

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情報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。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。



アプリケーション・ノート

V850シリーズ マイクロコンピュータ (USBファンクション内蔵品)

32ビット・シングルチップ・マイクロコントローラ

USB HID (ヒューマン・インタフェース・デバイス・クラス) ドライバ編

対象デバイス

V850シリーズ マイクロコントローラ USBファンクション内蔵品

資料番号 R01AN0007JJ0100(U20328JJ1V0AN00)
発行年月 May 2010

(メモ)

MINICUBEは、NECエレクトロニクス株式会社の登録商標です。

Windows, Windows XP, およびWindows Vistaは、米国Microsoft Corporationの米国およびその他の国における登録商標または商標です。

PC/ATは、米国IBM社の商標です。

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デバイスの入力にノイズなどに起因して、VIL (MAX.) からVIH (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、VIL (MAX.) からVIH (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VDD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。

はじめに

対象者 このアプリケーション・ノートは、V850シリーズ マイクロコントローラ(USBファンクション内蔵品)の機能を理解し、それを用いたアプリケーション・システムを開発しようとするユーザを対象とします。

目的 このアプリケーション・ノートは、V850シリーズ マイクロコントローラ(USBファンクション内蔵品)に内蔵のUSBファンクション・コントローラを使用するためのサンプル・ドライバの仕様をユーザに理解していただくことを目的とします。

構成 このアプリケーション・ノートは、大きく分けて次の内容で構成しています。

- ・ USB規格の概要
- ・ サンプル・ドライバ、サンプル・アプリケーションの仕様
- ・ 開発環境
- ・ サンプル・ドライバ、サンプル・アプリケーションの応用

読み方 このアプリケーション・ノートの読者には、電気、論理回路、マイクロコンピュータの一般知識を必要とします。

V850シリーズ マイクロコントローラ(USBファンクション内蔵品)のハードウェア機能、および電気的特性を知りたいとき

別冊の対象となる**V850**シリーズマイクロコントローラのユーザーズ・マニュアル ハードウェア編を参照してください。

V850シリーズ マイクロコントローラ(USBファンクション内蔵品)の命令機能を知りたいとき

別冊の**V850ES**ユーザーズ・マニュアル アーキテクチャ編、**V850E** ユーザーズ・マニュアル アーキテクチャ編を参照してください。

凡例 データ表記の重み：左側が上位桁，右側が下位桁

注：本文中につけた注の説明

注意：特に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進数または10進数 ... XXXX

16進数 ... 0xXXXX

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） ... $2^{30} = 1024^3$

T（テラ） ... $2^{40} = 1024^4$

P（ペタ） ... $2^{50} = 1024^5$

E（エクサ） ... $2^{60} = 1024^6$

関連資料 関連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。

V850シリーズマイクロコンピュータ(USBファンクション内蔵品)に関する資料

資料名	資料番号
V850ES ユーザーズ・マニュアル アーキテクチャ編	U15943J
V850E1 ユーザーズ・マニュアル アーキテクチャ編	U14559J
V850ES/JG3-H, V850ES/JH3-H ユーザーズ・マニュアル ハードウェア編	U19181J
V850シリーズ マイクロコンピュータ(USBファンクション内蔵品) アプリケーション・ノート USB HID (ヒューマン・インタフェース・デバイス・クラス) ドライバ編	このアプリケーション・ノート

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	
QB-V850ESJX3H インサーキット・エミュレータ	U19170J	
QB-V850MINI オンチップ・デバック・エミュレータ	U17638J	
QB-MINI2 フラッシュ・プログラミング機能付きオンチップ・デバック・エミュレータ	U18371J	
CA850 Ver.3.30 Cコンパイラ・パッケージ	操作編	U18512J
	C言語編	U18513J
	アセンブリ言語編	U18514J
	リンク・ディレクティブ編	U18515J
PM+ Ver.6.31 プロジェクト・マネージャ	U18416J	
ID850QB Ver.3.60 統合デバッガ	操作編	U18604J
SM850 Ver.2.50 システム・シミュレータ	操作編	U16218J
SM850 Ver.2.00以上 システム・シミュレータ	外部部品ユーザ・オープン・インタフェース仕様編	U14873J
SM+ システム・シミュレータ	操作編	U18601J
	ユーザ・オープン・インタフェース編	U18212J
	基礎編	U13430J
	インストレーション編	U17419J
	テクニカル編	U13431J
RX850 Ver.3.20 リアルタイムOS	タスク・デバッガ編	U17420J
	基礎編	U18165J
	内部構造編	U18164J
RX850 Pro Ver.3.21 リアルタイムOS	タスク・デバッガ編	U17422J
AZ850 Ver.3.30 システム・パフォーマンス・アナライザ	U17423J	
PG-FP5 フラッシュ・メモリ・プログラマ	U18865J	

目次

はじめに.....	4
第1章 概 説.....	8
1.1 概 要.....	8
1.1.1 USBファンクション・コントローラの特徴.....	8
1.1.2 サンプル・ドライバの特徴.....	9
1.1.3 サンプル・ドライバの構成.....	9
1.2 USBファンクション内蔵V850シリーズ マイクロコントローラ.....	10
1.2.1 適用製品.....	10
第2章 USBの概要.....	11
2.1 転送方式.....	11
2.2 エンドポイント.....	12
2.3 デバイス・クラス.....	12
2.4 リクエスト.....	13
2.4.1 種 類.....	13
2.4.2 フォーマット.....	15
2.5 ディスクリプタ.....	16
2.5.1 種 類.....	16
2.5.2 フォーマット.....	18
2.5.3 HIDクラス固有ディスクリプタのフォーマット.....	20
第3章 サンプル・ドライバの仕様.....	23
3.1 概 要.....	23
3.1.1 機 能.....	23
3.1.2 リクエストへの対応.....	23
3.1.3 ディスクリプタの設定.....	25
3.2 各部の動作.....	30
3.2.1 CPU初期化処理.....	31
3.2.2 USBファンクション・コントローラ初期化処理.....	33
3.2.3 INTUSBF0割り込み処理.....	37
3.3 関数の仕様.....	38
3.3.1 関数一覧.....	38
3.3.2 関数の相関関係.....	39
3.3.3 関数の機能.....	42
第4章 サンプル・アプリケーションの仕様.....	66
4.1 概 要.....	66
4.2 動 作.....	67
4.3 関数の利用.....	69
第5章 開発環境.....	72
5.1 開発環境.....	72

5.1.1	プログラム開発.....	72
5.1.2	デバッグ.....	72
5.2	環境設定.....	73
5.2.1	ホスト環境整備.....	73
5.2.2	ターゲット環境整備.....	81
5.3	オンチップ・デバッグ.....	81
5.3.1	ロード・モジュール生成.....	81
5.3.2	ロードと実行.....	82
5.4	動作確認.....	85
第6章	サンプル・ドライバの応用.....	86
6.1	概要.....	86
6.2	カスタマイズ.....	87
6.2.1	アプリケーション部.....	87
6.2.2	レジスタの設定.....	89
6.2.3	ディスクリプタの内容.....	89
6.3	関数の利用.....	89
付録A	スタータ・キット.....	90
.1	概要.....	90
.1.1	特徴.....	90
.2	主な仕様.....	91

第1章 概 説

このアプリケーション・ノートは、マイクロコントローラV850シリーズ マイクロコントローラ(USBファンクション内蔵品)のUSBファンクション・コントローラ用に作成されたUSBヒューマン・インタフェース・デバイス・クラス用サンプル・ドライバについて説明します。主に次に示す内容で構成されます。

- ・ サンプル・ドライバの仕様
- ・ サンプル・ドライバの利用、およびアプリケーション・プログラム開発のための環境

この章では、サンプル・ドライバの概要と、適用対象となるマイクロコントローラについて説明します。

1.1 概 要

1.1.1 USBファンクション・コントローラの特徴

サンプル・ドライバの制御の対象であるV850シリーズ マイクロコントローラ(USBファンクション内蔵品)のUSBファンクション・コントローラには、次のような特徴があります。

- ・ Universal Serial Bus Specification Rev.2.0に準拠
- ・ フル・スピード (12 Mbps) デバイスとして動作
- ・ エンドポイントを次のように構成

表 1-1 V850シリーズ USBファンクション内蔵マイクロコントローラのエンドポイント構成

エンドポイント名	FIFOサイズ (バイト)	転送タイプ	備 考
Endpoint0 Read	64	コントロール転送 (IN)	-
Endpoint0 Write	64	コントロール転送 (OUT)	-
Endpoint1	64 × 2	バルク転送1 (IN)	2バッファ構成
Endpoint2	64 × 2	バルク転送1 (OUT)	2バッファ構成
Endpoint3	64 × 2	バルク転送2 (IN)	2バッファ構成
Endpoint4	64 × 2	バルク転送2 (OUT)	2バッファ構成
Endpoint7	8	インタラプト転送 (IN)	-

- ・ USB標準リクエストには自動応答 (一部のリクエストを除く)
- ・ バス・パワーとセルフ・パワーを選択可能
- ・ 内部クロックと外部クロックを選択可能 ^{※1}

Jx3-H :

内部クロック : 外部6 MHz × 内部8通倍 (48 MHz)

外部クロック : UCLK端子へ入力 ($f_{USB} = 48 \text{ MHz}$)

注1. サンプル・ドライバでは内部クロックを選択します。

1.1.2 サンプル・ドライバの特徴

V850シリーズ マイクロコントローラ(USBファンクション内蔵品)向けUSBヒューマン・インタフェース・デバイス・クラス用サンプル・ドライバには、次の様な特徴があります。動作の詳細は第3章 サンプル・ドライバの仕様を参照してください。

- ・ USBヒューマン・インタフェース・デバイス・クラス（以下HIDと表す事があります） に準拠
- ・ キーボード・デバイスとして動作
- ・ Suspend/Resume機能対応
- ・ 次に示すサイズのメモリを占有（ベクタ・テーブルを除く）

V850ES/Jx3-H

ROM：約4.3 Kバイト

RAM：約0.8 Kバイト

1.1.3 サンプル・ドライバの構成

このサンプル・ドライバは次のようなファイルで構成されています。

表 1-2 サンプル・ドライバのファイル構成

フォルダ	ファイル	概要
Src	main.c	メイン・ルーチン，初期化，サンプル・アプリケーション
	usbf850.c	USB初期化，エンドポイント制御，バルク転送，コントロール転送
	usbf850_hid.c	ヒューマン・インタフェース・デバイス・クラス固有処理
include	main.h	main.c関数プロトタイプ宣言
	usbf850.h	usbf850.c関数プロトタイプ宣言
	usbf850_hid.h	usbf850_hid.c関数プロトタイプ宣言
	usbf850_desc.h	ディスクリプタ定義
	usbf850_error.h	エラー・コード定義
	usbf850_types.h	ユーザ型定義

備考 このほか，PM+（NECエレクトロニクス製統合開発ツール）で開発環境を構築した場合に生成されるプロジェクト関連ファイル一式も同梱されています。詳細は5.2.1 ホスト環境整備を参照してください。

1.2 USBファンクション内蔵V850シリーズ マイクロコントローラ

サンプル・ドライバの制御の対象であるV850シリーズ マイクロコントローラ(USBファンクション内蔵品)の詳細については該当する製品のユーザーズ・マニュアル ハードウェア編を参照してください。

1.2.1 適用製品

サンプル・ドライバは、次に示す製品に適用できます。

表 1 - 3 USBファンクション内蔵V850シリーズ マイクロコントローラ製品一覧

愛 称	品 名	内蔵メモリ		内蔵USB機能	割り込み		UM
		フラッシュ ユ・メモリ	RAM		内部 ^{注1}	外部 ^{注1}	
V850ES/JG3-H	μ PD70F3760	256 KB	40 KB	Function	69	17	ユーザーズ・マニュアル V850ES/JG3-H,V850ES/JH3 -H 32ビット・シングルチップ・マイクロコントローラ
	μ PD70F3761	384 KB	48 KB	Function	69	17	
	μ PD70F3762	512 KB	56 KB	Function	69	17	
	μ PD70F3770	256 KB	40 KB	Function	73	17	
V850ES/JH3-H	μ PD70F3765	256 KB	40 KB	Function	69	20	ハードウェア編(U19181J)
	μ PD70F3766	384 KB	48 KB	Function	69	20	
	μ PD70F3767	512 KB	56 KB	Function	69	20	
	μ PD70F3771	256 KB	40 KB	Function	73	20	

注1. ノンマスクブル割り込みを含みます。

第2章 USBの概要

この章では、サンプル・ドライバが準拠するUSB規格の概要を説明します。

USB (Universal Serial Bus) は共通のコネクタでさまざまな周辺機器をホスト・コンピュータに接続できるようにするためのインタフェース規格です。ハブと呼ばれる分岐点を追加することで最大127個の機器を接続でき、Plug&Playで機器を認識できるホットプラグに対応しているなど、従来のインタフェースより柔軟で使いやすいのが特徴です。現在ではPCのUSBインタフェース搭載率はほぼ100%になってきており、PCと周辺機器間の標準インタフェースとして定着したと言えます。

USB規格の策定と管理はUSB Implementers Forum(USB-IF)という団体が行っています。USB規格の詳細はUSB-IFの公式ウェブサイト(www.usb.org)を参照してください。

2.1 転送方式

USB規格では、4種類の転送方式(コントロール、バルク、インタラプト、アイソクロナス)が定義されています。各転送方式には表 2-1に示す特徴があります。

表 2-1 USBの転送方式

項目		転送方式	コントロール転送	バルク転送	インタラプト転送	アイソクロナス転送
特徴			周辺機器の制御などに必要な情報のやりとりに使用される転送方式	非周期的に大量データを扱う転送方式	周期的でバンド幅が低いデータ転送方式	リアルタイム性が要求される転送方式
設定可能 パケット・サイズ	ハイ・スピード 480 Mbps	64バイト	64バイト	512バイト	1-1024バイト	1-1024バイト
	フル・スピード 12 Mbps	8, 16, 32, 64バイト	8, 16, 32, 64バイト	8, 16, 32, 64バイト	1-64バイト	1-1023バイト
	ロウ・スピード 1.5 Mbps	8バイト	8バイト	-	1-8バイト	-
転送の優先順位			3	3	2	1

2.2 エンドポイント

エンドポイントはホスト・デバイスが通信相手を特定するための情報の1つで、0-15の番号と方向（IN/OUT）で指定されます。エンドポイントは周辺機器で使用するデータ通信経路ごとに用意しなければならず、複数の通信経路で共用できません。たとえば、SDカードへの書き込み/読み出しとプリント出力の機能を持った機器の場合、SDカードへの書き込み用エンドポイント、読み出し用エンドポイント、プリント出力用エンドポイントを個別に持つ必要があります。どのような機器でも必ず使用するコントロール転送には、エンドポイント0を使用します。

データ通信を行うとき、ホスト・デバイスは機器を特定するUSBデバイス・アドレスとともにエンドポイント（番号と方向）を使用して、機器内部の通信先を特定します。

エンドポイントのための物理的な回路として周辺機器内にバッファ・メモリを装備し、USBと通信先（メモリ等）の速度差を吸収するFIFOの役割も果たします。

注 オルタナティブ設定という仕組みを使い、排他的に切り替える方法があります。

2.3 デバイス・クラス

USBを介して接続する周辺機器（ファンクション・デバイス）には、その機能によりさまざまなデバイス・クラスが定義されています。代表的なクラスとしてマス・ストレージ・クラス（MSC）、コミュニケーション・デバイス・クラス（CDC）、ヒューマン・インタフェース・デバイス・クラス（HID）などがあります。各デバイス・クラスにはプロトコルなどで標準仕様が定められているため、これに準拠していれば共通のホスト・ドライバを使用できます。

ヒューマン・インタフェース・デバイス（HID）クラスは、キーボードやマウスなどの入力機器を接続するためのクラスです。HIDクラスのデバイスは、インタフェース・ディスクリプタの**InterfaceClass**フィールドが0x03になります。

詳細はHID仕様書（Device Class Definition for Human Interface Devices (HID) Specification Version 1.11）を参照してください。

2.4 リクエスト

USB規格では、ホスト・デバイスからファンクション・デバイスに対してリクエストと呼ばれるコマンドを発行することにより通信が開始されます。リクエストには処理の方向、種類、ファンクション・デバイスのアドレスなどのデータが含まれています。

2.4.1 種類

標準リクエスト、クラス・リクエスト、ベンダ・リクエストの3種類があります。

サンプル・ドライバでは次に示すリクエストに対応しています。

(1) 標準リクエスト

すべてのUSB対応機器で共通のリクエストです。

表 2-2 標準リクエスト一覧

リクエスト名	対象ディスクリプタ	概要
GET_STATUS	デバイス	電源（セルフ/バス）とリモート・ウエイクアップの設定の読み取り
	エンドポイント	Halt状態の読み取り
CLEAR_FEATURE	デバイス	リモート・ウエイクアップのクリア
	エンドポイント	Haltの解除（DATA PID = 0）
SET_FEATURE	デバイス	リモート・ウエイクアップまたはテスト・モードの設定
	エンドポイント	Haltの設定
GET_DESCRIPTOR	デバイス	対象ディスクリプタの読み取り
	コンフィギュレーション	
	ストリング	
SET_DESCRIPTOR	デバイス	対象ディスクリプタの変更（オプション）
	コンフィギュレーション	
	ストリング	
GET_CONFIGURATION	デバイス	現行設定のコンフィギュレーション値の読み取り
SET_CONFIGURATION	デバイス	コンフィギュレーション値の設定
GET_INTERFACE	インタフェース	対象インタフェースの現行設定のうちオルタナティブ設定値の読み取り
SET_INTERFACE	インタフェース	対象インタフェースのオルタナティブ設定値の設定
SET_ADDRESS	デバイス	USBアドレスの設定
SYNCH_FRAME	エンドポイント	フレーム同期のデータ読み取り

(2) クラス・リクエスト

デバイス・クラス固有のリクエストです。bmRequestTypeフィールドのビット6の値が0、ビット5の値が1のとき、そのリクエストはクラス・リクエストです。詳細はHID仕様書「Device Class Definition for Human Interface Devices (HID) Specification Version 1.11」を参照してください。

応答可能なリクエストは次のとおりです。

- ・ Get Report
ホストがコントロール転送を用いてファンクション・デバイスからデータを取得するためのリクエストです。すべてのHIDクラス準拠のドライバは、このリクエストをサポートする必要があります。
- ・ Get Idle
ホストがファンクション・デバイスの現在のアイドル率を取得するためのリクエストです。キーボード・デバイスは、このリクエストをサポートする必要があります。
- ・ Get Protocol
ホストがファンクション・デバイスの現在のプロトコル・コードを取得するためのリクエストです。ブート・デバイスは、このリクエストをサポートする必要があります。
- ・ Set Report
ホストがファンクション・デバイスにデータを送信するためのリクエストです。
- ・ Set Idle
ホストがファンクション・デバイスのアイドル率を設定するためのリクエストです。キーボード・デバイスは、このリクエストをサポートする必要があります。
- ・ Set Protocol
ホストがファンクション・デバイスのプロトコル・コードを設定するためのリクエストです。ブート・デバイスは、このリクエストをサポートする必要があります。

(3) ベンダ・リクエスト

ベンダ・リクエストは、ベンダが独自に定義するリクエストです。ベンダ・リクエストを使用する場合、ベンダはそのリクエストに対応するホスト・ドライバを提供する必要があります。bmRequestTypeフィールドのビット6の値が1、ビット5の値が0のとき、そのリクエストはベンダ・リクエストです。

2.4.2 フォーマット

USBリクエストは8バイト長で、次のようなフィールドで構成されています。

表 2 - 3 USBリクエストのフォーマット

オフセット	フィールド	説明	
0	bmRequestType	リクエストの属性	
		ビット7	データ転送方向
		ビット6, 5	リクエスト・タイプ
		ビット4-0	対象ディスクリプタ
1	bRequest	リクエスト・コード	
2	wValue	下位	リクエストで使用する任意の数値
3		上位	
4	wIndex	下位	リクエストで使用するインデックスまたはオフセット
5		上位	
6	wLength	下位	データ・ステージでの転送バイト数 (データ長)
7		上位	

2.5 ディスクリプタ

USB規格では、各ファンクション・デバイス固有の情報を定められた形式でコード化したものをディスクリプタと呼んでいます。ファンクション・デバイスは、ホスト・デバイスからのリクエストに応じてディスクリプタを送信します。

2.5.1 種類

次に示す5種類のディスクリプタが定義されています。

- ・ デバイス・ディスクリプタ

どのデバイスにも必ず存在するディスクリプタで、対応しているUSB仕様のバージョン、デバイス・クラス、プロトコル、Endpoint0に対する転送で利用可能な最大パケット長、ベンダID、プロダクトIDなどの基本情報が含まれています。

GET_DESCRIPTOR_Deviceリクエストに応答して送信するディスクリプタです。
- ・ コンフィギュレーション・ディスクリプタ

すべてのデバイスに1つ以上存在するディスクリプタで、デバイスの属性（電源供給方法）、消費電力などの情報を含みます。

GET_DESCRIPTOR_Configurationリクエストに応答して送信するディスクリプタです。
- ・ インタフェース・ディスクリプタ

インタフェースごとに必要なディスクリプタで、インタフェース識別番号、インタフェース・クラス、サポートするエンドポイントの数などが含まれます。

GET_DESCRIPTOR_Configurationリクエストに応答して送信するディスクリプタです。
- ・ エンドポイント・ディスクリプタ

インタフェース・ディスクリプタに指定されたエンドポイントごとに必要なディスクリプタで、転送タイプ（転送方向）、転送で利用可能な最大パケット長、転送のインターバルを定義します。ただし、Endpoint0はこのディスクリプタを持ちません。

GET_DESCRIPTOR_Configurationリクエストに応答して送信するディスクリプタです。
- ・ スtring・ディスクリプタ

任意の文字列を含むディスクリプタです。

GET_DESCRIPTOR_Stringリクエストに応答して送信するディスクリプタです。

また、HIDクラス規格では、次に示す3種類のディスクリプタが定義されています。

- ・ HID（ヒューマン・インタフェース・デバイス）ディスクリプタ

HIDに属するディスクリプタの種類とサイズを定義します。GET_DESCRIPTOR_HIDリクエストに応答して送信するディスクリプタです。
- ・ レポート・ディスクリプタ

ホストとファンクション・デバイス間で送受信するデータの形式を定義します。他のディスクリプタと違い必要なデータ・フィールドの内容に伴い、長さや並びが変化します。レポート・ディスクリプタはアイテムと呼ばれる情報群から構成されます。アイテムにはショート・アイテムとロング・アイテムが

あり、3つのアイテム（メイン、グローバル、ローカル）GET_DESCRIPTOR_Reportリクエストに応答して送信するディスクリプタです。

- ・ フィジカル・ディスクリプタ

ファンクション・デバイスの制御に使用する人の体の部位に関する情報を定義します。このディスクリプタはオプションのため、省略可能です。このサンプル・ドライバではフィジカル・ディスクリプタを使用しません。

2.5.2 フォーマット

ディスクリプタのサイズとフィールドは、次のように種類ごとに異なります。

備考 各フィールドのデータ並びはリトル・エンディアンです。

表 2-4 デバイス・ディスクリプタのフォーマット

フィールド	サイズ (バイト)	説 明
bLength	1	ディスクリプタのサイズ
bDescriptorType	1	ディスクリプタの種類
bcdUSB	2	USB仕様リリース番号
bDeviceClass	1	クラス・コード
bDeviceSubClass	1	サブクラス・コード
bDeviceProtocol	1	プロトコル・コード
bMaxPacketSize0	1	Endpoint0の最大パケット・サイズ
idVendor	2	ベンダID
idProduct	2	プロダクトID
bcdDevice	2	デバイスのリリース番号
iManufacturer	1	製造者を表すstring・ディスクリプタへのインデックス
iProduct	1	製品を表すstring・ディスクリプタへのインデックス
iSerialNumber	1	デバイスの製造番号を表すstring・ディスクリプタへのインデックス
bNumConfigurations	1	コンフィギュレーションの数

備考 ベンダID：USBデバイスを開発する各企業がUSB-IFから取得する識別番号

プロダクトID：ベンダIDを取得後、各企業が自社製品に割り振る識別番号

表 2-5 コンフィギュレーション・ディスクリプタのフォーマット

フィールド	サイズ (バイト)	説 明
bLength	1	ディスクリプタのサイズ
bDescriptorType	1	ディスクリプタの種類
wTotalLength	2	コンフィギュレーション、インタフェース、エンドポイント・ディスクリプタの総バイト数
bNumInterfaces	1	このコンフィギュレーションが持つインタフェースの数
bConfigurationValue	1	このコンフィギュレーションの識別番号
iConfiguration	1	このコンフィギュレーションを記述するstring・ディスクリプタへのインデックス
bmAttributes	1	このコンフィギュレーションの特徴
bMaxPower	1	このコンフィギュレーションの最大消費電流 (2 μ A単位)

表 2-6 インタフェース・ディスクリプタのフォーマット

フィールド	サイズ (バイト)	説明
bLength	1	ディスクリプタのサイズ
bDescriptorType	1	ディスクリプタの種類
bInterfaceNumber	1	このインタフェースの識別番号
bAlternateSetting	1	このインタフェースに対するオルタナティブ設定の有無
bNumEndpoints	1	このインタフェースが持つエンドポイントの数
bInterfaceClass	1	クラス・コード
bInterfaceSubClass	1	サブクラス・コード
bInterfaceProtocol	1	プロトコル・コード
iInterface	1	このインタフェースを記述するstring・ディスクリプタへのインデックス

表 2-7 エンドポイント・ディスクリプタのフォーマット

フィールド	サイズ (バイト)	説明
bLength	1	ディスクリプタのサイズ
bDescriptorType	1	ディスクリプタの種類
bEndpointAddress	1	このエンドポイントの転送方向 このエンドポイントのアドレス
bmAttributes	1	このエンドポイントの転送タイプ
wMaxPacketSize	2	この転送の最大パケット・サイズ
bInterval	1	このエンドポイントのポーリング間隔

表 2-8 string・ディスクリプタのフォーマット

フィールド	サイズ (バイト)	説明
bLength	1	ディスクリプタのサイズ
bDescriptorType	1	ディスクリプタの種類
bString	任意	任意のデータ列

2.5.3 HIDクラス固有ディスクリプタのフォーマット

HIDクラス・ディスクリプタのレポート・ディスクリプタのフォーマットは以下の通りです。

表 2 - 9 HIDクラス・ディスクリプタのフォーマット

フィールド	サイズ (バイト)	説明
bLength	1	ディスクリプタのサイズ (0x09 固定)
bDescriptorType	1	ディスクリプタの種類 (0x21 固定)
bcdHID	2	HIDバージョン (BCD 表現)
bCountryCode	1	地域識別番号
bNumDescriptors	1	クラス・ディスクリプタ数
bDescriptorType	1	クラス・ディスクリプタの種類 (1 番目)
wDescriptorLength	2	クラス・ディスクリプタのサイズ (1 番目)
[bDescriptorType]...	1	クラス・ディスクリプタの種類 (2 番目以降) (オプション)
[wDescriptorLength]...	2	クラス・ディスクリプタのサイズ (2 番目以降) (オプション)

表 2 - 10 レポート・ディスクリプタのフォーマット

ショート・アイテム																			
Bit	39/23/15	8	7	6	5	4	3	2	1	0								
Item	[data] (0 ~ 3 バイト)			bTag		bType		bSize											
ロング・アイテム																			
Bit	2041	24	23	16			15	8	7	6	5	4	3	2	1	0		
Item	[data] (0 ~ 255 バイト)				bLongItemTag				bDataSize				1	1	1	1	1	1	0
フィールド	説明																		
bTag	アイテムの内容を指定する4ビットのフィールドです。bTypeと組み合わせて使用します。ロング・アイテムのときは1111固定です。																		
bType	アイテムの種類を指定する2ビットのフィールドです。00:メイン, 01:グローバル, 10:ローカルの3種類があります。ロング・アイテムのときは11固定です。																		
bSize	バイト単位でデータ (dataフィールド) のサイズを指定する2ビットのフィールドです。ロング・アイテムのときは10 (2バイト) 固定です。																		
bLongItemTag	ロング・アイテム使用時にアイテムの内容を指定する8ビットのフィールドです。																		
bDataSize	ロング・アイテム使用時にバイト単位でデータ (dataフィールド) のサイズを指定する8ビットのフィールドです。																		

表 2 - 11 メイン・アイテム (bType = 0x00) のフォーマット

bTag	タグ名	データ	
		ビット	設 定
1000	Input	入力するデータの形式を指定します。	
		b31-b9	Reserved (0 固定)
		b8	0 : ビット単位フィールド, 1 : バイト単位のバッファ
		b7	Reserved (0 固定)
		b6	0 : NULL データを不許可, 1 : NULL データを許可
		b5	0 : 優先, 1 : 非優先
		b4	0 : 線形, 1 : 非線形
		b3	0 : ロールオーバーなし, 1 : ロールオーバーあり
		b2	0 : 絶対値, 1 : 相対値
		b1	0 : 配列, 1 : 変数
		b0	0 : データ, 1 : 定数
		1001	Output
b31-b8	Input (bTag = 1000) と同じ		
b7	0 : 変化しない値, 1 : 変化する値		
b6-b0	Input (bTag = 1000) と同じ		
1011	Feature	デバイスの構成情報を指定します。	
		b31-b0	Output (bTag = 1001) と同じ
1010	Collection	データ (Input, Output, Feature) の集合を指定します。 0x00 : Physical (座標) 0x01 : Application (マウスやキーボードなど) 0x02 : Logical (割り込みデータ) 0x03 : Report 0x04 : Named Array 0x05 : Usage Switch 0x06 : Usage Modifier 0x07-0x7F : Reserved 0x80-0xFF : Vendor-defined	
1100	End Collection	Collection の終了を指定します。データを持ちません。(bSize = 0)。	

表 2 - 12 グローバル・アイテム (bType = 0x01) のフォーマット

bTag	タグ名	データ
0000	Usage Page	アイテムの ID 番号
0001	Logical Minimum	変数および配列の値の最小値
0010	Logical Maximum	変数および配列の値の最大値
0011	Physical Minimum	可変アイテムの物理的な限界の最小値
0100	Physical Maximum	可変アイテムの物理的な限界の最大値
0101	Unit Exponent	基数を 10 としたときの指数 (2 の補数)
0110	Unit	単位
0111	Report Size	各レポートのサイズ (ビット単位)
1000	Report ID	レポートの ID 番号
1001	Report Count	レポート数
1010	Push	グローバル・アイテムの状態一覧をスタックに保管します。
1011	Pop	スタックの先頭に保管されているグローバル・アイテムの状態一覧を取り出します。

表 2 - 13 ローカル・アイテム (bType = 0x02) のフォーマット

bTag	タグ名	データ
0000	Usage	アイテムの ID 番号
0001	Usage Minimum	配列およびビットマップの開始位置
0010	Usage Maximum	配列およびビットマップの終了位置
0011	Designator Index	フィジカル・ディスクリプタの ID
0100	Designator Minimum	配列およびビットマップと関連する識別情報の開始位置
0101	Designator Maximum	配列およびビットマップと関連する識別情報の終了位置
0111	String Index	ストリング・ディスクリプタの ID
1000	String Minimum	配列およびビットマップと関連するストリング・ディスクリプタが複数ある場合の先頭 ID
1001	String Maximum	配列およびビットマップと関連するストリング・ディスクリプタが複数ある場合の最終 ID
1010	Delimiter	1 : ローカル・アイテムの開始, 0 : ローカル・アイテムの終了

第3章 サンプル・ドライバの仕様

この章では、V850シリーズ マイクロコントローラ(USBファンクション内蔵品)向けUSBヒューマン・インタフェース・デバイス・クラス用サンプル・ドライバの機能と処理内容の詳細、および実装している関数の仕様について説明します。

3.1 概要

3.1.1 機能

サンプル・ドライバには次のような処理が実装されています。

(1) 初期化

USBファンクション・コントローラを使用できるようにするため、各種レジスタを操作して設定します。大きく分けて、CPUレジスタに対する設定とUSBファンクション・コントローラのレジスタに対する設定があります。詳細は3.2.1 CPU初期化処理、3.2.2 USBファンクション・コントローラ初期化処理を参照してください。

(2) エンドポイントに対する処理

USBファンクション・コントローラ内の転送用エンドポイントの状態は、コントロール転送用エンドポイント (Endpoint0) に対してFWでデコードを行うリクエストがある事を示すINTUSB割り込み (CPUDEC割り込み) によって通知されます。必要に応じて、Endpoint0を用い、リクエスト応答も行います。詳細は、3.2.3 INTUSB割り込み処理を参照してください。また、インタラプト転送用エンドポイント (Endpoint7) でキーデータを送信する処理を行っております。

(3) サンプル・アプリケーション

HIDキーボード・デバイスとして動作します。SW押下に応じてキー割り込みが発生し、キーデータ送信処理を行います。詳細は第4章 サンプル・アプリケーションの仕様を参照してください。

3.1.2 リクエストへの対応

ここでは、サンプル・ドライバが対応するUSBリクエストについて説明します。

(1) 標準リクエスト

V850シリーズ マイクロコントローラ(USBファンクション内蔵品)が自動的に応答しないリクエストに対し、サンプル・ドライバは次のような応答処理を行います。

(a) GET_DESCRIPTOR_string

ホストがファンクション・デバイスのストリング・ディスクリプタを取得するためのリクエストです。このリクエストを受信すると、サンプル・ドライバは要求されたストリング・ディスクリプタの送信処理 (コントロール・リード転送) を行います。

(b) その他

サンプル・ドライバはSTALL応答します。

(2) クラス・リクエスト

HIDの各クラス・リクエストに対し、サンプル・ドライバは次のような応答処理を行います。

(a) Get Report

ホストがコントロール転送を用いてファンクション・デバイスからデータを取得するためのリクエストです。このリクエストを受信すると、サンプル・ドライバは記憶されているキー・コードの送信処理を行います。

(b) Get Idle

ホストがファンクション・デバイスの現在のアイドル率を取得するためのリクエストです。このリクエストを受信すると、サンプル・ドライバは現在のアイドル率 (= 0) を送信します。

(c) Set Idle

ホストがファンクション・デバイスの現在のアイドル率を取得するためのリクエストです。サンプル・ドライバのサポートしているアイドル率は「0」のみです。このリクエストで指定したアイドル率が「0」の場合、サンプル・ドライバはNULL応答を返します。指定したアイドル率が「0」以外の場合、STALL応答します。

(d) Get Protocol , Set Report , Set Protocol

これらのリクエストはサンプル・ドライバではサポートしません。これらのリクエストを受信するとサンプル・ドライバはSTALL応答します。

3.1.3 ディスクリプタの設定

サンプル・ドライバでの各ディスクリプタの設定を次に示します。各ディスクリプタの設定は、ヘッダ・ファイル "usb850_desc.h" に記述されています。

(1) デバイス・ディスクリプタ

GET_DESCRIPTOR_deviceリクエストに 응답して送信されるディスクリプタです。

GET_DESCRIPTOR_deviceリクエストにはハードウェアが自動的に 응답するため、設定内容はUSBファンクション・コントローラの初期化時にUF0DDnレジスタ (n = 0-17) に格納します。

表 3-1 デバイス・ディスクリプタの設定

フィールド	サイズ (バイト)	設定値	説明
bLength	1	0x12	ディスクリプタのサイズ: 18バイト
bDescriptorType	1	0x01	ディスクリプタの種類: デバイス
bcdUSB	2	0x0200	USB仕様リリース番号: USB 2.0
bDeviceClass	1	0x00	クラス・コード: なし(インタフェース・ディスクリプタで指定)
bDeviceSubClass	1	0x00	サブクラス・コード: なし
bDeviceProtocol	1	0x00	プロトコル・コード: 固有プロトコル未使用
bMaxPacketSize0	1	0x40	Endpoint0の最大パケット・サイズ: 64
idVendor	2	0x0409	ベンダID: NEC
idProduct	2	0x1D2	プロダクトID: V850ES/Jx3-H : 0x1D2
bcdDevice	2	0x0001	デバイスのリリース番号: 第1版
iManufacturer	1	0x01	製造者を表すstring・ディスクリプタへのインデックス: 1
iProduct	1	0x02	製品を表すstring・ディスクリプタへのインデックス: 2
iSerialNumber	1	0x03	デバイスの製造番号を表すstring・ディスクリプタへのインデックス: 3
bNumConfigurations	1	0x01	コンフィギュレーションの数: 1

(2) コンフィギュレーション・ディスクリプタ

GET_DESCRIPTOR_configurationリクエストに回答して送信されるディスクリプタです。

GET_DESCRIPTOR_configurationリクエストにはハードウェアが自動的に応答するため、設定内容はUSBファンクション・コントローラの初期化時にUF0CIEnレジスタ (n = 0-255) に格納します。

表 3-2 コンフィギュレーション・ディスクリプタの設定

フィールド	サイズ (バイト)	設定値	説明
bLength	1	0x09	ディスクリプタのサイズ：9バイト
bDescriptorType	1	0x02	ディスクリプタの種類：コンフィギュレーション
wTotalLength	2	0x0022	コンフィギュレーション、インタフェース、エンドポイント・ディスクリプタの総バイト数：34バイト
bNumInterfaces	1	0x01	このコンフィギュレーションが持つインタフェースの数：1
bConfigurationValue	1	0x01	このコンフィギュレーションの識別番号：1
iConfiguration	1	0x00	このコンフィギュレーションを記述するstring・ディスクリプタへのインデックス：0
bmAttributes	1	0xA0	このコンフィギュレーションの特徴：バス・パワー、リモート・ウエイクアップあり
bMaxPower	1	0x1B	このコンフィギュレーションの最大消費電流：54 mA

(3) インタフェース・ディスクリプタ

GET_DESCRIPTOR_configurationリクエストに回答して送信されるディスクリプタです。

GET_DESCRIPTOR_configurationリクエストにはハードウェアが自動的に応答するため、設定内容はUSBファンクション・コントローラの初期化時にUF0CIEnレジスタ (n = 0-255) に格納します。

サンプル・ドライバではインタフェースを1つ使用するため、ディスクリプタを1種類設定しています。

表 3-3 Interface0のインタフェース・ディスクリプタの設定

フィールド	サイズ (バイト)	設定値	説明
bLength	1	0x09	ディスクリプタのサイズ：9バイト
bDescriptorType	1	0x04	ディスクリプタの種類：インタフェース
bInterfaceNumber	1	0x00	このインタフェースの識別番号：0
bAlternateSetting	1	0x00	このインタフェースに対するオルタナティブ設定の有無：なし
bNumEndpoints	1	0x01	このインタフェースが持つエンドポイントの数：1
bInterfaceClass	1	0x03	クラス・コード：ヒューマン・インタフェース・デバイス・クラス
bInterfaceSubClass	1	0x00	サブクラス・コード：非ブート・デバイス
bInterfaceProtocol	1	0x01	プロトコル・コード：キーボード・デバイス
iInterface	1	0x00	このインタフェースを記述するstring・ディスクリプタへのインデックス：0

(4) エンドポイント・ディスクリプタ

GET_DESCRIPTOR_configurationリクエストに回答して送信されるディスクリプタです。

GET_DESCRIPTOR_configurationリクエストにはハードウェアが自動的に回答するため、設定内容はUSBファンクション・コントローラの初期化時にUF0CIEnレジスタ (n = 0-255) に格納します。

サンプル・ドライバではエンドポイントを1つ使用するため、ディスクリプタを1つ設定しています。

表 3 - 4 Endpoint7のエンドポイント・ディスクリプタの設定

フィールド	サイズ (バイト)	設定値	説明
bLength	1	0x07	ディスクリプタのサイズ：7バイト
bDescriptorType	1	0x05	ディスクリプタの種類：エンドポイント
bEndpointAddress	1	0x87	このエンドポイントの転送方向：IN方向 このエンドポイントのアドレス：7
bmAttributes	1	0x03	このエンドポイントの転送タイプ：インタラプト
wMaxPacketSize	2	0x0008	この転送の最大パケット・サイズ：8バイト
bInterval	1	0x0A	このエンドポイントのポーリング間隔：10 ms

(5) スtring・ディスクリプタ

GET_DESCRIPTOR_stringリクエストに回答して送信されるディスクリプタです。

GET_DESCRIPTOR_stringリクエストを受信すると、サンプル・ドライバはString・ディスクリプタをUSBファンクション・コントローラのUF0E0Wレジスタに格納します。

表 3 - 5 String・ディスクリプタの設定

(a) String 0

フィールド	サイズ (バイト)	設定値	説明
bLength	1	0x04	ディスクリプタのサイズ：4バイト
bDescriptorType	1	0x03	ディスクリプタの種類：String
bString	2	0x09, 0x04	言語コード：英語 (U.S.)

(b) String 1

フィールド	サイズ (バイト)	設定値	説明
bLength ^{注1}	1	0x2A	ディスクリプタのサイズ：42バイト
bDescriptorType	1	0x03	ディスクリプタの種類：String
bString ^{注2}	40	-	ベンダ：NEC Electronics Co.

注1. bStringフィールドのサイズにより設定値が異なります。

2. ベンダにより任意に設定できる領域のため、サイズや設定値は一定ではありません。

(c) String 2

フィールド	サイズ (バイト)	設定値	説 明
bLength ^{注1}	1	0x0E	ディスクリプタのサイズ：14バイト
bDescriptorType	1	0x03	ディスクリプタの種類：ストリング
bString ^{注2}	12	-	製品の種類：HIDDrv (HIDドライバ)

注1. bStringフィールドのサイズにより設定値が異なります。

2. ベンダにより任意に設定できる領域のため、サイズや設定値は一定ではありません。

(d) String 3

フィールド	サイズ (バイト)	設定値	説 明
bLength ^{注1}	1	0x18	ディスクリプタのサイズ：24バイト
bDescriptorType	1	0x03	ディスクリプタの種類：ストリング
bString ^{注2}	22	-	シリアル番号： V850ES/Jx3-H : 01D203000011

注1. bStringフィールドのサイズにより設定値が異なります。

2. ベンダにより任意に設定できる領域のため、サイズや設定値は一定ではありません。

(6) HIDクラス・ディスクリプタ

HID (ヒューマン・インタフェース・デバイス) ディスクリプタは、レポート・ディスクリプタおよびフィジカル・ディスクリプタの数と形式を定義するためのディスクリプタです。

GET_DESCRIPTOR_HIDリクエストを受信すると、サンプル・ドライバはHIDディスクリプタをコントロール・エンドポイントから送信します。

表 3 - 6 HIDクラス・ディスクリプタの設定

フィールド	サイズ (バイト)	設定値	説 明
bLength	1	0x09	ディスクリプタのサイズ：9バイト
bDescriptorType	1	0x21	ディスクリプタの種類：HID
bcdHID	2	0x0110	HIDバージョン (BCD 表現)
bCountryCode	1	0x00	地域識別番号：なし
bNumDescriptors	1	0x01	クラス・ディスクリプタ数：1
bDescriptorType	1	0x22	クラス従属ディスクリプタの種類：HID レポート
wDescriptorLength	2	0x002E	クラス従属ディスクリプタのサイズ：46バイト

(7) レポート・ディスクリプタ

レポート・ディスクリプタは、ホストとファンクション・デバイス間で送受信するデータ（HIDデータ）の形式（レポート・プロトコル）を定義するためのディスクリプタです。

GET_DESCRIPTOR_Reportリクエストを受信すると、サンプル・ドライバはレポート・ディスクリプタをコントロール・エンドポイントから送信します。

各アイテムの詳細については**Universal Serial Bus HID Usage Tables Version 1.12**を参照してください。

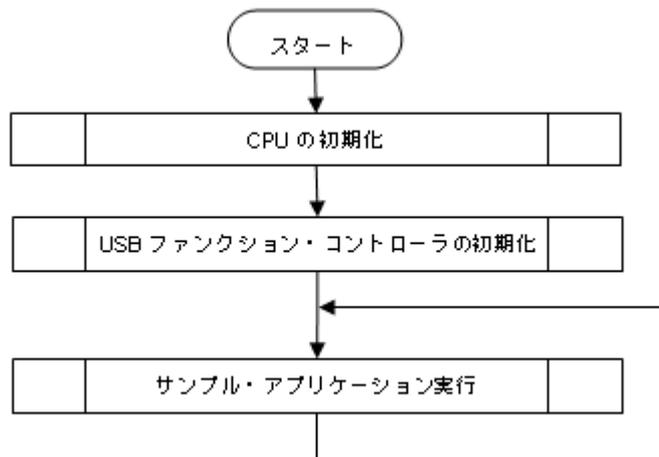
表 3-7 レポート・ディスクリプタの設定

値	アイテム	設 定
0x05, 0x01	Usage Page	Generic Desktop
0x09, 0x06	Usage	Keyboard
0xA1, 0x01	Collection	Application
0x05, 0x07	Usage Page	Keyboard
0x19, 0xE0	Usage Minimum	LEFT CTRL
0x29, 0xE7	Usage Maximum	RIGHT GUI
0x15, 0x00	Logical Minimum	0
0x25, 0x01	Logical Maximum	1
0x95, 0x08	Report Size	8 ビット
0x75, 0x01	Report Count	1
0x81, 0x02	Input	Variable
0x95, 0x01	Report Count	1
0x75, 0x08	Report Size	8 ビット
0x81, 0x01	Input	Constant
0x95, 0x06	Report Count	6
0x75, 0x08	Report Size	8 ビット
0x15, 0x00	Logical Minimum	0
0x26, 0xFF, 0x00	Logical Maximum	255
0x05, 0x07	Usage Page	Keyboard
0x19, 0x00	Usage Minimum	0
0x29, 0x91	Usage Maximum	145
0x81, 0x00	Input	-
0xC0	End Collection	-

3.2 各部の動作

サンプル・ドライバを実行すると、次のような一連の処理を行います。ここでは、それぞれの処理について説明します。サンプル・アプリケーションの詳細は第4章 サンプル・アプリケーションの仕様を参照してください。

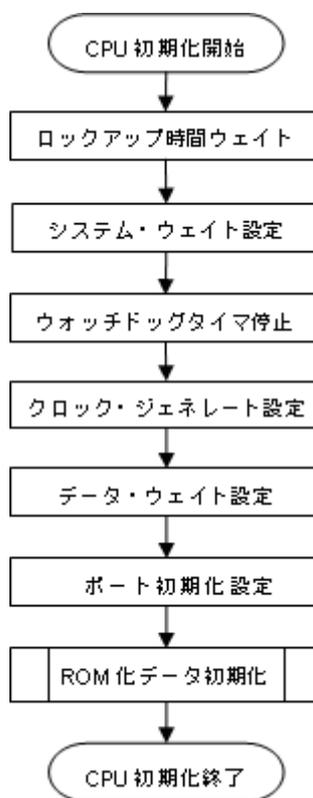
図 3-1 サンプル・ドライバの処理フロー



3.2.1 CPU初期化処理

USBファンクション・コントローラを使用するために必要な項目を設定します。

図 3 - 2 CPU初期化の処理フロー



(1) ロックアップ時間ウェイト(V850ES/Jx3-Hのみ)

CPUのクロック周波数が安定するまで（ロック状態）待機します。ここではLOCKRレジスタのLOCKビットが "0" であるかを監視します。

(2) システム・ウェイト設定

内蔵周辺I/Oレジスタに対するバス・アクセスのウェイト数を設定します。

ここではVSWCレジスタに "0x12" を書き込みます。動作周波数に応じたウェイト数を設定して下さい。

(3) ウォッチドッグ・タイマ停止(V850ES/Jx3-Hのみ)

ウォッチドッグ・タイマの動作モードを切り替えます。

ここではWDTM2レジスタのWDM21, WDM20ビットに "00" を書き込みます。この設定により、ウォッチドッグ・タイマを停止させます。

(4) クロック・ジェネレート設定

CPUの内部クロックの動作を設定します。ここでは次の3つのレジスタにアクセスします。

(a) CKCレジスタに "0x0B" を書き込みます。この設定により、内部発振回路で生成されたクロックの周波数がPLLで8週倍されます（V850ES/Jx3-Hのみ）。

(b) PLLCTLレジスタに "0x03" を書き込みます。この設定により、PLLモードが指定され、PLLの動作を開始させます。

(c) PCCレジスタに "0x00" を書き込みます。この設定により、内部クロック周波数として f_{xx} が指定されます。サンプル・ドライバではメイン・クロックでの動作を想定して設定しています。

(5) データ・ウェイト設定(V850ES/Jx3-Hのみ)

動作速度の異なる複数のI/Oそれぞれに対し、データ・アクセスのウェイト数を設定します。

ここではDWC0レジスタに "0x7717" を書き込みます。この設定により、3つのI/Oのいずれも1ウェイトでアクセスされます。

(6) ポート初期化設定(V850ES/Jx3-Hのみ)

スイッチに接続しているポートの兼用機能設定及びスイッチ押下時の割り込み信号設定を行います。

(a) ポート兼用機能の設定を行います。HIDの入力端子としてSW3(INTP12)及びSW4(INTP13)を使用します。これらはポート9の兼用端子を使用しているため、PFC9H・PMC9Hレジスタのbit0・1に1を、PFCE9Hレジスタのbit0・1に0を書き込み、INTP12・13の入力を受け付けるように設定します。

(b) 割り込み信号設定を行います。SW3及びSW4の割り込み検出は立ち下りエッジで検出するため、INTF9Hレジスタのbit0・1に1を、INTR9Hレジスタのbit0・1に0を書き込みます。

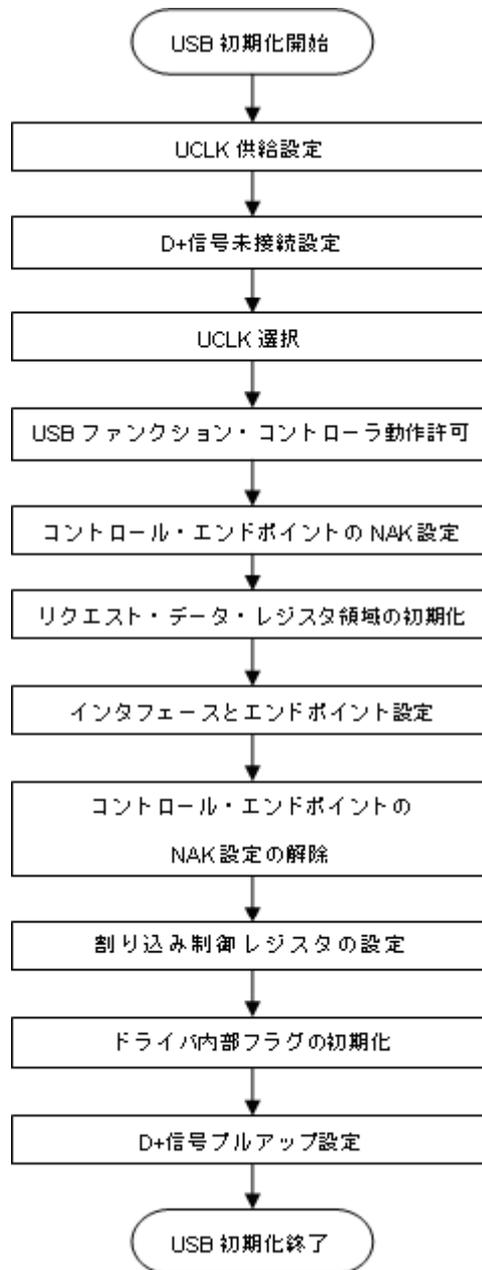
(7) ROM化データ初期化

ROM化データの初期化を行い、CPU初期化処理を終了します。

3.2.2 USBファンクション・コントローラ初期化処理

USBファンクション・コントローラの使用を開始するために必要な項目を設定します。

図 3 - 3 USBファンクション・コントローラ初期化の処理フロー



(1) D+信号プルダウン設定

V850ES/Jx3-Hでは、CPUのPM4.1ビットに"0"を、PM4.2ビットに"1"を書き込み、P4.1を出力モードにして、P4.1に"0"を設定します。これにより、D+信号をロウ・レベルの出力にして、ホスト側にデバイスの接続を検知されない様にします。

(2) UCLK選択

UCKSELレジスタに"0x02"を設定し、USBコントローラ動作クロックをUCLK端子入力からの外部クロックに設定します。

(3) USBファンクション・コントローラ動作許可

V850ES/Jx3-Hでは、UFCKMSKレジスタに"0x00"を設定します。これにより、USBファンクション・コントローラ動作許可を行います。

(4) コントロール・エンドポイントのNAK設定

ここではUF0E0NAレジスタのEP0NKAビットに "1" を書き込みます。この設定により、自動応答リクエストを含むすべてのリクエストに対してハードウェアがNAKで応答します。

このビットは、自動応答リクエストで使用するデータの登録が完了するまで、ハードウェアが自動応答リクエストに対して意図しないデータを返さないようにするために、ソフトウェアが使用します。

(5) リクエスト・データ・レジスタ領域の初期化

GET_DESCRIPTORリクエストに自動応答するためのディスクリプタ・データなどを各種レジスタに登録します。

ここでは次に示すレジスタにアクセスします。

(a) UF0DSTLレジスタに "0x02" を書き込みます。この設定により、リモート・ウエイクアップ機能の使用が許可され、USBファンクション・コントローラはバス・パワード・デバイスとして動作します。

(b) UF0EnSLレジスタ (n = 0, 7) に "0x00" を書き込みます。この設定により、Endpoint nが正常に動作していることを示します。

(c) UF0DSCLレジスタに、必要なディスクリプタのデータ長の合計 (バイト数) を書き込みます。この設定により、使用されるUF0CIEnレジスタ (n = 0-255) の範囲が決まります。

(d) UF0DDnレジスタ (n = 0-7) にデバイス・ディスクリプタのデータを書き込みます。

(e) UF0CIEnレジスタ (n = 0-255) にコンフィギュレーション・ディスクリプタ、インタフェース・ディスクリプタ、およびエンドポイント・ディスクリプタのデータを書き込みます。

(f) UF0MODCレジスタに "0x00" を書き込みます。この設定により、GET_DESCRIPTOR_configurationリクエストへの自動応答が許可されます。

(6) 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、オルタナティブ設定の状態、インタフェースとエンドポイントの関係などの情報を各種レジスタに設定します。

ここでは次に示すレジスタにアクセスします。

(a) UF0AIFNレジスタに "0x00" を書き込みます。この設定により、1つまでのインタフェースを有効にします。

(b) UF0AASレジスタに "0x00" を書き込みます。この設定により、オルタナティブ設定を無効にします。

(c) UF0E7IMレジスタに "0x20" を書き込みます。この設定により、Endpoint7がInterface0にリンクされません。

(7) コントロール・エンドポイントのNAK設定の解除

ここではUF0E0NAレジスタのEP0NKAビットに "0" を書き込みます。この設定により、自動応答リクエストを含むすべてのリクエストに対して、それぞれに応じた応答が再開されます。

(8) 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要因ごとのマスクを設定します。

ここでは次に示すレジスタにアクセスします。

- (a) UF0ICnレジスタ (n = 0-7) に "0x00" を書き込みます。この設定により、すべての割り込み要因がクリアされます。
- (b) UF0FICnレジスタ (n = 0, 1) に "0x00" を書き込みます。この設定により、すべての転送用FIFOがクリアされます。
- (c) V850ES/Jx3-Hでは、UF0IM0レジスタに "0x1D" を書き込みます。この設定により、UF0IS0レジスタに示される割り込み要因のうち、BUSRST割り込み、RSUSPD割り込み、SETRQ割り込み以外の要因がすべてマスクされます。
- (d) UF0IM1レジスタに "0x7E" を書き込みます。この設定により、UF0IS1レジスタに示される割り込み要因のうち、CPUDEC割り込み以外の要因がすべてマスクされます。
- (e) UF0IM2レジスタに "0xF1" を書き込みます。この設定により、UF0IS2レジスタに示される割り込み要因がすべてマスクされます。
- (f) UF0IM3レジスタに "0xFF" を書き込みます。この設定により、UF0IS3レジスタに示される割り込み要因がすべてマスクされます。
- (g) UF0IM4レジスタに "0x20" を書き込みます。この設定により、UF0IS4レジスタに示される割り込み要因がすべてマスクされます。
- (i) BRGINTEレジスタに"0x0003"を書き込み、EPCINT0BEN、EPCINT1BENビットが立った時の割り込みを有効にします。
- (j) UFIF0に"0"を、UFMK0に"0"を書き込み、INTUSBF0を有効にします。更に、V850ES/Jx3-Hでは、UFIF1に"0"を、UFMK1に"0"を書き込みINTUSBF1を有効にします。

(9) 割り込み制御レジスタの設定

SW3・4で使用する割り込みマスクを解除します。PMK12・PMK13に"0"を書き込み、割り込み処理を許可します。

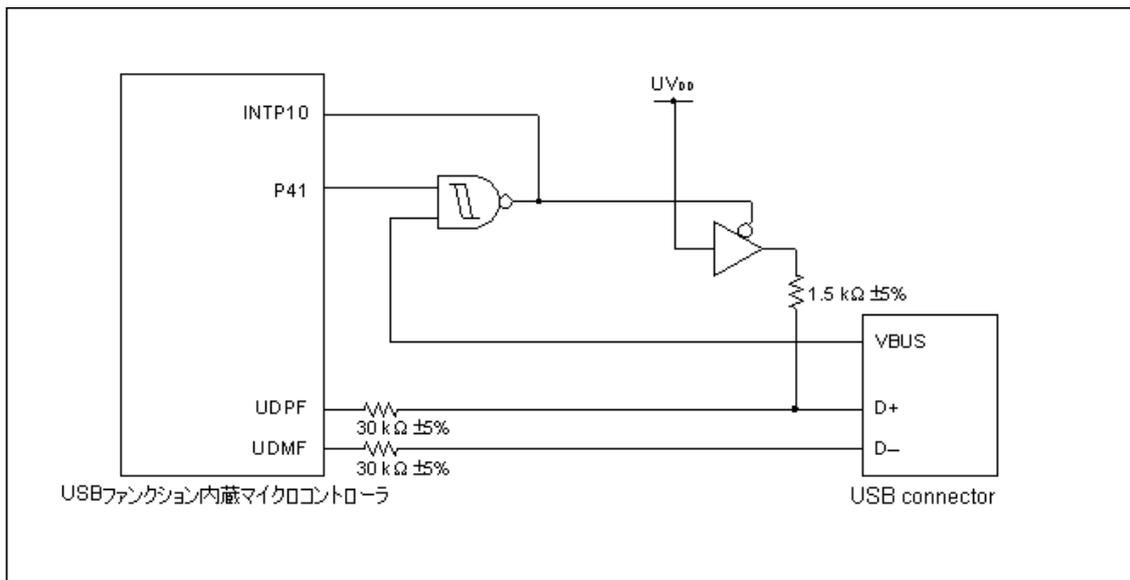
(10) ドライバ内部フラグの初期化

ドライバ内部で使用するフラグ(usb850_usbstate_flg)の初期化を行います。

(11) D+信号プルアップ設定

V850ES/Jx3-Hでは、CPUのP4レジスタに "0x02" を書き込みます。この設定により、P41から "1" が出力されます。これにより、D+信号からハイ・レベルを出力して、ホスト側にデバイスが接続されたことを通知します。サンプル・ドライバでは図 3 - 4に示すような接続を想定しています。

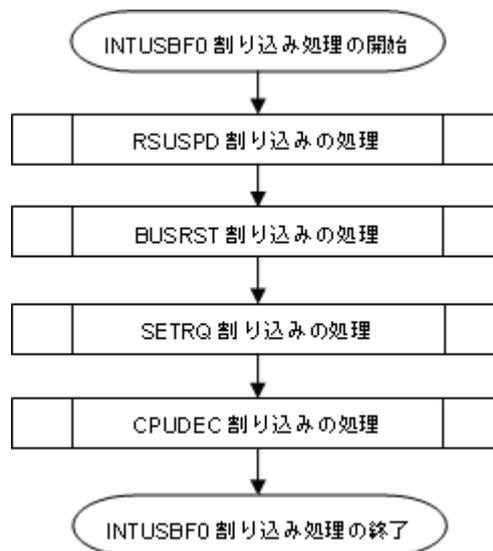
図 3 - 4 USBファンクション・コントローラ接続例



3.2.3 INTUSBF0割り込み処理

USBファンクション・コントローラからの割り込み要求 (INTUSBF0) は、初期化時にマスク解除された割り込みについてのみ通知されます。必要な割り込みについては、初期化時に割り込みマスクを解除して下さい。通知された割り込みについて、それぞれ必要な処理を行います。

図 3 - 5 Endpoint0監視の処理フロー



(1) RSUSPD割り込み処理(V850ES/Jx3-Hのみ)

UF0IS0レジスタのRSUSPDビットが "1" であれば、UF0IC0のRSUSPDCビットをクリア(0)します。次に、UF0EPS1のRSUMビットが"1"であるか確認し、更にフラグ(usb850_usbstate_flg)を確認してConfigured状態だった場合、サスペンド状態を示すフラグ(usb850_usbstate_flg)を更新(サスペンド発生)します。

(2) BUSRST割り込み処理

UF0IS0レジスタのBUSRSTビットが "1"であれば、UF0IC0のBUSRSTCビットをクリア(0)します。次に、バスリセットが発生した事を示すフラグ(usb850_usbstate_flg)を更新(バスリセット発生)し、UF0E0NAに1を書き込んでNAK応答を設定します。次にBusRestによってリフレッシュされてしまった設定を再設定します。その後UF0E0NAに0を書き込んでNAK応答設定をRESETします。

(3) SETRQ割り込み処理

UF0IS0レジスタのSETRQビットが "1"であれば、UF0IC0のSETRQCビットをクリア(0)します。次に、UF0SETレジスタのSETCONビットとUF0MODSレジスタのCONFビットが"1"だった場合、Set Configurationリクエストが処理された事を示すフラグ(usb850_usbstate_flg)を更新(Configured状態)します。

(4) CPUDEC割り込み処理

UF0IS1レジスタのCPUDECビットが "1"であれば、UF0IC1のCPUDECCビットをクリア(0)します。次に、UF0ESTレジスタを8回読み出し、リクエスト・データを取り込んでデコードします。リクエストが標準リクエストならusb850_standardreq()関数を呼び出し、クラス・リクエストならusb850_classreq()関数を呼び出します。

3.3 関数の仕様

ここでは、サンプル・ドライバに実装されている各種関数について説明します。

3.3.1 関数一覧

サンプル・ドライバでは、ソース・ファイルそれぞれに次のような関数が実装されています。

表 3 - 6 サンプル・ドライバ内の関数

ソース・ファイル	関数名	説明
main.c	main	メイン・ルーチン
	cpu_init	CPUの初期化
usbf850.c	usbf850_init	USBファンクション・コントローラの初期化
	usbf850_intusbf0	Endpoint0の監視とリクエストへの応答制御
	usbf850_intusbf1	レジューム割り込み処理
	usbf850_intkr	INTKR割り込み処理
	usbf850_stop_mode	STOPモード処理
	usbf850_standardreq	標準リクエスト処理
	usbf850_getdesc	GET_DESCRIPTOR (String、HID、Report) 処理
	usbf850_send_EP0	Endpoint0の送信処理
	usbf850_receive_EP0	Endpoint0の受信処理
	usbf850_sendnullEP0	Endpoint0のNULL/パケット送信処理
	usbf850_sendstallEP0	Endpoint0のSTALL応答処理
	usbf850_ep_status	バルク/インタラプト・イン・エンドポイントのFIFO状態通知処理
	usbf850_send_null	バルク/インタラプト・イン・エンドポイントのNULLパケット送信処理
	usbf850_data_send	バルク/インタラプト・イン・エンドポイントの送信処理
usbf850_hid.c	usbf850_classreq	HIDクラス・リクエスト処理関数
	usbf850_get_report	Get Reportリクエストの処理
	usbf850_get_idle	Get Idleリクエストの処理
	usbf850_get_protocol	Get Protocolリクエストの処理
	usbf850_set_report	Set Reportリクエストの処理
	usbf850_set_idle	Set Idleリクエストの処理
	usbf850_set_protocol	Set Protocolリクエストの処理

3.3.2 関数の相関関係

関数によっては、処理の中で別の関数を呼び出しているものもあります。関数の呼び出し関係を次に示します。

図 3-6 メイン・ルーチンでの関数の呼び出し

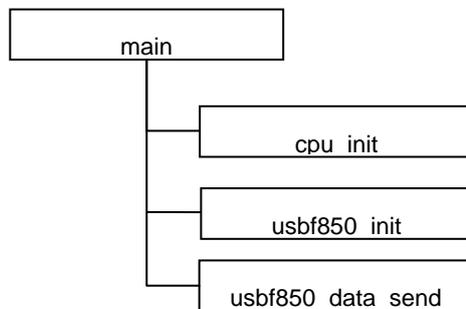


図 3 - 7 USBファンクション・コントローラ用処理での関数の呼び出し

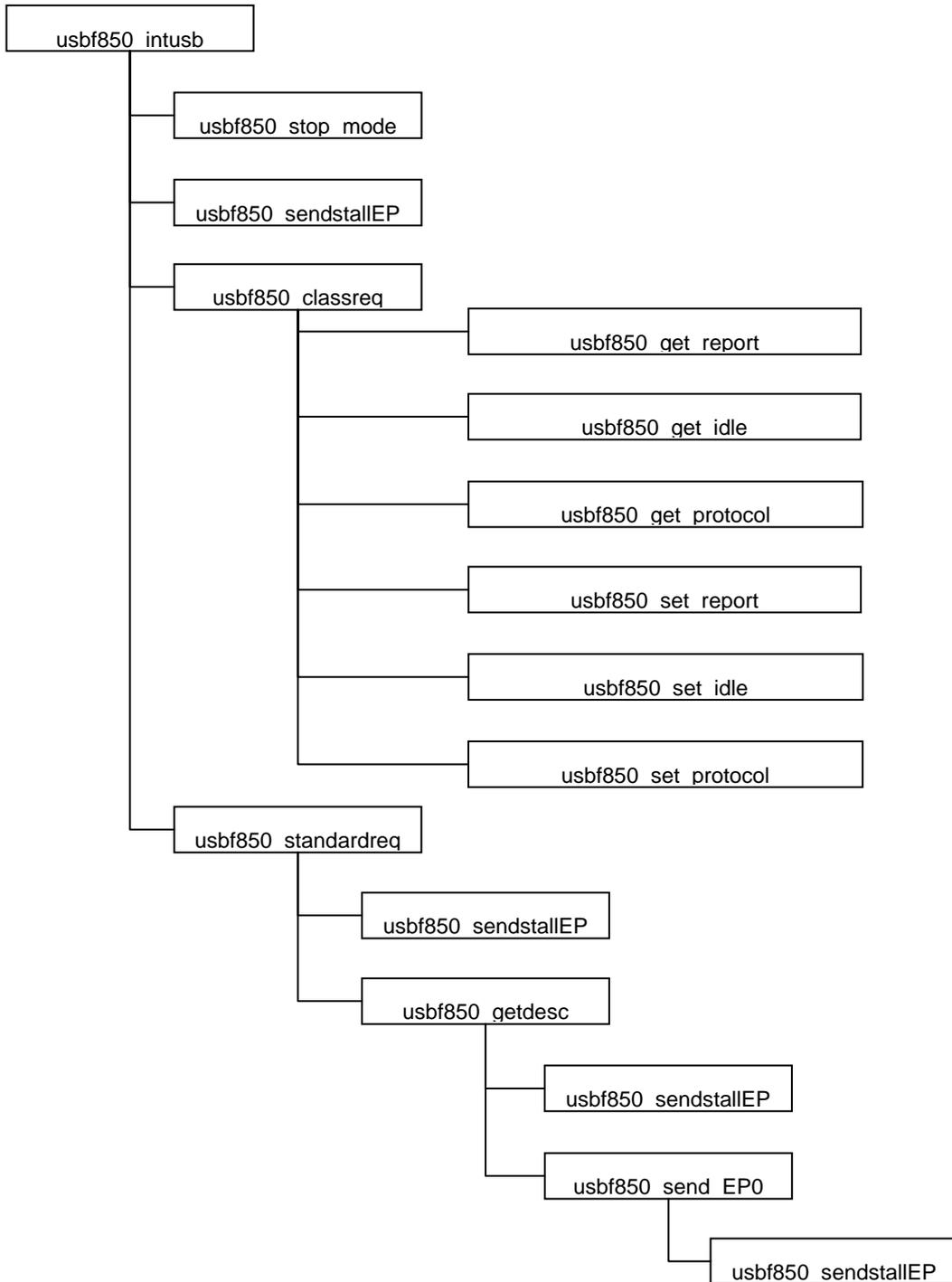
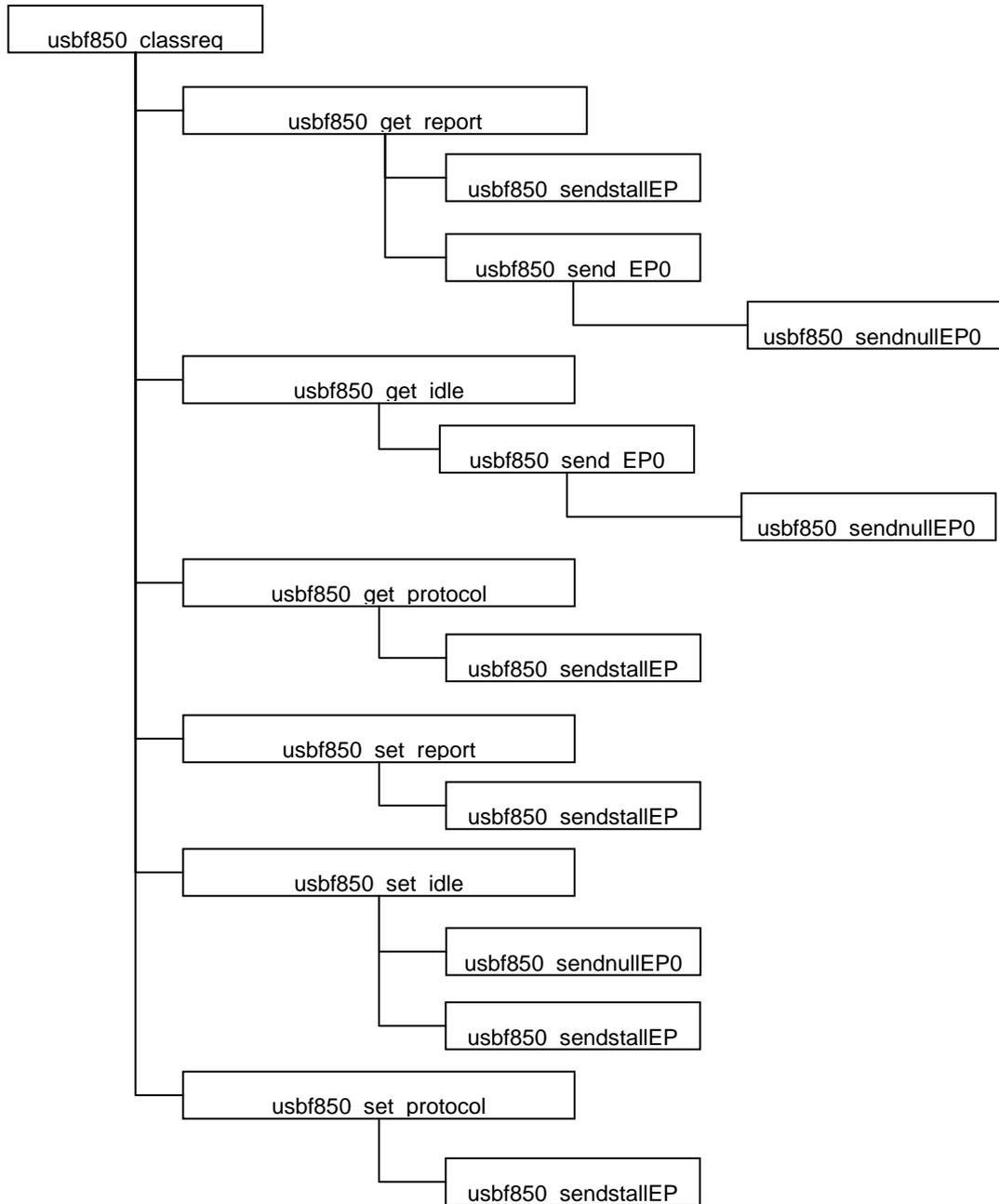


図 3 - 8 USBヒューマン・インタフェース・デバイス・クラス用処理での関数の呼び出し



3.3.3 関数の機能

ここでは、サンプル・ドライバに実装されている各種関数について解説します。

(1) 関数解説フォーマット

解説は、関数ごとに次の形式で記述されます。

関数名称

【概要】

概要説明

【C言語記述形式】

C言語上の記述形式

【パラメータ】

パラメータ(引数)の説明

パラメータ	説明
パラメータ型, 名称	パラメータ概要説明

【戻り値】

戻り値の説明

シンボル	説明
戻り値型, 名称	戻り値概要説明

【機能】

機能説明

(2) メイン・ルーチンの関数

main

【概要】

メイン処理

【C言語記述形式】

```
void main(void)
```

【パラメータ】

なし

【戻り値】

なし

【機能】

サンプル・ドライバを実行すると最初に呼び出される関数です。CPUの初期化処理、USBファンクション・コントローラの初期化処理、および、サンプル・アプリケーション処理を順に実行します。

cpu_init

【概要】

CPU初期化処理

【C言語記述形式】

```
void cpu_init(void)
```

【パラメータ】

なし

【戻り値】

なし

【機能】

初期化処理で呼び出される関数です。

バス・アクセス時のウェイト数やクロック周波数、動作モードなど、USBファンクション・コントローラを使用するために必要な項目等を設定します。

(3) USBファンクション・コントローラ用処理の関数

usb850_init

【概要】

USBファンクション・コントローラ初期化処理

【C言語記述形式】

```
void usb850_init(void)
```

【パラメータ】

なし

【戻り値】

なし

【機能】

初期化処理で呼び出される関数です。

データ領域の確保と設定，割り込み要求のマスクなど，USBファンクション・コントローラの使用を開始するために必要な項目を設定します。

usbfs850_intusbfs0

【概要】

Endpoint0監視処理

Endpoint2監視処理

【C言語記述形式】

```
void usbfs850_intusbfs0(void)
```

【パラメータ】

なし

【戻り値】

なし

【機能】

INTUSBFS0割り込みにより呼び出される割り込みサービスルーチンです。

USBファンクション・コントローラのマスクされていない割り込みについて、割り込み内容を確認し、発生している割り込みについて処理を行います。

本サンプル・ドライバでは、Suspend発生時にusbfs850_stop_mode()関数を呼び出します。

usb850_intusb1

【概要】

Resume割り込み処理

【C言語記述形式】

```
void usb850_intusb1(void)
```

【パラメータ】

なし

【戻り値】

なし

【機能】

INTUSB1割り込みにより呼び出される割り込みサービスルーチンです。

USBファンクション・コントローラのマスクされていない割り込みについて、割り込み内容を確認し、発生している割り込みについて処理を行います。

本サンプル・ドライバではResume割り込み発生時に呼び出され、フラグ操作のみ処理を行います。

usb850_intkr

【概要】

INTKR割り込み処理

【C言語記述形式】

```
void usb850_intkr(void)
```

【パラメータ】

なし

【戻り値】

なし

【機能】

INTKR割り込みにより呼び出される割り込みサービスルーチンです。
キーを押された事を示すフラグを更新します。

usb850_stop_mode**【概要】**

Suspend時にStopモードに入る為の処理

【C言語記述形式】

```
void usb850_stop_mode(void)
```

【パラメータ】

なし

【戻り値】

なし

【機能】

INTUSB割り込み処理のCPUDEC割り込み要因の処理から呼び出される関数です。

本サンプル・ドライバでは、Suspend発生時にusb850_intusb0()割り込みサービスルーチンより呼び出され、そのままUSBクロック等を停止してSTOPモードに入り、CPU動作を停止します。INTRSUM割り込み、INTKR割り込みによりSTOPモードは解除され、再度USBクロック供給等を再開して関数を抜けます。

usbfs850_standardreq**【概要】**

USBファンクション・コントローラが自動応答しない標準リクエストの処理

【C言語記述形式】

```
void usbfs850_standardreq(USB_SETUP *req_data)
```

【パラメータ】

パラメータ	説明
USB_SETUP *req_data	リクエストデータ格納ポインタ・アドレス

【戻り値】

なし

【機能】

INTUSB割り込み処理のCPUDEC割り込み要因の処理から呼び出される関数です。

デコードされたリクエストがGET_DESCRIPTORの場合、GET_DESCRIPTORリクエスト処理関数 (usbfs850_getdesc) を呼び出します。それ以外のリクエストの場合は、Endpoint0用STALL応答処理関数 (usbfs850_sendstallEP0) を呼び出します。

usb850_getdesc**【概要】**

GET_DESCRIPTORリクエスト処理

【C言語記述形式】

```
void usb850_getdesc(USB_SETUP *req_data)
```

【パラメータ】

パラメータ	説明
USB_SETUP *req_data	リクエストデータ格納ポインタ・アドレス

【戻り値】

なし

【機能】

USBファンクション・コントローラが自動応答しない標準リクエストの処理で呼び出される関数です。デコードされたリクエストがistring・ディスクリプタを要求している場合、Endpoint0用USBデータ送信関数 (usb850_send_EP0) を呼び出して、Endpoint0からistring・ディスクリプタを送信させます。同様にHIDディスクリプタ及びレポート・ディスクリプタの要求の場合も送信処理を行います。それ以外のディスクリプタを要求している場合は、Endpoint0用STALL応答処理関数 (usb850_sendstallEP0) を呼び出します。

usbfs850_send_EP0**【概要】**

Endpoint0用USBデータ送信処理

【C言語記述形式】

```
void usbfs850_send_EP0(UINT* data, INT32 len)
```

【パラメータ】

パラメータ	説明
UINT* data	送信データ・バッファ・ポインタ
INT32 len	送信データサイズ

【戻り値】

シンボル	説明
DEV_OK	正常終了
DEV_ERROR	異常終了

【機能】

送信データ・バッファに格納されているデータをEndpoint0用送信FIFOに1バイトずつ格納します。

usbfs850_receive_EP0**【概要】**

Endpoint0用USBデータ受信処理

【C言語記述形式】

```
INT32 usbfs850_receive_EP0(UINT* data, INT32 len)
```

【パラメータ】

パラメータ	説明
UINT* data	受信データ・バッファ・ポインタ
INT32 len	受信データサイズ

【戻り値】

シンボル	説明
DEV_OK	正常終了
DEV_ERROR	異常終了

【機能】

Endpoint0用受信FIFOから1バイトずつ読み出し、受信データ・バッファに格納します。

usbfs850_sendnullEP0

【概要】

Endpoint0用NULLパケット送信処理

【C言語記述形式】

```
void usbfs850_sendnullEP0(void)
```

【パラメータ】

なし

【戻り値】

なし

【機能】

Endpoint0用のFIFOをクリアし、データ終了を示すビットをセット(1)することで、USBファンクション・コントローラからNULLパケットを送信させます。

usbfs850_sendstallEP0

【概要】

Endpoint0用STALL応答処理

【C言語記述形式】

```
void usbfs850_sendstallEP0(void)
```

【パラメータ】

なし

【戻り値】

なし

【機能】

STALLハンドシェイク使用を示すビットをセット(1)することで、USBファンクション・コントローラからSTALL応答させます。

usbfs850_ep_status**【概要】**

Bulk/ Interrupt In Endpoint用FIFO状態通知処理

【C言語記述形式】

```
INT32 usbfs850_ep_status(INT8 ep)
```

【パラメータ】

パラメータ	説明
INT8 ep	データ送信Endpoint番号

【戻り値】

シンボル	説明
DEV_OK	正常終了
DEV_RESET	Bus Reset処理中
DEV_ERROR	異常終了

【機能】

指定されたEndpoint(送信用)のFIFO状態を通知します。

usbfs850_send_null**【概要】**

バルク / インタラプト・イン・エンドポイント用NULLパケット送信処理

【C言語記述形式】

```
INT32 usbfs850_send_null(INT8 ep)
```

【パラメータ】

パラメータ	説明
INT8 ep	データ送信Endpoint番号

【戻り値】

シンボル	説明
DEV_OK	正常終了
DEV_ERROR	異常終了

【機能】

指定されたエンドポイント(送信用)のFIFOをクリアし、データ終了を示すビットをセット(1)することで、USBファンクション・コントローラからNULLパケットを送信させます。

usbfs850_data_send

【概要】

バルク / インタラプト・イン・エンドポイント用USBデータ送信処理

【C言語記述形式】

```
INT32 usbfs850_data_send(UINT8* data, INT32 len, INT8 ep)
```

【パラメータ】

パラメータ	説明
UINT8* data	送信データ・バッファ・ポインタ
INT32 len	送信データサイズ
INT8 ep	データ送信Endpoint番号

【戻り値】

パラメータ	説明
len (>= 0)	正常に送信したデータサイズ
DEV_ERROR	異常終了

【機能】

送信データ・バッファに格納されているデータを指定されたエンドポイント用の送信FIFOに1バイトずつ格納します。

(4) USBヒューマン・インタフェース・デバイス・クラス用処理の関数

usb850_classreq

【概要】

クラス・リクエストの処理

【C言語記述形式】

```
void usb850_classreq(USB_SETUP *req_data)
```

【パラメータ】

パラメータ	説明
USB_SETUP *req_data	リクエストデータ格納ポインタ・アドレス

【戻り値】

なし

【機能】

INTUSB割り込み処理のCPUDEC割り込み要因の処理から呼び出される関数です。

デコードされたリクエストがヒューマン・インタフェース・デバイス・クラスのリクエストの場合、各リクエスト処理関数を呼び出します。それ以外のリクエストの場合は、Endpoint0用STALL応答処理関数（usb850_sendstallEP0）を呼び出します。

usb850_get_report**【概要】**

Get Reportリクエスト処理

【C言語記述形式】

```
void usb850_get_report(USB_SETUP *req_data)
```

【パラメータ】

パラメータ	説明
USB_SETUP *req_data	リクエストデータ格納ポインタ・アドレス

【戻り値】

なし

【機能】

クラス・リクエストの処理でデコードされたリクエストがGet Reportリクエストだった場合、呼び出される関数です。レポートIDが0のリクエストを受信した場合のみ、現在記録されているキー・コードをEndpoint0から送信します。それ以外は、STALL応答します。

usbf850_get_idle**【概要】**

Get Idleリクエスト処理

【C言語記述形式】

```
void usbf850_get_idle(USB_SETUP *req_data)
```

【パラメータ】

パラメータ	説明
USB_SETUP *req_data	リクエストデータ格納ポインタ・アドレス

【戻り値】

なし

【機能】

クラス・リクエストの処理でデコードされたリクエストがGet Idleだった場合、呼び出される関数です。アイドル率をEndpoint0から送信します。サンプル・ドライバのアイドル率は0固定（変化があったときだけ送信）です。

usb850_get_protocol**【概要】**

Get Protocolリクエスト処理

【C言語記述形式】

```
void usb850_get_protocol(USB_SETUP *req_data)
```

【パラメータ】

パラメータ	説明
USB_SETUP *req_data	リクエストデータ格納ポインタ・アドレス

【戻り値】

なし

【機能】

クラス・リクエストの処理でデコードされたリクエストがGet Protocolだった場合、呼び出される関数です。STALL応答します。

usb850_set_report**【概要】**

Set Reportリクエスト処理

【C言語記述形式】

```
void usb850_set_report(USB_SETUP *req_data)
```

【パラメータ】

パラメータ	説明
USB_SETUP *req_data	リクエストデータ格納ポインタ・アドレス

【戻り値】

なし

【機能】

クラス・リクエストの処理でデコードされたリクエストがSet Reportだった場合、呼び出される関数です。STALL応答します。

usbf850_set_idle**【概要】**

Set Idleリクエスト処理

【C言語記述形式】

```
void usbf850_set_idle(USB_SETUP *req_data)
```

【パラメータ】

パラメータ	説明
USB_SETUP *req_data	リクエストデータ格納ポインタ・アドレス

【戻り値】

なし

【機能】

クラス・リクエストの処理でデコードされたリクエストがSet Reportだった場合、呼び出される関数です。ホストが送信したアイドル率が0の時、NULLパケットを送信し、それ以外の時には、STALL応答します。

usb850_set_protocol**【概要】**

Set Protocolリクエスト処理

【C言語記述形式】

```
void usb850_set_protocol(USB_SETUP *req_data)
```

【パラメータ】

パラメータ	説明
USB_SETUP *req_data	リクエストデータ格納ポインタ・アドレス

【戻り値】

なし

【機能】

クラス・リクエストの処理でデコードされたリクエストがSet Protocolだった場合、呼び出される関数です。STALL応答します。

第4章 サンプル・アプリケーションの仕様

この章では、サンプル・ドライバに含まれるサンプル・アプリケーションについて説明します。

4.1 概 要

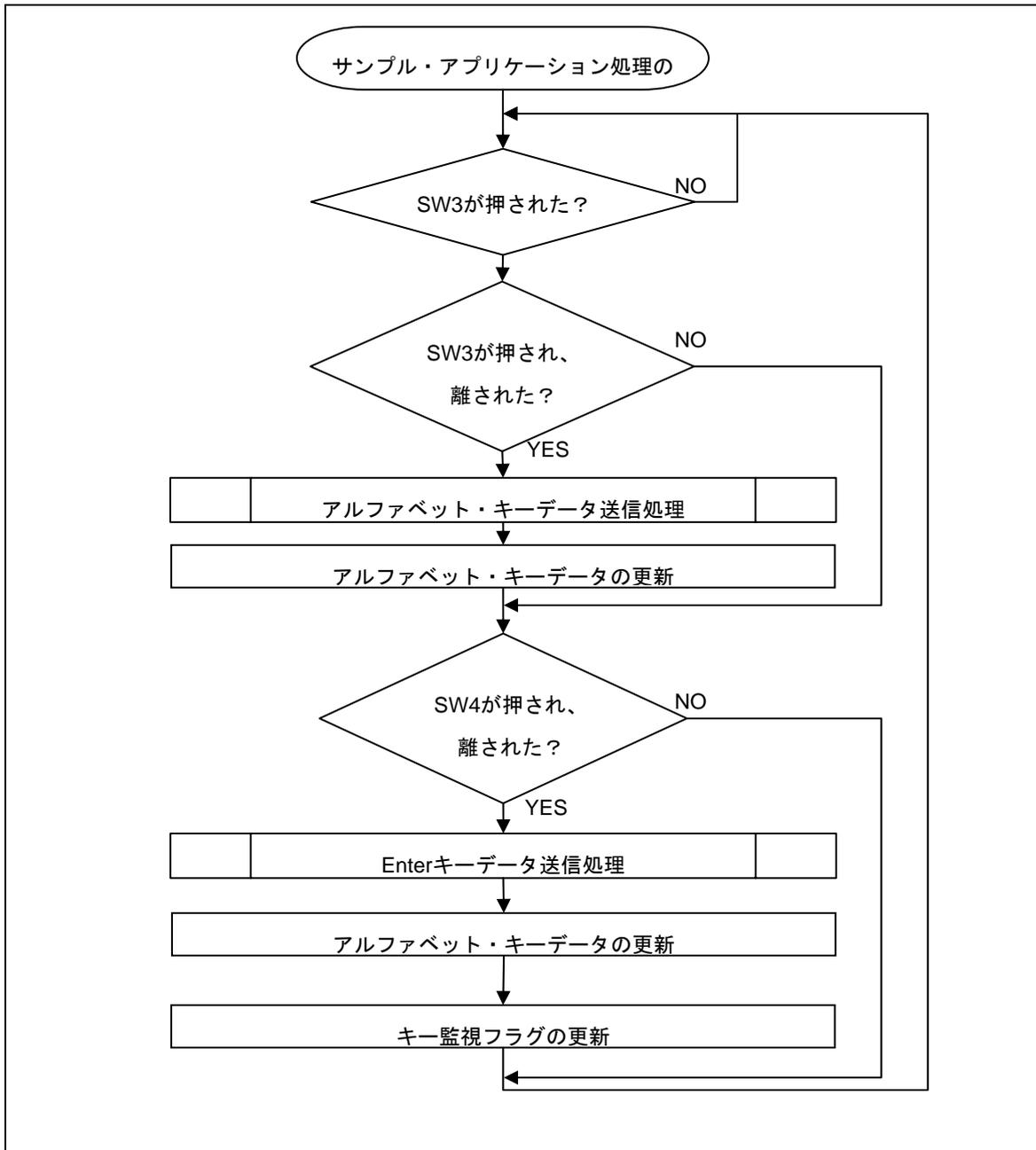
サンプル・アプリケーションは、USBヒューマン・インタフェース・デバイス・クラス用ドライバの利用の簡単な例として用意されたもので、サンプル・ドライバのメイン・ルーチンに組み込まれています。

このサンプル・アプリケーションは、TK-850/JG3H ボード上のスイッチSW3が押された時にアルファベット「a」～「z」を送信、SW4が押された時に「Enter」を送信する処理を実行します。この処理の過程で、サンプル・ドライバの各種関数を利用します。

4.2 動作

サンプル・アプリケーションでは、次のようなフローで一連の処理が行われます。

図 4-1 サンプル・アプリケーションの処理フロー



(1)SW3が押されたかを確認

TK-850/JG3H ボード上のスイッチSW3が押されたかを、キー押下確認用フラグ (usbf850_keytouch_flg) によって確認し、フラグの値が " 1 " であれば、キーデータ送信処理へと進み、フラグの値が " 0 " であれば何もしません。

(2)SW3が押されたのかの確認

P9Hのビット0が “ 0 ” であれば、SW3が押されたと判断し、離されるまで待ちます。

(3)アルファベット・キーデータ送信処理

アルファベット・キーデータとリリース・キーデータをusb850_data_send関数で送信します。

(4)アルファベット・キーデータ情報更新

アルファベット・キーデータを更新（トグル）します。キーデータが「z」であれば「a」にします。

(5)SW4が押されたのかの確認

P9Hのビット1が “ 0 ” であれば、SW 4 が押されたと判断し、離されるまで待ちます。

(6)Enterキーデータ送信処理

Enterキーデータとリリース・キーデータをusb850_data_send関数で送信します。

(7)アルファベット・キーデータ情報更新

アルファベット・キーデータを更新（「a」に戻す）します。

(8)キー押下確認用フラグ更新

キー押下確認用フラグ（usb850_keytouch_flg）を更新（ “ 0x00 ” に設定）します。

4.3 関数の利用

このサンプル・アプリケーションを含むソース・ファイル "main.c" では、サンプル・ドライバの関数を利用するために次のように記述しています。なお、各関数の詳細は**3.3 関数の仕様**を参照してください。

リスト 4-1 サンプル・アプリケーションの記述 (部分)

```

1  #pragma ioreg
2  #include "main.h"
3  #include "usb850_errno.h"
4  #include "usb850.h"
5  #include "usb850_hid.h"
6
7  int _rcopy(unsigned long *, long);
8  extern unsigned long _S_romp;
9
10 extern UINT8    usb850_keytouch_flg;
11      :
12      中略
13      :
14 void main(void)
15 {
16     UINT8 keycode[REPORT_DATA_LENGTH];
17     UINT8 keydata = A_KEY;
18
19     cpu_init();
20
21     usb850_init();      /* initial setting of the USB Function */
22
23     EI();
24
25     memset(keycode, 0, sizeof(keycode));      /* Report Data Clear */
26
27     while (1) {
28         if( usb850_keytouch_flg == F_SW_ON ){
29             if( !(UF0EPS0 & C_IT1) ){
30                 /* SW3 */
31                 if((P9H & 0x01) == 0x00) {
32                     while((P9H & 0x01) == 0x00){
33                         /* SW3 OFF WAIT */
34                     }
35                     keycode[KEY1_SCAN_CODE] = keydata;      /* Press Key Data */
36                     usb850_data_send(keycode, sizeof(keycode), C_INT1);      . . . (4)
37                     memset(keycode, 0, sizeof(keycode));      /* Release Key Data */
38                     usb850_data_send(keycode, sizeof(keycode), C_INT1);      . . . (4)
39                     keydata++;      /* a to z */
40                     if(keydata == EXCLAMATION_KEY) {
41                         keydata = A_KEY;
42                     }
43                 }
44                 /* SW4 */
45                 if((P9H & 0x02) == 0x00) {
46                     while((P9H & 0x02) == 0x00){
47                         /* SW4 OFF WAIT */
48                     }
49                     keycode[KEY1_SCAN_CODE] = ENTER_KEY; /* Press Key Data(Enter) */
50                     usb850_data_send(keycode, sizeof(keycode), C_INT1);      . . . (4)
51                     memset(keycode, 0, sizeof(keycode));      /* Release Key Data */
52                     usb850_data_send(keycode, sizeof(keycode), C_INT1);      . . . (4)
53                     keydata = 0x04;
54                 }
55             }
56             usb850_keytouch_flg = F_SW_OFF;
57         }
58     }
59 }

```

(1)各種定義と宣言

サンプル・ドライバの関数を使用するため、"usb850r.h" ヘッダ・ファイルをインクルードしています。
また、キーデータ格納用配列 (keycode) を用意しています。

(2)CPU初期化処理

CPU初期化処理関数 (cpu_init) を呼び出します。

(3)USBファンクション・コントローラ初期化処理

USBファンクション・コントローラ初期化処理関数 (usb850_init) を呼び出します。

(4)データ送信処理

データ、データのサイズ、エンドポイントを指定して、データ送信関数 (usb850_data_send) でデータを送信します。

第5章 開発環境

この章では、V850シリーズ マイクロコンピュータ(USBファンクション内蔵品)向けUSB HIDサンプル・ドライバを利用したアプリケーション・プログラムを開発する際の環境構築例と、デバッグの手順について説明します。

5.1 開発環境

ここでは、ハードウェア・ツールとソフトウェア・ツールの製品構成例を示します。

5.1.1 プログラム開発

サンプル・ドライバを利用したシステムを開発する際は、次の様なハードウェアとソフトウェアが必要です。

表 5 - 1 プログラム開発環境構成例

構成部品		製品例	備考
ハードウェア	ホスト・マシン	-	PC/AT™互換機 (OS : Windows XP)
ソフトウェア	統合開発ツール	PM+	V6.31
	コンパイラ	CA850	W3.30

5.1.2 デバッグ

サンプル・ドライバを利用したシステムをデバッグする際は、次の様なハードウェアとソフトウェアが必要です。

表 5 - 2 デバッグ環境構成例

構成部品		製品例	備考
ハードウェア	ホスト・マシン	-	PC/AT互換機 (OS : Windows XP)
	ターゲット	TK-850/JG3H	テセラ・テクノロジー社製
		TK-850/JH3U-SP TK-850/JH3E+NET	テセラ・テクノロジー社製 テセラ・テクノロジー社製
	USBケーブル	-	miniBコネクタ - Aコネクタ
ソフトウェア	統合開発ツール	PM+	V6.31
	デバッガ	ID850QB	V3.60
ファイル	デバイス・ファイル	DF703771	V850ES/JG3-H用
	デバッグ・ポート用ホスト・ドライバ	-	注1
	プロジェクト関連ファイル	-	注2

注1. 製品や入手方法については弊社までお問い合わせください。

2. PM+で構築した場合のファイルがサンプル・ドライバに同梱されています。

5.2 環境設定

ここでは、5.1 開発環境に示した製品構成で開発やデバッグを行うための準備について、TK-850/JG3Hでの環境設定を例にして説明します。

5.2.1 ホスト環境整備

ホスト・マシン上に専用のワークスペースを作成します。

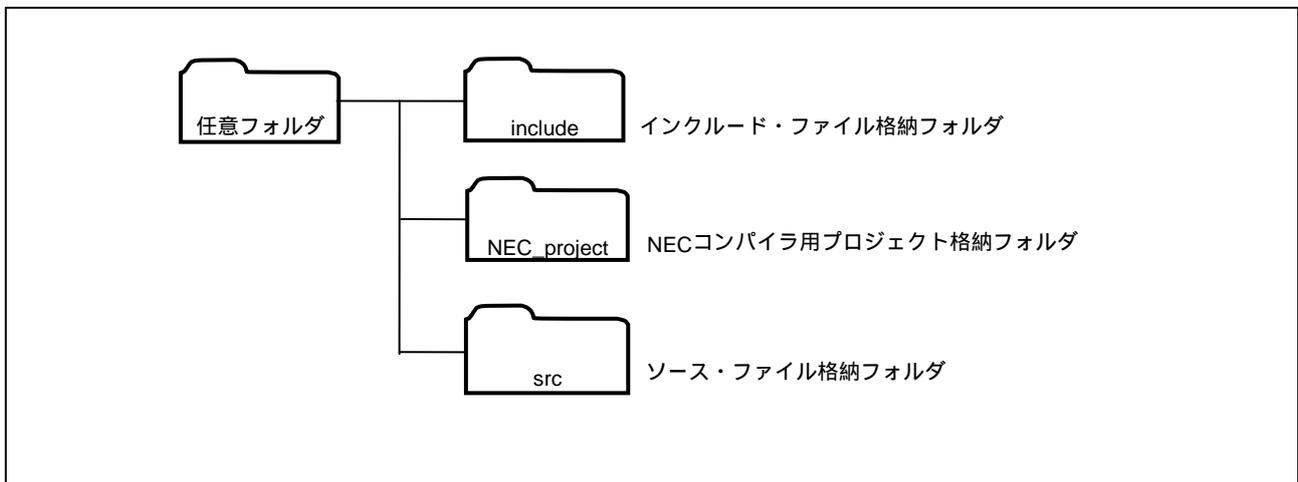
(1) 統合開発ツールのインストール

PM+をインストールします。詳細はPM+のユーザズ・マニュアルを参照してください。

(2) ドライバ類のダウンロード

サンプル・ドライバの提供ファイル一式を、フォルダ構成を変えずに任意のディレクトリに格納します。

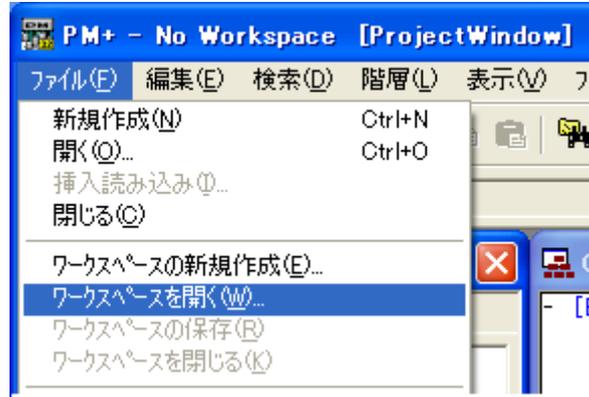
図 5 - 1 サンプル・ドライバのフォルダ構成



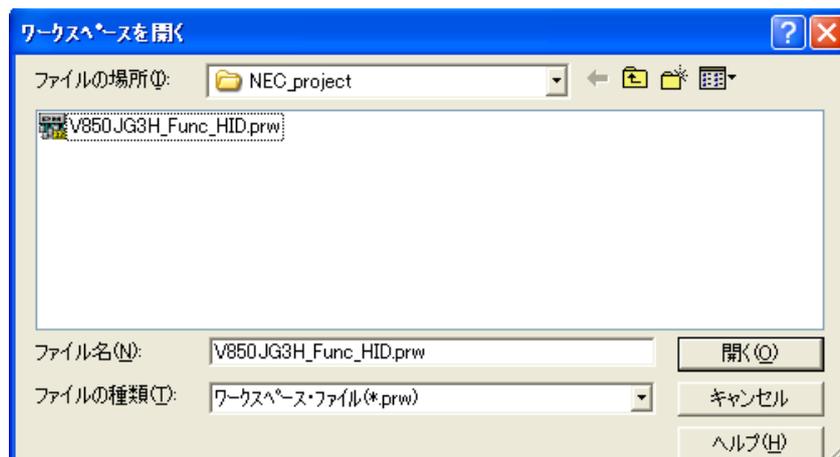
(3) ワークスペースの設定

ここではサンプル・ドライバに同梱のプロジェクト関連ファイルを使用する場合の手順を示します。

<1> PM+を起動し、「ファイル」メニューから「ワークスペースを開く」を選択します。



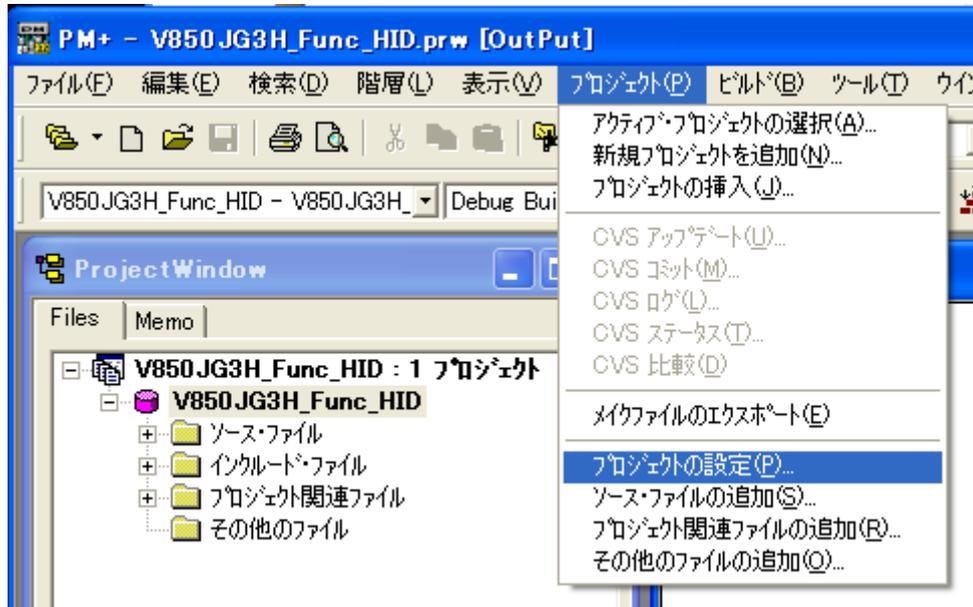
<2> 「ワークスペースを開く」ダイアログが開きます。サンプル・ドライバを格納したディレクトリの「NEC_project」フォルダにあるワークスペース・ファイルを指定します。



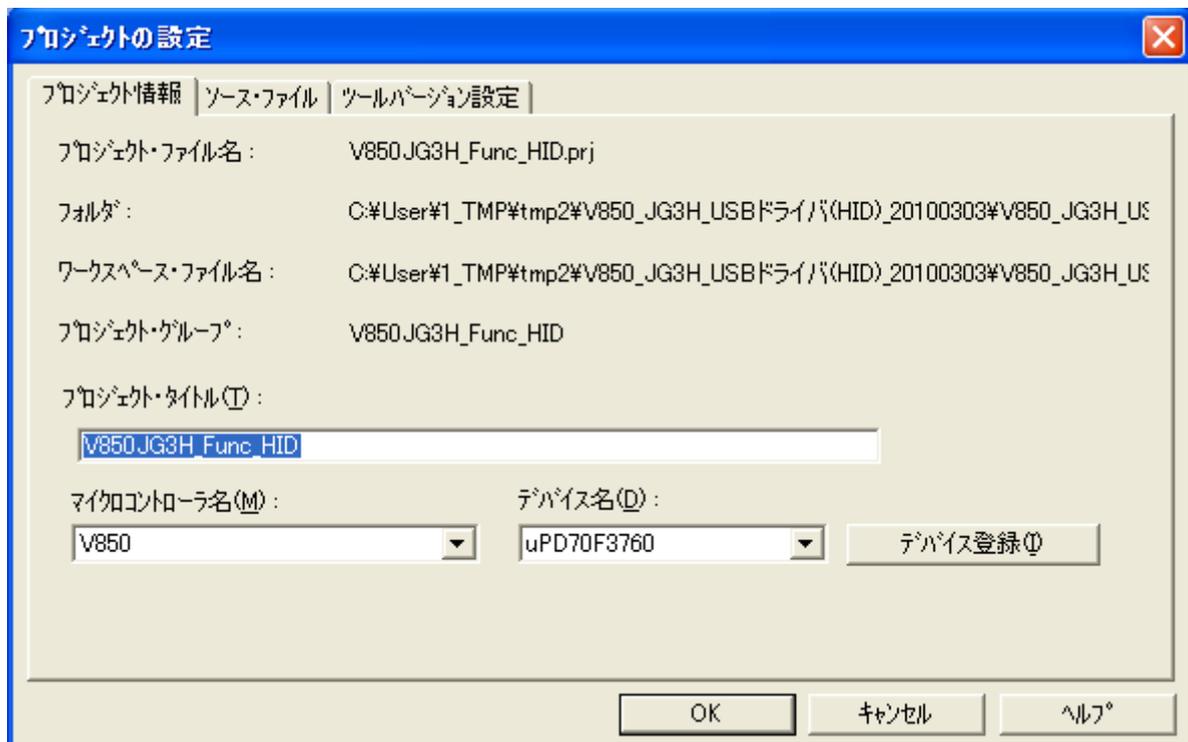
(4) デバイス・ファイルのインストール

ここではV850ES/JG3-H用のデバイス・ファイルを使用する場合の手順を示します。

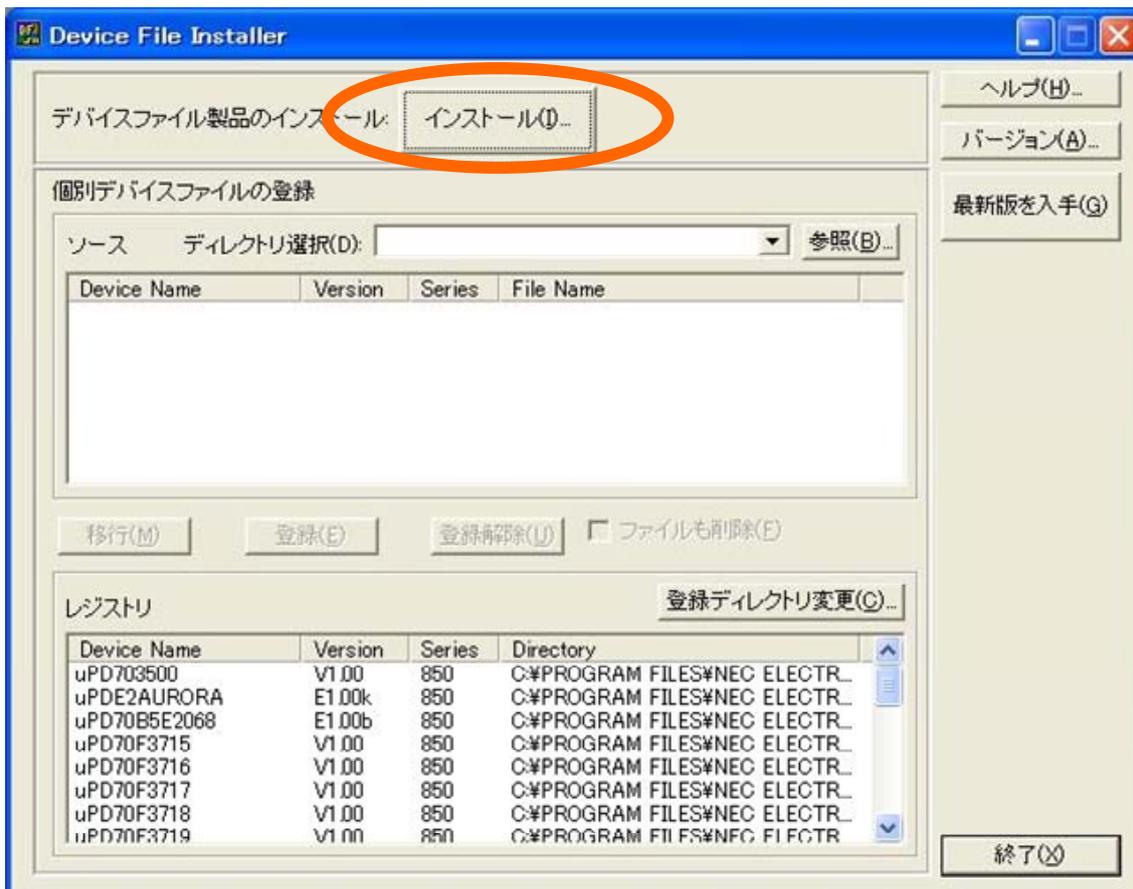
<1> PM+の「プロジェクト」メニューから「プロジェクトの設定」を選択します。



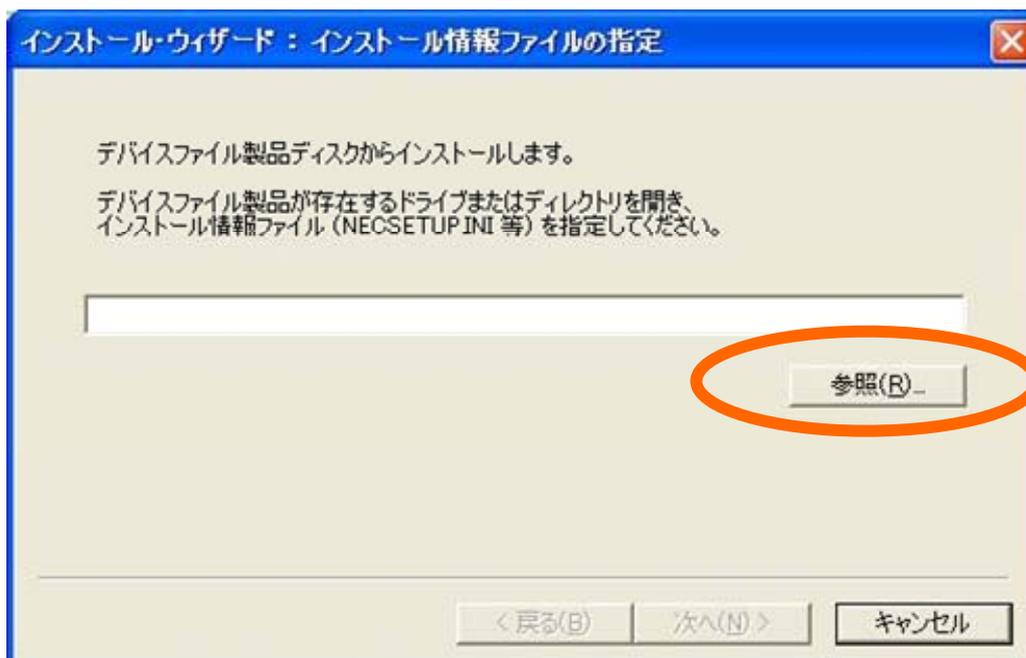
<2> 「プロジェクトの設定」ダイアログが開きます。「プロジェクト情報」タブの「デバイス登録」ボタンを押下してDevice File Installerを起動します。



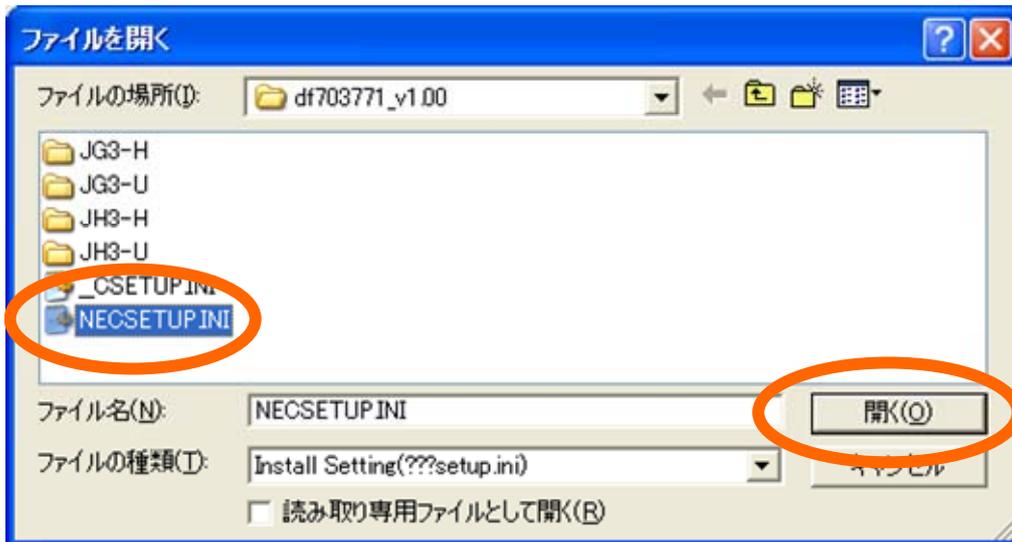
<3> 「Device File Installer」ダイアログが開きます。「インストール」ボタンを押下してインストール・ウィザードを起動します。



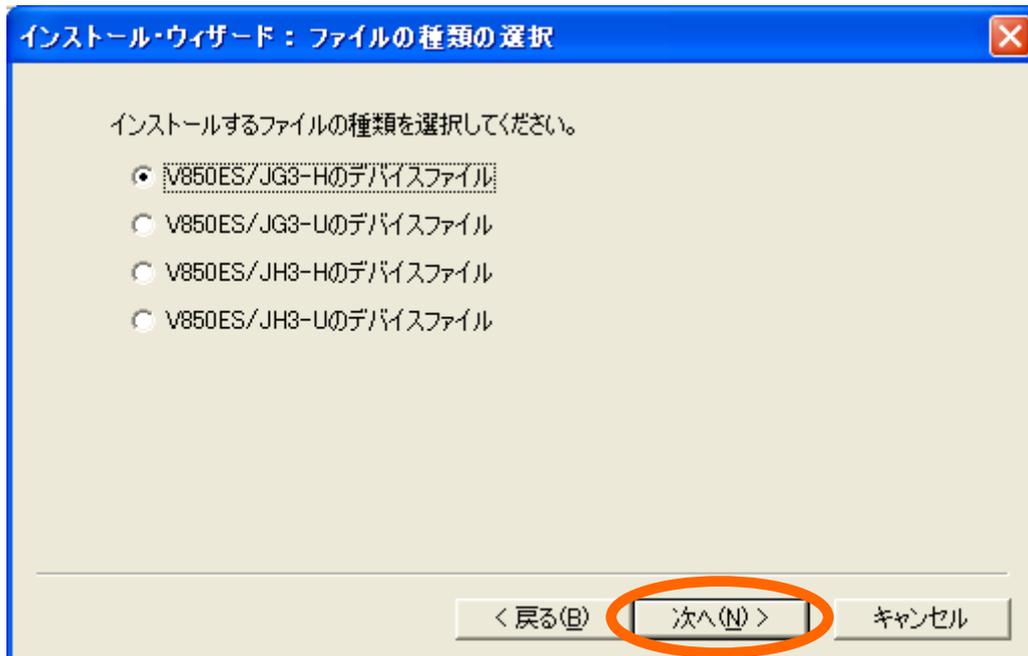
<4> 「インストール情報ファイルの指定」ダイアログが開きます。「参照」ボタンを押下します。



- <5> 「ファイルを開く」ダイアログが表示されます。デバイス・ファイルを格納したディレクトリを開き、「NECSETUP.INI」ファイルを選択して「開く」ボタンを押下します。



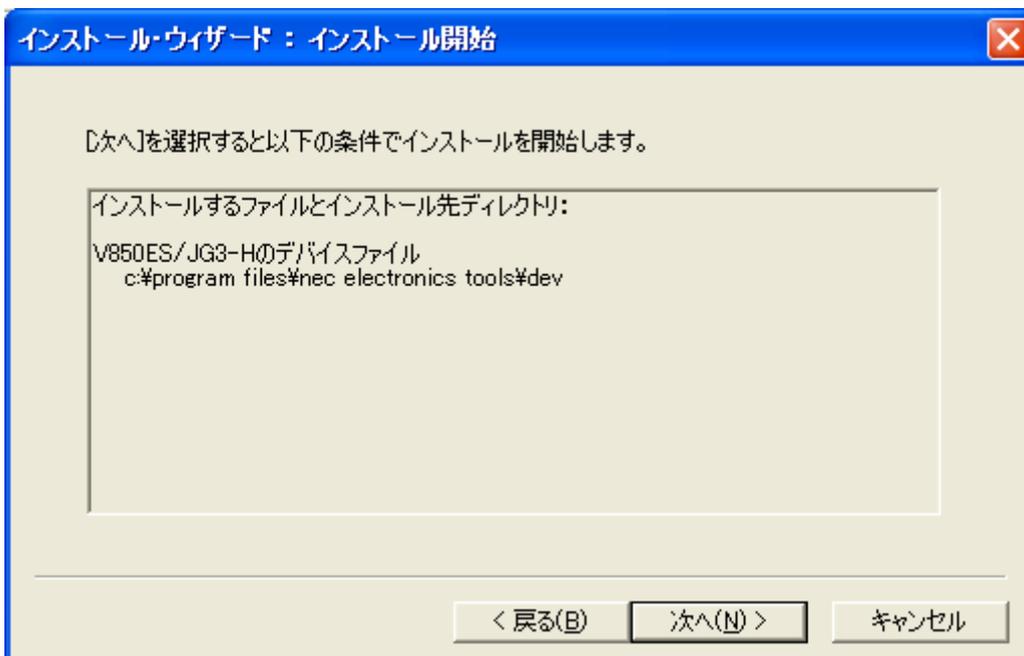
- <6> 使用許諾に関するメッセージが表示されます。「次へ」ボタンを押下します。
- <7> 「ファイルの種類の選択」ダイアログが開きます。該当するデバイス・ファイルを選択して「次へ」ボタンを押下します。



<8> 「インストール先の指定」ダイアログが開きます。パスが表示されていますので、そのまま「次へ」ボタンを押下します。

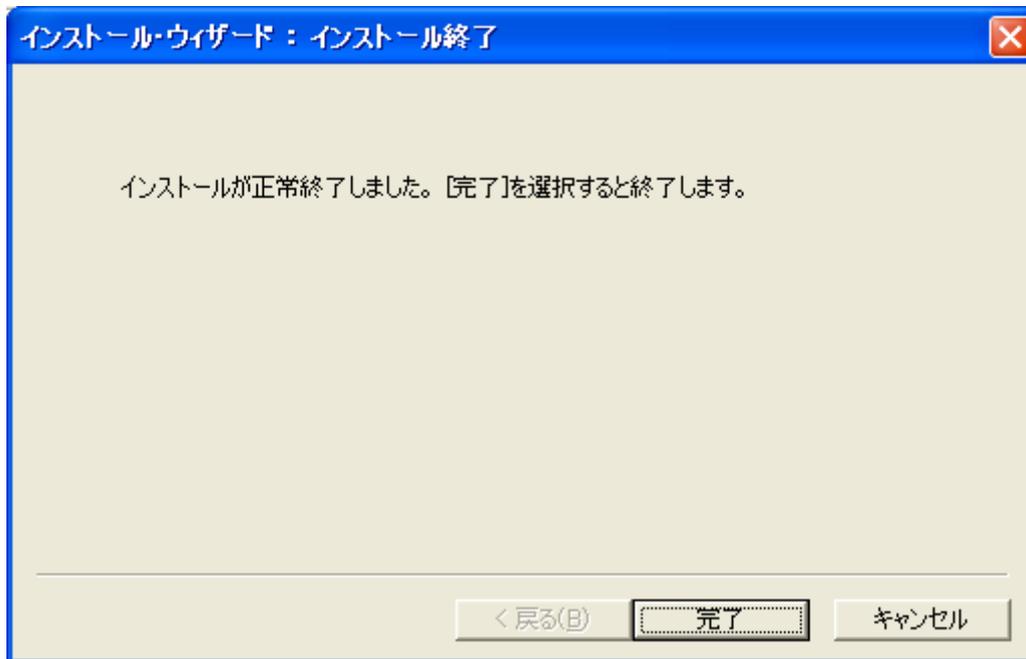


<9> 「インストール開始」ダイアログが開きます。「次へ」ボタンを押下します。



<10> デバイス・ファイルがプロジェクトにインストールされます。環境により、時間がかかる場合があります。

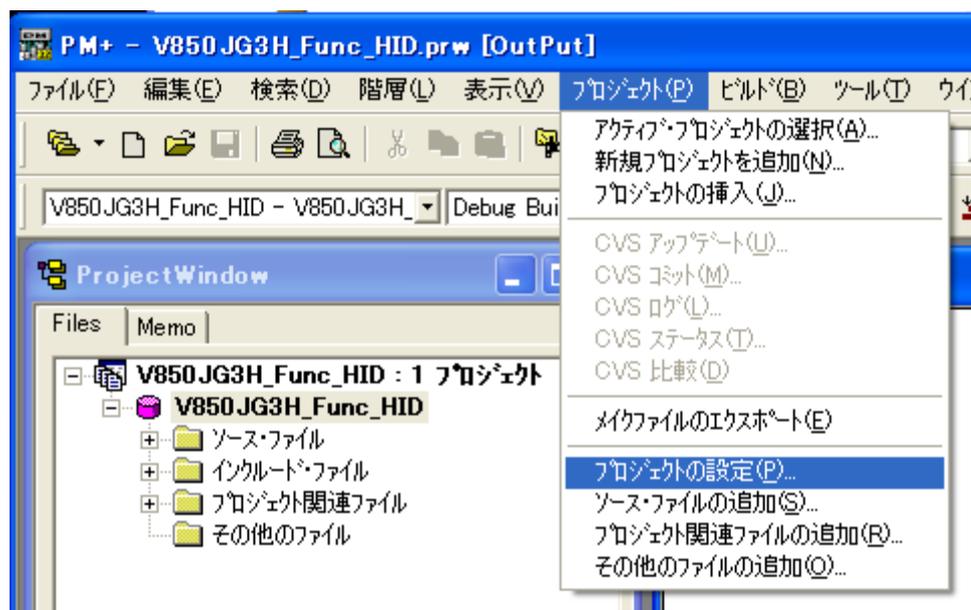
<11> インストールが終わると「インストール終了」ダイアログが開きます。「完了」ボタンを押下します。



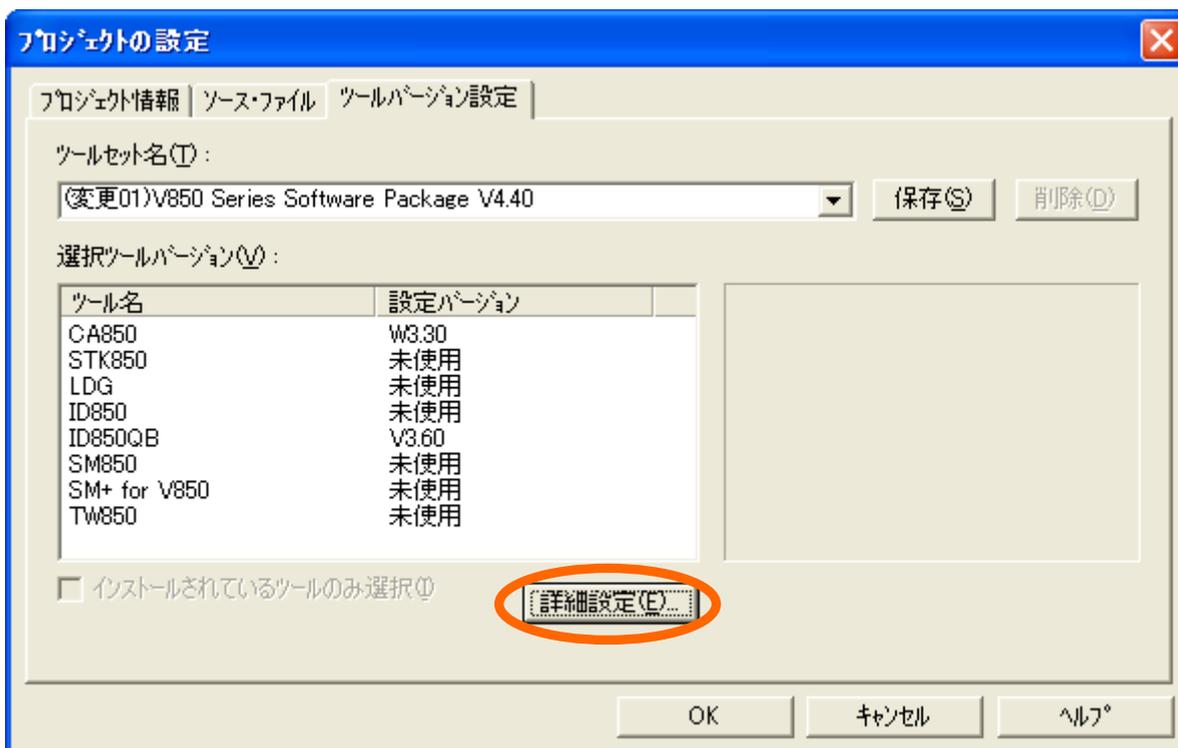
(5) ビルド・ツールの設定

ここではビルド・ツールとしてCA850を、デバッグ・ツールとしてID850QBを使用する場合の手順を示します。

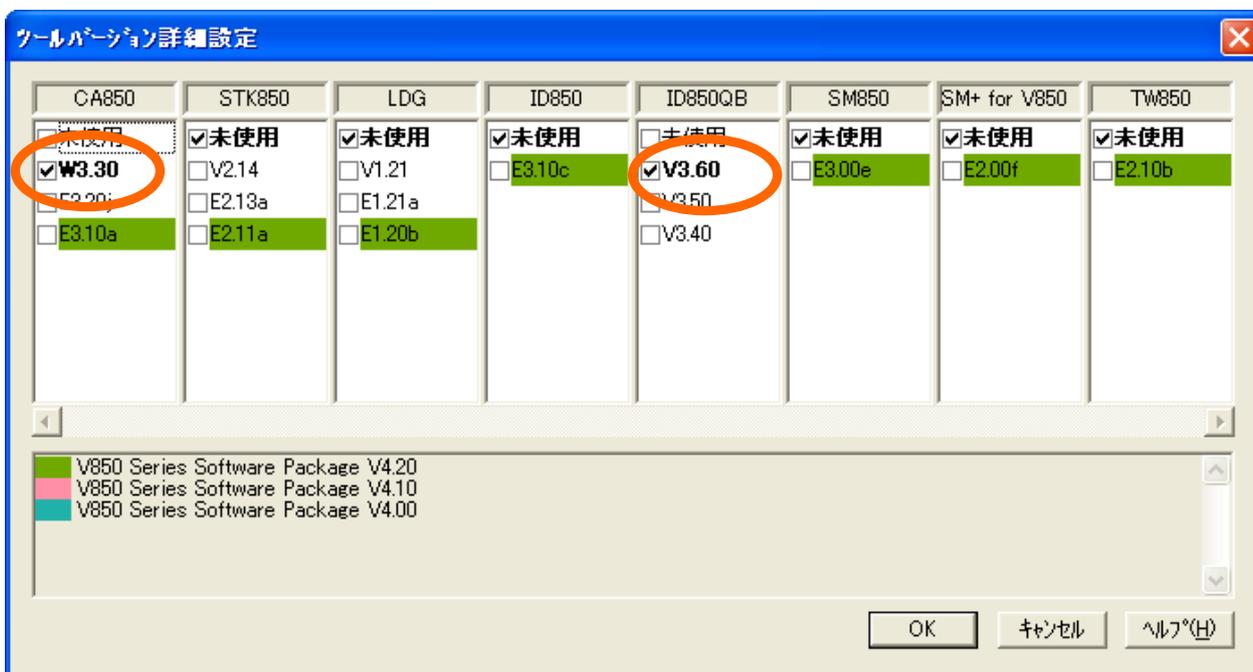
<1> PM+の「プロジェクト」メニューから「プロジェクトの設定」を選択します。



<2> 「プロジェクトの設定」ダイアログが開きます。「ツールバージョン設定」タブの「詳細設定」ボタンを押下します。



<3> 「ツールバージョン詳細設定」ダイアログが開きます。「CA850」の欄で使用するコンパイラのバージョンを、「ID850QB」の欄で使用するデバッガのバージョンを選択します。



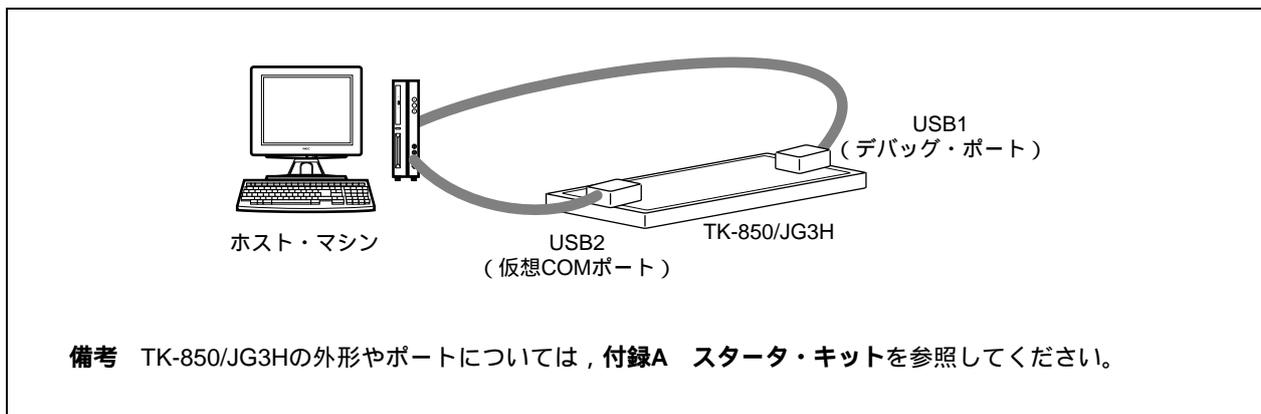
5.2.2 ターゲット環境整備

デバッグに使用するターゲット・デバイスを接続します。

(1) ターゲット・デバイスの接続

TK-850/JG3Hの2つのUSBポートとホスト・マシンのUSBポートをUSBケーブルでそれぞれ接続します。

図 5 - 2 TK-850/JG3Hの接続



5.3 オンチップ・デバッグ

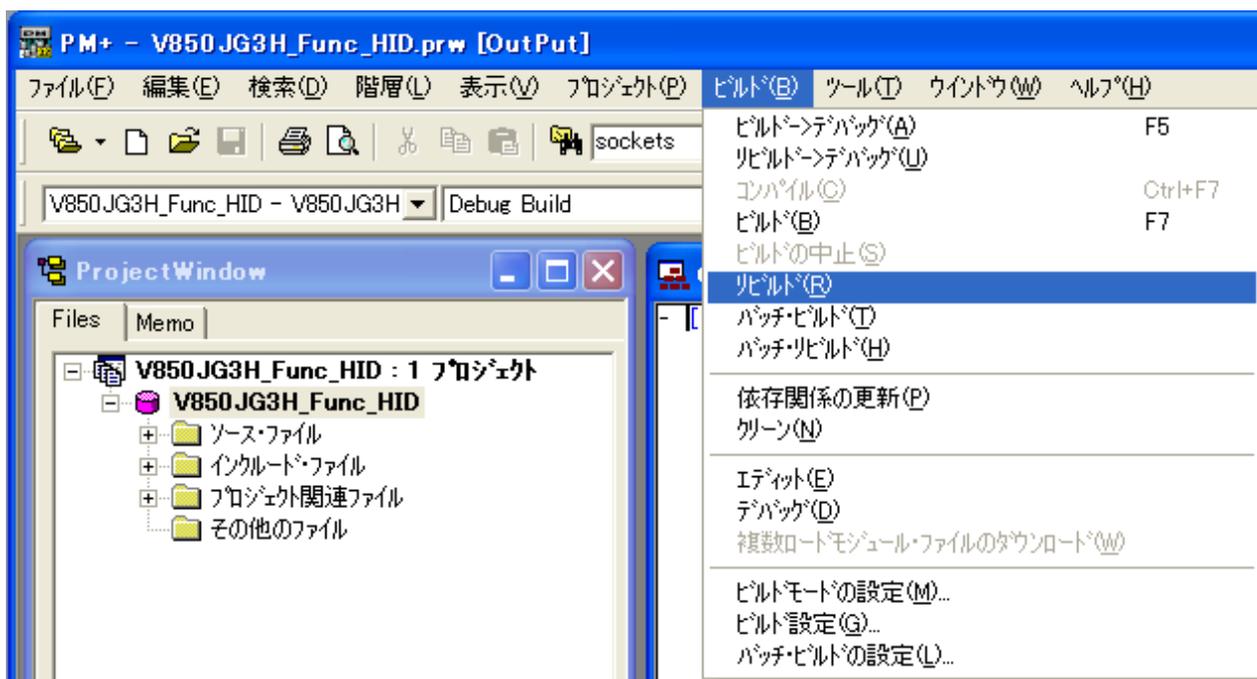
ここでは、5.2 環境設定に示したワークスペースで開発したアプリケーション・プログラムのデバッグ手順について説明します。

V850ES/Jx3-Hでは、内蔵のフラッシュ・メモリにプログラムを書き込み、デバッガなどから直接実行させて動作を検証すること（オンチップ・デバッグ）が可能です。

5.3.1 ロード・モジュール生成

ターゲット・デバイスにプログラムを書き込むには、C言語やアセンブリ言語で記述されたファイルをCコンパイラなどで変換してロード・モジュールを生成します。

PM+では、「ビルド」メニューから「リビルド」を選択すると、ロード・モジュールが生成されます。



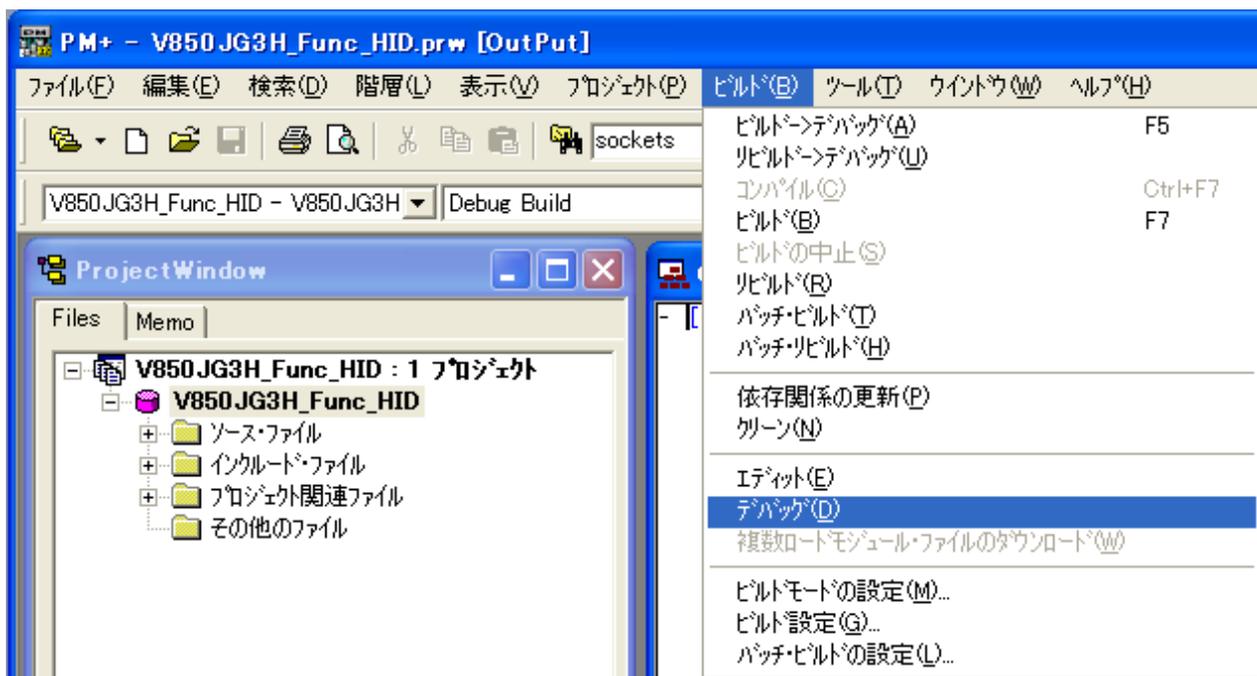
5.3.2 ロードと実行

生成したロード・モジュールをターゲットに書き込んで（ロード）実行させます。

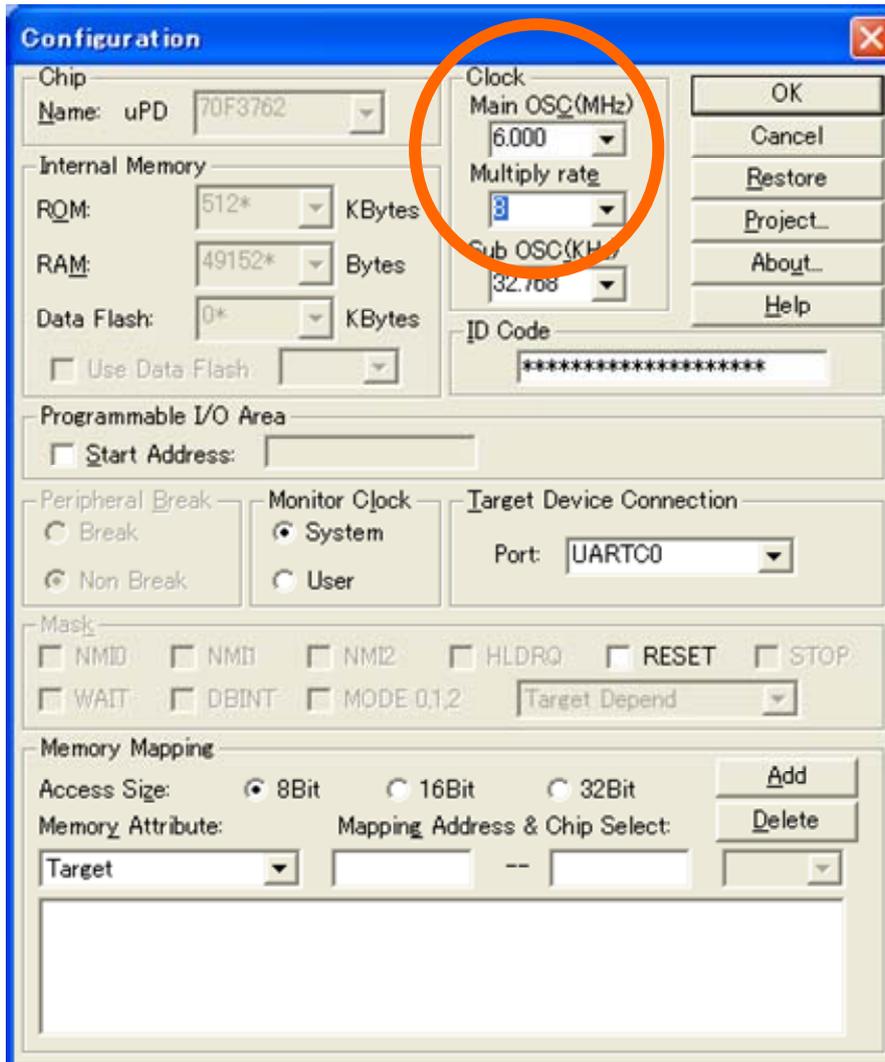
(1) ロード・モジュールの書き込み

ここではPM+を介してTK-850/JG3Hにロード・モジュールを書き込む手順を示します。

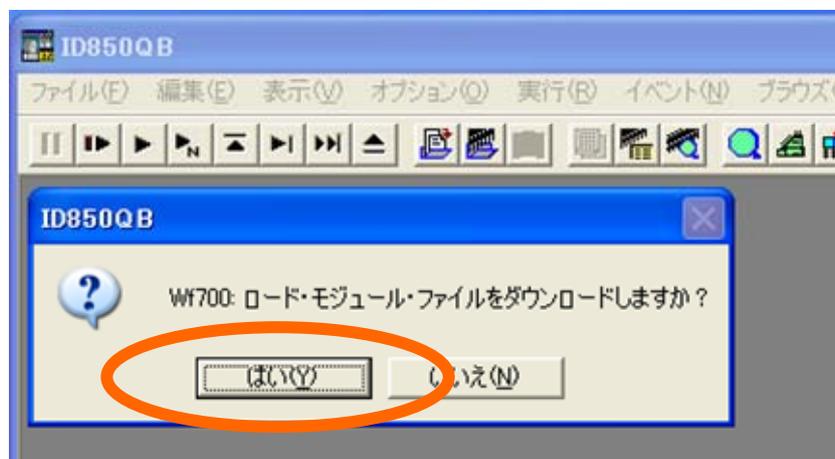
<1> 「ビルド」メニューから「デバッグ」を選択してID850QBを起動します。



- <2> 「Configuration」ダイアログが開きます。「Main OSC」欄に "6.000" (MHz) を、「Multiply rate」欄に "8" を設定します。

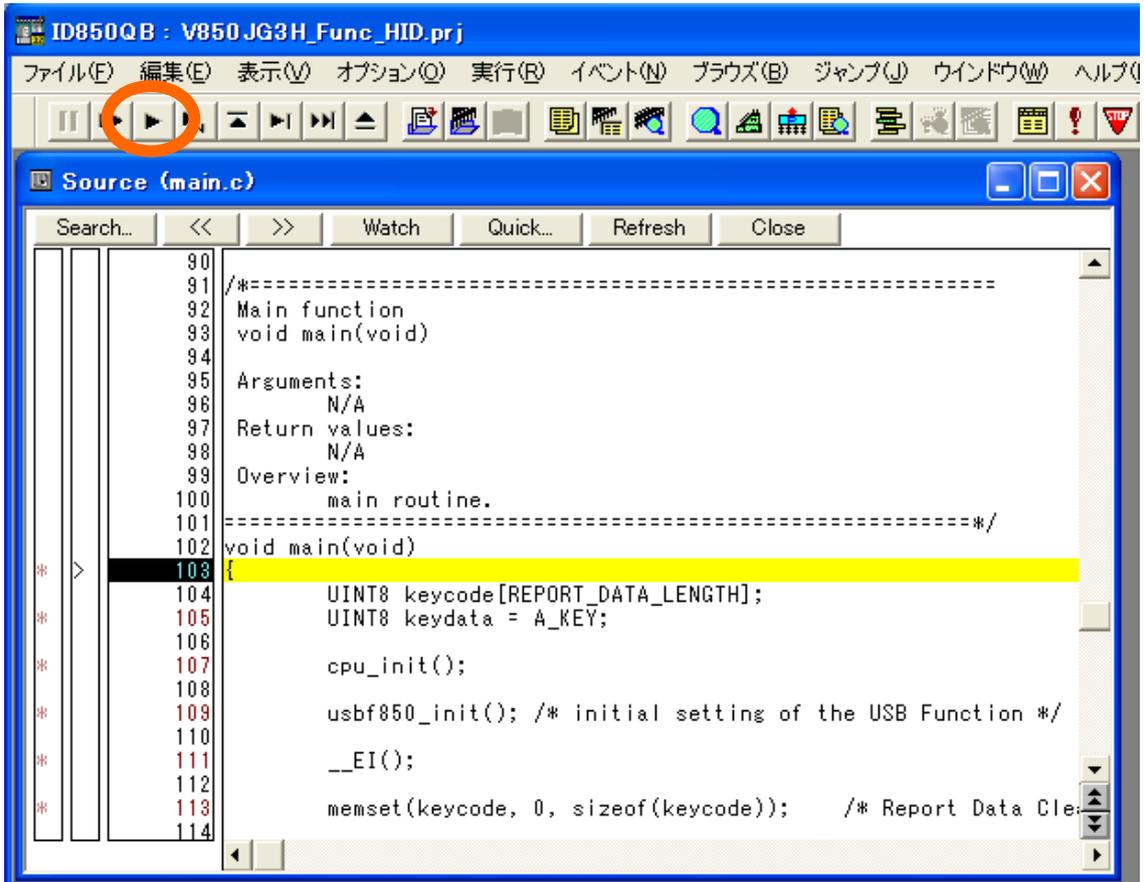


- <3> サンプル・ドライバに同梱のプロジェクト・ファイルを使用している場合は、次の画面が表示されます。「はい」ボタンを押下してロード・モジュール・ファイルの書き込みを開始させます。



(2) プログラムの実行

ID850QBの  ボタンを押下します。または「実行」メニューから「継続して実行」を選択します。



5.4 動作確認

サンプル・ドライバをロードしたターゲット・デバイスとホスト・マシンがUSBで接続されていれば、ドライバ内のサンプル・アプリケーションの実行結果を確認できます。

ホスト上でエディタ・ソフト（メモ帳など）を起動し、動作を確認してください。TK-850/JG3H 上のSW2のPUSHで「Enter」キーを、UPでアルファベット「a」～「z」を順次トグルして（「z」の次は「a」）、送信します。

備考 サンプル・アプリケーションの詳細は第4章 サンプル・アプリケーションの仕様を参照してください。



第6章 サンプル・ドライバの応用

この章では、V850シリーズ マイクロコンピュータ(USBファンクション内蔵品)向けUSBヒューマン・インタフェース・デバイス・クラス用サンプル・ドライバを利用する際に、知っておいていただきたい情報について説明します。

6.1 概 要

サンプル・ドライバ利用の際に、通常、編集する必要があるのは以下の部分になります。

(1) カスタマイズ

次に示す部分を必要に応じて書き換えます。

- ・ "main.c" ファイル内のサンプル・アプリケーション部
- ・ "usbf850.h" ファイル内の各種レジスタの設定値
- ・ "usbf850_desc.h" ファイル内の各種ディスクリプタの内容

備考 サンプル・ドライバのファイル構成については**1.1.3 サンプル・ドライバの構成**を参照してください。

(2) 関数の利用

アプリケーション・プログラム内で必要に応じて呼び出します。実装されている関数の詳細は、**3.3 関数の仕様**を参照してください。

6.2 カスタマイズ

ここでは、サンプル・ドライバの利用にあたり、必要に応じて書き換える部分について説明します。

6.2.1 アプリケーション部

"main.c" ファイルの次に示す部分は、サンプル・ドライバの利用例として簡単な処理を記述したものです。実際にアプリケーションとして必要な処理をこの部分に適宜、記述して下さい。その前にある初期化処理関数やアプリケーション内にあるデータ送受信関数はそのまま利用する事も出来ます。使用の際には **3.3 関数の仕様**、**6.3 関数の利用**を確認して使用して下さい。

リスト 6-1 サンプル・アプリケーションの記述

```

1  #pragma ioreg
2  #include "main.h"
3  #include "usbf850_erno.h"
4  #include "usbf850.h"
5  #include "usbf850_hid.h"
6
7  int _rcopy(unsigned long *, long);
8  extern unsigned long _S_romp;
9
10 extern UINT8   usbf850_keytouch_flg;
11      :
12      中略
13      :
14 void main(void)
15 {
16     UINT8 keycode[REPORT_DATA_LENGTH];
17     UINT8 keydata = A_KEY;
18
19     cpu_init();
20
21     usbf850_init();      /* initial setting of the USB Function */
22
23     EI();
24
25     memset(keycode, 0, sizeof(keycode));      /* Report Data Clear */
26
27     while (1) {
28         if( usbf850_keytouch_flg == F_SW_ON ){
29             if( !(UF0EPS0 & C_IT1) ){
30                 /* SW3 */
31                 if((P9H & 0x01) == 0x00) {
32                     while((P9H & 0x01) == 0x00){
33                         /* SW3 OFF WAIT */
34                     }
35                     keycode[KEY1_SCAN_CODE] = keydata;      /* Press Key Data */
36                     usbf850_data_send(keycode, sizeof(keycode), C_INT1);
37                     memset(keycode, 0, sizeof(keycode));      /* Release Key Data */
38                     usbf850_data_send(keycode, sizeof(keycode), C_INT1);
39                     keydata++;      /* a to z */
40                     if(keydata == EXCLAMATION_KEY) {
41                         keydata = A_KEY;
42                     }
43                 }
44                 /* SW4 */
45                 if((P9H & 0x02) == 0x00) {
46                     while((P9H & 0x02) == 0x00){
47                         /* SW4 OFF WAIT */
48                     }
49                     keycode[KEY1_SCAN_CODE] = ENTER_KEY; /* Press Key Data(Enter) */
50                     usbf850_data_send(keycode, sizeof(keycode), C_INT1);
51                     memset(keycode, 0, sizeof(keycode));      /* Release Key Data */
52                     usbf850_data_send(keycode, sizeof(keycode), C_INT1);
53                     keydata = 0x04;
54                 }
55             }
56             usbf850_keytouch_flg = F_SW_OFF;
57         }
58     }
59 }

```

6.2.2 レジスタの設定

サンプル・ドライバが使用する（書き込みを行う）レジスタとその設定値は、"usbf850.h" ファイルに定義されています。これらのファイル内の値を実際のアプリケーションでの使用法に合わせて書き換えることで、サンプル・ドライバを通してターゲット・デバイスの動作を設定できます。

(1) "usbf850.h" ファイル

USBファンクション・コントローラのレジスタ設定値の定義が記述されています。

6.2.3 ディスクリプタの内容

初期化処理時にサンプル・ドライバがUSBファンクション・コントローラに登録するデータ（**3.1.3 ディスクリプタの設定参照**）が "usbf850_desc.h" ファイルに定義されています。このファイル内の値を実際のアプリケーションでの使用法に合わせて書き換えることで、サンプル・ドライバを通してターゲット・デバイスの属性などの情報を設定できます。

また、ストリング・ディスクリプタには任意の情報を登録できます。サンプル・ドライバでは製造元や製品情報を定義していますので、適宜、書き換えてください。

6.3 関数の利用

使用頻度と汎用性の高い処理が定義済みの関数として用意されていますので、アプリケーションの記述を単純化でき、コード・サイズの節減にもつながります。各関数の詳細は**3.3 関数の仕様**を参照してください。

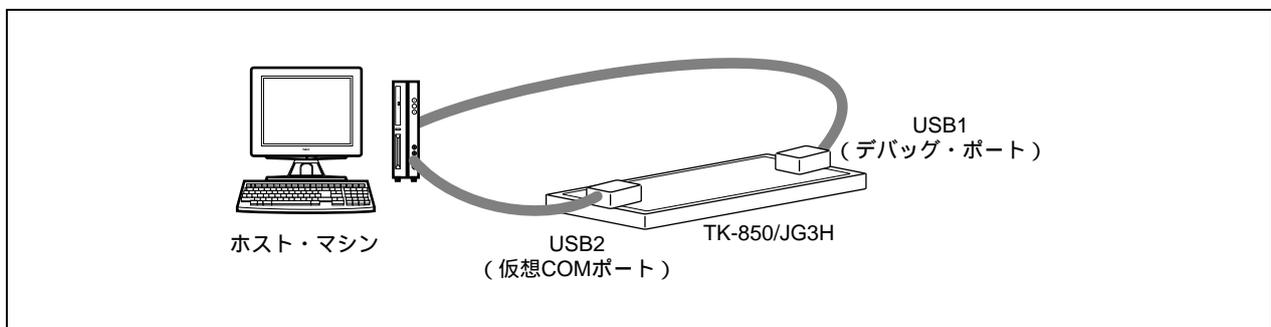
付録A スタータ・キット

この章では、テセラ・テクノロジー社製のV850ES/Jx3-H向けスタータ・キット TK-850/JG3Hについて説明します。

. 1 概 要

TK-850/JG3Hは、V850ES/Jx3-Hを使用したアプリケーション・システムの開発を体験できるキットです。ホスト・マシンに開発ツールやUSBドライバをインストールしてこのキットをUSB接続するだけで、プログラム作成からビルド、デバッグ、動作確認といった一連の開発フローに対応できます。このキットではモニタ・プログラムを使用しており、エミュレータを接続しない状態でのデバッグ（オンチップ・デバッグ）を実現します。

図 0 - 1 TK-850/JG3Hの接続イメージ



. 1.1 特 徴

TK-850/JG3Hには次のような特徴があります。

- ・ 内蔵USBファンクション・コントローラ用USB miniBコネクタを装備
- ・ 最大で84本のI/Oポートを装備
- ・ コンパクトな名刺サイズ
- ・ 統合開発環境（PM+）と組み合わせで効率的な開発を実現

. 2 主な仕様

TK-850/JG3Hの主な仕様は次のとおりです。

CPU	μ PD70F3760 (V850ES/JG3-H)
動作周波数	48 MHz (サブシステム・クロック : 32.768 kHz)
インタフェース	USBコネクタ (miniB) × 2基 N-Wire用コネクタ (パットのみ) MINICUBE [®] 2用コネクタ (SICA : パットのみ) 周辺ボード・コネクタ 2基 (パットのみ)
対応機種	ホスト・マシン : USBインタフェース付きPC/AT互換機 OS : Windows 2000 , Windows XP
動作電圧	5.0 V (内部動作3.3 V)
本体寸法	W89 × D52 (mm)

(メ モ)

【発行】NECエレクトロニクス株式会社(<http://www.necel.co.jp>)

【問い合わせ先】<http://www.necel.com/contact/ja/>