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

32

SH7727グループ USB ホストモジュール

アプリケーションノート（応用編）

ルネサス32ビットRISC マイクロコンピュータ
SuperH™ RISC engine ファミリ／ SH7700 シリーズ

ルネサス エレクトロニクス
www.renesas.com

Rev.1.00　2003.04

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、SH7727 内蔵の USB ファンクションモジュールと USB ホストモジュールを用いたファームウエ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および USB ホスト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並びに USB ホストモジュールの使用例として説明しているものであり、その内容を保証するものではありません。

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

【関連マニュアル】

- Universal Serial Bus Specification Revision 1.1
- Open Host Controller Interface Specification for USB Preliminary Release: 1.0a
- Open Universal Serial Bus Driver Interface (OpenUSBDI) Specification Revision 1.0
- Universal Serial Bus Mass Storage Class Specification Overview Revision 1.1
- Universal Serial Bus Mass Storage Class(Bulk-Only Transport) Revision 1.0
- Device Class Definition for Human Interface Devices (HID) Version 1.1
- SH7727 ハードウェアマニュアル
- SH7727 Solution Engine（MS7727SE01）取扱説明書
- SH7727 E10A エミュレータユーザーズマニュアル
- SH7727 USBホストモジュールアプリケーションノート

【ご注意】 **本アプリケーションノートに記載してあるサンプルプログラムでは、USB の転送タイプのうち「アイソクロナス」に関するファームウエアは準備しておりません。「アイソクロナス」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 SH7727、SH7727 Solution Engine のハードウェア仕様を記載してありますが、詳細は SH7727 のハードウェアマニュアル、ならびに SH7727 Solution Engine の取扱説明書を御覧ください。

目次

1. 概要

本アプリケーションノートは、SH7727 の USB ファンクションモジュールと USB ホスト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。

SH7727 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
- EXCPGによる内部システムクロック／外部入力（48MHz）を選択可能
- 低消費電力モードを搭載
- バストランシーバを内蔵

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	8 Byte	8 Byte	―
	EP0o	コントロールアウト	8 Byte	8 Byte	―
エンドポイント 1	EP1	バルクアウト	64 Byte	64×2（128 Byte）	可能
エンドポイント 2	EP2	バルクイン	64 Byte	64×2（128 Byte）	可能
エンドポイント 3	EP3	インタラプト	8 Byte	8 Byte	―

SH7727 内蔵 USB ホストモジュールの特長を以下に示します。

- USB1.1準拠
- Open Host Controller Interface（OHCI）1.0レジスタセット準拠
- ルートハブ機能内蔵
- Low-Speed（1.5Mbps）とFull-Speed（12Mbps）に対応
- 過電流検出機構に対応
- 最大127のEndpointに対応
- CPUに接続されたユーザメモリ領域を転送用データ、及びディスクリプタ用として利用可能

システム構成例を**図** 1.1 に示します。

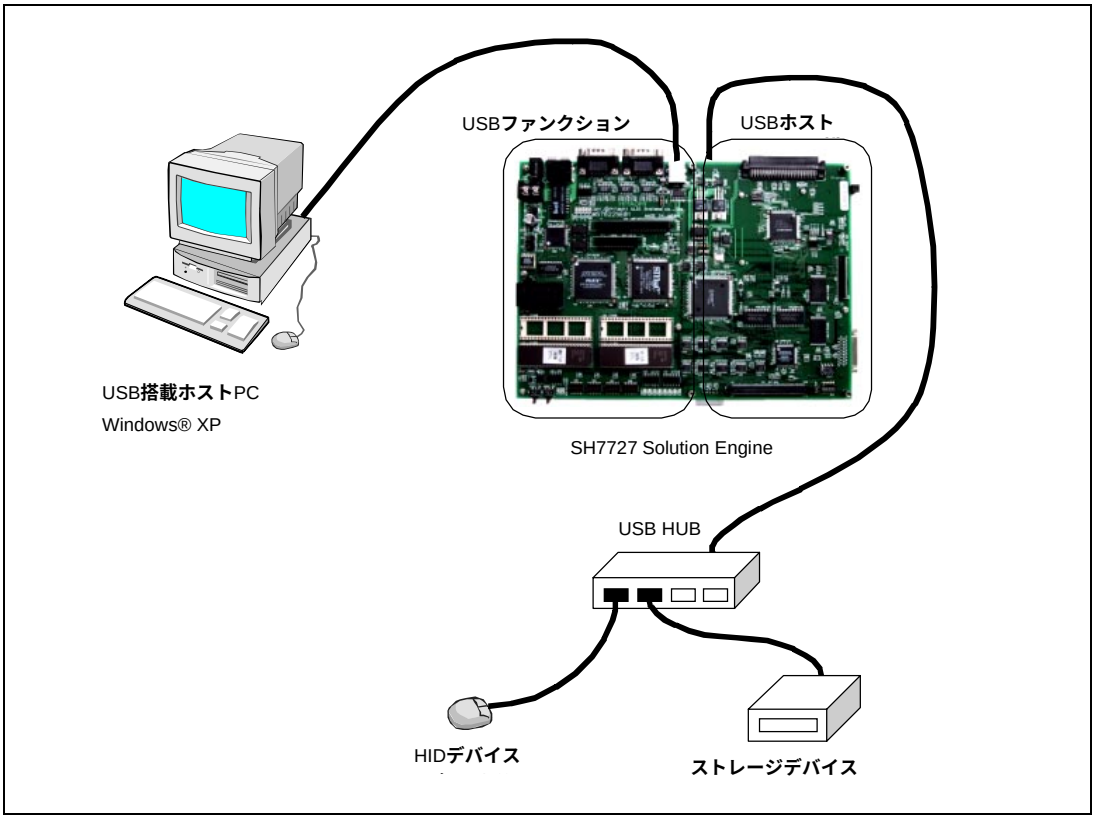


図 1.1　**システム構成例**

本システムは、SH7727 を搭載した日立超 LSI システムズ社製の SH7727 Solution Engine（以下 SH7727SE）、「Windows® XP」を搭載した PC（以下ホスト PC）、USB ハブ、USB ファンクションデバイス（USB Mass Storage Class、HID Class）によって構成されています。

本アプリケーションは、ホスト PC、SH7727SE、USB ハブ、USB ファンクションデバイスを USB で接続し、ホスト PC と USB ファンクションデバイス間でデータの保存、読み出しを行います。

SH7727SE に搭載している SH7727 には USB ホストモジュールを内蔵しており、SH7727SE が USB のホストとして、USB ハブとファンクションデバイスに対し制御を行います（以降 SH7727 ホストと呼びます）。

また、SH7727 は USB ファンクションモジュールも内蔵しており、ホスト PC に対しては SH7727SE がファンクションデバイスとして動作します（以降 SH7727 ファンクションと呼びます）。

USB ファンクションデバイスは、デバイス自身の種類・特性・属性を示す「ディスクリプタ」情報を持っています。SH7727SE は SH7727 ホストに接続された USB ファンクションデバイスのディスクリプタ情報を元にして、新たに複数のインタフェースを持つデバイスのディスクリプタ情報（マルチファンクションデバイス）を生成します。

その後 SH7727 ファンクションは SH7727 ホストにより生成されたディスクリプタ情報を使いホスト PC と接続

する事により、マルチファンクションデバイスとしてホスト PC とデータ転送を行います。

その結果 SH7727SE はホスト PC と USB ファンクションデバイスの中継ホストとなりホスト PC から USB ファンクションデバイスへデータの保存、読み出しができます。

本アプリケーションが動作する為に SH7727 内蔵の USB ホストモジュール、USB ファンクションモジュール、タイマを使用します。

また、上記 OS に標準で付属しているデバイスドライバを使用することが可能です。

本システムの特長を以下に示します。

1. サンプルプログラムにより、SH7727のUSBモジュール（ホスト、ファンクション）を短期間で評価可能。
2. サンプルプログラムにデバッグ支援機能を用意しました。ソースプログラムの設定によりホストモジュールで異常を検知するとシリアルを使用して異常内容の出力が可能です。
3. 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、インタラプト転送をサポート
4. E10A（PCカード型エミュレータ）を使用することができ、効率的なデバッグが可能です。

2. USB HUB の概要

この章では、USB HUB について説明します。

USB のホスト関連システムを開発する際にご参考として御使用ください。なお、規格の詳細については、

「Universal Serial Bus Specification Revision 1.1」

をご覧ください。

2.1 HUB の役割

USB HUB とは、ホストとファンクションの間に位置して接続ノード数を増やすと共に、USB の特徴である「データ転送方式の厳格な規定」「USB ポートから給電可能」「プラグ&プレイ」「機器（ファンクション）を最大 127 台まで接続可能」を提供する為に存在するデバイスで次に示す機能があります。

- 信号の伝達と遮断
- プラグ&プレイの実現
- 電源管理

これらの機能を使用する為にホストとハブは決められた方法で通信する必要があり、USB ハブクラスとして規格化されています。

【注】 USB **ホストは、**Device Descriptor **の** bDeviceClass **フィールドと** Interface Descriptor **の** bInterfaceClass **フィールド値が** 0x09 **であることを確認すると、接続されたデバイスがハブクラスであると認識します。**

2.2 Hub Descriptor

USB デバイスは、デバイス自身の種類・特性・属性を示す「ディスクリプタ」情報を持っています。

標準 USB デバイスの場合、「デバイス」「コンフィギュレーション」「インタフェース」「エンドポイント」の各ディスクリプタを持っていますが、ハブデバイスの場合これらに加えハブクラス専用のディスクリプタ情報（Hub Descriptor）を持っています。「Hub Descriptor」はクラスコマンドを使用して取得します。以下、**表** 2.1 にハブディスクリプタを示します。

表 2.1 ハブディスクリプタフォーマット

フィールド	サイズ(バイト)	内　　容
bDescLength	0x01	ディスクリプタのサイズ
bDescriptorType	0x01	Hub Descriptor タイプの値（0x29 で固定）
bNbrPorts	0x01	このハブがサポートするポート数
wHubCharacteristics	0x02	ハブの特徴 ビット 1〜0：ポート電力制御モード 00：ポートを一括制御 01：ポートを個別に制御 1X：リザーブ ビット 2：ハブの複合確認 0：ハブは複合的な装置の一部でない 1：ハブは複合的な装置の一部である ビット 4〜3：過度電流の保護モード 00：ポート全体で過度電流の保護 01：ポート個別で過度電流の保護 1X：過度電流の保護なし ビット 15〜5：予約
bPwrOn2PwrGood	0x01	ポートに給電を開始してからアクセスを開始するまでの待ち時間を表す。単位は 2ms を 1 とする。
bHubContrCurrent	0x01	ハブが必要とする最大電流値。単位は mA。
DeviceRemovable	0x01	ポートに取り外し可能なファンクションが有るかを示す。 ビット 0：予約 ビット 1〜n：ポート 1〜n を表す（n は最大 255） 0：取り外し可能 1：取り外し不可
PortPwrCtrlMask	ポート数	このフィールドは USB1.0 に対応したハブで使用していたが、USB1.1 では使用しない。USB1.1 では互換性のために存在する。

2.3 Hub Class-specific Request

クラスコマンドとは、USB の各クラス定義ごとに定められているコマンドです。クラスコマンドはコントロール転送を使用します。

USB HUB へ送る事が出来るクラスコマンドとしては 9 種類あります。**表** 2.2 にリクエストを示します。

表 2.2 **クラスコマンド一覧**

リクエスト	**コマンドの意味**	BmRequestType **フィールド値**	BRequest **フィールド値**
ClearHubFeature	**ハブステータスで報告された値をリセットする**	0x20	0x01
ClearPortFeature	**ポートステータスで報告された値をリセットする。**	0x23	0x01
GetBusState	**バス状態を返す**	0xA3	0x02
GetHubDescriptor	**ハブディスクリプタを取得する**	0xA0	0x06
GetHubStatus	**現在のハブ状態と前の承認から変化した状態を返す。**	0xA0	0x00
GetPortStatus	**現在のポート状態と前の承認から変化したポート状態を返す。**	0xA3	0x00
SetHubDescriptor	**ハブディスクリプタを上書きする**	0x20	0x07
SetHubFeature	**前のハブ状態からの変化を承認する**	0x20	0x03
SetPortFeature	**前のポート状態からの変化を承認する**	0x23	0x03

2.4 Host-Hub-Function **接続状態**

ハブは内蔵するポートを使い上流のホストと下流のデバイスを接続します。

ポートは上流のホスト側と接続するポートをアップストリームポートと言い、下流のデバイス側と接続するポートをダウンストリームポート（以下ポート）と言います。ハブはポートの制御を行い状態を取り纏めホストに通知します。Host-Hub-Function 接続を**図** 2.1 に示します。ここではハブの制御（ポート制御）に必要な事柄を説明します。

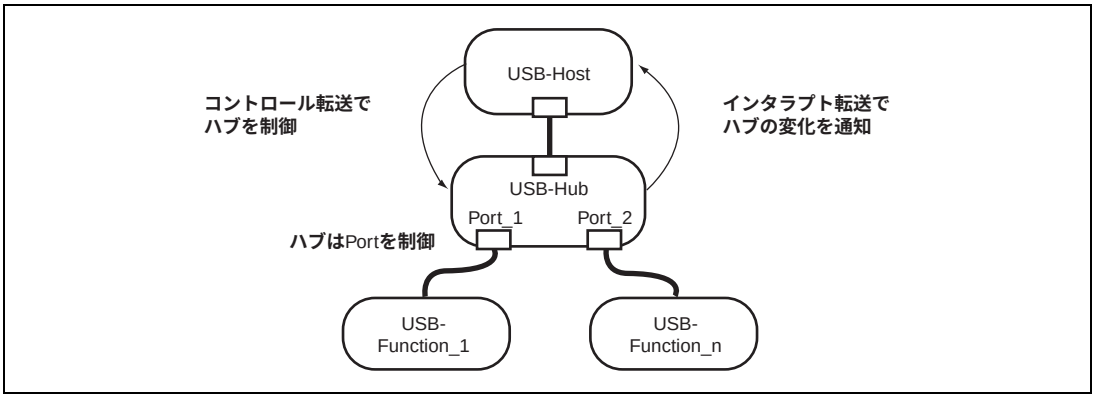


図 2.1 Host-Hub-Function **接続図**

2.5 ポートの状態遷移

ハブに内蔵されているポートは次に示す 7 つの状態「NotConfigured」、「Powered-off」、「Disconnected」、「Disabled」、「Resetting」、「Enabled」、「Suspended」があります。これらの状態は「ホストからの設定コマンド」、「規定時間」、「接続デバイスの動作」を要因として状態遷移します。

ポートの状態を設定するにはまずポートの状態を確認し、次にポートの状態を設定します。最後にポートの変化を承認することによりポートは状態遷移します。状態遷移を**図** 2.2 に示します。

ポートの状態を確認するには「GetPortStatus」を使用します。

ホストからの設定による遷移にはクラスコマンドの「SetPortFeature」を使用します。

ポート遷移の承認には「ClearPortFeature」を使用します。

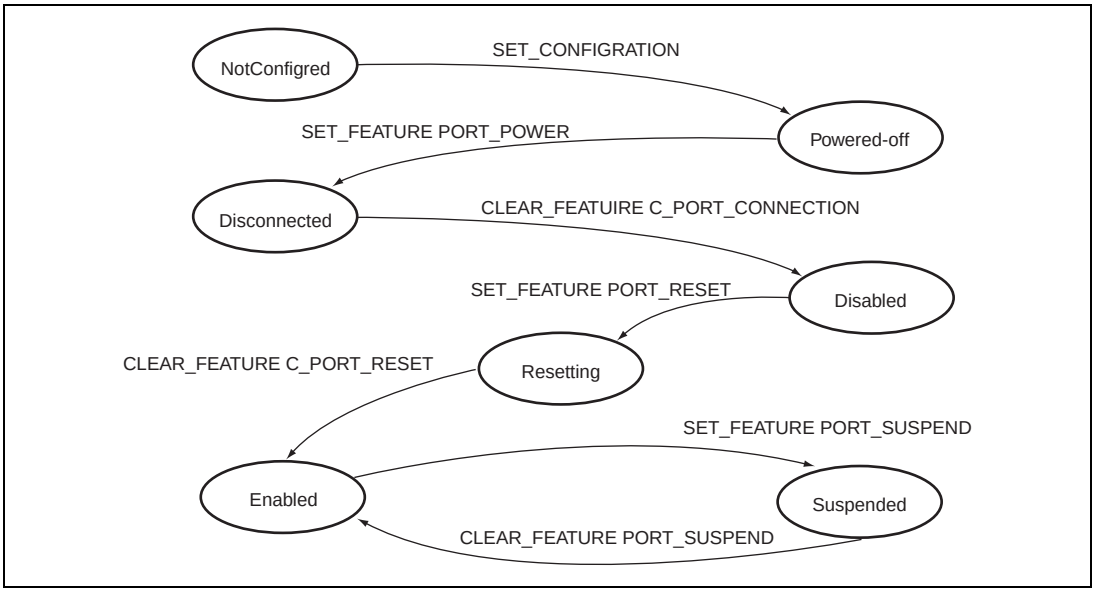


図 2.2 **ポート状態遷移図**

NotConfigured はハブが未構成で、ポートも不定状態です。

Powered-off はポートの給電が停止している状態です。

Disconnected はポートの給電が開始されているが、何も接続していない状態です。

Disabled はポートへファンクションが接続しているがバスリセット前です。

ポートは接続したファンクションへパケットを転送しません。

Resetting はバスリセット中の状態です。

Enabled はバスリセット後の状態で、ポートは接続したファンクション－ホスト間でパケットの転送を行います。

Suspended はサスペンドしている状態です。

2.5.1 ポートの状態を設定するには

ポートの状態を設定するにはまずポートの状態を確認し、次にポートの状態を設定します。

クラスコマンドの「SetPortFeature」と「ClearPortFeature」を使用します。SetPortFeature でポートに状態の設定を行い、ClearPortFeature でポートの変化を承認します。

SetPortFeature でポートの状態を設定した場合またはポートの変化通知があった場合「GetPortStatus」でポートの状態を確認します。ポートの変化は ClearPortFeature でポートの変化を承認することでポートの状態変化が確定します。

2.5.2 ポートの状態を知るには

ポートの状態を知るにはクラスコマンドの「GetPortStatus」を使用して現在の状態を知る事が出来ます。GetPortStatus コマンドの返値は 4 バイト長で返答され、フォーマットは**表** 2.3 に示します。

表 2.3 GetPortStatus **コマンド返答フォーマット**

Byte	Bit	内　　容
00-01	0	Current Connect Status：（PORT_CONNECTION） **現在このポートにファンクションが接続しているかどうかを表すビット。** 0：**ポートにファンクションは接続していない** 1：**ポートにファンクションが接続している**
	1	Port Enabled/Disabled：（PORT_ENABLE） **ポートのイネーブル／ディゼーブル状態を表すビット。** 0：**ポートはディゼーブル** 1：**ポートはイネーブル**
	2	Suspend：（PORT_SUSPEND） **ポートがサスペンド状態にあるかを表すビット。** 0：**ポートはサスペンドしていない** 1：**ポートがサスペンドしている**
	3	Over-current Indicator：（PORT_OVER_CURRENT） **ポートがオーバーカレント状態にあるかを表すビット。** 0：**ポートはオーバーカレントしていない** 1：**ポートがオーバーカレントしている**
	4	Reset：（PORT_RESET） **ポートがバスリセット状態にあるかを表すビット。** 0：**ポートはバスリセットしていない** 1：**ポートがバスリセットしている**

Byte	Bit	内　　容
00-01	5-7	Reserved
	8	Port Power：（PORT_POWER） ポートの電力供給状態を表すビット。 0：ポートは電力供給停止状態にある 1：ポートは電力供給状態にある
	9	Low Speed Device Attached:：（PORT_LOW_SPEED） ポートに接続されたファンクションはロースピードデバイスであることを表すビット。ポートにファンクションが接続されている時のみ有効。 0：接続されたファンクションはフルスピードデバイス 1：接続されたファンクションはロースピードデバイス
	10-15	Reserved
02-03	0	Connect Status Change：（C_PORT_CONNECTION） ポートの接続状態が変化したかを表すビット。 0：無変化 1：変化
	1	Port Enable/Disable Change：（C_PORT_ENABLE） ポートの状態が変化したかを表すビット。 0：無変化 1：変化
	2	Suspend Change：（C_PORT_SUSPEND） ポートのサスペンド状態が変化したかを表すビット。 0：無変化 1：変化
	3	Over-Current Indicator Change：（C_PORT_OVER_CURRENT） ポートのオーバーカレント状態が変化したかを表すビット。 0：無変化 1：変化
	4	Reset Change：（C_PORT_RESET） ポートのバスリセット状態が変化したかを表すビット。 0：無変化 1：変化
	5-15	Reserved

2.5.3 ポートからの給電について

ポートの電源供給はハブが構成（Configuration）した後、ホストからクラスコマンドを使い各ポートへ給電が指示され、各ポートは給電を開始します。これによりバスパワー動作のファンクションも電源を得る事が出来ます。ポートは供給電流の監視を行い、異常電流が流れた場合“ オーバーカレント ”を検出し給電を停止します。

ファンクション接続時の供給電流値としては 1 ポート当たり 100mA まで供給されます。ファンクションはハイパワーのバスパワーデバイスであっても、構成以前は 100mA で動作する必要があります。ファンクションのディスクリプタ情報で要求した電流値をホストが認め、ファンクションを構成した後に要求電流値まで使用する事が出来ます。

2.5.4 ポートのデバイス検知について

デバイスの接続・切断検知はポート毎に D+端子と D-端子の電位をチェックすることにより検知します。デバイスが未接続の場合、D+端子と D-端子は 15kΩのプルダウン抵抗により VIL（最大 0.8V）以下になります。ファンクションを接続するとファンクション内の D+端子又は D-端子に接続された 1.5 kΩのプルアップ抵抗により VIH（最低 2.0V）以上になりこの状態を 2.5 μs 以上検出することで接続を検知します。

接続を検知するとインタラプト転送を使用してホストに知らせます。インタラプト転送のパケットフォーマットはビットマップで、状態変化の発生したポートは１、変化のないポートには 0 を書き込みホストに知らせます。

Bit	n	…	2	1	0
変化発生箇所	**ポート n の変化**	…	**ポート 2 の変化**	**ポート 1 の変化**	**ハブの変化**

この際 D+端子がプルアップされていればフルスピードデバイス。D-端子がプルアップされていればロースピードデバイスであると認識します。

2.5.5 給電開始からパケット送信までについて

ポートへ給電を開始してから接続デバイスへ最初のパケットを送信するまでには、

幾つかの規定された時間がありそのタイミングに従って、パケットを送る必要があります。**図** 2.3 にハブ側から見た時間の規定とポートに起こる変化を示します。

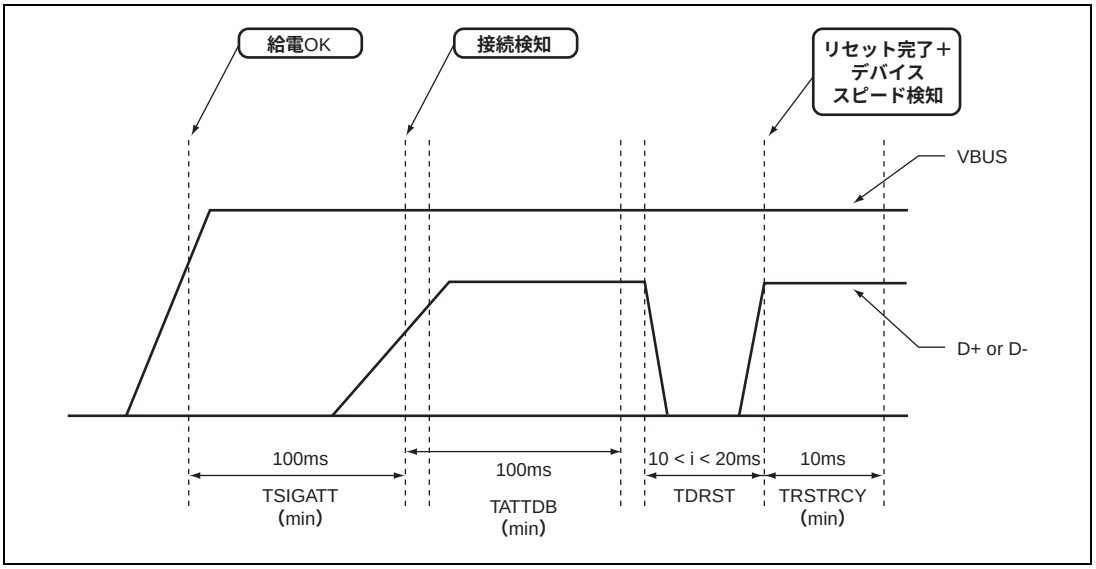


図 2.3 Port **変化タイミング**

2.5.6 ポートの状態遷移について

ホストはハブと協力しポートが現在どの状態にあるのかを知り、どの状態に遷移させるか、を指示する必要があります。

図 2.4 にハブが未構成状態から、ポートの給電開始、バスパワーファンクション接続、バスパワーでファンクションが起動し、HUB のポートにファンクションが接続されている事を認識し承認するまでのホスト、ハブ、ポート、ファンクションの状態を示します。

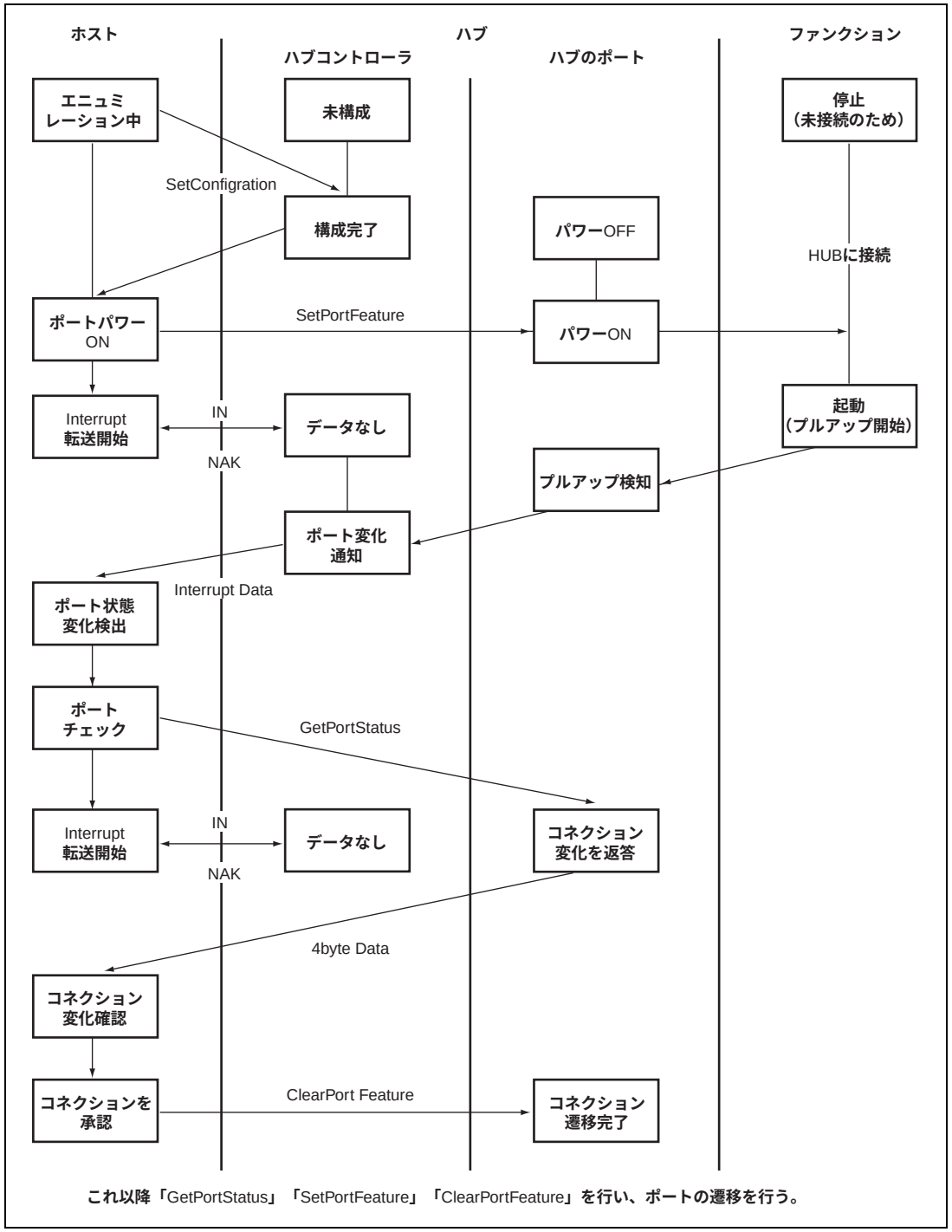


図 2.4　**ホスト、ハブ、ポート、ファンクションの状態遷移図**

3. 開発環境

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

- SH7727 Solution Engine（以下SH7727SE　型名MS7727SE01）日立超LSIシステムズ社製
- SH7727 E10A Emulator　ルネサス テクノロジ製
- PCMCIAスロット搭載のPC（Windows® 95/98）
- USBホスト用PC（Windows® XP）
- USBケーブル
- USBハブ
- High-Performance Debugging Interface（以下HDI）ルネサス テクノロジ製
- High-Performance Workshop（以下HEW）ルネサス テクノロジ製

3.1 ハードウェア環境

図 3.1 に各デバイスの接続形態を示します。

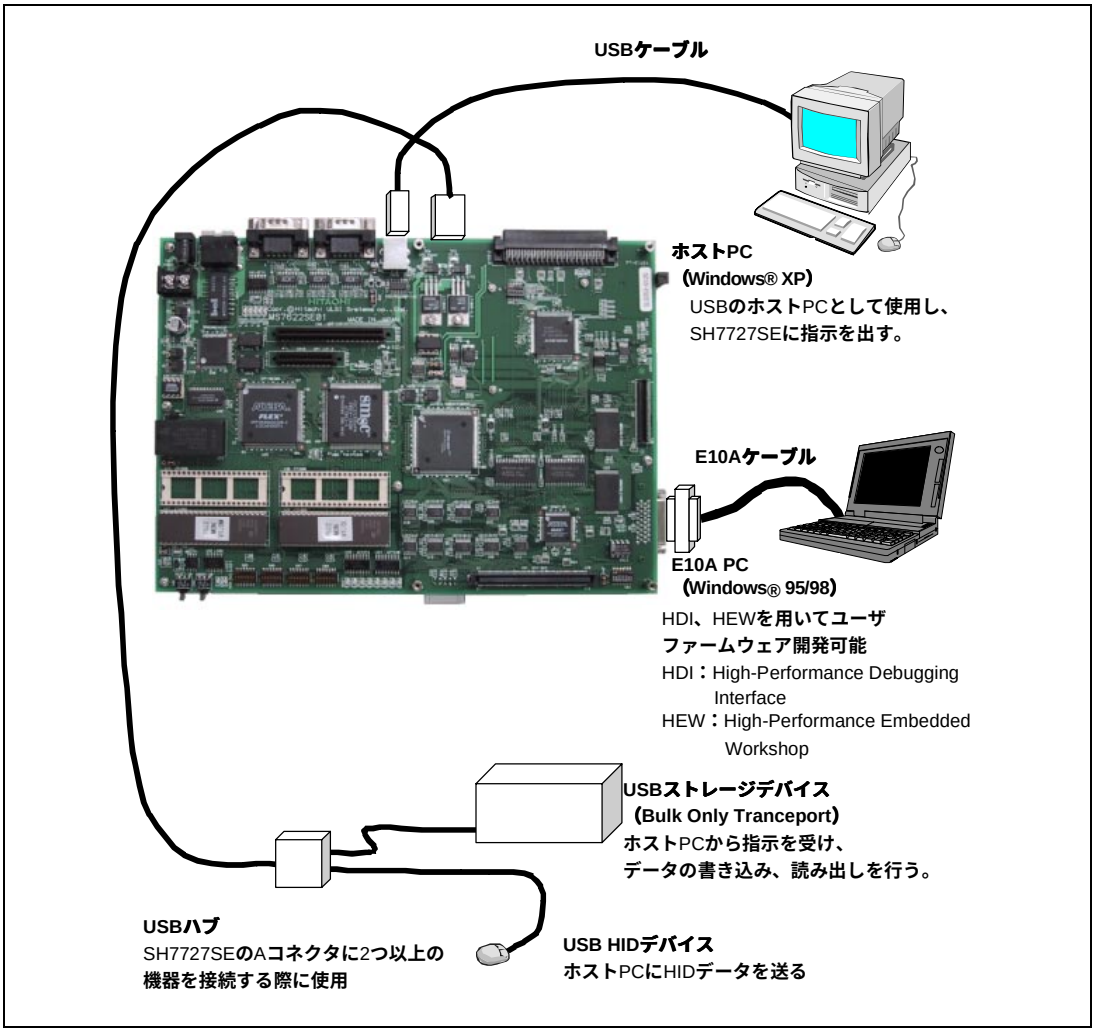


図 3.1　デバイスの接続形態

(1) SH7727SE

SH7727SE ボードのディップスイッチのいくつかを出荷時の設定から変更する必要があります。電源を投入する前に、これらの設定をよくご確認ください。その他のディップスイッチを変更する必要はありません。

表 3.1 ディップスイッチの設定

出荷時	変更後	ディップスイッチの機能
SW1-6 OFF	SW1-6 ON	エンディアンの選択
SW1-8 OFF	SW1-8 ON	E10A エミュレータの選択
JP14 1-2 Short	JP14 2-3 Short	USB D+プルアップコントロール信号選択
JP15 1-2 Short	JP15 2-3 Short	USB VBUS 信号検出ポート選択

(2) ホスト PC

USB ポート搭載の、Windows® XP をインストールしたパソコンを USB ホストとして使用します。本システムでは、上記 OS に標準で搭載されている USB Mass Storage Class（Bulk-Only Transport）、HID Class のデバイスドライバを使用しますので、新たにドライバを用意する必要はありません。

(3) E10A PC

PC カードスロットに E10A を挿入し、接続用のケーブルを介して SH7727SE と接続してください。接続後、HDI を起動してエミュレーションを行います。

(4) USB ハブ

SH7727SE の A コネクタに 2 つ以上の機器を接続する際に使用します。

(5) USB ストレージ機器

USB ハブを介して SH7727SE と接続してください。SH7727SE を通じて USB ホスト PC とデータの記録、読み出しを行います。

(6) USB HID 機器

USB ハブを介して SH7727SE と接続してください。SH7727SE を通じて USB ホスト PC にデータを入力します。

3.2 ソフトウェア環境

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について説明します。

3.2.1 サンプルプログラム

サンプルプログラムとして必要なファイルは、すべて SH7727 フォルダ内に収められています。HEW、HDI がインストールされたパソコンに、このフォルダごと移動して頂くと、すぐにサンプルプログラムを使用することができます。フォルダに含まれるファイルを以下**図** 3.2 に示します。

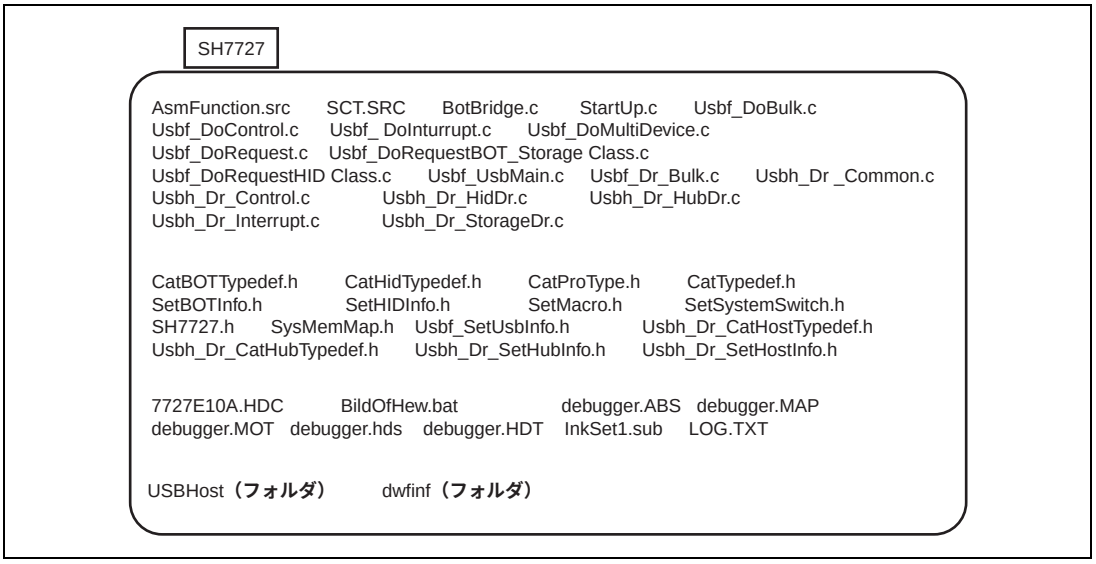


図 3.2 **フォルダに含まれるファイル**

3.2.2 コンパイルおよびリンク

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

High-Performance Embedded Workshop　Version1.0（release9）　（以下 HEW）

HEW を C:¥Hew にインストール*した場合、コンパイルおよびリンクの手順は以下のようになります。
まず、コンパイル時に作業用として、Tmp という名前のフォルダを C:¥Hew のフォルダ内に作成してください（**図** 3.3）。

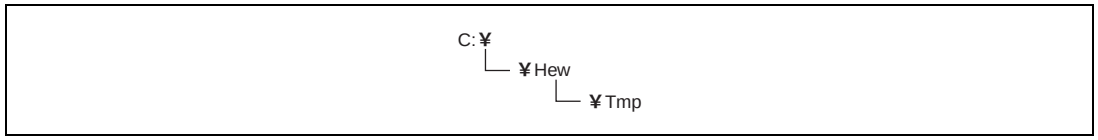


図 3.3　**作業フォルダの作成**

次に、サンプルプログラムが格納されているフォルダ（SH7727）を、任意のドライブ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.ABS の実行ファイルが作成されます。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（**図** 3.4 参照）。

【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** shc.exe **のパス、コンパイラが使用する環境変数** shc_lib **の設定は** shc.exe **のフォルダ、**shc_inc **の設定は** machine.h **のフォルダ、**shc_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** shcpic.lib **のパスを指定してください。**

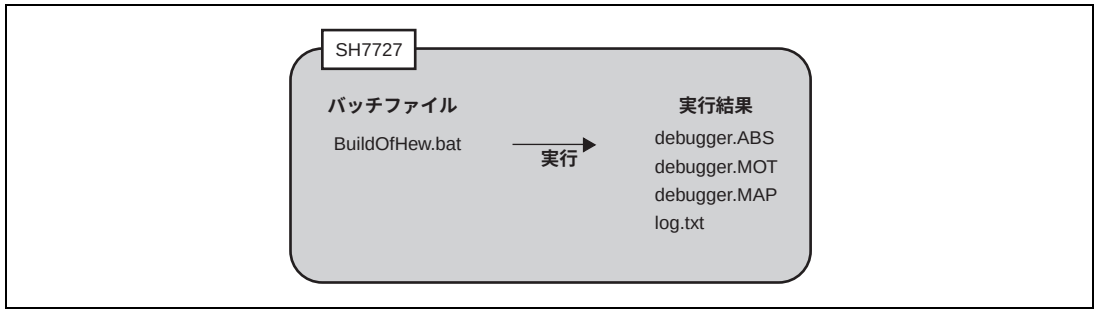
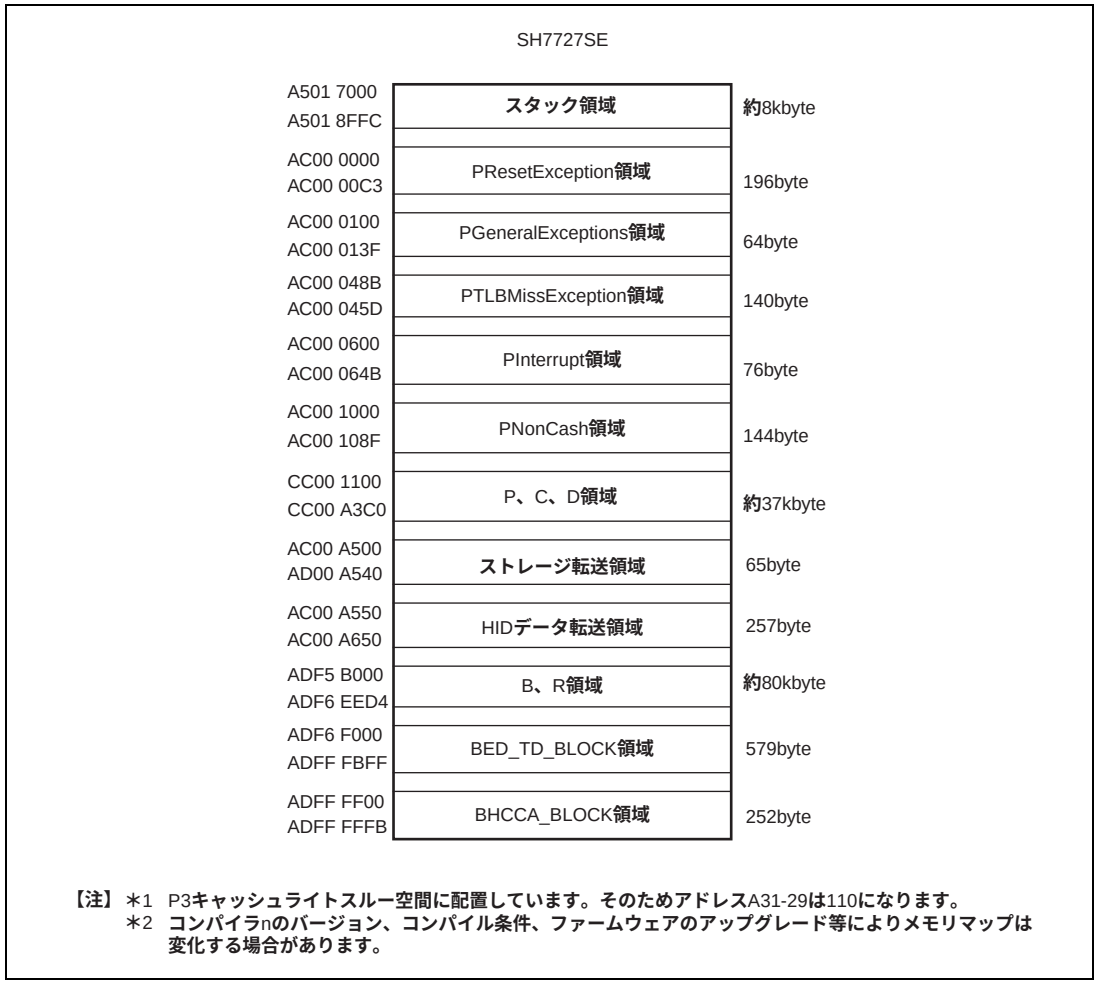


図 3.4　**コンパイル結果**

3.3 プログラムのロードと実行方法

図 3.5 にサンプルプログラムのメモリマップを示します。



3.5

図 3.5 のように、本サンプルプログラムは PResetException、PGeneralException、PTLBMissException、PInterrupt、PNonCash、P、C、D、R、B の領域を SDRAM 上に、スタック領域を内蔵メモリ上に配置しています。E10A でブレイク等の機能を使用するためには、この様にプログラムを RAM に配置する必要があります。これらのメモリへの割り付けは、SH7727 フォルダ内に含まれる lnkSet1.sub で指定します。フラッシュ等にプログラムを書き込み ROM 化する場合は、このファイルを変更してください。

3.3.1 プログラムのロード

SH7727SE の SDRAM へ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、ユーザケーブルでE10AとSH7727SEを接続してください。
- E10A用PCの電源を投入し、起動してください。
- HDIを起動してください。
- SH7727SEの電源を投入してください。
- PCの画面にダイアログ（**図**3.6参照）が表示されるので、SH7727SEのリセットスイッチ（SW1）をONにし、CPUをリセット後、「OK」ボタンをクリックまたは、<Enter>キーを押してください。
- メニューバーのView→CommandLineを選択し、ウインドを開き（**図**3.7参照）、左上のBatchFileボタンをクリックし、SH7727フォルダ内の7727E10A.hdcを指定してください。この操作によりBSCが設定され、SDRAMへのアクセスが可能となります。
- ファイルメニューからLoadProgram…を選択し、Load Programダイアログボックスで、SH7727フォルダ内のdebugger.ABSを指定してください。

以上の操作で、サンプルプログラムを SH7727SE の SDRAM 上にロードすることができます。

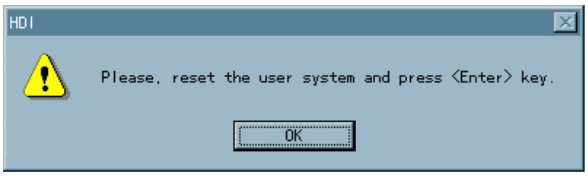


図 3.6　**リセット要求ダイアログ**

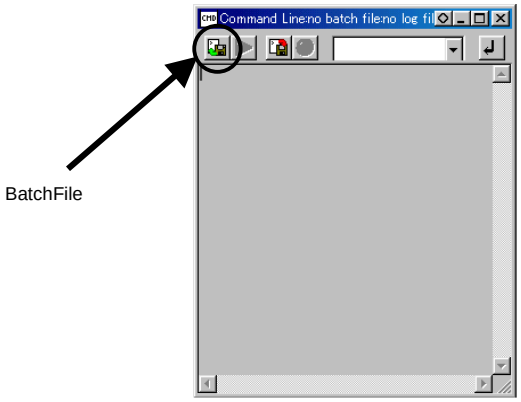


図 3.7　**コマンドライン入力**

3.3.2 プログラムの実行

メニューバーの Run→Go を選択するとプログラムが実行されます。

3.4 Multi Function Device の使用方法

Windows® XP を用いた場合を例に以下に説明します。

プログラムを実行した状態で、USB ケーブルのシリーズ A コネクタを SH7727SE に挿入し、反対側のシリーズ B コネクタを USB ハブに接続します。その後 USB ハブのポートにストレージデバイス、マウスを接続してください。

次に USB ケーブルのシリーズ B コネクタを SH7727SE に挿入し、反対側のシリーズ A コネクタを USB ホスト PC に接続します。

コントロール転送およびバルク転送を用いた周辺デバイスとしての起動終了後、デバイスマネージャーの USB コントローラの下に USB 大容量記憶装置デバイスが表示されます。その結果、ホスト PC は SH7727SE をストレージとマウスで構成される Multi Function Device として認識し、マイコンピュータの中にローカルディスクがマウントされ、USB マウスが使用できます。

以上で SH7727SE を USB 接続の Multi Function Device として使用できます。

4. サンプルプログラム概要

この章ではサンプルプログラムの特長やその構成について説明します。

本サンプルプログラムは SH7727SE 上で動作し USB ホストモジュールと USB ファンクションモジュールを使用します。SH7727SE が USB のホストとして、SH7727SE 上の USB シリーズ A コネクタに接続する USB ハブとファンクションデバイスに対し制御を行います。また、USB ファンクションモジュールも使い、ホスト PC に対して SH7727SE がファンクションデバイスとして動作します。

サンプルプログラムが使用する SH7727 の内蔵モジュールは、USB ホストモジュール、USB ファンクションモジュール、タイマ（1 チャネル）を使用します。

USB ファンクションモジュールを使用する USB 転送は USB ファンクションモジュールからの割り込みによって開始します。

USB ホストモジュールを使用する USB 転送は定常ルーチン内で USB ホストモジュールに要求した処理の完了を感知することにより開始します。

SH7727 内蔵モジュールの割り込みのうち、USB ファンクションモジュールに関連する割り込みは、USBFI0、USBFI1 の 2 種類ですが、本サンプルプログラムでは USBFI0 のみ使用しています。USB ホストモジュールに関連する割り込みは、USBH の 1 種類です。

本サンプルプログラムの特長を以下に示します。

- USBホストモジュールを使用してコントロール転送を行うことができます。
- USBホストモジュールを使用し、バルクアウト転送でデバイスにデータを送信することができます。
- USBホストモジュールを使用し、バルクイン転送でデバイスからデータを受信することができます。
- USBホストモジュールを使用し、インタラプトイン転送でデバイスからデータを受信することができます。
- USBファンクションモジュールを使用し、ホストPCとデータ転送ができます。
- USBホストモジュールとUSBファンクションモジュールを連携動作する事により、ホストPCとデバイス間のデータ転送をSH7727SEが中継できます。

4.1 サンプル全体図

本サンプルプログラムの全体図を**図** 4.1 に示します。1～8 の番号はデータ転送を表します。

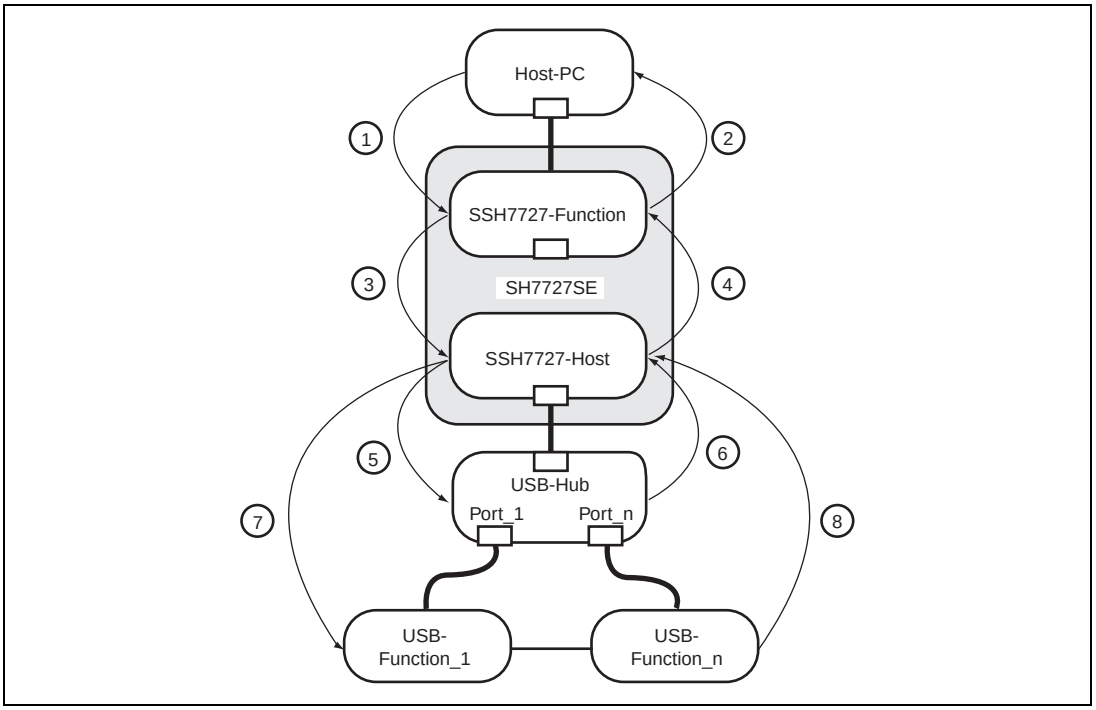


図 4.1 **全体図**

サンプルプロクラムの階層を**図** 4.2 に示します。

図 4.1 に示したデータ転送の 1 と 2 は SH7727SE の「Device layer」で行われます。

図 4.1 に示したデータ転送の 3 と 4 は SH7727SE の「Host-Function link layer」で行われます。

図 4.1 に示したデータ転送の 5～8 は SH7727SE の「USB system software layer」で行われます。

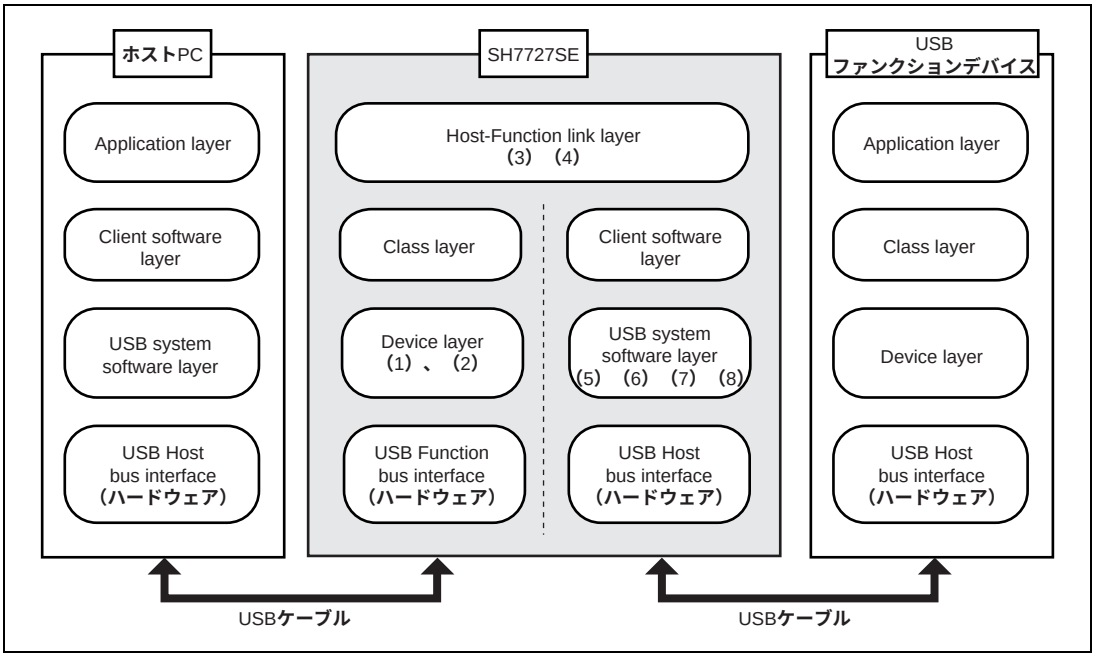


図 4.2　**システム階層図**

本サンプルプログラムは大きく 3 つに分ける事が出来ます。

1 つ目はホスト PC とデータ転送を行う SH7727 ファンクションのプログラム。

2 つ目は USB ハブ、USB ファンクションデバイスの制御とデータ転送を行う SH7727 ホストのプログラム。

3 つ目は SH7727 ファンクションのプログラムと、SH7727 ホストのプログラムをつなぎ転送データの橋渡しを行うリンクプログラム。

この 3 つのプログラムを**図** 4.2 に当てはめると、

SH7727 ファンクションのプログラムは「Class layer」「Device layer」に該当します。

SH7727 ホストのプログラムは「Client software layer」に該当します。この中ではさらに制御する対象により、ハブドライバ、HID ドライバ、ストレージドライバ、に分かれます。

リンクプログラムは「Host-Function link layer」に該当します。

4.2 状態遷移図

図 4.3 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** 4.3 のように 6 つの状態に遷移します。

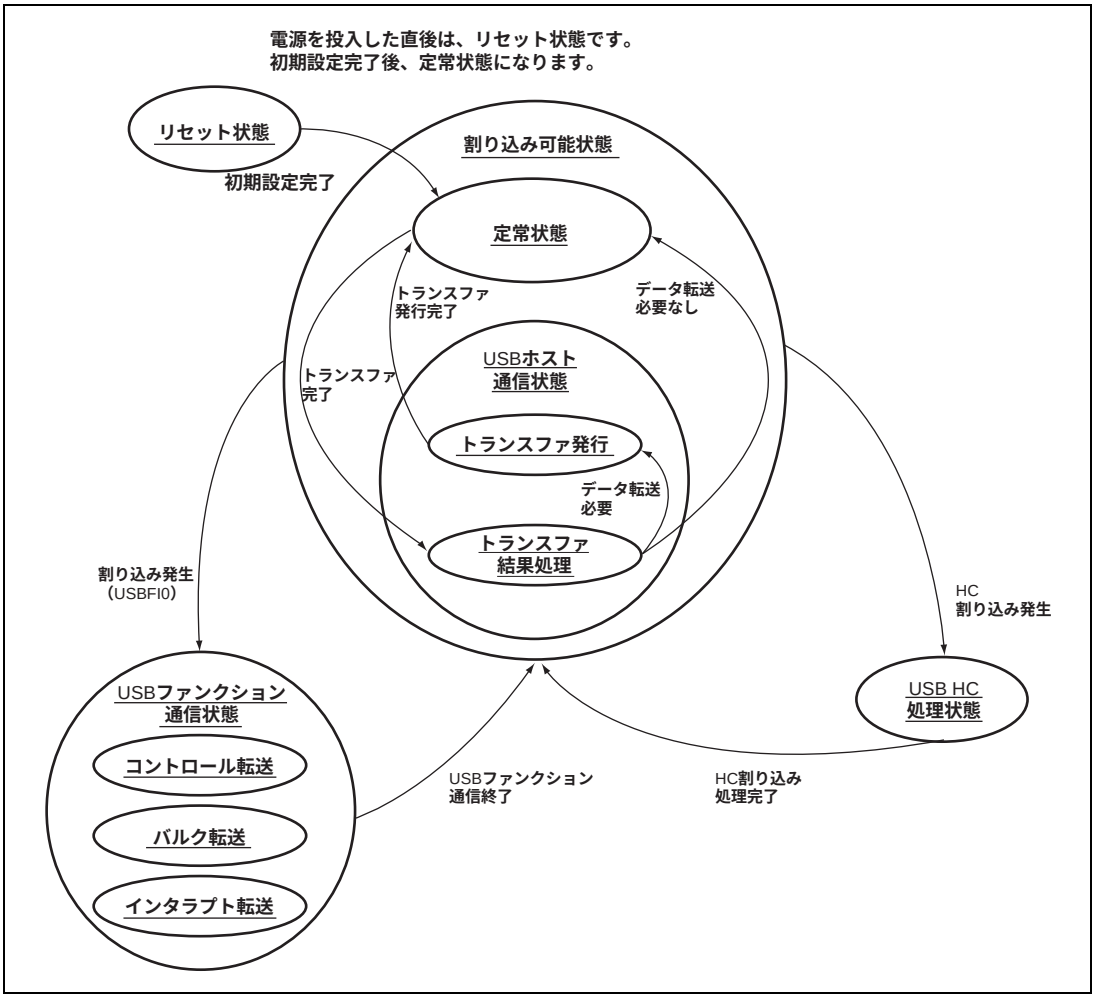


図 4.3 **状態遷移図**

- **リセット状態**

  パワーオンリセット・マニュアルリセットの際には、この状態になります。リセット状態では、主にSH7727の初期設定を行います。

- **割り込み可能状態**

  本状態時に割り込みが受け付けられます。この状態は「定常状態」「USBホスト通信状態」を内包しています。

- **定常状態**

  初期設定が完了すると、メインループで定常状態となります。

- USB**ホスト通信状態**

  SH7727ホストに、USBハブ、USBファンクションデバイスを接続すると、この状態になります。

- USB HC**処理状態**

  割り込み可能状態において、USBホスト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**ファンクション通信状態**

  割り込み可能状態において、USBファンクション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ファンクション通信状態では、割り込みの種類に応じた転送方式によるデータ転送を行います。本サンプルプログラムで使用する割り込みは割り込みフラグレジスタ0（USBIFR0）によって示される計８種類です。割り込み要因が発生すると、USBIFR0の対応するビットに1がセットされます。

4.3 USB ファンクション通信状態

USB ファンクション通信状態は、転送方式ごとに 3 つの状態に分類することができます（**図** 4.4 参照）。割り込みが発生すると、まず USB ファンクション通信状態へと遷移し、さらに割り込みの種類に応じて各転送状態へ分岐します。分岐の方法については**「第 5 章　サンプルプログラムの動作」**で説明します。

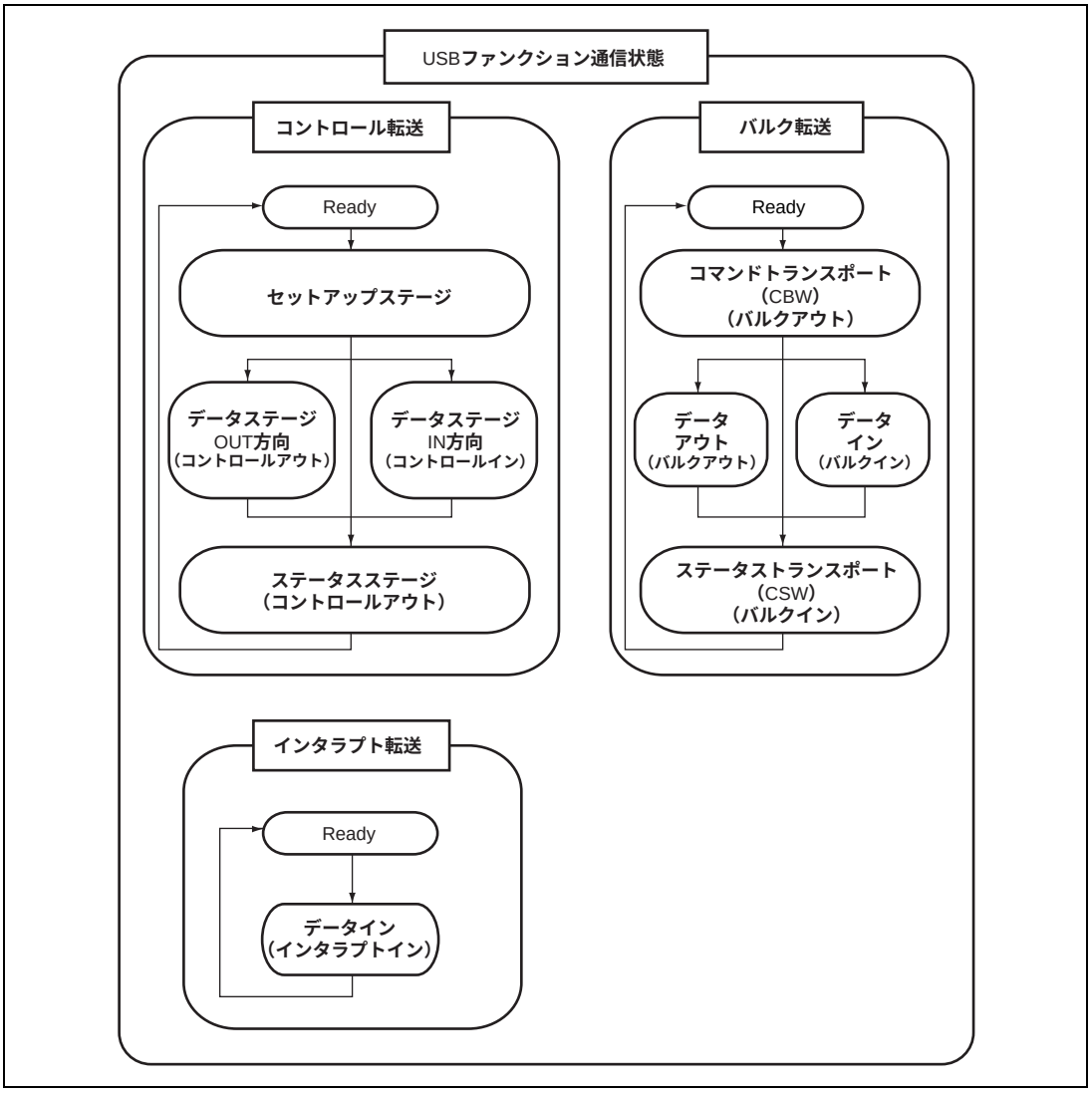


図 4.4　USB **ファンクション通信状態図**

4.3.1 コントロール転送について

本サンプルプログラムではコントロール転送をホスト PC から送られたデバイスリクエスト（標準、クラス）の返答に使用しています。返答するディスクリプタ情報は SH7727 ホストに接続されたファンクションから取得したディスクリプタ情報を元に作成します。

4.3.2 バルク転送について

本サンプルプログラムではバルク転送をストレージデータの転送に使用しています。

- ホストPCから送られたストレージデバイスへのパケットを受信し、リンクプログラムに渡す。
- リンクプログラムを介してストレージデバイスから送られたパケットをホストPCに送信する。

この 2 つを行う事により、ホスト PC とストレージファンクション間のデータを SH7727SE で転送します。

4.3.3 インタラプト転送について

本サンプルプログラムではインタラプト転送を HID デバイスからのデータ転送に使用しています。ホスト PC からの IN 要求で、転送バッファに HID デバイスから受信したデータが有れば送信し、無ければハードウェアにより NAK を送信します。なお、送信するデータは SH7727 ホストのプログラムを使用して、HID デバイスから転送バッファに書き込まれます。

4.4 USB ホスト通信状態

USB ホスト通信状態は、トランスファ（転送）を発行する状態と、その結果を処理する状態の２つの状態に分類することができます。

トランスファとは SH7727 ホストが USB ハブ、USB ファンクションデバイスに対して行うデータ転送の事です。

本サンプルソフトでのトランスファは、コントロール転送ではセットアップステージからステータスステージ完了。バルク転送、インタラプト転送では 1 パケットの転送が完了するとトランスファ完了になります。

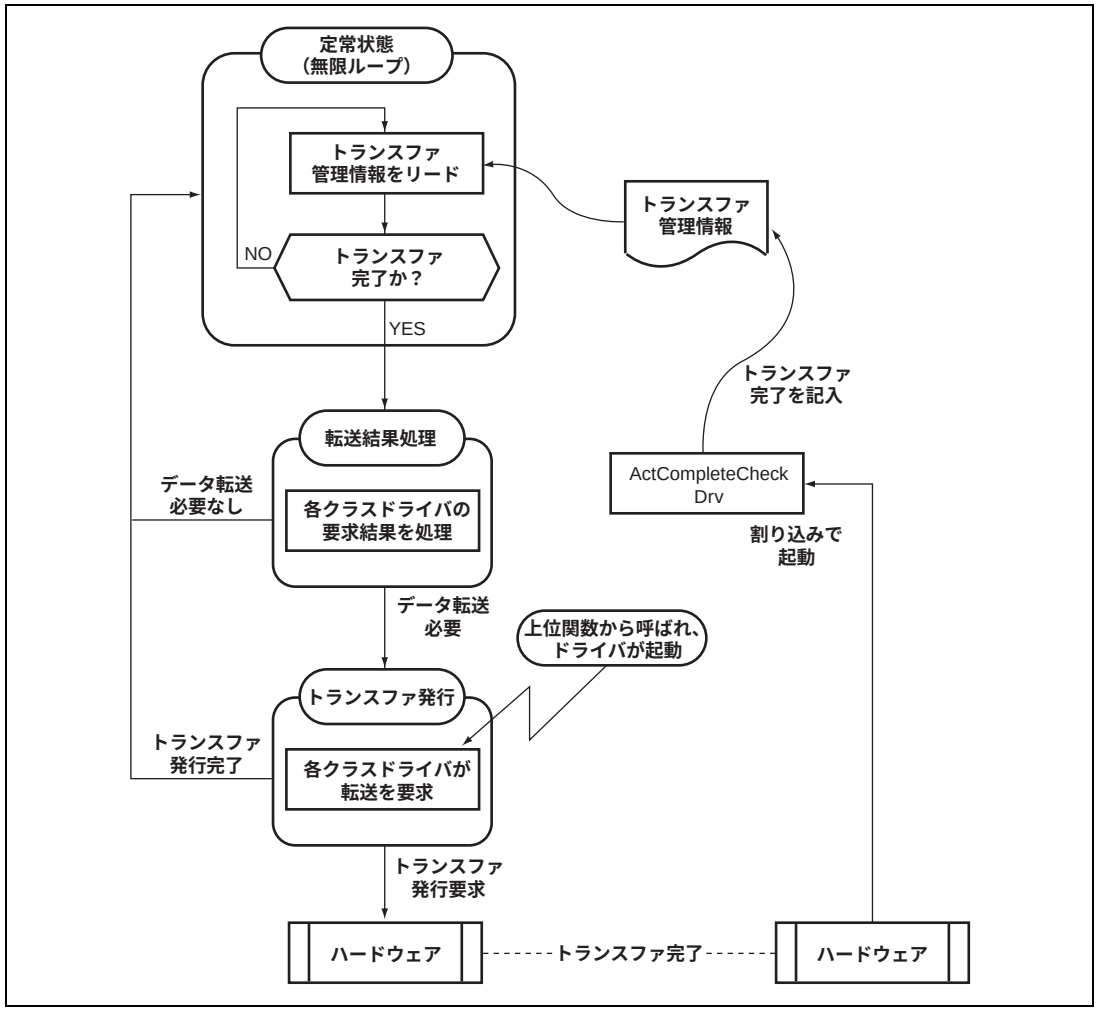


図 4.5　USB ホスト通信状態図

トランスファ（転送）を発行する状態では各転送方式（コントロール転送、バルク転送、インタラプト転送）を用いてデバイスとデータ転送を行います。

結果を処理する状態ではデバイスとデータ転送を行った結果に基づいた処理を行います。

各々のドライバは必要に応じて２つの状態、3 つの転送方式を使用しデバイスとデータ転送を行います。

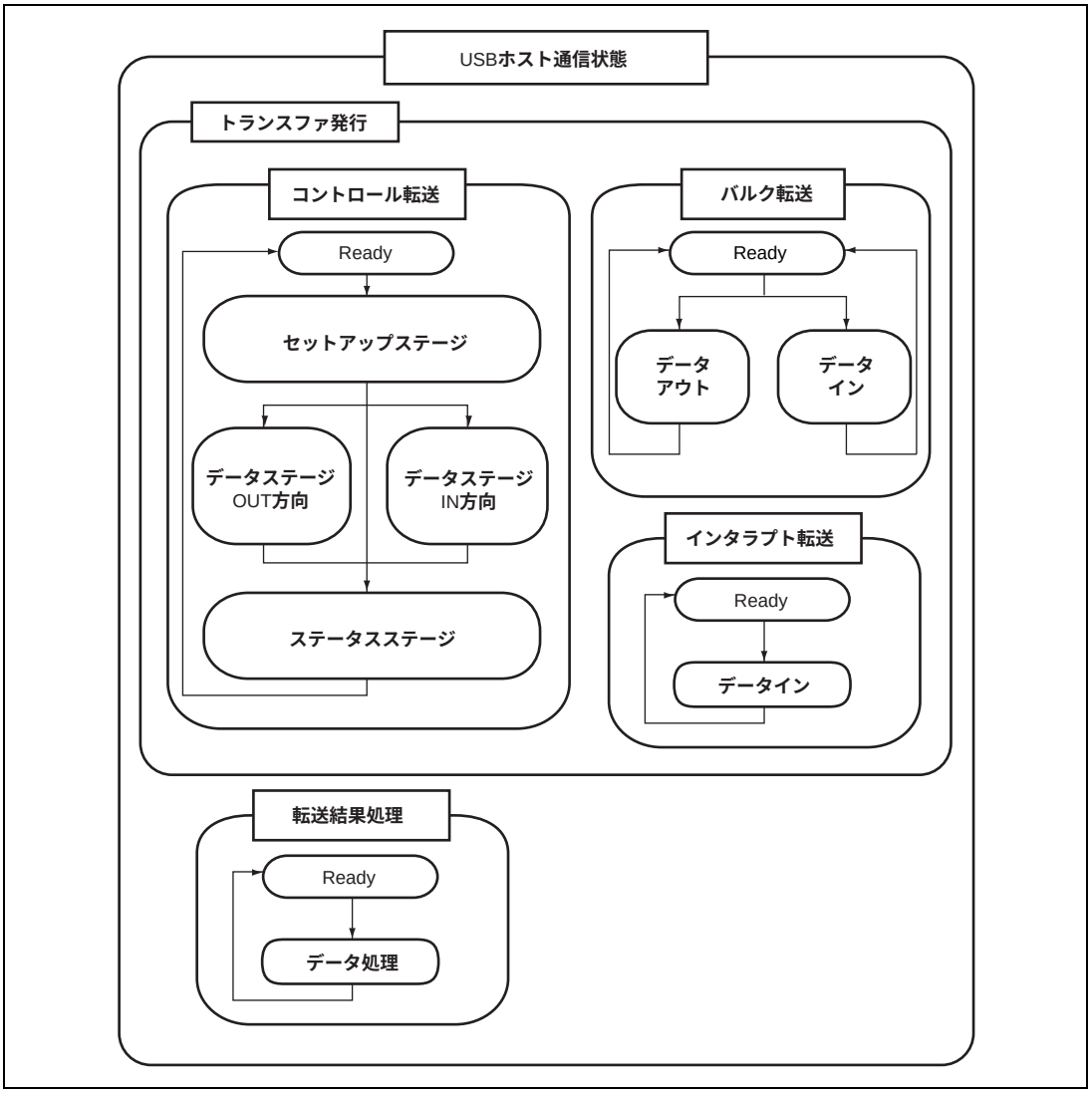


図 4.6　USB **ホスト通信状態**

4.4.1　トランスファ発行について

トランスファ（転送）を発行する状態では各転送方式（コントロール転送、バルク転送、インタラプト転送）を用いてデバイスとデータ転送を行います。USB 転送では必ずホスト側から転送を始める必要があるので、この状態からデバイスとのデータ転送が始まります。

どの転送方法を用いてどんな内容の転送を行うかは SH7727 ホストに接続した USB ハブ、USB ファンクションデバイスにより変わりますが、“ GET_DESCRIPTOR ”を用いて接続デバイスの情報取得、“ SET_ADDRESS ”を用いて接続デバイスへデバイスアドレスの割り付け、“ SET_CONFIGURATION ”を用いて接続デバイスを使用可能にする。この 3 種類の動作は必ず必要になります。

4.4.2 コントロール転送について

コントロール転送は主に、デバイス情報の取得、デバイスの動作状態を設定する際などに使用されます。そのため、SH7727SE にデバイスを接続した際、最初に行われる転送でもあります。

コントロール転送の一連の転送処理は、2 または 3 つのステージから構成されます。コントロール転送のステージは、「セットアップステージ」「データステージ」「ステータスステージ」に分類することができます。

4.4.3 バルク転送について

バルク転送は時間的制約がない大量のデータを、エラーなく転送する場合に使用します。データの転送速度は保証されませんが、データの内容は保証されます。SH7727SE とストレージデバイス間はバルク転送を用いて通信しています。

4.4.4 インタラプト転送について

インタラプト転送はあらかじめ定められた一定周期に少なくとも一度データ転送を行います。1 回に扱うデータは少量ですが、データの最低転送レートと、データの内容は保証されます。SH7727SE とハブ又は HID デバイス間はインタラプト転送を用いて通信しています。

4.4.5 転送結果処理について

トランスファ発行状態でホスト側からデバイスに対して生成したトランスファが完了（コントロール転送はステータスステージが終了、バルク、インタラプト転送は 1 トランザクションが終了）するとこの状態になります。

この状態では各転送方式（コントロール転送、バルク転送、インタラプト転送）を用いてデバイスとデータ転送した結果のチェック、デバイスからのデータ取得、次に行う処理の起動を行います。

4.5 ファイル構成

本サンプルプログラムは、18 個のソースファイルと 15 個のヘッダーファイルで構成されています。全構成ファイルを**表** 4.1 に示します。各関数は、転送方式または機能ごとに 1 つのファイルにまとめてあります。**図** 4.7 にシステム全体の階層構造を示します。**図** 4.8 と**図** 4.9 に各ファイルの関係を階層構造で示します。

表 4.1 ファイル構成

ファイル名	主な役割
StartUp.c	マイコンの初期設定
Usbf_UsbMain.c	割り込み要因の判定とパケットの送受信
Usbf_DoControl.c	コントロール転送を実行
Usbf_DoBulk.c	バルク転送を実行
Usbf_DoInterrupt.c	インタラプト転送を実行
Usbf_DoRequest.c	ホストが発行するセットアップコマンドの処理
Usbf_DoRequestBOT_StorageClass.c	Mass Storage Class（Bulk-Only Transport）クラスコマンドの処理
Usbf_DoRequestHIDClass.c	HID クラスコマンドの処理
BotBridge.c	Mass Storage Class（Bulk-Only Transport）データを USB ホストモジュール<->USB ファンクションモジュール間で橋渡しする
Usbf_DoMultiDevice.c	ディスクリプタ情報の作成を行う
Usbh_Dr_Common.c	Driver 層の共通関数
Usbh_Dr_Control.c	USB ホストで Control 転送の転送要求を行う
Usbh_Dr_Bulk.c	USB ホストで Bulk 転送の転送要求を行う
Usbh_Dr_Interrupt.c	USB ホストで Interrupt 転送の転送要求を行う
Usbh_Dr_HubDr.c	ハブの制御処理
Usbh_Dr_HidDr.c	HID デバイスの制御処理
Usbh_Dr_StorageDr.c	ストレージデバイスの制御処理
AsmFunction.src	スタックの設定
CatHidTypedef.h	HID クラスに必要な型の設定
CatBOTTypedef.h	BOT クラスに必要な型の設定
Usbh_Dr_CatHubTypedef.h	ハブクラスに必要な型の設定
CatProType.h	グローバル変数と関数のプロトタイプ宣言
CatTypedef.h	USB ファームウエアで使用する基本の構造体定義
Usbh_Dr_CatHostTypedef.h	USB Host に必要な型の設定
SetBOTInfo.h	BOT クラスの制御に必要な変数の初期設定
SetHIDInfo.h	HID クラスの制御に必要な変数の初期設定
Usbh_SetHostInfo.h	USB Host の制御に必要な変数の初期設定
Usbh_Dr_SetHubInfo.h	USB ハブの制御に必要な変数の初期設定
SetMacro.h	マクロ定義
SetSystemSwitch.h	システムの動作設定
Usbf_SetUsbInfo.h	USB 対応に必要な変数の初期設定
SysMemMap.h	SH7727SE のメモリマップのアドレス定義
SH7727.h	SH7727 レジスタ定義

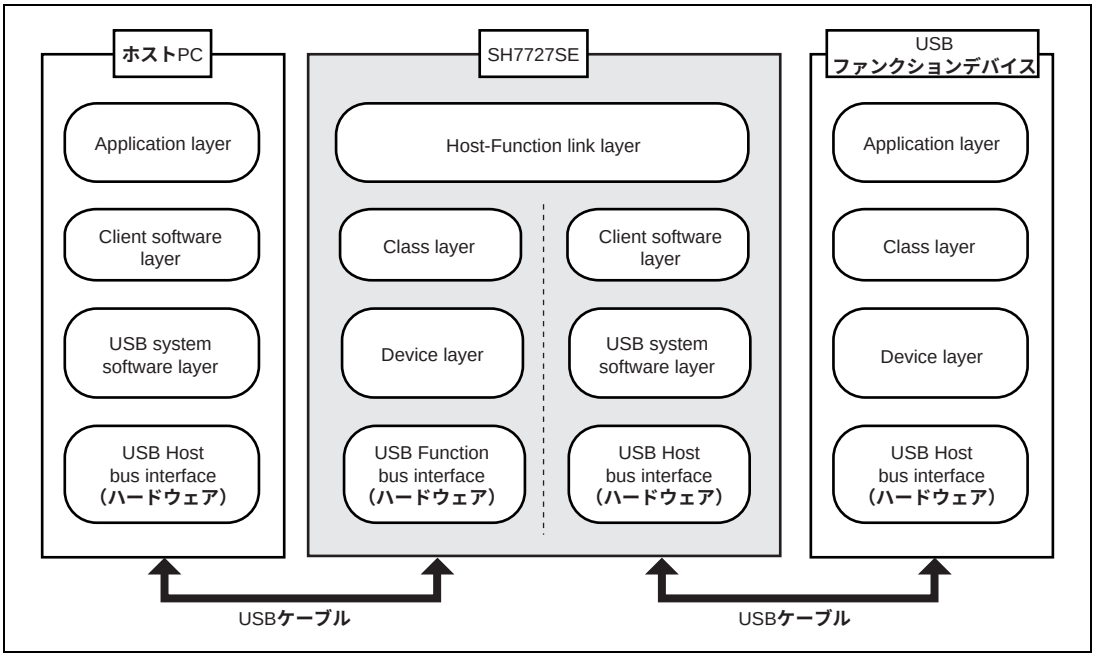


図 4.7　**システム階層構造**

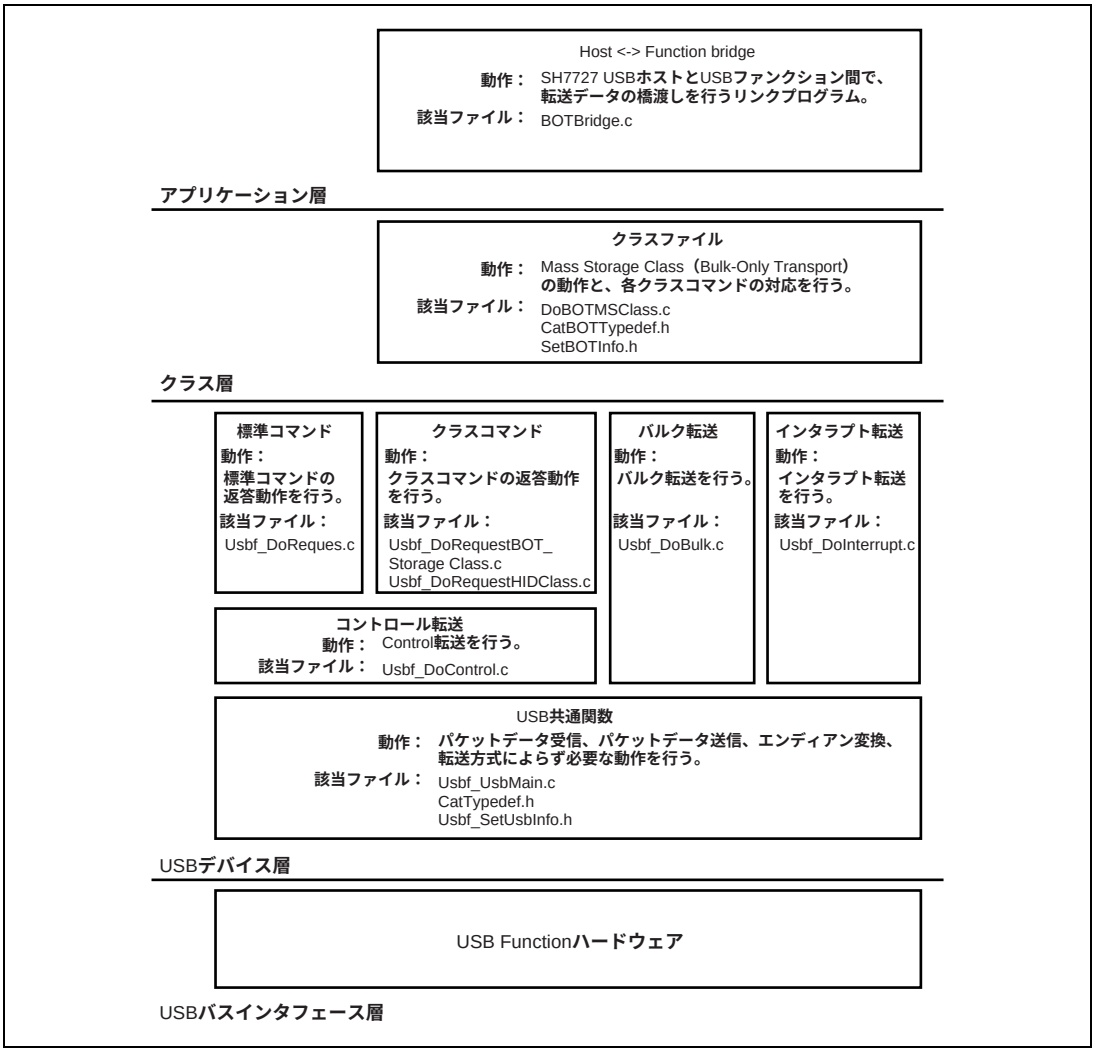


図 4.8　SH7727SE **ファンクション側の階層構造**

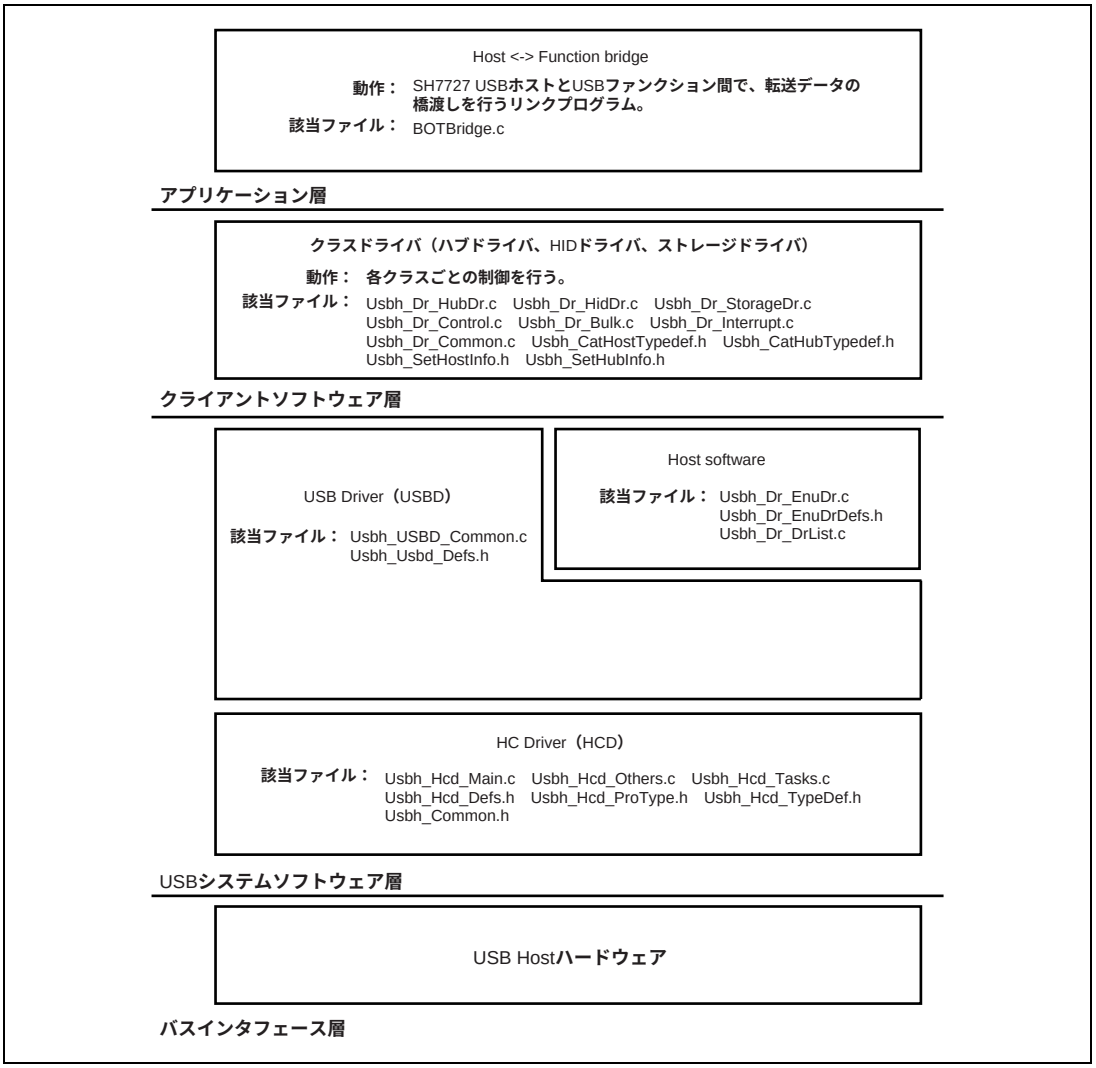


図 4.9　SH7727SE **ホスト側の階層構造**

表 4.2 に各ファイルに含まれる関数とその機能を示します。

表 4.2-1　StartUp.c

格納ファイル	関数名	機　　能
StartUp.c	CallReseException	リセット例外に対応する動作をし、引き続き実行する関数を呼び出す
	CallGeneralException	TLB ミス発生以外の一般例外に対応する関数を呼び出す
	CallTLBMissException	TLB ミス発生に対応する関数を呼び出す
	CallInterrupt	割り込み要求に対応する関数を呼び出す
	SetPowerOnSection	モジュールおよ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

パワーオンリセット、またはマニュアルリセットの際には、StartUp.c の SetPowerOnSection が呼び出されます。ここでは SH7727 の初期設定や、バルク転送に使用する RAM 領域のクリアを行います。

表 4.2-2　Usbf_UsbMain.c

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

Usbf_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 PC のモジュールと 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 4.2-3　Usbf_DoControl.c

格納ファイル	関数名	機　　能
Usbf_DoControl.c	ActControl	**コントロール転送のセットアップステージの制御を行う**
	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の制御を行う**
	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の制御を行う**
	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 4.2-4　Usbf_DoBulk.c

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

バルク転送に関する処理を行います。Mass Storage Class（Bulk-Only Transport）では ActBulkInReady を使用しません。

表 4.2-5　Usbf_DoInterrupt.c

格納ファイル	関数名	機　　能
Usbf_DoInterrupt.c	ActInterruptIn	**ホスト** PC **のホストモジュール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**

インタラプト転送に関する処理を行います。

表 4.2-6　Usbf_DoRequest.c

格納ファイル	関数名	機　　能
Usbf_DoRequest.c	DecStandardCommands	**インタラプトイン転送を行う**
	DecVenderCommands	**ベンダコマンドの対応を行う**

コントロール転送時に、ホスト PC のホストモジュール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：ルネサス）を使用しています。お客様にて製品を開発される場合は「USB Implementers Forum」にてお客様のベンダ ID を取得願います。また、ベンダコマンドは使用していないため、DecVenderCommands では何も行っていません。ベンダコマンドを使用する際には、お客様でプログラムを作成願います。

表 4.2-7　Usbf_DoRequestBOT _StorageClass.c

格納ファイル	関数名	機　　能
Usbf_DoRequestBOT _StorageClass.c	DecBOTClassCommands	Mass Storage Class（Bulk-Only Transport）クラスコマンドの対応を行う

コントロール転送時に、ホスト PC のホストモジュールから送られてくる Mass Storage Class（Bulk-Only Transport）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。

表 4.2-8　Usbf_DoRequestHIDClass.c

格納ファイル	関数名	機　　能
Usbf_DoRequestHID Class.c	DecHIDClassCommands	HID クラスコマンドの対応を行う

コントロール転送時に、ホスト PC のホストモジュールから送られてくる HID クラス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。

表 4.2-9　BotBridge.c

格納ファイル	関数名	機　　能
BotBridge.c	ActBulkOnly	Bulk-Only Transport のステートと割り込み要因別に振り分けを行う
	ActBulkOnlyCommand	CBW がホスト PC から送られてきた際の処理を行う。 CBW データを読み出し、ストレージデバイスに CBW を送信する。
	ActChangStateCBW	CBW 送信完了通知を受け取る関数。ステートを変更する。
	ActBulkOnlyIn	データトランスポート（データステージがイン方向の転送）とステータストランスポートの制御を行う
	ActBulkOnlyOut	データトランスポート（データステージがアウト方向の転送）とステータストランスポートの制御を行う
	ActCallBulkIn	BULK-IN 要求の返答受信完了通知を受け取る関数。STALL が返ってきたら、ステートを変更し Clear Feature を発行する。
	ActNop	何もせずに呼び出し先へ戻る関数。
	ActBulkOnlyStallIn	ステートが STALL になっている時、データトランスポート（データステージがイン方向の転送）とステータストランスポートの制御を行う。
	ActStallAfterCSW	データトランスポートの転送中に STALL したストレージデバイスに対して CSW を送信する。
	ActFreeBKOUT	BULK-OUT 送信完了通知を受け取る関数。

BotBridge.c では、Mass Storage Class（Bulk-Only Transport）のデータを SH7727 ホストと SH7727 ファンクション間で橋渡しします。

表 4.2-10　Usbf_DoMultiDevice.c

格納ファイル	関数名	機　　能
Usbf_DoMultiDevice.c	BuildDescriptorHid	HID クラスのデバイスが接続されると Descriptor 情報を作成する
	BuildDescriptor	Mass Storage Class Bulk-Only Transport クラスのデバイスが接続されると Descriptor 情報を作成する

Usbf_DoMultiDevice.c では、ホスト PC に SH7727SE を接続した時に使用する Descriptor 情報を作成します。

表 4.2-11　Usbh_Dr_Common.c

格納ファイル	関数名	機　　能
Usbh_Dr_Common.c	ActCompleteCheckClassDrv	クラスドライバがドライバに要求した作業の完了検知関数
	ActCompleteCheckDrv	クラスドライバが要求した JOB の検出関数
	EntryMemory	使用するメモリエリアを設定する
	FreeUpMemory	使用していたメモリエリアを空ける
	EntryJobMemory	ホストドライバに要求する時に使用する「要求するのに必要な領域」をエントリーする
	ActFindTable	指定したデバイス情報から指定したデバイスアドレスを探す
	ActFindInterfaceTable	指定したインタフェース情報から指定したインタフェース番号を探す
	errorHost	USB ホストのエラー動作時この関数に来る

Usbh_Dr_Common.c では、各クラスドライバの共通関数を集めた。

表 4.2-12　Usbh_Dr_Control.c

格納ファイル	関数名	機　　能
Usbh_Dr_Control.c	SendControl	指定したデバイスに対して Control 転送要求を行う。

Usbh_Dr_Control.c では、指定したデバイスアドレスに対してコントロール転送要求を行う。

表 4.2-13　Usbh_Dr_Bulk.c

格納ファイル	関数名	機　　能
Usbh_Dr_Bulk.c	SendBulk	指定したデバイスに対して Bulk 転送要求を行う。

Usbh_Dr_Bulk.c では、指定したデバイスアドレスに対してバルク転送要求を行う。

表 4.2-14　Usbh_Dr_Interrupt.c

格納ファイル	関数名	機　　能
Usbh_Dr_Interrupt.c	SendInterrupt	指定したデバイスに対して Interrupt 転送要求を行う。

Usbh_Dr_Interrupt.c では、指定したデバイスアドレスに対してインタラプト転送要求を行う。

表 4.2-15　Usbh_Dr_HubDr.c

格納ファイル	関数名	機　　能
Usbh_Dr_HubDr.c	OpenHubInterface	コンフィグレーションを行う。デバイス、インタフェース、エンドポイント情報を収集する為コントロール転送を使いその後、インタフェースを設定する。
	WriteHubInterface	OpenHubInterface 関数と対で動作し、デバイス、インタフェース、エンドポイント情報を記録する。
	ActChangPortState	ハブから送られてくるポート変更通知を受け取る関数
	ActAfterJob	タイマー割り込みで起動し、次に起動する関数を呼び出す。
	ActCallNewDevice	新しく接続されたデバイスのバス・エニュミレーションを開始する関数
	ActCheckHubPort	ハブポートの状態を調べる関数
	ActContorlHubPort	ポートの状態に応じて必要な設定を行う
	ActCheckPortState	指定したポートの状態を調べる（GET_STATUS For Port を発行する）関数
	ActSendSetFeature	Set Feature を発行する関数
	ActSendClearFeature	Clear Feature を発行する関数
	ActCount	指定した時間（ms 単位）分タイマーユニットでカウントするよう設定する関数
	ActFindHubPort	指定されたデバイスアドレスが、どのハブのダウンストリームポートに接続されているかを返す。

Usbh_Dr_HubDr.c では、接続されたハブの制御を行う。

表 4.2-16　Usbh_Dr_HidDr.c

格納ファイル	関数名	機　　能
Usbh_Dr_HidDr.c	OpenHidInterface	コンフィグレーションを行う。デバイス、インタフェース、エンドポイント情報を収集する為コントロール転送を使いその後、インタフェースを設定する。
	WriteHidInterface	OpenHidInterface 関数と対で動作し、デバイス、インタフェース、エンドポイント情報を記録後、interrupt 転送を開始する。
	ActReceiveHidData	Interrupt 転送で送ってきた Data を受信する。

Usbh_Dr_HidDr.c では、接続された HID デバイスと通信を行いディスクリプタ情報を取得する。その他に、インタラプト転送で送られたデータを SH7727 ファンクション側へ渡す動作を行う。

表 4.2-17　Usbh_Dr_StorageDr.c

格納ファイル	関数名	機　　能
Usbh_Dr_StorageDr.c	OpenMscBotInterface	**コンフィグレーションを行う。デバイス、インタフェース、エンドポイント情報を収集する為コントロール転送を使いその後、インタフェースを設定する。**
	WriteMscBotInterface	OpenMscBotInterface **関数と対で動作し、デバイス、インタフェース、エンドポイント情報を記録する。**

Usbh_Dr_StorageDr.c では、接続されたストレージデバイスと通信を行いディスクリプタ情報を取得する。

図 4.10～4.16 に、**表** 4.2 で説明した関数の相関関係を示します。上位側の関数が、下位側の関数を呼び出すことができます。また、複数の関数が同一の関数を呼び出すこともあります。定常状態では、CallResetException が他の関数を呼び出します。割り込みの発生によって遷移する USB 通信状態では、割り込み関数である CallInterrupt が BranchOfInt を呼び出し、BranchOfInt が他の関数を呼び出します。**図** 4.10　4.16 は、関数の上下関係を示しているもので、関数が呼び出される順序は示していません。関数がどのような順序で呼び出されるかについては、**「第 5 章　サンプルプログラムの動作」**のフローチャートをご覧ください。

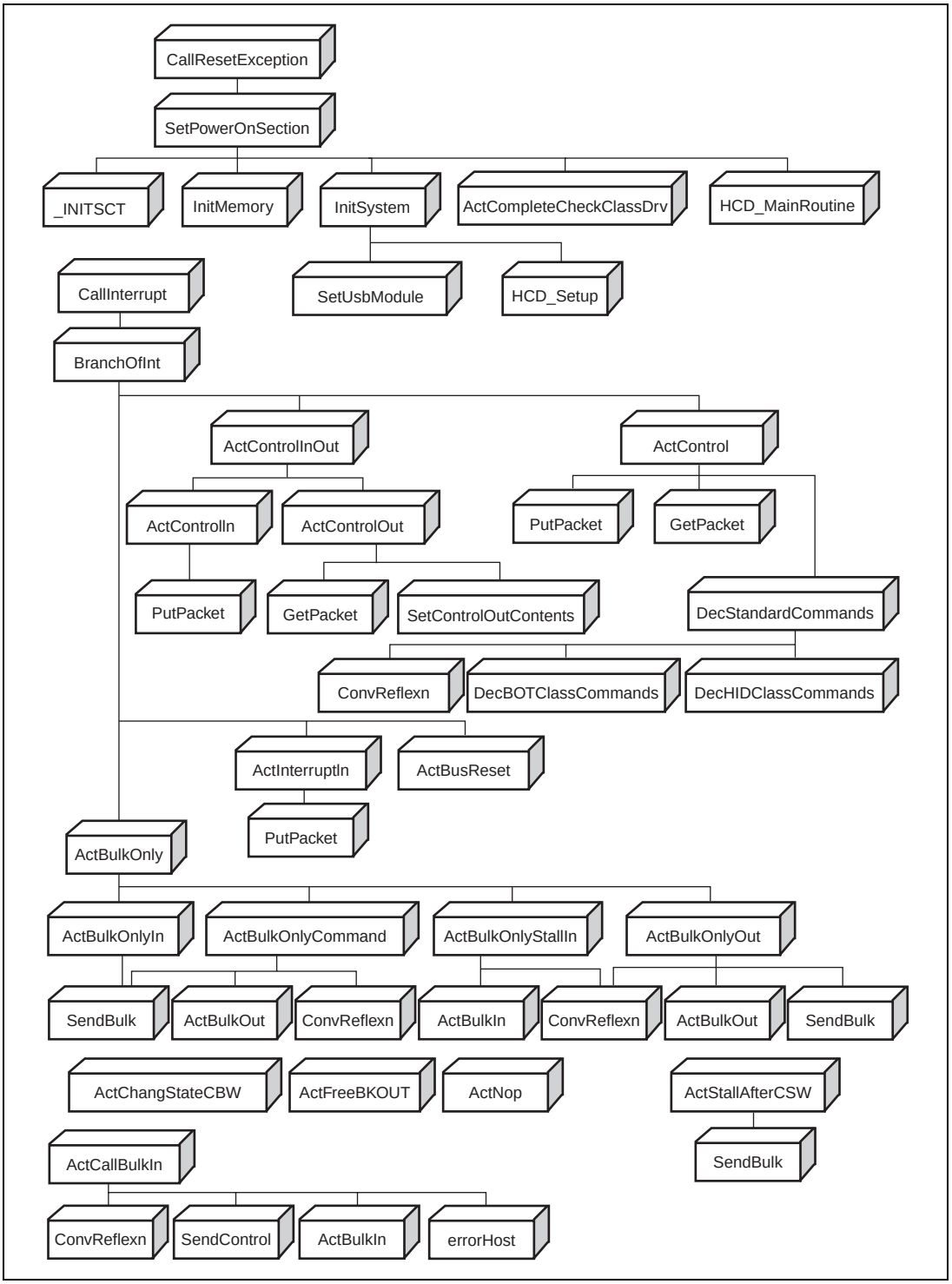


図 4.10　**関数の相関関係**

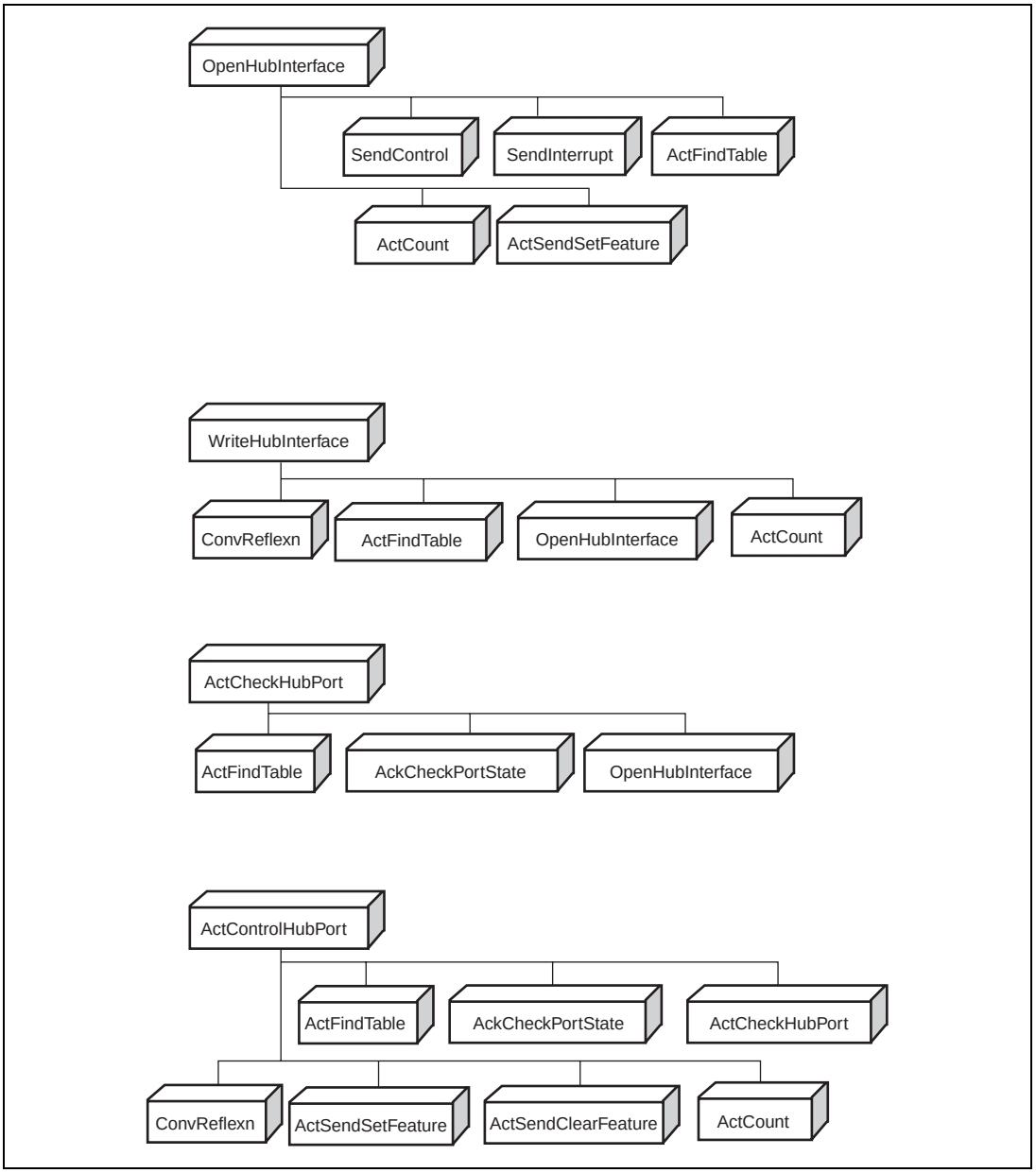


図 4.11　**関数の相関関係**

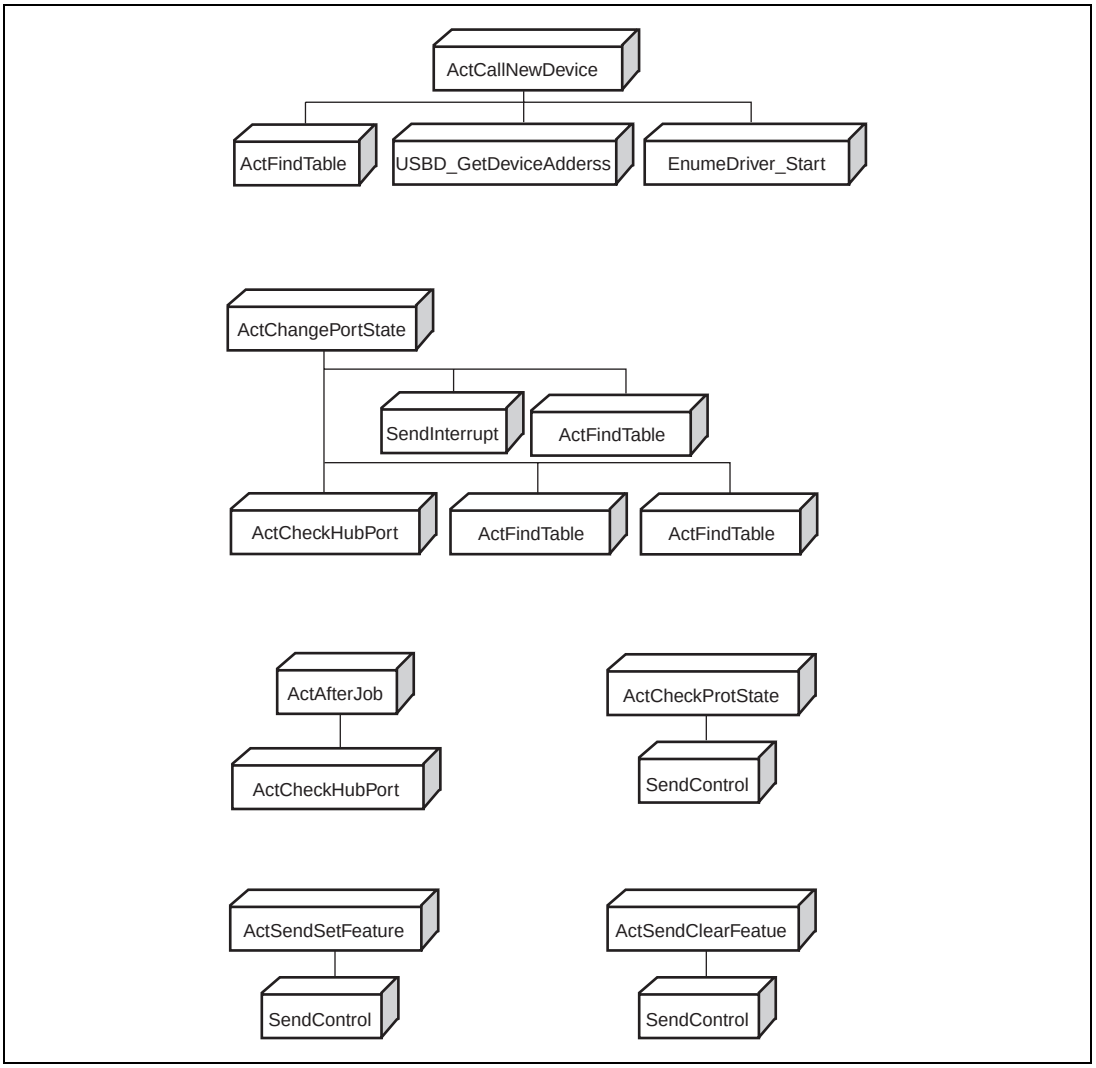


図 4.12　**関数の相関関係**

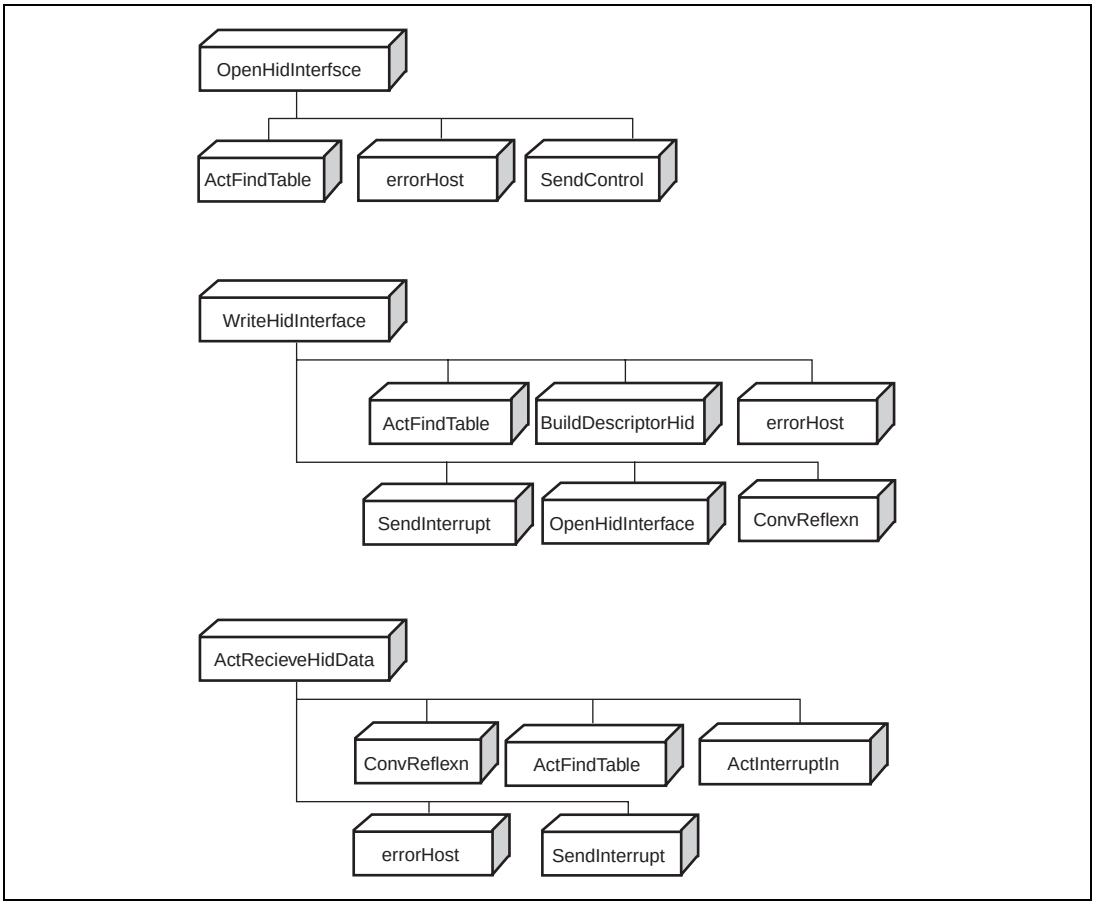


図 4.13　**関数の相関関係**

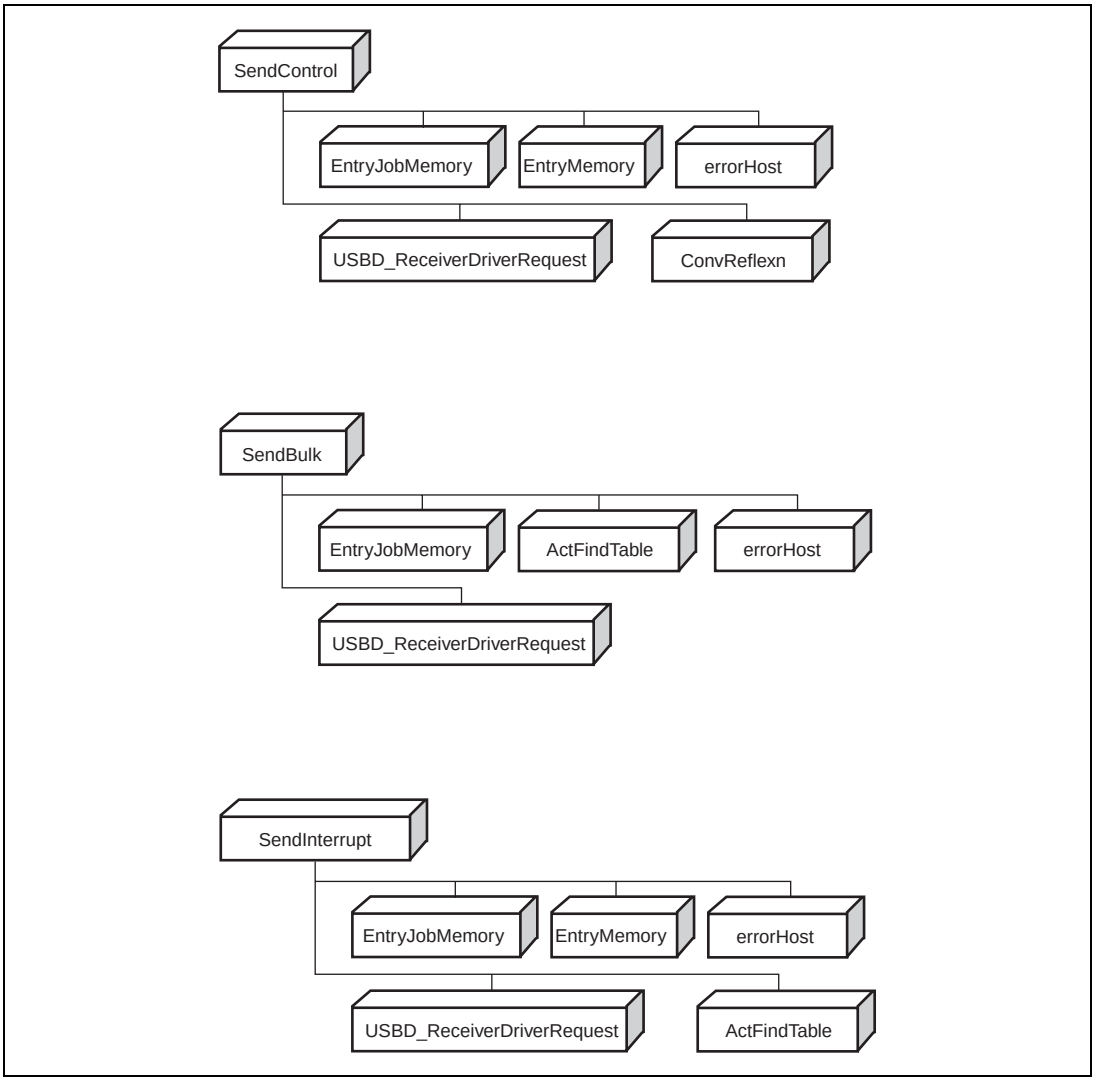


図 4.14　**関数の相関関係**

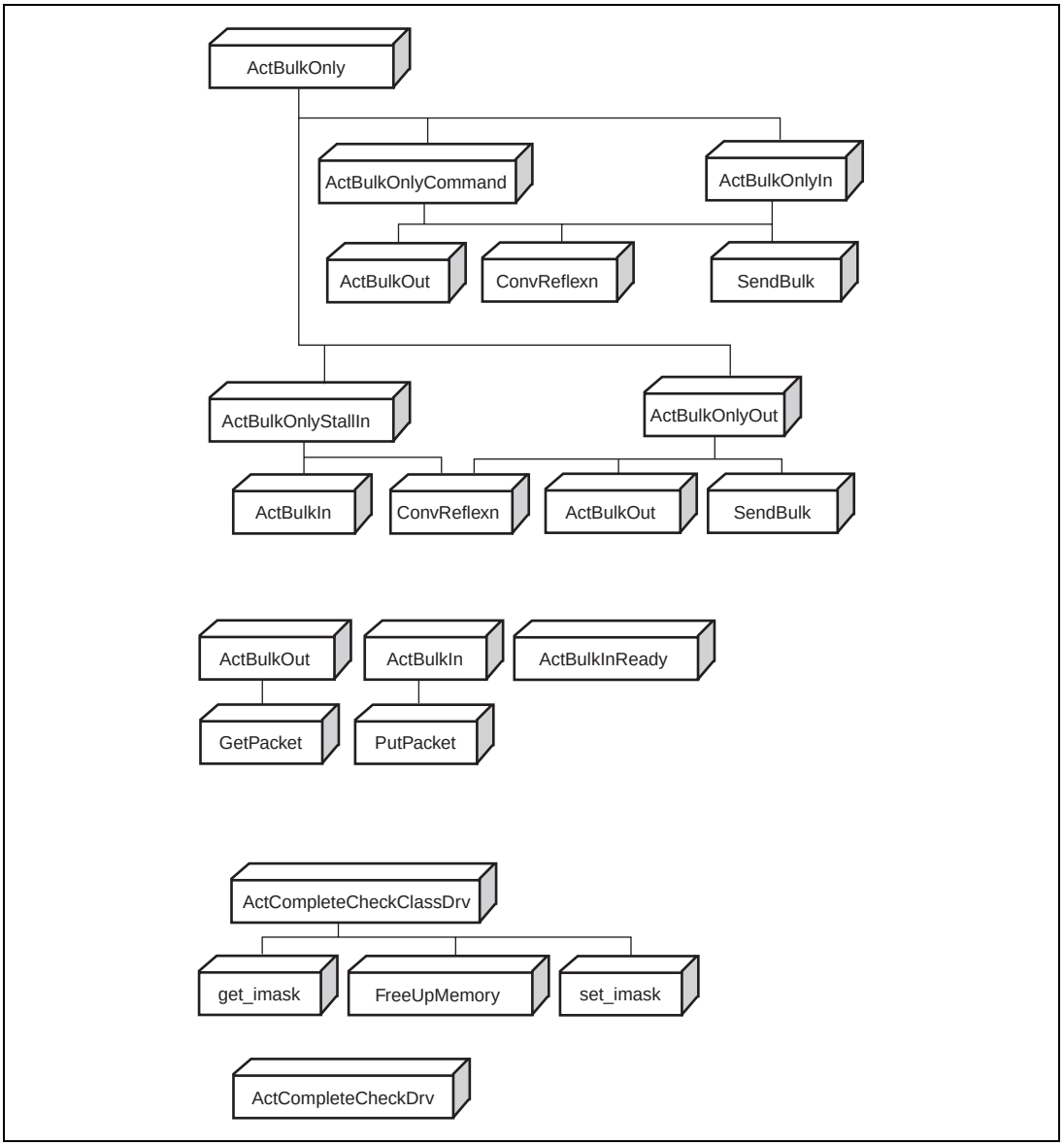


図 4.15　**関数の相関関係**

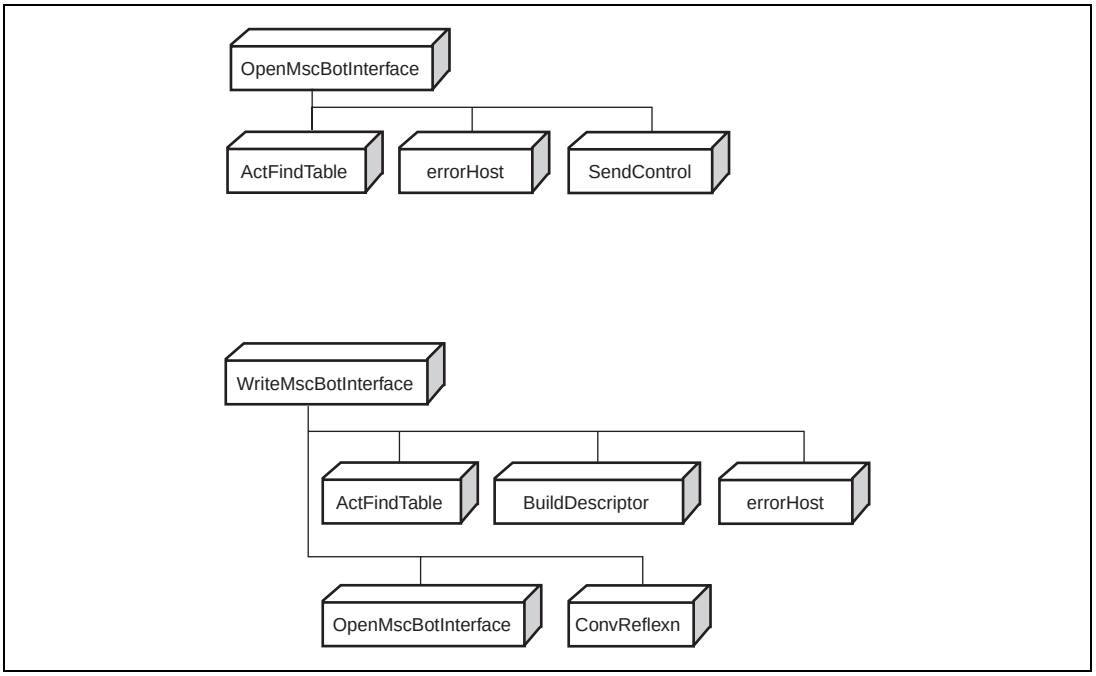


図 4.16　**関数の相関関係**

4.6 引数の型について

本サンプルプログラムでは C 言語が標準で備えている“ char ”などの変数型以外に変数の役割に基づいた型を定義し使用しています。型の定義は Usbh_Dr_CatHostTypedef.h の中で定義しています。

代表的な関数への引数の型としては、USB システムソフトウエア層の USBD に属する関数 USBD_ReceiveDriverRequest への引数は DriverRequestType 型。バルク転送の用意をする関数 SendBulk と、インタラプト転送の用意をする関数 SendInterrupt への引数は ClassDriverRequestType 型を、コントロール転送の用意をする関数 SendControl への引数は ClassDriverRequestType 型と SetupDataType 型を使用します。定義した変数の型名と要素を示します。

表 4.3-1 DriverRequestType **構造体のメンバ**

	メンバ型	メンバ名	内　　容
1	unsigned char	jobNum	**トランスファ管理番号**
2	unsigned char	sendDeviceNum	**デバイス番号**
3	unsigned char	sendEPsNum	**エンドポイント番号**
4	unsigned char*	Buffer	**データ転送領域へのポインタ**
5	unsigned long	BufferLength	**データ転送サイズ**
6	unsigned long	direction	**データ転送方向**
7	unsigned long	Status	**トランスファ完了状態**
8	unsigned char	Setup[8]	**セットアップパケットデータ**
9	unsigned char	Ep_Type	**エンドポイントのタイプ**
10	unsigned short	Ep_Mps	**エンドポイントのマックスパケットサイズ**
11	unsigned char	Ep_Speed	**エンドポイントのスピード**
12	unsigned char	*Function(unsigned char)	**トランスファが完了した際の呼び出し関数**

クラスドライバが USB システムソフトウエア層の関数 USBD_ReceiveDriverRequest への引数型として使用します。

表 4.3-2 ClassDriverRequestType **構造体のメンバ**

	メンバ型	メンバ名	内　　容
1	DeviceListType*	deviceList	**各クラスのホストドライバが管理するデバイス情報へのポインタ**
2	unsigned char	deviceAddress	**転送相手のデバイスアドレスナンバ**
3	unsigned char	interfaceNum	**転送相手のインタフェースナンバ**
4	unsigned char	sendEPNum	**転送相手のエンドポイントナンバ**
5	unsigned char	direction	**データ転送の向き**
6	unsigned short	bufferLength	**データ転送サイズ**
7	unsigned char*	buffer	**データ転送領域へのポインタ**
8	void	*CallOn	**トランスファが完了した際の呼び出し関数**

トランスファを作る SendControl、SendBulk、SendInterrupt 関数への引数型として使用します。

表 4.3-3　DeviceListType **構造体のメンバ**

	メンバ型	メンバ名	内　　容
1	unsigned char	dAddres	**デバイスアドレスナンバ**
2	unsigned char	dSpeed	**デバイススピード**
3	unsigned char	dReady	**デバイス準備**
4	unsigned char	dIfNum	**デバイスが持っているインタフェース数**
5	IFListType*	dIfInfoPtr	**デバイスのインタフェース情報へのポインタ**

この構造体は各クラスのホストドライバがデバイスを管理する際使用するデバイス情報の構造体です。
この構造体は ClassDriverRequestType 構造体に含まれるデバイス情報への型としても使用します。

表 4.3-4　IFListType **構造体のメンバ**

	メンバ型	メンバ名	内　　容
1	unsigned char	ifConfigNum	**構成番号**
2	unsigned char	ifNum	**インタフェース番号**
3	unsigned char	ifAltemateSetNum	**アルタネートセッティング番号**
4	unsigned char	ifHaveEpNum	**このインタフェースが扱えるエンドポイント数**
5	unsigned char	ifClass	**インタフェースクラス番号**
6	unsigned char	ifSubClass	**サブクラス番号**
7	unsigned char	ifProtocol	**使用プロトコル番号**
8	EPListType*	ifEpInfoPtr	**このインタフェースが扱えるエンドポイント情報へのポインタ**

この構造体は各クラスのホストドライバがデバイスを管理する際使用するインタフェース情報の構造体です。
この構造体は DeviceListType 構造体に含まれるインタフェース情報への型としても使用します。

4.3-5　EPListType

	メンバ型	メンバ名	内　　容
1	unsigned char	epAddress	**エンドポイント番号と転送方向**
2	unsigned char	epAttributes	**転送方式**
3	unsigned char	epMaxPacketSize	**マックスパケットサイズ**
4	unsigned char	epInterval	**エンドポイントへの最低アクセス周期**

この構造体は各クラスのホストドライバがデバイスを管理する際使用するエンドポイント情報の構造体です。
この構造体は IFListType 構造体に含まれるエンドポイント情報への型としても使用します。

表 4.3-6　SetupDataType 構造体のメンバ

	メンバ型	メンバ名	内　　容
1	unsigned char	ByteVal[0]	BmRequest を格納
2	unsigned char	ByteVal[1]	bRequest を格納
3	unsigned char	ByteVal[2]	wValue を格納
4	unsigned char	ByteVal[3]	wValue を格納
5	unsigned char	ByteVal[4]	wIndex を格納
6	unsigned char	ByteVal[5]	wIndex を格納
7	unsigned char	ByteVal[6]	wLength を格納
8	unsigned char	ByteVal[7]	wLength を格納

SendControl 関数への引数型として使用します。コントロール転送で送信するセットアップに続く 8 バイトのセットアップパケットデータを扱う際に使用します。この構造体に格納する値はビックエンディアンの値を格納してください。

4.7 マルチファンクションについて

本サンプルプログラムではインタフェースを複数持っているとホスト PC に通知します。これによりホスト PC では必要に応じて本ファンクションのエンドポイントにアクセスを行います。

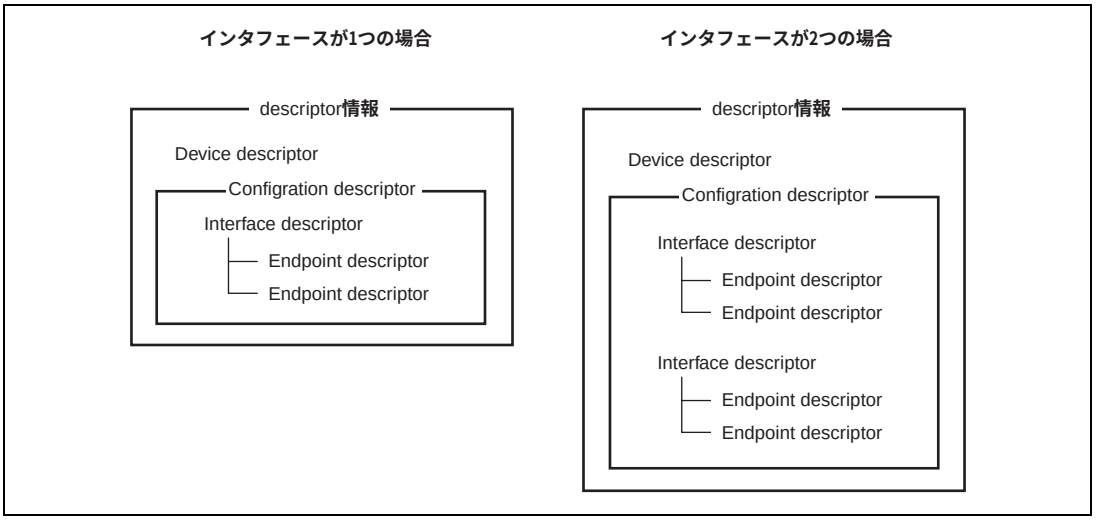


図 4.17 ディスクリプタ情報構造図

4.7.1 ディスクリプタについて

ディスクリプタ情報は SH7727 ホストに接続したファンクション（ストレージ、HID クラス）のディスクリプタ情報を元に作成します。接続したファンクションがハブの場合ディスクリプタ情報は作成しません。

ハブ以外のファンクション接続を検知すると 1 つのインタフェースを持つディスクリプタ情報を作成します。この後 2 つ目のファンクション接続を検知すると、既に作成済みのディスクリプタ情報にインタフェースディスクリプタを追加することにより 2 つのインタフェースを持つディスクリプタ情報を持つ事が出来ます。

尚、一度接続したファンクションを切断した場合、作成されたディスクリプタ情報は削除されません。再度ディスクリプタ情報の作成が必要になりますので、ホスト PC に SH7727SE を接続する前にリセットもしくは NMI を入れてください。

4.8 デバイスドライバについて

本サンプルプログラムでは SH7727 ホストに USB ハブ、USB ファンクションデバイスを接続します。接続を検知すると SH7727 ホストはデータ転送を行い USB ハブ、USB ファンクションデバイスを制御する必要があります。その制御内容は USB のクラスにより異なります。本サンプルプログラムでは 3 種類のドライバ（ハブドライバ、HID ドライバ、ストレージドライバ）を持ち、各クラスの制御を行っています。

4.8.1 ハブドライバで行う事

ハブドライバではディスクリプタ情報の取得、取得したディスクリプタ情報を元にハブクラスデバイスの管理、コンフィギュレーションの設定、下位ポートの設定、下位ポートの状態監視と対応などを行っています。

尚、制限事項としてダウンストリームポートの電力監視&制御を行っていません。

また、接続出来るハブは 4 個までとなっています。

4.8.2 HID ドライバで行う事

HID ドライバではディスクリプタ情報の取得、取得したディスクリプタ情報を元に HID クラスデバイスの管理、コンフィギュレーションの設定、HID デバイスからのデータ受信などを行っています。

尚、制限事項として接続出来る HID デバイスは 1 個です。

4.8.3 ストレージドライバで行う事

ストレージドライバではディスクリプタ情報の取得、取得したディスクリプタ情報を元にストレージデバイスの管理、コンフィギュレーションの設定などを行っています。

尚、制限事項として接続出来るストレージデバイスは 1 個です。

4.9 ホストとファンクションの連携について

本サンプルプログラムでは、SH7727 ホストと SH7727 ファンクションが連携してデータを転送することにより、ホスト PC と USB ファンクションのデータ送受信を実現しています。

SH7727 ホストに USB ファンクションを接続し、SH7727 ファンクションにホスト PC を接続します。ホスト PC は SH7727 ファンクションの接続を検知するとバス・エニュミレーションを行います。

その後 SH7727SE が返答したディスクリプタ情報に基づき各エンドポイントとデータ転送を行います。

4.9.1 HID クラスでの連携

HID クラスの場合、SH7727 ホストと SH7727 ファンクションは SH7727SE に搭載されたメモリ上にリングバッファを用意し、データの受け渡しを行います。

SH7727 ホストは、コンフィグレーション終了後 HID デバイスから申告されたエンドポイントに対しインタラプト転送を使用し、IN トークンを発行します。HID デバイスは、送るべきデータがある場合データを送り返し、無い場合“ NAK ”を返します。データを受信するとリングバッファにデータをコピーし、インタラプト転送を再開します。

ホスト PC は SH7727 ファンクションのエンドポイント 3 に対して IN トークンを送ってくるので、リングバッファに転送するデータがある場合はデータを送り返し、無い場合“ NAK ”を返します。

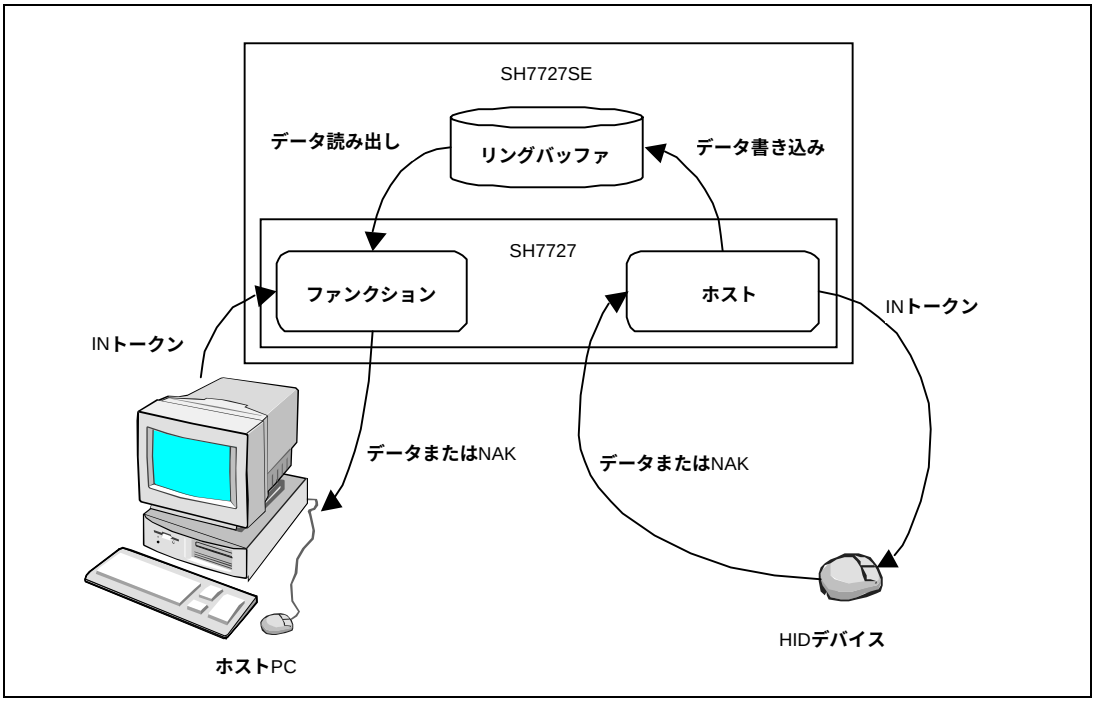


図 4.18 HID クラス連携図

4.9.2 ストレージクラスでの連携

ストレージクラスの場合、SH7727 ホストと SH7727 ファンクションは SH7727SE に搭載されたメモリ上に転送バッファを用意し、データの受け渡しを行います。データの受け渡しは 1 パケット毎に行われます。ここでは USB Mass Storage Class の Bulk-Only Transport の転送手順に従い、どのような流れになるかを説明します。

ホスト PC が発行した CBW を SH7727 ファンクションが受信すると受信割り込みを起こし、USB 受信割り込みルーチンを起動します。この割り込みルーチンは

- SH7727ファンクションからCBWを転送バッファへ移動
- SH7727ホストを使用しCBWを転送バッファからストレージデバイスに転送

の 2 つを順次行います。

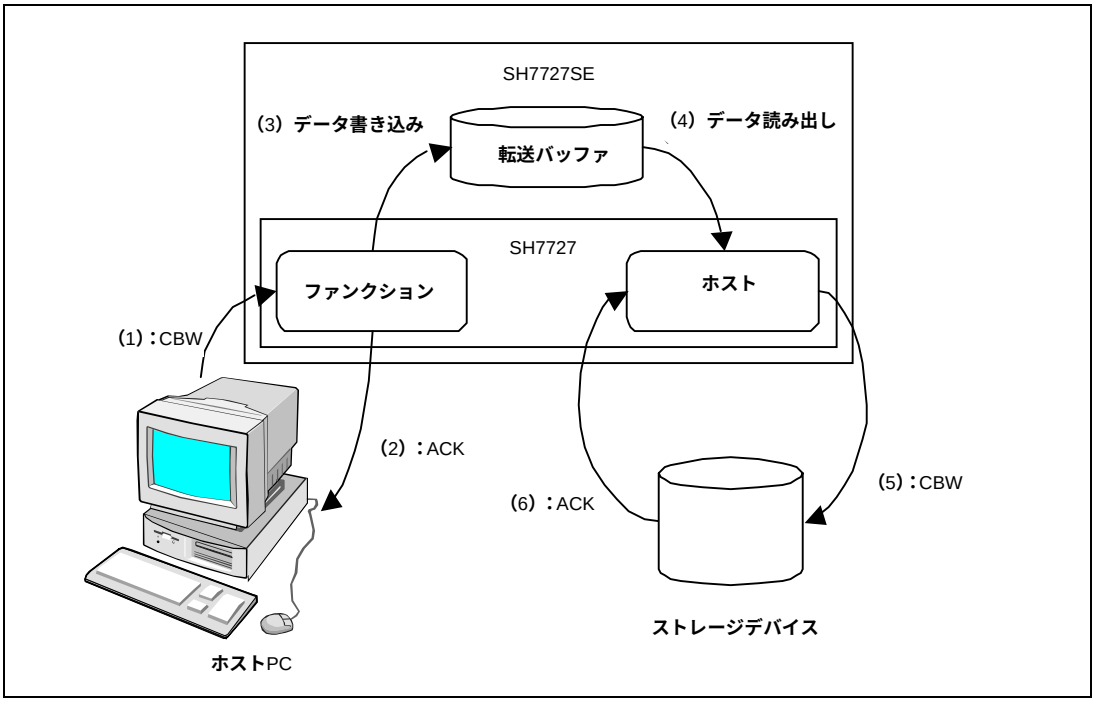


図 4.19-1 ストレージクラス連携図

CBW の転送が完了すると、データトランスポート又は CSW の処理にかかります。

データトランスポートがイン転送の場合、ホスト PC は SH7727 ファンクションに対して IN トークンを発行しデータを要求してきます。しかし SH7727 ファンクションにはホスト PC に送るデータがないので“ NAK ”を返します。その際“ 割り込み ”が発生するので割り込み処理ルーチンで処理を行います。その処理は

- SH7727ホストを使用して転送するデータをストレージデバイスから転送バッファに格納
- SH7727ファンクションを使用して転送バッファに格納したデータをホストPCに転送
- 割り込みイネーブルビットの変更

の 3 つです。SH7727 ファンクションにはデータ転送（バルクイン）に使用する FIFO バッファが 2 面あり、このバッファに空きがある場合、“ 割り込み ”が発生するので割り込み処理ルーチンで順次処理を行います。割り込みは CSW を送信するまで続きます。

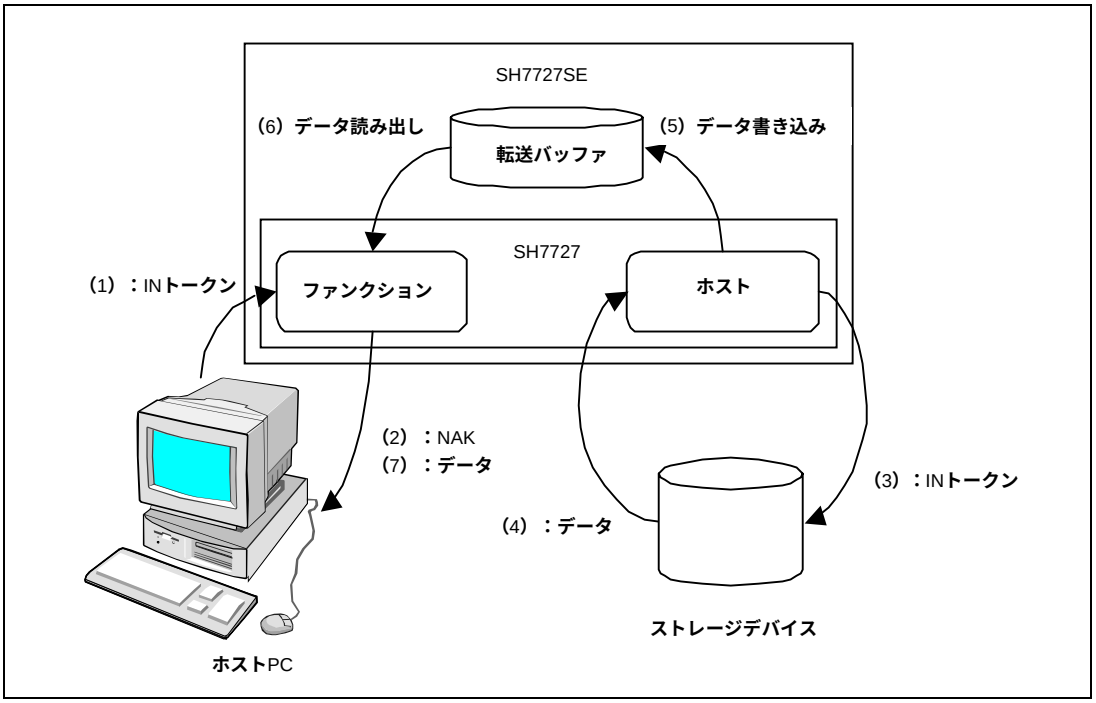


図 4.19-2　**ストレージクラス連携図**

データトランスポートがアウト転送の場合、ホスト PC は SH7727 ファンクションに対して OUT トークンとそれに続く転送データを発行します。SH7727 ファンクションにはまだ受信データがないので、ホスト PC からのデータを受信し、“ ACK ”を返答します。その際“ 割り込み ”が発生するので割り込み処理ルーチンで

- 受信したデータをSH7727ファンクションから転送バッファに格納
- 転送バッファに格納してあるデータをSH7727ホストを使用してストレージデバイスに転送

の 2 つを順次行います。

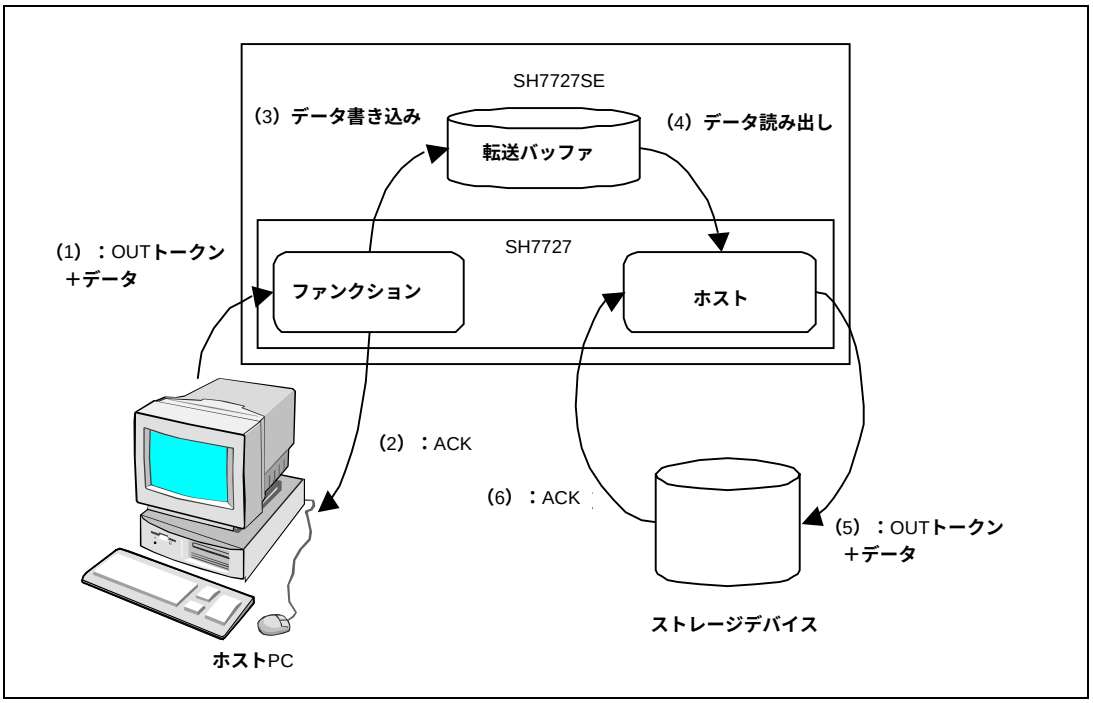


図 4.19-3　**ストレージクラス連携図**

データトランスポートがない場合、またはアウト方向のデータトランスポートが終了した場合、ホスト PC は SH7727 ファンクションに対して IN トークンを発行し、CSW を要求してきます。しかし SH7727 ファンクションにはホスト PC に送るデータがないので、“ NAK ”を返します。その際”割り込み”が発生するので、割り込み処理ルーチンで

- SH7727ホストを使用してCSWをファンクションから転送バッファに格納
- SH7727ファンクションを使用して転送バッファ内のCSWをホストPCに転送

の 2 つを順次行います。

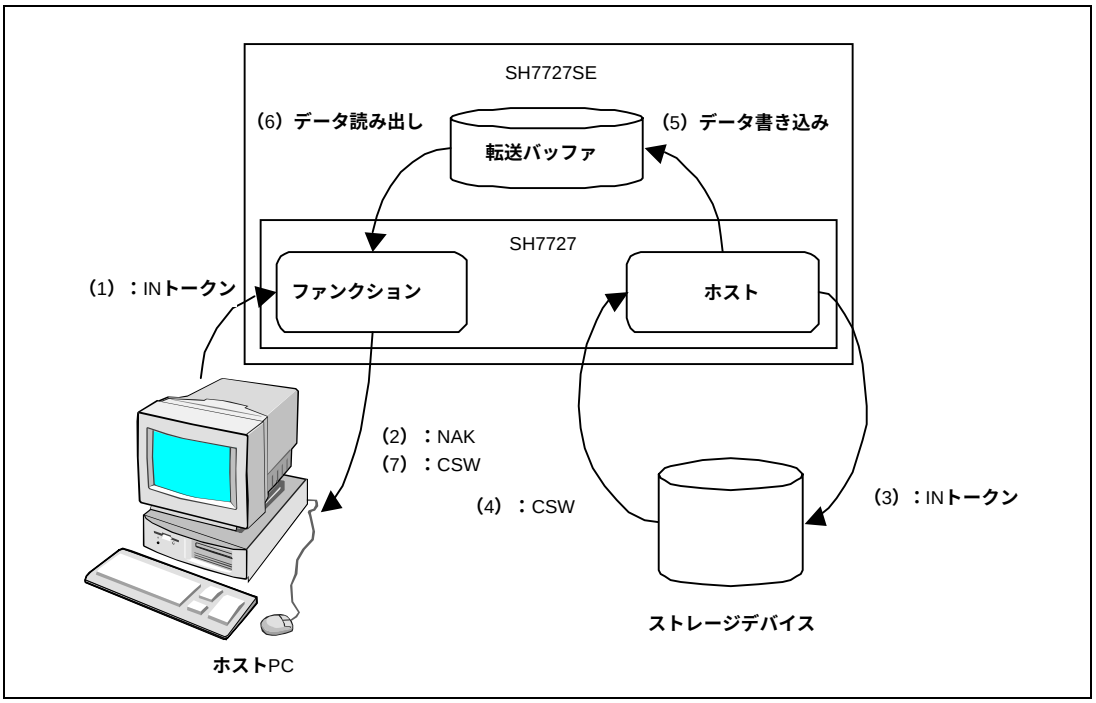


図 4.19-4　**ストレージクラス連携図**

なお､データトランスポートがイン転送で SH7727 ホストがストレージデバイスに転送するデータを要求してストレージデバイスから“ STALL ”が送られてきた場合、ホスト PC とストレージデバイスに対してそれぞれ個別の動作を行います。

ホスト PC に行う処理は

- 0x00データをCBW内で要求したデータ数と同じバイト数ホストPCに送信する
- CSWはCBWを元に作成し、CSWのステータスコードを0x01にしてホストPCに送信する

の 2 つです。SH7727 ファンクションにはデータ転送（バルクイン）に使用する FIFO バッファが 2 面あり、このバッファに空きがある場合、“ 割り込み ”が発生するので割り込み処理ルーチンで順次処理を行います。割り込みは CSW を送信するまで続きます。

一方、ストレージデバイスに行う処理としては

- Clear Featureを発行しストール状態を解除する
- INトークンをストレージデバイスに発行しストレージデバイスからCSWを受信する

の 2 つです。SH7727 ホストはストレージデバイスを制御して、次の CBW に備える必要があります。

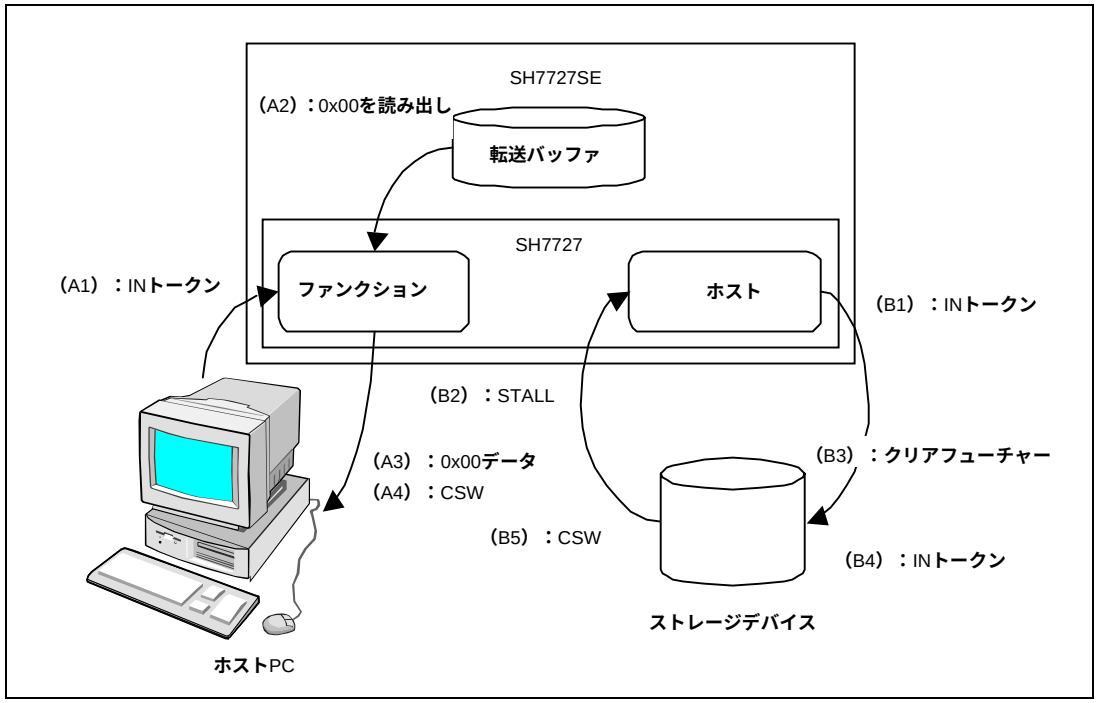


図 4.19-5　**ストレージクラス連携図**

5. サンプルプログラムの動作

この章ではサンプルプログラムの動作を、SH7727 の USB ファンクションモジュールと USB ホストモジュールの動作と関連付けて説明します。

5.1 メインループ

マイコンがリセット状態になると、CPU の内部状態と内蔵周辺モジュールのレジスタが初期化されます。次にリセット割り込み関数 CallResetException が呼び出されリセット例外処理を行い、関数 SetPowerOnSection を呼び出します。**図 5.1** にリセット割り込み発生から定常状態までのフローチャートを示します。

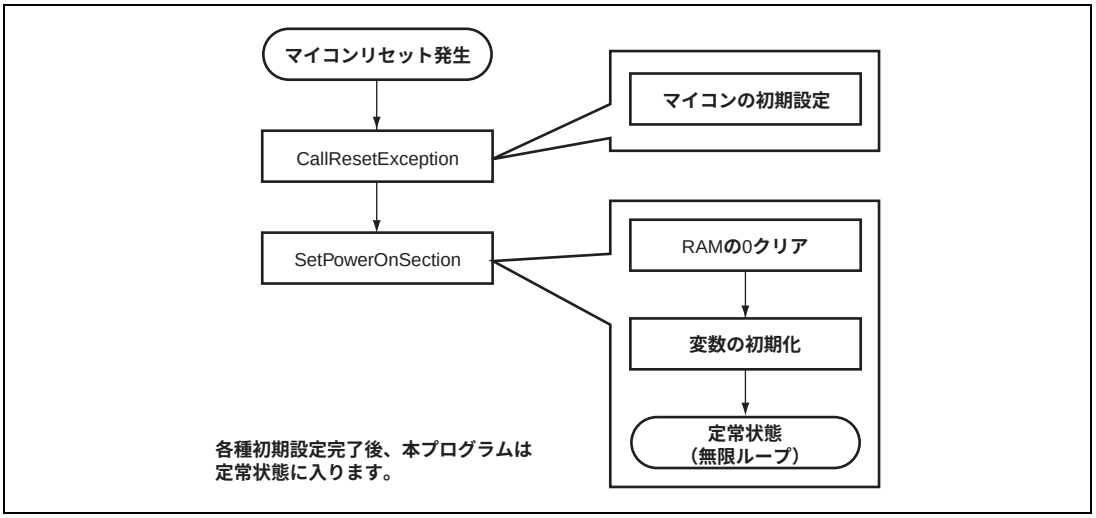


図 5.1 メインループ

5.2 割り込みの種類

本サンプルプログラムで使用する割り込みは割り込みフラグレジスタ 0（USBIFR0）、1（USBIFR1）、HcInterruptStatus レジスタによって示される計 10 種類です。割り込み要因が発生すると、割り込みフラグレジスタの対応するビットに 1 がセットされ、CPU に対して USBFI0 割り込みを要求します。サンプルプログラムでは、この割り込み要求によって割り込みフラグレジスタをリードし、それに対応する動作を行います。**図** 5.2 に割り込みフラグレジスタと、USB 通信の関係を示します。

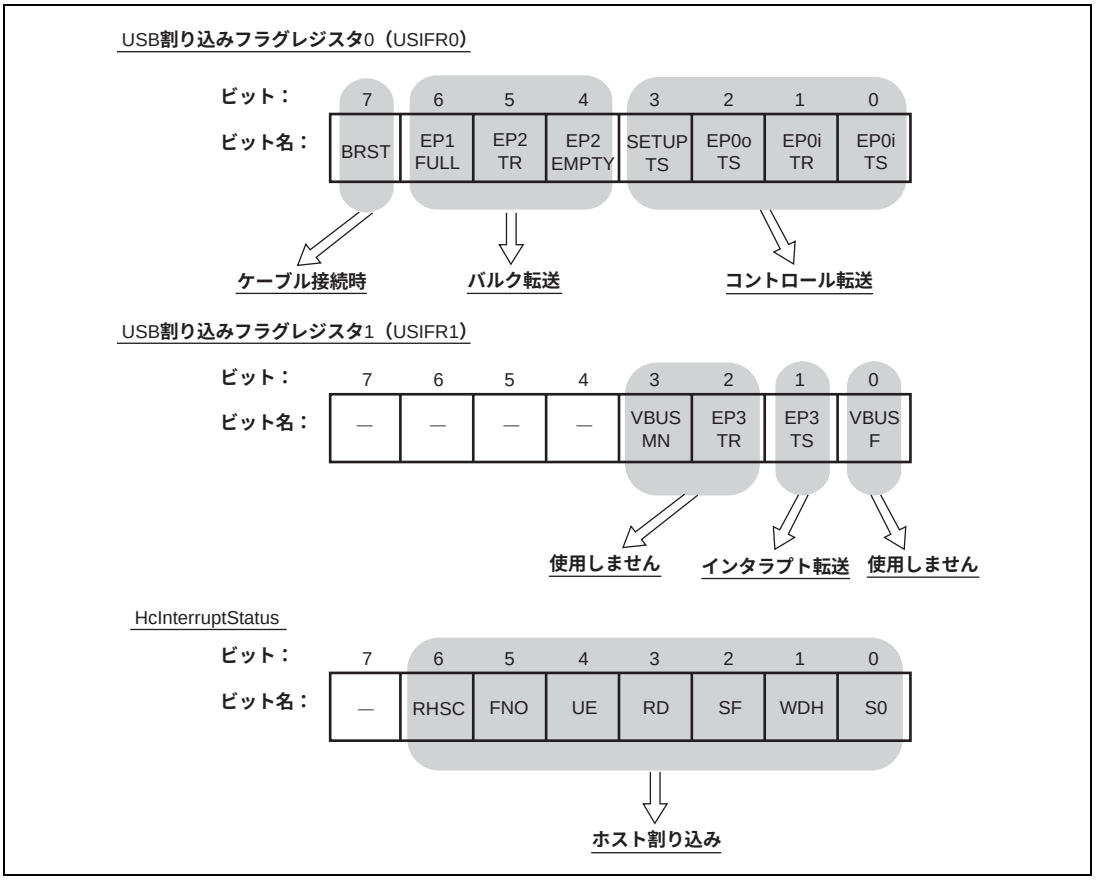


図 5.2 **割り込みフラグの種類**

5.2.1 SH7727 ファンクションの分岐方法

サンプルプログラムでは、USB ファンクションモジュールからの割り込みの種類によって転送方式を決定しています。各転送方式への分岐は、UsbMain.c の BranchOfInt が実行します。**表 5.1** に割り込みの種類と、BranchOfInt が呼び出す関数の関係を示します。

表 5.1 割り込みの種類と分岐先関数

レジスタ名	ビット	ビット名	呼び出す関数名
USBIFR0	0	EP0i TS	ActControlInOut
	1	EP0i TR	ActControlInOut
	2	EP0o TS	ActControlInOut
	3	SETUP TS	ActControl
	4	EP2 EMPTY	ActBulkIn
	5	EP2 TR	ActBulkInReady
	6	EP1 FULL	ActBulkOut
	7	BRST	ActBusReset
USBIFR1	1	EP3 TS	ActInterruptIn

EP0i TS と EP0o TS 割り込みは、コントロールイン、アウト転送の両方で使用します。したがって、コントロール転送の方向とステージを管理するために、サンプルプログラムは TRANS_IN、TRANS_OUT、WAIT の 3 つのステートを持っています。詳細は、**「5.4 SH7727 ファンクションのコントロール転送」**をご覧ください。

5.3 ケーブル接続時（BRST）割り込み

USB ファンクションモジュールを USB ケーブルを使い、ホスト PC に接続した際に発生します。ソフトウェアはマイコンの初期設定完了後、汎用専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 PC はデバイスが接続された事を認識します。（**図** 5.3 参照）

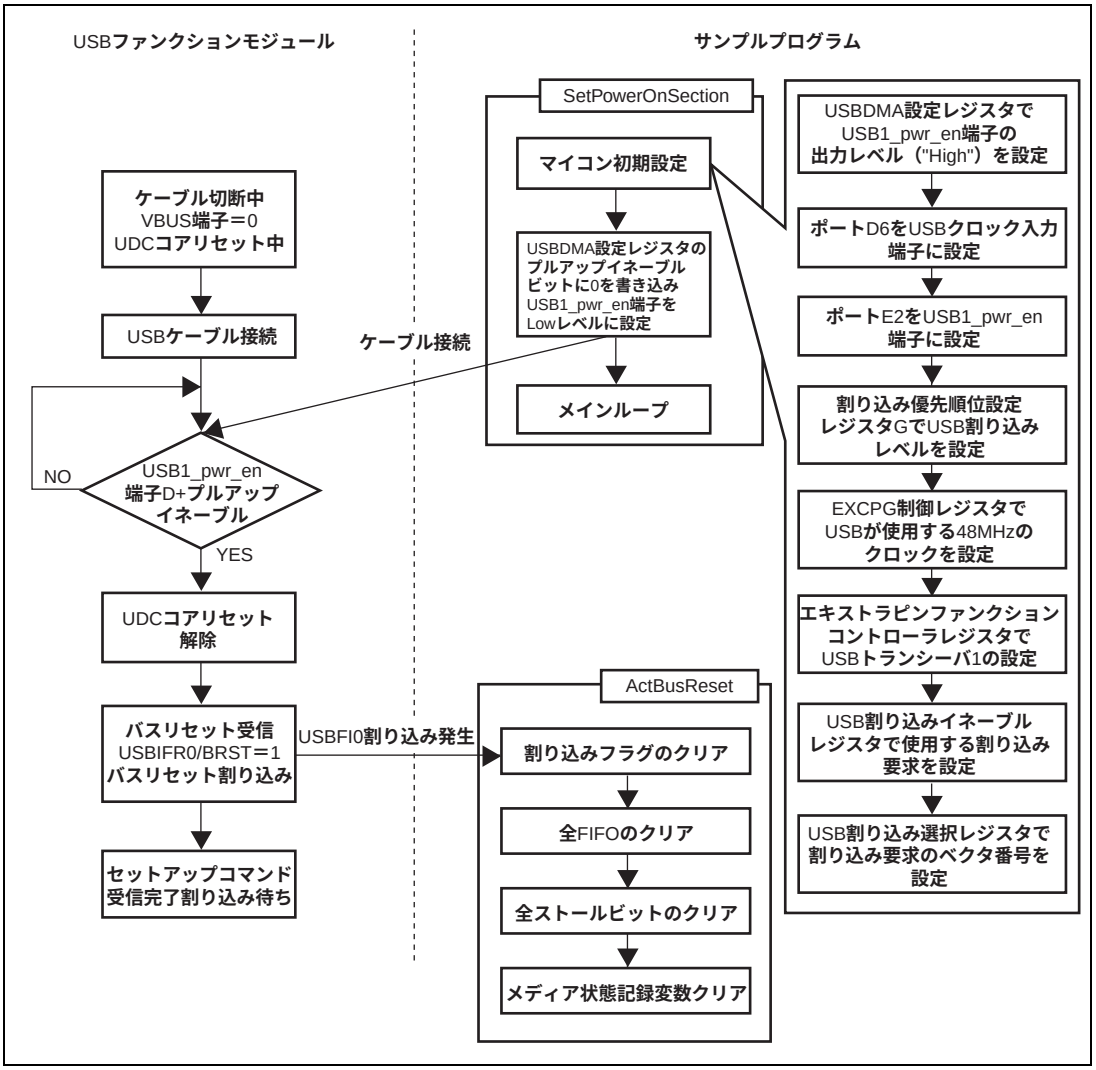


図 5.3 **ケーブル接続時割り込み**

5.4 SH7727 ファンクションのコントロール転送

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** 5.4 参照）

データステージにおいて、ホスト PC から SH7727 ファンクションへデータ転送する場合がコントロールアウト転送、反対の場合がコントロールイン転送です。

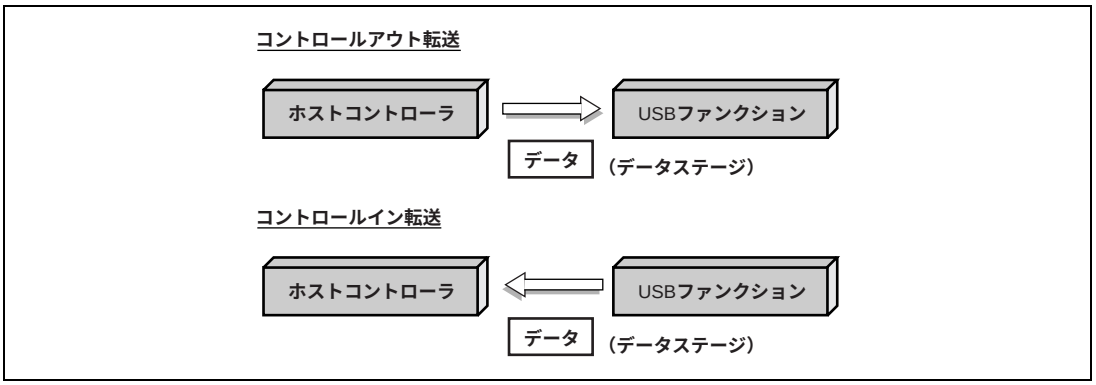


図 5.4 **コントロール転送**

コントロール転送は、セットアップ、データ（ない場合もある）、ステータスの 3 つのステージで構成されます（**図** 5.5 参照）。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** 5.1 参照）。このため、現在イン･アウトどちらのコントロール転送が行われているかをファームウエアがステートによって管理し（**図** 5.5 参照）、適切な関数を呼び出す必要があります。データステージにおけるステート（TRANS_IN、TRANS_OUT）は、セットアップステージで受信するコマンドによって決定します。

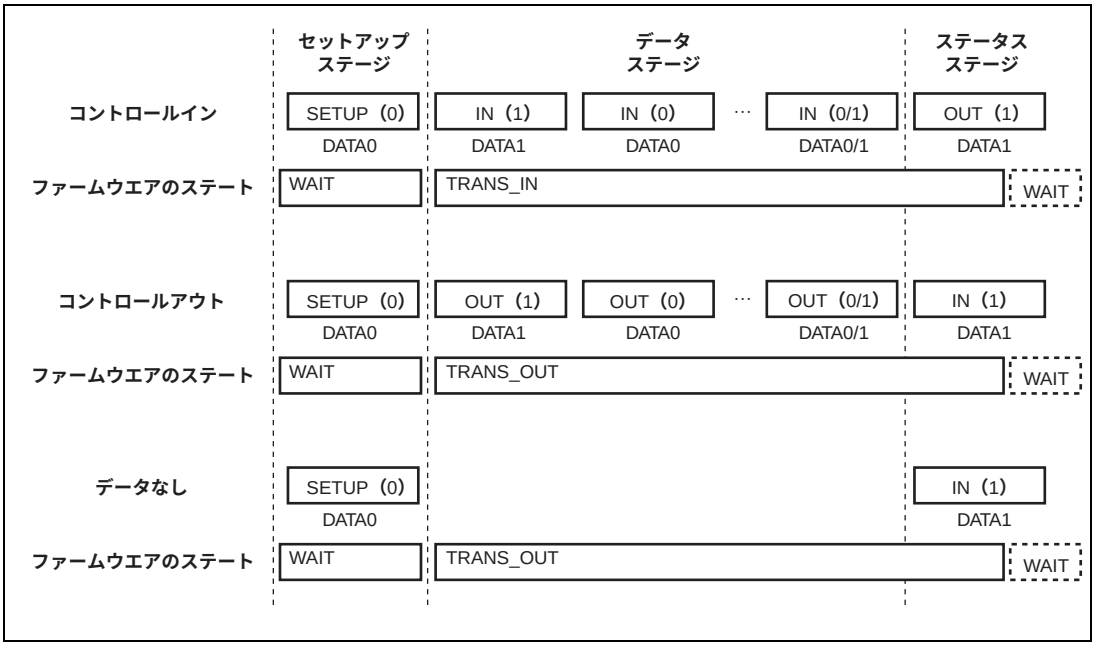


図 5.5　**コントロール転送における各ステージ**

5.4.1　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 PC と SH7727 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。

TRANS_IN となるコマンド：GetDescriptor（標準コマンド）

図 5.6-1 にセットアップステージにおけるサンプルプログラムの動作を示します。図の左側は、USB ファンクションモジュールの動作を示しています。

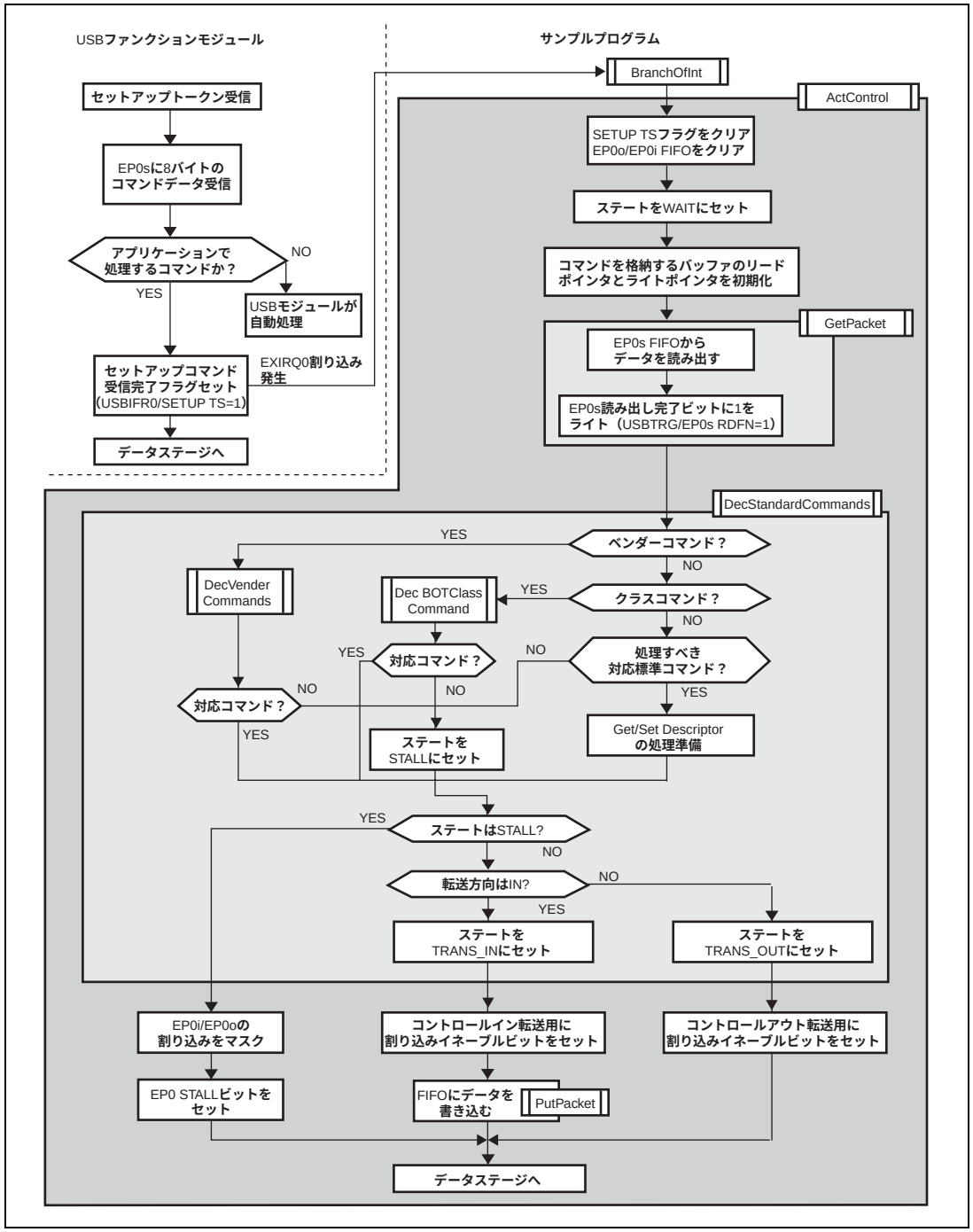


図 5.6-1　**セットアップステージ**

5.4.2 データステージ

データステージでは、ホスト PC と SH7727 ファンクションがデータの送受信を行います。ファームウエアのステートは、セットアップステージで行ったコマンドのデコード結果によって、コントロールイン転送の場合は TRANS_IN に、コントロールアウト転送の場合は TRANS_OUT になります。**図** 5.6-2、**図** 5.6-3 にコントロール転送のデータステージにおけるサンプルプログラムの動作を示します。

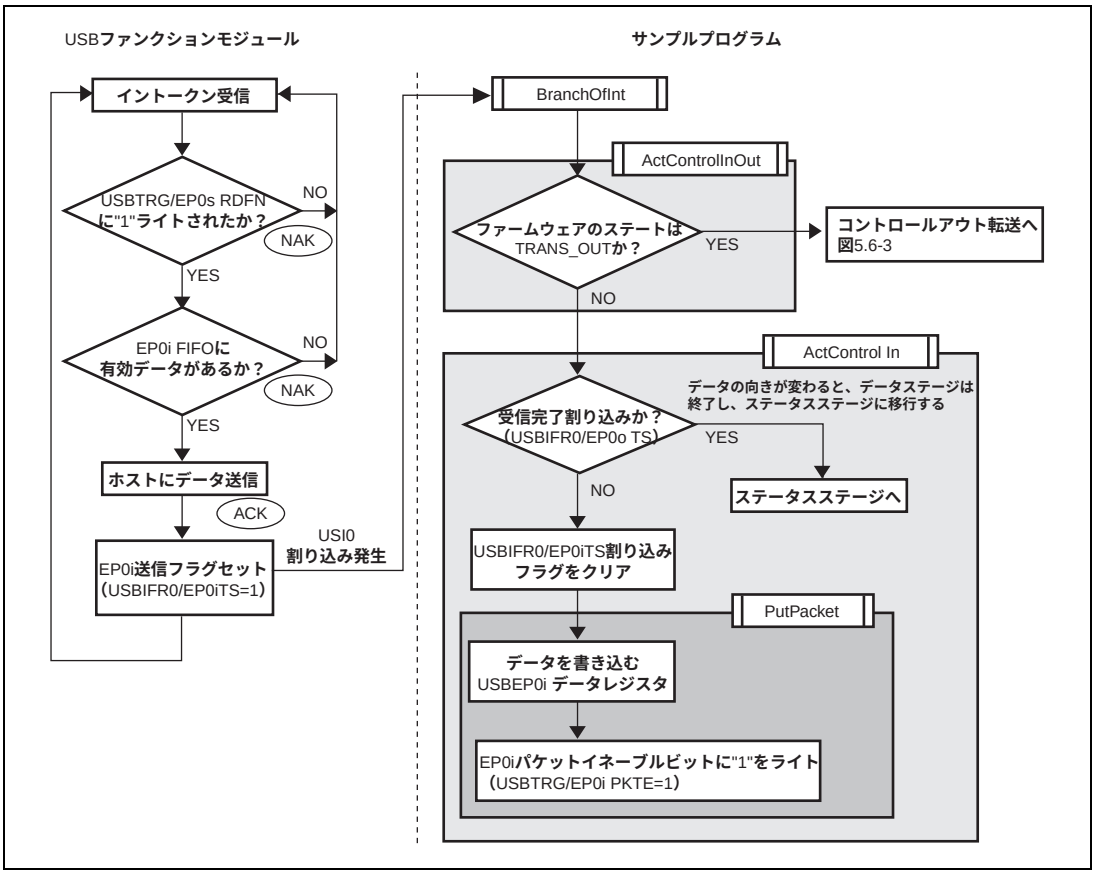


図 5.6-2 **データステージ（コントロールイン転送）**

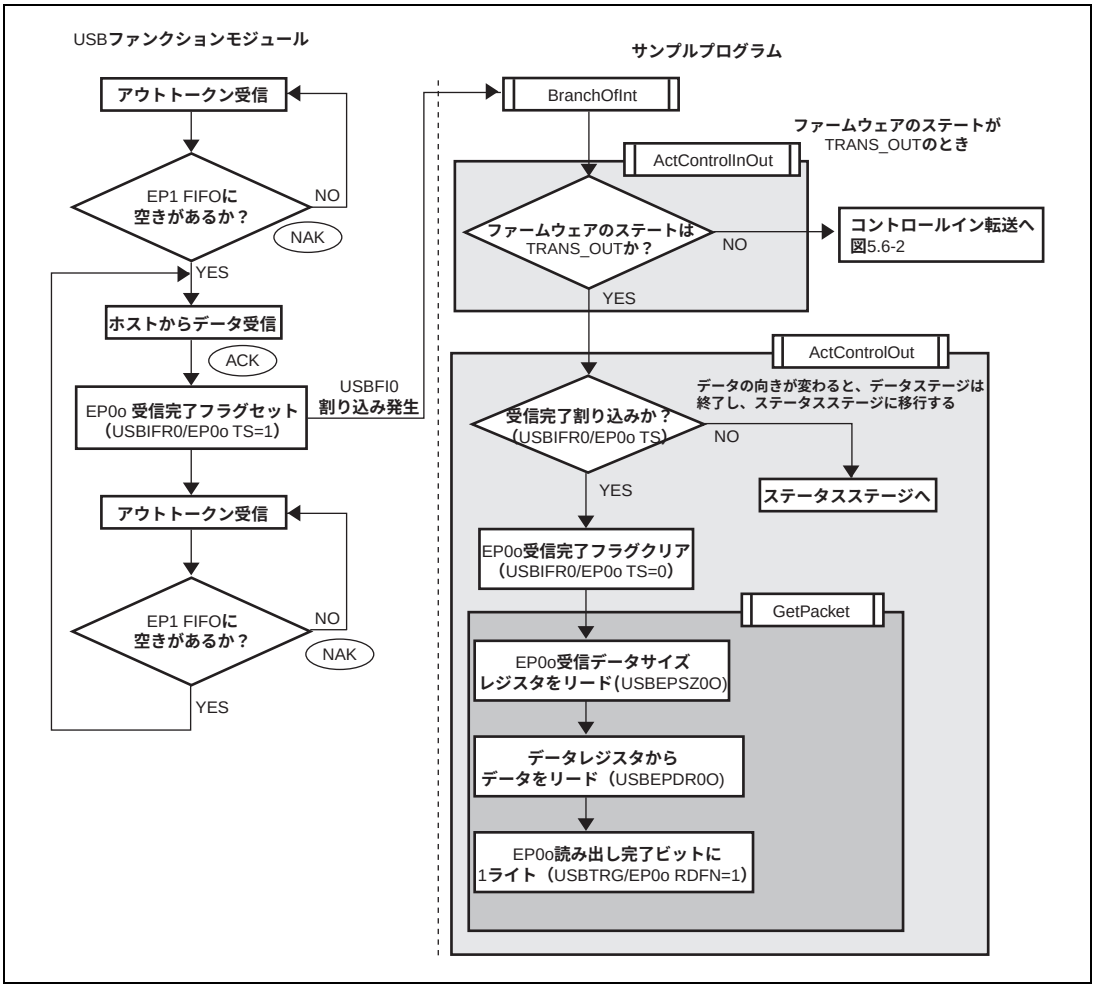


図 5.6-3　**データステージ（コントロールアウト転送）**

5.4.3 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 PC 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 PC からのイントークンによってステータスステージが開始されます。

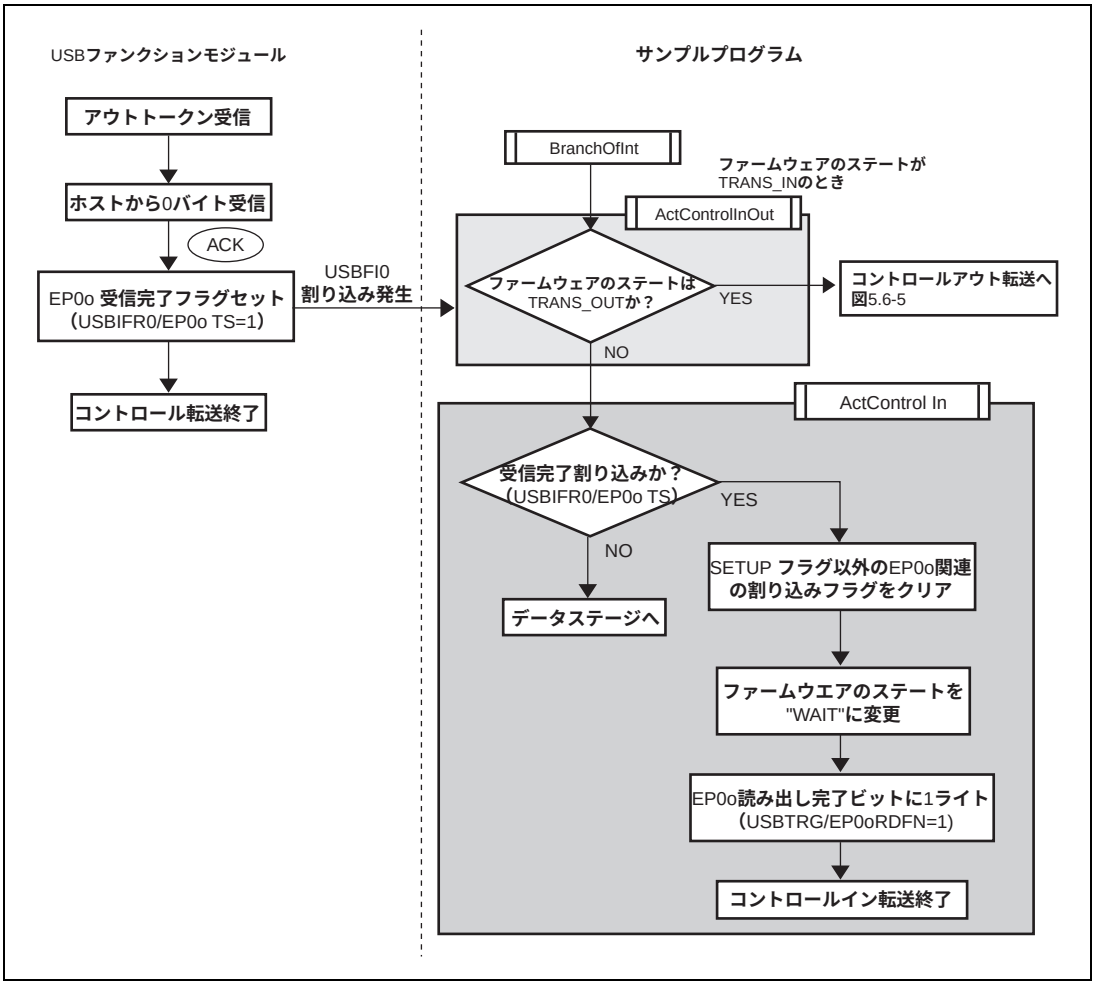


図 5.6-4 **ステータスステージ（コントロールイン転送）**

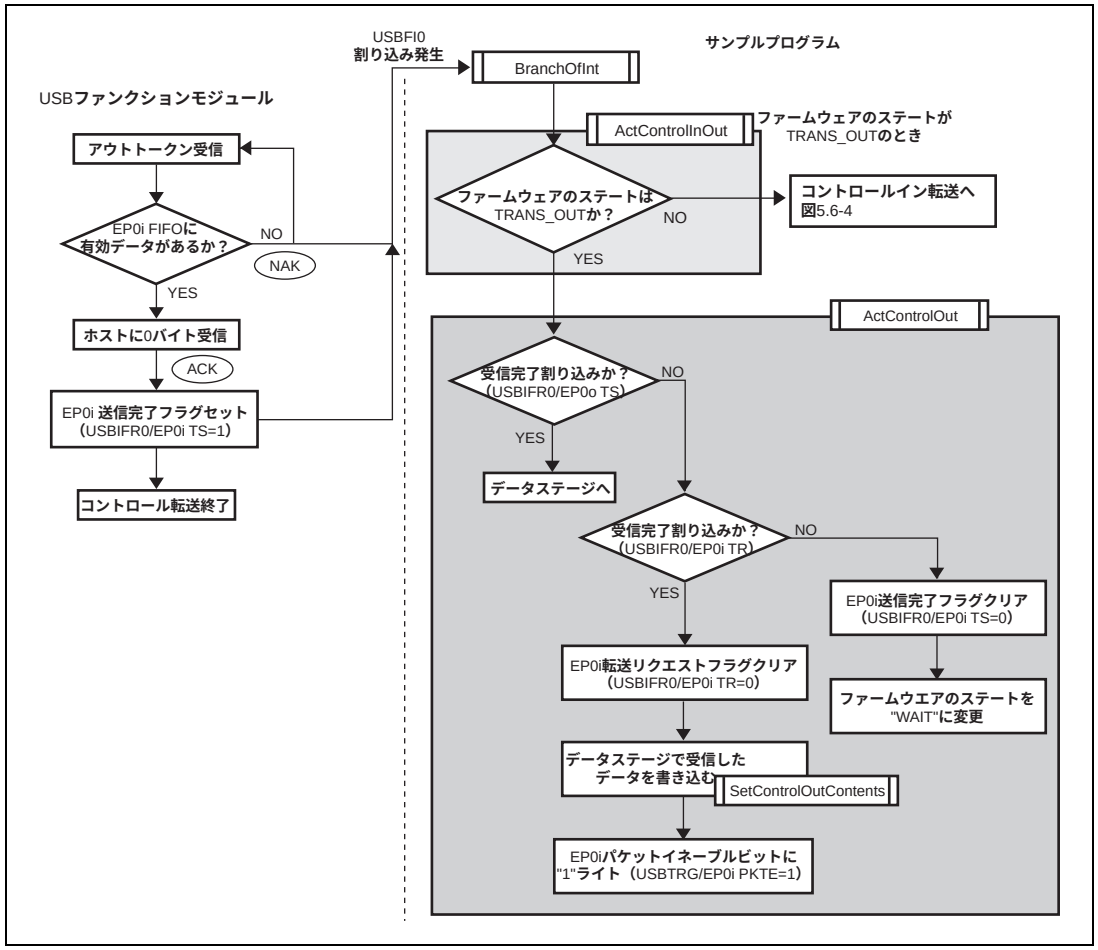


図 5.6-5　**ステータスステージ（コントロールアウト転送）**

5.5 SH7727 ファンクションのバルク転送

バルク転送には、割り込みフラグレジスタのビット 4～6 を使用します。バルク転送もデータを送信する向きによって、2 つに分けることができます。（**図** 5.7 参照）

ホスト PC から SH7727 ファンクションへデータ転送する場合をバルクアウト転送、反対の場合をバルクイン転送と呼びます。

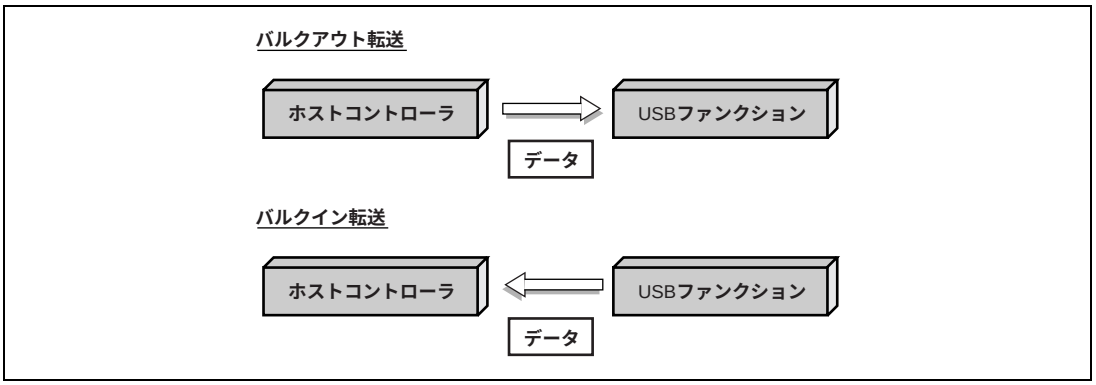


図 5.7 **バルク転送**

USB Mass Storage Class の Bulk-Only Transport は、バルクイン転送とバルクアウト転送で構成されています。

Bulk-Only Transport は、「コマンドトランスポート（CBW）」「データトランスポート」（ない場合もある）「ステータストランスポート（CSW）」の 2 ないし 3 つのステージで構成されます（**図** 5.8 参照）。また、データトランスポートは、複数のバストランザクションで構成されます。

Bulk-Only Transport では、コマンドトランスポート（CBW）はバルクアウト転送。ステータストランスポート（CSW）はバルクイン転送。データトランスポートはデータを送信する向きによってバルクイン転送･バルクアウト転送どちらかの転送が行われます。

データトランスポートでバルクイン･バルクアウトどちらの転送が行われるかは、コマンドトランスポートで受信する CBW データにより決定します。ファームウエアはデータトランスポートがバルクイン転送･バルクアウト転送どちらの転送が行われるかをステート（TRANS_IN、TRANS_OUT）で管理し（**図** 5.8 参照）、適切な関数を呼び出す必要があります。

また、データトランスポートからステータストランスポートへのステージの遷移は、ホスト PC がリクエストしたデータトランスポートでの予定データ長のデータを送信または受信する事で、ステータストランスポートへステージが遷移します。

本サンプルソフトではホスト PC からバルクアウト転送で送られてきた「CBW」「データトランスポート」を受信し、共有メモリに保存します。また、バルクイン転送では共有メモリに送信可能な「データトランスポート」「CSW」があればホスト PC に送り、無ければ USB ファンクションモジュール（ハードウェア）が NAK を送ります。

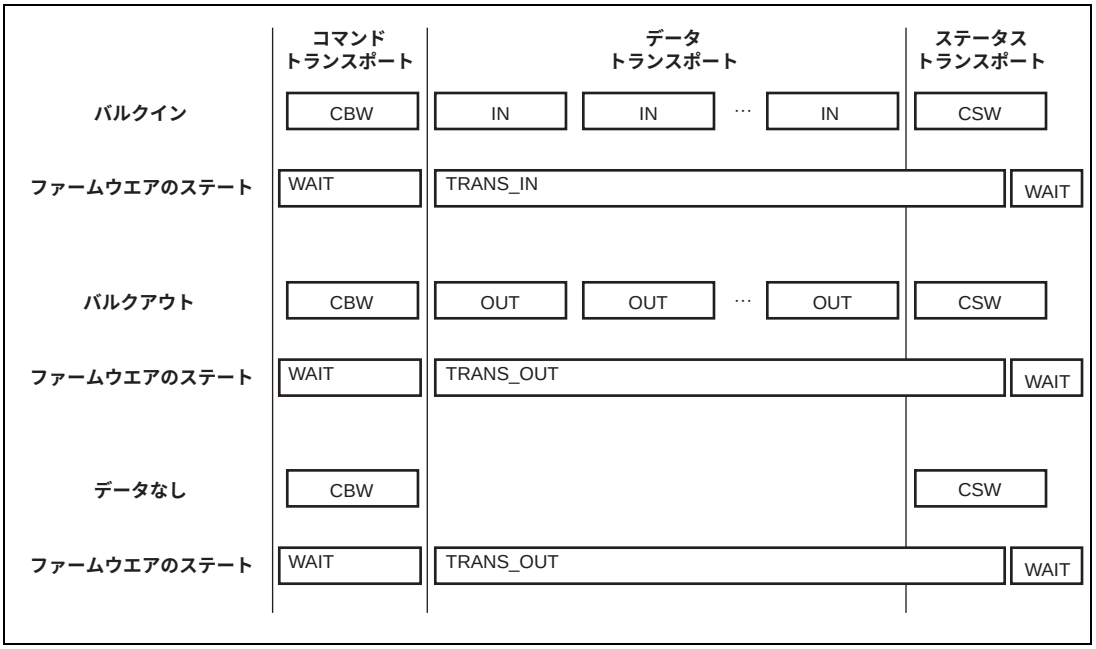


図 5.8　Bulk-Only Transport **における各ステージ**

5.5.1　バルクアウト転送

バルクアウト転送では、ホスト PC から SH7727 ファンクションへの転送データを受信します。バルクアウト転送を使用するのは CBW データ受信時、データトランスポート受信時の 2 つです。

CBW データ受信時ファームウエアのステートは WAIT 状態で実行されます。データトランスポート受信時ファームウエアのステートは TRANS_OUT 状態で実行されます。

5.5.2　バルクイン転送

バルクイン転送では、SH7727 ファンクションからホスト PC への転送データを送信します。バルクイン転送を使用するのは「データトランスポート送信時」、「CSW 送信時」、「リンクプログラムがストレージデバイスの代わりに 0x00 データを CBW 内で要求したデータ数と同じバイト数ホスト PC に送信する時」の 3 つです。

データトランスポート送信時ファームウエアのステートは TRANS_IN 状態で実行されます。CSW データ送信時ファームウエアのステートは TRANS_IN または TRANS_OUT どちらかの状態で実行されます。リンクプログラムがストレージデバイスの代わりに 0x00 データを送信時する際、ファームウエアのステートは STALL 状態で実行されます。

5.6 SH7727 ファンクションのインタラプト転送

インタラプト転送には、割り込みフラグレジスタ 1（USBIFR1）のビット 1 を使用します。SH7727 ではインタラプト転送のデータを送信する向きは 1 つです。（**図** 5.9 参照）

ホスト PC へ SH7727 ファンクションからデータ転送する場合でインタラプトイン転送と呼びます。

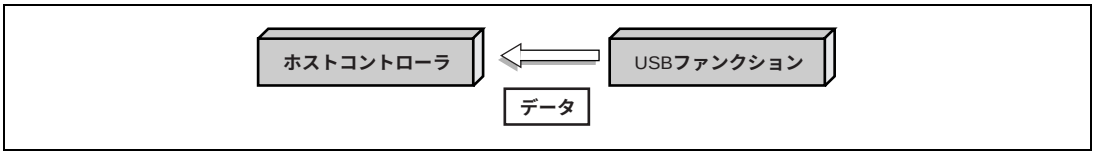


図 5.9 **インタラプト転送**

インタラプト転送では、ホスト PC が SH7727 ファンクションからのデータを受信します。この時ファームウエアのステートは“ WAIT ”又は“ TRANS_IN ”どちらかの状態です。サンプルプログラムのインタラプトイン転送における動作を**図** 5.10 に示します。図の左側は、USB ファンクションモジュールの動作を示しています。

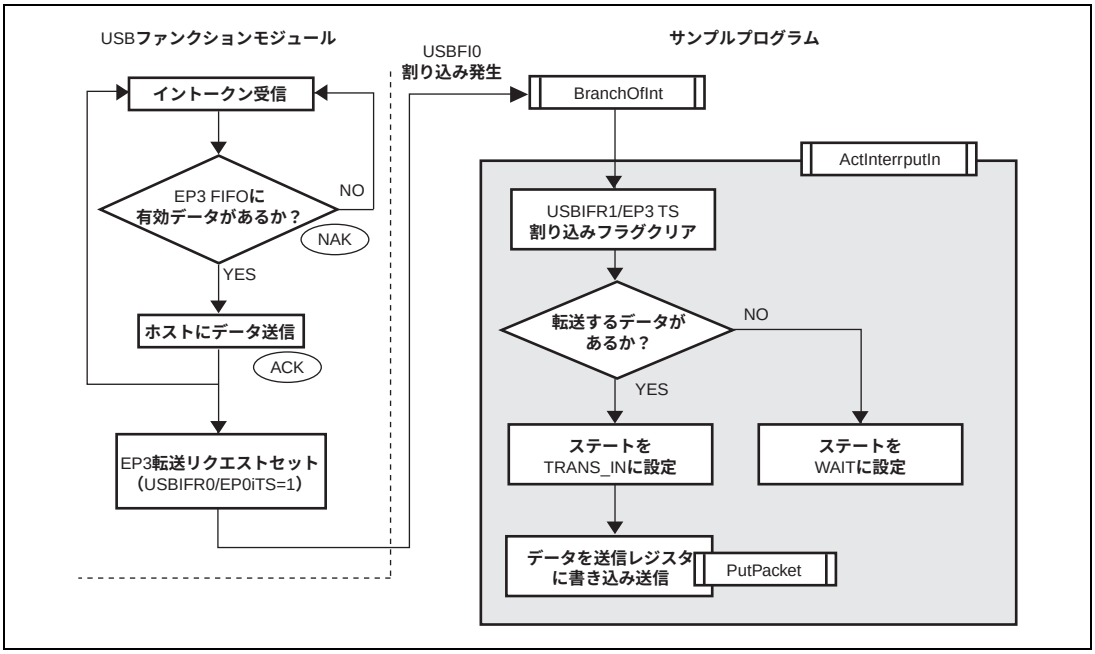


図 5.10 **インタラプトイン転送**

5.7 SH7727 ホストのハブ制御

ハブドライバは、

- 接続されたハブの起動処理
- ダウンストリームポートに接続されたUSBハブまたは、USBファンクションデバイスのリセット
- ハブのダウンストリームポート管理

の 3 つを行います。

図 5.11 にハブドライバ群全体図を示します。

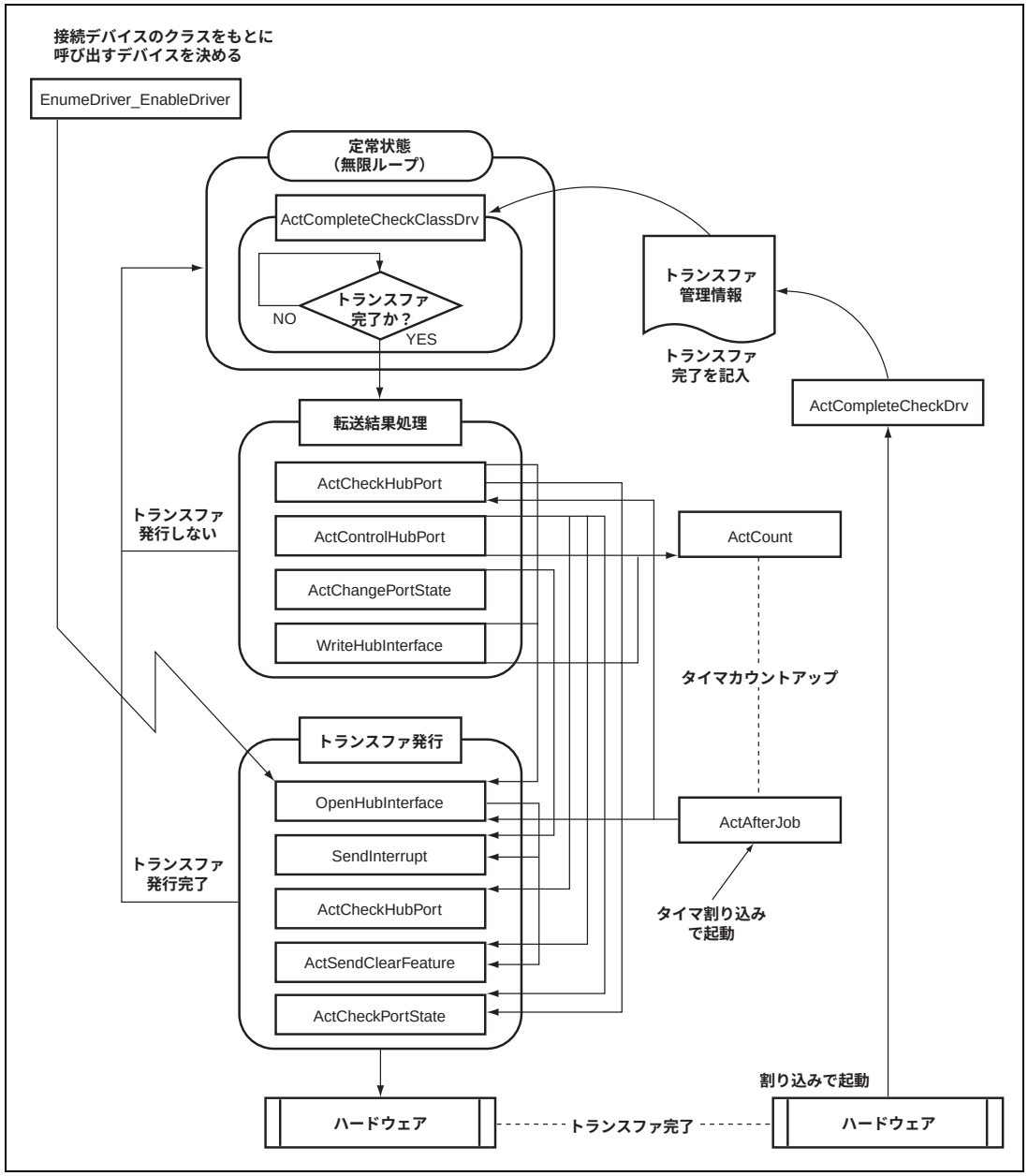


図 5.11　**ハブドライバ群全体図**

USBD 層の関数でデバイスアドレスの設定が完了すると OpenHubInterface 関数が呼び出され、ハブドライバによる起動処理が開始されます。

図 5.12-1 にディスクリプタ取得から起動処理完了までの SH7727 ホストとハブ間で行われる転送を示します。

図 5.12-2 に起動処理完了後からハブに生じた変化をインタラプト転送による変化の通知、通知を受けダウンストリームポートの状態検出、ダウンストリームポート操作までの SH7727 ホストとハブ間で行われる転送を示します。

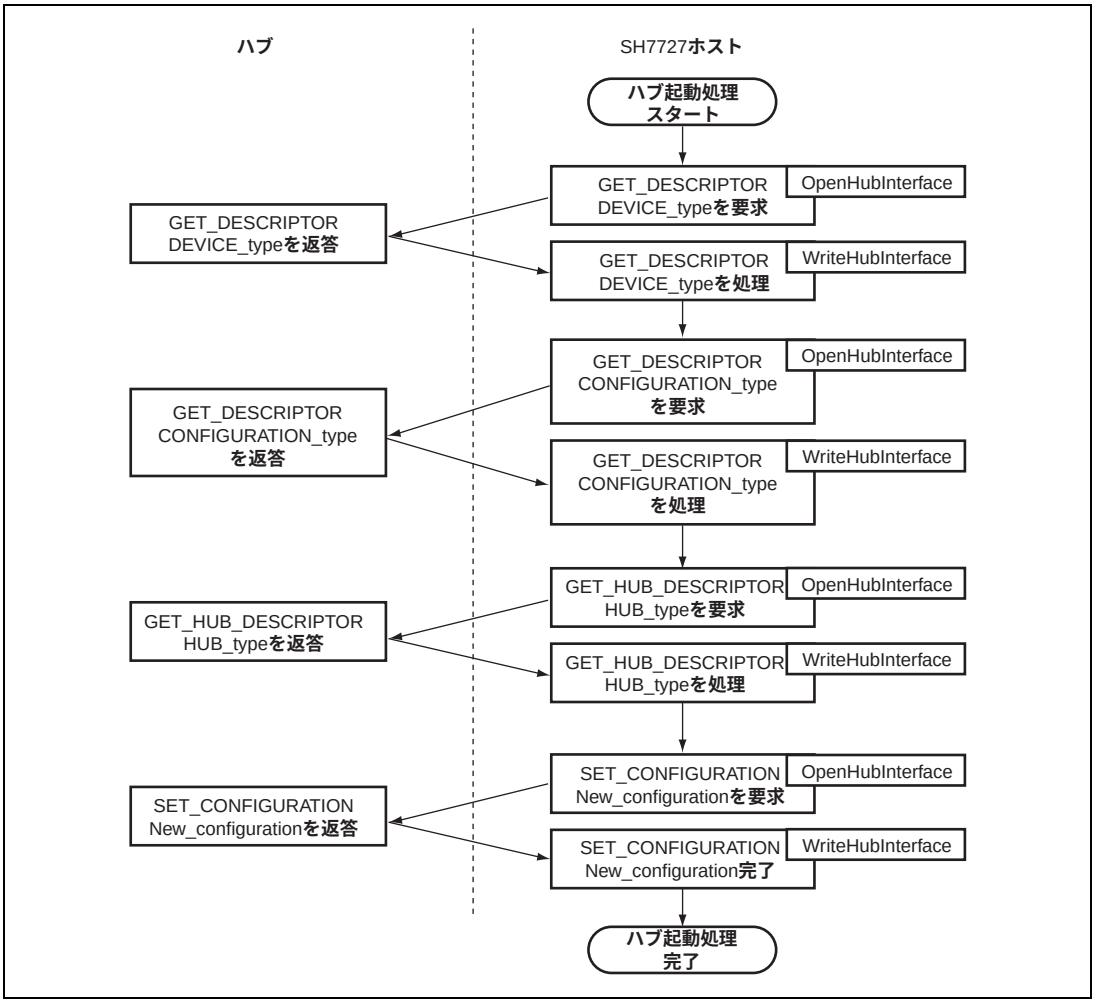


図 5.12-1　SH7727 **ホスト-ハブ転送図** 1

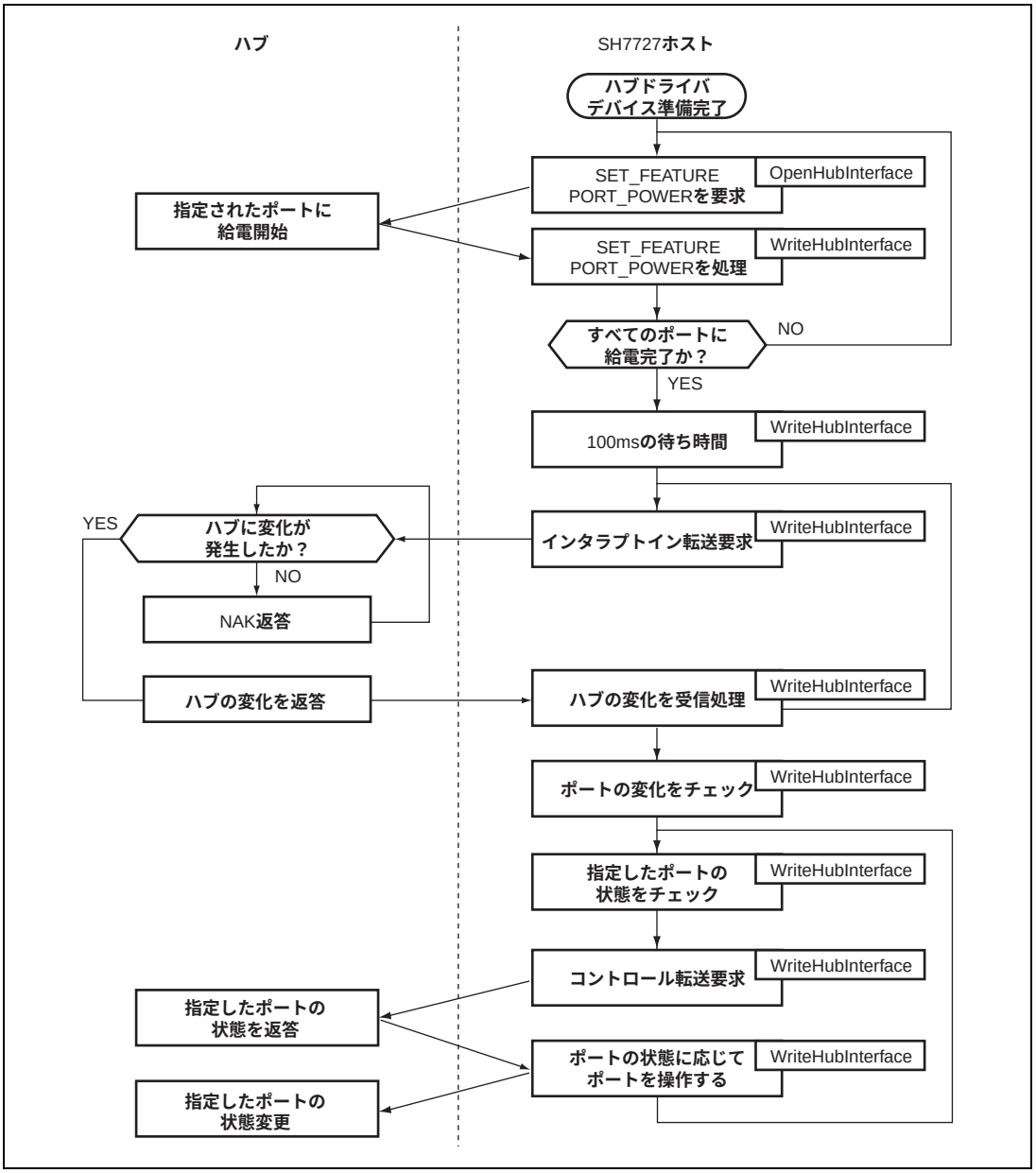


図 5.12-2　SH7727 **ホスト-ハブ転送図** 2

次にトランスファを発行する関数、5 つのフロー図を示します。

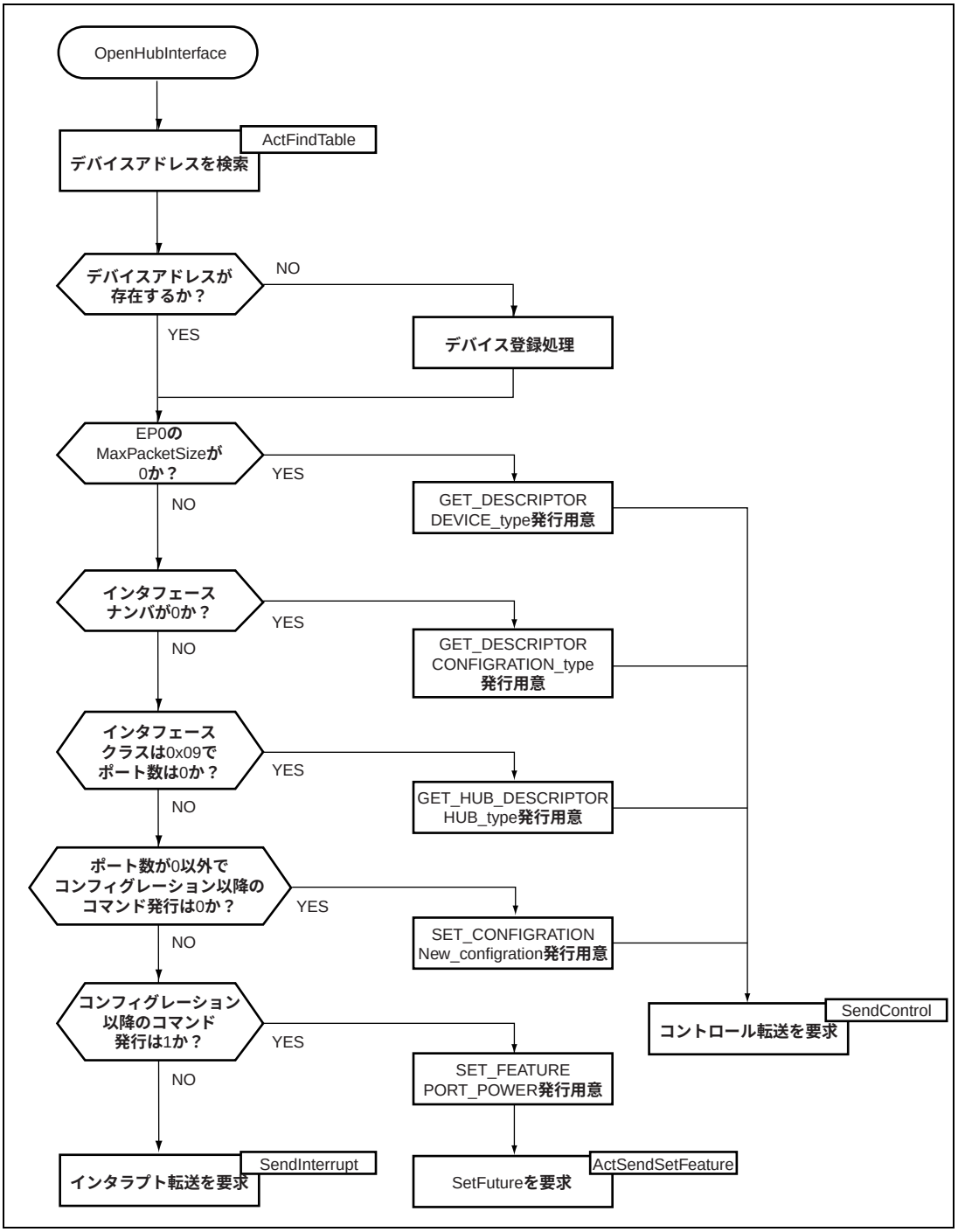


図 5.13-1 **トランスファ発行関数**（1）

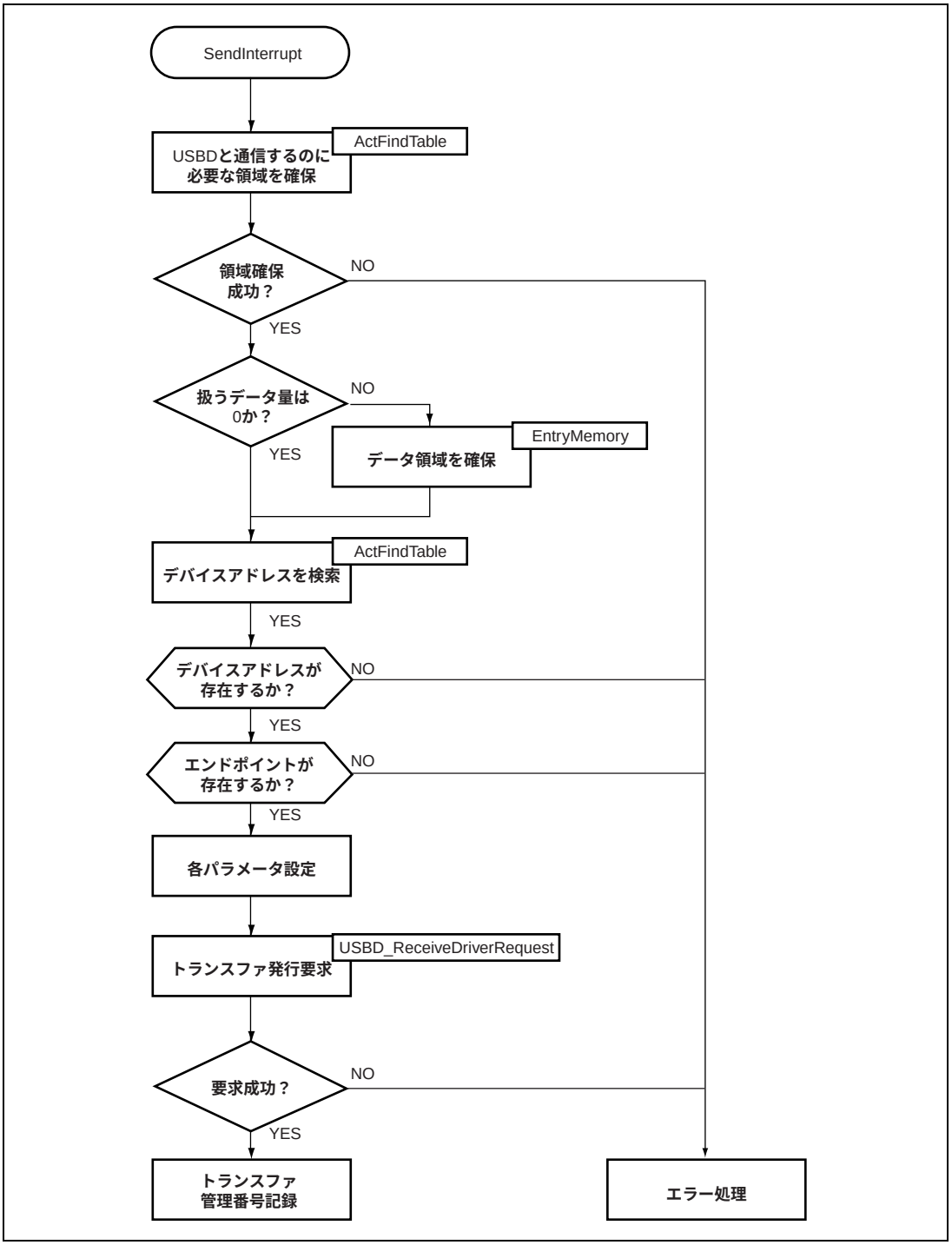


図 5.13-2　**トランスファ発行関数**（2）

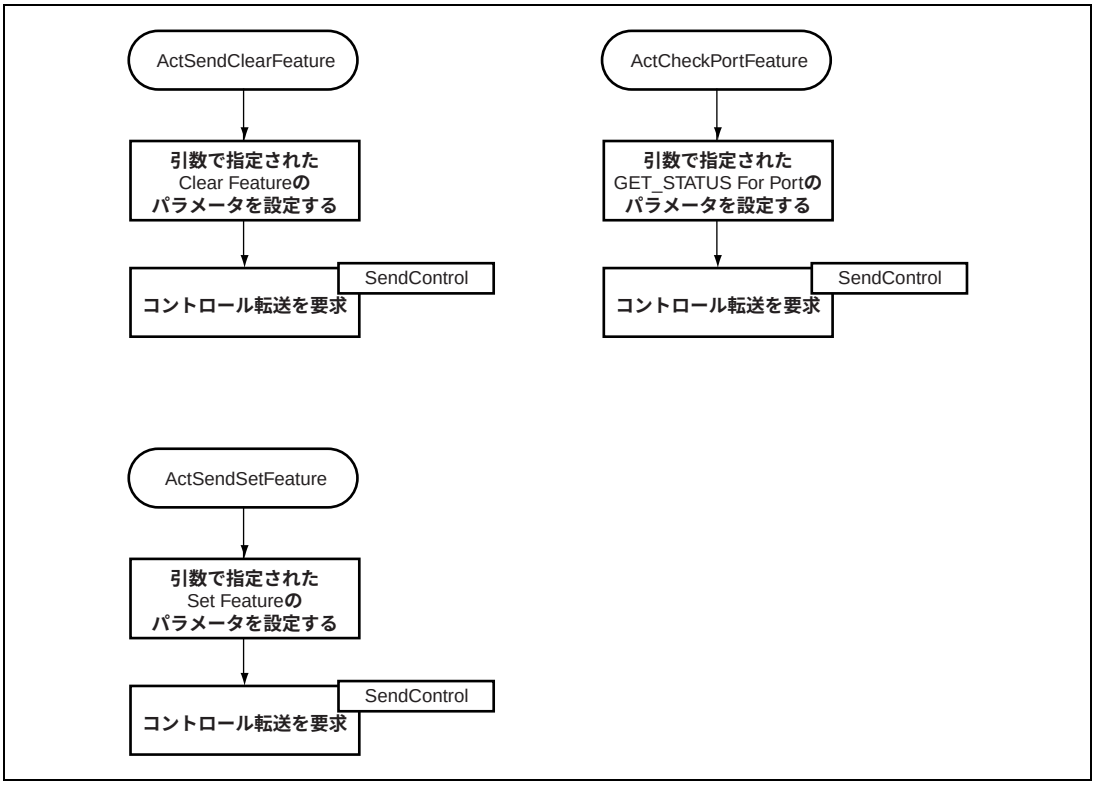


図 5.13-3　**トランスファ発行関数**（3）

次にトランスファの完了後に処理を開始する転送結果処理関数、4 つのフロー図を示します。

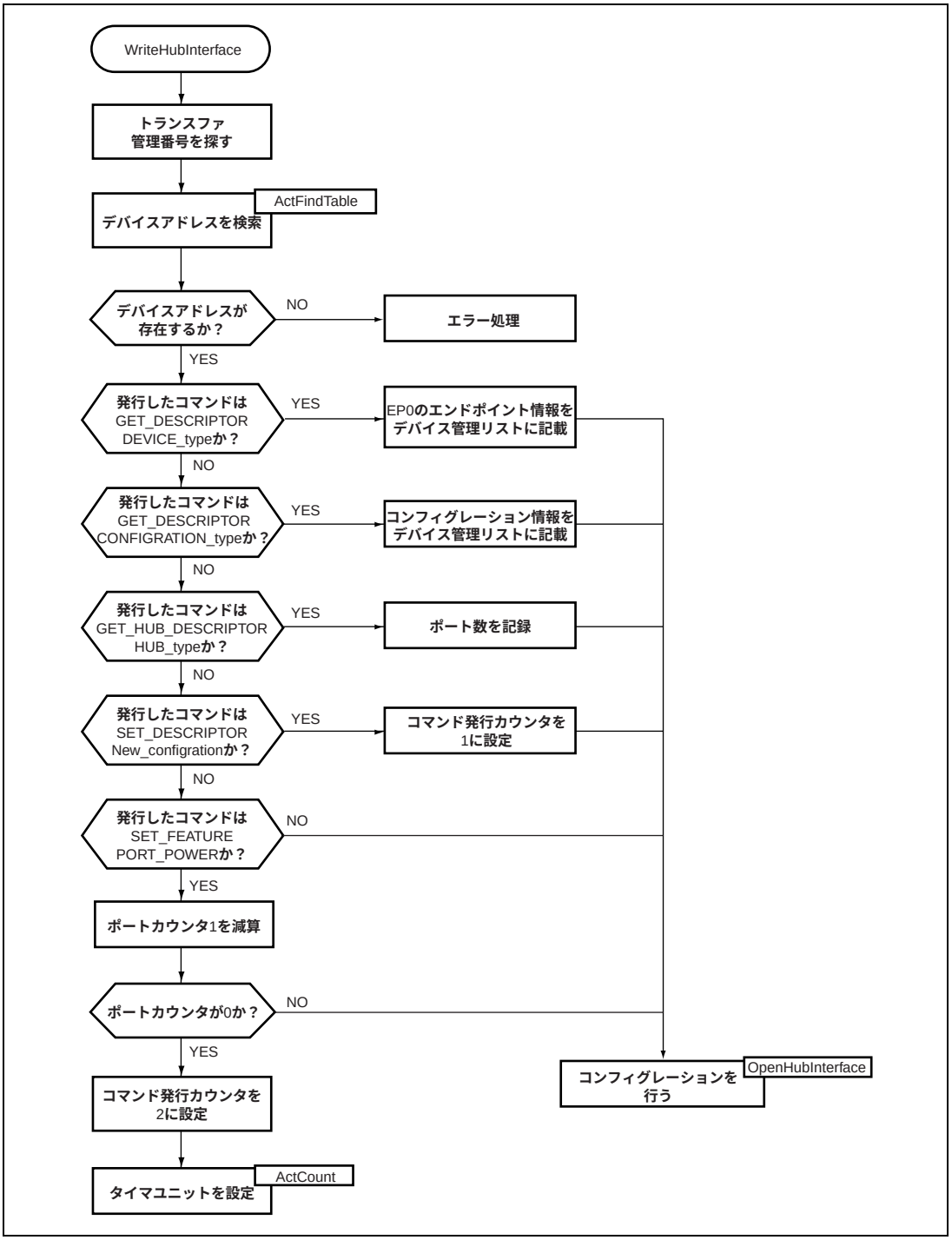


図 5.14-1　**転送結果処理**（1）

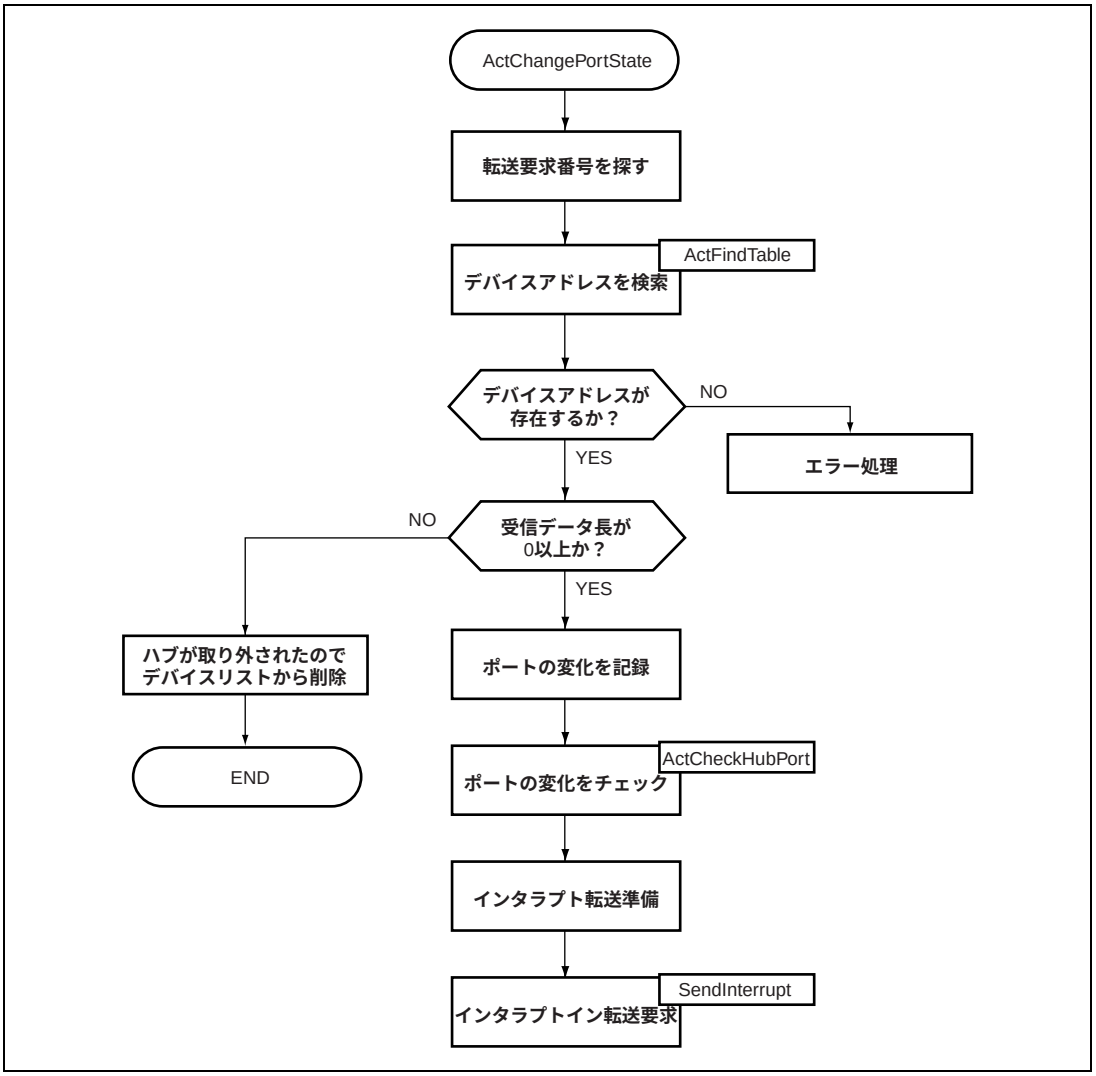


図 5.14-2　**転送結果処理（2）**

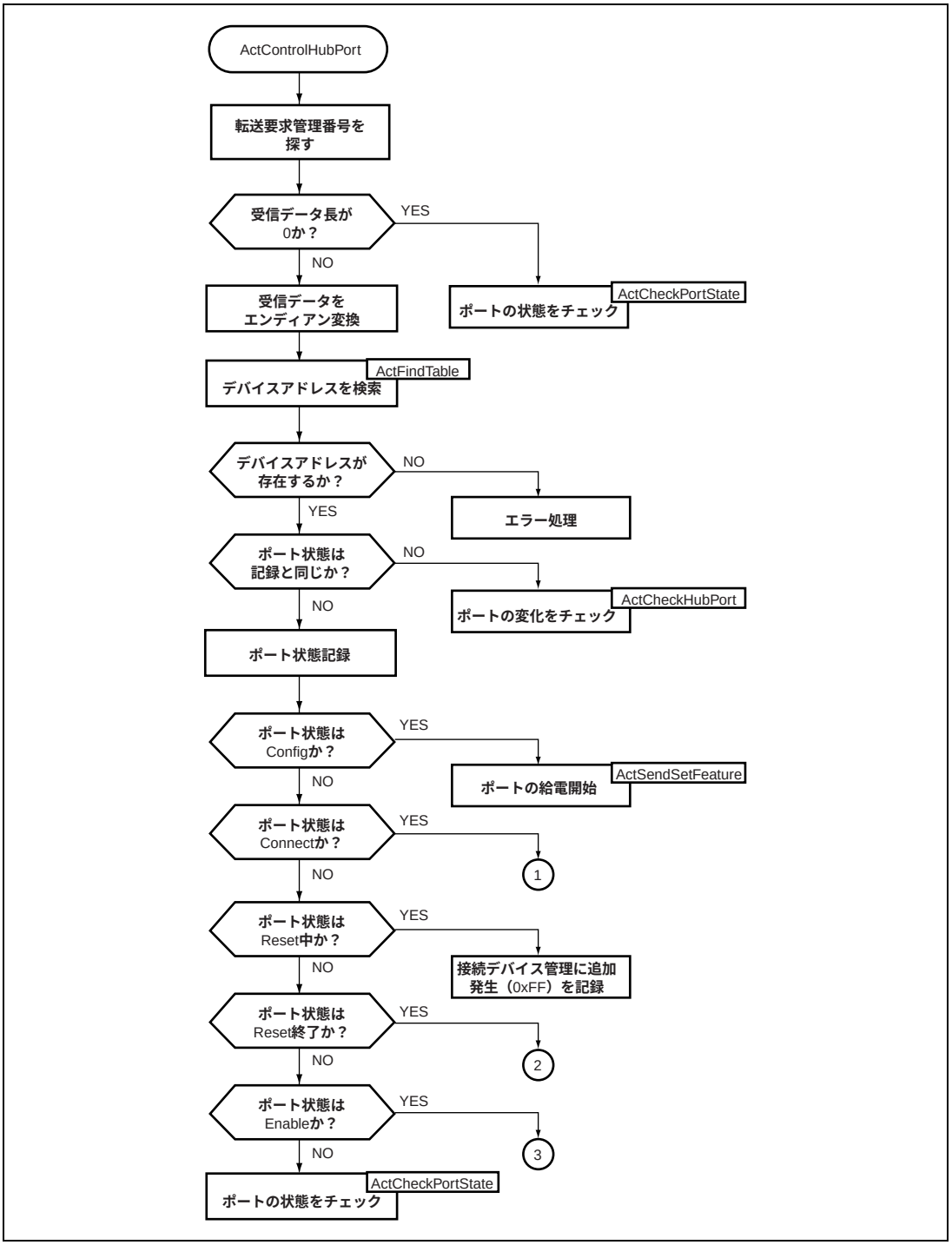


図 5.14-3　**転送結果処理**（3）

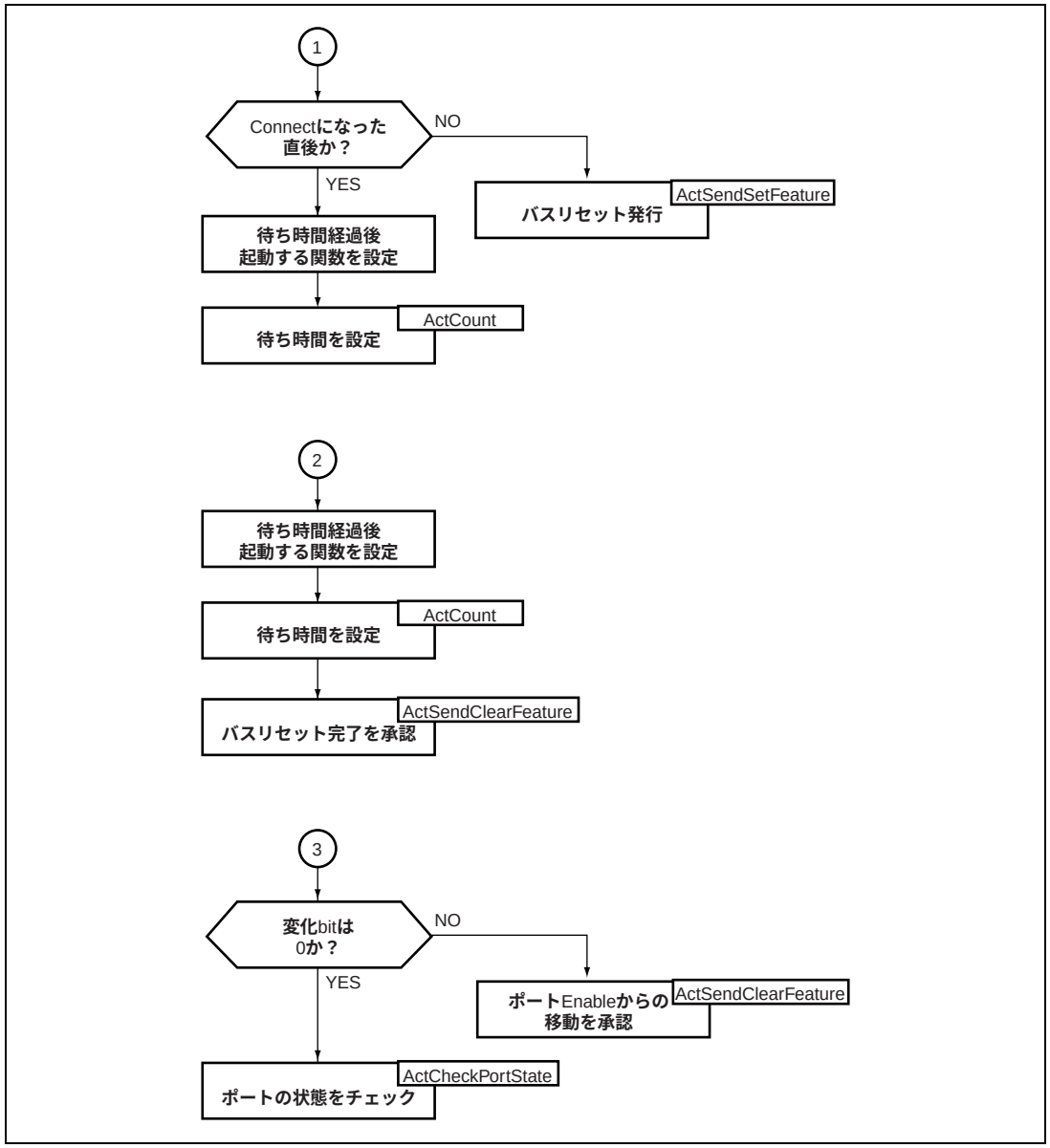


図 5.14-4　**転送結果処理**（4）

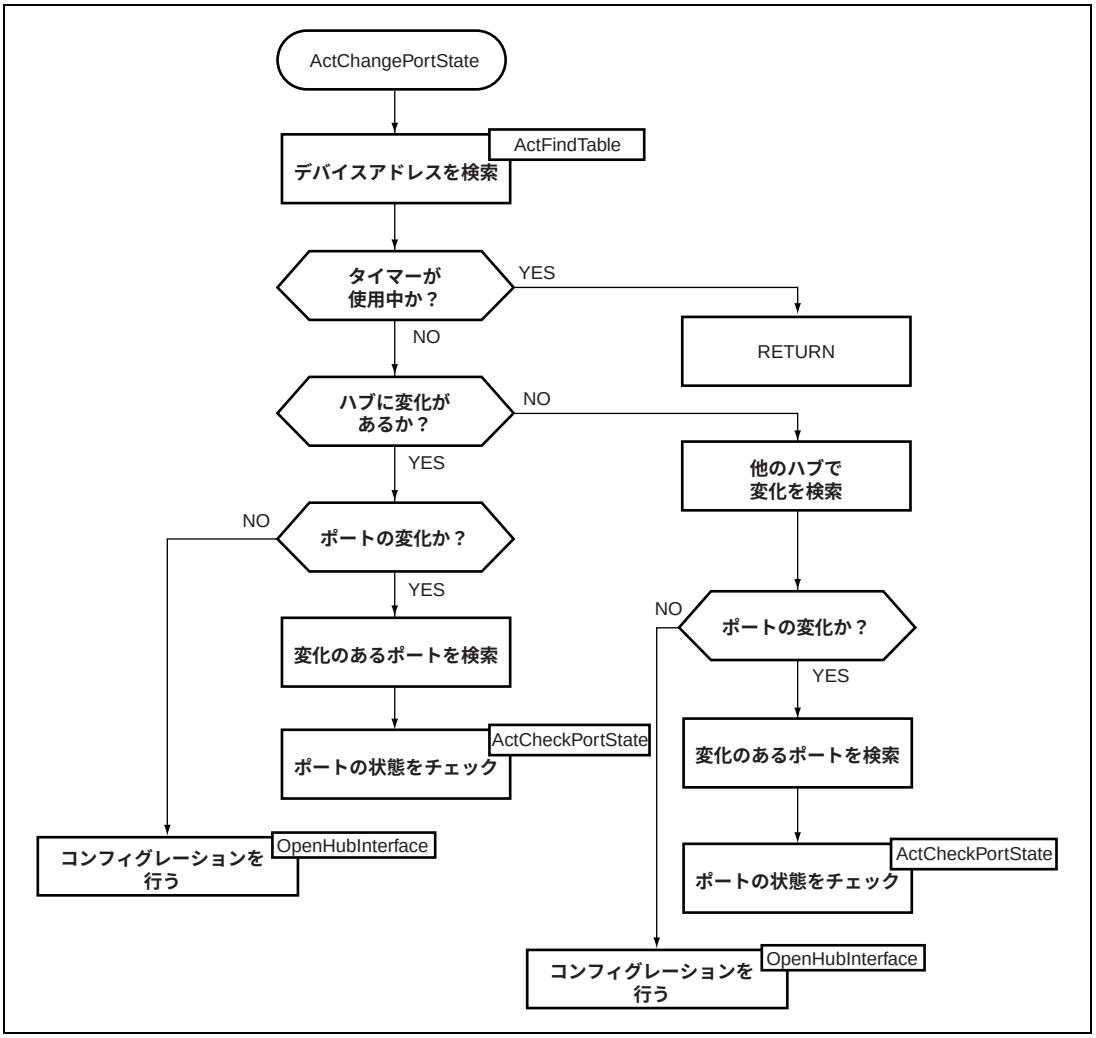


図 5.14-5　**転送結果処理（5）**

5.8 SH7727 ホストの HID 制御

HID ドライバは、接続された HID デバイスの起動処理、HID データ転送、の 2 つを行います。

図 5.15 に HID ドライバ群全体図を示します。

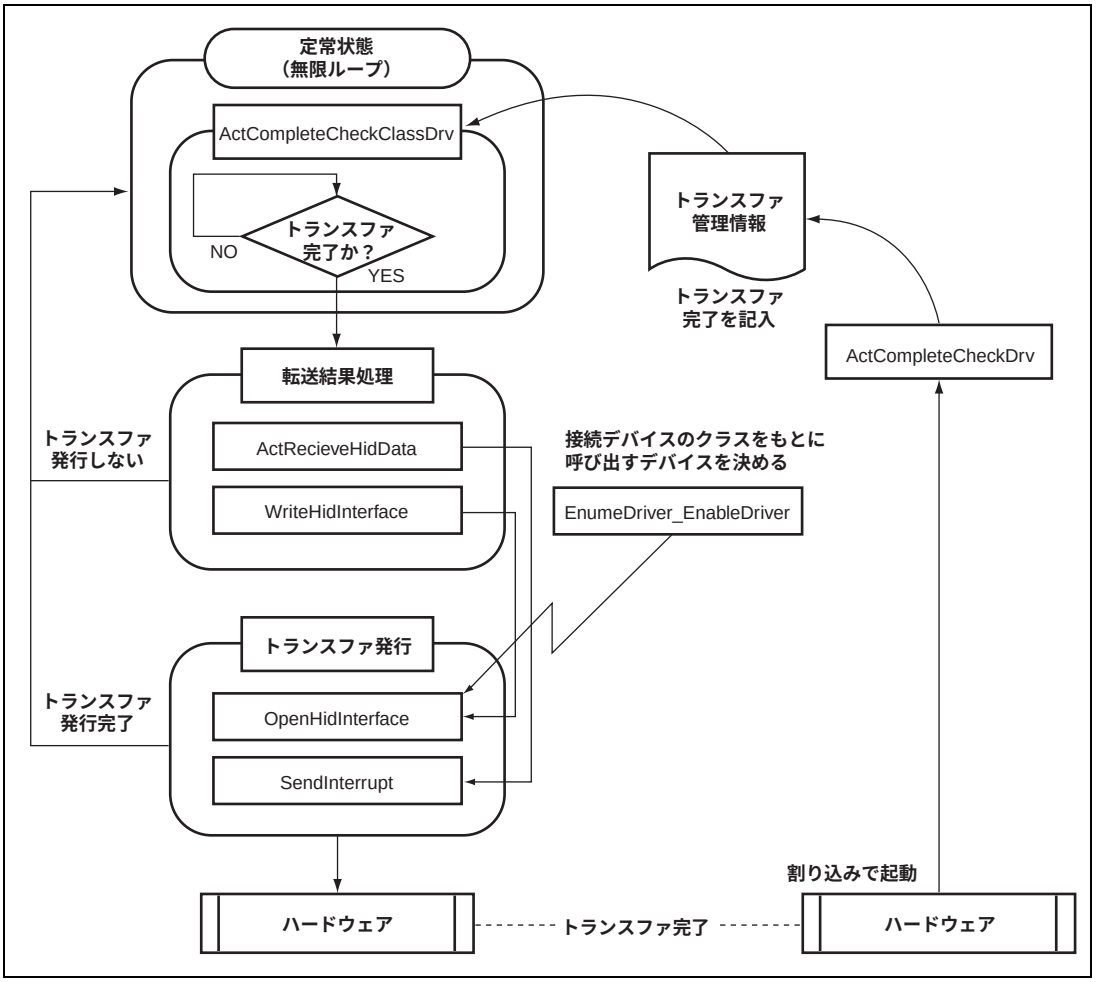


図 5.15　HID **ドライバ群全体図**

USBD 層の関数でデバイスアドレスの設定が完了すると OpenHidInterface 関数が呼び出され、HID ドライバによる起動処理が開始されます。

図 5.16-1 にディスクリプタ取得から起動処理完了までの SH7727 ホストと HID デバイス間で行われる転送を示します。

図 5.16-2 にインタラプト転送による HID データの転送で SH7727 ホストと HID デバイス間で行われる転送を示します。

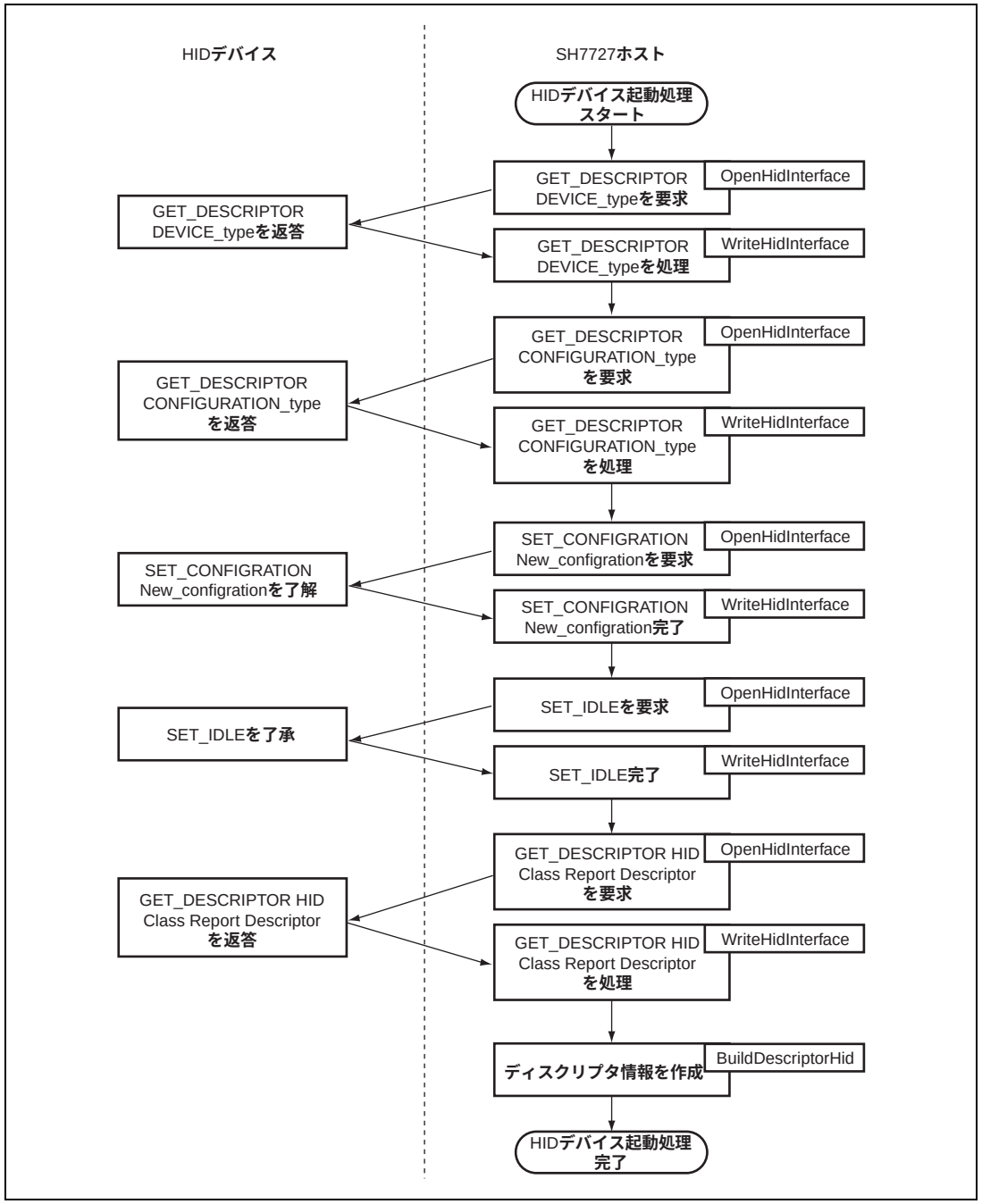


図 5.16-1　SH7727 **ホスト**-HID **デバイス転送図** 1

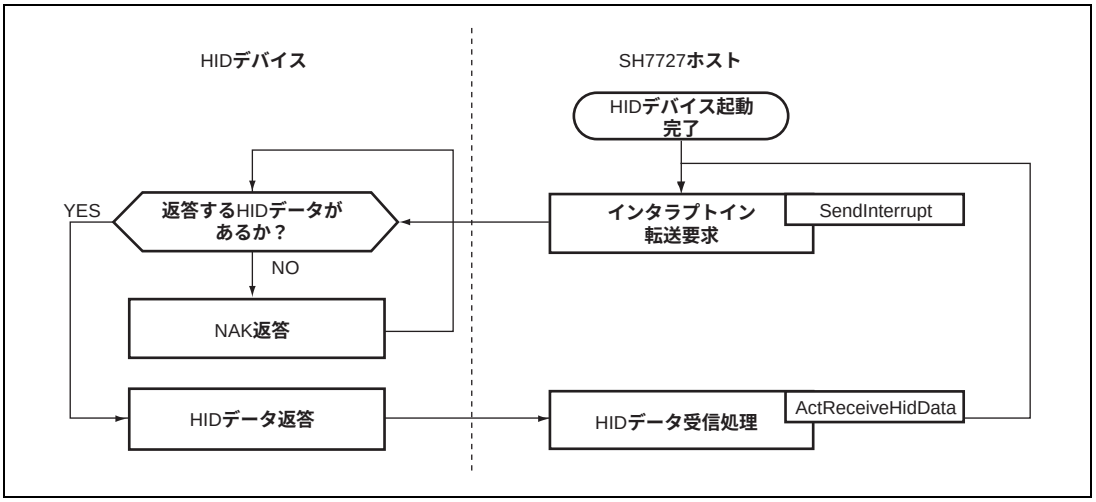


図 5.16-2　SH7727 **ホスト**-HID **デバイス転送図** 2

次にトランスファを発行する関数、2 つのフロー図を示します。

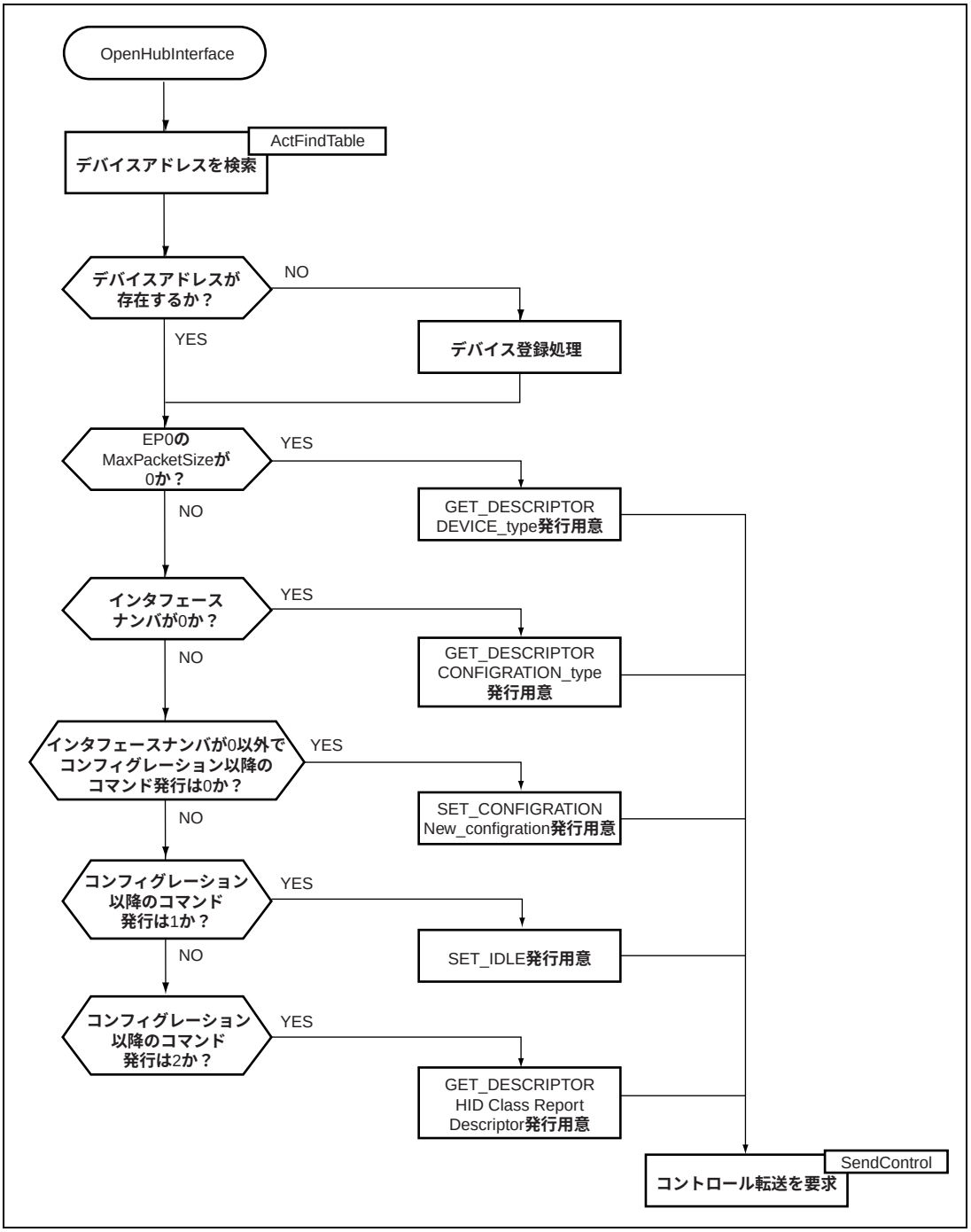


図 5.17-1　**トランスファ発行関数**（1）

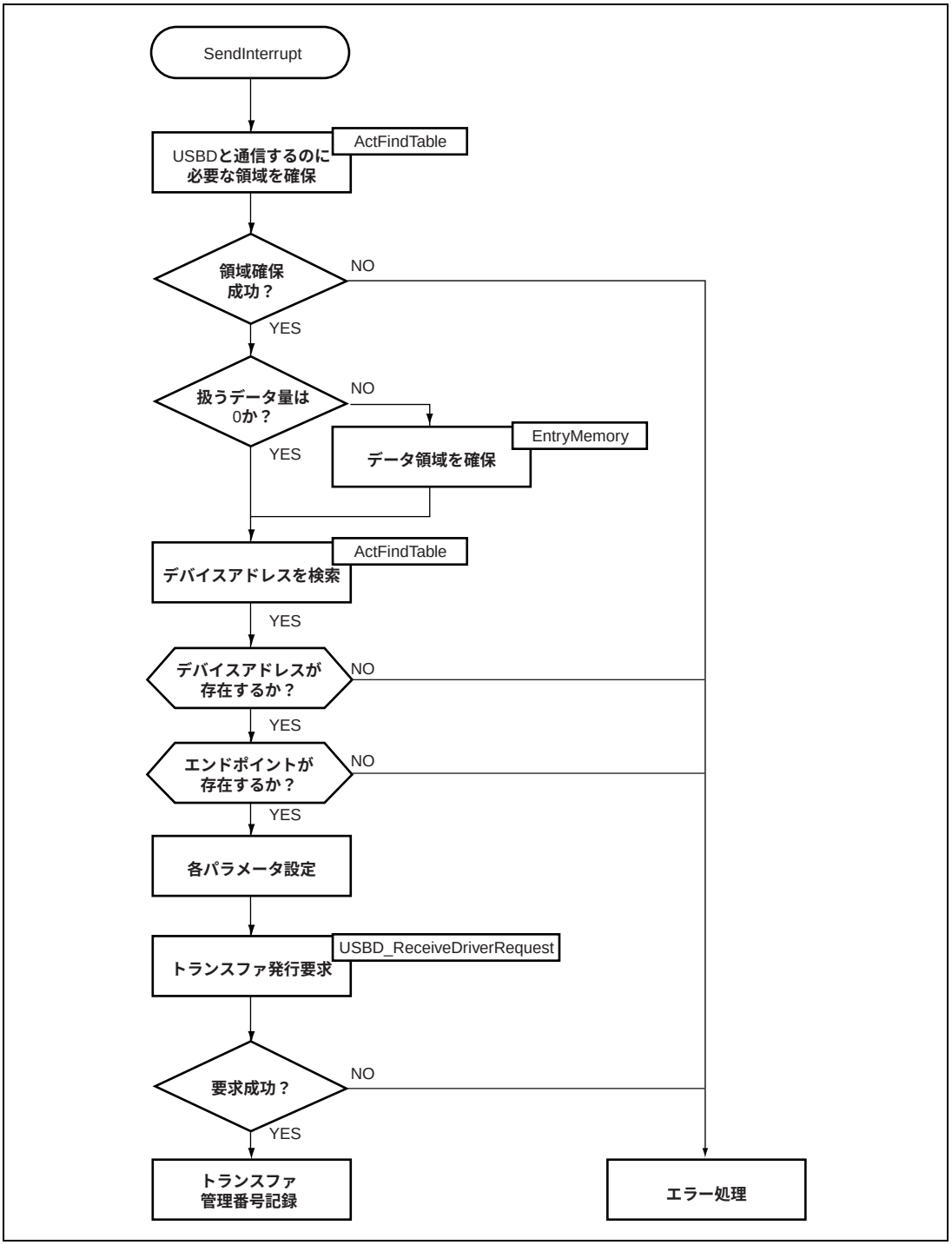


図 5.17-2　**トランスファ発行関数（2）**

次にトランスファの完了後に処理を開始する転送結果処理関数、2 つのフロー図を示します。

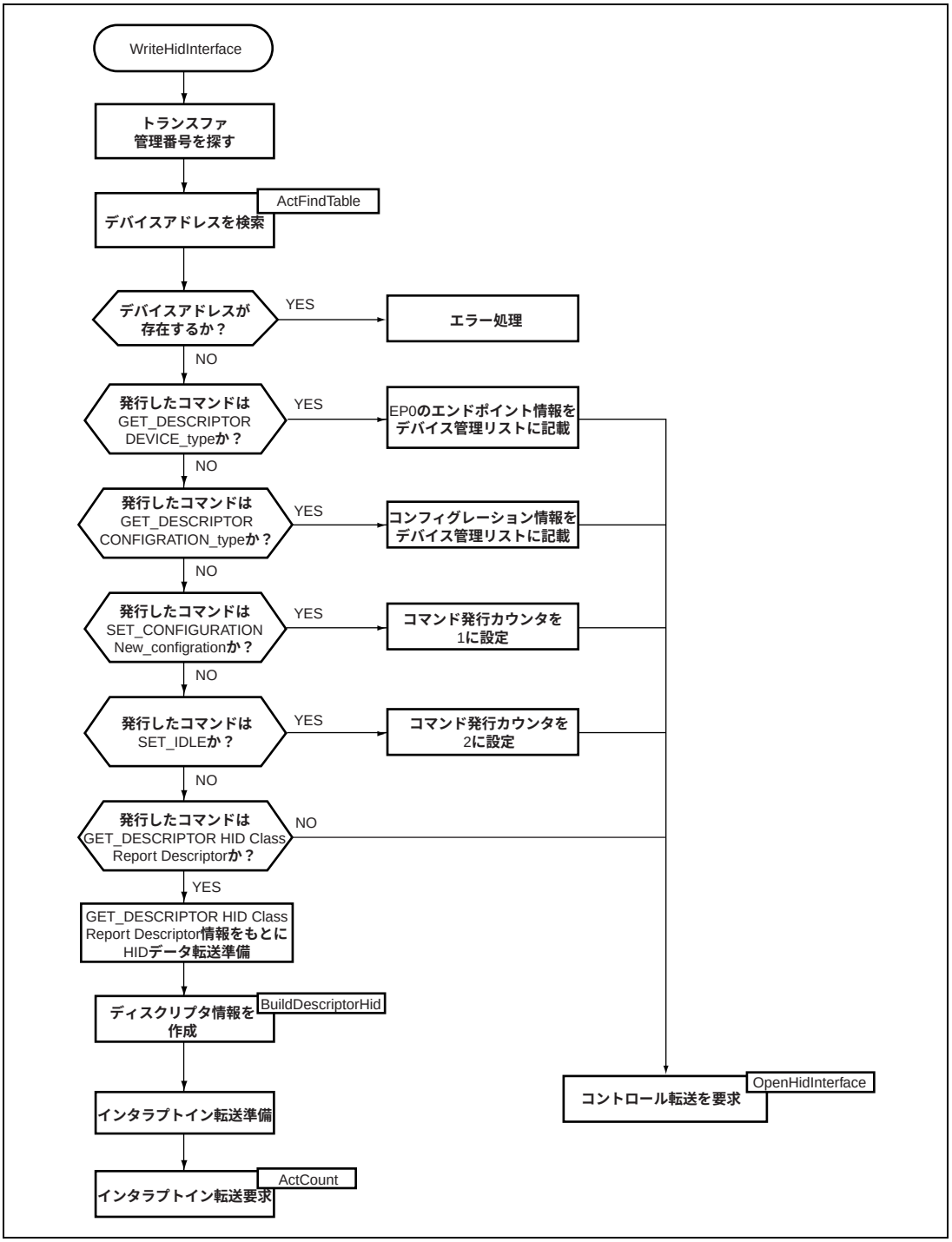


図 5.18-1　**転送結果処理（1）**

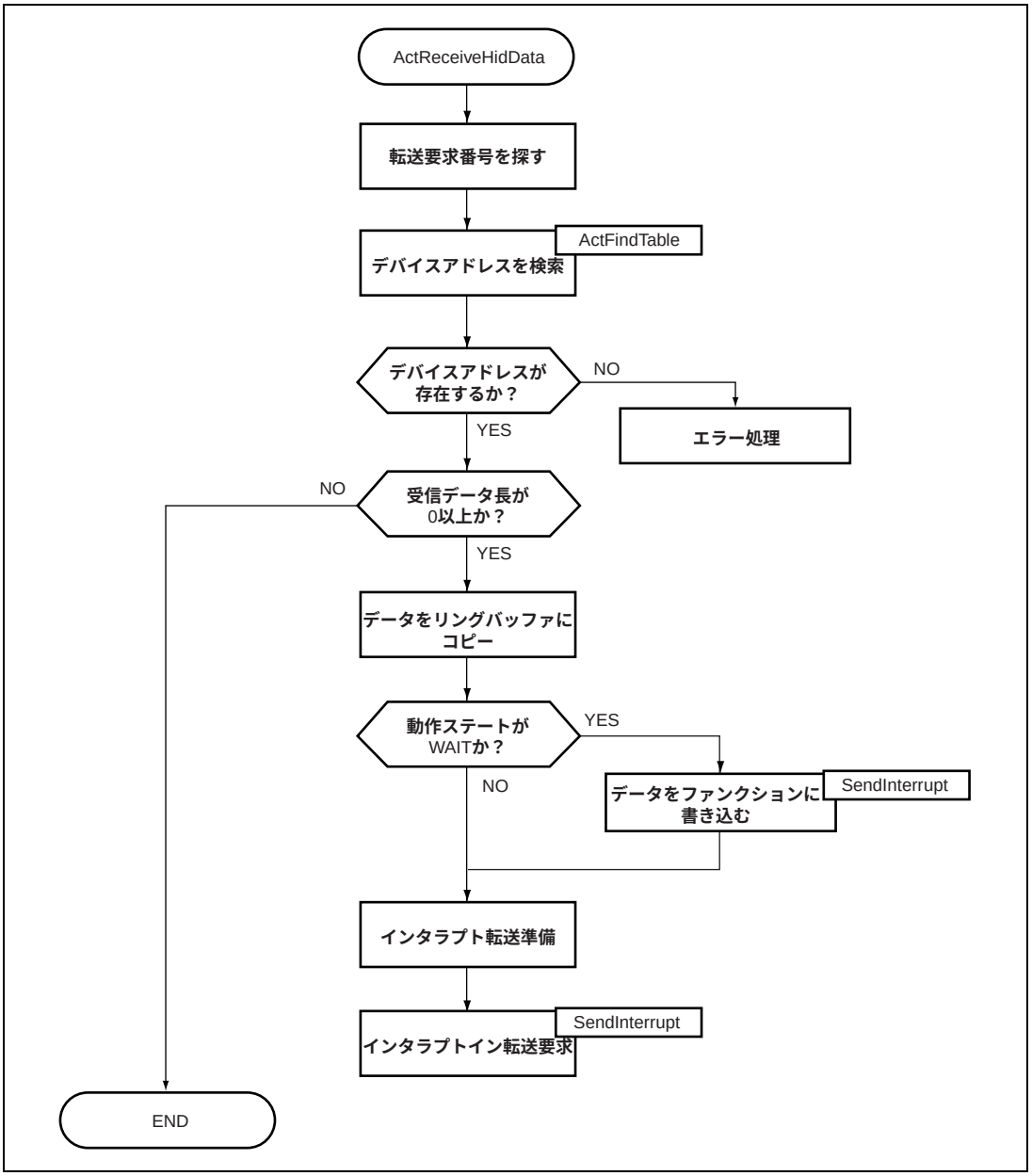


図 5.18-2　**転送結果処理（2）**

5.9 SH7727 ホストのストレージ制御

ストレージドライバは接続されたストレージデバイスの起動処理を行います。実際のストレージデータ転送はリンクプログラムがホスト PC とストレージデバイス間を取り持っています。

図 5.19 にストレージドライバ群全体図を示します。

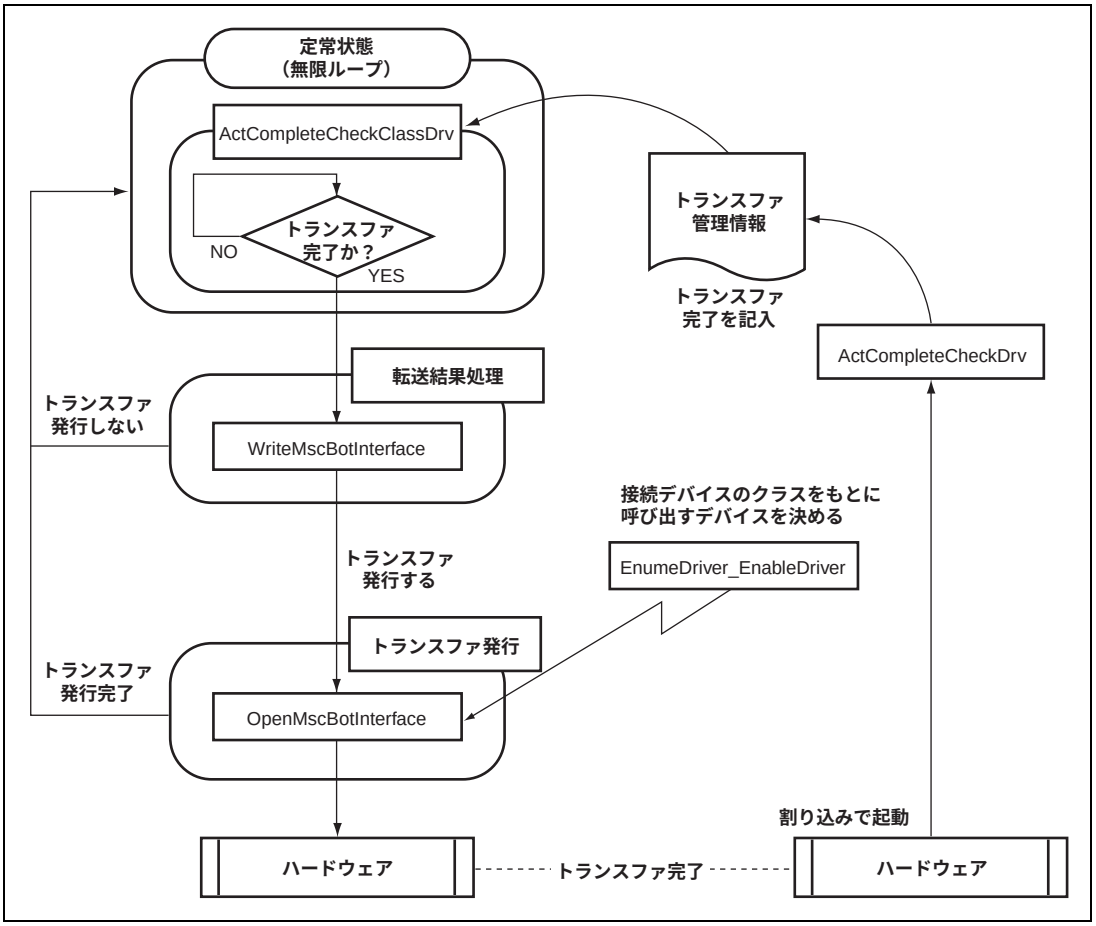


図 5.19　**ストレージドライバ群全体図**

USBD 層の関数でデバイスアドレスの設定が完了すると OpenMscBotInterface 関数が呼び出され、ストレージドライバによる起動処理が開始されます。

図 5.20 にディスクリプタ取得から起動処理完了までの SH7727 ホストとストレージデバイス間で行われる転送を示します。

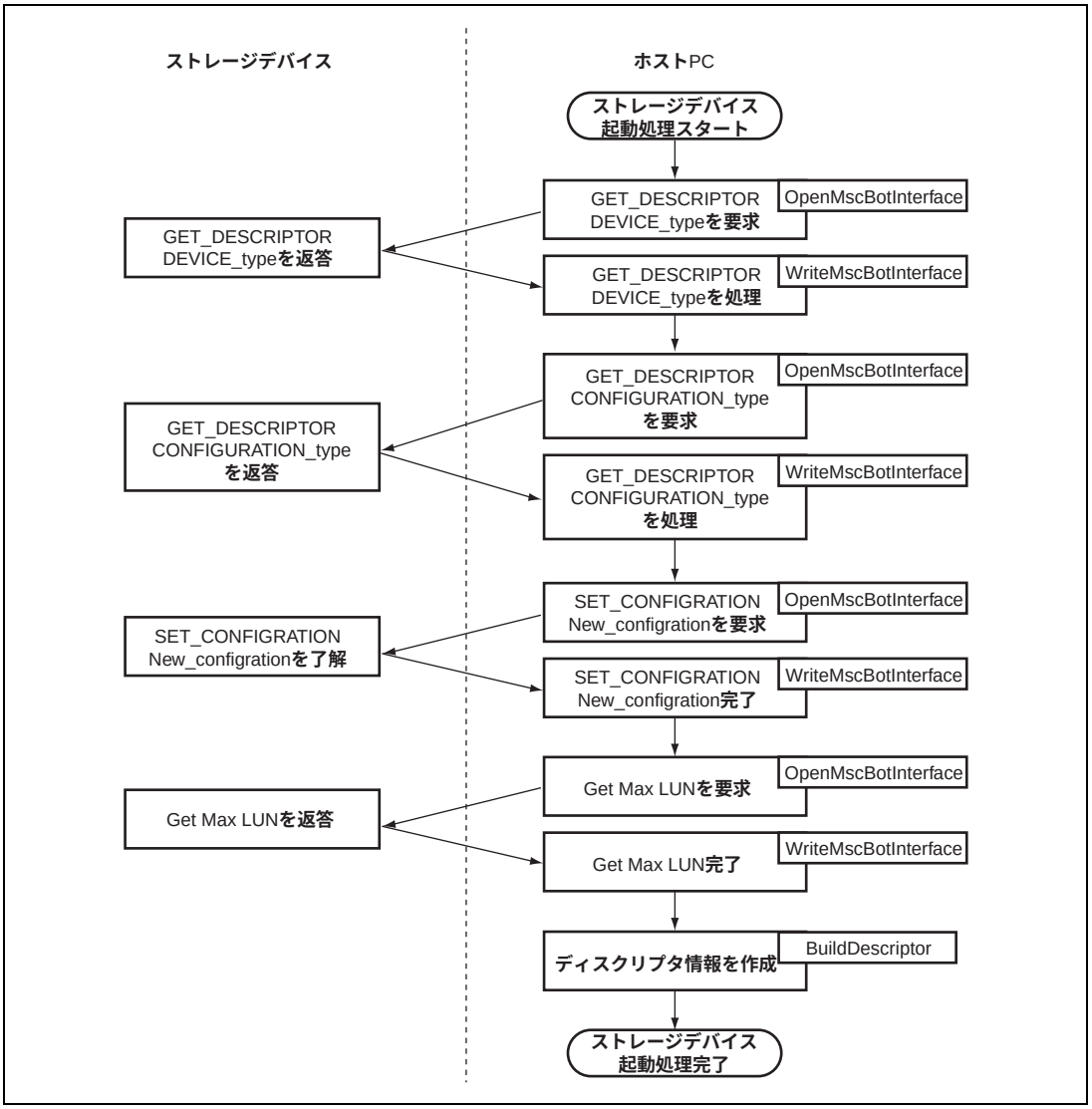


図 5.20　SH7727 **ホスト－ストレージデバイス転送図**

次にトランスファ発行関数のフロー図を示します。

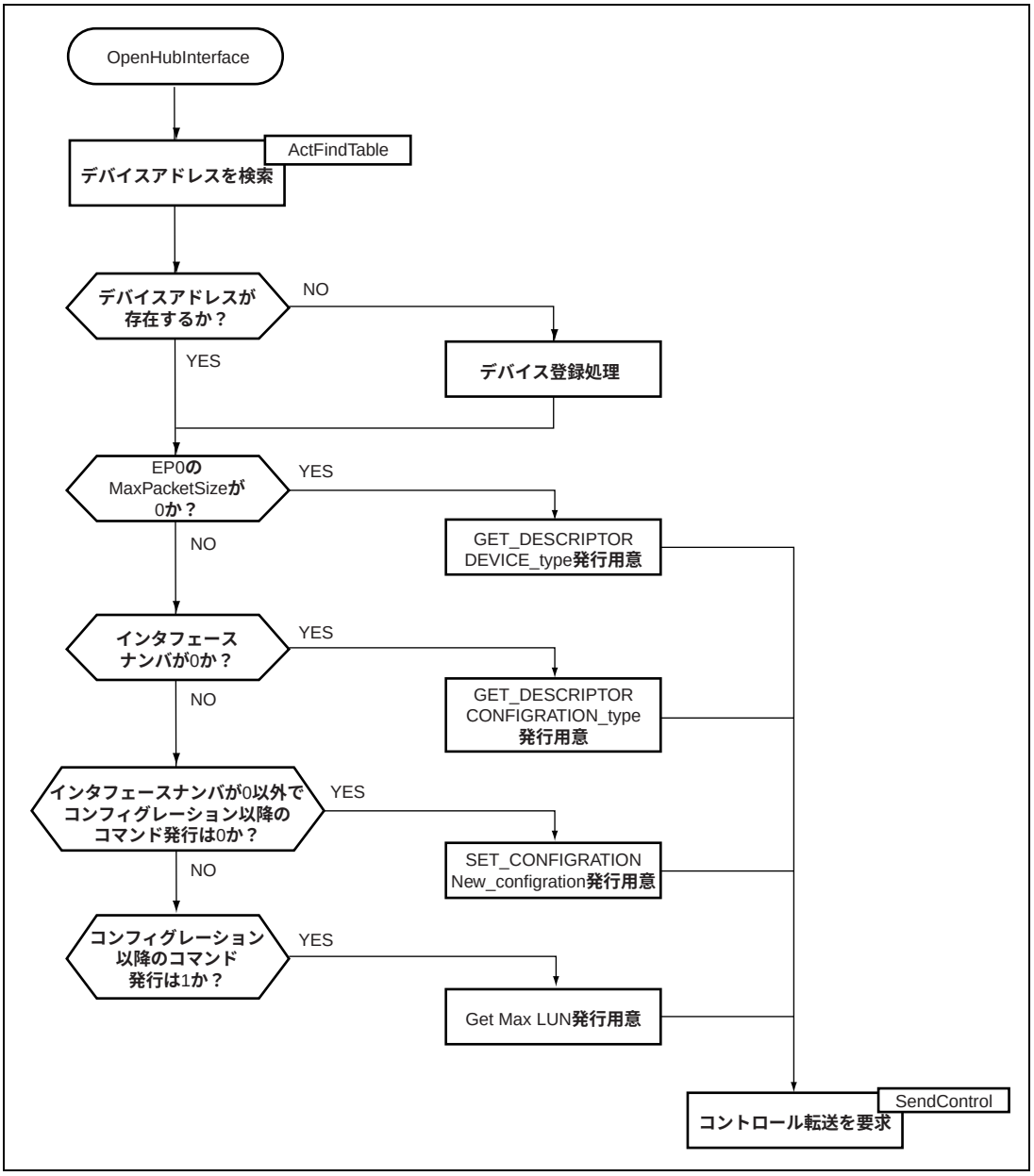


図 5.21 **トランスファ発行関数**

次にトランスファの完了後に処理を開始する転送結果処理関数のフロー図を示します。

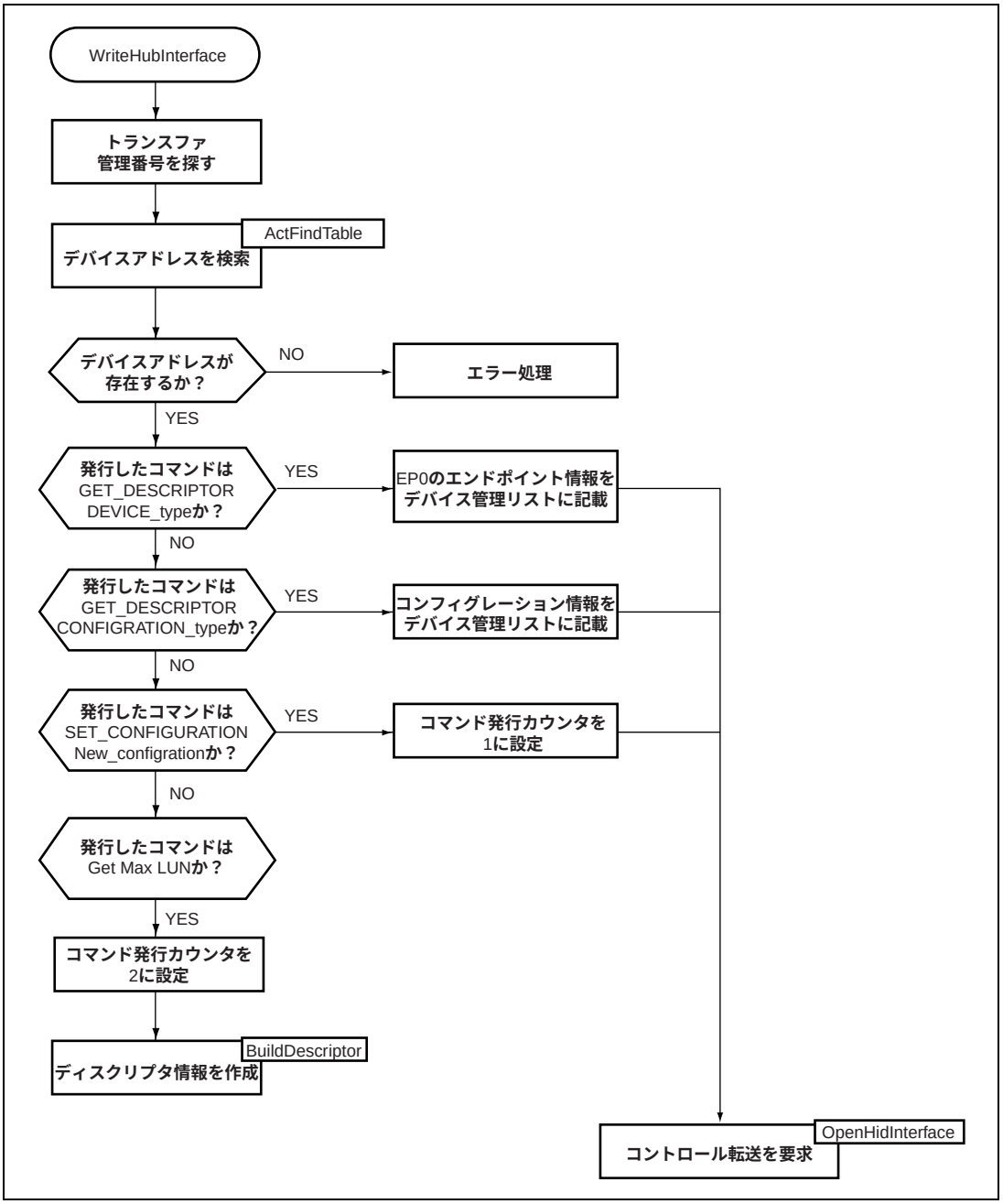


図 5.22　**転送結果処理**

5.10 SH7727 ホスト-SH7727 ファンクションのリンク動作

本サンプルプログラムはリンクプログラム、SH7727 ホストのプログラム、SH7727 ファンクションのプログラム、と大きく 3 つに分かれます。

- SH7727ホストのプログラムは接続されたUSBハブ、USBファンクションデバイスの制御を行い、データの送受信を行います。
- SH7727ファンクションのプログラムはホストPCとデータの送受信を行います。
- リンクプログラムはホストPCとストレージデバイス間でストレージデータを扱う場合、SH7727ホストとSH7727ファンクションのデータ転送を受け持ちます。

SH7727 ファンクションのプログラムが通信する相手はホスト PC に限定されます。SH7727 ファンクションが転送するデータ内容はそれぞれ異なりますが、何をどのように転送するかは USB の転送方式毎にまとまっている為ファームウエアの詳細は USB の転送方式ごとに示しました。

SH7727 ホストのプログラムは接続された USB ハブまたは、USB ファンクションデバイスにより制御内容が大きく違うので、ファームウエアの詳細は接続デバイス毎にどの様な制御を行うかを示しました。

リンクプログラムはストレージデバイスを扱う際に使用し、SH7727 ホスト、SH7727 ファンクションそれぞれの間を取り持ちデータ転送、機器の制御を行います。**図** 5.23 にリンクプログラム全体図を示します。

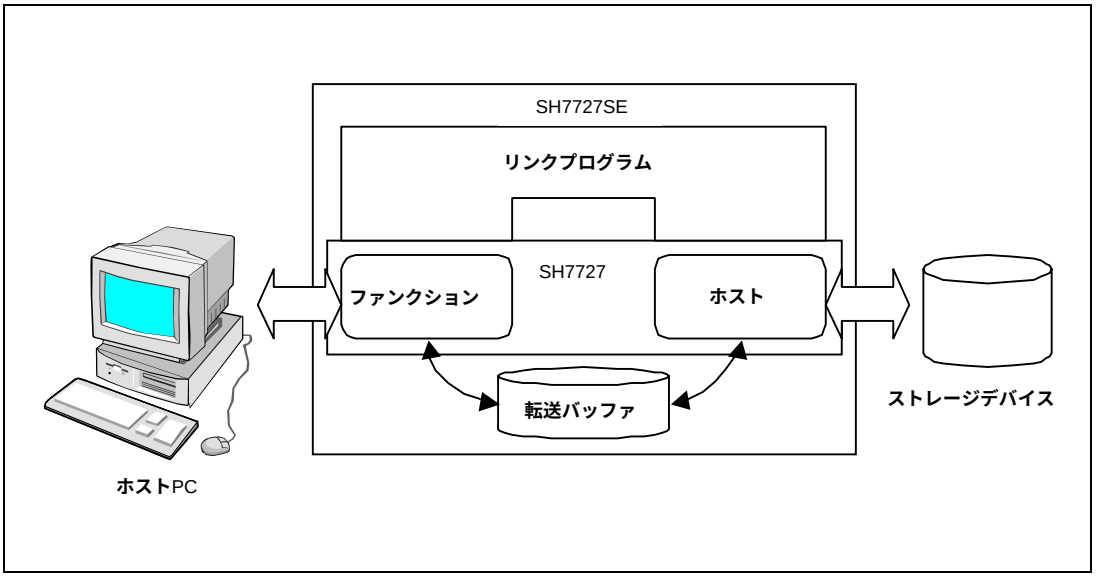


図 5.23 **リンクプログラム全体図**

次に SH7727 ファンクションからの割り込みを起動要因とする関数のフロー図を示します。

リンクプログラムでは SH7727 ファンクションからの割り込みが起動要因となり動作が開始しますが、SH7727 ファンクションに対する動作と SH7727 ホストに対する動作の 2 つを行います。つまり 1 つの要因で 2 つの異なるモジュールを制御する必要があります。通常のデータ転送時は SH7727 ファンクションから SH7727 ホストへまたは、SH7727 ホストから SH7727 ファンクションへ、と言う様にデータパケットをリレーしますが、ストレージデバイスが STALL を返答した場合、データパケットの流れと、SH7727 ファンクション、SH7727 ホスト、の動作がそれぞれ異なります。

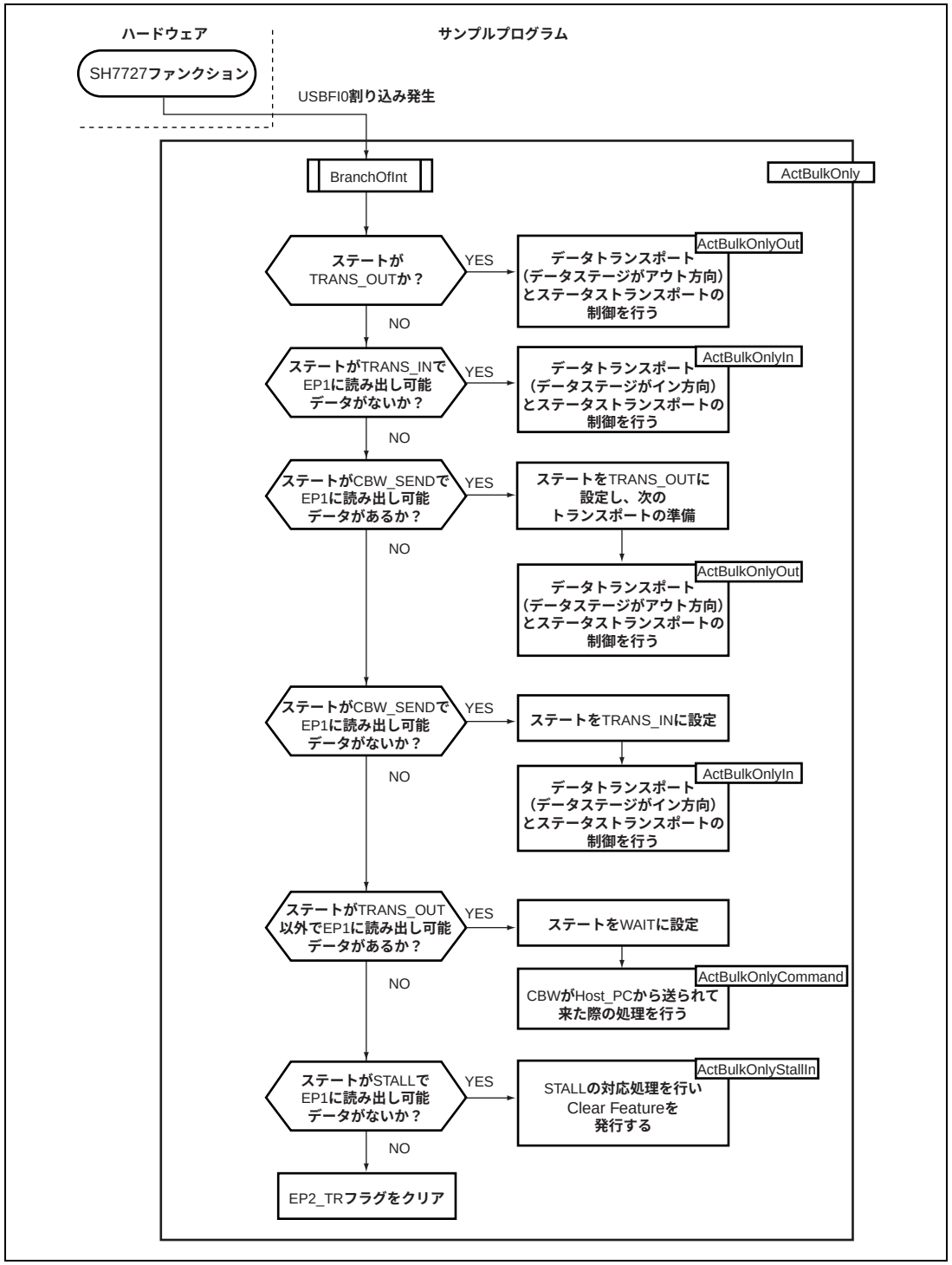


図 5.24-1　SH7727 **ホスト**ーSH7727 **ファンクションのリンク動作関数**

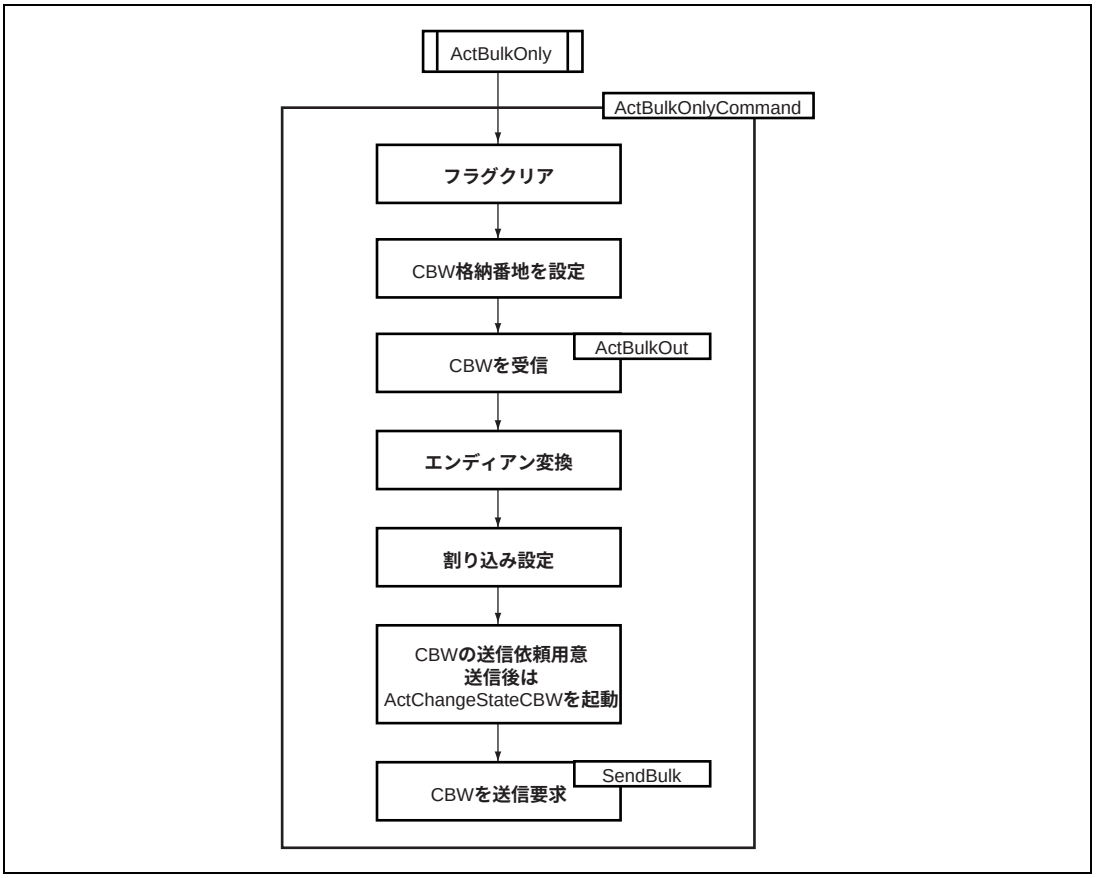


図 5.24-2　CBW **処理関数**

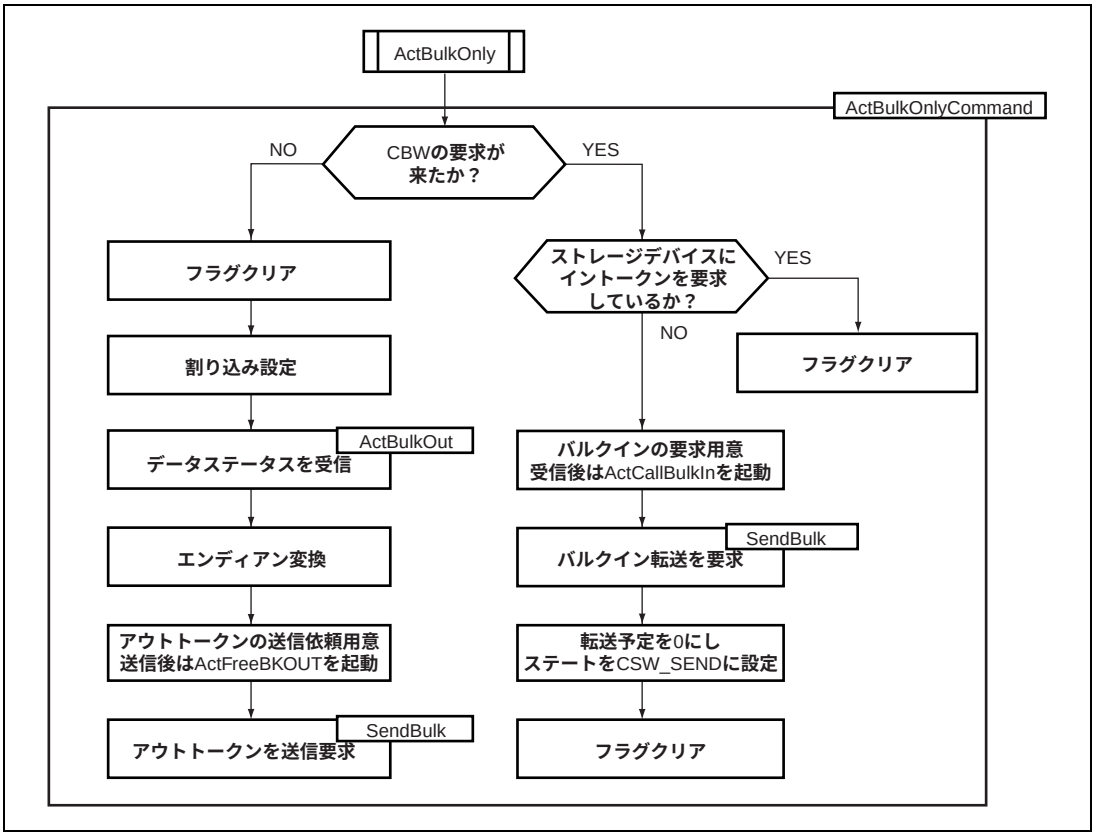


図 5.24-3　**データトランスポート（データステージがアウト方向）とステータストランスポートの制御関数**

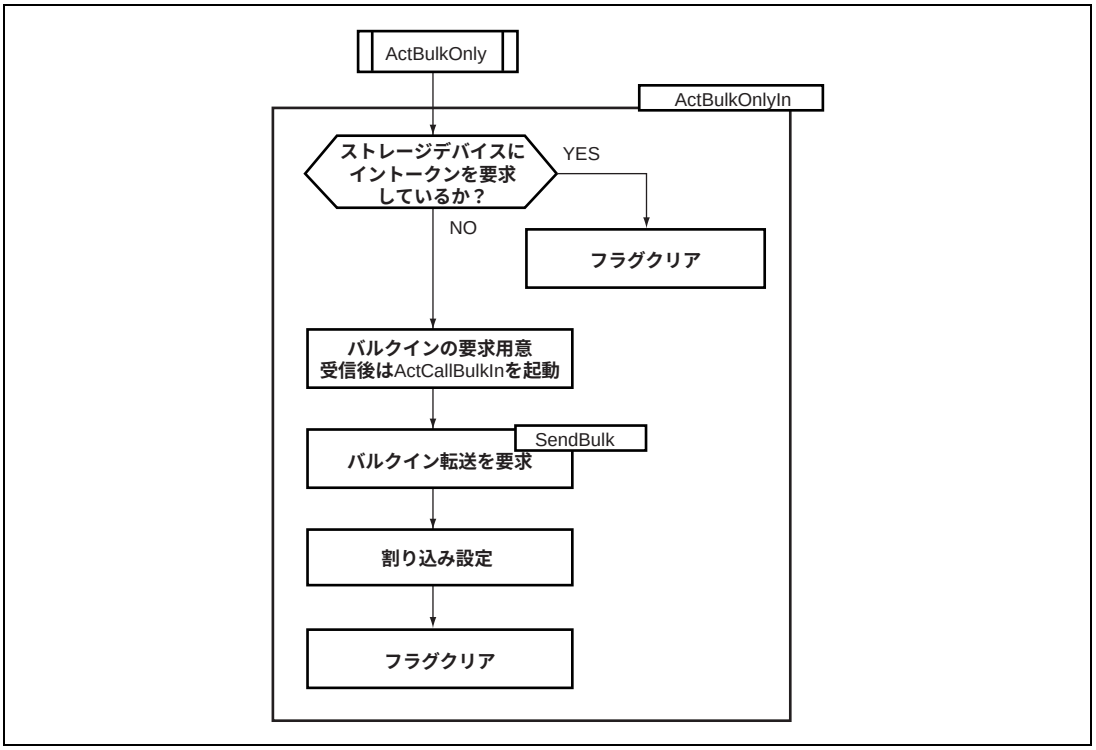


図 5.24-4　**データトランスポート（データステージがイン方向）とステータストランスポートの制御関数**

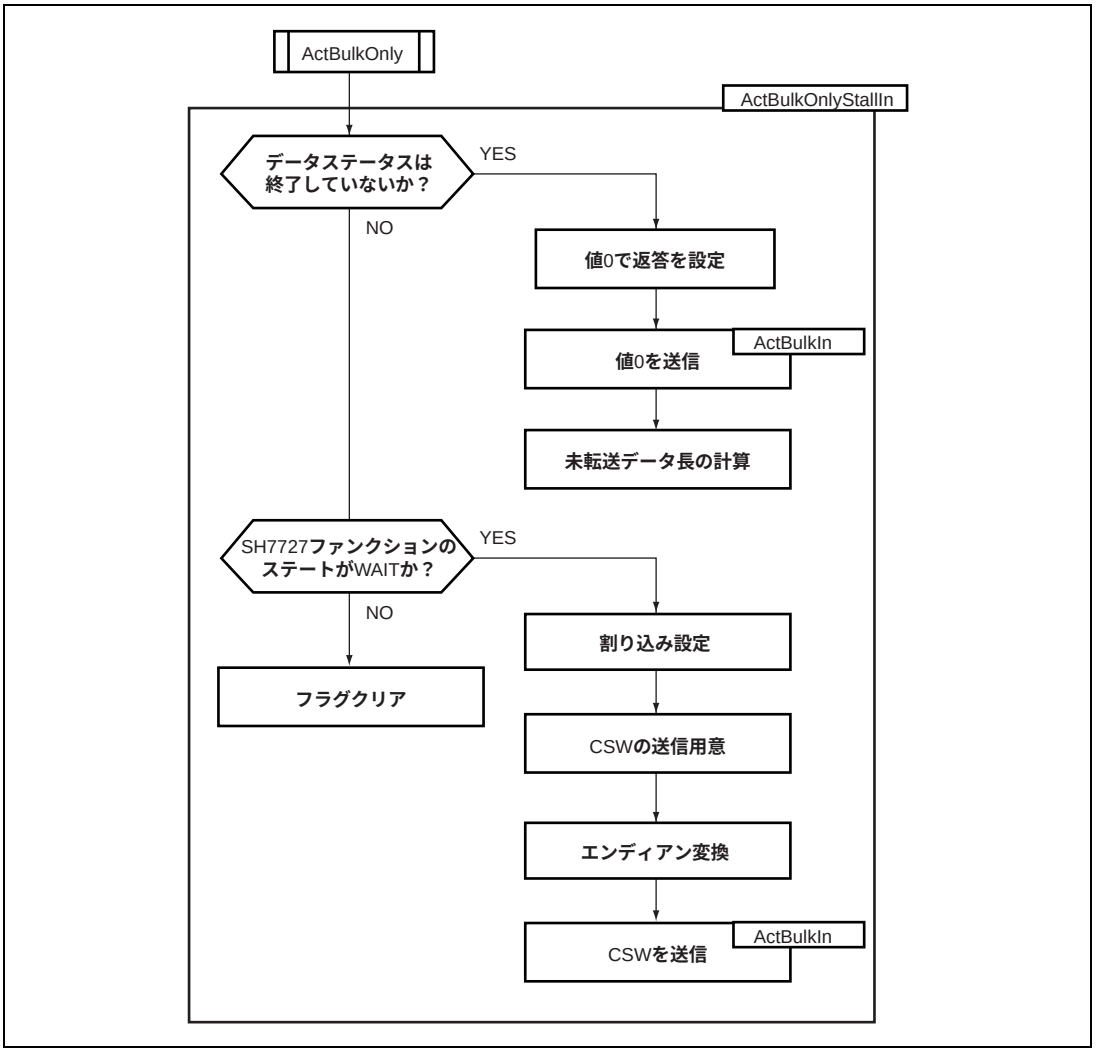


図 5.24-5　STALL **対応関数**

次に SH7727 ホストの処理終了を起動要因とする関数のフロー図を示します。

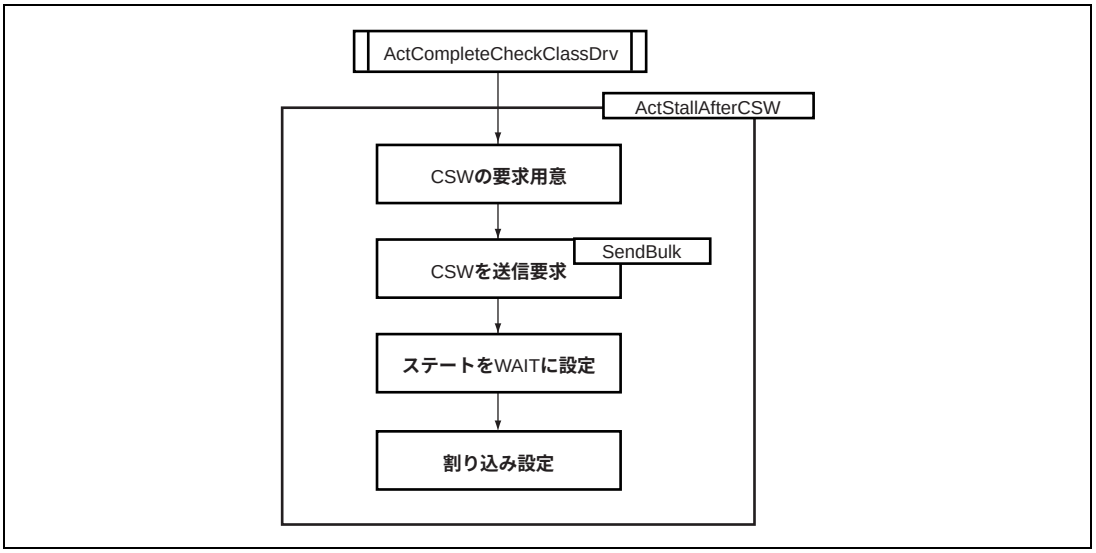


図 5.25-1　Clear Feature **完了後対応関数**

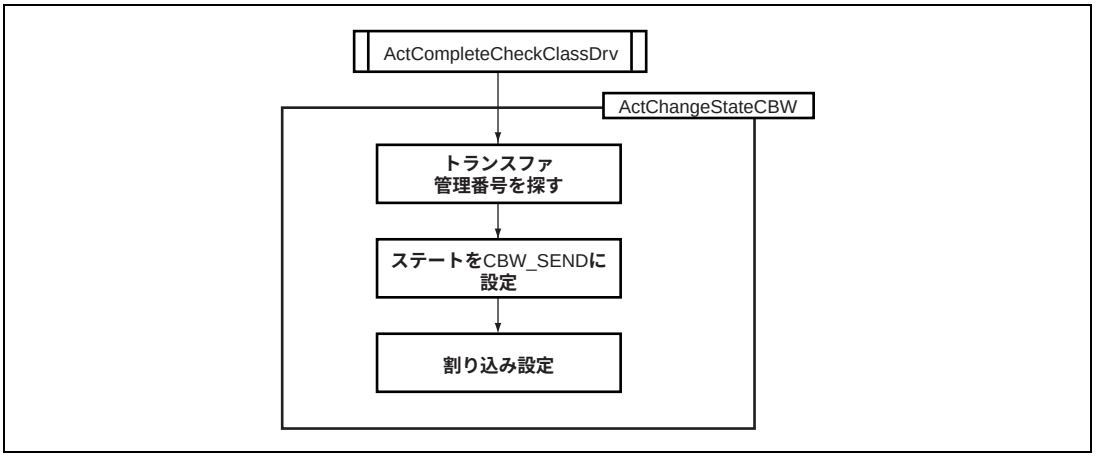


図 5.25-2　CBW **送信完了後対応関数**

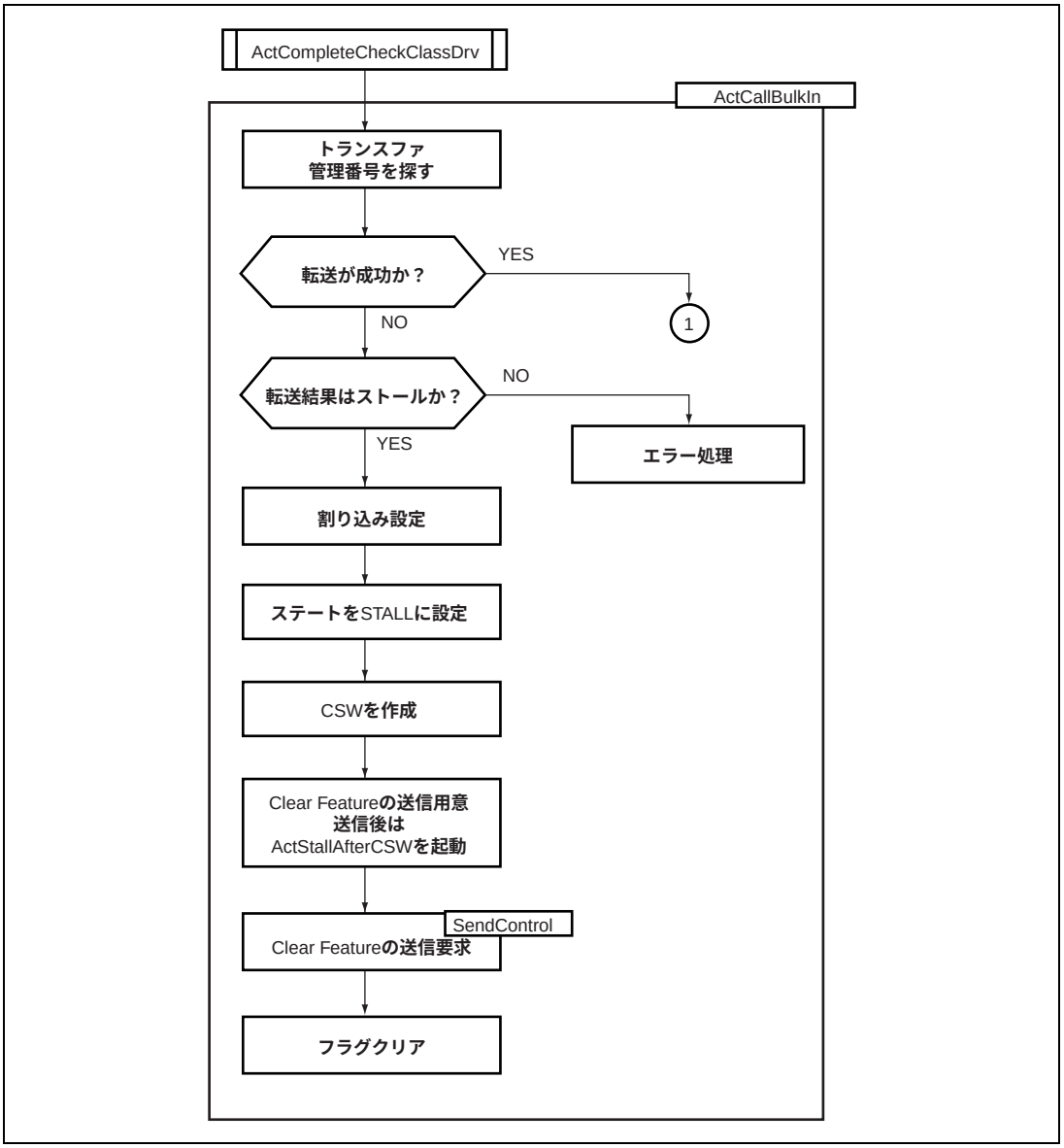


図 5.25-3　**バルクイン転送完了後対応関数（1）**

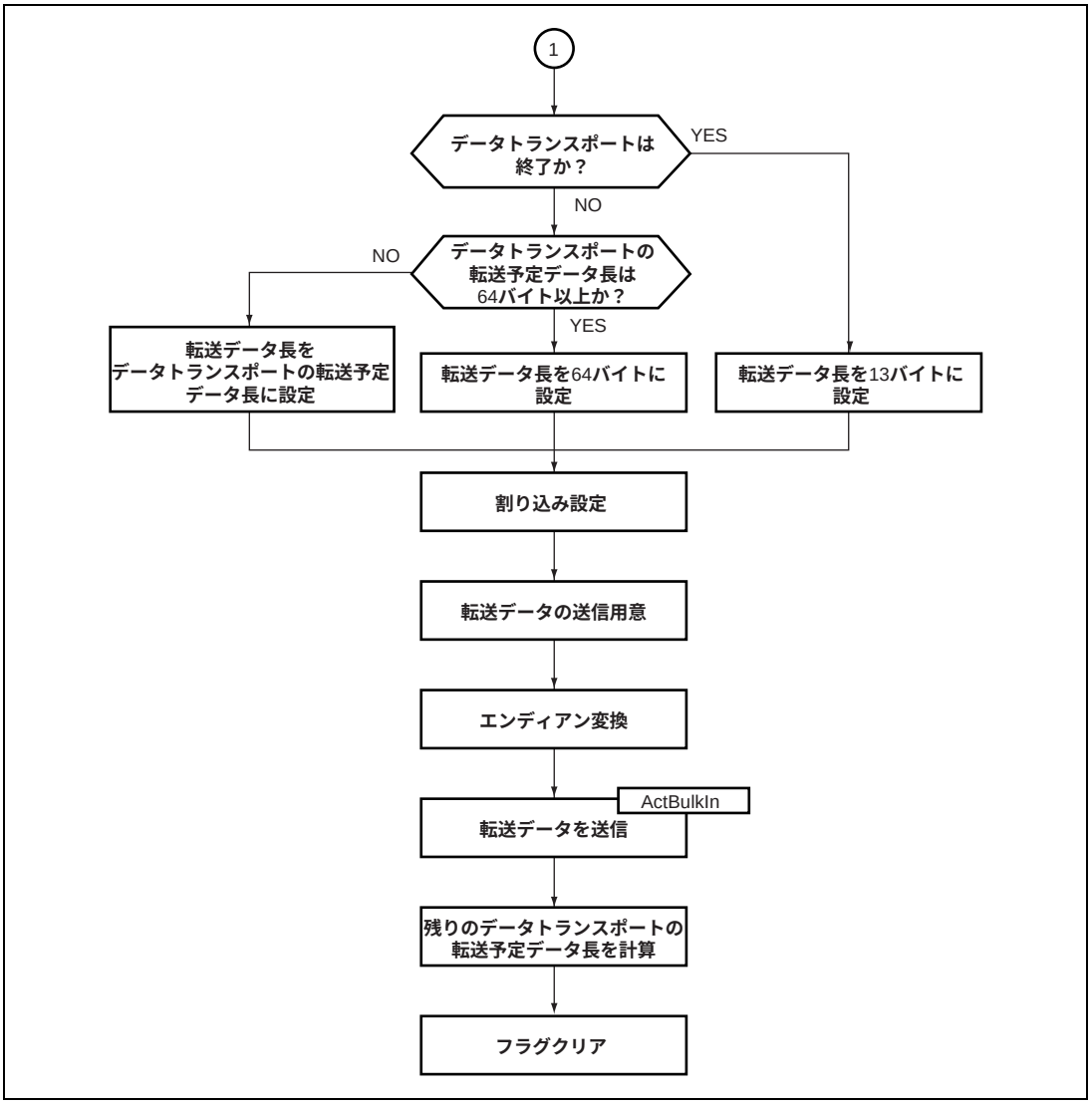


図 5.25-4　**バルクイン転送完了後対応関数（2）**

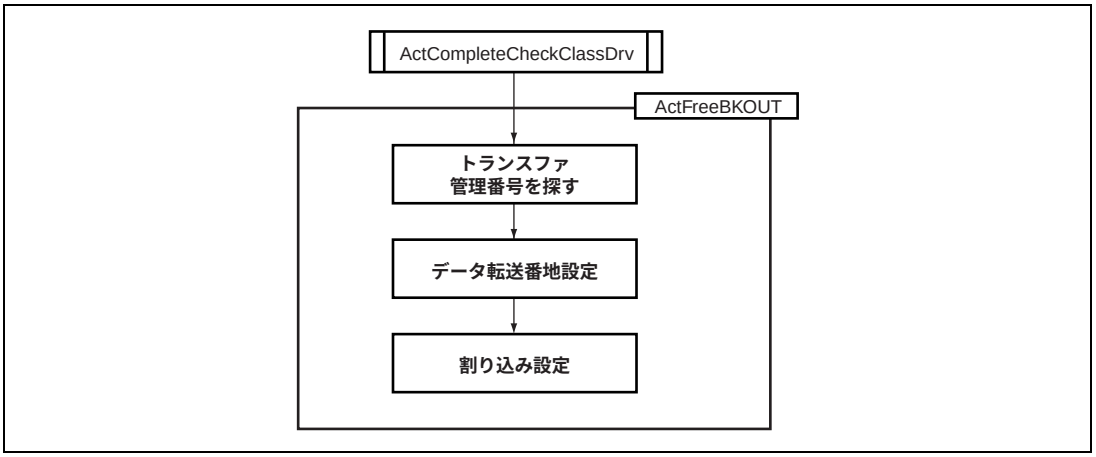


図 5.25-5　**データトランスポート（データステージがアウト方向）の送信完了後対応関数**

SH7727 USBホストモジュールアプリケーションノート（応用編）

発行年月　2003年4月15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 http://www.renesas.com

ルネサス販売本社	〒100-0005	千代田区丸の内1-8-2（第二鉄鋼ビル）	(03) 3215-8600
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) 256-9021

■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 0.0

SH7727 グループ　USB ホストモジュール
アプリケーションノート（応用編）

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

RJJ05B0016-0100Z