お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注 1. 本資料において使用されている「当社」とは、ルネサスエレクトロニクス株式会社およびルネサスエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

お客様各位

資料中の「日立製作所」、「日立XX」等名称の株式会社ルネサス テクノロジへの変更について

2003年4月1日を以って三菱電機株式会社及び株式会社日立製作所のマイコン、ロジック、アナログ、ディスクリート半導体、及びDRAMを除くメモリ(フラッシュメモリ・SRAM等)を含む半導体事業は株式会社ルネサス テクノロジに承継されました。従いまして、本資料中には・「日立製作所」、「株式会社日立製作所」、「日立半導体」、「日立XX」といった表記が残っておりますが、これらの表記は全て「株式会社ルネサス テクノロジ」に変更されておりますのでご理解の程お願い致します。尚、会社商標・ロゴ・コーポレートステートメント以外の内容については一切変更しておりませんので資料としての内容更新ではありません。

ルネサステクノロジ ホームページ (http://www.renesas.com)

2003年4月1日
株式会社ルネサス テクノロジ
カスタマサポート部

RENESAS

ご注意

安全設計に関するお願い

1. 弊社は品質、信頼性の向上に努めておりますが、半導体製品は故障が発生したり、誤動作する場合があります。弊社の半導体製品の故障又は誤動作によって結果として、人身事故、火災事故、社会的損害などを生じさせないような安全性を考慮した冗長設計、延焼対策設計、誤動作防止設計などの安全設計に十分ご留意ください。

本資料ご利用に際しての留意事項

1. 本資料は、お客様が用途に応じた適切なルネサス テクノロジ製品をご購入いただくための参考資料であり、本資料中に記載の技術情報についてルネサス テクノロジが所有する知的財産権その他の権利の実施、使用を許諾するものではありません。
2. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例の使用に起因する損害、第三者所有の権利に対する侵害に関し、ルネサス テクノロジは責任を負いません。
3. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他全ての情報は本資料発行時点のものであり、ルネサス テクノロジは、予告なしに、本資料に記載した製品または仕様を変更することがあります。ルネサス テクノロジ半導体製品のご購入に当たりましては、事前にルネサス テクノロジ、ルネサス販売または特約店へ最新の情報をご確認頂きますとともに、ルネサス テクノロジホームページ(http://www.renesas.com)などを通じて公開される情報に常にご注意ください。
4. 本資料に記載した情報は、正確を期すため、慎重に制作したものですが万一本資料の記述誤りに起因する損害がお客様に生じた場合には、ルネサス テクノロジはその責任を負いません。
5. 本資料に記載の製品データ、図、表に示す技術的な内容、プログラム及びアルゴリズムを流用する場合は、技術内容、プログラム、アルゴリズム単位で評価するだけでなく、システム全体で十分に評価し、お客様の責任において適用可否を判断してください。ルネサス テクノロジは、適用可否に対する責任は負いません。
6. 本資料に記載された製品は、人命にかかわるような状況の下で使用される機器あるいはシステムに用いられることを目的として設計、製造されたものではありません。本資料に記載の製品を運輸、移動体用、医療用、航空宇宙用、原子力制御用、海底中継用機器あるいはシステムなど、特殊用途へのご利用をご検討の際には、ルネサス テクノロジ、ルネサス販売または特約店へご照会ください。
7. 本資料の転載、複製については、文書によるルネサス テクノロジの事前の承諾が必要です。
8. 本資料に関し詳細についてのお問い合わせ、その他お気付きの点がございましたらルネサス テクノロジ、ルネサス販売または特約店までご照会ください。

Application Note

RENESAS

H8S/2215
USB ファンクションモジュール

アプリケーションノート

ルネサスSuperH™ RISC engine

H8S/2215　　HD64F2215

ルネサス エレクトロニクス
www.renesas.com

Rev.1.00　2002.03

ご注意

1　本書に記載の製品及び技術のうち「外国為替及び外国貿易法」に基づき安全保障貿易管理関連貨物・技術に該当するものを輸出する場合，または国外に持ち出す場合は日本国政府の許可が必要です。

2　本書に記載された情報の使用に際して，弊社もしくは第三者の特許権，著作権，商標権，その他の知的所有権等の権利に対する保証または実施権の許諾を行うものではありません。また本書に記載された情報を使用した事により第三者の知的所有権等の権利に関わる問題が生じた場合，弊社はその責を負いませんので予めご了承ください。

3　製品及び製品仕様は予告無く変更する場合がありますので，最終的な設計，ご購入，ご使用に際しましては，事前に最新の製品規格または仕様書をお求めになりご確認ください。

4　弊社は品質・信頼性の向上に努めておりますが，宇宙，航空，原子力，燃焼制御，運輸，交通，各種安全装置，ライフサポート関連の医療機器等のように，特別な品質・信頼性が要求され，その故障や誤動作が直接人命を脅かしたり，人体に危害を及ぼす恐れのある用途にご使用をお考えのお客様は，事前に弊社営業担当迄ご相談をお願い致します。

5　設計に際しては，特に最大定格，動作電源電圧範囲，放熱特性，実装条件及びその他諸条件につきましては，弊社保証範囲内でご使用いただきますようお願い致します。
保証値を越えてご使用された場合の故障及び事故につきましては，弊社はその責を負いません。また保証値内のご使用であっても半導体製品について通常予測される故障発生率，故障モードをご考慮の上，弊社製品の動作が原因でご使用機器が人身事故，火災事故，その他の拡大損害を生じないようにフェールセーフ等のシステム上の対策を講じて頂きますようお願い致します。

6　本製品は耐放射線設計をしておりません。

7　本書の一部または全部を弊社の文書による承認なしに転載または複製することを堅くお断り致します。

8　本書をはじめ弊社半導体についてのお問い合わせ，ご相談は弊社営業担当迄お願い致します。

はじめに

本アプリケーションノートは、H8S/2215 内蔵の USB ファンクションモジュールについて説明したものであり、お客様が USB ファンクションモジュールファームウエア作成の際に、御参考として役立てて頂ける様にまとめました。本アプリケーションノートではプリンタクラスの通信を例に、H8S/2215 内蔵 USB ファンクションモジュールの構成を説明します。本アプリケーションノートの内容およびソフトウェアは、USB ファンクションモジュールの使用例として説明しているものであり、その内容を保証するものではありません。

また、開発に際しましては、本書のほかに以下の関連マニュアルもあわせて御覧ください。

【関連マニュアル】

- Universal Serial Bus Specification Revision 1.1
- Universal Serial Bus Device Class Definition for Printing Devices
- H8S/2215 ハードウエアマニュアル
- H8S/2215 Solution Engine（MS2215CP01_C/S）取扱説明書
- Solution Engineシングルチップマイコンベースボード（MSSCBB01）取扱説明書
- H8S/2215 E10A エミュレータユーザーズマニュアル

【注意】 本アプリケーションノートに記載してあるサンプルプログラムでは、USB **の転送タイプのうち「インタラプト」に関するファームウエアは準備しておりません。「インタラプト」**(H8S/2215 **ハードウエアマニュアル** 19-1 **参照)の転送タイプを御使用になる場合は、別途お客様でプログラムを作成していただく必要があります。**

また、本アプリケーションノートには、上記システムの開発時に必要と思われる H8S/2215、H8S/2215 Solution Engine **のハードウエア仕様を記載してありますが、詳細は** H8S/2215 **のハードウエアマニュアル、ならびに** H8S/2215 Solution Engine **の取扱説明書を御覧ください。**

目次

1. 概要

本アプリケーションノートは、H8S/2215 の USB ファンクションモジュールの使用方法、およびファームウエアの作成例について説明したものです。

H8S/2215 内蔵 USB ファンクションモジュールの特長を以下に示します。

- USB1.1に準拠したUDC（USB Device Controller）を内蔵
- USBプロトコルを自動処理
- エンドポイント0に対するUSB標準コマンドを自動処理（一部コマンドはファームウエアで処理する必要があります。）
- フルスピード（12Mbps）転送に対応
- USB送受信に必要な各種割り込み信号を生成
- クロック発振器内のUSBクロック選択回路により、USB動作クロックを内部システムクロック（16MHz）3逓倍／外部入力（48MHz）を選択可能
- バストランシーバを内蔵
- 任意のエンドポイント構成が設定可能

エンドポイントの構成

エンドポイント名	名称	転送タイプ	最大パケットサイズ	FIFO バッファ容量	DMA 転送
エンドポイント 0	EP0s	セットアップ	8 Byte	8 Byte	―
	EP0i	コントロールイン	64 Byte	64 Byte	―
	EP0o	コントロールアウト	64 Byte	64 Byte	―
エンドポイント任意	EPn	インタラプト（イン）	64 Byte	64 Byte (可変)	―
エンドポイント任意	EPn	バルクイン	64 Byte	64×2（128 Byte）	可能
エンドポイント任意	EPn	バルクアウト	64 Byte	64×2（128 Byte）	可能
エンドポイント任意	EPn	アイソクロナス（イン）	128 Byte	128×2（可変）	―
エンドポイント任意	EPn	アイソクロナス（アウト）	128 Byte	128×2（可変）	―
エンドポイント任意	EPn	バルクイン	64 Byte	64×2（128 Byte）	可能
エンドポイント任意	EPn	バルクアウト	64 Byte	64×2（128 Byte）	可能
エンドポイント任意	EPn	インタラプト（イン）	64 Byte	64 Byte（可変）	―

システム構成例を**図** 1.1 に示します。

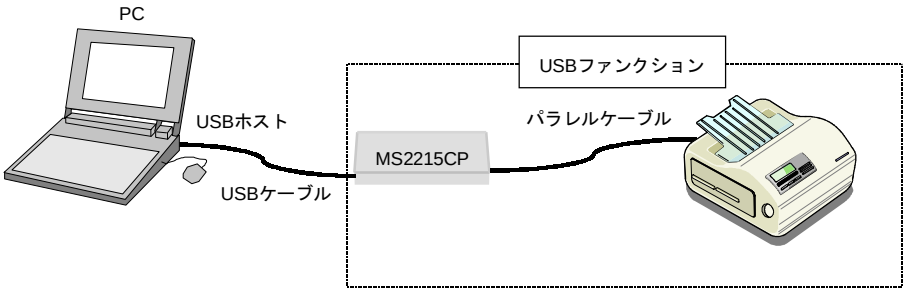


図 1.1　**システム構成**

本システムは、H8S/2215 を搭載した日立超 LSI システムズ社製の H8S/2215 Solution Engine（以下 MS2215CP）、パラレルポート搭載のプリンタ、Windows2000 を OS とした PC によって構成されています。

本システムは、ホスト PC から USB にて送信される印刷データを、MS2215CP によって受信しパラレルに変換後、プリンタへ印刷データを出力することができます。また、Windows2000 に標準で付属している USB プリンタクラスのデバイスドライバ、およびプリンタのデバイスドライバを使用することが可能です。

本システムの特長を以下に示します。

1. サンプルプログラムにより、H8S/2215のUSBモジュールを短期間で評価可能
2. サンプルプログラムは、USBのコントロール転送、バルク転送をサポート
3. E6000を使用することができ、効率的なデバッグが可能
4. プログラムを追加作成することにより、インタラプト、アイソクロナス転送についても対応可能*

【注】* インタラプト、アイソクロナス転送のプログラムは、お客様で作成していただく必要があります。

2. USB の概要

この章では、接続トポロジ、転送方式、データフォーマット等 USB の規格について説明します。USB のシステムを開発する際に、ご参考としてお使いください。なお、規格の詳細につきましては、「Universal Serial Bus Specification Revision 1.1（USB1.1 **規格書）」**をご覧ください。

2.1 USB の接続トポロジ

図 2.1 に USB の接続トポロジを示します。USB は PC に搭載されたホストコントローラおよびホストコントローラに接続されるデバイスによって構成されています。また、ハブと呼ばれる特別なデバイスを用いることによりバスを拡張して、接続するデバイス数を増やすことが可能です。特に、ホストコントローラに直結されているハブをルートハブと呼び、一般的には PC 本体内に格納されています。ハブ（ルートハブを除く）は 5 階層まで（直列で 5 台）接続可能となっています。

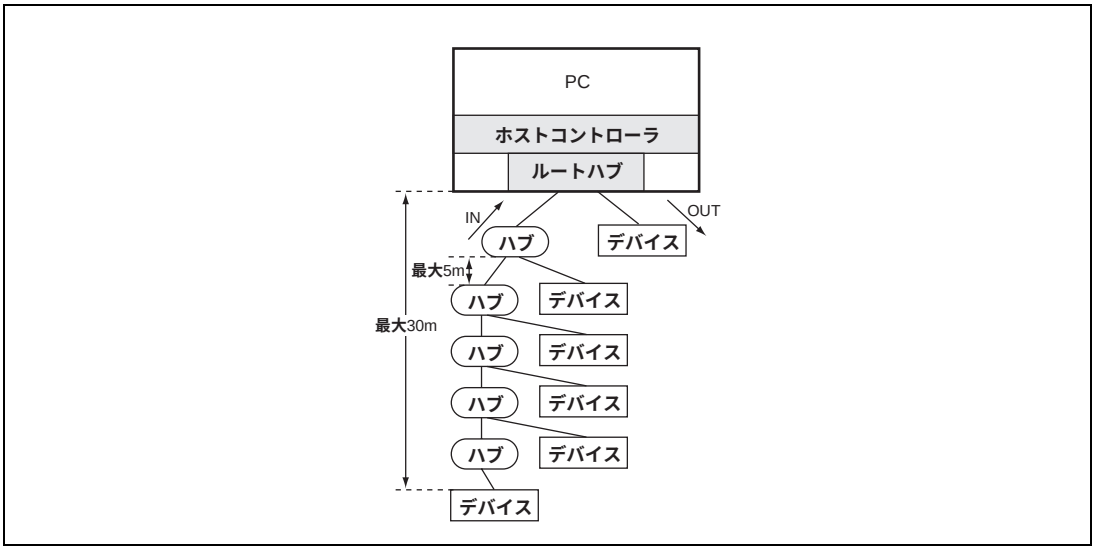


図 2.1　**接続トポロジ**

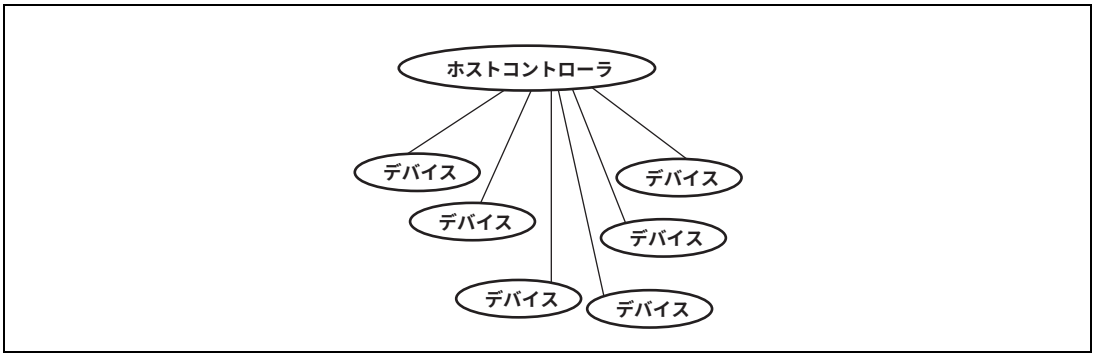


図 2.2　**論理トポロジ**

ホストコントローラは、各デバイスに 7bit のアドレスを割り当てて管理しています。このうち、デバイスが接続されてから、アドレスが割り当てられるまでに使用される仮のアドレス（デフォルトアドレス：0000000b）が必要となるため、ホストコントローラに接続できるデバイスは、ハブも含み最大 127 台となります。

なお、実際の接続トポロジは**図** 2.1 で示したツリー型になりますが、論理トポロジは**図** 2.2 で示すスター型となり、ホストコントローラとデバイスが時分割で 1 対 1 の通信を行う形になります（実際にハブを経由しても、ホストコントローラに直結されているイメージになります）。時分割のスケジュールはすべてホストコントローラが決定します。そのため、ホストコントローラ側からの命令（詳細は**「**2.6.1　**パケットの概要」**をご参照ください）がない限り、デバイス側からデータを転送することはありません。

また、デバイスには高速転送（12Mbps）を行う「フルスピードデバイス」および低速転送（1.5Mbps）を行う「ロースピードデバイス」があります。

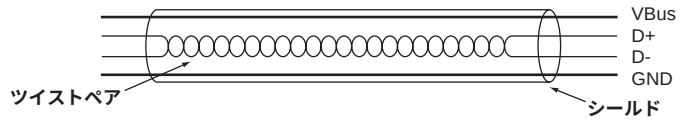
データの転送方向はホストコントローラを中心に定義されており、データがホストコントローラからデバイスに向かう場合を OUT 方向、デバイスからホストコントローラに向かう場合が IN 方向です。

OUT 方向のデータは接続されている全デバイスに向けて転送されるブロードキャスト型となっています。なお、ロースピードデバイスに対しては 1.5Mbps のデータのみ転送されます（ルートハブあるいはハブによって、12Mbps のデータがフィルタリングされます。詳細は**「**2.6.1　**パケットの概要」**で説明します）。

ブロードキャストされる OUT 方向の「トークンパケット」には、デバイスがデータを識別するために必要な「アドレス情報」（詳細は**「**2.6.1　**パケットの概要」**をご参照ください）が含まれており、このアドレス情報を基に該当するアドレスのデバイスのみが動作・応答する形になります。

2.2　USB の信号転送方式

USB は、2 本の信号ライン（D+、D-）と 2 本の電源ライン（Vbus、GND）で構成されています。それに合わせ、**図** 2.3 に示すよう USB ケーブルの内部も 4 本で構成されています。このうち、フルスピードデバイスで使用されるケーブルでは、信号ライン（D+、D-）がツイストペア（撚り線）構造になっています。また、フルスピードデバイス用ケーブルにはシールド処理も必要となりますが、ロースピードデバイスで使用されるケーブルではツイストペア・シールド共に不要です。なお、ケーブルの最大長はフルスピードデバイス用で 5m ですが、ツイストペア・シールド処理を行わないロースピードデバイス用では 3m までとなります。



【注】ロースピードデバイス用ケーブルではツイストペア・シールド共に不要です。

図 2.3　USB ケーブルの構造（フルスピードデバイス用）

データは、D+、D-を使用した差動信号によって転送されます。転送方式は**図 2.4** に示すように、元データが「0」であるときには D+、D-それぞれが反転し、「1」であるときには反転しない「NRZI」（Non Return to Zero Invert）方式が用いられています。NRZI では、元データに「1」が続いた場合は信号変化がなくなるため、ホストコントローラとデバイス間の同期がずれる恐れがあります。そこで、**図 2.5** のように、「1」が 6 ビット以上連続した場合は、強制的に「0」を挿入させて反転を起こすことにより、同期がずれることを防いでいます（「ビットスタッフ」と呼びます）。ここで挿入された「0」は転送後に受信側で除去されます。

データが転送されていない状態ではデバイス内のプルアップ抵抗に従い、フルスピードデバイスでは D+がハイレベル・D-がローレベルに、ロースピードデバイスでは D+がローレベル・D-がハイレベルになっています。この状態を「アイドル」と呼びます。

USB ではデータをパケット単位で転送します（パケットについての詳細は**「2.6　USB のパケットとデータ転送」**をご参照ください）。

パケットの先頭は SYNC（synchronization）と呼ばれ、「00000001」の固定値となっています。

アイドル状態から SYNC の第 1 ビットにより、D+、D-が反転する部分を「SOP」（Start Of Packet）と呼んでいます（**図 2.6**）。

また、パケットの終端部はパケット終了を識別するため D+、D-共にローレベル（2 ビット時間）である特殊な信号となっていて、「EOP」（End Of Packet）と呼んでいます（**図 2.7**）。

なお、**図 2.4**、**図 2.5**、**図 2.6**、**図 2.7** 中の NRZI 後差動信号波形は、フルスピードデバイス接続時のものです。ロースピードデバイス接続時は D+と D-の関係が逆になります（EOP はデバイスの転送速度に関わらず、D+、D-共にローレベルになります）。

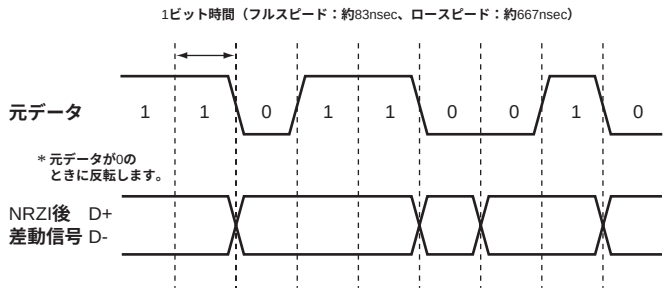


図 2.4　NRZI 伝送方式

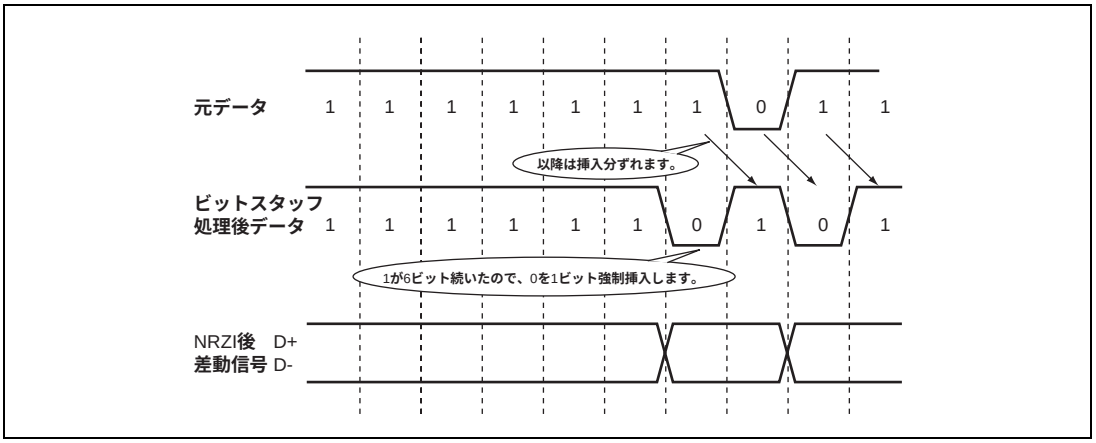


図 2.5 **ビットスタッフ**

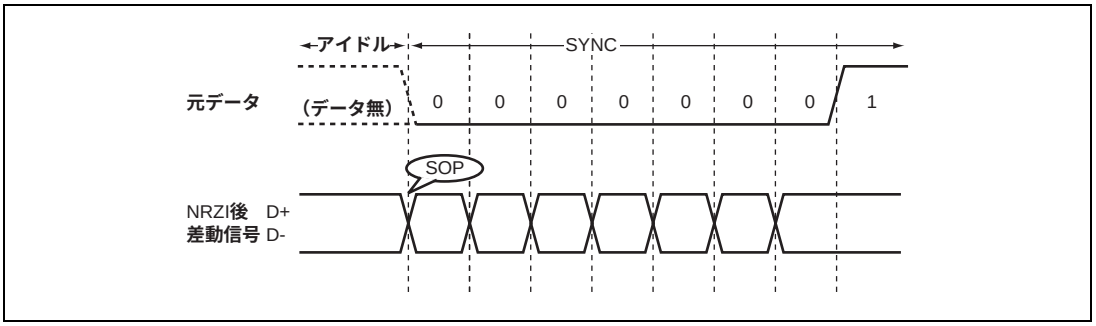


図 2.6 SOP **と** SYNC

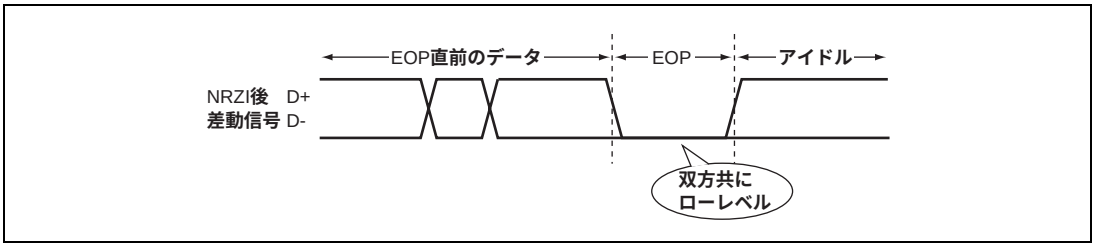


図 2.7 EOP

電源ライン（Vbus、GND）は１つのデバイスに対して、電源電圧 5V・最大 500mA までの電流を供給することが可能です。

ただし、接続直後に使用できる電流は 100mA 以内です。接続後、100mA 以内の消費電流で「標準コマンド」（**「**2.7.2 **デバイスリクエスト」**をご参照ください）を用いた初期設定を行います。

この設定の中で、ホストコントローラは接続されたデバイスの最大使用電流情報を読み込みます。（**「**2.8 **ディスクリプタ」**にて後述いたします「ディスクリプタ」情報中に含まれています。）この情報に基づき、ホストコント

ローラが電源供給に対して問題がないと判断した場合、デバイスは初めて 500mA まで消費電流を増加することが許可されます。

なお、500mA を超えた電流が必要であるデバイスについては、デバイス自体に電源を持つ必要があります。

【注】　自己電源供給機能を持たないハブ（バスパワードハブ）を使用した場合、1 ポート当りで使用できる電流は最大 100mA までに制限されます。バスパワードハブに 100mA を超える電流を使用するデバイスを接続した場合、ホストコントローラは上記の初期設定で電源供給が不可と判断します。この場合ホストコントローラは、バスパワードハブに対して接続されたデバイスへ向けての電源供給を行わないように制御します。

2.3　接続・非接続の認識

ホストコントローラおよびハブのダウンストリーム側（デバイス側）では、D+、D-を 15KΩでプルダウンしています。それに対してデバイス側では、フルスピードデバイスの場合は D+を、ロースピードデバイスの場合は D-を 1.5KΩでプルアップしています。したがって、デバイスをホストコントローラ、もしくはハブに接続した場合、D+と D-のどちらがプルアップされたかによって、ホストコントローラまたはハブは接続されたデバイスの転送速度を認識することができます。**表** 2.1 にホストコントローラ／ハブの D+と D-の状態と接続デバイスの関係、および**図** 2.8、**図** 2.9 に実際の回路構成を示します。

表 2.1　**信号ラインと接続デバイスの関係**

D+	D-	接続デバイス
プルアップ	**プルダウン**	**フルスピードデバイス**
プルダウン	**プルアップ**	**ロースピードデバイス**
プルダウン	**プルダウン**	**デバイス未接続**
プルアップ	**プルアップ**	**禁止**

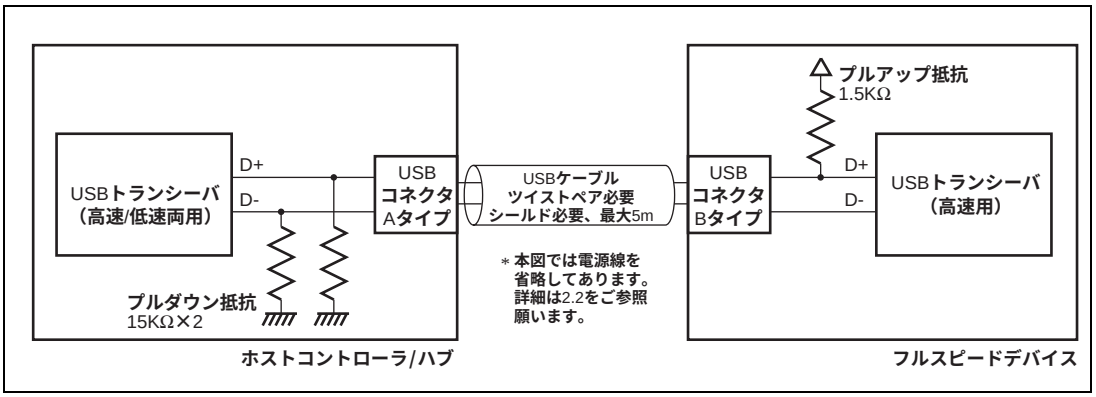


図 2.8　**フルスピードデバイスの場合**

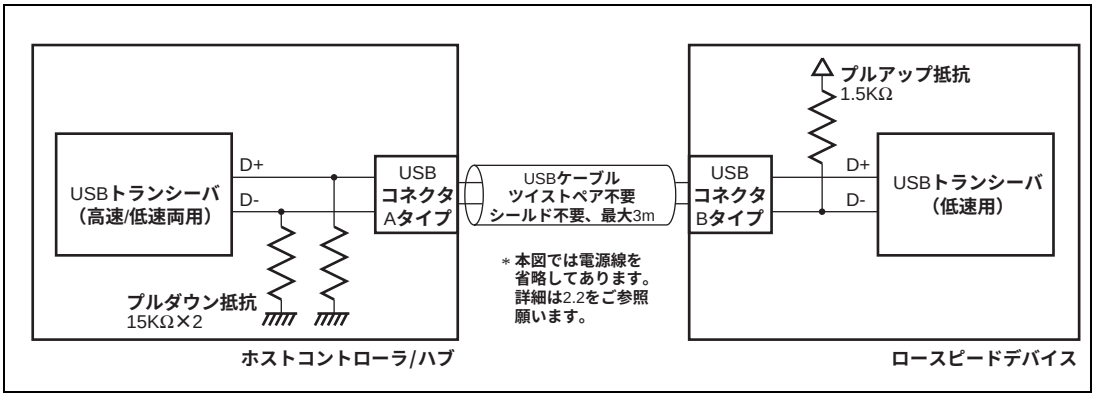


図 2.9　**ロースピードデバイスの場合**

2.4　USB コネクタ

USB で使用されるコネクタには、ホストコントローラ側で使用される平形の「A タイプコネクタ」（**図** 2.10）およびデバイス側で使用される角型の「B タイプコネクタ」（**図** 2.11）の 2 種類があります。コネクタの形状を分けることにより、物理的に誤接続を避ける構造になっています。（USB ではホストコントローラ同士やデバイス同士の接続は禁止されています。）

ハブの場合、アップストリーム側（ホストコントローラ接続側）に「B タイプコネクタ」およびダウンストリーム側（デバイス接続側）に「A タイプコネクタ」が使用されています。

図2.10　Aタイプコネクタ

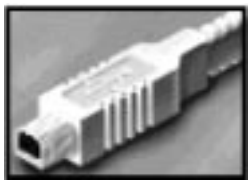


図2.11　Bタイプコネクタ

2.5　エンドポイント

デバイスは各々エンドポイント（EP）と呼ばれる FIFO を持っています。ホストコントローラとデバイスがデータを送受信する際には、このエンドポイントを介して行います。1 つのデバイスが持つことができるエンドポイントの数は、デバイスの転送速度によって**表** 2.2 のように規定されています。

表 2.2　**設定可能なエンドポイント数**

デバイス転送速度	エンドポイント番号	最大設定数
フルスピード（12Mbps）	0 ～ 15	**イン・アウトそれぞれ** 16 **個**
ロースピード(1.5Mbps)	0 ～ 2	**イン・アウトそれぞれ** 3 **個**

表 2.2 のうち、番号が 0 のエンドポイントは、コントロール転送（**「**2.6.2　**コントロール転送」**参照）に使用されるエンドポイントです。エンドポイント 0 は、すべてのデバイスが必ず設けなければなりません。一方、番号 1～15 のエンドポイントは必要な数だけ使用することができ、各エンドポイントのデータの向きや用途は、デバイス設計時に自由に設定することができます。ただし、USB1.0 でのインタラプト転送（**「**2.6.5 **インタラプト転送」**参照）は IN 方向のみとなります。

エンドポイントは、転送方式によって一度に送受信可能なデータの最大値が規定されています。規定より大きいサイズのデータを送受信することはできませんが、小さいサイズのデータ（ショートパケット）は、送受信可能です。転送方式ごとのエンドポイントのデータサイズを**表** 2.3 に示します。**表** 2.3 の範囲内で、エンドポイントごとに任意のデータサイズを設定することができます。

表 2.3　**データサイズの最大値（単位はバイト）**

	転送方式			
デバイス転送速度	**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**
フルスピード	8、16、32、64	8、16、32、64	0～64（**任意の整数値**）	0～1023（**任意の整数値**）
ロースピード	8	**使用不可**	0～8（**任意の整数値**）	**使用不可**

【注】なお、各転送方式については、2.6.2～2.6.5 **をご参照ください。**

2.6　USB のパケットとデータ転送

USB ではデータをパケットの単位で転送します。パケットは、USB のデータにおける最小の単位です。USB のプロトコルは、いくつかのパケットを組み合わせたものを用いて通信しています。この組み合わせをトランザクションと呼びます。トランザクションはトークン、データ、ハンドシェークの順にパケットが並べられています。

また、トランザクションが集まって構成されたものをフレームと呼びます（**図** 2.12 参照）。

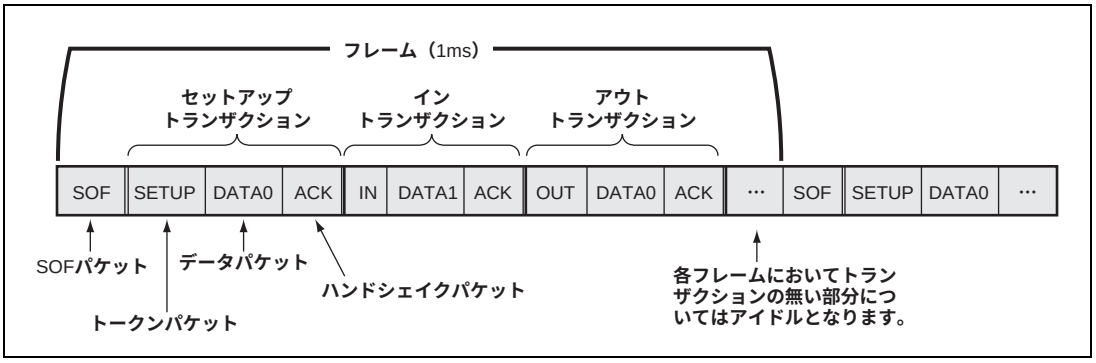


図 2.12　**トランザクションとフレーム**

1 つのフレームは 1ms ごとに発行される「SOF パケット」より開始され、次の SOF まで続きます。フレーム内のトランザクションのスケジューリングは、すべてホストコントローラが行います。

なお、各フレーム内においてトランザクションで満たされない部分（データがない部分）については、**「**2.2 USB **の信号転送方式」**で説明いたしました「アイドル」状態となります。

トランザクションは、ある規定された順序にしたがってホストコントローラとデバイス間で送受信されています。以下に USB で使用されるパケットおよびそれぞれの転送方式の特長とフォーマットを示します。

2.6.1 パケットの概要

USB で使用されるパケットは、規格によりフォーマットが決められています。**表** 2.4 に示すように、パケットは「SOF」「トークン」「データ」「ハンドシェーク」「スペシャル」の 5 種類に分類できます。これらは 4bit の PID（PacketID）により区別されます。

表 2.4 PID **の一覧**

PID **タイプ**	PID **名**	**送信側**	PID[3：0]
SOF	SOF	**ホストコントローラ**	0101
トークン	OUT	**ホストコントローラ**	0001
	IN	**ホストコントローラ**	1001
	SETUP	**ホストコントローラ**	1101
データ	DATA0	**ホストコントローラ／デバイス**	0011
	DATA1	**ホストコントローラ／デバイス**	1011
ハンドシェイク	ACK	**ホストコントローラ／デバイス**	0010
	NAK	**デバイス**	1010
	STALL	**デバイス**	1110
スペシャル	PRE	**ホストコントローラ**	1100

次に、各パケットのフォーマットを示します。パケットは、SYNC を先頭に PID、$\overline{\text{PID}}$ と続き、CRC（ハンドシェーク・スペシャルには CRC がありません）、EOP で終わります。SYNC（synchronization）は、パケットの先頭を表し、00000001 の固定値が送信されます。パケットの受信側はこの SYNC を用いて同期をとります。PID は、パケットの種類を示すもので、種類ごとにそれぞれ固有の値を持っています。$\overline{\text{PID}}$ は、各ビットごとにおける PID のバイナリ補数です。PID の補数を持つことにより PID エラーの検出が可能になっています。CRC（Cyclic Redundancy Check）は、各パケットの SYNC、PID、$\overline{\text{PID}}$ 以外を CRC チェックした結果です。

- SOF（Start Of Frame）

  SOFとは、ホストコントローラが1msごとに発行するパケットです。SOFが発行されてから次のSOFが発行されるまでの間を「フレーム」と呼びます。SOFは、デバイス全体の同期をとるために使用されるほか、定期的にデータの転送を行う必要があるアイソクロナス転送（**「**2.6.4 **アイソクロナス転送」**参照）の基準およびロースピードデバイス向けのサスペンド防止用信号（キープアライブ信号：SOFの受信によりハブ・ルートハブにて生成）の生成にも使用されます。分類上はトークンパケットに属していますが、使用方法が上記の様に他のトークンと異なるため、分離してあります。

図 2.13　SOF **パケット**

- **トークン**

  トークンは、ホストコントローラのみが発行可能で、デバイスに対してコマンドを送る際やデータの向きを知らせるために使用します。トークンパケットには、以下の種類があります。また、デバイス側にてホストコントローラより転送されるデータが自デバイス宛であるかを識別するために用いるアドレス情報、およびデバイスのエンドポイントを識別するエンドポイント情報も含まれています。

  「OUT」トークン

  ホストコントローラが、デバイスに対してデータを送る前に発行します。

  「IN」トークン

  ホストコントローラが、デバイスに対してデータの送信を要求する際に発行します。

  「SETUP」トークン

  コントロール転送でコマンドを送る際に発行します。コントロール転送の詳細は、**「**2.6.2 **コントロール転送」**をご覧ください。

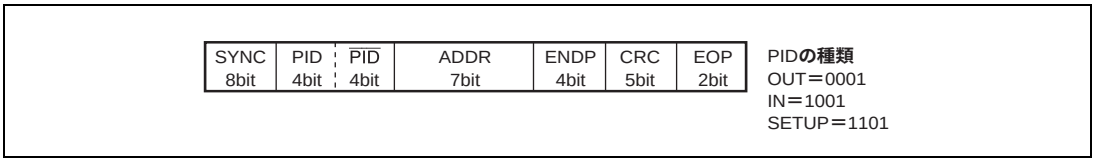


図 2.14　**トークンパケット**

- **データ**

  データパケットは、ホストコントローラや、デバイスがデータを送信する場合に使用します。データパケットにはPIDがDATA0とDATA1の2種類あり、これらを交互に送信することによりデータの欠落を検出でき、信頼度を高めています（アイソクロナス転送の場合はDATA0固定となります）。

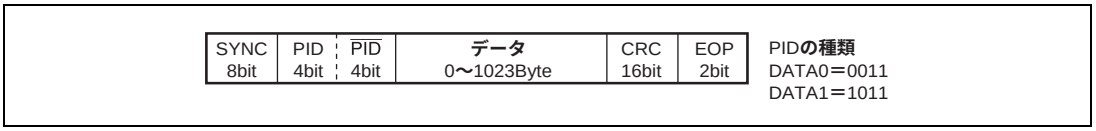


図 2.15　**データパケット**

- **ハンドシェーク**

ハンドシェークは、受信側がデータを正常に受け取れたかどうかを送信側に知らせるために使用します。ハンドシェークには、以下の種類があります（アイソクロナス転送の場合、ハンドシェークは発行されません）。

「ACK」

ホストコントローラおよびデバイスが、データパケットを正常に受信できた際に発行します。

「NAK」

NAKはデバイスからホストに対して発行するもので、次のような場合に発行します。

- ホストからOUTトークンパケットおよびデータパケットを受信したが、エンドポイントがいっぱいでデータを受信できない場合。
- ホストからINトークンパケットを受信したが、送信すべきデータが準備できていない場合。

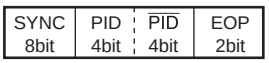
ホストコントローラがNAKを受信した際、アウトトランザクションの場合ではOUTトークンおよび受信できなかったデータを、またイントランザクションの場合ではINトークンを後でホストコントローラが再発行します。ホストコントローラは、常にデータパケットを送受信できるように定義されているため、ホストコントローラがデバイスに対してNAKを返すことはありません。

「STALL」

STALLはデバイスが何らかの理由でエラー状態になり、ホストに対して介入を要求する場合に発行します。

「無応答」（ハンドシェイクパケット発行なし）

PIDにエラーがあった場合や、CRCの計算結果が不一致であった場合はハンドシェークを行わず、無応答となります。ホストコントローラやデバイスはデータを送信した後、一定時間（16～18ビット時間）以上無応答であった場合はタイムアウトとなり、通信エラーが発生したと認識します。その後、ホストコントローラはエラーと認識されたトークンおよびデータを再発行します。



PIDの種類
ACK＝0010
NAK＝1010
STALL＝1110

【注】「無応答」の場合、パケットは発行されません

図 2.16 **ハンドシェイクパケット**

- **スペシャル**

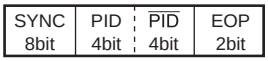
  スペシャルパケットには、PRE（PREAMBLE）パケットが定義されています。このパケットは、後に続いてロースピードの転送が行われることをデバイスに示すために使用します。

  ロースピードデバイスに対するフルスピードデータの転送はエラーの原因となります。

  そこで、このPREパケットを用いてエラーの回避を行います。

  ロースピードデバイスに対しては、ハブ（ルートハブも含む）によりフルスピードデータがフィルタリングされ、転送されないようになっていますが、ハブがPREパケットを受信するとフィルタリングを解除し、以降にホストコントローラ側より転送されるロースピードデータをロースピードデバイスに対して転送開始します。

  なお、ロースピードデータはフルスピードデバイスにも転送されますが、ロースピードデータは有効なフルスピードのPIDを生成できないため、フルスピードデバイスがロースピードデータによって誤った動作を起こすことはありません。



PIDの種類
PRE＝1100

【注】このパケットの後にロースピードデータが続きます。

図 2.17　**スペシャルパケット**

2.6.2　コントロール転送

コントロール転送は、デバイスに対してコマンドを発行する際に利用します。また、ホストコントローラにデバイスを接続した際、最初に行われる転送でもあります。この場合ホストコントローラはデバイスの情報を取得するためにコントロール転送を使用します。したがって、フルスピードデバイス、ロースピードデバイスに関わらず、すべてのデバイスがこの転送方式をサポートする必要があります。

コントロール転送は、「セットアップステージ」「データステージ」「ステータスステージ」に分類することができます。

【注】　以下からの各種転送方法の説明において、パケットの右部にパケットの送信側を略称で記載してあります。（H）はホストコントローラ側、（D）はデバイス側となります。

- **セットアップステージ**

  セットアップステージは、コントロール転送の最初のステージです。セットアップステージでは、ホストコントローラがデバイスに対してコマンドを発行し、送信または受信する内容の指示を与えます。このコマンドにしたがって、デバイスはホストコントローラに送信するデータの準備、またはホストコントローラからデータを受信する準備を行います。

  コントロール転送のセットアップステージは、セットアップトランザクションで構成されています。セットアップトランザクションのデータパケットの容量は、必ず8Byteです。このデータパケットに、ホストコントローラはコマンドを格納します。

  また、データパケットのPIDは必ずDATA0になります。セットアップトランザクションのハンドシェイクパケ

ットは、デバイスがホストに対して送信するパケットで、ここでは常にACKを返す必要があります。セットアップトランザクションでは、NAKもしくはSTALLを返すことは規格で禁止されています。したがって、デバイスは常にセットアップトランザクションを受信する準備をしておく必要があります。

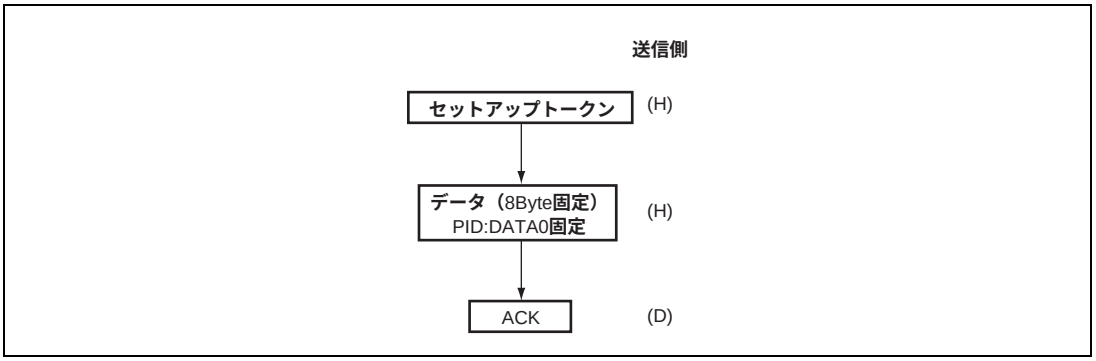


図 2.18　**セットアップステージ**

- **データステージ**

  データステージでは、デバイスはセットアップステージで受信したコマンドの命令に従い、送られてくるデータの受信もしくは、送るべきデータの送信を繰り返します。ただし、データステージの途中でデータの方向が変わることはありません。

  イン方向のデータステージにおいて、デバイス側から送信すべきデータが終了した場合は、ショートパケット（デバイスごとに指定されている最大データサイズよりバイト数が少ないデータパケット）または0バイトデータパケットを用い、ホストコントローラに送信終了を知らせます。

  コマンドによっては、受信すべきデータまたは、送信すべきデータ自体がない場合があります。この場合、データステージそのものが省略されます。

  また、データが繰り返して送信／受信される場合、データパケットのPIDがDATA1→DATA0→DATA1…とトグルします。

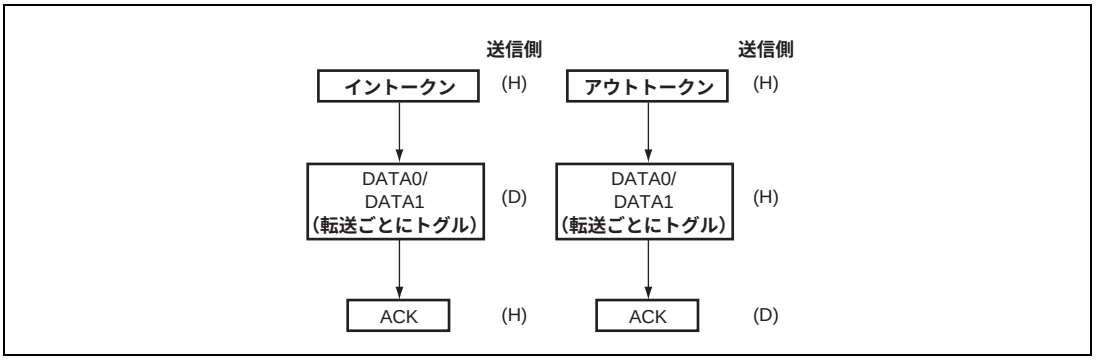


図 2.19　**データステージ（左：イン、右：アウト）**

- **ステータスステージ**

  ステータスステージは、データステージ（データステージなしの場合はセットアップステージ）と反対方向のトークンを送信することで開始されます。たとえば、データステージでイントークンが発行され、デバイスからホストコントローラへデータの転送が行われていた場合、ステータスステージはアウトトークンが発行されることにより開始されます。このように、データステージはデータの向きが反転することによって終了します。

  以下**図**2.20に示すように、ステータスステージはイン方向のデータステージ・アウト方向のデータステージ・データステージなしにより3種類のパターンがあります。

  なお、本ステージにてトークンが送信された後に続くデータパケットには、DATA1のPIDを持つ0Byteデータ長のパケットを使用します。

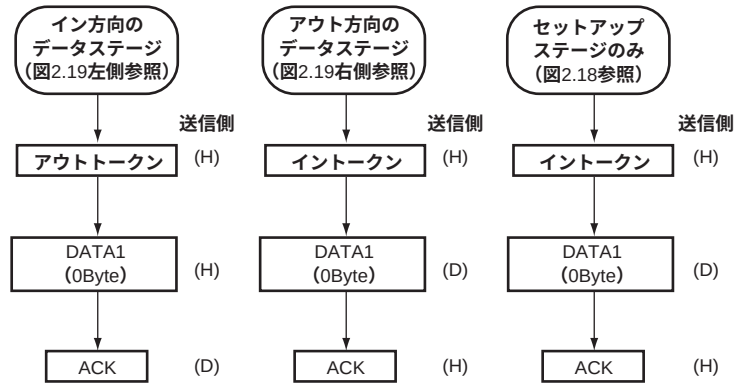


図 2.20　**ステータスステージ**

また、向きが反転することによりステータスステージに入る理由は、ホストコントローラがセットアップステージのコマンドで要求したデータをすべて受信、もしくは送信する前であってもデータステージを終了させることができるように定義されているためです。

図 2.21 にイン方向のデータステージを持つ、コントロール転送の例を示します。セットアップステージで、ホストコントローラが 32Byte のデータを要求したとします。セットアップステージ終了後、ホストコントローラはイントークンを発行します。この命令に従いデバイスが 8Byte（最大パケットサイズが 8Byte の場合）のデータを送り、ホストコントローラは ACK を発行したとします。この時点でデバイスが送信したデータは 32Byte 中 8Byte です。ホストコントローラは、更にデータが必要なときは再びイントークンを発行します。しかし、これ以上データが必要でない場合、ホストコントローラはアウトトークンを発行します。発行されたアウトトークンによってデータの向きが変更されたため、このタイミングでステータスステージに入り、コントロール転送が終了します。

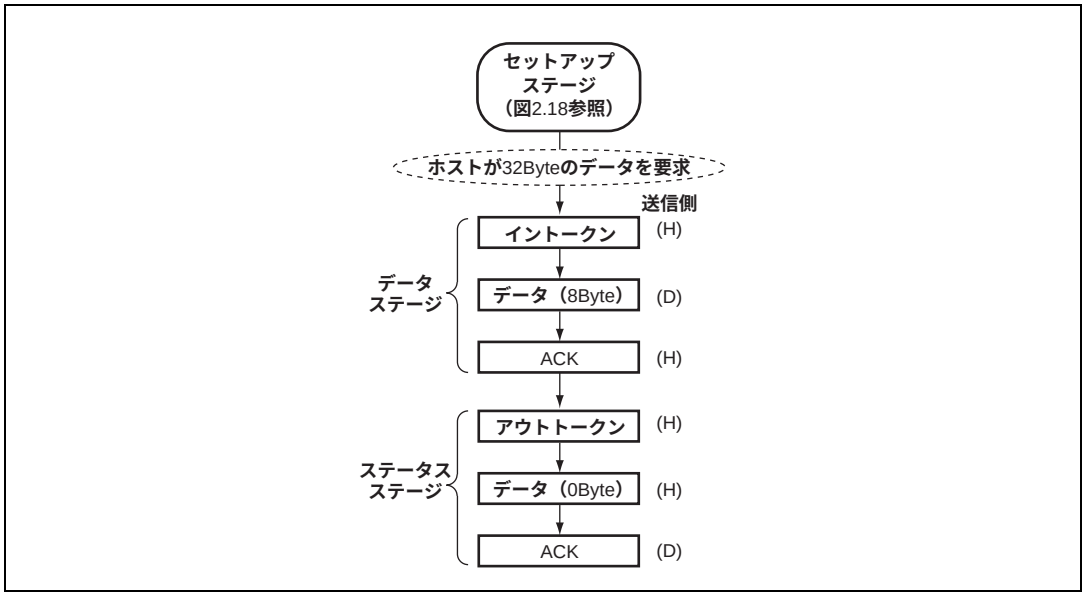


図 2.21　**データステージの中断**

2.6.3　バルク転送

バルク転送は、時間的制約がない大量のデータを、エラーなく転送する場合に使用します。データの転送速度は保証されませんが、データの内容は保証されます。データに誤りがあった場合（CRC 不一致等）、受信側はハンドシェークを発行しません。ACK が返信されなかった場合、送信側はそのデータを再転送します。FIFO に空きがない場合や、送信すべきデータの準備ができていないときには NAK が発行されます。1 回に転送可能なデータ量は、MAX パケットサイズ内で指定することができます。なお、本転送はロースピードデバイスで使用することはできません。

ホストコントローラからイントークンが発行された場合（**図** 2.22 左側）、デバイスからデータが送信され、ホストコントローラからハンドシェークが発行されます。

ホストコントローラからアウトトークンが発行された場合（**図** 2.22 右側）、ホストコントローラからデータが送信され、デバイスからハンドシェークが発行されます。

また、バルクイン／アウト双方において、データ送信／受信が繰り返されるごとにデータパケットの PID が DATA0→DATA1→DATA0…とトグルします。

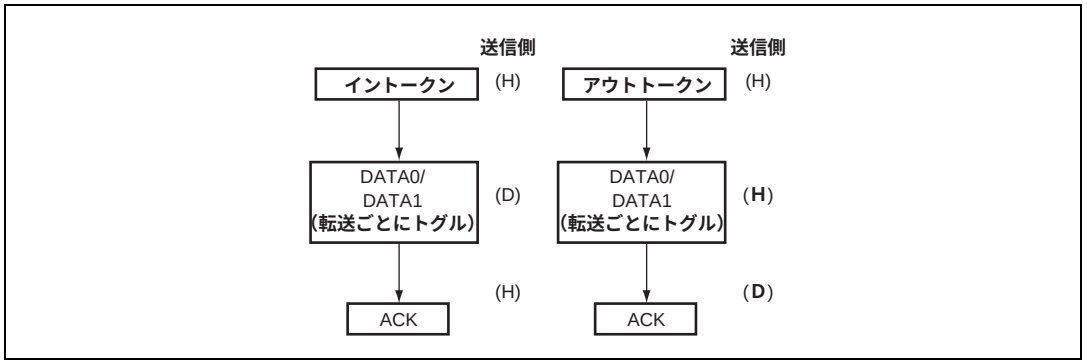


図 2.22 **バルク転送（左：イン、右：アウト）**

2.6.4 アイソクロナス転送

アイソクロナス転送は、音声や動画等の連続したデータを転送する際に使用します。アイソクロナス転送は、1 フレーム（1ms）に 1 回の割り合いで必ずデータ転送が発生するよう優先的にスケジューリングされます。しかし、SOF パケットからのオフセット値は保証されていません。つまり、1 回目の転送はフレームの最後で発生し、次の転送はフレームの最初で行われる可能性もあります。デバイスには、このような状況にも対応することが要求されます。

なお、本転送はロースピードデバイスで使用することはできません。

図 2.23 に示すように、アイソクロナスのトランザクションには、ハンドシェイクパケットがありません。

したがって、転送されたデータにエラーがあったとしても、バルク転送のように再送されません。データパケットの容量は、最大 1023Byte まで設定することができます。

また、データパケットの PID は DATA0 固定です（PID のトグルは行いません）。

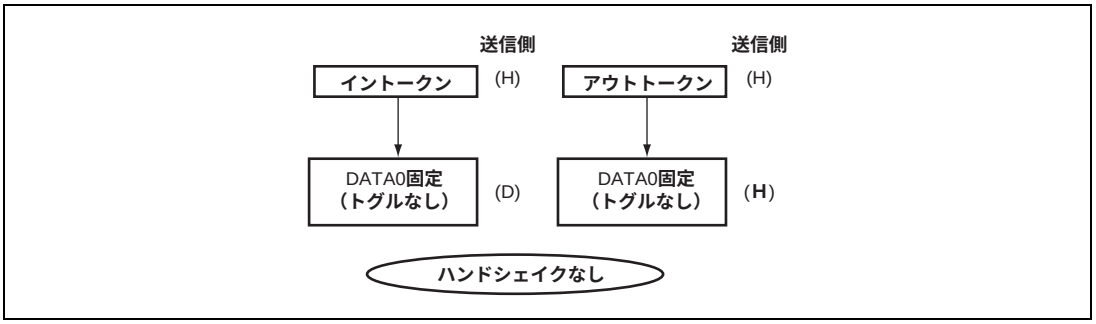


図 2.23 **アイソクロナス転送（左：イン、右：アウト）**

2.6.5 インタラプト転送

インタラプト転送は、決められた周期でホストコントローラがデバイスに対してトランザクションを発生させます。デバイスは、トランザクションが発生する周期をホストコントローラに指定することができます。その周期は 1～255 フレームごとに指定可能です。ホストコントローラは、少なくとも指定された周期に 1 回はイントランザクションを起動します。指定以下の周期でデバイスにアクセスすることはありませんが、指定以上の周期でアクセスする可能性はありますので、注意が必要です（USB1.0 でのインタラプト転送は IN 方向の搬送のみでしたが、USB1.1 でのインタラプト転送は OUT の方向の搬送も追加され、IN、OUT 共にサポートしています）。

本転送はフルスピード／ロースピードの双方で使用可能です。データパケットの容量は、フルスピードデバイスで最大 64Byte まで、ロースピードデバイスで最大 8Byte まで指定できます。また、データ受信が繰り返されるごとにデータパケットの PID が DATA0→DATA1→DATA0…とトグルします。

インタラプトイン転送は、ホストコントローラがイントークンを発行し、デバイス側に送信すべきデータが存在する場合、**図** 2.24a（左）のようにデバイス側からデータパケットを転送します。イントークン発生時、デバイス側で送るべきデータが存在しない場合は、**図** 2.24a（右）に示すように、データパケットを送らず NACK パケットのみ発行します。

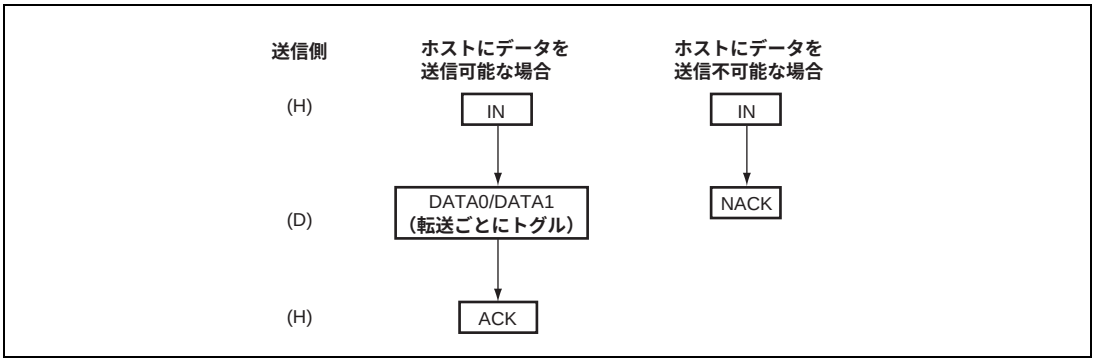


図 2.24a **インタラプト転送**

インタラプトアウト転送は、ホストコントローラがアウトトークンに続きデータを送信します。デバイス側がデータを受信した場合、**図** 2.24b（左）のようにデバイス側から ACK パケットを転送します。

ホストコントローラがアウトトークンに続きデータを送信したが、デバイス側で受信できない場合は、**図** 2.24b（右）に示すように、ACK パケットを送らず NACK パケットを発行します。

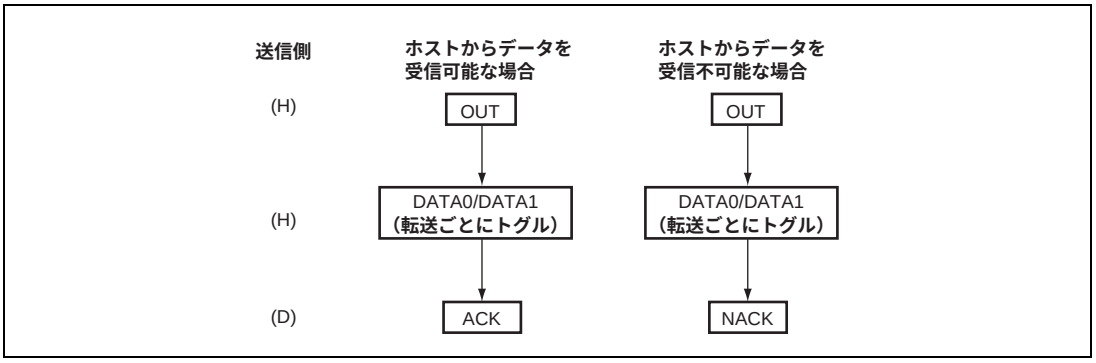


図 2.24b　**インタラプトアウト転送**

2.7　USB デバイスフレームワーク

USB ではプラグアンドプレイを実現するために、USB ケーブルを接続してからコンフィギュレーションが終了するまでの手順が細かく決められています。この章では、その手順について説明します。

2.7.1　デバイスのステート

USB のデバイスには、**図** 2.25 に示すステートがあります。デバイスが構成（コンフィギュレーション）ステートに遷移しているときのみ、ユーザはデバイスを使用することができます。

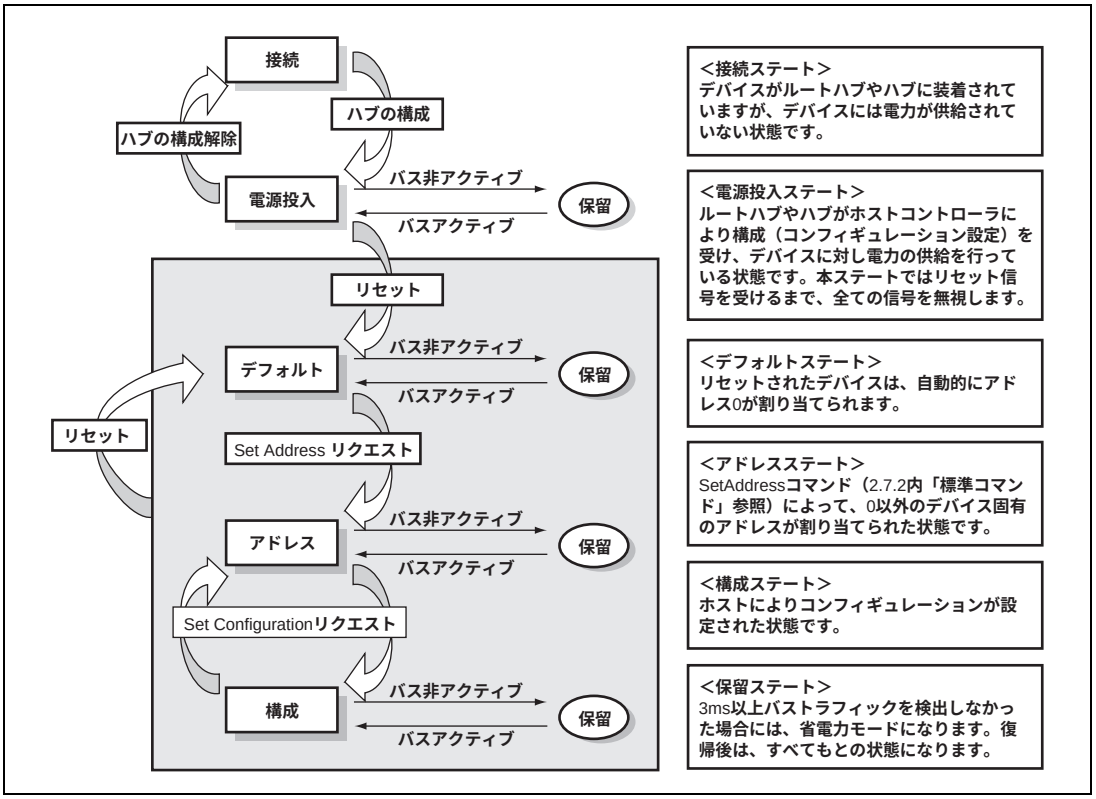


図 2.25　USB **デバイスのステート**

2.7.2　デバイスリクエスト

デバイスをコンフィギュレーションステートに遷移させるためには、ホストコントローラが発行するコマンドに、デバイスが返答する必要があります。ホストコントローラが発行するコマンドは、デバイスリクエストと呼ばれ USB の規格でフォーマットが決められています。ホストコントローラは、デバイスリクエストをコントロール転送のセットアップステージで発行します。

デバイスリクエストは、以下のように 3 種類に分類できます。

- **標準コマンド**

  USBの規格で定義されているもので、原則としてすべてのデバイスがサポートする必要があります。以下、**表** 2.5に標準コマンドの一覧を示します。

  なお、標準コマンドの詳細につきましては規格書をご覧ください。

表 2.5　標準コマンド一覧

コマンド名	機能	データステージ	データステージ方向
Clear_Feature（Endpoint_Halt）	エンドポイントのストールを解除する	無	
Clear_Feature（Device_Remote_Wakeup）	デバイスのリモートウェイクアップを解除する	無	
Get_Configuration	コンフィギュレーション情報を得る	有	イン
Get_Descriptor（Device）	デバイスディスクリプタ情報を得る	有	イン
Get_Descriptor（Config）	コンフィギュレーションディスクリプタ情報を得る	有	イン
Get_Descriptor（String）	ストリングディスクリプタ情報を得る	有	イン
Get_Interface	インタフェース情報を得る	有	イン
Get_Status（Device）	デバイスステータス情報を得る	有	イン
Get_Status（Interface）	インタフェースステータス情報を得る	有	イン
Get_Status（EndPoint）	エンドポイントステータス情報を得る	有	イン
Set_Address	デバイスアドレスの設定	無	
Set_Descriptor（Device）	デバイスディスクリプタを設定	有	アウト
Set_Descriptor（Config）	コンフィギュレーションディスクリプタを設定	有	アウト
Set_Descriptor（String）	ストリングディスクリプタを設定	有	アウト
Set_Configuration	コンフィギュレーションを設定	無	
Set_Feature（EndPoint_Halt）	エンドポイントをストール状態に設定	無	
Set_Feature（Device_Remote_Wakeup）	デバイスをリモートウェイクアップ状態に設定	無	
Set_Interface	インタフェースを設定	無	
Sync_Frame	アイソクロナス転送時、エンドポイントに特定のフレーム番号を通知（特定番号が必要時のみ）	有	アウト

- **クラスコマンド**

  ハブ以外のクラスコマンドは、企業等のグループによって策定され、USB-IF（USB Implementers Forum）によって認可されます。クラスには、オーディオクラス、コモンクラス、HID（Human Interface Device）クラス、プリンタクラス等があります。

- **ベンダコマンド**

  ベンダコマンドは、デバイス設計者が自由に定義することができます。しかし、そのフォーマットは他のコマンドと同一にする必要があります。

2.8 ディスクリプタ

USB デバイスは、デバイス自身の種類・特性・属性を示す「ディスクリプタ」情報を持っています。ホストコントローラがディスクリプタ情報を得ることにより、どのようなデバイスがバスに接続されているかを認識することが可能となっています。

標準 USB デバイスの場合、「デバイス」「コンフィギュレーション」「インタフェース」「エンドポイント」の各ディスクリプタを持っています。

以下、**表** 2.6、**表** 2.7、**表** 2.8、**表** 2.9 に各ディスクリプタを示します。

表 2.6 **デバイスディスクリプタ**

フィールド	サイズ（バイト）	内　　容
bLength	1	**ディスクリプタのサイズ**（0x12 **で固定**）
bDescriptorType	1	**ディスクリプタのタイプ**（0x01 **で固定**）
bcdUSB	2	USB **のバージョンを** BCD **で表現**
bDeviceClass	1	**クラスコード** 0：**クラスなし** 0xFF：**ベンダクラス** 1～0xFE：**特定クラス**
bDeviceSubClass	1	**サブクラスコード**
bDeviceProtocol	1	**プロトコルコード** 0：**固有プロトコル使用せず** 0xFF：**ベンダ固有プロトコル**
bMaxPacketSize0	1	**エンドポイント** 0 **の最大パケットサイズ**
idVendor	2	**ベンダ** ID（USB-IF*が**製造メーカごとに割り当て**）
idProduct	2	**プロダクト** ID（**製造メーカがデバイスごとに割り当て**）
bcdDevice	2	**デバイスのバージョンを** BCD **で表現**
iManufacturer	1	**メーカ名を表すストリングディスクリプタへのインデックス**
iProduct	1	**デバイス名を表すストリングディスクリプタへのインデックス**
iSerialNumber	1	**デバイスの製造番号を表すストリングディスクリプタへのインデックス**
bNumConfigurations	1	**構成可能数**

【注】* USB Implementers Forum

表 2.7　コンフィギュレーションディスクリプタ

フィールド	サイズ（バイト）	内　　容
bLength	1	ディスクリプタのサイズ（0x09 で固定）
bDescriptorType	1	ディスクリプタのタイプ（0x02 で固定）
wTotalLength	2	全ディスクリプタの長さ
bNumInterface	1	このディスクリプタが持つインタフェース数
bConfigurationValue	1	Set_Configuration で、このディスクリプタを選択するための引数値（1 以上）
iConfiguration	1	ストリングディスクリプタへのインデックス
bmAttributes	1	デバイスの電源 ビット 7：予約（1） ビット 6：自己電源 ビット 5：リモートウェイクアップ ビット 4～0：予約（0）
MaxPower	1	最大バス電流消費量を 2mA 単位で指定

表 2.8　インタフェースディスクリプタ

フィールド	サイズ（バイト）	内　　容
bLength	1	ディスクリプタのサイズ（0x09 で固定）
bDescriptorType	1	ディスクリプタのタイプ（0x04 で固定）
bInterfaceNumber	1	コンフィギュレーションの中で、このインタフェースを表す 0 ベースのインデックス番号
bAlternateSetting	1	Set_Interface で、代替設定を選択するための引数値
bNumEndpoints	1	デバイスの持つエンドポイント数（エンドポイント 0 を除く）
bInterfaceClass	1	クラスコード 0：クラスなし 0xFF：ベンダクラス 1～0xFE：特定クラス
bInterfaceSubClass	1	サブクラスコード
bInterfaceProtocol	1	プロトコルコード 0：固有プロトコル使用せず 0xFF：ベンダ固有プロトコル
iInterface	1	このインタフェースを表すストリングディスクリプタへのインデックス

表 2.9 エンドポイントディスクリプタ

種類	サイズ（バイト）	内　　容
bLength	1	ディスクリプタのサイズ（0x07 で固定）
bDescriptorType	1	ディスクリプタのタイプ（0x05 で固定）
bEndpointAddress	1	エンドポイントアドレス ビット 7：方向（0：OUT、1：IN） ビット 6～4：予約（0） ビット 3～0：エンドポイント番号
bmAttributes	1	エンドポイントの転送方法 ビット 7～2：予約（0） ビット 1～0：転送方法（0：コントロール、1：アイソクロナス、2：バルク、3：インタラプト）
wMaxPacketSize	2	最大パケットサイズ
bInterval	1	1ms 単位でポーリング間隔を指定 アイソクロナス転送の場合には 1 を指定 バルク、コントロール転送では無視される

3. 開発環境

この章では、本システムの開発に使用した開発環境について説明します。本システムの開発には、以下のデバイス（ツール）を使用しました。

- H8S/2215 Solution Engine（以下MS2215CP　型名MS2215CP01_C/S）日立超LSIシステムズ社製
- Solution Engineシングルチップマイコンベースボード（以下ベースボード　型名MSSCBB01）日立超LSIシステムズ社製
- E6000（型名HS2214EPI61H）エミュレータ　日立製作所製
- H8S/2215 シリーズTFP120用ユーザシステムインターフェイスケーブル（以下H8S/2215ユーザケーブル　型名HS2215ECN61H）　日立制作所製
- ISA（またはPCI/PCMCIA）スロット搭載のPC（Windows95/98）
- USBホスト用PC（Windows 2000/Windows Millennium Edition またはMac OS9）
- パラレルポート搭載プリンタ
- USBケーブル
- パラレルケーブル
- Hitachi Debugging Interface（以下HDI）日立製作所製
- Hitachi Embedded Workshop（以下HEW）日立製作所製

3.1 ハードウエア環境

図 3.1 に各デバイスの接続形態を示します。

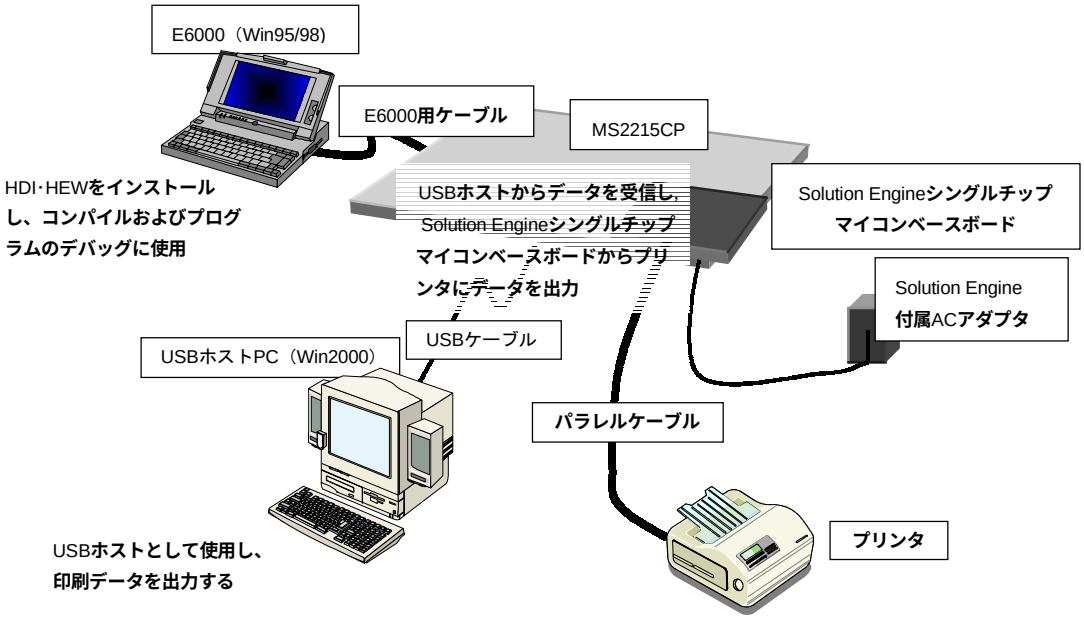


図 3.1 **デバイスの接続形態**

1. MS2215CP

   MS2215CPボードのジャンパピンのいくつかを出荷時の設定から変更する必要があります。電源を投入する前に、これらの設定をよくご確認ください。その他のジャンパピンを変更する必要はありません。

表 3.1 **ディップスイッチの設定**

出荷時	変更後	ディップスイッチの機能
J9 1-2 **ショート**	J9 2-3 **ショート**	EXTAL48 **端子切り換え**
J14 1-2 **ショート**	J14 1-2 **オープン**	SRAM **イネーブル**
J15 1-2 **ショート**	J15 1-2 **オープン**	LED **イネーブル**

2. Solution Engine**シングルチップマイコンベースボード**

   MS2215CPとの接続方法は、ベースボードの取扱説明書をご覧ください。このベースボードはSolution Engineに付属しておりませんので、別途購入していただく必要があります。

   なお、セントロニクスインタフェースコネクタは26Pinピン.コネクタ（CN4）となっています。使用するパラレルケーブルのコネクタ形状が合わない場合は、ベースボード取扱説明書の**「**6.5.2 **セントロニクスインタフェース」**の**「表**6.7 **コネクタ信号配置」**に従い変換コネクタを作成してください。

3. USBホストPC

USBポート搭載の、Windows2000をインストールしたパソコンをUSBホストとして使用します。本システムでは、Windows2000に標準で搭載されているプリンタクラスのデバイスドライバを使用しますので、新たにドライバをインストールする必要はありません。

4. E6000

E6000用PCとE6000エミュレータの接続インタフェースはISAを使用しました。

ISAスロットにE6000I/Fボードを挿入し、接続用のケーブルを介してE6000とE6000I/Fボードを接続してください。次に、H8S/2215ユーザケーブルを用いて、E6000とMS2215CPを接続してください。接続後、HDIを起動してエミュレーションを行います。

3.2 ソフトウエア環境

サンプルプログラムと、今回使用したコンパイラおよびリンカについて説明します。

3.2.1 サンプルプログラム

サンプルプログラムとして必要なファイルは、すべて H8S2215 フォルダ内に収められています。HEW、HDI がインストールされたパソコンに、このフォルダごと移動して頂くと、すぐにサンプルプログラムを使用することができます。フォルダに含まれるファイルを以下**図** 3.2 に示します。

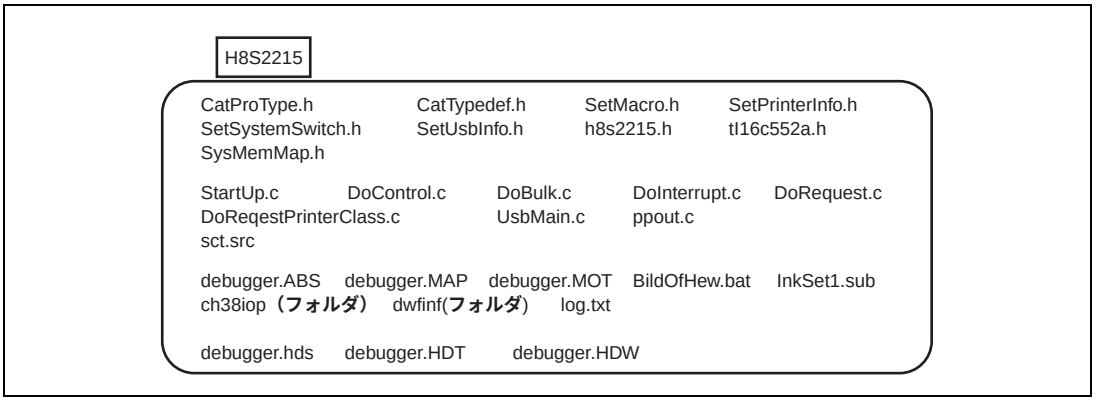


図 3.2 **フォルダに含まれるファイル**

3.2.2 コンパイルおよびリンク

サンプルプログラムのコンパイルおよびリンクは、以下のソフトウエアにより行いました。

- Hitachi Embedded Workshop　Version1.0（release9）　**（以下**HEW**）**

HEW を C:¥Hew にインストール˝した場合、コンパイルおよびリンクの手順は以下のようになります。

まず、コンパイル時に作業用として、Tmp という名前のフォルダを C:¥Hew のフォルダ内に作成してください（**図** 3.3）。

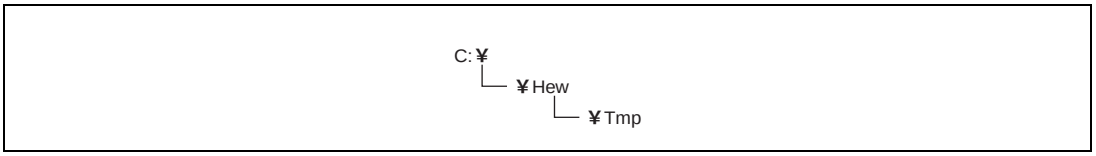


図 3.3　**作業フォルダの作成**

次に、サンプルプログラムが格納されているフォルダ（H8S2215）を、C:¥Usr にコピーしてください（もしくは、任意の場所にコピーし、フォルダ内に含まれている debugger.hds ファイルに記述されている C:¥Usr¥h8s2215 をフォルダをコピーしたパスに変更してください）。この中には、サンプルプログラムと共に BuildOfHew.bat というバッチファイルが含まれています。このバッチファイルでは、パスの設定、コンパイルオプションの指定、コンパイルおよびリンク結果を示すログファイルの指定等を行っています。BuildOfHew.bat を実行すると、コンパイルおよびリンクが行われます。その結果、フォルダ内にはファイル名 debugger.MOT のモトローラ S タイプフォーマットファイルが作成されます。これが実行ファイルとなります。このとき同時にマップファイル debugger.MAP とログファイル log.txt が作成されます。マップファイルにはプログラムのサイズ、および変数のアドレスが示されています。コンパイルの結果（エラーの有無等）はログファイルに記録されます。

【注】*　HEW **を** C:¥HEW **以外にインストールした場合、**BuildOfHew.bat **内の「コンパイラパスの設定」と「コンパイラが使用する環境変数の設定」、**lnkSet1.sub **内の「ライブラリーの指定」を変更する必要があります。この場合、コンパイラパスの設定は** ch38.exe **のパス、コンパイラが使用する環境変数** ch38 **の設定は** machine.h **のフォルダ、**ch38tmp **の設定はコンパイル作業フォルダをそれぞれ指定してください。また、ライブラリーの指定は** c8s26a.lib **のパスを指定してください。**

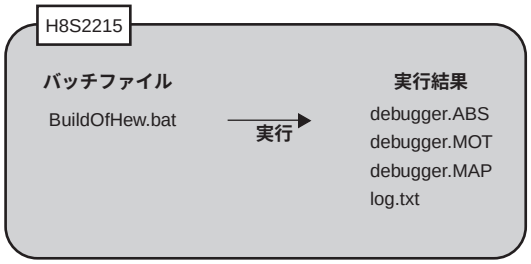
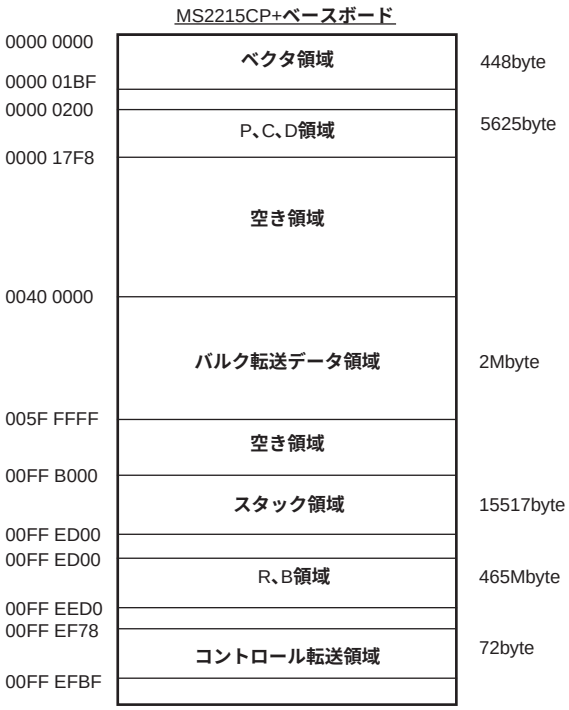


図 3.4　**コンパイル結果**

3.3　プログラムのロードと実行方法

図 3.5 にサンプルプログラムのメモリマップを示します。



【注】 ＊1　**コンパイラバージョン、コンパイル条件、ファームウェアのアップグレード等によりメモリマップは変化する場合があります。**

図 3.5　**メモリマップ**

図 3.5 のように、本サンプルプログラムはベクタ、P、C、D の各領域をエリア 1 内蔵 ROM 領域（E6000 の貸し出しメモリ）上に配置。スタック、B、R、コントロール転送各領域を内蔵 RAM 上に配置。プリントデータを格納する領域を SRAM 上に配置しています。

これらのメモリへの割り付けは、H8S2215 フォルダ内に含まれる lnkSet1.sub で指定します。プログラム配置を変更する場合は、このファイルを変更してください。

3.3.1 プログラムのロード

MS2215CP にサンプルプログラムをロードするには、以下のような手順で行います。

- HDIをインストールしたE6000用PCにE6000を接続してください。
- H8S2215ユーザケーブルでE6000とMS2215CPを接続してください。
- E6000用PCの電源を投入し、起動してください。
- H8S2215フォルダ内のdebugger.hdsを実行してください。

以上の操作で、サンプルプログラムを MS2215CP にロードすることができます。

3.3.2 プログラムの実行

「3.3.1 **プログラムのモード」**でロードしたプログラムを実行するためには、プログラムカウンタ（PC）を設定する必要があります。

メニューバーの View → Register Window を選択し、Registers ウインドを開きます。ウインド内の該当レジスタ（PC）の数値エリアをダブルクリックすると、ダイアログボックスが開きレジスタの値を変更することができます。このダイアログボックスで PC を H'0000 0200 に設定してください。

以上の設定後、メニューバーの Run → Go を選択するとプログラムが実行されます。

3.4 プリントアウトの方法

プログラムを実行した状態で、USB ケーブルのシリーズ B コネクタを MS2215CP に挿入し、反対側のシリーズ A コネクタを USB ホスト PC に接続します。コントロール転送終了後、デバイスマネージャーの USB ホストコントローラの下に USB 印刷サポートが表示され、ホスト PC は MS2215CP をプリンタデバイスとして認識します。

次にプリンタのドライバ*1をインストールします。スタートメニューの設定からプリンタを開き、プリンタの追加をダブルクリックします。セットアップウイザードが開始されますので、ポートの選択で USB001 Virtual Printer Port for USB*2にチェックをしてください。次に、お客様がお使いのプリンタ（メーカー名とプリンタの機種）を指定してください。ウイザード終了の際にテストプリントをして頂くと、ドライバが正常にインストールされた場合は、プリンタからテストプリントが出力されます。

【注】 *1 **本サンプルプログラムではプリンタとの双方向通信をサポートしていないため、必ず** Windows2000 **に標準で付属しているプリンタドライバを使用してください。**

*2 **以前にホスト** PC **にプリンタクラスのデバイスを接続したことがある場合は、異なる数字（**USB002、USB003 **等）になる場合があります。この場合は、最も大きい数字のポートを選択してください。**

4. サンプルプログラム概要

この章ではサンプルプログラムの特長やその構成について説明します。本サンプルプログラムは MS2215CP+ベースボード上で動作し、USB ファンクションモジュールからの割り込みによって USB 転送を開始します。H8S/2215 内蔵モジュールの割り込みのうち、USB ファンクションモジュールに関連する割り込みは、EXIRQ0、EXIRQ1、IRQ6 の 3 種類ですが、本サンプルでは EXIRQ0 のみ使用しています。

本サンプルプログラムの特長を以下に示します。

- コントロール転送を行うことができます。
- バルクアウト転送でホストコントローラからデータを受信することができます。
- バルクイン転送でホストコントローラにデータを送信することができます。
- ベースボードに搭載のTL16C552Aを使用し、プリンタにデータを出力することができます。

4.1 状態遷移図

定常状態において、USB モジュールから割り込みが発生するとこの状態になります。USB 通信状態では、割り込みの種類に応じた転送方式によるデータ転送を行います。本サンプルプルプログラムで使用する割り込みは、USB 割り込みフラグレジスタ 0～3（UIFR0～3）によって示される計 9 種類です。割り込み要因が発生すると、UIFR0～3 の対応するビットに 1 がセットされます。

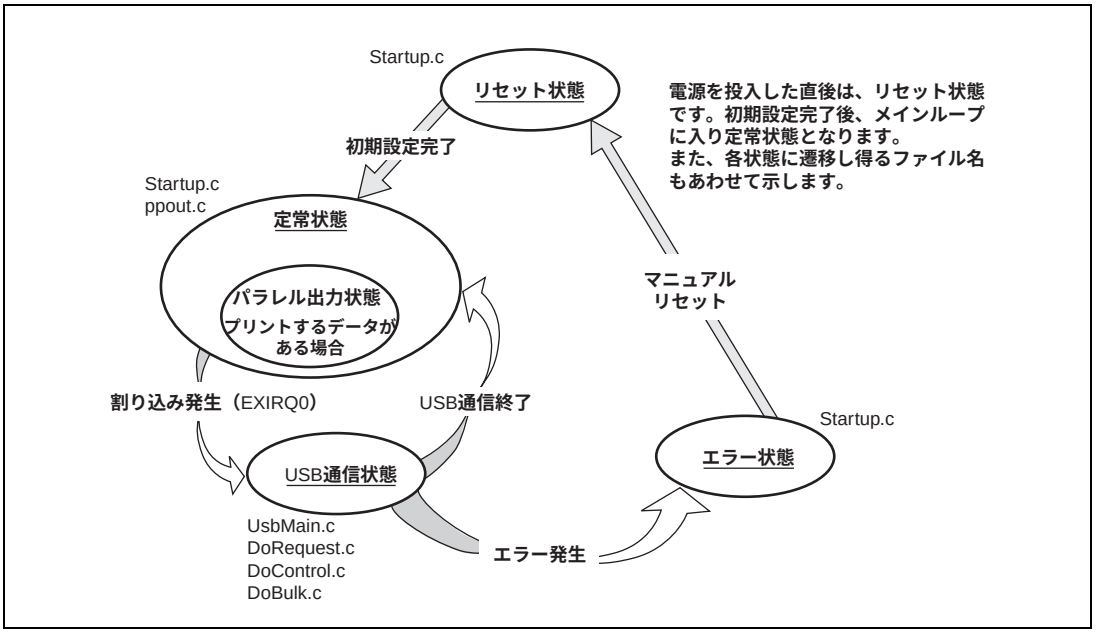


図 4.1　状態遷移図

- **リセット状態**

  パワーオンリセット・マニュアルリセットの際には、この状態になります。リセット状態では、主にH8S/2215の初期設定を行います。

- **定常状態**

  初期設定が完了すると、メインループで定常状態となります。ここでは、ホストからの印刷データの有無を常に監視し、データがあればパラレル出力状態となりプリンタへデータを出力します。

- **USB通信状態**

  定常状態において、USBモジュールから割り込みが発生するとこの状態になります。USB通信状態では、割り込みの種類に応じた転送方式によるデータ転送を行います。本サンプルプログラムで使用する割り込みは割り込みフラグレジスタ0～3（USBIFR0～3）によって示される計9種類です。割り込み要因が発生すると、USBIFR0～3の対応するビットに1がセットされます。

- **エラー状態**

  USB通信状態においてエラーが発生した場合、この状態になります。エラー状態に遷移する場合は、USBの通信内容に問題があります。通信が正常に行われている場合は、エラー状態に遷移することはありません。エラー状態になる場合は、ファームウエアを見直していただく必要があります。エラー状態から復帰させるためには、パワーオンリセット、もしくはマニュアルリセットを実行してください。

4.2 USB 通信状態

USB 通信状態は、転送方式ごとに 3 つの状態に分類することができます（**図** 4.2 参照）。割り込みが発生すると、まず USB 通信状態へと遷移し、さらに割り込みの種類に応じて各転送状態へ分岐します。分岐の方法については**「第 5 章　サンプルプログラムの動作」**で説明します。

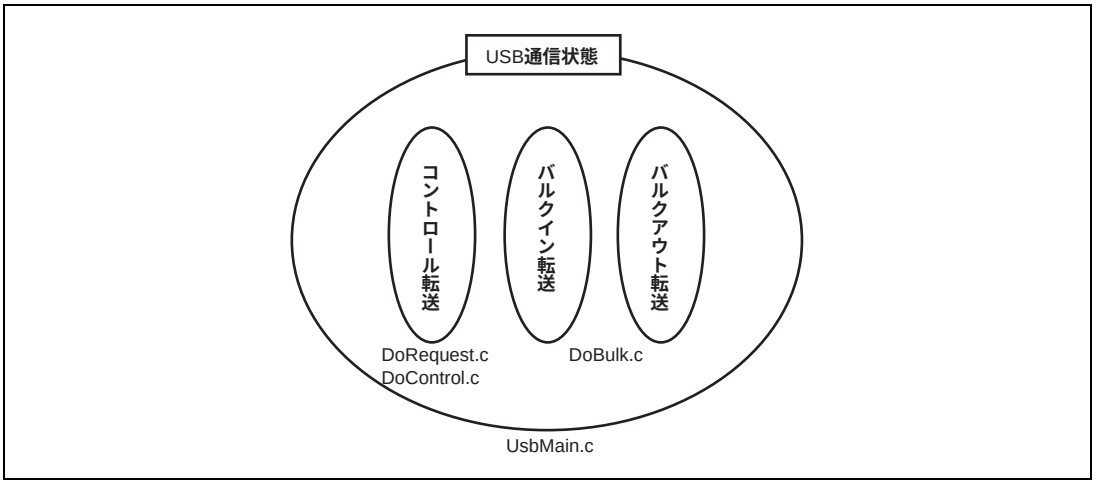


図 4.2　USB **通信状態**

4.3 ファイル構成

本サンプルプログラムは、7 個のソースファイルと 9 個のヘッダーファイルで構成されています。全構成ファイルを**表** 4.1 に示します。各関数は、転送方式または機能ごとに 1 つのファイルにまとめてあります。

表 4.1　**ファイル構成**

ファイル名	主な役割
StartUp.c	**マイコンの初期設定**
UsbMain.c	**割り込み要因の判定** **パケットの送受信**
DoRequest.c	**ホストが発行するセットアップコマンドの処理**
DoRequestPrinterClass.c	**プリンタークラスコマンドの処理**
DoControl.c	**コントロール転送を実行**
DoBulk.c	**バルク転送を実行**
ppout.c	**リングバッファの管理** **プリンタの初期化** **プリンタへのデータ出力**

ファイル名	主な役割
CatProType.h	プロトタイプ宣言
CatTypedef.h	USB ファームウエアで使用する基本の構造体定義
SysMemMap.h	MS2215CP＋ベースボードのメモリマップのアドレス定義
h8s2215.h	H8S/2215 レジスタ定義
tl16c552a.h	TL16C552A のレジスタ定義
SetSystemSwitch.h	システムの動作設定
USBFuncModu.h	USB ファンクションモジュールのレジスタ定義
SetMacro.h	マクロ定義
SetUsbInfo.h	USB 対応に必要な変数の初期設定
SetPrinterInfo.h	Bulk-Only Transport 対応に必要な変数の初期設定

4.4 関数の機能

表 4.2-1 に、各ファイルに含まれる関数と、その機能を示します。

表 4.2-1 UsbMain.c

格納ファイル	関数名	機　　能
UsbMain.c	BranchOfInt	割り込み要因の判定と，割り込みに応じた関数を呼び出す
	GetPacket	ホストコントローラから転送されたデータを、RAM に書き込む
	GetPacket4	ホストコントローラから転送されたデータを、ロングワードサイズで RAM に書き込む。リングバッファ対応版。
	GetPacket4S	ホストコントローラから転送されたデータを、ロングワードサイズで RAM に書き込む。高速版。 （本サンプルプログラムでは使用しません）
	PutPacket	ホストコントローラに転送するデータを USB モジュールに書き込む
	PutPacket4	ホストコントローラに転送するデータを、ロングワードサイズで USB モジュールに書き込む。リングバッファ対応版（本サンプルプログラムでは使用しません）
	PutPacket4S	ホストコントローラに転送するデータを、ロングワードサイズで USB モジュールに書き込む。高速版 （本サンプルプログラムでは使用しません）
	SetControlOutContents	ホストから送られたデータを書き換える
	SetUsbModule	USB モジュールの初期設定
	ActBusReset	バスリセット受信時に FIFO のクリアを行う
	ActBusVcc	USB ケーブル接続、切断時に D+ プルアップと USB モジュールの制御を行う。
	ConvRealn	指定した番地から指定バイト長のデータを読み出す。
	ConvReflexn	指定した番地から指定バイト長のデータを逆順に読み出す。

UsbMain.c では、主に USB 割り込みフラグレジスタによって割り込み要因を判定し、割り込みの種類に応じた関数の呼び出しを行います。また、ホストコントローラとファンクションモジュール間におけるパケットの送受信を行います。

表 4.2-2 StartUp.c

格納ファイル	関数名	機　　能
StartUp.c	SetPowerOnSection	BSC、端子、割り込み割り込みコントローラの設定、各初期化ルーチン呼び出しを行い、メインループへ移行
	_INITSCT	初期値を持つ変数を、RAM のワークエリアにコピー
	InitMemory	バルク通信で使用する RAM 領域をクリア
	InitSystem	USB クロックの設定、システム割り込みマスクの設定

パワーオンリセット、またはマニュアルリセットの際には、StartUp.c の SetPowerOnSection が呼び出されます。ここでは H8S/2215 の初期設定や、コントロール転送、バルク転送に使用する RAM 領域のクリアを行います。

表 4.2-3 ppout.c

格納ファイル	関数名	機　　能
ppout.c	ActPrintOut	バッファの空き容量を監視し必要に応じてバルクアウト転送を一時停止する。 バルクアウト関数を呼び出す。
	LptMain	バッファの空き容量を監視し必要に応じてバルクアウト転送を再開する。 LptPortWrite に引数としてリードポインタを渡す
	LptPortOpen	プリンタの初期化
	LptPortWrite	パラレルポートからデータを出力する

ppout.c では、RAM に格納された印刷データを TL16C552A のレジスタに書き込み、ストローブ信号等を制御してプリンタにデータを出力します。

表 4.2-4 DoRequest.c

格納ファイル	関数名	機　　能
DoRequest.c	DecStandardCommands	ホストコントローラが発行したコマンドをデコードし、そのうち標準コマンドの対応を行う
	DecVenderCommands	ベンダコマンドの対応を行う

コントロール転送時に、ホストコントローラから送られてくるコマンドをデコードし、コマンドに応じた処理を行います。本サンプルプログラムでは、ベンダ ID の値に 045B（ベンダ：日立）を使用しています。お客様にて製品を開発される場合は「USB Implementers Forum」にてお客様のベンダ ID を取得願います。また、ベンダコマンドは使用していないため、DecVenderCommands では何も行っていません。ベンダコマンドを使用する際には、お客様でプログラムを作成願います。

表 4.2-5 DoControl.c

格納ファイル	関数名	機　　能
DoControl.c	ActControl	コントロール転送の、セットアップステージを行う
	ActControlIn	コントロールイン転送（データステージがイン方向の転送）のデータステージとステータスステージを行う
	ActControlOut	コントロールアウト転送（データステージがアウト方向の転送）のデータステージとステータスステージを行う

コントロール転送の割り込み（EP0oTS）が入ると、ActControl がコマンドを取得し、DecStandardCommands でデコードを行います。その後コマンドの種類に応じて、ActControlIn または ActControlOut によってデータステージと、ステータスステージを行います。

表 4.2-6　DoBulk.c

格納ファイル	関数名	機　　能
DoBulk.c	ActBulkOut	**バルクアウト転送を行う**
	ActBulkIn	**バルクイン転送を行う**
	ActBulkInReady	**バルクイン転送の準備を行う**

バルク転送に関する処理を行います。バルクイン転送でのみ ActBulkInReady を使用します。

表 4.2-7　DoRequestPrinterClass.c

格納ファイル	関数名	機　　能
DoRequestPrinter Class.c	DecPrinterClassCommands	**プリンタークラスコマンドの対応を行う**

プリンタークラスコマンドに応じた処理を行います。本サンプルプログラムでは、IEEE1284 デバイス ID を使用していないため、0 を出力します。IEEE1284 デバイス ID を使用する際には、お客様で出力値を設定願います。

図 4.3 に、**表** 4.2 で説明した関数の相関関係を示します。上位側の関数が、下位側の関数を呼び出すことができます。また、複数の関数が同一の関数を呼び出すこともあります。定常状態では、SetPowerOnSection が他の関数を呼び出し、割り込みの発生によって遷移する USB 通信状態では、BranchOfInt が他の関数を呼び出します。**図** 4.2 は、関数の上下関係を示しているもので、関数が呼び出される順序は示していません。関数がどのような順序で呼び出されるかについては、**「第 5 章　サンプルプログラムの動作」**のフローチャートをご覧ください。

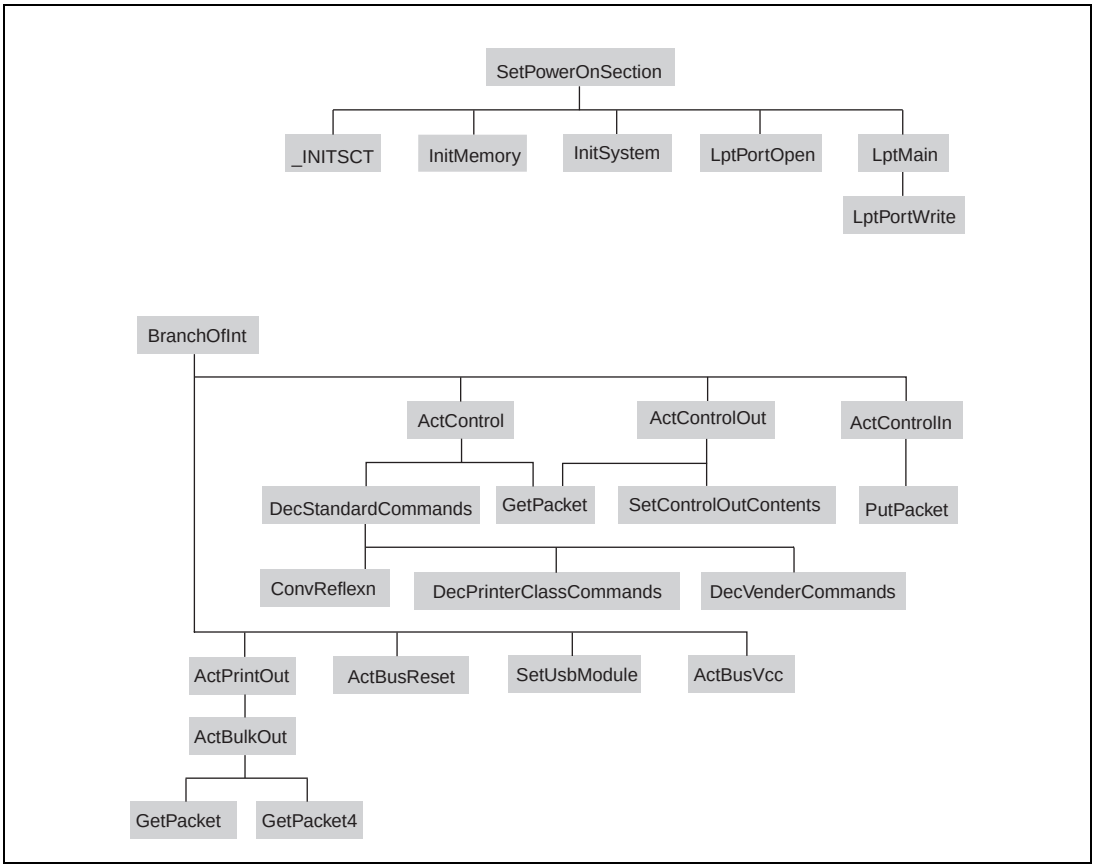

SetPowerOnSection
_INITSCT
InitMemory
InitSystem
LptPortOpen
LptMain
LptPortWrite
BranchOfInt
ActControl
ActControlOut
ActControlIn
DecStandardCommands
GetPacket
SetControlOutContents
PutPacket
ConvReflexn
DecPrinterClassCommands
DecVenderCommands
ActPrintOut
ActBusReset
SetUsbModule
ActBusVcc
ActBulkOut
GetPacket
GetPacket4

図 4.3　**関数の相関関係**

5. サンプルプログラムの動作

この章ではサンプルプログラムの動作を、USB ファンクションモジュールの動作と関連付けて説明します。

5.1 メインループ

マイコンがリセット状態になると、CPU の内部状態と内蔵周辺モジュールのレジスタが初期化されます。次に StartUp.c の関数 SetPowerOnSection が呼び出され、CPU の初期化をします。**図** 5.1 に SetPowerOnSection のフローチャートを示します。

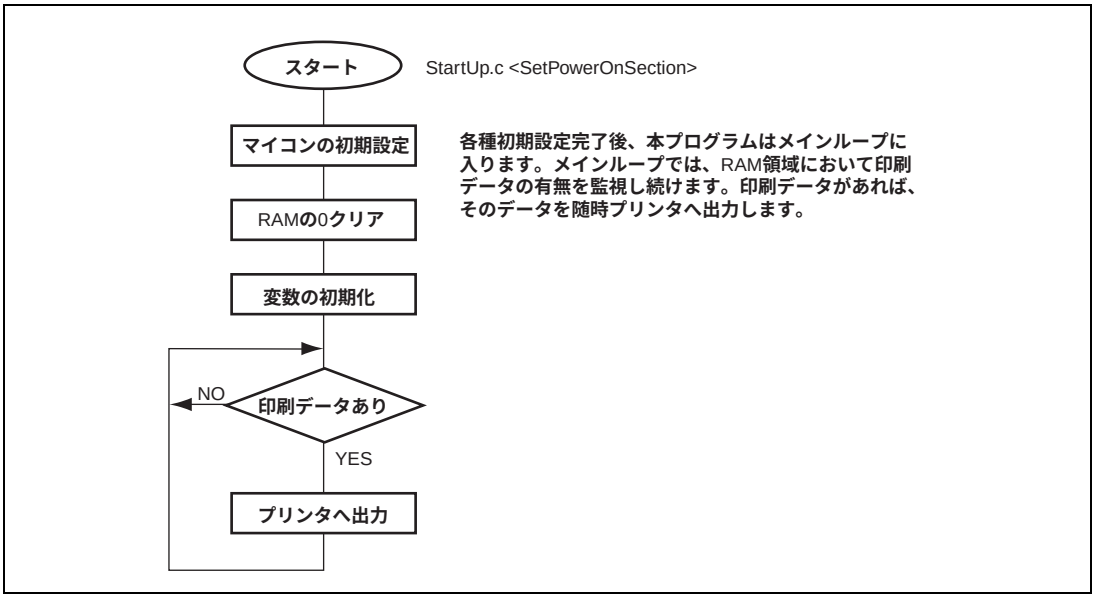
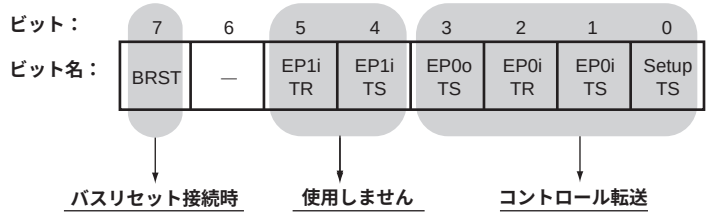


図 5.1 **メインループ**

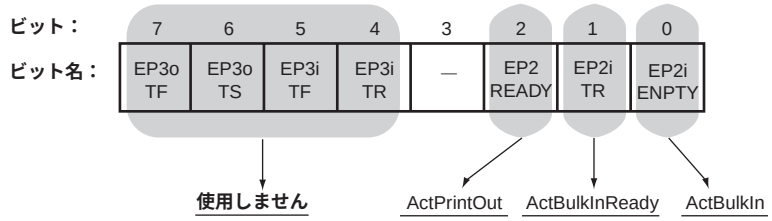
5.2 割り込みの種類

「4.1 **状態遷移図」**で説明したように、本サンプルプログラムで使用する割り込みは割り込みフラグレジスタ 0～3（UIFR0～3）によって示される計 9 種類です。割り込み要因が発生すると、割り込みフラグレジスタの対応するビットに 1 がセットされ、CPU に対して EXIRQ0 割り込みを要求します。サンプルプログラムでは、この割り込み要求によって割り込みフラグレジスタをリードし、それに対応する USB 通信を行います。**図** 5.2 に割り込みフラグレジスタと、USB 通信の関係を示します。

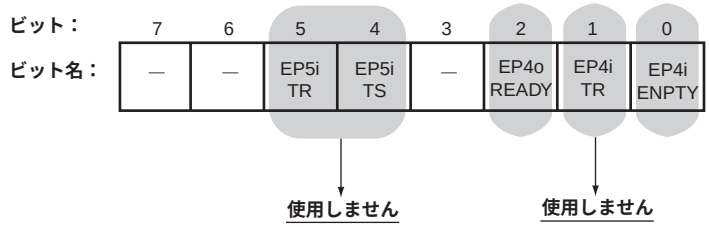
USB割り込みフラグレジスタ0（UIFR0）



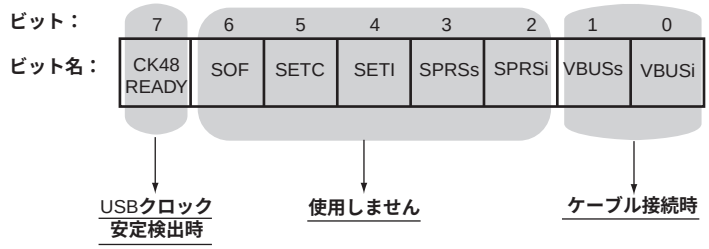
USB割り込みフラグレジスタ1（UIFR1）



USB割り込みフラグレジスタ2（UIFR2）



USB割り込みフラグレジスタ3（UIFR3）



【注意】＊ サンプルプログラムはインタラプト転送、アイソクロナス転送を使用していません。

図 5.2　**割り込みフラグの種類**

5.2.1 各転送への分岐方法

「第 4 章　サンプルプログラム概要」で説明したようにサンプルプログラムでは、USB モジュールからの割り込みの種類によって転送方式を決定しています。各転送方式への分岐は、UsbMain.c の BranchOfInt が実行します。**表** 5.1 に割り込みの種類と、BranchOfInt が呼び出す関数の関係を示します。

表 5.1　**割り込みの種類と分岐先関数**

レジスタ名	ビット	ビット名	呼び出す関数名
UIFR0	7	BRST	ActBusReset
	6	—	—
	5	EP1i TR	—
	4	EP1i TS	—
	3	EP0o TS	ActControlIn、ActControlOut
	2	EP0i TR	ActControlOut
	1	EP0i TS	ActControlIn、ActControlOut
	0	SETUP TS	ActControl
UIFR1	7	EP3o TF	—
	6	EP3o TS	—
	5	EP3i TF	—
	4	EP3i TR	—
	3	—	—
	2	EP2o Ready	ActPrintOut
	1	EP2i TR	ActBulkIn
	0	EP2i EMPTY	ActBulkInReady

レジスタ名	ビット	ビット名	呼び出す関数名
UIFR3	7	CK48 Ready	SetUSBModule
	6	SOF	―
	5	SETC	―
	4	SETI	―
	3	SPRSs	―
	2	SPRSi	―
	1	VBUSs	―
	0	VBUSi	ActBusVcc

EP0i TS と EP0o TS 割り込みは、コントロールイン、アウト転送の両方で使用します。したがって、コントロール転送の方向とステージを管理するために、サンプルプログラムは TRANS_IN、TRANS_OUT、WAIT の 3 つのステートを持っています。詳細は、**「5.6 コントロール転送」**をご覧ください。

H8S/2215 のハードウエアマニュアルには、割り込み発生時の USB ファンクションモジュールの動作と、アプリケーション側の動作概略が示してあります。次節からは、アプリケーション側ファームウエアの詳細を USB の転送方式ごとに説明します。

5.3 USB 動作クロック安定割り込み

この割り込みは、USB モジュールストップ解除後、48MHz の USB 動作クロック安定時間を自動的にカウントした後、発生します。割り込み受信後、EndPoint 構成情報を USB エンドポイントフォーメーションレジスタ（UEPIR00_0～22_4）に書き込み、各割り込みを設定して、USB ケーブル接続待ち状態へ移行します。

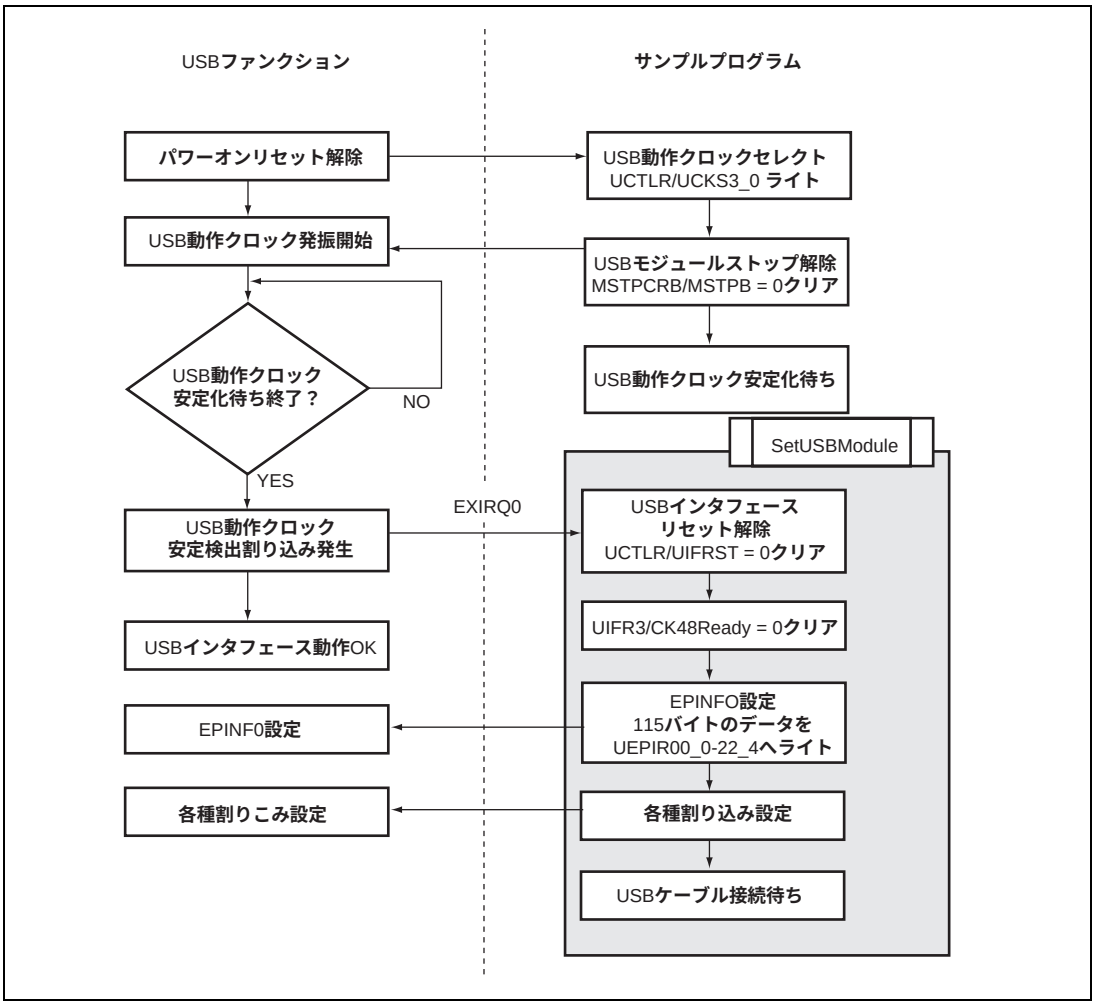


図 5.3 USB 動作クロック安定検出割り込み

5.3.1 エンドポイント構成

H8S/2215 搭載の USB ファンクションモジュールは初期化時にエンドポイント構成をソフトウエアにて設定することが可能です。設定可能な転送内部を次に示します。

- Control転送 ：1系統
- BulkIn転送 ：2系統
- BulkOut転送 ：2系統
- Interrupt In転送 ：2系統
- Isochronous In転送 ：1系統
- Isochronous Out転送 ：1系統

USB エンドポイントインフォメーションレジスタ（以下 UEPIR）を使用し、Control 転送以外は EndPoint 番号、Inferface 番号、Alternate 番号、MaxPacketSize の設定が可能です。

表 5.2 転送タイプと UEPIR との関係

転送タイプ	数	対応する UEPIR
Control 転送	1	00
Interruot In 転送	2	01、22
BulkIn 転送	2	02、20
BulkOut 転送	2	03、21
Isochronous In 転送	1	04、06、08、10、12、14、16、18
Isochronous Out 転送	1	05、07、09、11、13、15、17、19

H8S/2215 ハードウエアマニュアルでは、エンドポイント情報を Bluetooth 規格に設定した場合でエンドポイント番号を説明しています。

本サンプルプログラムで使用するエンドポイント構成と H8S/2215 ハードウエアマニュアル記載のエンドポイント番号を比較したものを**図** 5.4 に示します。

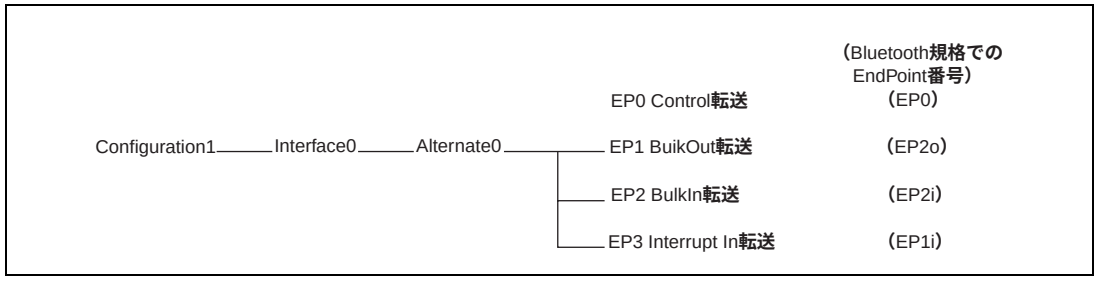


図 5.4 サンプルプログラムの EndPoint 構成

図 5.4 に示す EndPoint 構成を実現する UEPIR00_0～22_4 の設定値を**表** 5.3 に示します。使用しない EndPoint についても必ずダミーデータ（0）をライトしてください。

表 5.3 uepir の設定値

UEPIR	設定数値 (16 進数)	転送タイプ	EP 番号	Interface 番号	Alternate 番号	MaxPaket Size (Byte)
00	00_00_40_00_00	Control	0	0	0	64
01	34_1C_08_00_01	Interrupt In	3	0	0	8
02	24_15_40_00_02	BulkIn	2	0	0	64
03	14_10_40_00_03	BulkOut	1	0	0	64
04	04_1C_00_00_04	Isochronous In	0	0	0	0
05	04_08_00_00_05	Isochronous Out	0	0	0	0
06	04_1C_00_00_06	Isochronous In	0	0	0	0
07	04_08_00_00_07	Isochronous Out	0	0	0	0
08	04_1C_00_00_08	Isochronous In	0	0	0	0
09	04_08_00_00_09	Isochronous Out	0	0	0	0
10	04_1C_00_00_0A	Isochronous In	0	0	0	0
11	04_08_00_00_0B	Isochronous Out	0	0	0	0
12	04_1C_00_00_0C	Isochronous In	0	0	0	0
13	04_08_00_00_0D	Isochronous Out	0	0	0	0
14	04_1C_00_00_0E	Isochronous In	0	0	0	0
15	04_08_00_00_0F	Isochronous Out	0	0	0	0
16	04_1C_00_00_10	Isochronous In	0	0	0	0
17	04_08_00_00_11	Isochronous Out	0	0	0	0
18	04_1C_00_00_12	Isochronous In	0	0	0	0
19	04_08_00_00_13	Isochronous Out	0	0	0	0
20	04_14_00_00_14	BulkIn	0	0	0	0
21	04_10_00_00_15	BulkOut	0	0	0	0
22	04_10_00_00_16	Interrupt In	0	0	0	0

5.4 ケーブル接続時（VBUS）割り込み

USB ファンクションモジュールのケーブルを、ホストコントローラに接続した際に発生します。アプリケーション側はマイコンの初期設定完了後、汎用出力ポートを使用して USB データバスの D+をプルアップします。このプルアップによって、ホストコントローラはデバイスが接続されたことを認識します（**図** 5.5 参照）。

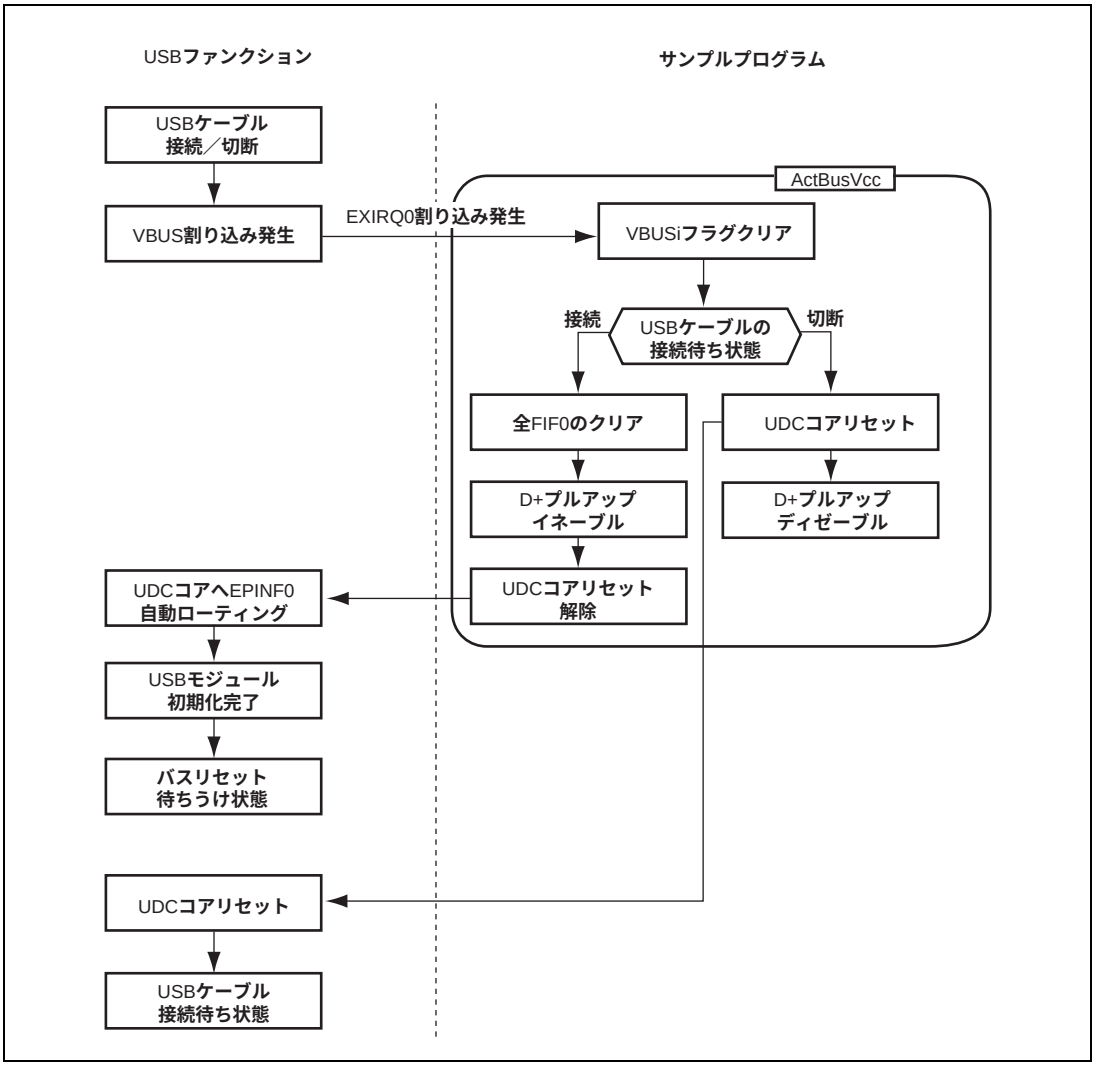


図 5.5 **ケーブル接続時割り込み**

5.5 バスリセット時（BRST）割り込み

ホストコントローラがUSBデータバスにデバイスが接続されたことを認識するとバスリセット信号を出力します。
ホストからのバスリセット信号を受信するとバスリセット割り込みが発生します。

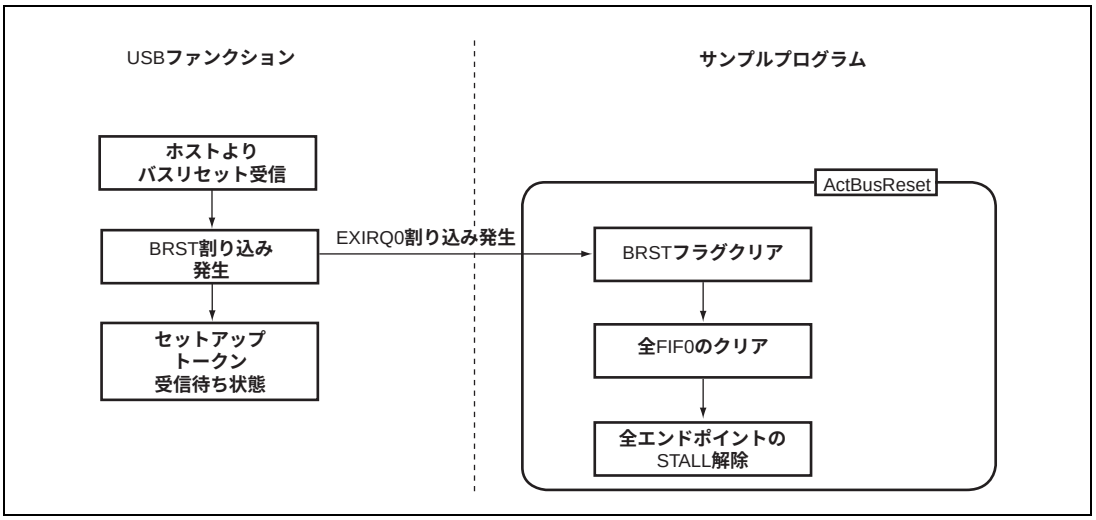


図 5.6 バスリセット割り込み

5.6 コントロール転送

コントロール転送には、割り込みフラグレジスタのビット 0～3 を使用します。コントロール転送は、データステージにおけるデータの向きによって、2 つに分けることができます（**図** 5.7 参照）。

データステージにおいて、ホストコントローラから USB ファンクションへデータ転送する場合がコントロールアウト転送反対の場合がコントロールイン転送です。

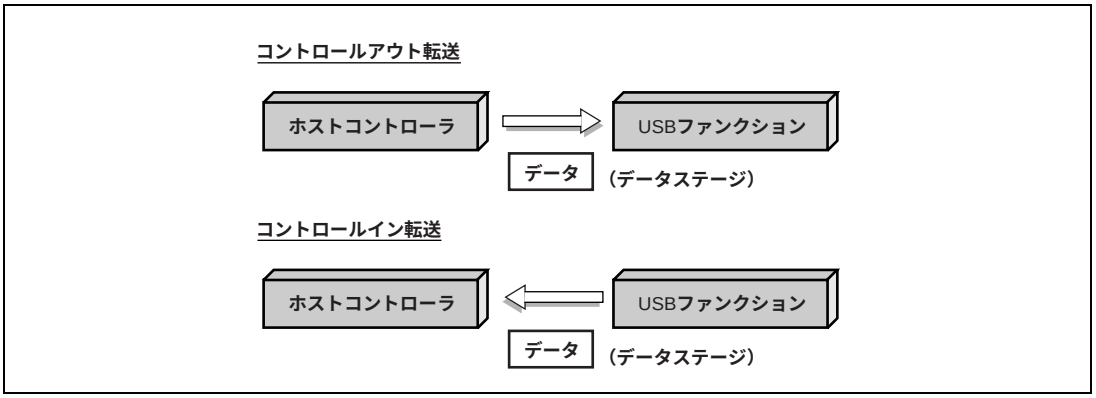


図 5.7 **コントロール転送**

コントロール転送は、セットアップ、データ（ない場合もある）、ステータスの 3 つのステージで構成されます（**図** 5.8）。また、データステージは、複数のバストランザクションで構成されます。

コントロール転送では、データの向きが反転することによってステージが切り替わったことを認識します。したがって同じ割り込みフラグを使用して、コントロールイン転送または、コントロールアウト転送を行う関数を呼び出します（**表** 5.1 参照）。このため、現在イン･アウトどちらのコントロール転送が行われているかをファームウエアがステートによって管理し（**図** 5.8 参照）、適切な関数を呼び出す必要があります。データステージにおけるステート（TRANS_IN、TRANS_OUT）は、セットアップステージで受信するコマンドによって決定します。

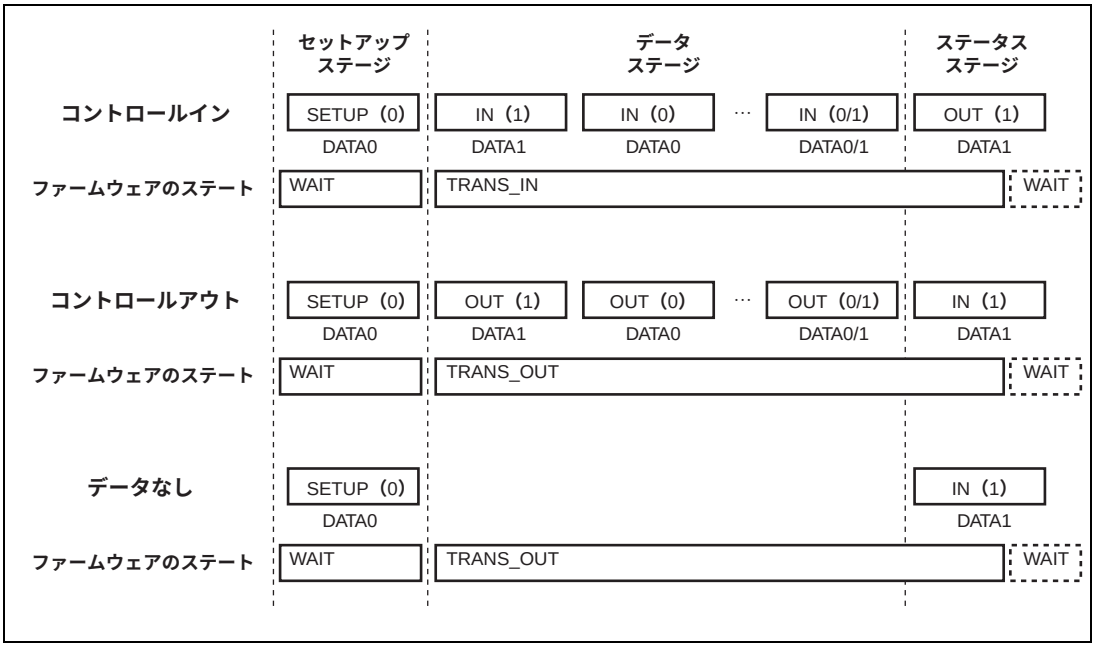


図 5.8 **コントロール転送におけるステージ**

5.6.1 セットアップステージ

セットアップステージでは、ホストとファンクションがコマンドの送受信を行います。コントロールイン転送、コントロールアウト転送共に、ファームウエアのステートは WAIT になります。また発行されるコマンドの種類によって、コントロールイン転送またはアウト転送の区別を行い、データステージにおけるファームウエアのステート（TRANS_IN、TRANS_OUT）を決定します。

- **コントロールインとなるコマンド**

  GetDescriptor （TRANS_IN）標準コマンド

  GetDeviceID （TRANS_IN）クラスコマンド

  GetPortStatus （TRANS_IN）クラスコマンド

- **コントロールアウトとなるコマンド**

  SoftReset （TRANS_OUT）クラスコマンド

図 5.9 にセットアップステージにおけるサンプルプログラムの動作を示します。図の左側は、USB ファンクションモジュールの動作を示しています。

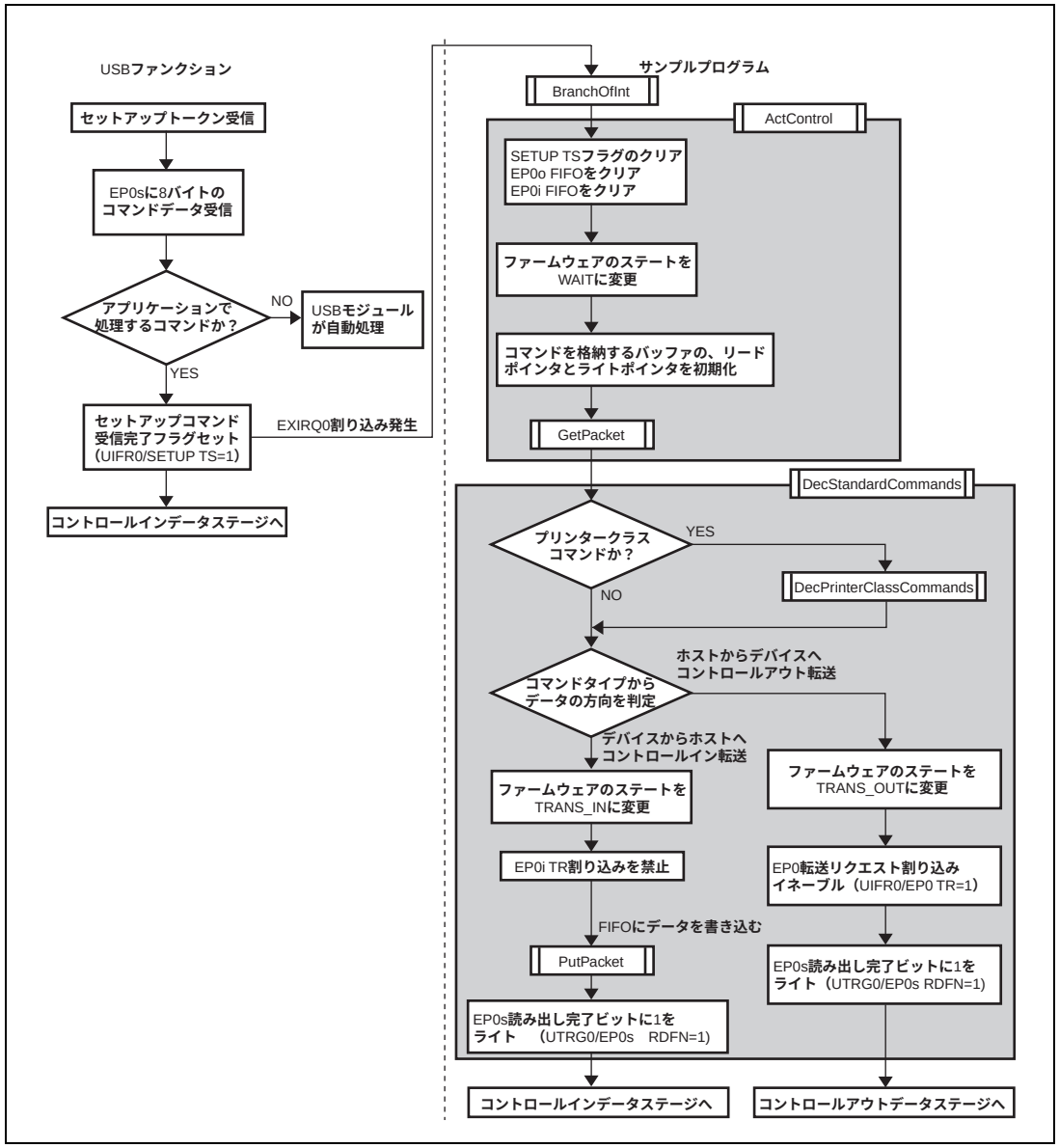


図 5.9　**セットアップステージ**

5.6.2 データステージ

データステージでは、ホストとファンクションがデータの送受信を行います。ファームウエアのステートは、セットアップステージで行ったコマンドのデコード結果によって、コントロールイン転送の場合は TRANS_IN に、コントロールアウト転送の場合は TRANS_OUT になります。

図 5.10、**図** 5.11 にコントロール転送のデータステージにおけるサンプルプログラムの動作を示します。

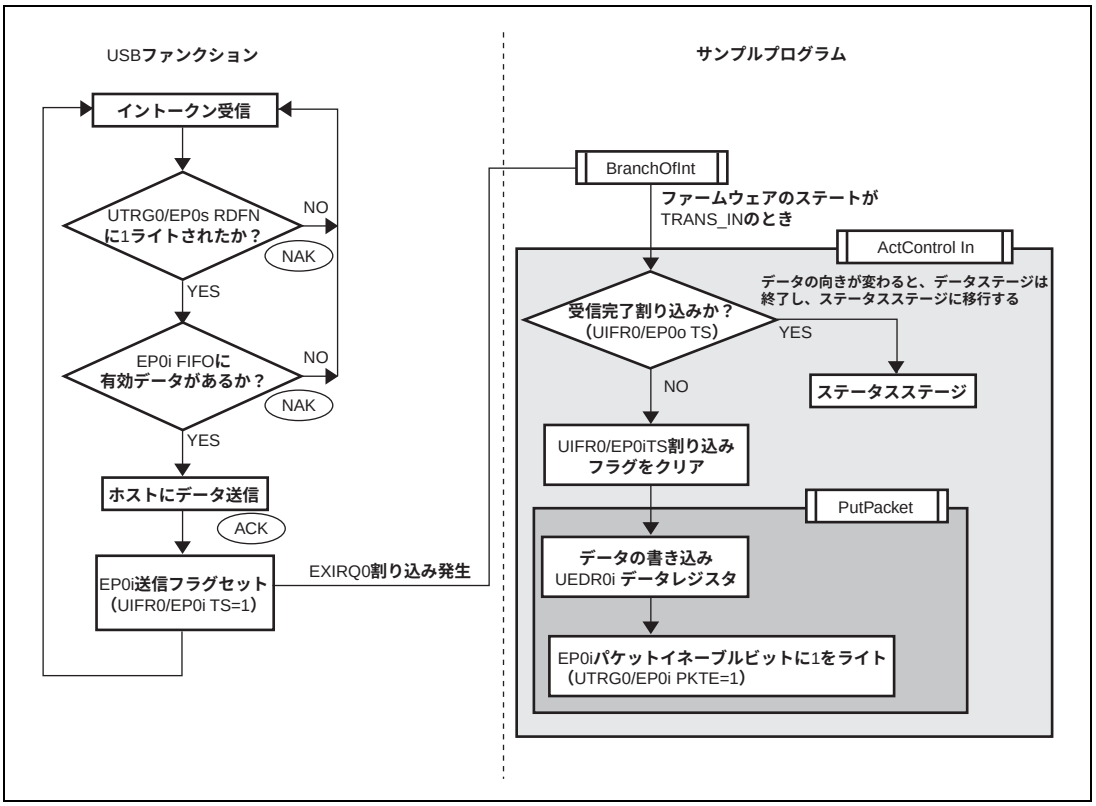


図 5.10 **データステージ（コントロールイン転送）**

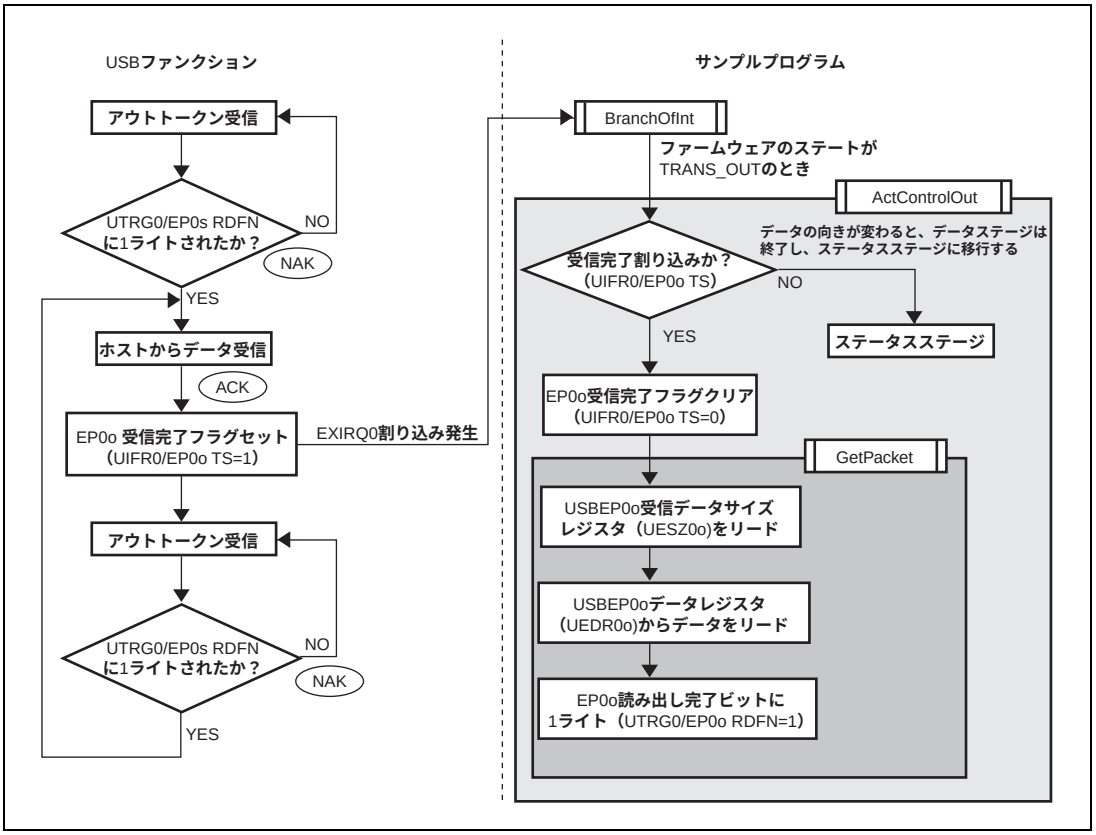


図 5.11　**データステージ（コントロールアウト転送）**

5.6.3 ステータスステージ

ステータスステージは、データステージと反対方向のトークンによって開始されます。つまり、コントロールイン転送では、ホストコントローラからのアウトトークンによってステータスステージが開始され、コントロールアウト転送では、ホストコントローラからのイントークンによってステータスステージが開始されます。

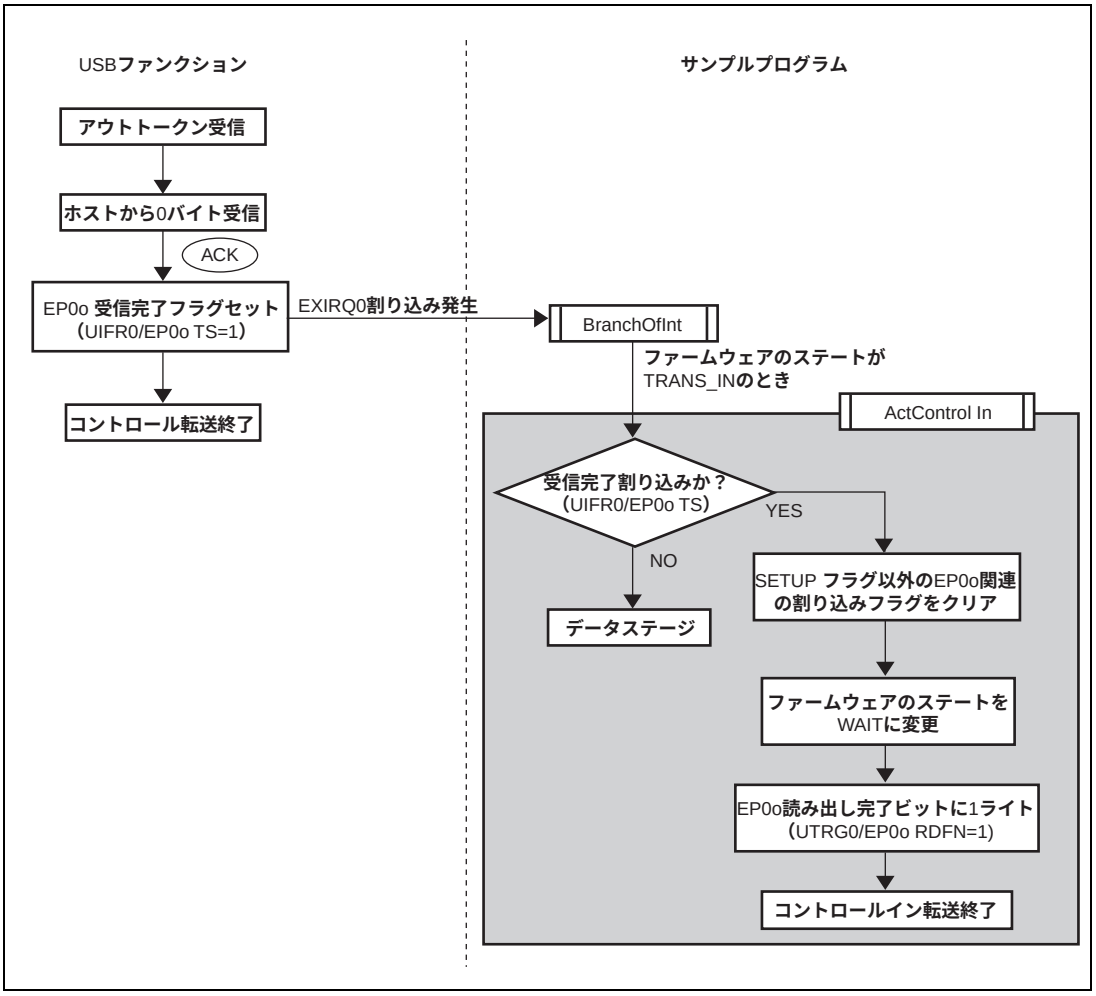


図 5.12 **ステータスステージ（コントロールイン転送）**

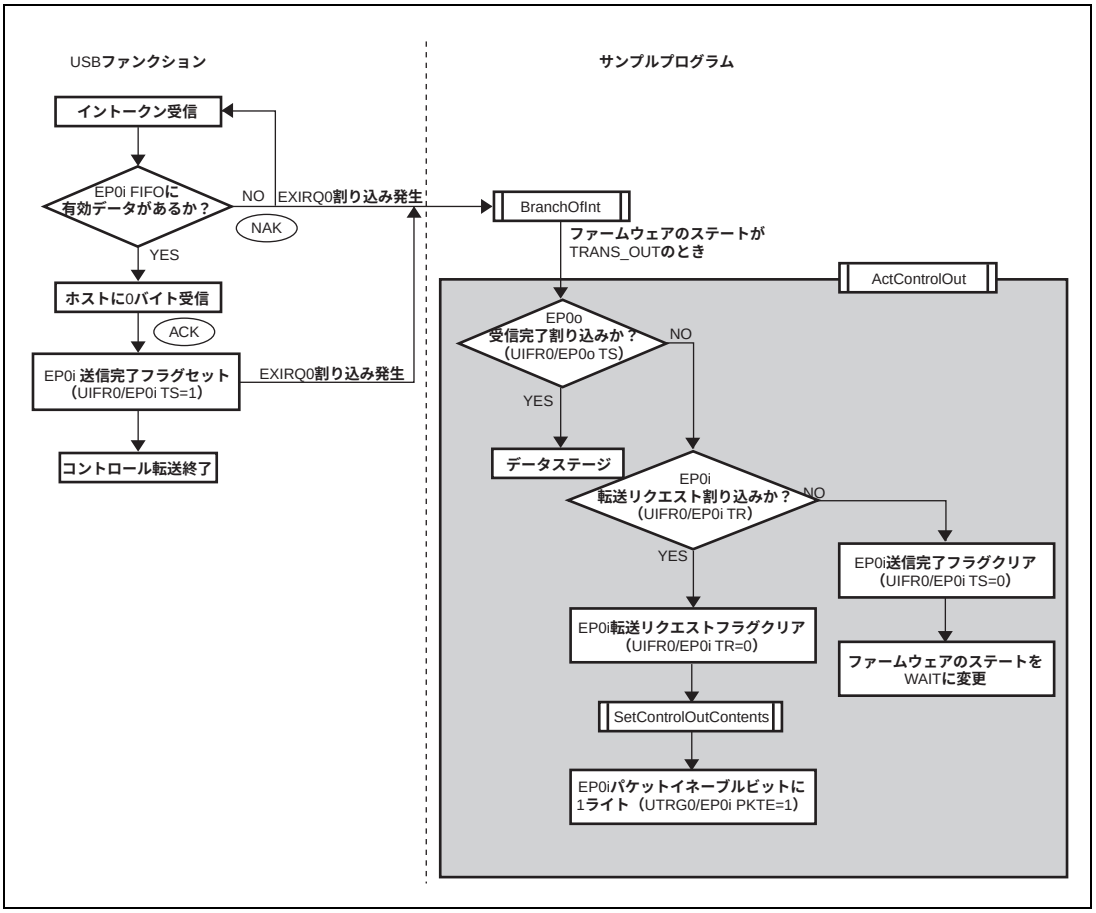


図 5.13　**ステータスステージ（コントロールアウト転送）**

5.7 バルク転送

バルク転送には、割り込みフラグレジスタ 1 ビット 0～2 を使用します。バルク転送もデータを送信する向きによって、2 つに分けることができます（**図** 5.14 参照）。

ホストコントローラから USB ファンクションへデータ転送する場合をバルクアウト転送、反対の場合をバルクイン転送と呼びます。

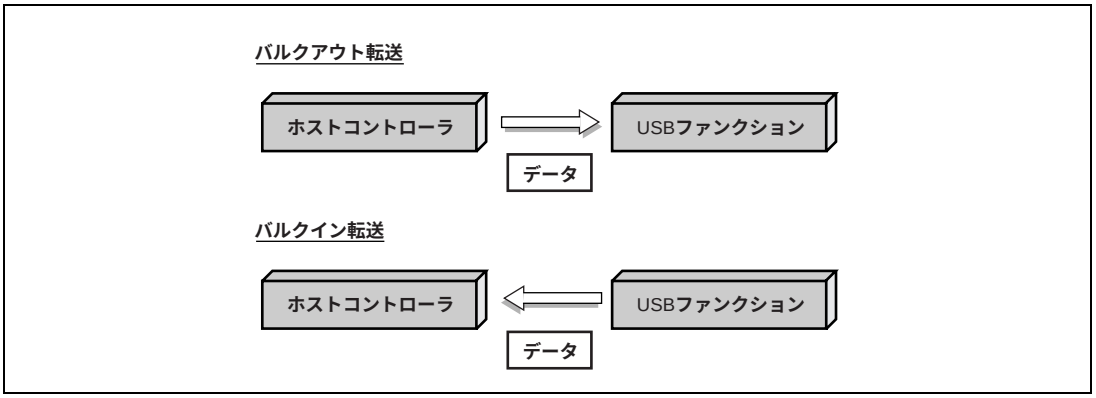


図 5.14 **バルク転送**

5.7.1 バルクアウト転送

図 5.15 にバルクアウト転送におけるサンプルプログラムの動作を示します。

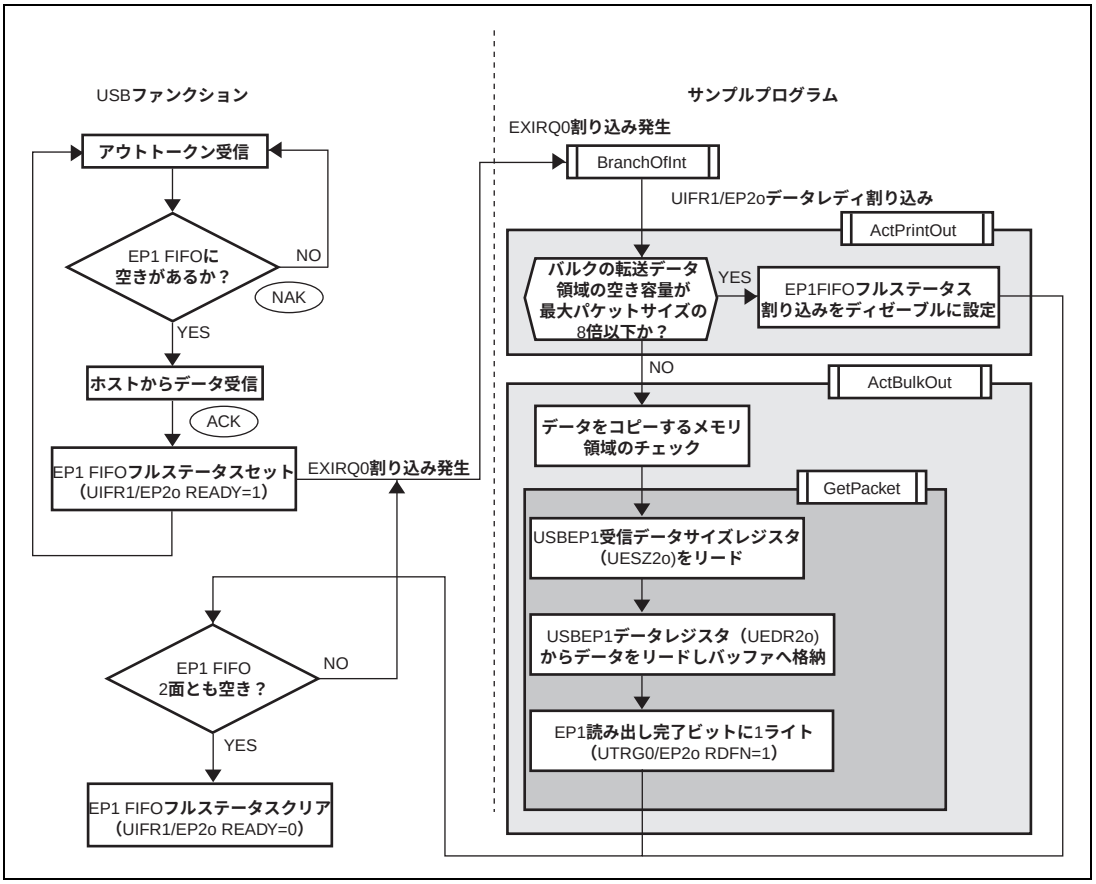


図 5.15 **バルクアウト転送**

5.7.2 バルクイン転送

図 5.16 にバルクイン転送におけるサンプルプログラムの動作を示します。

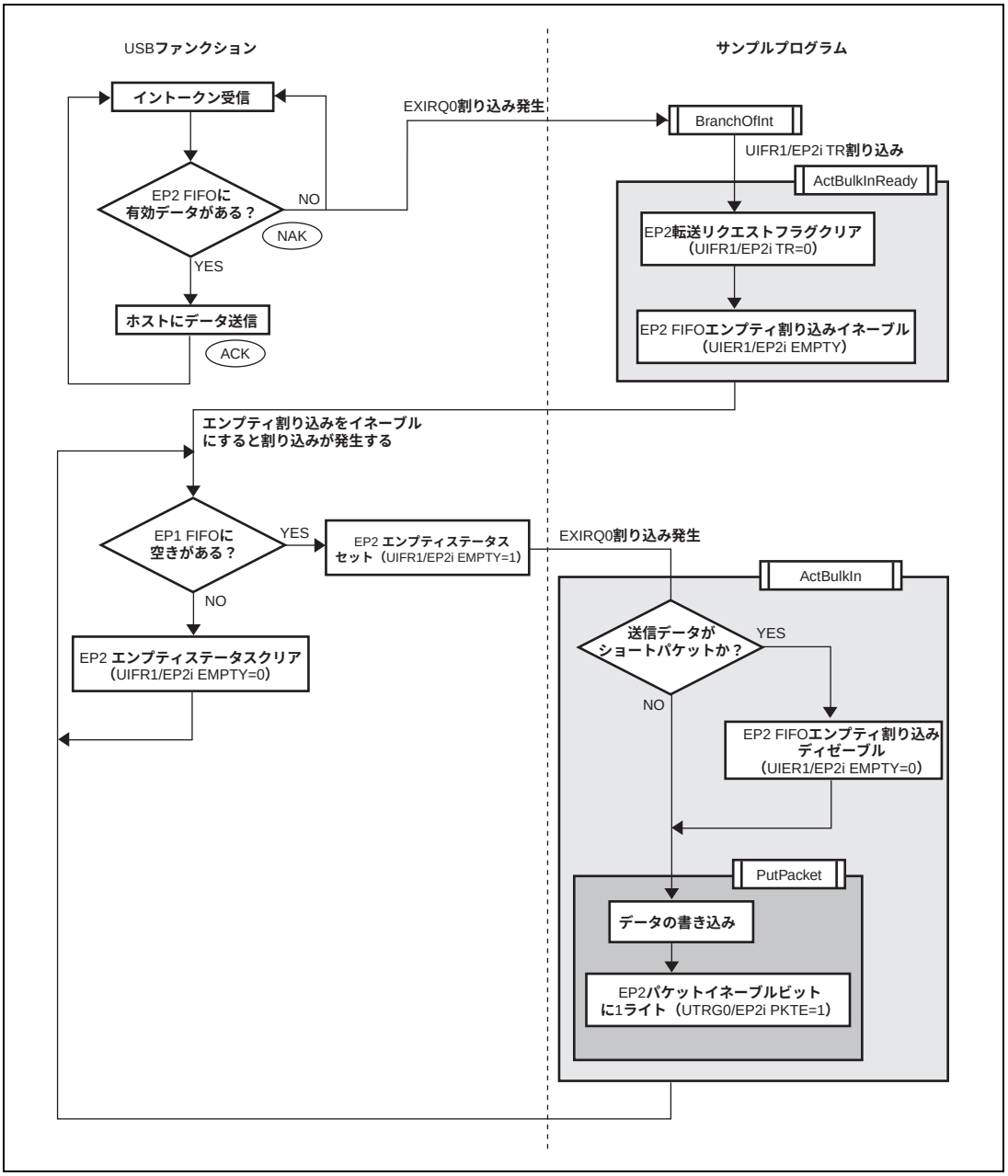


図 5.16 **バルクイン転送**

6. アナライザのデータ

この章では、H8S/2215 内蔵 USB ファンクションモジュールを使用して、CATC 社製 USB プロトコルアナライザ「USB Inspector」（国内：（株）東陽テクニカ（http://www.toyo.co.jp/））を用いた測定を行い、実際にバスを流れているデータについて「デバイス接続時のコントロール転送」および「プリントアウト時のバルクアウト転送」を例に説明します。なお、パケットの詳細につきましては**「2.6.1　パケットの概要」**をご参照下さい。

【注】　各パケットの前部にある「Packet#」は測定時のパケット通し番号です。

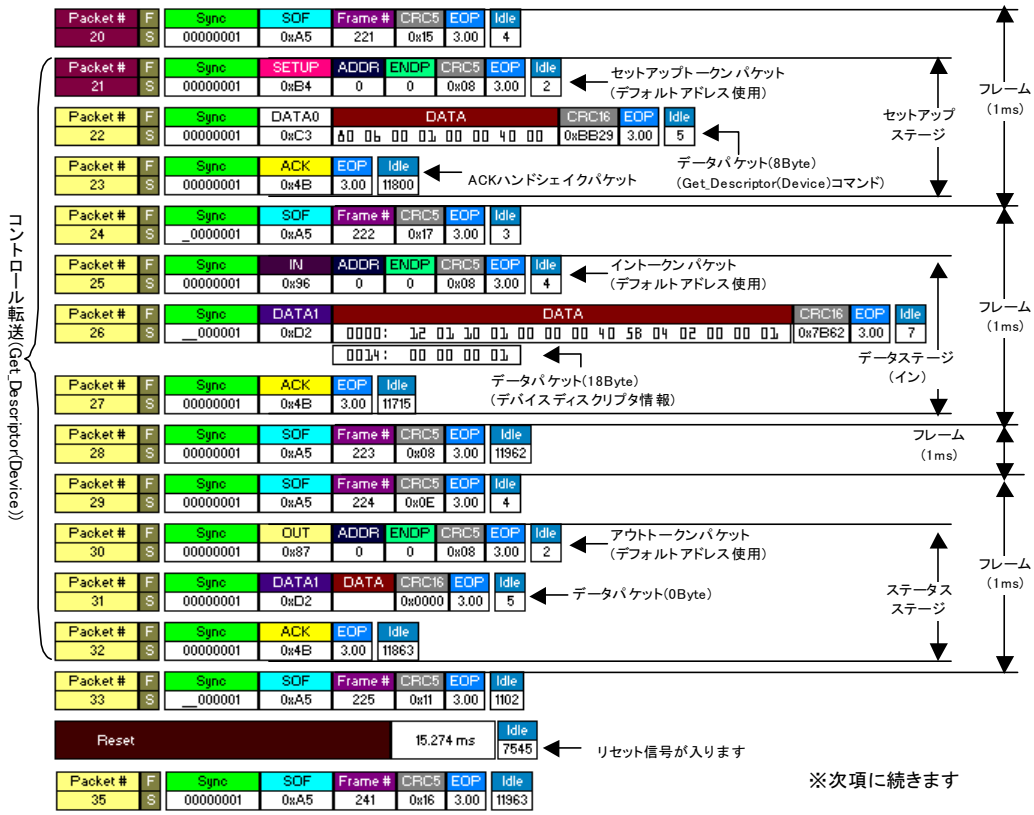
後部にある「Idle」はパケット間のアイドル（「2.2　USB の信号転送方式」および「2.6　USB のパケットとデータ転送」参照）となります。

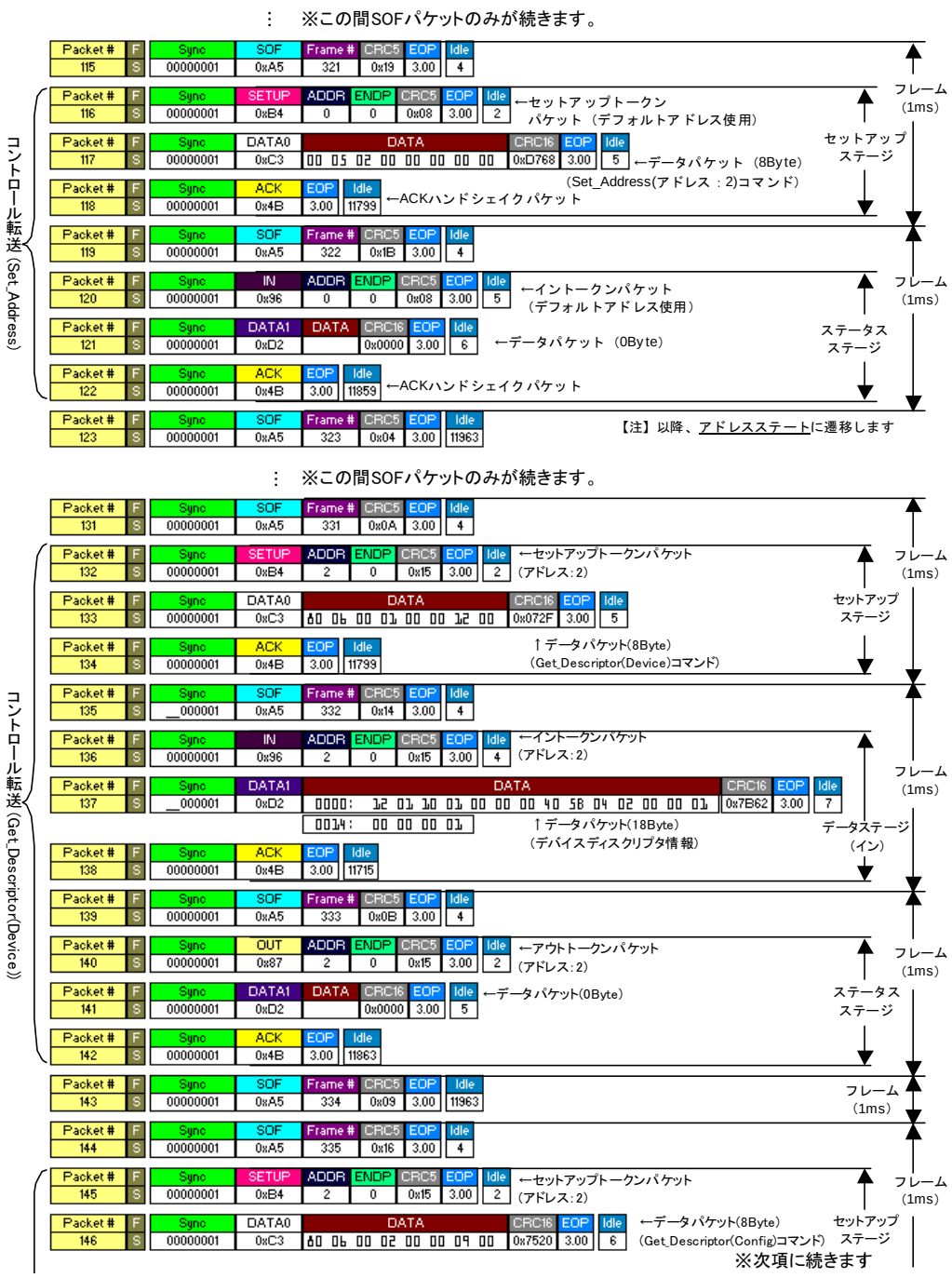
6.1 デバイス接続時のコントロール転送

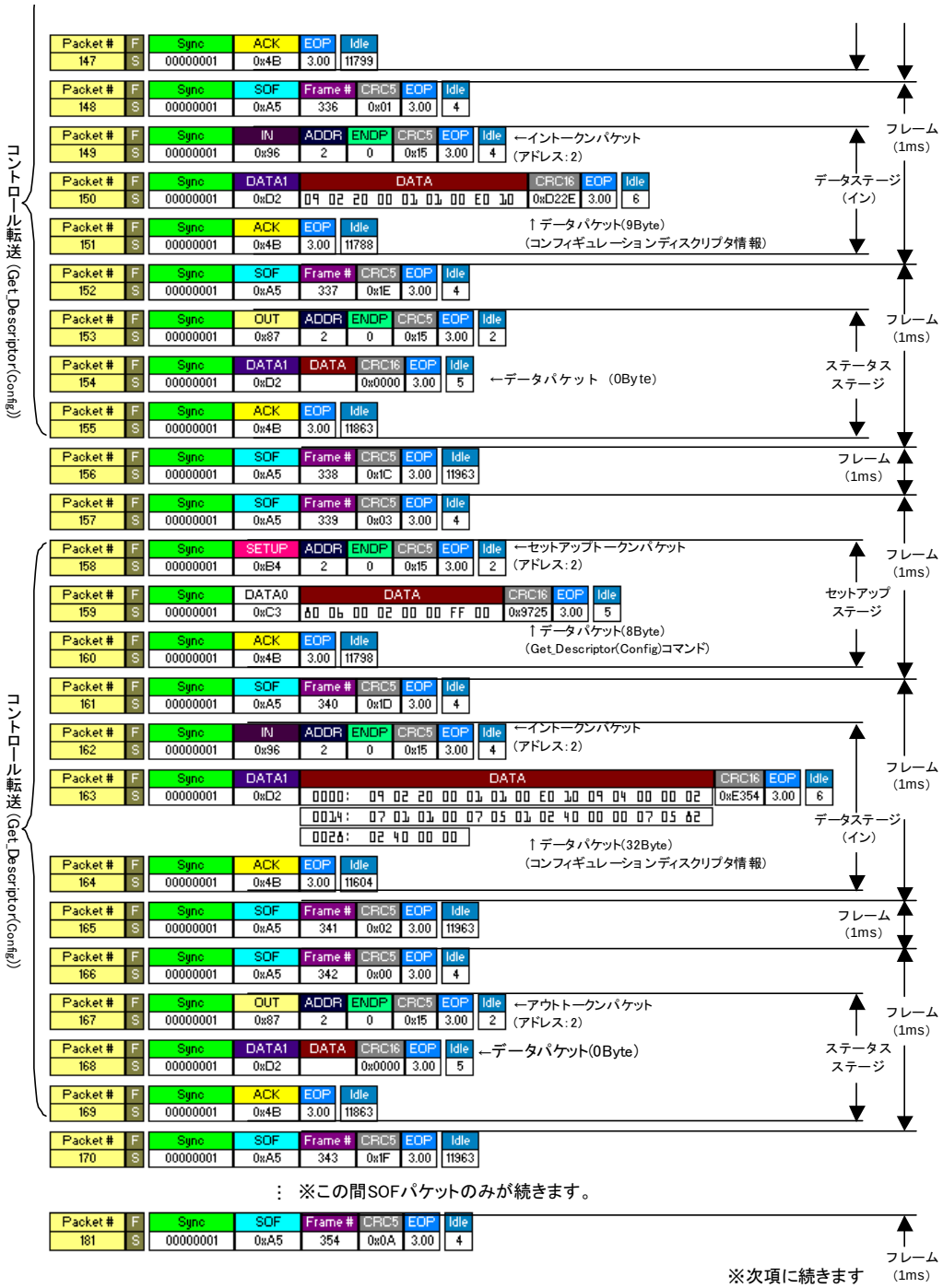
図 6.1 は本デバイスをホストコントローラに接続し、Vbus に電源が供給されている（電源投入ステート）状態から、デバイスが使用可能になる状態（構成ステート）に至るまでを測定したものです（ステートの遷移は**「2.7.1 デバイスのステート」**をご参照下さい）。

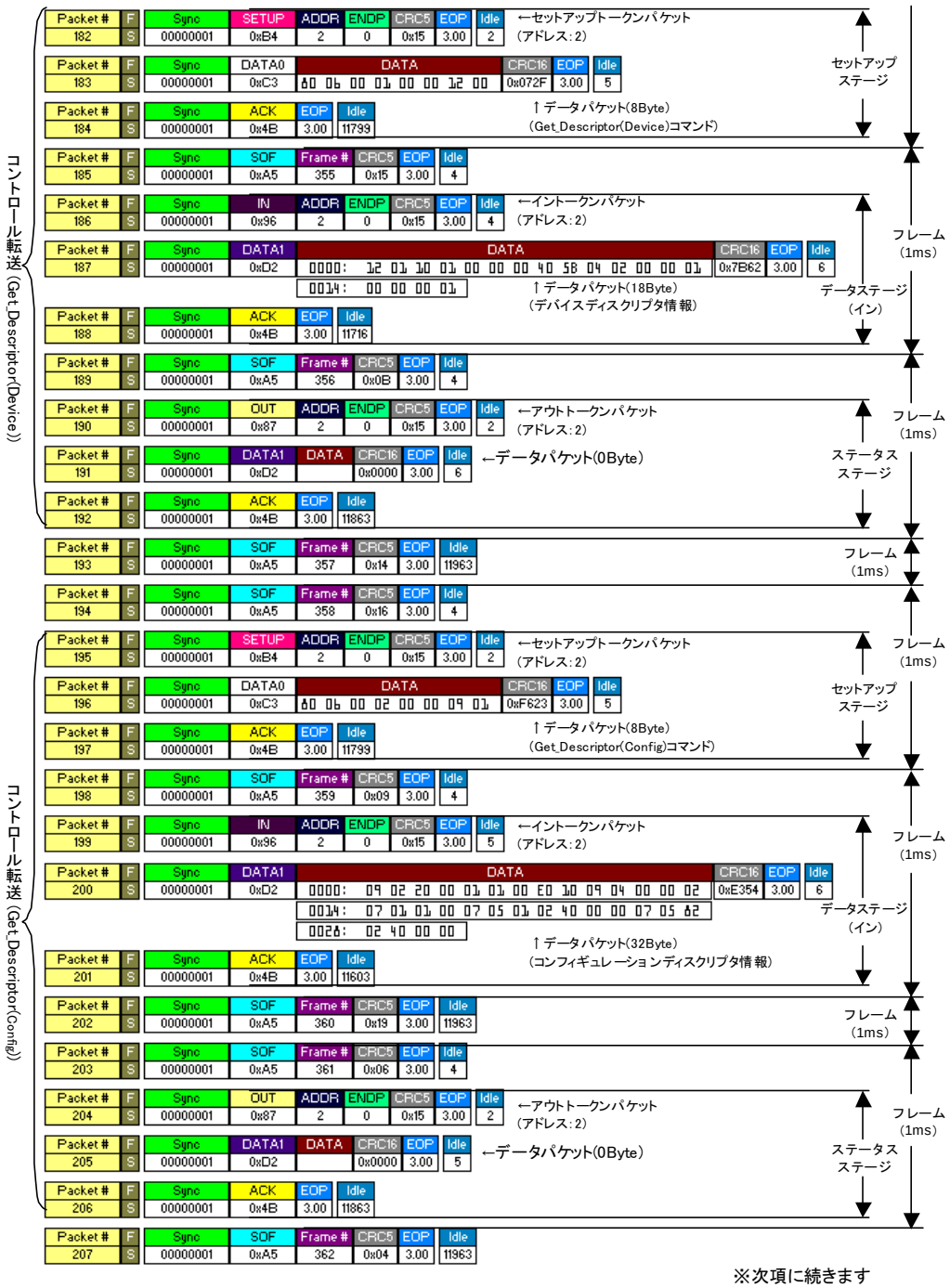
なお、ホストコントローラによりパケットのスケジューリングが本図と異なる場合がありますが、構成ステートに至るまでのコマンドの流れは同一です。

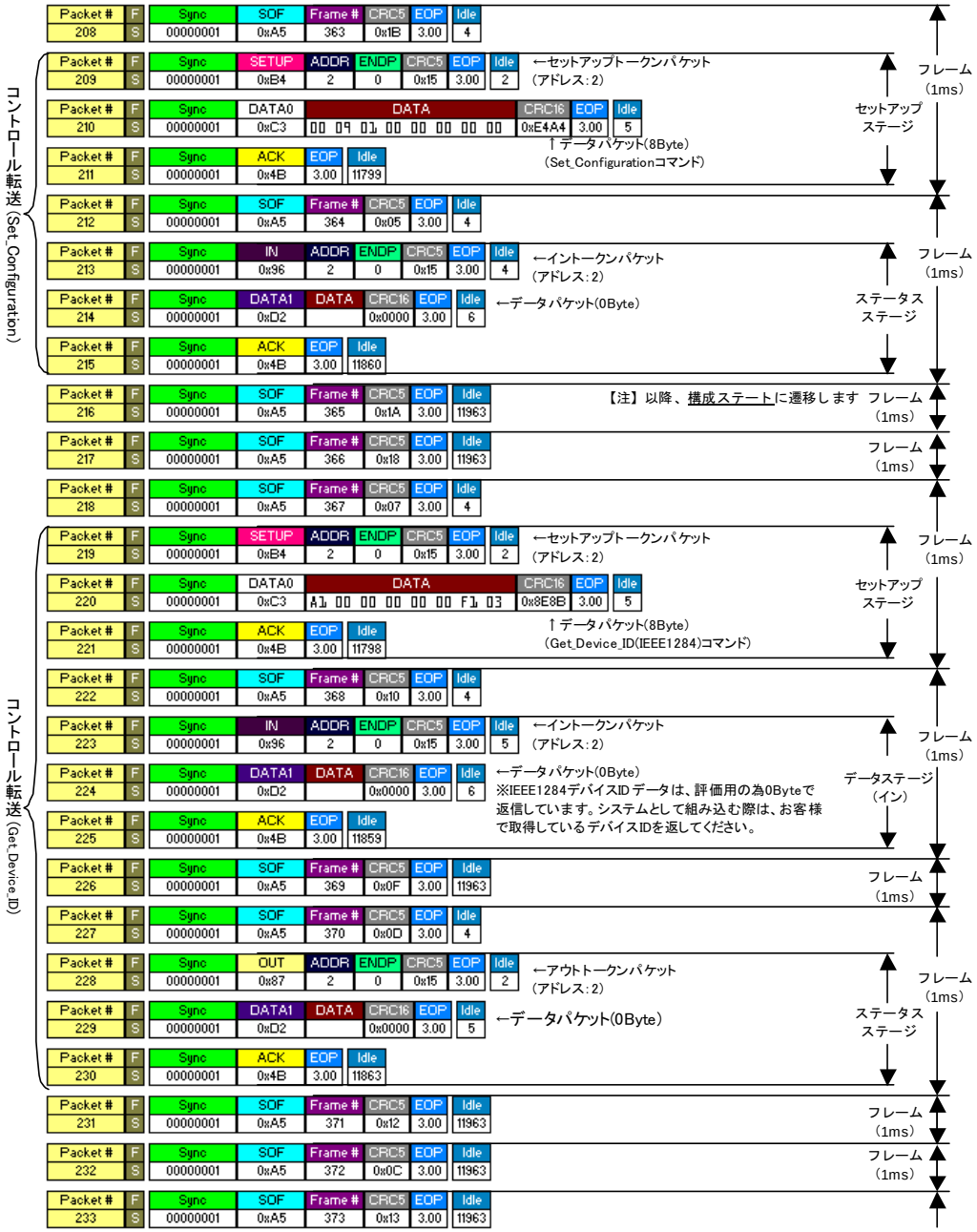
※リセット信号を受信する事により電源投入ステートから、デフォルトステートに移行します。




：　※この間SOFパケットのみが続きます。
コントロール転送（Set_Address）
Packet # 115 | Sync 00000001 | SOF 0xA5 | Frame # 321 | CRC5 0x19 | EOP 3.00 | Idle 4
Packet # 116 | Sync 00000001 | SETUP 0xB4 | ADDR 0 | ENDP 0 | CRC5 0x08 | EOP 3.00 | Idle 2 ←セットアップトークンパケット（デフォルトアドレス使用）
Packet # 117 | Sync 00000001 | DATA0 0xC3 | DATA 00 05 02 00 00 00 00 00 | CRC16 0xD768 | EOP 3.00 | Idle 5 ←データパケット（8Byte）（Set_Address(アドレス：2)コマンド）
Packet # 118 | Sync 00000001 | ACK 0x4B | EOP 3.00 | Idle 11799 ←ACKハンドシェイクパケット
セットアップステージ
Packet # 119 | Sync 00000001 | SOF 0xA5 | Frame # 322 | CRC5 0x1B | EOP 3.00 | Idle 4
フレーム（1ms）
Packet # 120 | Sync 00000001 | IN 0x96 | ADDR 0 | ENDP 0 | CRC5 0x08 | EOP 3.00 | Idle 5 ←イントークンパケット（デフォルトアドレス使用）
Packet # 121 | Sync 00000001 | DATA1 0xD2 | DATA | CRC16 0x0000 | EOP 3.00 | Idle 6 ←データパケット（0Byte）
Packet # 122 | Sync 00000001 | ACK 0x4B | EOP 3.00 | Idle 11859 ←ACKハンドシェイクパケット
ステータスステージ
フレーム（1ms）
Packet # 123 | Sync 00000001 | SOF 0xA5 | Frame # 323 | CRC5 0x04 | EOP 3.00 | Idle 11963
【注】以降、アドレスステートに遷移します
：　※この間SOFパケットのみが続きます。
コントロール転送（Get_Descriptor(Device)）
Packet # 131 | Sync 00000001 | SOF 0xA5 | Frame # 331 | CRC5 0x0A | EOP 3.00 | Idle 4
Packet # 132 | Sync 00000001 | SETUP 0xB4 | ADDR 2 | ENDP 0 | CRC5 0x15 | EOP 3.00 | Idle 2 ←セットアップトークンパケット（アドレス:2）
Packet # 133 | Sync 00000001 | DATA0 0xC3 | DATA 80 06 00 01 00 00 12 00 | CRC16 0x072F | EOP 3.00 | Idle 5
Packet # 134 | Sync 00000001 | ACK 0x4B | EOP 3.00 | Idle 11799 ↑データパケット(8Byte)（Get_Descriptor(Device)コマンド）
セットアップステージ
Packet # 135 | Sync 00000001 | SOF 0xA5 | Frame # 332 | CRC5 0x14 | EOP 3.00 | Idle 4
フレーム（1ms）
Packet # 136 | Sync 00000001 | IN 0x96 | ADDR 2 | ENDP 0 | CRC5 0x15 | EOP 3.00 | Idle 4 ←イントークンパケット（アドレス:2）
Packet # 137 | Sync 00000001 | DATA1 0xD2 | DATA 0000: 12 01 10 01 00 00 00 40 58 04 02 00 00 01 0014: 00 00 00 01 | CRC16 0x7B62 | EOP 3.00 | Idle 7 ↑データパケット(18Byte)（デバイスディスクリプタ情報）
Packet # 138 | Sync 00000001 | ACK 0x4B | EOP 3.00 | Idle 11715
データステージ（イン）
Packet # 139 | Sync 00000001 | SOF 0xA5 | Frame # 333 | CRC5 0x0B | EOP 3.00 | Idle 4
フレーム（1ms）
Packet # 140 | Sync 00000001 | OUT 0x87 | ADDR 2 | ENDP 0 | CRC5 0x15 | EOP 3.00 | Idle 2 ←アウトトークンパケット（アドレス:2）
Packet # 141 | Sync 00000001 | DATA1 0xD2 | DATA | CRC16 0x0000 | EOP 3.00 | Idle 5 ←データパケット(0Byte)
Packet # 142 | Sync 00000001 | ACK 0x4B | EOP 3.00 | Idle 11863
ステータスステージ
Packet # 143 | Sync 00000001 | SOF 0xA5 | Frame # 334 | CRC5 0x09 | EOP 3.00 | Idle 11963
フレーム（1ms）
Packet # 144 | Sync 00000001 | SOF 0xA5 | Frame # 335 | CRC5 0x16 | EOP 3.00 | Idle 4
フレーム（1ms）
Packet # 145 | Sync 00000001 | SETUP 0xB4 | ADDR 2 | ENDP 0 | CRC5 0x15 | EOP 3.00 | Idle 2 ←セットアップトークンパケット（アドレス:2）
Packet # 146 | Sync 00000001 | DATA0 0xC3 | DATA 80 06 00 02 00 00 09 00 | CRC16 0x7520 | EOP 3.00 | Idle 6 ←データパケット(8Byte)（Get_Descriptor(Config)コマンド）
セットアップステージ
※次項に続きます


コントロール転送(Get_Descriptor(Config))
Packet # 147 F S Sync 00000001 ACK 0x4B EOP 3.00 Idle 11799
Packet # 148 F S Sync 00000001 SOF 0xA5 Frame # 336 CRC5 0x01 EOP 3.00 Idle 4
Packet # 149 F S Sync 00000001 IN 0x96 ADDR 2 ENDP 0 CRC5 0x15 EOP 3.00 Idle 4
←イントークンパケット
(アドレス:2)
Packet # 150 F S Sync 00000001 DATA1 0xD2 DATA 09 02 20 00 01 01 00 E0 10 CRC16 0xD22E EOP 3.00 Idle 6
↑データパケット(9Byte)
(コンフィギュレーションディスクリプタ情報)
Packet # 151 F S Sync 00000001 ACK 0x4B EOP 3.00 Idle 11788
Packet # 152 F S Sync 00000001 SOF 0xA5 Frame # 337 CRC5 0x1E EOP 3.00 Idle 4
Packet # 153 F S Sync 00000001 OUT 0x87 ADDR 2 ENDP 0 CRC5 0x15 EOP 3.00 Idle 2
Packet # 154 F S Sync 00000001 DATA1 0xD2 DATA CRC16 0x0000 EOP 3.00 Idle 5
←データパケット (0Byte)
Packet # 155 F S Sync 00000001 ACK 0x4B EOP 3.00 Idle 11863
Packet # 156 F S Sync 00000001 SOF 0xA5 Frame # 338 CRC5 0x1C EOP 3.00 Idle 11963
Packet # 157 F S Sync 00000001 SOF 0xA5 Frame # 339 CRC5 0x03 EOP 3.00 Idle 4
Packet # 158 F S Sync 00000001 SETUP 0xB4 ADDR 2 ENDP 0 CRC5 0x15 EOP 3.00 Idle 2
←セットアップトークンパケット
(アドレス:2)
Packet # 159 F S Sync 00000001 DATA0 0xC3 DATA 80 06 00 02 00 00 FF 00 CRC16 0x9725 EOP 3.00 Idle 5
↑データパケット(8Byte)
(Get_Descriptor(Config)コマンド)
Packet # 160 F S Sync 00000001 ACK 0x4B EOP 3.00 Idle 11798
Packet # 161 F S Sync 00000001 SOF 0xA5 Frame # 340 CRC5 0x1D EOP 3.00 Idle 4
Packet # 162 F S Sync 00000001 IN 0x96 ADDR 2 ENDP 0 CRC5 0x15 EOP 3.00 Idle 4
←イントークンパケット
(アドレス:2)
Packet # 163 F S Sync 00000001 DATA1 0xD2 DATA
0000: 09 02 20 00 01 01 00 E0 10 09 04 00 00 02
0014: 07 01 01 00 07 05 01 02 40 00 00 07 05 82
0028: 02 40 00 00 CRC16 0xE354 EOP 3.00 Idle 6
↑データパケット(32Byte)
(コンフィギュレーションディスクリプタ情報)
Packet # 164 F S Sync 00000001 ACK 0x4B EOP 3.00 Idle 11604
Packet # 165 F S Sync 00000001 SOF 0xA5 Frame # 341 CRC5 0x02 EOP 3.00 Idle 11963
Packet # 166 F S Sync 00000001 SOF 0xA5 Frame # 342 CRC5 0x00 EOP 3.00 Idle 4
Packet # 167 F S Sync 00000001 OUT 0x87 ADDR 2 ENDP 0 CRC5 0x15 EOP 3.00 Idle 2
←アウトトークンパケット
(アドレス:2)
Packet # 168 F S Sync 00000001 DATA1 0xD2 DATA CRC16 0x0000 EOP 3.00 Idle 5
←データパケット(0Byte)
Packet # 169 F S Sync 00000001 ACK 0x4B EOP 3.00 Idle 11863
Packet # 170 F S Sync 00000001 SOF 0xA5 Frame # 343 CRC5 0x1F EOP 3.00 Idle 11963
⋮ ※この間SOFパケットのみが続きます。
Packet # 181 F S Sync 00000001 SOF 0xA5 Frame # 354 CRC5 0x0A EOP 3.00 Idle 4
※次項に続きます
フレーム (1ms)
データステージ (イン)
ステータス ステージ
セットアップ ステージ


コントロール転送(Get_Descriptor(Device))
コントロール転送(Get_Descriptor(Config))
←セットアップトークンパケット
(アドレス:2)
セットアップ
ステージ
↑データパケット(8Byte)
(Get_Descriptor(Device)コマンド)
←イントークンパケット
(アドレス:2)
フレーム
(1ms)
↑データパケット(18Byte)
(デバイスディスクリプタ情報)
データステージ
(イン)
←アウトトークンパケット
(アドレス:2)
←データパケット(0Byte)
ステータス
ステージ
↑データパケット(8Byte)
(Get_Descriptor(Config)コマンド)
↑データパケット(32Byte)
(コンフィギュレーションディスクリプタ情報)
※次項に続きます



※以降、バルク転送が有るまで定常状態となります。

図6.1 デバイス接続時のコントロール転送

6.2 プリントアウト時のバルクアウト転送（バルクアウト転送については 2.6.3 項をご参照下さい）

図 6.2 は、ホストコントローラから本デバイスに向けてバルクアウト転送（プリントアウト）を行なっている際の測定結果です。

データパケットの PID が転送毎に「DATA0」→「DATA1」→「DATA0」…とトグルしています。

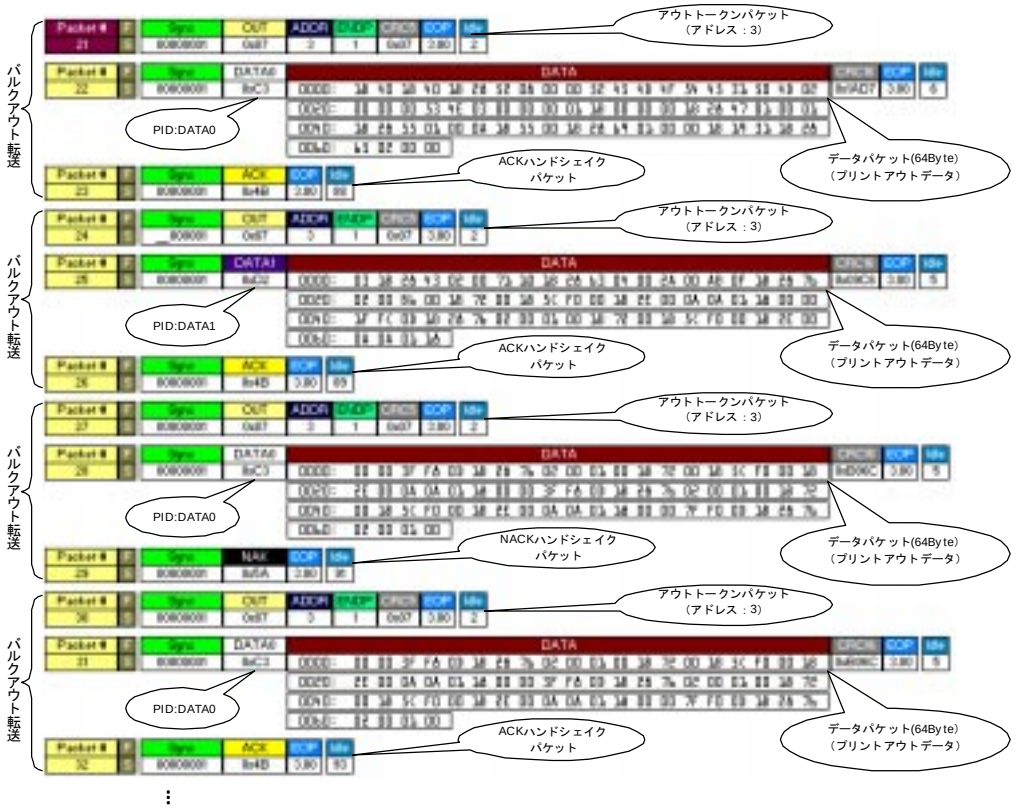


図6.2 プリントアウト時のバルクアウト転送

H8S/2215　USBファンクションモジュール
アプリケーションノート

発行年月　2002年 3 月　第 1 版
発　　行　株式会社　日立製作所
　　半導体グループビジネス企画本部
編　　集　株式会社　日立小平セミコン
　　技術ドキュメントグループ

H8S/2215　USB ファンクションモジュール
アプリケーションノート

ルネサス エレクトロニクス株式会社
神奈川県川崎市中原区下沼部1753 〒211-8668

ADJ-502-089