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

μPD78F0730

8 ビット・シングルチップ・マイクロコントローラ

USB HID クラス・ドライバ編

資料番号　U19661JJ1V0AN00（第 1 版）
発行年月　March 2009　NS

［メ　　モ］

目次要約

- 本資料に記載されている内容は2009年3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　このアプリケーション・ノートは，μPD78F0730 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，μPD78F0730 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。

- μPD78F0730 の USB ファンクション・コントローラの概要
- USB 規格の概要
- サンプル・ドライバの仕様
- 開発環境
- サンプル・ドライバの応用

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。

- μPD78F0730 のハードウエア機能，および電気的特性を知りたいとき
  → 別冊の**μPD78F0730　ユーザーズ・マニュアル　ハードウエア編**を参照してください。

- μPD78F0730 の命令機能を知りたいとき
  → 別冊の **78K/0 シリーズ　ユーザーズ・マニュアル　命令編**を参照してください。

凡　　例

データ表記の重み：	左が上位桁，右が下位桁
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：	本文中の補足説明
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。**

○ μPD78F0730 **に関する資料**

資 料 名	資料番号
μPD78F0730　**ユーザーズ・マニュアル**	U19014J
μPD78F0730　**ユーザーズ・マニュアル**　HID **クラス・ドライバ編**	U19339J
μPD78F0730　**アプリケーション・ノート**　USB HID **クラス・ドライバ編**	**この資料**
78K/0 **シリーズ　ユーザーズ・マニュアル　命令編**	U12326J

○ **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
CC78K0 Ver.3.70　C **コンパイラ**	**操作編**	U17201J
	言語編	U17200J
RA78K0 Ver.3.80　**アセンブラ・パッケージ**	**操作編**	U17199J
	言語編	U17198J
	構造化アセンブリ言語編	U17197J
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
SM78K Ver.2.52　**システム・シミュレータ**	**操作編**	U16768J
PM plus Ver.5.10　**プロジェクト・マネージャ**		U16569J
ID78K0-NS Ver.2.70　**統合デバッガ**	**操作編**	U17729J
ID78K0-QB Ver.3.00　**統合デバッガ**	**操作編**	U18492J
QB-780731　**インサーキット・エミュレータ**		U17804J
QB-MINI2　**プログラミング機能付きオンチップ・デバッグ・エミュレータ**		U18371J
PG-FP5　**フラッシュ・メモリ・プログラマ**		U18865J
PG-FP4　**フラッシュ・メモリ・プログラマ**		U15260J

目　次

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 μPD78F0730 内蔵の USB ファンクション・コントローラ用に作成された USB HID（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・ドライバを利用したアプリケーション・プログラム開発のための環境
- サンプル・ドライバを利用するための参考情報

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

μPD78F0730 内蔵の USB ファンクション・コントローラ（USBF）には，次の特徴があります。

- Universal Serial Bus Specification Rev.2.0 に準拠
- 12 Mbps（フル・スピード）転送に対応
- 転送用のエンドポイントを内蔵

表 1－1　μPD78F0730 内蔵 USB ファンクション・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

- 内部クロックと外部クロックを選択可能（f_{USB} = 48 MHz）注

  X1 発振回路で生成するクロック（f_X = 12 または 16 MHz）を 4 または 3 逓倍

  外部入力クロック（f_{EXCLK} = 12 または 16 MHz）を 4 または 3 逓倍

注　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

このサンプル・ドライバには，次の特徴があります。機能や動作の詳細は第 3 章　サンプル・ドライバの仕様を参照してください。

- USB HID クラスのキーボードとして動作
- TK-78F0730 ボードの SW4（INTP0），SW5（INTP1）をキー入力として使用
- Control Endpoint，Interrupt Endpoint 注を使用
- リモート・ウエイクアップ機能サポート
- バス・パワード・デバイスとして動作
- 次に示すサイズのメモリを占有（ベクタ・テーブルを除く）
  ROM：約 2.6 K バイト
  RAM：約 0.2 K バイト

注　μPD78F0730 は Interrupt Endpoint を搭載していないため，Endpoint1（バルク転送用）を Interrupt Endpoint として使用します。

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	初期化，メイン・ルーチン
	usbf78k.c	USBF 初期化，割り込み処理，バルク転送，コントロール転送
	usbf78k_human_interface.c	HID クラス処理
	boot.asm	ブート処理ルーチン
include	errno.h	エラー・コード定義
	main.h	main.c 関数プロトタイプ宣言
	RegDef.h	レジスタ定義
	Types.h	ユーザ型宣言
	usbf78k.h	usbf78k.c 関数プロトタイプ宣言
	usbf78k_desc.h	ディスクリプタ定義
	usbf78k_sfr.h	USB ファンクション・コントローラ用レジスタ・アクセス用マクロ定義
	usbf78k_human_interface.h	usbf78k_human_interface.c 関数プロトタイプ宣言

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4. 2. 1　ホスト環境整備を参照してください。

1.2 μPD78F0730 の概要

ここでは，サンプル・ドライバの制御の対象であるμPD78F0730 について説明します。

μPD78F0730 は，NEC エレクトロニクスの 8 ビット・シングルチップ・マイクロコントローラです。ROM/RAM，タイマ／カウンタ，シリアル・インタフェース，A/D コンバータ，D/A コンバータ，DMA コントローラ，USB ファンクション・コントローラなどの周辺機能を内蔵しています。詳細はμPD78F0730　ユーザーズ・マニュアル（U19014J）を参照してください。

μPD78F0730 には，主に次の特徴があります。

○ 高速（0.125 μs：高速システム・クロック 16 MHz 動作時）で命令実行が可能

○ 汎用レジスタ：8 ビット×32 レジスタ（8 ビット×8 レジスタ×4 バンク）

○ ROM, RAM 容量

項 目 ＼ 品 名	プログラム・メモリ（ROM）	データ・メモリ	
	フラッシュ・メモリ注	内部高速 RAM 注	内部拡張 RAM 注
μPD78F0730	16 K バイト	1 K バイト	2 K バイト

注　内部フラッシュ・メモリ，内部高速 RAM，内部拡張 RAM の容量は，レジスタにより変更可能です。

○ USB ファンクション・コントローラ（USBF）を搭載

○ 単一電源のフラッシュ・メモリ内蔵

○ セルフ・プログラミング内蔵（ブート・スワップ機能あり）

○ オンチップ・デバッグ機能内蔵

○ パワーオン・クリア（POC）回路，低電圧検出（LVI）回路内蔵

○ ウォッチドッグ・タイマ（低速内蔵発振クロックで動作可能）内蔵

○ I/O ポート：19 本（N-ch オープン・ドレーン：2 本）

○ タイマ：5 チャネル

- 16 ビット・タイマ／イベント・カウンタ　：1 チャネル
- 8 ビット・タイマ／イベント・カウンタ　：2 チャネル
- 8 ビット・タイマ　：1 チャネル
- ウォッチドッグ・タイマ　：1 チャネル

○シリアル・インタフェース：3 チャネル

- UART ：1 チャネル
- CSI　：1 チャネル
- USB　：1 チャネル

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くなっています。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
設定可能なパケット・サイズ	ハイ・スピード 480 Mbps	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	1-64 バイト	1-1023 バイト
	ロウ・スピード 1.5 Mbps	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[注]。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（メモリなど）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はあります。

2. 3　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），プリンタ・クラス，ヒューマン・インタフェース・デバイス（HID）クラスなどがあります。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 bInterfaceClass 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　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 2－2　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. 4. 2 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・ドライバが対応するリクエストについては，3. 1. 3　**リクエストへの対応**を参照してください。

(1) 標準リクエスト

すべての USB 対応機器で共通に使用するリクエストです。bmRequestType フィールドのビット 6，5 の値がともに 0 のとき，そのリクエストは標準リクエストです。各標準リクエストの処理内容については，**USB 仕様書（Universal Serial Bus Specification Rev.2.0）**を参照してください。

表 2−3　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，コンフィギュレーション，ストリング，HID，レポート	対象ディスクリプタの読み取り
SET_DESCRIPTOR	デバイス，コンフィギュレーション，ストリング，HID，レポート	対象ディスクリプタの変更（オプション）
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。

HID クラスでは次のリクエストが定義されています。

(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。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。

表 2−4　Get_Report リクエストのフォーマット

フィールド	サイズ	設定値
bmRequestType	1	リクエスト・タイプ：0xA1
bRequest	1	リクエスト識別子：0x01（Get_Report）
wValue	2	上位 1 バイト：レポートの種類，下位 1 バイト：レポート ID
wIndex	2	処理対象のインタフェース番号
wLength	2	レポート長

(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。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。

表 2−5　Get_Idle リクエストのフォーマット

フィールド	サイズ	設定値
bmRequestType	1	リクエスト・タイプ：0xA1
bRequest	1	リクエスト識別子：0x02（Get_Idle）
wValue	2	上位 1 バイト：レポートの種類，下位 1 バイト：レポート ID
wIndex	2	処理対象のインタフェース番号
wLength	2	0x0001

(c) 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。

表 2−6　Get_Protocol リクエストのフォーマット

フィールド	サイズ	設定値
bmRequestType	1	リクエスト・タイプ：0xA1
bRequest	1	リクエスト識別子：0x03（Get_Protocol）
wValue	2	上位 1 バイト：レポートの種類，下位 1 バイト：レポート ID
wIndex	2	処理対象のインタフェース番号
wLength	2	0x0001

(d) 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。

表 2－7　Set_Report リクエストのフォーマット

フィールド	サイズ	設定値
bmRequestType	1	リクエスト・タイプ：0x21
bRequest	1	リクエスト識別子：0x09（Get_Report）
wValue	2	上位 1 バイト：レポートの種類，下位 1 バイト：レポート ID
wIndex	2	処理対象のインタフェース番号
wLength	2	レポート長

(e) 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。

表 2－8　Get_Idle リクエストのフォーマット

フィールド	サイズ	設定値
bmRequestType	1	リクエスト・タイプ：0x21
bRequest	1	リクエスト識別子：0x0A（Set_Idle）
wValue	2	上位 1 バイト：レポート間の最大時間間隔（4 ms 単位）（0 の場合，変化があった場合のみレポートを返し，変化がないときは NAK 応答します） 下位 1 バイト：レポート ID（0 の場合，すべての入力レポートに適用します）
wIndex	2	処理対象のインタフェース番号
wLength	2	0x0000

(f) 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。

表 2－9　Set_Protocol リクエストのフォーマット

フィールド	サイズ	設定値
bmRequestType	1	リクエスト・タイプ：0x21
bRequest	1	リクエスト識別子：0x0B（Set_Protocol）
wValue	2	0x0000
wIndex	2	処理対象のインタフェース番号
wLength	2	0x0001

(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. 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

USB 規格では，次に示す 5 種類のディスクリプタが標準で定義されています。

- 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。
- ストリング・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。
  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　標準ディスクリプタのフォーマット

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 2－10　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	製造者を表すストリング・ディスクリプタへのインデックス
iProduct	1	製品を表すストリング・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ストリング・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 2－11　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，およびエンドポイント・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ストリング・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 μA 単位）

表 2－12　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ストリング・ディスクリプタへのインデックス

表 2－13　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 2－14　ストリング・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 ディスクリプタのフォーマット

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。

表 2－15　HID 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）

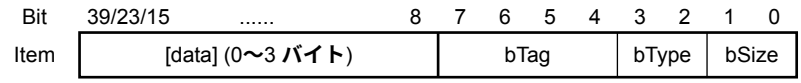
2. 5. 4　レポート・ディスクリプタのフォーマット

レポート・ディスクリプタは，USB ホストと USB デバイスの間でやりとりするデータのフォーマットを定義します。レポート・ディスクリプタの記述は，ほかのディスクリプタとは違い，必要なデータ・フィールドの数に伴って，長さと内容が変化します。

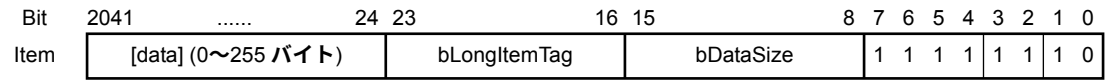
レポート・ディスクリプタは，アイテムと呼ばれる情報群から構成されます。アイテムにはロング・アイテムとショート・アイテムがあり（図 2－1参照），3 つのタイプ（メイン，グローバル，ローカル）が定義されています。

図 2－1　レポート・ディスクリプタのアイテム・フォーマット

（a）ショート・アイテム



（b）ロング・アイテム



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。

(1) メイン・アイテム（bType = 00）

データの種類や集合を定義します。

表 2－16　レポート・ディスクリプタのメイン・アイテム

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：ロールオーバなし，1：ロールオーバあり
		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	出力するデータの形式を指定します。	
		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）。	

(2) グローバル・アイテム（bType = 01）

データを記述します。

表 2－17　レポート・ディスクリプタのグローバル・アイテム

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。

(3) ローカル・アイテム（bType = 10）

特性を定義します。

表 2－18　レポート・ディスクリプタのローカル・アイテム

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，μPD78F0730 向け USB HID 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

このサンプル・ドライバには次のような処理が実装されています。

(1) 初期化処理

CPU 初期化処理では，μPD78F0730 のメモリ・サイズ，クロック，ポート（SW4，SW5）の設定を行います。USBF 初期化処理では，USB ファンクション・コントローラ（USBF）を使用するために必要な項目を設定します。詳細は**3.2　初期化処理**，を参照してください。

(2) 割り込み処理

INTP0 および INTP1 割り込みハンドラでは，キー入力があったことを示すフラグの更新を行います。INTUSB0 割り込みハンドラでは，コントロール・エンドポイントの状態を監視し，リクエストに対する応答処理などを行います。詳細は**3.3　割り込み処理**を参照してください。

サンプル・ドライバでは，INTRSUM 割り込みハンドラも定義していますが，何も行いません。

(3) メイン・ルーチン

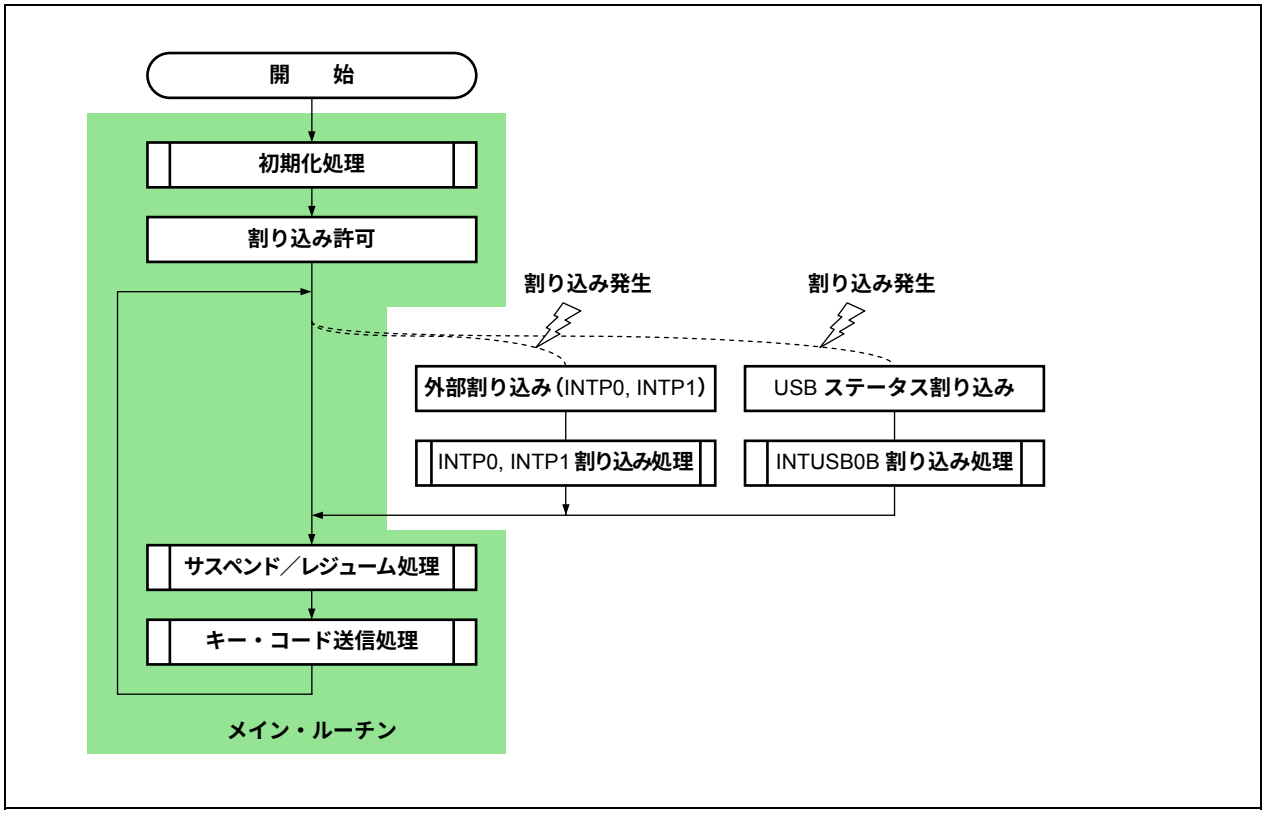
各割り込みハンドラを使用してサスペンド／レジューム処理および入力キー・コード送信処理を行うアプリケーション例です。詳細は**3.4　メイン・ルーチン**を参照してください。

備考　μPD78F0730 内部では USB ケーブルが抜かれたことを検知することができません。検知する必要のある場合は，VBUS の状態を任意のポートに入力するなど，外部回路を構成する必要があります。また，このサンプル・ドライバでも，USB ケーブルを抜いたことに対する処理を実装していません。電源投入後の処理手順については，**μPD78F0730　ユーザーズ・マニュアル（U19014J）**を参照してください。

3. 1. 2　処理の流れ

サンプル・ドライバを実行すると図 3－1および図 3－2に示す処理を行います。

図 3－1　サンプル・ドライバの処理の流れ 1

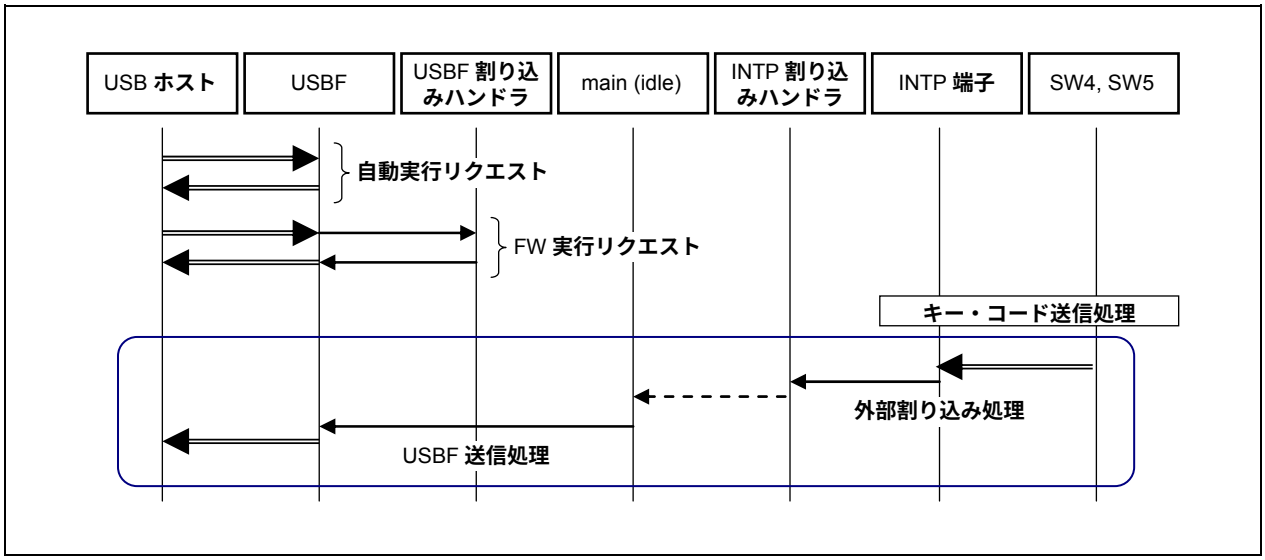


<1> ボード起動時の初期化が終了すると，各種割り込み信号による割り込み処理を開始します。メイン処理関数（main）ではループしてフラグの変化を監視します。割り込み処理によりフラグが変化すると，サスペンド／レジューム処理またはキー・コード送信処理を行います。

<2> USB の状態が変化すると，INTUSB0B 割り込みハンドラが起動されます。このハンドラでは，サスペンド状態を検出する処理，バス・リセットに対する処理，リクエストに対する処理を行います。
サスペンド状態が発生した場合，このハンドラではレジューム／サスペンド・フラグ（rs_flg）を設定するのみで，サスペンド／レジューム処理を行いません。割り込み処理が終了したあと，メイン処理関数（main）内でサスペンド／レジューム処理を実行します。
また，ハードウエアが自動自動応答しない USB リクエストを受信した場合は，応答処理も実行します。

<3> 外部割り込み（INTP0，INTP1）を検出すると，外部割り込み割り込みハンドラが起動されます。このハンドラでは，キー押下フラグ（key_touch）を設定するのみで，入力キーの判別および情報の送信を行いません。割り込み処理が終了したあと，メイン処理関数（main）内で入力キーを判別し，USB からキー・コードを送信します。

図 3−2　サンプル・ドライバの処理の流れ 2



<1>　**自動実行リクエスト**

ハードウエア（μPD78F0730**）が自動的に応答します。**

<2>　FW **実行リクエスト**

ファームウエア（サンプル・ドライバ）が USBF **割り込みハンドラ処理内で応答します。**

<3>　**キー・コード送信処理**

外部割り込み処理でスイッチの押下を検出し，メイン処理で押下されたスイッチの判別とキー・コードの送信を行います。

3. 1. 3　リクエストへの対応

表 3－1にハードウエア（μPD78F0730）およびファームウエア（サンプル・ドライバ）で定義されている USB リクエストを示します。

表 3－1　USB リクエストの処理

リクエスト名	コード								処　　理
	0	1	2	3	4	5	6	7	
標準リクエスト									
GET_INTERFACE	0x81	0x0A	0x00	0x00	0xXX	0xXX	0x01	0x00	HW 自動応答
GET_CONFIGURATION	0x80	0x08	0x00	0x00	0x00	0x00	0x01	0x00	HW 自動応答
GET_DESCRIPTOR Device	0x80	0x06	0x00	0x01	0x00	0x00	0xXX	0xXX	HW 自動応答
GET_DESCRIPTOR Configuration	0x80	0x06	0x00	0x02	0x00	0x00	0xXX	0xXX	HW 自動応答
GET_DESCRIPTOR String	0x80	0x06	0x00	0x03	0x00	0x00	0xXX	0xXX	FW 応答
GET_DESCRIPTOR HID	0x80	0x06	0x00	0x21	0x00	0x00	0xXX	0xXX	FW 応答
GET_DESCRIPTOR Report	0x80	0x06	0x00	0x22	0x00	0x00	0xXX	0xXX	FW 応答
GET_STATUS Device	0x80	0x00	0x00	0x00	0x00	0x00	0x02	0x00	HW 自動応答
GET_STATUS Interface	0x81	0x00	0x00	0x00	0xXX	0xXX	0x02	0x00	HW 自動 STALL 応答
GET_STATUS Endpoint n	0x82	0x00	0x00	0x00	0xXX	0xXX	0x02	0x00	HW 自動応答
CLEAR_FEATURE Device	0x00	0x01	0x01	0x00	0x00	0x00	0x00	0x00	HW 自動応答
CLEAR_FEATURE Interface	0x01	0x01	0x00	0x00	0xXX	0xXX	0x00	0x00	HW 自動 STALL 応答
CLEAR_FEATURE Endpoint n	0x02	0x01	0x00	0x00	0xXX	0xXX	0x00	0x00	HW 自動応答
SET_DESCRIPTOR	0x00	0x07	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	FW STALL 応答
SET_FEATURE Device	0x00	0x03	0x01	0x00	0x00	0x00	0x00	0x00	HW 自動応答
SET_FEATURE Interface	0x02	0x03	0xXX	0xXX	0xXX	0xXX	0x00	0x00	HW 自動 STALL 応答
SET_FEATURE Endpoint n	0x02	0x03	0x00	0x00	0xXX	0xXX	0x00	0x00	HW 自動応答
SET_INTERFACE	0x01	0x0B	0xXX	0xXX	0xXX	0xXX	0x00	0x00	HW 自動応答
SET_CONFIGURATION	0x00	0x09	0xXX	0xXX	0x00	0x00	0x00	0x00	HW 自動応答
SET_ADDRESS	0x00	0x05	0xXX	0xXX	0x00	0x00	0x00	0x00	HW 自動応答
クラス・リクエスト									
Get_Report	0xA1	0x01	0xXX	0xXX	0x00	0x00	0xXX	0xXX	FW 応答
Get_Idle	0xA1	0x02	0x00	0x00	0x00	0x00	0x00	0x01	FW 応答
Get_Protocol	0xA1	0x03	0xXX	0xXX	0x00	0x00	0x00	0x01	FW STALL 応答
Set_Report	0x21	0x09	0xXX	0xXX	0x00	0x00	0xXX	0xXX	FW STALL 応答
Set_Idle	0x21	0x0A	0xXX	0xXX	0x00	0x00	0x00	0x00	FW 応答
Set_Protocol	0x21	0x0B	0xXX	0xXX	0x00	0x00	0x00	0x01	FW STALL 応答
その他のリクエスト	**上記以外**								FW STALL 応答

備考　HW：ハードウエア（μPD78F0730）

FW：ファームウエア（サンプル・ドライバ）

0xXX：不定値

(1) 標準リクエスト

ハードウエア（μPD78F0730）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，GET_DESCRIPTOR_HID，GET_DESCRIPTOR_Report

ホストがファンクション・デバイスのディスクリプタ（ストリング／HID／レポート）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ディスクリプタ（ストリング／HID／レポート）の送信処理（コントロール・リード転送）を行います。

(b) SET_DESCRIPTOR

ホストがファンクション・デバイスのディスクリプタを設定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 STALL 応答を返します。

(2) クラス・リクエスト

サンプル・ドライバには，次のクラス・リクエストへの応答処理が実装されています。

(a) Get_Report（bRequest = 0x01）

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 HID 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（bRequest = 0x02）

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，アイドル率の送信処理を行います。

(c) Set_Idle（bRequest = 0x0A）

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。

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 応答を返します。

(d) Get_Protocol，Set_Report，Set_Protocol（bRequest = 0x03，0x09，0x0B）

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 応答を返します。

(3) 定義されていないリクエスト

定義されていないリクエストを受信すると，サンプル・ドライバは STALL 応答を返します。

3. 1. 4　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 "usbf78k_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 リクエストに応答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エ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 UF0DDn レジスタ（n = 0-17）に格納します。

表 3−2　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
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	0x01CE	プロダクト ID：μPD78F0730
bcdDevice	2	0x0001	デバイスのリリース番号：第 1 版
iManufacturer	1	0x01	製造者を表すストリング・ディスクリプタへのインデックス：1
iProduct	1	0x02	製品を表すストリング・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ストリング・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 リクエストに応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エ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 3−3　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，およびエンドポイント・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ストリング・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・パワード，リモート・ウエイクアップあり
bMaxPower	1	0x32	このコンフィギュレーションの最大消費電流：100 mA

(3) インタフェース・ディスクリプタ

GET_DESCRIPTOR_configuration リクエストに応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エ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 3−4　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：HID
bInterfaceSubClass	1	0x00	サブクラス・コード：なし
bInterfaceProtocol	1	0x00	プロトコル・コード：固有プロトコル使用せず
iInterface	1	0x00	このインタフェースを記述するストリング・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 リクエストに応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エ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ラプト・エンドポイントを 1 つ使用しています。

表 3−5　Endpoint1 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	0x81	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：1
bmAttributes	1	0x03	このエンドポイントの転送タイプ：割り込み
wMaxPacketSize	2	0x0040	この転送の最大パケット・サイズ：64 バイト
bInterval	1	0x00	このエンドポイントのポーリング間隔：0 ms

(5) ストリング・ディスクリプタ

GET_DESCRIPTOR_string リクエストに応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・ドライバはストリング・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　ストリング・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：ストリング
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：ストリング
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

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	0x16	ディスクリプタのサイズ：22 バイト
bDescriptorType	1	0x03	ディスクリプタの種類：ストリング
bString 注2	20	–	シリアル番号：0_98765432

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 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 3−7　HID 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 3−8　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	–

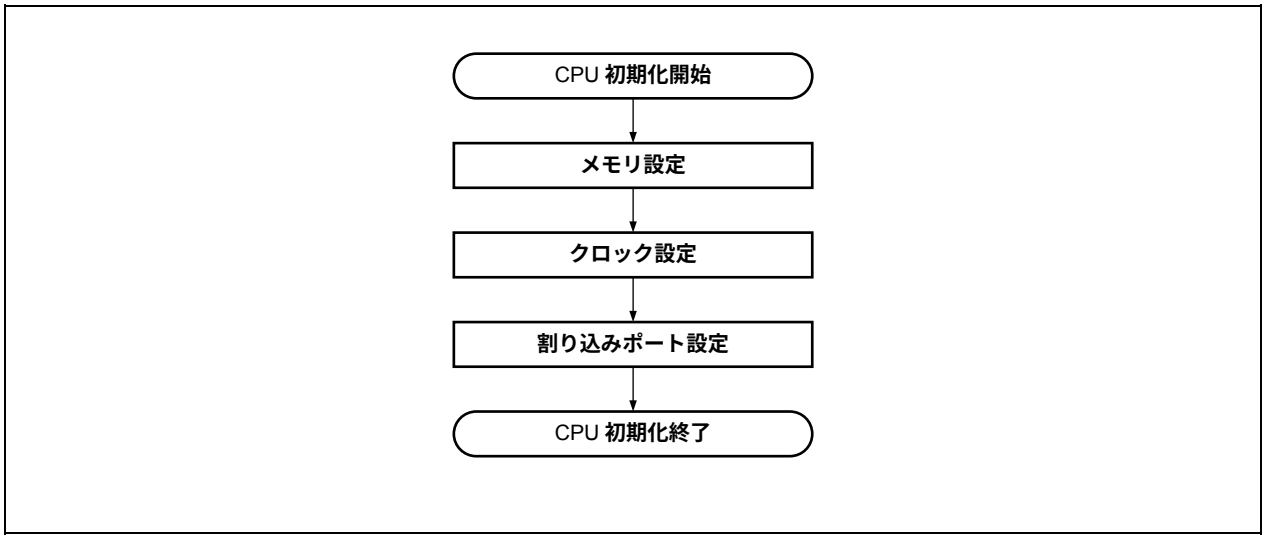
3. 2　初期化処理

メイン・ルーチンの初期化処理では，CPU 初期化処理，USBF 初期化処理を呼び出します。

3. 2. 1　CPU 初期化処理

メモリ・サイズ，クロック，ポート（SW4，SW5）など，μPD78F0730 を使用するために必要な項目を設定します。

図 3－3　CPU 初期化処理フロー



(1) メモリ設定

ROM サイズ（16 KB），内部拡張 RAM サイズ（ハイスピード：1 KB，拡張 2 KB）の設定を行います。

IMS レジスタに "0xC4"，IXS レジスタに "0x08" を書き込みます。

(2) クロック設定

高速システム・クロック，CPU クロック，PLL への供給クロックの設定を行います。

この設定により，高速内蔵発振クロック（f_{RH} = 16 MHz）で動作し，USB 動作クロック（f_{USB}）が 48 MHz になります。

(3) 割り込みポート設定

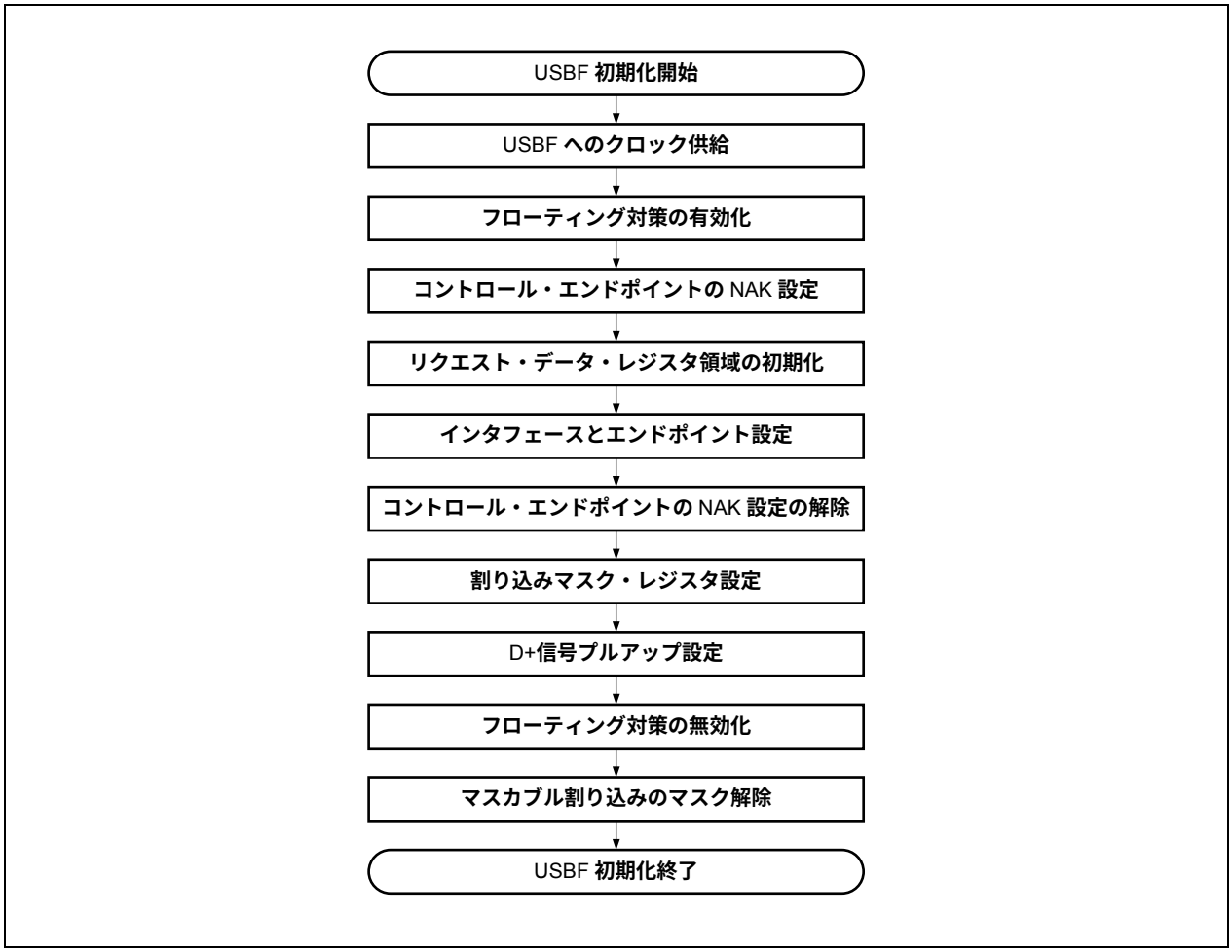
INTP0（SW4 入力），INTP1（SW5 入力）の設定を行います。

- PM12 レジスタに "0x01"，PU12 レジスタに "0x01" を書き込みます。この設定により，P120/INTP0 端子の機能が入力ポートになり，プルアップ機能が有効になります。
- PM3 レジスタに "0x01"，PU3 レジスタに "0x01" を書き込みます。この設定により，P30/INTP1 端子の機能が入力ポートになり，プルアップ機能が有効になります。
- EGP レジスタに "0x00"，EGN レジスタに "0x03" を書き込みます。この設定により，P120/INTP0 および P30/INTP1 端子の立ち下がりエッジを検出して割り込み要求が発生するようになります。

3. 2. 2　USBF 初期化処理

USB ファンクション・コントローラ（USBF）を使用するために必要な項目を設定します。

図 3－4　USBF 初期化処理フロー



(1) USBF へのクロック供給

USB ファンクション・コントローラ（USBF）へのクロック供給を許可します。

- UCKC レジスタの UCKCNT ビットに "1" を書き込みます。

(2) フローティング対策の有効化

USB 用バッファのフローティング対策を有効にし，ケーブル未接続時の不定値による Bus Reset などの誤認識を防止します。

- UF0BC レジスタの UBFIOR ビットに "0" を書き込みます。

(3) 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 応答するように設定します。

- 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します。

(4) 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，デバイス・ディスクリプタ，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・ディスクリプタです。

① UF0DSTL レジスタに "0x00" 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禁止され，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。

② UF0EnSL レジスタ（n = 0-2）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。

③ 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，使用する UF0CIEn レジスタ（n = 0-255）の範囲が決まります。

④ UF0DDn レジスタ（n = 0-17）にデバイス・ディスクリプタのデータを書き込みます。

⑤ 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。

⑦ 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を許可します。

(5) 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レジスタに設定します。

① UF0AIFN レジスタに "0x80" を書き込みます。
この設定により，Interface0 を有効にします。

② F0AAS 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。

③ UF0E1IM レジスタに "0x40"，UF0E2IM レジスタに "0x40" を書き込みます。
この設定により，Endpoint1 が Interface0 にリンクされます。

(6) 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・エンドポイントの NAK 設定を解除します。

- 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。

(7) 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。

① UF0ICn レジスタ（n = 0-4）のすべての有効ビットに "1" 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。

② UF0FIC0，UF0FIC1 レジスタのすべての有効ビットに "1" 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。

③ UF0IM0 レジスタに "0x07" 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，RSUSPDM，BUSRSTM 割り込み以外の要因がマスクされます。

④ 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マスクされます。

⑤ UF0IM2 レジスタに "0x30" 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。

⑥ UF0IM3 レジスタに "0x0E" 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のうち，BKO1DT 割り込み以外の要因がすべてマスクされます。

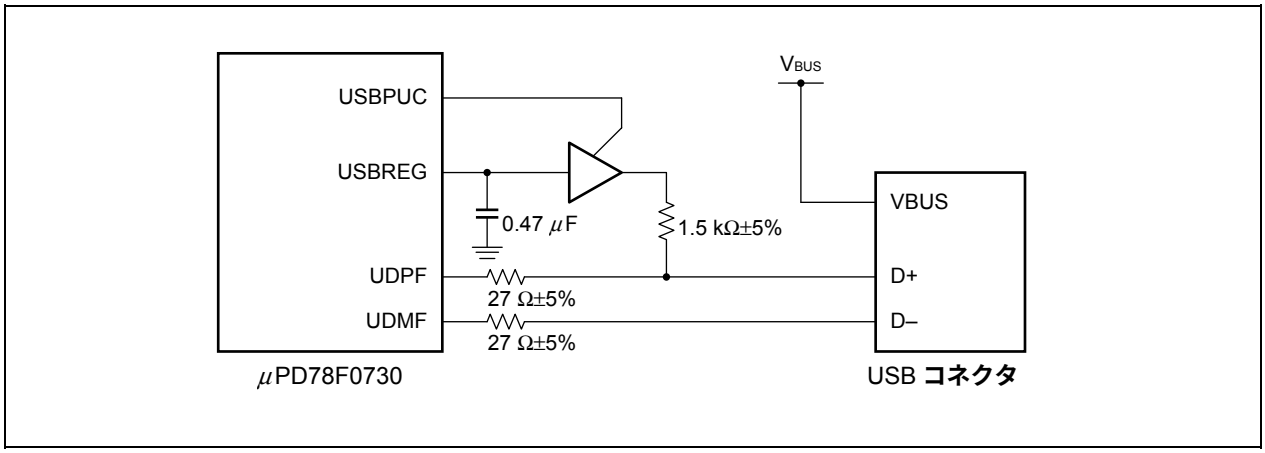
⑦ UF0IM4 レジスタに "0x20" を書き込みます。この設定により，UF0IS4 レジスタに示される割り込み要因がすべてマスクされます。

(8) D+信号プルアップ設定

D+信号をプルアップし，ホスト側にデバイスが接続されたことを認識させます。

- UF0GPR レジスタの CONNECT ビットに "1" を書き込みます。この設定により，USBPUC 端子がハイ・レベルになります。

図 3−5　USB ファンクション・コントローラ接続例



(9) フローティング対策の無効化

フローティング対策を無効化します。

- UF0BC レジスタに "0x03" を書き込みます。

この設定により，フローティング対策が無効になり，USB 用バッファが有効になります。

(10) マスカブル割り込みのマスク解除

割り込み要因 INTUSB0，INTUSB1，INTRSUM，INTP0，INTP1 のマスクを解除します。

- 割り込みマスク・フラグ MK0L の USBMK0，USBMK1 ビットに "0" を書き込みます。
- 割り込みマスク・フラグ MK1L の RSUMMK ビットに "0" を書き込みます。
- 割り込みマスク・フラグ MK0L の PMK0，PMK1 ビットに "0" を書き込みます。

3. 3　割り込み処理

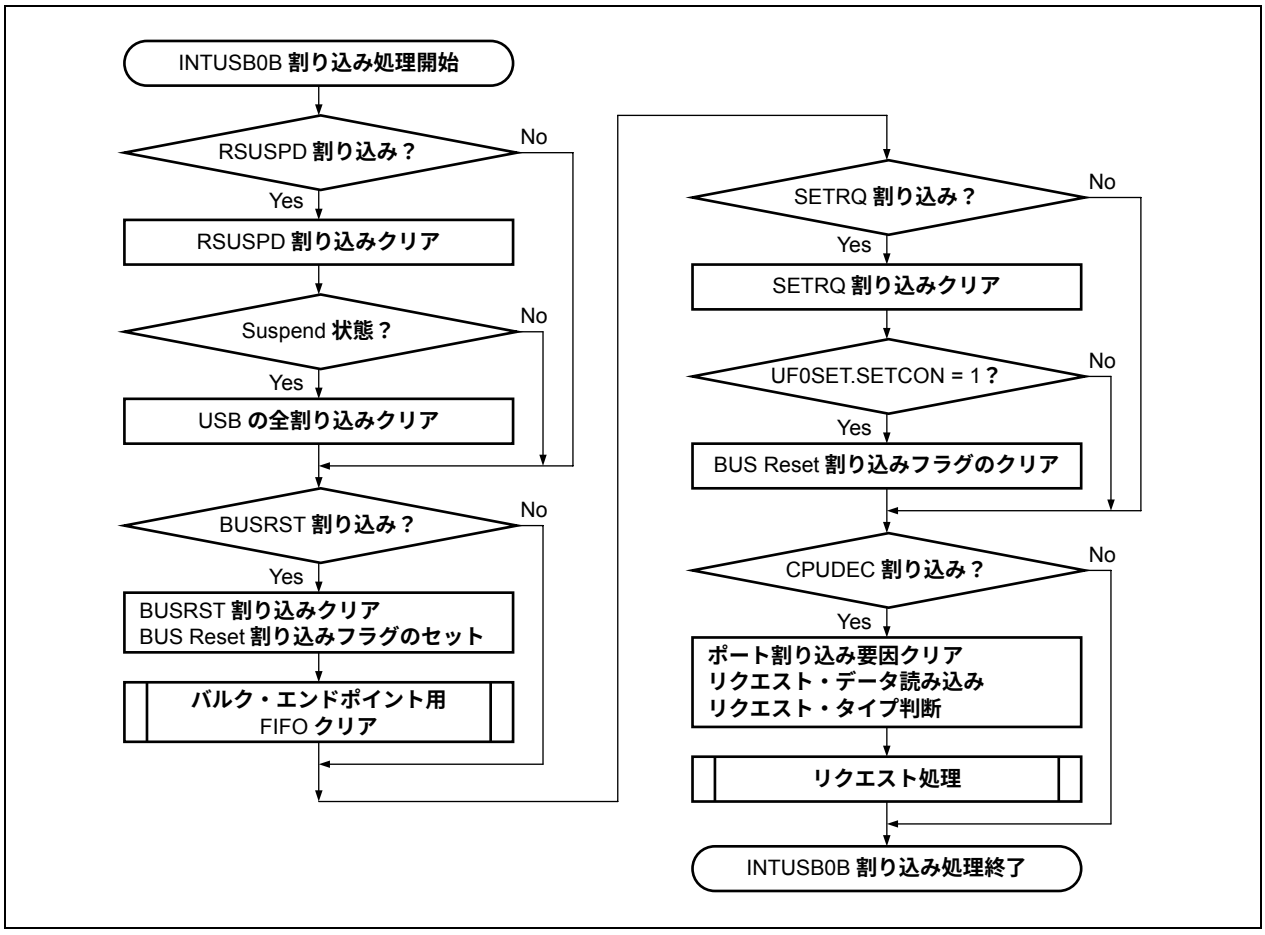
割り込みハンドラとして次の処理を登録します。

- INTUSB0：USBF 割り込み処理
- INTRSUM：USB レジューム割り込み処理
- INTP0，INTP1：外部割り込み処理

3. 3. 1　USBF 割り込み処理（INTUSB0B）

INTUSB0B 割り込みハンドラでは，RSUSPD，BUSRST，SETRQ，CPUDEC 割り込みの処理を行います。

図 3−6　INTUSB0B 割り込みハンドラ処理フロー



(1) RSUSPD 割り込みの処理

UF0IS0 レジスタの RSUSPD ビットが "1" のとき，RSUSPD 割り込みが発生していると判断します。

RSUSPD 割り込みが発生している場合，次の処理を行います。

- 割り込み要因をクリア（UF0IC0 レジスタの RSUSPDC ビットに "0" を書き込む）
- Suspend/Resume 状態の判定

UF0EPS1 レジスタの RSUM ビットが "1" のとき，Suspend 状態にあると判断します。

Suspend 状態の場合，次の処理を行います。

- レジューム／サスペンド・フラグ（rs_flg）を "SUSPEND（0x00）" に変更
- USB の割り込み要因をすべてクリア（これにより，以降の INTUSB0B 割り込み処理は省略されます。）

(2) BUSRST 割り込みの処理

UF0IS0 レジスタの BUSRST ビットが "1" のとき，BUSRST 割り込みが発生していると判断します。

BUSRST 割り込みが発生している場合，次の処理を行います。

- 割り込み要因をクリア（UF0IC0 レジスタの BUSRST ビットに "0" を書き込む）
- BUS Reset 割り込みフラグ（usbf78k_busrst_flg）に "1" を設定
- FIFO クリア関数（usbf78k_clearFIFO）を呼び出し，バルク・エンドポイント用 FIFO をすべてクリア

(3) SETRQ 割り込みの処理

UF0IS0 レジスタの SETRQ ビットが "1" のとき，割り込みが発生していると判断します。

SETRQ 割り込みが発生している場合，次の処理を行います。

- 割り込み要因をクリア（UF0IC0 レジスタの SETRQ ビットに "0" を書き込む）
- 自動応答リクエスト（SET_XXXX）の処理

UF0SET レジスタの SETCON ビットが "1" のとき，SET_CONFIGURATION リクエストを受信し，自動処理を行った状態にあると判断します。

自動処理を行った場合，BUS Reset 割り込みフラグ（usbf78k_busrst_flg）を "0" にします。

注意　厳密に Configured ステートに入ったことを確認する場合は，UF0CNF レジスタの値を確認してください。

(4) CPUDEC 割り込みの処理

UF0IS1 レジスタの CPUDEC ビットが "1" のとき，割り込みが発生していると判断します。

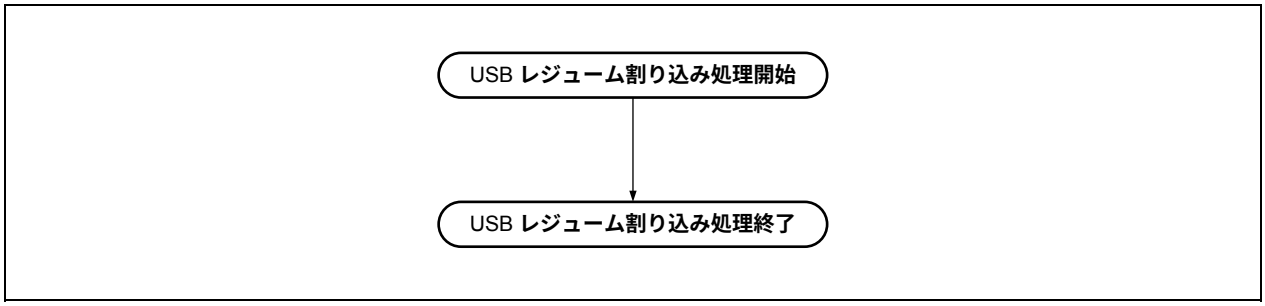
CPUDEC 割り込みが発生している場合，次の処理を行います。

- ポート割り込み要因クリア（UF0IC1 レジスタの PORT ビットに "0" を書き込む）
- 受信データを FIFO から読み込み，リクエスト・データを構成
- ハードウエアで自動応答しないリクエストのリクエスト・タイプに応じて，各リクエストの処理を実行

3. 3. 2　USB レジューム割り込み処理（INTRSUM）

サンプル・ドライバの USB レジューム割り込み処理では，処理を何も行いません。

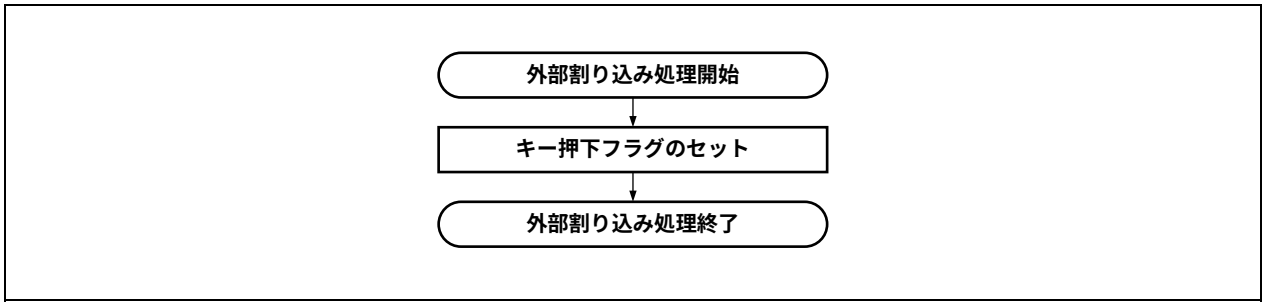
図 3－7　USB レジューム割り込み処理フロー



3. 3. 3　外部割り込み処理（INTP0，INTP1）

INTP0，INTP1 割り込みハンドラでは，キー押下フラグ（key_touch）のセット（1）のみを行います。メイン・ルーチンでは，キー押下フラグを監視し，キー押下に対応する処理を実行します。

図 3－8　外部割り込み処理フロー



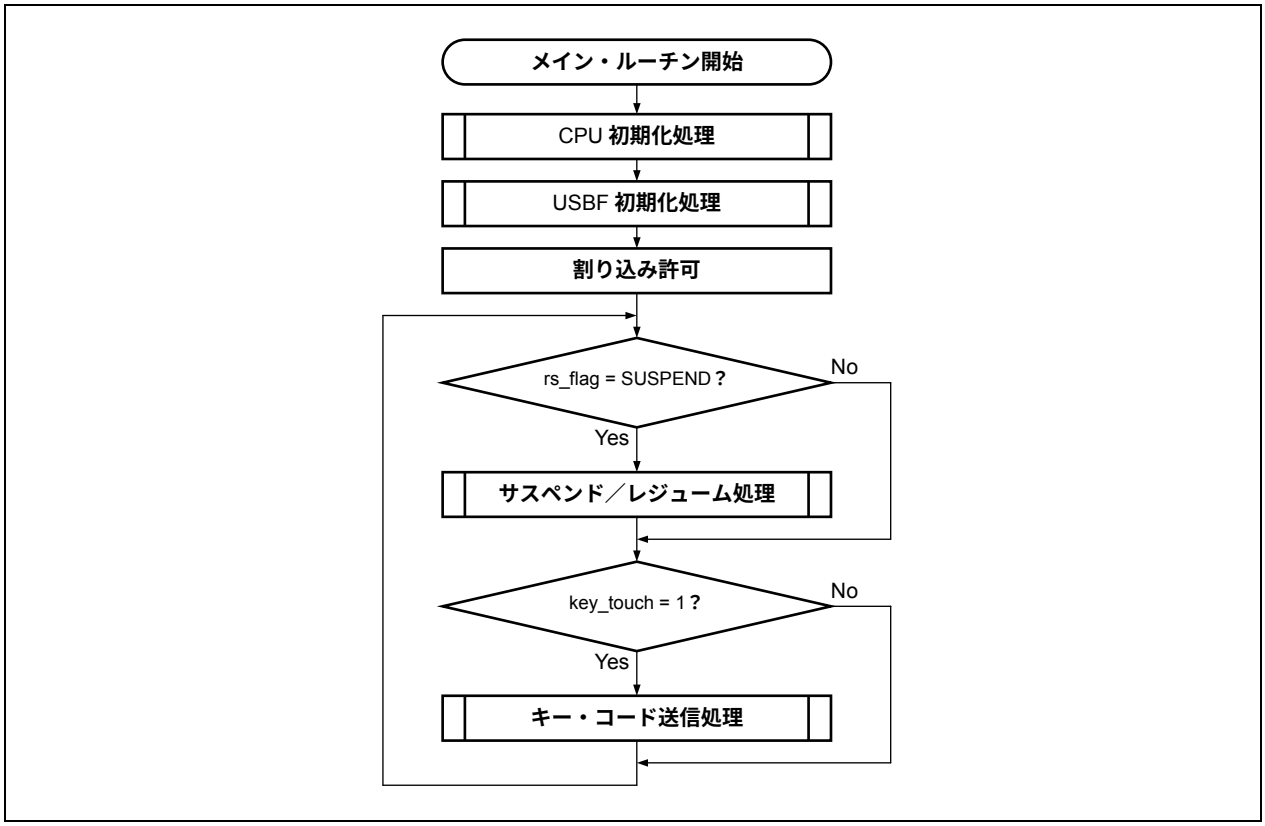
3. 4　メイン・ルーチン

メイン・ルーチンは，サンプル・ドライバを利用したアプリケーションの簡単な例として用意されたものです。

このメイン・ルーチンは，割り込みハンドラ処理でセットされるフラグを監視し，USB バスがサスペンド状態になった場合の処理，押下されたキーのコードを送信する処理を実現します。この処理の過程で，サンプル・ソフトウエアの各種関数を利用します。

メイン・ルーチンでは，次のフローで一連の処理を行います。

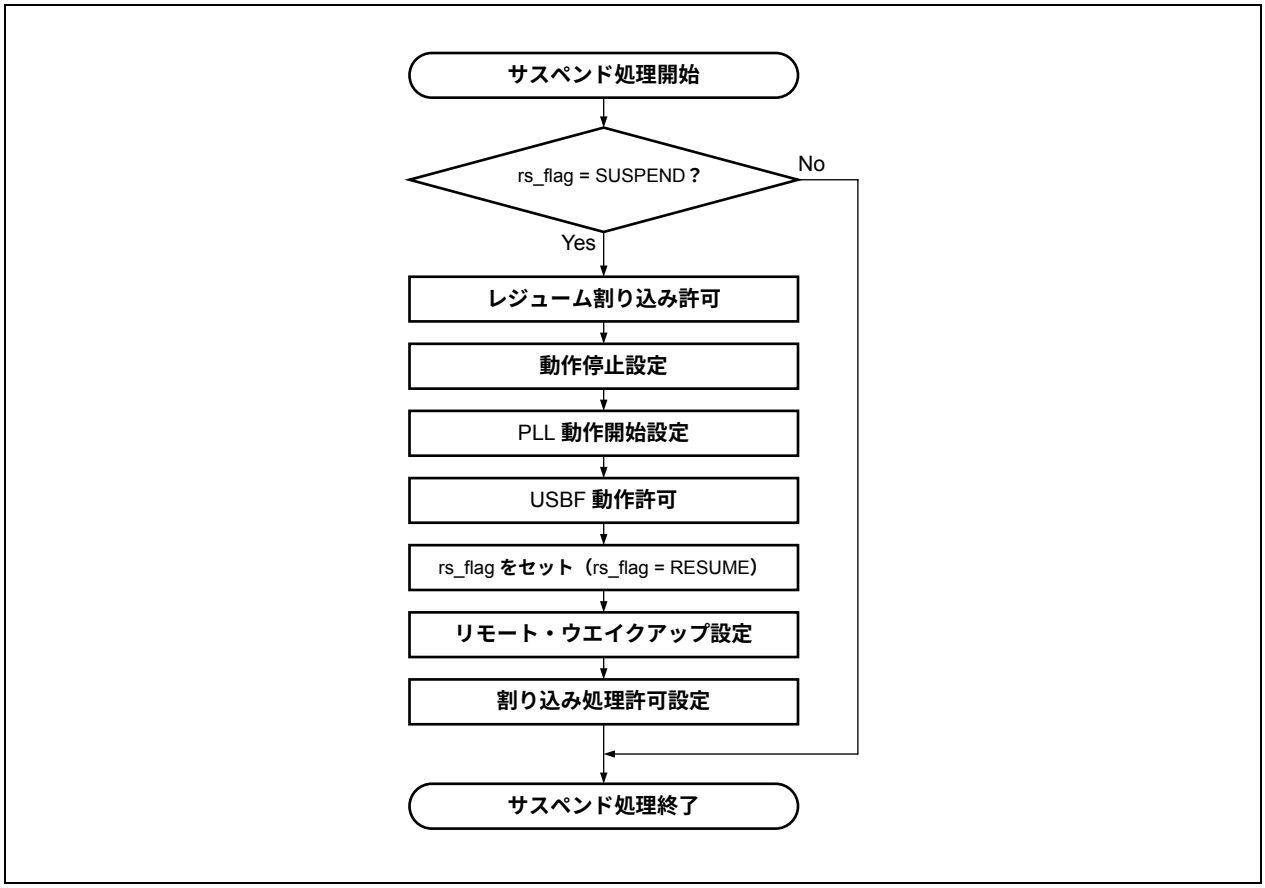
図 3−9　メイン・ルーチンの処理フロー



3. 4. 1　サスペンド／レジューム処理

レジューム／サスペンド・フラグ（rs_flg）をチェックし，"SUSPEND（0x00）" に設定されている場合，サスペンド処理を実行します。

図 3－10　サスペンド処理フロー



(1) レジューム／サスペンド・フラグ（rs_flag）の監視

サンプル・ドライバにより設定されるレジューム／サスペンド・フラグ（rs_flag）を監視します。このフラグが "SUSPEND（0x00）" の場合，USB バスがサスペンド状態になったことを示します。

(2) レジューム割り込み許可

外部割り込み（INTP0, INTP1），USBF レジューム割り込み（INTRSUM）を有効にします。
ここでは次に示すレジスタにアクセスします。

① MK0L レジスタの PMK1，PMK0 ビットに "0" を書き込みます。この設定により，INTP0, INTP1 割り込み処理が許可されます。
② MK1L レジスタの RSUMMK ビットに "0" を書き込みます。この設定により，INTRSUM 割り込み処理が許可されます。
③ IF1L レジスタの RSUMIF ビットに "0" を書き込みます。この設定により，USBF レジューム割り込み要因がクリアされます。
④ UF0IM0 レジスタの RSUSPDM ビットに "1" を書き込みます。この設定により，USB による Resume/Suspend 割り込みがマスクされます。

(3) 動作停止設定

μPD78F0730 をストップ・モードにします。ここでは次に示すレジスタにアクセスします。

① UF0BC レジスタの UBFIEN ビットに "0" を書き込みます。この設定により，USBF のバッファが無効になります。
② PLLC レジスタの PLLSTOP ビットに "1" を書き込みます。この設定により，PLL の動作が停止します。
③ STOP 命令を発行し，μPD78F0730 をストップ・モードにします。

(4) PLL 動作開始設定

PLL の動作を開始させます。

① OSTS レジスタで指定されている発振安定時間が経過してから，次の処理を実行します。
② PLLC レジスタの PLLSTOP ビットに "0" を書き込みます。この設定により，PLL の動作を開始します。
③ PLL の発振安定待ち時間（約 800 μs）が経過してから，次の処理を実行します。

(5) USBF 動作許可

USB ファンクション・コントローラの動作を設定します。ここでは次のレジスタにアクセスします。

① UCKC レジスタに "0x80" を書き込みます。この設定により，USB ファンクション・コントローラに内部クロックを供給します。
② UF0BC レジスタの UBFIEN ビットに "1" を書き込みます。この設定により，USBF のバッファが有効になります。

(6) レジューム／サスペンド・フラグ（rs_flag）のセット

レジューム／サスペンド・フラグ（rs_flag）を "RESUME（0x01）" に設定します。

(7) リモート・ウエイクアップ設定

リモート・ウエイクアップが許可されており，INTP0，INTP1 割り込みが発生している場合，リモート・ウエイクアップの処理を行います。リモート・ウエイクアップは次の手順で行います。

① UF0SDS レジスタの RSUMIN ビットに "1" を書き込みます。USB バス上にレジューム信号を出力させます。
② 1 ms 待ちます。
③ UF0SDS レジスタの RSUMIN ビットに "0" を書き込みます。この設定により，レジューム信号の出力を停止します。

(8) 割り込み処理許可設定

すべての割り込み要因をクリアし，割り込み要因 INTUSB0，INTUSB1，INTRSUM，INTP0，INTP1 のマスクを解除します。

① UF0ICn レジスタ（n = 0-4）のすべての有効ビットに "1" 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。

② UF0FIC0，UF0FIC1 レジスタのすべての有効ビットに "1" 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。

③ UF0IM0 レジスタの RSUSPDM ビットに "0" を書き込みます。RSUSPDM 割り込みのマスクが解除されます。

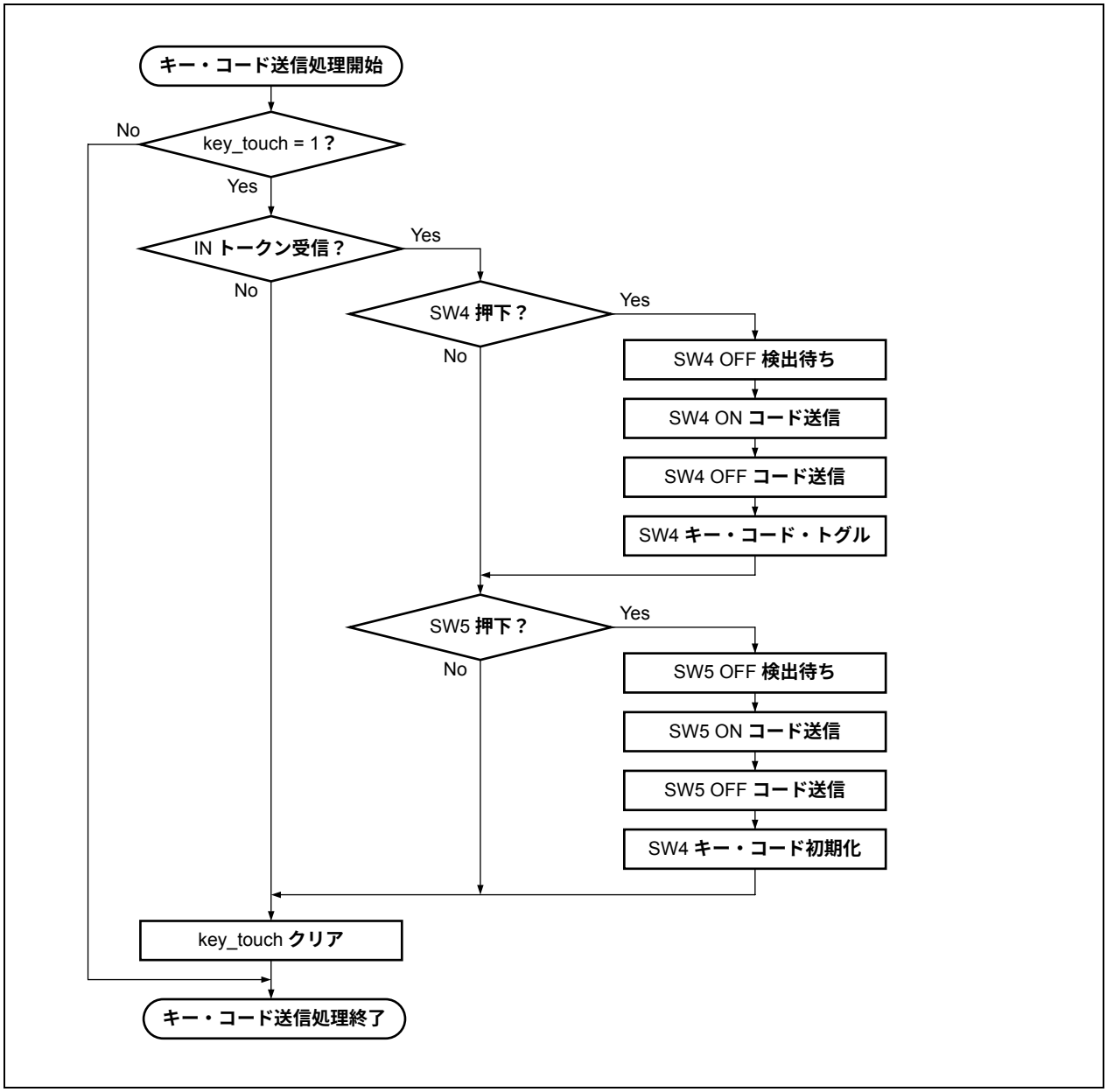
④ MK1L レジスタの RSUMMK ビット以外に "1" を書き込みます。この設定により，MK1L レジスタに示される割り込み要因のうち，INTRSUM 割り込み以外の要因がマスクされます。

⑤ IF1L レジスタの RSUMIF ビットに "0"，IF0L レジスタの USBIF0 ビットに "0" を書き込みます。この設定により，INTRSUM，INTUSB0 割り込み要因がクリアされます。

3. 4. 2　キー・コード送信処理

キー押下フラグ（key_touch）をチェックし，フラグがセットされている場合，押下されたキーのコードを USB で送信します。

図 3－11　キー・コード送信処理フロー



(1) キー押下の判断

キー押下フラグ（key_touch）をチェックし，セット（1）されている場合，処理を行います。

キー押下フラグは，外部割り込み処理（INTP0，INTP1）でセットされます。

(2) IN トークン受信確認

UF0IS2 レジスタの BKI1IN ビットがセット（1）されていること（USB バスで IN トークンを受信し NAK を送出していること）を確認します。

BKI1IN ビットが“1”の場合，割り込み要因をクリアし，押下されたキーの判断を行います。

BKI1IN ビットが“0”の場合，キー押下フラグ（key_touch）をクリア（0）して処理を終了します。

(3) SW4 押下判断

P120（ポート 12）が“0”のとき SW4 が押下されていると判断します。

(4) SW4 OFF 検出待ち

SW4 が押下されたあとに離される（P120 が“1”）まで待ちます。

(5) SW4 ON コード送信処理

SW4 が押下されたことを示すキー・コードをバルク・イン転送します。

(6) SW4 OFF コード送信処理

SW4 が離されたことを示すキー・コードをバルク・イン転送します。

(7) SW4 キー・コード・トグル

SW4 のキー・コードを更新します（アルファベットの更新）。

(8) SW5 押下判断

P30（ポート 3）が 0”のとき SW5 が押下されていると判断します。

(9) SW5 OFF 検出待ち処理

SW5 が押下されたあとに離される（P30 が“1”）まで待ちます。

(10) SW5 ON コード送信処理

SW5 が押下されたことを示すキー・コードをバルク・イン転送します。

(11) SW5 OFF コード送信処理

SW5 が離されたことを示すキー・コードをバルク・イン転送します。

(12) SW4 キー・コード初期化

SW4 のキー・コードを初期化します（“a”（0x04）に設定）。

3. 5　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. 5. 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 3－9　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 初期化
usbf78k.c	usbf78k_init	USB ファンクション・コントローラの初期化
	usbf78k_standardreq	標準リクエストの処理
	usbf78k_getdesc	GET_DISCRIPTOR リクエストの処理
	usbf78k_data_send	USB ホストへデータ送信
	usbf78k_data_receive	USB ホストからデータ受信
	usbf78k_clearFIFO	エンドポイント（FIFO）データのクリア
	usbf78k_sendnullEP0	Endpoint0 の NULL パケット発行
	usbf78k_sendstallEP0	Endpoint0 の STALL 応答
	usbf78k_sstall_ctrl	非対応リクエストの STALL 応答処理
	usbf78k_intusb0b	INTUSB0B 割り込みの処理
	usbf78k_intrsum	INTRSUM 割り込みの処理（処理なし）
	usbf78k_intp	INTP0，INTP1 割り込みの処理
usbf78k_human_interface.c	usbf78k_get_report	Get_Report リクエスト処理
	usbf78k_get_idle	Get_Idle リクエスト処理
	usbf78k_get_protocol	Get_Protocol リクエスト処理
	usbf78k_set_report	Set_Report リクエスト処理
	usbf78k_set_idle	Set_Idle リクエスト処理
	usbf78k_set_protocol	Set_Protocol リクエスト処理
	usbf78k_setfunction_human_interface	HID クラス・リクエスト処理関数登録
boot.asm	－	ブート処理ルーチン

3. 5. 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 3－12　メイン・ルーチンでの関数の呼び出し

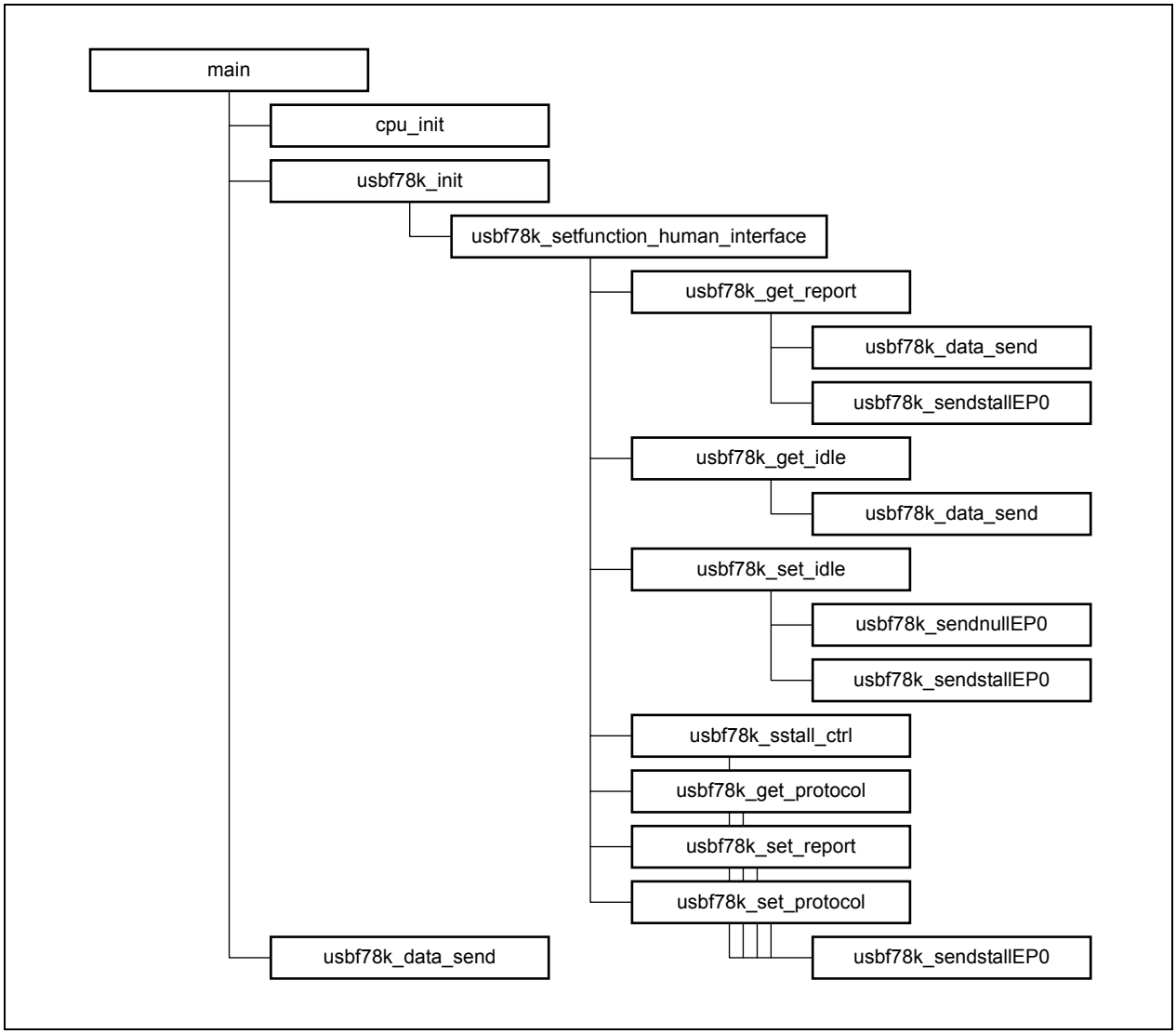
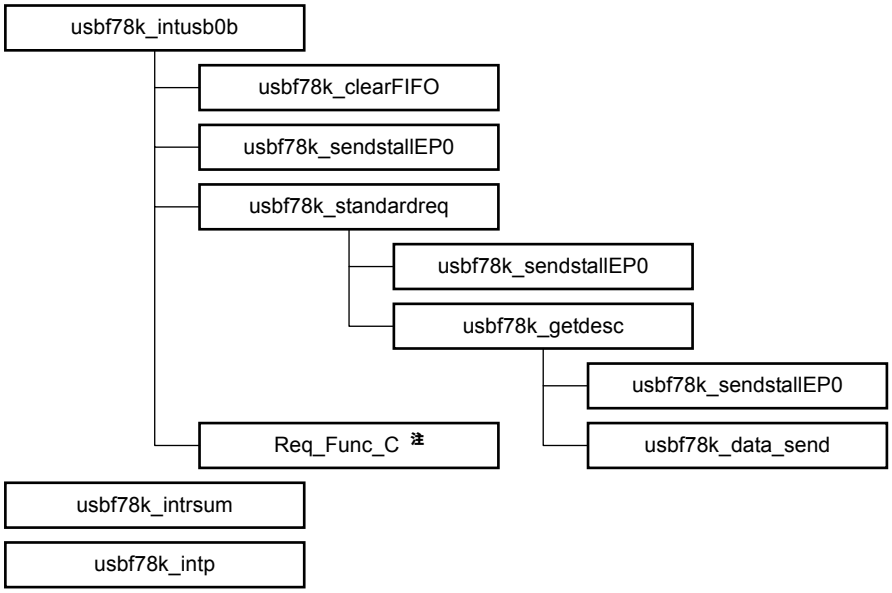


図 3－13　割り込み処理での関数の呼び出し



注　HID **クラス・リクエスト処理関数登録関数**（usbf78k_setfunction_human_interface）**で登録した関数の呼び出し処理**

3. 5. 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) メイン処理用の関数

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 初期化処理関数（cpu_init），USBF 初期化処理関数（usbf78k_init）を順に呼び出して初期化を実行します。
初期化処理を終了したら，ここで割り込みを待ちます。
サスペンド／レジューム割り込みが発生したら，サスペンド／レジューム処理を行います。また，SW 押下割り込みが発生したら，キー・コード送信処理を行います。

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 】
メイン処理で呼び出される関数です。
メモリ・サイズやクロック周波数など，μPD78F0730 を使用するために必要な項目を設定します。

(3) USB ファンクション・コントローラ用の関数

usbf78k_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 usbf78k_usbf_init(void)
```

【パラメータ】
なし

【 戻り値 】
なし

【 機　能 】
メイン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・コントローラを使用するために必要な項目を設定します。

usbf78k_intusb0b

【 概　要 】
INTUSB0B 割り込みハンドラ処理

【C 言語記述形式】
```
void usbf78k_intusb0b(void)
```

【パラメータ】
なし

【 戻り値 】
なし

【 機　能 】
INTUSB0B 割り込みが発生したときに実行する関数です。
割り込み要因が RSUSPD，BUSRST，SETRQ および CPUDEC の場合，各要求に合わせて処理を実行します。
割り込み要因が CPUDEC の場合，リクエスト・タイプから該当するリクエストの処理を行います。

usbf78k_intrsum

【 概　要 】

INTRSUM 割り込みハンドラ処理

【C 言語記述形式】

```
void usbf78k_intrsum(void)
```

【パラメータ】

なし

【 戻り値 】

なし

【 機　能 】

なし

usbf78k_intp

【 概　要 】

INTP0，INTP1 割り込みハンドラ処理

【C 言語記述形式】

```
void usbf78k_intp(void)
```

【パラメータ】

なし

【 戻り値 】

なし

【 機　能 】

キー押下フラグ（key_touch）をセット（1）します。

usbf78k_data_send

【 概　要 】

USB データ送信処理

【C 言語記述形式】

```
INT32 usbf78k_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	送信データ長
INT8 ep	データ送信エンドポイント番号

【 戻り値 】

シンボル	説　明
DEV_OK	正常終了
DEV_ERROR	異常終了

【 機　能 】

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

usbf78k_data_receive

【 概　要 】

USB データ受信処理

【C 言語記述形式】

```
INT32 usbf78k_data_receive(UINT8 *data, INT32 len, INT8 ep)
```

【パラメータ】

パラメータ	説　　明
UINT8 *data	受信データ・バッファ・ポインタ
INT32 len	受信データ長
INT8 ep	データ受信エンドポイント番号

【 戻り値 】

シンボル	説　　明
DEV_OK	正常終了
DEV_ERROR	異常終了

【 機　能 】

指定したエンドポイント用の FIFO からデータを 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。

usbf78k_clearFIFO

【 概　要 】

FIFO クリア処理

【C 言語記述形式】

```
void usbf78k_clearFIFO(INT8 ep)
```

【パラメータ】

パラメータ	説　　明
INT8 ep	エンドポイント番号

【 戻り値 】

なし

【 機　能 】

指定したエンドポイントのすべての FIFO をクリアします。

usbf78k_sendnullEP0

【 概　要 】

Endpoint0 用 NULL パケット送信処理

【C 言語記述形式】

```
void usbf78k_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。

usbf78k_sendstallEP0

【 概　要 】

Endpoint0 用 STALL 応答処理

【C 言語記述形式】

```
void usbf78k_sendstallEP0(void)
```

【パラメータ】

なし

【 戻り値 】

なし

【 機　能 】

STALL ハンドシェー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。

usbf78k_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 usbf78k_standardreq(void)
```

【パラメータ】

なし

【 戻り値 】

なし

【 機　能 】

USB 割り込み処理（INTUSB0B）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（usbf78k_getdesc）を呼び出します。それ以外のリクエストの場合は STALL 応答します。

usbf78k_getdesc

【 概　要 】

GET_DESCRIPTOR リクエスト処理

【C 言語記述形式】

```
void usbf78k_getdesc(void)
```

【パラメータ】

なし

【 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ストリング／HID／レポート・ディスクリプタを要求している場合，USB データ送信処理関数（usbf78k_data_send）を呼び出して，Endpoint0 から対象のディスクリプタを送信します。
それ以外のディスクリプタを要求している場合は STALL 応答します。

usbf78k_sstall_ctrl

【 概　要 】

非対応リクエストの STALL 応答処理

【C 言語記述形式】

```
void usbf78k_sstall_ctrl(void)
```

【パラメータ】

なし

【 戻り値 】

なし

【 機　能 】

HID クラス・リクエスト処理関数登録関数（usbf78k_setfunction_human_interface）で非対応リクエスト受信時の処理として登録される関数です。リクエスト・データ受信時に INTUSB0B 割り込みハンドラ処理関数（usbf78k_intusb0b）から呼び出されます。
STALL 応答します。

(4) クラス・リクエスト処理

usbf78k_get_report

【 概　要 】

Get_Report リクエスト処理

【C 言語記述形式】

```
void usbf78k_get_report(void)
```

【パラメータ】

なし

【 戻り値 】

なし

【 機　能 】

HID クラス・リクエスト処理関数登録関数（usbf78k_setfunction_human_interface）でリクエスト受信時の処理として登録される関数です。リクエスト・データ受信時に INTUSB0B 割り込みハンドラ処理関数（usbf78k_intusb0b）から呼び出されます。

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。

usbf78k_get_idle

【 概　要 】

Get_Idle リクエスト処理

【C 言語記述形式】

```
void usbf78k_get_idle(void)
```

【パラメータ】

なし

【 戻り値 】

なし

【 機　能 】

HID クラス・リクエスト処理関数登録関数（usbf78k_setfunction_human_interface）でリクエスト受信時の処理として登録される関数です。リクエスト・データ受信時に INTUSB0B 割り込みハンドラ処理関数（usbf78k_intusb0b）から呼び出されます。

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。

usbf78k_get_protocol

【 概　要 】

Get_Protocol リクエスト処理

【C 言語記述形式】

```
void usbf78k_get_protocol(void)
```

【パラメータ】

なし

【 戻り値 】

なし

【 機　能 】

HID クラス・リクエスト処理関数登録関数（usbf78k_setfunction_human_interface）でリクエスト受信時の処理として登録される関数です。リクエスト・データ受信時に INTUSB0B 割り込みハンドラ処理関数（usbf78k_intusb0b）から呼び出されます。

STALL 応答します。

usbf78k_set_report

【 概　要 】

Set_Report リクエスト処理

【C 言語記述形式】

```
void usbf78k_set_report(void)
```

【パラメータ】

なし

【 戻り値 】

なし

【 機　能 】

HID クラス・リクエスト処理関数登録関数（usbf78k_setfunction_human_interface）でリクエスト受信時の処理として登録される関数です。リクエスト・データ受信時に INTUSB0B 割り込みハンドラ処理関数（usbf78k_intusb0b）から呼び出されます。

STALL 応答します。

usbf78k_set_idle

【 概　要 】

Set_Idle リクエスト処理

【C 言語記述形式】

```
void usbf78k_set_idle(void)
```

【パラメータ】

なし

【 戻り値 】

なし

【 機　能 】

HID クラス・リクエスト処理関数登録関数（usbf78k_setfunction_human_interface）でリクエスト受信時の処理として登録される関数です。リクエスト・データ受信時に INTUSB0B 割り込みハンドラ処理関数（usbf78k_intusb0b）から呼び出されます。

ホストが送信したアイドル率が“ 0 ”の場合，NULL パケットで応答します。

“ 0 ”以外の場合，STALL 応答します。

usbf78k_set_protocol

【 概　要 】

Set_Protocol リクエスト処理

【C 言語記述形式】

```
void usbf78k_set_protocol(void)
```

【パラメータ】

なし

【 戻り値 】

なし

【 機　能 】

HID クラス・リクエスト処理関数登録関数（usbf78k_setfunction_human_interface）でリクエスト受信時の処理として登録される関数です。リクエスト・データ受信時に INTUSB0B 割り込みハンドラ処理関数（usbf78k_intusb0b）から呼び出されます。

STALL 応答します。

usbf78k_setfunction_human_interface

【 概　要 】

HID クラス・リクエスト処理関数登録

【C 言語記述形式】

```
void usbf78k_setfunction_human_interface(void)
```

【パラメータ】

シンボル	説　　明
UINT8 len	データ長

【 戻り値 】

なし

【 機　能 】

USBF 初期化処理で呼び出される関数です。
配列 Req_Func_C に実行するリクエスト処理関数を登録します。

3. 6　データ構造体

USB デバイス・リクエスト構造体は，"usbf78k.h" ファイルで定義されています。

プログラム中ではグローバル変数 "UsbSetup_Data" として使用します。

```
/*------------------------------------------------------------
 * SETUP DATA structure
 *------------------------------------------------------------*/
typedef struct {
     UINT8    RequstType;              /* bmRequestType  */
     UINT8    Request;                 /* bRequest       */
     UINT16   Value;                   /* wValue         */
     UINT16   Index;                   /* wIndex         */
     UINT16   Length;                  /* wLength        */
     UINT8*   Data;                    /* index to Data  */
} USB_SETUP;

/*------------------------------------------------------------
 *   global variable
 *------------------------------------------------------------*/
extern USB_SETUP      UsbSetup_Data;
```

第4章　開発環境

この章では，μPD78F0730 向け USB-シリアル変換用サンプル・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の例と，そこでのデバッグの手順について説明します。

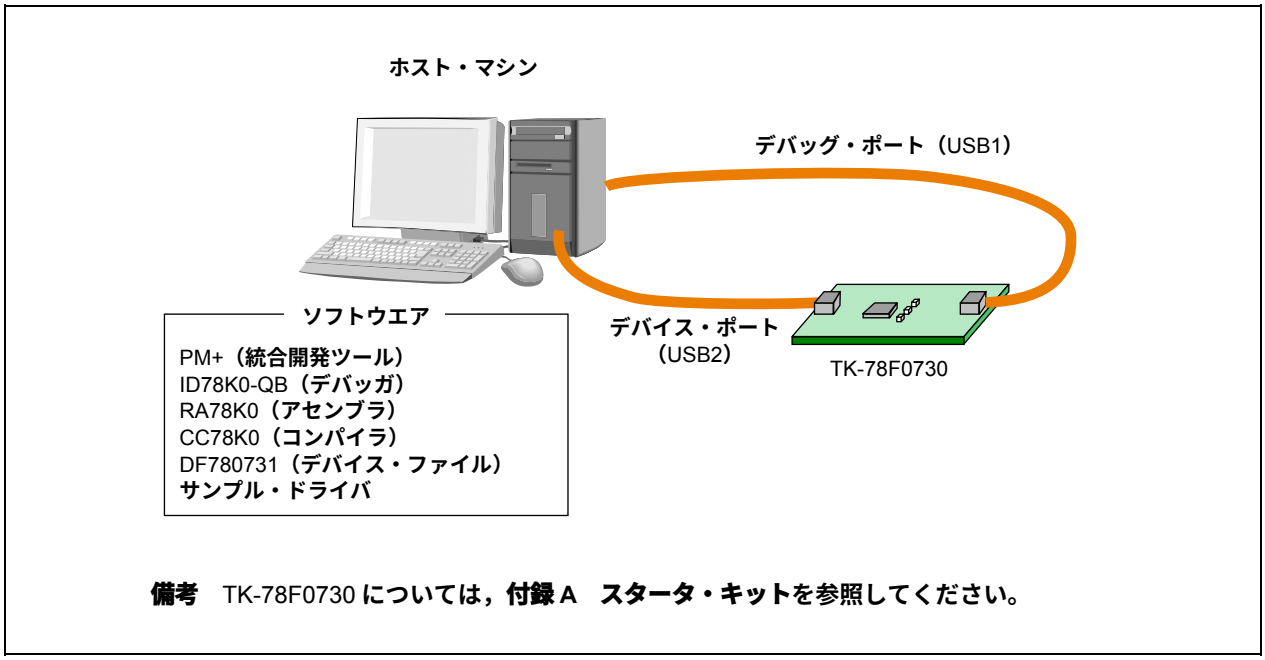
4. 1　製品構成

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

4. 1. 1　システム構成

サンプル・ドライバを利用するシステムの構成を図 4－1に示します。

図 4－1　システムの構成



備考　TK-78F0730 については，**付録 A　スタータ・キット**を参照してください。

4.1.2 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には，次のハードウエアとソフトウエアが必要です。

表 4－1　プログラム開発環境構成例

構成品		製品例	備　　考
ハードウエア	ホスト・マシン	－	PC/AT®互換機（OS：Windows® XP または Windows Vista®）
ソフトウエア	統合開発ツール	PM+	V6.30
	アセンブラ	RA78K0	W4.00
	コンパイラ	CC78K0	W4.01
ファイル	デバイス・ファイル	DF780731	V1.10（μPD78F0730 用）
	ソース・ファイル	サンプル・ソフトウエア	
	インクルード・ファイル		

4.1.3 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には，次のハードウエアとソフトウエアが必要です。

表 4－2　デバッグ環境構成例

構成品		製品例	備　　考
ハードウエア	ホスト・マシン	－	PC/AT 互換機（OS：Windows XP または Windows Vista）
	ターゲット	TK-78F0730 注1	テセラ・テクノロジー社製
	ケーブル類	－	USB ケーブルなど
ソフトウエア	統合開発ツール	PM+	V6.30
	デバッガ	ID78K0-QB	V3.00
ファイル	デバイス・ファイル	DF780731	V1.10（μPD78F0730 用）
	ソース・ファイル	サンプル・ソフトウエア	
	インクルード・ファイル		
	プロジェクト関連ファイル		注 2

注 1. TK-78F0730 については，付録 A　スタータ・キットを参照してください。

2. PM+で構築した場合のファイルがサンプル・ソフトウエアに同梱されています。

4. 2　環境設定

ここでは，4. 1　**製品構成**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説明します。

4. 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。

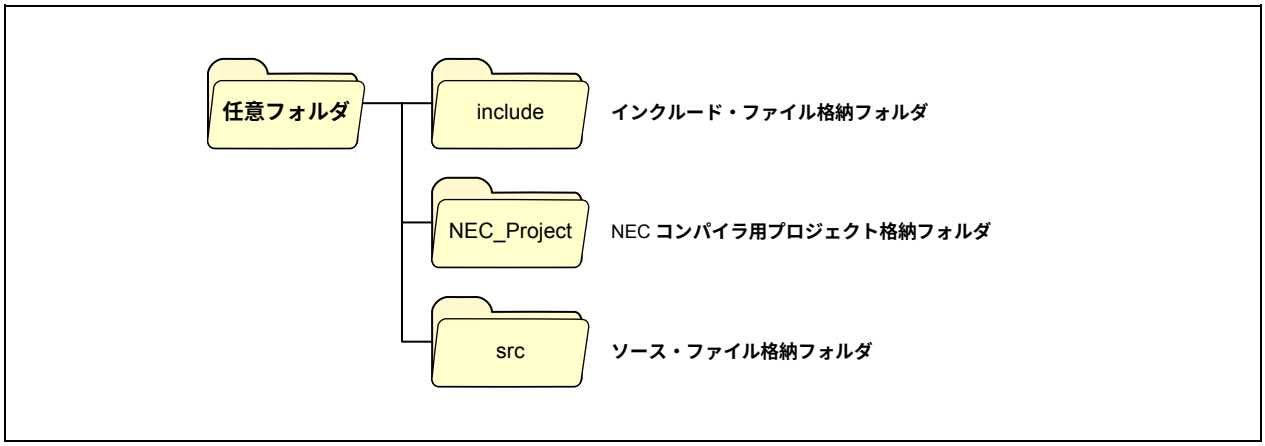
(1) 統合開発ツールのインストール

PM+をインストールします。詳細は PM+のユーザーズ・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。
また，デバッグ・ポート用ホスト・ドライバを任意のディレクトリに格納します。

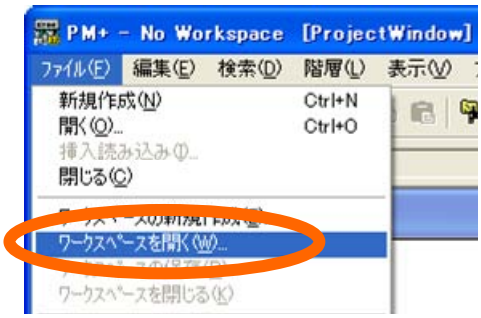
図 4−2　サンプル・ドライバのフォルダ構成



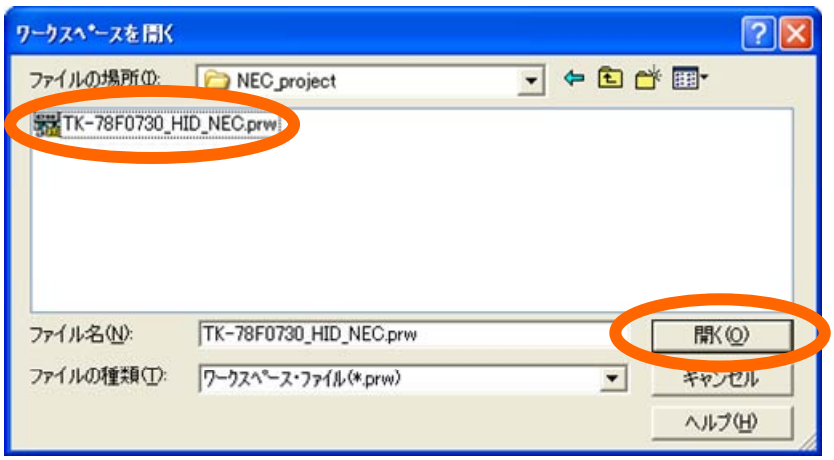
(3) ワークスペースの設定

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。



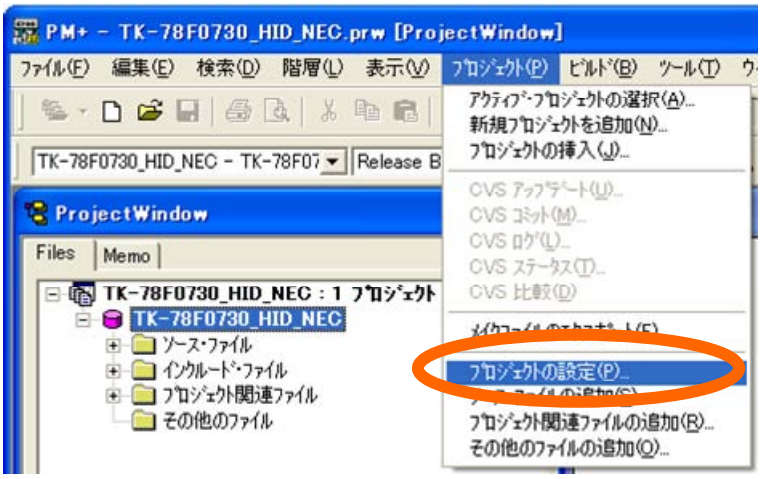
<2>　「ワークスペースを開く」ダイアログが開きます。サンプル・ドライバを格納したディレクトリの「NEC_project」フォルダにあるワークスペース・ファイルを指定します。



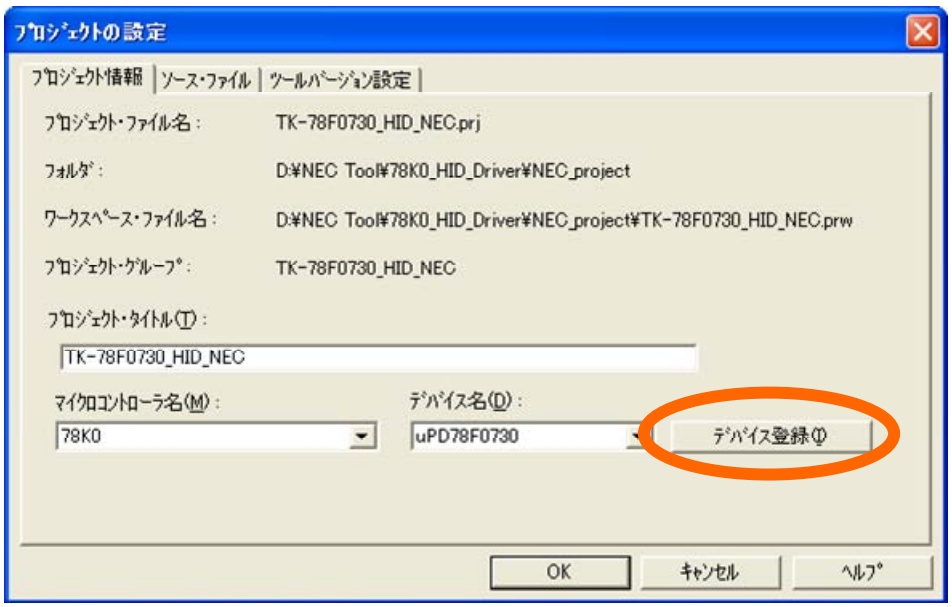
(4) デバイス・ファイルのインストール

ここではμPD78F0730 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。

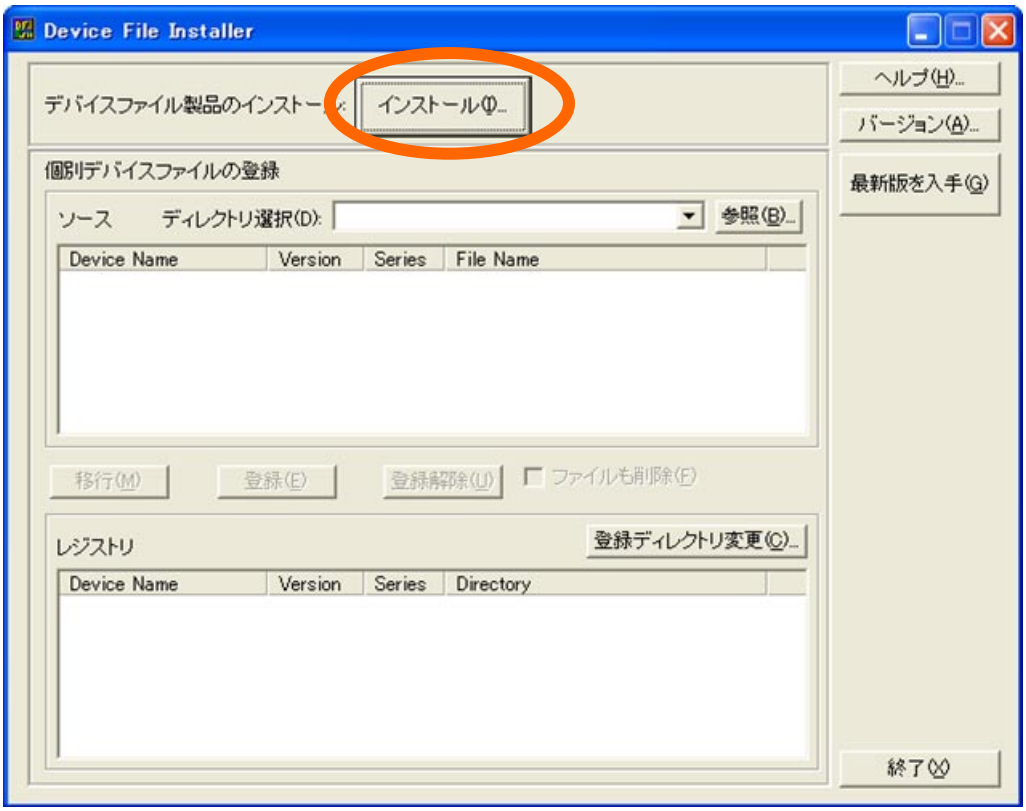
<1>　PM+の「プロジェクト」メニューから「プロジェクトの設定」を選択します。



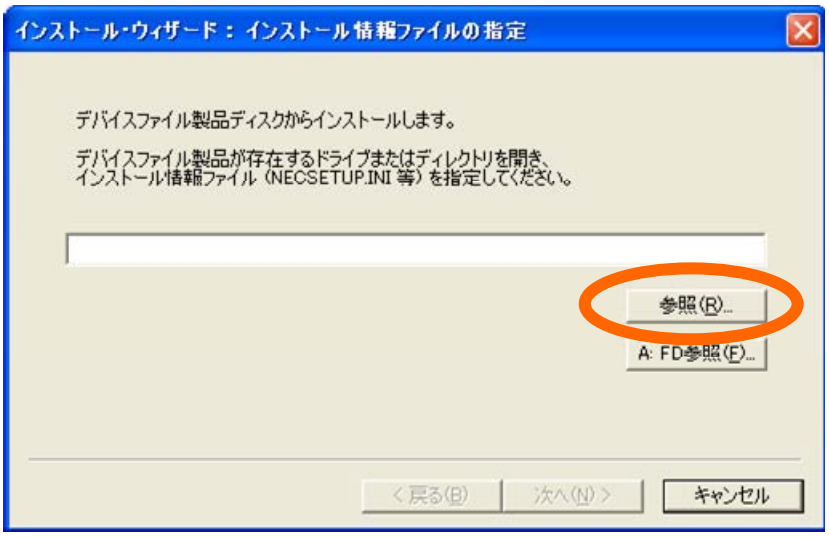
<2>　「プロジェクトの設定」ダイアログが開きます。「プロジェクト情報」タブの「デバイス登録」ボタンを押下して Device File Installer を起動します。



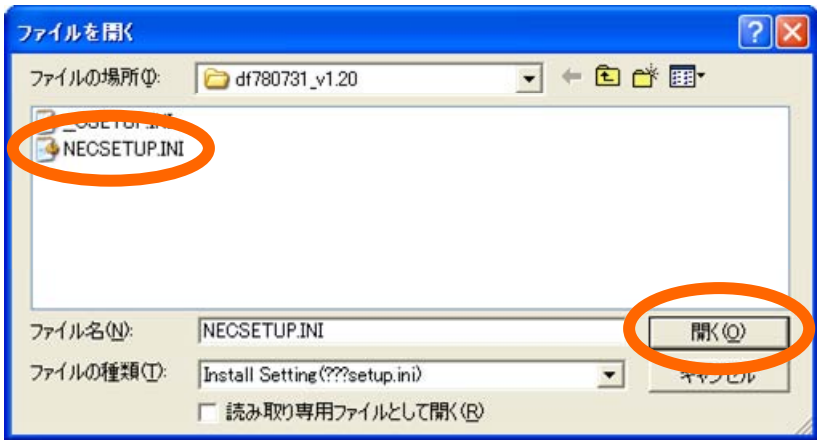
<3>　「Device File Installer」ダイアログが開きます。「インストール」ボタンを押下してインストール・ウィザードを起動します。



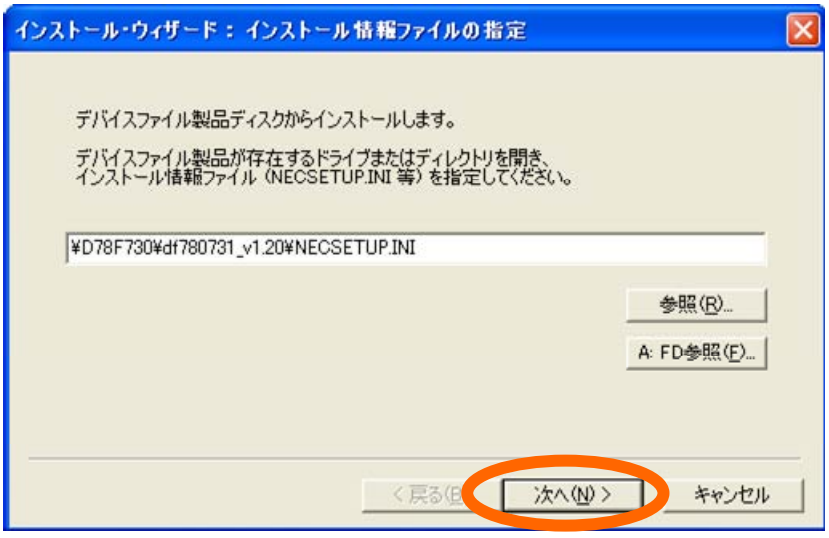
<4>　「インストール情報ファイルの指定」ダイアログが開きます。「参照」ボタンを押下します。



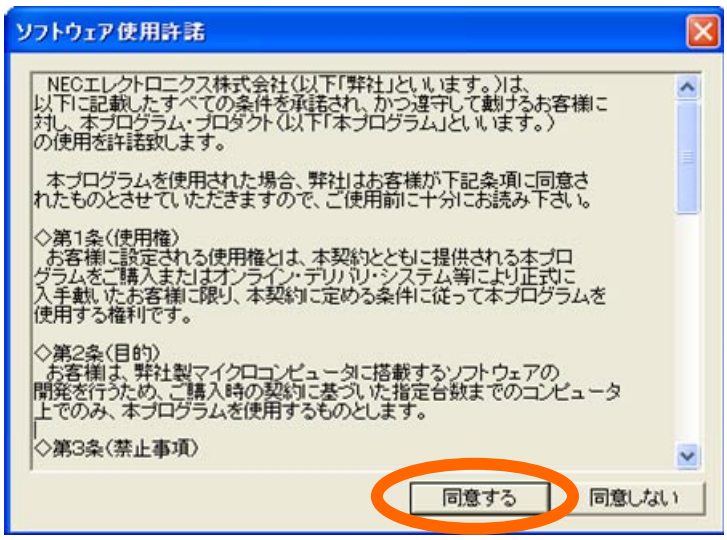
<5>　「ファイルを開く」ダイアログが表示されます。デバイス・ファイルを格納したディレクトリを開き，"NECSETUP.INI" ファイルを選択して「開く」ボタンを押下します。



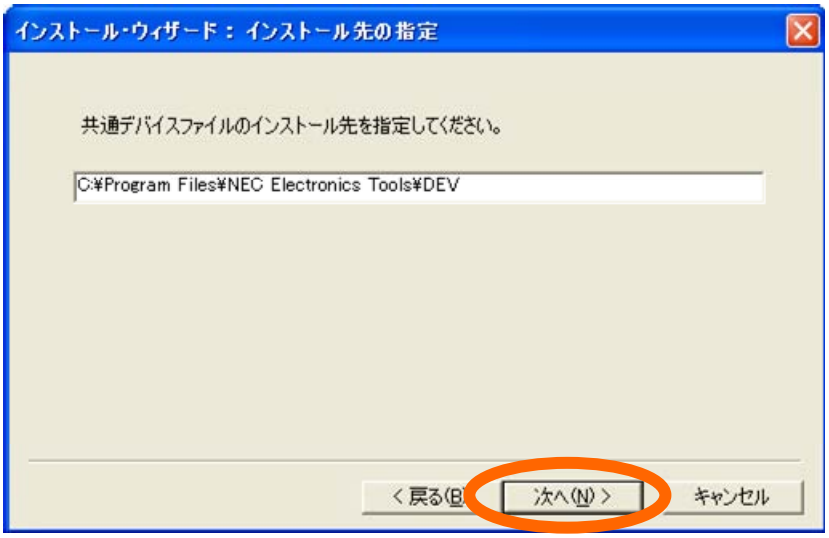
<6>　「インストール情報ファイルの指定」ダイアログに戻ります。「次へ」ボタンを押下します。



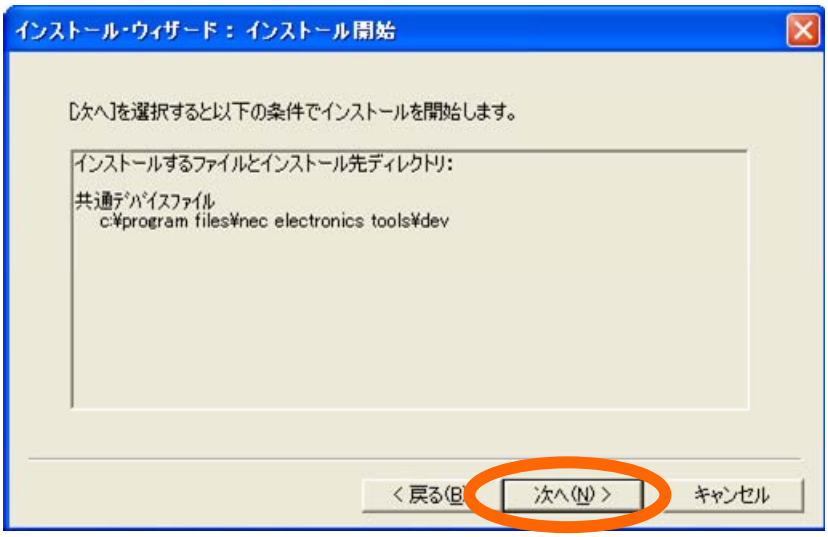
<6> 使用許諾に関するメッセージが表示されます。使用許諾に同意する場合は「同意する」ボタンを押下します。



<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。

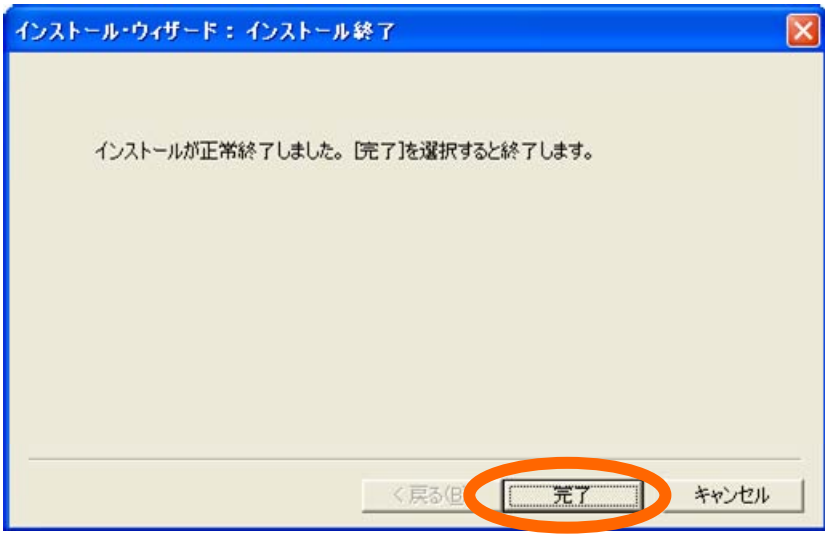


<9>　「インストール開始」ダイアログが開きます。「次へ」ボタンを押下します。



<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。

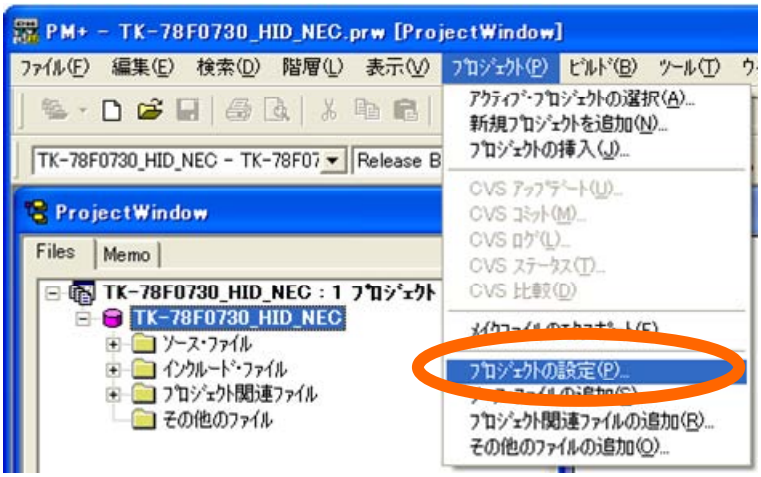
<11> インストールが終わると「インストール終了」ダイアログが開きます。「完了」ボタンを押下します。



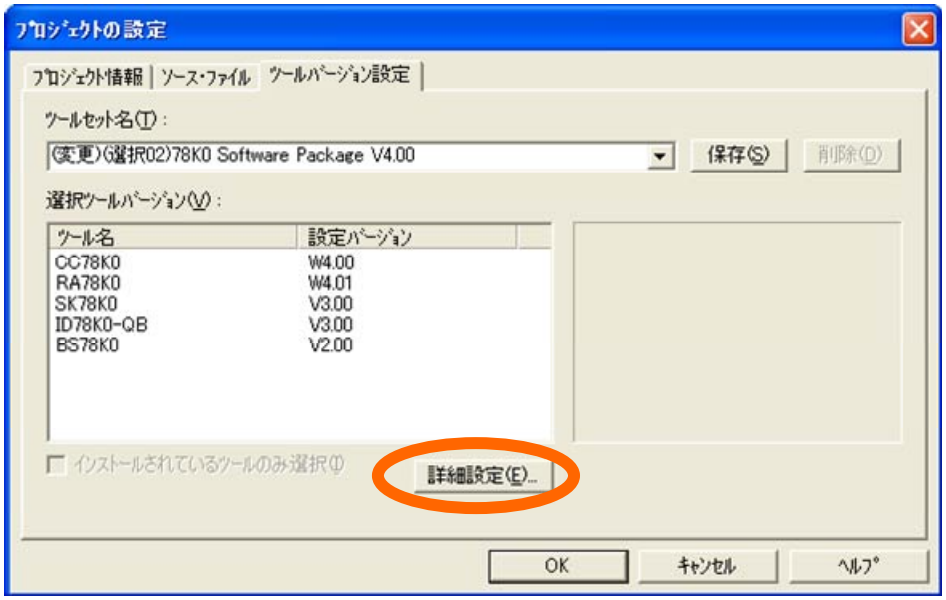
(5) ビルド・ツールの設定

ここではビルド・ツールとして CC78K0 を，デバッグ・ツールとして ID78K0-QB を使用する場合の手順を示します。

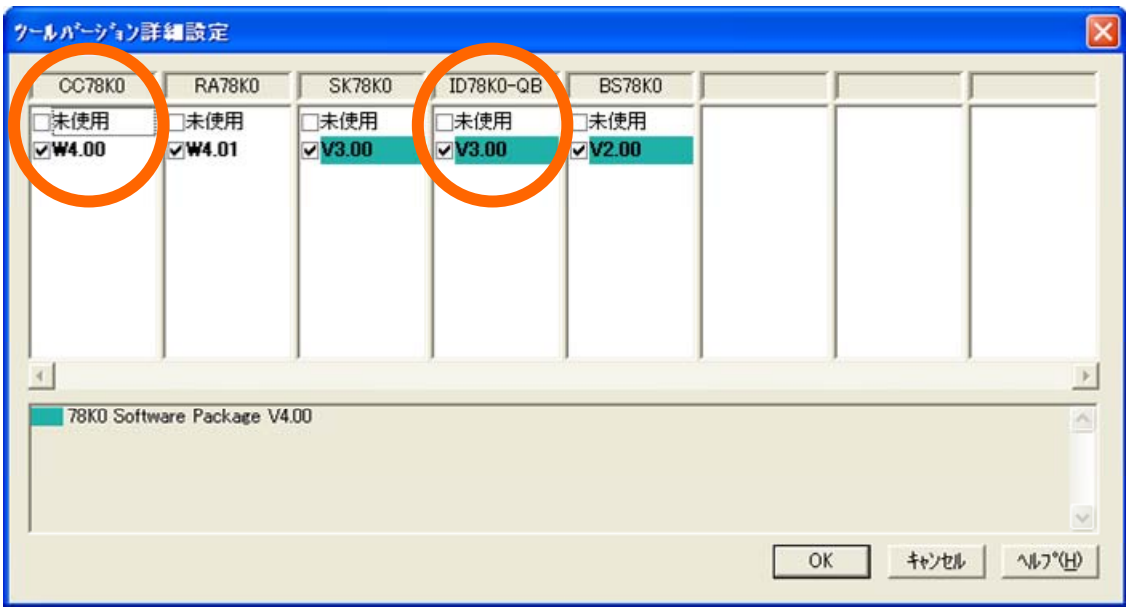
<1> PM+の「プロジェクト」メニューから「プロジェクトの設定」を選択します。



<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。「CC78K0」の欄で使用するコンパイラのバージョンを，「ID78K0-QB」の欄で使用するデバッガのバージョンを選択します。



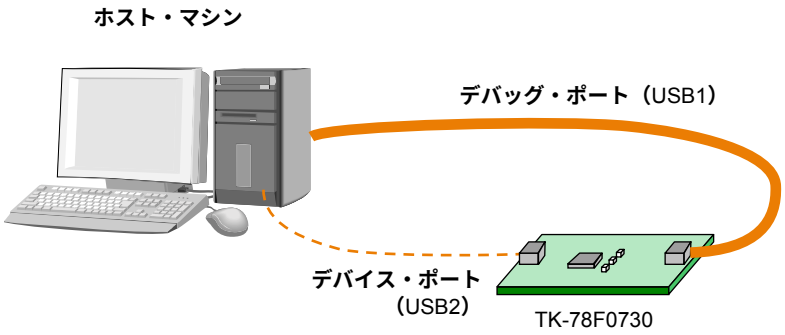
4. 2. 2　ターゲット環境整備

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-78F0730 のデバッグ・ポート（USB1）とホスト・マシンの USB ポートを USB ケーブルで接続します。

図 4−3　ターゲット・ボードの接続



備考　ここではデバイス・ポート（USB2）を接続しません。

(2) ホスト・ドライバのインストール

デバッグ・ポート（USB1），デバイス・ポート（USB2）を使用してホスト・マシンに接続するには，ドライバをインストールする必要があります。

(a) デバッグ・ポート（USB1）

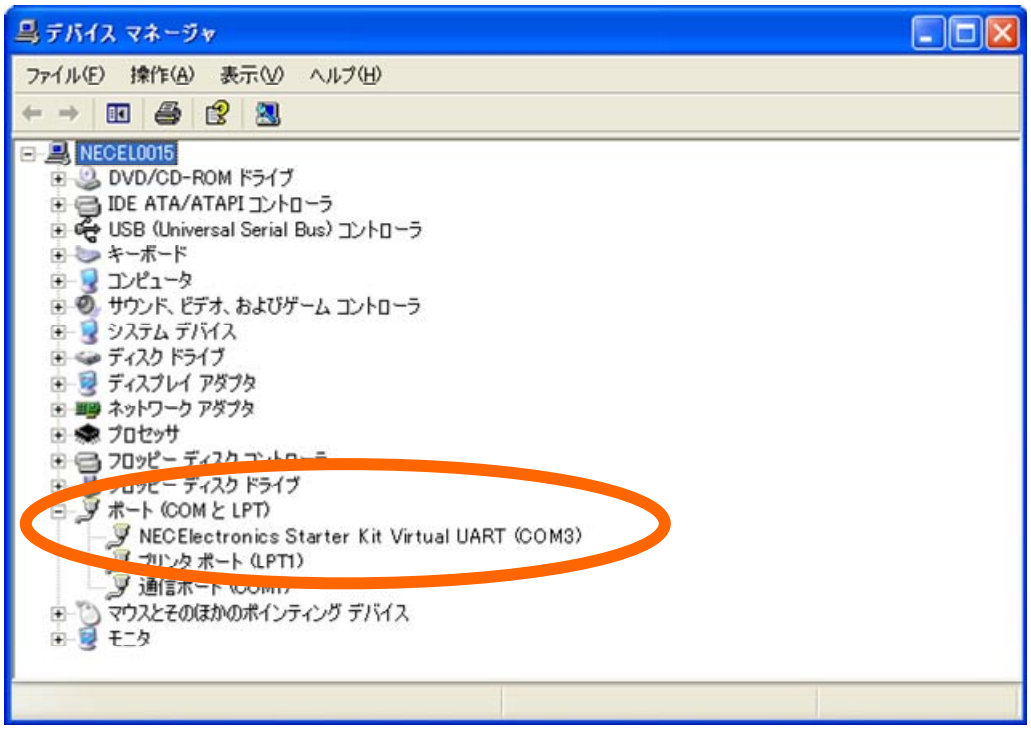
TK-78F0730 の一方の USB ポート（USB1）はデバッグ・ポートです。このポートを使用するには，別途専用のホスト・ドライバが必要になります。インストール方法については，TK-78F0730 のユーザーズ・マニュアルを参照してください。

(b) デバイス・ポート（USB2）

デバイス・ポートのドライバには，Windows 標準の HID クラス用ホスト・ドライバを使用します。詳細は4. 4　動作確認を参照してください。

(3) デバイス割り当ての確認

Windows のデバイス マネージャを開きます。「ポート（COM と LPT)」のツリーを展開し，デバッグ・ポート「NEC Electronics Starter Kit Virtual UART 注」が表示されていることを確認します。



注　表示されるドライバの名称はインストールするドライバによって変わります。詳細は，TK-78F0730 のユーザーズ・マニュアルを参照してください。

4. 3　オンチップ・デバッグ

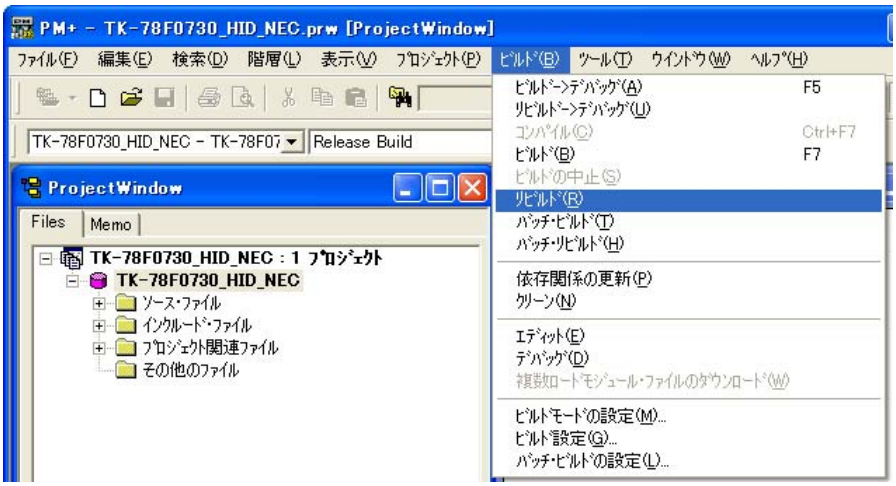
ここでは，4. 2　**環境設定**に示したワークスペースで開発したアプリケーション・プログラムのデバッグ手順について説明します。

μPD78F0730 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。

4. 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。



4. 3. 2　ロードと実行

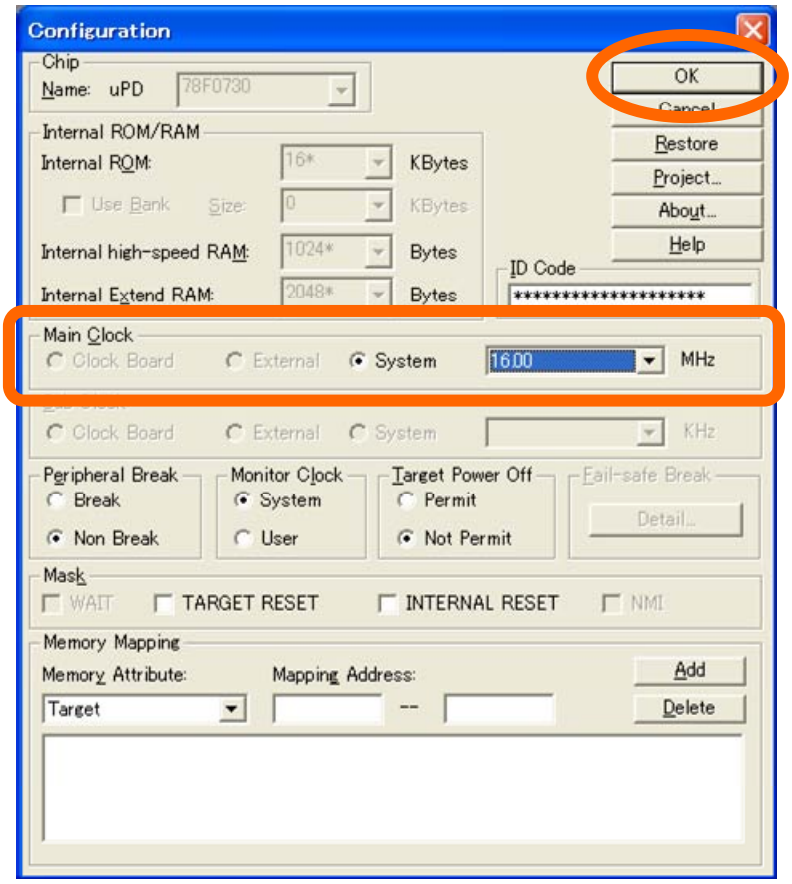
生成したロード・モジュールをターゲットに書き込んで（ロード）実行させます。

(1) ロード・モジュールの書き込み

ターゲット・ボード上のμPD78F0730 に PM+を介してロード・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 ID78K0-QB-EZ を起動します。

<2>　「Configuration」ダイアログが開きます。「Main Clock」の設定を確認し，「OK」ボタンを押下します。



(2) プログラムの実行

ID78K0-QB-EZ 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。

4. 4　動作確認

ここでは，サンプル・ドライバのプログラムを実行したあと，実行結果を確認する手順を示します。

(1) デバイス・ポートの接続

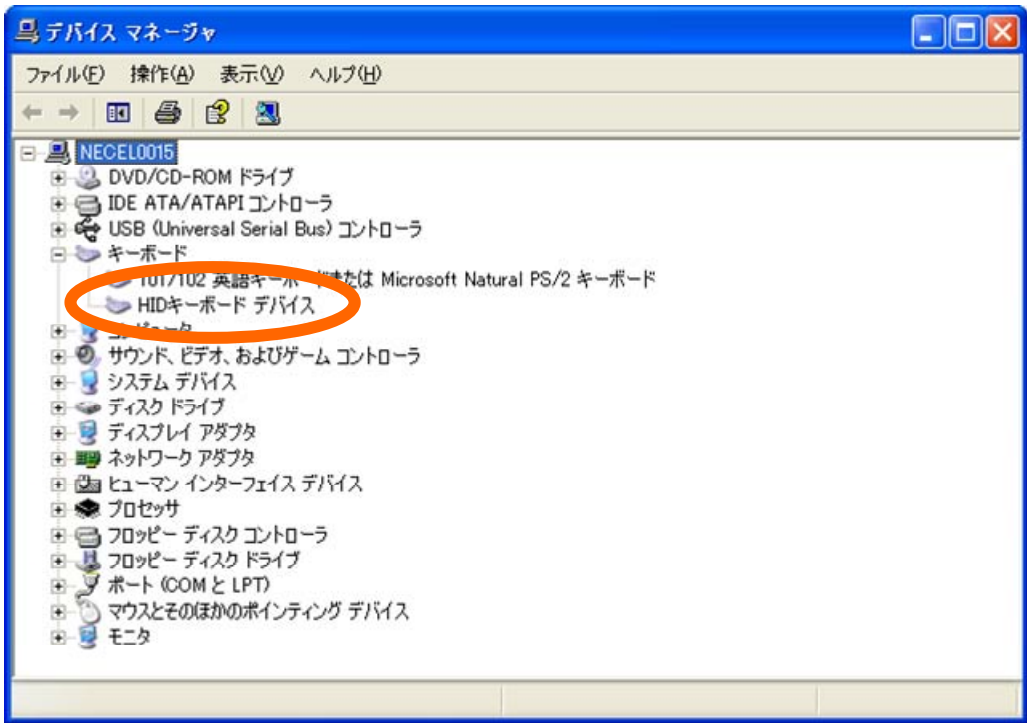
TK-78F0730 のデバイス・ポート（USB2）とホスト・マシンの USB ポートを USB ケーブルで接続します。

(2) ホスト・ドライバのインストール

デバイス・ポートのドライバには，Windows 標準の HID クラス用ホスト・ドライバを使用します。サンプル・ドライバを実行している状態でホスト・マシンに接続すると，自動的にドライバがインストールされます。

(3) USB デバイスの接続確認

Windows のデバイス マネージャを開きます。「キーボード」のツリーを展開し，「HID キーボード デバイス」が表示されていることを確認します。

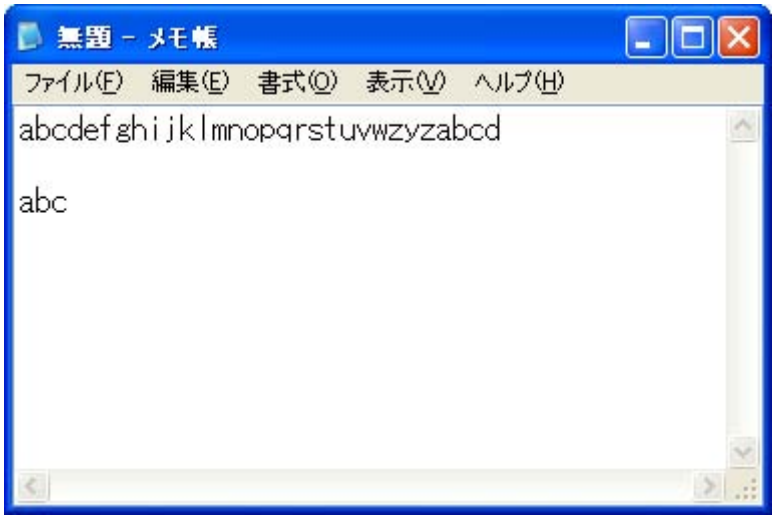


(4) 送信情報の確認

「メモ帳」を開き，次の動作を確認します。

- SW4 押下時にアルファベット「a」～「z」のキー・コード（ASCII）を順次トグルして出力（「z」の次は「a」として循環）。
- SW5 押下時は「Enter」キー・コードを出力。

たとえば，SW4 を 30 回，SW5 を 2 回，SW4 を 3 回の順に押下すると，次のようになります。



第5章　サンプル・ドライバの応用

この章では，μPD78F0730 向け USB HID 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。

5.1 概　　要

サンプル・ドライバの利用には，主に次の 2 つの方法が考えられます。

(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" ファイル内のアプリケーション部
- "usbf78k_human_interface.c" ファイル内のリクエスト処理
- "usbf78k_sfr.h" ファイル内の各種レジスタの設定値
- "usbf78k_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，**3.5　関数の仕様**を参照してください。

5. 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。

5. 2. 1　アプリケーション部

"main.c" ファイルのメイン・ルーチン処理関数（main）には，サンプル・ドライバの利用例として簡単な処理を記述しています。実際にアプリケーションで使用する処理をこの部分に記述することで，既存の初期化処理や割り込み処理をそのまま利用できます。

リスト 5−1　キー・コード送信処理の記述

```
1   if( key_touch == 1 ){
2       if(( UF0IS2 & 0x20 ) == 0x20 ){                 /* BKI1IN */
3           UF0IC2 = 0xDF;
4           /* SW4 */
5           if(( P12 & 0x01 ) == 0x00 ){
6               while(( P12 & 0x01 ) == 0x00 ){
7                   /* SW4 OFF WAIT */
8               }
9               keycode[KEY1_SCAN_CODE] = keydata;       /* Press Key Data */
10              usbf78k_data_send(keycode, sizeof(keycode), BKI1);
11              memset(keycode, 0, sizeof(keycode));     /* Release Key Data */
12              usbf78k_data_send(keycode, sizeof(keycode), BKI1);
13              keydata++;                               /* a to z */
14              if(keydata == EXCLAMATION_KEY) {
15                  keydata = A_KEY;
16              }
17          }
18
19          /* SW5 */
20          if(( P3 & 0x01 ) == 0x00 ) {
21              while(( P3 & 0x01 ) == 0x00 ){
22                  /* SW5 OFF WAIT */
23              }
24              keycode[KEY1_SCAN_CODE] = ENTER_KEY;     /* Press Key Data(Enter) */
25              usbf78k_data_send(keycode, sizeof(keycode), BKI1);
26              memset(keycode, 0, sizeof(keycode));     /* Release Key Data */
27              usbf78k_data_send(keycode, sizeof(keycode), BKI1);
28              keydata = 0x04;
29          }
30      }
31      key_touch = 0;
32  }
```

たとえば，SW4 を押下したときに出力されるキー・コードを 0x11（N キー）に固定したい場合，リスト 5−1に示すキー・コード送信処理部分の 9 行目の「keydata」を「0x11」に書き換えます。

リスト 5−2　SW4 のキー・コードを 0x11（N キー）に固定する場合の記述

```
    :
9               keycode[KEY1_SCAN_CODE] = 0x11;          /* Press Key Data */
    :
```

5. 2. 2　リクエスト処理

"usbf78k_human_interface.c" ファイルには，クラス・リクエストを受信したときに実行する関数が記述されています。

サンプル・ドライバは，Get_Protocol，Set_Report，Set_Protocol リクエストを受信しても，STALL 応答しか行いません。この関数内にリクエストに対応する処理を記述することで，各リクエストに対する処理を実装できます。

リスト 5－3　Set_Report リクエスト処理関数

```
/*============================================================
 For Human Interface Device Class
 void usbf78k_set_report(void)

 Arguments:
       N/A
 Return values:
       N/A
 Overview:
       It accepts Set_Report request.
============================================================*/
void usbf78k_set_report(void)
{
       /* not supported */
       usbf78k_sendstallEP0();
}

/*============================================================
```

5. 2. 3　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，"usbf78k_sfr.h" ファイルに定義されています（3. 2. 2　USBF 初期化処理参照）。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・デバイスの動作を設定できます。

たとえば，サンプル・ドライバでは，D+信号をプルアップするときは UF0GPR レジスタに 0x02 を書き込みます（USBPUC 端子はハイ・アクティブです）。このため，USBPUC 端子がロウ・アクティブの回路構成の場合，"usbf78k_sfr.h" ファイル内の C_CONNECT を 0x00，C_UCONNECT を 0x02 に書き換えます。

リスト 5－4　"usbf78k_sfr.h"の UF0GPR レジスタ設定部

```
   :
/* UF0GPR register */
#define C_MRST       0x01          /* same result almost as H/W Reset */

#define C_CONNECT    0x02          /* D+ PULLUP ON(HI ACT)  */
#define C_UCONNECT   0x00          /* D+ PULLUP OFF(HI ACT) */
   :
```

5. 2. 4　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. 4　**ディスクリプタの設定**参照）が "usbf78k_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。

リスト 5－5　"usbf78k_desc.h"のストリング・ディスクリプタ設定部

```
   :
/* 0 : Language Code */
DSTR(LangString, 2, (0x09,0x04));

/* 1 : Manufacturer  */
USTR(ManString, 19, ('N','E','C',' ','E','l','e','c','t','r','o','n','i','c','s',' ','C','o','.'));

/* 2 : Product */
USTR(ProductString, 6, ('H','I','D','D','r','v'));

/* 3 : Serial Number */
USTR(SerialString,  10, ('0','_','9','8','7','6','5','4','3','2'));
   :
```

5. 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. 5　関数の仕様を参照してください。

たとえば，"usbf78k_human_interface.c" ファイルの Get_Report リクエスト処理関数（usbf78k_get_report）はリスト 5－6のように記述されています。

リスト 5－6　Get_Report リクエスト処理関数（usbf78k_get_report）

```
1   void usbf78k_get_report(void)
2   {
3         UINT8 report_type = 0;
4         UINT8 report_id = 0;
5         UINT8 keycode[REPORT_DATA_LENGTH];
6         UINT8 rqen_flg = 0;                              /* request enable flag */
7
8         /* request data check */
9         report_type = (UINT8)(( UsbSetup_Data.Value >> 8 ) & 0x00FF );
10        if( report_type == 1 ){                          /* INPUT? */
11              report_id = (UINT8)( UsbSetup_Data.Value & 0x00FF );
12              if( report_id == 0 ){                      /* ReportID:0? */
13                    if( UsbSetup_Data.Index == 0 ){
14                          if( UsbSetup_Data.Length == REPORT_DATA_LENGTH ){
15                                rqen_flg = 1;
16                          }
17                    }
18              }
19        }
20        if( rqen_flg ){
21              memset(keycode, 0, sizeof(keycode));       /* Release Key Data */
22              usbf78k_data_send(keycode, sizeof(keycode), EP0);
23        }
24        else{
25              usbf78k_sendstallEP0();
26        }
27  }
```

(1) データ送信処理

22 行目では，USB からデータを送信する関数（usbf78k_data_send）を呼び出しています。

(2) STALL 応答処理

25 行目では，Endpoint0 で STALL 応答を行うための関数（usbf78k_sendstallEP0）を呼び出しています。

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μPD78F0730 向けスタータ・キット TK-78F0730 について説明します。

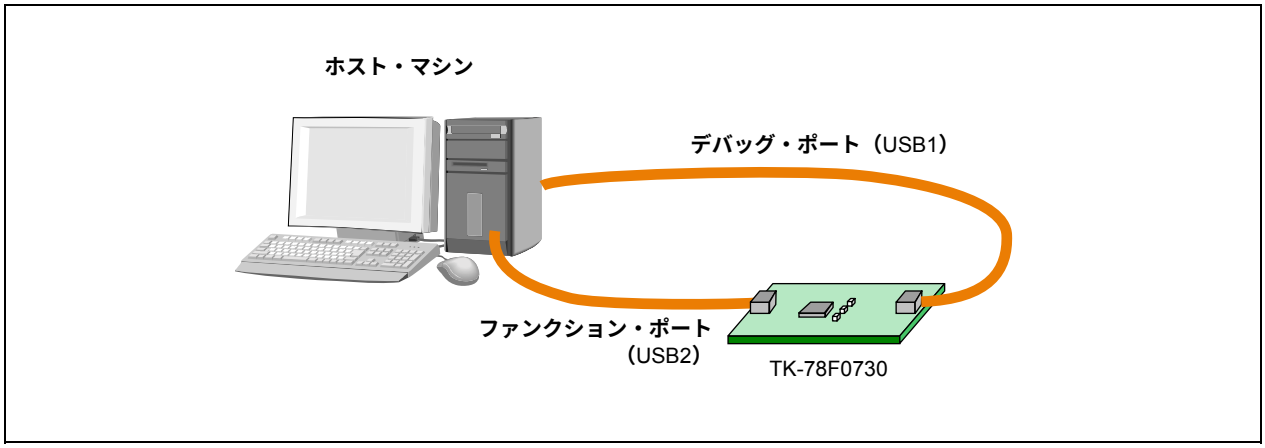
A. 1　概　　要

TK-78F0730 は，μPD78F0730 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。

TK-78F0730 には次の特徴があります。

- 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
- デバッグ用 USB miniB コネクタを装備
- コンパクトな名刺サイズ
- 統合開発環境（PM+）と組み合わせて効率的な開発を実現

図 A－1　TK-78F0730 の接続イメージ



A. 2　主な仕様

TK-78F0730 の主な仕様は次のとおりです。

○ CPU	μPD78F0730
○ 動作周波数	16 MHz
○ インタフェース	USB コネクタ（miniB）× 2 基
	MINICUBE®2 用コネクタ（SICA：パットのみ）
	拡張コネクタ 30 ピン（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 DOS/V 機
	OS：Microsoft Windows 2000，Microsoft Windows XP
○ 動作電圧	5.0 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

〒211-8668　神奈川県川崎市中原区下沼部1753

電話（代表）：(044)435−5111

【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/

【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。

お問い合わせ先

【営業関係，デバイスの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

【マイコン開発ツールの技術関係お問い合わせ先】

開発ツールサポートセンター

E-mail　：toolsupport-micom@ml.necel.com

C0901-A-01