EPSON
EXCEED YOUR VISION

FSA ライブラリ vol.2
マニュアル

加速度応用システムライブラリ

セイコーエプソン株式会社

Rev.1.1

本資料のご使用につきましては、次の点にご留意願います。

本資料の内容については、予告無く変更することがあります。

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

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

3. 特性値の数値の大小は、数直線上の大小関係で表しています。

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

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

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

目　次

1. 概要

本ライブラリは、加速度、地磁気を用いた以下に示す機能を提供します。

- **歩数計測／移動状態推定／消費カロリー計測**
- **移動ベクトル計測**
- **地磁気方位角計測**
- **タッピング検出**
- **重力方向軸検出**

本ライブラリは、いくつかの必須ライブラリとオプションライブラリから構成されます。ライブラリの全体構成を図 1-1 に示します。

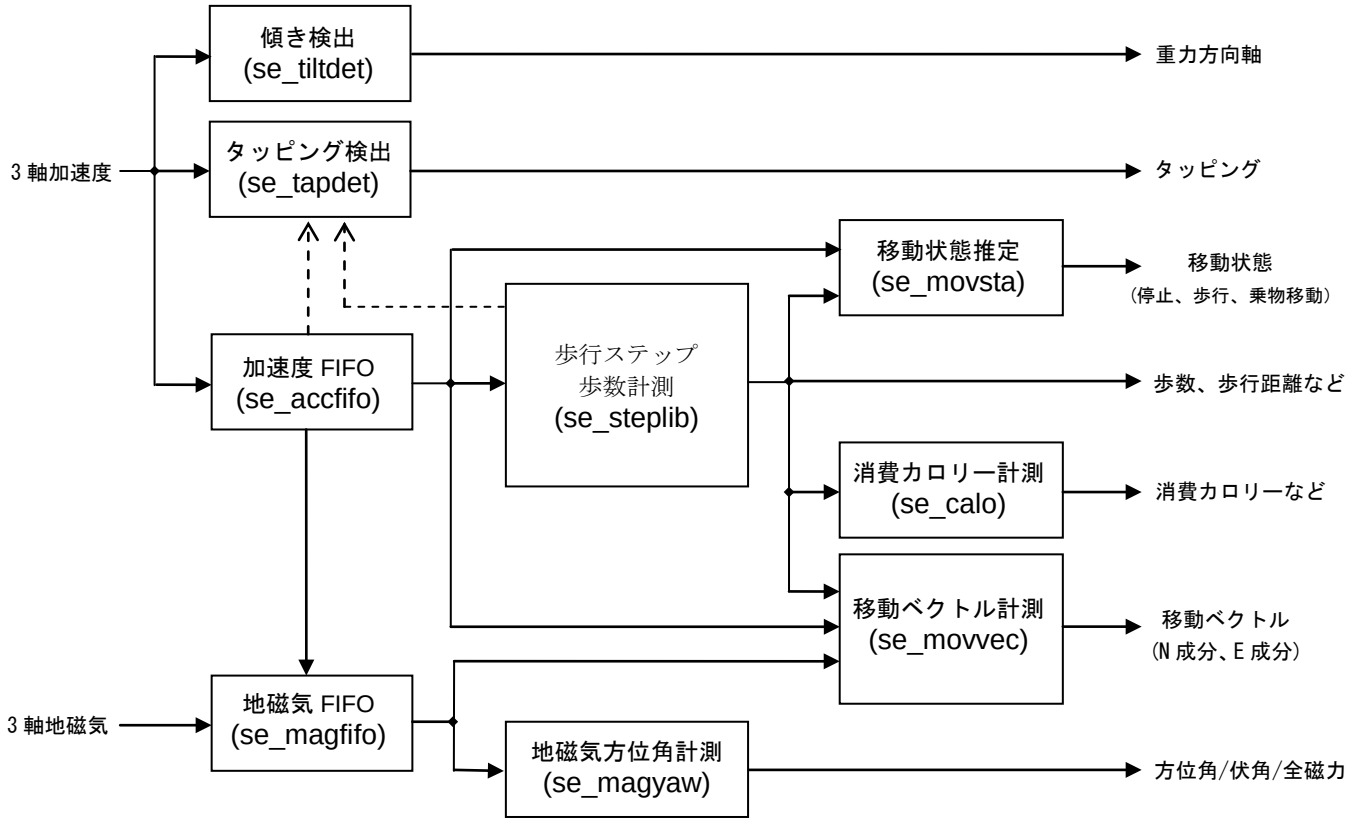

図 1-1　ライブラリ構成

1.1 歩数計測／移動状態推定／消費カロリー計測

歩数計測／移動状態推定／消費カロリー計測機能は25Hzあるいは12.5Hzの3軸加速度を入力とし、ユーザーの歩行ステップや歩数、移動状態、消費カロリーなどを計測するFSAライブラリです。

下記に示すような特徴を備えています。

1）リスト装着を想定した歩数計測に対応
一般的な歩数計に要求される胸、腰等の体幹部や、カバン、ズボンポケット等の装着に対応するだけでなく、リスト装着による歩数計測にも対応します。具体的には、歩行時に腕を前後に振った場合でも振らない場合でも、アルゴリズムが状態を自動判別し、正確に歩数な計測を行います。

2）歩行以外の振動による誤検出を抑圧
クルマなどの歩行以外の振動、水平方向の振動、腕の上げ下ろしによる姿勢変化を伴う振動などによる歩行ステップの誤検出を抑圧するアルゴリズムを採用しています。

3）歩行距離キャリブレーション
歩行距離推定精度を向上させるキャリブレーション機能をサポートしています。

4）移動状態推定機能
se_movstaライブラリを追加することで、ユーザーの移動状態を推定することができます。推定可能な状態は停止、歩行、乗物移動（電車、バス、クルマ等に乗車中の状態）の3状態です。

5）消費カロリー計測
se_caloライブラリを追加することで、消費カロリー、エクササイズ、脂肪燃焼量、METsの計測機能をサポートします。

上記の歩数計測関連機能のオブジェクトファイルを表1-1に示します。

表 1-1 歩数計測関連機能のオブジェクトファイル

パッケージ名	オブジェクトファイル	内容
sensfifo	se_accfifo.o se_accfifo_fsa.o fsa_symbols_accfifo.def	加速度データのFIFOバッファを構成し、加速度値の正規化やLPFフィルタ処理を行います。
stepdet	se_steplib.o se_steplib_fsa.o fsa_symbols_steplib.def	se_accfifoから加速度データを入力して、歩行ステップ検出、歩行ステップカウント、歩行距離推定、持ち変え変更検出等の機能を提供します。
movsta	se_movsta.o	se_steplibのアドオンライブラリです。移動状態推定機能を提供します。移動状態推定を行わない場合は不要です。
calo	se_calo.o se_calo_fsa.o fsa_symbols_calo.def	se_steplibのアドオンライブラリです。消費カロリー、エクササイズ、脂肪燃焼量を算出します。これらの算出を行わない場合は不要です。

1.2 移動ベクトル計測

移動ベクトル計測は 25Hz あるいは 12.5Hz の加速度および地磁気により、ユーザーの移動ベクトル（North 成分、East 成分）を算出します。

下記に示すような特徴を備えています。

1）任意の端末姿勢に対応
移動ベクトルを算出する se_movvec ライブラリは、端末座標におけるユーザーの移動方向を加速度から自動推定する機能を内蔵しており、任意の端末姿勢（装着位置）で移動ベクトルを算出できます。
＊但し、歩行時の体動を正確に計測できる胸や腰装着に比べ、バッグ、ズボンポケット、リスト装着では、十分な精度が得られない場合があります。

2）移動方位補正に対応
移動方位を補正するための外部からの補正角入力に対応しています。

上記の移動ベクトル計測に必要なオブジェクトファイルを表 1-2 に示します。

表 1-2　移動ベクトル計測のオブジェクトファイル

パッケージ名	オブジェクトファイル	内容
sensfifo	se_accfifo.o se_accfifo_fsa.o fsa_symbols_accfifo.def	加速度データの FIFO バッファを構成し、加速度値の正規化や LPF フィルタ処理を行います。
	se_magfifo.o	地磁気データの FIFO バッファを構成し、LPF フィルタ処理を行います。また、加速度により地磁気の座標をボディ座標からグローバル座標に変換し、その推定成分を抽出します。
stepdet	se_steplib.o se_steplib_fsa.o fsa_symbols_steplib.def	se_accfifo から加速度データを入力して、歩行ステップ検出、歩行ステップカウント、歩行距離推定、持ち変え変更検出等の機能を提供します。
calo	se_movvec.o se_movvec_fsa.o fsa_symbols_movvec.def	加速度から自動推定した端末座標（ENU 座標）と移動方向の相対角、地磁気から算出した端末座標の方位により、se_steplib が出力する移動距離を N 成分と E 成分に分解します。

1.3 地磁気方位角計測

地磁気方位角計測は 25Hz あるいは 12.5Hz の加速度および地磁気により、方位角、伏角および全磁力を算出します。

方位角計測に必要なオブジェクトファイルを表 1-3 に示します。

表 1-3 方位角計測のオブジェクトファイル

パッケージ名	オブジェクトファイル	内容
sensfifo	se_accfifo.o se_accfifo_fsa.o fsa_symbols_accfifo.def	加速度データの FIFO バッファを構成し、加速度値の正規化や LPF フィルタ処理を行います。
	se_magfifo.o	地磁気データの FIFO バッファを構成し、LPF フィルタ処理を行います。また、加速度により地磁気の座標をボディ座標からグローバル座標に変換し、その推定成分を抽出します。
magyaw	se_magyaw.o	方位角、伏角、全磁力を算出します。
fsatrig	fsatrig.o fsatrig_fsa.o fsa_symbols_fsatrig.def	方位角算出、伏角算出に必要な atan2 関数のライブラリです。FSA による atan2 関数は、FSA_package_vol.1 の fsatrig ライブラリに含まれています。
fsasqrt	fsasqrt.o fsasqrt_fsa.o fsa_symbols_fsasqrt.def	全磁力の算出に必要な FsaSumSqrt16 関数のオブジェクトです。FsaSumSqrt16 関数は、FSA_package_vol.1 の fsasqrt ライブラリに含まれています。

1.4 傾き（重力方向軸）検出

25Hz あるいは 12.5Hz の加速度から端末の重力方向軸を検出します。具体的には、XY 軸、YZ 軸、ZX 軸の各ペアのどちらの軸のどの方向がより下向きかを検出します。

傾き検出に必要なオブジェクトファイルを表 1-4 に示します。

表 1-4 縦横検出のオブジェクトファイル

パッケージ名	オブジェクトファイル	内容
tiltdet	se_tiltdet.o	XY 軸、YZ 軸、ZX 軸の各ペアのどちらの軸のどの方向がより下向きかを検出します。

1.5 タッピング検出

25Hz あるいは 12.5Hz の加速度からタッピングを検出します。

タッピング検出に必要なオブジェクトファイルを表 1-5 に示します。

表 1-5 タッピング検出のオブジェクトファイル

パッケージ名	オブジェクトファイル	内容
tapdet	se_tapdet.o	タッピンングを検出します。
sensfifo	se_accfifo.o se_accfifo_fsa.o fsa_symbols_accfifo.def	se_accfifo に実装されている一部の FSA 関数を se_tapdet からコールするため、本オブジェクトのリンクが必要になります。
stepdet	se_steplib.o se_steplib_fsa.o fsa_symbols_steplib.def	se_steplib に実装されている一部の FSA 関数を se_tapdet からコールするため、本オブジェクトのリンクが必要になります。

2. API 関数仕様－se_accfifo

se_ACCFIFOInit

インクルード

```
#include “se_sensfifo.h”
```

形式　`void se_ACCFIFOInit(unsigned short ui16ScaleFactor)`

引数

(in)*ui16ScaleFactor*　　加速度のスケールファクタ
1G に相当する加速度の数値を指定します。

戻り値

なし

説明

加速度データの FIFO バッファの初期化および設定を行います。この関数を実行すると FIFO バッファのデータはすべて 0 クリアされます。

se_ACCFIFOUpdate

インクルード

```
#include “se_sensfifo.h”
```

形式　`void se_ACCFIFOUpdate(FSAREG *pFsaReg, short ai16AccData[3])`

引数

(in)*pFsaReg*　　FSA レジスタ構造体のポインタ
(in)*ai16AccData*　　入力加速度データの配列

戻り値

なし

説明

加速度データの FIFO バッファに、ai16AccData[3]で渡されたデータを入力します。入力するデータのサンプリング周波数は 25Hz としてください。ai16AccData[3]のデータの並びは、端末座標を ENU 座標と定義し、ai16AccData[0]に X 軸、ai16AccData[1]に Y 軸、ai16AccData[2]に Z 軸のデータをセットします。

3. API 関数仕様－se_magfifo

se_MAGFIFOInit

インクルード

```
#include “se_sensfifo.h”
```

形式　`void se_MAGFIFOInit(void)`

引数

なし

戻り値

なし

説明

地磁気データの FIFO バッファの初期化および設定を行います。この関数を実行すると FIFO バッファのデータはすべて 0 クリアされます。

se_MAGFIFOUpdate

インクルード

```
#include “se_sensfifo.h”
```

形式　`void se_MAGFIFOUpdate(FSAREG *pFsaReg, short ai16MagData[3])`

引数

(in)*pFsaReg*	FSA レジスタ構造体のポインタ
(in)*ai16MagData*	入力地磁気データの配列

戻り値

なし

説明

地磁気データの FIFO バッファに、ai16MagData[3]で渡されたデータを入力します。入力するデータのサンプリング周波数は 25Hz としてください。ai16MagData[3]のデータの並びは、端末座標を ENU 座標と定義し、ai16MagData[0]に X 軸、ai16MagData[1]に Y 軸、ai16MagData[2]に Z 軸のデータをセットしてください。

4. API 関数仕様－se_steplib

se_STEPInit

インクルード

```
#include “se_steplib.h”
```

形式　`void se_STEPInit(void *pParam1, void *pParam2)`

引数

(in)*pParam1*　　通常は NULL ポインタを渡してください。
(in)*pParam2*　　通常は NULL ポインタを渡してください。

戻り値

なし

説明

ライブラリの内部バッファ、変数の初期化を行います。

se_STEPDetect

インクルード

```
#include “se_steplib.h”
```

形式　`int se_STEPDetect(FSAREG *pFsaReg)`

引数

(in)*pFsaReg*　　FSA レジスタ構造体のポインタ

戻り値

歩行ステップを検出すると、検出した数に応じて、1 あるいは 2 が返されます。そうでない場合、0 が返されます。

説明

加速度データの FIFO バッファからデータを入力し、歩行ステップを検出します。この関数は、se_ACCFIFOUpdate 関数と同期して 25Hz の周期でコールしてください。

se_STEPGetWalkStep

インクルード

```
#include “se_steplib.h”
```

形式　unsigned short se_STEPGetWalkStep(void)

引数

なし

戻り値

後述する se_STEPFlushStep 関数をコールしてから現在までに検出した歩数が返されます。

説明

最後に se_STEPFlushStep 関数をコールしてから現在までの累計の歩数を取得する関数です。コールするタイミングに制約はなく、取得したい時間間隔でコールできます。

se_STEPGetWalkTime

インクルード

```
#include “se_steplib.h”
```

形式　unsigned long se_STEPGetWalkTime(void)

引数

なし

戻り値

後述する se_STEPFlushStep 関数をコールしてから現在までの歩行時間（単位：1/25 秒）が返されます。

説明

最後に se_STEPFlushStep 関数をコールしてから現在までの累計の歩行時間を取得する関数です。コールするタイミングに制約はなく、取得したい時間間隔でコールできます。

se_STEPGetWalkDist

インクルード

```
#include “se_steplib.h”
```

形式　unsigned long se_STEPGetWalkDist(void)

引数

なし

戻り値

後述する se_STEPFlushStep 関数をコールしてから現在までの歩行距離（単位：1/256 メートル）が返されます。

説明

最後に se_STEPFlushStep 関数をコールしてから現在までの累計の歩行距離を取得する関数です。コールするタイミングに制約はなく、取得したい時間間隔でコールできます。

se_STEPFlushStep

インクルード
#include “se_steplib.h”

形式　void se_STEPFlushStep(void)

引数
なし

戻り値
なし

説明
内部変数の歩数、歩行時間、歩行距離をゼロクリアします。

se_STEPGetWalkSpeed

インクルード
#include “se_steplib.h”

形式　unsigned short se_STEPGetWalkSpeed(void)

引数
なし

戻り値
歩行速度（単位：1/256 メートル／秒）が返されます。

説明
コールした時点の歩行速度が返されます。歩行状態でない場合は、0 が返ります。コールするタイミングに制約はなく、任意のタイミングで歩行速度を取得できます。

se_STEPInitCalib

インクルード
#include “se_steplib.h”

形式　void se_STEPInitCalib(unsigned long ui32CurrDist)

引数

(in)*ui32CurrDist*	歩行距離キャリブレーション開始時点の歩行距離 （単位：1/256 メートル）

戻り値
なし

説明
歩行距離キャリブレーションは、開始時点で現在の歩行距離をセットし、終了時点で開始時点からの真の移動距離をセットすることで行われます。本関数は、歩行距離キャリブレーション開始時に実行する関数で、ui32CurrDist には、開始時点の歩行距離をセットします。基本的には、se_STEPGetWalkDist 関数の戻り値をそのままセットしてください。キャリブレーションは歩行状態であっても開始することができます。

se_STEPCalibDist

インクルード

```
#include “se_steplib.h”
```

形式　`int se_STEPCalibDist(unsigned long ui32CurrDist, unsigned long ui32ExpectDist)`

引数

(in)*ui32CurrDist*	歩行距離キャリブレーション終了時点の歩行距離 （単位：1/256 メートル）
(in)*ui32ExpectDist*	歩行距離キャリブレーション終了時点の真の歩行距離 （単位：1/256 メートル）

戻り値

キャリブレーションが成功すると 1、そうでない場合は 0 が返されます。

説明

歩行距離キャリブレーションの終了時にコールする関数です。se_STEPGetWalkDist 関数により取得した終了時点の歩行距離を ui32CurrDist にセットし、ui32ExpectDist にキャリブレーション開始時点からの真の歩行距離をセットします。関数内部で、計測された歩行距離と与えられた真の歩行距離を用いて、歩行距離補正係数を算出します。補正係数の精度を確保するため、キャリブレーションを行う距離は 100m 以上としてください。なお、キャリブレーションは歩行状態であっても終了することができます。

se_STEPGetCalibCoef

インクルード

```
#include “se_steplib.h”
```

形式　`unsigned short se_STEPGetCalibCoef(void)`

引数

なし

戻り値

歩行距離補正係数が返されます。補正係数は小数点以下 14 ビットの数値です。

説明

歩行距離キャリブレーションによって算出された補正係数を取得します。

se_STEPSetCalibCoef

インクルード

```
#include “se_steplib.h”
```

形式　`void se_STEPSetCalibCoef(unsigned short ui16CalibCoef)`

引数

(in)*ui16CalibCoef*　歩行距離補正係数。小数点以下 14 ビットの数値で、設定範囲は 0 以上 2.0 未満です。但し、0 をセットした場合、1.0 と等価です。

戻り値

なし

説明

歩行距離計数を直接設定する関数です。se_STEPGetCalibCoef 関数により取得した補正係数をライブラリ初期化後に再セットする場合や外部で算出した補正係数をセットする場合において、本関数により補正係数を設定できます。

se_STEPGetWalkState

インクルード

```
#include “se_steplib.h”
```

形式　`int se_STEPGetWalkState(void)`

引数

なし

戻り値

下記に示す歩行検出状態が返されます。

戻り値	歩行検出状態
0	歩行未検出
1	歩行仮検出（有効歩数判定期間）
2	歩行検出

説明

歩行検出状態を取得します。任意のタイミングでコールできます。

se_STEPGetTiltState

インクルード

#include “se_steplib.h”

形式　int se_STEPGetTiltState(void)

引数

なし

戻り値

端末の装着状態に変化があった場合に 1 が返されます。そうでない場合は 0 が返されます。

説明

端末の装着状態に変化があったと考えられる有意な姿勢変化を検出します。

5. API 関数仕様－se_movsta

se_MSTAInit

インクルード
#include “se_steplib.h”

形式　void se_MSTAInit(void *pParam)

引数
(in)*pParam*　　通常は NULL ポインタを渡してください。

戻り値
なし

説明
ライブラリの内部バッファ、変数の初期化を行います。

se_MSTAEst

インクルード
#include “se_steplib.h”

形式　int se_MSTAEst(FSAREG *pFsaReg)

引数
(in)*pFsaReg*　　FSA レジスタ構造体のポインタ

戻り値
推定移動状態に変化があった場合に 1 が返されます。そうでない場合は 0 が返されます。

説明
加速度データの FIFO バッファからデータを入力し、移動状態を推定します。この関数は、se_ACCFIFOUpdate 関数と同期して 25Hz の周期でコールしてください。

se_MSTAGetState

インクルード

```
#include “se_steplib.h”
```

形式　`int se_MSTAGetState(void)`

引数

なし

戻り値

下記に示す移動状態推定結果が返されます。

戻り値	推定移動状態
0	停止
1	乗物移動
2	歩行移動

説明

推定した移動状態を取得する関数です。基本的には、se_MSTAEst 関数の戻り値が 1 のとき、コールして移動状態を取得します。但し、任意のタイミングでコールしてもコールした時点の移動状態を取得できます。

6. API 関数仕様－se_calo

se_CALReset

インクルード

```
#include “se_calo.h”
```

形式　`void se_CALReset(SE_CALOINFO *pCaloInfo)`

引数

(in)*pCaloInfo*　　SE_CALOINFO 構造体のポインタ

戻り値

なし

説明

SE_CALOINFO 構造体の初期化を行います。SE_CALOINFO 構造体の定義は下記の通りです。

```
typedef struct {
  unsigned char   ui8Weight;
  unsigned char   ui8AgeSex;
  short           i16Mets;
  long            i32Exercise;
  long            i32Calorie;
  long            i32Fat;
} SE_CALOINFO;
```

表 6-1　SE_CALOINFO 構造体のメンバー変数

メンバー変数	説　　明
ui8Weight	消費カロリー、脂肪燃焼量の計算に使用するユーザーの体重が格納されます。単位は kg の整数値です。
ui8AgeSex	ユーザーの年齢および性別を格納する変数です。下位 7 ビットに年齢を格納し、上位 1 ビットに性別フラグを格納します。0：女性、1：男性です。但し、現状では消費カロリー等の計算に使用していません。
i16Mets	METs が格納されます。単位は METs、小数点以下 8 ビットの数値です。
i32Exercise	エクササイズが格納されます。小数点以下 8 ビットの数値です。単位はありません。
i32Calorie	消費カロリーが格納されます。単位はキロカロリー、小数点以下 8 ビットの数値です。
i32Fat	脂肪燃焼量が格納されます。単位は g、小数点以下 8 ビットの数値です。

本関数を実行すると、METs、エクササイズ、消費カロリー、脂肪燃焼量はゼロ初期化され、体重、年齢、性別は、各々60Kg、30 才、男性に初期化されます。

なお、SE_CALOINFO 構造体は、FSA がアクセス可能なメモリ上に配置してください。

se_CALInit

インクルード

```
#include “se_calo.h”
```

形式　`void se_CALInit(SE_CALOINFO *pCaloInfo)`

引数

(in)*pCaloInfo*　　SE_CALOINFO 構造体のポインタ

戻り値

なし

説明

SE_CALOINFO 構造体のメンバーのうち、METs、エクササイズ、消費カロリー、脂肪燃焼量をゼロ初期化します。体重、年齢、性別は初期化されません。

se_CALSetParam

インクルード

```
#include “se_calo.h”
```

形式　`void se_CALSetParam (SE_CALOINFO *pCaloInfo, int iWeight, int iAge, int iSex)`

引数

(in)*pCaloInfo*　　SE_CALOINFO 構造体のポインタ
(in)*iWeight*　　ユーザーの体重（設定範囲：0～255、単位：kg）
(in)*iAge*　　ユーザーの年齢（設定範囲：0～127、単位：才）
(in)*iSex*　　ユーザーの性別（0：女性、1：男性）

戻り値

なし

説明

ユーザーの体重、年齢、性別を設定する関数です。

se_CALUpdate

インクルード

```
#include “se_calo.h”
```

形式

```
void se_CALUpdate(
        FSAREG *pFsaReg,
        SE_CALOINFO *pCaloInfo,
        short i16Dist,
        short i16Time
)
```

引数

(in)*pFsaReg*	FSA レジスタ構造体のポインタ
(in)*pCaloInfo*	SE_CALOINFO 構造体のポインタ
(in)*i16Dist*	移動距離（単位：1/256 メートル）
(in)*i16Time*	経過時間（単位：1/25 秒）

戻り値

なし

説明

与えられた移動距離 i16Dist とその移動に要した時間 i16Time から METs を計算すると共に、エクササイズ、消費カロリー、脂肪燃焼量を計算し、SE_CALOINFO 構造体の該当するメンバー変数に加算します。

se_CALGetMETs

インクルード

```
#include “se_calo.h”
```

形式

```
unsigned short se_CALGetMETs(SE_CALOINFO *pCaloInfo)
```

引数

(in)*pCaloInfo*	SE_CALOINFO 構造体のポインタ

戻り値

SE_CALOINFO 構造体の METs が小数点以下 8 ビットの数値で返されます。

説明

SE_CALOINFO 構造体から METs（ui16Mets）を取得します。

se_CALGetExercise

インクルード
#include “se_calo.h”

形式　unsigned long se_CALGetExercise(SE_CALOINFO *pCaloInfo)

引数
(in)*pCaloInfo*　　SE_CALOINFO 構造体のポインタ

戻り値
SE_CALOINFO 構造体のエクササイズが小数点以下 8 ビットの数値で返されます。

説明
SE_CALOINFO 構造体からエクササイズ（i32Exercise）を取得します。

se_CALGetCalorie

インクルード
#include “se_calo.h”

形式　unsigned long se_CALGetCalorie(SE_CALOINFO *pCaloInfo)

引数
(in)*pCaloInfo*　　SE_CALOINFO 構造体のポインタ

戻り値
SE_CALOINFO 構造体の消費カロリーが小数点以下 8 ビットの数値で返されます。

説明
SE_CALOINFO 構造体から消費カロリー（i32Calorie）を取得します。

se_CALGetFat

インクルード
#include “se_calo.h”

形式　unsigned long se_CALGetFat(SE_CALOINFO *pCaloInfo)

引数
(in)*pCaloInfo*　　SE_CALOINFO 構造体のポインタ

戻り値
SE_CALOINFO 構造体の脂肪燃焼量が小数点以下 8 ビットの数値で返されます。

説明
SE_CALOINFO 構造体から脂肪燃焼量（i32Fat）を取得します。

7. API 関数仕様－se_movvec

se_MVECInit

インクルード

```
#include “se_movvec.h”
```

形式　`void se_MVECInit(void *pParam)`

引数

(in)*pParam*　　通常は NULL ポインタを渡してください。

戻り値

なし

説明

ライブラリの内部バッファ、変数の初期化を行います。

se_MVECCalcDir

インクルード

```
#include “se_movvec.h”
```

形式　`void se_MVECCalcDir(FSAREG *pFsaReg)`

引数

(in)*pFsaReg*　　FSA レジスタ構造体のポインタ

戻り値

なし

説明

加速度データの FIFO バッファからデータを入力し、端末座標における移動方向を推定し内部変数に格納します。この関数は、se_ACCFIFOUpdate 関数と同期して 25Hz の周期でコールしてください。基本的には、この関数で推定された移動方向と地磁気から算出した方位角により、後述する se_CalcMovVec 関数にて、ユーザーの移動方位が決定されます。
なお、se_MVECInit 関数を実行した直後の移動方向は、0 度（ENU 端末座標の Y 軸方向）になっています。

se_MVECCalcMovVec

インクルード

```
#include “se_movvec.h”
```

形式　`void se_MVECCalcMovVec(FSAREG *pFsaReg, short i16MovDist, short ai16MovVec[2])`

引数

(in)*pFsaReg*	FSA レジスタ構造体のポインタ
(in)*i16MovDist*	移動距離
(out)*ai16MovVec[2]*	移動ベクトル（North 成分、East 成分）

戻り値

なし

説明

地磁気データの FIFO バッファからデータを入力し、端末座標における方位角を算出します。さらに se_MVECDirectionEst 関数で算出した端末座標における移動方向を用いて、与えられた移動距離（i16MovDist）を、North 成分と East 成分に分解し、その移動ベクトルを ai16MovVec[2]に格納します。

se_MVECSetDeclination

インクルード

```
#include “se_movvec.h”
```

形式　`void se_MVECSetDeclination(short i16Ang)`

引数

(in)*i16Ang*	偏角（単位：1/128 度）

戻り値

なし

説明

地磁気偏角を設定する関数です。

se_MVECSetMovAngOnBody

インクルード

```
#include “se_movvec.h”
```

形式　`void se_MVECSetMovAngOnBody(short i16Ang)`

引数

(in)*i16An*	相対移動方位角（小数点以下 14 ビットで、-180°～+180°を-2.0～+2.0 に正規化した値）

戻り値

なし

説明

水平面に投射した端末座標（ENU 座標）の Y 軸を基準に、移動方向を相対角度で指定する関数です。プラス Y 軸を 12 時としたとき、例えば、移動方向を 3 時方向とする場合は+90 度、9 時方向とする場合は-90 度を、i16Ang にセットします。
この関数をコールすると、移動方向を自動推定する機能はオフされます。再び移動方向自動推定機能をオンする場合は、se_MVECSetDirEstEn 関数によりオンできます。

se_MVECSetCorrectAng

インクルード

```
#include “se_movvec.h”
```

形式　`void se_MVECSetCorrectAng(short i16Ang)`

引数

(in)*i16Ang*	補正角（小数点以下 14 ビットで、-180°～+180°を-2.0～+2.0 に正規化した値）

戻り値

なし

説明

移動方位を補正する関数です。与えられた補正角（i16Ang）に従って se_MVECCalcMovVec 関数により算出される移動ベクトルが回転します。具体的には、i16Ang が正の値の時、移動ベクトルは時計回りに回転します。
補正角は i16Ang で与えられた角度がそのままセットされるわけではなく、本関数をコールした時点の補正角に i16Ang の角度が加算されます。したがって、i16Ang を 0 として本関数をコールした場合、補正角に 0 がセットされるわけではなく、現在の補正角を変更しないという処理になります。補正角を直ちに 0 クリアする場合は、後述する se_MVECResetCorrectAng 関数を使用してください。

se_MVECResetCorrectAng

インクルード
#include “se_movvec.h”

形式　void se_MVECResetCorrectAng()

引数
なし

戻り値
なし

説明
移動方位の補正角を 0 クリアします。

se_MVECSetDirEstEn

インクルード
#include “se_movvec.h”

形式　void se_MVECSetDirEstEn(int iEnable)

引数
(in)*iEnable*　　移動方向推定の制御フラグ（Disable：0、Enable：1）

戻り値
なし

説明
移動方向推定の更新を制御する関数です。iEnable に 0 をセットして本関数をコールすると、se_MVECDirectionEst 関数により、移動方向の推定が行われていても、se_MVECCalcMovVec 関数で使用する移動方向の更新が停止します。

se_MVECSetMagDirEn

インクルード
#include “se_movvec.h”

形式　void se_MVECSetMagDirEn(int iEnable)

引数
(in)*iEnable*　　**地磁気方位角**の制御フラグ（Disable：0、Enable：1）

戻り値
なし

説明
地磁気方位角の更新を制御する関数です。iEnable に 0 をセットして本関数をコールすると、地磁気方位角の更新が停止します。

se_MVECGetTurnDet

インクルード

```
#include “se_movvec.h”
```

形式　`int se_MVECGetTurnDet()`

引数

なし

戻り値

ユーザーの移動方向（移動ベクトル）に有意な変化があった時、1 が返されます。それ以外は 0 が返されます。

説明

ユーザーが移動中（歩行中）のとき、移動方向の有意な変化を検出する関数です。ユーザーが移動中でないときは、常に 0 が返されます。地磁気の乱れによって、誤検出される場合があります。

8. API 関数仕様－se_magyaw

se_CalcMagYaw

インクルード

```
#include “se_magyaw.h”
```

形式　`void se_CalcMagYaw(FSAREG *pFsaReg, short *pi16Yaw, short *pi16Inc)`

引数

(in)*pFsaReg*	FSA レジスタ構造体のポインタ
(out)*pi16Yaw*	方位角が格納されるポインタ（単位：1/128 度）
(out)*pi16Inc*	伏角が格納されるポインタ（単位：1/128 度）

戻り値

なし

説明

地磁気データの FIFO バッファからデータを入力し、端末座標における方位角（-180 度～+180 度）および伏角（-180 度～+180 度）を算出し、各々*pi16Yaw、*pi16Inc に格納します。pi16Inc には NULL ポインタを指定でき、その場合は、伏角の算出を行いません。

se_CalcMagIntensity

インクルード

```
#include “se_magyaw.h”
```

形式　`short se_CalcMagIntensity(FSAREG *pFsaReg)`

引数

(in)*pFsaReg*	FSA レジスタ構造体のポインタ

戻り値

全磁力が返されます。

説明

地磁気データの FIFO バッファからデータを入力し、全磁力を算出して戻り値として返します。全磁力のスケールは、se_MAGFIFOUpdate 関数に入力された地磁気データのスケールと同じです。

9. API 関数仕様－se_tiltdet

se_TILTInit

インクルード

```
#include “se_tiltdet.h”
```

形式　`void se_TILTInit(unsigned short ui16ScaleFactor, void *pParam)`

引数

(in)*ui16ScaleFactor*	加速度のスケールファクタ 1G に相当する加速度の数値を指定します。
(in)*pParam*	通常は NULL ポインタを渡してください。

戻り値

なし

説明

ライブラリの内部バッファ、変数の初期化を行います。

se_TILTDetect

インクルード

```
#include “se_tiltdet.h”
```

形式　`int se_TILTDetect(short ai16AccData[3])`

引数

(in)*ai16AccData*　　入力加速度データの配列

戻り値

XY 軸、YZ 軸、ZX 軸の各ペアの重力方向軸の変化の有無を返します。戻り値の詳細は、表 8-1 の通りです。但し、同時に複数の軸変化が発生した場合、それぞれの値を加算した値を返します。

表 9-1 se_TILTDetect 関数の戻り値

値	検出した状態
0	未検出
1	XY 軸で重力方向軸の変化を検出
2	YZ 軸で重力方向軸の変化を検出
4	XZ 軸で重力方向軸の変化を検出

説明

ai16AccData[3]から入力した 25Hz あるいは 12.5Hz の加速度により端末の重力方向軸の変化を検出します。具体的な重力方向軸の変化を、図 8-1 を用いて説明します。

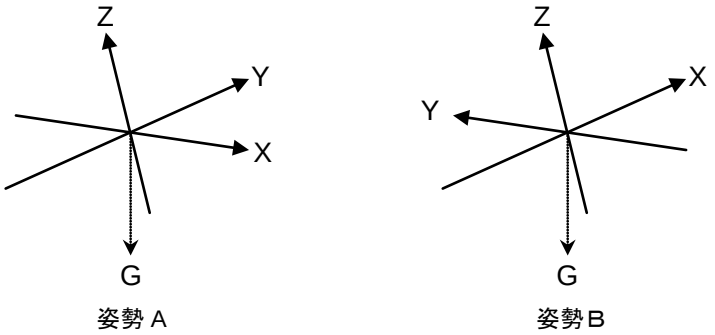

図 8-1 XY 軸における重力方向軸の変化

図 8-1 において、破線の矢印は重力方向（鉛直方向）を示します。XY 軸に着目すると、姿勢 A は、X 軸はほぼ水平で Y 軸が若干傾いている状態を表しています。つまり、Y 軸により大きな重力方向成分の加速度が観測される状態で、XY 軸のみの比較でみると、Y 軸が重力方向軸になります。
この姿勢 A から Z 軸を回転軸として反時計回りに 90° 回転させた状態が姿勢 B です。姿勢 B では、Y 軸がほぼ水平、X 軸が若干傾きますので、XY 軸のみの比較でみると、X 軸が重力方向軸になります。
重量方向軸の変化とは、このように姿勢 A から姿勢 B への変化またはその逆の変化を指し、se_TILTDetect 関数は、その変化を XY 軸、YZ 軸、ZX 軸の各ペアについて検出し、戻り値として返します。但し、各軸の重力加速度成分の差が 1/4G 未満の場合は、変化として検出されません。別の言い方をすれば、図 8-1 において、XY 平面が水平面に対して 15° 以上傾いていない場合、姿勢 A から姿勢 B に変化しても、軸変化は検出されません。

なお、具体的にどの軸が重量方向軸か取得する場合、後述する se_TILTGetXYState 関数等を使用してください。

se_TILTGetXYState

インクルード

```
#include “se_tiltdet.h”
```

形式　`int se_TILTGetXYState ()`

引数

なし

戻り値

表 9-2 se_TILTXYState 関数の戻り値

値	検出した状態
0	未検出
+1	X 軸のプラス方向が鉛直方向
-1	X 軸のマイナス方向が鉛直方向
+2	Y 軸のプラス方向が鉛直方向
-2	Y 軸のマイナス方向が鉛直方向

説明

X 軸、Y 軸のうち、どちらの軸のどの方向が鉛直方向を向いているを戻り値として返します。

se_TILTGetYZState

インクルード

```
#include “se_tiltdet.h”
```

形式　`int se_TILTYZDetect()`

引数

なし

戻り値

表 9-3 se_TILTGetYZState 関数の戻り値

値	検出した状態
0	未検出
+1	Y 軸のプラス方向が鉛直方向
-1	Y 軸のマイナス方向が鉛直方向
+2	Z 軸のプラス方向が鉛直方向
-2	Z 軸のマイナス方向が鉛直方向

説明

Y 軸、Z 軸のうち、どちらの軸のどの方向が鉛直方向を向いているを戻り値として返します。

se_TILTGetZXState

インクルード

```
#include “se_tiltdet.h”
```

形式　`int se_TILTGetZXDetect()`

引数

なし

戻り値

表 9-4 se_TILTGetZXState 関数の戻り値

値	検出した状態
0	未検出
+1	Z 軸のプラス方向が鉛直方向
-1	Z 軸のマイナス方向が鉛直方向
+2	X 軸のプラス方向が鉛直方向
-2	X 軸のマイナス方向が鉛直方向

説明

Z 軸、X 軸のうち、どちらの軸のどの方向が鉛直方向を向いているを戻り値として返します。

10. API 関数仕様－se_tapdet

se_TAPInit

インクルード

```
#include “se_tapdet.h”
```

形式　`void se_TAPInit(unsigned short ui16ScaleFactor, void *pParam)`

引数

(in)*ui16ScaleFactor*　加速度のスケールファクタ
1G に相当する加速度の数値を指定します。
(in)*pParam*　感度パラメータ（通常は NULL ポインタを渡してください）

戻り値

なし

説明

ライブラリの内部バッファ、変数の初期化を行います。
pParam には検出感度(=閾値)とダブルタッピング検出時間を設定することができます。

検出感度の数値を小さくすることでタッピングを検出しやすくなりますが､誤検出する可能性があります。感度を変更する場合は、*pParam[0]* に任意の値を設定してください。感度の初期値は 1000000h、型は unsigned long です。

また､最初のタッピングを検出後､所定の時間内は最初のタッピングの分散値の半分を閾値とし、2 回目のタップを検出しやすくしています。その所定時間を *pParam[1]* で設定することが可能です。初期値は 12、型は unsigned char です。 例えば、本関数を 12.5Hz でコールする場合、12/12.5 でおよそ 1 秒間となります。

se_TAPDetect

インクルード

```
#include “se_tapdet.h”
```

形式　`int se_TAPDetect(short ai16AccData[3])`

引数

(in)*ai16AccData*　入力加速度データの配列

戻り値

タッピングを検出すると 1 が返されます。それ以外は 0 が返されます。

説明

ai16AccData[3]から入力した 25Hz あるいは 12.5Hz の加速度によりタッピングを検出します。加速度センサのサンプリング周期に合わせて、本関数をコールしてください。

各軸毎に、連続する 3 サンプルの分散を求め、その合計値が所定の閾値を超えた場合、タッピングとして検出します。

11. 要求メモリ

表 11-1 要求 ROM サイズ（byte）

オブジェクト	C17	C33	FSA
se_accfifo	938	654	402
se_magfifo	150	132	0
se_steplib	2620	2104	326
se_movsta	1878	1476	0
se_calo	236	158	192
se_movvec	1058	886	440
se_magyaw	424	300	532
se_tiltdet	750	562	0
se_tapdet	502	356	0

表 11-2 要求 RAM サイズ（byte）

オブジェクト	IRAM	FSA-RAM	FSA スタック(*1)
se_accfifo	20	710	44
se_magfifo	20	300	44
se_steplib	68	578	50
se_movsta	0	64	0
se_calo	0	192	8
se_movvec	24	568	12
se_magyaw	0	532	20
se_tiltdet	18	0	0
se_tapdet	32	18	6

(*1) FSA スタックメモリとは、FSA が一時的な作業領域として使用するメモリです。
具体的には、FSA がアクセス可能なメモリの最上位の領域が使用されます。

改訂履歴表

付－1

Rev. No.	日付	ページ	種別	改訂内容（旧内容を含む） および改訂理由
Rev 1.0	2015/4/3	全ページ	新規	新規制定
Rev 1.1	2016/10/3	29	追記	se_tapdet の関数仕様を追記

セイコーエプソン株式会社
マイクロデバイス事業部　デバイス営業部

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

大阪　〒541-0059　大阪市中央区博労町 3-5-1　御堂筋グランタワー15F
TEL（06）6120-6000（代表）　FAX（06）6120-6100

ドキュメントコード：412956701
2015 年 4 月 作成
2016 年 10 月 改定