

RA4M1-100 LCD タッチキー評価キット [QE CapTouch 導入編] 取扱説明書

ルネサス エレクトロニクス社 RA4M1 搭載 HSB シリーズマイコンボード向け評価キット

-本書を必ずよく読み、ご理解された上でご利用ください





一目 次一

注意事	耳項	1
安全上	このご注意	2
概要		4
サンプ	゚ルプログラム CD	5
1. LC	CD タッチキー基板(RA4-LCD-TOUCH)での使用	7
1.1.	プロジェクトの作成	7
1.2.	QE CapTouch の起動	
1.3.	タッチキーインタフェースの作成	17
1.4.	閾値のチューニング	21
1.5.	パラメータファイル生成	24
1.6.	プログラムの実装	24
1.7.	変数のモニタリング	
1.8.	CapTouch のモニタ機能	
1.9.	サンプルプログラム CD に格納されているプロジェクトに関して	
取扱	2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

	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
8-5,	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











概要

本書は、フラッシュメモリ内蔵のルネサスエレクトロニクス製RA4M1マイコン搭載ボードを使用したLCDタッチキーの 評価キットでルネサスエレクトロニクス製のツールである、QEforTouchの使用の導入部を説明する資料です。

QE CapTouch は、タッチキーのプログラムの生成やチューニングを行う事ができるツールで、使用のためには

e2studio FSP QE CapTouch[RA 向け]

を予めインストールを行ってください。 ※FSP のインストーラには、e2studio がセットになっています

本書では、QE CapTouch の使用(導入部分)に関する説明を行っております。タッチキーのプログラムをフルスクラ ッチで記載したいという場合は、「ソフトウェア編」マニュアルを参照ください。ユーザプログラム以外のところをツール で生成したいという場合は、本書を参照ください。





サンプルプログラム CD

ーソースファイルー

フォルダ		内容
SOURCE¥RA4M1_LCD_CTSU¥	src¥common	共通で使用する関数等
	src¥ctsu	タッチキー読み取り
	src¥intr	割り込み設定
	src¥lcd_seg	セグメント LCD 制御
	src¥rtc	リアルタイムクロックドライバ
	src¥sci	SCI(UART)ドライバ
	¥settings	設定ファイル
	src¥timer	タイマドライバ
	src¥hal_entry.c	エントリ関数サンプル

本キットは、ソースファイルの形で構成されていますので、RA マイコン向けのどの開発環境で動作させる事は可能 かと考えますが、開発環境は e2studio を想定しています。

サンプルプログラムは、e2studio(V7.8.0) + GNU Tools for ARM Embedded Processors (arm-none-eabi-gcc) 2019-q4 + FSP v1.1.0 で動作を確認しています。

ーバイナリファイルー

コンパイル、リンク後の mot ファイル(srec)を格納しているフォルダです。マイコンボードに書き込んで動作確認を行う用途等に使用してください。

フォルダ		内容
BINARY¥	RA4M1_LCD_CTSU.srec	LCD 制御とタッチキー読み取り

ーアーカイブファイルー

プロジェクトを、zip ファイルに固めたものです。e2studio のワークスペースにインポートする事が可能です。

フォルダ		内容
ARCHIVE¥	RA4M1_LCD_CTSU.zip	LCD 制御とタッチキー読み取り

※本書で説明している部分です

フォルダ		内容
ARCHIVE_QE¥	RA4M1_LCD_CTSU_QE.zip	QEforTouch を使用したプロジェクト





ードキュメントー

本書を含むマニュアル類を格納しているフォルダです。

フォルダ	内容
MANUAL¥	マニュアル類



RA4M1-100 LCD タッチキー評価キット 取扱説明書 株式会社 **北三日電子**



1. LCD タッチキー基板(RA4-LCD-TOUCH)での使用

1.1. プロジェクトの作成

RAのC/C++プロジェクトを作成願います(プロジェクトの作成手順は、「ソフトウェア編」にも記載がありますので、そちらも参照ください)。

作成時のポイントとなる点をハードコピーを交えて説明します。

(RA4M1_LCD_CTSU_QE というプロジェクト名で自己容量タッチキー基板向けのプロジェクトを作成する事とします)

r						
e ² workspace - RA4M1_LCD_CTSU_QE/configuration.xml	🔄 workspace - RA4M1_LCD_CTSU_QE/configuration.xml - e² studio					
ファイル(E) 編集(E) ソース(<u>S</u>) リファクタリング(T) 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 案行(<u>R</u>) ウィンドウ(<u>W</u>) ヘルプ(<u>H</u>)						
後 ■ 株 デバッグ(B) RA4M1	_LCD_CTSU_QE Debug 🗸 🌼 🗄 🖛 🔚 🐚 🛞 🔻 🔦	• 🔊 😒 🖉 🔊 🖉 🕺 🐘 🗉 🖉 🖓 •	∜ ⊪ Ш 🖌 श 🕹 😻 🛱 💋 🖾			
1 🖆 ▼ 😂 ▼ 🖻 ▼ 🚱 ▼ 🛊 ▼ 🂁 ▼ 🎒 🔑 🔗 ▼	▶ 圓 面 : 如 ▼ 奇 ▼ 🏷 🗢 ▼ 🔿 ▼ クイック・アク	Zス 📄 📴 C/C++ な デバッグ <captouchモニタ・< td=""><td>ー (QE)> - 鬱 RA Configuration ミン CapTouchモニタ RA (QE)</td></captouchモニタ・<>	ー (QE)> - 鬱 RA Configuration ミン CapTouchモニタ RA (QE)			
🎦 プロジェクト・エクスプローラー 🛛 🖓 🖓						
A 😂 KA4M1_LCD_CTSU_QE	Clocks Configuration		Generate Project Content			
 ▷ \$\$ \frac{\}}}}}}}}}{}}} } } } } } } } } } } } }	XTAL 8MHz → PLL Src: XTAL PLL Div /2 PLL Mul x12 PLL 48MHz Summary BSP Clocks Pins Interrupts Event Links Stack	s Components				
🔝 問題 🔊 タスク 🔲 プロパティー 🔋 メモリー使用量 💽	a スタック解析 🧠 スマート・ブラウザー 📓 Debugger Console	: 🎋 デバッグ 📋 メモリー 🖳 コンソール 🛿 🔗 検索	ê ê 😨 📰 🔐 = 🐘 🛃 ▼ 😁 マ 🗆			
CDT ビルド・コンソール [RA4M1_LCD_CTSU_QE]						
	(2) ≠ × (3)					

クロック設定は、XTAL(8), PLL Mul(x12)をデフォルトから変更してください。



RA4M1-100 LCD タッチキー評価キット 取扱説明書 株式会社



🔅 *[RA4M1_LCD_CTSU_QE] RA Configur	ation 🛛		
Pins Configuration			O Generate Project Content
Select pin configuration			<u>∕a</u> - ⊡,
R7FA4M1AB3CFP.pincfg	Generate data: g_bsp_pin_cfg		
Pin Selection	Pin Configuration		
フィルタ入力 🖉 🖻 🖻			M
Analog:DAC12	Module name:	CTSU0	<u>^</u>
 Connectivity:CAN Connectivity:IIC 	Operation Mode:	Enabled 🔹	E
Connectivity:SCI	Input/Output		
Connectivity:SPI Connectivity:SSI	TSCAP: 🗸	P112 •	\$
Connectivity:USBFS	TS00:	None	\$
▲ ✓ Input:CTSU ✓ CTSU0	TS01:	None	\Rightarrow
▷ Input:IRQ =	TS02:	None	\$
▶ Input:KINT	TS03:	None	\$
Graphics:SCLDC System:CGC	TS04·	None	
▷ ✓ System:DEBUG ▼	<	III.	••••••••••••••••••••••••••••••••••••••
Summary BSP Clocks Pins Interrupts	Event Links Stacks Components		

同じく設定 (RA Configuration タブ) Pins タブ

Peripherals - Input:CTSU - CTSU0

Operation Mode: Enabled に変更 TSCAP: P112 を選択(*1) TS17: P403 を選択(P403 しか選択できません) TS21: P000 を選択(P000 しか選択できません) TS22: P001 を選択(P001 しか選択できません) TS28: P015 を選択(P015 しか選択できません)

(*1)タッチキーとLCDを併用する場合、P112を選択してください LCDを使用しない場合は、P205を使用する事もできます





∰ *[RA4M1_LCD_CTSU_QE] RA Configuration 🕅		- 8
Stacks Configuration		Generate Project Content
Threads 🔊 New Thread 🕷 Remove 🕞	HAL/Common Stacks	🖗 New Stack > 🚔 Extend Stack > 🙀 Remove
 ✓ ALL/Common ④ g_ioport I/O Port Driver on r_ioport 	 <i>g_ioport I/O Port</i> Driver on r_ioport 	
Objects		
Objects New Object > Remove Summary BSP Clocks Pins Interrupts Event Links Stack	s Components	

同じく設定(RA Configuration タブ)Stacks タブ

New Stack> をクリック

🗿 New St	ack	> 🐣 Extend S	tack :	> 🔊 Remove			
		Arm	•				
		Driver	•				
		FreeRTOS	•				
		FreeRTOS+	•				
		Middleware	•	CapTouch	×	•	TOUCH Driver on rm_touch
	S	Search		USB	•	Γ	
5	-		_				

Middleware - CapTouch - TOUCH Driver on rm_touch を選択





e ² workspace - RA4M1_LCD_CTSU_QE/configuration.xml - e ² st	udio		
ファイル(E) 編集(E) ソース(<u>S</u>) リファクタリング(T) 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>) ヘルプ(<u>H</u>)	
後 ■ 本 デバッグ(B) マ に RA4M1_LCD_	CTSU_QE Debug 🗸 🌼 🗄 🕶 🐨 🔚 🐚 🛞 🖛 🗞 🕶 🔜 🤅 ۱	■ N 3. 9. R	≒ ∞ 0, - *, ⊪ Ш * 2 & 0 00 00 00 00 00 00 - 60 - 60 - 60 -
* • • • • • • • • • • • • • • • • • • •	 ▼ □) ▼ 	ク・アクセス 🔹 🖻 🔤 C/C++	+ 茶 デバッグ <captouchモニター (qe)=""> 戀 RA Configuration 🖏 CapTouchモニタ RA (QE)</captouchモニター>
🎦 プロジェクト・エクスプローラー 🛛 🔍 🗖	Image: Image		
	Stacks Configuration Threads	HAL/Common Stacks	Generate Project Content €) New Stack >Extend Stack >R Remove
 ▷ In Induces ▷ Øra ▷ Øra_gen ▷ Øra_gen ▷ Øra_gen 		g_ioport I/O Port Driver on r_ioport	TOUCH Driver on rm_touch
 ▷ [a] ra_cfg ▷ [b] script 		(i)	
 is configuration.xml is R7FA4M1AB3CFP.pincfg is ra_cfg.bxt 			for monitor of QE
RA4M1_LCD_CTSU_QE Debug.launch			
▷ ⑦ Developer Assistance	Objects 🐑 New Object > 🎪 Remove		Add DTC Driver for Transmission [Recommended but optional] Add DTC Driver for Reception [Recommended but optional] CTSU Driver on r_cts をクリック
< +	Summary BSP Clocks Pins Interrupts Event Links 📀 Stack	s Components	
👔 問題 🔊 タスク 🛄 プロパティー 🛙 🔋 メモリー使用量 📱 🤉	スタック解析 👒 スマート・ブラウザー 🖳 Debugger Console 🎄	デバッグ 🚺 メモリー 🖳 コンン	
CTSU Driver on r_ctsu			
Settings プロパティ	値		
API Info Common			
Parameter Checking	Default (BSP)		
Support for using DTC	Enabled		
Interrupt priority level	Priority 12		
Module CISU Driver on r_ctsu			· · ·
		Enable DTC support for the CTSU	U module.

追加した Stacks の CTSU Driver on r_ctsu をクリック、プロパティウィンドウの

Support for using DTC Enabled に変更

※プロパティウィンドウが表示されない場合は、メニューの

ウィンドウ – ビューの表示 – プロパティ

で、表示を追加できます







Image: Image		- 6
Stacks Configuration		Generate Project Content
Threads 🔄 New Thread 🗟 Remove 📄	HAL/Common Stacks	🚯 New Stack > 🔌 Extend Stack > 🔬 Remove
A B HAL/Common G g_ioport I/O Port Driver on r_ioport TOUCH Driver on rm_touch	<pre> g_ioport I/O Port Driver on r_ioport </pre>	TOUCH Driver on rm_touch
	(i)	0
		CTSU Driver on r_ctsu Add SCI UART Driver for monitor of QE
Objects New Object > Remove		Add DTC Driver for Transmission Add DTC Driver for Reception [Recommended New + Transfer Driver on r_dtc optional] optionary
Summary BSP Clocks Pins Interrupts Event Links 3 Stack	components	Add DTC for Driver Transmission をクリック

Add DTC Driver for Transmission をクリック、

New - Transfer Driver on r_dtc

を選択

🔅 *[RA4M1_LCD_CTSU_QE] RA Configuration 🛛			
Stacks Configuration			Generate Project Content
Threads 🐑 New Thread 🐑 Remove 📄	HAL/Common Stacks	🗿 New Stack	> 🐣 Extend Stack > 🔬 Remove
Bergen All All All All All All All All All Al	g_ioport I/O Port Driver on r_ioport	UCH Driver on rm_touch	
	(j)	0	
		CTSU Driver on r_ctsu	Stadd SCI UART Driver for monitor of QE
		0	
Objects 🖗 New Object > දි Remove		g_transfer0 Transfer Driver on r_dtc CTSU WRITE (Write request interrupt) Add DTC Driver for Reception [Recommended.hu optional]	New + + Transfer Driver on r_dtc
Summary BSP Clocks Pins Interrupts Event Links 🕴 Stac	ks Components		

同様に、Add DTC Driver for Reception の方も、

New - Transfer Driver on r_dtc

を選択





Image: Image							
Stacks Configuration Generate Project Content							
Threads 🖗 New Thread 🐔 Remove 🕞	HAL/Common Stacks	🕢 New Stack > 🚊 Extend Stack > 🚪	Remove				
All AL/Common Gord Driver on r_ioport Gord Driver on r_ioport TOUCH Driver on rm_touch	<pre> g_ioport I/O Port Driver on r_ioport </pre>	UCH Driver on rm_touch					
	i	0					
		CTSU Driver on r_ctsu	Driver QE				
Objects		 ⊕ g_transfer0 Transfer Driver on r_dtc CTSU WRITE (Write request ① interrupt) ⊕ g_transfer1 Transfer Driver on r_dtc CTSU READ (Measurement ① data transfer request 					
Summary BSP Clocks Pins Interrupts Event Links Stacks C	Components						

DTC 設定後は、上記の様になり CTSU WRITE と CTSU READ の動作が、DTC と紐付けされます。 ※Stacks タブの × 印が消えれば OK です

ー連の設定後、Generate Project Contentを押すと、GUI で設定した項目が反映されたコードが生成されます。





・デバッガの設定



作成したプロジェクトを右クリック

e ² デバック構成	×
構成の作成、管理、および実行	- Alexandre and Alexandre a
 ○ ● ※ ● ⇒ ▼ フィルタ入力 ● C/C++ アブリケーション ● C/C++ リモート・アブリケーション ● EASE Script ● GDB OpenOCD Debugging ● GDB Simulator Debugging (RH850) ● © GDB /\- ドウェア・デ/(ッギング) Java アブリケーション Java アブリケーション Java アブレット ▶ Launch Group (Deprecated) ▲ © Renesas GDB Hardware Debugging [○ Renesas Simulator Debugging (RX) リモート Java アプリケーション ■ 起動グループ 	名前(M): RA4M1_LCD_CTSU_QE Debug → メイン
プロジェクト名 Debugを選択 ※複数の名称が表示されて プロジェクトが開いています ¹ <u>別プロジェクトを閉じる</u> が、 ロジェクトを選択してください (別プロジェクトを閉じるのが	Additional GDB Se ver Arguments いる場合は別 新規作成したプ 前回保管した状態に戻す(火) 適用(Y) 推奨です)



13



構成の作成、管理、および実行			
			1
P B ¥ C * v	•		<u> </u>
	名前(N): RA4M1_LCD_CTSU_QE Debug	-	
	📔 メイン (参 Debugger 🔪 🕨 Startup) 🔲 共通(<u>C</u>) 🦆 ソース	रो	
 C/C++ アプリケーション C/C++ リモート・アプリケーション FASE Scrint 	Debug hardware: E2 Lite (ARM) Target Device: R7F	A4M1AB	
GDB OpenOCD Debugging	GDB Settings Connection Settings デバッグ・ツール設定		
GDB Simulator Debugging (RH850	▲ クロック		
▶ 💽 GDB ハードウェア・デバッギング	メイン・クロック・ソース	外部クロック	-
Java アプリケーション	外部クロック入力周波数 (MHz)	8	
Java アプレット	内蔵フラッシュ・メモリー書き換え時にクロック・ソー	スの変更はい	-
Launch Group (Deprecated)	▲ ターゲット・ボードとの接続		
Renesas GDB Hardware Debugging	エミュレーター	(Auto)	
RA4M1_LCD_CTSU_QE Debug	タイプ	SWD	* _E
Renesas Simulator Debugging (RX	接続速度 (kHz)	Auto	·
リモート Java アプリケーション	▲ 電源		
■ 起動グループ	エミュレーターから電源を供給する (MAX 200mA)	いいえ	_ 「はい」にするとデバッガ接続が失敗します
	供給電圧 (V)	3.3	
	▲ 接続		
	接続時にリセット状態を維持する	はい	
	IDコード (バイト単位)	FFFFFFFFFF	FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
		1+1 \	
< III >>			
18 項目のうち 15 項目がフィルターに一致		前日	回保管した状態に戻す(⊻) 適用(Y)
(\mathfrak{I})			デバッグ(D) 閉じる
\odot			

Connection Settings タブ「外部クロック」「8」「いいえ」を選択。他はデフォルト値で問題ありません。

e ² デバッグ構成							x
構成の作成、管理、および実行						Ŕ	Š.
 □ ※ □ ※ ▼ □ / <i>IVβλλ</i> □ C/C++ <i>Pブ</i>¹<i>Vf</i>-ション □ C/C++ <i>UE</i>-ト·<i>PプUf</i>-ション □ EASE Script □ GDB Simulator Debugging □ GDB Simulator Debugging (RH850 □ GDB <i>JL</i>-F'<i>DTFJV#XJ3va PJUP</i> - ション Java <i>PJUP</i> - ション Java <i>PJUP</i> - ション Java <i>PJUP</i> - ション Uaunch Group (Deprecated) ■ Renesas GDB Hardware Debugging □ RA4M1_LCD_CTSU_QE Debug □ Renesas Simulator Debugging (RX U<i>TE</i> - ト Java <i>PJUF</i> - ション □ 起動<i>JN- <i>J</i></i> 	名前(N): RA4M1_LCD_CT5U_ ③ メイン 体 Debugger ● S 初期化コマンド ③ リセットと遅延(秒): 3 ③ Halt イメージとシンボルをロード ファイル名 ② プログラム・バイナ	QE Debug Startup 日共通(<u>C</u>)	↓ ソース	接続時 Yes		<u>追加…</u> 編集… 除去 上へ 下へ	
< <u>""</u>) 18 項目のうち 15 項目がフィルターに一致	ランタイム・オプション □ プログラム・カウンター設 ☑ ブレークポイント設定先: ☑ 再開	定先(16進):			前回保管した状態に戻す(Y)	」 適用(Y) 間心る	



RA4M1-100 LCD タッチキー評価キット 取扱説明書 株式会社 ノレート電子



Startup タブ

再開 にチェック

適用 閉じる でウィンドウを閉じてください。



15





RenesasViews - RenesasQE - CapTouch メイン/センサ・チューナ RA

を選択してください。



新しいウィンドウ(CapTouch メイン/センサ・チューナ RA)内に、フローが表示されますので、フローに従い手順を 実行してください。





1.3. タッチキーインタフェースの作成



1.準備

プロジェクトの選択 では、先程作成したプロジェクトを指定します。 構成の選択「タッチインタフェース構成の新規作成」を選びます。

ッチインタフェース構成のファイル名: RA4M1_LCD_CTSU_QE	構成(メソッド)の設定	構成の流用/再編集
8:		
		- ダッチ1/F 静電容量方式
	静電容量方式は自己容量」です	自己容量
		ボタン
	ボタンを押し、ボタンをレイアウトに追加	スライダ(横方向)
	· · · · · · · ·	スライダ(縦方向)
Button00 Button01 Button02 Button0	93	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
ICD タッチキー基板(BA4-ICD-TOI		容量センサ
合計4個のボタンを配置しています		電流センサ
L		
		タッチI/Fの削除
Ē		
タッチI/Fの設定 総抵抗の設定 割り付けTSxの解	除	
没定内容に問題があります。		
	作成(<u>C</u>) キャンセル	へレプ(<u>H</u>)





ボタンの配置は、ここではキーパッドの位置に合わせていますが配置は任意です。

ッチョンダフェース構成のファヨル石: R	A4M1_LCD_CTSU_QE	構成(メソット)の設定	構成の流用/再編集
92			タッチI/F 静電容量方式 自己容量
			ボタン スライダ (横方向) スライダ (縦方向)
Button00	Button01 Button02 Button(33	ホイール キーパッド
	● タッチインタフェースの設計	ŧ	タッチパッド シールド端子
	ボタン(自己) 名前 But タッチセンサ	ton00 抵抗値[Ω]	温度補正端子 容量センサ 電流センサ
定	ОК	560 キャンセル ヘルプ(H)	タッチI/Fの削除
タッチI/Fの設定総抵抗の 設定内容に問題があります。	設定 割り付けTSxの解	\$	

Button00をダブルクリックして、設定ダイアログを出してください。





e ² 夕少	チインタフェース	の設定	Ē				×
	ボタン(自己)						
	名前	К1					
	タッチセンサ		抵抗値[Ω]				
	TS28	Ŧ	560	Ŧ			
	ОК			キャンセ	١	へルプ(<u>H</u>)	

Button00 は、キーパッド名"K1"、タッチセンサ電極は TS28 に接続されていますので、上記の様に変更します。抵抗値は、560 Ωから変更の必要はありません。(基板上に、560 Ωの抵抗が実装されています)

同様に Button01~Button03 の設定を行います。

	名前	タッチセンサ	抵抗值
Button00	K1	TS28	560
Button01	K2	TS22	560
Button02	K3	TS21	560
Button03	K4	TS17	560

TS??の設定は、上記の様にしてください。

※名前は任意ですが、"0"の様に数字で始まると弾かれる様です





● 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: RA4M1_LCD_CTSU_QE 構成(メソッド)の設定 説明:	構成の流用/再編集
	タッチI/F 静電容量方式
	目己容量 ▼ ボタン
	スライダ (横方向) スライダ (縦方向)
K1 K2 K3 K4	ホイール
	 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
タッチI/Fの設定 総抵抗の設定 割り付けTSxの解除	
作成(<u>C</u>) キャンセル	へルプ(<u>H</u>)

設定後は上記の様になります。(ボタンの色が赤の場合は、設定に誤りがあります。TSxxの選択が、端子設定 で TSxx を選択していないものを選択している等が考えられます。ボタンが緑に変わったものは OK です。)

作成を押してください。





1.4. 閾値のチューニング

・マイコンボードと、自己容量タッチキー基板を接続

・マイコンボードとデバッガを接続

・マイコンボードに電源を入れてください

一通りの接続を終えたあと、「チューニングを開始」してください。



チューニングを開始する

e ² パース^	ペクティブ切り替えの確認
\bigcirc	この種類の起動は、中断時に デバッグ パースペクティブが開くように構成されています。
•	このデバッグ・パースペクティブは、アプリケーション・デバッグをサポートするため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。
	このパースペクティブを開きますか?
🔲 常にさ	この設定を使用する(<u>R</u>)
	(はい(Y) いいえ(<u>N</u>)

上記画面が出た場合は、「はい」で問題ありません。



21





自動的に処理が進み、キーに触れた際の変化の計測が始まります。

e ² 自動調整処理中	x
5/9: ボタン (K4, TS17 @ config ターゲットボード上のボタンを指で ださい。キーを押すタイミングの目 K4, TS17 @ config01: 15395	01)の変化量を計測します。 触れながら、キーボードで何かキーを押してく 安は、数値の変動が安定した時点となります。
	キャンセル ヘルプ(<u>H</u>)

上記の表示まで進んだ場合、キー「K4」にタッチし、PCのキーボードを(任意のキー)叩いてください。 その後、3個のキーに関して同様の操作を繰り返します。

※キーの出現順は K4, K3, K2, K1 の順番となりますのでご注意ください





e ² 自動調整処理中						10		×
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/エラー	
	config01	ボタン	К1	TS28	1120			
	config01	ボタン	K2	TS22	907			
	config01	ボタン	K3	TS21	894			
	config01	ボタン	K4	TS17	1099			
リトライ 調整処理の継続 キャンセル ヘルプ(出)								

「調整処理の継続」を押してください。

[参考]

e	2 自動調整処理中					-		×		
1	惑度調整で警告や	ѷオーバ	-70	一等の	のエラーが林	食出され	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									
1	処理の継続」ボタ	タンを押	して処	理を網	迷続してくナ	ぎさい。	,			
	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/エラー		
		config01	ボタン	K1	TS28	1122				
		config01	ボタン	K2	TS22	862				
	✓	config01	ボタン	K3	TS21	65535	_			
	V	config01	ボタン	К4	TS17	65535				
	 リトライ 調整処理の継続									
	キャンセル ヘルプ(圧)									

上記の様に、「閾値」の欄が極端に大きな値(または、小さな値)となっている場合は、キーが押されていなかった事 が考えられます。

この場合、K3, K4の「リトライ対象の選択」のチェックを入れて「リトライ」を行ってください。



23





ファイル生成ボタンを押すと、パラメータファイルが出力されます。

<workspace>/RA4M1_LCD_CTSU_QE/qe_gen/qe_touch_define.h
<workspace>/RA4M1_LCD_CTSU_QE/qe_gen/qe_touch_config.h
<workspace>/RA4M1_LCD_CTSU_QE/qe_gen/qe_touch_config.c

1.6. プログラムの実装



上記 3.実装「例を表示する」を押してください。





e ² サンプルコードの表示			×						
main()関数のコード例:									
/**************************************									
* FILE OF Capacitive Touch Sample c									
* DATE : 2020-03-04									
* DESCRIPTION : Main Program *	1		E						
* NOTE: THIS IS A TYPICAL EXA	AMPLE.								
*	*****	******							
#include "qe_touch_config.h"		1							
#define TOUCH_SCAN_INTERV	AL_EXAMPLE (20) /* millis	econds */							
<pre>void qe_touch_main(void);</pre>									
uint64 t button;									
_ /									
void ge_touch_main(void)									
{									
fsp_err_t err;			-						
•	III		•						
クリップボードにコピー	ファイルに出力	アプリケーションノー	トの表示						
	ОК								
L									

サンプルコードが表示されますので、「ファイルに出力」を押して、その後 OK を押してください。

<Workspace>/RA4M1_LCD_CTSU_QE/qe_gen/qe_touch_sample.c

に、サンプルコードが出力されます。







e' workspace - RA4M1_LCD_CTSU_QE/src/hal_entry.c - e	tudio	- 0 - X -						
ファイル(E) 編集(E) ソース(S) リファクタリング(T) ナ	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>) ヘルプ(<u>H</u>)							
係 た に な た に 、 、 、 、 、 、 、 、 、 、 、 、 、	🖏 🛊 🔳 🎋 デバッグ(6) 💎 🔄 RA4M1_LCD_CTSU_QE Debug 🗸 🌼 🗄 🐂 📓 🕲 🔸 🐐 📓 🖄 🗮 🛸 🔌 🗈 📾 🗮 🕅 🕺 🖉 🔅 🖏 👘 💷 👘 😵 🐇 👘 💷 👘 😵 🔅 🖉							
📸 • 🛍 • 🖻 • 🞯 • 🚸 • 隆 • 🎒 🖨 🔗 •								
	クイック・アクセス 😰 🔤 C/C++ 🏠 デバッグ <captouchモニター (qe)=""> 👼 RA Configuration 🖏 CapTouchモ</captouchモニター>	三夕 RA (QE)						
🎦 プロジェクト・エクスプローラー 🛛 🖓 🗖	② CapTouchメイン/センサ・チューナ RA (QE) 🛛 🔒 *hal_entry.c 🕴 🗟 qe_touch_sample.c	- 8						
🖹 😫 👘 🗢	1 #include "hal_data.h"	~						
▲ 👺 RA4M1_LCD_CTSU_QE [Debug] ^	3 FSP CPP HEADER							
▷ 🖑 バイナリー	<pre>4 void R_BSP_WarmStart(bsp_warm_start_event_t event);</pre>							
Includes	5 FSP_CPP_FOOTER							
⊿ 😕 qe_gen	<pre>void qe_touch_main(void);</pre>							
de_touch_config.c	8							
A qe_touch_config.h	10 # * main() is generated by the RA Configuration editor and is used to generate threads if an RTOS is used. This function 13 # word bal entry(word) I	onL. E						
A ge_touch_define.h	/* TODO: add your own code here */	=						
a ge touch sample.c	15 qe_touch_main();	-						
b 🕞 ra	16 }							
N Cara gen	19 ⊕ * This function is called at various points during the startup process. This implementation uses the event that is∏							
	24							
	25							
Ic nal_entry.c	20 #11 BSP_FEATURE_FLASH_LP_VERSION != 0							
Debug	28 /* Enable reading from data flash. */							
QE-Touch	29 R_FACI_LP->DFLCTL = 1U;							
b > ra_cfg	30							
Script	31 7 Wolld normally nave to wait toshor(ous) for data flash recovery. Flating the endpie nere, before clock and 32 * C runtime initialization, should negate the need for a delay since the initialization will twoically take may	one than						
🔅 configuration.xml	33 #endif							
R7FA4M1AB3CFP.pincfg	34 }							
a cfa.txt		-						
	٠	۲.						

src/hal_entry.c に、下記赤字部分を追加してください。

hal_entry.c(変更例)

```
void qe_touch_main(void);
void hal_entry(void) {
/* TODO: add your own code here */
    qe_touch_main();
}
```

1.7. 変数のモニタリング

ビルド後、デバッグを開始。







r	
e ² workspace - RA4M1_LCD_CTSU_QE/qe_gen/qe_touch_sample	le.c - e ² studio
ファイル(<u>E</u>) 編集(<u>E</u>) ソース(<u>S</u>) リファクタリング(T) 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>) ヘルプ(<u>H</u>)
🔦 🎄 🔳 🗞 デバッグ(B) 🗸 🖻 RA4M1_LCD_C	CTSU_QE Debug 🗸 🌞 🗄 🕶 🔚 🐚 🕸 🗸 🦠 🗙 🕨 💷 🚔 🛠 🧟 🕸 🖗 🖉 🖉 🖉 🖉
🍅 💪 🛷 🔹 🗟 🗊 📝 🐓 🔹 🖓 🔹	クイック・アクセス 🕴 🕼 C/C++ 🎄 デバッグ <captouchモニター (qe)=""></captouchモニター>
🍋 プロジェクト・エクスプローラー 🛛 🕒 😫 💟 🖓 🗖	コ 🖏 CapTouchメイン/センサ・チューナ RA (QE) 🔒 hal_entry.c 📝 qe_touch_sample.c 🛿
 	<pre> 18</pre>

qe_touch_Sample.c の button という変数が、ボタンのタッチ状態の結果が格納される変数です。

e workspace - RA4M1_LCD_CTSU_QE/qe_gen/qe_touch_sample.c - e ² studio						X
ファイル(F) 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)					
🔦 株 🔳 株 デバッグ(B) 🗸 配 RA4M1_LCD_CTSU_QE Debug く 禁 注意 マ 🗟 🐚 👋 マ 🗞 マ 🗟 🗽 🗎	🕨 🗉 🛤 🗷 🖘 .e i+	≅ ≋≣∿ - 🇞 🗰	uu 😭 🖏 🎸 🕸	🐵 🖋 😋 🚸 • 💁 •	2 2 6 1 1	
○ 第 · · · · · · · · · · · · · · · · · ·	クイック・アクセス 📑	職 C/C++ 森 デバッ	グ <captouchモニタ< td=""><td>7— (QE)> 💮 RA Configura</td><td>tion 🖏 CapTouchモニタ R</td><td>JA (QE)</td></captouchモニタ<>	7— (QE)> 💮 RA Configura	tion 🖏 CapTouchモニタ R	JA (QE)
☆デバック 🛛 🦌 🖬 🧔 ▽ 🖓 👔	」 (x)= 変数 💊 ブレークポイン	ト 調 レジスター 🐴 モ	ジュール 🕅 式 🛛 🕯	🗣 イベントポイント 📄 IO I	Registers 🛛 🔒 Peripherals	
 	式 R button 参新しい式を追加	タイプ uint64_t	(直 0	参加 (10) アドレス 0x20000398	 二 中 ※ ※ ご ご ! 名前:button Details:0 デフォルト:0 10 進数:0 16 進:0x0 	¢ ⊋ ⊽ ^
					バイナリー:0 8 道:0	
	•				4	

デバッグビュー 式タブ

新しい式を追加 button

で、button 変数のウォッチが可能です。





(x)= 変数 🍳 ブレークポイント 闘	レジスター 🛋 モジュー	ル 🧐 式 🛛 🥐 イベ	ントポイント 📄 IO Re
			ti 🎫 🗖
式	タイプ	値	アドレス
R button	uint64_t	8	0x20000398
🛖 新しい式を追加			
-			
•	III		•

実行中に、キー"K1"を押し、一時停止ボタンを押すと、キーK1(TS28)に相当する8がモニタできます。

+-	TS	値
K4	TS17	0x1
K3	TS21	0x2
K2	TS22	0x4
K1	TS28	0x8

キーと値の関係は、上記です。本プロジェクトで有効化している、4個のキーに対して表の値が OR 取りされた値 (例えば、キーK4とキーK1に触れた場合、button 変数は 0x9)となります。

1.8. CapTouch のモニタ機能

l						
	Ren	esas Views)実行(R) ウィント	[×] ウ(W)	\sim l	プ(H)	
		C/C++	•	1.8	3. 🤋 .e i> 🗮 📰 🔩 - 🇞 ា 💷 😭	ئە 😭
		Pin Configurator				
		Renesas QE	•	C	<u>CapTouchボード・モニタ RA (QE)</u>	- ¹ 9
		Tracing	•	Q	CapTouchメイン/センサ・チューナ RA (QE)	3
		ソリューション・ツールキット	•	C	CapTouchマルチ・ステータス・チャート RA (QE)	
		デバッグ	•	C	CapTouchパラメータ一覧 RA (QE)	-
		パートナーOS	•	C	CapTouchステータス・チャート RA (QE)	
		ルネサス OS	•	Į\∕×	消費電流測定 (QE)	
	2	Renesas Software Installer	L			_
	_					





・CapTouch ボード・モニタ

RenesasViews - RenesasQE - CapTouch ボード・モニタ



モニタリングを有効にするを押す



キー"K1"と"K3"にタッチしている際、タッチしているキーに指のアイコンが表示されます。



29



・CapTouch マルチ・ステータス・チャート

RenesasViews - RenesasQE - CapTouch マルチ・ステータス・チャート



プルダウンメニューから、適当なキーを選択してください。下記は、キーK1を1:(赤)に割り当てています。

🖏 CapTouchマルチ・ステータス・チャート RA (QE) 🛛	
K1, TS28 @ confit マ 15402 センサ値	4:
17204 16767	
16330	
15893	
15456	
リアルタイムで、センサの値がグラフに表示されます。	タッチ中





・CapTouch ステータス・チャート

RenesasViews - RenesasQE - CapTouch ステータス・チャート

	~
タッチI/F: K1 @ config01 ▼ 🔲 選択状態を同期する	Â
I/F種別: ボタン(自己), チャネル: TS28	=
カウント値: 15450 基準値: 15471 閾値: 1120 差分値: -21	
データ収集の開始	
	Ť
16839	
16373	
10075	
15907	
15441	

マルチ・ステータスチャートに似ていますが、閾値(緑線)や、タッチ判定(画面下の赤バー)等が表示されます。

QE CapTouch では、タッチキー関連のコードの自動生成に加え、パラメータの最適化、値のリアルタイムモニタの 機能があります。ユーザがプログラムコードを書き下すことなく、タッチキー関連の部分のコード生成ができますので、 タッチキーアプリケーション開発時には活用できるかと考えます。





1.9. サンプルプログラム CD に格納されているプロジェクトに関して

サンプルプログラム CD には、QE CapTouch のサンプルプロジェクトが格納されています。e2studio のワークスペースにインポートして使用する事が可能です。

サンプルプロジェクトでは、QE CapTouch で生成したコード(qe_touch_sample.c)に、LCD 画面出力を追加したサンプルとなっています。

キーパッド K1, K2 に触れた場合



キーに触れている間、キー番号の右に(*)が表示されます。(QEを使用しないサンプルプログラムと同様の動作です)

qe_touch_sample.c 内で元々用意されている、タッチ状態の結果が格納される button 変数から、キーの情報を抜き出して表示させたものです。





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	2020.6.30	_	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。



RA4M1-100 LCD タッチキー評価キット 取扱説明書 #

ルネサス エレクトロニクス RA4M1 搭載 HSB シリーズマイコンボード向け評価キット

RA4M1-100 LCD タッチキー評価キット [QE CapTouch 導入編] 取扱説明書

_{株式会社} 北斗電子

©2020 北斗電子 Printed in Japan 2020 年 6 月 30 日改訂 REV.1.0.0.0 (200630)