

RM-120-RFB-1 アプリケーションノート 第1版

※必ず以下についてお守り下さい※

マニュアルに反した使い方をした場合、弊社は責任を負いかねます。

- 日本国内の法規に準拠して設計しています。サポートは日本国内限定とします。
弊社では、海外での保守・技術サポートなど行っておりません。
- 医療、原子力、航空宇宙、輸送など、人命に関わる設備や機器、および高度な信頼性を必要とする設備や機器などへは組み込まないで下さい。
人身事故、財産損害などが生じても、弊社はいかなる責任も負いかねます。
- 本製品は、無分別の一般ごみと一緒に廃棄しないで下さい。
お客様の責任で、別途、認可された収集リサイクル施設に委託して、使用済みの機器を正しく廃棄して下さい。

* 本マニュアルに記載の全ての情報は発行時点のものであり、予告なしに仕様を変更することがあります。最新情報は弊社ホームページをご確認下さい。

目次

1. はじめに	4
1.1. 本書について	4
1.2. 概要説明	4
2. 動作確認方法	5
2.1. 起動方法	5
2.2. 動作確認	7
2.2.1. 基本動作	7
2.2.2. 制御方法・動作の差分	7
2.2.3. ソースコードの差分	8
2.3. ファームウェア書き込み	9
3. 機能	13
3.1. 機能一覧	13
4. 機能評価	14
4.1. LED 制御	14
4.1.1. 使用端子	14
4.2. ポテンションメータ	15
4.2.1. 機能説明	16
4.2.2. 構成図	16
4.2.3. 使用端子	17
4.2.4. HW の設定	17
4.2.5. サンプルコード変更手順	18
4.3. 制御スイッチ	19
4.3.1. 使用端子	19
4.3.2. HW の設定	20
4.4. USB シリアル変換	21
4.4.1. 機能説明	21
4.4.2. 構成図	22
4.4.3. 使用端子	22
4.4.4. HW の設定	23
4.4.5. サンプルコード変更手順	24
4.5. USB ホスト	25
4.5.1. 機能説明	25
4.5.2. 構成図	26
4.5.1. 使用端子	26
4.5.2. HW の設定	26
4.5.3. サンプルコード変更手順	27
4.6. USB ファンクション	29
4.6.1. 機能説明	29
4.6.2. 構成図	30
4.6.3. 使用端子	30
4.6.4. HW の設定	30
4.7. CAN 通信	31
4.7.1. 機能説明	31
4.7.2. 構成図	32
4.7.3. 使用端子	32
4.7.4. サンプルコード変更手順	33

4.8. タッチキー・スライダー	35
4.8.1. 機能説明	35
4.8.2. 構成図	36
4.8.3. 使用端子	36
4.8.4. サンプルコード変更手順	37
4.8.5. タッチセンサチューニング	42
4.9. BLUETOOTH HCI	44
4.9.1. 機能説明	44
4.9.2. 構成図	45
4.9.3. 使用端子	45
4.9.4. HW の設定	45
4.10. BLUETOOTH SERVER	46
4.10.1. 機能説明	46
4.10.2. 構成図	47
4.10.3. 使用端子	47
4.10.4. HW の設定	47
4.10.5. サンプルコード変更手順	48
4.11. BLUETOOTH CLIENT	49
4.11.1. 機能説明	49
4.11.2. 構成図	50
4.11.3. 使用端子	50
4.11.4. HW の設定	50
4.12. BLUETOOTH MESH	51
4.12.1. 機能説明	51
4.12.2. 構成図	52
4.12.3. 使用端子	52
4.12.4. HW の設定	53
4.12.5. サンプルコード変更手順	53
5. 参考情報	54
5.1. 出荷時ソフトウェアへの復元	54
5.2. 新規開発プロジェクト作成時の注意事項	54
6. 改定履歴	55

RM-120-RFB-1	2021/11/01	SBAL-210166-00	4/55
アプリケーションノート			

1. はじめに

1.1. 本書について

本アプリケーションノートは、RM-120-RFB-1 (以下、RX23W モジュールボード) と EV-120-USB-1 (以下、RX23W 評価用ベースボード) が持つ各種機能を動作させる際の、サンプルプログラムの設定手順について記載したものです。

また、本書では RX23W モジュールボードと RX23W 評価用ベースボードの総称として RX23W 評価キットと称します。

1.2. 概要説明

RX23W モジュールボードはルネサスエレクトロニクス株式会社より提供されている『Bluetooth Low Energy プロトコルスタック』の基本パッケージを元に RX23W モジュールボードと RX23W 評価用ベースボードに合わせて修正を行ったサンプルプログラムが書き込まれています。

サンプルプログラムを使用することで、「Bluetooth 汎用属性 (GATT) Server」の動作を確認することが可能です。

サンプルプログラムの基本的な動作は「RX23W グループ Target Board for RX23W クイックスタートガイド (発行元: ルネサスエレクトロニクス株式会社)」を参照してください。

『Bluetooth Low Energy プロトコルスタック』の基本パッケージに対して行った修正内容の詳細は、「4.10 Bluetooth Server」を参照してください。

2. 動作確認方法

2.1. 起動方法

本製品は USB コネクタからの電源供給をサポートしています。
次の手順に従って電源を投入してください。

① ジャンパの接続設定

RX23W 評価用ベースボード上に用意されているジャンパを表 2-1 に合わせて接続してください

表 2-1 ジャンパ設定

ジャンパ	設定
J6	Short
J7	Open
J20	Open

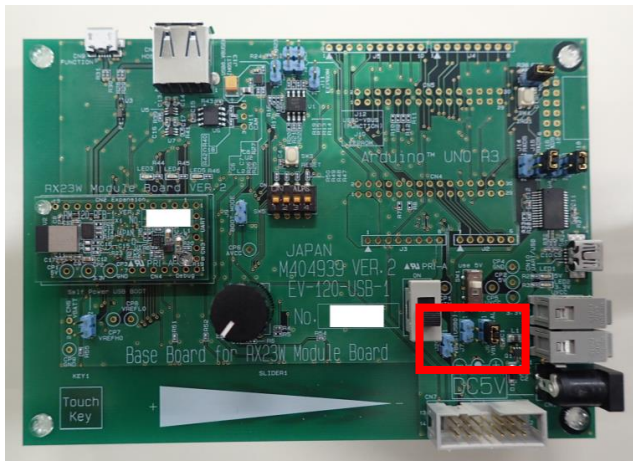


図 2-1 ジャンパ J6・J7・J20 設定例

② USB ケーブルの接続

RX23W 評価用ベースボードのパワースイッチが OFF であることを確認し、USB シリアル変換用コネクタ (VBUS1) に USB ケーブルを接続して下さい。

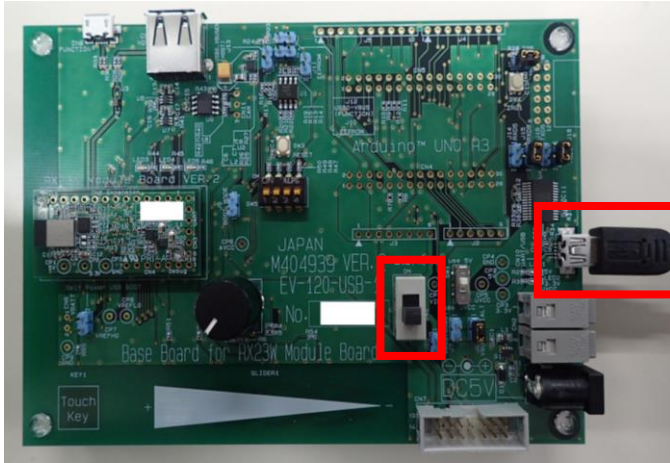


図 2-2 USB ケーブルの接続

③ 電源投入

RX23W 評価用ベースボードのパワースイッチを ON してください。
電源が正常に供給されると、LED1 (緑)、LED2 (緑) が点灯します。

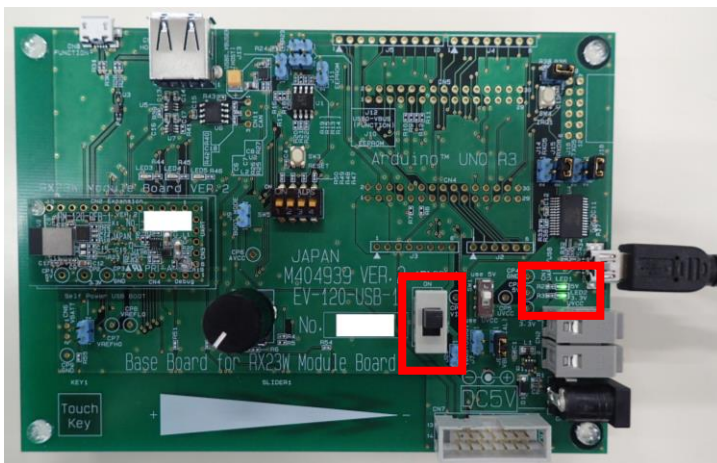


図 2-3 電源投入

- RX23W 評価用ベースボードは電源供給方法として「USB シリアル変換用コネクタ (VBUS1)」の他に、「DC ジャック (5V)」、「USB Function (VBUS2)」、「外部入力電源」が用意されています。ご使用の際は、「EV-120-USB-1 (RX23W 評価用ベースボード) ユーザーズ・マニュアル」を参照していただき、対応するジャンパ設定を行ってください。

2.2. 動作確認

2.2.1. 基本動作

RX23W モジュールボードに書き込まれているサンプルプログラムは、ルネサスエレクトロニクス株式会社が提供している「Bluetooth Low Energy プロトコルスタック 基本パッケージ」の「Target Board 向け GATT サーバプロジェクト」を元に、修正を行ったものとなります。

基本的な動作は「Target Board 向け GATT サーバプロジェクト」と同等のものとなります。

動作の概要は「RX23W グループ Target Board for RX23W クイックスタートガイド (発行元: ルネサスエレクトロニクス株式会社)」をご参照ください。

2.2.2. 制御方法・動作の差分

RX23W 評価キットに合わせてプログラムの修正を行ったことにより、「Target Board 向け GATT サーバプロジェクト」の動作の一部が変化します。

➤ LED 制御ポートの変更

RX23W 評価用ベースボードの回路構成に合わせて LED を制御しているポートを変更しています。

また、制御対象となる LED のアクティブレベルが反転します。そのため、「RX23W グループ Target Board for RX23W クイックスタートガイド (発行元: ルネサスエレクトロニクス株式会社)」に記載されている点灯／点滅動作が反転します。

表 2-2 LED 制御ポートの変更

Target Board for RX23W		⇒	RX23W 評価用ベースボード	
制御ポート	LED		制御ポート	LED
PC0	LED0		PE0	LED4
PB0	LED1		P03	LED5

表 2-3 アクティブレベルの変更

ポート出力	Target Board for RX23W	RX23W 評価用ベースボード
High	消灯	点灯
Low	点灯	消灯

➤ 制御 SW の変更

RX23W 評価用ベースボードの回路構成に合わせてスイッチ入力を制御しているポートを変更しています。

表 2-4 SW 制御ポートの変更

Target Board for RX23W		⇒	RX23W 評価用ベースボード	
制御ポート	LED		制御ポート	LED
P15/IRQ5	SW1		P31/IRQ1	SW4

RM-120-RFB-1	2021/11/01	SBAL-210166-00	8/55
アプリケーションノート			

2.2.3. ソースコードの差分

「Target Board 向け GATT サーバプロジェクト」に対して行った修正内容の詳細は、「4.10 Bluetooth Server」を参照してください。

2.3. ファームウェア書き込み

RX23W モジュールボードにファームウェアの書き込みを行う場合は以下の手順で実施してください。

- ① RX23W モジュールボードを RX23W 評価用ベースボードに接続してください。

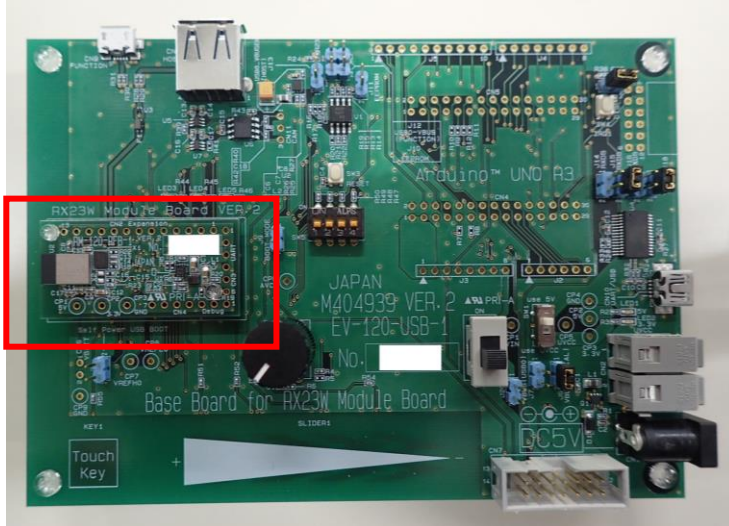


図 2-4 RX23W モジュールボードの接続

- ② RX23W 評価用ベースボードのデバッグコネクタに“E2 emulator”または、“E2 emulator Lite”を接続してください。

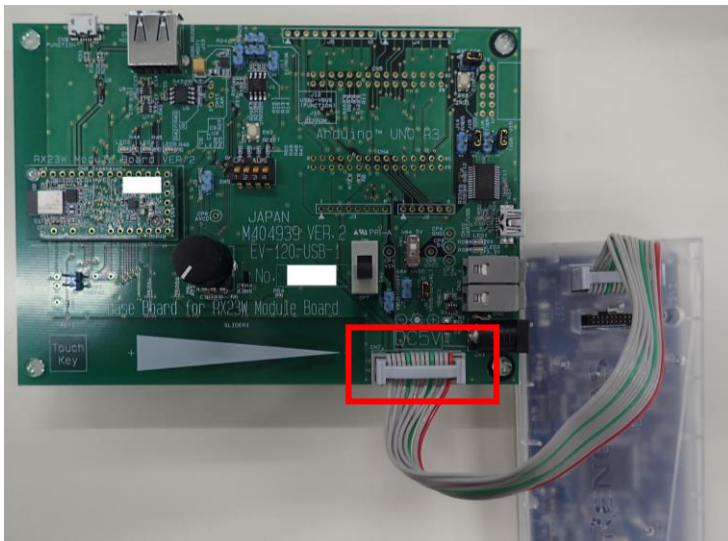


図 2-5 emulator の接続 (例: E2 emulator Lite 使用時)

- ③ 「2.1 起動方法」に従って電源を投入してください。
- ④ 「Renesas Flash Programmer (以下、RFP)」を起動し、[ファイル]→[新しいプロジェクト]を選択してください。

RM-120-RFB-1	2021/11/01	SBAL-210166-00	10/55
アプリケーションノート			

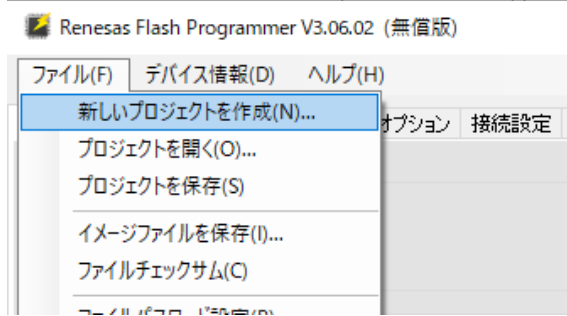


図 2-6 RFP 新しいプロジェクトを作成

- ⑤ [新しいプロジェクトの作成]ウィンドウで以下の設定を行い、[接続]ボタンをクリックしてください。
 - マイクロコントローラ: RX200
 - プロジェクト名: 任意のプロジェクト名を入力
 - 作成場所: 任意のフォルダを選択
 - 通信 ツール: ②で接続した emulator を選択
 - 通信 インタフェース: FINE を選択
 - 電源: 供給しないを選択

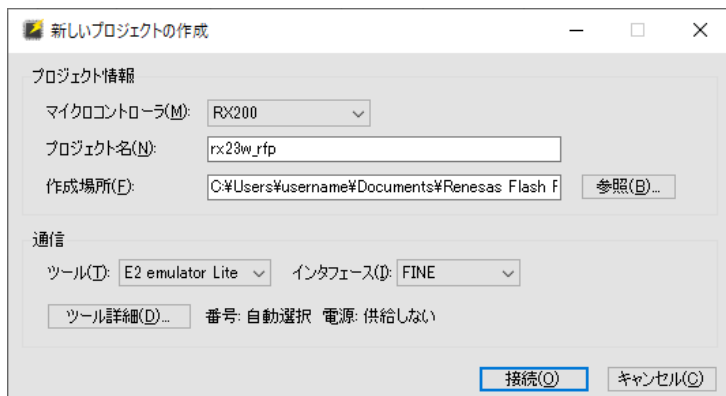


図 2-7 RFP プロジェクト設定

- ⑥ [ID コードの設定]の入力を求められるため、「45FFF」を入力し[OK]ボタンをクリックしてください。

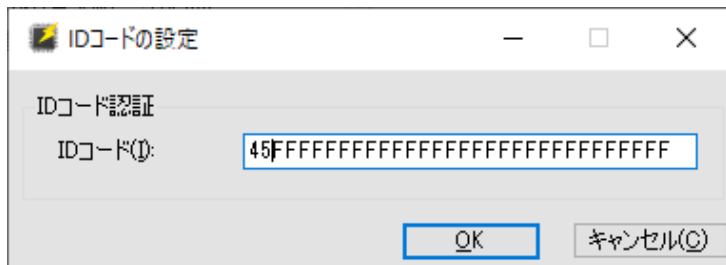


図 2-8 RFP IDコードの設定

- ⑦ 接続に成功すると、「操作が成功しました。」と表示されます。

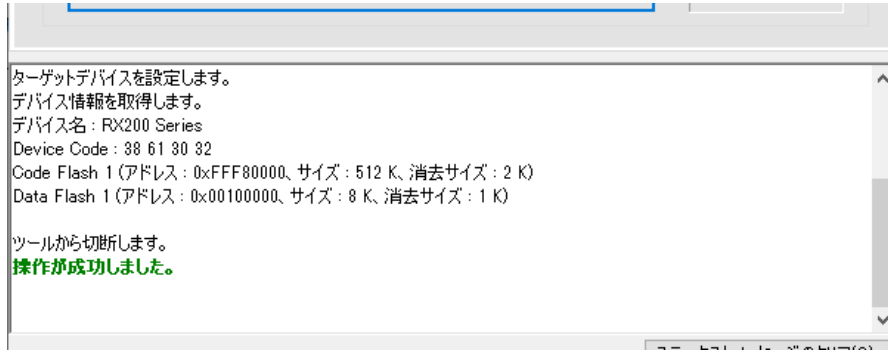


図 2-9 RFP 接続の成功

⑧ [参照]ボタンをクリックし書き込み対象となるファイルを選択してください。



図 2-10 RFP プロジェクトの選択

⑨ [ブロック設定]タブを選択し、[Data Flash]→[Block 256]のチェックボックスを外してください。

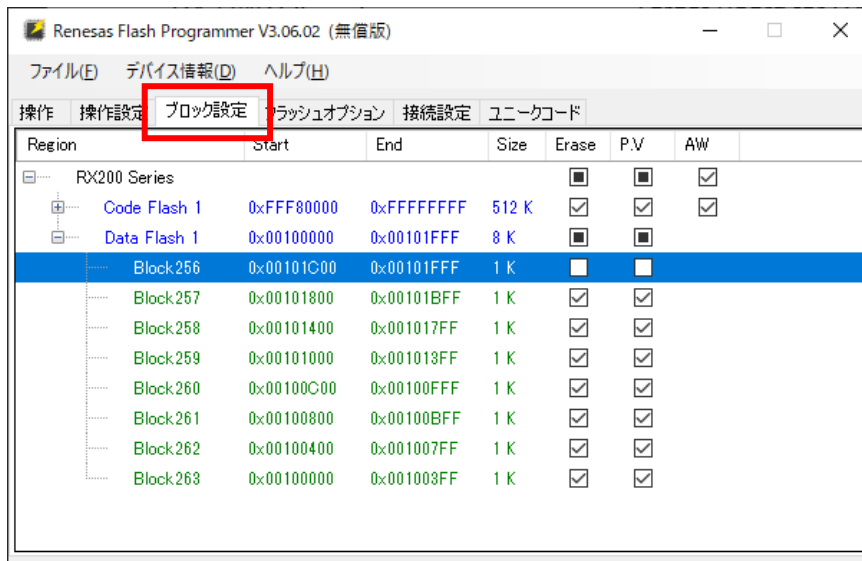


図 2-11 RFP ブロック設定

- ⑩ [操作]タブの[スタート]ボタンをクリックし、ファームウェアの書き込みを開始してください。

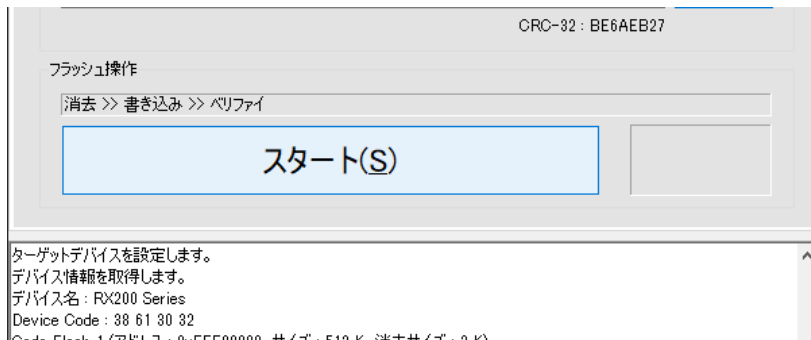


図 2-12 RFP 書き込み開始

- ⑪ 書き込みが正常に終了すると「操作が成功しました。」及び「正常終了」と表示されます。

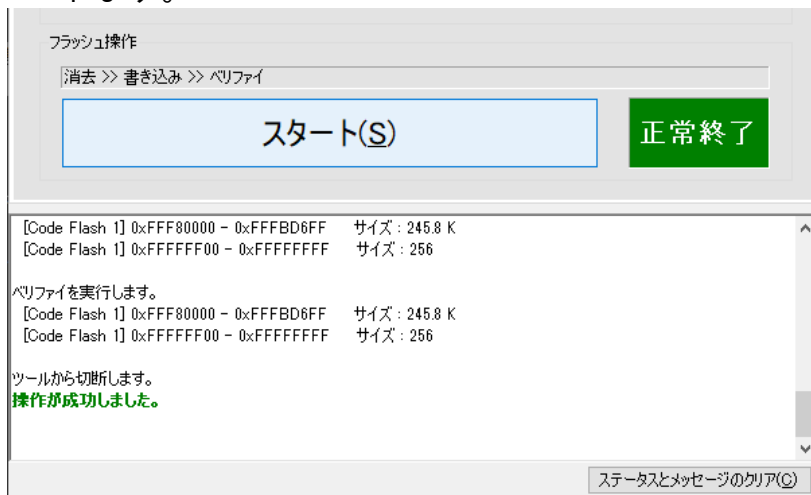


図 2-13 RFP 書き込み完了

- ⑫ 書き込み終了後、RX23W 評価用ベースボードの電源を OFF し、emulator をデバッグコネクタから外してください。

3. 機能

RX23W 評価キットは次のような機能を評価することができます。

各機能はルネサスエレクトロニクス株式会社様より提供されている、RX23W-Starter-Kit 用のサンプルコードを修正することで簡単に確認することができます。

3.1. 機能一覧

RX23W 評価キットが持つ機能の一覧は次の通りです。

表 3-1 機能一覧

機能	概要
LED 制御	ポート制御により LED の点灯状態を制御します。
ポテンショメータ	ポテンショメータを制御することで A/D コンバータ (P40 /AN000) にかかる電圧を制御します。
制御スイッチ	プッシュスイッチによる外部割込み、スライドスイッチによるポートへの入力電圧を制御します。
USB シリアル変換	USB シリアル変換機能を使用し、シリアル通信を行います。
USB ホスト	USB ホスト機能を使用して接続対象の制御を行います。
USB ファンクション	USB ファンクション機能を使用して接続対象の制御を行います。
CAN 通信	CAN(Controller Area Network)に準拠したシリアル通信の制御を行います。
タッチキー・スライダー	タッチキー・スライダーの入力信号の制御を行います。
Bluetooth HCI	RX23W を HCI モードとして動作・制御を行います。
Bluetooth Server	Bluetooth 汎用属性 (GATT) Server として動作・制御を行います。
Bluetooth Client	Bluetooth 汎用属性 (GATT) Client として動作・制御を行います。
Bluetooth Mesh	Bluetooth Mesh Networking 仕様に準拠した多対多の無線通信機能の制御を行います。

4. 機能評価

RX23W 評価キットが持つ機能の動作確認するための手順を記載します。

4.1. LED 制御

RX23W 評価用ベースボードは、3つのLEDが用意されています。

具体的な動作については『4.8 タッチキー・スライダー』にてLEDの制御を行っていますので、そちらを参照してください。

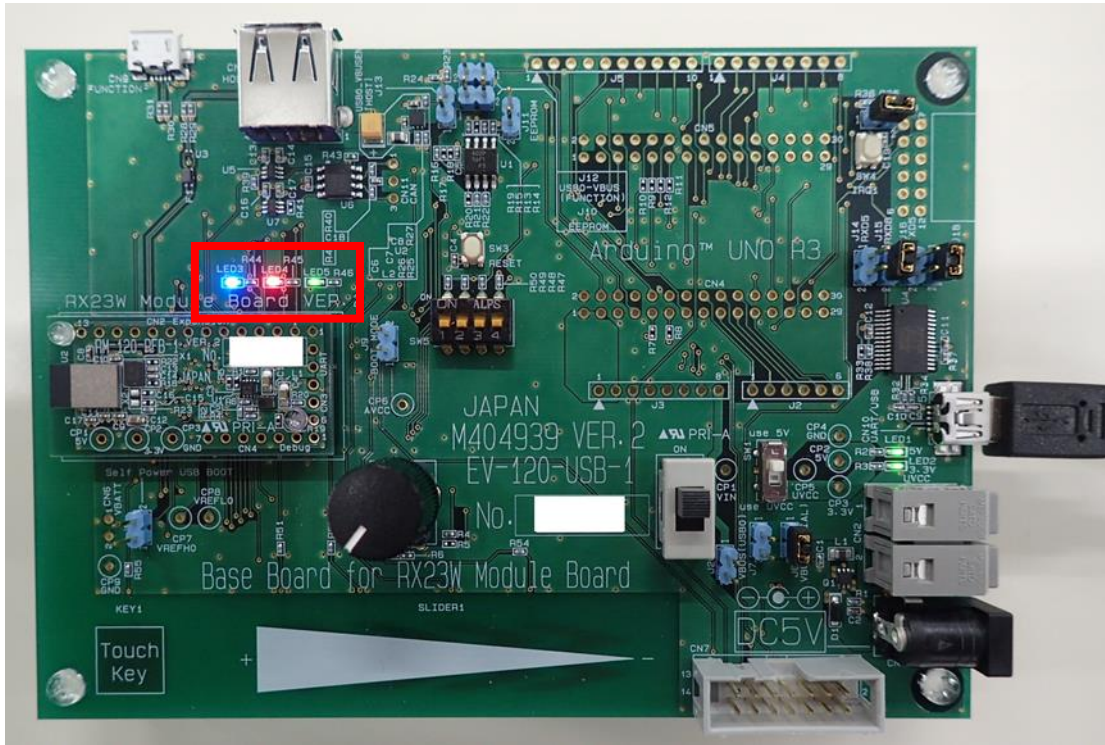


図 4-1 LED 制御

4.1.1. 使用端子

本機能で使用する端子は以下の通りです。

表 4-1 使用端子一覧

端子名	機能	説明	アクティブレベル
PE3/CLKOUT	LED3 (青)	LED3 の制御	High
PE0/AN016	LED4 (赤)	LED4 の制御	High
P03/DA0	LED5 (緑)	LED5 の制御	High

4.2. ポテンションメータ

RX23W 評価用ベースボード上のポテンショメータ (VR1) を操作することで RX23W モジュールボードの P40/AN000 端子へ入力される電圧値が変化します。

※ 本章の修正を行ったサンプルコードを使用し動作させると LED3~5 が点灯します。RSSK のボードと RX23W 評価用ベースボードでは LED の極性が異なるためです。

本機能は表 4-2 のサンプルプログラムを修正することで動作確認を行います。

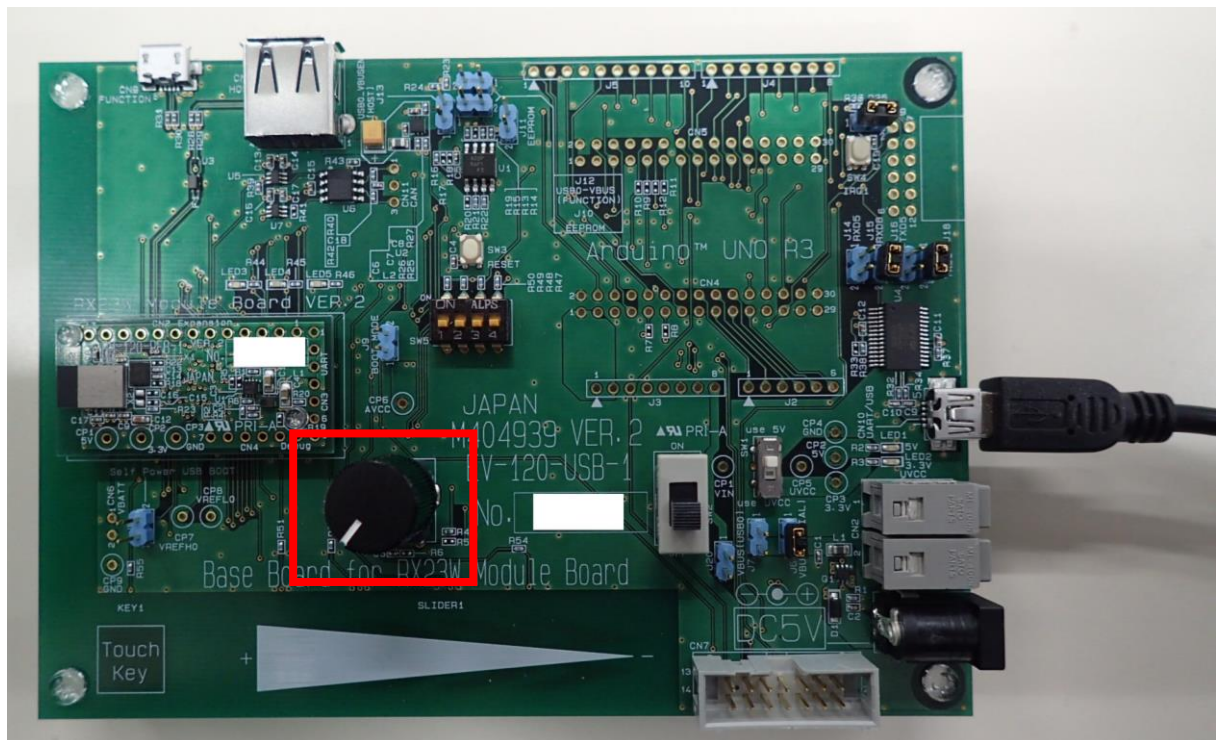


図 4-2 ポテンショメータ

表 4-2 サンプルプログラム

サンプルプログラム名	RX23W Renesas Solution Starter Kit Sample Code (e2 studio for CC-RX)
プロジェクト名	Tutorial
配布元	ルネサス エレクトロニクス株式会社

4.2.1. 機能説明

本サンプルプログラムでは、SW4 が押される (IRQ1 の割り込みが発生する)、もしくはターミナルソフト上で“C”キーが押されると、ポテンショメータ (VR1) によって変化した電圧のA/D変換結果 (AN000) をシリアルコミュニケーションインターフェース (SCI8) を使用して出力します。

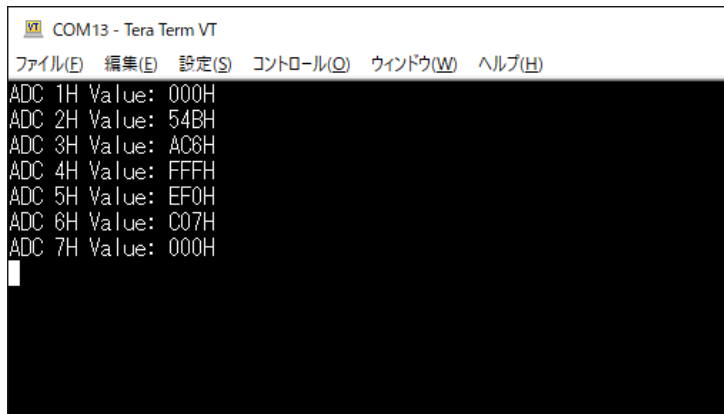


図 4-3 A/D 変換結果出力画面

4.2.2. 構成図

本機能を動作させる場合の接続構成は以下の通りです。

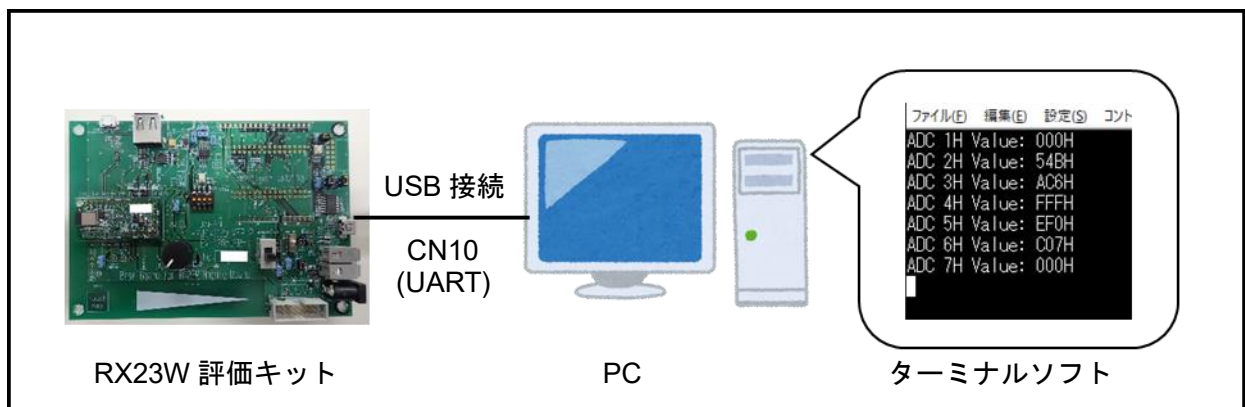


図 4-4 ポテンショメータ 接続構成図

シリアルポートの通信仕様は以下のようになります。

表 4-3 通信仕様

項目	設定
ボーレート	19200 bps
データ長	8 bit
パリティ	なし
ストップビット	1 bit
フロー制御	無し

RM-120-RFB-1	2021/11/01	SBAL-210166-00	17/55
アプリケーションノート			

4.2.3. 使用端子

本機能で使用する端子は以下の通りです。

表 4-4 使用端子一覧

端子名	機能	説明
P40/AN000	AN000	ポテンショメータ (VR1) によって変化する電圧の A/D 変換を行います。
P31/IRQ1	IRQ1	SW4 の入力を検出します
PC7/TXD8/SMOSI8	TXD8	SCI8 の送信データ出力端子
PC6/RXD8/SMISO8	RXD8	SCI8 の受信データ入力端子

4.2.4. HW の設定

RX23W 評価用ベースボード上に用意されているジャンパを次のように設定してください。

表 4-5 ジャンパ設定

端子名	設定
J6	Short
J14	Open
J15	Short
J16	Open
J17	Short

4.2.5. サンプルコード変更手順

本機能を使用するために、サンプルプログラムに対して以下の修正を行います。

No.	修正内容
1	<p>◆不要な LED 制御の削除 ファイル名: main.c main 関数の 99,103,106,109,131 行目をコメントアウトしてください。</p> <pre> 85 /* Function Name: main[] 90 @void main(void) 91 { 92 /* Initialize the switch module */ 93 R_SWITCH_Init(); 94 95 /* Set the call back function when SW1 or SW2 is pressed */ 96 R_SWITCH_SetPressCallback(cb_switch_press); 97 98 /* Initialize the debug LCD */ 99 R_LCD_Init(); 100 101 /* Displays the application name on the debug LCD. 102 Casting for use as characters. */ 103 R_LCD_Display(0, (uint8_t*)" R55KRX23W"); 104 105 /* Casting for use as characters. */ 106 R_LCD_Display(1, (uint8_t*)" Tutorial "); 107 108 /* Casting for use as characters. */ 109 R_LCD_Display(2, (uint8_t*)" Press Any Switch "); 110 111 /* Start the A/D converter */ 112 R_Config_S12AD0_Start(); 113 114 /* Set up SCIB receive buffer and callback function */ 115 R_Config_SCIB_Serial_Receive((uint8_t *)&g_rx_char, 1); 116 117 /* Enable SCIB operations */ 118 R_Config_SCIB_Start(); 119 120 while (1U) 121 { 122 uint16_t adc_result; 123 124 /* Wait for user requested A/D conversion flag to be set (SW1 or SW2) */ 125 if (TRUE == g_adc_trigger) 126 { 127 /* Call the function to perform an A/D conversion */ 128 adc_result = get_adc(); 129 130 /* Display the result on the LCD */ 131 lcd_display_adc(adc_result); 132 133 /* Increment the adc count and display using the LEDs */ 134 if (16 == (++g_adc_count)) </pre> <div style="border: 1px solid black; padding: 5px; width: fit-content; margin-left: 20px;"> <ul style="list-style-type: none"> ・ 99行目 ・ 103行目 ・ 106行目 ・ 109行目 ・ 131行目 <p>の5か所をコメントアウトする。</p> </div>

4.3. 制御スイッチ

RX23W 評価用ベースボードにはプッシュスイッチ 2 つと DIP スイッチ 4 つが実装されています。

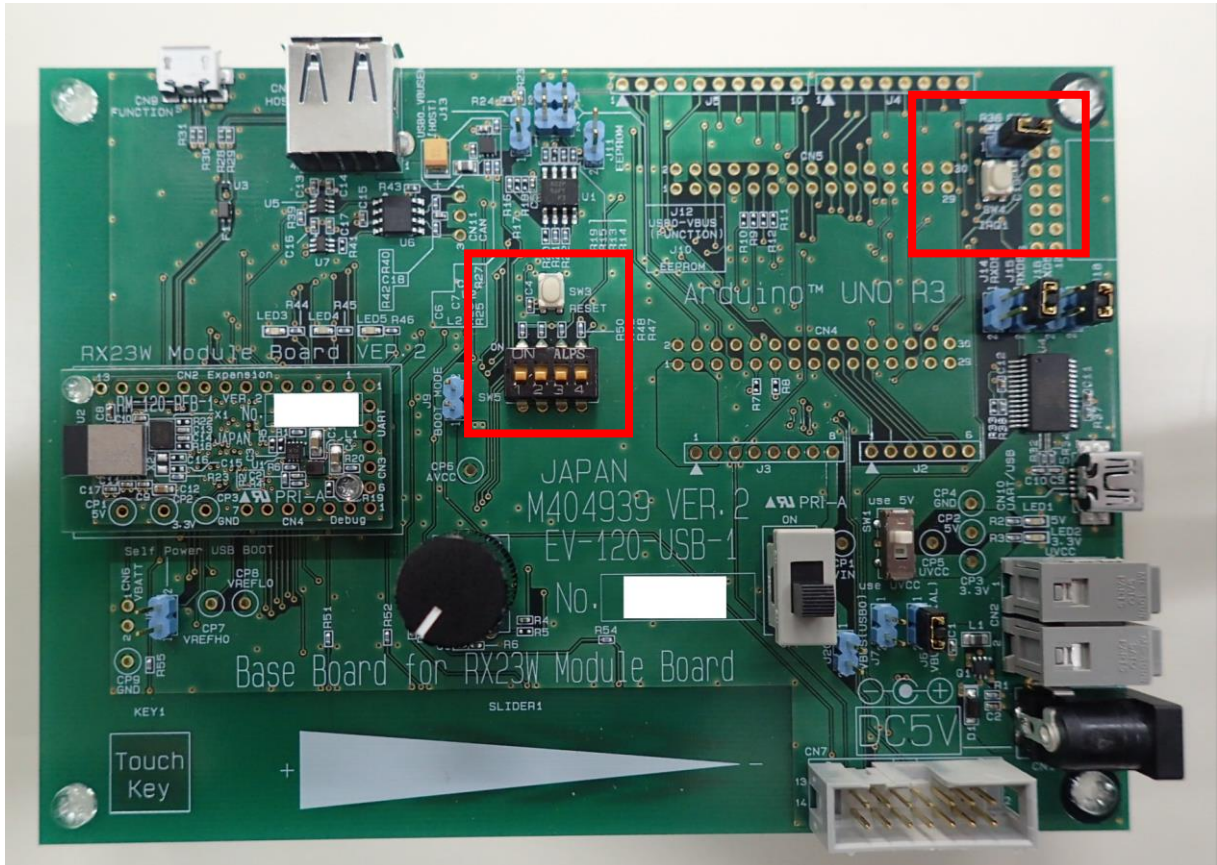


図 4-5 制御スイッチ

4.3.1. 使用端子

本機能で使用する端子は以下の通りです。

表 4-6 使用端子一覧

端子名	説明	入力信号
P31/IRQ1	SW4 の入力を検出します	ON: Low OFF: High
RES#	SW3 を ON することでデバイスをリセットします。	ON: Low OFF: High
P47/AN007	SW5-1 の入力を検出します	ON: Low OFF: High
P07/ADTRG0#	SW5-2 の入力を検出します	ON: Low OFF: High
PB0	SW5-3 の入力を検出します	ON: Low OFF: High
PC5/SCK8	SW5-4 の入力を検出します	ON: Low OFF: High

RM-120-RFB-1	2021/11/01	SBAL-210166-00	20/55
アプリケーションノート			

4.3.2. HW の設定

SW4 の使用時は RX23W 評価用ベースボード上に用意されているジャンパを次のように設定してください。

表 4-7 ジャンパ設定

端子名	設定
J19	Short

4.4. USB シリアル変換

RX23W 評価用ベースボードにはシリアルポート SCI5 及び SCI8 が USB/シリアル変換 IC に接続されており、仮想 COM ポートとして使用できます。SCI5 と SCI8 の選択はジャンパ接続で行います。

また RX23W 評価用ベースボードの電源入力として VBUS1 (シリアル / UART 変換 USB) も使用可能です。

本機能は次のサンプルプログラムを修正することで動作確認を行います。

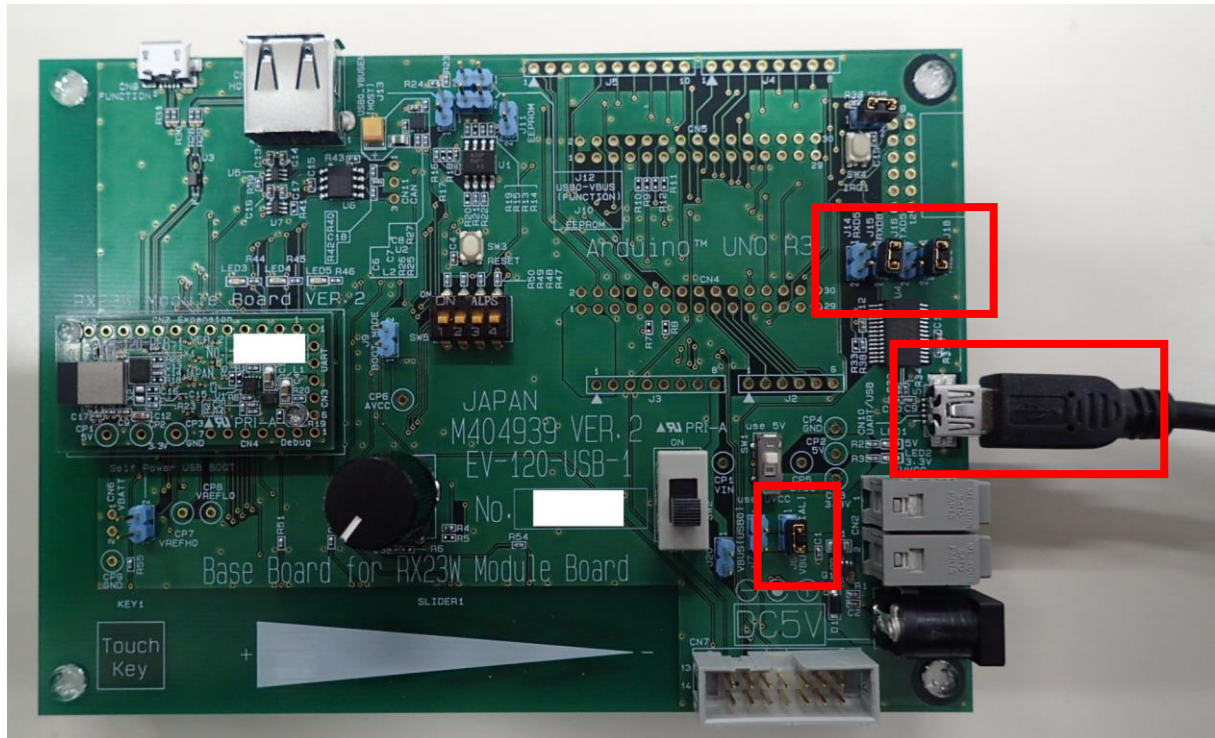


図 4-6 USB シリアル変換

表 4-8 サンプルプログラム

サンプルプログラム名	RX23W Renesas Solution Starter Kit Sample Code (e2 studio for CC-RX)
プロジェクト名	Tutorial
配布元	ルネサス エレクトロニクス株式会社

4.4.1. 機能説明

本サンプルプログラムでは、SW4 が押される (IRQ1 の割り込みが発生する)、もしくはターミナルソフト上で“C”キーが押されると、ポテンショメータ (VR1) によって変化した電圧の A/D 変換結果 (AN000) をシリアルコミュニケーションインターフェース (SCI8) を使用して出力します。

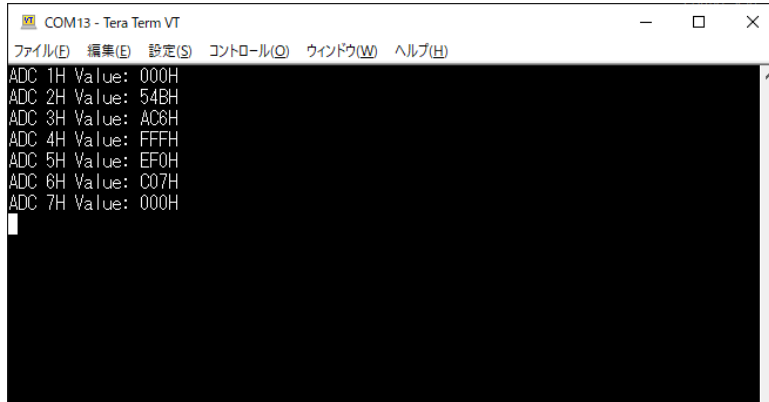


図 4-7 A/D 変換結果出力画面

4.4.2. 構成図

本機能を動作させる場合の接続構成は以下の通りです。

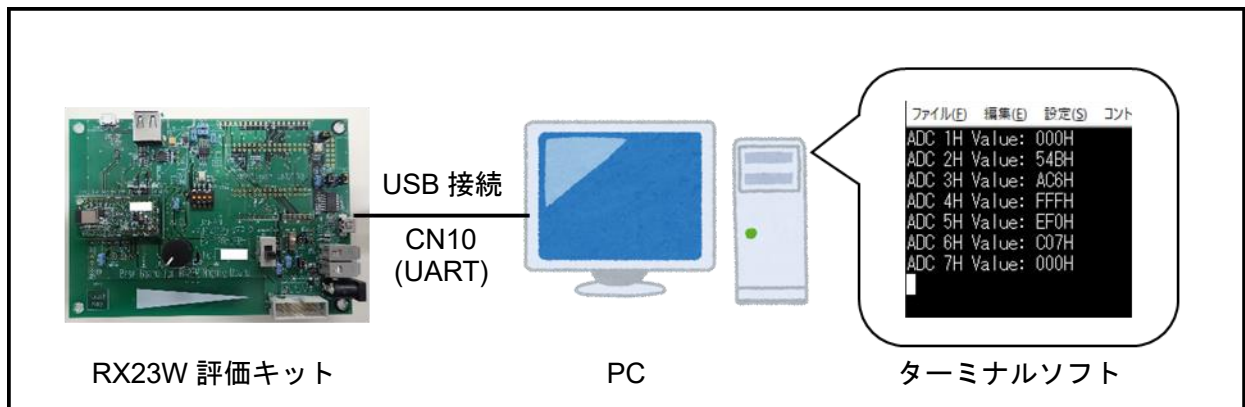


図 4-8 USB シリアル変換 接続構成図

シリアルポートの通信仕様は以下のようになります。

表 4-9 通信仕様

項目	設定
ボーレート	19200 bps
データ長	8 bit
パリティ	なし
ストップビット	1 bit
フロー制御	無し

4.4.3. 使用端子

本サンプルプログラムで使用する端子は以下の通りです。

表 4-10 使用端子一覧

端子名	機能	説明
P40/AN000	AN000	ポテンショメータ (VR1) によって変化する電圧の A/D 変換を行います。

RM-120-RFB-1	2021/11/01	SBAL-210166-00	23/55
アプリケーションノート			

P31/IRQ1	IRQ1	SW4 の入力を検出します
PC7/TXD8/SMOSI8	TXD8	SCI8 の送信データ出力端子
PC6/RXD8/SMISO8	RXD8	SCI8 の受信データ入力端子

4.4.4. HW の設定

RX23W 評価用ベースボード上に用意されているジャンパを次のように設定してください。

表 4-11 ジャンパ設定

端子名	設定
J6	Short
J14	Open
J15	Short
J16	Open
J17	Short
J19	Short

4.4.5. サンプルコード変更手順

本機能を使用するために、サンプルプログラムに対して以下の修正を行います。

No.	修正内容
1	<p>◆不要な LED 制御の削除 ファイル名: main.c main 関数の 99,103,106,109,131 行目をコメントアウトしてください。</p> <pre> 85 /* Function Name: main[] 90 @void main(void) 91 { 92 /* Initialize the switch module */ 93 R_SWITCH_Init(); 94 95 /* Set the call back function when SW1 or SW2 is pressed */ 96 R_SWITCH_SetPressCallback(cb_switch_press); 97 98 /* Initialize the debug LCD */ 99 R_LCD_Init(); 100 101 /* Displays the application name on the debug LCD. 102 Casting for use as characters. */ 103 R_LCD_Display(0, (uint8_t*)" R55KRX23W"); 104 105 /* Casting for use as characters. */ 106 R_LCD_Display(1, (uint8_t*)" Tutorial "); 107 108 /* Casting for use as characters. */ 109 R_LCD_Display(2, (uint8_t*)" Press Any Switch "); 110 111 /* Start the A/D converter */ 112 R_Config_S12AD0_Start(); 113 114 /* Set up SCIB receive buffer and callback function */ 115 R_Config_SCIB_Serial_Receive((uint8_t *)&g_rx_char, 1); 116 117 /* Enable SCIB operations */ 118 R_Config_SCIB_Start(); 119 120 while (1U) 121 { 122 uint16_t adc_result; 123 124 /* Wait for user requested A/D conversion flag to be set (SW1 or SW2) */ 125 if (TRUE == g_adc_trigger) 126 { 127 /* Call the function to perform an A/D conversion */ 128 adc_result = get_adc(); 129 130 /* Display the result on the LCD */ 131 lcd_display_adc(adc_result); 132 133 /* Increment the adc count and display using the LEDs */ 134 if (16 == (++g_adc_count)) </pre> <div style="border: 1px solid black; padding: 5px; width: fit-content; margin-left: 20px;"> <ul style="list-style-type: none"> ・ 99行目 ・ 103行目 ・ 106行目 ・ 109行目 ・ 131行目 <p>の5か所をコメントアウトする。</p> </div>

4.5. USB ホスト

USB2.0 ホスト/ ファンクションモジュールを用いて、USB ホストコントローラとして使用することができます。

本機能は次のサンプルプログラムを修正することで動作確認を行います。

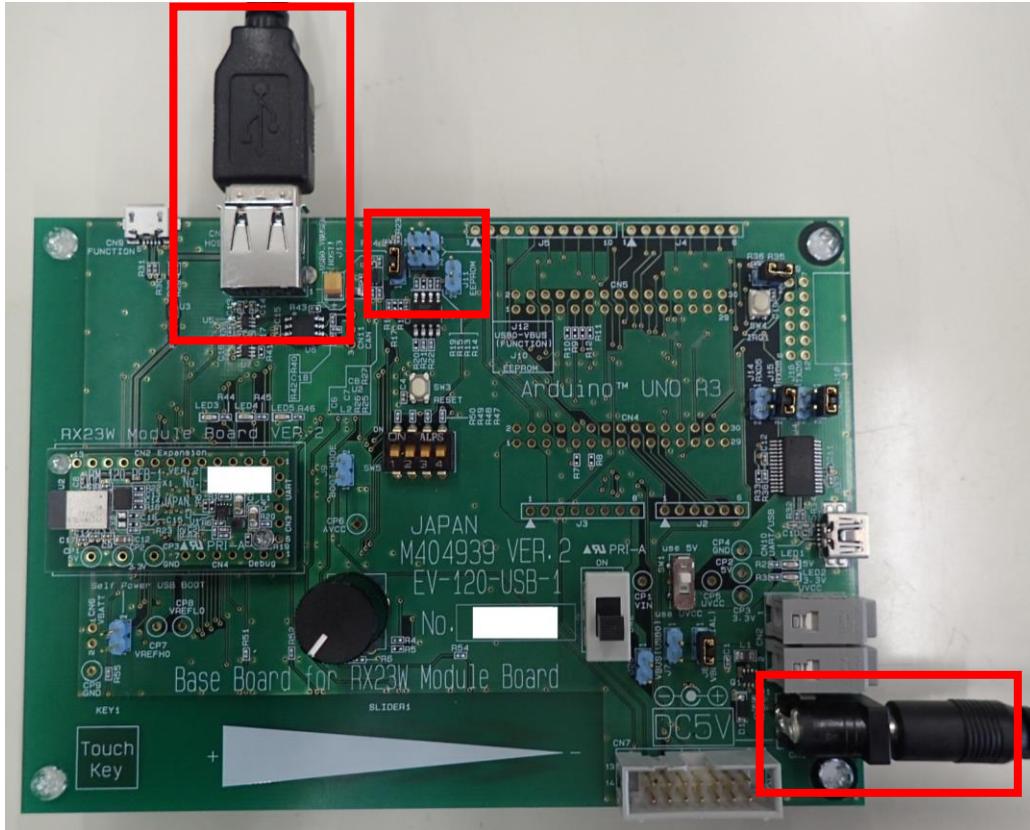


図 4-9 USB ホスト

表 4-12 サンプルプログラム

サンプルプログラム名	RX ファミリ USB ホストヒューマンインタフェースデバイスクラスドライバ for USB Mini Firmware による HID デバイスとの USB 通信を行うサンプルプログラム Firmware Integration Technology
プロジェクト名	r01an2294xx0120_usb¥workspace¥nonOS¥RX23W
配布元	ルネサス エレクトロニクス株式会社

4.5.1. 機能説明

本サンプルプログラムでは、ホストキーボードのキーが押されると値が変わる配列を判定し、ホストキーボードのスペースキーが押されている間、RX23W 評価用ベースボードの LED5 (緑) を点灯させます。

4.5.2. 構成図

本機能を動作させる場合の接続構成は以下の通りです。

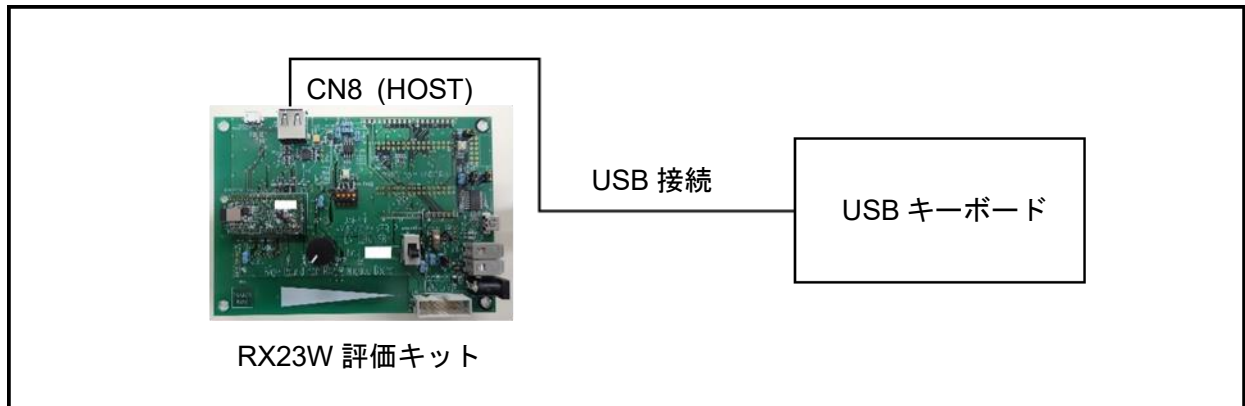


図 4-10 USB ホスト 接続構成図

4.5.1. 使用端子

本サンプルプログラムで使用する端子は以下の通りです。

表 4-13 使用端子一覧

端子名	機能	説明
P03/DA0	LED5	LED5 の駆動管理
P16/VBUSEN	USB0-VBUSEN	外部へのVBUS (5V) の供給許可信号 (ホスト)
P22/USB0_OVRCURB	USB0-OVRCURB	外部オーバカレント検出信号 (ホスト)

4.5.2. HW の設定


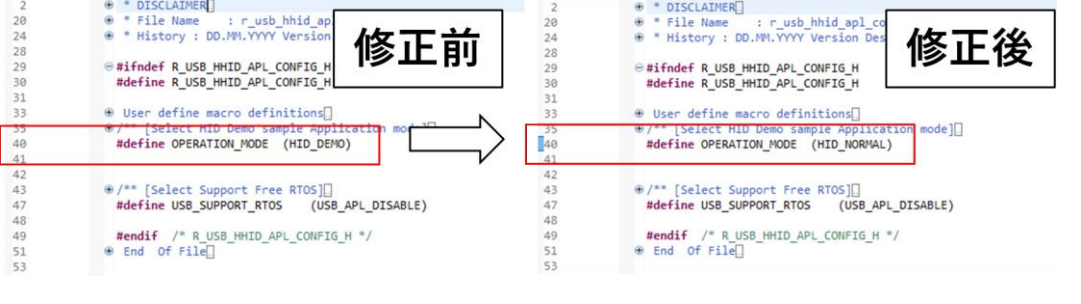
RX23W 評価用ベースボード上に用意されているジャンパを次のように設定してください。

表 4-14 ジャンパ設定

端子名	設定
J13	Short

4.5.3. サンプルコード変更手順

本機能を使用するために、サンプルプログラムに対して以下の修正を行います。

No.	修正内容
1	<p>◆ポートの変更 ファイル名: r_usb_basic_mini_pinset.c 49,50 行目の記述を修正します。</p> <ul style="list-style-type: none"> 修正前 MPC.P26PFS.BYTE = 0x11U; PORT2.PMR.BIT.B6 = 1U; 修正後 MPC.P16PFS.BYTE = 0x12U; PORT1.PMR.BIT.B6 = 1U; 
2	<p>◆動作モードの変更 ファイル名: r_usb_hhid_apl_config.h 40 行目の記述を修正します。</p> <ul style="list-style-type: none"> 修正前 #define OPERATION_MODE (HID_DEMO) 修正後 #define OPERATION_MODE (HID_NORMAL) 

3

◆LEDの初期化

ファイル名: r_usb_hhid_apl.c

129行目に記述を追加します。

・追加

PORT0.PDR.BIT.B3 = 1; /*OUT*/ //led_green

<pre> 121 ctrl.type = USB_HHID; 122 cfg.usb_mode = USB_HOST; 123 cfg.usb_speed = USB_FS; 124 R_USB_Open(&ctrl, &cfg); 125 126 #if USB_SUPPORT_RTOS == USB_APL_ENABLE 127 R_USB_Callback(usb_apl_callback); 128 #endif /* USB_SUPPORT_RTOS == USB_APL_ENABLE */ 129 while (1) 130 { 131 #if USB_SUPPORT_RTOS == USB_APL_ENABLE 132 #else /* USB_SUPPORT_RTOS == USB_APL_ENABLE */ 133 #endif /* USB_SUPPORT_RTOS == USB_APL_ENABLE */ 134 event = R_USB_GetEvent(&ctrl); 135 136 switch (event) 137 #endif /* USB_SUPPORT_RTOS == USB_APL_ENABLE */ 138 { 139 case USB_STS_CONFIGURED : </pre>	修正前	<pre> 121 ctrl.type = USB_HHID; 122 cfg.usb_mode = USB_HOST; 123 cfg.usb_speed = USB_FS; 124 R_USB_Open(&ctrl, &cfg); 125 126 #if USB_SUPPORT_RTOS == USB_APL_ENABLE 127 R_USB_Callback(usb_apl_callback); 128 #endif /* USB_SUPPORT_RTOS == USB_APL_ENABLE */ 129 PORT0.PDR.BIT.B3 = 1; /*OUT*/ //led_green 130 while (1) 131 { 132 #if USB_SUPPORT_RTOS == USB_APL_ENABLE 133 #else /* USB_SUPPORT_RTOS == USB_APL_ENABLE */ 134 #endif /* USB_SUPPORT_RTOS == USB_APL_ENABLE */ 135 event = R_USB_GetEvent(&ctrl); 136 137 switch (event) 138 #endif /* USB_SUPPORT_RTOS == USB_APL_ENABLE */ 139 { 140 case USB_STS_CONFIGURED : </pre>	修正後
---	-----	---	-----

4

◆LEDの動作追加

ファイル名: r_usb_hhid_apl.c

155行目に記述を追加します。

・追加

```

if(g_data[2] == 44){
    PORT0.PODR.BIT.B3 = 1;
}else{
    PORT0.PODR.BIT.B3 = 0;
}

```

<pre> 152 case USB_STS_READ_COMPLETE : 153 R_USB_Read(&ctrl, (uint8_t *) g_data, get_size()); 154 break; 155 case USB_STS_REQUEST_COMPLETE : 156 if (USB_HID_SET_PROTOCOL == (ctrl.setup.type & USB_BREQU 157 { 158 ctrl.type = USB_HHID; 159 R_USB_Read(&ctrl, (uint8_t *) g_data, get_size()); 160 } 161 break; 162 default : 163 break; </pre>	修正前	<pre> 152 case USB_STS_READ_COMPLETE : 153 R_USB_Read(&ctrl, (uint8_t *) g_data, get_size()); 154 break; 155 case USB_STS_REQUEST_COMPLETE : 156 if(g_data[2] == 44){ 157 PORT0.PODR.BIT.B3 = 1; 158 }else{ 159 PORT0.PODR.BIT.B3 = 0; 160 } 161 break; 162 case USB_STS_REQUEST_COMPLETE : 163 if (USB_HID_SET_PROTOCOL == (ctrl.setup.type & USB_BREQU 164 { 165 ctrl.type = USB_HHID; 166 R_USB_Read(&ctrl, (uint8_t *) g_data, get_size()); </pre>	修正後
---	-----	--	-----

4.6. USB ファンクション

USB2.0 ホスト/ ファンクションモジュールを用いて、USB ファンクションコントローラとして使用することができます。

本機能は次のサンプルプログラムを修正することで動作確認を行います。

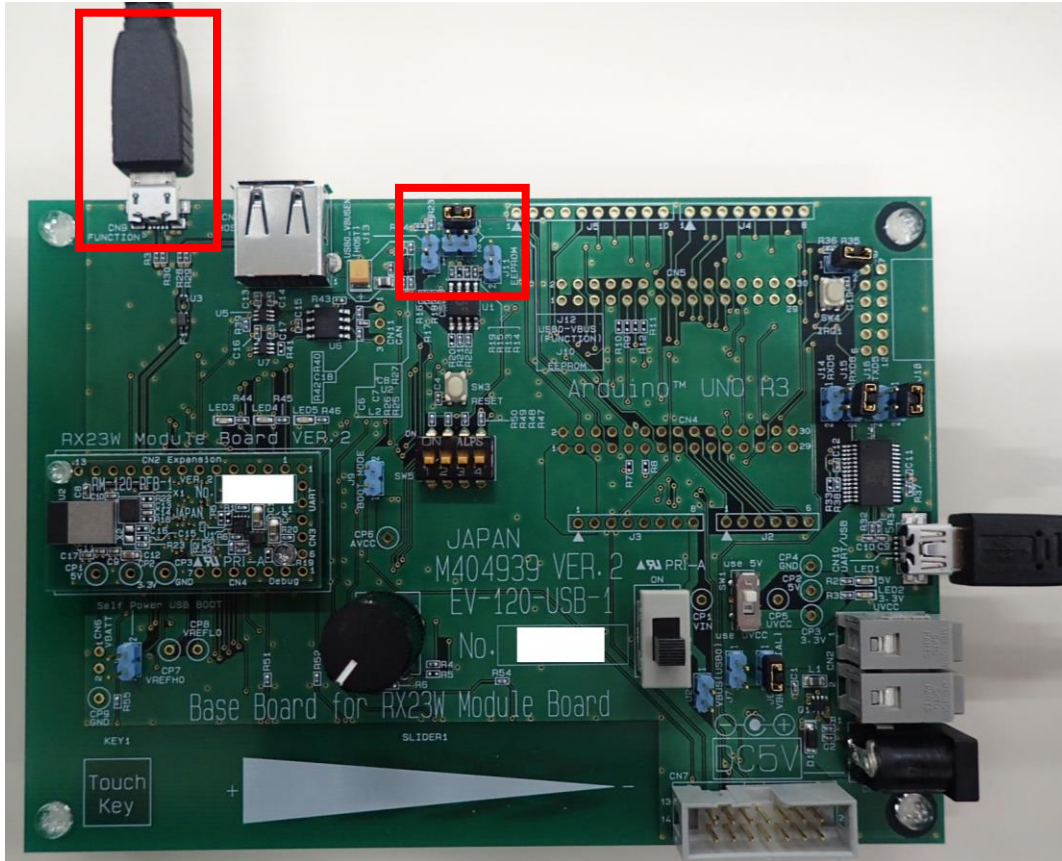


図 4-11 USB ファンクション

表 4-15 サンプルプログラム

サンプルプログラム名	RX ファミリ 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) for USB Mini Firmware 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
プロジェクト名	r01an2296xx0120_usb¥workspace¥nonOS¥RX23W
配布元	ルネサス エレクトロニクス株式会社

4.6.1. 機能説明

本サンプルプログラムでは、PC と RX23W 評価用ベースボードの CN9 を USB ケーブルで接続し、TeraTerm 上でキーを押下すると文字が返ってくるループバック処理をおこないます。

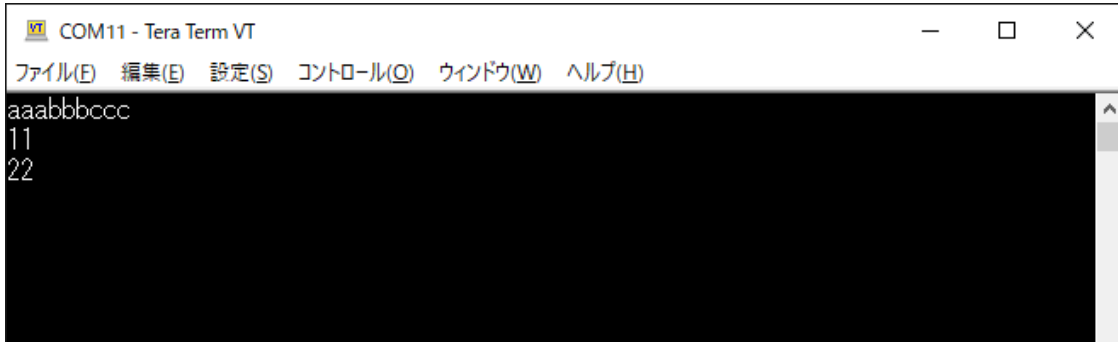


図 4-12 ターミナルソフト動作例

4.6.2. 構成図

本機能を動作させる場合の接続構成は以下の通りです。

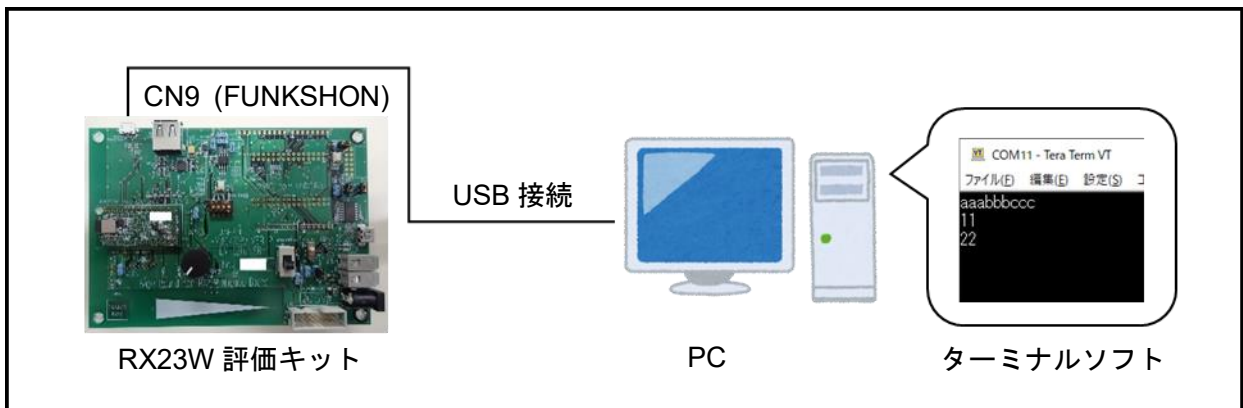


図 4-13 USB ファンクション 接続構成図

4.6.3. 使用端子

本サンプルプログラムで使用する端子は以下の通りです。

表 4-16 使用端子一覧

端子名	機能	説明
P16/ VBUS (USB0)	USB0-VBUS	USB ケーブル接続モニタ端子 (ファンクション)

4.6.4. HW の設定

RX23W 評価用ベースボード上に用意されているジャンパを次のように設定してください。

表 4-17 ジャンパ設定

端子名	設定
J12	Short

4.7. CAN 通信

CAN(Controller Area Network)に準拠したシリアル通信の制御を行います。
本機能は次のサンプルプログラムを修正することで動作確認を行います

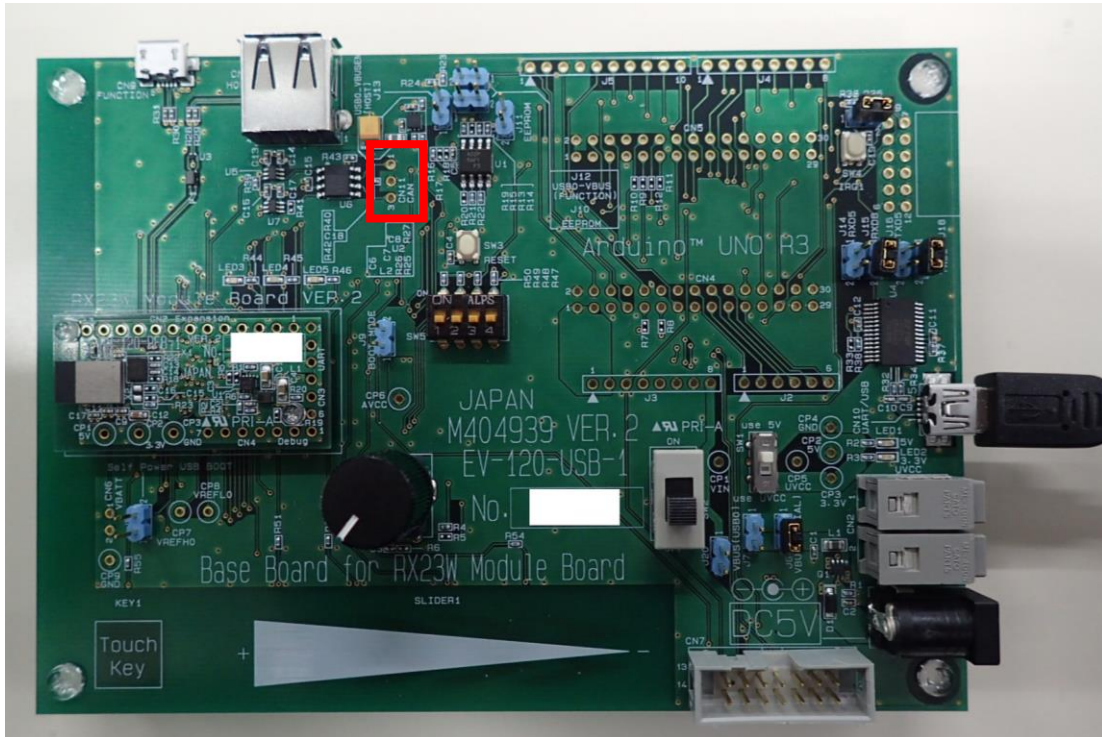


図 4-14 ターミナルソフト動作例

表 4-18 サンプルプログラム

サンプルプログラム名	RX Family RSCAN Module Using Firmware Integration Technology
プロジェクト名	rscan_demo_rx231
配布元	ルネサス エレクトロニクス株式会社

4.7.1. 機能説明

CAN 通信に対応した通信対象と用意することで、データ通信を行うことが可能です。
動作の詳細は、「RX ファミリー RSCAN モジュール Firmware Integration Technology」の 5 章を確認してください。

4.7.2. 構成図

本機能を動作させる場合の接続構成は以下の通りです。



図 4-15 CAN 接続構成図

4.7.3. 使用端子

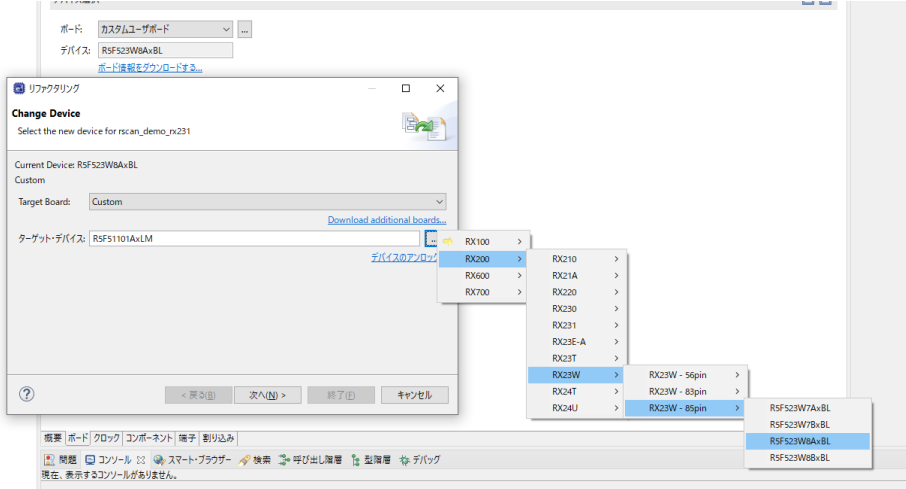
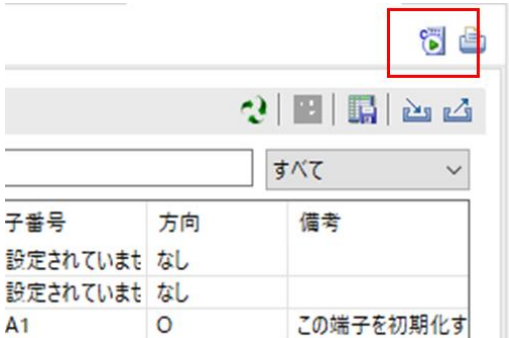
本サンプルプログラムで使用する端子は以下の通りです。

表 4-19 使用端子一覧

端子名	機能	説明
P14/CTXD0	CTXD0	CAN データ送信
P15/CRXD0	CRXD0	CAN データ受信

4.7.4. サンプルコード変更手順

本機能を使用するために、サンプルプログラムに対して以下の修正を行います。

No.	修正内容
1	<p>◆ターゲットデバイスの変更 スマートコンフィグレータの修正 修正箇所: ボード→カスタムユーザボード横の"..."→ ターゲットデバイス→R5F523W8AxBL を選択</p> 
2	<p>ターゲットデバイス変更後、スマートコンフィグレータ右上の自動生成をクリックして下さい。</p> 
3	<p>◆ポートの変更 ファイル名 : main.c 282～290 行目の記述を修正します。</p> <ul style="list-style-type: none"> 修正前 <pre> PORT5.PODR.BIT.B5 = 1; PORT5.PODR.BIT.B4 = 0; MPC.P54PFS.BYTE = 0x10; // Pin Func Select P54 CTXD0 MPC.P55PFS.BYTE = 0x10; // Pin Func Select P55 CRXD0 PORT5.PDR.BIT.B4 = 1; // set TX pin direction to output PORT5.DSCR.BIT.B4 = 1; // High-drive output PORT5.PDR.BIT.B5 = 0; // set RX pin direction to input (dflt) PORT5.PMR.BIT.B4 = 1; // set TX pin mode to peripheral </pre>

```
PORT5.PMR.BIT.B5 = 1; // set RX pin mode to peripheral
```

修正後

```
PORT1.PODR.BIT.B5 = 1;
PORT1.PODR.BIT.B4 = 0;
MPC.P14PFS.BYTE = 0x10; // Pin Func Select P14 CTXD0
MPC.P15PFS.BYTE = 0x10; // Pin Func Select P15 CRXD0
PORT1.PDR.BIT.B4 = 1; // set TX pin direction to output
PORT1.DSCR.BIT.B4 = 1; // High-drive output
PORT1.PDR.BIT.B5 = 0; // set RX pin direction to input (dflt)
PORT1.PMR.BIT.B4 = 1; // set TX pin mode to peripheral
PORT1.PMR.BIT.B5 = 1; // set RX pin mode to peripheral
```

修正前

```
271  /* Function Name: can_init_ports[]
272  =void can_init_ports(void)
273  {
274
275      R_BSP_RegisterProtectDisable(BSP_REG_PROTECT_MPC);
276
277      /* init CAN channel @ */
278
279      PORT5.PODR.BIT.B5 = 1;
280      PORT5.PODR.BIT.B4 = 0;
281      MPC.P14PFS.BYTE = 0x10; // Pin Func Select P14 CTXD0
282      MPC.P15PFS.BYTE = 0x10; // Pin Func Select P15 CRXD0
283      PORT5.PDR.BIT.B4 = 1; // set TX pin direction to output
284      PORT5.DSCR.BIT.B4 = 1; // High-drive output
285      PORT5.PDR.BIT.B5 = 0; // set RX pin direction to input (dflt)
286      PORT5.PMR.BIT.B4 = 1; // set TX pin mode to peripheral
287      PORT5.PMR.BIT.B5 = 1; // set RX pin mode to peripheral
288  }
```

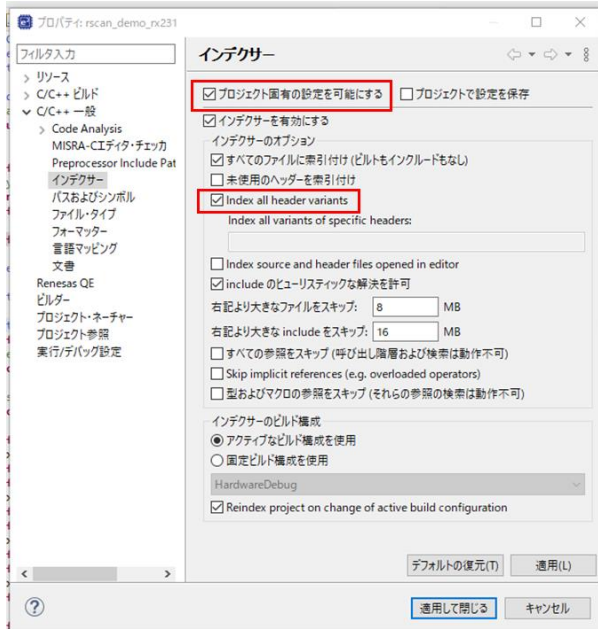
修正後

```
271  /* Function Name: can_init_ports[]
272  =void can_init_ports(void)
273  {
274
275      R_BSP_RegisterProtectDisable(BSP_REG_PROTECT_MPC);
276
277      /* init CAN channel @ */
278
279      PORT1.PODR.BIT.B5 = 1;
280      PORT1.PODR.BIT.B4 = 0;
281      MPC.P14PFS.BYTE = 0x10; // Pin Func Select P14 CTXD0
282      MPC.P15PFS.BYTE = 0x10; // Pin Func Select P15 CRXD0
283      PORT1.PDR.BIT.B4 = 1; // set TX pin direction to output
284      PORT1.DSCR.BIT.B4 = 1; // High-drive output
285      PORT1.PDR.BIT.B5 = 0; // set RX pin direction to input (dflt)
286      PORT1.PMR.BIT.B4 = 1; // set TX pin mode to peripheral
287      PORT1.PMR.BIT.B5 = 1; // set RX pin mode to peripheral
288  }
```

4

◆エラー対処

シンボル'○○'が解決できません」というコード解析エラーが出る場合
プロジェクト→プロパティ→C/C++一般→インデクサーの項目を変更し
適用後再ビルド



4.8. タッチキー・スライダー

静電容量式タッチセンサユニットを用いて、タッチキー・スライダーの制御を行います。

本機能は次のサンプルプログラムを修正することで動作確認を行います。

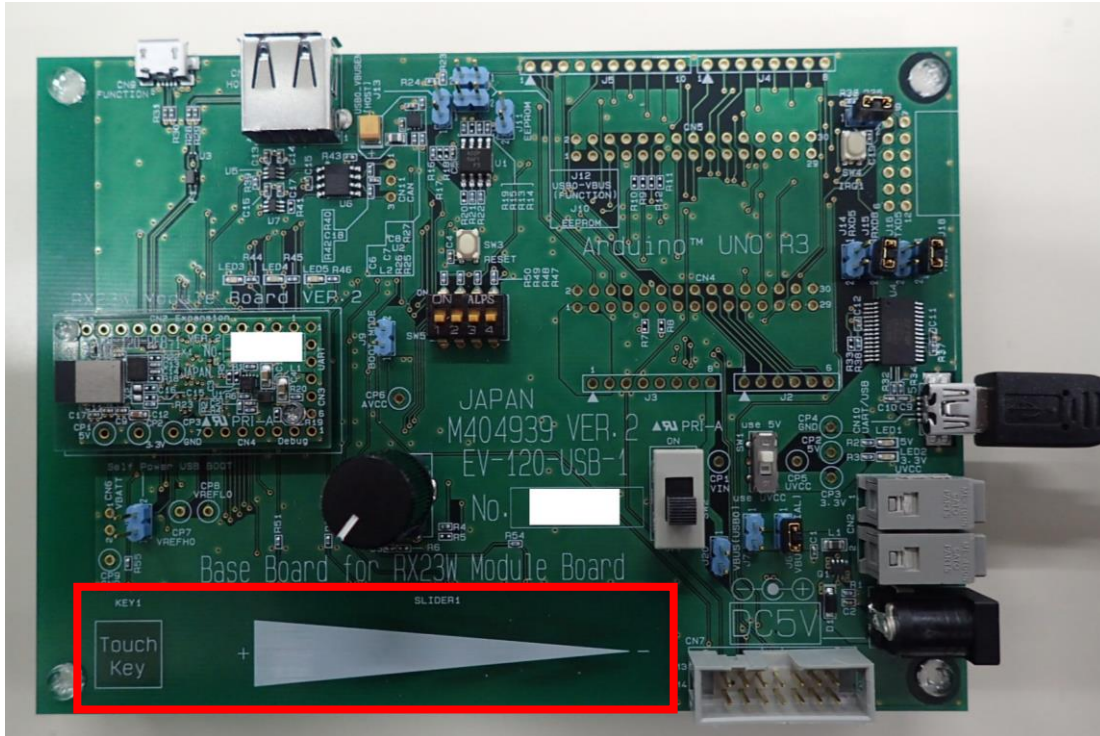


図 4-16 タッチキー・スライダー

表 4-20 サンプルプログラム

サンプルプログラム名	RX ファミリ QE Touch モジュール Firmware Integration Technology アプリケーションノート
プロジェクト名	touch_demo_rsskrx23w
配布元	ルネサス エレクトロニクス株式会社

4.8.1. 機能説明

本サンプルプログラムでは、タッチキーを押下している間緑のLEDが点灯し、スライダーを押下している間押している位置に対応したLEDが点灯します。

4.8.2. 構成図

本機能を動作させる場合の接続構成は以下の通りです。

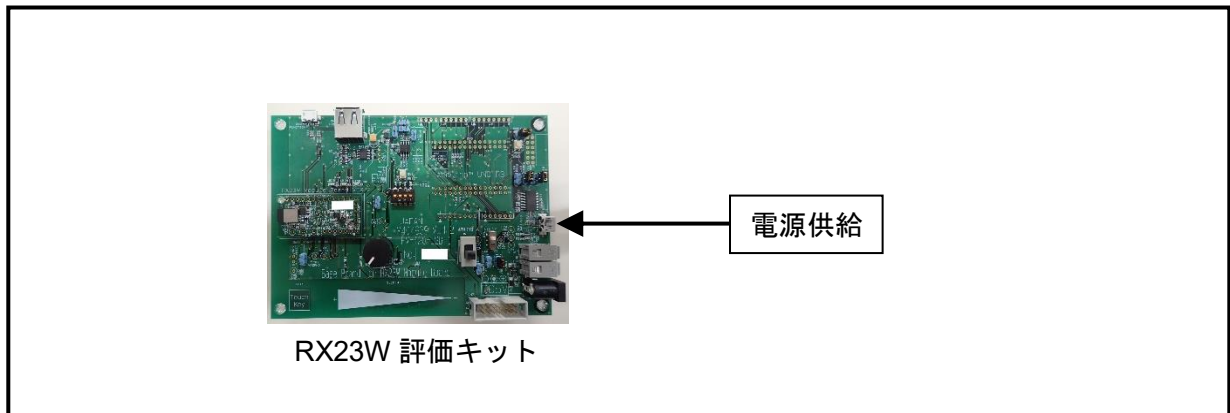


図 4-14 タッチキー・スライダー 接続構成図

4.8.3. 使用端子

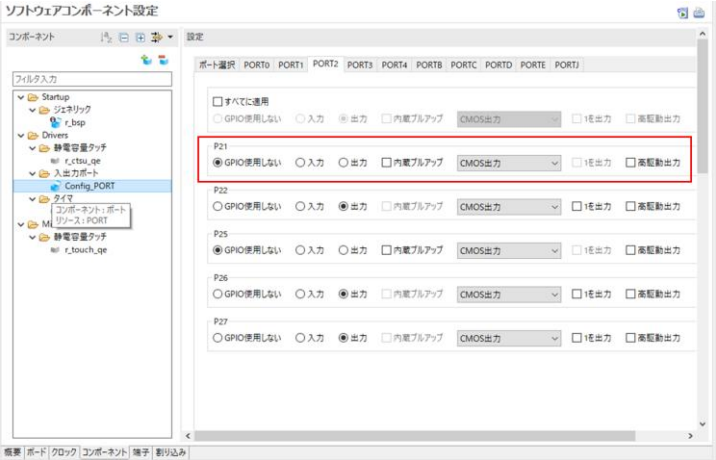
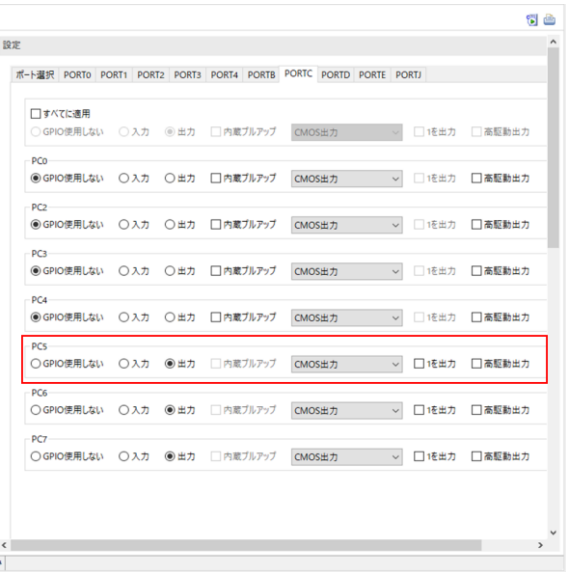
本機能を動作させる場合の接続構成は以下の通りです。

表 4-21 使用端子一覧

端子名	機能	説明
PE3/CLKOUT	LED3	LED3 の駆動管理
PE0/AN016	LED4	LED4 の駆動管理
P03/DA0	LED5	LED5 の駆動管理
P25/TS4	静電容量計測端子	タッチキー
P21/TS8	静電容量計測端子	タッチスライダー1
PC3/TS27/TXD5	静電容量計測端子	タッチスライダー2
PC2/TS30/RXD5	静電容量計測端子	タッチスライダー3
PC0/TS35/CTS5#	静電容量計測端子	タッチスライダー4
PC4/TSCAP	Low-pass filter 接続用端子	Low-pass filter 接続用端子

4.8.4. サンプルコード変更手順

本機能を使用するために、サンプルプログラムに対して以下の修正を行います。

No.	修正内容
1	<p>◆ポートの変更 スマートコンフィグレータの修正 1 修正箇所: コンポーネント→入出力ポート→Config_PORT “P21 GPIO 使用しない”にチェックを入れて下さい。</p> 
2	<p>◆ポートの変更 スマートコンフィグレータの修正 2 修正箇所: コンポーネント→入出力ポート→Config_PORT。 “PC5 出力”にチェックを入れて下さい。</p> 

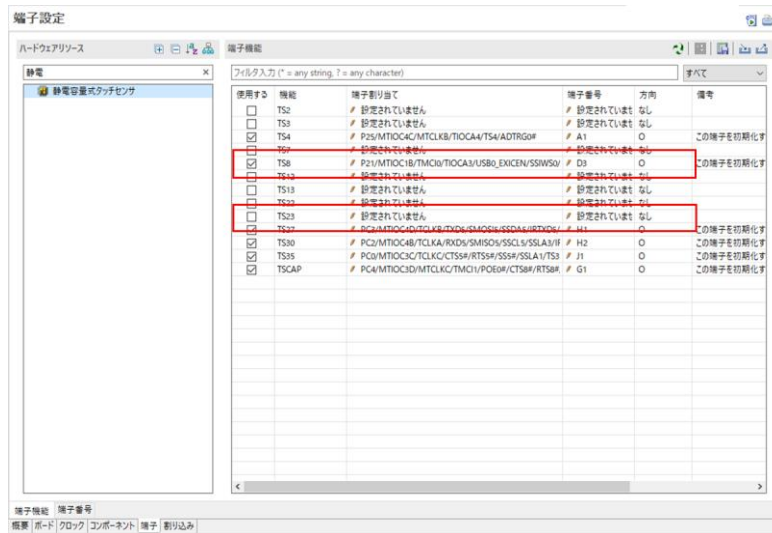
3

◆ポートの変更

スマートコンフィグレータの修正 3

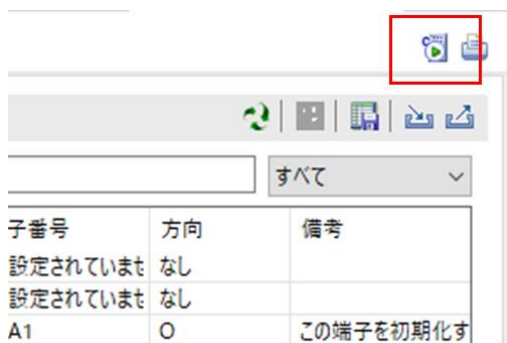
修正箇所: 端子→静電容量式タッチセンサ

TS8 にチェックを入れ、TS23 のチェックを外して下さい。



4

上記 3 項目完了後、スマートコンフィグレータ右上の自動生成をクリックして下さい。



5 ◆ポートの変更

ファイル名: r_ctsu_qe_pinset.c
55~57 行目の記述を修正します。

・修正前

```
/* Set TS23 pin */
MPC.PC5PFS.BYTE = 0x19U;
PORTC.PMR.BIT.B5 = 1U;
```

・修正後

```
/* Set TS8 pin */
MPC.P21PFS.BYTE = 0x19U;
PORT2.PMR.BIT.B1 = 1U;
```

<pre>43 void R_CTSU_PinSetInit() 44 { 45 R_BSP_RegisterProtectDisable(46 47 /* Set TSCAP pin */ 48 MPC.PC4PFS.BYTE = 0x19U; 49 PORTC.PMR.BIT.B4 = 1U; 50 51 /* Set TS4 pin */ 52 MPC.P25PFS.BYTE = 0x19U; 53 PORT2.PMR.BIT.B5 = 1U; 54 55 /* Set TS23 pin */ 56 MPC.PC5PFS.BYTE = 0x19U; 57 PORTC.PMR.BIT.B5 = 1U; 58 59 /* Set TS27 pin */ 60 MPC.PC3PFS.BYTE = 0x19U; 61 PORTC.PMR.BIT.B3 = 1U;</pre>	修正前	→	<pre>43 void R_CTSU_PinSetInit() 44 { 45 R_BSP_RegisterProtectDisable(46 47 /* Set TSCAP pin */ 48 MPC.PC4PFS.BYTE = 0x19U; 49 PORTC.PMR.BIT.B4 = 1U; 50 51 /* Set TS4 pin */ 52 MPC.P25PFS.BYTE = 0x19U; 53 PORT2.PMR.BIT.B5 = 1U; 54 55 /* Set TS8 pin */ 56 MPC.P21PFS.BYTE = 0x19U; 57 PORT2.PMR.BIT.B1 = 1U; 58 59 /* Set TS27 pin */ 60 MPC.PC3PFS.BYTE = 0x19U; 61 PORTC.PMR.BIT.B3 = 1U;</pre>	修正後
--	-----	---	---	-----

6 ◆LEDの初期化

ファイル名: touch_demo_rsskrx23w.c
66 行目に記述を追加します。

・追加

```
/*Set led*/
PORTE.PDR.BIT.B3 = 1;
PORTE.PDR.BIT.B0 = 1;
PORT0.PDR.BIT.B3 = 1;
```

<pre>49 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74</pre>	修正前	→	<pre>49 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74</pre>	修正後
---	-----	---	---	-----

7

◆LEDの動作追加

ファイル名: touch_demo_rsskrx23w.c

R_TOUCH_UpdateDataAndStartScan(); から下の行を修正します。

・修正前

```
R_TOUCH_GetAllBtnStates(QE_METHOD_CONFIG01, &btn_states);
if (btn_states & CONFIG01_MASK_BUTTON00)
{
    nop(); // button pressed; do something
}
```

```
R_TOUCH_GetSliderPosition(CONFIG01_ID_SLIDER00, &sldr_pos);
if (sldr_pos != 65535)
{
    nop(); // slider touched; do something
}
```

・修正後

```
R_TOUCH_GetAllBtnStates(QE_METHOD_CONFIG01, &btn_states);
R_TOUCH_GetSliderPosition(CONFIG01_ID_SLIDER00, &sldr_pos);
if (btn_states & CONFIG01_MASK_BUTTON00){
    // led mode
    PORTE.PODR.BIT.B3 = 0; //Blue end
    PORTE.PODR.BIT.B0 = 0; //Red end
    PORT0.PODR.BIT.B3 = 1; //Green start
}else{
    if(sldr_pos != 65535){
        if (sldr_pos < 21){
            PORTE.PODR.BIT.B0 = 0; //Red end
            PORT0.PODR.BIT.B3 = 0; //Green end
            PORTE.PODR.BIT.B3 = 1; //Blue start
        }else if(sldr_pos < 41){
            PORT0.PODR.BIT.B3 = 0; //Green end
            PORTE.PODR.BIT.B3 = 1; //Blue start
            PORTE.PODR.BIT.B0 = 1; //Red start
        }else if(sldr_pos < 61){
            PORTE.PODR.BIT.B3 = 0; //Blue end
            PORT0.PODR.BIT.B3 = 0; //Green end
            PORTE.PODR.BIT.B0 = 1; //Red start
        }else if(sldr_pos < 81){
            PORTE.PODR.BIT.B3 = 0; //Blue end
            PORTE.PODR.BIT.B0 = 1; //Red start
            PORT0.PODR.BIT.B3 = 1; //Green start
        }else if(sldr_pos < 101){
            PORTE.PODR.BIT.B3 = 0; //Blue end
            PORTE.PODR.BIT.B0 = 0; //Red end
            PORT0.PODR.BIT.B3 = 1; //Green start
        }
    }
}else{
    PORTE.PODR.BIT.B3 = 0; //Blue end
    PORTE.PODR.BIT.B0 = 0; //Red end
}
```



```
PORT0.PODR.BIT.B3 = 0; //Green end
```

修正前

修正後

```

85     }
86     }
87     }
88     {
89     if (g_getouch_timer_flg == true)
90     {
91     g_getouch_timer_flg = false;
92     R_TOUCH_UpdateDataAndStartScan();
93     }
94     /* Put "btn_states" and "sldr_pos" in Expressions window with
95     * enabled and observe values change as slider and buttons
96     */
97     R_TOUCH_GetAllBtnStates(QE_METHOD_CONFIG01, &btn_states);
98     if (btn_states & CONFIG01_MASK_BUTTON00)
99     {
100     nop(); // button pressed; do something
101     }
102     R_TOUCH_GetSliderPosition(CONFIG01_ID_SLIDER00, &sldr_pos);
103     if (sldr_pos != 65535)
104     {
105     nop(); // slider touched; do something
106     }
107     }
108     }
109     }
110     }
111     }
112     }
113     }
114     }
115     }
116     }
117     }
118     }
119     }
120     }
121     }
122     }
123     }
124     }
125     }
126     }
127     }
128     }
129     }
130     }
131     }

```

```

97     R_TOUCH_GetAllBtnStates(QE_METHOD_CONFIG01, &btn_states);
98     if (btn_states & CONFIG01_MASK_BUTTON00)
99     {
100     nop(); // button pressed; do something
101     }
102     R_TOUCH_GetSliderPosition(CONFIG01_ID_SLIDER00, &sldr_pos);
103     if (sldr_pos != 65535)
104     {
105     nop(); // slider touched; do something
106     }
107     }
108     }

```

→

```

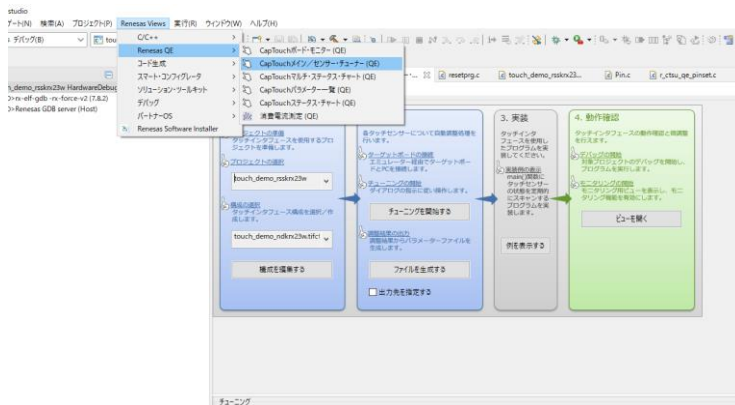
97     R_TOUCH_GetAllBtnStates(QE_METHOD_CONFIG01, &btn_states);
98     R_TOUCH_GetSliderPosition(CONFIG01_ID_SLIDER00, &sldr_pos);
99     if (btn_states & CONFIG01_MASK_BUTTON00){
100     // led mode
101     PORT0.PODR.BIT.B3 = 0; //Blue end
102     PORT0.PODR.BIT.B0 = 0; //Red end
103     PORT0.PODR.BIT.B3 = 1; //Green start
104     }else{
105     if(sldr_pos != 65535){
106     if (sldr_pos < 21){
107     PORT0.PODR.BIT.B0 = 0; //Red end
108     PORT0.PODR.BIT.B3 = 0; //Green end
109     PORT0.PODR.BIT.B3 = 1; //Blue start
110     }else if(sldr_pos < 41){
111     PORT0.PODR.BIT.B3 = 0; //Green end
112     PORT0.PODR.BIT.B0 = 1; //Blue start
113     PORT0.PODR.BIT.B0 = 1; //Red start
114     }else if(sldr_pos < 61){
115     PORT0.PODR.BIT.B3 = 0; //Blue end
116     PORT0.PODR.BIT.B3 = 0; //Green end
117     PORT0.PODR.BIT.B0 = 1; //Red start
118     }else if(sldr_pos < 81){
119     PORT0.PODR.BIT.B3 = 0; //Blue end
120     PORT0.PODR.BIT.B0 = 1; //Red start
121     PORT0.PODR.BIT.B3 = 1; //Green start
122     }else if(sldr_pos < 101){
123     PORT0.PODR.BIT.B3 = 0; //Blue end
124     PORT0.PODR.BIT.B0 = 0; //Red end
125     PORT0.PODR.BIT.B3 = 1; //Green start
126     }
127     }
128     }else{
129     PORT0.PODR.BIT.B3 = 0; //Blue end
130     PORT0.PODR.BIT.B0 = 0; //Red end
131     PORT0.PODR.BIT.B3 = 0; //Green end
132     }
133     }

```

4.8.5. タッチセンサチューニング

タッチセンサのチューニングをおこないます。静電容量式タッチセンサ対応開発支援ツール QE for Capacitive Touch を使用します。ルネサスエレクトロニクス株式会社のホームページからインストールして下さい。インストール方法は QE for Capacitive Touch V1.1.0 リリースノートをご確認ください。

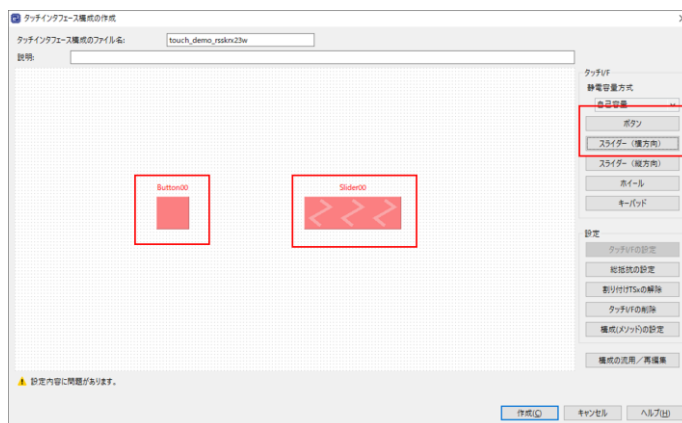
インストール後、ツールバーから”Renesas Views”→”Renesas QE”→”CapTouch メイン/センサー・チューナー(QE)”を選択し、タブを表示します。



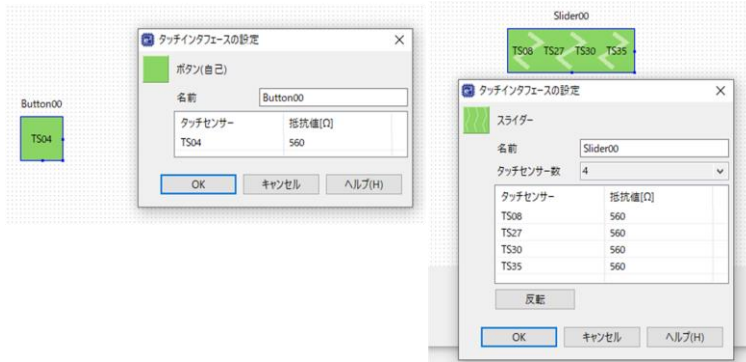
“構成を編集する”をクリックし、構成画面を表示させます。

構成の選択では、現在開いているプロジェクト名を選択します。作成していない場合、新規作成を選択して下さい。

“ボタン”、“スライダー (横方向)”をクリックし、画面上にインタフェースを配置します。

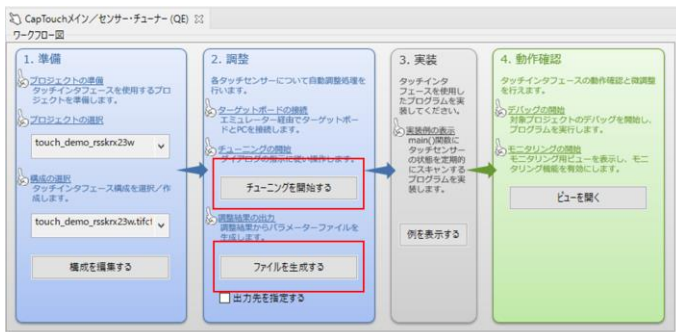


それぞれダブルクリックし、図のように設定します。完了後右下の“作成”をクリックし、構成画面を閉じます。



エミュレータと RX23W 評価用ベースボードを接続し、RX23W 評価用ベースボードの電源を入れた後“チューニングを開始する”をクリックし、画面の指示に従ってチューニングをおこないます。

完了後に“ファイルを生成する”をクリックし、再びビルドをおこないます。



以上でチューニングは終了です。

QE for Capacitive Touch のその他操作の詳細は e2studio 内のヘルプをご確認ください。

4.9. Bluetooth HCI

Bluetooth Low Energy プロトコルスタックを使用して、Bluetooth 機能を制御します。HCI(Host Controller Interface)モードは PC などのシリアルインタフェース接続されたホストデバイスから HCI コマンドを用いて BLE 通信を行うことが可能です。本機能は次のサンプルプログラムを修正することで動作確認を行います。

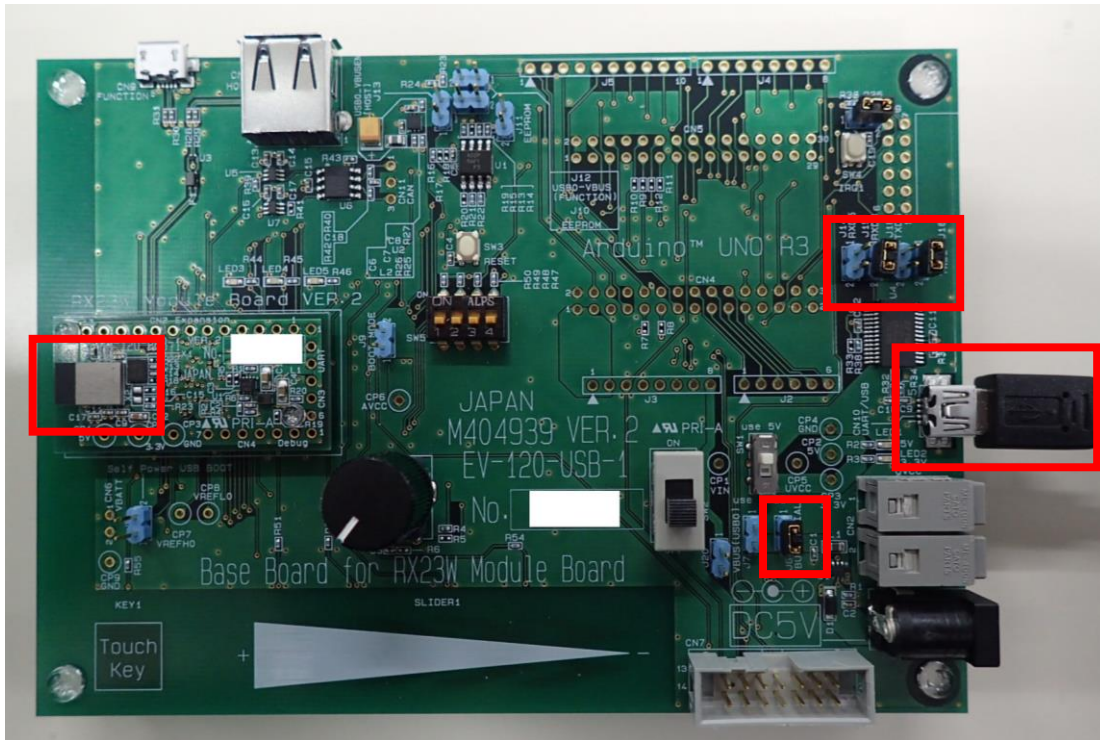


図 4-17 Bluetooth HCI

表 4-22 サンプルプログラム

サンプルプログラム名	RX23W Group BLE Module Firmware Integration Technology Application Note r01an4860xx0100-rx23w-ble-fit.zip
プロジェクト名	ble_demo_rsskrx23w_uart_hci
配布元	ルネサス エレクトロニクス株式会社

4.9.1. 機能説明

ルネサスエレクトロニクス株式会社より提供されている『Bluetooth Low Energy プロトコルスタック』と同等の動作を行うことが可能です。

詳細は、『Bluetooth Low Energy プロトコルスタック 基本パッケージ ユーザーズ・マニュアル』を参照してください。

Bluetooth Test Tool Suite についての詳細は、『Bluetooth Low Energy MCU Bluetooth Test Tool Suite 操作説明書』を参照してください。

シリアルポートの通信仕様は以下のようになります。

表 4-23 通信仕様

項目	設定
ボーレート	2000000 bps
データ長	8 bit
パリティ	なし
ストップビット	1 bit
フロー制御	無し

4.9.2. 構成図

本機能を動作させる場合の接続構成は以下の通りです。

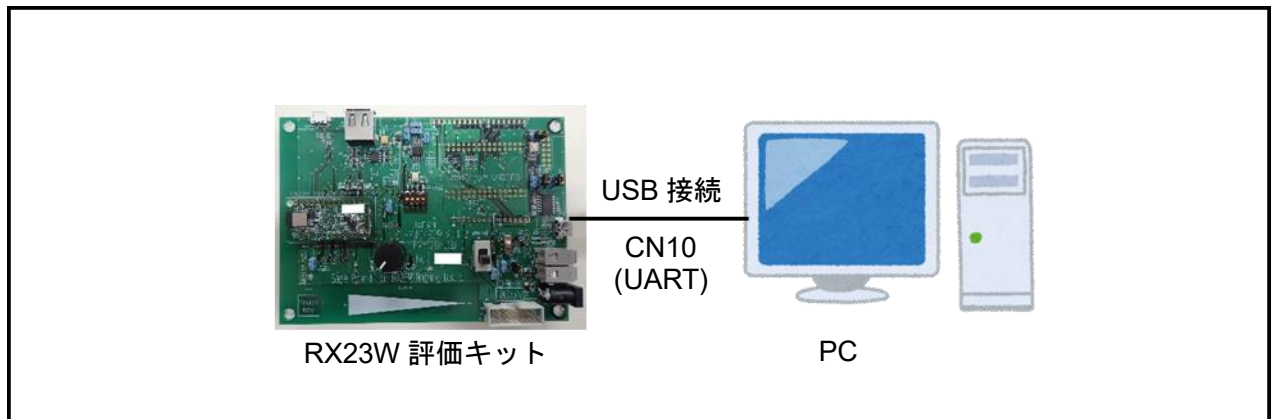


図 4-18 Bluetooth HCI 接続構成図

4.9.3. 使用端子

本機能で使用する端子は以下の通りです。

表 4-24 使用端子一覧

端子名	機能	説明
PC7/TXD8/SMOSI8	TXD8	SCI8 の送信データ出力端子
PC6/RXD8/SMISO8	RXD8	SCI8 の受信データ入力端子

4.9.4. HW の設定

RX23W 評価用ベースボード上に用意されているジャンパを次のように設定してください。

表 4-25 ジャンパ設定

端子名	設定
J6	Short
J14	Open
J15	Short
J16	Open
J17	Short

4.10. Bluetooth Server

ルネサスエレクトロニクス株式会社より提供されている『Bluetooth Low Energy プロトコルスタック』の基本パッケージを使用することで GATT サーバアプリケーションの動作確認が可能です。

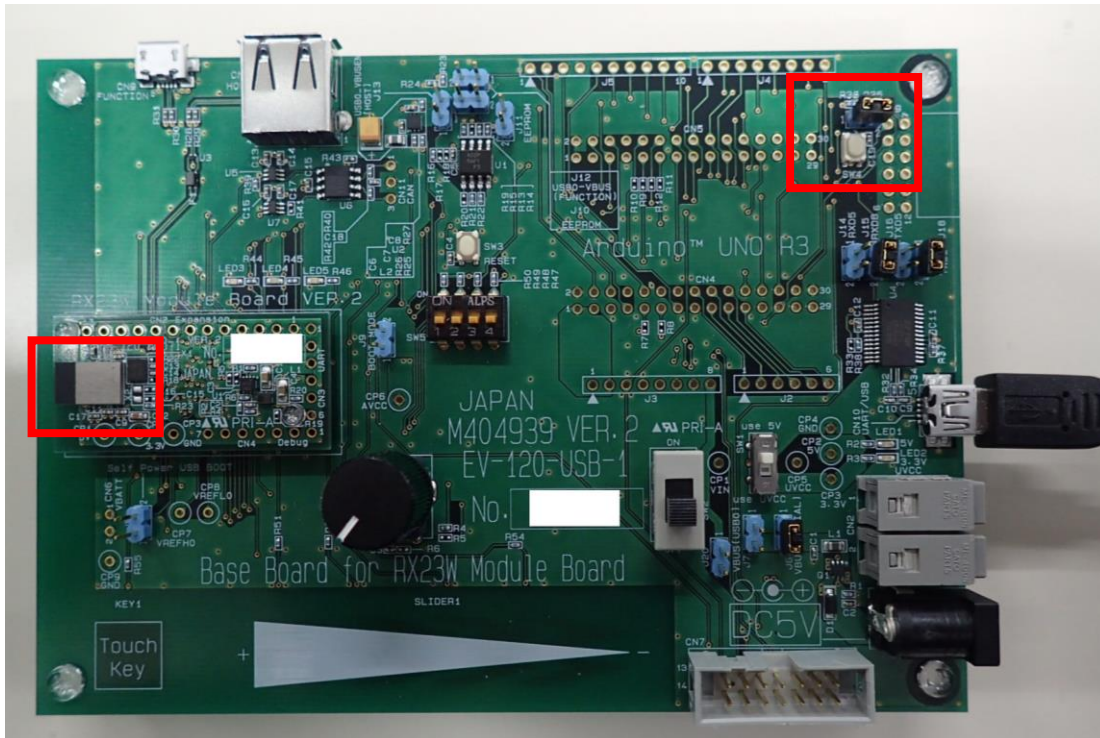


図 4-19 Bluetooth Server

表 4-26 サンプルプログラム

サンプルプログラム名	RX23W Group BLE Module Firmware Integration Technology Application Note r01an4860xx0100-rx23w-ble-fit.zip
プロジェクト名	ble_demo_rsskrx23w_profile_server
配布元	ルネサス エレクトロニクス株式会社

4.10.1. 機能説明

ルネサスエレクトロニクス株式会社より提供されている『Bluetooth Low Energy プロトコルスタック』と同等の動作を行うことが可能です。

詳細は、『Bluetooth Low Energy プロトコルスタック 基本パッケージ ユーザーズ・マニュアル』を参照してください。

シリアルポートの通信仕様は以下のようになります。

表 4-27 通信仕様

項目	設定
ボーレート	115200 bps
データ長	8 bit
パリティ	なし

RM-120-RFB-1	2021/11/01	SBAL-210166-00	47/55
アプリケーションノート			

ストップビット	1 bit
フロー制御	無し

4.10.2. 構成図

本機能を動作させる場合の接続構成は以下の通りです。

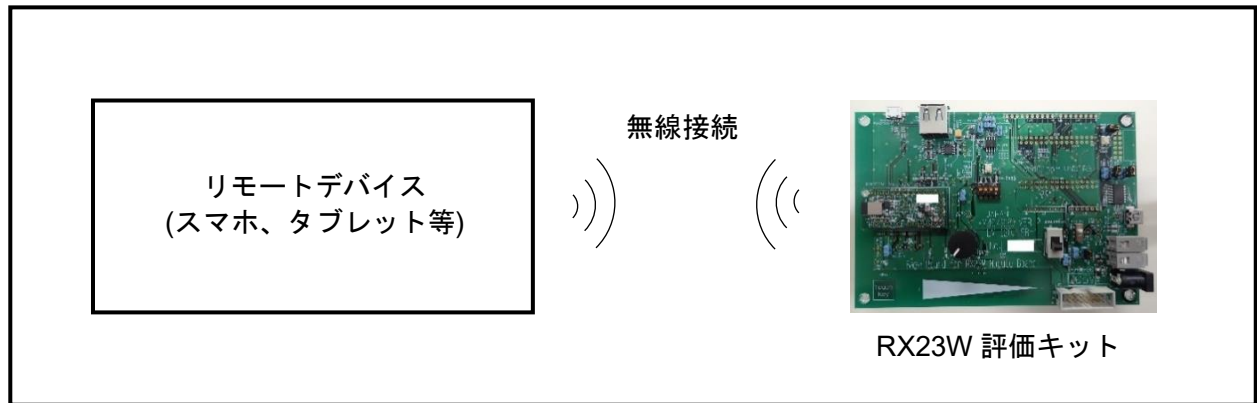


図 4-20 Bluetooth Server 接続構成図

4.10.3. 使用端子

本機能で使用する端子は以下の通りです。

表 4-28 使用端子一覧

端子名	機能	説明
P31/IRQ1	IRQ1	SW4 の入力を検出します
PE0/AN016	LED4	LED4 の駆動管理
P03/DA0	LED5	LED5 の駆動管理

4.10.4. HW の設定

RX23W 評価用ベースボード上に用意されているジャンパを次のように設定してください。

表 4-29 ジャンパ設定

端子名	設定
J19	Short

4.10.5. サンプルコード変更手順

本機能を使用するために、サンプルプログラムに対して以下の修正を行います。

No.	修正内容
1	<p>◆ポートの変更 ファイル名: r_ble_board.c 33~36 行目の記述を修正します。</p> <ul style="list-style-type: none"> 修正前 <pre>#define BLE_BOARD_SW1_IRQ (IRQ_NUM_1) #define BLE_BOARD_SW2_IRQ (IRQ_NUM_0) #define BLE_BOARD_LED1_PIN (GPIO_PORT_4_PIN_2) #define BLE_BOARD_LED2_PIN (GPIO_PORT_4_PIN_3)</pre> 修正後 <pre>#define BLE_BOARD_SW1_IRQ (IRQ_NUM_1) #define BLE_BOARD_LED1_PIN (GPIO_PORT_E_PIN_0) //red #define BLE_BOARD_LED2_PIN (GPIO_PORT_0_PIN_3) //green</pre> <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="border: 1px solid black; padding: 5px; text-align: center;">修正前</div> <div style="font-size: 2em;">→</div> <div style="border: 1px solid black; padding: 5px; text-align: center;">修正後</div> </div>
2	<p>◆対応するスイッチ番号の変更 ファイル名: app_main.c 463 行目の記述を修正します。</p> <ul style="list-style-type: none"> 修正前 <pre>R_BLE_BOARD_RegisterSwitchCb(BLE_BOARD_SW2, sw_cb);</pre> 修正後 <pre>R_BLE_BOARD_RegisterSwitchCb(BLE_BOARD_SW1, sw_cb);</pre> <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="border: 1px solid black; padding: 5px; text-align: center;">修正前</div> <div style="font-size: 2em;">→</div> <div style="border: 1px solid black; padding: 5px; text-align: center;">修正後</div> </div>

4.11. Bluetooth Client

ルネサスエレクトロニクス株式会社より提供されている『Bluetooth Low Energy プロトコルスタック』の基本パッケージを使用することで GATT クライアントアプリケーションの動作確認が可能です。

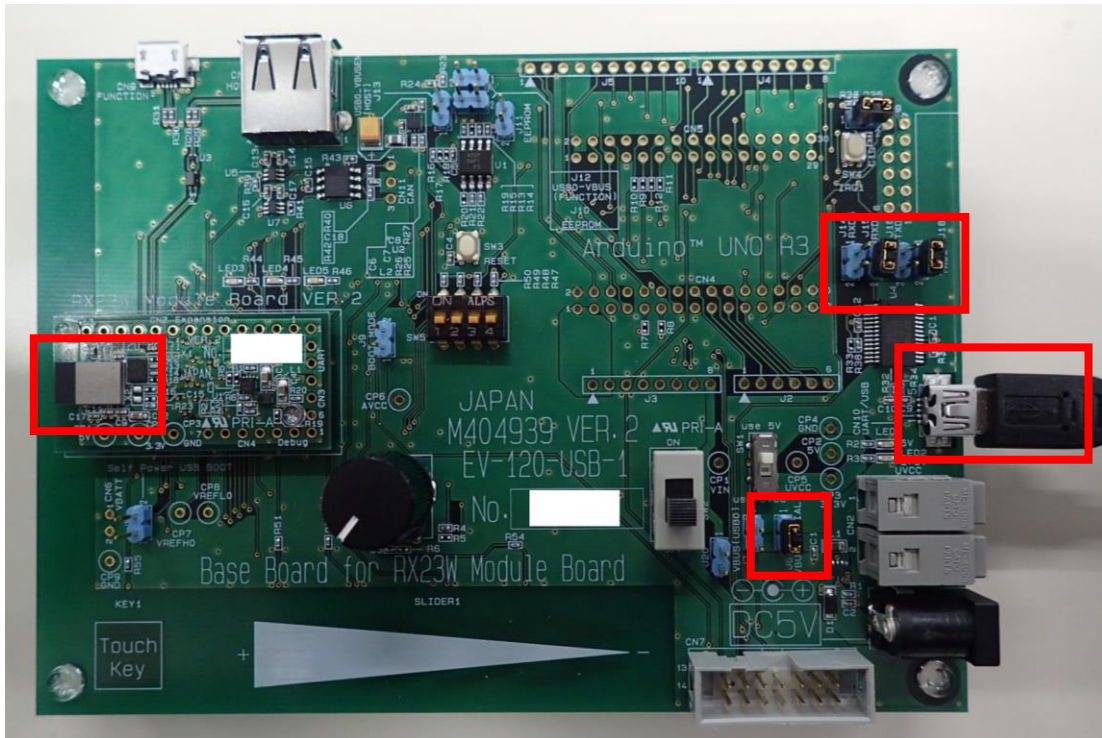


図 4-21 Bluetooth Client

表 4-30 サンプルプログラム

サンプルプログラム名	RX23W Group BLE Module Firmware Integration Technology Application Note r01an4860xx0100-rx23w-ble-fit.zip
プロジェクト名	ble_demo_rsskrx23w_profile_client
配布元	ルネサス エレクトロニクス株式会社

4.11.1. 機能説明

ルネサスエレクトロニクス株式会社より提供されている『Bluetooth Low Energy プロトコルスタック』と同等の動作を行うことが可能です。

詳細は、『Bluetooth Low Energy プロトコルスタック 基本パッケージ ユーザーズ・マニュアル』を参照してください。

シリアルポートの通信仕様は以下のようになります。

表 4-31 通信仕様

項目	設定
ボーレート	115200 bps
データ長	8 bit
パリティ	なし

ストップビット	1 bit
フロー制御	無し

4.11.2. 構成図

本機能を動作させる場合の接続構成は以下の通りです。

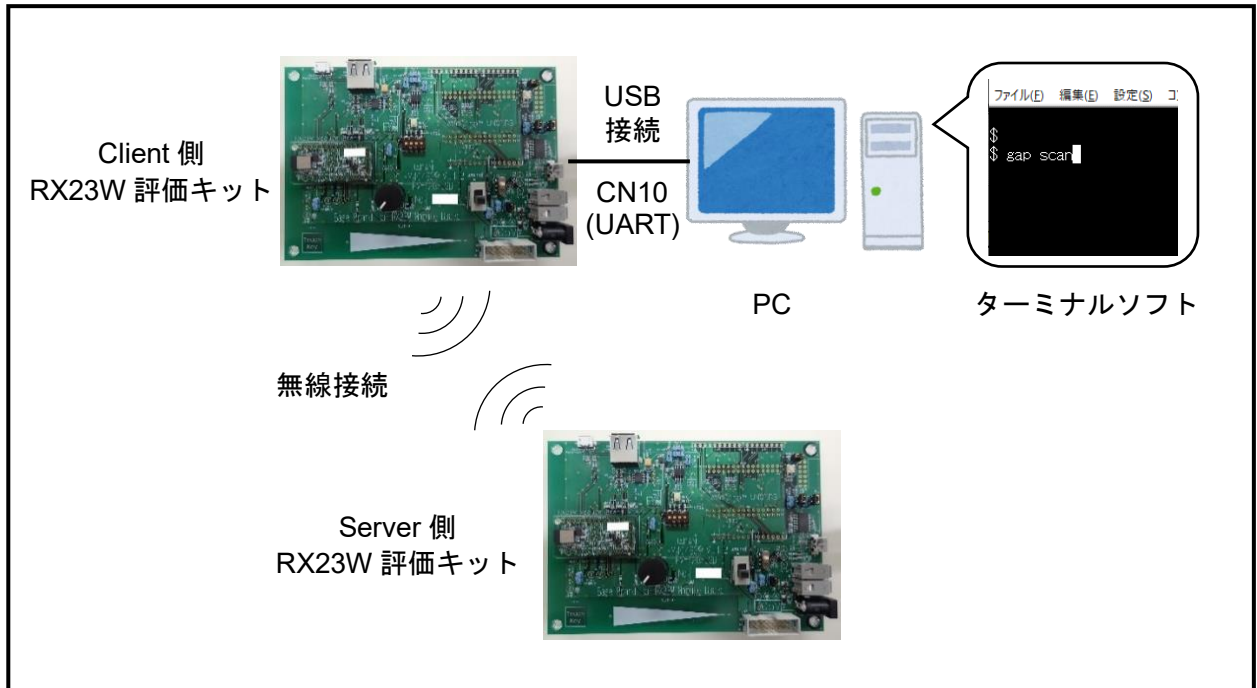


図 4-22 Bluetooth Client 接続構成図

4.11.3. 使用端子

本機能で使用する端子は以下の通りです。

表 4-32 使用端子一覧

端子名	機能	説明
PC7/TXD8/SMOSI8	TXD8	SCI8 の送信データ出力端子
PC6/RXD8/SMISO8	RXD8	SCI8 の受信データ入力端子

4.11.4. HW の設定

RX23W 評価用ベースボード上に用意されているジャンパを次のように設定してください。

表 4-33 ジャンパ設定

端子名	設定
J6	Short
J14	Open
J15	Short
J16	Open
J17	Short

4.12. Bluetooth Mesh

ルネサスエレクトロニクス株式会社より提供されている『RX23W グループ Bluetooth メッシュスタック スタートアップガイド』のメッシュスタックパッケージを使用することでメッシュアプリケーションの動作確認が可能です。

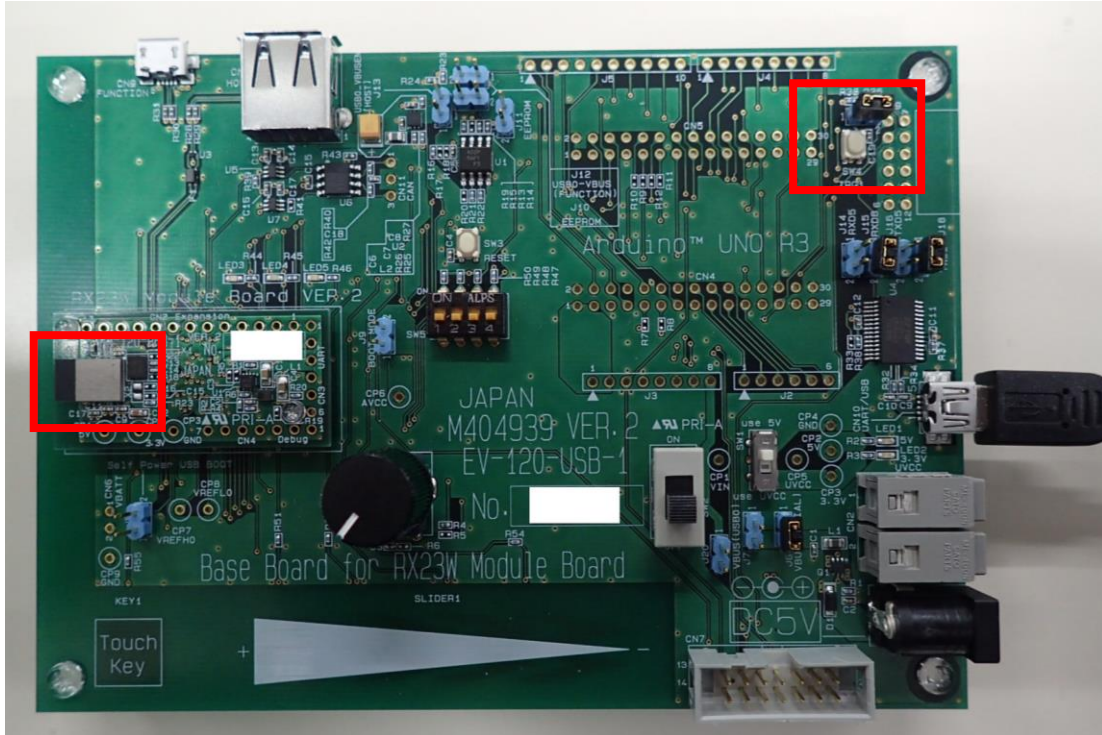


図 4-23 Bluetooth Mesh

表 4-34 サンプルプログラム

サンプルプログラム名	RX23W Group Bluetooth Mesh Module Firmware Integration Technology REN_r01an4930xx0110-rx23w-blemesh-fit_SCD_20200929.zip
プロジェクト名	rsskrx23w_mesh_server rsskrx23w_mesh_client
配布元	ルネサス エレクトロニクス株式会社

4.12.1. 機能説明

ルネサスエレクトロニクス株式会社より提供されている『Bluetooth メッシュスタック』と同等の動作を行うことが可能です。

詳細は、『RX23W グループ Bluetooth メッシュスタック スタートアップガイド』を参照してください。

シリアルポートの通信仕様は以下のようになります。

表 4-35 通信仕様

項目	設定
ボーレート	115200 bps
データ長	8 bit
パリティ	なし
ストップビット	1 bit
フロー制御	無し

4.12.2. 構成図

本機能を動作させる場合の接続構成は以下の通りです。

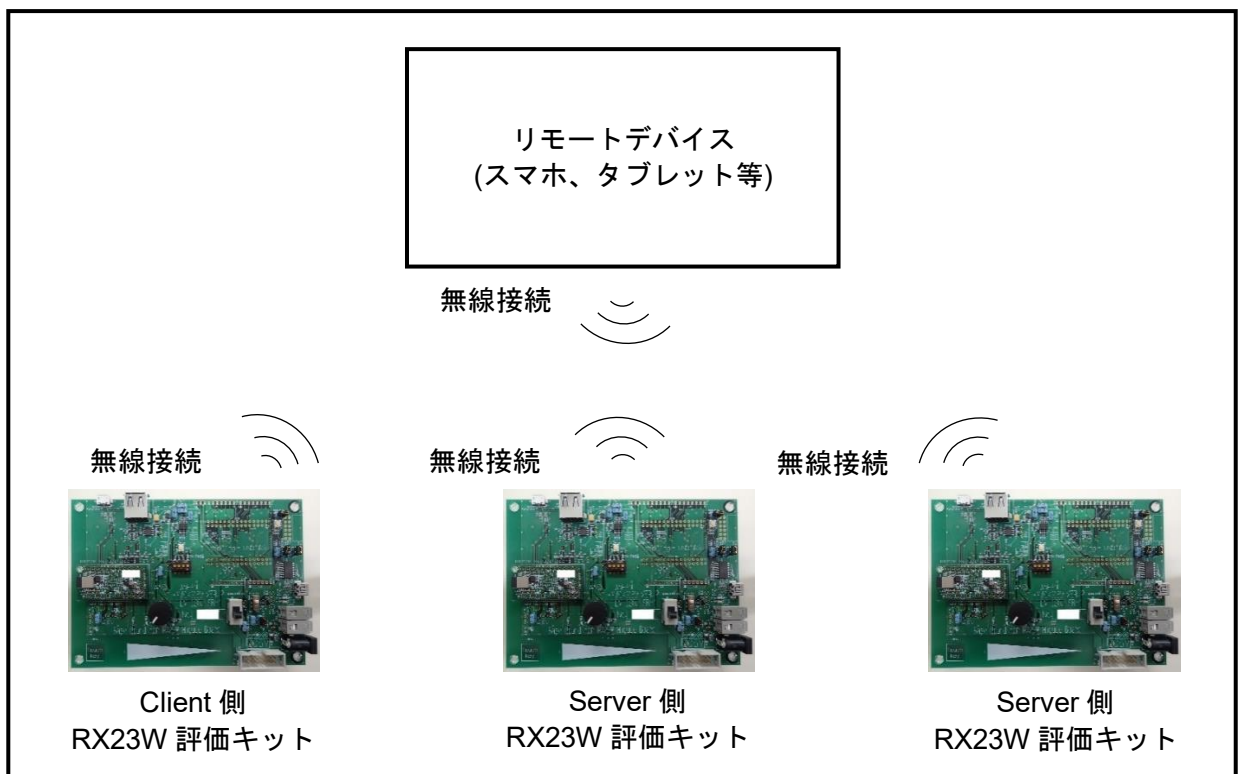


図 4-24 Bluetooth Mesh 接続構成図

4.12.3. 使用端子

本機能で使用する端子は以下の通りです。

表 4-36 サンプルプログラム

端子名	機能	説明
P31/IRQ1	IRQ1	SW4 の入力を検出します

4.12.4. HW の設定

RX23W 評価用ベースボード上に用意されているジャンパを次のように設定してください。

表 4-37 ジャンパ設定

端子名	設定
J19	Short

4.12.5. サンプルコード変更手順

本機能を使用するために、サンプルプログラムに対して以下の修正を行います。パッケージに同梱されている make_workspace_rsskrx23w.bat を開いてプロジェクトを作成、修正します。

No.	修正内容 (Server、Client 共通)
1	<p>◆ポートの変更 ファイル名: r_ble_board.c 33~36 行目の記述を修正します。</p> <ul style="list-style-type: none"> 修正前 <pre>#define BLE_BOARD_SW1_IRQ (IRQ_NUM_1) #define BLE_BOARD_SW2_IRQ (IRQ_NUM_0) #define BLE_BOARD_LED1_PIN (GPIO_PORT_4_PIN_2) #define BLE_BOARD_LED2_PIN (GPIO_PORT_4_PIN_3)</pre> 修正後 <pre>#define BLE_BOARD_SW1_IRQ (IRQ_NUM_1) #define BLE_BOARD_LED1_PIN (GPIO_PORT_0_PIN_3) //green #define BLE_BOARD_LED2_PIN (GPIO_PORT_E_PIN_0) //red</pre> <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="border: 1px solid black; padding: 5px; text-align: center;">修正前</div> <div style="font-size: 2em;">➡</div> <div style="border: 1px solid black; padding: 5px; text-align: center;">修正後</div> </div> <pre> 2 /* * DISCLAIMER[19 #include "r_ble_board.h" 20 21 #if (BLE_CFG_BOARD_LED_SW_EN == 1) 22 23 #include "r_gpio_rx_if.h" 24 #include "r_irq_rx_if.h" 25 26 27 #if (BLE_CFG_BOARD_TYPE == 1) /* for RX23W Target Board(TB) */ 28 #define BLE_BOARD_SW1_IRQ (IRQ_NUM_1) 29 #define BLE_BOARD_SW2_IRQ (IRQ_NUM_0) 30 #define BLE_BOARD_LED1_PIN (GPIO_PORT_C_PIN_0) 31 #define BLE_BOARD_LED2_PIN (GPIO_PORT_B_PIN_0) 32 #elif (BLE_CFG_BOARD_TYPE == 2) /* for RX23W RSSK board */ 33 #define BLE_BOARD_SW1_IRQ (IRQ_NUM_1) 34 #define BLE_BOARD_SW2_IRQ (IRQ_NUM_0) 35 #define BLE_BOARD_LED1_PIN (GPIO_PORT_4_PIN_2) 36 #define BLE_BOARD_LED2_PIN (GPIO_PORT_4_PIN_3) 37 #else /* BLE_CFG_BOARD_TYPE */ /* for Custom board */ 38 #define BLE_BOARD_SW1_IRQ (IRQ_NUM_7) 39 #define BLE_BOARD_SW2_IRQ (IRQ_NUM_5) 40 #define BLE_BOARD_LED1_PIN (GPIO_PORT_C_PIN_5) 41 #define BLE_BOARD_LED2_PIN (GPIO_PORT_C_PIN_6) 42 #endif /* BLE_CFG_BOARD_TYPE */ 43 44 static irq_handle_t gs_irq_hdl[BLE_BOARD_SW_MAX]; 45 static ble_sw_cb_t gs_sw_cb[BLE_BOARD_SW_MAX]; 46 47 #static void ble_sw_cb(void) 48 { 49 R_BLE_LPC_SetInhibitSoftwareStandby(true); 50 } </pre>

RM-120-RFB-1	2021/11/01	SBAL-210166-00	54/55
アプリケーションノート			

5. 参考情報

5.1. 出荷時ソフトウェアへの復元

本製品にユーザープログラムを書き込んだ後、出荷時ソフトウェアに復元する場合は、以下の手順を実行してください。

- ① 出荷時ソフトウェアは本製品に同梱されています。また、弊社 web サイトよりダウンロードすることができます。

<https://www.ndk-m.co.jp/software-download/>

- ② 「2.3 ファームウェア書き込み」に従って、「xxxxxxx.mot」を書き込んでください。

5.2. 新規開発プロジェクト作成時の注意事項

本製品に出荷時に書き込まれているソフトウェアは、不用意な書き換えを防ぐためにフラッシュメモリプロテクト機能を有効にしています。

本製品でコード開発のための新規開発プロジェクト作成する場合には、統合開発環境から本製品のオンボードエミュレータに接続する前に ID コード「45FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF」を設定する必要があります。

RM-120-RFB-1	2021/11/01	SBAL-210166-00	55/55
アプリケーションノート			

6. 改定履歴

版数	日付	内容
1 版	2021/11/01	新規作成

※記載の製品名、社名は各社の商標または登録商標です。