

SmartRA 学習キット チュートリアル 8

ルネサス エレクトロニクス社 RA マイコン搭載 HSB シリーズマイコンボード 評価キット

-本書を必ずよく読み、ご理解された上でご利用ください





注意事項	1
安全上のご注意	2
1. RA2L1_TOUCHKEY_QE	4
1.1. QE ツールのインストール	4
1.2. エミュレータ(デバッガ)の接続設定	6
1.3. FSP の設定	9
1.4. タッチキー動作を確認するためのボード設定	
1.5. QE for CapTouch の使用	
1.5.1. プロジェクトの作成	
1.5.2. チューニング	
1.5.3. プログラムコードへの反映	
1.5.4. モニタリング	
1.5.5. COM ポートでのモニタリング	
2. RA2L1 CTSU	
	22
2.1. フロップムの動作	
2.2. タリナ刊との原理	
2.3. ダッナキー処理の流れ	
2.6. CTSU 関連のクローハル変数	
2.7. CTSU 関連の定数値	
2.8. まとめ	
取扱説明書改定記録	
お問合せ窓口	





注意事項

本書を必ずよく読み、ご理解された上でご利用ください

【ご利用にあたって】

- 本製品をご利用になる前には必ず取扱説明書をよく読んで下さい。また、本書は必ず保管し、使用上不明な点がある場合は再読し、よく理解して使用して下さい。
- 2. 本書は株式会社北斗電子製マイコンボードの使用方法について説明するものであり、ユーザシステムは対象ではありません。
- 3. 本書及び製品は著作権及び工業所有権によって保護されており、全ての権利は弊社に帰属します。本書の無断複 写・複製・転載はできません。
- 弊社のマイコンボードの仕様は全て使用しているマイコンの仕様に準じております。マイコンの仕様に関しましては 製造元にお問い合わせ下さい。弊社製品のデザイン・機能・仕様は性能や安全性の向上を目的に、予告無しに変更 することがあります。また価格を変更する場合や本書の図は実物と異なる場合もありますので、御了承下さい。
- 5. 本製品のご使用にあたっては、十分に評価の上ご使用下さい。
- 6. 未実装の部品に関してはサポート対象外です。お客様の責任においてご使用下さい。

【限定保証】

- 1. 弊社は本製品が頒布されているご利用条件に従って製造されたもので、本書に記載された動作を保証致します。
- 2. 本製品の保証期間は購入戴いた日から1年間です。

【保証規定】

保証期間内でも次のような場合は保証対象外となり有料修理となります

- 1. 火災・地震・第三者による行為その他の事故により本製品に不具合が生じた場合
- 2. お客様の故意・過失・誤用・異常な条件でのご利用で本製品に不具合が生じた場合
- 3. 本製品及び付属品のご利用方法に起因した損害が発生した場合
- 4. お客様によって本製品及び付属品へ改造・修理がなされた場合

【免責事項】

弊社は特定の目的・用途に関する保証や特許権侵害に対する保証等、本保証条件以外のものは明示・黙示に拘わらず 一切の保証は致し兼ねます。また、直接的・間接的損害金もしくは欠陥製品や製品の使用方法に起因する損失金・費用 には一切責任を負いません。損害の発生についてあらかじめ知らされていた場合でも保証は致し兼ねます。 ただし、明示的に保証責任または担保責任を負う場合でも、その理由のいかんを問わず、累積的な損害賠償責任は、弊 社が受領した対価を上限とします。本製品は「現状」で販売されているものであり、使用に際してはお客様がその結果に 一切の責任を負うものとします。弊社は使用または使用不能から生ずる損害に関して一切責任を負いません。 保証は最初の購入者であるお客様ご本人にのみ適用され、お客様が転売された第三者には適用されません。よって転 売による第三者またはその為になすお客様からのいかなる請求についても責任を負いません。 本製品を使った二次製品の保証は致し兼ねます。





製品を安全にお使いいただくための項目を次のように記載しています。絵表示の意味をよく理解した上で お読み下さい。

表記の意味



取扱を誤った場合、人が死亡または重傷を負う危険が切迫して生じる可能性が ある事が想定される

取扱を誤った場合、人が軽傷を負う可能性又は、物的損害のみを引き起こすが 可能性がある事が想定される

絵記号の意味

0	一般指示 使用者に対して指示に基づく行為を 強制するものを示します	\bigcirc	一般禁止 一般的な禁止事項を示します
	電源プラグを抜く 使用者に対して電源プラグをコンセ ントから抜くように指示します		一般注意 一般的な注意を示しています











SmartRA 学習キット チュートリアル 8





1. RA2L1_TOUCHKEY_QE

本チュートリアルでは、RA2L1の持つタッチキー機能(CTSU2)に関して触れます。

CTSU は、Capacitive Sensing Unit の略で静電容量を使用して、人の指でタッチしたかどうかを識別する機能です。非接触スイッチ等で使用されています。

ルネサスの CTSU は、R8C/33T でマイコンの機能として実装され、RX113/RX130/RX231 で 2 世代目に進化して います。RA シリーズのマイコンは基本的には、CTSU を持っているのですが、RA2L1 ではさらに進化した CTSU2 が搭載されています。

本キットのベースボードには、タッチキーパッドを3つ持たせていますので、このパッドを用いたチュートリアルとなります。

1.1. QE ツールのインストール

タッチキーは、Renesas QE Cap-touch でサポートされる機能となりますので、e2studio に、Renesas QE Cap-touch をインストールしてください。

※[重要] Renesas QE Cap-touch を使って PC からタッチキー動作を見る場合は、エミュレータ(ルネサス E2Lite も しくは E2)が必要です(エミュレータがない場合は、1.5.5 に進み、タッチ動作を確認することができます)

_∧JL	プ(H)	
3	ようこそ(W)	
? ??	ヘルプ目次(H) 検索(E) ダイナミック・ヘルプ(D)	
	キー・アシスト(K) 虎の巻の表示(C)	Ctrl+シフト+L
-	RA Helpdesk	
R	RenesasRulz コミュニティー・フォーラム	1
Ø	Renesas ツールチェーンの追加	
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~	Perform Setup Tasks	
<i>e</i>	更新の検査	
<b>6</b> 3.	新規ソフトウェアのインストール	
	Renesas e2 studio feedback	
G	IAR Embedded Workbench plugin manager	
0	e² studio について(A)	

ヘルプー新規ソフトウェアのインストールから、





1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2			— 🗆 X
利用できるソフトウェア			
インストールしたい項目をチェック			()
作業対象(W): QE Common Update Site - http://tool-support.renesas.com/	e2studio/qe/qe-common/	~ 追加( <u>A</u> )…	管理( <u>M</u> )
7-гиядл			すべて選択( <u>S</u> )
名前	パージョン		選択をすべて解除( <u>D</u> )
V III Renesas QE			
We Renesas QE common     Renesas QE common     Renesas QE for Connecting Towney (RA RL 79)	1.7.0.v20191204-1122		
V igp= kenesas QE for Capacitive Iouch[KA, KL76]	1.5.0.720210216-1154		
2 項目が避けたかました			
2項目が選択されました。			
2 項目が選択されました。 詳細			
2 項目が選択されました。 - 詳細			Â
2 項目が選択されました。 詳細	□ 既にインフトールネれた酒日を隠す(日)		Å
2 項目が選択されました。 - 詳細 - ジョンだけを表示(L) - ジョンだけを表示(L)	□ 既にインストールされた項目を隠す( <u>H</u> )		Å
2 項目が選択されました。 詳細 ②利用できるソフトウェアの最新パージョンだけを表示(L) ②カテゴリーで項目を分類(G)	□ 既にインストールされた項目を隠す( <u>H</u> ) <u>すでにインストール済み</u> なのは何?		ĉ
2 項目が選択されました。 詳細 ☑ 利用できるソフトウェアの最新パージョンだけを表示(L) ☑ カテゴリーで項目を分類(G) □ ターゲット環境に適用できるソフトウェアのみ表示	□既にインストールされた項目を隠す( <u>H</u> ) <u>すでにインストール済み</u> なのは何?		ĉ
2 項目が選択されました。 詳細 ☑ 利用できるソフトウェアの最新パージョンだけを表示(L) ☑ カテゴリーで項目を分類(G) □ ターゲット環境に適用できるソフトウェアのみ表示 ☑ 必要なソフトウェアを見つけるために、インストール中に更新サイト全てに接続(C)	□ 既にインストールされた項目を隠す( <u>H</u> ) <u>すでにインストール済み</u> なのは何?		Å
2 項目が選択されました。 詳細 ☑ 利用できるソフトウェアの最新バージョンだけを表示(L) ☑ カテゴリーで項目を分類(G) □ ターゲット環境に適用できるソフトウェアのみ表示 ☑ 必要なソフトウェアを見つけるために、インストール中に更新サイト全てに接続( <u>C</u> )	□ 既にインストールされた項目を隠す( <u>H</u> ) <u>すでにインストール済み</u> なのは何?		Å
2 項目が選択されました。 詳細 ②利用できるソフトウェアの最新パージョンだけを表示(L) ③カテゴリーで項目を分類(G) ③ターゲット環境に適用できるソフトウェアのみ表示 ②必要なソフトウェアを見つけるために、インストール中に更新サイト全てに接続(C)	□既にインストールされた項目を隠す( <u>H</u> ) <u>すでにインストール済み</u> なのは何?		ĉ
2 項目が選択されました。 詳細 ジ 利用できるソフトウェアの最新パージョンだけを表示(L) ジ カテゴリーで項目を分類(G) ロターゲット環境に適用できるソフトウェアのみ表示 ジ 必要なソフトウェアを見つけるために、インストール中に更新サイト全てに接続(C)	□ 既にインストールされた項目を隠す( <u>H</u> ) <u>すでにインストール済み</u> なのは何 ?		
2 項目が選択されました。 詳細 ☑ 利用できるソフトウェアの最新パージョンだけを表示(L) ☑ カテゴリーで項目を分類(G) □ ターゲット環境に適用できるソフトウェアのみ表示 ☑ 必要なソフトウェアを見つけるために、インストール中に更新サイト全てに接続( <u>C</u> )	□ 既にインストールされた項目を隠す( <u>H</u> ) <u>すでにインストール済み</u> なのは何?		Å
2 項目が選択されました。 詳細 ② 利用できるソフトウェアの最新パージョンだけを表示(L) ③ カテゴリーで項目を分類(G) ③ ターゲット環境に適用できるソフトウェアのみ表示 ② 必要なソフトウェアを見つけるために、インストール中に更新サイト全てに接続(C)	□ 既にインストールされた項目を隠す(止) すでにインストール済みなのは何?	40.7 m	A contrall

QE Common Update Site を選択して、

Renesas QE common

Resnesas QE for Capacitive Touch

をインストールしてください。

(インターネットに直接接続していない場合は、ルネサスエレクトロニクスのサイトから zip ファイルをダウンロードして、ローカルファイルからインストールしてください)



## 1.2. エミュレータ(デバッガ)の接続設定

プロジェクトエクスプローラーから、対象プロジェクトを右クリック。





💽 デバッグ構成	— D X
構成の作成、管理、および実行	To the second se
<ul> <li>C/C++ アプリケーション</li> <li>C/C++ アプリケーション</li> <li>C/C++ リモート・アプリケーション</li> <li>EASE Script</li> <li>GDB OpenOCD Debugging</li> <li>GDB Simulator Debugging (RH850)</li> <li>GDB /トドウェア・デバッギング</li> <li>Java アプリケーション</li> <li>Java アプリケーション</li> <li>Java アプリケーション</li> <li>Facesas GDB Hardware Debugging</li> <li>Renesas GDB Hardware Debugging</li> <li>Renesas Simulator Debugging (RX, RL78)</li> <li>リモート Java アプリケーション</li> <li>起動グループ</li> </ul>	<ul> <li>名前(N): RA2L1_TOUCHKEY_QE Debug_Flat</li> <li>メイン 参 Debugger ▶ Startup ♥ ソース ■ 共通(C)</li> <li>プロジェクト(P):</li> <li>RA2L1_TOUCHKEY_QE</li> <li>Ø(B)</li> <li>C/C++ アブリケーション:</li> <li>Debug/RA2L1_TOUCHKEY_QE.elf</li> <li>変数(V)</li> <li>プロジェクトの検索(H)</li> <li>参照(B)</li> <li>起動前に必要に応じてビルド</li> <li>Build Configuration:</li> <li>Use Active</li> <li>〇 自動ビルドを有効にする</li> <li>〇 自動ビルドを有効にする</li> <li>〇 一クスペース設定の使用</li> </ul>
プロジェクト名 ここに複数の項目が見え ます。不要なプロジェクトを なるようにしてください。	ている時には、複数のプロジェクトが開いてい 閉じて、開いているプロジェクトが 1 つだけと 第回 尾音した状態に戻す(M) 適用(M)

Debugger タブを開く。





デバッグ構成 構成の作成、管理、および実行	
□       □       □       □       □       □       □         □       □       ○       ○       ○       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □       □ <th>名前(N): RA2L1_TOUCHKEY_QE Debug_Flat メイン 参 Debugger Startup ラ ソース 日共通(C) Debug hardware: E2 Lite (ARM) 「Target Device: R7FA2L1AB … GDR Settinger Comparing Control = Fifted AN USE In Later</th>	名前(N): RA2L1_TOUCHKEY_QE Debug_Flat メイン 参 Debugger Startup ラ ソース 日共通(C) Debug hardware: E2 Lite (ARM) 「Target Device: R7FA2L1AB … GDR Settinger Comparing Control = Fifted AN USE In Later
<ul> <li>C GDB Simulator Debugging (RH850)</li> <li>C GDB K→ドウェア・デバッギング</li> <li>Java アブリケーション</li> <li>Java アブリケーション</li> <li>Java アブレット</li> <li>C Renesas GDB Hardware Debugging</li> <li>C RA2L1_TOUCHKEY_QE Debug_Flat</li> </ul>	GDB Settings       Connection Settings       デバッグ・ツール設定       デバッグ方を選択する         GDB 接続設定:       ● ローカル GDB サーバーを自動起動       ホスト名または IP アドレス:       localhost         ○ リモート GDB サーバーへ接続       GDB ポート番号:       61234
ご Renesas Simulator Debugging (RX, RL78) 記。リモート Java アプリケーション 電 起動グループ	GDB GDB コマンド: arm-none-eabi-gdb 参照 変数 □ Step Mode
15 項目のうち 13 項目がフィルターに一致	前回保管した状態に戻す(⊻) 適用(⊻)
(?)	デパッグ( <u>D</u> ) 閉じる

Connection Settings タブを開く

📴 デバッグ構成		_	
構成の作成、管理、および実行			Ť.
<ul> <li>         「ご ご ご ご ご ご ご ご ご ご ご ご ご ご ご</li></ul>	名前(N): RA2L1_TOUCHKEY_QE Debug_Flat メイン      な Debugger     ト Startup      ソース      ・ 共通(Q) Debug hardware: E2 Lite (ARM)      Target Device: R7FA2L1AE GDB Settings Connection Settings      デパッグ・ツール設定 メイン・クロック メイン・クロック・ソース 外部クロック・ソース 内蔵フラッシュ・メモリー書き換え時にクロック・ソースの変更を許可す	3 … 内部クロック でる はい	~
<ul> <li>ご Renesas GDB Hardware Debugging</li> <li>ご RA2L1_TOUCHKEY_QE Debug_Flat</li> <li>ご Renesas Simulator Debugging (RX, RL78)</li> <li>!! リモート Java アブリケーション</li> <li>記動グルーブ</li> </ul>	<ul> <li>&gt; ターゲット・ボードとの接続 エミュレーター タイプ 接続速度 (kHz)</li> <li>&gt; 電源 エミュレーターから電源を供給する (MAX 200mA) 供給電圧 (V)</li> </ul>	(Auto) SWD Auto いいえ 3.3	···· · · · · · · · · · · · · · · · · ·
	<ul> <li>✓ 接続 接続時にリセット状態を維持する</li> <li>ID3-ド(バイト単位)</li> <li>前回保管し</li> </ul>	はい FFFFFFFFFFFFFFFFFF た状態に戻す( <u>V</u> )	、いいえを選 FFFF ···· ▼ 適用(Y)
15 項目のうち 13 項目がフィルターに一致		デパッグ( <u>D</u> )	閉じる





エミュレータから電源を供給するする:「いいえ」 を選択してください。

「適用」、「閉じる」で設定画面を閉じてください。

※エミュレータから電源を供給する事も可能ですが、その場合は ・<u>別な箇所(USB 等)から電源を供給しない</u> 様にしてください。

※なお、E2Lite をお使いの場合は、エミュレータから供給する電圧は、3.3V しか選択できません(E2 では、5V と 3.3V が選択可能です)

※「エミュレータ」と「デバッガ」、「Debugger」は同義です





## 1.3. FSP の設定



New Stack – Middleware – CapTouch – TOUCH Driver on rm_touch を追加。



Image: Image: Barrier (Barrier Content of the second s	Configuration	22				
Pin Configuration Generate Project Content						
Select Pin Configuration			Export	to CSV file 🛛 🖺 Co	nfigure Pin Driver	Warnings
R7FA2L1AB2DFP:pincfg		Manage configurations		ienerate data: g_t	bsp_pin_cfg	
Pin Selection	⊞ ⊟ ↓ <mark>a</mark>	Pin Configuration				😲 Cycle Pin Group
Type filter text		Name	Value	Lock	Link	^
Connectivity	•	CFCTS27	✓ P113		$\Rightarrow$	
Connectivity.Sci	~	CFCTS28	None		$\Rightarrow$	
y / Input/CTSU		CFCTS29	✓ P114	i i i i i i i i i i i i i i i i i i i	$\Rightarrow$	
<ul> <li>CTSU0</li> </ul>		CFCTS30	None		$\Rightarrow$	
> Input/CU		CFCTS31	None		$\Rightarrow$	
> Input/CO		CFCTS32	None		$\Rightarrow$	
Monitoring:CAC		CFCTS33	None		$\Rightarrow$	
Sustem:CGC		CFCTS34	None		$\Rightarrow$	
System DEBLIG		CFCTS35	🗸 P115	<b></b>	$\Rightarrow$	~
System:SYSTEM		<				>
Timers:AGT		Martula assess CTCUs				
> Timers:GPT		Module name: CISO0				
Timerc:RTC	~					
端子機能 端子番号						
Summer BCD Charles Directed		inte Charles Commente				
Summary   BSP   Clocks   Pins   Int	errupts   Event L	inks   Stacks   Components				



SmartRA 学習キット チュートリアル 8



#### Pins タブ、Peripherals – Input:CTSU – CTSU0

Operation Mode -> Enabled

	割り当てる端子	備考
TSCAP	P112	マイコンボード M-J16 をショートに設定
CFCTS27	P113	ベースボード B-K3
CFCTS29	P114	ベースボード B-K2
CFCTS35	P115	ベースボード B-K1

※割り当てる端子は、複数の中から選ぶわけではなく、有効にするか None かの選択です

IRA2L1_TOUCHKEY_QE] FSP Configuration ⋈					👩 Package 📑 MCUパッケージ 😒
Stacks Configuration				Generate Project Content	
Threads     New Thread     Remove       Image: Second state     Image: Second state     Image: Second state       Image: Second state     Image: Second state     Image: Second state       Image: Second state     Image: Second state     Image: Second state       Image: Second state     Image: Second state     Image: Second state       Image: Second state     Image: Second state     Image: Second state       Image: Second state     Image: Second state     Image: Second state       Image: Second state     Image: Second state     Image: Second state       Image: Second state     Image: Second state     Image: Second state       Image: Second state     Image: Second state     Image: Second state       Image: Second state     Image: Second state     Image: Second state       Image: Second state     Image: Second state     Image: Second state       Image: Second state     Image: Second state     Image: Second state       Image: Second state     Image: Second state     Image: Second state       Image: Second state     Image: Second state     Image: Second state       Image: Second state     Image: Second state     Image: Second state       Image: Second state     Image: Second state     Image: Second state       Image: Second state     Image: Second state     Image: Second state       Image: Seco	HAL/Common Stacks g_ioport I/O Port Driver on r_ioport	TOUCH Driver on rm_tout	New Stack >	Extend Stack > 🔬 Remove	
		CTSU Driver on r_ctsu	Add DTC Driver for Reception	Add SCI UART Driver for monitor of QE New >	UART Driver on r_sci_uart
Objects New Object > 1 Remove	<	[Recommended but optional]	[Recommended but optional]	>	
Summary  BSP   Clocks   Pins   Interrupts   Event Links   Stacks	Components				▶ 凡例

Stacks タブ、Add SCI UART Driver for monitor of QE に New – UART Driver on r_sci_uart を追加

SmartRA 学習キット チュートリアル 8 株式会社 北手電子



∰ *[RA2L1_TOUCHKEY_QE] FSP Configuration ⊠				
Stacks Configuration				Generate Project Content
Threads	HAL/Common S	itacks	🛃 New Stack > 🔮	Extend Stack > 📓 Remove
<ul> <li>✓</li></ul>	[}] TOUCH Driver	on rm_touch		
	CTSU Driver or	n r_ctsu	g_uart_qe UART Driver	I on r_sci_uart
Objects 🐑 New Object > 🔊 Remove	Add DTC Drive Transmission [Recommende optional]	r for d but Add DTC Driver for Reception [Recommended but optional]	Add DTC Driver for Transmission [Recommended but optional]	Add DTC Driver for Reception [Not recommended]
Summary BSP Clocks Pins Interrupts Event Links 😔 S	<	s		>
□ プロパティー 🛛 🔝 問題 🁒 スマート・ブラウザ				2 8 -
g_uart_qe UART Driver on r_sci_uart				
Settings プロパティ API Info DTC Support		値 Disable		^
RS232/RS485 Flow Control Support V Module g_uart_qe UART Driver on r_sci_u V General	uart	Disable		
Name		🔒 g_uart_qe		
Data Bits		Bbits		
Parity		🔒 None		
Stop Bits		🔒 1bit		
> Baud				
> Flow Control				¥

追加した、UARTの Channel を 9 に変更。チュートリアル 5 を参考に、端子設定(TX:P109, RX:P110)を割り当て。





*[RA2L1_TOUCHKEY_QE] FSP Configuration	1 🛙				
Pin Configuration					Generate Project Content
Select Pin Configuration		Export to CSV	file 🖺 Co	onfigure Pin Driver	Warnings
R7FA2L1AB2DFR.pincfg	Manage configurations	Generate	data: g_	bsp_pin_cfg	
Pin Selection 🕀 🕞 🕌	Pin Configuration				😲 Cycle Pin Group
Type filter text	Name	Value	Lock	Link	
	Pin Group Selection	Mixed			
Analog:ACMP A	Operation Mode	Asynchronous UART			
Analog ANALOG	✓ Input/Output			$\langle \rangle$	
Analog:DAC	TXD	✓ P109	- É	$\Rightarrow$	
Connectivity/CAN	RXD	🗸 P110	a di seconda di second	4	
Connectivity.ICAN	SCK	None		$\Rightarrow$	
Connectivity.iic	CTS	None		$\Rightarrow$	
SCI0	SDA	None		$\Rightarrow$	
SCI1	SCL	None		$\Rightarrow$	
SCI2	<				>
SCI3	Madula name: SCI0				^
✓ SCI9	Module name: SCI9	1.100			
Connectivity/SDI	Usage: When using Sil	nple I2C mode, ensure port pin: a between I2C and other mode	s output ty first disat	pe is n-ch open dra	an. 🗸
端子機能端子番号					
Summary BSP Clocks Pins Interrupts Event	Links 🙆 Stacks Components				

戀 *[RA2L									
Stacks (	itacks Configuration Generate Project Content								
Threads	🐔 New Thread 💼 Remove 📄	HAL/Common	Stacks	🗿 New Stack > 🔮 Exter	nd Stack > 📓 Remove				
Objects	AL/Common g_ioport I/O Port Driver on r_ioport TOUCH Driver on rm_touch	TOUCH  CTSU Dr  CTSU Dr  CTSU Dr  Cases  Ca	Driver on rm_touch iver on r_ttsu	<ul> <li>              g_uart_qe UART Driver o             í             í</li></ul>	Add DTC Driver for Reception [Not recommended]				
Summary	BSP Clocks Pins Interrupts Event Links 🐼 Sta	acks Component	ts						
TOUCH	Driver on rm_touch								
Settings API Info	プロパティ ✓ Common		値						
	Parameter Checking Support for QE monitoring using UART	Г	Default (BSP) Enabled						

Support for QE monitoring using UART を Enabled に変更。(この変更で、Stacks のエラー(×印)が消えるはず です。)





Stacks Configuration				Generate Project Content
Threads       New Thread       Remove         HAL/Common       g_ioport I/O Port Driver on r_ioport         TOUCH Driver on rm_touch         TOUCH Driver on rm_touch	HAL/Common Stacks g_ioport I/O Port Driver on r_ioport (1)	<ul> <li>TOUCH Driver on rm_to</li> <li>CTSU Driver on r_ctsu</li> <li>Add DTC Driver for Transmission [Recommended but optional]</li> </ul>	New Stack > Extended buch Add DTC Driver for Reception [Recommended but optional]	Add DTC Driver for Transmission [Recommended but optional]
	<			>
Summary   BSP   Clocks   Pins   Interrupts   Event Links   Stack	s Components			

一通り設定が終われば、Generate Project Content で、プロジェクトファイルー式を出力。





1.4. タッチキー動作を確認するためのボード設定



マイコンボードとPCを接続してください。ジャンパは上記設定とする。

SmartRA 学習キット チュートリアル 8 株式会社



## 1.5. QE for CapTouch の使用

#### 1.5.1. プロジェクトの作成



Renesas Views - Renesas QE – CapTouch メイン/センサ・チューナを起動。



プロジェクトの選択

作成したプロジェクト(ここでは、RA2L1_TOUCHKEY_QE プロジェクト)を選択。







#### 構成の選択 – タッチインタフェース構成の新規作成

静電容量方式 – 自己容量を選択。ボタンを押す。





7ッチインタフェー) 1989-	ス構成のファイル名:	RA2L1_TOUCHKEY_QE	構成(メソッド)の設定	構成の流用/再編集
лчл;				
				静電容量方式
	Button01 Button02	2 Button03		自己容量
				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
				 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
				タッチI/Fの自ll空
設定				
タッチ1/	Fの設定	総抵抗の設定割り付けTSxの解	除	
3 設定内容に	問題があります。			

基板に合わせて、3つのボタンを配置。

Button01 をダブルクリック。

📴 97	チインタフェースの設定		×	📴 97	チインタフェースの認	定		×
	ボタン(自己)				ボタン(自己)			
	名前	Button01			名前	K1		
	タッチセンサ	抵抗値[Ω]			タッチセンサ		抵抗値[Ω]	
	TSOO	560			TS35	~	560	~
	ОК	キャンセル	ヘルプ( <u>H</u> )		OK	キャン	'ชม	ヘルプ( <u>H</u> )

名前を K1、タッチセンサを TS35 に変更。560 Ωはそのままで良いです。



SmartRA 学習キット チュートリアル 8 株式会社 北手電子



タッチインタフ:	エ−ス構成の作♬	戓			,
タッチインタフェー	-ス構成のファイノ	し名:	RA2L1_TOUCHKEY_QE	構成(メソッド)の設定	構成の流用/再編集
党明:					
					- タッチI/F 静電容量方式
	K1	K2	K3		自己容量 🗸
	T\$35	002T	TS27		ボタン
	1555	1525			スライダ(横方向)
					スライダ(縦方向)
					ホイール
					<b>キ</b> ーパッド
					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
					タッチI/Fの削除
設定					
タッチし	/Fの設定	総	抵抗の設定 割り付けTS×の解	除	
					/+-++/>> ↓->→    ↓    ↓    ↓    ↓    ↓    ↓    ↓
					1FRX(L) キャノセル ハルノ( <u>H</u> )

同様に、K2-TS29, K3-TS27を設定してください。作成を押す。

## 1.5.2. チューニング

<ul> <li>準備</li> <li>プロジェクトの準備 タッチインタフェースを使用するプロ ジェクトを準備します。</li> <li>プロジェクトの選択</li> <li>RA2L1_TOUCHKEY_QE</li> <li>構成の選択 タッチインタフェース構成を選択/ 作成します。</li> <li>RA2L1_TOUCHKEY_QE.tifcft 、</li> </ul>	<ol> <li>2.調整</li> <li>各タッチセンサについて自動調整処理を 行います。</li> <li>ターゲットボードの接続 エミュレータ経由でターゲットボード とPCを接続します。</li> <li>チューニングの開始 ダイアログの指示に従い操作します。</li> <li>チューニングを開始する</li> <li>「高度な設定を有効にする</li> <li>「酸整結果の出力 調整結果からパラメータファイルを</li> </ol>	<ul> <li>3.実装</li> <li>タッチインタ フェースを使用したプログラムを実 装してください。</li> <li>実装例の表示 main()間数に タッチセンサの 状態を定期的に スキャンするプログラムを実装 します。</li> </ul>	<ul> <li>4.動作確認</li> <li>タッチインタフェースの動作確認と微調整を行えます。</li> <li>デバッグの開始 対象プロジェクトのデバッグを開始し、 プログラムを実行します。</li> <li>モニタリングの開始 モニタリング開ビューを表示し、モニ タリング機能を有効にします。</li> <li>ビューを開く</li> <li>シリアル優越 シリアル優越</li> </ul>
構成を編集する	のう 調整結果からパラメータファイルを 生成します。		<ul> <li>シリアル通信によるモニタリング機能 を有効にします。</li> <li>自動</li> </ul>

チューニングを開始する。





◙ 自動調整処理中	×
1/8: 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 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	直接置くと正しく測定できませ ご触れないでください。
	キャンセル ヘルプ(H)

上記の様な画面が表示され、バックグラウンドで処理が進みます。



上記画面が表示された場合は、「切り替え」を押してください。

📴 自動調整処理中	×
5/8: ボタン(K3, TS27@ config01)の変化量を計測し タ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 イミングの目安は、数値の変動が安定した時点となります。	,ます。 Jかキーを押してください。キーを押すタ
K3, TS27 @ config01: 15350	
	キャンセル ヘルプ( <u>H</u> )

<u>K3のタッチパッド</u>を指でタッチする様にしてください。(※最初にタッチするのは K3 です)

📴 自動調整処理中	×							
5/8: ボタン(K3, TS27 @ config01)の変化量を計測します。 タ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、数値の変動が安定した時点となります。								
K3, TS27 @ config01: <u>18371</u>								
	キャンセル ヘルプ( <u>H</u> )							

表示されている数値が大きくなれば、タッチを検出しています。

e2studioの画面(e2studioがフォアグラウンドになっている状態で)でいずれかキーを叩いてください。 (次のキーに進みます)

SmartRA 学習キット チュートリアル 8





📴 自動調整処理中	×
6/8: ボタン(K2, TS29 @ config01)の変化 ターゲットボード上のボタンを指で触れながら、キ イミングの目安は、数値の変動が安定した時点	と量を計測します。 ニーボードで何かキーを押してください。キーを押すタ ことなります。
K2, TS29 @ config01: 15304	
	キャンセル ヘルプ( <u>H</u> )

K2, K1 も同様に処理します。

⑤ 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	閾値	オーバーフロー	警告/Iラ-		
	config01	ボタン	K1	TS35	1837				
	config01	ボタン	K2	TS29	1860				
	config01	ボタン	K3	TS27	1822				
リトライ 調整処理の継続									
						I	キャンセル	ヘルプ( <u>H</u> )	

閾値の欄がタッチしたことによる数値の増加に伴って決定された値です。3値が大体同じ値になっている事が期待 値です。極端に外れた値があれば、「リトライ対象の選択」のチェックボックスにチェックを入れてリトライしてください。 問題なさそうであれば、「調整処理の継続」で先に進んでください。





#### 1.5.3. プログラムコードへの反映



「ファイルを生成する」で、チューニング結果をソースファイルに反映させることができます。

「例を表示する」で、サンプルコートが表示されますので、hal_entry.c に反映させてください。

```
/* Main loop */
while (true)
{
   /* for [CONFIG01] configuration */
   err = RM_TOUCH_ScanStart(g_qe_touch_instance_config01.p_ctrl);
   if (FSP_SUCCESS != err)
   {
       while (true) {}
   }
   while (0 == g_qe_touch_flag) {}
   g_qe_touch_flag = 0;
   err = RM_TOUCH_DataGet(g_qe_touch_instance_config01.p_ctrl, &button_status, NULL, NULL);
   if (FSP_SUCCESS == err)
   {
       /* TODO: Add your own code here. */
   }
   /* FIXME: Since this is a temporary process, so re-create a waiting process yourself. */
   R BSP SoftwareDelay(TOUCH_SCAN_INTERVAL_EXAMPLE, BSP_DELAY_UNITS_MILLISECONDS);
}
```

上記が「例を表示する」で表示される、メインループ(無限ループ)の部分のコードです。「Add your own code here.」のところにユーザコードを追加します。

button_status が、キーをタッチしている情報が格納される変数です。

SmartRA 学習キット チュートリアル 8 株式会社



タッチキー	button_status	
K3	xx1	キーに触れている際、b0 が1となる
K2	x <mark>1</mark> x	キーに触れている際、b1 が1となる
K1	<b>1</b> xx	キーに触れている際、b2 が1となる

button_status は、64bit の変数で、K3 に触れている場合は、b0 に 1 が立ちます。

・K3のみに触れている button_status = 0x1 ・K2のみに触れている button status = 0x2 ・K1のみに触れている button_status = 0x4 •K3, K2 に触れている button status = 0x3 ・K1~K3 に触れている button_status = 0x7

となります。

```
/* Main loop */
while (true)
{
   /* for [CONFIG01] configuration */
   err = RM TOUCH ScanStart(g qe touch instance config01.p ctrl);
   if (FSP_SUCCESS != err)
   {
       while (true) {}
   }
   while (0 == g_qe_touch_flag) {}
   g_qe_touch_flag = 0;
   err = RM_TOUCH_DataGet(g_qe_touch_instance_config01.p_ctrl, &button_status, NULL, NULL);
   if (FSP SUCCESS == err)
   {
       /* TODO: Add your own code here. */
       if(button status & 0x1) led on(3); //K3
       else led off(3);
       if(button status & 0x2) led on(2); //K2
       else led_off(2);
       if(button_status & 0x4) led_on(1); //K1
       else led_off(1);
   }
```

「Add your own code here.」のところに、LED を ON, OFF させるコードを追加すると上記の様になります。

※FSP の端子設定で、P000~P007 を出力に設定する必要があります

上記のコードを追加すると、K1を押したとき LED0 が点灯します。K2(→LED1), K3(→LED2)も同様です。 (キーにタッチしたときの動作とユーザプログラムを結びつけることができます)





#### 1.5.4. モニタリング

CapTouchには、タッチキーに触れている強さ等の情報をモニタする機能があります。



「ビューを開く」

Sworkspace_RA - e ² studio		- 🗆 ×
ファイル(E) 編集(E) ナビゲート(N) 検索(A) プロジェクト(P) Renesas Views 実行(B) ウインドウ(W)	へいブロ	
🔦 🎄 🔳 🎋 デバッグ(B) 🗸 🔂 RA2L1_TOUCHKEY_QE Debug_Flat 🗸 🌼	! 🗅 + 🗟 🖏   🗞 + 🗞 + 📓 ! 🖻 ! 💌   🖿 🗰 🗱 2. ○e.  🚧 🗮 3. ○e.  🌾 🗮 🛞   🌾 + 🎧 + ! ‰ + 🎋 🗰 💷 🐩 2. ②	> ⊄* <> ▼   ⊡
	Q、 😢 🔢 C/C++ - @ FSP Configuration	h 恭 デパッグ 🐑 CapTouchモニタ RA, RL78 (QE)
🖏 CapTouch#-F-E_9 RA,RL78 (QE) 😒 🛼 🗔 🗔 🖓 📮 🗆	🕲 [RA2L1_TOUCHKEY_QE] FSP Configuration 🗋 hal_entry.c 🖏 CapTouchステークス・チャート RA,RL78 (QE) 😒 🛛 🖳 🗔 🖂 🖂 😫 🙂 🗆	🖏 CapTouchパラメーター覧 R 💥 😐 🗆
モニタリングを有効にする モニタリング操縦: 無効 通信状態: OCDTミュレータで接続中	タッチ//f:	5. ja 22 22 23 (* i
		タッチI/F:
^		
		項目 値
K1 K2 K3		
	21 Alle[1]:	
	5535	
	43143	
< >		
3.3. CapTouchマルチ・ステータス・チャート RA, RL78 (QE) ※ 見 の の の の の の の の の の の の の の の の の の	22766	
	16383	
	1000	^
49149		
32766		
	RALI_TOUCKIKEY QE Debug, Flat [Renesas GDB Hardware Debugging]	
	9000_0=H896	
16385	ハンビロ 1001 ハードウェア・ブレーウボイントはアドレス9x632(1登定します。	
	ダワンロード開始 ダンロード開始 ダンロード開発	
0	ダウンロード開始	·
実行中	- (j)	······································

モニタリングを有効にする。



SmartRA 学習キット チュートリアル 8 株式会

Наниса.			
FIECTIONIC デバッグの終了			
I workspace - RA2L1_TOUCHKEY_QE/ra_gen/main.c - e ² studio			– 🗆 X
ファイル(E) 編集(E) ソース(S) リファクタリング(T) ナビゲート(N) 検索(A) プロジェクト(P) Re	nesas <u>V</u> iews 実行( <u>R</u> ) ウィンドウ( <u>W</u> ) ヘルプ( <u>H</u> )		
▲ 茶 デバッグ(B) ✓ 〒 RA2L1_TOUCHKEY_QE Debug_Flat	<u>∽ ∰</u>   <mark>11 •                                      </mark>	/ ③ / () 🕪 弓 () (梁 (梁) 将 * 9 • 10, * 1	\$ III III \$ \$ 4 8 4 4 5 4 1
	CanTaurah 77-87.58-1 PA PI 70 (OD) 92	Q : E   4	출C/C++ @ FSP Configuration 참 7/(ック 값 CapTouch ₹= 5 RA, RL78 (QE)
	<ul> <li>&lt;) Capiouch X7-9X-7F-F KA, KL/8 (QE) 23</li> <li>タッモリノE: K1@ config01</li> </ul>		
モニアリアノを有効にする。モニアリアノ機能に有効、適応化能、OCDエミュレーア(技術中	「「「「「」」、「「」」、「」、「」、「」、「」、「」、「」、「」、「」、「」、		タッチI/F: K1@ config01
>>>>	カウント値: 17972 基準値: 15416 関値:	1909 差分值: 2556	I/F種別: ボタン(自己), チャネル: TS35
	データ収集の開始		項目值
K1 K2 K3	ノイズ値[NT]: 平均値[NT]: 最小値:	最大值:	ドリフト補正間隔 255
S-m	ノイズ値[T]: 平均値[T]: シグナル値:	SNR值:	ポジティブ・ノイズフィルタのサイクル 3
2			ネガティフ・ノイスフィルタのサイクル 3 移動平均フィルタの深度 4
	18209		タッチ関値 1909 ドファドリンフ 95
	17303		副值
	16800		
	16097		
CapTouchマルチ・ステータス・チャート RA, RL78 (QE) 23	15394		
K1 TS35 @ config0 v K2 TS29 @ config0 v K3 TS27 @ config0 v			
17972 2 15390 5 16401	RA2L1_TOUCHKEY_QE] FSP Configuration 🗈 startup.c 💽 m	ain.c 🛙	
	3 ⊖ int main(void) 4 {		^
	6 0000067a return 0;		
18209	8	タッチしていると認識され	×
17471	א-עעב 🖳	ている期間	
16736	QE for Capacitive Touch	ていの当時]	
16001			
15266	<		
			QEはモニタリンク機能を実行中です。
測定数値(3キーを表示)			
			タッチしているキーに
			指のアイコンが表示され
赤線の部分(タッチ IF の選択)で	、表示させたいキーを選択。		ます

タッチキーはタッチしていると、観測数値が上昇します。閾値(緑のライン)を超えるとタッチしているとみなされます。

モニタを終了させる時は、

・「モニタリングを有効にする」のボタンを押し、モニタを停止する

・赤の四角のアイコンでデバッグを終了する

としてください。

※本機能は、エミュレータを使ったモニタなので、エミュレータ経由でプログラムを実行する必要があります。エミュレータ経由でプログラムが実行されていない場合は、以下の手順でプログラム実行後に、モニタリングを実行してください。

workspace_RA - RA2L1_TOUCHKEY_QE/ra/fsp/src/bsp/cmsis/Device/RENESAS/Source/startup.c - e ² studio	
ファイル(トー編集(E) ソース(S) リファクタリング(T) ナビゲート(N) 検索(A) プロジェクト(P) Renesas Views 実行(R) ウィンドウ(W) ヘルプ(H)	
「「「「「」」」」     「「」」     「「」」     「「」」     「「」」     「「」」     「「」」     「「」」     「「」」     「「」」     「「」」     「「」」     「「」」     「「」」     「「」」     「「」」     「「」」     「「」」     「「」」     「「」     「「」     「「」     「「」     「「」     「「」     「「」     「「」     「「」     「「」     「「」     「「」     「「」     「「」     「「」     「「」     「「」     「「」     「「」     「「」     「「」     「「」     「「」     「「」     「「」     「「」     「「」     「「」     「「」     「「」     「「」     「「」     「「」     「「」     「「」     「「」     「「」     「「」     「「」     「「」     「「」     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「     「       「	• 00 🔳

(1)エミュレータ接続(プログラムをエミュレー	タ経由でマイコンボードにダウンロード)
(2)実行	





1.5.5. COM ポートでのモニタリング

🔯 workspace_RA - e² 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) ウィンドウ(W)
🌾 🔳 🎄 デパッグ(B)	✓ RA2L1_TOUCHKEY_QE Debug_Flat ✓ ☆
🖏 CapTouchボード・モニタ RA,RL78 (QE) 🕴	
モニタリングを有効にする モニタリング機能:有効	b, 通信状態: OCDIミュレータで接続中 ^
/	· ·

Cap Touch のモニタは、エミュレータ経由で実行されますが、COM ポートを使ってモニタする方法もあります。 COM ポート経由では、ボード単体で動作します。(エミュレータ不要です) ※上記画面はエミュレータ接続の場合です



CaTouch メイン/センサ・チューナの画面で、シリアル接続「接続」を押す

SmartRA 学習キット チュートリアル 8



Workspace_RA - RA2L1_TOUCHKEY_QE/ra/fsp/src/bsp/cmsis/Device/RENESAS/Source/sta	rtup.c - e² studio		- 🗆 X		
ファイル(D) 編集(E) ソース(S) リファクタリング(T) ナビゲー(N) 検索(A) プロジェクト(D) Reneass <u>V</u> ews 実行(E) ウインドク(M) ヘルプ(L)					
▲ # デバッグ(B) ~ 〒RA2L1_TOUCHKEY_QE Debug_Flat 、	👾 - [ 11 - 12 - 12 - 12 - 12 - 12 - 12 -	, • 1 · · · · · · · · · · · · · · · · · ·	2 + 1		
월 • 월 • 한 랴 � • 아 • M M	c	、 🖻 🔤 C/C++ 徳 FSP Configuration 🎄 デバック	グ 🖏 CapTouchモニタ RA,RL78 (QE)		
🖏 CapTouchボード・モニタ RA,RL78 (QE) 😒 🛛 💀 🗔 🗔 🖓 📮 🔋 🖓 👘	② CapTouchステータス・チャート RA, RL78 (QE) ※ ② CapTouchメイン/センサ・チューナ RA, RL78 (QE)		apTouc 🛛 🎋 デバッグ 🖳 🗖		
モニタリングを有効にする モニタリング機能:有効,通信状態:シリアル通信(UART/USB)で接 ^	タッチ//F: K1@ config01                  □ 選択状態を同期する				
タッチl/F: ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・	I/F種別:ボタン(自己), チャネル: TS35	97	≠I/F:		
^	カウント値: 18774 基準値: 15472 関値: 1909 差分値: 3302				
	データ収集の開始	項	目値		
K1 K2 K3	ノイズ値[NT]:   平均値[NT]:   最小値:   最大値:				
	ノイズ値[T]: 平均値[T]: シグナル値: SNR値:				
5					
$\sim$	18785				
	17940				
	17005				
· · · · · · · · · · · · · · · · · · ·	17055				
<	16250				
CapTouch マルチ・ステータス・チャート RA, RL78 (QE) ※	15405				
K1, TS35 @ config0' v K2, TS29 @ config0' v K3, TS27 @ config0' v	@ [RA2L1_TOUCHKEY_QE] FSP Configuration				
18774 15378 15408	64 SystemInit();	^ _	,		
	66 /* Call user application. */ 67 main():		^		
	68 69 9 while (1)				
	<	>			
	U-V/L ※	🗟 🚮 🔛 🛃 🖬 + 🗂 - 🗆			
17017	QE for Capacitive Touch				
16122					
16133					
15249	<	×	~		
		QEはシリアル接続	売から…をポーリング中です。 🛛 🔳 🕾		
(m) = 2 (0)					

🔯 workspace_RA - e² studio	
ファイル( <u>F)</u> 編集( <u>E)</u> ナビゲート( <u>N</u> ) 検索( <u>A</u> ) プロジェクト( <u>P</u> ) Renesas <u>V</u> iews 実行( <u>R</u> ) ウィンドウ( <u>W</u> )	
🔦 🎄 🔳 🔅 デバッグ(B) 🗸 💽 RA2L1_TOUCHKEY_QE Debug_Flat 🗸 🔅	
🔾 CapTouchボード・モニタ RA, RL78 (QE) 🛛 🗔 🗔 🗔 🗔 🕄 🕄 🗧 🗆	
モニタリングを有効にする モニタリング機能: 有効, 通信状態: シリアル通信 (UART / USB) で接続中	
⁄ ማንቻI/F: ✓	

エミュレータ接続と同じように、キーの状態をモニタ可能です。初期設定時(最初のチューニング)はエミュレータが必 要ですが、動作をモニタする際は、ボード単体(マイコンボード+ベースボードの組み合わせ)でも実行できます。

(サンプルプロジェクト RA2L1_TOUCHKEY_QE の動作を見るのであれば、エミュレータがなくても、シリアル通信 経由で行えます。新規にプロジェクトを作成して、チューニングを行う場合は、E2Lite エミュレータ等が必要になりま す)

SmartRA 学習キット チュートリアル 8 株式会社



## 2. RA2L1_CTSU

前の章では、ミドルウェア(CapTouch)を使って、タッチキーの動作環境を構築しました。本チュートリアルは、タッチ キー(CTSU)のプログラムを(自動コード生成を使わずに)書き下してみたというサンプルとなります。

## 2.1. プログラムの動作







Copyright (C) 2021 HokutoDenshi. All Rights Reserved. SmartRA KIT CTSU(TouchKey) TUTORIAL Please do NOT touch key pad! offset optimize & no-touch data scanning... SO = 0S0 = 20S0 = 40(中略) S0 = 600S0 = 620K1(TS35) SO setting value = 622 K2(TS29) SO setting value = 628 K3(TS27) SO setting value = 634 initial sens value measure... initialize finished. TEST MENU: m : sens value monitor t : tuning key parameter p : print initial reference value s : statistics data print i : re-initialize >

起動すると、シリアル端末には上記の様な表示が出ます。

また、LCD画面には、



の表示が出ます。ここで、タッチキーパッドの K1 に触れてみてください。

RA2L1 SelfCap >K1(TS35)

上記の様な表示となるはずです。

K1 を押している時には、LED0 が点灯します。同様に、LED1(K2), LED2(K3)が点灯します。

タッチ(触れた)キーが LCD に表示されます(複数のキーにタッチした場合は、一番強く(しっかりと)触れているキー が表示されます。





シリアル端末側は、キーボードからコマンドを受け付ける様になっていて、"m"コマンドを打つと

・mコマンド

sens valu	ue monito	r(press a	ny key to	o exit)>
key :	K3(TS27)	K2(TS29	K1(TS35)	;)
initial	2626	2600	2598	2 秒毎に1 行表示
sens val	ie :			
	2600	2601	2596	
	2672	2633	3478*	1/1 1-600 4 4 4 4
	2652	2620	4060*	NII-開れし/こ場合
	2645	2600	2601	
	2662	4012*	2669	K2 に触れた場合
	2601	2607	2607	
	3998*	2655	2603	K3 に触れた場合
	2670	2618	2603	
	2656	3759*	4118*	
	2669	3807*	4111*	K1とK2に触れた場合
	2719	2656	2673	<b>V</b>
				▼
			1/4	-
	<b>N</b> 3	KZ	KT	

上記の様に、リアルタイムにキー毎の測定値(どのぐらいタッチしているか)が表示されます。

・tコマンド

タッチの閾値をチューニングするコマンドです。

threshold value optimize...

Please touch key pad -> K1 & PC keyboard enter

上記表示がでますので、K1のタッチキーにタッチして、PCのキーボードを(任意)叩いてください。

```
touch data scaning...
key scan end.
key pad(TS ch) = K1(TS35)
touch(ave) = 4054 no-touch(ave) = 2589
```

タッチ時と、非タッチ時の数値が表示され、次のキー(K2)のチューニングに移ります。





```
Please touch key pad -> K2
& PC keyboard enter
touch data scaning...
key scan end.
 key pad(TS ch) = K2(TS29)
 touch(ave) = 4025 no-touch(ave) = 2626
Please touch key pad -> K3
& PC keyboard enter
touch data scaning...
key scan end.
 key pad(TS ch) = K3(TS27)
 touch(ave) = 4040 no-touch(ave) = 2630
tuning end.
```

#### ・pコマンド

print init	ial re	ference	value>
TSxx :	SO :	Sens :	threshold
K3(TS27):	635 :	2630 :	1.100
K2(TS29):	629 :	2626 :	1.100
K1(TS35):	624 :	2589 :	1.100

SO(後述)値と、タッチしていない場合の測定値(Sens)、タッチキー判定の閾値(割合)が表示されます。上記は、チ ューニング(tコマンド)前の値です。

```
print initial reference value>
      : SO : Sens : threshold
TSxx
K3(TS27): 635 : 2630 : 1.268
K2(TS29): 629 : 2626 : 1.266
K1(TS35): 624 : 2589 : 1.283
```

tコマンドでチューニングを行うと、thresholdの値が更新されます。

SmartRA 学習キット チュートリアル 8 株式会社



・ベースボードのタッチキーに直接タッチした場合の測定値、チューニング後の閾値例

	非タッチ時の	タッチ時の	閾値
	測定値	測定値	(非タッチ時の
			測定値に対する
			倍率のセンター)
K1	2589	4054	1.283
K2	2626	4025	1.266
K3	2630	4040	1.268

通常タッチキーは、導体(本キットでは基板パターンの銅箔)の上に、アクリル板等を設けます。本キットでは、アクリ ル板が無い代わりに基板(1.6mm 厚)を誘電体として使用しています。

・ベースボードのタッチキーにアクリル板(3mm 厚)を載せてタッチした場合の測定値、チューニング後の閾値例

	非タッチ時の 測定値	タッチ時の 測定値	閾値 (非タッチ時の 測定値に対する 倍率)
K1	2605	2875	1.052
K2	2592	2836	1.047
K3	2618	2849	1.044

アクリル板を挟んだ場合は、タッチ時の測定値の上がり方がマイルドになります。(このケースでは、導体の上に、基板+アクリル板の 4.6mm 厚の誘電体が挟まっています。)

・sコマンド

sens statistics(measure 15 sec)> sens value statistics data

key(TSxx) : Ave ( Max - Min , 3sigma ) K3(TS27) : 2640 ( 2711 - 2576 , 77 ) K2(TS29) : 2652 ( 2721 - 2622 , 36 ) K1(TS35) : 3866 ( 3901 - 3832 , 35 )

15 秒間の測定を行い、15 秒間の最大-最小、標準偏差 x3(3 d)値を表示します。上記は、15 秒間ずっと K1 のみ タッチしていた場合の表示例です。

・iコマンド

```
re-initialize(please do not touch key)>
ref current, current offset optimize.
S0 = 0
S0 = 20
.. (中略)
initial sens value measure...
initialize finished.
```

再初期化を行います(リセットボタンを押した場合と同じです)。

SmartRA 学習キット チュートリアル 8





#### 2.2. タッチ判定の原理

本キットでは、「自己容量」タイプのタッチキーとなっています。

自己容量タイプのタッチ判定は単純です。



自己容量は、マイコンの端子(TSxx)に 1:1 でキーパッドが接続される形となります。

(キーパッドの数=測定 ch 数となります)

TSxx のラインを L→H に遷移させると、端子からは容量に比例した電流(i)が流れ出します。電流を測定する事により、TSxx ラインにつながっている容量の大きさを測る仕組みです(詳細は、マイコンのハードウェアマニュアルや、ル ネサスエレクトロニクスより提供されている、CTSU 測定の原理の資料を参照ください)。

・非タッチ時の TS35 のセンサ値(CTSUSC)を取得(C1 に相当する値)

・タッチ判定の閾値を決める(*1)

・タッチ時は TS35 のノードの容量値が増加=センサ値が増加する(C1+C2 に相当する値)

・センサ値が閾値を超えていたら「タッチしている」と判断する

(*1)判定の閾値(g_touch_threshold)は、初期値は 1.1(10%以上センサ値が増加した場合)としています サンプルプログラムでは、"t"コマンドで、実際にタッチした際のセンサ値を元に閾値を最適化できる様になっています 閾値は、「割合で判断する」手法と「絶対値で判断する」手法があるかと考えますが、本サンプルプログラムでは前者 としています。

※実際のタッチキーアプリケーションでは、

・起動後に実際にタッチして、閾値の最適化が可能

・装置組み立て後に、装置毎に閾値の最適化が可能

・予め規定値を代入

等、用途に応じて閾値の決め方は変わるかと考えます。



#### 2.3. タッチキー処理の流れ

タッチキー機能(CTSU)を使用する基本的なフローを示します。

(1)CTSU 初期化

•TSCAP 端子設定

·TSxx 端子設定

・CTSU レジスタ設定

#### (2)電流オフセット値の設定

・CTSUSO.SO レジスタ値の最適化

(3) 非タッチ時のセンサ値取得

・非タッチ時のセンサ値(基準値)取得

(4)タッチ閾値の最適化(省略可)(tコマンド)

・タッチ時のセンサ値(基準値)取得

(5)タッチ判定

・センサ値を取得して非タッチ時の基準値と比較してタッチ/非タッチの判定を行う。

SmartRA 学習キット チュートリアル 8

RA マイコンのタッチキー(CTSU)は、容量を測定して、センサ値(数値)に変換しています。センサ値は「電流オフセット値」の設定により変化するため、上記(2)の部分で適切な設定値を求める事が測定精度に関わってきます。







電流オフセット値(SO 値)と、センサ測定値(CTSUSC)の関係の実測例を示します。

<u>図 2-1 センサ値 SO 依存性</u>

電流オフセット値(SO)(0~1024の範囲で設定可能)を振った際の、非タッチ時とタッチ時のセンサ測定値(CTSUSC レジスタ値、容量値に対応)をプロットしたものです。

このグラフからは、SO 値を大きく(740 以上)した場合は、センサ測定値は 0 に張り付く事が判ります。また、SO 値 が小さいときは、タッチと非タッチの測定値の差分が小さい事も見て取れます。

できるだけ右側(SO が大きな値)としたいが、動作限界近傍にすると経時変化等で動作しなくなってしまう事も考えられます。そのため、上図では 600 あたりを選ぶのが良いかと考えます。

※どの動作点を選ぶかは考え方次第であると思います。SO 値は大きく取った方がタッチと・非タッチの差分が大きく 取れるが、大きくし過ぎるとそもそも測定値が取れなくなるという事を念頭に、値を決めればよいと思います。

本サンプルプログラムでは、SO=0の非タッチ時のセンサ測定値の20%(ctsu.h内で定数定義)を下回らない点とします。

非タッチ時, SO=0 のセンサ測定値が、13,000 程度なので、その 20%である 2,600 のあたり、SO=640 を電流オフ セット値とします。

※オフセット値はマイコンチップの個体差があるので、お手元のボードでは640ぐらいにはならない事もあり得ます。

SmartRA 学習キット チュートリアル 8



本サンプルプログラムでは、

src¥ctsu¥ctsu.h

//電流オフセット最適化 **#define CTSU_OFFSET_MERGIN (0.20f)** //SO=0のセンス値の 20%の点で動作させる

上記ファイル内でこの割合値を定義しており、初期値は 0.2(20%)です。

タッチの感度を上げたい場合は、この値を小さくする(但し誤作動の確率は上がる)。感度を下げたいときは、大きく すると良いと思います。設定可能なのは、0~1の範囲内です。

電流オフセット調整レジスタ値(SO)を決めたら、次に非タッチ時のセンサ値(CTSUSC)を取得します。

起動時の初期化としては、ここまでとなります。

本サンプルプログラムは、閾値の最適化を行わなくても動作させることができますが、実際のタッチ時のセンサ測定 値を取得し、非タッチ/タッチ時の中間のセンサ測定値に閾値を設定する事により、動作マージンの向上を図ること ができます。

但し、実際のタッチキーアプリケーションでは、起動後に毎回キーにタッチして閾値を調整する事は難しいかと思います。

閾値の初期値は、非タッチ時のセンサ測定値に対して、センサ測定値が10%増加した点としています。

src¥ctsu¥ctsu.h

//判定閾値(初期値)
#define CTSU_DEFAULT_THRESHOLD (1.1f)

tコマンドで、閾値の最適化が行えますので、tコマンドで取得した閾値を初期値にしてしまっても良いかと考えます。

プログラム内では無限ループで、センサ測定値(CTSUSC)を取得して、非タッチ時のセンサ測定値(CTSUSC)と比較を行い、タッチの判定を行います。

SmartRA 学習キット チュートリアル 8





## 2.4. フローチャート



SmartRA 学習キット チュートリアル 8 株式会社



ータッチキーメインループ(*1) フローチャートー





-CTSU 割り込み処理 フローチャート-

・CTSU WRITE 割り込み[¥ctsu¥ctsu intr.c, intr ctsu ctsuwr()]



CTSU WRITE 割り込みでは、電流設定値等のレジスタにデータをセットする処理を行っています。CTSU の処理を スタートさせると、この割り込みが入りますので、そのタイミングで上記レジスタに値を書き込みます。 (CTSUSO レジスタに値が書き込まれると、実際のセンサ値の測定に移ります)

・CTSU READ 割り込み[¥ctsu¥ctsu_intr.c, intr_ctsu_ctsurd()]



CTSU RD 割り込みは、1ch(1 つのキー)の測定が終わった段階で呼び出される割り込みです。測定値を、グロー バル変数にコピーする処理を行っています。

※CTSU WRITE, CTSU READ 割り込みでは、(処理が判り易い様に)直接レジスタへの書き込み、読み出しを行っ ていますが、ここで使用しているレジスタアクセスは DTC が使用可能です。(実際のアプリケーションプログラムで は、DTC の使用もご検討ください。)

・CTSU FN 割り込み[¥ctsu¥ctsu_intr.c, intr_ctsu_ctsufn()]







CTSU FN 割り込みは、全チャネルの測定が終わった段階で呼び出される割り込みです。インデックス変数やフラグ 変数の設定を行っています。

タッチキー機能(CTSU)使用時のセンサ値測定のフローは、以下の様になります。



CTSU の処理は基本的には、上記の流れとなります。

測定開始(CTSUSTRT=1)とすると、CTSU マクロは(内部で準備をした後)CTSU WRITE 割り込みを掛けます。ユ ーザプログラム側で、CTSU WRITE 割り込みルーチン内では、CTSUSO(オフセット電流設定レジスタ)に、値を書き 込みます。

※CTSUSO 設定値(SO 値)は、測定 ch 毎に最適化されている事が望ましいです

CTSUSOレジスタに値を書き込むと、(端子容量の)センサ測定値が測定され、1ch(1つのキー)の測定が終わると、CTSU READ 割り込みが入ります。ここでは、センサ値の測定値(CUSUSC)のレジスタを読み出します。

※センサ値が格納されるレジスタ(CTSUSC)は、1 つしかありませんので、CTSU READ 割り込みの時点で保存する 必要があります





(例えば、A/Dコンバータの結果を格納するレジスタは、ch 毎に別になっていますが、CTSU は 1 つしかありません)

CTSUSC レジスタ読み出し後に、CTSU マクロは次の測定 ch の準備に入り、準備ができると、再度 CTSU WRITE 割り込みを掛けます。

2ch 目の CTSUSO 値を書き込むと、2ch 目の測定後 CTSU READ 割り込みが入り、同様の処理が、測定 ch 分続きます。

※測定 ch は、予め測定対象と設定している ch 数分です

最後の測定 ch まで一連の処理が終わると、CTSU FN 割り込みが入ります。

ここで、全 ch の結果まとめや、フラグレジスタのクリア等必要な処理を行います。

※本サンプルプログラムでは、全 ch の測定が終わると、フラグをクリアし、無限ループ内で次の測定を開始するよう になっていますが、CTSUSTRT=1のセットは、例えばタイマで一定時間毎に行う等、タイミングは自由です





電流オフセット値(SO)は、非タッチ時の測定値をベースに最適化しています。

電流オフセット値の最適化フロー ctsu_offset_optimize()



上記フローチャートでは記載を省略していますが、測定 ch 毎に SO 値を算出しています。

SmartRA 学習キット チュートリアル 8

## 2.5. CTSU 関連の関数

ctsu_init

概要:初期化関数

宣言:

void ctsu_init(void);

説明:

·変数初期化

•TSCAP 端子放電

・TS 端子設定

・CTSU レジスタの設定

を行います

引数:

なし

戻り値:

なし





ctsu measure

概要:測定指示関数

宣言:

void ctsu_measure(void);

説明:

・測定の開始(CTSUSTRT=1)

を行います

引数:

なし

戻り値:

なし

ctsu_set_param

概要:パラメータの設定関数 宣言:

void ctsu_set_param(void);

説明:

・測定 ch 毎の CTSUSO レジスタ値のセット

を行います

引数:

なし

戻り値:

なし

ctsu_read

概要:測定値の読み出し関数

宣言:

void ctsu_read(void);

説明:

・センサ値(CTSUSC), ステータスのレジスタ値をグローバル変数へコピー

を行います

引数:

なし

戻り値:

なし





#### ctsu_offset_optimize

概要:電流オフセット値(SO)の最適化関数

#### 宣言:

void ctsu_offset_optimize(void);

説明:

・SO 値の最適値の算出

を行います。

引数:

なし

戻り値:

なし

ctsu_ref_initial_value

概要:センサ初期値取得

宣言:

void ctsu_initial_value(void);

説明:

・非タッチ時のセンサ値(CTSUSC)の取得

を行います。

引数:

なし

戻り値:

なし

ctsu_result

概要:タッチ結果取得関数

宣言:

unsigned char ctsu_result(unsigned char *key_state);

説明:

・タッチ結果の判定

を行います。

引数:

*key_state: 押しているキーが格納されます

unsigned char 型の key_state[n]が、押しているキー0x01, 押されていないキー0x00 となります。キー数(3) 以上の配列のポインタを引数に指定してください。

戻り値:

タッチ判定されている中で、一番容量の変動が大きなキー

SmartRA 学習キット チュートリアル 8 株式会社 北手電子

Hatuta

補足:

複数のキーが押されている場合、key_stateには、押されているキーの情報が入ります。 戻り値は、一番しっかり押しているキーとなります。

・割り込み関数

intr ctsu ctsuwr

概要:CTSU WRITE 割り込み関数

#### 宣言:

intr_ctsu_ctsuwr(void)

#### 説明:

・ctsu_set_param()(測定 ch 毎の SO 設定)の呼び出し、セット)

を行います

#### intr_ctsu_ctsurd

概要:CTSU READ 割り込み関数

#### 宣言:

intr_ctsu_ctsurd(void)

#### 説明:

・ctsu_read()(測定結果のコピー)の呼び出し を行います

intr_ctsu_ctsufn

概要:CTSU FN 割り込み関数

宣言:

intr_ctsu_ctsufn(void)

説明:

・測定 ch を示すインデックス変数のクリア(0代入) ・測定中を示すフラグ変数を「測定終了」のステータス変更 を行います





#### 2.6. CTSU 関連のグローバル変数

#### unsigned char g_key

現在(一番強く)押されているキー番号を示す変数。(Oxff の時は、どのキーも押されていないと判断) ctsu_result()の戻り値を g_key に代入。メインループ内で随時更新。

unsigned char g_key_state[]

現在押されているキーの情報を示す変数。

キー数(3)の配列変数となっており、押しているキーは 1, 押されていないキーは 0 が格納されます。g_key 同様に、 メインループ内で随時更新されます。

unsigned char g_sens[]

センサ値が格納される変数です。測定 ch 数(3)の配列変数で、メインループ内で随時更新されます。

long g_initial_sens[]

初期センサ値で、非タッチ時の基準値として使用されます。256回測定した平均値が格納されています。(256回の 平均を計算するために、long型で定義していますが、最終的には2バイト(0~65535)内の値となります)測定 ch 数の配列です。

unsigned long g_status[]

測定 ch 毎のステータスが格納される変数です。メインループ内で随時更新されます。

unsigned long g_ctsu_ctsuso[ ]

測定 ch 毎の、CTSUSO(SO)レジスタ設定値を保存している変数です。CTSU WRITE 割り込みの際に、本変数が レジスタにコピーされます。測定 ch 数分の配列です。

#### unsigned short g_data_index

測定 ch のインデックス変数です。CTSU READ 割り込み(測定完了)時に、インクリメントされ、CTSU FN 割り込み (全 ch の測定終了)時にクリアされます。



float g_touch_threshold[]

タッチ、非タッチの閾値を示す変数です。初期値 1.1 で、"t"コマンドで閾値の最適化を行うと更新されます。キー数 分の配列です。

volatile unsigned char g_in_measure

測定中を示すフラグ変数です。 CTSU_STATE_IDLE(0):測定中ではない CTSU_STATE_IN_MEASURE(1):測定中 CTSU_STATE_FINISHED(2):1 セット(全測定 ch)の測定完了

volatile unsigned short g_initialize_finished

初期化完了を示すフラグ変数です。 CTSU_INITIAL_UNFINISHED(0):初期化が終わっていない CTSU_INITIAL_FINISHED(1):初期化済み

## 2.7. CTSU 関連の定数値

#define CTSU_STATE_IDLE (0)	//非測定中(測定開始待ち)
#define CTSU_STATE_IN_MEASURE (1)	//測定中
#define CTSU_STATE_FINISHED (2)	//全チャネルの測定が終了

フラグ変数 g_in_measure の状態を示す定数値。メインループ内では、CTSU_STATE_FINISHED になったタイミングで、キーの判定。CTSU_STATE_IDLE であれば、測定開始指示。CTSU_STATE_IN_MEASURE であれば何もお行いません。

#define CTSU_INITIAL_UNFINISHED (0) //初期化完了前 #define CTSU_INITIAL_FINISHED (1) //初期化完了後

初期化終了状態を示す定数値。

#define TS00 (0)

•••

#define TS17 (35)

キー等を、TS??の名称でアクセスするための定義値。

HALA



#define TS_MAX (35)

TS 端子最大数。

#define CTSU_VALID_TERMINALS (3)

TS??で使用している端子数。

#define CTSU_CH_NUM (3)

測定 ch 数。自己容量タイプの場合は、=キーパッド数となります。

#define CTSU_OFFSET_MERGIN (0.20f)

SO=0 のセンサ測定値の 20%を下回らない点を SO(電流オフセット)の設定値とします。

#define CTSU_CTSUSO_SO (140)

電流オフセット(SO 値)の最適化を省略した際に使用される SO 値です。(通常は未使用です)

#define CTSU_DEFAULT_THRESHOLD (1.1f)

判定閾値の初期値です。測定値が10%以上増加した際、タッチしているとみなされます。"t"コマンドで最適化(値の 更新)が行われます。

#define CTSU_MONITOR_PERIOD (2)

"m"コマンドでの表示更新頻度。2秒に1回。

#### //#define CTSU_OFFSET_OPTIMIZE_OMIT

本定数定義時は、起動後のオフセット電流(SO)は、固定値とします。

#define DEBUG_PRINT

定義時は端末(仮想 COM ポート)に対するメッセージの表示が多くなります。

SmartRA 学習キット チュートリアル 8

#define DEBUG_PORT

定義時は測定終了毎に、ポート(P400)を反転させます。



#### 2.8. まとめ

本チュートリアルは、RA2L1 が持つタッチキー(CTSU2)の機能を扱いました。前半は、ミドルウェアを使った場合。 後半は、スクラッチ(ミドルウェアを使わずにプログラムを書き下した場合)です。

タッチキーは、実際にタッチしなくても、手を近づけるだけでもタッチ判定が可能です。(ボタンのパターンの作り方や 感度調整にも拠ります。)

物理的なスイッチを持たず、触れたり手をかざすことにより反応する機器は、身近にも結構ありますが、その中身は 本チュートリアルで紹介しているものかと思います(容量を測定して、容量値の変化から判定を行っている)。

RA2L1の持つ CTSU2の機能を使うと、タッチの判定が比較的簡単に精度良く行えますので、タッチで動作するア プリケーションを作成する場合は、専用の IC 等を使うことなく、RA2L1のマイコンのみで作る事を検討してみてください。



## 取扱説明書改定記録

バージョン	発行日	ページ	改定内容
REV.1.0.0.0	2021.7.6		初版発行

#### お問合せ窓口

最新情報については弊社ホームページをご活用ください。 ご不明点は弊社サポート窓口までお問合せください。

# _{株式会社} 北丰電子

〒060-0042 札幌市中央区大通西 16 丁目 3 番地 7 TEL 011-640-8800 FAX 011-640-8801 e-mail:support@hokutodenshi.co.jp (サポート用)、order@hokutodenshi.co.jp (ご注文用) URL:http://www.hokutodenshi.co.jp

商標等の表記について

- ・ 全ての商標及び登録商標はそれぞれの所有者に帰属します。
- ・ パーソナルコンピュータを PC と称します。



ルネサス エレクトロニクス RA マイコン搭載 HSB シリーズマイコンボード 評価キット

SmartRA 学習キット チュートリアル 8

©2021 北斗電子 Printed in Japan 2021 年 7 月 6 日改訂 REV.1.0.0.0 (210706)