて、SOF パケットの PID は、フルスピードバスの SOF を同じモノを使用します。ハイスピード上のμフレームでは、８回連続した同じフレーム番号をもつ SOF パケットが発行されます。

5.4.2 アウトトランザクションでの PING フロー制御

これは、ハイスピード (HS) 転送時のアウトトランザクションにおいて、追加されたフロー制御のプロトコルです。大きな OUT データパケットを転送した後に、デバイスからの NAK 応答で大きな OUT データパケットをの再送を繰り返すことを未然に防止します。デバイス側の処理スピードが間に合わない場合、OUT データパケットの再送を繰り返すと、バス上のトラフィックが、アウトトランザクションの再送のため、消費されてしまうことになります。この問題は、ハイスピード (HS) 転送時に顕著に現われることとなり、HS バス効率の低下を抑える目的で、PING フロー制御が導入されました。ホストから HS デバイスに、OUT データパケットを送信する前に、ホストから PING パケットを送り、このときのデバイス反応が ACK であるか NAK であるかを確認します。もし、NAK であった場合には、図 5 に示す HS デバイス遷移図のように、受信できるスペースの準備ができてないということなので、次に DATA0/DATA1 データパケットをデバイスに送信したとしても、NAK を返す可能性が高くなります。このような場合、図 6 の USB2.0 ホストの状態遷移図にしたがい、OUT および DATA0/DATA1 パケットを転送せず、再度 PING パケットで、デバイスの状態を再確認を繰り返します。最終的に、PING パケットに対して ACK 応答が得られた後に、OUT および DATA0/DATA1 パケットを送信します。このような手順をとることで、データ転送後に NAK が発生することを防止することができます。

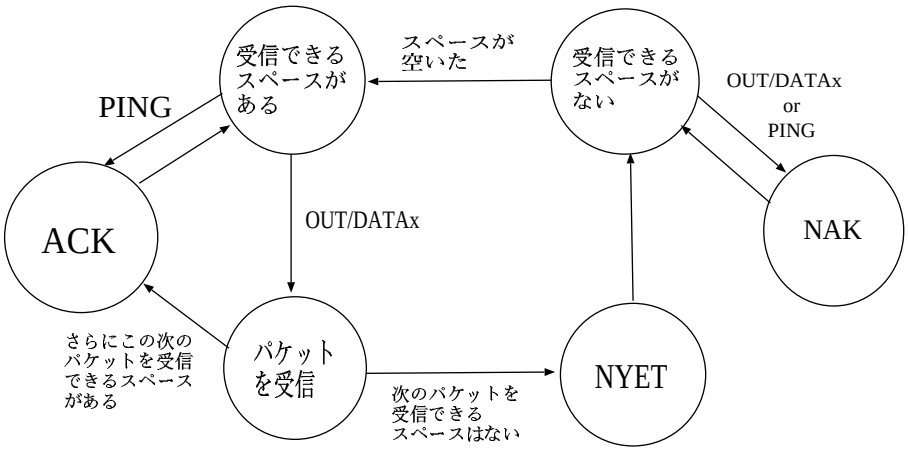

図 5: HS デバイス遷移図

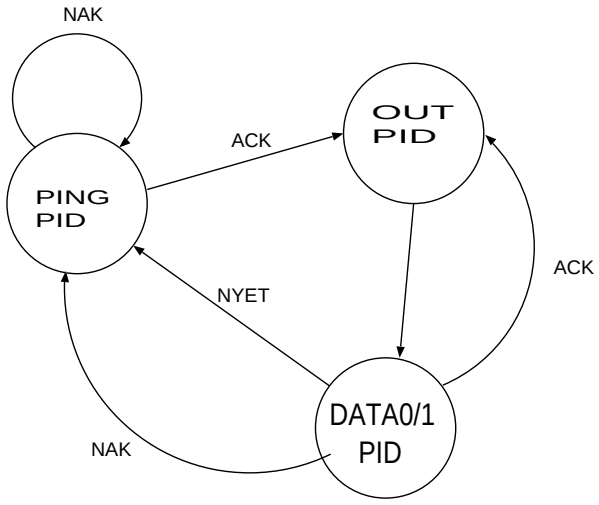

図 6: USB2.0 ホストの状態遷移図

6　デバイスフレームワーク

USB 規格では、デバイスがもつべき共通の属性と動作を定義しています。ホストは、これらの USB デバイスの機能を使い、プラグ＆プレイ、上位アプリケーションのためのデータ転送、パワーマネジメント等の機能を実現します。

6.1　アドレスとエンドポイント

USB の通信は、ホストとエンドポイントの間で行なわれます。ホストは、一つのシステムの中でユニークなアドレスをデバイスに割り当てて、デバイスのエンドポイントと通信します。ホストの発行するトークンパケットの ADDR、ENDP フィールドは、アドレスとエンドポイントを示します。ADDR フィールドは 7 ビット分あります。アドレスの 0 はデフォルトアドレスを呼ばれ、初期状態のデバイスがホストからのリクエストを受けつけるためのものです。デバイスは、０～１５の番号を持つエンドポイントを持つことができますが、必ずコントロール転送用のエンドポイントを持つ必要があり、標準デバイスリクエストは、エンドポイント 0 で行なわれます。

6.2　デバイス構成インターフェース・エンドポイント

デバイスは複数のエンドポイントを持つことができますが、それらをインターフェースという論理的な単位で機能別にグループ化します。もちろん一つのデバイスが複数のインターフェース (機能) を持つことができますし、それらの集合がデバイスの構成になります。

6.3　ディスクリプタ

ディスクリプタとは、デバイスの特性や属性などの情報を表現するためのものです。そのフォーマットは定義されています。USB1.X 仕様では標準なものとして 5 種類のディスクリプタがあります。USB2.0 仕様では USB1.X のスーパーセットになっていて、多少フィールドの意味も HS デバイス用に拡張されている箇所もあります。

6.3.1　デバイスディスクリプタ

ベンダ ID、プロダクト ID、USB 仕様のリリース番号といった基本的な情報が表現されます。

6.3.2　コンフィグレーションディスクリプタ

インターフェース数、最大バス消費電力などの情報が表現されます。

6.3.3　インターフェースディスクリプタ

エンドポイント数、インターフェースのクラスコードが表現されます。

6.3.4　エンドポイントディスクリプタ

エンドポイントの詳細情報が表現されます。

6.3.5　ストリングディスクリプタ

ストリングの UNICODE などが表現されます。

6.4　デバイスリクエスト

コントロール転送のセットアップステージは、そのデータフェーズのフォーマットが定義されていて、デバイスリクエストといいます。タイプが標準のときは、標準デバイスリクエストと呼ばれ、全ての USB デバイスはこれをサポートしなければいけません。

7　プラグ＆プレイの概要

USB のホストは、デバイスがハブのポートに接続されたことを検出することができます。分かりやすく言うなら、PC に USB 機器を接続すると、PC は勝手に検出して必要なドライバをロードして使える状態にするといものです。

その流れは、デバイスがハブのポートに接続されるとハブがそれをホストに知らせ、ホストは新しいでばいすが接続されたポートを開くように指示。ホストはアドレス 0 に対して標準デバイスリクエストを要求して応答させます。デバイスからのデバイスディスクリプタを取得すると、ホストはデバイスにユニークなアドレスを割り当てるとともに、デバイスディスクリプタの情報から関連するドライバをロードします。これが大まかな検出までの流れです。

第 III 部
USB Msss Storage Class

8　USB Mass Storage Class

「USB Mass Storage Class」は大規模記憶装置 (ストレージ) をホストに接続し、データの書き込み、読み出しなどを行なう機器に合うように規格化されたクラスです。ホストに、このクラスのデバイスであることを伝えるためには、インターフェースディスクリプターの bInterfaceClass フィールドの値を 0X08 にする必要があります。データ転送プロトコルとして
バルクオンリー転送 (Bulk-Only Transport)
ＣＢＩ転送 (Control/Bulk/Interrupt(CBI) Transport)
の２種類があります。またＣＢＩ転送はインタラプト転送を含むものと含まないものの２種類に分けられます。

しかしＣＢＩ転送はフルスピードのフロッピーディスクドライブにのみ使用が推奨されています。ハイスピードドライブには使用すべきでありません。またこれから将来にデザインされるどんな機器にも使用を推奨されていません。
よって今回使用するデータ転送プロトコルはバルクオンリー転送とします。インターフェースディスクリプターの bInterfaceProtocol フィールドの値は 0X80 とします。

9　サブクラスコードについて

ホストがデータのロードやセーブをするためにデバイスを使用する場合、ホストからデバイスに対して命令 (コマンド) を与えます。デバイスは送られてきたコマンドを実行することによりデータのロードやセーブが行なえます。ホストからデバイスに対して送られる命令 (コマンド) はサブクラスコードとして定義されています。コマンドフォーマットの種類は全部で７種類あります。それぞれのホストにつながれた機器が正常に動作するためには、正しいサブクラスコードを指示をする必要があります。ホストに機器が対応しているコマンドフォーマットを伝えるためにはインターフェースディスクリプターの bInterfaceSubClass フィールドの値で指示します。表２にサブクラスコードの一覧を示します。

10　バルクオンリー転送

バルクオンリー転送は名前のとおり、バルク転送のみを使用してホストとデバイス間でデータの転送を行なわれます。バルク転送は、データを送信する方向で２つに分けることができます。ホストからデバイスにデータを送信する転送をバルクアウト転送、ホストにデバイスからデータを送信する転送をバルクイン転送を言います。バルクオンリー転送では、バルクアウト転送、バルクイン転送をあらかじめ定めた組み合わせにすることによりホスト-デバイス間のデータを転送を行ないます。つまりバルクオンリー転送は図７のようになります。それぞれのバルク転送には異なった意味 (目的) があり、トランスポートとして管理します。図 7 のとおり、バルクオンリー転送の流れは
コマンドトランスポート (CBW) →データトランスポート→ステータストランスポート (CSW)
のようになります。アプリケーションレベルで見ると、
IN パイプ、OUT パイプ、コントロールパイプ
の３つのみになります。図 8 にその様子を示します。コントロールパイプはコマンドトランスポート、ステータストランスポートのことを示しています。IN パイプ、OUT パイプはデータトランスポートのことを示しています。

11　CBW と CSW

Command Block Wrapper(CBW) とはホストがデバイスに対して、「コマンド」を伝えるために使用するものです。３１バイトで構成されており、図 9 のような内訳になっています。それぞれ構成要素は LSB ＝リトルエンディアン (バイトが反転) で送られてきます。

それに対してデバイスは１３バイトから成る Command Status Wrapper(CSW) により、コマンドを正常に達成できたか、エラーしたか等の「ステータス」をホストに伝えます。その構成を図 10 に示します。CSW もリトルエンディアンです。

この２つの重要な要素によりバルクオンリー転送は成立するといっても過言ではありません。裏を返せば USB マスストレージデバイス構築を考える際に重要なポイントとなります。

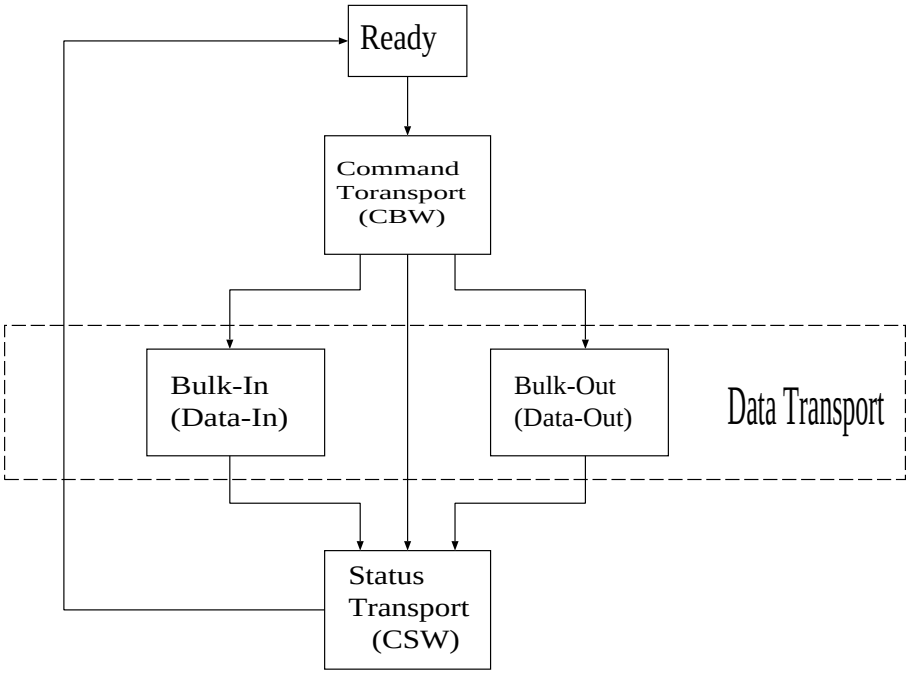

図 7: バルクオンリー転送

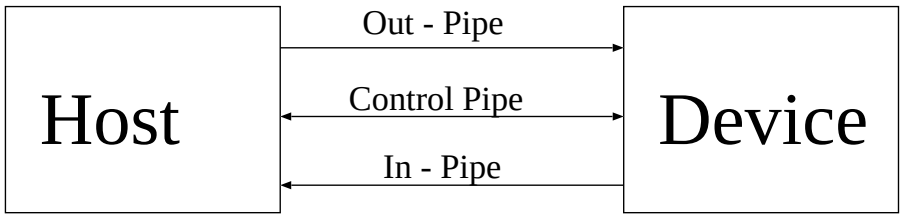

図 8: アプリケーションレベルで見るバルクオンリー転送

byte \ bit	7	6	5	4	3	2	1	0
0~3	dCBWSignature							
4~7	dCBWTag							
8~11 (08h~0Bh)	dCBWDataTransferLength							
12(0Ch)	bmCBWFlags							
13(0Dh)	Reserved(0)					bCBWLUN		
14(0Eh)	Reserved(0)				bCBWCBLength			
15~30 (0Fh~1Eh)	CBWCB							

図 9: Command Block Wrapper

byte \ bit	7	6	5	4	3	2	1	0
0~3	dCSWSignature							
4~7	dCSWTag							
8~11 (8~0Bh)	dCSWDataResidue							
12 (0Ch)	bCSWStatus							

図 10: Command Statsu Wrapper

11.1 CBW の詳細

11.1.1 CBWSignature

データパケットが CBW であることを示すためのものであり、その値は 43425355h です。この値でデバイスは CBW であることを認識します。

11.1.2 CBWTag

ホストが CBWTag で送った値を、デバイスは CSWTag でその値をそのまま返さなければなりません。この値でホストは CBW と対応する CSW

であったかを識別します。

11.1.3 CBWDataTransferLength

データトランスポートの予定データ長。ここが０の場合、データトランスポートは存在しません。

11.1.4 CBWFlags

このフィールドのビットは、ビット７が０の場合、データトランスポートはバルクアウト転送で行なわれ、１の場合、バルクイン転送が行なわれます。ビット６～０は０で固定です。

11.1.5 CBWLUN

コマンドブロックが送られている装置の論理ユニット番号。

11.1.6 CBWCBLength

CBWCB フィールドの有功バイト数を示すものです。

11.1.7 CBWCB

デバイスによって実行されるコマンドブロックを格納するフィールド。ここにホストが実行したいコマンドが入ります。なお、デバイスのサブクラスコードにも依存します。

11.2 CSW の詳細

11.2.1 CSWSignature

データパケットが CSW であることを示すためのものであり、その値は 53425355h です。この価でホストは CSW であることを認識します。

11.2.2 CSWTag

デバイスはホストから送られて来た CBWTag と同じ値を CSWTag にセットして返します。

11.2.3 CSWDataResidue

ホストが CBWDataTransferLength で要求したデータ量とデバイスが実際に送るか、受け取ったデータ量の違いを示すものです。基本的にエラーが起きない限りここの値はゼロです。

11.2.4 CSWStatus

コマンドの成功の成功、あるいは失敗を示します。コマンドが正常に完了した場合、デバイスはこのフィールドを 0h00 にセットします。ゼロ以外の値は次の値とし、コマンド実行時の不具合を示します。
コマンドフェイルは 0h01、フェーズエラーは 0h02 です。

11.3 CBW、CSW が有功、正常である条件

これから述べる点はマスストレージデバイス構築の重要なポイントになってきます。これらに準拠してないとバルクオンリー転送の核である CBW、CSW が正しく動作しません。

11.3.1 CBW であると有功とされる条件

デバイスは次の条件を満たしている場合、CBW であると認識します。
１、デバイスの CSW を受け取った後、あるいはリセットの後に送信したものであること。
２、CBW のデータ長が３１バイトであること。
３、CBWSignature フィールドが 43425355h であること。

11.3.2 CBW が正常である条件

デバイスは次の条件が満たされている場合、CBW は有功であるとみなします。
１、全てのデータに値が乗っていること。つまり言いかえれば、使用しないデータ部分でも何らかの数値が乗っていけなければなりません。
２、CBWLUN フィールドで指示した論理ユニット番号がデバイスにサポートされていること。
３、CBWCBLength で指示したデータ長及び CBWCB の内容がサブクラスコマンドと一致していること。

11.3.3　CSW であると有功とされる条件

ホストは次の条件を満たしている場合、CSW であると認識します。
１、CSW のデータ長が１３バイトであること。
２、CSWSignature フィールドが 53425355h であること。
３、CSWTag フィールドと応じている CBW の CBWTag フィールドが一致していること。

11.3.4　CSW が正常である条件

ホストは次の条件のどちらかが満たされている場合、CSW は有功であるとみなします。
１、CSWStatus フィールドの値が 00h or 01h であるときで、CSWDataResidue フィールドの値が CBWDataTransferLength フィールドの値より少ないか等しい場合。
２、CSWStatus フィールドの値が 02h である場合。

11.4　ステータストランスポートフローチャート

図 11 にステータストランスポートの流れを示します。

11.4.1　フェーズエラー

ホストは CSW でフェーズエラーが起こった報告された場合、リセットリカバリー (Reset Recovery) を実行します。

11.4.2　リセットリカバリー (Reset Recovery)

ホストはリセットリカバリーを次の手順で行ないます。
１、クラスコードのリセット (Reset) を行なう。
２、クリアーな状態になるまでバルクインエンドポイントを停止する。
３、クリアーな状態になるまでバルクアウトエンドポイントを停止する。

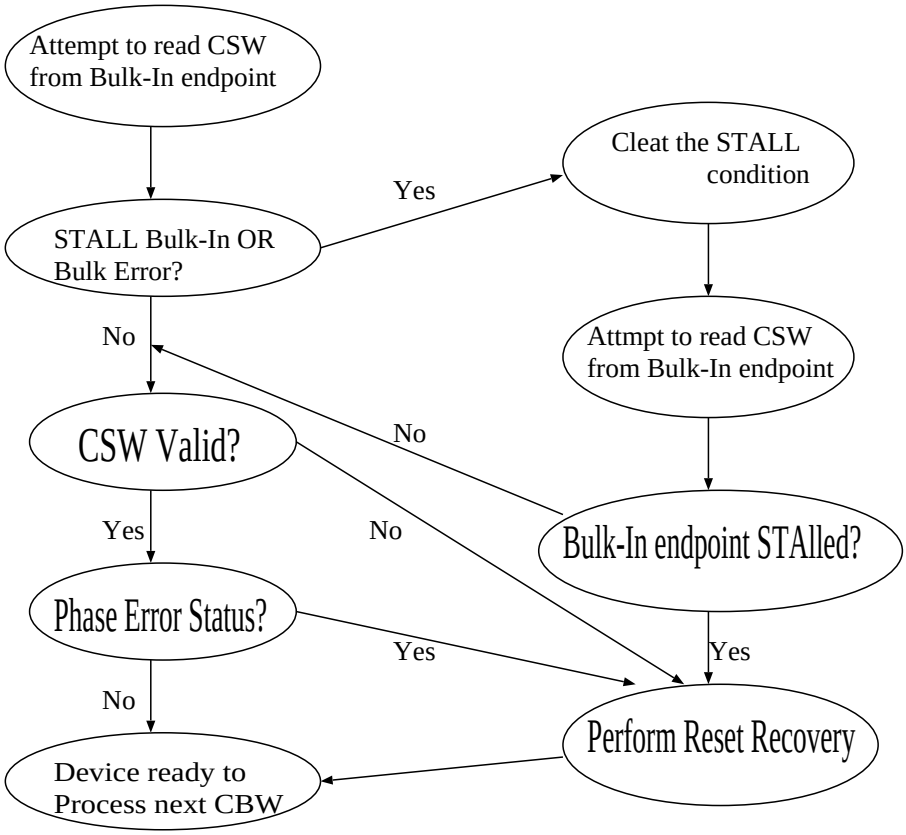

図 11: Status Transport Flow

12 Host/Device Data Transfers

12.1 概要

前述のように、バルクオンリー転送はホストがデバイスに、形成しようとするデータ転送を CBW を用いて伝えることから始まります。デバイスは CBW を受けたらチェック、解析をしてホストのリクエストをみたすように試みます。そして、その処理の結果を CSW にてホストに伝えます。ここではどのようにデータトランスポートを解析しているか、またエラーをどのように扱うかを説明します。

12.2 デバイスエラーの処理

デバイスがホストの要求を完全にみたせないときがあります。デバイスはホストの要求を完全に満たすことができないと分かったときは、バス

上でバルクイン転送またはバルクアウト転送が途中かもしれないし、ホストも他の要求を送ろうとしているところかもしれない。結果的にデバイスは適切なパイプを STALLing することにより、ホストにそのような転送を強制的に終らさせます。また有功でない CBW を受け取った場合のデバイスの対応は特に定められていません。覚えておきたいのは、バス上に STALL ハンドシェイクが現われるかどうかは、デバイスがパイプを STALL しようとしているときが転送の途中かどうかに依存しているということです。

12.3 ホストエラーの処理

ホストは有功ではない CSW 受け取った場合、リセットリカバリーを行ないます。また、正常でない CSW を受け取った場合でも、リセットリカバリーを行なうことが推奨されています。

12.4 CBW が有功でなかった場合

CBW が有功でなかった場合、デバイスはバルクインパイプを STALL します。またデバイスはバルクアウトパイプを STALL するか、受け取ったバルクアウトデータを破棄するかをする必要があります。デバイスはリセットリカバリーが行なわれるまでこの状態を維持し続けます。

12.5 デバイス内部エラー

デバイスはリセット以上に有効な手段のない内部エラーを発見したとき、次のどちらかの反応をして回復をはかります。
1、すべての進行中のデータ転送を STALL して CSWStatus フィールドの値を 02h にセットしてフェーズエラーステータスをホストに伝える。
2、バルクインパイプ、バルクアウトパイプの全てのリクエストをリセットリカバリーが行なわれるまで STALL し続ける。

12.6 コマンドフェイル

CBW で有効、正常であると認識された後で、デバイスがコマンドの実行に失敗した場合、デバイスはコマンドフェイルステータスを CSWStatus フィールドの値を 01h にセットしてホストに報告します。

12.7 １３ Cases

ホストとデバイスのデータのやりとりは図 12 のように１３通りの場合が考えられます。この関係を用いてエラー等の検出を行ないます。

		Host		
		Hn	Hi	Ho
Device	Dn	Hn = Dn	Hi > Dn	Ho > Dn
	Di	Hn < Di	Hi < Di	Ho <> Di
			Hi = Di	
			Hi > Di	
	Do	Hn < Do	Hi <> Do	Ho < Do
				Ho = Do
				Ho > Do

Hn	ホストはデータ転送を望んでいない
Hi	ホストはバルクイン転送を望んでいる
Ho	ホストはバルクアウト転送を望んでいる

Dn	デバイスはデータ転送をしようとしていない
Di	デバイスはバルクイン転送をしようとしている
Do	デバイスはバルクアウト転送をしようとしている

図 12: 13 Cases

12.7.1 ホストがデータ転送を望んでいない場合

ホストがデータ転送を望んでいない場合、最も基本に置くのはCBWFlags内のDirection bitがこれらの結果に全く影響しない(してはいけない)ということです。

ホストに要求されること
１、ホストは有効で正常であるCBWを送信すること。
２、ホストはCSWを受け取ろうと試みること。
３、STALLコンディションのときにCSWを受信した場合、バルクインパイプをクリアーし、もう一度新たにCSWを受け取ろうと試みること。
４、CSWが有効で正常であったとき
・Ｈｎ＝ Ｄｎのとき
CSWStatusフィールドを00h or 01h、CSWDataResidue = 0にセットします。この場合、データ転送は正しく行なわれたということになります。
・Ｈｎ＜ Ｄｉ, Ｈｎ＜ Ｄｏのとき
もしCSWStatus = 02hだったらホストはCSWDataResidueの値を無視します。そしてリセットを実行します。

デバイスに要求されること
１、CBWをしっかり受け取ること。
２、CBWが有効で正常であったとき、デバイスはコマンド実行をしようを試みます。
・Ｈｎ＝ Ｄｎのとき
送受信したデータがなかったとき、デバイスはCSWStatusフィールドを00h or 01h、CSWDataResidue = 0にセットします。
・Ｈｎ＜ Ｄｉ、Ｈｎ＜ Ｄｏのとき
送受信データが少しでもあったなら、デバイスはCSWStatus = 0にセットしてホストに対してフェーズエラーであることを伝えます。
３、デバイスは有効で正常なCSWをホストに返さなければなりません。
・デバイスはCSWStatus ≠ 00h or 01hのとき、バルクインパイプをSTALLします。

12.7.2 ホストがデバイスに対してデータを要求する場合

これらの場合、CBWDataTransferLength ≠ 0であり、Direction bit はバルクイン転送 (Data-In) のために１である必要があります。

ホストに要求されること
１、ホストは有効で正常な CBW を送信すること。
２、ホストはしっかりとデバイスからデータを受け取ろうとすること。また CSW も受け取ろうとすること。３、STALL コンディションで CSW を受信した場合、バルクインパイプをクリアーし、もう一度新たに CSW を受け取ろうと試みること。
４、CSW が有効で正常であったとき、
・Ｈｉ＞ Ｄｎ、Ｈｉ＞ Ｄｉ、Ｈｉ＝ Ｄｉのとき
CSWStatus = 00h or 01h であったなら、ホストは CBWDataTransferLength と CSWDataResidue の値から送られたきたデータ量が正しいかを判断し適切な受信データ量をを決定します。
・Ｈｉ＜ Ｄｉ、Ｈｉ＜＞Ｄｏのとき
ホストは CSWDataResidue の値を無視し、リセットリカバリーを実行します。

デバイスに要求されること
１、CBW をしっかりと受け取ること。
２、CBW が有効で正常であったとき、デバイスはコマンドを実行しようと試みて
・Ｈｉ＝ Ｄｉのとき
つまりデバイスは CBWDataTransferLength バイト長のデータを送ろうとしているときデバイスは CSWStatus = 00h or 01h、CSWDataResidue = 0 にセットします。
・Ｈｉ＞ Ｄｎ、Ｈｉ＞ Ｄｉのとき
つまりデバイスがホストから指示されたデータ量よりも少ないデータ量を送ろうとしているとき。デバイスは意図していたデータを送る。もちろん、CBWDataTransferLength のデータ量を満たすように送ろうとするのだが、もし実際の転送量がホストが指示したデータ量に満たなかったとき、デバイスはショートパケット (実際より少ない転送?) を用いて転送を終了させます。そしてバルクインパイプを STALL させます。デバイスは CSWStatus を 00h or 01h

にセットし、CSWDataResidueで実際に送った量と指示されたデータ量の差をホストに伝えます。

・Ｈｉ＜Ｄｉ、Ｈｉ<>Ｄｏのとき

つまりデバイスがホストに指示されたよりも過剰にデータを送ろうとするか、ホストからデータを受け取ろうとしてしまっている場合は、デバイスはショートパケットを用いて転送を終了させてバルクインパイプを STALL させ、フェーズエラーを伝えます。

３、デバイスは有効で正常な CSW を返します。

12.7.3 ホストがデバイスに対してデータを送る場合

これらの場合、CBWDataTransferLength ≠ 0であり、Direction bit はバルクアウト転送 (Data-Out) のため０となります。基本的な最低条件としてホストがデータを全く送らないというのは許されません。

ホストに要求されること

１、ホストは有効で正常な CBW を送信すること。

２、ホストはデバイスにデータを送ること。

３、STALL コンディションのときにデータを送ってしまった場合はバルクアウトパイプをクリアーすること。

４、しっかり CSW を受け取ろうとすること。

５、STALL コンディションで CSW を受信した場合、ホストはバルクインパイプをクリアーし、、もう一度新たに CSW を受け取ろうと試みること。

６、CSW が有効で正常であったとき、

・Ｈｏ＞Ｄｎ、Ｈｏ＞Ｄｏ、Ｈｏ＝Ｄｏのとき

CSWStatus = 00h or 01h にセットします。また CSWDataResidue の値と CBWDataTransferLength の違いから実際に転送処理された量を決定します。

・Ｈｏ<>Ｄｉ、Ｈｏ＜Ｄｏのとき

CSWStatus フィールドの値が 02h だった場合、ホストは CSW-DataResidue の値を無視します。またリセットリカバリーを行ないます。

デバイスに要求されること
１、デバイスはしっかりCBWを受け取ること。
２、CBWが有効で正常であったとき、コマンドを実行しようと試みます。
・Ｈｏ＞ Ｄｎ、Ｈｏ＞ Ｄｏ、Ｈｏ＝ Ｄｏのとき
デバイスはホストから指示されたデータ量よりも少ないか等しく転送処理をする場合、デバイスはデータを受け取り、指示されたデータ量を受け取るか、早急にバルクアウトパイプをSTALLして転送を終了させるかのどちらかをします。デバイスはCSWStatus = 00h or 01hにセットして、デバイスに指示されたデータ量とデバイスに実際に処理された転送量の違いをCSWDataResidueにセットします。

・Ｈｏ <> Ｄｉ、Ｈｏ＜ Ｄｏのとき
デバイスがホストが指示したより多くのデータを処理しようとしているか、データを送ろうとしているとき、デバイスはCBW-DataTransferLengthより多くデータを受け取ることになります。デバイスはCBWDataTransferLengthのデータ分だけ受け取るか、早急にバルクアウトパイプをSTALLして転送を終了させます。デバイスはCSWStatus = 02hにセットしてフェーズエラーをホストに伝えます。
３、デバイスは有効で正常なCSWを返さなければなりません。 ・この場合、CSWStatus ≠ 00h or 01hのときは、バルクインパイプをSTALLさせる必要があります。

第 IV 部
製作にむけて

13　実機ログの検証

USB の外付けハードディスクのログをとり、実際の検証してみます。まず基本的事項であるプロトコルコードですが、実際に USB マスストレージクラスのバルクオンリー転送を使用していました。

13.1　サブクラスコード

注目すべきはサブクラスコードで SubClass = 06h =SCSI でした。外付けハードディスクドライブを構築する際、コマンドを知らないと、実際にはデータの読み出し、書き出しなど一切できないので、SCSI コマンドもある程度知っておく必要があるということです。

13.2　CBW

CBW についてですが、３１バイト全てにデータが乗っていなければいけないのは前述のとおりです。そこで実際に使わないデータ部分ですが、どうやらゴミデータなのでデータの値は気にしないでいいようです。実機では全て値０で送られていました。また、CBW はバイトが反転して送られてくるはずなのに、なぜか CBWCB はバイトが反転して来ていませんでした。この点は、バルクオンリー転送の核になる部分なのでデバイス構築の際に注意しなければいけないと思われます。

13.3　USB の仕様

今回ログをとったドライブはペイロードサイズ等から USB1.1 仕様だと思われます。しかし多少転送速度に不満がでてくるのではないでしょうか? やはり、USB2.0 仕様のマスストレージデバイスの構築を目指すのがいいと思われます。デバイスコントローラーに前述した「HS デバイスの状態遷移」等の概念を取り込み、PING フロー制御のプロトコル等を採用するといいてしょう。

14　実機のログにて使用されていたコマンドの解説

今回使用されていて確認したSCSIコマンドは全部で6種類ありました。SCSIコマンドはバージョンも多く存在しているので、目的に合わせたバージョンを使用するといいでしょう。

14.1　Inquiry(オペコード=12h)

Inquiryコマンドはホストがデバイスに対して、論理ユニットの構成要素やパラメーター情報を要求するときに使用します。
具体的にはSCSIのバージョン情報、プロダクト情報などです。

14.2　Read Capacity(オペコード=25h)

Read Capacityコマンドはホストがデバイスから記憶領域の容量情報を欲しているときに
使用します。

14.3　Read(10)(オペコード=28h)

Read(10)コマンドはホストにデバイスサーバー(集配信装置)からデータを転送するときに使用します。

14.4　Mode Sense(6)(オペコード=1A)

Mode Sense(6)コマンドはデバイスサーバーがホストにパラメータを伝えるときに使用するコマンドです。

14.5　Test Uniti Ready(オペコード=00h)

Test Uniti Readyは論理ユニットが準備できているかをチェックするためのコマンドです。ホストからのコマンドに応えられるときは、GOOD Statusで応答します。

14.6 Vendor-Specific(オペコード =23h)

Vender-Specific がその機器が特有に持っているコマンドで、自由に設定できます。オペコードは
02h,05h,06h,09h,0Ch,0Dh,0Eh,0Fh,10h,13h,14h,19h,
20h,21h,22h,23h,24h,26h,27h,29h,2Ch,2Dh,C0h〜FFh
で、これらのオペコードはすべてベンダ特有のコマンドを表します。

15 まとめ

USB マスストレージデバイスを構築するためには、まず当たり前ですが、USB プロトコルの基本的な部分をおさえてなければいけません。特にディスクリプタの部分をおさえなければ、デバイスがどんなもので、どんな機能を持っているかを理解できません。また、将来的に USB2.0 仕様対応を目指すためにも USB2.0 仕様と USB1.X 仕様の違いも理解する必要があるでしょう。特にデバイスコントローラーに PING フロー制御を導入するといいと思われます。
重要部分のマスストレージクラスについてですが、転送の流れ (概念) をしっかり理解する必要があります。ファームウェアを構築しようとする際、転送の流れは主要部分となってきます。その関係で CBW、CSW についてもしっかり理解する必要があります。また、エラー検出等のためにもバルクオンリー転送が全て、１３に場合分けができるというのも、頭にいれる必要があるでしょう。それと同時に、それぞれのケースにはどういう対応をとればいいか、確認する必要があります。
具体的にはファームウェアを構築する際に、主に２つのパートに分けて考えます。
１つはデバイスの制御部分。これはは CBW、CSW の概念をを取りこむこと。２つ目がデバイスの CSW 処理方法部分。これには１３ Cases の概念を採用することで、全ての処理方法をカバーできます。データの値を比較するだけで処理方法が判断できます。つまり比較演算さえすればいいのです。
この２つパートを以下のように組み合わせます。
１、デバイス制御部分により、CBW を取りこむ。これにより、必要なレジスタに値が入力される。また実行内容が決まる。

２、レジスタを CSW 処理部分に与え、CSW の値を決める。
３、制御部分に CSW をホストに伝えさせる。
この機能をハードディスクに組みこめばいいと思われます。

最後にサブクラスコードです。今回サブクラスコードは SCSI を使用していました。記憶装置では現在、SCSI コマンドが標準的につかわれています。実機でも採用されているので、ハードディスク構築する際には、SCSI コマンドを採用するのがいいでしょう。他のマスストレージデバイスを構築する際は、機器にあわせてサブクラスコードを選択する必要があるでしょう。

第Ⅴ部
感想

USBデバイスの基礎から勉強してきて、マスストレージデバイスを構築するためにはどんな手順でどんな知識が必要かが理解できました。よければ後輩がUSB2.0仕様対応の研究も進めて高性能で独自のハードディスクドライブ、またはマスストレージデバイスを研究していってくれれば自分の研究は有意義なものになったと言えることでしょう。

参考文献

[1] 'TECHI Vol.8 USBハード&ソフト開発のすべて"、CQ出版社

[2] 'Universal Serial Bus Mass Storage Class Specification Overview"、http://www.usb.org

[3] 'Universal Serial Bus Mass Storage Class Bulk-Only Transport"、http://www.usb.org

[4] 'Infomation technology-SCSI Primary Commands-3(SPC3)"、http://www.t10.org

[5] 'Infomation technology-SCSI-3 Block Commands(SBC)"、http://wwwt10.org