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

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 ホストモジュールを用いたファームウェアについて説明したものであり、お客様が USB ホストモジュールファームウェア作成の際に、御参考として役立てて頂けるようにまとめました。

本アプリケーションノートの内容およびサンプルプログラムは、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 Revision 1.0a
- 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

【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 内蔵の USB ホストモジュールの使用方法、およびサンプルプログラムについて説明したものです。

SH7727 内蔵 USB ホストモジュールの特長を以下に示します。

- Open Host Controller Interface（OHCI）1.0 レジスタセット準拠
- USB1.1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
- CPUに接続されたユーザメモリ領域を転送用データ、デスクリプタ用として利用可能

サンプルプログラムを動作させるためのシステム構成を**図** 1.1 に示します。

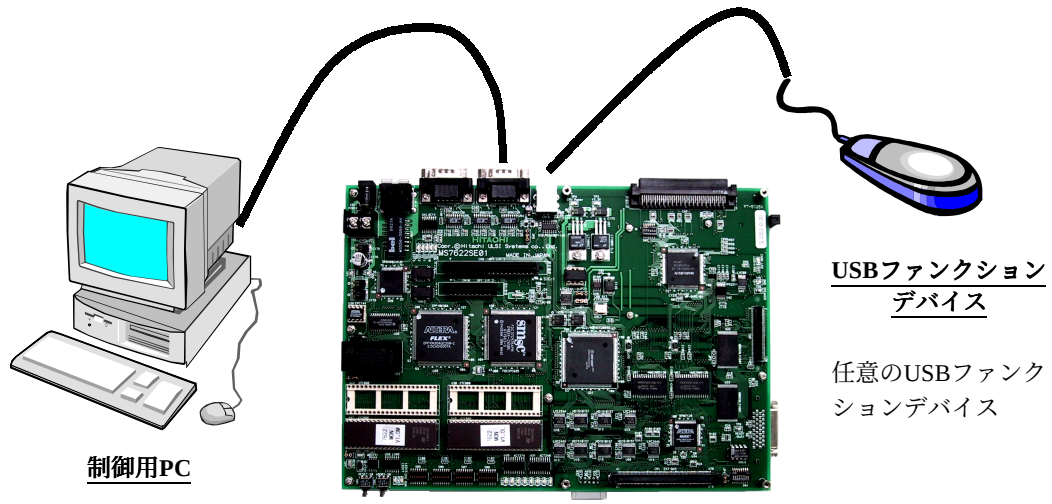

図 1.1　**システム構成**

1. 概要

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® 98/2000」を搭載した PC、USB ファンクションデバイスによって構成されています。

本システムは、制御用 PC と SH7727SE をシリアルケーブルで接続します。制御用 PC 上の USB パケット生成ツール「Request Generator」から、SH7727SE に接続した USB ファンクションデバイスに対して、様々な USB パケットを生成することができます。

本システムの特長を以下に示します。

1. サンプルプログラムにより、SH7727のUSBホストモジュールを短期間で評価可能
2. サンプルプログラムは、USBのコントロール転送、バルク転送、インタラプト転送をサポート
3. プログラムを追加作成することにより、アイソクロナス転送についても対応可能*

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

2. Open Host Controller Interface（OHCI）規格の概要

SH7727 に搭載されている USB ホストモジュールは、Open Host Controller Interface（以下、OHCI）規格を採用しています。この章では、その OHCI 規格について説明します。USB ホストのシステムを開発する際に、ご参考としてお使いください。なお、OHCI 規格の詳細につきましては、

「Open Host Controller Interface Specification for USB Revision 1.0a」

をご覧ください。

2.1 規格の範囲

USB ホストのシステムは、**図** 2.1 のように、USB Device、Host Controller（HC）、Host Controller Driver（HCD）、USB Driver（USBD）、Client Software の階層に分けることが出来ます。

OHCI 規格では、HCD と HC の機能、および両者のインタフェース、つまり Software と Hardware のインタフェースを規定しています。

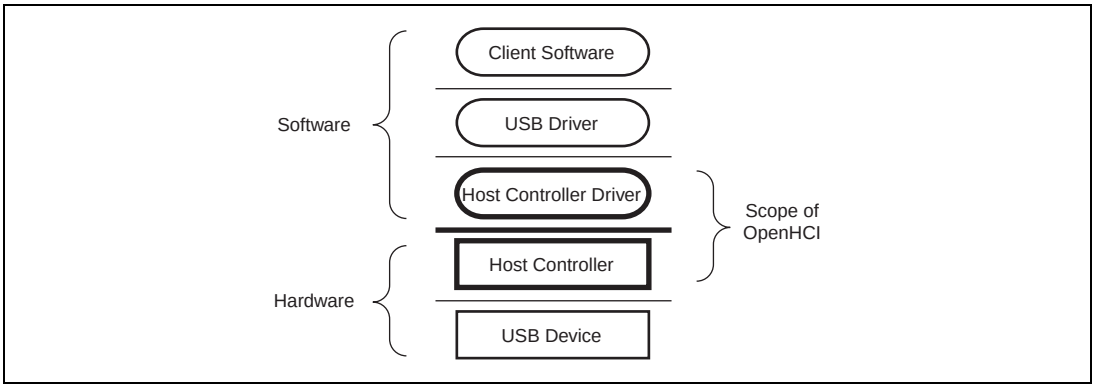

図 2.1　USB Focus Ares

2.2 データ転送のタイプ

USB には以下に示す 4 つの転送タイプがあります。

- Interrupt転送

  少量のデータを一定の間隔で転送します。転送の間隔は、対象となるデバイスに最適化することができます。

- Isochronous転送

  一定のデータレート、一定の周期で転送する方式です。

- Control転送

  デバイスのConfiguration情報、コマンド情報、Status情報を非周期的に転送する方式です。

- Bulk転送

  大量のデータを非周期的に転送する方式です。

OHCI 規格では、この 4 つの転送タイプをさらに Periodic タイプ、Non-Periodic タイプの 2 つに分類します。Interrupt 転送・Isochronous 転送は一定の周期で転送するようにスケジューリングされているので Periodic 転送、Control 転送・Bulk 転送は特定の周期で転送するようにはスケジューリングされていないので Non-Periodic 転送となります。

2.3 Host Controller Interface

HCD と HC の間では、EndpoingtDescriptor（ED）、TransferDescriptor（TD）というデータを受け渡しすることで、両者間の通信を行っています。ED は転送の対象となる Endpoint についての情報を纏めておくもので、その対象となる Endpoint が属しているデバイスのアドレス、Endpoint の番号、デバイスの Speed、MaxPacketSize 情報などを含んでいます。TD は、Endpoint に転送するデータパケットについての情報を纏めておくもので、PID、データトグル情報、メモリ上の送受信データへのアドレス、転送完了時のステータス情報などの情報を含んでいます。HCD は、転送要求を ED、TD に纏め、その ED、TD 群を転送タイプ毎に List という形に纏めて、List の先頭アドレスを HC に渡します。List の先頭アドレスを受け渡すには、HC の内部レジスタを介する経路と、メモリ上に用意する HostControllerCommunicationArea（HCCA）を介する経路の 2 つの方法があります。また、HC は転送処理を完了した TD を List 化してある DoneQueue の先頭アドレスを HCCA 経由で HCD に渡します。

以下に、List、ED、TD、HCCA、HCD－HC 間のインタフェース方法の詳細について説明します。

2.3.1 List

一つの USB システムでは、USB ファンクションデバイスは最大 127 台接続可能であり、一つの USB ファンクションデバイスは最大 15 個の Endpoint を持つことができます。このように、Endpoint は複数存在することがありえるため、Endpoint 毎にその情報を纏める ED も複数存在しうることになります。OHCI 規格では、同一の転送タイプ毎に各 ED をリンクした ED 群を用意します。これを List と呼びます。また、TD には転送対象となる Endpoint が必ずあるため、TD はその TD の転送対象となる ED にキューイングされます。つまり、一つの ED に 1 つもしくは複数の TD が FIFO の形でキューイングされることになります。List は転送タイプ毎に**図** 2.2 のような ED・TD の集合体を持つことになります。

HCD は作成した List の先頭 ED のアドレスをその List へのポインタとして管理しています（**図** 2.2 の HeadPointer

に相当）。List の HeadPointer 情報を HC に渡す（HC のレジスタに書き込む）ことで、HC が List へのアクセスを可能とします。

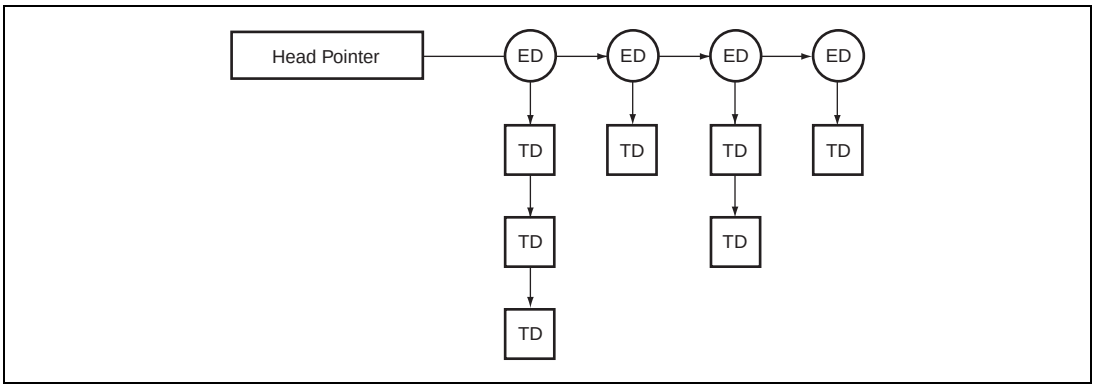

図 2.2　Typical List Strucure

（1） Non-Periodic 転送用 List

Non-Periodic 転送の Bulk 転送、Control 転送用 List はそれぞれ**図** 2.2 のような形になります。HeadPointer はそれぞれ一つであり、その情報は HC のレジスタ（HcControlHeadED、HcBulkHeadED）に格納されます。Non-Periodic であるため、転送は非同期的に発生しますが、HC が転送を実行する際にそれぞれの HeadPointer を読み込みます。

（2） Periodic 転送用 List

Periodic 転送である Interrupt 転送用 List と Isochronous 転送用 List は一体となっています。Interrupt 転送用 List の末尾の ED に Isochronous 転送用 List の先頭 ED がリンクされる形態になります。つまり、OHCI 規格では、Non-Periodic 転送である Control 転送用、Bulk 転送用の List と Periodic 転送用の List と 3 つの List があることになります。

Periodic 転送用 List は**図** 2.3 のようになっています。Non-Periodic 転送用の List とは違い、HeadPointer が 32 個あり、それぞれの HeadPointer は 32ms 毎（つまり 32Frame 毎）に参照されます。Periodic 転送用の ED は、必ず決まった間隔で参照される必要があるため、List を図 2.3 のような Tree 構造にします。

OHCI 規格では、Interrupt 転送の PollingRate は 32ms、16ms、8ms、4ms、2ms、1ms と固定になっており、Interrupt 用の ED はいずれかの PollingRate になります。一方 Isochronous 用 ED は、必ず 1ms 間隔でアクセスされる必要があるため、PollingRate が 1ms の Interrupt 用 ED にリンクします。**図** 2.3 のように、PollingRate が 32ms の Interrupt 用 ED はいずれかの HeadPointer にリンクすることで、32ms 毎に一度アクセスするようになります。PollingRate が 16ms の Interrupt 用 ED は、2 つの HeadPointer から参照されるようにリンクすることで、32ms に 2 回、つまり 16ms 毎に一度アクセスするようになります。同様にして、PollingRate が 8ms の Interrupt 用 ED は 4 つの HeadPointer から、4ms の Interrupt 用 ED は 8 つの Headpointer から、2ms の Interrupt 用 ED は 16 の HeadPointer からリンクされるようにします。そして、PollingRate が 1ms の Interrupt 用 ED はすべての HeadPointer からリンクされるようにします。そして、Isochronous 用 ED は PollingRate が 1ms の Interrupt 用 ED の後ろにリンクさせます。このように、Periodic 転送用の ED は ED を Tree 構造に構成します。

図 2.4 に Periodic 用 List の例を示します。この例では、Polling 間隔が 4ms の Interrupt 用 ED が 1 つ、その他の

32ms、16ms、8ms、2ms、1ms の Interrupt 用 ED がそれぞれ 2 つ、Isochronous 用 ED が 1 つという構成になっています。

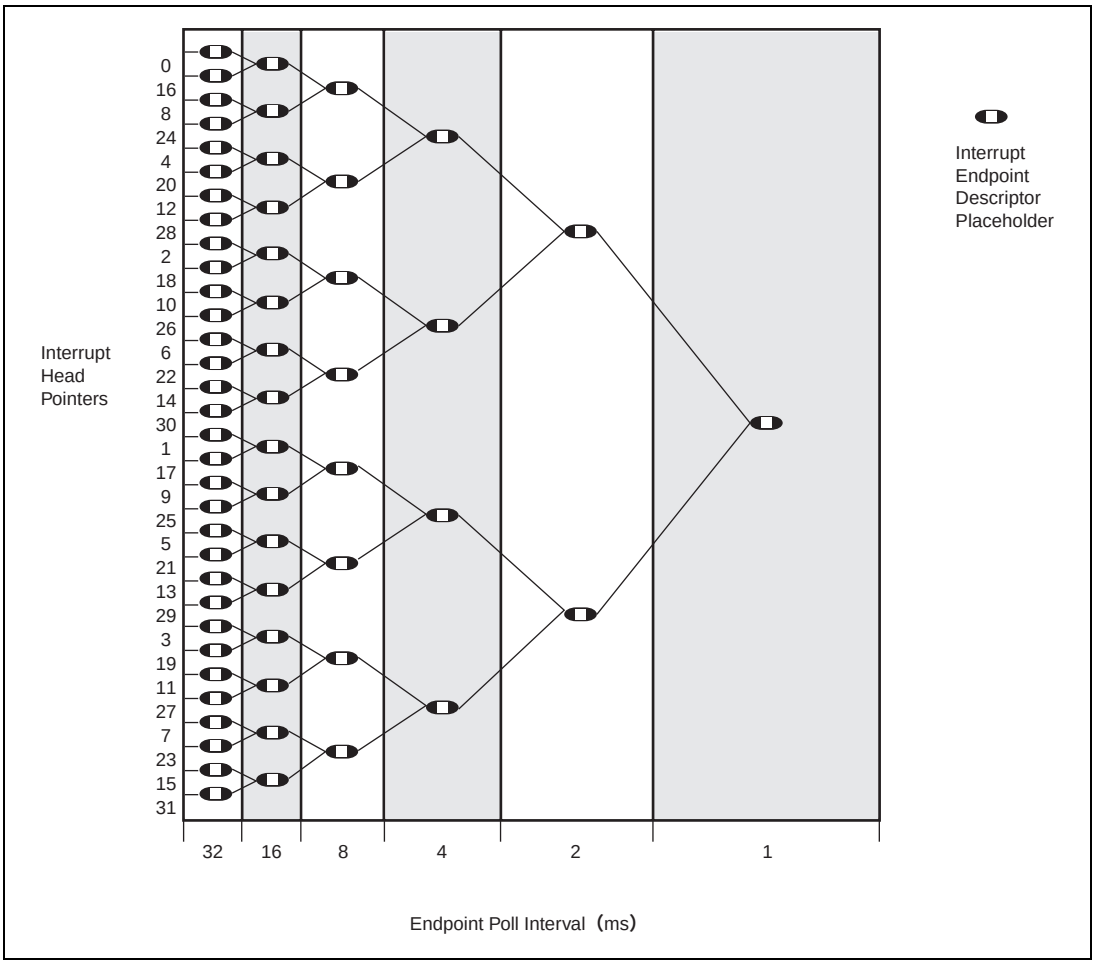

図 2.3　EndPoint Poll Interval（ms）

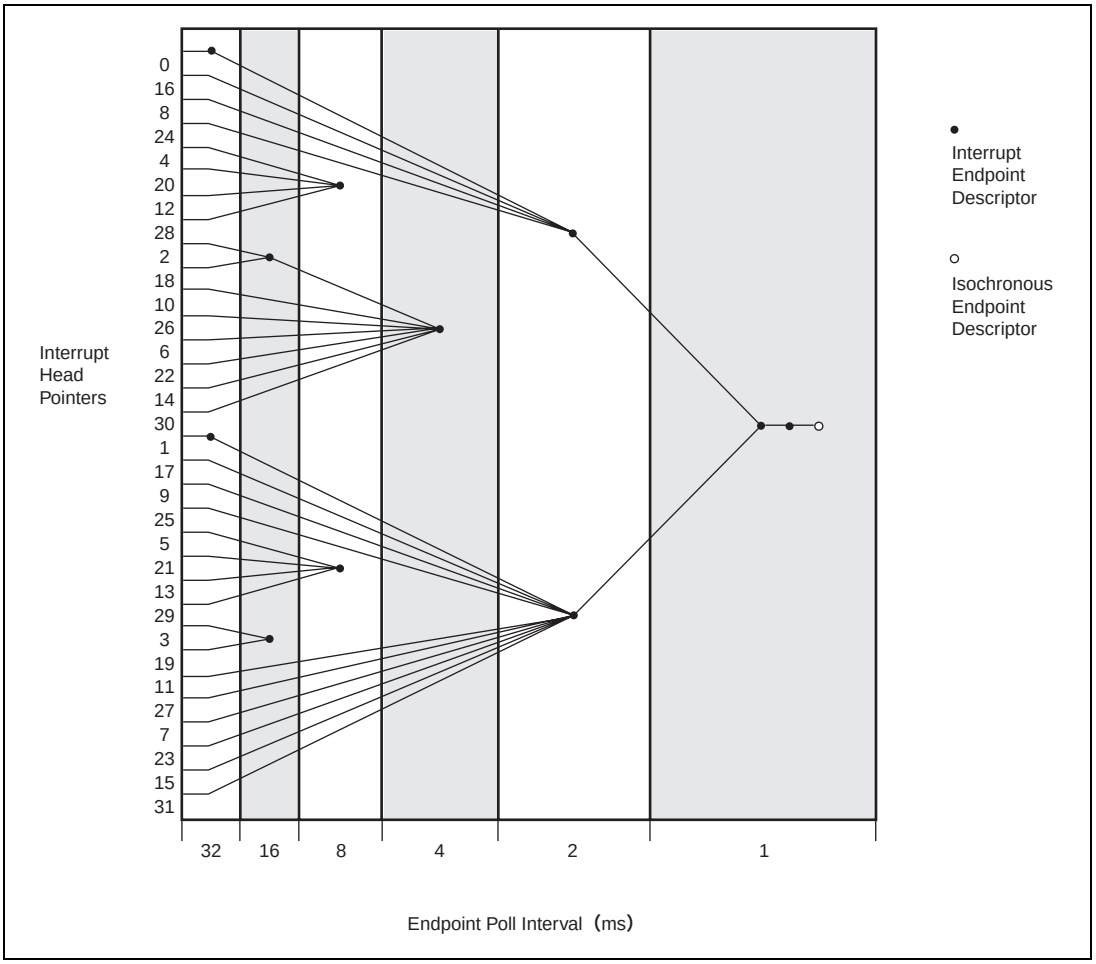

図 2.4　Example of Periodic ED List

2.3.2　Endpoint Descriptor（ED）

HCD は Endpoint と通信するため、通信の対象となる Endpoint ごとに Endpoint Descriptor（ED）を作成して、Endpoint と通信するために必要な情報を纏めておきます。ED は 16Byte 長であり、必ず 16Byte 境界上にセットする必要があります。**図** 2.5 に ED のフォーマットを、**表** 2.1 に各フィールドの詳細を示します。

ED は「2.3.1　List」のように、同一転送タイプの ED 同士をリンクさせて、List という形態に纏めます。これは、NextED フィールドに、次の ED のアドレスをセットしておくことで実現しています。NextED がゼロの場合、その ED にはリンクされている ED がないことを示しています。

HC は、まず Skip・Halt に 1 がセットされていないことを確認し、そして、TailP と HeadP が同一かどうか確認します。同一であれば、処理すべき TD がないと判断し、次の ED の処理に移ります。同一でない場合、HeadP に示されている TD の処理を始めます。TD のバッファエリアのデータを 1 パケットずつ送信、もしくはバッファエリアに受信パケットを書き込みます。TD の処理が完了したら、その TD は DoneQueue に移しつつ、その TD の NextTD フィールドの値を ED の HeadP にコピーします。

	31	26	25	24	23	22	21	10	7	6	5	4	3	2	1	0
Dword 0	−	MPS	F	K	S	D		EN	FA							
Dword 1	TD Queue Tail Pointer（TailP）														−	
Dword 2	TD Queue Head Pointer（HeadP）														0	C	H
Dword 3	Next Endpoint Descriptor（NextED）														−	

1. “ − ”のフィールドは、HCからアクセスされないフィールドです。HCによって値が修正されることもありません。そのため、HCDはこのフィールドを自由に使用できます。
2. “ 0 ”のフィールドは、HCDは必ず0を設定してからHCに処理させるようにします。

図 2.5　Endpoint Descriptor

表 2.1　Field Definitions for Endpoint Descriptor

Name	HC Access	Description
FA	R	FunctionAddress **通信対象となる** USB **ファンクションデバイスのアドレス**
EN	R	EndpointNumber **通信対象となる** Endpoint **番号**
D	R	Dorection **通信の方向（**INorOUT**）。方向は、**TD **の** PID **フィールドで規定することもできる。** Code / Direction 00b / Get direction From TD 01b / OUT 10b / IN 11b / Get direction From TD
S	R	Speed **対象となる** Endpoint **のスピード。**full-speed（S＝0）or low-speed（S＝1）
K	R	Skip **このビットをセットすることで、この** Endpoint **に関する通信は行わずに次に** ED **へ制御を移すことができる。**
F	R	Format **この** ED **にリンクされる** TD **のタイプを示す。**Control**、**Bulk**、**Interrupt **用** ED **と** Isochronous **用の** ED **では、用いる** TD **が異なる。前者の場合、**F＝0 **とし** GeneralTD **を使い、後者の場合は** F＝1 **とし** IsochronousTD **を使う。**
MPS	R	MaximumPacketSize **対象となる** Endpoint **が** 1 **つの** USB **パケットで受信できる、または送信できる最大値（**Byte**）を示す。**
TailP	R	TDQueueTailPointer **この** ED **にリンクされている最後の** TD **のアドレスが格納されている。** TailP **と** HeadP **が同一の場合、この** ED **には** HC **が処理すべき** TD **はないことになり、**TailP **と** HeadP **が異なる場合は、**HC **が処理すべき** TD **がまだ残っていることになる。**

Name	HC Access	Description
H	R/W	Halted HC がこの ED の処理を停止させたいときにセットされる。通常、TD の処理がエラーだったときにセットされる。
C	R/W	ToggleCarry TD がリタイアする場合、その最後のトグル情報（TD の data Toggle フィールドの LSb）がセットされる。Isochronous の場合、このフィールドは使用しない。
HeadP	R/W	TDQueueHeadPointer この ED にリンクされている最初の TD。このフィールドが示す TD の処理が完了するたびに、この TD にリンクされている次の TD アドレスに値が更新される。
NextED	R	NextED この ED にリンクされている ED へのアドレス。リンクする ED がない場合、0 をセットする。

HCD が ED の処理を一旦停止させたい場合、Skip ビットをセットします。HC はこのビットに 1 がセットされていると、この ED の処理を行わず、NextED に示されている ED の処理に移ります。

また、TD 処理中にエラーが発生した場合、HC は Halt ビットをセットし、そのエラーになった TD を DoneQueue に移し、HeadP をアップデートします。そして、エラー発生要因が解消できたら HCD が Halt ビットをクリアし、ED 処理を再開させます。

2.3.3 Transfer Descriptor（TD）

TD は HC がメモリ上に用意するデータ構造体で、Endpoint に対して送信する、もしくは Endpoint から受信するデータパケットについて定義するものです。TD には General Transfer Descriptor（GTD）と Isochronous Transfer Descriptor（ITD）の 2 つの種類があります。GeneralTD は、Control 転送、Bulk 転送、Interrupt 転送に用い、一方 IsoTD は Isochronous 転送に用います。GTD、ITD は共に、0～8192 Byte のバッファを持つことができます。

TD は、その転送対象となる Endpoint に対する ED にリンクします。HCD は、TD を生成し、ED にリンクさせます。その TD は HC によって処理され、処理終了後、HC によって ED から DoneQueue に移されます。

以下、GTD、ITD の詳細を示します。

（1） General Transfer Descriptor（GTD）

Control 転送、Bulk 転送、Interrupt 転送で用いる TransferDescriptor（TD）は共通になっており、General TD（GTD）と呼びます。そのフォーマットを**図** 2.6 に、各フィールドの詳細を**表** 2.2 に示します。この GTD は 16Byte 長であり、また必ず 16Byte 境界上にセットする必要があります。

	31–28	27–26	25–24	23–21	20–19	18	17–0
Dword 0	CC	EC	T	DI	DP	R	－
Dword 1	Current Buffer Pointer（CBP）						
Dword 2	Next TD（NextTD）						0
Dword 3	Buffer End（BE）						

1. HCがTDをアクセス中は、HCDはどのフィールドも修正しないでください。
2. “－”のフィールドはHCから書き込まれることはなく、HCが値を変更することはありません。

図 2.6 General Transfer Descriptor

表 2.2 Field Definitions for General TD

<table>
<tr><th>Name</th><th>HC Access</th><th>Description</th></tr>
<tr><td>R</td><td>R</td><td>bufferRounding
この TD から生成される最後のパケットがショートパケットになる際、DataUnderrun エラーが発生します。このフィールドに 1 を立てておくと、DataUnderrun エラーを無視します。</td></tr>
<tr><td>DP</td><td>R</td><td>Direction/PID
転送の方向と PID を設定します。
<table>
<tr><th>Code</th><th>PID Type</th><th>Data Direction</th></tr>
<tr><td>00b</td><td>SETUP</td><td>to endpoint</td></tr>
<tr><td>01b</td><td>OUT</td><td>to endpoint</td></tr>
<tr><td>10b</td><td>IN</td><td>form endpoint</td></tr>
<tr><td>11b</td><td>Reserved</td><td></td></tr>
</table></td></tr>
<tr><td>DI</td><td>R</td><td>DelayInterrupt
この TD が処理完了した際に発生する割り込み（WriteBackDoneHead）のタイミングを規定します。このフィールドの設定値だけの Frame を待ってから割り込みを発生します。例えば 0 ならすぐ割り込み発生し、1 なら TD 処理完了した Frame の 1Frame 後に割り込みを発生します。111b なら、割り込みは発生しません。</td></tr>
<tr><td>T</td><td>R/W</td><td>DataToggle
LSb で Data パケットの PID（Data0/Data1）を設定します。送信・受信が成功するたびにアップデートします。DataToggle 情報を ED の ToggleCarry フィールドから取得する場合は MSb を 0、このフィールドの LSb から取得する場合は MSb を 1 に設定します。</td></tr>
<tr><td>EC</td><td>R/W</td><td>ErrorCount
転送エラーが発生するたびに、この値がインクリメントされます。この値は 2 の場合に、転送エラーが発生した場合、エラーの種類を ConditionCode に記録し、この TD を DoneQueue に移動します。エラーなしに転送完了した場合は 0 になります。</td></tr>
<tr><td>CC</td><td>R/W</td><td>ConditionCode
この TD から生成された最後の転送のステータスを示します。</td></tr>
<tr><td>CBP</td><td>R/W</td><td>CurrentBufferPointer
Endpoint と転送（送信・受信）するためのバッファ領域を指します。常に、次にアクセスすべきバッファ領域のアドレスを指します。0 の場合は、サイズ 0 のデータを転送する場合か、もしくは、転送が完了した場合です。</td></tr>
<tr><td>NextTD</td><td>R/W</td><td>NextTD
次の TD へのポインタです。</td></tr>
<tr><td>BE</td><td>R</td><td>BufferEnd
バッファ領域の末尾アドレスを示します。</td></tr>
</table>

CurrentBufferPointer は、DataBuffer のアドレスを指しています。HC がこの TD がリンクされている ED で示される EndpointAddress に対して DataPacket を生成する際に参照されます。この TD から生成されたパケットが転送完了するたびに、ConditionCode に転送の成否情報が書き込まれます。そして、転送成功の場合、その転送したサイズだけ、CurrentBufferPointer の値をアップデートします。

DataToggle フィールドの MSb は、この TD から生成される最初の DataPacket の DataPID 情報を、この TD の

DataToggle の LSb から取得するのか、ED の ToggleCarry から取得するのかを示しています。MSb＝0 の場合、DataToggle の LSb は無視して ED の ToggleCarry の値を採用します。MSb＝1 の場合、DataToggle の LSb を採用します。

Bulk 転送、Interrupt 転送の場合、通常 DataToggle＝00b としておき、TD がリタイアして次の TD へ切り替わる際にトグル情報も引き継ぐようにします。TD がリタイアする時、DataToggle の LSb 値が ED の ToggleCarry へコピーされ、その値が次 TD の DataToggle の LSb へコピーされます。このようにして、トグル情報が引き継がれ、TD 切り替わりにも DataPID はトグルされることになります。

一方、Control 転送の場合、Setup ステージの Data Packet PID は Data0、Data ステージの最初のパケットは Data1、ステータスステージも Data1 となるようにする必要があります。よって、それぞれのステージ用 TD では、DataToggle の MSb は 1 にしておき、ED 経由で前の TD からトグル情報を受取るのではなく、DataToggle の LSb からトグル情報を受け取るようにします。

TD が完了するのは次の 2 つの場合があります。TD で指定した CurrentBufferPointer から BufferEnd のすべてのデータを送受信完了した場合と、Endpoint から MaxPacketSize 以下のサイズで受信した場合です。前者の場合、TD が ED から DoneQueue へ移されます。後者の場合は、BufferRounding ビットがセットされている場合は、前者と同様、通常の転送完了処理となります。しかし BufferRounding がセットされていない場合は、DataUnderrun エラーとなり、ED の Halt フィールドがセットされます。

（2） Isochronous Transfer Descriptor（ITD）

Isochronous Endpoint との転送には、GeneralTD ではなく、専用の IsochronousTD を用います。そのフォーマットを**図** 2.7、各フィールドの詳細を**表** 2.3 に示します。

	31–28	27	26–24	23–21	20–16	15–12	11–5	4–0
Dword 0	CC	－	FC	DI	－	SF		
Dword 1	Buffer Page 0（BP0）						－	
Dword 2	Next TD							0
Dword 3	Buffer End（BE）							
Dword 4	Offset1/PSW1					Offset0/PSW0		
Dword 5	Offset3/PSW3					Offset2/PSW2		
Dword 6	Offset5/PSW5					Offset4/PSW4		
Dword 7	Offset7/PSW7					Offset6/PSW6		

図 2.7　Isochronous TD Format

IsochronousTD から生成したデータパケットは、必ず特定のフレームに転送する必要があります。IsochrnousTD には、1～8 の連続したフレームで転送するデータをセットしておき、StartingFrame で指定されたフレームから、FrameCount＋1 のフレーム数だけ、連続的に各フレームでデータ転送を行います。

StartingFrame に対する HcFmNumber レジスタ（FN フィールド）の相対値を R とすると、R＝0 から R＝FrameCount になるまで、毎フレーム Isochronous データパケットを転送します。各フレームの転送開始アドレスは、Offset[R] から定められます。下位 12 ビットは Offset[R]の値になり、上位 20 ビットは、Offset[R]の 12 ビット目が 0 なら BufferPage0 の値、1 なら BufferEnd の上位 20 ビットになります。各フレームの転送終了アドレスは R＝0～R＝FrameCount－1 まで、つまり、最後のデータパケット以外では、終了アドレスは Offset[R＋1]－1 で決定できます。上位 20 ビットは転送開始アドレスと同様、Offset[R+1]の 12 ビット目が 0 なら BufferPage0 の値、1 なら BufferEnd

の上位 20 ビットになります。また、R＝FrameCount つまり最後のデータパケットは、終了アドレスは BufferEnd となります。

各フレームで転送完了した際は、それぞれの OffsetN/PSWN フィールドが PSWN 側のフォーマットに切り替わり、ConditionCode、SizeOfPacket の値が書き込まれます。

表 2.3 Field Definitions for Isochronous TD

Name	HC Access	Description
SF	R	StartingFrame Data Packet **の転送開始フレームを示します。**
DI	R	DelayInterrupt GeneralTD **と同様に、この** TD **が完了した際に、割り込み発生をどれだけ待たせるかを示します。**
FC	R	FrameCount **この** TD **から転送される** Data Packet **の個数（＝転送が行われるフレーム数）を示します。**FrameCount＝0 **は** 1 data packet **を示し、**FrameCount＝7 **は** 8 **を示します。**
CC	R/W	ConditionCode **この** IsochronousTD **が** DoneQueue **に移動した際の** ConditionCode **が格納されます。**
BP0	R	BufferPage0 **データバッファの先頭アドレスを示します。**
NextTD	R/W	NextTD **次の** IsochronousTD **へのアドレスを示します。**
BE	R	BufferEnd **バッファの最終アドレスを示します。**
OffsetN	R	Offset **それぞれの** Isochronous Data Packet **の先頭アドレスを示します。**

15	13	12	0
7		OFFSET	

Name	HC R/W	Description
OFFSET	R	OFFSET 13**ビット。下位**12**ビットで転送開始アドレスの下位**12**ビットを設定します。**12**ビット目で、上位**20**ビットの値の取得方法を選択します。**0**なら** BufferPage0**の値を、**1**なら**BufferEnd**の上位**20**ビットの値を使用します。**

<table>
<tr><th>Name</th><th>HC
Access</th><th>Description</th></tr>
<tr><td>PSWN</td><td>W</td><td>PacketStatusWord
それぞれの Isochronous Data Packet が転送完了した際の ConditionCode を格納します。
<table>
<tr><td>1
5</td><td>1
2</td><td>1
1</td><td>0
0</td></tr>
<tr><td colspan="2">CC</td><td colspan="2">SIZE</td></tr>
</table>
<table>
<tr><th>Name</th><th>HC
R/W</th><th>Description</th></tr>
<tr><td>SIZE</td><td>W</td><td>Size of Packet
11ビット。IN転送時は、受信したサイズを示します。OUT転送時は0が書かれます。</td></tr>
<tr><td>CC</td><td>W</td><td>Condition Code
このフィールドがNotAccessedを示している場合、OffsetN/PSWNはOffsetのフォーマットになります。NotAccessed以外の場合、PacketStatusWordのフォーマットになります。</td></tr>
</table>
</td></tr>
</table>

（3） Condition Code

TD の ConditionCode フィールドに設定されうる値の詳細を表 2.4 に示します。GeneralTD の場合は、TD が DoneQueue に移動する際に設定されます。IsochronousTD の場合、ConditionCode は、Dword0 内の ConditionCode フィールド、Offset/PacketStatusWord フィールドの 2 箇所に示されます。それぞれのデータパケットの転送が完了した際は Offset/PacketStatusWord で示され、TD が DoneQueue に移動する際は Dword0 の ConditionCode で示されます。

表 2.4 Condition Code

Code	Meaning	Description
0000	NoError	エラー検出なしにデータパケット処理が終了したことを示す
0001	CRC	Endpoint からの最後のデータパケットに CRC エラーが含まれていたことを示す
0010	BitStuffing	Endpoint からの最後のデータパケットに bit stuffing violation が含まれていたことを示す
0011	DataToggleMismatch	Endpoint からの最後のデータパケットの Data Toggle PID が誤っていたことを示す
0100	Stall	Endpoint から Stall PID が返ってきて、TD が DoneQueue に移されたことを示す
0101	DeviceNotResponding	デバイスが、IN トークンに無反応だった、もしくは OUT Handshake を供給されなかったことを示す
0110	PIDCheckFailure	Endpoint からの PataPID（IN）、Handshake（OUT）が誤った PID だったことを示す
0111	UnexpectedPID	受信した PID が無効の値だったことを示す
1000	DataOverrun	Endpoint の MaxPacketSize 以上のデータパケットを受信したこと、もしくは、トータル受信サイズが予定していた転送サイズ以上であったことを示す

Code	Meaning	Description
1001	DATAUNDERRUN	MaxPacketSize 以下のサイズのデータパケットを受信したこと、もしくは、トータル転送サイズが予定していた転送サイズ以下であったことを示す
1010	reserved	
1011	reserved	
1100	BUFFEROVERRUN	IN 転送時、Endpoint からのデータ受信速度が、SystemMemory への書き込み速度よりも速かったことを示す。IsochronousTD のみ。
1101	BUFFERUNDERRUN	OUT 転送時、USB の転送転送レートを保持するために十分な速度で SystemMemory のリードアクセスが行えなかったことを示す。IsochronousTD のみ。
111x	NOT ACCESSED	TD が生成・リスト化される際にソフトウェアがセットする。ConditionCode がこの値であれば、まだ HC に処理されていないことを示す

エラーが起こった場合、そのエラーに対応する ConditionCode 値が TD の ConditionCode フィールドにセットされ、ED の Halt ビットがセットされます。

エラーは次の 4 種類に分類できます。

- Transmission Error
- Sequence Error
- System Error
- Time Error

Transmission Error は USB パス上の情報にエラーが発生するもので、CRC、BITSTUFFING、DATATOGGLEMISMATCH、DEVICENOTRESPONDING、PIDCHECKFAILURE、UNEXPECTEDPID の各エラーがこれに属します。このエラーの場合は、すぐには DoneQueue に移されず、再転送を試みます。このエラーが 3 回起こった場合、もしくは 2 回起こったあと他のエラーが発生したときは DoenQueue に移されます。

Sequence Error は Endpoint からの受信データが予定のサイズと異なる際に発生します。STALL、DATAOVERRUN、DATAUNDERRUN がこれに当たります。このエラーは一度発生したらすぐに DoneQueue に移されます。

System Error は、HC のシステム環境に問題がある際に発生します。BUFFEROVERRUN、BUFFERUNDERRUN がこれに当たります。このエラーは IsochronousTD の場合のみ発生し、GeneralTD の場合は発生しません。

Time Error も、IsochronousTD の場合のみに発生します。これには、Skip Packet と Late Retirement の 2 つのエラータイプがあります。IsochronousTD のそれぞれのデータパケットは特定のフレームで実行される必要があり、そのため、あるフレームで Isochoronous データパケットが送信できない場合があります。転送されるべきフレームでデータパケットが処理できなかった場合、Skip packet となり、予定されていたデータパケットは ConditionCode=NotAccessed のままになります。そのパケットが処理されないまま次のパケットの処理に移ります。途中のフレームで Skip Packet が発生しても、IsochronousTD からの最後のデータパケットが処理成功なら、Dword0 の ConditionCode には NoError が入り、TD がリタイアします。しかし、最後のデータパケットで Skip Packet が発生した場合は、転送完了予定フレームの次のフレームで再度処理され、Dword0 の ConditionCode には DATAOVERRUN が入り、TD がリタイアします。ただしこの場合は ED が Halt になることなく、すぐに次の IsochronousTD を処理できます。

2.3.4 Host Controller Communication Area（HCCA）

Host Controller Communication Area（HCCA）は HCD と HC の間で様々な情報をやり取りするためのものです。フォーマットを図 2.8 に示します。HCCA は 256Byte 長であり、必ず 256Byte 境界上に領域を確保する必要があります。この HCCA によって、HCD は HC に直接アクセスすることなく、メモリアクセスのみで、HCCA からの情報を取得できます。

表 2.5 Host Controller Communications Area Format

Offset	Size (bytes)	Name	HC R/W	Description
0	128	HccaInterrruptTable	R	32 個の Interrupt ED へのポインタ群
0x80	2	HccaFrameNumber	W	現在のフレーム番号。このフィールドは、各フレームの ED 処理開始前に HC がアップデートします。
0x82	2	HccaPad1	W	HC が HccaFrameNumber をアップデートした際に、0 がセットされます。
0x84	4	HccaDoneHead	W	フレームの終了時、WriteBackDoneHead 割り込みが有効になっていた場合、HC はこのフィールドに HcDoneHead 値を書き込みます。一度 HC が書き込むと、Software がこのフィールドをクリアして、HcInterruptStatus の WD ビットをクリアしない限り、HC はこのフィールドに書き込みません。 また、このフィールドの LSb が 1 の場合、HC がこのフィールドに書き込んだ際、WriteBackDoneHead 以外の割り込みも発生したことを示します。
0x88	116	reserved	R/W	Reserved

HccaInterruptTable には 32 個の InterruptED へポインタ情報をセットします。「2.3.1 List」のように Peroidic 転送用 List は 32 個の HeadPointer を持ちます。そのポインタを格納するための場所として、HccaInterruptTable があります。HC は、毎フレーム HccaInterruptTable をアクセスして、32 個のポインタのうち、いずれか一つのポインタを取得します。

HccaFrameNumber は、HC によって毎フレームアップデートされます。HcFrameNumber をアップデートして SOF を発行した後で、最初に処理すべき ED を読み込む前に、HccaFrameNumber をアップデートします。

HccaDoneHead は、HcDoneHead の値が書き込まれます。TD 処理が完了した際、DelayInterrupt で示された値だけの Frame 数を待って、その次のフレーム開始時に HccaDoneHead へ書き込まれます。

2.3.5 List Processing

図 2.10 のように、4 つの ED（ED1、ED2、ED3、ED4）があり、そのそれぞれに一つの以上の TD がリンクされている場合を考えます。HC は、一つの ED にリンクされているすべての TD がリタイアするまで一つの ED を処理し続けるわけではなく、また、一つの TD がリタイアするまでその TD を実行し続けるわけでもありません。HC は、それぞれの ED の先頭 TD からデータパケットを一つ生成・転送するという処理を繰り返し、どの ED にも偏ることなく処理を進めます。

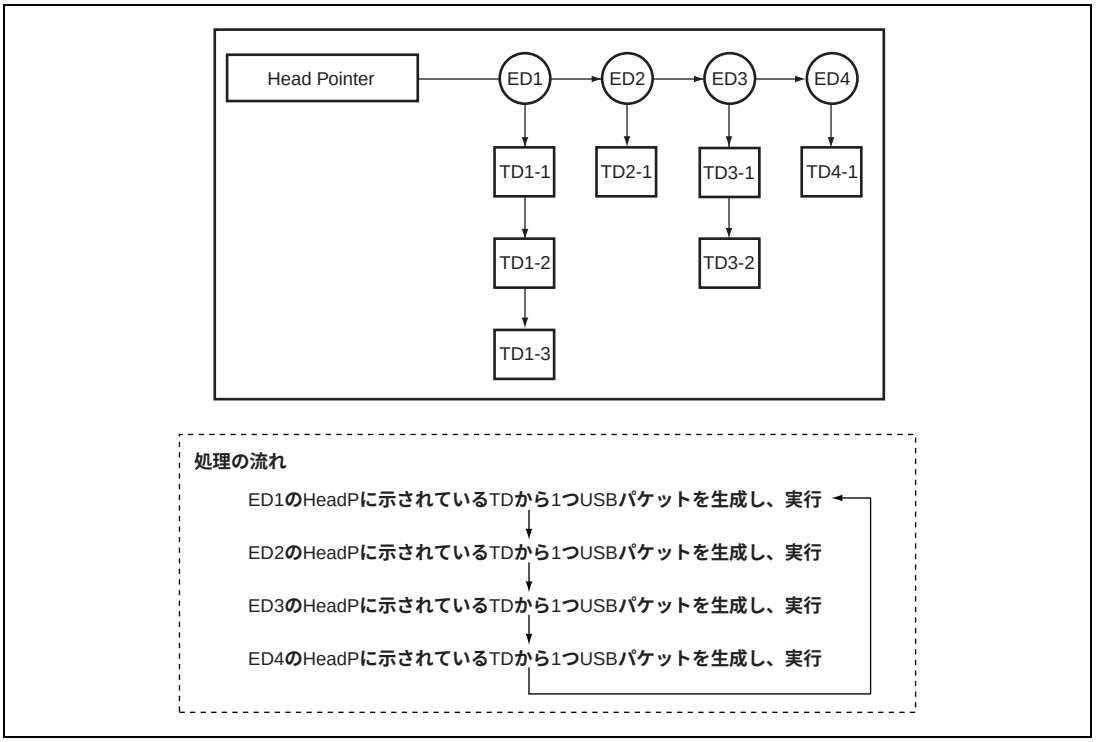

図 2.8 List Processing

2.3.6 DoneQueue

HC は、転送処理が完了した TD をリンクさせて List 化しておきます。これを DoneQueue と呼びます。**図** 2.11 に DoneQueue の一例を示します。

HC が処理すべき List は「List（初期状態）」で示されている List のみで、Frame R 中にこの List の TD1-1、TD2-1、TD3-1、TD4-1 の 4 つが転送完了になった場合を考えます。TD が転送完了するたびに、TD が DoneQueue にリンクされることになります。図のように、DoneQueue は、常に転送完了したものが先頭にくるようにリンクします。常に、最後に転送完了した TD が先頭に、転送完了が一番古い TD が末尾にリンクされています。

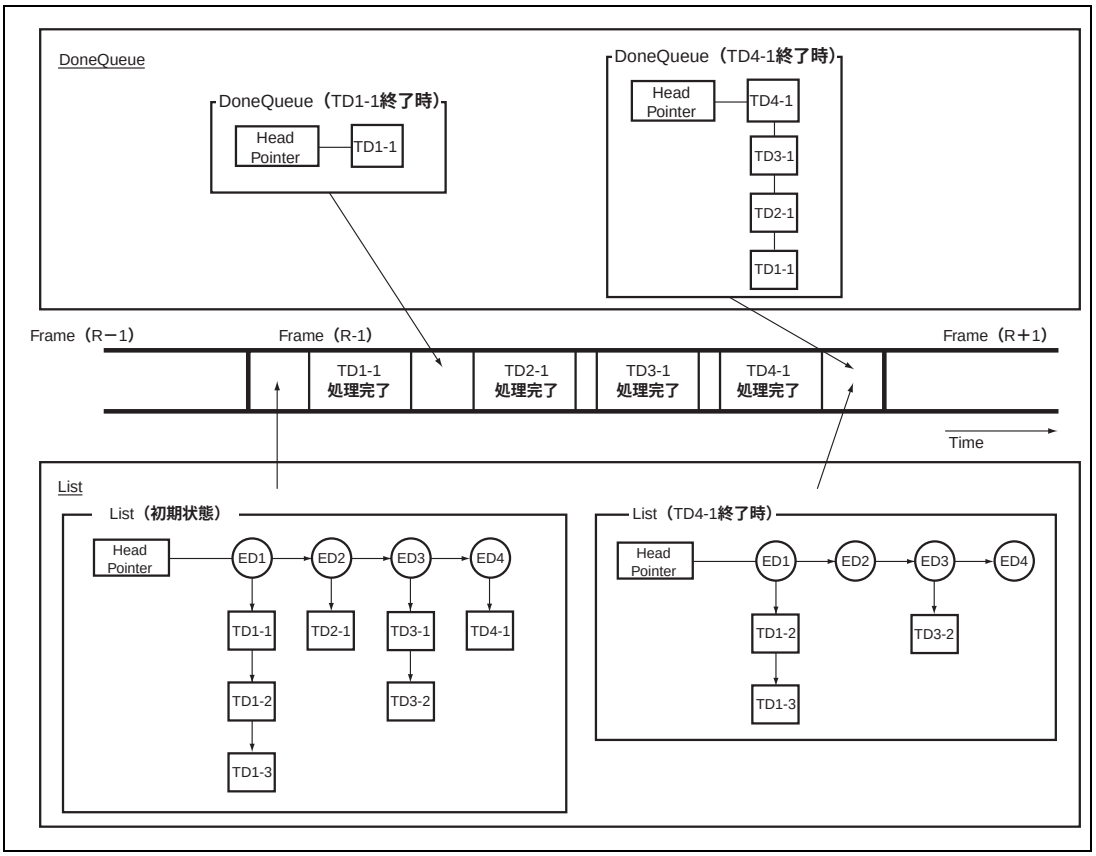

図 2.9　DoneQueue

図 2.12 に転送処理が完了した TD が DoneQueue にリンクされる様子を示します。**図** 2.12 の TD1 が転送完了すると、HC は以下の処理を行います。

1. EDのHeadPフィールドに、転送完了したTD（TD1）のNextTDの値を書き込む
2. 転送完了したTD（TD1）のNextTDフィールドに、HcDoneHeadレジスタの値を書き込む
3. HcDoneHeadレジスタに、転送完了したTD（TD1）の先頭アドレスを書き込む

これらの処理によって、転送完了した TD1 を DoneQueue にリンクさせることができます。

さらに、TD2 が転送完了した場合も同様に、上記 1~3 の操作を行うことで、転送完了した TD2 を DoneQueue にリンクさせることができます。

転送完了した TD を DoneQueue にリンクさせるのは HC が行います。WriteBackDoneHead 割り込みが発生した際、HC は HcDoneHead の値を HccaDoneHead に書き込みます。HCD は、この割り込みを受けて、HccaDoneHead を読むことで、転送完了した TD 群を知ることができます。

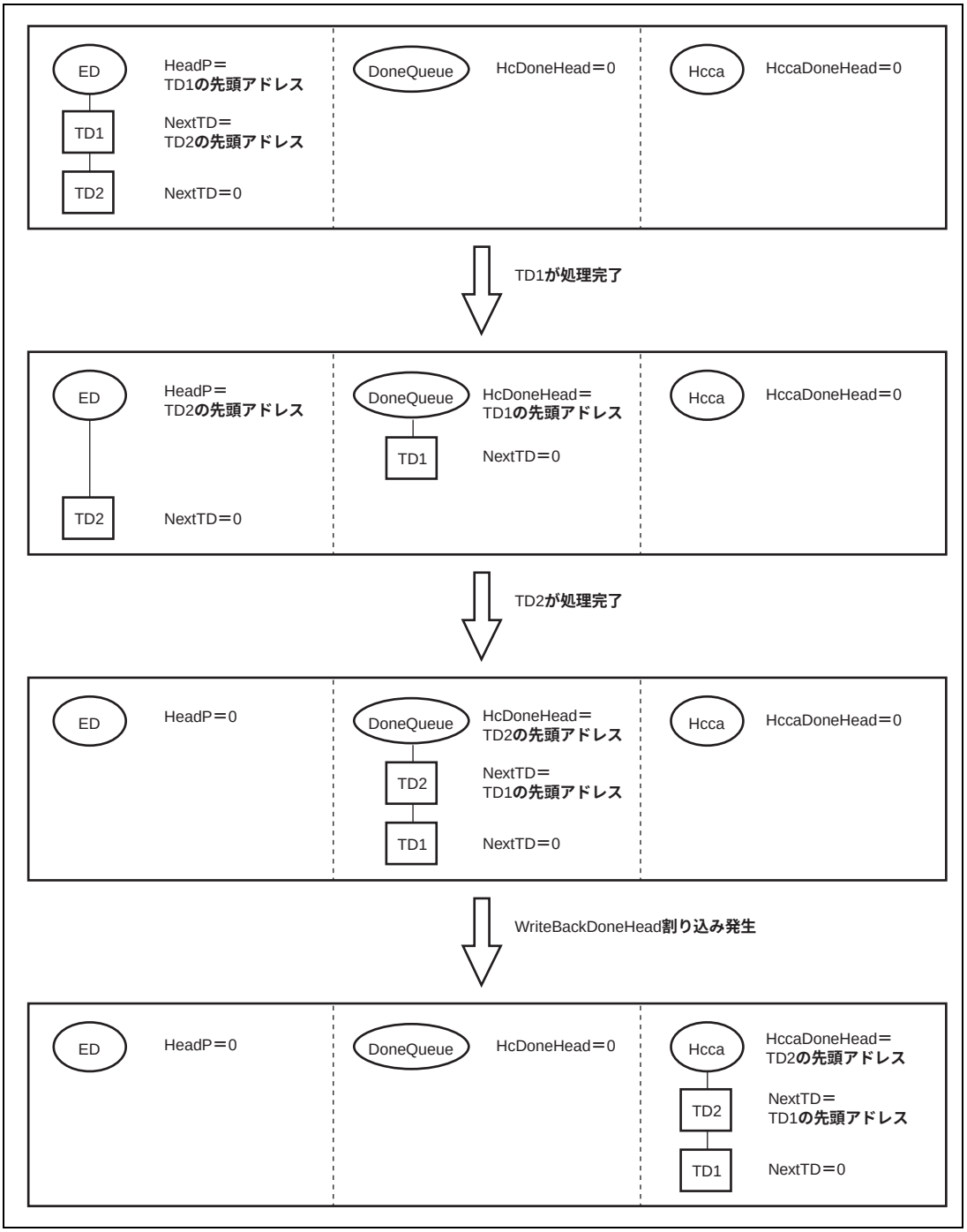

図 2.10　DoneQueue **の動作**

2.3.7 Communication Channels

HC と HCD の間の通信チャネルは 2 つあります。ひとつは、HC のレジスタ経由で通信する方法で、もうひとつはメモリ上の HCCA 経由で通信する方法です。

HCD と HC 間では、ED、TD をやり取りします。HCD が転送対象である ED、転送データである TD を作成して、それらを List として纏め、その List を HC に渡します。HC は渡された ED・TD から USB パケットを生成し、ED で示された Endpoint に転送します。そして、転送完了した TD を DoneQueue としてキューイングしておきます。そして定期的に、処理済みの TD 群である DoneQueue を HCD に返します。

つまり、HCD－HC 間では、Non-Periodic 転送である Control、Bulk の 2 つの List、Periodic 転送用の List、DoneQueue という 4 つの情報をやり取りします。それぞれ、先頭アドレスである HeadPointer を受け渡しすることで両者間の通信を行います。Non-Periodic の 2 つの HeadPointer は、レジスタ経由、Periodic の HeadPointer 群と DoneQueue の HeadPointer は HCCA 経由、つまりメモリ経由で List を受け渡しします。それを**図** 2.13 に示します。

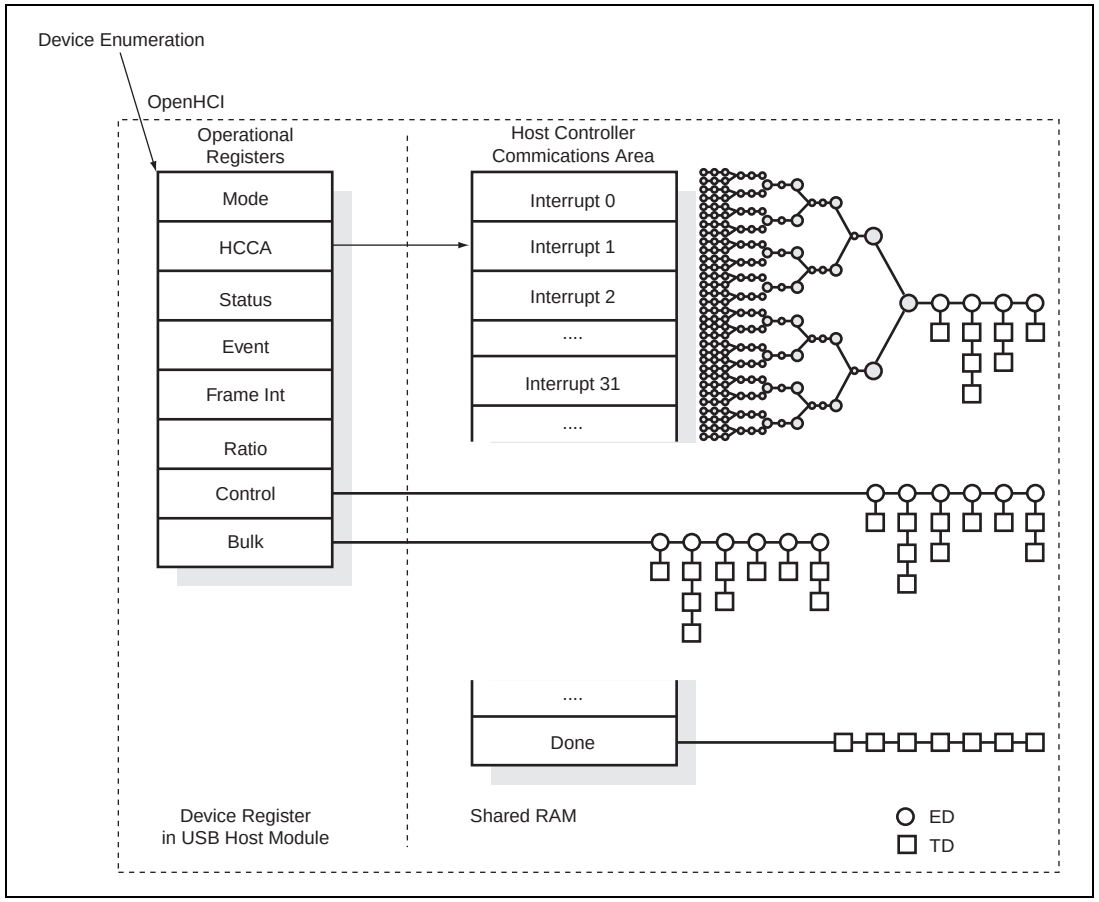

図 2.11 Communication Channels

2.4 Host Controller Driver Responsibilities

2.4.1 Host Controller Management

HCD は、HC を管理・コントロールします。HC のレジスタを直接アクセスしたり、Interrupt の EDList の HeadPointer 群を HCCA に登録したりすることで、HC をコントロールします。

そのため、HCD は、HC の状態についての情報、各 List の HeadPointer の情報、List 処理の Enable/Disable、割り込みの Enanle/Disable などの情報を保持します。

2.4.2 Bandwidth Allocation

USB では、1.0ms 間隔でフレームを生成しています。OHCI 規格では、**図** 2.14 のようにフレーム内を 3 つのパートに分けて、Non-PeriodicList を処理する期間、PeriodicList を処理する期間を分けます。それぞれのフレームで、SOF パケット発行が終了後、HcFmRemaining の FrameRemaining フィールドが、HcFmInterval の FrameInterval フィールドの値になるまで、Non-Periodic 転送を行います。そして、FrameInterval フィールドの値を超えたら、Periodic 転送を行い、すべての Periodic 転送が終わると、再び Non-Periodic 転送を行います。

HCD は Periodic 転送の要求に対して、十分な Bandwidth があるかどうか判定し、その要求を受け入れるかどうか判定します。

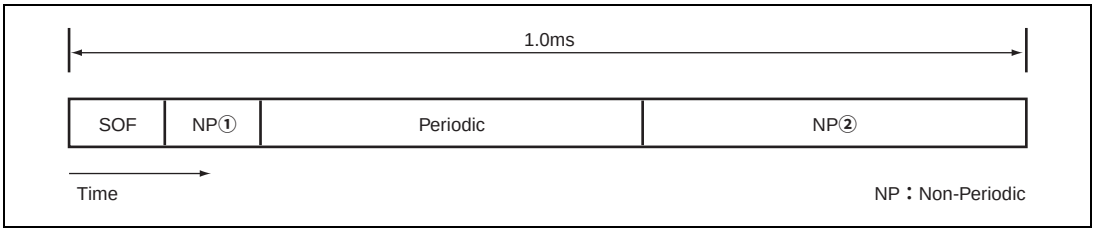

図 2.12 Frame Bandwidth Allocation

2.4.3 List Management

USB Data Packet は、ED にリンクされている TD から生成されます。HCD は、この ED、TD のリンク構造つまり List を作成します。

新規に ED を追加する際は、List 中の末尾に追加すれば良く、HC の List 処理を一旦停止させることなしに簡単に追加できます。一方、既存の ED を削除する際は、HC による List 処理を停止させてから、所望の ED を削除し、再び List 処理を有効にします。List 処理の Enable/Disable は HcControl レジスタの ControlListEnable、BulkListEnable、PeriodicListEnable、IsochronousListEnable で制御することが可能で、本ビットを 0 クリアした後の次の SOF 送信後に List 処理が停止します。

TD の追加、削除についても ED と同様で、削除の際は List 処理を停止させる必要があります。

2.4.4 RootHub

HC には RootHub に関するレジスタも含まれており、HCD は HC だけでなく、RootHub もコントロールする必要があります。

2.5 Host Controller Responsibilities

2.5.1 USB State

OHCI 規格では、4 つのステート（Operational、Reset、Suspend、Resume）を規定しています。HC は、USB バスを最適なステートに設定します。

- Operational状態

  SOFトークンを発行できて、各Listを処理できる状態。

- Reset状態

  ハードウェアリセット後、まずこの状態に遷移する。USBバスがリセット状態になっている。SOFトークンは発行されず、List処理もFrame番号のインクリメントも行われない。

- Suspend状態

  SOFトークンは発行されず、List処理もFrame番号のインクリメントも行われない。HCのRemote WakeUp回路がWakeUp信号がアクティブになるのを待っている状態。

- Resume状態

  Suspend状態からのみ遷移。HCDによるレジスタアクセスか、もしくはRootHubからRemoteWakeUp信号を受けることで、この状態に遷移する。

2.5.2 Frame Management

各フレームの開始時に、SOF パケットを生成し、メモリ上の FrameCounter をアップデートします。

2.5.3 List Processing

HC は HCD によって生成された ED、TD を実行します。各フレームの Periodic 転送期間で、HCCA から 32 個のうち 1 つの InterruptList HeadPointer を読み込んで、Interrupt ED List、Isochronous ED List を実行します。List 中のすべての ED について、それぞれの ED にリンクされている最初の TD から 1 つ USB パケットを生成、転送します。

また、Non-Periodic 転送期間では、Control、Bulk それぞれの List を処理します。ControlList 処理と BulkList 処理は交互に実行されます。2 つの List の切り替えタイミングは、HCD で設定します。HCD が定めた比率で、Control 転送ｎ回実行、Bulk 転送１回実行、Control 転送ｎ回実行…と切り替えながら Non-Periodic 期間が終わるまで、ControlList、BulkList を実行し続けます。

転送成功・失敗に関わらず、処理完了した TD は、ED のリンクから外し DoneQueue に移します。DoneQueue は処理完了した TD 群がリンクされていますが、常に一番最後に転送終了した TD から順々にリンクさていています。HC は、転送終了した TD の NextTD フィールドに、DoneQueue の先頭アドレスにある TD をセットし、その今終了完了した TD を DoneQueue にセットします。DoneQueue の情報は、定期的に HCCA 経由で HC から HCD へ渡されます。

2.6 レジスタ仕様

HC は、**表** 2.5 に示すようなレジスタを持っています。いずれのレジスタも HCD から 32Bit アクセスされます。HC のレジスタ群は「Control and Status」「Memory Pointer」「Frame Counter」「RootHub」の 4 つに大別できます。以下、それぞれのレジスタを説明します。

表 2.6 Host Controller Operational Registers

Offset	3 0 1 0
0	HcRevision
4	HcControl
8	HcCommandStatus
C	HcInterruptStatus
10	HcInterruptEnable
14	HcInterruptDisable
18	HcHCCA
1C	HcPeriodCurrentED
20	HcControlHeadED
24	HcControlCurrentED
28	HcBulkHeadED
2C	HcBulkCurrentED
30	HcDoneHead
34	HcFmInterval
38	HcFmRemaining
3C	HcFmNumber
40	HcPeriodicStart
44	HcLSThreshold
48	HcRhDescriptorA
4C	HcRhDescriptorB
50	HcRhStatus
54	HcRhPortStatus[1]
...	...
54+4*NDP	HcRhPortStatus[NDP]

【注】 * NDP：Number Downstream Ports（SH7727 **の場合、**NDP＝2）

2.6.1 Control and Status Partition

（1） HcRevision Register

31 … 08	07 … 00
reserved	REV

図 2.13 HcRevision Register

表 2.7 HcRevision Regi

Key	Reset	Read/Write		Description
		HCD	HC	
REV	10h	R	R	Revision HC **の** Revision **を示すリードオンリーのフィールドです。たとえば** H'11 **だと** Version1.1 **を示します。現在は** HC **の** Revision **は** 1.0 **しかないため、このフィールドは必ず** H'10 **です。**

（2） HcControl Regsiter

このレジスタは、HC の動作モードを設定します。

31 … 11	10	09	08	07 06	05	04	03	02	01 00
reserved	RWE	RWC	IR	HCFS	BLE	CLE	IE	PLE	CBSR

図 2.14 HcControl Register

表 2.8 HcControl Register

Key	Reset	Read/Write		Description
		HCD	HC	
CBSR	00b	R/W	R	ControlBulkServiceRatio HC **が処理する** ControlED **と** BulkED **の比率を規定します。**Non－Periodic **期間中、**HC **はこのフィールドで規定する比率に従って、**1～4 **個の** ControlED **を処理後、**BulkED **をひとつ処理します。**Non－Periodic **期間中、これを繰り返します。**HC **は一つの** ControlED **の処理が完了後、このフィールドで規定する比率に従って、別の** ControlED **を処理するか、**BulkED **を処理するかを決めます。**

CBSR	No. of Control EDs Over Bulk EDs Served
0	1 : 1
1	2 : 1
2	3 : 1
3	4 : 1

Key	Reset	Read/Write		Description
		HCD	HC	
PLE	0b	R/W	R	PeriodicListEnable 次のフレームで PeriodicList を HC に処理させるかどうかを指定します。このビットの変更は、次の SOF 以後有効になります。HC は PeriodicList を処理し始める際、必ずこのビットをチェックする必要があります。
IE	0b	R/W	R	IsochronousEnable 次のフレームで IsochronousED を HC に処理させるかどうかを指定します。このビットの変更は、次の SOF 以後有効になります。Periodic 期間中、HC が IsochronousED を処理する際、まずこのビットをチェックします。このビットがセットされていると IsochronousED を処理し、セットさていなければ PeriodicList 処理を終了し（PeriodicList に IsochronousED のみが設定されている場合）、Non－Periodic 転送に移ります。
CLE	0b	R/W	R	ControlListEnable 次のフレームで ControlList を処理させるかどうかを指定します。このビットの変更は、次の SOF 以後有効になります。HC は ControlList を処理するときは常にこのビットをチェックします。ControlList を修正したい場合、このビットをクリアします。HcControlCurrentED に示されている ControlED を削除する際は、再び ControlList 処理を開始する前に、必ず HcControlCurrentED の値を更新してください。
BLE	0b	R/W	R	BulkListEnable 次のフレームで BulkList を処理させるかどうかを指定します。このビットの変更は、次の SOF 以後有効になります。HC は BulkList を処理するときは常にこのビットをチェックします。BulkList を修正したい場合、このビットをクリアします。HcBulkCurrentED に示されている BulkED を削除する際は、再び BulkList 処理を開始する前に、必ず HcBulkCurrentED の値を更新してください。
HCFS	00b	R/W	R/W	HostControllerFunctionalState HC のステートを設定します。 00b：UsbReset 01b：UsbResume 10b：UsbOperational 11b：UsbSuspend HCD は、他のステートから USBOPERATIONAL に遷移させることで、1ms 後に SOF パケットを生成できます。HC は USBSUSPEND のときのみステートを変更できます。USBSUSPEND で DownStream ポートから Resume 信号を検出したときに、USBRESUME へ遷移させることができます。 ソフトウェアリセット時には USBSUSPEND になり、ハードウェアリセット時には USBRESET になります。後者の場合、RootHub もリセットします。

Key	Reset	Read/Write		Description
		HCD	HC	
IR	0b	R/W	R	InterruptRouting HcInterruptStatus レジスタに示される割り込みの経路を設定します。このビットがセットされていない場合、すべての割り込みは通常の経路となります。このビットをセットすると、SMI（System Management Interrupt）に設定されます。このビットは HC のオーナーシップを示す標識として用いられます。
RWC	0b	R/W	R/W	RemoteWakeupConnected このビットは、HC が Remote Wakeup 信号をサポートするかどうかを示します。Remote Wakeup がサポートされていてシステムで使用する際は、POST（Power On Self Test）期間中にこのビットをセットします。HC はハードウェアリセット時にはこのビットをクリアします。ソフトウェアリセットでは変更されません。
RWE	0b	R/W	R	RemoteWakeupEnable このビットはアップストリームの RemoteWakeup 信号の検出による RemoteWakeup 機能を有効にするかどうかを設定します。このビットがセットされていて、HcInterruptStatus の ResumeDetect ビットがセットされている場合、ホストシステムへ RemoteWakeup 信号を生成します。

（3） HcCommandStatus Register

このレジスタでは、HCD が HC に List 処理要求、HC リセット要求、オーナーシップチェンジ要求を行います。また、スケジュールオーバーランエラーを検出したフレーム数をカウントします。

31–18	17–16	15–04	03	02	01	00
reserved	SOC	reserved	OCR	BLF	CLF	HCR

図 2.15　HcCommandStatus Register

表 2.9　HcCommandStatus Register

Key	Reset	Read/Write		Description
		HCD	HC	
HCR	0b	R/W	R/W	HostControllerReset HC に対してソフトウェアリセットを実行させ、ほとんどのレジスタが初期化される USBSuspend 状態に移行します。ただし HcControl レジスタの InterruptRouting フィールドは初期化されず、また、この Suspend 状態ではホストバスアクセスはできません。このビットは、リセット処理が完了した際に HC はクリアします。リセット処理は 10μsec 以内に完了させる必要があります。

Key	Reset	Read/Write		Description
		HCD	HC	
CLF	0b	R/W	R/W	ControlListFilled このビットは、ControlList に処理すべき TD が存在するかどうかを示します。HCD が ControlList 中の ED に TD を追加した際に、このビットをセットして、HC に処理すべき TD があることを知らせます。 HC は ControlList の先頭 ED を処理を開始する際、このビットをまずチェックします。このビットが 0 である限り、HC は ControlList を処理しません。セットされると、このビットをクリアして List 処理を開始します。そして、HC が ED 中に TD を見つけた場合、このビットをセットして List 処理を続けます。TD を見つけられなかった場合、このビットは 0 のままであり、ControlList 処理は停止します。
BLF	0b	R/W	R/W	BulkListFilled このビットは、BulkList に処理すべき TD が存在するかどうかを示します。HCD が BulkList 中の ED に TD を追加した際に、このビットをセットして、HC に処理すべき TD があることを知らせます。 HC は BulkList の先頭 ED を処理を開始する際、このビットをまずチェックします。このビットが 0 である限り、HC は BulkList を処理しません。セットされると、このビットをクリアして List 処理を開始します。そして、HC が ED 中に TD を見つけた場合、このビットをセットして List 処理を続けます。TD を見つけられなかった場合、このビットは 0 のままであり、BulkList 処理は停止します。
OCR	0b	R/W	R/W	OwnershipChangeRequest このビットは HC の制御権を要求するために、OS 側の HCD によってセットされます。このビットがセットされると、HC は HcInterruptStatus の OwnershipChange ビットをセットします。制御権が移行完了すると、このビットはクリアされ、次に OS 側 HCD から制御権移行を要求されるまでこの状態を維持します。
SOC	00b	R	R/W	SchedulingOverrunCount スケジュールオーバーランエラーが発生するたびにインクリメントされます。初期値は B'00 で、B'11 から再び B'00 に戻ります。HCD はこのフィールドを監視することで、現在のスケジューリングの問題を知ることができます。

（4） HcInterruptStatus Register

このレジスタは、ハードウェア割り込みを起こす様々なイベントの状態を示します。イベントが発生すると、その対象となるビットをセットします。そのビットがセットされた場合、HcInterruptEnable において、そのイベントが有効になっていて、MasterInterruptEnable もセットされていた場合、ハードウェア割り込みが発生します。

ビットをクリアするためには、HCD によって 1 を書き込みます。HCD はクリアできますが、セットはできません。逆に HC はセットはできますが、クリアはできません。

31	30	29 … 07	06	05	04	03	02	01	00
0	OC	reserved	RHSC	FNO	UE	RD	SF	WDH	SO

図 2.16 HcInterruptStatus Register

表 2.10 HcInterruptStatus Register

Key	Reset	Read/Write		Description
		HCD	HC	
SO	0b	R/W	R/W	SchedulingOverrun **現在のフレームでオーバーランが発生し、**HccaFrameNumber **をアップデートした後にこのビットをセットされます。この割り込み発生時には、**HcControlStatus **の** ScheduleOverrunCount **フィールドもインクリメントされます。**
WDH	0b	R/W	R/W	WritebackDoneHead HcDoneHead **の値を** HccaDoneHead **に書き込んだ直後にこのビットがセットされます。**HCD **は、**HccaDoneHead **の値を読み込んだ後に、このビットをクリアします。このビットがクリアされるまでは、**HC **が再び** HcDoneHead **の値を** HccaDoneHead **に書き込むことはありません。**
SF	0b	R/W	R/W	StartofFrame **このビットは、毎フレームの開始時に** HccaFrameNumber **がアップデートされた後にセットされます。**HC **が同時に** SOF **トークンを生成します。**
RD	0b	R/W	R/W	ResumeDetected **このビットは** USB **バス上のデバイスがレジューム信号をアサートした際にセットされます。**HCD **が** HcControl **レジスタの** HostControllerFunctionalState **フィールドを書き換えることで** HC **を** USBResume **状態にセットした際は、このビットはセットされません。**
UE	0b	R/W	R/W	UnrecoverableError **このビットは** USB **に無関係のシステムエラーが発生した際にセットされます。**HC **はシステムエラーは解消されるまで全ての処理を行いません。**HCD **は** HC **をリセットした後にこのビットをクリアします。**
FNO	0b	R/W	R/W	FrameNumberOverflow **このビットは、**HcFmNumber **の** MSb **（**15 **ビット目**） **が変化して、**HccaFrameNumber **がアップデートされた後にセットされます。**

Key	Reset	Read/Write		Description
		HCD	HC	
RHSC	0b	R/W	R/W	RootHubStatusChange **このビットは、**HcRhStatus**、**HcRhPortStatus **に何らかの変化が発生した際にセットされます。**
OC	0b	R/W	R/W	OwnershipChange **このビットは、**HCD **が** HcControlStatus **の** OwnershipChangeRequest **フィールドをセットした際に、**HC **によってセットされます。このイベントは、常に** SystemManagementInterrupt（SMI）**を生成します。** SMI **ピンがない場合は、このビットは** 0b **のままです。**

（5） HcInterruptEnable Register

HcInterrupt レジスタの各ビットは HcInterruptStatus の各ビットに対応しており、ハードウェア割り込みを発生するイベントを制御するために使われます。HcInterruptStatus のビットがセットされ、その対象となるこの HcInterruptEnable のビットもセットされていて、さらに MasterInterruptEnable ビットもセットされている場合、ハードウェア割り込みが発生します。

各ビットにセットするには 1 を書き込みます。ただし 0 を書き込んでもクリアできません。クリアをするには HcInterruptDisable レジスタの対象ビットに 1 を書き込んでください。

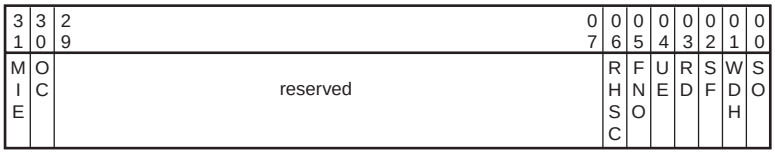

図 2.17　HcInterruptEnable Register

表 2.11　HcInterruptEnable Register

Key	Reset	Read/Write		Description
		HCD	HC	
SO	0b	R/W	R	0：**無視されます。** 1：Scheduling Overrun **による割り込みを有効にします。**
WDH	0b	R/W	R	0：**無視されます。** 1：HcDoneHead Writeback **による割り込みを有効にします。**
SF	0b	R/W	R	0：**無視されます。** 1:Enable interrupt generation due to Start of Frame **による割り込みを有効にします。**
RD	0b	R/W	R	0：**無視されます。** 1：Resume Detect **による割り込みを有効にします。**
UE	0b	R/W	R	0：**無視されます。** 1：Unrecoverable Error **による割り込みを有効にします。**

Key	Reset	Read/Write		Description
		HCD	HC	
FNO	0b	R/W	R	0：無視されます。 1：Frame Number Overflow による割り込みを有効にします。
RHSC	0b	R/W	R	0：無視されます。 1：Root Hub Status Change による割り込みを有効にします。
OC	0b	R/W	R	0：無視されます。 1：Ownership Change による割り込みを有効にします。
MIE	0b	R/W	R	0：無視されます。 1：このレジスタの他のビットによる割り込みを有効にします。 HCD は本ビットによってマスタ割り込みイネーブル機能として使用することが可能です。

（6） HcInterruptDisable Register

このレジスタの各ビットは HcInterruptStatus の各ビットに対応しています。このレジスタは、HcInterruptEnable レジスタと対になっており、このレジスタのビットに 1 を立てることで、対応する HcInterruptEnable のセットをクリアできます。0 を書き込んだ際は、HcInterruptEnable の対象ビットは変化しません。読み出した際は、HcInterruptEnable レジスタの現在値が得られます。

31	30	29 … 07	06	05	04	03	02	01	00
MIE	OC	reserved	RHSC	FNO	UE	RD	SF	WDH	SO

図 2.18　HcInterruptDisable Register

表 2.12　HcInterruptDisable Register

Key	Reset	Read/Write		Description
		HCD	HC	
SO	0b	R/W	R	0：無視されます。 1：Scheduling Overrun による割り込みを無効にします。
WDH	0b	R/W	R	0：無視されます。 1：HcDoneHead Writeback による割り込みを無効にします。
SF	0b	R/W	R	0：無視されます。 1：Start of Frame による割り込みを無効にします。
RD	0b	R/W	R	0：無視されます。 1：Resume Detect による割り込みを無効にします。
UE	0b	R/W	R	0：無視されます。 1：Unrecoverable Error による割り込みを無効にします。

Key	Reset	Read/Write		Description
		HCD	HC	
FNO	0b	R/W	R	0：無視されます。 1：Frame Number Overflow による割り込みを無効にします。
RHSC	0b	R/W	R	0：無視されます。 1：Root Hub Status Change による割り込みを無効にします。
OC	0b	R/W	R	0：無視されます。 1：Ownership Change による割り込みを無効にします。
MIE	0b	R/W	R	0：無視されます。 1：このレジスタの他のビットによる割り込みを無効にします。 ハードウェア、ソフトウェアリセット後、本ビットはセットされます。

2.6.2 Memory Pointer Partition

（1） HcHCCA Register

このレジスタは、Host Controller Communication Area(HCCA)の物理アドレスを保持しています。HCCA は 256Byte 境界にセットする必要があるため、このレジスタの下位 8 ビットは常に 0 が読み出されます。

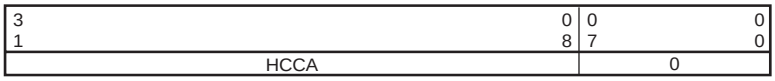

図 2.19 HcHCCA Register

表 2.13 HcHCCA Register

Key	Reset	Read/Write		Description
		HCD	HC	
HCCA	0h	R/W	R	Host Controller Communication Area のベースアドレスを示します。

（2） HcPeriodCurrentED Register

このレジスタは、現在の IsochronousED もしくは InterruptED の物理アドレスを示しています。

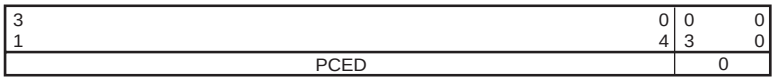

図 2.20 HcPeriodCurrentED Register

表 2.14 HcPeriodCurrentED Register

Key	Reset	Read/Write		Description
		HCD	HC	
PCED	0h	R	R/W	PeriodCurrentED 現在のフレームで処理する一つの PeriodicList の先頭アドレスを示します。HC は一つの PeriodicED の処理が終わるたびに、このレジスタの値をアップデートします。HCD は、このレジスタを読むことで、どの ED が現在処理されているかを知ることができます。

（3） HcControlHeadED Register

このレジスタは、Control List の先頭 ED の物理アドレスを示します。

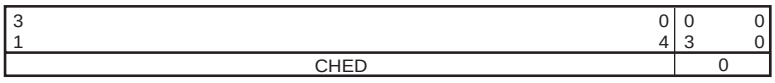

図 2.21 HcControlHeadED Register

表 2.15 HcControlHeadED Register

Key	Reset	Read/Write		Description
		HCD	HC	
CHED	0h	R/W	R	ControlHeadED HC は、このフィールドに示されている ED から Control List の処理を開始します。

（4） HcControlCurrentED Register

このレジスタは、現在処理中の Control ED の物理アドレスを示します。

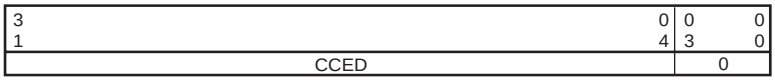

図 2.22 HcControlCurrentED Register

表 2.16 HcControlCurrentED Register

Key	Reset	Read/Write		Description
		HCD	HC	
CCED	0h	R/W	R/W	ControlCurrentED このフィールドは、現在処理中の ED へのポインタを示しています。現在の ED の処理が終わると、このポインタを次の ED へと進めます。ControlList の末尾 ED に達した際は、HcCommandStatus の ControlListFilled ビットをチェックします。このビットがセットされていた場合、HcControlHeadED の値を HcControlCurrentED にコピーして、ビットをクリアします。セットされていない場合、何も処理しません。 HCD がこのレジスタの値を修正するには、HcControl の ControlListEnable ビットをクリアする必要があります。ControlListEnable ビットがセットされている場合は、HCD はリードのみ行えます。

(5) HcBulkHeadED Register

このレジスタは、Bulk List の先頭 ED の物理アドレスを示します。

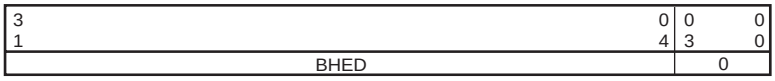

図 2.23 HcBulkHeadED Register

表 2.17

Key	Reset	Read/Write		Description
		HCD	HC	
BHED	0h	R/W	R	BulkHeadED HC は、このフィールドに示されている ED から Bulk List の処理を開始します。

(6) HcBulkCurrentED Register

このレジスタは、現在処理中の Bulk ED の物理アドレスを示します。

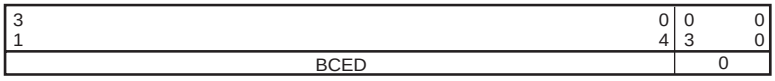

図 2.24 HcBulkCurrentED Register

表 2.18 HcBulkCurrentED Register

Key	Reset	Read/Write		Description
		HCD	HC	
BCED	0h	R/W	R/W	BulkCurrentED このフィールドは、現在処理中の ED へのポインタを示しています。現在の ED の処理が終わると、このポインタを次の ED へと進めます。BulkList の末尾 ED に達した際は、HcCommandStatus の BulkListFilled ビットをチェックします。このビットがセットされていた場合、HcBulkHeadED の値を HcBulkCurrentED にコピーして、ビットをクリアします。セットされていない場合、何も処理しません。 HCD がこのレジスタの値を修正するには、HcControl の BulkListEnable ビットをクリアする必要があります。BulkListEnable ビットがセットされている場合は、HCD はリードのみ行えます。

（7） HcDoneHead Register

このレジスタは、リードした時点で、一番最後に DoneQueue へ追加された TD の物理アドレスを示します。HCD はこのレジスタの値は HCCA から得るので、通常は HCD がこのレジスタをアクセスすることはありません。

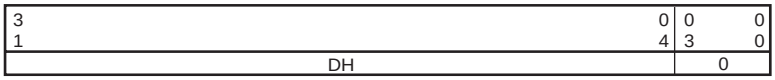

図 2.25 HcDoneHead Register

表 2.19 HcDoneHead Register

Key	Reset	Read/Write		Description
		HCD	HC	
DH	0h	R	R/W	DoneHead TD の処理が完了した場合、HC は HcDoneHead の値をその TD の NextTD フィールドに書き込みます。そして、その TD のアドレスを HcDoneHead に書き込みます。 HC がこのフィールドの値を HCCA に書き込んだ際に、0 にクリアされます。その際は、HcInterruptStatus の WriteBackDoneHead もセットされます。

2.6.3 Frame Counter Partition

（1） HcFmInterval Register

このレジスタは、フレームビットタイム間隔(連続する SOF の間隔)を示す 14 ビット値と、Scheduling Overrun が発生せずに転送できる最大のパケットサイズ情報を示す 15 ビット値が保持されています。HCD は毎フレームの間隔を微調整できます。このレジスタによって、外部クロック源との同期化、未知のローカルクロックと同期化をする際に、ソフトウェアレベルで調整することができます。

31	16	15 14	13 0
FIT	FSMPS	reserved	FI

図 2.26 HcFmInterval Register

表 2.20 HcFmInterval Register

Key	Reset	Read/Write		Description
		HCD	HC	
FI	2EDFh	R/W	R	FrameInterval **二つの連続した** SOF **の間隔を示します。初期値は** 11,999 **となります。** HC **をリセットすると初期値に戻るため、**HCD **が** HcCommandStatus **の** HostControllerReset **をセットして** HC **をリセットする際には、リセットに元の値に戻すには、あらかじめこのフィールド値を保持する必要があります。**
FSMPS	TBD	R/W	R	FSLargestDataPacket **各フレームの最初に** Largest Data Packet Counter **にロードされる値を示します。このカウンタ値は、**scheduling overrun **を起こさずに転送できるパケットの最大サイズを示します。この値は** HCD **により計算されます。**
FIT	0b	R/W	R	FrameIntervalToggle HCD **は** FrameInterval **の値を更新するたびに、この値をトグルします。**

（2） HcFmRemaining Register

このレジスタは、現在のフレーム残りビットタイムを示す 14 ビットのダウンカウンタです。

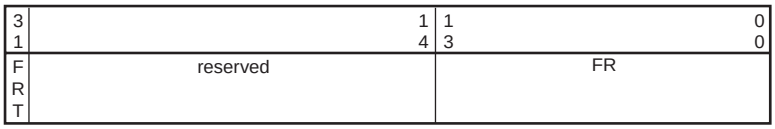

図 2.27　HcFmRemaining Register

表 2.21　HcFmRemaining Register

Key	Reset	Read/Write		Description
		HCD	HC	
FR	0h	R	R/W	FrameRemaining **ビットタイムを示します。0 になった場合、次のフレームで** HcFmInterval **の** FrameInterval **の値をロードします。**USBOperationa **状態に遷移する際は、**HcFmInterval **の** FrameInterval **の値を読み込んで、そのアップデートされた値を次の** SOF **から用います。**
FRT	0b	R	R/W	FrameRemainingToggle **このビットは、**FrameRemaining **が 0 になるときはいつでも** HcFmInterval **の** FrameIntervalToggle **ビットの値をコピーします。** **これらのビットは、**FrameInterval **と** FrameRemaining **の間の同期をとるために使用します。**

（3） HcFmNumber Register

このレジスタは 16 ビットカウンタで、現在のフレーム番号を示します。HCD はこのレジスタと FrameNumberOverrun 割り込みを用いて、32 ビットのフレーム番号を生成することができます。

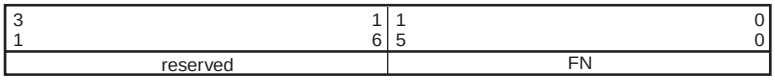

図 2.28　HcFmNumber Register

表 2.22

Key	Reset	Read/Write		Description
		HCD	HC	
FN	0h	R	R/W	FrameNumber HcFmRemaining **がリロードされる毎にインクリメントされます。** H'FFFF **の後は** H'0 **に戻ります。**USBOperational **状態に遷移すると、自動的にカウントアップされます。各フレームの境界で、**SOF **を発行後、最初の** ED **を処理する前までに、**HC **はこの値を** HCCA **に書き込みます。** HCCA **に書き込んだ後、**HC **は** HcInterruptStatus **の** StartOfFrame **をセットします。**

（4） HcPeriodicStart Register

このレジスタは、PeriodicList の処理を開始するタイミングを示します。

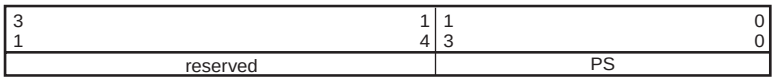

図 2.29　HcPeriodicStart Register

表 2.23　HcPeriodicStart Register

Key	Reset	Read/Write		Description
		HCD	HC	
PS	0h	R/W	R	PeriodicStart **ハードウェアリセット後、このフィールドはクリアされます。通常、この値は、**HcFmInterval **から** 10%Off **した程度の値に設定されます。典型的な値は** H'2A2F **です。**HcFmRemaining **がこの値に達した場合、現在処理中の** Control **転送、**Bulk **転送が完了次第、**PeriodicList **を処理し始めます。**

（5） HcLSThreshold Register

このレジスタは、HC が EOF 前に最大 8Byte の LS パケットを送信するかどうか判断する際に参照されます。

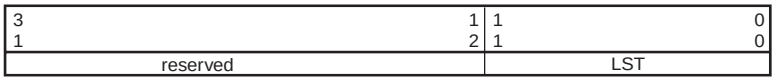

図 2.30　HcLSThreshold Register

表 2.24　HcLSThreshold Register

Key	Reset	Read/Write		Description
		HCD	HC	
LST	0628h	R/W	R	LSThreshold LowSpeed **転送を行う際、このフィールド値と** FrameRemaining **の値が比較されます。**LowSpeed **転送は、「**FrameRemaining ≥ LSThreshold**」の場合のみ開始されます。このフィールドの値は、**HCD **が転送とセットアップのオーバーヘッドを考慮して計算します。**

2.6.4 Root Hub Partition

(1) HcRhDescriptorA Register

このレジスタは、RootHub の特徴を設定するレジスタで、その一つ目のレジスタです。

31-24	23-13	12	11	10	09	08	07-00
POTPGT	reserved	NOCP	OCPM	DT	NPS	PSM	NDP

図 2.31　HcRhDescriptorA Register

表 2.25　HcRhDescriptorA Register

Field	Power On Reset	Read/Write		Description
		HCD	HC	
NDP	IS	R	R	NumberDownstreamPorts RootHub **が持つダウンストリームポートの数を示します。最小値は** 1 **で、最大値は** 15 **です。**
NPS	IS	R/W	R	NoPowerSwitching **パワー切り替えをサポートするか、常にパワーオンかを示します。このビットがクリアされた場合、つまりパワー切り替えが有効の場合は、**PowerSwitchMode **によって、パワーモードを設定します。** 0**：ポートのパワーは切り替え可能** 1**：（**HC **がパワーオンであり限り）ポートのパワーは常にオン。**
PSM	IS	R/W	R	PowerSwitchingMode RootHub **ポートのパワースイッチの方法を設定します。このビットは、**NoPowerSwitch **がクリアされている場合のみ有効です。** 0**：すべてのポートは同時にパワー供給されます。** 1**：それぞれのポートは個々にパワー供給されます。この場合、ポートパワーは** GlobalSwitch**、**Per-PortSwitch **のいずれかの方法によって制御されます。**PortPowerControlMask **ビットがセットされていると、ポートは、**port power commands（Set/ClearPortPower）**のみに反応します。クリアされている場合、**global power switch（Set/ClearGlobalPower）**でのみ制御できます。**
DT	0b	R	R	DeviceType **このビットは、**RootHub **が複合デバイスであるかどうかを示します。**RootHub **は複合デバイスとなることは許可されないため、このビットは常に** 0 **にセットしてください。**

Field	Power On Reset	Read/Write		Description
		HCD	HC	
OCPM	IS	R/W	R	OverCurrentProtectionMode このビットは、RootHub ポートのオーバーカレント状態のレポート方法を示します。リセット時、PowerSwitchMode と同じ設定になります。このビットは、NoOverCurrentProtection ビットがクリアされているときのみ有効です。 0：オーバーカレント状態は、すべてのダウンストリームポートを集約して報告される。 1：オーバーカレント状態は、個々のポートごとに報告される。
NOCP	IS	R/W	R	NoOverCurrentProtection このビットは、RootHub ポートのオーバーカレント状態のレポート方法を示します。このビットがクリアされている場合、OverCurrentProtectionMode によって、Global か Per-Port かを設定します。 0：オーバーカレント状態は、すべてのダウンストリームポートを集約して報告される。 1：オーバーカレントはサポートしない。
POTPGT	IS	R/W	R	PowerOnToPowerGoodTime このビットは、HCD がパワーオンとなったポートにアクセスする際に、必要となる待ち時間を示します。2ms 単位であり、待ち時間は『このフィールド値*2ms』として計算されます。

（2） HcRhDescriptorB Register

このレジスタは、RootHub の特徴を設定するレジスタで、その二つ目のレジスタです。

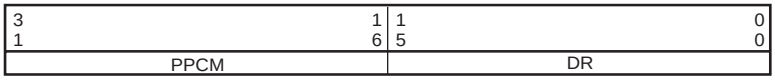

図 2.32　HcRhDescriptorB Register

表 2.26 HcRhDescriptorB Register

Field	Power-On Reset	Read/Write		Description
		HCD	HC	
DR	IS	R/W	R	DeviceRemovable それぞれのビットは RootHub の各ポートに対応しています。各ビットは、クリアされている場合、接続されているデバイスがリムーバブルであることを示します。セットされている場合、リムーバブルにはなりません。 bit 0：Reserved bit 1：Port #1 bit 2：Port #2 ... bit15：Port #15
PPCM	IS	R/W	R	PortPowerControlMask それぞれのビットは、各ポートが PowerSwitchMode がセットされている場合に、グローバルパワーコマンドによって影響を受けるかどうかを示します。セット時は、ポートのパワー状態は Per-port のパワー制御（Set/ClearPortPower）のみによって影響を受けます。クリア時は、ポートはグローバルパワースイッチ（Set/ClearGlobalPower）によってコントロールされます。グローバルパワーモード（PowerSwitchMode＝0）の場合は、このフィールドは無効です。 bit 0：Reserved bit 1：Port #1 のパワーをマスク bit 2：Port #2 のパワーをマスク ... bit15：Port #15 のパワーをマスク

（3） HcRhStatus Register

このレジスタでは、RootHub のステータスを設定します。

31	18	17	16	15	14	02	01	00
CRWE	reserved	OCIC	LPSC	DRWE	reserved		OCI	LPS

図 2.33 HcRhStatus Register

表 2.27 HcRhStatus Register

Field	Root Hub Reset	Read/Write		Description
		HCD	HC	
LPS	0b	R/W	R	（read）LocalPowerStatus RootHub はローカルパワーステータス機能をサポートしません。このビットは常に 0 が読み出されます。 （write）ClearGlobalPower グローバルパワーモード（PowerSwitchingMode＝0）で、1 を書き込むことですべてのポートをパワーオフにします（PortPowerStatus をクリア）。per-port モードでは、このビットをクリアすることで、PortPowerControlMask がセットされていないポートの PortPowerStatus をクリアします。0 の書き込みは無効です。
OCI	0b	R	R/W	OverCurrentIndicator オーバーカレント状態が存在するかどうかを示します。グローバルモードの時、セットされていると、オーバーカレント状態が存在することを示します。セットされていない場合は、パワー状態は正常です。Per-Port モードの時は、このビットは常に 0 です。
DRWE	0b	R/W	R	（read）DeviceRemoteWakeupEnable このビットは RemoteWakeup が有効にするかどうかを示します。セットされている場合、レジュームイベントによって ConnectStatusChange ビットが有効になり、USBSuspend 状態から USBResume 状態への遷移が起こり、ResumeDetect 割り込みが発生します。 0：ConnectStatusChange は RemoteWakeup イベントでは変化しない。 1：ConnectStatusChange は RemoteWakeup イベントでは変化する。 （write）SetRemoteWakeupEnable 1 を書き込むことで、DeviceRemoveWakeupEnable をセットします。0 の書き込みは無効です。
LPSC	0b	R/W	R	（read）LocalPowerStatusChange RootHub はローカルパワーステータス機能はサポートしません。このビットは常に 0 が読み出されます。 （write）SetGlobalPower グローバルパワーモード（PowerSwitchingMode＝0）の時、1 を書き込むことで、すべてのポートをパワーオンにすることができます（PortPowerStatus をクリア）。Per-Port パワーモードの時、PortPowerStatus は、PortPowerControlMask ビットがセットされていないときのみセットされます。0 の書き込みは無効です。
OCIC	0b	R/W	R/W	OverCurrentIndicatorChange HC が OCI フィールドに変更があった場合にセットします。HCD はクリアするために 1 を書き込みます。

Field	Root Hub Reset	Read/Write		Description
		HCD	HC	
CRWE	－	W	R	（write）ClearRemoteWakeupEnable 1 を書き込むことで、DeviceRemoteWakeupEnable をクリアできます。 0 の書き込みは無効です。

（4） HcRhPortStatus[1:NDP] Register

このレジスタは、ポートごとに存在し、各ポートイベントの設定、レポートを行います。下位ワードでポートの状態を設定します。上位ワードでは、ポートの状態変化をモニタできます。

31 21	20	19	18	17	16	15 10	09	08	07 05	04	03	02	01	00
reserved	PRSC	OCIC	PSSC	PESC	CSC	reserved	LSDA	PPS	rsvd	PRS	POCI	PSS	PES	CCS

図 2.34 HcRhPortStatus Register

表 2.28 HcRhPortStatus Register

Field	Root Hub Reset	Read/Write		Description
		HCD	HC	
CCS	0b	R/W	R/W	（read）CurrentConnectStatus **このビットは、ダウンストリームポートの現在の状態を示します。** 0：**デバイスは接続されていない。** 1：**デバイスが接続されている。** （write）ClearPortEnable HCD **はこのビットに** 1 **を書き込むことで、**PortEnableStatus **ビットをクリアします。**0 **の書き込みは無効です。**CurrentConnectStatus **はどのような書き込みにも影響されません。** **【注】このビットは接続されたデバイスがリムーバブルデバイスでない場合**（DeviceRemovable[NDP]）**、常に** 1b'**が読み出されます。**

Field	Root Hub Reset	Read/Write		Description
		HCD	HC	
PES	0b	R/W	R/W	（read）PortEnableStatus このビットはポートが有効か無効かを示します。RootHub はこのビットを、オーバーカレント、切断イベント、パワーオフスイッチ、バスエラー発生の場合にクリアします。また、PortEnabledStatusChange がセットされた時も、このビットが変化します。HCD は、SetPortEnable、ClearPortEnable に書き込むことで、このビットを変更できます。このビットは、CurrentConnectStatus がクリアされている場合はセットできません。このビットは、ResetStatusChange がセットされてポートリセットが完了した際、もしくは SuspendStatusChange がセットされてポートサスペンドが完了した際にもセットされます。 0：ポートは無効。 1：ポートは有効。 （write）SetPortEnable このビットに 1 を書き込むことで、PortEnableStatus をセットできます。0 の書き込みは無効です。CurrentConnectStatus がクリアされている場合、このビットに 1 を書き込んでも PortEnableStatus はセットされません。しかし、その代わりに ConnectStatusChange がセットされます。これにより、切断されているポートに接続を試みたことを HCD に知らせることができます。
PSS	0b	R/W	R/W	（read）PortSuspendStatus このビットはポートがサスペンド状態、もしくはレジュームシーケンスにいることを示します。このビットは SetSuspendState への書き込みでセットされ、レジューム期間の最後に PortSuspendStatusChange がセットされることでクリアされます。このビットは、CurrentConnectStatus がクリアされている時はセットできません。このビットは、ポートリセット処理の最後に PortResetStatusChange がセットされたとき、もしくは、HC が USBResume 状態にあるときにもクリアされます。もしアップストリームのレジューム処理が進行中なら、それが HC に伝わります。 0：ポートはサスペンド状態ではない。 1：ポートはサスペンド状態である。 （write）SetPortSuspend このビットに 1 を書き込むことで、PortSuspendStatus ビットをセットできます。0 の書き込みは無効です。CurrentConnectStatus がクリアされている場合、このビットに 1 を書き込んでも PortSuspendStatus はセットされず、その代わりに、ConnectStatusChange がセットされます。これにより、切断されているポートにサスペンド処理が行われたことを HCD に知らせることができます。

Field	Root Hub Reset	Read/Write		Description
		HCD	HC	
POCI	0b	R/W	R/W	（read）PortOverCurrentIndicator このビットは、オーバーカレント状態の報告を Pre-Port モードに設定している時のみ有効です。オーバーカレント状態の Pre-Port 報告モードが無効の場合、このビットは 0 がセットされます。クリアされている場合は、このポートのパワー状態は正常です。このビットがセットされている場合は、オーバーカレント状態が存在しています。このビットは常にオーバーカレント入力信号の状態を示します。 0：オーバーカレント状態ではない。 1：オーバーカレント状態を検出。 （write）ClearSuspendStatus HCD はレジュームを開始するために 1 を書き込みます。0 の書き込みは無効です。レジュームは PortSuspendState がセットされているときのみ開始できます。
PRS	0b	R/W	R/W	（read）PortResetStatus このビットは SetPortReset に書き込みことでセットされ、ポートリセット信号がアサートされます。リセットが完了すると、PortResetStatusChange がセットされている場合はこのビットがクリアされます。このビットは CurrentConnectStatus がクリアされている場合はセットできません。 0：ポートリセット信号はアクティブではない。 1：ポートリセット信号がアクティブ。 （write）SetPortReset HCD はこのビットに 1 を書き込むことで、ポートリセット信号を発行できます。0 の書き込みは無効です。CurrentConnectStatus がクリアされている場合、このビットへの書き込みによって PortResetStatus はセットされません。しかし、その代わりに ConnectStatusChange がセットされます。これにより、切断されているポートにリセット処理が行われたことを HCD に知らせることができます。

Field	Root Hub Reset	Read/Write		Description
		HCD	HC	
PPS	0b	R/W	R/W	（read）PortPowerStatus このビットは、パワースイッチモードの設定によらず、ポートのパワー状態を示します。このビットは、オーバーカレント検出時にクリアされます。HCD は SetPortPower ビットもしくは SetGlobalPower ビットに書き込むことで、このビットをセットできます。同様に、このビットのクリアは ClearPortPower もしくは ClearGlobalPower に書き込むことで行えます。どちらのパワースイッチモードが有効かは、PowerSwitchMode ビット、PortPowerControlMask[NDP] によって決めることができます。グローバルパワーモード（PowerSwitchingMode＝0）の場合、Set/ClearGlobalPower でこのビットをコントロールできます。Per-port パワーモード（PowerSwitchingMode＝1）の場合、このビットに対応する PortPowerControlMask[NDP]ビットがセットされているなら Set/ClearPortPower が有効です。このビットに対応する対応する PortPowerControlMask[NDP]ビットがセットされていないなら、Set/ClearGlobalPower が有効になります。ポートパワーが無効になっている時は、CurrentConnectStatus、PortEnableStatus、PortSuspendStatus、PortResetStatus がリセットされます。 0：ポートパワーはオフ。 1：ポートパワーはオン。 （write）SetPortPower このビットに 1 を書き込むことで、PortPowerStatus ビットをセットできます。0 の書き込みは無効です。 【注】パワースイッチをサポートしていない場合、常に 1b'が読み出されます。
LSDA	Xb	R/W	R/W	（read）LowSpeedDeviceAttached このビットは、接続されているデバイスのスピードを示します。セットされている場合、LowSpeed デバイスが接続されていることを示します。クリアされている場合、FullSpeed デバイスが接続されていることを示します。このビットは CurrentConnectState がセットされているときのみ有効です。 0：full speed デバイスが接続されている。 1：low speed デバイスが接続されている。 （write）ClearPortPower このビットに 1 を書き込むことで、PortPowerStatus ビットをクリアできます。0 の書き込みは無効です。

Field	Root Hub Reset	Read/Write		Description
		HCD	HC	
CSC	0b	R/W	R/W	ConnectStatusChange このビットは、接続・切断イベントは起こった際にセットされます。HCD はクリアのために 1 を書き込みます。0 の書き込みは無効です。 SetPortReset、SetPortEnable、SetPortSuspend への書き込みが起こったときに、CurrentConnectStatus がクリアされた場合は、このビットは１にセットされ続けます。これは、ポート切断時の書き込みが起こらなくなるために、HCD に接続の状態を再評価する必要性を示すために、1 をセットし続けます。 0：CurrentConnectStatus は変化していない。 1：CurrentConnectStatus が変化した。 【注】DeviceRemovable[NDP]がセットされている場合は、デバイスが接続されたことをシステムに知らせるために、RootHub リセット後のみセットされます。
PESC	0b	R/W	R/W	PortEnableStatusChange このビットは、ハードウェアにより PortEnableStatus ビットがクリアされた際にセットされます。HCD によって PortEnableStatus ビットが変更されてもこのビットは変化しません。HCD はクリアのために 1 を書き込みます。0 の書き込みは無効です。 0：PortEnableStatus は変化していない。 1：PortEnableStatus が変化。
PSSC	0b	R/W	R/W	PortSuspendStatusChange このビットは、レジューム処理がすべて完了した際にセットされます。その処理には 20msec のレジュームパルス、LS EOP、3msec の resychronization 遅延が含まれます。HCD はクリアのために 1 を書き込みます。0 の書き込みは無効です。このビットは ResetStatusChange がセットされるときもクリアされます。 0：レジュームは完了していない。 1：レジュームが完了。
OCIC	0b	R/W	R/W	PortOverCurrentIndicatorChange このビットは、オーバーカレント状態を Per-Port で報告する設定のときのみ有効です。このビットは PortOverCurrentIndicator ビットが変化した際にセットされます。HCD はクリアのために 1 を書き込みます。0 の書き込みは無効です。 0：PortOverCurrentIndicator に変化はない。 1：PortOverCurrentIndicator が変化した。
PRSC	0b	R/W	R/W	PortResetStatusChange このビットはポートリセット信号が 10ms 間発行された後にセットされます。HCD はクリアのために 1 を書き込みます。0 の書き込みは無効です。 0：ポートリセットは完了していない。 1：ポートリセットは完了した。

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（型名MS7727SE01、以下SH7727SE）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）
- 制御用PC（Windows® 98/2000）
- シリアルクロスケーブル
- USBケーブル
- High-Performance Debugging Interface（以下HDI）ルネサス テクノロジ製
- High-Performance Embedded Workshop（以下HEW）ルネサス テクノロジ製
- USBファンクションデバイス（任意）

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。

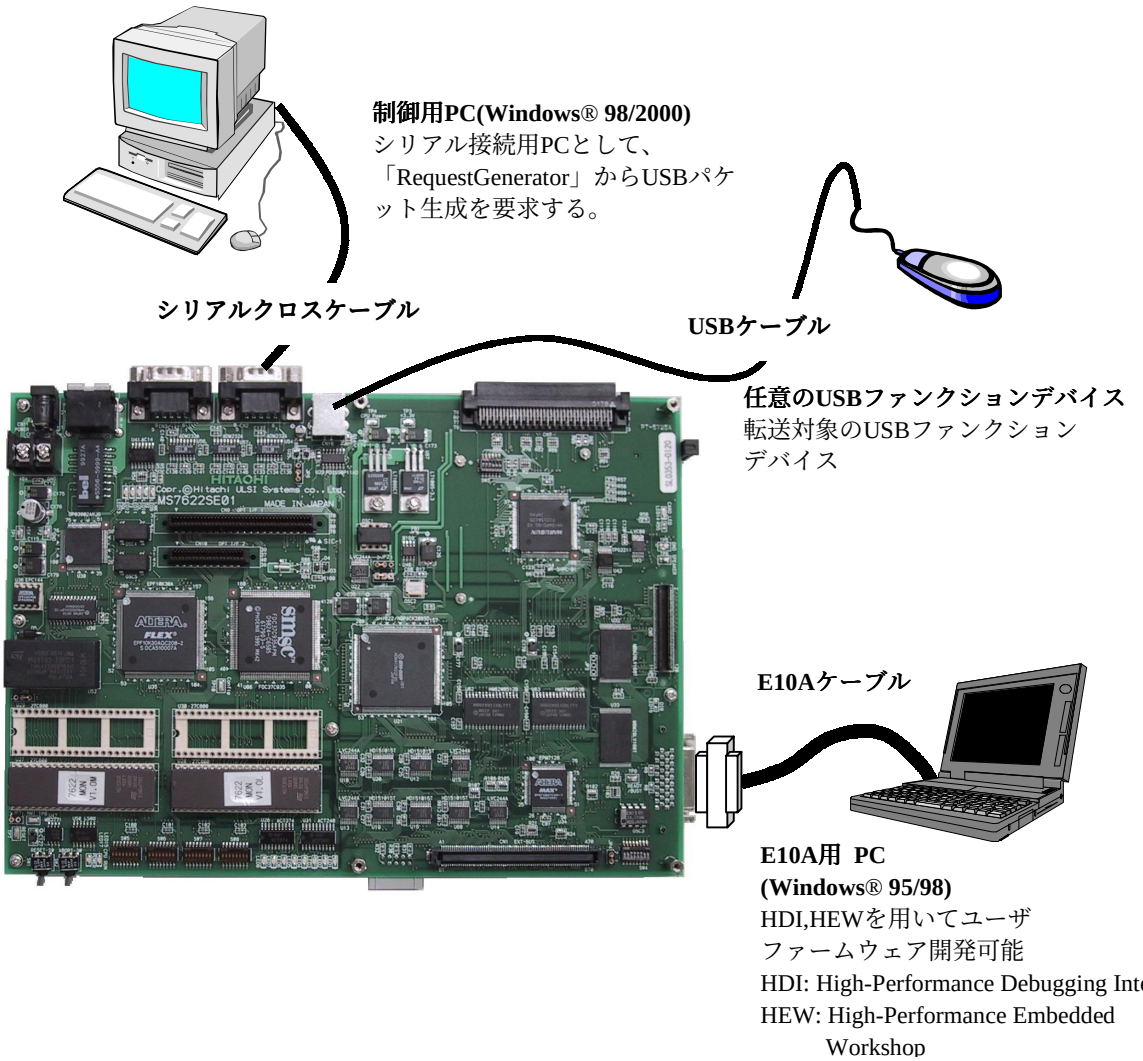

図 3.1 **デバイスの接続形態**

(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

(2) 制御用 PC

シリアル端子搭載の Windows® 98/2000 を USB パケット生成ツール「RequestGenerator」用 PC として使用します。

(3) PCMCIA スロット搭載の PC（E10A 用 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。

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、およびリンカについて説明します。

3.2.1 サンプルプログラム

サンプルプログラムとして必要なファイルは、すべて 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。

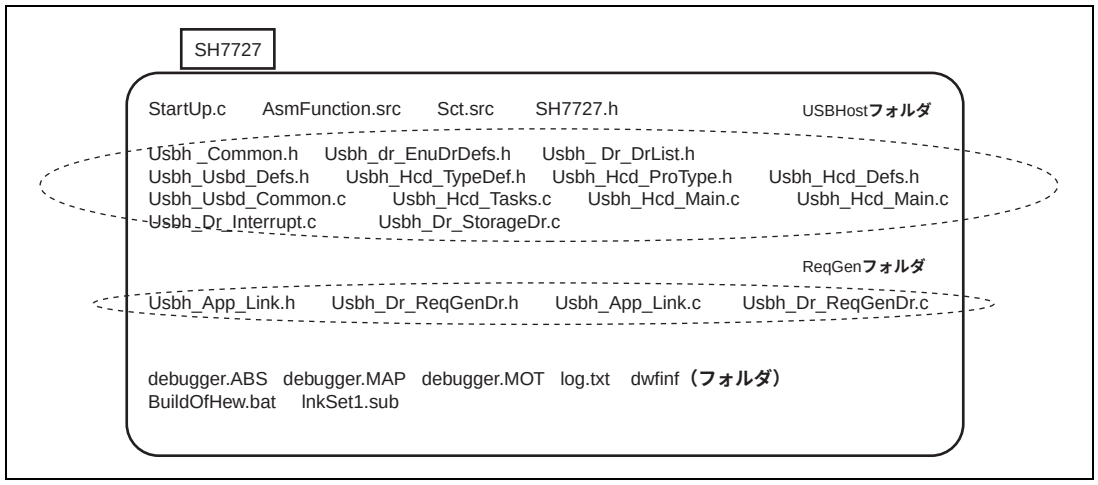

図 3.2 SH7277 **フォルダに含まれるファイル**

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）。

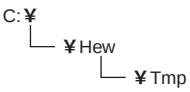

図 3.3 **作業フォルダの作成**

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）。

【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。**

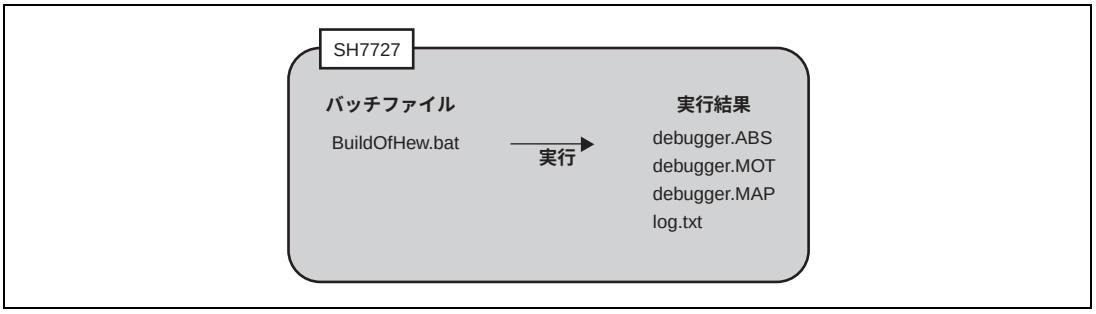

図 3.4 **コンパイル結果**

3.2.3 RequestGenerator

PC 上の USB パケット生成ツール『RequestGenerator』に関するファイルは全て「ReqGen」フォルダに収められています。

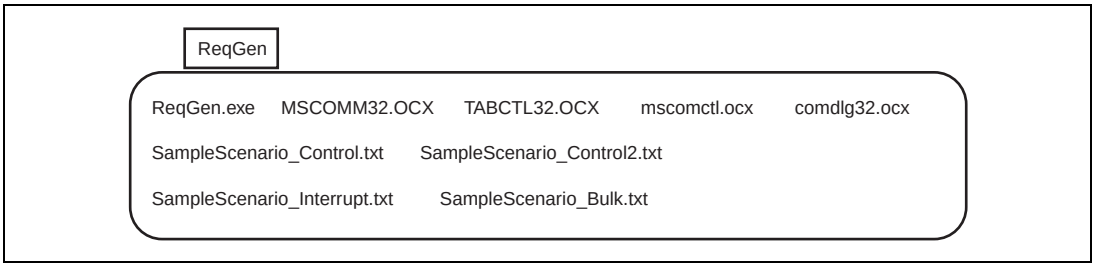

図 3.5 ReqGen **フォルダに含まれるファイル**

3.3 プログラムのロードと実行方法

図 3.6 にサンプルプログラムのメモリマップを示します。

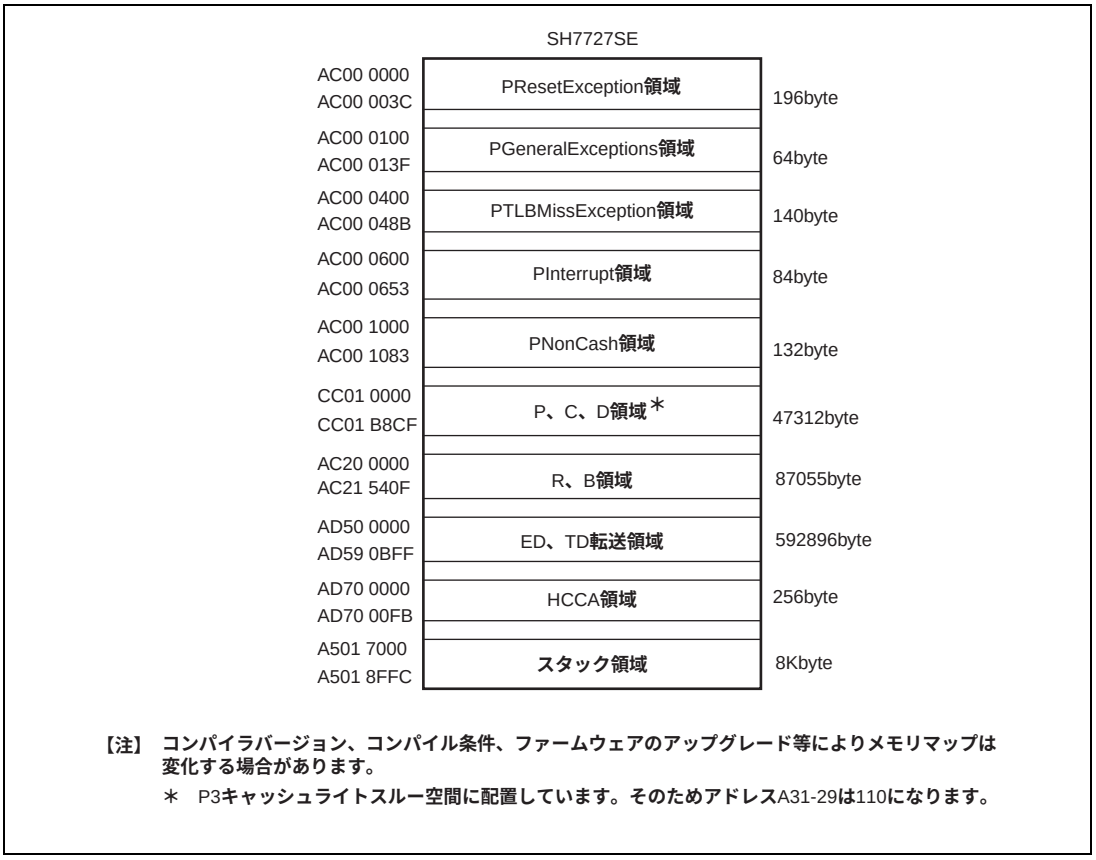

図 3.6 **メモリマップ**

図 3.6 のように、本サンプルプログラムは P、C、D、R、B すべての領域を SDRAM 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、このようにプログラムを 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と実行

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。
- シリアル接続PCのCOM1ポートとSH7727SEをシリアルケーブルで接続してください。
- E10A用PC、シリアル接続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**3.7参照）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」ボタンをクリック又は、<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**3.8参照）、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。
- 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。

以上の操作で、サンプルプログラムを SH7727SE の SDRAM 上にロードし、実行する事ができます。

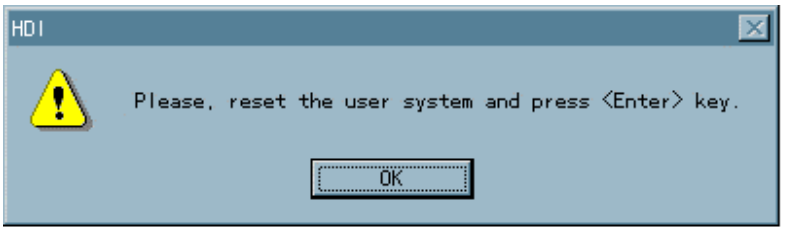

図 3.7　リセット要求ダイアログ

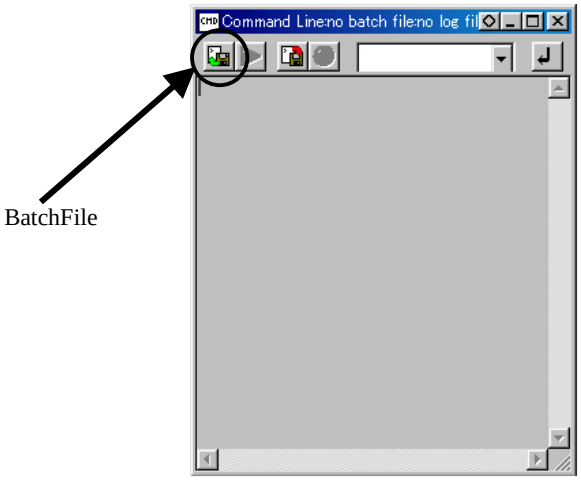

図 3.8　コマンドライン入力

3.4 実行方法

- RequestGeneratorの起動方法

1. 「3.3.1 **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SH7727SE上の8ビットLEDが0xAAと表示されます。
2. 「ReqGen」フォルダ内の.ocxファイルをWindows¥System32フォルダ（Windows® 98の場合）、WinNT¥System32（Windows® 2000の場合）にコピーします。
3. 「ReqGen」フォルダ内のReqGen.exeを実行します。これによりCOM1ポートを開きます。

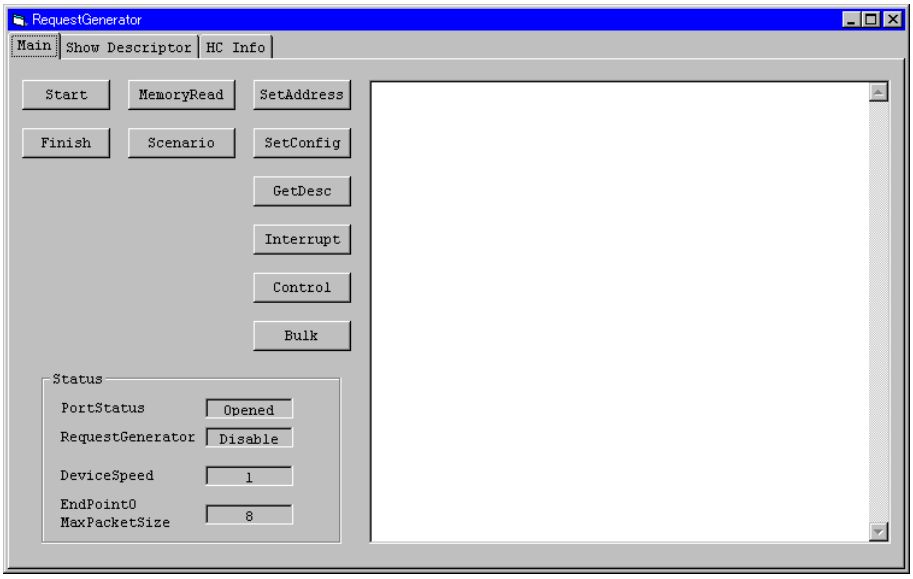

図 3.9 RequestGenerator **初期画面**

4. “Start”ボタンを押します。

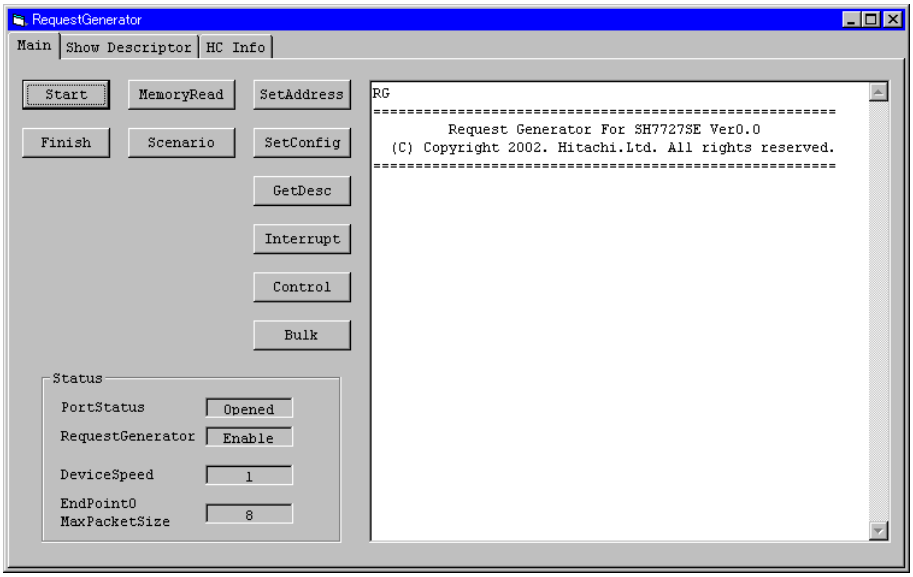

図 3.10　RequestGenerator **起動直後**

5. SH7727SEのUSB-Aコネクタに USBファンクションデバイスを接続します。接続を検出すると、GetDescriptor（Device）コマンドを実行し、Endpoint0のMaxPacketSize情報を取得します。また、接続されたデバイスのスピード情報も取得できます。

【注】＊ RequestGenerator **を使用する場合は、必ず、“** Start”**を押してからデバイスを接続してください。**

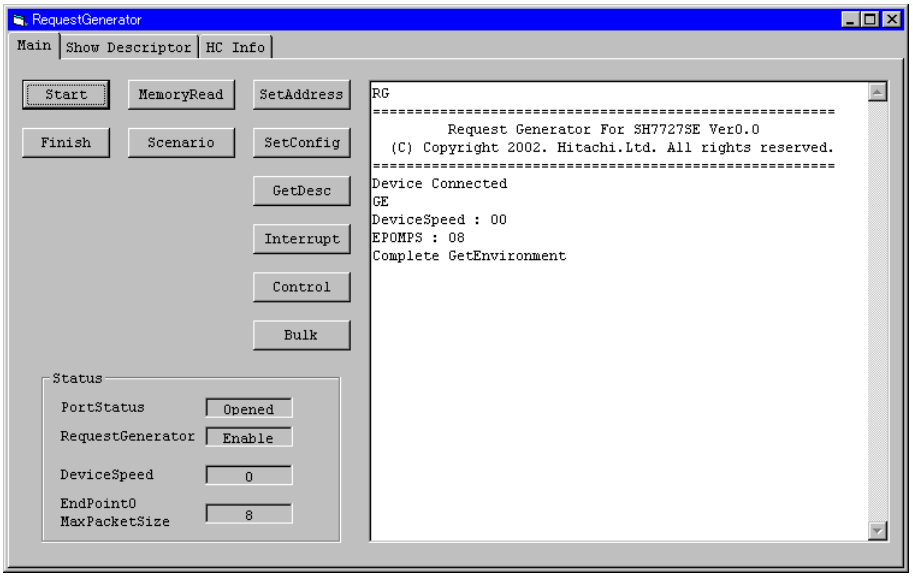

図 3.11　RequestGenerator**（デバイス接続直後）**

6. 所望の操作を行ってください。
7. SH7727SEのUSB-AコネクタからUSBファンクションデバイスを抜く。
8. “Finish”ボタンを押します。
9. 以後、4～8を繰り返すことで、様々のUSBファンクションデバイスを「RequestGenerator」でコントロールすることができます。

以下、RequestGenerator の各ボタンの機能について説明します。

- Start

  “Start”ボタンを押すことで、RequestGeneratorを起動することができます。起動すると、“ RequestGenerator”欄の表示が「Disable」から「Enable」に変わります。

- Finish

  “ Finish ”ボタンを押すことで、RequestGeneratorを終了することができます。終了すると、“ RequestGenerator”欄の表示が「Enable」から「Disable」に変わります。

- SetAddress

  “ SetAddress ”ボタンを押すことで、SetAddressコマンドを発行することができます。パラメータ入力画面で、割り当てるアドレスを入力（2以上の値）してください。

- SetConfig

  “ SetConfig ”ボタンを押すことで、SetConfiguratoinコマンドを発行することができます。パラメータ入力画面で、Configuration値を入力してください。

- GetDesc

  “ GetDesc ”ボタンを押すことで、GetDescriptorコマンドを発行することができます。パラメータ入力画面でDescriptorタイプ（Device、Configurationのみに対応）、Descriptorサイズを入力してください。

- Control

  “ Control ”ボタンを押すことで、任意のControl転送を行うことができます。パラメータ入力画面で、8ByteのDeviceRequest値、データステージのサイズ、データステージの転送方向、データステージで転送するデータ（In転送時は設定不要）を指定します。SetAddressやSetConfigurationなどDataStageの無いものはデータステージの転送方向はOut方向を指定してください。

【注】 1. **この機能で** SetAddress **コマンドは実行できません。**SetAddress **は“** SetAddress **”ボタンで実行してください。**

2. **誤った設定を入力すると、誤動作する恐れがあります。**

- Interrupt

  “ Interrupt ”ボタンを押すことで、Interrupt転送を実行できます。パラメータ入力画面で、転送対象のEndpoint番号、転送方向、MaxPacketSize、PollingRate、転送サイズ、転送データ（In転送時は設定不要）、生成するInterruptパケットの個数を指定します。

- Bulk

  “ Bulk ”ボタンを押すことで、Bulk転送を実行できます。パラメータ入力画面で、Endpoint番号、MaxPacketSize、転送方向、転送サイズ、転送データ（In転送時は設定不要）を指定します。

- MemoryRead

  “ MemoryRead ”ボタンを押すことで、SH7727SEに対して、指定したアドレスからデータリードを行うことができます。パラメータ入力画面で、アクセスするアドレス、リードするサイズ、アクセスサイズ（Byteアクセス、Wordアクセス、LongWordアクセス）を指定します。

- Scenario

  “ Scenario ”ボタンを押すことで、実行させたい処理を書いたテキストファイルを読み込み、そのファイルに従って処理を実行させることができます。このシナリオ機能では、“ Start ”、“ Finish ”、“ Scenario ”以外のボタンの機能をあらかじめテキストファイルに用意しておくことができます。

  表3.2に示す、各ボタンに対応するコマンド名とそのパラメータを示します。コマンド名とパラメータをスペースで区切り、また各パラメータもスペースで区切り、最後に改行を入力します。たとえば、SetConfigを実行したい時は、

  “ SC 1 ”と入力し改行します。**表**3.3に記述例を示します。

  パラメータは必ず**表**3.2の記載順に書いてください。各パラメータは、各ボタンを押した際に現れるパラメータ入力画面で指定するパラメータと同一です。パラメータの並びも同一になります。

表 3.2　**シナリオ機能でのコマンド名**

ボタン名	**シナリオ機能でのコマンド名**	**パラメータ**（D：Dec、H：Hex）
SetAddress	SA	**アドレス**（D、2 **以上の値**）
SetConfig	SC	**コンフィグレーション値**（D）
GetDesc	DeviceDescriptor GDD ConfigDescriptor GDC	**転送サイズ**（H）
Interrupt	INT	EndPoint **番号**（D）、**転送サイズ**（H）、**ポーリングレート**（D）、**転送方向**（D、1：OUT、2：IN）、**転送データ**（H）、MaxPacketSize（H）、**転送回数**（H）
Control	CNT	DeviceRequest（H）、**転送サイズ**（H）、**転送方向**（D、1：OUT、2：IN）、**転送データ**（H）
Bulk	BLK	**転送サイズ**（H）、**転送方向**（D、1：OUT、2：IN）、**転送データ**（H）、EndPoint **番号**（D）、MaxPacketSize（H）
MemoryRead	MR	**アクセスアドレス**（H）、**転送サイズ**（D）、**アクセス単位**（D、**1**：Byte、**2**：Word、**3**：LongWord）

表 3.3　シナリオファイルの記述例

記述例	転送内容
SA 2（改行）	アドレス＝2 として SetAddress コマンドを発行
SC 1（改行）	コンフィグレーション＝1 として SetConfig コマンド発行
GDD 12（改行）	転送サイズ＝0x12 として、GetDescriptor（Device）コマンド発行
GDC 400（改行）	転送サイズ＝0x400 として、GetDescriptor（Config）コマンド発行
CNT 0009010000000000 0 1（改行）	コンフィグレーション＝1 として SetConfig コマンド発行
CNT 8006000100001200 12 2（改行）	転送サイズ＝0x12 として、GetDescriptor（Device）コマンド発行
CNT 8006000200000004 0400 2（改行）	転送サイズ＝0x400 として、GetDescriptor（Config）コマンド発行
INT 1 4 10 2 0 4 20（改行）	以下の設定で Interrupt 転送。 Endpoint＝1、転送サイズ＝0x4Byte、ポーリングレート＝10msec、転送方向＝2（IN）、転送データ＝0、MaxPacketSize＝0x4Byte、転送回数＝0x20 回
BLK 40 1（転送データ）1 40（改行）	以下の設定で Bulk 転送。 転送サイズ＝0x40Byte、転送方向＝1（OUT）、 転送データ＝（省略）、Endpoint＝1、 MaxPacketSize=0x40Byte
BLK 40 2 0 2 40（改行）	以下の設定で Bulk 転送。 転送サイズ＝0x40Byte、転送方向＝2（IN）、転送データ＝0、Endpoint＝2、 MaxPacketSize＝0x40Byte

【注】 1. 必ず各行の先頭からコマンドを記述してください。

2. “//”以降はコメントとして無視します。

3. タブコードは無視します。

4. 必ず改行してください。改行コードがない場合、その行のコマンドは処理されません。

5. CNT コマンドで SetAddress は行えません。必ず SA コマンドで実行してください。

次に、その他の機能および注意事項を示します。

- Descriptor表示機能

  “Show Descriptor”タブを開きます。「Descriptor」をクリックすることで、“GetDesc”ボタンを押すことで取得したDescriptor情報を閲覧することができます。

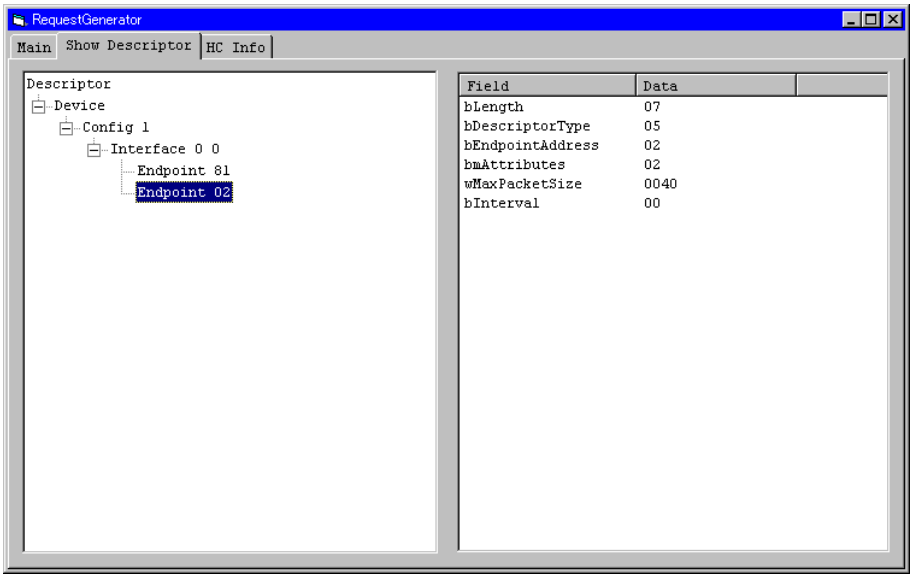

図 3.12　RequestGenerator（Descriptor **表示**）

Descriptor情報の表示は、“ GetDesc ”ボタンの実行によりDescriptorを取得した場合にのみ行えます。“ Control ”ボタン、“ Scenario ”ボタンにより取得したDescriptor情報は表示できません。

- レジスタ閲覧機能

  “ HC Info ”タブを開きます。「HC_Regs」をクリックする度に、その時点のUSBホストモジュールのレジスタ値を閲覧することができます。また「HC_Regs」をダブルクリックすることで、HcControlHeadED、HcControlCurrentED、

  HcBulkHeadED、HcBulkCurrentEDに示されているEDの各フィールド値を閲覧することできます。

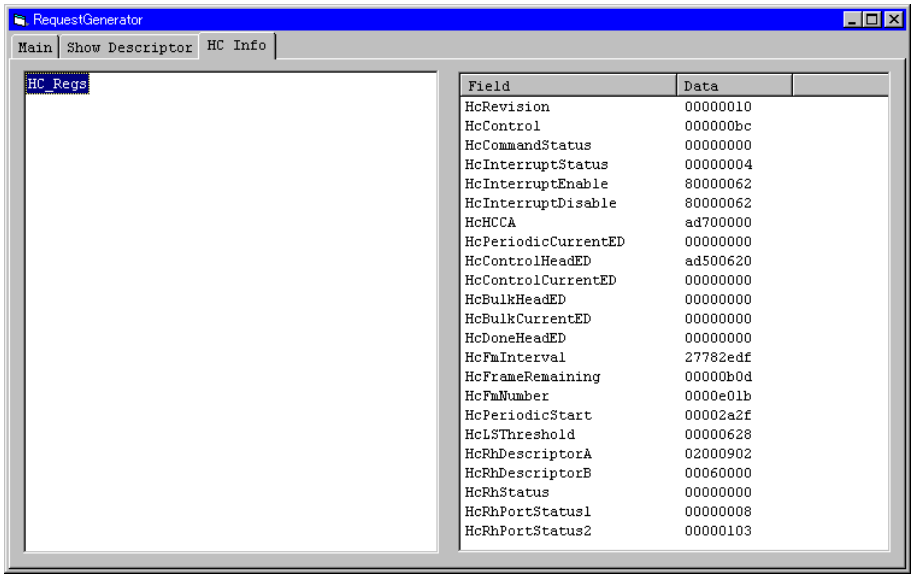

図 3.1-1　RequestGenerator（**レジスタ表示**）

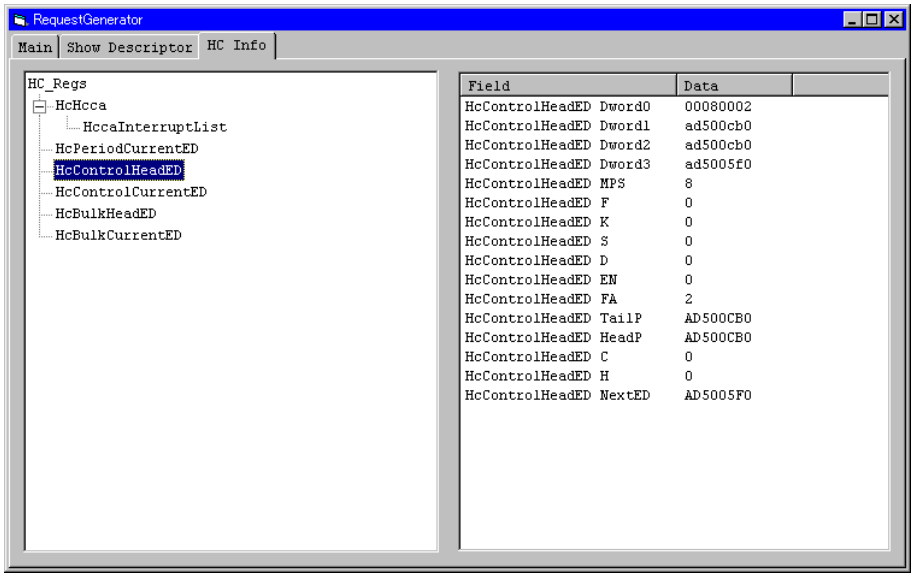

図 3.13-2　RequestGenerator（**レジスタ表示**）

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 ホストモジュールの割り込みもしくはメインルーチンからの分岐によって、USB ホストモジュールとしての処理を行います。また、PC 上の USB パケット生成ツール「RequestGenerator」との通信は SCIF モジュールを使って行います。SH7727 内蔵モジュールの割り込みのうち、本サンプルプログラムが使用する割り込みは、USB ホストモジュールに関する割り込みが RootHubStatusChange、WritebackDoneHead、FrameNumberOverflow の 3 種、SCIF に関する割り込みが、ERI2（受信エラー）、BRI2（ブレーク受信）、RXI2（受信データ FIFO フル）の 3 種です。

本サンプルプログラムの特長を以下に示します。

- コントロール転送を行うことが出来ます。
- バルク転送を行うことが出来ます。
- インタラプト転送を行うことが出来ます。
- シリアル接続PC上のパケット生成ツール「RequestGenerator」を使い、PC上から転送要求を生成することができます。

【注】 1. Isochronous **転送には対応していません。**

2. Suspend/Resume **には対応していません。**

4.1 状態遷移図

図 4.1 に本サンプルプログラムの状態遷移図を示します。本サンプルプログラムは、**図** 4.1 のように 6 つの状態に遷移します。

- 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。

- 接続待ち状態

  リセット状態で初期設定処理が終了するとこの状態になります。また、定常状態中にデバイスがRootHubから切断された場合も、この状態になります。この状態では、RootHubStatusChange割り込みを待ちます。RootHubにデバイスを接続すると、RootHubStatusChange割り込みが発生し、接続処理を行い、定常状態に移行します。

- 定常状態

  接続待ち状態で、RootHubにデバイスが接続された場合、この状態に移行します。この状態では、ED・TDを生成し、HCに転送要求を生成します。そして、その要求した転送の処理が終了した際の処理を行います。また、SCIFモジュールを制御してシリアル出力を行います。

- RootHub処理状態

  接続待ち状態もしくは定常状態から、RootHubStatusChange割り込みが発生した際に、この状態に移行します。この割り込みはRootHubの状態に変化が発生した場合に起こるので、その変化の原因を判定し、切断状態から接続状態に変化した際は接続処理を、接続状態から切断状態に変化した際は切断処理を、OverCurrent状態になった際はOverCurrent解除処理を行います。

- DoneQueue処理

  定常状態から、WriteBackDoneHead割り込みが発生した際に、この状態へ移行します。この状態は、HCから処理済みTD群であるDoneQueueを受け取ります。

- シリアル通信状態

  定常状態で、シリアル受信割り込みが発生した際に、この状態に移行します。この状態は、PC上のツール『RequestGeneraor』と通信する状態です。

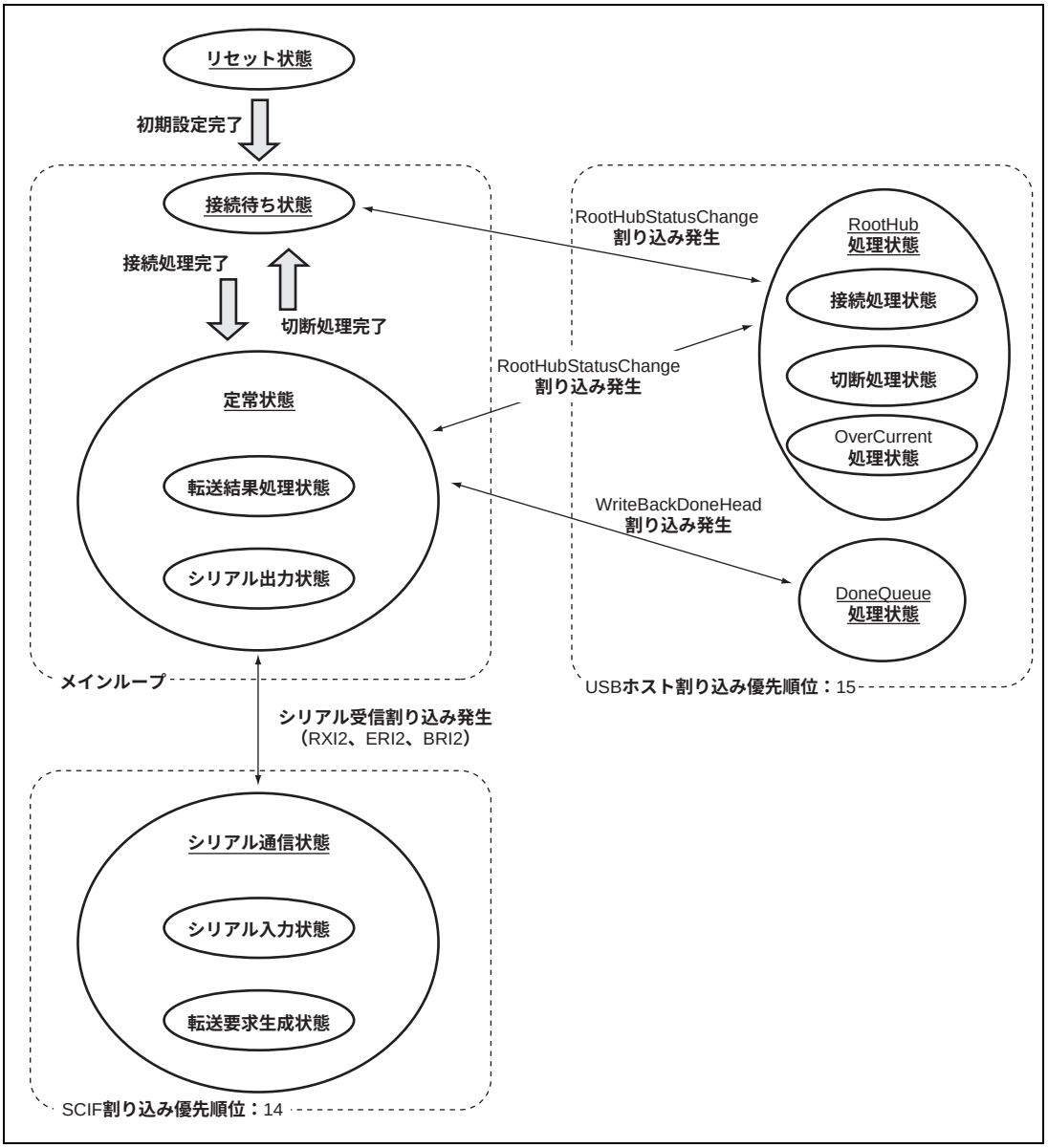
リセット状態
初期設定完了
接続待ち状態
接続処理完了
切断処理完了
定常状態
転送結果処理状態
シリアル出力状態
メインループ
RootHubStatusChange
割り込み発生
RootHub
処理状態
接続処理状態
切断処理状態
OverCurrent
処理状態
RootHubStatusChange
割り込み発生
WriteBackDoneHead
割り込み発生
DoneQueue
処理状態
USBホスト割り込み優先順位：15
シリアル受信割り込み発生
（RXI2、ERI2、BRI2）
シリアル通信状態
シリアル入力状態
転送要求生成状態
SCIF割り込み優先順位：14

図 4.1　**状態遷移図**

4.2 割り込みの種類

4 章冒頭で説明したように、本サンプルプログラムでは、USB ホストモジュールおよび SCIF の割り込みを使用します。使用する割り込み要因は、USB ホストが HcInterruptStatus レジスタ、SCIF はシリアルステータスレジスタ SCSSR によって示され、それぞれ計 3 種類です。割り込みが発生すると、その要因に対応するフラグビットに 1 がセットされ、CPU に対して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、それぞれ発生した割り込みに対応する処理を行います。**図** 4.2 に割り込みの種類を纏めます。

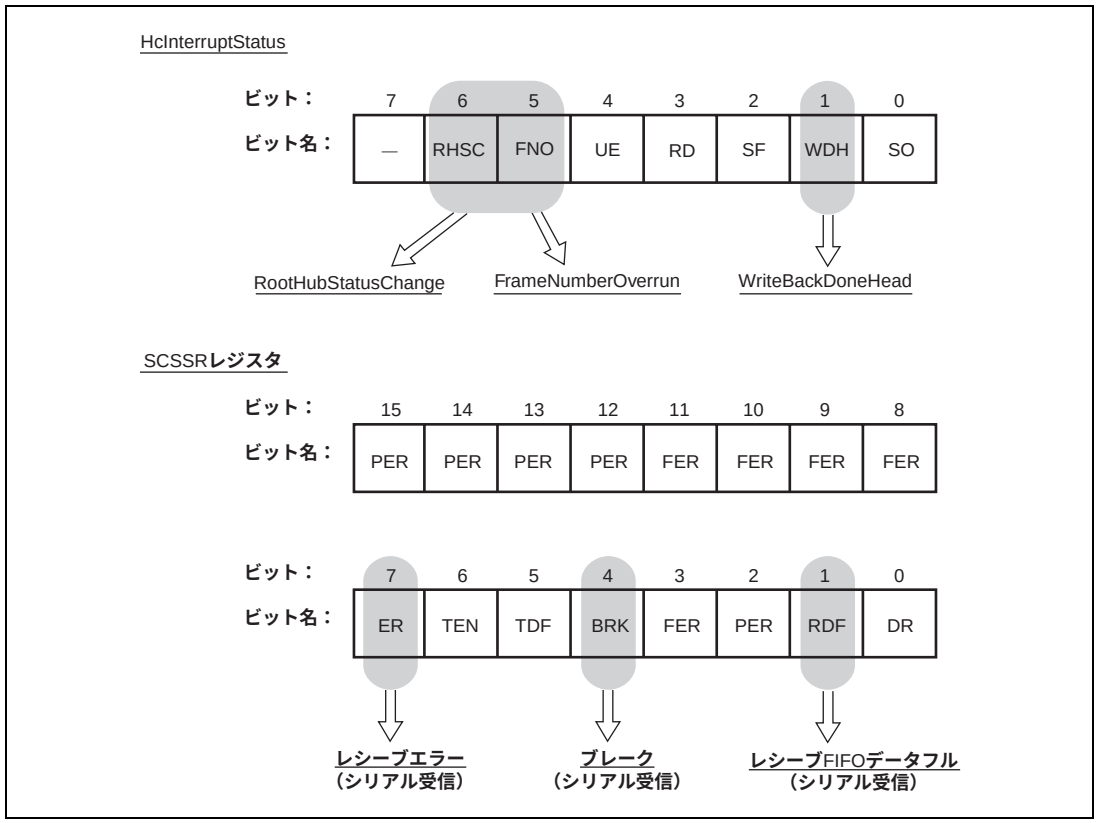

図 4.2 **割り込みフラグの種類**

4.3 ファイル構成

本サンプルプログラムは、9 個のソースファイルと、10 個のヘッダ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。また**図** 4.3 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**
Usbh_Hcd_Tasks.c	OHCI **仕様書の関数群**
Usbh_Hcd_Others.c	HCD **層の共通関数**
Usbh_Hcd_Main.c	HCD **層の** MainRoutine,**割り込み関数など**
Usbh_Usbd_Common.c	USBD **層の共通関数**
Usbh_Dr_EnuDr.c	Enumeration **処理用の** Driver **層関数**
Usbh_Dr_ReqGenDr.c	RequestGenerator **用の** Driver **層関数**
Usbh_App_Log.c	**シリアル出力関数**
Usbh_App_Link.c	USB **ホストモジュールと** SCIF **モジュールの連動処理関数**
SH7727.h	SH7727 **レジスタ定義**
Usbh_Hcd_TypeDef.h	HCD **層の構造体宣言**
Usbh_Hcd_ProType.h	HCD **層の** Prototype **宣言**
Usbh_Hcd_Defs.h	HCD **層の各種宣言**
Usbh_Usbd_Defs.h	USBD **層の各種宣言**
Usbh_Dr_EnuDrDefs.h	EnumerationDriver **の各種宣言**
Usbh_Dr_DrList.h	EnumerationDr **が呼び出す** DeviceDriver **のリスト**
Usbh_Dr_ReqGenDr.h	ReqGenDr **の各種宣言**
Usbh_App_Link.h	USB **ホストモジュールと** SCIF **モジュールの連動処理に必要な各種宣言**
Usbh_Common.h	USBHost **全体の共通宣言**

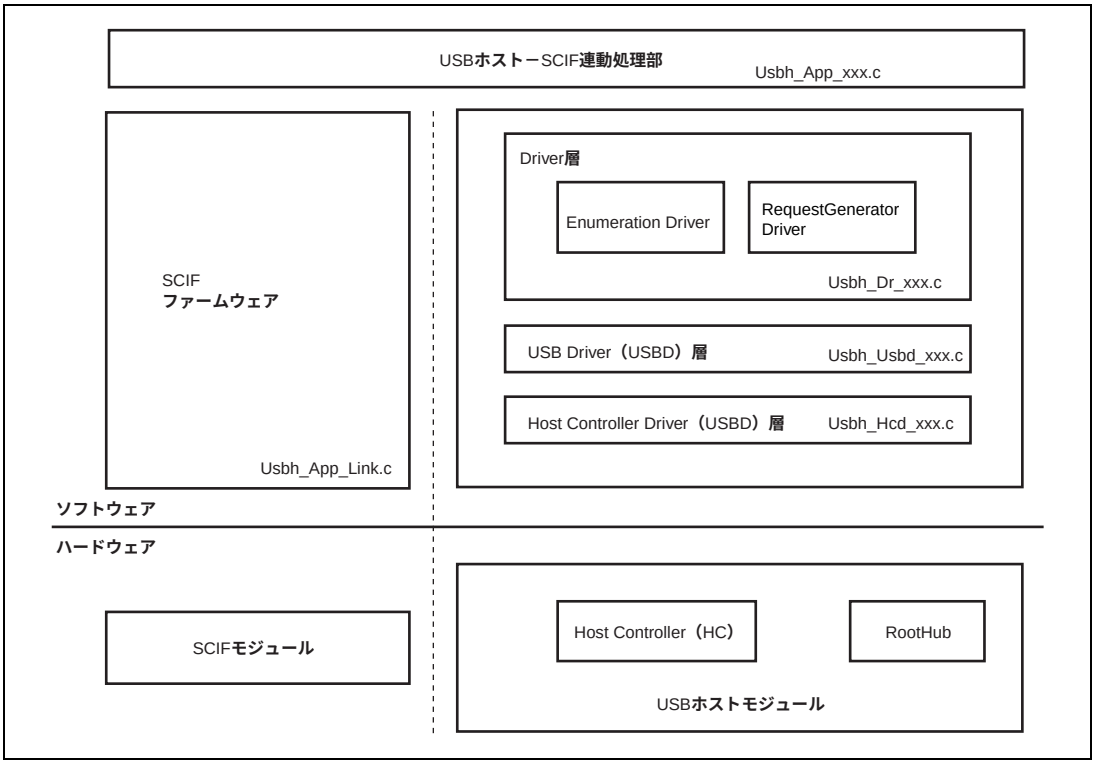

図 4.3　**ファームウェアの階層構造**

4.4 関数の機能

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	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	SCIF モジュールの初期化

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、転送に使用する RAM 領域のクリアを行います。

表 4.2-2 Usbh_Hcd_Main.c

格納ファイル	関数名	機　　能
Usbh_Hcd_Main.c	HCD_Setup	HCD を初期化する
	HCD_IntRoutine	割り込み処理ルーチン
	HCD_MainRoutine	HCD 層のメインルーチン
	HCD_ControlRootHub	RootHub を制御

Usbh_Hcd_Main.c では、HCD の初期化、各種割り込み発生時の処理を行います。また RootHubStatusChange 割り込み発生時には、RootHub の制御を行います。

表 4.2-3　Usbh_Hcd_Tasks.c

格納ファイル	関数名	機　　能
Usbh_Hcd_Tasks.c	InsertEDForEndpoint	ED を生成し、List に追加する
	QueueGeneralRequest	TD を生成し、ED にリンクさせる
	ProcessDoneQueue	Done Queue の処理を行う
	InitailizeInterruptLists	Interrupt List を構築する
	OpenPipe	Periodic ED を生成、List に追加する
	CheckBandwidth	Bandwidth をチェックする
	PauseED	ED を一旦停止させる
	ProcessPausedED	ED を一旦停止させる
	RemoveED	既存の ED を削除する
	CancelRequest	既存の USBDRequest に関する処理をキャンセルする。生成済みの TD を削除する
	UnscheduleIsochronousOrInterruptEndpoint	Periodic ED を削除する
	SetFrameInterval	Frame Interval の値を調整する
	Get32BitFrameNumber	16Bit 長の HccaFrameNumber から 32Bit の Frame Number を作成する

Usbh_Hcd_Tasks.c は、すべて OHCI 仕様書**「第 5 章　サンプルプログラムの動作」**掲載のサンプルコードがベースになっています。ED、TD の生成、削除に関する処理を行います。

表 4.2-4 Usbh_Hcd_Others.c

格納ファイル	関数名	機　　能
Usbh_Hcd_Others.c	HCD_CreateDeviceData	DeviceData 構造体変数を初期化する
	HCD_CreateEndPoint	Endpoint 構造体変数を初期化する
	HCD_SetupEndpoint	Endpoint 構造体変数をセットアップする
	AllocateEndpointDescriptor	ED 変数を確保する
	PhysicalAddressOf	アドレスを取得する
	IsListEmpty	List_Entry 変数が Null かどうかを判定
	AllocateTransferDescriptor	TD 変数を確保
	InsertHeadList	List_Entry 変数を List の先頭に挿入する
	InsertTailList	List_Entry 変数を List の最後に挿入する
	Containing_Record	条件に合う ED,TD を探す
	InitializeListHead	List_Entry 変数を初期化する
	min	最小値を判定
	CompleteUsbdRequest	処理完了した USBDRequest 変数を USBD 層に渡す
	FreeTransferDescriptor	TD 変数を削除する
	RemoveListHead	LIST_ENTRY 変数を削除
	RemoveListEntry	LIST_ENTRY 変数を削除
	VirtualAddressOf	アドレスを取得
	HCD_InitializeEndpoint	Endpoint 変数を生成し、ED 変数を作成する
	HCD_GetRootDeviceSpeed	RootHub に接続されたデバイスのスピード情報を取得する
	HCD_ScanEndpoint	Endpoint 管理用配列にアクセスする
	HCD_StoreEndpointInfo	Endpoint 管理用配列にアクセスする
	HCD_Request	受け取った USBDRequest から、Endpoint 変数、ED 変数、TD 変数を生成する
	HCD_CheckDiffEPs	Endpoint の設定が変化したか確認する
	HCD_CheckRemainedTDs	残りの TD 変数の個数を調べる
	HCD_CheckRemainedEDs	残りの ED 変数の個数を調べる
	HCD_CheckRemainedEPs	残りの Endpoint 変数の個数を調べる
	HCD_PauseEndpoint	ED の中断処理を行う
	HCD_RemoveEndpoint	Endpoint 変数、ED 変数を削除する
	HCD_CancelRequest	転送要求済みの USBDRequest を削除する
	HCD_FreeEndpoint	Endpoint 変数をクリア
	FreeEndpointDescriptor	ED 変数をクリア
	HCD_ClearDeviceData	DeviceData 変数をクリア
	HCD_ClearList	Endpoint、ED、TD の各変数用配列を初期化する
	HCD_ClearHCCA	HCCA 領域を初期化する

格納ファイル	関数名	機　　能
Usbh_Hcd_Others.c	HCD_WaitConnectionComplete	RootHub へのデバイス接続処理を行う
	HCD_WaitRoutine	Wait ルーチン

Usbh_Hcd_Others.c は、HCD 層関数から呼ばれる共通関数や、USBD 層から呼ばれる関数を纏めてあります。

表 4.2-5　Usbh_Usbd_Common.c

格納ファイル	関数名	機　　能
Usbh_Usbd_Common.c	USBD_SetupDriverRequestTo USBDRequestndpoint	DriverRequest 構造体変数から USBDRequest 構造体変数を作成する
	USBD_ReceiveDriverRequest	Dr 層から DriverRequest 構造体変数を受け取る
	USBD_ReceiveUSBDRequest	HCD 層から処理完了した USBDRequest 構造体変数を受け取る
	USBD_CreateRequest	USBDRequest 構造体変数を確保し、初期化する
	USBD_FreeRequest	USBDRequest 構造体変数の初期化する
	USBD_GetDeviceAddress	使用可能な DeviceAddress の値を取得する
	USBD_ReportConnection	RootHub にデバイスが接続された際に呼ばれ、USBD 層の初期化を行う
	USBD_ReportDisConnection	RootHub に接続されているデバイスが切断された際に呼ばれ、切断処理を行う
	USBD_RemoveUSBDRequest	HCD に処理要求済みの USBDRequest を削除する
	USBD_RemoveDevice	指定した DeviceAddress に属する全てに Endpoint 構造体変数を削除する
	USBD_RemoveEndpoint	指定した Endpoint 構造体変数を削除する
	USBD_ReadDAArray	DeviceAddress 管理用配列にアクセスする
	USBD_WriteDAArray	DeviceAddress 管理用配列にアクセスする
	USBD_RootDeviceSpeedInfo	RootHub に接続された Device の Speed 情報を取得する

Usbh_Usbd_Common.c は、USBD 層の関数群であり、Driver 層から転送リクエストを受け取り、それを HCD 層に転送し、またその結果を HCD 層から受け取り、要求元の Driver に連絡します。

表 4.2-6　Usbh_Dr_EnuDr.c

格納ファイル	関数名	機　　能
Usbh_Dr_EnuDr.c	EnumeDriver_Start	EnumerationDr を初期化し、転送開始の準備を行う
	EnumeDriver_Result	USBD 層に転送リクエストを行った結果を受け取り、次の転送リクエストを行う
	EnumeDriver_Request	USBD 層に転送リクエストを行う
	EnumeDriver_GetInfoFromDescriptor	取得した各種 Descriptor 情報から、必要な情報を取得する
	EnumeDriver_GetDeviceDescriptor	取得した DeviceDescriptor から、必要な情報を取得する
	EnumeDriver_GetConfigDescriptor	取得した ConfigDescriptor から、必要な情報を取得する
	EnumeDriver_GetInterfaceDescriptor	取得した InterfaceDescriptor から、必要な情報を取得する
	EnumeDriver_GetEndpointDescriptor	取得した EndpointDescriptor から、必要な情報を取得する
	EnumeDriver_Initialize	EnumerationDriver を初期化する
	EnumeDriver_Clear	EnumerationDriver の内部フラグをクリアする
	EnumeDriver_EnableDriver	接続されたデバイスの Class を判定し、最適な Driver 関数を呼び出す
	EnumeDriver_Enable	EnumerationDriver を有効にする
	EnumeDriver_Disable	EnumerationDriver を無効にする
	DummyDriver_Start	ダミーの Driver 実行関数
	DriverCommon_SetupDriverRequest	転送要求を DriverRequest 構造体変数に纏める
	DriverCommon_DoDriverRequest	DriverRequest 構造体変数を USBD 層に渡し、転送リクエストを行う
	DriverCommon_Endian	Endian 変換を行う

Usbh_Dr_EnuDr.c は、EnumerationDriver に関する処理を行います。接続されたデバイスに対して、SetAddres を行い、DeviceClass を判定して、呼び出すべき Driver を判定します。

表 4.2-7　Usbh_App_Log.c

格納ファイル	関数名	機　　能
Usbh_App_Log.c	App_LogOut_Char	文字列を出力する関数
	App_LogOut_Data	文字列を出力する関数
	App_ExOutPut	１Byte の文字変数を出力する関数

Usbh_App_Log.c では、デバッグ情報をシリアル出力します。

表 4.2-8 Usbh_Dr_ReqGenDr.c

格納ファイル	関数名	機　　能
Usbh_Dr_ReqGenDr.c	App_ReceiveCommand	RequestGenerator からコマンドデータを受け取る
	App_CheckReceiveCommand	受信したコマンドデータを判別する
	DisableRequestGenerator	RequestGenerator の終了処理を行う
	DoRequestGenerator	RequestGeneratorDriver の初期化処理を行う
	DoSetAddress	SetAddress 要求を行う
	ReceiveSetAddressResult	SetAddress 要求の結果を受け取る
	DoGetDescriptorDevice	GetDescriptor（Device）要求を行う
	ReceiveGetDescriptorDevice Result	GetDescriptor（Device）要求の結果を受け取る
	DoGetDescriptorConfig	GetDescriptor（Config）要求を行う
	ReceiveGetDescriptorConfig Result	GetDescriptor（Config）要求の結果を受け取る
	DoSetConfiguration	SetConfiguration 要求を行う
	ReceiveSetConfigurationResult	SetConfiguration 要求の結果を受け取る
	DoIntTransfer	Interrupt 転送要求を行う
	ReceiveIntTransferResult	Interrupt 転送要求の結果を受け取る
	DoSetEnvironment	DeviceSpeed 情報、EP0 の MaxPacketSize 情報をセットする
	DoGetEnvironment	DeviceSpeed 情報、EP0 の MaxPacketSize 情報の取得要求を行う
	ReceiveGetEnvironmentResult	DeviceSpeed 情報、EP0 の MaxPacketSize 情報を受け取る
	DoControlTransfer	任意の Control 転送要求を行う
	ReceiveControlTransferResult	Control 転送要求の結果を受け取る
	DoBulkTransfer	Bulk 転送要求を行う
	ReceiveBulkTransferResult	Bulk 転送要求の結果を受け取る
	DoMemoryRead	指定したアドレスからデータリードを行う
	mystrtoul	文字列を 16 進数に変換
	DisplayTitle	RequestGenerator のタイトルを出力する
	DisplayHelp	Help 情報を表示する

Usbh_Dr_ReqGenDr.c では、PC 上の USB パケット生成ツール「RequestGenerator」からの転送要求を受け取り、その結果を返します。

表 4.2-9 Usbh_App_Link.c

格納ファイル	関数名	機　　能
Usbh_App_Link.c	App_SerialIn	SerialIn 処理を行う
	App_SerialOut	SerialOut 処理を行う

Usbh_App_Link.c は、SCIF モジュールをコントロールし、RequestGeneratorDriver と PC 上のツール「RequestGenerator」の通信を行います。

図 4.4 に**表** 4.2 で説明した主な関数の相関関係を示します。上位側の関数が、下位側の関数を呼び出しています。また、複数の関数が同一の関数を呼び出す事もあります。定常状態では、CallResetException が他の関数を呼び出します。USB ホストの割り込みの発生によって遷移する USB ホスト通信状態では、割り込み関数である CallInterrupt が HCD_IntRoutine を呼び出します。さらに、SCIF 割り込みでは、App_SerialIn 関数を呼び出します。**図** 4.4 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。呼び出される順序については、**「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。

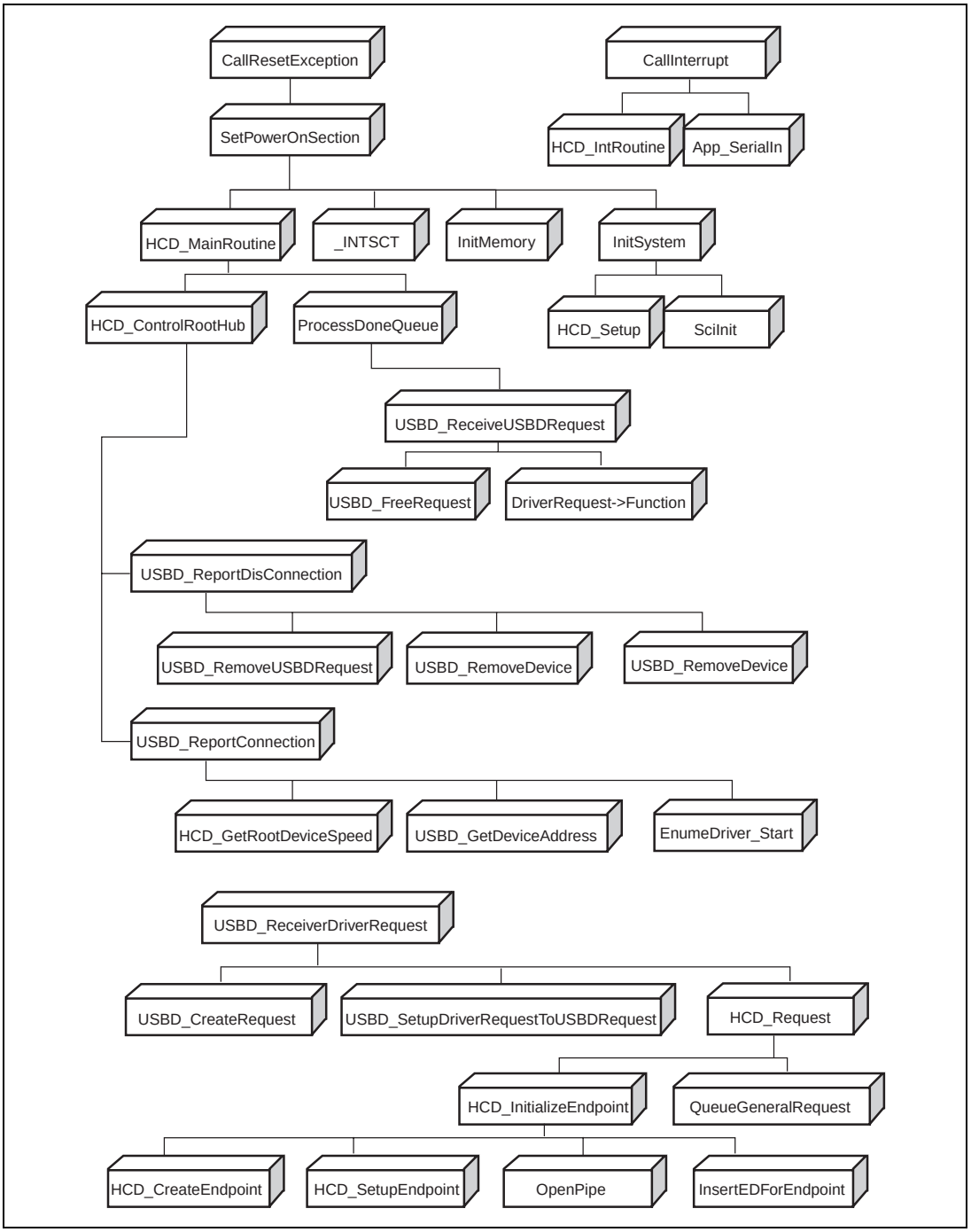
CallResetException
CallInterrupt
SetPowerOnSection
HCD_IntRoutine
App_SerialIn
HCD_MainRoutine
_INTSCT
InitMemory
InitSystem
HCD_ControlRootHub
ProcessDoneQueue
HCD_Setup
SciInit
USBD_ReceiveUSBDRequest
USBD_FreeRequest
DriverRequest->Function
USBD_ReportDisConnection
USBD_RemoveUSBDRequest
USBD_RemoveDevice
USBD_RemoveDevice
USBD_ReportConnection
HCD_GetRootDeviceSpeed
USBD_GetDeviceAddress
EnumeDriver_Start
USBD_ReceiverDriverRequest
USBD_CreateRequest
USBD_SetupDriverRequestToUSBDRequest
HCD_Request
HCD_InitializeEndpoint
QueueGeneralRequest
HCD_CreateEndpoint
HCD_SetupEndpoint
OpenPipe
InsertEDForEndpoint

図 4.4　**関数の相関関係**

5. サンプルプログラムの動作

この章ではサンプルプログラムの動作について説明します。

5.1 リセット状態

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、SetPowerOnSection 関数を呼び出します。**図** 5.1 にリセット割り込み発生から定常状態（メインループ）までのフローチャートを示します。

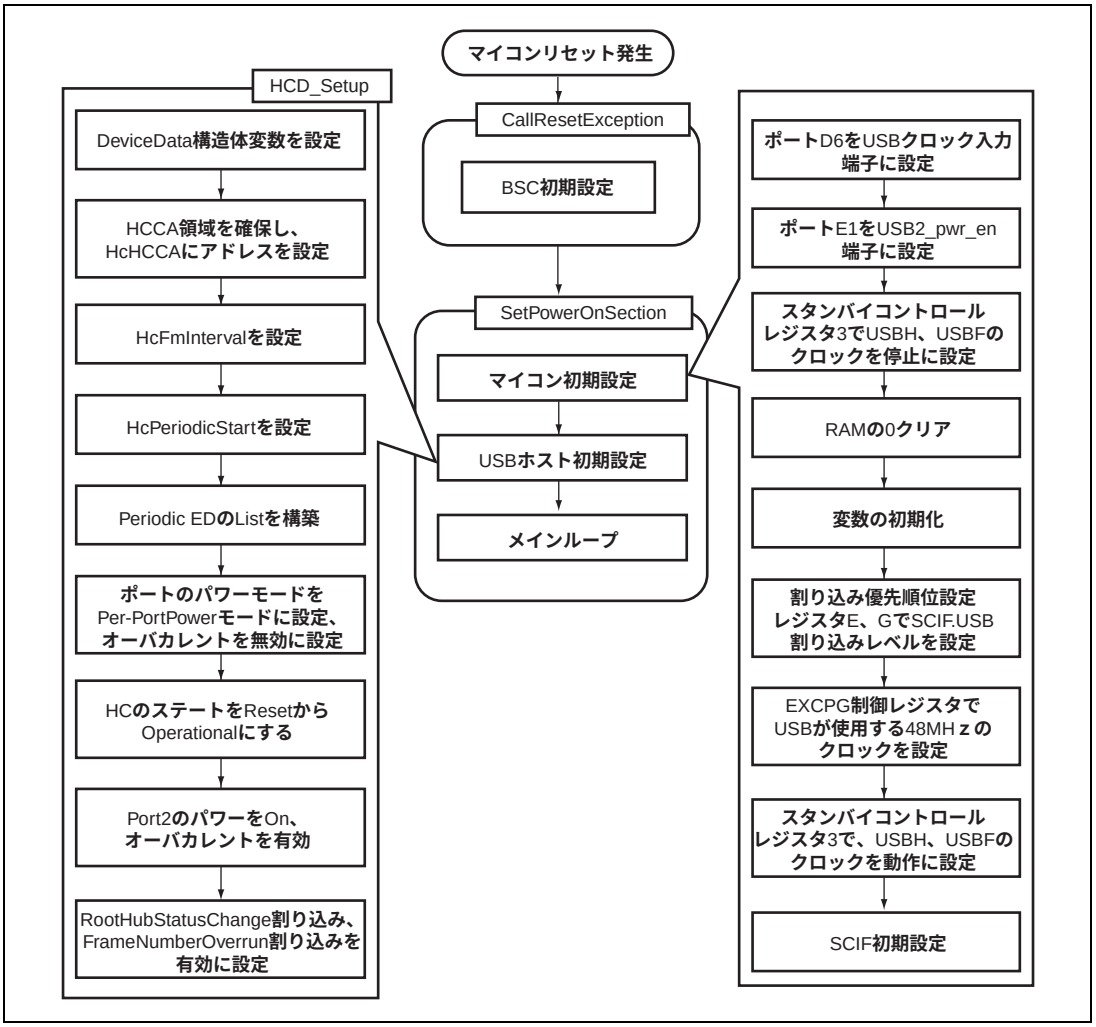

図 5.1 **リセット状態**

USB ホストに関するピン設定、割り込みレベル設定などを行い、USB ホストを使用できる状態に設定した後、USB ホストの初期設定を行います。USB ホストは初期設定完了後、接続待ち状態となます。

5.2 メインループ（接続待ち状態、定常状態）

リセット状態で種々の初期設定が完了後、この状態に遷移します。この状態では、USB ホストモジュールの割り込みが発生するのをメインループで待ちつづけます。**図** 5.2 にメインループのフローチャートを示します。

USB ホストモジュールの割り込みが発生すると、HCD_IntRoutine が呼ばれます。この関数は、USB ホストモジュールの割り込みが発生したことを示すフラグ（IntFlag）をセットし、USB ホストモジュールの割り込みを無効にして、すぐ終了します。実際に発生した割り込みに対応した処理は、メインルーチンで行います。

メインルーチンでは、常に USB ホストモジュールの割り込みが発生したかどうか監視しています。これは、IntFlag＝1 かどうかで判定しています。IntFlag＝1 の場合、発生した割り込みに応じた処理を行い、再び割り込みを有効にします。

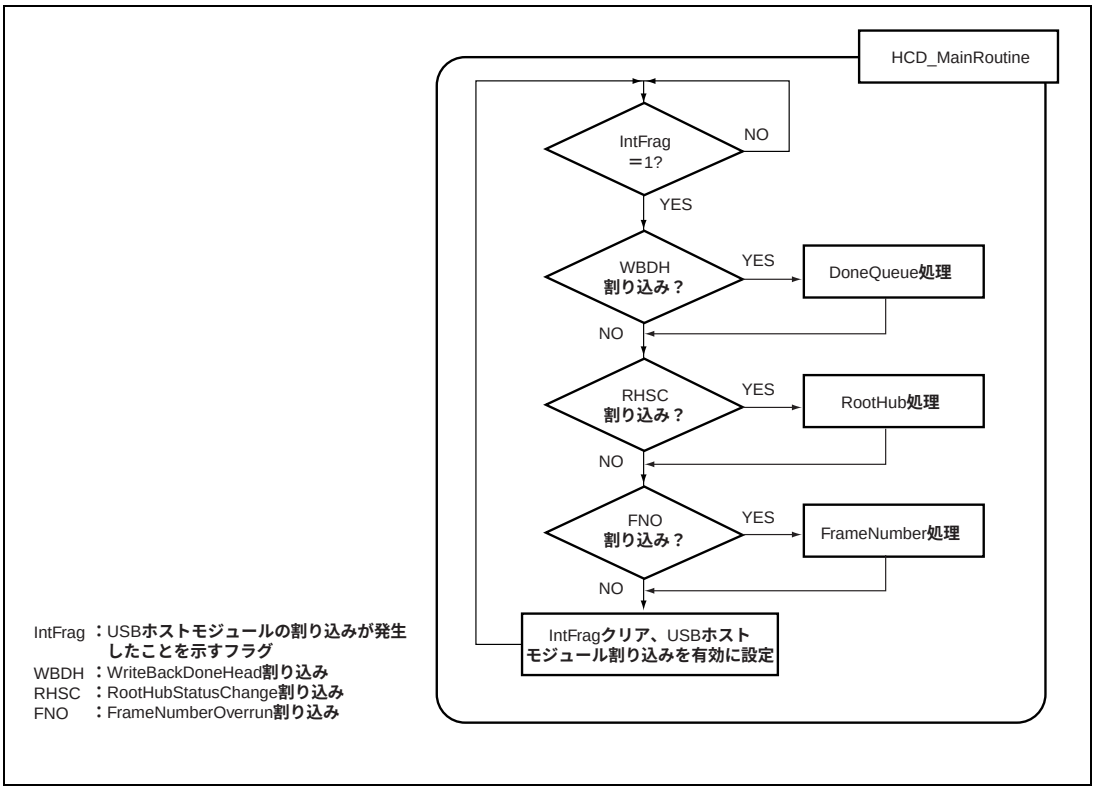

図 5.2 **メインループ**

5.3 RootHub 処理状態

RootHub 処理のフローチャートを**図** 5.3 に示します。この状態は、接続待ち状態もしくは定常状態、つまりメインループから、RootHubStatusChange 割り込みが発生した際に遷移します。この割り込みは、RootHub の状態が変化した際に発生します。「RootHub にデバイスが接続された場合」「RootHub に接続されていたデバイスが切断された場合」「RootHub のポートがオーバーカレント状態になった場合」の 3 つの状態を検出し、それぞれ図のように処理を行います。

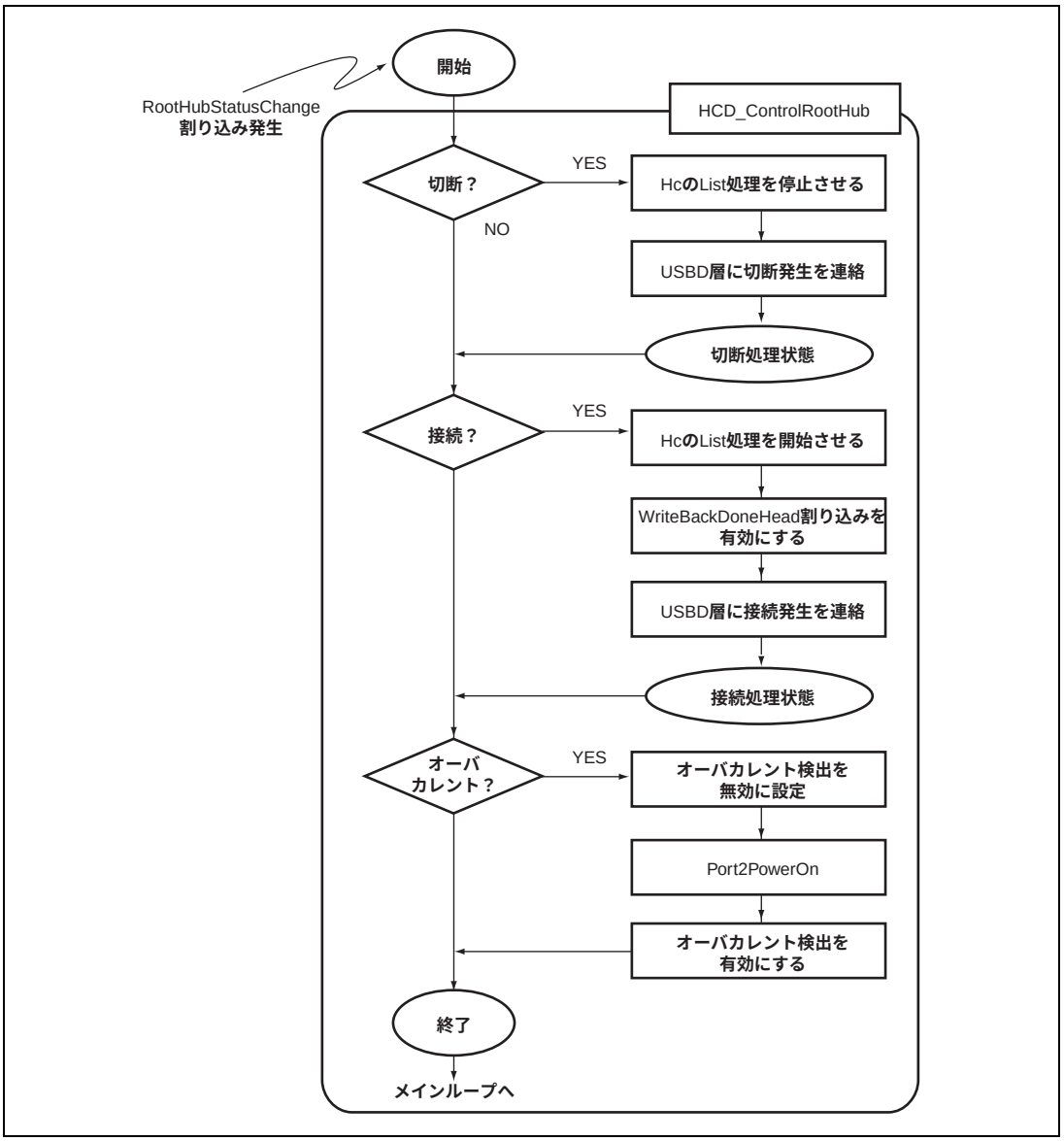

図 5.3 RootHub **処理**

5.4 接続処理状態

図 5.4 に接続されたデバイスに対する接続処理の詳細を示します。

「5.3 RootHub **処理状態」**で示したように、デバイスが接続された場合、RootHubStatusChange 割り込みが発生し、USBD 層にデバイス接続されたことが連絡されます。このとき HCD_ControlRootHub 関数から、USBD_ReportConnection 関数が呼ばれます。この関数では、USBD 層を初期化して、EnumerationDriver を呼び出します。EnumerationDriver では、SetAddress 処理を行い DeviceAddress を割り当て、GetDescriptor 処理によって接続されたデバイスの Descriptor 情報を取得します。そして、取得した Descriptor 情報から、接続されたデバイスの Class 情報により、最適な Driver を判定し、呼び出します。

【注】 なお、本サンプルプログラムでは、接続されたデバイスの制御は PC **上のツール「**RequestGenerator**」から行いますので、**Driver **の呼び出しは行われません。**

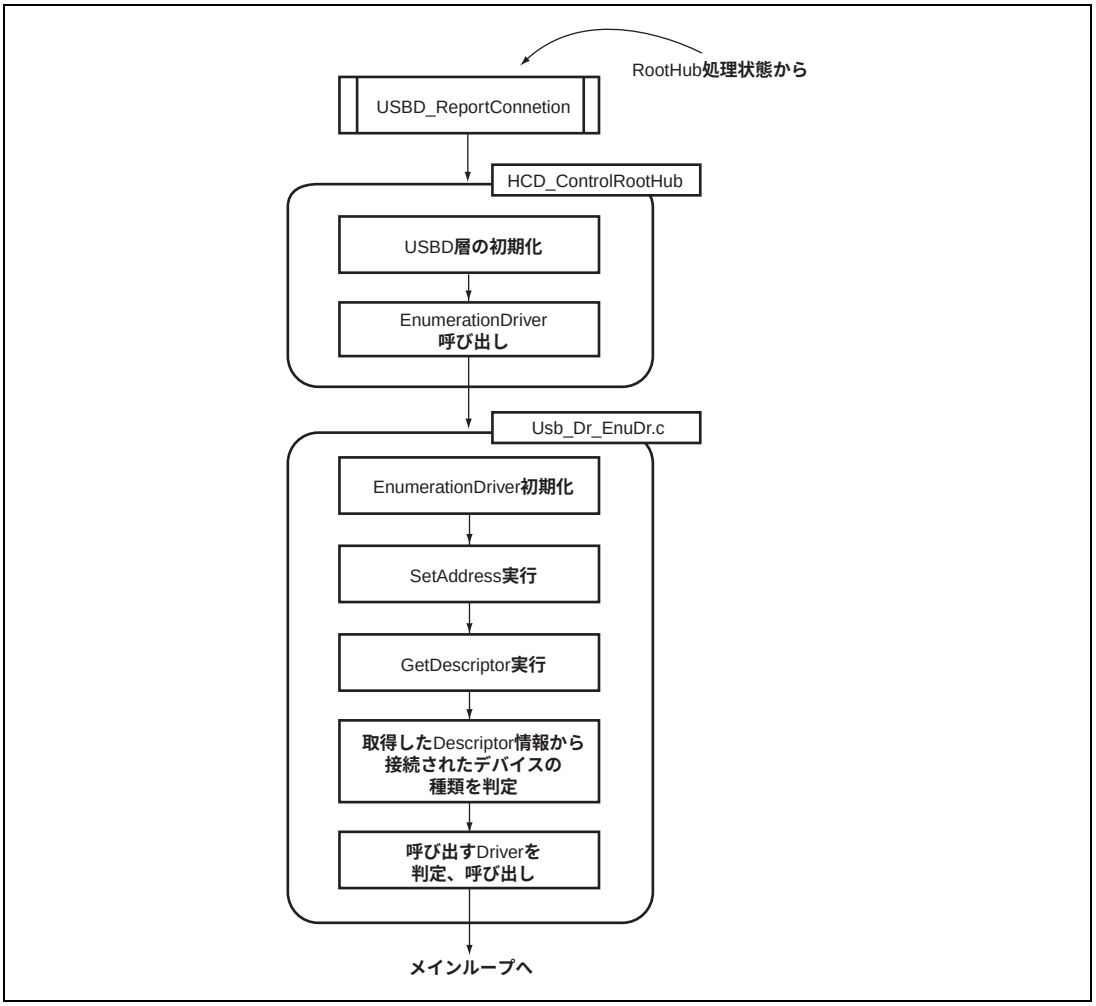

図 5.4 **接続処理**

5.5 シリアル入力状態（RequestGeneratorDriver 処理状態）

図 5.5 にシリアル入力処理のフローチャートを示します。

PC 上のツール「RequestGenerator」で“ Start ”ボタンを押すことで、USB パケット機能を有効にするためのコマンド「RG」を受信します。これにより、SH7727SE 側では、RequestGeneratorDriver が読み込まれ、EnumerationDriver が無効状態になります。USB-A コネクタに任意の USB ファンクションデバイスを接続すると、EnumerationDriver が読み込まれず、RequestGeneratorDriver・RequestGenerator からコントロールできる状態になります。以後は、RequestGenerator からの転送要求を RequestGeneratorDriver がデコードし、実行し、結果を RequestGenerator に返します。

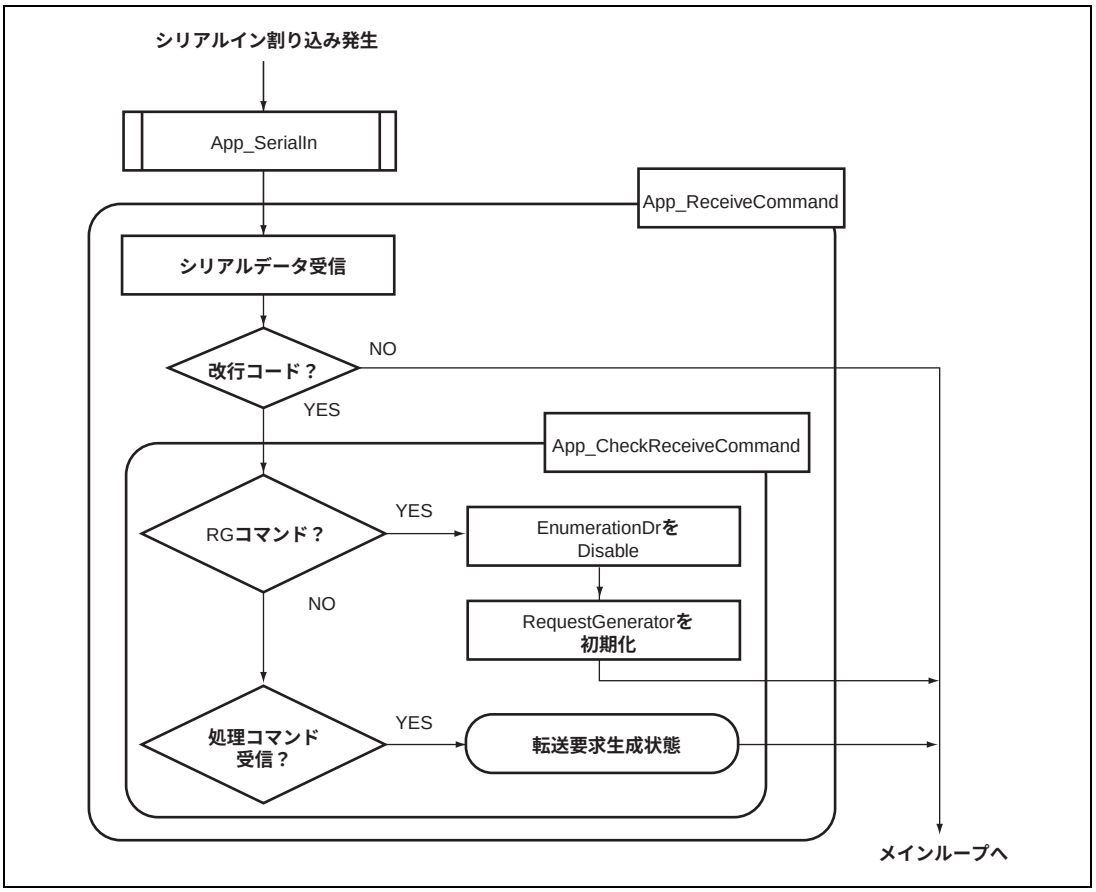

図 5.5 **シリアル受信状態（RequestGeneratorDriver 処理状態）**

5.6 転送要求生成状態

この状態は、RootHub にデバイスが接続され、RootHubStatusChange 割り込みが発生し、接続処理が完了した際に遷移します。この状態では、Driver 層からの転送要求を USBD 層経由で HCD が受け、その転送要求を ED・TD に纏め、List を生成し、HC に渡します。転送要求時のフローチャートを**図** 5.6 に示します。

本サンプルソフトウェアでは、シリアル受信割り込みにより PC 上の「RequestGenerator」からの転送要求を RequestGeneratorDriver が受けて、HCD への転送要求を行います。HCD はそれに従い、ED、TD を生成し、HC へ渡します。

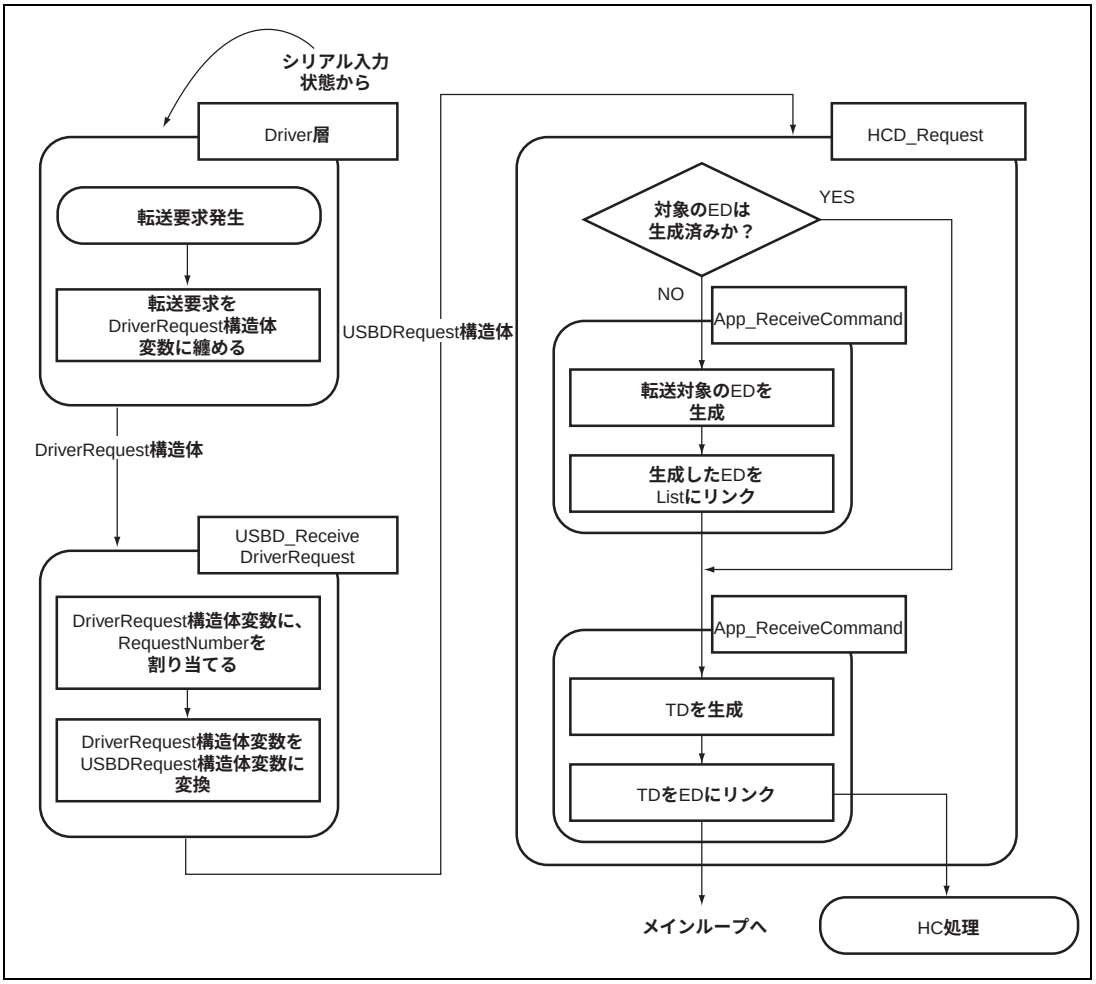

図 5.6 **転送要求処理**

5.7 DoneQueue 処理状態

DoneQueue 処理状態は、定常状態から、WriteBackDoneHead 割り込みが発生した際に遷移します。**図** 5.6 で示したように転送要求を行うと、HC は受取った ED・TD からデータパケットを生成して USB 通信を行います。そして受け取った TD に関する処理が完了した際に、この WriteBackDoneHead 割り込みが発生します。**図** 5.7 にフローチャートを示します。

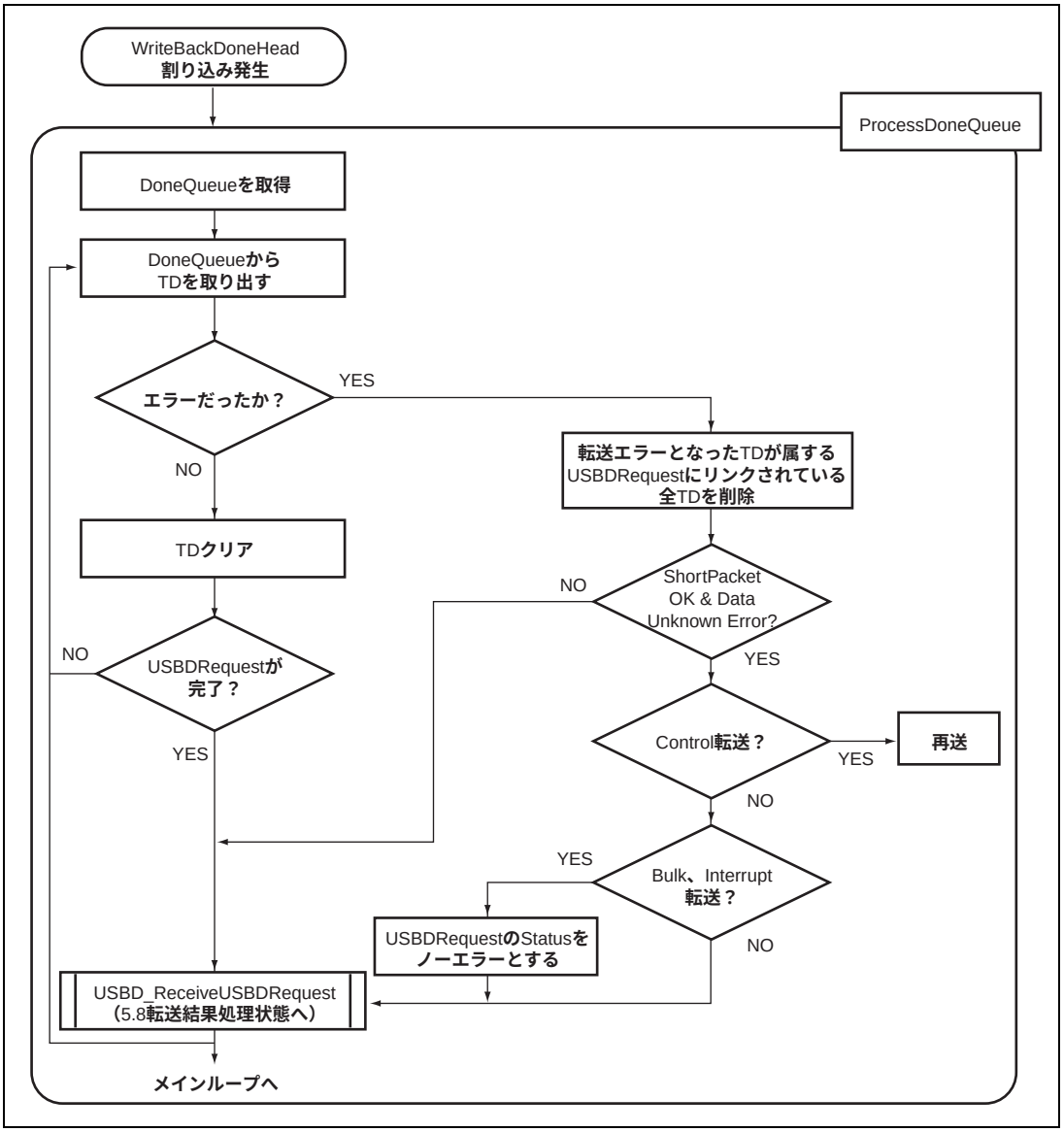

図 5.7　DoneQueue **処理状態**

5.8 転送結果処理状態

この状態は、WriteBackDoneHead 割り込みにより実行される DoneQueue 処理状態から遷移します。DoneQueue 処理関数から、処理完了した TD を USBDRequerst の形で受け取り、転送結果の判定を行い、転送依頼元の Driver に結果を返します。その詳細を**図** 5.8 に示します。

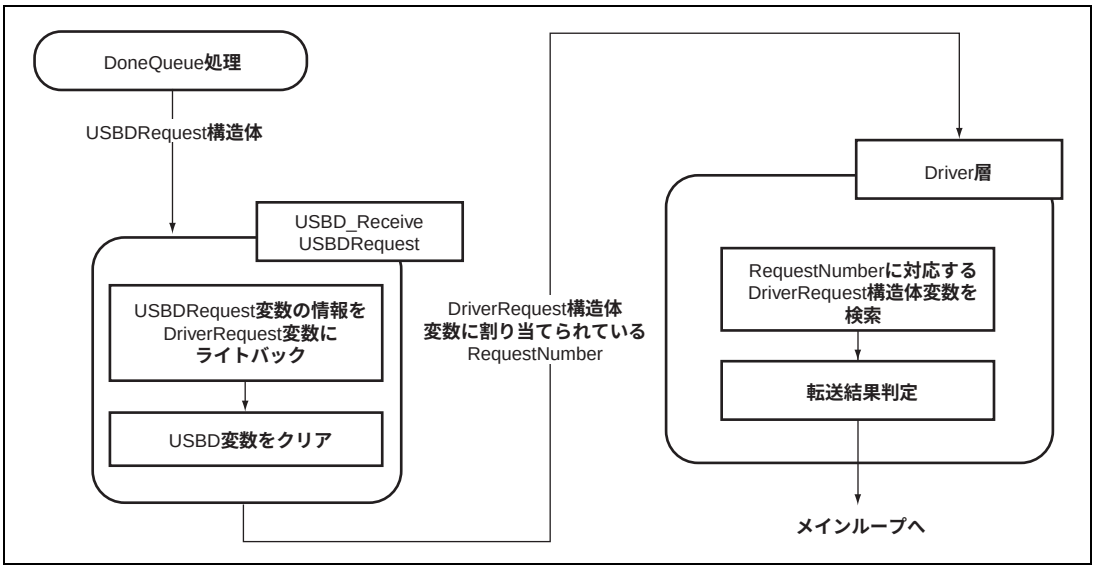

図 5.8 **転送結果処理時のフローチャート**

SH7727 USBホストモジュールアプリケーションノート

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

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

RJJ05B0015-0100Z