RX113 タッチキー評価キット
[ソフトウェア編]
マニュアル

-本書を必ずよく読み、ご理解された上でご利用ください

株式会社北斗電子

REV.1.0.0.0

ー目　次ー

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】

1. 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。
4. 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。

概要

本書は、RX113 タッチキー評価キット付属 CD 内のサンプルプログラムについて解説を行うものです。

本書では、「自己容量タイプキーパッド」及び「相互容量タイプキーパッド」の読み出し。電流オフセット値設定のサンプルプログラムを示しています。

本書で解説するサンプルプログラムはキーパッドに触れた場合に触れているキーを検出する部分のエッセンスを抜き出したものとなっています。そのため、単純で短いコードとなっています。

実際に、誤検出を排除した、タッチキーアプリケーションを組む際には、ノイズや経時変化等を考慮する必要がある場合がありますので、お客様のシステムに応じたプログラム設計をお願い致します。

1. 全体構成

1.1. CD フォルダ構成

SOURCE/	HSBRX113F100_Self_cap_sample/	自己容量キー読み込みサンプル
	HSBRX113F100_Mutual_cap_sample/	相互容量キー読み込みサンプル
	HSBRX113F100_Self_cap_offset/	自己容量電流オフセット値補正サンプル
	HSBRX113F100_Mutual_cap_offset/	相互容量電流オフセット値補正サンプル
	HSBRX113F100_Self_cap_appli/	自己容量アプリケーションサンプル
DOCUMENT/	HSBRX113F100_Software_REV_*x*_s.pdf	本マニュアル
	HSBRX113F100_REV_*x*_s.pdf	マイコンボードマニュアル

本 CD に格納されているサンプルプログラムソースは、ルネサスエレクトロニクス社 CS+向けのプロジェクトとなっております。(x は、ファイルのバージョンが入ります)

1.2. タッチキーボードの説明

RX113 タッチキー評価キットは、

・マイコンボード(HSBRX113F100)
・LCD 基板
・自己容量タイプキーパッド
・相互容量タイプキーパッド
・タッチキー保護パネル（アクリル板）

の 4 枚のボード及びタッチキー保護パネルで構成され、「自己容量タイプキーパッド」と「相互容量タイプキーパッド」は、どちらか一方を接続して使用する。

タッチキー保護パネルは、キーパッドの上に重ねて使用してください。
（タッチキー保護パネルは、サイズの大きな相互容量タイプキーパッドに合わせた大きさになっていますが、自己容量タイプキーパッドでも使用可能です）
※相互容量タイプのキーパッドを使用する際には必ず、タッチキー保護パネルを使用してください
（タッチキー保護パネルがない場合は、キーの読み取りが出来ません）
※タッチキー保護パネルとキーパッドの間には隙間が出来ない様、注意願います

1.3. 変数等の説明

プログラムで共通で使用している変数について。

(1)#define 定義変数

CTSU_CH_NUM

測定チャネル数。自己容量タイプでは 12 キーとなっているので、"12"を定義している。相互容量タイプでは、キーが 6x6 の 36 キーとなるが、1 キーで同相と逆相の 2 回測定するので、"72"を定義している。

CTSU_KEY_NUM

相互容量のキーの数。6x6 の"36"を定義している。

CTSU_COLUMN_NUM

相互容量の列数。相互容量キーパッドでは 6 列となっているので"6"を定義している。

CTSU_CTSUSNUM

測定回数。サンプルでは、1 回としているので、"0"を定義している。マイコンレジスタの CTSUSNUM[5:0]に対応。

CTSU_CTSUSO_TS_x [x=0~CTSU_CH_NUM]

センサ ICO 入力電流オフセット値。0-1023 の値を定義する。初期値は"300"としている。デバイスばらつきにより最適値は異なる。センサ ICO 入力電流オフセット値を算出するサンプルプログラムを実行すると、最適値が算出される。マイコンレジスタの CTSUSO[9:0]に対応。

CTSU_CTSUICOG

ゲイン設定値。"0"でゲイン 100%。マイコンレジスタの CTSUICOG[1:0]に対応。

CTSU_CTSUSDPA

ベースクロック分周比。"0"で、2 分周となる。マイコンレジスタの CTSUSDPA[4:0]に対応。

CTSU_CTSURICOA_TS_ x [x=0~CTSU_CH_NUM]

リファレンス ICO 電流調整。オフセット電流調整用のリファレンス ICO に与える電流値を決める。初期値は"63"としている。マイコンレジスタの CTSURICOA[7:0]に対応。

(2)グローバル変数

unsigned short sens[CTSU_CH_NUM]

測定値格納変数。キーの容量によって変わる測定値を格納している。

unsigned short ref[CTSU_CH_NUM]

リファレンス測定値格納変数。

long initial_sens[CTSU_CH_NUM]

初期値格納変数。プログラム起動時に、キーに触っていないときの初期値を拾い、後にキーに触れた場合との差分を見るために使用。

unsigned short base_diff[CTSU_KEY_NUM];

相互容量タイプで、2 回の測定値の差分の初期値（パッドに触れていないときの値）

unsigned short diff[CTSU_KEY_NUM];

相互容量タイプで、2 回の測定値の差分

unsigned char status[CTSU_CH_NUM]

測定時のステータス格納変数。

unsigned short ctsu_ctsuso0[CTSU_CH_NUM]

マイコンレジスタ CTSUSO0 設定変数。

unsigned short ctsu_ctsuso1[CTSU_CH_NUM]

マイコンレジスタ CTSUSO1 設定変数。

unsigned short data_index

測定チャネルを示す変数。

unsigned short err

測定エラーを格納する変数。マイコンレジスタ CTSUERRS の結果が格納される。

unsigned char key

そのキーが押されているかを示す変数。

unsigned char in_measure, result_update, initialize_finished

フラグ変数。

2. 自己容量キー読み出しサンプル

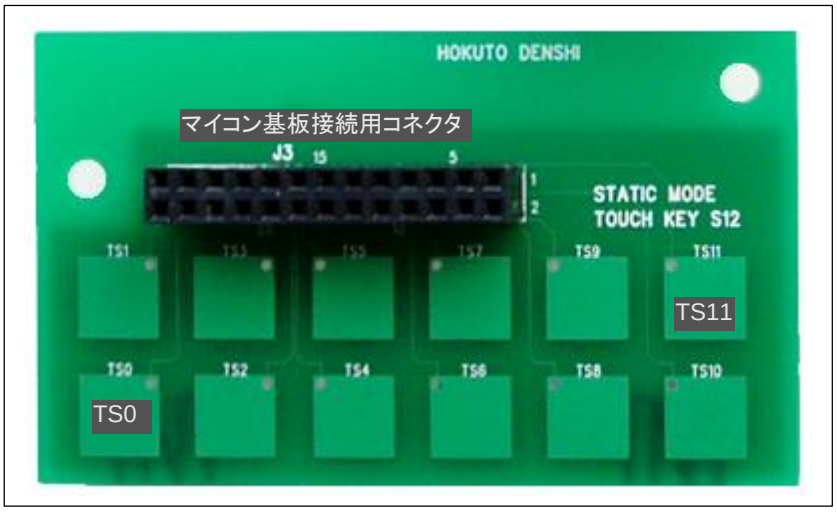
2.1. ソースツリー構成

HSBRX113F100_Self_cap_sample/ 以下

コード生成	r_cg_main.c	メイン関数
ctsu	ctsu.h	CH 数等の定義ファイル
	ctsu_selfcap.c	自己容量キー読み出し
	ctsu_intr.h	割込みプログラムヘッダ
	ctsu_intr.c	割込みプログラム
	ctsu_offset_value.h	電流オフセット値定義ファイル

ソースで主要なものを上表に示す。

2.2. 自己容量キーパッド



自己容量タイプはキーパッドが 2 行 6 列の合計 12 個並んでいる。それぞれ、マイコンの TS0～TS11 に接続されている。検出は、パッドを手で触れた際、それぞれの PAD に見える容量値が変化することから、どのパッドに触れたかを判断する。

2.3. 関数

(1)ctsu_selfcap.c 内で定義

void ctsu_init(void)

引数:なし
戻り値:なし

タッチセンサ初期化。各種マイコンレジスタの設定。

void ctsu_initial_value(void)

引数:なし
戻り値:なし

センサ測定値の初期値を求める関数。キーにタッチしていないときに実行する必要がある。
キーの測定を256回行い、測定値の平均をinitial_sens[ ]変数に格納する。

void ctsu_set_param(void)

引数:なし
戻り値:なし

マイコンのCTSUSSC, CTSUSO0, CTSUSO1レジスタにそれぞれ、0, ctsu_ctsuso0[ ],ctsu_ctsuso1[ ]に格納されている値を転送する。

void ctsu_measure(void)

引数:なし
戻り値:なし

測定を開始する。

void ctsu_read(void)

引数:なし
戻り値:なし

センサ測定結果をsens[ ]、リファレンス測定値をref[ ]、測定ステータスをstatus[ ]、エラーをerr変数に格納する。

unsigned char ctsu_result(void)

引数:なし
戻り値:押されているキー番号

(2)ctsu_intr.c 内で定義

void int_ctsuwr(void)

引数:なし
戻り値:なし

チャネル測定前に入る割込み関数。

void int_ctsurd(void)

引数:なし
戻り値:なし

チャネル測定後に入る割込み関数。

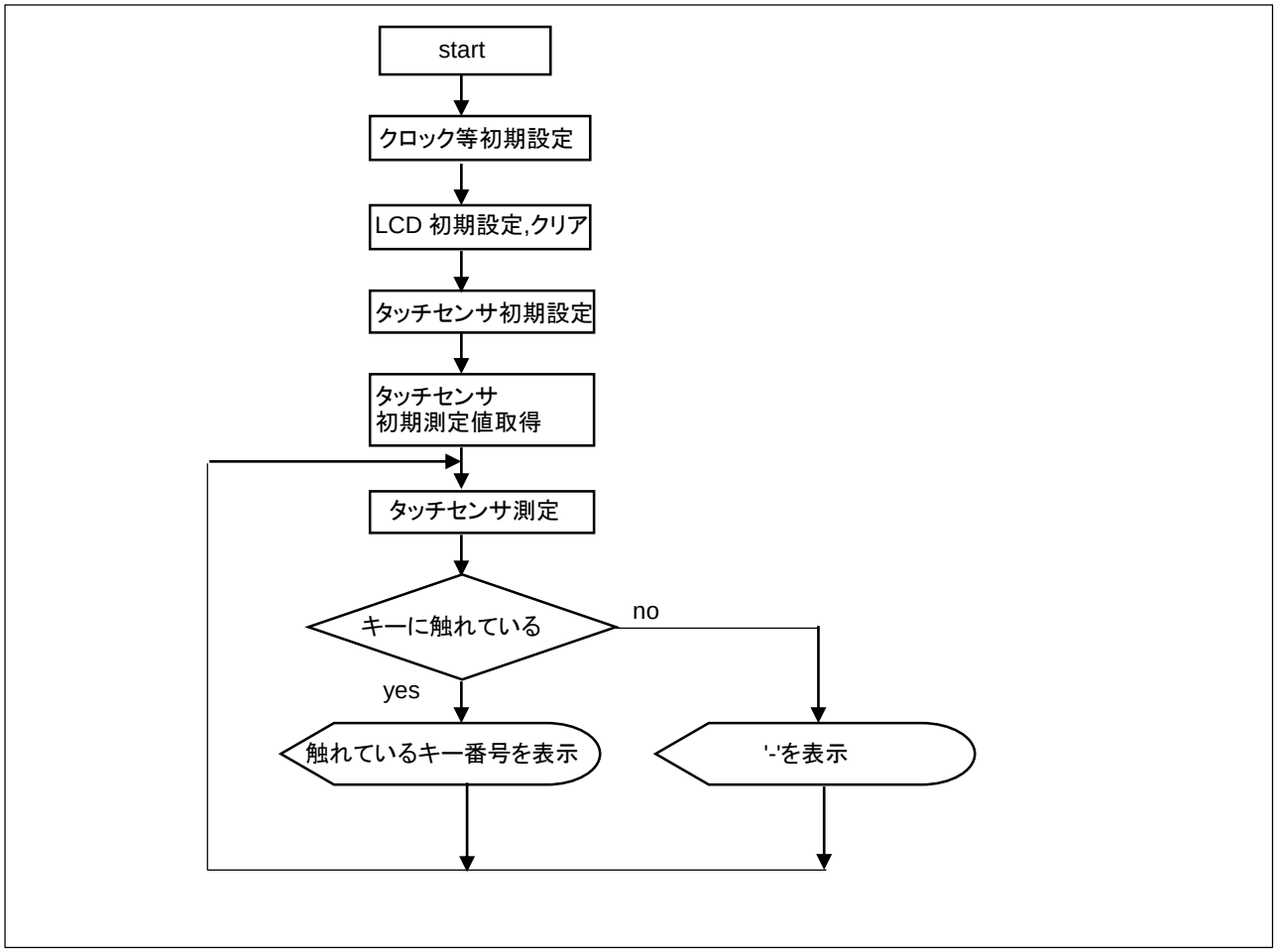
void int_ctsufn(void)

引数:なし
戻り値:なし

全チャネルの測定終了時に入る割込み関数。

2.4. プログラム説明

2.4.1. フローチャート



タッチセンサの測定値は、パッドの容量によって変化する。パッド毎にパッドに触れていないときの初期容量値は微妙に異なるため、「タッチセンサ初期測定値取得」時に、パッドに触れていない時の測定値（256 回測定の平均値）を取得しておく。

「タッチセンサ測定」時の測定値を見て、

・一定の変化量があるか（本プログラムでは、20%以上値が増加したか）

・一番変化の大きなキーはどれか（タッチの位置が微妙にずれている場合触れているパッド以外の測定値も増加するため、一番大きく変化した測定値のキーが押されていると判断する）

という基準で触れているパッドを判定する。

2.4.2. 実際の測定値の変化例

	initial_sens	sens(1)	sens(2)
TS0	3798	4049	4131
TS1	2599	2664	2728
TS2	3355	3384	**4670**
TS3	2429	2506	2502
TS4	3297	3371	3520
TS5	2341	2344	2416
TS6	3363	3411	3480
TS7	2705	2768	2841
TS8	3931	4006	4049
TS9	3223	3265	3354
TS10	4283	4337	4403
TS11	4186	4280	4371

initial_sens が初期に取得した値。sens(1)がパッドに触れていないときの測定値。sens(2)が TS2 のパッドに触れたときの測定値。パッドに触れた際測定値が大きく変化するため、プログラムでの判別は容易である。

3. 相互容量キー読み出しサンプル

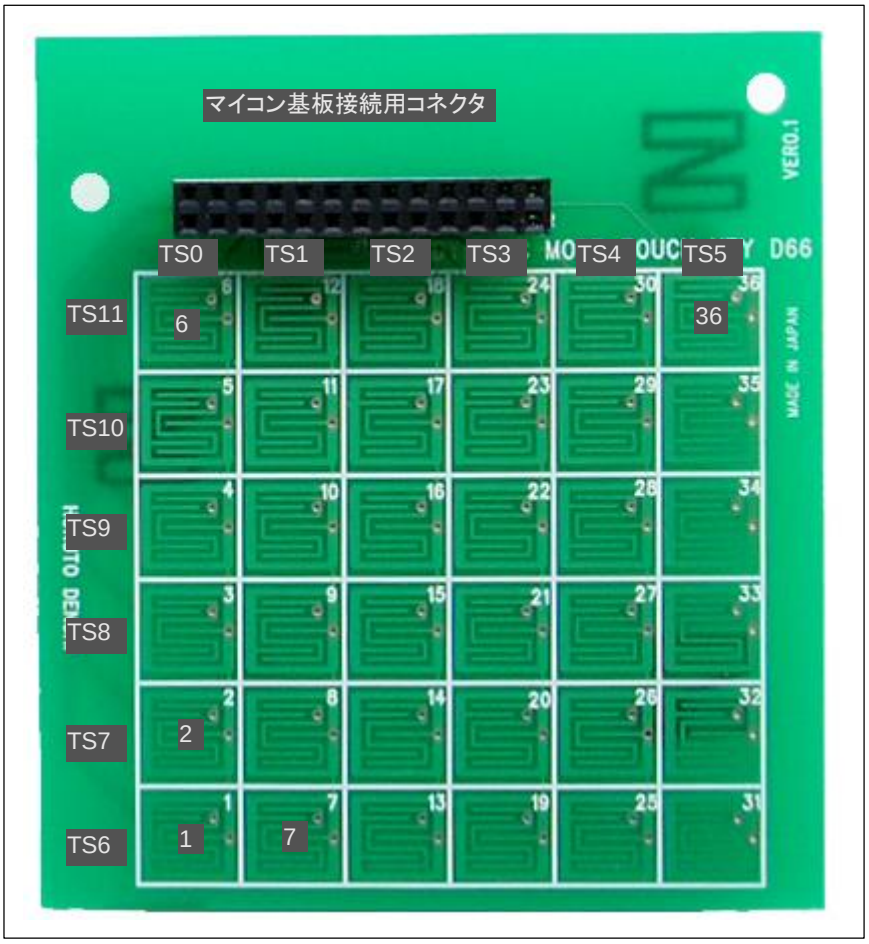
3.1. ソースツリー構成

HSBRX113F100_Mutual_cap_sample/　以下

コード生成	r_cg_main.c	メイン関数
ctsu	ctsu.h	CH 数等の定義ファイル
	ctsu_mutual.c	相互容量キー読み出し
	ctsu_intr.h	割込みプログラムヘッダ
	ctsu_intr.c	割込みプログラム
	ctsu_offset_value.h	電流オフセット値定義ファイル

ソースで主要なものを上表に示す。

3.2. 相互容量キーパッド



相互容量キーパッドは、6x6 のマトリックスで構成されており、列側がマイコン TS0~TS5 端子、行側が TS6~TS11 に接続されている。プログラムでは、TS0~TS5 出力、TS6~TS11 を入力に設定する必要がある。(なお、相互容量測定モードでは、入力に設定した TS6~TS11 端子も測定時には信号が出力される)

3.3. 関数

関数の構成に関しては、基本的に自己容量タイプと同一構成となっている。相違のあるもののみ示す。

(1)ctsu_mutualcap.c 内で定義

void ctsu_row_column(unsigned char num, unsigned char *result)

引数:

num diffのインデックス値

result インデックス値をrow(列番号), column(行番号)に変換した値(関数内で書き換え)

戻り値:なし

キーのインデックス値を、列番号、行番号に変換する関数。

3.4. プログラム説明

3.4.1. フローチャート

全体フローチャートは 2.4 自己容量タイプと同一である。

3.4.2. 実際の測定値の変化例

	初期値	sens(1)	sens(2)
diff_0	4512	4504	4147
diff_1	3956	3940	3816
diff_2	3952	3921	3817
diff_3	3955	3955	3838
diff_4	5128	5099	4971
diff_5	5957	5607	5481
diff_6	4464	4456	**3767**
diff_7	3851	3836	3695
…	…	…	…
diff_35	5273	5250	5159

測定は 1 キーパッド 2 回(同相、逆相)ずつ、合計 72 回行われる。上記表は、同相ー逆相の差分の値(パッドの容量値相当の値)である。手でキーにタッチすると、信号線間の容量値が減少するため、初期値に対して値が減少する。

i 初期値が初期に取得した値。sens(1)がパッドに触れていないときの測定値。sens(2)がパッド 2(TS0-TS6 交点)のに触れたときの測定値となっている。

ここで、パッド 2(diff_6)に触れた sens(2)の、測定値が他の値に関して減少している事が見て取れる。

この値の減少幅が一番大きなキーをプログラムで拾い、LCD 画面に表示している。(全てのキーの減少幅が 5%未満の場合は、どのキーもタッチされていないと判断)

4. 電流オフセット値設定サンプル

4.1. ソースツリー構成

HSBRX113F100_Self_cap_offset 以下　　…自己容量タイプ
HSBRX113F100_Mutual_cap_offset/ 以下　…相互容量タイプ

コード生成	r_cg_main.c	メイン関数
ctsu	ctsu.h	CH 数等の定義ファイル
	ctsu_selfcap.c ctsu_mutual_cap.c	メイン処理 [自己容量タイプ] メイン処理 [相互容量タイプ]
	ctsu_intr.h	割込みプログラムヘッダ
	ctsu_intr.c	割込みプログラム
	ctsu_offset_value.h	電流オフセット値定義ファイル

ソースで主要なものを上表に示す。

4.2. 関数

関数定義で初出のものを示す。

(1)ctsu_selfcap.c/ ctsu_mutual_cap.c 内で定義

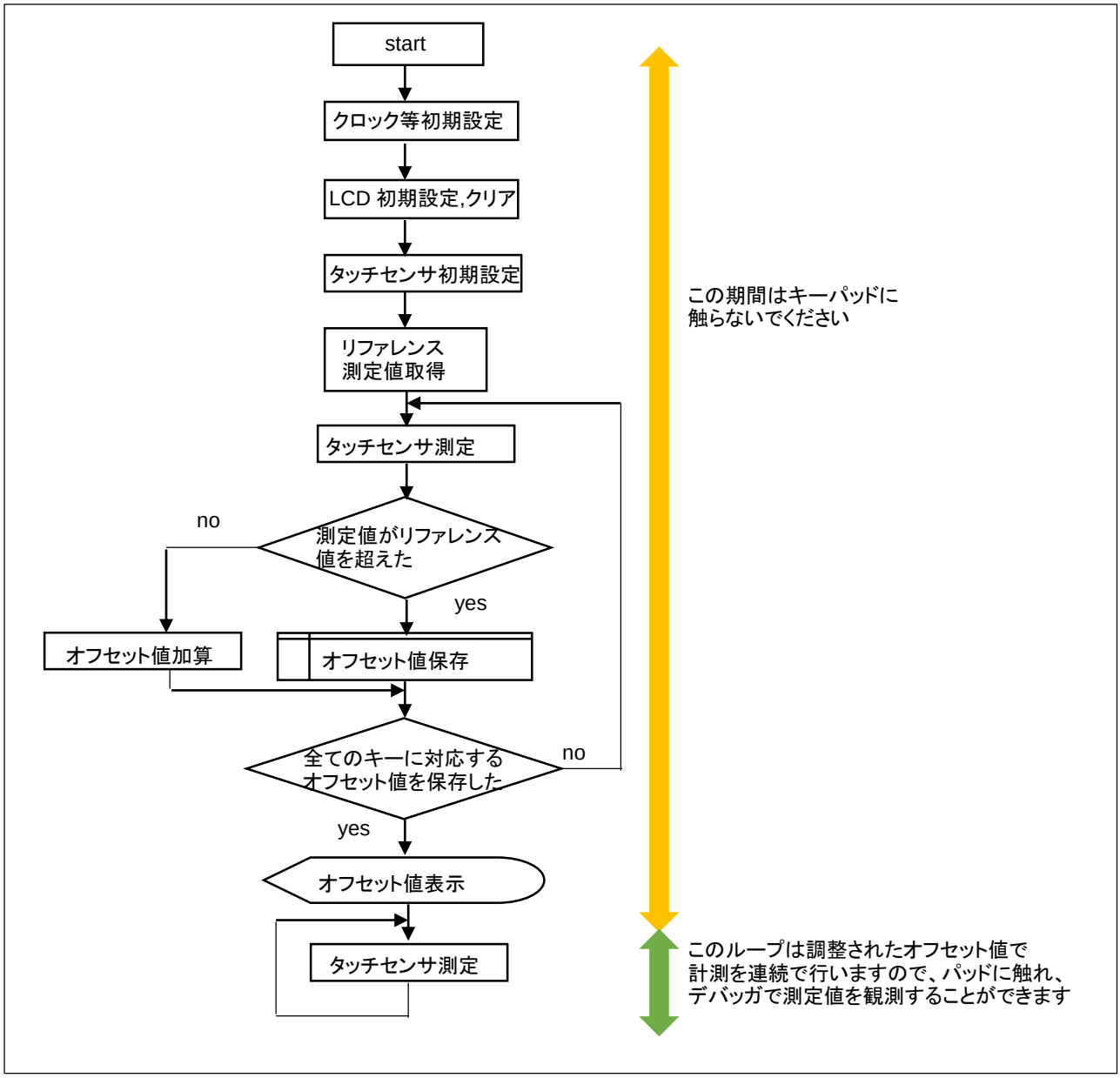
void ctsu_offset_value(void)

引数:なし
戻り値:なし

電流オフセットの最適値を ctsuso_val[ ] 変数に格納する。

4.3. プログラム説明

4.3.1. フローチャート



電流オフセット値の調整は、マイコンの電流オフセット CTSUSO0 レジスタを加算（本プログラムでは 10 刻み）して、リファレンス値を越えた際、リファレンス値を超える 1 つ前の値を、調整されたオフセット値として保存する。、

電流オフセット値は、マイコンチップ毎（チップの製造ばらつきに依存）に最適値が異なる値で、マイコンチップ毎に最適値を求める必要があります。前出のキー読み出しサンプルでは、電流オフセットの初期値として 200(/1023)を設定してありますが、本プログラムで最適値を検索すると、動作マージンが広がる方向になると考えます。

プログラム実行中にエラー(err=0x8000)となった場合、電流オフセットの設定値が大きすぎてオーバフローしていますので、タッチキーパッドとの接続が正しいか等確認してください。

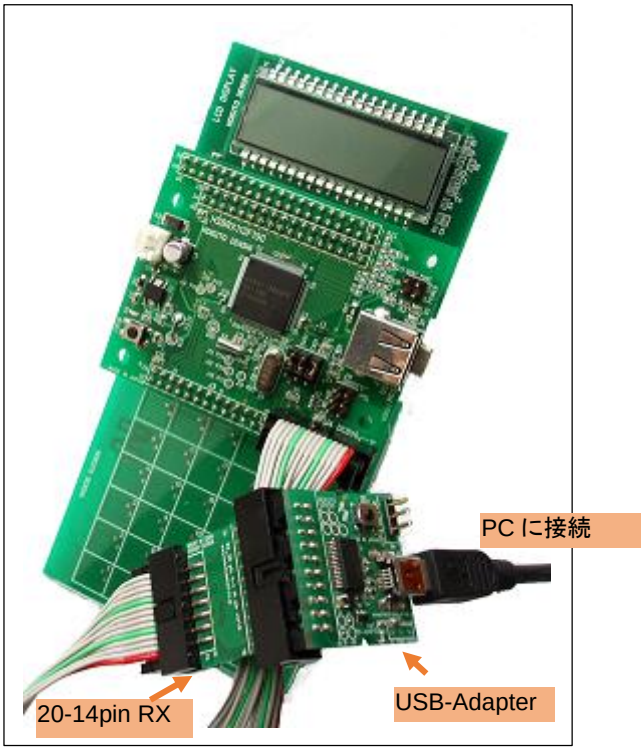
4.4. 出力例

```
//HSBRX113F100 Self capacitance offset value calc program.//
//Copyright (C) 2015 HokutoDenshi. All Rights Reserved.
//
//CTSU_CTSUSO(cuttent offset value) optimization result//
#define CTSU_CTSUSO_TS_0  (0x0186)
#define CTSU_CTSUSO_TS_1  (0x014A)
#define CTSU_CTSUSO_TS_2  (0x0168)
#define CTSU_CTSUSO_TS_3  (0x014A)
#define CTSU_CTSUSO_TS_4  (0x015E)
#define CTSU_CTSUSO_TS_5  (0x014A)
#define CTSU_CTSUSO_TS_6  (0x0168)
#define CTSU_CTSUSO_TS_7  (0x0154)
#define CTSU_CTSUSO_TS_8  (0x017C)
#define CTSU_CTSUSO_TS_9  (0x015E)
#define CTSU_CTSUSO_TS_10  (0x0186)
#define CTSU_CTSUSO_TS_11  (0x0186)
//--end of CTSU_CTSUSO
```

SCI0(RXD:P15, TXD:P16)にシリアルで出力する。
[38400bps, 8bit, パリティなし, ストップビット 1bit]

上記フォーマットは、ctsu_offset_value.h 内のオフセット値の定義と同じフォーマットとなっているので、シリアル端末から、ctsu_offset_value.h 内の当該定義箇所にコピー＆ペーストを行う事ができる。
なお、プログラム上は、ctsuso_val[ ]変数にオフセット値の調整結果が入っているので、デバッガで値をモニタしても同じである。

〇接続例



弊社のオプションを使用する場合は、「20-14pin RX」及び「USB-Adapter」を使用して PC と接続可能です。
市販の USB シリアル変換を用いる事も可能です。その場合は、ハードウェア編のマニュアルでピン番号等を参照して接続願います。

5. アプリケーション例サンプル

5.1. ソースツリー構成

HSBRX113F100_Self_cap_appli　以下　　…自己容量タイプアプリケーションサンプル

コード生成	r_cg_main.c	メイン関数
	r_cg_rtc_user.c	割込みでキー読み込みを処理するメインルーチン
ctsu	ctsu.h	CH 数等の定義ファイル
	ctsu_selfcap.c	測定プログラム
	ctsu_intr.h	割込みプログラムヘッダ
	ctsu_intr.c	割込みプログラム
	ctsu_offset_value.h	電流オフセット値定義ファイル

ソースで主要なものを上表に示す。

5.2. 関数

関数定義で初出のものを示す。

(1) r_cg_rtc_user.c 内で定義

void r_rtc_prd_interrupt (void)

引数:なし
戻り値:なし

タイマ割込み(1/64秒毎)に呼び出され、タッチキーの測定及びメインの処理を行う。

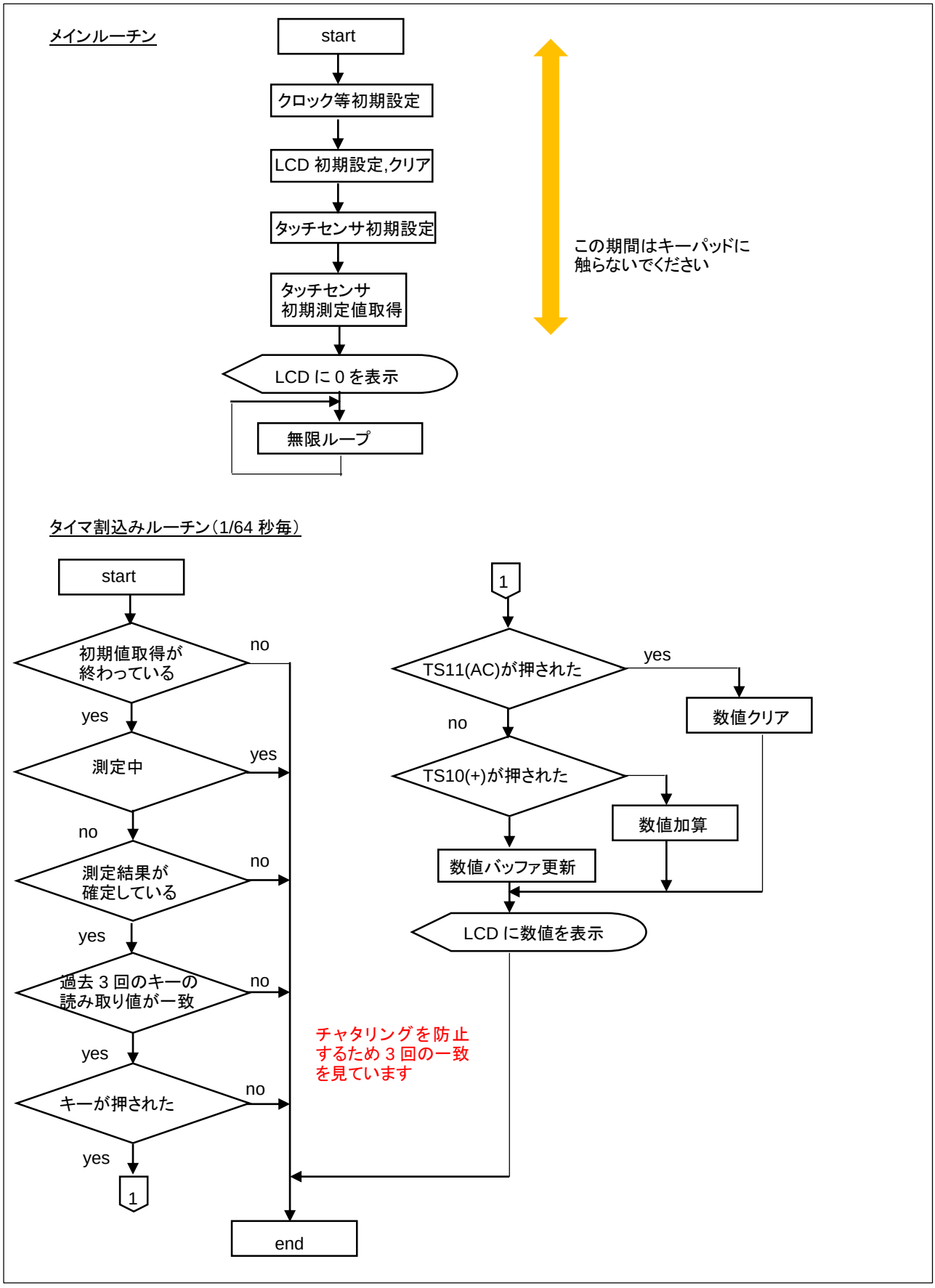
5.3. アプリケーション動作

加算を行う電卓のアプリケーションとなっている。
自己容量キーパッドを接続し、以下のキーで操作を行う。
TS0-9 : 0-9の数値入力
TS10: +入力
TS11: AC(オールクリア)

TS10(+)を押した時点で、現在までの入力加算値を LCD に表示する。数値の入力の終了時に押す。

表示で最上位の'(シングルクオート)が点滅した場合は、計算結果がオーバフローした事を示す。

5.3.1. フローチャート


メインルーチン
start
クロック等初期設定
LCD 初期設定,クリア
タッチセンサ初期設定
タッチセンサ
初期測定値取得
LCD に 0 を表示
無限ループ
この期間はキーパッドに
触らないでください
タイマ割込みルーチン（1/64 秒毎）
start
初期値取得が
終わっている
no
yes
測定中
yes
no
測定結果が
確定している
no
yes
過去 3 回のキーの
読み取り値が一致
no
yes
キーが押された
no
yes
1
end
チャタリングを防止
するため 3 回の一致
を見ています
1
TS11(AC)が押された
yes
数値クリア
no
TS10(+)が押された
数値加算
数値バッファ更新
LCD に数値を表示

6. LCD 表示サンプルプログラム

6.1. ソースツリー構成

LCD	lcd.c	ソースファイル
	lcd.h	ヘッダファイル

6.2. 関数

void lcd_init (void)

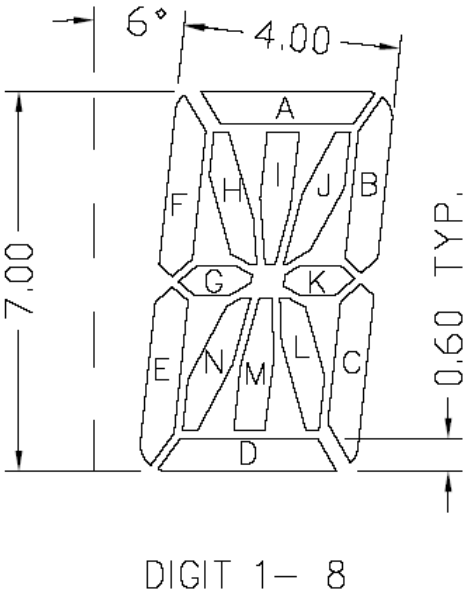
引数:なし
戻り値:なし

LCDの初期化を行う。

unsigned char lcd_ctrl(unsigned char column, unsigned char chara, unsigned char flag)

引数:
column ： 表示する桁を指定 1-8(1:最上位桁, 8:最下位桁)
chara ： 表示するセグメントを指定 'A'-'N'のセグメントと'q'(':クオート), 'd'(.:ドット)
flag ： フラグを指定 0:指定されたセグメントを消灯, 1:点灯, 2:点滅

戻り値:なし
0： 引数OK
1： 引数NG



LCDのセグメント単位での、点灯、消灯、点滅を制御する。

使用例:

lcd_ctrl(1, 'A', 1);

lcd_ctrl(1, 'B', 1);

lcd_ctrl(1, 'C', 1);

1 桁目（左詰）の、A, B, C のセグメントを点灯。（数値の 7 を表示）

unsigned char lcd_clear_disp(unsigned char flag)

引数:

flag: フラグを指定 0:全セグメントを消灯, 1:点灯, 2:点滅

戻り値:

0 : 引数OK

1 : 引数 NG

LCD のクリアを行う。

使用例:

lcd_clear_disp(0); //LCD 消灯

lcd_clear_disp(1); //LCD 全点灯

unsigned char lcd_disp_val(unsigned char column, unsigned char num)

引数:

column: 表示する桁を指定 1-8（1:最上位桁, 8:最下位桁）

num: 表示する数値を指定 0-9

'.' 'd': . を表示

'' 'q': ' を表示

'+' : + を表示

'-' : - を表示

'*' : * を表示

'/' 's': / を表示

'b' '¥': ¥（バックスラッシュ）を表示

'x' : x を表示

'<' : < を表示

'>' : > を表示

'|' : | を表示

戻り値:

0：引数OK
1：引数NG

1桁の数値、記号を表示する。

使用例:
lcd_disp_val(1, 5);　//1 桁目（左詰）に 5 を表示する
lcd_disp_val(1, '+');　//1 桁目（左詰）に+を表示する

unsigned char lcd_disp(unsigned long num, unsigned char fillzero)

引数:
num: 表示する数値（符号なし）
fillzero: 数値の0埋めフラグ 0:0埋めを行わない, 1:先頭の0埋めを行う

戻り値:
0：引数OK
1：引数NG

数値を表示する。

使用例:
lcd_disp(123, 0)　//数値 123 を表示

7. 付録

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	2015.11.17	—	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 北斗電子

〒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

ルネサス エレクトロニクス RX113(QFP-100 ピン)搭載
HSB シリーズマイコンボード

RX113 タッチキー評価キット[ソフトウェア編] マニュアル

株式会社 北斗電子

 2015 年 11 月 17 日改訂 REV.1.0.0.0 (151117)