RXファミリ

R01AN2238JJ0140
Rev.1.40
Jun 30, 2022

USB ペリフェラルコミュニケーションデバイスクラスドライバ(PCDC)による USB ホストとの USB 通信を行うサンプルプログラム Firmware Integration Technology

要旨

本資料は、USB Peripheral Communications Devices Class Driver Firmware Integration Techology を使用したサンプルファームウェアの説明資料です。以降、本サンプルファームウェアを PCDC と記述します。

実際のソフトウェア開発時には、必ず” USB Basic Host and Peripheral Driver Firmware Integration Technology アプリケーションノート”(Document No:R01AN2025)および各マイコンのユーザーズマニュアル(ハードウェア編)と併用してご利用ください。また、必要に応じて USB Peripheral Communication Device Class Driver (PCDC) Firmware Integration Technology アプリケーションノート”(Document No:R01AN2030)も参照してください。なお、USB Basic Host and Peripheral Driver Firmware Integration Technology アプリケーションノート”(Document No:R01AN2025)は、パッケージ内の"reference_documents"フォルダにあります。

対象デバイス

RX65N/RX651 Group
RX64M Group
RX71M Group
RX66T Group
RX72T Group
RX72M Group
RX66N Group
RX72N Group
RX671 Group

本プログラムは Renesas Starter Kits (RSK)を使って動作確認を行っています。

目次

1. はじめに

1.1 機能概要

PCDC は、USB コミュニケーションデバイスクラス仕様（以降 CDC と記述）の Abstract Control Model に準拠し、USB ホスト PC と通信を行うことが可能です。
PCDC の機能を以下に示します。

- USB－シリアル変換機能及び、USB ループバック通信機能（エコーモード）を実装しています。
- USB ホストと接続時、コミュニケーションクラス（仮想 COM）として認識される。
- ターミナルソフトで仮想 COM ポートを指定することで、通信を行うことが可能。

1.2 FIT モジュール構成

PCDC は以下の FIT モジュールとサンプルアプリケーションで構成されています。

Table 1-1　FIT モジュール構成

FIT モジュール名	フォルダ名
RX Family Board Support Package Module Firmware Integration Technology	r_bsp
RX Family USB Basic Host and Peripheral Driver Firmware Integration Technology	r_usb_basic
RX Family USB Peripheral Communications Devices Class Driver(PCDC) Firmware Integration Technology	r_usb_pcdc
RX Family DTC Module Firmware Integration Technology	r_dtc_rx
RX Family DMA Controller DMACA Control Module Firmware Integration Technology	r_dmaca_rx
RX Family BYTEQ Module Using Firmware Integration Technology	r_byteq
RX Family SCI Multi-Mode Module Firmware Integration Technology	r_sci_rx

各 FIT モジュールの詳細は、関連ドキュメントを参照してください。また、本サンプルファームウェアで使用している FIT モジュールの最新バージョンは下記のホームページよりダウンロードが可能です。

ルネサスエレクトロニクスホームページ http://japan.renesas.com/

1.3 注意事項

本ドライバは、USB 通信動作を保証するものではありません。システムに適用される場合は、お客様における動作検証はもとより、多種多様なデバイスに対する接続確認を実施してください。

1.4 動作確認済環境

動作確認環境を以下に示します。

項目	内容
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler for RX Family V.3.03.00
	コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
リアルタイム OS	FreeRTOS V.10.0.0 RI600V4 Azure RTOS (USBX) 6.1.11
エンディアン	リトルエンディアン / ビッグエンディアン
モジュールのリビジョン	Rev.1.40
使用ボード	Renesas Starter Kits for RX64M Renesas Starter Kits for RX71M Renesas Starter Kits for RX65N, Renesas Starter Kits for RX65N-2MB Renesas Starter Kits for RX72T Renesas Starter Kits for RX72M Renesas Starter Kits for RX72N Renesas Starter Kits for RX671
ホスト環境	下記の OS に接続し動作確認を行っています。 1. Windows® 8.1 2. Windows® 10

1.5 用語一覧

本資料で使用される用語と略語は以下のとおりです。

APL	:	Application program
CDC	:	Communications Devices Class
H/W	:	Renesas USB device
Non-OS	:	USB Driver for OS-less
PCD	:	Peripheral Control Driver for USB-BASIC-FW
PCDC	:	Peripheral Communications Devices Class
RTOS		USB Driver for the real-time OS
RSK	:	Renesas Starter Kits
SCI	:	Serial Communication Interface
USB-BASIC-FW	:	USB Basic Host and Peripheral Driver

2.　ソフトウェア構成

2.1　モジュール構成

PCDC は USB－シリアル変換機モード及び、USB ループバックモード（エコーモード）があり、USB-シリアル変換モードではシリアル・コミュニケーション・インタフェース（SCI）を使用します。LCD 表示や低消費電力制御処理はサンプルアプリケーションとして実装しています。

Figure 2-1に PCDC のモジュール構成、Table 2-1にモジュール機能概要を示します。

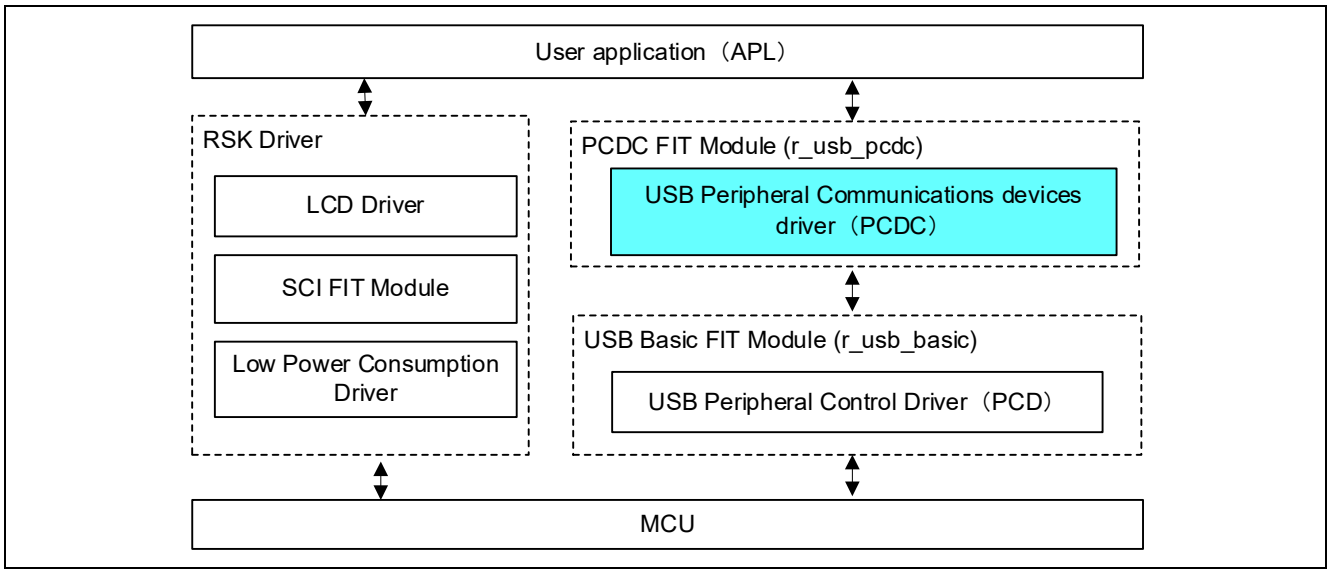


Figure 2-1　モジュール構成図

Table 2-1　モジュール機能概要

モジュール名	機能概要
APL	サンプルアプリケーションプログラム
RSK driver	RSK 上の各周辺機能を使用するためのサンプルアプリケーション
PCDC (r_usb_pcdc)	CDC クラスドライバ ・USB ホストからの要求を解析する。 ・PCD を介して APL と USB ホスト間のデータ転送サービスを提供する。
PCD (r_usb_basic)	USB Peripheral H/W 制御ドライバ

3. セットアップ

3.1 ハードウェア

3.1.1 動作環境例

PCDC の動作環境例をFigure 3-1とFigure 3-2に示します。評価ボードのセットアップ、エミュレータなどの使用方法については各取扱説明書を参照してください。

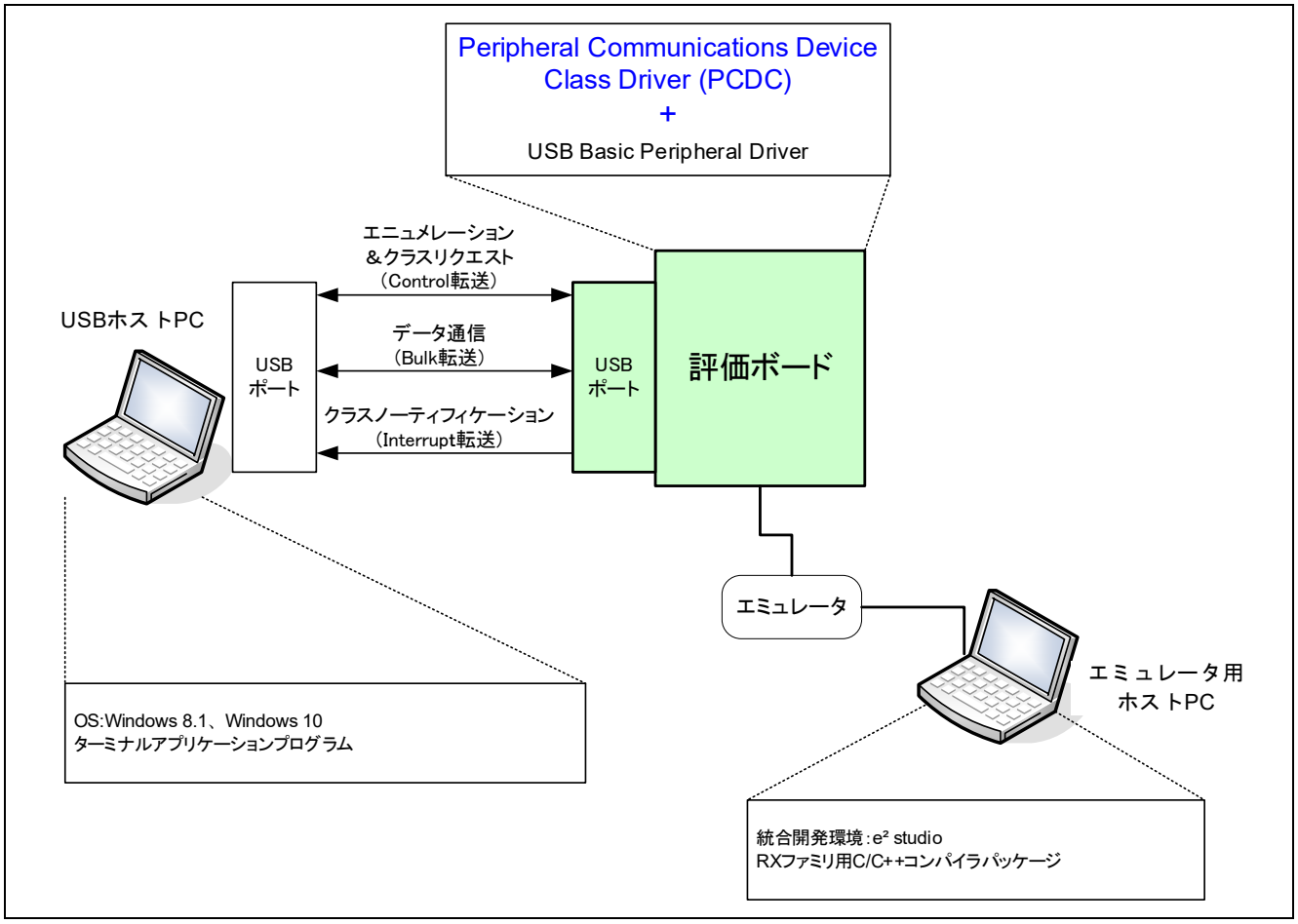


Figure 3-1　Example Operating Environment　(Echo モード)

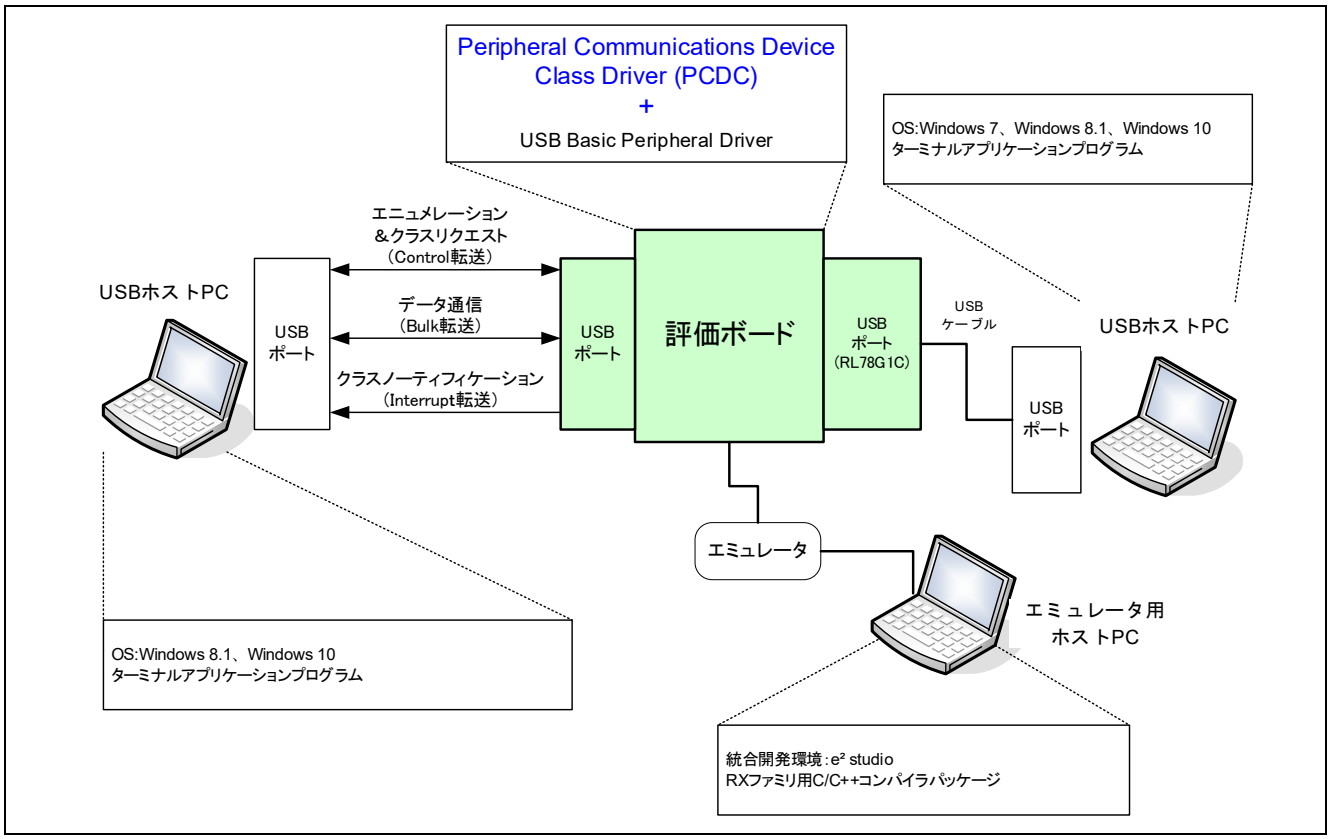


Figure 3-2　Example Operating Environment　(USB-シリアル変換モード)

動作確認済みの評価ボードをTable 3-1に示します。

Table 3-1　PCDC 動作確認済みの評価ボード

MCU	評価ボード
RX65N	RSK+RX65N, RSK+RX65N-2MB
RX64M	RSK+RX64M
RX71M	RSK+RX71M
RX72T	RSKRX72T
RX72M	RSK+RX72M
RX72N	RSK+RX72N
RX671	RSK+RX671

3.1.2 RSK 設定

RSK を USB Peripheral モードに設定する必要があります。設定内容は以下を参照してください。

Table 3-2 **RSK 設定**

RSK	ジャンパ設定
RSK+RX65N	J8: Shorted Pin2-3
RSK+RX65N_2MB	J7: Shorted Pin2-3 J16: Shorted Pin1-2
RSK+RX64M (USB0)	J2: Shorted Pin2-3 J6: Shorted Pin1-2
RSK+RX64M (USBH)	J7: Shorted Pin2-3 J9: Shorted Pin1-2
RSK+RX71M (USB0)	J1: Shorted Pin2-3 J3: Shorted Pin1-2
RSK+RX71M (USBA)	J4: Shorted Pin2-3 J7: Shorted Pin1-2
RSKRX72T	J13: Shorted Pin2-3
RSK+RX72M	J8: Shorted Pin1-2 J10: Shorted Pin1-2
RSK+RX72N	J7: Shorted Pin1-2 J8: Shorted Pin1-2
RSK+RX671	J8: Shorted Pin1-2 J13: Shorted Pin1-2

[Note]

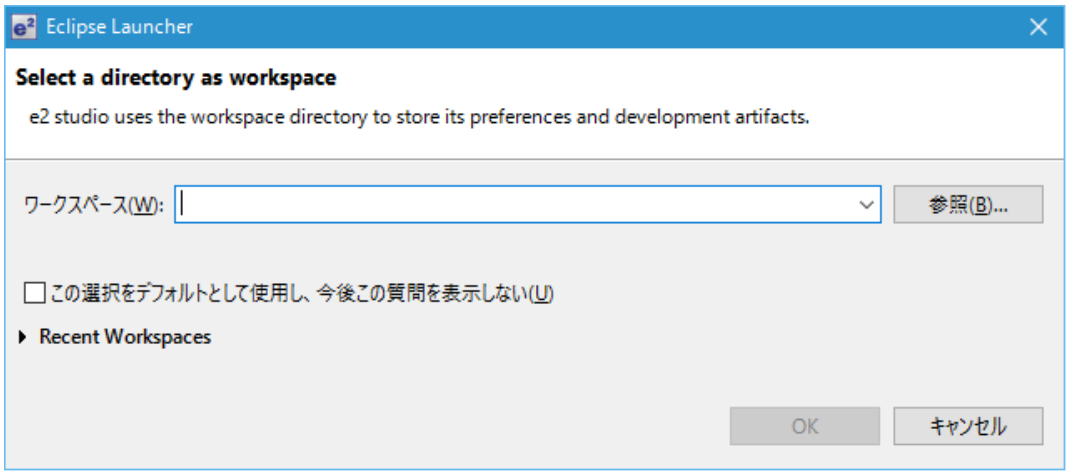
RSK 設定の詳細については、RSK のユーザーズマニュアルを参照してください。

3.2 ソフトウェア

(1). e² studio を起動

a) e² studio を起動してください。

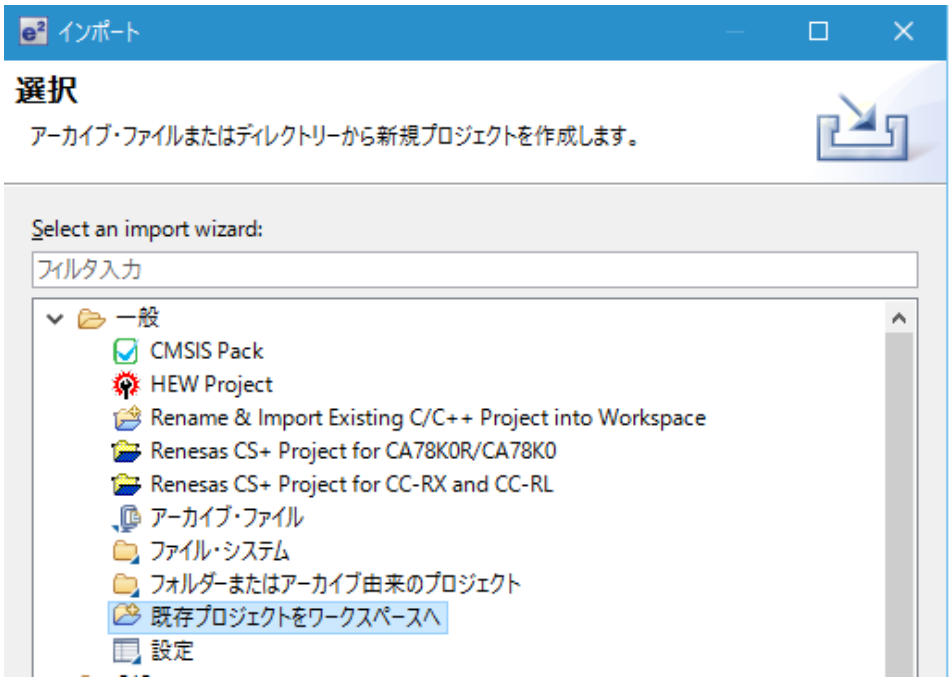
b) はじめて e² studio を起動する場合、Workspace Launcher ダイアログが表示されますので、プロジェクトを格納するためのフォルダを指定してください。



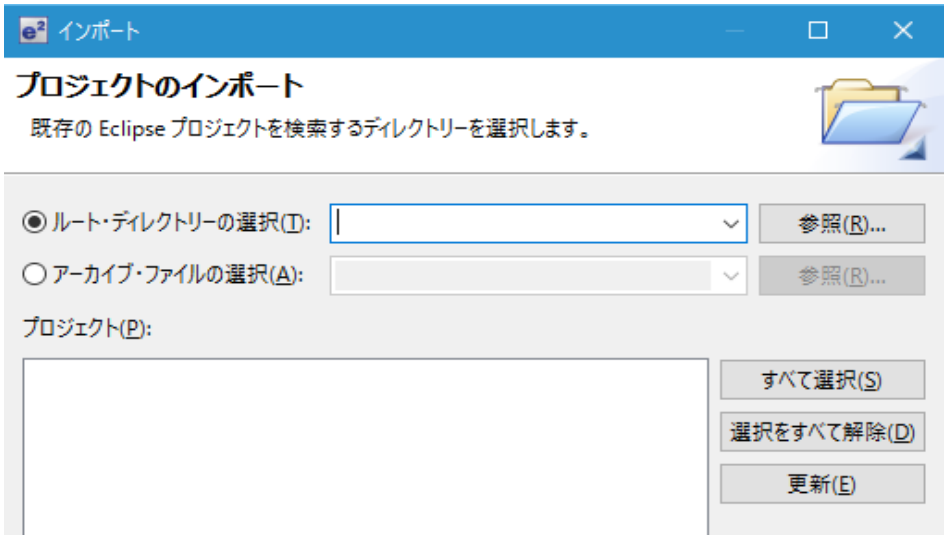
(2). プロジェクトをワークスペースへインポート

a) [ファイル] --> [インポート]を選択してください。

b) [一般] => [既存プロジェクトをワークスペースへ]を選択してください。



c) プロジェクトファイル".cproject"が格納されたフォルダを”Select root directory”に入力してください。



d) “Finish”をクリック

プロジェクトのワークスペースへのインポートが完了しました。同様の方法で他のプロジェクトを同一のワークスペースへインポートすることができます。

(3). “Build”ボタンをクリックし、実行プログラムを生成してください。

(4). デバッガへの接続を行い、実行プログラムをダウンロードしてください。“Run”ボタンをクリックすると、プログラムが実行されます。

4. サンプルアプリケーション

4.1 アプリケーション仕様

APL の主な機能を以下に示します。

1) Echo(ループバック)モード（注 1）

USB ホストから受信したデータを、USB ホストへ送信します。

2) USB-シリアル変換モード（注 1）(注 2) (注 3) (注 4)

USB ホストから受信したデータを COM ポートに送信し、COM ポートから受信したデータを USB ホストに送信します。なお、COM ポートのエラー（パリティエラー、フレーミングエラー、オーバーランエラー）が発生した場合、USB ホストにクラスノーティフィケーション“SerialState”を通知します。

3) 消費電力低減機能

USB の状態に応じて MCU を消費電力低減モードに遷移させる機能です。

a) USB サスペンド状態時に MCU をスリープモードに遷移させます。
b) USB デタッチ（切断）状態時に、MCU をソフトウエアスタンバイモードに遷移させます。

（注1）Echo モード/USB-シリアル変換モードの選択は、“r_usb_pcdc_apl_config.h”ファイル内で行ってください。

（注2）COM ポートは RSK 上で RL78G1C と接続しており、G1CUSB0 を使用した USB-USB 通信となります。

（注3）USB-シリアル変換モードの場合、RSK に対し以下のジャンパ設定が必要です。

Table 4-1　USB-シリアル変換モードジャンパ設定

RSK+RX64M / RSK+RX71M		
ポート	信号	ジャンパ設定
P90	A-TXD7	J16.2-3
P92	A-RXD7	J19.2-3

RSK+RX65N		
ポート	信号	ジャンパ設定
P50	TXD2	--
P52	RXD2	--

RSKRX72T		
ポート	信号	ジャンパ設定
PB5	TXD5	--
PB6	RXD5	--

RSK+RX65N_2MB		
ポート	信号	ジャンパ設定
PJ2	TXD8	--
PJ1	RXD8	--

RSK+RX72M		
ポート	信号	ジャンパ設定
P00	TXD6	--
P01	RXD6	--

RSK+RX72N		
ポート	信号	ジャンパ設定
PL2	TXD9	--
PL1	RXD9	--

RSK+RX671		
ポート	信号	ジャンパ設定
P87	TXD	--
P86	RXD	--

（注4）USB-シリアル変換モードの場合、以下の FIT モジュールが必要です。

a) RX Family Serial Communication Interface Firmware Integration Technology
b) RX Family BYTEQ Module Firmware Integration Technology

4.2 アプリケーション処理概要 (Non-OS)

APL は、初期設定、メインループの 2 つの部分から構成されます。以下にそれぞれの処理概要を示します。

4.2.1 初期設定

初期設定では、MCU の端子設定、USB ドライバの設定、USB コントローラの初期設定を行います。

4.2.2 メインループ (Echo モード)

Echo モードでは、USB Host から送信されるデータを受信し、そのまま USB Host へ送信するループバック処理をメインに行います。以下にメインループの処理概要を示します。

1. USB Host との Enumeration 完了後に R_USB_GetEvent 関数をコールすると戻り値に USB_STS_CONFIGURED がセットされます。APL では、USB_STS_CONFIGURED を確認すると R_USB_Read 関数をコールし、USB Host から送信されるデータのデータ受信要求を行います。
2. USB Host との Enumeration が完了すると USB Host は、CDC クラスリクエストを CDC デバイスに送信します。CDC クラスリクエスト受信後、R_USB_GetEvent 関数をコールすると戻り値に USB_STS_REQUST がセットされます。APL では、USB_STS_REQUEST を確認すると、受信したクラスリクエストを解析し、そのクラスリクエストに対応する処理を行います。
3. USB Host からのデータ受信が完了し、R_USB_GetEvent 関数をコールすると戻り値に USB_STS_READ_COMPLETE がセットされます。APL では、USB_STS_READ_COMPLETE を確認すると R_USB_Write 関数をコールし、受信データを USB Host に送信するためのデータ送信要求を行います。
4. USB Host へのデータ送信が完了し、R_USB_GetEvent 関数をコールすると戻り値に USB_STS_WRITE_COMPLETE がセットされます。APL では、USB_STS_WRITE_COMPLETE を確認すると R_USB_Read 関数をコールし、USB Host から送信されるデータのデータ受信要求を行います。
5. 上記3と4の処理が繰り返し行われます。
6. USB Host からのサスペンド信号の受信や DETACH を確認すると、APL は CDC デバイス(RSK)を低消費電力モードに移行するための処理を行います。消費電力低減モードについては、「4.4 MCU消費電力低減処理」を参照してください。なお、サスペンド信号の受信や DETACH の確認は、R_USB_GetEvent 関数の戻り値(USB_STS_SUSPEND/USB_STS_DETACH)により行います。

以下に、APL の処理概要を示します。

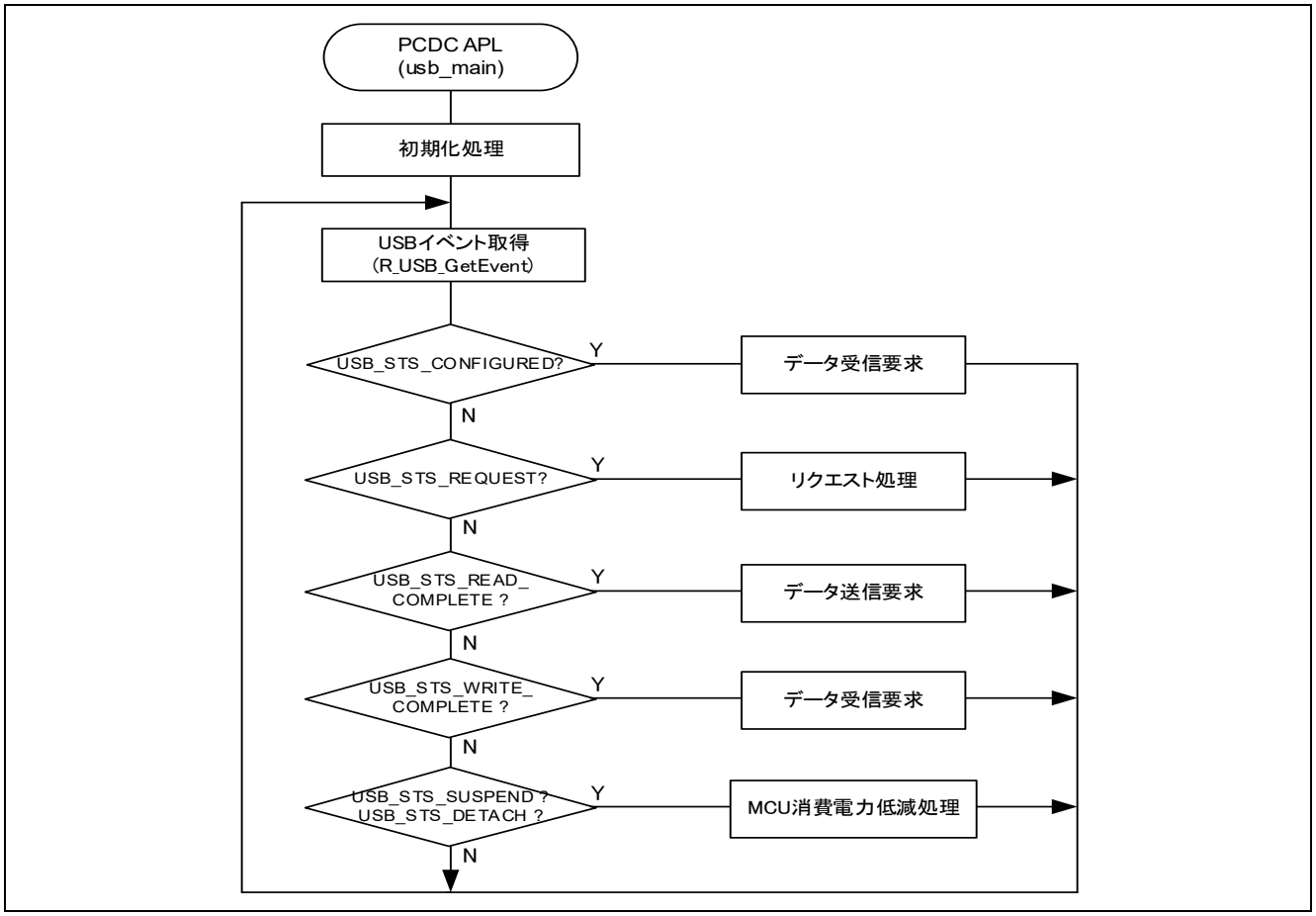


Figure 4-1　メインループ処理 (ループバックモード)

4.2.3 メインループ (USB-シリアル変換モード)

USB-シリアル変換バックモードは、以下の処理を行います。

a. USB ホストからデータを受信し、その受信したデータの COM ポートへの送信。

b. COM ポートから受信したデータの USB ホストへの送信。

以下にメインループの処理概要を示します。

1. USB Host との Enumeration 完了後に R_USB_GetEvent 関数をコールすると戻り値に USB_STS_CONFIGURED がセットされます。APL では、USB_STS_CONFIGURED を確認すると R_USB_Read 関数をコールし､USB Host から送信される Bulk データのデータ受信要求を行います。
2. USB Host との Enumeration が完了すると USB Host は、CDC クラスリクエストを CDC デバイスに送信します。CDC クラスリクエスト受信後、R_USB_GetEvent 関数をコールすると戻り値に USB_STS_REQUST がセットされます。APL では、USB_STS_REQUEST を確認すると、受信したクラスリクエストを解析し、そのクラスリクエストに対応する処理を行います。
3. 上記2でのクラスリクエスト処理完了後、R_USB_GetEvent 関数をコールすると戻り値に USB_STS_REQUST_COMPLETE がセットされます。APL では、リクエスト情報の設定処理等を行っています。
4. USB Host からの Bulk データ受信が完了し、R_USB_GetEvent 関数をコールすると戻り値に USB_STS_READ_COMPLETE がセットされます。APL では、USB_STS_READ_COMPLETE を確認すると受信データサイズを外部変数にセットします。
5. USB Host への USB データ送信(下記6参照)が完了し、R_USB_GetEvent 関数をコールすると戻り値に USB_STS_WRITE_COMPLETE がセットされます。APL では、USB_STS_WRITE_COMPLETE を確認すると usb_ctrl_t 構造体のメンバ type を参照し、データ送信が完了したデバイスクラス種別を確認します。送信が完了したデバイスクラス種別に応じ、該当の完了フラグをセットします。このフラグは、下記6の処理で参照されます。
6. 上記の処理が完了した後、以下の送信処理が行われます。なお、以下の送信処理を行う前に上記4と5でセットされたフラグを参照し、送信可能かどうかを判断しています。
   (1). USB Host から受信した Bulk データを COM ポートへ送信する SCI 送信処理。および USB Host からのデータ受信要求処理。
   (2). SCI エラー(Parity error/Framing error/Overrun error など)を検出した場合、USB Host へ通知するための Class Notification (Serial State)送信要求処理
   (3). COM ポートから受信したデータを USB Host へ送信するためデータ送信要求処理。
7. 上記の処理が繰り返し行われている間に、USB Host からのサスペンド信号の受信や DETACH を確認すると、APL は CDC デバイス(RSK)を低消費電力モードに移行するための処理を行います。消費電力低減モードについては、「4.4 MCU消費電力低減処理」を参照してください。なお、サスペンド信号の受信や DETACH の確認は、R_USB_GetEvent 関数の戻り値(USB_STS_SUSPEND / USB_STS_DETACH)により行います。

以下に、APL の処理概要を示します。

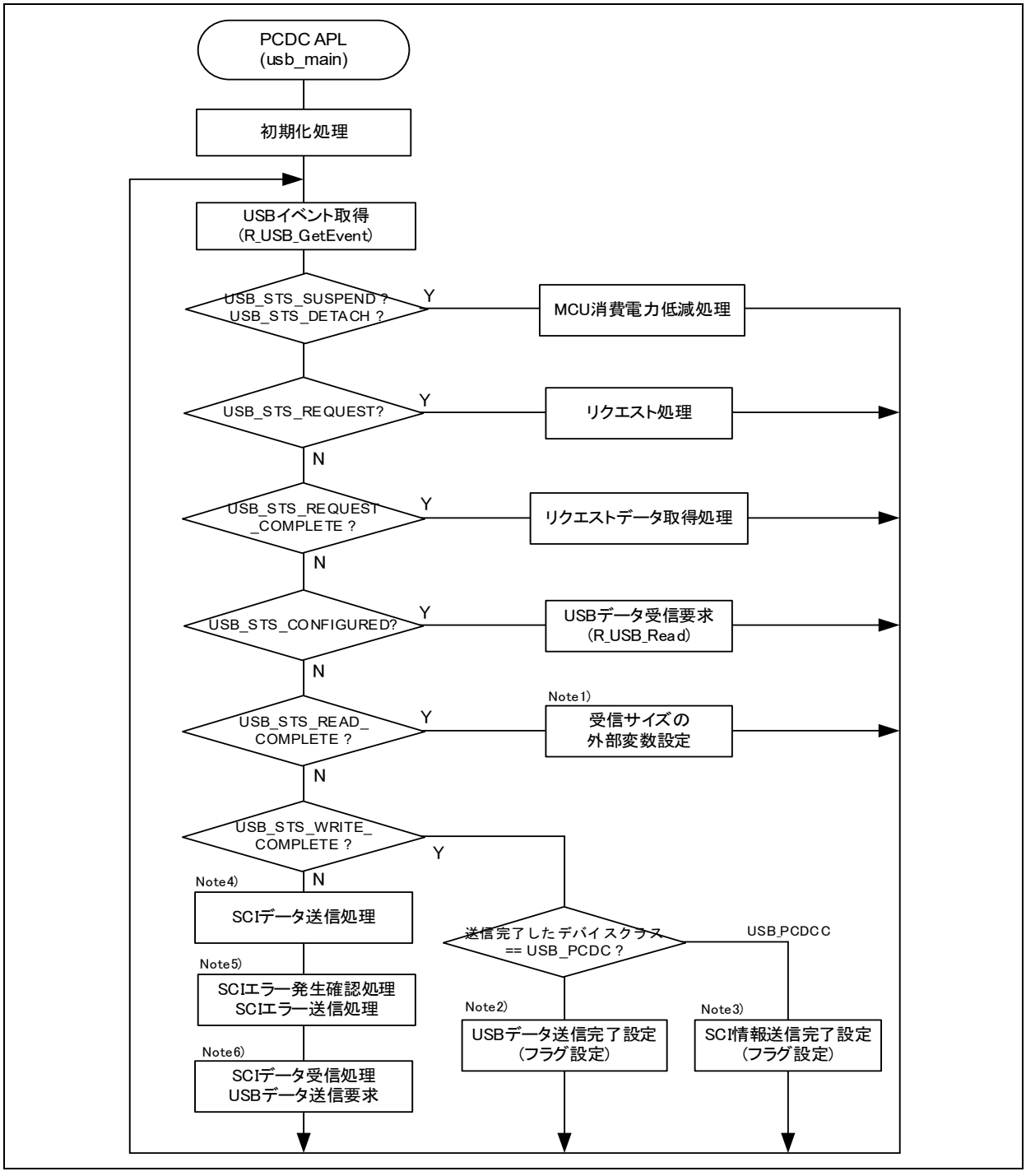


Figure 4-2　メインループ処理 (USB-シリアル変換モード)

（Note1） 受信した USB データサイズが外部変数に設定されます。この変数は（Note4）の処理で参照されます。

（Note2） USB データ送信完了のフラグ設定を行います。このフラグは（Note6）の処理で参照されます。

（Note3） SCI 情報(SerialState)送信完了のフラグ設定を行います。このフラグは（Note5）の処理で参照されます。

（Note4） （Note1）の外部変数を参照し、SCI データ送信処理を行います。SCI データ送信が正常終了した場合、SCI 送信要求フラグに対するクリア設定を行います。SCI データ送信に失敗した場合は、再度 SCI データ送信処理を行うため SCI 送信要求フラグのクリア設定を行いません。

（Note5） SCI 状態にエラーが発生したかどうかの確認処理を行います。エラーが発生した場合、R_USB_Write 関数を使用し、USB Host に対しそのエラー情報を送信します。

（Note6） （Note2）のフラグ状態を参照し、USB 送信が可能かどうかを確認します。送信可能であれば、USB_Write 関数を使用し、受信した SCI データを USB Host に送信します。なお、USB Host へのデータ送信が完了するまでは、COM ポートからの受信確認は行いません。

4.3 アプリケーション処理概要 (RTOS)

APL は、初期設定、メインループの 2 つの部分から構成されます。以下にそれぞれの処理概要を示します。

4.3.1 初期設定

初期設定では、MCU の端子設定、USB ドライバの設定、USB コントローラの初期設定を行います。

4.3.2 メインループ (Echo モード)

このループ処理では、USB Host から送信されるデータを受信し、そのまま USB Host へ送信するループバック処理をメインに行います。以下にメインループの処理概要を示します。以下の処理は FreeRTOS 版及び uITRON 用の処理です。

1. USB 関連のイベントが完了すると USB ドライバはコールバック関数(usb_apl_callback)をコールします。コールバック関数(usb_apl_callback)では、リアルタイム OS の機能を使って USB 完了イベントを APL(アプリケーションタスク)に通知します。
2. APL ではコールバック関数から通知された USB 完了イベント等の情報をリアルタイム OS の機能を使って取得します。
3. 上記2で取得した USB 完了イベント(usb_ctrl_t 構造体:メンバ event)が USB_STS_CONFIGURED の場合、APL は、R_USB_Read 関数をコールし、USB Host からの送信されるデータのデータ受信要求を行います。
4. 上記2で取得した USB 完了イベント(usb_ctrl_t 構造体:メンバ event)が USB_STS_REQUEST の場合、APL は、受信したリクエストに対応する処理を行います。
5. 上記2で取得した USB 完了イベント(usb_ctrl_t 構造体:メンバ event)が USB_STS_REQUEST の場合、APL では、リクエスト情報の設定処理等を行っています。
6. 上記2で取得した USB 完了イベント(usb_ctrl_t 構造体:メンバ event)が USB_STS_READ_COMPLETE の場合、APL は R_USB_Write 関数をコールし、受信したデータを USB Host へ送信するためのデータ送信要求を行います。
7. 上記2で取得した USB 完了イベント(usb_ctrl_t 構造体:メンバ event)が USB_STS_WRITE_COMPLETE の場合、APL は R_USB_Read 関数をコールし、USB Host から送信されるデータの受信要求を行います。
8. 上記2で取得した USB 完了イベント(usb_ctrl_t 構造体:メンバ event)が USB_STS_SUSPEND または、USB_STS_DETACH の場合、APL は CDC デバイス(RSK)を低消費電力モードに移行するための処理を行います。消費電力低減モードについては、「4.4 MCU消費電力低減処理」を参照してください。

以下に、APL の処理概要を示します。

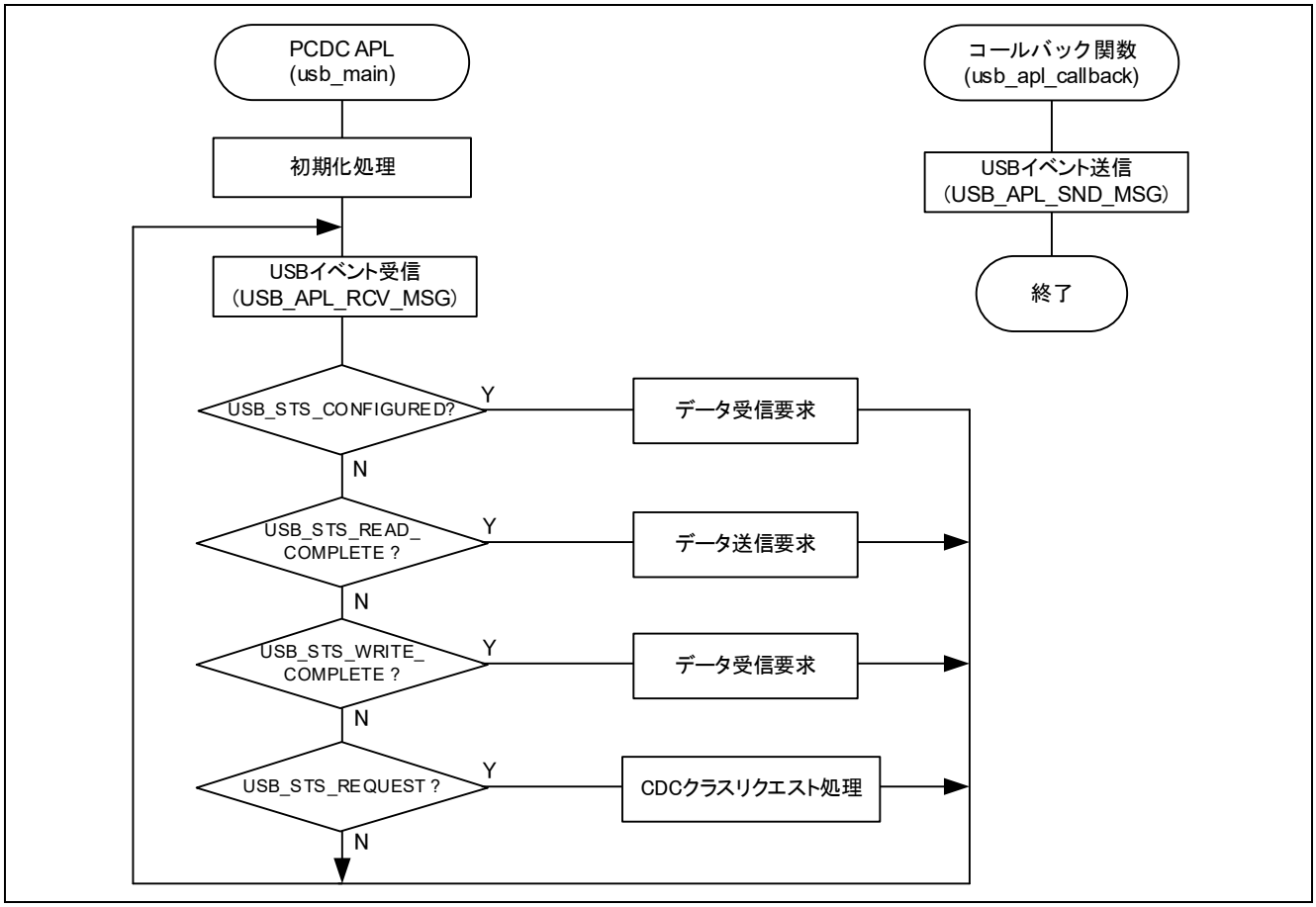


Figure 4-3　メインループ処理 (Echo モード)

4.3.3 メインループ (USB-シリアル変換モード)

USB-シリアル変換バックモードは、以下の処理を行います。なお、Azure RTOS ではこのモードはサポートされていません。

a. USB ホストからデータを受信し、その受信したデータの COM ポートへの送信。

b. COM ポートから受信したデータの USB ホストへの送信。

以下にメインループの処理概要を示します。

1. USB 関連のイベントが完了すると USB ドライバはコールバック関数(usb_apl_callback)をコールします。コールバック関数(usb_apl_callback)では、リアルタイム OS の機能を使って USB 完了イベントを APL(アプリケーションタスク)に通知します。
2. APL ではコールバック関数から通知された USB 完了イベント等の情報を FreeRTOS の機能を使って取得します。
3. 上記2で取得した USB 完了イベント(usb_ctrl_t 構造体:メンバ event)が USB_STS_CONFIGURED の場合、APL はこの USB 完了イベントをもとに、R_USB_Read 関数をコールし、USB Host からの送信されるデータのデータ受信要求を行います。
4. 上記2で取得した USB 完了イベント(usb_ctrl_t 構造体:メンバ event)が USB_STS_REQUEST の場合、APL はこの USB 完了イベントをもとに、受信したリクエストに対応する処理を行います。
5. 上記2で取得した USB 完了イベント(usb_ctrl_t 構造体:メンバ event)が USB_STS_READ_COMPLETE の場合、APL は受信データサイズを外部変数にセットします。
6. 上記2で取得した USB 完了イベント(usb_ctrl_t 構造体:メンバ event)が USB_STS_WRITE_COMPLETE の場合、APL は usb_ctrl_t 構造体のメンバ type を参照し、データ送信が完了したデバイスクラス種別を確認します。送信が完了したデバイスクラス種別に応じ、該当の完了フラグをセットします。このフラグは、下記8の処理で参照されます。
7. 上記で取得した USB 完了イベント(usb_ctrl_t 構造体:メンバ event)が USB_STS_SUSPEND または、USB_STS_DETACH の場合、APL は CDC デバイス(RSK)を低消費電力モードに移行するための処理を行います。消費電力低減モードについては、「4.4 MCU消費電力低減処理」を参照してください。
8. 上記の処理後、以下の送信処理が行われます。なお、以下の送信処理を行う前に上記5と6でセットされたフラグを参照し、送信可能かどうかを判断しています。
   (1). USB Host から受信した Bulk データを COM ポートへ送信する SCI 送信処理。および USB Host からのデータ受信要求処理。
   (2). SCI エラー(Parity error/Framing error/Overrun error など)を検出した場合、USB Host へ通知するための Class Notification (Serial State)送信要求処理
   (3). COM ポートから受信したデータを USB Host へ送信するためデータ送信要求処理。

以下に、APL の処理概要を示します。

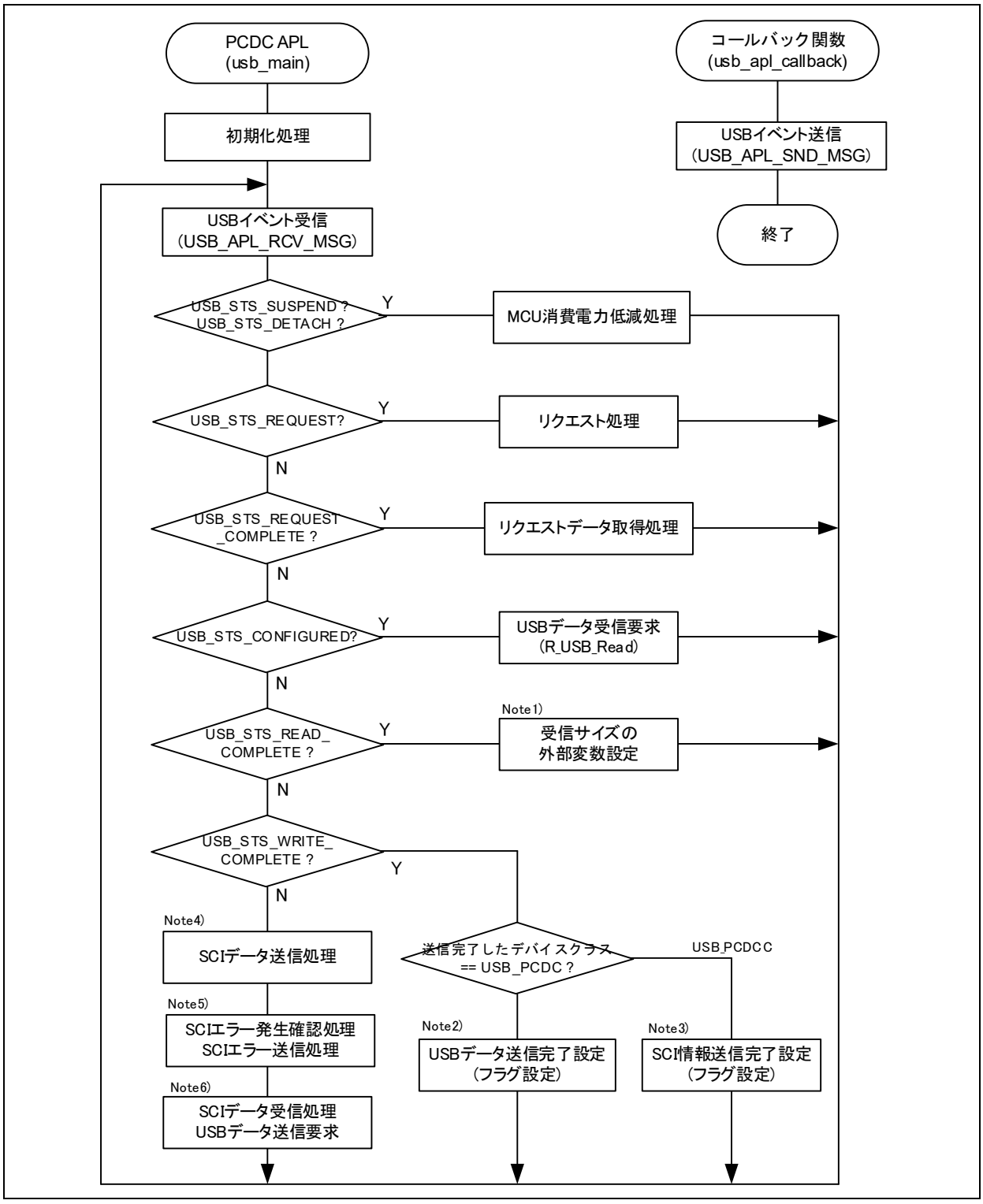


Figure 4-4　メインループ処理 (USB-シリアル変換モード)

（Note1） 受信した USB データサイズが外部変数に設定されます。この変数は（Note4）の処理で参照されます。

（Note2） USB データ送信完了のフラグ設定を行います。このフラグは（Note6）の処理で参照されます。

（Note3） SCI 情報(SerialState)送信完了のフラグ設定を行います。このフラグは（Note5）の処理で参照されます。

（Note4） （Note1）の外部変数を参照し、SCI データ送信処理を行います。SCI データ送信が正常終了した場合、SCI 送信要求フラグに対するクリア設定を行います。SCI データ送信に失敗した場合は、再度 SCI データ送信処理を行うため SCI 送信要求フラグのクリア設定を行いません。

（Note5） SCI 状態にエラーが発生したかどうかの確認処理を行います。エラーが発生した場合、R_USB_Write 関数を使用し、USB Host に対しそのエラー情報を送信します。

（Note6） （Note2）のフラグ状態を参照し、USB 送信が可能かどうかを確認します。送信可能であれば、USB_Write 関数を使用し、受信した SCI データを USB Host に送信します。なお、USB Host へのデータ送信が完了するまでは、COM ポートからの受信確認は行いません。

4.4　MCU 消費電力低減処理

MCU 消費電力低減処理は、Table 4-2またはTable 4-3の条件が成立すると消費電力低減モードに移行する処理を行います。なお、この処理を有効にするには、“r_usb_pcdc_apl_config.h”ファイル内に“USB_SUPPORT_LPW”定義に対し、"USB_APL_ENABLE"を指定してください。

1. Non-OS の場合

Table 4-2 消費電力低減機能状態遷移条件

遷移条件		遷移状態
VBUS	USB ステート	
OFF	－	ソフトウエアスタンバイモード
ON	Suspend Configured	スリープモード
ON	Suspend Configured 以外	通常モード（プログラム実行状態）

(1). CDC デバイス(RSK)が USB Host からデタッチ（VBUS OFF）されると、APL は MCU をソフトウエアスタンバイモードに遷移するための処理を行います。ソフトウエアスタンバイモードからの復帰は、CDC デバイス(RSK)を USB Host にアタッチすることにより行われます。

(2). CDC デバイス(RSK)を USB Host に接続した状態で、USB Host から送信されるサスペンド信号を受信すると APL は、MCU をスリープモードに遷移するための処理を行います。なお、スリープモードからの復帰は、USB Host から送信されるレジューム信号の受信により行われます。

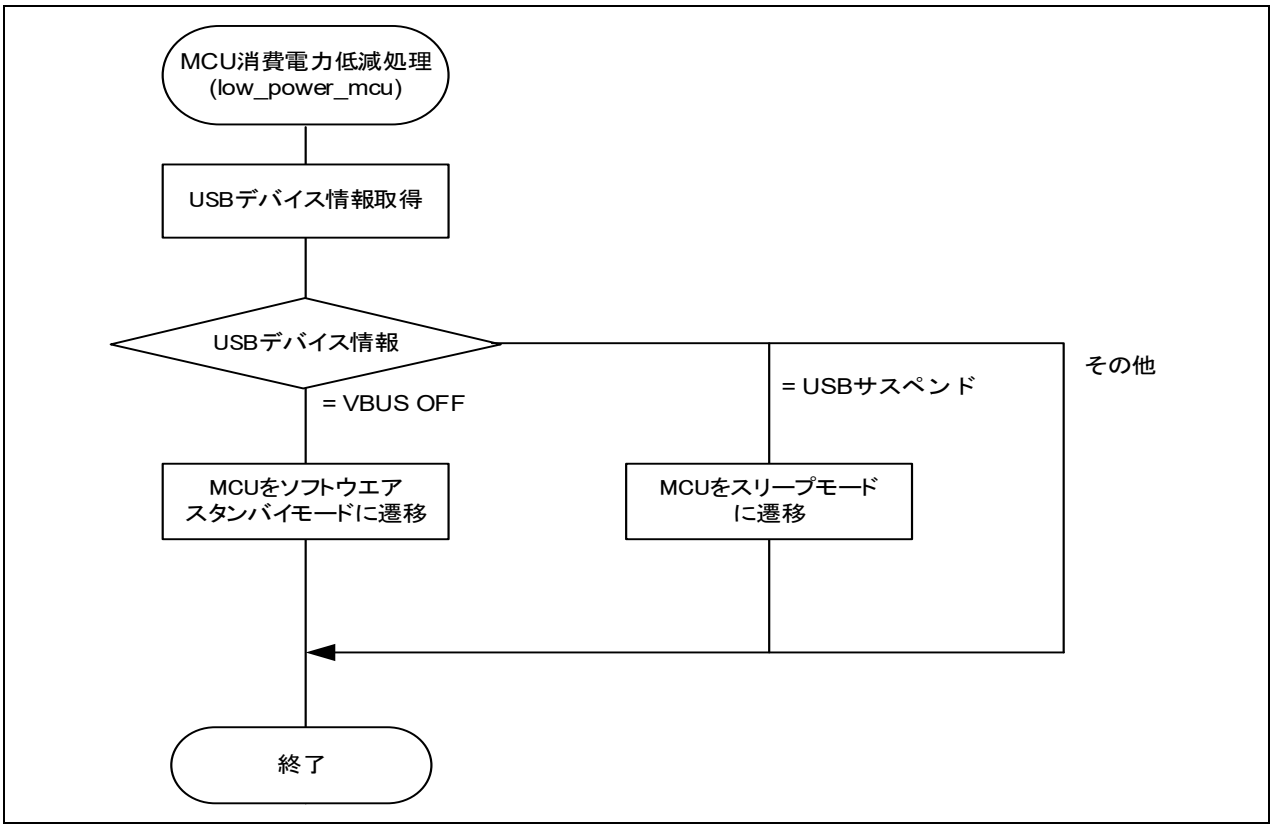


Figure 4-5　MCU 消費電力低減処理概略フロー

2. RTOS の場合 (FreeRTOS のみ)

Table 4-3 消費電力低減機能状態遷移条件

遷移条件		遷移状態
VBUS	USB ステート	
OFF	ー	ソフトウエアスタンバイモード
ON	Suspend Configured	ソフトウエアスタンバイモード
ON	Suspend Configured 以外	通常モード（プログラム実行状態）

(1). CDC デバイス(RSK)が USB Host からデタッチ（VBUS OFF）されると、APL は MCU をソフトウエアスタンバイモードに遷移するための処理を行います。ソフトウエアスタンバイモードからの復帰は、CDC デバイス(RSK)を USB Host にアタッチすることにより行われます。

(2). CDC デバイス(RSK)を USB Host に接続した状態で、USB Host から送信されるサスペンド信号を受信すると APL は、MCU をソフトウエアスタンバイモードに遷移するための処理を行います。なお、ソフトウエアスタンバイモードからの復帰は、USB Host から送信されるレジューム信号の受信により行われます。

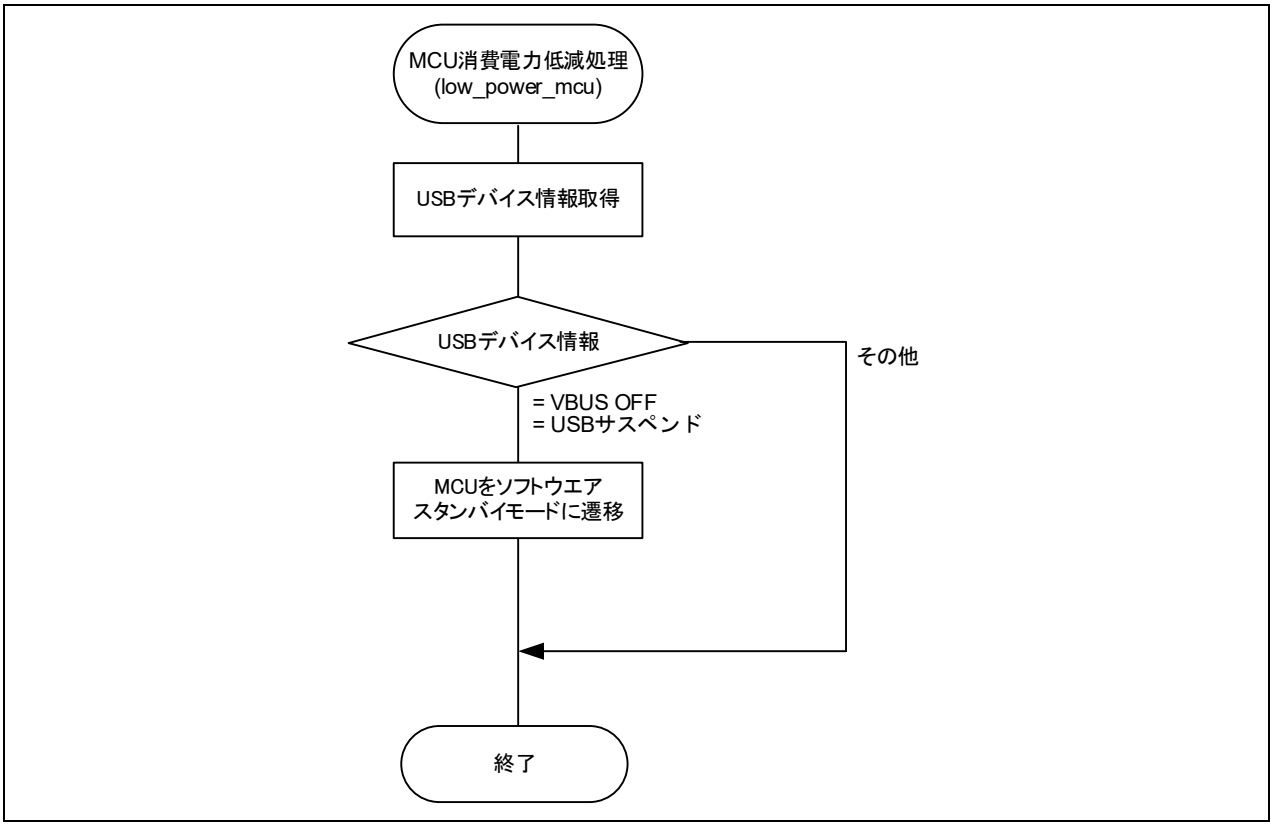


Figure 4-6 MCU 消費電力低減処理概略フロー

4.5 アプリケーションプログラム用コンフィグレーションファイル (r_usb_pcdc_apl_config.h)

以下の各定義に対する設定を行ってください。

1. USE_USBIP 定義

使用する USB モジュールのモジュール番号を指定してください。USB_IP0/USB_IP1 のいずれかを指定してください。

```
#define    USE_USBIP    USE_USBIP0    // USB0 モジュールを使用する場合
#define    USE_USBIP    USE_USBIP1    // USB1 モジュールを使用する場合
```

2. USB_SUPPORT_SPEED 定義

CDC デバイス(RSK)がサポートする USB 動作スピード(Hi-speed/Full-speed)を指定してください。

```
#define    USB_SUPPORT_SPEED    HI_SPEED      // Hi-Speed 設定
#define    USB_SUPPORT_SPEED    FULL_SPEED    // Full-Speed 設定
```

[Note]

RX71M をご使用の場合に、HI_SPEED を指定することができます。RX71M 以外の MCU をご使用の場合は、FULL_SPEED を指定してください。

3. OPERATION_MODE 定義

OPERATION_MODE 定義に対し、以下のいずれかを指定してください。

```
#define    OPERATION_MODE    USB_ECHO     // Echo モード
#define    OPERATION_MODE    USB_UART     // USB-シリアル変換モード
```

4. 消費電力低減機能定義

消費電力低減機能の使用/非使用を指定してください。消費電力低減機能を使用する場合は、USB_SUPPORT_LPW 定義に対し USB_APL_ENABLE を指定し、消費電力低減機能を使用しない場合は、USB_SUPPORT_LPW 定義に対し USB_APL_DISABLE を指定してください。

```
#define    USB_SUPPORT_LPW    USB_APL_DISABLE    //消費電力低減機能を非使用
#define    USB_SUPPORT_LPW    USB_APL_ENABLE     //消費電力低減機能を使用
```

5. USB_SUPPORT_RTOS 定義

リアルタイム OS を使用するかどうかを指定します。リアルタイム OS を使用する場合は、USB_SUPPORT_RTOS 定義に対し USB_APL_ENABLE を指定してください。

```
#define    USB_SUPPORT_RTOS    USB_APL_DISABLE    // RTOS 非使用
#define    USB_SUPPORT_RTOS    USB_APL_ENABLE     // RTOS 使用
```

6. 注意事項

上記はアプリケーションプログラム用のコンフィグレーション設定です。上記の設定の他に USB ドライバのコンフィグレーション設定(r_usb_basic_config.h)が必要です。USB ドライバのコンフィグレーション設定については、「USB Basic Host and Peripheral Driver Firmware Integration Technology アプリーションノート」(Document No. R01AN2025JJ)を参照してください。

4.6 ディスクリプタ

PCDC のディスクリプタ情報は r_usb_pcdc_descriptor.c に記述しています。なお、Vendor ID は、必ずお客様用の Vendor ID をご使用いただきますようお願いします。

5. CDC ドライバのインストール

USB HostがPCの場合、そのPCに対しCDCドライバをインストールする必要があります。本サンプルプログラムの書き込みを行ったRSKをPCに接続すると、Figure 5-1に示すウィザードが表示され、CDCドライバのインストールが行われます。

(1). デバイス・マネージャより、ドライバーソフトウェアの更新を選択します。

(2). 《コンピューターを参照してドライバーソフトウェアを検索します（R）》を選択します。

Note:

(1). PCのOSがWindows 10の場合、CDCドライバのインストール作業は不要です。

(2). PCのOSがWindows® 8.1の場合、デジタル署名済のカタログファイルが必要になります。デジタル署名済のカタログファイルはお客様により作成いただく必要があります。

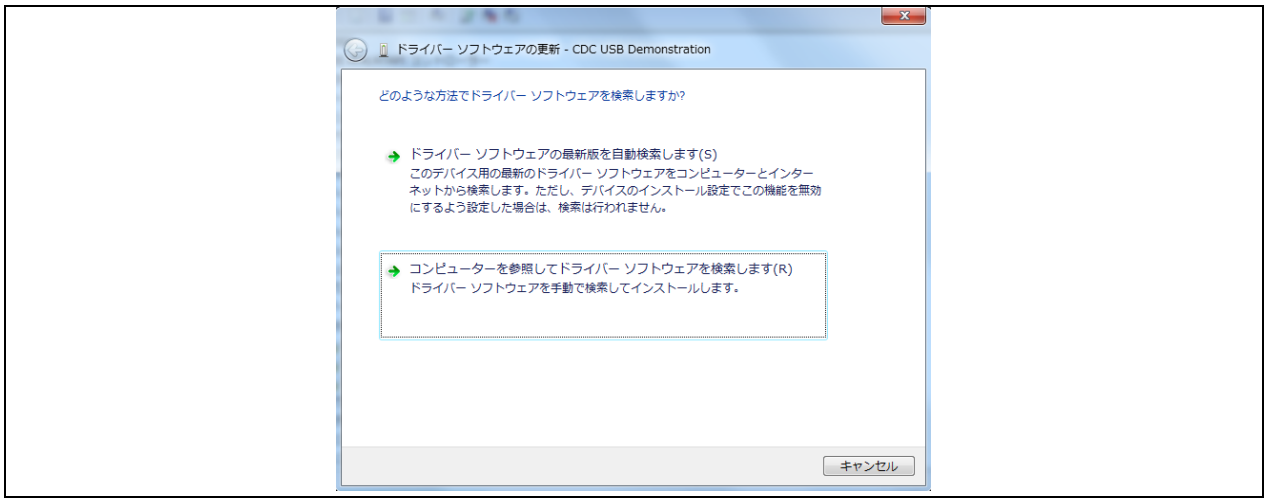


Figure 5-1　新しいハードウェアの検索ウィザード

(3). 《次の場所で最適のドライバーソフトウェアを検索します》を選択します。

“参照（R）”をクリックして“CDC_Demo.inf”の存在するフォルダを指定し，“次へ（N）”をクリックしてください。

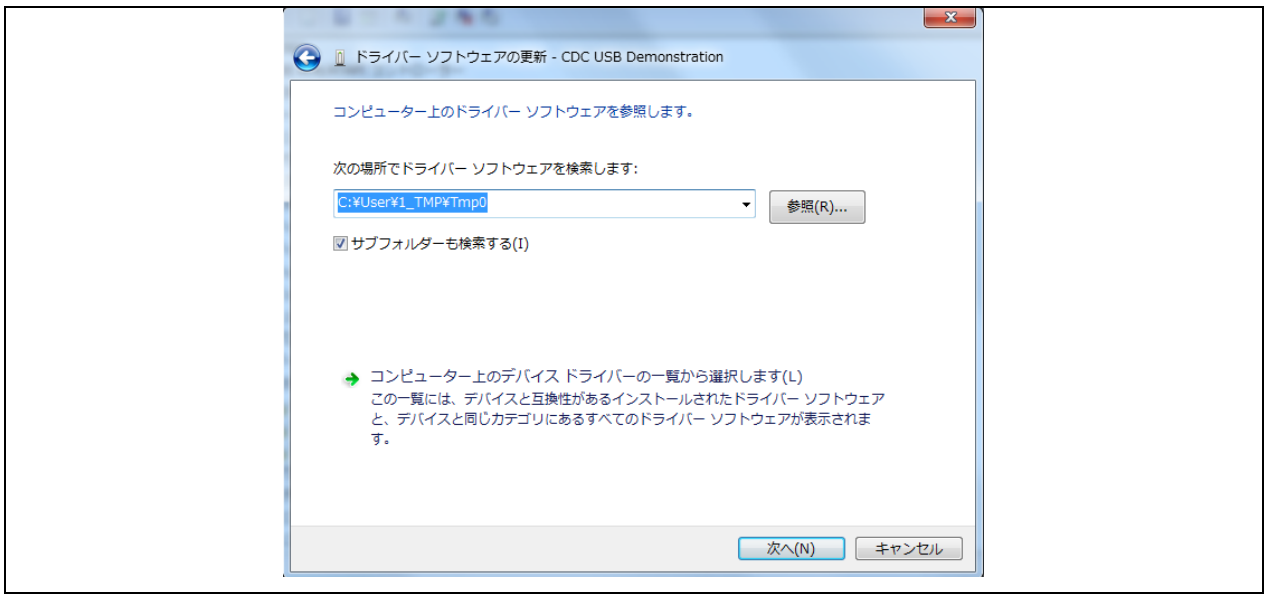


Figure 5-2　ドライバの場所の選択

Note:

CDC_Demo.inf ファイルは、パッケージ内の"r_usb_pcdc¥utilities"フォルダに格納されています。

(4). 次のインストール確認画面が表示される場合は，“このドライバーソフトウェアをインストールします (I) ”をクリックしてください。

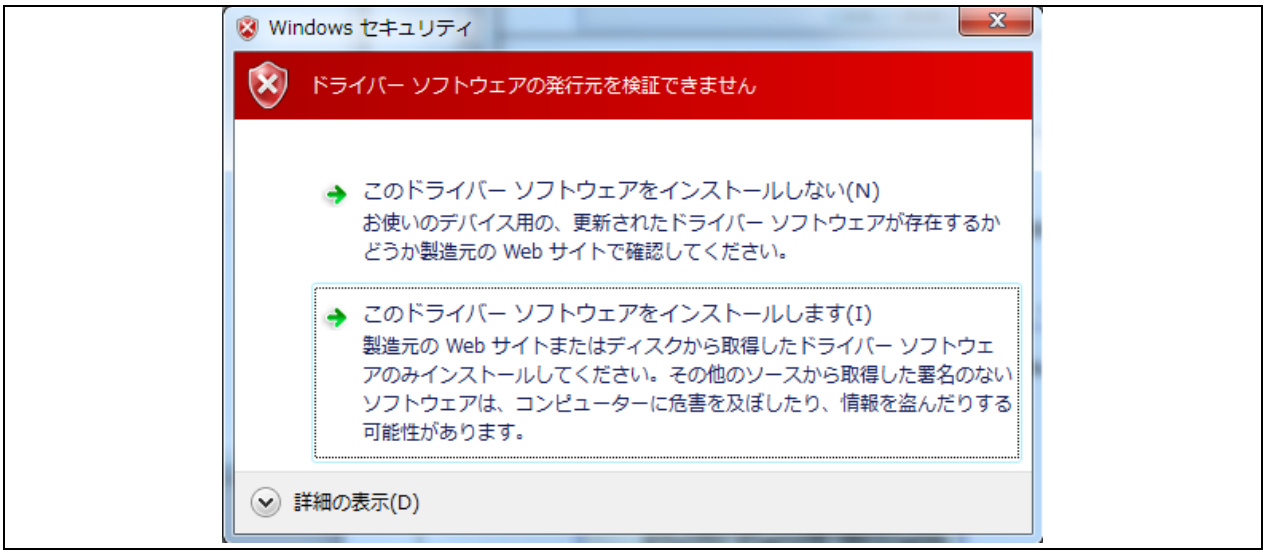


Figure 5-3　インストール確認

(5). 次のウインドウが表示されたら, CDC ドライバのインストールは完了です。”閉じる“ をクリックしてください。

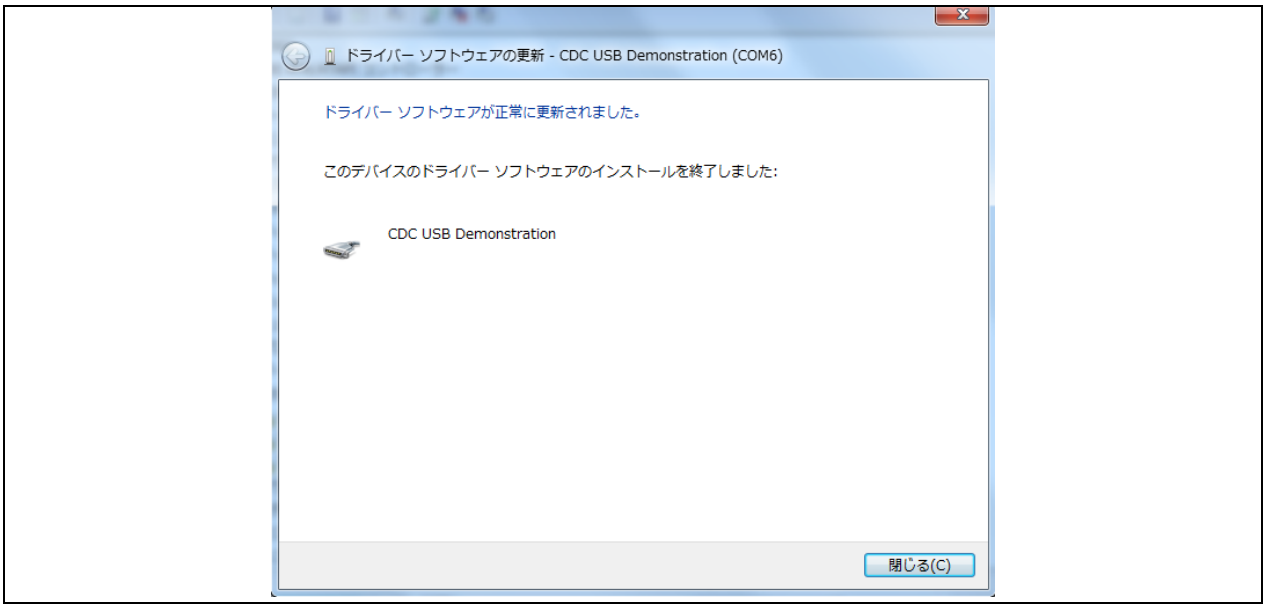


Figure 5-4　インストール完了

6. クラスドライバ概要

6.1 クラスリクエスト（ホストからデバイスへの要求）

PCDC がサポートしているクラスリクエストをTable 6-1に示します。

Table 6-1 対応する CDC クラスリクエスト

リクエスト	コード	説明
SetLineCoding	0x21	通信回線設定を行う（通信速度、データ長、パリティビット、ストップビット長）
GetLineCoding	0x22	通信回線設定を通知する。
SetControlLineState	0x23	通信回線制御信号 RTS、DTR の設定を行う。

6.2 データフォーマット

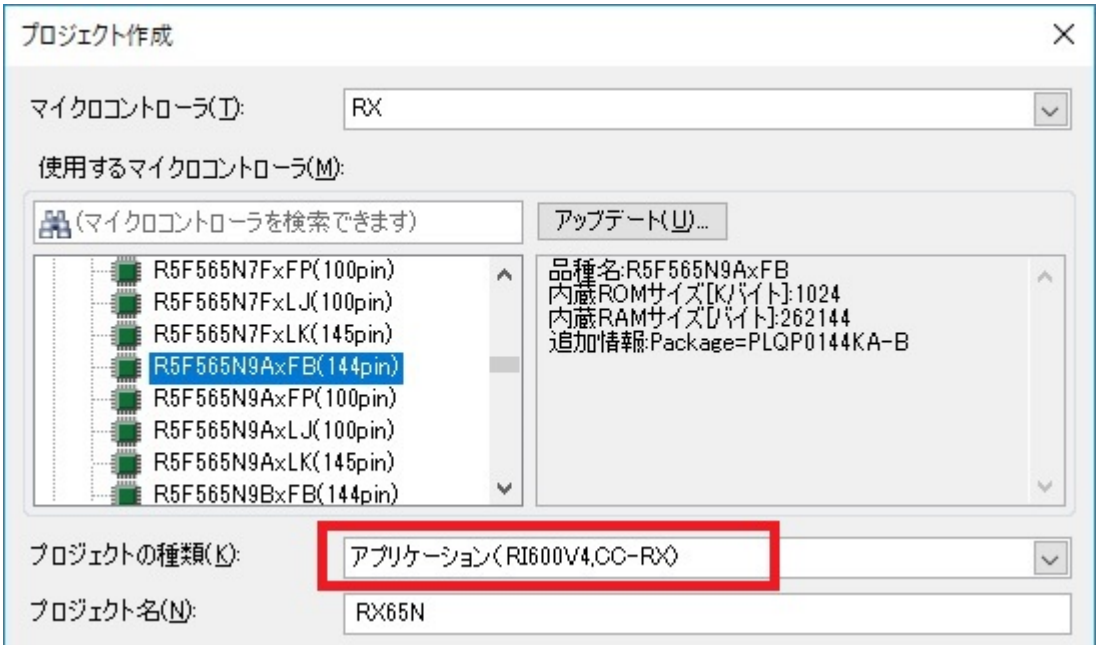
CDC データクラスでは、データフォーマットはありません。任意のデータ転送が可能です。

7. RI600V4 プロジェクトを CS+で使用する場合

パッケージ内の RI600V4 用プロジェクトは CS+をサポートしていません。RI600V4 用プロジェクトを CS+で使用する場合、以下の手順に従って CS+用のプロジェクトを作成する必要があります。

7.1 CS+上で新規プロジェクトを作成

プロジェクトの種類には、「アプリケーション(RI600V4, CC-RX)」を選択してください。



7.2 スマートコンフィグレータを起動

1. クロック設定 (「クロック」タブを選択)

(1). USB クロック(UCLK)に 48MHz が設定されるよう関連クロックを設定してください。

(2). メインクロックの発振安定時間(赤枠)を"61"に変更してください。

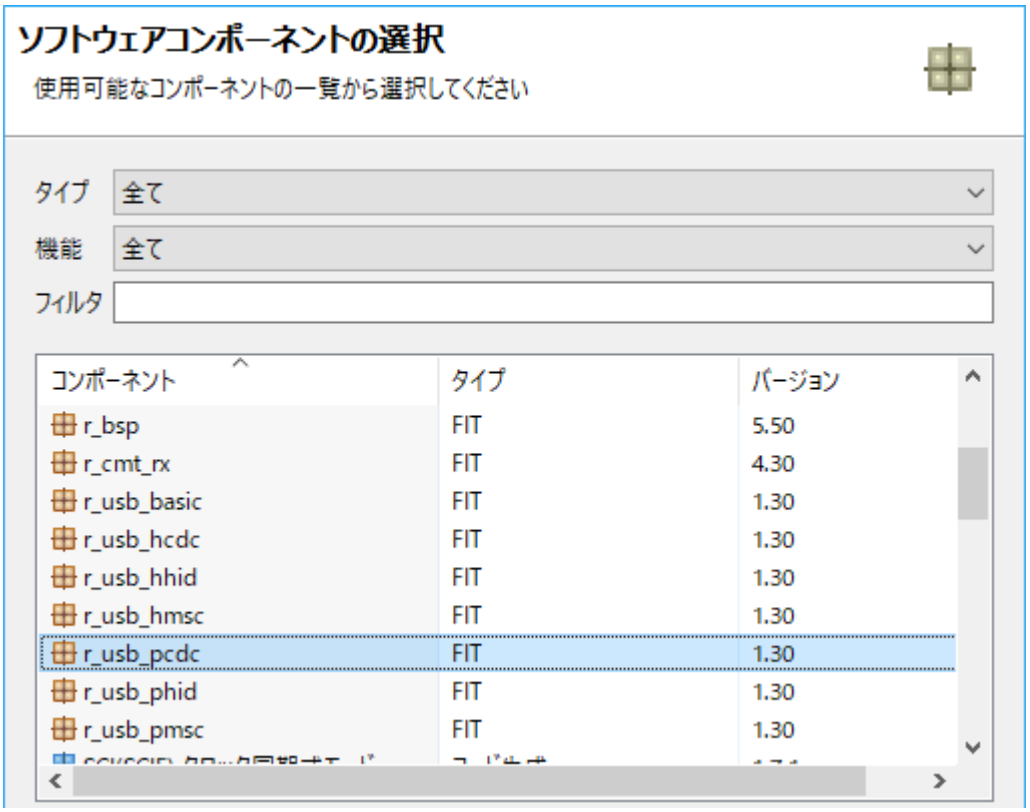
2. コンポーネント設定 (「コンポーネント」タブを選択)

(1). USB FIT モジュールをインポート

r_usb_pcdc モジュールを選択し、「終了」ボタンを押してください。r_usb_basic モジュールも同時に組み込まれます。

Note:

DTC/DMA を使用する場合、r_dtc_rx/r_dmaca_rx モジュールも選択してください。

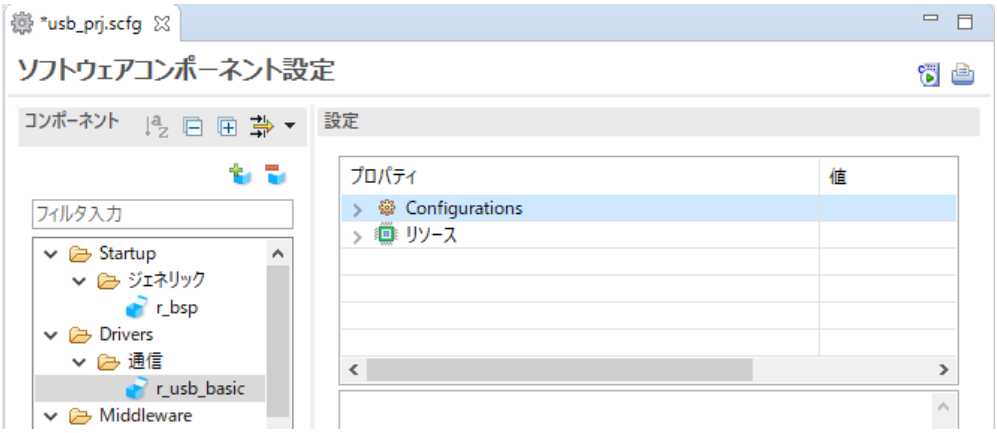


(2). コンフィグレーション

a. r_bsp

DTC 使用時、Heap size を変更してください。設定値は DTC FIT モジュールのドキュメントを参照してください。

b. r_usb_basic

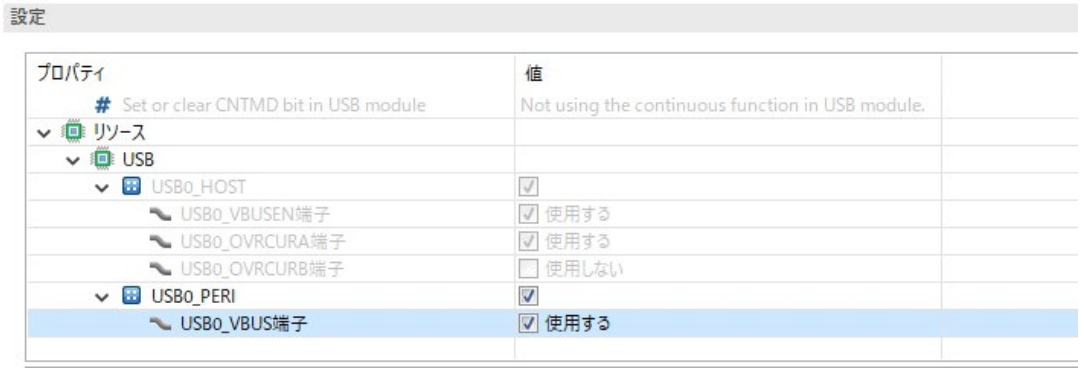


(a). Configurations

USB Basic Host and Peripheral Driver Firmware Integration Technology アプリケーションノート(ドキュメント No.R01AN2025)の「コンフィグレーション」章を参照いただきますようお願いします。

(b). リソース

USBx_VBUS 端子をチェックしてください。

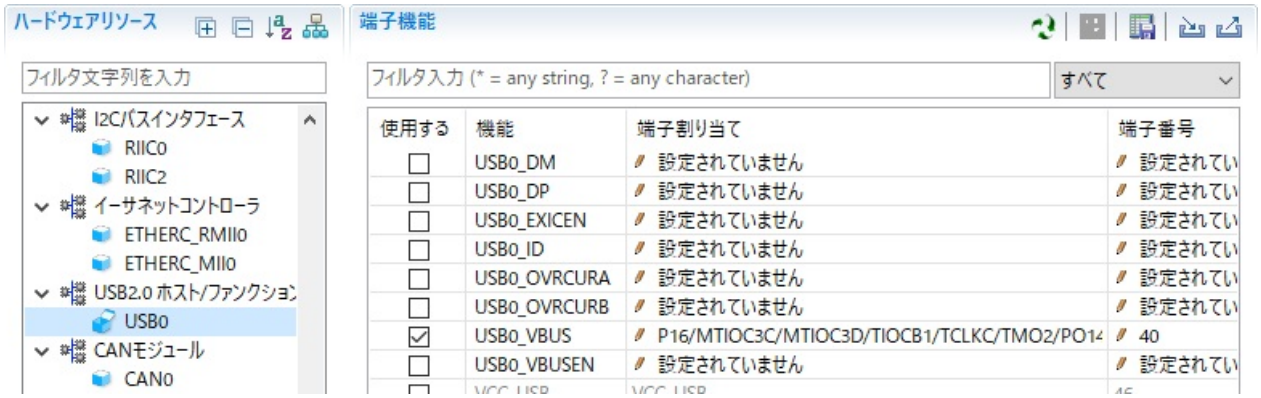


c. r_usb_pcdc

USB Peripheral Communications Devices Class Driver (PCDC) Firmware Integration Technologyアプリケーションノート(ドキュメントNo. R01AN2030)の「コンフィグレーション」章を参照いただきますようお願いします。

3. 端子設定 (「端子」タブを選択)

お客様のシステムに合った USB 端子のポート選択を行ってください。

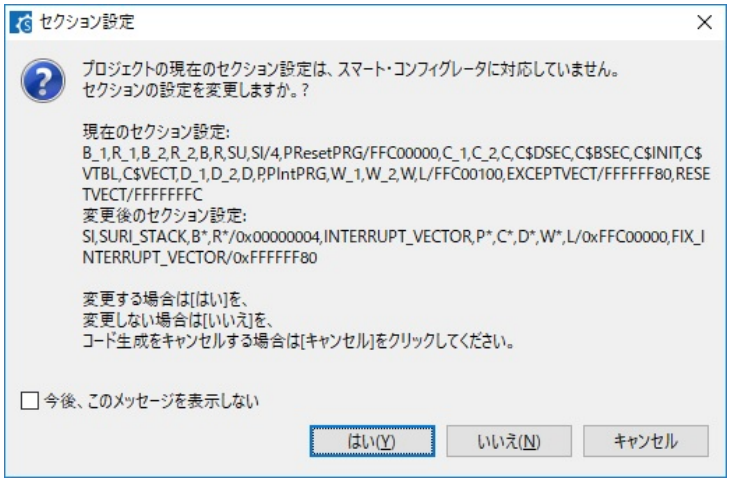


4. コード生成

「コードの生成」ボタンをクリックすると、スマートコンフィグレータは<ProjectDir>¥src¥smc_gen フォルダに USB FIT モジュールのソースコードおよび端子設定のコードを生成します。

Note:

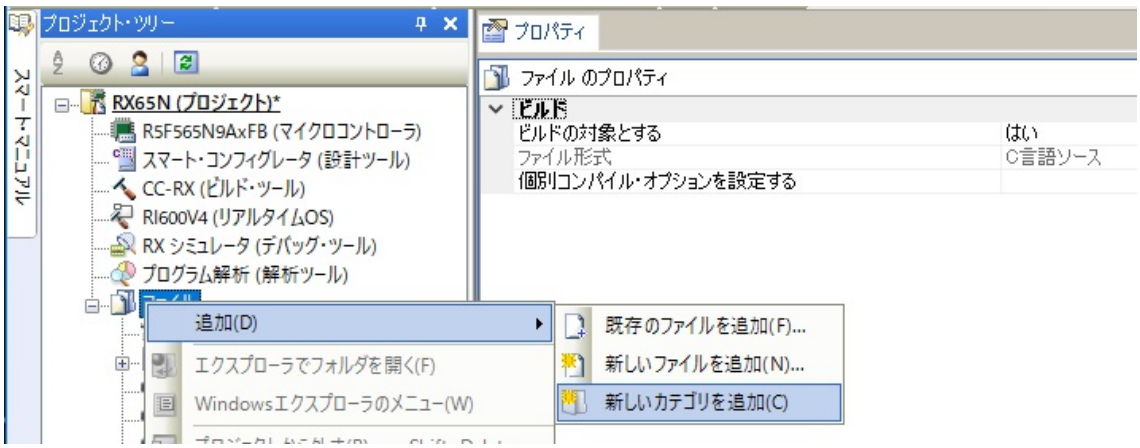
以下のダイアログが出力されますので、「はい(Y)」を選択してください。



7.3　アプリケーションプログラムおよびコンフィグレーションファイルの追加

本パッケージ内の demo_src フォルダを<ProjectDir>¥src フォルダにコピーしてください。

1. 本パッケージ内の demo_src フォルダを<ProjectDir>¥src フォルダにコピーしてください。
2. 本パッケージ内の RI600V4 用コンフィグレーションファイル(.cfg ファイル)を<ProjectDir>フォルダにコピーしてください。
3. プロジェクトツリー内の「ファイル」を選択、右クリック。「追加」→「新しいカテゴリを追加」を選択し、アプリケーションプログラムを格納するカテゴリを作成してください。次に「既存のファイルを追加」を選択し、上記2でコピーしたアプリケーションプログラムおよびコンフィグレーションファイルを登録してください。



Note:

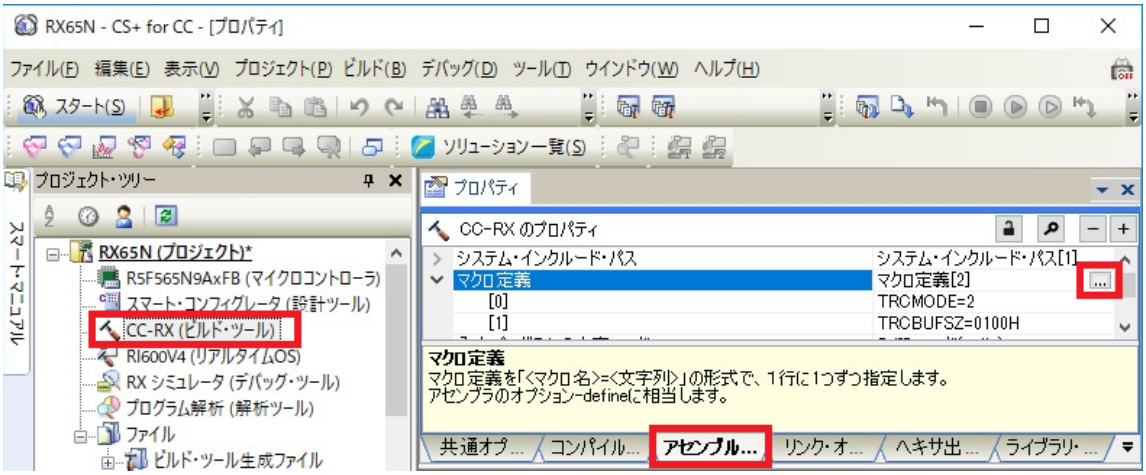
新規に生成された<ProjectDir>フォルダ内の task.c ファイルおよび sample.cfg ファイルを削除してください。

7.4　マクロ定義削除

新規生成したプロジェクトには以下のマクロが定義されていますので、これらのマクロ定義を削除してください。

CC-RX(ビルド・ツール) -> 「アセンブルオプション」タブを選択し、以下のマクロ定義を削除。

1. TRCMODE = 2
2. TRCBUFSZ = 0100H



7.5　ビルド実行

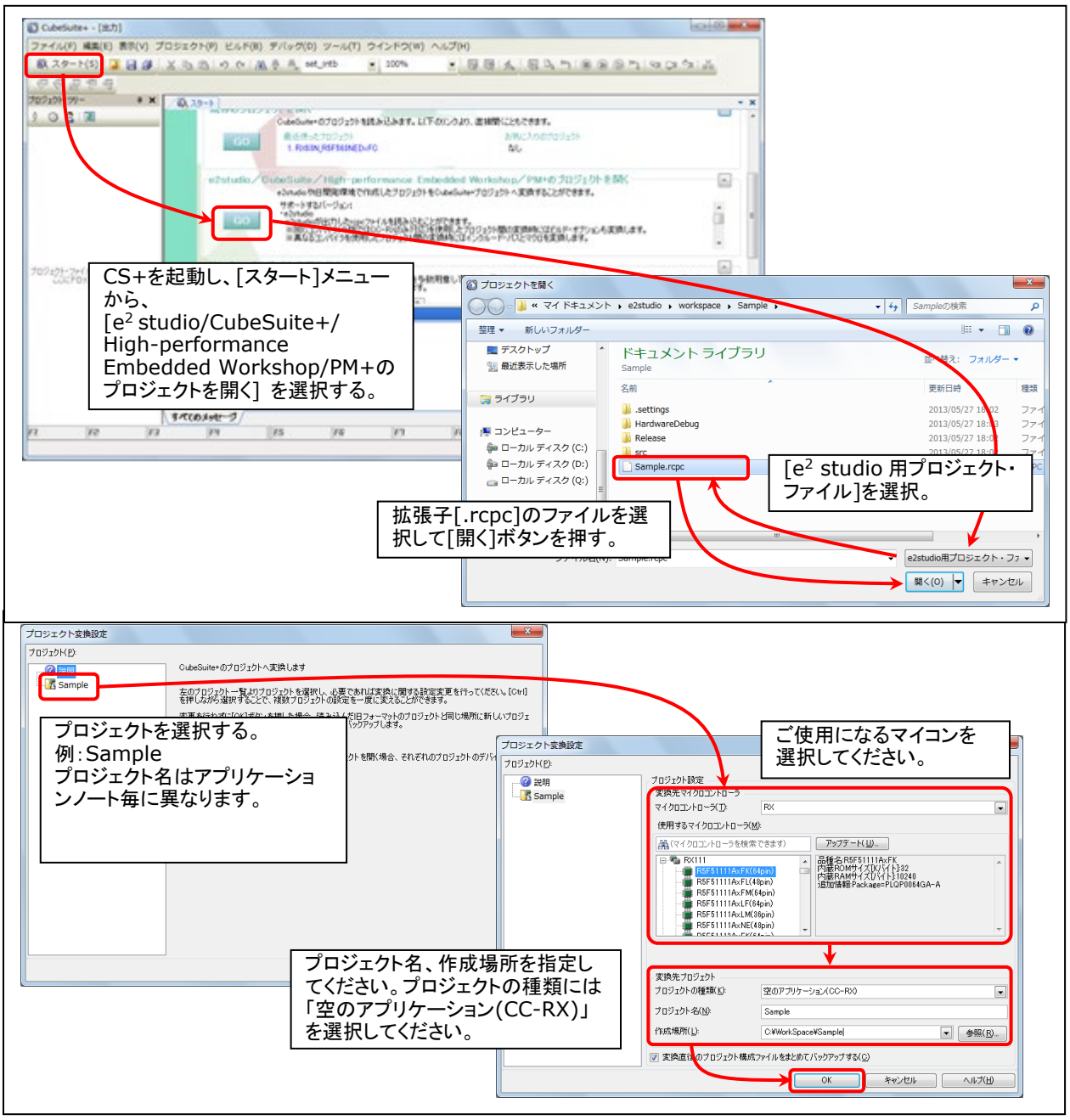
ビルドを実行し、実行プログラムを生成してください。

8. e² studio 用プロジェクトを CS+で使用する場合

PCDC のプロジェクトは、統合開発環境 e² studio で作成されています。PCDC を CS+で動作させる場合は、下記の手順にて読み込んでください。

[Note]

1. 「プロジェクト変換設定」ウィンドウ内の「変換直前のプロジェクト構成ファイルをまとめてバックアップする」のチェックを外してください。
2. RI600V4 をご使用の場合、以下の方法をサポートしていません。「**7. RI600V4プロジェクトをCS+で使用する場合**」を参照してください。



ホームページとサポート窓口

ルネサス エレクトロニクスホームページ
http://japan.renesas.com/

お問合せ先
http://japan.renesas.com/contact/

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	Oct 16, 2014	—	初版発行
1.10	Dec 26, 2014	—	RX71M を対象デバイスに追加。
1.11	Sep 30, 2015	—	RX63N/RX631 を対象デバイスに追加。
1.20	Sep 30, 2016	—	1. 対象デバイスに RX65N/RX651 を追加 2. DMA 転送をサポート 3. USB Host and Peripheral Interface Driver アプリケーションノートに対応
1.21	Mar 31, 2017	—	USB Basic driver のリビジョンを Up しました。
1.22	Sep 30, 2017	—	RX65N/RX651-2M をサポート
1.23	Mar 31, 2018	—	USB Basic driver のリビジョンを Up しました。
1.24	Dec 28, 2018	—	リアルタイム OS をサポート
1.25	Apr 16, 2019	—	RX66T/RX72T を対象デバイスに追加。
1.27	Jul 31, 2019	—	1. RX72M を対象デバイスに追加。 2. 対象デバイスから RX63N を削除しました。
1.30	Mar 1, 2020	—	1. リアルタイム OS(uITRON:RI600V4)をサポートしました。 2. 対象デバイスに RX72N/RX66N を追加
1.31	Mar 1, 2021	—	対象デバイスに RX671 を追加
1.40	Jun 30, 2022	—	Azure RTOS(USBX)をサポートしました。

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 静電気対策
   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品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、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、CMOS 製品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。
2. 電源投入時の処置
   電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、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「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。
5. クロックについて
   リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
6. 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 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス（予約領域）のアクセス禁止
   リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
8. 製品間の相違について
   型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 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。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、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品または本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 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。
4. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
6. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   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、産業用ロボット等

   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合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いますが、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、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等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、お客様の機器・システムとしての出荷保証を行ってください。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、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
10. 当社製品の環境適合性等の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。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. 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・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
13. 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禁じます。
14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

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直接的、間接的に支配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。

(Rev.5.0-1　2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。