FSA Sample Program Manual
- Pedestrian Dead Reckoning -
NOTICE

No part of this material may be reproduced or duplicated in any form or by any means without the written permission of Seiko Epson. Seiko Epson reserves the right to make changes to this material without notice. Seiko Epson does not assume any liability of any kind arising out of any inaccuracies contained in this material or due to its application or use in any product or circuit and, further, there is no representation that this material is applicable to products requiring high level reliability, such as, medical products. Moreover, no license to any intellectual property rights is granted by implication or otherwise, and there is no representation or warranty that anything made in accordance with this material will be free from any patent or copyright infringement of a third party. When exporting the products or technology described in this material, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You are requested not to use, to resell, to export and/or to otherwise dispose of the products (and any technical information furnished, if any) for the development and/or manufacture of weapon of mass destruction or for other military purposes.

All brands or product names mentioned herein are trademarks and/or registered trademarks of their respective companies.

©SEIKO EPSON CORPORATION 2015, All rights reserved.
Table of Contents

1. Overview .......................................................................................................................... 1

2. Files Constituting Sample Program ............................................................................. 3

3. Structures.......................................................................................................................... 5
   SE_DR_MOVVEC structure ................................................................................................. 5
   SE_DRC structure ............................................................................................................... 6
   SE_DRC_LOG structure ....................................................................................................... 7
   SE_POSITION structure .................................................................................................... 8
   SE_DR_LOG structure ......................................................................................................... 9
   SE_DR structure ................................................................................................................. 10

4. Macro Definitions .......................................................................................................... 12
   Enabling logging positioning function ............................................................................ 12
   Setting correction lower limit distance ......................................................................... 12
   Setting azimuth correction angle range ......................................................................... 12
   Setting distance correction coefficient range ............................................................... 12
   Configuration of FSA general-purpose libraries .......................................................... 13

5. Control of Sample Program .......................................................................................... 14
   5.1 Initialization .................................................................................................................. 14
   5.2 Operation Settings ........................................................................................................ 14
      Relative moving azimuth setting .................................................................................. 14
      Distance correction gain and azimuth correction gain settings .................................... 15
      Configuration of logging positioning function ............................................................ 15
   5.3 Setting Dead Reckoning Start Position ..................................................................... 15
   5.4 Executing Dead Reckoning ......................................................................................... 15
     Inputting sensor data ...................................................................................................... 15
      Inputting reference position ......................................................................................... 15
   5.5 Obtaining Positioning Values ...................................................................................... 15
      Obtaining realtime positioning values ......................................................................... 15
      Obtaining logging positioning values .......................................................................... 15

6. se_drcore Library API Function Specifications ......................................................... 16
   se_DRCInit ......................................................................................................................... 16
   se_DRCExe ........................................................................................................................... 16
   se_DRCSetLogBuf .............................................................................................................. 17
   se_DRCSetLogMeasInt ...................................................................................................... 17
   se_DRCSetDeclination ...................................................................................................... 18
   se_DRCSetDirEstEn ......................................................................................................... 18
   se_DRCSetMovAngOnBody ............................................................................................... 18
   se_DRCSetCorrectScale ................................................................................................. 19
   se_DRCSetCorrectAngle ................................................................................................. 19
   se_DRCGetMovVec .......................................................................................................... 19
   se_DRCGetLogMovVec ..................................................................................................... 20
   se_DRCGetLogDataNum ................................................................................................. 20
   se_DRCGetState .............................................................................................................. 21
   se_DRCGetWalkStep ......................................................................................................... 22
   se_DRCGetWalkDist ......................................................................................................... 22

7. se_dr Library API Function Specifications ................................................................. 23
   se_DRInitialize .................................................................................................................. 23
   se_DRSetAddress .............................................................................................................. 24
   se_DRGetPosition ............................................................................................................. 25
8. Memory Requirement ........................................................................................................27
Revision History ...................................................................................................................28
1. Overview

This FSA sample program implements a pedestrian dead reckoning function using the acceleration application system library. It consists of the se_drcore library, which is designed to be embedded in a sensor hub microcontroller, and the se_dr library, which is designed to be embedded in the host CPU. However, these libraries are provided as an integrated library capable of being used in both environments.

Figure 1-1 shows the configuration of entire this FSA sample program.

se_drcore library

The se_drcore library inputs 25 Hz or 12.5 Hz 3-axis acceleration data and 3-axis geomagnetism data, and calculates the user’s relative movement vector (North and East components) per a prescribed time. This library does not treat absolute position information typified by latitude/longitude. It also has a function to correct moving distance and direction according to the distance correction coefficient and azimuth correction angle fed back from the se_dr library.

The se_movvec library, which is executed in the se_drcore library to calculate movement vectors, has a function to automatically estimate the user’s relative moving azimuth in the local coordinate system of the terminal from the acceleration data, thus the movement vector can be calculated in any terminal attitude (wearing condition). (* Note, however, that a sufficient accuracy may not be achieved when the terminal is put in a bag or a trouser pocket or worn on a wrist as compared to wearing on chest or waist for measuring body motion during walking with accuracy.)
1. Overview

The se_drcore library uses the acceleration application system libraries (se_accfifo, se_magfifo, se_steplib, and se_movvec) included in the FSA_package_vol.2.

se_dr library

The se_dr library inputs the relative movement vector output from the se_drcore library and the reference position obtained from the GPS or other positioning system, and calculates the current position. It also calculates the distance correction coefficient for correcting moving distance (speed) and correction angle for correcting the azimuth measured in dead reckoning by comparing the position measured in dead reckoning and the currently input reference position with the previously input reference position as the starting point. The se_dr library uses the the general-purpose libraries (fsasqrt, fsatrig, and fsatrig2) included in the FSA_package_vol.1.

Logging positioning function

In addition to the realtime positioning function that obtains the current position at any desired time, this library supports the logging positioning function that accumulates movement vectors in a buffer throughout a period of time and converts them into latitude/longitude at a time when a reference position is input.

Table 1-1  List of se_drcore Library Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>se_DRInit</td>
<td>Performs initialization of the library.</td>
</tr>
<tr>
<td>se_DRCExe</td>
<td>Main function for calculating movement vectors. Call this function in 25 Hz or 12.5 Hz cycles.</td>
</tr>
<tr>
<td>se_DRCSetLogBuf</td>
<td>Configures the logging buffer.</td>
</tr>
<tr>
<td>se_DRCSetLogMeasInt</td>
<td>Sets the logging positioning interval.</td>
</tr>
<tr>
<td>se_DRCSetDeclination</td>
<td>Sets declination of the geomagnetism.</td>
</tr>
<tr>
<td>se_DRCSetDirEstEn</td>
<td>Controls update of the estimated moving direction in the terminal coordinate system.</td>
</tr>
<tr>
<td>se_DRCSetMovAngOnBody</td>
<td>Sets the moving direction with a relative angle in the local coordinate system.</td>
</tr>
<tr>
<td>se_DRCSetCorrectScale</td>
<td>Sets the correction coefficient for correcting the magnitude of movement vectors.</td>
</tr>
<tr>
<td>se_DRCSetCorrectAngle</td>
<td>Sets the correction angle for correcting the direction of movement vectors.</td>
</tr>
<tr>
<td>se_DRCGetMovVec</td>
<td>Obtains the movement vector.</td>
</tr>
<tr>
<td>se_DRCGetLogMovVec</td>
<td>Obtains the movement vectors for logging positioning.</td>
</tr>
<tr>
<td>se_DRCGetLogDataNum</td>
<td>Obtains the number of unread logged movement vectors.</td>
</tr>
<tr>
<td>se_DRCGetState</td>
<td>Obtains state change information (stationary, moving, corner, holding state change).</td>
</tr>
<tr>
<td>se_DRCGetWalkStep</td>
<td>Obtains the cumulative total of the number of steps.</td>
</tr>
<tr>
<td>se_DRCGetWalkDist</td>
<td>Obtains the cumulative total of moving distance.</td>
</tr>
</tbody>
</table>

Table 1-2  List of se_dr Library Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>se_DRInitialize</td>
<td>Performs initialization of the library.</td>
</tr>
<tr>
<td>se_DRSetAddress</td>
<td>Sets the reference latitude/longitude.</td>
</tr>
<tr>
<td>se_DRGGetSetPosition</td>
<td>Obtains the latitude/longitude measured by realtime positioning.</td>
</tr>
<tr>
<td>se_DRGGetLogPosition</td>
<td>Obtains the latitude/longitude measured by logging positioning.</td>
</tr>
<tr>
<td>se_DRSetAddressCorrectGain</td>
<td>Sets the gain for azimuth correction.</td>
</tr>
<tr>
<td>se_DRSetAddressScaleCorrectGain</td>
<td>Sets the gain for distance correction.</td>
</tr>
</tbody>
</table>
2. Files Constituting Sample Program

The se_drcore library consists of the acceleration application system library listed in Table 2-1 and the source files listed in Table 2-2.

### Table 2-1  Object Files Used in se_drcore Libraly

<table>
<thead>
<tr>
<th>Package</th>
<th>Object file</th>
<th>Description</th>
</tr>
</thead>
</table>
| sensfifo | se_accfifo.o  
|          | se_accfifo_fsa.o  
|          | fsa_symbols_accfifo.def               | This object program configures the FIFO buffer for acceleration data, and executes the normalization of acceleration values and the LPF filter processing. |
|          | se_magfifo.o                          | This object program configures the FIFO buffer for geomagnetism data, and executes the LPF filter processing. It converts the geomagnetism coordinates from body coordinates into global coordinates based on the acceleration data to extract the estimation components. |
| stepdet  | se_steplib.o  
|          | se_steplib_fsa.o  
|          | fsa_symbols_steplib.def               | With acceleration data input from accfifo, this object program provides the walking step detection, step counting, walking distance estimation, and holding state change detection functions. |
| movvec   | se_movvec.o  
|          | se_movvec_fsa.o  
|          | fsa_symbols_movvec.def               | This object program decomposes the moving distance output from se_steplib into N and E components based on the relative angle between the terminal coordinates (ENU coordinates) automatically estimated from the acceleration data and moving direction, and the azimuth of the terminal coordinates calculated from the geomagnetism data. |

### Table 2-2  Source Files in se_drcore Library

<table>
<thead>
<tr>
<th>Source file</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>se_drcore.h</td>
<td>This is a header file in which the library functions are declared and structure types are defined.</td>
</tr>
<tr>
<td>se_drcore.c</td>
<td>This is a source file in which the pedestrian dead reckoning library functions are defined.</td>
</tr>
<tr>
<td>fsaleftshift32.c</td>
<td>This is the constant table for 32-bit left shift operation.</td>
</tr>
</tbody>
</table>

The se_dr library consists of the FSA general-purpose libraries listed in Table 2-3 and the source files listed in Table 2-4.

### Table 2-3  FSA General-Purpose Libraries Used in se_dr Library

<table>
<thead>
<tr>
<th>Package</th>
<th>Object file</th>
<th>Description</th>
</tr>
</thead>
</table>
| fsasqrt  | fsasqrt.o  
|          | fsasqrt_fsa.o  
|          | fsa_symbols_fsasqrt.def               | This is the square root library included in the FSA_package_vol.1. The FsaSumSqrt16 and FsaSumSqrt32 functions are used. |
| fsatrig  | fsatrig.o  
|          | fsatrig_fsa.o  
|          | fsa_symbols_fsatrig.def               | This is a trigonometric function library included in the FSA_package_vol.1. The FsaAtan2 function is used. |
| fsatrig2 | fsatrig2.o  
|          | fsatrig2_fsa.o  
|          | fsa_symbols_fsatrig2.def              | This is a trigonometric function library included in the FSA_package_vol.1. The FsaCos32, FsaSin16, and FsaCos16 functions are used. |

### Table 2-4  Source Files in se_dr Library

<table>
<thead>
<tr>
<th>Source file</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>se_dr.h</td>
<td>This is a header file in which the library functions are declared and structure types are defined.</td>
</tr>
<tr>
<td>se_dr.c</td>
<td>This is a source file in which the pedestrian dead reckoning library functions are defined.</td>
</tr>
<tr>
<td>fsaleftshift32.c</td>
<td>This is the constant table for 32-bit left shift operation.</td>
</tr>
</tbody>
</table>
2. Files Constituting Sample Program

In addition to the above, the object files shown below, that contain the files constituting each of the se_drcore and se_dr libraries except fsaleftshift32.c, are provided by the FSA_package_vol.2.

Table 2-3 The Integrated Object Files of se_drcore and se_dr Libraries

<table>
<thead>
<tr>
<th>Package</th>
<th>Object file</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pdr</td>
<td>se_drcore.o</td>
<td>Integrated object files of the se_drcore library</td>
</tr>
<tr>
<td></td>
<td>se_drcore_fsa.o</td>
<td></td>
</tr>
<tr>
<td></td>
<td>fsa_symbols_se_drcore.def</td>
<td></td>
</tr>
<tr>
<td></td>
<td>se_dr.o</td>
<td>Integrated object files of the se_dr library</td>
</tr>
<tr>
<td></td>
<td>se_dr_fsa.o</td>
<td></td>
</tr>
<tr>
<td></td>
<td>fsa_symbols_se_dr.def</td>
<td></td>
</tr>
</tbody>
</table>
3. Structures

**SE_DR_MOVVEC structure**

The SE_DR_MOVVEC structure stores a movement vector output by the realtime and logging positioning functions and a time stamp. Although the structure type is defined in the se_drcore.h, it is used in both the se_dr and se_drcore libraries.

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

- **i16East**
  East component of the movement vector. This is a number with a 4-bit decimal fraction in meter units.

- **i16North**
  North component of the movement vector. This is a number with a 4-bit decimal fraction in meter units.

- **ui16TimeStamp**
  The elapsed time from the last positioning point is stored in 1/25-second units.
3. Structures

SE_DRC structure
The SE_DRC structure is used in the se_drcore library. The structure type is defined in the se_drcore.h and its substance is defined as a global variable in the se_drcore.c.

```c
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]
This variable is used to add up movement vectors for realtime positioning. The East component is stored to ai32Posi[0] and the North component to ai32Posi[1]. Both elements are a number with an 8-bit decimal fraction in meter units.

ui32WalkDist
This variable is used to add up walking distance. This is a number with an 8-bit decimal fraction in meter units.

ui16WalkStep
This variable is used to add up the number of steps.

ui16TimeCount
This variable is used to measure time. It is incremented in 25 Hz cycles.

ui16TimeStamp
This variable is used to store positioning time temporarily. The ui16TimeCount value is copied to this variable when a positioning is performed.

ui16SampleDelay
This variable is used to set the sampling cycle of the input acceleration and geomagnetism sensor values. It is set to 40 (ms) in 25 Hz sampling or 80 (ms) in 12.5 Hz sampling.

ui8TiltState
This variable is used to store presence of holding state change. It is set to 1 when a change of holding state is detected or set to 0 otherwise.

ui8WalkState
This variable is used to store walking state. It is set to 0 in stationary state or set to 1 in walking state.

ui8TurnState
This variable is used to store presence of moving azimuth change. It is set to 1 when a significant change of moving azimuth is detected or set to 0 otherwise.
### SE_DRC_LOG structure

The SE_DRC_LOG structure is used for logging positioning function in the se_drcore library. The structure type is defined in the se_drcore.h and its substance is defined as a global variable in the se_drcore.c.

```c
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]**

This variable is used to add up movement vectors for logging positioning. The East component is stored to ai32Posi[0] and the North component to ai32Posi[1]. Both elements are a number with an 8-bit decimal fraction in meter units.

**pstLogTop**

This variable is used to store the pointer to the beginning of the SE_DR_MOVVEC type logging buffer.

**pstLogEnd**

This variable is used to store the pointer to the end of the SE_DR_MOVVEC type logging buffer.

**pstWritePos**

This variable is used to store the pointer to the write position of the SE_DR_MOVVEC type logging buffer.

**pstReadPos**

This variable is used to store the pointer to the read position of the SE_DR_MOVVEC type logging buffer.

**ui16TimeCount**

This variable is used to measure time. It is incremented in 25 Hz cycles.

**ui16TimeStamp**

This variable is used to store positioning time temporarily. The ui16TimeCount value is copied to this variable when a positioning is performed.

**ui16LogSize**

This variable is used to store the number of positioning points that can be recorded in the SE_DR_MOVVEC type logging buffer.

**ui16UnReadNum**

This variable is used to store the number of unread positioning points in the SE_DR_MOVVEC type logging buffer.

**ui16MeasInt**

This variable is used to store the logging positioning interval in ms units.
3. Structures

SE_POSITION structure

The SE_POSITION structure stores the position information treated in the se_dr library. The structure type is defined in the se_dr.h.

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

i32latitude

This variable is used to store latitude. The value to be stored is a number with a 30-bit decimal fraction and is within -2.0 to +2.0 to which -180 to +180 degrees are normalized.

i32longitude

This variable is used to store longitude. The value to be stored is a number with a 30-bit decimal fraction and is within -2.0 to +2.0 to which -180 to +180 degrees are normalized.

ui16Speed

This variable is used to store horizontal speed. The value to be stored is a number with a 4-bit decimal fraction in m/s units.

i16Bearing

This variable is used to store a horizontal moving azimuth. The value to be stored is a number with a 14-bit decimal fraction and is within -2.0 to +2.0 to which -180 to +180 degrees are normalized.

i64Utc

This variable is used to store UTC time in ms units.
3. Structures

**SE_DR_LOG structure**

The SE_DR_LOG structure is used for logging positioning functions in the se_dr library. The structure type is defined in the se_dr.h.

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

**stRefPosition**

This structure variable is used to store reference position information for logging positioning. The contents are updated when the se_DRSetPosition function is executed.

**i32CorrectScale**

This variable is used to store the distance correction coefficient for logging positioning. This is a number with a 14-bit decimal fraction.

**i16CorrectAngle**

This variable is used to store the azimuth correction angle for logging positioning. The value to be stored is a number with a 14-bit decimal fraction and is within -2.0 to +2.0 to which -180 to +180 degrees are normalized.

**i16Reserved**

This is a dummy variable for alignment.

**ui16State**

This variable is used to manage the reference position setting status. It is set to 0 at initialization and is set to 1 when the first reference position is set.

**ui16LogSize**

This variable is used to store the number of logging positioning data obtained in the se_DRGetLogPosition function.
3. Structures

SE_DR structure
The SE_DR structure is used in the se_dr library. The structure type is defined in the se_dr.h. The substance of this structure should be allocated to outside the library.

```c
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
This structure variable is used to store the latest position information by the dead reckoning. The contents are updated when the se_DRGetPosition function is executed.

stRefPosition
This structure variable is used to store the latest reference position information. The contents are updated when the se_DRSetPosition function is executed.

ai32MovVec[2]
This variable is used to store the current position in movement vector using stRefPosition as the starting point. The ai32MovVec[0] stores the North component and the ai32MovVec[1] stores the East component. Both elements are a number with a 4-bit decimal fraction in meter units.

ai32ErrPosi[2]
This variable is used to store the difference between the reference position set in the se_DRSetPosition function and the dead reckoning point at the time at an error. The value to be stored is a number with a 30-bit decimal fraction and is within -2.0 to +2.0 to which -180 to +180 degrees are normalized. The ai32ErrPosi[0] stores a latitude error and the ai32ErrPosi[1] stores a longitude error.

i32DrawTime
This variable is used to set the time to gradually reflect the positioning error set to the ai32ErrPosi[2] to the subsequent dead reckoning outputs. The value by dividing 0x40000000 by the argument i32DrawTime of the se_DRSetPosition function is set to this variable. However, if the argument i32DrawTime is 0, 0 is set to this variable.

i32CorrectScale
This variable is used to store the distance correction coefficient. This is a number with a 14-bit decimal fraction.
i16CorrectAngle

This variable is used to store the azimuth correction angle. The value to be stored is a number with a 14-bit decimal fraction and is within -2.0 to +2.0 to which -180 to +180 degrees are normalized.

i16ScaleCorrectGain

This variable is used to store the distance correction gain set in the se_DRSetScaleCorrectGain function. This is a number with a 14-bit decimal fraction.

i16AngleCorrectGain

This variable is used to store the azimuth correction gain set in the se_DRSetAngleCorrectGain function. This is a number with a 14-bit decimal fraction.

i16ErrGain

This variable is used to store the gain to reflect the positioning error set to the ai32ErrPosi[2] to the subsequent dead reckoning outputs. This is a number with a 14-bit decimal fraction.

i16GainTheta

This variable is used to calculate i16ErrGain. The value to be stored is a number with a 14-bit decimal fraction and is within -2.0 to +2.0 to which -π to +π are normalized. This variable is set to +π/2 when the se_DRSetPosition function is executed. Then the angle is reduced every time the se_DRGetPosition function is executed. Finally it becomes -π/2 when the time set to the i32DrawTime has elapsed. The i16ErrGain is calculated by the equation below using the i16GainTheta.

\[
i16ErrGain = \frac{1.0 + \sin(i16GainTheta)}{2.0}
\]

i16State

This variable is used to manage the reference position setting status. It is set to 0 at initialization and is set to 1 when the reference position that is effective for calculating the distance correction coefficient and azimuth correction angle is set.

stDRLog

This SE_DR_LOG structure variable stores the variables used for logging positioning function.
4. Macro Definitions

Enabling logging positioning function
The logging positioning function can be disabled by commenting the #define definition line in the se_drcore.h shown below out. If this line is commented out, the functions and heap area required for logging positioning are excluded from compilation objects.

Definition in se_drcore.h

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

Setting correction lower limit distance
This macro sets the minimum distance for calculating the distance correction coefficient and azimuth correction angle. The correction coefficient and correction angle are not updated if the distance in a straight line and moving distance by dead reckoning between reference positions are shorter than this setting. The value should be set in meter units.

Definition in se_dr.h

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

Setting azimuth correction angle range
These macros set the effective azimuth correction angle range. If the difference (error) in azimuth between the reference positioning and dead reckoning is out of the range set here, the correction angle is clipped to the upper or lower limit value.

Definition in se_dr.h

```c
// 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
```

Setting distance correction coefficient range
These macros set the distance correction coefficient range. If the value obtained by subtracting 1.0 from the distance ratio of reference positioning to dead reckoning is out of the range set here, the correction coefficient is clipped to the upper or lower limit value.

Definition in se_dr.h

```c
// Setting of correction scale limitation
#define SE_DR_CORRECT_SCALE_UPPRT_LIMIT +0.2
#define SE_DR_CORRECT_SCALE_LOWER_LIMIT -0.2
```
Configuration of FSA general-purpose libraries

By commenting out the definitions of the functions that are not used in the se_dr library as shown below, they can be excluded from compilation objects. However, do not comment out a function definition if the function is used in other embedded applications.

In the fsasqrt.h, comment out all function definitions except for _FSASUMSQRT16 and _FSASUMSQRT32.

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

In the fsatrig.h, comment out all function definitions except for _FSAATAN2.

```
// Function selection
// Disable the definitions except the function you will use.
#define _FSA
#define _FSA
#define _FSA
#define _FSA
#define _FSA
#define _FSA
```

In the fsatrig2.h, comment out only the _FSASIN32 definition.

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

5. Control of Sample Program

5.1 Initialization

Call the se_DRInitialize function to initialize the se_dr library. At this initialization, set the sampling cycle of the acceleration data and geomagnetism data, and scale factor of the acceleration data to be input to the se_DRCExe function. For more information, see the description of the se_DRInitialize function under Chapter 7, “se_dr Library API Function Specifications.”

Note that it is not necessary to call the se_DRCInit function separately to initialize the se_drcore library, as the se_DRCInit function is executed in the se_DRInitialize function.

5.2 Operation Settings

Relative moving azimuth setting

To determine a moving azimuth in dead reckoning, two pieces of information are required, absolute azimuth of the terminal local coordinate system in the global coordinate system and relative moving azimuth that points the moving direction of the user in the terminal local coordinate system. This library estimates the former using geomagnetism data and determines the relative moving azimuth of the latter by one of the two methods shown below.

[Performing automatic estimation]

The relative moving azimuth can automatically be estimated from acceleration data. When using this method, call the se_DRCSetDirEstEn function with the argument set to 1. The automatic estimation function is enabled when the se_DRInitialize function is executed. Calling the se_DRCSetDirEstEn function with the argument set to 0 stops the update operation with the latest moving azimuth automatically estimated. For more information, see the description of the se_DRCSetDirEstEn function under Chapter 6, “se_drcore Library API Function Specifications.”

[Specifying relative moving azimuth directly]

When the moving azimuth in the terminal local coordinate system is known, for example, when the terminal wearing method is predetermined, set the relative moving azimuth directly using the se_DRCSetMovAngOnBody function. Specifying the relative moving azimuth disables the automatic estimation function. For more information, see the description of the se_DRCSetMovAngOnBody function under Chapter 6, “se_drcore Library API Function Specifications.”
5. Control of Sample Program

Distance correction gain and azimuth correction gain settings
The distance correction gain and the azimuth correction gain should be set using the `se_DRSetScaleCorrectGain` function and the `se_DRSetAngleCorrectGain` function, respectively. The correction gains are both set to 0 when the `se_DRInitialize` function is executed. Set these correction gains when correcting distance and azimuth using reference positioning data. Although the correction gains can be set at any time including while the dead reckoning is being executed, the settings take effect when the `se_DRSetPosition` function is executed. For more information, see the descriptions of the `se_DRSetScaleCorrectGain` and `se_DRSetAngleCorrectGain` functions under Chapter 7, “se_dr Library API Function Specifications.”

Configuration of logging positioning function
When performing logging positioning, the buffer for storing logging positioning data and logging positioning intervals shown below must be configured.

**[Logging buffer configuration]**
Configure the buffer for storing logging positioning data with the size specified using the `se_DRCSetLogBuf` function. For more information, see the description of the `se_DRCSetLogBuf` function under Chapter 6, “se_drcore Library API Function Specifications.”

**[Logging positioning interval setting]**
Set the logging positioning interval using the `se_DRCSetLogMeasInt` function. For more information, see the description of the `se_DRCSetLogMeasInt` function under Chapter 6, “se_drcore Library API Function Specifications.”

5.3 Setting Dead Reckoning Start Position
Set the dead reckoning start position using the `se_DRSetPosition` function. Before dead reckoning can be started, the first reference position that becomes the starting point must be set.

5.4 Executing Dead Reckoning

**Inputting sensor data**
After setting the first reference position as the start position, call the `se_DRCExe` function repeatedly by passing acceleration data and geomagnetism data in the sampling cycle specified at initialization to perform dead reckoning. For more information, see the description of the `se_DRCExe` function under Chapter 6, “se_drcore Library API Function Specifications.”

**Inputting reference position**
Performing the dead reckoning for a long time using the sensors only increases the positioning error gradually. Therefore, reset the reference position at appropriate times using the `se_DRSetPosition` function.

5.5 Obtaining Positioning Values

**Obtaining realtime positioning values**
The realtime positioning values can be obtained by calling the `se_DRGetPosition` function when it is necessary to perform positioning. For more information, see the description of the `se_DRGetPosition` function under Chapter 7, “se_dr Library API Function Specifications.”

**Obtaining logging positioning values**
Generally, the logging positioning values should be obtained by calling the `se_DRGetLogPosition` function immediately after the `se_DRSetPosition` function is executed. For more information, see the description of the `se_DRGetLogPosition` function under Chapter 7, “se_dr Library API Function Specifications.”
6. se_drcore Library API Function Specifications

se_DRCInit

Header file to be included:
#include “se_drcore.h”

Format:
void se_DRCInit(unsigned short ui16SampleDelay, unsigned short ui16AccScaleFactor)

Argument:
(in) ui16SampleDelay Accelerator and geomagnetism data sampling cycles
(in) ui16AccScaleFactor Acceleration scale factor

Return value:
None

Description:
This function initializes the SE_DRC structure variable and the acceleration application system library to be used. With the arguments ui16SampleDelay and ui16AccScaleFactor, specify the sampling cycle, either 40 (25 Hz) or 80 (12.5 Hz), and the acceleration scale factor value equivalent to 1G, respectively. This function is executed in the se_DRInitialize function.

se_DRCExe

Header file to be included:
#include “se_drcore.h”

Format:
int se_DRCExe(FSAREG *pFsaReg, short ai16Acc[3], short ai16Mag[3])

Argument:
(in) pFsaReg Pointer to the FSA register structure
(in) ai16Acc[3] 3-axis acceleration data
(in) ai16Mag[3] 3-axis geomagnetism data

Return value:
This function returns 1 if the movement vector is updated and 0 otherwise.

Description:
This is the main function to perform pedestrian dead reckoning. Call this function by passing acceleration data and geomagnetism data in the sampling cycle specified using the se_DRCInit. The acceleration data should be passed by setting the X-axis, Y-axis, and Z-axis data in the ENU coordinate system to ai16Acc[0], ai16Acc[1], and ai16Acc[2], respectively. The geomagnetism data should be passed by set the X-axis, Y-axis, and Z-axis data in the ENU coordinate system to ai16Mag[0], ai16Mag[1], and ai16Mag[2], respectively.
6. *se_drcore* Library API Function Specifications

### se_DRCSetLogBuf

Header file to be included:

```c
#include "se_drcore.h"
```

Format:

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

Argument:

- **(in) astLogBuf** Pointer to the array for storing the logged movement vectors
- **(in) uiLogSize** Size of the astLogBuf array

Return value:

None

Description:

This function sets the pointer to the SE_DR_MOVVEC type array for storing the logged movement vectors and the array size. It is not necessary to call this function if logging positioning is not performed.

### se_DRCSetLogMeasInt

Header file to be included:

```c
#include "se_drcore.h"
```

Format:

```c
void se_DRCSetLogMeasInt(unsigned int ui16LogMeasInt)
```

Argument:

- **(in) ui16LogMeasInt** Logging positioning interval (unit: 1/25 seconds)

Return value:

None

Description:

This function sets the logging positioning interval. When the argument is specified as 0, logging positioning is not performed. At an initialization by the se_DRCInit function, the logging positioning interval is set to 0. It is not necessary to call this function if logging positioning is not performed.

### se_DRCSetDeclination

Header file to be included:

```c
#include "se_drcore.h"
```

Format:

```c
void se_DRCSetDeclination(short i16Angle)
```

Argument:

- **(in) i16Angle** Declination of geomagnetism (with a 7-bit decimal fraction, unit: degree)

Return value:

None

Description:

This function sets the declination of geomagnetism (difference between true north and magnetic north).
### se_DRCSetDirEstEn

**Header file to be included:**
```
#include "se_drcore.h"
```

**Format:**
```
void se_DRCSetDirEstEn(int iEnable)
```

**Argument:**
- **(in) iEnable**: Automatic relative moving azimuth estimation enable flag

**Return value:**
None

**Description:**
To automatically estimate the relative moving azimuth in the local coordinate system (ENU coordinate system) of the terminal, set 1 to `iEnable`. If the automatic estimation function is enabled, the relative moving azimuth is automatically estimated from acceleration data while the user is in walking state. If the automatic estimation is not performed, set 0 to `iEnable`. At an initialization by the `se_DRCInit` function, the automatic estimation function is enabled. When the automatic estimation function is disabled immediately after an initialization, the relative moving azimuth and moving direction configured are 0 degrees and the plus direction of Y-axis, respectively. When it is disabled after performing automatic estimation for a while, the relative moving azimuth before being disabled is retained.

### se_DRCSetMovAngOnBody

**Header file to be included:**
```
#include "se_drcore.h"
```

**Format:**
```
void se_DRCSetMovAngOnBody(short i16Angle)
```

**Argument:**
- **(in) i16Angle**: Relative moving azimuth (a number with a 14-bit decimal fraction in which -180 to +180 degrees are normalized into -2.0 to +2.0)

**Return value:**
None

**Description:**
This function sets the angle specified using `i16Angle` directly as the relative moving azimuth in the local coordinate system of the terminal. Executing this function disables the automatic estimation function. If the user’s terminal wearing method can be specified, set the specified angle using this function. Table 6-1 shows the relationship between the angle setting and the moving azimuth in the ENU terminal coordinate system.

#### Table 6-1  Relationship Between Angle Setting and Moving Azimuth

<table>
<thead>
<tr>
<th>Angle setting</th>
<th>Relative moving azimuth</th>
</tr>
</thead>
<tbody>
<tr>
<td>0°</td>
<td>Y-axis plus direction</td>
</tr>
<tr>
<td>+90°</td>
<td>X-axis plus direction</td>
</tr>
<tr>
<td>-180°</td>
<td>Y-axis minus direction</td>
</tr>
<tr>
<td>-90°</td>
<td>X-axis minus direction</td>
</tr>
</tbody>
</table>
6. se_drcore Library API Function Specifications

**se_DRCSetCorrectScale**

Header file to be included:
```
#include "se_drcore.h"
```

Format:
```
void se_DRCSetCorrectScale(unsigned short ui16Scale)
```

Argument:
- **(in) ui16Scale**
  Distance correction coefficient (a number with a 14-bit decimal fraction)

Return value:
None

Description:
This function sets the distance correction coefficient. After this function is executed, walking distance will be corrected to the value multiplied by the coefficient specified using `ui16Scale`.

**se_DRCSetCorrectAngle**

Header file to be included:
```
#include "se_drcore.h"
```

Format:
```
void se_DRCSetCorrectAngle(short i16Angle)
```

Argument:
- **(in) i16Angle**
  Moving azimuth correction angle (a number with a 14-bit decimal fraction in which -180 to +180 degrees are normalized into -2.0 to +2.0)

Return value:
None

Description:
This function sets the azimuth correction angle. After this function is executed, moving azimuth will be corrected to the value added by the angle specified using `i16Angle`.

**se_DRCGetMovVec**

Header file to be included:
```
#include "se_drcore.h"
```

Format:
```
void se_DRCGetMovVec(SE_DR_MOVVEC *pstMovVec)
```

Argument:
- **(out) pstMovVec**
  Pointer to the SE_DR_MOVVEC type structure

Return value:
None

Description:
This function obtains the movement vectors from the last execution of the se_DRCMovVec function to the present time. For details of the SE_DR_MOVVEC structure variable, see Chapter 3, “Structures.”
6. se_drcore Library API Function Specifications

se_DRCGetLogMovVec

Header file to be included:
#include “se_drcore.h”

Format:
unsigned short se_DRCGetLogMovVec(SE_DR_MOVVEC astMovVec[], unsigned short ui16Num)

Argument:
(out) astMovVec Pointer to the SE_DR_MOVVEC type structure array
(in) ui16Num Number of logged movement vectors that can be obtained

Return value:
This function returns the number of logged movement vectors obtained.

Description:
This function obtains the logged movement vectors and stores them to astMovVec. With ui16Num, specify the number of logged movement vectors that can be accepted (the number of astMovVec array elements allocated). The actual obtained number of the logged movement vectors is returned as the return value. The logged movement vectors obtained by this function become read state and cannot be obtained again.

se_DRCGetLogDataNum

Header file to be included:
#include “se_drcore.h”

Format:
unsigned short se_DRCGetLogDataNum()

Argument:
None

Return value:
This function returns the number of unread logged movement vectors.

Description:
This function obtains the number of unread logged movement vectors. Check the number of logged movement vectors entered in the buffer using this function, and obtain logging positioning data using the se_DRCGetLogMovVec function before an overflow occurs in the buffer.
se_DRCGetState

Header file to be included:

```c
#include "se_drcore.h"
```

Format:

```c
int se_DRCGetState()
```

Argument:

None

Return value:

This function returns a value listed in Table 6-2 depending on the state change detected in dead reckoning. If multiple state changes occur at the same time, this function returns the sum total of the respective values.

<table>
<thead>
<tr>
<th>Value</th>
<th>State detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not detected.</td>
</tr>
<tr>
<td>1</td>
<td>Change in attitude (change in holding state) is detected.</td>
</tr>
<tr>
<td>2</td>
<td>Transition to stationary state is detected.</td>
</tr>
<tr>
<td>4</td>
<td>Transition to walking state is detected.</td>
</tr>
<tr>
<td>8</td>
<td>Change in moving azimuth (corner) is detected.</td>
</tr>
</tbody>
</table>

Description:

This function is used to check a state change during dead reckoning. After a state change is obtained by this function, perform a processing according to the state detected. Note that a state change from stationary state is output after transiting to a walking state.
6. se_drcore Library API Function Specifications

se_DRCGetWalkStep

Header file to be included:
#include “se_drcore.h”

Format:
unsigned short se_DRCGetWalkStep()

Argument:
None

Return value:
This function returns the cumulative total of steps.

Description:
This function obtains the cumulative total of steps and then clears it to 0.

se_DRCGetWalkDist

Header file to be included:
#include “se_drcore.h”

Format:
unsigned long se_DRCGetWalkDist()

Argument:
None

Return value:
This function returns the cumulative total of walking distance (with an 8-bit decimal fraction, unit: meter).

Description:
This function obtains the cumulative total of walking distance and then clears it to 0.
7. se_dr Library API Function Specifications

**se_DRInitialize**

Header file to be included:
```
#include “se_dr.h"
```

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

Argument:
- **(in) pstDR**
  Pointer to the handler structure of the se_dr library
- **(in) ui16SampleDelay**
  Acceleration and geomagnetism data sampling cycles
- **(in) ui16AccScaleFactor**
  Acceleration scale factor

Return value:
None

Description:
This function initializes the se_dr library. *pstDR* is the handler of the se_dr library. Pass the pointer to this function to perform initialization. With the arguments *ui16SampleDelay* and *ui16AccScaleFactor*, specify the sampling cycle, either 40 (25 Hz) or 80 (12.5 Hz), and the acceleration scale factor value equivalent to 1G, respectively.

This function also executes the se_DRCInit function, so the se_drcore library is initialized as well. The substance of *pstDR* must be arranged in a memory area that can be accessed from the FSA.
7. **se_dr Library API Function Specifications**

### se_DRSetPosition

**Header file to be included:**

```c
#include "se_dr.h"
```

**Format:**

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

**Argument:**

- (in) `pFsaReg` Pointer to the FSA register structure
- (in) `pstDR` Pointer to the handler structure of the se_dr library
- (in) `i16Declination` Declination of the geomagnetism (with a 7-bit decimal fraction, unit: degree)
- (in) `pstPosition` Pointer to the SE_POSITION structure in which the position information is stored
- (in) `i16DrawTime` Time to draw into the setting position (unit: 1/25 seconds)

**Return value:**

This function returns 1 when the distance correction coefficient or azimuth correction angle is updated, and 0 otherwise.

**Description:**

This function sets the position information stored at `pstPosition` as the reference position. At the same time it passes the declination of the geomagnetism at the setting position to `i16Declination`. `i16DrawTime` is the parameter to control the time to draw the dead reckoning position into the reference position. For example, when `i16DrawTime` is set to 60 seconds, the dead reckoning draws a local so that an error from the reference position set will be eliminated for 60 seconds. When `i16DrawTime` is set to 0, the dead reckoning draws immediately the current position into the reference position set.

![Smooth Drawing Image by i16DrawTime Parameter](image-url)
7. se_dr Library API Function Specifications

se_DRGetPosition

Header file to be included:
#include “se_dr.h”

Format:
int se_DRGetPosition(SE_DR *pstDR, SE_POSITION pstPosition)

Argument:
(in) pstDR Pointer to the handler structure of the se_dr library
(out) pstPosition Pointer to the SE_POSITION structure that receives the position information

Return value:
This function returns 1 if the positioning information is successively obtained and 0 otherwise.

Description:
This function obtains the realtime positioning information and stores it to pstPosition.

se_DRGetLogPosition

Header file to be included:
#include “se_dr.h”

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

Argument:
(in) pFsaReg Pointer to the FSA register structure
(in) pstDR Pointer to the handler structure of the se_dr library
(out) pstLogPosition Pointer to the SE_POSITION structure that receive the logged position information
(in) uiSize Number of logged position information that can be obtained
(in) iCorrectEn Position information correction control flag

Return value:
This function returns the number of logged position information obtained.

Description:
This function obtains the logged position information and stores it to pstLogPosition. With uiSize, specify the number of logged position information that can be stored at pstLogPosition. When 1 is specified to iCorrectEn, the logged position information is corrected using the reference position information. The correction is not performed if 0 is specified.
When correcting the logged position information, call this function to obtain all logged position information immediately after inputting the reference position using the se_DRSetPosition function.
The logged position information storing area that is pointed by pstLogPosition must be arranged within a memory area that can be accessed from the FSA.
7. se_dr Library API Function Specifications

se_DRSetScaleCorrectGain

Header file to be included:
#include “se_dr.h”

Format:
void se_DRSetScaleCorrectGain(SE_DR *pstDR, short i16Gain)

Argument:
(in) pstDR Pointer to the handler structure of the se_dr library
(in) i16Gain Distance correction gain (with a 14-bit decimal fraction)

Return value:
None

Description:
This function sets the distance correction gain. The distance correction coefficient is calculated by multiplying the distance ratio of the reference position calculated at executing the se_DRSetPosition function to dead reckoning position by the distance correction gain specified with i16Gain. If i16Gain is specified as 0, update of the distance correction coefficient that is performed at executing the se_DRSetPosition function is disabled.

se_DRSetAngleCorrectGain

Header file to be included:
#include “se_dr.h”

Format:
void se_DRSetAngleCorrectGain(SE_DR *pstDR, short i16Gain)

Argument:
(in) pstDR Pointer to the handler structure of the se_dr library
(in) i16Gain Moving azimuth correction gain (with a 14-bit decimal fraction)

Return value:
None

Description:
This function sets the moving azimuth correction gain. The moving azimuth correction angle is calculated by multiplying the error angle between the reference position calculated at executing the se_DRSetPosition function and dead reckoning position by the moving azimuth correction gain specified with i16Gain. If i16Gain is specified as 0, update of the moving azimuth correction angle that is performed at executing the se_DRSetPosition function is disabled.
## 8. Memory Requirement

### Table 8-1 ROM Size Requirement (byte)

<table>
<thead>
<tr>
<th>Object</th>
<th>C17</th>
<th>C33</th>
<th>FSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>se_drcore</td>
<td>6,172</td>
<td>4,794</td>
<td>1,168</td>
</tr>
<tr>
<td>se_dr</td>
<td>4,578</td>
<td>2,388</td>
<td>1,008</td>
</tr>
<tr>
<td>Total</td>
<td>10,750</td>
<td>7,182</td>
<td>2,176</td>
</tr>
</tbody>
</table>

### Table 8-2 RAM Size Requirement (byte)

<table>
<thead>
<tr>
<th>Object</th>
<th>IRAM</th>
<th>FSA-RAM</th>
<th>FSA stack(^{(1)})</th>
</tr>
</thead>
<tbody>
<tr>
<td>se_drcore</td>
<td>188</td>
<td>1,476</td>
<td>50</td>
</tr>
<tr>
<td>se_dr</td>
<td>0</td>
<td>1,008</td>
<td>44</td>
</tr>
<tr>
<td>Total</td>
<td>188</td>
<td>2,484(^{(2)})</td>
<td>50</td>
</tr>
</tbody>
</table>

\(^{(1)}\) The FSA stack memory, for which the top of the memory that can be accessed from the FSA is allocated, is used by the FSA as a temporary work area.

\(^{(2)}\) In addition to this size, an area for the handler structure of the se_dr library and a logging buffer area when performing logging positioning are separately required.
## Revision History

<table>
<thead>
<tr>
<th>Rev. No.</th>
<th>Date</th>
<th>Page</th>
<th>Category</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rev 1.0</td>
<td>2015/04/03</td>
<td>All</td>
<td>New</td>
<td>New establishment</td>
</tr>
</tbody>
</table>
AMERICA
EPSON ELECTRONICS AMERICA, INC.
214 Devcon Drive,
San Jose, CA 95112, USA
Phone: +1-800-228-3964 FAX: +1-408-922-0238

EUROPE
EPSON EUROPE ELECTRONICS GmbH
Riesstrasse 15, 80992 Munich,
GERMANY
Phone: +49-89-14005-0 FAX: +49-89-14005-110

ASIA
EPSON (CHINA) CO., LTD.
7F, Jinbao Bldg., No.89 Jinbao St.,
Dongcheng District,
Beijing 100005, CHINA
Phone: +86-10-8522-1199 FAX: +86-10-8522-1125

SHANGHAI BRANCH
7F, Block B, Hi-Tech Bldg., 900 Yishan Road,
Shanghai 200233, CHINA
Phone: +86-21-5423-5577 FAX: +86-21-5423-4677

SHENZHEN BRANCH
12F, Dawning Mansion, Keji South 12th Road,
Hi-Tech Park, Shenzhen 518057, CHINA
Phone: +86-755-2699-3628 FAX: +86-755-2699-3838

EPSON HONG KONG LTD.
Unit 715-723, 7/F Trade Square, 681 Cheung Sha Wan Road,
Kowloon, Hong Kong.
Phone: +852-2585-4600 FAX: +852-2827-4346

EPSON TAIWAN TECHNOLOGY & TRADING LTD.
14F, No. 7, Song Ren Road,
Taipei 110, TAIWAN
Phone: +886-2-8786-6688 FAX: +886-2-8786-6660

EPSON SINGAPORE PTE., LTD.
1 HarbourFront Place,
#03-02 HarbourFront Tower One, Singapore 098633
Phone: +65-6586-5500 FAX: +65-6271-3182

SEIKO EPSON CORP.
KOREA OFFICE
19F, KLI 63 Bldg., 60 Yoido-dong,
Youngdeungpo-Ku, Seoul 150-763, KOREA
Phone: +82-2-784-6027 FAX: +82-2-767-3677

SEIKO EPSON CORP.
MICRODEVICES OPERATIONS DIVISION
Device Sales & Marketing Department
421-6, Hino, Hino-shi, Tokyo 191-8501, JAPAN
Phone: +81-42-587-5814 FAX: +81-42-587-5117