NEC

アプリケーション・ノート

78K0R/KC3-L, 78K0R/KE3-L
（USBコントローラ内蔵製品）

16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

78K0R/KC3-L
μPD78F1022
μPD78F1023
μPD78F1024

78K0R/KE3-L
μPD78F1025
μPD78F1026

© NEC Electronics Corporation 2009

資料番号　ZUD-CB-09-0193（第1版）
発行年月　2009/12/25

〔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，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。

●本資料に記載されている内容は2009年7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。当社製品の不具合により生じた生命，身体および財産に対する損害の危険を最小限度にするために，冗長設計，延焼対策設計，誤動作防止設計等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。

は じ め に

対 象 者　このマニュアルは78K0R/KC3-L, KE3-Lの機能を理解し，その応用システムや応用プログラムを設計，開発するユーザのエンジニアを対象としています。

対象製品は，次に示す各製品です。

愛称	標準製品	USBコントローラ内蔵製品
78K0R/KC3-L	μPD78F1000, 78F1001, 78F1002, 78F1003	μPD78F1022, 78F1023, 78F1024
78K0R/KE3-L	μPD78F1007, 78F1008, 78F1009	μPD78F1025, 78F1026

目　　的　このマニュアルは，次の構成に示す機能をユーザに理解していただく事を目的としています。

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。

・78K0R/KC3-L, KE3-L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
・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。

78K0R/KC3-L, KE3-L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78K0R/KC3-L, KE3-L　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。

78K0R/KC3-L, KE3-Lの命令機能を知りたいとき
→ 別冊の78K0R 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

目次

第1章 概　　説 6

1.1 概　　要 6

1.1.1 USBファンクション・コントローラの特徴 6

1.1.2 サンプル・ドライバの特徴 6

1.2 78K0R/Kx3-Lの概要 8

1.2.1 適用製品 8

1.2.2 特　　徴 9

第2章 USBの概要 10

2.1 転送方式 10

2.2 エンドポイント 11

2.3 デバイス・クラス 11

2.4 リクエスト 12

2.4.1 種　　類 12

2.4.2 フォーマット 14

2.5 ディスクリプタ 15

2.5.1 種　　類 15

2.5.2 フォーマット 16

2.5.3 HIDクラス固有ディスクリプタのフォーマット 18

第3章 サンプル・ドライバの仕様 21

3.1 概　　要 21

3.1.1 機　　能 21

3.1.2 リクエストへの対応 21

3.1.3 ディスクリプタの設定 23

3.2 各部の動作 27

3.2.1 CPU初期化処理 29

3.2.2 USBファンクション・コントローラ初期化処理 31

3.2.3 INTUSB割り込み処理 35

3.3 関数の仕様 37

3.3.1 関数一覧 37

3.3.2 関数の相関関係 38

3.3.3 関数の機能 41

第4章 サンプル・アプリケーションの仕様 65

4.1 概　　要 65
4.2 動　　作 65
4.3 関数の利用 67

第5章 開発環境 69

5.1 開発環境 69
5.1.1 プログラム開発 69
5.1.2 デバッグ 69
5.2 環境設定 70
5.2.1 ホスト環境整備 70
5.2.2 ターゲット環境整備 78
5.3 オンチップ・デバッグ 78
5.3.1 ロード・モジュール生成 78
5.3.2 ロードと実行 79
5.4 動作確認 82

第6章 サンプル・ドライバの応用 83

6.1 概　　要 83
6.2 カスタマイズ 84
6.2.1 アプリケーション部 84
6.2.2 レジスタの設定 85
6.2.3 ディスクリプタの内容 85
6.3 関数の利用 85

第7章 スタータ・キット 86

7.1 概　　要 86
7.1.1 特　　徴 86
7.2 主な仕様 87

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78K0R/KC3-L, 78K0R/KE3-L(78K0R/Kx3-L)に内蔵の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

サンプル・ドライバの制御の対象である78K0R/Kx3-L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ー1　78K0R/Kx3-L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）	シングルバッファ構成
Endpoint0 Write	64	コントロール転送（OUT）	シングルバッファ構成
Endpoint1	64×2	バルク転送1（IN）	ダブルバッファ構成
Endpoint2	64×2	バルク転送1（OUT）	ダブルバッファ構成
Endpoint3	64×2	バルク転送2（IN）	ダブルバッファ構成
Endpoint4	64×2	バルク転送2（OUT）	ダブルバッファ構成
Endpoint7	64	インタラプト転送1（IN）	シングルバッファ構成
Endpoint8	64	インタラプト転送2（IN）	シングルバッファ構成

・ 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・パワードとセルフ・パワードを選択可能 注1
・ 内部クロックと外部クロックを選択可能 注2

内部クロック： 外部20 MHz÷内部5分周×内部12逓倍／外部16 MHz÷内部4分周×内部12逓倍／
外部12 MHz÷内部2分周×内部8逓倍（48 MHz）

注1. サンプル・ドライバではバス・パワードを選択します。
2. 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

78K0R/Kx3-L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，次のような特徴があります。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ヒューマン・インタフェース・デバイス・クラスに準拠
・ キーボード・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3.1Kバイト
　・ RAM：約0.4K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
	usbf78k0r.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
	usbf78k0r_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
	usbf78k0r.h	usbf78k0r.c関数プロトタイプ宣言
	usbf78k0r_hid.h	usbf78k0r_hid.c関数プロトタイプ宣言
	usbf78k0r_desc.h	ディスクリプタ定義
	errno.h	エラー・コード定義
	types.h	ユーザ型宣言

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 5.2.1 ホスト環境整備 を参照してください。

1. 2　78K0R/Kx3-Lの概要

ここでは，サンプル・ドライバの制御の対象である78K0R/KC3-L, KE3-Lについて説明します。

78K0R/KC3-L, KE3-Lは，NECエレクトロニクスのシングルチップ・マイクロコントローラ78K0Rマイコンのロウ・パワー・シリーズの製品群です。78K0R CPUコアを使用し，ROM/RAM，タイマ／カウンタ，，POC/LVI，シリアル・インタフェース，A/D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78K0R/KC3-L, KE3-L　**USBコントローラ内蔵製品 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。

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　78K0R/Kx3-L 製品一覧

愛　　称	品　　名	内蔵メモリ		内蔵USB機能	割り込み	
		フラッシュ・メモリ	RAM		内部	外部
78K0R/KC3-L (48pin)	μPD78F1022	64 KB	6 KB	ファンクション・コントローラ	36	7
	μPD78F1023	96KB	8 KB	ファンクション・コントローラ	36	7
	μPD78F1024	128KB	8 KB	ファンクション・コントローラ	36	7
78K0R/KE3-L (64pin)	μPD78F1025	96KB	8 KB	ファンクション・コントローラ	41	11
	μPD78F1026	128KB	8 KB	ファンクション・コントローラ	41	11

注意　このアプリケーション・ノートでは，特に違いのないかぎり，対象のマイクロコントローラを総括して78K0R/Kx3-Lの呼称で記載します。

1. 2. 2 特　　徴

78K0R/Kx3-Lには，主に次のような特徴があります。詳細は、78K0R/Kx3-Lユーザーズ・マニュアルを参照下さい。

メモリ空間
・1Mバイト・リニア・アドレス空間（プログラム／データ共用）

内蔵メモリ

・ RAM：6K/ 8K バイト

・ フラッシュ・メモリ：64K/ 96K/ 128K バイト

乗除算機

・16ビット×16ビット = 32ビット（乗算）

・32ビット÷32ビット = 32ビット（除算）

キー割り込み
・4チャネル
・8チャネル

DMAコントローラ
・2チャネル

シリアル・インタフェース
・CSI：1チャネル／UART：1チャネル
・CSI：1チャネル／UART：1チャネル／簡易I2C：1チャネル
・CSI：1チャネル注／UART：1チャネル注／簡易I2C：1チャネル注
・UART（LIN-bus対応）：1チャネル
・I2C：1チャネル

USBコントローラ
・USBファンクション（フル・スピード）：1チャネル

A/Dコンバータ
・10ビット分解能A/Dコンバータ（AVREF = 1.8～3.6 V）：8チャネル

電源電圧
・VDD = 1.8～3.6 V（USB未使用時）
・VDD = 3.0～3.6 V（USB使用時）

クロック出力／ブザー出力

・2.44 kHz, 4.88 kHz, 9.76 kHz, 1.25 MHz, 2.5 MHz, 5 MHz, 10 MHz（周辺ハードウエア・

クロック：f_{MAIN} = 20 MHz動作時）

・256 Hz, 512 Hz, 1.024 kHz, 2.048 kHz, 4.096 kHz, 8.192 kHz, 16.384 kHz, 32.768 kHz

（サブシステム・クロック：f_{SUB} = 32.768 kHz動作時）

オンチップ・デバッグ機能内蔵

注：78K0R/KE3-Lのみ

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はあります。

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　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。

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ー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 エンドポイント	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 Halt状態の読み取り
CLEAR_FEATURE	デバイス エンドポイント	リモート・ウエイクアップのクリア Haltの解除（DATA PID = 0）
SET_FEATURE	デバイス エンドポイント	リモート・ウエイクアップまたはテスト・モードの設定 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	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	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

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。

- 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。

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。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 2−5　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，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 2−6　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ストリング・ディスクリプタへのインデックス

表 2ー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ー8　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・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	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）。	

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，78K0R/Kx3-L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，78K0R/Kx3-Lの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) エンドポイントの処理

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。

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

78K0R/Kx3-L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

ホストがファンクション・デバイスのストリング・ディスクリプタやクラス固有の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応答を返します。

(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HID・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 "usbf78k0r_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 3−1 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	0x02	クラス・コード：HID
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	0x01CD	プロダクトID：78K0R /Kx3-L
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 3−2　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，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	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フェースを2つ使用するため，ディスクリプタも2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・ドライバではエンドポイントを3つ使用するため，ディスクリプタも3種類設定しています。

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	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　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（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	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 3−6　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。

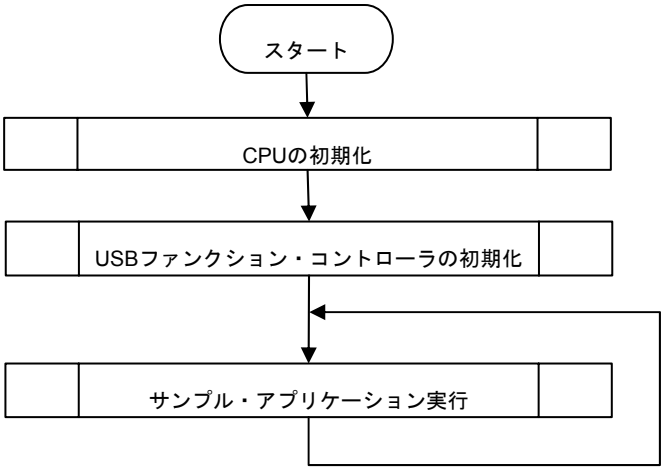
表 3−7　レポート・ディスクリプタの設定

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。

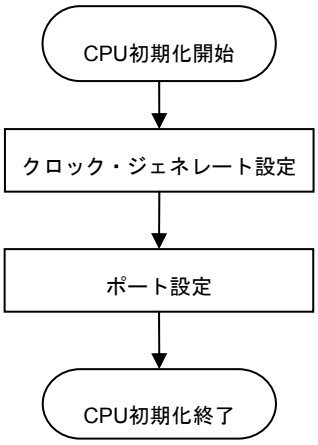
図 3−1　サンプル・ドライバの処理フロー



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) 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、5つのレジスタにアクセスします。

(a) CMC レジスタに “0x41” を書き込みます。この設定により、X1 発振モード、10MHz < f_{MX} <= 20MHz が設定されます。

(b) CSC レジスタの MSTOP ビットに “0” を書き込みます。この設定により、X1 発振回路の動作を開始させます。

(c) OSTC レジスタにより発振安定時間を確認します。

(d) PLLC レジスタに “0x01” を書き込みます。この設定により、PLL の動作を停止させます。

(e) CKC レジスタに “0x38” を書き込みます。この設定により、CPU/周辺ハードウエア･クロックをメイン･システム･クロック（f_{MAIN}）に、メイン･システム･クロックを高速システム・クロック（f_{MX}）に、分周比を f_{MX} に設定します。

(f) CSC レジスタの HIOSTOP ビットに “1” を書き込みます。この設定により、高速内蔵発振回路を停止させます。

(g) PLLC レジスタの PLLM ビットに “1” を書き込みます。この設定により、PLL への供給クロックの逓倍率を 12 逓倍に設定します。

(h) 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 の動作を開始させます。

(2) ポート設定

(a) スイッチに接続しているポートのプルアップオプションの設定を行います。P17、P43、P42、P70、P71、P72、P73、P74 に”1 を”書き込みます。

(b) 常時シュミット入力タイプの端子がオープンになると貫通電流が流れる為、未使用端子にはリセット解除後に処置が必要になります。サンプルドライバ（64pin 版の設定）では、P05、P06、P30、P44、P46、P47、P54、P55、P65、P66、P67 に対して ”0” を書き込みます。

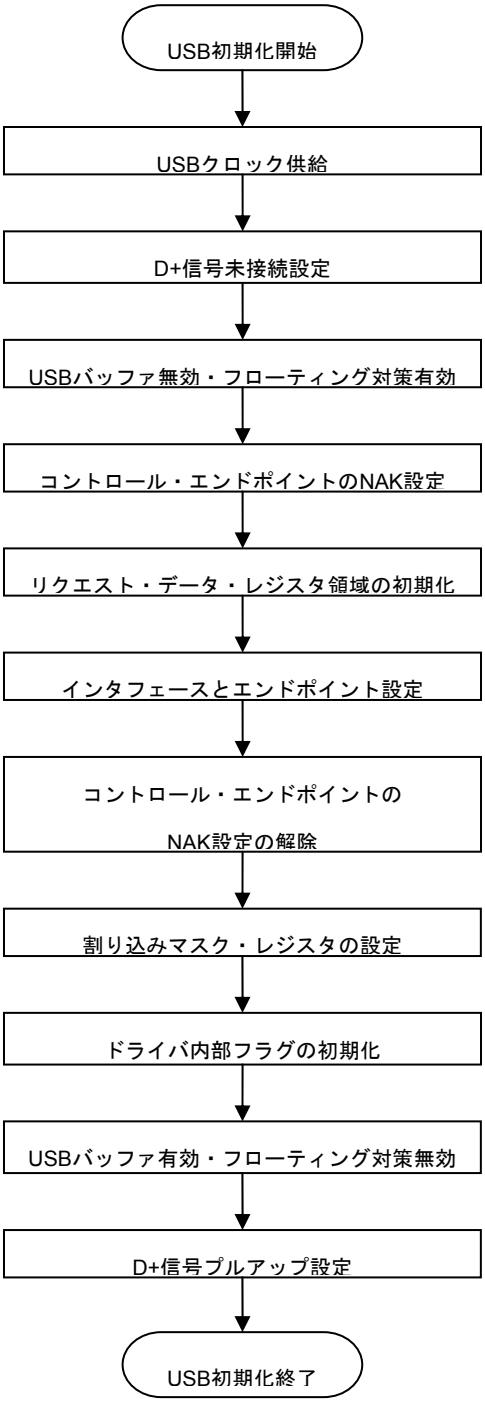
注意：48pin の場合、上記端子に加え、P42、P43、P53、P74、P75、P77、P110、P142、P143 にも処置が必要になります。

(c) LED の設定。PM2、PM5 、PM110、PM110 に ”0” を書き込みます。LED 消灯の為に、P2、P5 に ”0xff” を書き込みます。P110 に ”0” を書き込んだ後、”1 を”書き込み、LED(7SegA)を消灯させます。次に、再び、P2、P5 に ”0xff” を書き込み、P111 に ”0” を書き込んだ後、”1 を”書き込み、LED(7SegB)を消灯させます。

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) USB クロック供給

UCKCレジスタに ”0x80” を設定し、USBファンクション・コントローラへUSBクロックを供給します。

(2) D+信号未接続設定

UF0GPRレジスタに ”0x02” を設定し、ホスト側にデバイスが接続されていることを検知されないようにします。

(3) USB バッファ無効・フローティング対策有効

UF0BCレジスタに “0x00” を設定し、USBバッファ無効、フローティング対策有効に設定しUSBファンクション･コントローラを動作不可にします。

(4) コントロール・エンドポイントの NAK 設定

すべてのリクエストに対するNAK応答の動作を切り替えます。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エアが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するために，ソフトウエアが使用します。

(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。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、リモート・ウエイクアップ機能を使用する設定となります。
(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 NAK 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にリンクされます。

(7) コントロール・エンドポイントの NAK 設定の解除

すべてのリクエストに対するNAK応答の動作を切り替えます。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) UF0IM0 レジスタに “0x3B” 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、RSUSPDM、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 “0xF3” 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 “0xFF” 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。

(h) CPU の KRM レジスタに“0x03” を書き込みます。この設定により、キー割り込み信号を検出します。

(i) CPU の USBIF ビットに “0” を書き込みます。この設定により、INTUSB 割り込みがクリアされます。

(j) CPU の RSUMIF ビットに “0” を書き込みます。この設定により、INTRSUM 割り込みがクリアされます。

(k) CPU の KRIF ビットに “0” を書き込みます。この設定により、INTKR 割り込みがクリアされます。

(l) CPU の USBMK ビットに “0” を書き込みます。この設定により、INTUSB 割り込みがマスク解除されます。

(m) CPU の RSUMMK ビットに “0” を書き込みます。この設定により、INTRSUM 割り込みがマスク解除されます。

(n) CPU の KRMK ビットに “0” を書き込みます。この設定により、INTKR 割り込みがマスク解除されます。

(9) ドライバ内部フラグの初期化

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，次に示すレジスタにアクセスします。

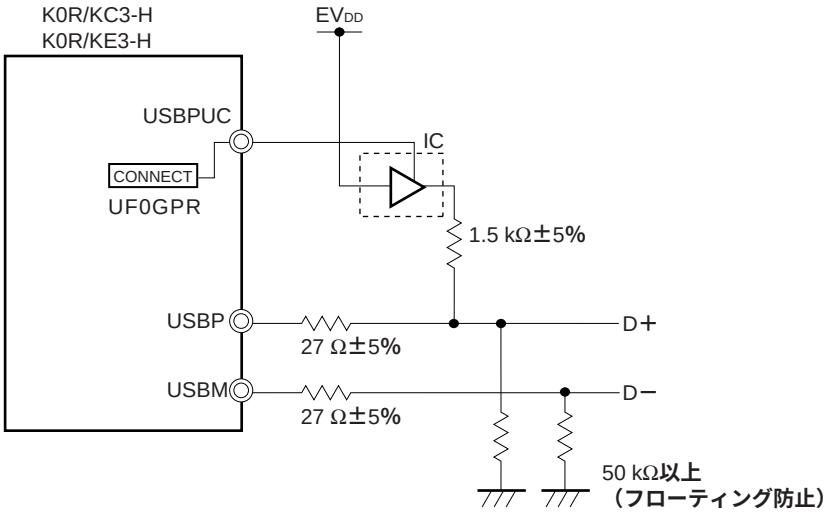
(10) USB バッファ有効・フローティング対策無効

UF0BCレジスタに “0x03” を設定し、USBバッファ有効、フローティング対策無効に設定しUSBファンクション･コントローラを動作可能にします。

(11) D+信号プルアップ設定

UF0GPRレジスタに ”0x02” を設定し、ホスト側にデバイスが接続されたことを通知します。

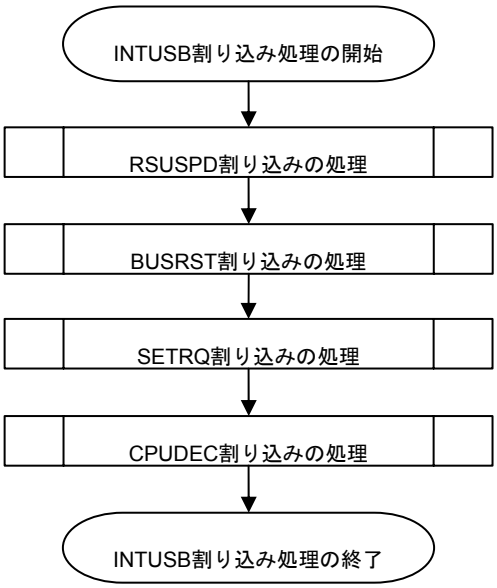
図 3－4　USBファンクション・コントローラ接続例



3. 2. 3 INTUSB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（INTUSB）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、初期化時にマスク解除してください。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 INTUSB割り込み処理フロー



(1) RSUSPD 割り込みの処理

Suspend/Resume 割り込みが発生した場合、通知されます。

以下の手順で処理を行います。

(a) この割り込みが Suspend 割り込みによって発生（UF0EPS1 が ”0x00” である事を確認）し、usbf78k0r_usbstate_flg フラグの値が “0x04” （Configured 状態）であれば usbf78k0r_usbstate_flg フラグに ”0x06”(Suspend 状態) を書き込みます。

(b) usbf78k0r_stop_mode()関数を呼び出します。

(2) BUSRST 割り込みの処理

Bus Resetが発生した場合、通知されます。

以下の手順で処理を行います。

(a) UF0IC0 レジスタに “0x7F” を書き込みます。この設定により、BUSRST 割り込みがクリアされます。

(b) usbf78k0r_busrst_flg フラグに “1” を書き込みます。

(c) usbf78k0r_buff_init() 関数を呼び出します。

(3) SETRQ 割り込みの処理

自動処理対象のSET_XXXXリクエストを受信し、自動処理を行った場合、通知されます。
以下の手順で処理を行います。

(a) UF0IC0 レジスタに “0xFB” を書き込みます。この設定により、SETRQ 割り込みがクリアされます。

(b) UF0SET レジスタの SETCON ビットと UF0MODS レジスタの CONF ビットが共にセット (”1”) されている事を確認します。共にセットされている場合、SET_CONFIGURATION リクエストにより、CONFIGURATION が “1” に設定されていることを示します。

(c) usbf78k0r_busrst_flg フラグに “0” を書き込みます。この設定により、リセット状態から通常状態に移行したことを通知します。

(4) CPUDEC 割り込みの処理

(a) UF0IC1 レジスタに “0xFD” を書き込みます。この設定により、PROT 割り込みがクリアされます。

(b) UF0E0ST 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。

(c) リクエストがクラス・リクエストなら、usbf78k0r_classreq() 関数を呼び出し、クラス・リクエスト処理を行います。

(d) リクエストがクラス・リクエストでなければ、usbf78k0r_standardreq() 関数を呼び出し、標準リクエスト処理を行います。

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 3－8　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	cpu_init	CPUの初期化
	main	メイン・ルーチン
usbf78k0r.c	usbf78k0r_init	USBファンクション・コントローラの初期化
	usbf78k0r_intusbf0	INTUSB割り込み処理
	usbf78k0r_intkr	INTKR割り込み処理
	usbf78k0r_intrsum	INTRSUM処理
	usbf78k0r_stop_mode	STOPモード処理
	usbf78k0r_standardreq	標準リクエスト処理
	usbf78k0r_getdesc	GET_DESCRIPTOR（String、HID、Report）処理
	usbf78k0r_send_EP0	Endpoint0の送信処理
	usbf78k0r_receive_EP0	Endpoint0の受信処理
	usbf78k0r_sendnullEP0	Endpoint0のNULLパケット送信処理
	usbf78k0r_sendstallEP0	Endpoint0のSTALL応答処理
	usbf78k0r_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
	usbf78k0r_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
	usbf78k0r_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
usbf78k0r_hid.c	usbf78k0r_classreq	HIDクラス・リクエスト処理関数
	usbf78k0r_get_report	Get Reportリクエストの処理
	usbf78k0r_get_idle	Get Idleリクエストの処理
	usbf78k0r_get_protocol	Get Protocolリクエストの処理
	usbf78k0r_set _report	Set Reportリクエストの処理
	usbf78k0r_set_idle	Set Idleリクエストの処理
	usbf78k0r_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

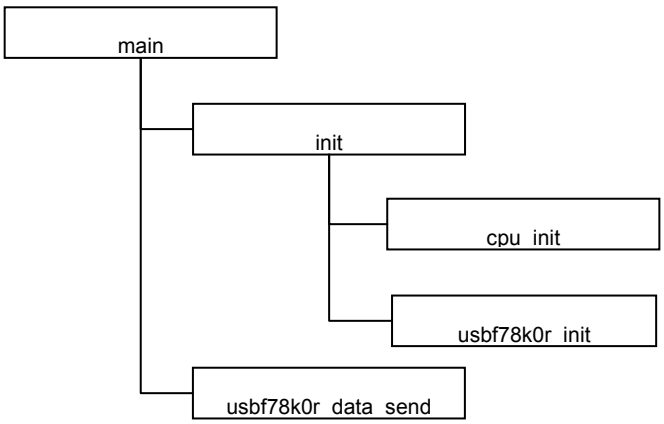


図 3―7　USBファンクション・コントローラ用処理での関数の呼び出し

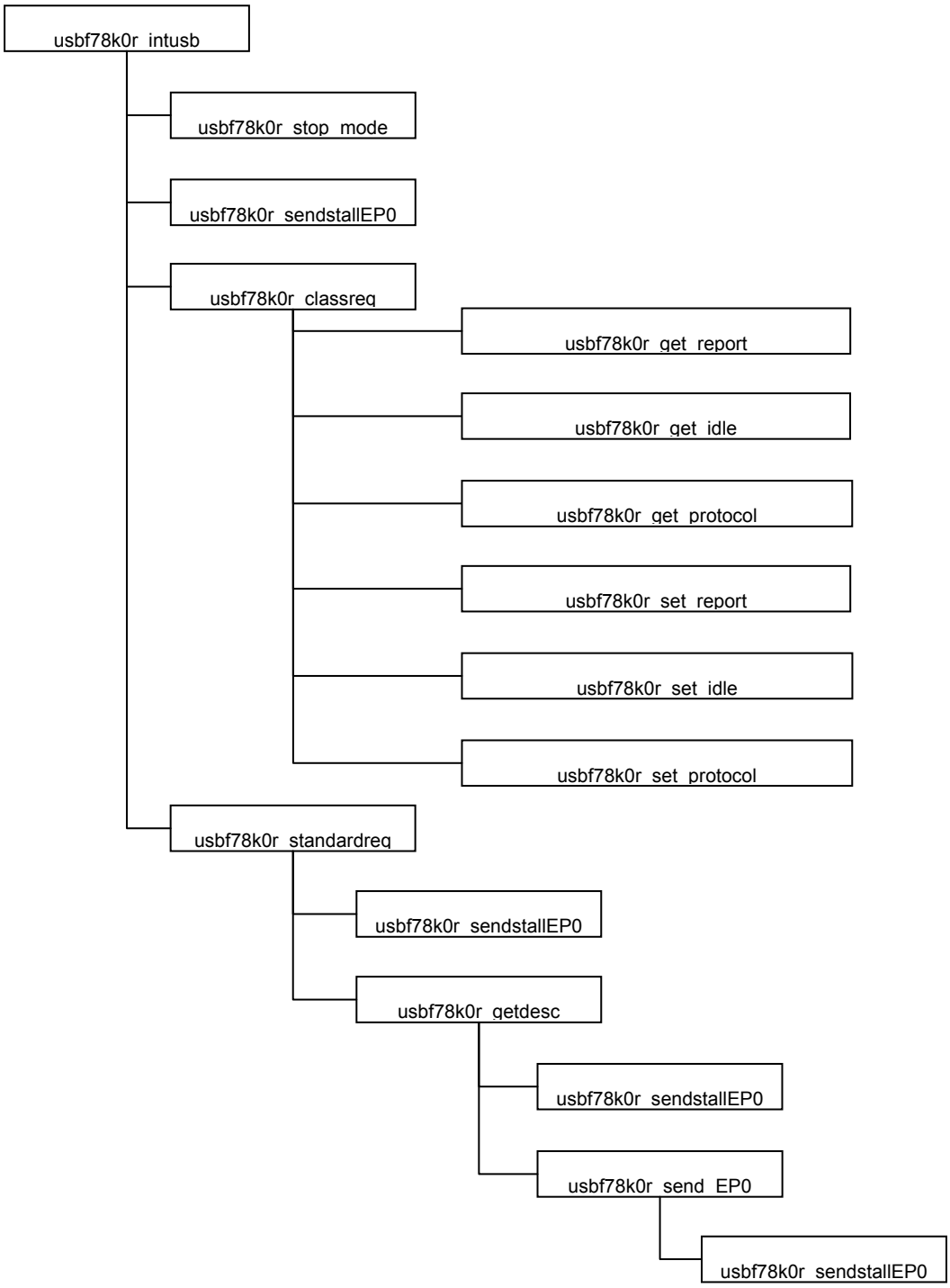
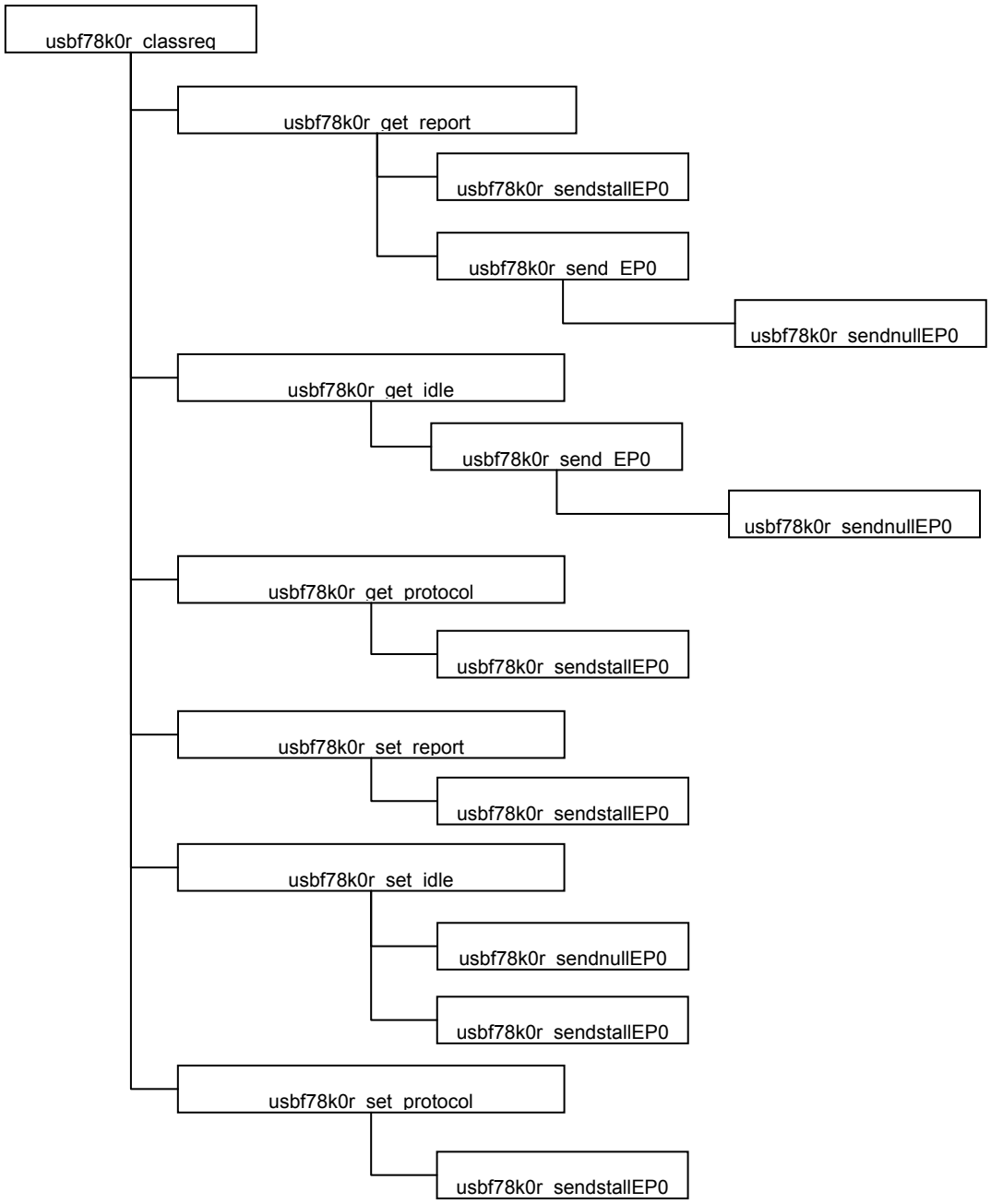


図 3－8　USB HID・クラス用処理での関数の呼び出し(1)



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

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】
メイン処理で呼び出される関数です。
クロック周波数や動作モードなど、78K0R/Kx3で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。

USBファンクション・コントローラ用処理の関数

usbf78k0r_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 usbf78k0r_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。
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。

usbf78k0r_intusb

【概要】

INTUSB割り込み処理

【C言語記述形式】

__interrupt void usbf78k0r_intusb (void)

【パラメータ】

なし

【戻り値】

なし

【機能】

INTUSB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、割り込み要求を確認しながら、発生している割り込みについて処理を行います。

usbf78k0r_intkr

【概要】

INTKR割り込み処理

【C言語記述形式】

__interrupt void usbf78k0r_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。

usbf78k0r_rsum

【概要】

INTRSUM割り込み処理

【C言語記述形式】

__interrupt void usbf78k0r_rsum(void)

【パラメータ】

なし

【戻り値】

なし

【機能】

INTRSUM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。

レジューム信号が発生された事を示すフラグを更新します。

usbf78k0r_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 usbf78k0r_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。
USBクロック等を停止して、STOPモードに入ります。INTRSUM割り込み、INTKR割り込みにより解除されます。

usbf78k0r_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 usbf78k0r_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（usbf78k0r_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（usbf78k0r_sendstallEP0）を呼び出します。

usbf78k0r_getdesc

【概要】

GET_DESCRIPTORリクエスト処理

【C言語記述形式】

void usbf78k0r_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ストリング・ディスクリプタを要求している場合、Endpoin0用USBデータ送信関数（usbf78k0r_send_EP0）を呼び出して、Endpoint0からストリング・ディスクリプタを送信させます。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（usbf78k0r_sendstallEP0）を呼び出します。

usbf78k0r_send_EP0

【概要】

Endpoint0用USBデータ送信処理

【C言語記述形式】

INT32 usbf78k0r_send_EP0(UINT8* data, INT32 len)

【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

【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。

usbf78k0r_receive_EP0

【概要】
Endpoint0用USBデータ受信処理

【C言語記述形式】
INT32 usbf78k0r_receive_EP0(UINT8* data, INT32 len)

【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。

usbf78k0r_sendnullEP0

【概要】
Endpoint0用NULLパケット送信処理

【C言語記述形式】
void usbf78k0r_sendnullEP0(void)

【パラメータ】
なし

【戻り値】
なし

【機能】
Endpoint0用の送信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。

usbf78k0r_sendstallEP0

【概要】

endpoint0用STALL応答処理

【C言語記述形式】

void usbf78k0r_sendstallEP0(void)

【パラメータ】

なし

【戻り値】

なし

【機能】

Endpoint0用の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。

usbf78k0r_ep_status

【概要】

バルク / インタラプト・イン・エンドポイント用FIFO状態通知処理

【C言語記述形式】

INT32 usbf78k0r_ep_status(INT8 ep)

【パラメータ】

パラメータ	説明
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（FIFO empty）
DEV_ERROR	異常終了（FIFO full）
DEV_RESET	Bus Reset処理中

【機能】

指定されたエンドポイント（送信用）のFIFO状態を通知します。

usbf78k0r_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 usbf78k0r_send_null(INT8 ep)

【パラメータ】

パラメータ	説明
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】

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。

usbf78k0r_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 usbf78k0r_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	データ送信エンドポイント番号

【戻り値】

シンボル	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。

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

usbf78k0r_classreq

【概要】

クラス・リクエストの処理

【C言語記述形式】

void usbf78k0r_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（usbf78k0r_sendstallEP0）を呼び出します。

usbf78k0r_get_report

【概要】

Get Reportリクエスト処理

【C言語記述形式】

void usbf78k0r_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。

usbf78k0r_get_idle

【概要】
Get Idleリクエスト処理

【C言語記述形式】

```
void usbf78k0r_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。

usbf78k0r_get_protocol

【概要】
Get Protocolリクエスト処理

【C言語記述形式】
void usbf78k0r_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。

usbf78k0r_set_report

【概要】

Set Reportリクエスト処理

【C言語記述形式】

void usbf78k0r_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。

usbf78k0r_set_idle

【概要】

Set Idleリクエスト処理

【C言語記述形式】

void usbf78k0r_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。

usbf78k0r_set_protocol

【概要】

Set Protocolリクエスト処理

【C言語記述形式】

void usbf78k0r_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

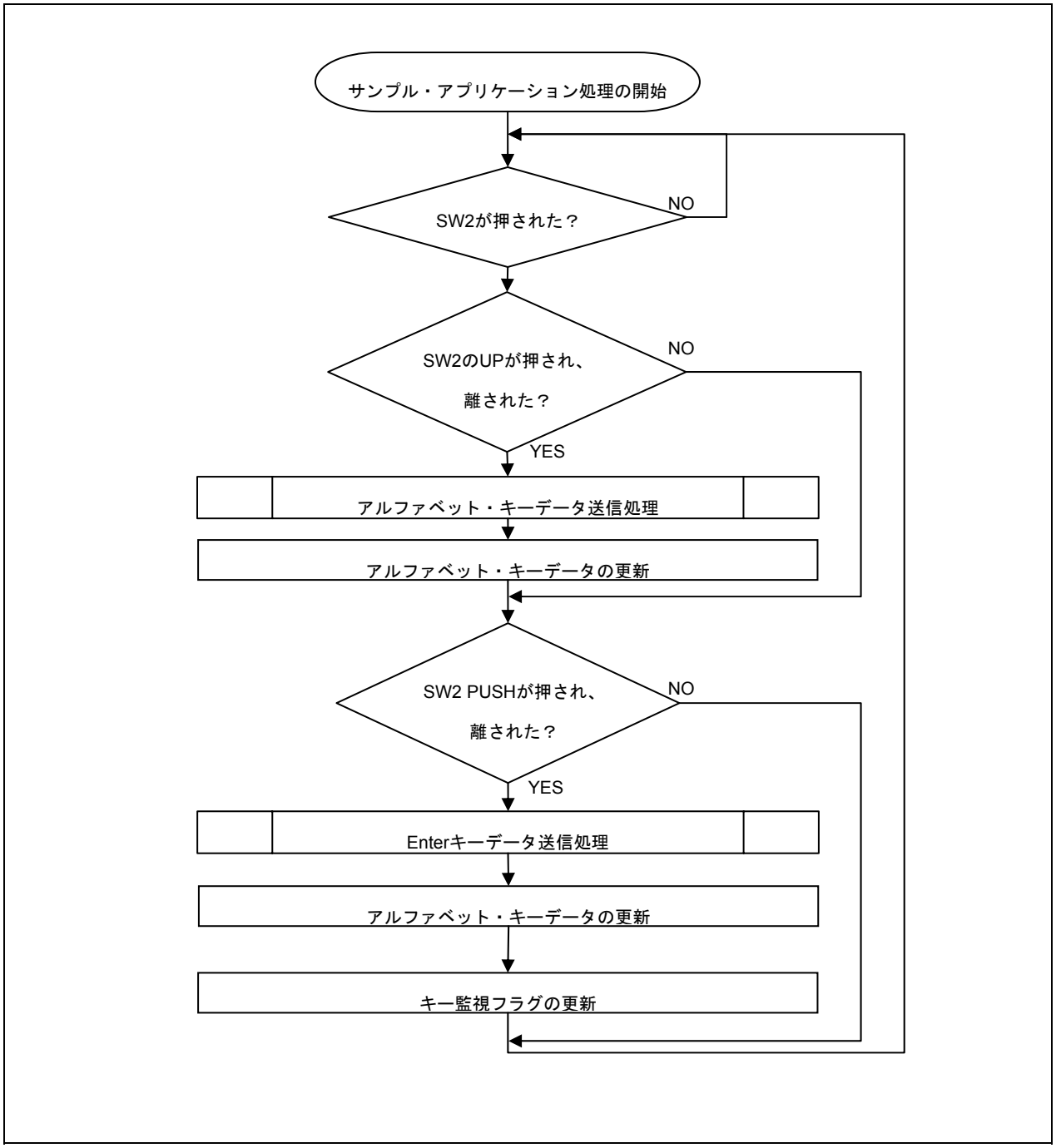
サンプル・アプリケーションは、USBヒューマン・インタフェース・デバイス・クラス用ドライバの利用の簡単な例として用意されたもので、サンプル・ドライバのメイン・ルーチンに組み込まれています。

このサンプル・アプリケーションは、TK-78K0R/KE3L+USB ボード上のスイッチSW2（PUSH）、が押された時にアルファベット「a」～「z」を送信、SW2（UP）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) SW2 が押されたかを確認

TK-78K0R/KE3L+USB ボード上のスイッチSW2のUP、またはSW2のPUSHが押されたかを、キー押下確認用フラグ（usbf78k0r_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) SW2 の UP が押されたのかの確認

P70が “0” であれば、SW2のUP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

usbf78k0r_ep_status関数で送信用エンドポイントのFIFOが空である事を確認し、アルファベット・キーデータとリリース・キーデータをusbf78k0r_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) SW2 の PUSH が押されたのかの確認

P71が“0”であれば、SW2のPUSH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

usbf78k0r_ep_status関数で送信用エンドポイントのFIFOが空である事を確認し、Enterキーデータとリリース・キーデータをusbf78k0r_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（usbf78k0r_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。

(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、”usbf78k0r.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（usbf78k0r_init）を呼び出します。

(4) FIFO 状態確認

FIFO状態確認用関数（usbf78k0r_ep_status）でFIFOの状態を確認します。

(5) 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（usbf78k0r_data_send）でデータを送信します。

リスト 4−1　サンプル・アプリケーションの記述（部分）

```
void main( void )
{
UINT8 keycode[REPORT_DATA_LENGTH];
UINT8 keydata = A_KEY;

init();

    memset(keycode, 0, sizeof(keycode));          /* Key Data Clear */
    usbf78k0r_keytouch_flg = F_SW_OFF;          /* Key Flag Clear */

    while(1)
    {
        if ( usbf78k0r_keytouch_flg == F_SW_ON ) {
            if((P7 & 0x01) == 0x00) {/* SW2 UP */
                while((P7 & 0x01) == 0x00){
                /* SW2 OFF WAIT */
                }
                keycode[KEY1_SCAN_CODE] = keydata;  /* Press Key Data */
                while (usbf78k0r_ep_status(C_INT1) != DEV_OK) {}
                usbf78k0r_data_send(keycode, sizeof(keycode), C_INT1);
                memset(keycode, 0, sizeof(keycode));       /* Release Key Data */
                while (usbf78k0r_ep_status(C_INT1) != DEV_OK) {}
                usbf78k0r_data_send(keycode, sizeof(keycode), C_INT1);
                keydata++;                                 /* a to z */
                if(keydata == EXCLAMATION_KEY) {
                    keydata = A_KEY;
                }
            }
            if((P7 & 0x02) == 0x00) {/* SW2 PUSH */
                while((P7 & 0x02) == 0x00){
                    /* SW2 OFF WAIT */
                }
                keycode[KEY1_SCAN_CODE] = ENTER_KEY;/* Press Key Data(Enter) */
                while (usbf78k0r_ep_status(C_INT1) != DEV_OK) {}
                usbf78k0r_data_send(keycode, sizeof(keycode), C_INT1);
                memset(keycode, 0, sizeof(keycode));       /* Release Key Data */
                while (usbf78k0r_ep_status(C_INT1) != DEV_OK) {}
                usbf78k0r_data_send(keycode, sizeof(keycode), C_INT1);
                keydata = 0x04;
            }
            usbf78k0r_keytouch_flg = F_SW_OFF;
        }
    }
}
```

第5章 開発環境

この章では，78K0R/Kx3-L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。

5. 1 開発環境

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。

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には，次のようなハードウエアとソフトウエアが必要です。

表 5－1 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)
ソフトウエア	統合開発ツール	PM+	V6.31
	コンパイラ	CC78K0R	W2.12
	アセンブラ	RA78K0R	W1.33

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には，次のようなハードウエアとソフトウエアが必要です。

表 5－2 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)
	ターゲット	TK-78K0R/KE3L+USB	
	インサーキット・エミュレータ	MINICUBE2	
	USBケーブル	－	miniBコネクタ・Aコネクタ
ソフトウエア	統合開発ツール	PM+	V6.31
	デバッガ	ID78K0R-QB	V3.60
ファイル	デバイス・ファイル	DF78102664.78K	78K0R/Kx3-L用
	プロジェクト関連ファイル	－	注1

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。

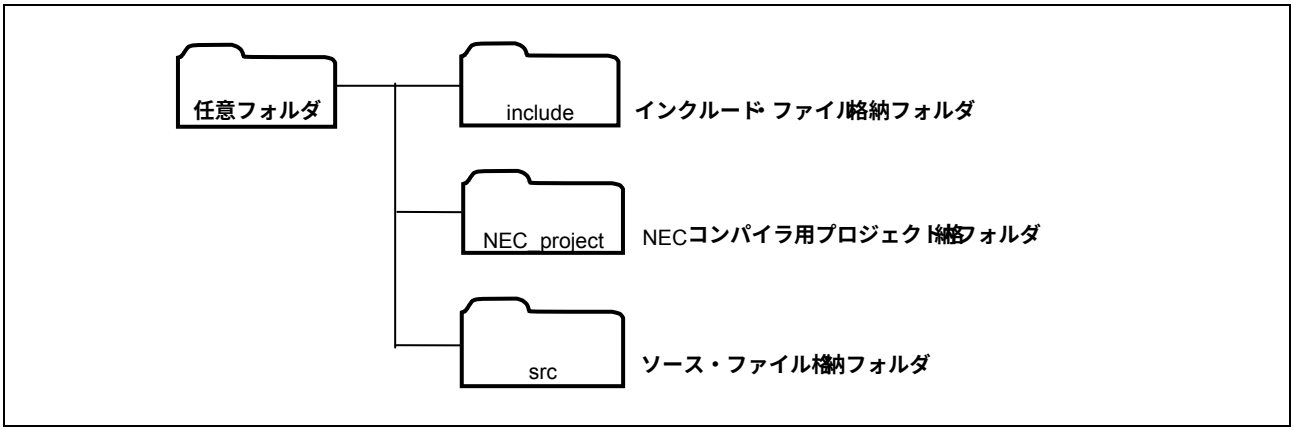
統合開発ツールのインストール

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。

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，デバイス・ドライバを任意のディレクトリに格納します。

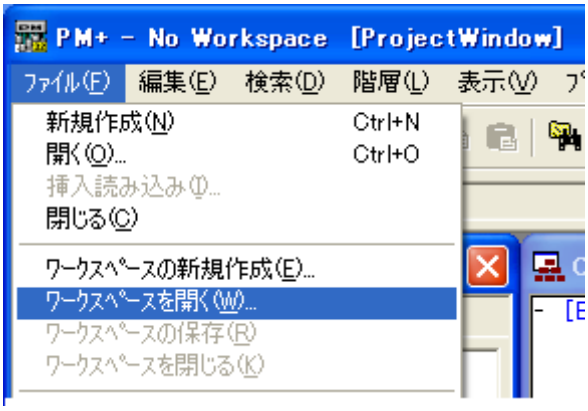
図 5−1　サンプル・ドライバのフォルダ構成



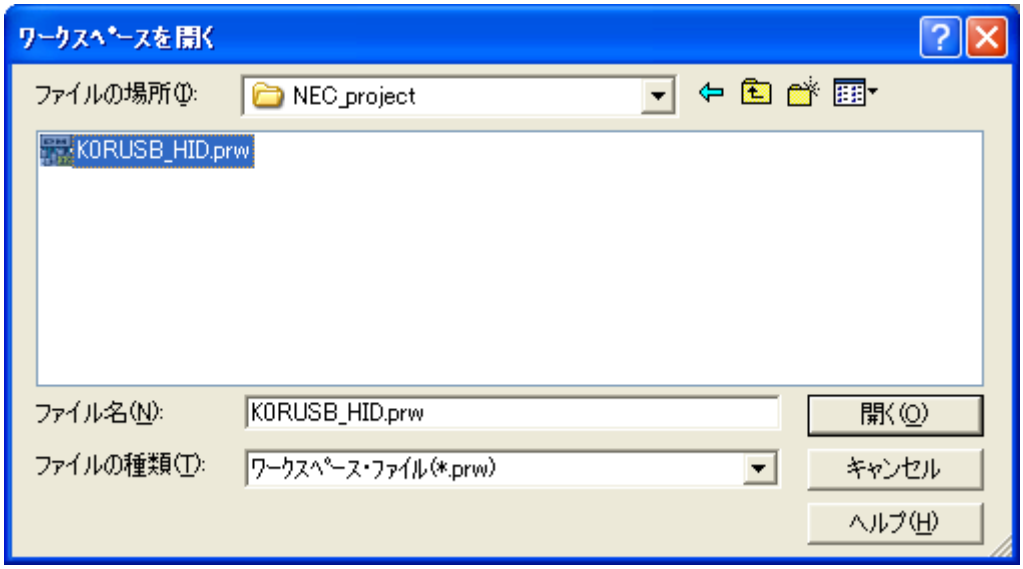
ワークスペースの設定

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。



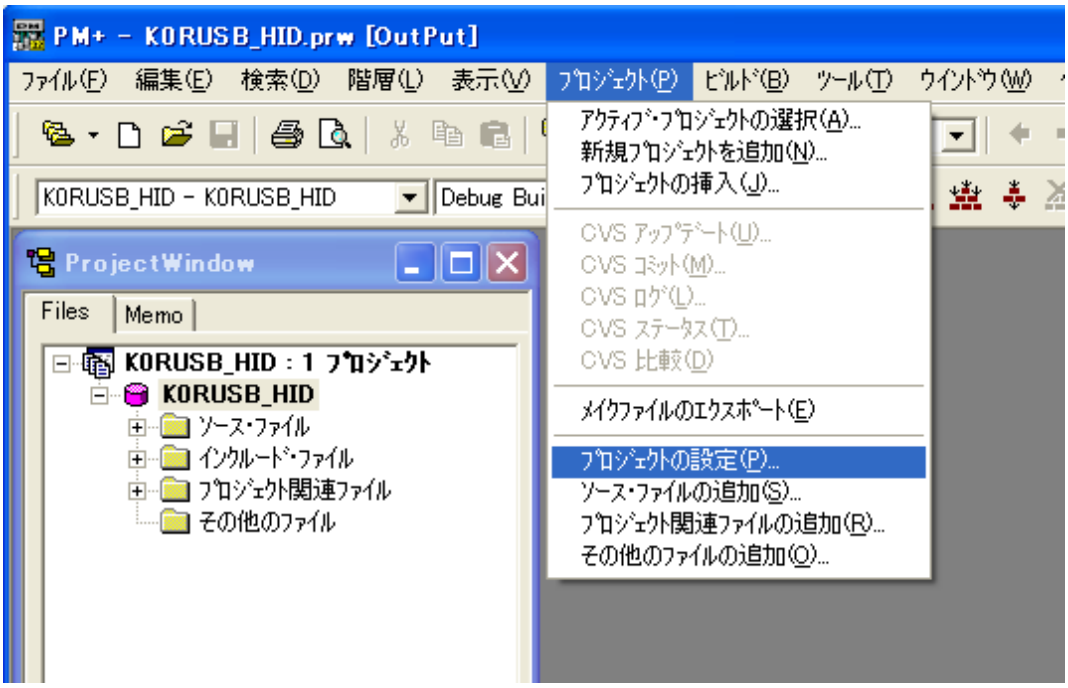
<2> 「ワークスペースを開く」ダイアログが開きます。サンプル・ドライバを格納したディレクトリの「NEC_project」フォルダにあるワークスペース・ファイルを指定します。



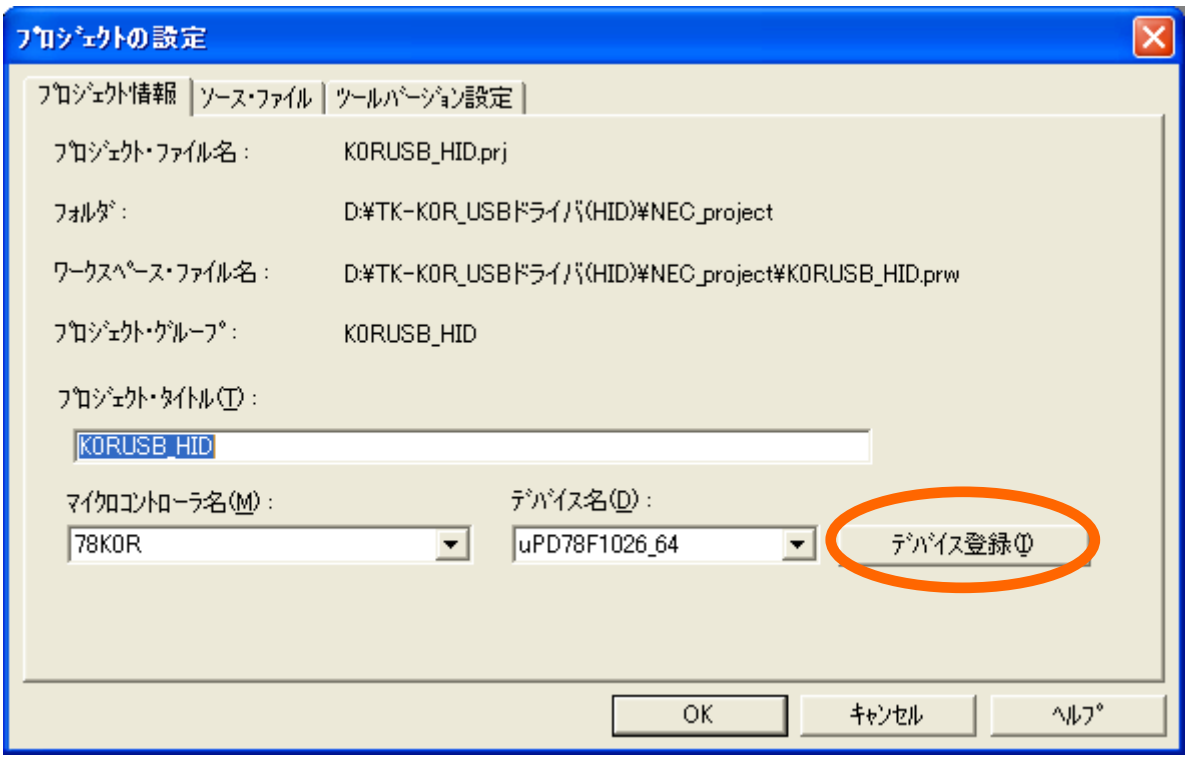
デバイス・ファイルのインストール

ここでは78K0R/Kx3-L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。

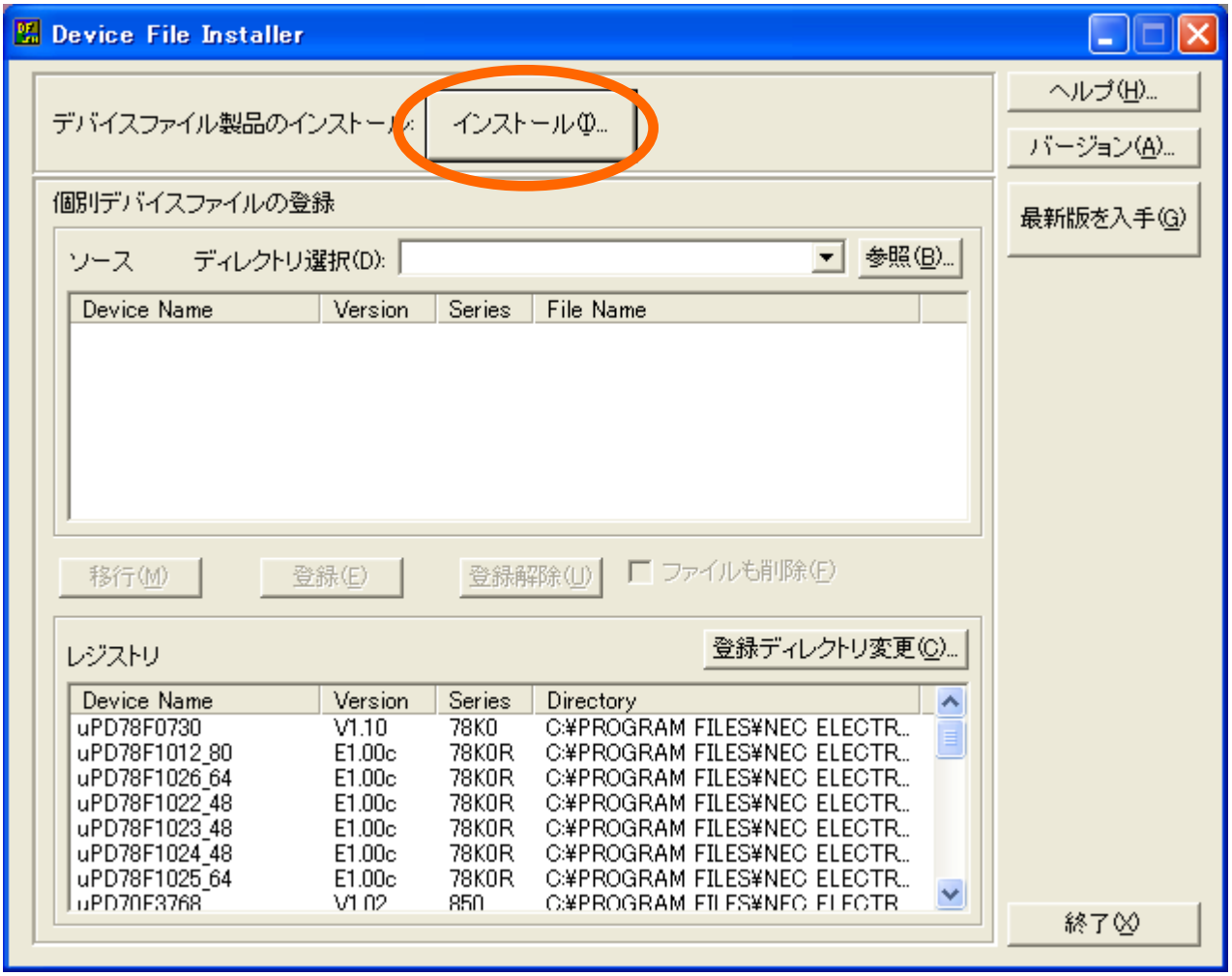
<1> PM+の「プロジェクト」メニューから「プロジェクトの設定」を選択します。



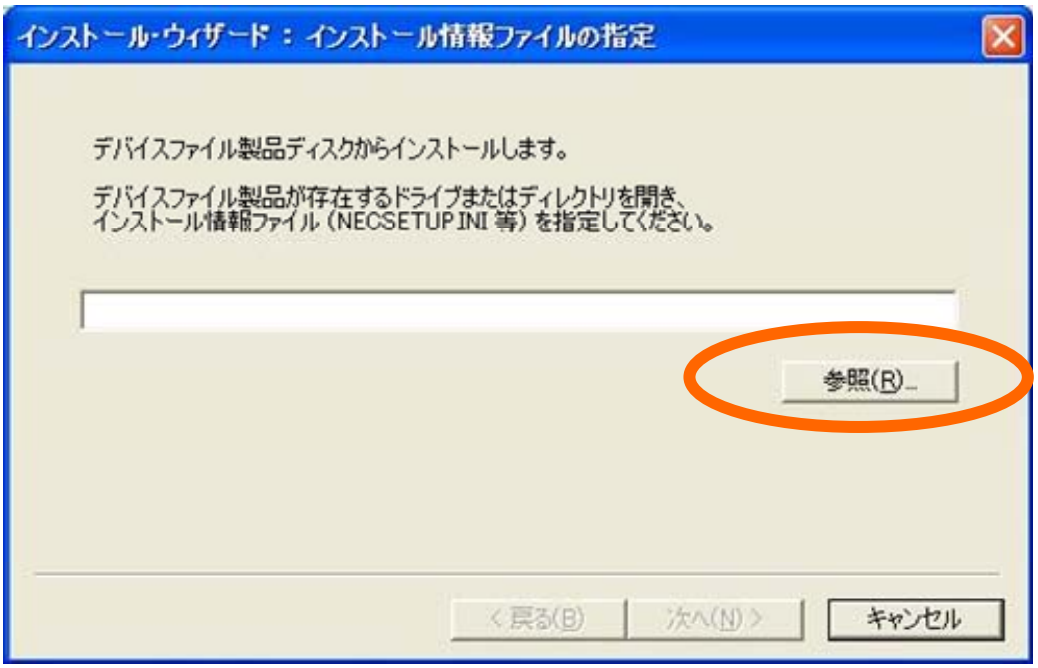
<2> 「プロジェクトの設定」ダイアログが開きます。「プロジェクト情報」タブの「デバイス登録」ボタンを押下してDevice File Installerを起動します。



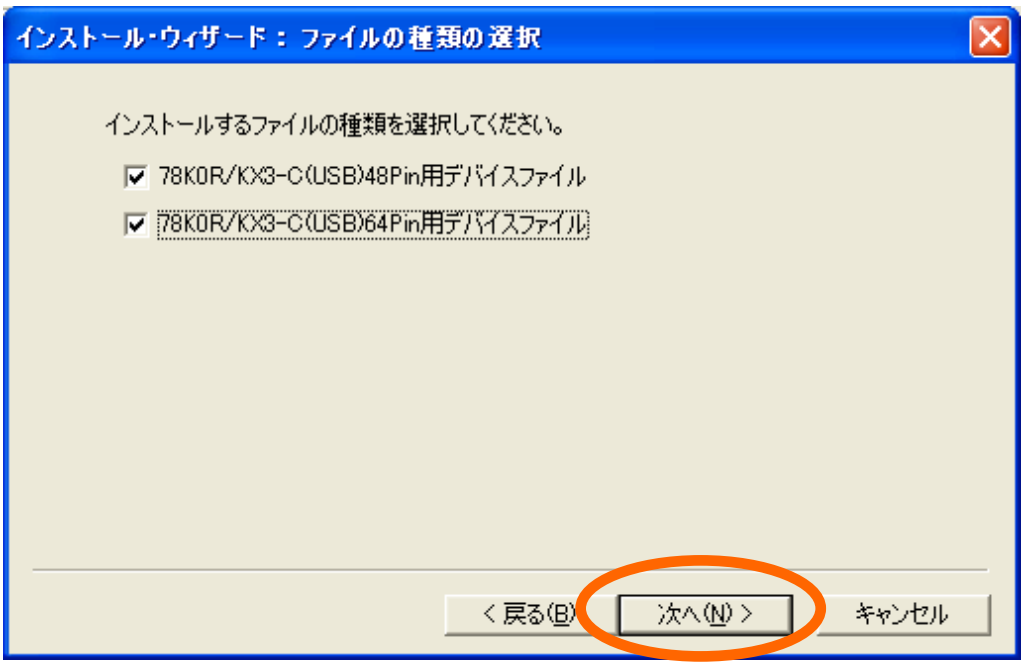
<3> 「Device File Installer」ダイアログが開きます。「インストール」ボタンを押下してインストール・ウィザードを起動します。



<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。

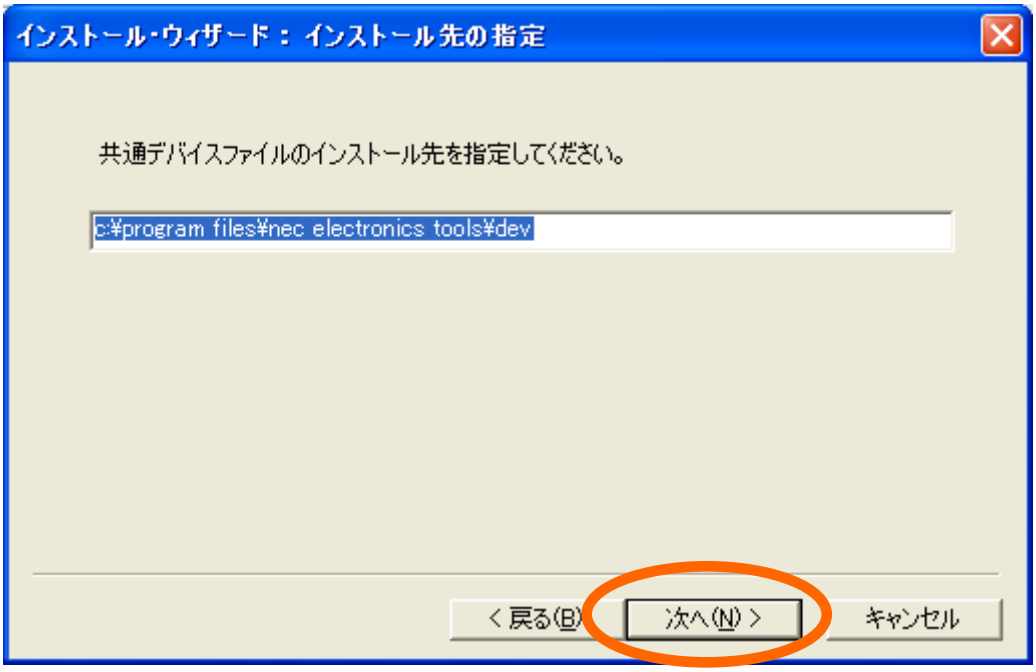


<5> 「ファイルを開く」ダイアログが表示されます。デバイス・ファイルを格納したディレクトリを開き，"NECSETUP.INI" ファイルを選択して「開く」ボタンを押下します。
<6> 使用許諾に関するメッセージが表示されます。「次へ」ボタンを押下します。
<7> 「ファイルの種類の選択」ダイアログが開きます。該当するデバイス・ファイルを選択して「次へ」ボタンを押下します。

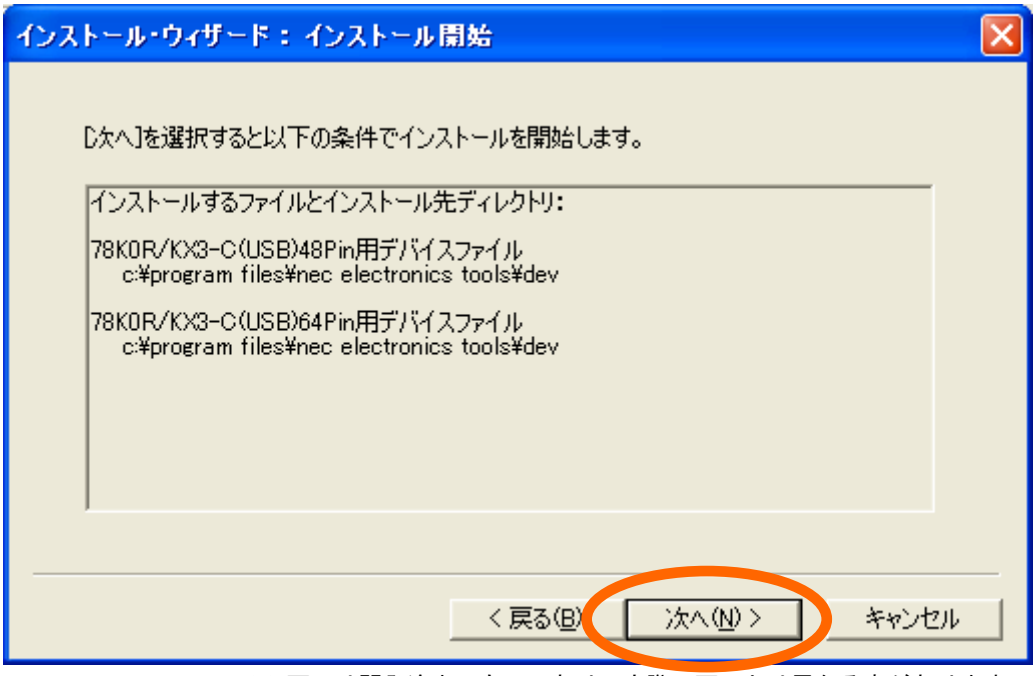


※ 画面は開発途中のものであり、実際の画面とは異なる事があります。

<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。

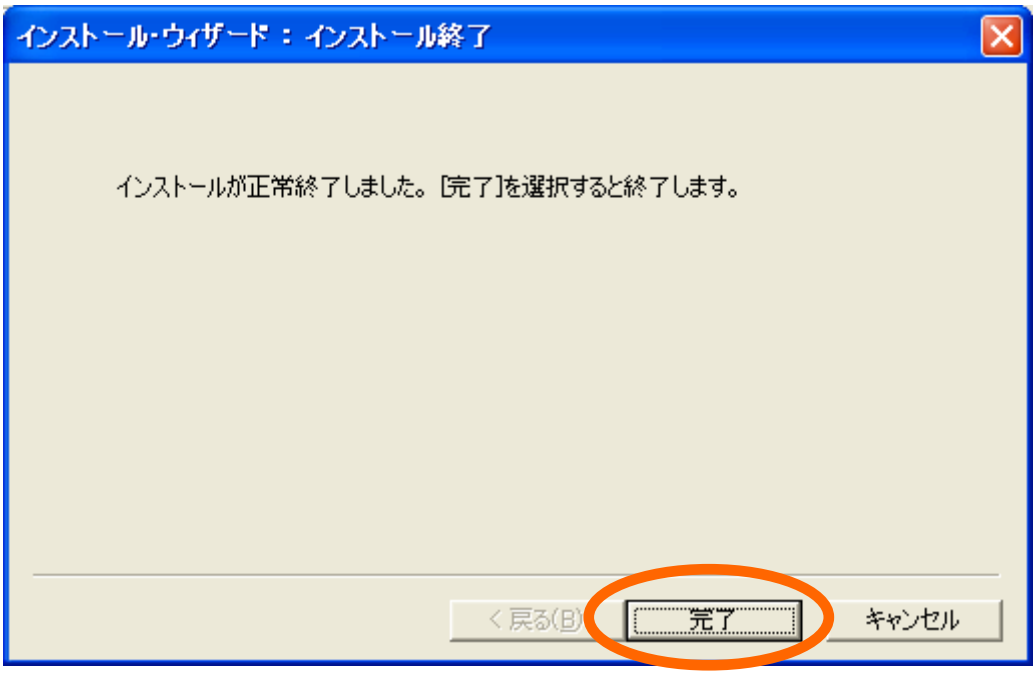


<9> 「インストール開始」ダイアログが開きます。「次へ」ボタンを押下します。



※ 画面は開発途中のものであり、実際の画面とは異なる事があります。

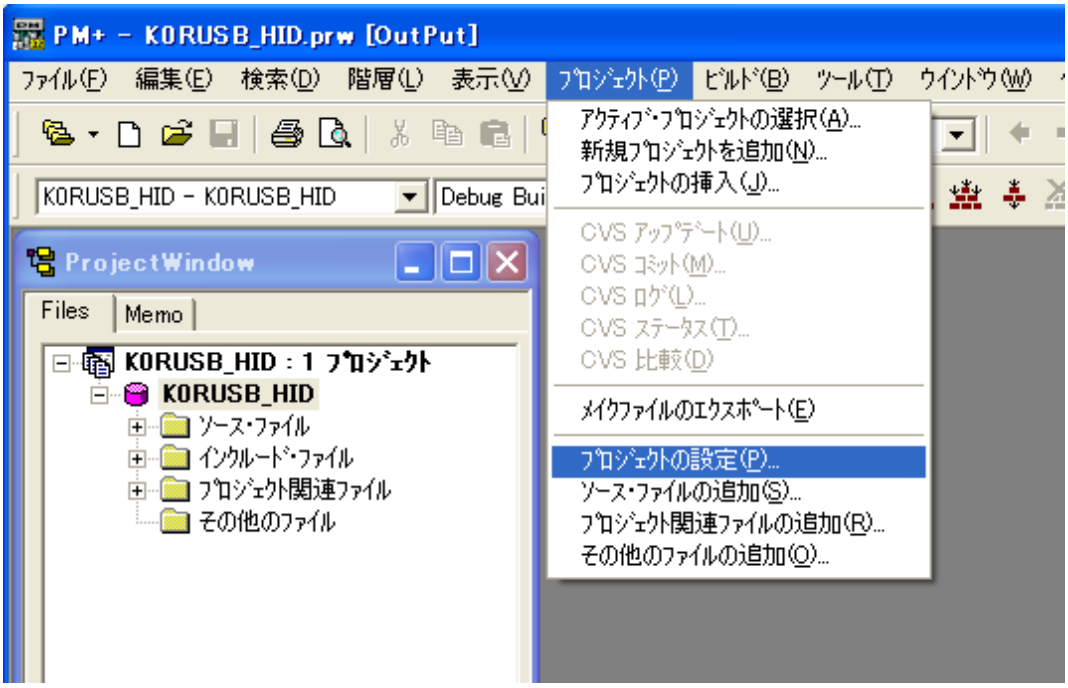
<10> デバイス・ファイルがプロジェクトにインストールされます。環境により，時間がかかる場合があります。
<11> インストールが終わると「インストール終了」ダイアログが開きます。「完了」ボタンを押下します。



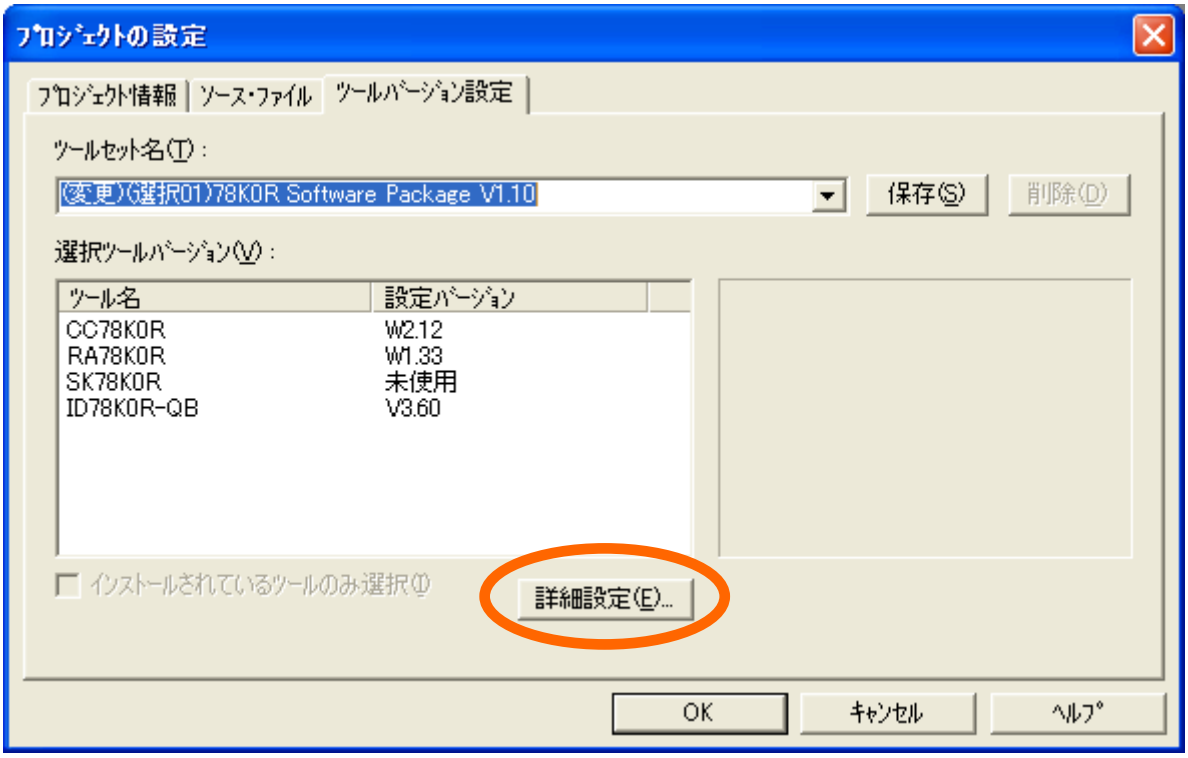
ビルド・ツールの設定

ここではビルド・ツールとしてCC78K0R、RA78K0Rを，デバッグ・ツールとしてID78K0R-QBを使用する場合の手順を示します。

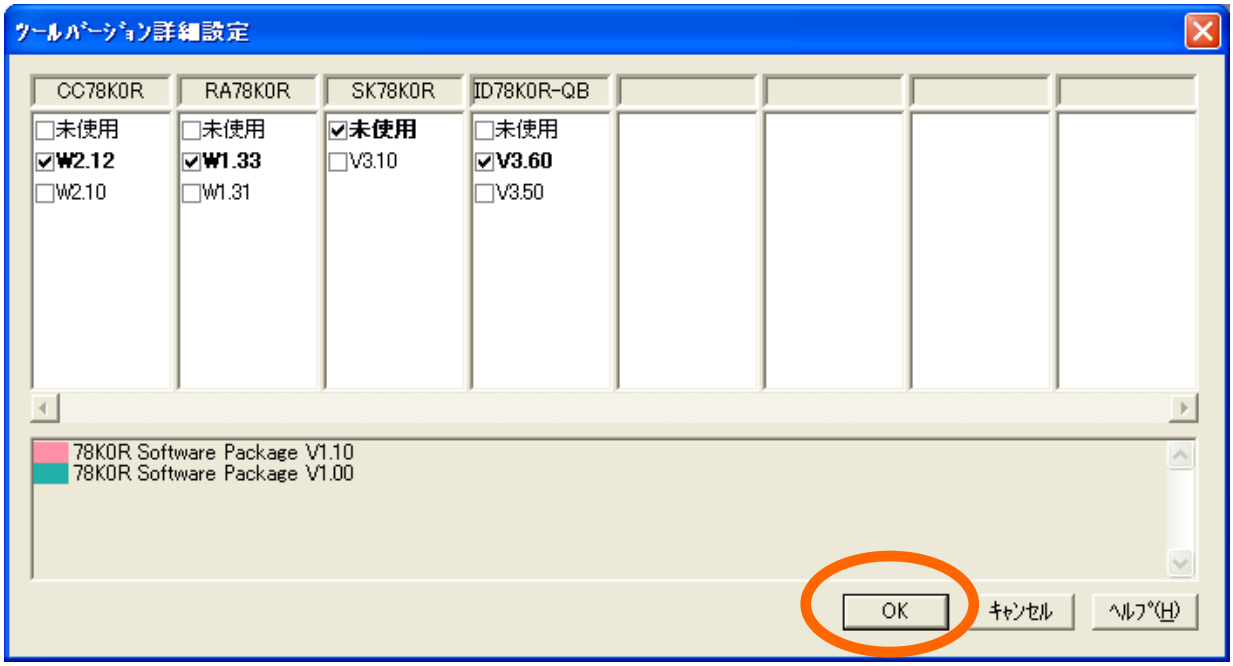
<1> PM+の「プロジェクト」メニューから「プロジェクトの設定」を選択します。



<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。「CC78K0R」「RA78K0R」の欄で使用するコンパイラのバージョンを，「ID78K0R-QB」の欄で使用するデバッガのバージョンを選択し「OK」ボタンを押下します。



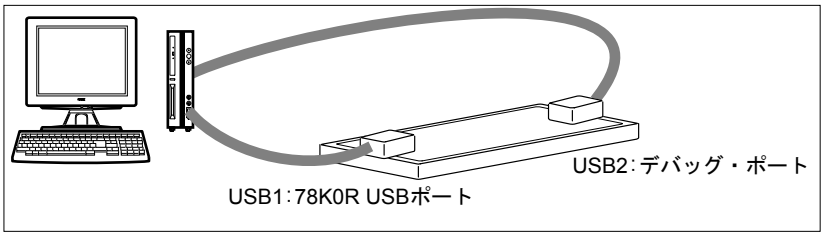
5. 2. 2　ターゲット環境整備

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-78K0R/KE3L+USB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-78K0R/KE3L+USBの接続



備考　TK-78K0R/KE3L+USBの外形やポートについては，**付録A　スタータ・キット**を参照してください。

5. 3　オンチップ・デバッグ

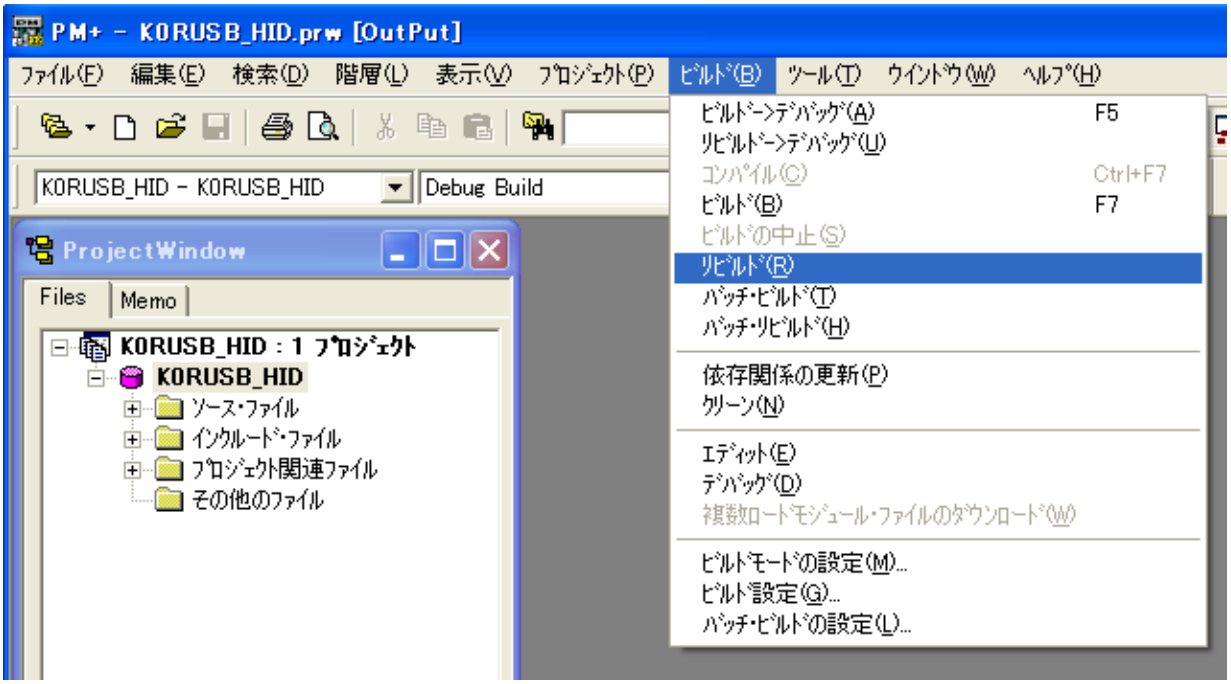
ここでは，**5. 2 環境設定**に示したワークスペースで開発したアプリケーション・プログラムのデバッグ手順について説明します。

78K0R/Kx3-L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。



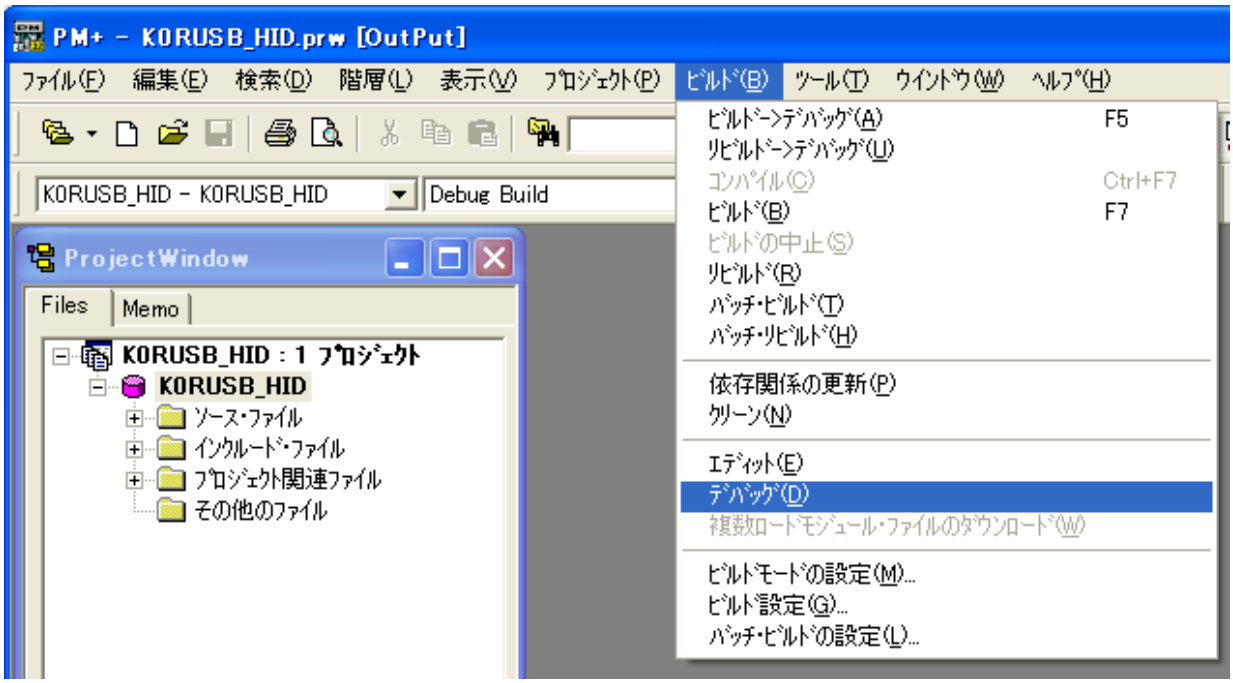
5. 3. 2　ロードと実行

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。

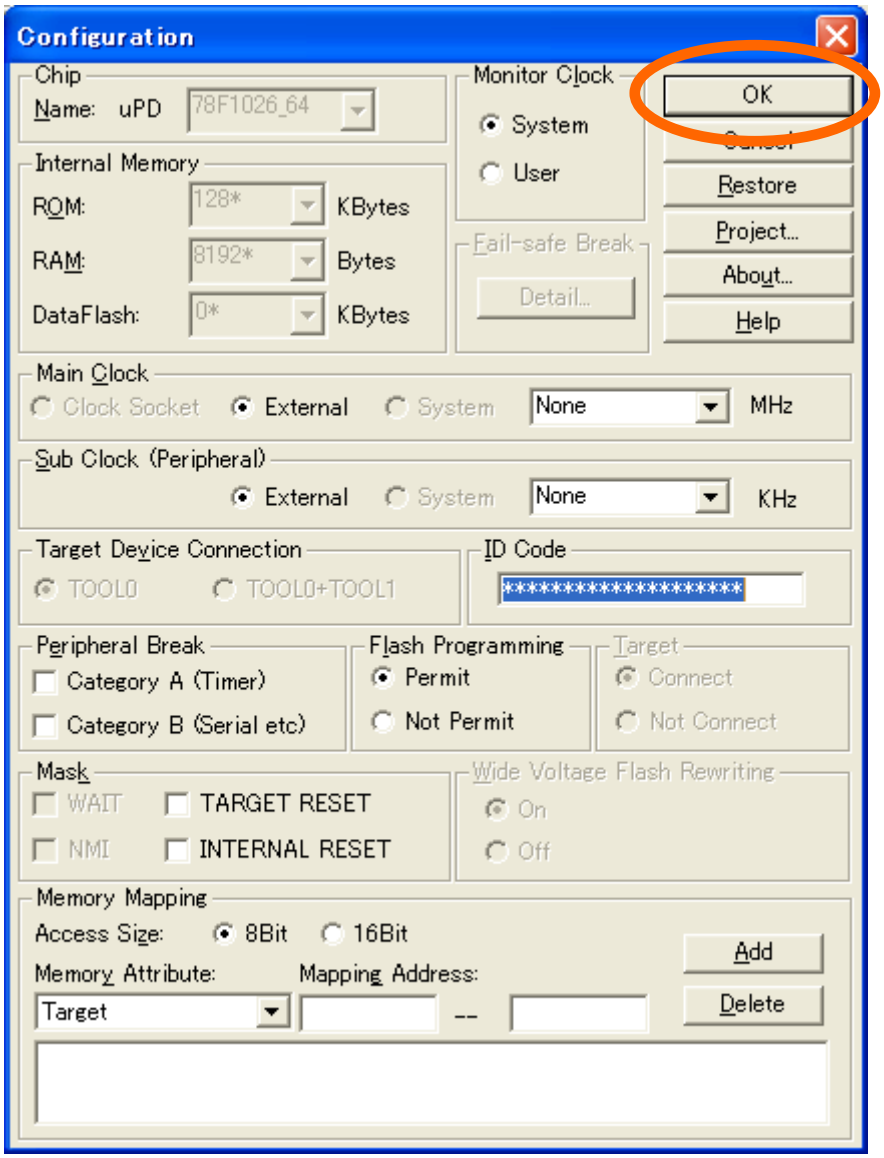
(2)　ロード・モジュールの書き込み

ここではPM+を介してTK-78K0R/KE3L+USBにロード・モジュールを書き込む手順を示します。

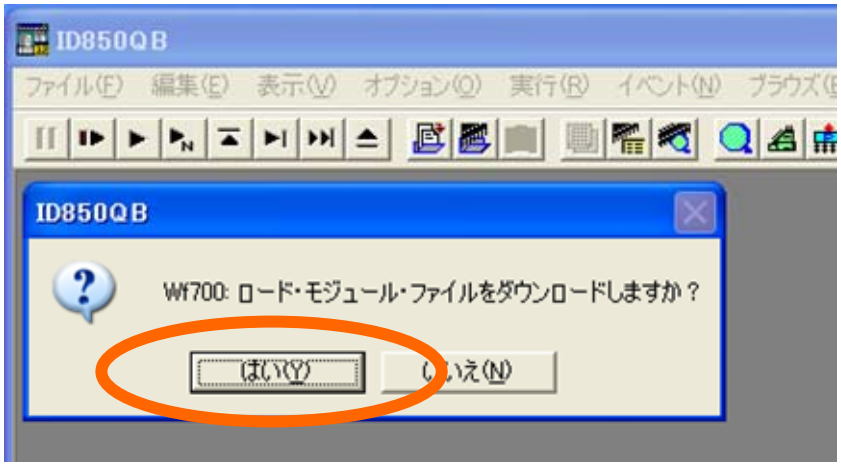
<1> 「ビルド」メニューから「デバッグ」を選択してID78K0R-QBを起動します。



<2> 「Configuration」ダイアログが開きます。「OK」ボタンを押下して下さい。

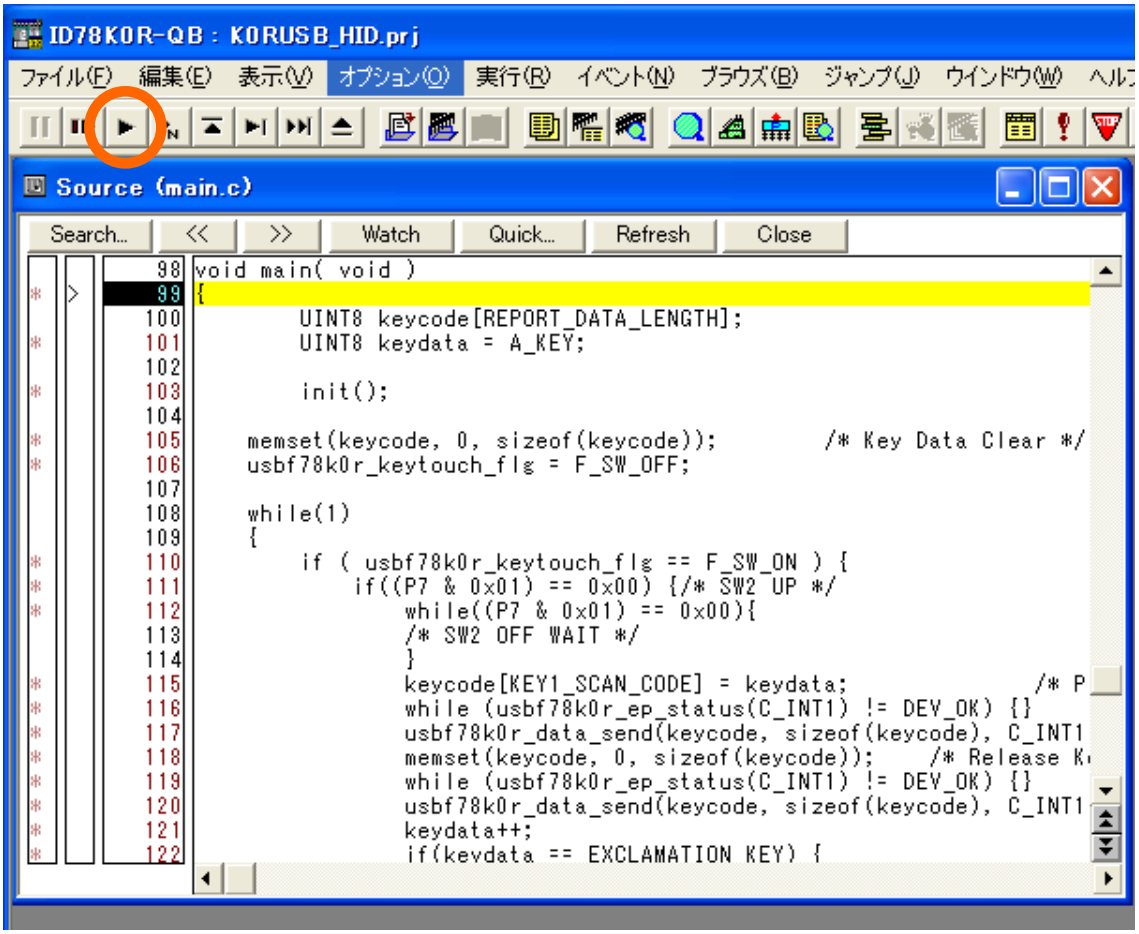


<3> サンプル・ドライバに同梱のプロジェクト・ファイルを使用している場合は，次の画面が表示されます。「はい」ボタンを押下してロード・モジュール・ファイルの書き込みを開始させます。



プログラムの実行

ID78K0R-QB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。

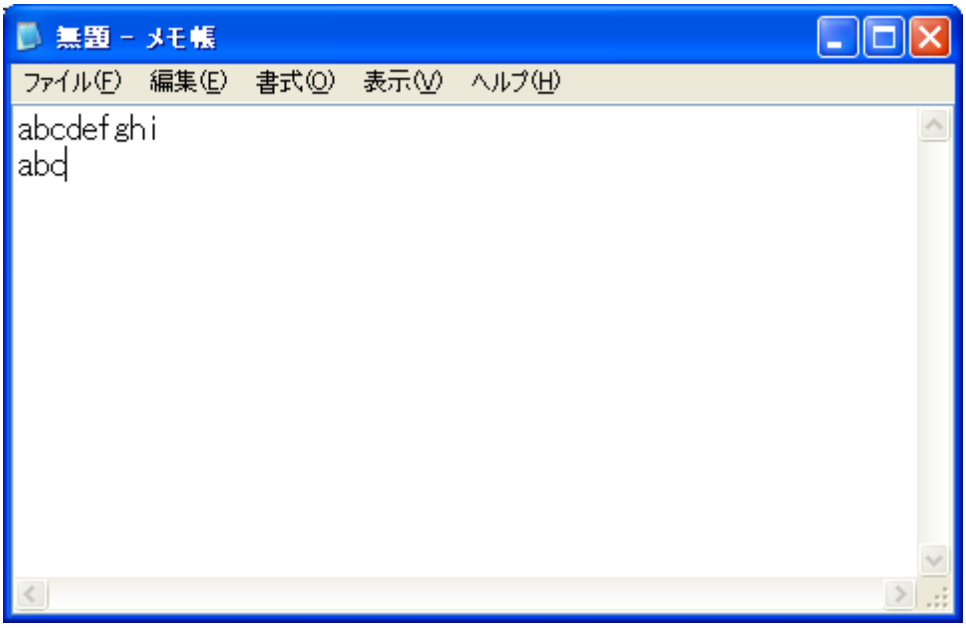


5. 4　動作確認

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-78K0R/KE3L+USB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，78K0R/Kx3-L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” ファイル内のサンプル・アプリケーション部
- “usbf78k0r.h” ファイル内の各種レジスタの設定値
- “usbf78k0r_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で使用する処理をこの部分に記述することで、その前後の初期化処理や送受信処理をそのまま利用できます。

リスト 6−1　サンプル・アプリケーションの記述

```
void main( void )
{
UINT8 keycode[REPORT_DATA_LENGTH];
UINT8 keydata = A_KEY;

init();

    memset(keycode, 0, sizeof(keycode));          /* Key Data Clear */
    usbf78k0r_keytouch_flg = F_SW_OFF;          /* Key Flag Clear */

    while(1)
    {
        if ( usbf78k0r_keytouch_flg == F_SW_ON ) {
            if((P7 & 0x01) == 0x00) {/* SW2 UP */
                while((P7 & 0x01) == 0x00){
                /* SW2 OFF WAIT */
                }
                keycode[KEY1_SCAN_CODE] = keydata;  /* Press Key Data */
                while (usbf78k0r_ep_status(C_INT1) != DEV_OK) {}
                usbf78k0r_data_send(keycode, sizeof(keycode), C_INT1);
                memset(keycode, 0, sizeof(keycode));       /* Release Key Data */
                while (usbf78k0r_ep_status(C_INT1) != DEV_OK) {}
                usbf78k0r_data_send(keycode, sizeof(keycode), C_INT1);
                keydata++;                                 /* a to z */
                if(keydata == EXCLAMATION_KEY) {
                    keydata = A_KEY;
                }
            }
            if((P7 & 0x02) == 0x00) {/* SW2 PUSH */
                while((P7 & 0x02) == 0x00){
                    /* SW2 OFF WAIT */
                }
                keycode[KEY1_SCAN_CODE] = ENTER_KEY;/* Press Key Data(Enter) */
                while (usbf78k0r_ep_status(C_INT1) != DEV_OK) {}
                usbf78k0r_data_send(keycode, sizeof(keycode), C_INT1);
                memset(keycode, 0, sizeof(keycode));       /* Release Key Data */
                while (usbf78k0r_ep_status(C_INT1) != DEV_OK) {}
                usbf78k0r_data_send(keycode, sizeof(keycode), C_INT1);
                keydata = 0x04;
            }
            usbf78k0r_keytouch_flg = F_SW_OFF;
        }
    }
}
```

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）レジスタの設定値は、”usbf78k0r.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。

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 "usbf78k0r_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。

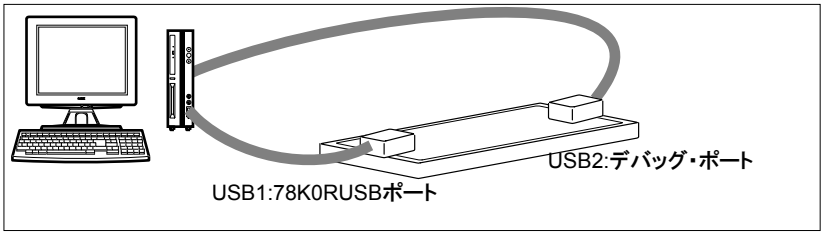
第7章 スタータ・キット

この章では，テセラ・テクノロジー社製の78K0R/Kx3-L向けスタータ・キット TK-78K0R/KE3L+USBについて説明します。

7. 1 概　　要

TK-78K0R/KE3L+USBは，78K0R/KE3-L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 7ー1　TK-78K0R/KE3L+USBの接続イメージ



7. 1. 1 特　　徴

TK-78K0R/KE3L+USB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
・ 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

7.2　主な仕様

TK-78K0R/KE3Lの主な仕様は次のとおりです。

○CPU	μPD78F1026（78K0R/KE3-L）
○動作周波数	20 MHz（USB：48 MHz）
○インタフェース	USBコネクタ（miniB）×2基 MINICUBE2用コネクタ 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DOS／V機 OS：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
〒211-8668　神奈川県川崎市中原区下沼部1753
電話（代表）：**044(435)5111**

お問い合わせ先

【ホームページ】
NECエレクトロニクスの情報がインターネットでご覧になれます。
URL（アドレス）　**http://www.necel.co.jp/**

営業関係，技術関係お問い合わせ先】
半導体ホットライン
（電話：午前 9:00～12:00，午後 1:00～5:00）
電　話　：**044-435-9494**
E-mail　：**info@necel.com**

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