

# 静電容量タッチ低消費電力アプリケーション(SMS 使用)の開発

#### 要旨

本アプリケーションノートでは、RL78 MCU の SNOOZE モード・シーケンサ (SMS) を使用した静電容量 タッチ低消費電力アプリケーションの作成に必要な手順を説明します。

SMS を使用した自動判定計測を行うことで、低消費電力なタッチアプリケーションを実現できます。

動作確認デバイス

RL78/G22

RL78/G23



## 目次

1.	システム概要	3
2.	関連ドキュメント	3
3. 3.1 3.2	SNOOZE モード・シーケンサ (SMS) を使用した自動判定計測方法 SMS 計測時に使用するモジュールのフロー 外部端子の結線 (RL78/G22 のみ)	4 4 5
4.	使用する周辺機能	6
5.	動作確認環境	6
6. 6.1 6.2 6.3	ソフトウェア設定 オプション・バイト設定 静電容量タッチ設定 使用コンポーネント	7 7 9 10
7.	アプリケーション例の概要	10
8. 8.1 8.2 8.2. 8.2. 8.3 8.4 8.5 8.6	<ul> <li>静電容量タッチアプリケーション開発手順</li></ul>	11 12 14 14 14 21 26 37 40 44
9.	参考ドキュメント	51



#### 1. システム概要

RL78 ファミリは SNOOZE モード・シーケンサ (SMS) を使用して静電容量タッチ機能 (CTSU2La) の自動 判定計測動作(SMS を使用した自動判定計測) を実装可能です。本アプリケーションノートでは、SMS を使用 した自動判定計測の作成手順について主に以下の項目に分けて説明します。

- RL78 ファミリ CPU ボードを使用したスマート・コンフィグレータによるプロジェクト作成
- QE for Capacitive Touch によるタッチインタフェース作成とチューニング

#### 2. 関連ドキュメント

本アプリケーションノートでは、実際に動作するアプリケーションを作成する手順を簡単に紹介します。 このアプリケーション例で使用されている各ツールに関する質問、より詳細な使用方法に関しては、  $e^2$  studio /スマート・コンフィグレータ、Software Integration System (SIS) のドライバ/ミドルウェア、 Renesas Code Generator や QE for Capacitive Touch のヘルプ ( $e^2$  studio のヘルプに含まれています) など のドキュメントを参照してください。



## RL78 ファミリ 静電容量タッチ低消費電力アプリケーション(SMS 使用)の開発

- 3. SNOOZE モード・シーケンサ (SMS) を使用した自動判定計測方法
- 3.1 SMS 計測時に使用するモジュールのフロー

RL78/G22 と RL78/G23 で、SMS 計測時に使用するモジュールのフローが異なります。

• RL78/G22 の場合

CTSU2La から DTC を使用してポート出力を行います<sup>注</sup>。ポートから出力された信号を用いて外部割り 込み信号を発生させます。割り込み信号によってELCをトリガさせ SMS 処理を開始します。

【注】 DTC 転送でポート・レジスタ (Pxx) を 8 ビット単位で書き換えます。 したがって SMS を使用した自動判定計測処理の実行中は、DTC の転送先となるポート・レジスタ

(Pxx) を他の機能で使用することはできません。他システムと競合しないように、使用するポート・ レジスタ (Pxx) を選択してください。



図 3-1 SMS 計測時に使用するモジュールのフロー(RL78/G22)

• RL78/G23の場合

CTSU2L から ELCL をトリガさせ SMS 処理を開始します。



図 3-2 SMS 計測時に使用するモジュールのフロー(RL78/G23)



3.2 外部端子の結線 (RL78/G22のみ)

RL78/G22 で SMS を使用した自動判定計測を行う場合は、上記フロー内で使用するポート端子 (Pxx) と 外部割り込み端子 (INTPxx) を結線してください。なお RL78/G23 では本処理は不要です。

RL78/G22 静電容量タッチ評価システム(製品型名: RTK0EG0042S01001BJ)を使用した場合で説明します。

CPU ボード側のCN2の32番ピン (P22/TS20) と16番ピン (P16/INTP5/TS17) を以下のように結線します。

P22 : CTSU2La から DTC を使用してポート出力するための端子です。

INTP5 : ポート端子から出力された信号を用いて割り込み信号を発生させるための端子です。



図 3-3 RL78/G22 で SMS 使用時のハードウェア結線パターン



#### 4. 使用する周辺機能

表 4-1,表 4-2 にサンプルコードで使用する周辺機能を示します。

表 4-1 周辺機能と用途 (RL78/G22)

周辺機能	用途
静電容量センサユニット (CTSU2La)	タッチ電極に発生する静電容量を計測
32 ビット・インターバル・タイマ (TML32)	STOP モードを解除し、SNOOZE モードへ遷移
	するためのタイマ (計測周期 : 20 ms)
データ・トランスファ・コントローラ (DTC)	DTC を使用してポート出力を行う。
ポート	ポートから出力された信号を用いて割り込み信号
割り込みコントローラ (INTP)	を発生させる。
イベント・リンク・コントローラ (ELC)	割り込み信号によって ELC をトリガさせる。
SNOOZE モード・シーケンサ (SMS)	そして、SMS 処理を開始する。

表 4-2 周辺機能と用途 (RL78/G23)

周辺機能	用途
静電容量センサユニット (CTSU2L)	タッチ電極に発生する静電容量を計測
32 ビット・インターバル・タイマ (TML32)	STOP モードを解除し、SNOOZE モードへ遷移
	するためのタイマ (計測周期 : 20 ms)
ロジック&イベント・リンク・コントローラ (ELCL)	計測データ転送要求 (INTCTSURD) で ELCL を
SNOOZE モード・シーケンサ (SMS)	トリガし、SMS 処理を開始する。

#### 5. 動作確認環境

表 5-1 にプロジェクトで使用する周辺環境を示します。

表 5-1 動作確認環境

項目	内容
使用マイコン	RL78/G22 (R7F102GGE2DFB)
	RL78/G23 (R7F100GSN2DFB)
動作電圧	3.3V
	LVD0 検出電圧:
	立ち上がり時 TYP. 2.67V (2.59 V~2.75 V)
	立ち下がり時 TYP. 2.62V (2.54 V~2.70 V)
動作周波数	高速オンチップ・オシレータ・クロック (fн) : 32 MHz
	低速オンチップ・オシレータ・クロック (f⊾) : 32.768 kHz
ターゲットボード	RL78/G22 静電容量タッチ評価システム
	(製品型名:RTK0EG0042S01001BJ)
	RL78/G23 静電容量タッチ評価システム
	(製品型名:RTK0EG0030S01001BJ)
統合開発環境	e <sup>2</sup> studio (2024-04)
スマート・コンフィグレータ	V24.4.0
Cコンパイラ	CC-RL V1.13.00
	最適化レベルのオプション: -Odefault
静電容量式タッチセンサ対応	QE for Capacitive Touch V3.4.0
開発支援ツール	
エミュレータ	E2 エミュレータ Lite (RTE0T0002LKCE00000R)



## 6. ソフトウェア設定

6.1 オプション・バイト設定

表 6-1,表 6-2 にオプション・バイトの設定を示します。

#### 表 6-1 オプション・バイト設定 (RL78/G22)

アドレス	設定値	内容
000C0H / 020C0H	1110 1111B (0xEF)	ウォッチドッグ・タイマのカウンタの動作停止
		(リセット解除後、カウント停止)
000C1H / 020C1H	1111 1100B (0xFC)	LVD0 検出電圧 : リセット・モード
		立ち上がり時:2.67V (TYP) (2.59 V~2.75 V)
		立ち下がり時:2.62V (TYP) (2.54 V~2.70 V)
000C2H / 020C2H	1110 1000B (0xE8)	HS (高速メイン) モード、
		高速オンチップ・オシレータの周波数:32MHz
000C3H / 020C3H	1000 0100B (0x84)	オンチップ・デバッグ動作許可

#### 表 6-2 オプション・バイト設定 (RL78/G23)

アドレス	設定値	内容
000C0H / 040C0H	1110 1111B (0xEF)	ウォッチドッグ・タイマのカウンタの動作停止
		(リセット解除後、カウント停止)
000C1H / 040C1H	1111 1100B (0xFC)	LVD0 検出電圧 : リセット・モード
		立ち上がり時:2.67V (TYP) (2.59 V~2.75 V)
		立ち下がり時:2.62V (TYP) (2.54 V~2.70 V)
000C2H / 040C2H	1110 1000B (0xE8)	HS (高速メイン)モード、
		高速オンチップ・オシレータの周波数:32MHz
000C3H / 040C3H	1000 0100B (0x84)	オンチップ・デバッグ動作許可



オプション・バイトの設定はコード生成後に、プロジェクトのプロパティから、 [C/C++ビルド] - [設定] - [ツール設定] - [Linker] - [デバイス]を開き、「ユーザー・オプション・バイト値」 及び 「オンチップ・デバッグ制御値」にて確認することができます。



図 6-1 オプション・バイト設定



## 6.2 静電容量タッチ設定

本アプリケーション例での静電容量タッチ設定を示します。

RL78/G22 の場合は、静電容量タッチセンサ(CTSU2La)の複数電極接続 (MEC) 機能を SMS と併用することで、SMS のみでのタッチ計測動作に比べてさらに低消費電力で動作が可能です。

※RL78/G23の静電容量タッチセンサは CTSU2L のため、MEC 機能は非搭載です。



図 6-2 本アプリケーション例での静電容量タッチ設定 (RL78/G22)



図 6-3 本アプリケーション例での静電容量タッチ設定 (RL78/G23)



6.3 使用コンポーネント

図 6-3, 図 6-4 にスマート・コンフィグレータで設定したコンポーネント及びモジュールを示します。

コンポーネント ^	バージョン	設定
Board Support Packages v1.62 (r_bsp)	1.62	r_bsp(使用中)
Capacitive Sensing Unit driver. (r_ctsu)	1.50	r_ctsu(使用中)
Touch middleware. (rm_touch)	1.50	rm_touch(使用中)
오 インターバル・タイマ	1.4.0	Config_ITL000(ITL000: 使用中)
⊘ ポート	1.4.1	Config_PORT(PORT: 使用中)
⊘割り込みコントローラ	1.4.0	Config_INTC(INTC: 使用中)
📀 電圧検出回路	1.3.0	Config_LVD0(LVD0: 使用中)

図 6-4 スマート・コンフィグレータで設定したコンポーネント及びモジュール (RL78/G22)

コンポーネント ^	バージョン	設定
😎 Board Support Packages v1.62 (r_bsp)	1.62	r_bsp(使用中)
Capacitive Sensing Unit driver. (r_ctsu)	1.50	r_ctsu(使用中)
Touch middleware. (rm_touch)	1.50	rm_touch(使用中)
오 インターバル・タイマ	1.4.0	Config_ITL000(ITL000: 使用中)
오 電圧検出回路	1.3.0	Config_LVD0(LVD0: 使用中)

図 6-5 スマート・コンフィグレータで設定したコンポーネント及びモジュール (RL78/G23)

#### 7. アプリケーション例の概要

アプリケーション例のメインループの実装は以下のとおりです。

- イニシャルオフセットチューニング後、RM\_TOUCH\_SmsSet 関数を実行することで SMS を設定 します。
- ② RM\_TOUCH\_ScanStart 関数の実行で、タッチ計測設定および SNOOZE 機能を有効にする設定を 行い、外部トリガ待ち状態にする。
- ③ TML32 のタイマカウントを開始します。 (計測周期: 20ms)
- ④ STOP 命令の実行で STOP モードへ遷移します。
- ⑤ TML32の割り込み要求が発生することで、ELC (G23の場合は ELCL) からの外部トリガにより、 SNOOZE モードへ遷移してタッチ計測を開始します。
- ⑥ 計測終了割り込みが発生すると SNOOZE モードのまま SMS によりタッチオン/オフ判定を 行います。
- ⑦ タッチオン判定時に通常モードに遷移します。タッチオフ判定時は④に戻ります。

- ⑧ RM\_TOUCH\_ScanStart 関数を実行して外部トリガ待ち状態にします。
- ⑨ TML32のタイマカウントを開始します。(計測周期: 20 ms)
- TML32 のタイマカウントが 20ms 経過で、ELC (G23 の場合は ELCL) からの外部トリガにより、 タッチ計測を行います。
- ① タッチ計測終了後、タッチオン/オフ判定を実行します。なお、本アプリケーション例では判定結果
   に関わらず①に遷移します。

【注】RL78/G23 は SMS を使用したタッチ計測 (低消費モード) になります。



## 8. 静電容量タッチアプリケーション開発手順

プロジェクトにタッチセンサ検出を統合するために必要な開発手順を以下に示します。これらの手順は 一般的なユーザアプリケーション開発に適用可能です。

- e<sup>2</sup> studio のプロジェクト作成ウィザードを使用して新規プロジェクトを作成します。
- スマート・コンフィグレータを使用して、必要なモジュールを作成したプロジェクトに追加します。
- QE for Capacitive Touch を使用して、静電容量タッチインタフェースを作成します。
- QE for Capacitive Touch を使用して、プロジェクトをチューニングします。
- 必要な SIS のモジュールの API コールをプロジェクトに追加し、静電容量タッチ制御を有効にします。

次項から各手順について説明します。

なお特に説明のない限り、RL78/G22 での設定手順を記載しています。



- 8.1 プロジェクト作成
   e<sup>2</sup> studio のプロジェクト作成ウィザードを使用して新規プロジェクトを作成します。
   以下に手順を示します。
- 1. Windows のスタートメニューまたはデスクトップのショートカットから e<sup>2</sup> studio を起動します。 ダイアログが表示されたら、任意の場所にワークスペースを作成します。
- 2. e<sup>2</sup> studio のメニュー[ファイル] [新規] [C/C++ Project]を選択し、新規プロジェクトの作成を開始します。
- 3. ダイアログが表示されたら"Renesas RL78" "Renesas CC-RL C/C++ Executable Project"を選択し、 [**次へ**]をクリックします。
- 次のダイアログで"プロジェクト名(P):"に任意のプロジェクト名を入力します。
   このアプリケーション例では"Capacitive\_Touch\_Project\_Example"を入力します。
   入力完了後、[次へ]をクリックします。
- 5. 次のダイアログでは、以下を選択します。
- 言語: C
- ツールチェーン: Renesas CCRL
- ツールチェーン・バージョン: v1.13.00
- Target Board: Custom
- ターゲット・デバイス: RL78/G22 (R7F102GGExFB)
  - RL78/G23 (R7F100GSNxFB)
- "Hardware Debug 構成を生成"をチェック。E2 Lite (RL78)をプルダウンメニューから選択。

•	– D X
New Renesas CC-RL Executable Project Select toolchain, device & debug settings	ightarrow
Toolchain Settings 言語: <ul> <li>C ○ C++</li> <li>ツールチェーン: Renesas CC-RL </li> <li>ツールチェーン・パージョン: v1.13.00 </li> <li>ツールチェーンの管理</li> </ul>	
Device Settings Target Board: Custom Download additional boards ターゲット・デパイス: R7F102GGExFB ゴバイスのアンワ エンディアン: Little プロジェクト・タイプ: デフォルト	Configurations <ul> <li>Hardware Debug 構成を生成</li> <li>E2 Lite (RL78) </li> </ul> <ul> <li>Debug 構成を生成</li> <li>RL78 Simulator </li> <li>Release 構成を生成</li> </ul>
? < 戻る(B) 次へ(N) >	終了(F) <b>キャンセル</b>

図 8-1 ツールチェーン, デバイス, デバッグ設定の選択



"ターゲット・デバイス"は、[…]ボタンを押下して"Device Selection"ウィンドウに表示されるデバイス から選択します。

							×
Device Selection							
You can filter devices by	regular expr	ression					
	regular expr	coston					
Search Device							
Device	RAM	ROM	Pin	RTOS	Smart Co	周辺コード	^
✓ RL78 - G22							
> RL78 - G22 16pin							
> RL78 - G22 20pin							
> RL78 - G22 24pin							
> RL78 - G22 25pin							
> RL78 - G22 30pin							
> RL78 - G22 32pin							
> RL78 - G22 36pin							
> RL78 - G22 40pin							
> RL78 - G22 44pin							
✓ RL78 - G22 48pin							
R7F102GGCxFB	4 KB	32 KB	48		✓	×	
R7F102GGCxNP	4 KB	32 KB	48		✓	×	λ
R7F102GGExFB	4 KB	64 KB	48		×	× <	
R7F102GGExNP	4 KB	64 KB	48		✓	×	Ν
✓ RL78 - G23							
> RL78 - G23 30pin							
> RL78 - G23 32pin							
> RL78 - G23 36pin							
> RL78 - G23 40pin							
> RL78 - G23 44pin							
> RL78 - G23 48pin							
> RL78 - G23 52pin							
> RL78 - G23 64pin							
> RL78 - G23 80pin							
> RL78 - G23 100pin							
✓ RL78 - G23 128pin							
R7F100GSJxFB	24 KB	256 KB	128		✓	×	
R7F100GSKxFB	32 KB	384 KB	128		✓	×	
R7F100GSLxFB	48 KB	512 KB	128		✓	×	
R7F100GSNxFB	48 KB	768 KB	128		✓	× 🔸	~
?					ОК	キャンセル	

図 8-2 ターゲット・デバイスの選択

- 【備考】RL78/G23 RSSK (RTK0EG0030S01001BJ) で動作確認する場合は、ターゲット・デバイスに 「R7F100GSNxFB」を選択してください。
- 6. 完了したら、[**OK**]をクリックします。
- 7. 次のダイアログが表示されたら、"Use Smart Configurator"をチェックし、[終了]をクリックしてください。

完了すると e<sup>2</sup> studio のデフォルトウィンドウにスマート・コンフィグレータのパースペクティブが表示 され、プロジェクトの設定が可能な状態になります。これで新規プロジェクトの作成は完了です。

RL78 ファミリ	静電容量タッチ低消費電力アプリケーション(SMS 使用)の開発

8.2 スマート・コンフィグレータによるモジュール作成

8.2.1 スマート・コンフィグレータによるコンポーネント追加

スマート・コンフィグレータを使用して、必要なモジュールのソースファイルをプロジェクトに追加します。 以下に手順を示します。

RL78/G22 と RL78/G23 では一部設定内容が異なる箇所があります。

 e<sup>2</sup> studio の中央下部のタブから[クロック]タブを選択し、RL78 MCU のクロック設定を下記のように行い ます。本アプリケーション例では低速周辺クロック (fSXP) に低速オンチップ・オシレータを使用します。 下図のとおり、XT1 発振回路のチェックを外し、fSXP に"低速オンチップ・オシレータ"を選択します。

しり設定						<b>じ</b> コードの生成 レポート
	B					
				~.		
助作モード:	高速メイン・モード2.7	7(V)~5.5(V)	•			
/ 高油オンチョーゴ・オン	1P		7			
- 周辺のシテラクラ ろン 別波数:	32	▼ (MHz)				
1000間始發生						fiHP
STOPE-EMEDUU-	-7時表よびSNOOZEE	Fードへの移行時に高速オッチップ	Ĩ	N		32 (MHz)
能振器を起動するための	の設定があります。)	C 1 - 10/15/13/13/16/140205/27/22			í	fMAIN
						GLK (MHz)
						32000 (kHz)
_						fimp 🕕
中速オンチップ・オシ	レータ		•			- (MHz)
波数:		∞ (MHz)				
			分周器			
X1% 惯回路						fMXP 🕕
前作モード:		Ŧ				- (MHz)
]波数:	5	(MHz)				
振安定時間:		▼ 52428.8(µs)				
	2.0					fiL attac
転用オンチップ・オシレー! 13世#6	5					32.708 (KHz)
1次900	32.708 FKw月·D/口折動作 #	(kHz) たけfSXDが低いゆオンチャヴ・オミン			ור	
一次を選択	1.22 STRUGULE 6	10101010111110120327332 322				32.768 (kHz)
XT1発振回路					-	ISXR 🕕
が作モード:				ר א <sup>י</sup> איז	7	- (KF12)
波数:		(kHz)				
F1発振						
総モート						
	and a share					

図 8-3 クロック設定



 本アプリケーション例では、MCU を高速メイン・モード (HS モード) かつ動作電圧範囲 2.7V~5.5V で 使用するため、下図のように"動作モード"を選択します。 RL78/G23 の場合は EVDD 設定も設定する必要があります。

動作モード:	高速メイン・モード2.7(V)~5.5(V)	
	図 8-4 動作モード設定 (RL78/G22)	

動作モード:	高速メイン・モード2.7(V)~5.5(V)	-
EVDD設定:	2.7 V ≤ EVDD0 ≤ 5.5 V	Ŧ

図 8-5 動作モードおよび EVDD 設定 (RL78/G23)

3. [**システム**]タブに移動します。

以下のように"**エミュレータを使う**"および"**E2 Lite**"を選択し、"**セキュリティ ID を設定する**"のチェック を外します。

シ	ステム設定		
	▼ オンチップ・デバッグ設定		
	オンチップ・デバッグ動作設定		
	○ 使用しない	<ul> <li>Iミュレータを使う</li> </ul>	⊖ сомポ−ト
	エミュレータ設定		
	○ E2	• E2 Lite	
	疑似RRM/DMM機能設定	•	
	○ 使用しない	<ul> <li>使用する</li> </ul>	
	Start/Stop関数機能設定		
	● 使用しない	○ 使用する	
	通過ポイント機能設定		
	◎ 使用しない	○使用する	
	セキュリティID設定		
	マキュリティIDを設定する		
	セキュリティID	0×000000000000000000000000000000000000	
	セキュリティID認証失敗時の設定		
	○ フラッシュ・メモリ – タを消去しない		
	<ul> <li>リフツンユ・メモマタを消去する</li> </ul>		
概	要 ボード クロック システム コンポーネント 端子 割り込み	<i>b</i>	

図 8-6 オンチップ・デバック設定

[コンポーネント]タブを選択し、r\_bspのバージョンを確認します。
 r\_bspを選択して右クリックし、"バージョンの変更"をクリックします。

ソフトウェアコンポー	ーネント設定	
コンポーネント	ڬ 🗳 🖧 🕞 🕀	設定
<ul> <li>マイルタ入力</li> <li>マンクレートアップ</li> <li>マシンクレートアップ</li> <li>マシンクレートアップ</li> <li>マシンクレートアップ</li> <li>マシンクレートアップ</li> <li>マシンクレートアップ</li> <li>マンクレートアップ</li> <li>マンクレ</li></ul>	<ul> <li>▲ ■</li> <li>パージョンの変更</li> <li>▲</li> <li>単除</li> <li>リセット時のデフォルト</li> </ul>	プロパティ ◆ ② Configur # Start u # Contr # Protect t
	サンブルブロジェクトのダウンロード	とインボート 自li # API fu # API fu # API fu # API fu # API fu # API fu # Param # Enable
慨要   ホート   クロック   シ	ステム コンホーイント 端子 割り込み	

図 8-7 r\_bsp のバージョン変更

"現在のバージョン"が 1.62 以降と異なる場合は "変更後のバージョン"で 1.62 以降を選択して[**次へ**]を クリックでバージョンを変更してください。

😨 バージョンの変更			-		×
パーションの選択 変更後のバージョン	を選択してください。				
コンポーネント名: 現在のパージョン: 変更後のパージョン:	1.62 1.61				~
?	< 戻る(B)	次へ(N) >	終了(F)	キャンセル	,

図 8-8 r\_bsp のバージョン選択



5. SIS モジュールと各種コンポーネントをプロジェクトに追加します。 コンポーネントの追加ボタン (赤枠の箇所) をクリックし、一覧から追加するコンポーネントを選択します。

ソフトウェアコンポーネン	卜設定	
コンポーネント	è d 🖡 E 🕀	設定
<ul> <li>マイルタ入力</li> <li>マートアップ</li> <li>マ (Del y 29 - トアップ)</li> <li>マ (Del y 29 - トアップ)</li> <li>マ (Del y 29 - トアップ)</li> <li>マ (Del y 20 - トアップ)</li> <li>マ (Del y 2</li></ul>		
概要 ポード クロック システム	コンポーネント 端子 割り	<u>込み</u>

図 8-9 コンポーネントの追加



6. "コンポーネントの追加"ダイアログが表示され、プロジェクトに追加可能なモジュールが表示されます。

😢 コンポーネントの追加					
ソフトウェアコンポーネントの選択					
使用可能なコンポーネントの一覧から選択	えしてください				
カテゴリ 全7					~
機能 今7					~
7411.9					=
コンポーネント		Short Name	タイプ	バージョン	^
冊 A/Dコンバータ			コード生成	1.5.0	
Board Support Packages v1.62		r_bsp	RL78 Software In	1.62	
H Capacitive Sensing Unit driver.		r_ctsu	RL78 Software In	1.50	
Hash Driver[Renesas Flash Drive	r RL78 Ty	r_rfd_rl78_t01_codefla	Generic SW	1.10	~
☑ 最新バージョンのみ表示					
説明					
アナログ-デジタル(A/D) 赤摘回路け ア	+ログ入 カを=	デジタル信号に変換する機能	77.7		~
//ロ//////////////////////////////////	10////2/				
					$\sim$
RL78 Software Integration System ES	<u> バュールをダウン</u>	<u>/ロードする</u>			
基本設定					
?	< 戻る(	B) 次へ(N) >	終了(F)	キャンセル	·

- 図 8-10 "コンポーネントの追加"ダイアログ
- 【注意】 SIS (Software Integration System) モジュールを初めて使用する場合は、以下の①~④の順に選 択してダウンロードしてください。



図 8-11 SIS モジュールのダウンロードリンク



図 8-12 リージョン選択

e							
<b>RL7</b> 8 ダウ	3 Software Integration Systemモジュ- ンロードするRL78 Software Integration System	- <b>ルのダウンロード</b> モジュールを選択してく	ください。			Ľ	1
	タイトル	ドキュメントNo.	リビジョン	発行日	^	すべて	選択
$\checkmark$	RL78ファミリ CTSUモジュール Software Integ	R11AN0484JJ0	Rev.1.50	2024-05-31			
	RL78ファミリ TOUCHモジュール Software Int	R11AN0485JJ0	Rev.1.50	2024-05-31		1	3件际
	RL78ファミリ Renesas Flash Driver RL78 Ty	R20AN0653JJ0	Rev.1.20	2024-05-20			
	RL78ファミリ Renesas Flash Driver RL78 Ty	R20AN0654JJ0	Rev.1.20	2024-05-20			
	RL78ファミリ Renesas Flash Driver RL78 Ty	R20AN0655JJ0	Rev.1.20	2024-05-20			
	RL78ファミリ Renesas Flash Driver RL78 Ty	R20AN0656JJ0	Rev.1.20	2024-05-20	1		
$\checkmark$	RL78ファミリ ボードサポートパッケージモジュール	R01AN5522JJ0	Rev.1.62	2023-11-30	K	(	3
モジ	1ール・フォルダー・パス: C:¥Users¥ ¥.eclipse¥com.renesas.p	olatform_download	d¥RL78_Mo	dules¥Gei	lodules 4	き キャンセ	▶照 !ル

図 8-13 SIS モジュールのダウンロード



7. スマート・コンフィグレータで以下のコンポーネントを選択してください。

【注】RL78/G23 の場合、SMS を使用した自動判定計測の動作に「ポート」「割り込みコントローラ」の 設定は不要です。

					$\times$
フトウェアコンポーネントの選択					
使用可能なコンポーネントの一覧から選択	してください			t	Þ
1					
カテコリ 全て					~
機能 全て					$\sim$
14119					
^ コンポーネント	Sh	ort Name	タイプ	バージョン	
₩ A/Dコンバータ			コード生成	1.5.0	
🖶 Board Support Packages v1.62	r_b	sp	RL78 Software In	1.62	
Capacitive Sensing Unit driver.	r_c	tsu	RL78 Software In	1.50	
🖥 Flash Driver[Renesas Flash Driver	<sup>.</sup> RL78 Ty r_r	fd_rl78_t01_codefla	Generic SW	1.10	
🖥 Flash Driver[Renesas Flash Driver	RL78 Ty r_r	fd_rl78_t01_dataflash	Generic SW	1.10	
🖶 Flash Driver[Renesas Flash Driver	RL78 Ty r_r	fd_rl78_t01_extraarea	Generic SW	1.10	
Flash Driver[Renesas Flash Driver]	RL78 Ty r_r	fd_rl78_t01_common	Generic SW	1.10	
FS1015 Sensor Middleware	r_f:	s1015	RL78 Software In	1.01	
FS2012 Sensor Middleware	r_f	s2012	RL78 Software In	1.12	
FS3000 Sensor Middleware	r_f:	s3000	RL78 Software In	1.01	
HS300x Sensor Middleware	r_h	s300x	RL78 Software In	1.22	
HS400x Sensor Middleware	ch 	IS400x	RL78 Software In	1.02	
IIC Communication Driver Interfa	ice Midd r_c	omms_12c	KL/8 Software In	1.11	
■ IIC通信 (スレーノ・モート)			コート生成	1.4.1	
■ IIC通信 (マスダ・モート)		L1202	J-F生成	1.5.1	
BOB 1203 Sensor Middleware	r_c	01203	RL/8 Software In	1.02	
			リート生成	1.0.0	
			クラノ1 川ル・コノノ1…	1.3.1	
	rm	touch	」「主成 RI 78 Software In	1.4.1	
UART通信			コード生成	1.6.0	•
ZMOD4XXX Sensor Middleware	rz	mod4xxx	RL78 Software In	1.22	
イベントリンクコントローラ			コード生成	1.2.0	
インターバル・タイマ			コード生成	1.4.0	
ウォッチドッグ・タイマ			コード生成	1.4.0	
┣ キー割り込み			コード生成	1.3.0	
▋クロック出力/ブザー出力制御回路			コード生成	1.4.1	
🖥 ディレイ・カウント			コード生成	1.4.1	
🖥 データ・トランスファ・コントローラ			コード生成	1.3.1	
■ ポート			コード生成	1.4.1	┣
- リアルタイム・クロック			コード生成	1.4.0	1
- ワンショット・パルス出力			コード生成	1.4.0	
➡入カパルス間隔/周期測定			コード生成	1.4.3	
■ 入力信号のハイ/ロウ・レベル幅測定	2		コード生成	1.4.2	
分周器機能			コード生成	1.4.2	
割り込みコントローラ			コード生成	1.4.0	▶
▶ 外部イベント・カウント			コード生成	1.4.1	
			コード生成	1.4.0	
万形波出力					

図 8-14 ソフトウェアコンポーネントの選択



- 8. 選択したコンポーネントに対してリソースを設定します。本アプリケーション例では以下の設定で使用 します。
  - 【注】RL78/G23 の場合、SMS を使用した自動判定計測の動作に「ポート」「割り込みコントローラ」の 設定は不要です。コンポーネントの追加を行わないため、表示されません。

択したコンボーネント します	のコンフィグレーションを追		
インターバル・タイマ			
コンフィグレーション名:	Config_ITL000		
動作:	8 ビット・カウンタ・モード	~	
リソース:	ITL000	$\sim$	
注意:			
16ビット・キャプチャ・モー	ドITL000_ITL001は、16ビット・カウント・モードITL012_ITL013と一緒に使用できません。	^	
16ビット・モードITLを使	Hする場合、8ビットITLおよび32ビットITLは使用できません。	~	
ボート			
	a / 0007		
コンフィグレーション名:	Config_PORT		
コンフィグレーション名: リソース:	Config_PORT PORT	~	
コンフィグレーション名: リソース: 割り込みコントローラ	Config_PORT PORT	~	
コンフィグレーション名: リソース: 割り込みコントローラ コンフィグレーション名:	Config_PORT PORT Config_INTC	<ul> <li></li> <li></li> <li></li> <li></li> </ul>	
ッファイグレーション名: リソース: 割り込みコントローラ コンフィグレーション名: リソース:	Config_PORT PORT Config_INTC INTC	<ul> <li></li> <li></li> <li></li> <li></li> <li></li> </ul>	
ッファイグレーション名: リソース: 割り込みコントローラ コンフィグレーション名: リソース:	Config_PORT PORT Config_INTC INTC	<ul> <li></li> <li></li> <li></li> <li></li> </ul>	
<ul> <li>コンフィグレーション名:</li> <li>リソース:</li> <li>割り込みコントローラ</li> <li>コンフィグレーション名:</li> <li>リソース:</li> <li>電圧検出回路</li> <li>コンフィグレーション名:</li> </ul>	Config_PORT PORT Config_INTC INTC Config_IVD0		
<ul> <li>コンフィグレーション名:</li> <li>リソース:</li> <li>割り込みコントローラ</li> <li>コンフィグレーション名:</li> <li>リソース:</li> <li>電圧検出回路</li> <li>コンフィグレーション名:</li> </ul>	Config_PORT PORT Config_INTC INTC Config_LVD0 VD0		

図 8-15 コンポーネントのリソース設定

以下に示すようにコンポーネントが追加されます。

ソフトウェアコンポーネン	卜設定	
コンポーネント	èn ⊿ lª₂ ⊟ ⊞	設定
- i i :::::::::::::::::::::::::::::::::	ت ن	ל <b>ם</b> ו <i>ו</i> דר
2イルタ入力	<b>▶機能</b>	
<ul> <li>Config_ILL00</li> <li>入出力ボート</li> <li>Config_PORT</li> <li>ごドルウェア</li> <li>ジェネリック</li> <li>r_ctsu</li> </ul>		<ul> <li># Bit number for external trigger output</li> <li># Interrupt port number for external trigger</li> </ul>
■ rm_touch 既要 ボード クロック システム (	コンポーネント 端子 割り込	794



- 8.2.2 スマート・コンフィグレータによるコンポーネント設定の変更 追加した各コンポーネントについて以下のように設定を行います。
- CTSU コンポーネント設定 RL78/G22 と RL78/G23 では設定内容が異なります。
  - ① "Data transfer of INTCTSUWR and INTCTSURD"の設定は"DTC"を選択します。
  - "Data storage address setting for CTSURD"の設定は、デバイスによって異なります。 RL78/G22の場合は、「0xFEF00」となります。 RL78/G23の場合は、「0xFF500」となります。
  - ③ RL78/G22 では本設定で、CTSU2La から DTC を介して伝達されるタッチ計測完了信号を出力する ためのポート端子を選択します。RL78/G23 では設定不要のため、表示されません。
    - Output port number for external trigger: 使用するポートグループを選択します。
    - Bit number for external trigger output: 上記ポートグループのうち、使用するビット番号を選択します。
    - 【注意】DTC 転送でポート・レジスタ (Pxx) を 8 ビット単位で書き換えます。
      - したがって SMS を使用した自動判定計測処理の実行中は、DTC の転送先となるポート・ レジスタ (Pxx) を他の機能で使用することはできません。 他システムと競合しないように使用するポート・レジスタ (Pxx) を選択してください。
  - ④ RL78/G22 ではポート端子からの出力信号を入力する外部割り込み端子を設定します。 RL78/G23 では設定不要のため、表示されません。
  - ⑤ タッチ計測に使用する TS 端子を有効にします。
    - 【注意】RL78/G22 では、SMS を使用した自動判定計測で使用するポート端子および外部割り込み端 子との、兼用機能となる TS 端子は使用できません。本アプリケーション例では[P22/TS20]と [P16/INTP5/TS17]を使用するため、TS17 端子と TS20 端子のチェックを外します。 RL78/G23 では任意の TS 端子を選択可能です。



Capacitive_Touch_Project_Example.scfg $\times$		a711	0
フトウェアコンボーネント設定		⑤ <sup>●</sup> □-ドの生成 レポートの生 <sup>●</sup>	成
パーネント 🚵 🖆 🗄 🔛 設定	1	0	D
📲 🏭 🚺 📷	プロパティ	值	
74ルタ入力	Configurations		
	# Parameter check	Use system default	
✓ ✓	# Data transfer of INTCTSUWR and INTCTSURD	DTC	(
V 2 914999	# DTC setting	Setting in r_ctsu	
	# Auto-judgment function in Snooze mode using SMS	Enable	
▼ ● Fノ1 ハ	# Data storage address setting for CTSURD	0xFEF00	(
◆ ● 電源管理CUUPF機能	# Data storage address setting for CTSUWR	0xFF800	
Coning_LVD0	# Interrupt level for INTCTSUWR	Level 2	
	# Interrupt level for INTCTSURD	Level 2	
Conig_INIC	# Interrupt level for INTCTSUFN	Level 2	
Y 📂 21 Y	# Output port number for external trigger	PORT2	
	# Bit number for exremal trigger output	BIT2	
	# Interrupt port number for external trigger	INTP5	(
Config_PORT	/ 回 リソース		
	V 🔲 CTSU		
V - 917090	SCAP端子	☑ 使用する	
r_ctsu	► TS00端子	使用する	
rm_touch	► TS01端子	☑ 使用する	
	► TS02端子	☑ 使用する	
	► TS03端子	<ul> <li>使用する</li> </ul>	
	~ TS04端子	<ul> <li>使用する</li> </ul>	
	► TS05端子	使用する	
	► TS06端子	✓ 使用する	
	~ TS07端子	▼ 使用する	
	~ TS08端子	✓ 使用する	
	~ TS09端子	✓ 使用する	
	► TS10端子	▼ 使用する	
	- TS11端子	☑ 使用する	
	► TS12端子	📝 使用する	
	► TS13端子	☑ 使用する	
	► TS14端子	▼使用する	
	► TS15端子	▼ 使用する	
	► TS16端子	▼ 使用する	
	~ TS17端子	使用する	
	► TS18端子	▼使用する	
	▶ TS19端子	✓ 使用する	•
		原使用する	
	► TS21端子	▼使用する	
		「使用する	
	- TS22端子	() () () () () () () () () () () () () (	
	TC2408-Z	使用する	
	1 J Z 4 2 1 J	C C C C C C C C C C C C C C C C C C C	
	■ TC25端子	使用する	1
	► TS25端子	✓ 使用する	
	<ul> <li>TS25端子</li> <li>TS26端子</li> <li>TS27端子</li> </ul>	<ul> <li>✓ 使用する</li> <li>✓ 使用する</li> <li>Ø 使用する</li> </ul>	

図 8-17 CTSU コンポーネント設定



 Touch コンポーネント設定 デフォルト設定のまま使用します。

コンポーネント 🚵 🛃 🖳 🕀	設定	G
	プロパティ         ✓ @ Configurations         # Parameter check         # Support QE monitor using UART         # Support QE tuning using UART         # UART channel         # Type of chattering suppression	值 Use system default Disable UART0 TypeA : Counter of exceed threshold is hold within hysteresis range.

図 8-18 Touch コンポーネント設定

 INTC コンポーネント設定 RL78/G23 の場合、SMS を使用した自動判定計測の動作に以下の設定は不要です。

コンボーネント 🚵 🛃 🖓 🕒 🕀	設定
▼4 除 フィルタ入力	INTPO設定           □INTPO         有効エッジ         立ち下がりエッジ         優先順位         レベル3(低優先順位)         ~
<ul> <li>         スダートアップ     </li> <li>         シェネリック         ご「r.bsp     </li> </ul>	►INTP1設定 □ INTP1 有効エッジ 立ち下がりエッジ ~ 優先順位 レベル3(低優先順位) ~
<ul> <li>         ドライバ              ・ 電源管理とリセット機能             ・ Config_LVD0          </li> </ul>	INTP2設定     二 INTP2     有効エッジ     立ち下がりエッジ     優先順位     レベル3(低優先順位)     ~
◇ (合) 割り込み Config_INTC ◇ (合) タイマ	INTP3設定 □ INTP3 有効エッジ 立ち下がリエッジ ~ 優先順位 レベル3(低優先順位) ~
<ul> <li>Config_ITL000</li> <li>✓</li></ul>	INTP4設定 □ INTP4 有効エッジ 立ち下がりエッジ ~ 優先順位 レベル3(低優先順位) ~
<ul> <li>         シードルウェア     </li> <li>         ジェネリック         デ r_ctsu     </li> </ul>	□NTP5設定 ✓INTP5 有効エッジ 立ち下がりエッジ > 優先順位 レベル3(低優先順位) >

図 8-19 INTC コンポーネント設定

• ITL コンポーネント設定

コンポーネント 🚵 🖆 📑 🔡	定			
	クロック設定 動作クロック (fTL0) クロック・ソース インター/(ル・タイマ設定 インター/(ル時間 割り込み設定 ☑ コンパアー致またはキャブチャ完了を検 優先順位	fSXP fiTL0/128 20 ms 出 (INTITL) レベル3(低優先順位)	> >	(クロック周波数:0.256 kHz) (実際の値:19.53125)

図 8-20 ITL コンポーネント設定



 PORT コンポーネント設定 RL78/G23 の場合、SMS を使用した自動判定計測の動作に以下の設定は不要です。

コンポーネント 🚵 🛃 🛃 🕀	設定	
<ul> <li>マルタ入力</li> <li>マクートアップ</li> <li>マ スタートアップ</li> <li>マ シジェネリック</li> <li>で r. bsp</li> <li>で r. bsp</li> <li>マ トライパ</li> <li>マ ご 電源管理とりセット機能</li> <li>Config_LVD0</li> <li>マ 割り込み</li> <li>Config_INTC</li> <li>マ タイマ</li> <li>Config_ITL000</li> <li>ロ シリキャガ</li> </ul>	ポート選択 PORT2 PORT0 PORT1 PORT2 PORT3 PORT4 PORT5 PORT6 PORT7 PORT12 PORT13	
◆ → 入田刀ホート	PORT14	
<ul> <li>◇ (⇒ ミドルウェア)</li> <li>◇ (⇒ ジェネリック)</li> <li>◇ 「こたい)</li> <li>◇ 「こたい)</li> </ul>	ポート・モード設定 ● Pmnレジスタ値を読み出す	○ デジタル出力レベルを読み出す

図 8-21 PORT コンポーネント設定 (ポート選択タブ)

コンボーネント 🚵 🛃 🗒 🕀	設定
<ul> <li>         ・</li> <li>         ・</li></ul>	ポート選択 PORT2
<ul> <li>         ◇ (⇒ スタートアップ         ◇ ジェネリック         ◇ 「」たちp         ◇ (⇒ ちっくげ      </li> </ul>	□ <b>すべてに適用</b> ◎ 使用しない ○ 入力 ○ 出力
<ul> <li>              ・ ごのでにす。             ・ ごのでにす。</li> <li>             ・ ごのでにす。</li> <li>             ・ ごのでのでは、</li>             ・ ごのでので、             ・ ごのでので、 </ul> <li>             ・ ごのでので、</li> <li>             ・ ごので、</li> <li>             ・ ごので、</li> <li>             ・ ごので、</li> <li>             ・ こので、</li> ・ こので、 <li>             ・ こので、</li> ・ こので、 <li>             ・ ・ こので、</li> <li>             ・ ・ こので、</li> <li>             ・ ・ ・             ・</li>	P20 ●使用しない ○ 入力 ○ 出力
	P21 ●使用しない ○入力 ○出力
	- P22 〇 使用しない 〇 入力 ⑧ 出力
r_ctsu	P23 ● 使用しない ○ 入力 ○ 出力

図 8-22 PORT コンポーネント設定 (PORT2 タブ)

• LVD コンポーネント設定

コンボーネント 🚵 🛃 🚊 🕀	設定
<ul> <li>         マロトアップ         マートアップ         マートアップ</li></ul>	
<ul> <li>Contig_LVD0</li> <li>ショリ込み</li> <li>Config_INTC</li> <li>シタイマ</li> <li>Config_ITL000</li> <li>シスカガート</li> <li>Config_PORT</li> <li>Config_PORT</li> <li>Config_PORT</li> <li>シェアルウェア</li> <li>ディctsu</li> <li>m_touch</li> </ul>	<ul> <li>一電圧検出設定</li> <li>リセット発生電圧(VLVD0)</li> <li>2.62 ✓ (V)</li> <li>割込み発生電圧(VLVD0)</li> <li>1.65 (V)</li> </ul>

図 8-23 LVD コンポーネント設定



• BSP コンポーネント設定

"Initialization of peripheral functions by Code Generator/Smart Configurator"が"Enable"に設定されている ことを確認します。

8			– 🗆 🗙
*Capacitive_Touch_Project_Example.scfg ×			
ソフトウェアコンポーネント設定			3−ドの生成 レポートの生成
コンポーネント 🚵 🛃 🛱 🖽	設定		
	JU/T71         ✓ ● Configurations         # Start up select         # Control of illicit memory access detection(IAWEN)         # Protection of the interrupt control registers(GPORT)         # Protection of the interrupt control registers(GINT)         # Data flash memory area access control(OFLEN)         # Initialization of peripheral functions by Code Generator/Smart Configurator         # API functions disable(R_BSP_GetFicKreqHz)         # API functions disable(R_BSP_SetClockSource)         # API functions disable(R_BSP_ChangeClockSetting)         # API functions disable(R_BSP_SoftwareDelay)         # Parameter check enable         # Enable user warm start callback (PRE)         # User warm start callback function name (PRE)         # Enable user warm start callback function name (POST)         # Watchdog Timer initialize user function name	僅 Enable (use BSP startup) Disabled Disabled Disabled Disabled Enable Disable Enable Disable Enable Disable Enable Unused Unused Unused Unused Unused Unused Unused Unused	
概要 ボード クロック システム ユンボーネント 端子 割り	込み		~

#### 図 8-24 BSP コンポーネント設定

スマート・コンフィグレータの右上の"コードの生成"アイコンをクリックして、プロジェクトに必要なコン ポーネントのコードを生成します。以上でコンポーネントの追加は完了です。



# RL78 ファミリ 静電容量タッチ低消費電力アプリケーション(SMS 使用)の開発

8.3 静電容量タッチインタフェース作成

QE for Capacitive Touch を使用して静電容量タッチインタフェースの設定を作成します。

以下に手順を示します。

なお RL78/G22 と RL78/G23 では一部設定内容が異なります。

 e<sup>2</sup> studio のメニュー[Renesas Views] - [Renesas QE] - [CapTouch メイン (QE)]/QE V3.2.0 以降では [CapTouch ワークフロー (QE)]を選択し、プロジェクトに静電容量タッチの設定をするためのメイン ウィンドウを表示します。

💽 ws-2404 - Capacitive_Touch_Project_Example/Capacitive_Touch_Project_Example.scfg - e <sup>2</sup> studio					
ファイル(F) 編集(E) ナビゲート(N) 検索(A) プロジェクト(P)	Renesas Views 実行(R) Renesa	ıs AI ウィンドウ(W) ヘルプ(H)			
🔚 🛞 🕶 🔦 🕶 🖓 😂 🗳 🐐 🕶 💁 🕶	C/C++	>			
🔁 プロジェクト・エクスプローラー 🗙 📄 😫 🍞 🖇 🖆	Renesas Al				
> 🚝 Capacitive Touch Project Example [HardwareDe	Renesas QE	Caplouch SIX + v + = y (QE)			
	コート生成	<sup>2</sup> 2 CapTouchボード・モニタ (QE)			
		<sup>2</sup> 💟 CapTouchパッド・モニタ (QE)			
	ソリュージョン・ツールキット	2 CapTouchマルチ・ステータス・チャート (QE)			
		く 😂 CapTouchパラメータ一覧 (QE)			
		👌 눬 CapTouchステータス・チャート (QE)			
	Renesas Software Installer	—— 😓 CapTouch調整結果 (QE)	h		
	🗸 🧁 スタートアップ	💫 CapTouchワークフロー (QE)			
	🗸 🗁 ジェネリック	่ 消費電流測定 (QE)			
	- r hen				

図 8-25 Cap Touch ワークフロー(QE)の選択

 "CapTouch メイン (QE)/QE V3.2.0 以降では[CapTouch ワークフロー (QE)]"の"プロジェクトの選択" のプルダウンメニューから"Capacitive\_Touch\_Project\_Example"を選択し、タッチインタフェースを 設定するプロジェクトを選択します。

i	🏷 CapTouchワークフロー (QE) 🗙 📑 🛱	3
	▲ 準備 調整 実装 動作確認	
	1.静電容量タッチの準備	
	● プロジェクトの選択	
	Ļ	
	✓ ✓	
	Canacitive Touch Project Evample	
	~	
	「「「「」「「」」「「」」「」」「「」」「」」「「」」「」」「」」「」」「」」	

図 8-26 プロジェクトの選択

3. "構成の選択"のプルダウンメニューから[**タッチインタフェース構成の新規作成**]を選択し、新しいタッチ インタフェース構成を生成します。



図 8-27 タッチインタフェース構成の新規作成



4. "タッチインタフェース構成の作成"ウィンドウが開き、タッチインタフェースを配置する領域が表示されます。

💽 タッチインタフェース構成の作成				×
タッチインタフェース構成のファイル名:	Capacitive_Touch_Project_Example	構成(メソッド)の設定	構成の流用/再編集	
説明:				
			ዎッチI/F	*
			静電容量方式	
			自己容量方式	~
			ボタン	
			スライダ(横方向)	
			スライダ(縦方向)	
			ホイール	
			キーパッド	
			3Dジェスチャ (Al)	
			タッチパッド	
			シールド端子	
			温度補正端子	
			容量センサ	
			電流センサ	
			診断コード用端子	
売白			タッチI/Fの削除	
タッチ1/Fの設定 総相	5.抗の設定 割り付けTSxの解除		構成(メソッド)の確認	*
		作	ず(C) キャンヤル ヘルプ(H)	

図 8-28 タッチインタフェース配置領域

- 5. ウィンドウの右側から[**ボタン**]を選択して3つのボタンを画面に追加してください。 キーボードの[**Esc**]キーを押してタッチインタフェースの追加を終了すると以下の状態になります。
  - 【備考】次の手順で各ボタンへタッチセンサが問題なく割り当てられると「設定内容に問題があります。」 というエラー表示は消えます。

そのため、このまま次の手順に進むことができます。

77712771 X16/001F/00		
ッチインタフェース構成のファイル名:	Capacitive_Touch_Project_Example 構成(メソッド)の設定	構成の流用/再編集
明:		
	Rutton00	タッチI/F
	Buttonio	静電容量方式
		自己容量方式
	Button01	ボタン
		スライダ(横方向)
		スライダ(縦方向)
	Button02	ホイール
		キーパッド
		3Dジェスチャ (AI)
		タッチパッド
		シールド端子
		温度補正端子
		容量センサ
		電流センサ
Ē		診断コード用端子
タッチI/Fの設定	総抵抗の設定 割り付けTSxの解除	タッチI/Fの削除
設定内容に問題があります。		構成(メソッド)の確認
🛿 設定内容に問題があります。		構成(メソッド)の確認
		作成( <u>C</u> ) キャンセル ヘルブ( <u>H</u> )

図 8-29 タッチインタフェース (ボタン3個) の追加

-



## RL78 ファミリ 静電容量タッチ低消費電力アプリケーション(SMS 使用)の開発

6. 各ボタンの名前とタッチセンサの割り当てを行います。

"Button00"をダブルクリックし、"タッチインタフェースの設定"ダイアログを表示します。ここでは 名前とタッチセンサ (赤枠) を変更して[OK]をクリックします。 RL78/G23 の場合は、Button00 に TS06 を割り当てます。

- 【注意】1. SMS 起動用の外部トリガとして使用するポート端子と端子機能を兼用するタッチセンサ (TSxx) は使用できません。
  - 2. ボタン名称は任意に設定できますが、文字数制限によりエラーが表示される場合があります。

Capacitive_Touch_Project_Example	構成(メソッド)の設定	
TS_B1		
TS28 •		
Button01	📴 タッチインタフェースの設定	×
	ボタン(自己)	
Button02	名前 TS_B1	
	タッチセンサ抵抗値[Ω]TS28560	
	OK キャンセル ヘル	プ(H)
<u>.</u>		

R01AN7261JJ0100 Rev.1.00 Jul.22.24



"Button00(TS\_B1)"と同様に、Button01 と Button02の設定を行います。
 タッチインタフェースの設定に問題が無い場合はエラー表示が消えます。
 RL78/G23の場合は、Button01に TS05、Button02に TS07を割り当てます。

▋ タッチインタフェース構成の作成				
タッチインタフェース構成のファイル名:	Capacitive_Touch_Project_Example	構成(メソッド)の設定		構成の流用/再編集
説明:				
	TS B1		タッチ	-I/F
	-		静雷	2容量方式
	TS28		自己	容量方式
	TS_B2			ボタン
	7010			スライダ(横方向)
	1518			スライダ(縦方向)
	TS_B3			ホイール
	TSOO			キーパッド
				3Dジェスチャ (AI)
				タッチパッド
				シールド端子
				温度補正端子
				容量センサ
				電流センサ
設定				診断コード用端子
タッチI/Fの設定	総抵抗の設定割り付けTSxの解除			タッチI/Fの削除
			構成	(メソッド)の確認
			作成( <u>C</u> )	キャンセル ヘルプ(日)

図 8-31 設定後のタッチインタフェース(ボタン3個)



## RL78 ファミリ 静電容量タッチ低消費電力アプリケーション(SMS 使用)の開発

- ボタンと同様にホイールとスライダを設定します。
   本アプリケーション例では以下の設定で使用します。
   すべてのタッチインタフェースの配置が完了したら、[構成(メソッド)の設定]をクリックします。
  - 【注意】「設定内容に問題があります。」というエラーが再度表示されますが、手順9を実行すると エラー表示は消えます。
  - 【備考】本アプリケーション例では、スライダおよびホイールに割り当てたタッチセンサを、低消費モー ド中は MEC 機能ボタンへと切り替えて使用します。 SMS、MEC を使用したタッチ計測(低消費モード)で使用するタッチインタフェースは 「TS\_xxMEC」と命名しています。

● ダッナイノダノエース構成の1F成	
タッチインタフェース構成のファイル名: Capacitive_Touch_Project_Example 構成(メソッド)の設定 説明:	構成の流用/再編集
TS_W2MEC TS_B1	タッチI/F
T519 Wheel T508 T528	静電容量方式 自己容量方式
T519 T508 T5_B2	<b>ポ</b> タン
TC10	スライダ(横方向)
TS W3MEC TS21 TS24 TS W4MEC	スライダ(縦方向)
TSB2	ホイール
TS21 TS24	キーパッド
TS00	3Dジェスチャ (AI)
Slider	タッチパッド
TS04 TS05 TS06 TS07 TS01	シールド端子
	温度補正端子
TS_S1MEC TS_S2MEC TS_S3MEC TS_S4MEC TS_S5MEC	容量センサ
TS04 TS05 TS06 TS07 TS01	電流センサ
	診断コード用端子
	タッチI/Fの削除
	構成(メソッド)の確認 ¥
A THE REPORT REPORT REPORT REPORT REPORT	

図 8-32 タッチインタフェース (ホイール, スライダ) の追加 (RL78/G22)



(メソッド)の設定"ウィンドウで自動判定機能と複数電極接続機能の設定をします。
 SMSによる自動判定計測を使用する場合は"自動判定 (SMS)"を、複数電極接続 (MEC) 機能を使用する場合は"複数電極接続"を、それぞれ"有効にする"に設定します。

本アプリケーション例では、通常モードと低消費モードを実装するためにタッチインタフェース構成 (メソッド)を分けています。[構成 (メソッド)の追加]をクリックし、config01の横に config02 を追加し ます。

各コンフィグに対して設定を行い、[**OK**]をクリックします。

--- config01:低消費モードでのタッチインタフェース構成 (メソッド)の機能設定 --- config02:通常モードでのタッチインタフェース構成 (メソッド)の機能設定

【注意】CTSU モジュールの V1.50 を使用する場合は、config01 にのみ自動判定 (SMS) 機能を適用できます。

TS_W1MEC(自己)       ダ 有効         TS_W2MEC(自己)       ダ 有効         TS_W3MEC(自己)       ダ 有効         TS_SIMEC(自己)       ダ 有効         TS_SIMEC(自己)       ダ 有効         TS_SAMEC(自己)       ダ 有効         TS_B1(自己)       ダ 有効         TS_B2(自己)       ダ 有効         Wheel(自己)       ダ 有効         Slider(自己)           Slider(自己)           Slider(自己)           Slider(自己)           Slider(日	TS_W1MEC(自己)       ダ 有効         TS_W2MEC(自己)       ダ 有効         TS_W3MEC(自己)       ダ 有効         TS_W4MEC(自己)       ダ 有効         TS_STMEC(自己)       ダ 有効         TS_S2MEC(自己)       ダ 有効         TS_S3MEC(自己)       ダ 有効         TS_S3MEC(自己)       ダ 有効         TS_S5MEC(自己)       ダ 有効         TS_S5MEC(自己)       ダ 有効         TS_S1(自己)       ダ 有効         TS_B1(自己)       ダ 有効         TS_B2(自己)       ダ 有効         SUB(自己)       ダ 有効         Silder(自己)       ダ 有効	复数雷板接続		
TS_W1MEC(自己)         有効          TS_W2MEC(自己)         有効          TS_W3MEC(自己)             TS_SUMEC(自己)              TS_S1MEC(自己)               TS_S3MEC(自己)                TS_S3MEC(自己) <th>TS_W1MEC(自己)       ☑ 有効         TS_W2MEC(自己)       ☑ 有効         TS_W3MEC(自己)       ☑ 有効         TS_W4MEC(自己)       ☑ 有効         TS_STMEC(自己)       ☑ 有効         TS_S2MEC(自己)       ☑ 有効         TS_S3MEC(自己)       ☑ 有効         TS_S3MEC(自己)       ☑ 有効         TS_S4MEC(自己)       ☑ 有効         TS_S4MEC(自己)       ☑ 有効         TS_S5MEC(自己)       ☑ 有効         TS_B1(自己)       ☑ 有効         TS_B2(自己)       ☑ 有効         Whee(自己)       ☑ 有効         Slider(自己)       ☑</th> <th>自動判定(SMS)</th> <th>□有効にする</th> <th>□有効にする</th>	TS_W1MEC(自己)       ☑ 有効         TS_W2MEC(自己)       ☑ 有効         TS_W3MEC(自己)       ☑ 有効         TS_W4MEC(自己)       ☑ 有効         TS_STMEC(自己)       ☑ 有効         TS_S2MEC(自己)       ☑ 有効         TS_S3MEC(自己)       ☑ 有効         TS_S3MEC(自己)       ☑ 有効         TS_S4MEC(自己)       ☑ 有効         TS_S4MEC(自己)       ☑ 有効         TS_S5MEC(自己)       ☑ 有効         TS_B1(自己)       ☑ 有効         TS_B2(自己)       ☑ 有効         Whee(自己)       ☑ 有効         Slider(自己)       ☑	自動判定(SMS)	□有効にする	□有効にする
TS_W1MEC(自己)       ☑ 有効       □         TS_W2MEC(自己)       ☑ 有効       □         TS_W3MEC(自己)       ☑ 有効       □         TS_SIMEC(自己)       ☑ 有効       □         TS_SAMEC(自己)       ☑ 有効       □         TS_B3(自己)       ☑ 有効       ☑ 有効         TS_B3(自己)       ☑ 有効       ☑ 有効         Wheel(自己)       □       ☑ 有効	TS_W1MEC(自己)       ✓ 有効         TS_W2MEC(自己)       ✓ 有効         TS_W3MEC(自己)       ✓ 有効         TS_W4MEC(自己)       ✓ 有効         TS_STMEC(自己)       ✓ 有効         TS_SAMEC(自己)       ✓ 有効         TS_B3(自己)       ✓ 有効         TS_B3(自己)       ✓ 有効         Wheel(自己)       ✓ 有効	Slider(自己)		[✔] 有効
TS_W1MEC(自己)       ☑ 有効       □         TS_W2MEC(自己)       ☑ 有効       □         TS_W3MEC(自己)       ☑ 有効       □         TS_SUMEC(自己)       ☑ 有効       □         TS_S1MEC(自己)       ☑ 有効       □         TS_S3MEC(自己)       ☑ 有効       □         TS_S3MEC(自己)       ☑ 有効       □         TS_S5MEC(自己)       ☑ 有効       □         TS_S5MEC(自己)       ☑ 有効       □         TS_S8MEC(自己)       ☑ 有効       □         TS_S81(自己)       ☑ 有効       ☑ 有効         TS_B3(自己)       ☑ 有効       ☑ 有効         TS_B3(自己)       ☑ 有効       ☑ 有効	TS_W1MEC(自己)       ✓ 有効         TS_W2MEC(自己)       ✓ 有効         TS_W3MEC(自己)       ✓ 有効         TS_STMEC(自己)       ✓ 有効         TS_S2MEC(自己)       ✓ 有効         TS_S3MEC(自己)       ✓ 有効         TS_S3MEC(自己)       ✓ 有効         TS_S3MEC(自己)       ✓ 有効         TS_S3MEC(自己)       ✓ 有効         TS_S5MEC(自己)       ✓ 有効         TS_S6(自己)       ✓ 有効         TS_B1(自己)       ✓ 有効         TS_B2(自己)       ✓ 有効         TS_B3(自己)       ✓ 有効	Wheel(自己)		☑ 有効
TS_W1MEC(自己)       ☑ 有効       □         TS_W2MEC(自己)       ☑ 有効       □         TS_W3MEC(自己)       ☑ 有効       □         TS_SUMEC(自己)       ☑ 有効       □         TS_S1MEC(自己)       ☑ 有効       □         TS_S3MEC(自己)       ☑ 有効       □         TS_S3MEC(自己)       ☑ 有効       □         TS_SAMEC(自己)       ☑ 有効       □         TS_S4MEC(自己)       ☑ 有効       □         TS_S8MEC(自己)       ☑ 有効       □         TS_S81(自己)       ☑ 有効       ☑ 有効         TS_B2(自己)       ☑ 有効       ☑ 有効	TS_W1MEC(自己)       ダ 有効       □         TS_W2MEC(自己)       ダ 有効       □         TS_W3MEC(自己)       ダ 有効       □         TS_W4MEC(自己)       ダ 有効       □         TS_S1MEC(自己)       ダ 有効       □         TS_S1MEC(自己)       ダ 有効       □         TS_S3MEC(自己)       ダ 有効       □         TS_S3MEC(自己)       ダ 有効       □         TS_S5MEC(自己)       ダ 有効       □         TS_S1(自己)       ダ 有効       □         TS_B1(自己)       ダ 有効       □         TS_B2(自己)       ダ 有効       □	TS_B3(自己)	☑ 有効	✓ 有効
TS_W1MEC(自己)       ☑ 有効       □         TS_W2MEC(自己)       ☑ 有効       □         TS_W3MEC(自己)       ☑ 有効       □         TS_SUMEC(自己)       ☑ 有効       □         TS_SAMEC(自己)       ☑ 有効       □         TS_SMEC(自己)       ☑ 有効       □         TS_B1(自己)       ☑ 有効       ☑ 有効	TS_W1MEC(自己)       ☑ 有効       □         TS_W2MEC(自己)       ☑ 有効       □         TS_W3MEC(自己)       ☑ 有効       □         TS_STMEC(自己)       ☑ 有効       □         TS_S1MEC(自己)       ☑ 有効       □         TS_S3MEC(自己)       ☑ 有効       □         TS_S3MEC(自己)       ☑ 有効       □         TS_S3MEC(自己)       ☑ 有効       □         TS_S5MEC(自己)       ☑ 有効       □         TS_S5MEC(自己)       ☑ 有効       □         TS_S1(自己)       ☑ 有効       □	TS_B2(自己)	✔ 有効	✓ 有効
TS_W1MEC(自己)       ☑ 有効       □         TS_W2MEC(自己)       ☑ 有効       □         TS_W3MEC(自己)       ☑ 有効       □         TS_S1MEC(自己)       ☑ 有効       □         TS_S2MEC(自己)       ☑ 有効       □         TS_S3MEC(自己)       ☑ 有効       □         TS_S3MEC(自己)       ☑ 有効       □         TS_S3MEC(自己)       ☑ 有効       □         TS_S3MEC(自己)       ☑ 有効       □	TS_W1MEC(自己)       ☑ 有効         TS_W2MEC(自己)       ☑ 有効         TS_W3MEC(自己)       ☑ 有効         TS_SUMEC(自己)       ☑ 有効         TS_S2MEC(自己)       ☑ 有効         TS_S3MEC(自己)       ☑ 有効         TS_S4MEC(自己)       ☑ 有効         TS_S3MEC(自己)       ☑ 有効         TS_S5MEC(自己)       ☑ 有効         TS_SSMEC(自己)       ☑ 有効	TS_B1(自己)	☑ 有効	☑ 有効
TS_W1MEC(自己)       ☑ 有効       □         TS_W2MEC(自己)       ☑ 有効       □         TS_W3MEC(自己)       ☑ 有効       □         TS_S1MEC(自己)       ☑ 有効       □         TS_S2MEC(自己)       ☑ 有効       □         TS_S3MEC(自己)       ☑ 有効       □         TS_S3MEC(自己)       ☑ 有効       □         TS_S4MEC(自己)       ☑ 有効       □	TS_W1MEC(自己)       ☑ 有効         TS_W2MEC(自己)       ☑ 有効         TS_W3MEC(自己)       ☑ 有効         TS_SUMEC(自己)       ☑ 有効         TS_S2MEC(自己)       ☑ 有効         TS_S3MEC(自己)       ☑ 有効         TS_S4MEC(自己)       ☑ 有効         TS_SAMEC(自己)       ☑ 有効         TS_SAMEC(自己)       ☑ 有効	TS_S5MEC(自己)	☑ 有効	
TS_W1MEC(自己)     ☑ 有効     □       TS_W2MEC(自己)     ☑ 有効     □       TS_W3MEC(自己)     ☑ 有効     □       TS_S1MEC(自己)     ☑ 有効     □       TS_S3MEC(自己)     ☑ 有効     □	TS_W1MEC(自己)     ☑ 有効       TS_W2MEC(自己)     ☑ 有効       TS_W3MEC(自己)     ☑ 有効       TS_S4MEC(自己)     ☑ 有効       TS_S2MEC(自己)     ☑ 有効       TS_S3MEC(自己)     ☑ 有効	TS_S4MEC(自己)	☑ 有効	
TS_W1MEC(自己)     ☑ 有効     □       TS_W2MEC(自己)     ☑ 有効     □       TS_W3MEC(自己)     ☑ 有効     □       TS_S1MEC(自己)     ☑ 有効     □       TS_S2MEC(自己)     ☑ 有効     □	TS_W1MEC(自己)     ☑ 有効       TS_W2MEC(自己)     ☑ 有効       TS_W3MEC(自己)     ☑ 有効       TS_SUMEC(自己)     ☑ 有効       TS_S1MEC(自己)     ☑ 有効	TS_S3MEC(自己)	☑ 有効	
TS_W1MEC(自己)     ✓ 有効       TS_W2MEC(自己)     ✓ 有効       TS_W3MEC(自己)     ✓ 有効       TS_V4MEC(自己)     ✓ 有効       TS_S1MEC(自己)     ✓ 有効	TS_W1MEC(自己)     ☑ 有効       TS_W2MEC(自己)     ☑ 有効       TS_W3MEC(自己)     ☑ 有効       TS_V4MEC(自己)     ☑ 有効       TS_S1MEC(自己)     ☑ 有効	TS_S2MEC(自己)	✔ 有効	
TS_W1MEC(自己)     ☑ 有効       TS_W2MEC(自己)     ☑ 有効       TS_W3MEC(自己)     ☑ 有効       TS_W4MEC(自己)     ☑ 有効	TS_W1MEC(自己)     ☑ 有効       TS_W2MEC(自己)     ☑ 有効       TS_W3MEC(自己)     ☑ 有効       TS_W4MEC(自己)     ☑ 有効	TS_S1MEC(自己)	✔ 有効	
TS_W1MEC(自己)     ☑ 有効       TS_W2MEC(自己)     ☑ 有効       TS_W3MEC(自己)     ☑ 有効	TS_W1MEC(自己)     ✓ 有効       TS_W2MEC(自己)     ✓ 有効       TS_W3MEC(自己)     ✓ 有効	TS_W4MEC(自己)	☑ 有効	
TS_W1MEC(自己)	TS_W1MEC(自己)	TS_W3MEC(自己)	✔ 有効	
TS_W1MEC(自己)	TS_W1MEC(自己)	TS_W2MEC(自己)	☑ 有効	
		TS W1MEC(自己)	✔ 有効	

図 8-33 構成 (メソッド)の設定 (RL78/G22)



10.エラーが表示されないことを確認して"タッチインタフェース構成の作成"ウィンドウの[**作成**]をクリックします。

これで、RL78/G22の場合のタッチインタフェースの作成は完了です。

🕑 タッチインタフェース構成の作成		×
タッチインタフェース構成のファイル名: 説明:	Capacitive_Touch_Project_Example 構成(メソッド)の設定	構成の流用/再編集
T5, W2MEC T519 T519 T519 T519 T519 T521 T521 T521 T521 T521 T521 T521 T525 T	TS_WIMEC TS_B1 TS0B TS2B TS_WAMEC TS_B2 TS24 TS1B TS24 TS_B3 TS00 ider TS24 TS01 SOB TS07 TS01	タッチルチ       ※         静電容量方式       自己容量方式         自己容量方式       >         水タン       スライダ(横方向)         スライダ(横方向)       スライダ(横方向)         スライダ(横方向)       ホイール         オーパッド       30ジェスチャ(Al)         タッチパッド       シールド端子         温度補正端子       客量センサ         電流センサ       電流センサ         参助コード用端子       参助コード用端子
設定	抵抗の設定 割り付けTSxの解除	タッチI/Fの創除 構成(メソッド)の確認     ×       作成(C)     キャンセル       へルブ(H)





11.以降では RL78/G23 のタッチインタフェース構成例を示します。 基本的な設定手順および諸注意は RL78/G22 と同様です。

- 【注意】RL78/G22 と RL78/G23 では電極に割り当てるタッチセンサが異なります。
- 【備考】SMS を使用したタッチ計測(低消費モード)で使用するタッチインタフェースは「TS\_xxBTN」 と命名しています。



図 8-35 タッチインタフェース (ホイール, スライダ) の追加 (RL78/G23)



- 12. "構成 (メソッド)の設定"ウィンドウで自動判定機能の設定をします。 本アプリケーション例では下記設定で使用します。
  - config01:低消費モードでのタッチインタフェースの機能設定
  - config02 : 通常モードでのタッチインタフェースの機能設定
  - 【注意】CTSU モジュールの V1.50 を使用する場合は、config01 にのみ自動判定 (SMS) 機能を適用できます。

🗃 構成(メソッド)の設定			>
構成(メソッド)の追加 株			
	config01	config02	
TS_W1BTN(自己)	☑ 有効		
TS_W2BTN(自己)	✔ 有効		
TS_W3BTN(自己)	✔ 有効		
TS_W4BTN(自己)	✔ 有効		
TS_S1BTN(自己)	✔ 有効		
TS_S2BTN(自己)	✔ 有効		
TS_S3BTN(自己)	✔ 有効		
TS_S4BTN(自己)	✔ 有効		
TS_S5BTN(自己)	✔ 有効		
TS_B1(自己)	✔ 有効	✓ 有効	
TS_B2(自己)	✔ 有効	✓ 有効	
TS_B3(自己)	✔ 有効	✓ 有効	
Wheel(自己)		✓ 有効	
Slider(自己)		✔ 有効	
自動判定(SMS)	✓ 有効にする	□有効にする	
	OK	キャンセル ヘルプ(H	)
	$\widehat{1}$		
図 8-36 構成(	ー メソッド) の	設定 (RL78/G23	3)



13.エラーがないことを確認して"タッチインタフェース構成の作成"ウィンドウの[**作成**]をクリックします。 これで、RL78/G23の場合のタッチインタフェースの作成は完了です。



図 8-37 設定後のタッチインタフェース構成図 (RL78/G23)

14. "CapTouch メイン (QE)/QE V3.2.0 以降では[CapTouch 調整結果 (QE)]"の"チューニング"パネルに タッチインタフェースの構成が表示されます。

e								—		×
<b>b</b> c	apToucl	h調整結果 (Q	E) ×						8	- 0
۶- ۲1-	ニング	ジェスチャ								
		22/17/								
タッ	チインタン	フェース構成: Ca	pacitive_1	Fouch_Project_Example						
X	/ッド	種別	名前	タッチセンサ	寄生容量[pF]	センサドライブパルス周波数[MHz]	しきい値	計測時間[ms]	オーバーフロー	^
со	nfig01	ボタン(自己)	Mec00	TS00	-	-	-	-	なし	
со	nfig02	ボタン(自己)	TS_B1	TS28	-	-	-	-	なし	
со	nfig02	ボタン(自己)	TS_B2	TS18	-	-	-	-	なし	
со	nfig02	ボタン(自己)	TS_B3	TS00	-	-	-	-	なし	
со	nfig02	ホイール	Wheel	TS19, TS08, TS24, TS21	-	-	-	-	なし	
со	nfig02	ホイールTS	(Wheel)	TS19	-	-	-	-	-	
со	nfig02	ホイールTS	(Wheel)	TS08	-	-	-	-	-	
со	nfig02	ホイールTS	(Wheel)	TS24	-	-	-	-	-	
со	nfig02	ホイールTS	(Wheel)	TS21	-	-	-	-	-	
со	nfig02	スライダ	Slider	TS04, TS05, TS06, TS07, TS01	-	-	-	-	なし	
со	nfig02	スライダTS	(Slider)	TS04	-	-	-	-	-	
со	nfig02	スライダTS	(Slider)	TS05	-	-	-	-	-	
со	nfig02	スライダTS	(Slider)	TS06	-	-	-	-	-	
со	nfig02	スライダTS	(Slider)	TS07	-	-	-	-	-	
со	nfig02	スライダTS	(Slider)	TS01	-	-	-	-	-	$\sim$

図 8-38 [CapTouch 調整結果 (QE)] の"チューニング"パネル



15. e<sup>2</sup> studio 左上の<sup>低</sup>アイコンをクリックしてプロジェクトのビルドを開始します。



図 8-39 プロジェクトのビルド

"コンソール"ウィンドウで、ビルドした結果にエラーがないことが確認できます。 これで静電容量タッチインタフェースの作成は完了です。

【注意】プロジェクトをビルドした際、コンソールに以下のエラー(E0562310) が出た場合は、作成した プロジェクトのプロパティから [**C/C++ ビルド**] - [**設定**] - [**Linker**] - [**入力**]を開き、 ランタイム・ライブラリを使用するにチェックを入れて再ビルドしてください。



図 8-40 コンソールウィンドウ (エラー: E0562310 発生時)

<ul> <li>&gt; リソース</li> <li>&gt; C/C++ ビルド スタック解析 ツールチェイン・エディター ビルド変数 ロギング 環境 設定</li> </ul>	<ul> <li>▼ いいの SMS Assembler</li> <li>※ ソース</li> <li>※ オブジェクト</li> <li>※ ユーザー</li> <li>▼ いいのの</li> <li>※ Common</li> <li>※ CPU</li> <li>※ デバイス</li> <li>※ その他</li> </ul>	<ul> <li>標準・数学ライブラリを使用する (-library)</li> <li>✓ C99版ライブラリを使用する (-library)</li> <li>メモリの解放時にメモリ破壊を検出する (-library)</li> <li>✓ ランタイム・ライブラリを使用する (-library)</li> <li>リンクするリロケータブル・ファイル、オブジェクト・ファイル</li> </ul>
---	--	--

図 8-41 ランタイム・ライブラリの設定



8.4 静電容量タッチセンサ・チューニング向けデバッグ構成の設定変更 デバッグセッション開始後にチューニングカーネルを MCU の RAM にダウンロードできるように、

デバッグ構成を変更します。

1. \*\* アイコン横の▼をクリックし、タブから"デバックの構成"を選択してください。



図 8-42 "デバックの構成"の選択

- デバッグ構成ウィンドウで[Debugger] [Connection Settings]とタブを選択します。
   以下のようにターゲット・ボードとの接続の設定を行ってください。
  - 【注意】1.動作確認を簡単に行うために、このアプリケーション例では、ターゲットボードの電源は エミュレータの電源から供給します。なお PC の USB ポートから、E2 emulator Lite を経由 してターゲットボードに電源を供給することは可能ですが、ルネサスではターゲットボード で生成する電源を使用することを推奨しています。
    - どのデバッグ方法が使用可能かは、ターゲットボードの仕様によります。使用するデバッグ 方法に応じて"Debug hardware:"を選択し、項目を設定してください。 例えば、COM port デバッグを行う場合は、設定が異なります。

— デバック方法: E2 Lite

● デバッグ構成		
構成の作成、管理、および実行		
'起動時にフラッシュを消去' オプションが有効になっています。正常に接続	したあと、このオプションを無効にしてください。	
	名前(N): Capacitive_Touch_Project_Example HardwareDe	ebug
7ィルタ入力	📄 メイン 🕸 Debugger 🔪 🦊 共通 🦤 ソーフ	ξ
C/C++ アプリケーション		
C/C++ リモート・アプリケーション	Debug hardware: E2 Lite (RL78) $\checkmark$ Target Devi	ce: R7F102GGE
EASE Script		
GDB Simulator Debugging (RH850)	GDB Settings Connection Settings	
C GDB ハードウェア・デバッギング	✓ クロック	
✓ C Renesas GDB Hardware Debugging	メイン・クロック周波数 [MHz]	内部クロックの使用
Capacitive_Touch_Project_Example HardwareDebug	サブ・クロック周波数 [kHz]	内部クロックの使用
Renesas Simulator Debugging (RX, RL78)	モニター・クロック	システム
🚭 起動グループ	✓ ターゲット・ボードとの接続	
	エミュレーター	(Auto)
	低電圧OCDボードを使用する	いいえ
	エミュレーターから電源供給 (最大 200mA)	はい
	供給電圧[V]	3.3
	Hot Plug	いいえ

#### 図 8-43 デバック構成の設定 (E2 Lite)

— デバック方法:COM port

) メイン 🏇 Deb	ugger 🕨 Startup 🔳	] 共	:通(C) 🏷 ソース	
Debug hardwa	re: COM Port (RL78)	~	et Device: R7F100GSN	] <
GDB Settings	Connection Settings	Ŧ	バッグ・ツール設定	
~ クロック				
メイン・ク	ロック周波数 [MHz]		内部クロックの使用	× .
サブ・クロ	]ック周波数 [kHz]		内部クロックの使用	~
モニター・	クロック		システム	~
~ ターゲット・オ	「ードとの接続			_
COM#-	-ト		COM9	
リセット制	间御端子	Ī	DTR	
> フラッシュ				- 1

図 8-44 デバック構成の設定 (COM port)

3. [**デバック・ツール設定**]タブを選択します。"メモリー"の"実行を一時停止してメモリアクセスする"を "はい"に設定してください。

💽 デパッグ構成		
構成の作成、管理、および実行		
'起動時にフラッシュを消去' オブションが有効になっています。 正常に接続し	たあと、このオプションを無効にしてください。	
🗋 🖻 🍋 🗎 🗶 🖻 🏹 🗸	名前(N): Capacitive_Touch_Project_Example Hardy	wareDebug
フィルタ入力	🖹 🔨 🔅 Deburgger 🖿 Startup 🔲 # 🗃 🗄	
	Startup H HE	
C/C++ //	Debug hardware: E2 Lite (RL78) V Targe	et Device: R7F102GGE
EASE Script		
GDB Simulator Debugging (RH850)	GDB Settings Connection Settings デバッグ・ツー	-ル設定
GDB ハードウェア・デバッギング	× 10	
✓	デフォルト IO ファイル名を使用	はい
Capacitive_Touch_Project_Example HardwareDebug	IO ファイル名	\${support_area_loc}
💽 Renesas Simulator Debugging (RX, RL78)	~ 一般	
🕞 起動グループ	ダウンロード後にリセットする	はい
	✓ 中断	
	停止中はタイマー・グループのエミュレーションを停	亭止する いいえ
	停止中はシリアル・グループのエミュレーションを得	亭止する いいえ
	◇ 入力信号のマスク	
	ターゲット・リセット信号のマスク	いいえ
	内部リセット信号のマスク	いいえ
	~ メモリー	
	メモリー書き込み時のベリファイ	
	実行を一瞬停止してメモリアクセスする	
	✓ Start / Stop 機能設定	N

図 8-45 デバック・ツール設定

 "Startup"タブを選択します。"ブレークポイント設定先:"と"再開"のチェックボックスが以下のように チェックされていることを確認し、[適用]をクリックします。 これでチューニングのためのデバック構成の設定は完了です。

● デバッグ構成	– 🗆 X
構成の作成、管理、および実行	TO THE REAL PROPERTY OF THE PR
『 ♪ ♪ ♪ ♪ ♪ ♪ ♪ ♪ ♪ ♪ ↑     [7/ルタ入力     [7/ルタ入力     [7/ルタ入力     [7/ルタ入力     [7/ルタ入力     [7/ルタ入力     [7/リケーション     [7/w     [7/w	名前(N): Capacitive_Touch_Project_Example HardwareDebug メイン 参 Debugger Startup 一 共通 写 ソース 初期化コマンド リセットと遅延 (秒): Halt イメージとシンボルをロード
	ファイル名       □-ド・タイブ       オフセット       接続時         ✓ プログラム・パイナリ-[Cap       イメージとシンボル       0       Yes          「       「       「             「 <t< td=""></t<>
	ランタイム・オブション       〕 プログラム・カウンター設定先(16進):       ☑ ブレークボイント設定先:       図 ブレークボイント設定先:       図 再開
11 項目のうち 9 項目がフィルターに一致	前回保管Lた状態に戻す(V) 適用(V)
?	デパッグ(D) 閉じる

図 8-46 ランタイム・オプションの設定



8.5 QE for Capacitive Touch を使用した静電容量タッチセンサ・チューニング
 QE for Capacitive Touch を使用してプロジェクトをチューニングします。
 以下に手順を示します。

- "CapTouch メイン (QE)/QE V3.2.0 以降では[CapTouch ワークフロー (QE)]"の[調整を開始する]を クリックし、自動チューニングを開始します。
  - 【注意】 動作確認を簡単に行うために、このアプリケーション例では、ターゲットボードの電源は エミュレータの電源から供給します。PC の USB ポートから、E2 emulator Lite を経由して ターゲットボードに電源を供給することは可能ですが、ルネサスではターゲットボードで生成 する電源を使用してチューニングすることを推奨しています。

e –		×	
🏷 CapTouchワークフロー (QE) 🛛 🛛	ê 🕅		3
● ● 準備 調整 実装	動作確	認	•
2.タッチセンサの調整		-	
🥚 調整の実行 (エデュレータ	接続)		
調整を開始する ロアドバンスドモード(高度	まな設定	定)	
調整結果を確認する			

図 8-47 自動チューニングの開始

 デバッグセッションの開始時、e<sup>2</sup> studio はデバッグ・パースペクティブに切り替える旨のメッセージを 表示することがあります。[常にこの設定を使用する(R)]をチェックし、[切り替え]をクリックして デバッグセッションと QE for Capacitive Touch の自動チューニングを続行してください。

🖸 パーフ	ペクティブ切り替えの確認 ×			
?	This kind of launch is configured to open the デバッグ perspective when it suspends. このデバッグ・パースペクティブは、アプリケーションのデバッグをサポートするように設計されています。 これには、デバッグ・スタック、変数、およびブレークポイント管理を表示するビューが組み込まれてい ます。			
	Switch to this perspective?			
☑ 常にこの設定を使用する(R)				
	切り替え( <u>S</u> ) いいえ( <u>N</u> )			

図 8-48 デバック・パースペクティブの切り替え



 QE for Capacitive Touch の自動チューニングが開始されます。チューニングプロセスをガイドする "自動調整処理中"ダイアログを適宜、確認してください。表示例を以下に示します。通常、初期のチュー ニングプロセス中は操作を必要としません。

■ 自動調整処理中 ×	ĸ
1/13: 調整処理を開始するための準備中です。 評価ボードは絶縁物などの上に置いてください。鉄板などの上に直接置くと正しく計測できません。調整中は、指 示があるまでターゲットボード上のタッチセンサに触れないでください。	
キャンセル ヘルプ(H)	]

いくつかの工程を経て、以下のようなダイアログが表示されます。ここではチューニングプロセスにお けるタッチ感度の計測をします。ダイアログで表示されているセンサ (Mec00,TS00) を通常の圧力で タッチします。センサに触れているとき、バーグラフは右に増加し、数値で示すタッチカウント値が増 えます。センサに触れたまま、PC のキーボードのいずれかのキーを押して計測を確定します。

2 自動調整処理中 X
;/13: ボタン(Mec00, TS00 @ config01)の変化量を計測します。 アーゲットボード上のボタンを指で触れながら、キーボードで何かキーを押してください。キーを押すタイミングの目安は、 牧値の変動が安定した時点となります。
Mec00, TS00 @ config01: 11634
キャンセル ヘルプ(H)

図 8-50 "自動調整処理中"ダイアログ (タッチ感度の計測中)

設定したすべてのボタンに対して、前の手順を繰り返します。



図 8-49 "自動調整処理中"ダイアログ (初期チューニングプロセス中)

4. チューニングが完了すると、以下のようなダイアログが表示されしきい値を確認できます。このしきい 値はミドルウェアでタッチのイベント判定に使用されます。

④ 自動調整処理中										
感度調整で警告やオーバーフロー等のエラーが検出されている場合、対象とするタッチセンサを選択してリトライ(再 調整)してください。問題がなければ、「調整処理の継続」ボタンを押して処理を継続してください。										
リトライ対象の選択	メソッド	種別	名前	タッチセンサ	しきい値	オーバーフロー	警告/I7-			
	config01	ボタン	Mec00	TS00	1113					
	config02	ボタン	TS_B1	TS28	1125					
	config02	ボタン	TS_B2	TS18	1131					
	config02	ボタン	TS_B3	TS00	1270					
	config02	ホイール	Wheel	TS19, TS08, TS24, TS21	635					
	config02	スライダ	Slider	TS04, TS05, TS06, TS07, TS01	896					
リトライ 調整処理の継続										
							+	1)/7I	A 11-7/14	
							+	V/UN	TOD/(H	)

図 8-51 調整結果の表示

5. 表示されたダイアログの[**調整処理の継続**]をクリックします。これでチューニングプロセスは終了し、 ターゲットボードとのデバッグセッションを切断します。"CapTouch メイン (QE)/QE V3.2.0 以降で は[CapTouch ワークフロー (QE)]"に戻ります。



図 8-52 調整処理の継続

6. チューニングされたパラメータファイルの出力を行います。以下のように外部トリガの設定を行い、 [ファイルを出力する]をクリックします。

•			-	- 🗆	×
CapTouchワークフロー (QE) ×				i 🏠	
▲ ② 準備	調整	実装	動作確認		
1.静電容量タッチの準備 ▼	~	調整結果の出た	<b>b</b>		Î
✓ プロジェクトの選択					
⊘ タッチインタフェース構成の選択	調整結果からパラメータフ	アイルを出力します。			- 1
2.タッチセンサの調整 -	(	2 ファイルを出力す	3		- 1
⊘ 調整の実行 (エミュレータ接続)	□出力フォルダを指定する				
✓ 調整の実行 (シリアル接続)			(1) ■ 外部トリガを	使用する	ר
● 調整結果の出力			□診断コードを	使用する	-
3.プログラムの実装 -	]		□ API互換モート	ヾを使用す	3
● サンプルコードの実装					
4.動作確認	出力したファイルは静電容量タッチミドルウェアで利用されます。				
モニタリングの開始 (エミュレータ接続)	調整を実施した場合、	タッチインタフェース構成を変勢	更した場合は、必ずファイルを	出力して	
モニタリングの開始 (シリアル接続)	くたさい。				• •

図 8-53 パラメータファイルの出力



 プロジェクト・エクスプローラー"ウィンドウで qe\_touch\_config.c と qe\_touch\_config.h、 qe\_touch\_define.h が追加されたことを確認できます。
 これらのファイルにはドライバを使用したタッチ検出を有効にするために必要なチューニング情報が 含まれています。



図 8-54 チューニング・パラメータのファイル出力

8. e<sup>2</sup> studio の左上の <sup>S</sup> アイコンをクリックしてプロジェクトをビルドします。"コンソール"ウィンドウ で、ビルドした結果にエラーがないことが確認できます。 これで QE for Capacitive Touch を使用した静電容量タッチセンサ・チューニングは完了です。



- 8.6 アプリケーションに rm\_touch ミドルウェアの API コールを追加 rm\_touch ミドルウェアの API コールをプロジェクトに追加し、静電容量タッチ制御を有効にします。 以下に手順を示します。
- 1. タッチセンサの状態をスキャンするプログラムを実装するために、"CapTouch メイン (QE)/QE V3.2.0 以降では[CapTouch ワークフロー (QE)]"の[例を表示する]をクリックします。

•			×
ఏ CapTouchワークフロー (QE) 🗙		r 🖏	
▲ 🖉 準備 🖉 調整	実装	動作確	122 1
3.プログラムの実装			•
🛑 サンプルコードの	実装		
例を表示	する		

図 8-55 サンプルコード例の表示

2. "サンプルコードの表示"ウィンドウが開き、サンプルコードが表示されます。サンプルコードを出力するために[**ファイルに出力**]をクリックしてください。

🐻 サンプルコードの表示		×
main()関数のコード例:		
/**************************************	**********	^
* FILE : qe_sample_main.c * DESCRIPTION : Main Program for R *	IL78	
* NOTE:THIS IS A TYPICAL EXAMPLE.		
*****	**********	
#include "qe_touch_config.h"		
/* TODO Support for SMS function /* include related modules. * /* Please modify the below example //#include "Config_ITL000.h" //#include "Config_INTC.h" //#include "r_cg_itl_common.h"	*/ / code. */	
<pre>#define TOUCH_SCAN_INTERVAL_EX void R_CTSU_PinSetInit(void); void qe_touch_main(void); void qe_touch_delay(uint16_t delay_ void qe_sms_init(void);</pre>	us);	nds */
クリップボードにつビー	ファイルに出力	アプリケーションノートの表示
	// ///EM/J	7777 7377 10/94/3
		OK ヘルプ(H)

図 8-56 サンプルコードのファイル出力



3. "プロジェクト・エクスプローラー"より"qe\_touch\_sample.c"ファイルが生成されたことを確認してください。



図 8-57 QE 出力コード (qe\_touch\_sample.c) の生成

4. "Capacitive\_Touch\_Project\_Example.c"ファイルを開きます。



図 8-58 Capacitive\_Touch\_Project\_Example.c の選択



5. main()関数から qe\_touch\_main()関数をコールします。"Capacitive\_Touch\_Project\_Example.c"ファイ ルヘ下記画像のように赤枠部分のコード ("void qe\_touch\_main(void);"および"qe\_touch\_main();") を 追加してください。



図 8-59 qe\_touch\_main()関数の呼び出し設定

QE 出力コード(qe\_touch\_sample.c)にインクルードファイルの設定を行います。
 以下のとおりコメント設定を変更し、必要なヘッダファイルをインクルードしてください。

【注】RL78/G23 の場合、「#include "Config\_INTC.h"」はコメント状態のままにしておいてください。



図 8-60 QE 出力コード (qe\_touch\_sample.c) のインクルードファイル設定



 qe\_touch\_sample.c ファイルに Config2 を対象としたイニシャルオフセットチューニング処理を追加 します。Config01 のコードをコピーして以下の赤枠のようにコードを追加し、赤線の箇所を Config02 用 のコードに変更してください。

```
\fbox{c} qe_touch_sample.c 	imes
  68
           /* Initial offset tuning for [CONFIG01] */
   69
           {
               err = RM_TOUCH_ScanStart(g_qe_touch_instance_config01.p_ctrl);
   70
   710
               if (FSP_SUCCESS != err)
   72
               {
  73
74
                   while (true) {}
               }
   75
   76
               /* Clear the flag indicating that CTSU processing is complete. */
   77
              g_qe_touch_flag = 0U;
   78
   798
               while (true)
   80
               {
  81
                   qe_sms_trigger_start();
  82
                  while (0U == g_qe_touch_flag) {}
   83
   84
                  g_qe_touch_flag = 0U;
  85
  86
                  err = RM_TOUCH_DataGet(g_qe_touch_instance_config01.p_ctrl, &button_status, NULL, NULL);
   879
                  if (FSP_SUCCESS == err)
   88
                  {
  89
                       qe_sms_trigger_stop();
   90
                       break;
   91
                  }
  92
              }
  93
          }
  94
   95
   96
           /* Initial offset tuning for [CONFIG02] */
  97
           {
  98
               err = RM_TOUCH_ScanStart(g_qe_touch_instance_config02.p_ctrl);
   998
               if (FSP_SUCCESS != err)
  100
               {
  101
                   while (true) {}
  102
               }
  103
  104
               /* Clear the flag indicating that CTSU processing is complete. */
  105
               g_qe_touch_flag = 0U;
  106
  1070
               while (true)
  108
               {
  109
                   qe_sms_trigger_start();
  110
  111
                   while (0U == g_qe_touch_flag) {}
  112
                  g_qe_touch_flag = 0U;
 114
                   err = RM_TOUCH_DataGet(g_qe_touch_instance_config02.p_ctrl, &button_status, slider_position, wheel_position);
  1150
                   if (FSP_SUCCESS == err)
  116
                   {
                       qe_sms_trigger_stop();
  118
                       break:
  119
                  }
  120
               }
 122
  123
           /* Main loop */
  1249
           while (true)
  125
```



8. qe\_touch\_sample.c ファイルに Config02 計測用の外部トリガの設定を追加します。



図 8-62 Config02 計測用外部トリガの追加



9. qe\_touch\_sample.c ファイル内の qe\_sms\_init() 関数の設定をします。 コメント設定を以下のように変更します。

🖻 qe_touch_sample.c 🗡	- 0
212 /* SMS support initialize function */	2
21 void qe_sms_init(void)	
214 {	
215 /* TODO Support for lowering power consumption when using SMS function */	
216 /* Connection of unused pins for P123 and P124 */	
217 /* The settings for the CMC register are described in the "mcu_clocks.c" file */	
218 /* Please modify the below example code. */	
219 CSC  = 0x400;	
220 /* TODO Support for lowering novem consumption when using SMS function */	
(* Discable interpoint of "weite power consumption when using and intertuing periods for each channel" */	
/* Please modify the help wanter code	
$\chi$ M(2) = $\theta \times d\theta$ :	
224	
226 /* TODD Support for lowering power consumption when using SMS function */	
227 /* Disable interrupt servicing of "measurement data transfer request interrupt" */	
<pre>228 /* Please modify the below example code. */</pre>	
229 MK3L  = 0x01U;	
236	
231 /* TODO Support for SMS function (RL78/G22 only) */	
232 /* Select the event source for the CTSU in the ELC */	
23 /* As an example, this is the code when using an ELC. Please modify the below code. */	
234 ELSELRIO = 0x060; /* ELCIILO (32bit interval timerO) -> CISU */	
231 236 /* TOPO Support for SMS function (DL79/C22 only) */	
231 /* Salect the event source for the CTSU in the ELCL */	
/* As an example this is the code when using an FLCL Please modify the helow code */	
//FLIST = 0x171: //FLISTS is one code with using an electric fields modify the below code. /	
//ELLISEL0 = 0x080; /* Event link L1 signal select registers 0 -> Signal selected by ELISEL7 is to be linked */	
241 //ELL1LNK0 = 0x01U; /* Event link L1 output select registers 0 -> Linked to input 0 of logic cell 0 in logic cell block L1	*/ 📫 注
24.2 //ELOSEL6 = 0x010; /* Output signal select registers 6 -> Output signal 0 from logic cell block L1 (output of logic cell0)	) is selected */
24: //ELOENCTL = 0x40U; /* Output signal enable register -> Enable the output of the signals selected by the ELOSEL6 registers	*/
244	
5	>

図 8-63 qe\_sms\_init 関数の設定

【注】 1. RL78/G23 の場合は、「ELSELR10 = 0x06U;」はコメント状態のままにしてください。

2. RL78/G23 の場合は、下記コードのコメントアウトを解除してください。 ELISEL7 = 0x17U; ELL1SEL0 = 0x08U; ELL1LNK0 = 0x01U; ELOSEL6 = 0x01U; ELOENCTL = 0x40U;



10. qe\_touch\_sample.c ファイル内の、qe\_sms\_trigger\_start() 関数と qe\_sms\_trigger\_stop() 関数の設定を します。

コメント設定を以下のように変更します。



図 8-64 qe\_sms\_trigger\_start 関数と qe\_sms\_trigger\_stop 関数の設定

【注】1. RL78/G23の場合、「R\_Config\_INTC\_INTP5\_Start();」はコメント状態のままにしておいてください。 2. RL78/G23の場合、「R\_Config\_INTC\_INTP5\_Stop();」はコメント状態のままにしておいてください。

11.e<sup>2</sup> studio 左上の <sup>S</sup> アイコンをクリックしてプロジェクトのビルドを開始します。 "コンソール"ウィンドウで、ビルドした結果にエラーがないことが確認できます。 SMS を使用した自動判定計測を活用した、静電容量タッチ低消費電力アプリケーションの開発に必要な 手順は以上となります。



9. 参考ドキュメント

RL78/Gxx ユーザーズマニュアル ハードウェア編 RL78 ファミリ ユーザーズマニュアル ソフトウェア編 (R01US0015) (最新版をルネサス エレクトロニクスホームページから入手してください。)

テクニカルアップデート/テクニカルニュース (最新の情報をルネサス エレクトロニクスホームページから入手してください。)

アプリケーションノート RL78 ファミリ 静電容量センサユニット(CTSU2L) 動作説明 (R01AN5744) アプリケーションノート RL78 ファミリ CTSU モジュール Software Integration System (R11AN0484) アプリケーションノート RL78 ファミリ TOUCH モジュール Software Integration System (R11AN0485) アプリケーションノート 静電容量センサマイコン 静電容量タッチ電極デザインガイド (R30AN0389) アプリケーションノート RL78/G23 静電容量タッチ低消費電力ガイド(SMS 機能) (R01AN6670) アプリケーションノート RL78/G22 静電容量タッチ低消費電力ガイド(SMS/MEC 機能) (R01AN6847) (最新版をルネサス エレクトロニクスホームページから入手してください。)

## ホームページとサポート窓口

ルネサス エレクトロニクスホームページ

http://www.renesas.com/

静電容量センサユニット関連ページ <u>https://www.renesas.com/solutions/touch-key</u> <u>https://www.renesas.com/ge-capacitive-touch</u>

お問い合わせ http://www.renesas.com/contact/



## 改訂記録

		改訂内容		
Rev.	発行日	ページ	ポイント	
1.00	2024.07.22	-	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. 静電気対策

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

2. 電源投入時の処置

電源投入時は、製品の状態は不定です。電源投入時には、LSIの内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部 リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオン リセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。

3. 電源オフ時における入力信号

当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入に より、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」について の記載のある製品は、その内容を守ってください。

4. 未使用端子の処理

未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識 されて誤動作を起こす恐れがあります。

5. クロックについて

リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した 後に切り替えてください。リセット時、外部発振子(または外部発振回路)を用いたクロックで動作を開始するシステムでは、クロックが十分安定 した後、リセットを解除してください。また、プログラムの途中で外部発振子(または外部発振回路)を用いたクロックに切り替える場合は、切り 替え先のクロックが十分安定してから切り替えてください。

#### 6. 入力端子の印加波形

入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V<sub>IL</sub>(Max.)からV<sub>IH</sub>(Min.)までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V<sub>IL</sub>(Max.)からV<sub>IH</sub>(Min.)までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。

# リザーブアドレス(予約領域)のアクセス禁止 リザーブアドレス(予約領域)のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス(予約領域)があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。

8. 製品間の相違について

型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合が あります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

## ご注意書き

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

標準水準 : コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等 高品質水準:輸送機器(自動車、電車、船舶等)、交通制御(信号)、大規模通信機器、金融端末基幹システム、各種安全制御装置等 当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のあ る機器・システム(生命維持装置、人体に埋め込み使用するもの等)、もしくは多大な物的損害を発生させるおそれのある機器・システム(宇宙機 器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等)に使用されることを意図しておらず、これら の用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その 青仟を負いません。

- 7 あらゆる半導体製品は、外部攻撃からの安全性を100%保証されているわけではありません。当社ハードウェア/ソフトウェア製品にはセキュリ ティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害(当社製品または当社製品が使用されてい るシステムに対する不正アクセス・不正使用を含みますが、これに限りません。)から生じる責任を負うものではありません。当社は、当社製品ま たは当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行 為(「脆弱性問題」といいます。)によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害に ついて、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア/ソフトウェア製品について、商品 性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
- 8. 当社製品をご使用の際は、最新の製品情報(データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導 体デバイスの使用上の一般的な注意事項」等)をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の 範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切 その責任を負いません。
- 9. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする 場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を 行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客 様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を 行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行って ください。
- 10. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用 を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことに より生じた損害に関して、当社は、一切その責任を負いません。
- 11. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品お よび技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、そ れらの定めるところに従い必要な手続きを行ってください。
- 12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたしま ォ
- 13. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
- 14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的 に支配する会社をいいます。

注2. 本資料において使用されている「当社製品」とは、注1において定義された当社の開発、製造製品をいいます。

(Rev.5.0-1 2020.10)

#### 本社所在地

〒135-0061 東京都江東区豊洲 3-2-24 (豊洲フォレシア) www.renesas.com

## お問合せ窓口

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

#### 商標について

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