QTouch®部品化ライブラリPTC

QTouch®部品化ライブラリ周辺機能接触制御器使用者の手引き

説明

Microchip QTouch®周辺機能接触制御器(PTC:Peripheral Touch Controller)は釦、摺動子、輪として機能する感知器での容量性接触測定用の組み込みﾊｰﾄﾞｳｪｱを提供します。PTCはどの外部部品の必要もなしに相互容量と自己容量の両測定を支援します。これは素晴らしい感度と雑音耐性だけなく、自己校正も提供し、使用者による感度調整の労力を最小にします。これは容量性接触面と手ぶり機能に対する支援にも広がります。

PTCは自律的に実行する容量性接触感知器測定に対して意図されています。外部容量性接触感知器は代表的にPCBで形成され、感知器電極はﾃﾞﾊﾞｲｽの入出力ﾋﾟﾝを使ってPTCのｱﾅﾛｸﾞ電荷積分器に接続されます。PTCは酸化ｲﾝｼﾞｳﾑ錫(ITO:Indium Tin Oxide)を含む各種X-Y構成設定での容量性接触配列として構成される相互容量感知器を支援します。相互容量感知器ではPTCがX線(駆動線)毎に1つのﾋﾟﾝとY線(感知線)毎に1つのﾋﾟﾝを必要とします。自己容量ではPTCが各自己容量感知器に対してY線駆動部で1つのﾋﾟﾝだけを必要とします。

要点

- 低電力、高感度、環境的に頑強な容量性接触釦の実装
- 相互容量と自己容量の検知を支援
- 自己容量動作で46個までの釦
- 相互容量動作で529個までの釦
- 一括動作構成設定を支援
- 電極毎に1つのﾋﾟﾝ – 外部部品なし
- 負荷補償容量検知
- 相互容量動作に対する規制容量補償
- 優れた感度のために調整可能な利得
- 温度とVDDの範囲に渡る変動0
- 温度やVDDの補償の必要なし
- 高い伝導耐性のためのﾊｰﾄﾞｳｪｱ雑音濾波器と雑音信号脱同期化
- Atmel | START QTouch構成部支援 – ｳｨｻﾞｰﾄﾞで案内される接触ﾌﾟﾛｼﾞｪｸﾄ作成

製品支援

QTouch容量性接触感知ｿﾌﾄｳｪｱ ﾗｲﾌﾞﾗﾘに関連する支援と関連する問題については最寄りのMicrochip代理店に問い合わせるか、またはhttps://www.microchip.com/support/を訪ねてください。

本書は一般の方々の便宜のため有志により作成されたもので、Microchip社とは無関係であることを御承知ください。しおりの[はじめに]での内容にご注意ください。

目次

1. 序説

QTouch®部品化ﾗｲﾌﾞﾗﾘ(QTML:QTouch Modular Library)は部品化構造下のQTouchﾗｲﾌﾞﾗﾘの接触感知機能を提供します。ﾗｲﾌﾞﾗﾘを機能的な単位部に分けることにより、応用開発者は目的対象応用に関連する機能を提供するそれらの単位部だけを含むことができ、それによってﾃﾞﾊﾞｲｽのﾒﾓﾘと処理時間の両方を節約します。

2. 容量性接触測定

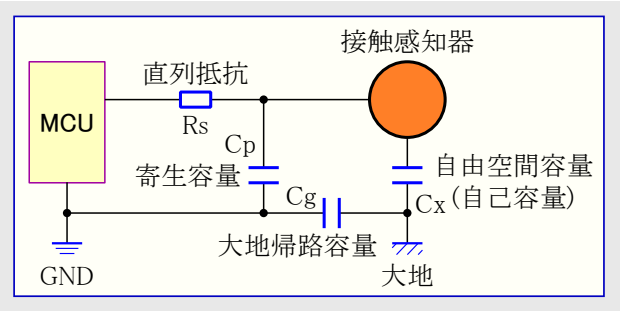
QTouch部品化ﾗｲﾌﾞﾗﾘはAVR®、Arm® Cortex-M0+、Arm Cortex-M23、Arm Cortex0M4のﾏｲｸﾛ ｺﾝﾄﾛｰﾗの選択で自己容量と相互容量の接触感知器のPTC測定を支援します。

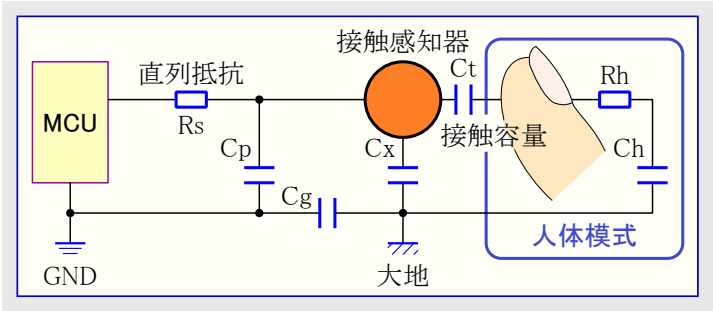
現在の全ての容量性接触測定法では2つの基本機能的な手法の自己容量と相互容量の1つが実装されます。

2.1. 自己容量

自己容量は電極と接触感知器MCU回路のDC接地間の見かけ上の容量を測定するのに単一感知器電極を使う容量測定を示します。

電源ONまたはﾘｾｯﾄで容量の基準値測定が記録され、'非接触'容量に仮定されます。参照基準容量はCgとCxの直列対と並列のCpの組み合わせです。

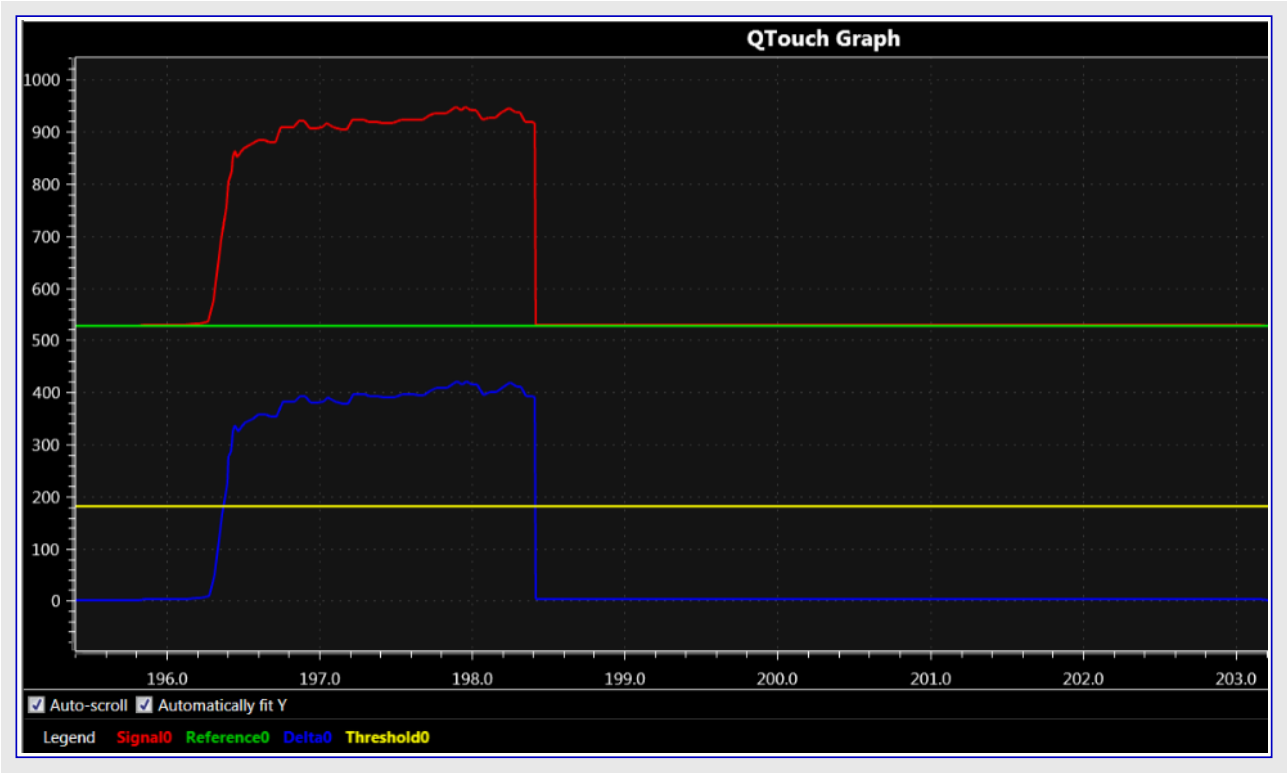




接触接点が加えられると、CtとChの直列組み合わせ経由で大地への並列経路の導入によって容量が増されます。増加は接触閾値と比較され、超えれば感知器は'接触'を示されます。

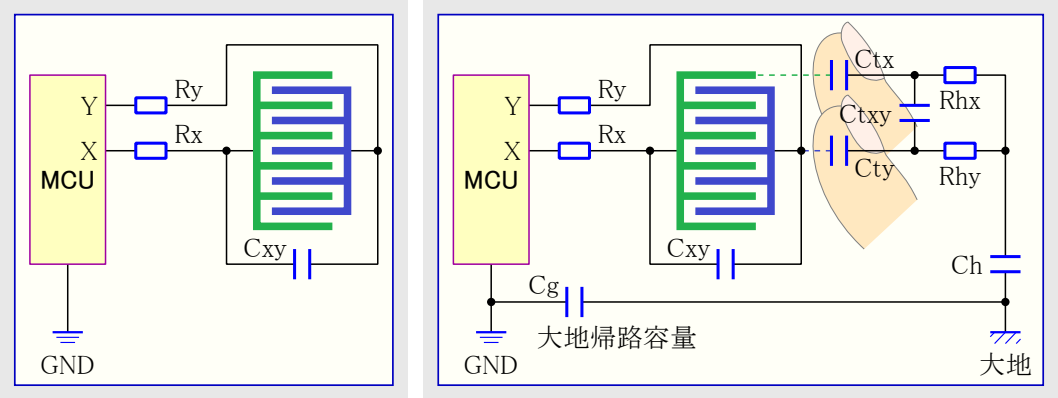
注: 人体容量(Cx)は人の周辺によって変わり、代表的に概略100～200pFです。けれども、接触接点容量(Ct)は代表的に1～5pFでより一貫してもっと小さく、主に接触感知器の設計と構造、そして次に感知器を活性(有効)にするのに使われる指の大きさに依存します。

直列容量対ではより小さなものが優性な成分なので、この場合のCtに於いて、上手く設計されて調整された感知器は使用者での小さな依存性に於いて接触接点に対する非常に一貫した感度を示します。



2.2. 相互容量

相互容量はそれら間の見かけ上の容量を測定するのに感知器電極の対を使う容量測定を示します。代表的に、1つの電極は駆動部(X)、同時に他は受信部(Y)として働きます。X電極がY電極に電荷を転送する各物理位置が感知器節点で、これは接触感度の位置です。



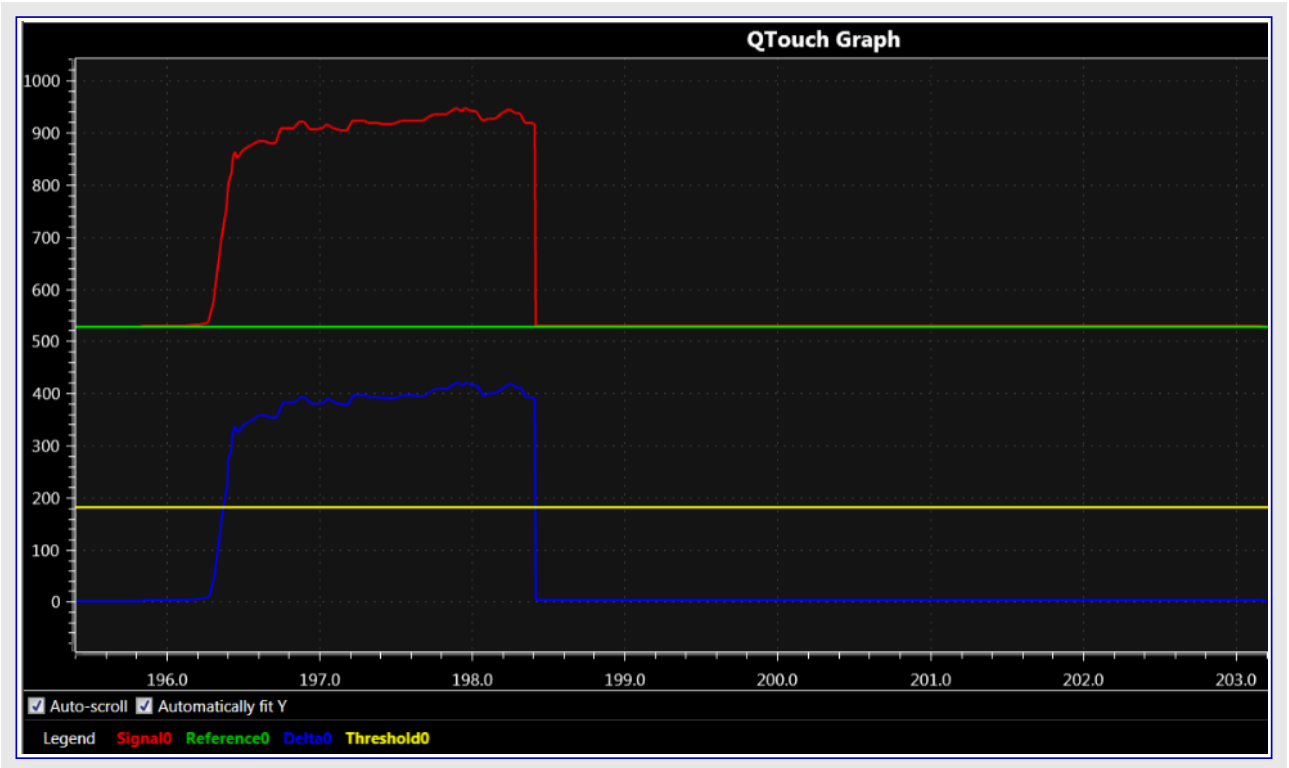
相互容量感知器と人体間の相互作用はもっと複雑です。それはXとYの電極に対する2つの独立した接触接点を考察することによって模式化することができ、そこで各々は人体と容量的に結合され、体内で互いに抵抗的に接続され、容量的に人体容量経由で大地に結合されます。

接触接点は以下の2つの矛盾する効果を持ちます。

- XとYの電極への導電板(指)の導入はXとY間の容量を増します。これは何れかの導電部が感知器上に置かれる場合に起こります。
- XY節点で別の容量(Ch+Cg)の追加はX電極によって放出されるエネルギーに対する代替経路を提供し、感知器上で累積される電荷量を減らし、そして導電部に接続される材料の本体がかなりの自己容量を持つ場合にだけ起きます。

実際の接触接点が置かれる時に、2つ目の(減少)効果が最初の(増加)効果よりももっと大きく、故に相互容量感知器での接触接点は感知器容量で見かけ上の減少によって示されます。

この容量での見かけ上の電荷(差)が構成設定した接触閾値と比較され、閾値を超える場合、感知器が検出されたと判断されます。



3. 接触感知器

容量性感知器は釦の置き換えとして単純に接触を検出するように、または相対的な距離の測定(近接)を提供するために機能的に拡張され、1D位置(摺動子または輪)、2D位置(QTouch表面)、3D位置(近接付きQTouch表面)に実装することができます。

各々の場合で、部品化ﾗｲﾌﾞﾗﾘは予め定義された閾値を超える容量での変化によって接触接点を検出します。一旦接触が確認されると、様々な後処理単位部が隣接する感知器間を補間するために計算された接触差(Delta)を使い、接触位置の場所や相対的な近さを計算します。

3.1. 釦

容量性感知器の最も簡単な実装が釦で、そこでは感知部が単一節点(自己容量に対しては1つの電極、相互容量に対しては1つの電極対)から成り、検出または非検出の2値状態として解釈されます。

3.2. 近接感知器

釦の拡張が近接感知器です。単一感知器節点は予め構成設定された閾値を超える容量での電荷に対して監視されます。釦と同じ方法で、感知器はその閾値を超える時に'検出'と見做されます。一旦検出になると、初期'検出'閾値と2つ目の'完全接触'閾値の2つの閾値間で接触差を測ることによって相対的な接触(近接)距離測定が行われます。

注: 近接感知は離れた物体の容量性負荷を頼るため、接触に対する'見かけ上の距離'は接触の形状や大きさに依存します。

即ち、10cmで開いた手は、同じ距離でより大きな表面領域なのでそれが容量でより大きな影響を持つため、10cmで広げた指よりも近くに現れます。

容量(C)は領域(A)に比例し、距離(d)に反比例します。また、設置は感度にかなりの影響を持ち、故に範囲にもです。

$$C \propto \frac{A}{d}$$

3.3. 一括感知器

一括感知器は1つの単一感知器として働くように、複数の感知線(自己容量測定)または複数の駆動と感知の線(相互容量測定)の組み合わせとして実装されます。これは接触感知器実装に於いてより大きな柔軟性を応用開発者に提供します。

- 測定数を減らすことによって接触感知器応答、従って初期接触検出に対して必要な時間を改善
- 2分検索による高速位置分解能
- 接触釦応用での'全部だけれども1つ'ｷｰ一括を通して改善された水分除去
- 全ｷｰに対して1つだけの感知器測定が必要とされるためにかなり低い電力消費で(最大容量制限までの)どのｷｰでも接触起き上がり機能を提供
- 二重目的感知器電極 – 例えば、個別ｷｰは近接感知器を形成するために共に一括にされるかもしれません。

一括検知器での接触検出は単一節点接触釦と同じ方法で実装されます。一括感知器の容量は個別感知器の容量以上です。多すぎに監視器の一括化は飽和に帰着するかもしれません。一般的に、自己容量感知器の容量は相互容量感知器よりも高いです。一括化することができる感知器電極数はに自己容量設計について相対的に減ります。

3.4. 補間された感知器

補間された感知器はその列に沿った接触接点の位置を計算するために列で配列された2つ以上の隣接感知器節点の接触差を利用します。感知器の配置が設計され、閾値は感知器に沿って何処かを接触する方法が以下を起こすように構成設定されます。

1. 最低1つの感知器節点で閾値を超える接触差。最強の接触差を持つ節点が接触接点の中心節点と判断され、接触接点の適切な位置が識別されます。
2. 節点間の位置補完に使われる隣接節点でのいくつかの接触差。中心節点の左右に対する節点での相対差は最強差を持つ側に向かって計算された接触位置を調整するのに使われます。

補間された感知器は最後の感知器から最初への丸めのありまたはなしでどの物理形状に形成することもできます。丸まった感知器は'輪'として構成設定され、一方でそうでないものは'摺動子'として構成設定されます。輪の場合、先頭ｷｰでの中心接触接点は'左'補完に対して最後のｷｰを使い、その逆もで、また一方で摺動子任意選択は最後に不感帯を実装します。

3.5. 2D位置感知器

ｷｰの格子を通す2D位置検出に対して直線感知器が物理的にｷｰの一直線として実装されるのと同じ手法を拡張することができます。ｷｰは補間が垂直または水平のどちらの方向でも行うことができ、2つの独立した接触接点はそれらの補完された位置で個別に解決されます。

3.6. 混合と一致

QTouch部品化ﾗｲﾌﾞﾗﾘは或る程度の前例のない各種感知器型と測定技法を実装する組み合わせを許し、多くの場合では同じﾌｧｰﾑｳｪｱ応用内で複数の方法で同じ感知器電極を利用します。

例えば、相互容量ｷｰ感知器を使う2D位置感知器は近接測定を提供するために相互容量動作で一括または部分的な一括に、そして耐水性改善するために自己容量で個別に測定されるY線にすることができます。

4. PTC

4.1. 概要

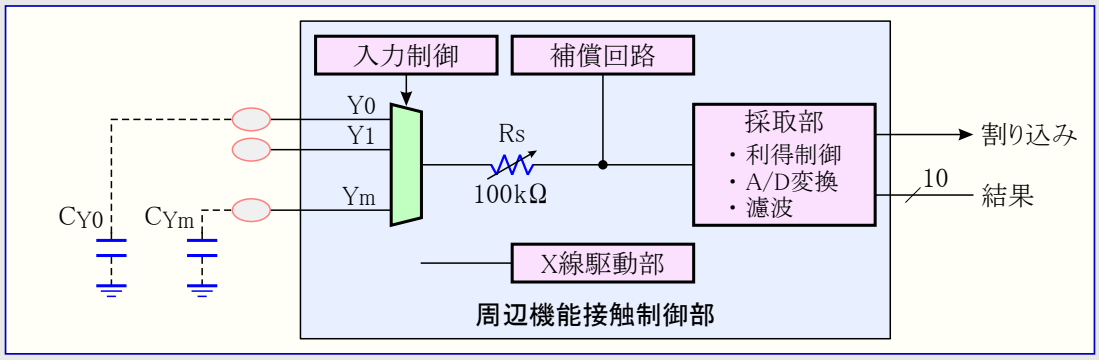
Microchip QTouch®周辺機能接触制御器(PTC:Peripheral Touch Controller)は釦、摺動子、輪として機能する感知器での容量性接触測定用の組み込みﾊｰﾄﾞｳｪｱを提供します。PTCはどの外部部品の必要もなしに相互容量と自己容量の両測定を支援します。これは素晴らしい感度と雑音耐性だけなく、自己校正も提供し、使用者による感度調整の労力を最小にします。

PTCは自律的に実行する容量性接触感知器測定に対して意図されています。外部容量性接触感知器は代表的にPCBで形成され、感知器電極はﾃﾞﾊﾞｲｽの入出力ﾋﾟﾝを使ってPTCのｱﾅﾛｸﾞ電荷積分器に接続されます。PTCは酸化ｲﾝｼﾞｳﾑ錫(ITO:Indium Tin Oxide)を含む各種X-Y構成設定での容量性接触配列として構成される相互容量感知器を支援します。

4.2. 自己容量

自己容量ではPTCが各自己容量感知器に対してY線駆動部で1つのﾋﾟﾝだけを必要とします。

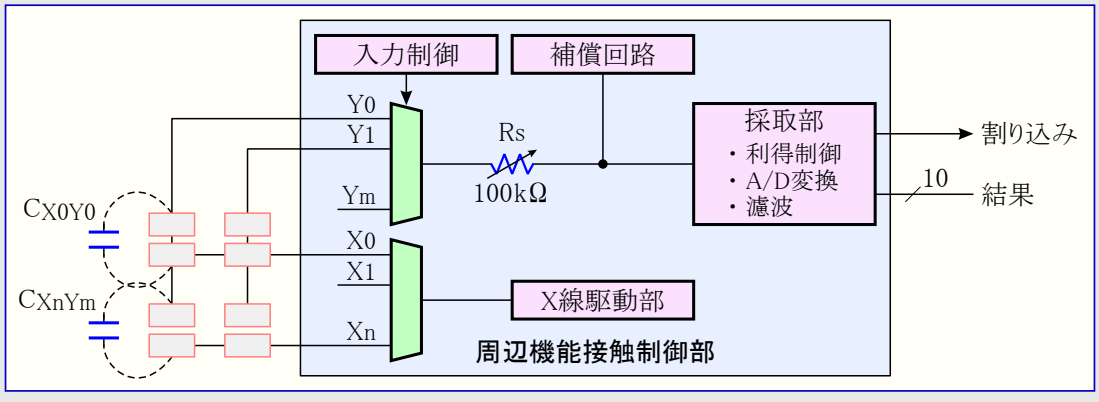
図4-1. 自己容量PTC測定



4.3. 相互容量

相互容量感知器ではPTCがX線(駆動線)毎に1つのﾋﾟﾝとY線(感知線)毎に1つのﾋﾟﾝを必要とします。

図4-2. 相互容量PTC測定



5. QTouch®部品化ﾗｲﾌﾞﾗﾘ

5.1. 序説

QTouch部品化ﾗｲﾌﾞﾗﾘは再設計された部品化構造下のQTouchﾗｲﾌﾞﾗﾘの接触感知機能を提供します。ﾗｲﾌﾞﾗﾘを機能的な単位部に分けることにより、応用開発者に対して目的対象応用に関連する機能を提供するそれらの単位部だけを含むことが可能で、それによってﾃﾞﾊﾞｲｽのﾒﾓﾘと処理時間の両方を節約します。

5.2. QTouch®ﾗｲﾌﾞﾗﾘ単位部

QTouchﾗｲﾌﾞﾗﾘ単位部は下で示されるように機能に基づいて3つの型に分類することができます。

採取単位部	信号調整単位部	後処理単位部
採取自動調節単位部	周波数飛び跳ね単位部	接触ｷｰ単位部
採取走行時単位部	周波数飛び跳ね自動調節単位部	移動器単位部
• 接触測定 • ﾁｬﾈﾙ順序付け • CC校正 • 前置分周器/直列抵抗･充電共用遅延の自動/手動校正	• 飛び跳ね周波数 • 中央値濾波器 • 雑音測定 • 雑音に基づく充電周波数	• 釦後処理 • 変動、検出の積分 • AKS群 • 摺動子/輪の位置 • 接触の活性/不活性の状態 • ﾋｽﾃﾘｼｽ、IIR濾波

5.3. 単位部命名規則

QTouchﾗｲﾌﾞﾗﾘ単位部が従う命名規則が下で与えられます。

qtm _〈単位部名識別子〉_〈ﾃﾞﾊﾞｲｽ基本構造〉_〈単位部ID〉.〈ﾌｧｲﾙ拡張子〉

qtm / libqtm	QTouch単位部を示略語。全てのQTouch単位部は容易な識別のために"qtm_"で始まります。 GCC単位部については単位部名に"lib"が先頭に追加され、従って"libqtm"になります。
単位部名識別子	acq – 自動調節付き採取単位部 acq_runtime – 自動調節ｺｰﾄﾞなし採取単位部 freq_hop – 周波数飛び跳ね単位部 freq_hop_auto_tune – 自動調節付き周波数飛び跳ね単位部
ﾃﾞﾊﾞｲｽ基本構造	cm0p - 全てのCortex M0+後処理用単位部 cm4 - 全てのCortex M4F後処理用単位部 samd1x - SAM D10/D11採取単位部専用 t81x - AVR tiny817ﾃﾞﾊﾞｲｽ系統の全単位部 t161x – AVR tiny1617ﾃﾞﾊﾞｲｽ系統の全単位部 t321x – AVR tiny3217ﾃﾞﾊﾞｲｽ系統の全単位部 m328pb – AVR mega328PBﾃﾞﾊﾞｲｽの全単位部 m324pb – AVR mega324PBﾃﾞﾊﾞｲｽの全単位部 saml21 – SAM L21採取単位部専用 saml22 – SAM L22採取単位部専用 samc21 – SAM C21採取単位部専用 samc20 – SAM C20採取単位部専用 samd21 – SAM D21採取単位部専用 samda1 – SAM DA1採取単位部専用 samha1 – SAM HA1採取単位部専用 samd20 – SAM D20採取単位部専用 saml10 – SAM L10採取単位部専用 saml11 – SAM L11採取単位部専用 cm23 – 全cortex M23後処理単位部用
単位部ID	各単位部に対して一意の16ﾋﾞｯﾄ識別子
ﾌｧｲﾙ拡張子	.a – AVR®とArm®ﾃﾞﾊﾞｲｽのGCC単位部、Armﾃﾞﾊﾞｲｽの IAR単位部 .r90 – 全てのAVR単位部のIAR単位部

下の例をご覧ください。

表5-1. 例

ｺﾝﾊﾟｲﾗ＼例	AVR® mega328PBﾃﾞﾊﾞｲｽの採取単位部	SAM D10/D11ﾃﾞﾊﾞｲｽの接触ｷｰ処理単位部	SAM L1xﾃﾞﾊﾞｲｽの接触ｷｰ処理単位部
GCC単位部	libqtm_acq_m328pb_0x0001.a	libqtm_touch_keys_cm0p_0x0002.a	libqtm_touch_key_cm23_0x0002.a
IAR単位部	qtm_acq_m328pb_0x0001.r90	qtm_touch_keys_cm0p_0x0002.a	qtm_touch_key_cm23_0x0002.a

5.4. QTouch®ﾗｲﾌﾞﾗﾘ応用ｲﾝﾀｰﾌｪｰｽ

ﾗｲﾌﾞﾗﾘ単位部に加えて、完全な接触応用を構築するのに必要とされる様々な部品が下で与えられます。

1. 単位部APIﾌｧｲﾙ
2. Touch.cとTouch.hのﾌｧｲﾙ
3. Common_components_api.h
4. Touch_api_ptc.h
5. 単位部再収集ﾌﾗｸﾞ
6. 結合層単位部

5.4.1. 単位部APIﾌｧｲﾙ

各単位部用のAPIは関連するﾍｯﾀﾞ ﾌｧｲﾙで定義されます。単位部間の依存性は最小化され、応用ﾚﾍﾞﾙで実装されます。これは或るﾃﾞﾊﾞｲｽから別のものへ応用ｺｰﾄﾞの容易な移植を許し、ﾊｰﾄﾞｳｪｱ依存単位部だけが調整されなければなりません。採取自動調節と採取手動調節の単位部は同じAPIﾌｧｲﾙです。他の全ての単位部は使用者応用にﾘﾝｸされることが必要なそれらのAPIﾌｧｲﾙを持ちます。

5.4.2. Touch.cとTouch.hのﾌｧｲﾙ

各単位部に対する使用者任意選択は応用ｺｰﾄﾞで構成設定され、一般的にtouch.hとtouch.cはﾎﾟｲﾝﾀ参照によってﾗｲﾌﾞﾗﾘ単位部で共用されます。同様に、配列は単位部の実行時ﾃﾞｰﾀに対して応用ｺｰﾄﾞで作成され、ﾎﾟｲﾝﾀ経由で単位部に提供されます。

構成設定は接触感知器の測定掃引間で応用ｺｰﾄﾞによって実行中に変更されるかもしれません。全ての実行時ﾃﾞｰﾀは応用ｺｰﾄﾞに対して利用可能です。

5.4.3. Common_components_api.h

応用は全ての単位部に共通する構造値と定義を必要とします。共通する定義、ﾏｸﾛ、構造体は”Common_components_api.h”ﾌｧｲﾙに置かれます。

5.4.4. Touch_api_ptc.h

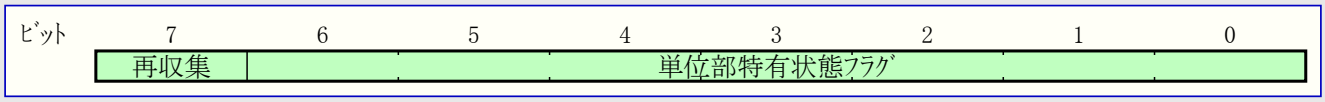
このﾌｧｲﾙは内容に於いて全ての単位部APIﾌｧｲﾙを含み、従って必要な状況に応じて応用ｿｰｽ ﾌｧｲﾙでこの単一ﾌｧｲﾙをｲﾝｸﾙｰﾄﾞすることで充分です。

5.4.5. 単位部再収集ﾌﾗｸﾞ

単位部の構成設定と機能は各単位部に対して固有ですが、おそらくどの単位部も特定感知器の繰り返し測定を必要とします。これを達成すのに、信号調整単位部は採取構成設定を一時的に変更、例えば、再収集を必要としないそれらの感知器を禁止するかもしれません。

これは信号調整群ﾃﾞｰﾀ構造体の最初の位置で共通’status(状態)’ﾊﾞｲﾄの実装によって応用に示されます。ﾋﾞｯﾄ7での’1’は測定周回時間経過を待つことなく、応用が感知器群での測定を再び開始すべきことを示します。

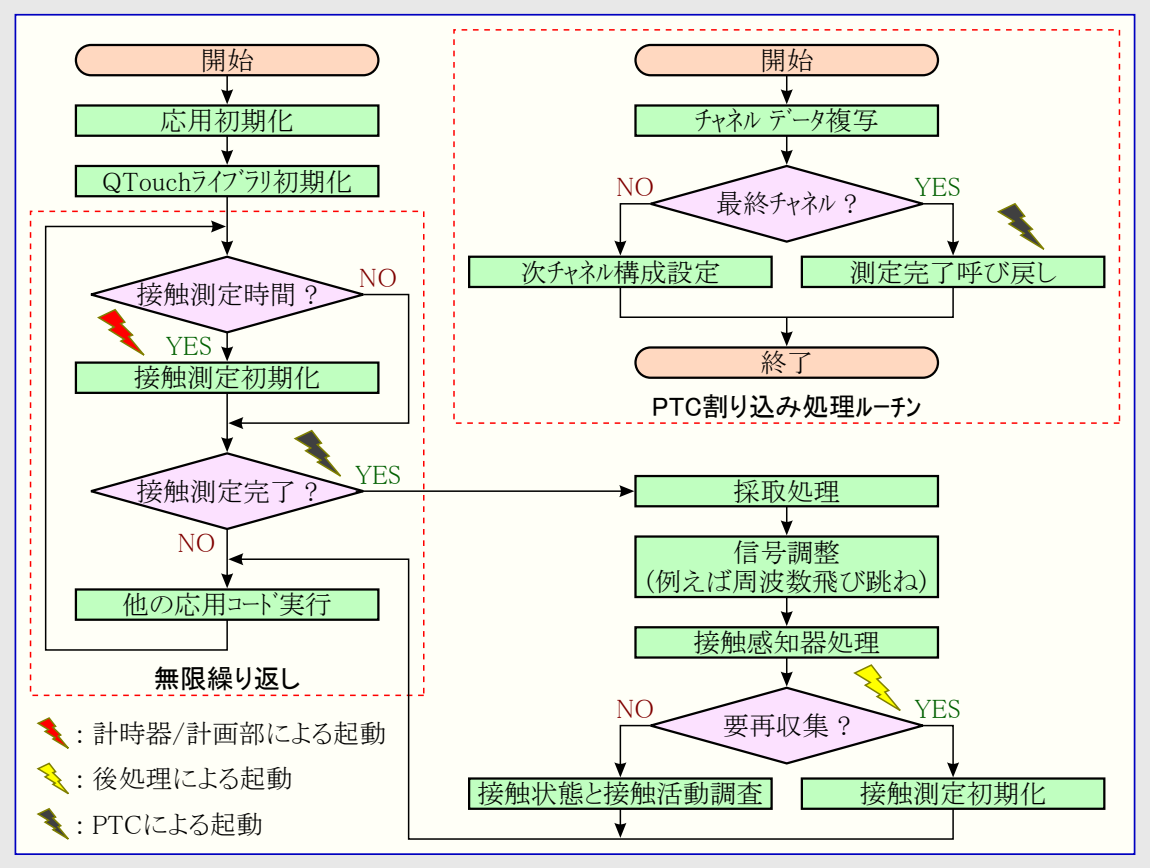
図5-1. uint8_t qtm_xxx_status



5.4.6. 結合層単位部

結合層単位部は使用者応用にQTouch単位部の容易なｲﾝﾀｰﾌｪｰｽを提供します。結合層は最小のAPI関数を使って適切な順序で構成設定された全ての単位部を結合します。これは単位部の初期化を処理し、手続き呼び出しを同期化し、そして異常状態を処理します。

5.5. 応用の流れ



5.6. MISRA準拠

QTouchﾗｲﾌﾞﾗﾘ単位部のｿｰｽ ｺｰﾄﾞは以下の例外付きでMISRA 2004の'required(必要とされる)'規則一式に従います。

表5-2. AVR® MCU採取単位部と例外

ATmega32xPB、ATtiny81x、ATtiny161x、ATtiny321xﾃﾞﾊﾞｲｽの採取単位部		
MISRA規則	定義	備考
1.1	全てのｺｰﾄﾞはISO/IEC 9899/COR1:1995、ISO/IEC9899/AMD1:1995、ISO/IEC 9899/COR2:1996によって改正/修正されたISO 9899:1990ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ言語-Cに従うべきです。	ｺﾝﾊﾟｲﾗは拡張を許すように構成設定されます。
8.5	ﾍｯﾀﾞ ﾌｧｲﾙにｵﾌﾞｼﾞｪｸﾄや関数の定義があるべきではありません。	ﾍｯﾀﾞ ﾌｧｲﾙでｲﾝﾗｲﾝ関数が使われます。
17.4	配列指標は許されたﾎﾟｲﾝﾀ演算の形式だけにすべきです。	単位部ﾃﾞｰﾀ構造体のﾎﾟｲﾝﾀはﾊﾟﾗﾒｰﾀとして渡され、個別のｵﾌﾞｼﾞｪｸﾄ ﾃﾞｰﾀは配列指標としてﾃﾞｰﾀ構造体を繰り返すことによって取得されます。

表5-3. AVR®後処理単位部と例外

接触ｷｰ、結合層、周波数飛び跳ね自動調節、周波数飛び跳ね、移動部、2D接触表面、手ぶり		
MISRA規則	定義	備考
17.4	配列指標は許されたﾎﾟｲﾝﾀ演算の形式だけにすべきです。	単位部ﾃﾞｰﾀ構造体のﾎﾟｲﾝﾀはﾊﾟﾗﾒｰﾀとして渡され、個別のｵﾌﾞｼﾞｪｸﾄ ﾃﾞｰﾀは配列指標としてﾃﾞｰﾀ構造体を繰り返すことによって取得されます。

表5-4. Arm®採取単位部と後処理単位部

単位部		
MISRA規則	定義	備考
1.1	全てのｺｰﾄﾞはISO/IEC 9899/COR1:1995、ISO/IEC9899/AMD1:1995、ISO/IEC 9899/COR2:1996によって改正/修正されたISO 9899:1990ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ言語-Cに従うべきです。	ｺﾝﾊﾟｲﾗは拡張を許すように構成設定されます。
17.4	配列指標は許されたﾎﾟｲﾝﾀ演算の形式だけにすべきです。	単位部ﾃﾞｰﾀ構造体のﾎﾟｲﾝﾀはﾊﾟﾗﾒｰﾀとして渡され、個別のｵﾌﾞｼﾞｪｸﾄ ﾃﾞｰﾀは配列指標としてﾃﾞｰﾀ構造体を繰り返すことによって取得されます。

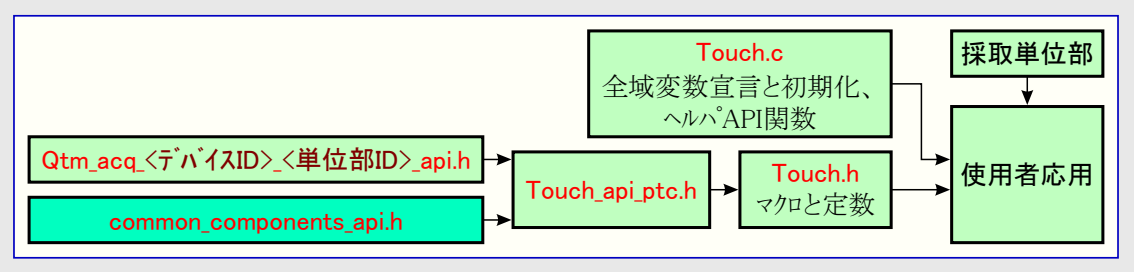
6. 採取単位部

6.1. 概要

接触感知器応用に対する最小要件は採取単位部で、これは容量性の接触や近接の感知器の構成設定と測定に対する全てのﾊｰﾄﾞｳｪｱ依存操作を実装します。

6.2. インターフェース

ﾃﾞｰﾀ構造体定義とAPI宣言はAPIﾌｧｲﾙの"qtm_acq_<ﾃﾞﾊﾞｲｽID>_<単位部ID>_api.h"に含まれます。ﾃﾞｰﾀ構造体は全ての構成設定と出力ﾃﾞｰﾀ変数を網羅します。このﾌｧｲﾙは共通APIの"touch_ptc_api.h"ﾌｧｲﾙでｲﾝｸﾙｰﾄﾞされるべきです。

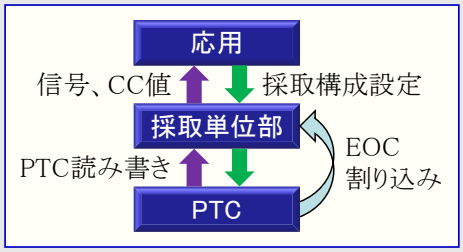


6.3. 機能的な説明

採取単位部は目的対象特有で、各々は接触感知技法と適用される方法に依存するﾊｰﾄﾞｳｪｱ構成設定構造体を持ちます。

この採取単位部に実装された機能

- 感知器節点用ﾊｰﾄﾞｳｪｱ校正
  - 感知器電極の時定数補償のための前置分周器/ﾚｼﾞｽﾀ/充電の共用遅延の校正
  - 感知器負荷を合わせるための内部補償回路の校正
- 標準手順での自己容量と相互容量の感知器接触測定
- 事象ｼｽﾃﾑを使う自動走査の低電力動作(現在Atmel START構成部で未支援)



6.4. 構成設定

6.4.1. ﾃﾞｰﾀ構造体

採取単位部は感知器容量の相対的な測定を行うのに機能的に必要とされる全てを実装します。これは接触感知器実装に使われるﾋﾟﾝをｱｸｾｽして制御しなければならないため、個別ﾃﾞﾊﾞｲｽに対して固有に構築された唯一の単位部です。

ﾃﾞﾊﾞｲｽが利用可能な異なるﾊｰﾄﾞｳｪｱ機能を持つため、各ﾃﾞﾊﾞｲｽで異なる構成設定任意選択が利用可能です。ｼｽﾃﾑ資源(ROMとRAM)の最も効率的な使用のため、各種感知器構成設定構造体が必要とされます。

けれども、構造体内で同じ変数名が使われる場合、その変数によって制御される機能は同じです。どの従属関数も変数に対する参照を利用し、構造体とﾎﾟｲﾝﾀ演算への参照に頼るべきではありません。

採取群構成設定

'&ptc_qtlib_acq_gen1.freq_option_select'へのポインタによる参照はデバイスに関わらず常に正しいメモリ位置を指し示します。けれども、ポインタ演算に基づくどの実装も、コードが或るデバイスから別のものへ再使用される場合にソース内部調整(移植作業)を必要とします。

パラメータ	大きさ	範囲/任意選択	使い方
num_sensor_nodes	16ビット	0～65535	群で構成設定した感知器節点数
acq_sensor_type	8ビット	NODE_SELFCAP NODE_MUTUAL	この節点群に適用する測定法を定義
calib_option_select	1バイト	**ビット7～4 – 校正目標:** CAL_CHRG_2TAU CAL_CHRG_3TAU CAL_CHRG_4TAU CAL_CHRG_5TAU	校正目標は許容電荷移動損失に対する限度を適用し、より高い目標の設定が電荷全体のより大きな割合が移動されることを保証します。
		ビット3～0 – 校正形式: CAL_AUTO_TUNE_NONE CAL_AUTO_TUNE_RSEL CAL_AUTO_TUNE_PRSC CAL_AUTO_TUNE_CSD*	校正形式は最適電荷移動のためにどのパラメータが自動的に調節されるべきかを選びます。
freq_option_select	1バイト	FREQ_SEL_0～FREQ_SEL_15 または FREQ_SEL_SPREAD	FREQ_SEL_0～FREQ_SEL_15は過採取中の測定間に遅延周期を挿入し、0が最短、15が最長の遅延です。 FREQ_SEL_SPREADは過採取設定中に鋸歯の規則でこの遅延を0～15まで変えます。
PTC_interrupt_priority**	1バイト	1～3	PTCに対する割り込み優先レベル

注: * – 全てのデバイスで利用可能な訳ではありません。
** – Arm® Cortexデバイスでだけ適用できます。

節点構成設定

同様に、節点構成設定構造体は使われるデバイスに応じて変わります。

- X線数
- Y線数
- 機能可用性

パラメータ	大きさ	範囲/任意選択	使い方
node_xmask	1/2/4/8バイト	(ビット領域)	X線番号に対応する位置でビットを設定。 例: X0だけ = 0b00000001 = $01 X0とX2 = 0b00000101 = $05 8つのX線までのデバイスには1バイトが使われます。 2バイト、4バイト、8バイトは各々16、32、46*本までのX線のデバイスに使われます。 注: * 最大64本までのX線を支援することができます。
node_ymask	1/2/4/8バイト	(ビット領域)	Y線番号に対応する位置でビットを設定。 例: Y5だけ = 0b00100000 = $20 Y1,Y2とX7 = 0b10000110 = $86 8つのY線までのデバイスには1バイトが使われます。 2バイト、4バイト、8バイトは各々16、32、46*本までのY線のデバイスに使われます。 注: * 最大64本までのY線を支援することができます。
node_csd*	1バイト	0～255	感知器節点容量の充電を確実にするための遅延周期数。(ATtiny,ATmegaのAVR®とSAM E54,SAMCx,SAML22系統のArm®のみ)

次頁へ続く

前頁からの続き

パラメータ	大きさ	範囲/任意選択	使い方
node_rsel_prsc	1バイト	ビット7～4 – RSEL: RSEL_VAL_0 RSEL_VAL_3* RSEL_VAL_6* RSEL_VAL_20 RSEL_VAL_50 RSEL_VAL_70* RSEL_VAL_75* RSEL_VAL_80* RSEL_VAL_100 RSEL_VAL_120* RSEL_VAL_200*	内部Y線直列抵抗選択。 * 全てのデバイスで利用可能でないかもしれません。 SAM E5x、SAM D5x：3kΩ、6kΩ、75kΩ、200kΩ SAM L22：75kΩ、200kΩ AVR-DA：70kΩ、80kΩ、120kΩ、200kΩ
		ビット3～0 – PRSC: ORSC_DIV_1 ORSC_DIV_2 ORSC_DIV_4 ORSC_DIV_6* ORSC_DIV_8 ORSC_DIV_12* ORSC_DIV_14* ORSC_DIV_16* ORSC_DIV_32* ORSC_DIV_64* ORSC_DIV_128*	**クロック前置分周器** 採取クロックはAVR®デバイスに対してCPUクロックから引き出されて尺度調整されます。 * 全てのデバイスで利用可能でないかもしれません。 SAM E5x、SAM D5x、ATtiny：16、32、64、128 AVR-DA：6、12、14 ** ** 前置分周器値に対応する数値
node_gain	1バイト	ビット7～4 – アナログ利得: GAIN_1, GAIN_2, GAIN_4, GAIN_8, GAIN_16	**アナログ利得設定** 積分器利得を制御するために積分コンデンサが調整されます。
		ビット3～0 – デジタル利得: GAIN_1, GAIN_2, GAIN_4, GAIN_8, GAIN_16	**デジタル利得設定** 累積された総和がデジタル利得に調整されます。
node_oversampling	1バイト	FILTER_LEVEL_1 FILTER_LEVEL_2 FILTER_LEVEL_4 FILTER_LEVEL_8 FILTER_LEVEL_16 FILTER_LEVEL_32 FILTER_LEVEL_64 FILTER_LEVEL_128* FILTER_LEVEL_256* FILTER_LEVEL_512* FILTER_LEVEL_1024*	各測定に対して累積する採取(試料)数。 注: 過採取は正しい動作のためにデジタル利得以上に構成設定されなければなりません。 (64を超える濾波水準値はArm® SAM E54系統でだけ利用可能です。)

注: * – 全てのデバイスで利用可能な訳ではありません。

6.4.2. 状態と出力データ

各種目的対象ハードウェアは感知器節点に対する構成設定構造体がデバイス毎に変わることを必要とする一方で、全ての採取単位部は標準感知器節点データ構造体に従います。処理された単位部出力データは走行時の間、このデータ構造体に格納されます。

出力/状態の情報は他の後処理単位部や応用によって使われるかもしれません。

パラメータ	大きさ	範囲/任意選択	使い方
		ビット7	節点校正異常を示します。NODE_CAL_ERROR
		ビット6	上昇時間校正完了
		ビット5	–
node_acq_status	1バイト	ビット4～2 (3ビット) **節点校正状況:** NODE_MEASURE NODE_CC_CAL NODE_PRSC_CAL NODE_RSEL_CAL NODE_CSD_CAL	校正が進行中かどうかとそれの現在の段階を示します。
		校正要求	この節点で校正手順を起動するには'1'を書いてください。 (一旦活動した単位部によって'0'にリセットされます。)
		許可	測定用にこの節点を許可するには'1'を書いてください。
node_acq_signals	2バイト	この感知器節点の最新測定	累積されてnode_oversamplingとnode_gain_digitalの設定に対して調整された16ビット符号なし値
node_comp_caps	2バイト	ハードウェア校正データ	この節点用補償回路の調整を示します。

表6-1. node_acq_status

ビット	7	6	5	4	3	2	1	0
項目	節点校正異常	上昇時間校正完了	–	節点状況			校正要求	許可

値	0	1	2	3	4
節点状況	NODE_MEASURE	NODE_CC_CAL	NODE_PRSC_CAL	NODE_RSEL_CAL	NODE_CSD_CAL*

注: * – CSD校正はSAM D10/D11, SAM D2x, SAM L21のデバイスで利用できません。

採取ライブラリ状況

表6-2. touch_lib_state_t

値	0	1	2	3	4
状況	TOUCH_STATE_NULL	TOUCH_STATE_INIT	TOUCH_STATE_READY	TOUCH_STATE_CALIBRATE	TOUCH_STATE_BUSY

戻りパラメータ

表6-3. 全てのQTML単位部によって使われるtouch_ret_t共通戻り値型

値	0	1	2	3	11	14
結果	TOUCH_SUCCESS	TOUCH_ACQ_INCOMPLETE	TOUCH_INVALID_INPUT_PARAM	TOUCH_INVALID_LIB_STATE	TOUCH_INVALID_POINTER	TOUCH_LIB_NODE_CAL_ERROR

注: 他の値は将来の使用のために予約されています。

7. 増加動作

7.1. 序説

選ばれたデバイスのPTCは相互容量動作で同時に4チャネルを採取する機能を含みます。各チャネルの節点測定を解決するため、最低4つの同時測定が必要とされます。これはSNRでの全体的な改善や応答時間での減少で最終的な効果を持つアナログとデジタル両方の過採取の組み合わせの効果を持ちます。

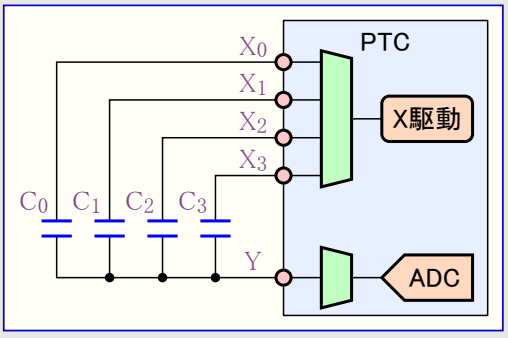
増加動作使用により、以下の恩恵の1つを得ることができます。

- 与えられた節点が4倍測定されるため、SNRが2倍(過採取倍の平方根)改善します。雑音が多い環境で、応答時間の妥協なしにSNRが2倍改善されます。
- 同じSNRを保つのにアナログ過採取を1/4に減らすことによって応答時間を1/4に減らすことができます。増加動作を許可することにより、SNRの妥協なしに濾波器の程度を32から8に減らすことができます。

支援されるデバイスは次のとおりです。

- SAM L1x
- ATtiny81x
- ATtiny161x
- ATtiny321x
- AVR-DA

図7-1. 並列X駆動



7.2. 構成設定

感知器節点は応用必要条件に応じて4X 1Yの群で増加動作用に整列されます。この単位部は相互容量4P測定だけを支援します。

7.2.1. 感知器節点群

qtm_acquisition_control_t

・採取群用最上位容器(ｺﾝﾃﾅ)

・構成と走行時ﾃﾞｰﾀを含む節点ﾒﾓﾘと群への位置指示子を含みます。

構造体	内容
qtm_acquisition_control_t	qtm_acq_node_group_config_t (*qtm_acq_node_group_config);
	qtm_acq_4p_saml10_config_t (*qtm_acq_node_config);
	qtm_acq_node_data_t (*qtm_acq_node_data);

qtm_acq_node_group_config_t

・群内の全ての感知器節点に適用される共通ﾊﾟﾗﾒｰﾀ

ﾊﾟﾗﾒｰﾀ	大きさ	範囲/任意選択	使い方
num_sensor_nodes	2ﾊﾞｲﾄ	4～65532	群で構成した感知器節点総数、例えば、4P数はX4を設定
acq_sensor_type	1ﾊﾞｲﾄ	NODE_MUTUAL_4P	4相互容量感知器節点の配列組。各組はXY容量測定用の4× PTC X遮蔽と1×PTC Y遮蔽を含みます。
calib_option_select	1ﾊﾞｲﾄ	ﾋﾞｯﾄ7～4: 校正目標	感知器容量用目標充電時間
		CAL_CHRG_2TAU	2×時定数間感知器充電
		CAL_CHRG_3TAU	3×時定数間感知器充電
		CAL_CHRG_4TAU	4×時定数間感知器充電
		CAL_CHRG_5TAU	5×時定数間感知器充電
		ﾋﾞｯﾄ3～0: 校正形式	どのﾊﾟﾗﾒｰﾀが電荷移動用に自動調節されるかを選択
		CAL_AUTO_TUNE_NONE	電荷移動に対して自動調節なし
		CAL_AUTO_TUNE_RSEL	完全電荷移動を許す最大値に調節される直列抵抗
		CAL_AUTO_TUNE_PRSC	完全電荷移動を許す最低(最速)値に調節される前置分周器
		CAL_AUTO_TUNE_CSD	完全電荷移動を許す最低値に調節される充電共用遅延
freq_option_select	1ﾊﾞｲﾄ	FREQ_SEL_0 ～FREQ_SEL_15	FREQ_SEL_0～FREQ_SEL_15は過採取中の測定間に遅延周期を挿入し、0は最短遅延、5が最長です。
		FREQ_SEL_SPREAD	FREQ_SEL_SPREADは過採取設定間で鋸歯様に0～15でこの遅延を変えます。
ptc_interrupt_priority	1ﾊﾞｲﾄ	1～3	Arm® NVIC割り込み優先権

qtm_acq_saml10_node_config_t

注: このﾃﾞｰﾀ構造体はSAML10 PTCﾊｰﾄﾞｳｪｱ用特定構成設定です。

ﾊﾟﾗﾒｰﾀ	大きさ	範囲/任意選択	使い方
node_xmask[4]	4×4ﾊﾞｲﾄ (16ﾊﾞｲﾄ) 4×8*ﾊﾞｲﾄ (32ﾊﾞｲﾄ) * AVR-DA	配列/ﾋﾞｯﾄ領域	NODE_MUTUAL_4P用Xﾋﾟﾝ遮蔽選択。X線番号に対応する位置のﾋﾞｯﾄを設定。 例: X0のみ=0b00000001=0x01、X0とX2=0b00000101=0x05
node_ymask	4/8*ﾊﾞｲﾄ * AVR-DA	(ﾋﾞｯﾄ領域)	Yﾋﾟﾝ遮蔽選択。Y線番号に対応する位置のﾋﾞｯﾄを設定。 例: Y5のみ=0b00100000=0x20、Y1,Y2,Y7=0b10000110=0x86
node_csd	1ﾊﾞｲﾄ	SAML1x用: 0～65534 ATtiny81x,161x,321x用: 0～65534	感知器節点容量の充電を保証するための遅延周期数。 注: 自動調節許可の場合、この値は初期補償ｺﾝﾃﾞﾝｻ校正に使われます。感知器を充電するのに充分な時間を許すことを保証してください。

次頁へ続く

前頁からの続き

パラメータ	大きさ	範囲/任意選択	使い方
node_rsel_prsc	1バイト	ビット7～4: RSEL RSEL_VAL_0 RSEL_VAL_20 RSEL_VAL_50 RSEL_VAL_75 RSEL_VAL_100 RSEL_VAL_200	内部Y線直列抵抗選択
		ビット3～0: PRSC PRSC_DIV_SEL_1 PRSC_DIV_SEL_2 PRSC_DIV_SEL_4 PRSC_DIV_SEL_8	クロック前置分周器。採取クロックはGCLK(SAM系列)またはCPUクロック(AVR系列)から分周されて尺度調整されます。
node_gain	1バイト	ビット7～4: アナログ利得 GAIN_1,GAIN_2,GAIN_4, GAIN_8,GAIN_16	アナログ利得設定。積分利得制御するために積分コンデンサ調節
		ビット3～0: デジタル利得 GAIN_1,GAIN_2,GAIN_4, GAIN_8,GAIN_16	デジタル利得設定。累積された合計がデジタル利得に尺度調整
node_oversampling	1バイト	FILTER_LEVEL_1 FILTER_LEVEL_2 FILTER_LEVEL_4 FILTER_LEVEL_8 FILTER_LEVEL_16 FILTER_LEVEL_32 FILTER_LEVEL_64	各測定で累積する採取数。 注: 過採取は正しい動作のためにデジタル利得と等しいかまたはより大きく構成設定されなければなりません。

qtm_acq_node_data_t

- 個別節点走行時データ(配列)

パラメータ	大きさ	範囲/任意選択	使い方
		ビット7	1=NODE_CAL_ERROR。感知器節点容量が支援される最大補償容量を超えます。
		ビット6	1=電荷移動調節完了
		ビット5	1=校正状況設定
node_acq_status	1バイト	ビット4～2: 節点校正状況 NODE_MEASURE NODE_CC_CAL NODE_PRSC_CAL NODE_RSEL_CAL NODE_CSD_CAL	どの校正が進行中かとその現在状況を示します。
		ビット1: 校正要求	この節点での校正手順を起動するには1を書いてください。(1度活性にされた単位部によって0にリセットされます。)
		ビット0: 校正要求	測定用にこの節点を許可するには1を書いてください。
node_acq_signals	2バイト	この感知器節点に対する最新の測定	node_oversamplingとnode_gain_digital設定のように累積されて尺度調整された16ビット符号なし値
node_comp_caps	2バイト	ハードウェア校正データ	この節点用補償回路の調節を示します。

7.2.2. 4P並列採取単位部

SAML1x単位部:

- libqtm_acq_4p_saml1x_0x0033.a
- qtm_acq_4p_saml1x_0x0033.a

ATtiny161x単位部:

- libqtm_acq_4p_t161x_0x001b.a
- qtm_acq_4p_t161x_0x001b.r90

ATtiny321x単位部:

- libqtm_acq_4p_t321x_0x001b.a
- qtm_acq_4p_t321x_0x001b.r90

AVR-DA単位部:

- libqtm_acq_4p_avr_da_0x0038.a
- qtm_acq_4p_avr_da_0x0038.r90

8. 周波数飛び跳ね単位部

8.1. 概要

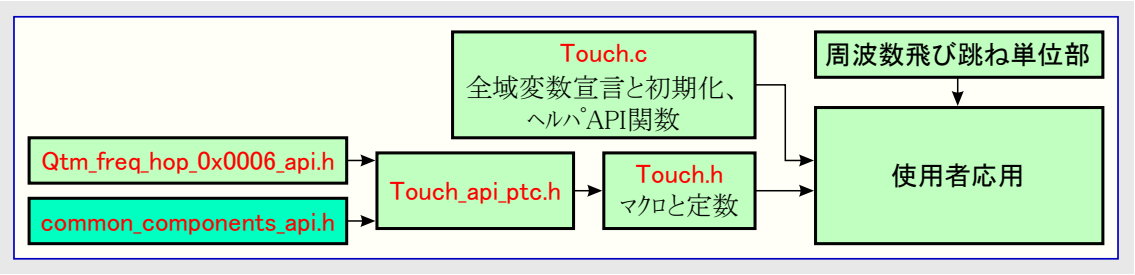
周波数飛び跳ね単位部は感知器収集周波数を変えることによって感知器測定中の雑音を濾波する方法を提供します。周波数飛び跳ね単位部用の単位部IDは'0x0006'で単位部名は下で与えられる形式です。

GCCコンパイラ	IARコンパイラ (AVR® MCU)	IARコンパイラ (Arm® MCU)
libqtm_freq_hop_xxxxx_0x0006.a	qtm_freq_hop_xxxxx_0x0006.r90	qtm_freq_hop_xxxxx_0x0006.a

注: "xxxxx" – 単位部が構築されるデバイス基本構造に基づく文字列

8.2. インターフェース

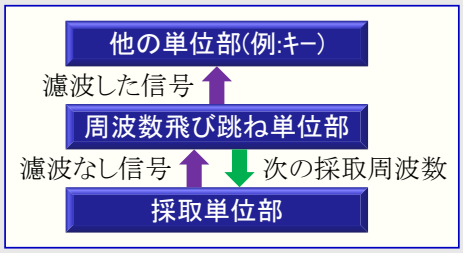
データ構造体定義とAPI宣言は'qtm_freq_hop_0x0006_api.h' APIファイルに含まれます。データ構造体は全ての構成設定と出力データ変数を網羅します。このファイルは'touch_ptc_api.h' 共通APIファイルでインクルードされるべきです。



構成設定の既定値はtouch.cとtouch.hのファイルで定義されるべきです。データ構造体の全域変数はtouch.cファイルで初期化されなければならず、構造体の参照は応用ファイルで使われなければなりません。

8.3. 機能的な説明

周波数飛び跳ね単位部は右で示されるように採取単位部と残りの後処理単位部間を繋ぎます。



周波数飛び跳ね単位部は各測定周回で違う採取周波数が使われるように構成設定可能な巡回周波数飛び跳ね算法を適用します。単位部は連続する測定周回間の巡回順によって設定される予め定義された周波数で初期化されます。

採取単位部からの測定された生信号値はその後に"中心値濾波器"を通して渡されます。最後に、後処理単位部によって更に処理されるためにメモリに濾波された値が保存し戻されます。

多数の周波数がより多くの採取を処理することによって効率的な濾波を提供します。けれども、これは中心値濾波によって使われる緩衝部の大きさも増し、濾波した値を報告するのにより多数の測定周回もかかります。故に、周波数の数は利用可能なRAMに基づいて構成設定されるべきです。

8.4. 構成設定

8.4.1. データ構造体

パラメータ	大きさ	範囲/任意選択	使い方
num_sensors	1バイト	0～255	中心値濾波のためにデータを緩衝する感知器数
num_freqs	1バイト	3～7	中心値濾波の周回/深さの周波数の数
*freq_option_select	2/4バイト	ポインタ	採取ライブラリ周波数選択パラメータへの位置指示子
*median_filter_freq	2/4バイト	ポインタ	選んだ周波数の配列への位置指示子

8.4.2. 状態と出力ﾃﾞｰﾀ

ﾊﾟﾗﾒｰﾀ	大きさ	範囲/任意選択	使い方
module_status	1ﾊﾞｲﾄ	該当なし	単位部状態 – 利用不可
current_freq	1ﾊﾞｲﾄ	0～15	現在の周波数段階
*freq_opti*filter_buffer	2/4ﾊﾞｲﾄ	ﾎﾟｲﾝﾀ	測定信号用濾波緩衝部配列への位置指示子
*qtm_acq_node_data	2/4ﾊﾞｲﾄ	ﾎﾟｲﾝﾀ	採取群の節点ﾃﾞｰﾀ構造体への位置指示子

表8-1. 支援周波数一覧 (PTCｸﾛｯｸ=4MHz)

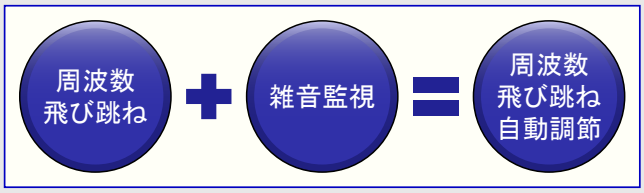
PTC周波数遅延周期		周波数 (kHz)	PTC周波数遅延周期		周波数 (kHz)	PTC周波数遅延周期		周波数 (kHz)
0	FREQ_SEL_0	66.67	6	FREQ_SEL_6	47.62	12	FREQ_SEL12	37.04
1	FREQ_SEL_1	62.5	7	FREQ_SEL_7	45.45	13	FREQ_SEL_13	35.71
2	FREQ_SEL_2	58.82	8	FREQ_SEL_8	43.48	14	FREQ_SEL_14	34.48
3	FREQ_SEL_3	55.56	9	FREQ_SEL_9	41.67	15	FREQ_SEL_15	33.33
4	FREQ_SEL_4	52.63	10	FREQ_SEL_10	40	16	FREQ_SEL_SPREAD	可変周波数
5	FREQ_SEL_5	50	11	FREQ_SEL_11	38.46			

9. 周波数飛び跳ね自動調節単位部

9.1. 概要

周波数飛び跳ね自動調節単位部は雑音監視と測定した雑音成分に従った周波数調節を付加的に提供する周波数飛び跳ね単位部の上位集合単位部です。

周波数飛び跳ね自動調節単位部の単位部IDは'0x0004'で単位部名は下で与えられる形式です。

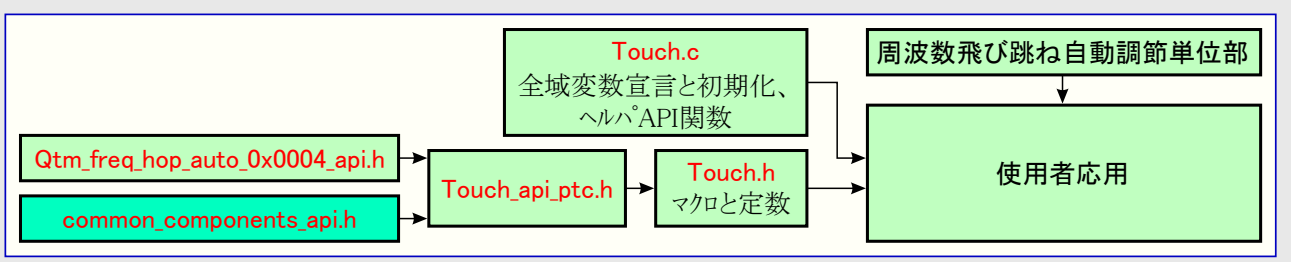


GCCｺﾝﾊﾟｲﾗ	IARｺﾝﾊﾟｲﾗ (AVR® MCU)	IARｺﾝﾊﾟｲﾗ (Arm® MCU)
libqtm_freq_hop_auto_xxxxx_0x0004.a	qtm_freq_hop_auto_xxxxx_0x0004.r90	qtm_freq_hop_auto_xxxxx_0x0004.a

注: "xxxxx" – 単位部が構築されるﾃﾞﾊﾞｲｽ基本構造に基づく文字列

9.2. ｲﾝﾀｰﾌｪｰｽ

ﾃﾞｰﾀ構造体定義とAPI宣言は'qtm_freq_hop_auto_0x0004_api.h' APIﾌｧｲﾙに含まれます。ﾃﾞｰﾀ構造体は全ての構成設定と出力ﾃﾞｰﾀ変数を網羅します。このﾌｧｲﾙは'touch_ptc_api.h' 共通APIﾌｧｲﾙでｲﾝｸﾙｰﾄﾞされるべきです。



構成設定の既定値はtouch.cとtouch.hのﾌｧｲﾙで定義されるべきです。ﾃﾞｰﾀ構造体の全域変数はtouch.cﾌｧｲﾙで初期化されなければならず、構造体の参照は応用ﾌｧｲﾙで使われなければなりません。

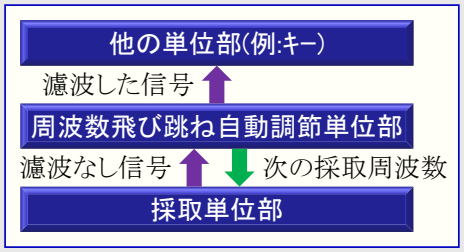
9.3. 機能的な説明

周波数飛び跳ね自動調節単位部は右下で示されるように採取単位部と残りの後処理単位部間を繋ぎます。

周波数飛び跳ね自動調節単位部は各測定周回で違う採取周波数が使われるように構成設定可能な巡回周波数飛び跳ね算法を適用します。連続する測定周回間でいくつかの予め構成設定された周波数が順に実行されます。

'n'個の周波数が周回に含まれる場合、出力ﾃﾞｰﾀに'n'点の中心値濾波が適用されます。

自動調節を実行するには各感知器節点で測定された信号が選ばれた各周波数に対して記録されます。1つの周波数が他よりも大きな変動を示す時にその周波数は測定手順から取り去られて別のもので置き換えられます。



9.4. 構成設定

9.4.1. ﾃﾞｰﾀ構造体

ﾊﾟﾗﾒｰﾀ	大きさ	範囲/任意選択	使い方
num_sensors	1ﾊﾞｲﾄ	0～255	中心値濾波のためにﾃﾞｰﾀを緩衝する感知器数
num_freqs	1ﾊﾞｲﾄ	3～7	中心値濾波の周回/深さの周波数の数
*freq_option_select	2/4ﾊﾞｲﾄ	ﾎﾟｲﾝﾀ	採取ﾗｲﾌﾞﾗﾘ周波数選択ﾊﾟﾗﾒｰﾀへの位置指示子
*median_filter_freq	2/4ﾊﾞｲﾄ	ﾎﾟｲﾝﾀ	選んだ周波数の配列への位置指示子
enable_freq_autotune	1ﾊﾞｲﾄ	0または1	飛び跳ね周波数自動再調節を禁止(0)または許可(1)
max_variance_limit	1ﾊﾞｲﾄ	1～255	飛び跳ね周波数自動再調節起動に必要とされる信号変動
Autotune_count_in	1ﾊﾞｲﾄ	1～255	飛び跳ね周波数自動再調節を起動するmax_variance_limit出現数

9.4.2. 状態と出力ﾃﾞｰﾀ

ﾊﾟﾗﾒｰﾀ	大きさ	範囲/任意選択	使い方
module_status	1ﾊﾞｲﾄ	該当なし	単位部状態 - 利用不可
current_freq	1ﾊﾞｲﾄ	0～15	現在の周波数段階
*filter_buffer	2/4ﾊﾞｲﾄ	ﾎﾟｲﾝﾀ	測定信号用濾波緩衝部配列への位置指示子
*qtm_acq_node_data	2/4ﾊﾞｲﾄ	ﾎﾟｲﾝﾀ	採取群の節点ﾃﾞｰﾀ構造体への位置指示子
*freq_tune_count_ins	2/4ﾊﾞｲﾄ	ﾎﾟｲﾝﾀ	周波数変更を起動する計数器配列への位置指示子

10. 接触キー単位部

10.1. 概要

接触キー単位部は一次元接触感知器としても呼ばれるキー感知器を扱うことができる機能を実装します。この単位部は採取単位部からの生出力を受け取り、それらを処理してキー感知器の接触状態を提供します。この処理は信号後処理、環境変動、接触検出、応用接触感知器の実装に対する接触状態機能とﾀｲﾐﾝｸﾞ管理を含みます。それら独自感知器基板を評価して設計する使用者を支援するために参照基準接触感知器設計が提供されます。接触感知器基板の見た目とQT3 Xplained Pro感知器基板の感知器設計が下で示されます。

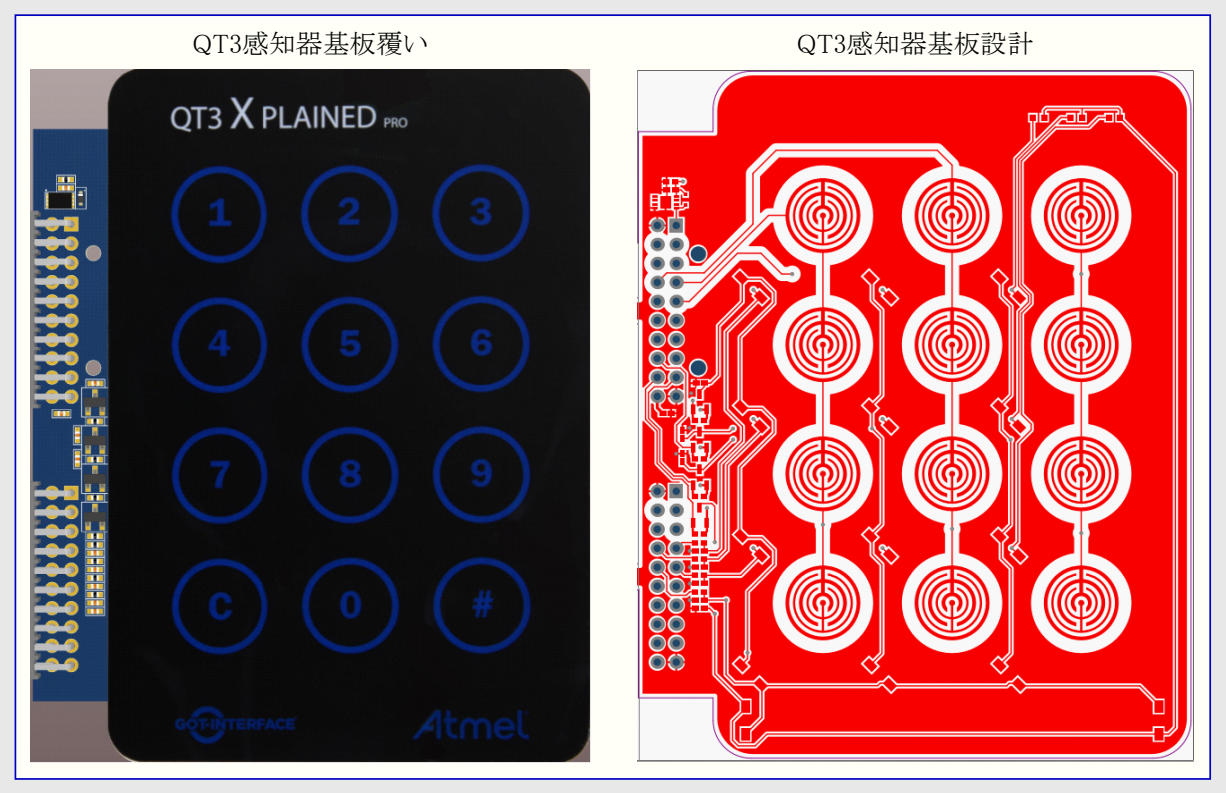


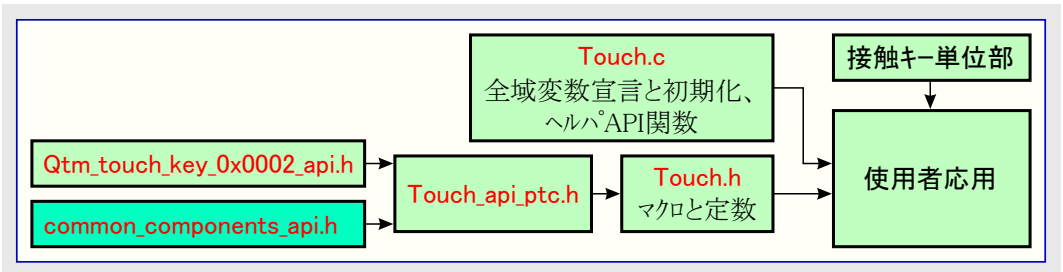
表10-1. 単位部形式

GCCｺﾝﾊﾟｲﾗ	IARｺﾝﾊﾟｲﾗ (AVR® MCU)	IARｺﾝﾊﾟｲﾗ (Arm® MCU)
libqtm_touch_key_xxxxx_0x0002.a	qtm_touch_key_xxxxx_0x0002.r90	qtm_touch_key_xxxxx_0x0002.a

注: “xxxxx” – 単位部が構築されるﾃﾞﾊﾞｲｽ基本構造に基づく文字列

10.2. インターフェース

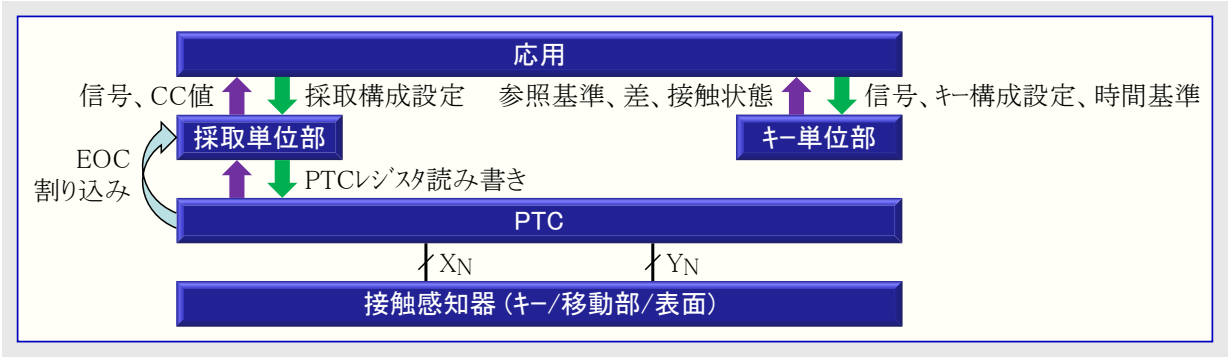
ﾃﾞｰﾀ構造体定義とAPI宣言は’qtm_touch_key_0x0002_api.h’APIﾌｧｲﾙに含まれます。ﾃﾞｰﾀ構造体は全ての構成設定と出力ﾃﾞｰﾀ変数を網羅します。このﾌｧｲﾙは’touch_ptc_api.h’共通APIﾌｧｲﾙでｲﾝｸﾙｰﾄﾞされるべきです。



10.3. 機能的な説明

接触キー単位部には上位単位部が位置補完、手ぶり認識、接触追跡などを実行する、接触接点の検出に対する責任があります。
接触キー単位部で実装される機能は次のとおりです。

- 接触に向かう、接触から離れるのを検出するためのタイミング管理
- ソフトウェア校正
  - 参照基準信号
  - 参照基準変動
- 接触検出状態機構



10.4. 構成設定

10.4.1. データ構造体

表10-2. 群構成設定

パラメータ	大きさ	範囲/任意選択	使い方
num_key_sensors	2バイト	1～65535	群内の感知器キー数
sensor_touch_di	1バイト	0～255	接触検出と非接触検出を確認するための測定繰り返し回数
sensor_max_on_time	1バイト	0(禁止),1～255	自動'再校正'に先立つ検出の感知器での計時器周期数
sensor_anti_touch_di	1バイト	0(禁止),1～255	必要とされる非接触再校正を確認するための測定繰り返し回数
sensor_anti_touch_recal_thr	1バイト	0～5	非接触閾値を設定するための接触閾値の縮小率 0=100% (接触閾値)、1=50%、2=25%、3=12.5%、4=6.25%、 5=最大再校正
sensor_touch_drift_rate	1バイト	0(禁止),1～255	接触に向かう変動間で下降計数する計時器周期数
sensor_anti_touch_drift_rate	1バイト	0(禁止),1～255	接触から離れる変動間で下降計数する計時器周期数
sensor_drift_hold_time	1バイト	0(禁止),1～255	接触事象後に変動を停止する計時器周期数
sensor_reburst_mode	1バイト	0=なし、 1=未解決 (即時再収集)、 2=全部	なし – 再収集は決して設定されず、応用の計画に従って測定。 未解決 – 再収集が設定され、目的対象感知器と同じAKS内のそれらを除き、全ての感知器が一時停止されます。 全部 – 再収集が設定され、感知器は一時停止されません。

表10-3. 個別感知器構成設定

パラメータ	大きさ	範囲/任意選択	使い方
channel_threshold	1バイト	0～255	接触接点を示す最大信号差
channel_hysteresis	1バイト	0(50%)～4(3.125%)	取り去られる接触接点を濾波して外す時の跳ね返り抑制のための接触閾値低減
channel_aks_group	1バイト	0～255	同時に接触検出を制御するキー感知器の群

10.4.2. 状態と出力ﾃﾞｰﾀ

表10-4. 群ﾃﾞｰﾀ

ﾊﾟﾗﾒｰﾀ	大きさ	範囲/任意選択	使い方
qtm_keys_status	1ﾊﾞｲﾄ	ﾋﾞｯﾄ7: 再収集要求 ﾋﾞｯﾄ6～1: (予約) ﾋﾞｯﾄ0: 接触検出	接触ｷｰ群の現在の状態を示します。
acq_group_timestamp	2ﾊﾞｲﾄ	0～65535	最後に処理した変動期間の時刻印
dht_count_in	1ﾊﾞｲﾄ	0～'sensor_drift_hold_time'	接触事象後に変動保持を解除するための下降計数
tch_drift_count_in	1ﾊﾞｲﾄ	0～'sensor_touch_drift_rate'	次に接触変動期間へ向かうための下降計数
antitch_drift_count_in	1ﾊﾞｲﾄ	0～'sensor_anti_touch_drift_rate'	次に接触変動期間から離れるための下降計数

個別ｷｰ感知器ﾃﾞｰﾀ

個別ｷｰ感知器ﾃﾞｰﾀは移動部のような他の後処理単位部によって必要とされます。故に、このﾃﾞｰﾀ構造体定義はcommon_compoenents_api.hﾌｧｲﾙに置かれます。

ﾊﾟﾗﾒｰﾀ	大きさ	範囲/任意選択	使い方
sensor_state	1ﾊﾞｲﾄ	ﾋﾞｯﾄ領域	接触ｷｰ感知器状況
sensor_state_counter	1ﾊﾞｲﾄ	0～255	接触検出と非接触検出を確認するための測定繰り返し回数
*node_data_struct_ptr	2/4ﾊﾞｲﾄ	ﾎﾟｲﾝﾀ	節点ﾃﾞｰﾀ構造体配列への位置指示子
Channel_reference	2ﾊﾞｲﾄ	0～65535	参照基準測定、接触検出の基準線

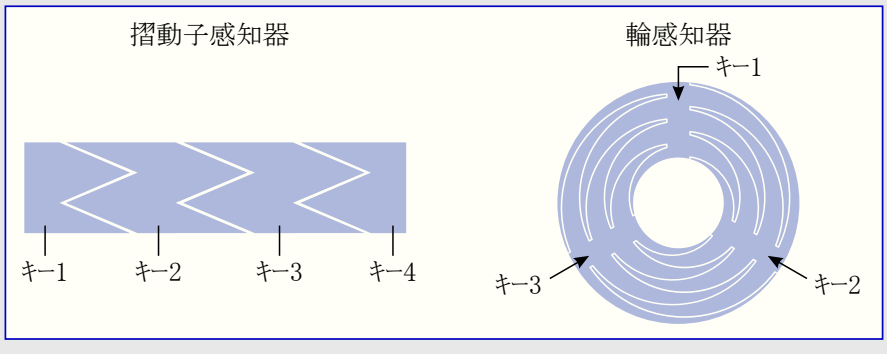
値	sensor_state	値	sensor_state	値	sensor_state
$00	QTM_KEY_STATE_DISABLE	$04	QTM_KEY_STATE_FILT_IN	$08	QTM_KEY_STATE_SUSPEND
$01	QTM_KEY_STATE_INIT	$85	QTM_KEY_STATE_DETECT	$09	QTM_KEY_STATE_CAL_ERR
$02	QTM_KEY_STATE_CAL	$86	QTM_KEY_STATE_FILT_OUT		
$03	QTM_KEY_STATE_NO_DET	$07	QTM_KEY_STATE_ANTI_TCH		

注: ﾋﾞｯﾄ7($80)は接触ｷｰ感知器が'検出中'の各状況で設定されます。

11. 移動器単位部

11.1. 概要

移動器(Scroller)単位部は下図で示されるように直線上の摺動子または円状の輪のどちらかで構築された接触感知器の群を処理します。摺動子/輪の感知器は一次元表面感知器としても知られ、それらの上を移動される接触の動きの追跡して状態と位置を使用者応用に報告します。摺動子/輪の大きさは直線状/円状の表面を形成する基礎となる接触ｷｰ感知器数です。



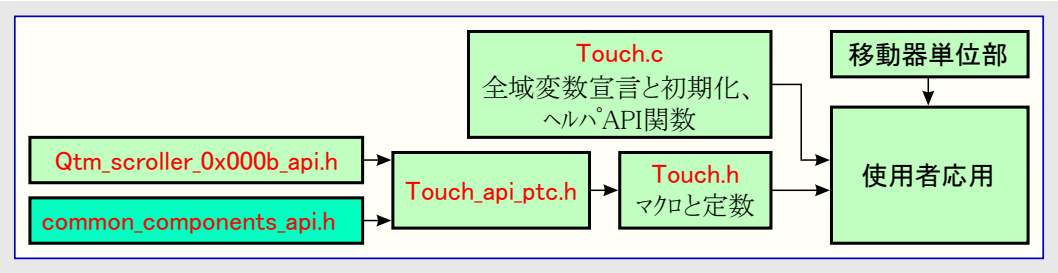
摺動子/輪は自己容量と相互容量の両感知器を使って形成することができます。上図は自己容量技法に基づく4ﾁｬﾈﾙ摺動子と3ﾁｬﾈﾙ輪の感知器を示します。接触が感知器表面上を移動される時に報告される接触位置での良好な直線性を得るため、接触ｷｰは上図で示されるように交互配置されるべきです。

GCCｺﾝﾊﾟｲﾗ	IARｺﾝﾊﾟｲﾗ (AVR® MCU)	IARｺﾝﾊﾟｲﾗ (Arm® MCU)
libqtm_scroller_xxxxx_0x000B.a	qtm_scroller_xxxxx_0x000B.r90	qtm_scroller_xxxxx_0x000B.a

注: "xxxxx" – 単位部が構築されるﾃﾞﾊﾞｲｽ基本構造に基づく文字列

11.2. インターフェース

データ構造体定義とAPI宣言は'qtm_scroller_0x000b_api.h'APIファイルに含まれます。データ構造体は全ての構成設定と出力データ変数を網羅します。このファイルは'touch_ptc_api.h'共通APIファイルでインクルードされるべきです。



11.3. 機能的な説明

移動器単位部処理は接触キー単位部出力に依存します。データ構造体でキーが処理されて状態が更新された後、それらは移動器単位部によって調べられます。キー状態に基づき、採取単位部変数で利用可能な現在の信号値から摺動子/輪の位置が計算されます。

有り得る使用事例と各使用事例下での動作の手順が下で与えられます。

使用事例1: 摺動子/輪の感知器で行われる接触接点

1. 単位部は接触接点検出のために移動部内の全キーの状態を調べます。
2. どれかのキーが検出状態なら、3つの隣接キーを使って接触位置が計算されます。
3. 生の位置と濾波された位置の両方が計算されます。
4. 移動部状況が"TOUCH_ACTIVE(有効接触)"になり、移動部再収集(reburst)フラグが設定されます。
5. 今や"POSITION_CHANGE(位置変更)"フラグが設定されます。このフラグは接触が止まっていて接触位置の変更がない場合、次の測定周回で解消されます。

使用事例2: 摺動子/輪の表面上で接触接点移動

1. 単位部は接触接点に対して全キーを調べます。
2. 検出中のキーがなければ、単位部は接触差が最小接点閾値を超える隣接キーの対を検索します。
3. そのような接点が見つかった場合、新しい位置が計算され、

または

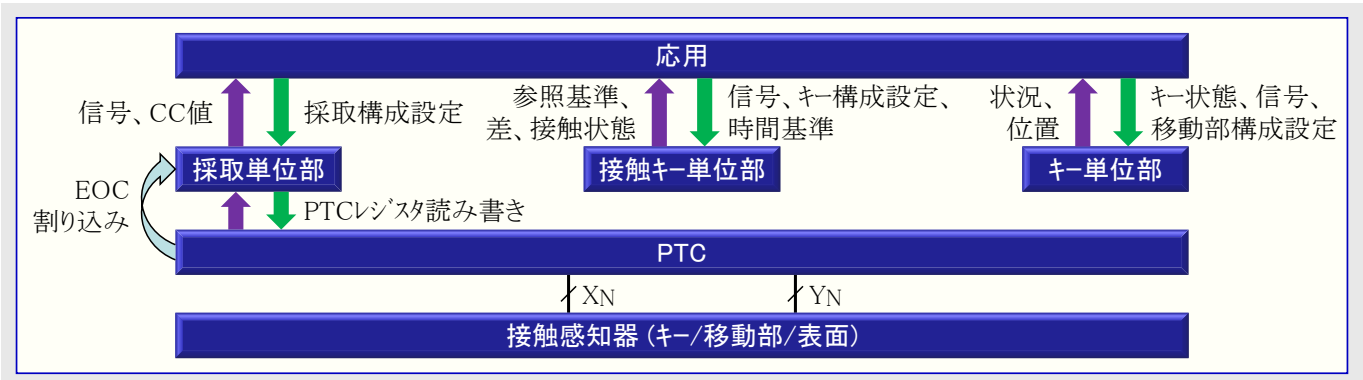
4. そのような接点が見つからない場合、移動部は'No Detect(検出なし)'を返します。

使用事例3: 摺動子/輪の感知器から接触接点取り去り

1. 単位部は接触接点検出のために移動部内の全キーの状態を調べます。
2. 検出中のキーがなければ、単位部は接触差が最小接点閾値を超える隣接キーの対を検索します。
3. そのような接点が見つかった場合、新しい位置が計算され、

または

4. そのような接点が見つからない場合、移動部は'No Detect(検出なし)'を返します。即ち"TOUCH_ACTIVE(有効接触)"フラグが解消されます。



11.4. 構成設定

11.4.1. ﾃﾞｰﾀ構造体

表11-1. 群構成設定

ﾊﾟﾗﾒｰﾀ	大きさ	範囲/任意選択	使い方
*qtm_touch_key_data	2/4ﾊﾞｲﾄ	qtm_touch_key_data_t	接触ｷｰの基礎となる組に対する接触ｷｰ ﾃﾞｰﾀへの位置指示子
num_scrollers	1ﾊﾞｲﾄ	1～255	この群内で実装される移動部数

表11-2. 個別感知器構成設定

ﾊﾟﾗﾒｰﾀ	大きさ	範囲/任意選択	使い方
type	1ﾊﾞｲﾄ	0=直線摺動子、1=輪	移動部の型式
start_key	2ﾊﾞｲﾄ	0～65535	移動部の最初の構成部品ｷｰを形成するｷｰ番号
number_of_keys	1ﾊﾞｲﾄ	2～255	移動部を形成する構成部品ｷｰ数。摺動子を作るのに必要とされる最小ｷｰ数は2、輪を作るのに必要とされる最小ｷｰ数は3です。
resol_deadband	1ﾊﾞｲﾄ	ﾋﾞｯﾄ7～4: 分解能 (2～12ﾋﾞｯﾄ)	移動部に対して報告される全尺位置分解能
		ﾋﾞｯﾄ3～0: 不感帯 (両側0～15%)	全尺範囲の%として端補正不感帯の大きさ
position_hysteresis	1ﾊﾞｲﾄ	0～255	接触または方向変更後に報告されるべき最小移動距離
contact_min_threshold	2ﾊﾞｲﾄ	0～65535	持続接触追跡用測定最小接点の大きさ。接点の大きさは接触接点を形成する2つの隣接ｷｰの接触差の合計です。

11.4.2. 状態と出力ﾃﾞｰﾀ

表11-3. 群ﾃﾞｰﾀ

ﾊﾟﾗﾒｰﾀ	大きさ	範囲/任意選択	使い方
scroller_group_status	1ﾊﾞｲﾄ	ﾋﾞｯﾄ7: 再収集要求 ﾋﾞｯﾄ6～1: (予約) ﾋﾞｯﾄ0: 接触検出	再収集要求=1：群内の1つ以上の移動部が感知器の再収集を必要としていることを示します。 接触検出=1：群内の1つ以上の移動部が'接触検出'であることを示します。

個別ｷｰ感知器ﾃﾞｰﾀ

ﾊﾟﾗﾒｰﾀ	大きさ	範囲/任意選択	使い方
scroller_status	1ﾊﾞｲﾄ	ﾋﾞｯﾄ7: 再収集要求 ﾋﾞｯﾄ6～2: (予約) ﾋﾞｯﾄ1: 接点移動 ﾋﾞｯﾄ0: 接点検出	再収集要求=1：群内の1つ以上の移動部が感知器の再収集を必要としていることを示します。 接点移動=1：報告された接点位置変更を示します。 接点検出=1：群内の1つ以上の移動部が'接触検出'であることを示します。
right_hyst	1ﾊﾞｲﾄ	ﾋｽﾃﾘｼｽ限度	接点が'右'移動、即ち、接触位置増加方向時を示します。
left_hyst	1ﾊﾞｲﾄ	ﾋｽﾃﾘｼｽ限度	接点が'左'移動、即ち、接触位置減少方向時を示します。
raw_position	2ﾊﾞｲﾄ	0～4095	動き選別前に計算された接触接点の場所
position	2ﾊﾞｲﾄ	0～4095	動き選別後に計算された接触接点の場所
contact_size	2ﾊﾞｲﾄ	0～65535	接触接点から成る2つの隣接ｷｰ接触差の合計

12. 2D表面(1指接触)CS単位部

12.1. 概要

この単位部は接触画面/接触ﾊﾟｯﾄﾞ応用に対する単一接点支援を持つ2D接触表面の機能を提供します。

- 接触接点XとYの位置
- 32×32までの感知器
  - 行/列に対する最大補償容量によって制限
- 自己容量
- 最適化された交差摺動子測定
  - (M+N)採取で測定される(M+N)感知器配列
- 永続接点追跡
- 位置濾過
  - IIR
  - 中央値
  - ﾋｽﾃﾘｼｽ
  - 不感帯

表面CS単位部は表面領域上に配列される感知器ｷｰの格子で使うのを意図されます。

表面CS単位部はQTML接触ｷｰ単位部(0x0002)または互換接触検出単位部とやり取りするように構成設定されます。位置指示子(ﾎﾟｲﾝﾀ)は共通APIﾌｧｲﾙで定義されるような標準形式で接触ｷｰ ﾃﾞｰﾀの場所に対して必要とされます。

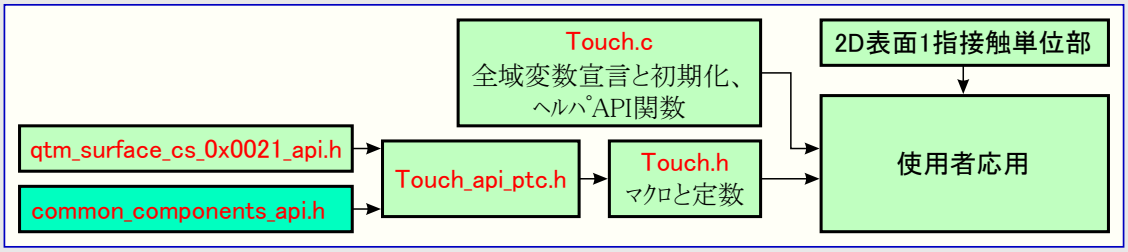
表12-1. 単位部ﾌｧｲﾙ

GCCｺﾝﾊﾟｲﾗ (例えば、Atmel Studio 7)	IARｺﾝﾊﾟｲﾗ (AVR®ﾃﾞﾊﾞｲｽ)	IARｺﾝﾊﾟｲﾗ (Arm®ﾃﾞﾊﾞｲｽ)
libqtm_surface_cs_xxxxx_0x0021.a libqtm_surface_cs_32x32_xxxxx_0x003b.a	qtm_surface_cs_xxxxx_0x0021.r90 qtm_surface_cs_32x32_xxxxx_0x003b.r90	qtm_surface_cs_xxxxx_0x0021.a qtm_surface_cs_32x32_xxxxx_0x003b.a

注: “xxxxx”は目的対象のﾌﾟﾛｾｯｻまたは基本構造を示します。
0x0021の単位部：最大16×16までを支援、0x003bの単位部：最大32×32までを支援

12.2. ｲﾝﾀｰﾌｪｰｽ

全ての使用者任意選択は応用ｺｰﾄﾞ(touch.h/touch.c)で構成設定され、ﾎﾟｲﾝﾀ参照によってﾗｲﾌﾞﾗﾘ単位部と共用されます。



qtm_surface_cs_0x0021_api.h qtm_surface_cs_0x003b_api.h	このﾍｯﾀﾞ ﾌｧｲﾙは単位部に関係する全てのAPI実装を含みます。それは応用ﾌﾟﾛｼﾞｪｸﾄでｺﾝﾊﾟｲﾙされた単位部と共に含められなければなりません。
qtm_common_components_api.h	このﾍｯﾀﾞ ﾌｧｲﾙは節点とｷｰ ﾃﾞｰﾀの構造体とtouch_ret_t戻り符号のような、全てのQTML単位部によってｱｸｾｽすることができるAPI宣言を含みます。

12.3. 機能的な説明

初期化

ﾃﾞｰﾀ構造体はﾗｲﾌﾞﾗﾘ使用前に構成設定と共に設定されなければなりません。’surface_cs’単位部は’qtm_init_surface_cs()’API呼び出しによって初期化されます。

単位部支援

感知器節点(’acquisition’単位部)とｷｰ(’touch_key’単位部)は’surface_cs’単位部によって必要とされるため、適切に構成設定されなければなりません。

感知器節点構成設定

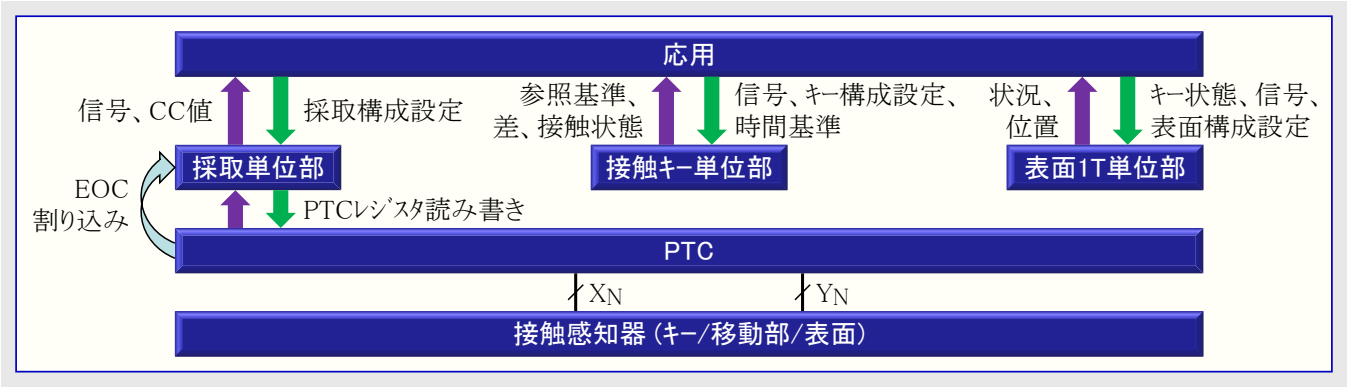
表面CSは1つの水平と1つの垂直の摺動子を実装するために構成設定されて群化された感知器を必要とします。Xﾋﾟﾝが水平に配列される場合、垂直摺動子が(全X)/各Yで構成されます。同様に水平摺動子は各X/(全Y)で構成されます。

走行時動作

表面CSは応用に接触接点情報を提供する最上位単位部として機能します。これは感知器校正、信号変動、接触検出と時間機能実装に対して接触キー ライブラリ単位部を利用します。接触キー単位部それ自身は容量性測定ハードウェアとやり取りするのに目的対象特有採取単位部を使います。

QTML応用では単位部処理順が以下のように正しく呼ばれなければなりません。

1. 全感知器節点の測定の採取 – qtm_ptc_start_measurement_seq();
2. 全感知器節点に対する採取処理 – qtm_acquisition_process();
3. 全キーに対する接触キー処理 – qtm_key_sensors_process();
4. 表面CS処理 – qtm_surface_cs_process();



12.4. 動作

採取と接触キー処理後に'qtm_surface_cs_process()'API関数が呼ばれます。

接点作成

表面感知器で接点が作られると、

1. 単位部は接触接点検出のために表面で全キーを調べます。
2. 検出中のキーが見つかったなら、その位置が計算されます。
3. 接点の大きさが計算され、最小接点閾値に対して比較されます。
4. 表面は'検出'状態になります。

接点の追跡/開放

1. 単位部は接触接点に関して全キーを調べます。
2. 検出中のキーがなかった場合、単位部は接触差が最小接点閾値を超える隣接キーの対を検索します。
3. そのような接点が見つかったなら、新しい位置が計算されるか、

または

4. そのような接点が見つからなかった場合、表面は'検出なし'状況を返します。

12.5. 構成設定

表面CS単位部は接触キーの基礎となるための動作パラメータとキー データの組に対する位置指示子(ポインタ)で構成設定されなければなりません。

12.5.1. データ構造体

qtm_surface_cs_control_t

- 表面構成設定用最上位容器(コンテナ)
- データと構成設定の構造体への位置指示子を含みます。

構造体	内容
qtm_surface_cs_control_t	qtm_surface_contact_data_t *qtm_surface_contact_data;
	qtm_surface_cs_config_t *qtm_surface_cs_config;

qtm_surface_contact_data_t

・接触表面用走行時ﾃﾞｰﾀ

ﾊﾟﾗﾒｰﾀ	大きさ	範囲/任意選択	使い方
qt_surface_status	1ﾊﾞｲﾄ	ﾋﾞｯﾄ7: 再収集要求	再収集要求=1：接点状態を解決/更新するのに更なる測定が必要とされることを示します。
		ﾋﾞｯﾄ6: -	-
		ﾋﾞｯﾄ5: POS_V_DEC	垂直位置減少
		ﾋﾞｯﾄ4: POS_V_INC	垂直位置増加
		ﾋﾞｯﾄ3: POS_H_DEC	水平位置減少
		ﾋﾞｯﾄ2: POS_H_INC	水平位置増加
		ﾋﾞｯﾄ1: POS_CHANGE	報告された位置の変更
		ﾋﾞｯﾄ0: 接触検出	接触検出=1：接触接点が表面に存在することを示します。
h_position_abs	2ﾊﾞｲﾄ	0～4095	見かけ上の水平位置
h_position	2ﾊﾞｲﾄ	0～4095	動き選別された水平位置
v_position_abs	2ﾊﾞｲﾄ	0～4095	見かけ上の垂直位置
v_position	2ﾊﾞｲﾄ	0～4095	動き選別された垂直位置
contact_size	2ﾊﾞｲﾄ	-	接点位置での接触差の合計

qtm_surface_cs_config_t

・接触表面用構成設定ﾊﾟﾗﾒｰﾀ

ﾊﾟﾗﾒｰﾀ	大きさ	範囲/任意選択	使い方
start_key_h	2ﾊﾞｲﾄ	0～65534	水平軸の開始ｷｰ
number_of_keys_h	1ﾊﾞｲﾄ	0～255	水平軸を形成するｷｰ数
start_key_v	2ﾊﾞｲﾄ	0～65534	垂直軸の開始ｷｰ
number_of_keys_v	1ﾊﾞｲﾄ	0～255	垂直軸を形成するｷｰ数
resol_deadband	1ﾊﾞｲﾄ	ﾋﾞｯﾄ7～4: 分解能 (2～12ﾋﾞｯﾄ)	軸に対して報告される全尺位置分解能
position_hysteresis	1ﾊﾞｲﾄ	0～255	接点または方向の変更後に報告される最小移動距離。水平と垂直に提供されます。
position_filter	1ﾊﾞｲﾄ	ﾋﾞｯﾄ7～5: -	-
		ﾋﾞｯﾄ4: 中央値濾波	中央値濾波器許可
		ﾋﾞｯﾄ3,2: -	-
		ﾋﾞｯﾄ1,0: IIR構成設定	IIR構成設定: 0=なし、1=25%、2=50%、3=75%
contact_min_threshold	2ﾊﾞｲﾄ	0～65535	永続接点追跡用測定の最小接点の大きさ。接点の大きさは接触接点を形成する隣接ｷｰの接触差の合計です。
*qtm_touch_key_data	2/4ﾊﾞｲﾄ	qtm_touch_key_data_t	接触ｷｰの基礎となる組に対する接触ｷｰ ﾃﾞｰﾀへの位置指示子

13. 2D表面(2指接触)CS/2T単位部

13.1. 概要

この単位部は接触画面/接触ﾊﾟｯﾄﾞ応用に対する2接点支援を持つ2D接触表面の機能を提供します。

- 接触接点XとYの位置
- 32×32までの感知器
  - 行/列に対する最大補償容量によって制限
- 相互容量と自己容量
- 最適化された交差摺動子測定
  - (M+N)採取で測定される(M+N)感知器配列
- 永続接点追跡
- 接点経路外挿
- 位置濾過
  - IIR
  - 中央値
  - ﾋｽﾃﾘｼｽ
  - 不感帯

表面CS/2T単位部は表面領域上に配列される感知器ｷｰの格子で使うのを意図されます。

表面CS/2T単位部はQTML接触ｷｰ単位部(0x0002)または互換接触検出単位部とやり取りするように構成設定されます。位置指示子(ﾎﾟｲﾝﾀ)は共通APIﾌｧｲﾙで定義されるような標準形式で接触ｷｰ ﾃﾞｰﾀの場所に対して必要とされます。

重要: この2D表面(2指接触)CS/2T単位部によって出力される2つの位置は手ぶり検出にだけ使うことができます。従って、手ぶり支援なしでこの単位部を使うことはお勧めできません。

2接触接点で信頼に足る性能を得るために各次元に最小8つの感知器が推奨されます。より少ない感知器の使用は独立した接点に対して分離された空間を厳しく制限します。

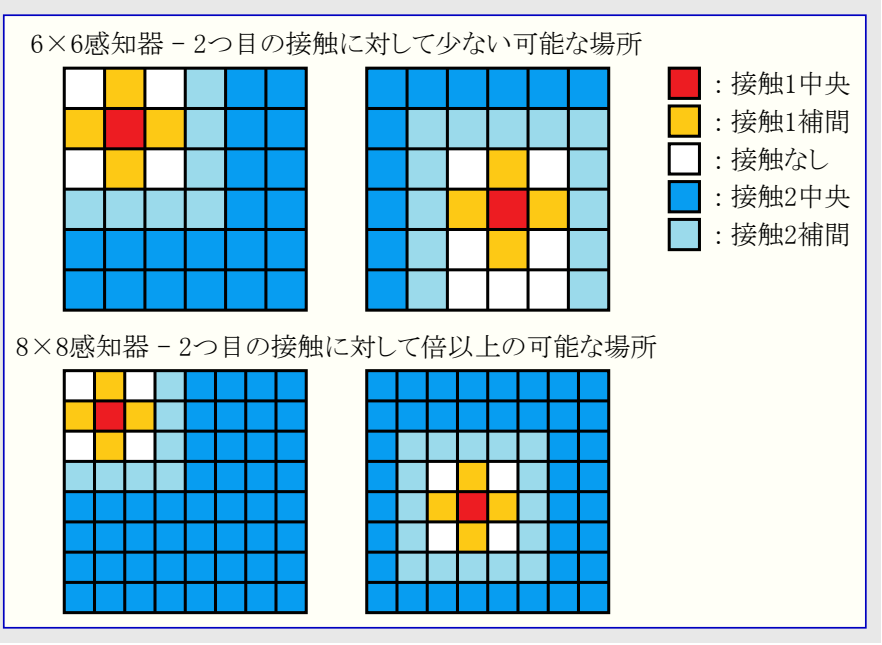


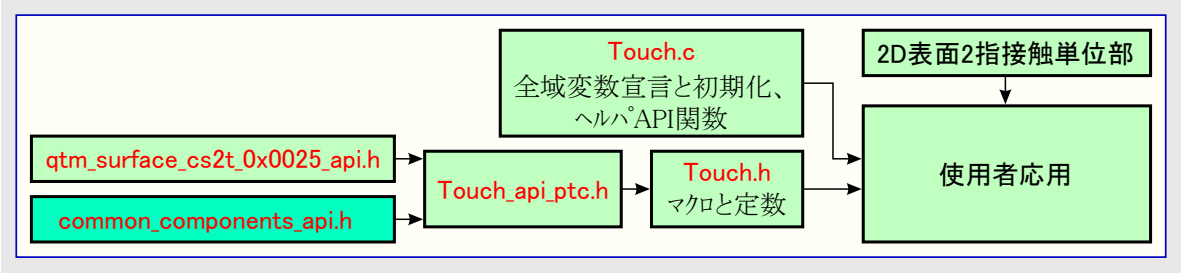
表13-1. 単位部ﾌｧｲﾙ

GCCｺﾝﾊﾟｲﾗ (例えば、Atmel Studio 7)	IARｺﾝﾊﾟｲﾗ (AVR®ﾃﾞﾊﾞｲｽ)	IARｺﾝﾊﾟｲﾗ (Arm®ﾃﾞﾊﾞｲｽ)
libqtm_surface_cs2t_xxxxx_0x0025.a libqtm_surface_cs2t_32x32_xxxxx_0x003c.a	qtm_surface_cs2t_xxxxx_0x0025.r90 qtm_surface_cs2t_32x32_xxxxx_0x003c.r90	qtm_surface_cs2t_xxxxx_0x0025.a qtm_surface_cs2t_32x32_xxxxx_0x003c.a

注: “xxxxx”は目的対象のﾌﾟﾛｾｯｻまたは基本構造を示します。
0x0025の単位部：最大16×16までを支援、0x003cの単位部：最大32×32までを支援

13.2. インターフェース

全ての使用者任意選択は応用ｺｰﾄﾞ(touch.h/touch.c)で構成設定され、ﾎﾟｲﾝﾀ参照によってﾗｲﾌﾞﾗﾘ単位部と共用されます。



qtm_surface_cs2t_0x0025_api.h qtm_surface_cs_0x003c_api.h	このﾍｯﾀﾞ ﾌｧｲﾙは単位部に関係する全てのAPI実装を含みます。それは応用ﾌﾟﾛｼﾞｪｸﾄでｺﾝﾊﾟｲﾙされた単位部と共に含められなければなりません。
qtm_common_components_api.h	このﾍｯﾀﾞ ﾌｧｲﾙは節点とｷｰ ﾃﾞｰﾀの構造体とtouch_ret_t戻り符号のような、全てのQTML単位部によってｱｸｾｽすることができるAPI宣言を含みます。

13.3. 機能的な説明

初期化

ﾃﾞｰﾀ構造体はﾗｲﾌﾞﾗﾘ使用前に構成設定と共に設定されなければなりません。'surface_cs2t'単位部は'qtm_init_surface_cs2t()'API呼び出しによって初期化されます。

単位部支援

感知器節点('acquisition'単位部)とｷｰ('touch_key'単位部)は'surface_cs2t'単位部によって必要とされるため、適切に構成設定されなければなりません。

感知器節点構成設定

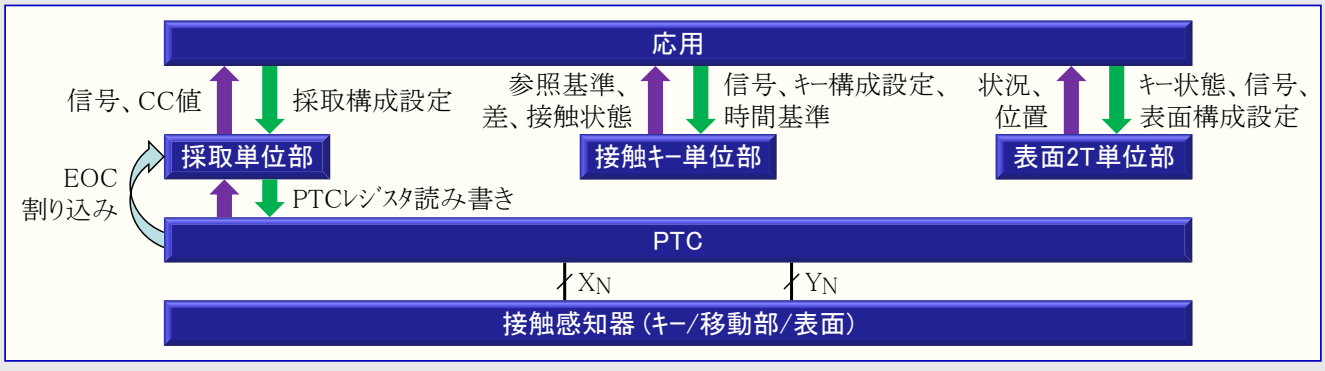
表面CS/2Tは1つの水平と1つの垂直の摺動子を実装するために構成設定されて群化された感知器を必要とします。Xﾋﾟﾝが水平に配列される場合、垂直摺動子が(全X)/各Yで構成されます。同様に水平摺動子は各X/(全Y)で構成されます。

走行時動作

表面CS/2Tは応用に接触接点情報を提供する最上位単位部として機能します。これは感知器校正、信号変動、接触検出と時間機能実装に対して接触ｷｰ ﾗｲﾌﾞﾗﾘ単位部を利用します。接触ｷｰ単位部それ自身は容量性測定ﾊｰﾄﾞｳｪｱとやり取りするのに目的対象特有採取単位部を使います。

QTML応用では単位部処理順が以下のように正しく呼ばれなければなりません。

1. 全感知器節点の測定の採取 – qtm_ptc_start_measurement_seq();
2. 全感知器節点に対する採取処理 – qtm_acquisition_process();
3. 全ｷｰに対する接触ｷｰ処理 – qtm_key_sensors_process();
4. 表面CS/2T処理 – qtm_surface_cs2t_process();



13.4. 動作

採取と接触キー処理後に'qtm_surface_cs2t_process()'API関数が呼ばれます。

接点作成

表面感知器で接点が作られると、

1. 単位部は接触接点検出のために表面で全キーを調べます。
2. 検出中のキーが見つかったなら、その位置が計算されます。
3. 接点の大きさが計算され、最小接点閾値に対して比較されます。
4. 表面は'検出'状態になります。

接点の追跡/開放

1. 単位部は接触接点に関して全キーを調べます。
2. 検出中のキーがなかった場合、単位部は接触差が最小接点閾値を超える隣接キーの対を検索します。
3. そのような接点が見つかったなら、新しい位置が計算されるか、

または

4. そのような接点が見つからなかった場合、表面は'検出なし'状況を返します。

13.5. 構成設定

表面CS/2T単位部は接触キーの基礎となるための動作ﾊﾟﾗﾒｰﾀとキー ﾃﾞｰﾀの組に対する位置指示子(ﾎﾟｲﾝﾀ)で構成設定されなければなりません。

13.5.1. ﾃﾞｰﾀ構造体

qtm_surface_cs2t_control_t

- 表面構成設定用最上位容器(ｺﾝﾃﾅ)
- ﾃﾞｰﾀと構成設定の構造体への位置指示子を含みます。

構造体	内容
qtm_surface_cs2t_control_t	qtm_surface_cs2t_data_t *qtm_surface_cs2t_data;
	qtm_surface_contact_data_t*qtm_surface_contact_data;
	qtm_surface_cs_config_t *qtm_surface_cs_config;

qtm_surface_cs2t_data_t

- 表面CS/2T単位部用走行時ﾃﾞｰﾀ

ﾊﾟﾗﾒｰﾀ	大きさ	範囲/任意選択	使い方
qt_surface_cs2t_status	1ﾊﾞｲﾄ	ﾋﾞｯﾄ7: 再収集要求	再収集要求=1：接点状態を解決/更新するのに更なる測定が必要とされることを示します。
		ﾋﾞｯﾄ6: -	-
		ﾋﾞｯﾄ5: POS_MERGED_V	2つの接点が存在し、分離するのに垂直位置が近すぎます。
		ﾋﾞｯﾄ4: POS_MERGED_H	2つの接点が存在し、分離するのに水平位置が近すぎます。
		ﾋﾞｯﾄ3: -	-
		ﾋﾞｯﾄ2: -	-
		ﾋﾞｯﾄ1: -	-
		ﾋﾞｯﾄ0: 接触検出	接触検出=1：接触接点が表面に存在することを示します。

qtm_surface_contact_data_t

・個別接触節点(配列)用走行時ﾃﾞｰﾀ

ﾊﾟﾗﾒｰﾀ	大きさ	範囲/任意選択	使い方
qt_contact_status	1ﾊﾞｲﾄ	ﾋﾞｯﾄ7: 再収集要求	再収集要求=1：接点状態を解決/更新するのに更なる測定が必要とされることを示します。
		ﾋﾞｯﾄ6: -	-
		ﾋﾞｯﾄ5: POS_V_DEC	垂直位置減少
		ﾋﾞｯﾄ4: POS_V_INC	垂直位置増加
		ﾋﾞｯﾄ3: POS_H_DEC	水平位置減少
		ﾋﾞｯﾄ2: POS_H_INC	水平位置増加
		ﾋﾞｯﾄ1: POS_CHANGE	報告された位置の変更
		ﾋﾞｯﾄ0: 接触検出	接触検出=1：接触接点が表面に存在することを示します。
h_position_abs	2ﾊﾞｲﾄ	0～4095	選別(濾波)されない水平位置
h_position	2ﾊﾞｲﾄ	0～4095	選別(濾波)された水平位置
v_position_abs	2ﾊﾞｲﾄ	0～4095	選別(濾波)されない垂直位置
v_position	2ﾊﾞｲﾄ	0～4095	選別(濾波)された垂直位置
contact_size	2ﾊﾞｲﾄ	-	接点位置での接触差の合計

qtm_surface_cs_config_t

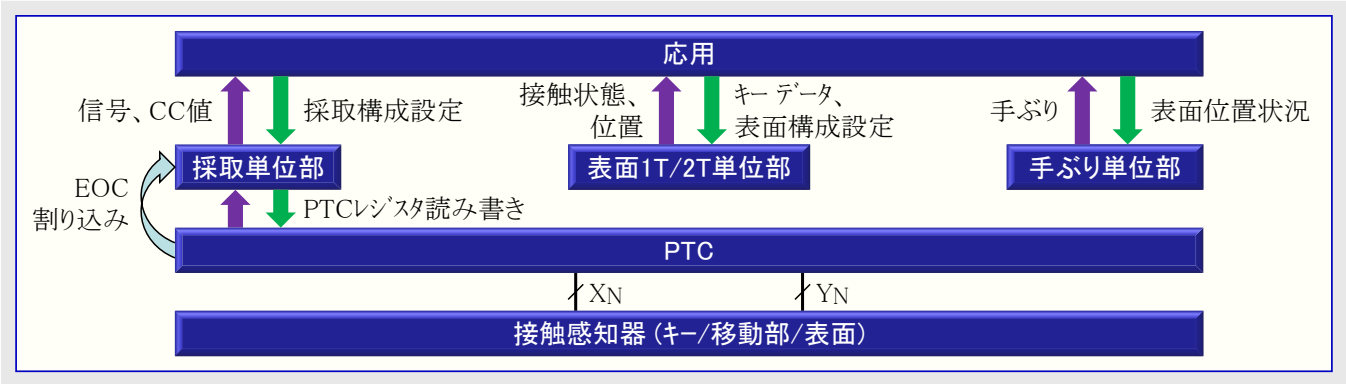
・接触表面用構成設定ﾊﾟﾗﾒｰﾀ

ﾊﾟﾗﾒｰﾀ	大きさ	範囲/任意選択	使い方
start_key_h	2ﾊﾞｲﾄ	0～65534	水平軸の開始ｷｰ
number_of_keys_h	1ﾊﾞｲﾄ	0～255	水平軸を形成するｷｰ数
start_key_v	2ﾊﾞｲﾄ	0～65534	垂直軸の開始ｷｰ
number_of_keys_v	1ﾊﾞｲﾄ	0～255	垂直軸を形成するｷｰ数
resol_deadband	1ﾊﾞｲﾄ	ﾋﾞｯﾄ7～4: 分解能 (2～12ﾋﾞｯﾄ)	軸に対して報告される全尺位置分解能
position_hysteresis	1ﾊﾞｲﾄ	0～255	接点または方向の変更後に報告される最小移動距離。水平と垂直に提供されます。
position_filter	1ﾊﾞｲﾄ	ﾋﾞｯﾄ7～5: -	-
		ﾋﾞｯﾄ4: 中央値濾波	中央値濾波器許可
		ﾋﾞｯﾄ3,2: -	-
		ﾋﾞｯﾄ1,0: IIR構成設定	IIR構成設定: 0=なし、1=25%、2=50%、3=75%
contact_min_threshold	2ﾊﾞｲﾄ	0～65535	永続接点追跡用測定の最小接点の大きさ。接点の大きさは接触接点を形成する隣接ｷｰの接触差の合計です。
*qtm_touch_key_data	2/4ﾊﾞｲﾄ	qtm_touch_key_data_t	接触ｷｰの基礎となる組に対する接触ｷｰ ﾃﾞｰﾀへの位置指示子

14. 手ぶり単位部

14.1. 概要

手ぶり単位部は表面単位部から受け取った接触位置に基づく手ぶりを識別して検出した手ぶりを報告します。この単位部は水平と垂直の接触位置を処理して何れかの手ぶりが識別された場合に報告します。



この単位部はQTouch表面単位部とやり取りするように構成設定されます。手ぶりを識別するのに表面単位部接点データへの位置指示子(ポインタ)が必要とされます。

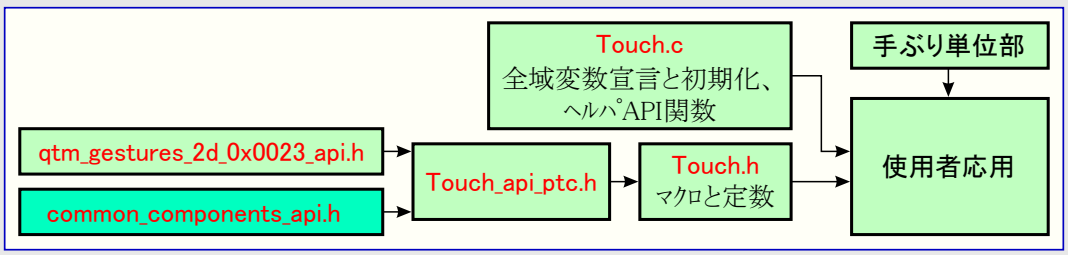
表13-1. 単位部ファイル

GCC コンパイラ	SAMデバイス	libqtm_surface_gestures_cm0p_0x0023.a	IAR コンパイラ	SAMデバイス	qtm_surface_gestures_cm0p_0x0023.r90
	AVR® デバイス	libqtm_surface_gestures_t1614_0x0023.a		AVR® デバイス	qtm_surface_gestures_t1614_0x0023.r90
		libqtm_surface_gestures_t1616_0x0023.a			qtm_surface_gestures_t1616_0x0023.r90
		libqtm_surface_gestures_t1617_0x0023.a			qtm_surface_gestures_t1617_0x0023.r90
		libqtm_surface_gestures_t3214_0x0023.a			qtm_surface_gestures_t3214_0x0023.r90
		libqtm_surface_gestures_t3216_0x0023.a			qtm_surface_gestures_t3216_0x0023.r90
		libqtm_surface_gestures_t3217_0x0023.a			qtm_surface_gestures_t3217_0x0023.r90

14.2. 単位部へのインターフェース

この単位部が表面単位部に依存するため、手ぶりの処理のために表面接触の位置と状態を読むために表面接点データへの位置指示子(ポインタ)を必要とします。

全ての使用者任意選択は応用コード(touch.h/touch.c)で構成設定され、ポインタ参照によってライブラリ単位部と共用されます。



qtm_gestures_2d_0x0023_api.h	このヘッダ ファイルは単位部に関係する全てのAPI実装を含みます。それは応用プロジェクトでコンパイルされた単位部と共に含められなければなりません。
qtm_common_components_api.h	このヘッダ ファイルは節点とキー データの構造体とtouch_ret_t戻り符号のような、全てのQTML単位部によってアクセスすることができるAPI宣言を含みます。

14.3. 構成設定

14.3.1. データ構造体

qtm_gestures_2d_control_t

qtm_gestures_2d_control_tデータ インターフェースはこの単位部の入力と出力を制御する容器(コンテナ)構造体

領域	単位	範囲/任意選択	パラメータ
qtm_gestures_2d_data	qtm_gestures_2d_data _t*	ポインタ	手ぶりデータ構造体への位置指示子(ポインタ)
qtm_gestures_2d_config	qtm_gestures_2d_config _t*	ポインタ	手ぶり構成設定構造体への位置指示子

qtm_gestures_2d_config _t

qtm_gestures_2d_config _tﾃﾞｰﾀ構造体は単位部へ渡される構成設定構造体です。

領域	単位	範囲/任意選択	ﾊﾟﾗﾒｰﾀ
horiz_position0	uint16_t	ﾎﾟｲﾝﾀ	水平接点0位置への位置指示子(ﾎﾟｲﾝﾀ)
vertical_position0	uint16_t	ﾎﾟｲﾝﾀ	垂直接点0位置への位置指示子
surface_status0	uint8_t	ﾎﾟｲﾝﾀ	接点0の状態への位置指示子
horiz_position1	uint16_t	ﾎﾟｲﾝﾀ	水平接点1位置への位置指示子
vertical_position1	uint16_t	ﾎﾟｲﾝﾀ	垂直接点1位置への位置指示子
surface_status1	uint8_t	ﾎﾟｲﾝﾀ	接点1の状態への位置指示子
surface_resolution	uint8_t	0～255	このﾊﾟﾗﾒｰﾀは表面の分機能を定義します。
tapReleaseTimeout	uint8_t	0～255	このﾊﾟﾗﾒｰﾀは初期押下と離昇間で許される時間量を制限します。この値超過は手ぶりをﾌｧｰﾑｳｪｱにﾀｯﾌﾟ手ぶりと見做させます。 注: この値はtapHoldTimeoutとswipeTimeoutより小さいべきです。
tapHoldTimeout	uint8_t	0～255	指がTAP_AREAによって設定された境界内に留まって取り去られない場合、一旦手ぶり計時器がこの値を超えると、ﾌｧｰﾑｳｪｱは'ﾀｯﾌﾟ保持'手ぶりを報告します。 注: この値はtapReleaseTimeoutとswipeTimeoutより大きいべきです。
swipeTimeout	uint8_t	0～255	この値はｽﾜｲﾌﾟ手ぶり(初期指押下、距離閾値を横切って特定方向へ移動、そして離昇)に許される時間量を制限します。 注: これはtapReleaseTimeoutより大きくてtapHoldTimeoutより小さいべきです。
xSwipeDistanceThreshold	uint8_t	0～255	これは左右ｽﾜｲﾌﾟ手ぶり検出のためのX軸方向移動距離を制御します。
ySwipeDistanceThreshold	uint8_t	0～255	これは上下ｽﾜｲﾌﾟ手ぶり検出のためのY軸方向移動距離を制御します。
edgeSwipeDistanceThreshold	uint8_t	0～255	これは端ｽﾜｲﾌﾟ手ぶりに対する移動距離を制御します。
tapDistanceThreshold	uint8_t	0～255	このﾊﾟﾗﾒｰﾀは指が外される時にﾀｯﾌﾟ手ぶり、一定時間指が外されない場合にﾀｯﾌﾟ/保持手ぶりと見做されるために指がその内に留まらなければならない領域に対する境界。
seqTapDistanceThreshold	uint8_t	0～255	このﾊﾟﾗﾒｰﾀは直前の接触の離昇位置から現在の接触の初期押下の許容可能な距離を制限します。複数ﾀｯﾌﾟ(2重ﾀｯﾌﾟ、3重ﾀｯﾌﾟなど)に使われます。最初に後続するﾀｯﾌﾟがこの閾値内の場合、ﾀｯﾌﾟ計数器が増やされます。 後続するﾀｯﾌﾟ手ぶりがこの閾値を超える場合、直前の接触は単一ﾀｯﾌﾟとして送られ、現在の接触はﾀｯﾌﾟ計数器をﾘｾｯﾄします。
edgeBoundary	uint8_t	0～255	ﾌｧｰﾑｳｪｱは接触感知器の境界に沿って端領域を定義するために変更することもできます。この定義で、端領域で始まるｽﾜｲﾌﾟ手ぶりは標準ｽﾜｲﾌﾟ手ぶりの代わりに端ｽﾜｲﾌﾟ手ぶりとして報告されます。
wheelPostscaler	uint8_t	-128～127	このﾊﾟﾗﾒｰﾀはGUIで輪手ぶりが更新される速度を調整します。
wheelStartQuadrantCount	uint8_t	-128～127	輪手ぶりの動きは90°の円弧に分解することができます。ﾌｧｰﾑｳｪｱは輪手ぶり情報を報告を始める前に円様式で起きる或る円弧数に対して監視します。最初に検出されなければならない円弧数はこのﾊﾟﾗﾒｰﾀによって決められます。このﾊﾟﾗﾒｰﾀに対するより小さな値は輪手ぶり開始をより速くしますが、輪手ぶり情報を途中で報告しがちなﾌｧｰﾑｳｪｱにもします。
wheelReverseQuadrantCount	uint8_t	-128～127	このﾊﾟﾗﾒｰﾀは新規開始の代わりに輪の方向が変わる時に使われるのを除き、wheelStartQuadrantCountのような機能です。これは方向間で急速な交互切り替わりを避けるのに使われます。
pinchZoomThreshold	uint8_t	0～255	このﾊﾟﾗﾒｰﾀはﾋﾟﾝﾁとｽﾞｰﾑの手ぶりを検出するために2つの指間で許容可能な距離を制限します。このﾊﾟﾗﾒｰﾀ値を横切った後、接点間の距離が減る場合、手ぶりは"PINCH"として報告されます。このﾊﾟﾗﾒｰﾀ値を横切った後、接点間の距離が増す場合、手ぶりは"ZOOM"として報告されます。

qtm_gestures_2d_data_t

qtm_gestures_2d_data_tﾃﾞｰﾀ構造体は手ぶりを識別するのと状態と情報を格納するために単位部内で内部的に使われます。

領域	単位	範囲/任意選択	ﾊﾟﾗﾒｰﾀ
gestures_status	uint8_t	0または1	これは手ぶりが復号されるか否かを示します。
gestures_which_gesture	uint8_t	0～255	これは現在復号された手ぶりを含みます。
gestures_info	uint8_t	0～255	これは付加的な手ぶり情報を含みます。

15. 結合層単位部

15.1. 概要

結合層はQTouchﾗｲﾌﾞﾗﾘ単位部を結合して単位部の初期化と処理を自動化する一般的な枠組みです。結合層は適切な手順で単位部API関数を実行するのに使われるQTouch単位部のﾃﾞｰﾀ位置指示子(ﾎﾟｲﾝﾀ)と関数位置指示子(ﾎﾟｲﾝﾀ)で構成設定されます。結合単位部は使用者応用に対する全ての段階の完了での呼び戻しも提供します。

結合は採取単位部、信号調整単位部、後処理単位部を含みます。統一した応用ｲﾝﾀｰﾌｪｰｽで全ての単位部を制御することは多数の単位部、それらの状態と異常と呼び戻し関数を扱う複雑さを減らします。使用者応用ｺｰﾄﾞはﾗｲﾌﾞﾗﾘ単位部としても構築することができ、使用者単位部がQTouch部品化ﾗｲﾌﾞﾗﾘ基本構造に従う場合に結合層を使って自動化することができます。

図15-1. 結合層枠組み構成図

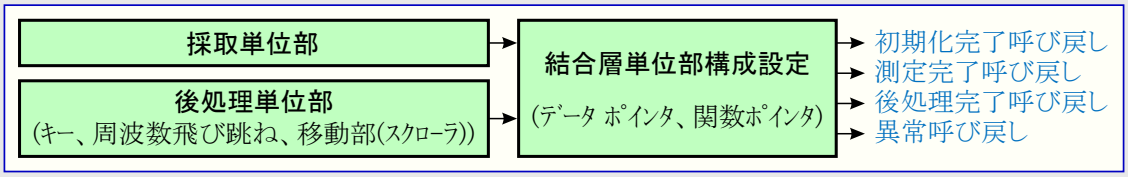


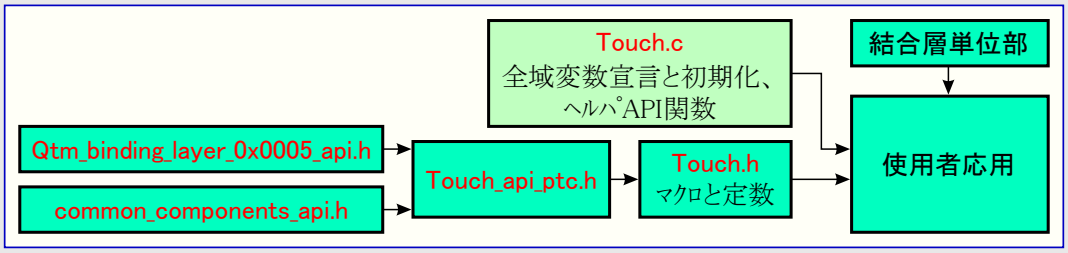
表15-1. 単位部ﾌｧｲﾙ

GCCｺﾝﾊﾟｲﾗ	IARｺﾝﾊﾟｲﾗ (AVR® MCU)	IARｺﾝﾊﾟｲﾗ (Arm® MCU)
libqtm_binding_layer_xxxxx_0x0005.a	qtm_binding_layer_xxxxx_0x0005.r90	qtm_binding_layer_xxxxx_0x0002.a

注: “xxxxx”は目的対象のﾌﾟﾛｾｯｻまたは基本構造を示します。

15.2. ｲﾝﾀｰﾌｪｰｽ

ﾃﾞｰﾀ構造体定義とAPI宣言は’qtm_binding_layer_0x0005_api.h’APIﾌｧｲﾙに含まれます。ﾃﾞｰﾀ構造体は全ての構成設定と出力ﾃﾞｰﾀ変数を網羅します。このﾌｧｲﾙは’touch_ptc_api.h’共通APIﾌｧｲﾙでｲﾝｸﾙｰﾄﾞされるべきです。

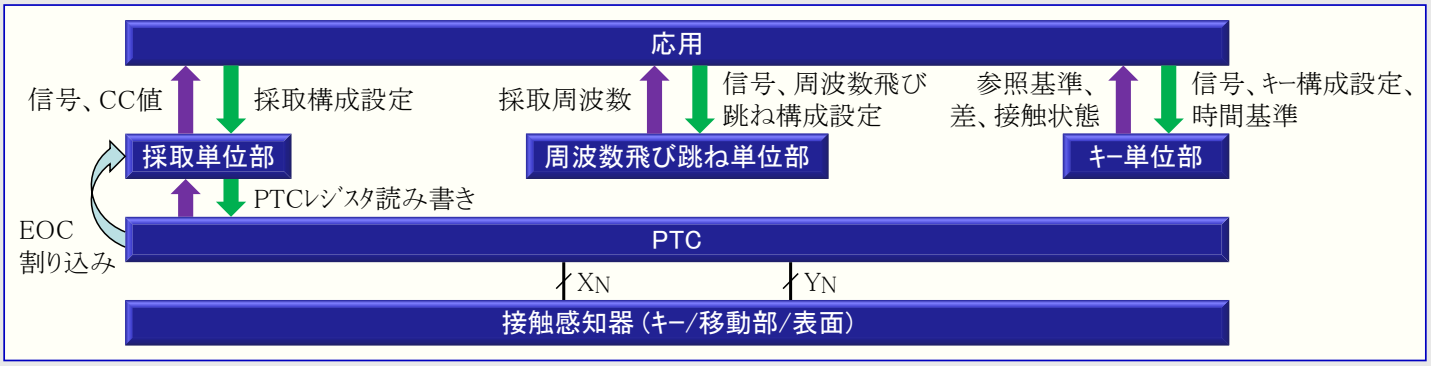


15.3. 機能的な説明

結合層は各単位部の以下の処理を自動化します。

1. 単位部初期化
2. 成功/異常の捕獲と呼び戻しを通す報告
3. 単位部後処理
4. 成功/異常の捕獲と呼び戻しを通す報告
5. ”単位部再収集”ﾌﾗｸﾞ捕獲と”再収集”状態に基づく採取の起動

図15-2. QTouch®応用に基づく結合層



結合層単位部によって支援される異常処理

個別単位部の異常は結合層内側で確認され、それらは符号化されて単一の異常符号として応用に渡されます。

異常符号はtouch.cファイルで復号されデータ可視器(Data Visualizer)ソフトウェアで表示されます。異常符号形式は下で与えられます。

採取単位部異常符号: 0x8<異常符号>
- 0x81 - QTM初期化
- 0x82 - 採取開始
- 0x83 - 感知器校正
- 0x84 - ハードウェア構成

後処理単位部異常符号: 0x4<処理ID>
- 0x40, 0x41, 0x42, ～

処理IDは#define LIB_MODULES_PROC_LISTマクロで一覧にされる処理IDの手順です。
処理IDは0から始まり、最大が15です。

例: 0x40 -> 後処理単位部1での異常
0x42 -> 後処理単位部3での異常

復号した単位部異常符号:
- 採取単位部異常 = 1
- 後処理単位部1異常 = 2
- 後処理単位部2異常 = 3
- ～ など

15.4. 構成設定

15.4.1. データ構造体

結合層の構成設定全体を保持する容器(コンテナ)構造体が下で与えられます。

```
typedef struct qtm_control_tag
{
	uint8_t binding_layer_flags;

	module_init_t *library_modules_init;
	module_proc_t *library_modules_proc;
	module_acq_t  *library_modules_acq;

	module_arg_t *library_module_init_data_model;
	module_arg_t *library_module_proc_data_model;
	module_arg_t *library_modules_acq_dm;

	qtm_acq_pp_t *qtm_acq_pp;

	/******************/
	/* 結合層用呼び戻し */
	/******************/
	qtm_library_init_complete_t qtm_init_complete_callback;
	qtm_error_callback_t	qtm_error_callback;
	qtm_measure_complete_t	qtm_measure_complete_callback;
	qtm_pre_process_callback_tqtm_pre_process_callback;
	qtm_post_process_callback_t	qtm_post_process_callback;
} qtm_control_t;
```

パラメータ	説明
*library_modules_init	単位部初期化関数ポインタの一覧を含む配列への位置指示子(ポインタ)
*library_modules_proc	単位部後処理関数ポインタの一覧を含む配列への位置指示子
*library_modules_acq	採取単位部関数ポインタの一覧を含む配列への位置指示子
*library_module_init_data_model	採取単位部のデータ ポインタを含む配列への位置指示子
*library_module_proc_data_model	後処理単位部のデータ ポインタを含む配列への位置指示子
*library_modules_acq_dm	採取群のポインタを含む配列への位置指示子
qtm_init_complete_callback	全単位部初期化実行後に結合層単位部によって提供される呼び戻し
qtm_error_callback	単位部処理中に結合層で遭遇した何かの異常がある場合にだけ起動される呼び戻し関数
qtm_measure_complete_callback	測定完了後の後処理前に結合層単位部によって起動される呼び戻し
qtm_pre_process_callback	採取処理後の後処理前に起動される呼び戻し。これは独自濾波単位部の実装を使用者に許すために提供されます。
qtm_post_process_callback	全後処理単位部完了後結合層単位部によって起動される呼び戻し

15.4.2. 状態と出力データ

パラメータ	説明
binding_layer_flags	更なる応用の処理を実行するために結合層呼び戻し関数内で3つのフラグが設定されます。 3つの結合層フラグは下のように現在の版で支援されます。
	time_to_measure_touch： このフラグは何れかの単位部再収集が要求された時に計時器ISR処理部で設定されます。このフラグは上の条件の1つが満たされる時に測定を起動するのに使われます。
	node_pp_request： このフラグは後処理が要求されたことを示すために測定完了呼び戻しで設定されます。このフラグはtouch_process関数で処理されます。
	reburst_request： このフラグは後処理完了呼び戻しで設定され、これは個別単位部再収集フラグに基づいて設定されます。このフラグはtouch_process関数で処理されます。

16. Atmel STARTを使う応用構築

Atmel STARTはMicrochipマイクロ コントローラに対してソフトウェア構成部品を選んで構成設定する使用者を手助けします。QTouchプロジェクトはAtmel STARTを使って作成されます。使用者は感知器を追加して画像的な方法で表されるQTouchパラメータを構成設定することができます。作成されたプロジェクトはGCCとIARのコンパイラを支援します。

Atmel START基盤を使ってQTouchプロジェクトを作成する方法のより多くの情報については以下のリンクを参照してください。
http://microchipdeveloper.com/touch:introduction-to-qtouch-project-creation

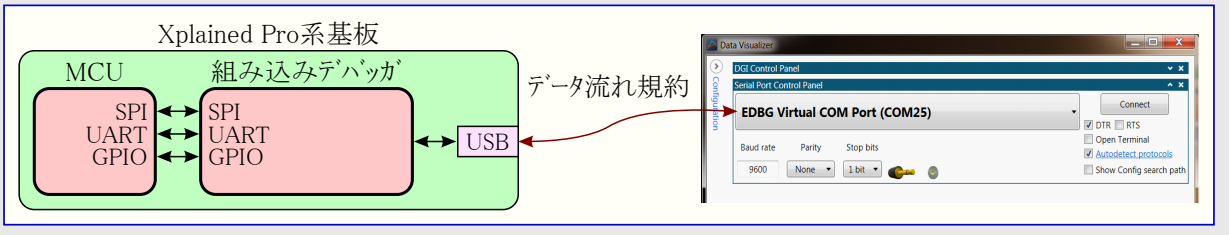
START基盤を使うSAML1xデバイス用の接触プロジェクト作成については以下のリンクを参照してください。
http://microchipdeveloper.com/touch:generate-saml1x-touch-project

17. QTouch®応用でのデータ可視器の使用

17.1. 概要

データ可視器(DV:Data Visualizer)は目的対象ハードウェアからの実行時データの処理と可視化に使われるプログラムです。データ可視器は組み込みデバッガ データ交換器インターフェース(DGI:Data Gateway Interface)とシリアル ポート(COMポート)のような様々な供給元からデータを受け取ることができます。

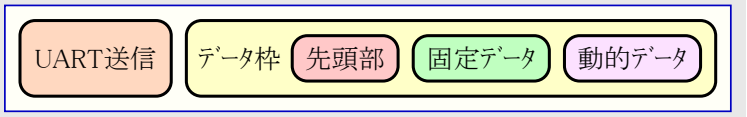
目的対象ハードウェアとのデータ可視器の代表的な接続模式が下で示されます。



17.2. ﾃﾞｰﾀ流し単位部

ﾃﾞｰﾀ流し単位部は単方向ﾃﾞｰﾀ転送規約とﾃﾞｰﾀ可視器(Data Visualizer)ｿﾌﾄｳｪｱに送られるﾃﾞｰﾀ枠を組み込みます。ﾃﾞｰﾀ流し部の現在の版はUARTﾎﾟｰﾄ通信だけの支援を提供します。

図17-1. ﾃﾞｰﾀ流し単位部構成図



UART送信

UART送信関数はﾃﾞﾊﾞｲｽ依存でAtmel STARTが自動的に正しいﾄﾞﾗｲﾊﾞを取り出して使用者基板/ｷｯﾄ例ﾌﾟﾛｼﾞｪｸﾄにそれを含めます。全てのﾃﾞﾊﾞｲｽで簡単な非同期動作(非割り込み駆動)のﾄﾞﾗｲﾊﾞが使われます。

ﾃﾞｰﾀ枠

ﾃﾞｰﾀ枠は先頭部、固定単位部ﾃﾞｰﾀ、動的単位部ﾃﾞｰﾀのﾊﾞｲﾄを含みます。

先頭部詳細:

先頭部は19ﾊﾞｲﾄを含み、ﾊﾟｹｯﾄの一部として送信されることが必要です。先頭部はﾊﾟｹｯﾄ毎に送信される必要はなく、むしろ15ﾊﾟｹｯﾄ送信毎に1度送信されます。先頭部ﾊﾟｹｯﾄ詳細は下で一覧にされます。

```
// uint8_t data[] =
// {
//     0x5F,
//     0xB4,  0x00,  0x86,  0x4A,
//     0x51,  0x54,  0x38,  0x31,  0x37,  0x54,  0x4F,  0x55,  0x43,  0x48,  0x55,  0xAA,
//     0xF9,
//     0xA0
// };
```

ﾊﾞｲﾄ	説明
ﾊﾞｲﾄ0	開始通票。固定値'$5F'を含みます。
ﾊﾞｲﾄ1～14	ﾁｪｯｸｻﾑ形式。LRC8に対応(開始と終了の通票を除くﾊﾟｹｯﾄのXOR総和)
ﾊﾞｲﾄ15,16	GUID、目的対象ﾊｰﾄﾞｳｪｱに対する識別子
ﾊﾞｲﾄ17	先頭部ﾊﾟｹｯﾄのﾁｪｯｸｻﾑ
ﾊﾞｲﾄ18	終了通票。固定値'$A0'を含みます。

固定単位部ﾃﾞｰﾀ:

1. 全ての構成設定された卸感知器の基本的な卸感知器ﾃﾞｰﾀ
2. 異常状態ﾃﾞｰﾀ

動的単位部ﾃﾞｰﾀ:

1. Atmel START QTouch構成設定部で自動調節が許可される時に採取自動調節ﾊﾟﾗﾒｰﾀが含まれます。
2. Atmel STARTで構成設定が行われたとおりに周波数飛び跳ね自動調節ﾃﾞｰﾀが含まれます。
3. Atmel STARTで摺動子/輪の感知器が構成設定された時に移動部(Scroller)単位部ﾊﾟﾗﾒｰﾀが送信されます。

17.3. ﾃﾞｰﾀ可視器を使うﾃﾞﾊﾞｯｸﾞ

ﾃﾞｰﾀ可視器(Data Visualizer)は端末、表題(Label)、図表などのようなﾃﾞｰﾀを可視化するための多くのGUI構成部品(ｳｨｼﾞｪｯﾄ)を支援します。継続するﾃﾞｰﾀとそれらの型が解析され、.db、.ds、.scの拡張子を持つ3つのｽｸﾘﾌﾟﾄ ﾌｧｲﾙを使って適切な構成部分で表示されます。これらのｽｸﾘﾌﾟﾄ ﾌｧｲﾙはﾌﾟﾛｼﾞｪｸﾄ構成設定に基づいてAtmel START基盤によって自動的に生成されます。、ｽｸﾘﾌﾟﾄのﾊﾟｽはﾃﾞｰﾀ可視器ｿﾌﾄｳｪｱで構成設定されることだけが必要です。ﾃﾞｰﾀ可視器ｿﾌﾄｳｪｱは独立型ｲﾝｽﾄｰﾙ可能版だけでなく、Atmel Studio IDEでの拡張としての両方で利用可能で、以下のﾘﾝｸからﾀﾞｳﾝﾛｰﾄﾞすることができます。

https://gallery.microchip.com/policies/studio

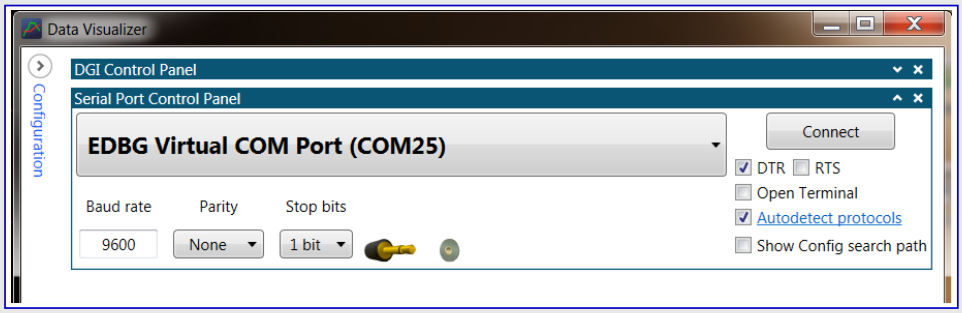
ﾃﾞﾊﾞｯｸﾞに対して使われる手順の流れが下で与えられます。

1. 望む位置でﾃﾞｰﾀ可視器用に"dv_config"構成設定ﾌｫﾙﾀﾞを作成してください。
2. "..¥thirdparty¥qtouch¥datastreamer"ﾌﾟﾛｼﾞｪｸﾄ ﾌｫﾙﾀﾞから"dv_config"ﾌｫﾙﾀﾞに計器盤構成設定(.db、.ds、.sc)ﾌｧｲﾙを複写してください。

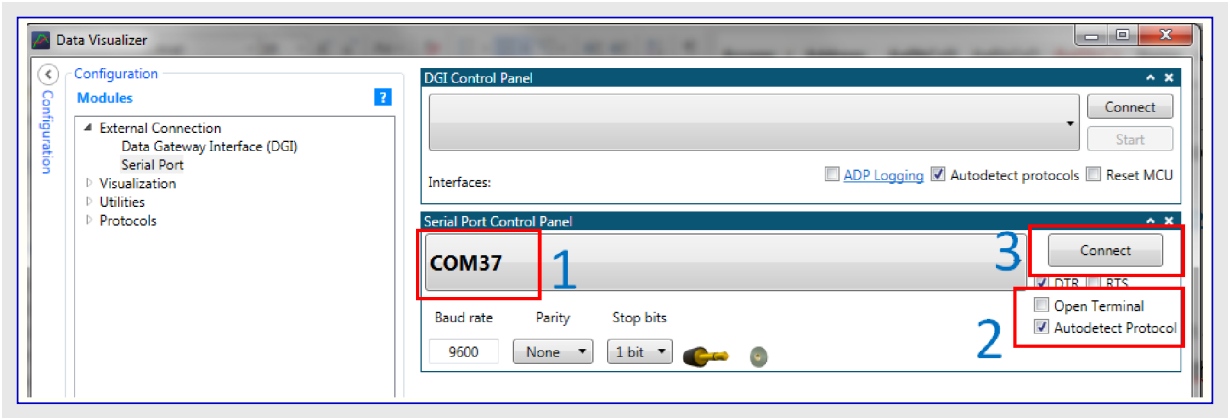
   注: これらのﾌｧｲﾙはｿｰｽ ﾌｧｲﾙではありません。これらはﾌﾟﾛｼﾞｪｸﾄ ﾌｫﾙﾀﾞへ自動的に抽出されません。これらのﾌｧｲﾙを抽出するにはselfcap_3ch.atzipﾌｧｲﾙをselfcap_3ch.zipに改名してください。その後に内容を(解凍)抽出してください。
3. ﾃﾞｰﾀ可視器(Data Visualizer)を開いてください。

注: QTouchﾃﾞﾊﾞｯｸﾞ ﾃﾞｰﾀがSPIまたはI²Cのｲﾝﾀｰﾌｪｰｽを使って送られる場合は段階6.へ行ってください。QTouchﾃﾞﾊﾞｯｸﾞ ﾃﾞｰﾀがCOMﾎﾟｰﾄを使って送られる場合は続けてください。

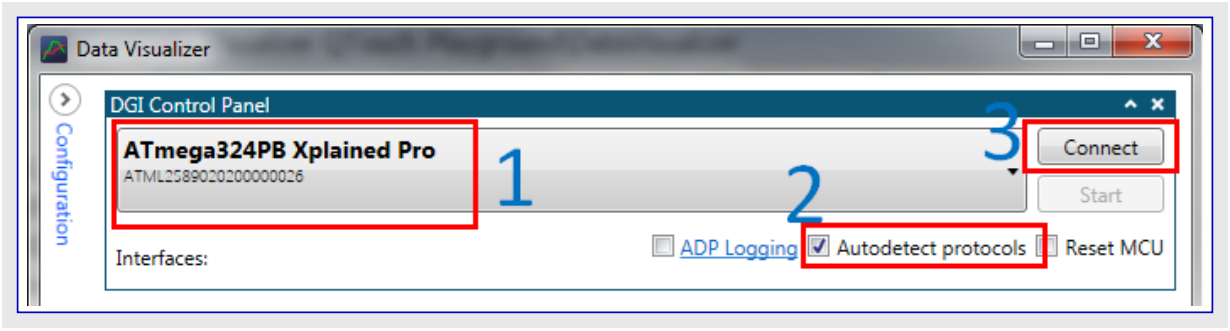
4. ｼﾘｱﾙ ﾎﾟｰﾄ制御操作盤(Control Panel)をﾀﾞﾌﾞﾙ ｸﾘｯｸし、目的対象への接続を作るために"接続(Connnect)"釦をｸﾘｯｸしてください。DGI制御操作盤(DGI Control Panel)ﾀﾌﾞを閉じてください。



5. 接続を作る別の方法は左側の構成設定(Configuration)任意選択を通すことです。構成設定(Configuration)任意選択を展開して外部接続(External Connection)任意選択下のｼﾘｱﾙ ﾎﾟｰﾄ(Serial Port)任意選択をﾀﾞﾌﾞﾙ ｸﾘｯｸし、ｼﾘｱﾙ ﾎﾟｰﾄ制御操作盤(Serial Port Control Panel)で"接続(Connect)"釦をｸﾘｯｸしてください。構成設定(Configuration)任意選択を最小化状態に戻してください。

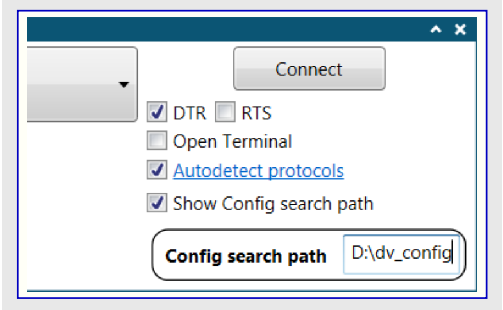


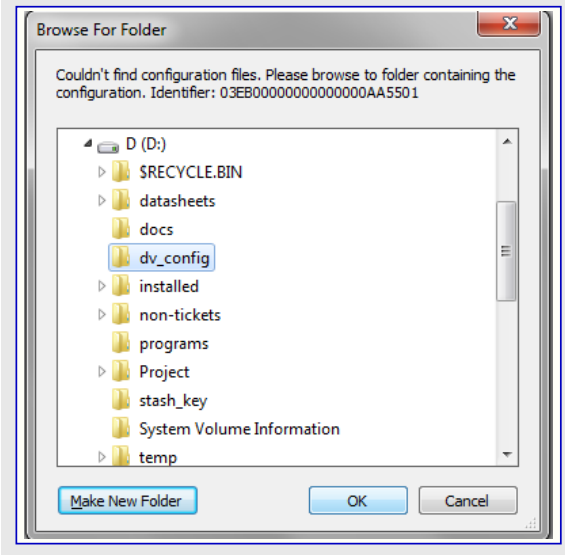
6. 望むｷｯﾄを選び、規約自動検出(Autodetect protocols)ﾁｪｯｸ枠を選んで接続(Connect)をｸﾘｯｸしてください。



7. まさに初回、ﾃﾞｰﾀ可視器(Data Visualizer)は右のように構成設定情報を含むﾌｫﾙﾀﾞを選ぶことを使用者に促します。検索して'dv_config'ﾌｫﾙﾀﾞを選んでOKをｸﾘｯｸしてください。

代わりに、'dv_config'構成設定ﾌｫﾙﾀﾞのﾊﾟｽを下で示されるように構成設定ﾊﾟｽ(config_path)ﾀﾌﾞで指定することができます。構成設定ﾊﾟｽ(config_path)ﾀﾌﾞを許可するには"構成設定検索ﾊﾟｽ表示(show config search path)"ﾁｪｯｸ枠任意選択をｸﾘｯｸしてください。





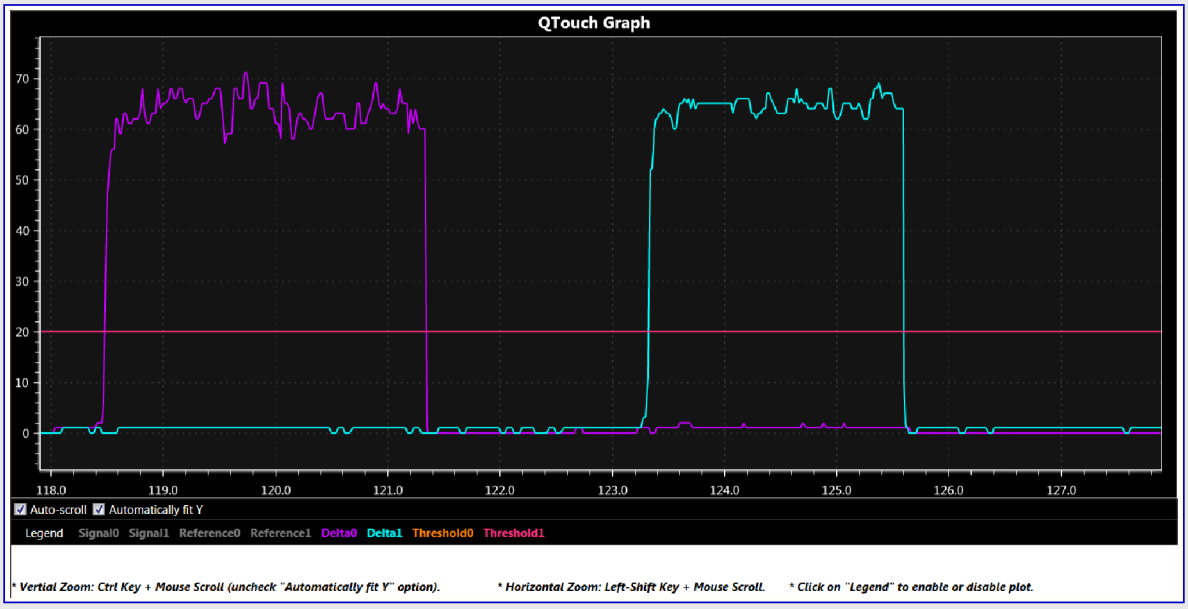
注: 選んだﾌｫﾙﾀﾞがﾃﾞｰﾀ可視器(Data Visualizer)に保存されます。ﾃﾞｰﾀ可視器は後続の接続に対してﾌｫﾙﾀﾞを選ぶことを使用者に促しません。感知器構成設定が変更された場合、Atmel STARTﾌﾟﾛｼﾞｪｸﾄからの新しい計器盤構成設定ﾌｧｲﾙがこのﾌｫﾙﾀﾞに複写されることが必要です。構成設定ﾌｧｲﾙ名が同じため、旧ﾌｧｲﾙは新しいもので置き換えられるべできます。ﾌｧｲﾙ名は変更されるべきではありません。

8. 計器盤表示は表示される3つのﾃﾞｰﾀ部分を含みます。最初の部分は差(Delta)と閾値(Threshold)に沿って構成設定された全ての釦の状態情報を変換します。

Button Data

Channel ID	Sensor Type	State	Delta	Threshold
0	Button 0	1	65	20
1	Button 1	0	2	20

9. 2つ目の部分は下で示されるように構成設定されたﾁｬﾈﾙの信号、参照基準、差値で作図された図表示を示します。作図は図表下部の凡例(Legend)上をｸﾘｯｸすることによって許可/禁止することができます。



10. 3つ目の部分は雑音耐性単位部からのﾃﾞｰﾀ、補償容量値を含む詳細な感知器情報、異常状態ﾃﾞｰﾀの表を表示します。

Sensor Data

Channel ID	Sensor Type	Signal	Reference	Delta	Compensation
0	Button 0	509	508	1	11495
1	Button 1	515	513	2	7287

Compensation: Represents PTC compensation circuit value which is equivalent to sensor capacitance

"Compenstation" value can be used to check whether sensor is saturated. Refer to User Guide

Frequency Hop Data

CurrentFrequency	1	HopFrequency0	0	HopFrequency1	1	HopFrequency2	2

Displayed frequencies are auto-tuned by QTouch Library based on noise levels

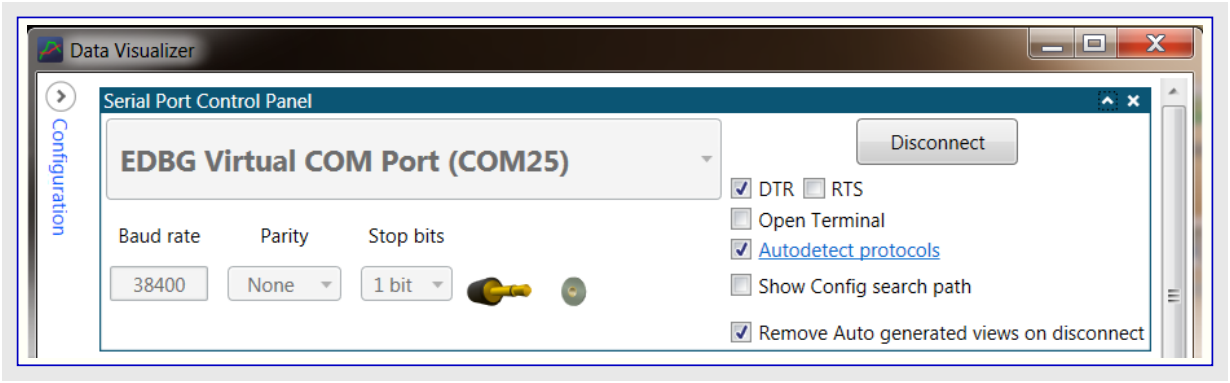
Debug Data

FrameCounter	41
QTouchLibError	0

Counter for datastreamer packets. Missing count indicate packet drop

Indicates library error state. Zero: no error. Refer "Error Code" section in User Guide

11. ハードウェアを切断するにはタブをダブル クリックすることによってシリアル ポート制御操作盤(Serial Port Control Panel)を開いて'切断(Disconnect)'釦をクリックしてください。



17.4. 2D接触表面ユーティリティを使うデバッグ

表面と手ぶりのプロジェクトのデバッグについては"2D接触表面ユーティリティ"と呼ばれるツールが使われます。

このツールと表面-手ぶりのプロジェクトについてより多くの詳細に関しては以下のリンクをご覧ください。

- http://microchipdeveloper.com/touch:generate-qtouch-surface-gesture-project
- http://microchipdeveloper.com/touch:guide-for-surface-sensor-design-using-modular-library
- http://microchipdeveloper.com/touch:guide-to-connect-to-touch-surface-utility

18. 調整手順

18.1. 雑音性能に対する調整

どの接触感知器応用でも、システム設計者は目的対象環境での電気的干渉が感知器の性能にどう影響を及ぼすかを考慮しなければなりません。

不充分な調整の接触感知器は放射または伝導される雑音のどちらかの試験で失敗を示すことがあり、それは機器の環境または電力領域で発生することがあり、または標準動作中に機器それ自身によって生成されるかもしれません。

多くの応用では、はっきりと定義されたEMC性能基準に合わなければならない品質規格があります。けれども規格に合わせることは規格が最も一般的に起こる雑音の形式と供給元を含むだけなので、システムが決してEMC問題を示さないことの証明と見做すことはできません。

雑音耐性は接触応答時間と電力消費の増加を犠牲にして手に入れます。システム設計者は最低電力消費を保証するために接触感知器の正しい調整を行わなければなりません。QTouch部品化ライブラリは接触応答時間、雑音耐性、電力消費間の最良の調和を与えるために調節することができるいくつかの使用者構成設定可能な機能を持ちます。

雑音源

接触感知器性能に影響を及ぼす雑音源は以下のような多種多様な応用で起こり得ます。

- 電動機
- 圧電ブザー
- PWM制御
- 蛍光灯
- 無線送信
- 電磁調理器
- 電源/充電器
- 主電源

適切なEMC規格

- 伝導耐性 EN61000-4-6
- 静電気放電(ESD) EN61000-4-2
- 電気的高速過渡現象(EFT) EN61000-4-4

雑音測定対策

雑音の影響は感知器上で誘発または注入された雑音信号の振幅とその雑音信号の周波数特性に依存します。

一般的に、この雑音は以下として分類することができます。

- 広帯域雑音

または

- 狭帯域雑音

広帯域雑音測定対策

広帯域雑音では周波数成分殆どが採取周波数の外側にあります。雑音信号と結合した採取信号の最大と最小の電圧水準が測定系の入力範囲内で充分大きな採取数が取られる場合、広帯域雑音妨害は大きな値の過採取設定によって平均化することができます。

過大な雑音振幅はｱﾅﾛｸﾞ前処理部を飽和し得ます。この場合、雑音信号と結合した採取信号は測定回路の入力範囲外で、測定の頭打ちに帰着します。

それが測定採取(試料)の一部でだけ起こるため、頭打ちは解決された測定で頻繁に観測できませんが、頭打ちした採取(試料)の存在は採取点の有効な平均化を妨げます。

この場合、採取(試料)の平均化は例え大きな過採取率でも雑音なし測定に帰着しません。解決された信号は信号頭打ちの非対称性のため、それの正しい水準からのずれを示ます。

18.1.1. 手順1: 頭打ち防止

これは採取信号と結合した雑音の規模を減らすためにﾊｰﾄﾞｳｪｱ低域通過濾波器の実装を必要とします。感知器容量はY(感知)線上でﾏｲｸﾛ ｺﾝﾄﾛｰﾗの内部または外部かもしれない直列抵抗と組み合わされます。

注: 内部直列抵抗はPTCでの相互容量動作でだけ利用可能です。

外部直列抵抗はESDの間中でその影響を減らすのに役立ちます。自己容量と相互容量の両方に対して可能な限り感知線でﾏｲｸﾛ ｺﾝﾄﾛｰﾗ ﾋﾟﾝの近くに最低1つの1kΩ外部直列抵抗を使ってください。

18.1.2. 手順2: 電荷転移試験

低域通過濾波器を形成する直列抵抗追加の影響のため、感知器の充電に対する時定数が増します。これは各測定採取間に感知器容量が完全に充電されて放電されることを保証するために重要です。

減らされた接触差や補償回路校正のために不充分な充電が観測され得ます。

けれども、この問題は接触感知器動作で直ぐに分からないかもしれず、応用は例え低水準雑音の存在でも上手く動くかもしれませんが、抵抗の追加での雑音試験の間、抵抗を除外した構成設定に比べてもっと悪い性能を示します。

電荷転移校正

QTouch部品化ﾗｲﾌﾞﾗﾘは完全な電荷転移を保証するために自動的にﾀｲﾐﾝｸﾞ ﾊﾟﾗﾒｰﾀを調節するための機能を提供します。

校正は目的対象ﾃﾞﾊﾞｲｽと測定技法に応じて、3つのﾊﾟﾗﾒｰﾀの1つを調節するように構成設定することができます。

CAL_AUTO_TUNE_RSEL　ｸﾛｯｸ前置分周器とCSDが構成設定された設定で保たれ、一方で内部直列抵抗は各感知器節点に対して適切な充電を許す最大値に調整されます。

- PTC相互容量採取でだけ利用可能

CAL_AUTO_TUNE_PRSC　直列抵抗とCSDが構成設定された設定で保たれ、一方で前置分周器は各感知器節点に対して適切な充電を許す最小値に調整されます。

- 1増加は採取時間を倍にし、1減少は採取時間を半分にします。

CAL_AUTO_TUNE_CSD　前置分周器と直列抵抗の両方が構成設定された設定で保たれ、充電共用遅延(CSD:Charge Share Delay)は各感知器節点に対して適切な充電を許す最小値に調整されます。

- CSDの1増加は採取手順の電荷転移段階に対して1周期を追加します。

18.1.3. 手順3: 過採取調整

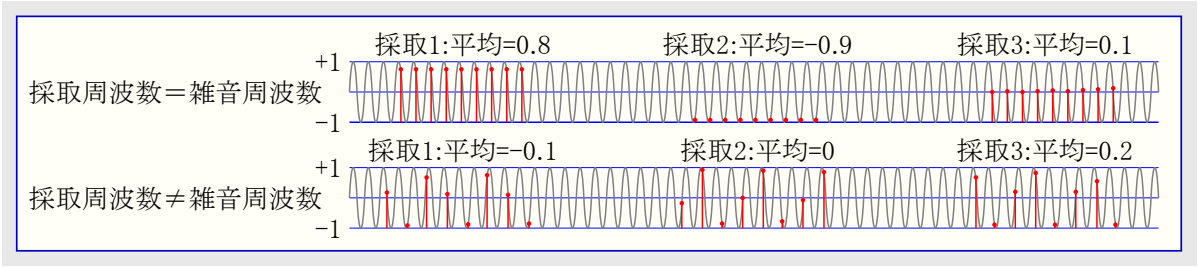
一旦、ﾊｰﾄﾞｳｪｱ濾波によって頭打ちが防がれて完全電荷転移が保証されると、次の段階は過採取に対する最適な設定を見つけることです。

これは応答時間と電力消費に対する雑音許容の間の二律背反です。より多くの採取はより良い品質ﾃﾞｰﾀを与えますが、取得するのにより長くかかります。

狭帯域雑音対策

雑音が容量測定周波数に関連する周波数成分を含む場合、過採取の量は雑音の影響を平均化しません。同じ採取周波数で取られた測定採取(試料)のどの束も、測定変位(ｵﾌｾｯﾄ)に帰着します。各測定から帰着する実際の変位は雑音成分の相対位相と採取周波数に依存します。

この影響は次の構成図で図解され、ここでの雑音は正弦波によって表現されます。



18.1.4. 手順4: 周波数動作選択

雑音が採取周波数と高調波的に関連する(または近い)周波数の場合、前で図解されたように、雑音問題は厳しくなります。この場合は雑音を避けるために過採取周波数が調節されなければなりません。

これはEN61000-4-6のような周波数掃引試験が必要とされる応用で特に重要です。

採取単位部(PTC)

利用可能な周波数 (PTCクロック=4MHz)

周波数選択	周波数 (kHz)	周波数選択	周波数 (kHz)	周波数選択	周波数 (kHz)
FREQ_SEL_0	66.67	FREQ_SEL_6	47.62	FREQ_SEL12	37.04
FREQ_SEL_1	62.5	FREQ_SEL_7	45.45	FREQ_SEL_13	35.71
FREQ_SEL_2	58.82	FREQ_SEL_8	43.48	FREQ_SEL_14	34.48
FREQ_SEL_3	55.56	FREQ_SEL_9	41.67	FREQ_SEL_15	33.33
FREQ_SEL_4	52.63	FREQ_SEL_10	40	FREQ_SEL_SPREAD	可変周波数
FREQ_SEL_5	50	FREQ_SEL_11	38.46		

採取単位部は周波数選択に対して以下の2つの方法を提供します。

1. 単一採取周波数が選ばれて過採取がこの周波数だけで行われます。FREQ_SEL_0は最高速、FREQ_SEL_15は最低速の測定を提供します。高性能EMC規格が必要とされないけれども、応用装置が特定採取周波数で妨害する雑音を生成する場合、設計者は単に周波数を変えるかもしれません。
2. 過採取中に可変周波数が使われます。FREQ_SEL_SPREADは取得過採取中に周波数を変えます。より広い採取周波数の分布を加えるために過採取中に連続する採取で遅延が鋸歯の規則で0～15に変えられます。単一周波数採取に比べて、周波数拡散任意選択は特定の'最悪'周波数での雑音に対する感度を減らしますが、雑音の影響の重大性を減らすにも関わらず、高調波妨害を示す最悪周波数周辺の雑音周波数の範囲を増します。多くの応用では必要とされる雑音許容を達成するのにFREQ_SEL_S PREADで充分です。

周波数飛び跳ね単位部 (単位部ID: 0x0006)

周波数飛び跳ね単位部は高調波妨害が伴われる測定を使うことを避けるため、3つ以上の基本周波数と中央値濾波器を利用します。周波数は問題の周波数帯内で交差高調波の組を最小にするように選ばれるべきです。

選ばれた周波数の各々が連続する測定周回中に過採取取得に使われます。

例18-1. 3周波数での周波数飛び跳ね

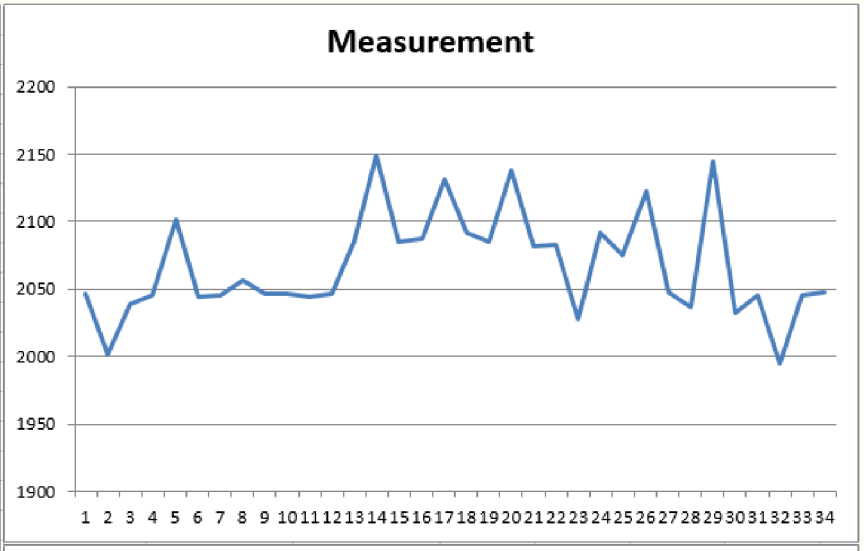
- **周回1**: 周波数0で全感知器測定
- **周回2**: 周波数1で全感知器測定
- **周回3**: 周波数2で全感知器測定
- **周回4**: 周波数0で全感知器測定
- **周回5**: 周波数1で全感知器測定

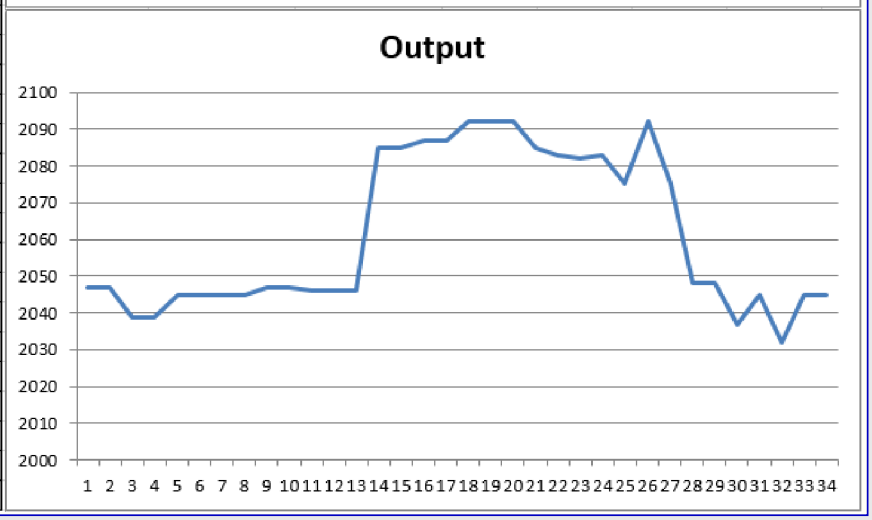
～

周波数0が雑音周波数に関係する場合、周波数0が伴われる測定は高い変化を示すでしょう。中央値濾波器を使うことにより、中心を離れた測定が除かれます。

図18-1. 雑音によって影響を及ぼされた周波数1を伴った測定

Cycle	Frequency	Measurement	Output
0	F0	2047	2047
1	F1	2001	2047
2	F2	2039	2039
3	F0	2045	2039
4	F1	2102	2045
5	F2	2044	2045
6	F0	2045	2045
7	F1	2056	2045
8	F2	2047	2047
9	F0	2046	2047
10	F1	2044	2046
11	F2	2046	2046
12	F0	2085	2046
13	F1	2149	2085
14	F2	2085	2085
15	F0	2087	2087
16	F1	2131	2087
17	F2	2092	2092
18	F0	2085	2092
19	F1	2138	2092
20	F2	2082	2085
21	F0	2083	2083
22	F1	2028	2082
23	F2	2092	2083
24	F0	2075	2075
25	F1	2122	2092
26	F2	2048	2075
27	F0	2037	2048
28	F1	2145	2048
29	F2	2032	2037
30	F0	2045	2045
31	F1	1995	2032
32	F2	2045	2045
33	F0	2048	2045





一般的な高調波

例えどの周波数が選ばれても、選んだ周波数の1つを超える高調波である、より高い周波数での雑音の可能性が存在します。

成分分布を更に上げると、利用可能な全周波数の全ての高調波である周波数がありますが、これらの高位群高調波はより高い周波数で、故に低域通過濾波器で阻止されます。

いくつかの応用では雑音周波数に晒される可能性が未知で変わりやすい量かもしれません。

例えば、USB充電器を利用する装置はそれで供給される充電器へ常に繋がれていないかもしれません。安価な代替充電器は度々可変周波数で、そして度々採取周波数と同じ帯域で、高い水準の同相雑音を生成します。

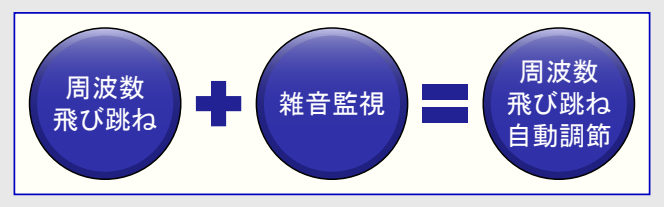
同様の状況は伝導耐性用のEN61000-4-6に対して検査される応用で起きます。検査装置は注入された150kHz～80MHzの雑音を通して1%の段階で掃引します。これは飛び跳ね周波数の共通高調波である妨害周波数を当てる素晴らしい機会を与えます。

両方の場合で、静的な周波数選択は中央値濾波器による高調波回避を保証できません。

自動調節付き周波数飛び跳ね単位部 (単位部ID: 0x0004)

自動調節付き周波数飛び跳ねは各個別周波数によって測定された信号の変化を定量化するために拡張された中央値濾波機能と共に巡回周波数飛び跳ねを提供します。

この単位部は安定限度で構成設定されます。特定の過採取周波数で測定された信号がこの限度を超える繰り返された変化を示す時に、単位部はこの周波数を別の周波数に切り替えてより良い実行任意選択を探します。



18.2. 摺動子/輪の感知器調整

例えば、2つの釦と3チャネルの自己容量摺動子が構成設定されます。仮に釦B0とB1、それと摺動子0[0]、摺動子0[1]、摺動子0[2]の摺動子感知器を形成するキー感知器とします。

釦と摺動子/輪のデータは下で示されるようにデータ可視器(Data Visualizer)制御操作盤表示部で表示されます。

Button Data

Channel ID	Sensor Type	State	Delta	Threshold
0	Button 0	0	0	20
1	Button 1	0	0	20
2	Slider 0[0]	0	1	70
3	Slider 0[1]	0	1	70
4	Slider 0[2]	0	0	70

Slider & Wheel Data

Sensor	State	SW Delta	SW Threshold	Position
Slider 0	0	0	60	0

以下の手順は摺動子/輪の感知器を調節するための手順を記述します。

手順1:

摺動子0[0]、摺動子0[1]、摺動子0[2]の各キー感知器で接触を行い、差の値を記録してください。例えば、摺動子0[0]で観測された差が下で示されます。

個別のキー感知器閾値を観測された差の値の半分の値に設定してください。この場合、キー閾値は60に設定されるべきです。

摺動子0[0]

Button Data

Channel ID	Sensor Type	State	Delta	Threshold
0	Button 0	0	10	20
1	Button 1	0	0	20
2	Slider 0[0]	1	119	70
3	Slider 0[1]	0	13	70
4	Slider 0[2]	0	7	70

摺動子0[1]

Button Data

Channel ID	Sensor Type	State	Delta	Threshold
0	Button 0	0	7	20
1	Button 1	0	0	20
2	Slider 0[0]	0	13	70
3	Slider 0[1]	1	123	70
4	Slider 0[2]	0	18	70

摺動子0[2]

Button Data

Channel ID	Sensor Type	State	Delta	Threshold
0	Button 0	0	14	20
1	Button 1	0	0	20
2	Slider 0[0]	0	3	70
3	Slider 0[1]	0	12	70
4	Slider 0[2]	1	115	70

手順2:

SW閾値として**手順1**.で計算されたキー閾値を設定して個別感知器で行われる接触が下で示されるような時に摺動子感知器が検出になることを確認してください。

Button Data

Channel ID	Sensor Type	State	Delta	Threshold
0	Button 0	0	10	20
1	Button 1	0	0	20
2	Slider 0[0]	1	123	60
3	Slider 0[1]	0	30	60
4	Slider 0[2]	0	8	60

Slider & Wheel Data

Sensor	State	SW Delta	SW Threshold	Position
Slider 0	1	149	60	4

Button Data

Channel ID	Sensor Type	State	Delta	Threshold
0	Button 0	0	11	20
1	Button 1	0	0	20
2	Slider 0[0]	0	10	60
3	Slider 0[1]	1	122	60
4	Slider 0[2]	0	35	60

Slider & Wheel Data

Sensor	State	SW Delta	SW Threshold	Position
Slider 0	1	152	60	157

Button Data

Channel ID	Sensor Type	State	Delta	Threshold
0	Button 0	0	15	20
1	Button 1	0	0	20
2	Slider 0[0]	0	2	60
3	Slider 0[1]	0	21	60
4	Slider 0[2]	1	121	60

Slider & Wheel Data

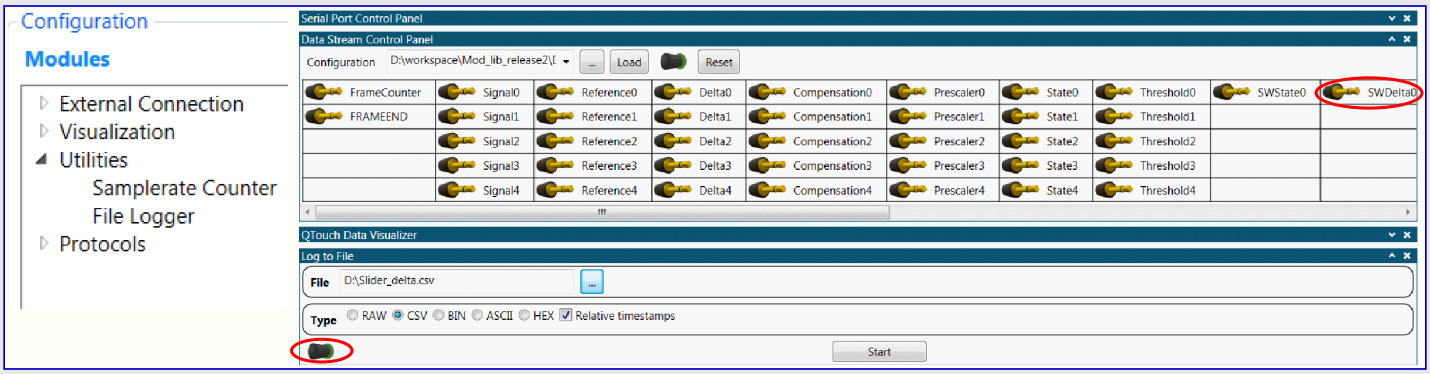
Sensor	State	SW Delta	SW Threshold	Position
Slider 0	1	137	60	255

手順3:

開始端と終了端間で摺動子感知器上を前後移動し、ファイル記録部(File Logger)ユーティリティを使ってCSVファイルでSW差(Delta)を記録してください。

ファイル記録部を使うには下の指示に従ってください。

1. デバッグ制御操作盤の下部で編集(Edit mode)枠をクリックすることによって編集動作に切り替えてください。
2. 構成設定(Configurations)⇒ユーティリティ(Utilities)⇒ファイル記録部(File Logger)でファイル記録部構成部分を開いてください。
3. 表題(タイトル)バーをダブル クリックすることによってQTouchデータ可視器(QTouch Data Visualizer)表示部を最小化してください。
4. ファイル記録部(File Logger)表示部でファイル検索部をクリックしてデータを記録するファイル名を選んでください。
5. 下図で示されるように、SWDelta0コネクタをクリックし、それを引き摺ってファイ記録部ソケットに接続してください。
6. 次にデータを記録するために開始/停止(Start/Stop)釦をクリックしてください。
7. 記録完了後、SWDelta0接続を取り外し、編集動作チェック枠のチェックを外してファイル記録部を閉じてください。



手順4:

ファイル記録を開いて採取試料から最低SWDelta0値を判別してください。Slider_delta.csvファイルから収集したSWDelta0の少数の見本が下で一覧にされます。

103,102,101,106,98,92,86,82,78,76,78,86,0,118,113,109,105,102,100,106,102,93,86,80,76,73,72,73,79,88,90,90,89,89,89,90,94,87,81,76,72,71,69,72,81,89,94,98,100,102

判別された最小値の"69"未満のSW閾値を設定してください。この場合、SW閾値は観測された値の54未満の15計数が設定されます。繰り返しの対について**手順3.**と**手順4.**を繰り返して記録に基づいてSW閾値を調節してください。

調節後の釦、摺動子/輪のデータが下で与えられます。

Button Data

Channel ID	Sensor Type	State	Delta	Threshold
0	Button 0	0	3	20
1	Button 1	0	0	20
2	Slider 0[0]	0	2	60
3	Slider 0[1]	0	49	60
4	Slider 0[2]	0	34	60

Slider & Wheel Data

Sensor	State	SW Delta	SW Threshold	Position
Slider 0	1	80	54	192

19. 既知の問題

通番	問題説明	区分	対処解決策
1	PTCが自己容量動作で使われる時に、内部直列抵抗が雑音低減に効果がありません。	PTC	外部直列抵抗を使うことが推奨されます。
2	いくつかのピンはより高い寄生容量を持ちます。詳細についてはhttps://microchip.wdfiles.com/local--files/touch:release-notes/QTouch_PTC_Information_7.1.xlsxの"Device Pin Capacitance(デバイス ピン容量)"シートを参照してください。	PTC	1. 代替Y線が利用可能な場合はこれらのピンを避けてください。 2. 接触にこのピンを使うことが必要とされる場合、応答時間に影響するかもしれないより高い充電時間を提供することが必要です。

20. 追補A – 改訂履歴

資料改訂	日付	注釈
42805A	2016年11月	初版資料公開
42805B	2017年2月	SAM D10/D11ﾗｲﾌﾞﾗﾘ情報追加
42805C	2017年6月	**改訂部分:** 接触ﾗｲﾌﾞﾗﾘ序説、ﾃﾞｰﾀ可視器(Data Visualizer)、Atmel START構成設定部 **追加:** 採取単位部、接触ｷｰ単位部、周波数飛び跳ね単位部、周波数飛び跳ね自動調節単位部、摺動子/輪単位部、結合層単位部、MISRA報告、各単位部用API参考、ｷｯﾄ例ﾌﾟﾛｼﾞｪｸﾄ、単位部命名規則、APIﾌｧｲﾙ ｲﾝﾀｰﾌｪｰｽ
A	2017年12月	Microchip DS40001986改訂AでAtmel 42805Cを置き換え **改訂部分:** 最新のQTouch構成設定部GUIでｷｯﾄ例/使用者基板ﾌﾟﾛｼﾞｪｸﾄ作成部分の複写画面を更新。 接触ｷｰ感知器用図画内包、序説項で図表内包 **追加:** 新しい「**既知の問題**」章追加
B	2018年5月	**追加**: 2D接触表面と手ぶりの単位部支援を追加
C	2018年6月	新ﾃﾞﾊﾞｲｽで「**単位部命名規則**」と「**追補B**」章を更新
D	2019年3月	新章「**4P並列採取単位部**」を追加
E	2020年3月	AVR-DAﾃﾞﾊﾞｲｽ用支援を追加 「**6.4.1. ﾃﾞｰﾀ構造体**」項で表を更新 「**7.1. 序説**」項に新しい内容を追加 「**7.2. 並列4P – 4から1**」項を削除 「**12.1. 概要**」、「**13.1. 概要**」、「**18.1. 雑音性能に対する調整**」、「**19. 既知の問題**」項で表を更新 他の些細な修正

21. 追補B – 採取単位部API参考

```
---------------------------------------------------------------------------
touch_ret_t qtm_acquisition_process(void)
---------------------------------------------------------------------------
目的：信号の捕獲と処理
入力：(測定された信号、構成設定)
出力：touch_ret_t
注意：'接触測定完了呼び戻し'後に応用によって呼ばれます。

---------------------------------------------------------------------------
touch_ret_t qtm_ptc_init_acquisition_module(qtm_acquisition_control_t* qtm_acq_control_ptr);
---------------------------------------------------------------------------
目的：PTC初期化とﾋﾟﾝ割り当て
入力：採取一式への位置指示子(ﾎﾟｲﾝﾀ)
出力：touch_ret_t：TOUCH_SUCCESS または INVALID_PARAM
注意：これは各構成設定一式への位置指示子と共に一度だけ呼ばれます。

---------------------------------------------------------------------------
touch_ret_t qtm_ptc_qtlib_assign_signal_memory(uint16_t* qtm_signal_raw_data_ptr);
---------------------------------------------------------------------------
目的：応用ｺｰﾄﾞで定義された配列への生信号位置指示子を割り当て
入力：生ﾃﾞｰﾀ配列への位置指示子
出力：touch_ret_t ：TOUCH_SUCCESS
注意：なし

---------------------------------------------------------------------------
touch_ret_t qtm_enable_sensor_node(qtm_acquisition_control_t* qtm_acq_control_ptr,
uint16_tqtm_which_node_number);
---------------------------------------------------------------------------
目的：測定用感知器節点を許可
入力：接点構成設定位置指示子、節点(ﾁｬﾈﾙ)番号
出力：touch_ret_t
注意：なし

---------------------------------------------------------------------------
touch_ret_t qtm_calibrate_sensor_node(ptc_seq_acq_settings* qtm_acq_control_l_ptr,
uint16_twhich_node_number)
---------------------------------------------------------------------------
目的：校正用感知器節点を記す。
入力：接点構成設定位置指示子、節点(ﾁｬﾈﾙ)番号
出力：touch_ret_t
注意：なし

---------------------------------------------------------------------------
touch_ret_t qtm_ptc_start_measurement_seq(qtm_acquisition_control_t* qtm_acq_control_pointer,
void (*measure_complete_callback) (void));
---------------------------------------------------------------------------
目的：最初のﾁｬﾈﾙに接触構成設定を設定して開始
入力：接点構成設定位置指示子、測定完了呼び戻し位置指示子
出力：touch_ret_t
注意：なし

---------------------------------------------------------------------------
touch_ret_t qtm_autoscan_sensor_node(qtm_auto_scan_config_t* qtm_auto_scan_config_ptr,
void(*auto_scan_callback)(void));
---------------------------------------------------------------------------
目的：窓比較器起き上がりで単一節点の休止動作測定にPTCを構成設定
入力：採取一式、ﾁｬﾈﾙ番号、閾値、走査起動元
出力：touch_ret_t
注意：なし

---------------------------------------------------------------------------
```

touch_ret_t qtm_autoscan_node_cancel(void)

目的：自動走査構成設定取り消し
入力：なし
出力：touch_ret_t
注意：なし

---
void qtm_ptc_de_init(void)

目的：PTCﾋﾟﾝ ﾚｼﾞｽﾀ解消、TOUCH_STATE_NULL設定
入力：なし
出力：なし
注意：このAPI関数はﾊｰﾄﾞｳｪｱの電力OFF/ONなしに実行時の間にPTCをﾘｾｯﾄするのに使われます。応用は走行時に応用を再始動するために他のｿﾌﾄｳｪｱ ﾘｾｯﾄ関数の一部としてこの関数を含めれるかもしれません。

---
uint16_t qtm_<device_family>_acq_module_get_id(void)
適用可能な<device_family> = m328pb, m324pb, t81x, t161x, samd1x, samd20, samd21, samda1, same51, same53, same54, samd51, tiny321x, samc20, samc21, saml21, saml22, samha1, saml10, saml11, avr_da
---
目的：単位部IDを返します。
入力：なし
出力：単位部ID
注意：なし

---
uint8_t qtm_<device_family>_acq_module_get_version(void)
適用可能な<device_family> = m328pb, m324pb, t81x, t161x, samd1x, samd20, samd21, samda1, same51, same53, same54, samd51, tiny321x, samc20, samc21, saml21, saml22, samha1, saml10, saml11, avr_da
---
目的：単位部のﾌｧｰﾑｳｪｱ版番号を返します。
入力：なし
出力：単位部ID - 上位ﾆﾌﾞﾙが主 / 下位ﾆﾌﾞﾙが副
注意：なし

---
void qtm_ptc_clear_interrupt(void) -> ARM Cortex SAMD10, SAMD11, SAME51/E53/E54/D51

目的：EOC/WCOMP割り込みﾋﾞｯﾄを解消
入力：なし
出力：なし
注意：なし

---
void qtm_<device_family>_ptc_handler_eoc(void)
適用可能な<device_family> = m328pb, m324pb, t81x, t161x, samd1x, samd20, samd21, samda1, same51, same53, same54, samd51, tiny321x, samc20, samc21, saml21, saml22, samha1, saml10, saml11, avr_da
---
目的：測定を捕獲、次を開始か、または手順処理部の終了
入力：なし
出力：なし
注意：なし

---
void qtm_<device_family>_ptc_handler_wcomp(void)
適用可能な<device_family> = m328pb, m324pb, t81x, t161x, samd1x, samd20, samd21, samda1, same51, same53, same54, samd51, tiny321x, samc20, samc21, saml21, saml22, samha1, saml10, saml11, avr_da
---
目的：測定を捕獲、呼び戻しを呼び出し
入力：なし
出力：なし
注意：なし

22. 追補C – 周波数飛び跳ね単位部API参考

```
------------------------------------------------------------------------------
touch_ret_t qtm_freq_hop(qtm_freq_hop_control_t *qtm_freq_hop_control);
------------------------------------------------------------------------------
目的：周波数飛び跳ね処理を走行
入力：容器構造体への位置指示子(ﾎﾟｲﾝﾀ)
出力：touch_ret_t
注意：なし

------------------------------------------------------------------------------
uint16_t qtm_get_freq_hop_module_id(void)
------------------------------------------------------------------------------
目的：単位部IDを返します。
入力：なし
出力：単位部ID
注意：なし

------------------------------------------------------------------------------
uint8_t qtm_get_freq_hop_module_ver(void)
------------------------------------------------------------------------------
目的：単位部のﾌｧｰﾑｳｪｱ版番号を返します。
入力：なし
出力：単位部ID – 上位ﾆﾌﾞﾙが主 / 下位ﾆﾌﾞﾙが副
注意：なし
```

23. 追補D – 周波数飛び跳ね自動調整単位部API参考

```
------------------------------------------------------------------------------
touch_ret_t qtm_freq_hop_autotune(qtm_freq_hop_autotune_control_t *qtm_freq_hop_autotune_control);
------------------------------------------------------------------------------
目的：周波数飛び跳ね自動調節処理を走行
入力：容器構造体への位置指示子(ﾎﾟｲﾝﾀ)
出力：touch_ret_t
注意：なし

------------------------------------------------------------------------------
uint16_t qtm_get_freq_auto_module_id(void);
------------------------------------------------------------------------------
目的：単位部IDを返します。
入力：なし
出力：単位部ID
注意：なし

------------------------------------------------------------------------------
uint8_t qtm_get_freq_auto_module_ver(void);
------------------------------------------------------------------------------
目的：単位部のﾌｧｰﾑｳｪｱ版番号を返します。
入力：なし
出力：単位部ID – 上位ﾆﾌﾞﾙが主 / 下位ﾆﾌﾞﾙが副
注意：なし
```

24. 追補E – 接触ｷｰ単位部API参考

```
-------------------------------------------------------------------------------
touch_ret_t qtm_init_sensor_key(qtm_touch_key_control_t* qtm_lib_key_group_ptr,
 uint8_t which_sensor_key, qtm_acq_node_data_t* acq_lib_node_ptr)
-------------------------------------------------------------------------------
目的：接触ｷｰ感知器を初期化
入力：ｷｰ群制御ﾃﾞｰﾀへの位置指示子(ﾎﾟｲﾝﾀ)、ｷｰ番号、感知器節点状態と信号への位置指示子
出力：TOUCH_SUCCESS
注意：なし

-------------------------------------------------------------------------------
touch_ret_t qtm_key_sensors_process(qtm_touch_key_control_t* qtm_lib_key_group_ptr)
-------------------------------------------------------------------------------
目的：感知器ｷｰ後処理 (接触検出状態機構)
入力：ｷｰ群制御ﾃﾞｰﾀへの位置指示子
出力：TOUCH_SUCCESS
注意：なし

-------------------------------------------------------------------------------
touch_ret_t qtm_key_suspend(uint16_t which_sensor_key, qtm_touch_key_control_t* qtm_lib_key_group_ptr)
-------------------------------------------------------------------------------
目的：ｷｰに対する採取測定を一時停止
入力：ｷｰ番号、ｷｰ群制御ﾃﾞｰﾀへの位置指示子
出力：TOUCH_SUCCESS
注意：全感知器での継続走査を避けることによって節電するように、ｷｰを一時的に停止するのに使われます。このAPIを使って単
      一ｷｰを起こし感知器として働くように定義し、他の感知器を一時停止にすることができます。このAPI関数は再開API関数と
      共に働きます。

-------------------------------------------------------------------------------
touch_ret_t qtm_key_resume(uint16_t which_sensor_key, qtm_touch_key_control_t* qtm_lib_key_group_ptr)
-------------------------------------------------------------------------------
目的：ｷｰに対する採取測定を再開
入力：ｷｰ番号、ｷｰ群制御ﾃﾞｰﾀへの位置指示子
出力：TOUCH_SUCCESS
注意：一時的に感知器の走査を避けるために一時停止API関数と共に使うことができます。例えば、ｱｲﾄﾞﾙ時の間にｷｰのいくつか
      を一時停止にして定義されたｷｰでの接触に基づいて再開することができます。

-------------------------------------------------------------------------------
void update_qtlib_timer(uint16_t time_elapsed_since_update)
-------------------------------------------------------------------------------
目的：時間期間を持つ局所変数を更新
入力：最後の更新からのms数
出力：なし
注意：なし

-------------------------------------------------------------------------------
uint16_t qtm_get_touch_keys_module_id(void)
-------------------------------------------------------------------------------
目的：単位部IDを返します。
入力：なし
出力：単位部ID
注意：なし

-------------------------------------------------------------------------------
uint8_t qtm_get_touch_keys_module_ver(void)
-------------------------------------------------------------------------------
目的：単位部のﾌｧｰﾑｳｪｱ版番号を返します。
入力：なし
出力：単位部ID – 上位ﾆﾌﾞﾙが主 / 下位ﾆﾌﾞﾙが副
注意：なし
```

25. 追補F – 移動器単位部API参考

```
------------------------------------------------------------------------------
touch_ret_t qtm_init_scroller_module(qtm_scroller_control_t *qtm_scroller_control)
------------------------------------------------------------------------------
目的：移動部(scroller)を初期化
入力：移動部群制御ﾃﾞｰﾀへの位置指示子(ﾎﾟｲﾝﾀ)
出力：接触戻り状態値
注意：なし

------------------------------------------------------------------------------
touch_ret_t qtm_scroller_process(qtm_scroller_control_t *qtm_scroller_control)
------------------------------------------------------------------------------
目的：移動部位置計算と濾波
入力：移動部群制御ﾃﾞｰﾀへの位置指示子
出力：接触戻り状態値
注意：なし

------------------------------------------------------------------------------
uint16_t qtm_get_scroller_module_id(void)
------------------------------------------------------------------------------
目的：単位部IDを返します。
入力：なし
出力：単位部ID
注意：なし

------------------------------------------------------------------------------
uint8_t qtm_get_scroller_module_ver(void)
------------------------------------------------------------------------------
目的：単位部のﾌｧｰﾑｳｪｱ版番号を返します。
入力：なし
出力：単位部ID – 上位ﾆﾌﾞﾙが主 / 下位ﾆﾌﾞﾙが副
注意：なし
```

26. 追補G – 2D表面(1指接触)CS単位部

```
------------------------------------------------------------------------------
touch_ret_t qtm_init_surface_cs(qtm_surface_cs_control_t *qtm_surface_cs_control);
------------------------------------------------------------------------------
目的：移動部(scroller)を初期化
入力：移動部群制御ﾃﾞｰﾀへの位置指示子(ﾎﾟｲﾝﾀ)
出力：TOUCH_SUCCESS
注意：なし

------------------------------------------------------------------------------
touch_ret_t qtm_surface_cs_process(qtm_surface_cs_control_t *qtm_surface_cs_control);
------------------------------------------------------------------------------
目的：移動部位置計算と濾波
入力：移動部群制御ﾃﾞｰﾀへの位置指示子
出力：TOUCH_SUCCESS
注意：なし

------------------------------------------------------------------------------
uint16_t qtm_get_surface_cs_module_id(void);
------------------------------------------------------------------------------
目的：単位部IDを返します。
入力：なし
出力：単位部ID
注意：なし

------------------------------------------------------------------------------
uint8_t qtm_get_surface_cs_module_ver(void);
------------------------------------------------------------------------------
目的：単位部のﾌｧｰﾑｳｪｱ版番号を返します。
入力：なし
出力：単位部ID – 上位ﾆﾌﾞﾙが主 / 下位ﾆﾌﾞﾙが副
注意：なし
```

27. 追補H – 2D表面(2指接触)CS/2T単位部

```
------------------------------------------------------------------------------
touch_ret_t qtm_init_surface_cs2t(qtm_surface_cs2t_control_t *qtm_surface_cs2t_control);
------------------------------------------------------------------------------
目的：移動部(scroller)を初期化
入力：移動部群制御ﾃﾞｰﾀへの位置指示子(ﾎﾟｲﾝﾀ)
出力：TOUCH_SUCCESS
注意：なし

------------------------------------------------------------------------------
touch_ret_t qtm_surface_cs2t_process(qtm_surface_cs2t_control_t *qtm_surface_cs2t_control);
------------------------------------------------------------------------------
目的：移動部位置計算と濾波
入力：移動部群制御ﾃﾞｰﾀへの位置指示子
出力：TOUCH_SUCCESS
注意：なし

------------------------------------------------------------------------------
uint16_t qtm_get_surface_cs2t_module_id(void);
------------------------------------------------------------------------------
目的：単位部IDを返します。
入力：なし
出力：単位部ID
注意：なし

------------------------------------------------------------------------------
uint8_t qtm_get_surface_cs2t_module_ver(void);
------------------------------------------------------------------------------
目的：単位部のﾌｧｰﾑｳｪｱ版番号を返します。
入力：なし
出力：単位部ID – 上位ﾆﾌﾞﾙが主 / 下位ﾆﾌﾞﾙが副
注意：なし
```

28. 追補I – 手ぶり単位部

```
------------------------------------------------------------------------------
void qtm_gestures_2d_clearGesture(void);
------------------------------------------------------------------------------
目的：-
入力：なし
出力：なし
注意：なし

------------------------------------------------------------------------------
touch_ret_t qtm_init_gestures_2d(void);
------------------------------------------------------------------------------
目的：手ぶり追跡変数を初期化
入力：なし
出力：TOUCH_SUCCESS
注意：なし

------------------------------------------------------------------------------
touch_ret_t qtm_gestures_2d_process(qtm_gestures_2d_control_t *qtm_gestures_2d_control);
------------------------------------------------------------------------------
目的：手ぶりｴﾝｼﾞﾝが更新された接触情報を処理
入力：手ぶり制御構造体への位置指示子(ﾎﾟｲﾝﾀ)
出力：?TOUCH_SUCCESS?
注意：なし
```

```
------------------------------------------------------------------------------
uint16_t qtm_get_gesture_2d_module_id(void);
------------------------------------------------------------------------------
目的：単位部IDを返します。
入力：なし
出力：単位部ID
注意：なし

------------------------------------------------------------------------------
uint8_t qtm_get_gesture_2d_module_ver(void);
------------------------------------------------------------------------------
目的：単位部のﾌｧｰﾑｳｪｱ版番号を返します。
入力：なし
出力：単位部ID - 上位ﾆﾌﾞﾙが主 / 下位ﾆﾌﾞﾙが副
注意：なし
```

29. 追補J – 結合層単位部API参考

```
------------------------------------------------------------------------------
void qtm_binding_layer_init(qtm_control_t *qtm_control);
------------------------------------------------------------------------------
目的：この関数は位置指示子(ﾎﾟｲﾝﾀ)を使って個別単位部初期化関数を内部的に実行します。初期化出力に基づいて、
      init_complete_callback または the error_callback の関数が起動されます。
入力：結合層容器構造体への位置指示子
出力：なし
注意：なし

------------------------------------------------------------------------------
void qtm_lib_start_acquisition(qtm_control_t *qtm_control);
------------------------------------------------------------------------------
目的：この関数は感知器の測定を開始するための“qtm_ptc_start_measurement_seq関数を内部的に実行します。複数の採取群の
      関数が連続的に実行されます。
入力：結合層容器構造体への位置指示子
出力：なし
注意：なし

------------------------------------------------------------------------------
touch_ret_t qtm_lib_acq_process(void)
------------------------------------------------------------------------------
目的：採取後処理関数を実行。構成設定に従って複数群の採取後処理が連続的に実行されます。
入力：なし
出力：接触戻り状態値
注意：なし

------------------------------------------------------------------------------
touch_ret_t qtm_lib_post_process(void)
------------------------------------------------------------------------------
目的：個別単位部後処理を実行。実行される後処理の手順はqtm_config_tの構成設定に基づきます。
入力：なし
出力：接触戻り状態値
注意：なし

------------------------------------------------------------------------------
qtm_control_t* qmt_get_binding_layer_ptr(void)
------------------------------------------------------------------------------
目的：結合層容器構造体への位置指示子を返します。
入力：なし
出力：結合層容器構造体への位置指示子
注意：なし
```

```
-------------------------------------------------------------------------------
uint16_t qtm_get_lib_state(void)
-------------------------------------------------------------------------------
目的：結合層の状況を返します。
入力：なし
出力：単位部の状況
注意：なし

-------------------------------------------------------------------------------
uint16_t qtm_get_binding_layer_module_id(void)
-------------------------------------------------------------------------------
目的：単位部IDを返します。
入力：なし
出力：単位部ID
注意：なし

-------------------------------------------------------------------------------
uint8_t qtm_get_binding_layer_module_ver(void)
-------------------------------------------------------------------------------
目的：単位部のﾌｧｰﾑｳｪｱ版番号を返します。
入力：なし
出力：単位部ID - 上位ﾆﾌﾞﾙが主 / 下位ﾆﾌﾞﾙが副
注意：なし
```

30. 追補K - 支援ﾃﾞﾊﾞｲｽ

最新の支援ﾃﾞﾊﾞｲｽ一覧が次のﾘﾝｸで提供されます。http://microchipdeveloper.com/touch:release-notes

Microchipウェブ サイト

Microchipはhttp://www.microchip.com/で当社のウェブ サイト経由でのオンライン支援を提供します。このウェブ サイトはお客様がファイルや情報を容易に利用可能にするのに使われます。利用可能な情報のいくつかは以下を含みます。

- **製品支援** – データシートと障害情報、応用記述と試供プログラム、設計資源、使用者の手引きとハードウェア支援資料、最新ソフトウェア配布と保管されたソフトウェア
- **全般的な技術支援** – 良くある質問(FAQ)、技術支援要求、オンライン検討グループ、Microchip設計協力課程会員一覧
- **Microchipの事業** – 製品選択器と注文の手引き、最新Microchip報道発表、セミナーとイベントの一覧、Microchip営業所の一覧、代理店と代表する工場

製品変更通知サービス

Microchipの製品変更通知サービスはMicrochip製品を最新に保つのに役立ちます。加入者は指定した製品系統や興味のある開発ツールに関連する変更、更新、改訂、障害情報がある場合に必ず電子メール通知を受け取ります。

登録するにはhttp://www.microchip.com/pcnへ行って登録指示に従ってください。

お客様支援

Microchip製品の使用者は以下のいくつかのチャネルを通して支援を受け取ることができます。

- 代理店または販売会社
- 最寄りの営業所
- 組み込み解決技術者(ESE:Embedded Solutions Engineer)
- 技術支援

お客様は支援に関してこれらの代理店、販売会社、またはESEに連絡を取るべきです。最寄りの営業所もお客様の手助けに利用できます。営業所と位置の一覧はこの資料の後ろに含まれます。

技術支援はhttp://www.microchip.com/supportでのウェブ サイトを通して利用できます。

Microchipデバイス コード保護機能

Microchipデバイスでの以下のコード保護機能の詳細に注意してください。

- Microchip製品はそれら特定のMicrochipデータシートに含まれる仕様に合致します。
- Microchipは意図した方法と通常条件下で使われる時に、その製品系統が今日の市場でその種類の最も安全な系統の1つであると考えます。
- コード保護機能を破るのに使われる不正でおそらく違法な方法があります。当社の知る限りこれらの方法の全てはMicrochipのデータシートに含まれた動作仕様外の方法でMicrochip製品を使うことが必要です。おそらく、それを行う人は知的財産の窃盗に関与しています。
- Microchipはそれらのコードの完全性について心配されているお客様と共に働きたいと思います。
- Microchipや他のどの半導体製造業者もそれらのコードの安全を保証することはできません。コード保護は当社が製品を”破ることができない”として保証すると言うことを意味しません。

コード保護は常に進化しています。Microchipは当社製品のコード保護機能を継続的に改善することを約束します。Microchipのコード保護機能を破る試みはデジタル ミレニアム著作権法に違反するかもしれません。そのような行為があなたのソフトウェアや他の著作物に不正なアクセスを許す場合、その法律下の救済のために訴権を持つかもしれません。

法的通知

デバイス応用などに関してこの刊行物に含まれる情報は皆さまの便宜のためにだけ提供され、更新によって取り換えられるかもしれません。皆さまの応用が皆さまの仕様に合致するのを保証するのは皆さまの責任です。**Microchipはその条件、品質、性能、商品性、目的適合性を含め、明示的にも黙示的にもその情報に関連して書面または表記された書面または黙示の如何なる表明や保証もしません。**Microchipはこの情報とそれの使用から生じる全責任を否認します。生命維持や安全応用でのMicrochipデバイスの使用は完全に購入者の危険性で、購入者はそのような使用に起因する全ての損害、請求、訴訟、費用からMicrochipを擁護し、補償し、免責にすることに同意します。他に言及されない限り、Microchipのどの知的財産権下でも暗黙的または違う方法で許認可は譲渡されません。

商標

品質管理ｼｽﾃﾑ

Microchipの品質管理ｼｽﾃﾑに関する情報についてはhttp://www.microchip.com/qualityを訪ねてください。

日本語© *HERO* 2020.

本応用記述はMicrochipのQTuch部品化ﾗｲﾌﾞﾗﾘPTC使用者の手引き(DS40001986E-2020年3月)の翻訳日本語版です。日本語では不自然となる重複する形容表現は省略されている場合があります。日本語では難解となる表現は大幅に意訳されている部分もあります。必要に応じて一部加筆されています。頁割の変更により、原本より頁数が少なくなっています。

必要と思われる部分には( )内に英語表記や略称などを残す形で表記しています。

青字の部分はﾘﾝｸとなっています。一般的に赤字の0,1は論理0,1を表します。その他の赤字は重要な部分を表します。

世界的な販売とサービス

米国	亜細亜/太平洋	亜細亜/太平洋	欧州
本社 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 技術支援: http://www.microchip.com/support ｳｪﾌﾞｱﾄﾞﾚｽ: http://www.microchip.com	**ｵｰｽﾄﾗﾘｱ - ｼﾄﾞﾆｰ** Tel: 61-2-9868-6733 **中国 - 北京** Tel: 86-10-8569-7000 **中国 - 成都** Tel: 86-28-8665-5511 **中国 - 重慶** Tel: 86-23-8980-9588 **中国 - 東莞** Tel: 86-769-8702-9880 **中国 - 広州** Tel: 86-20-8755-8029 **中国 - 杭州** Tel: 86-571-8792-8115 **中国 - 香港特別行政区** Tel: 852-2943-5100 **中国 - 南京** Tel: 86-25-8473-2460 **中国 - 青島** Tel: 86-532-8502-7355 **中国 - 上海** Tel: 86-21-3326-8000 **中国 - 瀋陽** Tel: 86-24-2334-2829 **中国 - 深圳** Tel: 86-755-8864-2200 **中国 - 蘇州** Tel: 86-186-6233-1526 **中国 - 武漢** Tel: 86-27-5980-5300 **中国 - 西安** Tel: 86-29-8833-7252 **中国 - 廈門** Tel: 86-592-2388138 **中国 - 珠海** Tel: 86-756-3210040	**ｲﾝﾄﾞ - ﾊﾝｶﾞﾛｰﾙ** Tel: 91-80-3090-4444 **ｲﾝﾄﾞ - ﾆｭｰﾃﾞﾘｰ** Tel: 91-11-4160-8631 **ｲﾝﾄﾞ - ﾌﾟﾈｰ** Tel: 91-20-4121-0141 **日本 - 大阪** Tel: 81-6-6152-7160 **日本 - 東京** Tel: 81-3-6880-3770 **韓国 - 大邱** Tel: 82-53-744-4301 **韓国 - ｿｳﾙ** Tel: 82-2-554-7200 **ﾏﾚｰｼｱ - ｸｱﾗﾙﾝﾌﾟｰﾙ** Tel: 60-3-7651-7906 **ﾏﾚｰｼｱ - ﾍﾟﾅﾝ** Tel: 60-4-227-8870 **ﾌｨﾘﾋﾟﾝ - ﾏﾆﾗ** Tel: 63-2-634-9065 **ｼﾝｶﾞﾎﾟｰﾙ** Tel: 65-6334-8870 **台湾 - 新竹** Tel: 886-3-577-8366 **台湾 - 高雄** Tel: 886-7-213-7830 **台湾 - 台北** Tel: 886-2-2508-8600 **ﾀｲ - ﾊﾞﾝｺｸ** Tel: 66-2-694-1351 **ﾍﾞﾄﾅﾑ - ﾎｰﾁﾐﾝ** Tel: 84-28-5448-2100	**ｵｰｽﾄﾘｱ - ｳﾞｪﾙｽ** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **ﾃﾞﾝﾏｰｸ - ｺﾍﾟﾝﾊｰｹﾞﾝ** Tel: 45-4485-5910 Fax: 45-4485-2829 **ﾌｨﾝﾗﾝﾄﾞ - ｴｽﾎﾟｰ** Tel: 358-9-4520-820 **ﾌﾗﾝｽ - ﾊﾟﾘ** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **ﾄﾞｲﾂ - ｶﾞﾙﾋﾝｸﾞ** Tel: 49-8931-9700 **ﾄﾞｲﾂ - ﾊｰﾝ** Tel: 49-2129-3766400 **ﾄﾞｲﾂ - ﾊｲﾙﾌﾞﾛﾝ** Tel: 49-7131-72400 **ﾄﾞｲﾂ - ｶｰﾙｽﾙｰｴ** Tel: 49-721-625370 **ﾄﾞｲﾂ - ﾐｭﾝﾍﾝ** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **ﾄﾞｲﾂ - ﾛｰｾﾞﾝﾊｲﾑ** Tel: 49-8031-354-560 **ｲｽﾗｴﾙ - ﾗｰﾅﾅ** Tel: 972-9-744-7705 **ｲﾀﾘｱ - ﾐﾗﾉ** Tel: 39-0331-742611 Fax: 39-0331-466781 **ｲﾀﾘｱ - ﾊﾟﾄﾞﾊﾞ** Tel: 39-049-7625286 **ｵﾗﾝﾀﾞ - ﾃﾞﾙｰﾈﾝ** Tel: 31-416-690399 Fax: 31-416-690340 **ﾉﾙｳｪｰ - ﾄﾛﾝﾊｲﾑ** Tel: 47-72884388 **ﾎﾟｰﾗﾝﾄﾞ - ﾜﾙｼｬﾜ** Tel: 48-22-3325737 **ﾙｰﾏﾆｱ - ﾌﾞｶﾚｽﾄ** Tel: 40-21-407-87-50 **ｽﾍﾟｲﾝ - ﾏﾄﾞﾘｰﾄﾞ** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **ｽｳｪｰﾃﾞﾝ - ｲｪｰﾃﾎﾞﾘ** Tel: 46-31-704-60-40 **ｽｳｪｰﾃﾞﾝ - ｽﾄｯｸﾎﾙﾑ** Tel: 46-8-5090-4654 **ｲｷﾞﾘｽ - ｳｫｰｷﾝｶﾞﾑ** Tel: 44-118-921-5800 Fax: 44-118-921-5820