Renesas Sample Demo Software

USB Speaker Demo Software for RX231HMI Kit

R01AN3828JJ0100
Rev.1.00
April 26, 2017

要旨

本アプリケーションノートは、”RX231 を用いた USB Audio Device Class 1.0 対応音楽再生ソフトウェア”である「USB Speaker Demo Software for RX231HMI Kit」(以下、本ソフトウェア)の動作・機能を説明します。

対象ボード

R0K5RX231D000BR (RX231 HMI ソリューションキット)

Web ページ
< https://www.renesas.com/products/software-tools/boards-and-kits/evaluation-demo-solution-boards/r0k5rx231d000br-rx231-hmi-solution-kit.html >

目次

1. はじめに

RX231 の USB 機能は Isochronous 転送をサポートしています。Isochronous 転送は一定間隔でデータ転送を行うタイプであり、データ転送エラー時のリトライがありません。このため、データの正確性よりリアルタイム性が重視される音楽や映像の再生に適しています。

本アプリケーションノートで説明する「USB Speaker Demo Software for RX231HMI Kit」は、USB Audio Device Class 1.0 規格の Isochronous OUT 転送を利用したスピーカ機能を持っています。そのため本ソフトウェアを書き込んだ RX231HMI Kit は、アンプ付きスピーカまたはヘッドフォンを接続することで、USB ホストの音楽を出力する「USB スピーカ」として機能します。

【注】 この資料に掲載している内容は、USB 規格をもとにした参考例であり、システムでの動作を保証するものではありません。実際のシステムに組み込む場合は、システム全体で十分検討評価し、お客様の責任において、適用可否判断してください。

[Note] USB Audio Device Class
USB Audio Device Class は、組み込み機器のオーディオ、ボイス、音楽関係諸機能を制御するための USB 規格です。オーディオデータの転送と、音量やトーンなどの音楽環境に直接影響を及ぼす機能の制御が含まれています。

(a) 用語と略語一覧

ADCD	: Audio Device Class driver
API	: Application Program Interface
APL	: Application
CS+	: ルネサス統合開発環境
e^2 studio	: Eclipse embedded studio
FIT	: Firmware Integration Technology
LCD	: Liquid Crystal Display
LPC	: Low Power Consumption
RX231HMI Kit	: RX231 Human Machine Interface Solution Kit
SSI	: Serial Sound Interface
USB	: Universal Serial Bus
USB Basic mini FIT	: USB Basic Mini Host and Peripheral Driver Firmware Integration Technology

(b) 関連ドキュメント

Table 1-1 関連ドキュメント一覧

No.	文書名	Revision	発行元
1.	Universal Serial Bus Specification	2.0	USB-IF
2.	Universal Serial Bus Device Class Definition for Audio Devices	1.0	
3.	Universal Serial Bus Device Class Definition for Terminal Types	1.0	
4.	Universal Serial Bus Device Class Definition for Audio Data Formats	1.0	
5.	PCM1774 データシート	-	TEXAS INSTRUMENTS
6.	RX ファミリ ボードサポートパッケージモジュール FIT アプリケーションノート（ドキュメント No. R01AN1685JJ）	3.30	Renesas Electronics
7.	USB Basic Mini Firmware アプリケーションノート（ドキュメント No. R01AN2166JJ ）	1.02	
8.	SSI モジュール FIT アプリケーションノート（ドキュメント No. R01AN2150EJ ）	1.20	
9.	簡易 I^{2}C モジュール FIT アプリケーションノート（ドキュメント No. R01AN1691JJ ）	1.60	
10.	LPC モジュール FIT アプリケーションノート（ドキュメント No. R01AN2769JJ ）	1.40	
11.	RX231 グループユーザーズマニュアル ハードウェア編（ドキュメント No. R01UH0496JJ ）	1.10	
12.	RX231 Human Machine Interface Solution Kit R0K5RX231D000BR 取扱い説明書（ドキュメント No. R01AN2586JJ ）	1.02	
13.	RX231 HMI Solution Kit Base Demo Software (機能限定版) RTK5RX2310P000F0ZR（ドキュメント No. R11AN0015JJ ）	1.00	

【注】 USB-IF <http://www.usb.org/developers/docs/>

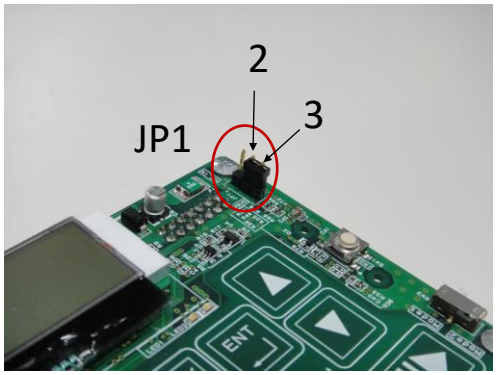
2. 使用方法

この章では、本ソフトウェアの使用方法を説明します。RX231HMI Kit の詳細な仕様は関連ドキュメント No.12 を参照してください。

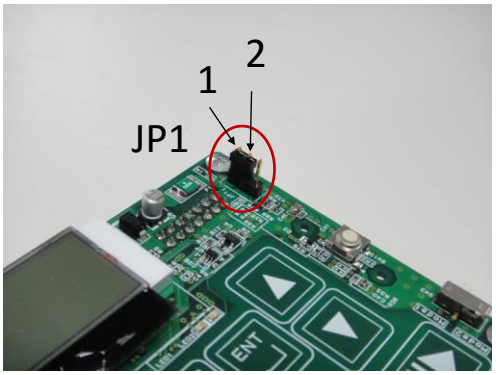
以下に示す手順に従って、機器の準備を進めてください。

1. 本ソフトウェアを統合開発環境（e² studio、CS+）もしくは Renesas Flash Programmer を使用して、RX231HMI Kit に書き込んでください。

2. Figure 2-1 に示すように、バスパワードで動作させる場合は JP1 の 2-3 間を短絡してください。セルフパワードで動作させる場合は JP1 の 1-2 間を短絡させてください。



(a) バスパワード
JP1　2-3間短絡



(b) セルフパワード
JP1　1-2間短絡

Figure 2-1 RX231HMI Kit の JP1 の接続

[Note]
Renesas Flash Programmer
< https://www.renesas.com/products/software-tools/tools/programmer/renesas-flash-programmer-programming-gui.html>

3. Figure 2-2、Figure 2-3 に示すように、アンプ付きスピーカまたはヘッドフォンを J2 ジャックに接続し、USB ホストと USB micro-B 端子で接続してください。J3 にスピーカまたはヘッドフォンを接続しないでください。

4. 接続すると USB ホストに自動的にドライバがインストールされます。

5. RX231HMI Kit の LCD のバックライトが点灯し、「RENESAS RX231 USB Audio Sample」と表示されていれば使用準備完了です。

6. USB ホスト上で音楽を再生すると音が鳴ります。音が鳴らない場合、Windows の再生デバイスに「スピーカー(USB Audio Sample)」が指定されているかを確認してください。

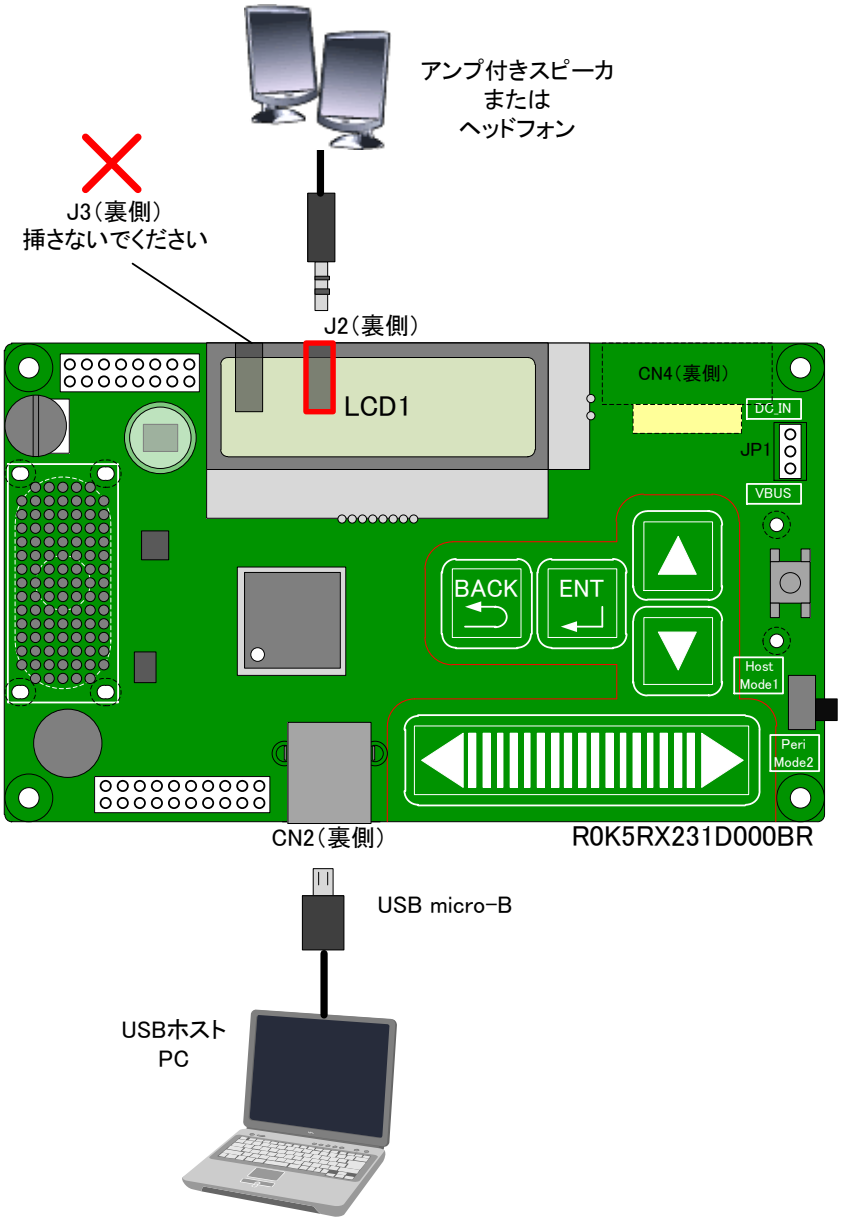


Figure 2-2 RX231HMI Kit と USB ホスト、アンプ付きスピーカまたはヘッドフォンの接続
（バスパワード時）

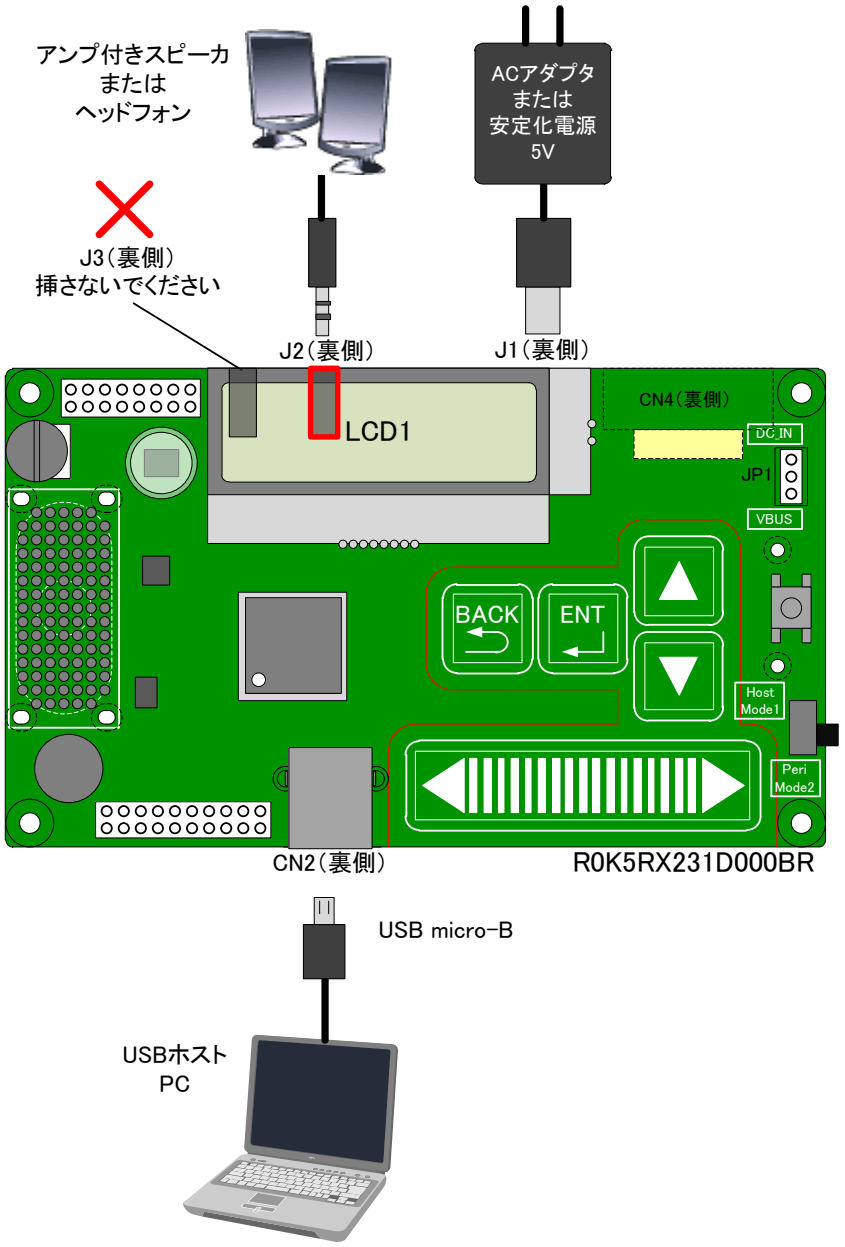


Figure 2-3 RX231HMI Kit と USB ホスト、アンプ付きスピーカまたはヘッドフォンの接続
（セルフパワード時）

3. デモソフトウェア概要

この章では本ソフトウェアの機能、ソフトウェア構造、使用している FIT、プロジェクトフォルダのファイル構成を示します。

3.1 機能

本ソフトウェアは、以下の機能を有しています。

- ホストとの USB 通信
- 44.1kHz 16bit 2ch(ステレオ)サンプリングの音源再生
- 音楽の再生・停止・一時停止
- 音量変更
- Mute 設定

【注】 クロック同期はしていませんので、注意してください。

3.2 ソフトウェア構造

本ソフトウェアのソフトウェア構造を Figure 3-1 に示します。

本ソフトウェアは RX231 の機能をベースとし、それらを弊社が提供する Firmware Integration Technology（以下、FIT）を利用して制御しています。各ドライバは FIT の API を利用して動作し、アプリケーション（以下、APL）は FIT と各ドライバが提供する API を利用して動作します。

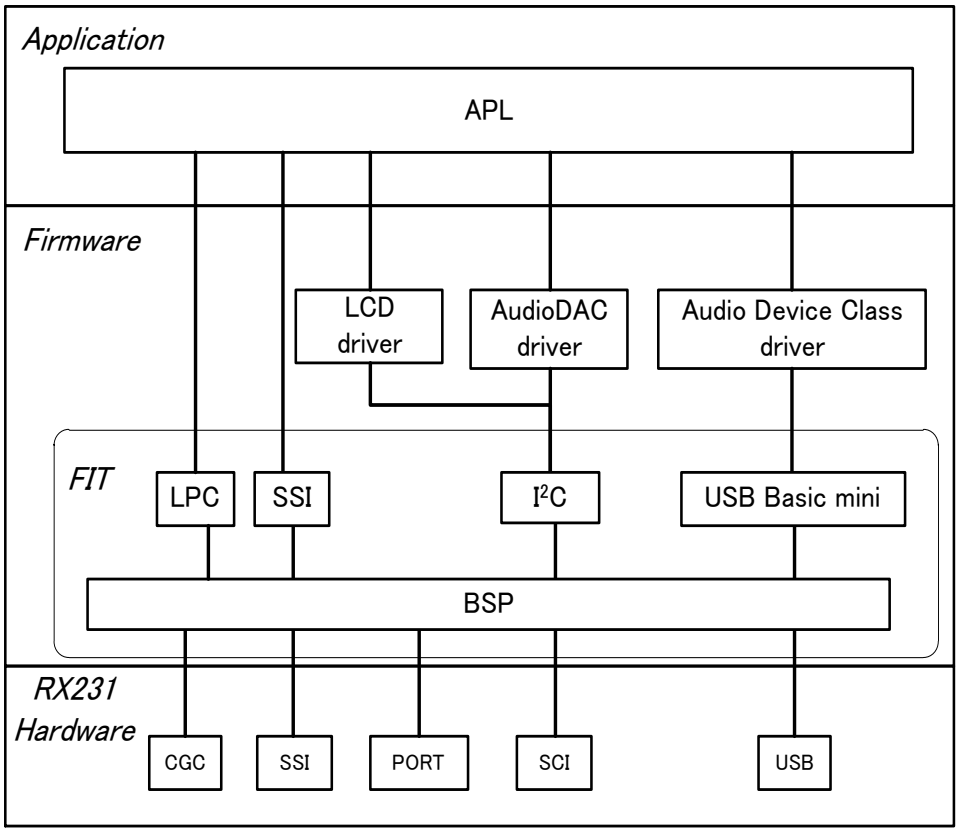


Figure 3-1 ソフトウェア構造

【補足】
RX231 Hardware の各機能はハードウェアマニュアルを参照してください。(関連ドキュメント No.11)

- BSP : RX ファミリ ボードサポートパッケージモジュール FIT
- LPC : LPC モジュール FIT
- SSI : SSI モジュール FIT
- I²C : 簡易 I²C モジュール FIT
- USB Basic mini : USB Basic mini FIT

ドライバは「5 Audio Device Class driver」および「4.6 AudioDAC driver」で説明します。LCD driver は RX231HMI Kit の無償版プロジェクト”RX231kit_free” RTK5RX2310P000F0ZR の流用です。詳細は無償版プロジェクトのアプリケーションノート（関連ドキュメント No.13）を参照してください。

3.3 アドレス空間

RX231HMI Kit には、R5F52318ADFP（RX231、ROM：512KB、RAM：64KB）が MCU として搭載されています。

本ソフトウェアが占有するアドレス空間を Figure 3-2 に示します。

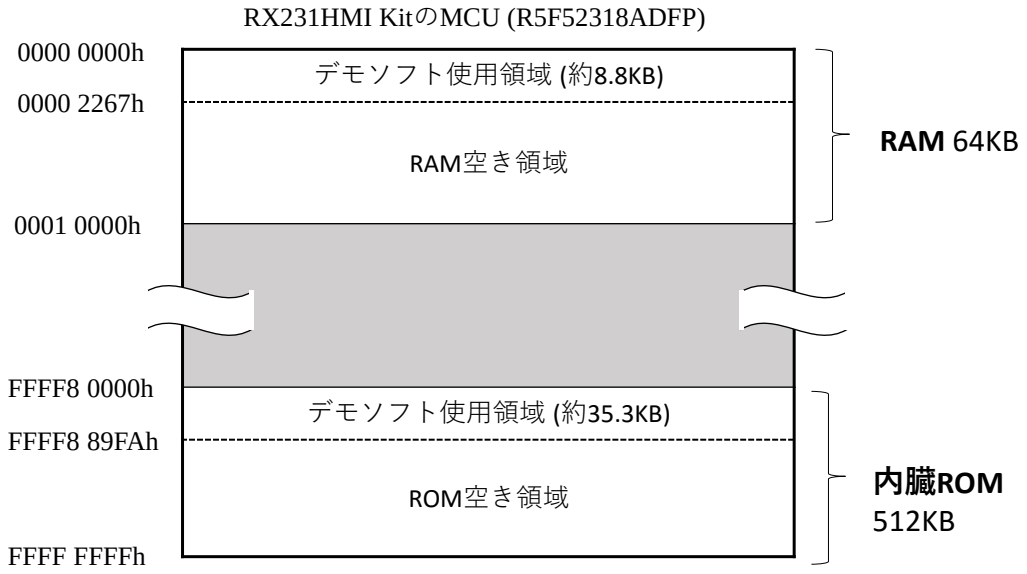


Figure 3-2 本ソフトウェアが占有するアドレス空間

3.4 ファイル構成

Table 3-1 に本ソフトウェアのファイル構成を示します。また Audio Device Class driver は、以下 ADCD と記述します。

FIT フォルダの詳細は各 FIT の関連ドキュメント No.6～10 を参照してください。

Table 3-1 本ソフトウェアのファイル構成

フォルダ名 / ファイル名	概要
RX231kit_paudio	--
.cproject	e² studio ファイル
.HardwareDebuglinker	
.info	
.project	
RX231kit_Audio.rcpc	
demo_src	APL ソースフォルダ
lcd.c	LCD 制御関数用ソースファイル
main.c	メイン関数用ソースファイル
audio_apl.c	APL 用ソースファイル
audio_apl_descriptor.c	USB Descriptor 用ソースファイル
inc	アプリケーションインクルードフォルダ
lcd.h	LCD 制御関数用インクルードファイル
audio_apl.h	アプリケーション用インクルードファイル
r_dac	AudioDAC driver フォルダ
r_dac_if.h	API 関数用インクルードファイル
readme.txt	—
src	AudioDAC driver ソースフォルダ
r_dac_api.c	API 関数用ソースファイル
r_dac_driver.c	ドライバ関数用ソースファイル
inc	AudioDAC driver インクルードフォルダ
r_dac.h	AudioDAC driver のドライバ関数用インクルードファイル
r_usb_paudio	ADCD フォルダ
r_usb_paudio_if.h	API 関数用インクルードファイル
readme.txt	—
ref	ADCD リファレンスフォルダ
r_usb_paudio_config_reference.h	ADCD リファレンスファイル
src	ADCD ソースフォルダ
r_usb_paudio_api.c	API 関数用ソースファイル
r_usb_paudio_driver.c	driver 関数用ソースファイル
inc	ADCD インクルードフォルダ
r_usb_paudio.h	ADCD のドライバ関数用インクルードファイル
r_config	コンフィグレーションフォルダ
r_usb_paudio_config.h	ADCD 用コンフィグレーションファイル
r_bsp_config.h	BSP モジュール FIT 用コンフィグレーションファイル
r_lpc_rx_config.h	LPC モジュール FIT 用コンフィグレーションファイル
r_sci_iic_rx_config.h	簡易 I²C モジュール FIT 用コンフィグレーションファイル
r_ssi_api_rx_config.h	SSI モジュール FIT 用コンフィグレーションファイル

r_usb_basic_mini_config.h	USB Basic mini FIT 用コンフィグレーションファイル
readme.txt	—
r_bsp	BSP モジュール FIT フォルダ
r_lpc_rx	LPC モジュール FIT フォルダ
r_sci_iic_rx	簡易 I²C モジュール FIT フォルダ
r_ssi_api_rx	SSI モジュール FIT フォルダ
r_usb_basic_mini	USB Basic mini FIT フォルダ

3.5 使用 FIT

本ソフトウェアで使用している FIT の一覧を Table 3-2 に示します。各 FIT モジュールの詳細は、関連ドキュメント No.6～10 を参照してください。また、本ソフトウェアで使用している FIT モジュールは弊社ホームページよりダウンロードが可能です。

Table 3-2 本ソフトウェア使用している FIT 一覧

FIT 名	Revision	フォルダ名	利用範囲
ボードサポートパッケージモジュール FIT	3.30	r_bsp	本ソフトウェア全体
USB Basic mini FIT	1.02	r_usb_basic_mini	ADCD
SSI モジュール FIT	1.20	r_ssi_api_rx	APL
簡易 I²C モジュール FIT	1.60	r_sci_iic_rx	APL AudioDAC driver LCD driver
LPC モジュール FIT	1.40	r_lpc_rx	APL

3.5.1 USB Basic mini FIT の修正

USB Basic mini FIT の機能を利用して ADCD は動作しますが、動作させるに当たり FIT の一部に変更を加えています。本ソフトウェアを参考に、弊社 FIT を利用したソフトウェアを開発する際はご注意ください。変更箇所および変更内容を Table 3-3 に示します。

Table 3-3 USB Basic mini FIT の変更箇所および変更内容

対象ファイル	対象関数	変更内容
r_usb_pdriver.c	usb_pstd_set_interface3()	正常値判定以外の処理がないため、クラスリクエストの処理を行うコールバック関数を呼ぶように追記。 (Alternate 値の変更・通知はコールバック関数内で行う)
	usb_pstd_get_interface1()	固定値(USB_0)を返す処理を削除し、クラスリクエストの処理を行うコールバック関数を呼ぶように変更。 (Alternate 値はコールバック関数内で返す)

4. アプリケーション

本ソフトウェアの APL は、各ドライバと FIT から提供される API を利用して動作しており、以下のような機能を有しています。

- USB で受信した Audio データを SSI を用いて Audio DAC IC PCM1774（以下、AudioDAC）に転送する機能。
- USB ホストからの Mute、音量変更指示に対応した AudioDAC 制御。
- USB の Attach、Detach、Suspend、Resume の検出に対応した状態遷移。（省電力状態への移行を含む）
- LCD の表示と、状態遷移に合わせた LCD バックライトの ON/OFF 切り替え

APL は、初期設定、メインループ、割り込み処理の 3 つから構成されています。以下に各処理について説明します。

4.1 初期設定

初期設定では、イベント情報のクリア、各ドライバの初期設定、FIT の初期設定、LCD の初期表示、そして使用しない IC やモジュールの動作停止処理を行っています。

一連の初期設定は関数 audio_apl_init()で処理しています。

4.2 メインループ

本ソフトウェアのメインループは、発生したイベントごとに処理が分けられています。この項では、イベントと各イベントの処理について説明します。

4.2.1 イベント

APL は、ADCD から通知を受け、各ドライバの制御を行います。通知はイベントとして管理され、メインループ内で常にイベントの発生を監視しています。

(a) 構造体

イベントとその関連データは、APL が用意する以下の構造体によって管理されています。

```
typedef struct                          /* Structure for event management */
{
    uint8_t     event[EVENT_MAX];        /* State for application */
    uint16_t    data[EVENT_MAX];         /* Event's data */
    uint16_t    event_cnt;              /* Event count */
} audio_eventinfo_t;
```

イベントを格納する変数を用意し、変数を引数として関数 audio_event_get()を呼ぶことで、イベントを取得できます。

Table 4-1 と Table 4-2 にそれぞれ関数 audio_event_get、関数 audio_event_set の仕様を示します。

Table 4-1 audio_event_get

機能	発生したイベントの取得		
宣言	void audio_event_get(uint8_t* event , uint8_t* data)		
引数	uint8_t*	event	イベント名格納先アドレス
	uint8_t*	data	関連データ格納先アドレス
戻り値	—	—	
仕様	引数として渡された変数 event にイベントコードを格納し、data に関連データを格納します。 イベントがない場合、APL_EV_NONE を event に格納します。 データがない場合、DATA_NONE を data に格納します。		

Table 4-2 audio_event_set

機能	イベントの格納		
宣言	uint8_t audio_event_set(uint8_t event_name , uint16_t data)		
引数	uint8_t	event	イベント名
	uint16_t	data	データ
戻り値	uint8_t	AUDIO_EVENT_SET_ERROR：イベント数オーバーエラー	
		AUDIO_EVENT_SET_SUCCESS：正常に処理終了	
仕様	引数として渡された event と data を保存します。 保存できるイベント数は初期設定で 5（マクロ定義：EVENT_MAX）までです。		

(b)　イベント一覧

APL で定義しているイベントの一覧を Table 4-3 に示します。

Table 4-3 イベントと関連データの一覧

イベント名	イベント内容	データ名	データ内容
APL_EV_NONE	イベントなし	DATA_NONE	データなし
APL_EV_USB_STREAM	USB 転送 開始・停止通知	STREAM_PLAY	Isochronous 転送開始
		STREAM_STOP	Isochronous 転送停止
APL_EV_USB_VOL	音量 変更通知	data	USB Audio Device Class で定められている音量データ
APL_EV_USB_MUTE	Mute 設定 変更通知	USB_MUTE_ON	Mute ON
		USB_MUTE_OFF	Mute OFF
APL_EV_USB_RX_COMPLETE	USB 受信 完了通知	DATA_NONE	データなし
APL_EV_USB_STS_CHANGE	USB 状態遷移 通知	USB_STS_DETACH	USB Basic mini FIT が提供する次状態のコード
		USB_STS_DEFAULT	
		USB_STS_ADDRESS	
		USB_STS_SUSPEND	
		USB_STS_RESUME	

【注】　音量データの詳細は、関連ドキュメント No.2「Universal Serial Bus Device Class Definition for Audio Devices rev1.0」の“Volume Control”の説明を参照してください。

4.2.2 動作フロー

メインループでは、以下の処理を行います。APL の動作フローを Figure 4-1 に示します。

(a) イベントおよび関連データを、関数 audio_event_get()を用いてループの最初に取得します。
(b) APL_EV_USB_STREAM ではデータ内容を判別し、STREAM_PLAY の場合は USB データの受信要求を行います。
(c) APL_EV_USB_VOL では、USB ホストより転送された音量データを AudioDAC の仕様に合わせて計算し、簡易 I²C 通信により AudioDAC のデジタル音量レジスタに書き込みます。
(d) APL_EV_USB_MUTE では、まずデータ内容が USB_MUTE_ON か USB_MUTE_OFF かを判断します。データの Mute 設定を簡易 I²C 通信により AudioDAC の Mute レジスタに書き込みます。
(e) APL_EV_USB_COMPLETE では USB データ受信要求を行い、SSI の開始条件が満たされていれば SSI をスタートさせます。
(f) APL_EV_USB_STS_CHANGE では、データの状態コードに従って状態遷移処理を行います。状態遷移処理は、関数 audio_change_sts()で行われます。

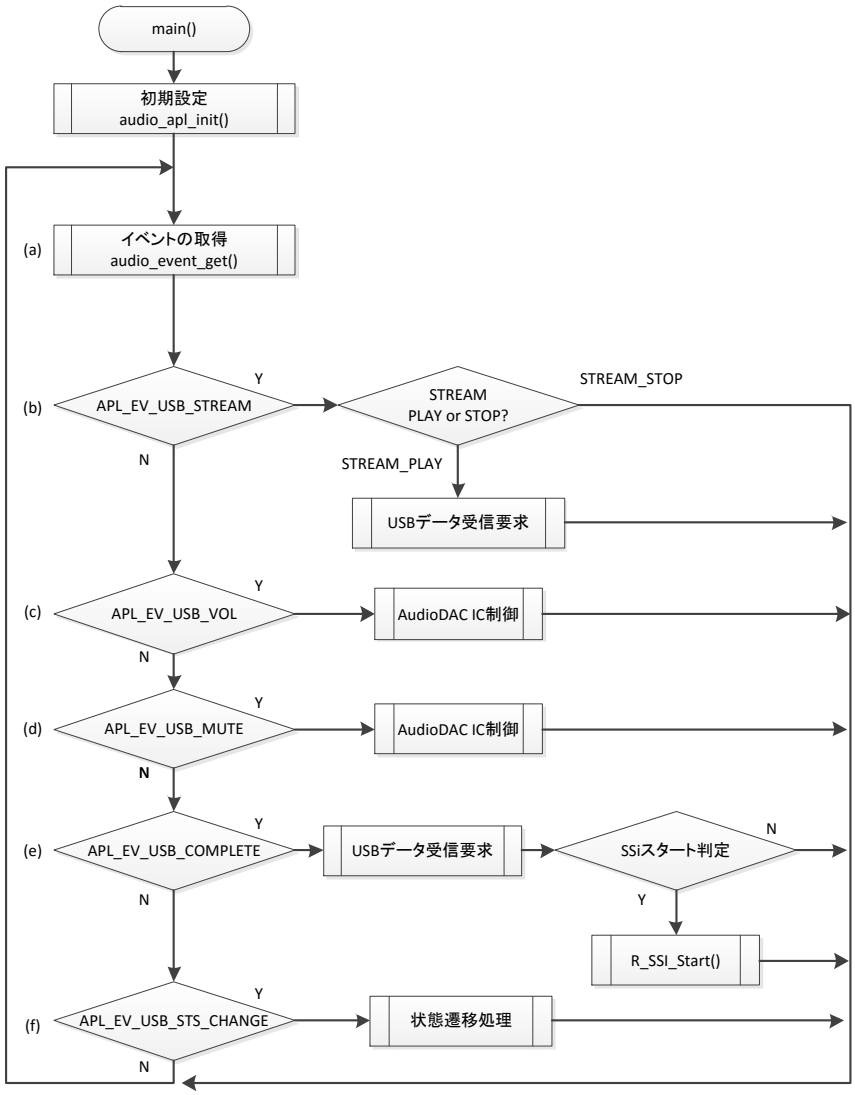


Figure 4-1 メインループの動作フロー

4.3 割り込み

APL で利用している割り込みを Table 4-4 に示します。

Table 4-4 APL で使用している割り込み

チャンネル	割り込み要因	内容
SSI0	SSITXI0 （送信データエンプティ割り込み）	SSI 送信 FIFO に格納された送信データの数が、送信 FIFO しきい値設定ビット(後述)の設定値以下になったとき発生します。

(a) SSITXI0 の発生タイミング

本ソフトウェアでは、Table 4-5 のように SSI FIT のコンフィグレーション設定を行っています。

Table 4-5 SSI FIT のコンフィグレーション定義の一部

コンフィグレーション定義	設定値	概要
SSI_CH0_DATA_WIDTH	16u	PCM データ幅
SSI_CH0_TTRG_NUMBER	4u	TDE フラグがセットされる値

この設定の場合、RX231 の送信 FIFO しきい値設定ビットは、1h と設定されます。このとき 32bit×8 段の FIFO のうち、2 段が空きになったときに割り込みが発生します。

Table 4-6 SSIFCR の TTRG 設定

レジスタ名	初期値	設定値
送信 FIFO しきい値設定ビット (TTRG)	0h	1h

(b) SSITXI0 での処理

SSITXI0 は SSI の転送開始処理で許可されます。割り込みが起こると、APL は RAM バッファから SSI FIFO へ 64 bit のデータを転送します。

本ソフトウェアでは、USB Isochronous 転送が停止し、RAM バッファから取り出せるデータがなくなると SSI 転送停止処理が実行されます。このとき SSITXI0 は禁止となります。APL は SSI 転送開始から停止まで、RAM バッファのデータを SSI FIFO に転送し続けます。

[Note]
割り込みの詳細は、RX231 のハードウェアマニュアル（関連ドキュメント No11）、および SSI FIT のアプリケーションノート（関連ドキュメント No.8）を参照してください。

4.4 省電力状態

本ソフトウェアは、USB ホストからの Suspend を検出して、RX231HMI Kit を省電力状態へと移行させます。Table 4-7 に MCU、AudioDAC、LCD の状態を示します。

なお、省電力状態は、USB ホストからの Resume を検出して解除されます。

Table 4-7 省電力状態

USB 状態遷移	MCU（RX231）	AudioDAC	LCD
Suspend	ソフトウェア スタンバイモード	省電力状態	バックライト OFF

4.5 Audio データ

本ソフトウェアは、Table 4-8 に示すレートとチャンネル数で Audio データを転送します。

Table 4-8 Audio データの情報

サンプリング周波数	ビットレート	チャンネル数
44.1 kHz	16 bit	2ch（ステレオ）

この仕様の場合、1ms に 44.1 個のサンプル転送が必要なので、

44.1[個/ms] × 16 [bit] × 2 [ch] = 176.4 [bytes/ms]

となり、1ms に 176.4 bytes の Audio データを転送する必要があります。

しかし、USB 転送は最小転送サイズが 1byte です。このため 176 bytes×9 回＋180 bytes×1 回の転送を繰り返すことで転送レートを合わせています。

後述する RAM バッファは、最大パケットサイズである 180 bytes を一つの面のデータサイズとしています。

4.5.1 RAM バッファ

Audio データを格納する RAM バッファは、180 bytes ×3 面で構成されています。以下に RAM バッファの構造を示します。この構造体は APL が用意しています。

```
typedef struct                                  /* Structure for PCM RAM buffer */
{
    uint8_t     data[PCM_BUF_NUM][PCM_BUF_SIZE];     /* PCM data */
    uint8_t     r_buf_num;                      /* The buffer number to write USB data  */
    uint8_t     w_buf_num;                      /* The buffer number to write in SSI */
    uint16_t    w_pos;                          /* SSI write pointer */
    uint16_t    r_len[PCM_BUF_NUM];         /* The length of stored data in each buffer */
} audio_buf_t;
```

4.5.2　Audio データの流れ

Audio データの流れを説明します。Figure 4-2 に一連の流れを図で示します。

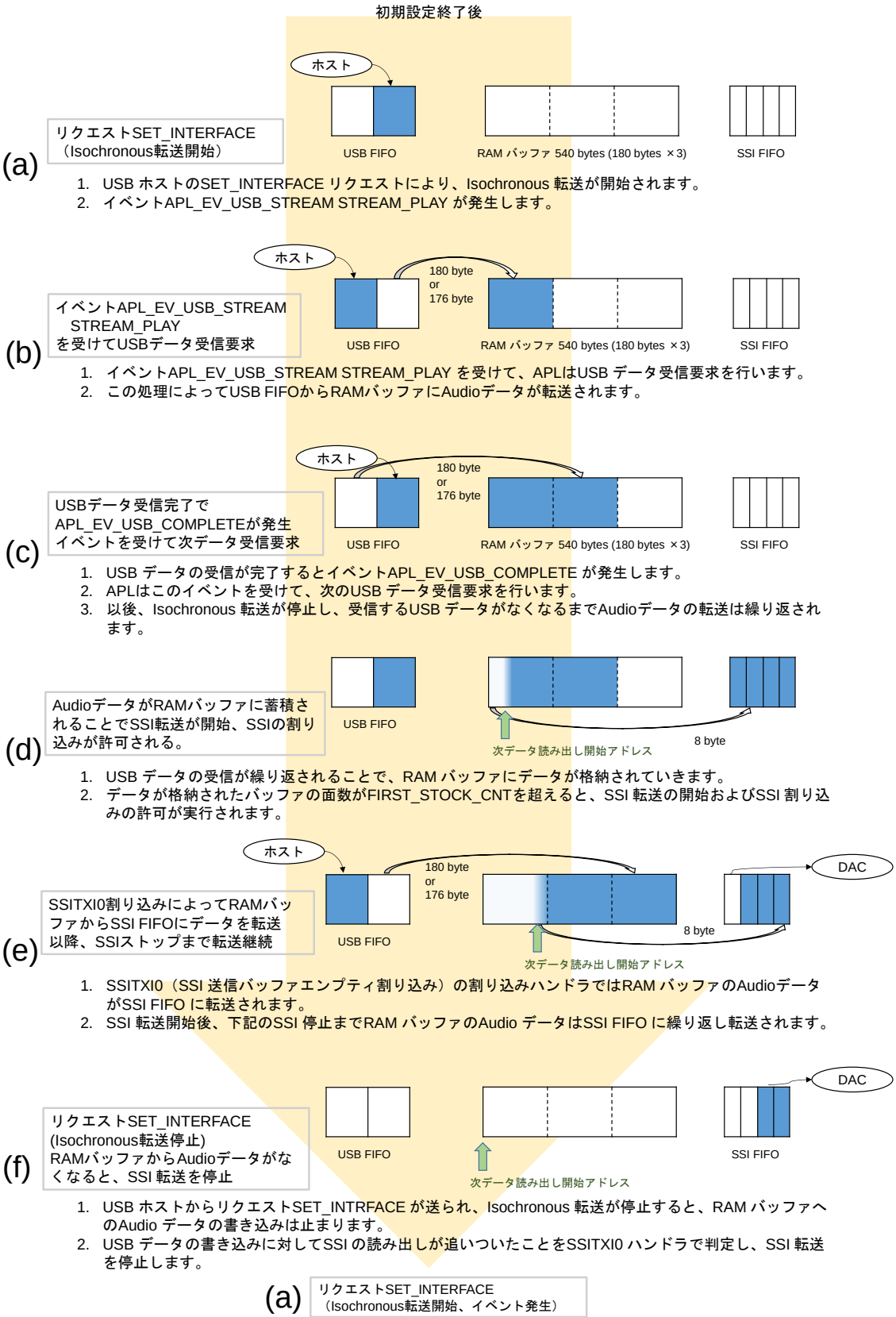


Figure 4-2 本ソフトウェアにおける Audio データの流れ

4.5.3 変更可能な設定

バッファ面数と SSI 転送開始の初期格納数は変更が可能です。これらの定義は audio_apl.h に記述されています。

Table 4-9 APL の変更可能な設定

マクロ名	初期値	内容
PCM_BUF_NUM	3	RAM バッファの面数
FIRST_STOCK_CNT	2	SSI 転送開始の条件。 RAM バッファにこの数値を超えてデータが格納されると、SSI 転送が開始されます。

4.6 AudioDAC driver

AudioDAC driver は、RX231HMI Kit に搭載されている TI 製 AudioDAC である PCM1774 の制御を行っています。

このドライバは簡易 I²C モジュール FIT を使用しています。ドライバを使用する前に簡易 I²C モジュールの初期設定を行ってください。

4.6.1 基本機能

AudioDAC driver は、以下の処理を行います。

- AudioDAC のパワーON および初期設定
- AudioDAC のパワーOFF
- AudioDAC の Mute 設定・音量変更
- AudioDAC の省電力状態移行
- AudioDAC の省電力状態解除

4.6.2 ヘッダファイル

すべての API 呼び出しとそれをサポートするインターフェース定義は r_dac_if.h に記載しています。

4.6.3 API 関数

以下に API 関数の機能について説明します。異なる AudioDAC を使用する場合は、処理を変更してください。

AudioDAC driver の API 関数とその機能を Table 4-10 に示します。

Table 4-10 AudioDAC driver の API 関数一覧

API 関数名	機能
void R_DAC_Open(void)	AudioDAC のパワーON および初期設定
void R_DAC_Close(void)	AudioDAC のパワーOFF
uint8_t R_DAC_Control(uint8_t param_type , uint16_t data)	Mute 設定・音量変更
void R_DAC_Suspend(void)	省電力状態への移行
void R_DAC_Resume(void)	省電力状態の解除

(a) R_DAC_Open

AudioDAC のパワーON および初期設定

形式

void R_DAC_Open(void)

引数

—	—

戻り値

—	—

解説

AudioDAC のパワーON と初期設定を行います。
以下の処理を行います。

1. SSI 通信のクロックを供給する発振器 SG-210 の電源を ON にします。
2. AudioDAC の初期化とパワーON を行います。

補足

この関数を呼ぶ前に簡易 I²C モジュール FIT の初期設定を実行してください。
AudioDAC driver の他関数よりも先に実行してください。

使用例

```
void sample_main( void )
{
    /* I2C module initialize */
    sample_iic_init();

    /* power ON AudioDAC */
    R_DAC_Open();

    while( 1 )
    {
        /* main loop process */
    }
}
```

(b) R_DAC_Close

AudioDAC のパワーOFF

形式

void R_DAC_Close(void)

引数

— —

戻り値

— —

解説

AudioDAC をパワーOFF します。
以下の処理を行います。

1. AudioDAC をパワーOFF します。
2. SSI 通信のクロックを供給する発振器(SG-210)の電源を OFF にします。

補足

—

使用例

```
void sample_task( void )
{
        :
    R_DAC_Close();    /* power OFF AudioDAC */
}
```

(c) R_DAC_Control

AudioDAC の Mute・音量変更

形式

uint8_t R_DAC_Control(uint8_t param_type, uint16_t data)

引数

param_type	変更対象のパラメータコード
data	設定値

戻り値

DAC_MUTE_PARAM_ERROR	Mute の設定が不正
DAC_VOLUME_PARAM_ERROR	音量の設定値が不正
DAC_PARAM_ERROR	パラメータコードが不正
DAC_CONTROL_SUCCESS	成功

解説

引数の値から指定された変更対象の設定値を書き換えます。
以下のように第 1 引数 param_type に値を渡し、第 2 引数 data に設定値を渡してください。

param_type	DAC_MUTE_SET	Mute 設定を変更します。
data	DAC_MUTE_ON	Mute を ON にします。
	DAC_MUTE_OFF	Mute を OFF にします。
param_type	DAC_VOL_SET	音量設定を変更します。
data	音量設定値（USB データ）	

第 1 引数のパラメータコードを判別し、第 2 引数の設定値を簡易 I^2C 通信で AudioDAC のレジスタに書き込みます。

補足

―

次ページに使用例を記載します。

使用例

```
void sample_main( void )
{
    uint8_t    event;
    uint16_t   data;
    uint8_t    err;

    /* I2C module initialize */
    sample_iic_init();

    /* power ON AudioDAC */
    R_DAC_Open();

        :
    while( 1 )
    {
        /* Get an event and related data */
        audio_event_get( event, data );

        switch (event)
        {
            case APL_EV_USB_VOL:
                /* Set the AudioDAC volume register */
                R_DAC_Control( DAC_VOLUME_SET, data );
                if ( err != DAC_CONTROL_SUCCESS )
                {
                    /* error process */
                }
            break;
            case APL_EV_USB_MUTE:
                if (USB_MUTE_ON == data)
                {
                    R_DAC_Control( DAC_MUTE_SET, USB_MUTE_ON );
                    if ( err != DAC_CONTROL_SUCCESS )
                    {
                        /* error process */
                    }
                }
                else if (USB_MUTE_OFF == data)
                {
                    R_DAC_Control( DAC_MUTE_SET, USB_MUTE_OFF );
                    if ( err != DAC_CONTROL_SUCCESS )
                    {
                        /* error process */
                    }
                }
            break;
            default:
                    /* No Action */
            break;
        }
        /* Other process */
            :
    }
}
```

(d) R_DAC_Suspend

AudioDAC を省電力状態へ移行

形式

void R_DAC_Suspend(void)

引数

— —

戻り値

— —

解説

AudioDAC を省電力状態に移行させます。
以下の処理を行います。

1. AudioDAC の一部モジュールをパワーOFF します。
2. SSI 通信のクロックを供給する発振器(SG-210)の電源を OFF にします。

補足

—

使用例

```
void sample_task( void )
{
        :
    R_DAC_Suspend();    /* Standby AudioDAC */
}
```

(e) R_DAC_Resume

AudioDAC を省電力状態から復帰

形式

void R_DAC_Resume (void)

引数

— —

戻り値

— —

解説

R_DAC_Suspend()によって省電力状態となった AudioDAC を通常状態に復帰させます。
以下の処理を行います。

1. SSI 通信のクロックを供給する発振器(SG-210)の電源を ON にします。
2. R_DAC_Suspend()で落としたモジュールをパワーON します。

補足

—

使用例

```
void sample_task( void )
{
        :
    R_DAC_Resume();    /* Resume AudioDAC */
}
```

4.7 Descriptor

本ソフトウェアの Descriptor 情報は、audio_apl_descriptor.c に記述されています。また、USB Audio Device Class 1.0 に含まれている音楽再生・停止、Mute・音量変更の機能をサポートしています。

なお Product ID および Vendor ID は、必ずお客様用の番号を使用してください。

Table 4-11 に本ソフトウェアが用意している各 Descriptor の内容を示します。

Table 4-11 Descriptor の概要

Descriptor	内容
Device Descriptor	機器固有の情報
Configuration Descriptor	デバイス構成に関する記述
AudioControl Interface Descriptor	デバイスが備える機能に関する記述
Standard AC Interface Descriptor	USB2.0 準拠の Interface 情報
Class-Specific AC Interface Header Descriptor	Audio Device Class のバージョン（1.0）や後続する Class-Specific, Terminal, Unit に関する記述
Input Terminal Descriptor	入力は Host からの USB Streaming であることを記述
Output Terminal Descriptor	Host に伝えるデバイスの種類に関する記述
AudioControl Feature Unit Descriptor	デバイスの制御可能要素に関する記述
AudioStreaming Interface Descriptor	Audio データ受信機能に関する記述
Standard AS Interface Descriptor (for Alternate Setting 0)	音楽出力停止に用いる Interface に関する記述
Standard AS Interface Descriptor (for Alternate Setting 1)	データ Stream に用いる Interface に関する記述
Class-Specific AS Interface Descriptor	Input Terminal と Data Format Type の紐付けに関する記述
Class-Specific AS Format Type Descriptor	通信データの Format に関する記述
Standard AS Isochronous Audio Data Endpoint Descriptor	USB2.0 準拠の Endpoint 情報
Class-Specific AS Isochronous Audio Data Endpoint Descriptor	データ Stream に用いる Endpoint に関する記述
String Descriptor	特定文字の情報（社名など）

Table 4-12 に本ソフトウェアの各 Descriptor と変数の関係を示します。

Table 4-12 本ソフトウェアにおける Descriptor と変数の関係

Descriptor 名	型	変数名
Device Descriptor	uint8_t	g_audio_device_descriptor []
Configuration Descriptor	uint8_t	g_audio_configuration []
Standard AC Interface Descriptor		
Class-Specific AC Interface Header Descriptor		
Input Terminal Descriptor		
Output Terminal Descriptor		
AudioControl Feature Unit Descriptor		
Standard AS Interface Descriptor (for Alternate Setting 0)		
Standard AS Interface Descriptor (for Alternate Setting 1)		
Class—Specific AS Interface Descriptor		
Class—Specific AS Format Type Descriptor		
Standard AS Isochronous Audio Data Endpoint Descriptor		
Class—Specific AS Isochronous Audio Data Endpoint Descriptor		
String Descriptor	uint8_t	*g_audio_str_ptr[]【注】
		g_audio_string_descriptor1[]
		g_audio_string_descriptor2[]
		g_audio_string_descriptor3[]
		g_audio_string_descriptor4[]
		g_audio_string_descriptor5[]

【注】 *g_audio_str_ptr[]は g_audio_string_descriptor1[] ~ g_audio_string_descriptor5[]を格納する配列

4.8 ユーザーコールバック

ADCD の API 関数 R_USB_PaudioReceiveData()を使用するにあたって、APL はコールバック関数を用意する必要があります。

コールバック関数は、引数に型 usb_utr_t を持った USB 通信構造体が渡され、送受信の残りデータ長、ステータス、及び転送終了の情報が格納されます。

本ソフトウェアでは、コールバック関数 cb_audio_receive_complete()内で RAM バッファの管理および USB 受信完了イベントの格納を行っています。RAM バッファの管理にはデータ通信構造体に格納されている受信データ長さを使用しています。

関数 R_USB_PaudioReceiveData()は、USB Basic mini FIT の API 関数 R_usb_pstd_TransferStart()を利用しています。処理内容やコールバック関数の詳細は、関連ドキュメント No.7 を参照してください。

5. Audio Device Class driver

ADCD は USB Basic mini FIT を組み合わせることで、RX231HMI Kit を USB Audio Device Class 1.0 に対応したデバイスとして動作させるドライバです。

ADCD の初期設定・登録・メインループ用関数の呼び出しを行えば、USB Basic mini FIT は自動的に実行されます。USB Basic mini FIT の処理を APL で行う必要はありません。

以下に ADCD の基本機能、コンフィグレーション定義、API について説明します。

5.1 基本機能

ADCD は以下の機能を持っています。

- Isochronous OUT 転送を用いた Audio データの転送
- コントロール転送を用いたオーディオパラメータ（Mute 設定、音量設定）の転送
- APL に対するコントロール転送受信完了通知（コールバック）
- APL に対する USB 状態遷移通知（コールバック）

通知の詳細は「5.3 クラスドライバ登録」で説明しています。

5.2 クラスリクエスト

ADCD は、Table 5-1 に示すクラスリクエストをサポートします。

Table 5-1 ADCD がサポートするクラスリクエスト

クラスリクエスト	コード	説明
GET_CUR	0x81	Control Selector で示されるオーディオパラメータの現在の値を返します。
SET_CUR	0x01	Control Selector で示されるオーディオパラメータの現在の値を変更します。
GET_MIN	0x82	Control Selector で示されるオーディオパラメータの最小値を返します。
GET_MAX	0x83	Control Selector で示されるオーディオパラメータの最大値を返します。
GET_RES	0x84	Control Selector で示されるオーディオパラメータの解像度を返します。

[Note]
スタンダードリクエスト（GET_INTERFACE および SET_INTERFACE を除く）は USB Basic mini FIT が処理を行っています。関連ドキュメント No.7 を参照してください。また本ソフトは、USB Basic mini FIT を一部変更して使用しています。詳細は「3.5.1 USB Basic mini FIT の修正」に記載しています。

USB Audio Device Class 1.0 は、各チャンネルの基本機能を制御する Feature Unit と呼ばれるユニットを持っています。制御する機能は Control Selector で選択されます。またチャンネルとの関連付けや、有効な制御機能の選択は、Descriptor に記述されます。

ADCD は Feature Unit を一つ持っており、サポートする Control Selector は Table 5-2 に示す通りです。

Table 5-2 Control Selector で ADCD がサポートする機能

Control Selector	コード	説明
MUTE_CONTROL	0x0100	Mute 設定を変更します。
VOLUME_CONTROL	0x0200	音量を変更します。

5.3 クラスドライバ登録

ADCD を使用するには、Descriptor とコールバック関数の登録を行う必要があります。

登録は初期設定関数 R_USB_PaudioOpen()を呼んだ後に、関数 R_USB_PaudioRegistration()を用いて行ってください。関数の詳細は「5.5API 関数仕様」に記載しています。

Table 5-3 に示す構造体 usb_paudio_reg_t は、Descriptor 登録・コールバック関数登録を行う際に使用する構造体です。

Table 5-3 usb_paudio_reg_t

型		概要
usb_paudio_reg_t		Descriptor 登録、コールバック関数登録を行うための構造体
型	メンバー	概要
uint16_t *	pipetbl	パイプ情報テーブルのアドレスを登録してください。
uint8_t *	devicetbl	Device Descriptor テーブルのアドレスを登録してください。
uint8_t *	configtbl	Configuration Descriptor テーブルのアドレスを登録してください。
uint8_t **	stringtbl	String Descriptor テーブルのアドレスを登録してください。
usb_cbinfo_t	statediagram	USB 状態遷移時に起動するコールバック関数を登録してください。
usb_paudio_cb_t	ctrlRxCB	オーディオ用コントロール転送受信時に呼ばれるコールバック関数を登録してください。詳細は「5.3.1 コールバック関数」で記述しています。

【注】 usb_cbinfo_t は USB Basic mini FIT が用意しています。詳細は関連ドキュメント No.7 を参照してください

5.3.1 コールバック関数

SET_INTERFACE、SET_CUR（MUTE_CONTROL、VOLUME_CONTROL）のコマンドが USB ホストより送られた場合、ADCD は登録された関数をコールバックします。コールバック関数は Table 5-4 に示す型で呼ばれ、イベントと関連データが引数として渡されます。

Table 5-4 (*usb_paudio_cb_t)(uint8_t, uint16_t)

型			概要
(*usb_paudio_cb_t)(uint8_t, uint16_t)			コントロール通知に使用するコールバック関数の型
引数	型	値	概要
	uint8_t	USB_PAUDIO_STREAM	音出力設定
		USB_PAUDIO_MUTE	Mute 設定
		USB_PAUDIO_VOLUME	音量設定
	uint16_t	STREAM_PLAY	音出力開始
		STREAM_STOP	音出力停止
		MUTE_ON	Mute ON
		MUTE_OFF	Mute OFF
		Volume 設定値	音量設定値
戻り値	—	—	

またコールバック関数が呼ばれるタイミングとその時に渡される引数は、Table 5-5 に示す関係を持ちます。

Table 5-5 コールバック関数が呼ばれるタイミングと引数の関係

第 1 引数 uint8_t	第 2 引数 uint16_t	発生タイミング
USB_PAUDIO_STREAM	STREAM_PLAY	SET_INTERFACE Alternate = 1 のリクエスト受信時
	STREAM_STOP	SET_INTERFACE Alternate = 0 のリクエスト受信時
USB_PAUDIO_MUTE	MUTE_ON	SET_CUR MUTE_CONTROL Parameter Block = 1 のリクエスト受信時
	MUTE_OFF	SET_CUR MUTE_CONTROL Parameter Block = 0 のリクエスト受信時
USB_PAUDIO_VOLUME	Volume 設定値	SET_CUR VOLUME_CONTROL のリクエスト受信時

5.4 API 情報

ADCD の API は、ルネサスの API 命名基準に従っています。

5.4.1 ハードウェア要求

ADCD は以下のハードウェアで動作を確認しています。

- R0K5RX231D000BR (RX231 HMI ソリューションキット)

5.4.2 ヘッダファイル

API 呼び出しとそれをサポートするインターフェース定義は r_usb_paudio_if.h に記載しています。

5.4.3 コンフィグレーション

ドライバ内の音量設定と使用するパイプの設定は、r_usb_paudio_config.h で行います。Table 5-6 にコンフィグレーション定義の一覧を示します。

Table 5-6 ADCD のコンフィグレーション定義

マクロ名	初期値	概要
USB_CFG_PAUDIO_PIPE_OUT	USB_PIPE1 (0x0001)	Isochronous 転送のアウトパイプのパイプ番号 以下から選択 USB_PIPE1 (0x0001) USB_PIPE2 (0x0002)
USB_CFG_PAUDIO_VOL_MAX	0x0000	音量の最大値
USB_CFG_PAUDIO_VOL_MIN	0xC100	音量の最小値
USB_CFG_PAUDIO_VOL_RES	0x0100	音量の解像度

RX231 の USB 機能では、USB_PIPE1 と USB_PIPE2 が Isochronous 転送をサポートしています。

本ソフトウェアの初期値は RX231HMI Kit に搭載されている AudioDAC の仕様（デジタル音量変更範囲：0 dB ~ -63 dB）に合わせたものになっています。

5.5 API 関数仕様

以下に API 関数の詳細を説明します。

Table 5-7 に ADCD の API 関数一覧を示します。

Table 5-7 ADCD の API 関数一覧

API 関数名	機能
void R_USB_PaudioOpen(void)	ADCD の初期設定
void R_USB_PaudioRegistration(usb_paudio_reg_t *paudio_reg)	テーブル登録、コールバック登録
void R_USB_PaudioReceiveData(uint8_t *Table, usb_leng_t size, usb_cb_t complete)	USB データ受信要求
void R_USB_PaudioDriver(void)	USB basic mini FIT のスケジュール管理・タスク処理。

5.5.1 R_USB_PaudioOpen

ADCD 起動関数

形式

void	R_USB_PaudioOpen(void)

引数

—	—

戻り値

—	—

解説

ADCD を起動させます。USB Basic mini FIT の起動処理も含まれています。
この関数は ADCD の他関数をコールする前に実行してください。
この関数は以下の処理を行っています。

1. USB 端子設定、USB モジュール初期化など
2. R_USB_Open()のコール
3. R_usb_pstd_PcdOpen()をコール

補足

—

使用例

```
void sample_main(void)
{
    /* Audio Device Class driver initializing and registration */
    R_USB_PaudioOpen();
    sample_usb_registraion();

    while( 1 )
    {
        /* Main loop process */
    }
}
```

5.5.2 R_USB_PaudioRegistration

ADCD 登録

形式

void R_USB_PaudioRegistration (usb_paudio_reg_t *paudio_reg)

引数

*paudio_reg	登録用構造体のアドレス

戻り値

—	—

解説

Descriptor と各種コールバック関数の登録を行います。
この関数は R_USB_PaudioOpen()の直後に実行してください。
この関数は以下の処理を行っています。

1. R_usb_pstd_DriverRegistration()をコールし以下の内容を USB Basic mini FIT に登録します。
   - パイプ情報テーブル
   - Device Descriptor テーブル
   - Configuration Descriptor テーブル
   - String Descriptor テーブル
   - USB 状態遷移時に起動するコールバック関数
   - ADCD のコントロール転送処理関数
2. オーディオ用コントロール転送の受信時に呼ばれるコールバック関数を登録

補足

- 登録する情報は「5.3 クラスドライバ登録 Table 5-3 usb_paudio_reg_t」を参照してください。
- R_usb_pstd_DriverRegistration()は USB Basic mini FIT の API 関数です。詳細は関連ドキュメント No.7 を参照してください。

使用例

例は、本ソフトウェアの APL の関数 audio_usb_registration()です。

```
void audio_usb_registration( void )
{
    usb_paudio_reg_t    audio_reg;

    /* USB endpoint Table */
    audio_reg.pipetbl     =    &g_audio_ep_tbl[0];
    /* Device descriptor */
    audio_reg.devicetbl    =    &g_audio_device_descriptor[0];
    /* Configuration Descriptor */
    audio_reg.configtbl    =    &g_audio_configuration[0];
    /* String Descriptor */
    audio_reg.stringtbl    =    (uint8_t**)&g_audio_str_ptr[0];
    /* USB state transition callback */
    audio_reg.statediagram =    cb_audio_change_device_state;
    /* USB control transfer complete callback */
    audio_reg.ctrlRxCB     =    cb_audio_usb_control_complete;

    /* Registration */
    R_USB_PaudioRegistration( &audio_reg );

}  /* eof audio_usb_registration() */
```

5.5.3 R_USB_PaudioReceiveData

USB データ受信要求

形式

void R_USB_PaudioReceiveData(uint8_t *Table, usb_leng_t size, usb_cb_t complete)

引数

*Table	受信データ格納バッファアドレス
size	受信したいデータのバイト数
complete	データ受信完了コールバック関数アドレス

戻り値

—	—

解説

USB Basic mini FIT に対して USB データ受信要求を行います。
受信完了時に第 3 引数のコールバック関数を呼び出します。受信完了のコールバック関数については「4.8 ユーザーコールバック」を参照してください。

補足

—

使用例

```
static audio_buf_t     g_audio_buf;
#define PCM_BUF_SIZE   (180)
void cb_sample_receive_complete( usb_utr_t * mess )


void sample_usb_receive( void )
{
    /* Receive USB data */
    R_USB_PaudioReceiveData( &g_audio_buf.data[0][0], PCM_BUF_SIZE,
                                                &cb_sample_receive_complete );
}

void cb_sample_receive_complete( usb_utr_t * mess )
{
    /* callback process */
}
```

5.5.4 R_USB_PaudioDriver

USB Basic mini FIT のスケジュール管理・タスク処理

形式

void R_USB_PaudioDriver(void)

引数

— —

戻り値

— —

解説

R_USB_cstd_Scheduler()をコールし、タスクメッセージをチェックします。
メッセージがある場合、USB Basic mini FIT の API 関数 R_usb_pstd_PcdTask()をコールします。

補足

この関数はメインループ内で呼び続けてください。

使用例

```
void sample_main( void )
{
    /* Initialize Audio Device Class driver and registration */
    R_USB_PaudioOpen();
    sample_usb_registration();

    while( 1 )
    {
        /* main loop process */
            :
        R_USB_PaudioDriver();
    }
}
```

6. 開発環境

本ソフトウェアは下記環境で開発、動作検証を行いました。各ツールの使用方法は、ツールのマニュアルを参照してください。

＜評価ボード＞

R0K5RX231D000BR (RX231 HMI ソリューションキット)

＜ツール＞

a) 統合環境 e² studio Ver.5.3.0.0.023 ルネサスエレクトロニクス製

b) RX ファミリ用 C/C++コンパイラパッケージ Ver2.05.00 ルネサスエレクトロニクス製

c) E1 エミュレータ ルネサスエレクトロニクス製

＜その他＞

a) USB ホスト PC （Windows® 7、Windows® 8.1、Windows® 10）

Windows 標準ドライバで動作可能

b) USB micro-B ケーブル

c) アンプ付きスピーカおよびヘッドフォン

(a) E1 接続

E1 エミュレータを接続して、RX231 のプログラム書換えやデバッグが可能です。下記の手順で接続して下さい。

(1) 本製品の電源がオフであることを確認します。

(2) 本製品の CN4 に E1 ケーブルを接続します。

E1 ケーブルの「誤挿入防止キー」位置に注意してください。

E1 の使用方法は E1 および開発環境の取扱説明書に従ってください。

E1 から電源供給する場合は、本製品の電源(DC ジャックまたは USB)をオフにしてください。JP1 を取り外すことで DC ジャックおよび USB の両電源入力が切断され安全に使用できます。

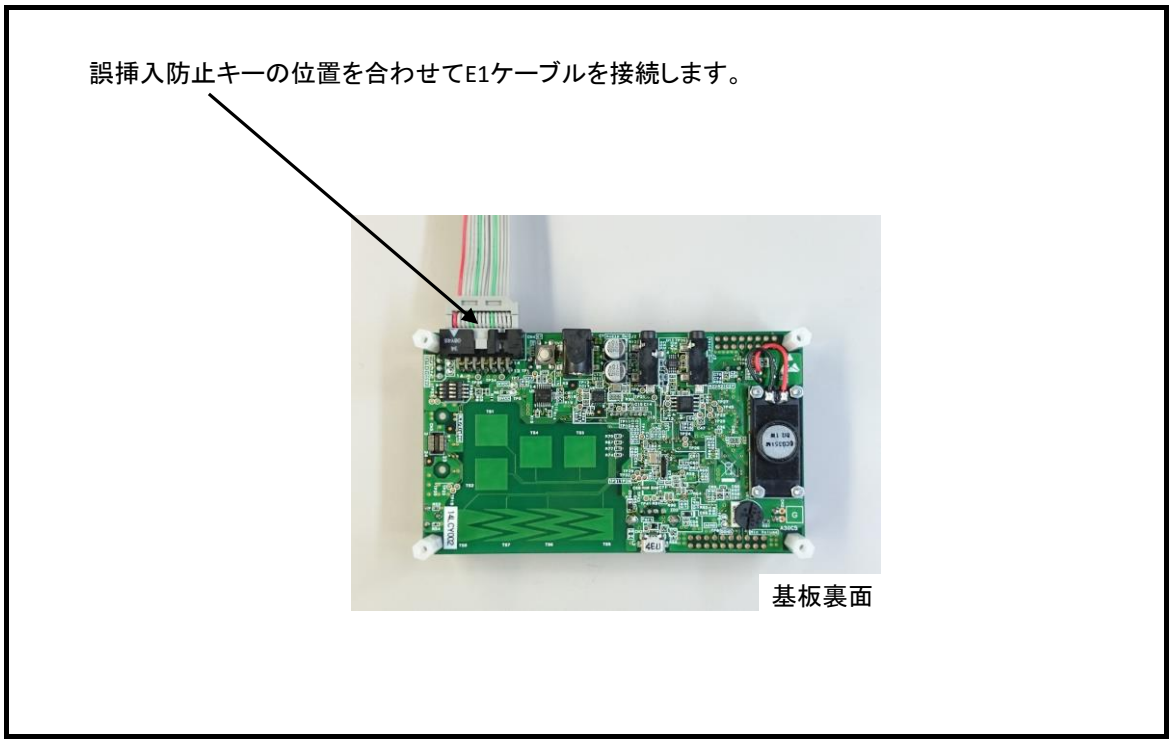


Figure 6-1 E1 接続図

(b) e² studio 用プロジェクトを CS+で使用する場合

本ソフトウェアは、統合環境 e² studio で作成されています。本ソフトウェアを CS+で動作させる場合は、下記の手順でインポート処理を行ってください。

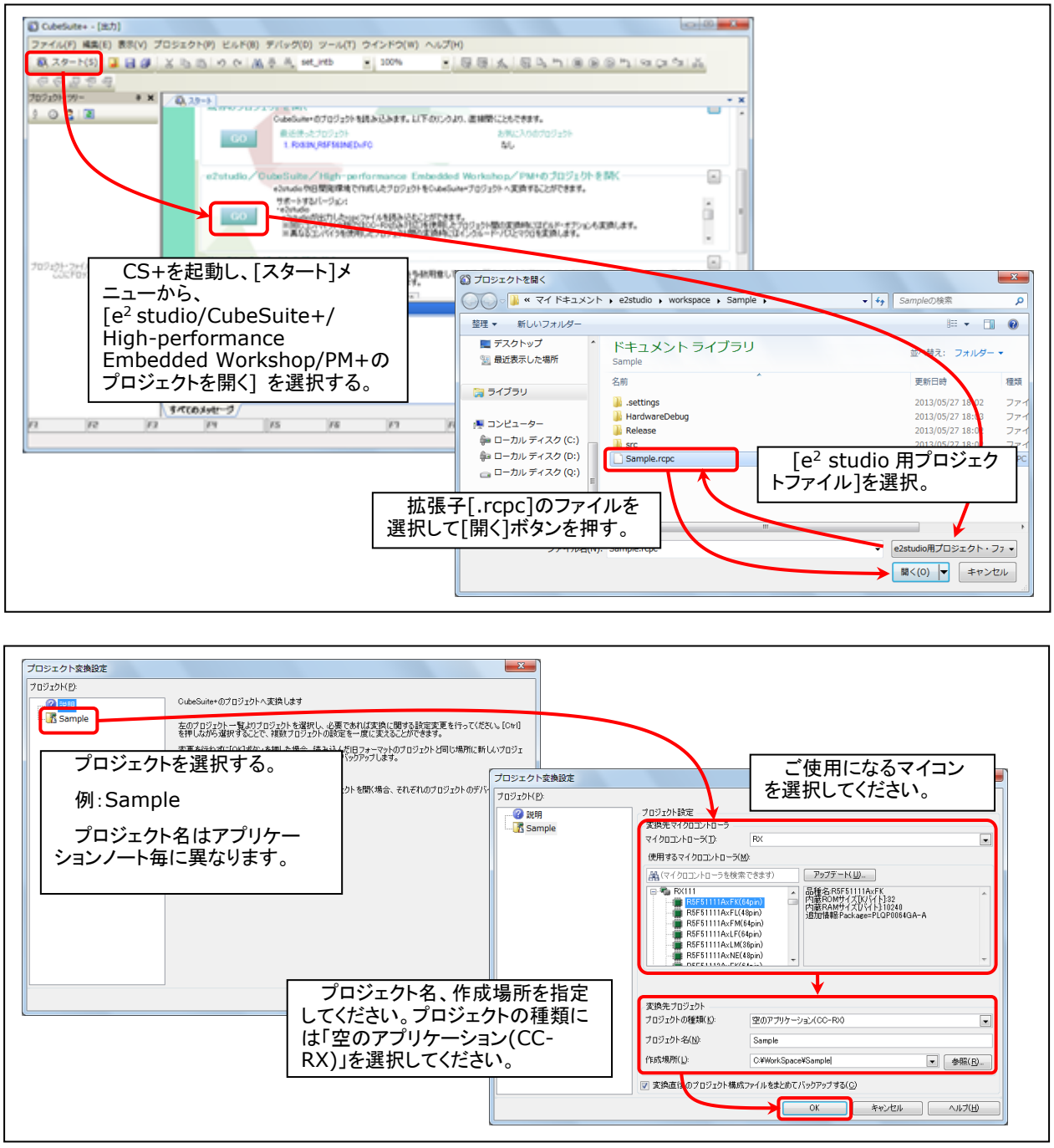


Figure 6-2 e² studio 用プロジェクトの CS+読み込み方法

7. 注記

7.1 未使用端子の処理

関連ドキュメント No.10「RX231 グループユーザーズマニュアル ハードウェア編」を参照してください。

7.2 USB ID の変更

本ソフトウェアに実装されている Vendor ID、Product ID はお客様の製品にはご使用できません。USB-IF から Vendor ID を必ず取得して下さい。

[Note] USB-IF <http://www.usb.org/home>

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
1.0	April 26, 2017	—	初版発行

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、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、全部または一部を問わず、改造、改変、複製、その他の不適切に使用しないでください。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、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、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損害が生じても、当社は一切その責任を負いません。
6. 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。
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。
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。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. 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、(1)核兵器、化学兵器、生物兵器等の大量破壊兵器およびこれらを運搬することができるミサイル（無人航空機を含みます。）の開発、設計、製造、使用もしくは貯蔵等の目的、(2)通常兵器の開発、設計、製造または使用の目的、または(3)その他の国際的な平和および安全の維持の妨げとなる目的で、自ら使用せず、かつ、第三者に使用、販売、譲渡、輸出、賃貸もしくは使用許諾しないでください。
   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。
10. お客様の転売、貸与等により、本書（本ご注意書きを含みます。）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、当社は一切その責任を負わず、お客様にかかる使用に基づく当社への請求につき当社を免責いただきます。
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。
12. 本資料に記載された情報または当社製品に関し、ご不明点がある場合には、当社営業にお問い合わせください。

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。

(Rev.3.0-1 2016.11)

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：https://www.renesas.com/contact/