M16C/6C グループ

USB シリアル変換

R01AN0751JJ0103
Rev.1.03
2011.07.29

1. 概要

このアプリケーションノートは、サンプルプログラムとM16C/6C内蔵USBファンクションモジュールを用い、USBシリアル変換システムの実現例を示します。

USB シリアル変換システムでは、USB シリアル変換機能により、USB インタフェースがシリアルインタフェースかのように動作します。通常、シリアル機器からUSB機器(レガシーフリー PC(注1)等)に置き換えると、インタフェースがシリアルインタフェースからUSBインタフェースへ変わるため、アプリケーションプログラムをも変更する必要があります。しかし、このシステムを使用することで、アプリケーションプログラムに変更を加えることなく接続することが可能になります。

注1. レガシーフリー PCとは、旧規格のポートを搭載せず、USB(Universal Serial Bus)等のPlug&Playに対応した新しい規格のインタフェースのみを搭載したPCのことです。

- 適用マイコン: M16C/6C グループマイコン

M16C/6Cグループと同様のSFR(周辺機能制御レジスタ)をもつ他のM16Cファミリのマイコンにも適用することが可能です。ただし、一部の機能を機能追加などで変更している場合がありますので、それぞれのマニュアルで確認してください。また、このアプリケーションノートのご使用に際して、十分な評価を行うようにしてください。

- 関連マニュアル

Universal Serial Bus Specification Revision 2.0
Universal Serial Bus Class Definitions for Communication Devices Version 1.1
M16C/6C グループユーザーズマニュアル ハードウェア編
E8aエミュレータユーザーズマニュアル

このアプリケーションに記載してい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ェアは準備しておりません。インタラプトの転送タイプをご使用になる場合は、別途プログラムを作成してください(M16C/6C グループユーザーズマニュアル ハードウェア編21-1章参照)。

1.1 USB ファンクションモジュール

M16C/6C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。
表1.1にエンドポイントの構成を示します。

- エンドポイント0に対するUSB標準コマンド(注1)の自動実行
- フルスピード(12Mbps)転送対応
- 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
- バストランシーバ内蔵
- バスパワーモードまたはセルフパワーモードをUSBコントロールレジスタ(USBCTLR)で選択可能
- Set_Configuration割り込みで現在のConfiguration値がチェック可能

注1. 一部コマンドはファームウェアで処理する必要があります。

表 1.1 エンドポイントの構成

エンドポイント	シンボル	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	セットアップ転送	8バイト	8バイト	×
	EP0I	コントロールIN	16バイト	16バイト	×
	EP0O	コントロールOUT	16バイト	16バイト	×
エンドポイント1	EP1	バルクOUT	64バイト	64×2 (128バイト)	○
エンドポイント2	EP2	バルクIN	64バイト	64×2 (128バイト)	○
エンドポイント3	EP3	インタラプトIN	16バイト	16バイト	×
エンドポイント4	EP4	バルクOUT	64バイト	64×2 (128バイト)	○
エンドポイント5	EP5	バルクIN	64バイト	64×2 (128バイト)	○
エンドポイント6	EP6	インタラプトIN	16バイト	16バイト	×

2. システム構成

この章では、USBシリアル変換システムの構成について説明します。

2.1 システム概要

USBシリアル変換のシステムの構成は以下の通りです。

- M16C CPU Board (Renesas Starter Kit for M16C/6C)
- シリアル接続PC
- USBホストPC (Windows 2000、Windows XP、Windows Vista)

図 2.1にシステム構成例を示します。

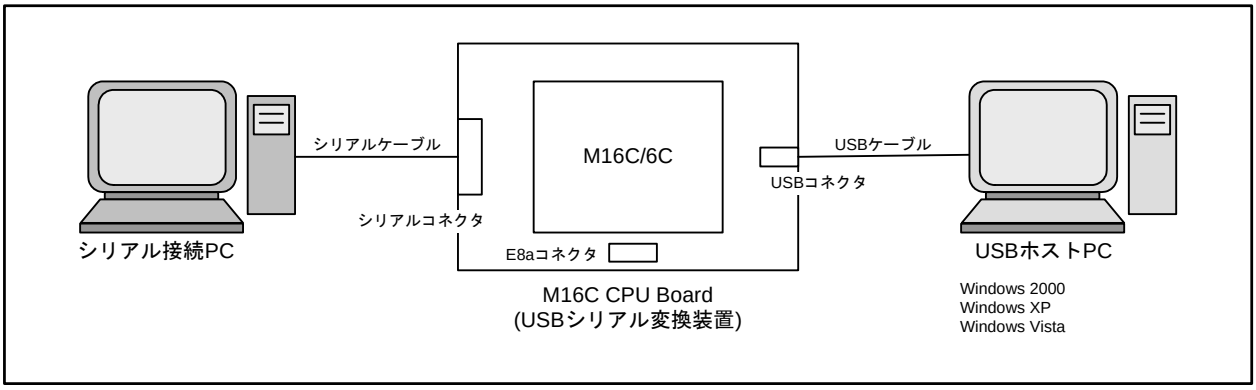


図 2.1 システム構成

システムの特長を以下に示します。

- サンプルプログラムによりM16C/6CのUSBモジュールを短期間で評価可能
- コントロール転送、バルク転送サポート
- E8a Emulator(ルネサスエレクトロニクス製)に対応しており効果的なデバッグが可能
- 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(注1)に対応可能

注1. 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。

USBホストPCとシリアル接続PCは、M16C CPU Boardを介して以下の手順で通信します。

USBホストPCからシリアル接続PCへデータ送信時：

(1) USBホストPCからUSBパケットにてM16C CPU Boardへデータを送信します
(2) M16C CPU Boardは受信データをシリアルデータに変換します
(3) M16C CPU Boardは変換したデータをシリアル接続PCへ送信します

シリアル接続PCからUSBホストPCへデータ送信時：

(1) シリアル接続PCからシリアルデータをM16C CPU Boardへ送信します
(2) M16C CPU Boardは受信データをUSBパケットに変換します
(3) M16C CPU Boardは変換したデータをUSBホストPCへ送信します

2.1.1 ハードウェア/ソフトウェア構成

2.1.2 シリアル接続PCとシリアル機器の接続

図 2.2 に、シリアル接続 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。

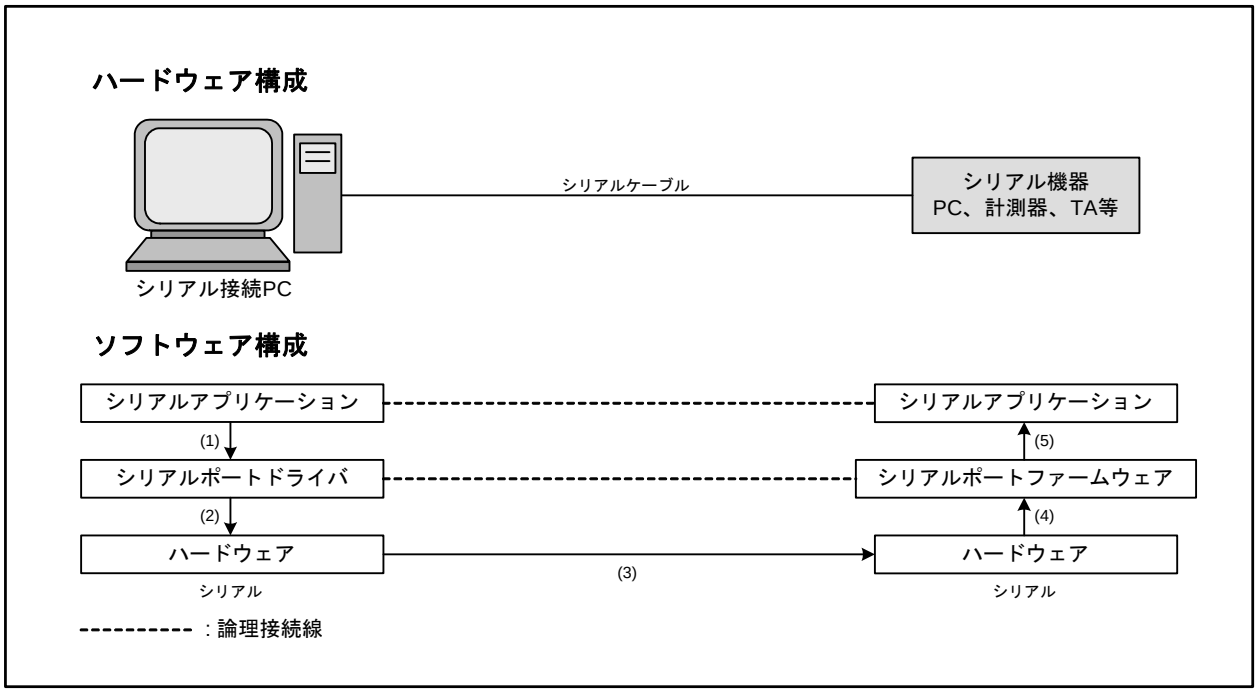


図 2.2 シリアル接続PCとシリアル機器接続時のハードウェア/ソフトウェア構成

シリアル接続PCとシリアル機器のデータ通信の流れを説明します。

シリアル接続PCからシリアル機器へのデータ送信手順:

(1) シリアル接続PCのシリアルアプリケーションは、シリアルポートドライバにデータを転送します。
(2) シリアルポートドライバはシリアル接続PCのハードウェアにデータを転送します。
(3) シリアル接続 PC のハードウェアは、シリアルケーブルを介してシリアル機器のハードウェアにデータを転送します。
(4) シリアル機器のハードウェアがデータを受信すると、シリアルポートファームウェアは受信データを読み出します。
(5) シリアルポートファームウェアは、読み出したデータをシリアルアプリケーションに転送します。

シリアル機器からシリアル接続PCへのデータ送信手順は、上記の逆となります。

2.1.3 USBホストPCとシリアル機器の接続

図 2.3に、USBホストPCとシリアル機器をUSBシリアル変換装置を介して接続した場合のハードウェア構成及びソフトウェア構成を示します。

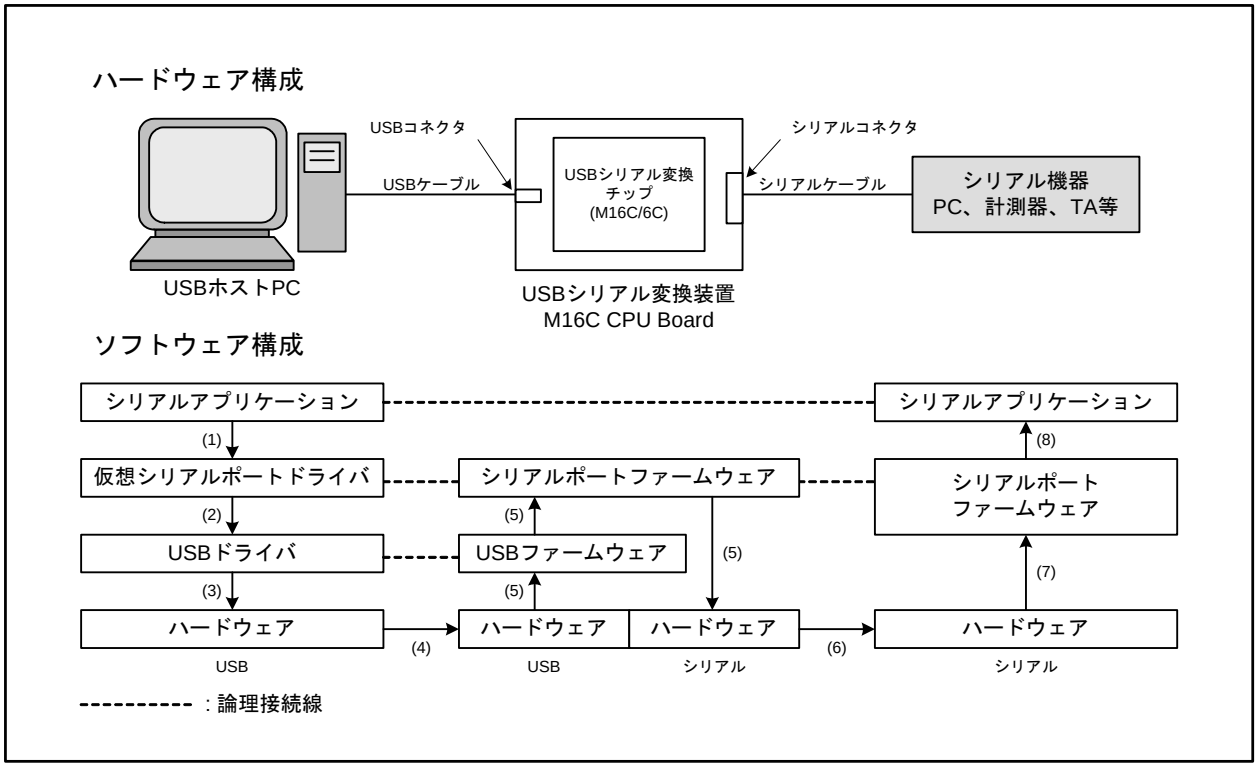


図 2.3 USBホストPCとシリアル機器接続時のハードウェア/ソフトウェア構成

USBシリアル変換装置を用いた、USBホストPCとシリアル機器のデータ通信の流れを説明します。

USBホストPCからシリアル機器へのデータ送信手順:

(1) USBホストPCのシリアルアプリケーションは、仮想シリアルポートドライバにデータを転送します。この仮想シリアルポートドライバはシリアルポートドライバと同じAPIを備えています。そのため、シリアルアプリケーションは、シリアル接続PCとシリアル機器接続時と同じシリアルアプリケーションを使用できます。
(2) 仮想シリアルポートドライバは、USB ドライバにデータを転送します。
(3) USB ドライバは、受信したデータをUSBホストPCのハードウェアに転送します。
(4) USBホストPCのハードウェアは、USBケーブルを介してUSBシリアル変換装置のUSBハードウェアにデータを転送します。
(5) 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ます。
(6) USB シリアル変換装置は、シリアルケーブルを介してシリアル機器のハードウェアに、シリアルデータを転送します。
(7) シリアル機器のハードウェアがデータを受信すると、シリアルポートファームウェアは受信データを読み出します。
(8) シリアルポートファームウェアは、読み出したデータをシリアルアプリケーションに転送します。

シリアル機器からUSBホストPCへのデータ送信手順は、上記の逆となります。

2.2 使用デバイス

- M16C CPU Board (Renesas Starter Kit for M16C/6C (注1))
- E8a Emulator(ルネサスエレクトロニクス製)
- E8a付属の接続ケーブル(ルネサスエレクトロニクス製)
- E8a用PC (Windows 2000、Windows Xp、Windows Vista)
- USBホスト用PC
- シリアル接続PC
- USBケーブル
- シリアルケーブル
- High-performance Embedded Workshop4 (HEW4)(ルネサスエレクトロニクス製)

注1. 本プログラムを使用する際は、Renesas Starter Kit for M16C/6C ボード上のシリアルコネクタ、7Pin と 8Pin を結線してください。

2.3 ハードウェア環境

図 2.4 に各デバイスの接続形態を示します。

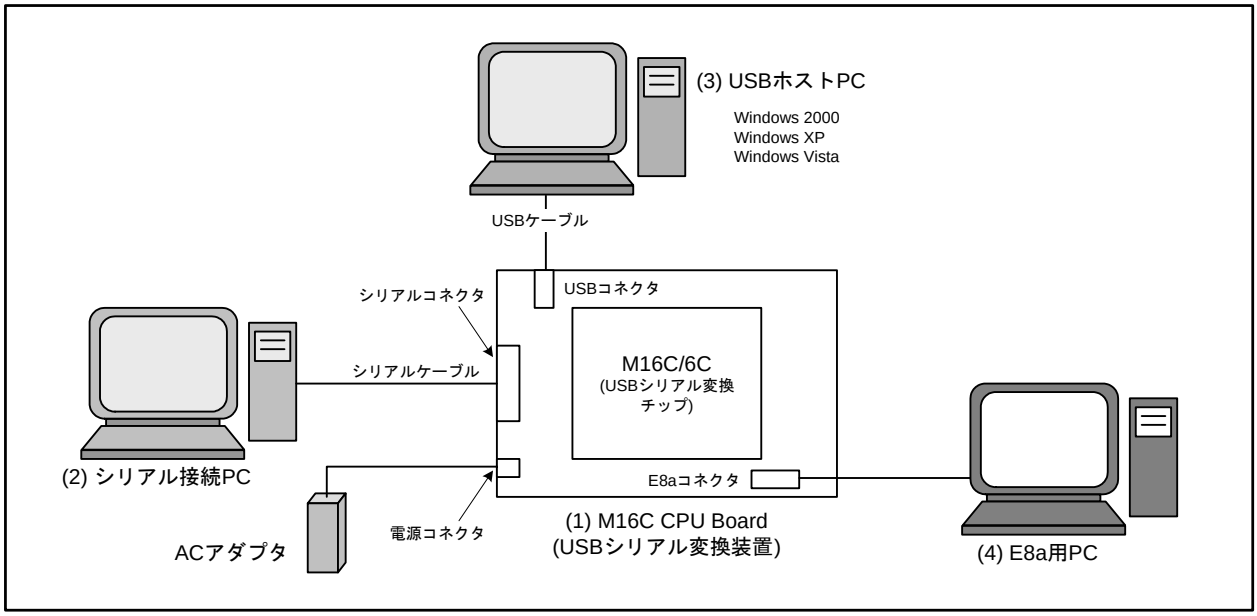


図 2.4 デバイス接続形態

(1) M16C CPU Board

表 2.1 に M16C CPU Board の各ポートに割り当てる機能を示します。

表 2.1 ポート割当て

端子名	信号名	入出力	機能
P1_0~P1_7	P1_0~P1_7	出力	通信状態出力
P6_3	TXD0	出力	シリアルデータ出力
P6_2	RXD0	入力	シリアルデータ入力

(2) シリアル接続 PC

シリアルポート搭載の PC を、シリアル接続 PC として使用してください。

(3) USB ホスト PC

USB ポートを搭載した、Windows 2000、Windows XP、Windows Vista のいずれかをインストールした PC を、USB ホスト PC として使用してください。OS に標準で搭載されている USB Communication Devices Class のデバイスドライバを使用します。新たにドライバをインストールする必要はありません。

USB 接続時に割り当てられる COM ポート番号は、PC 環境によって異なります。「システムのプロパティ」、「デバイスマネージャ」で、割り当てられた COM ポート番号を確認してください。

(4) E8a 用 PC

E8a 用 PC の USB コネクタに E8a エミュレータを接続し、接続用のケーブルを介して M16C CPU Board と接続してください。接続後、HEW4 を起動してエミュレーションを行います。

2.4 ソフトウェア環境

サンプルプログラムと、コンパイラ、リンカ、USB シリアル変換ドライバについて説明します。

2.4.1 サンプルプログラム

サンプルプログラムおよび“COM.hws”というワークスペースファイルは、COM フォルダに格納されています。サンプルプログラムを使用するには、このフォルダを HEW4 がインストールされた PC にコピーしてください。

図 2.5 に COM フォルダ内のファイルを示します。

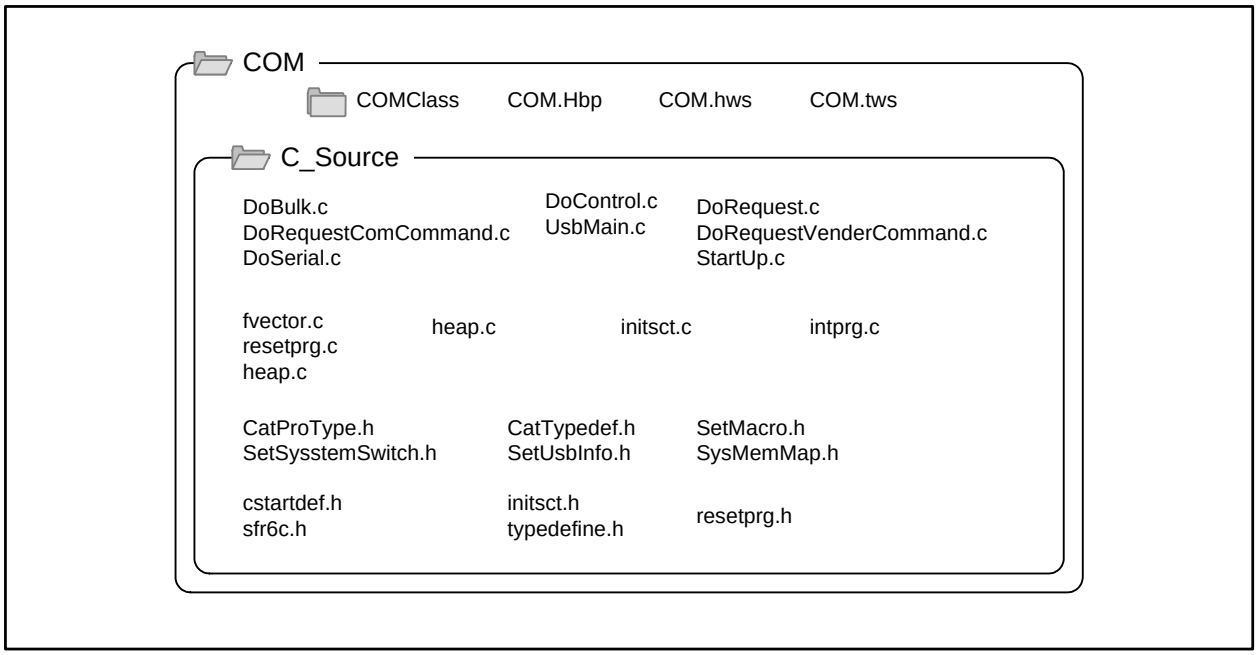


図 2.5 フォルダ内ファイル

2.4.2 コンパイルおよびリンク

サンプルプログラムは、High-performance Embedded Workshop4(以下、HEW4)によりコンパイルし、リンクします。以下に手順を示します。

(1) COMフォルダを任意の場所にコピーしてください。
(2) “COM.hws”というワークスペースファイルをダブルクリックし、HEW4を起動してください。
(3) 起動したHEW4で、「ビルド」から「全てをビルド」を選択すると、コンパイルおよびリンクが行われます。

(3)が完了すると、“COM\COMClass\Debug”フォルダ内に、実行ファイルである“COM.mot”と、マップファイルである“COM.map”が作成されます。COM.motはモトローラSタイプフォーマットファイルです。COM.mapにはプルグラムのサイズ、変数のアドレスが格納されています。

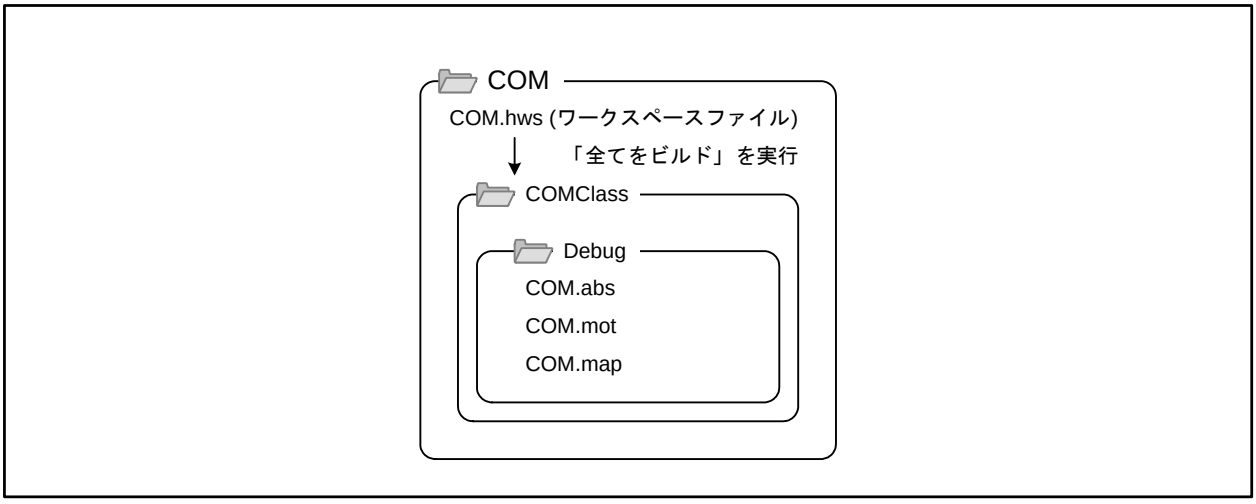


図 2.6 コンパイル結果

2.4.3 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“USB_CommClass_INF”フォルダに格納されています。

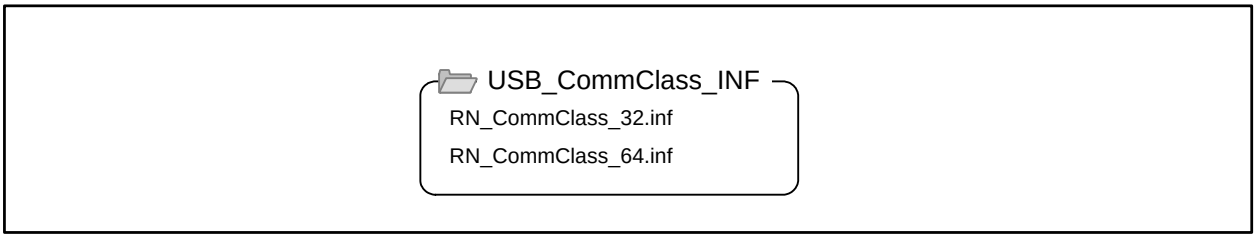
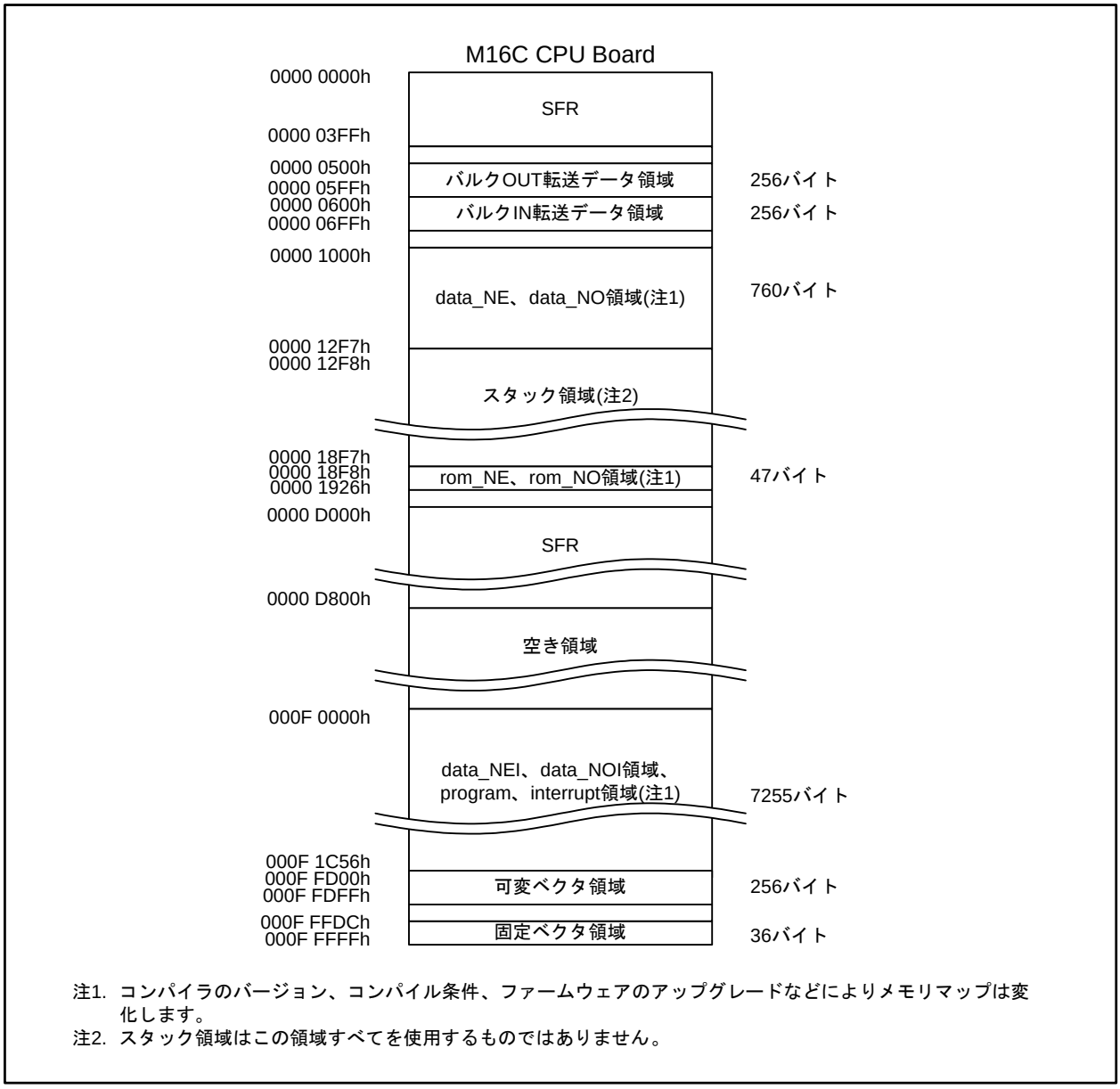


図 2.7 USB_CommClass_INFフォルダ内ファイル

2.5 プログラムのロードと実行方法

図 2.8にサンプルプログラムのメモリマップを示します。



注1. コンパイラのバージョン、コンパイル条件、ファームウェアのアップグレードなどによりメモリマップは変化します。
注2. スタック領域はこの領域すべてを使用するものではありません。

図 2.8 メモリマップ

プログラム各領域への割り付けは、HEW4で行ってください。割り付けの結果はCOM\COMClass\Debugフォルダ内のCOM.mapに出力されます。プログラムの配置を変更する場合は、HEW4を変更してください。

2.5.1 プログラムのロードと実行

以下に、サンプルプログラムのロード手順を示します。

(1) HEW4をインストールしたE8a用PCにE8aを接続してください。
(2) E8a付属の接続ケーブルでE8aとM16C CPU Boardを接続してください。
(3) M16C CPU Boardの電源を投入してください。
(4) COMフォルダ内のCOM.hwsを実行してください。
(5) デバッグ/接続 を選択してください。
(6) デバッグ/ダウンロード/ALL Download Modules を選択してください。プログラムがダウンロードされます。
(7) デバッグ/リセット実行 を選択してください。プログラムが実行されます。

2.6 PC間の通信方法

2.6.1 USBホストPCの設定

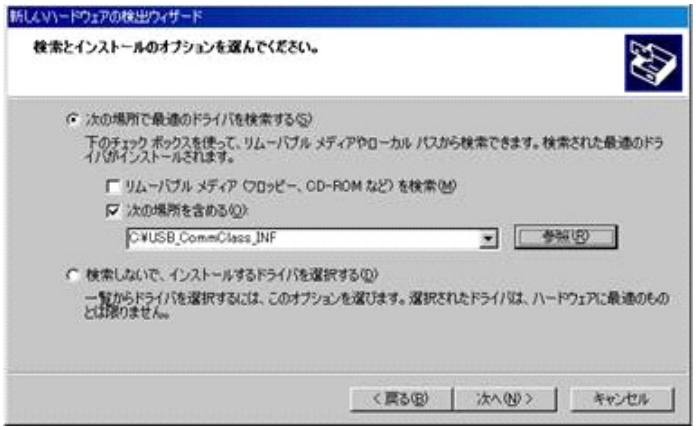
USBホストPCの設定を説明します。ここではWindows XPを用います。

USB Communication Class で動作させる場合、ドライバインストール時のインフォメーションファイルは、RN_CommClass.infを使用してください。また、USB接続した際に割り当てられるポート番号は、PC環境により異なります。「システムのプロパティ」、「デバイスマネージャ」で、割り当てられたCOMポート番号を確認してください。ここでは、COM4へ割り当てられたとして説明します。

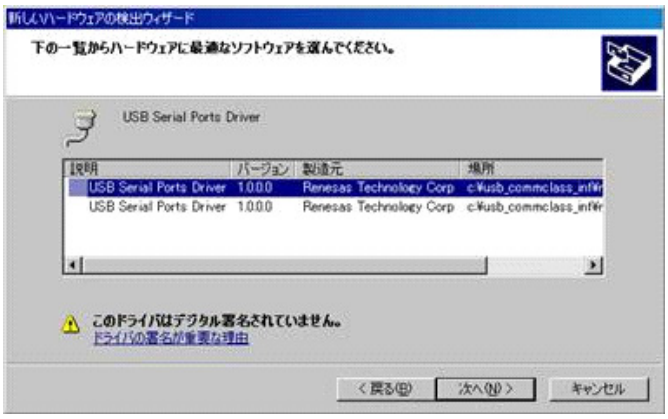
最初にドライバをインストールします。

(1) 2.5.1 プログラムのロードと実行に従い、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、ポートP1は0xAAを出力します。
(2) USBケーブルのシリーズBコネクタをM16 C CPU Board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。
(3)「一覧または特定の場所からインストールする(詳細)」を選択し、「次へ」をクリックしてください。

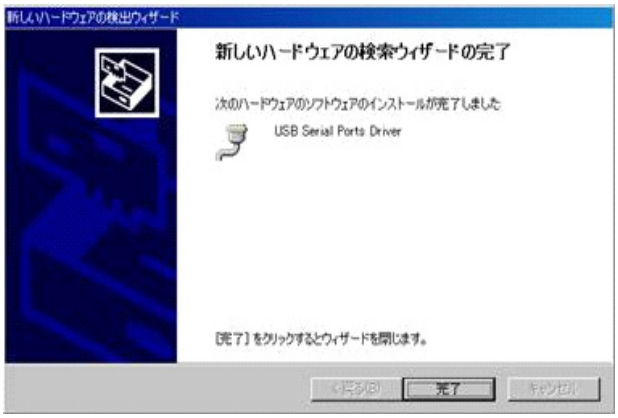
(4)「USB_CommClass_INF」フォルダを選択し、「次へ」をクリックしてください。



(5)「RN_CommClass.inf」フォルダを選択し、「次へ」をクリックしてください。



(6)「完了」をクリックしてください。



ドライバのインストール作業が終了しました。USB ホスト PC は M16C CPU Board をシリアル COM ポートとして認識します。

次に、WindowsOS 標準添付の通信ソフトであるハイパーターミナルを設定します。

(7) Windows キーをクリックし、スタート / プログラム / アクセサリ（もしくは「通信」の下）を選択してください。ハイパーターミナルが起動します。
(8) 任意の名前を入力してください。ここでは “USB-Serial” と入力します。名前を入力したら「OK」をクリックしてください。

(9)「COM4」を接続方法として選択し、「OK」をクリックしてください。

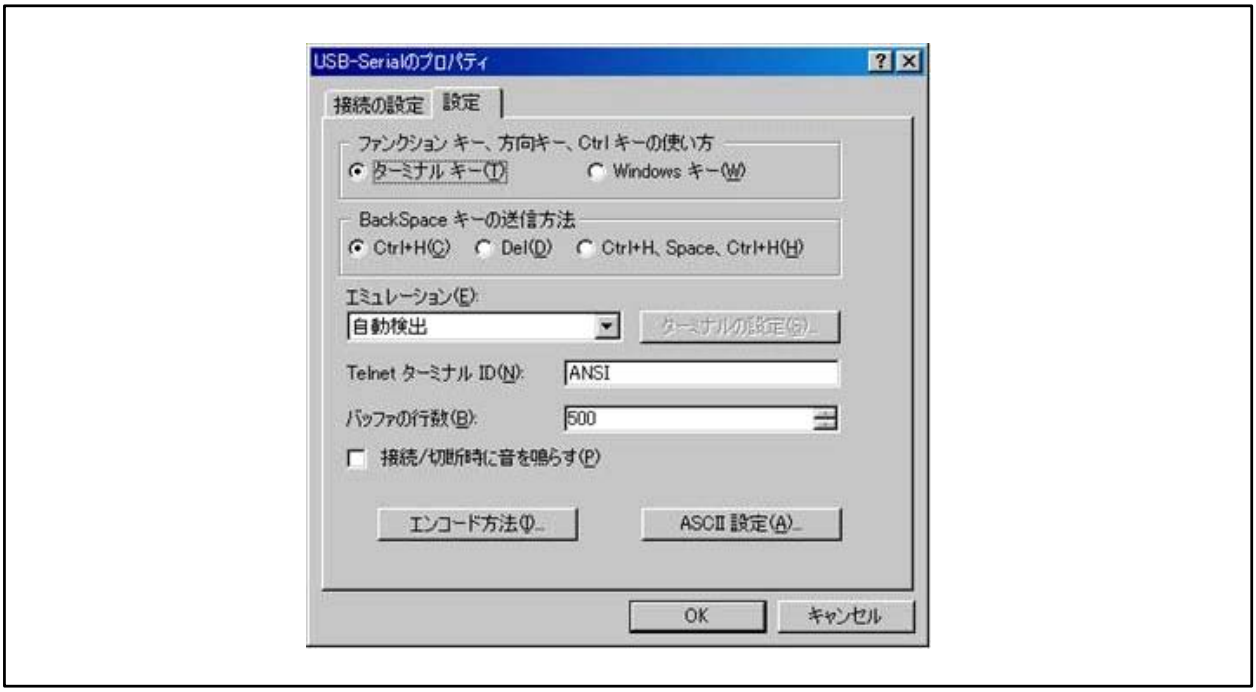
(10)シリアルポートを設定します。表 2.2 に示す値を設定し、「OK」をクリックしてください。表 2.2 に示す値を設定すると、ポート P1 は 0xAA を出力します。それ以外の値を設定すると、ポート P1 は 0x30 を出力します。

表 2.2 シリアルポートの設定値

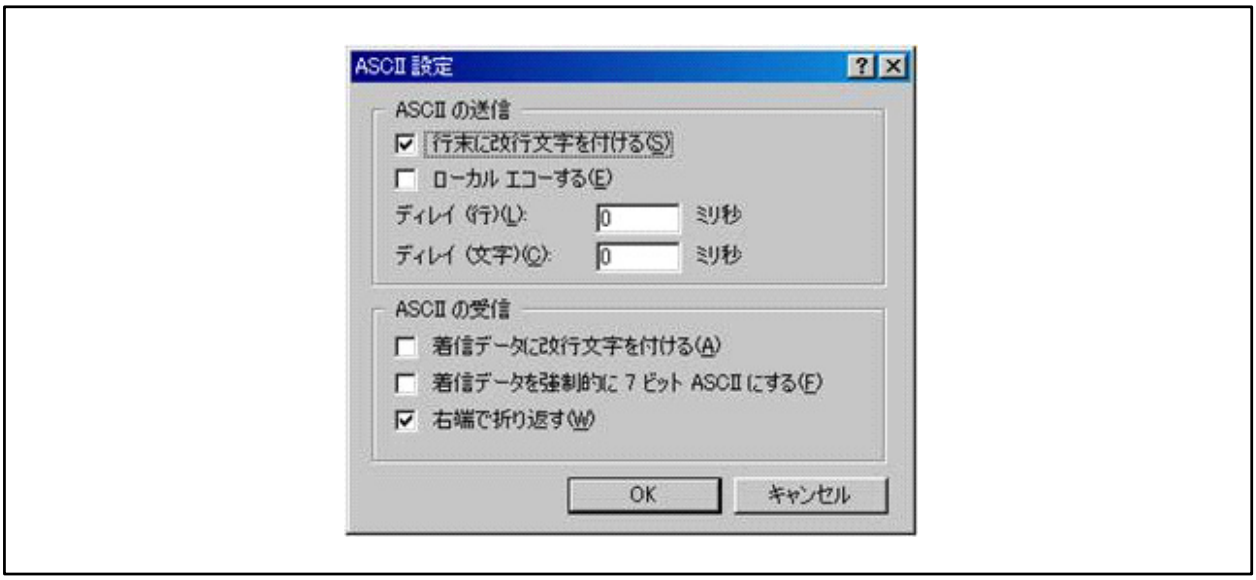
	設定値
ビット/秒	38400
データビット	8
パリティ	なし
ストップビット	1
フロー制御	Xon/Xoff

ハイパーターミナルの設定が完了しました。

(11)ASCIIの設定を行います。ファイルメニュー /プロパティ /設定 を選択し、「ASCII設定」をクリックしてください。



(12)「行末に改行文字を付ける」にチェックを入れて「OK」をクリックしてください。



ASCIIの設定が完了しました。

以上でUSBホストPCの設定が完了します。

2.6.2 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てください。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USBホストPCと同じ設定にしてください。

2.6.3 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を行うことができます。

USBホストPC側でキー入力すると、その入力文字がシリアル接続PCへと転送されます。同様に、シリアル接続PC側でキー入力すると、その文字はUSBホストPCへと転送されます。

テキストファイルおよびバイナリファイルを相手側PCへ送信するには、「転送」から「テキストファイルの転送」を選択してください。この際、まず受信側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ください。その後、送信側PCで転送/ファイルの送信/ZMODEMと選択してください。

注1. 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。

注2. 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)に対応したプロトコルを選択する必要があります。

3. サンプルプログラム概要

このサンプルプログラムは、M16C CPU Board上で動作します。
サンプルプログラムの特長を以下に示します。

- コントロール転送可能
- バルクOUT転送でUSBホストPCからデータを受信可能
- バルクIN転送でUSBホストPCにデータを送信可能
- シリアル接続PCとのシリアルデータの送受信可能
- バルクOUT転送で受信したデータをシリアル送信可能
- シリアル受信したデータをバルクIN転送可能

3.1 状態遷移

このサンプルプログラムの状態遷移図を図3.1に示します。

(1) リセット状態
ハードウェアリセット後、この状態になります。リセット状態では、主にM16C/6C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、メインルーチン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モニタし、データがあれば相手側PCへデータを出力します。

(3) USB通信状態
定常状態でUSB割り込みが受け付けられるとこの状態になります。USB割り込み要因は10種類あります。割り込み要求が発生すると、USB割り込みフラグレジスタ0～3の、対応するビットが“1”になります。USB通信状態では、割り込み要因に応じた転送方式でデータを送受信します。

(4) シリアル通信状態
定常状態でUART0受信割り込みが受け付けられるとこの状態になります。UART割り込み要因は、シリアル受信完了のみです。割り込み要求が発生すると、UART0送受信制御レジスタ1(U0C1)のビット3が“1”になります。

(5) サスペンド状態
サスペンドを検出すると、サスペンド状態に遷移します。USB RESUME割り込みが発生すると停止状態から定常状態に復帰します。

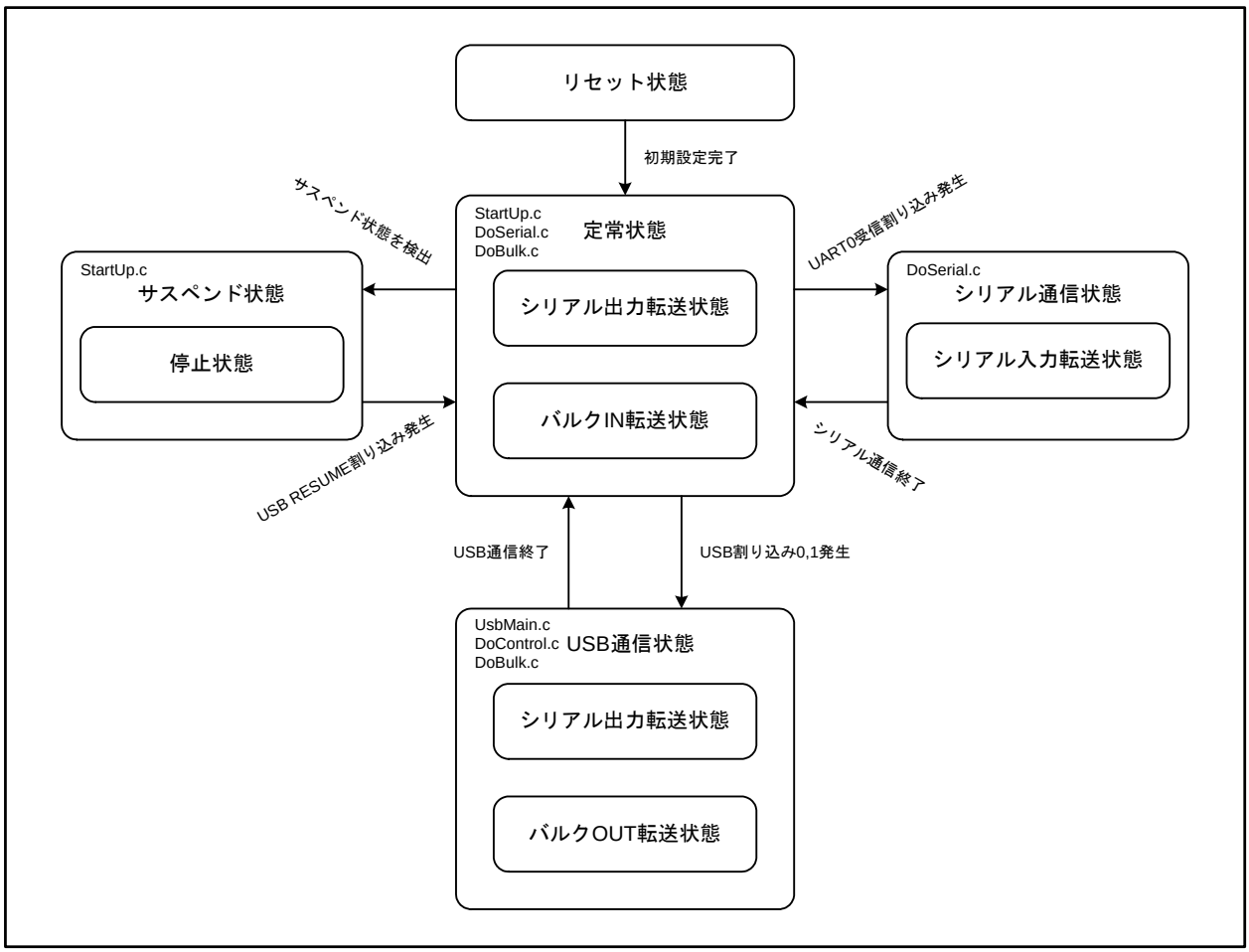


図 3.1　状態遷移図

割り込み優先順位は以下のとおりです:

USB RESUME 割り込み:　1
USB 割り込み 0、1:　2
UART0 受信割り込み:　3

この設定により、UART0 受信割り込み処理中に USB 割り込みが受け付けられて、シリアル受信処理が待たされることがないようになっています。

3.2 PC間通信概要

PC間の通信形態は、USB通信とシリアル通信に分けられます。

USB通信にはバルク転送が使用されます。バルク転送は、データの方向によってバルクIN転送とバルクOUT転送に分けられます。同様に、シリアル通信も、データの方向によってシリアル入力転送とシリアル出力転送に分けられます。

M16C CPU Boardは、バルクOUT転送用RAM領域(バッファ)とバルクIN転送用RAM領域(バッファ)にデータがあればそのデータを受信側PCへ出力します。

通信形態	USB通信	バルクIN転送
		バルクOUT転送
	シリアル通信	シリアル入力転送
		シリアル出力転送

図 3.2にPC間通信の概要示します。

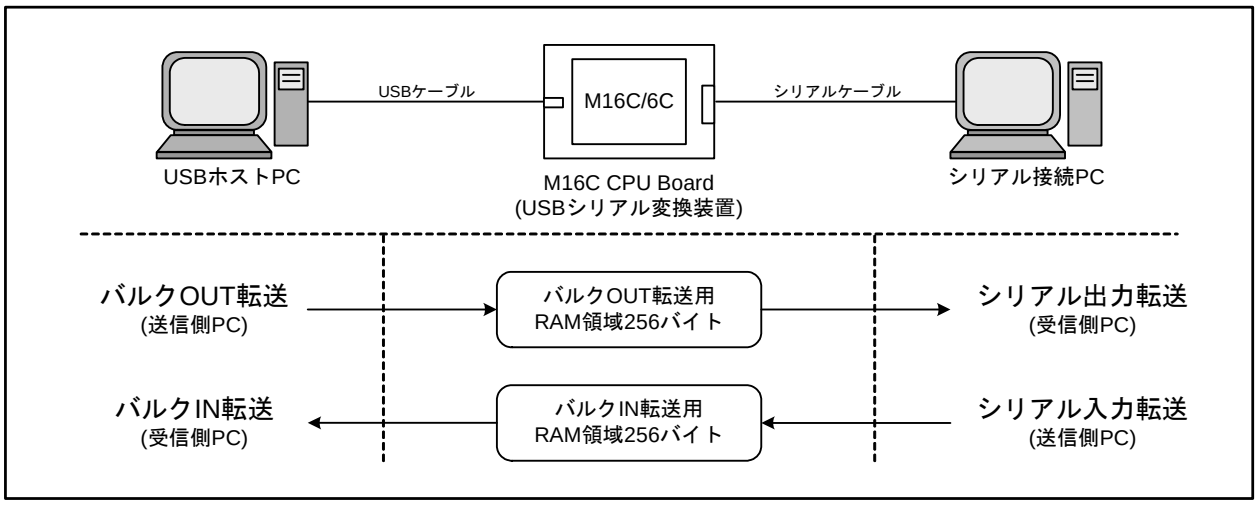


図 3.2 PC間通信概念

3.3 ファイル構成

このサンプルプログラムは、13個のソースファイルと11個のヘッダファイルで構成されています。
関数は、転送方式または機能ごとに一つのファイルにまとめてあります。
表 3.1に構成ファイルを示します。

表 3.1

ファイル名	主な役割
DoBulk.c	バルク転送を実行
DoControl.c	コントロール転送を実行
DoRequest.c	USBホストPCが発行するセットアップコマンドの処理
DoRequestComCommand.c	USB Communication Classで規定されたクラスコマンドの処理
DoRequestVenderCommand.c	ベンダコマンドの処理
DoSerial.c	シリアル送受信を実行、UART0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
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
fvector.c	固定ベクタテーブル
heap.c	ヒープエリアの割り付け
Initsct.c	各セクションのクリアと割り付け
Intprg.c	可変ベクタテーブル
resetprg.c	電源投入時マイコンの初期設定
CatProType.h	プロトタイプ宣言
CatTypedef.h	USBファームウェアで使用する基本の構造体定義
SetMacro.h	マクロ定義
SetSystemSwitch.h	システムの動作設定
SetUsbInfo.h	USB対応に必要な変数の初期設定
SysMemMap.h	メモリマップのアドレス定義
cstartdef.h	スタックおよびヒープサイズ設定ファイル
initsct.h	セクション定義ファイル
resetprg.h	スタックサイズ定義ファイル
sfr6c.h	M16C/6Cのレジスタ定義
typedefine.h	各アクセスサイズの定義

3.4 関数の機能

3.4.1から3.4.8に、ファイルに含まれる関数とその機能を示します。
図 3.3に関数の相関関係を示します。

3.4.1 UsbMain.c

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を呼び出します。また、USB ホスト PC と USB 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 3.2 UsbMain.c

格納ファイル	関数名	機能
UsbMain.c	BranchOhInt0	割り込み要因の判定および割り込みに応じた関数の呼び出し
	BranchOfInt1	割り込み要因の判定および割り込みに応じた関数の呼び出し
	BranchOfInt （注 1）	割り込み要因の判定および割り込みに応じた関数の呼び出し
	GetPacket	USBホストPCから転送されたデータをRAMに書く
	PutPacket	USBホストPCに転送するデータをUSBモジュールに書く
	SetControlOutContents	USBホストPCから送られたデータの書き換え
	BE2ByteRead	2バイトのデータをビッグエンディアンに変換する
	LE2ByteRead	2バイト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、フラグ、FIFOをクリアする
	SetUsbModule	USBモジュールの初期設定
	USBclear	リングバッファ、フラグをクリアする

注1. 本サンプルファームウェアでは、BranchOfInt関数は使用しません。

3.4.2 StartUp.c

ハードウェアリセット時、StartUp.c の SetPowerOnSection が呼び出されます。ここでは主に、M16C/6C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 3.3 StartUp.c

格納ファイル	関数名	機能
UsbMain.c	SetPowerOnSection	BSCの設定、モジュールおよびメモリの初期化を行い、メインルーチンへの移行
	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
	SciInit	UART0の初期化
	Set_SMR	UART0のSMRレジスタの初期設定を行う
	Set_EPInfoR	エンドポイント情報の書き込み
	error	エラー発生時、CPUをスリープモードに遷移
	SuspendResume	停止状態、定常状態の判定を行う
	Resume_int	停止状態からの復帰

3.4.3 DoSerial.c

DoSerial.c では、シリアル送受信および UART0 モジュールを制御します。

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	シリアル入力転送したデータをバルク IN 転送用領域に書く
	WriteBulkInArea	データをバルク IN 転送用領域に書く
	ExSerialOut	1 バイトデータを UART0 からシリアル出力転送する

3.4.4 DoRequest.c

コントロール転送時に、USB ホスト PC 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。このサンプルプログラムでは、ベンダ ID の値に “045B”(ベンダ : Renesas Electronics Corp.) を使用します。必要があれば「USB Implementers Forum」にて任意のベンダ ID を取得してください。

表 3.5 DoRequest.c

格納ファイル	関数名	機能
DoRequest.c	DecStandardCommands	USB ホスト PC が発行したコマンドをデコードし、標準コマンドに応じた処理を行う

3.4.5 DoControl.c

コントロール転送割り込み (SETUPTS) が受け付けられると、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 (EP0OTS、EP0ITR、EP0ITS) が発生すると、ActControlInOut がコマンドの転送方向に応じて、ActControlIn または ActControlOut を呼び出します。呼び出された関数は、データステージとステータスステージを制御します。

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制御
	ActControlIn	コントロール IN 転送 (データステージが IN 方向の転送) のデータステージとステータスステージを制御
	ActControlOut	コントロール OUT 転送 (データステージが OUT 方向の転送) のデータステージとステータスステージを制御
	ActControlInOun	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

3.4.6 DoBulk.c

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を制御します。

表 3.7 DoBulk.c

格納ファイル	関数名	機能
DoBulk.c	ActBulkOut	バルク OUT 転送を制御
	ActBulkIn	バルク IN 転送を制御

3.4.7 DoRequestVenderCommand.c

ベンダコマンドに応じた処理を行います。

表 3.8 DoRequestVenderCommand.c

格納ファイル	関数名	機能
DoRequestVenderCommand.c	DecVenderCommands	ベンダコマンドに応じた処理を行う

3.4.8 DoRequestComCommand.c

USB Communication Class コマンドに応じた処理を行います。

表 3.9 DoRequestComCommand.c

格納ファイル	関数名	機能
DoRequestComCommand.c	DecComCommands	USB Communication Class コマンドに応じた処理を行う

上側の関数は下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ます。USB 割り込みの発生によって遷移する USB 通信状態では、BranchOfInt0、BranchOfInt1 が他の関数を呼び出します。UART0 受信割り込みは、ActSerialIn 関数を呼び出します。なお、図 3.3 は関数の上下関係を示すもので、関数が呼び出される順序を示すものではありません。関数がどのような順序で呼び出されるかについては、4. サンプルプログラムの動作の図 4.1 メインルーチンを参照してください。

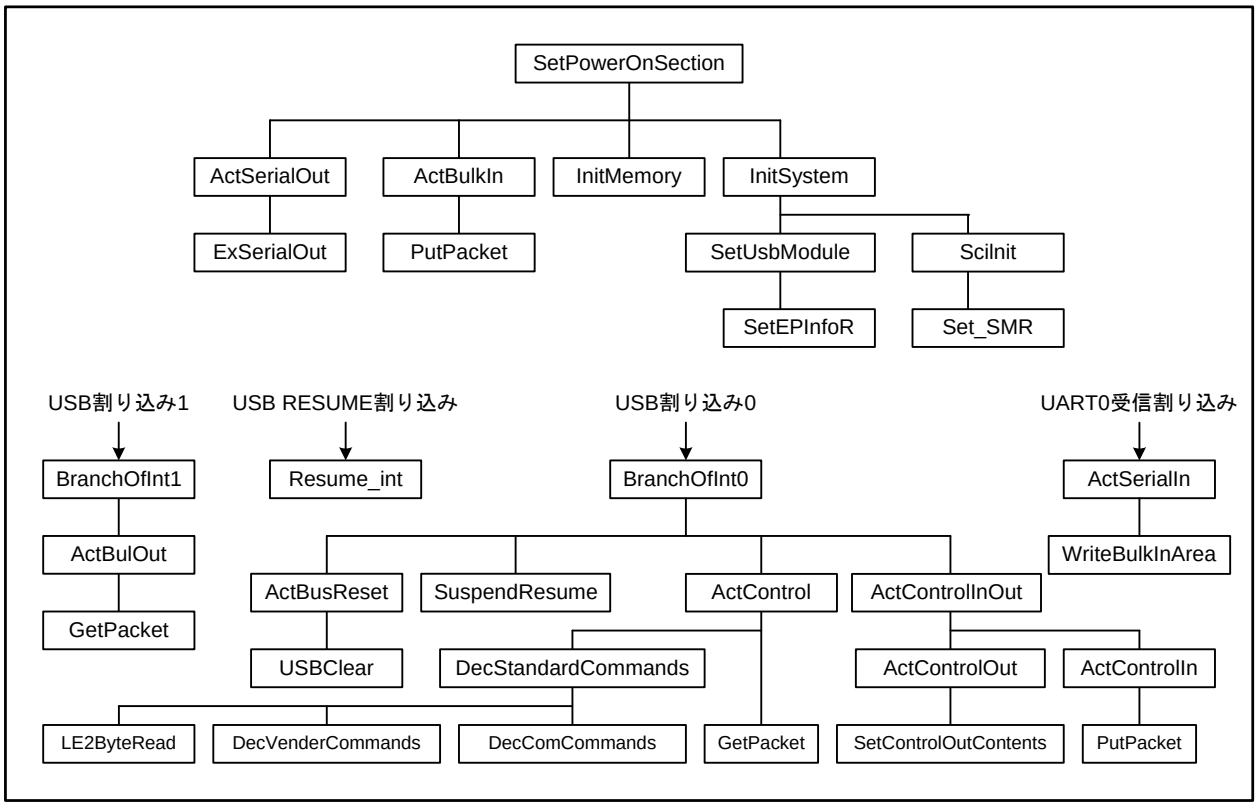


図 3.3 関数の相関関係

4. サンプルプログラムの動作

サンプルプログラムの動作を、M16C CPU Boardの動作と関連付けて説明します。

4.1 メインルーチン

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を初期化します。

図4.1にSetPowerOnSectionのフローチャートを示します。

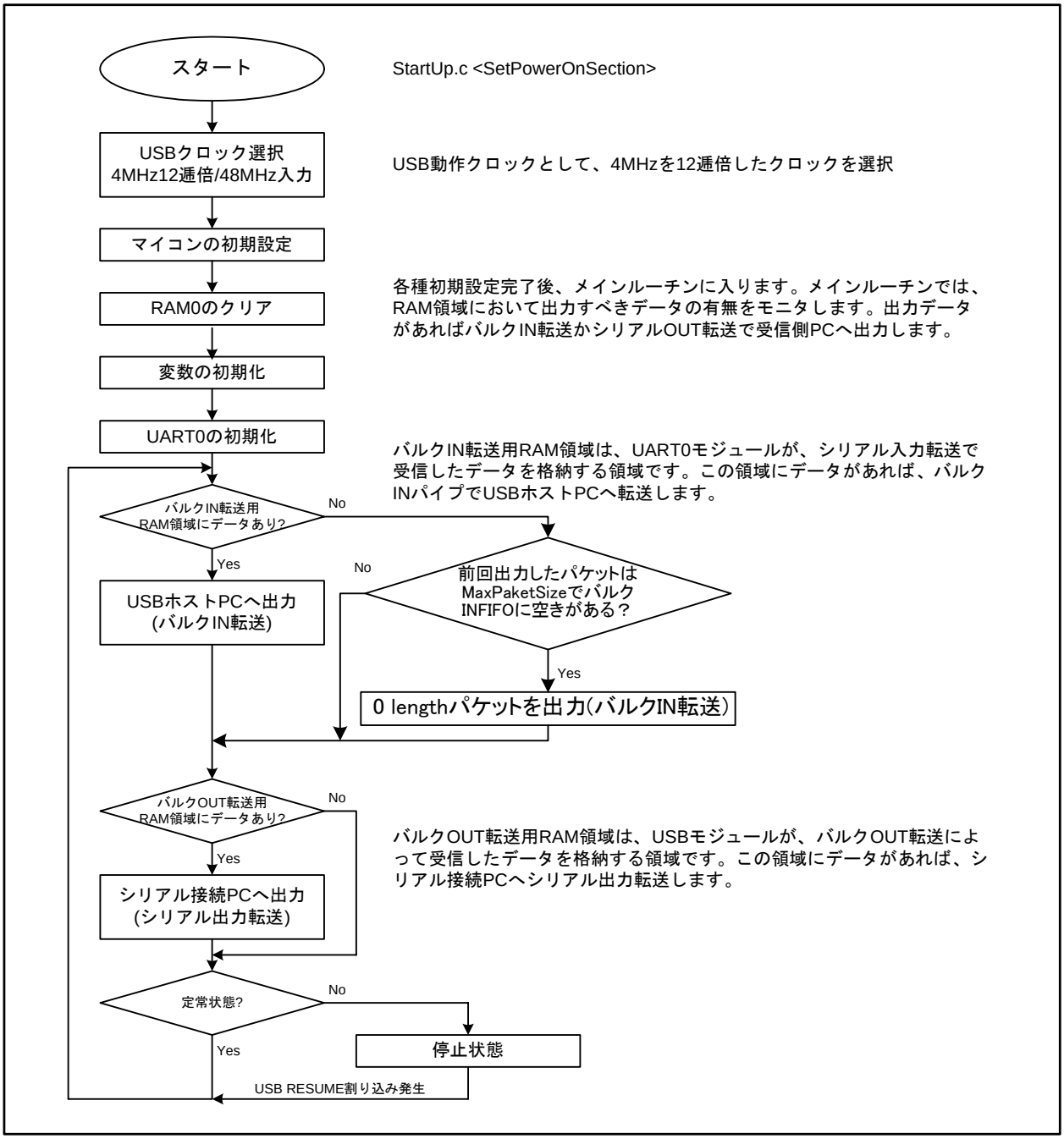


図4.1 メインルーチン

4.2　割り込みの種類

4.2.1　USB 割り込み

USB 割り込みには USB 割り込み 0、1 の 2 種類があります。定常状態で USB 割り込み要求が受け付けられると、USB 通信状態になり、割り込み要因に応じた処理へ分岐します。

USB 割り込みが呼び出す処理は、ケーブル接続(バスリセット)、コントロール転送、バルク OUT 転送です。なお、バルク IN 転送は、USB 割り込みではなくメインルーチンからの分岐で呼び出されます。詳細は 4.4.2 バルク転送を参照してください。

4.2.1.1　USB 割り込み要因

USB 割り込み i (i=0, 1) の割り込み要因は 10 種類あります。割り込み要因が検出されると、USB 割り込みフラグレジスタ i (USBIFRi) (i = 0～3) の対応するビットが“1”になり、USB 割り込み要求が発生します。割り込み要求が受け付けられると、USBISRi レジスタ (i=0～3) の設定値により USB 割り込み 1 か USB 割り込み 0 が発生します。割り込みが発生すると、BranchOfInt0 関数、BranchOfInt1 関数で、割り込み要因に応じた関数へ分岐します。

図 4.2 に USB 割り込みフラグレジスタと USB 割り込み要因を示します。

表 4.1 に USB 割り込みフラグレジスタと分岐先関数を示します。

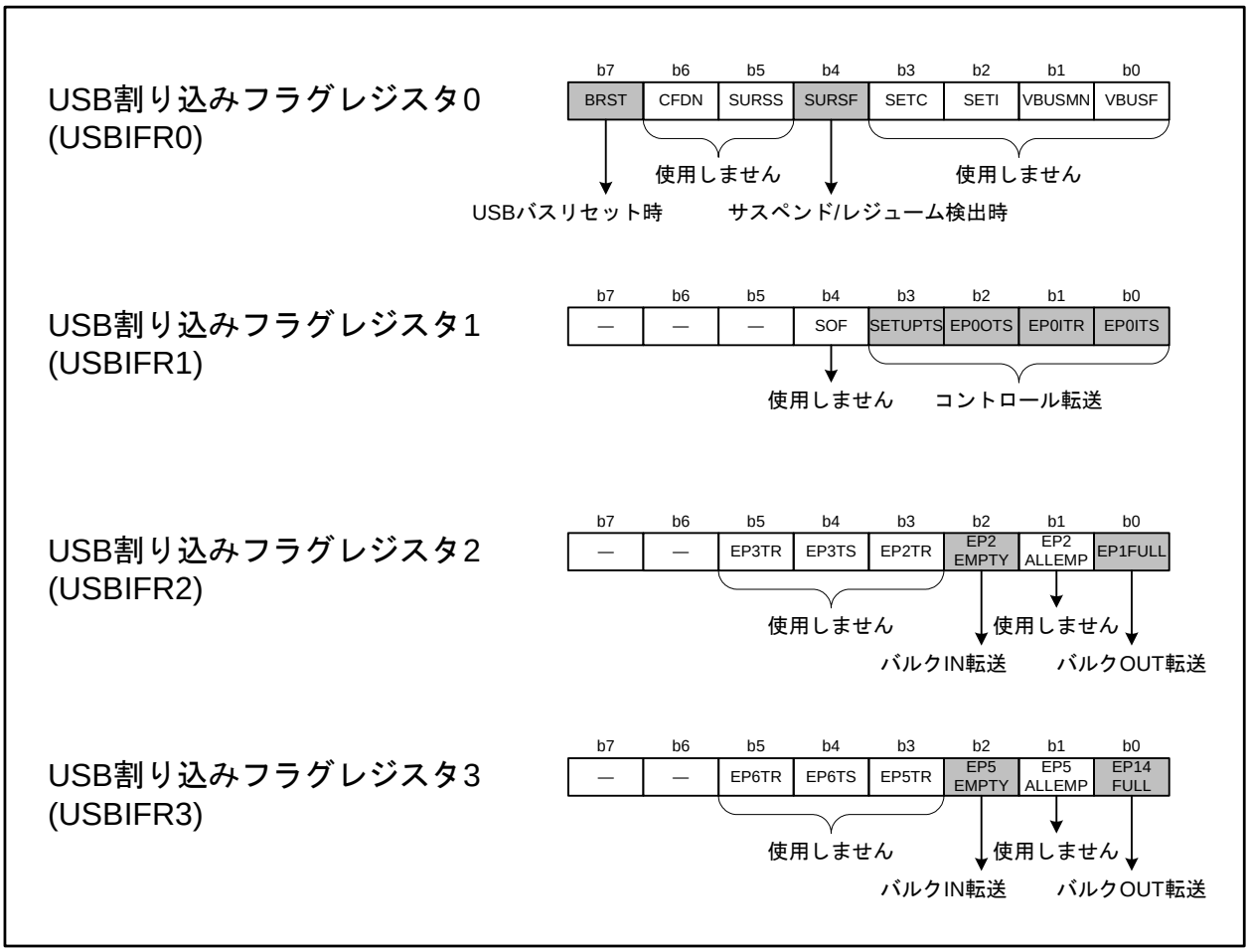


図 4.2　USB 割り込みフラグレジスタと USB 割り込み要因

表 4.1　USB割り込みフラグレジスタと分岐先関数

レジスタ名	ビット	ビット名	呼び出す関数名
USBIFR0	7	BRST	ActBusReset
	6	CFDN	—
	5	SURSS	—
	4	SURSF	SuspendResume
	3	SETC	—
	2	SETI	—
	1	VBUSMN	—
	0	VBUSF	—
USBIFR1	7	—	—
	6	—	—
	5	—	—
	4	SOF	—
	3	SETUPTS	ActControl
	2	EP0OTS(注1)	ActControlInOut
	1	EP0ITR(注1)	ActControlInOut
	0	EP0ITS	ActControlInOut
USBIFR2	7	—	—
	6	—	—
	5	EP3TR	—
	4	EP3TS	—
	3	EP2TR	—
	2	EP2EMPTY	— (メインルーチンからの分岐)
	1	EP2ALLEMP	—
	0	EP1FULL	ActBulkOut
USBIFR3	7	—	—
	6	—	—
	5	EP6TR	—
	4	EP6TS	—
	3	EP5TR	—
	2	EP5EMPTY	—
	1	EP5ALLEMP	— (メインルーチンからの分岐)
	0	EP4FULL	ActBulkOut

注1.　EP0OTSとEP0ITS割り込みは、コントロールIN転送、コントロールOUT転送の両方で使用します。コントロール転送の方向とステージを管理するためにTRANS_IN、TRANS_OUT、WAITの3つのステートがあります。詳細は4.4.1 コントロール転送を参照してください。

4.2.2 UART 割り込み

UART 割り込みは、UART0 受信割り込みの 1 種類のみです。定常状態で UART0 受信割り込みが受け付けられると、シリアル通信状態になり、シリアル入力転送 (ActSerialIN) でデータを送受信します。詳細は？？を参照してください。

4.2.2.1 UART 割り込み要因

UART 割り込み要因は、シリアル受信完了の 1 種類のみです。UART0 モジュールがデータを受信すると、UART0 送受信制御レジスタ 1(U0C1) のビット 3 が “1” になり、UART0 受信割り込みが発生します。割り込み要求が受け付けられると、ActSerialIn 関数が直接呼び出されます。

図 4.3 に USB 割り込みフラグレジスタと USB 割り込み要因を示します。

表 4.2 に USB 割り込みフラグレジスタと分岐先関数を示します。

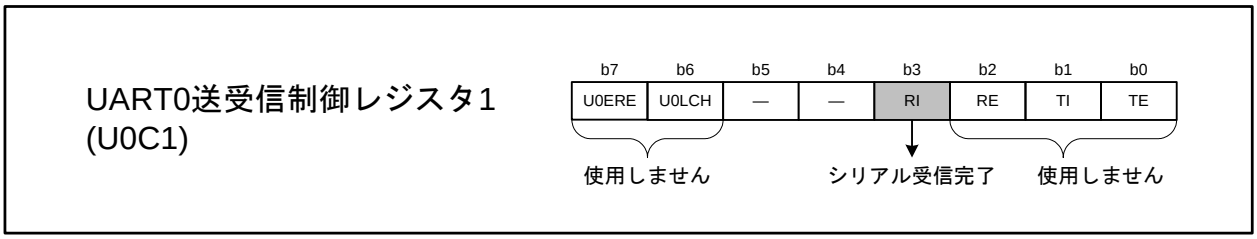


図 4.3 UART 割り込みフラグ

表 4.2 USB 割り込みフラグレジスタと分岐先関数

レジスタ名	ビット	ビット名	呼び出す関数名
U0C1	7	U0ERE	—
	6	U0LCH	—
	5	—	—
	4	—	—
	3	Ri	ActSerialIn
	2	RE	—
	1	TI	—
	0	TE	—

4.2.3 バスリセット(BRST)割り込み

USB ホスト PC は、USB データバスにデバイスが接続されると、バスリセット信号を出力します。M16C CPU Boardが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。

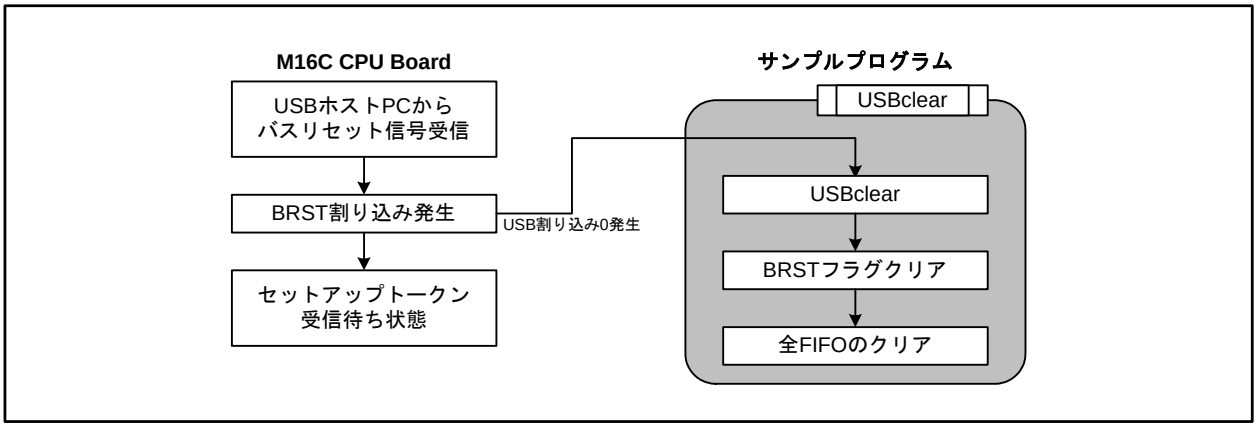


図 4.4 BRST割り込み

4.2.4 サスペンド/レジューム検出(SURSF)割り込み

サスペンド/レジューム状態を検出した時に発生します。

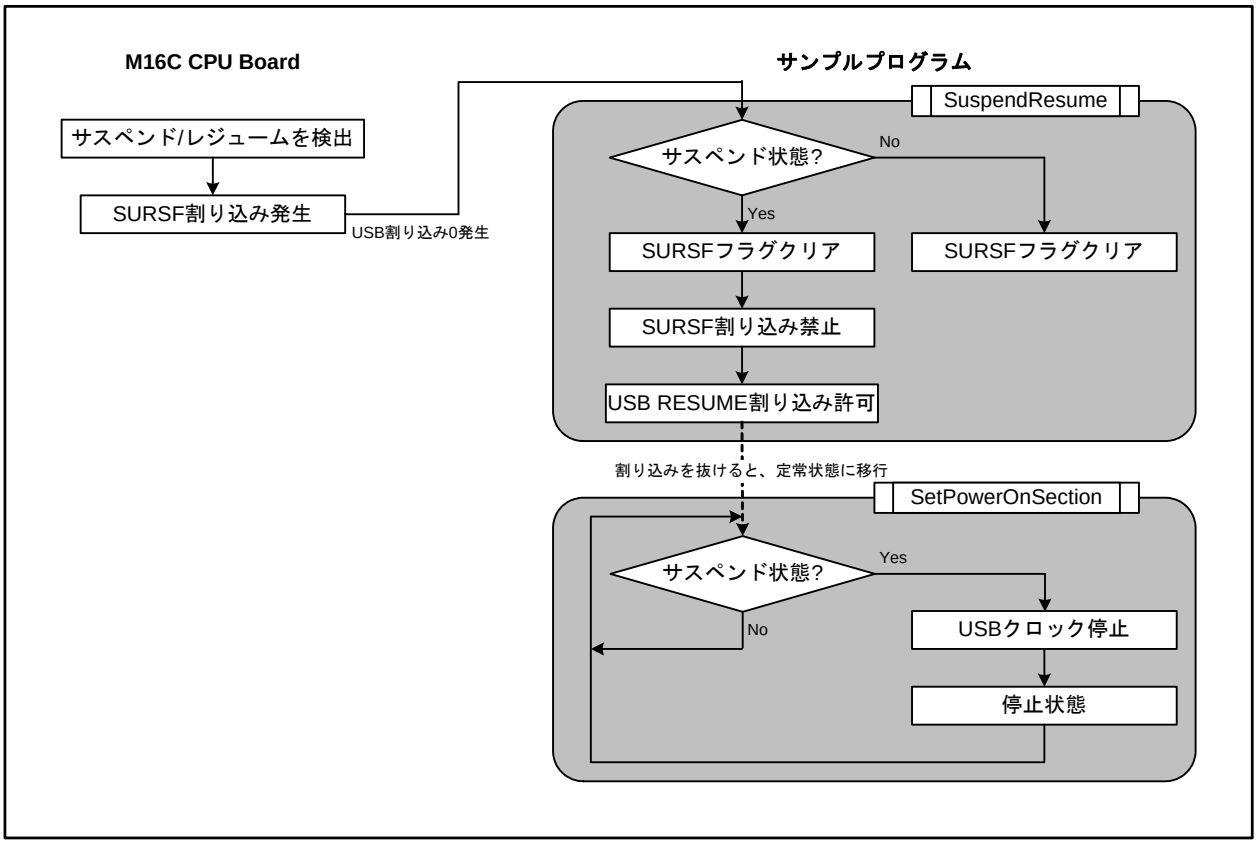


図 4.5 SURSF割り込み

4.3 EPINFO

M16C/6CのUSBファンクションモジュールは、初期化時に、ソフトウェアでエンドポイント構成を設定する必要があります。設定可能な転送タイプを以下に示します。

- コントロール転送
- バルクOUT転送
- バルクIN転送
- インタラプトIN転送

コントロール転送以外は、USBエンドポイント情報レジスタ(以下、USBEPIR)で、ENDPoint番号、Interface番号、Alternate番号、MaxPacketSizeを設定できます。

図4.6にエンドポイント構成を示します。

表4.3にエンドポイント構成を実現するUSBEPIRの設定値を示します。詳細はM16C/6Cユーザーズマニュアル ハードウェア編を参照してください。

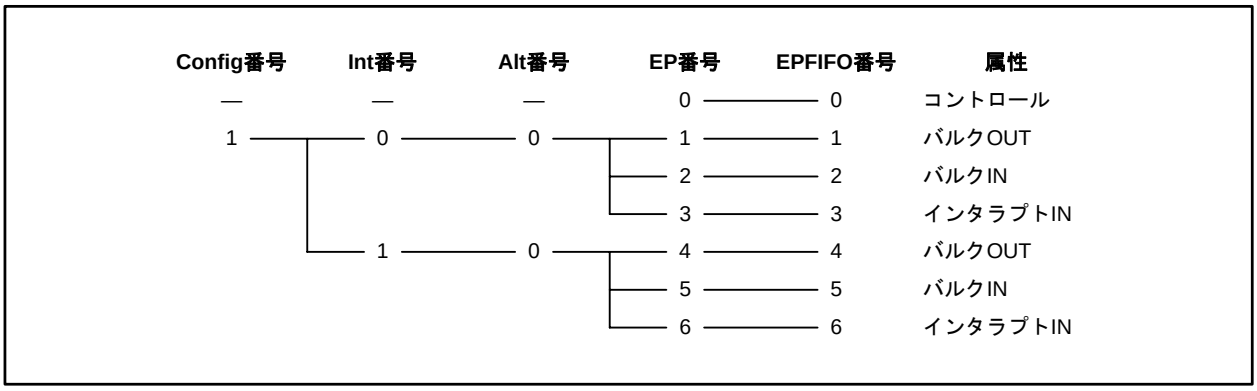


図4.6 エンドポイント構成

表4.3 USBEPIRの設定値

EPIR	設定値(16進数)	転送タイプ	EP番号	Conf	Int	Alt	MaxPacketSize	EPFIFO番号
0	00_00_20_00_00	コントロール	0	—	—	—	16バイト	0
1	14_20_80_00_01	バルクOUT	1	1	0	0	64バイト	1
2	24_28_80_00_01	バルクIN	2	1	0	0	64バイト	2
3	34_38_20_00_03	インタラプトIN	3	1	0	0	16バイト	3
4	45_20_80_00_05	バルクOUT	4	1	1	0	64バイト	4
5	55_28_80_00_04	バルクIN	5	1	1	0	64バイト	5
6	65_38_20_00_06	インタラプトIN	6	1	1	0	16バイト	6

4.4　転送方式

4.4.1　コントロール転送

コントロール転送は、USB 割り込みフラグレジスタ 1 のビット 3 ～ 0 を使用した USB 転送処理です。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（ない場合もあります）、ステータスステージで構成されています。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ています。

コントロール転送は、データステージにおけるデータの方向によって、2 つに分けることができます。USB ホスト PC から M16C CPU Board へデータを送信する場合がコントロール OUT 転送、その逆がコントロール IN 転送です。

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データステージからステータスステージへ切り替わります。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 IN 転送、コントロール OUT 転送を行う関数を呼び出します。このため、現在 IN、OUT 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ェアがステートによって管理し (図 4.8 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。

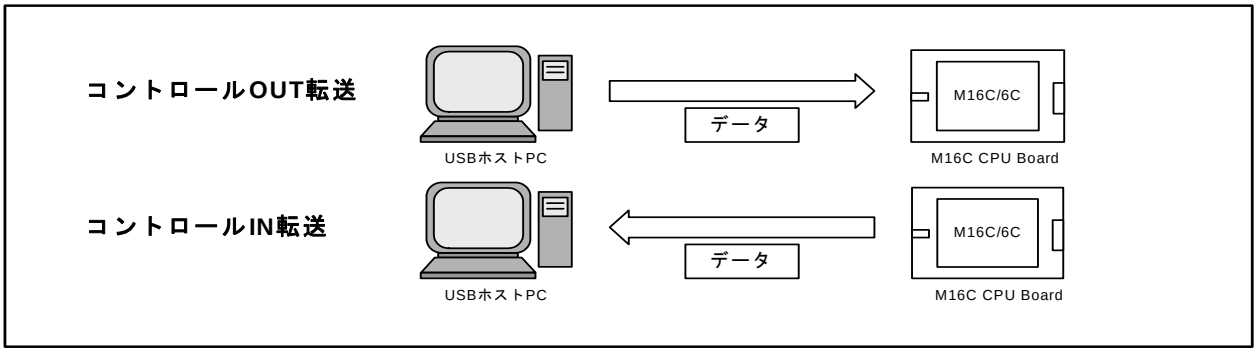


図 4.7　データステージにおけるデータの方向 (コントロール転送)

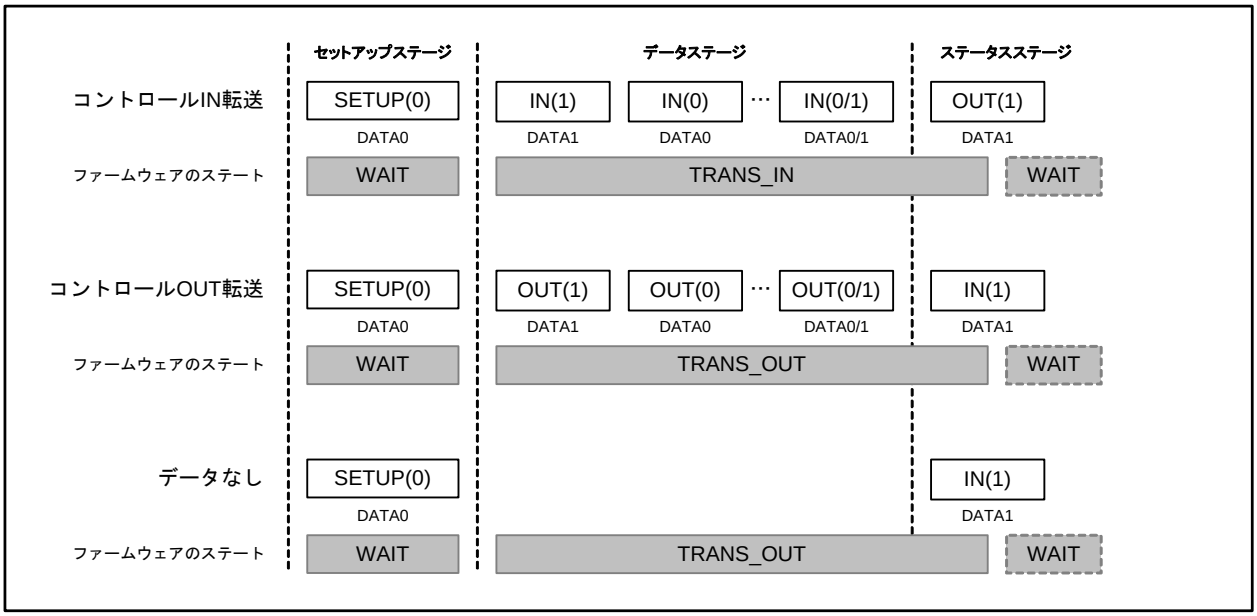


図 4.8　コントロール転送における各ステージの構成

4.4.1.1 セットアップステージ

セットアップステージでは、USB ホスト PC と M16C CPU Board がコマンドの送受信を行います。コントロール IN 転送、コントロール OUT 転送ともに、ファームウェアのステートは 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 IN 転送またはコントロール OUT 転送の区別を行い、データステージにおけるファームウェアのステート (TRANS_IN、TRANS_OUT) を決定します。

- コントロール IN となるコマンド　GetDescriptor　(TRANS_IN)　標準コマンド

図 4.9 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。

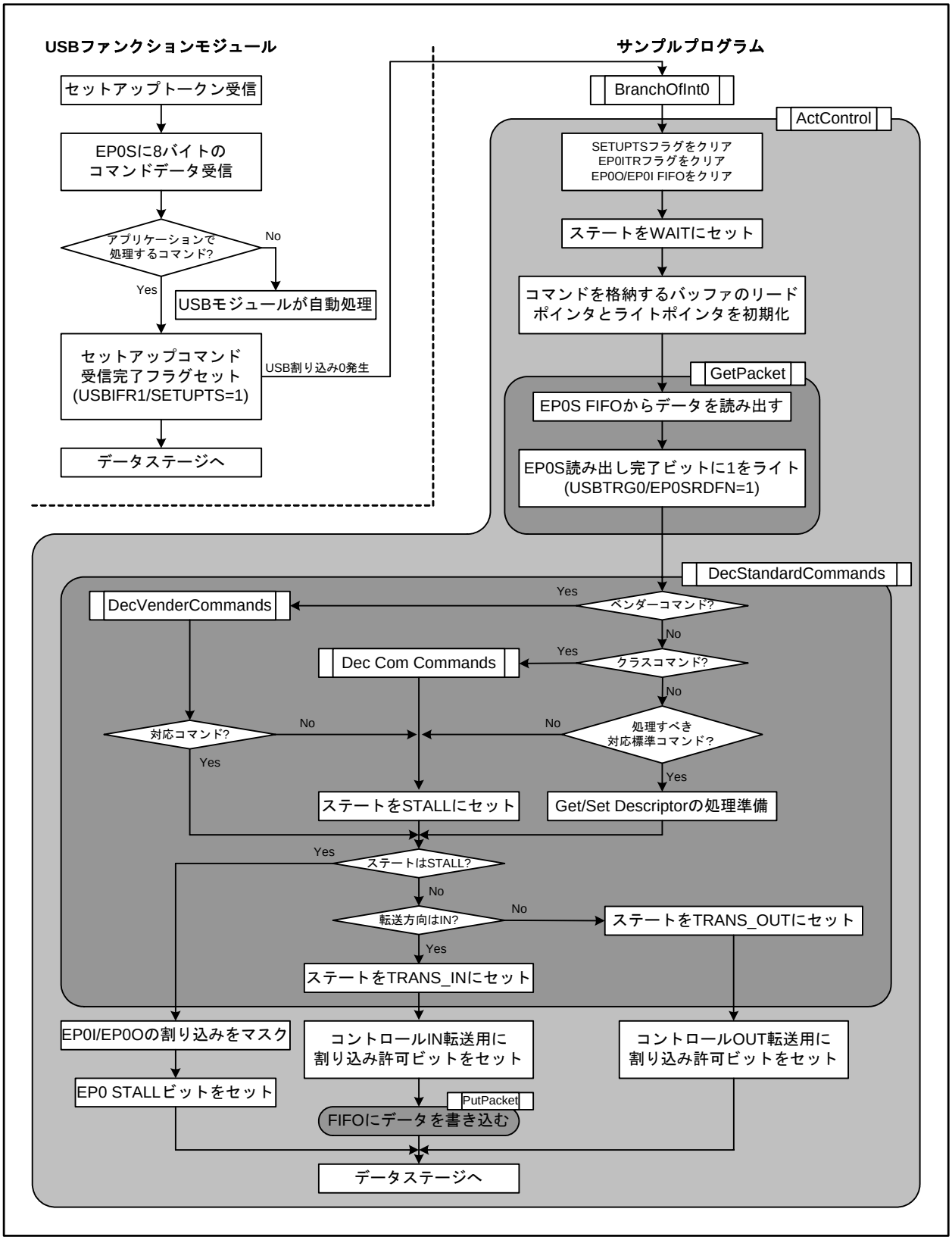
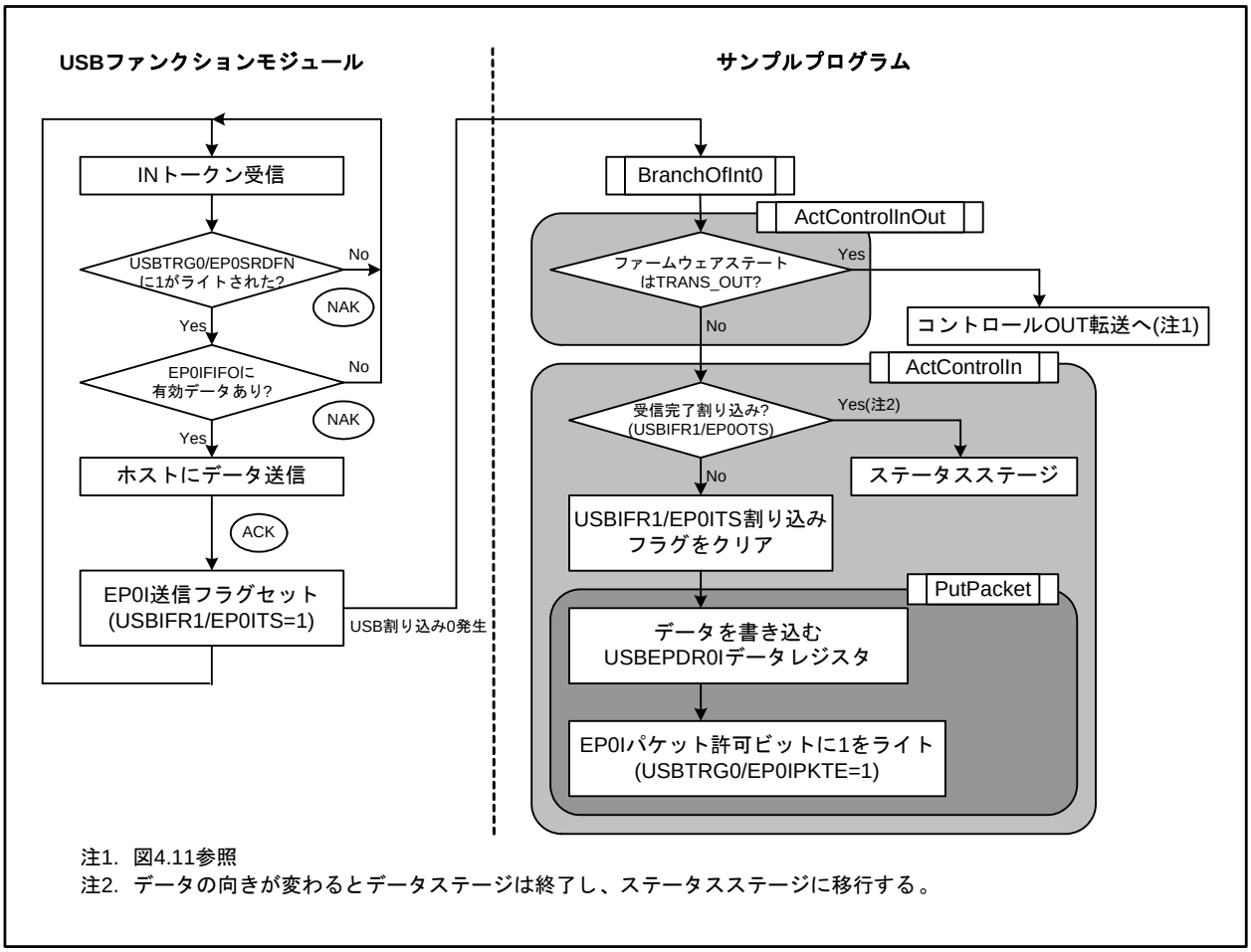


図 4.9　セットアップステージ

4.4.1.2 データステージ

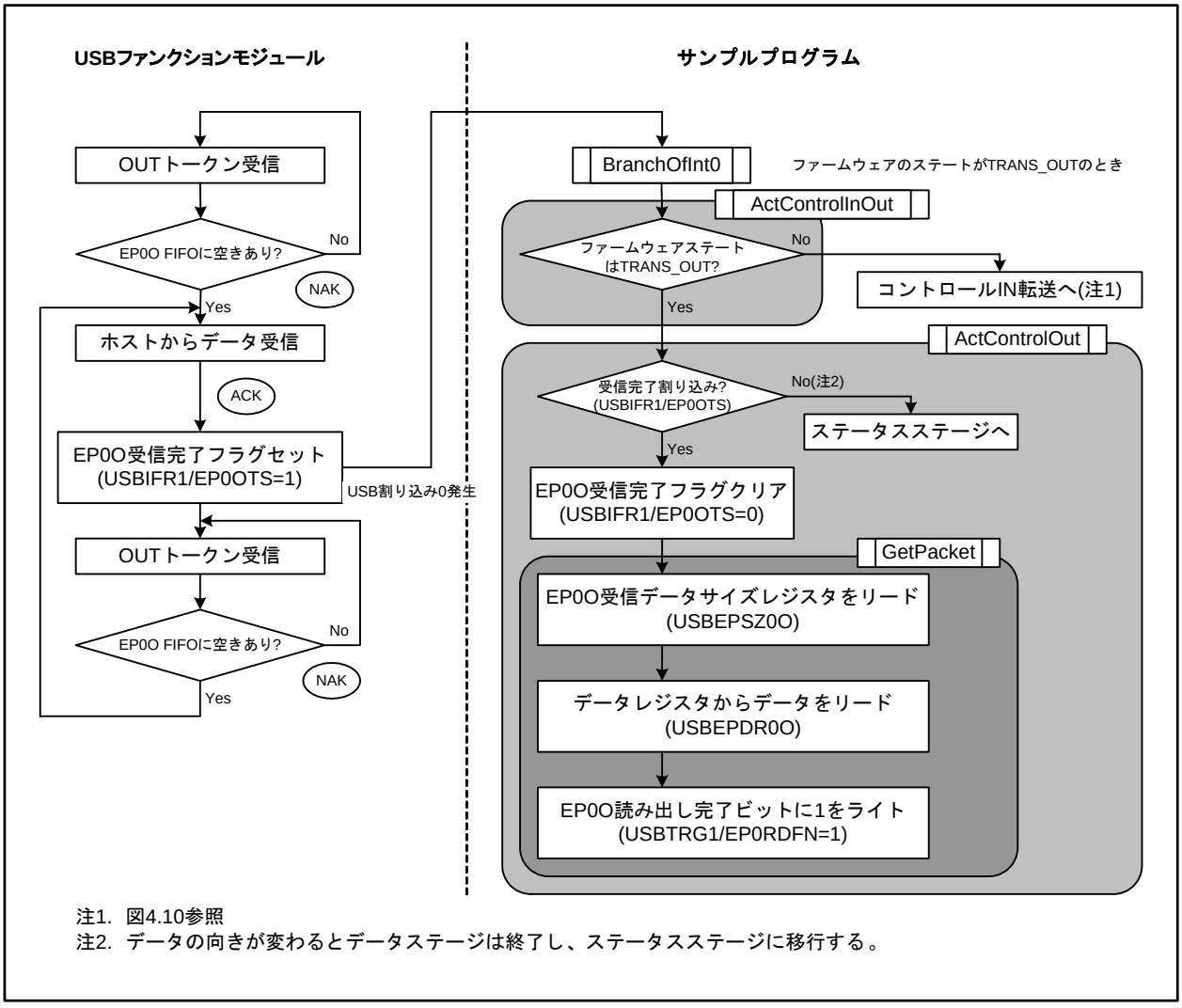
データステージでは、USB ホスト PC と M16C CPU Board がデータの送受信を行います。ファームウェアのステートは、セットアップステージで行ったコマンドのデコードの結果により、コントロール IN 転送であれば TRANS_IN に、コントロールアウト転送であれば TRANS_OUT になります。

図 4.10、図 4.11 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。



注1. 図4.11参照
注2. データの向きが変わるとデータステージは終了し、ステータスステージに移行する。

図 4.10 データステージ(コントロール IN 転送)



注1. 図4.10参照
注2. データの向きが変わるとデータステージは終了し、ステータスステージに移行する。

図 4.11　データステージ(コントロールOUT転送)

4.4.1.3 ステータスステージ

ステータスステージは、データステージと逆方向のトークンによって開始されます。コントロール IN転送では、、ホストコントローラからのOUTトークンによってステータスステージへ移行します。コントロールOUT転送では、ホストコントローラからのINトークンによってステータスステージへ移行します。

図 4.12、図 4.13 にコントロール転送のステータス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。

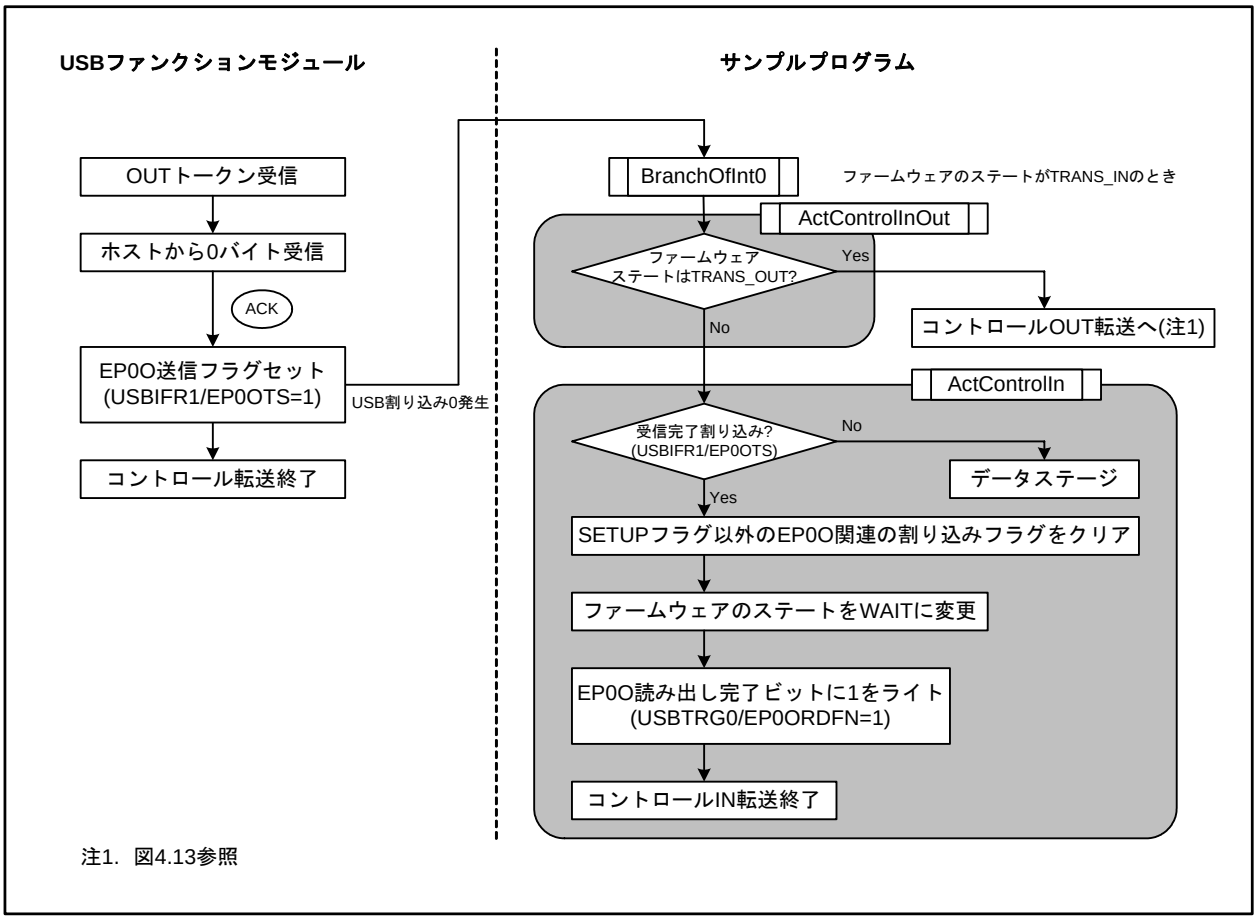


図 4.12 ステータスステージ(コントロールIN転送)

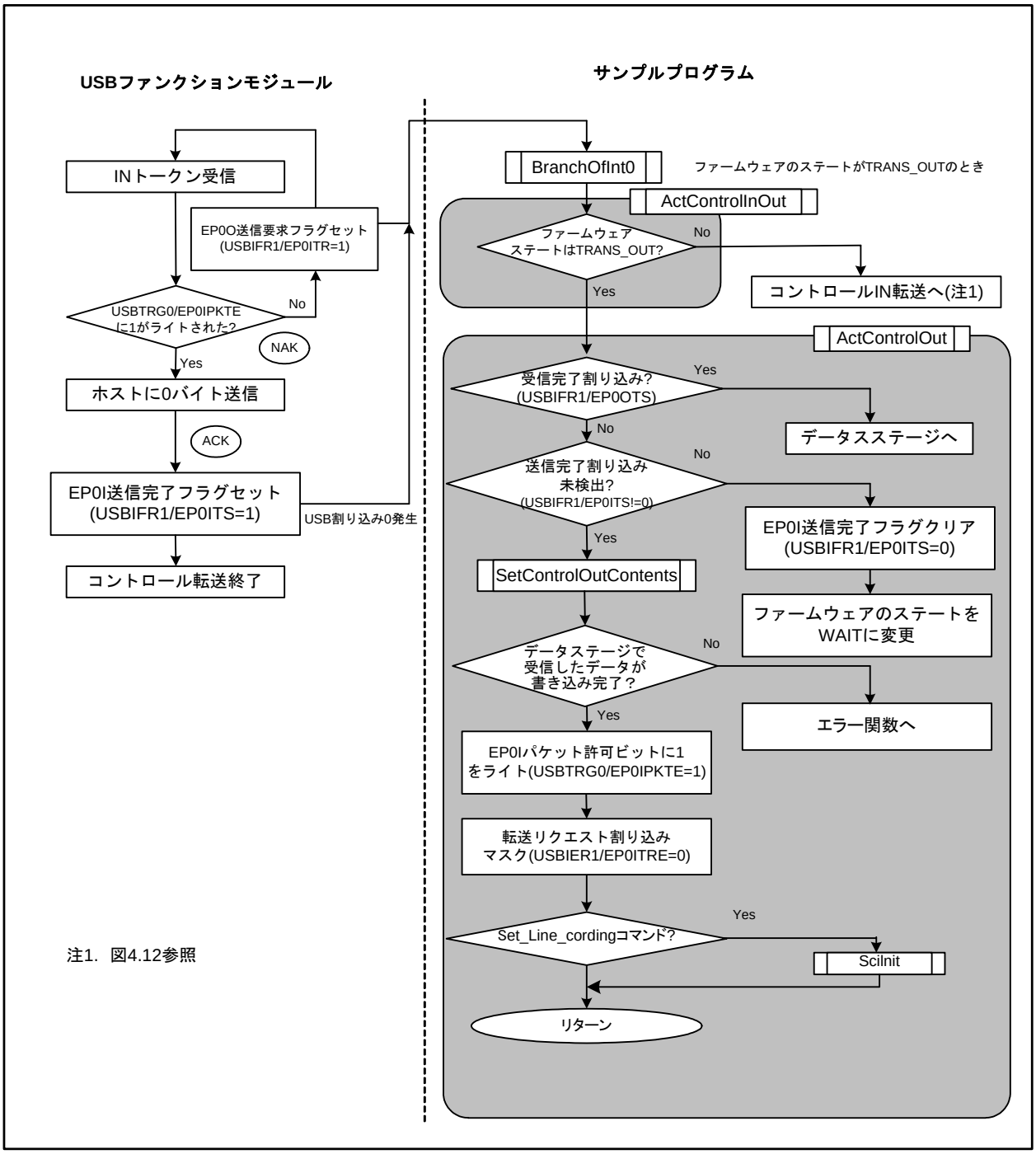


図 4.13　ステータスステージ(コントロールOUT転送)

4.4.2 バルク転送

バルク転送は、USB割り込みフラグレジスタ2のビット2～0を使用したUSB転送処理です。なお、バルクIN転送はUSB割り込みではなくメインルーチンからの分岐で呼び出されるため、このサンプルプログラムではビット3は使用しません。

バルク転送は、データの方向によって、2つに分けることができます。USBホストPCからM16C CPU Boardへデータを送信する場合がバルクOUT転送、その逆がバルクIN転送です。

なお、以降のバルク転送の説明は、エンドポイント1と2を使用した場合を例に挙げ説明します。

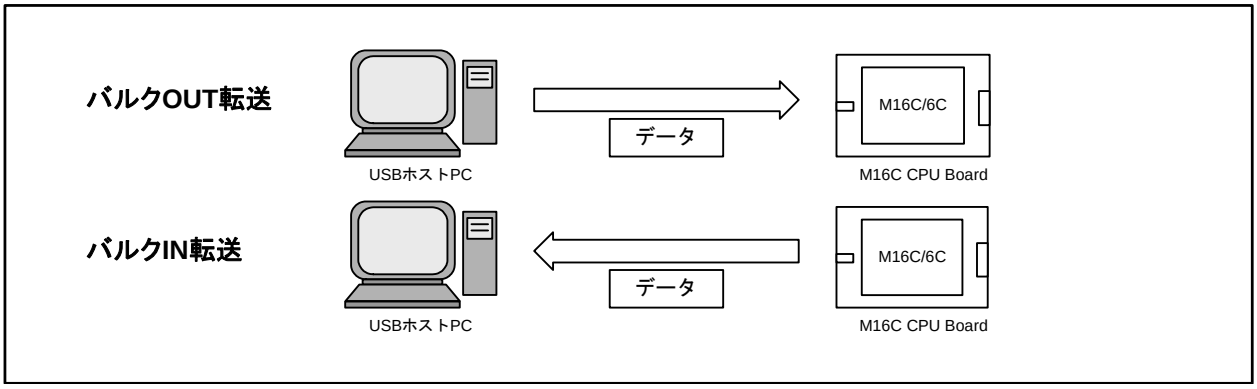


図 4.14 バルク転送

4.4.2.1 バルクOUT転送

バルクOUT転送におけるサンプルプログラムの動作を図4.15に示します。

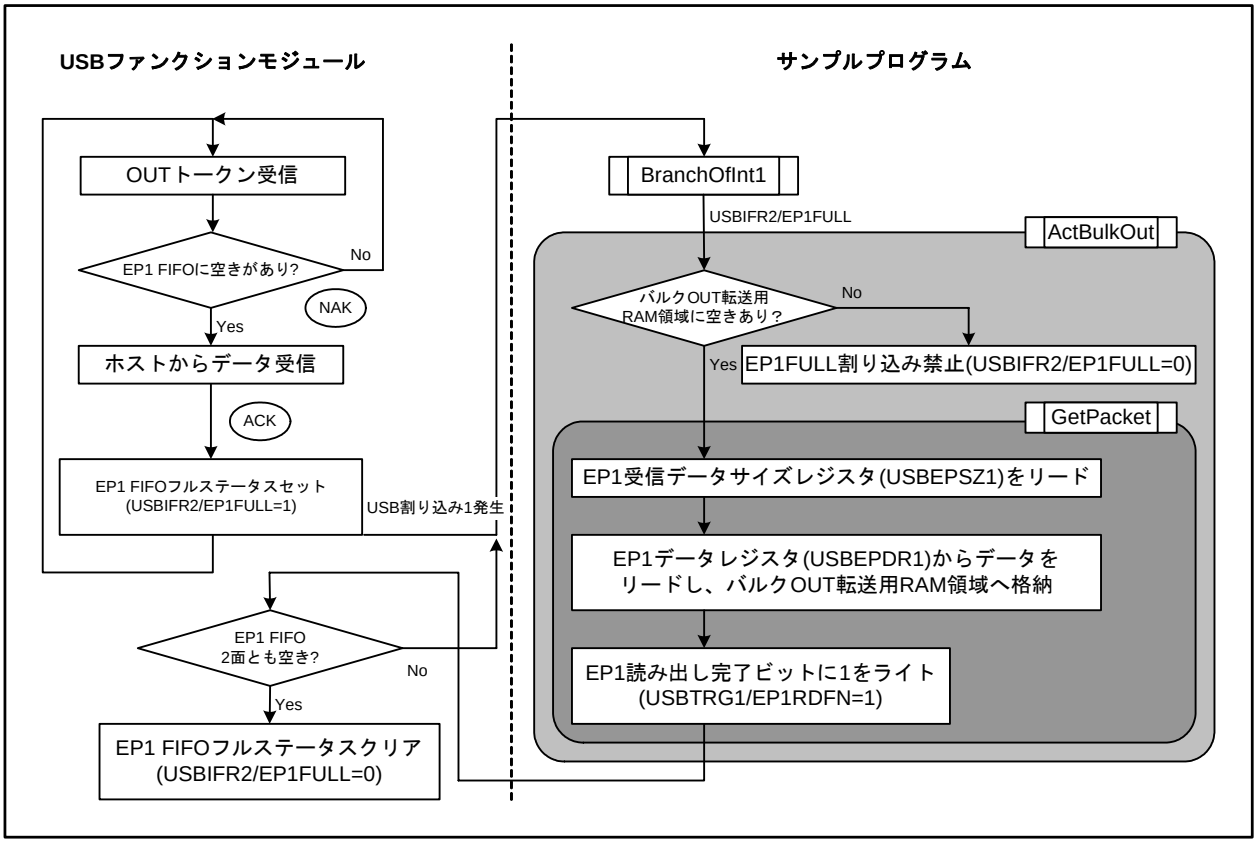


図 4.15 バルクOUT転送

4.4.2.2　バルク IN 転送

図 4.16 にバルク IN 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 IN 転送は USB 割り込みではなくメインルーチンからの分岐で呼び出されます。

バルク IN 転送用 RAM 領域に格納されているデータを EP2 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入力転送が禁止の場合は、USBEPDR2 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入力転送を許可します。

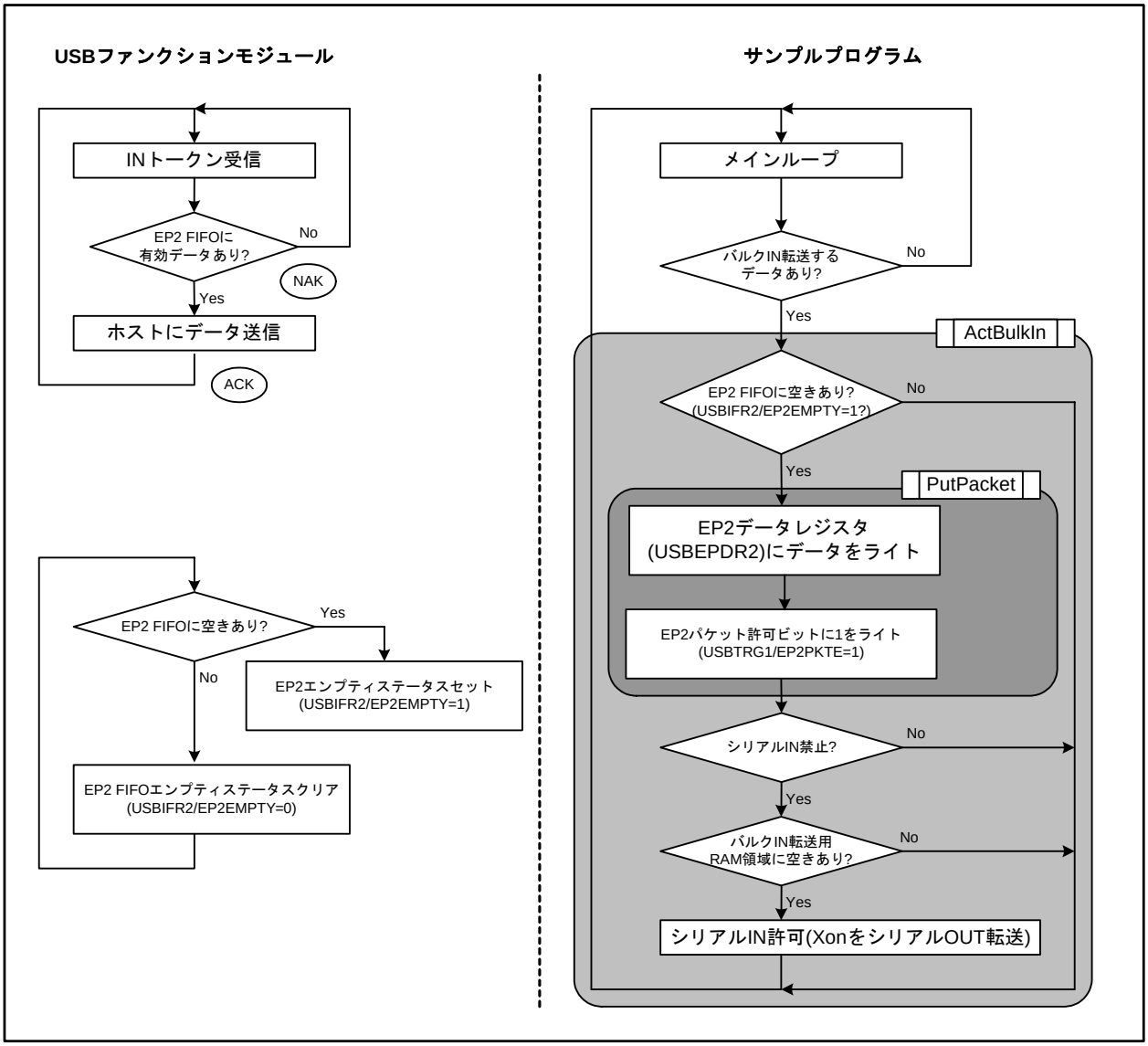


図 4.16　バルク IN 転送

4.4.3　シリアル転送

シリアル転送には、UART0 モジュールを使います。シリアル出力転送はメインルーチンからの分岐によって実行され、シリアル入力転送は割り込みによって実行されます。シリアル入力転送は、UART0 送受信制御レジスタ 1(U0C1)の RI フラグを使用します。

4.4.3.1　シリアル出力転送

シリアル出力転送におけるサンプルプログラムの動作を図 4.17 に示します。シリアル出力転送は、バルク OUT 転送用 RAM 領域にデータがあれば、メインルーチンから分岐して ActSerialOut 関数を呼び出し、UART0 モジュールを制御してシリアル出力転送によりデータを送信します。もし、バルク OUT 転送用 RAM 領域に空き容量がない状態でバルク OUT 転送が禁止である場合、このシリアル出力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 OUT 転送を許可します。

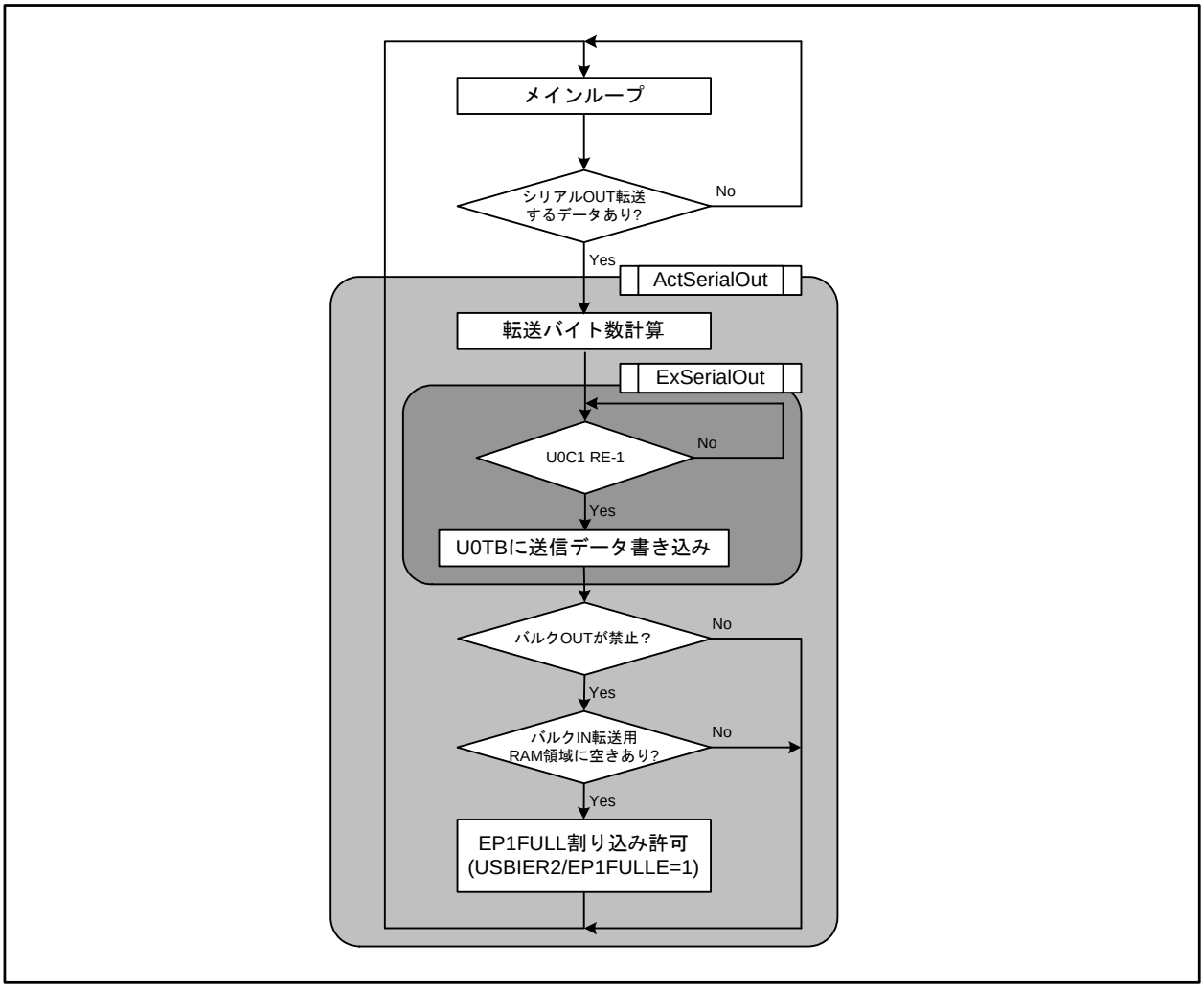


図 4.17　シリアル出力転送

4.4.3.2 シリアル入力転送

シリアル入力転送におけるサンプルプログラムの動作を図 4.17に示します。UART0受信割り込みが発生した場合、ActSerialIn関数が呼ばれます。

UART1 受信では、OER が発生した場合は、UART0 受信時と同様に U0RB からデータを読み出します。OER が伴わない受信エラーの場合は、U0RB の値をダミーリードして捨て、エラーフラグをクリアします。このときブレーク信号も受信している場合は、シリアル受信を禁止にし、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”を保持するので、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 関数が繰り返し呼ばれます。また、この期間中に USB 割り込みを受け付けるために、USB と UART0 の割り込み優先順位を逆転させます。

OERエラーもしくは、正常受信の場合は、U0RBからデータを受信し、バルク IN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、シリアル入力転送を禁止します。

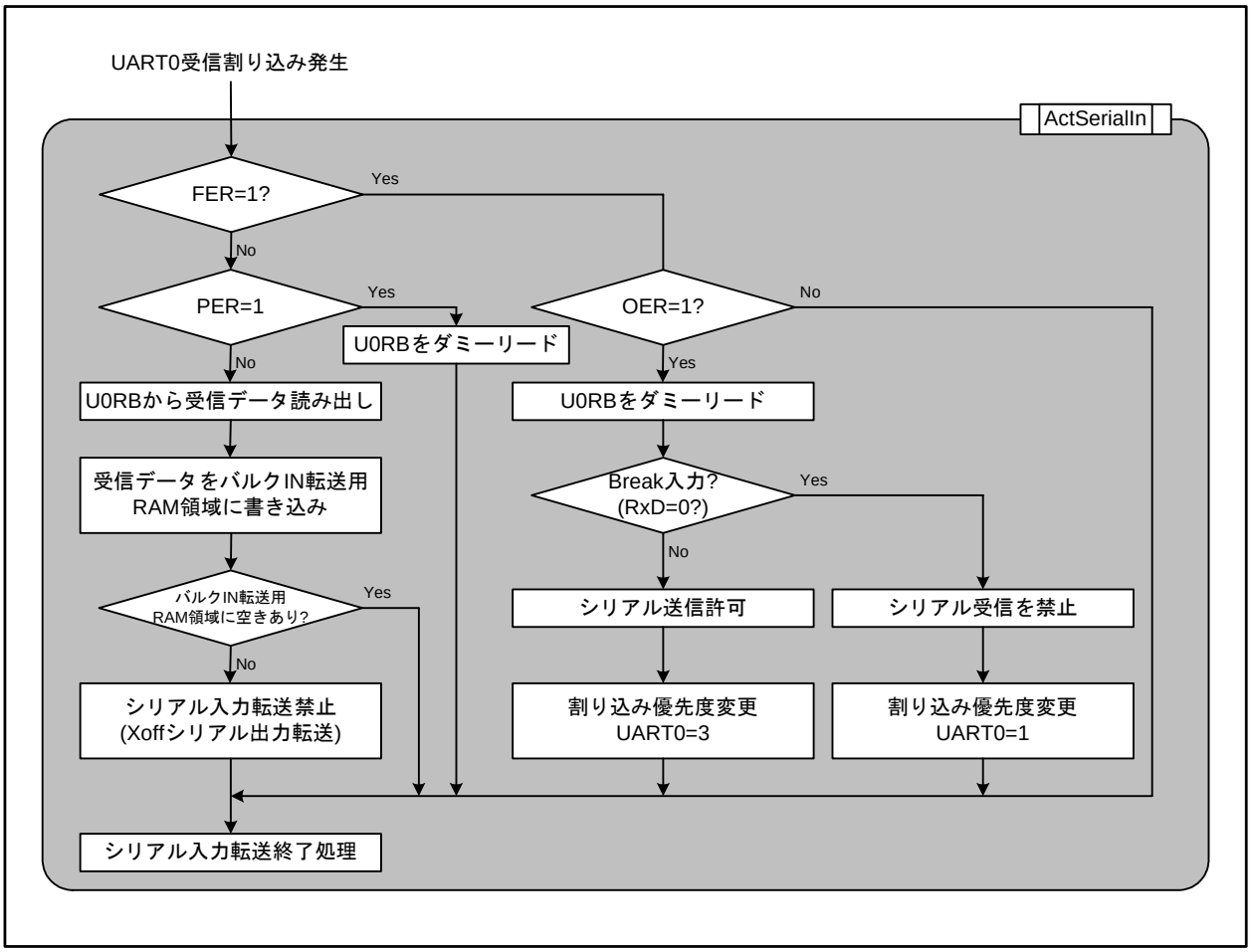


図 4.18 シリアル入力転送

4.5 サンプルプログラムの動作

この章では、M16C/6C内蔵USBファンクションモジュールを使用して、ヒロテック社製USBプロトコルアナライザ「HUSB200」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」を例に説明します。

なお、各パケットの前部にある「NO.」は測定時のトランザクションおよび、パケット通し番号です。

4.5.1 デバイス接続時のコントロール転送

以下に示す図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。

5.　アナライザのデータ

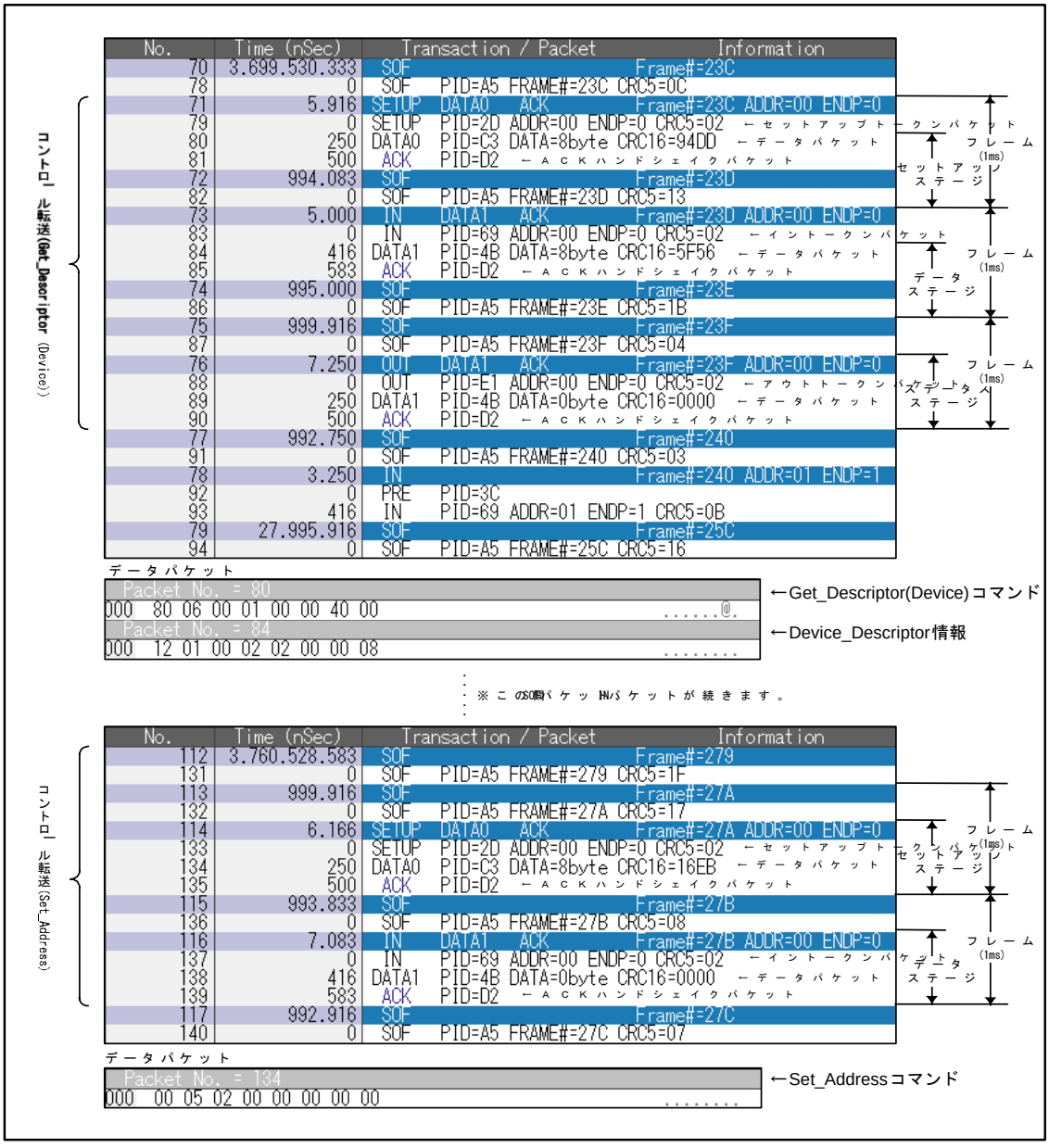
この章では、M16C/6C内蔵USBファンクションモジュールを使用して、ヒロテック社製USBプロトコルアナライザ「HUSB200」を用いた測定を行い、実際にバスを流れているデータについて「デバイス接続時のコントロール転送」を例に説明します。

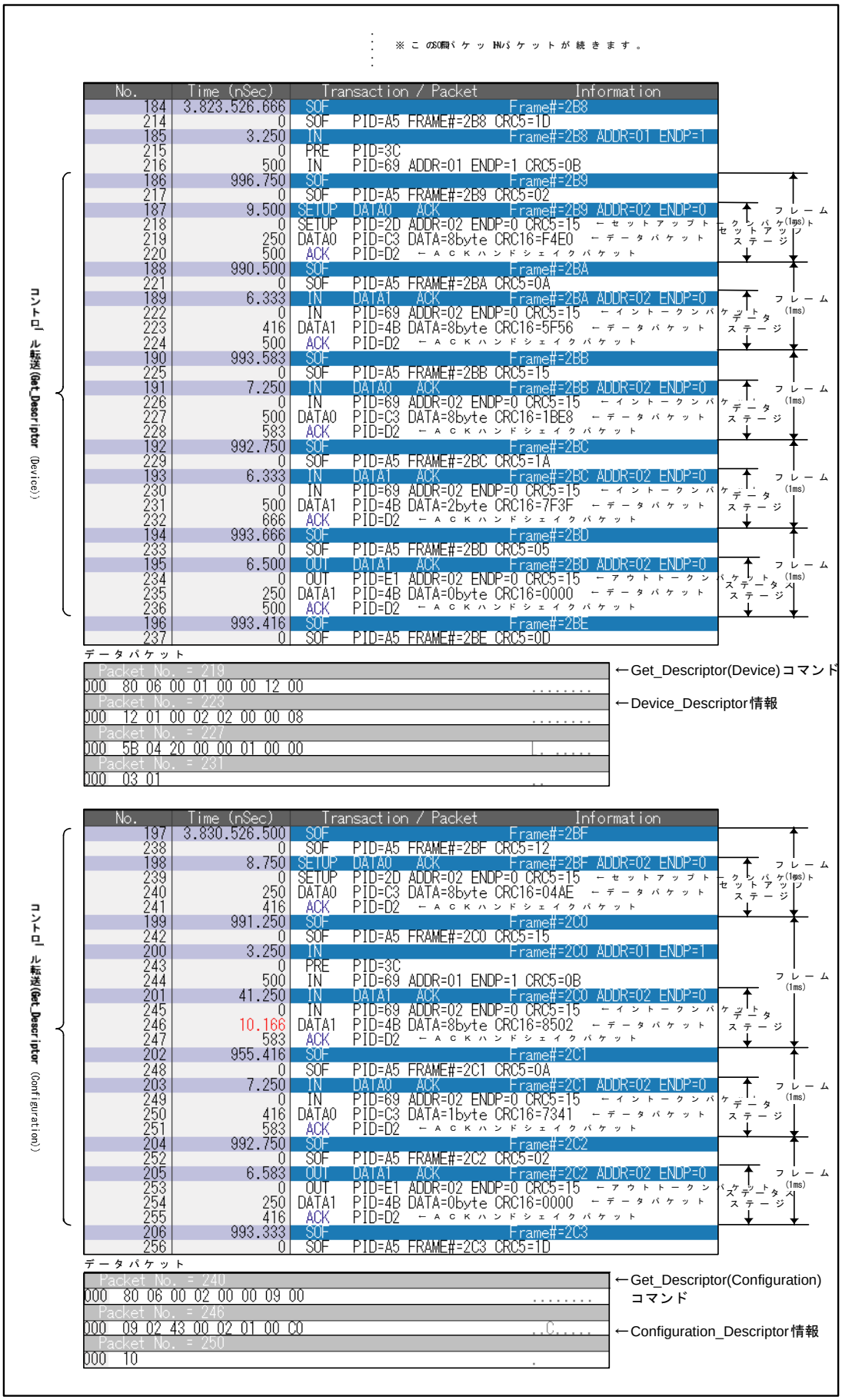
注1. 各パケットの前部にある「NO.」は測定時のトランザクションおよび、パケット通し番号です。

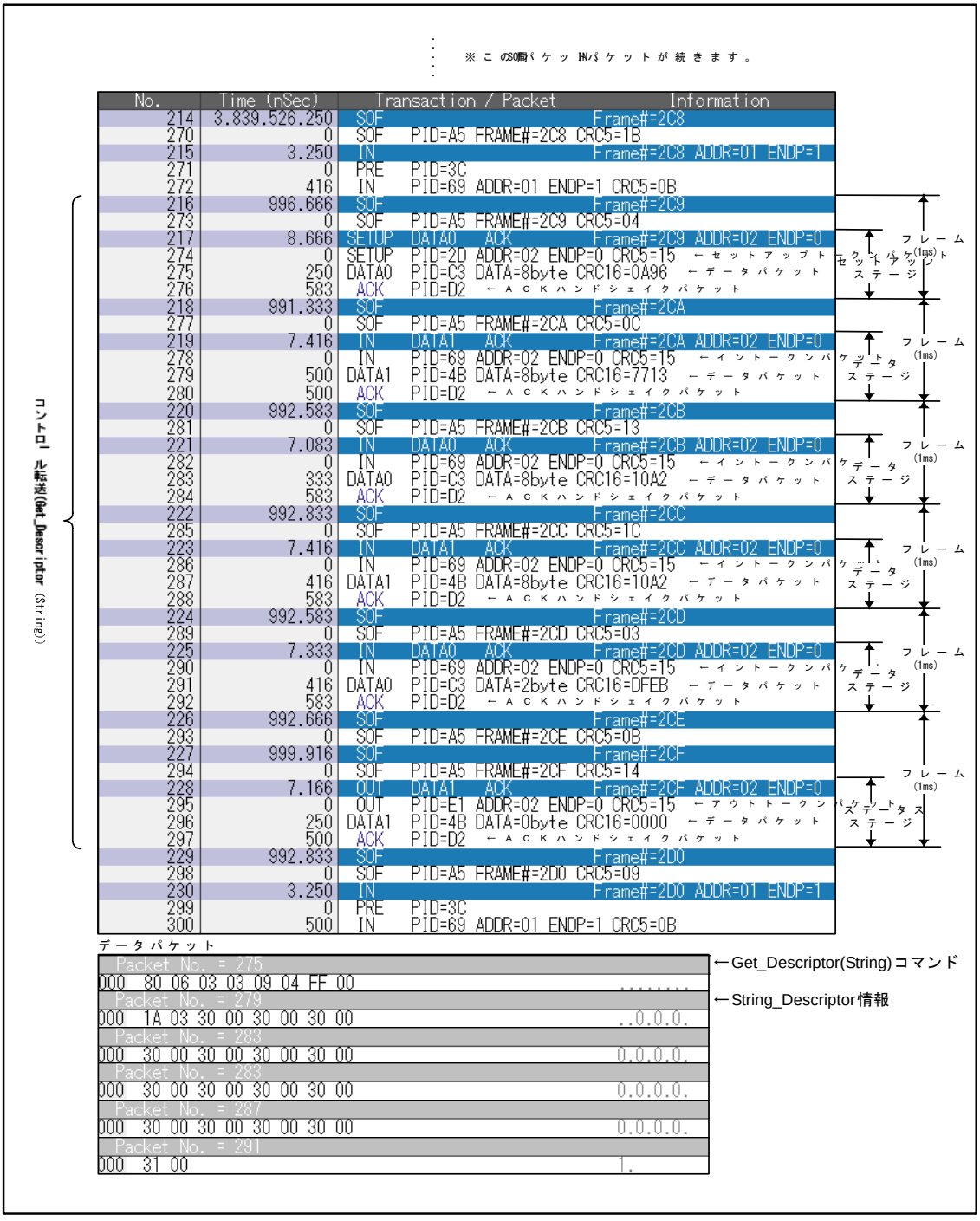
5.1　デバイス接続時のコントロール転送

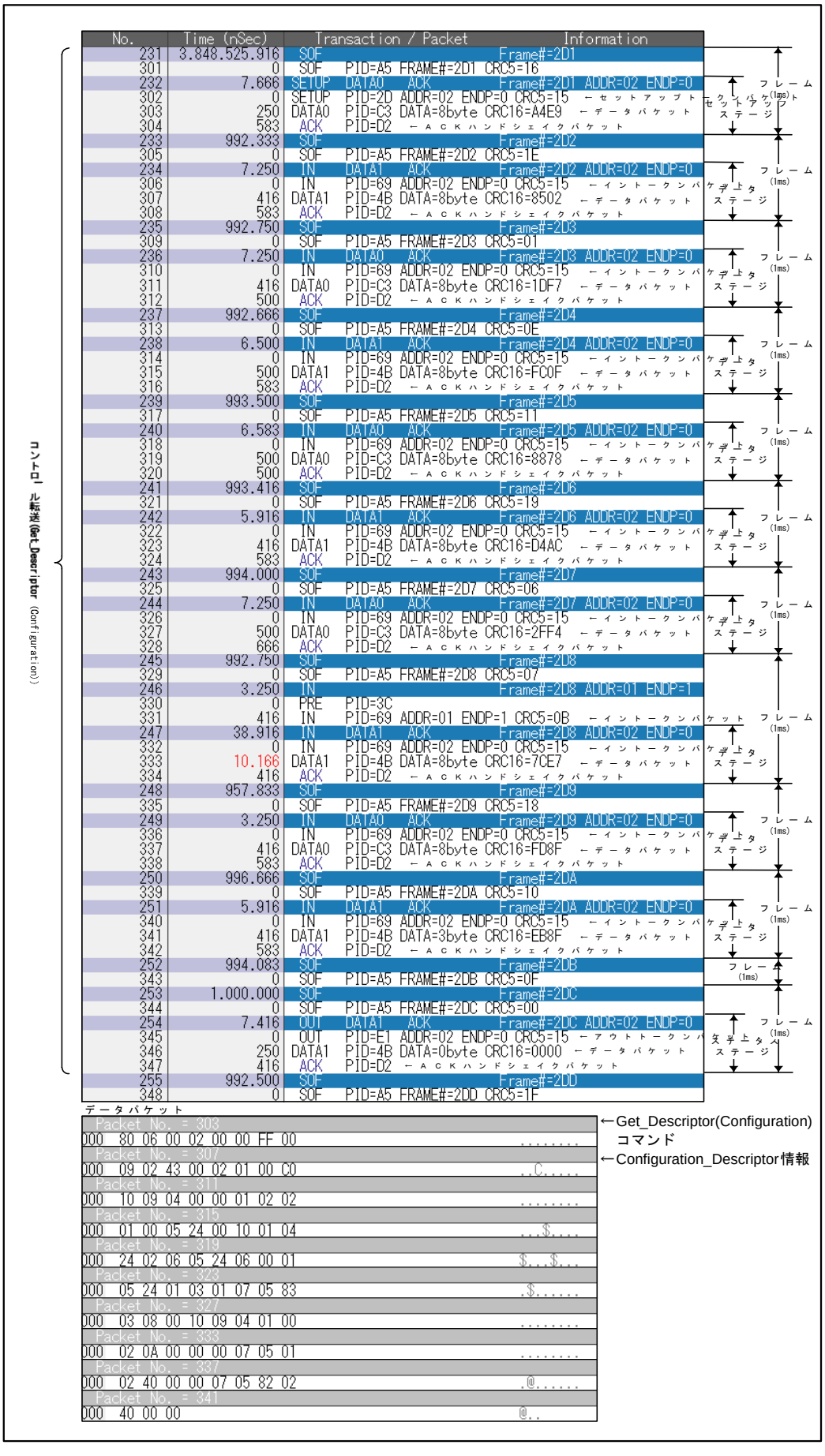
以下に示す図5.1は本デバイスをホストコントローラに接続し、Vbusに電源が供給されている（電源投入ステート）状態から、デバイスが使用可能になる状態（構成ステート）に至るまでを測定したものです。

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。




※ この50個パケット IN パケットが続きます。
コントロール転送(Get_Descriptor (Device))
No. Time (nSec) Transaction / Packet Information
184 3.823.526.666 SOF Frame#=2B8
214 0 SOF PID=A5 FRAME#=2B8 CRC5=1D
185 3.250 IN Frame#=2B8 ADDR=01 ENDP=1
215 0 PRE PID=3C
216 500 IN PID=69 ADDR=01 ENDP=1 CRC5=0B
186 996.750 SOF Frame#=2B9
217 0 SOF PID=A5 FRAME#=2B9 CRC5=02
187 9.500 SETUP DATA0 ACK Frame#=2B9 ADDR=02 ENDP=0
218 0 SETUP PID=2D ADDR=02 ENDP=0 CRC5=15 ←セットアップトークンパケット
219 250 DATA0 PID=C3 DATA=8byte CRC16=F4E0 ←データパケット
220 500 ACK PID=D2 ←ACKハンドシェイクパケット
188 990.500 SOF Frame#=2BA
221 0 SOF PID=A5 FRAME#=2BA CRC5=0A
189 6.333 IN DATA1 ACK Frame#=2BA ADDR=02 ENDP=0
222 0 IN PID=69 ADDR=02 ENDP=0 CRC5=15 ←インタークンパケット
223 416 DATA1 PID=4B DATA=8byte CRC16=5F56 ←データパケット
224 500 ACK PID=D2 ←ACKハンドシェイクパケット
190 993.583 SOF Frame#=2BB
225 0 SOF PID=A5 FRAME#=2BB CRC5=15
191 7.250 IN DATA0 ACK Frame#=2BB ADDR=02 ENDP=0
226 0 IN PID=69 ADDR=02 ENDP=0 CRC5=15 ←インタークンパケット
227 500 DATA0 PID=C3 DATA=8byte CRC16=1BE8 ←データパケット
228 583 ACK PID=D2 ←ACKハンドシェイクパケット
192 992.750 SOF Frame#=2BC
229 0 SOF PID=A5 FRAME#=2BC CRC5=1A
193 6.333 IN DATA1 ACK Frame#=2BC ADDR=02 ENDP=0
230 0 IN PID=69 ADDR=02 ENDP=0 CRC5=15 ←インタークンパケット
231 500 DATA1 PID=4B DATA=2byte CRC16=7F3F ←データパケット
232 666 ACK PID=D2 ←ACKハンドシェイクパケット
194 993.666 SOF Frame#=2BD
233 0 SOF PID=A5 FRAME#=2BD CRC5=05
195 6.500 OUT DATA1 ACK Frame#=2BD ADDR=02 ENDP=0
234 0 OUT PID=E1 ADDR=02 ENDP=0 CRC5=15 ←アウトトークンパケット
235 250 DATA1 PID=4B DATA=0byte CRC16=0000 ←データパケット
236 500 ACK PID=D2 ←ACKハンドシェイクパケット
196 993.416 SOF Frame#=2BE
237 0 SOF PID=A5 FRAME#=2BE CRC5=0D
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
フレーム(1ms)
データパケット
Packet No. = 219
000 80 06 00 01 00 00 12 00 ....
←Get_Descriptor(Device)コマンド
Packet No. = 223
000 12 01 00 02 02 00 00 08 ....
←Device_Descriptor情報
Packet No. = 227
000 5B 04 20 00 00 01 00 00 [. .....
Packet No. = 231
000 03 01 ..
コントロール転送(Get_Descriptor (Configuration))
No. Time (nSec) Transaction / Packet Information
197 3.830.526.500 SOF Frame#=2BF
238 0 SOF PID=A5 FRAME#=2BF CRC5=12
198 8.750 SETUP DATA0 ACK Frame#=2BF ADDR=02 ENDP=0
239 0 SETUP PID=2D ADDR=02 ENDP=0 CRC5=15 ←セットアップトークンパケット
240 250 DATA0 PID=C3 DATA=8byte CRC16=04AE ←データパケット
241 416 ACK PID=D2 ←ACKハンドシェイクパケット
199 991.250 SOF Frame#=2C0
242 0 SOF PID=A5 FRAME#=2C0 CRC5=15
200 3.250 IN Frame#=2C0 ADDR=01 ENDP=1
243 0 PRE PID=3C
244 500 IN PID=69 ADDR=01 ENDP=1 CRC5=0B
201 41.250 IN DATA1 ACK Frame#=2C0 ADDR=02 ENDP=0
245 0 IN PID=69 ADDR=02 ENDP=0 CRC5=15 ←インタークンパケット
246 10.166 DATA1 PID=4B DATA=8byte CRC16=8502 ←データパケット
247 583 ACK PID=D2 ←ACKハンドシェイクパケット
202 955.416 SOF Frame#=2C1
248 0 SOF PID=A5 FRAME#=2C1 CRC5=0A
203 7.250 IN DATA0 ACK Frame#=2C1 ADDR=02 ENDP=0
249 0 IN PID=69 ADDR=02 ENDP=0 CRC5=15 ←インタークンパケット
250 416 DATA0 PID=C3 DATA=1byte CRC16=7341 ←データパケット
251 583 ACK PID=D2 ←ACKハンドシェイクパケット
204 992.750 SOF Frame#=2C2
252 0 SOF PID=A5 FRAME#=2C2 CRC5=02
205 6.583 OUT DATA1 ACK Frame#=2C2 ADDR=02 ENDP=0
253 0 OUT PID=E1 ADDR=02 ENDP=0 CRC5=15 ←アウトトークンパケット
254 250 DATA1 PID=4B DATA=0byte CRC16=0000 ←データパケット
255 416 ACK PID=D2 ←ACKハンドシェイクパケット
206 993.333 SOF Frame#=2C3
256 0 SOF PID=A5 FRAME#=2C3 CRC5=1D
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
フレーム(1ms)
データパケット
Packet No. = 240
000 80 06 00 02 00 00 09 00 ....
←Get_Descriptor(Configuration)コマンド
Packet No. = 246
000 09 02 43 00 02 01 00 C0 ..C.....
←Configuration_Descriptor情報
Packet No. = 250
000 10 .

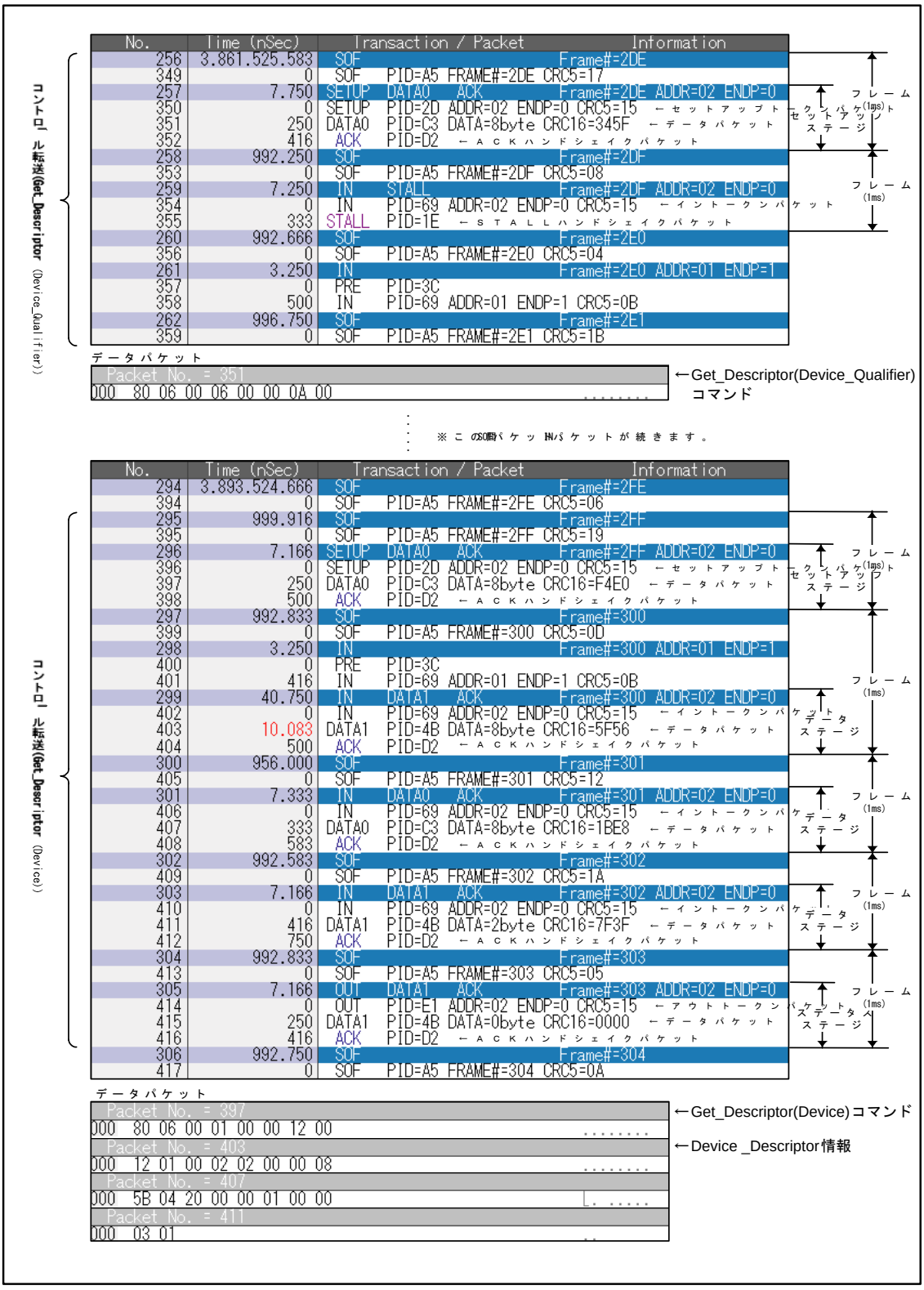

※このSOF間パケットINパケットが続きます。
No. Time (nSec) Transaction / Packet Information
214 3,839,526,250 SOF Frame#=2C8
270 0 SOF PID=A5 FRAME#=2C8 CRC5=1B
215 3,250 IN Frame#=2C8 ADDR=01 ENDP=1
271 0 PRE PID=3C
272 416 IN PID=69 ADDR=01 ENDP=1 CRC5=0B
216 996,666 SOF Frame#=2C9
273 0 SOF PID=A5 FRAME#=2C9 CRC5=04
217 8,666 SETUP DATA0 ACK Frame#=2C9 ADDR=02 ENDP=0
274 0 SETUP PID=2D ADDR=02 ENDP=0 CRC5=15 ←セットアップトークンパケット
275 250 DATA0 PID=C3 DATA=8byte CRC16=0A96 ←データパケット
276 583 ACK PID=D2 ←ACKハンドシェイクパケット
218 991,333 SOF Frame#=2CA
277 0 SOF PID=A5 FRAME#=2CA CRC5=0C
219 7,416 IN DATA1 ACK Frame#=2CA ADDR=02 ENDP=0
278 0 IN PID=69 ADDR=02 ENDP=0 CRC5=15 ←インタークンパケット
279 500 DATA1 PID=4B DATA=8byte CRC16=7713 ←データパケット
280 500 ACK PID=D2 ←ACKハンドシェイクパケット
220 992,583 SOF Frame#=2CB
281 0 SOF PID=A5 FRAME#=2CB CRC5=13
221 7,083 IN DATA0 ACK Frame#=2CB ADDR=02 ENDP=0
282 0 IN PID=69 ADDR=02 ENDP=0 CRC5=15 ←インタークンパケット
283 333 DATA0 PID=C3 DATA=8byte CRC16=10A2 ←データパケット
284 583 ACK PID=D2 ←ACKハンドシェイクパケット
222 992,833 SOF Frame#=2CC
285 0 SOF PID=A5 FRAME#=2CC CRC5=1C
223 7,416 IN DATA1 ACK Frame#=2CC ADDR=02 ENDP=0
286 0 IN PID=69 ADDR=02 ENDP=0 CRC5=15 ←インタークンパケット
287 416 DATA1 PID=4B DATA=8byte CRC16=10A2 ←データパケット
288 583 ACK PID=D2 ←ACKハンドシェイクパケット
224 992,583 SOF Frame#=2CD
289 0 SOF PID=A5 FRAME#=2CD CRC5=03
225 7,333 IN DATA0 ACK Frame#=2CD ADDR=02 ENDP=0
290 0 IN PID=69 ADDR=02 ENDP=0 CRC5=15 ←インタークンパケット
291 416 DATA0 PID=C3 DATA=2byte CRC16=DFEB ←データパケット
292 583 ACK PID=D2 ←ACKハンドシェイクパケット
226 992,666 SOF Frame#=2CE
293 0 SOF PID=A5 FRAME#=2CE CRC5=0B
227 999,916 SOF Frame#=2CF
294 0 SOF PID=A5 FRAME#=2CF CRC5=14
228 7,166 OUT DATA1 ACK Frame#=2CF ADDR=02 ENDP=0
295 0 OUT PID=E1 ADDR=02 ENDP=0 CRC5=15 ←アウトトークンパケット
296 250 DATA1 PID=4B DATA=0byte CRC16=0000 ←データパケット
297 500 ACK PID=D2 ←ACKハンドシェイクパケット
229 992,833 SOF Frame#=2D0
298 0 SOF PID=A5 FRAME#=2D0 CRC5=09
230 3,250 IN Frame#=2D0 ADDR=01 ENDP=1
299 0 PRE PID=3C
300 500 IN PID=69 ADDR=01 ENDP=1 CRC5=0B
コントロール転送(Get_Descriptor (String))
フレーム (1ms) セットアップステージ
フレーム (1ms) データステージ
フレーム (1ms) データステージ
フレーム (1ms) データステージ
フレーム (1ms) データステージ
フレーム (1ms) ステータスステージ
データパケット
Packet No. = 275
000 80 06 03 03 09 04 FF 00 ....
←Get_Descriptor(String)コマンド
Packet No. = 279
000 1A 03 30 00 30 00 30 00 ..0.0.0.
←String_Descriptor情報
Packet No. = 283
000 30 00 30 00 30 00 30 00 0.0.0.0.
Packet No. = 283
000 30 00 30 00 30 00 30 00 0.0.0.0.
Packet No. = 287
000 30 00 30 00 30 00 30 00 0.0.0.0.
Packet No. = 291
000 31 00 1.

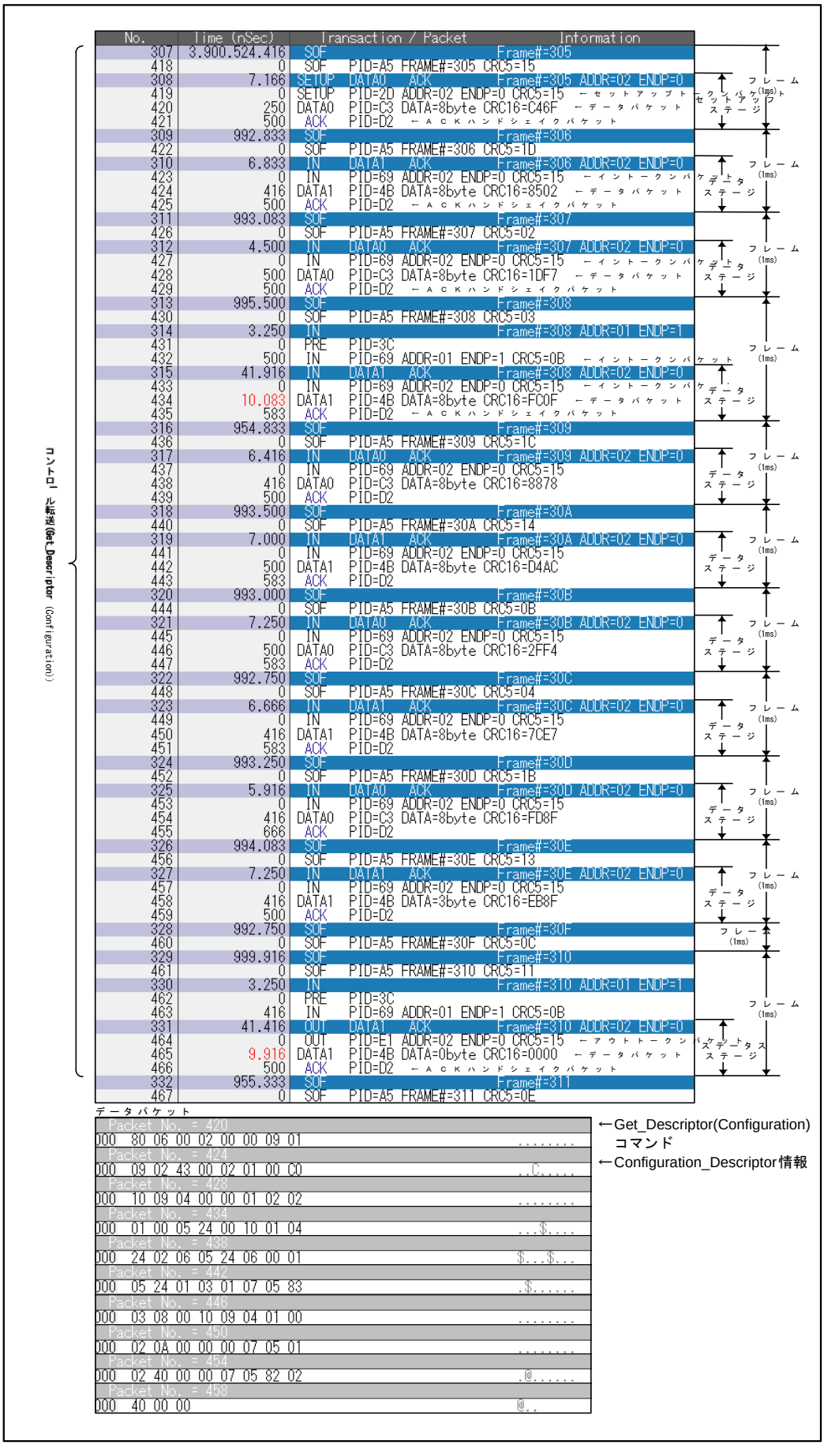


No.	Time (nSec)	Transaction / Packet	Information
231	3,848,525,916	SOF	Frame#=2D1
301	0	SOF PID=A5 FRAME#=2D1 CRC5=16	
232	7,666	SETUP DATA0 ACK	Frame#=2D1 ADDR=02 ENDP=0
302	0	SETUP PID=2D ADDR=02 ENDP=0 CRC5=15	←セットアップトークンパケット
303	250	DATA0 PID=C3 DATA=8byte CRC16=A4E9	←データパケット
304	583	ACK PID=D2	←ACKハンドシェイクパケット
233	992,333	SOF	Frame#=2D2
305	0	SOF PID=A5 FRAME#=2D2 CRC5=1E	
234	7,250	IN DATA1 ACK	Frame#=2D2 ADDR=02 ENDP=0
306	0	IN PID=69 ADDR=02 ENDP=0 CRC5=15	←イントークンパケット
307	416	DATA1 PID=4B DATA=8byte CRC16=8502	←データパケット
308	583	ACK PID=D2	←ACKハンドシェイクパケット
235	992,750	SOF	Frame#=2D3
309	0	SOF PID=A5 FRAME#=2D3 CRC5=01	
236	7,250	IN DATA0 ACK	Frame#=2D3 ADDR=02 ENDP=0
310	0	IN PID=69 ADDR=02 ENDP=0 CRC5=15	←イントークンパケット
311	416	DATA0 PID=C3 DATA=8byte CRC16=1DF7	←データパケット
312	500	ACK PID=D2	←ACKハンドシェイクパケット
237	992,666	SOF	Frame#=2D4
313	0	SOF PID=A5 FRAME#=2D4 CRC5=0E	
238	6,500	IN DATA1 ACK	Frame#=2D4 ADDR=02 ENDP=0
314	0	IN PID=69 ADDR=02 ENDP=0 CRC5=15	←イントークンパケット
315	500	DATA1 PID=4B DATA=8byte CRC16=FC0F	←データパケット
316	583	ACK PID=D2	←ACKハンドシェイクパケット
239	993,500	SOF	Frame#=2D5
317	0	SOF PID=A5 FRAME#=2D5 CRC5=11	
240	6,583	IN DATA0 ACK	Frame#=2D5 ADDR=02 ENDP=0
318	0	IN PID=69 ADDR=02 ENDP=0 CRC5=15	←イントークンパケット
319	500	DATA0 PID=C3 DATA=8byte CRC16=8878	←データパケット
320	500	ACK PID=D2	←ACKハンドシェイクパケット
241	993,416	SOF	Frame#=2D6
321	0	SOF PID=A5 FRAME#=2D6 CRC5=19	
242	5,916	IN DATA1 ACK	Frame#=2D6 ADDR=02 ENDP=0
322	0	IN PID=69 ADDR=02 ENDP=0 CRC5=15	←イントークンパケット
323	416	DATA1 PID=4B DATA=8byte CRC16=D4AC	←データパケット
324	583	ACK PID=D2	←ACKハンドシェイクパケット
243	994,000	SOF	Frame#=2D7
325	0	SOF PID=A5 FRAME#=2D7 CRC5=06	
244	7,250	IN DATA0 ACK	Frame#=2D7 ADDR=02 ENDP=0
326	0	IN PID=69 ADDR=02 ENDP=0 CRC5=15	←イントークンパケット
327	500	DATA0 PID=C3 DATA=8byte CRC16=2FF4	←データパケット
328	666	ACK PID=D2	←ACKハンドシェイクパケット
245	992,750	SOF	Frame#=2D8
329	0	SOF PID=A5 FRAME#=2D8 CRC5=07	
246	3,250	IN	Frame#=2D8 ADDR=01 ENDP=1
330	0	PRE PID=3C	
331	416	IN PID=69 ADDR=01 ENDP=1 CRC5=0B	←イントークンパケット
247	38,916	IN DATA1 ACK	Frame#=2D8 ADDR=02 ENDP=0
332	0	IN PID=69 ADDR=02 ENDP=0 CRC5=15	←イントークンパケット
333	10,166	DATA1 PID=4B DATA=8byte CRC16=7CE7	←データパケット
334	416	ACK PID=D2	←ACKハンドシェイクパケット
248	957,833	SOF	Frame#=2D9
335	0	SOF PID=A5 FRAME#=2D9 CRC5=18	
249	3,250	IN DATA0 ACK	Frame#=2D9 ADDR=02 ENDP=0
336	0	IN PID=69 ADDR=02 ENDP=0 CRC5=15	←イントークンパケット
337	416	DATA0 PID=C3 DATA=8byte CRC16=FD8F	←データパケット
338	583	ACK PID=D2	←ACKハンドシェイクパケット
250	996,666	SOF	Frame#=2DA
339	0	SOF PID=A5 FRAME#=2DA CRC5=10	
251	5,916	IN DATA1 ACK	Frame#=2DA ADDR=02 ENDP=0
340	0	IN PID=69 ADDR=02 ENDP=0 CRC5=15	←イントークンパケット
341	416	DATA1 PID=4B DATA=3byte CRC16=EB8F	←データパケット
342	583	ACK PID=D2	←ACKハンドシェイクパケット
252	994,083	SOF	Frame#=2DB
343	0	SOF PID=A5 FRAME#=2DB CRC5=0F	
253	1,000,000	SOF	Frame#=2DC
344	0	SOF PID=A5 FRAME#=2DC CRC5=00	
254	7,416	OUT DATA1 ACK	Frame#=2DC ADDR=02 ENDP=0
345	0	OUT PID=E1 ADDR=02 ENDP=0 CRC5=15	←アウトトークンパケット
346	250	DATA1 PID=4B DATA=0byte CRC16=0000	←データパケット
347	416	ACK PID=D2	←ACKハンドシェイクパケット
255	992,500	SOF	Frame#=2DD
348	0	SOF PID=A5 FRAME#=2DD CRC5=1F	

データパケット

```
Packet No. = 303
000  80 06 00 02 00 00 FF 00                          ........
Packet No. = 307
000  09 02 43 00 02 01 00 C0                          ..C.....
Packet No. = 311
000  10 09 04 00 00 01 02 02                          ........
Packet No. = 315
000  01 00 05 24 00 10 01 04                          ...$....
Packet No. = 319
000  24 02 06 05 24 06 00 01                          $...$...
Packet No. = 323
000  05 24 01 03 01 07 05 83                          .$......
Packet No. = 327
000  03 08 00 10 09 04 01 00                          ........
Packet No. = 333
000  02 0A 00 00 00 07 05 01                          ........
Packet No. = 337
000  02 40 00 00 07 05 82 02                          .@......
Packet No. = 341
000  40 00 00                                         @..
```


コントロール転送(Get_Descriptor (Device_Qualifier))
No. Time (nSec) Transaction / Packet Information
256 3,861,525,583 SOF Frame#=2DE
349 0 SOF PID=A5 FRAME#=2DE CRC5=17
257 7,750 SETUP DATA0 ACK Frame#=2DE ADDR=02 ENDP=0
350 0 SETUP PID=2D ADDR=02 ENDP=0 CRC5=15 ←セットアップトークンパケット
351 250 DATA0 PID=C3 DATA=8byte CRC16=345F ←データパケット
352 416 ACK PID=D2 ←ACKハンドシェイクパケット
258 992,250 SOF Frame#=2DF
353 0 SOF PID=A5 FRAME#=2DF CRC5=08
259 7,250 IN STALL Frame#=2DF ADDR=02 ENDP=0
354 0 IN PID=69 ADDR=02 ENDP=0 CRC5=15 ←インタークンパケット
355 333 STALL PID=1E ←STALLハンドシェイクパケット
260 992,666 SOF Frame#=2E0
356 0 SOF PID=A5 FRAME#=2E0 CRC5=04
261 3,250 IN Frame#=2E0 ADDR=01 ENDP=1
357 0 PRE PID=3C
358 500 IN PID=69 ADDR=01 ENDP=1 CRC5=0B
262 996,750 SOF Frame#=2E1
359 0 SOF PID=A5 FRAME#=2E1 CRC5=1B
フレーム(1ms)
セットアップトークンパケット
セットアップステージ
フレーム(1ms)
データパケット
Packet No. = 351
000 80 06 00 06 00 00 0A 00 ....
←Get_Descriptor(Device_Qualifier)コマンド
※このSOFパケットINパケットが続きます。
コントロール転送(Get_Descriptor (Device))
No. Time (nSec) Transaction / Packet Information
294 3,893,524,666 SOF Frame#=2FE
394 0 SOF PID=A5 FRAME#=2FE CRC5=06
295 999,916 SOF Frame#=2FF
395 0 SOF PID=A5 FRAME#=2FF CRC5=19
296 7,166 SETUP DATA0 ACK Frame#=2FF ADDR=02 ENDP=0
396 0 SETUP PID=2D ADDR=02 ENDP=0 CRC5=15 ←セットアップトークンパケット
397 250 DATA0 PID=C3 DATA=8byte CRC16=F4E0 ←データパケット
398 500 ACK PID=D2 ←ACKハンドシェイクパケット
297 992,833 SOF Frame#=300
399 0 SOF PID=A5 FRAME#=300 CRC5=0D
298 3,250 IN Frame#=300 ADDR=01 ENDP=1
400 0 PRE PID=3C
401 416 IN PID=69 ADDR=01 ENDP=1 CRC5=0B
299 40,750 IN DATA1 ACK Frame#=300 ADDR=02 ENDP=0
402 0 IN PID=69 ADDR=02 ENDP=0 CRC5=15 ←インタークンパケット
403 10,083 DATA1 PID=4B DATA=8byte CRC16=5F56 ←データパケット
404 500 ACK PID=D2 ←ACKハンドシェイクパケット
300 956,000 SOF Frame#=301
405 0 SOF PID=A5 FRAME#=301 CRC5=12
301 7,333 IN DATA0 ACK Frame#=301 ADDR=02 ENDP=0
406 0 IN PID=69 ADDR=02 ENDP=0 CRC5=15 ←インタークンパケット
407 333 DATA0 PID=C3 DATA=8byte CRC16=1BE8 ←データパケット
408 583 ACK PID=D2 ←ACKハンドシェイクパケット
302 992,583 SOF Frame#=302
409 0 SOF PID=A5 FRAME#=302 CRC5=1A
303 7,166 IN DATA1 ACK Frame#=302 ADDR=02 ENDP=0
410 0 IN PID=69 ADDR=02 ENDP=0 CRC5=15 ←インタークンパケット
411 416 DATA1 PID=4B DATA=2byte CRC16=7F3F ←データパケット
412 750 ACK PID=D2 ←ACKハンドシェイクパケット
304 992,833 SOF Frame#=303
413 0 SOF PID=A5 FRAME#=303 CRC5=05
305 7,166 OUT DATA1 ACK Frame#=303 ADDR=02 ENDP=0
414 0 OUT PID=E1 ADDR=02 ENDP=0 CRC5=15 ←アウトトークンパケット
415 250 DATA1 PID=4B DATA=0byte CRC16=0000 ←データパケット
416 416 ACK PID=D2 ←ACKハンドシェイクパケット
306 992,750 SOF Frame#=304
417 0 SOF PID=A5 FRAME#=304 CRC5=0A
フレーム(1ms)
インタークンパケット
データステージ
フレーム(1ms)
フレーム(1ms)
アウトトークンパケット
ステータスステージ
フレーム(1ms)
データパケット
Packet No. = 397
000 80 06 00 01 00 00 12 00 ....
←Get_Descriptor(Device)コマンド
Packet No. = 403
000 12 01 00 02 02 00 00 08 ....
←Device _Descriptor情報
Packet No. = 407
000 5B 04 20 00 00 01 00 00 [. .....
Packet No. = 411
000 03 01 ..



No.	Time (nSec)	Transaction / Packet	Information
307	3.900.524.416	SOF	Frame#=305
418	0	SOF PID=A5	FRAME#=305 CRC5=15
308	7.166	SETUP DATA0 ACK	Frame#=305 ADDR=02 ENDP=0
419	0	SETUP PID=2D	ADDR=02 ENDP=0 CRC5=15 ←セットアップトークンパケット
420	250	DATA0 PID=C3	DATA=8byte CRC16=C46F ←データパケット
421	500	ACK PID=D2	←ACKハンドシェイクパケット
309	992.833	SOF	Frame#=306
422	0	SOF PID=A5	FRAME#=306 CRC5=1D
310	6.833	IN DATA1 ACK	Frame#=306 ADDR=02 ENDP=0
423	0	IN PID=69	ADDR=02 ENDP=0 CRC5=15 ←インタークンパケット
424	416	DATA1 PID=4B	DATA=8byte CRC16=8502 ←データパケット
425	500	ACK PID=D2	←ACKハンドシェイクパケット
311	993.083	SOF	Frame#=307
426	0	SOF PID=A5	FRAME#=307 CRC5=02
312	4.500	IN DATA0 ACK	Frame#=307 ADDR=02 ENDP=0
427	0	IN PID=69	ADDR=02 ENDP=0 CRC5=15 ←インタークンパケット
428	500	DATA0 PID=C3	DATA=8byte CRC16=1DF7 ←データパケット
429	500	ACK PID=D2	←ACKハンドシェイクパケット
313	995.500	SOF	Frame#=308
430	0	SOF PID=A5	FRAME#=308 CRC5=03
314	3.250	IN	Frame#=308 ADDR=01 ENDP=1
431	0	PRE PID=3C	
432	500	IN PID=69	ADDR=01 ENDP=1 CRC5=0B ←インタークンパケット
315	41.916	IN DATA1 ACK	Frame#=308 ADDR=02 ENDP=0
433	0	IN PID=69	ADDR=02 ENDP=0 CRC5=15 ←インタークンパケット
434	10.083	DATA1 PID=4B	DATA=8byte CRC16=FC0F ←データパケット
435	583	ACK PID=D2	←ACKハンドシェイクパケット
316	954.833	SOF	Frame#=309
436	0	SOF PID=A5	FRAME#=309 CRC5=1C
317	6.416	IN DATA0 ACK	Frame#=309 ADDR=02 ENDP=0
437	0	IN PID=69	ADDR=02 ENDP=0 CRC5=15
438	416	DATA0 PID=C3	DATA=8byte CRC16=8878
439	500	ACK PID=D2	
318	993.500	SOF	Frame#=30A
440	0	SOF PID=A5	FRAME#=30A CRC5=14
319	7.000	IN DATA1 ACK	Frame#=30A ADDR=02 ENDP=0
441	0	IN PID=69	ADDR=02 ENDP=0 CRC5=15
442	500	DATA1 PID=4B	DATA=8byte CRC16=D4AC
443	583	ACK PID=D2	
320	993.000	SOF	Frame#=30B
444	0	SOF PID=A5	FRAME#=30B CRC5=0B
321	7.250	IN DATA0 ACK	Frame#=30B ADDR=02 ENDP=0
445	0	IN PID=69	ADDR=02 ENDP=0 CRC5=15
446	500	DATA0 PID=C3	DATA=8byte CRC16=2FF4
447	583	ACK PID=D2	
322	992.750	SOF	Frame#=30C
448	0	SOF PID=A5	FRAME#=30C CRC5=04
323	6.666	IN DATA1 ACK	Frame#=30C ADDR=02 ENDP=0
449	0	IN PID=69	ADDR=02 ENDP=0 CRC5=15
450	416	DATA1 PID=4B	DATA=8byte CRC16=7CE7
451	583	ACK PID=D2	
324	993.250	SOF	Frame#=30D
452	0	SOF PID=A5	FRAME#=30D CRC5=1B
325	5.916	IN DATA0 ACK	Frame#=30D ADDR=02 ENDP=0
453	0	IN PID=69	ADDR=02 ENDP=0 CRC5=15
454	416	DATA0 PID=C3	DATA=8byte CRC16=FD8F
455	666	ACK PID=D2	
326	994.083	SOF	Frame#=30E
456	0	SOF PID=A5	FRAME#=30E CRC5=13
327	7.250	IN DATA1 ACK	Frame#=30E ADDR=02 ENDP=0
457	0	IN PID=69	ADDR=02 ENDP=0 CRC5=15
458	416	DATA1 PID=4B	DATA=3byte CRC16=EB8F
459	500	ACK PID=D2	
328	992.750	SOF	Frame#=30F
460	0	SOF PID=A5	FRAME#=30F CRC5=0C
329	999.916	SOF	Frame#=310
461	0	SOF PID=A5	FRAME#=310 CRC5=11
330	3.250	IN	Frame#=310 ADDR=01 ENDP=1
462	0	PRE PID=3C	
463	416	IN PID=69	ADDR=01 ENDP=1 CRC5=0B
331	41.416	OUT DATA1 ACK	Frame#=310 ADDR=02 ENDP=0
464	0	OUT PID=E1	ADDR=02 ENDP=0 CRC5=15 ←アウトトークンパケット
465	9.916	DATA1 PID=4B	DATA=0byte CRC16=0000 ←データパケット
466	500	ACK PID=D2	←ACKハンドシェイクパケット
332	955.333	SOF	Frame#=311
467	0	SOF PID=A5	FRAME#=311 CRC5=0E

```
Packet No. = 420
000  80 06 00 02 00 00 09 01                          ........
Packet No. = 424
000  09 02 43 00 02 01 00 C0                          ..C.....
Packet No. = 428
000  10 09 04 00 00 01 02 02                          ........
Packet No. = 434
000  01 00 05 24 00 10 01 04                          ...$....
Packet No. = 438
000  24 02 06 05 24 06 00 01                          $...$...
Packet No. = 442
000  05 24 01 03 01 07 05 83                          .$......
Packet No. = 446
000  03 08 00 10 09 04 01 00                          ........
Packet No. = 450
000  02 0A 00 00 00 07 05 01                          ........
Packet No. = 454
000  02 40 00 00 07 05 82 02                          .@......
Packet No. = 458
000  40 00 00                                         @..
```

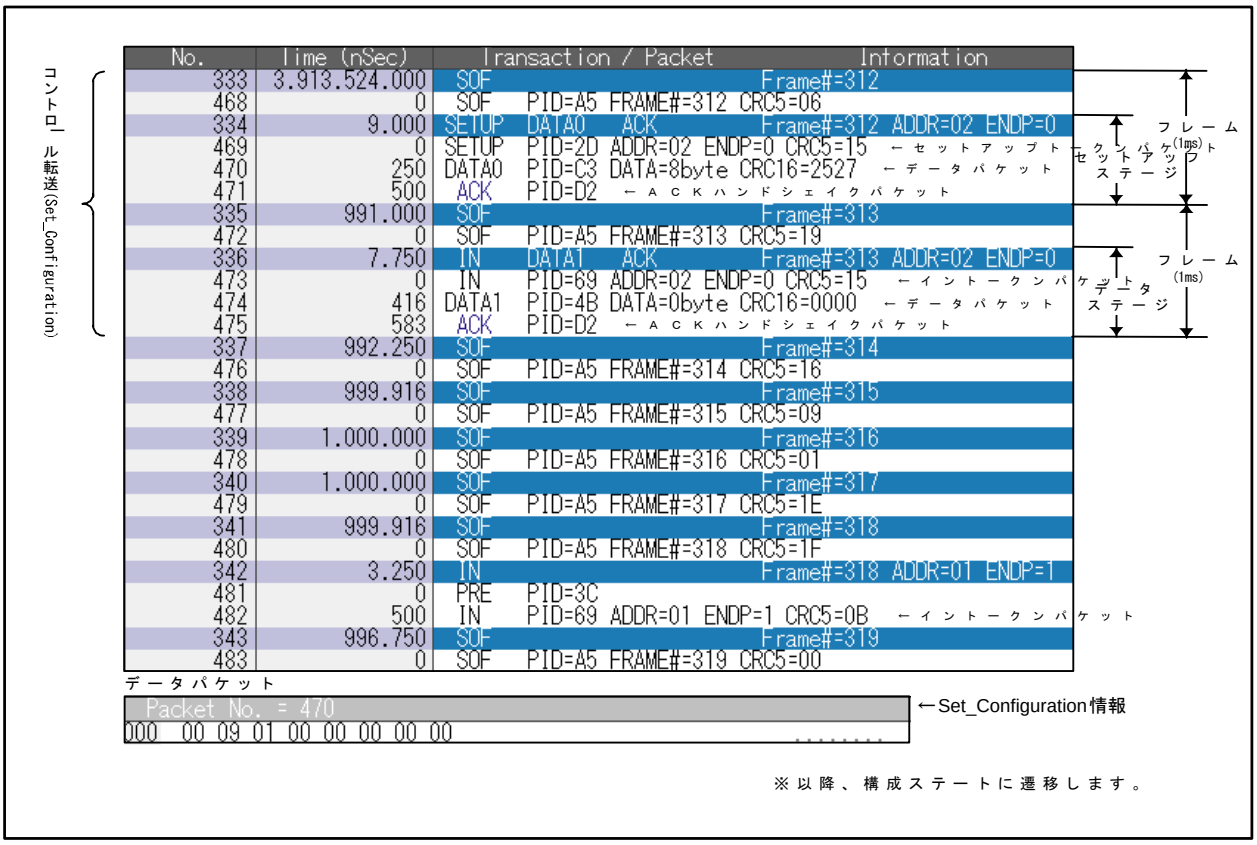


図 5.1　　デバイス接続時のコントロール転送

5. 参考ドキュメント

M16C/6C グループユーザーズマニュアル ハードウェア編
（最新版をルネサス エレクトロニクスホームページから入手してください。）

テクニカルアップデート/テクニカルニュース
(最新の情報をルネサス エレクトロニクスホームページから入手してください。)

C コンパイラマニュアル
M16C シリーズ, R8C ファミリ C コンパイラパッケージ　V.5.44
C コンパイラユーザーズマニュアル　Rev.1.00
(最新版をルネサス エレクトロニクスホームページから入手してください。)

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ
http://japan.renesas.com/

お問合せ先
http://japan.renesas.com/inquiry

改訂記録	M16C/6Cグループ USBシリアル変換

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2009.08.31	−	初版発行
1.01	2009.11.30	18	図3.1 状態遷移図　シリアル通信状態のシリアル出力をシリアル入力に変更
		21	表3.2 UsMain.c BranchOfInt関数を追加
		24	図4.1 メインルーチン　前回出力したパケットサイズを確認する経路を追加
		36	図4.13 ステータスステージ　一部修正
1.02	2010.01.20	24	図4.1 メインルーチン　一部修正
		36	図4.13 ステータスステージ　一部修正
1.03	2011.07.29	9	図2.7 USB_CommClass_INFフォルダ内ファイル　一部修正

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本文を参照してください。なお、本マニュアルの本文と異なる記載がある場合は、本文の記載が優先するものとします。

1. 未使用端子の処理
   【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。
   CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】電源投入時は，製品の状態は不定です。
   電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。
   外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。
   同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. リザーブアドレス（予約領域）のアクセス禁止
   【注意】リザーブアドレス（予約領域）のアクセスを禁止します。
   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
4. クロックについて
   【注意】リセット時は、クロックが安定した後、リセットを解除してください。
   プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。
   リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
5. 製品間の相違について
   【注意】型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。
   同じグループのマイコンでも型名が違うと、内部 ROM、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

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。

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口 http://www.renesas.com

※営業お問合せ窓口の住所・電話番号は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス販売株式会社　〒100-0004　千代田区大手町2-6-2（日本ビル） (03)5201-5307

■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：http://japan.renesas.com/inquiry