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。

RENESAS

アプリケーション・ノート

PFESiP/V850EP1

PFESiP® EP-1 専用 32 ビット・マイクロコントローラ

USB ファンクション・サンプル・ソフトウエア編

資料番号 A19349JJ1V0AN00（第 1 版）
発行年月 August 2008 NS

[メ　モ]

目次要約

CMOSデバイスの一般的注意事項

① 入力端子の印加波形

入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。

CMOSデバイスの入力がノイズなどに起因して，V_{IL}（MAX.）からV_{IH}（MIN.）までの領域にとどまるような場合は，誤動作を引き起こす恐れがあります。入力レベルが固定な場合はもちろん，V_{IL}（MAX.）からV_{IH}（MIN.）までの領域を通過する遷移期間中にチャタリングノイズ等が入らないようご使用ください。

② 未使用入力の処理

CMOSデバイスの未使用端子の入力レベルは固定してください。

未使用端子入力については，CMOSデバイスの入力に何も接続しない状態で動作させるのではなく，プルアップかプルダウンによって入力レベルを固定してください。また，未使用の入出力端子が出力となる可能性（タイミングは規定しません）を考慮すると，個別に抵抗を介してV_{DD}またはGNDに接続することが有効です。

資料中に「未使用端子の処理」について記載のある製品については，その内容を守ってください。

③ 静電気対策

MOSデバイス取り扱いの際は静電気防止を心がけてください。

MOSデバイスは強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には，当社が出荷梱包に使用している導電性のトレーやマガジン・ケース，または導電性の緩衝材，金属ケースなどを利用し，組み立て工程にはアースを施してください。プラスチック板上に放置したり，端子を触ったりしないでください。

また，MOSデバイスを実装したボードについても同様の扱いをしてください。

④ 初期化以前の状態

電源投入時，MOSデバイスの初期状態は不定です。

電源投入時の端子の出力状態や入出力設定，レジスタ内容などは保証しておりません。ただし，リセット動作やモード設定で定義している項目については，これらの動作ののちに保証の対象となります。

リセット機能を持つデバイスの電源投入後は，まずリセット動作を実行してください。

⑤ 電源投入切断順序

内部動作および外部インタフェースで異なる電源を使用するデバイスの場合，原則として内部電源を投入した後に外部電源を投入してください。切断の際には，原則として外部電源を切断した後に内部電源を切断してください。逆の電源投入切断順により，内部素子に過電圧が印加され，誤動作を引き起こしたり，異常電流が流れ内部素子を劣化させたりする場合があります。

資料中に「電源投入切断シーケンス」についての記載のある製品については，その内容を守ってください。

⑥ 電源OFF時における入力信号

当該デバイスの電源がOFF状態の時に，入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により，誤動作を引き起こしたり，異常電流が流れ内部素子を劣化させたりする場合があります。

資料中に「電源OFF時における入力信号」についての記載のある製品については，その内容を守ってください。

本製品が外国為替及び外国貿易法の規定により規制貨物等に該当するか否かは，ユーザ（仕様を決定した者）が判定してください。該当する場合，日本国外に輸出する際に，同法に基づき日本国政府の輸出許可が必要です。

- **本資料に記載されている内容は2008年8月現在のもので，今後，予告なく変更することがあります。**
  量産設計の際には最新の個別データ・シート等をご参照ください。
- 文書による当社の事前の承諾なしに本資料の転載複製を禁じます。当社は，本資料の誤りに関し，一切その責を負いません。
- 当社は，本資料に記載された当社製品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，一切その責を負いません。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。
- 本資料に記載された回路，ソフトウエ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，ソフトウエ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，一切その責を負いません。
- 当社は，当社製品の品質，信頼性の向上に努めておりますが，当社製品の不具合が完全に発生しないことを保証するものではありません。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。当社製品をお客様の機器にご使用の際には，当社製品の不具合の結果として，生命，身体および財産に対する損害や社会的損害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を行ってください。
- 当社は，当社製品の品質水準を「標準水準」，「特別水準」およびお客様に品質保証プログラムを指定していただく「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。

  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

  特別水準：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

  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，生命維持のための装置またはシステム等

  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。意図されていない用途で当社製品の使用をお客様が希望する場合には，事前に当社販売窓口までお問い合わせください。

（注）

（1）本事項において使用されている「当社」とは，NECエレクトロニクス株式会社およびNEC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う。

（2）本事項において使用されている「当社製品」とは，（1）において定義された当社の開発，製造製品をいう。

M8E0710J

はじめに

対 象 者　このマニュアルはV850E2 CPUコア内蔵マイクロコントローラ機能チップ「PFESiP/V850EP1」の機能を理解し，それを用いたPFESiP EP-1シリーズ製品を開発検討するユーザを対象とします。

目　　的　このマニュアルは，PFESiP/V850EP1のUSBファンクション・サンプル・ソフトについて，ユーザに理解していただくことを目的としています。

読 み 方　このマニュアルの読者には，電気，論理回路，マイクロコンピュータ，SRAM，ページROM，SDRAMに関する一般知識を必要とします。

凡　　例

データ表記の重み	：左が上位桁，右が下位桁
アクティブ・ロウの表記	：xxxZ（端子，信号名称のあとにZ）
注	：本文中につけた注の説明
注意	：気をつけて読んでいただきたい内容
備考	：本文の補足説明
数の表記	：2進数 … xxxx または xxxxB
	10進数 … xxxx
	16進数 … xxxxH
2 のべき数を示す接頭語（アドレス空間，メモリ容量）	：K（キロ）… 2^{10} = 1024
	M（メガ）… 2^{20} = 1024^2
	G（ギガ）… 2^{30} = 1024^3
データ・タイプ	：ワード … 32ビット
	ハーフワード … 16ビット
	バイト … 8ビット

関連資料　**関係資料は暫定版の場合がありますが，この資料では「暫定」の表示をしておりません。あらかじめ，ご了承ください。また各コアの開発・企画段階で資料を作成しているため，関連資料は個別のお客様向け資料の場合があります。**

PFESiP EP-1 **シリーズに関する資料**

資料名	資料番号
V850E2　**ユーザーズ・マニュアル　アーキテクチャ編**	U17135J
PFESiP　EP-1 **シリーズ　設計マニュアル**	A19068J
PFESiP/V850EP1　**ユーザーズ・マニュアル　製品データ編**	A19069J
PFESiP/V850EP1　**ユーザーズ・マニュアル　ハードウエア編（**CPU **機能）**	A19070J
PFESiP/V850EP1　**ユーザーズ・マニュアル　ハードウエア編（**USB **機能）**	A19071J
PFESiP/V850EP1　**アプリケーション・ノート** USB **ファンクション・サンプル・ソフトウエア編**	**このマニュアル**

PFESiP EP-1 Evaluation Board **に関する資料**

資料名	資料番号
PFESiP EP-1 Evaluation Board　**ユーザーズ・マニュアル　技術情報編**	A19350J
PFESiP EP-1 Evaluation Board　**ユーザーズ・マニュアル　オーダ情報編**	A19352J
PFESiP EP-1 Evaluation Board　**ユーザーズ・マニュアル**　FPGA **設計ガイド編**	A19351J
PFESiP EP-1 Evaluation Board Lite　**ユーザーズ・マニュアル　技術情報編**	A19354J

開発ツールに関する資料（ユーザーズ・マニュアル）

資料名		資料番号
RX850 Pro（Ver.3.20）**（リアルタイム** OS）	**基礎編**	U13773J
	インストレーション編	U17421J
	テクニカル編	U13772J
	タスク・デバッガ編	U17422J

目　次

図の目次

表の目次

第1章　イントロダクション

1.1 概　　要

USB ストレージ・クラス用ドライバは，PFESiP/V850EP1 に内蔵している USB ファンクション・コントローラ用のサンプル・プログラムです。Universal Serial Bus Specification Revision 1.1，および Universal Serial Bus Mass Storage Class Bulk-Only Transport Revision 1.0 に準拠しており，組み込み型制御用リアルタイム・オペレーティング・システム RX850 Pro（μITRON 3.0 仕様準拠）上で動作します。

このサンプル・プログラムでは，コントロール・エンドポイント（エンドポイント番号 0），およびバルク・エンドポイントの IN と OUT（エンドポイント番号 1, 2）の 3 つを使用します。その上で，Windows XP に標準のストレージ・クラス用ホスト・ドライバと接続し，ストレージ・デバイス（仮想デバイス）の制御を行います。クラスとしては，Mass Storage クラスが定義されています。

このサンプル・プログラムは，ハードウエアの実行環境として開発評価ボード PFESiP EP-1 Evaluation Board（PFESiP EP-1 Evaluation Board Lite も含む。以降省略）を使用しています。PFESiP EP-1 Evaluation Board とサンプル・プログラムをそのまま使用する場合は，第 6 章　ロード・モジュールの手順に従って実行オブジェクトを作成し，第 2 章　ロード・モジュールの実行に従って動作を確認してください。

PFESiP EP-1 Evaluation Board からほかのターゲット・ボードに変更して使用する場合は，第 3 章　システム構築，第 4 章　RX850 Pro 依存処理プログラム，第 5 章　リンク・ディレクティブ・ファイルを参照し，ボードの仕様に従って変更してください。

PFESiP EP-1 Evaluation Board およびサンプル・プログラムの両方を変更して使用する場合は，第 3 章　システム構築，第 4 章　RX850 Pro 依存処理プログラム，第 5 章　リンク・ディレクティブ・ファイル，第 6 章　ロード・モジュール，第 7 章　USB ストレージ・クラス用ドライバの機能を参照し，必要な変更を行ってください。

USB ストレージ・クラス用ドライバの位置付けを次に示します。

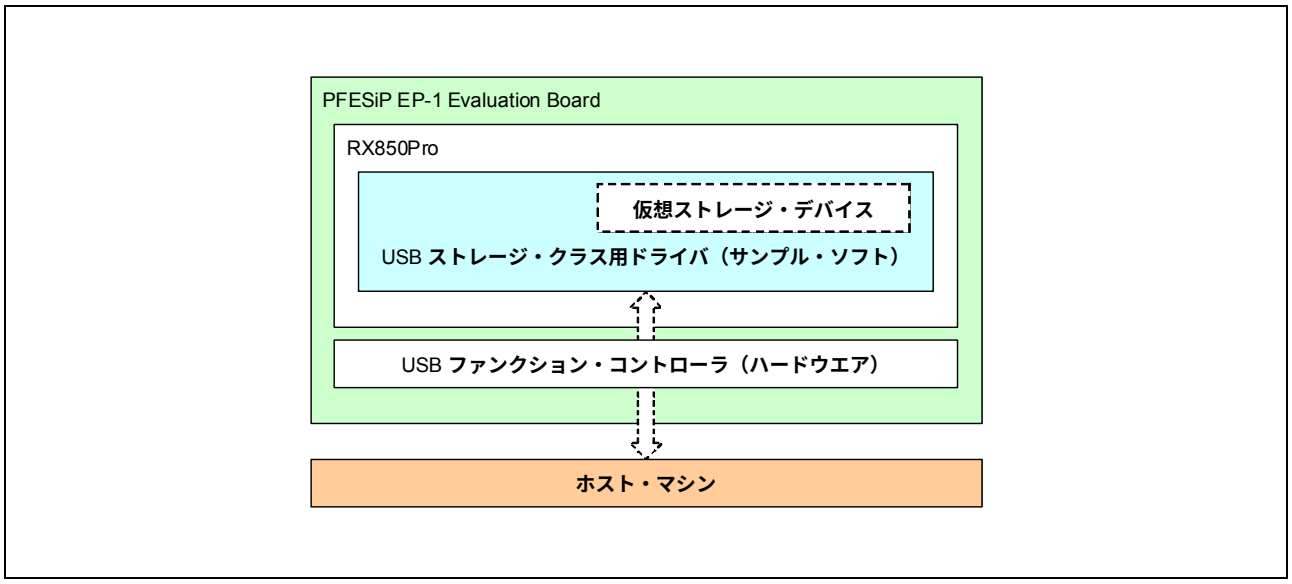
注意　このサンプル・プログラムは，Mass Storage デバイス（インタフェース・クラス：マス・ストレージ，インタフェース・サブクラス：SCSI，インタフェース・プロトコル：Bulk-Only Transport プロトコル）として動作します。今回使用するストレージ・デバイスは，結合される論理ユニットはなく，メモリ領域を確保し擬似的にリムーバブル・ディスクが繋がれているように動作するものです（ブロック・サイズ：512 バイト，論理ブロック数：192，容量：96 K バイト）。

備考 1.　USB の Mass Storage クラスの詳細は，以下を参照してください。

- Universal Serial Bus Mass Storage Class Specification Overview Revision 1.1
- Universal Serial Bus Mass Storage Class Bulk-Only Transport Revision 1.0
- Universal Serial Bus Mass Storage Class UFI Command Specification Revision 1.0

2.　2.1　ロード・モジュールの実行手順は，1.3　実行環境で示した環境を想定して記述しています。

図1－1　USBストレージ・クラス・ドライバの位置付け



1.2　開発環境

サンプル・プログラムを使用してシステムを開発するうえで，必要となるハードウエア環境およびソフトウエア環境として，次に示す環境を想定して記述しています。

・ハードウエア環境

ホスト・マシン	：PC/ATTM 互換機（OS：Windows XP）

・ソフトウエア環境

リアルタイム OS	：RX850 Pro Version 3.21
USB ストレージ・クラス・ドライバ	：この章で説明するサンプル・プログラム一式
プロジェクト・マネージャ	：PM+ Version 6.30
C コンパイラ・パッケージ	：CA850 Version 3.10

注意　サンプルのプロジェクト・ファイルで扱うディレクトリ構成と違う場合は，環境にあわせてプロジェクト・ファイルを編集してください。

備考　プロジェクト・ファイルの編集方法については，プロジェクト・マネージャのヘルプを参照してください。

1. 3　実行環境

サンプル・プログラムを使用したロード・モジュールを実行するときのハードウエア環境およびソフトウエア環境として，次に示す環境を想定して記述しています。

・ハードウエア環境

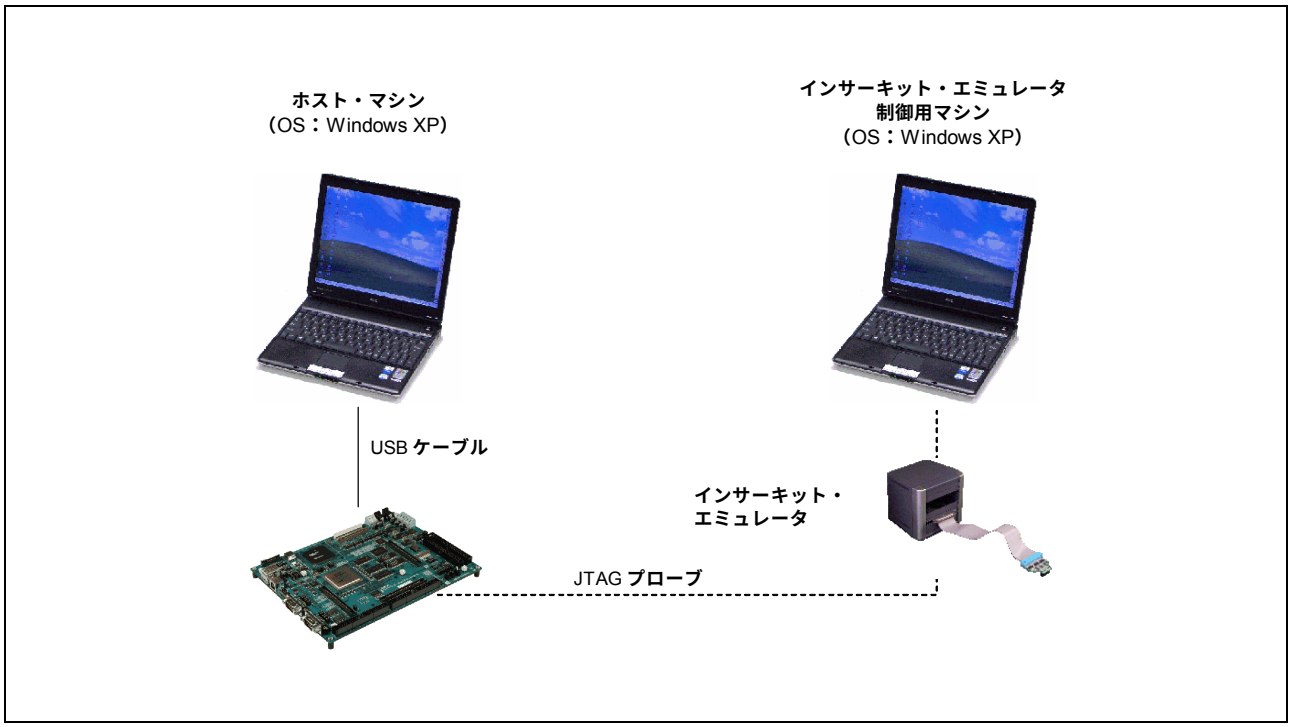
ホスト・マシン　　：PC/AT 互換機（OS：Windows XP）
IE 制御用マシン　　：PC/AT 互換機（OS：Windows XP）
ターゲット・ボード　　：PFESiP EP-1 Evaluation Board / PFESiP EP-1 Evaluation Board Lite
インサーキット・エミュレータ（IE）：RTE-2000-TP（マイダス・ラボ社製）
JTAG プローブ
USB ケーブル

・ソフトウエア環境

プロジェクト・マネージャ　：PM+ Version 6.30
IE 用ソフトウエア　　：ID850QB Version 3.40

備考 1. 実行環境のセットアップ方法についての詳細は，PFESiP EP-1 Evaluation Board のユーザーズ・マニュアルを参照してください。
2. インサーキット・エミュレータ（RTE-2000-TP）セットアップ方法についての詳細は，RTE-2000-TP のユーザーズ・マニュアルを参照してください。
3. ID850QB についての詳細は，ID850QB のユーザーズ・マニュアルを参照してください。

図1－2　実行環境



第2章　ロード・モジュールの実行

2. 1　ロード・モジュールの実行手順

このサンプル・プログラムを使用したロード・モジュールを例にとり，1. 3　**実行環境**で示した環境において実行するときの手順を，次に示します。

① **インサーキット・エミュレータ**（IE）**制御用マシンの準備**

IE **制御用マシンに電源を投入し起動しておきます。**

また，インサーキット・エミュレータにも電源を投入し準備しておきます。

② **ホスト・マシンの準備**

ホスト・マシンに電源を投入し起動しておきます（ホスト・マシンは，IE **制御用マシンで兼用できますが，開発時には別のホスト・マシンを用意することを強く推奨します)。**

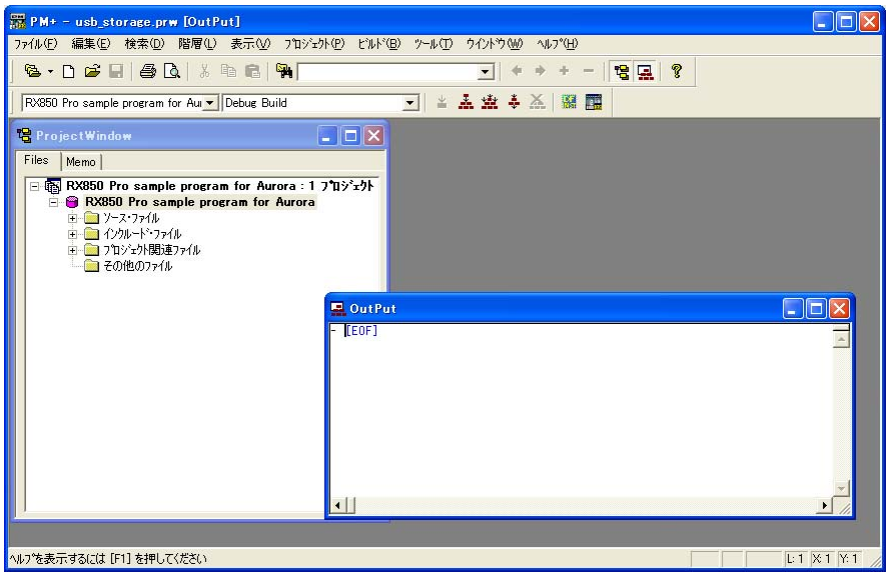
③ PFESiP EP-1 Evaluation Board **のリセット**

PFESiP EP-1 Evaluation Board **の** SW_RESET **を手前に引き，**PFESiP EP-1 Evaluation Board **のリセットを行います。**

④ **プロジェクト・マネージャ**（PM+）**の起動**

Windows **のエクスプローラにて，サンプル・プログラムの「**V850USB_Storage¥rx85p¥conf**」ディレクトリに格納されているワークスペース・ファイル：**usb_storage.prw **をダブル・クリックし，**PM+**を起動します。**

[PM+**の起動画面**]



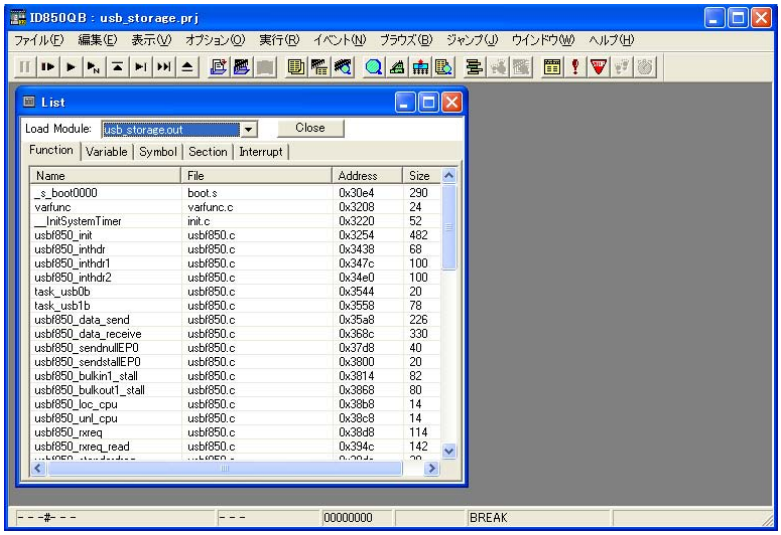
備考　**サンプル・プログラムのディレクトリ構成の詳細は，**2.2　**ディレクトリ構成を参照してください。**

⑤ **デバッガ（ID850QB）の起動およびロード・モジュールの読み込み**

PM+**のツールバーの「ビルド」－「デバッグ」を選択し，**ID850QB **を立ち上げます。**

サンプル・プログラムのデバッガ情報ファイル：usb_storage.pri **では，ロード・モジュールの読み込みも自動的に行われるよう設定していますので,**PFESiP EP-1 Evaluation Board **の** PFESiP/V850EP1 **内蔵命令** RAM **にロード・モジュールが読み込まれると共に，下図のようにデバッグ情報もデバッガに読み込まれます。**

［ID850QB **の起動画面］**



⑥ **実行**

F5 **キー，**Go **ボタンまたはツールバーの「実行」－「継続して実行」で，ボード上に読み込まれたコードを実行します。**

［プログラム実行例］

⑦　USB 接続

USB ケーブルを接続します。

ボード側に B コネクタを接続し，ホスト・マシン側に A コネクタを接続します。

ホスト・マシン側でデバイスが認識されると，Windows XP 標準の USB ストレージ・クラス用ホスト・ドライバが自動的にインストールされます。

正常に認識されると，「マイ コンピュータ」上に「リムーバブル ディスク」として表示されます。

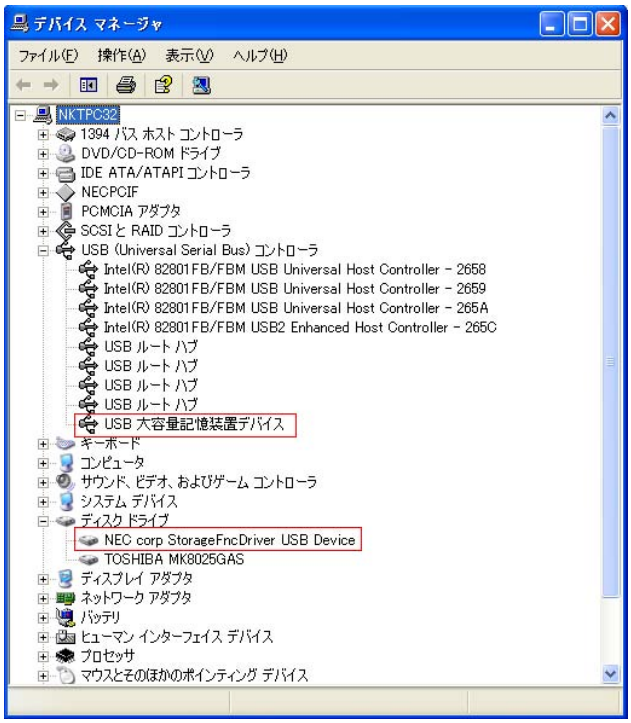
［マイコンピュータ表示例］

⑧　デバイス マネージャの起動

「マイコンピュータ」の「プロパティ」から「ハードウエア」のタブを選択します。

「デバイスマネージャ」の項目を選択し，「デバイス マネージャ」を起動します。

［デバイスマネージャ表示例］



⑨　USB デバイス接続の確認

「デバイス マネージャ」の画面上で，「USB コントローラ」の下に「USB 大容量記憶装置デバイス」が，「ディスクドライブ」の下に「NEC corp StorageFncDriver USB Device」が表示されていることを確認します。

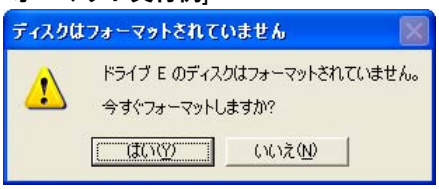
⑩　デバイスの使用方法

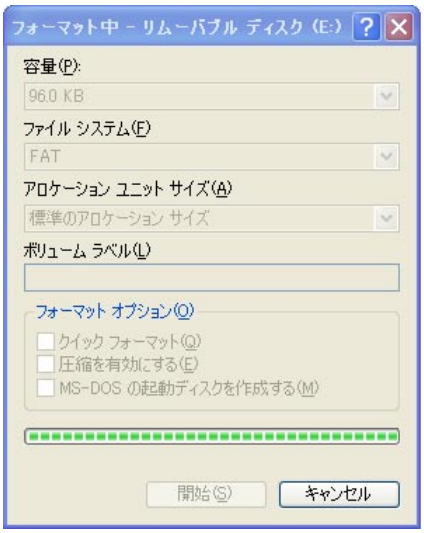
まず，「マイコンピュータ」上に表示された「リムーバブル ディスク」の「開く」を表示すると，ディスクのフォーマットが要求されます。画面の指示にしたがってフォーマットを実行してください。

フォーマットが正常に終了すると，ファイルの読み書き，ファイルの削除など，通常のディスク・デバイスと同じように使用できることが確認できます。

また，ロード・モジュールの実行を止めるまでは，ディスクの内容が保持されます。

[フォーマット実行例]



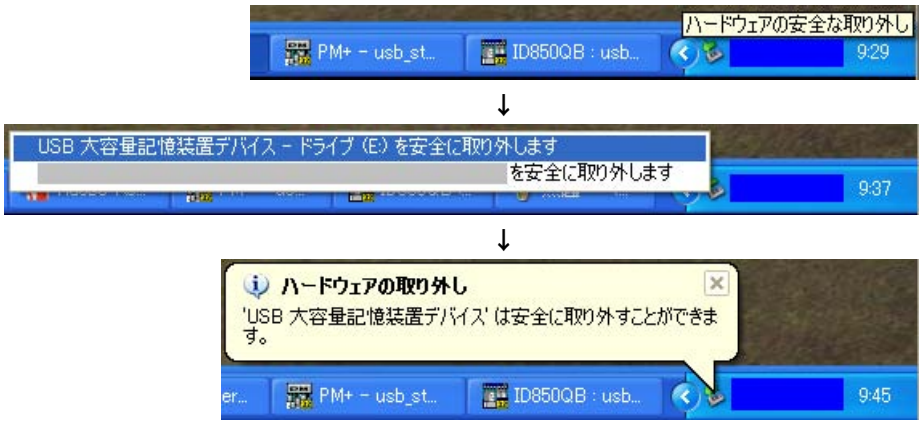


⑪　**デバイスの使用停止**

Windows **のタスクトレイに入っている「ハードウエアの安全な取り外し」を左クリックします。**

⑩で使用していたドライブの USB **大容量記憶装置デバイスを選択し，デバイスを停止します。**

[デバイス使用停止例]



⑫　USB **ケーブルの取り外し**

⑪でデバイスの使用停止が確実に行われたことを確認し，USB **ケーブルを取り外してください。**

⑬　**プログラムの終了方法**

実行中のプログラムを終了します。

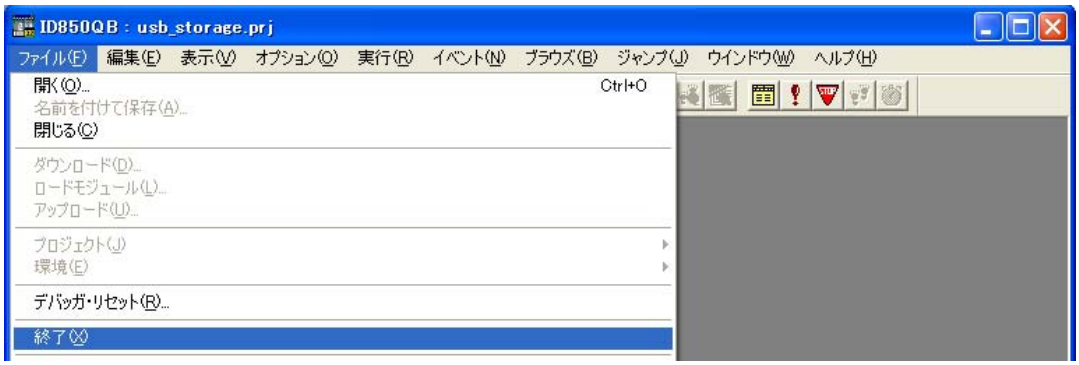
F2 **キー，**Stop **ボタンまたはツールバーの「実行」－「ストップ」で，プログラムの実行を停止します。**

[プログラム終了例]

⑭　ID850QB **の終了方法**

ツールバーの「ファイル」－「終了」を選択し，ID850QB **を終了してください。**

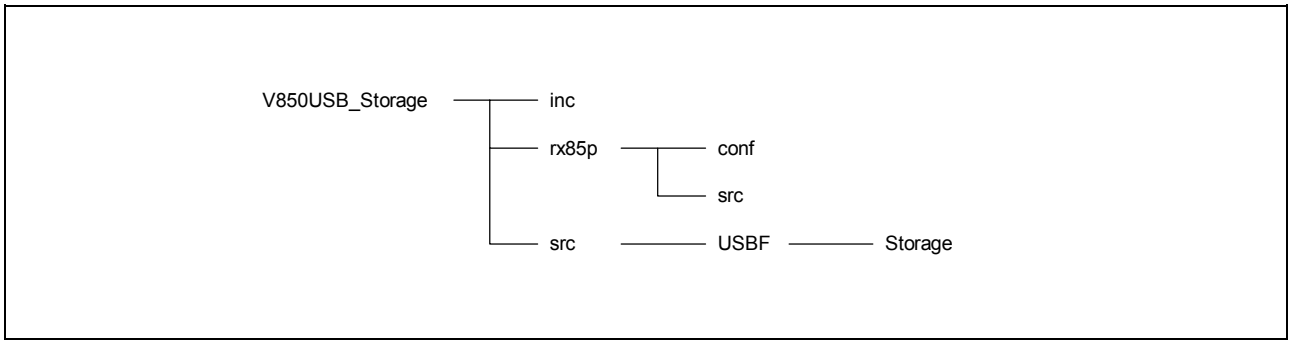
[ID850QB **の終了例]**



2.2 ディレクトリ構成

このサンプル・プログラム一式に含まれるファイル群のディレクトリ構成を，次に示します。

図2－1　サンプル・プログラムのディレクトリ構成



各ディレクトリの概要を次に示します。

（1）V850USB_Storage
USB ストレージ・クラス用ドライバのディレクトリです。

（2）V850USB_Storage¥inc
USB ストレージ・クラス用ドライバのヘッダ・ファイルが格納されているディレクトリです。

* aurora_usb_reg.h　：PFESiP/V850EP1 の USB 関連レジスタ定義ファイル
* errno.h　：戻り値用ヘッダ・ファイル
* types.h　：データ・タイプ用ヘッダ・ファイル

（3）V850USB_Storage¥rx85p
RX850 Pro 用のファイル群が格納されているディレクトリです。

（4）V850USB_Storage¥rx85p¥conf
RX850 Pro 用のシステム・ファイルが格納されているディレクトリです。

* usb_storage.prw　：USB ストレージ・クラス用ドライバのワークスペース・ファイル
* usb_storage.prj　：USB ストレージ・クラス用ドライバのプロジェクト・ファイル
* usb_storage.pri　：USB ストレージ・クラス用ドライバのデバッガ情報ファイル
* usb_storage.tcl　：ID850QB 起動時動作設定スクリプト・ファイル
* usb_storage.out　：USB ストレージ・クラス用ドライバのロード・モジュール
* sys.h　：システム情報ヘッダ・ファイル
* sys.s　：システム情報テーブル
* sct.s　：システム・コール・テーブル

注意　sys.h（システム情報ヘッダ・ファイル），sys.s（システム情報テーブル），sct.s（システム・コール・テーブル）は，ビルド時に生成されるファイルです。本来，これらのファイルは，コンフィギュレー

タを使用してコマンド操作により生成されます。しかし，サンプルのビルド・ファイルを使用すると，ビルド実行時にコマンドを実行する設定になっているため，あらかじめ生成しておく必要はありません。

（5）V850USB_Storage¥rx85p¥src

RX850 Pro 用のファイルが格納されているディレクトリです。

* boot.s　：ブート処理用アセンブラ・ファイル
* entry.s　：エントリ処理用アセンブラ・ファイル
* init.c　：ハードウエア初期化部ソース・ファイル
* init.h　：ハードウエア初期化部ヘッダ・ファイル
* sys.cf　：CF 定義ファイル
* varfunc.c　：ソフトウエア初期化部ソース・ファイル
* usb_storage.dir　：リンク・ディレクティブ・ファイル

（6）V850USB_Storage¥src

USB ストレージ・クラス用ドライバのボード依存部のファイルが格納されているディレクトリです。

* port.c　：ポート設定用ソース・ファイル
* port.h　：ポート設定用ヘッダ・ファイル

（7）V850USB_Storage¥src¥USBF

USB ストレージ・クラス用ドライバの USB 処理部のファイルが格納されているディレクトリです。

* usbf850.c　：USB デバイス用ソース・ファイル
* usbf850.h　：USB デバイス用ヘッダ・ファイル
* usbf850desc.h　：USB 用ディスクリプタ定義ファイル
* usbf850_dma.c　：DMA 制御用ソース・ファイル
* usbf850_dma.h　：DMA 制御用ヘッダ・ファイル
* usbf850_storage.c　：USB-ストレージ間インタフェース用ソース・ファイル
* usbf850_storage.h　：USB-ストレージ間インタフェース用ヘッダ・ファイル

（8）V850USB_Storage¥src¥USBF¥Storage

USB ストレージ・クラス用ドライバのストレージ・デバイス処理部のファイルが格納されているディレクトリです。

* ata_ctrl.c　：ストレージ・デバイス・コントロール用ソース・ファイル
* ata.h　：ストレージ・デバイス用ヘッダ・ファイル
* scsi_cmd.c　：SCSI コマンド処理用ソース・ファイル
* scsi.h　：SCSI コマンド処理用ヘッダ・ファイル

第3章　システム構築

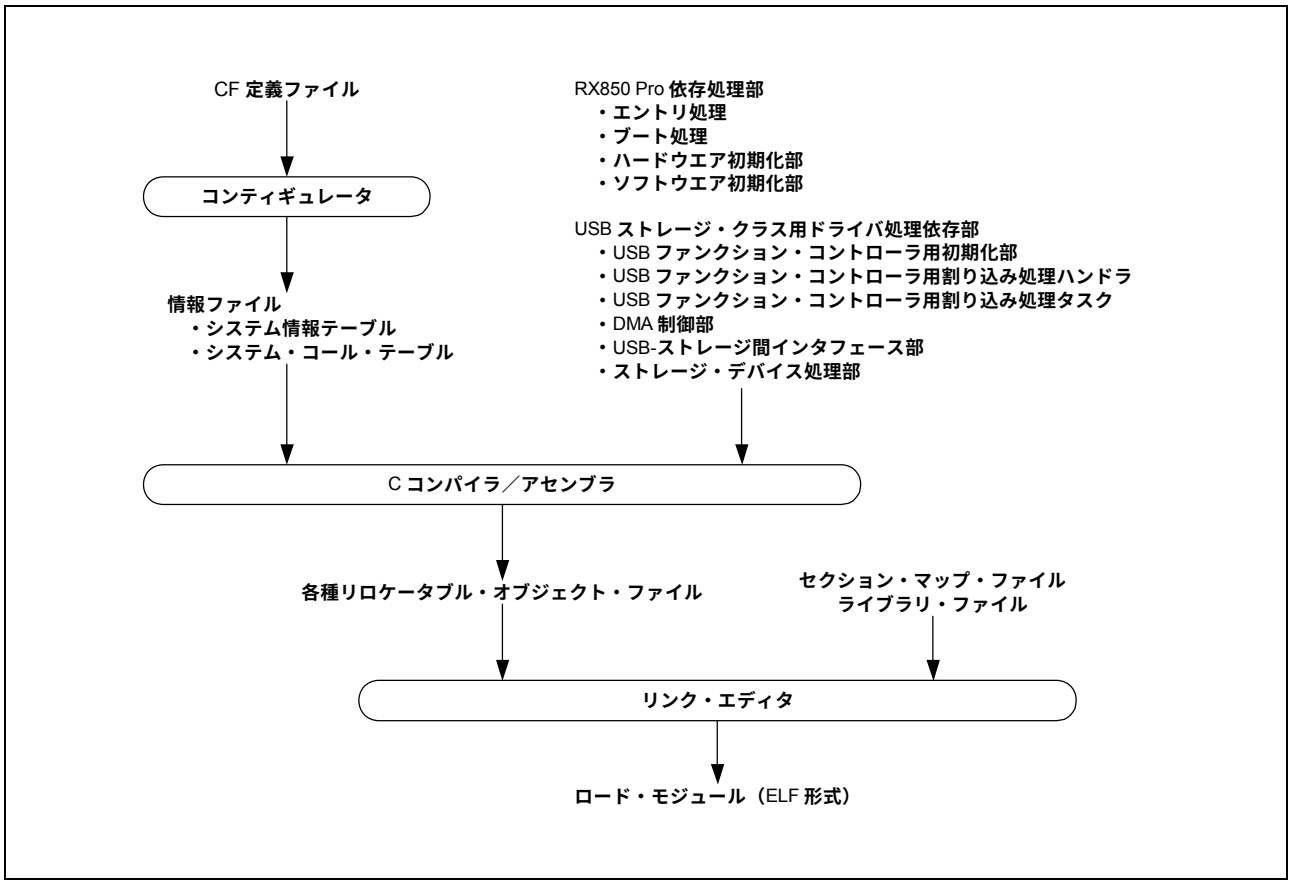
3.1 概　　要

システム構築とは，USB ストレージ・クラス用ドライバ提供媒体からユーザの開発環境（ホスト・マシン）上にインストールしたファイル群を使用して，ロード・モジュールを生成することです。

USB ストレージ・クラス用ドライバのシステム構築手順を次に示します。

① RX850 Pro 依存処理部の記述
② ボード依存部の記述
③ USB ストレージ・クラス用ドライバ処理依存部の記述
④ セクション・マップ・ファイルの記述
⑤ ロード・モジュールの生成

図3－1　システム構築手順



3. 2　RX850 Pro 依存処理部の記述

USB ストレージ・クラス用ドライバが提供する一部の機能は，リアルタイム OS（RX850 Pro）の機能を利用しています。また，ユーザが記述した処理プログラムは，RX850 Pro の管理下で実行することになります。

したがって，RX850 Pro を正常に動作させるために，RX850 Pro 依存処理部の記述が必要となります。

RX850 Pro 依存処理部の一覧を次に示します。

○　CF 定義ファイル
○　エントリ処理
○　システム初期化処理
　・ブート処理
　・ハードウエア初期化部
　・ソフトウエア初期化部

備考　RX850 Pro 依存処理プログラムの詳細は，**第 4 章　RX850 Pro 依存処理プログラム**を参照してください。

3. 3　ボード依存部の記述

USB ストレージ・クラス用ドライバのソース・プログラムは，ユーザの実行環境／アプリケーション・システムに依存した処理に関する初期化処理を，ボード依存部として切り出しています。

ボード依存部の一覧を次に示します。

○　CPU ボード依存部

USB バス・ドライバで必要となる I/O ポート入出力操作については，CPU ボード依存部として切り出しています。

PFESiP EP-1 Evaluation Board では，PFESiP/V850EP1 の P157 からロウ・レベルを出力している場合のみ，VBUSDET による VBUS 検出ができます。そのため，ポート設定用ソース・プログラム（port.c）にて，P157 からロウ・レベルを出力するよう記述しています。

注意　ポートの設定は，ほかのレジスタ設定と同じように扱うため，専用の関数は用意していません。
処理方法は，ブート処理部（boot.s）およびソフトウエア初期化部から呼ばれるポート設定用ソース・プログラム（port.c）を参照してください。
なお，ポートのレジスタは，PFESiP/V850EP1 のデバイス・ファイルにて定義されています。

3.4　USB ストレージ・クラス用ドライバ処理依存部の記述

このサンプル・プログラムでは，USB ストレージ・クラス用ドライバ機能を実現するためのドライバ関数を，USB ストレージ・クラス用ドライバ処理依存部として切り出しています。

USB ストレージ・クラス用ドライバ処理依存部の一覧を次に示します。

- ○ USB ファンクション・コントローラ初期化部
- ○ USB ファンクション・コントローラ割り込みハンドラ
- ○ USB ファンクション・コントローラの割り込み処理タスク
- ○ USB ファンクション・コントローラ用汎用関数
- ○ DMA 制御部
- ○ USB-ストレージ間インタフェース部
- ○ ストレージ・デバイス処理部

備考　USB ストレージ・クラス用ドライバ処理依存部の詳細は，**第7章　USB ストレージ・クラス用ドライバの機能**を参照してください。

3.5　セクション・マップ・ファイルの記述

セクション・マップ・ファイルは，リンク・エディタが行うアドレス割り付けをユーザが固定化するためのファイルです。

RX850 Pro を使用するとき，次の 5 つのテキスト領域が必須のセクションとなっています。

- ○ 共通部分配置領域　：.system セクション
- ○ 割り込み処理関連配置領域　：.system_int セクション
- ○ スケジューラ関連配置領域　：.system_cmn セクション
- ○ システム情報領域　：.sit セクション
- ○ インタフェース・ライブラリ／システム・コール配置領域　：.text セクション

備考　セクション・マップ・ファイルの詳細は，**第5章　リンク・ディレクティブ・ファイル**を参照してください。

3.6　ロード・モジュールの生成

コーディングを終了した RX850 Pro 依存処理プログラム，USB ストレージ・クラス用ドライバ処理依存部，セクション・マップ・ファイルに対して，C コンパイラ，アセンブラ，リンカなどを実行することにより，ELF 形式のロード・モジュールを生成します。

備考　ロード・モジュールの生成手順の詳細は，**第6章　ロード・モジュール**を参照してください。

第4章　RX850 Pro 依存処理プログラム

4.1 概　　要

USB ストレージ・クラス用ドライバが提供する一部の機能は，リアルタイム OS（RX850 Pro）の機能を利用しています。また，ユーザが記述した処理プログラムは，RX850 Pro の管理下で実行することになります。

したがって，RX850 Pro を正常に動作させるために，RX850 Pro 依存処理部の記述が必要となります。

RX850 Pro 依存処理部の一覧を次に示します。

- ○ CF 定義ファイル
- ○ エントリ処理
- ○ システム初期化処理
  - ・ブート処理
  - ・ハードウエア初期化部
  - ・ソフトウエア初期化部

4. 2　CF 定義ファイル

RX850 Pro を使用したシステムを構築する場合，RX850 Pro に提供する各種データを保持した情報ファイル（CF 定義ファイル）が必要となります。

USB ストレージ・クラス用ドライバ機能を実現するうえで必要となる情報を次に示します。

○　リアルタイム OS 情報
　・RX シリーズ情報
○　SIT 情報
　・システム情報
　　・システム最大値情報
　　・システム・メモリ情報
　　・タスク情報
　　・割り込みハンドラ情報
　　・初期化ハンドラ情報
○　SCT 情報
　　・タスク管理／タスク付属同期管理機能システム・コール情報
　　・割り込み処理管理機能システム・コール情報
　　・時間管理機能システム・コール情報

注意　このサンプル・プログラムでは，5 個のタスク，3 個の割り込みハンドラ，7 種類のシステム・コールを使用して，各種機能を実現しています。このため，CF 定義ファイルにおいては，システム最大値情報のタスクの最大生成数に“5 個”を，割り込みハンドラの最大生成数に“3 個”を USB ストレージ・クラス用ドライバのために確保するほか，タスク管理／タスク付属同期管理機能システム・コール情報に“sta_tsk, ext_tsk, slp_tsk, wup_tsk システム・コール”を，割り込み処理管理機能システム・コール情報に“loc_cpu, unl_cpu システム・コール”を，時間管理機能システム・コール情報に“dly_tsk システム・コール”を使用するものとして，定義する必要があります。

備考　CF 定義ファイルのコーディング方法の詳細は，RX850 Pro ユーザーズ・マニュアル インストレーション編，およびサンプルの CF 定義ファイル（sys.cf）を参照してください。

4. 2. 1　情報ファイルの生成手順

情報ファイル（システム情報テーブル，システム・コール・テーブル，システム情報ヘッダ・ファイル）の生成手順を，次に示します。なお，情報ファイルの生成は，Windows のコマンド・プロンプト上で行います。

注意　サンプルのビルド・ファイルを使用する場合，情報ファイルはビルド時に生成されます。このため，次の手順で生成する必要はありません。

① ディレクトリの移動

Windows の cd コマンドを使用して，CF 定義ファイルが格納されているディレクトリに移動します。
なお，CF 定義ファイルが格納されているディレクトリが C:¥sample の場合の入力例を次に示します。

［コマンド入力例］

```
C:>cd C:¥sample¥rx850<Enter>
```

② 情報ファイルの生成

コンフィギュレータ cf850pro.exe を使用して,独自の記述形式で作成された CF 定義ファイルから情報ファイルを生成します。なお，入力ファイル（CF 定義ファイル名：sys.cf）から 3 つの情報ファイル（システム情報テーブル：sit.850，システム・コール・テーブル：svc.850，システム情報ヘッダ・ファイル：sys.h）を生成する場合の入力例を，次に示します。

［コマンド入力例］

```
C:>cf850pro ?i sys.s ?c scy.s ?d sys.h sys.cf<Enter>
```

上記の操作により，CF 定義ファイルから情報ファイルを生成できます。

注意　このサンプル・プログラムでは，情報ファイルを生成するためのサンプル・ファイル（CF 定義ファイル）を提供しています。

備考　コンフィギュレータ cf850pro.exe の起動オプションおよび実行方法の詳細は，RX850Pro ユーザーズ・マニュアル インストレーション編を参照してください。

4. 3　エントリ処理

マスカブル割り込みが発生したときに，プロセッサが強制的に制御を移すハンドラ・アドレスに対して，割り込みハンドラへの分岐処理を割り付けています。

なお，RX850 Pro の管理下で実行される割り込みハンドラ（CF 定義ファイルの割り込みハンドラ情報で定義した割り込みハンドラ）に対応したハンドラ・アドレスに対しては,RX850 Pro が提供するマクロ RTOS_IntEntry_Indirect（RX850 Pro が提供する割り込み処理管理機能への分岐処理）を割り付けてください。

備考　エントリ処理のコーディング方法についての詳細は，添付プログラム entry.s を参照してください。

4.4　システム初期化処理

システム初期化処理は，RX850 Pro が正常に動作するうえで必要となるハードウエアの初期化処理（ブート処理，ハードウエア初期化部），およびソフトウエアの初期化処理（ニュークリアス初期化部，初期化ハンドラ）から構成されています。

したがって，システムが起動したとき，最初に行われる処理がシステム初期化処理となります。

注意　4 種類のシステム初期化処理のうち，ニュークリアス初期化部については，RX850 Pro が提供する機能の一部であるため，ユーザがその処理を記述する必要はありません。

ニュークリアス初期化部で行われる処理を次に示します。

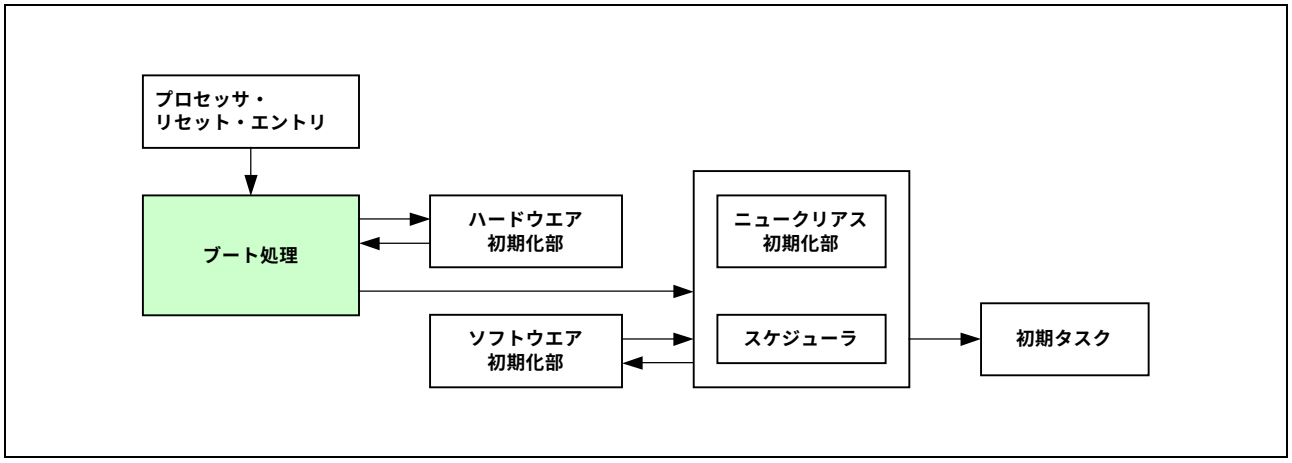
- ○ CF 定義ファイルで定義されたシステム・メモリの確保
  - System Pool 0,1
  - User Pool 0
- ○ CF 定義ファイルで定義された管理オブジェクトの生成／初期化
  - タスクの生成／起動
  - 割り込みハンドラの登録
- ○ 初期タスクの起動
- ○ アイドル・タスクの生成／起動
- ○ ソフトウエア初期化部の呼び出し
- ○ スケジューラに制御を移す

なお，アイドル・タスクとは，RX850 Pro の管理下にある処理プログラム（タスク）が run 状態または ready 状態でなくなったとき，すなわち，RX850 Pro のスケジューリング対象となる処理プログラムがシステム内に 1 つも存在しなくなったときに，スケジューラから起動される処理ルーチンであり，HALT 命令の発行を行っています。

4.4.1　ブート処理

ブート処理は，プロセッサのリセット・エントリに割り付けられた関数であり，システム初期化処理の中でも最初に処理が実行されます。ブート処理の位置付けを次に示します。

図4－1　ブート処理の位置付け



ブート処理で実行すべき処理内容を次に示します。

備考　ブート処理のコーディング方法についての詳細は，サンプルの boot.s を参照してください。

○　tp, gp, ep レジスタの設定

システム起動時，各種処理プログラム（ブート処理を含む）を実行するうえで必要となるテキスト・ポインタ tp，グローバル・ポインタ gp，スタック・ポインタ ep の値は不定です。そこで，ブート処理の最初の処理として，これらのレジスタの初期設定を行います。

注意　この章では，tp に“0”を，gp に“コンパイラが出力するグローバル・ポインタ・シンボル_gp”を，ep に“コンパイラが出力するエレメント・ポインタ・シンボル_ep”を設定することを推奨します。

○　ハードウエア初期化部の呼び出し

ターゲット・システム上のハードウエアを初期化するために用意された関数（ハードウエア初期化部）を呼び出します。なお，ハードウエア初期化部で実行すべき処理内容（内部ユニットの初期化）をほかのところで行う場合は，“ ハードウエア初期化部の呼び出し ” は不要となります。

注意　この章では，ハードウエア初期化部で実行すべき処理内容（内部ユニットの初期化）をソフトウエア初期化部で行うため，“ハードウエア初期化部の呼び出し”は不要です。
詳細は，RX850 Pro ユーザーズ・マニュアル インストレーション編を参照してください。

○　ニュークリアス初期化部への制御の移行

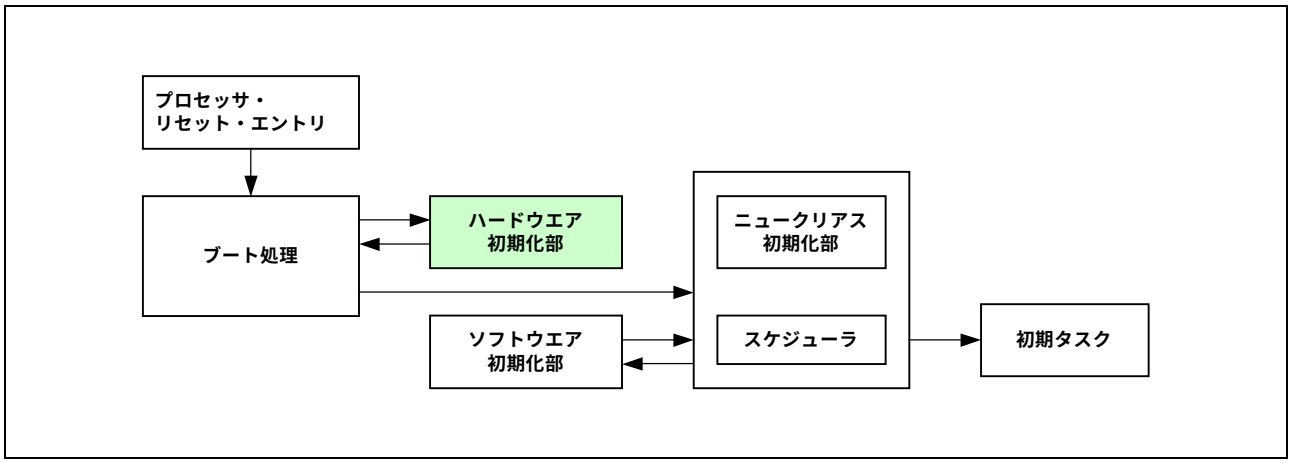
ニュークリアス初期化部では，システム情報テーブルに記述された情報をもとに，システム・メモリ（System Pool 0,1, User Pool 0）の確保，および管理オブジェクトの生成／初期化などを行っています。そこで，ニュークリアス初期化部に制御を移す前には，r10 レジスタにシステム情報テーブルの先頭アドレス_sit を設定しておく必要があります。

注意　システム情報テーブルとは，独自の記述形式で作成された CF 定義ファイルを，RX850 Pro が提供するユーティリティ・ツール（コンフィギュレータ cf850pro.exe）を使用して，アセンブリ言語形式に変換したものです。

4. 4. 2　ハードウエア初期化部

ハードウエア初期化部は，ターゲット・システム上のハードウエアを初期化するために用意された関数であり，ブート処理から呼び出されます。ハードウエア初期化部の位置付けを次に示します。

図4－2　ハードウエア初期化部の位置付け



ハードウエア初期化部で実行すべき処理内容を次に示します。

注意 1. マスカブル割り込みは，初期化時ディフォールトでマスクされていますので，特に禁止設定をする必要はありません。

2. サンプル・プログラムでは，ハードウエアの初期化をソフトウエア初期化部で行っています。
ハードウエア初期化部の詳細は，RX850 Pro ユーザーズ・マニュアル インストレーション編を参照してください。

○ ブート処理に制御を戻す

ブート処理の呼び出し関数であるハードウエア初期化部からブート処理に制御を戻す場合，ブート処理におけるハードウエア初期化部の呼び出し時，lp レジスタに対する戻りアドレスの設定が行われているため，“ return( ); ”命令を発行することにより実現されます。

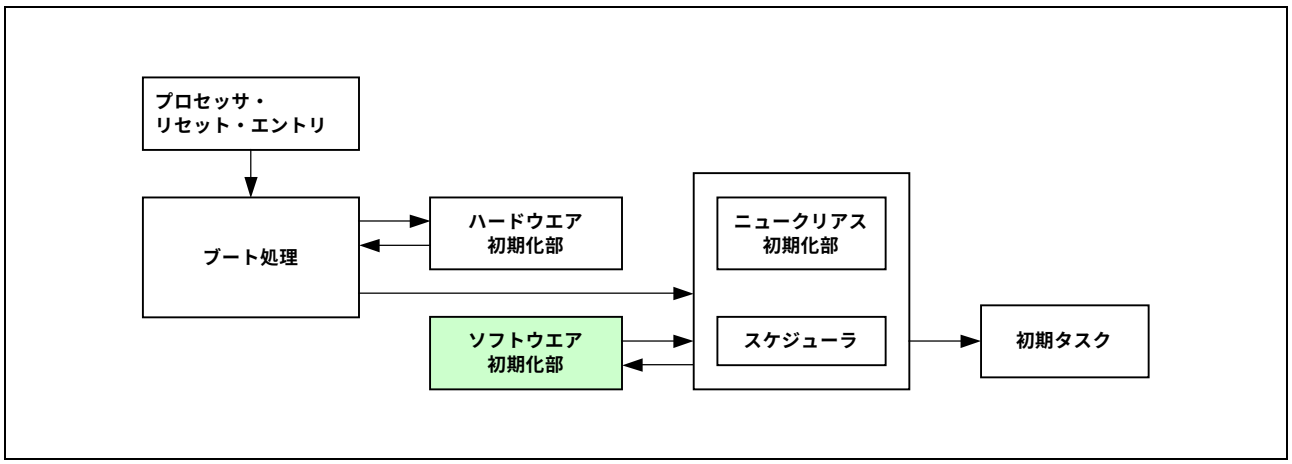
なお，ハードウエア初期化部をアセンブリ言語で記述した場合は，“ jmp [lp] ”命令を発行することにより実現されます。

4.4.3　ソフトウエア初期化部

初期化ハンドラは，ユーザのソフトウエア環境を快適なものとするために用意された関数であり，ニュークリアス初期化部から呼び出されます。

ソフトウエア初期化部の位置付けを次に示します。

図4－3　ソフトウエア初期化部の位置付け



ソフトウエア初期化部で実行すべき処理内容を次に示します。

備考　ソフトウエア初期化部のコーディング方法の詳細は，サンプルの varfunc.c を参照してください。

○　内部ユニット（リアルタイム・パルス・ユニット（RPU））の初期化

RX850 Pro では，一定周期で発生するタイマ割り込みを利用してタイマ・オペレーション機能（タスクの遅延起床，周期起動ハンドラの起動，タイムアウトなど）を実現しています。このため，RX850 Pro が処理を開始する前にリアルタイム・パルス・ユニットを初期化しておく必要があります。

なお，リアルタイム・パルス・ユニットのコンペア・レジスタ CMD0 には，CF 定義ファイルのシステム情報で定義した基本クロック周期でタイマ割り込みが発生するような値を設定する必要があります。

○　タイマ割り込みの受け付け許可

タイマ割り込みの受け付けを許可します。これにより，ニュークリアス初期化部の処理が終了したとき，RX850 Pro が提供するタイマ・オペレーション機能（タスクの遅延起床，タイムアウト，周期起動ハンドラの起動など）の利用が可能となります。

○　ニュークリアス初期化部に制御を戻す

ニュークリアス初期化部の呼び出し関数である初期化ハンドラからニュークリアス初期化部に制御を戻す場合，ニュークリアス初期化部における初期化ハンドラの呼び出し時に，lp レジスタに対する戻りアドレスの設定が行われているため，“ return( ); ”命令を発行することにより実現されます。

なお，初期化ハンドラをアセンブリ言語で記述した場合は，“ jmp [lp] ”命令を発行することにより実現されます。

4.5　時間管理機能

RX850 Pro における時間管理は，ハードウエア（クロック・コントローラなど）により一定周期で発生するクロック割り込みを利用しています。

クロック割り込みが発生すると，RX850 Pro のシステム・クロック処理が呼び出され，システム・クロックの更新やタスクの遅延起床，周期ハンドラの起動などの，時間に関連した処理が行われます。

システム・クロックは，RX850 Pro が時間管理のときに使用する時刻（48 ビット幅，単位：ms）を保持したソフトウエア・タイマです。

システム・クロックは，システム初期化処理で“ 0H ”に設定されたあと，システム・クロック処理で基本クロック周期（コンフィギュレーション時に指定）を単位として更新されます。

注意　RX850 Pro が管理するシステム・クロックは，48 ビット幅で構成されています。このため，RX850 Pro では，オーバフローした数値（48 ビットでは表せない数値）は無視されます。RX850 Pro の時間管理機能の詳細は，RX850 Pro ユーザーズ・マニュアル 基礎編を参照してください。

第5章　リンク・ディレクティブ・ファイル

5.1 概　　要

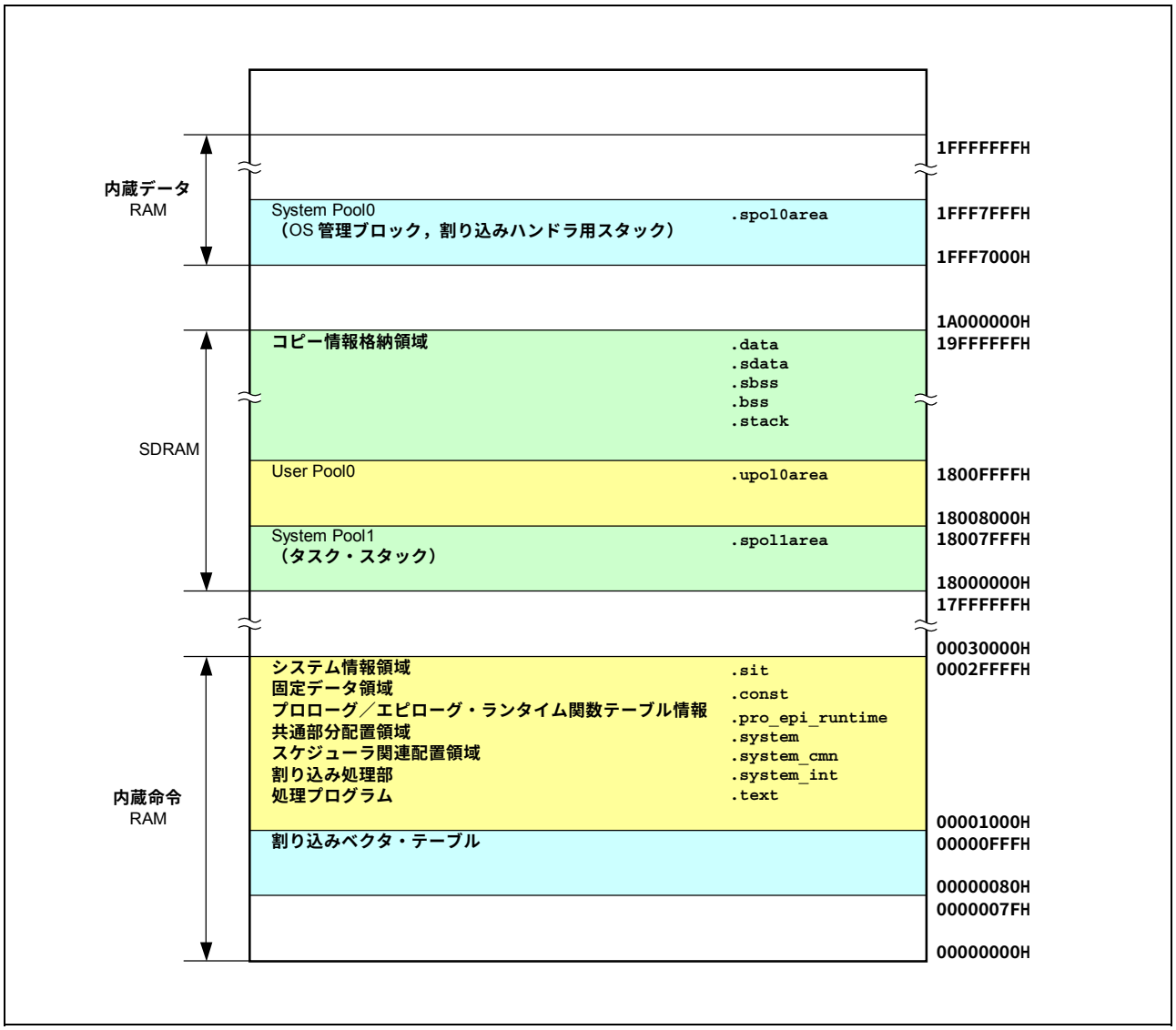
リンク・ディレクティブ・ファイルとは，リンク・エディタが行うアドレス割り付けを，ユーザが固定化するためのファイルです。

5.2 RX850 Pro のアドレス割り付け，5.3 その他のアドレス割り付けに，ユーザの処理プログラム（.data, .bss セクションなど）以外に必要となるアドレス割付けについて示します。

サンプルの usb_storage.dir で行われているアドレス割り付けを次に示します。

備考　リンク・ディレクティブ・ファイルのコーディング方法の詳細は，サンプルの usb_storage.dir を参照してください。

図5－1　アドレス割り付け例



5. 2　RX850 Pro のアドレス割り付け

RX850 Pro は，5 つのテキスト領域（共通部分配置領域，割り込み処理関連配置領域，スケジューラ関連配置領域，システム情報領域，インタフェース・ライブラリ／システム・コール配置領域）から構成されています。これにより，大きなサイズが要求されるメモリ領域については外部 RAM に，高速なアクセスが要求されるメモリ領域（割り込み処理部，スケジューリング処理部）については内蔵命令 RAM（00000000H-0002FFFFH）に割り付けるといったことが可能となります。

注意　サンプル・プログラムでは，5 つのテキスト領域のすべてを内蔵命令 RAM に配置しています。

○　共通部分配置領域（.system セクション）

RX850 Pro の本体処理（タスク管理機能，タスク付属同期機能など）が割り付けられる領域です。

○　割り込み処理関連配置領域（.system_int セクション）

RX850 Pro が提供する割り込み処理管理機能のうち，割り込みハンドラに制御を移すときに行われる割り込み前処理，およびマスカブル割り込みが発生した処理プログラムに制御を戻すときに行われる割り込み後処理から構成されています。

したがって，割り込み処理部を内蔵命令 RAM に割り付けることにより，割り込みハンドラに対する応答性能が向上します。

注意　この章では，割り込み処理部を内蔵命令 RAM に割り付けることを推奨します。

○　スケジューラ関連配置領域（.system_cmn セクション）

RX850 Pro が提供するスケジューリング機能のうち，タスクの起床処理およびタスクのスケジューリング処理から構成されています。

したがって，スケジューリング処理部を内蔵命令 RAM に割り付けることにより，タスクの起床処理およびタスクのスケジューリング処理が高速化されるほか，スケジューリング処理を伴ったシステム・コールの処理も高速化されます。

注意　この章では，スケジューリング処理部を内蔵命令 RAM に割り付けることを推奨します。

○　システム情報領域（.sit セクション）

CF 定義ファイルに対してコンフィギュレータ cf850.exe を実行することにより生成されたシステム情報テーブルが割り付けられる領域です。

なお，システム情報テーブルは，ニュークリアス初期化部（システム・メモリの確保，管理オブジェクトの生成／初期化）を実行するうえで必要となる各種データから構成されています。

○　インタフェース・ライブラリ／システム・コール配置領域（.text セクション）

システム・コールを含む命令が割り付けられる領域です。

○ システム・メモリ

RX850 Pro が提供する機能を実現するうえで必要となる各種管理ブロック（タスク管理ブロック，セマフォ管理ブロックなど）および割り込みハンドラ用スタックが割り付けられる領域（System Pool 0），タスク用スタックが割り付けられる領域（System Pool 1），および処理プログラムからダイナミックなメモリ操作（メモリ・ブロックの獲得／開放）を可能とした領域（User Pool 0）から構成されています。

注意 1. CF 定義ファイル作成時，“システム・メモリの先頭アドレス”を指定する必要があります。
したがって，セクション・マップ・ファイルにシステム・メモリの定義を行うときには，必ずアドレスを指定してください。

2. システム・メモリのセクション名については，ユーザが自由に指定できます。

5.3 その他のアドレス割り付け

次に，RX850 Pro 以外にアドレス割り付けが必要となるセクションについて示します。

○ CA850 予約領域（.pro_epi_runtime セクション）

CA850 にてプロローグ／エピローグ・ランタイム関数のテーブルを配置する領域です。

備考　.pro_epi_runtime セクションの詳細は，CA850 のマニュアルを参照してください。

第6章　ロード・モジュール

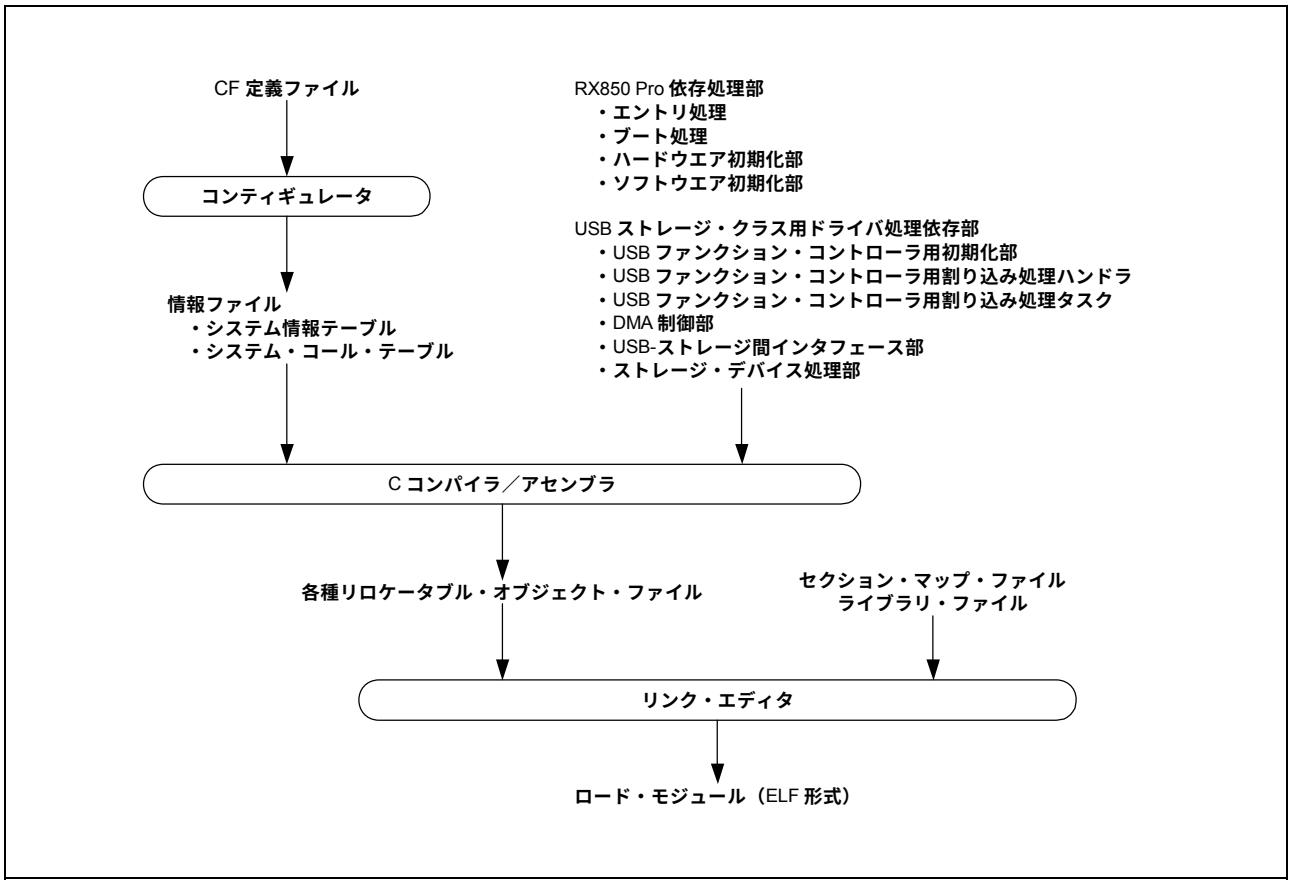
6.1 概　　要

コーディングを終了したRX850 Pro依存処理プログラム，USBストレージ・クラス用ドライバ処理依存部，セクション・マップ・ファイルに対して，Cコンパイラ，アセンブラ，リンカなどを実行することにより，ELF形式のロード・モジュールを生成します。

ロード・モジュールの生成手順を次に示します。

注意　サンプルの.prjファイルを実行することにより，サンプル・プログラムに対応したロード・モジュールを生成できます。ただし，.prjファイルのパスの定義については，ユーザの開発環境にあわせた修正が必要です。

図6－1　ロード・モジュールの生成手順



6.2 ロード・モジュールの生成

コーディングを終了したRX850 Pro依存処理プログラム，USBストレージ・クラス用ドライバ処理依存部，セクション・マップ・ファイルは，次の手順によりELF形式のロード・モジュールとなります。

① ステム情報テーブル，システム・コール・テーブルの生成
独自の記述形式で作成されたCF定義ファイルは，ロード・モジュール作成時，リンク・エディタが行うリンク処理の対象外のファイル形式です。
そこで，RX850 Proが提供するユーティリティ・ツール（コンフィギュレータcf850.exe）を使用して，アセンブル可能なファイル（システム情報テーブル，システム・コール・テーブル）を生成します。

備考　システム情報テーブル，システム・コール・テーブルの生成方法についての詳細は，4. 2. 1　**情報ファイルの生成手順**を参照してください。

② オブジェクト・ファイルの生成
次に示す各種処理プログラム（C言語／アセンブリ言語形式のファイル）に対して，Cコンパイラ／アセンブラを実行することにより，リロケータブル・オブジェクト・ファイルを生成します。

○ RX850 Pro依存処理プログラム
- システム情報テーブル
- システム・コール・テーブル
- エントリ処理
- ブート処理
- ハードウエア初期化部
- 初期化ハンドラ

○ USBストレージ・クラス用ドライバ処理依存部

③ ロード・モジュールの生成
②で生成したリロケータブル・オブジェクト・ファイル，および，各種ライブラリ・ファイル，セクション・マップ・ファイルに対して，リンク・エディタを実行することにより，ELF形式のロード・モジュールを生成します。

libc.a	CA850標準ライブラリ
rxtmcore.o	ニュークリアス共通部分オブジェクト
librxp.a	ニュークリアス・ライブラリ
libchp.a	インタフェース・ライブラリ

なお，rxtmcore.o, librxp.a, libchp.aは“RX850 Pro”から，libc.aは“CA850”から提供されています。

第7章 USB ストレージ・クラス用ドライバの機能

7.1 概　　要

USB ストレージ・クラス用ドライバでは，USB ストレージ・クラス用ドライバ処理を実現するためのタスク，割り込みハンドラのほかに，USB ファンクション・コントローラの初期化処理の記述が必要となります。

USB ストレージ・クラス用ドライバ処理依存部の一覧を次に示します。

・USB ファンクション・コントローラの初期化処理

RX850 Pro のソフトウエア初期化部から呼び出され，USB ファンクション・コントローラの初期化処理を行います。

○ USB ファンクション・コントローラの割り込みハンドラ

USB ファンクション・コントローラの割り込み発生ごとに呼び出される割り込み処理専用ルーチンであり，CF 定義ファイルに定義されています。

注意　このサンプル・プログラムでは，必要な割り込み以外はマスクされています。
このサンプル・プログラムで使用する割り込みは，次の 4 つです。

- INTUSBF0 信号で通知される CPUDEC 割り込み
  （UF0E0ST レジスタに FW でデコードを行うリクエストがあることを示します）
- INTUSBF0 信号で通知される BKO1DT 割り込み
  （UF0BO1 レジスタにデータが正常受信されたことを示します）
- INTUSBF1 信号で通知される EP1_ENDINT 割り込み
  （エンドポイント 1 の DMA 転送が終了したことを示します）
- INTUSBF2 信号で通知される EP2_ENDINT 割り込み
  （エンドポイント 2 の DMA 転送が終了したことを示します）

○ USB ファンクション・コントローラの割り込み処理タスク

USB ファンクション・コントローラの割り込みハンドラから呼び出され，割り込み要因ごとの処理（レジスタ設定，データの送信／受信処理など）を行います。

○ USB ファンクション・コントローラ用汎用関数

USB ストレージ・クラス用ドライバで使用される汎用関数として，エンドポイントごとの STALL 応答の設定や，送信，受信の処理を行う関数が用意されています。

備考　USB ストレージ・クラス用ドライバ処理依存部のコーディング方法の詳細は，サンプルの usbf850.c を参照してください。

○　DMA 制御部

DMA の初期化および DMA の起動処理を行います。

サンプル・プログラムでは，バルク・エンドポイントのデータが MaxPacket サイズ（40 バイト）を越える場合，DMA 転送が使用されます。

備考　DMA 制御部のコーディング方法の詳細は，サンプルの usbf850_dma.c を参照してください。

○　Bulk-Only Transport 処理部

USB ストレージ・クラス用のデバイス・クラス固有のリクエスト処理，CBW 処理，および CSW の送信処理を行います。

注意　このサンプル・プログラムで受け付けるデバイス・クラス固有のリクエストは，次の 2 つです。
各リクエストの詳細は，Universal Serial Bus Mass Starage Class Bulk-Only Transport Revision1.0 を参照してください。

- Bulk-Only Mass Storage Reset リクエスト
- Get Max LUN リクエスト

備考　Bulk-Only Transport 処理部のコーディング方法の詳細は，サンプルの usbf850_storage.c を参照してください。

○　ストレージ・デバイス処理部

ストレージ・デバイスの初期化，SCSI コマンドの処理を行います。

注意 1.　このサンプル・プログラムは，Mass Storage デバイス（インタフェース・クラス：マス・ストレージ，インタフェース・サブクラス：SCSI，インタフェース・プロトコル：Bulk-OnlyTransport プロトコル）として動作します。今回，使用するストレージ・デバイスは，結合される論理ユニットはなく，メモリ領域を確保し疑似的にリムーバブル・ディスクが繋がれているように動作するものです（ブロック・サイズ：512 バイト，論理ブロック数：192，容量：96 K バイト）。仮想デバイス用のメモリ領域は，storage_data という名前の配列で確保されています。詳細は，サンプルの ata.h を参照してください。

2.　仮想デバイスを変更し実際のデバイスを制御する場合は，サンプル・プログラム内で使用するデータやデータ処理について変更が必要です。環境にあわせて変更してください。

備考　ストレージ・デバイス処理部のコーディング方法の詳細は，サンプルの ata_ctrl.c および scsi_cmd.c を参照してください。

○　USB のサスペンド／レジュームの処理

USB のサスペンド／レジュームの処理はシステムに依存するため，このサンプル・プログラムではサポートしていません。システム上，処理が必要な場合は，次のことに注意して処理を追加してください。

PFESiP/V850EP1 に内蔵している USB ファンクション・コントローラでは，サスペンド／レジュームが割り込み（INTUSBF0 信号）により通知されます。このため，割り込みハンドラ（INTUSBF0 信号用）内で UF0IS0.RSUSPD ビットを確認し，セット（1）されていれば，UF0EPS1.RSUM ビットを確認してサスペンド状態なのかレジューム状態なのかを判断できます。

処理を追加する一例として，割り込みハンドラ（INTUSBF0 信号用）に上記の状態判断のコードを追加し，そこから必要な処理を行うタスクを起床するようにすることで実現できます。

7. 2　処理の流れ

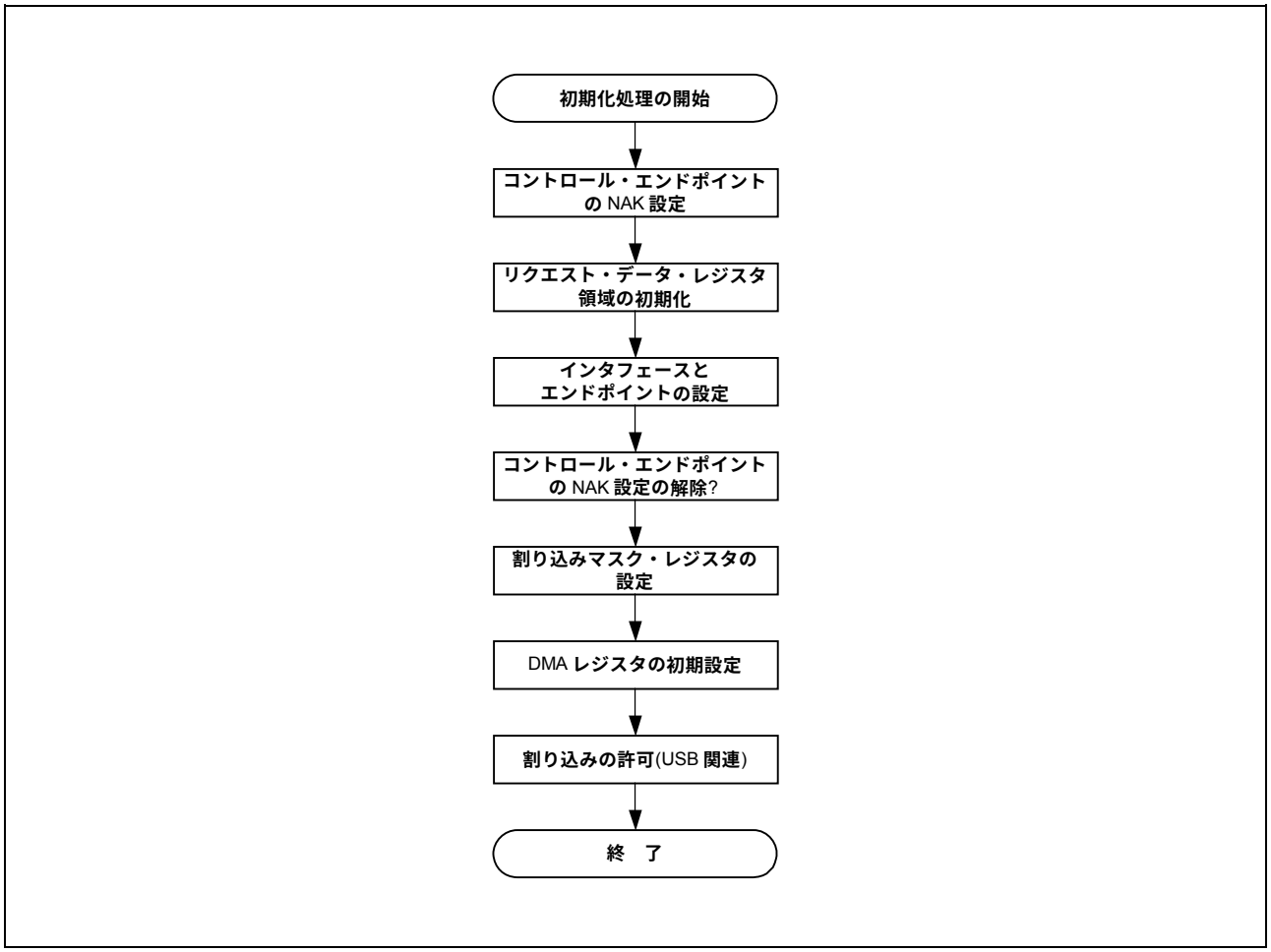
サンプル・プログラムの初期化処理，割り込み処理，CBW データの処理についての処理の流れを，次に示します。

7. 2. 1　初期化処理

USB デバイスの初期化は，ソフトウエア初期化部から呼び出され実行されます。

サンプル・プログラムにおける USB デバイス初期化処理（電源投入時）の流れを，次に示します。

図7－1　初期化処理のフロー・チャート



初期化処理で実行すべき処理内容を次に示します。

注意　ポートの処理以外では，初期化処理が必要です。ターゲット・ボードが違う場合，端子の割り付けが違うことがありますので，ターゲット・ボードの仕様にあわせて読み替えてください。

○　コントロール・エンドポイントの NAK 設定

自動実行リクエストを含むすべてのリクエストに NAK 応答します。

自動実行リクエストで使用するデータの登録が完了するまで，ハードウエアが自動実行リクエストに意図しないデータを返さないように設定します。

○　リクエスト・データ・レジスタ領域の初期化

Get Descriptor リクエストに応答するためのディスクリプタ・データなどをレジスタに登録します。

登録するデータは，デバイス・ステータス，エンドポイント 0 ステータス，Device Descriptor,Configuration Descriptor, Interface Descriptor, Endpoint Descriptor です。

注意　クラスによっては，そのクラス用のディスクリプタの登録が必要な場合があります。

USB ストレージ・クラスでは，USB の標準ディスクリプタ以外は使用しません。

○　インタフェースとエンドポイントの設定

サポートするインタフェースの数，Alternative 設定の状態，インタフェースとエンドポイントの関係などの情報を，レジスタに設定します。

○　コントロール・エンドポイントの NAK 設定の解除

自動実行リクエスト用のデータ登録が終わったところで，コントロール・エンドポイント（エンドポイント番号 0）の NAK 設定を解除します。

○　割り込みマスク・レジスタの設定

USB ファンクション・コントローラの割り込みステータス・レジスタに示される割り込み要因ごとのマスクを設定します。

○　DMA レジスタの初期設定

DMA を使用するエンドポイントごとに，DMA の初期化処理を呼び出し，初期化を行います。

○　割り込みの許可

INTUSBF0，INTUSBF1，INTUSBF2 の割り込みを許可します。

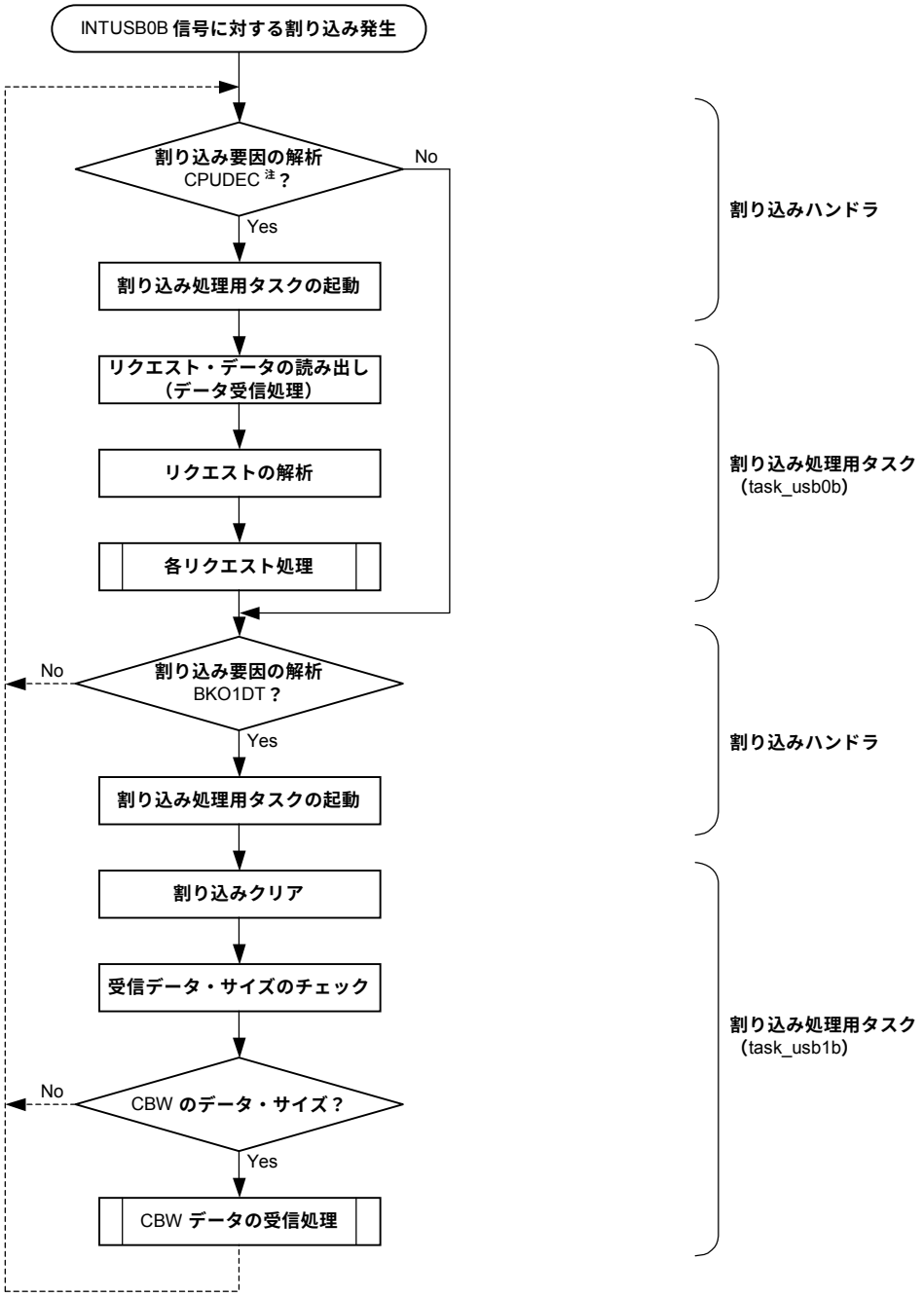
7. 2. 2　割り込み処理

サンプル・プログラムでは，初期化完了後，割り込みイベントによって動作します。イベントがない場合は，常にアイドル状態です。また，ストレージ・デバイスからイベントを起こすことはなく，すべてホスト・ドライバからのイベントにより動作します。

サンプル・プログラムでの割り込み処理の流れを，図 7－2，図 7－3，図 7－4 に示します。

注意　図 7－2 に示すフロー・チャートは，USB ファンクション・コントローラの INTUSBF0 信号により通知される割り込み処理の流れを示しています。
図 7－3 に示すフロー・チャートは，USB ファンクション・コントローラの INTUSBF1 信号により通知される割り込み処理の流れを示しています。
図 7－4 に示すフロー・チャートは，USB ファンクション・コントローラの INTUSBF2 信号により通知される割り込み処理の流れを示しています。

図7－2　割り込み処理のフロー・チャート（1）



注 USB ストレージ・クラス用ドライバの INTUSBF0 信号では，CPUDEC, BKO1DT 割り込みを受け付けます

INTUSBF0 信号による割り込みでのサンプル・プログラムの処理内容を，次に示します。

[割り込みハンドラ内での処理]

○　割り込み要因の解析
サンプル・プログラムでは，実行される割り込みハンドラにより，解析する割り込みステータスが違います。
INTUSBF0 信号で通知される割り込みについては，CPUDEC，BKO1DT 割り込みに対応しています。これらの割り込みが発生すると，INTUSBF0 信号による割り込みハンドラが起動します。この割り込みハンドラの中で，UF0IS1 レジスタを読み，割り込み要因が CPUDEC 割り込みかどうかを確認します。さらに，UF0IS3 レジスタを読み，BKO1DT 割り込みかどうかを確認します。

注意　このサンプル・プログラムでは，使用する割り込みハンドラを CF 定義ファイルであらかじめ登録してあります。

○　割り込み処理用のタスクの起動
割り込み要因が CPUDEC だった場合，task_usb0b タスクを起動します。

注意　このサンプル・プログラムでは，起動するタスクを CF 定義ファイルであらかじめ登録してあります。

○　割り込み処理用のタスクの起動
割り込み要因が BKO1DT だった場合，task_usb1b タスクを起動します。

注意　このサンプル・プログラムでは，起動するタスクを CF 定義ファイルであらかじめ登録してあります。

[task_usb0b タスクでの処理]

○　リクエスト・データの読み出し
UF0E0ST レジスタから SETUP データを読み出します。

○　リクエストの解析
読み出した SETUP データを解析し，リクエスト内容を確認します。

○　各リクエスト処理
解析したリクエスト内容に対応した処理を行います。
サンプル・プログラムでは，標準デバイス・リクエストの Get Descriptor（String Descriptor）リクエスト，およびデバイス・クラス固有のリクエストの処理を行います。

[task_usb1b タスクでの処理]

○ 割り込み要因の解析

INTUSBF1 信号で通知される割り込みについては，BKO1DT 割り込みだけに対応しています。割り込みが発生すると，INTUSBF1 信号による割り込みハンドラが起動します。

割り込みハンドラ内では割り込み要因を確認せず，起動されたタスクで割り込み要因が BKO1DT であるかを確認します。

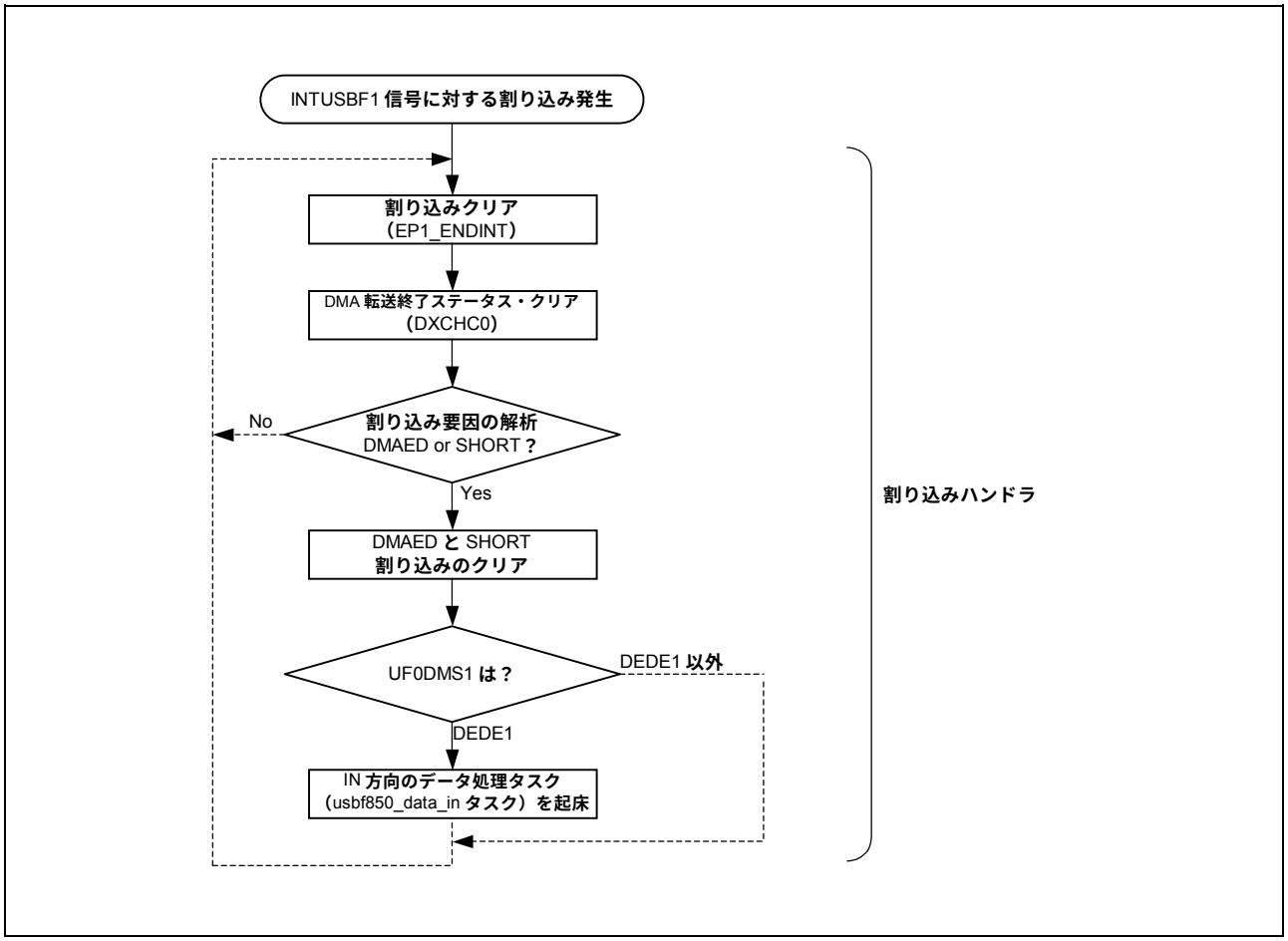
注意　このサンプル・プログラムでは，使用する割り込みハンドラを CF 定義ファイルであらかじめ登録してあります。

○ 受信データ・サイズのチェック

割り込み要因が BKO1DT であれば，UF0BO1L レジスタを読み出し，受信データ長が CBW のデータ長と等しいかを確認します。CBW のデータ長と等しければ，usbf850_rx_cbw 関数を呼び出し，CBW の処理を開始します。

備考　CBW の処理については，7. 2. 3　CBW データの処理を参照してください。

図7－3　割り込み処理のフロー・チャート（2）



INTUSBF1 信号による割り込みでのサンプル・プログラムの処理内容を，次に示します。

○　エンドポイント 1 の DMA 転送終了割り込み要因クリア

USB ファンクション Bridge の割り込み要因をクリアします。

○　DMA 転送終了ステータス・クリア

DMA コントローラの DMA 転送終了ステータスをクリアします。

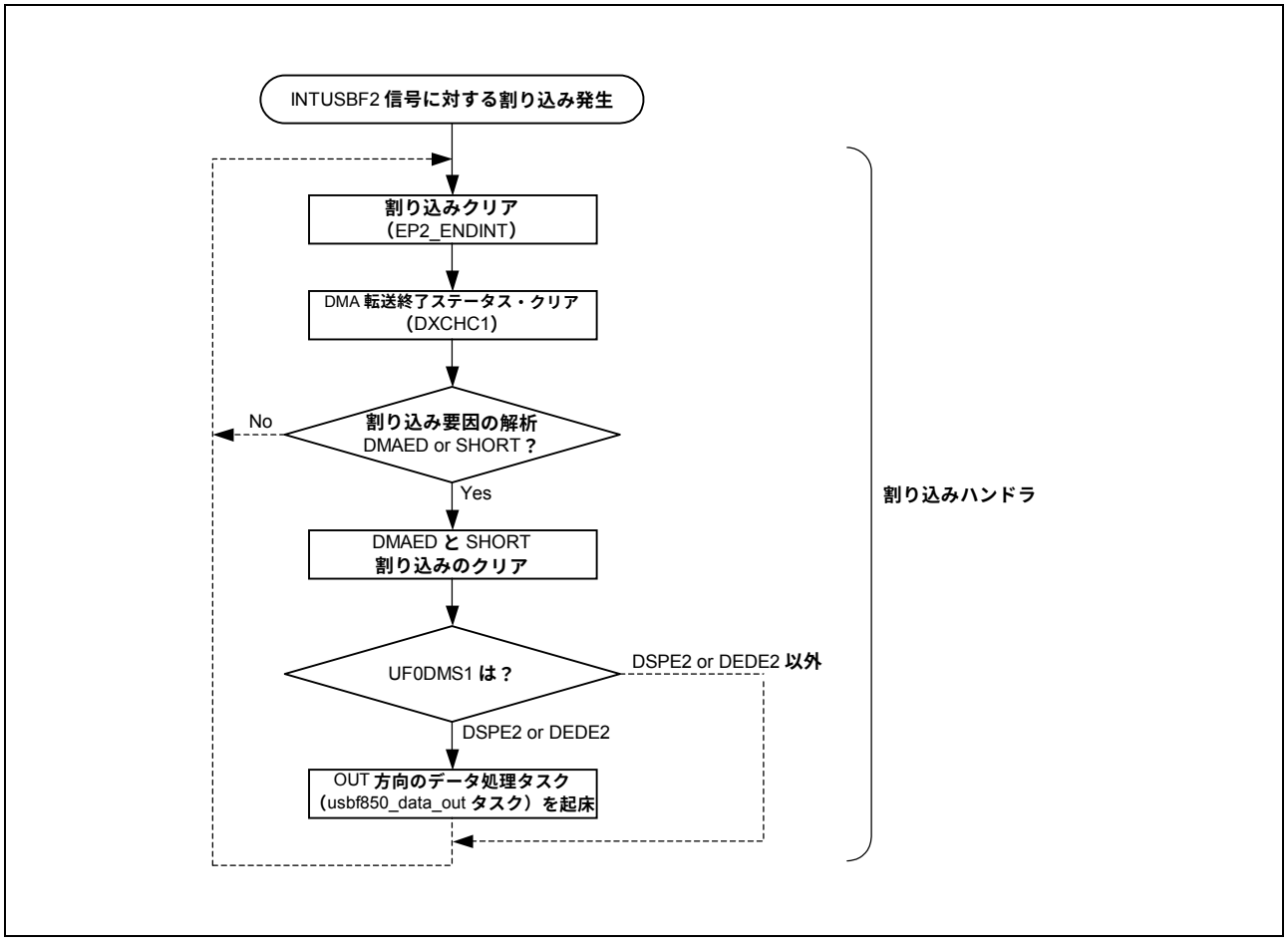
○　usbf850_data_in タスクの起床

usbf850_no_data タスク，usbf850_data_in タスク，および usbf850_data_out タスクは，SCSI コマンド処理を行うタスクです。これらのタスクは，BKO1DT 割り込みで正常な CBW を受信することで起動されます。このうち，usbf850_data_in タスクおよび usbf850_data_out タスクは，DMA 起動後スリープします。

INTUSBF1 信号用の割り込みハンドラでは，割り込み要因が DMAED または SHORT だった場合，UF0DMS1 レジスタを読みだし，DEDE1 であるかどうかを確認します。

DEDE1 だった場合は usbf850_data_in タスクを起床させます。

図7－4　割り込み処理のフロー・チャート（3）



INTUSBF2 信号による割り込みでのサンプル・プログラムの処理内容を，次に示します。

○　エンドポイント 2 の DMA 転送終了割り込み要因クリア

USB ファンクション Bridge の割り込み要因をクリアします。

○　DMA 転送終了ステータス・クリア

DMA コントローラの DMA 転送終了ステータスをクリアします。

○ usbf850_data_ out タスクの起床

usbf850_no_data タスク，usbf850_data_in タスク，および usbf850_data_out タスクは，SCSI コマンド処理を行うタスクです。これらのタスクは，BKO1DT 割り込みで正常な CBW を受信することで起動されます。このうち，usbf850_data_in タスクおよび usbf850_data_out タスクは，DMA 起動後スリープします。

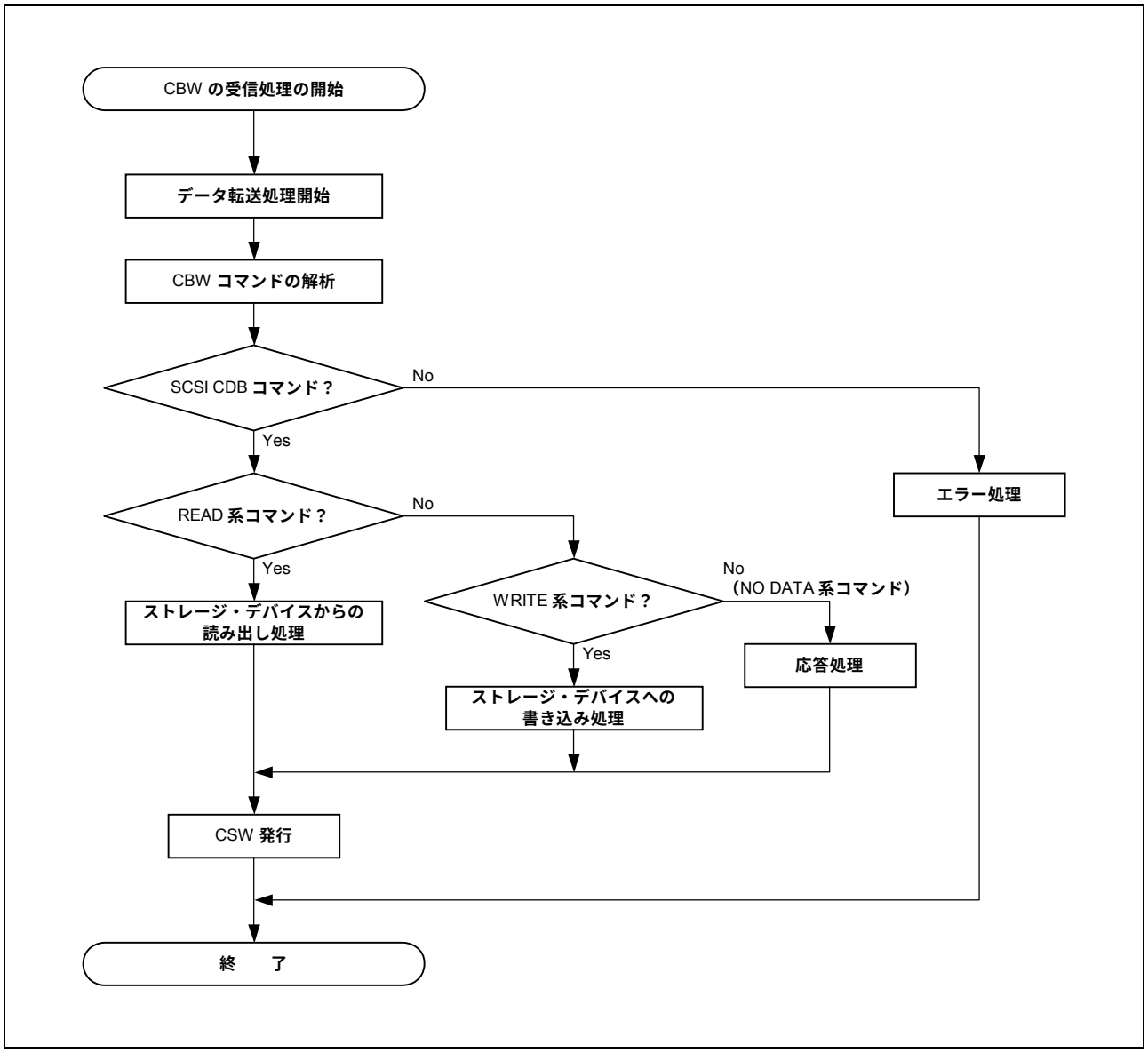
INTUSBF2信号用の割り込みハンドラでは,割り込み要因がDMAEDまたはSHORTだった場合,UF0DMS1レジスタを読みだし，DSPE2, DEDE2 であるかどうかを確認します。

DSPE2 または DEDE2 だった場合は usbf850_data_out タスクを起床させます。

7. 2. 3　CBW データの処理

CBW データの処理は，USB で CBW データを受信すると開始されます。CBW データの処理の流れを次に示します。また，表 7－1 に CBW のデータ・フォーマットを，表 7－2 に CSW のデータ・フォーマットを示します。

図7－5　CBWデータの処理のフロー・チャート



サンプル・プログラムでの CBW データの処理内容を，次に示します。

○　CBW コマンドの解析

CBW データの受信後，CBW の内容について解析を行います。

ここでは，CBW のタグを保存し，CBWCB の有効データ数，コマンドの方向について確認を行い，READ 系，WRITE 系，NO DATA 系のそれぞれの処理タスクを起動します。

○　READ 系コマンドの処理

SCSI コマンドの READ 系コマンド処理を行うタスク（usbf850_data_in）を起動します。

このタスクでは，SCSI コマンド処理部を呼び出し，実行結果から CSW の送信ステータスを判定し，CSW 発行処理を呼び出します。

○　WRITE 系コマンドの処理

SCSI コマンドの WRITE 系コマンド処理を行うタスク（usbf850_data_out）を起動します。

このタスクでは，SCSI コマンド処理部を呼び出し，実行結果から CSW の送信ステータスを判定し，CSW 発行処理を呼び出します。

○　NO DATA 系コマンドの処理

SCSI コマンドの NO DATA 系コマンド処理を行うタスク（usbf850_no_data）を起動します。

このタスクでは，SCSI コマンド処理部を呼び出し，実行結果から CSW の送信ステータスを判定し，CSW 発行処理を呼び出します。

○　CSW 発行

各コマンド処理タスクからコマンド実行結果を引数として呼び出されます。

引数から CSW データを生成し，送信処理を行います。

[CBW の形式]

CBW は次に示す 31 バイトのデータで構成されています。

CBWCB の値によりホストからの処理内容を判断できます。

表 7－1　CBW のデータ・フォーマット

ビット / バイト	7	6	5	4	3	2	1	0
0	dCBWSignature（55H）							
1	dCBWSignature（53H）							
2	dCBWSignature（42H）							
3	dCBWSignature（43H）							
4-7	dCBWTag（処理対象となった CBW のタグ）							
8-11	dCBWDataTransferLength（転送データ長）							
12	bmCBWFlag（Data-OUT/IN の指定）							
13	Reserved				bCBWLUN（対象デバイスの番号）			
14	Reserved			bCBWCBLength（CBWCB の有効バイト数）				
15-30	CBWCB（コマンド）							

[CSW の形式]

CSW は次に示す 13 バイトのデータで構成されています。

表 7－2　CSW のデータ・フォーマット

ビット / バイト	7	6	5	4	3	2	1	0
0	dCSWSignature（53H）							
1	dCSWSignature（42H）							
2	dCSWSignature（53H）							
3	dCSWSignature（55H）							
4-7	dCSWTag（処理対象となった CBW のタグ）							
8-11	dCSWDataResidue（CBW 指定の転送データ長と処理したデータ長の差）							
12	bmCSWStatus（CBW 処理結果のステータス）							

7. 2. 4　SCSI コマンドの処理

SCSI コマンドの処理は，USB で CBW データの受信処理後開始されます。

サンプル・プログラムでは，表 7－3 に示す 19 種類の SCSI CDB コマンドに対応しています。

また，図 7－6 に，SCSI コマンド（READ 系コマンド）処理の流れを示します。

注意　サンプル・プログラムで用意しているコマンドは，サンプル・プログラムの動作に必要な最低限のコマンドだけです。ユーザの環境によっては，サンプル・プログラムにないコマンド，および応答データの生成や処理方法について，必要な処理を追加してください。

表 7－3　SCSI コマンド一覧

Command	Code	動　作
READ 系		
REQUEST SENSE	03H	SENSE データをホストに転送します。
READ（6）	08H	指定された範囲の論理データ・ブロックのデータをホストに転送します。
INQUIRY	12H	ターゲットとロジカル・ユニットについての構成情報や属性をホストに通知します。
MODE SENSE（6）	1AH	ロジカル・ユニットのモード・セレクト・パラメータの値や属性を読み出します。
READ FORMAT CAPACITIES	23H	ロジカル・ユニットの容量（ブロック数，ブロック長）をホストに通知します。
READ CAPACITY	25H	ロジカル・ユニット上のデータ容量をホストに通知します。
READ（10）	28H	READ（6）と同じです。
MODE SENSE（10）	5AH	MODE SENSE（6）と同じです。
WRITE 系		
WRITE（6）	0AH	ホストからのデータを媒体上の指定されたブロックに書き込みます。
MODE SELECT（6）	15H	ロジカル・ユニットのデータ形式などの各種パラメータの設定や変更を行います。
WRITE（10）	2AH	WRITE（6）と同じです。
WRITE VERIFY	2EH	データを媒体に書き込んだあと，読み出して正常性の確認を行います。
VERIFY	2FH	ドライブ・ユニットの媒体上のデータの正常性確認を行います。
WRITE_ BUFF	3BH	ターゲットのメモリに任意のデータを書き込みます。
MODE_SELECT（10）	55H	MODE SELECT（6）と同じです。
NO DATA 系		
TEST UNIT READY	00H	ロジカル・ユニットの状態をイニシエータ（ホスト・デバイス）に通知します。
SEEK	0BH	指定された記録媒体上の位置へのシーク動作を行います。
START STOP UNIT	1BH	ロジカル・ユニットの媒体へのアクセスを可能にしたり不可能にしたりします。
SYNCHRONIZE CACHE	35H	指定範囲のブロックについて，キャッシュ・メモリと媒体の値を一致させます。

図7－6　READ系コマンドの処理のフロー・チャート（1/2）

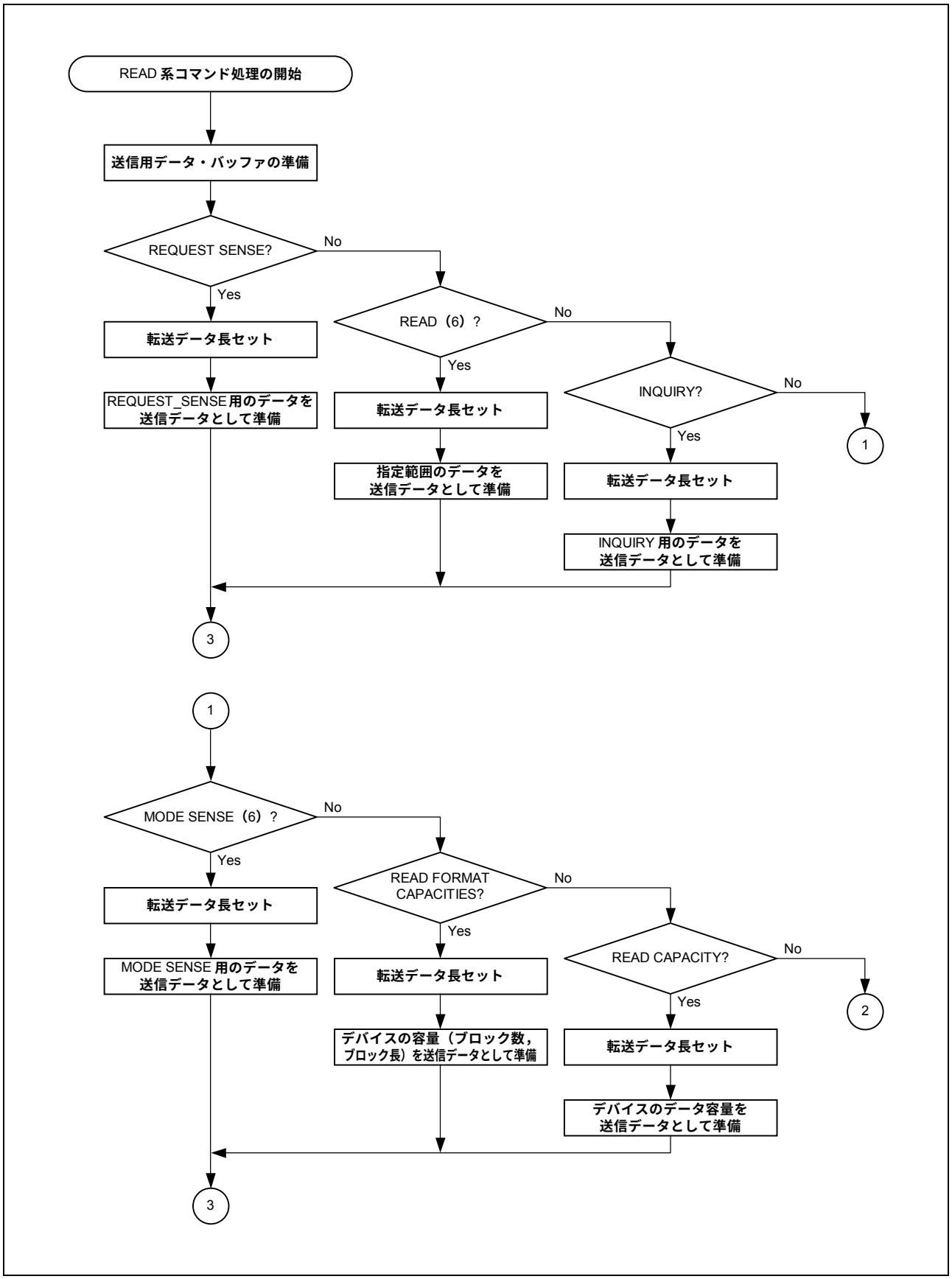
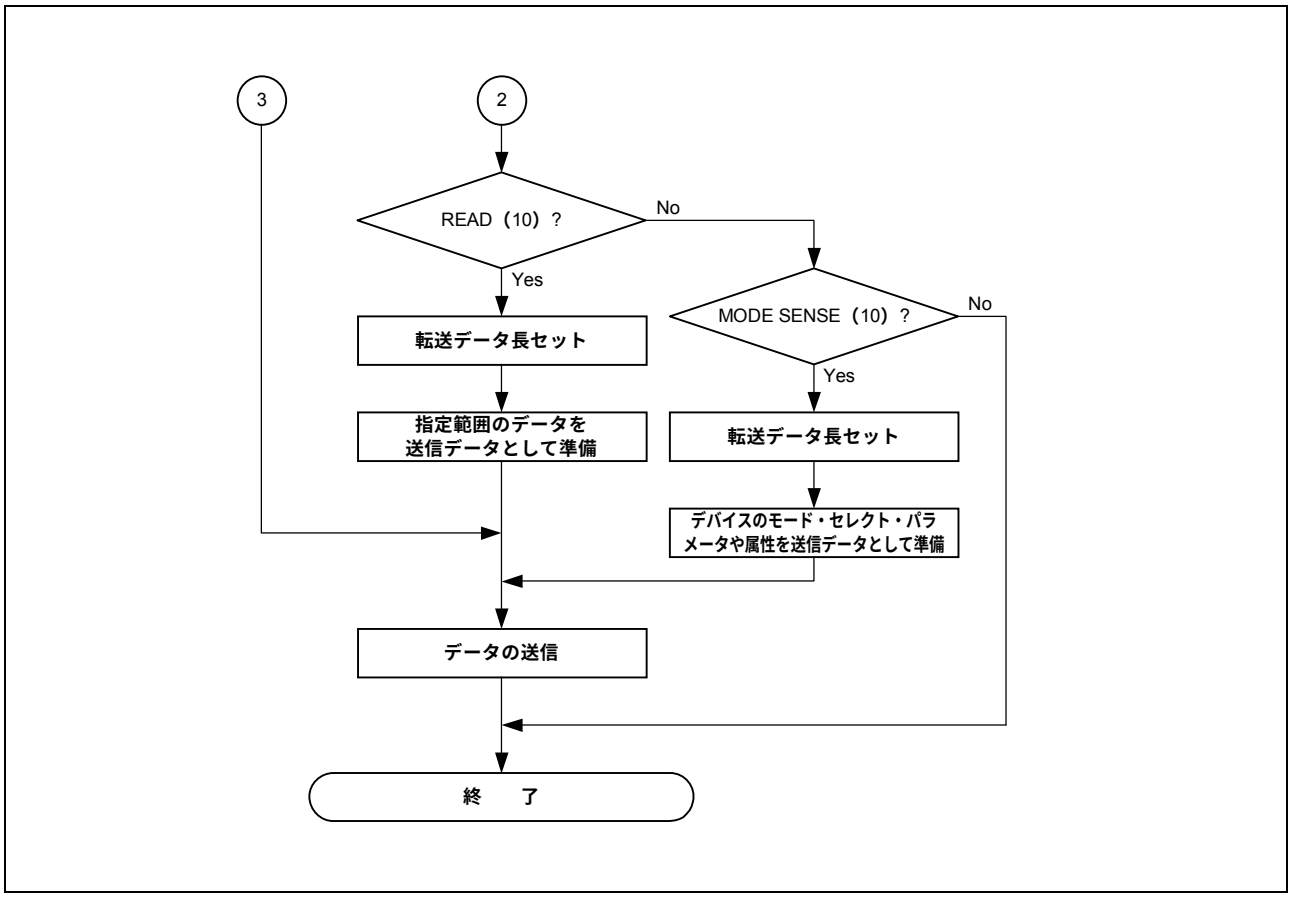


図7－6　READ系コマンドの処理のフロー・チャート（2/2）



[REQUEST SENSE コマンド処理]

センス・データをホストに報告します。

センス・データのフォーマットを次に示します。また，データ値として，サンプル・プログラムで使用するセンス・データについて示します。サンプル・プログラムでは仮想デバイスを扱うため，次に示すデータ値として用意されたデータを返します。

表 7－4　センス・データ・フォーマット

ビット / バイト	7	6	5	4	3	2	1	0	データ値
0	VALID	エラー・コード							70H
1	Reserved								00H
2	Reserved		ILI	Reserved	センス・キー				00H
3	インフォメーション								00H
4	インフォメーション								00H
5	インフォメーション								00H
6	インフォメーション								00H
7	追加センス・データ長（n-7）								0AH
8	コマンド固有インフォメーション								00H
9	コマンド固有インフォメーション								00H
10	コマンド固有インフォメーション								00H
11	コマンド固有インフォメーション								00H
12	アディショナル・センス・コード（ASC）								00H
13	アディショナル・センス・コード・クォリファイア（ASCQ）								00H
14	FRU（Field Replaceable Unit）コード								00H
15	SKSV	センス・キー固有インフォメーション							00H
16	センス・キー固有インフォメーション								00H
17	センス・キー固有インフォメーション								00H

サンプル・ドライバ内でホストに送信するセンス・キーの一覧を，次に示します。

表 7－5　センス・キー一覧

センス・キー	ASC	ASCQ	Description of Error
00	00	00	NO SENSE
05	00	00	ILLEGAL REQUEST.
05	20	00	INVALID COMMAND OPERATION CODE
05	24	00	INVALID FIELD IN COMMAND PACKET

[READ（6）コマンド処理]

ストレージ・デバイスから指定範囲のデータを読み出し，読み出したデータをホストへ送信します。

サンプル・プログラムでは，仮想デバイスから読み出したデータをホストへ送信します。

[INQUIRY コマンド処理]

デバイスについての情報をホストに報告します。

INQUIRY データのフォーマットを次に示します。また，サンプル・プログラムでは仮想デバイスを扱うため，次に示すデータ値として用意されたデータを返します。

表 7－6　INQUIRY データ・フォーマット

ビット / バイト	7	6	5	4	3	2	1	0	データ値
0	クォリファイア			デバイス・タイプ・コード					00H
1	RMB	デバイス・タイプ修飾子							80H
2	バージョン		ECMA バージョン			00H			00H
3	AENC	TrmIOP	01H			01H			02H
4	追加データ長（n－4 バイト）								1FH
5	Reserved								00H
6	Reserved								00H
7	Reserved								00H
8	ベンダ ID（ASCII）								注 1
：	：								注 1
15	ベンダ ID（ASCII）								注 1
16	プロダクト ID（ASCII）								注 2
：	：								注 2
31	プロダクト ID（ASCII）								注 2
32	プロダクト版数（ASCII）								注 3
：	：								注 3
35	プロダクト版数（ASCII）								注 3
36	ベンダ固有								none
：	：								none
55	ベンダ固有								none
56	Reserved								none
：	：								none
95	Reserved								none
96	ベンダ固有								none
：	：								none
n	ベンダ固有								none

注 1.　“ NEC Corp ” の ASCII 文字コード

2.　“ StorageFncDriver ” の ASCII 文字コード

3.　“ 0.12 ” の ASCII 文字コード

[MODE SENSE（6）コマンド処理]

デバイスのモード・セレクト・パラメータや属性をホストに報告します。

MODE SENSE データのフォーマットを次に示します。また，サンプル・プログラムでは仮想デバイスを扱うため，次に示すデータ値として用意されたデータを返します。対応しているページ・コードは，01H だけです。コマンドのページ・コードに関係なく，このデータを返します。

表7−7　MODE SENSE データ・フォーマット

ビット / バイト	7	6	5	4	3	2	1	0	データ値
0	モード・パラメータ長								注1
1	メディア・タイプ								00H
2	デバイス固有パラメータ								00H
3	ブロック・ディスクリプタ長								08H
4	デンシティ・コード								00H
5	00H								00H
6	00H								00H
7	C0H								C0H
8	Reserved								00H
9	00H								00H
10	00H								02H
11	00H								00H
12	PS	Reserved	ページ・コード						注2
13	ページ長（n−13）								0AH
14	モード・パラメータ								注3
：	：								注3
n	モード・パラメータ								注3

注 1. CDB の DBD とページ・コードで指定されるパラメータ・リストか，またはアロケーション長で指定される分のパラメータ・リストかの，どちらか少ない方のバイト数

2. CDB のページ・コード

3. 08H, 0BH, 00H, 00H, 00H, 00H, 00H, 00H, 00H, 00H

[READ FORMAT CAPACITY コマンド処理]

デバイスの容量（ブロック数，ブロック長）をホストに報告します。

READ FORMAT CAPACITY データのフォーマットを次に示します。また，サンプル・プログラムでは仮想デバイスを扱うため，次に示すデータ値として用意されたデータを返します。

表 7－8　READ FORMAT CAPACITY 用データ・フォーマット

ビット / バイト	7	6	5	4	3	2	1	0	データ値
0	Reserved								00H
1	Reserved								00H
2	Reserved								00H
3	キャパシティ・リスト長（バイト）								08H
4	ブロック数								00H
5	ブロック数								00H
6	ブロック数								00H
7	ブロック数								C0H
8	Reserved						Descriptor Code		01H
9	ブロック長								00H
10	ブロック長								02H
11	ブロック長								00H
12	ブロック数								00H
13	ブロック数								00H
14	ブロック数								00H
15	ブロック数								C0H
16	Reserved								00H
17	ブロック長								00H
18	ブロック長								02H
19	ブロック長								00H

[READ CAPACITY コマンド処理]

デバイスのデータ容量をホストに報告します。

READ CAPACITY データのフォーマットを次に示します。また，サンプル・プログラムでは仮想デバイスを扱うため，次に示すデータ値として用意されたデータを返します。

表 7－9　READ CAPACITY 用データ・フォーマット

ビット / バイト	7	6	5	4	3	2	1	0	データ値
0	論理ブロック・アドレス								00H
1	論理ブロック・アドレス								00H
2	論理ブロック・アドレス								00H
3	論理ブロック・アドレス								BFH
4	ブロック長								00H
5	ブロック長								00H
6	ブロック長								02H
7	ブロック長								00H

[READ（10）コマンド処理]

ストレージ・デバイスから指定範囲のデータを読み出し，読み出したデータをホストへ送信します。

サンプル・プログラムでは，仮想デバイスから読み出したデータをホストへ送信します。

[MODE SENSE（10）コマンド処理]

デバイスのモード・セレクト・パラメータや属性をホストに報告します。

MODE SENSE（10）データのフォーマットを次に示します。また，サンプル・プログラムでは仮想デバイスを扱うため，次に示すデータ値として用意されたデータを返します。対応しているページ・コードは，01H だけです。コマンドのページ・コードに関係なく，このデータを返します。

表7－10　MODE SENSE（10）データ・フォーマット

ビット / バイト	7	6	5	4	3	2	1	0	データ値
0	モード・パラメータ長								注1
1	モード・パラメータ長								注1
2	メディア・タイプ								00H
3	デバイス固有パラメータ								00H
4	Reserved								00H
5	Reserved								00H
6	ブロック・ディスクリプタ長								00H
7	ブロック・ディスクリプタ長								08H
8	デンシティ・コード								00H
9	ブロック数								00H
10	ブロック数								00H
11	ブロック数								C0H
12	Reserved								00H
13	ブロック長								00H
14	ブロック長								02H
15	ブロック長								00H
16	PS	Reserved	ページ・コード						注2
17	ページ長（n－17）								0AH
18	モード・パラメータ								注3
：	：								注3
N	モード・パラメータ								注3

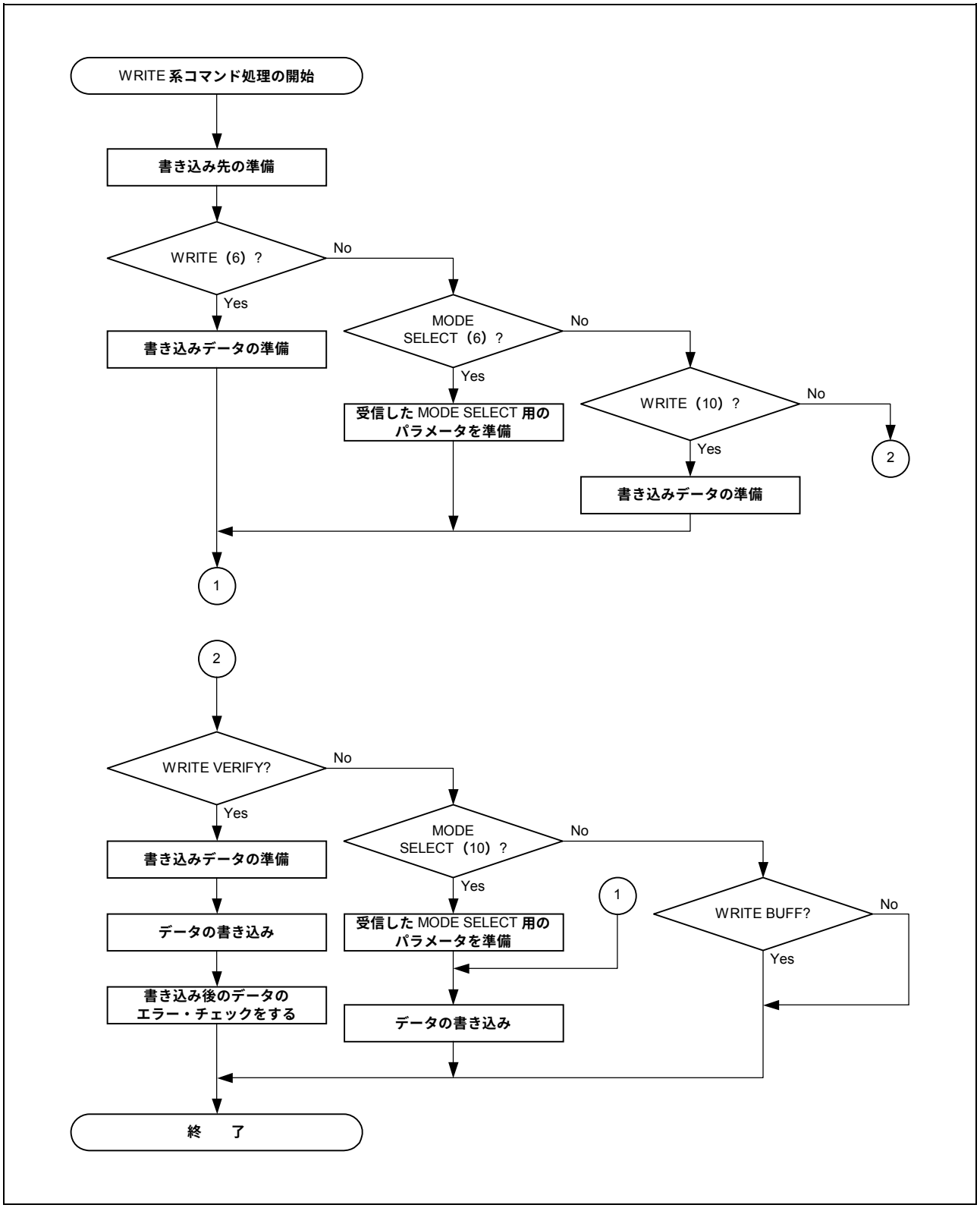
注 1. CDB の DBD とページ・コードで指定されるパラメータ・リストか，またはアロケーション長で指定される分のパラメータ・リストかの，どちらか少ない方のバイト数

2. CDB のページ・コード

3. 08H, 0BH, 00H, 00H, 00H, 00H, 00H, 00H, 00H, 00H

SCSI コマンド（WRITE 系コマンド）処理の流れを次に示します。

図7－7　WRITE系コマンドの処理のフロー・チャート



[WRITE（6）コマンド処理]

受信データをストレージ・デバイスの指定された領域に書き込みます。

サンプル・プログラムでは，受信データを仮想デバイスの指定された領域に書き込みます。

[MODE SELECT（6）コマンド処理]

ロジカル・ユニットの物理的属性，記憶媒体上のデータ形式，エラー・リカバリの方法や手順など，各種パラメータの設定や変更を行います。

MODE SELECT データのフォーマットを次に示します。また，サンプル・プログラムでは仮想デバイスを扱うため，コマンドのページ・コードに関係なく，受信したデータを MODE SELECT TABLE に書き込むだけで，すべて正常終了します。なお，データ値をテーブルの初期値として使用するものとします。

表 7－11　MODE SELECT（6）データ・フォーマット

ビット / バイト	7	6	5	4	3	2	1	0	データ値
0	モード・パラメータ長								17H
1	メディア・タイプ								00H
2	デバイス固有パラメータ								00H
3	ブロック・ディスクリプタ長								08H
4	デンシティ・コード								00H
5	00H								00H
6	00H								00H
7	C0H								C0H
8	Reserved								00H
9	00H								00H
10	00H								02H
11	00H								00H
12	PS	1	ページ・コード						注 1
13	ページ長（n－13）								0AH
14	モード・パラメータ								注 2
：	：								注 2
n	モード・パラメータ								注 2

注 1.　CDB のページ・コード

2.　08H, 0BH, 00H, 00H, 00H, 00H, 00H, 00H, 00H, 00H

[WRITE（10）コマンド処理]

受信データを，ストレージ・デバイスの指定された領域に書き込みます。

サンプル・プログラムでは，受信データを仮想デバイスの指定された領域に書き込みます。

[WRITE VERIFY コマンド処理]

受信データをストレージ・デバイスに書き込みます。書き込み後，データのエラー・チェックをします。サンプル・プログラムでは，受信データを仮想デバイスに書き込み，データのエラー・チェックはせずに正常終了します。

[VERIFY コマンド処理]

ストレージ・デバイス上のデータの正常性を確認します。サンプル・プログラムでは仮想デバイスを扱うため，何も処理せず正常終了します。

[WRITE BUFF コマンド処理]

メモリ（データ・バッファ）にデータを書き込みます。サンプル・プログラムでは仮想デバイスを扱うため，何も処理せず正常終了します。

[MODE SELECT（10）コマンド処理]

ロジカル・ユニットの物理的属性，記憶媒体上のデータ形式，エラー・リカバリの方法や手順など，各種パラメータの設定や変更を行います。

MODE SELECT（10）データのフォーマットを次に示します。また，サンプル・プログラムでは仮想デバイスを扱うため，コマンドのページ・コードに関係なく，受信したデータを MODE SELECT（10）TABLE に書き込むだけで，すべて正常終了します。なお，データ値をテーブルの初期値として使用するものとします。

表 7－12　MODE SELECT（10）データ・フォーマット

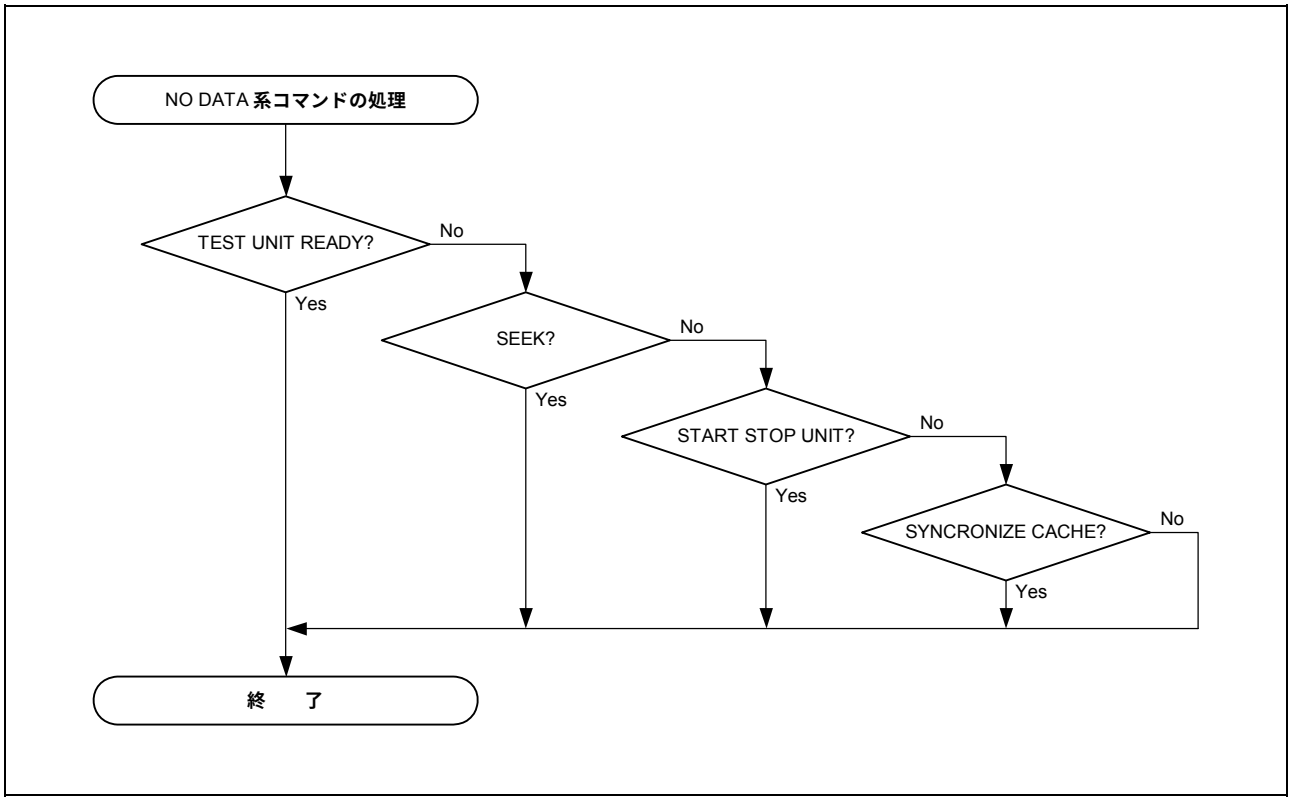
ビット / バイト	7	6	5	4	3	2	1	0	データ値
0	モード・パラメータ長								00H
1	モード・パラメータ長								1AH
2	メディア・タイプ								00H
3	デバイス固有パラメータ								00H
4	Reserved								00H
5	Reserved								00H
6	ブロック・ディスクリプタ長								00H
7	ブロック・ディスクリプタ長								08H
8	デンシティ・コード								00H
9	ブロック数								00H
10	ブロック数								00H
11	ブロック数								C0H
12	Reserved								00H
13	ブロック長								00H
14	ブロック長								02H
15	ブロック長								00H
16	PS	Reserved	ページ・コード						注 1
17	ページ長（n－17）								0AH
18	モード・パラメータ								注 2
：	：								注 2
n	モード・パラメータ								注 2

注 1.　CDB のページ・コード

2.　08H, 0BH, 00H, 00H, 00H, 00H, 00H, 00H, 00H, 00H

SCSI コマンド（NO DATA 系コマンド）処理の流れを次に示します。

図7－8　NO DATA系コマンドの処理のフロー・チャート



[TEST UNIT READY コマンド処理]

ユニットの状態を報告します。サンプル・プログラムでは仮想デバイスを扱うため，何も処理せず正常終了します。

[SEEK コマンド処理]

指定ブロック位置へのシーク動作を行います。サンプル・プログラムでは仮想デバイスを扱うため，何も処理せず正常終了します。

[START STOP UNIT コマンド処理]

ユニットへのアクセス制限を設定します。サンプル・プログラムでは仮想デバイスを扱うため，何も処理せず正常終了します。

[SYNCHRONIZE CACHE コマンド処理]

ユニットと CACHE とのデータの整合をとります。サンプル・プログラムでは仮想デバイスを扱うため，何も処理せず正常終了します。

7. 3　USB ストレージ・クラス用ドライバのディスクリプタ情報

サンプル・プログラムで定義されている USB 標準ディスクリプタについて，次に示します。
（a）-（d）までのディスクリプタは，必ず用意してください。

備考　詳細は，Universal Serial Bus Specification Revision 1.1 を参照してください。

（a）Device Descriptor

デバイスの一般的な情報を持ち，デバイスごとに 1 つの Device Descriptor を用意してください。この情報は，デバイスのコンフィギュレーションで唯一のデバイスを認識するために使用されます。USB ストレージ・クラスは，デバイス・レベルで現在のクラスにおける具体的な情報は使用しません。

表 7－13　Device Descriptor

オフセット	サイズ（バイト）	値	説　明
0	1	12H	このディスクリプタの Length 値（バイト）
1	1	01H	ディスクリプタ・タイプ（デバイス）
2	2	10H/01H	USB のバージョン（USB1.1）
4	1	00H	クラス・コード
5	1	00H	サブクラス・コード
6	1	00H	プロトコル・コード
7	1	40H	Endpoint0 の最大パケット・サイズ
8	2	09H/04H	ベンダ ID（NEC エレクトロニクス）
10	2	FCH/FFH	プロダクト ID
12	2	01H/00H	デバイス・リリース番号
14	1	01H	ストリング・ディスクリプタへのインデクス（Manufacturer）
15	1	00H	ストリング・ディスクリプタへのインデクス（Product）
16	1	00H	ストリング・ディスクリプタへのインデクス（Serial Number）
17	1	01H	可能なコンフィギュレーションの数

(b) Configuration Descriptor

具体的なデバイス・コンフィギュレーションについての情報を保持しています。

USB ストレージ・クラスは，コンフィギュレーション・レベルで現在のクラスにおける具体的な情報は使用しません。

表 7－14　Configuration Descriptor

オフセット	サイズ(バイト)	値	説　明
0	1	09H	このディスクリプタの Length 値（バイト）
1	1	02H	ディスクリプタ・タイプ（コンフィギュレーション）
2	2	20H/00H	Get Descriptor 要求でコンフィギュレーション・ディスクリプタと一緒に返されるディスクリプタのトータル Length 値
4	1	01H	この構成でサポートされているインタフェース数
5	1	01H	コンフィギュレーション値
6	1	00H	ストリング・ディスクリプタへのインデクス（Configuration）
7	1	C0H	デバイスの構成（Self-powered/Remote Wakeup 機能）
8	1	00H	デバイスの最大消費電力

(c) Interface Descriptor

コンフィギュレーション内の具体的なインタフェース情報を保持しています。

サンプル・プログラムでは，コンフィギュレーションは 1 つのインタフェースを提供します。また，このインタフェースは 2 つの Endpoint をサポートし，この数だけ Endpoint Descriptor を持っています。

Interface Descriptor は，常に Configuration Descriptor の一部として戻され，Interface Descriptor において，Get Descriptor および Set Descriptor 要求による直接アクセスはされません。

表 7－15　Interface Descriptor（1）

オフセット	サイズ(バイト)	値	説　明
0	1	09H	このディスクリプタの Length 値（バイト）
1	1	04H	ディスクリプタ・タイプ（インタフェース）
2	1	00H	インタフェース値
3	1	00H	Alternate 設定値
4	1	02H	Endpoint 数（Endpoint0 を除く）
5	1	08H	インタフェース・クラス（マス・ストレージ・クラス）
6	1	06H	インタフェース・サブクラス（SCSI）
7	1	50H	インタフェース・プロトコル（Bulk-Only）
8	1	00H	ストリング・ディスクリプタへのインデクス（インタフェース）

（d）Endpoint Descriptor

ホストが個々の Endpoint のバンド幅要件を決定するために必要な情報を保持しています。

Endpoint Descriptor は，常にコンフィギュレーション・ディスクリプタの一部として戻され，Endpoint Descriptor において，Get Descriptor および Set Descriptor 要求による直接アクセスはされません。

表 7－16　Endpoint Descriptor（Bulk IN）

オフセット	サイズ（バイト）	値	説　明
0	1	07H	このディスクリプタの Length 値（バイト）
1	1	05H	ディスクリプタ・タイプ（Endpoint）
2	1	81H	Endpoint のアドレス値
3	1	02H	Endpoint の転送タイプ
4	2	40H/00H	Endpoint の最大パケット・サイズ
6	1	00H	インターバル（ms）：Isochronous, Interrupt Endpoint だけ有効

表 7－17　Endpoint Descriptor（Bulk OUT）

オフセット	サイズ（バイト）	値	説　明
0	1	07H	このディスクリプタの Length 値（バイト）
1	1	05H	ディスクリプタ・タイプ（Endpoint）
2	1	02H	Endpoint のアドレス値
3	1	02H	Endpoint の転送タイプ
4	2	40H/00H	Endpoint の最大パケット・サイズ
6	1	00H	インターバル（ms）：Isochronous, Interrupt Endpoint だけ有効

（e）String Descriptor

このサンプル・プログラムでは，デバイスの製造社名情報を保持します。

表 7－18　String Descriptor（1）

オフセット	サイズ（バイト）	値	説　明
0	1	04H	このディスクリプタの Length 値（バイト）
1	1	03H	ディスクリプタ・タイプ（ストリング）
2	2	09H/04H	ストリング・ディスクリプタで使用する言語タイプ（English/U.S.）

表 7－19　String Descriptor（2）

オフセット	サイズ（バイト）	値	説　明
0	1	2AH	このディスクリプタの Length 値（バイト）
1	1	03H	ディスクリプタ・タイプ（ストリング）
2	40	'N','E','C','','E','l','e','c','t','r','o','n','i','c','s',' ','C','o','.'	製造社名（Manifacturer） NEC Electronics Co.

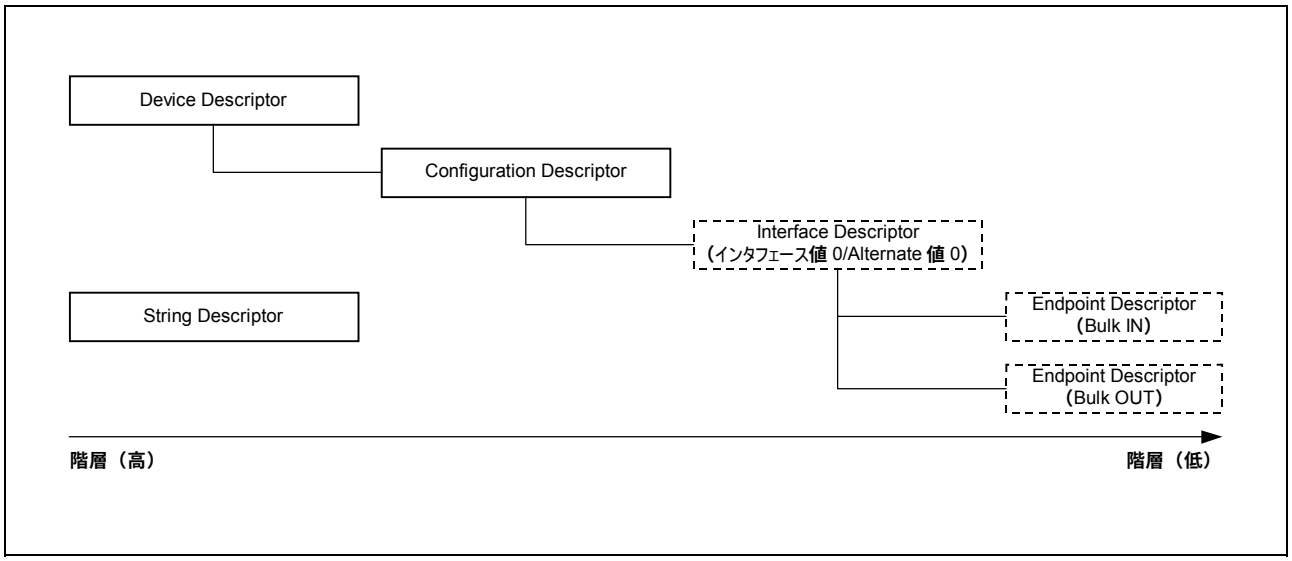
7. 3. 1　ディスクリプタの構成

サンプル・プログラムにおけるディスクリプタの構成を次に示します。

前述したディスクリプタ類を次の構成で準備します。

注意　Device Descriptor/Configuration Descriptor/String Descriptor は，それぞれ別々の Get Descriptor リクエストによってアクセスされます。Interface Descriptor および Endpoint Descriptor は，Configuration Descriptor の一部としてアクセスされます。

図7－9　ディスクリプタ構成図



7. 4　データ・マクロ

USB ストレージ・クラス用ドライバ内で使用する各種データ・マクロ（データ・タイプ，戻り値など）について，次に示します。

7. 4. 1　データ・タイプ

USB ストレージ・クラス用ドライバ関数を発行するときに指定する各種パラメータのデータ・タイプのマクロ定義は，ヘッダ・ファイル nectools32¥USB_Storage¥inc¥types.h で行われています。

データ・タイプ一覧を，次に示します。

表 7－20　データ・タイプ一覧

マクロ	型	意　味
ULONG	unsigned long	符号なし 32 ビット整数
WORD	unsigned long	符号なし 32 ビット整数
HWORD	unsigned short	符号なし 16 ビット整数
BYTE	unsigned char	符号なし 8 ビット整数
(*PFV) ()	void	処理プログラムの起動アドレス

7. 4. 2　戻り値

USB ストレージ・クラス用ドライバ関数からの戻り値のマクロ定義は，ヘッダ・ファイル nectools32¥ USB_BUS¥inc¥errno.h で行われています。

戻り値一覧を次に示します。

表 7－21　戻り値一覧

マクロ	数 値	意　味
DEV_OK	0	正常終了
DEV_ERROR	－1	異常終了
DEV_ERR_NODATA	－2	NO DATA 系コマンドで転送方向エラー
DEV_ERR_READ	－3	READ 系コマンドで転送方向エラー
DEV_ERR_WRITE	－4	WRITE 系コマンドで転送方向エラー
DEV_ERR_VERIFY	－5	ベリファイ・エラー
DEV_ERR_CBWLENGTH	－6	CBW レングス・エラー
DEV_ERR_CBWCBW	－7	CBW 処理中に CBW を受信（エラー）

7. 5　データ構造体

USB ストレージ・クラス用ドライバが使用するデータ構造体について，次に示します。

7. 5. 1　USB デバイス・リクエスト構造体

USB デバイス・リクエスト構造体の定義は，USB 用ヘッダ・ファイル nectools32¥V850USB_Storage¥src¥USBF¥usbf850.h で行われています。USB デバイス・リクエスト構造体 USB_SETUP を次に示します。

```
typedef struct {
      unsigned char RequstType;         /*bmRequestType */
      unsigned char Request;            /*bRequest */
      unsigned short Value;             /*wValue */
      unsigned short Index;             /*wIndex */
      unsigned short Length;            /*wLength */
      unsigned char* Data;              /*index to Data */
} USB_SETUP;
```

7. 5. 2　CBW データ構造体

USB ストレージ・クラス用ドライバが扱う CBW（Command Block Wrapper）用のデータ構造体の定義は，ヘッダ・ファイル nectools32¥USB_Storage¥inc¥types.h で行われています。CBW データ構造体を次に示します。

```
typedef struct {
      unsigned char dCBWSignature[4];             /*CBW シグネチャ*/
      unsigned char dCBWTag[4];                   /*CBW タグ*/
      unsigned char dCBWDataTransferLength[4];    /*転送データ長*/
      unsigned char bmCBWFlags;                   /*データ方向（OUT/IN）の指定*/
      unsigned char bCBWLUN;                      /*対象デバイスの番号*/
      unsigned char bCBWCBLength;                 /*CBWCB の有効バイト数*/
      unsigned char CBWCB[16];                    /*CBWCB（コマンド）*/
} CBW_INFO,*PCBW_INFO;
```

7. 5. 3　CSW データ構造体

USB ストレージ・クラス用ドライバが扱う CSW（Command Status Wrapper）用のデータ構造体の定義は，ヘッダ・ファイル nectools32¥USB_Storage¥inc¥types.h で行われています。CSW データ構造体を次に示します。

```
typedef struct {
      unsigned char dCSWSignature[4];       /*CSW シグネチャ*/
      unsigned char dCSWTag[4];             /*CSW タグ*/
      unsigned char dCSWDataResidue[4];     /*CBW 指定の転送データ長と処理したデータ長の差*/
      unsigned char bmCSWStatus;            /*CBW 処理結果のステータス*/
} CSW_INFO,*PCSW_INFO;
```

7. 6　関数解説

7. 6. 1　概　　要

この章で説明している各処理プログラムの一覧を次に示します。

表 7－22　サンプル・プログラムの処理プログラム一覧（1/3）

処理プログラム名	関数名	ファイル名	備　　考
RX850 Pro 依存処理プログラム			
CF 定義ファイル	－	sys.cf	－
エントリ処理	－	entry.s	アセンブリ言語
ブート処理	boot	boot.s	アセンブリ言語
ハードウエア初期化部	__InitSystemTimer	init.c	C 言語
初期化ハンドラ	varfunc	varfunc.c	C 言語
ヘッダ・ファイル	－	init.h	－
リンク・ディレクティブ・ファイル	－	usb_storage.dir	－
ボード依存部処理プログラム			
ポートの初期化	port850_reset	port.c	C 言語
ヘッダ・ファイル	－	port.h	－
ヘッダ・ファイル			
PFESiP/V850EP1 の USB 関連レジスタ定義ファイル		aurora_usb_reg.h	
データ・タイプ宣言	－	types.h	－
戻り値宣言	－	errno.h	－

表 7－22　サンプル・プログラムの処理プログラム一覧（2/3）

処理プログラム名	関数名	ファイル名	備考
USB ストレージ・クラス用ドライバ処理プログラム			
初期化関数	usbf850_init	usbf850.c	C 言語
割り込みハンドラ（INTUSBF0 信号用）	usbf850_inthdr	usbf850.c	C 言語
割り込みハンドラ（INTUSBF1 信号用）	usbf850_inthdr1	usbf850.c	C 言語
割り込みハンドラ（INTUSBF2 信号用）	usbf850_inthdr2	usbf850.c	C 言語
コントロール転送処理用タスク（エンドポイント 0）	task_usb0b	usbf850.c	C 言語
バルク・アウト処理用タスク（エンドポイント 2）	task_usb1b	usbf850.c	C 言語
データ送信関数	usbf850_data_send	usbf850.c	C 言語
データ受信関数	usbf850_data_receive	usbf850.c	C 言語
Null データ送信関数（エンドポイント 0）	usbf850_sendnullEP0	usbf850.c	C 言語
Stall 応答処理関数（エンドポイント 0）	usbf850_sendstallEP0	usbf850.c	C 言語
Stall 応答処理関数（エンドポイント 1）	usbf850_bulkin1_stall	usbf850.c	C 言語
Stall 応答処理関数（エンドポイント 2）	usbf850_bulkout1_stall	usbf850.c	C 言語
システム・コール呼び出し関数（loc_cpu）	usbf850_loc_cpu	usbf850.c	C 言語
システム・コール呼び出し関数（unl_cpu）	usbf850_unl_cpu	usbf850.c	C 言語
リクエスト処理関数	usbf850_rxreq	usbf850.c	C 言語
リクエスト・データ読み出し関数	usbf850_rxreq_read	usbf850.c	C 言語
標準リクエスト処理関数	usbf850_standardreq	usbf850.c	C 言語
Get Descriptor リクエスト処理関数	usbf850_getdesc	usbf850.c	C 言語
リクエスト処理関数設定用 Stall 応答処理関数（エンドポイント 0）	usbf850_sstall_ctrl	usbf850.c	C 言語
USB 用ヘッダ・ファイル	－	usbf850.h	－
USB 用ディスクリプタ宣言	－	usbf850desc.h	－
Bulk-Only Mass Storage Reset リクエスト処理関数（デバイス・クラス固有のリクエスト処理）	usbf850_blkonly_mass_storage_reset	usbf850_storage.c	C 言語
Max LUN リクエスト処理関数（デバイス・クラス固有のリクエスト処理）	usbf850_max_lun	usbf850_storage.c	C 言語
USB ストレージ・クラス用デバイス・クラス固有のリクエスト処関数の登録処理関数理	usbf850_setfunction_storage	usbf850_storage.c	C 言語
CBW 受信処理関数	usbf850_rx_cbw	usbf850_storage.c	C 言語
CBW チェック関数	usbf850_storage_cbwchk	usbf850_storage.c	C 言語
CBW のエラー処理関数	usbf850_cbw_error	usbf850_storage.c	C 言語
CBW の NO DATA 系コマンド処理関数	usbf850_no_data	usbf850_storage.c	C 言語
CBW の DATA IN 系コマンド処理関数	usbf850_data_in	usbf850_storage.c	C 言語
CBW の DATA OUT 系コマンド処理関数	usbf850_data_out	usbf850_storage.c	C 言語
CSW 送信処理関数	usbf850_csw_ret	usbf850_storage.c	C 言語
USB-ストレージ間インタフェース関数用ヘッダ・ファイル	－	usbf850_storage.h	－
USB 用 DMA 初期化処理関数	usbf850_dma_init	usbf850_dma.c	C 言語
USB 用 DMA 開始処理関数	usbf850_dma_start	usbf850_dma.c	C 言語
DMA 用ヘッダ・ファイル	－	usbf850_dma.h	－

表 7－22　サンプル・プログラムの処理プログラム一覧（3/3）

処理プログラム名	関数名	ファイル名	備考
ストレージ・デバイス処理プログラム			
ストレージ・デバイス初期化関数	storageDev_Init	ata_ctrl.c	C 言語
ストレージ・デバイス用ヘッダ・ファイル	－	ata.h	－
CBWCB **コマンド解析処理関数**	scsi_command_to_ata	scsi_cmd.c	C 言語
TEST UNIT READY **コマンド処理関数**	ata_test_unit_ready	scsi_cmd.c	C 言語
SEEK **コマンド処理関数**	ata_seek	scsi_cmd.c	C 言語
START STOP UNIT **コマンド処理関数**	ata_start_stop_unit	scsi_cmd.c	C 言語
SYNCHRONIZE CACHE **コマンド処理関数**	ata_synchronize_cache	scsi_cmd.c	C 言語
REQUEST SENSE **コマンド処理関数**	ata_request_sense	scsi_cmd.c	C 言語
INQUIRY **コマンド処理関数**	ata_inquiry	scsi_cmd.c	C 言語
MODE SELECT **コマンド処理関数**	ata_mode_select	scsi_cmd.c	C 言語
MODE SELECT（10）**コマンド処理関数**	ata_mode_select10	scsi_cmd.c	C 言語
MODE SENSE **コマンド処理関数**	ata_mode_sense	scsi_cmd.c	C 言語
MODE SENSE(10)**コマンド処理関数**	ata_mode_sense10	scsi_cmd.c	C 言語
READ FORMAT CAPACITIES **コマンド処理関数**	ata_read_format_capacities	scsi_cmd.c	C 言語
READ CAPACITY **コマンド処理関数**	ata_read_capacity	scsi_cmd.c	C 言語
READ（6）**コマンド処理関数**	ata_read6	scsi_cmd.c	C 言語
READ（10）**コマンド処理関数**	ata_read10	scsi_cmd.c	C 言語
WRITE（6）**コマンド処理関数**	ata_write6	scsi_cmd.c	C 言語
WRITE（10）**コマンド処理関数**	ata_write10	scsi_cmd.c	C 言語
VERIFY **コマンド処理関数**	ata_verify	scsi_cmd.c	C 言語
WRITE VERIFY **コマンド処理関数**	ata_write_verify	scsi_cmd.c	C 言語
WRITE BUFF **コマンド処理関数**	ata_write_buff	scsi_cmd.c	C 言語
SCSI **から** USB **向けデータ送信処理関数**	scsi_to_usb	scsi_cmd.c	C 言語
SCSI **コマンド処理用ヘッダ・ファイル**	－	scsi.h	－
関数マクロ			
PFESiP/V850EP1 **周辺** I/O **レジスタ設定関数（**1 **バイト単位：**8 **ビット）**	USBF850REG_SET	usbf850.h	C 言語
PFESiP/V850EP1 **周辺** I/O **レジスタ読み出し関数（**1 **バイト単位：**8 **ビット）**	USBF850REG_READ	usbf850.h	C 言語
PFESiP/V850EP1 **周辺** I/O **レジスタ設定関数（**1 **ワード単位：**16 **ビット）**	USBF850REG_SET_W	usbf850.h	C 言語
PFESiP/V850EP1 **周辺** I/O **レジスタ読み出し関数（**1 **ワード単位：**16 **ビット）**	USBF850REG_READ_W	usbf850.h	C 言語

7. 6. 2　関数ツリー

サンプル・プログラムの呼び出し関係（関数ツリー）を次に示します。

注意　usbf850_init および storageDev_Init は，初期化ハンドラから呼び出されます。
また，usbf850_dma_init は，usbf850_init から呼び出されます。

図7－10　サンプル・プログラムの関数ツリー（1/4）

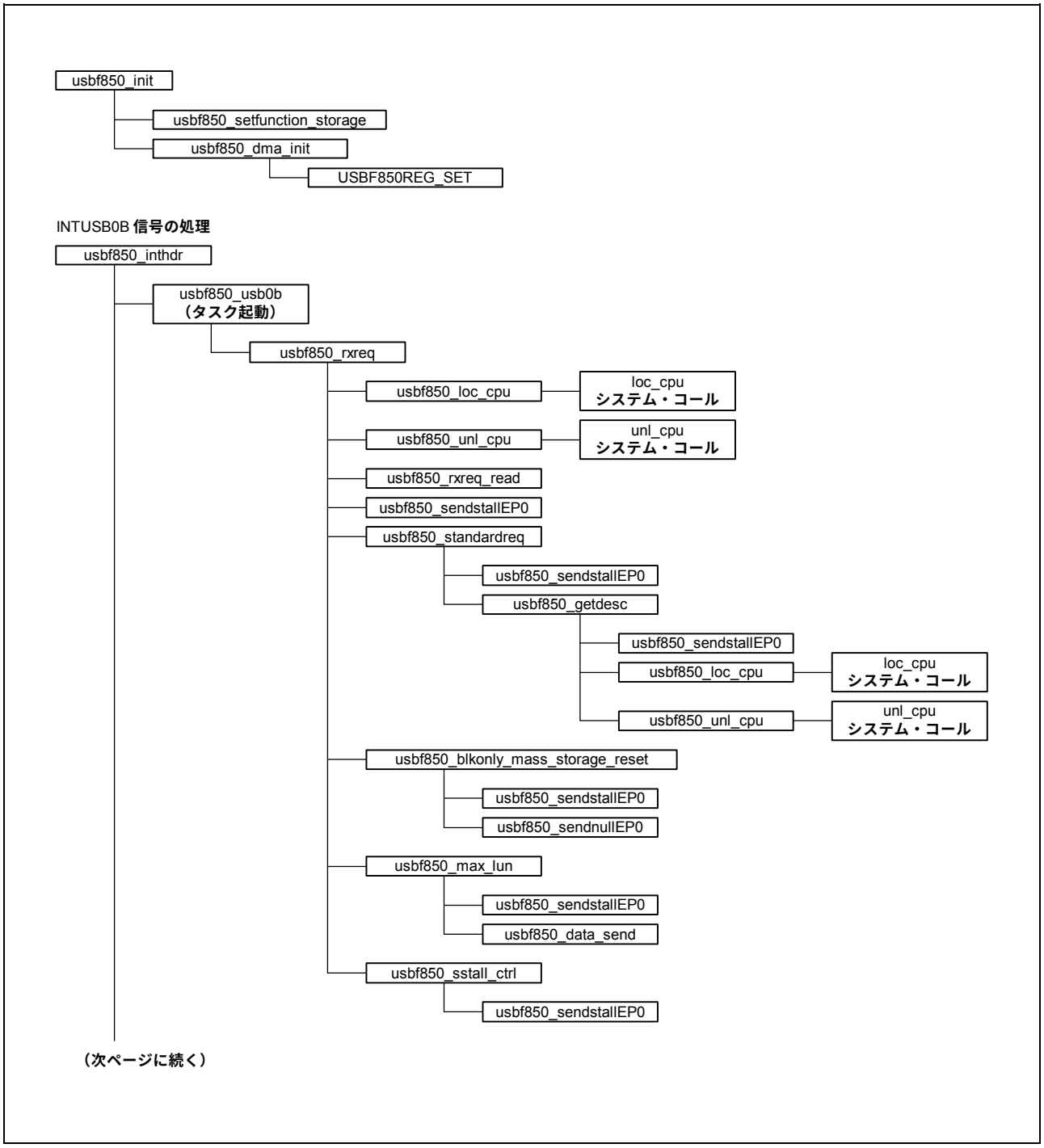


図7－10　サンプル・プログラムの関数ツリー（2/4）

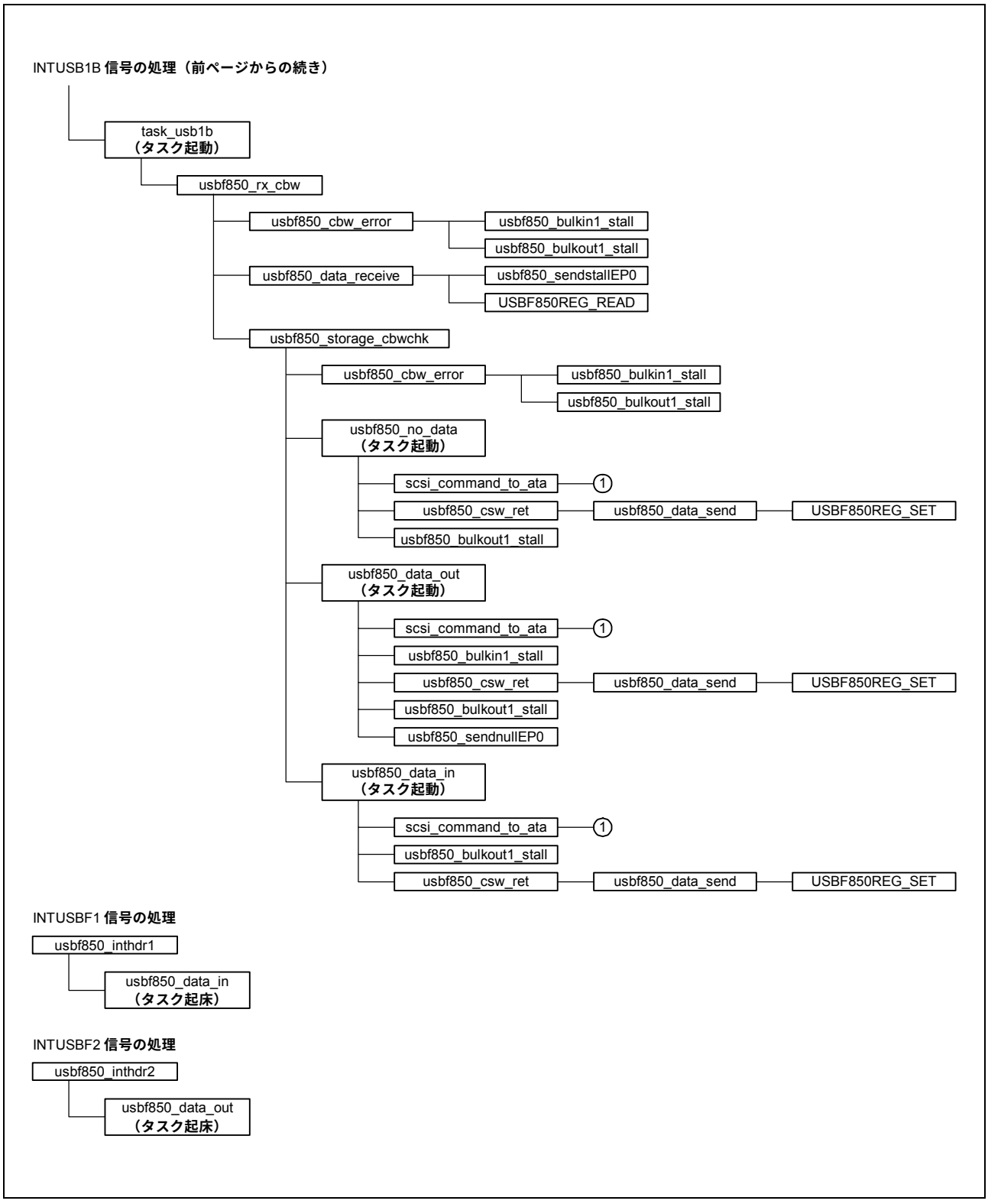


図7－10　サンプル・プログラムの関数ツリー（3/4）

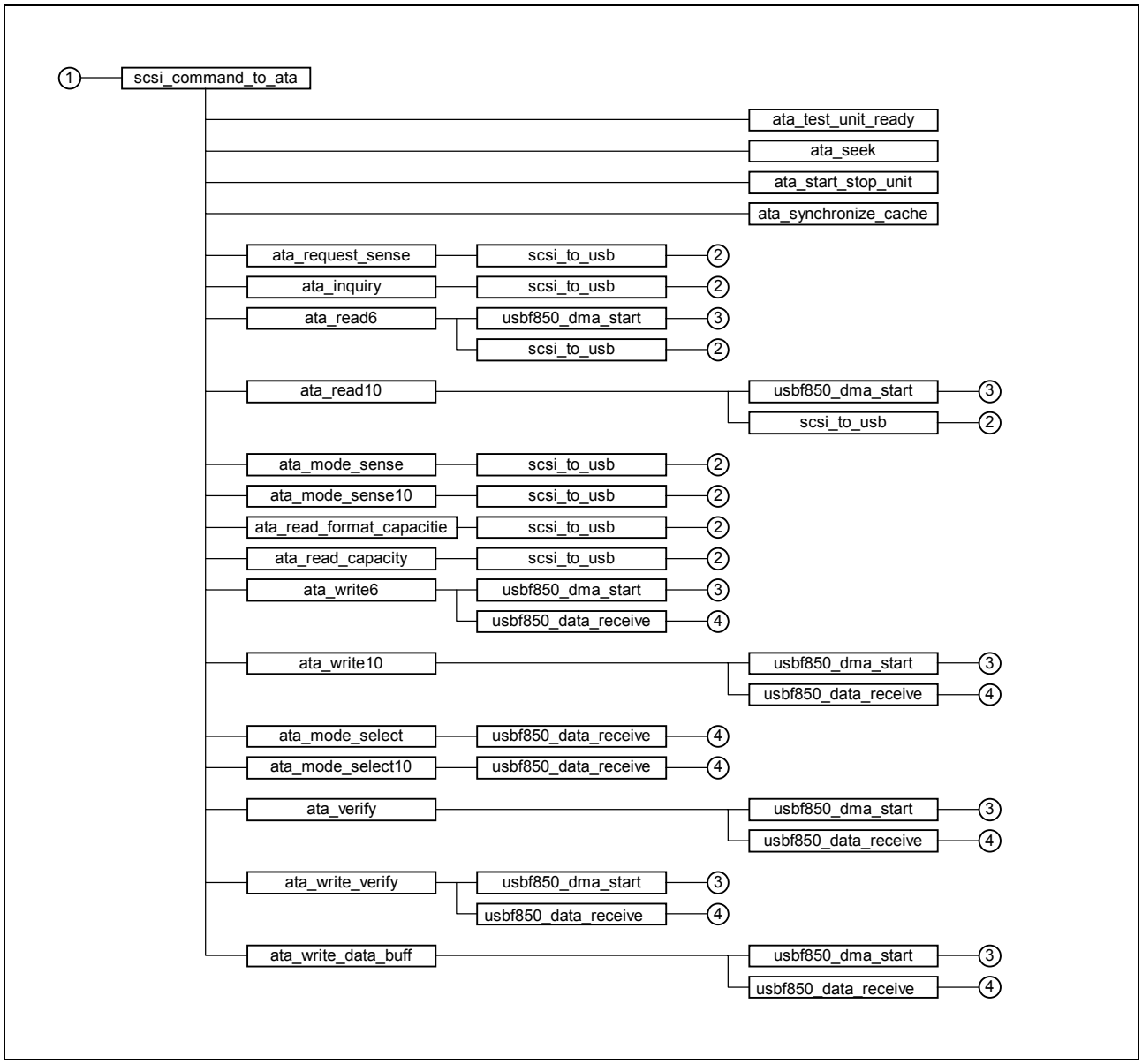
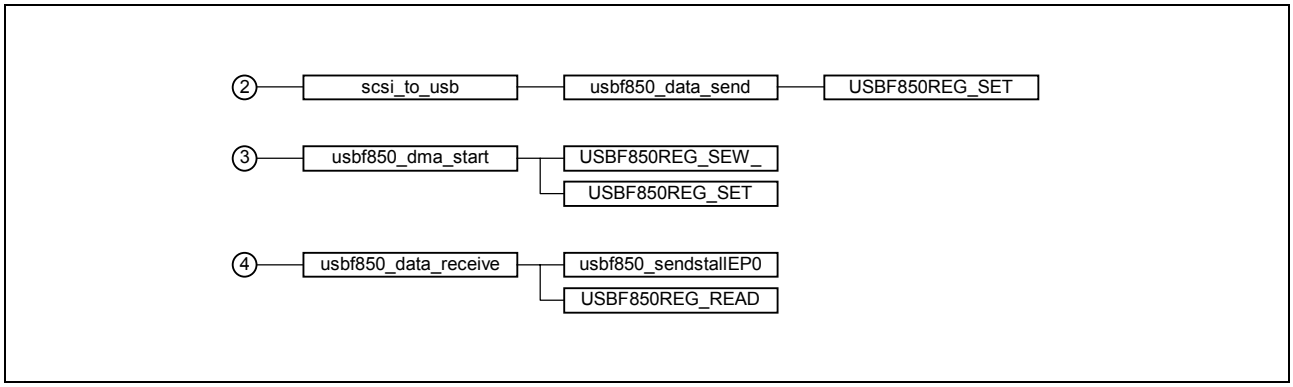


図7－10　サンプル・プログラムの関数ツリー（4/4）



7.6.3 関数解説

このサンプル・プログラムの関数群について，次の記述フォーマットにしたがって解説します。

xxxx … ①	発行有効範囲：－－－－ … ②

① 名称　関数の名称を示しています。

② 発行有効範囲　関数の発行が可能な処理プログラムの種別を示しています。

タスク	：タスクからだけ発行可能
非タスク	：非タスクからだけ発行可能
タスク｜非タスク	：タスク，非タスクのどちらからも発行可能
－	：割り込みハンドラまたはタスクのため，関数コールできない

［概　要］ … ③　関数の機能概要を示しています。

［C 言語形式］ … ④　関数を C 言語で記述された処理プログラムから発行するときの記述形式を示しています。

［パラメータ］ … ⑤　関数のパラメータを次の形式で示しています。

I/O	パラメータ	説 明
A	B	C

A ：パラメータの種類

I　… USB ファンクション・コントローラへの入力パラメータ

O　… USB ファンクション・コントローラからの出力パラメータ

B ：パラメータのデータ・タイプ

C ：パラメータの説明

［機　能］ … ⑥　関数の機能詳細を示しています。

［戻り値］ … ⑦　関数からの戻り値を，データ・マクロおよび数値で示しています。

usbf850_init	発行有効範囲：非タスク｜タスク

［概　要］　PFESiP/V850EP1 に内蔵している USB ファンクション・コントローラの初期化処理を行います。

［C 言語形式］void usbf850_init (void)

［パラメータ］

I/O	パラメータ	説 明
–	–	–

［機　能］　ソフトウエア初期化部から呼び出され，PFESiP/V850EP1 に内蔵している USB ファンクション・コントローラの初期化処理を行います。

備考　初期化処理についての詳細は，7. 2. 1　**初期化処理**を参照してください。

［戻り値］　なし

usbf850_inthdr	発行有効範囲：−

［概　要］　PFESiP/V850EP1 に内蔵している USB ファンクション・コントローラ用割り込みハンドラ（INTUSBF0 信号用）。

［C 言語形式］ID usbf850_inthdr (void)

［パラメータ］

I/O	パラメータ	説 明
−	−	−

［機　能］　INTUSBF0 信号（USB ファンクション・ステータス 0）により起動される割り込みハンドラです。
サンプル・プログラムでは，割り込み要因を調べ CPUDEC 割り込みだった場合，コントロール転送処理用のタスク（task_usb0b）を起動します。
BKO1DT 割り込みだった場合は，バルク・アウト処理用のタスク（task_usb1b）を起動します。
このハンドラは，CF 定義ファイルで定義されています。

備考　割り込み処理についての詳細は，7. 2. 2　**割り込み処理**を参照してください。

［戻り値］　オブジェクト ID 番号（タスクの ID 番号）

usbf850_inthdr1	発行有効範囲：－

［概　要］　PFESiP/V850EP1 に内蔵している USB ファンクション・コントローラ用割り込みハンドラ（INTUSBF1 信号用）。

［C 言語形式］ ID usbf850_inthdr1 (void)

［パラメータ］

I/O	パラメータ	説 明
－	－	－

［機　能］　INTUSBF1 信号（USB ファンクション・ステータス 1）により起動される割り込みハンドラです。
UF0IS0 レジスタ（UF0 INT ステータス 0 レジスタ）を読み出し割り込み要因を調べ，DMAED 割り込みまたは SHORT 割り込みだった場合は，さらに UF0DMS1 レジスタ（DMA ステータス 1 レジスタ）を読み出し割り込み要因を調べ，usbf850_data_in タスクを起床させます（usbf850_data_in は DMA 起動後スリープ状態）。
このハンドラは，CF 定義ファイルで定義されています。

備考　割り込み処理についての詳細は，7. 2. 2　**割り込み処理**を参照してください。

［戻り値］　オブジェクト ID 番号（タスクの ID 番号）

usbf850_inthdr2	発行有効範囲：−

［概　要］　PFESiP/V850EP1 に内蔵している USB ファンクション・コントローラ用割り込みハンドラ（INTUSBF2 信号用）。

［C 言語形式］ID usbf850_inthdr2 (void)

［パラメータ］

I/O	パラメータ	説 明
−	−	−

［機　能］　INTUSBF2 信号（USB ファンクション・ステータス 2）により起動される割り込みハンドラです。
UF0IS0 レジスタ（UF0 INT ステータス 0 レジスタ）を読み出し割り込み要因を調べ，DMAED 割り込みまたは SHORT 割り込みだった場合は，さらに UF0DMS1 レジスタ（DMA ステータス 1 レジスタ）を読み出し割り込み要因を調べ，usbf850_data_out タスクを起床させます（usbf850_data_ out は DMA 起動後スリープ状態）。
このハンドラは，CF 定義ファイルで定義されています。

［戻り値］　オブジェクト ID 番号（タスクの ID 番号）

task_usb0b	発行有効範囲：－

［概　要］　INTUSBF0 信号による割り込み処理を行います。

［C 言語形式］void task_usb0b (VP exinf)

［パラメータ］

I/O	パラメータ	説 明
I	VP	exinf 拡張情報

対象タスクに関する“ユーザ独自の情報”を格納するための領域で，ユーザが自由に利用できます。exinf に設定された情報は，処理プログラム（タスク，非タスク）から ref_tsk システム・コールを発行することにより，ダイナミックに獲得できます。

備考　システム・コールの詳細は，RX850 Pro ユーザーズ・マニュアル 基礎編を参照してください。

［機　能］　INTUSBF0 割り込み信号（USB ファンクション・ステータス 0 割り込み）用の割り込みハンドラから起動されるタスクです。サンプル・プログラムでは，usbf850_rxreq 関数を呼び出し，USB 標準デバイス・リクエストおよびデバイス・クラス固有のリクエストの処理を行います。

注意　このサンプル・プログラムでは，PFESiP/V850EP1 に内蔵している USB ファンクション・コントローラで自動応答しない標準デバイス・リクエスト「Get Descriptor（String Descriptor）」だけを処理します。

備考　割り込み処理についての詳細は，7. 2. 2　**割り込み処理**を参照してください。

［戻り値］　なし

task_usb1b	発行有効範囲：－

［概　要］　INTUSBF1 信号による割り込み処理を行います。

［C 言語形式］void task_usb1b (VP exinf)

［パラメータ］

I/O	パラメータ	説 明
I	VP	exinf 拡張情報

対象タスクに関する“ユーザ独自の情報”を格納するための領域で，ユーザが自由に利用できます。exinf に設定された情報は，処理プログラム（タスク，非タスク）から ref_tsk システム・コールを発行することにより，ダイナミックに獲得できます。

備考　システム・コールの詳細は，RX850 Pro ユーザーズ・マニュアル 基礎編を参照してください。

［機　能］　INTUSBF1 割り込み信号（USB ファンクション・ステータス 1 割り込み）用の割り込みハンドラから起動されるタスクです。サンプル・プログラムでは，割り込み要因を確認し，割り込み要因が BKO1DT で受信データ長が CBW のデータ・サイズに等しければ，usbf850_rx_cbw 関数を呼び出します。

備考　割り込み処理についての詳細は，7. 2. 2　**割り込み処理**を参照してください。

［戻り値］　なし

usbf850_data_send	発行有効範囲：非タスク｜タスク

［概　要］　USB ファンクション・コントローラ用データ送信関数。

［C 言語形式］long usbf850_data_send (unsigned char* data, long len, char ep)

［パラメータ］

I/O	パラメータ	説 明
I	unsigned char* data	送信データの先頭アドレス
I	long len	データ・サイズ
i	char ep	エンドポイント番号

［機　能］　data で指定されたアドレスから，len で指定されたサイズ分のデータを，ep で指定されたエンドポイントで送信処理を行います。

備考

［戻り値］　送信時のステータス

DEV_ERROR　：エンドポイント番号が不正

DEV_OK　：正常終了

usbf850_data_receive	発行有効範囲：非タスク｜タスク

［概　要］　USB ファンクション・コントローラ用データ受信関数。

［C 言語形式］long usbf850_data_receive (unsigned char* data, long len, char ep)

［パラメータ］

I/O	パラメータ	説 明
I	unsigned char* data	受信データ用バッファの先頭アドレス
I	long len	データ・サイズ
i	char ep	エンドポイント番号

［機　能］　ep で指定されたエンドポイントのバッファから，len で指定されたサイズ分データを読み出し，data で指定されたアドレスに格納します。

［戻り値］　受信時のステータス

DEV_ERROR　：受信データ・サイズが不正，またはエンドポイント番号が不正
DEV_OK　：正常終了

usbf850_sendnullEP0	発行有効範囲：非タスク｜タスク

［概　要］　コントロール・エンドポイント（エンドポイント 0）用 Null データ送信関数。

［C 言語形式］void usbf850_sendnullEP0 (void)

［パラメータ］

I/O	パラメータ	説 明
–	–	–

［機　能］　コントロール・エンドポイント（エンドポイント 0）で，Null データ（データ・サイズ 0 のデータ）の送信処理を行います。

［戻り値］　なし

usb850_sendstallEP0	発行有効範囲：非タスク｜タスク

［概　要］　コントロール・エンドポイント（エンドポイント 0）用 STALL 応答関数。

［C 言語形式］void usbf850_sendstallEP0 (void)

［パラメータ］

I/O	パラメータ	説 明
–	–	–

［機　能］　コントロール・エンドポイント（エンドポイント 0）で STALL 応答を設定します。

［戻り値］　なし

usbf850_bulkin1_stall	発行有効範囲：非タスク｜タスク

［概　要］　バルク・エンドポイント（エンドポイント 1）用 STALL 応答関数。

［C 言語形式］void usbf850_bulkin1_stall (void)

［パラメータ］

I/O	パラメータ	説 明
–	–	–

［機　能］　バルク・エンドポイント（エンドポイント 1）で STALL 応答を設定します。

［戻り値］　なし

usbf850_bulkout1_stall	発行有効範囲：非タスク｜タスク

［概　要］　バルク・エンドポイント（エンドポイント 2）用 STALL 応答関数。

［C 言語形式］void usbf850_bulkout1_stall (void)

［パラメータ］

I/O	パラメータ	説 明
–	–	–

［機　能］　バルク・エンドポイント（エンドポイント 2）で STALL 応答を設定します。

［戻り値］　なし

usbf850_loc_cpu	発行有効範囲：タスク

［概　要］　マスカブル割り込みの受け付けとディスパッチ処理を禁止します。

［C 言語形式］void usbf850_loc_cpu (void)

［パラメータ］

I/O	パラメータ	説 明
–	–	–

［機　能］　loc_cpu システム・コールを呼び出します。

備考　システム・コールの詳細は，RX850 Pro ユーザーズ・マニュアル 基礎編を参照してください。

［戻り値］　なし

usbf850_unl_cpu	発行有効範囲：タスク

［概　要］　マスカブル割り込みの受け付けとディスパッチ処理を許可します。

［C 言語形式］void usbf850_unl_cpu (void)

［パラメータ］

I/O	パラメータ	説 明
–	–	–

［機　能］　unl_cpu システム・コールを呼び出します。

備考　システム・コールの詳細は，RX850 Pro **ユーザーズ・マニュアル 基礎編を参照してください。**

［戻り値］　なし

usbf850_rxreq	発行有効範囲：非タスク｜タスク

［概　要］　USB リクエスト処理を行います。

［C 言語形式］void usbf850_rxreq (void)

［パラメータ］

I/O	パラメータ	説 明
–	–	–

［機　能］　INTUSBF0 割り込み信号により起動される task_usb0b タスクによって,呼び出されます。SETUP データの読み出し処理を呼び出し，読み出したデータを解析します。解析結果に基づき，USB のリクエスト処理を呼び出します。

［戻り値］　なし

usbf850_rxreq_read	発行有効範囲：非タスク｜タスク

［概　要］　USB リクエスト・データを読み出します。

［C 言語形式］void usbf850_rxreq_read (void)

［パラメータ］

I/O	パラメータ	説 明
–	–	–

［機　能］　コントロール転送（エンドポイント 0）で，Setup トークンに続いて受信される SETUP データを読み出します。
SETUP データは，通常のデータと区別され専用のレジスタに格納されており，必ず 8 バイト・リードします。

［戻り値］　なし

usbf850_standardreq	発行有効範囲：非タスク｜タスク

［概　要］　USB 標準リクエストの処理を行います。

［C 言語形式］void usbf850_standardreq (void)

［パラメータ］

I/O	パラメータ	説 明
–	–	–

［機　能］　SETUP データ読み出し後，リクエストの内容が標準リクエストであった場合に呼び出されます。
Get Descriptor リクエストであることを確認し，usbf850_getdesc 関数を呼び出します。

［戻り値］　なし

usbf850_getdesc	発行有効範囲：非タスク｜タスク

［概　要］　USB 標準リクエストの Get Descriptor（String Descriptor）の処理を行います。

［C 言語形式］void usbf850_getdesc (void)

［パラメータ］

I/O	パラメータ	説 明
–	–	–

［機　能］　usbf850_standardreq 関数から呼び出され，USB 標準リクエストの Get Descriptor（String Descriptor）の処理を行います。

Get Descriptor（String Descriptor）リクエスト以外の場合は，STALL 応答します。

［戻り値］　なし

usbf850_sstall_ctrl	発行有効範囲：非タスク｜タスク

［概　要］　コントロール・エンドポイント（エンドポイント 0）用 STALL 応答処理関数。

［C 言語形式］void usbf850_sstall_ctrl (void)

［パラメータ］

I/O	パラメータ	説 明
–	–	–

［機　能］　コントロール・エンドポイント（エンドポイント 0）で STALL 応答を設定します。
usbf850_setfunction_storage 関数で，クラス・リクエスト処理関数を関数ポインタとして配列に準備するときに，配列の添え字にリクエスト・コードを使用します。該当するリクエストがない部分にこの関数を登録することで，サポートしないリクエスト・コードが来た場合に STALL 応答するよう設定されます。

［戻り値］　なし

usbf850_blkonly_mass_storage_reset	発行有効範囲：非タスク｜タスク

［概　要］　USB Mass Storage クラス固有のリクエスト（Bulk-Only Mass Storage Reset）処理関数。

［C 言語形式］void usbf850_blkonly_mass_storage_reset (void)

［パラメータ］

I/O	パラメータ	説 明
–	–	–

［機　能］　Bulk-Only Mass Storage Reset リクエストの処理を行います。このリクエストを受け取ると，ストレージ・デバイスの初期化処理を行います。サンプル・プログラムでは，バルク・エンドポイント（エンドポイント番号 1, 2）のバッファをクリアし，STALL 応答を設定します。

［戻り値］　なし

usbf850_max_lun	発行有効範囲：非タスク｜タスク

［概　要］　USB Mass Storage クラス固有のリクエスト（Get Max LUN）処理関数。

［C 言語形式］void usbf850_max_lun (void)

［パラメータ］

I/O	パラメータ	説 明
–	–	–

［機　能］　Get Max LUN リクエストの処理を行います。
このリクエストを受け取ると，デバイスがサポートする論理ユニットの総数を，1 バイトのデータで返します。サンプル・プログラムでは，ストレージ・デバイスが仮想デバイスであるため，00H をデータとして用意し，コントロール・エンドポイント（エンドポイント番号 0）で送信します。

［戻り値］　なし

usbf850_setfunction_storage	発行有効範囲：非タスク｜タスク

［概　要］　USB Mass Storage クラス固有のリクエスト処理関数を関数ポインタとして配列に登録する処理を行います。

［C 言語形式］void usbf850_setfunction_storage (void)

［パラメータ］

I/O	パラメータ	説 明
–	–	–

［機　能］　USB の初期化処理から呼ばれ，USB Mass Storage クラス固有のリクエスト処理関数を関数ポインタとして，配列（配列名：Req_Func_C）に登録します。
サポートしないリクエスト・コードには，usbf850_sstall_ctrl 関数を登録し，サポートしないリクエストが来た場合に STALL 応答するよう設定します。

［戻り値］　なし

usbf850_rx_cbw	発行有効範囲：非タスク｜タスク

［概　要］　CBW データの受信処理関数。

［C 言語形式］void usbf850_rx_cbw (void)

［パラメータ］

I/O	パラメータ	説 明
–	–	–

［機　能］　割り込み処理用タスクから呼び出され，CBW データを読み出します。
その後，usbf850_storage_cbwchk 関数を呼び出します。

備考　CBW の受信処理の詳細は，7. 2. 3　CBW データの処理を参照してください。

［戻り値］　なし

usbf850_storage_cbwchk	発行有効範囲：非タスク｜タスク

［概　要］　CBW データのコマンド解析処理関数。

［C 言語形式］int usbf850_storage_cbwchk (void)

［パラメータ］

I/O	パラメータ	説 明
–	–	–

［機　能］　読み出した CBW データを解析し，該当するデータ方向の処理タスクを起動します。

備考　CBW の受信処理の詳細は，7. 2. 3　CBW **データの処理**を参照してください。

［戻り値］　CBW チェック時のステータス

DEV_ERROR　：CBWCB のレングスが不正
DEV_OK　：正常終了

usbf850_cbw_error	発行有効範囲：非タスク｜タスク

［概　要］　CBW データのエラー処理関数。

［C 言語形式］void usbf850_cbw_error (void)

［パラメータ］

I/O	パラメータ	説 明
–	–	–

［機　能］　CBW のエラー検出時，バルク・エンドポイント（エンドポイント番号 1，2）に対して STALL 応答を設定します。

［戻り値］　なし

usbf850_no_data	発行有効範囲：非タスク｜タスク

［概　要］　SCSI の NO DATA 系コマンド処理タスク。

［C 言語形式］void usbf850_no_data (VP exinf)

［パラメータ］

I/O	パラメータ	説 明
I	VP	exinf 拡張情報

対象タスクに関する " ユーザ独自の情報 " を格納するための領域で，ユーザが自由に利用できます。exinf に設定された情報は，処理プログラム（タスク，非タスク）から ref_tsk システム・コールを発行することにより，ダイナミックに獲得できます。

備考　システム・コールの詳細は，RX850 Pro ユーザーズ・マニュアル 基礎編を参照してください。

［機　能］　SCSI の NO DATA 系コマンド処理用のタスクです。
scsi_command_to_ata 関数を呼び出し，実行結果から CSW 応答処理関数に CBW の処理状態（GOOD, FAIL, PHASE）を渡します。

備考　SCSI コマンドの処理についての詳細は，7. 2. 4　SCSI コマンドの処理を参照してください。

［戻り値］　なし

usbf850_data_in	発行有効範囲：非タスク｜タスク

［概　要］　SCSI の DATA IN 系コマンド処理タスク。

［C 言語形式］void usbf850_data_in (VP exinf)

［パラメータ］

I/O	パラメータ	説 明
I	VP	exinf 拡張情報

対象タスクに関する " ユーザ独自の情報 " を格納するための領域で，ユーザが自由に利用できます。exinf に設定された情報は，処理プログラム（タスク，非タスク）から ref_tsk システム・コールを発行することにより，ダイナミックに獲得できます。

備考　システム・コールの詳細は，RX850 Pro ユーザーズ・マニュアル 基礎編を参照してください。

［機　能］　SCSI の DATA IN 系コマンド処理用のタスクです。
scsi_command_to_ata 関数を呼び出し，実行結果から CSW 応答処理関数に CBW の処理状態（GOOD, FAIL, PHASE）を渡します。

備考　SCSI コマンドの処理についての詳細は，7. 2. 4　SCSI **コマンドの処理**を参照してください。

［戻り値］　なし

usbf850_data_out	発行有効範囲：非タスク｜タスク

［概　要］　SCSI の DATA OUT 系コマンド処理タスク。

［C 言語形式］void usbf850_data_out (VP exinf)

［パラメータ］

I/O	パラメータ	説 明
I	VP	exinf 拡張情報

対象タスクに関する " ユーザ独自の情報 " を格納するための領域で，ユーザが自由に利用できます。exinf に設定された情報は，処理プログラム（タスク，非タスク）から ref_tsk システム・コールを発行することにより，ダイナミックに獲得できます。

備考　システム・コールの詳細は，RX850 Pro ユーザーズ・マニュアル 基礎編を参照してください。

［機　能］　SCSI の DATA OUT 系コマンド処理用のタスクです。
scsi_command_to_ata 関数を呼び出し，実行結果から CSW 応答処理関数に CBW の処理状態（GOOD, FAIL, PHASE）を渡します。

備考　SCSI コマンドの処理の詳細は，7. 2. 4　SCSI **コマンドの処理**を参照してください。

［戻り値］　なし

usbf850_csw_ret	発行有効範囲：非タスク｜タスク

［概　要］　CSW 応答処理関数。

［C 言語形式］long usbf850_csw_ret (BYTE status)

［パラメータ］

I/O	パラメータ	説 明
I	BYTE status	送信する状態（GOOD, FAIL, PHASE）

［機　能］　CSW 応答処理用の関数です。
引き数で渡された CBW の処理状態（GOOD, FAIL, PHASE）をホストに送信します。

［戻り値］　送信時のステータス
DEV_OK ：正常終了

usbf850_dma_init	発行有効範囲：

［概　要］　DMA 初期化関数。

［C 言語形式］void usbf850_dma_init (char ep)

［パラメータ］

I/O	パラメータ	説 明
I	char ep	DMA を使用するエンドポイント番号

［機　能］　引き数で指定されたエンドポイントに対して，使用する DMA の初期化処理を行います。

［戻り値］　なし

usbf850_dma_start	発行有効範囲：非タスク｜タスク

［概　要］　DMA 起動関数。

［C 言語形式］void usbf850_dma_start (unsigned char* data, long len, char ep)

［パラメータ］

I/O	パラメータ	説 明
I	unsigned char* data	送信データ，受信データを格納するバッファのポインタ
I	long len	データ長
I	char ep	DMA を使用するエンドポイント番号

［機　能］　ep が 1 ならば，len で指定された長さ分，data で指定されたバッファから DMA で EP1_BULK_IN レジスタにデータを転送します。
ep が 2 ならば，len で指定された長さ分，data で指定されたバッファに EP1_BULK_OUT レジスタから DMA でデータを読み出します。

［戻り値］　なし

storageDev_Init	発行有効範囲：非タスク｜タスク

［概　要］　ストレージ・デバイス初期化処理関数。

［C 言語形式］void storageDev_Init (void)

［パラメータ］

I/O	パラメータ	説 明
–	–	–

［機　能］　ストレージ・デバイスの初期化処理を行います。
サンプル・プログラムでは，メモリ上にストレージ用の領域を確保しただけの仮想デバイスを扱うため，確保したメモリ領域を 0 クリアするだけです。

［戻り値］　なし

scsi_command_to_ata	発行有効範囲：非タスク｜タスク

［概　要］　SCSI コマンド処理関数。

［C 言語形式］long scsi_command_to_ata

(BYTE *ScsiCommandBuf, BYTE *pbData, long lDataSize, long TransFlag)

［パラメータ］

I/O	パラメータ	説 明
I	BYTE *ScsiCommandBuf	SCSI プロトコルの時の CBWCB
I	BYTE *pbData	各エンドポイント用データ・レジスタのアドレス
I	long lDataSize	送信／受信データ・サイズ
I	long TransFlag	転送方向

［機　能］　CBW で通知された SCSI コマンドから，コマンド処理関数を呼び出します。

［戻り値］　コマンド処理時のステータス

DEV_ERR_NODATA　：NO DATA 系コマンドで転送方向エラー
DEV_ERR_READ　：READ 系コマンドで転送方向エラー
DEV_ERR_WRITE　：WRITE 系コマンドで転送方向エラー
DEV_ERROR　：各コマンドの実行結果で上記 3 つのステータス以外，またはリクエストが不正
DEV_OK　：正常終了

ata_test_unit_ready	発行有効範囲：非タスク｜タスク

［概　要］　TEST UNIT READY コマンド処理関数。

［C 言語形式］long ata_test_unit_ready (long TransFlag)

［パラメータ］

I/O	パラメータ	説 明
I	long TransFlag	データ転送方向

［機　能］　TEST UNIT READY コマンドの処理を行います。サンプル・プログラムでは，仮想デバイスを扱うため，何もせず OK を返して終了します。

備考　SCSI コマンドの処理の詳細は，7. 2. 4　SCSI コマンドの処理を参照してください。

［戻り値］　コマンド処理時のステータス

DEV_ERR_NODATA　：NO DATA 系コマンドで転送方向エラー
DEV_OK　：正常終了

ata_seek	発行有効範囲：非タスク｜タスク

［概　要］　SEEK コマンド処理関数。

［C 言語形式］long ata_seek (long TransFlag)

［パラメータ］

I/O	パラメータ	説 明
I	long TransFlag	データ転送方向

［機　能］　SEEK コマンドの処理を行います。
サンプル・プログラムでは，仮想デバイスを扱うため，何もせず OK を返して終了します。

備考　SCSI コマンドの処理の詳細は，7. 2. 4　SCSI コマンドの処理を参照してください。

［戻り値］　コマンド処理時のステータス

DEV_ERR_NODATA　：NO DATA 系コマンドで転送方向エラー
DEV_OK　：正常終了

ata_start_stop_unit	発行有効範囲：非タスク｜タスク

［概　要］　START STOP UNIT コマンド処理関数。

［C 言語形式］long ata_start_stop_unit (long TransFlag)

［パラメータ］

I/O	パラメータ	説 明
I	long TransFlag	データ転送方向

［機　能］　START STOP UNIT コマンドの処理を行います。
サンプル・プログラムでは，仮想デバイスを扱うため，何もせず OK を返して終了します。

備考　SCSI コマンドの処理の詳細は，7. 2. 4　SCSI コマンドの処理を参照してください。

［戻り値］　コマンド処理時のステータス

DEV_ERR_NODATA　：NO DATA 系コマンドで転送方向エラー
DEV_OK　：正常終了

ata_synchronize_cache	発行有効範囲：非タスク｜タスク

［概　要］　SYNCHRONIZE CACHE コマンド処理関数。

［C 言語形式］long ata_synchronize_cache (long TransFlag)

［パラメータ］

I/O	パラメータ	説 明
I	long TransFlag	データ転送方向

［機　能］　SYNCHRONIZE CACHE コマンドの処理を行います。
サンプル・プログラムでは，仮想デバイスを扱うため，何もせず OK を返して終了します。

備考　SCSI コマンドの処理の詳細は，7. 2. 4　SCSI コマンドの処理を参照してください。

［戻り値］　コマンド処理時のステータス

DEV_ERR_NODATA　：NO DATA 系コマンドで転送方向エラー
DEV_OK　：正常終了

ata_request_sense	発行有効範囲：非タスク｜タスク

［概　要］　REQUEST SENSE コマンド処理関数。

［C 言語形式］long ata_request_sense

(BYTE *ScsiCommandBuf, BYTE *pbData, long lDataSize, long TransFlag)

［パラメータ］

I/O	パラメータ	説 明
I	BYTE *ScsiCommandBuf	SCS I プロトコルの時の CBWCB
I	BYTE *pbData	各エンドポイント用データ・レジスタのアドレス
I	long lDataSize	送信／受信データ・サイズ
I	long TransFlag	データ転送方向

［機　能］　REQUEST SENSE コマンド処理を行います。

サンプル・プログラムでは，仮想デバイスを扱うため，コマンドで指定された送信データ・サイズが 0 の場合は，何もせず OK を返して終了します。

コマンドで指定されたデータ・サイズが 0 でない場合は，そのデータ・サイズ分，REQUEST SENSE データを準備し送信します。

用意した REQUEST SENSE データのデータ長を越える場合は，用意した REQUEST SENSE データのデータ長の分だけ送信します。

備考　SCSI コマンドの処理の詳細は，7. 2. 4　SCSI コマンドの処理を参照してください。

［戻り値］　コマンド処理時のステータス

DEV_ERR_NODATA　：NO DATA 系コマンドで転送方向エラー
DEV_ERR_READ　：READ 系コマンドで転送方向エラー
DEV_OK　：正常終了

ata_inquiry	発行有効範囲：非タスク｜タスク

［概　要］　INQUIRY コマンド処理関数。

［C 言語形式］long ata_inquiry (BYTE *ScsiCommandBuf, BYTE *pbData, long lDataSize, long TransFlag)

［パラメータ］

I/O	パラメータ	説 明
I	BYTE *ScsiCommandBuf	SCSI プロトコルの時の CBWCB
I	BYTE *pbData	各エンドポイント用データ・レジスタのアドレス
I	long lDataSize	送信／受信データ・サイズ
I	long TransFlag	データ転送方向

［機　能］　INQUIRY コマンド処理を行います。

サンプル・プログラムでは，コマンドで指定されたデータ・サイズ分，INQUIRY データを準備し送信します。用意した INQUIRY データのデータ長を越える場合は，用意した INQUIRY データのデータ長の分だけ送信します。

備考　SCSI コマンドの処理の詳細は，7. 2. 4　SCSI **コマンドの処理**を参照してください。

［戻り値］　コマンド処理時のステータス

DEV_ERR_READ　：READ 系コマンドで転送方向エラー
DEV_ERROR　：リクエストが不正，または scsi_to_usb の実行結果が異常終了
DEV_OK　：正常終了

ata_mode_select	発行有効範囲：非タスク｜タスク

［概　要］　MODE SELECT（6）コマンド処理関数。

［C 言語形式］long ata_mode_select (BYTE *ScsiCommandBuf, BYTE *pbData, long lDataSize, long TransFlag)

［パラメータ］

I/O	パラメータ	説 明
I	BYTE *ScsiCommandBuf	SCS I プロトコルの時の CBWCB
I	BYTE *pbData	各エンドポイント用データ・レジスタのアドレス
I	long lDataSize	送信／受信データ・サイズ
I	long TransFlag	データ転送方向

［機　能］　MODE SELECT（6）コマンド処理を行います。

サンプル・プログラムでは，指定されたデータ・サイズ分，MODE SELECT テーブルにデータを読み込みます。用意した MODE SELECT テーブルのデータ長を越える場合は，用意した MODE SELECT テーブルのデータ長の分だけ読み込みます。

備考　SCSI コマンドの処理の詳細は，7. 2. 4　SCSI コマンドの処理を参照してください。

［戻り値］　コマンド処理時のステータス

DEV_ERR_WRITE　：WRITE 系コマンドで転送方向エラー
DEV_ERROR　：CDB の内容が不正
DEV_OK　：正常終了

ata_mode_select10	発行有効範囲：非タスク｜タスク

［概　要］　MODE SELECT（10）コマンド処理関数。

［C 言語形式］long ata_mode_select10 (BYTE *ScsiCommandBuf, BYTE *pbData, long lDataSize, long TransFlag)

［パラメータ］

I/O	パラメータ	説 明
I	BYTE *ScsiCommandBuf	SCS I プロトコルの時の CBWCB
I	BYTE *pbData	各エンドポイント用データ・レジスタのアドレス
I	long lDataSize	送信／受信データ・サイズ
I	long TransFlag	データ転送方向

［機　能］　MODE SELECT（10）コマンド処理を行います。

サンプル・プログラムでは，指定されたデータ・サイズ分，MODE SELECT（10）テーブルにデータを読み込みます。用意した MODE SELECT（10）テーブルのデータ長を越える場合は，用意した MODE SELECT（10）テーブルのデータ長の分だけ読み込みます。

備考　SCSI コマンドの処理の詳細は，7. 2. 4　SCSI コマンドの処理を参照してください。

［戻り値］　コマンド処理時のステータス

DEV_ERR_WRITE　：WRITE 系コマンドで転送方向エラー
DEV_ERROR　：CDB の内容が不正
DEV_OK　：正常終了

ata_mode_sense	発行有効範囲：非タスク｜タスク

［概　要］　MODE SENSE（6）コマンド処理関数。

［C 言語形式］long ata_mode_sense (BYTE *ScsiCommandBuf, BYTE *pbData, long lDataSize, long TransFlag)

［パラメータ］

I/O	パラメータ	説 明
I	BYTE *ScsiCommandBuf	SCS I プロトコルの時の CBWCB
I	BYTE *pbData	各エンドポイント用データ・レジスタのアドレス
I	long lDataSize	送信／受信データ・サイズ
I	long TransFlag	データ転送方向

［機　能］　MODE SENSE（6）コマンド処理を行います。

サンプル・プログラムでは，指定されたデータ・サイズ分，MODE SENSE データを準備し送信します。用意した MODE SENSE データのデータ長を越える場合は，用意した MODESENSE データのデータ長の分だけ読み込みます。

備考　SCSI コマンドの処理の詳細は，7. 2. 4　SCSI コマンドの処理を参照してください。

［戻り値］　コマンド処理時のステータス

DEV_ERR_READ　：READ 系コマンドで転送方向エラー
DEV_ERROR　：scsi_to_usb の実行結果が異常終了
DEV_OK　：正常終了

ata_mode_sense10	発行有効範囲：非タスク｜タスク

［概　要］　MODE SENSE（10）コマンド処理関数。

［C 言語形式］long ata_mode_sense10 (BYTE *ScsiCommandBuf, BYTE *pbData, long lDataSize, long TransFlag)

［パラメータ］

I/O	パラメータ	説 明
I	BYTE *ScsiCommandBuf	SCS I プロトコルの時の CBWCB
I	BYTE *pbData	各エンドポイント用データ・レジスタのアドレス
I	long lDataSize	送信／受信データ・サイズ
I	long TransFlag	データ転送方向

［機　能］　MODE SENSE（10）コマンド処理を行います。

サンプル・プログラムでは，指定されたデータ・サイズ分，MODE SENSE（10）データを準備し送信します。用意した MODE SENSE（10）データのデータ長を越える場合は，用意した MODE SENSE（10）データのデータ長の分だけ読み込みます。

備考　SCSI コマンドの処理の詳細は，7. 2. 4　SCSI コマンドの処理を参照してください。

［戻り値］　コマンド処理時のステータス

DEV_ERR_READ　：READ 系コマンドで転送方向エラー
DEV_ERROR　：scsi_to_usb の実行結果が異常終了
DEV_OK　：正常終了

ata_read_format_capacities	発行有効範囲：非タスク｜タスク

［概　要］　READ FORMAT CAPACITIES コマンド処理関数。

［C 言語形式］long ata_read_format_capacities

(BYTE *ScsiCommandBuf, BYTE *pbData, long lDataSize, long TransFlag)

［パラメータ］

I/O	パラメータ	説 明
I	BYTE *ScsiCommandBuf	SCSI プロトコルの時の CBWCB
I	BYTE *pbData	各エンドポイント用データ・レジスタのアドレス
I	long lDataSize	送信／受信データ・サイズ
I	long TransFlag	データ転送方向

［機　能］　READ FORMAT CAPACITIES コマンド処理を行います。

サンプル・プログラムでは，指定されたデータ・サイズ分，READ FORMAT CAPACITIES データを準備し送信します。用意した READ FORMAT CAPACITIES データのデータ長を越える場合は，用意した READ FORMAT CAPACITIES データのデータ長の分だけ送信します。

備考　SCSI コマンドの処理の詳細は，7. 2. 4　SCSI コマンドの処理を参照してください。

［戻り値］　コマンド処理時のステータス

DEV_ERR_READ　：READ 系コマンドで転送方向エラー
DEV_ERROR　：scsi_to_usb の実行結果が異常終了
DEV_OK　：正常終了

ata_read_capacity	発行有効範囲：非タスク｜タスク

［概　要］　READ CAPACITY コマンド処理関数。

［C 言語形式］long ata_read_capacity (BYTE *ScsiCommandBuf, BYTE *pbData, long lDataSize, long TransFlag)

［パラメータ］

I/O	パラメータ	説 明
I	BYTE *ScsiCommandBuf	SCSI プロトコルの時の CBWCB
I	BYTE *pbData	各エンドポイント用データ・レジスタのアドレス
I	long lDataSize	送信／受信データ・サイズ
I	long TransFlag	データ転送方向

［機　能］　READ CAPACITY コマンド処理を行います。

サンプル・プログラムでは，指定されたデータ・サイズ分，READ CAPACITY データを準備し送信します。用意した READ CAPACITY データのデータ長を越える場合は，用意した READ CAPACITY データのデータ長の分だけ送信します。

備考　SCSI コマンドの処理の詳細は，7. 2. 4　SCSI コマンドの処理を参照してください。

［戻り値］　コマンド処理時のステータス

DEV_ERR_READ　：READ 系コマンドで転送方向エラー
DEV_ERROR　：scsi_to_usb の実行結果が異常終了
DEV_OK　：正常終了

ata_read6	発行有効範囲：非タスク｜タスク

［概　要］　READ（6）コマンド処理関数。

［C 言語形式］long ata_read6 (BYTE *ScsiCommandBuf, BYTE *pbData, long lDataSize, long TransFlag)

［パラメータ］

I/O	パラメータ	説 明
I	BYTE *ScsiCommandBuf	SCS I プロトコルの時の CBWCB
I	BYTE *pbData	各エンドポイント用データ・レジスタのアドレス
I	long lDataSize	送信／受信データ・サイズ
I	long TransFlag	データ転送方向

［機　能］　READ（6）コマンド処理を行います。
指定された場所から指定されたサイズ分，ストレージ・デバイス上のデータを読み出します。
サンプル・プログラムでは，仮想デバイス上のデータを読み出します。

備考　SCSI コマンドの処理の詳細は，7. 2. 4　SCSI コマンドの処理を参照してください。

［戻り値］　コマンド処理時のステータス

DEV_ERR_READ　：READ 系コマンドで転送方向エラー
DEV_ERROR　：CDB の内容が不正，または scsi_to_usb の実行結果が異常終了
DEV_OK　：正常終了

ata_read10	発行有効範囲：非タスク｜タスク

［概　要］　READ（10）コマンド処理関数。

［C 言語形式］long ata_read10 (BYTE *ScsiCommandBuf, BYTE *pbData, long lDataSize, long TransFlag)

［パラメータ］

I/O	パラメータ	説 明
I	BYTE *ScsiCommandBuf	SCS I プロトコルの時の CBWCB
I	BYTE *pbData	各エンドポイント用データ・レジスタのアドレス
I	long lDataSize	送信／受信データ・サイズ
I	long TransFlag	データ転送方向

［機　能］　READ（10）コマンド処理を行います。
指定された場所から指定されたサイズ分，ストレージ・デバイス上のデータを読み出します。
サンプル・プログラムでは，仮想デバイス上のデータを読み出します。

備考　SCSI コマンドの処理の詳細は，7. 2. 4　SCSI コマンドの処理を参照してください。

［戻り値］　コマンド処理時のステータス

DEV_ERR_READ　：READ 系コマンドで転送方向エラー
DEV_ERROR　：CDB の内容が不正，または scsi_to_usb の実行結果が異常終了
DEV_OK　：正常終了

ata_write6	発行有効範囲：非タスク｜タスク

［概　要］　WRITE（6）コマンド処理関数。

［C 言語形式］long ata_write6 (BYTE *ScsiCommandBuf, BYTE *pbData, long lDataSize, long TransFlag)

［パラメータ］

I/O	パラメータ	説 明
I	BYTE *ScsiCommandBuf	SCS I プロトコルの時の CBWCB
I	BYTE *pbData	各エンドポイント用データ・レジスタのアドレス
I	long lDataSize	送信／受信データ・サイズ
I	long TransFlag	データ転送方向

［機　能］　WRITE（6）コマンド処理を行います。

指定された場所から指定されたサイズ分，ストレージ・デバイス上に受け取ったデータを書き込みます。サンプル・プログラムでは，仮想デバイス上にデータを書き込みます。

備考　SCSI コマンドの処理の詳細は，7. 2. 4　SCSI コマンドの処理を参照してください。

［戻り値］　コマンド処理時のステータス

DEV_ERR_WRITE　：WRITE 系コマンドで転送方向エラー
DEV_ERROR　：CDB の内容が不正
DEV_OK　：正常終了

ata_write10	発行有効範囲：非タスク｜タスク

［概　要］　WRITE（10）コマンド処理関数。

［C 言語形式］long ata_write10 (BYTE *ScsiCommandBuf, BYTE *pbData, long lDataSize, long TransFlag)

［パラメータ］

I/O	パラメータ	説 明
I	BYTE *ScsiCommandBuf	SCS I プロトコルの時の CBWCB
I	BYTE *pbData	各エンドポイント用データ・レジスタのアドレス
I	long lDataSize	送信／受信データ・サイズ
I	long TransFlag	データ転送方向

［機　能］　WRITE（10）コマンド処理を行います。

指定された場所から指定されたサイズ分，ストレージ・デバイス上に受け取ったデータを書き込みます。サンプル・プログラムでは，仮想デバイス上にデータを書き込みます。

備考　SCSI コマンドの処理の詳細は，7. 2. 4　SCSI コマンドの処理を参照してください。

［戻り値］　コマンド処理時のステータス

DEV_ERR_WRITE　：WRITE 系コマンドで転送方向エラー
DEV_ERROR　：CDB の内容が不正
DEV_OK　：正常終了

ata_verify	発行有効範囲：非タスク｜タスク

［概　要］　VERIFY コマンド処理関数。

［C 言語形式］long ata_verify (BYTE *ScsiCommandBuf, BYTE *pbData, long lDataSize, long TransFlag)

［パラメータ］

I/O	パラメータ	説 明
I	BYTE *ScsiCommandBuf	SCSI プロトコルの時の CBWCB
I	BYTE *pbData	各エンドポイント用データ・レジスタのアドレス
I	long lDataSize	送信／受信データ・サイズ
I	long TransFlag	データ転送方向

［機　能］　VERIFY コマンド処理を行います。
指定された場所から指定されたサイズ分，ストレージ・デバイス上のデータをチェックします。
サンプル・プログラムでは，仮想デバイスを扱うため，データのチェックは行いません。

備考　SCSI コマンドの処理の詳細は，7. 2. 4　SCSI **コマンドの処理**を参照してください。

［戻り値］　コマンド処理時のステータス

DEV_ERR_NODATA　：NO DATA 系コマンドで転送方向エラー
DEV_ERROR　：CDB の内容が不正
DEV_OK　：正常終了

ata_write_verify	発行有効範囲：非タスク｜タスク

［概　要］　WRITE VERIFY コマンド処理関数。

［C 言語形式］long ata_write_verify (BYTE *ScsiCommandBuf, BYTE *pbData, long lDataSize, long TransFlag)

［パラメータ］

I/O	パラメータ	説 明
I	BYTE *ScsiCommandBuf	SCS I プロトコルの時の CBWCB
I	BYTE *pbData	各エンドポイント用データ・レジスタのアドレス
I	long lDataSize	送信／受信データ・サイズ
I	long TransFlag	データ転送方向

［機　能］　WRITE VERIFY コマンド処理を行います。

指定された場所から指定されたサイズ分，ストレージ・デバイス上にデータを書き込み，書き込まれたデータが正しいかチェックします。サンプル・プログラムでは，仮想デバイスを扱うため，指定されたメモリ上にデータを書き込むだけで，データのチェックは行いません。

備考　SCSI コマンドの処理の詳細は，7. 2. 4　SCSI **コマンドの処理**を参照してください。

［戻り値］　コマンド処理時のステータス

DEV_OK　：正常終了

ata_write_buff	発行有効範囲：非タスク｜タスク

［概　要］　WRITE BUFF コマンド処理関数。

［C 言語形式］long ata_write_buff (BYTE *ScsiCommandBuf, BYTE *pbData, long lDataSize, long TransFlag)

［パラメータ］

I/O	パラメータ	説 明
I	BYTE *ScsiCommandBuf	SCS I プロトコルの時の CBWCB
I	BYTE *pbData	各エンドポイント用データ・レジスタのアドレス
I	long lDataSize	送信／受信データ・サイズ
I	long TransFlag	データ転送方向

［機　能］　WRITE BUFF コマンド処理を行います。

メモリ（データ・バッファ）にデータを書き込みます。

サンプル・プログラムでは，仮想デバイスを扱うため，何も処理せず正常終了します。

備考　SCSI コマンドの処理の詳細は，7. 2. 4　SCSI コマンドの処理を参照してください。

［戻り値］　コマンド処理時のステータス

DEV_OK　：正常終了

scsi_to_usb	発行有効範囲：非タスク｜タスク

［概　要］　仮想デバイスから USB 方向への送信処理関数。

［C 言語形式］long scsi_to_usb (BYTE *pbData, long TransFlag)

［パラメータ］

I/O	パラメータ	説 明
I	BYTE *pbData	送信データの先頭アドレス
I	long TransFlag	データ転送方向

［機　能］　仮想デバイスから USB 方向へのデータの送信処理を行います。

備考

［戻り値］　コマンド処理時のステータス

DEV_ERR_READ　：READ 系コマンドで転送方向エラー
DEV_OK　：正常終了

USBF850REG_SET	発行有効範囲：非タスク｜タスク

［概　要］　PFESiP/V850EP1 周辺 I/O レジスタ設定関数（1 バイト単位：8 ビット）。

［C 言語形式］USBF850REG_SET (offset, val)

［パラメータ］

I/O	パラメータ	説 明
I	offset	周辺 I/O レジスタのアドレス
I	val	設定用データ

［機　能］　PFESiP/V850EP1 周辺 I/O レジスタ（offset で指定されたレジスタ・アドレス）へ，val で指定されたデータを設定します。なお，このマクロは，1 バイト（8 ビット）単位でアクセス可能なレジスタだけ使用できます。

［戻り値］　なし

USBF850REG_READ	発行有効範囲：非タスク｜タスク

［概　要］　PFESiP/V850EP1 周辺 I/O レジスタ読み出し関数（1 バイト単位：8 ビット）。

［C 言語形式］USBF850REG_READ (offset)

［パラメータ］

I/O	パラメータ	説 明
I	offset	周辺 I/O レジスタのアドレス

［機　能］　PFESiP/V850EP1 周辺 I/O レジスタ（offset で指定されたレジスタ・アドレス）の値を読み出します。
なお，このマクロは，1 バイト（8 ビット）単位でアクセス可能なレジスタだけ使用できます。

［戻り値］　なし

USBF850REG_SET_W	発行有効範囲：非タスク｜タスク

［概　要］　PFESiP/V850EP1 周辺 I/O レジスタ設定関数（1 ワード単位：16 ビット）。

［C 言語形式］USBF850REG_SET_W (offset, val)

［パラメータ］

I/O	パラメータ	説 明
I	offset	周辺 I/O レジスタのアドレス
I	val	設定用データ

［機　能］　PFESiP/V850EP1 周辺 I/O レジスタ（offset で指定されたレジスタ・アドレス）へ，val で指定されたデータを設定します。なお，このマクロは，1 ワード（16 ビット）単位でアクセス可能なレジスタだけ使用できます。

［戻り値］　なし

USBF850REG_READ_W	発行有効範囲：非タスク｜タスク

［概　要］　PFESiP/V850EP1 周辺 I/O レジスタ読み出し関数（1 ワード単位：16 ビット）。

［C 言語形式］USBF850REG_READ_W (offset)

［パラメータ］

I/O	パラメータ	説 明
I	offset	周辺 I/O レジスタのアドレス

［機　能］　PFESiP/V850EP1 周辺 I/O レジスタ（offset で指定されたレジスタ・アドレス）の値を読み出します。
なお，このマクロは，1 ワード（16 ビット）単位でアクセス可能なレジスタだけ使用できます。

［戻り値］　なし

[メ　モ]

【発　行】

NEC**エレクトロニクス株式会社**

〒211-8668　**神奈川県川崎市中原区下沼部**1753

電話（代表）：044(435)5111

お問い合わせ先

【ホームページ】

NEC**エレクトロニクスの情報がインターネットでご覧になれます。**

URL(**アドレス**)　**http://www.necel.co.jp/**

【営業関係，技術関係お問い合わせ先】

半導体ホットライン

（電話：午前 9:00～12:00，**午後** 1:00～5:00）

電　話　：044−435−9494

E-mail　：info@necel.com

【資料請求先】

NEC**エレクトロニクスのホームページよりダウンロードいただくか，**NEC**エレクトロニクスの販売特約店へお申し付けください。**