EPSON
EXCEED YOUR VISION

FSA サンプルプログラム
マニュアル

-歩行自律測位-

セイコーエプソン株式会社

Rev.1.0

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 FSA サンプルプログラムは、加速度応用システムライブラリを用いて歩行自律測位を行います。センサハブマイコンに組み込むことを想定してデザインされた se_drcore ライブラリと、ホスト CPU に組み込むことを想定した se_dr ライブラリの 2 つのパートで構成されます。但し、提供形態としては、両者が一体化され 1 つのライブラリとして機能する形になっています。

本 FSA サンプルプログラム全体の構成を図 1-1 に示します。

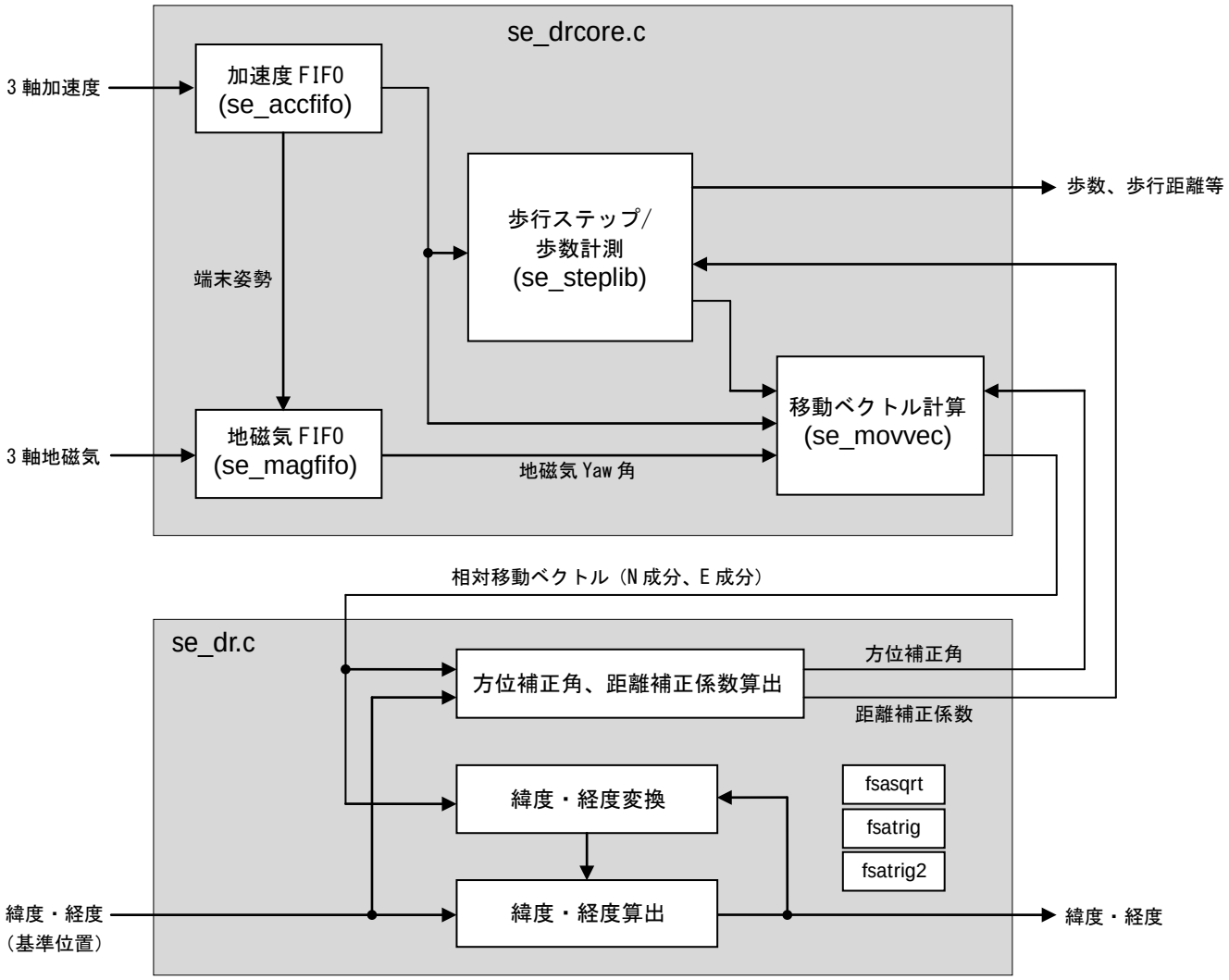

図 1-1　歩行自律測位の FSA ライブラリ構成

se_drcore ライブラリ

se_drcore ライブラリは、緯度・経度に代表される絶対位置情報は扱わず、25Hz もしくは 12.5Hz の 3 軸加速度、3 軸地磁気データを入力として、ユーザーの所定時間当たりの相対移動ベクトル（North 成分、East 成分）を算出します。また、se_dr ライブラリからフィードバックされる距離補正係数、方位補正角に従って、移動距離と移動方位を補正する機能も備えます。

se_drcore ライブラリで実行される移動ベクトルを算出する se_movvec ライブラリは、端末のローカル座標におけるユーザーの相対移動方位角を、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（装着状態）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。）

se_drcore ライブラリは、FSA_package_v2 に同梱された加速度応用システムライブラリ（se_accfifo、

se_magfifo、se_steplib、se_movvec）を使用します。

se_dr ライブラリ

se_dr ライブラリは、se_drcore ライブラリが出力する相対移動ベクトルと、GPS 等から得られる基準位置を入力して、基準位置間の位置を計算します。また、se_dr ライブラリは、基準位置を入力すると、前回の基準位置入力を起点として、自律測位による位置と入力された基準位置とを比較し、自律測位の方位を補正する補正角と移動距離（速度）を補正する距離補正係数を算出します。

se_dr ライブラリは、FSA_package_v1 に同梱された汎用ライブラリ（fsasqrt、fsatrig、fsatrig2）を使用します。

ロギング測位

本ライブラリは、任意のタイミングで現在位置を取得するリアルタイム測位の他に、一定期間の移動ベクトルをバッファに蓄積し、基準測位を入力した直後に緯度・経度に変換し、一気に取得するロギング測位機能をサポートしています。

表 1-1　se_drcore のサポート関数一覧

関数	概仕様
se_DRCInit	ライブラリの初期化を行います。
se_DRCExe	移動ベクトルを計算するメイン関数です。25Hz もしくは 12.5Hz でコールします。
se_DRCSetLogBuf	ロギングバッファを設定します。
se_DRCSetLogMeasInt	ロギング測位間隔を設定します。
se_DRCSetDeclination	地磁気偏角を設定します。
se_DRCSetDirEstEn	端末座標における推定進行方向の更新を制御します。
se_DRCSetMovAngOnBody	移動方向をローカル座標上の相対角として設定します。
se_DRCSetCorrectScale	移動ベクトルの大きさを補正する補正係数を設定します。
se_DRCSetCorrectAngle	移動ベクトルの方向を補正する補正角を設定します。
se_DRCGetMovVec	移動ベクトルを取得します。
se_DRCGetLogMovVec	ロギング測位用の移動ベクトルを取得します。
se_DRCGetLogDataNum	未読状態のロギング移動ベクトルの数を取得します。
se_DRCGetState	停止、移動、曲がり角、持ち変え変更の状態変化を取得します。
se_DRCGetWalkStep	累計歩数を取得します。
se_DRCGetWalkDist	累計歩行移動距離を取得します。

表 1-2　se_dr のサポート関数一覧

関数	概仕様
se_DRInitialize	ライブラリの初期化を行います。
se_DRSetPosition	基準となる緯度・経度を設定します。
se_DRGetPosition	リアルタイム測位による緯度・経度を取得します。
se_DRGetLogPosition	ロギング測位による緯度・経度を取得します。
se_DRSetAngleCorrectGain	方位の補正のゲインを設定します。
se_DRSetScaleCorrectGain	距離の補正ゲインを設定します。

2. サンプルプログラム構成ファイル

se_drcore ライブラリは、表 2-1 に示す加速度応用システムライブラリと表 2-2 に示すソースファイルで構成されます。

表 2-1　se_drcore ライブラリが使用する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	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。
movvec	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と進行方向の相対角、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。

表 2-2　se_drcore ライブラリのソースファイル

ソースファイル	内容
se_drcore.h	ライブラリ関数宣言、構造体型定義が記載されたヘッダーファイル
se_drcore.c	歩行自律測位ライブラリの関数定義
fsaleftshift32.c	32 ビット左シフト用定数テーブル

se_dr ライブラリは、表 2-3 に示す FSA 汎用ライブラリと表 2-4 に示すソースファイルで構成されます。

表 2-3　se_dr ライブラリが使用する FSA 汎用ライブラリ

パッケージ名	オブジェクトファイル	内容
fsasqrt	fsasqrt.o fsasqrt_fsa.o fsa_symbols_fsasqrt.def	FSA_package_v1 に同梱されている平方根ライブラリです。FsaSumSqrt16 関数、FsaSumSqrt32 関数を使用します。
fsatrig	fsatrig.o fsatrig_fsa.o fsa_symbols_fsatrig.def	FSA_package_v1 に同梱されている三角関数ライブラリです。FsaAtan2 関数を使用します。
fsatrig2	fsatrig2.o fsatrig2_fsa.o fsa_symbols_fsatrig2.def	FSA_package_v1 に同梱されている三角関数ライブラリです。FsaCos32 関数、FsaSin16 関数、FsaCos16 関数を使用します。

表 2-4　se_dr ライブラリのソースファイル

ソースファイル	内容
se_dr.h	ライブラリ関数宣言、構造体型定義が記載されたヘッダーファイル
se_dr.c	歩行自律測位ライブラリの関数定義
fsaleftshift32.c	32 ビット左シフト用定数テーブル

なお、FSA_package_vol.2 では、“fsaleftshift32.c”を除く、se_drcore および se_dr の各ライブラリの構成ファイルを統合するビルド環境を用意しています。

表 2-3　se_drcore および se_dr ライブラリを構成する統合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
pdr	se_drcore.o se_drcore_fsa.o fsa_symbols_se_drcore.def	se_drcore ライブラリの構成ファイル群を統合したオブジェクトファイルです。
	se_dr.o se_dr_fsa.o fsa_symbols_se_dr.def	se_dr ライブラリの構成ファイルを統合したオブジェクトファイルです。

3. 構造体

SE_DR_MOVVEC 構造体

SE_DR_MOVVEC 構造体は、リアルタイムおよびロギング測位の出力となる移動ベクトル、タイムスタンプを格納するための構造体です。SE_DRC 構造体は、se_dr ライブラリ、se_drcore ライブラリの両方で使用します。構造体定義は se_drcore.h で型定義されています。

```
typedef struct {
     short           i16East;
     short           i16North;
     unsigned short  ui16TimeStamp;
} SE_DR_MOVVEC;
```

i16East
移動ベクトルの東方向の成分です。小数点以下 4 ビットの数値で、単位はメートルです。

i16North
移動ベクトルの北方向の成分です。小数点以下 4 ビットの数値で、単位はメートルです。

ui16TimeStamp
直前の測位点からの経過時間が格納されます。単位は 1/25 秒です。

SE_DRC 構造体

se_drcore ライブラリが使用する構造体です。SE_DRC 構造体は、se_drcore.h で型定義され、実体は se_drcore.c でグローバル変数として定義されます。

```
typedef struct {
     long            ai32Posi[2];
     unsigned long   ui32WalkDist;
     unsigned short  ui16WalkStep;
     unsigned short  ui16TimeCount;
     unsigned short  ui16TimeStamp;
     unsigned short  ui16SampleDelay;
     unsigned char   ui8TiltState;
     unsigned char   ui8WalkState;
     unsigned char   ui8TurnState;
} SE_DRC;
```

ai32Posi[2]
リアルタイム測位用の移動ベクトルを積算するための変数です。ai32Posi[0]に East 成分、ai32Posi[1]に North 成分が格納され、各々小数点以下 8 ビットの数値で単位はメートルです。

ui32WalkDist
歩行移動距離を積算するための変数です。小数点以下 8 ビットの数値で、単位はメートルです。

ui16WalkStep
歩数を積算するための変数です。

ui16TimeCount
時間を計測するための変数です。25Hz の周期でインクリメントされます。

ui16TimeStamp
測位時刻を一時的に保持する変数です。測位が行われたタイミングで、ui16TimeCount の値がコピーされます。

ui16SampleDelay
入力する加速度、地磁気のサンプリング周期が設定される変数です。25Hz のとき 40（ms)、12.5Hz のとき 80（ms）になります。

ui8TiltState
持ち変え変更の有無を格納する変数です。持ち変え変更を検出すると 1 がセットされます。それ以外は 0 です。

ui8WalkState
歩行状態を格納する変数です。停止状態は 0、歩行状態は 1 がセットされます。

ui8TurnState
移動方位の変更検出を格納する変数です。有意な移動方位の変化を検出すると 1 がセットされます。それ以外は 0 です。

SE_DRC_LOG 構造体

se_drcore ライブラリのロギング測位で使用する構造体です。SE_DRC_LOG 構造体は、se_drcore.h で型定義され、実体は se_drcore.c でグローバル変数として定義されます。

```
typedef struct {
     long            ai32Posi[2];
     SE_DR_MOVVEC    *pstLogTop;
     SE_DR_MOVVEC    *pstLogEnd;
     SE_DR_MOVVEC    *pstWritePos;
     SE_DR_MOVVEC    *pstReadPos;
     unsigned short  ui16TimeCount;
     unsigned short  ui16TimeStamp;
     unsigned short  ui16LogSize;
     unsigned short  ui16UnReadNum;
     unsigned short  ui16MeasInt;
} SE_DRC_LOG;
```

ai32Posi[2]
ロギング測位用の移動ベクトルを積算するための変数です。ai32Posi[0]に East 成分、ai32Posi[1]に North 成分が格納され、各々小数点以下 8 ビットの数値で単位はメートルです。

pstLogTop
SE_DR_MOVVEC 型のロギングバッファの先頭ポインタが格納されます。

pstLogEnd
SE_DR_MOVVEC 型のロギングバッファの後尾ポインタが格納されます。

pstWritePos
SE_DR_MOVVEC 型のロギングバッファの書き込み位置を示すポインタが格納されます。

pstReadPos
SE_DR_MOVVEC 型のロギングバッファの読み出し位置を示すポインタが格納されます。

ui16TimeCount
時間を計測するための変数です。25Hz の周期でインクリメントされます。

ui16TimeStamp
測位時刻を一時的に保持する変数です。測位が行われたタイミングで、ui16TimeCount の値がコピーされます。

ui16LogSize
SE_DR_MOVVEC 型のロギングバッファに記録可能な測位点の数が格納されます。

ui16UnReadNum
SE_DR_MOVVEC 型のロギングバッファ上の未読の測位点の数が格納されます。

ui16MeasInt
ロギング測位間隔が格納されます。単位は ms です。

SE_POSITION 構造体

se_dr ライブラリで扱う位置情報を格納する構造体です。SE_POSITION 構造体は、se_dr.h で型定義されています。

```
typedef struct {
     long            i32latitude;
     long            i32longitude;
     unsigned short  ui16Speed;
     short           i16Bearing;
     long long       i64Utc;
} SE_POSITION;
```

i32latitude
緯度を格納する変数です。格納される値は小数点以下 30 ビットで、-180° ～+180° を-2.0～+2.0 に正規化した数値です。

i32longitude
経度を格納する変数です。格納される値は小数点以下 30 ビットで、-180° ～+180° を-2.0～+2.0 に正規化した数値です。

ui16Speed
水平速度を格納する変数です。格納される値は小数点以下 4 ビットで単位は m/s です。

i16Bearing
水平移動方位を格納する変数です。格納される値は小数点以下 14 ビットで、-180° ～+180° を-2.0～+2.0 に正規化した数値です。

i64Utc
UTC 時刻を格納する変数です。ms です。

SE_DR_LOG 構造体

se_dr ライブラリのロギング測位に使用する構造体です。SE_DR_LOG 構造体は、se_dr.h で型定義されています。

```
typedef struct {
     SE_POSITION     stRefPosition;
     long            i32CorrectScale;
     short           i16CorrectAngle;
     short           i16Reserved;
     unsigned short  ui16State;
     unsigned short  ui16LogSize;
} SE_DR_LOG;
```

stRefPosition
ロギング測位の基準位置情報を格納する構造体変数です。se_DRSetPosition 関数を実行すると更新されます。

i32CorrectScale
ロギング測位の距離補正係数を格納する変数です。小数点以下 14 ビットの数値です。

i16CorrectAngle
ロギング測位の方位補正角を格納する変数です。格納される値は小数点以下 14 ビットで、-180° ～ +180° を-2.0～+2.0 に正規化した数値です。

i16Reserved
アライメントのための予約変数です。

ui16State
基準位置の設定状態を管理する変数です。初期化時は 0 で、最初の基準位置がセットされると 1 になります。

ui16LogSize
se_DRGetLogPosition 関数内で取得したロギング測位データの数を格納する変数です。

SE_DR 構造体

se_dr ライブラリで使用する構造体です。SE_DR 構造体は、se_dr.h で型定義されています。この構造体の実体はライブラリ外部で確保します。

```
typedef struct {
     SE_POSITION     stDrPosition;
     SE_POSITION     stRefPosition;
     long            ai32MovVec[2];
     long            ai32ErrPosi[2];
     long            i32DrawTime;
     long            i32CorrectScale;
     short           i16CorrectAngle;
     short           i16ScaleCorrectGain;
     short           i16AngleCorrectGain;
     short           i16ErrGain;
     short           i16GainTheta;
     unsigned short  ui16State;
     long            ai32Temp[4];
#ifdef SE_DR_LOGGING_ENABLE
     SE_DR_LOG       stDRLog;
#endif
} SE_DR;
```

stDrPosition
最新の自律測位による位置情報を格納する構造体変数です。se_DRGetPosition 関数を実行すると更新されます。

stRefPosition
最新の基準位置情報を格納する構造体変数です。se_DRSetPosition 関数を実行すると更新されます。

ai32MovVec[2]
stRefPosition を起点とした現在位置を移動ベクトルで格納する変数です。ai32MovVec[0]に North 成分、ai32MovVec[1]に East 成分が格納され、各々小数点以下 4 ビットで単位 m の数値です。

ai32ErrPosi[2]
se_DRSetPosition 関数でセットした基準位置とその時点の自律測位による測位点との誤差を格納する変数です。格納される値は小数点以下 30 ビットで、-180° ～+180° を-2.0～+2.0 に正規化した数値です。ai32ErrPosi [0]に緯度の誤差、ai32ErrPosi [1]に経度の誤差が格納されます。

i32DrawTime
ai32ErrPosi[2]にセットされた測位誤差を徐々にその後の自律測位出力に反映させていく際の時間を設定する変数です。0x40000000 を se_DRSetPosition 関数の引数 i32DrawTime で除した数値がセットされます。但し、引数 i32DrawTime が 0 の場合は 0 がセットされます。

i32CorrectScale
距離補正係数を格納する変数です。小数点以下 14 ビットの数値です。

i16CorrectAngle
方位補正角を格納する変数です。格納される値は小数点以下 14 ビットで、-180° ～+180° を-2.0～+2.0 に正規化した数値です。

i16ScaleCorrectGain
se_DRSetScaleCorrectGain 関数で設定される距離補正ゲインを格納する変数です。小数点以下 14

ビットの数値です。

i16AngleCorrectGain
se_DRSetAngleCorrectGain 関数で設定される方位補正ゲインを格納する変数です。小数点以下 14 ビットの数値です。

i16ErrGain
ai32ErrPosi[2]にセットされた測位誤差をその後の自律測位出力に反映させるゲインを格納する変数です。小数点以下 14 ビットの数値です。

i16GainTheta
i16ErrGain を算出するための変数です。格納される値は小数点以下 14 ビットで、-π～+πを-2.0～+2.0 に正規化した数値です。se_DRSetPosition 関数実行後に+π/2 にセットされ、se_DRGetPosition 関数を実行する毎に角度が減じられ、i32DrawTime で設定された時間が経過すると-π/2 になります。この i16GainTheta を用いて、i16ErrGain は下記の式により算出されます。

```
i16ErrGain = (1.0 + sin(i16GainTheta)) / 2.0
```

i16State
基準位置の設定状態を管理する変数です。初期化時は 0 で、距離補正係数、方位補正角の算出に有効な基準位置がセットされると 1 になります。

stDRLog
ロギング測位のための変数が格納される SE_DR_LOG 構造体変数です。

4. マクロ定義

ロギング測位の有効設定

se_drcore.h の下記の#define 定義を変更することで、ロギング測位機能を無効にすることが可能です。無効にした場合、ロギング測位に必要な関数、ヒープ領域がコンパイル対象から除外されます。

se_drcore.h の記述

```
// When the logging function is needed, the following definition must be valid.
#define SE_DR_LOGGING_ENABLE
//-----------------------------------------------------------------------------
```

補正下限距離の設定

距離補正係数、方位補正角を算出するための下限距離の設定です。基準位置間の直線距離およびその間の自律測位の移動距離がここで定義された距離より短い場合、補正係数、補正角の更新を行いません。設定の単位はメートルです。

se_dr.h の記述

```
// Setting of minimum distance between reference positions for correction.
#define SE_DR_CORRECT_MIN_DISTANCE                          10.0
//-----------------------------------------------------------------------------
```

方位補正角の範囲設定

方位補正角の範囲設定です。基準測位と自律測位の方位誤差がここで設定する範囲を超えた場合、設定範囲にクリップされます。

se_dr.h の記述

```
// Setting of correction angle limitation. The unit is degree.
#define SE_DR_CORRECT_ANGLE_UPPRT_LIMIT                     +30.0
#define SE_DR_CORRECT_ANGLE_LOWER_LIMIT                     -30.0
//-----------------------------------------------------------------------------
```

距離補正係数の範囲設定

距離補正係数の範囲設定です。基準測位と自律測位の距離比から 1.0 を差し引いた値が、ここで設定する範囲を超えた場合、設定範囲にクリップされます。

se_dr.h の記述

```
// Setting of correction scale limitation
#define SE_DR_CORRECT_SCALE_UPPRT_LIMIT                     +0.2
#define SE_DR_CORRECT_SCALE_LOWER_LIMIT                     -0.2
//-----------------------------------------------------------------------------
```

FSA 汎用ライブラリの設定

以下に示すように、se_dr ライブラリで使用しない関数をコメントアウトします。コメントアウトすることで、使用しない関数をコンパイル対象から除外できます。但し、同時に組み込む別のアプリケーションで使用する場合は、この限りではありません。

fsasqrt.h は、_FSASUMSQRT16 と_FSASUMSQRT32 の定義以外をコメントアウトします。

fsasqrt.h の設定

```
// Make necessary function only valid in the following definitions.
// #define _FSASQRTINV
// #define _FSASQRT
#define _FSASUMSQRT16
#define _FSASUMSQRT32
```

fsatrig.h は、_FSAATAN2 以外の定義をコメントアウトします。

fsatrig.h の設定

```
// Function selection
// Disable the definitions except the function you will use.
// #define _FSACOS
// #define _FSASIN
//#define _FSACOSSIN
#define _FSAATAN2
//#define _FSAASIN
```

fsatrig2.h は、_FSASIN32 のみをコメントアウトします。

fsatrig2.h の設定

```
// Function selection
// Disable the definitions except the function you will use.
// #define _FSASIN32
#define _FSACOS32
#define _FSASIN16
#define _FSACOS16
//--------------------------------------------------------------------------
```

5. サンプルプログラムの制御

5.1 初期化

se_DRInitialize 関数を実行して se_dr ライブラリの初期化を行います。その際、se_DRCExe 関数に入力する加速度データ、地磁気データのサンプリング周期および加速度データのスケールファクタを設定します。詳細は、7. API 関数仕様の se_DRInitialize 関数の仕様を参照してください。
なお、se_DRInitialize 関数内部で se_DRCInit 関数が実行されるため、se_drcore ライブラリの初期化のために、別途 se_DRCInit 関数を実行する必要はありません。

5.2 動作設定

相対移動方位角の設定

自律測位の移動方位を決定するためには、ワールド座標における端末のローカル座標の絶対方位角と、端末のローカル座標におけるユーザーの進む方向を示す相対移動方位角の２つの情報が必要です。本ライブラリでは、前者は地磁気により推定し、後者の相対移動方位角は、下記の２つの方法の何れかで決定します。

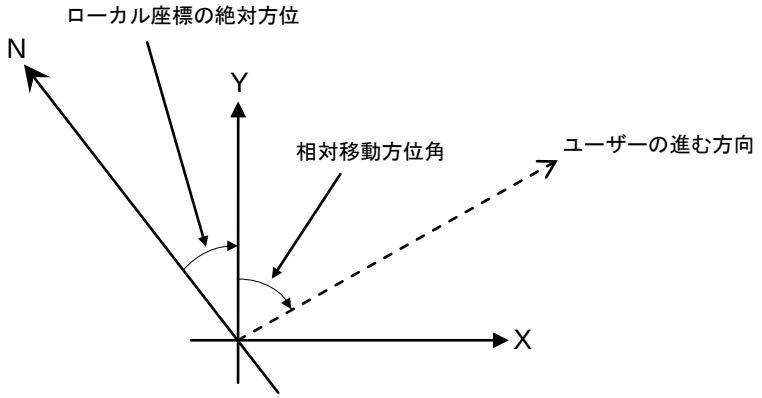

図 5-1 端末ローカル座標（ENU 座標）における相対移動方位角の定義

［自動推定を行う］
加速度から相対移動方位角を自動推定することができます。その場合は、se_DRCSetDirEstEn 関数の引数を 1 にして実行します。なお、se_DRInitialize 関数を実行した時点で、自動指定はオンになっています。se_DRCSetDirEstEn 関数の引数を 0 にして実行すると、直前までに自動推定した移動方位角の更新が停止します。
詳細は、6. API 関数仕様の se_DRCSetDirEstEn 関数の仕様を参照してください。

［相対移動方位角を直接指定する］
端末の装着方法が既定されているなど、端末のローカル座標における移動方位角が既知の場合は、相対移動方位角を、se_DRCSetMovAngOnBody 関数を用いて直接設定します。相対移動方位角を指定すると、自動推定はオフになります。
詳細は、6. API 関数仕様の se_DRCSetMovAngOnBody 関数の仕様を参照してください。

距離補正、方位補正ゲインの設定

se_DRSetScaleCorrectGain 関数および se_DRSetAngleCorrectGain 関数を用いて、距離補正ゲインと方位補正ゲインを設定します。se_DRInitialize 関数を実行した時点で、各補正ゲインは 0 になっています。基準測位による補正を行う場合は、各補正ゲインを設定してください。なお、補正ゲインの設定は、自律測位実行中も含めて随時可能ですが、設定が反映されるタイミングは、se_DRSetPosition 関数の実行時になります。
詳細は、7. API 関数仕様の se_DRSetScaleCorrectGain 関数、se_DRSetAngleCorrectGain 関数の仕

様を参照してください。

ロギング測位の設定

ロギング測位を行う場合は、下記に示すロギング測位データを格納するバッファの設定と、ロギング測位間隔の設定が必要になります。

［ロギングバッファの設定］
se_DRCSetLogBuf関数を用いて、ロギング測位データを格納するバッファとそのサイズを設定します。詳細は、6. API関数仕様のse_DRCSetLogBuf関数の仕様を参照してください。

［ロギング測位間隔の設定］
se_DRCSetLogMeasInt関数を用いて、ロギング測位間隔を設定します。詳細は、6. API関数仕様のse_DRCSetLogMeasInt関数の仕様を参照してください。

5.3 自律測位開始位置の設定

se_DRSetPosition 関数により自律測位の開始位置を設定します。自律測位を開始するには、起点となる最初の基準位置設定が必要です。

5.4 自律測位の実行

センサデータの入力

スタート位置となる最初の基準位置を設定後、se_DRCExe 関数に初期化時に指定したサンプリング周期の加速度データ、地磁気データを与え、繰り返し実行することで、自律測位が行われます。詳細は、6. API 関数仕様の se_DRCExe 関数の仕様を参照してください。

基準位置の入力

センサのみによる自律測位を長時間行うと徐々に測位誤差が大きくなります。従って、適当なタイミングで se_DRSetPosition 関数により基準位置を再セットします。

5.5 測位値の取得

リアルタイム測位値の取得

リアルタイム測位値を取得する場合は、測位を行うタイミングで、se_DRGetPosition 関数を実行して取得します。詳細は、7. API 関数仕様の se_DRGetPosition 関数の仕様を参照してください。

ロギング測位値の取得

ロギング測位値を取得する場合は、原則として se_DRSetPosition 関数を実行した直後に、se_DRGetLogPosition 関数を実行して取得します。詳細は、7. API 関数仕様の se_DRGetLogPosition 関数の仕様を参照してください。

6. se_drcore ライブラリの API 関数仕様

se_DRCInit

インクルード

```
#include “se_drcore.h”
```

形式　`void se_DRCInit(unsigned short ui16SampleDelay, unsigned short ui16AccScaleFactor)`

引数

(in)*ui16SampleDelay*　　加速度、地磁気のサンプリング周期
(in) *ui16AccScaleFactor*　加速度のスケールファクタ

戻り値

なし

説明

SE_DRC 構造体変数の初期化および使用する加速度応用ライブラリの初期化を行います。サンプリング周期は 40（25Hz）もしくは 80（12.5Hz）のいずれかを選択し、ui16SampleDelay に設定します。ui16AccScaleFactor には 1G に相当する加速度の数値を指定します。本関数は、se_DRInitialize 関数内で実行されます。

se_DRCExe

インクルード

```
#include “se_drcore.h”
```

形式　`int se_DRCExe(FSAREG *pFsaReg, short ai16Acc[3], short ai16Mag[3])`

引数

(in)*pFsaReg*　　FSA レジスタ構造体のポインタ
(in)*ai16Acc[3]*　3 軸加速度データ
(in)*ai16Mag[3]*　3 軸地磁気データ

戻り値

移動ベクトルが更新されたとき 1 が返されます。そうでない場合は 0 が返されます。

説明

歩行自律測位を行うメイン関数です。se_DRCInit 関数で指定したサンプリング周期の加速度データ、地磁気データを与え実行します。
加速度データは ENU 座標の X 軸を ai16Acc[0]に、Y 軸を ai16Acc[1]に、Z 軸を ai16Acc[2]にセットします。地磁気データは ENU 座標の X 軸を ai16Mag[0]に、Y 軸を ai16Mag[1]に、Z 軸を ai16Mag[2]にセットします。

se_DRCSetLogBuf

インクルード

```
#include “se_drcore.h”
```

形式　`void se_DRCSetLogBuf(SE_DR_MOVVEC astLogBuf[], unsigned int uiLogSize)`

引数

(in)*astLogBuf*	ロギング移動ベクトルを格納する配列のポインタ
(in)*uiLogSize*	astLogBuf 配列のサイズ

戻り値

なし

説明

ロギング移動ベクトルを格納するための SE_DR_MOVVEC 型配列のポインタとそのサイズを設定します。ロギング測位を行わない場合、本関数を実行する必要はありません。

se_DRCSetLogMeasInt

インクルード

```
#include “se_drcore.h”
```

形式　`void se_DRCSetLogMeasInt(unsigned int ui16LogMeasInt)`

引数

(in)*ui16LogMeasInt*	ロギング測位間隔（単位：1/25 秒）

戻り値

なし

説明

ロギング測位間隔を設定します。0 を設定した場合はロギング測位を行いません。se_DRCInit 関数による初期化後のロギング測位間隔は 0 です。ロギング測位を行わない場合、本関数を実行する必要はありません。

se_DRCSetDeclination

インクルード

```
#include “se_drcore.h”
```

形式　`void se_DRCSetDeclination(short i16Angle)`

引数

(in)*i16Angle*	地磁気偏角（小数点以下 7 ビット、単位：度）

戻り値

なし

説明

地磁気偏角（真北と磁北の差）を設定します。

se_DRCSetDirEstEn

インクルード

```
#include “se_drcore.h”
```

形式　`void se_DRCSetDirEstEn(int iEnable)`

引数

(in)*iEnable*　　相対移動方位角自動推定のイネーブルフラグ

戻り値

なし

説明

端末のローカル座標（ENU 座標）における相対移動方位角を自動推定する場合は、iEnable に 1 をセットします。自動推定をオンにすると、ユーザーが歩行状態のとき、加速度から相対移動方位角を自動推定します。
自動推定を行わない場合は iEnable に 0 をセットします。se_DRCInit 関数による初期化直後は、自動推定はオン状態です。初期化後直ちに自動推定をオフした場合、相対移動方位角は 0°で、Y 軸のプラス方向を移動方向とします。
しばらく自動推定を行った後にオフした場合、オフ直前の相対移動方位角が保持されます。

se_DRCSetMovAngOnBody

インクルード

```
#include “se_drcore.h”
```

形式　`void se_DRCSetMovAngOnBody(short i16Angle)`

引数

(in)*i16Angle*　　相対移動方位角
（小数点以下 14 ビット、*-180°～+180°を-2.0～+2.0 に正規化した値）

戻り値

なし

説明

i16Angle で指定した角度を、端末のローカル座標における相対移動方位角として直接設定します。本関数を実行すると、自動推定機能はオフされます。ユーザーが端末を装着する際の装着方法が規定可能な場合は、相対移動方位角の自動推定は行わず、本関数で規定された角度を設定してください。設定角度と ENU 端末座標における移動方位角の関係を表 6-1 に示します。

表 6-1 設定角度と移動方位角の関係

設定角度	相対移動方位
0°	Y 軸プラス方向
+90°	X 軸プラス方向
-180°	Y 軸マイナス方向
-90°	X 軸マイナス方向

se_DRCSetCorrectScale

インクルード

```
#include “se_drcore.h”
```

形式　`void se_DRCSetCorrectScale(unsigned short ui16Scale)`

引数

(in)*ui16Scale*　　距離補正係数（小数点以下 14 ビットの数値）

戻り値

なし

説明

距離補正係数を設定します。本関数を実行すると、以降の歩行距離が ui16Scale で指定した係数倍の歩行距離に補正されます。

se_DRCSetCorrectAngle

インクルード

```
#include “se_drcore.h”
```

形式　`void se_DRCSetCorrectAngle(short i16Angle)`

引数

(in)*i16Angle*　　移動方位補正角
（小数点以下 14 ビット、*-180°～+180°を-2.0～+2.0 に正規化した値））

戻り値

なし

説明

方位補正角を設定します。本関数を実行すると、以降の移動方位角に i16Angle で指定した角度が加算されます。

se_DRCGetMovVec

インクルード

```
#include “se_drcore.h”
```

形式　`void se_DRCGetMovVec(SE_DR_MOVVEC *pstMovVec)`

引数

(out)*pstMovVec*　　SE_DR_MOVVEC 型構造体のポインタ

戻り値

なし

説明

最後に se_DRCMovVec 関数を実行してから現在に至るまでの移動ベクトルを取得します。SE_DR_MOVVEC 構造体変数の詳細は、3. 構造体を参照してください。

se_DRCGetLogMovVec

インクルード

```
#include “se_drcore.h”
```

形式　`unsigned short se_DRCGetLogMovVec(SE_DR_MOVVEC astMovVec[], unsigned short ui16Num)`

引数

(out)*astMovVec*	SE_DR_MOVVEC 型構造体配列のポインタ
(in)*ui16Num*	取得可能なロギング移動ベクトルの数

戻り値

取得したロギング移動ベクトルの数が返されます。

説明

ロギング移動ベクトルを取得し astMovVec に格納します。ui16Num には受け取り可能なロギング移動ベクトルの数（確保した astMovVec 配列の要素数）を指定してください。実際に取得したロギング移動ベクトルの数は戻り値として返されます。本関数で取得したロギング移動ベクトルは既読状態となり、再度の取得はできません。

se_DRCGetLogDataNum

インクルード

```
#include “se_drcore.h”
```

形式　`unsigned short se_DRCGetLogDataNum()`

引数

なし

戻り値

未読状態のロギング移動ベクトルの数が返されます。

説明

未読状態のロギング移動ベクトルの数を取得します。
本関数により、バッファに入力されたロギング移動ベクトルの数を調べ、バッファがオーバーフローしそうな場合は、se_DRCGetLogMovVec 関数により、ロギング測位データを取得してください。

se_DRCGetState

インクルード

```
#include “se_drcore.h”
```

形式　`int se_DRCGetState()`

引数

なし

戻り値

自律測位における状態変化に応じて値を返します。戻り値の詳細は、表 6-2 の通りです。
但し、同時に複数の状態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 6-2 se_DEGetState 関数の戻り値

値	検出した状態
0	未検出
1	姿勢変化（持ち方変更）の検出
2	停止状態への移行
4	歩行状態への移行
8	移動方位変化（曲がり角）の検出

説明

自律測位における状態変化を取得します。本関数の戻り値から、自律測位の状態変化を確認することができます。必要に応じて戻り値に応じた処理を行ってください。なお、停止状態における姿勢変化は、歩行状態移行後に出力されます。

se_DRCGetWalkStep

インクルード
```
#include “se_drcore.h”
```

形式　unsigned short se_DRCGetWalkStep()

引数
なし

戻り値
累計歩数が返されます。

説明
累計歩数を取得します。累計歩数は、本関数を実行するとゼロ初期化されます。

se_DRCGetWalkDist

インクルード
```
#include “se_drcore.h”
```

形式　unsigned long se_DRCGetWalkDist()

引数
なし

戻り値
累計歩行距離（少数点以下 8 ビット、単位：メートル）が返されます。

説明
累計歩行距離を取得します。累計歩行距離は、本関数を実行するとゼロ初期化されます。

7. se_dr ライブラリの API 関数仕様

se_DRInitialize

インクルード

```
#include “se_dr.h”
```

形式

```
void se_DRInitialize(
        SE_DR           *pstDR,
        unsigned short  ui16SampleDelay,
        unsigned short  ui16AccScaleFactor
)
```

引数

(in)*pstDR*	se_dr ライブラリのハンドラ構造体のポインタ
(in)*ui16SampleDelay*	加速度、地磁気のサンプリング周期
(in) *ui16AccScaleFactor*	加速度のスケールファクタ

戻り値

なし

説明

se_dr ライブラリの初期化を行います。pstDR は se_dr ライブラリのハンドラで、そのポインタを本関数に渡して初期化します。サンプリング周期は 40（25Hz）もしくは 80（12.5Hz）のいずれかを選択し、ui16SampleDelay に設定します。ui16AccScaleFactor には 1G に相当する加速度の数値を指定します。
本関数内では、se_DRCInit 関数も実行されますので、se_drcore ライブラリの初期化も行われます。なお、pstDR の実体は必ず FSA がアクセス可能なメモリ領域に確保してください。

se_DRSetPosition

インクルード

```
#include “se_dr.h”
```

形式

```
int se_DRSetPosition(
        FSAREG          *pFsaReg,
        SE_DR           *pstDR,
        short           i16Declination,
        SE_POSITION     *pstPosition,
        short           i16DrawTime
)
```

引数

(in)*pFsaReg*	FSA レジスタ構造体のポインタ
(in)*pstDR*	se_dr ライブラリのハンドラ構造体のポインタ
(in)*i16Declination*	地磁気偏角（小数点以下 7 ビット、単位：度）
(in)*pstPosition*	位置情報が格納された SE_POSITION 構造体のポインタ
(in) *i16DrawTime*	設定位置への引き込み時間（単位：1/25 秒）

戻り値

距離補正係数、方位補正角を更新した場合は 1 が返されます。それ以外は 0 が返されます。

説明

pstPosition に格納された位置情報を基準位置として設定します。同時に、設定位置に応じた地磁気偏角を i16Declination に渡します。i16DrawTime は、自律測位を設定した基準位置に引き込むまでの時間を制御するパラメータです。例えば、i16DrawTime に 60 秒をセットすると、自律測位は、設定した基準位置との誤差を 60 秒かけて解消するような軌跡になります。0 を設定すると自律測位は、設定した基準位置に直ちに引き込まれます。

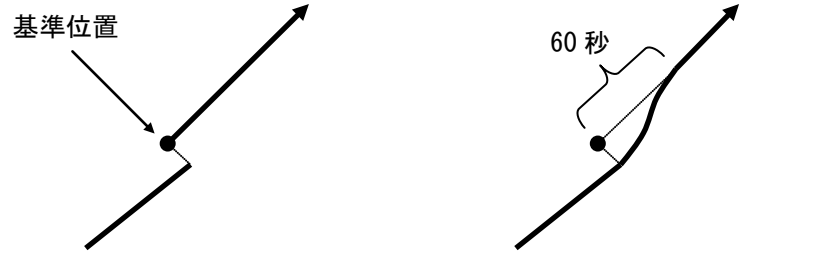

図 7-1 i16DrawTime パラメータによる滑らかな引き込みイメージ

se_DRGetPosition

インクルード

```
#include “se_dr.h”
```

形式

```
int se_DRGetPosition(SE_DR *pstDR, SE_POSITION  *pstPosition)
```

引数

(in)*pstDR*	se_dr ライブラリのハンドラ構造体のポインタ
(out)*pstPosition*	位置情報を受け取る SE_POSITION 構造体のポインタ

戻り値

測位情報の取得に成功した場合は 1 が返されます。それ以外は 0 が返されます。

説明

リアルタイム測位情報を取得し、pstPosition に格納します。

se_DRGetLogPosition

インクルード

```
#include “se_dr.h”
```

形式

```
unsigned int se_DRGetLogPosition(
        FSAREG          *pFsaReg,
        SE_DR           *pstDR,
        SE_POSITION     *pstLogPosition,
        unsigned int    uiSize,
        int             iCorrectEn
)
```

引数

(in)*pFsaReg*	FSA レジスタ構造体のポインタ
(in)*pstDR*	se_dr ライブラリのハンドラ構造体のポインタ
(out)*pstLogPosition*	ロギング位置情報を受け取る SE_POSITION 構造体のポインタ
(in) *iSize*	取得可能なロギング位置情報の数
(in) *iCorrectEn*	位置情報補正の制御フラグ

戻り値

取得したロギング位置情報の数が返されます。

説明

ロギング測位情報を取得し、pstLogPosition に格納します。uiSize には、pstLogPosition に格納可能なロギング位置情報の数を指定します。iCorrectEn に 1 を指定すると、基準位置情報を用いてロギング測位情報が補正されます。0 を指定すると補正はされません。
ロギング位置情報の補正を行う場合、se_DRSetPosition 関数による基準位置を入力した直後に本関数を実行し、すべてのロギング測位情報を取得してください。
なお、pstLogPosition が示すロギング測位情報の格納先は、必ず FSA がアクセス可能なメモリ領域に確保してください。

se_DRSetScaleCorrectGain

インクルード

```
#include “se_dr.h”
```

形式　`void se_DRSetScaleCorrectGain(SE_DR *pstDR, short i16Gain)`

引数

(in)*pstDR*	se_dr ライブラリのハンドラ構造体のポインタ
(in)*i16Gain*	距離補正ゲイン（小数点以下 14 ビット）

戻り値

なし

説明

距離補正ゲインを設定します。距離補正係数は、se_DRSetPosition 実行時に算出する基準位置と自律測位の距離比に、i16Gain で指定した距離補正ゲインを乗じて算出されます。i16Gain に 0 を指定すると se_DRSetPosition 実行時に行う距離補正係数の更新が停止します。

se_DRSetAngleCorrectGain

インクルード

```
#include “se_dr.h”
```

形式　`void se_DRSetAngleCorrectGain(SE_DR *pstDR, short i16Gain)`

引数

(in)*pstDR*	se_dr ライブラリのハンドラ構造体のポインタ
(in)*i16Gain*	移動方位補正ゲイン（小数点以下 14 ビット）

戻り値

なし

説明

移動方位補正ゲインを設定します。移動方位補正角は、se_DRSetPosition 実行時に算出する基準位置と自律測位がなす誤差角に、i16Gain で指定した移動方位補正ゲインを乗じて算出されます。i16Gain に 0 を指定すると se_DRSetPosition 実行時に行う動方位補正角の更新が停止します。

8. 要求メモリ

表 8-1 要求 ROM サイズ（byte）

オブジェクト	C17	C33	FSA
se_drcore	6172	4794	1168
se_dr	4578	2388	1008
Total	10750	7182	2176

表 8-2 要求 RAM サイズ（byte）

オブジェクト	IRAM	FSA-RAM	FSA スタック(*1)
se_drcore	188	1476	50
se_dr	0	1008	44
Total	188	2484(*2)	50

(*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。

(*2) se_dr ライブラリのハンドラ構造体の領域、また、ロギング測位を行う場合は、ロギングバッファのサイズに応じた領域が別途必要になります。

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	2015/04/03	全ページ	新規	新規制定

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　エプソン大阪ビル 15F
TEL（06）6120-6000（代表）　FAX（06）6120-6100

ドキュメントコード：412957300
2015 年 4 月 作成