静電容量センサマイコン

静電容量センサユニット ソフトウエアフィルタ サンプルプログラム

要旨

本アプリケーションノートは、静電容量センサユニットシステム向けのソフトウエアフィルタについて説明します。

動作確認デバイス

RA2L1 グループ (R7FA2L1AB2DFP)

RX130 グループ (R5F51305ADFN)

RL78/G16 グループ (R5F121BCAFP)

目次

1. 概要

本アプリケーションノートでは、ソフトウエアフィルタ サンプルプログラムの動作及び、既存プロジェクトへの組み込み手順について説明します。

ソフトウエアフィルタの詳細については静電容量センサマイコン静電容量タッチノイズイミュニティガイド(R30AN0426)を参照してください。

1.1 動作確認条件

表 1-1～表 1-3 に本アプリケーションノートにおけるサンプルプログラムの動作確認条件を示します

表 1-1 動作確認条件(RA2L1 グループ)

項目	内容
使用マイコン	RA2L1 (R7FA2L1AB2DFP)
動作周波数	高速オンチップオシレータ 48MHz
動作電圧	5V
使用ボード	RA2L1 搭載静電容量タッチ評価システム (製品型名：RTK0EG0022S01001BJ) ● RA2L1 CPU ボード(型名：RTK0EG0018C01001BJ) ● 自己容量タッチボタン／ホイール／スライダボード (型名：RTK0EG0019B01002BJ)
統合開発環境	e² studio Version 2024-04 (24.4.0)
C コンパイラ	GCC Arm Embedded 13.2.1.arm-13-7
FSP	V5.3.0
静電容量式タッチセンサ対応開発支援ツール	QE for Capacitive Touch V3.3.0
エミュレータ	Renesas E2 エミュレータ Lite

表 1-2 動作確認条件(RX130 グループ)

項目	内容
使用マイコン	RX130 (R5F51305ADFN)
動作周波数	高速オンチップオシレータ 32MHz
動作電圧	5V
使用ボード	RX130 搭載静電容量タッチ評価システム (製品型名：RTK0EG0003S02001BJ) ● RX130 CPU ボード(型名：RTK0EG0004C01002BJ) ● 自己容量タッチボタン／ホイール／スライダボード (型名：RTK0EG0007B01002BJ)
統合開発環境	e² studio Version 2024-04 (24.4.0)
C コンパイラ	Renesas CC-RX V3.06.00
静電容量式タッチセンサ対応開発支援ツール	QE for Capacitive Touch V3.3.0
エミュレータ	Renesas E2 エミュレータ Lite

表 1-3 動作確認条件(RL78/G16 グループ)

項目	内容
使用マイコン	RL78/G16 (R7F100GSN2DFB)
動作周波数	高速オンチップオシレータ 32MHz
動作電圧	5V
使用ボード	RL78/G16 搭載静電容量タッチ評価システム (製品型名：RTK0EG0047S01001BJ) • RL78/G16 CPU ボード (型名：RTK0EG0046C01001BJ) • 自己容量タッチボタン／ホイール／スライダボード (型名：RTK0EG0019B01002BJ)
統合開発環境	e² studio Version 2024-04 (24.4.0)
C コンパイラ	Renesas CC-RL V1.13.00
静電容量式タッチセンサ対応開発支援ツール	QE for Capacitive Touch V3.3.0
エミュレータ	Renesas E2 エミュレータ Lite

図 1-1 に機器接続図を示します。

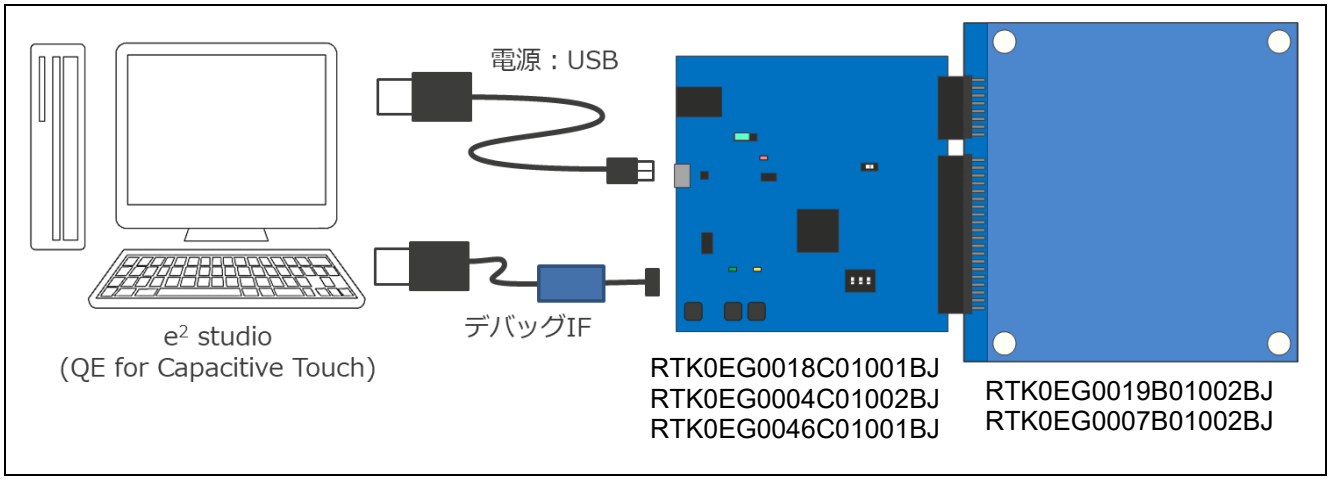


図 1-1 機器接続図

1.2　サンプルプロジェクト/サンプルコードと本書の対応

本サンプルプロジェクトを使用する前に「2.1 ソフトウエア構成図」および「2.3 ファイル構成」を参照してください。

既存の静電容量センサユニットプロジェクトへのフィルタモジュールの組み込み方法は「7.2 フィルタサンプルコード」、サンプルプロジェクトの使用方法は「7.1 サンプルプロジェクト」を参照してください。

各フィルタの仕様およびパラメータ設定方法は「2. ソフトウエア仕様」「3. FIR フィルタ」「4. IIR フィルタ」「5. 単極 IIR フィルタ」「6. メディアンフィルタ」「7.2.3 フィルタ特性の調整方法」を参照してください。

2.　ソフトウエア仕様

本サンプルプログラムは Touch API と CTSU API で取得した計測値にフィルタ API を適用することでソフトウエアフィルタとして動作します。使用するソフトウエアフィルタは実行するフィルタ API とフィルタ構成定義データで管理します。本サンプルプログラムで定義しているフィルタ構成は FIR フィルタ、IIR フィルタ、単極 IIR フィルタ、メディアンフィルタになります。

本サンプルプログラムの使用例として 1 ボタンを使用し、5 種類のフィルタを適用したタッチ検出を行うサンプルプロジェクトを提供します。

2.1　ソフトウエア構成図

図 2-1 に本サンプルプログラムのデータ処理フローを示します。

CTSU Driver（R_CTSU_DataGet()）で取得した計測値にソフトウエアフィルタを適用します。TOUCH Middleware でフィルタ適用計測値を使用するために、CTSU Driver（R_CTSU_DataInsert()）を使用して計測値とフィルタ適用計測値を差し換えます。フィルタ適用計測値を使用して、TOUCH Middleware（RM_TOUCH_DataGet()）でタッチ判定を行います。

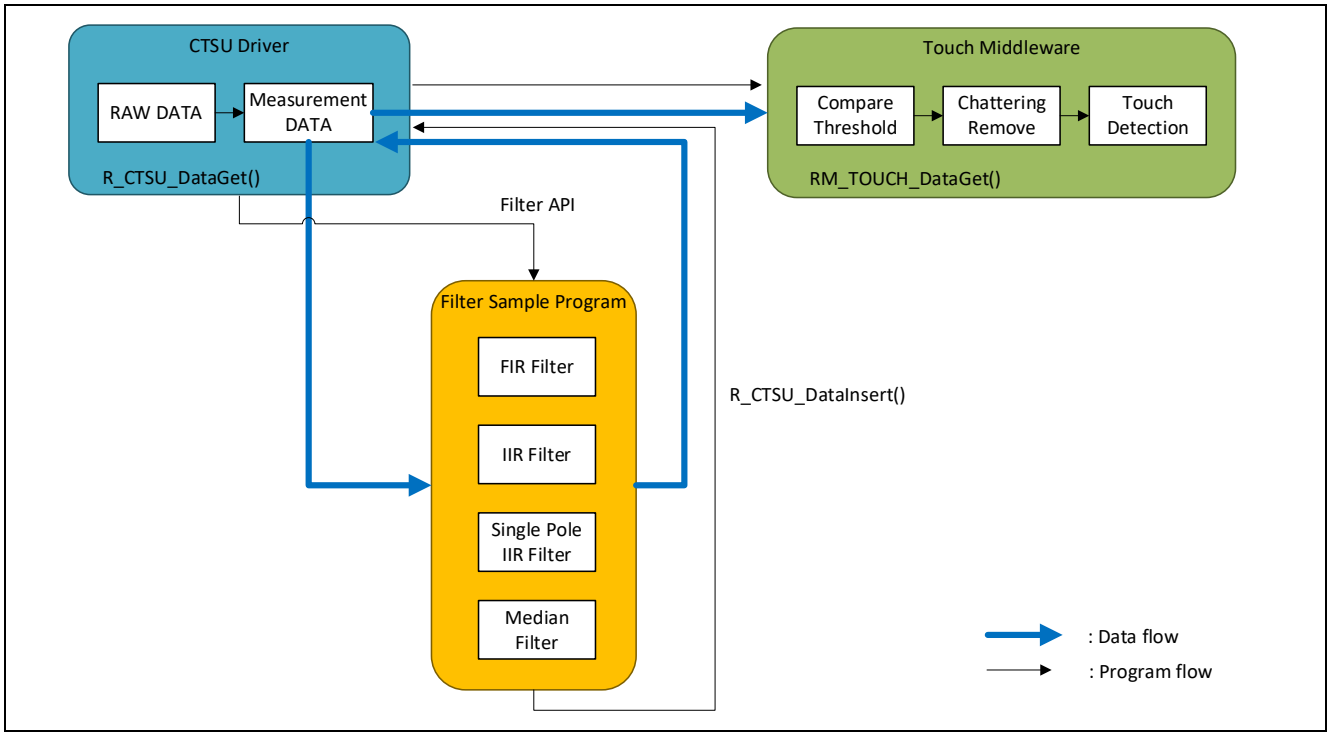


図 2-1 サンプルプログラムのデータ処理フロー

表 2-1~表 2-3 にコンポーネントとバージョンの一覧を示します。コンポーネントの設定は FSP Configuration で参照してください。

表 2-1 コンポーネント一覧（RA2L1）

コンポーネント	バージョン
Board Support Packages Common Files	v5.3.0
I/O Port	v5.3.0
Arm CMSIS Version 5 – Core(M)	v5.9.0+renesas.1.fsp.5.3.0
RA2L1-RSSK Board Support Files	v5.3.0
Board support package for R7FA2L1AB2DFP	v5.3.0
Board support package for RA2L1	v5.3.0
Board support package for RA2L1 – FSP Data	v5.3.0
Asynchronous General Purpose Timer	v5.3.0
Capacitive Touch Sensing Unit	v5.3.0
Touch	v5.3.0

表 2-2 コンポーネント一覧（RX130）

コンポーネント	バージョン
Board Support Packages	v7.42
CMT driver	v5.60
CTSU QE API	v2.20
Touch QE API	v2.20

表 2-3 コンポーネント一覧（RL78/G16）

コンポーネント	バージョン
Board Support Packages	v1.62
Capacitive Touch Sensing Unit driver	v1.40
Touch middleware	v1.40
UART 通信	v1.6.0
インターバル・タイマ	v1.4.0

2.2　ソフトウエアフィルタ種類

本サンプルプログラムは表 2-4 に記したフィルタ種類と使用例で構成されます。

表 2-4 フィルタ種類

フィルタ種類	使用例	想定ノイズ
FIR フィルタ	FIR 移動平均フィルタ	ホワイトノイズ、高周波ノイズ ハムノイズ(50Hz)
IIR フィルタ	IIR ノッチフィルタ	ハムノイズ(60Hz)
	IIR ローパスフィルタ	ハムノイズ(60Hz)
単極 IIR フィルタ	単極 IIR ローパスフィルタ	
メディアンフィルタ	メディアンフィルタ	

2.3　ファイル構成

本サンプルコードのファイル構成を示します。

```
an-r01an0427jj0400-capacitive-touch
├──filter_sample
│      ├──fir                          ・・・FIR フィルタサンプルモジュール格納フォルダ
│      │      fir_config_sample1.c     ・・・FIR 移動平均フィルタ用構成定義ソース
│      │      r_ctsu_fir_sample.c      ・・・FIR フィルタサンプルプログラムソース
│      │      r_ctsu_fir_sample.h      ・・・FIR フィルタサンプルプログラムヘッダ
│      ├──iir                          ・・・IIR フィルタサンプルモジュール格納フォルダ
│      │      iir_config_sample1.c     ・・・IIR ノッチ/ローパスフィルタ用構成定義ソース
│      │      r_ctsu_iir_sample.c      ・・・IIR フィルタサンプルプログラムソース
│      │      r_ctsu_iir_sample.h      ・・・IIR フィルタサンプルプログラムヘッダ
│      ├──spiir                        ・・・単極 IIR フィルタサンプルモジュール格納フォルダ
│      │      spiir_config_sample1.c   ・・・単極 IIR ローパスフィルタ用構成定義ソース
│      │      r_ctsu_spiir_sample.c    ・・・単極 IIR フィルタサンプルプログラムソース
│      │      r_ctsu_spiir_sample.h    ・・・単極 IIR フィルタサンプルプログラムヘッダ
│      └──median                       ・・・メディアンフィルタサンプルモジュール格納フォルダ
│             median_config_sample1.c  ・・・メディアンフィルタ用構成定義ソース
│             r_ctsu_median_sample.c   ・・・メディアンフィルタサンプルプログラムソース
│             r_ctsu_median_sample.h   ・・・メディアンフィルタサンプルプログラムヘッダ
├──ra2l1_filter_sample                 ・・・RA2L1 用サンプルプロジェクト
├──rx130_filter_sample                 ・・・RX130 用サンプルプロジェクト
└──rl78g16_filter_sample               ・・・RL78/G16 用サンプルプロジェクト
```

2.3.1　アプリケーション用データー覧

本サンプルプロジェクトのアプリケーションで使用している定数・グローバル変数について説明します。

表 2-5 サンプルアプリケーション用定数

定数名	設定値	内容
ファイル名：qe_touch_sample.c		
TOUCH_SCAN_CYCLE	((20 * 1000) / 100)	タッチ計測周期(20ms/100us)

表 2-6 サンプルアプリケーション用グローバル変数

変数名	型	説明
ファイル名：qe_touch_sample.c		
touch_cycle_count	uint16_t	タッチ計測周期カウンタ
touch_cycle_flag	volatile uint8_t	タッチ計測周期経過フラグ

2.3.2　アプリケーション API

表 2-7 に本サンプルプロジェクトで実装されているアプリケーション API の一覧を示します。

表 2-7 アプリケーション API 一覧

関数名	処理概要
ファイル名：qe_touch_sample.c	
qe_touch_main	メイン処理
timer0_callback	AGT 割り込みコールバック （RA2L1 グループ）
r_timer_callback	タイマ割り込みコールバック （RX130 グループ、RL78/G16 グループ）

2.4　サイズと実行時間

2.4.1　RA2L1 グループ

表 2-8 に本サンプルプログラム(タッチインターフェース構成が 5 つ、ボタン×1、シールド端子有り)の場合のフィルタ処理のデータサイズおよび各フィルタ演算 API の実行時間を示します。

表 2-8 フィルタ処理のデータサイズ及び増加分と実行時間(RA2L1)

条件	メモリ使用量[Bytes]			実行時間[us]
	text	data	bss	
フィルタ追加前	13648	16	3280	
FIR フィルタ	+396	+0	+24	6.268
IIR フィルタ	+252	+0	+16	2.468
単極 IIR フィルタ	+220	+0	+8	1.798
メディアンフィルタ	+224	+0	+16	2.216

2.4.2　RX130 グループ

表 2-9 に本サンプルプログラム(タッチインターフェース構成が 5 つ、ボタン×1)の場合のフィルタ処理のデータサイズおよび各フィルタ演算 API の実行時間を示します。

表 2-9 フィルタ処理のデータサイズ及び増加分と実行時間（RX130）

条件	メモリ使用量 [Bytes]			実行時間[cycle]
	RAMDATA	ROMDATA	PROGRAM	
フィルタ追加前	4750	2684	9984	
FIR フィルタ	+20	+18	+267	171
IIR フィルタ	+10	+12	+221	46
単極 IIR フィルタ	+4	+6	+179	27
メディアンフィルタ	+10	+0	+238	133

2.4.3　RL78/G16 グループ

表 2-10 に本サンプルプログラム(タッチインターフェース構成が 5 つ、ボタン×1)の場合のフィルタ処理のデータサイズおよび各フィルタ演算 API の実行時間を示します。

表 2-10 フィルタ処理のデータサイズ及び増加分と実行時間（RL78/G16）

条件	メモリ使用量 [Bytes]			実行時間[cycle]
	RAMDATA	ROMDATA	PROGRAM	
フィルタ追加前	1338	1318	12201	
FIR フィルタ	+20	+14	+461	861
IIR フィルタ	+10	+12	+486	578
単極 IIR フィルタ	+0	+6	+300	268
メディアンフィルタ	+10	+0	+361	286

3. FIR フィルタ

FIR(Finite Impulse Response)フィルタはランダムノイズおよび周期ノイズ低減用途として定常的に使用するフィルタです。

詳細については静電容量センサマイコン静電容量タッチノイズイミュニティガイド(R30AN0426)を参照してください。

3.1 動作説明

FIR フィルタは入力値をサンプリングし、サンプリングしたデータと定義した係数の積和演算結果を出力し、係数の定義内容によってローパスフィルタ・バンドパスフィルタ等の異なるフィルタ特性に対応することができます。

本サンプルプログラムでは移動平均フィルタとして扱います。

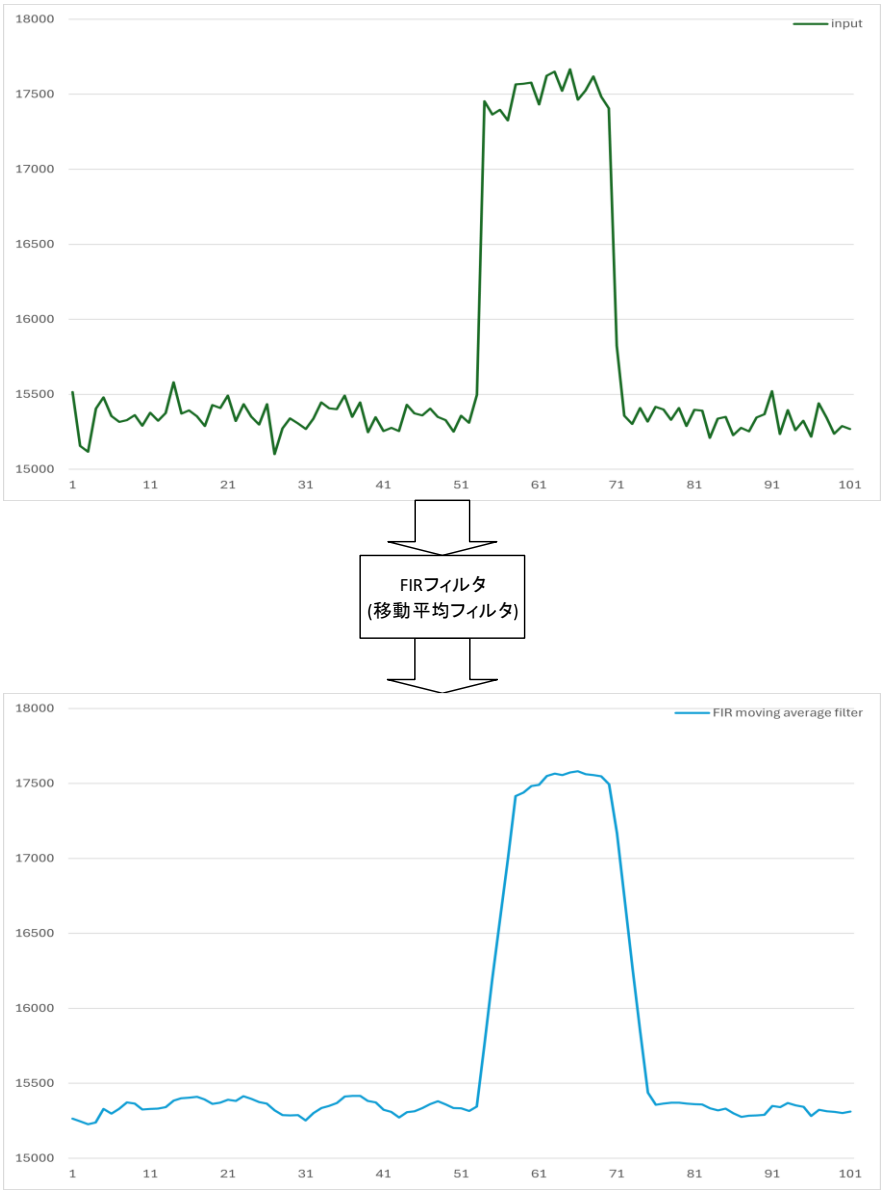


図 3-1 FIR フィルタ動作(移動平均フィルタ)

以下に FIR フィルタの演算式を、図 3-2 に FIR フィルタのブロック図を示します。

$$y(n) = \sum_{m=0}^{M-1} h(m) * x(n-m)$$

M はフィルタのタップ数、nはサンプルのインデックス、h(m)は係数、$x(n-\mathrm{m})$は m サンプル遅延の入力データ、$y(n)$は出力データを示します。

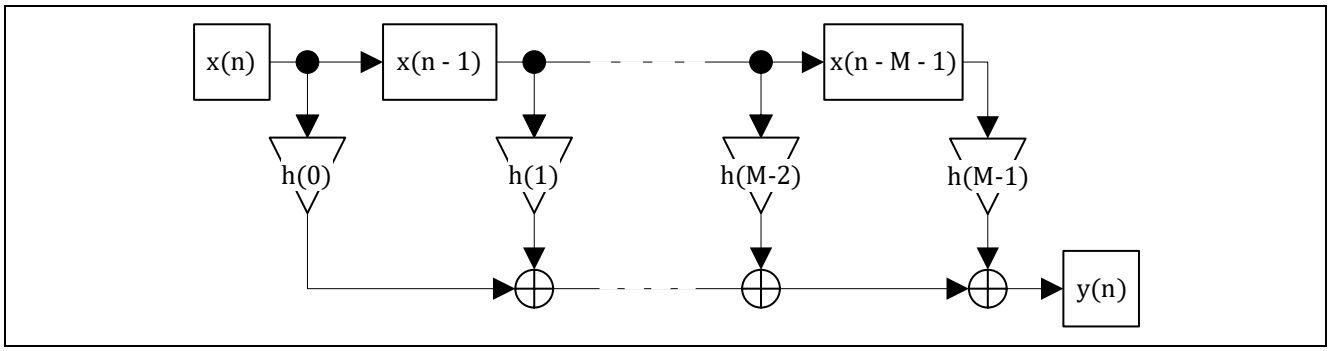


図 3-2 FIR フィルタブロック図

3.1.1　検出遅延について

FIR フィルタではサンプリングしたタッチ計測値からフィルタ結果を演算してノイズ信号の除去を行うため通常のタッチ検出に遅延が発生します。

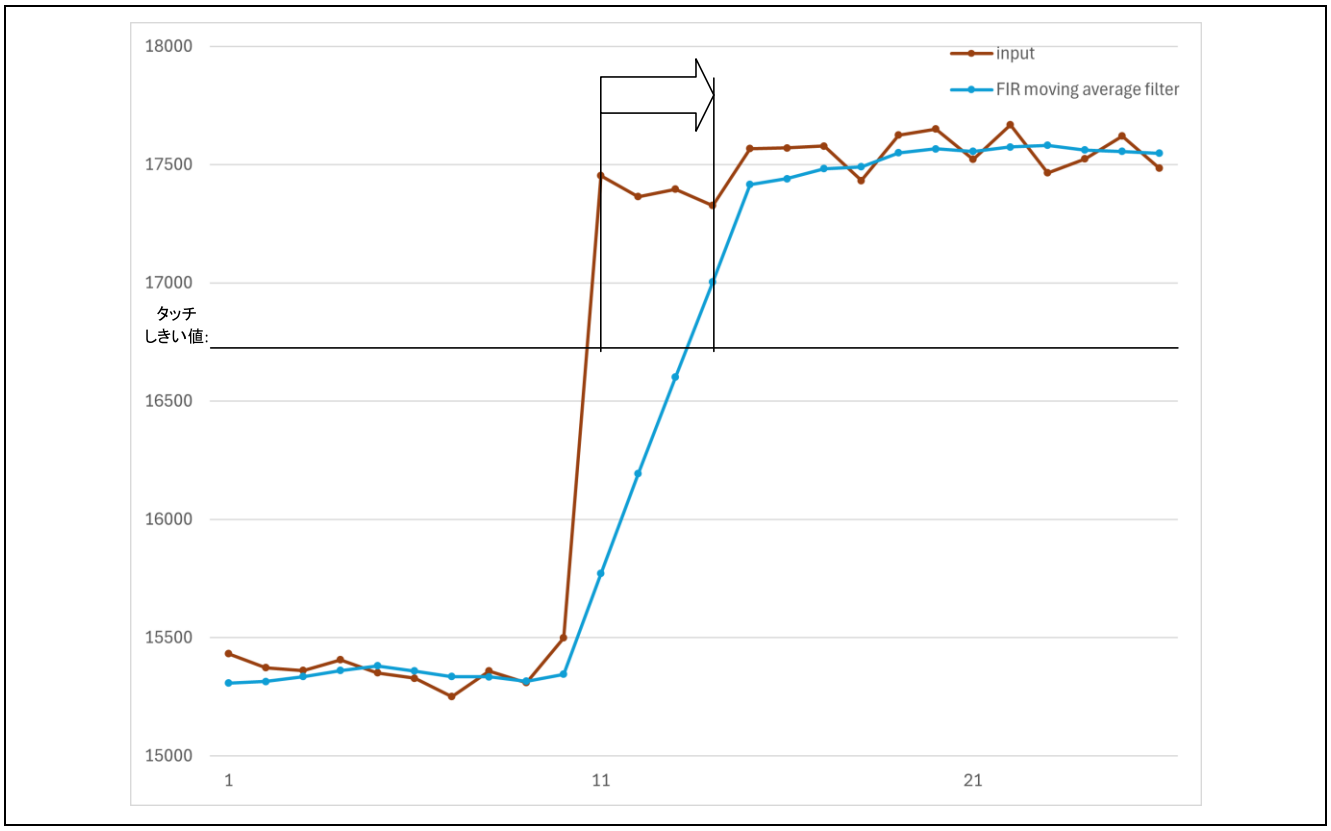


図 3-3 検出遅延(FIR 移動平均フィルタ)

3.1.2　フィルタ安定時間について

FIR フィルタは初期化後、サンプリングするデータ個数回分のフィルタ処理を実行するまでは入力データに対して低い演算結果が出力されます。

この期間中のフィルタ演算結果をタッチ検出に使用すると、タッチ検出の基準値が本来の基準値より低く設定されて常時タッチ ON 状態となることがあるため、この期間中はフィルタ演算結果を破棄（RM_TOUCH_DataGet()をコールせずに RM_TOUCH_ScanStart()をコールして次の計測を開始）して下さい。

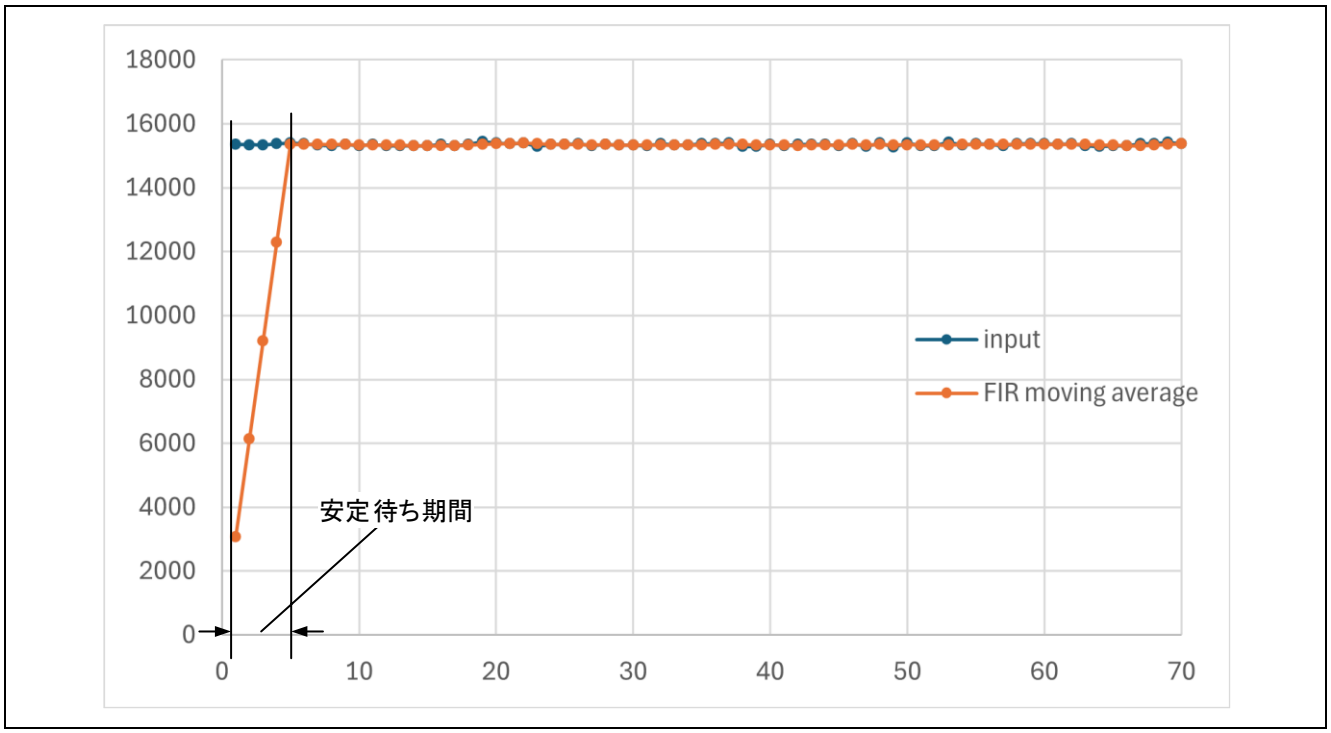


図 3-4 フィルタ安定待ち時間(FIR 移動平均フィルタ)

3.2　フィルタ仕様

表 3-1 に本サンプルプログラムの FIR フィルタの仕様を示します。

表 3-1 FIR フィルタ仕様

項目	仕様	備考
入力データ型	符号無し 16bit 整数型	
出力データ型	符号無し 16bit 整数型	
係数データ型	符号付き 16bit 整数	小数部 14bit の固定少数点
タップ数	1 < M ≦9	
フィルタ安定時間までの出力結果	安定時間中の演算結果と バッファ充填中応答を返す	フィルタ安定時間=タップ数

3.2.1　フィルタ係数についての注意事項

FIR フィルタの係数データは 1.0 以上、もしくは-1.0 以下の定義はできません。

FIR フィルタは係数データの合計が 2.0 以上、もしくは-2.0 以下となる定義はできません。

3.3　FIR フィルタ用データー覧

FIR フィルタ用に用意している定数・グローバル変数について説明します

3.3.1　定数

表 3-2 に定数の一覧を示します。

表 3-2 FIR フィルタ用定数

定数名	設定値	内容
ファイル名：r_ctsu_fir_sample.h		
FIR_TAP_SIZE_MAX	9	最大タップ数
FIR_DATA_SIZE	1	フィルタ対象のデータ数
ファイル名：r_ctsu_fir_sample.c		
FIR_TAP_SIZE_MIN	2	最小タップ数
FIR_CFG_DECIMAL_POINT	14	固定小数点桁数
MAX_FIR_COEFFICIENT_SUM	0x00008000	係数合計の最大値
MIN_FIR_COEFFICIENT_SUM	0xFFFF8000	係数合計の最小値
MAX_FIR_COEFFICIENT	0x4000	係数値の最大値
MIN_FIR_COEFFICIENT	0xC000	係数値の最小値
FIR_RESULT_MAX	0x0000FFFF	フィルタ結果最大値
FIR_RESULT_MIN	0x00000000	フィルタ結果最小値

3.3.2　構造体

FIR フィルタ API アクセス用の管理データと構成定義用構造体を示します。

3.3.2.1　FIR フィルタ構成定義(fir_config_t)

表 3-3　FIR フィルタ構成定義構造体(fir_config_t)

メンバ	データ型	内容
taps	uint16_t	タップ数
p_coefficient	int16_t *	FIR フィルタ係数配列へのポインタ フィルタ係数は $h_0..h_{M-1}$ 順に格納

3.3.2.2　FIR フィルタ管理データ(fir_ctrl_t)

表 3-4 FIR フィルタ管理データ構造体(fir_ctrl_t)

メンバ	データ型	内容
count	uint16_t	安定待ち時間カウンタ
fir_data	uint16_t [FIR_DATA_SIZE][FIR_TAP_SIZE_MAX]	FIR フィルタ用遅延バッファ

3.4　FIR フィルタ API

3.4.1　r_ctsu_fir_initial

この関数は、FIR フィルタ処理用管理データの初期化を行う関数です。

この関数は r_ctsu_fir_filter を使用する前に実行する必要があります。

Format

fsp_err_t r_ctsu_fir_initial(fir_ctrl_t * const p_ctrl , fir_config_t const * const p_cfg);

Parameters

p_ctrl

FIR フィルタ管理用データへのポインタ

p_cfg

FIR フィルタの構成定義へのポインタ

ReturnValues

FSP_SUCCESS	/* 正常終了 */
FSP_ERR_INVALID_ARGUMENT	/* 構成定義のパラメータが不正です */

Properties

r_ctsu_fir_sample.h にプロトタイプ宣言されています。

Description

この関数は FIR フィルタ処理用管理データの初期化を行います。

構成定義のパラメータが有効範囲外の場合はエラー応答を返します。

Example

```
err = r_ctsu_fir_initial(&g_ctsu_fir_control, &g_fir_cfg);
if (FSP_SUCCESS != err)
{
 while (true) {}
}
```

3.4.2　r_ctsu_fir_filter

この関数は FIR フィルタ演算を行います。

Format

fsp_err_t r_ctsu_fir_filter (fir_ctrl_t * const p_ctrl , fir_config_t const * const p_cfg , uint16_t *p_data);

Parameters

p_ctrl

FIR フィルタ管理用データへのポインタ

p_cfg

FIR フィルタの構成定義へのポインタ

p_data

FIR フィルタを適用する計測値データへのポインタ

ReturnValues

FSP_SUCCESS	/* 正常終了 */
FSP_ERR_BUFFER_EMPTY	/* バッファ充填中のため未適用のフィルタがあります */

Properties

r_ctsu_fir_sample.h にプロトタイプ宣言されています。

Description

この関数は FIR フィルタ処理を行います。

演算結果は符号なし 16bit 整数の範囲(65535～0)までとなり、それを超える場合は上限値もしくは下限値に丸められます。

初期化後、構成定義で指定したタップ数回の処理実行中はエラー応答を返します。

Example

```
/* FIR moving average filter sample */
err = R_CTSU_DataGet(g_qe_ctsu_instance_config01.p_ctrl, filter_buffer);
if(err == FSP_SUCCESS)
{
 err=r_ctsu_fir_filter(&g_ctsu_fir_control,&g_fir_cfg, filter_buffer);
 if(err == FSP_SUCCESS)
 {
    R_CTSU_DataInsert(g_qe_ctsu_instance_config01.p_ctrl, filter_buffer);
  }
}
```

Special Notes:

エラー応答時もフィルタ演算は実施されます。

複数のフィルタを縦続構成する場合はエラー応答時も後段のフィルタ処理は実行してください。

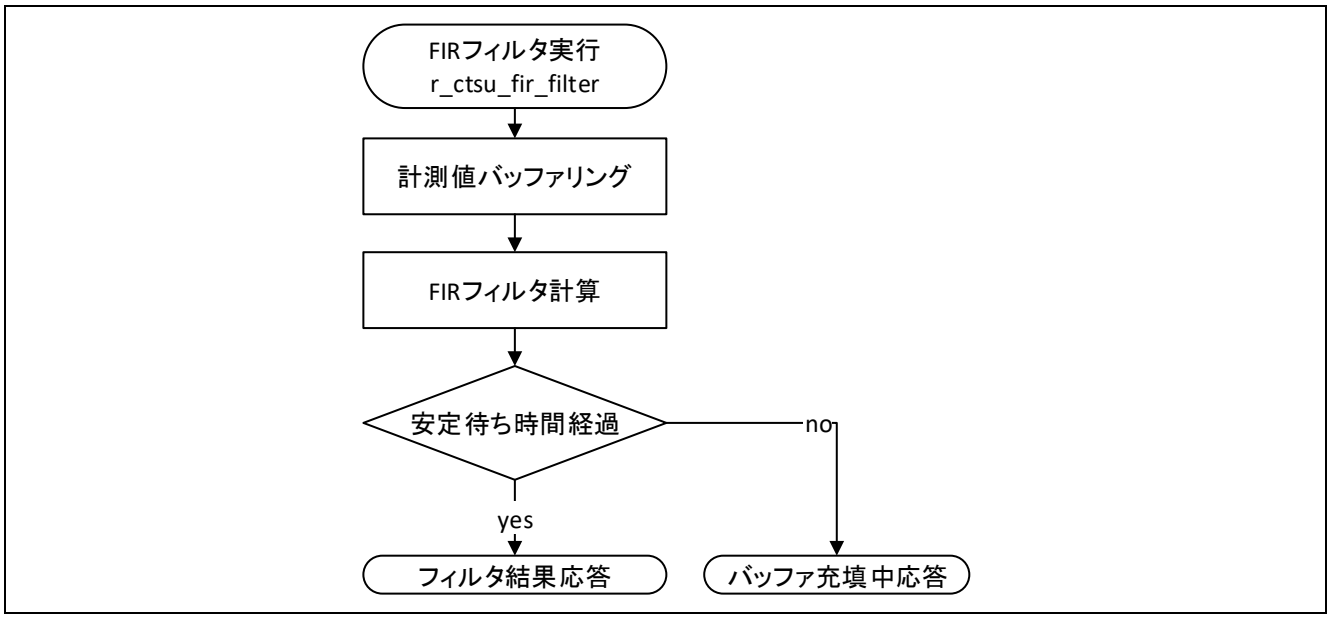


図 3-5 FIR フィルタ API フロー

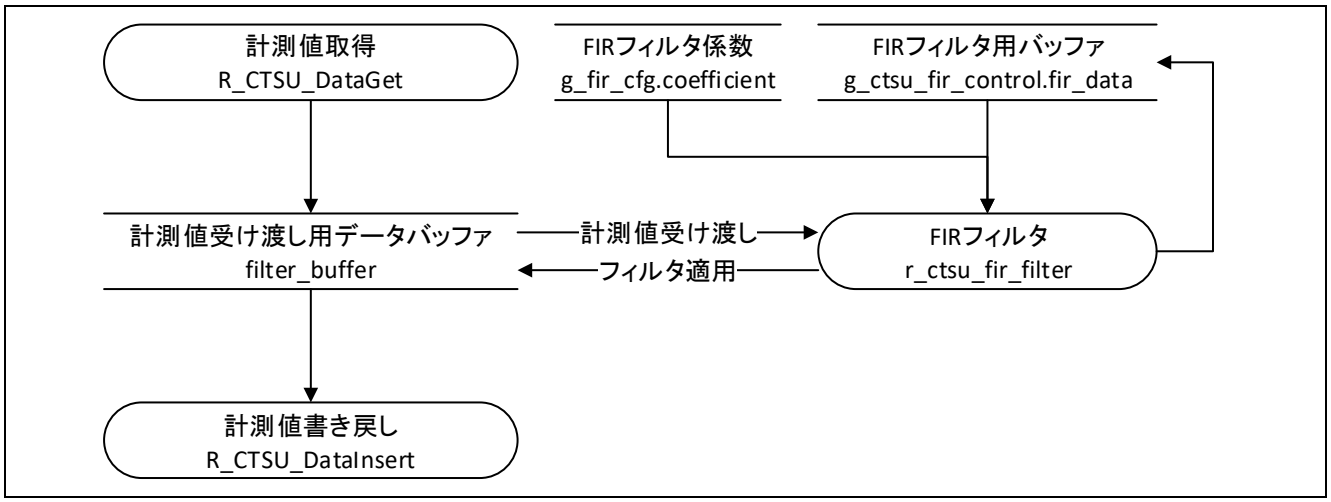


図 3-6 FIR フィルタデータ フロー

3.5　使用例

本サンプルプログラムでは FIR フィルタの使用例として FIR 移動平均フィルタを提供します。

3.5.1　フィルタ特性

表 3-5 にサンプル FIR フィルタの特性指定の定義を示します。

表 3-5 サンプル FIR フィルタ特性定義

	g_fir_cfg
	FIR 移動平均フィルタ
タップ数	5
係数	3276 (0.199951171875)
	3276 (0.199951171875)
	3276 (0.199951171875)
	3276 (0.199951171875)
	3276 (0.199951171875)

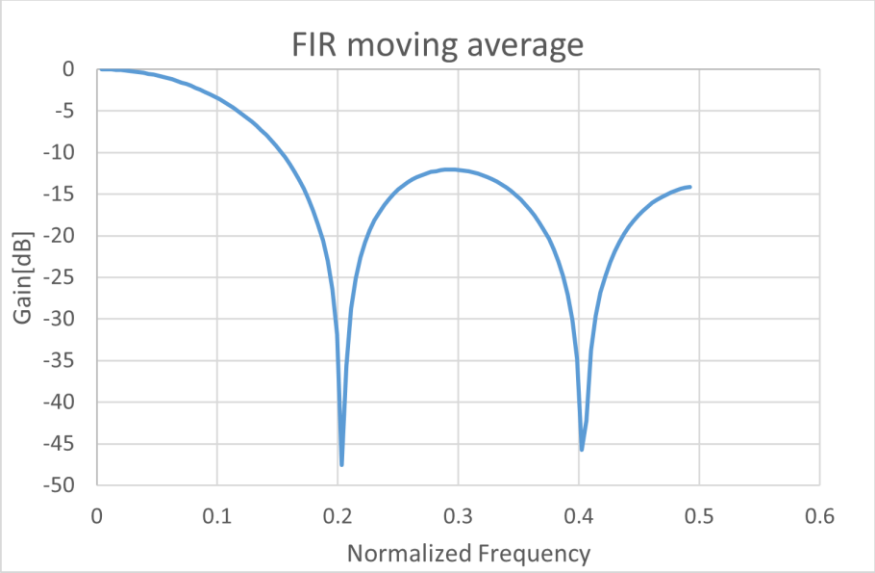


図 3-7 サンプル FIR フィルタ周波数特性

4. IIR フィルタ

IIR(Infinite Impulse Response)フィルタは省メモリ、小演算負荷で高周波成分低減用途として定常的に使用するフィルタです。

詳細については静電容量センサマイコン静電容量タッチノイズイミュニティガイド(R30AN0426)を参照してください。

4.1 動作説明

IIR フィルタは入力値と出力値をサンプリングし、サンプリングしたデータと定義した係数の積和演算結果を出力し、係数の定義内容によってローパスフィルタ・ハイパスフィルタ等の異なるフィルタ特性に対応することができます。

本サンプルプログラムではノッチフィルタとローパスフィルタとして扱います。

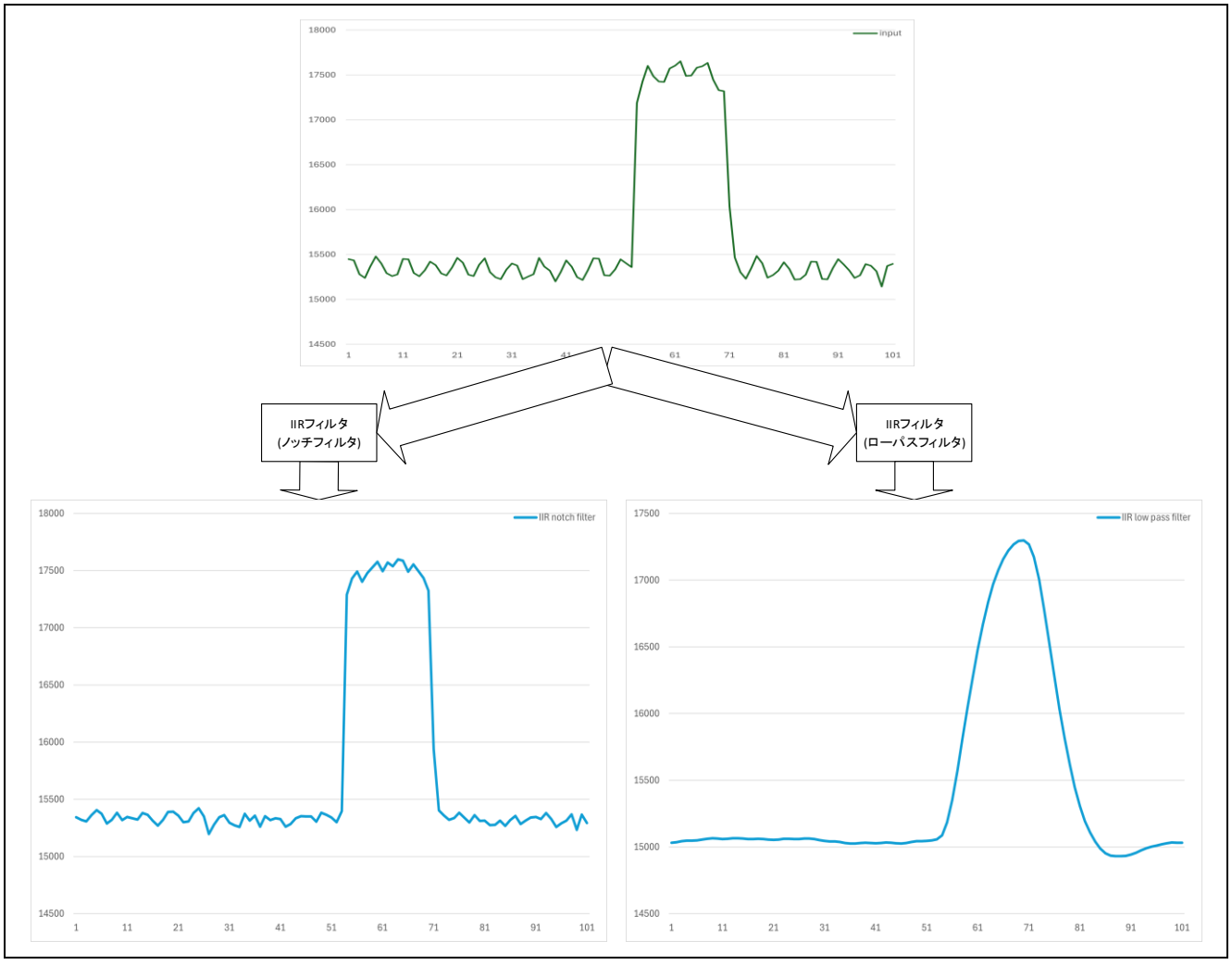


図 4-1 IIR フィルタ動作(ノッチフィルタ、ローパスフィルタ)

以下に IIR フィルタの演算式を、図 4-2 に IIR フィルタのブロック図を示します。

$$y(n) = \sum_{k=0}^{2} b(k) * x(n-k) + \sum_{m=1}^{2} a(m) * y(n-m)$$

nはサンプルのインデックス、a(m)、b(k)は係数、$x(n-k)$は k サンプル遅延の入力データ、$y(n-m)$は m サンプル遅延の出力データ、$y(n)$は出力データを示します。

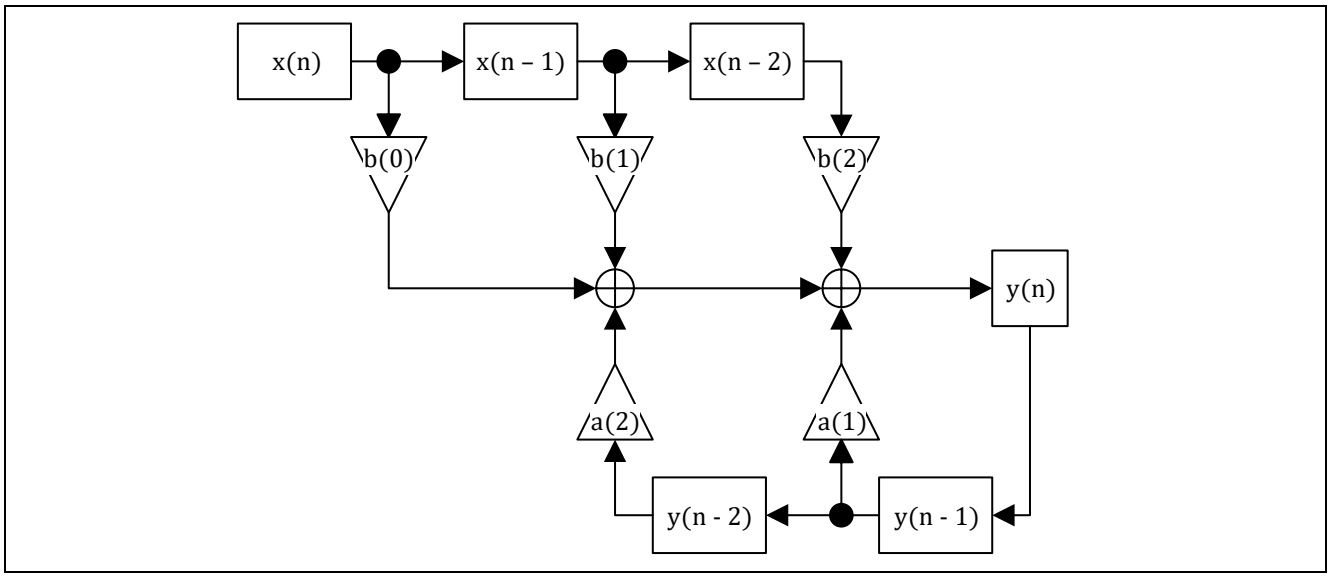


図 4-2 IIR フィルタブロック図

4.1.1　検出遅延について

IIR フィルタではフィルタ特性によっては通常のタッチ検出に遅延が発生する場合があります。

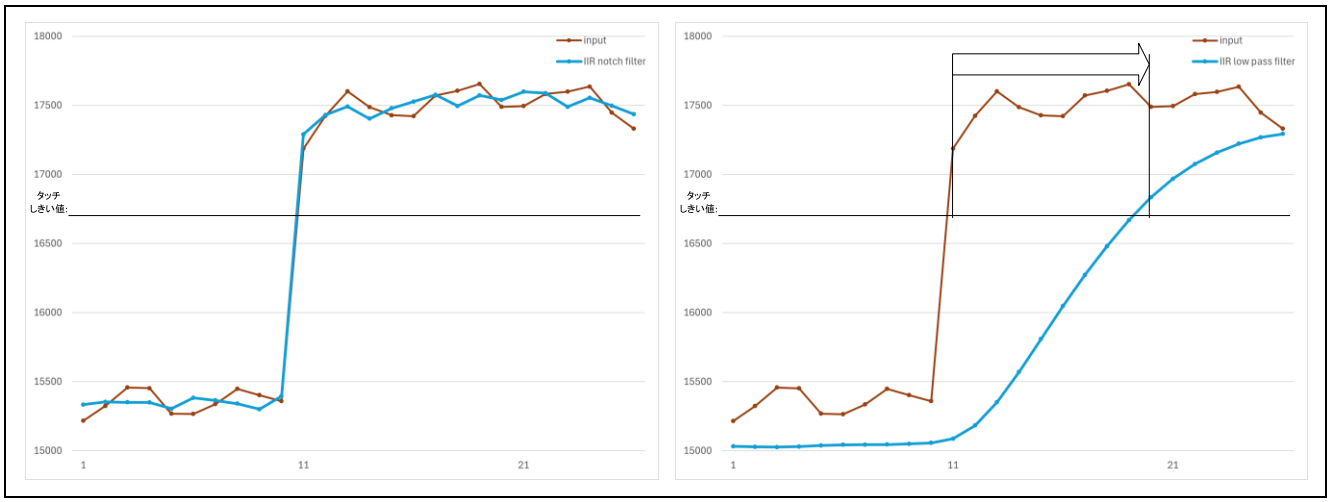


図 4-3 検出遅延(IIR ノッチフィルタ、ローパスフィルタ)

4.1.2　フィルタ安定待ち時間について

IIR フィルタは初期化後、一定回数のフィルタ処理を実行するまでは入力データから外れた演算結果が出力されます。

この期間中のフィルタ演算結果をタッチ検出に使用すると、タッチ検出の基準値が本来の基準値より低くなり常時タッチ ON が検出される状態となるため、この期間中はフィルタ演算結果を破棄して下さい。

この期間はフィルタ特性によって異なるため、十分に安定した演算結果が出力される時間を調査して安定待ち時間として指定し、期間中はフィルタ演算結果を破棄する必要があります。

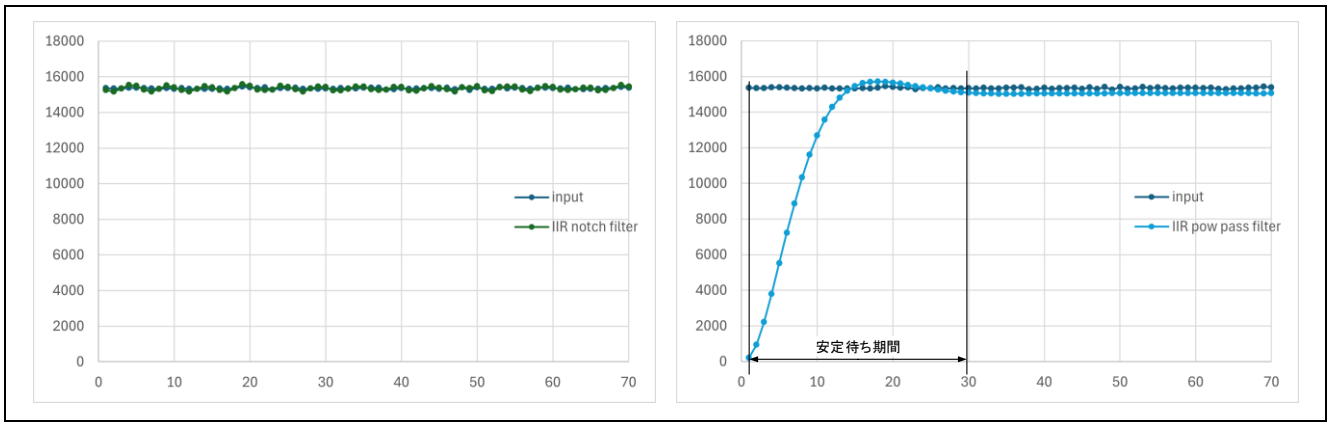


図 4-4 フィルタ安定待ち時間(IIR ノッチフィルタ、ローパスフィルタ)

4.2　フィルタ仕様

表 4-1 に本サンプルプログラムの IIR フィルタの仕様を示します。

表 4-1 IIR フィルタ仕様

項目	仕様	備考
入力データ型	符号無し 16bit 整数型	
出力データ型	符号無し 16bit 整数型	
係数データ型	符号付き 16bit 整数	小数部 11bit の固定少数点
次数	2	
フィルタ安定時間までの出力結果	安定時間中の演算結果と バッファ充填中応答を返す	フィルタ安定時間は構成定義で指定したサンプル数 （安定時間の指定範囲はタップ数～255）

【注】　係数：IIR フィルタを構成する定数乗算器に与える一連の定数。

4.2.1　フィルタ処理方法

本サンプルプログラムでは積和演算処理と遅延バッファのバッファリングを交互に行います。

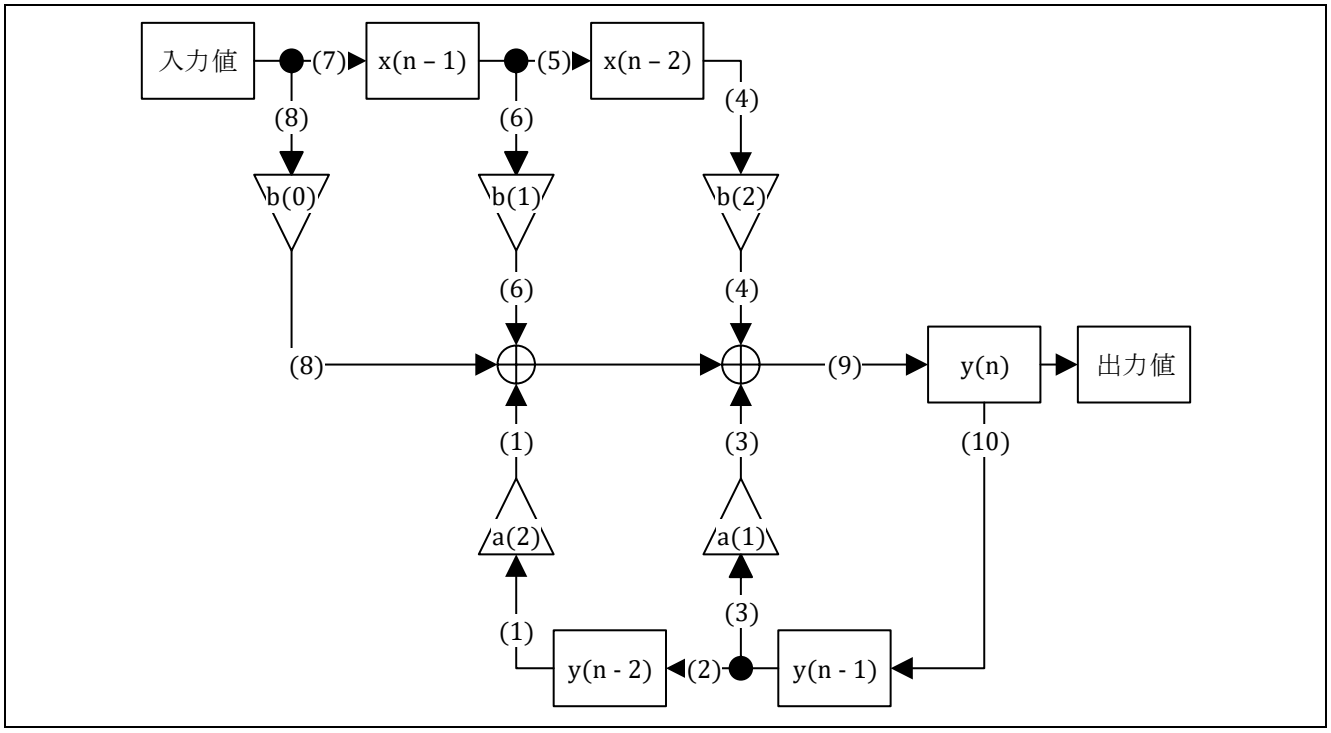


図 4-5 IIR フィルタ処理

4.2.2　フィルタ係数についての注意事項

IIR フィルタの係数データは 2.0 以上、もしくは-2.0 以下の定義はできません。

IIR フィルタの演算式は書籍、ツールによって式内の符号が異なる場合があります。

この場合は係数 a(1),a(2)の符号が逆になる事に注意してください。

式	$y(n) = \sum_{k=0}^{2} b(k) * x(n-k) + \sum_{m=1}^{2} a(m) * y(n-m)$	$y(n) = \sum_{k=0}^{2} b(k) * x(n-k) - \sum_{m=1}^{2} a(m) * y(n-m)$
b(0)	0.9931640625	0.9931640625
b(1)	-0.61376953125	-0.61376953125
b(2)	0.9931640625	0.9931640625
a(1)	0.61376953125	-0.61376953125
a(2)	-0.98681640625	0.98681640625

表 4-2 IIR フィルタ演算式と係数の符号について

4.3　IIR フィルタ用データ一覧

IIR フィルタ用に用意している定数・構造体・グローバル変数について説明します

4.3.1　定数

表 4-3 に定数の一覧を示します。

表 4-3 IIR フィルタ用定数

定数名	設定値	内容
ファイル名：r_ctsu_iir_sample.h		
IIR_COEF_SIZE	5	
IIR_BUFFER_SIZE	4	
IIR_DATA_SIZE	1	フィルタ対象のデータ数
ファイル名：r_ctsu_iir_sample.c		
IIR_SETTLINGS_MAX	255	最大安定待ち時間
IIR_SETTLINGS_MIN	3	最小安定待ち時間
IIR_CFG_DECIMAL_POINT	11	固定小数点桁数
MAX_IIR_COEFFICIENT	0x1000	係数定義最大値
MIN_IIR_COEFFICIENT	0xF000	係数定義最小値
IIR_RESULT_MAX	0x0000FFFF	フィルタ結果最大値
IIR_RESULT_MIN	0x00000000	フィルタ結果最小値

4.3.2　構造体

IIR フィルタ API アクセス用の管理データと構成定義用構造体を示します。

4.3.2.1　IIR フィルタ構成定義(iir_config_t)

表 4-4 IIR フィルタ構成定義構造体(iir_config_t)

メンバ	データ型	内容
settlings	uint16_t	安定待ち時間
coefficient	int16_t [IIR_COEF_SIZE]	IIR フィルタ係数 係数格納順序は b0,b1,b2,a1,a2

4.3.2.2　IIR フィルタ管理データ(iir_ctrl_t)

表 4-5 IIR フィルタ管理データ構造体(iir_ctrl_t)

メンバ	データ型	内容
count	uint16_t	安定待ち時間カウンタ
iir_data	uint16_t [IIR_DATA_SIZE][IIR_BUFFER_SIZE]	IIR フィルタ用遅延バッファ

4.4　IIR フィルタ API

4.4.1　r_ctsu_iir_initial

この関数は、IIR フィルタ処理用管理データの初期化を行う関数です。

この関数は r_ctsu_iir_filter を使用する前に実行する必要があります。

Format

fsp_err_t r_ctsu_iir_initial(iir_ctrl_t * p_ctrl , iir_config_t const * const p_cfg);

Parameters

p_ctrl

IIR フィルタ管理用データへのポインタ

p_cfg

IIR フィルタの構成定義へのポインタ

ReturnValues

FSP_SUCCESS	/* 正常終了 */
FSP_ERR_INVALID_ARGUMENT	/* 構成定義のパラメータが不正です */

Properties

r_ctsu_iir_sample.h にプロトタイプ宣言されています。

Description

この関数は IIR フィルタ処理用管理データの初期化を行います。

構成定義のパラメータが有効範囲外の場合はエラー応答を返します。

Example

```
err = r_ctsu_iir_initial(&g_ctsu_iir_control1, &g_iir_cfg);
if (FSP_SUCCESS != err)
{
 while (true) {}
}
```

4.4.2　r_ctsu_iir_filter

この関数は IIR フィルタ演算を行います。

Format

fsp_err_t r_ctsu_iir_filter(iir_ctrl_t * const p_ctrl , iir_config_t const * const p_cfg , uint16_t *p_data);

Parameters

p_ctrl

IIR フィルタ管理用データへのポインタ

p_cfg

IIR フィルタの構成定義へのポインタ

p_data

IIR フィルタを適用する計測値データへのポインタ

ReturnValues

FSP_SUCCESS	/* 正常終了 */
FSP_ERR_BUFFER_EMPTY	/* バッファ充填中のため未適用のフィルタがあります */

Properties

r_ctsu_iir_sample.h にプロトタイプ宣言されています。

Description

この関数は IIR フィルタ処理を行います。

演算結果は符号無し 16bit 整数の範囲(65535～0)までとなり、それを超える場合は上限値もしくは下限値に丸められます。

初期化後、構成定義で指定した安定時間回の処理実行中はエラー応答を返します。

Example

```
/* IIR notch filter sample */
err = R_CTSU_DataGet(g_qe_ctsu_instance_config02.p_ctrl, filter_buffer);
if(err == FSP_SUCCESS)
{
 err=r_ctsu_iir_filter(&g_ctsu_iir_control1, &g_iir_cfg1, filter_buffer);
 if(err == FSP_SUCCESS)
 {
    R_CTSU_DataInsert(g_qe_ctsu_instance_config02.p_ctrl, filter_buffer);
 }
}
```

Special Notes:

エラー応答時もフィルタ演算は実施されます。

複数のフィルタを縦続構成する場合はエラー応答時も後段のフィルタ処理は実行してください。

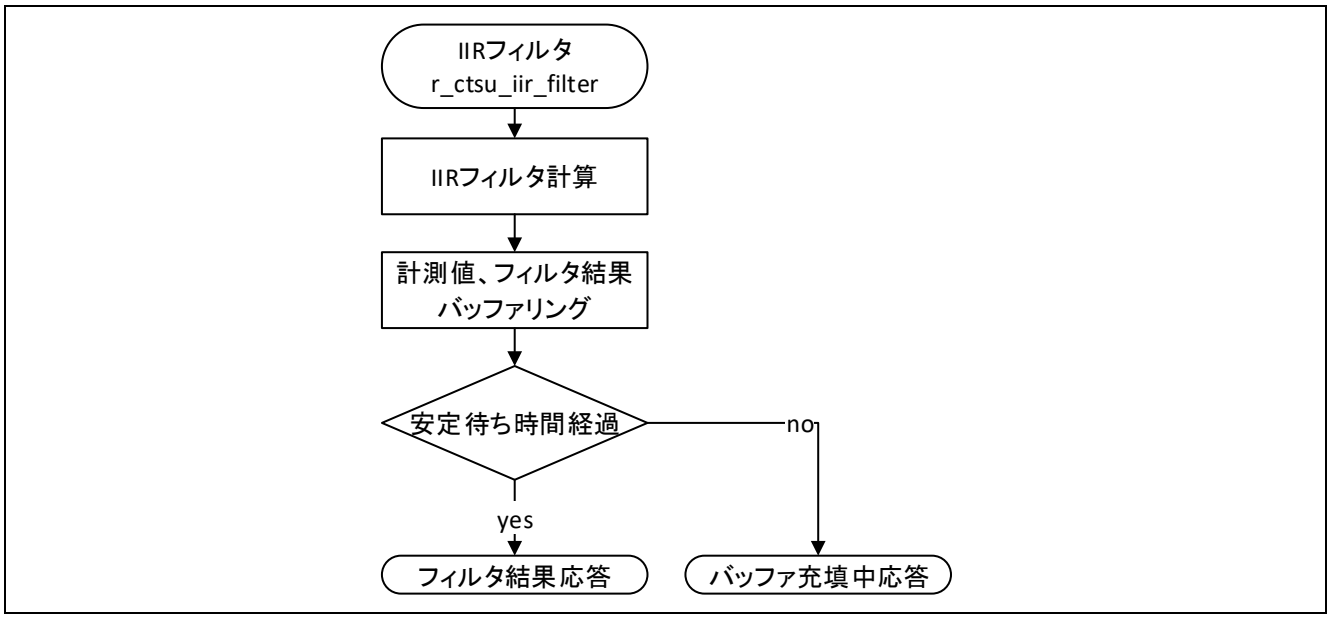


図 4-6 IIR フィルタ実行 API フロー

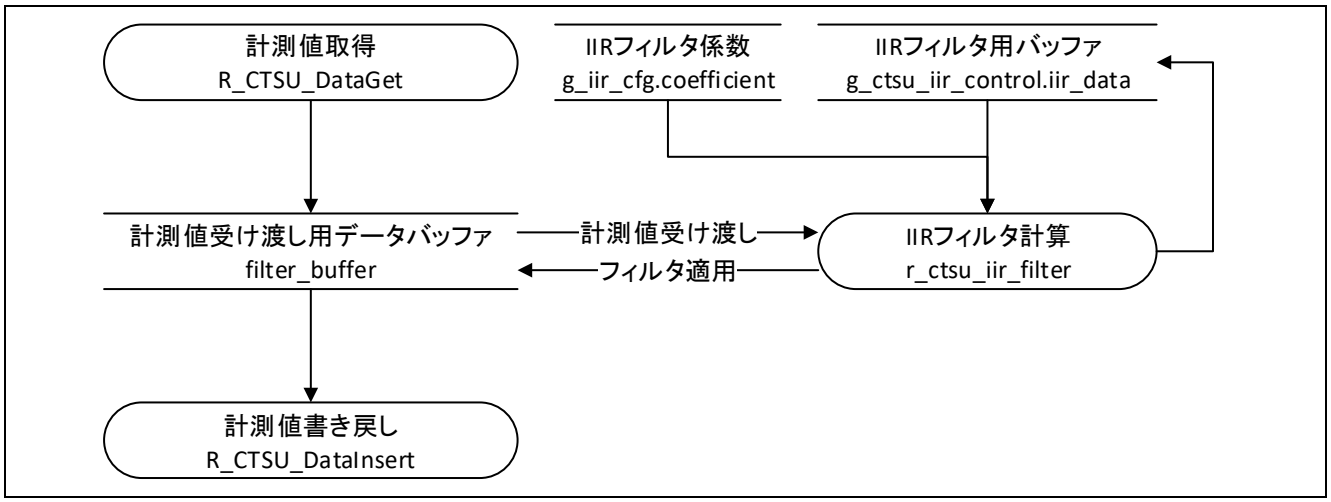


図 4-7 IIR フィルタデータフロー

4.5　使用例

本サンプルプログラムでは使用例としてノッチフィルタとローパスフィルタを提供します。

4.5.1　フィルタ特性

表 4-6 に係数定義およびフィルタ特性を示します。

表 4-6 サンプル IIR フィルタ特性定義

	g_iir_cfg1	g_iir_cfg2
	IIR ノッチフィルタ	IIR ローパスフィルタ
係数 B	2034 (0.9931640625)	27 (0.01318359375)
	-1257 (-0.61376953125)	54 (0.0263671875)
	2034 (0.9931640625)	27 (0.01318359375)
係数 A	1257 (0.61376953125)	3373 (1.64697265625)
	-2021 (-0.98681640625)	-1435 (-0.70068359375)

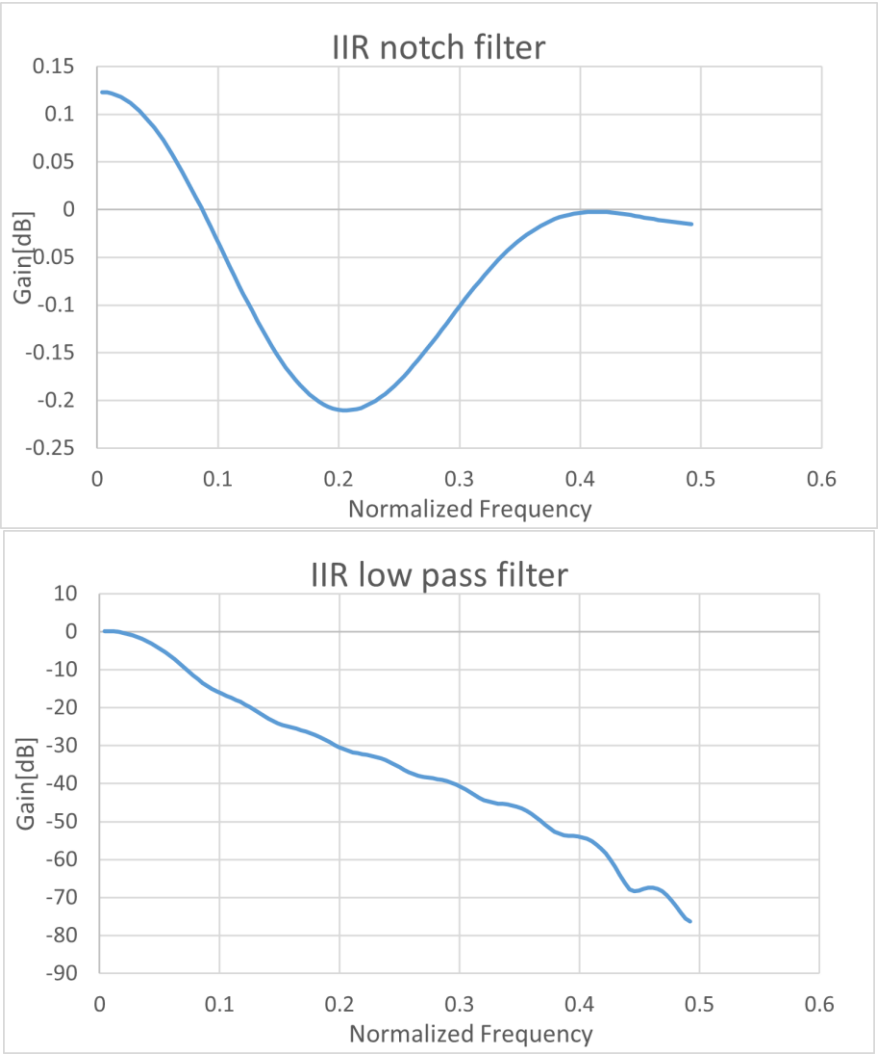


図 4-8 サンプル IIR フィルタ周波数特性

5.　単極 IIR フィルタ

単極 IIR(Single Pole Infinite Impulse Response)フィルタは最小構成の IIR フィルタです。

5.1　動作説明

単極 IIR フィルタは入力値と前回の出力値と定義した係数の積和演算結果を出力し、係数の定義内容によってローパスフィルタ・ハイパスフィルタ等の異なるフィルタ特性に対応することができます。

本サンプルプログラムではローパスフィルタとして扱います。

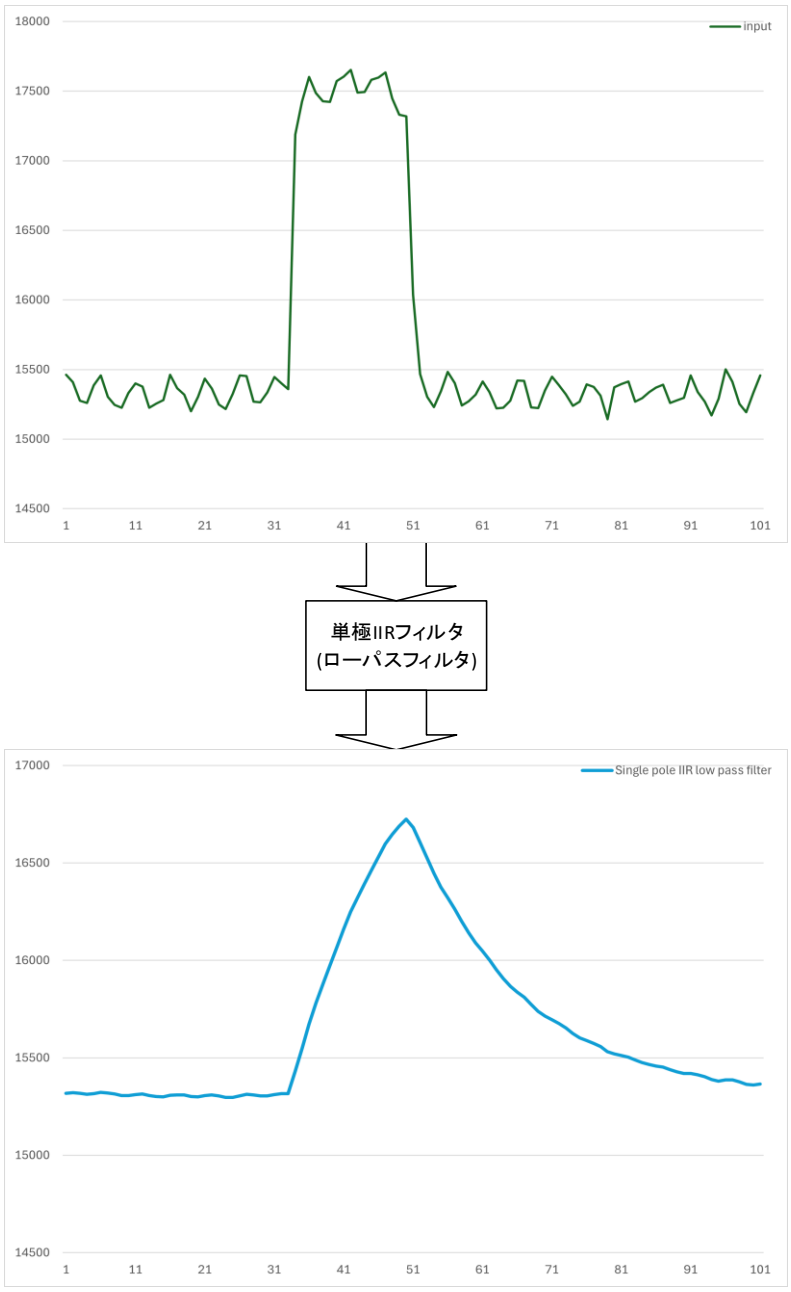


図 5-1 単極 IIR フィルタ動作(ローパスフィルタ)

以下に単極 IIR フィルタの演算式を、図 5-2 に単極 IIR フィルタのブロック図を示します。

$$y(n) = b * x(n) + a * y(n-1)$$

nはサンプルのインデックス、a、b は係数、$x(n)$は入力データ、$y(n-1)$は 1 サンプル遅延の出力データ、$y(n)$は出力データを示します。

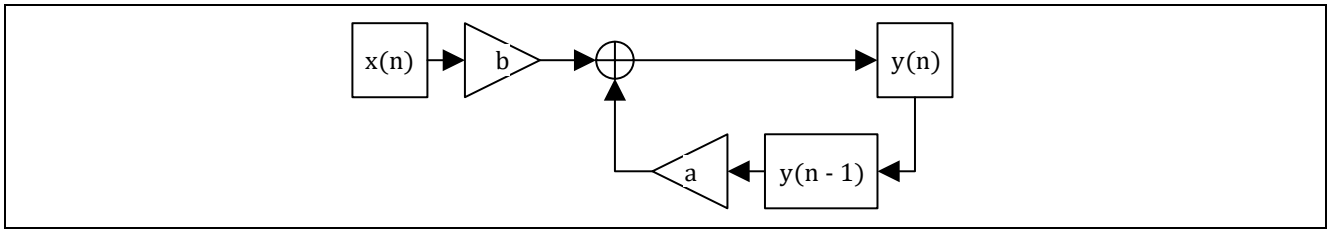


図 5-2 単極 IIR フィルタブロック図

5.1.1　検出遅延について

単極 IIR フィルタでは通常のタッチ検出に遅延が発生します。

遅延する時間はフィルタ特性によって異なります。

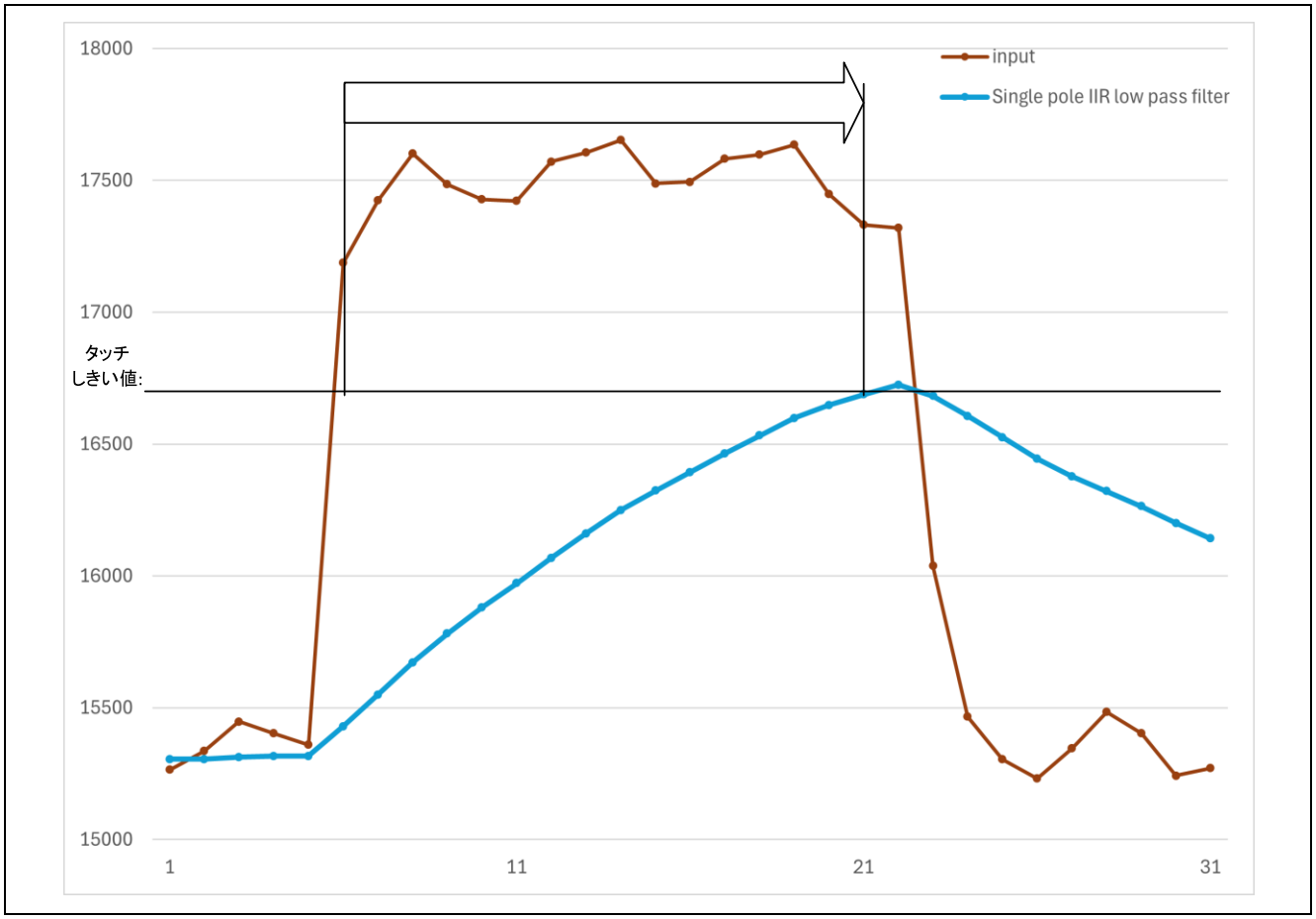


図 5-3 検出遅延(単極 IIR ローパスフィルタ)

5.1.2　フィルタ安定時間について

単極 IIR フィルタは初期化後、一定回数のフィルタ処理を実行するまでは入力データから外れた演算結果が出力されます。

この期間中のフィルタ演算結果をタッチ検出に使用すると、タッチ検出の基準値が本来の基準値より低くなり常時タッチ ON が検出される状態となるため、この期間中はフィルタ演算結果を破棄して下さい。

この期間はフィルタ特性によって異なるため、十分に安定した演算結果が出力される時間を調査して安定待ち時間として指定してください。

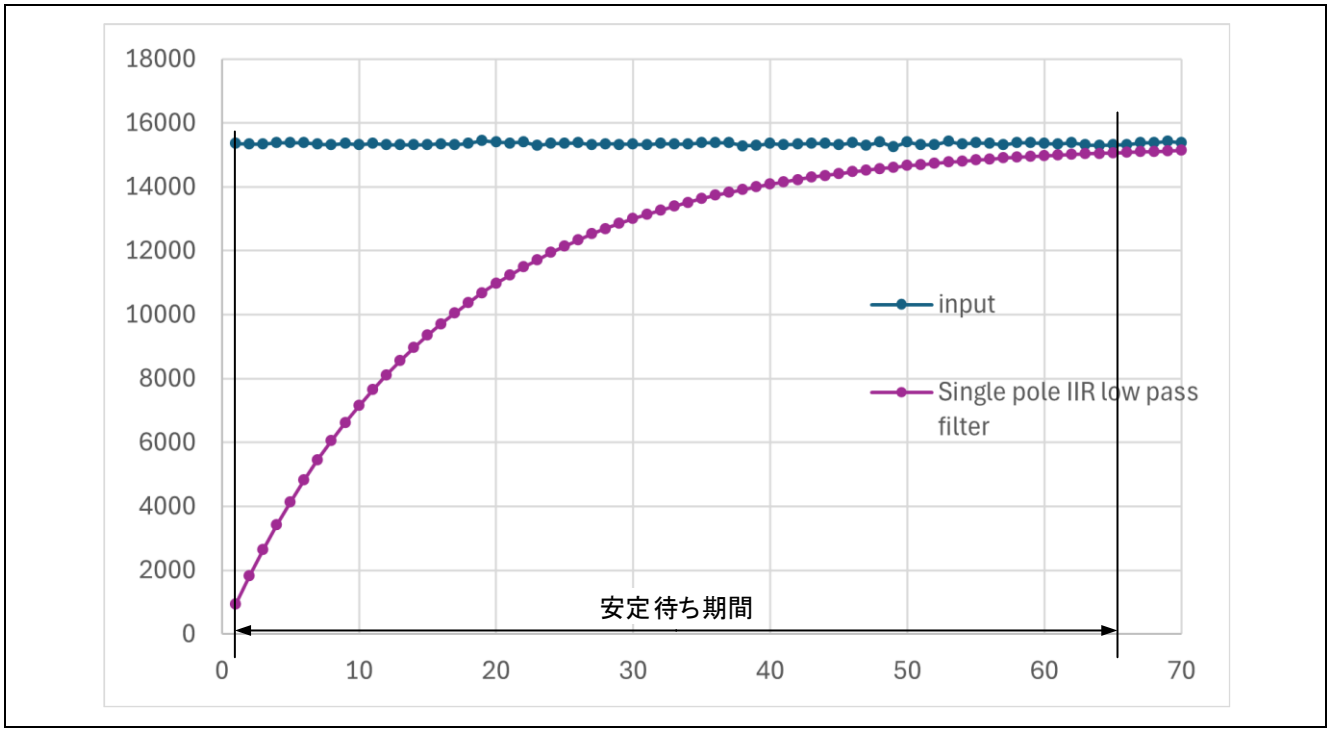


図 5-4 フィルタ安定待ち時間(単極 IIR ローパスフィルタ)

5.2　フィルタ仕様

表 5-1 に本サンプルプログラムの単極 IIR フィルタの仕様を示します。

表 5-1 単極 IIR フィルタ仕様

項目	仕様	備考
入力データ型	符号無し 16bit 整数型	
出力データ型	符号無し 16bit 整数型	
係数データ型	符号付き 16bit 整数	小数部 14bit の固定少数点
フィルタ安定時間までの出力結果	安定時間中の演算結果と バッファ充填中応答を返す	フィルタ安定時間は構成定義で指定したサンプル数 （安定時間の指定範囲は 1～255）

【注】　係数：単極 IIR フィルタを構成する定数乗算器に与える一連の定数。

5.2.1　フィルタ処理方法

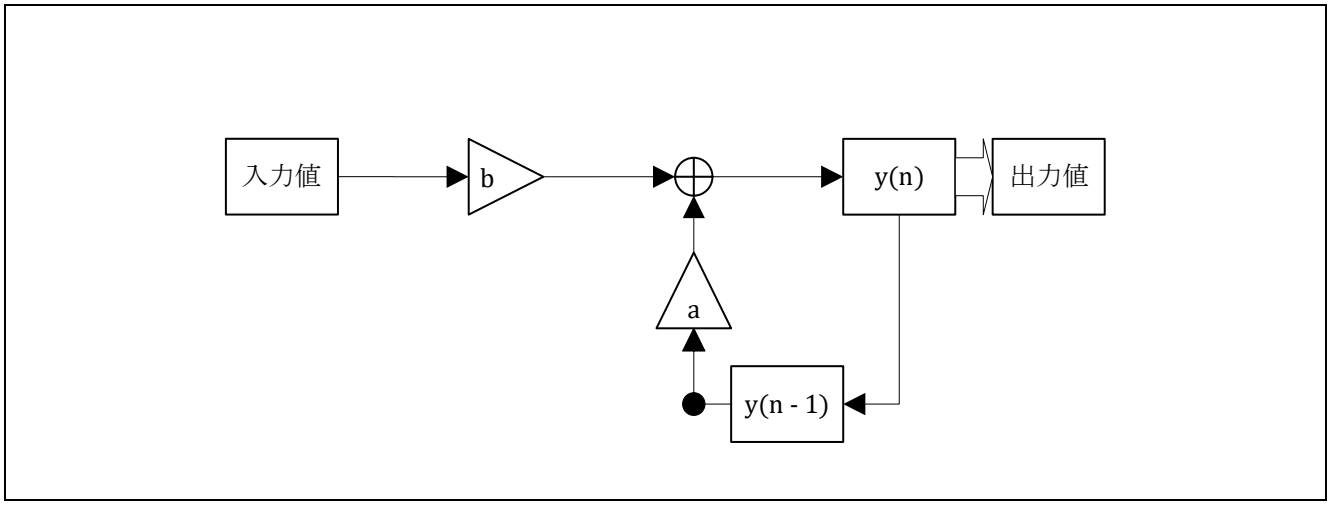


図 5-5 単極 IIR フィルタ処理

5.2.2　フィルタ係数についての注意事項

単極 IIR フィルタの係数データは 1.0 以上、もしくは-1.0 以下の定義はできません。

5.3　単極 IIR フィルタ用データ一覧

単極 IIR フィルタ用に用意している定数・構造体・グローバル変数について説明します

5.3.1　定数

表 5-2 に定数の一覧を示します。

表 5-2 単極 IIR フィルタ用定数

定数名	設定値	内容
ファイル名：r_ctsu_spiir_sample.h		
SPIIR_COEF_SIZE	2	係数用配列サイズ
SPIIR_DATA_SIZE	1	フィルタ対象のデータ数
ファイル名：r_ctsu_spiir_sample.c		
SPIIR_SETTLINGS_MAX	255	最大安定待ち時間
SPIIR_SETTLINGS_MIN	1	最小安定待ち時間
SPIIR_CFG_DECIMAL_POINT	14	固定小数点桁数
SPIIR_COEF_MAX	0x4000	係数定義最大値
SPIIR_COEF_MIN	0xC000	係数定義最小値
SPIIR_RESULT_MAX	0x0000FFFF	フィルタ結果最大値
SPIIR_RESULT_MIN	0x00000000	フィルタ結果最小値

5.3.2　構造体

単極 IIR フィルタ API アクセス用の管理データと構成定義用構造体を示します。

5.3.2.1　単極 IIR フィルタ構成定義(spiir_config_t)

表 5-3 単極 IIR フィルタ構成定義構造体(spiir_config_t)

メンバ	データ型	内容
settlings	uint16_t	安定待ち時間
coefficient	int16_t [SPIIR_COEF_SIZE]	単極 IIR フィルタ係数

5.3.2.2　単極 IIR フィルタ管理データ(spiir_ctrl_t)

表 5-4 単極 IIR フィルタ管理データ構造体(spiir_ctrl_t)

メンバ	データ型	内容
count	uint16_t	安定待ち時間カウンタ
spiir_data	uint16_t [SPIIR_DATA_SIZE]	単極 IIR フィルタ用遅延バッファ

5.4　単極 IIR フィルタ API

5.4.1　r_ctsu_spiir_initial

この関数は、単極 IIR フィルタ処理用管理データの初期化を行う関数です。

この関数は r_ctsu_spiir_filter を使用する前に実行する必要があります。

Format

fsp_err_t r_ctsu_spiir_initial(spiir_ctrl_t * p_ctrl , spiir_config_t const * const p_cfg);

Parameters

p_ctrl

単極 IIR フィルタ管理用データへのポインタ

p_cfg

単極 IIR フィルタの構成定義へのポインタ

ReturnValues

FSP_SUCCESS	/* 正常終了 */
FSP_ERR_INVALID_ARGUMENT	/* 構成定義のパラメータが不正です */

Properties

r_ctsu_spiir_sample.h にプロトタイプ宣言されています。

Description

この関数は単極 IIR フィルタ処理用管理データの初期化を行います。

構成定義のパラメータが有効範囲外の場合はエラー応答を返します。

Example

```
err = r_ctsu_spiir_initial(&g_ctsu_spiir_control1, &g_spiir_cfg);
if (FSP_SUCCESS != err)
{
 while (true) {}
}
```

5.4.2　r_ctsu_spiir_filter

この関数は単極 IIR フィルタ演算を行います。

Format

fsp_err_t r_ctsu_spiir_filter(spiir_ctrl_t * const p_ctrl, spiir_config_t const * const p_cfg,uint16_t *p_data);

Parameters

p_ctrl

単極 IIR フィルタ管理用データへのポインタ

p_cfg

単極 IIR フィルタの構成定義へのポインタ

p_data

単極 IIR フィルタを適用する計測値データへのポインタ

ReturnValues

FSP_SUCCESS	/* 正常終了 */
FSP_ERR_BUFFER_EMPTY	/* バッファ充填中のため未適用のフィルタがあります */

Properties

r_ctsu_spiir_sample.h にプロトタイプ宣言されています。

Description

この関数は単極 IIR フィルタ処理を行います。

演算結果は符号無し 16bit 整数の範囲(65535～0)までとなり、それを超える場合は上限値もしくは下限値に丸められます。

初期化後、構成定義で指定した安定時間回の処理実行中はエラー応答を返します。

Example

```
/* Single Pole IIR low pass filter sample */
err = R_CTSU_DataGet(g_qe_ctsu_instance_config04.p_ctrl, filter_buffer);
if(err == FSP_SUCCESS)
{
  err=r_ctsu_spiir_filter(&g_ctsu_spiir_control1, &g_spiir_cfg1,
filter_buffer);
  if(err == FSP_SUCCESS)
  {
     R_CTSU_DataInsert(g_qe_ctsu_instance_config04.p_ctrl, filter_buffer);
  }
}
```

Special Notes:

エラー応答時もフィルタ演算は実施されます。

複数のフィルタを縦続構成する場合はエラー応答時も後段のフィルタ処理は実行してください。

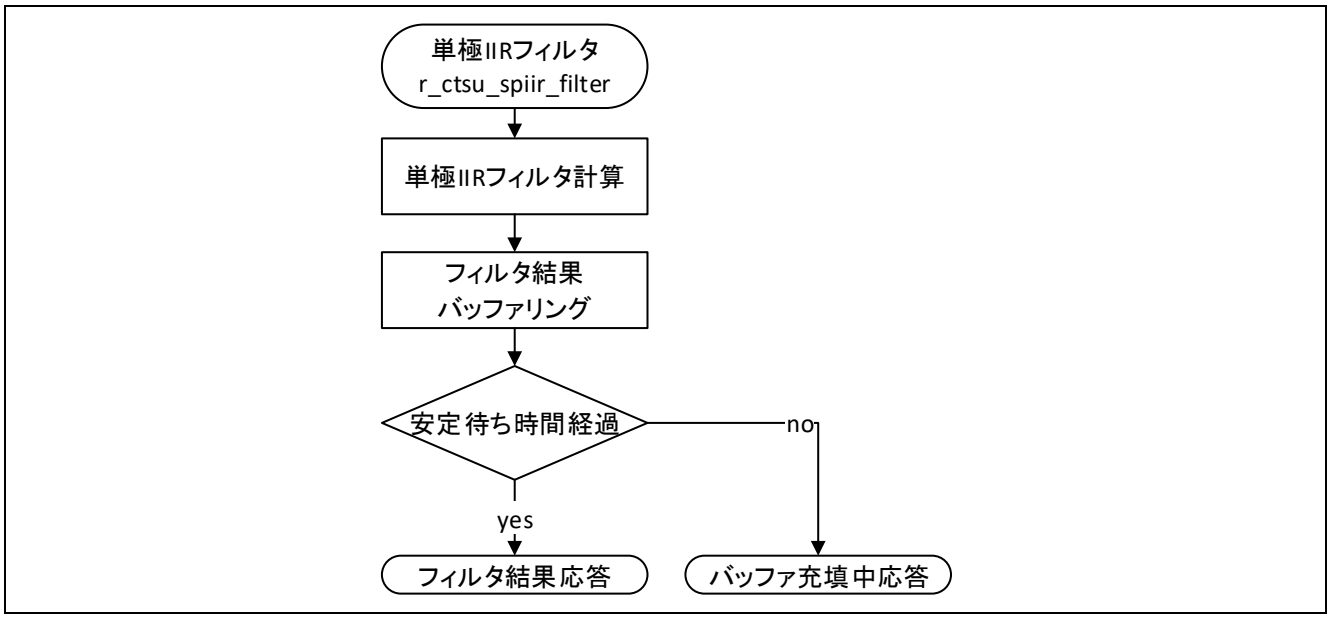


図 5-6 単極 IIR フィルタ実行 API フロー

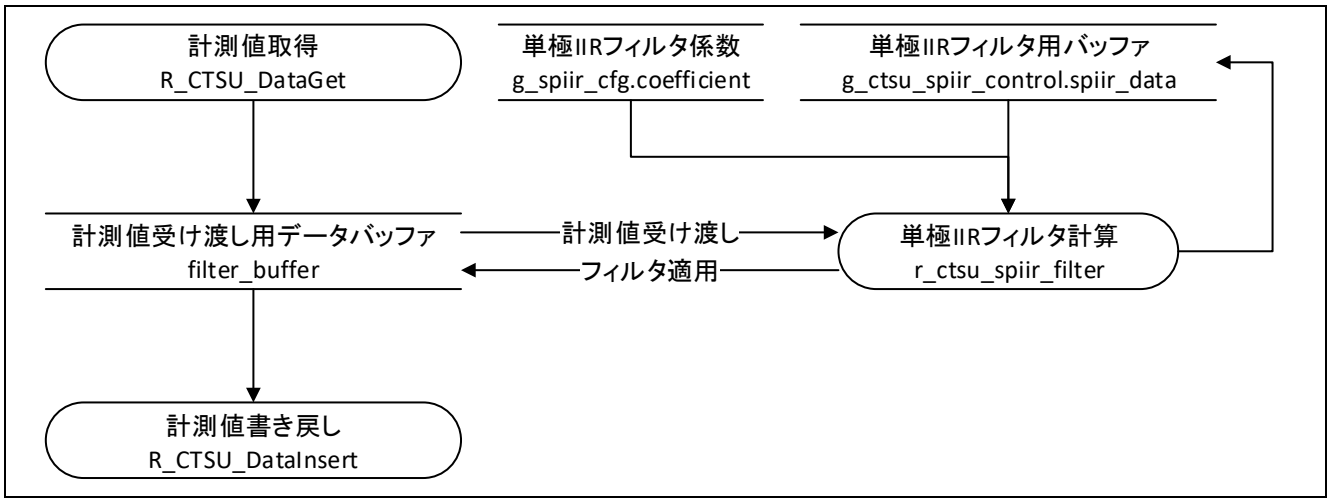


図 5-7 単極 IIR フィルタデータフロー

5.5　使用例

本サンプルプログラムでは使用例としてローパスフィルタを提供します。

5.5.1　フィルタ特性

表 5-5 に係数定義およびフィルタ特性を示します。

表 5-5 サンプル単極 IIR フィルタ特性定義

	g_spiir_cfg
	単極 IIR ローパスフィルタ
係数 A	15386 (0.9390869140625)
係数 B	997 (0.06085205078125)

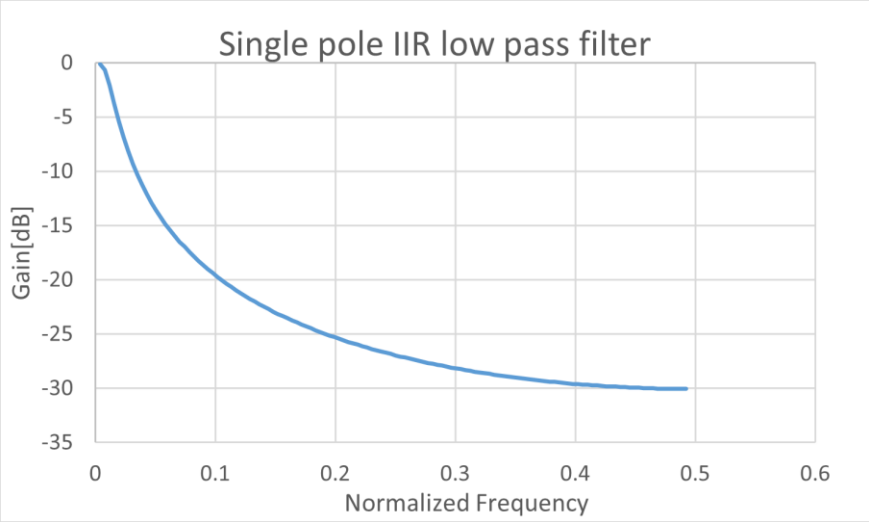


図 5-8 サンプル単極 IIR フィルタ周波数特性

6.　メディアンフィルタ

メディアンフィルタはパルス性ノイズの除去に使用できるフィルタです。ランダムノイズや低周期ノイズに対しては効果が限定的となるため、そのような場合は FIR や IIR フィルタと組み合わせて使用することもできます。

6.1　動作説明

メディアンフィルタは入力値をサンプリングし、サンプリングしたデータの中央値を出力値として採用します。

パルス性ノイズは中央値から大きく外れたデータとして入力されるため、メディアンフィルタの動作により除去されます。

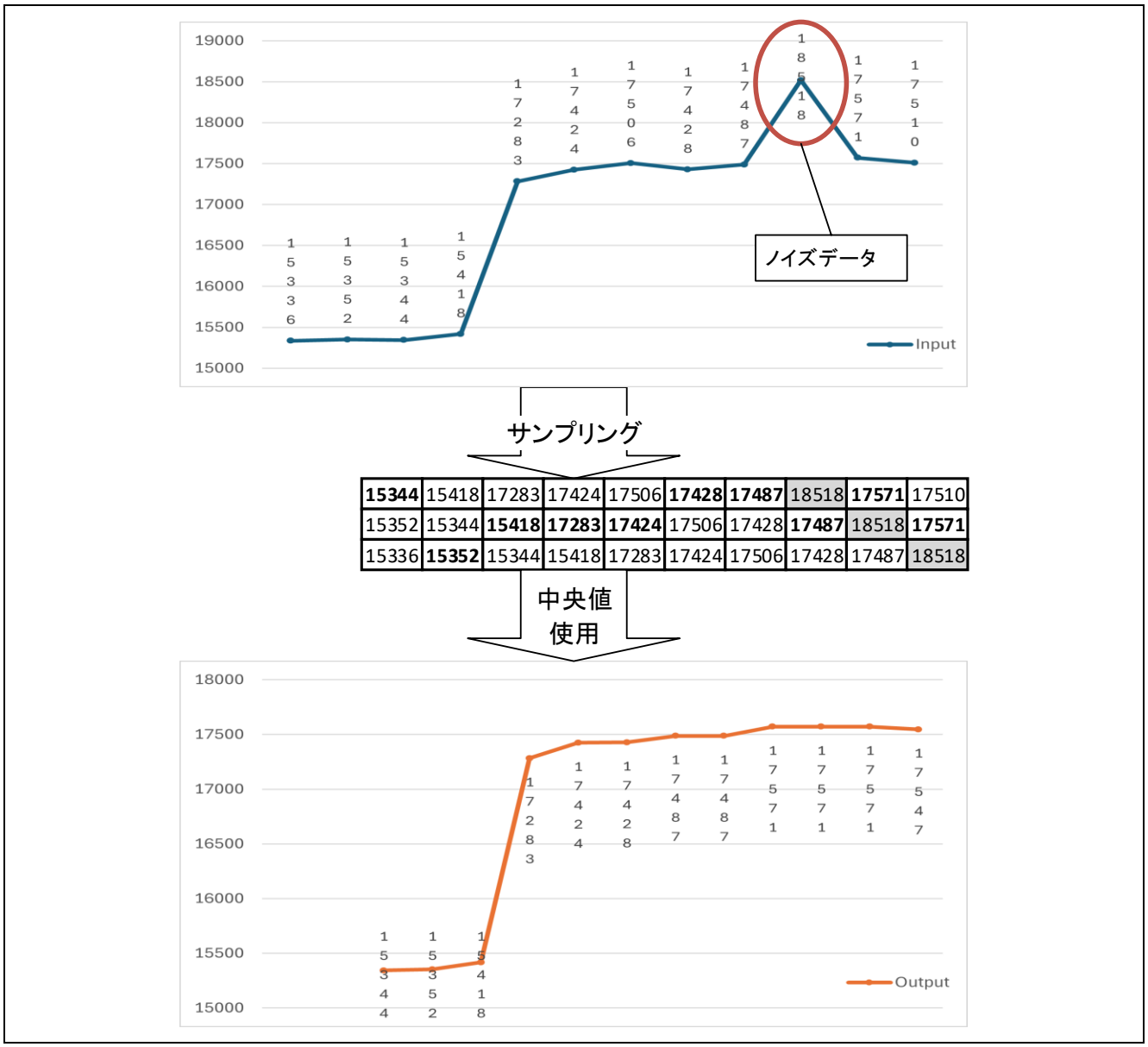


図 6-1 メディアンフィルタ動作

6.1.1　検出遅延について

　メディアンフィルタではタッチ計測値のサンプリングによってノイズ信号の除去を行うため通常のタッチ検出に遅延が発生します。

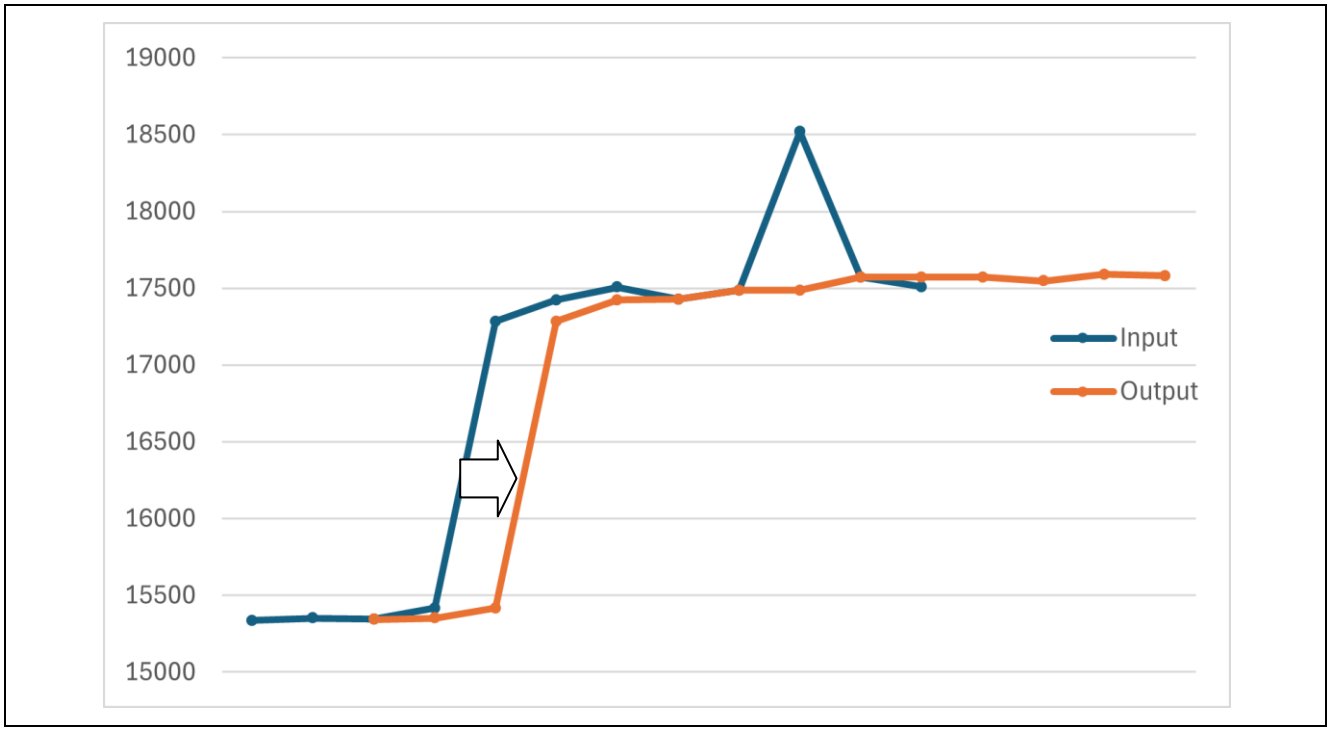


図 6-2 メディアンフィルタの検出遅延

6.1.2　フィルタ安定時間について

メディアンフィルタは初期化後、サンプリングするデータ個数回分のフィルタ処理を実行するまでは入力データに対して低い演算結果が出力されます。

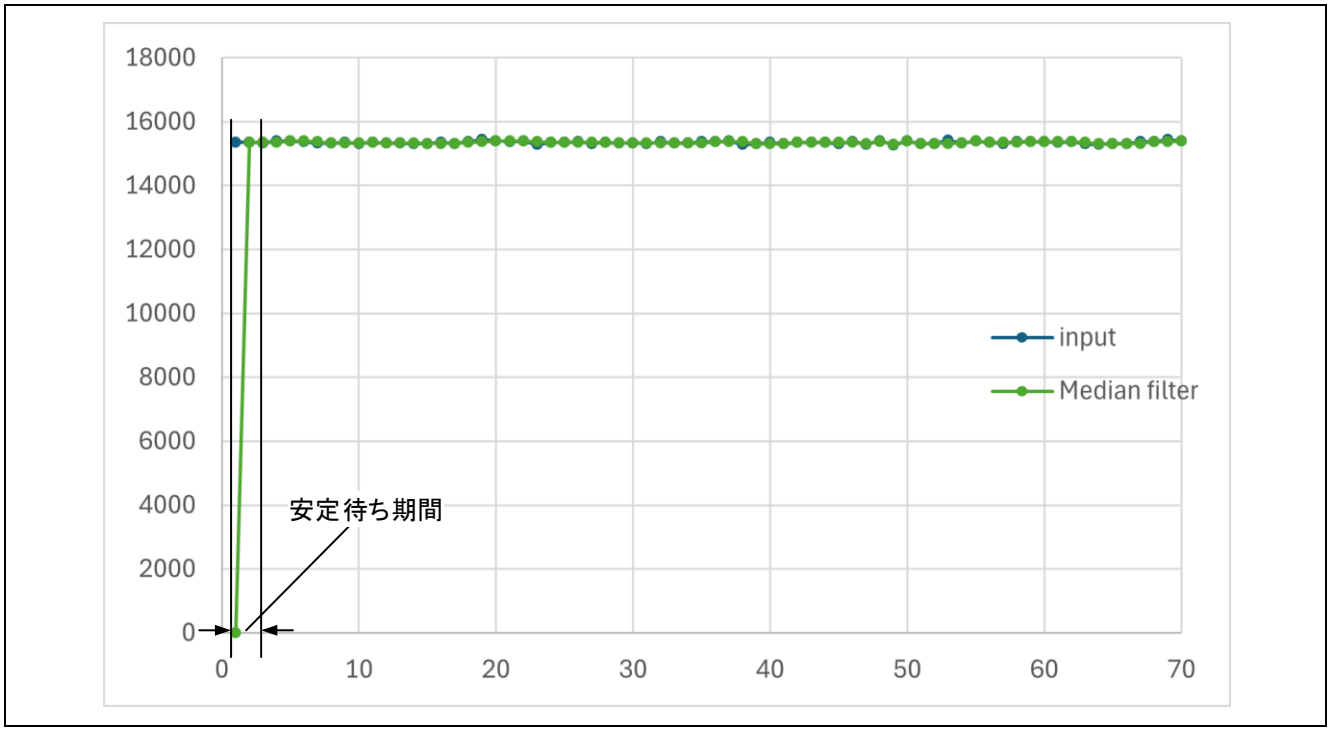


図 6-3 フィルタ安定待ち時間(メディアン)

6.2　フィルタ仕様

表 6-1 に本サンプルプログラムのメディアンフィルタの仕様を示します。

表 6-1 メディアンフィルタ仕様

項目	仕様	備考
入力データ型	符号無し 16bit 整数型	
出力データ型	符号無し 16bit 整数型	
サンプル参照範囲	3,5,7,9	入力値と過去サンプルの合計個数
処理方法	挿入ソートによる中央値検出	
フィルタ初期化後の出力結果	フィルタ安定時間内の演算結果およびバッファ充填中応答を返す	フィルタ安定時間はサンプル参照期間

6.2.1　フィルタ処理方法

メディアンフィルタは入力データをサンプリングし、サンプリングデータを一時バッファにコピーしてソートし、中央位置のデータを出力します。

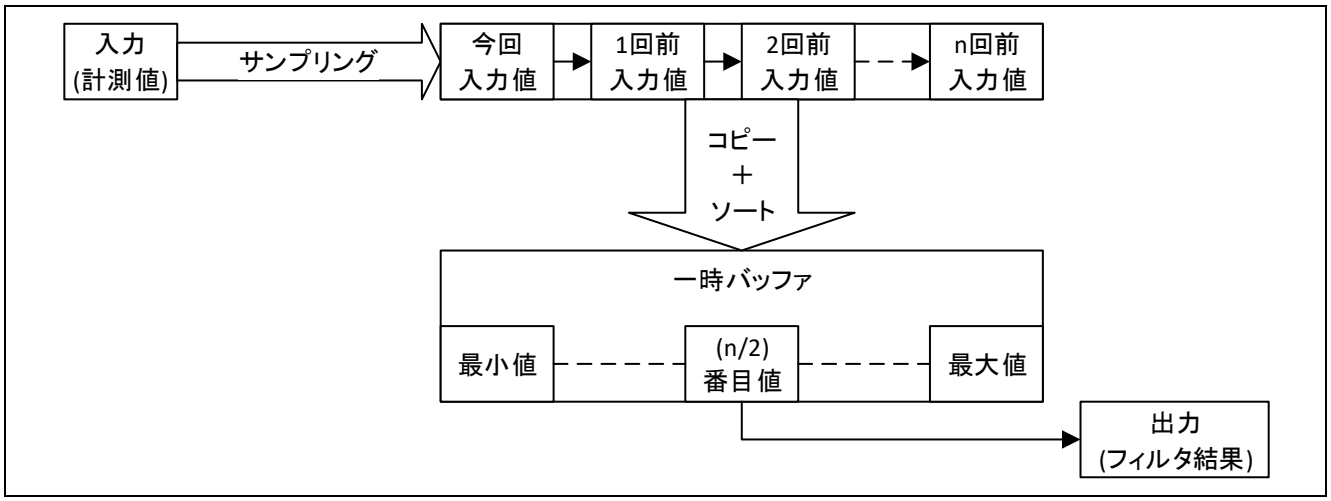


図 6-4 メディアンフィルタ処理

6.3　メディアンフィルタ用データ一覧

本節ではメディアンフィルタ用に用意している定数・グローバル変数について説明します。

6.3.1　定数

表 6-2 にメディアンフィルタ用定数の一覧を示します。

表 6-2 メディアンフィルタ用定数

定数名	設定値	内容
ファイル名：r_ctsu_median_sample.h		
MEDIAN_SAMPLE_SIZE	3	サンプリング数
MEDIAN_DATA_SIZE	1	フィルタ対象データ数
ファイル名：r_ctsu_median_sample.c		
MEDIAN_SAMPLE_SIZE_MIN	3	最小サンプル参照期間
MEDIAN_SAMPLE_SIZE_MAX	9	最大サンプル参照期間

6.3.2　構造体

メディアンフィルタ API アクセス用の管理データ用構造体を示します。

6.3.2.1　メディアンフィルタ管理データ(median_ctrl_t)

表 6-3 メディアンフィルタ管理データ構造体(median_ctrl_t)

メンバ	データ型	内容
count	uint16_t	安定待ち時間カウンタ
index	uint16_t	メディアンフィルタ用サンプリングバッファ入力データ格納位置
median_data	uint16_t [MEDIAN_DATA_SIZE][MEDIAN_SAMPLE_SIZE]	メディアンフィルタ用サンプリングバッファ

6.4　メディアンフィルタ API

6.4.1　r_ctsu_median_initial

この関数は、メディアンフィルタ処理用管理データの初期化を行う関数です。

この関数は r_ctsu_median_filter を使用する前に実行する必要があります。

Format

fsp_err_t r_ctsu_median_initial(median_ctrl_t * const p_ctrl);

Parameters

p_ctrl

メディアンフィルタ管理用データへのポインタ

ReturnValues

FSP_SUCCESS	/* 正常終了 */
FSP_ERR_INVALID_ARGUMENT	/* 構成定義のパラメータが不正です */

Properties

r_ctsu_median_sample.h にプロトタイプ宣言されています。

Description

この関数は、メディアンフィルタ処理用管理データの初期化を行います。

サンプリング数の定義が有効範囲外の場合はエラー応答を返します。

Example

```
err = r_ctsu_median_initial(&g_ctsu_median_control);
if (FSP_SUCCESS != err)
{
 while (true) {}
}
```

6.4.2　r_ctsu_median_filter

この関数はメディアンフィルタ演算を行います。

Format

fsp_err_t r_ctsu_median_filter(median_ctrl_t * const p_ctrl , uint16_t *p_data);

Parameters

p_ctrl

メディアンフィルタ管理用データへのポインタ

p_data

メディアンフィルタを適用する計測値データへのポインタ

ReturnValues

FSP_SUCCESS	/* 正常終了 */
FSP_ERR_BUFFER_EMPTY	/* バッファ充填中のため未適用のフィルタがあります */

Properties

r_ctsu_median_sample.h にプロトタイプ宣言されています。

Description

この関数はのメディアンフィルタ処理を行います。

初期化後、サンプリング数回の処理実行中はエラー応答を返します。

Example

```
/* Median filter sample */
err = R_CTSU_DataGet(g_qe_ctsu_instance_config05.p_ctrl, filter_buffer);
if(err == FSP_SUCCESS)
{
 err=r_ctsu_median_filter(&g_ctsu_median_control, filter_buffer);
 if(err == FSP_SUCCESS)
 {
    R_CTSU_DataInsert(g_qe_ctsu_instance_config05.p_ctrl, filter_buffer);
 }
}
```

Special Notes:

エラー応答時もフィルタ演算は実施されます。

複数のフィルタを縦続構成する場合はエラー応答時も後段のフィルタ処理は実行してください。

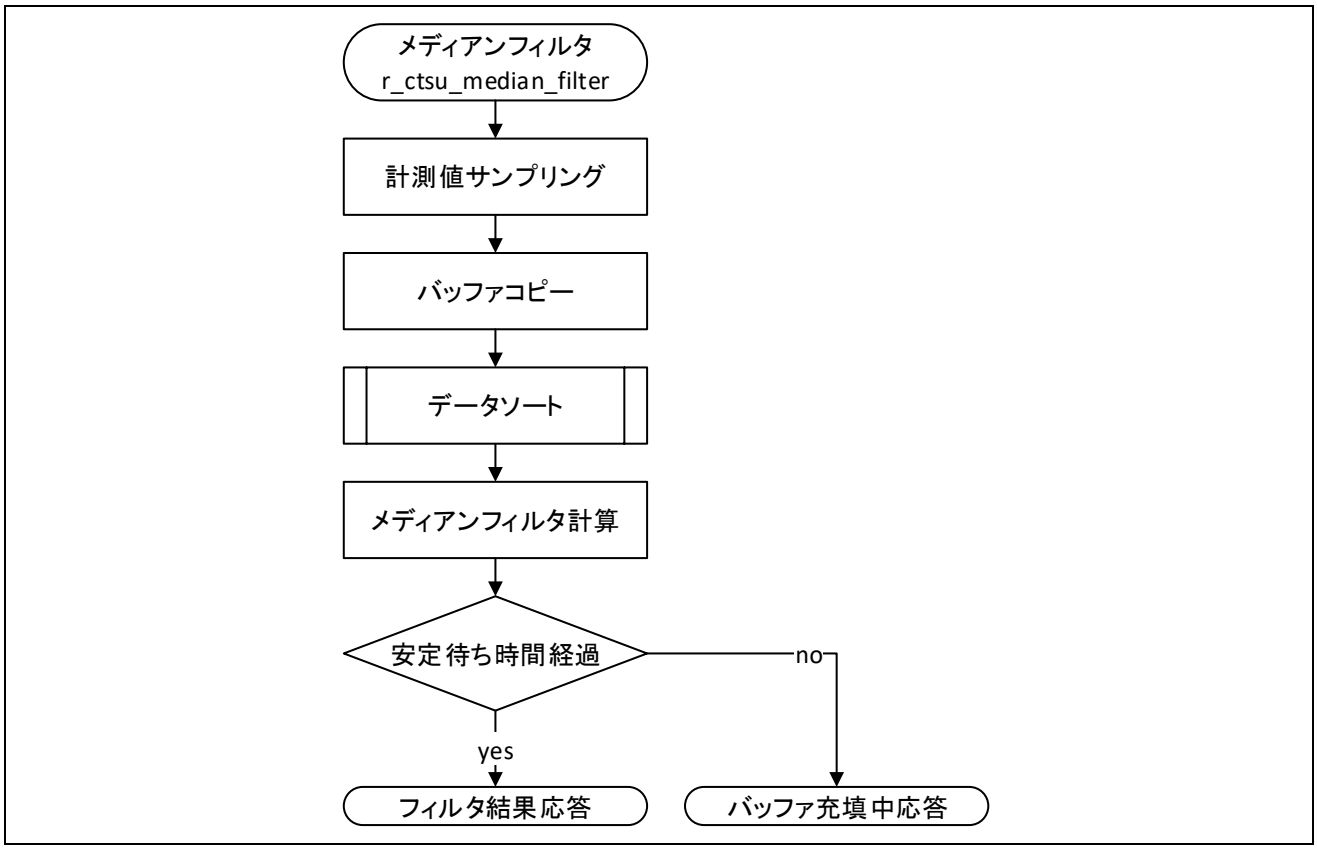


図 6-5 メディアンフィルタ実行 API フロー

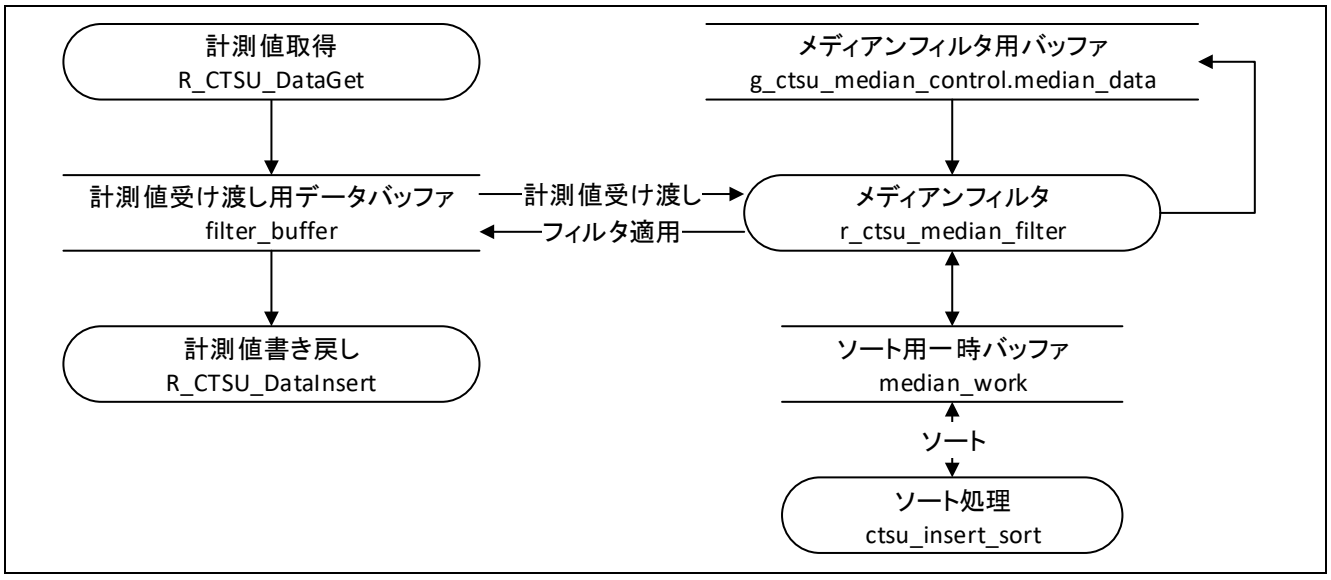


図 6-6 メディアンフィルタデータフロー

6.4.1　ctsu_insert_sort

この関数は、指定したデータを昇順に並べ替えます。

Format

static void ctsu_insert_sort(uint16_t * list , uint16_t size);

Parameters

p_list

並び替えを行うデータへのポインタ

size

並び替えを行うデータの個数

ReturnValues

なし

Properties

r_ctsu_median_sample.c にプロトタイプ宣言されています。

Description

この関数は、指定したデータを昇順に並び替えます。

6.5　使用例

本サンプルプログラムでは使用例としてサンプル参照期間が 3 のメディアンフィルタを提供します。

6.5.1　フィルタ特性

表 6-4 に本サンプルプログラムのメディアンフィルタ特性を示します。

表 6-4 サンプルメディアンフィルタ特性

サンプル参照期間	3(60ms)
除去ノイズ幅	1(20ms)
検出遅延	1(20ms)

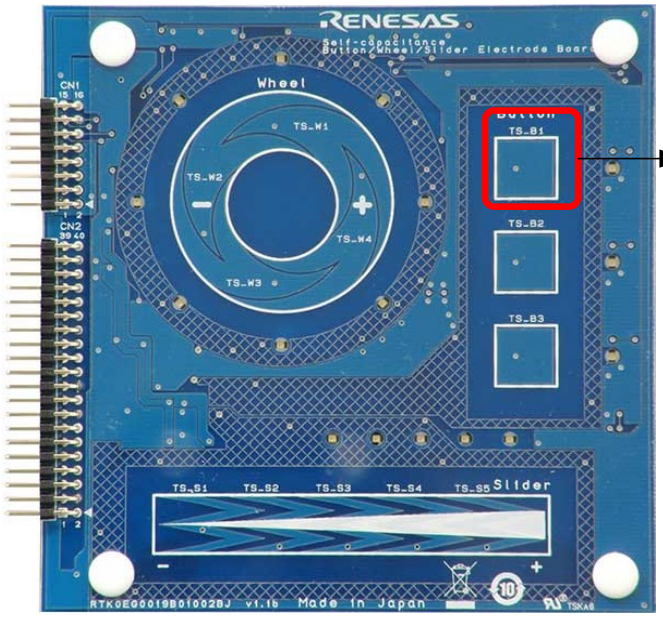
7.　本サンプルプロジェクト/サンプルコードの使用方法

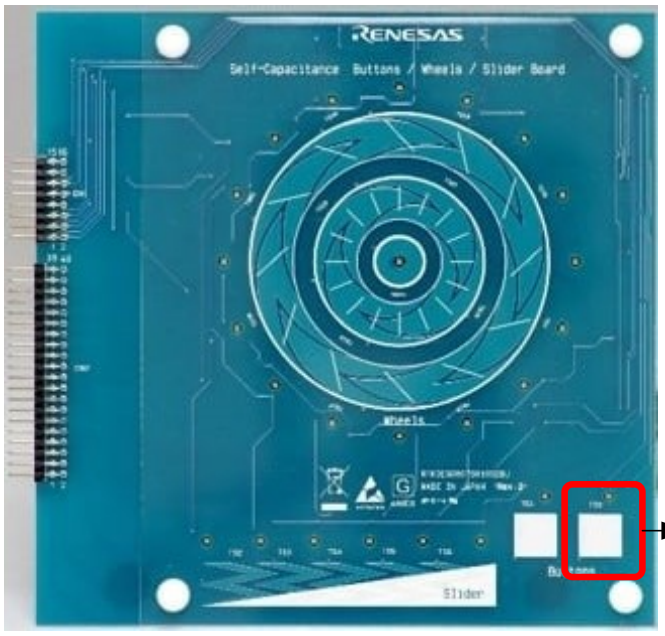
7.1　サンプルプロジェクトの使用方法

ソフトウエアフィルタサンプルプログラムを適用したサンプルプロジェクトの使用方法を説明します。

7.1.1　サンプルアプリケーション

本サンプルプロジェクトのアプリケーションでは 1 ボタンのみ使用し、5 種類のフィルタを適用したタッチ検出を行っています。





FIR移動平均フィルタ
IIRノッチフィルタ
IIRローパスフィルタ
単極IIRローパスフィルタ
メディアンフィルタ

7.1.2　機能

以下にサンプルプロジェクトの機能を示します。

- 使用するタッチスイッチは自己容量方式電極ボードのタッチボタン 1(RA2L1 と RL78G16 は TS-B1、RX130 は TS0)のみとなります。
- 自己容量方式電極ボードのタッチボタン 1 の計測結果に対して 5 つのメソッドを定義し、それぞれに異なるソフトウエアフィルタを適用しています。
- ソフトウエアフィルタを適用した計測結果は QE for Capacitive Touch のシリアルモニター機能で確認できます。

表 7-1 メソッド定義と適用フィルタ

メソッド	フィルタ内容
CONFIG01	FIR 移動平均フィルタ
CONFIG02	IIR ノッチフィルタ
CONFIG03	IIR ローパスフィルタ
CONFIG04	単極 IIR ローパスフィルタ
CONFIG05	メディアンフィルタ

7.1.3　ファイル構成

サンプルプロジェクトのファイル構成を示します。

7.1.3.1　RA2L1 グループ

開発環境のプロジェクト構成ファイルとスマート・コンフィグレータの生成ファイルは省略しています。

```
ra2l1_filter_sample
 │
 ├──QE-Touch
 │      ├ ra2l1_filter_sample_log_tuning20240322151628.log   ・・・QE チューニングログ
 │      └ ra2l1_filter_sample.tifcfg                         ・・・タッチインターフェース構成ファイル
 │
 ├──qe_gen
 │      ├ qe_touch_config.c                                  ・・・タッチ構成ソース
 │      ├ qe_touch_config.h                                  ・・・タッチ構成ヘッダ
 │      ├ qe_touch_define.h                                  ・・・タッチ定義ヘッダ
 │      └ qe_touch_sample.c                                  ・・・タッチサンプルアプリケーション
 │
 ├──src
 │      └ hal_entry.c                                        ・・・main ファイル
 │
 └──filter_sample
        ├ filter_define_sample.h                             ・・・MCU 関連用ヘッダ
        ├ fir
        │      ├ fir_config_sample1.c                       ・・・FIR 移動平均フィルタ用構成定義ソース
        │      ├ r_ctsu_fir_sample.c                        ・・・FIR フィルタ処理用ソース
        │      └ r_ctsu_fir_sample.h                        ・・・FIR フィルタ処理用ヘッダ
        ├ iir
        │      ├ iir_config_sample1.c                       ・・・IIR フィルタ用構成定義ソース
        │      ├ r_ctsu_iir_sample.c                        ・・・IIR フィルタ処理用ソース
        │      └ r_ctsu_iir_sample.h                        ・・・IIR フィルタ処理用ヘッダ
        ├ median
        │      ├ median_config_sample1.c                    ・・・メディアンフィルタ用構成定義ソース
        │      ├ r_ctsu_median_sample.c                     ・・・メディアンフィルタ処理用ソース
        │      └ r_ctsu_median_sample.h                     ・・・メディアンフィルタ処理用ヘッダ
        └ spiir
               ├ spiir_config_sample1.c                     ・・・単極 IIR ローパスフィルタ用構成定義ソース
               ├ r_ctsu_spiir_sample.c                      ・・・単極 IIR フィルタ処理用ソース
               └ r_ctsu_spiir_sample.h                      ・・・単極 IIR フィルタ処理用ヘッダ
```

7.1.3.2　RX130 グループ

開発環境のプロジェクト構成ファイルとスマート・コンフィグレータの生成ファイルは省略しています。

```
rx130_filter_sample
 │
 ├──QE-Touch
 │      ├ rx130_filter_sample_log_tuning20240322164158.log  ・・・QE チューニングログ
 │      └ rx130_filter_sample.tifcfg                ・・・タッチインターフェース構成ファイル
 │
 ├──qe_gen
 │      ├ qe_touch_config.c                         ・・・タッチ構成ソース
 │      ├ qe_touch_config.h                         ・・・タッチ構成ヘッダ
 │      ├ qe_touch_define.h                         ・・・タッチ定義ヘッダ
 │      └ qe_touch_sample.c                         ・・・タッチサンプルアプリケーション
 │
 ├──src
 │      └ rx130_filter_sample.c                     ・・・main ファイル
 │
 └──filter_sample
        ├ filter_define_sample.h                    ・・・MCU 関連用ヘッダ
        ├ fir
        │      ├ fir_config_sample1.c              ・・・FIR 移動平均フィルタ用構成定義ソース
        │      ├ r_ctsu_fir_sample.c               ・・・FIR フィルタ処理用ソース
        │      └ r_ctsu_fir_sample.h               ・・・FIR フィルタ処理用ヘッダ
        ├ iir
        │      ├ iir_config_sample1.c              ・・・IIR フィルタ用構成定義ソース
        │      ├ r_ctsu_iir_sample.c               ・・・IIR フィルタ処理用ソース
        │      └ r_ctsu_iir_sample.h               ・・・IIR フィルタ処理用ヘッダ
        ├ median
        │      ├ median_config_sample1.c           ・・・メディアンフィルタ用構成定義ソース
        │      ├ r_ctsu_median_sample.c            ・・・メディアンフィルタ処理用ソース
        │      └ r_ctsu_median_sample.h            ・・・メディアンフィルタ処理用ヘッダ
        └ spiir
               ├ spiir_config_sample1.c            ・・・単極 IIR ローパスフィルタ用構成定義ソース
               ├ r_ctsu_spiir_sample.c             ・・・単極 IIR フィルタ処理用ソース
               └ r_ctsu_spiir_sample.h             ・・・単極 IIR フィルタ処理用ヘッダ
```

7.1.3.3　RL78/G16 グループ

開発環境のプロジェクト構成ファイルとスマート・コンフィグレータの生成ファイルは省略しています。

```
rl78g16_filter_sample
 │
 ├──QE-Touch
 │      ├ rl78g16_filter_sample_log_tuning20240325091840.log        ・・・QE チューニングログ
 │      └ rl78g16_filter_sample.tifcfg              ・・・タッチインターフェース構成ファイル
 │
 ├──qe_gen
 │      ├ qe_touch_config.c                        ・・・タッチ構成ソース
 │      ├ qe_touch_config.h                        ・・・タッチ構成ヘッダ
 │      ├ qe_touch_define.h                        ・・・タッチ定義ヘッダ
 │      └ qe_touch_sample.c                        ・・・タッチサンプルアプリケーション
 │
 ├──src
 │      └ rl78g16_filter_sample.c                        ・・・main ファイル
 │
 └──filter_sample
        ├ filter_define_sample.h                   ・・・MCU 関連用ヘッダ
        ├ fir
        │       ├ fir_config_sample1.c            ・・・FIR 移動平均フィルタ用構成定義ソース
        │       ├ r_ctsu_fir_sample.c             ・・・FIR フィルタ処理用ソース
        │       └ r_ctsu_fir_sample.h             ・・・FIR フィルタ処理用ヘッダ
        ├ iir
        │       ├ iir_config_sample1.c            ・・・IIR フィルタ用構成定義ソース
        │       ├ r_ctsu_iir_sample.c             ・・・IIR フィルタ処理用ソース
        │       └ r_ctsu_iir_sample.h             ・・・IIR フィルタ処理用ヘッダ
        ├ median
        │       ├ median_config_sample1.c         ・・・メディアンフィルタ用構成定義ソース
        │       ├ r_ctsu_median_sample.c          ・・・メディアンフィルタ処理用ソース
        │       └ r_ctsu_median_sample.h          ・・・メディアンフィルタ処理用ヘッダ
        └ spiir
                ├ spiir_config_sample1.c          ・・・単極 IIR ローパスフィルタ用構成定義ソース
                ├ r_ctsu_spiir_sample.c           ・・・単極 IIR フィルタ処理用ソース
                └ r_ctsu_spiir_sample.h           ・・・単極 IIR フィルタ処理用ヘッダ
```

7.1.4　インポート方法

本サンプルコードに付属している「xxxx_filter_sample.zip」ファイル（xxxx は MCU グループ名）を e²studio のインポート機能によりワークスペースへインポートしてください。

図 7-1 にサンプルプロジェクトのインポート方法を示します。

インポート後の操作については各 MCU ファミリのクイックスタートガイドまたはアプリケーションノートを参照してください

- RA ファミリ
  Renesas RA ファミリ RA2L1 グループ 静電容量タッチ評価システム クイックスタートガイド (Q12QS0040)
- RX ファミリ
  QE と FIT を使用した静電容量タッチアプリケーションの開発(R01AN4516)
- RL78 ファミリ
  RL78 ファミリ QE と SIS を使用した静電容量タッチアプリケーションの開発(R01AN5512)

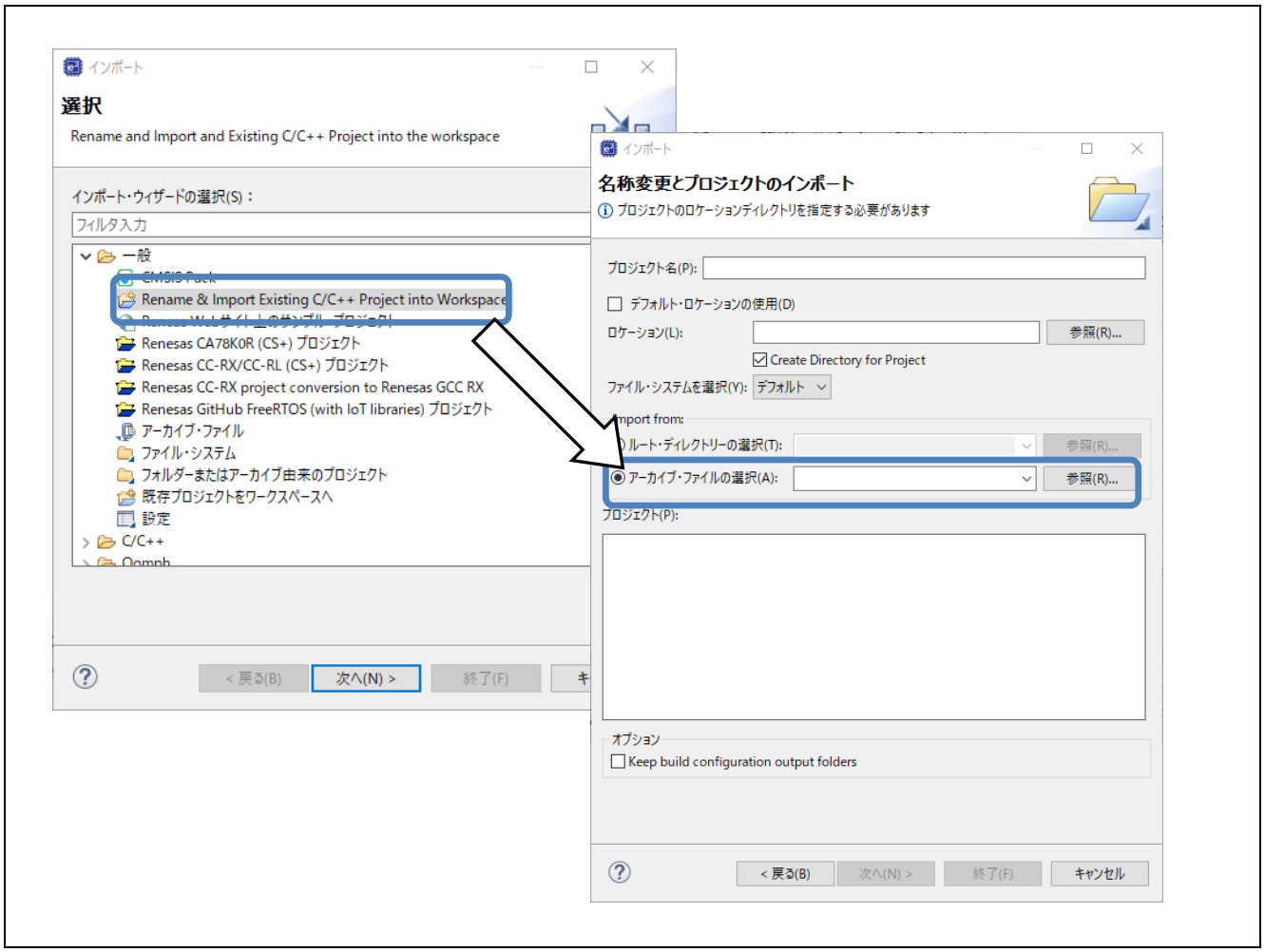


図 7-1 サンプルプロジェクトのインポート

7.2　フィルタサンプルコードの使用方法

7.2.1　既存プロジェクトへのサンプルコード組み込み手順

既存の静電容量センサユニットアプリケーションにソフトウエアフィルタを組み込む場合は以下の手順で行います。

1.filter_sample フォルダを対象のプロジェクトにコピーします。
filter_sample フォルダ内の使用しないフィルタ名のフォルダは削除可能です。

2.メニューのプロジェクト⇒C/C++Project Settings を開き、C/C++一般⇒パスおよびシンボルのインクルードとソース・ロケーションに filter_sample フォルダを追加します。

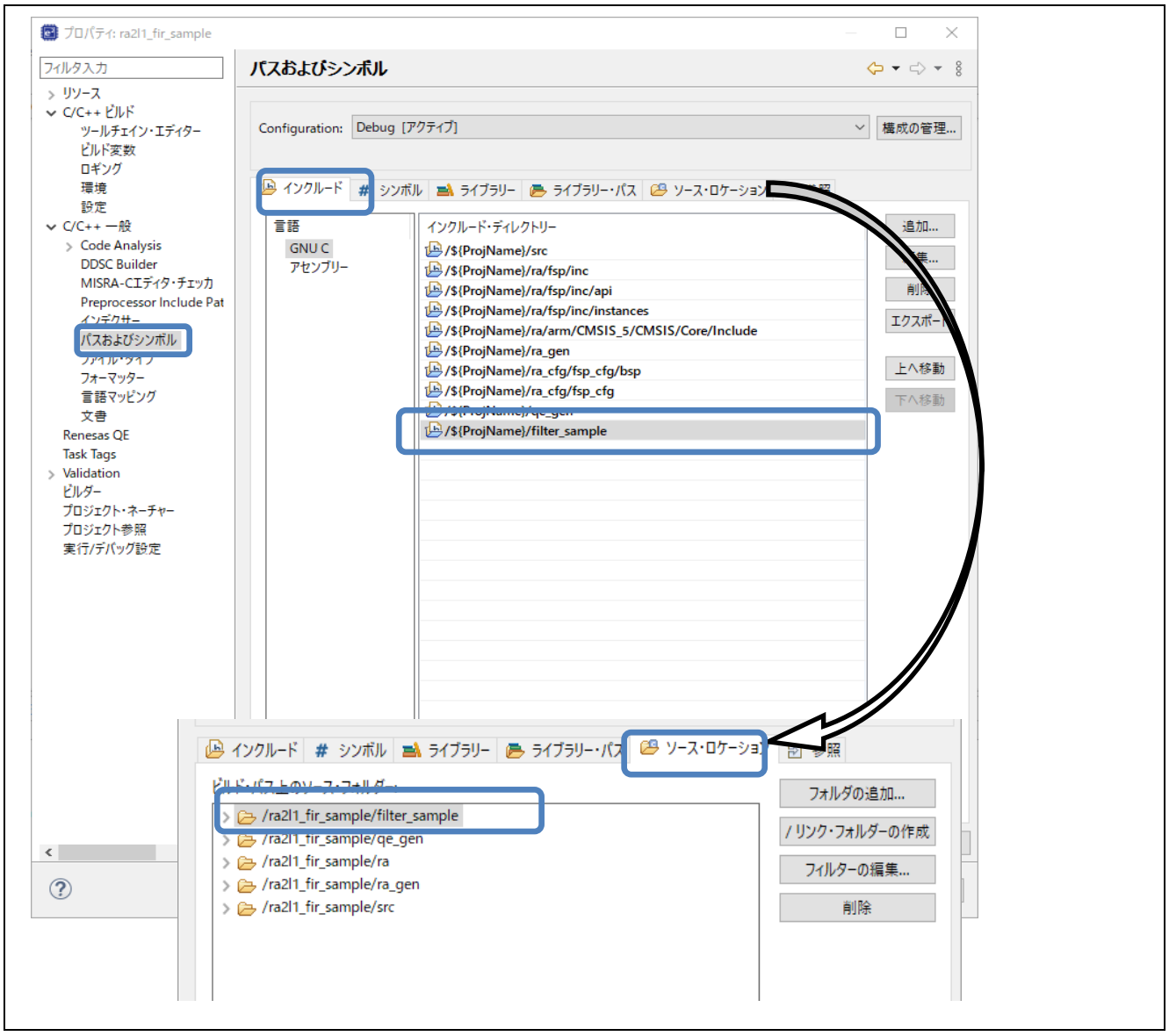


図 7-2 既存環境へのサンプルプログラムの組み込み

3. タッチインターフェースのメソッド構成に応じて XXX_config_sample1.c(XXX は使用するフィルタ名)内のフィルタ管理用データを追加します。

例：FIR フィルタを 3 つのメソッドに使用する場合

fir_config_sample1.c

```
fir_ctrl_t g_ctsu_fir_control;
fir_ctrl_t g_ctsu_fir_control2;
fir_ctrl_t g_ctsu_fir_control3;
```

3 つのメソッドで FIR フィルタを使用するので管理データが 3 個になるように追加する

4. ソフトウエアフィルタを使用するタッチインターフェースのメソッド構成のタッチ計測データ数に合わせて r_ctsu_XXX_sample.h(XXX は使用するフィルタ名)内の XXX_DATA_SIZE 定義を変更してください。
メソッドのタッチ計測データ数は自己容量方式では TS 端子数、相互容量方式ではマトリックス数の 2 倍となります。
XXX_DATA_SIZE 定義はフィルタを使用するメソッドのタッチ計測データ数の内、最大の値を使用してください。

例：FIR フィルタを TS 端子数 3/4/5 の 3 つのメソッドに使用する場合

r_ctsu_fir_sample.h

```
#define FIR_DATA_SIZE    (5)
```

FIR フィルタを使用するメソッドの内、TS 端子数が最も多いメソッドの端子数に変更する

5. qe_touch_sample.c ファイル(もしくはそれに準ずるファイル)に使用するフィルタに応じたファイルのインクルード指定を追加し、フィルタ処理実施コードを追加します(7.2.2 章を参照してください)。

【注】 1. フィルタ処理用のデータ読み出し、データ書き戻しは Touch API ではなく CTSU API であることに注意してください。
2. フィルタ処理の実行記載はタッチインターフェース構成のメソッド毎に必要であることに注意してください。

表 7-2 追加インクルードファイル

使用するフィルタ	インクルード指定ファイル
FIR フィルタ	\fir\r_ctsu_fir_sample.h
IIR フィルタ	\iir\r_ctsu_iir_sample.h
単極 IIR フィルタ	\spiir\r_ctsu_spiir_sample.h
メディアンフィルタ	\median\r_ctsu_median_sample.h

6. qe_touch_config.c ファイル(もしくはそれに準ずるファイル)内の CTSU ドライバ構成定義(QE for Capacitive Touch 生成では g_qe_ctsu_ctrl_XXX)の num_moving_average 設定を 1 に変更しデフォルトの移動平均処理を無効にします。デフォルトの移動平均処理とサンプルコードを併用する場合は変更は不要です。
   タッチインターフェース構成のメソッドが複数ある場合は全てのメソッドの CTSU ドライバ構成定義を変更してください。

- タッチインターフェース構成定義の記述例

qe_touch_config.c（QE for Capacitive Touch で生成されるファイル）

```
const ctsu_cfg_t g_qe_ctsu_cfg_config01 =
{
(省略)
    .num_moving_average = 1,
    .tunning_enable     = true,
    .p_callback    = &qe_touch_callback,
(省略)
};

ctsu_instance_ctrl_t g_qe_ctsu_ctrl_config01;

const ctsu_instance_t g_qe_ctsu_instance_config01 =
{
    .p_ctrl = &g_qe_ctsu_ctrl_config01,
    .p_cfg  = &g_qe_ctsu_cfg_config01,
    .p_api  = &g_ctsu_on_ctsu,
};


const ctsu_cfg_t g_qe_ctsu_cfg_config02 =
{
(省略)
    .num_moving_average = 1,
    .tunning_enable     = true,
    .p_callback    = &qe_touch_callback,
(省略)
};

ctsu_instance_ctrl_t g_qe_ctsu_ctrl_config02;

const ctsu_instance_t g_qe_ctsu_instance_config02 =
{
    .p_ctrl = &g_qe_ctsu_ctrl_config02,
    .p_cfg  = &g_qe_ctsu_cfg_config02,
    .p_api  = &g_ctsu_on_ctsu,
};
```

1 に変更する

1 に変更する

7.2.2　サンプルアプリケーション構成と動作

QE for Capacitive Touch で出力したサンプルコード(qe_touch_sample.c)に FIR フィルタのサンプルプログラムを組み込む場合のフローチャートを示します。

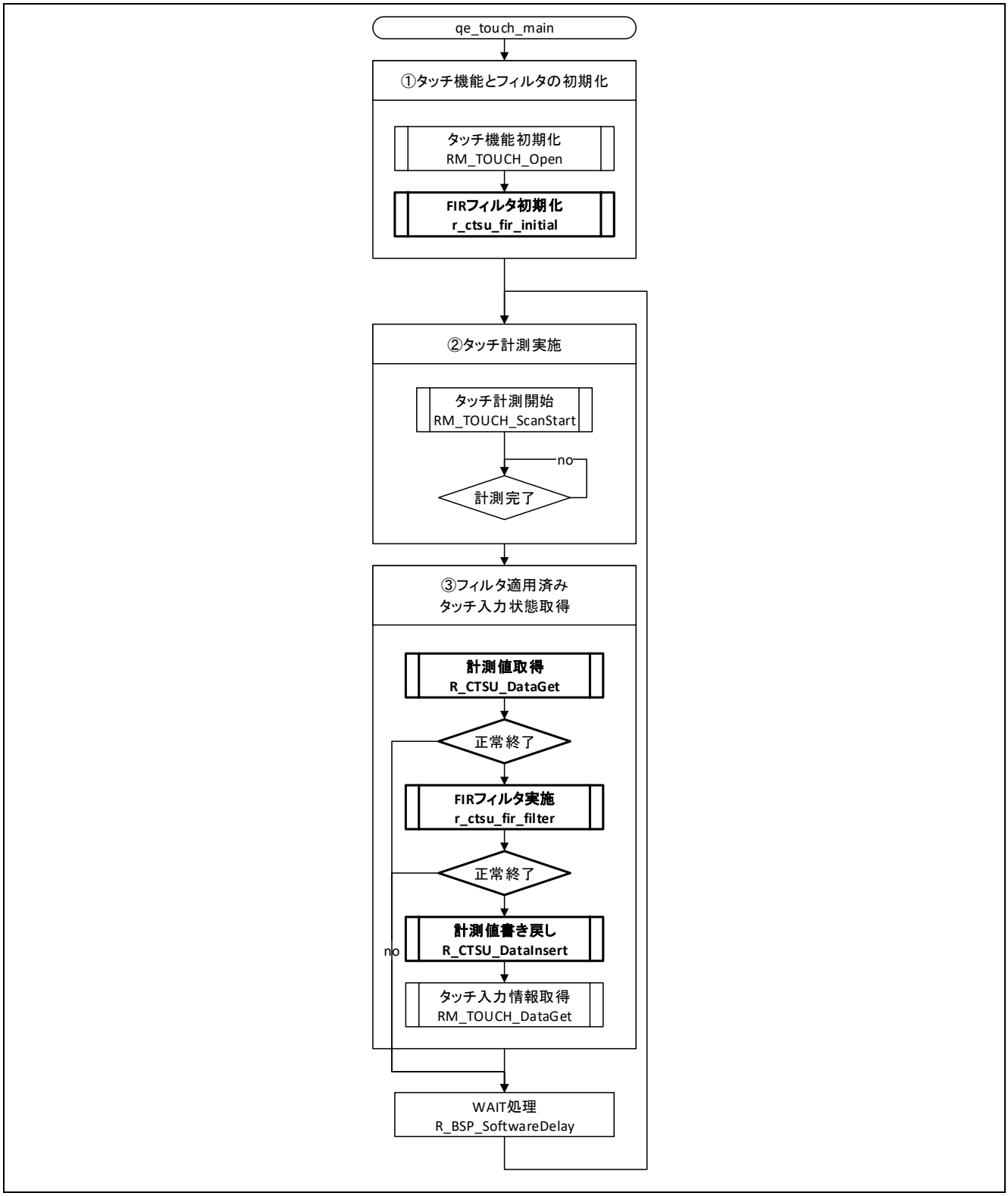


図 7-3 サンプルアプリケーションフロー

図 7-3 の図中番号の説明を示します。

① フィルタの初期化

フィルタ構成定義の有効範囲をチェックしフィルタ管理用データを初期化します。

```
/* Check filter configuration & Initialize filter control data */
err = r_ctsu_fir_initial(&g_ctsu_fir_control, &g_fir_cfg);
if (FSP_SUCCESS != err)
{
 while (true) {}
}
```

② タッチ計測実施

タッチ計測を行い、計測完了を待ちます。

③ フィルタ適用とタッチ入力状態取得

CTSU ドライバ API を使用して計測結果を取得し、フィルタ適用後に CTSU ドライバに書き戻し、タッチ入力情報を取得します。

CTSU ドライバ API の詳細は Renesas Flexible Software Package (FSP) User's Manual (R11UM0155)の v4.3.0 以降を参照してください。

```
/* FIR moving average filter sample */
err = R_CTSU_DataGet(g_qe_ctsu_instance_config01.p_ctrl, filter_buffer);
if(err == FSP_SUCCESS)
{
 err=r_ctsu_fir_filter(&g_ctsu_fir_control,&g_fir_cfg, filter_buffer);
 if(err == FSP_SUCCESS)
 {
    R_CTSU_DataInsert(g_qe_ctsu_instance_config01.p_ctrl, filter_buffer);
    err = RM_TOUCH_DataGet(g_qe_touch_instance_config01.p_ctrl,
&button_status, NULL, NULL);
    if (FSP_SUCCESS == err)
    {
       /* TODO: Add your own code here. */
    }
 }
}
```

7.2.3　フィルタ特性の調整方法

7.2.3.1　固定少数点定義について

FIR/IIR/単極 IIR フィルタの係数定義は int16_t 型の固定少数点で定義します。

固定少数点定義の小数部桁数はフィルタ種類によって異なります。

固定少数点定義は「小数値 × $2^{\text{固定少数点桁数}}$」で計算してください。

表 7-3 に固定小数点定義の対応例を示します。

表 7-3 固定小数点定義例

小数	固定少数点桁数	10 進数	備考
0.199951171875	14	3276=0.199951171875 x 16384	FIR、単極 IIR
-0.61376953125	11	-1257= -0.61376953125 x 2048	IIR

7.2.3.2　FIR フィルタ

FIR フィルタの係数定義を変更し、フィルタ特性を調整することができます。

係数定義は固定少数点桁数 14 の符号付き 16bit 固定少数点で最大 9 個まで定義でき、記載順に図 3-2 の係数 h(0)から h(8)の順として扱われます。

タップ数には定義した係数値の個数を記載します。

```
const int16_t g_fir_coefficient[] =
{
  3276,         /* h0 : 0.199951171875 */
  3276,         /* h1 : 0.199951171875 */
  3276,         /* h2 : 0.199951171875 */
  3276,         /* h3 : 0.199951171875 */
  3276,         /* h4 : 0.199951171875 */
};

const fir_config_t g_fir_cfg =
{
  .taps = 5,
  .p_coefficient = g_fir_coefficient,
};
```

係数値を固定小数点桁数 14 の固定少数値で 9 個まで記載する

係数値の個数を記載する

7.2.3.3　IIR フィルタ

IIR フィルタの係数定義を変更し、フィルタ特性を調整することができます。

係数定義は固定少数点桁数 11 の符号付き 16bit 固定少数点で 5 個定義し、図 4-2 の係数 b(0)、b(1)、b(2)、a(1)、a(2)の順として扱われます。

係数設定後は安定待ち時間に暫定の値を設定してリセットスタート時の計測値を確認してください。

安定待ち時間が不十分な場合はタッチ計測の基準値が低い値となり、タッチ検出が正常に行われない状態となるので安定待ち時間を調整して正常にタッチ計測が行われるようにしてください。

```
const iir_config_t g_iir_cfg1 =
{
  .settlings = 3,
  .coefficient = {
   /* coefficient b */
   2034,                /* b0 : 0.9931640625    */
   -1257,               /* b1 : -0.61376953125  */
   2034,                /* b2 : 0.9931640625    */
   /* coefficient a */
   1257,                /* a1 : 0.61376953125   */
   -2021,               /* a2 : -0.98681640625  */
  }
};
```

リセットスタート時の計測値を確認して調整する

係数値を固定小数点桁数 11 の固定少数値で係数 b、係数 a の順に記載する

7.2.3.4　縦続型 IIR フィルタ構成

IIR フィルタを縦続構成にすることで 3 次以上の IIR フィルタを構成することができます。

アプリケーションから IIR フィルタ処理を 2 回以上連続して実行してください。

連続実行するフィルタ処理にはそれぞれ個別の管理データと構成定義が必要となります。

```
  err = R_CTSU_DataGet(g_qe_ctsu_instance_config02.p_ctrl, filter_buffer);
  if(err == FSP_SUCCESS)
  {
    err1=r_ctsu_iir_filter(&g_ctsu_iir_control1, &g_iir_cfg1, filter_buffer);
    err2=r_ctsu_iir_filter(&g_ctsu_iir_control2, &g_iir_cfg2, filter_buffer);
    if((err1 == FSP_SUCCESS) && (err1 == FSP_SUCCESS))
    {
       R_CTSU_DataInsert(g_                                          ffer);
       err = RM_TOUCH_DataG
&button_status, NULL, NULL
       if (FSP_SUCCESS == err)
       {
          /* TODO: Add your own code here. */
       }
    }
  }
```

管理データ、構成定義の異なるフィルタ処理を連続して実行する

7.2.3.5　単極 IIR フィルタ

単極 IIR フィルタの係数定義を変更し、フィルタ特性を調整することができます。

係数定義は固定少数点桁数 14 の符号付き 16bit 固定少数点で 2 個定義し、図 5-2 の係数 a、b の順として扱われます。

係数設定後は安定待ち時間に暫定の値を設定してリセットスタート時の計測値を確認してください。

安定待ち時間が不十分な場合はタッチ計測の基準値が低い値となり、タッチ検出が正常に行われない状態となるので安定待ち時間を調整して正常にタッチ計測が行われるようにしてください。

```
const spiir_config_t g_spiir_cfg =
{
  .settlings = 128,
  .coefficient = {
   /* coefficient a */
   15386,              /* a : 0.9390869140625   */
   /* coefficient b */
   997,                /* b : 0.06085205078125 */
  },
};
```

リセットスタート時の計測値を確認して調整する

係数値を固定小数点桁数 14 の固定少数値で係数 a、係数 b の順に記載する

7.2.3.6　メディアンフィルタ

メディアンフィルタのサンプリング数を変更し、除去可能なノイズ幅を調整することができます。

```
#define MEDIAN_SAMPLE_SIZE  (3)
```

サンプリング数を指定する

8. 参考資料

- 静電容量センサマイコン 静電容量タッチ ノイズイミュニティガイド(R30AN0426)
- Renesas RA ファミリ RA2L1 グループ 静電容量タッチ評価システム クイックスタートガイド (Q12QS0040)
- RA ファミリ QE と FSP を使用した静電容量タッチアプリケーションの開発(R01AN4934)
- QE と FIT を使用した静電容量タッチアプリケーションの開発(R01AN4516)
- RL78 ファミリ QE と SIS を使用した静電容量タッチアプリケーションの開発(R01AN5512)

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ

https://www.renesas.com/

静電容量センサユニット関連ページ

https://www.renesas.com/rssk-touch-ra2l1

https://www.renesas.com/qe-capacitive-touch

お問い合わせ

https://www.renesas.com/support

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	Jun.12.23	-	新規発行
2.00	Aug.31.23		文書全体の再構成 4. IIR フィルタを追加 1.1 のフォルダ構造に IIR フィルタ関連の項目を追加。 図 2-1 を修正 2.2 のファイル構造に IIR フィルタ関連の項目を追加 表 3-1 の係数データ型に関する注記を修正 3.5.4 誤記修正 IIR フィルタ関連項目をセクションに追加
3.00	Nov.30.23	4 5 6 7 8 9 10 13 19 24 24 26 28 28 31 45 49 56 68	1.1 のフォルダ構造にメディアンフィルタ関連の項目を追加 表 1-1 動作確認条件のバージョンを更新 2. 追加予定機能についての記載を削除 表 2-1 コンポーネント一覧のバージョンを更新 2.2 のファイル構造にメディアンフィルタ関連の項目を追加、フォルダの説明記載を追加 表 2-2 各フィルタの有効、無効定義を追加 表 2-3 メディアンフィルタを追加 表 2-12 メディアンフィルタを追加 図 2-5 メディアンフィルタを追加 2.4.6 メディアンフィルタを追加 2.4.6 メディアンフィルタ用初期設定 API を追加 2.4.7 メディアンフィルタについて記載を追加 表 2-13 データサイズ及び増分量を更新 表 2-14 フィルタ処理の実行時間を更新 3.3.1 誤記修正 4.3.1 誤記修正 表 4.4 誤記修正 5. メディアンフィルタを追加 6. 章の構成を変更
4.00	Jun.30.24	-	全面改訂

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 静電気対策
   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。
2. 電源投入時の処置
   電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。
5. クロックについて
   リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子(または外部発振回路)を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子(または外部発振回路)を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
6. 入力端子の印加波形
   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}(Max.)から V_{IH}(Min.)までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}(Max.)から V_{IH}(Min.)までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス(予約領域)のアクセス禁止
   リザーブアドレス(予約領域)のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス(予約領域)があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
8. 製品間の相違について
   型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。回路、ソフトウェアおよびこれらに関連する情報を使用する場合、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害(お客様または第三者いずれに生じた損害も含みます。以下同じです。)に関し、当社は、一切その責任を負いません。
2. 当社製品または本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
6. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等

   高品質水準：輸送機器(自動車、電車、船舶等)、交通制御(信号)、大規模通信機器、金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム(生命維持装置、人体に埋め込み使用するもの等)、もしくは多大な物的損害を発生させるおそれのある機器・システム(宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等)に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害(当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。)から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為(「脆弱性問題」といいます。)によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報(データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等)をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
10. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
13. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

(Rev.5.0-1 2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24(豊洲フォレシア)

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。