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。

Application Note

16

H8S/2218　USB ファンクションモジュール USB シリアル変換

アプリケーションノート

ルネサス16 ビットシングルチップマイクロコンピュータ
H8S ファミリ/H8S/2200 シリーズ

ルネサス エレクトロニクス
www.renesas.com

Rev.1.00　2003.10

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。

はじめに

本アプリケーションノートは、H8S/2218 内蔵の USB ファンクションモジュールを用いた USB シリアル変換ファームウェア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を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
- H8S/2218グループ、H8S/2212グループ ハードウェアマニュアル
- H8S/2218 Solition Engine CPUボード（MS2218CP01）取扱説明書
- H8Sファミリ用E10A エミュレータユーザーズマニュアル

【注】 **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/2218 **グループ、**H8S/2212 **グループ ハードウエアマニュアル** 14.1 **章参照）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/2218**、**H8S/2218 CPU **ボードのハードウエア仕様を記載してありますが、詳細は** H8S/2218 **グループ、**H8S/2212 **グループのハードウエアマニュアル、ならびに** H8S/2218 CPU **ボードの取扱説明書を御覧ください。**

目次

1. 概要

1.1 概要

本アプリケーションノートは、H8S/2218 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/2218 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
- USB動作クロック生成用のPLL回路内蔵（24 MHz×2＝48 MHz、または16 MHz×3＝48 MHz）
- バストランシーバを内蔵
- 専用端子（$\overline{\text{UBPM}}$）により、バスパワーモードとセルフパワーモードを選択可能
- Set_Configuration割り込みにより、現在のConfiguration値がチェック可能

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 1	EP1	バルクイン	64 Byte	64×2（128 Byte）	可能
エンドポイント 2	EP2	バルクアウト	64 Byte	64×2（128 Byte）	可能
エンドポイント 3	EP3	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。

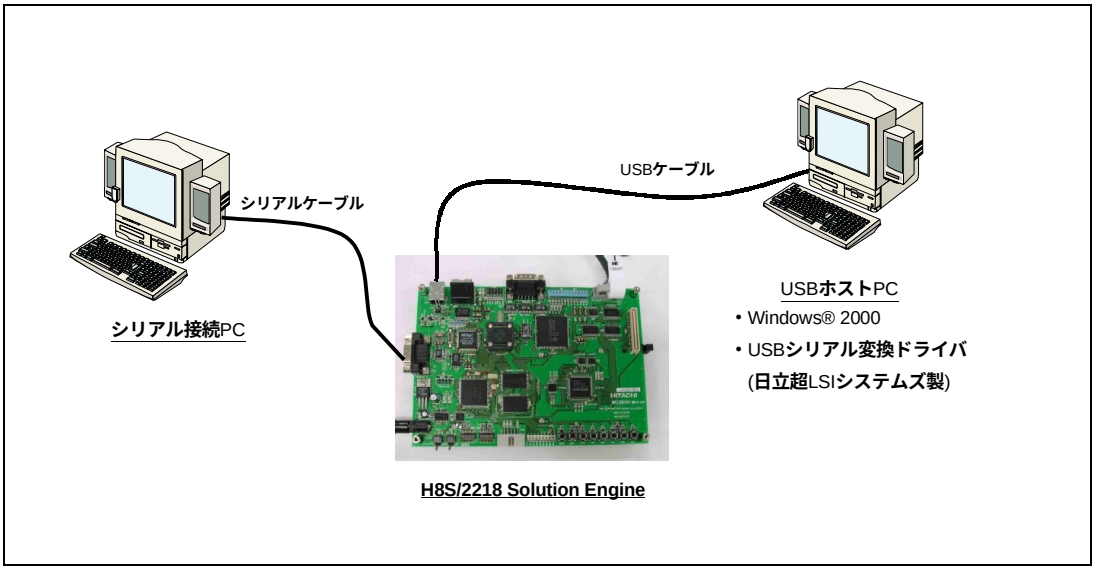


図 1.1　**システム構成**

本システムは、H8S/2218 を搭載した日立超 LSI システムズ社製の H8S/2218 CPU ボード（以下、MS2218CP）、シリアル接続 PC、USB シリアル変換ドライバ*[1]（日立超 LSI システムズ製）を搭載した USB ホスト PC（Windows® 2000）によって構成されています。

本システムは、USB ホスト PC から USB パケットにて送信されるデータを、MS2218CP によって受信しシリアルに変換後、シリアル接続 PC へ送信することができます。また、その逆の、シリアル接続 PC からのシリアルデータを MS2218CP によって受信し USB パケットに変換後、USB ホスト PC へ送信することができます。

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/2218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. E10A（カードエミュレータ）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、インタラプト転送についても対応可能*[2]

【注】 1. **システム（サンプルプログラム、**USB **シリアル変換ドライバ）に関するお問い合わせは、弊社営業担当までお願いします。**

USB **シリアル変換ドライバは、**Renesas Technology Corp.**のベンダ** ID：045B **以外では動作しません。お客様が製品化される場合は、別途日立超** LSI **システムズ社と** USB **シリアル変換ドライバに関し契約していただく必要があります。**

2. **インタラプト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。**

1.2 本システムの目的

最近、レガシーフリーPC（シリアルポートなどの旧規格のポートがなく、USB（Universal Serial Bus）などの Plug & Play に対応した新しい規格のポートしかない PC）が、PC の低価格化が進むなか、数多く出荷され始めています。これに伴い、既存のシリアル機器が PC に接続できなくなり、数多く存在するシリアル機器が使用できなくなるおそれがあります。この問題を解決するためには、既存のシリアル回線を USB に変換する装置が必要となってきます。本アプリケーションノートは、この問題を解決するための USB シリアル変換機能の実現例を提供することを目的としています。

本アプリケーションノートでは、既存のシリアル機器を新たに USB 機器に置き換えたとき、シリアルの API を用意することで、既存のシリアルアプリケーションから見てあたかも USB が存在しないように見せかけます。これにより、アプリケーションプログラムに変更を加える事なく使用することができるようになります。

図 1.2 に、PC とシリアル機器を既存のシリアル回線で接続した場合のハードウェア構成およびソフトウェア構成を示します。また、**図** 1.3 に USB シリアル変換装置を使用して PC とシリアル機器を接続した場合のハードウェア構成およびソフトウェア構成を示します。

図 1.2（a）に示すように、既存のシステムでは、シリアル機器と PC をシリアルケーブルで接続します。これに対し、**図** 1.3（a）に示すように既存シリアル機器を USB 経由で PC に接続する場合、PC とシリアル機器の間に USB シリアル変換装置が必要となります。USB シリアル変換装置は、USB 信号とシリアル信号を相互に変換する機能を持ちます。接続構成は、PC と USB シリアル変換装置を USB ケーブルで接続し、USB シリアル変換装置とシリアル機器をシリアルケーブルで接続する構成となります。これにより、PC とシリアル機器が相互に通信可能となります。

図 1.2（b）、**図** 1.3（b）は、ソフトウェア構成を階層構造で表わしたものです。破線で示した接続は、論理的に接続されるイメージを表わします。

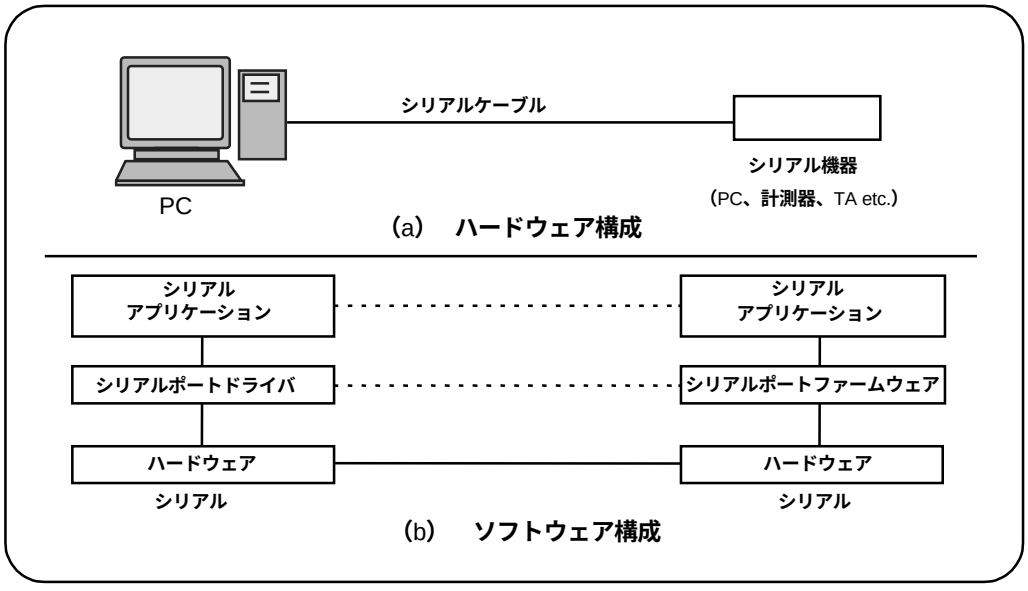


図 1.2 **既存シリアル回線での** PC **とシリアル機器接続構成例**

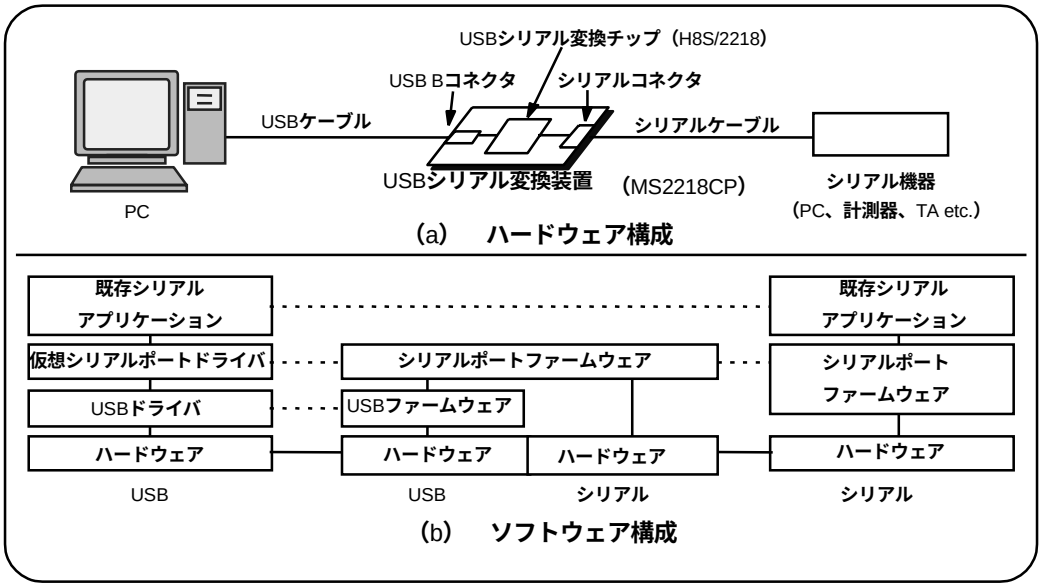


図 1.3 USB **経由での** PC **とシリアル機器接続構成例**

図 1.2（b）では、PC のシリアルアプリケーションからの送信データは、シリアルポートドライバに渡され、シリアルポートドライバは、そのデータを PC のシリアルハードウェアに送ります。シリアルハードウェアは、シリアル回線を通して、相手側のシリアルハードウェアにデータを送信します。シリアル機器のシリアルポートファームウェアは、データを受信したハードウェアからデータを取り出し、シリアルアプリケーションにデータを渡します。これにより、シリアルアプリケーション同士で、データのやり取りが可能となります。

一方、**図** 1.3（b）では、PC のシリアルアプリケーションからの送信データは、仮想シリアルポートドライバに渡されます。この仮想シリアルポートドライバは、既存のシリアルポートドライバと同じアプリケーションインタフェースを持っています。そのため、既存のシリアルアプリケーションは、アプリケーションから見て、あたかも USB が存在しない様に見せかける事が可能となり、既存のシリアルアプリケーションに変更を加える事なくデータ通信が可能となります。仮想シリアルポートドライバは、アプリケーションからのデータを下位の USB ドライバに渡し、USB ドライバは、そのデータを PC の USB ハードウェアに送ります。USB ハードウェアは、USB バスを通して、USB シリアル変換装置の USB ハードウェアにデータを送信します。USB シリアル変換装置では、受信した USB データをシリアルデータに変換し、シリアル機器にデータを送信します。USB シリアル変換装置とシリアル機器間の通信は、**図** 1.2 と同じ形となります。これにより、既存のシリアルアプリケーション同士で、データのやり取りが可能となります。

本アプリケーションノートは、**図** 1.3（b）の USB シリアル変換装置上のファームウェアに相当する、MS2218CP で動作するファームウェアの実現例を示します。

2. 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/2218 CPUボード（型名MS2218CP01）日立超LSIシステムズ社製
- E10Aカードエミュレータ　ルネサス テクノロジ製
- PCI（またはPCMCIA/USB）スロット搭載のPC（Windows® 95/Windows® 98/Windows® Me/Windows® 2000/Windows® XP）
- USBホスト用PC（Windows® 2000/Windows® XP）
- USBシリアル変換ドライバ　日立超LSIシステムズ社製
- シリアル接続PC
- USBケーブル
- シリアルケーブル（クロスケーブル）
- Debugging Interface（以下HDI）　ルネサス テクノロジ製
- High-performance Embedded Workshop（以下HEW）　ルネサス テクノロジ製

2.1 ハードウエア環境

図 2.1 に各デバイスの接続形態を示します。

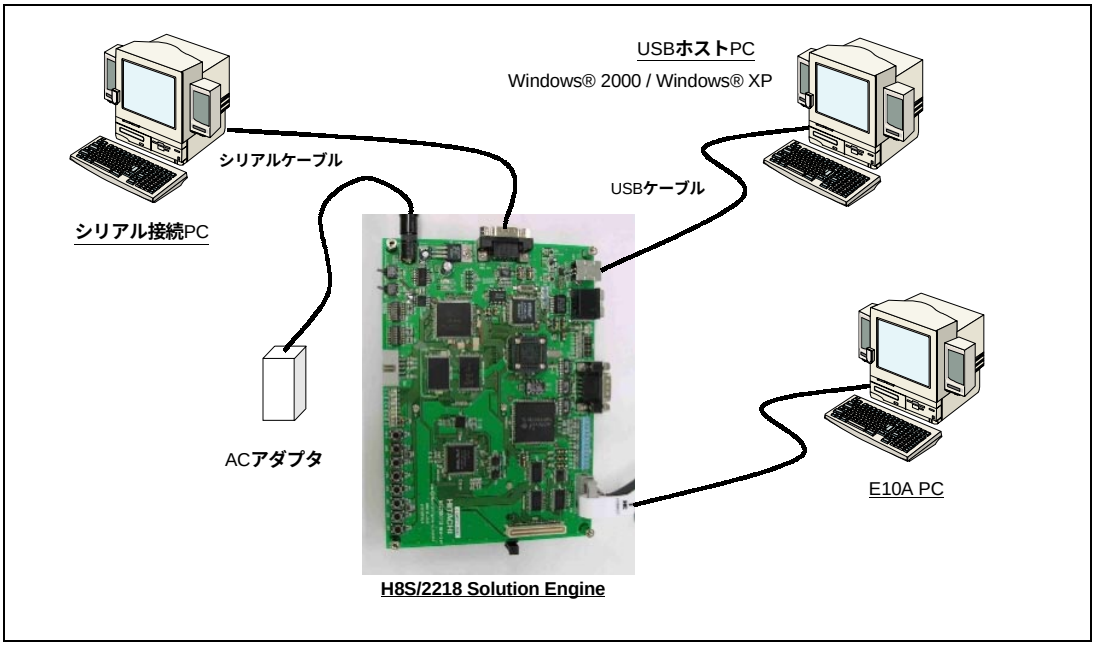


図 2.1 **デバイスの接続形態**

(1) MS2218CP

表 2.1 に示す MS2218CP ボードのディップスイッチとジャンパピン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、このディップスイッチとジャンパピンの設定をよくご確認ください。その他のジャンパピン、ディップスイッチを変更する必要はありません。

表 2.1 **ディップスイッチとジャンパピンの設定**

スイッチ	出荷時	変更後	ディップスイッチとジャンパピンの機能
SW1-1	OFF	ON	**選択モード** 6 **を選択**
SW1-2	OFF	OFF	
SW1-3	OFF	OFF	
SW1-5	OFF	ON	E10A **エミュレータモードを選択**
J-3	CLOSE	OPEN	USB **セルフパワーモードを選択**
J-9	CLOSE	OPEN	**ビッグエンディアンモードを選択**

(2) USB **ホスト** PC

USB ポート搭載の、Windows® 2000 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。このパソコンに USB シリアル変換ドライバ（日立超 LSI システムズ社製）をインストールします。

（3） シリアル接続 PC

シリアルポート搭載のパソコンをシリアルデータ送受信用のパソコンとして使用します。

（4） E10A **用** PC

PC カードスロットに E10A を挿入し、接続用ケーブルを会して MS2218CP と接続してください。

2.2 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ク、および、USB シリアル変換ドライバについて説明します。

2.2.1 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「H8S2218」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** 2.2 に示します。

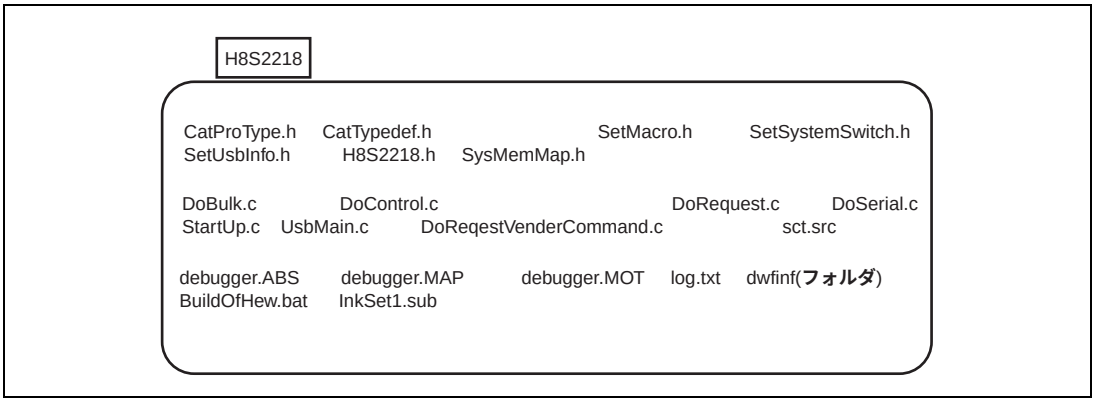


図 2.2 H8S2218 **フォルダに含まれるファイル**

2.2.2 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** 2.3）

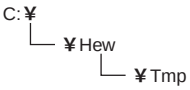


図 2.3　**作業フォルダの作成**

次に、サンプルプログラムが格納されているフォルダ（H8S2218）を任意の場所にコピーしてください。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、**”ch38_tmp” **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。**

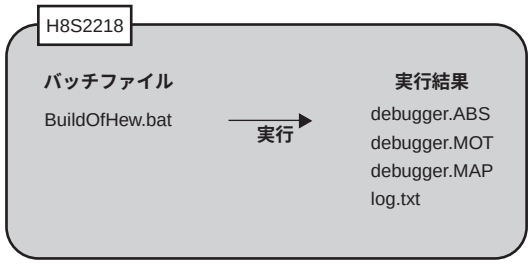


図 2.4　**コンパイル結果**

2.2.3 USB シリアル変換ドライバ

USB シリアル変換ドライバに関するファイルは、すべて“ UST-03 ” フォルダ内に収められています。

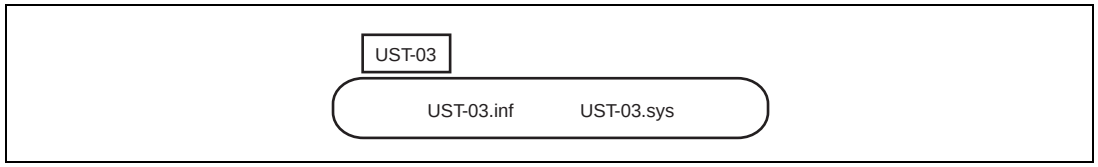


図 2.5 UST-03 **フォルダに含まれるファイル**

2.3 プログラムのロードと実行方法

図 2.6 にサンプルプログラムのメモリマップを示します。

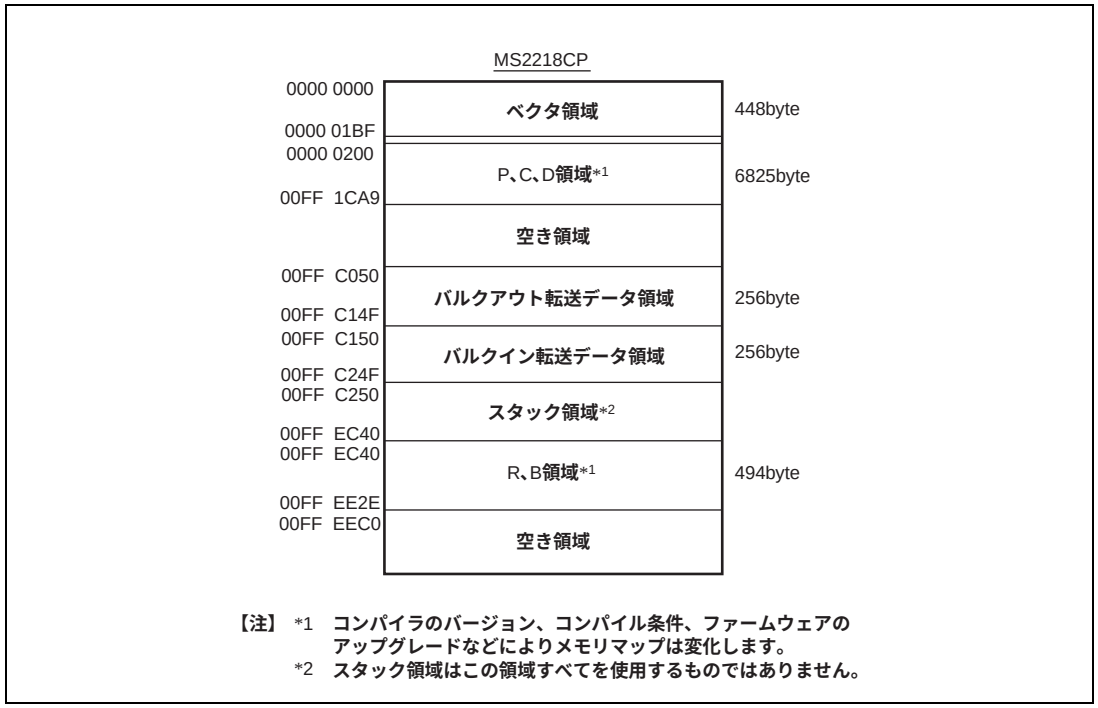


図 2.6 **メモリマップ**

図 2.6 のように、本サンプルプログラムは P、C、D を内蔵フラッシュメモリ領域に、R、B を内蔵 RAM 領域に配置しています。これらのメモリへの割り付けは、H8S2218 フォルダ内に含まれる“ lnkSet1.sub ”で指定します。

2.3.1 プログラムのロードと実行

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E10A用PCにE10Aを挿入してください。
- E10AケーブルでE10AとMS2218CPを接続してください。
- シリアルケーブルでシリアル接続PCとMS2218CPを接続してください。
- E10A用PC、シリアル接続PC、USBホストPCの電源を投入し、起動してください。
- MS2218CPの電源を投入してください。
- H8S2218フォルダ内のDebugger.hdsを実行してください。
- HDI for E10A H8S2218Fを起動してください。（詳細は**「H8S/2218 E10Aエミュレータユーザーズマニュアル」**を参照してください）
- メニューバーの File → LoadProgramを選択し、「debugger.ABS」をロードします。

以上の操作で、サンプルプログラムを MS2218CP 上にロードすることができます。

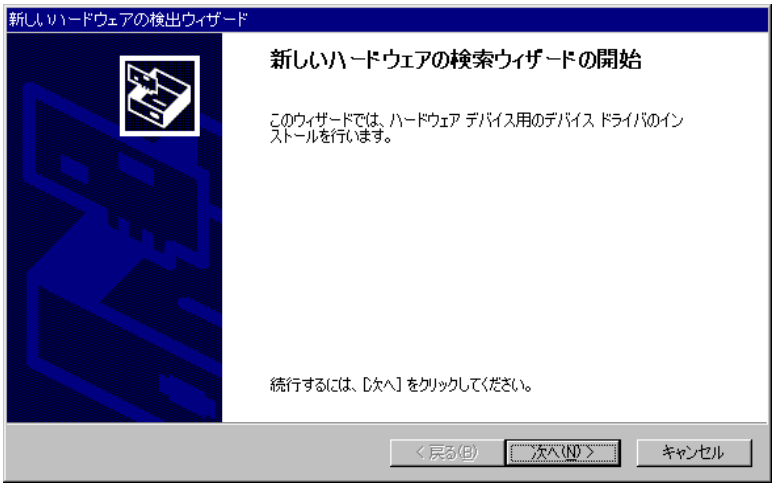
メニューバーの Veiw → Register Window を選択し、Register ウインドウを開きます。ウインドウ内の該当レジスタ（PC）の数値エリアをダブルクリックすると、ダイアログボックスが開き、レジスタの値を変更することができます。このダイアログボックスで PC を H'0000 0200 に設定してください。

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。

2.4 PC 間通信の方法

2.4.1 USB ホスト PC の設定

- 2.3.1、2.3.2に従い、サンプルプログラムを実行してください。サンプルプログラムが正常に起動すると、MS2218CP上の8ビットLEDが0xAAと表示されます。
- USBケーブルのシリーズBコネクタをMS2218CPに挿入し、反対側のシリーズAコネクタをUSBホストPCに接続してください。
- 画面に以下のダイアログが表示されるので、[次へ]をクリックします。

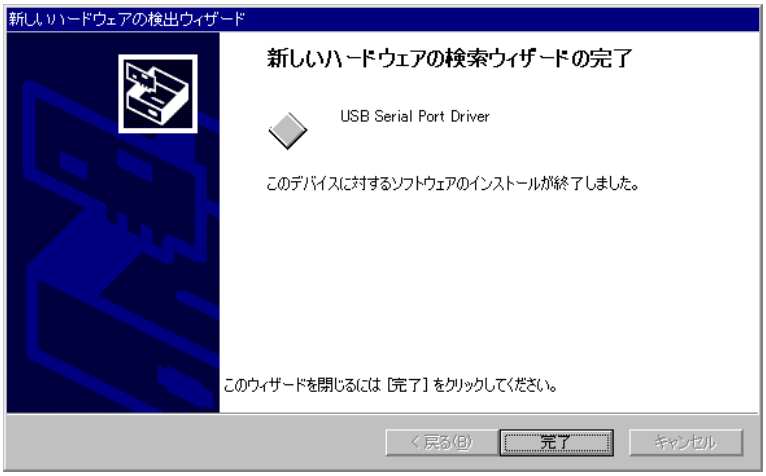


- 「デバイスに最適なドライバを検索する（推奨）」を選択し、[次へ]をクリックします。

- 「フロッピーディスクドライブ」を選択し、[次へ]をクリックします。

- UST-03.infを読み込むようになっていることを確認し、[次へ]をクリックします。

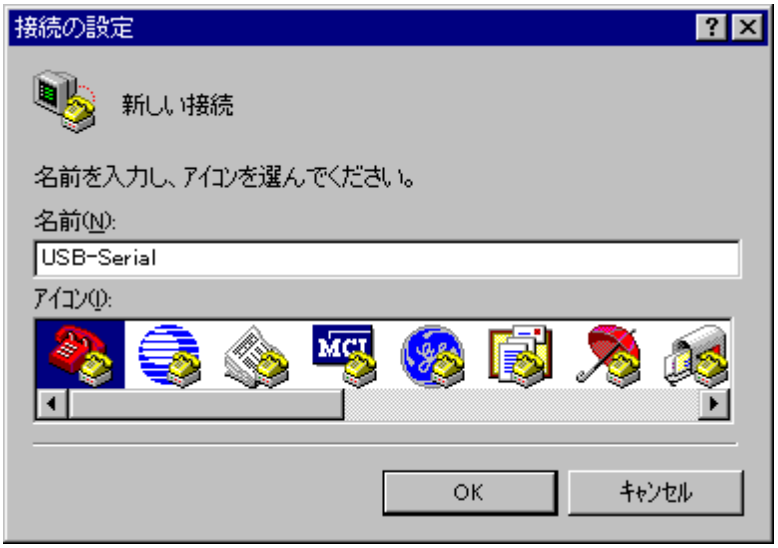
- [完了]をクリックします。



以上で、ドライバのインストール作業が終了し、USB ホスト PC は MS2218CP をシリアル COM ポートとして認識するようになります。

次に、WindowsOS 標準添付の通信ソフト、ハイパーターミナルを起動します。

- Windowsキーを押し、“ スタート→プログラム→アクセサリ（もしくはさらに『通信』の下） ” の順に選択し、ハイパーターミナルを起動します。
- 名前を入力し（任意、画面では、USB-Serialと入力）、「OK」を押します。



- 接続方法を「COM3」を選択し、「OK」をクリックします。

- シリアルポートの設定を行います。**表**2.2に示す値の範囲で所望の設定を行います。下図は、本プログラムのデフォルト値を入力した場合の例です。入力後、「OK」をクリックします。

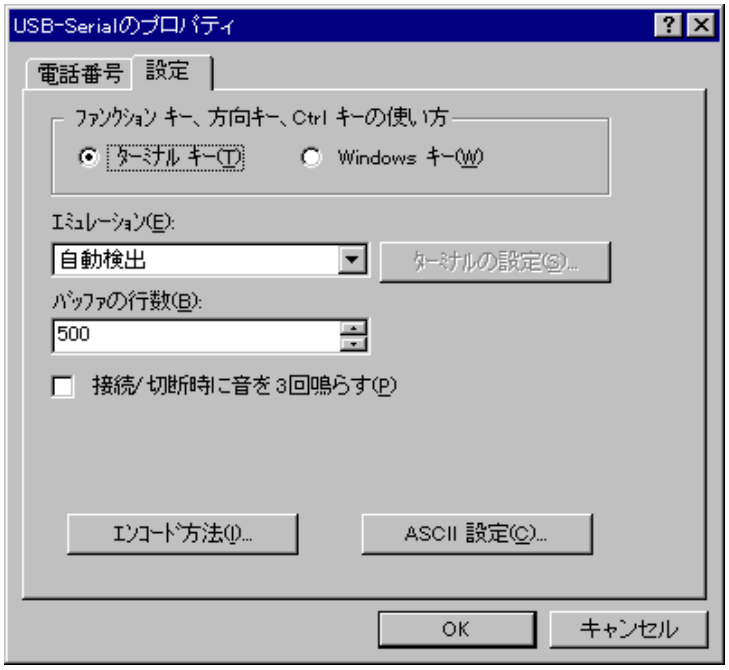
これで、ハイパーターミナルが起動します。**表** 2.2 に示す設定可能値以外の設定を行うと、MS2218CP 上の 8 ビット LED が 0x30 を表示して、**表** 2.2 に示す本プログラムのデフォルト値を設定します。設定可能値を設定すると、8 ビット LED は 0xAA を表示し続けます。

表 2.2　**シリアルポートの設定可能値**

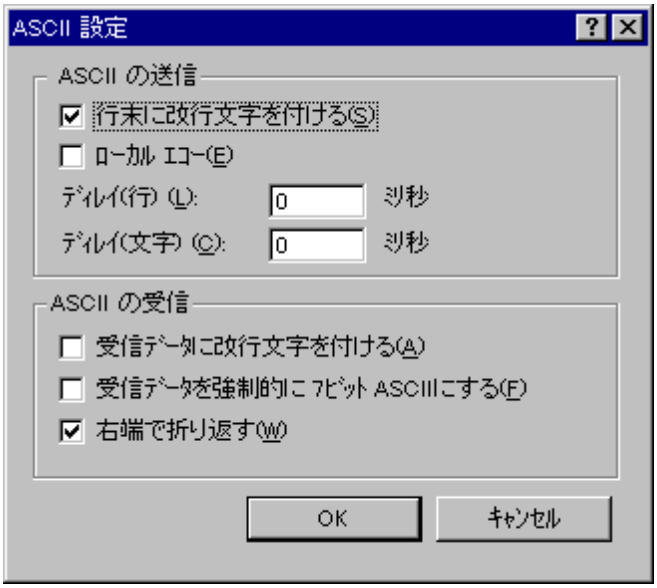
	本プログラムのデフォルト設定	**設定可能値**
ビット／秒[bps]	38400	9600、19200、38400*
データビット	8	8、7
パリティ	**なし**	**なし、奇数、偶数**
ストップビット	1	1、2
フロー制御	Xon/Xoff	Xon/Xoff **のみ**

【注】　*　**本サンプルプログラムでは** CPU **を** 16/24MHz **で動作させているため、**57600bps**、**115200bps **では誤差が大きくなり、正常動作しないおそれがあります。本サンプルプログラムでは** 57600bps**、**115200bps **は設定可能ですが、動作保障していません。**

- ハイパーターミナル起動後、通信を始める前に、「ファイルメニュー→プロパティ→設定」と順に選択して、「ASCII設定」をクリックします。



- ASCII送信欄の「行末に改行文字を付ける」にチェックを入れて、「OK」をクリックします。



2.4.2 シリアル接続 PC の設定

USB ホスト PC と同様にハイパーターミナルを起動します。シリアル通信の設定（ビット／秒、データビット、パリティ、ストップビット、フロー制御）は必ず USB ホスト PC と同じ設定にしてください。

2.4.3 PC 間通信

USB ホスト PC とシリアル接続 PC の両者のハイパーターミナルが起動すると、この両者間で、キー入力文字の転送、テキストファイルの転送、および、バイナリファイルの転送が行えます。

USB ホスト PC 側でキー入力することで、その入力文字がシリアル接続 PC へ転送されます。また、その逆で、シリアル接続 PC 側でキー入力することで、その文字を USB ホスト PC へと転送されます。

「転送→テキストファイルの転送」を選択することで、テキストファイルを相手側 PC へ送信する事ができます。

受信側 PC で「転送→ファイルの受信→ZMODEM」を選択して受信側 PC を受信待ち受け状態にしてから、送信側 PC で「転送→ファイルの送信→ZMODEM」と選択します。これによって、テキストファイル、バイナリファイルを受信側 PC へと送信できます。

【注】 本アプリケーションノートでは、PC 上で動作するシリアルアプリケーションとして、ハイパーターミナルを使用しております。そのため、その他のシリアルアプリケーションでは、別途動作確認が必要となります。

本サンプルプログラムでは、ソフトフロー制御（X-ON/X-OFF）を行っております。そのため、ファイルの送信は、ZMODEM などのソフトフロー制御（X-ON/X-OFF）対応プロトコルを必ず選択してください。

3. 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 MS2218CP 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もしくはメインループからの分岐によって USB 転送を開始します。また、SCI0 の割り込みもしくはメインループからの分岐によってシリアル転送を開始します。H8S/2218 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する割り込みは、EXIRQ0、EXIRQ1、IRQ6 の 3 種類ですが、本サンプルプログラムでは EXIRQ0 のみを使用します。また、SCI0 モジュールに関する割り込みは、ERI0（受信エラー）、RXI0（受信データフル）、TXI0（送信データエンプティ）、TEI0（送信終了）の 4 種類のうち、ERI0、RXI0 の 2 種類を使用します。

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。
- シリアル接続PCからシリアルデータを受信することができます。
- シリアル接続PCへシリアルデータを送信することができます。
- バルクアウト転送で受信したデータをシリアル送信することができます。
- シリアル受信したデータをバルクイン転送することができます。

3.1 状態遷移図

図 3.1 に、本サンプルプログラムの状態遷移図を示します。本サンプルプログラムは、**図** 3.1 のように 4 つの状態に遷移します。

1. 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/2218の初期設定を行います。

2. 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、USBホストPC、シリアル接続PCからのデータの有無を常に監視し、データがあればそれぞれ相手側PCへデータを出力します。つまり、MS2218CPへの入力データを常に監視し、入力データがあればそれぞれの相手側PCへデータ出力します。

3. 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要因は“割り込みフラグレジスタ0～3（UIFR0～3）”によって示され、計5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。

4. シリアル通信状態

定常状態において、SCI1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。本サンプルプログラムで使用する割り込み要因は、シリアルステータスレジスタ（SSR0）によって示され、ERI0、RXI0の計2種類です。

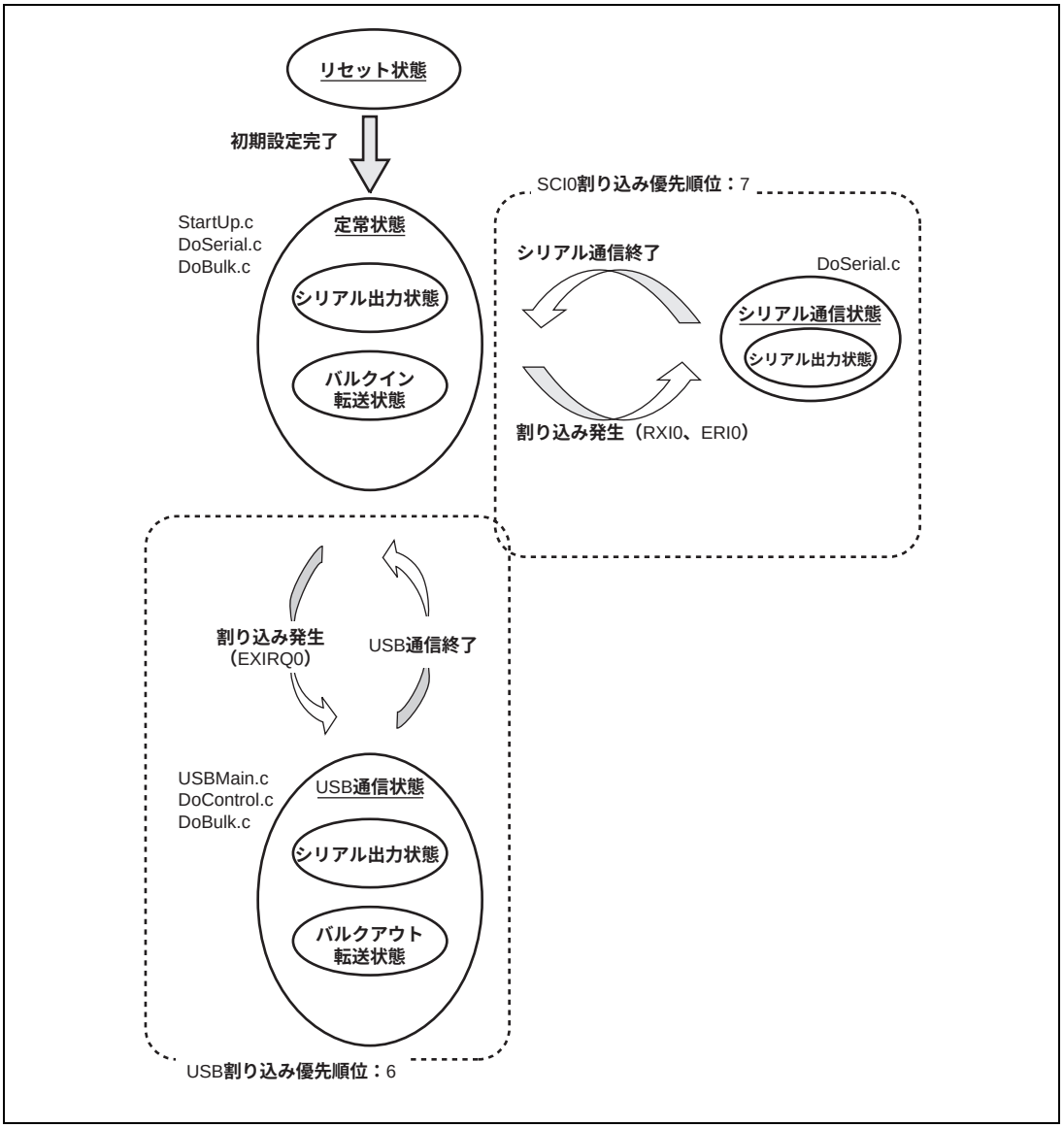


図 3.1　**状態遷移図**

本サンプルプログラムでは、USB の割り込み優先順位を 6、SCI0 の割り込み優先順位を 7 に設定します。この設定により、SCI0 割り込み処理中は USB 割り込みを受け付けることがなく、USB 割り込みによってシリアル受信処理が待たされることを防止します。

3.2 PC 間通信概要

図 3.2 に PC 間通信の概要を示します。本サンプルプログラムでは、通信形態は大きく分けて USB 通信とシリアル通信の二つあります。データの送受信を考えると、USB 通信はバルクイン転送、バルクアウト転送のさらに二つの通信形態に分ける事ができ、シリアル通信の方も同様にシリアル入力、シリアル出力の 2 つに分ける事ができるので、本サンプルプログラムでは計 4 つの通信形態があることになります。

本サンプルプログラムでは、データの流れを、『バルクアウト転送→シリアル出力』系と『シリアル入力→バルクイン転送』系の二つに分け、それぞれに 256Byte のバッファを持たせてあります。それぞれの系のバッファへの入力系は割り込み処理をし、バッファからの出力系はメインループからの分岐で行うという制御を行います。メインループでは、常に両者のバッファであるバルクイン／バルクアウト転送用 RAM 領域を監視し、データが存在していればそのデータをバッファから出力します。

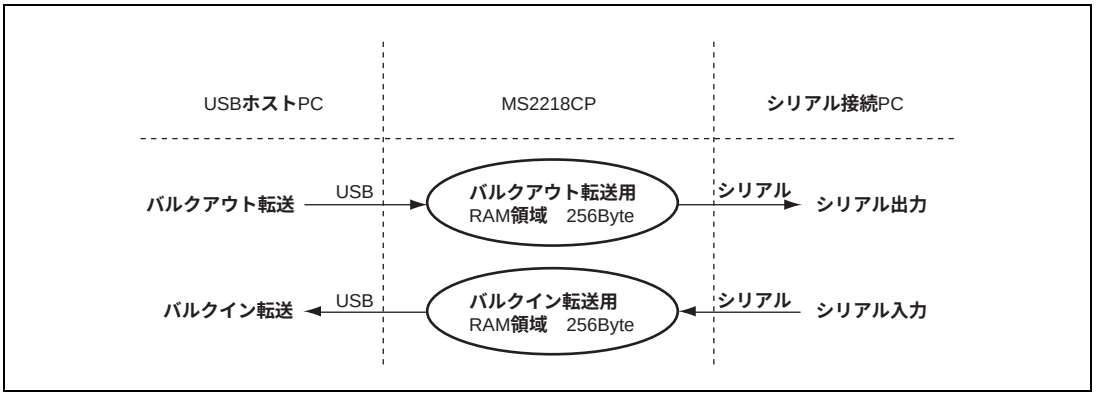


図 3.2　PC **間通信概要**

3.3 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 7 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** 3.1 に示します。各関数は、転送方式または機能毎に 1 つのファイルにまとめてあります。

表 3.1 **ファイル構成**

ファイル名	主な役割
StartUp.c	**ベクタテーブルの設定、マイコンの初期設定、リングバッファのクリア**
DoSerial.c	**シリアル送受信を実行、**SCI0 **モジュールの制御**
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**
DoControl.c	**コントロール転送を実行**
DoBulk.c	**バルク転送を実行**
DoRequestVenderCommand.c	**ベンダコマンドの処理**
SysMemMap.h	MS2218CP **のメモリマップのアドレス定義**
SetUsbInfo.h	USB **の構成を定義**
SetMacro.h	**マクロ定義**
SetSystemSwitch.h	**システムの動作設定**
H8S2218.h	H8S/2218 **レジスタ定義**
CatTypedef.h	**構造体定義**
CatProType.h	**プロトタイプ宣言**

3.4 関数の機能

表 3.2～**表** 3.8 に、各ファイルに含まれる関数と、その機能を示します。

表 3.2 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の呼び出し**
	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**
	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**
	SetControlOutContents	**ホストから送られたデータを書き換える**
	BE2ByteRead	2Byte **のデータを、ビッグエンディアンに変換する**
	LE2ByteRead	2Byte **のデータを、リトルエンディアンに変換する**
	ActBusReset	**バスリセット受信時にバッファとフラグ、**FIFO **のクリアを行う**
	SetUsbModule	USB **モジュールの初期設定**
	USBclear	**リングバッファ、フラグのクリアを行う**

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 3.3　StartUp.c

格納ファイル	関数名	機　　能
StartUp.c	SetPowerOnSection	BSC の設定、モジュール及びメモリの初期化を行い、メインループへ移行
	_INITSCT	初期値がある変数を、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 バスのプルアップ制御
	Error	エラー発生時、CPU をスリープモードに移行する
	SciInit	SCI0 の初期化
	Set_SMR	SCI0 の SMR レジスタの初期設定を行う
	ActBusVcc	VBUS 受信時の処理

パワーオンリセット、又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。ここでは主に H8S/2218 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 3.4　DoSerial.c

格納ファイル	関数名	機　　能
DoSerial.c	ActSerialOut	リードポインタからデータを読み出し、1 バイトづつ ExSerialOut に引数として渡す
	ActSerialIn	シリアル入力したデータをバルクイン転送用領域に書き込む
	WriteBulkInArea	データをバルクイン転送用領域に書き込む
	ExSerialOut	1 バイトデータを SCI0 からシリアル出力する

DoSerial.c では、シリアル送受信を行います。SCI0 モジュールの制御も行います。

表 3.5　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

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：Renesas Technology Corp.）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。

表 3.6　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**
	ActControlInOut	**コントロール転送のデータステージとステータスステージを、**ActControlIn **と** ActControlOut **に振り分ける**

コントロール転送の割り込み（SETUP TS）が入ると、ActControl がコマンドを取得し、DecStandardCommands でデコードを行いコマンドの転送方向を判別します。その後、コントロール転送の割り込み（EP0o TS、EP0i TR、EP0i TS）が発生すると、ActControlInOut がコマンドの転送方向により、ActControlIn または ActControlOut を呼び出しデータステージと、ステータスステージを行います。

表 3.7　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制御する**
	ActBulkIn	**バルクイン転送を制御する**

バルク転送に関する処理を行います。データ送受信および、フローの制御を行います。

表 3.8　DoRequestVenderCommand.c

格納ファイル	関数名	機　　能
DoRequestVender Command.c	DecVenderCommands	**ベンダコマンドの対応を行う**

ベンダ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、日立超 LSI システムズ社製の USB シリアル変換ドライバがサポートする 4 つのベンダコマンドに対する処理を行います。詳細は**「**4.8　**ベンダコマンド」**を参照ください。

図 3.3 に、**表** 3.2～**表** 3.8 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しています。また、複数の関数が同一の関数を呼び出す事もあります。定常状態では、SetPowerOnSection が他の関数を呼び出し、USB 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。さらに、SCI0 割り込みでは、ActSerialIn 関数を呼び出します。**図** 3.3 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** 4 **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。

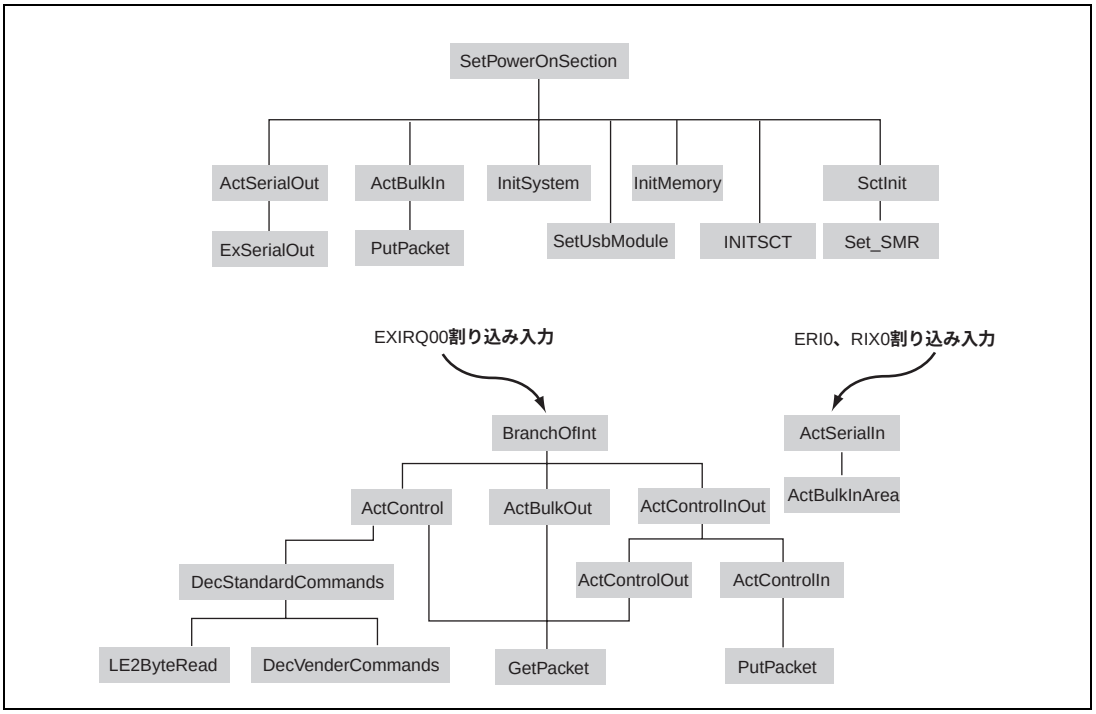

SetPowerOnSection
ActSerialOut
ActBulkIn
InitSystem
InitMemory
SctInit
ExSerialOut
PutPacket
SetUsbModule
INITSCT
Set_SMR
EXIRQ00割り込み入力
ERI0、RIX0割り込み入力
BranchOfInt
ActSerialIn
ActBulkInArea
ActControl
ActBulkOut
ActControlInOut
DecStandardCommands
ActControlOut
ActControlIn
LE2ByteRead
DecVenderCommands
GetPacket
PutPacket

図 3.3　**関数の相関関係**

4. 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。

4.1 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** 4.1 に SetPowerOnSection のフローチャートを示します。

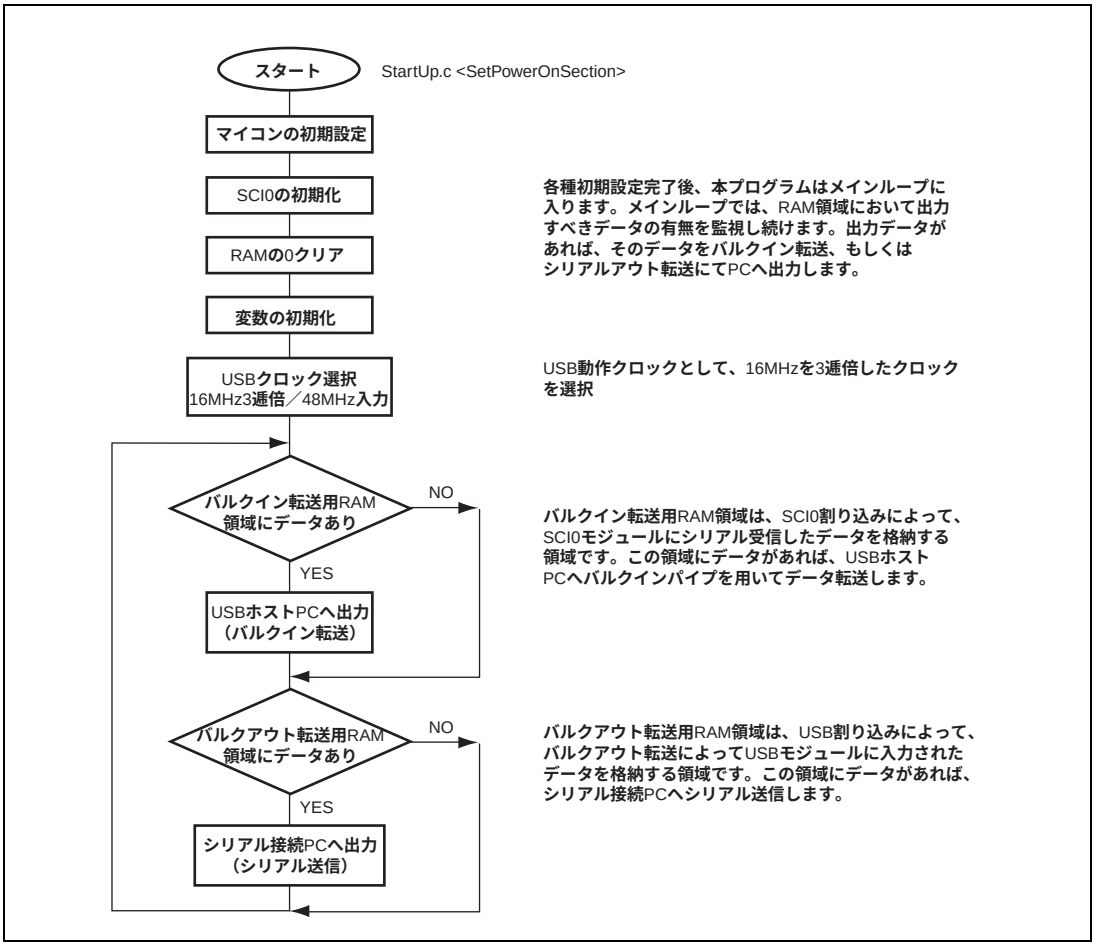


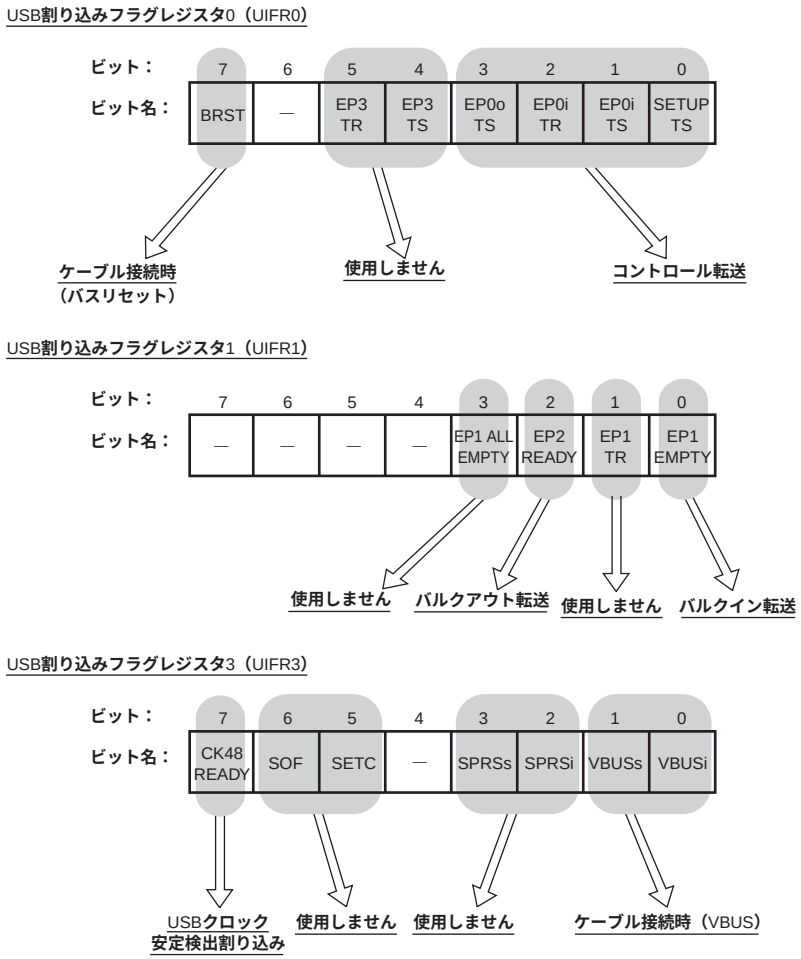
図 4.1 **メインループ**

4.2 割り込みの種類

「3.1 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要因は"割り込みフラグレジスタ 0～3（UIFR0～3）"、"シリアルステータスレジスタ（SSR0）"によって示され、USB 系が 5 種類、シリアル系が 2 種類です。

USB 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 4.2** に割り込みフラグレジスタと、USB 通信の関係を示します。

なお、バルクイン転送は、本サンプルプログラムでサポートしていますが、割り込み動作ではなく、メインルーチンからの分岐によって動作します。そのため、バルクイン割り込みはディゼーブルに設定し、EP1 EMPTY フラグを見ることでバルクイン転送を起動します。EP1 TR は使用しません。



【注】 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はサポートしていません。

図 4.2 USB **割り込みフラグの種類**

シリアル割り込み要因が発生した場合も同様にシリアルステータスレジスタの対応するフラグに 1 がセットされ、CPU に対して割り込み要求します。本サンプルプログラムでは、送信データエンプティ、受信データフル、つまりシリアル送信、シリアル受信機能をサポートします。しかし、シリアル送信は割り込み動作ではなくメインループからの分岐により動作するため、フラグとして使用し、割り込み機能は使用しません。

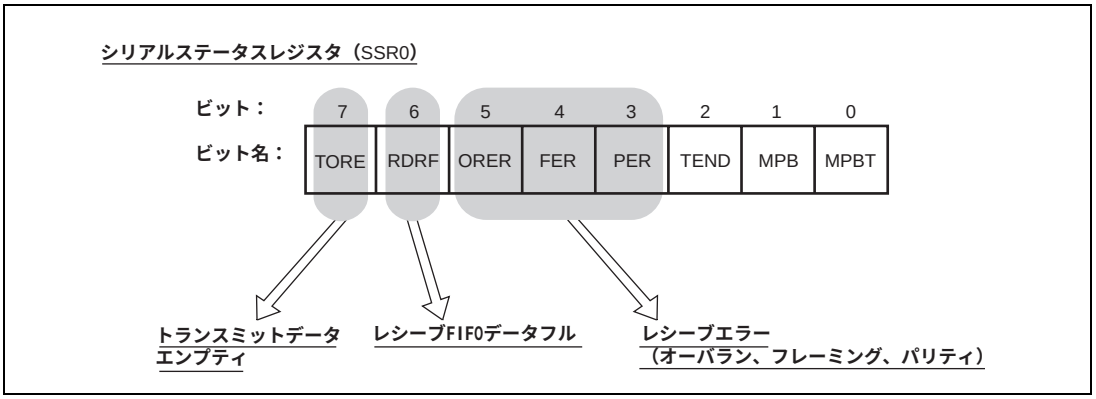


図 4.3　シリアル割り込みフラグの種類

4.2.1　各転送への分岐方法

本サンプルプログラムでは、メインループからの分岐、もしくは USB モジュール、SCI0 モジュールからの割り込みによって転送方式を決定しています。**表 4.1** に各転送方式の呼び出し方法をまとめます。

メインループからの分岐の場合は、直接関数が呼ばれます。シリアルアウト転送（ActSerialOut）、バルクイン転送（ActBulkIn）がこれに当たります。

USB 割り込みによる各転送方式への分岐は UsbMain.c の BranchOfInt が行います。USB 動作クロック安定検出時（SetUsbModule）、ケーブル接続時（ActBusReset、ActBusVcc）、コントロール転送（ActControl）、バルクアウト転送（ActBulkOut）がこれに当たります。

SCI0 割り込みによる各転送方式への分岐は、SCI0 割り込みは ERI0、RXI0、TXI0 と転送方式毎に割り込み要因が設定されているため、直接関数を呼び出します。シリアルイン転送（ActSerialIn）がこれに当たります。

表 4.1　各転送方式の呼び出し方法

モジュール	転送方式	呼び出し方法
USB	USB 動作クロック安定検出	USB 割り込み
	ケーブル接続（バスリセット）	USB 割り込み
	ケーブル接続（BusVcc）	USB 割り込み
	コントロール転送	USB 割り込み
	バルクアウト転送	USB 割り込み
	バルクイン転送	メインループからの分岐
SCI1	シリアルイン転送	SCI0 割り込み
	シリアルアウト転送	メインループからの分岐

表 4.2 に USB 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 4.2　USB **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	EP3 TR	−
	4	EP3 TS	−
	3	EP0o TS	ActControlInOut
	2	EP0i TR	ActControlInOut
	1	EP0i TS	ActControlInOut
	0	SETUP TS	ActControl
UIFR1	7	−	−
	6	−	−
	5	−	−
	4	−	−
	3	EP1 ALL EMPTY	−
	2	EP2 READY	ActBulkOut
	1	EP1 TR	−
	0	EP1 EMPTY	ActBulkIn
UIFR3	7	CK48 READY	SetUSBModule
	6	SOF	ActIdleCount
	5	SETC	−
	4	−	−
	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。従って、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、**「**4.5　**コントロール転送」**をご覧ください。

表 4.3 に SCI0 割り込みの種類と、呼び出す関数を示します。

表 4.3　SCI0 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
SSR1	7	TDRE	－（メインループからの分岐）
	6	RDRF	ActSerialOut
	5	ORER	ActSerialOut
	4	FER	ActSerialOut
	3	PER	ActSerialOut
	2	TEND	－
	1	MPB	－
	0	MPBT	－

以下の項では、USB、SCI0 の各転送方式ごとにアプリケーション側ファームウェアの動作詳細を説明します。

4.3 USB 動作クロック安定検出割り込み

USB モジュールストップ解除後の 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に発生します。割り込み受信後、各割り込みを設定して、USB ケーブル接続待ち状態へ以降します。

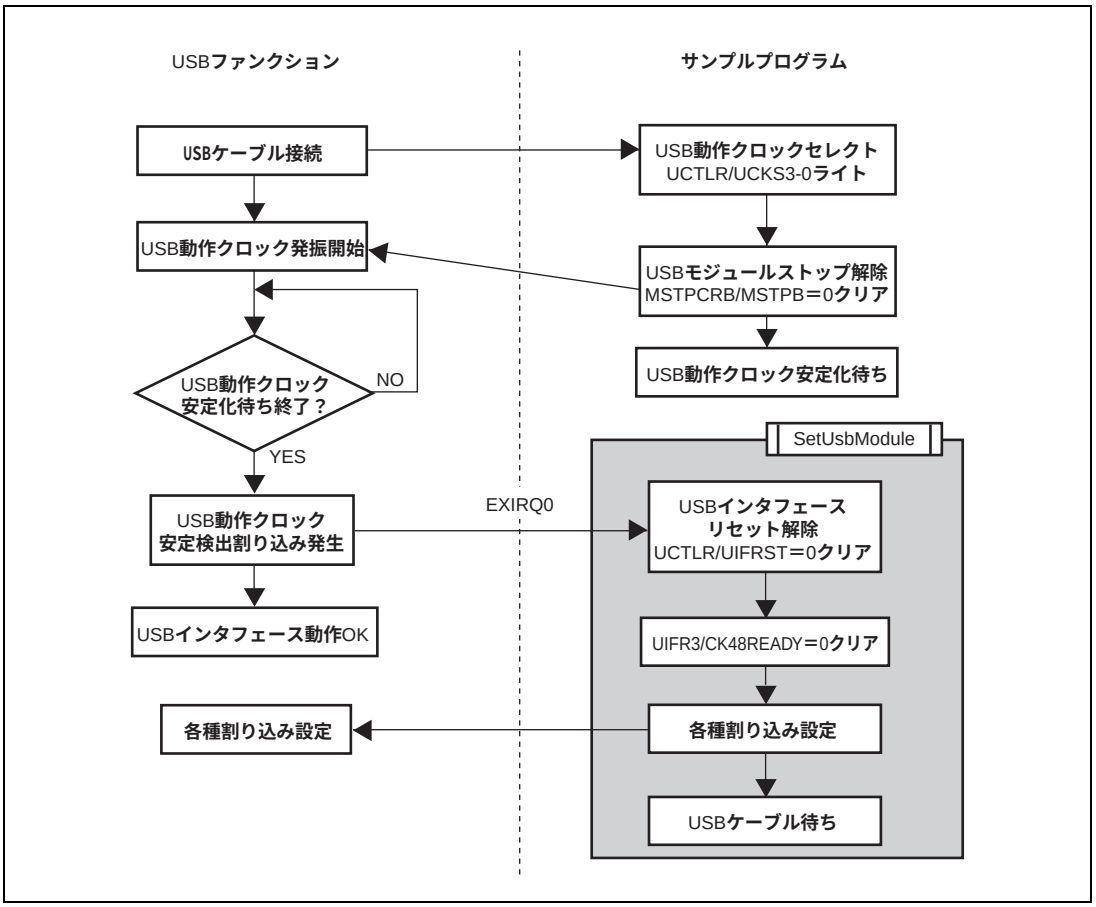


図 4.4 USB 動作クロック安定検出割り込み

4.4 ケーブル接続時（BRST、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事を認識します。（**図** 4.6 参照）

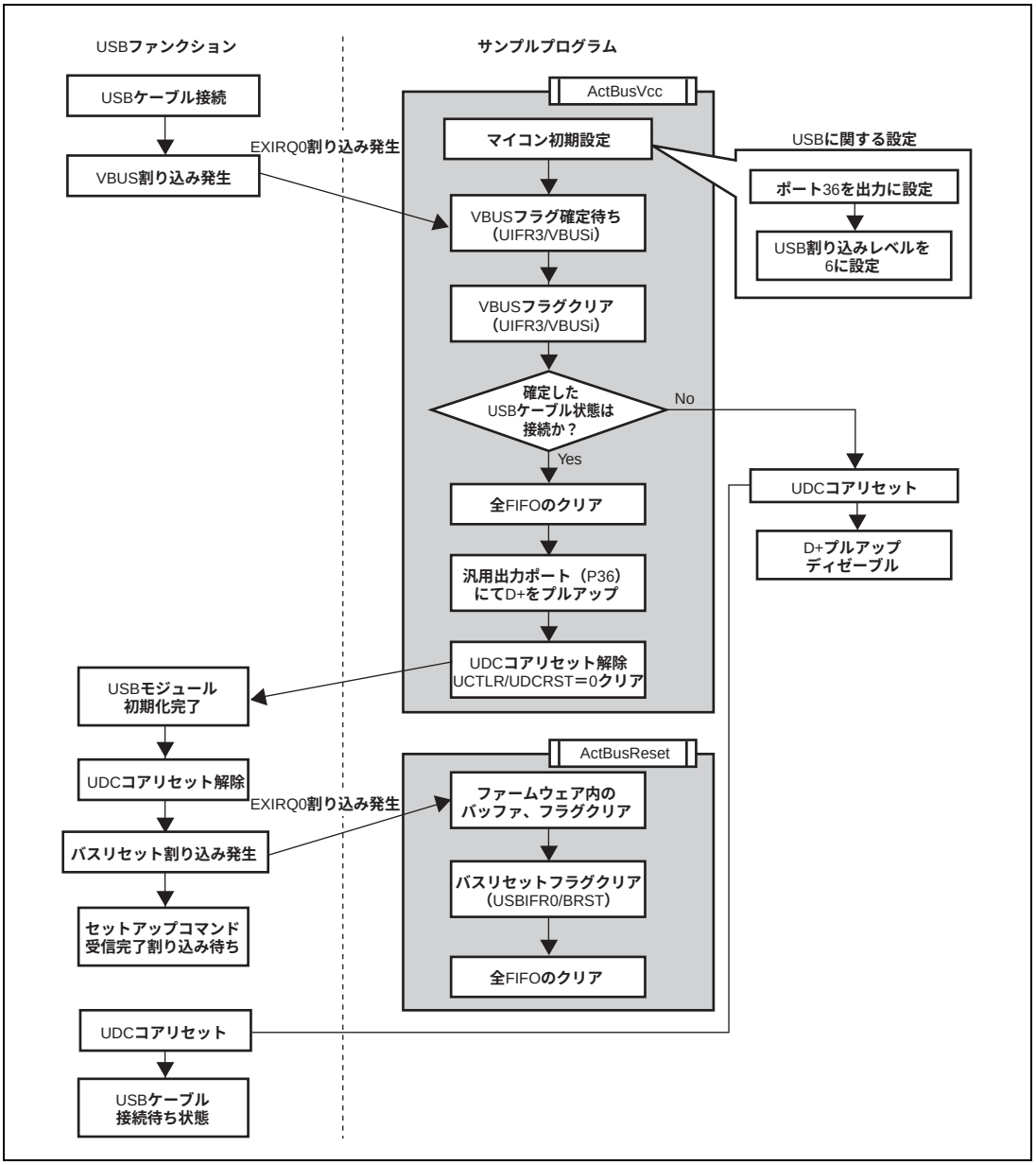


図 4.5 **ケーブル接続時割り込み**

4.5 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 つに分ける事ができます。（**図** 4.6 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。

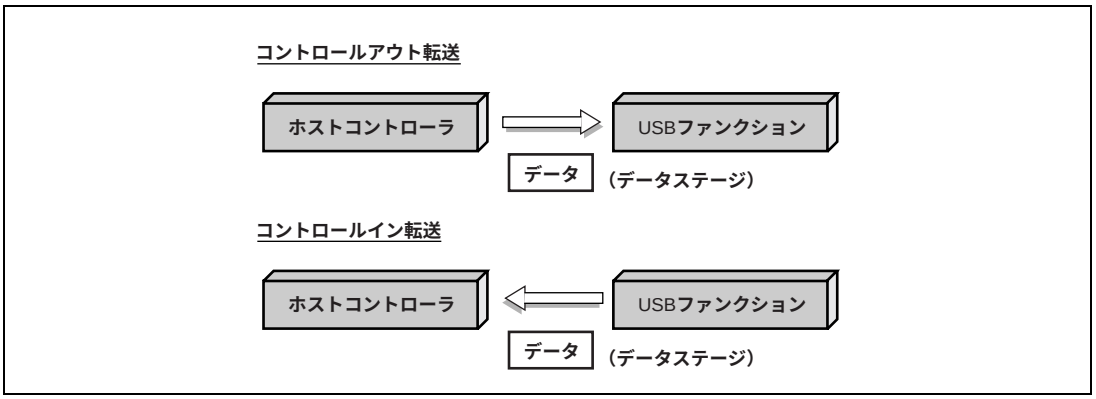


図 4.6 **コントロール転送**

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** 4.7）。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事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** 4.1 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** 4.7 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。

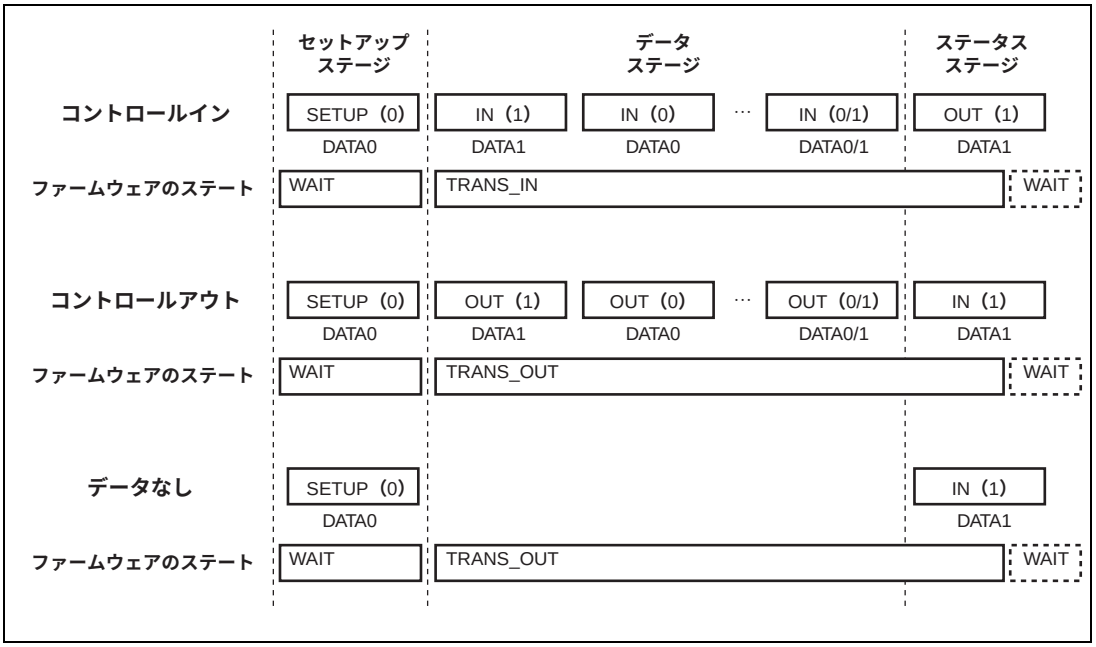


図 4.7　**コントロール転送における各ステージ**

4.5.1　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
  GetLineCoding　（TRANS_IN）　ベンダコマンド
- コントロールアウトとなるコマンド　SetLineCoding　（TRANS_OUT）ベンダコマンド
  SetControlLineState　（TRANS_OUT）ベンダコマンド
  SendBreak　（TRANS_OUT）ベンダコマンド

図 4.8 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。

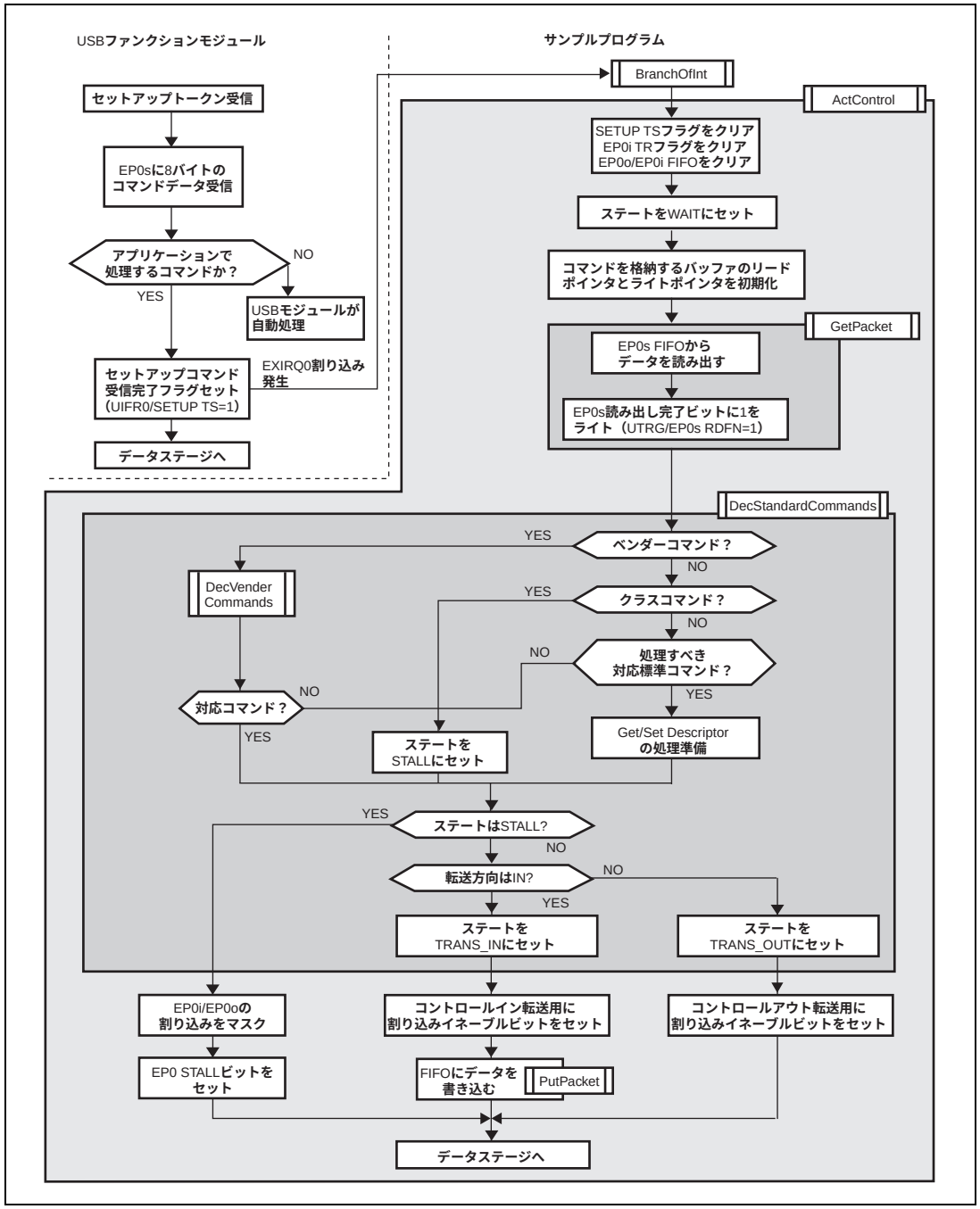


図 4.8　**セットアップステージ**

4.5.2 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 4.9、**図** 4.10 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。

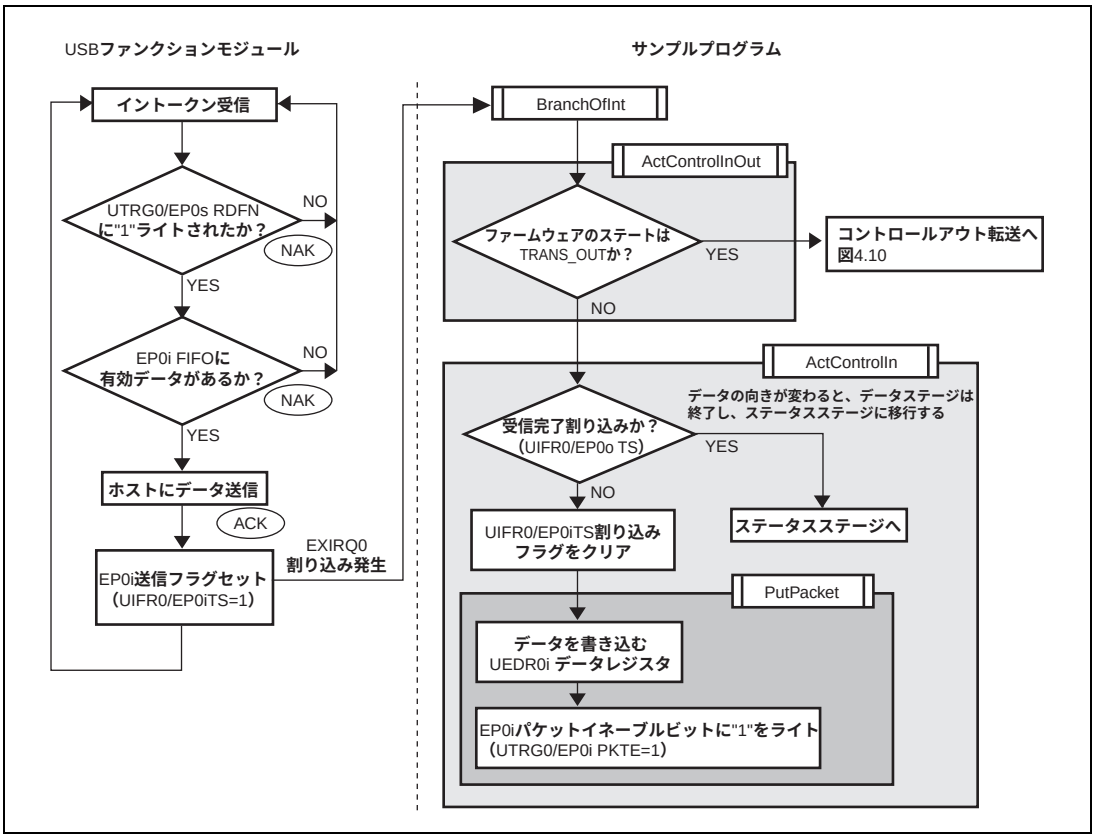


図 4.9 **データステージ（コントロールイン転送）**

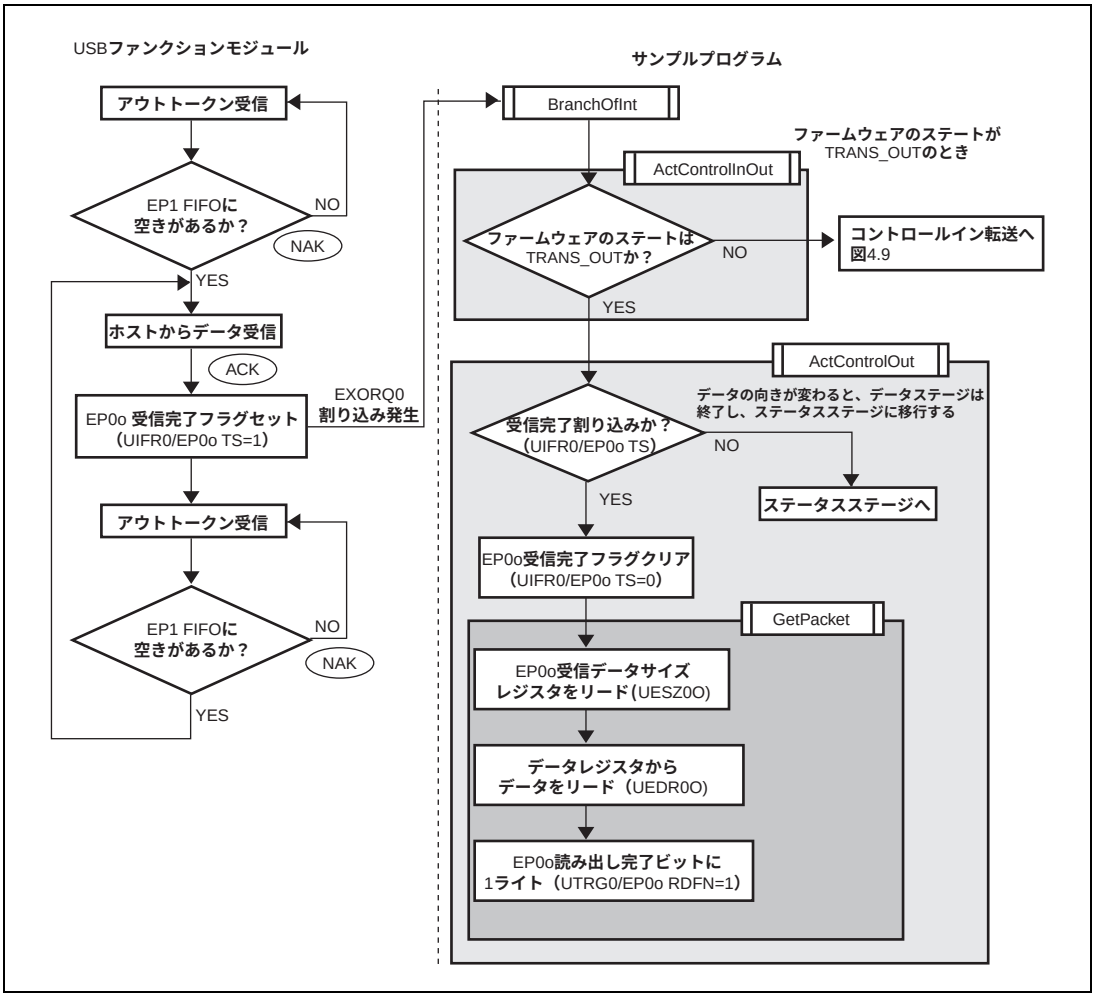


図 4.10　**データステージ（コントロールアウト転送）**

4.5.3 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。

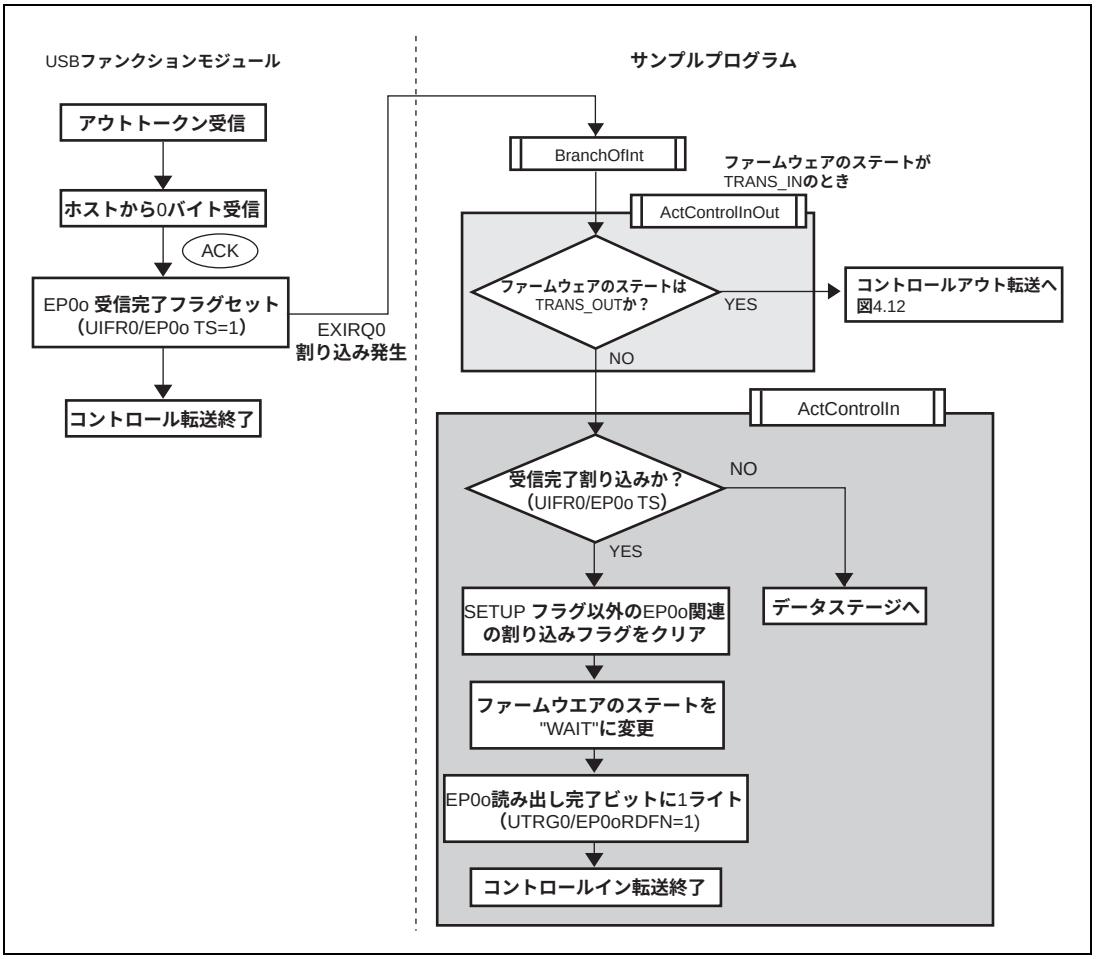


図 4.11 **ステータスステージ（コントロールイン転送）**

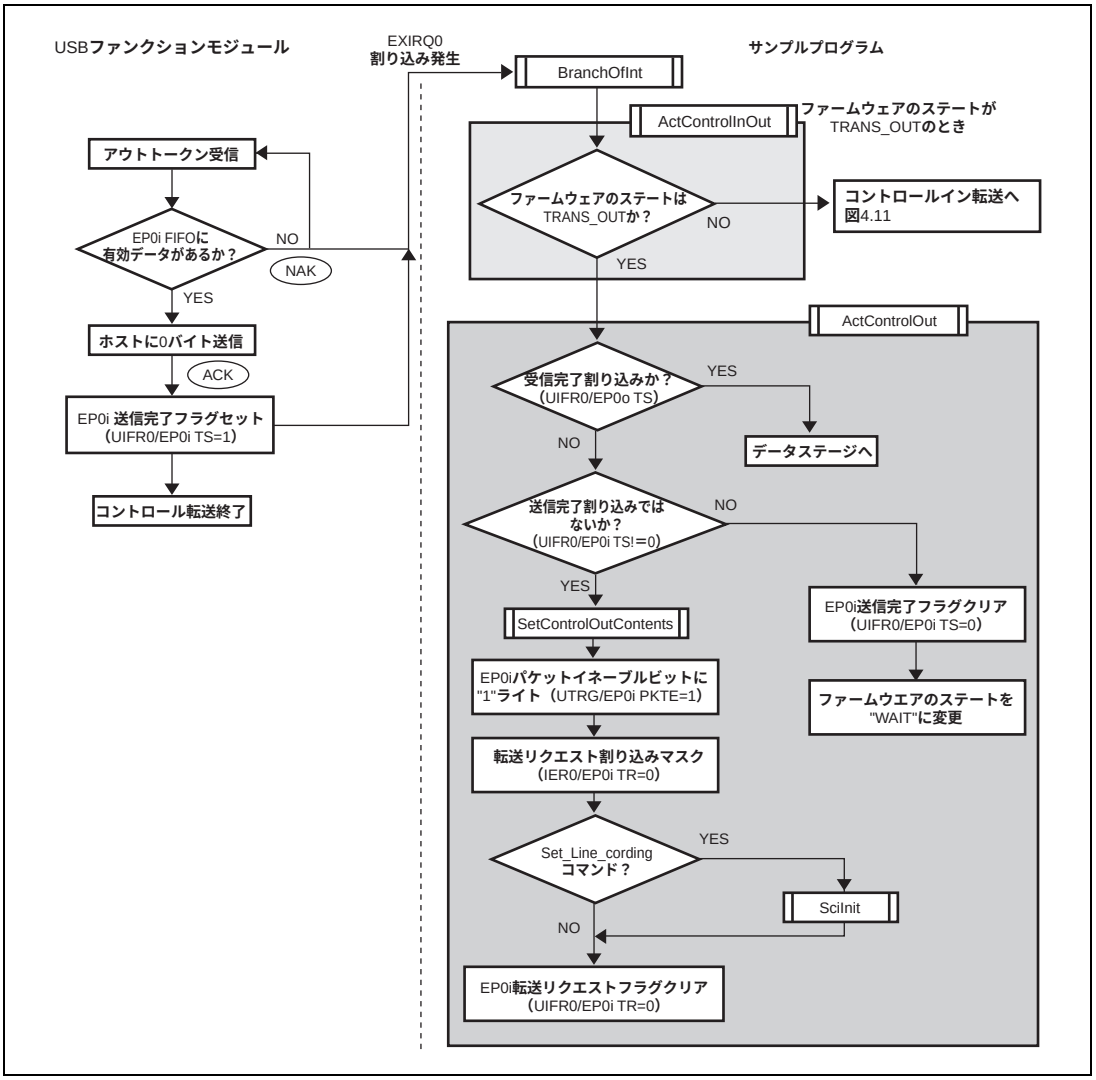


図 4.12　**ステータスステージ（コントロールアウト転送）**

4.6 バルク転送

バルク転送には、割り込みフラグレジスタ 1 のビット 0～2 を使用します（本プログラムではバルクイン転送は割り込み起因での起動ではないためビット 0、1 は使用しません）。バルク転送は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** 4.13 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。

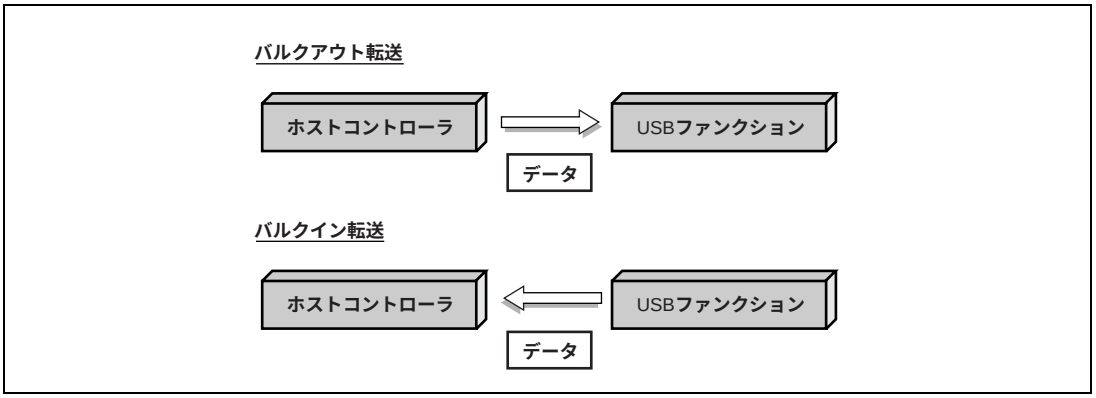


図 4.13 **バルク転送**

4.6.1 バルクアウト転送

図 4.14 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。

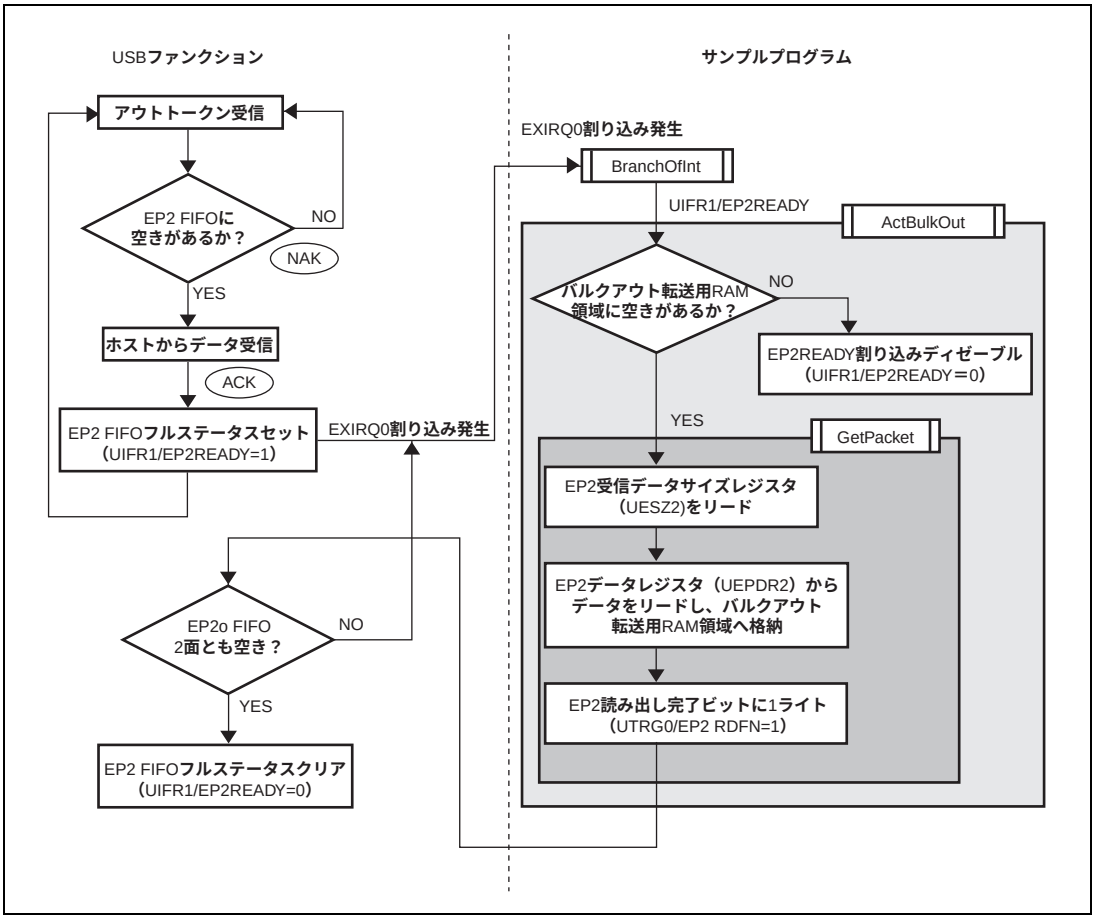


図 4.14 **バルクアウト転送**

4.6.2 バルクイン転送

図 4.15 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。バルクアウト転送と異なり、割り込み起因で起動するのではなく、メインループから呼ばれます。

バルクイン転送用 RAM 領域に格納されているデータを EP1 データレジスタに書き込みます。その後、この RAM 領域に空き領域がなくシリアルイン転送がディゼーブル状態の場合は、UEDR1 データレジスタにデータを書き込んだことで、この RAM 領域に空き領域ができたか確認します。空き領域ができた場合はシリアルイン転送をイネーブルにします。

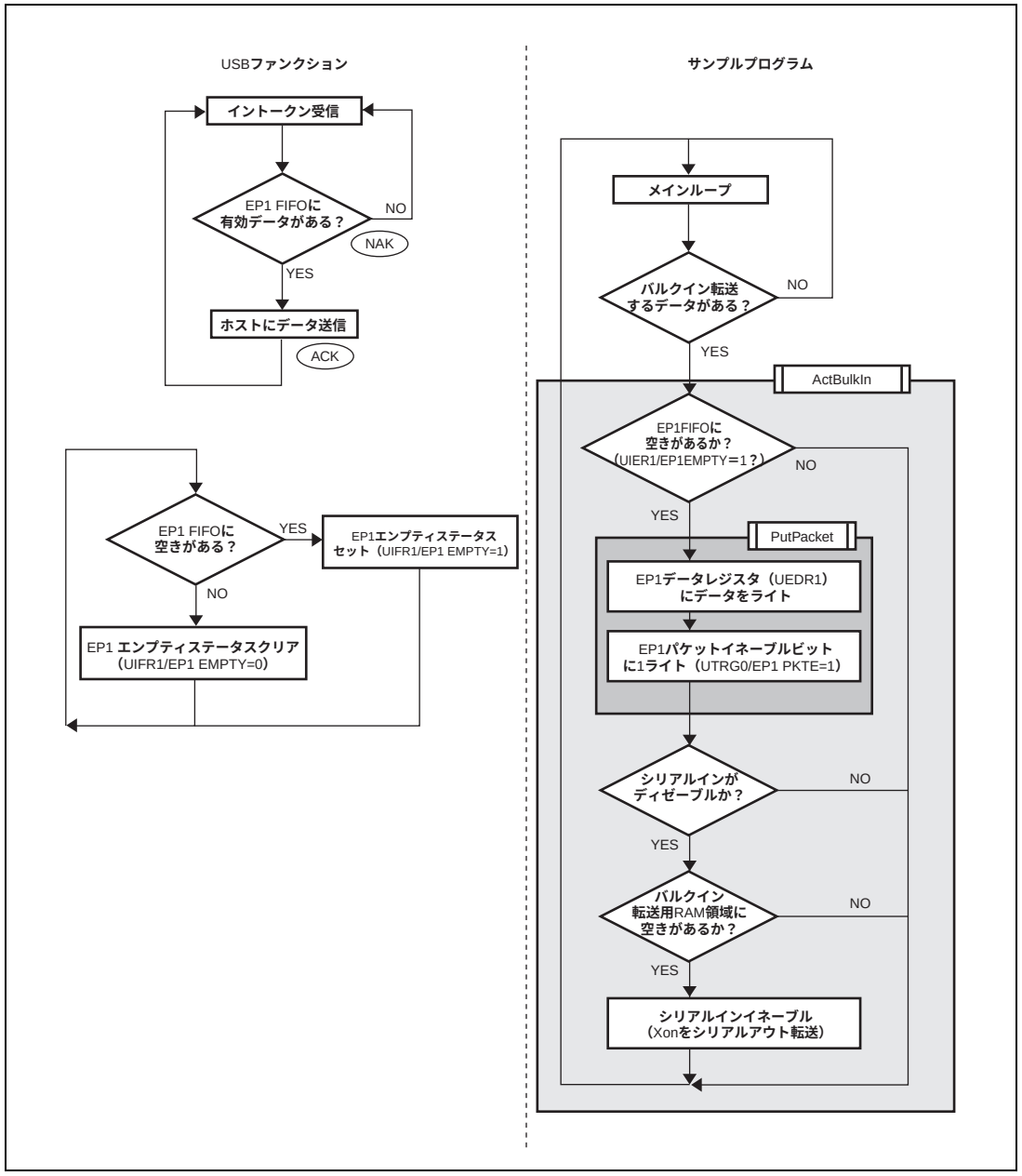


図 4.15　**バルクイン転送**

4.7 シリアル転送

シリアル転送には、SCI0 モジュールを使います。シリアルアウト転送はメインループからの分岐によって実行され、シリアルイン転送は割り込みによって実行されます。シリアルイン転送は、シリアルステータスレジスタ（SSR0）の RDRF フラグを使用します。

4.7.1 シリアルアウト転送

図 4.16 にシリアル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。シリアルアウト転送は、バルクアウト転送用 RAM 領域にデータがあれば、メインループから分岐して ActSerialOut 関数を呼び出し、SCI0 モジュールを制御してシリアルアウト転送によりデータを送信します。もし、バルクアウト転送用 RAM 領域に空き容量がない状態でバルクアウト転送がディゼーブル状態である場合、このシリアルアウト転送によって空き容量が確保できたか確認し、確保できた場合は、バルクアウト転送をイネーブルにします。

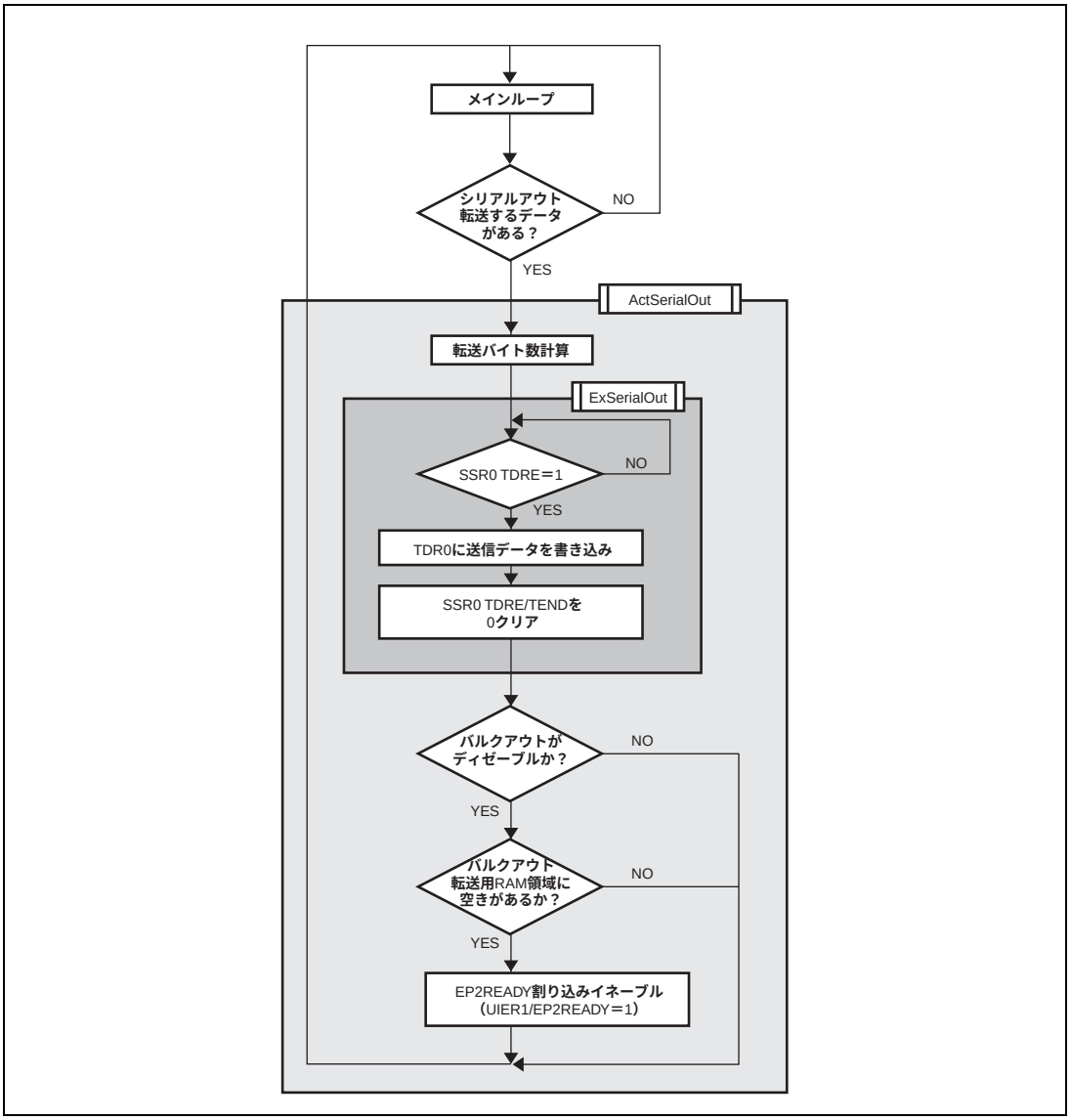


図 4.16　**シリアルアウト転送**

4.7.2 シリアルイン転送

図 4.17、**図** 4.18 にシリアル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。ERI0、RXI0 の受信割り込みが発生した場合、ActSerialIn 関数が呼ばれます。

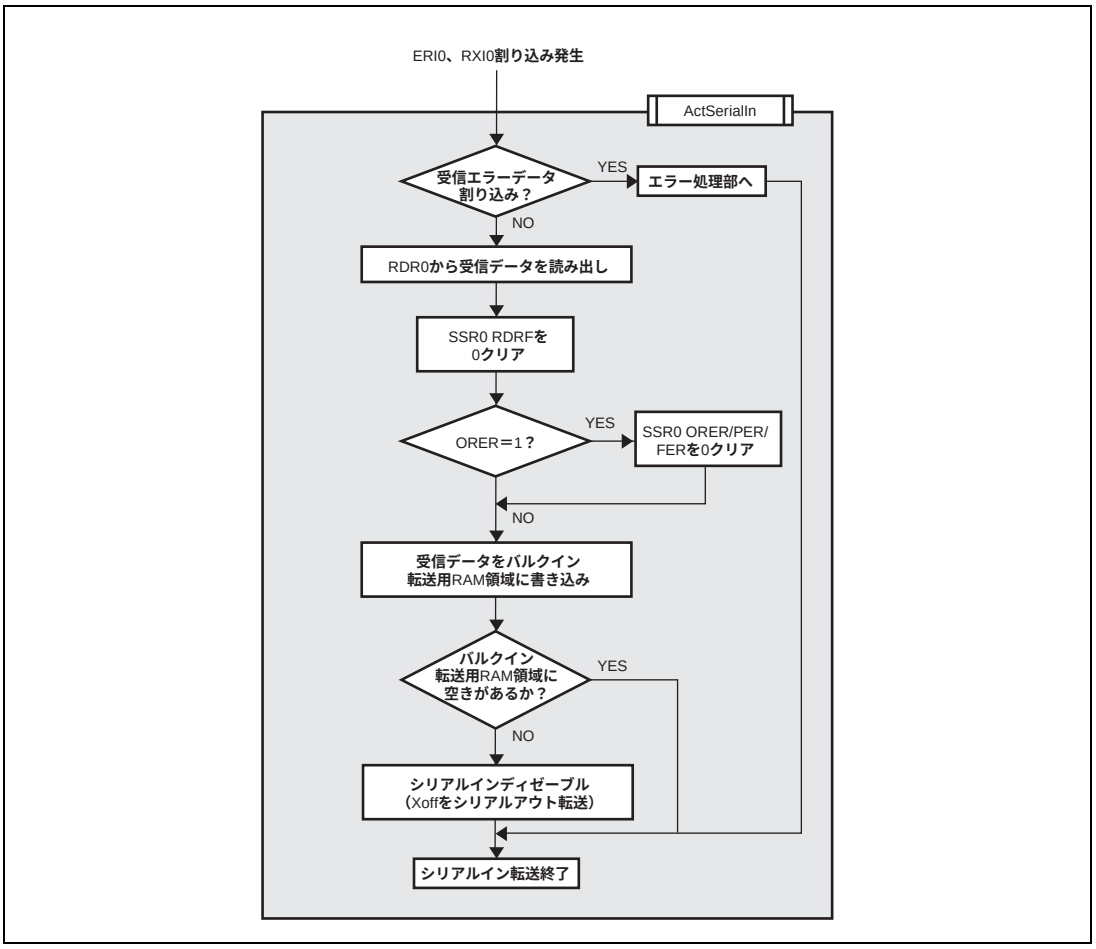


図 4.17 **シリアルイン転送（受信データ処理部）**

ERI1 受信では、ORER が発生した場合は、RXI0 受信時と同様に RDR0 からデータを読み出します。ORER が伴わない受信エラーの場合は、RDR0 の値をダミーリードして捨てて、エラーフラグをクリアします。この時、ブレーク信号も受信している場合は、シリアル受信をディゼーブルにし、FER フラグをクリアせずに関数を抜けます。この場合、FER は 1 が立ったままなので、ブレーク信号を受信しなくなるまで、連続して割り込みが発生し、ActSerialIn 関数が呼ばれ続けます。また、この期間中に USB 割り込みを受け付けるために、USB と SCI0 の割り込み優先順位を逆転させます。

ORER エラーもしくは、正常受信の場合は、RDR0 からデータを受信し、バルクイン転送用 RAM 領域に書き込みます。その後、この RAM 領域の空き容量をチェックし、空き容量がない場合はデータの欠落を防止するために、シリアル接続 PC へ Xoff を出力し、シリアルイン転送をディゼーブルにします。

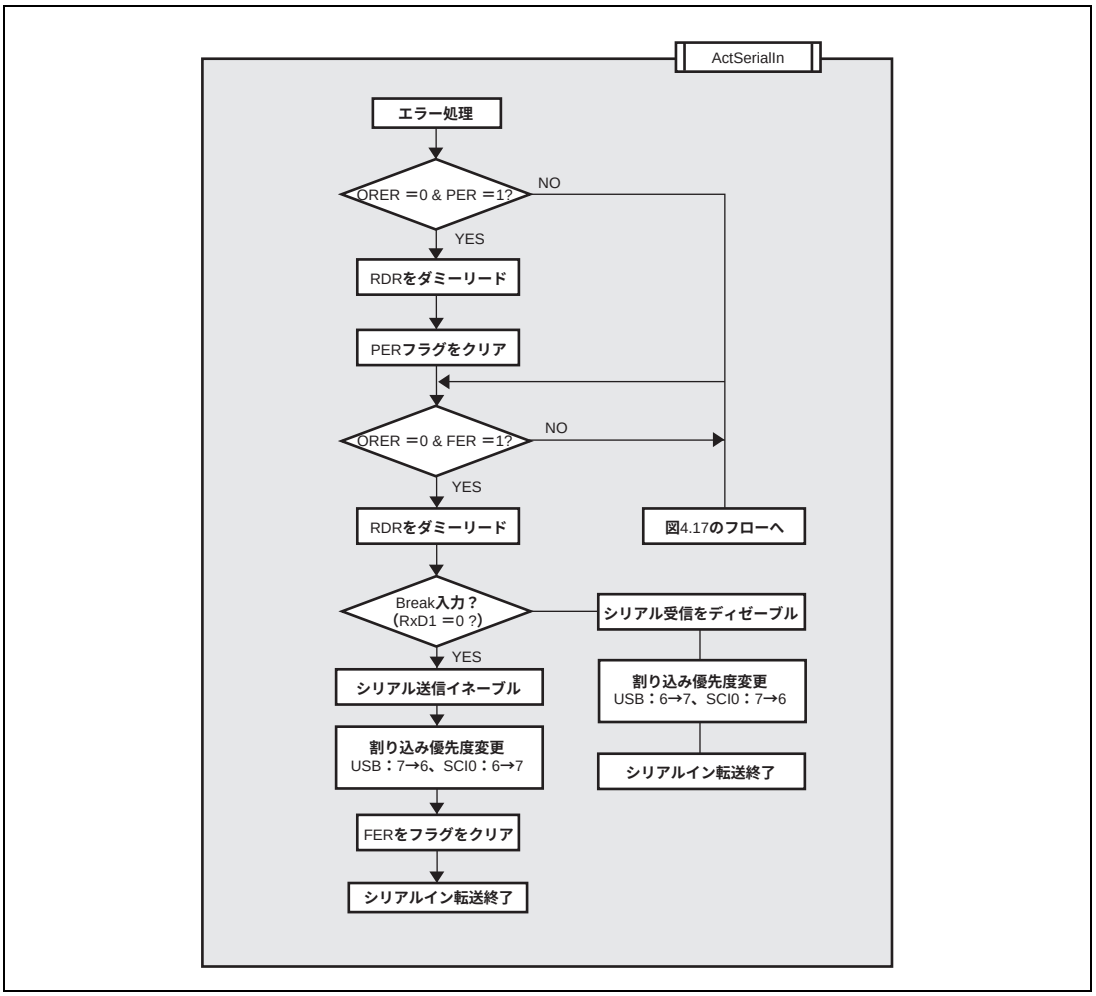


図 4.18　シリアルイン転送（エラー処理部）

4.8 ベンダコマンド

本サンプルプログラムでは、日立超 LSI システムズ社製の USB シリアル変換ドライバがサポートする 4 つのベンダコマンドをデコードします。

表 4.4 に USB シリアル変換ドライバが規定する 4 つのベンダコマンドを示します。

表 4.4（a） ベンダリクエスト

bmRequestType	bRequest	wValue	wIndex	wLength	Data
01000001b	SET_LINE_CODING	Zero	Interface	8	Line Coding Structure
11000001b	GET_LINE_CODING	Zero	Interface	8	Line Coding Structure
01000001b	SET_CONTROL_LINE_STATE	Control Signal Bitmap	Interface	Zero	None
01000001b	SEND_BREAK	Duration of Break	Interface	Zero	None

表 4.4（b） ベンダリクエストコード

bRequest	Value
SET_LINE_CODING	0
GET_LINE_CODING	1
SET_CONTROL_LINE_STATE	2
SEND_BREAK	3

次節以降に、各コマンドの詳細を示します。

4.8.1 SetLineCoding

本リクエストは、調歩同期式のデータ伝送で使用されるパラメータを設定します。

bmRequestType	bRequest	wValue	wIndex	wLength	Data
01000001b	SET_LINE_CODING	Zero	Interface	8	Line Coding Structure

Line Coding Structure の定義を**表** 4.5 に示します。

本サンプルプログラムでは、このコマンド受信により、受信した Line Coding Structure の設定で SCI0 を再起動します。

表 4.5　Line Coding Structure

Offset	Field	Size	Value	Description
0	DwDTERate	4	Number	**データ端末の速度**（bps）
4	BcharFormat	1	Number	**ストップビット** 0 - 1 Stop bits 1 - 1.5 Stop bits 2 - 2 Stop bits
5	BparityType	1	Number	**パリティ** 0 - None 1 - Odd 2 - Even 3 - Mask 4 - Space
6	BdataBits	1	Number	**データビット**（5, 6, 7, 8）
7	BflowType	1	Number	**フローコントロール** 0 - **ソフトウェア** or **なし** 1 - **ハードウェア**

4.8.2　GetLineCoding

本リクエストは、ホストがデバイスの現在のパラメータについて確認するコマンドです。

本サンプルプログラムは、このコマンドを受信すると、**表** 4.6 で示すデフォルト値を返します。

bmRequestType	bRequest	wValue	wIndex	wLength	Data
11000001b	GET_LINE_CODING	Zero	Interface	8	Line Coding Structure

表 4.7　Line Coding Structure **のデフォルト値**

Offset	Field	Size	Value	Description
0	DwDTERate	4	0x1C200	**データ端末の速度**（115200bps）
4	BcharFormat	1	0x0	**ストップビット**（1 Stop bit）
5	BparityType	1	0x0	**パリティ**（None）
6	BdataBits	1	0x8	**データビット**（8）
7	BflowType	1	0x0	**フローコントロール** （**ソフトウェア** or **なし**）

4.8.3 SetControlLineState

本リクエストは、制御信号を設定します。

bmRequestType	bRequest	wValue	wIndex	wLength	Data
01000001b	SET_CONTROL_LINE _STATE	Control Signal Bitmap	Interface	Zero	None

表 4.7 Control Signal Bitmap

Bit Position	Description
D15...D2	**予約（0 にリセット）**
D1	DCE **の送信機能を制御** 0 - RTS OFF 1 - RTS ON
D0	DTE **がレディ状態かの通知** 0 - DTR OFF 1 - DTR ON

H8S/2218 が RTS、DTR の信号線を持たないため、本サンプルプログラムでは、このリクエストについてはデコードのみを行い、DCE の制御は行いません。

なお、本サンプルプログラムでは、D1=1＆D0=1 を検知することで、USB ホスト PC 側ハイパーターミナルの通信準備が完了したと認識し、このタイミングで、バルクイン、バルクアウト転送データ領域を示すポインタ、および本サンプルプログラムの持つ内部フラグの初期化を行います。

4.8.4 SendBreak

本リクエストは、デバイスにブレーク信号を発生させます。

bmRequestType	bRequest	wValue	wIndex	wLength	Data
01000001b	SEND_BREAK	Duration of Break	Interface	Zero	None

wValue フィールドは、ブレーク信号送出時間（msec）を記入。wValue が 0xFFFF の場合、デバイスは、wValue が 0x0000 の SendBreak リクエストを受信するまで、ブレーク信号を送出し続けます。

本サンプルプログラムでは、このリクエストのデコードは行いますが、ブレーク送信は行いません。

5. アナライザのデータ

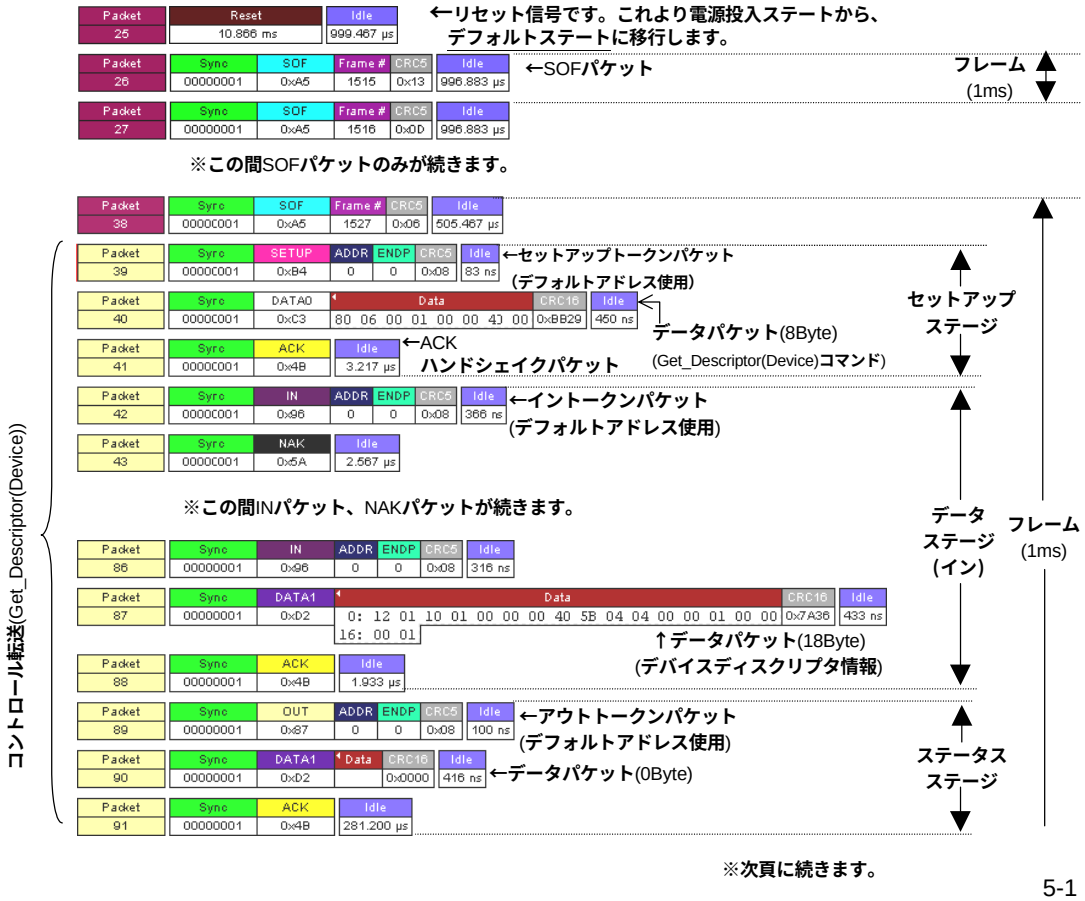
この章では、H8S/2218 内蔵 USB ファンクションモジュールを使用して、CATC 社製 USB プロトコルアナライザ「USB Advisor」（国内：（株）東陽テクニカ(http://www.toyo.co.jp/)）を用いた測定を行い、実際にバスを流れているデータについて「デバイス接続時のコントロール転送」および「ベンダコマンド送信時のバルクアウト転送」を例に説明します。

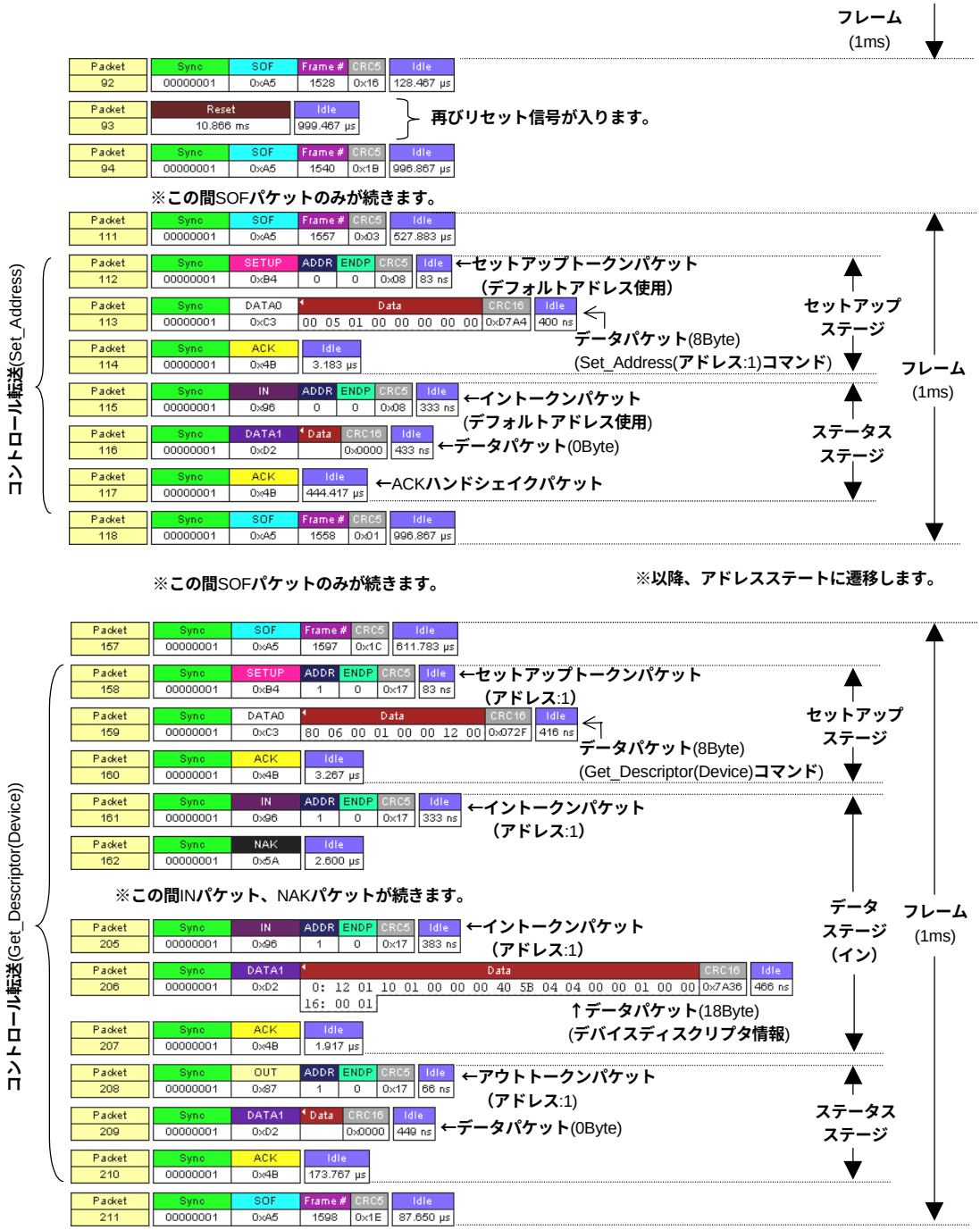
【注】　各パケットの前部にある「Packet#」は測定時のパケット通し番号です。
後部にある「Idle」はパケット間のアイドルとなります。

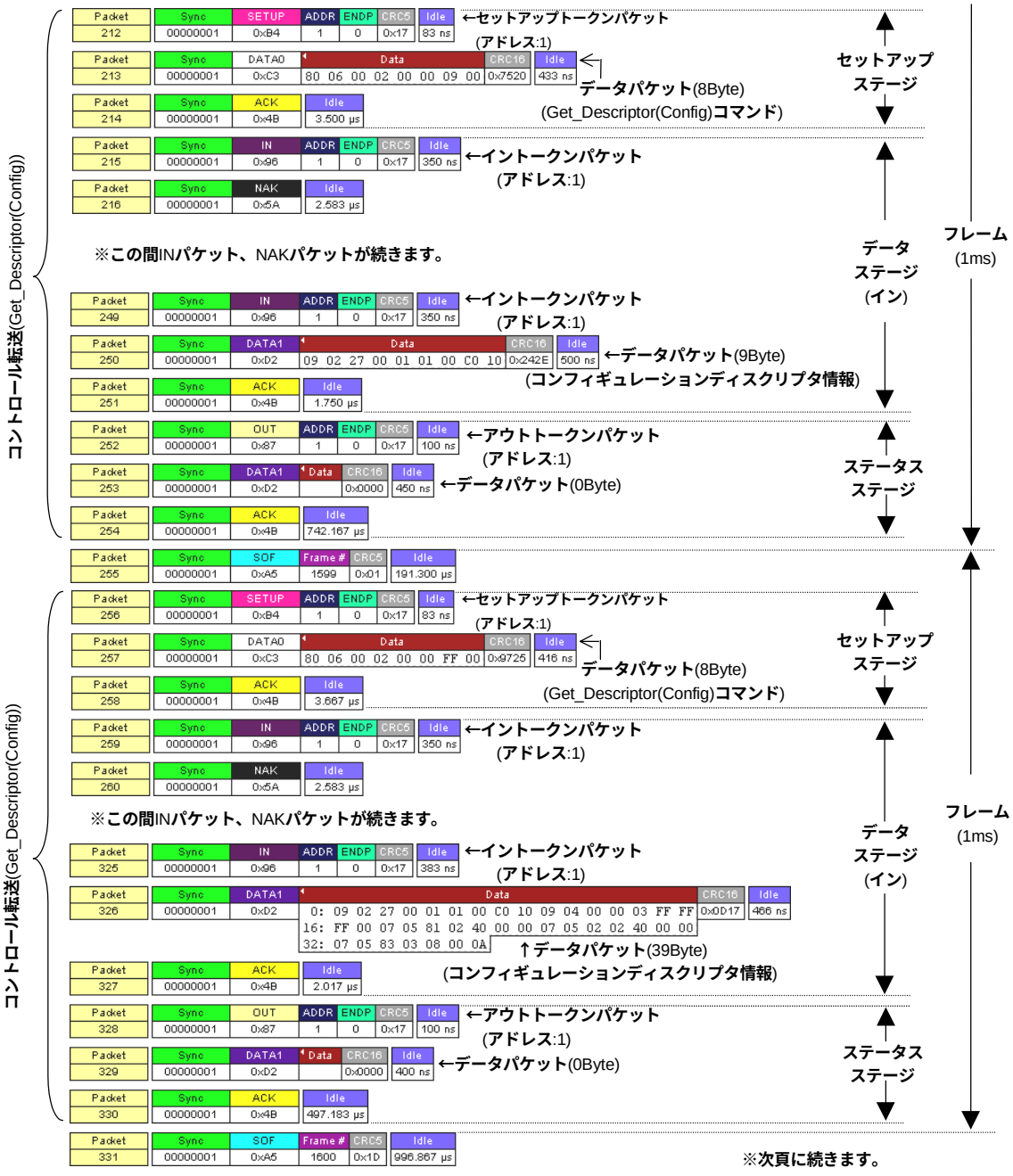
5.1 デバイス接続時のコントロール転送

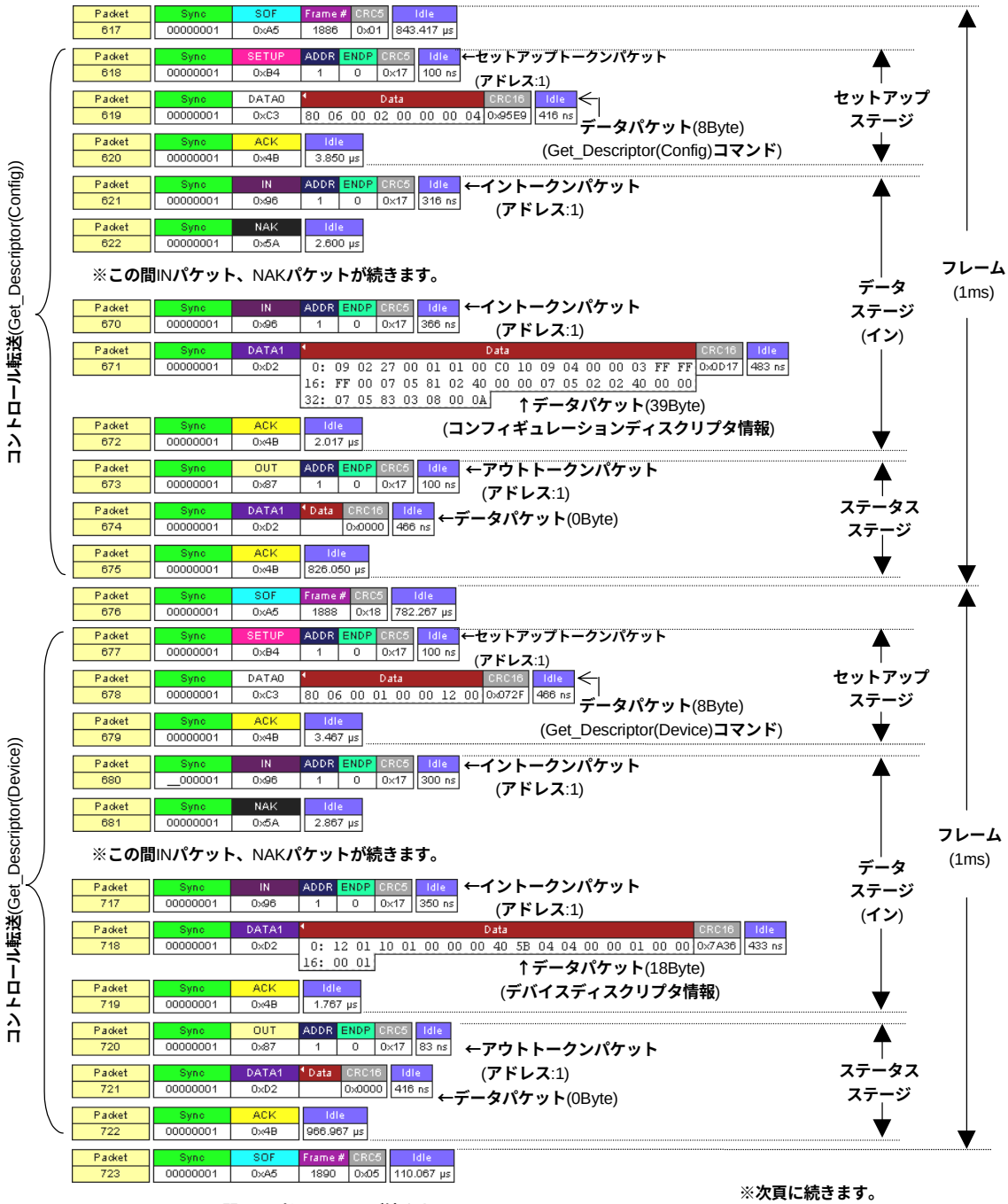
図 5.1 は本デバイスをホストコントローラに接続し、Vbus に電源が供給されている（電源投入ステート）状態から、デバイスが使用可能になる状態（構成ステート）に至るまでを測定したものです。

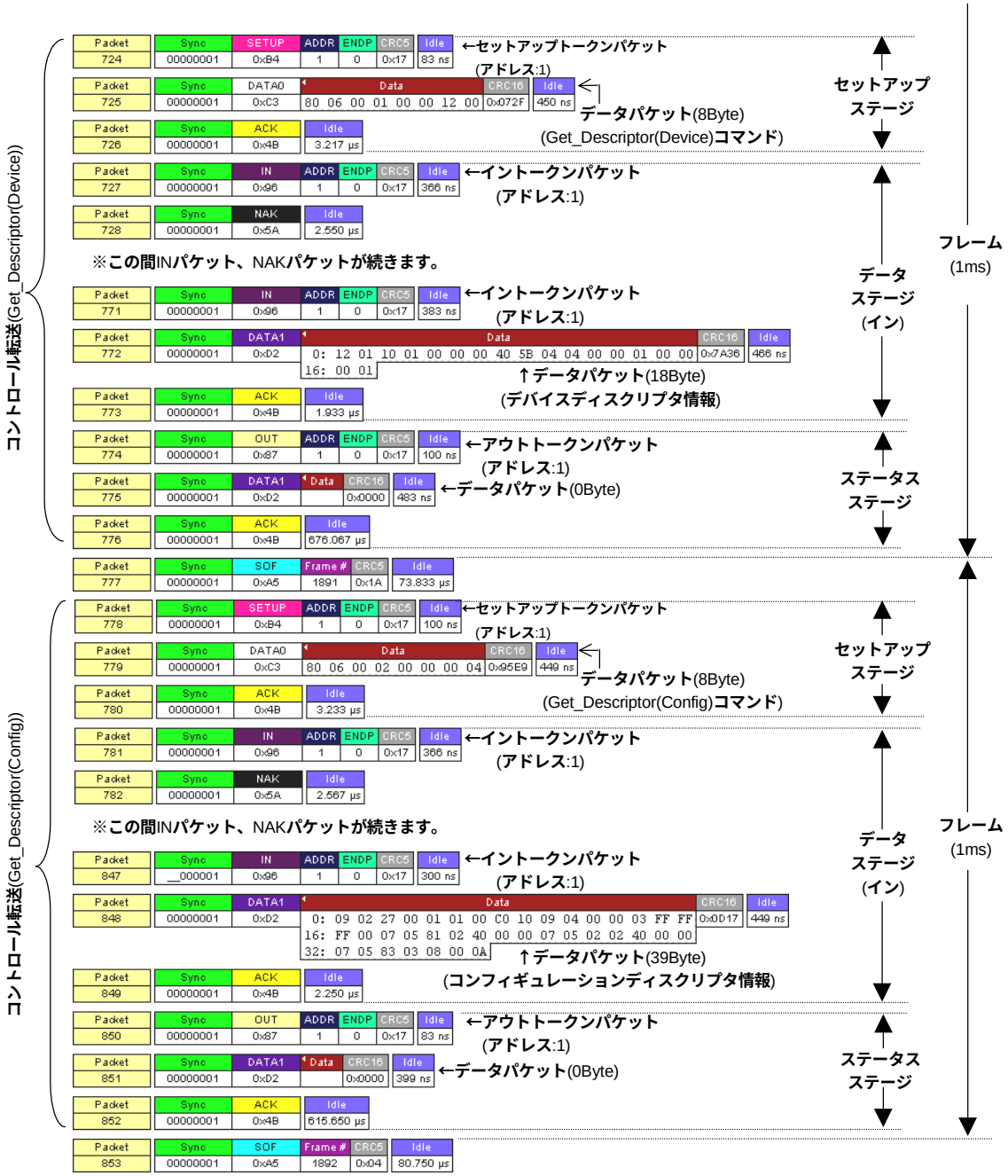
なお､ホストコントローラによりパケットのスケジューリングが本図と異なる場合がありますが構成ステートに至るまでのコマンドの流れは同一です。



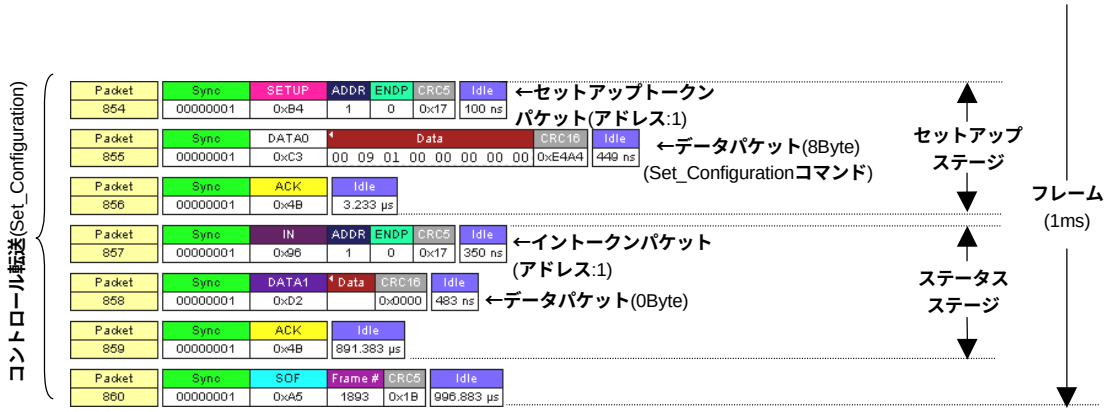

フレーム
(1ms)
Packet 92 | Sync 00000001 | SOF 0xA5 | Frame # 1528 | CRC5 0x16 | Idle 128.467 µs
Packet 93 | Reset 10.866 ms | Idle 999.467 µs
再びリセット信号が入ります。
Packet 94 | Sync 00000001 | SOF 0xA5 | Frame # 1540 | CRC5 0x1B | Idle 996.867 µs
※この間SOFパケットのみが続きます。
Packet 111 | Sync 00000001 | SOF 0xA5 | Frame # 1557 | CRC5 0x03 | Idle 527.883 µs
コントロール転送(Set_Address)
Packet 112 | Sync 00000001 | SETUP 0xB4 | ADDR 0 | ENDP 0 | CRC5 0x08 | Idle 83 ns
←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 113 | Sync 00000001 | DATA0 0xC3 | Data 00 05 01 00 00 00 00 00 | CRC16 0xD7A4 | Idle 400 ns
データパケット(8Byte)
(Set_Address(アドレス:1)コマンド)
Packet 114 | Sync 00000001 | ACK 0x4B | Idle 3.183 µs
Packet 115 | Sync 00000001 | IN 0x96 | ADDR 0 | ENDP 0 | CRC5 0x08 | Idle 333 ns
←イントークンパケット
(デフォルトアドレス使用)
Packet 116 | Sync 00000001 | DATA1 0xD2 | Data | CRC16 0x0000 | Idle 433 ns
←データパケット(0Byte)
Packet 117 | Sync 00000001 | ACK 0x4B | Idle 444.417 µs
←ACKハンドシェイクパケット
セットアップ
ステージ
ステータス
ステージ
フレーム
(1ms)
Packet 118 | Sync 00000001 | SOF 0xA5 | Frame # 1558 | CRC5 0x01 | Idle 996.867 µs
※この間SOFパケットのみが続きます。
※以降、アドレスステートに遷移します。
Packet 157 | Sync 00000001 | SOF 0xA5 | Frame # 1597 | CRC5 0x1C | Idle 611.783 µs
コントロール転送(Get_Descriptor(Device))
Packet 158 | Sync 00000001 | SETUP 0xB4 | ADDR 1 | ENDP 0 | CRC5 0x17 | Idle 83 ns
←セットアップトークンパケット
(アドレス:1)
Packet 159 | Sync 00000001 | DATA0 0xC3 | Data 80 06 00 01 00 00 12 00 | CRC16 0x072F | Idle 416 ns
データパケット(8Byte)
(Get_Descriptor(Device)コマンド)
Packet 160 | Sync 00000001 | ACK 0x4B | Idle 3.267 µs
Packet 161 | Sync 00000001 | IN 0x96 | ADDR 1 | ENDP 0 | CRC5 0x17 | Idle 333 ns
←イントークンパケット
(アドレス:1)
Packet 162 | Sync 00000001 | NAK 0x5A | Idle 2.600 µs
※この間INパケット、NAKパケットが続きます。
Packet 205 | Sync 00000001 | IN 0x96 | ADDR 1 | ENDP 0 | CRC5 0x17 | Idle 383 ns
←イントークンパケット
(アドレス:1)
Packet 206 | Sync 00000001 | DATA1 0xD2 | Data 0: 12 01 10 01 00 00 00 40 5B 04 04 00 00 01 00 00 16: 00 01 | CRC16 0x7A36 | Idle 466 ns
↑データパケット(18Byte)
(デバイスディスクリプタ情報)
Packet 207 | Sync 00000001 | ACK 0x4B | Idle 1.917 µs
Packet 208 | Sync 00000001 | OUT 0x87 | ADDR 1 | ENDP 0 | CRC5 0x17 | Idle 66 ns
←アウトトークンパケット
(アドレス:1)
Packet 209 | Sync 00000001 | DATA1 0xD2 | Data | CRC16 0x0000 | Idle 449 ns
←データパケット(0Byte)
Packet 210 | Sync 00000001 | ACK 0x4B | Idle 173.767 µs
データ
ステージ
(イン)
ステータス
ステージ
フレーム
(1ms)
Packet 211 | Sync 00000001 | SOF 0xA5 | Frame # 1598 | CRC5 0x1E | Idle 87.650 µs
※次頁に続きます。


コントロール転送(Get_Descriptor(Config))
Packet 212 | Sync 00000001 | SETUP 0xB4 | ADDR 1 | ENDP 0 | CRC5 0x17 | Idle 83 ns
←セットアップトークンパケット
(アドレス:1)
Packet 213 | Sync 00000001 | DATA0 0xC3 | Data 80 06 00 02 00 00 09 00 | CRC16 0x7520 | Idle 433 ns
←データパケット(8Byte)
(Get_Descriptor(Config)コマンド)
Packet 214 | Sync 00000001 | ACK 0x4B | Idle 3.500 µs
Packet 215 | Sync 00000001 | IN 0x96 | ADDR 1 | ENDP 0 | CRC5 0x17 | Idle 350 ns
←イントークンパケット
(アドレス:1)
Packet 216 | Sync 00000001 | NAK 0x5A | Idle 2.583 µs
※この間INパケット、NAKパケットが続きます。
セットアップ ステージ
データ ステージ (イン)
Packet 249 | Sync 00000001 | IN 0x96 | ADDR 1 | ENDP 0 | CRC5 0x17 | Idle 350 ns
←イントークンパケット
(アドレス:1)
Packet 250 | Sync 00000001 | DATA1 0xD2 | Data 09 02 27 00 01 01 00 C0 10 | CRC16 0x242E | Idle 500 ns
←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 251 | Sync 00000001 | ACK 0x4B | Idle 1.750 µs
Packet 252 | Sync 00000001 | OUT 0x87 | ADDR 1 | ENDP 0 | CRC5 0x17 | Idle 100 ns
←アウトトークンパケット
(アドレス:1)
Packet 253 | Sync 00000001 | DATA1 0xD2 | Data | CRC16 0x0000 | Idle 450 ns
←データパケット(0Byte)
Packet 254 | Sync 00000001 | ACK 0x4B | Idle 742.167 µs
ステータス ステージ
フレーム (1ms)
Packet 255 | Sync 00000001 | SOF 0xA5 | Frame # 1599 | CRC5 0x01 | Idle 191.300 µs
コントロール転送(Get_Descriptor(Config))
Packet 256 | Sync 00000001 | SETUP 0xB4 | ADDR 1 | ENDP 0 | CRC5 0x17 | Idle 83 ns
←セットアップトークンパケット
(アドレス:1)
Packet 257 | Sync 00000001 | DATA0 0xC3 | Data 80 06 00 02 00 00 FF 00 | CRC16 0x9725 | Idle 416 ns
←データパケット(8Byte)
(Get_Descriptor(Config)コマンド)
Packet 258 | Sync 00000001 | ACK 0x4B | Idle 3.667 µs
Packet 259 | Sync 00000001 | IN 0x96 | ADDR 1 | ENDP 0 | CRC5 0x17 | Idle 350 ns
←イントークンパケット
(アドレス:1)
Packet 260 | Sync 00000001 | NAK 0x5A | Idle 2.583 µs
※この間INパケット、NAKパケットが続きます。
セットアップ ステージ
データ ステージ (イン)
Packet 325 | Sync 00000001 | IN 0x96 | ADDR 1 | ENDP 0 | CRC5 0x17 | Idle 383 ns
←イントークンパケット
(アドレス:1)
Packet 326 | Sync 00000001 | DATA1 0xD2 | Data 0: 09 02 27 00 01 01 00 C0 10 09 04 00 00 03 FF FF 16: FF 00 07 05 81 02 40 00 00 07 05 02 02 40 00 00 32: 07 05 83 03 08 00 0A | CRC16 0x0D17 | Idle 466 ns
↑データパケット(39Byte)
(コンフィギュレーションディスクリプタ情報)
Packet 327 | Sync 00000001 | ACK 0x4B | Idle 2.017 µs
Packet 328 | Sync 00000001 | OUT 0x87 | ADDR 1 | ENDP 0 | CRC5 0x17 | Idle 100 ns
←アウトトークンパケット
(アドレス:1)
Packet 329 | Sync 00000001 | DATA1 0xD2 | Data | CRC16 0x0000 | Idle 400 ns
←データパケット(0Byte)
Packet 330 | Sync 00000001 | ACK 0x4B | Idle 497.183 µs
ステータス ステージ
フレーム (1ms)
Packet 331 | Sync 00000001 | SOF 0xA5 | Frame # 1600 | CRC5 0x1D | Idle 996.867 µs
※次頁に続きます。
※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(Config))
コントロール転送(Get_Descriptor(Device))
セットアップステージ
データステージ(イン)
ステータスステージ
フレーム(1ms)
←セットアップトークンパケット(アドレス:1)
データパケット(8Byte)(Get_Descriptor(Config)コマンド)
←イントークンパケット(アドレス:1)
※この間INパケット、NAKパケットが続きます。
←データパケット(0Byte)
↑データパケット(39Byte)(コンフィギュレーションディスクリプタ情報)
←アウトトークンパケット(アドレス:1)
データパケット(8Byte)(Get_Descriptor(Device)コマンド)
↑データパケット(18Byte)(デバイスディスクリプタ情報)
※この間SOFパケットのみが続きます。
※次頁に続きます。


コントロール転送(Get_Descriptor(Device))
Packet 724 Sync 00000001 SETUP 0xB4 ADDR 1 ENDP 0 CRC5 0x17 Idle 83 ns ←セットアップトークンパケット (アドレス:1)
Packet 725 Sync 00000001 DATA0 0xC3 Data 80 06 00 01 00 00 12 00 CRC16 0x072F Idle 450 ns ←データパケット(8Byte) (Get_Descriptor(Device)コマンド)
Packet 726 Sync 00000001 ACK 0x4B Idle 3.217 µs
セットアップステージ
Packet 727 Sync 00000001 IN 0x96 ADDR 1 ENDP 0 CRC5 0x17 Idle 366 ns ←イントークンパケット (アドレス:1)
Packet 728 Sync 00000001 NAK 0x5A Idle 2.550 µs
※この間INパケット、NAKパケットが続きます。
Packet 771 Sync 00000001 IN 0x96 ADDR 1 ENDP 0 CRC5 0x17 Idle 383 ns ←イントークンパケット (アドレス:1)
Packet 772 Sync 00000001 DATA1 0xD2 Data 0: 12 01 10 01 00 00 00 40 5B 04 04 00 00 01 00 00 16: 00 01 CRC16 0x7A36 Idle 466 ns ↑データパケット(18Byte) (デバイスディスクリプタ情報)
Packet 773 Sync 00000001 ACK 0x4B Idle 1.933 µs
データステージ(イン)
Packet 774 Sync 00000001 OUT 0x87 ADDR 1 ENDP 0 CRC5 0x17 Idle 100 ns ←アウトトークンパケット (アドレス:1)
Packet 775 Sync 00000001 DATA1 0xD2 Data CRC16 0x0000 Idle 483 ns ←データパケット(0Byte)
Packet 776 Sync 00000001 ACK 0x4B Idle 676.067 µs
ステータスステージ
フレーム(1ms)
Packet 777 Sync 00000001 SOF 0xA5 Frame # 1891 CRC5 0x1A Idle 73.833 µs
コントロール転送(Get_Descriptor(Config))
Packet 778 Sync 00000001 SETUP 0xB4 ADDR 1 ENDP 0 CRC5 0x17 Idle 100 ns ←セットアップトークンパケット (アドレス:1)
Packet 779 Sync 00000001 DATA0 0xC3 Data 80 06 00 02 00 00 00 04 CRC16 0x95E9 Idle 449 ns ←データパケット(8Byte) (Get_Descriptor(Config)コマンド)
Packet 780 Sync 00000001 ACK 0x4B Idle 3.233 µs
セットアップステージ
Packet 781 Sync 00000001 IN 0x96 ADDR 1 ENDP 0 CRC5 0x17 Idle 366 ns ←イントークンパケット (アドレス:1)
Packet 782 Sync 00000001 NAK 0x5A Idle 2.567 µs
※この間INパケット、NAKパケットが続きます。
Packet 847 Sync 00000001 IN 0x96 ADDR 1 ENDP 0 CRC5 0x17 Idle 300 ns ←イントークンパケット (アドレス:1)
Packet 848 Sync 00000001 DATA1 0xD2 Data 0: 09 02 27 00 01 01 00 C0 10 09 04 00 00 03 FF FF 16: FF 00 07 05 81 02 40 00 00 07 05 02 02 40 00 00 32: 07 05 83 03 08 00 0A CRC16 0x0D17 Idle 449 ns ↑データパケット(39Byte) (コンフィギュレーションディスクリプタ情報)
Packet 849 Sync 00000001 ACK 0x4B Idle 2.250 µs
データステージ(イン)
Packet 850 Sync 00000001 OUT 0x87 ADDR 1 ENDP 0 CRC5 0x17 Idle 83 ns ←アウトトークンパケット (アドレス:1)
Packet 851 Sync 00000001 DATA1 0xD2 Data CRC16 0x0000 Idle 399 ns ←データパケット(0Byte)
Packet 852 Sync 00000001 ACK 0x4B Idle 615.650 µs
ステータスステージ
フレーム(1ms)
Packet 853 Sync 00000001 SOF 0xA5 Frame # 1892 CRC5 0x04 Idle 80.750 µs

※次頁に続きます。



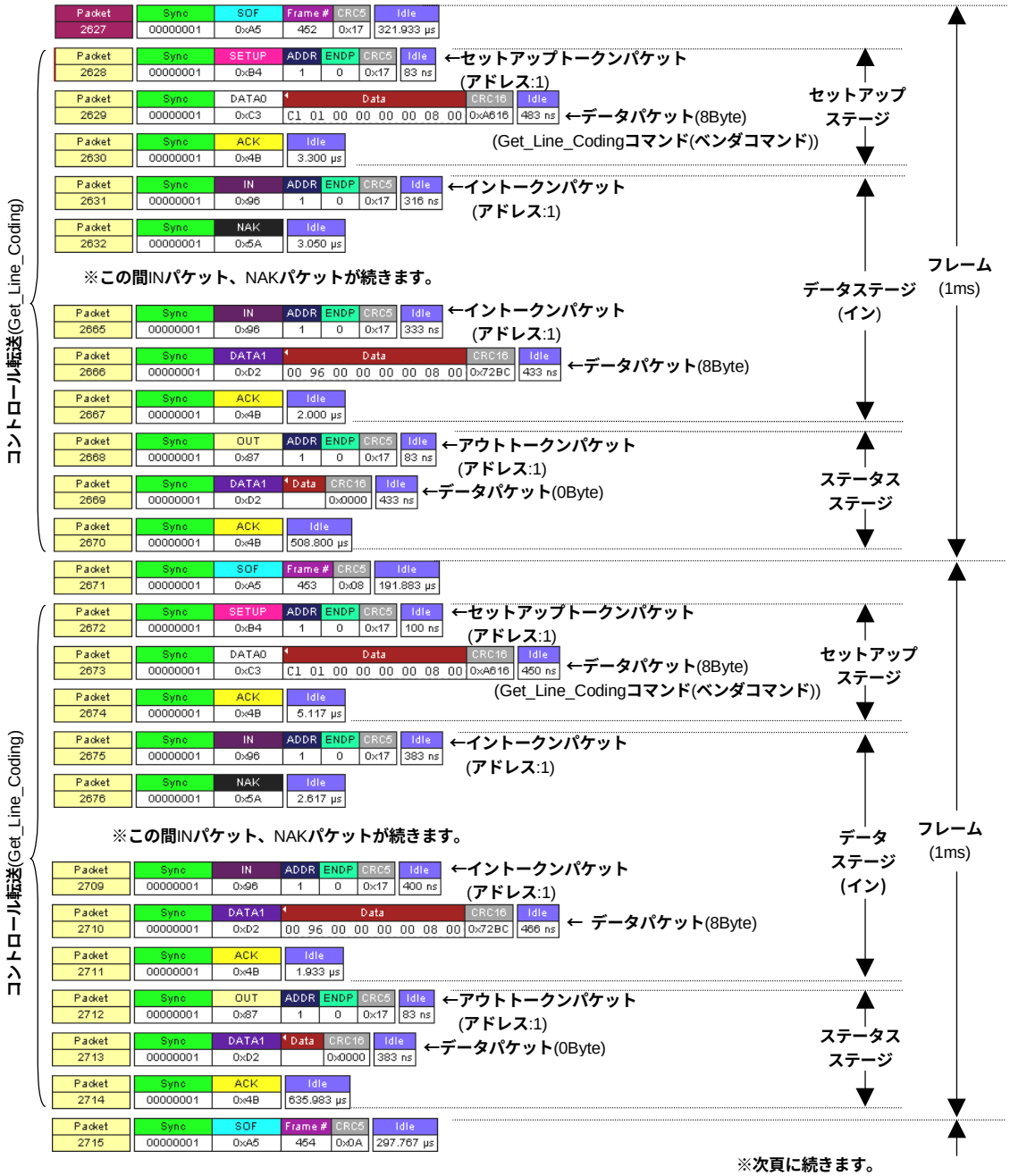
※以降、構成ステートに遷移します。

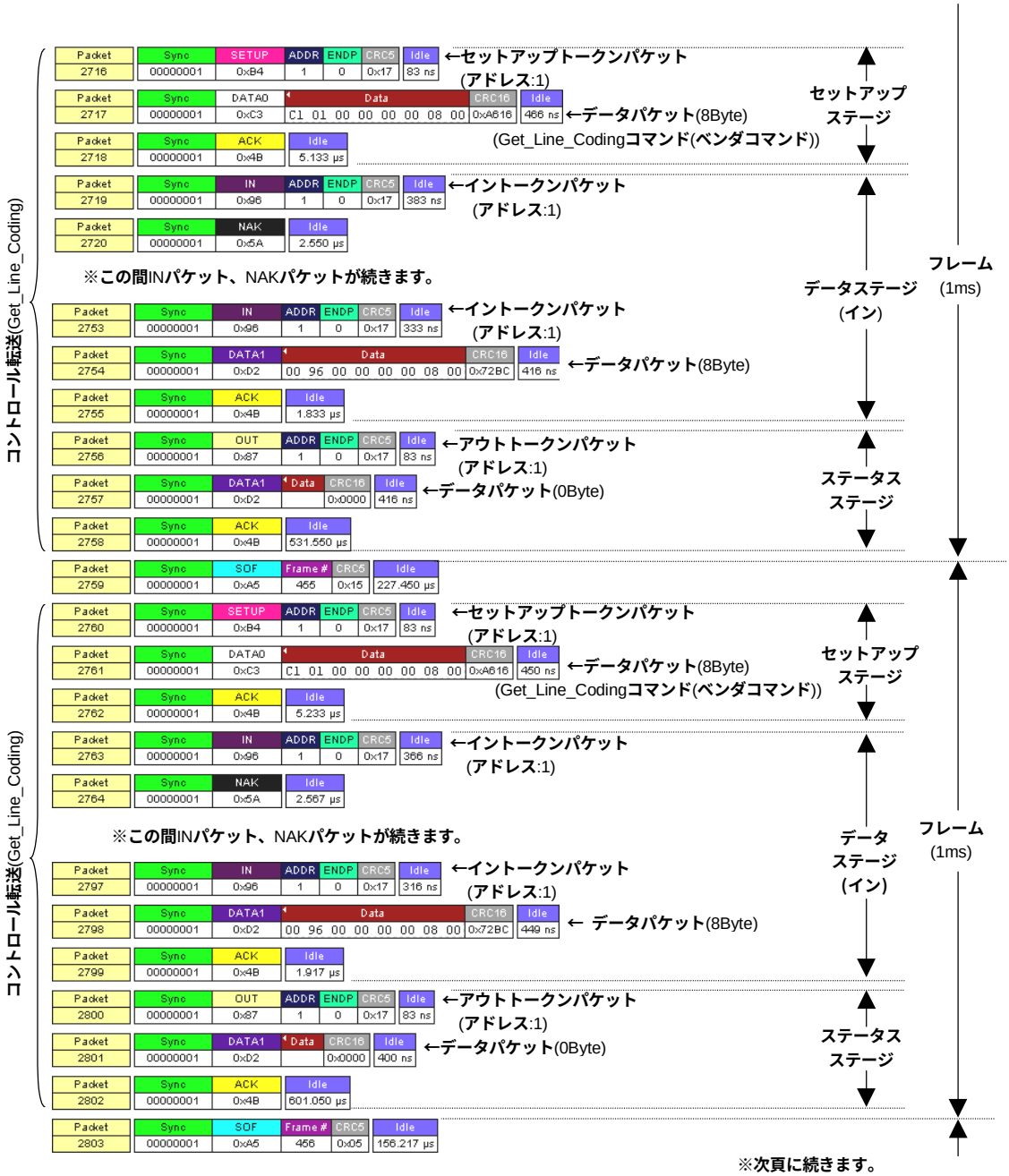
※以降、コントロール転送（ベンダーコマンド）があるまで定常状態となります。

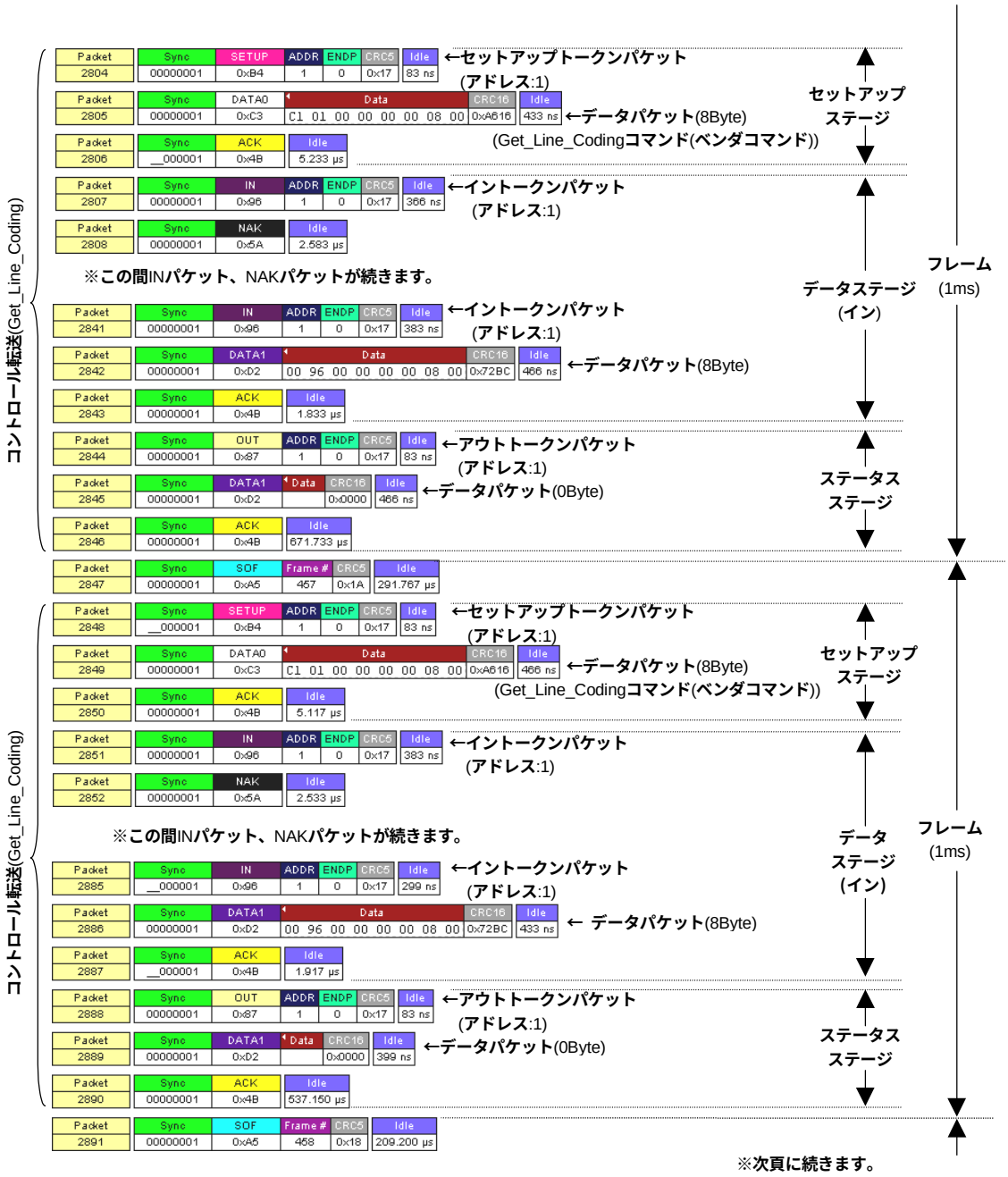
図 5.1　デバイス接続時のコントロール転送

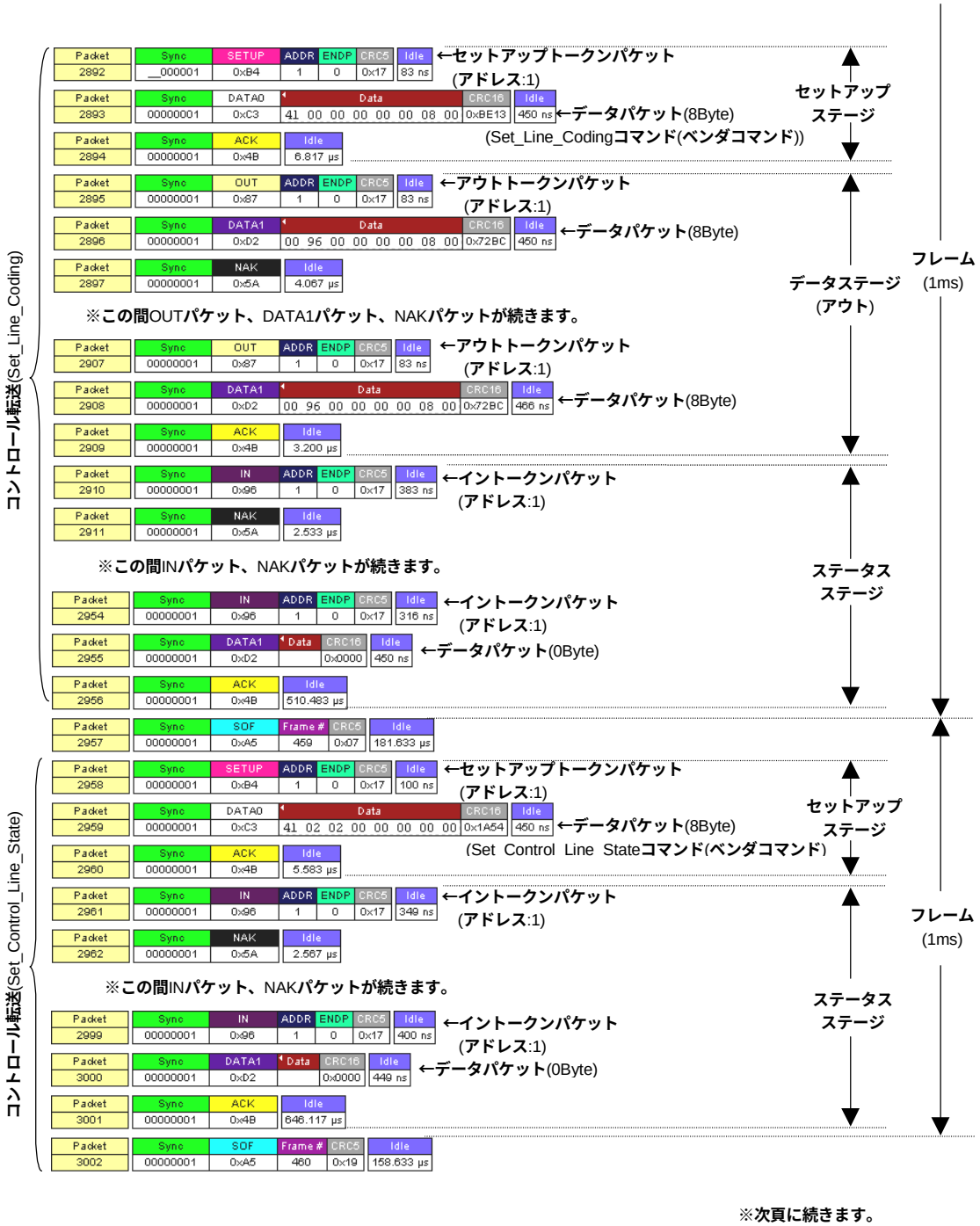
5.2 ベンダコマンド転送時のコントロール転送

図 5.2 は、ホストコントローラとデバイス間でベンダコマンドをコントロール転送で送信している際の測定結果です（ベンダコマンドについては**「4.8 ベンダコマンド」**をご参照ください）。

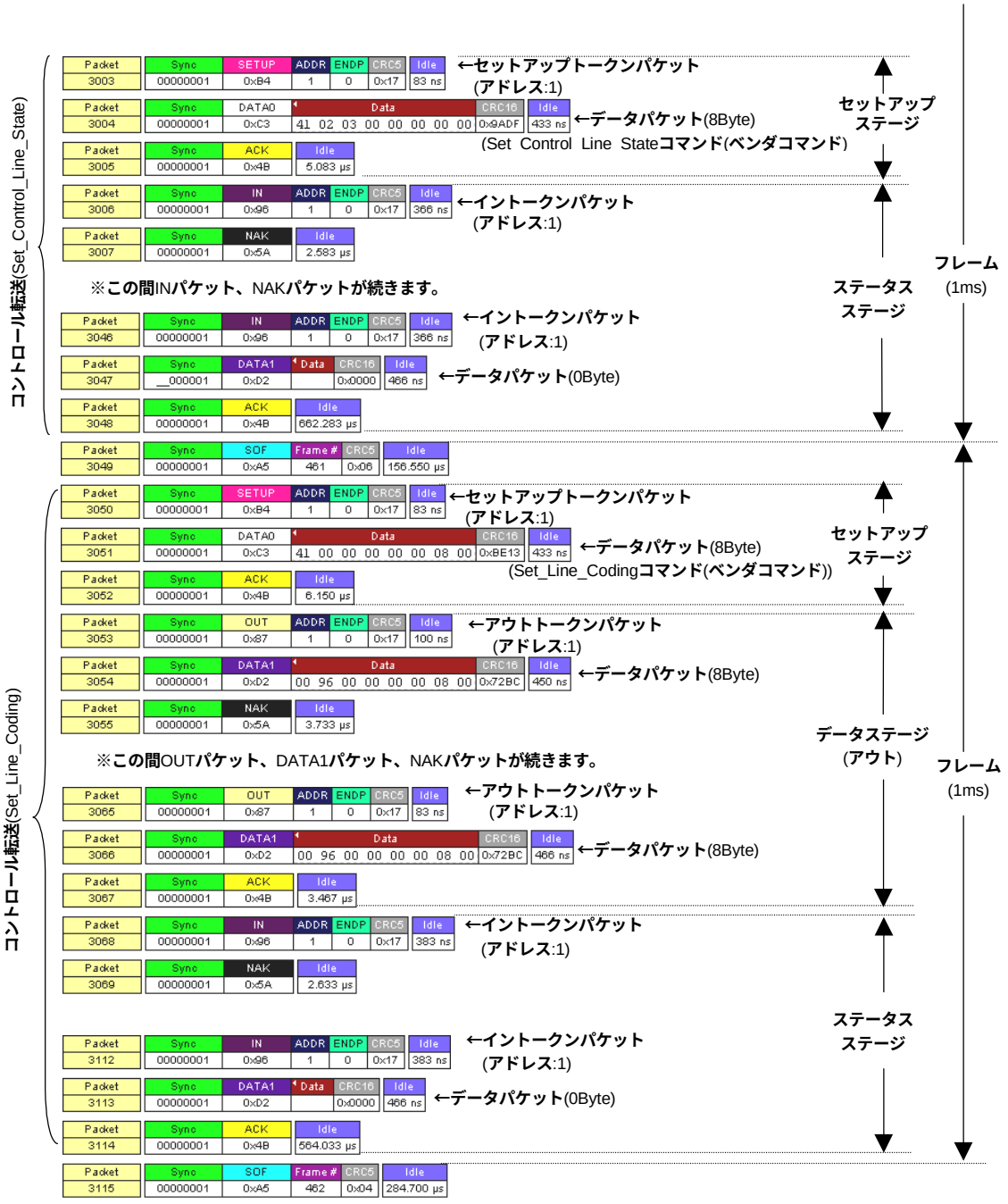



コントロール転送(Get_Line_Coding)
Packet 2716 Sync 00000001 SETUP 0xB4 ADDR 1 ENDP 0 CRC5 0x17 Idle 83 ns ←セットアップトークンパケット (アドレス:1)
Packet 2717 Sync 00000001 DATA0 0xC3 Data C1 01 00 00 00 00 08 00 CRC16 0xA616 Idle 466 ns ←データパケット(8Byte) (Get_Line_Codingコマンド(ベンダコマンド))
Packet 2718 Sync 00000001 ACK 0x4B Idle 5.133 µs
セットアップ ステージ
Packet 2719 Sync 00000001 IN 0x96 ADDR 1 ENDP 0 CRC5 0x17 Idle 383 ns ←イントークンパケット (アドレス:1)
Packet 2720 Sync 00000001 NAK 0x5A Idle 2.550 µs
※この間INパケット、NAKパケットが続きます。
Packet 2753 Sync 00000001 IN 0x96 ADDR 1 ENDP 0 CRC5 0x17 Idle 333 ns ←イントークンパケット (アドレス:1)
Packet 2754 Sync 00000001 DATA1 0xD2 Data 00 96 00 00 00 00 08 00 CRC16 0x72BC Idle 416 ns ←データパケット(8Byte)
Packet 2755 Sync 00000001 ACK 0x4B Idle 1.833 µs
データステージ (イン)
Packet 2756 Sync 00000001 OUT 0x87 ADDR 1 ENDP 0 CRC5 0x17 Idle 83 ns ←アウトトークンパケット (アドレス:1)
Packet 2757 Sync 00000001 DATA1 0xD2 Data CRC16 0x0000 Idle 416 ns ←データパケット(0Byte)
Packet 2758 Sync 00000001 ACK 0x4B Idle 531.550 µs
ステータス ステージ
フレーム (1ms)
Packet 2759 Sync 00000001 SOF 0xA5 Frame # 455 CRC5 0x15 Idle 227.450 µs
コントロール転送(Get_Line_Coding)
Packet 2760 Sync 00000001 SETUP 0xB4 ADDR 1 ENDP 0 CRC5 0x17 Idle 83 ns ←セットアップトークンパケット (アドレス:1)
Packet 2761 Sync 00000001 DATA0 0xC3 Data C1 01 00 00 00 00 08 00 CRC16 0xA616 Idle 450 ns ←データパケット(8Byte) (Get_Line_Codingコマンド(ベンダコマンド))
Packet 2762 Sync 00000001 ACK 0x4B Idle 5.233 µs
セットアップ ステージ
Packet 2763 Sync 00000001 IN 0x96 ADDR 1 ENDP 0 CRC5 0x17 366 ns ←イントークンパケット (アドレス:1)
Packet 2764 Sync 00000001 NAK 0x5A Idle 2.567 µs
※この間INパケット、NAKパケットが続きます。
Packet 2797 Sync 00000001 IN 0x96 ADDR 1 ENDP 0 CRC5 0x17 Idle 316 ns ←イントークンパケット (アドレス:1)
Packet 2798 Sync 00000001 DATA1 0xD2 Data 00 96 00 00 00 00 08 00 CRC16 0x72BC Idle 449 ns ← データパケット(8Byte)
Packet 2799 Sync 00000001 ACK 0x4B Idle 1.917 µs
データ ステージ (イン)
Packet 2800 Sync 00000001 OUT 0x87 ADDR 1 ENDP 0 CRC5 0x17 Idle 83 ns ←アウトトークンパケット (アドレス:1)
Packet 2801 Sync 00000001 DATA1 0xD2 Data CRC16 0x0000 Idle 400 ns ←データパケット(0Byte)
Packet 2802 Sync 00000001 ACK 0x4B Idle 601.050 µs
ステータス ステージ
フレーム (1ms)
Packet 2803 Sync 00000001 SOF 0xA5 Frame # 456 CRC5 0x05 Idle 156.217 µs
※次頁に続きます。


コントロール転送(Get_Line_Coding)
Packet 2804 Sync 00000001 SETUP 0xB4 ADDR 1 ENDP 0 CRC5 0x17 Idle 83 ns ←セットアップトークンパケット (アドレス:1)
Packet 2805 Sync 00000001 DATA0 0xC3 Data C1 01 00 00 00 00 08 00 CRC16 0xA616 Idle 433 ns ←データパケット(8Byte) (Get_Line_Codingコマンド(ベンダコマンド))
Packet 2806 Sync __000001 ACK 0x4B Idle 5.233 µs
セットアップ ステージ
Packet 2807 Sync 00000001 IN 0x96 ADDR 1 ENDP 0 CRC5 0x17 Idle 366 ns ←イントークンパケット (アドレス:1)
Packet 2808 Sync 00000001 NAK 0x5A Idle 2.583 µs
※この間INパケット、NAKパケットが続きます。
データステージ (イン)
Packet 2841 Sync 00000001 IN 0x96 ADDR 1 ENDP 0 CRC5 0x17 Idle 383 ns ←イントークンパケット (アドレス:1)
Packet 2842 Sync 00000001 DATA1 0xD2 Data 00 96 00 00 00 00 08 00 CRC16 0x72BC Idle 466 ns ←データパケット(8Byte)
Packet 2843 Sync 00000001 ACK 0x4B Idle 1.833 µs
Packet 2844 Sync 00000001 OUT 0x87 ADDR 1 ENDP 0 CRC5 0x17 Idle 83 ns ←アウトトークンパケット (アドレス:1)
Packet 2845 Sync 00000001 DATA1 0xD2 Data CRC16 0x0000 Idle 466 ns ←データパケット(0Byte)
Packet 2846 Sync 00000001 ACK 0x4B Idle 671.733 µs
ステータス ステージ
フレーム (1ms)
Packet 2847 Sync 00000001 SOF 0xA5 Frame # 457 CRC5 0x1A Idle 291.767 µs
コントロール転送(Get_Line_Coding)
Packet 2848 Sync __000001 SETUP 0xB4 ADDR 1 ENDP 0 CRC5 0x17 Idle 83 ns ←セットアップトークンパケット (アドレス:1)
Packet 2849 Sync 00000001 DATA0 0xC3 Data C1 01 00 00 00 00 08 00 CRC16 0xA616 Idle 466 ns ←データパケット(8Byte) (Get_Line_Codingコマンド(ベンダコマンド))
Packet 2850 Sync 00000001 ACK 0x4B Idle 5.117 µs
セットアップ ステージ
Packet 2851 Sync 00000001 IN 0x96 ADDR 1 ENDP 0 CRC5 0x17 Idle 383 ns ←イントークンパケット (アドレス:1)
Packet 2852 Sync 00000001 NAK 0x5A Idle 2.533 µs
※この間INパケット、NAKパケットが続きます。
データステージ (イン)
Packet 2885 Sync __000001 IN 0x96 ADDR 1 ENDP 0 CRC5 0x17 Idle 299 ns ←イントークンパケット (アドレス:1)
Packet 2886 Sync 00000001 DATA1 0xD2 Data 00 96 00 00 00 00 08 00 CRC16 0x72BC Idle 433 ns ← データパケット(8Byte)
Packet 2887 Sync __000001 ACK 0x4B Idle 1.917 µs
Packet 2888 Sync 00000001 OUT 0x87 ADDR 1 ENDP 0 CRC5 0x17 Idle 83 ns ←アウトトークンパケット (アドレス:1)
Packet 2889 Sync 00000001 DATA1 0xD2 Data CRC16 0x0000 Idle 399 ns ←データパケット(0Byte)
Packet 2890 Sync 00000001 ACK 0x4B Idle 537.150 µs
ステータス ステージ
フレーム (1ms)
Packet 2891 Sync 00000001 SOF 0xA5 Frame # 458 CRC5 0x18 Idle 209.200 µs
※次頁に続きます。


コントロール転送(Set_Line_Coding)
Packet 2892 Sync __000001 SETUP 0xB4 ADDR 1 ENDP 0 CRC5 0x17 Idle 83 ns ←セットアップトークンパケット (アドレス:1)
Packet 2893 Sync 00000001 DATA0 0xC3 Data 41 00 00 00 00 00 08 00 CRC16 0xBE13 Idle 450 ns ←データパケット(8Byte) (Set_Line_Codingコマンド(ベンダコマンド))
Packet 2894 Sync 00000001 ACK 0x4B Idle 6.817 µs
セットアップステージ
Packet 2895 Sync 00000001 OUT 0x87 ADDR 1 ENDP 0 CRC5 0x17 Idle 83 ns ←アウトトークンパケット (アドレス:1)
Packet 2896 Sync 00000001 DATA1 0xD2 Data 00 96 00 00 00 00 08 00 CRC16 0x72BC Idle 450 ns ←データパケット(8Byte)
Packet 2897 Sync 00000001 NAK 0x5A Idle 4.067 µs
※この間OUTパケット、DATA1パケット、NAKパケットが続きます。
Packet 2907 Sync 00000001 OUT 0x87 ADDR 1 ENDP 0 CRC5 0x17 Idle 83 ns ←アウトトークンパケット (アドレス:1)
Packet 2908 Sync 00000001 DATA1 0xD2 Data 00 96 00 00 00 00 08 00 CRC16 0x72BC Idle 466 ns ←データパケット(8Byte)
Packet 2909 Sync 00000001 ACK 0x4B Idle 3.200 µs
データステージ (アウト)
Packet 2910 Sync 00000001 IN 0x96 ADDR 1 ENDP 0 CRC5 0x17 Idle 383 ns ←イントークンパケット (アドレス:1)
Packet 2911 Sync 00000001 NAK 0x5A Idle 2.533 µs
※この間INパケット、NAKパケットが続きます。
Packet 2954 Sync 00000001 IN 0x96 ADDR 1 ENDP 0 CRC5 0x17 Idle 316 ns ←イントークンパケット (アドレス:1)
Packet 2955 Sync 00000001 DATA1 0xD2 Data CRC16 0x0000 Idle 450 ns ←データパケット(0Byte)
Packet 2956 Sync 00000001 ACK 0x4B Idle 510.483 µs
ステータスステージ
フレーム (1ms)
Packet 2957 Sync 00000001 SOF 0xA5 Frame # 459 CRC5 0x07 Idle 181.633 µs
コントロール転送(Set_Control_Line_State)
Packet 2958 Sync 00000001 SETUP 0xB4 ADDR 1 ENDP 0 CRC5 0x17 Idle 100 ns ←セットアップトークンパケット (アドレス:1)
Packet 2959 Sync 00000001 DATA0 0xC3 Data 41 02 02 00 00 00 00 00 CRC16 0x1A54 Idle 450 ns ←データパケット(8Byte) (Set Control Line Stateコマンド(ベンダコマンド)
Packet 2960 Sync 00000001 ACK 0x4B Idle 5.583 µs
セットアップステージ
Packet 2961 Sync 00000001 IN 0x96 ADDR 1 ENDP 0 CRC5 0x17 Idle 349 ns ←イントークンパケット (アドレス:1)
Packet 2962 Sync 00000001 NAK 0x5A Idle 2.567 µs
※この間INパケット、NAKパケットが続きます。
Packet 2999 Sync 00000001 IN 0x96 ADDR 1 ENDP 0 CRC5 0x17 Idle 400 ns ←イントークンパケット (アドレス:1)
Packet 3000 Sync 00000001 DATA1 0xD2 Data CRC16 0x0000 Idle 449 ns ←データパケット(0Byte)
Packet 3001 Sync 00000001 ACK 0x4B Idle 646.117 µs
ステータスステージ
フレーム (1ms)
Packet 3002 Sync 00000001 SOF 0xA5 Frame # 460 CRC5 0x19 Idle 158.633 µs

※次頁に続きます。


コントロール転送(Set_Control_Line_State)
Packet 3003 Sync 00000001 SETUP 0xB4 ADDR 1 ENDP 0 CRC5 0x17 Idle 83 ns ←セットアップトークンパケット (アドレス:1)
Packet 3004 Sync 00000001 DATA0 0xC3 Data 41 02 03 00 00 00 00 00 CRC16 0x9ADF Idle 433 ns ←データパケット(8Byte) (Set Control Line Stateコマンド(ベンダコマンド)
Packet 3005 Sync 00000001 ACK 0x4B Idle 5.083 µs
Packet 3006 Sync 00000001 IN 0x96 ADDR 1 ENDP 0 CRC5 0x17 Idle 366 ns ←イントークンパケット (アドレス:1)
Packet 3007 Sync 00000001 NAK 0x5A Idle 2.583 µs
※この間INパケット、NAKパケットが続きます。
Packet 3046 Sync 00000001 IN 0x96 ADDR 1 ENDP 0 CRC5 0x17 Idle 366 ns ←イントークンパケット (アドレス:1)
Packet 3047 Sync 00000001 DATA1 0xD2 Data CRC16 0x0000 Idle 466 ns ←データパケット(0Byte)
Packet 3048 Sync 00000001 ACK 0x4B Idle 662.283 µs
Packet 3049 Sync 00000001 SOF 0xA5 Frame # 461 CRC5 0x06 Idle 156.550 µs
Packet 3050 Sync 00000001 SETUP 0xB4 ADDR 1 ENDP 0 CRC5 0x17 Idle 83 ns ←セットアップトークンパケット (アドレス:1)
Packet 3051 Sync 00000001 DATA0 0xC3 Data 41 00 00 00 00 00 08 00 CRC16 0xBE13 Idle 433 ns ←データパケット(8Byte) (Set_Line_Codingコマンド(ベンダコマンド))
Packet 3052 Sync 00000001 ACK 0x4B Idle 6.150 µs
Packet 3053 Sync 00000001 OUT 0x87 ADDR 1 ENDP 0 CRC5 0x17 Idle 100 ns ←アウトトークンパケット (アドレス:1)
Packet 3054 Sync 00000001 DATA1 0xD2 Data 00 96 00 00 00 00 08 00 CRC16 0x72BC Idle 450 ns ←データパケット(8Byte)
Packet 3055 Sync 00000001 NAK 0x5A Idle 3.733 µs
※この間OUTパケット、DATA1パケット、NAKパケットが続きます。
コントロール転送(Set_Line_Coding)
Packet 3065 Sync 00000001 OUT 0x87 ADDR 1 ENDP 0 CRC5 0x17 Idle 83 ns ←アウトトークンパケット (アドレス:1)
Packet 3066 Sync 00000001 DATA1 0xD2 Data 00 96 00 00 00 00 08 00 CRC16 0x72BC Idle 466 ns ←データパケット(8Byte)
Packet 3067 Sync 00000001 ACK 0x4B Idle 3.467 µs
Packet 3068 Sync 00000001 IN 0x96 ADDR 1 ENDP 0 CRC5 0x17 Idle 383 ns ←イントークンパケット (アドレス:1)
Packet 3069 Sync 00000001 NAK 0x5A Idle 2.633 µs
Packet 3112 Sync 00000001 IN 0x96 ADDR 1 ENDP 0 CRC5 0x17 Idle 383 ns ←イントークンパケット (アドレス:1)
Packet 3113 Sync 00000001 DATA1 0xD2 Data CRC16 0x0000 Idle 466 ns ←データパケット(0Byte)
Packet 3114 Sync 00000001 ACK 0x4B Idle 564.033 µs
Packet 3115 Sync 00000001 SOF 0xA5 Frame # 462 CRC5 0x04 Idle 284.700 µs
セットアップステージ
ステータスステージ
データステージ(アウト)
フレーム(1ms)

※次頁に続きます。

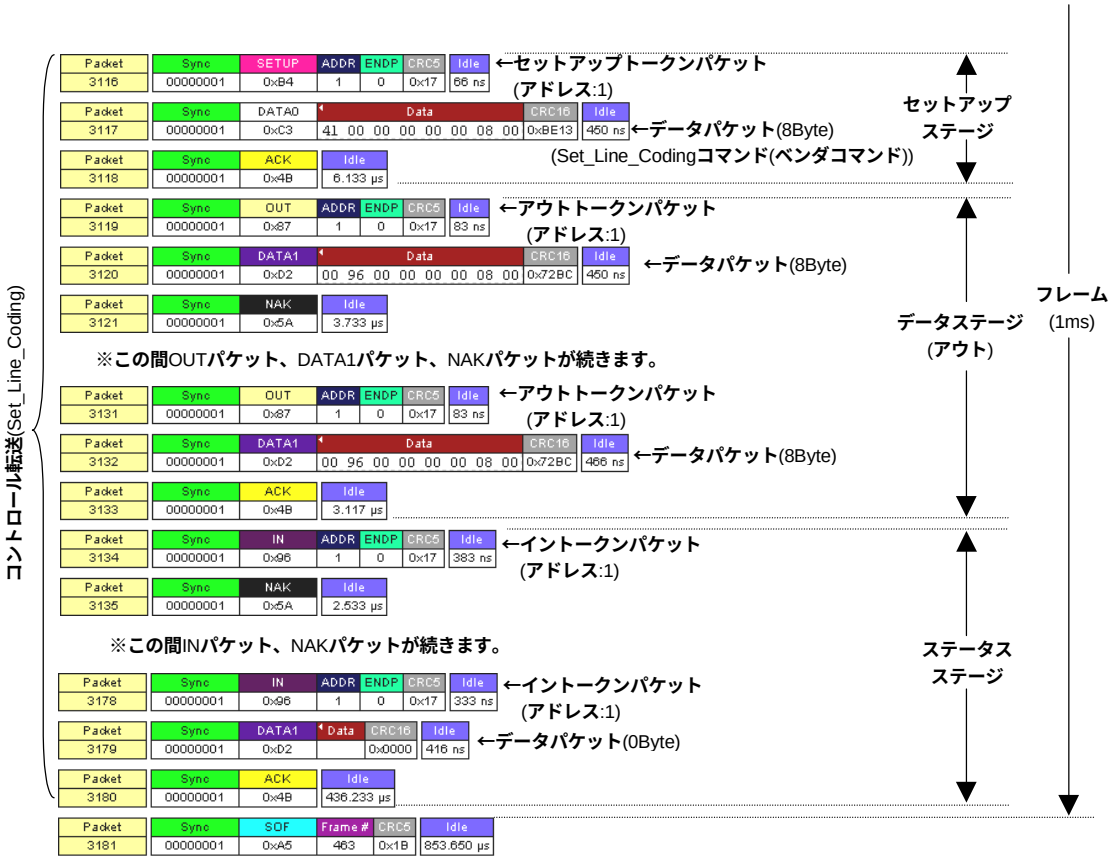


図 5.2　ベンダコマンド送信時のコントロール転送

H8S/2218 USBファンクションモジュール
USBシリアル変換アプリケーションノート

発行年月　2003年10月20日　Rev.1.00
発　　行　株式会社ルネサス テクノロジ 営業企画統括部
　　〒100-0004 東京都千代田区大手町 2-6-2
編　　集　株式会社ルネサス小平セミコン 技術ドキュメント部

株式会社ルネサス テクノロジ　営業企画統括部　〒100-0004 東京都千代田区大手町2-6-2 日本ビル

RENESAS

営業お問合せ窓口
株式会社ルネサス販売

http://www.renesas.com

本社	〒100-0004	千代田区大手町2-6-2 (日本ビル)	(03) 5201-5350
京浜支社	〒212-0058	川崎市幸区鹿島田890-12 (新川崎三井ビル)	(044) 549-1662
西東京支社	〒190-0023	立川市柴崎町2-2-23 (第二高島ビル2F)	(042) 524-8701
札幌支店	〒060-0002	札幌市中央区北二条西4-1 (札幌三井ビル5F)	(011) 210-8717
東北支社	〒980-0013	仙台市青葉区花京院1-1-20 (花京院スクエア13F)	(022) 221-1351
いわき支店	〒970-8026	いわき市平小太郎町4-9 (損保ジャパンいわき第二ビル3F)	(0246) 22-3222
茨城支社	〒312-0034	ひたちなか市堀口832-2 (日立システムプラザ勝田1F)	(029) 271-9411
新潟支店	〒950-0087	新潟市東大通1-4-2 (新潟三井物産ビル3F)	(025) 241-4361
松本支社	〒390-0815	松本市深志1-2-11 (昭和ビル7F)	(0263) 33-6622
中部営業本部	〒460-0008	名古屋市中区栄3-13-20 (栄センタービル4F)	(052) 261-3000
浜松支店	〒430-7710	浜松市板屋町111-2 (浜松アクトタワー10F)	(053) 451-2131
西部営業本部	〒541-0044	大阪市中央区伏見町4-1-1 (大阪明治生命館ランドアクシスタワー10F)	(06) 6233-9500
北陸支社	〒920-0031	金沢市広岡3-1-1 (金沢パークビル8F)	(076) 233-5980
中国支社	〒730-0036	広島市中区袋町5-25 (広島袋町ビルディング8F)	(082) 244-2570
松山支店	〒790-0003	松山市三番町4-4-6 (GEエジソンビル松山2号館3F)	(089) 933-9595
鳥取支店	〒680-0822	鳥取市今町2-251 (日本生命鳥取駅前ビル)	(0857) 21-1915
九州支社	〒812-0011	福岡市博多区博多駅前2-17-1 (ヒロカネビル本館5F)	(092) 481-7695
鹿児島支店	〒890-0053	鹿児島市中央町12-2 (明治生命西鹿児島ビル2F)	(099) 284-1748

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：カスタマサポートセンタ　E-Mail: csc@renesas.com

Colophon 1.0

H8S/2218　USB ファンクションモジュール　USB シリアル変換
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

RJJ06B0216-0100Z