EPSON
EXCEED YOUR VISION

# S1C17W22/S1C17W23
# タッチキー
# アプリケーションノート

セイコーエプソン株式会社

Rev.1.0

評価ボード・キット、開発ツールご使用上の注意事項

1. 本評価ボード・キット、開発ツールは、お客様での技術的評価、動作の確認および開発のみに用いられることを想定し設計されています。それらの技術評価・開発等の目的以外には使用しないで下さい。本品は、完成品に対する設計品質に適合していません。

2. 本評価ボード・キット、開発ツールは、電子エンジニア向けであり、消費者向け製品ではありません。お客様において、適切な使用と安全に配慮願います。弊社は、本品を用いることで発生する損害や火災に対し、いかなる責も負いかねます。通常の使用においても、異常がある場合は使用を中止して下さい。

3. 本評価ボード・キット、開発ツールに用いられる部品は、予告無く変更されることがあります。

本資料のご使用につきましては、次の点にご留意願います。

本資料の内容については、予告無く変更することがあります。

1. 本資料の一部、または全部を弊社に無断で転載、または、複製など他の目的に使用することは堅くお断りいたします。

2. 本資料に掲載される応用回路、プログラム、使用方法等はあくまでも参考情報であり、これらに起因する第三者の知的財産権およびその他の権利侵害あるいは損害の発生に対し、弊社はいかなる保証を行うものではありません。また、本資料によって第三者または弊社の知的財産権およびその他の権利の実施権の許諾を行うものではありません。

3. 特性値の数値の大小は、数直線上の大小関係で表しています。

4. 製品および弊社が提供する技術を輸出等するにあたっては「外国為替および外国貿易法」を遵守し、当該法令の定める手続きが必要です。大量破壊兵器の開発等およびその他の軍事用途に使用する目的をもって製品および弊社が提供する技術を費消、再販売または輸出等しないでください。

5. 本資料に掲載されている製品は、生命維持装置その他、きわめて高い信頼性が要求される用途を前提としていません。よって、弊社は本（当該）製品をこれらの用途に用いた場合のいかなる責任についても負いかねます。

6. 本資料に掲載されている会社名、商品名は、各社の商標または登録商標です。

# 要旨

本資料は、S1C17W22 または S1C17W23 にて、入出力ポート（PPORT）、ユニバーサルポートマルチプレクサ（UPMUX）、16 ビット PWM タイマ（T16B）を用いてタッチキーの容量変化を検知し、16 ビットタイマ（T16）で一定時間ごと繰り返しタッチキーの状態をスキャンして、その容量相当値を UART で送信したり、キーの状態を LCD に表示させたりするための参考資料です。

# 動作環境

- S5U1C17W23T（以下 SVT17W23: Software Evaluation Tool for S1C17W23）
  ICDmini との接続には専用ケーブル（4pin－4pin）2 種が必要です
- ICDmini（S5U1C17001H）
  パソコンとの接続には USB ケーブルが必要です
- パソコン
  ・GNU17（S5U1C17001C）開発ツールインストール済み ※
  ・ICDmini USB ドライバインストール済み
- 最新版の FLS17W22（ファイル名： fls17w22.elf）/FLS17W23（ファイル名： fls17w23.elf）
  内蔵 FLASH へのプログラムの書き込みに必須
- S1C17W22/S1C17W23 タッチキー用プログラミングパッケージ（本パッケージ）
  ・タッチキーの容量相当値の時間ごと変化の視覚化 プログラミングパッケージ
  （s1c17w22_w23_touch_oscillo）、VBA マクロを含む Excel ファイル（MeasTouch.xlsm）、
  Active-X コントロールファイル（NonComSck.ocx）
  ・タッチキーをスキャンしてその状態を LCD 表示 プログラミングパッケージ
  （s1c17w22_w23_touch_demo）

※ 本パッケージは、GNU17 V2.3.0 で動作確認しています。

目　次

# 1. 仕様

本アプリケーションノートでは、S1C17W22/S1C17W23 に内蔵されている入出力ポート（PPORT）、ユニバーサルポートマルチプレクサ（UPMUX）、16 ビット PWM タイマ（T16B）を用いてタッチキーの容量変化を検知します。

サンプルプログラムのうち、s1c17w22_w23_touch_oscillo は、VBA マクロを含む Excel と連携して、一定の周期でタッチキーの容量で変わるカウンタ値（容量相当値）をサンプリングし、UART を用いて PC に送信して Excel の表に書き込みます。Excel の表に書かれた配列の値はグラフ化されるため、時間ごと容量相当値の変化を視覚化することができます。

またサンプルプログラムのうち、s1c17w22_w23_touch_demo は、16 ビットタイマ（T16）で一定時間ごと繰り返しタッチキーの状態をスキャンして、その状態を LCD に表示します。

# 2. 使用機能説明

| | |
|---|---|
| UPMUX | タッチキー端子を PPORT 出力／T16B Ch.0 のキャプチャ入力 に切り替えるのに使用します。 |
| PPORT | P0[7:0]、P1[2:0]、P1[7:5]、P2[1:0] 計 16 ポートをタッチキー用に割り当て、”L”出力／”H”出力／Hi-Z の何れかに設定します。 |
| T16B Ch.0 | キャプチャーモードのカウント値から、タッチキーの容量相当値を得るために使用します。 |
| T16 Ch.0 | タッチキーを定期的にスキャンするためのインターバルタイマ。 |

以下は、s1c17w22_w23_touch_oscillo の場合です。

| | |
|---|---|
| UART Ch.0 | UPMUX で PPORT の P36 を USIN0 に、P37 を USOUT0 に割り当てます。PC 上で実行する VBA マクロを含む Excel との通信に使用します。 |

以下は、s1c17w22_w23_touch_demo の場合です。

| | |
|---|---|
| LCD24 | タッチキーの状態を LCD に表示するために使用します。 |
| SNDA | タッチキーのクリック音を出力するために使用します。 |

| | |
|---|---|
| 動作モード | T16B Ch.0： キャプチャ 0 割り込み、キャプチャ 1 割り込み機能を用いて、タッチキー端子の容量相当値をカウントします。 |
| システムクロック | システムクロックは OSC3（内蔵発振 4MHz）を使用します。<br>s1c17w22_w23_touch_demo は OSC1（32.768kHz）も使用します。 |
| 割り込み | T16B Ch.0 のベクタ番号とベクタアドレスは以下の通りです。<br>T16B Ch.0　ベクタ番号： 14（0x0e）<br>　　ベクタアドレス： 0x8038<br>本サンプルでは、次の 2 種類の割り込みを使用します。<br>　　キャプチャ 0 割り込み<br>　　キャプチャ 1 割り込み<br><br>T16 Ch.0 のベクタ番号とベクタアドレスは以下の通りです。<br>T16 Ch.0　ベクタ番号： 9（0x09）<br>　　ベクタアドレス： 0x8024<br>本サンプルでは、次の割り込みを使用します。<br>　　アンダーフロー割り込み<br><br>以下は、s1c17w22_w23_touch_oscillo の場合。<br>UART Ch.0 のベクタ番号とベクタアドレスは以下の通りです。<br>UART Ch.0　ベクタ番号： 10（0x0a）<br>　　ベクタアドレス： 0x8028<br>本サンプルでは、次の割り込みを使用します。<br>　　受信バッファ 1 バイトフル割り込み |

図 2-1 に UPMUX の構成を、図 2-2 に T16B の構成を示します。

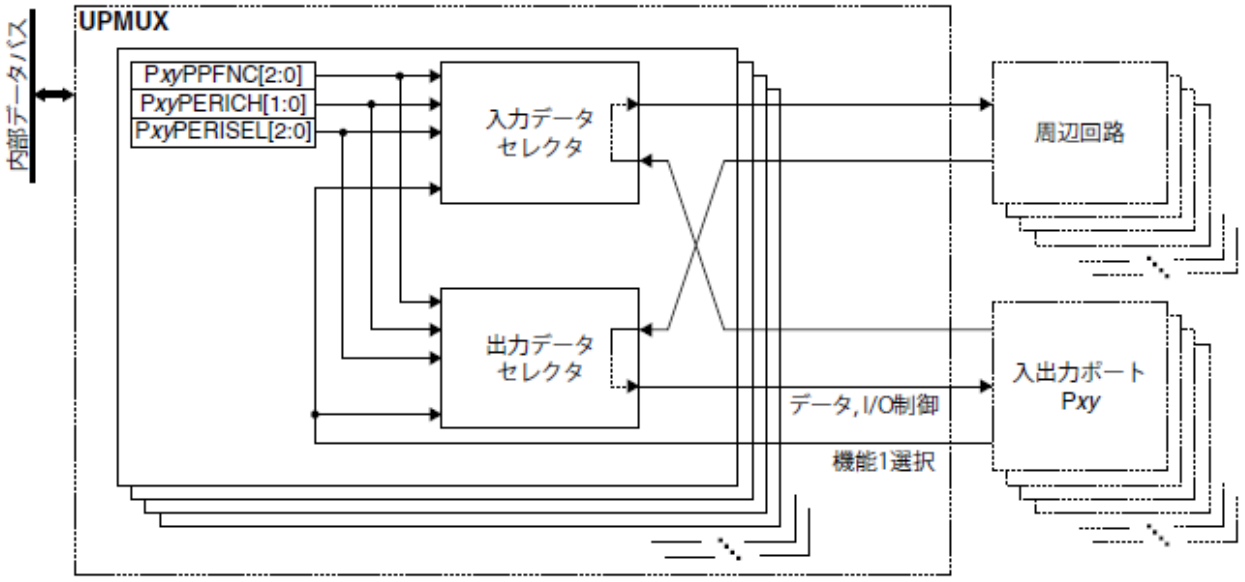



図 2-1　UPMUX の構成

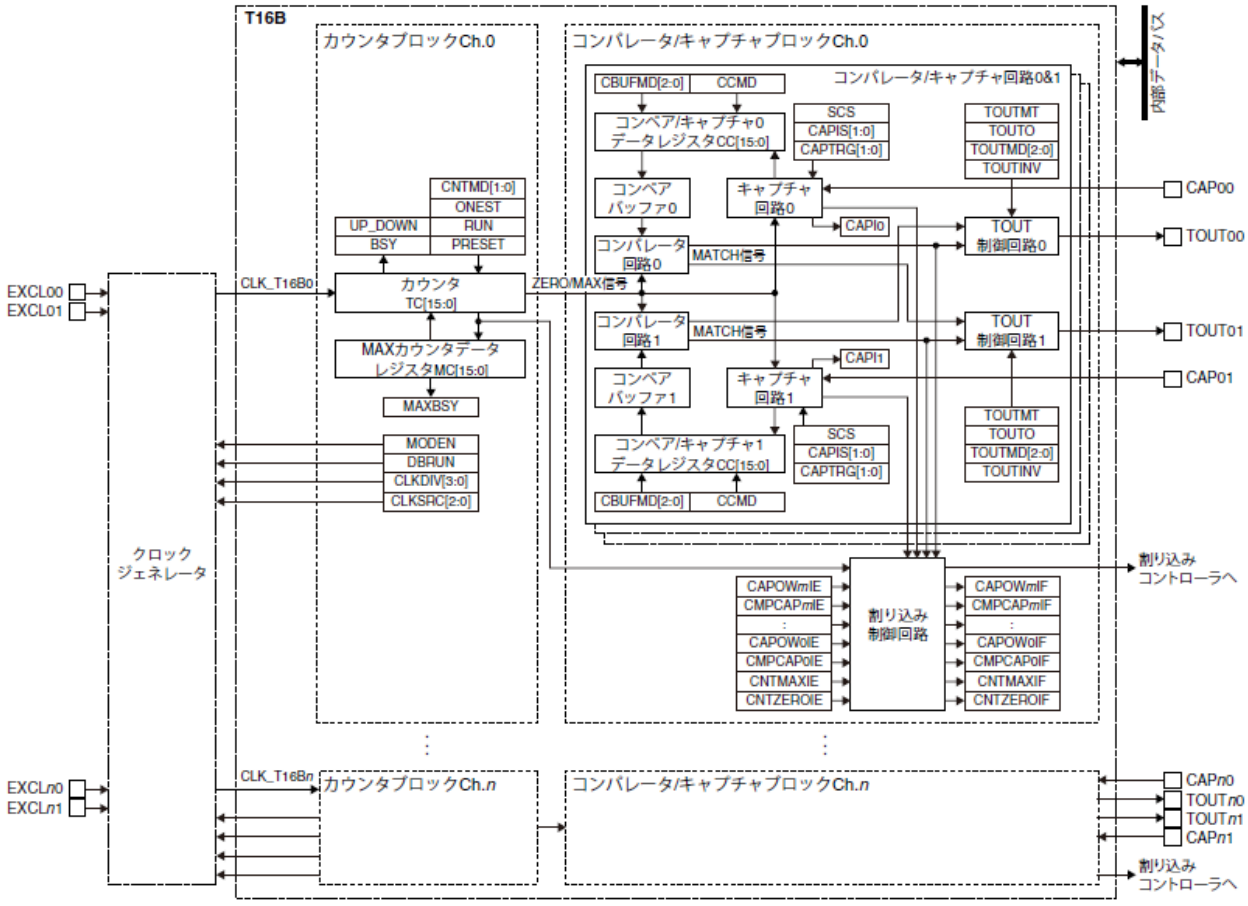



図 2-2　T16B の構成

# 3. 動作原理

## 3.1 検出原理

空気やプラスチックなどの絶縁体（誘電体）で隔離された状態で複数の導体を設置すると、それら導体間には一定の容量値のコンデンサが形成されます。例えば図 3-1 のような構造でプリント基板上にセンサーパッド用の電極とグランドパターンを設けると、C1 と C2 の 2 つのコンデンサが形成されるため、両コンデンサを並列接続した（C1+C2）の容量値がセンサーパッド用の電極とグランドパターンの間で観察されることになります。

一方、人体は電気を通す良導体であり、また静電気を帯びることでもわかる通り、たとえ靴など絶縁性の高いものでグランドと隔離されても人体とグランドの間には大きな容量があるため、先のセンサーパッドに近づくことで近接効果を起こすことができます。具体的には、センサーパッド上面の誘電体の表面に指を触れることでセンサーパッドに新たに容量 C3 が形成され、センサーパッドの容量は（C1+C2+C3）と増大することになります。

この容量値の変化を検出することで、タッチセンサを作ることができます。

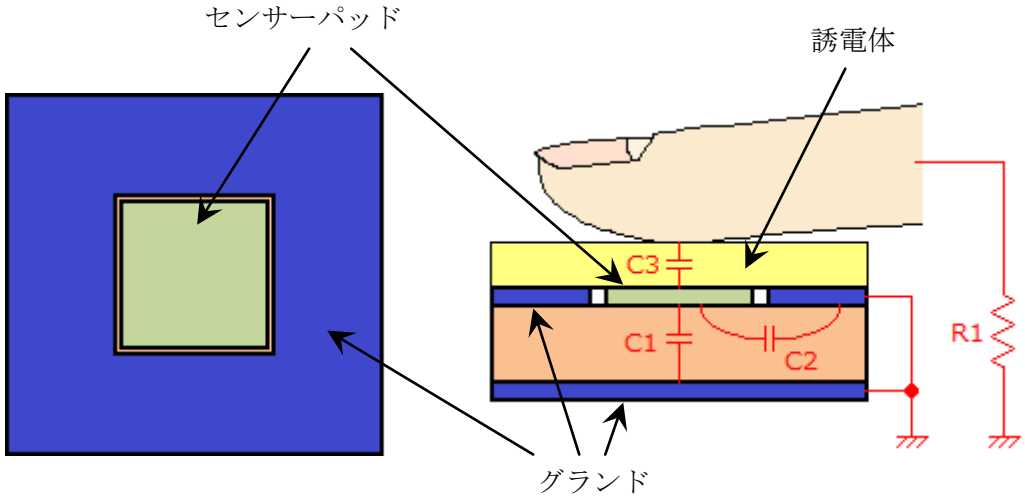



図 3-1　センサーパッドの容量（上面図・断面図）

## 3.2 検出方法

容量の検出方法を、図 3-2 の容量検出回路の概念図と、図 3-3 の検出波形の時間変化を用いて説明します。

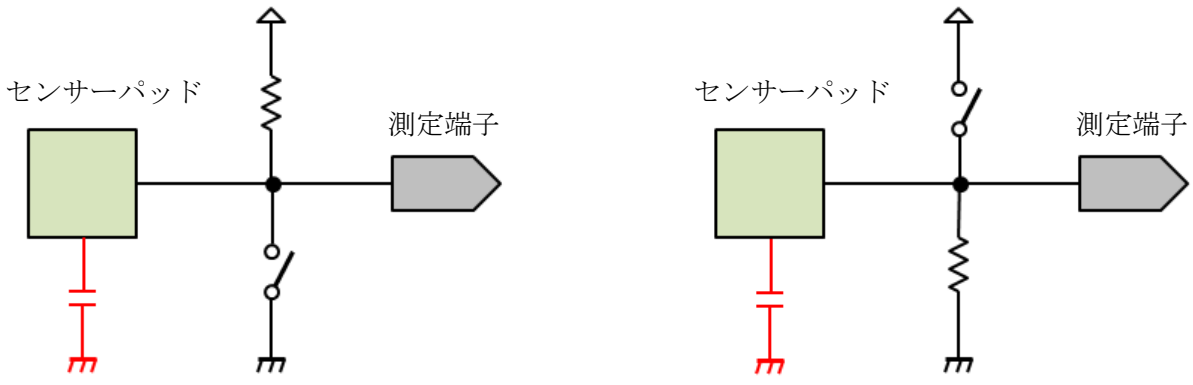



図 3-2　容量検出回路の概念図

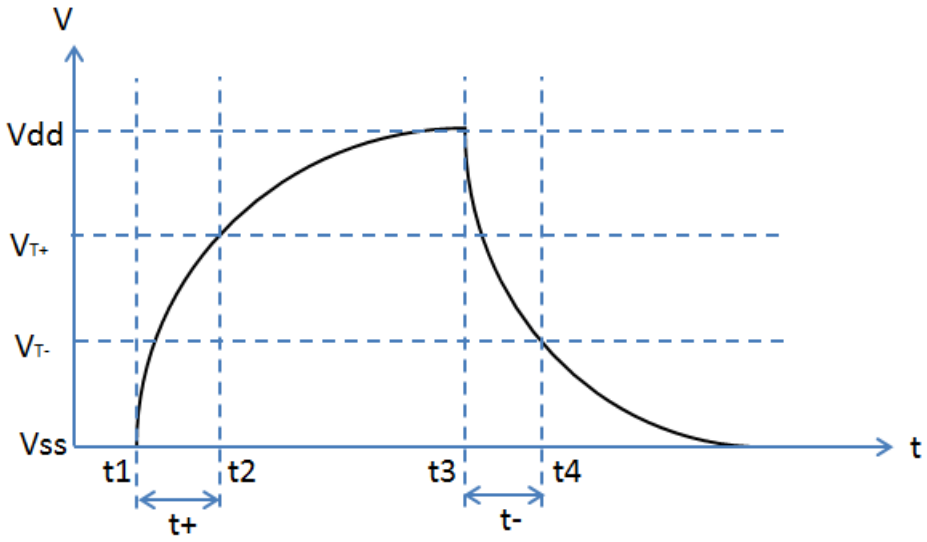



図 3-3　検出波形の時間変化

図 3-2 のセンサーパッドとグランドの間には容量成分が存在します。最初に図 3-2 の(a)のスイッチを一定時間オンにし、その容量成分に蓄えられた電荷をすべて放電させます。次に図 3-2 の(a)のスイッチをオフにした瞬間が図 3-3 の時刻 t1 に相当しますが、それ以降は抵抗を通して容量成分に電荷が充電されます。従い測定端子の電位は、τ=CR の時定数で上昇しますが、ここでは測定端子として CMOS シュミット入力回路を用います。CMOS シュミット入力回路は、$V_{T-}$以下の電圧の場合に”L”レベル、$V_{T+}$以上の電圧の場合に”H”レベルとして認識されます。すなわち、電圧が $V_{T+}$に達する時刻 t2 までの間は、CMOS シュミット入力回路は”L”レベルを認識し、時刻 t2 以降は”H”レベルを認識することになります。時刻 t1 と t2 の間の時間 t+を計測することでセンサーパッドの容量値を測定することができます。

逆の電位変化を起こすことでも容量値の測定は可能です。

図 3-2 の(b)のスイッチを一定時間オンにし、センサーパッドの容量成分に充電して Vdd まで電位を上昇させます。次に図 3-2 のスイッチをオフした瞬間が図 3-3 の時刻 t3 に相当しますが、それ以降は抵抗を通して容量成分の電荷は放電されます。従い測定端子の電位は、τ=CR の時定数で下降しますが、電圧が $V_{T-}$に達する時刻 t4 までの間は、CMOS シュミット入力回路は”H”レベルを認識し、時刻 t4 以降は”L”レベルを認識することになります。時刻 t3 と t4 の間の時間 t-を計測することでも、先と同様にセンサーパッドの容量値を測定することができます。

本アプリケーションノートでは、t+、t-の時間カウントのためには T16B を、図 3-2 のスイッチとしては PPORT を用いています。具体的には、図 3-2 の(a)のスイッチのオン/オフは、PPORT を”L”レベル出力/Hi-Z に切り替えることで、図 3-2 の(b)のスイッチのオン/オフは、PPORT を”H”レベル出力/Hi-Z に切り替えることで代替しています。また T16B による測定は、対象のセンサーパッドを UPMUX で T16B のキャプチャトリガ信号入力に割り当て、カウンタ値をキャプチャすることで容量相当値を得ています。

次の章で示す図 3-4 の、指を触れているセンサーパッドをセンシングしているときにアクティブな PPORT を二重枠線で、UPMUX の T16B に切り替えられていることを示すスイッチをオン状態で示しましたが、あわせて確認していただければ動作理解の一助になると思います。

## 3.3 検出回路

S1C17W22/S1C17W23 に搭載されている PPORT、UPMUX、T16B を用いた検出回路の一例を図 3-4 に示します。

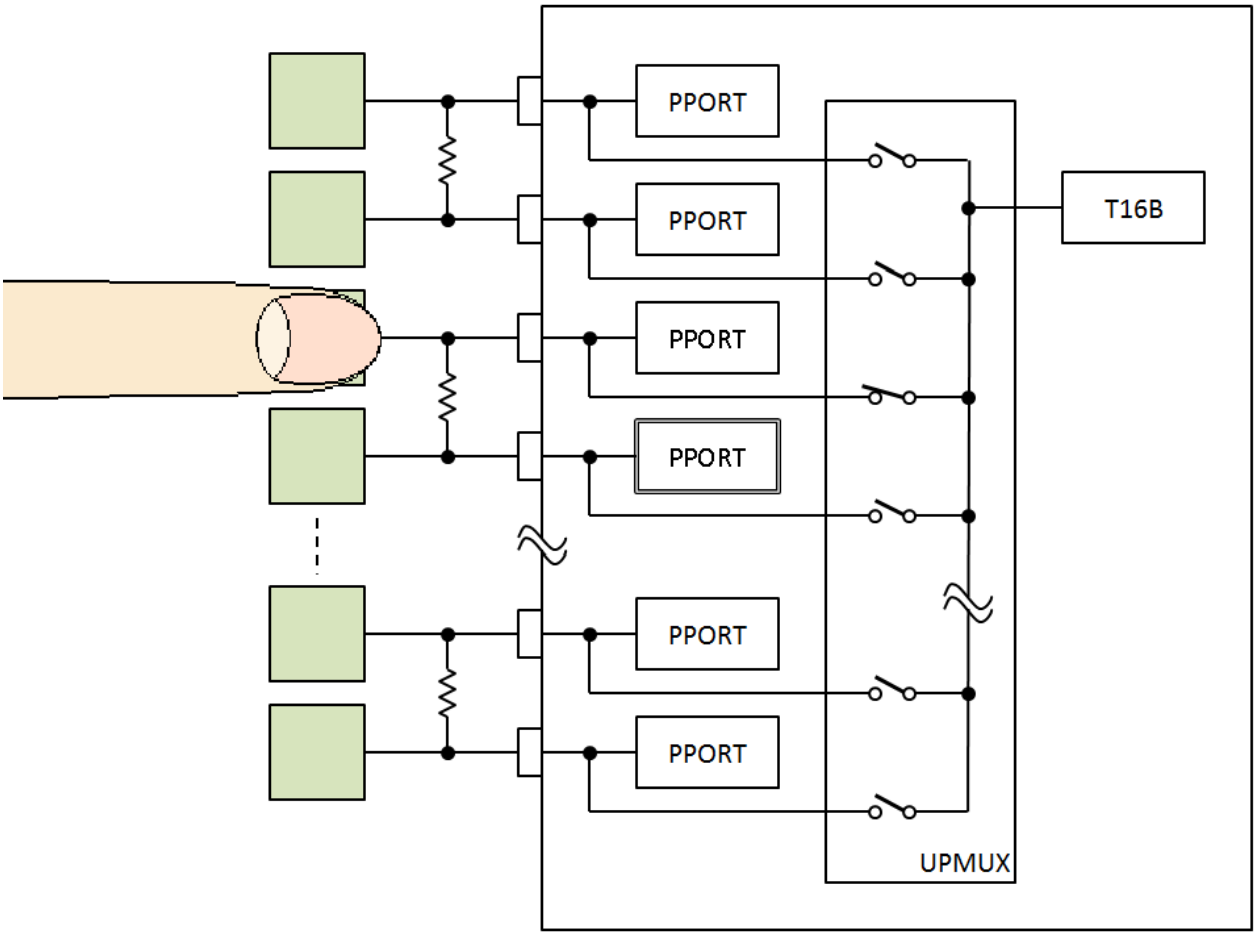



図 3-4　検出回路の一例

タッチキーは 2 つで一組になっており、間に抵抗器が設置されています。この抵抗器の値は、S1C17 シリーズの PPORT のリーク電流が $I_{LEAK}$＝±150nA の範囲であること、シュミット入力スレッショルド電圧の平均値が $V_{T+}$＝0.7Vdd、$V_{T-}$＝0.3Vdd であることを考慮し、以下のように決めることができます。

タッチキーの形状が 12mm×12mm の正方形、プリント基板として厚さ d＝1.2mm の FR-4 ガラスエポキシ基板を使用、タッチキーが単純な平行平板コンデンサであると仮定すると、面積 S＝144mm$^2$、比誘電率 $\varepsilon_r$＝4.7、真空の誘電率 $\varepsilon_0$＝8.855×10$^{-12}$F/m を用いて容量 C を計算すると、

$$C = \frac{\varepsilon_r \varepsilon_0 S}{d} = \frac{4.7 \times 8.855 \times 10^{-12} \times 144 \times 10^{-6}}{1.2 \times 10^{-3}} = 5.0\text{pF}$$

図 3-3 の t1 から t2 の間の時間 t+で、電圧が Vss から $V_{T+}$に変化する期間に着目して計算します。

抵抗器の抵抗値 R とタッチキーの容量 C による時定数 CR の影響で、検出波形の電圧 V は t1 からの経過時間 t に伴い以下のように変化します。

$$V = Vdd \times \left(1 - \exp\left(-\frac{1}{CR} \cdot t\right)\right)$$

上の式を変換して、

$$R = - \frac{t}{\ln\left(1 - \frac{V}{Vdd}\right)} \cdot \frac{1}{C}$$

時間 t+の経過で電圧 V＝$V_{T+}$＝0.7Vdd に変化するので、

$$R = - \frac{t+}{\ln\left(1 - \frac{0.7 \cdot Vdd}{Vdd}\right)} \cdot \frac{1}{5.0 \times 10^{-12}}$$

$$= 1.66 \times 10^{11} \cdot t+$$

t+の値として、T16B に入力される 4MHz のクロックの 100 カウントを用いると、

$$R = 1.66 \times 10^{11} \cdot t+$$

$$= 1.66 \times 10^{11} \cdot \frac{100}{4 \times 10^{6}}$$

$$= 4.15M\Omega$$

また、C17 シリーズのリーク電流が $I_{LEAK}$＝±150nA の範囲であり、Vdd=3.3V 時の抵抗値に換算すると、

$$R = \frac{V}{I} = \frac{3.3}{150 \times 10^{-9}}$$

$$= 22M\Omega$$

すなわち、先の 4.15MΩはリーク電流に比べて十分小さな抵抗値で、本センシング方式が実用的なものであることがわかります。

ところで、R の計算値は 4.15MΩになりましたが、実際には入手しやすい E24 系列の抵抗値 4.7MΩ、5.1 MΩ、5.6 MΩあたりが通常使用に適した値になります。

ただし、後ほど基板設計の章でも説明しますが、実際の C の値は、平行平板コンデンサとして計算される値より小さくなるようパターン設計しますが、それでも配線の引き回しなどに起因する容量が付加されるため、実際の C の値は 5.0pF よりも大きな値になり、非タッチ状態に於いて T16B でカウントされる値は先の 100 カウントより大きな値が観察されるのが通常です。

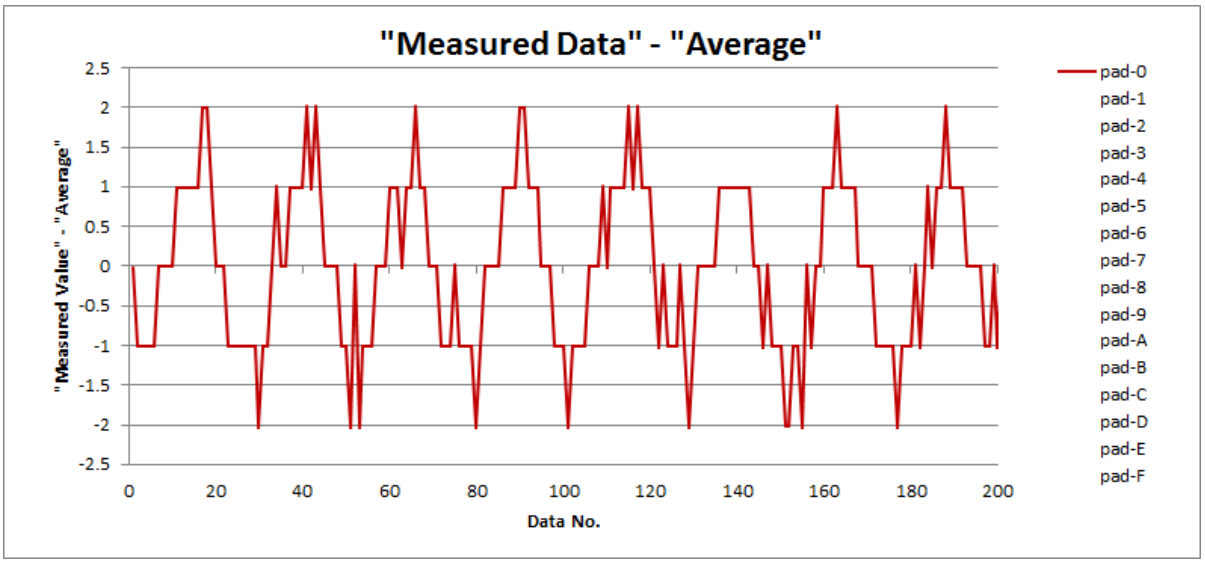
なお、図 3-3 の t3 から t4 の間の時間 t-で、電圧が Vdd から $V_T$に変化する期間に着目して計算しても、同様な結果が得られます。

# 4. ノイズの影響の抑制

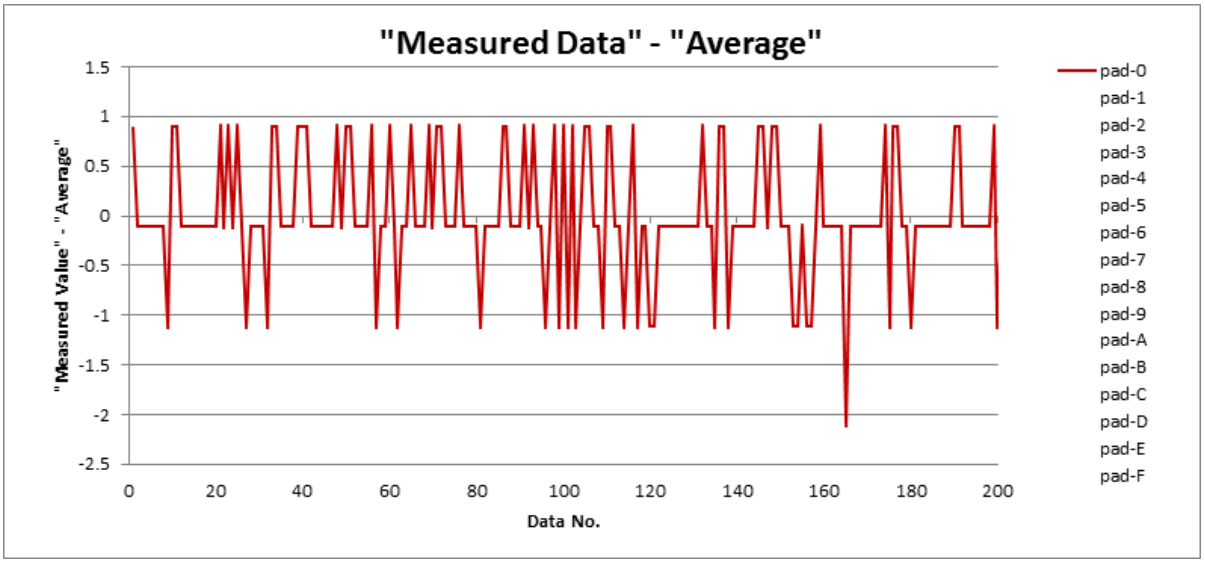
図 3-3 の検出波形で示したように、タッチキーの容量相当値は、t+、t-の 2 つを計測することができますが、非常に短い時間の間に測定されたこの 2 つの値を用いることで、ノイズの影響を抑制することができます。

図 4-1 の(a)は、50ms ごとにサンプリングされた 200 点の、容量相当値 t+の平均値との差をプロットしたものです。測定は電灯線に 60Hz の交流が用いられている地域で行われたものですが、約 25 点の周期の波形が乗っているのが見て取れます。サンプリング周波数 50ms×25 点＝1.25s なので、直接 60Hz のノイズが乗っていると言うより、サンプリング周期と干渉したノイズが見えているのだと思われます。

それに対し、図 4-1 の(b)は、50ms ごとにサンプリングされた 200 点の、容量相当値 t+と t-の値を足したものの平均値との差をプロットしたものです。先ほど±2 程度で見られた周期的な波形のノイズが、±1 程度の範囲に収まり、かつ明確な周期を持った波形は観察されません。すなわち、容量相当値 t+と t-の値を足すことで、一番大きなノイズ源である電灯線に起因するノイズの影響をある程度抑制できることがわかります。



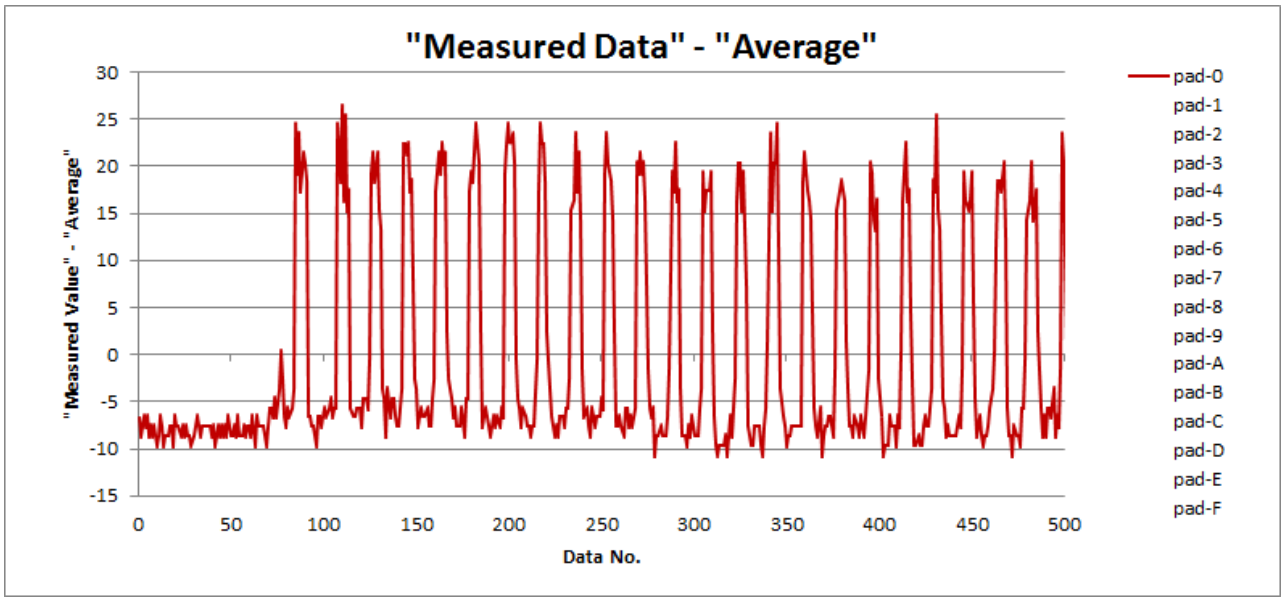

(a) t+



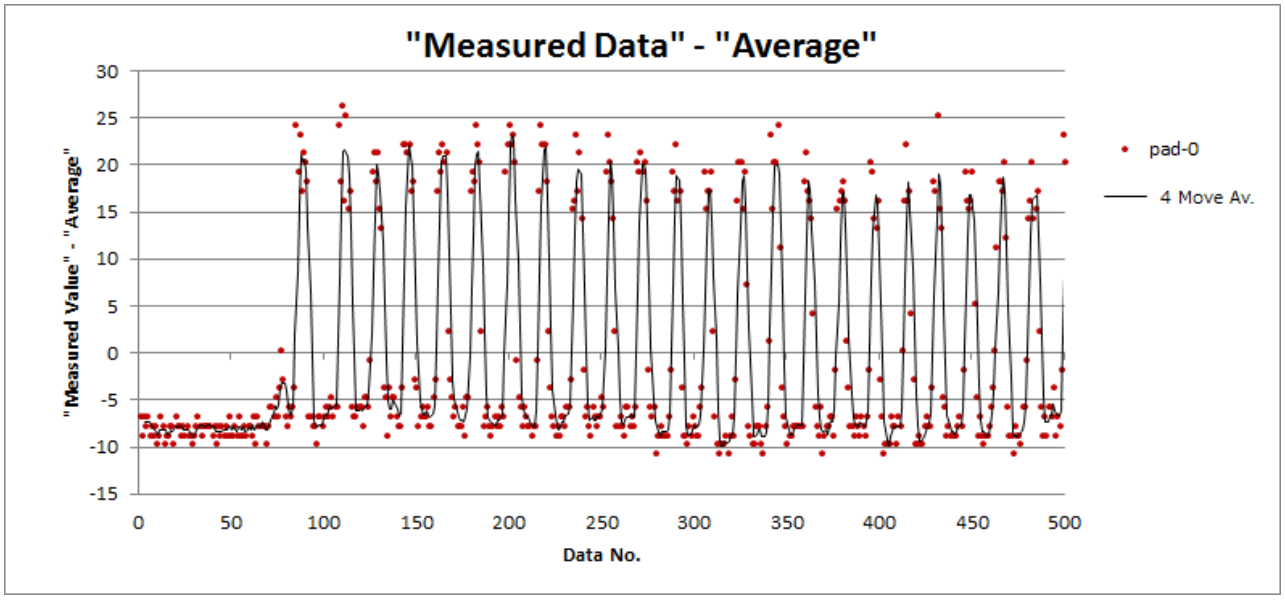

(b) t+と t-の値を足したもの

図 4-1　50ms ごとにサンプリングされた 200 点の容量相当値の平均値との差のプロット

また、図 4-2 の(a)は、12mm×12mm の正方形の形状のタッチキーの表面に 4mm の厚さのアクリル板を設置し、指のタッチを繰り返した時の、25ms ごとにサンプリングされた 500 点の容量相当値の平均値との差のプロットです。これだけでも明瞭にタッチ状態とそうでないときの状態の区別が可能ですが、図 4-2 の(b)に示したように、単純に 4 回の移動平均演算することで、更に明確に状態の区別が可能であることがわかります。




(a) 生データ




(b) 4 回の移動平均

図 4-2　25ms ごとにサンプリングされた 500 点の容量相当値の平均値との差のプロット

# 5. 基板の設計

図 5-1 にタッチキーの基板パターンのイメージを示します。

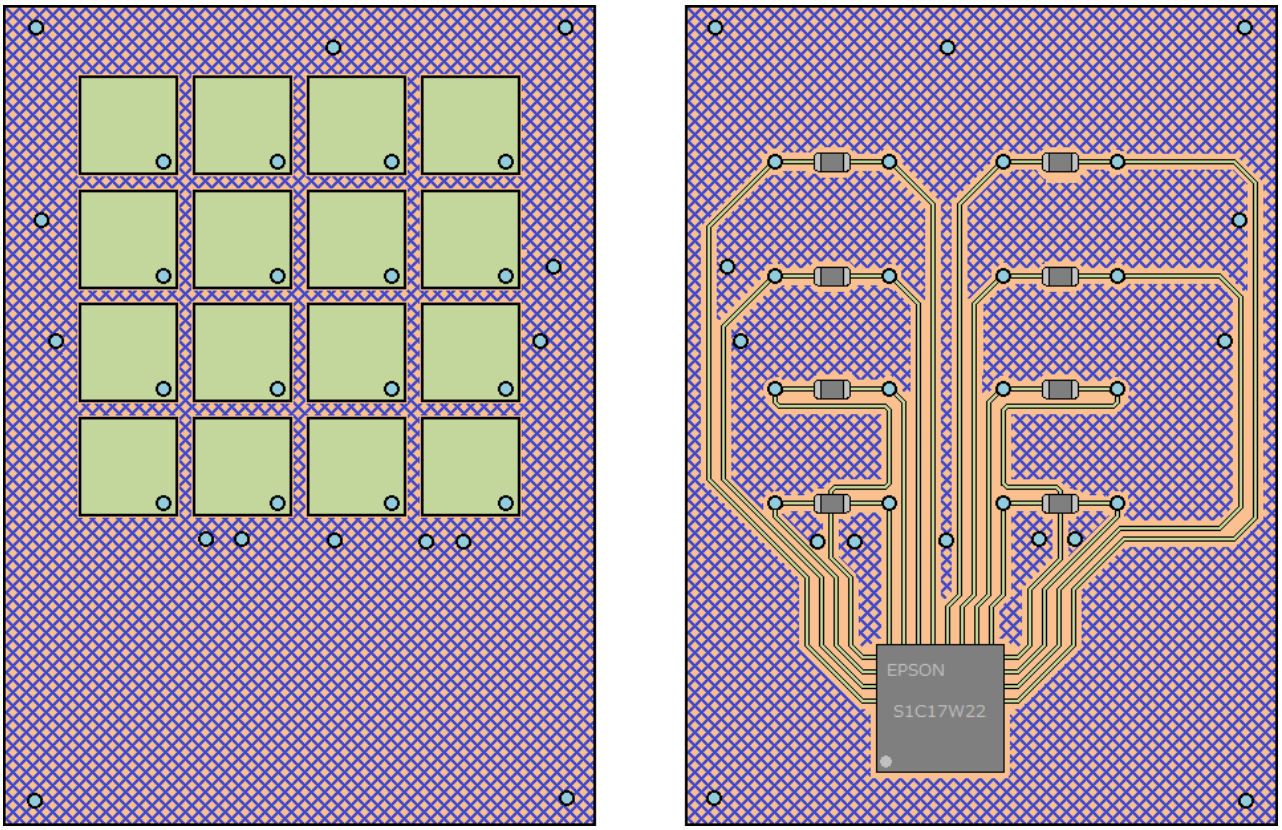



図 5-1 タッチキーの基板パターンのイメージ（表・裏）

タッチキーは、人の指の大きさを考慮し、10 mm□～20mm□の大きさ（形状は丸でも四角でも他の形状でも構いません）でパターンを作るのが一般ですが、タッチキーの電極は非常に入力インピーダンスが高くなる使われ方をするため、しっかりとしたグランドパターンでシールドする必要があります。しかし基板の横方向はともかく、縦方向 － 基板の裏と表の方向 － は、基板の裏側にベタでグランドパターンを敷いてしまうと、タッチされていない状態のときの容量が増大するために、検出感度が低下するという問題を引き起こします。その問題を回避するため、図 5-1 のようにグランド（青色で描画）は、メッシュ状にできるだけ細い線でパターニングするようにします。さらにその線の方向も、他の配線パターンとの干渉をできるだけ少なくするため、斜め 45° の方向にするのが望ましいでしょう。なお、線のピッチは 1mm 程度にするのが一般です。このようなグランドパターンをパッド裏全面に配置した 3 層以上の基板を用いるのが最もシールド効果が高いのは言うまでもありません。

# 6. ソフトウェア説明

## 6.1 s1c17w22_w23_touch_oscillo について

VBA マクロを含む Excel と連携して、一定の周期でタッチキーの容量で変わるカウンタ値（容量相当値）をサンプリングし、UART を用いて PC に送信して Excel の表に書き込むソフトウェア、s1c17w22_w23_touch_oscillo について説明します。

### 6.1.1 ファイル構成（src 内）

| ファイル名 | 機能 |
|---|---|
| boot.c | スタートアップモジュールファイル |
| init.c | 初期化関数ファイル |
| main.c | メイン関数ファイル |
| osc.c | OSC ドライバファイル |
| t16_ch0.c | T16 Ch.0 ドライバファイル |
| t16b_ch0.c | T16B Ch.0 ドライバファイル |
| uart.c | UART ドライバファイル |

### 6.1.2 ファイル構成（inc 内）

| ファイル名 | 機能 |
|---|---|
| reg | S1C17W22 周辺機器レジスタ定義ファイル格納フォルダ |
| c17w22_reg.h | S1C17W22 周辺機器ヘッダ定義ファイル |
| init.h | 初期化関数ヘッダ定義ファイル |
| osc.h | OSC ドライバヘッダ定義ファイル |
| t16_ch0.h | T16 Ch.0 ドライバヘッダ定義ファイル |
| t16b_ch0.h | T16B Ch.0 ドライバヘッダ定義ファイル |
| touch_oscillo.h | タッチキー容量値視覚化ヘッダ定義ファイル |
| uart.h | UART ドライバヘッダ定義ファイル |

### 6.1.3 モジュール説明

ファイル中のモジュールのうち、一般的でない設定を中心に、関数名とその機能、配列変数名とその内容について説明します。

ファイル名： main.c

| 関数名 | 機能 |
|---|---|
| measCap | to.scanPort で指定されたセンサーパッドの容量相当値を測定し、測定が終了したら戻ります。 |
| intUartCh0 | UART Ch.0 割り込み関数。PC 上の VBA からのコントロール情報の受信に伴い、センサーパッドの容量相当値の測定を開始するため、T16 Ch.0 の動作をスタートします。 |
| intT16Ch0 | T16 Ch.0 割り込み関数。すでに受け取っているコントロール情報で、測定するよう指示されているセンサーパッドの容量相当値を順に測定し、PC 上の VBA に送信します。指定個数のデータの送信が終了したところで、T16 Ch.0 の動作をストップします。 |
| intT16bCh0 | T16B Ch.0 割り込み関数。キャプチャ 0 割り込みの時には、t+に相当する容量相当値をキャプチャ 0 データ値から得て、t-に相当する容量相当値を測定するよう PPORT や UPMUX の状態を変更します。キャプチャ 1 割り込みの時には、t-に相当する容量相当値をキャプチャ 1 データ値から得て、当該センサーパッドの測定完了の状態になるよう PPORT や UPMUX の状態を変更します。 |

本ソフトウェア、s1c17w22_w23_touch_oscillo は、以下の 16 個の PPORT にタッチキーを割り当てているものとしてコーディングされています。なお、背景の色が同じもの同士でペアになっています。すなわち、それらのタッチキー間には 5MΩ前後の抵抗器が接続されます。

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| P00 | P01 | P02 | P03 | P04 | P05 | P06 | P07 |
| P10 | P11 | P12 | | | P15 | P16 | P17 |
| P20 | P21 | | | | | | |

ファイル中の配列変数の値を確認する場合は、上記の前提を考慮し、以下の表とあわせ、眺めるようにするのが望ましいでしょう。

ファイル名： touch_oscillo.h

| 配列変数名 | 内容 |
|---|---|
| touchPort | PPORT のポートグループ番号。0： P0x、1： P1x、2： P2x |
| touchOen | PPORT のアウトプットイネーブル。同じペアは同じ値になるので、ペアで兼用。0： disable、1： enable |
| touchDat | センシングする PPORT の、ペアのもう一つの PPORT の出力値。<br>0： “L”を出力、1： “H”を出力 |
| touchDatHH | ペアになっている PPORT 共々”H”を出力する。同じペアは同じ値になるので、ペアで兼用。0： “L”を出力、1： “H”を出力 |
| touchIOEN | センシングする PPORT の、ペアのもう一つの PPORT のアウトプットイネーブル値。0： disable、1： enable |
| touchUpmux1 | t+計測時、PPORT を UPMUX に割り当てる。PPORT のポート番号の偶数、奇数の 2 種類分あれば十分なので、配列の要素数は 2。 |
| touchUpmux2 | t-計測時、PPORT を UPMUX に割り当てる。PPORT のポート番号の偶数、奇数の 2 種類分あれば十分なので、配列の要素数は 2。 |
| touchUpmuxNo | UPMUX を設定する際のレジスタ P[0-2]UPMUXx の x の値。 |
| modSel | 周辺機器を使うようにするときの P[0-2]MODSEL の設定値。 |
| fncSel | Func#1 を使うようにするときの P[0-2]FNCSEL の設定値。 |

### 6.1.4 操作手順

プロジェクトのインポート

(1) IDE を起動して、「s1c17w22_w23_touch_oscillo」プロジェクトをインポートしてください。

※ インポートの方法は、S5U1C17001C Manual “3.ソフトウェア開発手順”を参照してください。

ビルド

(1) IDE を使用して、「s1c17w22_w23_touch_oscillo」プロジェクトをビルドしてください。

Excel VBA を使用するための準備

(1) デスクトップに MeasTouch フォルダをコピーしてください。このフォルダには、Excel ファイル「MeasTouch.xlsm」、Active-X コントロールファイル「NonComSck.ocx」が入っています。同梱されていない、マイクロソフトが提供する Active-X コントロールファイル「MSCOMM32.OCX」は、種々のサイトからダウンロードすることが可能ですが、それを入手し、先の MeasTouch フォルダ内にコピーしてください。

(2) [すべてのプログラム]→[アクセサリ]→[コマンド プロンプト]上で右ボタンを押し、[管理者として実行…]を選択すると、「管理者: コマンド プロンプト」が起動します。

(3) 「cd C:¥Desktop¥MeasTouch」、「regsvr32.exe MSCOMM32.OCX」、「regsvr32.exe NonComSck.ocx」を実行して Active-X コントロールファイルを有効にします。

(4) 上記操作でうまくいかない場合、Active-X コントロールファイルを「C:¥Windows¥System32」にコピーし、「cd C:¥Windows¥System32」、「regsvr32.exe MSCOMM32.OCX」、「regsvr32.exe NonComSck.ocx」を実行してください。

接続、電源投入

(1) SVT17W23、ICDmini、USB ケーブル、PC を接続してください。

(2) SVT17W23 の UART Ch.0 と、Excel VBA を実行する PC のシリアル端子（USB 接続のシリアル通信 I/F を使うのが一般的です。市販品の例を Excel VBA ファイルのシート「Note」に示しました。）を接続してください。PC に搭載された CPU の処理能力が十分なら、この PC は先の PC と兼用することもできます。

(3) SVT17W23、ICDmini をリセットしてください。

Excel の VBA の実行

(1) Excel ファイル「MeasTouch.xlsm」をダブルクリックして実行してください。

実行

(1) 必要に応じて、コマンドファイル（s1c17w22_w23_touch_oscillo_gnu17IDE.cmd）内の fls17w22.elf へのパスを変更してください。デフォルトでは、C:/EPSON/GNU17/mcu_model/17W22/fls/fls17w22.elf となっています。
※ コマンドファイルの修正方法は、S5U1C17001C Manual を参照してください。

(2) IDE を使用して、「s1c17w22_w23_touch_oscillo」プロジェクトを実行してください。

(3) 「s1c17w22_w23_touch_oscillo」の実行後、Excel VBA からのコマンドで、タッチキーをスキャンします。

### 6.1.5 MeasTouch.xlsm の使い方

MeasTouch.xlsm の外観を図 6-1 に示します。

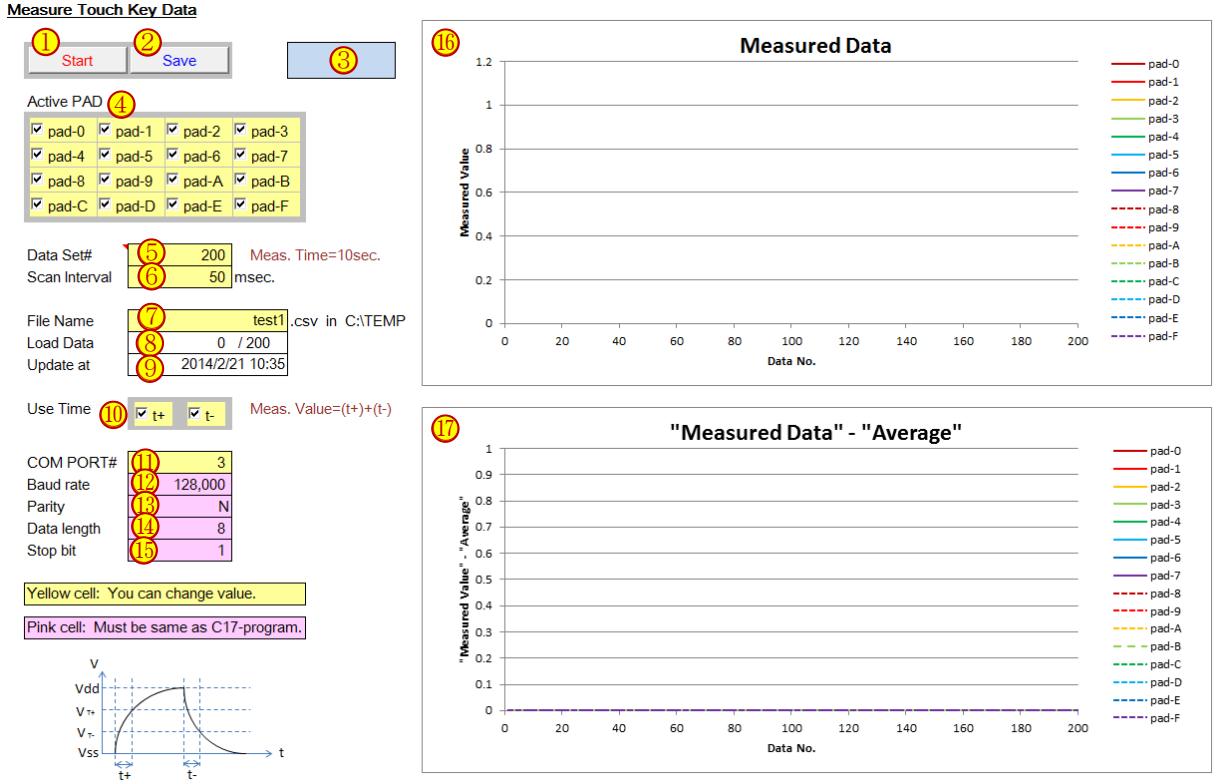



図 6-1　MeasTouch.xlsm の外観

| | |
|---|---|
| ① 測定開始ボタン | タッチキーの容量相当値の取得を開始します。 |
| ② データセーブボタン | 計測データを、CSV 形式でセーブします。 |
| ③ 状態インジケータ | 測定中、セーブ中にその旨 表示します。 |
| ④ 測定対象タッチキー選択 | 測定対象のタッチキーを選択します。 |
| ⑤ データセット数 | 取り込むデータセット数を指定します。 |
| ⑥ スキャンインターバル | データの取り込み間隔を指定します。 |
| ⑦ セーブファイル名 | 計測データを保管するファイルの名称を指定します。 |
| ⑧ データ取り込み数 | 計測したデータセットの数を逐次更新して表示します。 |
| ⑨ データ採取日時 | データ採取の最終日時を表示します。 |
| ⑩ t+、t- 使用選択 | t+と t-のどちら(または両方)を容量相当値変換するか指定します。 |
| ⑪ COM ポート番号 | シリアル通信の COM ポートの番号を指定します。 |
| ⑫ ボーレート | シリアル通信のボーレートを指定します。 |
| ⑬ パリティー | シリアル通信のパリティー使用/不使用、偶数/奇数を指定します。 |
| ⑭ データ長 | シリアル通信のデータ長を指定します。 |
| ⑮ ストップビット数 | シリアル通信のストップビットの数を指定します。 |
| ⑯ 測定データ グラフ | 縦軸「測定データ」のグラフをプロットします。 |
| ⑰ 測定データ － 平均値 グラフ | 縦軸「測定データ － 平均値」のグラフをプロットします。 |

注１） 本サンプルプログラムで用いているUARTによるシリアル通信は2線式通信ですが、SVT17W23はデータを一方的に送る仕様で作られているため、PC 側の受信バッファが追随できなくなると通信データが欠落します。その検出はデータの上位、下位の値が範囲外になることで検知する方法で行っていますが、通信が追随できない旨のエラーが表示された場合、④、⑤、⑥の設定値を変更することで回避するようにしてください。

注２） シリアル通信に関する設定、⑪、⑫、⑬、⑭、⑮は、通常は変更する必要のない情報です。もしもこれらの情報を変更した場合、s1c17w22_w23_touch_oscillo の当該箇所の変更が必要になりますのでご注意願います。

注３） グラフ⑯、⑰の描画は全データ取得後に行われます。すなわち、データ取得時には逐次更新されず、以前のデータがプロットされている状態をキープします。

### 6.1.6 サンプルプログラム動作概要

① 割り込みレベルのセット、変数の初期化の後、システムクロックを IOSC から OSC3（内蔵発振 4MHz）に切り替えます。

② UART、T16、T16B を初期化します。

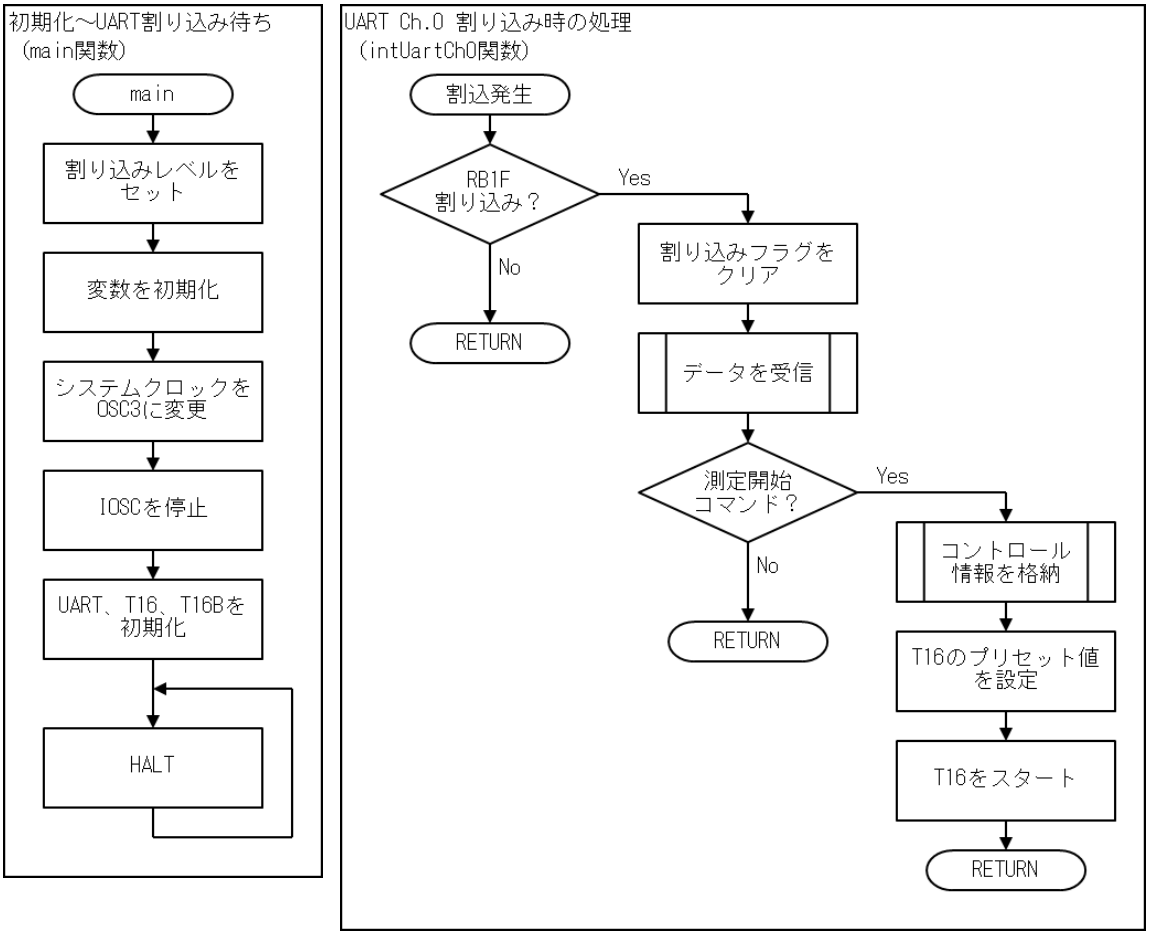
③ UART の割り込みを待つために HALT します。

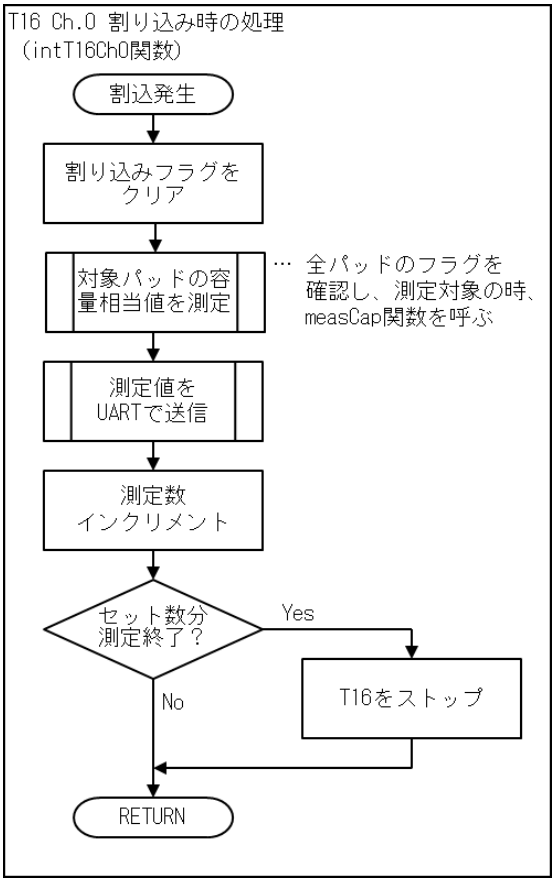
④ UART の割り込みが発生すると、PC から送られてきたコントロール情報を受信し、センサーパッドの容量相当値の測定を開始するため、T16 Ch.0 の動作をスタートします。

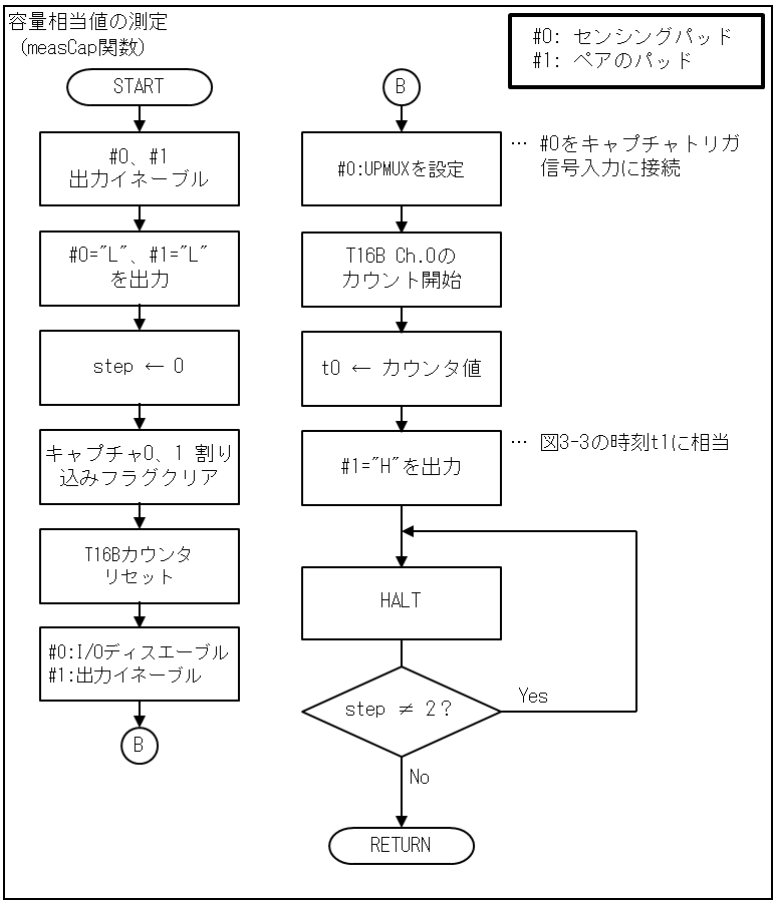
⑤ T16 Ch.0 の割り込みが発生するたび、受け取っているコントロール情報を基に、測定するよう指示されているセンサーパッドの容量相当値を順に測定し、PC に測定データを送信します。指定個数のデータの送信が終了したところで、T16 Ch.0 の動作をストップします。

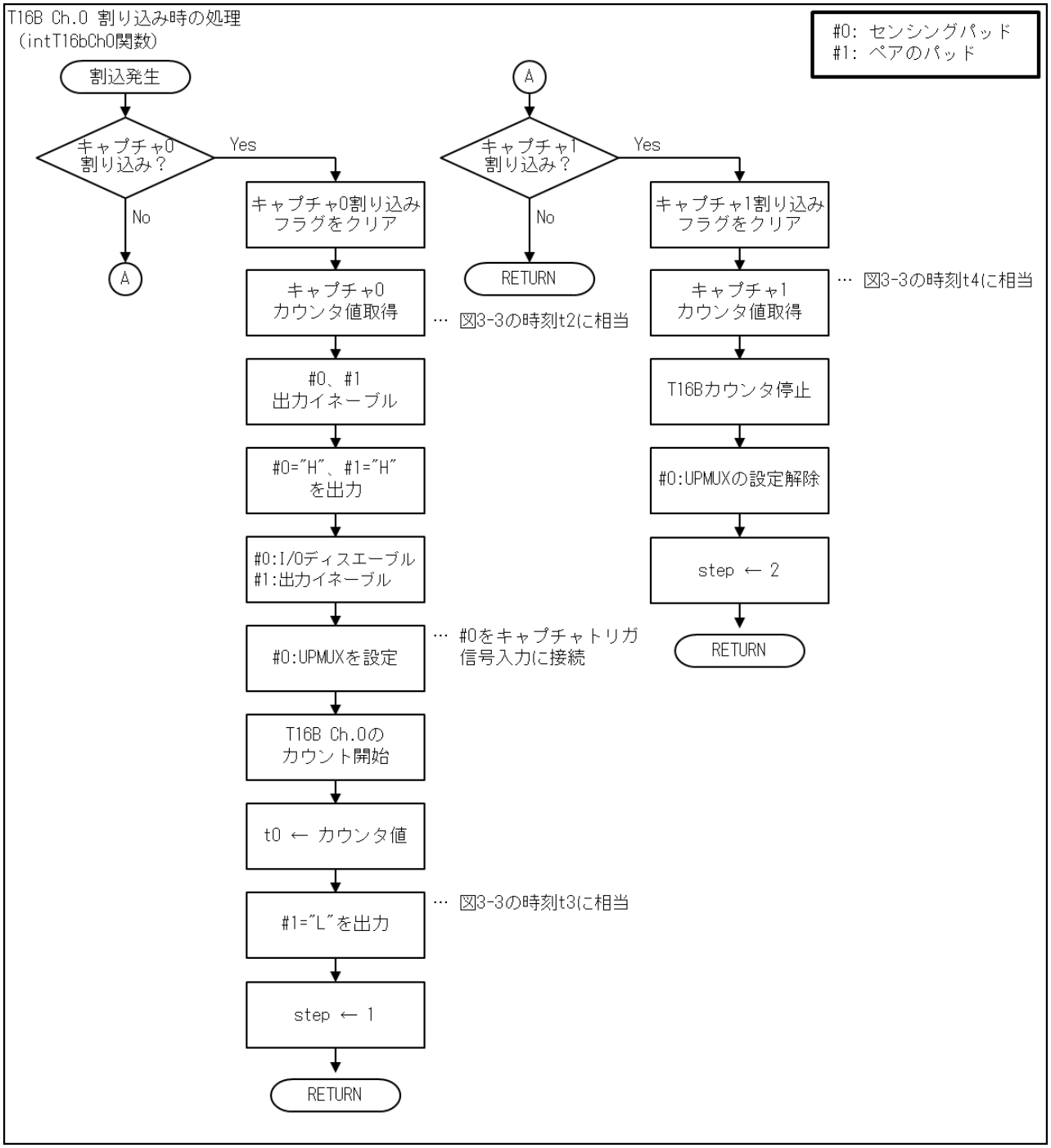
⑥ T16B Ch.0 の割り込みが発生すると、キャプチャ 0 割り込みの時には、t+に相当する容量相当値をキャプチャ 0 データ値から得て、t-に相当する容量相当値を測定するよう PPORT や UPMUX の状態を変更します。キャプチャ 1 割り込みの時には、t-に相当する容量相当値をキャプチャ 1 データ値から得て、当該センサーパッドが測定完了の状態になるよう PPORT や UPMUX の状態を変更します。

フローチャートを以下に記します。




T16 Ch.0 割り込み時の処理
(intT16Ch0関数)
割込発生
割り込みフラグを
クリア
対象パッドの容
量相当値を測定
… 全パッドのフラグを
確認し、測定対象の時、
measCap関数を呼ぶ
測定値を
UARTで送信
測定数
インクリメント
セット数分
測定終了？
Yes
No
T16をストップ
RETURN


容量相当値の測定
(measCap関数)
#0: センシングパッド
#1: ペアのパッド
START
#0、#1
出力イネーブル
#0="L"、#1="L"
を出力
step ← 0
キャプチャ0、1 割り
込みフラグクリア
T16Bカウンタ
リセット
#0:I/Oディスエーブル
#1:出力イネーブル
B
B
#0:UPMUXを設定
… #0をキャプチャトリガ
信号入力に接続
T16B Ch.0の
カウント開始
t0 ← カウンタ値
#1="H"を出力
… 図3-3の時刻t1に相当
HALT
step ≠ 2?
Yes
No
RETURN


T16B Ch.0 割り込み時の処理
（intT16bCh0関数）
#0: センシングパッド
#1: ペアのパッド
割込発生
キャプチャ0
割り込み？
Yes
No
A
キャプチャ0割り込み
フラグをクリア
キャプチャ0
カウンタ値取得
… 図3-3の時刻t2に相当
#0、#1
出力イネーブル
#0="H"、#1="H"
を出力
#0:I/Oディスエーブル
#1:出力イネーブル
#0:UPMUXを設定
… #0をキャプチャトリガ
信号入力に接続
T16B Ch.0の
カウント開始
t0 ← カウンタ値
#1="L"を出力
… 図3-3の時刻t3に相当
step ← 1
RETURN
A
キャプチャ1
割り込み？
Yes
No
RETURN
キャプチャ1割り込み
フラグをクリア
キャプチャ1
カウンタ値取得
… 図3-3の時刻t4に相当
T16Bカウンタ停止
#0:UPMUXの設定解除
step ← 2
RETURN

## 6.2 s1c17w22_w23_touch_demo について

タッチキーのデモ用に、タッチキーの状態、容量相当値などを液晶表示するソフトウェア、s1c17w22_w23_touch_demo について説明します。

### 6.2.1 ファイル構成（src 内）

| ファイル名 | 機能 |
|---|---|
| boot.c | スタートアップモジュールファイル |
| display.c | LCD 表示関数ファイル |
| init.c | 初期化関数ファイル |
| main.c | メイン関数ファイル |
| osc.c | OSC ドライバファイル |
| snda.c | SNDA ドライバファイル |
| t16_ch0.c | T16 Ch.0 ドライバファイル |
| t16b_ch0.c | T16B Ch.0 ドライバファイル |

### 6.2.2 ファイル構成（inc 内）

| ファイル名 | 機能 |
|---|---|
| reg | S1C17W22 周辺機器レジスタ定義ファイル格納フォルダ |
| c17w22_reg.h | S1C17W22 周辺機器ヘッダ定義ファイル |
| display.h | LCD ドライバヘッダ定義ファイル |
| init.h | 初期化関数ヘッダ定義ファイル |
| lcd_font.h | LCD フォント定義ファイル |
| osc.h | OSC ドライバヘッダ定義ファイル |
| snda.h | SNDA ドライバヘッダ定義ファイル |
| t16_ch0.h | T16 Ch.0 ドライバヘッダ定義ファイル |
| t16b_ch0.h | T16B Ch.0 ドライバヘッダ定義ファイル |
| touch_sw.h | タッチキー制御ヘッダ定義ファイル |

### 6.2.3 モジュール説明

ファイル中のモジュールのうち、一般的でない設定を中心に、関数名とその機能、配列変数名とその内容について説明します。

ファイル名： main.c

| 関数名 | 機能 |
|---|---|
| measCap | ts.scanPort で指定されたセンサーパッドの容量相当値を測定し、測定が終了したら戻ります。 |
| val2Str3 | 引数 str の位置 pos から digit 桁の範囲を、引数 val の値を数値文字列化したもので置換します。 |
| dispStat | 引数 touched の値が 0 の時には文字列”NoTouchKey”を、1 の時には”ボタン番号=検出値/当該ボタンのバックグランドレベル値”を、2 の時には”Chattering”を 1 行目に表示します。2 行目には、タッチの有無（*/空欄）、タッチ判断レベル、全キーのタッチ状態を 16 進数 4 桁で表示します。 |
| intT16Ch0 | T16 Ch.0 割り込み関数。システムクロックを、待機状態の OSC1(32.768kHz)から OSC3(内蔵発振 4MHz)に切り替えます。16 個のセンサーパッドの容量相当値を順に測定し、移動平均計算用にデータ処理します。得られた結果は、以下のように ts.stage の値により取り扱いが異なります。<br><br>0 の時： 移動平均を求めるのに必要なデータ数に達したら次のステージに移りますが、それまでのデータ収集期間です。<br><br>1 の時： バックグランドレベルを決めるために必要なデータ数に達したら次のステージに移りますが、それまでバックグランドレベル算出用のデータを収集する期間です。<br><br>2 の時： 次のステージ（指で指定パッドをタッチ）に移るまでに若干の間隔を置くための待ちの期間です。<br><br>3 の時： P00 に接続されたセンサーパッドの容量相当値を計測することで、タッチ状態でどれだけ容量相当値が変化するか計測するための期間です。<br><br>4 の時： 通常動作の期間です。全センサーパッドの容量相当値の変化値を見てオン状態かオフ状態か判断するとともに、一番変化量が大きなセンサーパッドについて LCD に情報表示します。また一定期間毎、バックグランドレベルの更新を行うことで環境変化に追従します。オン状態が規定数以上に連続した場合にも、それを新たなバックグランドレベルとしてみなすことで、ポケット内に機器が置かれた場合などに生じる動作異常からの回復を図ります。そしてセンサーパッドのオフ状態が継続しているときにはスキャンインターバルを疎らにしその旨 LCD 表示し、逆にセンサーパッドのオン状態が検出された時には速やかにスキャンインターバルを短くすることで高レスポンスと低消費電力化を両立させます。センサーパッドがオフ状態からオン状態に変化した時にはクリック音を発生させます。<br><br>上記の何れのステージでも各ステージの処理が終了したら、システムクロックを、OSC3(内蔵発振 4MHz)から待機状態の OSC1(32.768kHz)に切り替えます。 |
| intT16bCh0 | T16B Ch.0 割り込み関数。キャプチャ 0 割り込みの時には、t+に相当する容量相当値をキャプチャ 0 データ値から得て、t-に相当する容量相当値を測定するよう PPORT や UPMUX の状態を変更します。キャプチャ 1 割り込みの時には、t-に相当する容量相当値をキャプチャ 1 データ値から得て、当該センサーパッドの測定完了の状態になるよう PPORT や UPMUX の状態を変更します。 |

本ソフトウェア、s1c17w22_w23_touch_demo は、先のソフトウェア、s1c17w22_w23_touch_oscillo と同様､以下の 16 個の PPORT にタッチキーを割り当てているものとしてコーディングされています。なお、背景の色が同じもの同士でペアになっています。すなわち、それらのタッチキー間には 5MΩ 前後の抵抗器が接続されます。

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| P00 | P01 | P02 | P03 | P04 | P05 | P06 | P07 |
| P10 | P11 | P12 | | | P15 | P16 | P17 |
| P20 | P21 | | | | | | |

ファイル中の配列変数の値を確認する場合は、上記の前提を考慮し、以下の表とあわせ、眺めるようにするのが望ましいでしょう。

ファイル名： touch_sw.h

| 配列変数名 | 内容 |
|---|---|
| touchPort | PPORT のポートグループ番号。0： P0x、1： P1x、2： P2x |
| touchOen | PPORT のアウトプットイネーブル。同じペアは同じ値になるので、ペアで兼用。0： disable、1： enable |
| touchDat | センシングする PPORT の、ペアのもう一つの PPORT の出力値。<br>0： “L”を出力、1： “H”を出力 |
| touchDatHH | ペアになっている PPORT 共々”H”を出力する｡同じペアは同じ値になるので、ペアで兼用。0： “L”を出力、1： “H”を出力 |
| touchIOEN | センシングする PPORT の、ペアのもう一つの PPORT のアウトプットイネーブル値。0： disable、1： enable |
| touchUpmux1 | t+計測時、PPORT を UPMUX に割り当てる。PPORT のポート番号の偶数、奇数の 2 種類分あれば十分なので、配列の要素数は 2。 |
| touchUpmux2 | t-計測時、PPORT を UPMUX に割り当てる。PPORT のポート番号の偶数、奇数の 2 種類分あれば十分なので、配列の要素数は 2。 |
| touchUpmuxNo | UPMUX を設定する際のレジスタ P[0-2]UPMUXx の x の値。 |
| modSel | 周辺機器を使うようにするときの P[0-2]MODSEL の設定値。 |
| fncSel | Func#1 を使うようにするときの P[0-2]FNCSEL の設定値。 |

なお、上記ファイルの記述内容は、s1c17w22_w23_touch_oscillo の touch_oscillo.h と基本的には同じで、LCD やブザー用の割り当てポートに関する記述が異なるだけです。

### 6.2.4 操作手順

プロジェクトのインポート

(1) IDE を起動して、「s1c17w22_w23_touch_demo」プロジェクトをインポートしてください。

※ インポートの方法は、S5U1C17001C Manual “3.ソフトウェア開発手順”を参照してください。

ビルド

(1) IDE を使用して、「s1c17w22_w23_touch_demo」プロジェクトをビルドしてください。

接続、電源投入

(1) SVT17W23、ICDmini、USB ケーブル、PC を接続してください。このとき、先の s1c17w22_w23_oscillo と同様な環境で操作を行うのなら、UART によるシリアル通信用の配線は取り除いてください。

(2) SVT17W23、ICDmini をリセットしてください。

実行

(1) 必要に応じて、コマンドファイル（s1c17w22_w23_touch_demo_gnu17IDE.cmd）内の fls17w22.elf へのパスを変更してください。デフォルトでは、C:/EPSON/GNU17/mcu_model/17W22/fls/fls17w22.elf となっています。
※ コマンドファイルの修正方法は、S5U1C17001C Manual を参照してください。

(2) IDE を使用して、「s1c17w22_w23_touch_demo」プロジェクトを実行してください。

(3) 「s1c17w22_w23_touch_demo」の実行後、タッチキー動作のデモンストレーションを開始します。

### 6.2.5 サンプルプログラム動作概要

① OSC1（32.768kHz）の発振を開始します。

② LCD を初期化し、表示をクリアします。

③ ブザー用に SNDA を初期化し、開始します。

④ T16 Ch0、T16B Ch0 の初期化を行います。

⑤ T16 Ch0 をスタートします。

⑥ “Init.-1/3 ”、”Chk BgLvl ”を LCD に表示します。この表示は、3 ステップの初期化の 1 ステップ目の行程､すなわちセンサーパッドのバックグランドレベルのデータを収集していることを示すためのものです。

⑦ システムクロックを IOSC から OSC1（32.768kHz）に切り替えます。

⑧ T16 Ch0 の割り込みを待つために HALT します。

⑨ T16 Ch0 の割り込みが発生する度、16 個のセンサーパッドの容量相当値を測定します。得られた測定値をもとに、順次初期化を開始します。まずは、“Init.-1/3 ”、”Chk BgLvl ”が LCD に表示されている間中、センサーパッドには手や指を近づけないようにします。

⑩ 次に LCD の表示が、”Init.-2/3 ”、”Waiting…”となりますが、この期間の間に、指を P00 に接続されたセンサーパッド上にタッチし、次のステージに備えます。この表示は、3 ステップの初期化の 2 ステップ目の行程､すなわち次のステップの前の待ちの行程であることを示すためのものです。

⑪ 次に LCD の表示が、”Init.-3/3 ”、”Touch Btn0”となりますが、この期間の間中、指を P00 に接続されたセンサーパッド上にタッチしたままにします。このステージが終了するとキーのクリック音が発生し、通常動作に移行します。

⑫ 通常動作では、全センサーパッドの容量相当値の変化値を見てオン状態かオフ状態かを判断するとともに、一番変化量が大きなセンサーパッドについて、LCD に情報を表示します。表示情報は以下の通りです。

1 行目： センサーパッド番号=計測値/バックグランドレベル　　または、
タッチ状態未検出の時は、　　NoTouchKey　　または、
チャタリングを検出した時は、　　Chattering

2 行目： マーク_タッチ判断閾値_全キーのタッチ状態
（マーク： タッチ検出時「*」、未検出時「_」（スペース）。）

例えば、12 番目（No.11）のセンサーパッドである P15 に接続されたセンサーパッドの容量相当値の変化が一番大きいと検出されたとき、以下のような表示になります。

1 行目： B=020/0625
2 行目： *_016_0800

2 行目の表示情報は、複数のセンサーパッドへの同時タッチの検出が可能なことを確認するためのもので、例えば全センサーパッドがタッチ状態とみなされた時の表示は以下のようになります。

1 行目： 3=034/0548
2 行目： *_016_FFFF

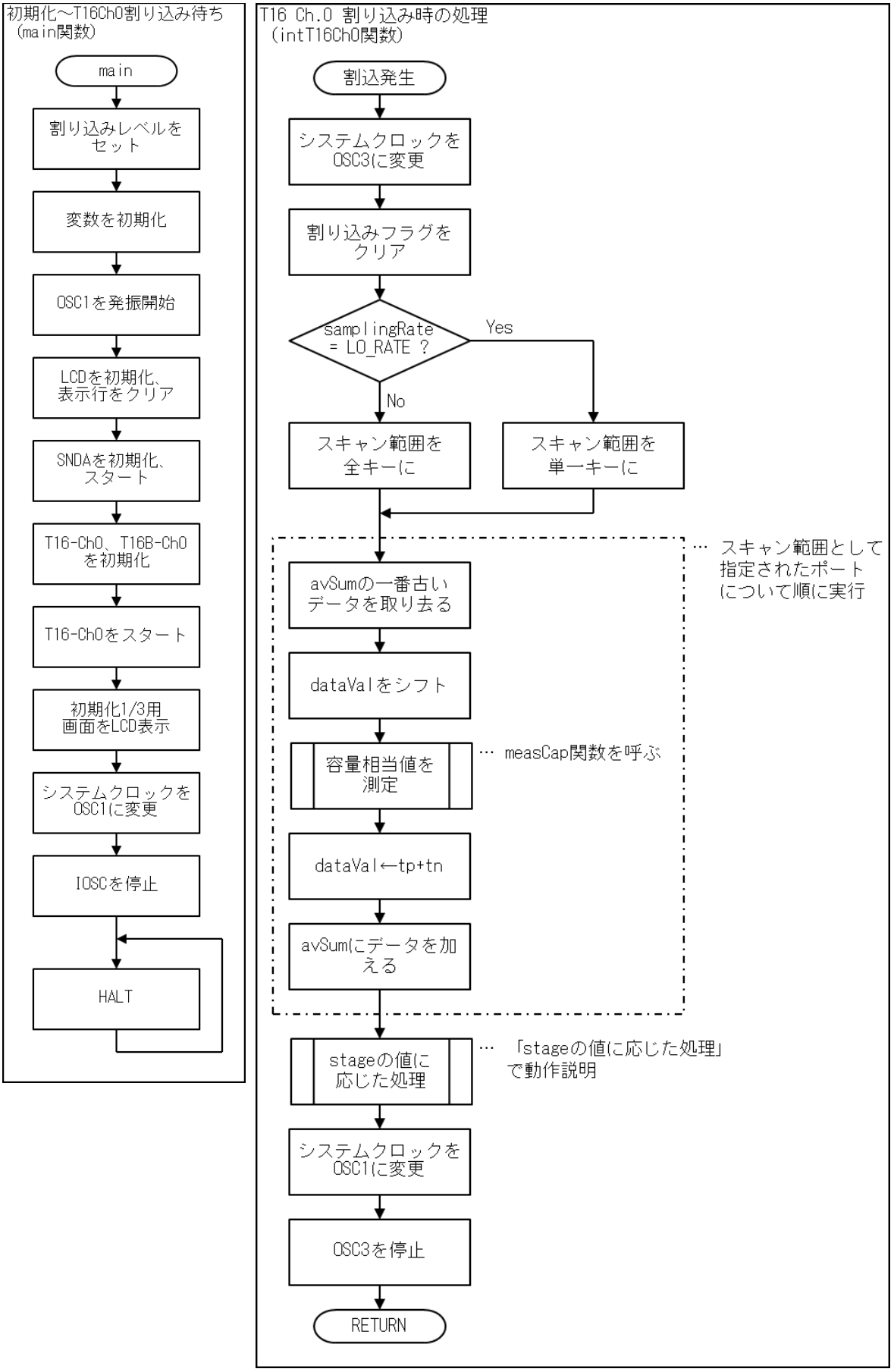
一定時間、キーへのタッチが検出されなくなると、センサーパッドの容量相当値計測が決められた 1 つのキー（電源 ON の機能に相当）に限定されるとともに、計測のサイクルの頻度を落とし、省電力化を図ります。その際には、LCD に「Sleeping…」と表示されます。
その決められたキーへのタッチが検出されると、通常動作に戻ります。

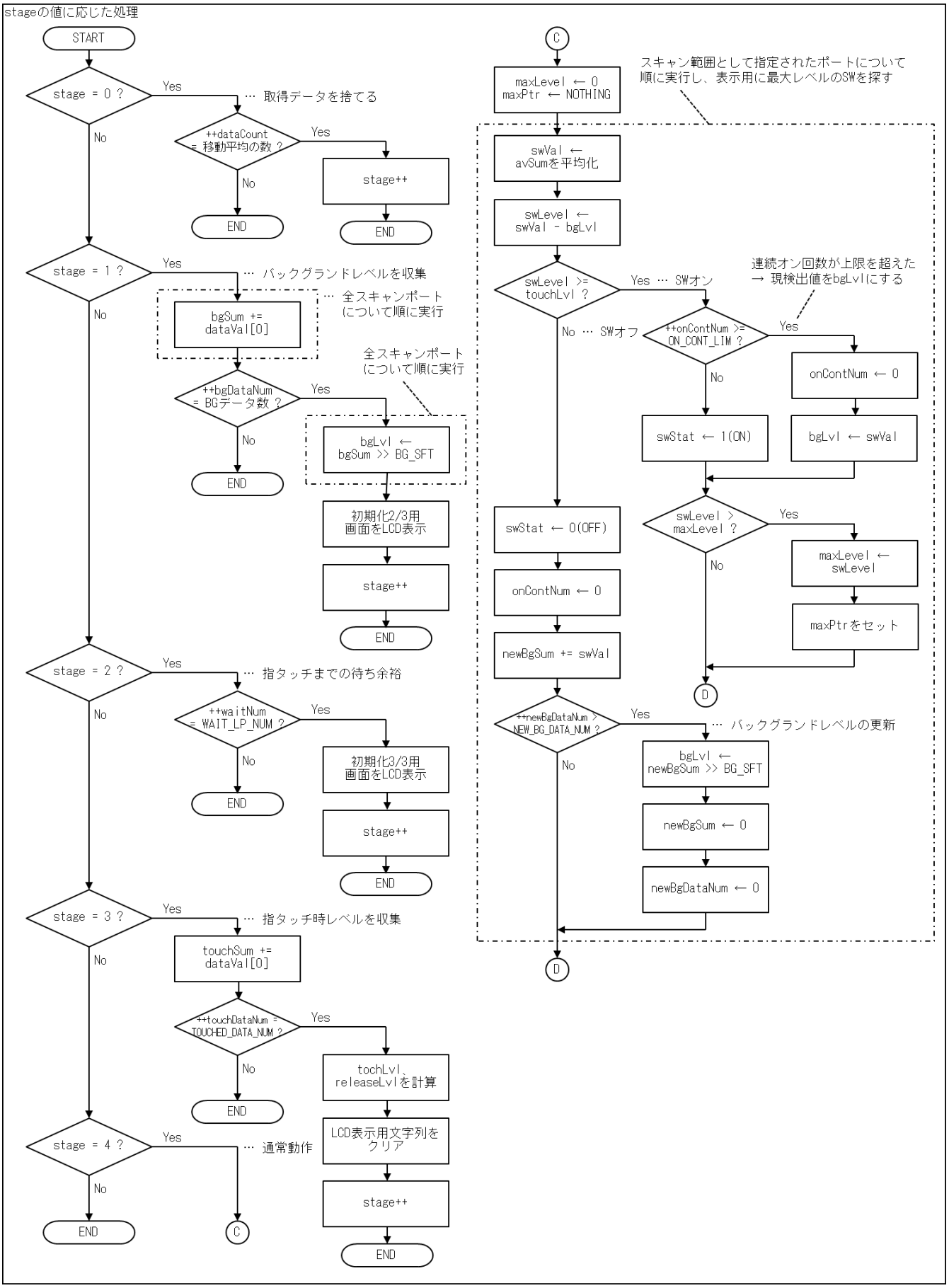
以下の 2 つの関数については、s1c17w22_w23_touch_oscillo/src/main.c と全く同じ処理フローで、構造体の変数名のみ、to から ts に変更になります。従い、フローチャートは掲載していません。

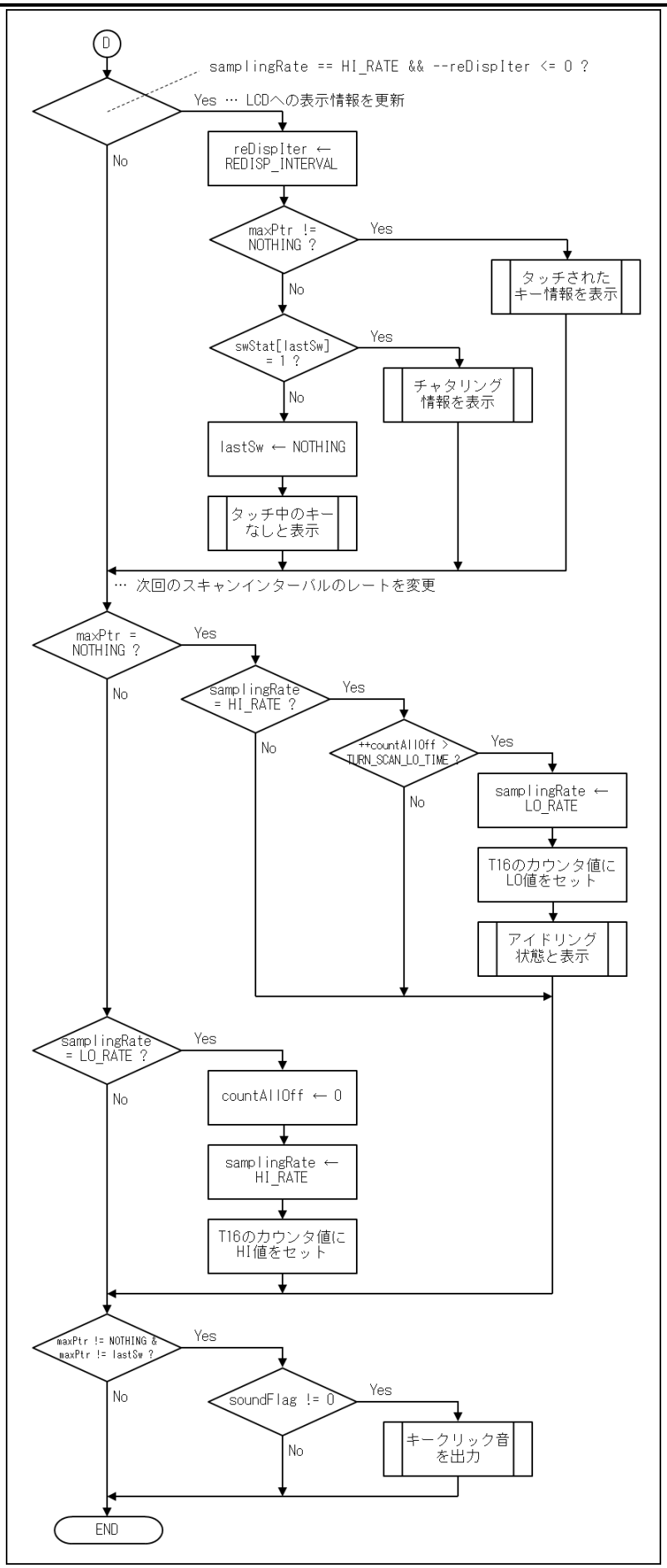
○ T16B Ch.0 割り込み時の処理（intT16bCh0 関数）

○ 容量相当値の測定（measCap 関数）

その他の処理のフローチャートを以下に記します。


初期化～T16Ch0割り込み待ち
(main関数)
main
割り込みレベルを
セット
変数を初期化
OSC1を発振開始
LCDを初期化、
表示行をクリア
SNDAを初期化、
スタート
T16-Ch0、T16B-Ch0
を初期化
T16-Ch0をスタート
初期化1/3用
画面をLCD表示
システムクロックを
OSC1に変更
IOSCを停止
HALT
T16 Ch.0 割り込み時の処理
(intT16Ch0関数)
割込発生
システムクロックを
OSC3に変更
割り込みフラグを
クリア
samplingRate
= LO_RATE ?
Yes
No
スキャン範囲を
全キーに
スキャン範囲を
単一キーに
… スキャン範囲として
指定されたポート
について順に実行
avSumの一番古い
データを取り去る
dataValをシフト
容量相当値を
測定
… measCap関数を呼ぶ
dataVal←tp+tn
avSumにデータを加
える
stageの値に
応じた処理
… 「stageの値に応じた処理」
で動作説明
システムクロックを
OSC1に変更
OSC3を停止
RETURN


stageの値に応じた処理
START
stage = 0 ?
Yes
… 取得データを捨てる
No
++dataCount = 移動平均の数 ?
Yes
No
stage++
END
END
stage = 1 ?
Yes
… バックグランドレベルを収集
No
… 全スキャンポートについて順に実行
bgSum += dataVal[0]
全スキャンポートについて順に実行
++bgDataNum = BGデータ数 ?
Yes
No
bgLvl ← bgSum >> BG_SFT
END
初期化2/3用画面をLCD表示
stage++
END
stage = 2 ?
Yes
… 指タッチまでの待ち余裕
No
++waitNum = WAIT_LP_NUM ?
Yes
No
初期化3/3用画面をLCD表示
END
stage++
END
stage = 3 ?
Yes
… 指タッチ時レベルを収集
No
touchSum += dataVal[0]
++touchDataNum = TOUCHED_DATA_NUM ?
Yes
No
tochLvl、releaseLvlを計算
END
LCD表示用文字列をクリア
stage = 4 ?
Yes
… 通常動作
No
stage++
END
C
END
C
maxLevel ← 0
maxPtr ← NOTHING
スキャン範囲として指定されたポートについて順に実行し、表示用に最大レベルのSWを探す
swVal ← avSumを平均化
swLevel ← swVal - bgLvl
swLevel >= touchLvl ?
Yes … SWオン
No … SWオフ
連続オン回数が上限を超えた → 現検出値をbgLvlにする
++onContNum >= ON_CONT_LIM ?
Yes
No
onContNum ← 0
bgLvl ← swVal
swStat ← 1(ON)
swLevel > maxLevel ?
Yes
No
maxLevel ← swLevel
maxPtrをセット
D
swStat ← 0(OFF)
onContNum ← 0
newBgSum += swVal
++newBgDataNum > NEW_BG_DATA_NUM ?
Yes
… バックグランドレベルの更新
No
bgLvl ← newBgSum >> BG_SFT
newBgSum ← 0
newBgDataNum ← 0
D

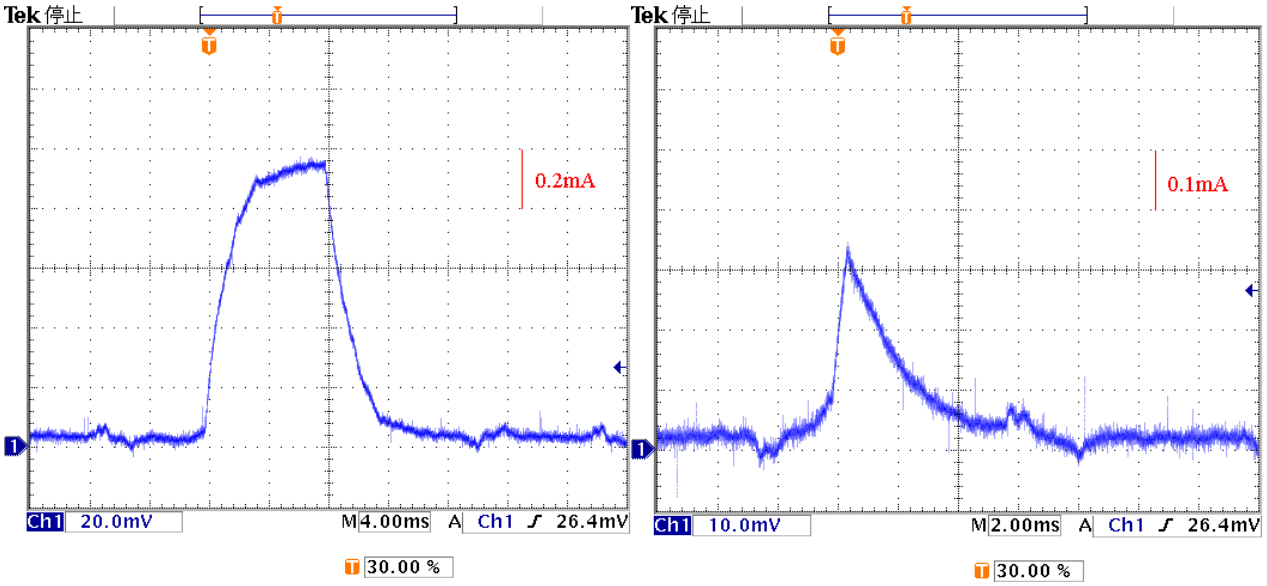

D
samplingRate == HI_RATE && --reDisplter <= 0 ?
Yes … LCDへの表示情報を更新
No
reDisplter ← REDISP_INTERVAL
maxPtr != NOTHING ?
Yes
タッチされたキー情報を表示
No
swStat[lastSw] = 1 ?
Yes
チャタリング情報を表示
No
lastSw ← NOTHING
タッチ中のキーなしと表示
… 次回のスキャンインターバルのレートを変更
maxPtr = NOTHING ?
Yes
No
samplingRate = HI_RATE ?
Yes
No
++countAllOff > TURN_SCAN_LO_TIME ?
Yes
No
samplingRate ← LO_RATE
T16のカウンタ値にLO値をセット
アイドリング状態と表示
samplingRate = LO_RATE ?
Yes
No
countAllOff ← 0
samplingRate ← HI_RATE
T16のカウンタ値にHI値をセット
maxPtr != NOTHING & maxPtr != lastSw ?
Yes
No
soundFlag != 0
Yes
No
キークリック音を出力
END

### 6.2.6 電池寿命の見積もり

サンプルプログラム s1c17w22_w23_touch_demo を SVT17W23 上で実行した時の消費電流の実測値から、ボタン電池で駆動した場合の電池寿命を見積もりました。その際の計算条件は以下の通りです。

- ボタン電池： CR2023（公称容量=220mAh） 1 個
- タッチキー読み取り時間： 1 日あたり 30 分間

タッチキー読み取り時、1 秒間に 30 回、全 16 個のキーに対してスキャンしていますが、その際の電流波形のモニター結果を図 6-2(a)に示します。またアイドル状態の時は、1 秒間に 5 回、1 個のキーのみタッチセンシングしていますが、その際の電流波形のモニター結果を図 6-2(b)に示します。




(a) タッチキー読み取り時　　(b) アイドル状態の時

図 6-2　電流波形のモニター結果

(1) タッチキー読み取り時
0.94mA×0.009s×1000/33.3 = 0.25mA

(2) アイドル状態の時
0.35mA×0.006s×1000/200 = 0.0105mA　と、　0.5uA（$I_{HALT2}$: OSC1 で HALT）

上記より、
220mAh÷(0.25mA×0.5h + (0.0105mA＋0.0005mA)×24h) = 566 日 ≒ 1.55 年

# 改訂履歴表

付－1

| Rev. No. | 日付 | ページ | 種別 | 改訂内容（旧内容を含む）<br>および改訂理由 |
|---|---|---|---|---|
| Rev 1.0 | 2014/05/23 | 全ページ | 新規 | 新規作成 |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

セイコーエプソン株式会社
マイクロデバイス事業部　IC 営業部

東京　〒191-8501　東京都日野市日野 421-8
TEL（042）587-5313（直通）　FAX（042）587-5116

大阪　〒541-0059　大阪市中央区博労町 3-5-1　エプソン大阪ビル 15F
TEL（06）6120-6000（代表）　FAX（06）6120-6100

ドキュメントコード：412786200
2014 年 6 月　作成