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。

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 CDC（コミュニケーション・デバイス・クラス）ドライバ編

μPD78F1022
μPD78F1023
μPD78F1024
μPD78F1025
μPD78F1026

資料番号　R01AN0003JJ0100（U20312JJ1V0AN00）
発行年月　2010/04/26

〔メ　モ〕

MINICUBEは，NECエレクトロニクス株式会社の登録商標です。

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

PC/ATは，米国IBM社の商標です。

CMOSデバイスの一般的注意事項

(1) 入力端子の印加波形：入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOSデバイスの入力がノイズなどに起因して，VIL（MAX.）からVIH（MIN.）までの領域にとどまるような場合は，誤動作を引き起こす恐れがあります。入力レベルが固定な場合はもちろん，VIL（MAX.）からVIH（MIN.）までの領域を通過する遷移期間中にチャタリングノイズ等が入らないようご使用ください。

(2) 未使用入力の処理：CMOSデバイスの未使用端子の入力レベルは固定してください。未使用端子入力については，CMOSデバイスの入力に何も接続しない状態で動作させるのではなく，プルアップかプルダウンによって入力レベルを固定してください。また，未使用の入出力端子が出力となる可能性（タイミングは規定しません）を考慮すると，個別に抵抗を介してVDDまたはGNDに接続することが有効です。資料中に「未使用端子の処理」について記載のある製品については，その内容を守ってください。

(3) 静電気対策：MOSデバイス取り扱いの際は静電気防止を心がけてください。MOSデバイスは強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には，当社が出荷梱包に使用している導電性のトレーやマガジン・ケース，または導電性の緩衝材，金属ケースなどを利用し，組み立て工程にはアースを施してください。プラスチック板上に放置したり，端子を触ったりしないでください。また，MOSデバイスを実装したボードについても同様の扱いをしてください。

(4) 初期化以前の状態　電源投入時，MOSデバイスの初期状態は不定です。電源投入時の端子の出力状態や入出力設定，レジスタ内容などは保証しておりません。ただし，リセット動作やモード設定で定義している項目については，これらの動作ののちに保証の対象となります。リセット機能を持つデバイスの電源投入後は，まずリセット動作を実行してください。

(5) 電源投入切断順序　内部動作および外部インタフェースで異なる電源を使用するデバイスの場合，原則として内部電源を投入した後に外部電源を投入してください。切断の際には，原則として外部電源を切断した後に内部電源を切断してください。逆の電源投入切断順により，内部素子に過電圧が印加され，誤動作を引き起こしたり，異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源投入切断シーケンス」についての記載のある製品については，その内容を守ってください。

(6) 電源OFF時における入力信号　当該デバイスの電源がOFF状態の時に，入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により，誤動作を引き起こしたり，異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源OFF時における入力信号」についての記載のある製品については，その内容を守ってください。

はじめに

対象者　このマニュアルは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

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コミュニケーション・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 1－1　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コミュニケーション・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コミュニケーション・デバイス・クラス Ver.1.1のAbstract Control Modelに準拠
・ 仮想COM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3.0K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_communication.c	コミュニケーション・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_communication.h	usbf78k0r_communication.c関数プロトタイプ宣言
	usbf78k0r_desc.h	ディスクリプタ定義
	usbf78k0r_errno.h	エラー・コード定義
	usbf78k0r_types.h	ユーザ型宣言
infファイル	K0R_CDC_XP.inf	Windows XP用INFファイル

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 1－2　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。

コミュニケーション・デバイス・クラス（CDC）は，ホスト・コンピュータに接続する通信機器のためのクラスで，モデム，FAX，ネットワーク・カードなどが対象となります。最近ではPCにRS-232Cインタフェースが搭載されなくなっていることから，PCとUART通信を行う際のUSBシリアル変換を実現するデバイスに使われることが多くなっています。なお，CDCには実装する機器によっていくつかのモデルが定義されています。サンプル・ドライバはこのなかのAbstract Control Modelを使用しています。

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 2−2　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。サンプル・ドライバではCDCのAbstract Control Modelに対応したクラス・リクエストへの応答処理を実装しています。応答可能なリクエストは次のとおりです。

- SendEncapsulatedCommand
  コミュニケーション・クラス・インタフェースの制御プロトコルのフォーマットでコマンドを発行するためのリクエストです。

- GetEncapsulatedResponse
  コミュニケーション・クラス・インタフェースの制御プロトコルのフォーマットで応答を要求するためのリクエストです。

- SetLineCoding
  シリアル通信の通信フォーマットを指定するためのリクエストです。

- GetLineCoding
  デバイス側の現在の通信フォーマット設定を取得するためのリクエストです。

- SetControlLineState
  RS-232/V.24形式の制御信号のためのリクエストです。

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。

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 2－7　エンドポイント・ディスクリプタのフォーマット

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

表 2－8　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

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コミュニケーション・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、INTUSB割り込みにより通知されます。大きく分けて、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CPUDEC割り込みと、バルク・アウト転送（受信）用エンドポイント(Endpoint2)に対してデータが正常受信されたことを示すBKO1DT割り込みがあります。Endpoint0の処理では、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。

(3) サンプル・アプリケーション

バルク・アウト転送（受信）用エンドポイントにあるデータを読み出し、読み出したデータをそのままバルク・イン転送（送信）用エンドポイントに書き込みます。詳細は、第4章 サンプル・アプリケーションの仕様 を参照してください。

3. 1. 2 リクエストへの対応

ここでは，サンプル・ドライバが対応するUSBリクエストについて説明します。

(1) 標準リクエスト

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_string

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

（b） その他

サンプル・ドライバはSTALL応答を返します。

(2) クラス・リクエスト

CDC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） SendEncapsulatedCommand

CDCインタフェースの制御プロトコルのフォーマットでコマンドを発行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・ドライバはリクエストに付随するデータを取り込み，送信処理（バルク・イン転送）を行います。

（b） GetEncapsulatedResponse

CDCインタフェースの制御プロトコルのフォーマットで応答を要求するためのリクエストです。サンプル・ドライバは現在，このリクエストをサポートしていません。

（c） SetLineCoding

シリアル通信の通信フォーマットを指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・ドライバはリクエストに付随するデータを取り込んで通信レートなどを設定し，NULLパケットの送信処理（コントロール・リード転送）を行います。

（d） GetLineCoding

デバイス側の現在の通信フォーマット設定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・ドライバは通信レートなどの設定を読み出し，送信処理（コントロール・リード転送）を行います。

（e） SetControlLineState

RS-232/V.24形式の制御信号の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NULLパケットの送信処理（コントロール・リード転送）を行います。

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：CDC
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	0x0030	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：48バイト
bNumInterfaces	1	0x02	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：2
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	0x80	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・ウエイクアップなし
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	0x02	クラス・コード：コミュニケーション・インタフェース・クラス
bInterfaceSubClass	1	0x02	サブクラス・コード：Abstract Control Model
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

表 3−4　Interface1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	0x01	このインタフェースの識別番号：1
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	0x02	このインタフェースが持つエンドポイントの数：2
bInterfaceClass	1	0x0A	クラス・コード：コミュニケーション・インタフェース・クラス
bInterfaceSubClass	1	0x00	サブクラス・コード：Abstract Control Model
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 3−5　Endpoint7のエンドポイント・ディスクリプタの設定

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

表 3−6　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	0x02	このエンドポイントの転送タイプ：バルク
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

表 3−7　Endpoint2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	0x02	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：2
bmAttributes	1	0x02	このエンドポイントの転送タイプ：バルク
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 3−8　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：CDCDrv（CDC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

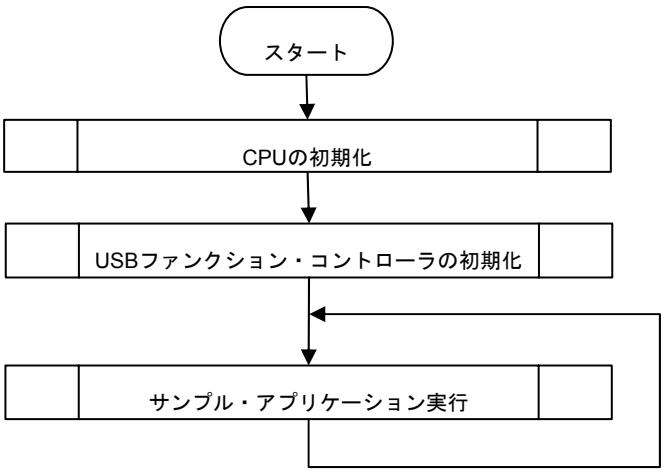
注1. bStringフィールドのサイズにより設定値が異なります。

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。

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。

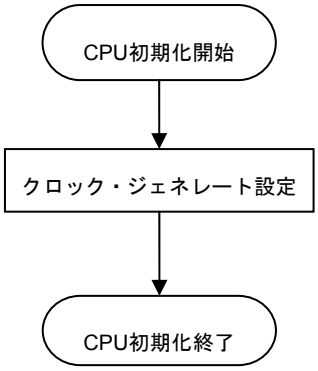
図 3−1　サンプル・ドライバの処理フロー



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/周辺ハードウェア･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。

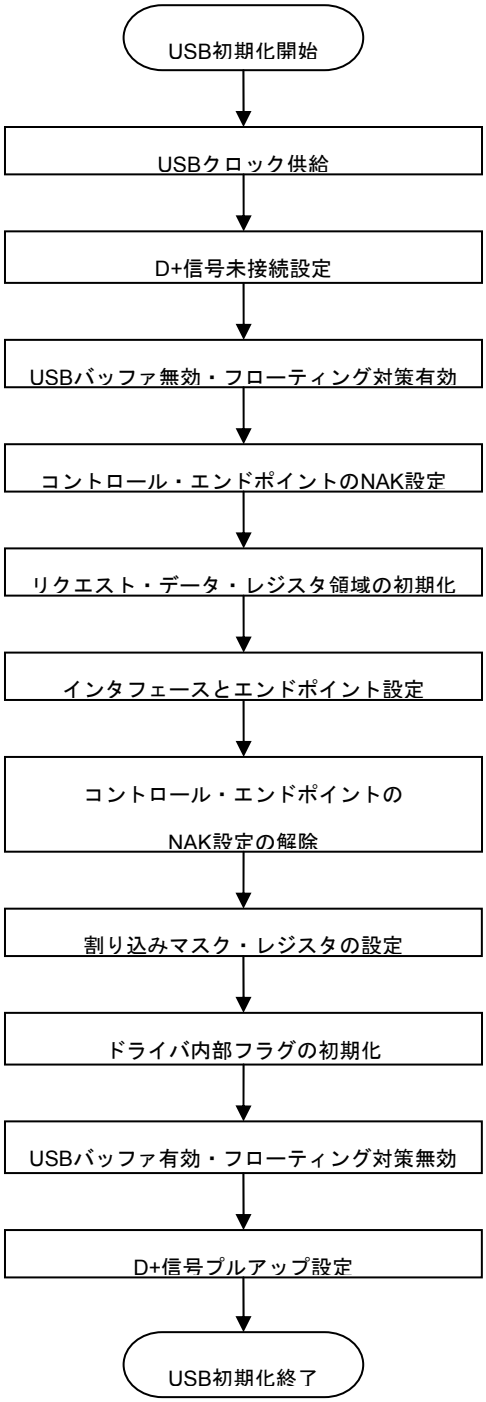
(g) PLLC レジスタの PLLM ビットに “1” を書き込みます。この設定により、PLL への供給クロックの逓倍率を 12 逓倍に設定します。

(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。

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 ″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。
(b) 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。
(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 ″0x80″ を書き込みます。この設定により，2つまでのインタフェースを有効にします。
(b) UF0AASレジスタに ″0x00″ を書き込みます。この設定により，オルタナティブ設定を無効にします。
(c) UF0E1IMレジスタに ″0x40″ を書き込みます。この設定により，Endpoint1がInterface1にリンクされます。
(d) UF0E2IMレジスタに ″0x40″ を書き込みます。この設定により，Endopoint2がIntreface1にリンクされます。
(e) UF0E7IMレジスタに ″0x20″ を書き込みます。この設定により，Endopoint7がIntreface0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 “0x7B” を書き込みます。この設定により、UF0IS0 レジスタに示される割り込み要因のうち、BUSRST 割り込みと 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 “0xFE” 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。

(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 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。

(i) 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。

(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。

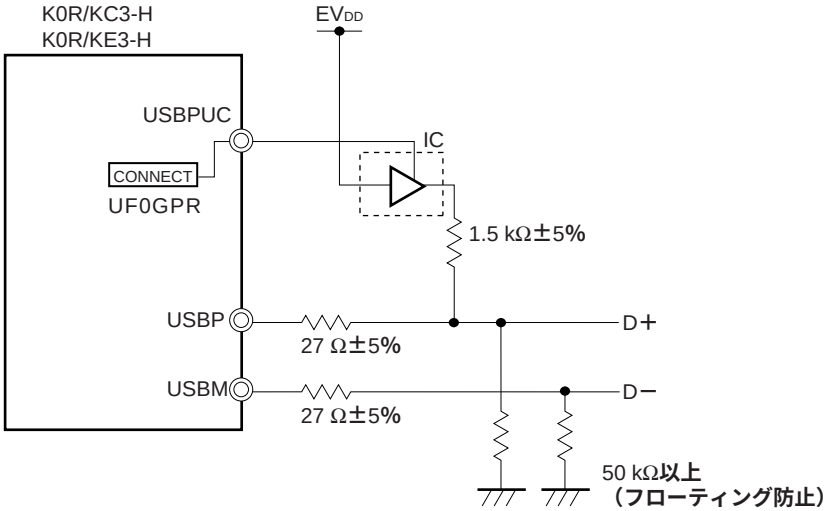
(10)USB バッファ有効・フローティング対策無効

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。

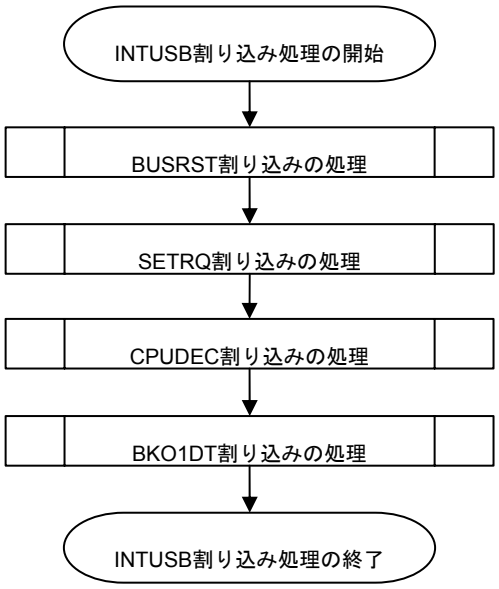
図 3−4　USBファンクション・コントローラ接続例



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　Endpoint0監視の処理フロー



(1)　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。

(2)　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。

(3) CPUDEC 割り込みの処理

FW処理のリクエストを受信した場合、通知されます。
以下の手順で処理を行います。

(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。

(4) BKO1DT 割り込みの処理

UF0BO1レジスタにデータを正常に受信した場合、通知されます。
以下の手順で処理を行います。

(a) UF0IC3 レジスタに “0xFE” を書き込みます。この設定により、BKO1DT 割り込みがクリアされます。

(b) 受信したデータの存在を示すフラグ（usbf78k0r_rdata_flg）をセット (1) します。この設定により、ドライバ内でバルク・アウト・エンドポイントに受信データがあることを示します。このフラグは、サンプル・ドライバで独自に定義しています。

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 3−9　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	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_intusb	INTUSB割り込み処理
	usbf78k0r_standardreq	標準リクエスト処理
	usbf78k0r_getdesc	GET_DESCRIPTOR（String）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_rdata_length	バルク・アウト・エンドポイントの受信データサイズ通知処理
	usbf78k0r_data_receive	バルク・アウト・エンドポイントの受信処理
	usbf78k0r_fifo_clear	バルク/インタラプト・イン・エンドポイント、および、バルク・アウト・エンドポイントのFIFOクリア処理
usbf78k0r_communication.c	usbf78k0r_classreq	CDCクラス・リクエスト処理関数
	usbf78k0r_send_encapsulated_command	Send Encapsulated Commandリクエストの処理
	usbf78k0r_get_encapsulated_response	Get Encapsulated Responseリクエストの処理
	usbf78k0r_set_line_coding	Set Line Codingリクエストの処理
	usbf78k0r_get_line_coding	Get Line Codingリクエストの処理
	usbf78k0r_set_control_line_state	Set Control Line Stateリクエストの処理
	usbf78k0r_buff_init	CDCデータ転送用エンドポイントのFIFOクリア処理
	usbf78k0r_get_bufinit_flg	FIFO初期化処理の実行状態通知処理
	usbf78k0r_send_buf	CDCデータの送信処理
	usbf78k0r_recv_buf	CDCデータの受信処理

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

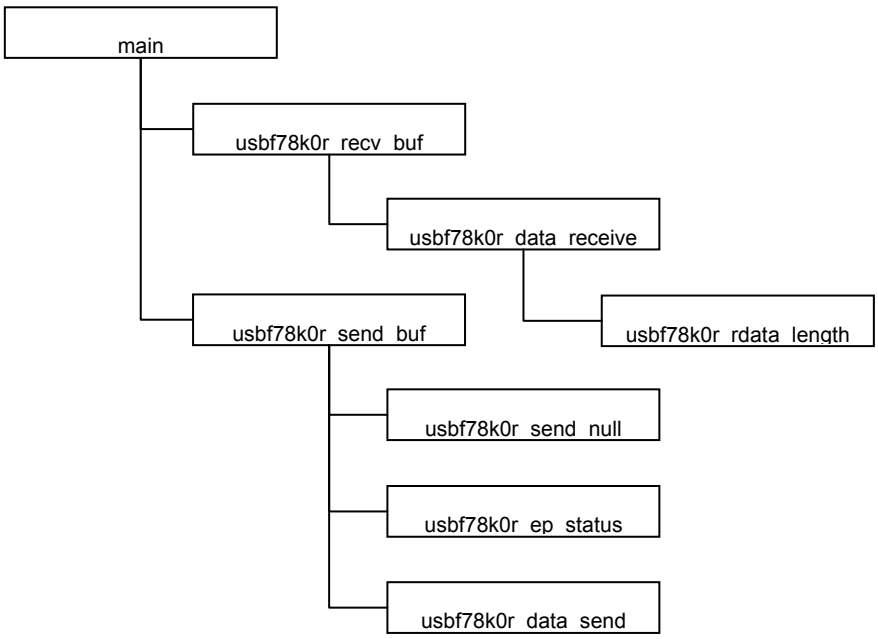


図 3−7　USBファンクション・コントローラ用処理での関数の呼び出し

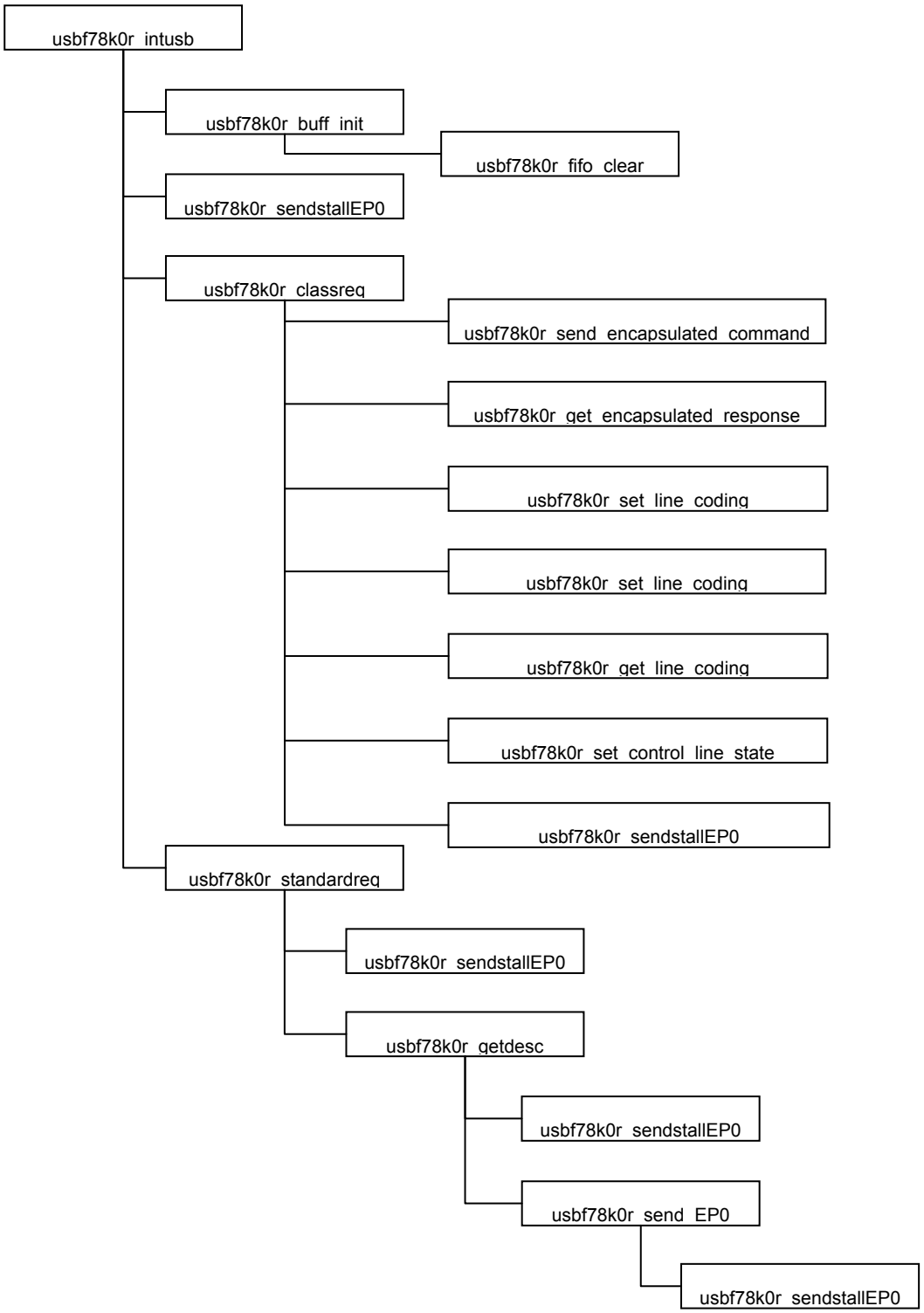


図 3−8　USBコミュニケーション・クラス用処理での関数の呼び出し(1)

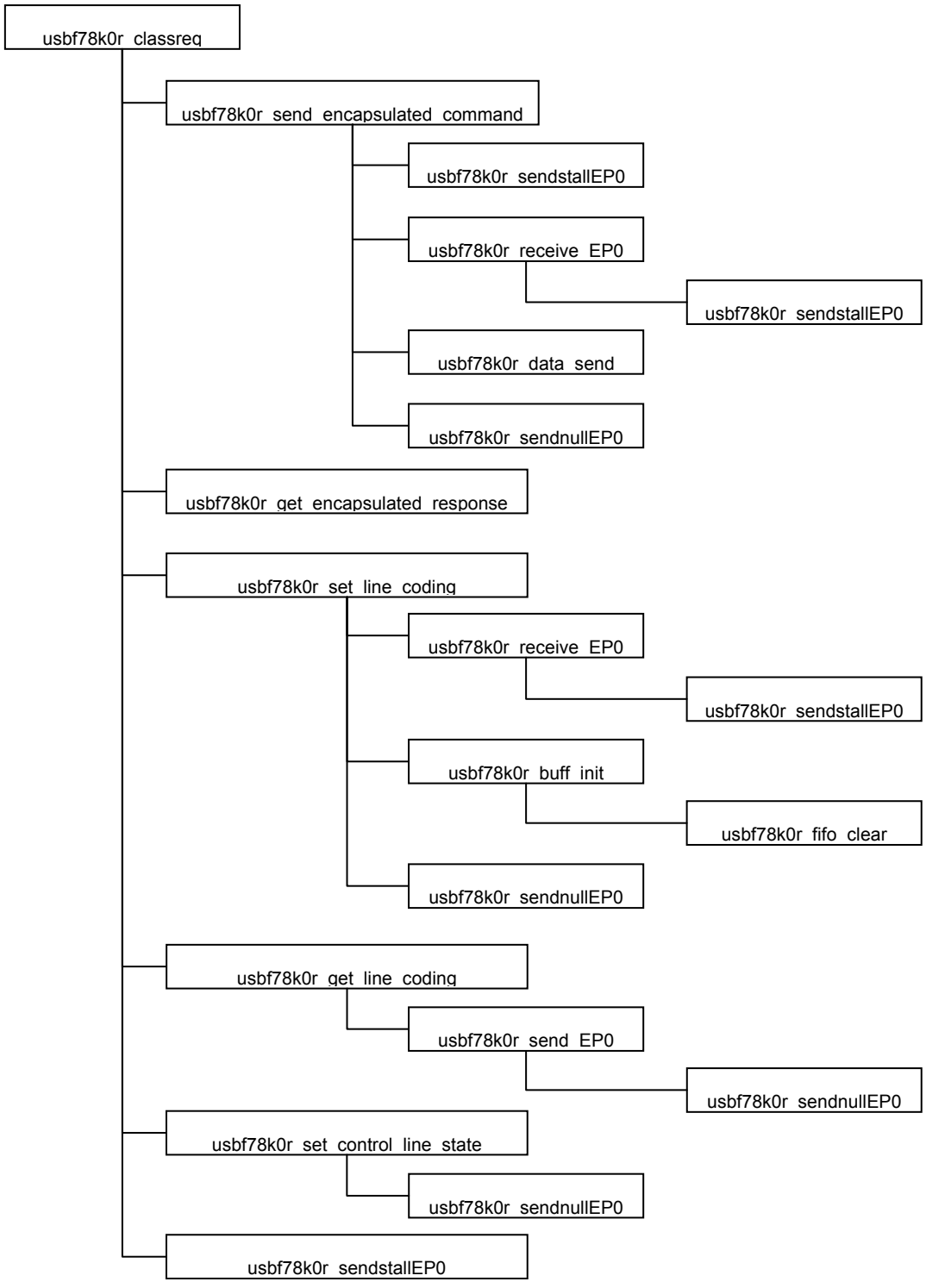
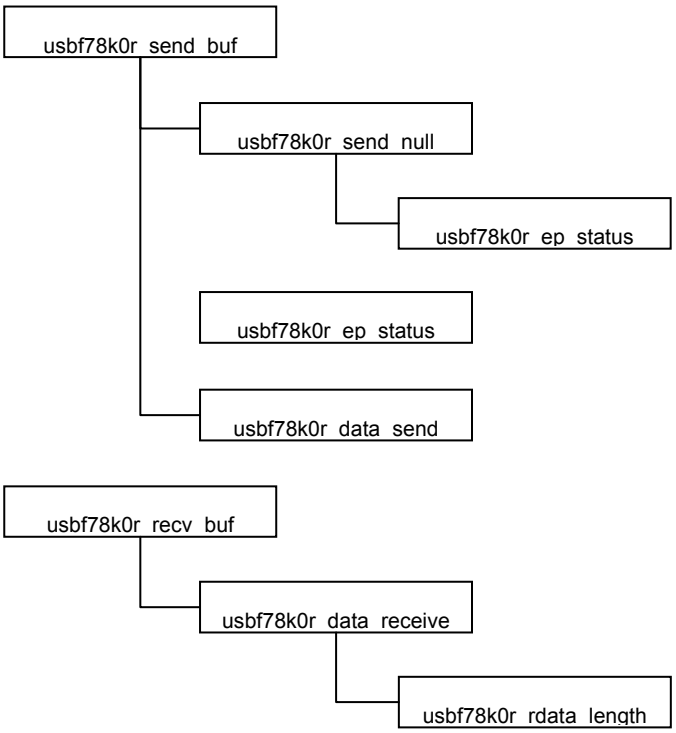


図 3－9　USBコミュニケーション・クラス用処理での関数の呼び出し(2)



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。

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

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_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。

usbf78k0r_rdata _length

【概要】
受信データサイズ取得

【C言語記述形式】

void usbf78k0r_rdata_length(INT32 *len , INT8 ep)

【パラメータ】

パラメータ	説明
INT32*　len	受信データサイズ格納アドレス・ポインタ
INT8　ep	データ受信エンドポイント番号

【戻り値】

なし

【機能】
指定されたエンドポイント（受信用）の受信データサイズを読み出します。

usbf78k0r_data _receive

【概要】

バルク・アウト・エンドポイント用USBデータ受信処理

【C言語記述形式】

INT32 usbf78k0r_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	受信データサイズ
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
len (>= 0)	正常に受信したデータサイズ
DEV_ERROR	異常終了

【機能】

指定されたエンドポイント用の受信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_fifo_clear

【概要】

バルク / インタラプト・エンドポイント用FIFOクリア処理

【C言語記述形式】

void usbf78k0r_fifo_clear(INT8 in_ep, INT8 out_ep)

【パラメータ】

パラメータ	説明
INT8　in_ep	データ送信エンドポイント番号
INT8　out_ep	データ受信エンドポイント番号

【戻り値】

なし

【機能】

バルク / インタラプト・エンドポイントにおいて指定されたエンドポイントのFIFOをクリアし、データ受信フラグ(usbf78k0r_rdata_flg)をクリア(0)します。

USBコミュニケーション・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コミュニケーション・クラス・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_send_encapsulated_command

【概要】

Send Encapsulated Commandリクエスト処理

【C言語記述形式】

void usbf78k0r_send_encapsulated_command(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Send Encapsulated Commandだった場合、呼び出される関数です。Endpoint0用USBデータ受信処理関数（usbf78k0r_receive_EP0）を呼び出して受信したデータを取り込み、データ送信処理関数（usbf78k0r_data_send）を呼び出してEndpoint2からバルク・イン転送（送信）で データを送信させます。その後、Endpoint0用NULLパケット送信処理関数（usbf78k0r_sendnullEP0）を呼び出します。

usbf78k0r_set_line_coding

【概要】
Set Line Codingリクエスト処理

【C言語記述形式】

void usbf78k0r_set_line_coding(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 Line Codingだった場合、呼び出される関数です。Endpoint0用USBデータ受信処理関数（usbf78k0r_receive_EP0）を呼び出して受信したデータを取り込み、UART_MODE_INFO構造体に書き込みます。その後、ユーザ・データ用FIFOの初期化処理関数(usbf78k0r_buff_init)、および、Endpoint0用NULLパケット送信処理関数（usbf78k0r_sendnullEP0）を呼び出します。

usbf78k0r_get_control_line_coding

【概要】
Get Line Codingリクエスト処理

【C言語記述形式】
void usbf78k0r_get_line_coding(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 Line Codingだった場合、呼び出される関数です。Endpoint0用USBデータ送信処理関数（usbf78k0r_send_EP0）を呼び出してUART_MODE_INFO構造体の値をEndpoint0から送信させます。

usbf78k0r_set_control_line_state

【概要】

Set Control Line Stateリクエスト処理

【C言語記述形式】

void usbf78k0r_set_control_line_state(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 Control Line Stateだった場合、呼び出される関数です。Endpoint0用NULLパケット送信処理関数（usbf78k0r_sendnullEP0）を呼び出します。

usbf78k0r_buff_init

【概要】

ユーザ・データ用FIFOの初期化処理

【C言語記述形式】

void usbf78k0r_buff_init(void)

【パラメータ】

なし

【戻り値】

なし

【機能】

バルク / インタラプト・エンドポイント用FIFOクリア処理関数(usbf78k0r_fifo_clear)を呼び出し、コミュニケーション・クラスのユーザ・データ用FIFOの初期化を行います。また、ドライバ内部の送信パケットサイズのクリア(0)、FIFOが初期化されたことを示すフラグ(usbf78k0r_bufinit_flg)のセット(1)を行います。

usbf78k0r_get_bufinit_flg

【概要】

ユーザ・データ用FIFOの状態通知処理

【C言語記述形式】

INT32 usbf78k0r_get_bufinit_flg(void)

【パラメータ】

なし

【戻り値】

シンボル	説明
DEV_OK	通常状態
DEV_ERROR	FIFO初期化状態

【機能】

FIFOが初期化されたことを示すドライバ内部のフラグ(usbf78k0r_bufinit_flg)の状態を通知します。フラグがセット(1)されていれば、FIFOが初期化されたことを示します。このとき、初期化された状態であることを通知すると共にフラグのクリア(0)を行います。

usbf78k0r_send_buf

【概要】

コミュニケーション・クラス用ユーザ・データ送信処理

【C言語記述形式】

INT32 usbf78k0r_send_buf(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
len (>= 0)	正常に送信したデータサイズ
DEV_ERROR	異常終了

【機能】

送信データサイズ（パラメータ：len）が0で、前に送信されたパケットのサイズ(g_send_size)がMax Packet Sizeに等しかった場合、バルク / インタラプト・イン・エンドポイント用NULLパケット送信処理関数(usbf78k0r_send_null)を呼び出しNULLパケットの送信を行います。送信データサイズ（パラメータ：len）が0より大きく、送信FIFOが空の状態（usbf78k0r_ep_status関数の返り値がDEV_OK）であれば、USBデータ送信処理関数(usbf78k0r_data_send)を呼び出します。データの送信が正常に終了すると、ドライバ内部で定義された送信完了パケットサイズ(g_send_size)に送信されたデータサイズを格納します。

usbf78k0r_recv_buf

【概要】

コミュニケーション・クラス用ユーザ・データ受信処理

【C言語記述形式】

INT32 usbf78k0r_recv_buf(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
len (>= 0)	正常に受信したデータサイズ
DEV_ERROR	異常終了

【機能】

USBデータ受信処理関数(usbf78k0r_data_receive)を呼び出します。

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

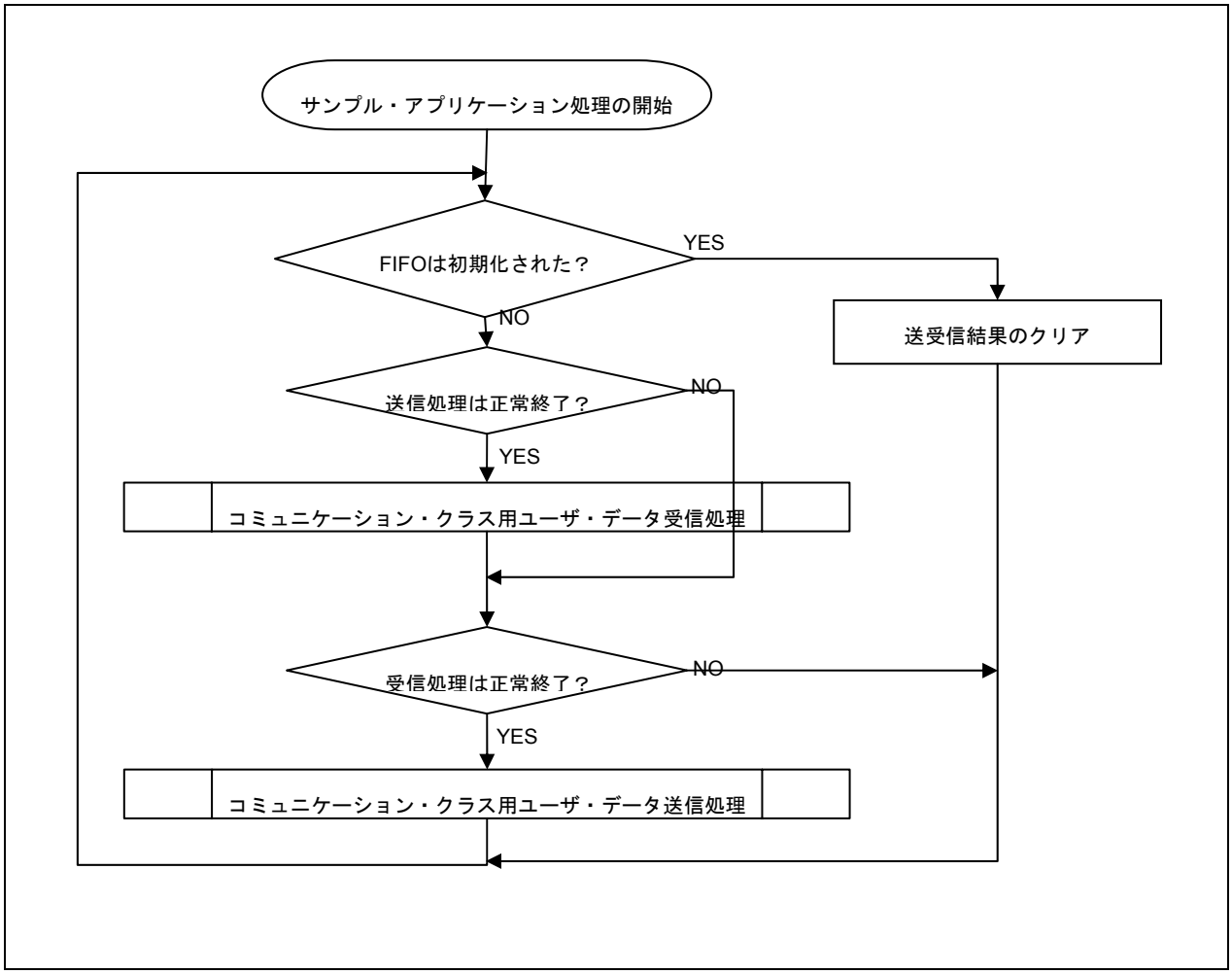
サンプル・アプリケーションは、USBコミュニケーション・デバイス・クラス用ドライバの利用の簡単な例として用意されたもので、サンプル・ドライバのメイン・ルーチンに組み込まれています。

このサンプル･アプリケーションは､USBファンクション･コントローラで受信したデータの読み出し、読み出したデータの送信、の一連の処理を実行します。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) ユーザ・データ用 FIFO 初期化確認

ユーザ・データ用FIFOの状態通知処理関数（usbf78k0r_get_bufinit_flg）を呼び出し、通常状態であれば送信処理結果の確認処理へ、初期化された状態であれば送受信結果のクリア処理（コミュニケーション・クラス用ユーザ・データ送信 / 受信処理結果の0クリア）を行います。

(2) コミュニケーション・クラス用ユーザ・データ送信処理結果確認

コミュニケーション・クラス用ユーザ・データ送信処理の結果が正常終了（および、初期状態）であればコミュニケーション・クラス用ユーザ・データ受信処理へ、異常終了状態であれば受信処理結果の確認処理へ移行します。

(3) コミュニケーション・クラス用ユーザ・データ受信処理

受信データを格納するバッファ・アドレス、バッファのサイズを指定し、コミュニケーション・クラス用ユーザ・データ受信処理関数（usbf78k0r_recv_buf）を呼び出します。

(4) コミュニケーション・クラス用ユーザ・データ受信処理結果確認

コミュニケーション・クラス用ユーザ・データ受信処理の結果が正常終了（および、初期状態）であればコミュニケーション・クラス用ユーザ・データ送信処理へ、異常終了状態であればユーザ・データ用FIFO初期化確認処理へ移行します。

(5) コミュニケーション・クラス用ユーザ・データ送信処理

送信したいデータが格納されているバッファ・アドレス、送信するデータサイズを指定し、コミュニケーション・クラス用ユーザ・データ送信処理関数（usbf78k0r_send_buf）を呼び出します。

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” と ”usbf78k0r_communication.h” の2つのヘッダ・ファイルをインクルードしています。また、ユーザ・データ用に1パケットのデータを処理するのに十分なサイズのユーザ・バッファ（UserBuf）を用意しています。（USBのFull Speedにおけるバルク・エンドポイントの最大パケット・サイズは64Byteです。）

(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) ユーザ・データ用 FIFO 状態確認

ユーザ・データ用FIFOの状態通知処理関数（usbf78k0r_get_bufinit_flg）を呼び出し、FIFOの状態を確認します。

(5) ユーザ・データの受信処理

コミュニケーション・クラス用ユーザ・データ受信処理関数（usbf78k0r_recv_buf）を呼び出し、結果を保存します。

(6) ユーザ・データの送信処理

コミュニケーション・クラス用ユーザ・データ送信処理関数（usbf78k0r_send_buf）を呼び出し、結果を保存します。

(7) 送受信処理結果クリア処理

ユーザ・データ用FIFOが初期化されると、(5), (6) で保存されている送受信処理結果をクリア(0)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)
        {
            INT32 rcv_ret = 0;
            INT32 snd_ret = 0;

            cpu_init();

            DI();
            usbf78k0r_init();           /* initial setting of the USB Function */
            EI();

            while(1)
            {
                if (usbf78k0r_get_bufinit_flg() != DEV_ERROR) {
                    if (snd_ret >= 0) {
                        rcv_ret = usbf78k0r_recv_buf(&UserBuf[0], USERBUF_SIZE);
                    }
                    if (rcv_ret >= 0) {
                        snd_ret = usbf78k0r_send_buf(&UserBuf[0], rcv_ret);
                    }
                }
                else {
                    snd_ret = 0;
                    rcv_ret = 0;
                }
            }
        }
```

第5章 開発環境

この章では，78K0R/Kx3-L向けUSBコミュニケーション・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。

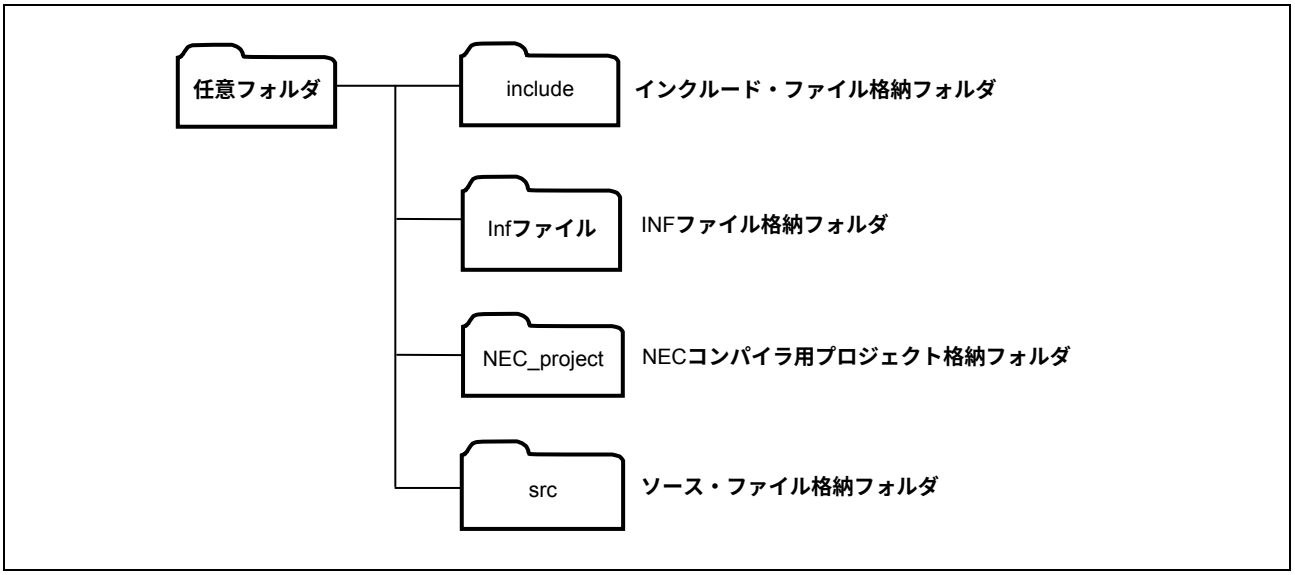
統合開発ツールのインストール

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。

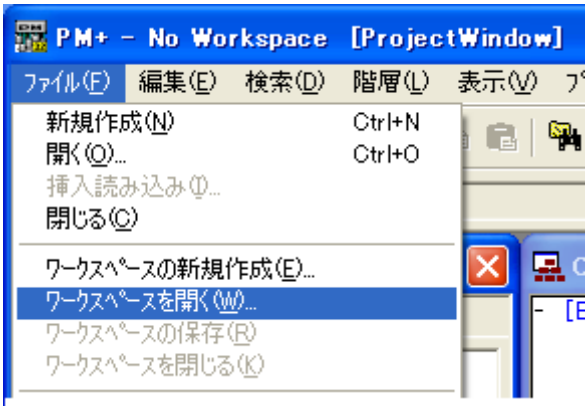
図 5−1　サンプル・ドライバのフォルダ構成



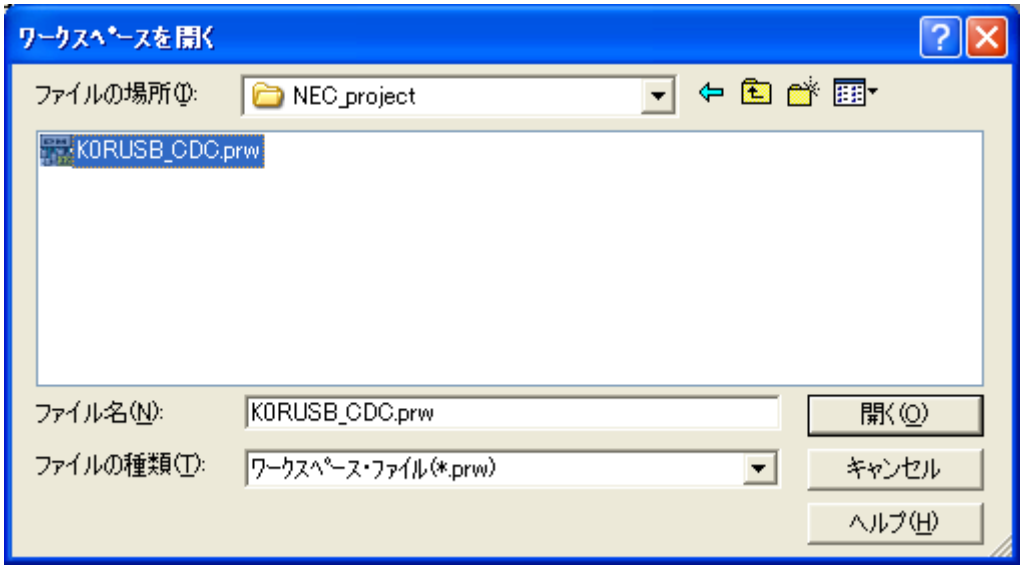
ワークスペースの設定

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。



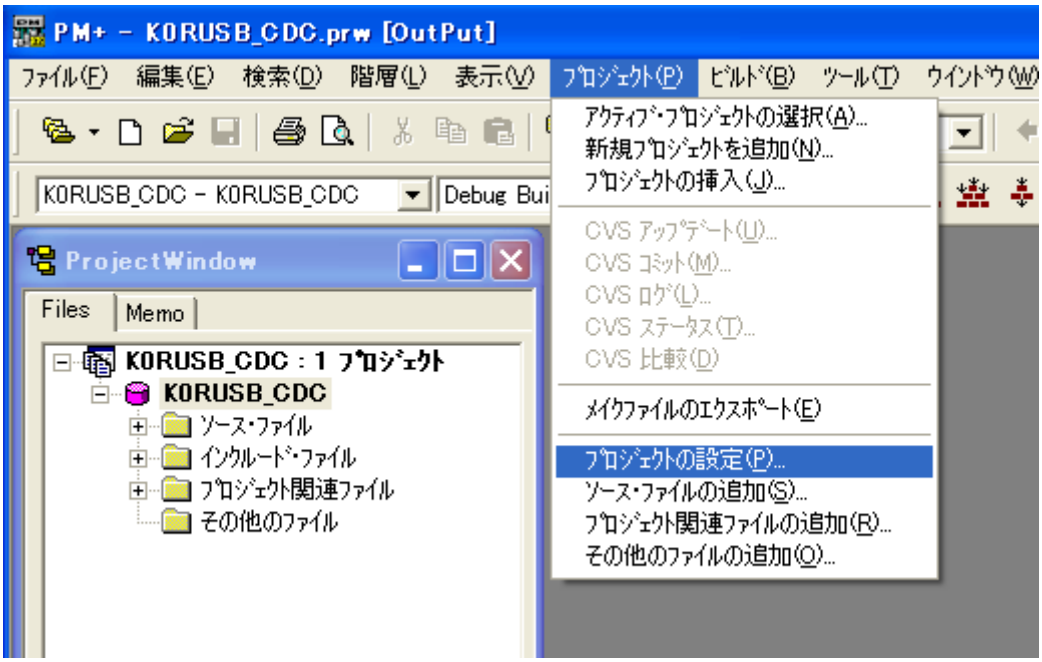
<2> 「ワークスペースを開く」ダイアログが開きます。サンプル・ドライバを格納したディレクトリの「NEC_project」フォルダにあるワークスペース・ファイルを指定します。



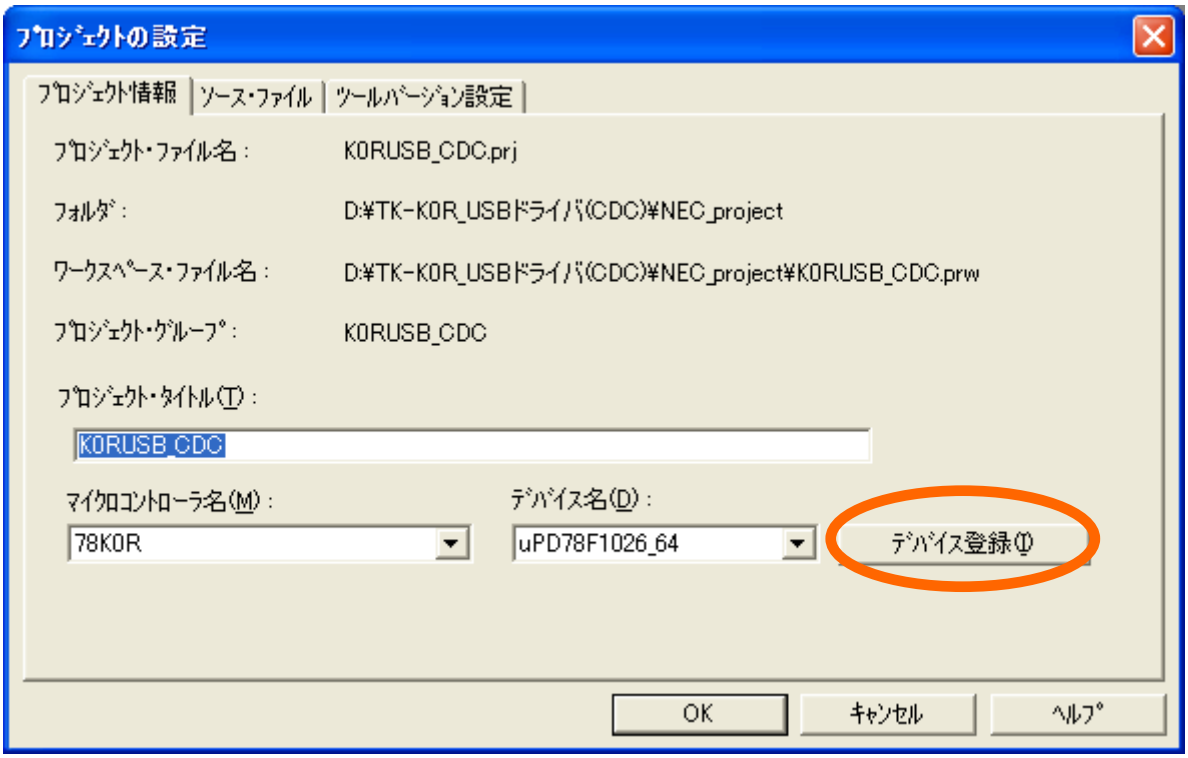
デバイス・ファイルのインストール

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。

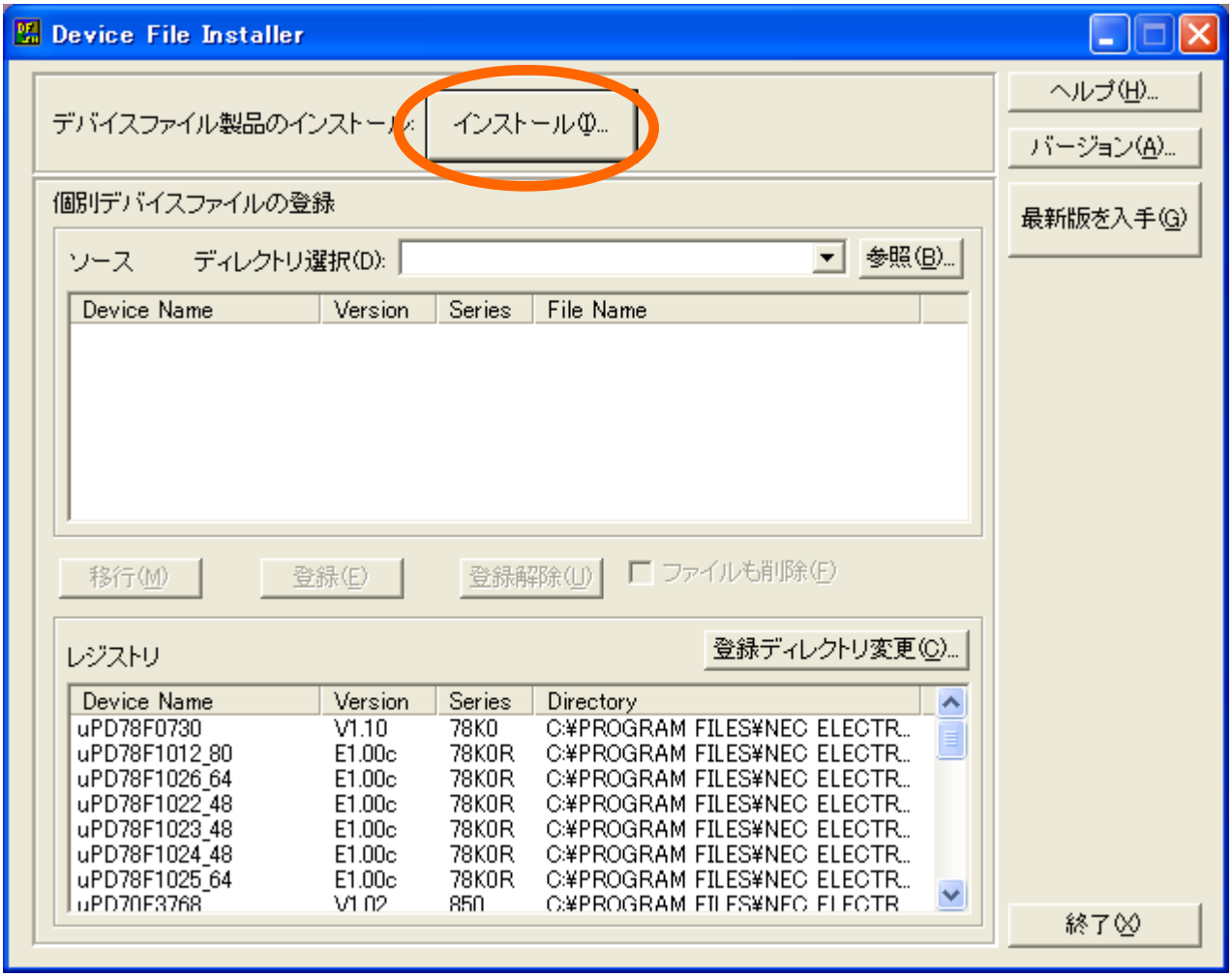
<1> PM+の「プロジェクト」メニューから「プロジェクトの設定」を選択します。



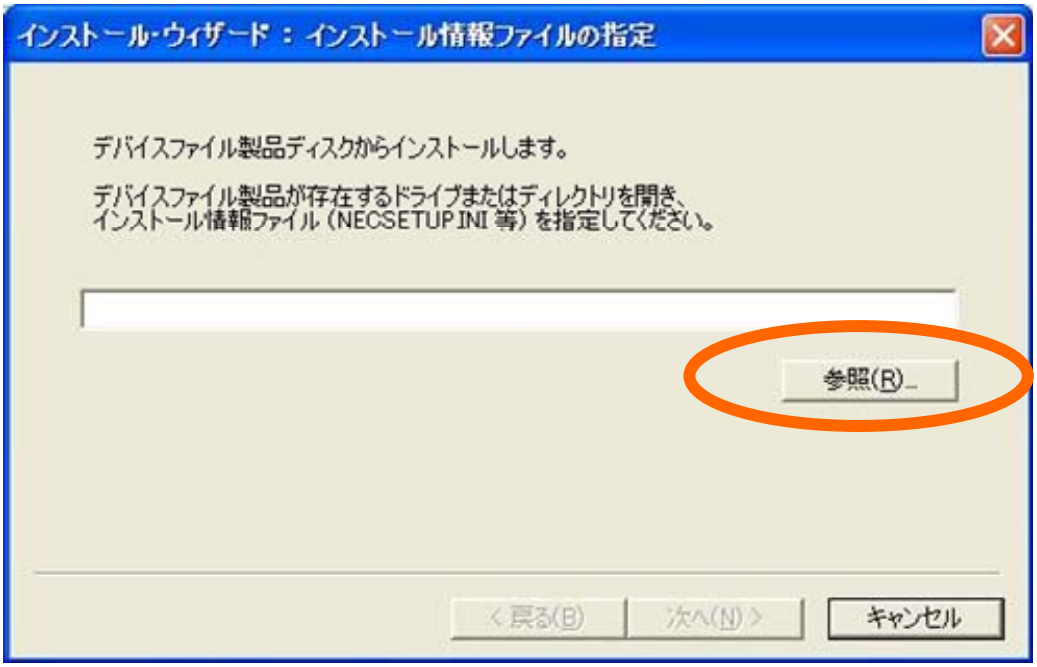
<2> 「プロジェクトの設定」ダイアログが開きます。「プロジェクト情報」タブの「デバイス登録」ボタンを押下してDevice File Installerを起動します。



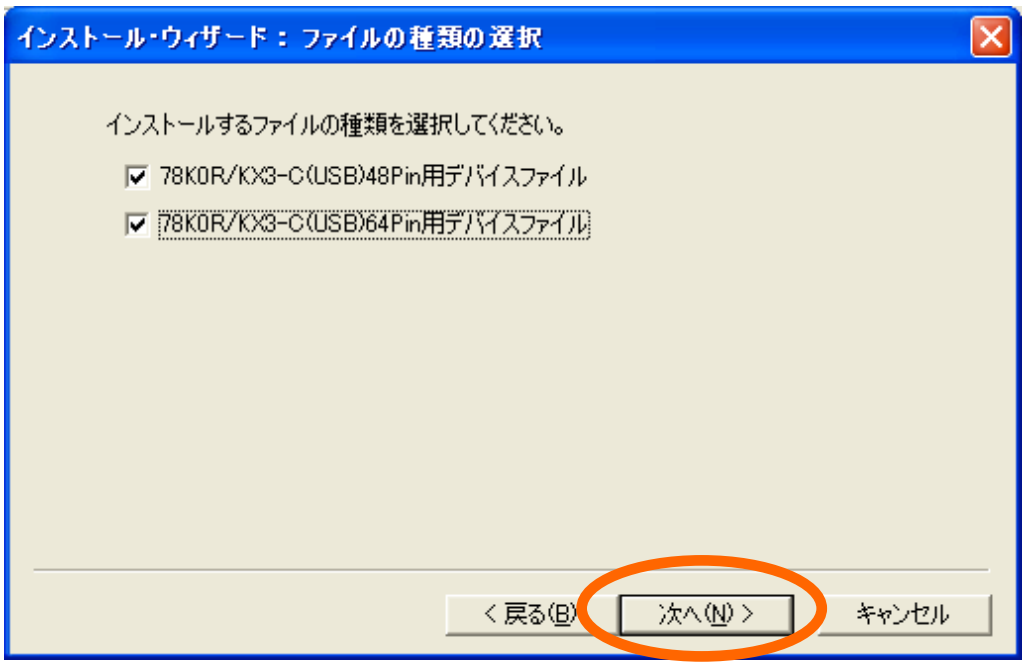
<3> 「Device File Installer」ダイアログが開きます。「インストール」ボタンを押下してインストール・ウィザードを起動します。



<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。

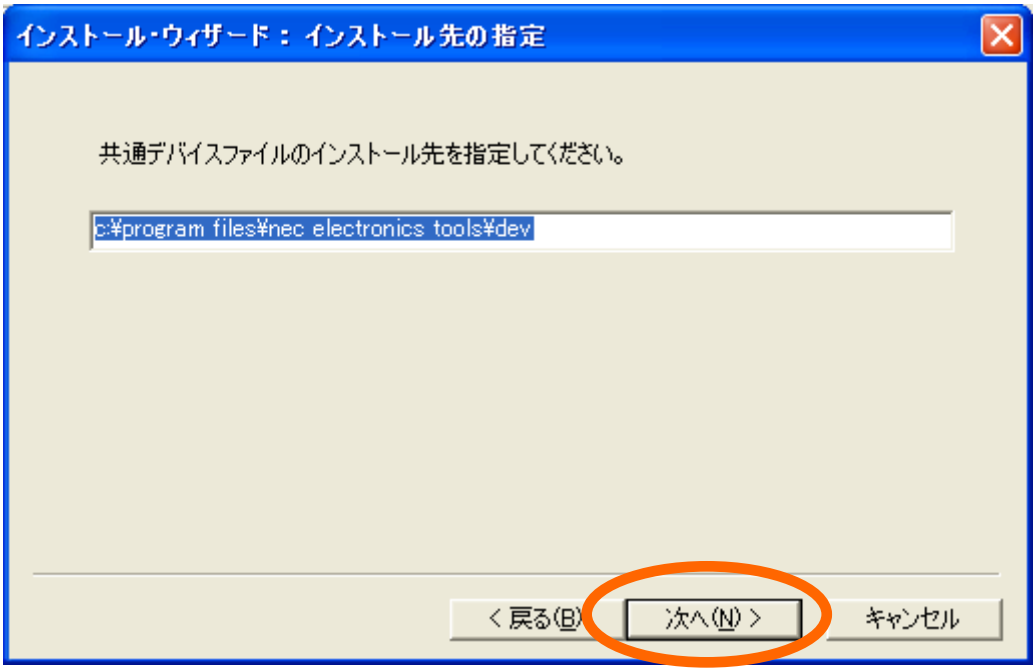


<5> 「ファイルを開く」ダイアログが表示されます。デバイス・ファイルを格納したディレクトリを開き，"NECSETUP.INI" ファイルを選択して「開く」ボタンを押下します。
<6> 使用許諾に関するメッセージが表示されます。「次へ」ボタンを押下します。
<7> 「ファイルの種類の選択」ダイアログが開きます。該当するデバイス・ファイルを選択して「次へ」ボタンを押下します。

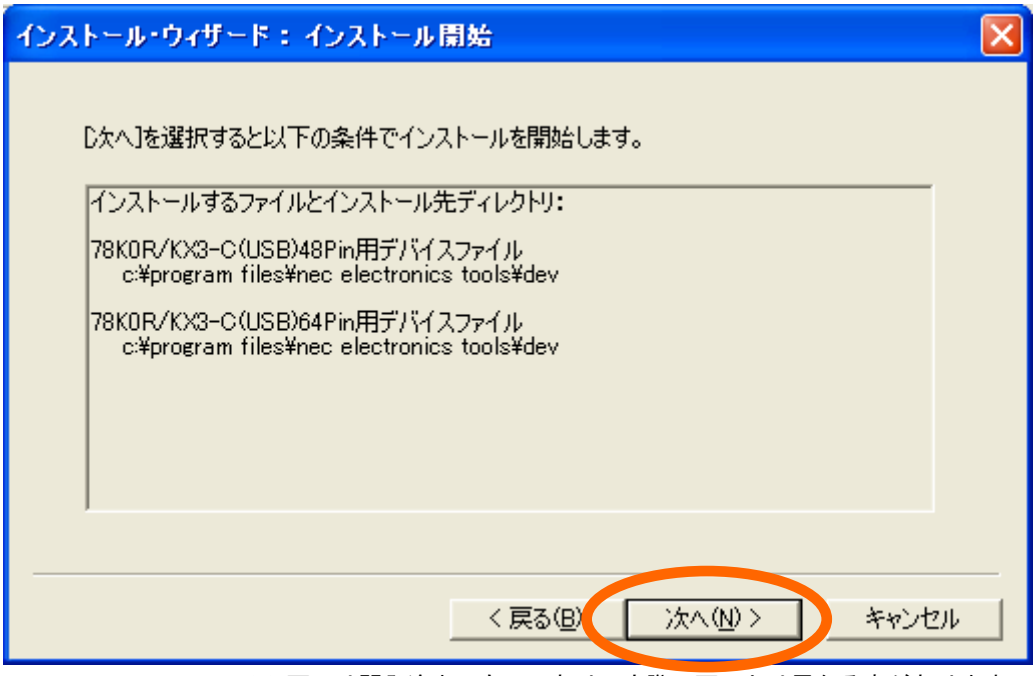


※ 画面は開発途中のものであり、実際の画面とは異なる事があります。

<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。

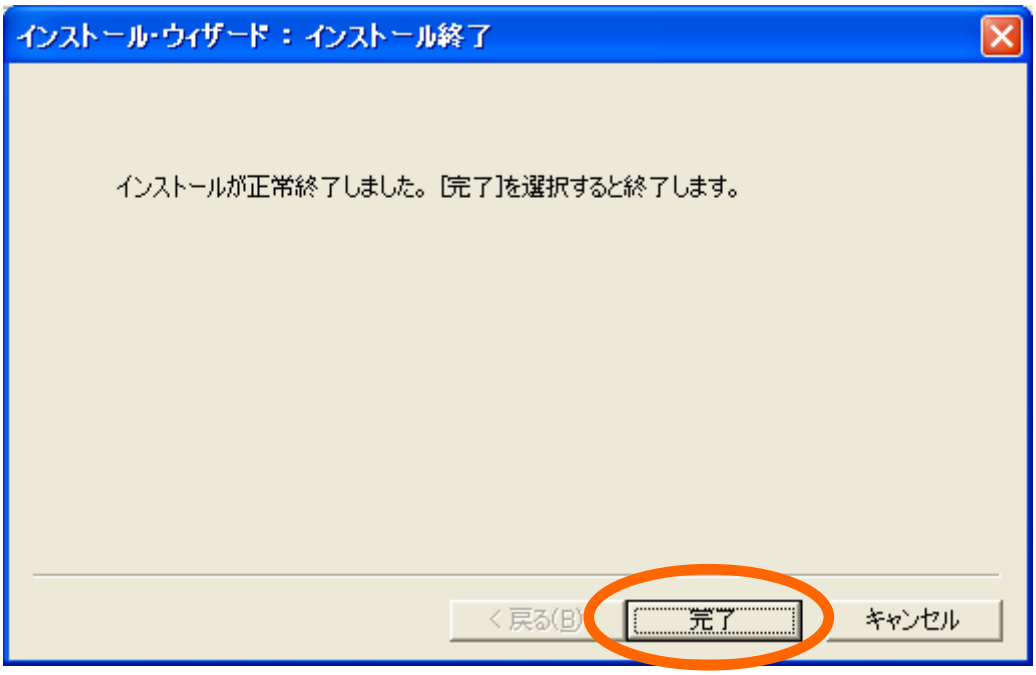


<9> 「インストール開始」ダイアログが開きます。「次へ」ボタンを押下します。



※ 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。

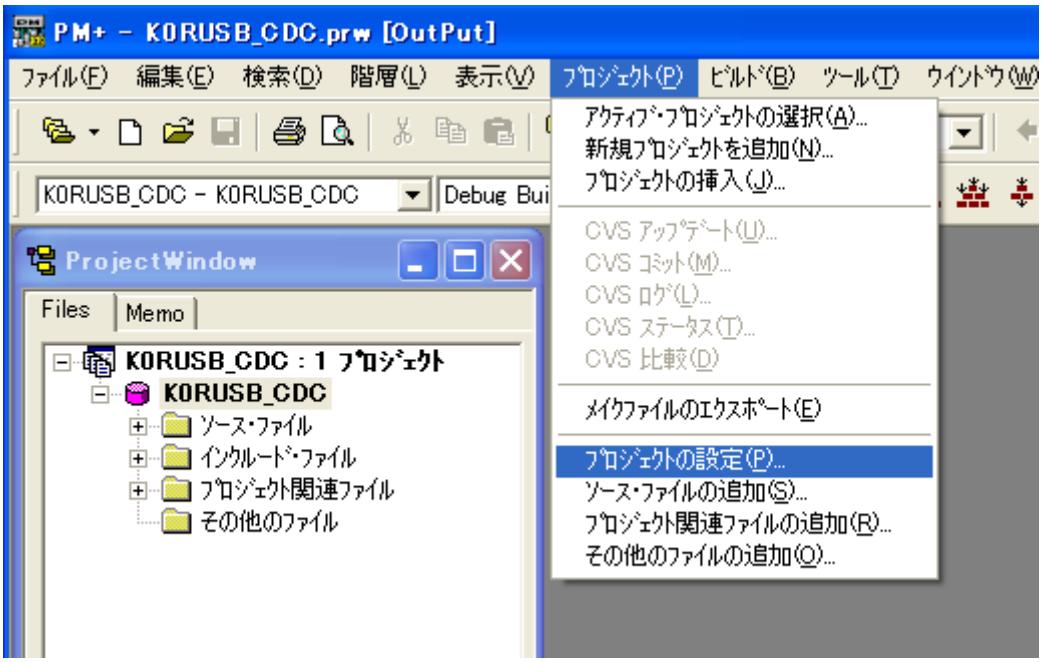
<10> デバイス・ファイルがプロジェクトにインストールされます。環境により，時間がかかる場合があります。
<11> インストールが終わると「インストール終了」ダイアログが開きます。「完了」ボタンを押下します。



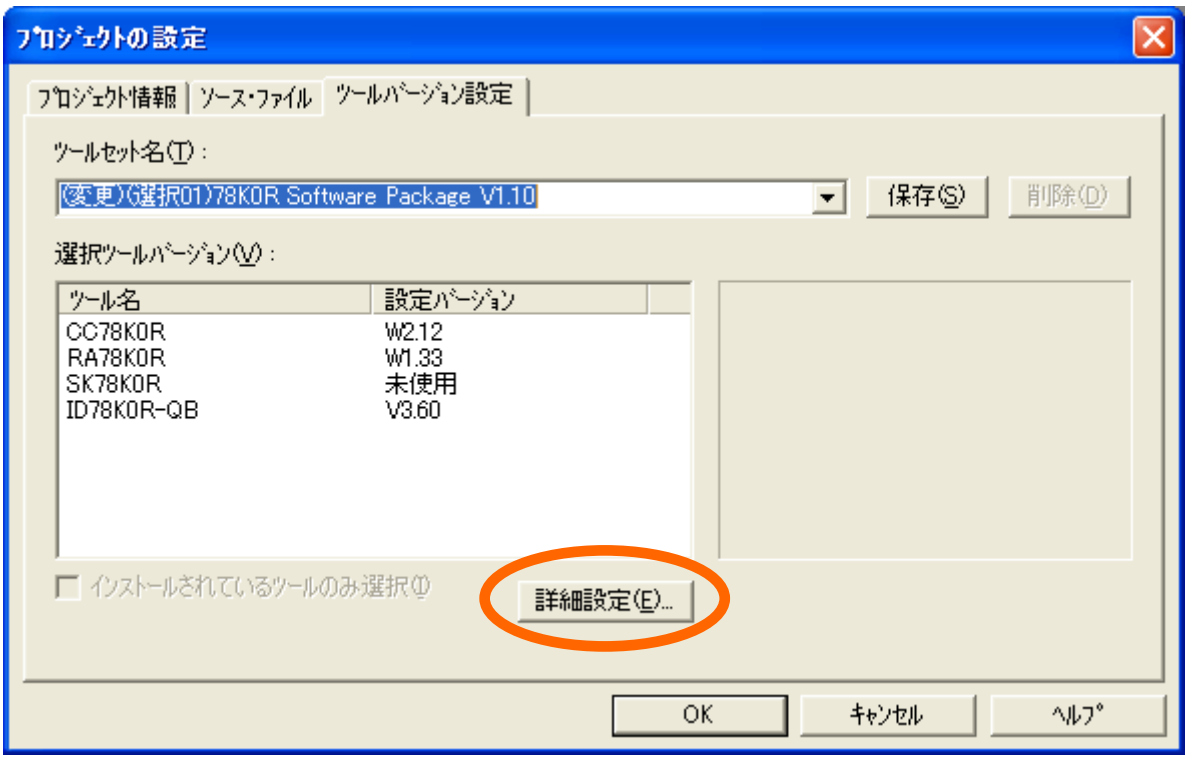
ビルド・ツールの設定

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。

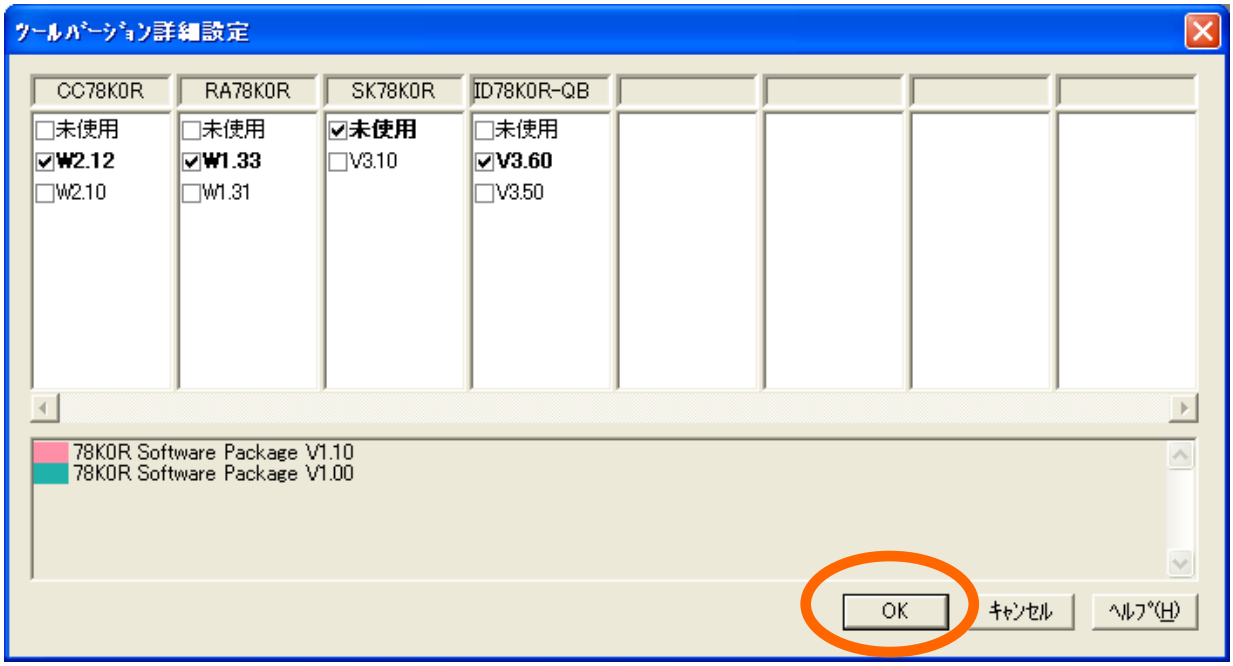
<1> PM+の「プロジェクト」メニューから「プロジェクトの設定」を選択します。



<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。



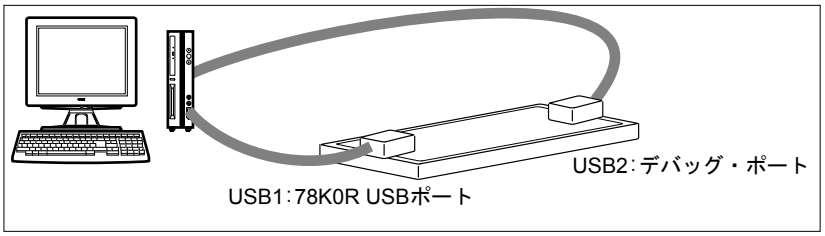
5. 2. 2　ターゲット環境整備

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



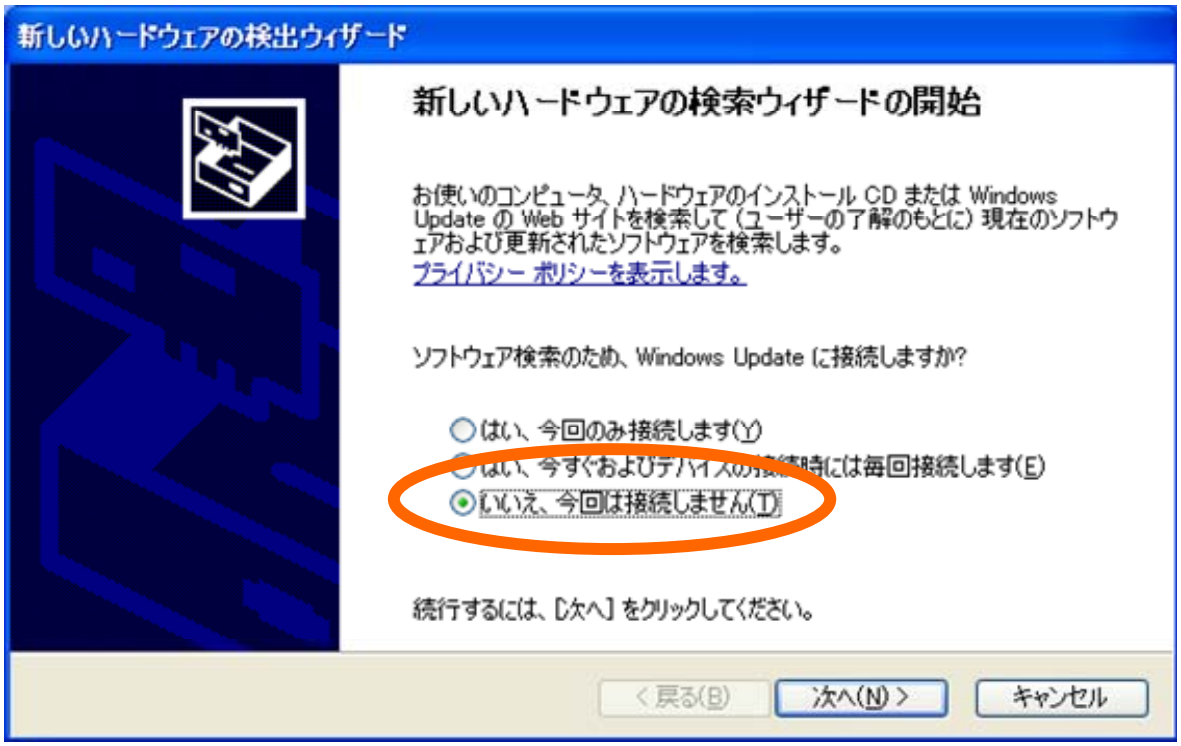
備考　TK-78K0R/KE3L+USBの外形やポートについては，**付録A　スタータ・キット**を参照してください。

ホスト・ドライバのインストール

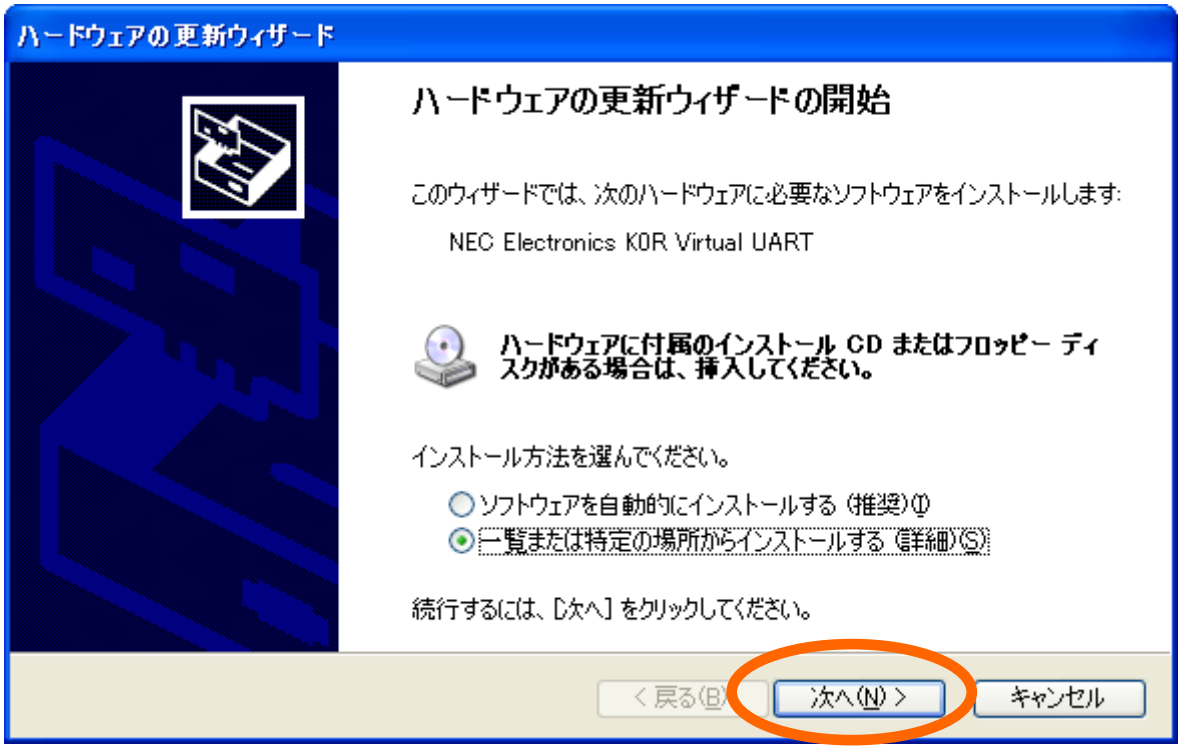
ここではサンプル・ドライバに同梱の仮想COMポート用ホスト・ドライバを使用する場合の手順を示します。

備考　78K0R/KE3-Lの他方のUSBポートはデバッグ・ポートで，別途専用のホスト・ドライバが必要になります。使用するファイルや入手方法については弊社までお問い合わせください。

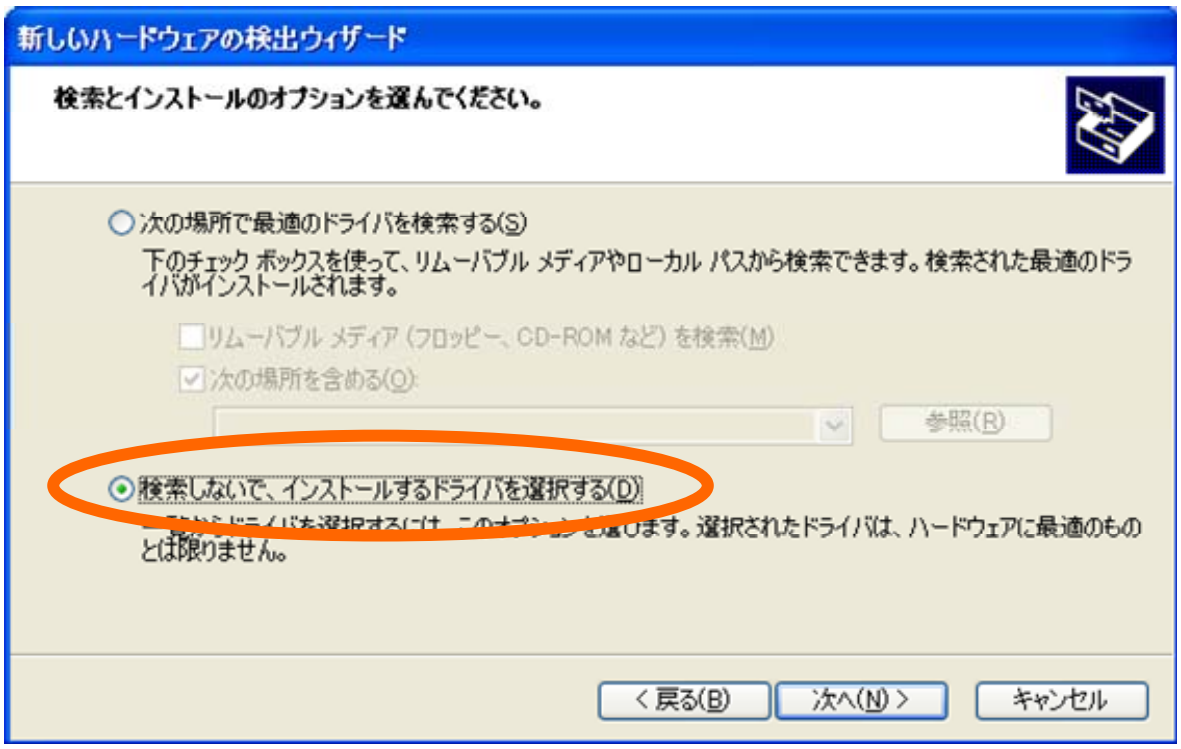
<1> TK-78K0R/KE3L+USBの接続がホスト・マシンに認識されると，「新しいハードウエアが見つかりました」というメッセージが表示されたあと，新しいハードウエアの検出ウィザードが起動します。
<2> 「新しいハードウエアの検出ウィザード」ダイアログが開きます。「いいえ、今回は接続しません」を選択して「次へ」ボタンを押下します。



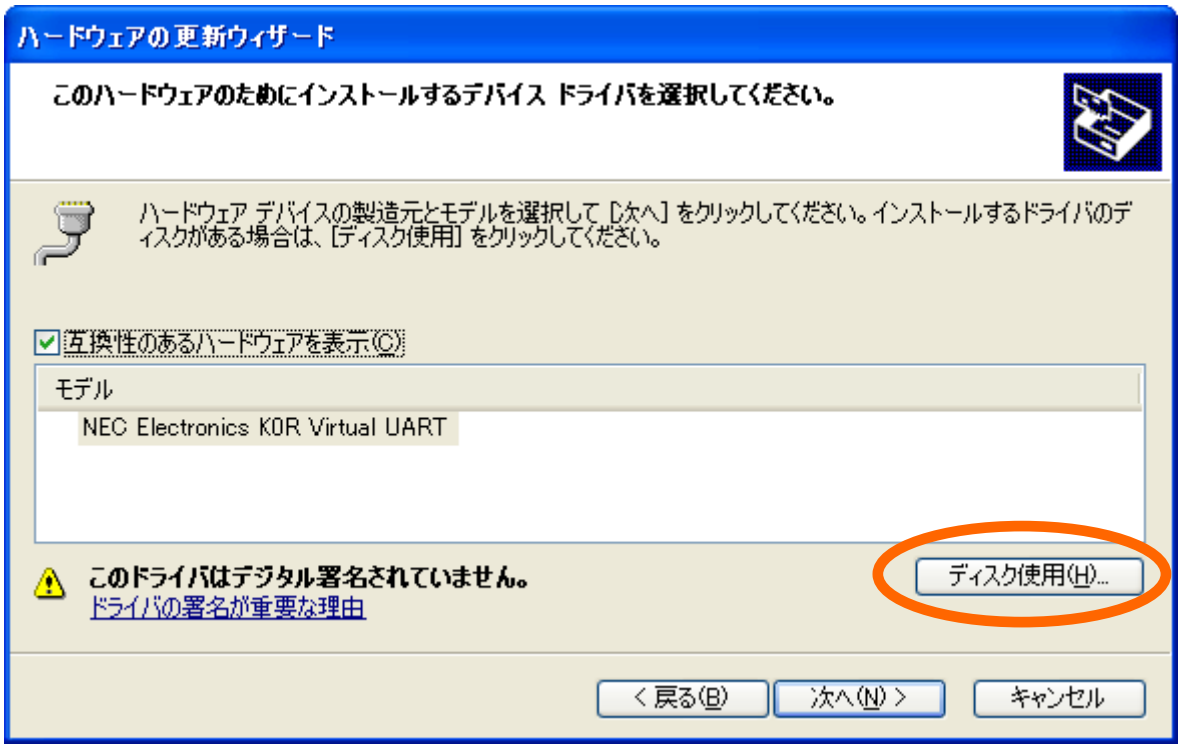
<3> 次の画面が表示されます。「一覧または特定の場所からインストールする（詳細）」を選択して「次へ」ボタンを押下します。



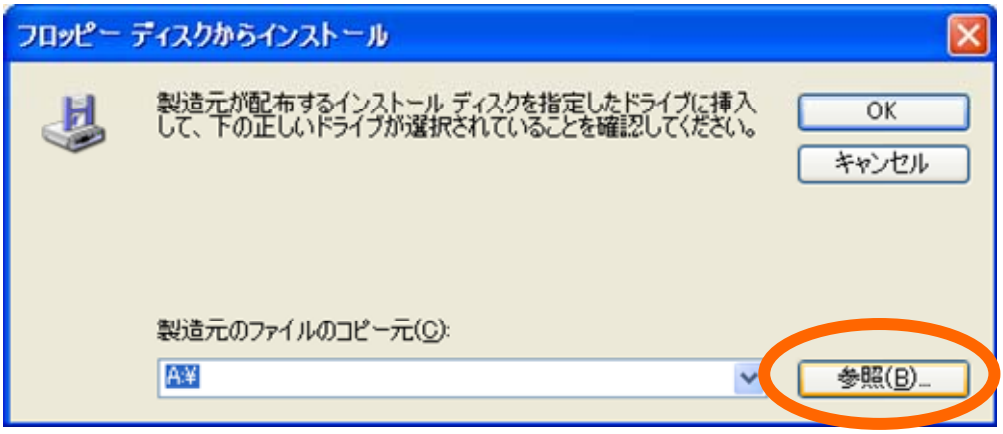
<4> 次の画面が表示されます。「検索しないで、インストールするドライバを選択する」を選択して「次へ」ボタンを押下します。



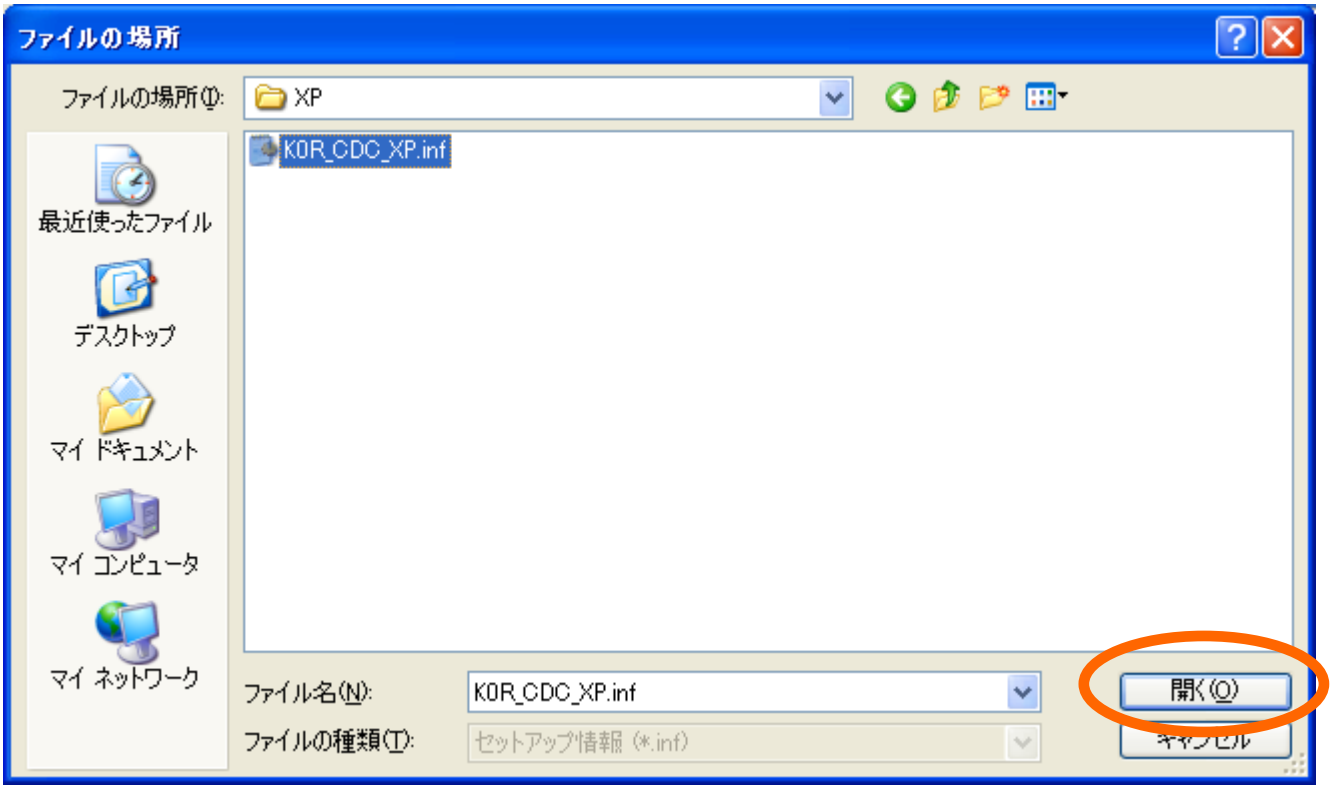
<5> 次の画面が表示されます。「ディスク使用」ボタンを押下します。



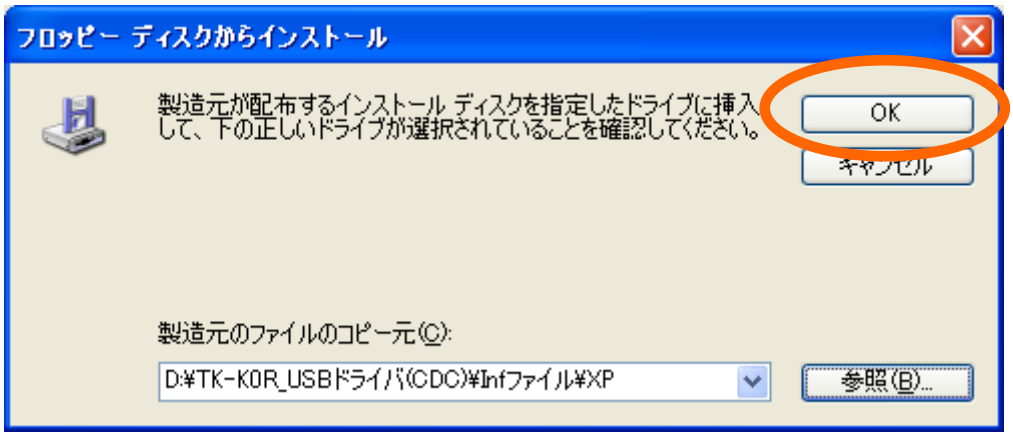
<6> 「フロッピーディスクからインストール」ダイアログが開きます。「参照」ボタンを押下して，サンプル・ドライバを格納したディレクトリの「Infファイル」フォルダを表示させます。



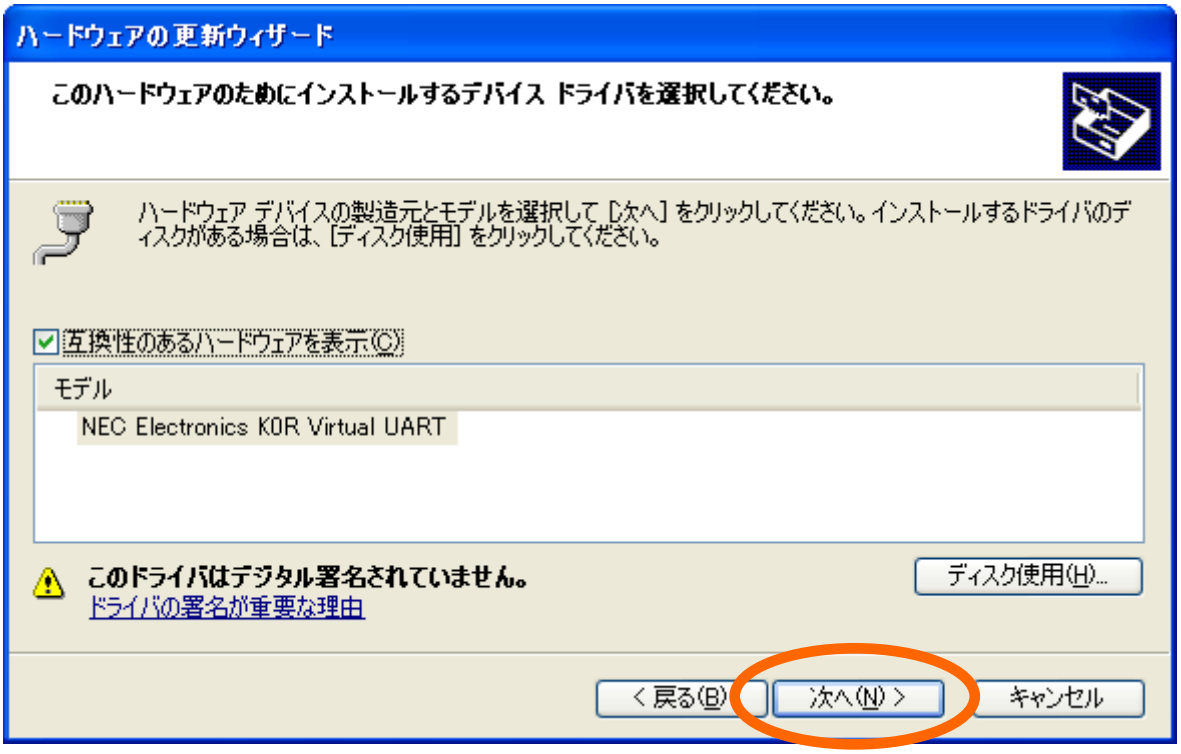
<7> ホスト・マシンで使用しているOSにあわせ「XP」フォルダの中のINFファイルを選択し，「開く」ボタンを押下します。



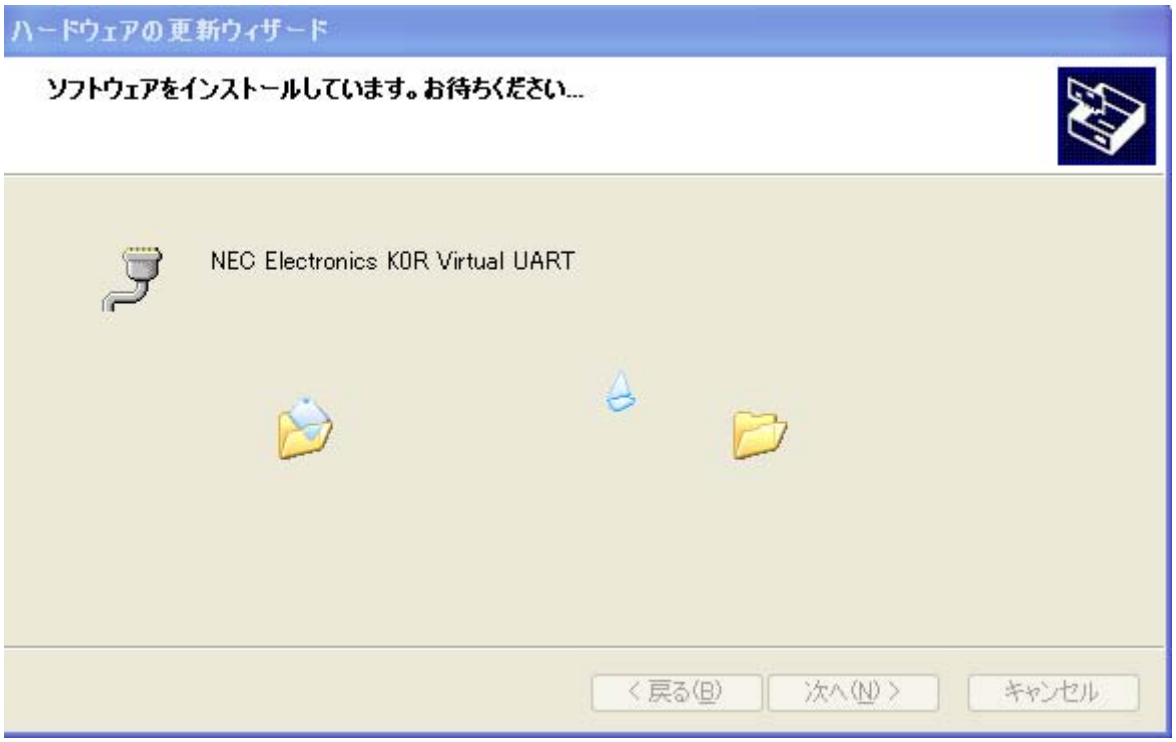
<8> 「フロッピーディスクからインストール」ダイアログに戻ります。「製造元のファイルのコピー元」欄が正しいことを確認し，「OK」ボタンを押下します。



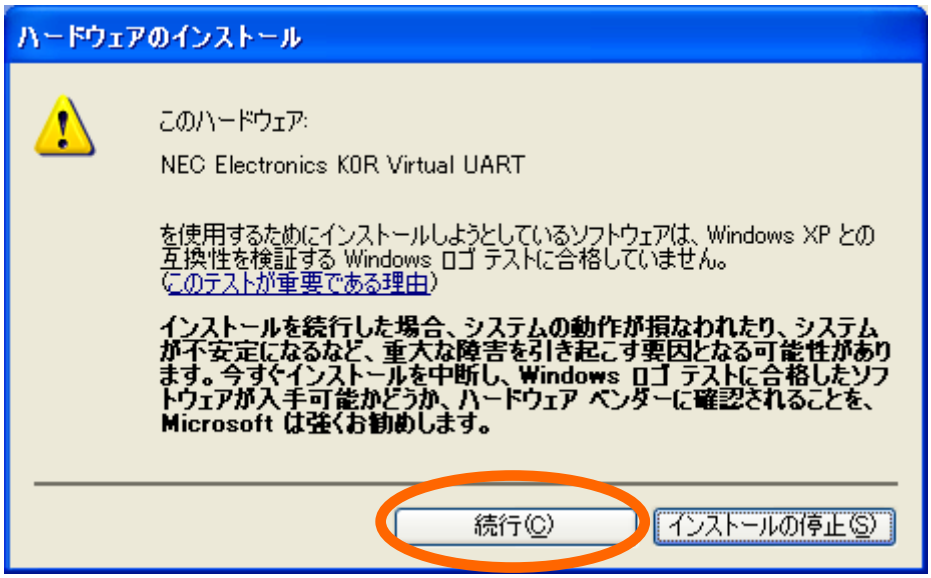
<9> 「新しいハードウエアの検出ウィザード」画面に戻ります。「モデル」欄で「NEC Electronics K0R Virtual UART」を選択して「次へ」ボタンを押下します。



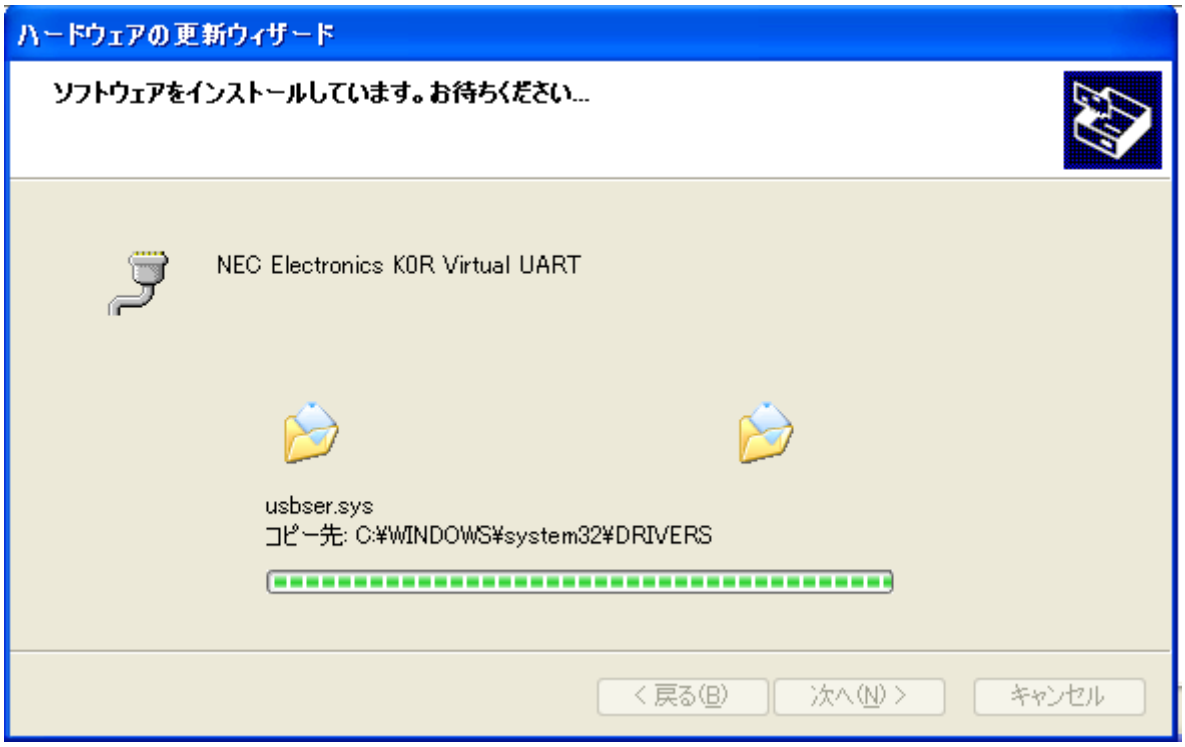
<10> ドライバのインストールが開始されます。



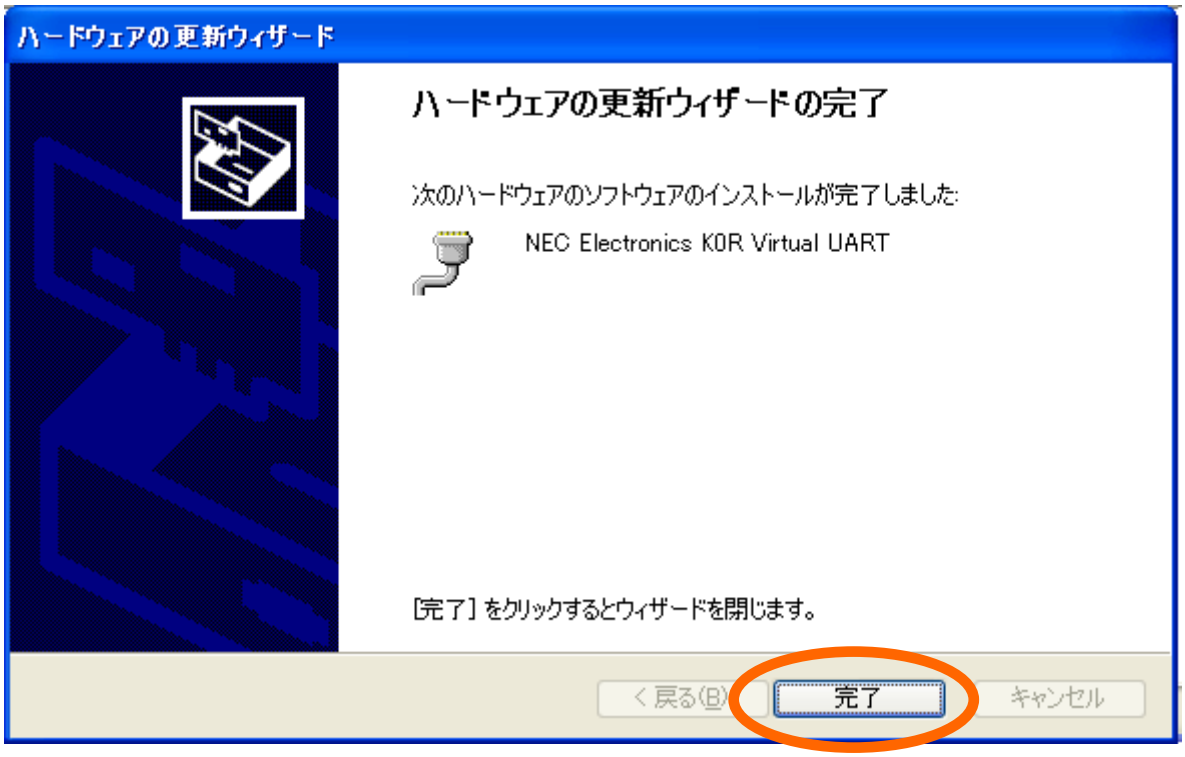
<11> 「ハードウエアのインストール」ダイアログが表示されます。「続行」を押下します。



<12> ドライバが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。

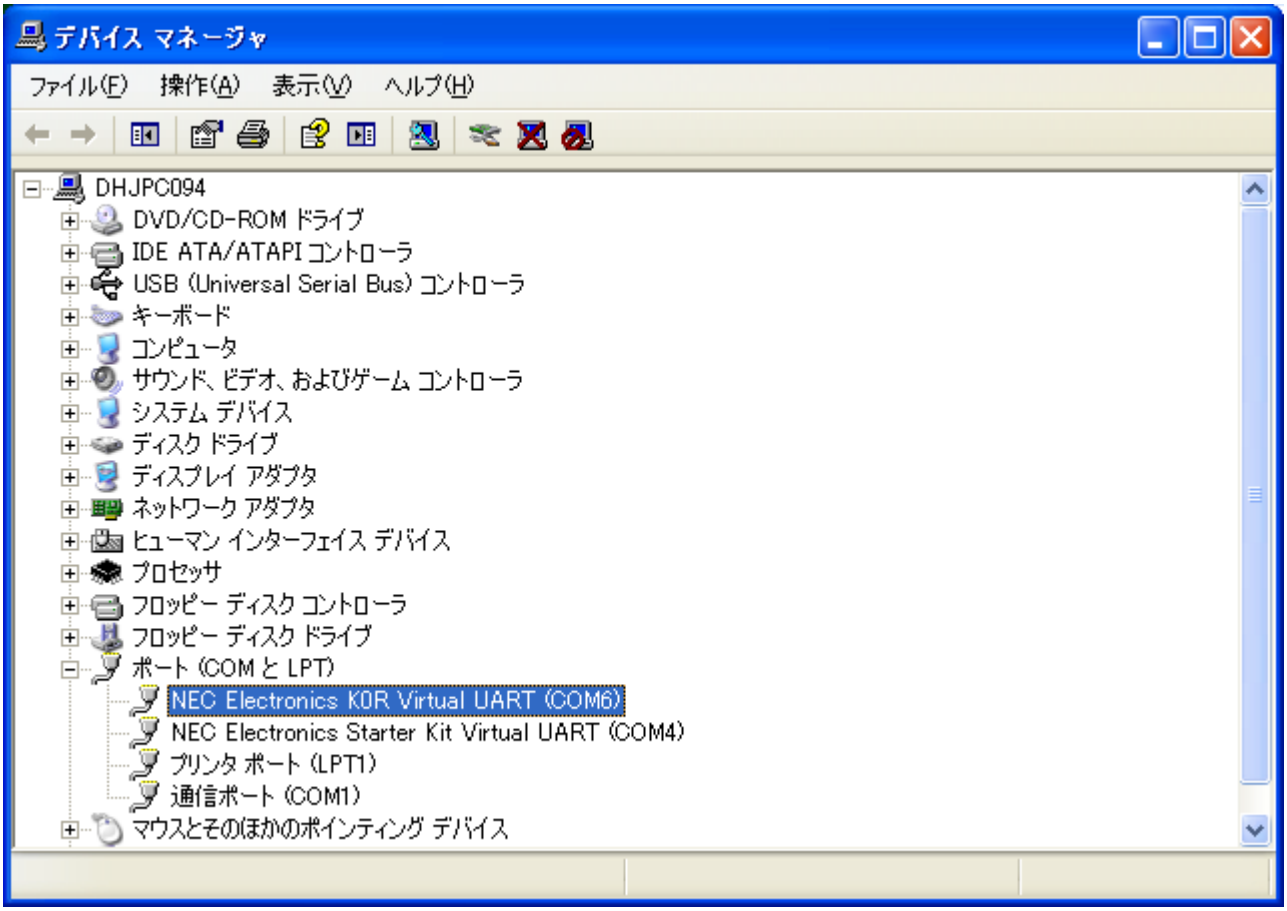


<13> 次の画面が表示されます。「完了」ボタンを押下します。



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。デバイスの一覧表示の「ポート」のツリーを展開し，「NEC Electronics K0R Virtual UART」が表示されていること，また割り当てられたCOMポート番号を確認します。



備考　デバイス名やポート番号は任意のものに変更できます。詳細は**6. 2 カスタマイズ**を参照してください。

5. 3　オンチップ・デバッグ

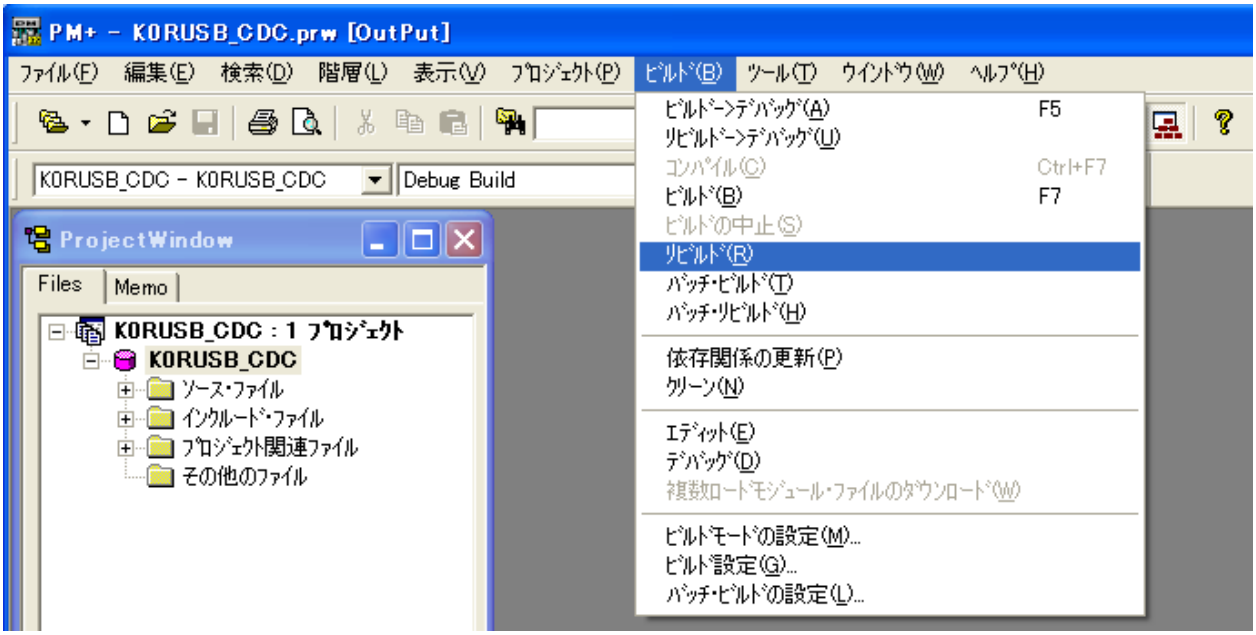
ここでは，5.2環境設定に示したワークスペースで開発したアプリケーション・プログラムのデバッグ手順について説明します。

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。



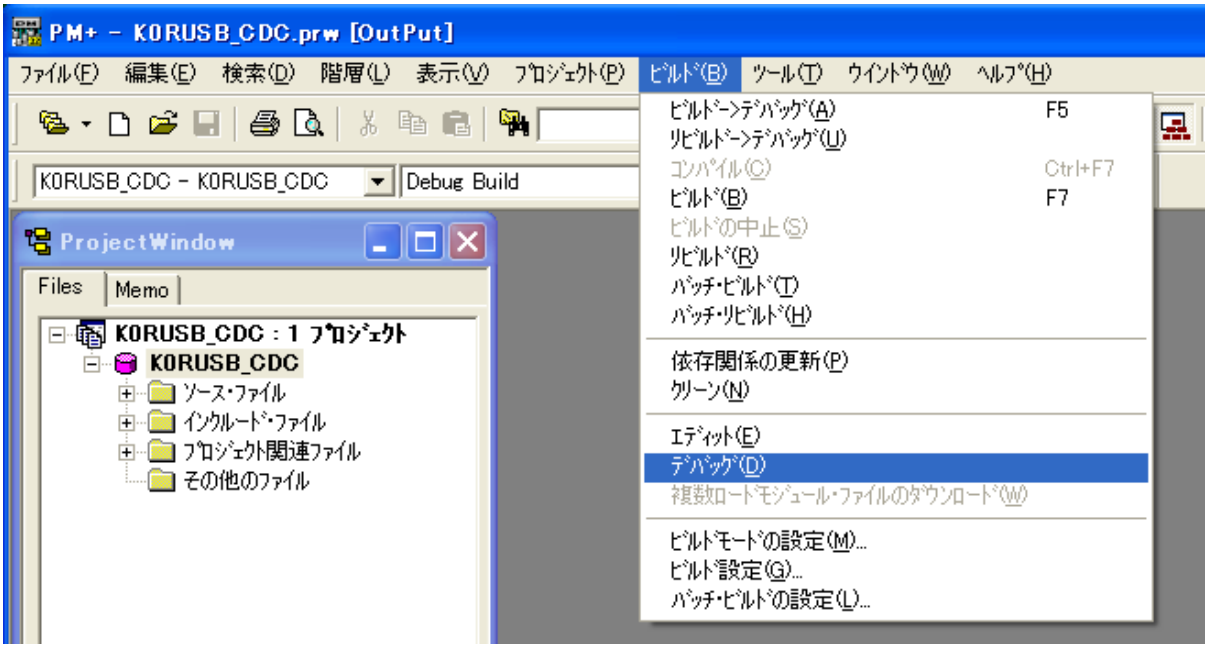
5. 3. 2　ロードと実行

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。

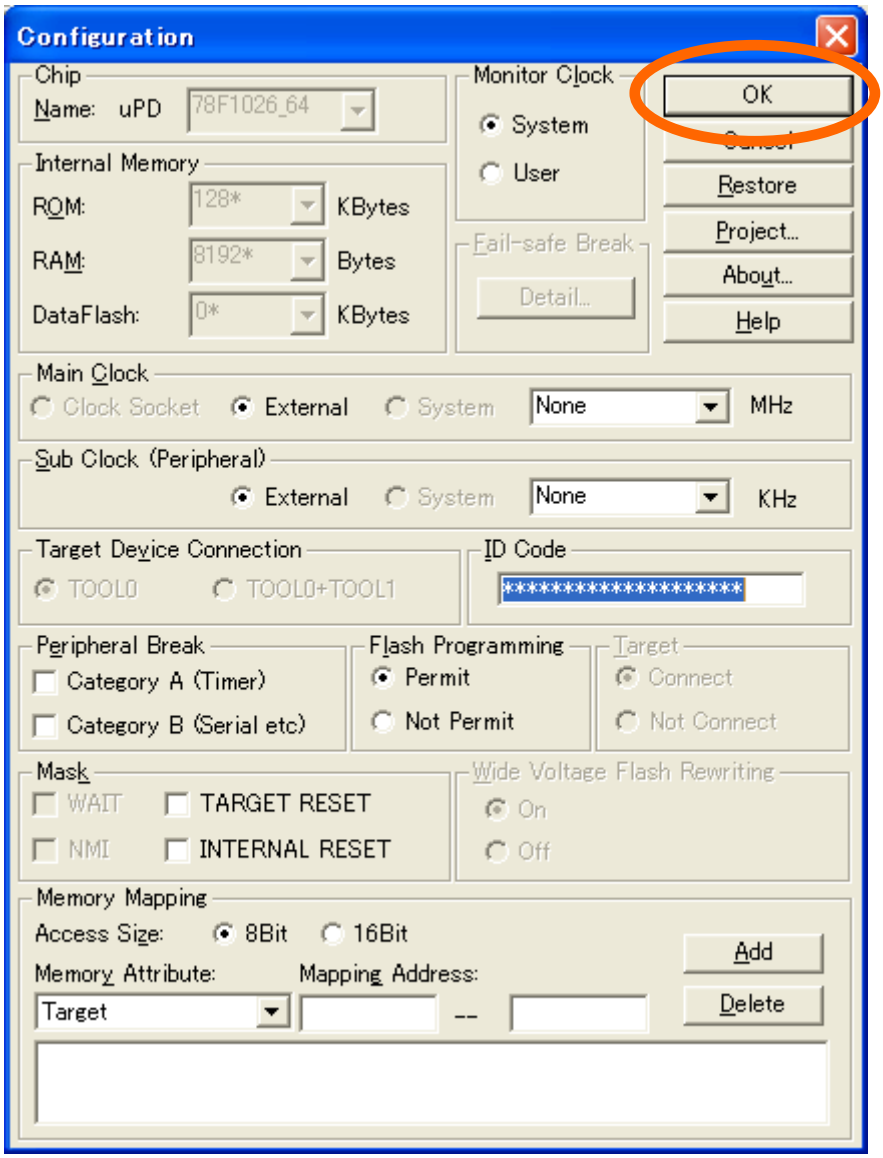
(2)　ロード・モジュールの書き込み

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。

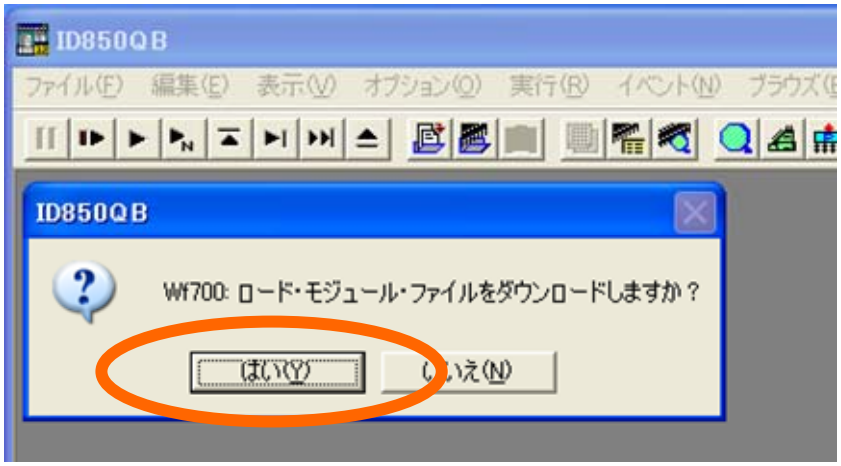
<1> 「ビルド」メニューから「デバッグ」を選択してID78K0R-QBを起動します。



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

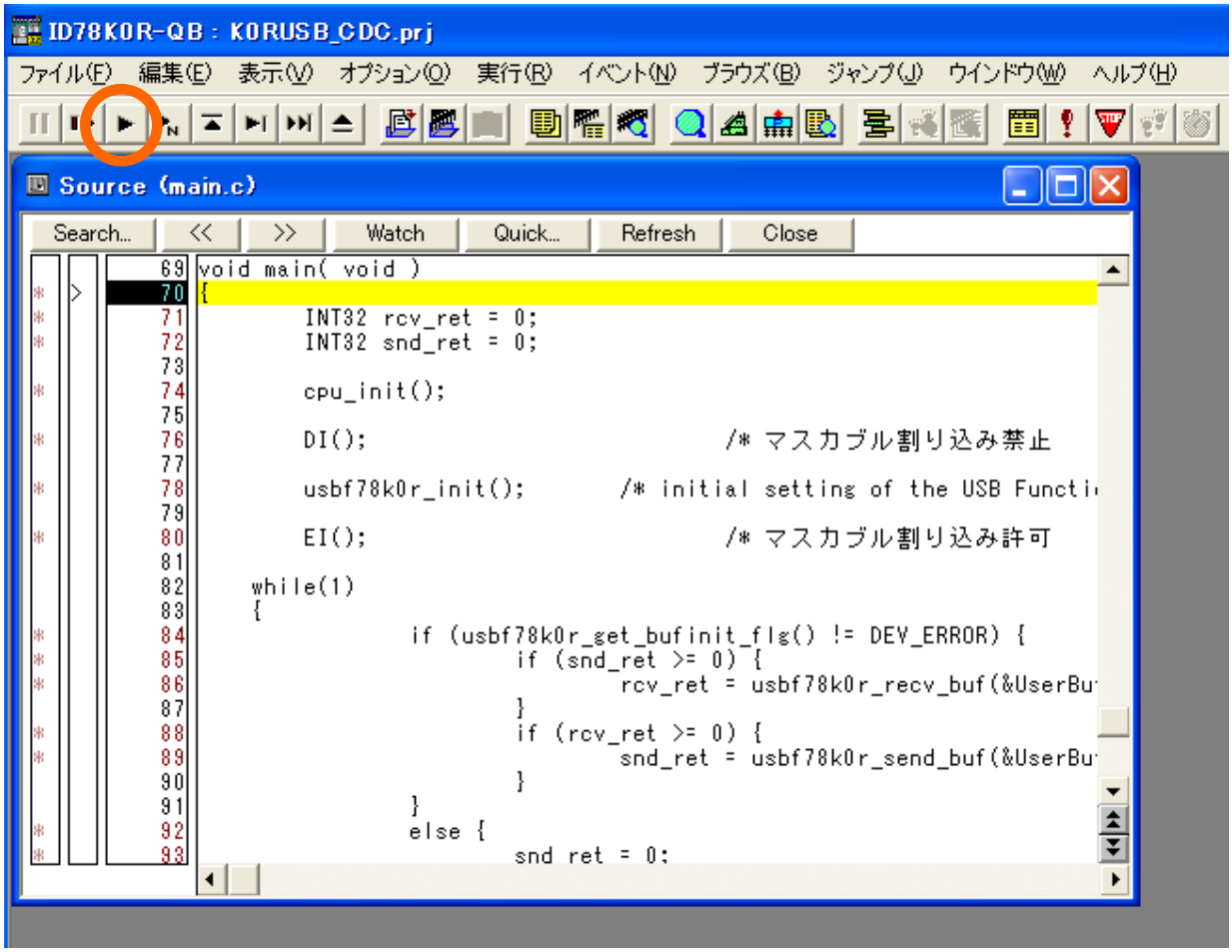


<3> サンプル・ドライバに同梱のプロジェクト・ファイルを使用している場合は，次の画面が表示されます。「はい」ボタンを押下してロード・モジュール・ファイルの書き込みを開始させます。



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。

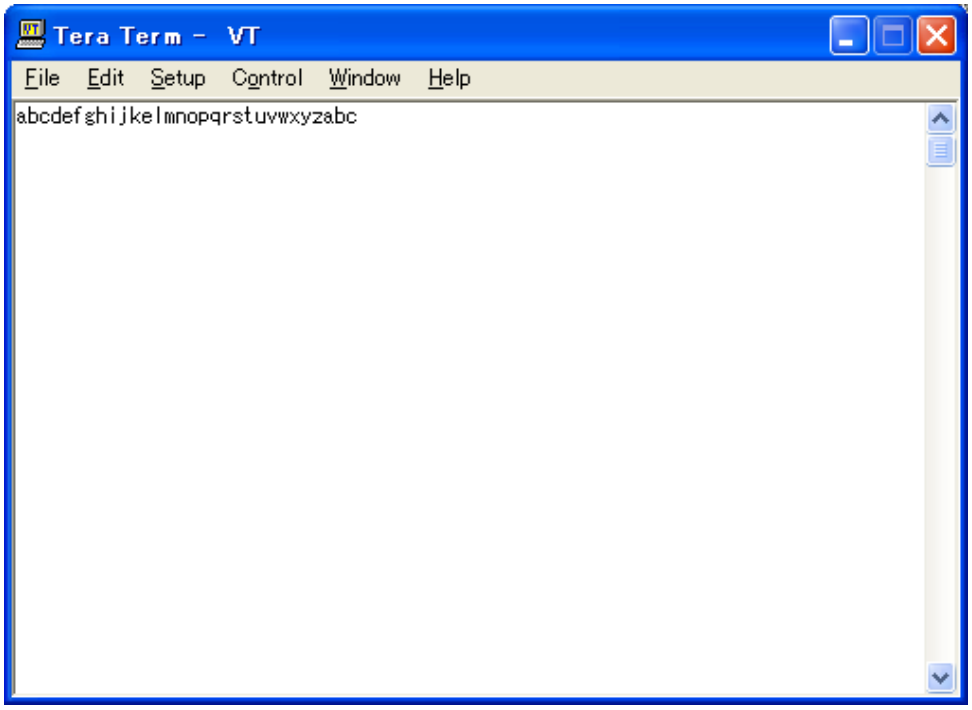


5. 4　動作確認

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。

ホスト上でターミナル・ソフトウエア（Tera Termなど）を起動し，動作を確認してください。キーボードから入力したキーデータをターミナル・ソフトウエア上で表示します。

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コミュニケーション・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
- 仮想 COM ポート用ホスト・ドライバ（INF ファイル）内のデバイス名やプロバイダ情報

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

```
/*============================================================
 Main function
 void main(void)

 Arguments:
      N/A
 Return values:
      N/A
 Overview:
      main routine.
============================================================*/
void main(void)
{
      INT32 rcv_ret = 0;
      INT32 snd_ret = 0;

      cpu_init();

      DI();

      usbf78k0r_init();   /* initial setting of the USB Function */

      EI();

      while(1)
      {
            if (usbf78k0r_get_bufinit_flg() != DEV_ERROR) {
                  if (snd_ret >= 0) {
                        rcv_ret = usbf78k0r_recv_buf(&UserBuf[0], USERBUF_SIZE);
                  }
                  if (rcv_ret >= 0) {
                        snd_ret = usbf78k0r_send_buf(&UserBuf[0], rcv_ret);
                  }
            }
            else {
                  snd_ret = 0;
                  rcv_ret = 0;
            }
      }
}
```

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。

なお，デバイス・ディスクリプタのベンダIDやプロダクトIDを書き換えた場合は，ターゲット・デバイス接続の際にインストールするホスト・ドライバ（INFファイル）でも同じように書き換える必要があります（6. 2. 4（3）ベンダIDとプロダクトIDの変更参照）。

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。

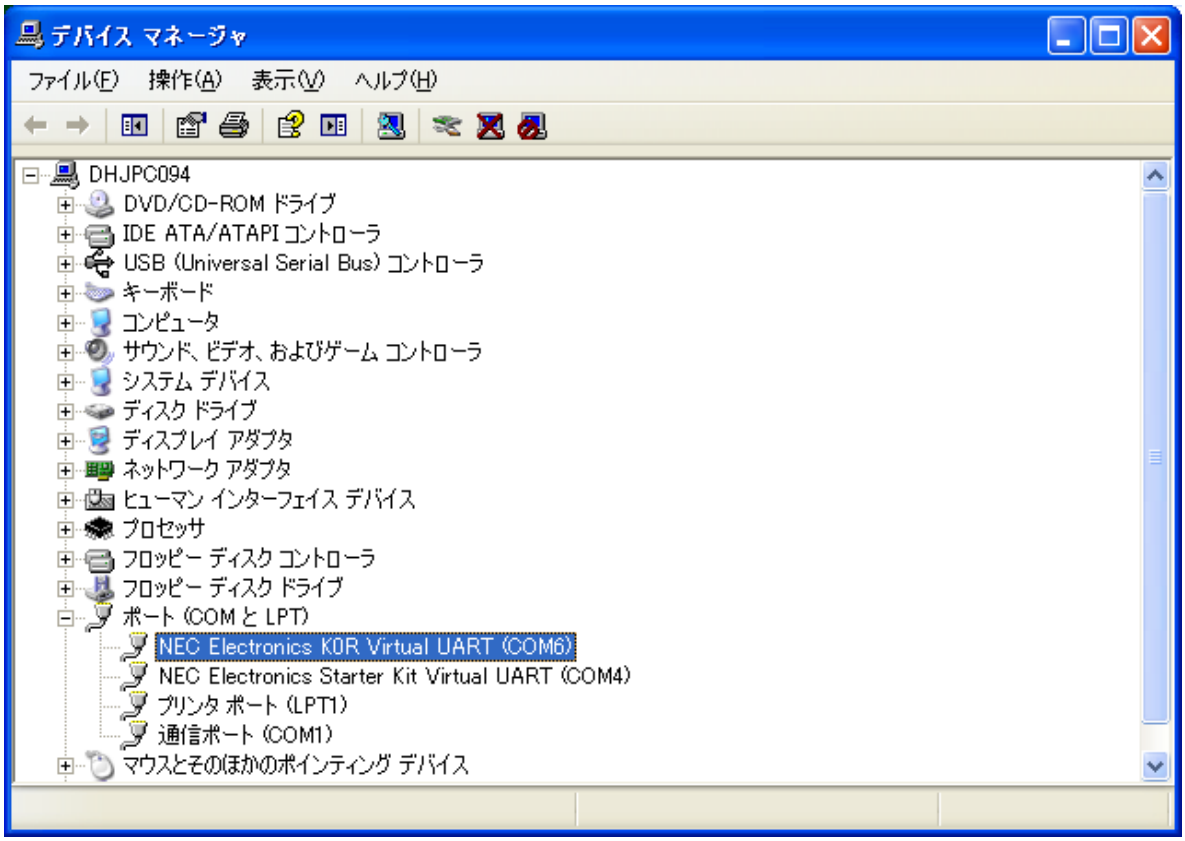
6. 2. 4　仮想COMポート用ホスト・ドライバの設定

5. 2. 2 ターゲット環境整備でインストールしたドライバに関連して，次のようなカスタマイズが可能です。

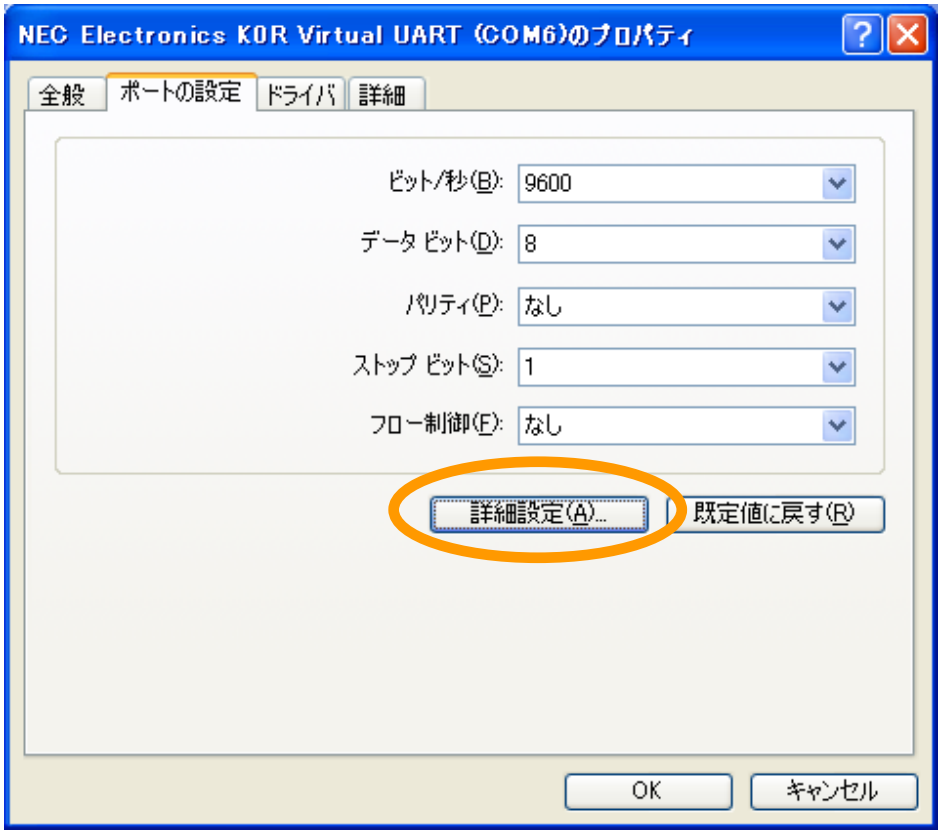
COMポート番号の変更

USBデバイスの接続を認識すると，そのデバイスのCOMポート番号をホストが自動的に割り付けますが，任意の番号に変更することもできます。ホスト・マシンでCOMポート番号を変更する手順は次のとおりです。

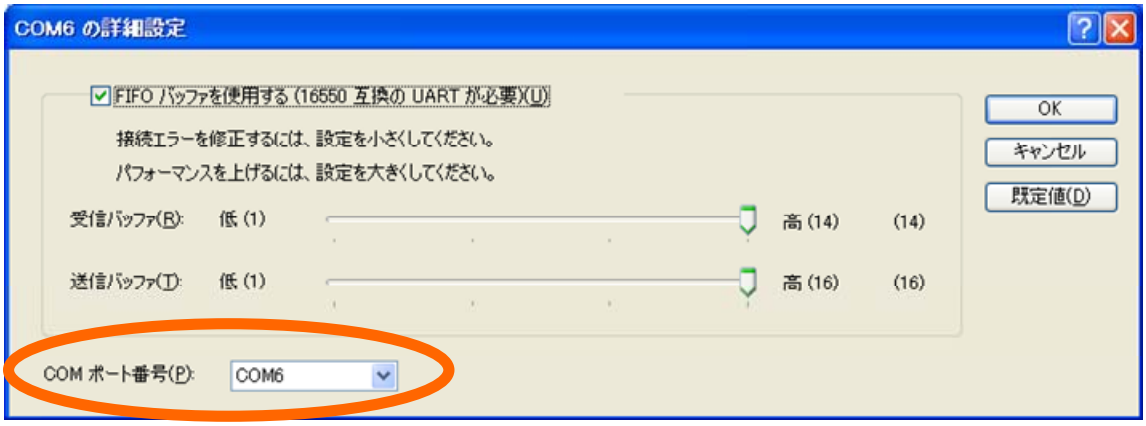
<1> Windowsのデバイスマネージャを開き，デバイスの一覧表示の「ポート」のツリーを展開します。



<2> 「NEC Electronics K0R Virtual UART (COMn)」（nはホストが割り付けた番号）を選択してプロパティを表示します。
<3> 「ポートの設定」タブの「詳細設定」ボタンを押下します。



<4> 「COMnの詳細設定」ダイアログ（nはホストが割り付けた番号）が開きます。「COMポート番号」欄のドロップダウン・リストから任意のポート番号を選択します。

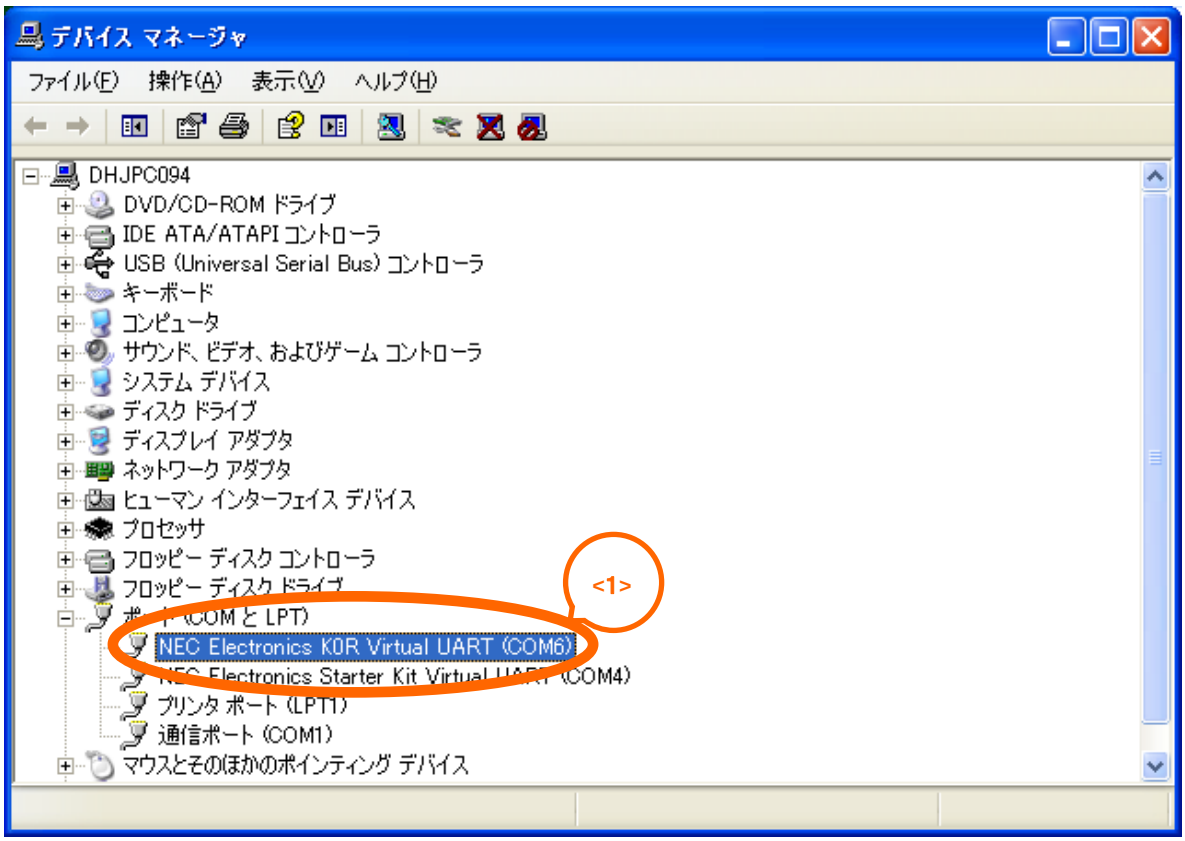


備考1. ほかのデバイスで使用するポート番号と重ならないようにしてください。
2. この変更後はすぐに新しいポート番号が有効になりますが，デバイスマネージャの一覧表示にはすぐに反映されないことがあります。

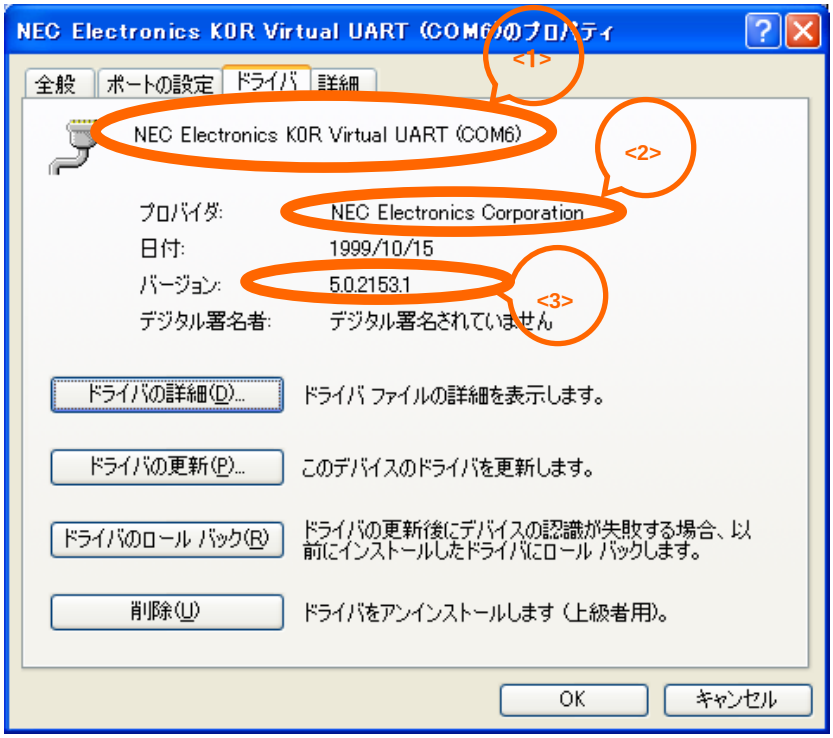
プロパティの変更

Windowsのデバイスマネージャで使用されるデバイスの属性などの情報は，適宜，変更する事ができます。

(a) デバイス名（デバイス一覧）



（b）デバイス名，製造元名，バージョン（デバイスのプロパティ）



これらは，ホスト・ドライバ（INFファイル）に記述されている情報を元に表示されるため，INFファイルを書き換えることで変更できます。INFファイル内で前述の例の番号に対応する部分は次のとおりです。

リスト 6－2　INFファイル "K0R_CDC_XP.inf" の記述

```
1    ; .inf file (Win2000,XP):
2    [Version]
3    Signature="$Windows NT$"
4    Class=Ports
5    ClassGuid={4D36E978-E325-11CE-BFC1-08002BE10318}
6
7    Provider=%NEC%
8    LayoutFile=layout.inf
9    DriverVer=10/15/1999,5.0.2153.1                              <3>
10
11   [Manufacturer]
12   %NEC%=NEC
13
14   [NEC]
15   %NEC78K0RKx3L%=Reader, USB¥VID_0409&PID_01D0
16
17   [Reader_Install.NTx86]
18   ;Windows2000
19
20   [DestinationDirs]
21   DefaultDestDir=12
22   Reader.NT.Copy=12
23
24   [Reader.NT]
25   CopyFiles=Reader.NT.Copy
26   AddReg=Reader.NT.AddReg
27
28   [Reader.NT.Copy]
29   usbser.sys
30
31   [Reader.NT.AddReg]
32   HKR,,DevLoader,,*ntkern
33   HKR,,NTMPDriver,,usbser.sys
34   HKR,,EnumPropPages32,,"MsPorts.dll,SerialPortPropPageProvider"
35
36   [Reader.NT.Services]
37   AddService = usbser, 0x00000002, Service_Inst
38
39   [Service_Inst]
40   DisplayName = %Serial.SvcDesc%
41   ServiceType = 1 ; SERVICE_KERNEL_DRIVER
42   StartType = 3 ; SERVICE_DEMAND_START
43   ErrorControl = 1 ; SERVICE_ERROR_NORMAL
44   ServiceBinary = %12%¥usbser.sys
45   LoadOrderGroup = Base
46
47   [Strings]
48   NEC = "NEC Electronics Corporation"                          <2>
49   NEC78K0RKx3L = "NEC Electronics K0R Virtual UART"            <1>
50   Serial.SvcDesc = "USB Serial emulation driver"
```

ベンダIDとプロダクトIDの変更

デバイス・ディスクリプタ内のベンダIDとプロダクトIDを変更した場合は，ホスト・ドライバ（INFファイル）にも同じ内容を設定する必要があります。

INFファイルでは，リスト 6－2の15行目に次のような形式でベンダIDとプロダクトIDを記述してください。

ベンダID： "VID_" に続けて4桁の16進数で表記
プロダクトID： "PID_" に続けて4桁の16進数で表記

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。リストに示したサンプル・アプリケーションの次に示す部分は、定義済みの各種処理の応用例として再利用可能です。

(1) ユーザ・データ用 FIFO 状態確認

27行目では、ユーザ・データ用FIFOの状態通知処理関数（usbf78k0r_get_bufinit_flg）を呼び出し、ユーザ・データ用FIFO初期化フラグ“usbf78k0r_bufinit_flg”を監視しています。このフラグはサンプル・ドライバで独自に定義されているフラグで、サンプル・ドライバのINTUSB割り込みで通知されるBus Reset 処理、および、クラス・リクエストのSet Line Codingリクエスト処理でFIFOの初期化が実行されるとセット(1)されます。

サンプル・アプリケーションでは、このFIFOの初期化を契機に、ユーザ・データの送受信処理のエラー状態をクリア(0)します。

(2) ユーザ・データの受信処理

サンプル・ドライバでは、受信データの取り込みをデータサイズの取得とデータのコピーの2段に分けて、それぞれの処理を定義した関数を用意しています。

実際に受信したサイズを元に受信処理を実行する場合は、受信データサイズの取得処理関数（usbf78k0r_rdata_length）を呼び出すことにより、受信処理の前に受信データサイズを確認することが出来ます。また、ユーザ・データ用バッファのサイズが決まっている場合は、バッファのサイズを基準に受信処理を呼び出すことも出来ます。ただし、1回の受信処理で処理できる最大のデータサイズは、1パケットで受信されるデータサイズ以下であることに注意してください。

サンプル・アプリケーションでは、バッファサイズが決まっている場合の使用例になっており、29行目のユーザ・データ受信処理関数（usbf78k0r_recv_buf）では、受信データがある場合、使用するエンドポイントから受信したデータを読み出します。

(3) ユーザ・データの送信処理

32行目のユーザ・データ送信処理関数（usbf78k0r_send_buf）では、送信データがある場合、使用するエンドポイントのFIFOの状態を確認し、FIFO Emptyであればデータの書き込みを実行します。FIFO Fullの場合、エラー終了します。また、送信データが無く、前に送信されたパケットのデータサイズがMax Packet Sizeに等しい場合、NULLパケットの送信処理を実行します。これは、コミュニケーション･デバイス･クラス固有の決まりで､データの最終パケットがMax Packet Sizeに等しい場合、最終データであることをホスト側に通知する目的でNULLパケットを送信することになっているためです。

サンプル・アプリケーションでは、送信処理がエラー終了した場合、送信待ちデータの送信FIFOへの書き込みが正常に終了するまで､受信処理を中止し送信処理を繰り返し実行します｡唯一の例外は､ユーザ・データ用FIFOの初期化です。ユーザ、または、ホストからのリクエストによりFIFOが初期化された場合、FIFOにある送受信データ、および、送信待ちのデータは破棄されます。

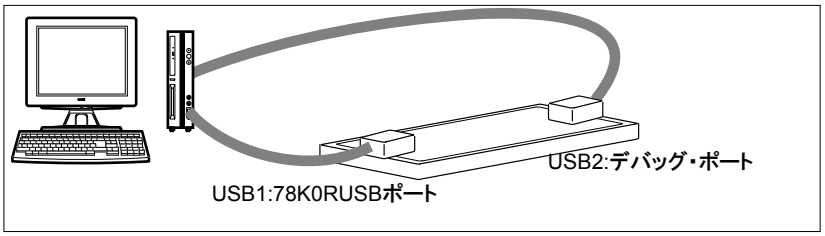
第7章 スタータ・キット

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 7－1　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 (http://www.necel.co.jp/)
【問い合わせ先】http://www.necel.com/contact/ja/