

USB2.0概說



Rev. 1.0E-05/21

目 次

USB(Universal Serial Bus)とは	3
USB構成概要	4
エンドポイント	4
インターフェース	4
パイプ	4
データ転送の構造	5
帯域幅の管理	5
装置接続と列挙(Enumeration)	6
 電気的な仕様	7
バス速度の識別	8
電力供給(VBUS)	8
一時停止(Suspend)電流	9
一時停止(Suspend)動作への移行	9
データ転送速度	9
USBコネクタと信号配置	9
NRZI(Non Return to Zero Invert)符号化	10
ビット挿入(0挿入)	10
 USB規約(Protocols)	11
USBパケット共通の領域(Fields)	11
同期(SYNC)	11
パケット識別(PID)	11
アドレス(ADDR)	11
エンドポイント(ENDP)	11
巡回冗長検査(CRC)	11
パケット終了(EOP)	11
処理単位(Transaction)とUSBパケット種別	12
フレーム開始(SOF)パケット	12
指示票(トークン)パケット	12
データパケット	12
ハンドシェークパケット	12
 転送種別	13
制御(Control)転送	13
設定(Setup)段階	13
データ(Data)段階	13
状態(Status)段階	14
制御(Control)転送に於ける大量データ	15
割り込み(Interrupt)転送	16
等時(Isochronous)転送	16
大量(Bulk)転送	17
 USB記述子(Descriptor)	18
USB記述子(Descriptor)の構成	19
装置記述子(Device Descriptor)	19
構成記述子(Configuration Descriptor)	20
インターフェース記述子(Interface Descriptor)	21
 エンドポイント記述子(Endpoint Descriptor)	22
文字列記述子(String Descriptor)	23
 設定(Setup)パケット	24
標準要求(Standard Request)	25
標準装置(Device)要求	25
標準インターフェース(Interface)要求	26
標準エンドポイント(Endpoint)要求	26
 引用文献	27
 付録	28
付録-A: CRC	28
付録-B: 言語識別子	29

USB(Universal Serial Bus)とは

USB1.1は12MbpsのFull-Speed動作と1.5MbpsのLow-Speed動作の2つの速度を支援します。1.5Mbps動作は低速でEMIの影響を受けていたため、部品性能とフェライトビーズの費用を削減します。例えばクリスタル発振子をより安いセラミック振動子に置換できます。USB2.0はFirewire(IEEE1493a)に匹敵するよう480MbpsのHigh-Speed動作が追加され、今日殆どのPCで採用されています。

USBのバスはホストによって制御されます。バスには唯一のホストだけが存在できます。仕様で如何なる形式での複数マスター(ホスト)も支援しませんが、ホストの規則について2つの装置での調停を許すホスト調停規約を導入するために、USB2.0に標準でOn-The-Go仕様が追加されました。これは単一の点-点間接続に限定した制限されたホスト機能を意図したものです。ホストは帯域幅の使用方法と全ての処理(転送)の保証について責任があります。データは指示票(Token:トークン)を基準とした規約を使い、様々な処理方法によって送信できます。

USBの接続形態は順次結線(Daisy chain)ではなく、イーサネットのような星状結線(Star chain)です。これは拡張に際して何処かに費用を増大させるハブ(HUB)の使用を生じさせます。けれども現状のPCでのUSBコネクタ数の多さと、ハブ内蔵機器により、これが問題になるとは思われません。星状結線は装置間を単に順次結線するよりもいくつかの利点があります。各装置の電力を監視し、過電流状態が起きた場合に他のUSB装置を分断(切断)することなくOFFに切り替えられます。Low-Speed装置がHigh-SpeedとFull-Speedの通信を受信しないようにハブがそれらを止めることで、High-Speed, Full-SpeedとLow-Speed装置を支援できます。

1つのUSBバスには127個までの装置が接続できます。初期のUSBは1つのホスト制御器と2つのコネクタで構成されたものが多く、従ってこれらは利用可能な同じUSB帯域幅を2つのコネクタで共用していました。現在では4,5個のホスト制御器と各制御器に対応した1つのコネクタの形式が一般的です。従って各コネクタに対してUSB帯域幅がそれぞれ最大限に利用できます。

USBホスト制御器にはそれ自体の仕様があります。USB1.1ではハードが軽くソフトが重いUHCI(Universal Host Controller Interface)とソフトが軽くハードが重いOHCI(Open Host Controller Interface)の2つのホスト制御器仕様があります。USB2.0の導入においてUSB2.0の詳細仕様をレジスタ段階で記述するために新しいホスト制御器仕様が必要とされ、EHCI(Enhanced Host Controller Interface)が生まれました。主要な制御器はこれに従い、故に新しい1つのドライバのみの実装で済むようになります。

USBはシリアルバスと言う名前のように、2本の電源線(VBUSとGND)と2本のツイスト対による差動データ信号線の合計4本を使います。信号線ではホストと装置のクロックを同期させる同期(SYNC)領域と送出データに対してNRZI(Non Return to Zero Invert)符号化が使われます。

USBは動的に読み込み/解除可能なドライバでプラグ&プレイを支援します。使用者は単に装置をバスに(プラグを)接続するだけです。ホストはこの追加を検知し、新規に挿入された装置に問い合わせ、適切なドライバを読み込むでしょう。初回接続時、その装置に対して提供されたドライバが要求されるかもしれません。使用者はIRQやI/Oアドレスのような用語や、コンピュータの再起動、終了について悩む必要はありません。その装置の使用が終われば、単にプラグを抜きケーブルを取り外せ、ホストはそれが存在しないことを検知して自動的に読み込まれたドライバを解除します。

適切なドライバの読み込みは供給者識別(VID:Vendor ID)と製品識別(PID:Product ID)の組み合わせを使って行われます。供給者識別(VID)は有償でUSB Implementor's Forumから供給されます。最新の情報は<http://www.usb.org/developers/vendor/>で得られます。

他の標準化機構では教育・研究または趣味のような非商業活動に余分なVIDを供給しています。しかしUSB Implementor's Forumは未だこのサービスを提供していません。このような状態では、開発システム製造業者に割り当てられたものを使いたいと思うかもしれません。例えば殆どのチップ製造業者は、商用装置として存在しないことが明白で、外部から依頼されたチップに使用可能なVIDとPIDの組み合わせを持っているでしょう。チップ製造業者はこの商用装置用にこれらのVIDと共に使うPIDを更に販売することができます。

特記すべきUSBの他の特徴として転送種別があります。USBは、制御(Control)転送、割り込み(Interrupt)転送、等時(Isochronous)転送、大量(Bulk)転送を支援します。等時(Isochronous)転送は遅れを保証するために、定義された帯域幅での巡回予約を装置に許します。混雑がデータ損失やフレーム落ちを起こすかもしれないオーディオやビデオの応用でこれは理想的です。各転送種別は保証された遅延と帯域幅と、異常の検出/回復のような領域の使用との間で交換条件の選択を設計者に提供します。

USB構成概要

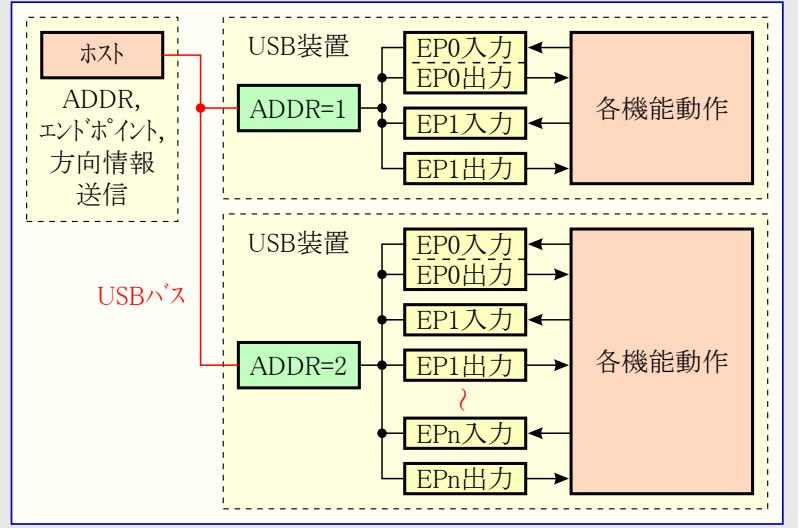
USB装置について考察すると、主にUSBバス側の制御とUSB装置としての本来の動作対象制御が考えられます。ここでは印刷機などの周辺装置の機能や能力を提供するUSB装置と見える標準化された仕組みについて記述します。

殆どのUSB機能はUSB制御器IC内で処理(Transaction)層までの低位USB規約が扱われます。けれども多くのUSB制御器はパケット識別(PID)符号復号化異常のような異常を報告してきます。従って最低限の下位層の知識も必要です。

多くのUSB機能は通常8バイト長の直列緩衝部を持ちます。各緩衝部は(EP0 IN, EP0 OUT等の)エンドポイントに所属します。例えばホストが装置記述要求を送信した場合を考えます。装置側では設定(Setup)パケットを受信して、そのパケットが自分宛かをアドレス領域から判断し、そなればその設定(SETUP)指示票(トークン)のエンドポイント領域の値で指定された適切なエンドポイントの緩衝部へ後続するデータパケット内のデータ部を複写します。そしてこのデータ受信に応答するためのハンドシェークパケットを送信し、パケットが受信されたことを示すエンドポイントに対応したマイクロコントローラへの割り込みを生成します。これは通常、USB制御器などのハードウェアによって行われます。

ソフトウェアはこの割り込みを受け、そのエンドポイントの緩衝部内容を読み、その内容の装置記述要求を解析すべきです。

図1. エンドポイント経由でのデータの流れ



エンドポイント

エンドポイントはデータの供給元または受け取り先として記することができます。バスがホスト中心のため、エンドポイントはUSB機能で通信路の終端と見做せます。例えばソフトウェア層でデバイスドライバがUSB装置のEP1へパケットを送信したとします。ホストから続いてデータが送られると、それはUSB装置のEP1出力緩衝部に行き着きます。その後USB装置のファームウェアが余裕のある時にそのデータを読み込むでしょう。それがデータ送信を要求している場合、ホストがバスを制御しているため、単にバスへ書く訳には行きません。従ってデータをEP1入力緩衝部に書き、このデータはホストがデータを要求したエンドポイントに(IN)パケットを送信するまで、その緩衝部内に保持されることになります。このようにエンドポイントは装置機能のハードウェアと装置機能上で走行するファームウェア間のインターフェースとして見ることもできます。

全ての装置はエンドポイント0(EP0)を支援しなければなりません。これは全ての装置に於いてバス上で装置が使用可能な間の列挙中の制御と状態情報の要求が受信されるエンドポイントです。

インターフェース

インターフェースは装置で特定動作を司るエンドポイント群の集合体です。従ってインターフェース自体は緩衝部などの資源を直接持たずに、エンドポイントを保持することで間接的に緩衝部を保持します。通常、このインターフェースと言う単位は、この上層である構成層内で異なる、構成間を切り替える時の単位に使います。

パイプ

装置側が直列のエンドポイントでデータを送受信する一方で、ホスト側プログラムはパイプを通じてデータを転送します。このパイプはホストとエンドポイント間の論理的な接続です。パイプは割り当てられた帯域幅、使う転送種別(制御(Control), 割り込み(Interrupt), 大量(Bulk), 等時(Isochronous))、データの転送方向、パケットや緩衝部の最大容量などに関連したパラメータ群も持ります。例えば既定のパイプはエンドポイント0入力とエンドポイント0出力で構成された制御(Control)転送種別の双方向パイプです。

USBでは2種類のパイプが定義されています。

メッセージ パイプ メッセージ パイプには定義されたUSB形式があります。これらはホストから送信された要求で始まり、ホストによって制御されます。そしてデータはその要求によって命令され、望まれた方向に転送されます。従ってメッセージ パイプは双方向の転送をデータに許しますが、これは制御(Control)転送だけを支援します。

ストリーム パイプ ストリーム パイプには定義されたUSB形式がなく、どのデータ形式でもストリーム パイプへ送信でき、他方からのデータを取得できます。データは入力(IN)または出力(OUT)の予め定義された方向で順次転送されます。ストリーム パイプは割り込み(Interrupt), 大量(Bulk), 等時(Isochronous)転送種別を支援します。ストリーム パイプはホストまたは装置のどちらかで制御できます。

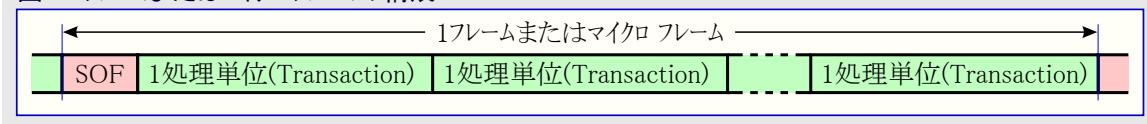
データ転送の構造

USBでは单一バス上で複数の装置機能を扱うため、多くの工夫が成されています。このため1対1のRS-232のように簡単な手順での通信は望めません。最初に理解しなければならないのは、複数の転送要求を見かけ上同時処理するためにバスが時分割で使われることです。これは**フレーム**と言う単位で処理されます。High-Speedバスの場合は更に1フレームが8つの**マイクロフレーム**に分割されます。従ってLow/Full-Speedバスでは(公称)1ms毎、High-Speedバスでは(公称)125μs毎にフレームまたはマイクロフレーム処理が繰り返されます。

これらのフレームまたはマイクロフレームは**フレーム開始(SOF)パケット**で実際の処理が開始されます。SOFパケットは単独で用いられ、後述の一般的な**処理単位(Transaction)**構造に従いません。即ち指示票(Token)パケット単独で使われます。

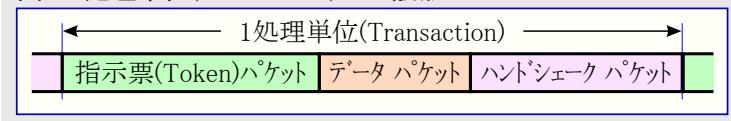
SOFパケットに続いて処理単位(Transaction)群の処理が行われます。応用での或るデータ転送は一般的にドライバなどの下層で制御部やデータ部などの複数部分で構成され、それらの転送によって応用でのデータが転送されます。USBではこれら下層での1転送を**転送(Transfer)単位**とし、1転送単位は複数の**処理(Transaction)単位**で構成されると考えます。従ってこの処理単位が各フレームまたはマイクロフレーム内で順次転送処理され、いくつかのフレームまたはマイクロフレームで1つの転送が完了します。1フレームまたはマイクロフレーム内の構成例を次に示します。

図3. 1フレームまたはマイクロフレームの構成



1処理単位は基本的に指示票(Token)パケット、データパケット、ハンドシェークパケットの3種類で構成されます。

図4. 1処理単位(Transaction)の一般形



転送単位は転送種別によって異なり、**制御(Control)転送**のみが複雑な構造です。制御(Control)転送は**設定段階**、**データ段階**、**状態段階**の3段階からなります。設定段階と状態段階は各々1つの処理単位(Transaction)で構成されます。データ段階は1つ以上の処理単位(Transaction)で構成されます。他の転送種別では基本的にデータ段階のみが用いられます。

フレーム/マイクロフレームと転送単位は直接的な関係はありません。転送単位がフレーム/マイクロフレーム内で完了するとは限りません。その場合は複数のフレーム/マイクロフレームを使って実行されます。フレーム/マイクロフレームでの実際の処理はバス速度に対応する帯域幅や転送種別などによってホストによって決定されます。

図2. フレームとマイクロフレームの関係

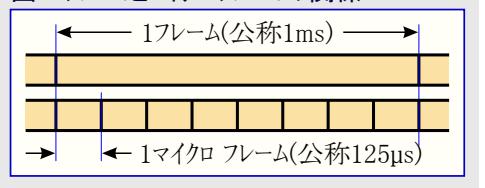


図5. 1転送単位との関係



帯域幅の管理

ホストにはバスの帯域幅を管理する責任があります。これは割り込みと等時のエンドポイント設定時の**列挙**によってバス動作全体に渡って行われます。仕様はFull-Speedバス上での周期的(割り込みと等時)転送に対して、どのフレームでも90%を越えて割り当てられないよう制限が設けられています。High-Speedバスでのこの制限はマイクロフレームの80%を越えないことに減らされていて、これを周期的転送に割り当することができます。

高度に飽和しているバスでも、残りの10%が制御(Control)転送と、一旦割り当てられた**大量(Bulk)転送**に残され、大量転送は細切れに転送されることになるでしょう。

装置接続と列挙(Enumeration)

列挙はどの装置が今バスに接続されたか、またそれが消費電力、エンドポイントの種別や数、製品のクラスなどどのようななどのパラメータを必要とするかを判定する過程の処理です。その後ホストは装置にアドレスを割り当て、バスにデータを転送することを装置に許す設定を許可します。この詳細に関してはUSB仕様書で詳述されますが、ここでは列挙中にホストがどう応答するかを中心に簡潔に概要を示します。

Windowsの列挙は共通で、次の手順で進められます。

1. ホスト(またはハブ)はデータ信号線上で装置のプルアップ抵抗によって新規装置の接続を検知します。ホストはプラグが完全に挿入され、装置への安定給電を可能とするため、最低100ms待機します。
2. ホストは装置を既定状態に置くためにリセットを送出します。これで装置は既定のアドレスで応答できるようになります。
3. Windowsのホストは装置記述(Device Descriptor)の先頭64バイトを要求します。
4. 装置記述(Device Descriptor)の最初の8バイト受信後直後、更にバスリセットを行います。
5. ホストは装置をアドレス指定状態にするアドレス設定(SET_ADDRESS)命令を送出します。
6. ホストは装置記述(Device Descriptor)の18バイト全体を要求します。
7. そして容量関係を決めるために構成記述(Configuration Descriptor)の9バイトを要求します。
8. ホストは設定記述(Configuration Descriptor)の255バイトを要求します。
9. それらで指定されているなら、ホストは対応する文字列記述(String Descriptor)を要求します。

手順9.の後、Windowsは装置に対するドライバについて要求します。その後、構成選択(SET_CONFIGURATION)命令を行う前に全ての記述を要求すると考えるのが一般的です。

上記の列挙手順はWindows 98SE, 2000, XPで共通です。手順4.は初めてファームウェアを作成する場合に度々混乱させます。ホストが装置記述(Device Descriptor)の先頭64バイトを要求し、その先頭8バイト受信後にホストが装置をリセットするのは、装置記述(Device Descriptor)に何か不正はないか、装置のファームウェアが要求をどう扱うかを調べるためにだけと考えるのが自然です。実際の装置記述(Device Descriptor)取得は手順6.で行われます。

通常、記述や送出法で何か不正があると、ホストは長い要求間隔で3回の読み込みを試みます。ホストは3度目の試行後に諦めて装置の異常を報告します。

電気的な仕様

USBホストやハブまたはUSB制御器や送受信部の設計を行わない限り、USB2.0仕様の電気的特性を全て知る必要はありません。ここでは要点だけを記述します。

USBはデータ用に1対の差動送信を使います。これはNRZI符号化され、データ列内に充分なレベル遷移を保証するためにビット挿入が行われます。Low-SpeedとFull-Speedの装置での差動1はGNDへの15kΩプルダウン抵抗付きでDATA+を2.8Vより高く、GNDへの15kΩプルダウン抵抗と3.6Vへの1.5kΩプルアップ抵抗付きでDATA-を0.3Vより低くにすることで送出されます。他方の差動0は同様のプルアップ/プルダウン抵抗付きでDATA+が0.3Vより低く、DATA-が2.8Vより高くなります。

受信側ではDATA+がDATA-より0.2Vより高い時に差動1、DATA+がDATA-より0.2V以上低い時に差動0として定義されます。信号の極性はバス速度に依存して反転されます。従って用語としてJとK状態が論理レベルの指示に使われます。Low-SpeedではJ状態が差動0で、High-SpeedではJ状態が差動1です。

USB送受信部はシングルエンドと差動の両出力を持ちます。或るバス状態はDATA+, DATA-または両方のシングルエンド信号で示されます。例えばシングルエンドの0またはSE0は10ms以上保持した場合に装置リセットの指示に使うことができます。SE0はDATA+とDATA-両方をLow(<0.3V)に保持することによって生成されます。

Low-SpeedとFull-Speedのバスは $90\Omega \pm 15\%$ の特性インピーダンスを持ちます。従ってDATA+とDATA-に対してインピーダンス整合の直列抵抗を選ぶときに、データシートでのこれらの確認が重要です。

High-Speed動作では雑音を削減するために17.78mAの定電流駆動を使います。

表1. Low/Full-Speed信号レベル一覧

バス状態 Speed	出力条件	入力条件	
		必要条件	許容条件
差動1	D+>V _{OH} (min) 且つ D-<V _{OL} (max)	D+>V _{IH} (min) 且つ (D+)-(D-)>0.2V	(D+)-(D-)>0.2V
差動0	D+<V _{OL} (max) 且つ D->V _{OH} (min)	D->V _{IH} (min) 且つ (D-)-(D+)>0.2V	(D-)-(D+)>0.2V
SE0(シングルエンド'0)	D+とD-<V _{OL} (max)	D+とD-<V _{IL} (max)	D+とD-<V _{IH} (min)
SE1(シングルエンド'1)	D+とD->V _{OSE1} (min)		D+とD->V _{IL} (max)
データ J状態	Low	差動0	
	Full	差動1	
データ K状態	Low	差動1	
	Full	差動0	
アイドル状態	Low	駆動なし	D->V _{IHZ} (min) 且つ D+<V _{IL} (max)
	Full	駆動なし	D+>V _{IHZ} (min) 且つ D-<V _{IL} (max)
回復(Resume)状態		データK状態	
SOP(パケット開始)		アイドルからK状態への遷移	
EOP(パケット終了)	約2ビットのSE0+1ビットのJ状態	1ビット以上のSE0+1ビットのJ状態	1ビット以上のSE0+J状態
切断	駆動なし		2.5μs以上のSE0
接続	駆動なし	2ms以上のアイドル状態	2.5μs以上のアイドル状態
リセット	10ms以上のD+とD-<V _{OL} (max)	10ms以上のD+とD-<V _{IL} (max)	2.5μs以上のD+とD-<V _{IL} (max)

注: D+, D-は各々DATA+, DATA-を表します。各指標に対する具体的な数値についてはUSB2.0仕様書をご覧ください。

バス速度の識別

USB装置はDATA+またはDATA-のどちらかの信号線を3.3Vにプルアップすることでその速度を示さなければなりません。Full-Speedの装置は右図で示されるように自分がFull-Speed装置であることを示すためにDATA+へ接続されたプルアップ抵抗を使っています。装置端でのこれらのプルアップ抵抗は接続された装置の存在を検知するためにホストやハブによって使われます。プルアップ抵抗がない場合、バスに何も接続されていないと見做されます。いくつかの装置ではファームウェア下でON/OFFできる抵抗が組み込まれたチップを持ちますが、それ以外では外部抵抗が必要になります。

ファームウェア下でON/OFF可能なプルアップ抵抗の場合、接続時の電源供給開始で装置の初期化を行った後にプルアップ抵抗をONにすることで自身の初期化時間を充分に取れます。また接続したままで一度OFFにしてからONにすることで、新規装置の認証となりますので、例えば装置機能を変更し、別の機能の装置としての使用が接続したままで行えます。

High-Speed動作についてはここで示されていません。High-Speed装置はFull-Speed装置(DATA+の1.5kΩでの3.3Vへのプルアップ)としての接続で始まります。そしてリセット中にHigh-Speed動作を要求し、ホストまたはハブがHigh-Speed動作を支援するなら、High-Speed接続を確立します。そして装置がHigh-Speedで動作する場合、プルアップ抵抗は取り去られ、平衡信号路になります。

USB2.0適合装置は必ずしもHigh-Speed動作を支援する必要はありません。これは速度が重要でない安価な装置の製造を可能にします。これはFull-Speedを支援しないUSB1.1適合のLow-Speed装置の場合にも言えます。

High-Speed装置はLow-Speed動作支援の必要はありません。接続の初めでFull-Speed動作が必要なだけで、交渉が成立すると、その後はHigh-Speedで動作します。USB2.0適合の下方向(Downstream)装置(ホストまたはハブ)はHigh-Speed, Full-Speed, Low-Speedの3つ全ての動作種別を支援しなければなりません。

電力供給(VBUS)

USBの利点の1つはバス電力供給装置(追加ケーブルや外部電源の必要がなく、バスから電力を得る装置)です。多くの人々は必要な判断基準の全てを最初に考慮することなく、この選択に飛び付くでしょう。

USB装置は後述される構成記述内に於いて2mA単位で表される消費電力を指定します。装置は例え外部電源を失ったとしても、列挙中に指定した値より大きく消費電力を増やせません。これらはUSB装置の3つ種別です。

- 低電力バス給電装置(Low-power bus powerd functions)
- 高電力バス給電装置(High-power bus powerd functions)
- 自己給電装置(Self-powerd functions)

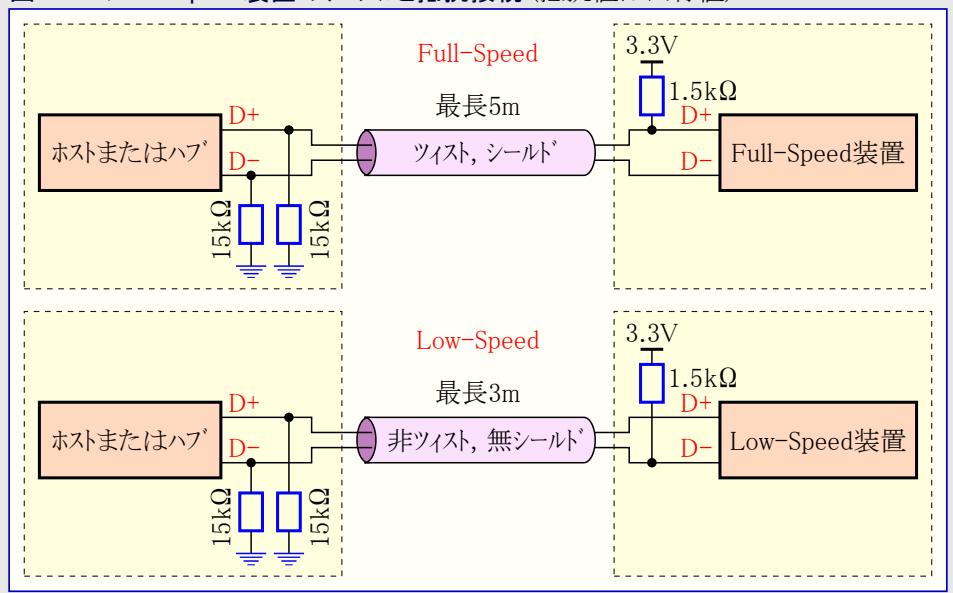
低電力バス給電装置はVBUSからその電力の全てを得ますが、1負荷単位を越える電力を得ることはできません。USB仕様で1負荷単位は100mAとして定義されます。低電力バス給電装置は装置からの上方向(Upstream)側プラグで4.40～5.25Vの範囲で動作するように設計されなければなりません。従って多くの3.3V動作装置では低損失電圧調整器が必須となるでしょう。

高電力バス給電装置はVBUSからその電力の全てを得ますが、初期設定が成されてしまうまで、1負荷単位を越えて電力を得ることはできません。状態情報(設定)記述の問い合わせ後、5負荷単位(最大500mA)が提供され、それを得ることができます。高電力バス給電装置は最小4.40Vで検出と列挙ができなければなりません。全負荷単位で動作する場合、4.75Vの最小VBUSは最大の5.25Vで指定されます。これらは装置からの上方向(Upstream)側プラグ上での値です。

自己給電装置はVBUSから1負荷単位までの電力を得、残りは外部電源から得ます。この外部電力が不足または失われた場合、バスからの給電が1負荷単位を越えない処置が成されなければなりません。自己給電装置は消費電力についての諸問題がないため、仕様に対する設計が容易です。このバスからの1負荷単位供給は他の電源なしでの装置の検出と列挙を可能にします。

バス給電または自己給電の何れでもないUSB装置は上方向(Upstream)側バスのVBUSを駆動できます。VBUSが失われたとき、速度識別に使ったDATA+/DATA-のプルアップ抵抗からの電力がかなりの時間残留することに注意してください。他の考慮点として突入電流は制限されなければなりません。これはUSB仕様書の7.2.4.1節に概要がありますが、一般的に見落とされがちです。突入電流は装置でのVBUSとGND間の容量によって引き起こされます。故に仕様で最大容量(10μF)が指定されます。インターフェンス性USBケーブルを通して電流が流れた後に装置を切断するとケーブル端に反転電圧が発生し得ます。これを防ぐために最小1μFのVBUSデカップリング容量が指定されています。

図6. Low/Full-Speed装置のケーブルと抵抗接続(抵抗値は公称値)



一時停止(Suspend)電流

一時停止動作は全ての装置で必須です。一時停止中は追加の制限が強制されます。最大一時停止電流は負荷単位に比例し、1負荷単位の装置(既定)は0.5mAです。これにはバスのプルアップ抵抗からの電流も含みます。ホストやハブはDATA+とDATA-に15kΩのプルダウン抵抗を持ちます。このプルダウン抵抗は消費電力に関して装置の1.5kΩプルアップ抵抗と直列になり、代表値3.3VのVTERMで16.5kΩの合計負荷になります。従って一時停止開始以前からこの抵抗が0.2mA電力を消費しています。

多くの装置での他の考慮点は3.3V電圧安定器です。USB装置の多くは3.3Vで走行します。シリーズバス型調整器は平均的な静止時電流が0.6mA程度と一般的にかなり非効率で、故により効率の良い、即ち高価な調整器が必要になります。多くの場合で0.5mAの制限内に抑えるため、マイクロコントローラのクロックを低下または停止させなければならないでしょう。

殆どのホストやハブは数mA程度のような過負荷を検知する能力がありませんので、USB仕様に違反してそのようにした場合でも動作を維持できるでしょう。けれども100mAまたは許可された負荷を超える試みは、ホストまたはハブがこれを検知してバスの保全のためにその装置を切断すると予想されます。

勿論自己給電装置としての設計を選択すれば、これら設計上の問題点は避けることができます。この一時停止電流はデスクトップコンピュータについてはあまり関係ありませんが、On-The-Go仕様でUSBホストが組み込まれた小型機器にとっては重要なことです。これらの機器から引き出される消費電力は電池での動作時間に大きく影響するでしょう。

一時停止(Suspend)動作への移行

USB装置はバス上の動きが3.0ms以上なければ一時停止動作へ移行するでしょう。その後停止(Shutdown)まで更に7.0msあり、従ってバスの動き停止後の10ms間は、指定された一時停止電流だけで動作しなければなりません。一時停止されたホストまたはハブとの接続を維持するため、装置は一時停止中も速度選択用のプルアップ抵抗に電力を供給しなければなりません。

USBはフレーム開始(SOF)パケットや周期的なバス送信でバスの活性状態を維持します。これはデータがない状態でアイドル状態のバスが一時停止動作へ移行するのを防ぎます。High-Speedバスでは $125.0\mu s \pm 62.5\text{ns}$ 毎にマイクロフレームを送信します。Full-Speedバスでは1ms $\pm 500\text{ns}$ 毎にフレームを送信します。Low-SpeedバスではどのLow-Speedデータもない状態でだけ、1ms毎のパケット終了(EOP)がバス活性を維持します。

用語の全一時停止(Global Suspend)はUSBバス全体を纏めて一時停止動作へ移行するときに使われます。また装置はそれが接続されているホストまたはハブからの命令によって一時停止することもできます。これは選択的一時停止(Selective Suspend)として参照されます。

装置はアイドル状態でない何かの受信時にその動作を再開します。装置が遠隔起動許可なら、装置は一時停止からの再開をホストへ告げるかもしれません。

データ転送速度

見落とされがちな項目としてUSBクロックの許容誤差があります。これはUSB仕様書の7.1.11節で規定されます。High-Speedデータは480Mbps $\pm 0.05\%$ 、Full-Speedデータは12Mbps $\pm 0.25\%$ 、Low-Speedデータは1.5Mbps $\pm 1.5\%$ で転送されます。これは低価格Low-Speed装置用にセラミック振動子の使用を可能にしますが、Full-SpeedやHigh-Speed装置に対しては許されません。

USBコネクタと信号配置

全ての装置はホストへの上方向(Upstream)接続で、全てのホストは装置への下方向(Downstream)接続になります。上方向と下方向のコネクタは構造的に交換できず、従って下方向ポートへ接続した下方向ポートのようにハブでの不正なループ接続をなくします。これらは右で示されるような俗にA型とB型と呼ばれる2つの形式のコネクタです。

A型プラグは常に上方向側です。A型ソケットは一般的にホストとハブで見つかるでしょう。A型ソケットはホストとなるPCとハブで共通です。B型プラグは常に下方向側で、その結果B型ソケットは装置で見れます。

いくつかの販売店で見られるA型とA型の直結ケーブルとオス/メス(プラグ/ソケット)変換器は面白い構成です。これはUSB仕様に矛盾します。A型プラグとA型プラグの装置は2つのPC間の接続に使うプリッジだけです。その他の禁制ケーブルは一端がプラグで他端がソケット(A型またはB型のどちらか)を持つUSB延長ケーブルです。これらのケーブルはUSBのケーブル長の必要条件に違反するかもしれません。

USB2.0にはUSBミニBコネクタ導入の障害情報を含みます。これらのコネクタの詳細は<http://www.usb.org>のMini-B Connector Engineering Change Noticeで得られます。ミニコネクタが遅れた理由は携帯電話などの小型電子装置の領域から来ました。現在のB型コネクタはこれらの装置内に統合するには大きすぎます。

USBにピアtoピアを付加するOn-The-Go仕様が公開されています。これは携帯電話などの小型機器にUSBホストを導入し、それ故ミニAコネクタ仕様を含みます。

標準内部線色はUSBケーブルで使われ、製造時の線識別を容易にします。標準仕様にはケーブルについて様々な電気的特性が示されています。元となるUSB1.0仕様書に含まれる詳細を読むことは重要です。

図7. USBコネクタ(レセクタブル)

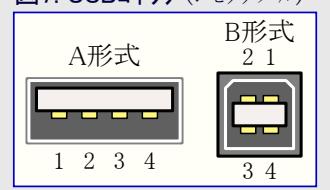


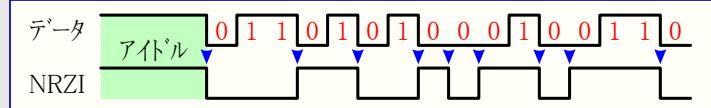
表2. USBピンの配色と機能

番号	配色	機能
1	赤	VBUS(+5V)
2	白	DATA-
3	緑	DATA+
4	黒	GND

NRZI(Non Return to Zero Invert)符号化

USB信号線上の信号はNRZIで符号化されます。NRZIでは各0が現在の信号レベルを変更することで表され、各1は現在レベルを保持することで表されます。従ってビット挿入に関しては論理データ列内の6つの連続する論理1後に1つの0ビットが挿入されることを意味します。

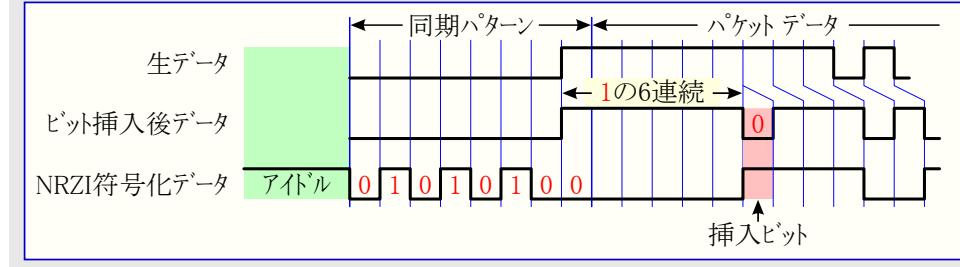
図8. データのNRZI符号化



ビット挿入(0挿入)

USBでのデータ受信は全ての時間で送信部と受信部が同期状態であることを満たさなければならず、従ってデータ列で連続する0または1の流れを送ることは許されません。これは信号線上のレベル遷移(端)で送受信部が同期を取るためです。従ってレベル遷移間が長いと同期ずれの危険が増すことになります。これから考えるとビット単位でレベル遷移があればよさそうですが、単純にこれを行うとEMIの増加や逆に雑音の影響を受け易くなります。USB規約ではビット挿入により同期を保証します。これはデータ列の6つの連続する0または1の後に1つの单一変更(1ビット)が挿入されることを意味します。USBではNRZI符号化が行われますので、元々のデータでの連続0または1の後に1つの0を追加することになります。従って、このビット挿入処理は元々のデータで連続する6つの1の後に1ビットの0を追加する操作になります。

図9. ビット挿入とデータ符号化手順



USB規約(Protocols)

送出データ形式が定義されていないRS-232や同様の直列インターフェースと異なり、USBは種々の規約階層が定義されています。一旦理解してしまえば、実際は上位階層についての心配だけです。現実に殆どのUSB制御器ICが下位階層の処理を行うため、最終製品設計者からそれらを隠します。USBの各 **処理単位(Transaction)**は次の3部分からなります。

- **指示票(Token)パケット** …… 本処理(Transaction)が何を行うかを装置へ通知するヘッダ
- **データ(Data)パケット** …… 一般で言う実データ部の転送(任意回数)
- **状態(Status)パケット** …… 本処理(Transaction)の応答と誤り訂正の意味提供に使用

既に説明されたようにUSBはホストが中心のバスです。従ってホストが全ての処理単位(Transaction)を開始します。この処理単位での最初のパケットは指示票(Token:トークン)と呼ばれ、どの装置アドレスのどのエンドポイントが指示され、その処理が読み込みか書き込みのどちらか、何が続くのかが記述されており、これはホストによって生成されます。次のパケットは一般的に実データ部を運ぶデータ(Data)パケットで、指示やデータが正しく受信されたか、エンドポイントが使用不能、または受け入れデータが使用不能などを報告するハンドシェークパケットがそれに続きます。

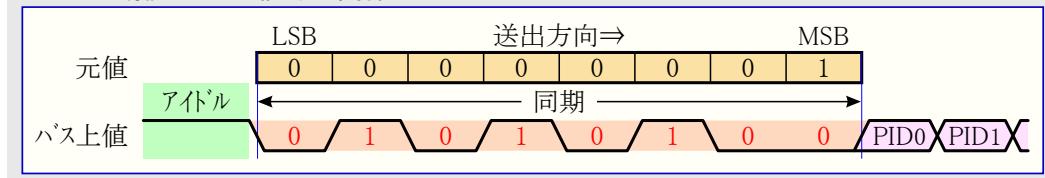
USBパケット共通の領域(Fields)

バス上のデータはLSBが先に送信されます。USBパケットは次の各領域から成ります。

同期 (SYNC)

全てのパケットはSYNC領域から始まらなければなりません。SYNC領域は8ビット長で、送信側と受信側の同期化に使われます。バス上での最後の2ビットはSYNC領域の最後(即ち次がPID領域であることを示します。下図に於いてバス上の論理値は装置速度によっては逆論理になります(Full-SpeedのDATA+として記載)。

図10. 同期値とバス上信号の関係



パケット識別 (PID)

PIDはPacket IDの略称です。本領域は送られるパケットの種別を示します。右表はこれに使われる値を示します。これらのPIDは4ビットですが、受信での正確さを保証するために論理反転値が繰り返され、合計8ビットのPIDになります。この結果の形態は以下で示されます。

図11. PID領域形式



アドレス (ADDR)

アドレス領域はパケットがどの装置に対する指示かを示します。7ビット長で127装置の支援を可能にします。アドレス0はアドレスがまだ割り当てられていない装置がこのアドレスへ送られるパケットに応答しなければならないため、装置へのアドレス0割り当ては無効です。本形式は以降のエンドポイント、CRCと共に併せて、図12.に示されます。

エンドポイント (ENDP)

エンドポイント領域は4ビットで構成され、16個のエンドポイントの使用を可能にします。Low-Speed装置では先頭の2つ(既定、最大4つ)の追加エンドポイントだけを持てます(エンドポイント0は既定かつ必須のため、これを除いての意)。本形式は次の図12.で示されます。

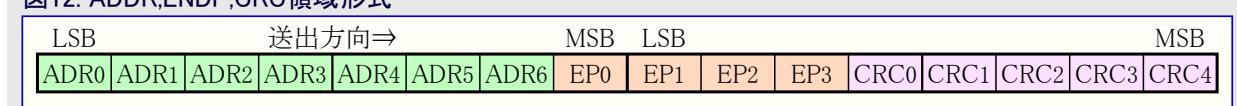
巡回冗長検査 (CRC)

巡回冗長検査はパケット内のデータ部分の検査を行います。全ての指示票(トークン)パケットは5ビットのCRCを、他方データパケットは16ビットのCRCを持ちます。

表3. PID種別一覧

目的別	PID値	パケット種別
指示票 (トークン)	0001	出力(OUT)指示
	1001	入力(IN)指示
	0101	フレーム開始(SOF)指示
	1101	設定(SEUP)指示
データ	0011	データ0(DATA0)
	1011	データ1(DATA1)
	0111	データ2(DATA2)
	1111	複データ(MDATA)
ハンド シェーク	0010	肯定応答(ACK)
	1010	否定応答(NAK)
	1110	不能応答(STALL)
	0110	未応答(NYET)
特殊	1100	前置部(PRE)指示票で使用
	1100	誤り(ERR)応答で使用
	1000	分割(SPLIT)指示票で使用
	0100	確認(Ping)指示票で使用
	0000	予約

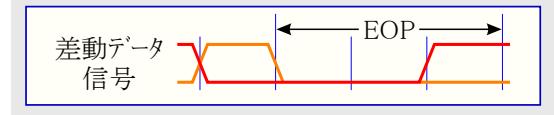
図12. ADDR,ENDP,CRC領域形式



ハケット終了(EOP)

ハケットの終了を示します。これは概ね2ビット時間のシングルエンド0(SE0)と後続する1ビット時間のJ状態で表されます。

図13. EOP信号



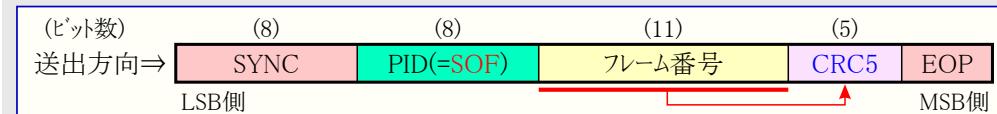
処理単位(Transaction)とUSBハケット種別

USBには4種類のハケット種別があります。フレーム開始ハケットは新規フレームの開始を示します。指示票(トークン)ハケットは後続するやり取りの種別を示し、データハケットはそのやり取りに必要なデータを含み、ハンドシェークハケットはそのデータに対する応答や異常の報告に使われます。一般的にUSBでは指示票(トークン)ハケット、データハケット、ハンドシェークハケットの組で或る意味を持つ通信を行い、この通信を処理単位(Transaction)と呼びます。

フレーム開始(SOF)ハケット

SOFハケットはホストによって $1\pm0.5\text{ms}$ 毎に送信される11ビット長のフレーム番号を含むハケットです。

図14. SOFハケット形式



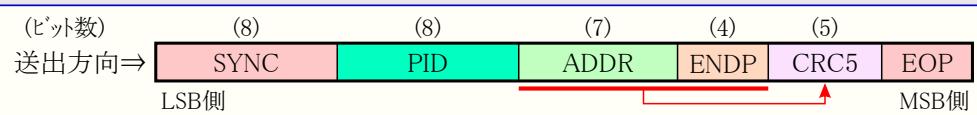
指示票(トークン)ハケット

指示票(トークン)ハケットにはハケット識別(PID)の内容により、次の3種類があります。

- 設定(SEUP) …… 必須の制御(Control)転送を開始するために使われます。
- 入力(IN) …… ホストが情報を送出したいことをUSB装置へ通知します。
- 出力(OUT) …… ホストが情報を取得したいことをUSB装置へ通知します。

指示票(トークン)ハケットは次の形式に従わなければなりません。

図15. 指示票(トークン)ハケット形式



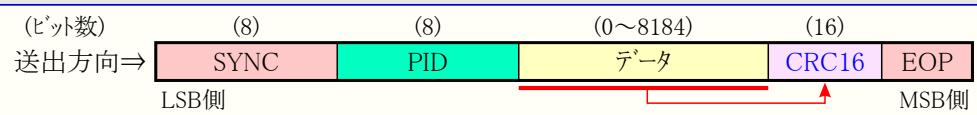
データハケット

データハケットには通常、PIDの内容により、次の2種類があります。

- DATA0
- DATA1

これらは複数データハケット時、交互に使われて欠落監視に使われます。ハケット内のデータ部はバイト単位で0～1023バイトの範囲ですが、Low-Speed装置にあっては8バイト以内でなければなりません。データハケットは次の形式です。

図16. データハケット形式



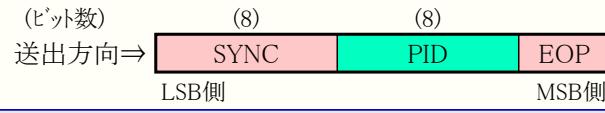
ハンドシェークハケット

ハンドシェークハケットは単にPIDから成るハケットで、次の3種類があります。

- 肯定応答(ACK) …… ハケットが成功裏に受信されたことを示す応答です。
- 否定応答(NAK) …… 装置の一時的なデータ送受信不能を報告します。割り込み(Interrupt)転送中に送信データがないことをホストに通知するのにも使われます。
- 不能応答(STALL) …… 装置が判断不能でホストの介在を必要とする状態であることを通知します。

ハンドシェークハケットは次の形式です。

図17. ハンドシェークハケット形式



転送種別

USB仕様では次の4つの転送種別が定義されています。

- 制御(Control)転送
- 割り込み(Interrupt)転送
- 等時(Isochronous)転送
- 大量(Bulk)転送

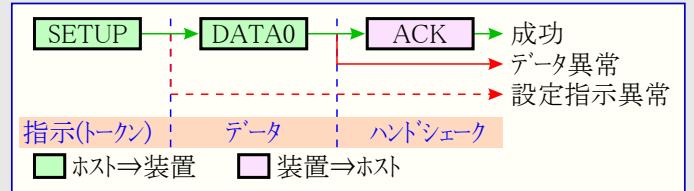
制御(Control)転送

制御転送は通常、命令と状態の操作用に使われます。これらは制御転送を使って行われる全機能の**列挙**でUSB装置を初期設定するのに必須です。これらは**最善効率配給**を用いてホストによって開始される集合的または離散的なパケット群です。Low-Speed装置の制御転送パケットのデータ長は8バイト、Full-Speed装置は64バイトでなければならず、High-Speed装置は8, 16, 32, 64バイト長が許されます。制御転送は3つの段階(Stage)からなります。

設定(Setup)段階

設定段階は要求が送信される段階です。これは3つのパケットからなります。アドレスとエンドポイント番号を含む**設定(Setup)指示票(トークン)**パケットが最初に送信されます。次に**設定(Setup)パケット**の要求種別詳細を含み、PID種別が常に**DATA0**のデータパケットが送信されます。最後のパケットは受信成功の応答または異常を示すに使われる**ハンドシェーク**パケットです。設定(Setup)データが正しく(PIDや**CRC検査**が正常に)受信されたなら**ACK**で応答し、さもなくばそのデータを無視し、ハンドシェークパケットを送信してはいけません。設定(Setup)パケットの応答では**NAK**や**STALL**のハンドシェークパケットを使えません。

図18. 設定(Setup)段階の流れ



データ(Data)段階

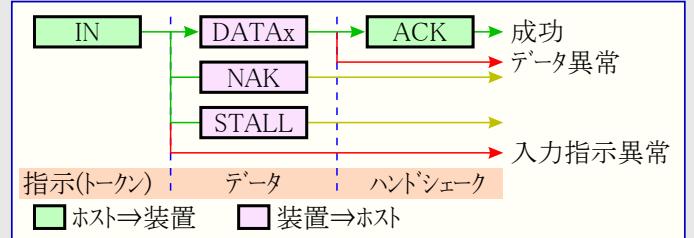
任意のデータ段階は1つ以上の入力(IN)または出力(OUT)パケット転送からなります。設定(Setup)要求は本段階で送信されるべきデータ量を指示します。これが1パケットの最大データ容量を越える場合、送信すべきデータは最終パケットを除いて各々のパケットが最大データ長の複数パケットで送信されます。

データ段階はデータの転送方向に依存した2つの異なる手順があります。

入力(IN)

ホストは制御データを受信する準備が整うと入力(IN)パケットを送信します。装置は誤り(例えばPIDビットと反転PIDビットの不一致)と共に入力(IN)指示票(トークン)を受信した場合、そのパケットを無視します。その指示票(トークン)が正しく受信されると、装置は送るべき制御データを含むデータ(DATAx)パケットか、またはエンドポイントの異常を示す不能(STALL)パケット、またはエンドポイントが作業中で一時的に送るデータがないことをホストへ示す否定応答(NAK)パケットの何れかを返すことができます。

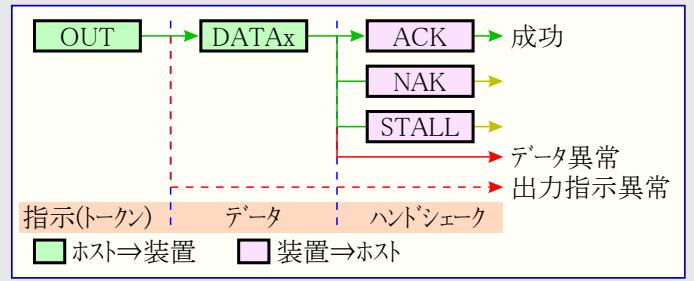
図19. データ入力(IN)段階の流れ



出力(OUT)

ホストが装置へ制御データパケットを送る必要があるとき、ホストは出力(OUT)指示票(トークン)パケットに続きデータ部として制御データを含むデータパケットを送信します。出力(OUT)指示票(トークン)パケットまたはデータパケットの何れかの部分に誤りがあれば、装置はそのパケットを無視します。装置のエンドポイント緩衝部が空で制御データをその緩衝部へ格納したなら、データの受信が成功したことをホストへ通知するために肯定応答(ACK)パケットを送信します。直前のパケットの処理中のためにエンドポイント緩衝部が空でない場合、装置は否定応答(NAK)パケットを返します。但しエンドポイントが何らかの異常で停止(または中止)状態の場合不能応答(STALL)パケットを返します。

図20. データ出力(OUT)段階の流れ



状態(Status)段階

状態段階はデータ段階全体に対する状態を報告しますが、これは転送方向によって形態が異なり、入出力(IN/OUT)指示票(トークン)がデータ段階と逆になります。この状態報告は常に装置側によって行われます。従つて入力(IN)指示で非正常終了を報告する場合、本来データパケットを返す位置で対応するハンドシェークパケットによって応答することに注意してください。

入力(IN)

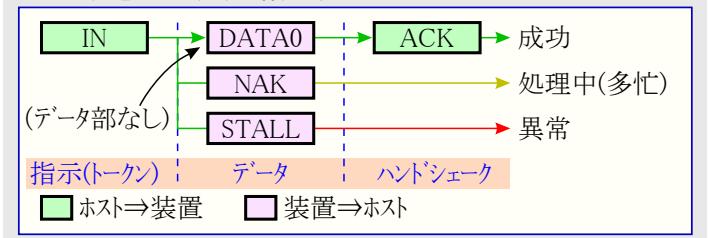
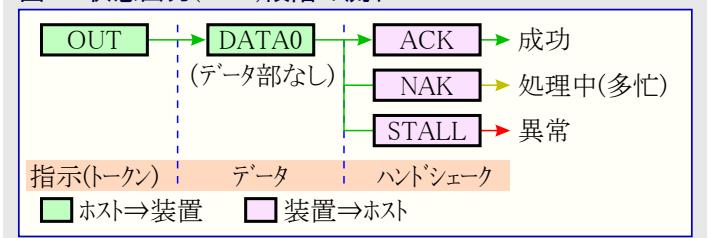
データ段階=出力

ホストがデータ段階中に出力(OUTパケット)を送信してデータを送信したなら、装置は入力(IN)指示票(トークン)パケットに応じてデータ部のない(データ長=0)データパケットを送信することによって、そのデータの受信成功で応答するでしょう。然しながら、異常が起きた場合は不能応答(STALL)パケットで、未だデータ処理中の場合は以降の状態段階での再試行をホストに頼む否定応答(NAK)パケットで応答すべきです。

出力(OUT)

データ段階=入力

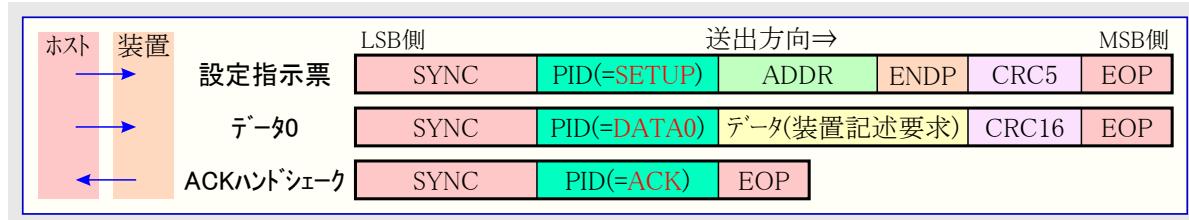
ホストはデータ段階中に入力(IN)パケットを送信してデータを受信したなら、そのデータの受信成功で応答しなければなりません。これは出力(OUT)指示票(トークン)パケットとそれに続くデータ部のない(データ長=0)データパケットをホストが送信することで行われます。装置はハンドシェークパケットで直ぐに状態を報告することができます。肯定応答(ACK)は装置が命令を完了し、直ぐに他の命令の受付準備ができる事を示します。この命令の処理中に異常が起きた場合、装置は不能応答(STALL)で応答します。未だ処理中の場合は以降の状態段階での再試行をホストに頼む否定応答(NAK)で応答します。

図21. 状態入力(IN)段階の流れ**図22. 状態出力(OUT)段階の流れ**

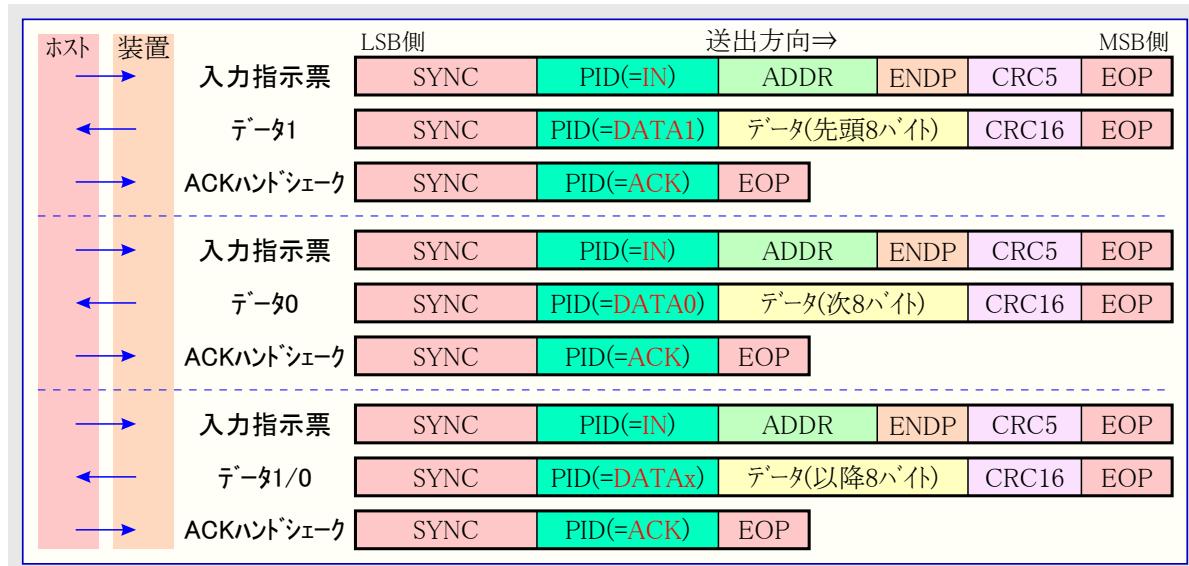
制御(Control)転送に於ける大量データ

例えばホストから列挙中に装置記述要求があり、その内容が1つのデータパケットに収まらない場合、それらのデータは次の手順で送信されます。

ホストは後続のパケットが装置記述要求を含む設定(SEUP)パケットであることを装置に通知する設定(SEUP)指示票(トークン)パケットを送信します。このパケットのアドレス(ADDR)領域には装置記述を要求する装置のアドレスが保持されます。エンドポイント番号は既定パイプを指示するため、0であるべきです。その後ホストはデータ(DATA0)パケットを送信します。このパケットはデータ部に装置記述要求を示す8バイトの値を持ちます。装置は異常なしで設定(SEUP)パケットが正しく読み込まれたことを応答で返します。そのパケットが正しく読み込まれなかった場合、そのパケットを無視します。その後、ホストは一定時間経過後にそのパケットを再送信するでしょう。

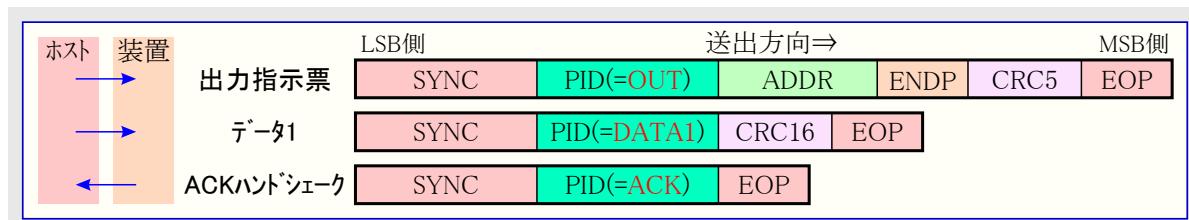


上の3つのパケットは最初のUSB処理単位(Transaction)を表します。装置は受信した8バイトを調べ、それが装置記述要求であることを知ります。そして装置は次からのUSB処理単位(Transaction)で装置記述の送信を試みます。



上記のデータパケットのデータ部最大容量は8バイトと仮定しています。ホストはエンドポイントからのデータを直ぐに送信できることを装置に告げるために入力(IN)指示票(トークン)を送信します。8バイトを超えるデータは最終データを含む処理単位(Transaction)を除いて、入力(IN)指示票(トークン)に対応して8バイト毎に送信されなければなりません。ホストはこの送信データ毎に応答します。

全データが送信されてしまうと、状態段階の処理単位(Transaction)が続きます。全転送が成功の場合、ホストはこれを示すデータ部なし(データ長=0)のデータパケットを送信します。そして装置はこのデータパケットに対してハンドシェークパケットで返答します。



割り込み(Interrupt)転送

割り込み転送の原則的な考え方方はマイクロ コントローラなどの割り込みと同様で、USB装置側が割り込みを生成します。然しながら、USB下ではホストに注意を促したくても、その前にホストによる巡回問い合わせ(ポーリング)まで待たなければなりません。割り込み転送には他の転送と比較して次の特徴があります。

- 保証された遅延
- 単方向のストリーム パイプ
- 異常検出と次周期での再試行

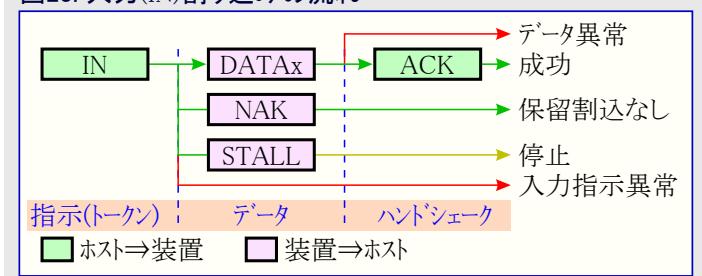
割り込み転送は、或る時点から一定の時間内に通信を開始しなければならない一般的に非周期性のもので使われます。割り込み要求はホストがデータについて装置に問い合わせるまで装置によって順に記録保存されます。

割り込み転送のパケット内データ部最大容量はLow-Speed装置が8バイト、Full-Speed装置が64バイト、High-Speed装置が1024バイトです。割り込み転送には割り込み入力(IN)と割り込み出力(OUT)の2種類の処理単位(Transaction)があります。

入力(IN) ホストは周期的に割り込みエンドポイントを調べます。この周期の間隔は後述のエンドポイント記述で指定されます。各周期での調査はホストが入力(IN)指示票(トークン)パケットを送信することで始まります。この入力(IN)指示票(トークン)が誤りの場合、装置はこのパケットを無視し、バス上で新規パケットの監視を継続します。

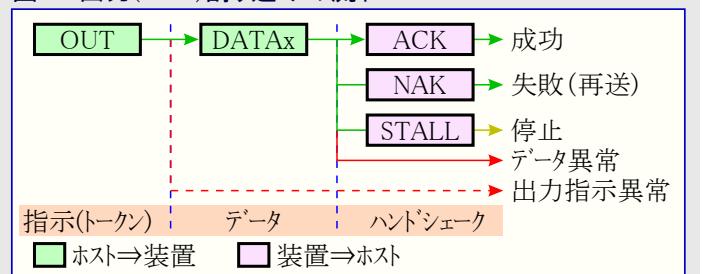
割り込みが装置内に記録保存されている場合、入力(IN)指示票(トークン)を受信した時にその割り込みに関連したデータを含むデータパケットを送信します。ホストでの受信が成功するとホストは肯定応答(ACK)を返します。データが不正な場合、ホストは情報を返しません。ホストが入力(IN)指示票(トークン)で割り込みエンドポイントを調査する時に割り込み状態が存在しない場合、装置は否定応答(NAK)を送信することで、この状態を示します。割り込みエンドポイントで異常が発生している場合は、入力(IN)指示票(トークン)の応答で代わりに不能応答(STALL)を送信します。

図23. 入力(IN)割り込みの流れ



出力(OUT) ホストが装置への割り込みデータ送信を欲するとき、出力(OUT)指示票(トークン)パケットと一緒に続けて割り込みデータを含むデータパケットを送信します。出力(OUT)指示票(トークン)パケットまたはデータパケットの何れかの部分が不正なら、装置はそのパケットを無視します。装置のエンドポイント緩衝部が空で、データがその緩衝部内に格納されたなら、そのデータ受信が成功したことをホストに通知する肯定応答(ACK)を送信します。そのエンドポイント緩衝部が直前パケットの処理中のために空でなかった場合、装置は否定応答(NAK)を返します。但しエンドポイントが何らかの異常で停止(または中止)状態の場合不能応答(STALL)パケットを返します。

図24. 出力(OUT)割り込みの流れ



等時(Isochronous)転送

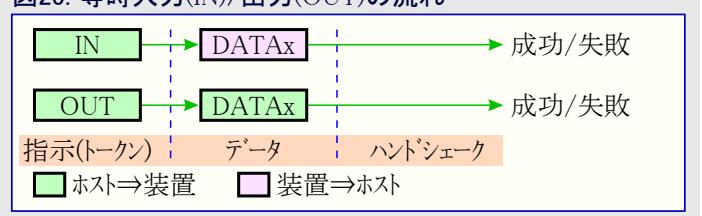
等時転送は周期的且つ継続的に起きます。通常、これらはオーディオやビデオのように、時間に敏感な情報を含みます。これらでは誤りによる部分欠落が時間的に瞬間のため、多くの場合、気付かれないのでしょう。再送の繰り返しによる時間軸の乱れの方がより深刻です。等時転送は他の転送と比較して次の特徴があります。

- USB帯域幅内での保証されたアクセス
- 単方向のストリーム パイプ
- Full/High-Speedでのみ可能
- 一定時間内の遅延
- CRCでの異常検出、再試行なし
- PIDのDATA1/0交互使用なし

データパケットのデータ部最大容量は等時エンドポイントのエンドポイント記述内で指定され、Full-Speedでは最大1023バイトまで、High-Speedでは1024バイトまでです。この最大容量はバスの必要条件である帯域幅に影響するため、控えめな値を指定します。大容量指定は様々な容量指定での一連の代替インターフェースで優位になるかもしれません。列挙中にホストが帯域幅制限のために希望する等時エンドポイントを許可できない場合、単に失敗で終わらせず、縮小調停(フォールバック)が行われます。等時エンドポイントで送られるデータは予め調停された容量より少なくでき、処理単位(Transaction)間で長さを変えることができます。

右図は等時入力(IN)/出力(OUT)転送の処理単位(Transaction)形式を示します。等時転送にはハンドシェーク段がなく、異常や停止状態の報告ができません。

図25. 等時入力(IN)/出力(OUT)の流れ



大量(Bulk)転送

大量転送は突発的大量データに使われます。これには印刷機やスキャナのデータ転送などが該当します。大量転送ではデータ部に対するCRC16領域の形式での異常訂正と、異常なしでデータが送受信されることを保証する異常検出/再送機構が提供されます。

大量転送は他の全ての処理単位(Transaction)が割り当てられてしまった後の余った未割り当て帯域幅を使います。割り込み(Interrupt)転送や等時(Isochronous)転送でバスが忙しい場合、大量データはバス上で少しずつゆっくりと転送されるかもしれません。この結果、遅れが保証されないため、大量転送は時間に敏感でない通信に対してだけ使われるべきです。大量転送には他の転送と比較して次の特徴があります。

- 突発的大量データ転送に使用
- 単方向のストリームパイプ
- Full/High-Speedでのみ可能
- 帯域幅や最小遅延の保証なし
- CRCでの異常検出と配給保証

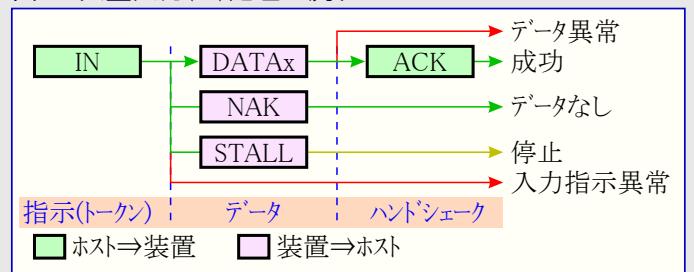
大量転送はFull-Speed装置とHigh-Speed装置でだけ支援されます。Full-Speedに於けるデータ部最大容量は8,16,32,64バイトの何れかで、High-Speedでは512バイトまでの長さにできます。データがこの最大容量未満の場合、残り部分を0で埋める必要はありません。大量転送は、正確な要求データ量のパケット転送、最大容量未満のパケット転送、データなし(データ長=0)で完了すると考えられます。

大量転送には大量入力(IN)と大量出力(OUT)の2種類の処理単位(Transaction)があります。

入力(IN)

ホストは大量データの受信準備が整うと入力(IN)指示票(トークン)を送信します。装置が異常と共にこの入力(IN)指示票(トークン)を受信した場合、そのパケットを無視します。その入力(IN)指示票(トークン)が正しく受信されると、装置は送るべき大量データを含むデータ(DATAx)パケットか、またはエンドポイントの異常を示す不能(STALL)パケット、またはエンドポイントが作業中で一時的に送るデータがないことをホストへ示す否定応答(NAK)パケットの何れかを返すことができます。

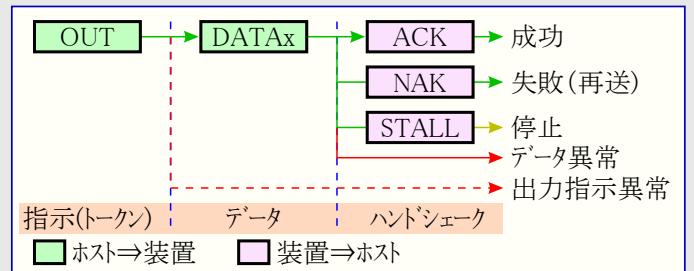
図26. 大量入力(IN)処理の流れ



出力(OUT)

ホストが装置への大量データ送信を欲するとき、出力(OUT)指示票(トークン)パケットと一緒に続けて大量データを含むデータパケットを送信します。出力(OUT)指示票(トークン)パケットまたはデータパケットの何れかの部分が不正なら、装置はそのパケットを無視します。装置のエンドポイント緩衝部が空で、データがその緩衝部内に格納されたなら、そのデータ受信が成功したことをホストに通知する肯定応答(ACK)を送信します。そのエンドポイント緩衝部が直前パケットの処理中のために空でなかった場合、装置は否定応答(NAK)を返します。但しエンドポイントが何らかの異常で停止(または中止)状態の場合不能応答(STALL)パケットを返します。

図27. 大量出力(OUT)処理の流れ



USB記述子(Descriptor)

全USB装置はどんな装置で、誰が作り、どのUSB版を支援し、何種類の方法で設定でき、エンドポイントの数とその種別は、などの情報をホストに告げるため記述子の階層を持ちます。多くで共通するUSB記述子には次のものがあります。

- 装置記述 (Device Descriptor)
- 構成記述 (Configuration Descriptor)
- インターフェース記述 (Interface Descriptor)
- エンドポイント記述 (Endpoint Descriptor)
- 文字列記述 (Strings Descriptor)

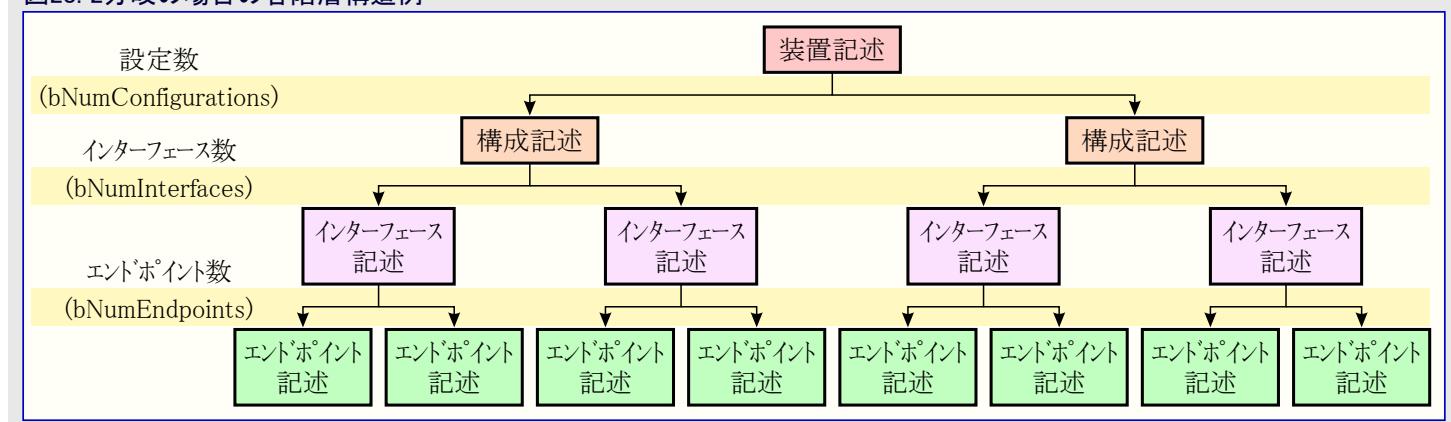
USB装置は1つの装置記述しか持つことができません。装置記述は装置が適合するUSB版、適切なドライバを読み込むのに使われる供給者識別(VID)と製品識別(PID)、装置が持っている利用可能な構成数のような情報を含みます。この構成数は構成記述が何個かに別れて続くことを示します。

構成記述は、この特定構成で使う電力量、装置がバス給電か自己給電か、この構成が持つインターフェース数のような値を指示します。装置が列挙されるとき、ホストは装置記述を読み、どの構成を許可すべきかを決定できます。これは一度に1つの構成だけを許可できます。

例えば、高電力バス給電構成と自己給電構成を持つことが可能な場合に、装置が主電力供給元と共にホストへ接続されると、デバイスドライバは主電源なしで給電される装置を許可する高電力バス給電の許可を選ぶかもしれません。けれどもノートPCに接続された場合、主電源に装置を接続する必要がある第2の構成(自己給電)を許可できます。

構成の指定は電力の違いに制限される訳ではありません。各構成は同じ方法、同じ電流での給電が有り得ますが、まだ下層に異なるインターフェースまたはエンドポイントの組み合わせを持ちます。けれども構成変更が各エンドポイントの全動作の停止を要求することに注目すべきです。USBがこの柔軟性を提供する一方、複数の構成を持つ実際の装置は非常に僅かです。

図28. 2分岐の場合の各階層構造例



インターフェース記述は装置の単一機能を実行する機能動作群内のエンドポイント群またはヘッダと見ることができます。例えば多機能印刷機/ファックス/スキャナの場合、最初のインターフェース記述は印刷機能のエンドポイントを、2つ目のインターフェース記述はファックス機能を、3つ目はスキャナ機能を記述できます。構成記述と異なり、これらは一度に1つしか許可できないと言う制限はありません。装置は一度に許可された複数のインターフェース記述を持てます。

インターフェース記述はインターフェース番号を指示する**bInterfaceNumber**領域と動作中の設定変更をインターフェースに許す**bAlternativeSetting**領域を持ちます。例えば装置がインターフェース0とインターフェース1を持っている場合、インターフェース0は最初のインターフェース(記述)を表す0に設定された**bInterfaceNumber**と(既定を表す)0の**bAlternativeSetting**で示されます。

インターフェース1は2番目のインターフェースを表す1に設定された**bInterfaceNumber**と(既定を表す)0の**bAlternativeSetting**で示されます。このインターフェース1に別の代替設定を付加する場合は、インターフェース1を示す1に設定された**bInterfaceNumber**と、今回は代替設定であることを表す1を**bAlternativeSetting**に設定します。

これらの設定が許可されると**bAlternativeSetting**が既定を表す0である最初の2つのインターフェース記述が使われます。然しながら、操作中にホストは別のインターフェース記述を許可するため、代替設定を持つ(=1)インターフェース1へ直接インターフェース設定(SET_INTERFACE)要求を送信できます。

これはインターフェース0でデータを送信しながら、一方でインターフェース0への影響なしにインターフェース1に関連したエンドポイント設定を変更できることで、2つの設定を持つ以上の優位性を与えます。

各エンドポイント記述は各々のエンドポイントに対する転送種別、方向、巡回検査(ポーリング)間隔及び最大データ部容量を指定するのに使われます。既定の制御(Control)用エンドポイント(エンドポイント0)は常に制御エンドポイントであると仮定され、このようなものは記述を持ちません。

USB記述子(Descriptor)の構成

全ての記述子は共通形式で作成されます。最初のバイトは記述子の長さ(バイト数)を示し、第2バイトは記述子種別を表します。記述子の長さが仕様での定義より短い場合、ホストはこれを無視します。けれどもその長さが予測より大きい場合、ホストは余分なバイトを無視して、実際に返された長さの終了後に次の記述子の検索を始めます。

表4. 記述子先頭の共通部分

変位	領域名	バイト数	値種別	意味
0	bLength	1	数値	記述子全体のバイト数
1	bDescriptorType	1	定数	記述子種別を表す数値
2	～	～	～	以降記述子内容(種別によって変化)

装置記述子(Device Descriptor)

USB装置の装置記述は装置全体を表します。USB装置は1つの装置記述しか持てないので、いくつかの基本的な事しか示しませんが、それらは支援するUSB版、パケットのデータ部最大容量、供給者識別(VID)と製品識別(PID)、装置で利用可能な設定数のように装置についての重要な情報です。

表5. 装置記述(Device Descriptor)の構成

変位	領域名	バイト数	値種別	意味
0	bLength	1	数値	本記述全体のバイト数
1	bDescriptorType	1	定数	装置記述を表す\$01
2	bcdUSB	2	BCD	装置が適合するUSB版(例: USB2.0の場合\$0200)
4	bDeviceClass	1	クラス符号	本値が0の場合は各インターフェースが自身のクラス符号を指定します。 \$FFの場合、クラス符号は供給者によって指定されます。他の値は有効なクラス符号です。
5	bDeviceSubClass	1	補助クラス符号	USB協会によって割り当てられた補助クラス符号
6	bDeviceProtocol	1	規約符号	USB協会によって割り当てられた規約符号
7	bMaxPacketSize	1	数値	エンドポイント0の最大データ容量(有効値は8,16,32,64)
8	idVendor	2	識別符号	USB協会によって割り当てられた供給者識別符号
10	idProduct	2	識別符号	製造(供給)者によって割り当てられた製品識別符号
12	bcdDevice	2	BCD	装置版番号
14	iManufacturer	1	指標番号	製造(供給)者文字列記述の指標番号
15	iProduct	1	指標番号	製品文字列記述の指標番号
16	iSerialNumber	1	指標番号	製品通番文字列記述の指標番号
17	bNumConfigurations	1	数値	使用可能な構成数

- **bcdUSB**は装置が支援する最上位のUSB版を報告します。この値は\$VVMS形式の2進化10進値で、VVが主板、Mが補助版、Sが補足版番号です。例えばUSB1.0は\$0100、USB1.1は\$0110、USB2.0は\$0200として報告されます。
- **bDeviceClass**,**bDeviceSubClass**,**bDeviceProtocol**は装置に対応するクラスドライバを探すためにオペレーティングシステムによって使われます。多くのクラス仕様がインターフェース段階でのクラス確認を選ぶため、通常は**bDeviceClass**だけが装置段階で\$00として設定されます。これは複数クラスの支援を1つの装置に許します。
- **bMaxPacketSize**はエンドポイント0の最大データ容量を報告します。全ての装置はエンドポイント0を支援しなければなりません。
- **idVendor**,**idProduct**は装置に対応するデバイスドライバを探すためにオペレーティングシステムによって使われます。供給者識別符号はUSB協会によって割り当てられます。
- **bcdDevice**は**bcdUSB**と同じ形式で、装置の版番号を提供するのに使われます。この値は開発業者によって割り当てられます。
- **iManufacturer**,**iProduct**,**iSerialNumber**は各々製造業者、製品、通番の詳細を提供するために存在します。これらは必ずしも対応する文字列記述子を持つ必要はなく、その場合は値として0が使われるべきです。
- **bNumConfigurations**は装置が現在の速度で支援する構成数を定義します。

構成記述子(Configuration Descriptor)

多くの装置は単純で1つの構成しか持ちませんが、USB装置はいくつかの異なる構成を持つことができます。この構成記述は装置がどう給電され、どれくらいの最大消費電力で、いくつのインターフェース数を持つかを指定します。このため装置がバス給電される場合と外部電力の場合で2つの構成を持つことが可能です。これはインターフェース記述へのヘッダであるため、異なる転送種別を使う別の構成を持つこともできます。

一旦ホストによって全ての設定が調べられてしまうと、ホストは構成記述の1つのbConfigurationValueと一致した0以外の値で構成選択(SET_CONFIGURATION)命令を送信します。これは希望する構成の選択に使われます。

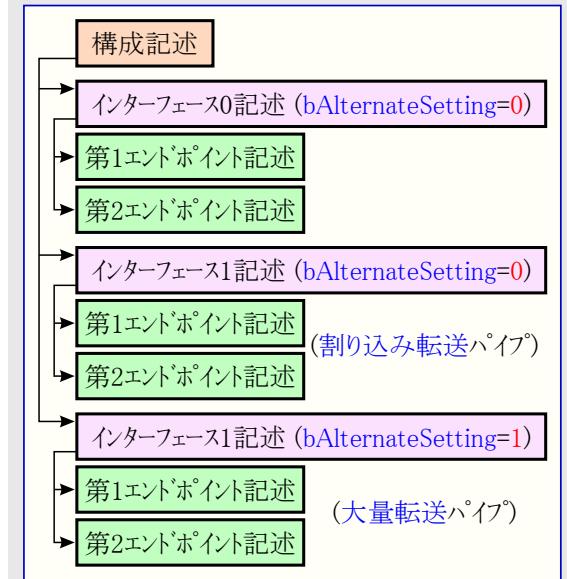
表6. 構成記述(Configuration Descriptor)の構成

変位	領域名	バイト数	値種別	意味	
0	bLength	1	数値	本記述全体のバイト数	
1	bDescriptorType	1	定数	構成記述を表す\$02	
2	wTotalLength	2	数値	返されるデータの総バイト数	
4	bNumInterfaces	1	数値	インターフェース数	
5	bConfigurationValue	1	数値	この構成選択の引数として使われる値	
6	iConfiguration	1	指標番号	本設定を記述する文字列記述の指標番号	
7	bmAttributes	1	ビット値	ビット7	バス給電
				ビット6	自己給電
				ビット5	遠隔起動
				ビット4~0	予約(=0)
8	bMaxPower	1	数値	2mA単位の最大消費電流値	

構成記述が読まれるとき、その構成階層内のインターフェースとエンドポイントに関連する全記述を含む階層全体(右図参照)を返します。wTotalLengthはその階層内のバイト数を反映します。

- bNumInterfacesはこの構成に存在するインターフェース数を示します。
- bConfigurationValueはこの構成を選ぶ構成選択(SET_CONFIGURATION)要求で使われます。
- iConfigurationは人間が読める形で記載する文字列記述の指標番号です。
- bmAttributesはこの構成についての電源種別値を指定します。装置が自己給電なら、ビット6を設定(=1)します。ビット7はバス給電装置であることを示すためにUSB1.1で使われますが、現在ではこれがbMaxPowerで行われます。装置が何らかの電力をバスから使う場合、自己給電またはバス給電のどちらかとして、bMaxPowerで消費電力を報告しなければなりません。
- bMaxPowerは装置がバスから得る最大電力を定義します。これは2mA単位でを行い、従って最大約500mAが指定できます。この仕様はバス(VBUS)から500mAを越えない範囲で電流を流すことを高電力バス給電装置に許します。装置が外部電力を失った場合であっても、bMaxPowerでの指定より多くの電流を流してはなりません。

図29. 1設定全体階層構成例



インターフェース記述子(Interface Descriptor)

インターフェース記述は装置の单一機能を実行する起動動作群内のエンドポイント群またはヘッダと見ることができます。インターフェース記述は次の形式に従います。

表7. インターフェース記述(Interface Descriptor)の構成

変位	領域名	バイト数	値種別	意味
0	bLength	1	数値	本記述全体のバイト数
1	bDescriptorType	1	定数	インターフェース記述を表す\$04
2	bInterfaceNumber	1	数値	インターフェース番号
3	bAlternateSetting	1	数値	代替設定の選択で使われる代替番号値
4	bNumEndpoints	1	数値	このインターフェースで使われるエンドポイント数
5	bInterfaceClass	1	クラス符号	USB協会により割り当てられたクラス符号
6	bInterfaceSubClass	1	補助クラス符号	USB協会により割り当てられた補助クラス符号
7	bInterfaceProtocol	1	規約符号	規約符号
8	iInterface	1	指標番号	このインターフェースを記載する文字列記述の指標番号

- bInterfaceNumberはこのインターフェースの番号を示します。
- bAlternativeSettingは代替インターフェースの指定に使えます。これらの代替インターフェースはインターフェース選択(SET_INTERFACE)要求で選択することができます。
- bNumEndpointsはこのインターフェースで使われるエンドポイント数を示します。この値はエンドポイント0を除外すべきで、このインターフェースに連なるエンドポイント記述の数を示すために使われます。
- bInterfaceClass,bInterfaceSubClass,bInterfaceProtocolは支援するクラス(例えば、HID,通信(communications),大容量記憶装置(mass storage)など)の指定に使えます。これはその装置に対する特定ドライバの作成なしで、クラスドライバの使用を多くの装置に許します。
- iInterfaceはこのインターフェースの文字列記述を許します。

エンドポイント記述子(Endpoint Descriptor)

エンドポイント記述はエンドポイント0以外のエンドポイント記述に使われます。エンドポイント0は常に制御(Contorol)エンドポイントとの認識で、どんな記述が要求されるよりも前に(そのように)設定されます。

表8. エンドポイント記述(Endpoint Descriptor)の構成

変位	領域名	バイト数	値種別	意味	
0	bLength	1	数値	本記述全体のバイト数	
1	bDescriptorType	1	定数	エンドポイント記述を表す\$05	
2	bEndpointAddress	1	ビット分割値	ビット7	方向 0=出力, 1=入力 (制御エンドポイント時無効)
				ビット6~4	予約(=0)
				ビット3~0	エンドポイント番号
3	bmAttributes	1	ビット分割値	ビット7,6	予約(=0)
				11	予約
				10	明示逆送データ エンドポイント
				01	暗黙逆送データ エンドポイント
				00	通常データ エンドポイント
				11	同期
				10	適応
				01	非同期
				00	同期なし
				11	割り込み(Interrupt)
4	wMaxPacketSize	2	数値	このエンドポイントの最大データ容量(バイト数)	
	bInterval	1	数値	フレーム単位のエンドポイント調査間隔(制御と大量転送時は無効) 等時転送時は常に1、割り込み転送時は1~255	
				等時転送以外時は予約(=0)	

- **bEndpointAddress**はこの記述でどんなエンドポイントが記述されるかを示します。
- **bmAttributes**は転送種別を指定します。これは制御(Contorol)、割り込み(Interrupt)、等時(Isochronous)、大量(Bulk)の何れかにできます。等時エンドポイントが指定されたなら、同期種別や使用種別のような付加属性が選択できます。
- **wMaxPacketSize**はこのエンドポイントの最大データバイト数を示します。
- **bInterval**は転送を行うための調査間隔を指定します。この単位はフレーム/マイクロフレームで表され、従ってLow/Full-Speed装置では1ms、High-Speed装置では125μsのどちらかと等しくなります。

文字列記述子(String Descriptor)

文字列記述は任意使用で、人間が読める情報を提供します。これらが使われない場合、どの記述の文字列指標番号も文字列記述を利用できないことを示す0に設定されなければなりません。

文字列はユニコード形式で符号化され、複数言語の支援ができます。文字列指標番号0は支援する言語一覧を返すのに使われます。USB言語識別(ID)の一覧は[付録](#)を参照してください。

表9. 文字列指標番号0の文字列記述構造

変位	領域名	バイト数	値種別	意味
0	bLength	1	数値	本記述全体のバイト数
1	bDescriptorType	1	定数	文字列記述を表す\$03
2	wLANGID(0)	2	識別符号	支援する言語番号0の言語識別(ID) (例: \$0409=米英語)
4	wLANGID(1)	2	識別符号	支援する言語番号1の言語識別(ID) (例: \$0411=日本語)
(n)	(n)	(n)	(n)	(n)
2n+2	wLANGID(n)	2	識別符号	支援する言語番号nの言語識別(ID)

上の文字列記述は文字列記述0の形式を示します。ホストはどの言語が利用可能かを調べるために、この記述を読むべきです。希望する言語が支援されていれば、文字列記述を得る記述取得(GET_DESCRIPTOR)要求のwIndex領域で言語識別(ID)を送ることによってそれを参照できます。

それに続く文字列は以下の形式で運ばれます。

表10. 文字列記述構造

変位	領域名	バイト数	値種別	意味
0	bLength	1	数値	本記述全体のバイト数
1	bDescriptorType	1	定数	文字列記述を表す\$03
2	bString	n	ユニコード	ユニコードで符号化された文字列

設定(Setup)パケット

全ての装置は既定パイプ^①上の設定(Setup)パケットに応答しなければなりません。この設定(Setup)パケットは装置の検出や設定に使われ、USB装置のアドレス設定、[装置記述](#)の要求、[エンドポイント](#)の状態検査のような共通機能を実行します。

USBに適合したホストは処理されるべき全ての要求について最大5秒間待ちます。また特定の要求に対しては厳密なタイミングも指定されています。

- データ段階を除く[標準装置要求\(Standard Device Request\)](#)は50ms以内に完了しなければなりません。
- 標準装置要求(Standard Device Request)のデータ段階はその要求後または直前の送出パケット完了後から500ms以内にデータパケットの送信を開始しなければなりません。
- 標準装置要求(Standard Device Request)の[状態\(Status\)段階](#)は最後のデータパケット送信完了後、50ms以内に完了されなければなりません。
- [アドレス設定\(SET_ADDRESS\)命令](#)は状態応答までを50ms以内に処理しなければなりません。装置はその後、次の命令が送られる前にアドレスを変更するために2msの余裕が与えられます。

これらの制限時間は低速装置であっても充分に受け入れ可能ですが、デバッグが制限され得ます。この50ms制限時間は、内部レジスタを調べるためのICEによる一時停止や1命令実行、非同期シリアルポートの9600bpsでのデータ送信のような多くのデバッグについて問題になります。従ってUSBのデバッグでは一般的なマイクロコントローラでのデバッグと異なる何らかのデバッグ方法が必要となります。

各要求は以下の形式を持つ8バイト長の設定(Setup)パケットで始まります。

表11. 設定(Setup)パケット構成

変位	領域名	バイト数	値種別	意味				
0	bmRequestType	1	ビット割数値	要求の特性種別				
				ビット7 (データ転送方向)	0	ホスト⇒装置		
				1	装置⇒ホスト			
				ビット6,5 (種別)	0	標準(Standard)		
				1	クラス(Class)			
				2	供給者(Vendor)			
				3	予約			
				ビット4～0 (受け取り部)	0	装置(Device)		
				1	インターフェース(Interface)			
				2	エンドポイント(Endpoint)			
				3	その他			
				4～31	予約			
				1	bRequest	1	値	要求指定
				2	wValue	2	値	要求に従って変化するワード領域
				4	wIndex	2	指標または相対位置	要求に従って変化するワード領域 (通常、指標または相対位置の進行に使用)
6	wLength	2	数値	データ状態の場合、転送バイト数				

[bmRequestType](#)は要求に対する受け取り部、方向、要求種別を決めます。

[bRequest](#)は要求種別を決めます。通常、[bmRequestType](#)は解析され、標準装置要求(Standard Device Request)処理、[標準インターフェース要求\(Standard Interface Request\)](#)処理、[標準エンドポイント要求\(Standard Endpoint Request\)](#)処理、クラス装置要求(Class Device Request)処理などのようないくつかの処理に分岐して実行されます。設定(Setup)パケットをどう解析するかは自由です。例えば初めに[bRwquest](#)を解析し、その後要求毎に種別と受け取り部を調べることもできます。

標準要求(Standard Request)は全USB装置で共通です。クラス要求(Class Request)はドライバのクラスに対して共通です。例えばHIDクラスに適合する全装置はそのクラスで共通の特有の要求群を持ちます。これらは通信(Communications)クラスに適合する装置とも、大容量記憶装置(Mass storage)クラスに適合する装置とも異なります。

この外に供給者定義の要求(Request)があります。これらは設計(開発)者が割り当てることができる要求です。通常、これらは装置毎に異なり、独自機能の構築に使われます。

共通の要求は各種機能を実行する各受け取り部を基準に直接行えます。例えば状態取得(GET_STATUS)標準要求は装置、インターフェース、エンドポイントに対して直接行えます。装置に対して行われた場合、装置は自己給電かどうかと遠隔起動の状態を示すフラグを返します。けれども同じ要求がインターフェースに行われた場合は常に0値を返し、エンドポイントについてはそのエンドポイントに対する停止フラグを返します。

[wValue](#)と[wIndex](#)は要求に関するパラメータを許し、[wLength](#)はデータ段階で転送されるべきバイト数を指定するのに使われます。

標準要求(Standard Request)

標準装置要求(Standard Device Request)は全てのUSB装置で実装が必要です。多くのファームウェアでは受け取り部で設定(Setup)パケットの解析を行うと思われ、以降はこれらに沿って記述されます。

標準装置要求(Standard Device Request)

これらは現在8つの標準装置要求があり、下表で示されます。

表12. 標準装置要求(Standard Device Request)一覧

bmRequestType	bRequest	wValue	wIndex	wLength	データ
1 00 00000	GET_STATUS (\$00)	0	0	2	装置状態値
0 00 00000	CLEAR_FEATURE (\$01)	機能選択値	0	0	なし
0 00 00000	SET_FEATURE (\$03)	機能選択値	0	0	なし
0 00 00000	SET_ADDRESS (\$05)	装置アドレス	0	0	なし
1 00 00000	GET_DESCRIPTOR (\$06)	記述銘別/指標番号	0または言語ID	記述長	記述内容
0 00 00000	SET_DESCRIPTOR (\$07)	記述銘別/指標番号	0または言語ID	記述長	記述内容
1 00 00000	GET_CONFIGURATION(\$08)	0	0	1	設定値(番号)
0 00 00000	SET_CONFIGURATION (\$09)	構成番号	0	0	なし

- 装置を指示した状態取得(GET_STATUS)要求はデータ段階で次の形式の2バイトを返します。

図30. 状態取得要求に対するデータ

ビット	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
内容	(予約)														遠隔起動	自己給電

ビット0が1ならば装置が自己給電であることを示します。0ならば装置はバス給電です。ビット1が1ならば装置が遠隔起動可能なことを示し、ホストは一時停止(Suspend)中の装置を通常動作へ起動できます。この遠隔起動ビットは機能設定(SET_FEATURE)と機能解除(CLEAR_FEATURE)の要求でDEVICE_REMOTE_WAKEUP(\$01)機能選択値によって設定/解除が行えます。

- 機能解除(CLEAR_FEATURE)と機能設定(SET_FEATURE)要求はビット番号での機能設定を使えます。装置(Device)が受け取り部の場合、利用可能な機能選択はDEVICE_REMOTE_WAKEUP(遠隔起動)とTEST_MODE(試験動作)の2つだけです。これらの詳細についてはUSB2.0仕様書をご覧ください。
- アドレス設定(SET_ADDRESS)要求はUSB装置に唯一のアドレスを割り当てるため、列举中に使われます。このアドレスはwValueで指定され、1~127の最大127個です。本要求は装置に於いて特異で、状態(Status)段階(制御(Control)転送参照)が完了するまで、そのアドレスを設定してはなりません。他の全ての要求は状態(Status)段階前にその処理を完了しなければなりません。
- 記述取得(GET_DESCRIPTOR)と記述設定(SET_DESCRIPTOR)はwValueで指定した記述内容を返すために使われます。構成記述に対する要求は1つの要求で装置記述と全てのインターフェースとエンドポイント記述を返します。インターフェース記述とエンドポイント記述はこれらの要求で直接アクセスできません。文字列記述は複数言語支援を許すためにwIndexでの言語識別(ID)を含みます。
- 構成取得(GET_CONFIGURATION)と構成選択(SET_CONFIGURATION)は現在の装置構成の選択または取得に使われます。構成取得要求の場合、装置状態を示すバイトがデータ段階中に返されます。0値は装置が未構成で、非0値は装置が構成済みであることを意味します。構成選択要求は装置を許可するのに使われます。許可する構成を選ぶためにwValueの下位バイトに希望する構成記述のbConfigurationValue値を含むべきです。

標準インターフェース要求(Standard Interface Request)

これらは現在5つの標準インターフェース要求があり、下表で示されます。

表13. 標準インターフェース要求(Stabdard Interface Request)一覧

bmRequestType	bRequest	wValue	wIndex	wLength	データ
1 00 00001	GET_STATUS (\$00)	0	インターフェース番号	2	インターフェース状態値
0 00 00001	CLEAR_FEATURE (\$01)	機能選択値	インターフェース番号	0	なし
0 00 00001	SET_FEATURE (\$03)	機能選択値	インターフェース番号	0	なし
1 00 00001	GET_INTERFACE (\$0A)	0	インターフェース番号	1	代替インターフェース値
0 00 00001	SET_INTERFACE (\$11)	代替設定値	インターフェース番号	0	なし

通常、wIndexは要求対象インターフェースを指定するのに使われます。この形式は以下で示されます。

図31. 標準インターフェース要求時のwIndex値構成

ビット	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
内容	(予約)															インターフェース番号

- 状態取得(GET_STATUS)要求はインターフェースの状態を返すのに使われます。インターフェースへのこの要求には2バイトの\$00を返すべきです(これらは将来用に予約されています)。
- 機能解除(CLEAR_FEATURE)と機能設定(SET_FEATURE)要求はビット番号での機能設定に使えます。インターフェース(Interface)が受け取り部の場合、現在のUSB2.0仕様で利用可能なインターフェース機能はありません。
- インターフェース取得(GET_INTERFACE)とインターフェース設定(SET_INTERFACE)はインターフェース記述下でその詳細が記述されている代替インターフェース設定の選択を行います。

標準エンドポイント要求(Standard Endpoint Request)

これらは現在4つの標準インターフェース要求があり、下表で示されます。

表14. 標準エンドポイント要求(Stabdard Endpoint Request)一覧

bmRequestType	bRequest	wValue	wIndex	wLength	データ
1 00 00010	GET_STATUS (\$00)	0	エンドポイント番号	2	エンドポイント状態値
0 00 00010	CLEAR_FEATURE (\$01)	機能選択値	エンドポイント番号	0	なし
0 00 00010	SET_FEATURE (\$03)	機能選択値	エンドポイント番号	0	なし
1 00 00010	SYNCH_FRAME (\$12)	0	エンドポイント番号	2	フレーム番号

通常、wIndexは要求対象エンドポイントを指定するのに使われます。この形式は以下で示されます。

図32. 標準エンドポイント要求時のwIndex値構成

ビット	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
内容	(予約)									方向	(予約)					エンドポイント番号

- 状態取得(GET_STATUS)要求はエンドポイントの状態(停止/動作不能)を示す2バイトを返します。この2バイトの形式は以下で示されます。

図33. 標準インターフェース状態取得要求時の戻り値構成

ビット	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
内容	(予約)															停止

- 機能解除(CLEAR_FEATURE)と機能設定(SET_FEATURE)要求はビット番号での機能設定に使われます。現在標準で定義されているエンドポイント機能選択は、エンドポイントの停止と解除をホストに許すENDPOINT_HALT(\$00)、1つだけです。既定のエンドポイント(エンドポイント0)以外のエンドポイントだけがこの機能を持つことを推奨されます。
- 同期フレーム(SYNCH_FRAME)要求はエンドポイントの同期フレーム番号を報告するのに使われます。

引用文献

■ usb_20.pdf	http://www.usb.org	USB協会著
■ USB_LANGIDs.pdf	http://www.usb.org	USB協会著
■ usb-in-a-nutshell.pdf	http://www.beyondlogic.org	Craig Peacock著

© HERO 2021

本書はUSBの概要説明のために引用文献を基に和訳再構成したものです。一部意味が想像し易いように本来の用語とは異なる用語で記述されています。本書未記載部分に関してはUSB2.0仕様書を参照してください。

必要と思われる部分には()内に英語表記や略称などを残す形で表記しています。

青字の部分はリンクとなっています。一般的に赤字の0,1は論理0,1を表します。その他の赤字は重要な部分を表します。

付録-A : CRC(Cyclic Redundant Check)

CRCはビット列データの誤り検出/訂正に用いられる方法の1つです。誤り検出についてはパリティやチェックサムなどの簡単な方法が多く用いられます。これらは誤り検出能力が低く、訂正に関してはその機能を持ちません。これらに対してCRCは誤り検出能力が高く、基本的に誤り訂正能力があります。けれども誤り訂正能力を高めると、CRC用の冗長ビットも増えるので、一般的に用いられているCRCは誤り検出能力に力点を置いて策定されています。USBで用いられているのはCRC5とCRC16の2種類です。

ここではソフトウェアによるCRCの扱いに関する情報を記述します。

多くのCRC関連情報では最初に多項式での説明が成されます。これらで記述される多項式はn進数に対する表現形式の1つで、以下の χ は基底(2進数ならば2、10進数ならば10)を示します。例えば\$4321は次のように表現されます。

$$\begin{aligned} \$4321 &= 0100\ 0011\ 0010\ 0001 \\ &= 0 \times x^{15} + 1 \times x^{14} + 0 \times x^{13} + 0 \times x^{12} + 0 \times x^{11} + 0 \times x^{10} + 1 \times x^9 + 1 \times x^8 + 0 \times x^7 + 0 \times x^6 + 0 \times x^5 + 1 \times x^4 + 0 \times x^3 + 0 \times x^2 + 0 \times x^1 + 1 \times x^0 \\ &= x^{14} + x^9 + x^8 + x^4 + x^0 \end{aligned}$$

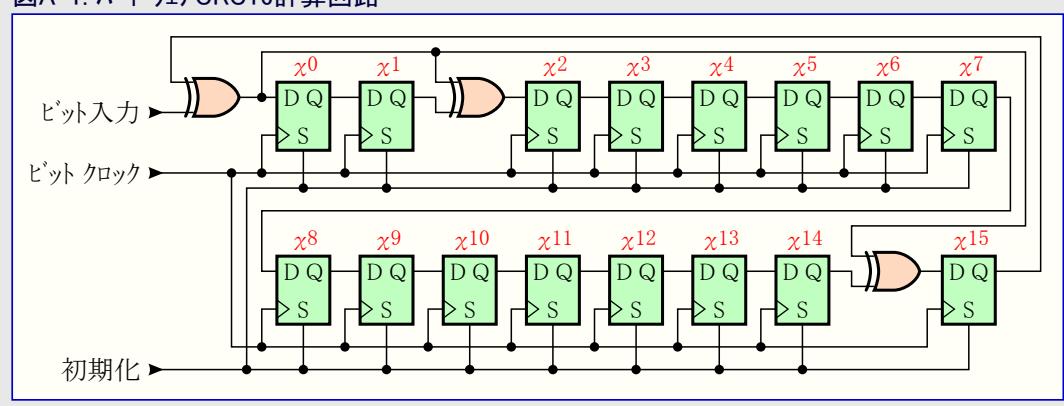
CRCの処理は簡単に言えば対象領域を特定の値で割った剰余値を誤り検出/訂正值として送信時に付加し、受信側では同様の演算結果と付加された誤り検出/訂正值の比較によって誤りの検出と訂正を行うことです。USBに於ける、この特定の値は次の2種類です。

$$\begin{array}{lll} \text{CRC5} \cdots x^5 + x^2 + x^0 & = & 10\ 0101 \quad (\text{5ビット剩余}) \\ \text{CRC16} \cdots x^{16} + x^{15} + x^2 + x^0 & = & 1\ 1000\ 0000\ 0000\ 0101 \quad (\text{16ビット剩余}) \end{array}$$

対象領域のビット列を上記の値で割った剰余が付加されるCRC値です。

USB仕様では上記の基本に対していくつか異なる点があります。またUSBではこれらのCRC計算がハードウェアで行われるのが前提ですので、ハードウェアでの実装に都合よく策定されています。USBに於けるハードウェアによる実際のCRC16計算回路を以下に示します。
(以下の記述ではCRC16を基本にしますが、CRC5も同様の手順で処理されます。)

図A-1. ハードウェアCRC16計算回路



- 除算は通常の筆算による方法と同一で、最上位桁側から比較桁数を増やしながら除数より大きな数値として、その数値÷除数を商の或る桁値とし、以降最終桁まで繰り返します。2進数の場合は「商の或る桁値」が0または1しか有り得ませんので、「その数値÷除数」は単純に「その数値-除数」、即ち減算で処理できます。この演算は2を法とするため、桁上がりや桁下がりがありません。このためビット単位に減算され、他のビットに影響を及ぼさないと言えます。従ってこの「減算」は「排他的論理和(EOR)」で実現されます。
 - 上図の各レジスタの初期値は全1です。これは0が先行し、それ以外が同一値のビット列に於いて、先行連続0の数に拘らず同一CRC値となるのを防止します。
 - 上図で得られた剩余値($x^{15} \sim x^0$)は論理反転され、且つビット位置が反転($x^0 \sim x^{15}$)されて最終的なCRC値になります。このビット位置反転はEOPの00の影響を排除するためです。受信側で同一の方法によってこのCRC値までを計算すると、上図での剩余値($x^{15} \sim x^0$)は正常に受信された場合、常に次の値になります。

$$\begin{array}{lll} \text{CRC5} \cdots x^3 + x^2 & = & 0\ 1100 \quad (\text{5ビット剩余}) \\ \text{CRC16} \cdots x^{15} + x^3 + x^2 + x^0 & = & 1000\ 0000\ 0000\ 1101 \quad (\text{16ビット剩余}) \end{array}$$

付録-B：言語識別子

言語識別子は16ビットで構成されます。これはビット9～0の10ビットで表される主言語識別子とビット15～10の6ビットで表される補助言語識別子の組み合わせです。

表B-1. 言語識別(ID)符号一覧

ID	言語	ID	言語	ID	言語	ID	言語
\$0401	アラビア語(サウジアラビア)	\$080A	スペイン語(メキシコ)	\$081A	セルビア語(テヘン文字)	\$0441	スワヒリ語(ケニア)
\$0801	アラビア語(イラク)	\$0C0A	スペイン語(近代)	\$0C1A	セルビア語(キリル文字)	-	-
\$0C01	アラビア語(エジプト)	\$100A	スペイン語(グアテマラ)	\$041B	スロバキア語	\$0443	ウズベク語(テヘン文字)
\$1001	アラビア語(リビア)	\$140A	スペイン語(コスタリカ)	\$041C	アルバニア語	\$0843	ウズベク語(キリル文字)
\$1401	アラビア語(アルジェリア)	\$180A	スペイン語(パナマ)	\$041D	スウェーデン語	\$0444	タール語(タールスタン)
\$1801	アラビア語(モロッコ)	\$1C0A	スペイン語(ミニカ)	\$081D	スウェーデン語(フィンランド)	\$0445	ヘンガル語
\$1C01	アラビア語(チニジア)	\$200A	スペイン語(ペネズエラ)	\$041E	タイ語	\$0446	パンジャブ語
\$2001	アラビア語(オーマン)	\$240A	スペイン語(コロンビア)	\$041F	トルコ語	\$0447	グジヤラート語
\$2401	アラビア語(イエメン)	\$280A	スペイン語(ペルー)	\$0420	ウルドゥー語(パキスタン)	\$0448	オリヤー語
\$2801	アラビア語(シリア)	\$2C0A	スペイン語(アルゼンチン)	\$0820	ウルドゥー語(インド)	\$0449	タミル語
\$2C01	アラビア語(ヨルダン)	\$300A	スペイン語(エクアドル)	\$0421	インドネシア語	\$044A	テルグ語
\$3001	アラビア語(レバノン)	\$340A	スペイン語(チリ)	\$0422	ウクライナ語	\$044B	カンナダ語
\$3401	アラビア語(クウェート)	\$380A	スペイン語(ウルグアイ)	\$0423	ペルルーシ語	\$044C	マラヤラム語
\$3801	アラビア語(U.A.E.)	\$3C0A	スペイン語(パラグアイ)	\$0424	スロベニア語	\$044D	アッサム語
\$3C01	アラビア語(バーレーン)	\$400A	スペイン語(ボリビア)	\$0425	エストニア語	\$044E	マラティー語
\$4001	アラビア語(カタール)	\$440A	スペイン語(エルサルバドル)	\$0426	ラビア語	\$044F	サンスクリット語
\$0402	ブルガリア語	\$480A	スペイン語(ホンジュラス)	\$0427	リトニア語	-	-
\$0403	カタロニア語	\$4C0A	スペイン語(ニカラグア)	\$0827	リトニア語(旧)	-	-
\$0404	中国語(繁体)	\$500A	スペイン語(ペルトリコ)	-	-	-	-
\$0804	中国語(簡体)	\$040B	フィンランド語	\$0429	ペルシャ語	-	-
\$0C04	中国語(ホンコン)	\$040C	仏語(標準)	\$042A	ベトナム語	-	-
\$1004	中国語(シンガポール)	\$080C	仏語(ベルギー)	\$042B	アルメニア語	\$0455	ビルマ語
\$1404	中国語(マカオ)	\$0C0C	仏語(カナダ)	\$042C	アゼリ語(テヘン文字)	-	-
\$0405	チェコ語	\$100C	仏語(スイス)	\$082C	アゼリ語(キリル文字)	\$0457	コンカニ語
\$0406	デンマーク語	\$140C	仏語(ルクセンブルグ)	\$042D	バスク語	\$0458	マニプリ語
\$0407	独語(標準)	\$180C	仏語(モナコ)	-	-	\$0459	シント語
\$0807	独語(スイス)	\$040D	ヘブライ語	\$042F	マケドニア語	-	-
\$0C07	独語(オーストリア)	\$040E	ハンガリー語	\$0430	スツ語	-	-
\$1007	独語(ルクセンブルグ)	\$040F	アイスランド語	-	-	-	-
\$1407	独語(リヒテンシュタイン)	\$0410	伊語(標準)	-	-	-	-
\$0408	ギリシャ語	\$0810	伊語(スイス)	-	-	-	-
\$0409	英語(米国)	\$0411	日本語	-	-	-	-
\$0809	英語(英国)	\$0412	韓国語	-	-	\$0860	カシミール語(インド)
\$0C09	英語(豪州)	\$0812	韓国語(ハンブル)	\$0436	公用オランダ語(南アフリカ)	\$0861	ネパール語(インド)
\$1009	英語(カナダ)	\$0413	オランダ語(オランダ)	\$0437	グルジア語	-	-
\$1409	英語(ニュージーランド)	\$0813	オランダ語(ベルギー)	\$0438	フェロー語	-	-
\$1809	英語(アイルランド)	\$0414	ノルウェー語(ブーケモール)	\$0439	ヒンディー語	-	-
\$1C09	英語(南アフリカ)	\$0814	ノルウェー語(ニーアシク)	-	-	-	-
\$2009	英語(ジャマイカ)	\$0415	ポーランド語	-	-	-	-
\$2409	英語(カリブ)	\$0416	ポルトガル語(ブラジル)	-	-	-	-
\$2809	英語(ヘリース)	\$0816	ポルトガル語(標準)	-	-	\$04FF	HID(データ記述子使用)
\$2C09	英語(トリニダートバゴ)	-	-	\$043E	マラヤ語(マレーシア)	\$FOFF	HID(供給者定義1)
\$3009	英語(シンハラブエ)	\$0418	ロマンス語	\$083E	マラヤ語(ブルネイ)	\$F4FF	HID(供給者定義2)
\$3409	英語(フィリピン)	\$0419	露語	\$043F	カザフ語	\$F8FF	HID(供給者定義3)
\$040A	スペイン語(伝統的)	\$041A	クロアチア語	-	-	\$FEFF	HID(供給者定義4)