R01AN0555JJ0215
Rev.2.15
Mar 28, 2016

Renesas USB MCU

USB Peripheral Communications Device Class Driver (PCDC) using Basic Mini Firmware

要旨

本資料は、Renesas USB MCU の USB Basic Mini Firmware を使用した USB Peripheral Communications Device Class Driver (PCDC)のアプリケーションノートです。

動作確認デバイス

RL78/G1C, RL78/L1C, R8C/3MU, R8C/34U, R8C/3MK, R8C/34K

動作確認デバイスと同様の USB モジュールを持つ他の MCU でも本プログラムを使用することができます。このアプリケーションノートのご使用に際しては十分な評価を行ってください。

なお、本プログラムは Renesas Starter Kit 上で動作確認を行っています。

目次

1. はじめに

本アプリケーションノートは、USB-BASIC-F/W (1.2 章を参照) を使用した USB Peripheral Communications Device Class Driver (PCDC) および、サンプルアプリケーションに関して記述しています。

1.1 機能と特長

PCDC は、USB Communication Device Class specification (CDC)の Abstract Control Model に準拠し、USB Host と通信を行うことができます。

本クラスドライバは弊社の USB Basic Mini Firmware と組み合わせて使用することを前提にしています。

1.2 関連ドキュメント

1. Universal Serial Bus Revision 2.0 specification
2. USB Class Definitions for Communications Devices Revision 1.2
3. USB Communications Class Subclass Specification for PSTN Devices Revision 1.2
   [http://www.usb.org/developers/docs/]
4. Renesas USB MCU ユーザーズマニュアル ハードウェア編
5. Renesas USB MCU USB Basic Mini Firmware アプリケーションノート
6. USB Peripheral Communications Device Class Driver (PCDC)インストレーションガイド for Basic Mini Firmware
7. FIT SCI Asynchronous Mode Module Application Note (Document No. R01AN1667EU)
   ルネサス エレクトロニクスホームページ　より入手できます。

・ ルネサス エレクトロニクスホームページ
   【http://japan.renesas.com/】
・ USB デバイスページ
   【http://japan.renesas.com/usb/】

1.3 用語と略語

本文書で使用される用語と略語は、以下のとおりです。

API	:	Application Program Interface
APL	:	Application program
ACM	:	Abstract Control Model. This is the USB interface subclass used for virtual COM ports, based in the old V.250 (AT) command standard. See PSTN below
CDC	:	Communications devices class
CDCC	:	Communications Devices Class － Communications Class Interface
CDCD	:	Communications Devices Class － Data Class Interface
CPD	:	Serial Communication Port Driver
cstd	:	USB-BASIC-F/Wの Peripheral & Host共通関数のprefix
Data Transfer	:	Generic name of Control transfer, Bulk transfer and Interrupt transfer
PCD	:	Peripheral control driver of USB-BASIC-F/W
PCDC	:	Communications Devices Class for peripheral
PCDCD	:	Peripheral Communications Devices Class Driver
PP	:	プリプロセス定義
pstd	:	USB-BASIC-F/WのPeripheral関数のprefix
PSTN	:	Public Switched Telephone Network. Contains the ACM (above) standard. See also Chapter 1.2
SCI	:	Serial Communication Interface
SW1/SW2/SW3	:	RSKに実装された3つのスイッチ
USB	:	Universal Serial Bus
USB-BASIC-FW	:	USB Basic Mini Firmware (Peripheral & Host USB Basic Firmware(USB low level) for Renesas USB MCU)
タスク	:	処理の単位
スケジューラ	:	タスク動作を簡易的にスケジューリングするもの
スケジューラマクロ	:	スケジューラ機能を呼び出すために使用されるもの
データ転送	:	Control転送、Bulk転送、Interrupt転送の総称

1.4 本書の読み方

本書は章の順番通りに読み進める必要はありません。はじめにサンプルプログラムの内容を確認し、ユーザ個別のソリューションに必要な関数およびインタフェースの情報をお読みください。

4.3 章にソース一覧を掲載しています。MCU 固有ソースは、"\devicename\src\HwResource"にあります。アプリケーションに必要なファイルを確認してください。

ユーザ独自のソリューションを作成するためにはアプリケーションの変更が必要です。5 章はペリフェラル CDC アプリケーションの動作を説明しています。

すべてのコードモジュールはタスクに分割されます。タスク間でメッセージの受け渡しが行われていることを予めご理解ください。関数(タスク)の実行順序はスケジューラが決定します。このため重要なタスクに優先権を持たせることができます。また、タスクに登録されたコールバックメカニズムを使用することで、各タスクは並列処理（ノンブロッキング）で動作します。タスクのメカニズムは 1.2 章の"Basic Mini Firmware Application Note"で説明しています。PCDC のタスクについては 4.4 章を参照してください。

2. デバイスクラスドライバの登録

ユーザが作成したクラスドライバは、USB-BASIC-F/W に登録することで USB デバイスクラスドライバとして機能します。*r_usb_pcdc_apl.c* ファイル内の *usb_papl_registration()* 関数を参考に USB-BASIC-F/W にクラスドライバを登録してください。詳細は、"Basic Mini Firmware Application Note"を参照してください。

3. 動作確認済環境

3.1 コンパイラ

動作確認を行ったコンパイラは以下の通りです。

a. CA78K0R コンパイラ　V.1.71
b. CC-RL コンパイラ V.1.01
c. IAR C/C++ Compiler for RL78 version 2.10.4
d. KPIT GNURL78-ELF v15.02
e. C/C++ Compiler Package for M16C Series and R8C Family V.6.00 Release 00

3.2 評価ボード

動作確認を行った評価ボードは以下の通りです。

a. Renesas Starter Kit for RL78/G1C (型名: R0K5010JGC001BR)
b. Renesas Starter Kit for RL78/L1C (型名: R0K50110PC010BR)
c. R8C/34K Group USB Peripheral 評価ボード(型名: R0K5R8C34DK2PBR)

4. ソフトウェア構成

4.1 モジュール構成

Figure 4-1 は PCDC に関連したモジュール構成を示します。Table 4-1 は個々のモジュール概要を提供します。

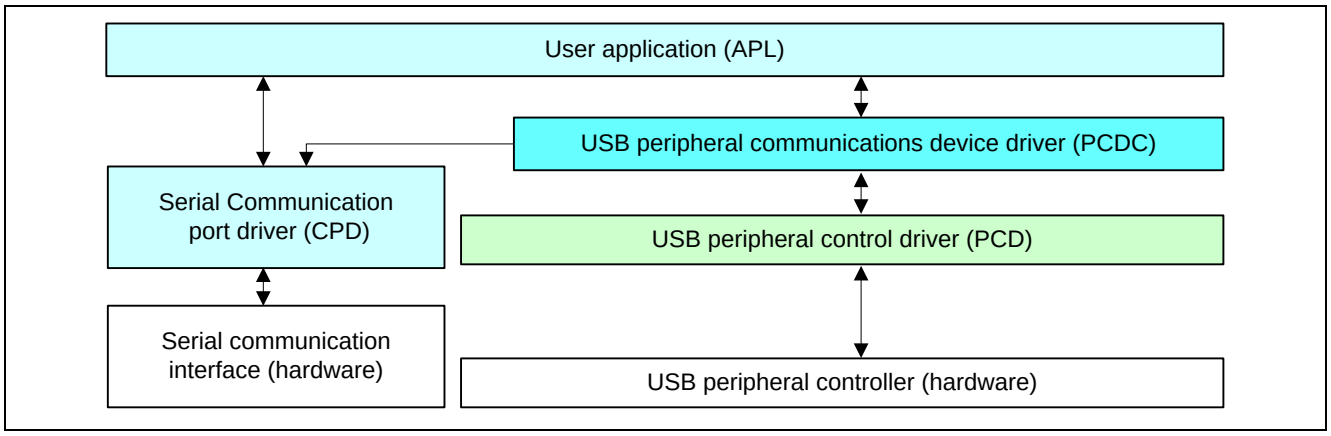


Figure 4-1 モジュール構成図

Table 4-1 モジュール機能説明

モジュール名	機能概要	備考
APL	ユーザアプリケーションのサンプルプログラムです。 ホストの COM ポートから送信されたデータをホストへとエコーバックします。	ユーザ作成
PCDC	登録されたデバイスクラスドライバは接続ホストからの要求に従って USB-BASIC-F/W に転送要求を行います。	
USB-BASIC-F/W	USB ペリフェラルコントロールドライバです。 (ハードウェア制御とデバイスステート管理を行います。)	

4.2 アプリケーションプログラム機能概要

ユーザアプリケーション (APL) と PCDC は、それぞれのスケジューラから呼び出されるタスクとして動作します。

APL 及び PCDC は、PCD を介してホストとのデータ通信を行います。

4.3 ファイルとフォルダの構成

4.3.1 フォルダ構成

以下に本デバイスクラスで提供するファイルのフォルダ構成を示します。

各 MCU と評価基板に依存するソースコードはそれぞれのハードウェアリソースフォルダ (***devicename***src**HwResource*)にあります。

```
workspace
    ＋[ RL78 / R8C ]
      ＋[ CCRL / CS+ / IAR / e2 studio / HEW ]
        ＋ [ RL78G1C / RL78L1C / R8C3MK / R8C3MU / R8C34K / R8C34U ]
            ＋ UART                                          UART ビルド結果
            ＋ ECHO                                          ECHO ビルド結果
            ＋ src
              ＋――― PCDC [ Communication Device  Class driver ]      Table 4-2 参照
              │          ＋――― inc                          CDC ドライバ共通ヘッダファイル
              │          ＋――― src                          CDC ドライバ
              ＋―――SmplMain [サンプルアプリケーション]
              │          ＋――― APL                          サンプルアプリケーション
              ＋―――USBSTDFW [全ての USB ドライバに共通な基本ファームウェア]
              │          ＋――― inc                          USB ドライバ共通ヘッダファイル
              │          ＋――― src                          USB ドライバ
              ＋――― HwResource [ MCU 初期化等のハードウェアアクセス層 ]
                         ＋――― inc                          H/W リソースヘッダファイル
                         ＋―――src                          H/W リソース
```

[Note]

a. CS+フォルダ下には、CA78K0R コンパイラ用のプロジェクトが格納されています。

b. e2 studio フォルダ下には、KPIT GNU コンパイラ用のプロジェクトが格納されています。

c. CS+上で CC-RL コンパイラをご使用になる場合は、「10 e2 studio 用プロジェクトを CS+で使用する場合」を参照してください。

4.3.2 CDC ファイルリスト

Table 4-2 は、PCDC で供給されるファイル構成を示します。

Table 4-2 ファイル構成

フォルダ名	ファイル名	説明
PCDC/inc	r_usb_pcdc_driver.h	USB ペリフェラル CDC ユーザ定義マクロ
PCDC/inc	r_usb_pcdc_define.h	PCDC 型定義、マクロ定義
PCDC/inc	r_usb_pcdc_api.h	PCDC API 関数プロトタイプ宣言
PCDC/src	r_usb_pcdc_api.c	PCDC API 関数
PCDC/src	r_usb_pcdc_driver.c	PCDC ドライバ関数
SmplMain	main.c	メインループ処理
SmplMain/APL	r_usb_pcdc_echo_apl.c	エコーモード用サンプルアプリケーションプログラム
	r_usb_pcdc_uart_apl.c	シリアル-USB 変換モード用サンプルアプリケーション プログラム
	r_usb_pcdc_descriptor.c	サンプルアプリケーション用 PCDC ディスクリプタテーブル

4.4 システムリソース

PCDC をスケジューラに登録して使用するためのタスク ID とタスク優先度定義を Table 4-3 に示します。

これらについては、*r_usb_ckernelid.h* ヘッダファイルで定義します。

Table 4-3 リソース定義

スケジューラ登録タスク	説明	備考
USB_PCDC_TSK	**PCDC** (R_usb_pcdc_Task) Task ID: USB_PCDC_TSK Task priority: 1	
USB_PSMP_TSK	**APL** (usb_psmpl_MainTask) Task ID: USB_PCDCSMP_TSK Task priority: 2	
USB_PCD_TSK	**PCD** (R_usb_pstd_PcdTask) Task ID: USB_PCD_TSK Task priority: 0	
メールボックス ID / デフォルト受信タスク	**メッセージ名称**	**備考**
USB_PCDC_MBX / USB_PCDC_TSK	PCDC -> PCDC / APL -> PCDC mailbox ID	
USB_PSMP_MBX / USB_PSMP_TSK	PCDC -> APL mailbox ID	
USB_PCD_MBX / USB_PCD_TSK	PCD task mailbox ID	

5. ペリフェラル CDC サンプルアプリケーションプログラム (APL)

この章は、ペリフェラル CDC サンプルアプリケーション (APL)を説明します。

5.1 動作環境

Figure 5-1 に本ソフトウェアの動作環境を示します。

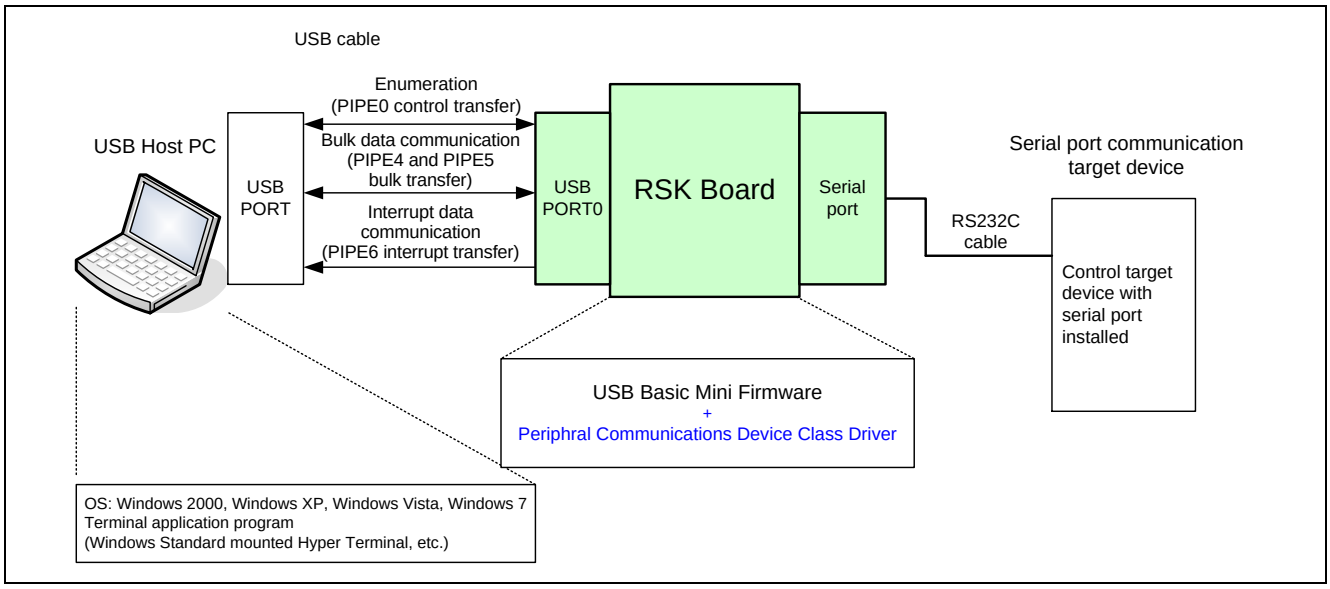


Figure 5-1 動作環境例

ホスト PC として Windows PC を使用する場合、システム定義ファイル (reference\cdc_inf\CDC_Demo.inf)のインストールが必要です。システム定義ファイルは、 USB Peripheral Communications Device Class Driver (PCDC)で設定した Vendor ID (VID) と Product ID (PID) に合わせて編集する必要があります。テキストエディタ等でドライバファイルの以下の箇所を編集してください。

[Model.NTx86]

%STRING_MODEL%=CDC, USB￥VID_0000&PID_0000←4 桁の数値の部分を 16 進数 4 桁で編集します

[Model.NTamd64]

%STRING_MODEL%=CDC, USB￥VID_0000&PID_0000←4 桁の数値の部分を 16 進数 4 桁で編集します

※「￥」マークは半角

VID が 0x1234 で、PID が 0x5678 の場合

[Model.NTx86]

%STRING_MODEL%=CDC, USB￥VID_1234&PID_5678

[Model.NTamd64]

%STRING_MODEL%=CDC, USB￥VID_1234&PID_5678

※「￥」マークは半角

ペリフェラルデバイスの PID, VID 定義は 、ファイル ***devidcename****src**SmplMain**APL**r_usb_echo_apl_descriptor.c* で、それぞれ USB_VENDORID 及び USB_PRODUCTID で定義します。

5.2 アプリケーションプログラム (APL) 概要

アプリケーションプログラムは、以下の 2 つのモードで動作します。各モードのアプリケーションプログラムのファイルは異なります。モードの選択については、5.2.3 章を参照してください。

5.2.1 シリアル-USB 変換モード

シリアル-USB 変換モードでは、USB-シリアル変換機として動作します。本モードは、USB ホストから受信したデータを UART ポートに送ります。また、UART ポートに送られてきたデータを USB COM ポートに送ります。

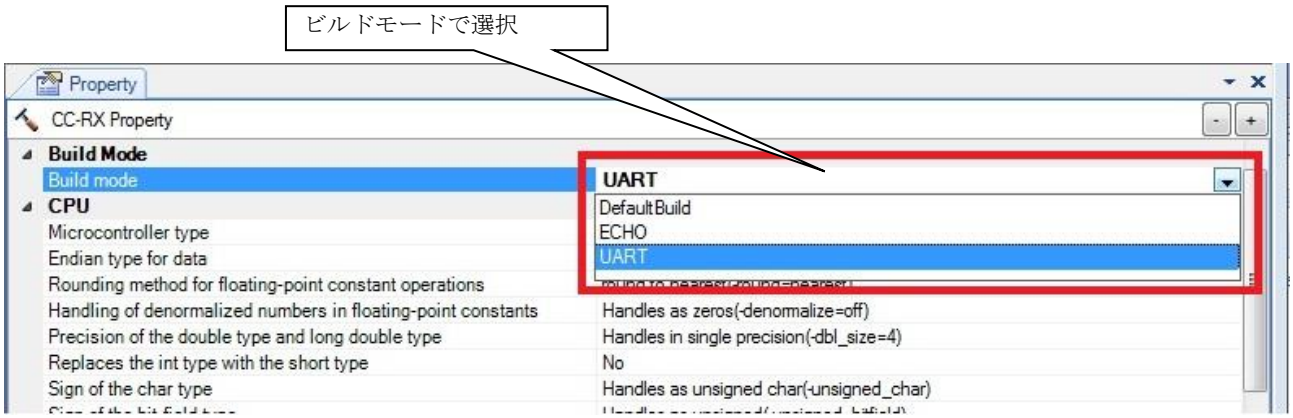
5.2.2 エコーモード

エコーモードは、USB ホストに USB ホストから受け取ったデータを折り返し送信します。 UART ポートは、使用しません。

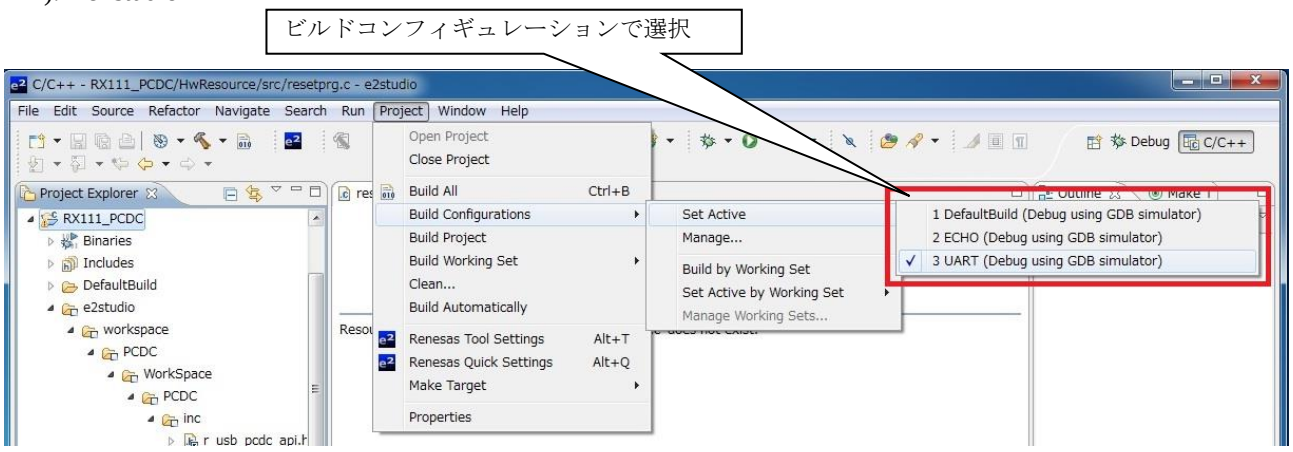
5.2.3 シリアル-USB 変換モード/エコーモードの選択

MCU ごとにサポートしている開発環境を立ち上げた後、開発環境上で各モードを選択します。

1). CS+



2). e² studio



5.3 APL メッセージ

アプリケーションモジュール (APL) は、メールボックス USB_PSMP_MBX からメッセージを受け取ります。APL は“Table 5-1”で示されるメッセージ処理を行います。

Table 5-1 APL 受信メッセージリスト

メッセージ	処理内容
USB_PCDC_RX_COMP	USB 受信完了時に呼び出されるコールバック関数 "usb_psmpl_RxCB"
USB_PCDC_TX_COMP	USB 送信完了時に呼び出されるコールバック関数 “usb_psmpl_TxCB”
USB_PCDC_STATUS_TX_COMP	クラスノーティフィケーション”SerialState”送信完了 “usb_psmpl_state_notification”
USB_PCDC_PERIODIC	サンプルアプリケーション周期起動処理 “usb_psmpl_periodic_request”

5.4 APL 関数一覧

APL の関数一覧を Table 5-2 に示します。

Table 5-2 サンプルアプリケーション関数一覧

関数名	説明
usb_cstd_task_start	タスクスタート処理
usb_pcdc_task_start	ペリフェラル USB 用の各種スタートアップ処理
usb_psmpl_driver_registration	PCDC ドライバ登録
usb_psmpl_open	PCDC オープン関数
usb_psmpl_close	PCDC クローズ関数
usb_apl_task_switch	タスクスイッチングループ
usb_psmpl_MainTask	デモサンプルアプリケーションメイン処理
usb_psmpl_RxCB	USB 受信完了コールバック処理
usb_psmpl_TxCB	USB 送信完了コールバック処理
usb_psmpl_GetRcvDataCnt	USB 受信データ数取得処理
usb_psmpl_change_device_state	Device State Callback Check
usb_psmpl_ReceiveDataStart	Start the date receive request for Host
usb_psmpl_LineCodingInitial	Line Coding Initial processing
usb_psmpl_DummyFunc	Dummy function for the callback
usb_psmpl_state_notification	Callback function for notifying the serial state
usb_psmpl_class_request_callback	Callback function for receiving the class request
usb_psmpl_periodic_request	Application program cyclic start request
usb_psmpl_uart_callback	Callback function for UART driver
usb_psmpl_serial_state_process	Serial State processing
usb_psmpl_is_connected	Return the USB connection state

5.5 APL データフロー

サンプルアプリケーションプログラムのデータフローを以下に示します。

5.5.1 シリアル-USB 変換モード

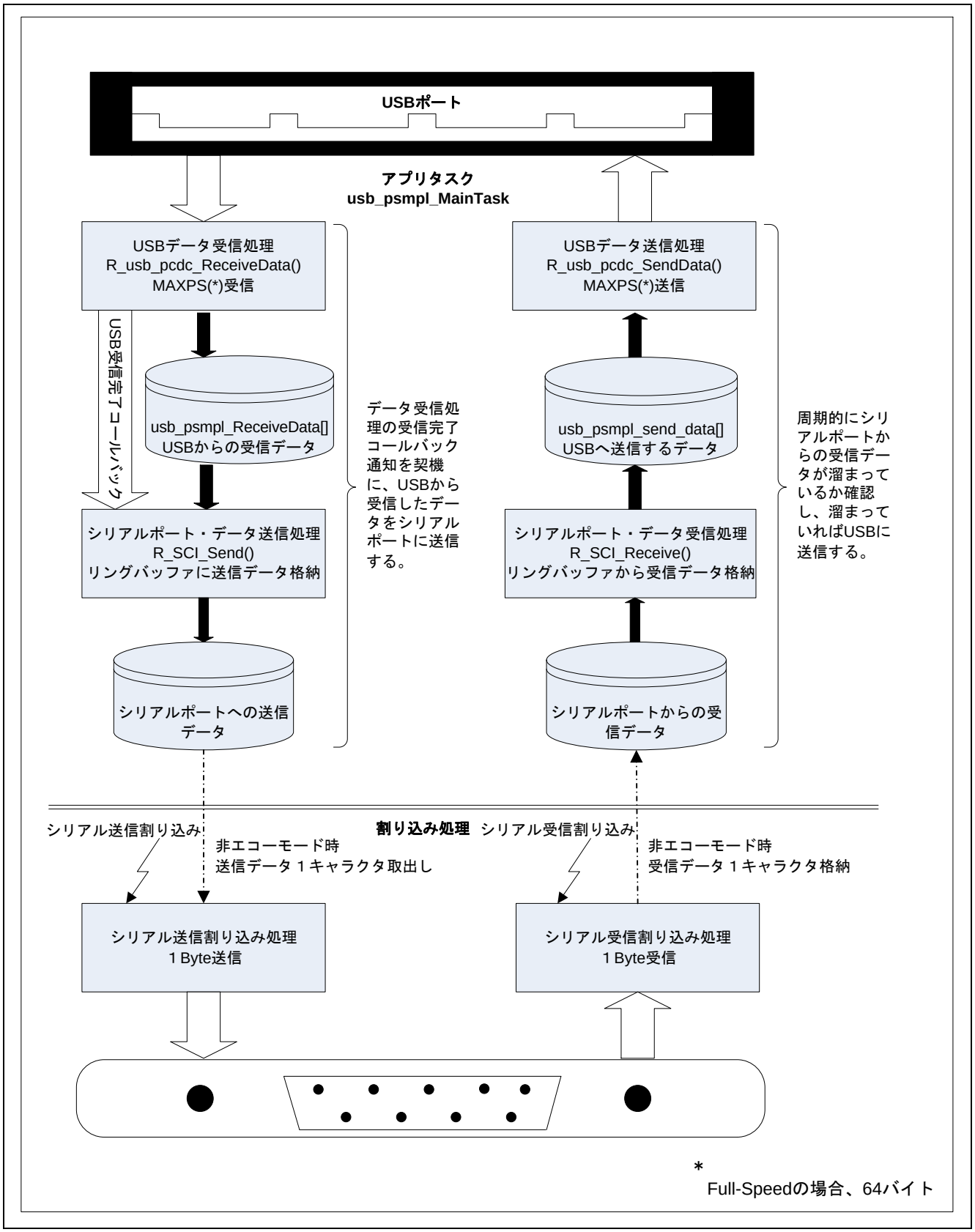


Figure 5-2 APL データフロー(シリアル-USB 変換モード)

5.5.2 エコーモード

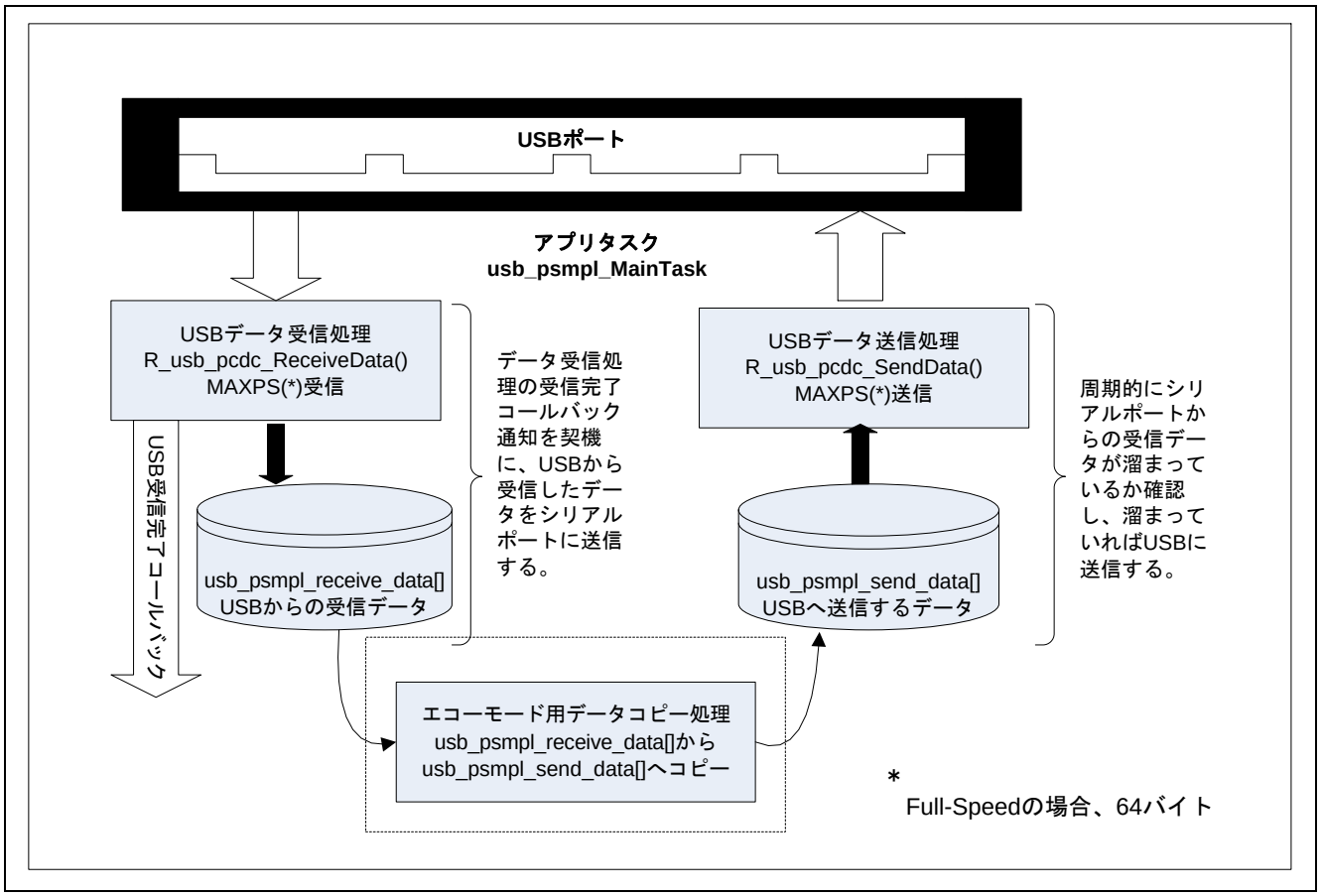


Figure 5-3 APL データフロー(エコーモード)

5.6 シーケンス

以下は、モジュール APL (application)、PCDC (device class driver)、 PCD (USB device HW control) の間のタイムシーケンスです。

5.6.1 シリアル-USB 変換モード

1. CDC ホストからのデータ受信、シリアルポート送信動作

CDC ホストからのデータ受信を行い、受信したデータをシリアルポートへ送信するシーケンスを以下に記します。

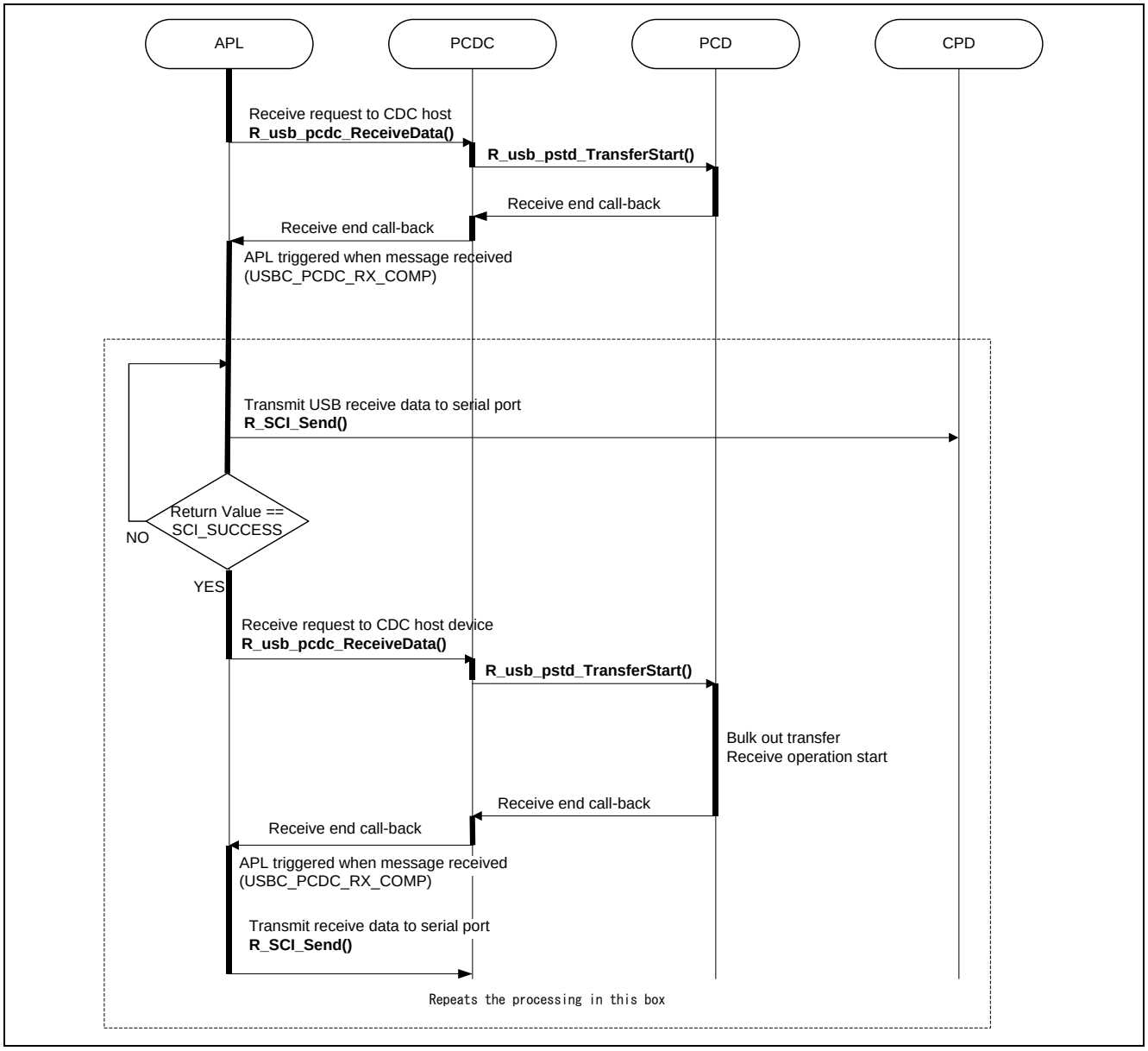


Figure 5-4 CDC ホストからのデータ受信・シリアルポート送信動作のシーケンス

2. シリアルポート受信、CDC ホストへの送信動作

シリアルポートから受信したデータを、USB ホストへ送信するシーケンスを以下に記します。

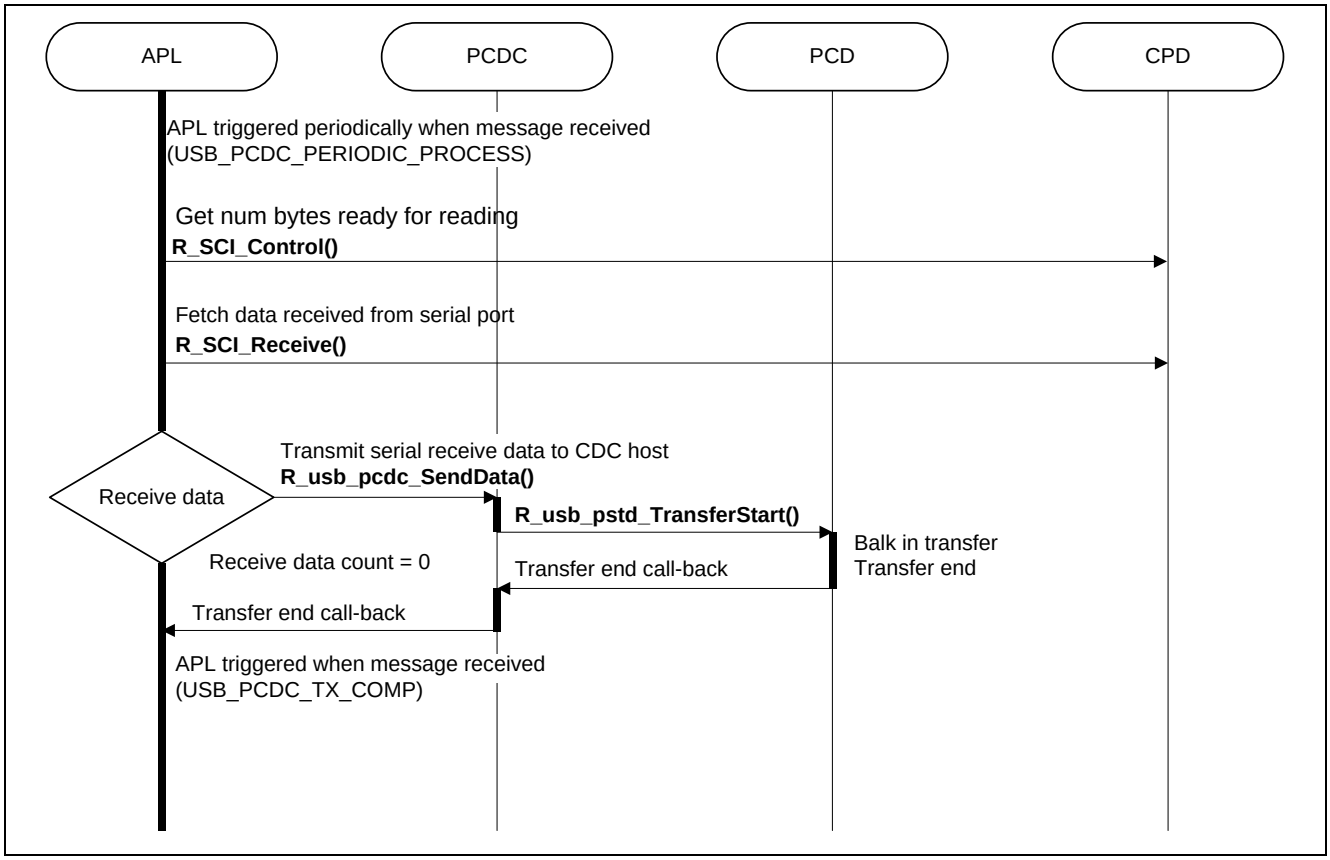


Figure 5-5 シリアルポート受信・CDC ホスト送信動作のシーケンス

3. シリアルエラー動作

シリアル受信エラーを検出し、USB ホストへクラスノーティフィケーション（SerialState）を送信するシーケンスを以下に記します。

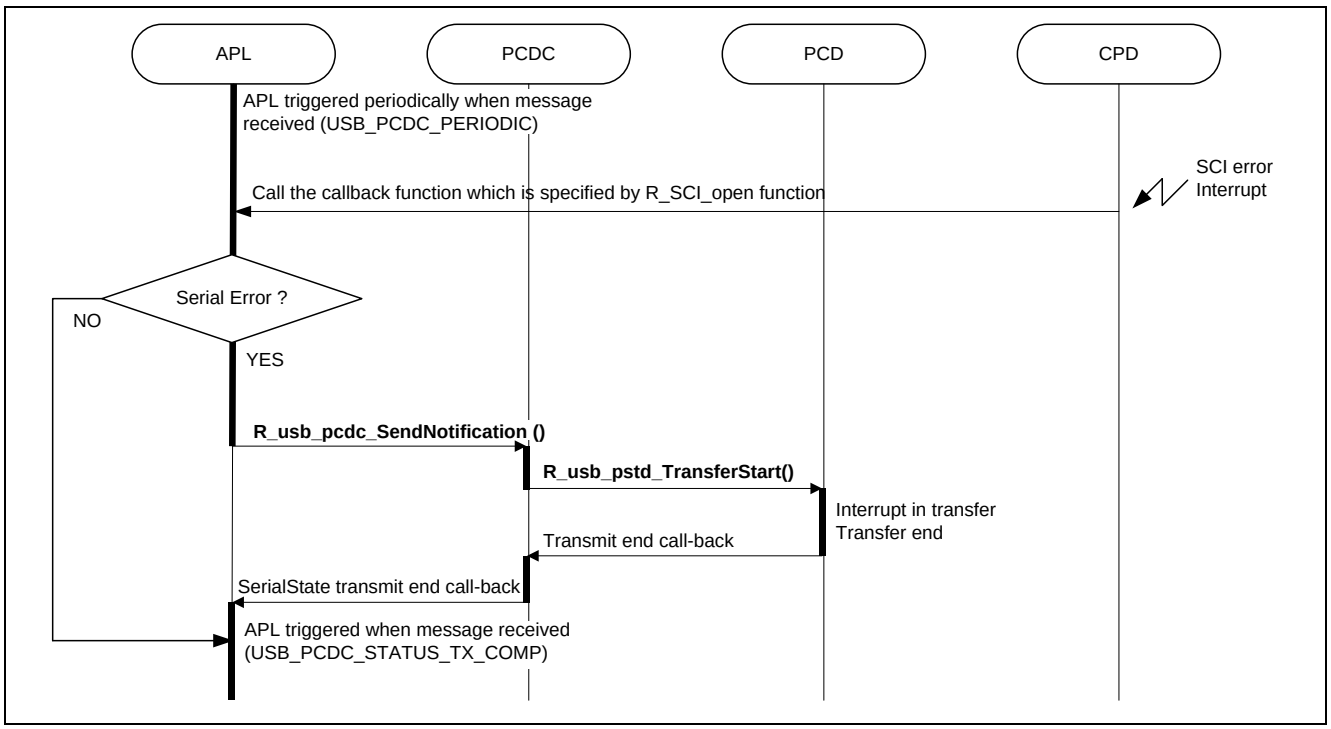


Figure 5-6 シリアルエラー動作のシーケンス

5.6.2 エコーモード

エコーモード動作で、USB ホストから受信したデータを USB ホストへ折り返し送信するシーケンスを、Figure 5-7 に示します。

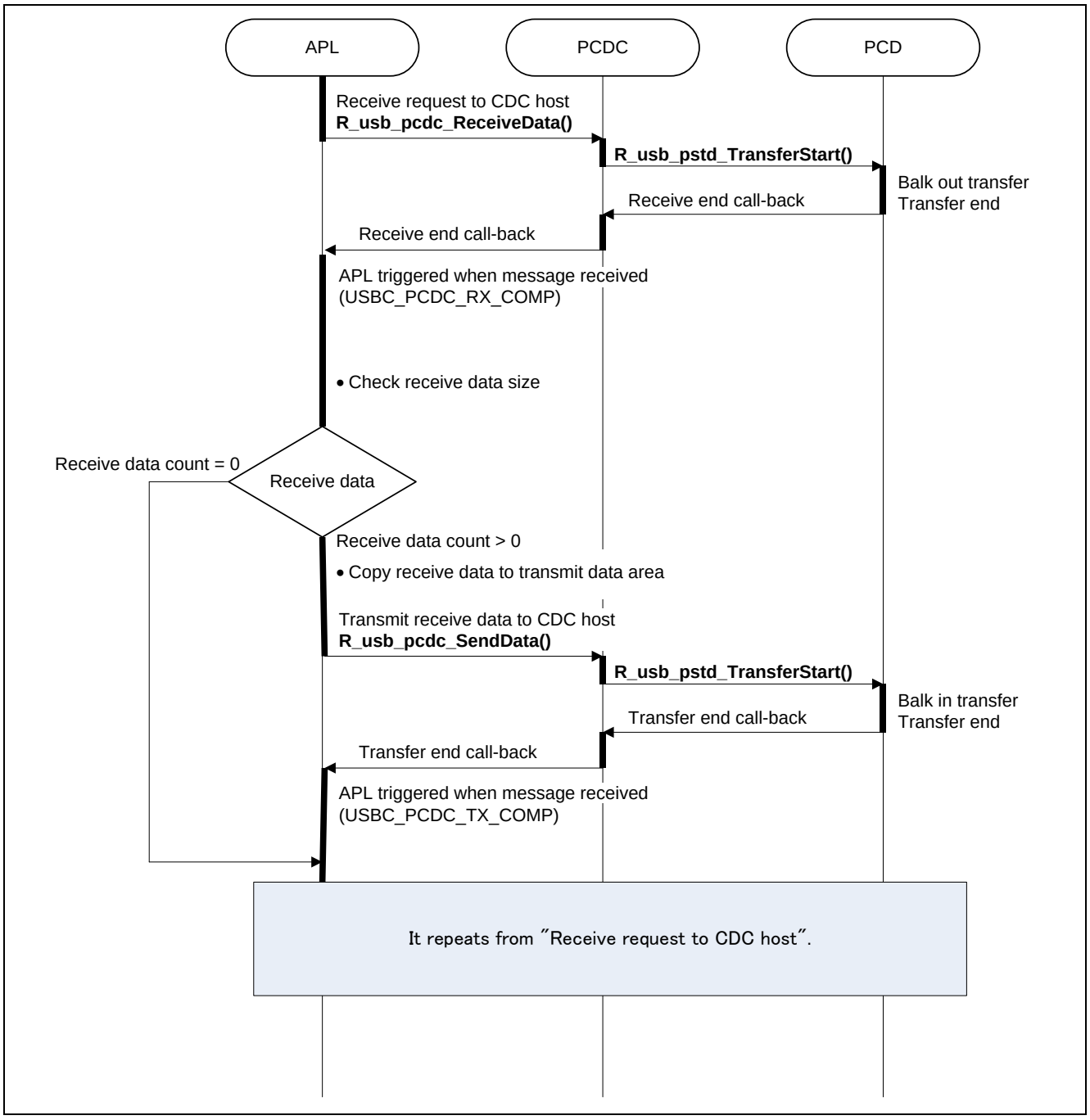


Figure 5-7 Echo Mode シーケンス

5.7 APL 処理動作フロー

処理概要フローでの USB-シリアル変換処理及び、USB ループバックでの主な処理経路を以下に記します。

【USBーシリアル変換処理経路】

・USB→シリアル（UART)

①データ受信処理 R_usb_pcdc_ReceiveData()

②シリアルポート送信処理 R_SCI_Send()

以降①・②の繰返し

・シリアル（UART)→USB

④シリアル受信データ取り出し処理 R_SCI_Receive()

⑤シリアル送信処理 R_usb_pcdc_SendData()

以降④・⑤の繰返し

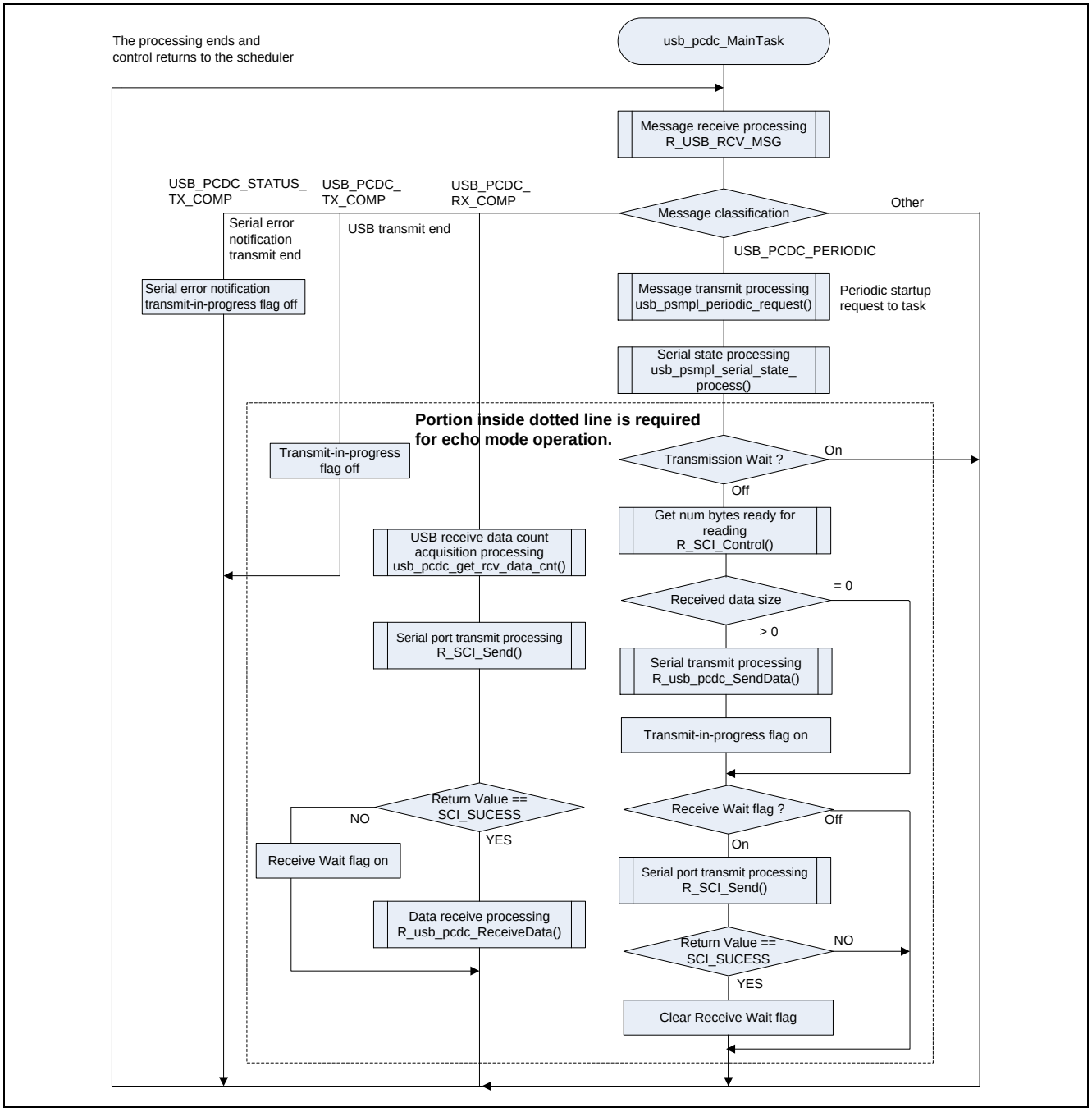


Figure 5-8 APL 処理概要フロー（シリアル-USB 変換モード）

【USB ループバック（エコーモード）変換処理経路】

① USB データ受信処理開始 *usb_psmpl_ReceiveDataStart()*

② USB データ送信処理 *R_usb_pcdc_SendData()*

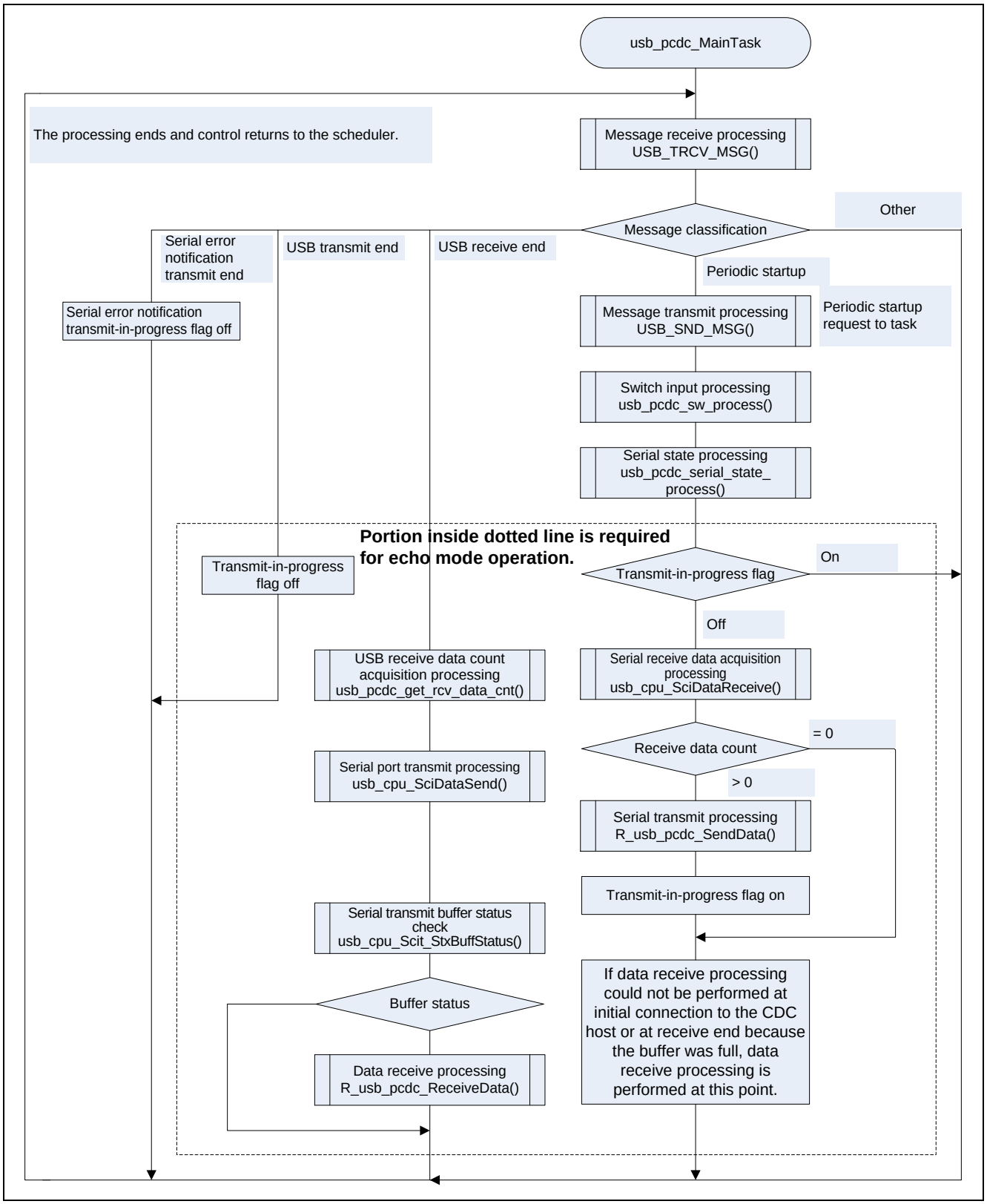


Figure 5-9 APL 処理概要フロー (エコーモード)

6. Communications Device Class (CDC)

6.1 基本関数

このソフトウェアは、コミュニケーションデバイスクラス仕様の Abstract Control Model サブクラスに準拠しています。

PCDC の主な機能は以下のとおりです。

1. USB ホストからの機能照会に対する応答
2. USB ホストからのクラスリクエストに対する応答
3. USB ホストとのデータ通信
4. USB ホストへのシリアル通信エラー報告

6.2 Abstract Control Model 概要

Abstract Control Model サブクラスは、USB 機器と従来のモデム（RS-232C 接続）との間を埋める技術で、従来のモデムを使用するアプリケーションプログラムが作成可能です。

以下に本ソフトウェアでサポートするクラスリクエスト・クラスノーティフィケーションを記します。

6.2.1 クラスリクエスト(ホスト→デバイスへの要求)

本 S/W で対応しているクラスリクエストを Table 6-1 に示します。

Table 6-1 CDC クラスリクエスト

リクエスト	コード	説明	対応
SendEncapsulatedCommand	0x00	プロトコルで定義された AT コマンド等を送信する。	No
GetEncapsulatedResponse	0x01	SendEncapsulatedCommand で送信したコマンドに対するレスポンスを要求する。	No
SetCommFeature	0x02	機器固有の 2 バイトコードや、カントリー設定の禁止／許可を設定する。	No
GetCommFeature	0x03	機器固有の 2 バイトコードや、カントリー設定の禁止／許可状態を取得する。	No
ClearCommFeature	0x04	機器固有の 2 バイトコードや、カントリー設定の禁止／許可設定をデフォルト状態に戻す。	No
SetLineCoding	0x20	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）	**Yes**
GetLineCoding	0x21	通信回線設定状態を取得する。	**Yes**
SetControlLineState	0x22	通信回線制御信号 RTS、DTR の設定を行う。	**Yes**
SendBreak	0x23	ブレーク信号の送信を行う。	No

Abstract Control Model リクエストについては、 “USB Communications Class Subclass Specification for PSTN Devices”, Revision 1.2 の Table 11 : Requests - Abstract Control Model を参照してください。

6.2.2 クラスリクエストのデータフォーマット

本クラスドライバソフトが対応するクラスリクエストのデータフォーマットを以下に記します。

(1) SetLineCoding

UART 回線設定を行う為にホストが送信するクラスリクエストです。

SetLineCoding データフォーマットを以下に示します。

Table 6-2 SetLineCoding フォーマット

bmRequestType	bRequest	wValue	wIndex	wLength	Data
0x21	SET_LINE_CODING (0x20)	0x00	0x00	0x07	Line Coding Structure Table 6-3 参照

Table 6-3 Line Coding Structure フォーマット

Offset	Field	Size	Value	説明
0	DwDTERate	4	Number	データ端末の速度 (bps)
4	BcharFormat	1	Number	ストップビット 0 - 1 Stop bit 1 - 1.5 Stop bits 2 - 2 Stop bits
5	BparityType	1	Number	パリティ 0 - None 1 - Odd 2 – Even
6	BdataBits	1	Number	データビット (5, 6, 7, 8)

本 S/W でサポートする設定を以下に示します。

DwDTERate : 1200bps/2400bps/4800bps/9600bps/14400bps/19200bps/38400bps/57600bps/115200bps
BcharFormat : 1Stop bit/2Stop bit
BparityType : None/Odd/Even
BdataBits : 7bit/8bit

(2) GetLineCoding

UART 回線設定状態を要求する為にホストが送信するクラスリクエストです。

GetLineCoding データフォーマットを以下に示します。

Table 6-4 SetLineCoding フォーマット

bmRequestType	bRequest	wValue	WIndex	wLength	Data
0xA1	GET_LINE_CODING (0x21)	0x00	0x00	0x07	Line Coding Structure Table 6-3 参照

(3) SetControlLineState

UART のフロー制御用信号を設定する為にホストが送信するクラスリクエストです。

本 S/W では RTS/DTR の制御をサポートしていません。

SET_CONTROL_LINE_STATE データフォーマットを以下に示します。

Table 6-5 SET_CONTROL_LINE_STATE フォーマット

bmRequestType	bRequest	WValue	wIndex	wLength	Data
0x21	SET_CONTROL_ LINE_STATE (0x22)	Control Signal Bitmap Table 6-6 参照	0x00	0x00	None

Table 6-6 Control Signal Bitmap フォーマット

Bit Position	説明
D15 ～ D2	予約 (0 にリセット)
D1	DCE の送信機能を制御 0 - RTS OFF 1 - RTS ON
D0	DTE がレディ状態かの通知 0 - DTR OFF 1 - DTR ON

6.2.3 クラスノーティフィケーション（デバイス→ホストへの通知）

本 S/W のクラスノーティフィケーション対応/非対応を Table 6-7 に示します。

Table 6-7 CDC クラスノーティフィケーション

ノーティフィケーション	コード	説明	対応
NETWORK_CONNECTION	0x00	ネットワーク接続状況を通知する	No
RESPONSE_AVAILABLE	0x01	GET_ENCAPSLATED_RESPONSE への応答	No
SERIAL_STATE	0x20	シリアル回線状態を通知する	Yes

(1) Serial State

UART ポートに状態変化を検出した場合、ホストへ状態通知を行います。

本 S/W ではオーバーランエラー、パリティエラー、フレーミングエラー検出をサポートしています。状態通知は正常状態からエラー検出した場合に行います。エラーを連続検出しても状態通知を連続送信しません。

SerialState データフォーマットを以下に示します。

Table 6-8 SerialState フォーマット

bmRequestType	bRequest	wValue	wIndex	wLength	Data
0xA1	SERIAL_STATE (0x20)	0x00	0x00	0x02	UART State bitmap Table 6-9 参照

Table 6-9 UART state bitmap フォーマット

Bits	Field	説明	対応
D15～D7		予約	-
D6	bOverRun	オーバーランエラー検出	Yes
D5	bParity	パリティエラー検出	Yes
D4	bFraming	フレーミングエラー検出	Yes
D3	bRingSignal	着信(Ring signal)検出	No
D2	bBreak	ブレーク信号検出	No
D1	bTxCarrier	Data Set Ready: 回線が接続されて通信が可能	No
D0	bRxCarrier	Data Carrier Detect: 回線にキャリア検出	No

6.3 エンドポイント仕様

本 S/W では Table 6-10 で示されるエンドポイントを使用していいます。

Table 6-10 エンドポイント仕様

bEndpointAddress		bmAttributes	wMaxPacketSize	説明
EP No	転送方向	転送タイプ	Max パケットサイズ	
EP0	In/Out	Control	64	標準リクエスト、クラスリクエスト
EP1	In	Bulk	64 (Full Speed)	デバイスからホストへのデータ転送
EP2	Out	Bulk	64 (Full Speed)	ホストからデバイスへのデータ転送
EP3	In	Interrupt	16	デバイスからホストへの状態通知

6.4 PC 仮想 COM ポートについて(参考)

Windows OS 搭載 PC は CDC デバイスを仮想 COM ポートとして利用することが可能です。

Windows OS 搭載 PC に本 S/W を実装した RSK ボードを接続すると、エニュメレーション設定に続き、CDC クラスリクエストの GetLineCoding 及び SetControlLineState を行った後、仮想 COM デバイスとしてデバイスマネージャに登録されます。

Windows デバイスマネージャに仮想 COM ポートとして登録された後は、Windows XP 標準搭載のハイパーターミナル等のターミナルアプリで CDC デバイスとデータ通信が可能です。

ターミナルアプリのシリアルポート設定を行うことで、クラスリクエスト SetLineCoding による UART 設定が可能です。ターミナルアプリのウインドウから入力したデータ（又はファイル送信）は EP2 を使用して RSK ボードへ転送され、RSK ボード側から PC へのデータ転送は EP1 を使用して行われます。

ターミナルアプリによっては最後に受信したデータが MAX パケットサイズ（Full-Speed:64Byte,Hi-Speed:512Byte）の場合、継続するデータがあると判断して受信データをターミナルに表示しないことがあります。

MAX パケットサイズ未満のデータを受信することで、それまでに受信したデータがターミナルに表示されます。

7. 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)

7.1 基本機能

PCDC の基本機能は以下のとおりです。

(1) USB ホストとのデータ転送
(2) CDC クラスリクエストに応答
(3) コミュニケーションデバイスクラスノーティフィケーション送信サービスの提供

7.2 PCDC API 関数

Table 7-1 に PCDC API 一覧を示します。

Table 7-1 API Functions

Function Name	Description
R_usb_pcdc_LineCodingInitial	LineConding 初期化
R_usb_pcdc_SendData	USB 送信処理
R_usb_pcdc_ReceiveData	USB 受信処理
R_usb_pcdc_ClassRequest	CDC コントロール転送処理
R_usb_pcdc_task	PCDC メインタスク

R_usb_pcdc_LineCodingInitial

LineCoding 初期化

形式

usb_er_t R_usb_pcdc_LineCodingInitial (usb_pcdc_LineCoding_t *linecoding)

引数

*linecoding	LineCoding 設定データポインタ

戻り値

USB_E_OK	成功

解説

LineCoding を初期化します。

補足

―	―

R_usb_pcdc_SendData

USB 送信処理

形式

```
void  R_usb_pcdc_SendData ( uint8_t*      table,
                            usb_leng_t    size,
                            usb_cbinfo_t  complete)
```

引数

*table	転送データアドレス
size	転送サイズ
complete	処理完了通知コールバック関数

戻り値

―	―

解説

転送データアドレス Table で指定されたアドレスから、転送サイズ size 分のデータを USB 送信します。

送信完了後、コールバック関数 complete が呼出されます。

補足

1. USB 送信処理結果はコールバック関数の引数” usb_utr_t *mess”で得られます。
2. USB-BASIC-F/W のアプリケーションノートの USB 通信用構造体 (usb_utr_t 構造体) を参照してください。

使用例

```
void usb_apl_task( void )
{
  uint8_t    send_data[] = {0x01,0x02,0x03,0x04,0x05};  /* USB send data */
  uint16_t   size = 5;                                  /* Data size */

  R_usb_pcdc_SendData((uint8_t *)send_data, size, (usb_cbinfo_t)&usb_complete)
}

/* Callback function */
void  usb_complete( usb_utr_t *mess )
{
  /* Processing at the time of the completion of USB transmitting */
}
```

R_usb_pcdc_ReceiveData

USB 受信処理

形式

void	R_usb_pcdc_ReceiveData (uint8_t *Table, usb_leng_t size, usb_cbinfo_t complete)

引数

*Table	USB 通信用構造体
size	転送サイズ
complete	処理完了通知コールバック関数

戻り値

―	―

解説

USB 受信要求を行います。

USB から転送サイズ size 分のデータ受信完了、又は MAX パケットサイズ未満のデータを受信した場合、コールバック関数 complete が呼出されます。

USB 受信データは、転送データアドレス Table で指定されたアドレスで指定された領域に格納されます。

補足

1. USB 受信処理結果はコールバック関数の引数” usb_utr_t *mess”で得られます。
2. USB-BASIC-F/W のアプリケーションノートの USB 通信用構造体 (usb_utr_t 構造体) を参照してください。

使用例

```
void usb_smp_task( void )
{
  uint8_t    receive_data[64];                    /* Data buff */
  uint16_t   size = 64;                           /* Data size */

  R_usb_pcdc_ReceiveData((uint8_t *)receive_data,size,
(usb_cbinfo_t)&usb_complete)
}

/* Callback function */
void  usb_complete( usb_utr_t *mess )
{
  /* Processing at the time of the completion of USB reception */
}
```

R_usb_pcdc_ClassRequest

CDC コントロール転送処理

形式

void R_usb_pcdc_ClassRequest(usb_request_t *request, uint16_t data)

引数

*request	クラスリクエストメッセージへのポインタ	
data	コントロール転送ステージ情報	
	USB_CS_IDST	Idle or setup stage
	USB_CS_RDDS	Control read data stage
	USB_CS_WRDS	Control write data stage
	USB_CS_WRND	Control write no data status stage
	USB_CS_RDSS	Control read status stage
	USB_CS_WRSS	Control write status stage
	USB_CS_SQER	Sequence error

戻り値

― ―

解説

リクエストタイプが CDC クラスリクエストの場合、コントロール転送ステージに対応した処理を呼び出します。

本 API はデバイスクラスドライバ・レジストレーションでコントロール転送時に呼出すコールバック関数として登録します。

補足

― ―

使用例

```
void usb_apl_task( void )
{
  usb_pcdreg_t  driver;

        :
  /* Control Transfer */
  driver.ctrltrans = &R_usb_pcdc_ClassRequest;
  R_usb_pcdc_Registration(&driver);
        :
}
```

R_usb_pcdc_Task

PCDC メインタスク

形式

void	R_usb_pcdc_Task(void)

引数

―	―

戻り値

―	―

解説

PCDC 処理タスク。

アプリから要求された処理を行い、アプリに処理結果を通知します。

補足

本 API はユーザプログラムで呼び出してください。

タスクスイッチング処理から本 API がスケジュールされるように登録します。

使用例

```
void usb_apl_task_switch(void)
{
  while( 1 )
  {
    if( USB_FLGSET == R_usb_cstd_Schedule() )
    {
       /* PCD Task */
       R_usb_pstd_PcdTask();

       /* Peripheral Communications Devices Class Task */
       R_usb_pcdc_Task();

       /* Peripheral Communications Class Application Task */
       usb_pcdc_main_task();
    }
  }
}
```

7.3 ユーザ定義テーブル

PCD が使用するディスクリプタテーブルおよびパイプ情報テーブルを作成する必要があります。サンプルの *r_usb_echo_apl_descriptor.c* のファイルを参考に作成してください。

詳細は、ルネサス *USB Device USB Basic Mini Firmware ユーザーズマニュアル*をご参照ください。

8. コミュニケーションポートドライバ(CPD)

コミュニケーションポートドライバ(以降 CPD) は、RSK　UART 用シリアル通信ドライバです。

RSK 以外の H/W 環境で使用される場合は、使用される環境に合わせたシリアル通信ドライバを御用意下さい。

8.1 RL78 シリーズ

RL78 シリーズで使用しているシリアル通信ドライバは、RX シリーズで使用しているアプリケーションノート(Document No. R01AN1667EU)のドライバに準拠しています。シリアル通信ドライバの各 API および引数については、このアプリケーションノートを参照してください。

8.1.1 機能概要

1. シリアルポート H/W(SCI)の通信仕様
   (1) 回線速度 1200bps～115200bps
   (2) パリティビット（なし、偶数、奇数）
   (3) ストップビット（1 ビット、2 ビット）
   (4) データ長（7 ビット、8 ビット）

2. 送信機能
   シリアルポートへの送信動作は、送信データを CPD の有するリングバッファへ格納し、送信データエンプティ割り込みにより 1 バイトずつ送信を行います。

3. 受信機能
   シリアルポートに対する受信動作は、受信データフル割り込みにより受信したデータを CPD の有するリングバッファへ格納します。
   CPD の提供する受信データ読み出し API を使用して、受信データ処理を行ってください。

9. e² studio 用プロジェクトのセットアップ

(1). e² studio を起動してください。

※ 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). [ファイル] → [インポート]を選択してください。インポートの選択ダイアログが表示されます。

(3). インポートの選択画面で、 [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。

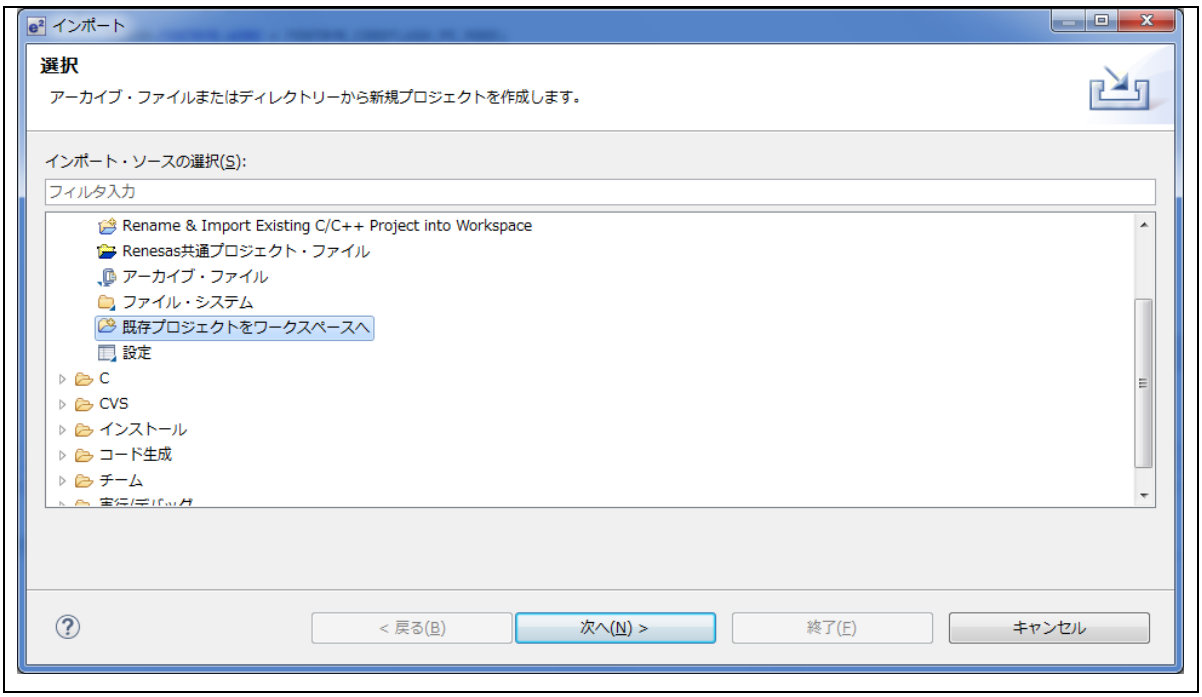


Figure 9-1 インポートの選択

(4). [ルートディレクトリの選択] の [参照] ボタンを押下して、「.cproject 」(プロジェクトファイル) が格納されたフォルダを選択して下さい。

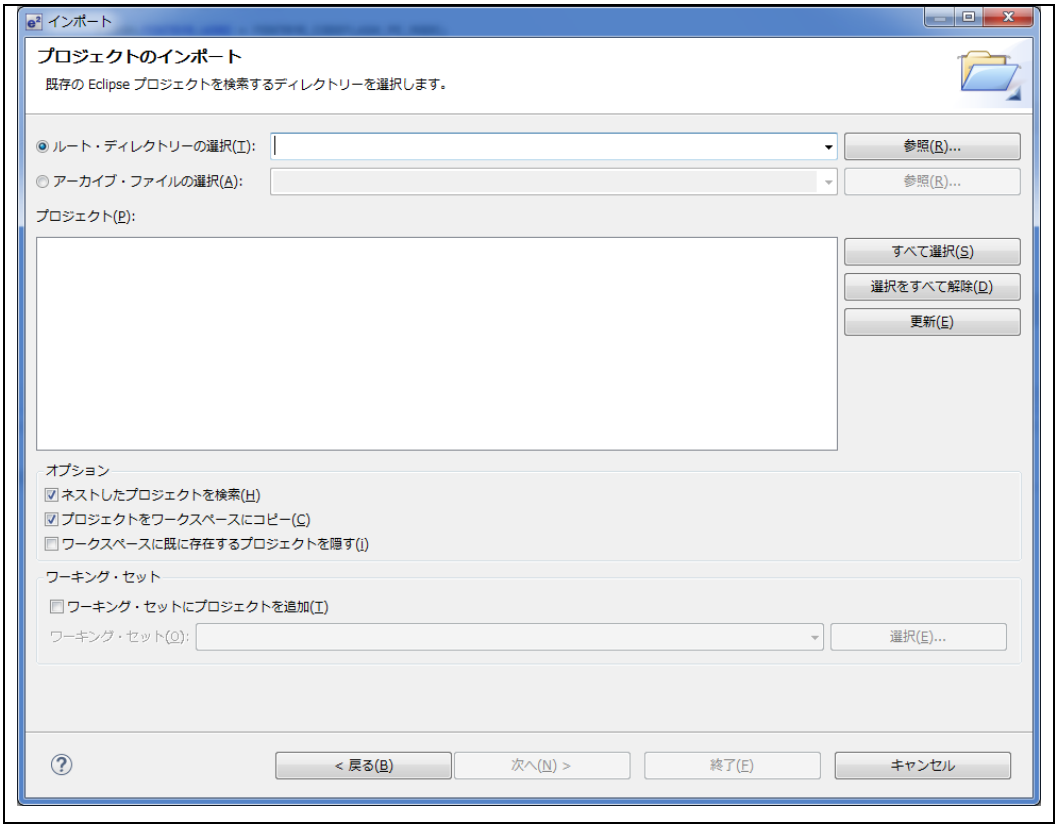


Figure 9-2　プロジェクトのインポート画面

(5). [終了]をクリック して下さい。

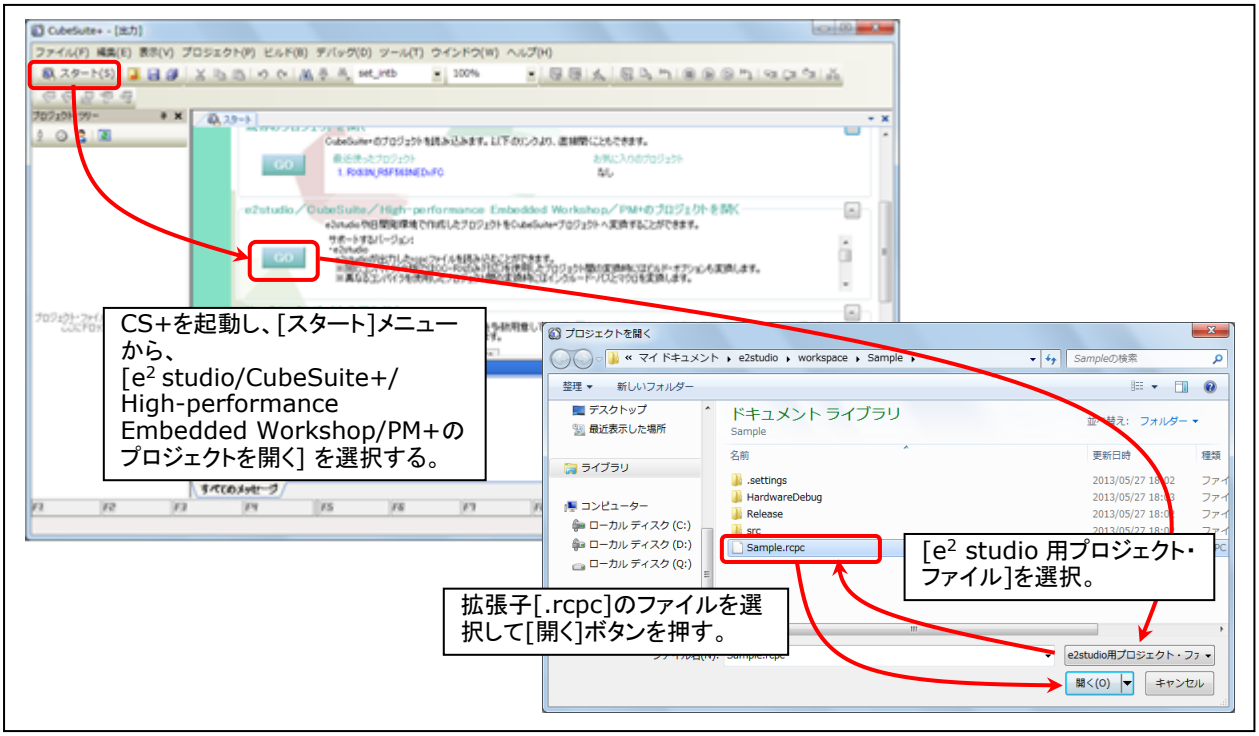
プロジェクトのワークスペースへのインポートが完了します。

10. e² studio 用プロジェクトを CS+で使用する場合

本プロジェクトは、統合環境 e² studio で作成されています。本プロジェクトを CS+で動作させる場合は、下記の手順を行ってください。

[Note]

rcpc ファイルは、workspace\RL78\CCRL\(MCU 名)フォルダ内に用意されています。



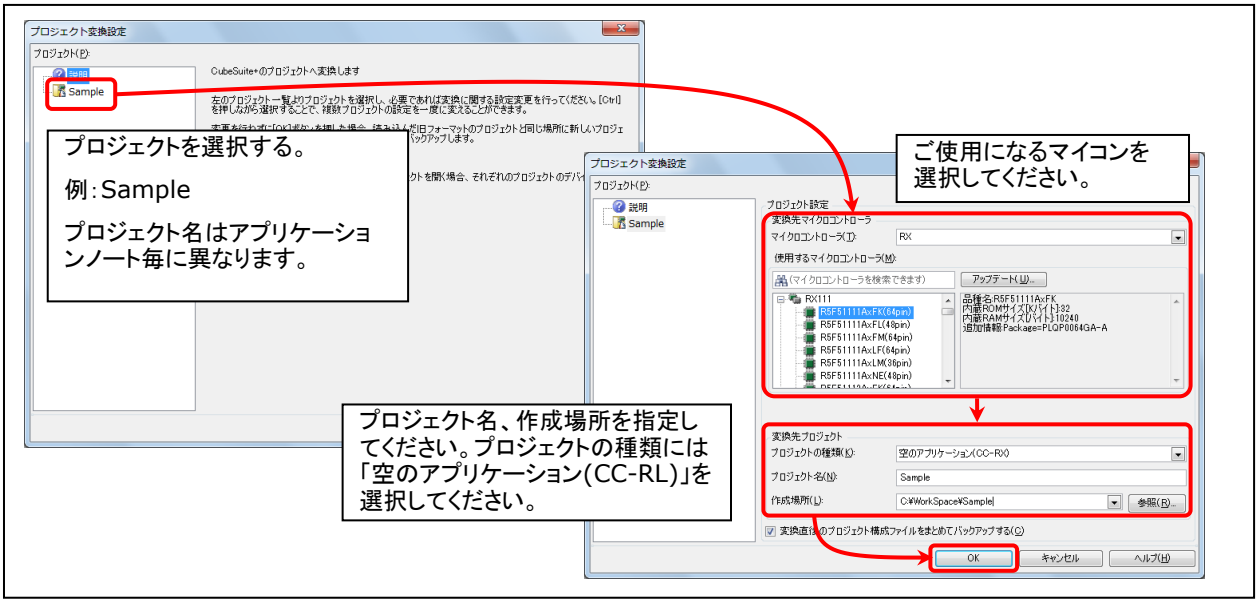


Figure 10-1 e² studio 用プロジェクトの CS+読み込み方法

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	2011.03.18	—	初版発行
2.00	2013.02.06	—	ファームウェアアップデートによるドキュメントの改訂
2.01	2013.03.26	—	IAR 環境について追記
2.10	2013.08.01	—	RL78/L1C, RX111 に対応、誤記訂正
2.11	2013.10.31	—	3.3.1 フォルダ構成を変更。これに伴い、1.4 のパス表記を修正。誤記訂正。
2.12	2014.03.31	—	R8C に対応、誤記訂正
2.13	2015.03.16	—	動作確認デバイスから RX111 を削除。
2.14	2016.01.18	—	Technical Update(発行番号: TN-RL*-A055A/J, TN-RL*-A033B/J)に対応しました。
2.15	2016.03.28	—	CC-RL コンパイラ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. 未使用端子の処理
   【注意】未使用端子は、本文の「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。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】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. リザーブアドレス（予約領域）のアクセス禁止
   【注意】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、将来の機能拡張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。
4. クロックについて
   【注意】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。
5. 製品間の相違について
   【注意】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、内部 ROM、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、ノイズ輻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。お客様の機器・システムの設計において、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て、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。
2. 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。
3. 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に関し、当社は、何らの責任を負うものではありません。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。
4. 当社製品を改造、改変、複製等しないでください。かかる改造、改変、複製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。
5. 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）、交通用信号機器、
   防災・防犯装置、各種安全装置等
   当社製品は、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（原子力制御システム、軍事機器等）に使用されることを意図しておらず、使用することはできません。 たとえ、意図し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損害が生じても、当社は一切その責任を負いません。 なお、ご不明点がある場合は、当社営業にお問い合わせください。
6. 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の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。
7. 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。
8. 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。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。
9. 本資料に記載されている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。また、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に使用しないでください。当社製品または技術を輸出する場合は、「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。
10. お客様の転売等により、本ご注意書き記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は何らの責任も負わず、お客様にてご負担して頂きますのでご了承ください。
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禁じます。

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。

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口

http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社　〒135-0061　東京都江東区豊洲3-2-24（豊洲フォレシア）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/contact/