AS9888 API Manual
Version 1.0.0.823

November 29, 2011

ALSEN Technology Inc.

Contents

1. Library revision record 4
2. Introduction 4
3. Development Environment 4
4. File Configuration 4
5. AKEC_AS9888.h File 5
5.1. Constant 5
5.1.1. AKEC_BDATA_SIZE 5
5.1.2. AKEC_HDATA_SIZE 5
5.1.3. AKEC_ADATA_SIZE 5
5.1.4. AKEC_HSENSE_DEFAULT, AKEC_HSENSE_TARGET 5
5.1.5. AKEC_ASENSE_DEFAULT, AKEC_ASENSE_TARGET 5
5.2. Enumerate 5
5.2.1. AS9888_MSMODE 5
5.3. Function 5
5.3.1. AKEC_InitDecompAS9888 5
5.3.2. AKEC_DecompAS9888 6
6. AKEC_Device.h File 7
6.1. Constant 7
6.1.1. AKEC_EPSILON 7
6.1.2. AKEC_FLTMAX 7
6.1.3. AKEC_ERROR 7
6.1.4. AKEC_SUCCESS 7
6.2. Type definition 7
6.2.1. int16 7
6.2.2. uint8 8
6.2.3. uint16 8
6.2.4. AKFLOAT 8
6.3. Structure 8
6.3.1. int16vec 8
6.3.2. AKFVEC 8
7. AKEC_Direction.h File 8
7.1. Enumerate 8
7.1.1. AKEC_PATNO 8
7.2. Function 9
7.2.1. AKEC_Direction 9
8. AKEC_DOE.h File 11
8.1. Constant 11
8.1.1. AKEC_HBUF_SIZE 11

8.1.2. AKEC_HOBUF_SIZE 11
8.2. Structure 11
8.2.1. AKEC_DOE_VAR 11
8.3. Function 11
8.3.1. AKEC_InitDOE 11
8.3.2. AKEC_DOE 12
9. AKMETS.h File 12
9.1. Constant 12
9.2. Structure 12
9.3. Function 13
9.3.1. AKEC_InitMETSCal 13
9.3.2. AKEC_METSCal 14
10. AKEC_VNorm.h File 15
10.1. Function 15
10.1.1. AKEC_VbNorm 15
10.1.2. AKEC_VbAve 16
11. libFST_AS9888.h File 16
11.1. Constant 16
11.2. Structure 17
11.3. Function Pointer 18
11.3.1. DEV_TX_DATA 18
11.3.2. DEV_RX_DATA 18
11.3.3. DEV_SET_MODE 18
11.3.4. DEV_GET_DATA 19
11.3.5. DEV_RESET 19
11.4. Function 19
11.4.1. AKEC_GetAccelerometerParam 19
11.4.2. AKEC_Test_And_Compensate 20

1. Library revision record

Release Date	Revision	Library change record (revise, modify、change)
2011/07/06	0.8.0.628	First version.
2011/08/24	1.0.0.823	Add METS.h and libFST_AS9888.h section.

2. Introduction

This document describes the functions used to calculate the azimuthal angle based on the magnetic data provided by AS9888.

Hereafter, this library is called as “DMT Compass library”.

3. Development Environment

These library programs have been compiled and checked for operation under the following development environment.

* OS: Windows XP SP3
* Compiler: Visual Studio 2005 SP1

4. File Configuration

This library contains the files shown in Table 1.

Table 1. File Configuration

File name	Description
libAS9888.lib	The library file.
AKEC_AS9888.h	Includes the definitions that depend on the device.
AKEC_Configure.h	Includes the definitions of library configuration.
AKEC_Device.h	Includes the definitions that form the basis for all other files, such as integer type.
AKEC_Direction.h	Includes the functions associated with the azimuthal angle calculation.
AKEC_DOE.h	Includes the functions associated with the DOE calculation.
AKEC_Math.h	Includes the definitions that define the mathematical constant value.

AKEC_Version.h	Includes the functions that return this library version.
AKEC_VNorm.h	Includes the functions that associated with the normalization and average process.

5. AKEC_AS9888.h File

5.1. Constant

5.1.1. AKEC_BDATA_SIZE

Number of elements in the array that holds the data retrieved from AS9888.

5.1.2. AKEC_HDATA_SIZE

The number of elements in the array that stores measured data of magnetic sensors.

5.1.3. AKEC_ADATA_SIZE

The number of elements in the array that stores measured data of acceleration sensors.

5.1.4. AKEC_HSENSE_DEFAULT, AKEC_HSENSE_TARGET

Default/Target sensitivity data for magnetic sensor.

5.1.5. AKEC_ASENSE_DEFAULT, AKEC_ASENSE_TARGET

Default/Target sensitivity data for acceleration sensor.

5.2. Enumerate

5.2.1. AS9888_MSMODE

Enumerate type variable that shows measurement mode.

```
typedef enum _AS9888_MSMODE {
    AKEC_SNG = 0,
    AKEC_CONT = 1
} AS9888_MSMODE;
```

5.3. Function

5.3.1. AKEC_InitDecompAS9888

Functionality:

Initializes the buffers required to run the function AKEC_DecompAS9888.

Definition:

```
void AKEC_InitDecompAS9888(
    const int16          nhdata,
          AKFVEC         hdata[AKEC_HDATA_SIZE],
    const int16          nadata,
          AKFVEC         adata[AKEC_ADATA_SIZE]
);
```

Return value:

None.

Argument:

nhdata

(Input) The size of the array hdata[ ].

hdata[ ]

(Output) Pointer to the structure array that holds magnetic data to be passed to the function AKEC_DecompAS9888.

nadata

(Input) The size of the array adata[ ].

adata[ ]

(Output) Pointer to the structure array that holds acceleration data to be passed to the function AKEC_DecompAS9888.

5.3.2. AKEC_DecompAS9888

Functionality:

Extracts the measured sensor data for each axis from one block of measured values read out from AS9888.

Definition:

```
int16 AKEC_DecompAS9888(
  const   uint8          bdata[AKEC_BDATA_SIZE],
  const   AS9888_MSMODE  msmode,
  const   uint8          EHC[3],
          AKFLOAT*       temperature,
  const   int16          nhdata,
          AKFVEC         hdata[AKEC_HDATA_SIZE],
  const   int16          nadata,
          AKFVEC         adata[AKEC_ADATA_SIZE],
          uint8*         st1,
          uint8*         st2
);
```

Return value:

AKEC_ERROR: DRDY bit or DERR bit is active.

AKEC_SUCCESS: Completed normally.

Argument:

bdata[ ]

(Input) Pointer to the array that holds one block data of measured values read out from AS9888.

The number of elements for this array must be equal to AKEC_BDATA_SIZE.

msmode

(Input) Variable that indicates the AS9888 operation mode.

EHC[ ]

(Input) Pointer to the array that holds the sensitivity adjustment coefficient of the sensor. Specify the value read from EEPROM (EHCX, EHCY, EHCZ) of AS9888.

**temperature*

(Output) Pointer to the variable that holds temperature data. The temperature is in Celsius degree.

nhdata

(Input) The size of the array hdata[ ].

hdata[ ]

(Input/Output) Pointer to the structure array that holds the magnetic data, which sensitivity is adjusted.

nadata

(Input) The size of the array adata[ ].

adata[ ]

(Input/Output) Pointer to the structure array that holds the acceleration data.

**st1*

(Output) Pointer to the variable which holds the ST1 register value.

**st2*

(Output) Pointer to the variable which holds the ST2 register value.

6. AKEC_Device.h File

6.1. Constant

6.1.1. AKEC_EPSILON

The minimum positive floating point value.

6.1.2. AKEC_FLTMAX

The maximum positive floating point value.

6.1.3. AKEC_ERROR

It represents fail.

6.1.4. AKEC_SUCCESS

It represents success.

6.2. Type definition

6.2.1. int16

16-bit signed integer.

6.2.2. uint8

8-bit unsigned integer.

6.2.3. uint16

16-bit unsigned integer.

6.2.4. AKFLOAT

Single/Double precision floating value.

6.3. Structure

6.3.1. int16vec

3D vectors consisting of signed 16-bit integers

```
typedef union _int16vec{
    struct {
        int16    x;
        int16    y;
        int16    z;
    }u;
    int16    v[3];
} int16vec;
```

6.3.2. AKFVEC

3D vectors consisting of floating point numbers

```
typedef union _AKFVEC{
    struct {
        AKFLOAT    x;
        AKFLOAT    y;
        AKFLOAT    z;
    }u;
    AKFLOAT v[3];
} AKFVEC;
```

7. AKEC_Direction.h File

7.1. Enumerate

7.1.1. AKEC_PATNO

Enumerate type variable that shows layout of mobile terminal and sensor.

```
typedef enum _AKEC_PATNO {
    PAT_INVALID = 0,
    PAT1,
    PAT2,
    PAT3,
```

```
    PAT4,
    PAT5,
    PAT6,
    PAT7,
    PAT8
} AKEC_PATNO;
```

7.2. Function

7.2.1. AKEC_Direction

Functionality:

Using normalized magnetic data (called as magnetic vector) buffer and normalized acceleration data (called as acceleration vector) buffer, calculates the azimuthal angle and the terminal tilting posture. The unit and coordinate system follows the Android definition.

Definition:

```
int16 AKEC_Direction(
    const   int16       nhvec,
    const   AKFVEC      hvec[],
    const   int16       hnave,
    const   int16       navec,
    const   AKFVEC      avec[],
    const   int16       anave,
    const   AKEC_PATNO  layout,
            AKFLOAT*    azimuth,
            AKFLOAT*    pitch,
            AKFLOAT*    roll
);
```

Return value:

AKEC_ERROR: The azimuth angle cannot be calculated.

AKEC_SUCCESS: Completed normally.

Argument:

nhvec

(Input) The size of the array hvec[ ].

hvec[ ]

(Input) Pointer to the structure array that holds the normalized magnetic vector data.

hnave

(Input) The number of magnetic vector data to be averaged.

navec

(Input) The size of the array avec[ ].

avec[ ]

(Input) Pointer to the structure array that holds the normalized acceleration vector data.

anave

(Input) The number of acceleration vector data to be averaged.

layout

(Input) This parameter specifies the layout of mobile terminal and sensor. The relationship between AKEC_PATNO and mounting direction is shown in Table 2.

**azimuth*

(Output) Pointer to the variable that holds the azimuth angle. The value is in degree.

**pitch*

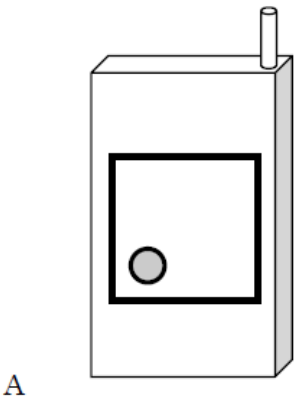
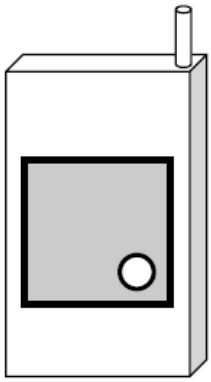
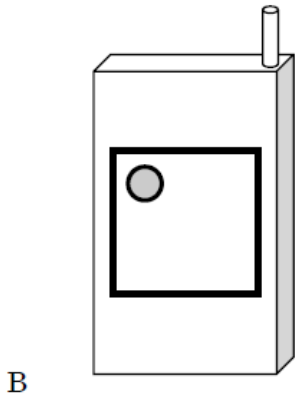
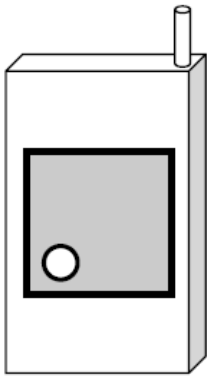
(Output) Pointer to the variable that holds the pitch angle. The value is in degree.

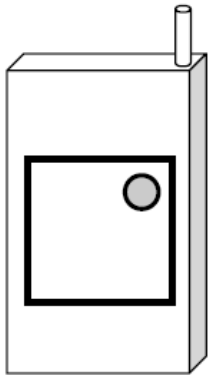
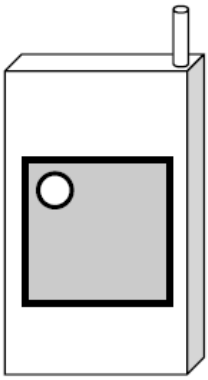
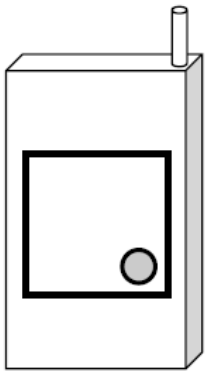
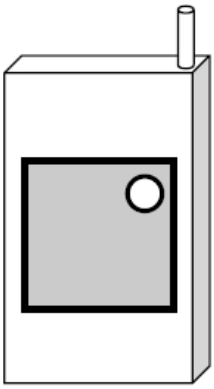
**roll*

(Output) Pointer to the variable that holds the roll angle. The value is in degree.

Table 2: Relationship between AKEC_PATNO and mounting direction.

Gray device represents that the device is mounted on backside of the equipment.

Mounting	Option	Mounting	Option
 A	PAT1	 E	PAT5
 B	PAT2	 F	PAT6

C 	PAT3	G 	PAT7
D 	PAT4	H 	PAT8

8. AKEC_DOE.h File

8.1. Constant

8.1.1. AKEC_HBUF_SIZE

Defines the buffer size to store magnetic data for DOE.

8.1.2. AKEC_HOBUF_SIZE

Defines the buffer size to store estimated offset data for DOE.

8.2. Structure

8.2.1. AKEC_DOE_VAR

Structure that collects variables necessary for DOE calculation.

```
typedef struct _AKEC_DOE_VAR{
    AKFVEC      hbuf[AKEC_HBUF_SIZE];
    AKFVEC      hobuf[AKEC_HOBUF_SIZE];
    AKFLOAT     hrdoe;
} AKEC_DOE_VAR;
```

8.3. Function

8.3.1. AKEC_InitDOE

Functionality:

Initializes the AKEC_DOE_VAR structure. To ensure that the function AKEC_DOE operates

correctly, the AKEC_DOE_VAR structure must be always initialized by this function.

Definition:

```
void AKEC_InitDOE(
    AKEC_DOE_VAR*       hdoev
);
```

Return value:

None

Argument:

**hdoev*

(Output) Pointer to the AKEC_DOE_VAR structure variable used for DOE calculation

8.3.2. AKEC_DOE

Functionality:

This function estimates the magnetic offset data from the magnetic data.

Definition:

```
int16 AKEC_DOE(
                AKEC_DOE_VAR*       hdoev,
    const       AKFVEC*             hdata,
                AKFVEC*             ho
);
```

Return value:

AKEC_ERROR: ho is not updated.

AKEC_SUCCESS: ho is updated.

Argument:

**hdoev*

(Input/Output) Pointer to the AKEC_DOE_VAR structure variable.

hdata[ ]

(Input) Pointer to the structure array that holds magnetic data.

**ho*

(Output) Pointer to the structure variable that holds the magnetic offset data. If the return value of the function is AKEC_ERROR, this value has not changed.

9. AKMETS.h File

9.1. Constant

9.1.1. AKEC_ME_LVNUM

The number of activity levels.

9.2. Structure

9.2.1. AKEC_EXTH

This structure stores the calculation results of AKEC_METSCal function.

```
typedef union _AKEC_EXTH{
```

```
    struct {
        AKFLOAT     ex1;
        AKFLOAT     ex2;
        AKFLOAT     ex3;
        AKFLOAT     ex4;
    }u;
    AKFLOAT    v[4];
} AKEC_EXTH
```

9.3. Function

9.3.1. AKEC_InitMETSCal

Functionality :

Initializes the AKEC_EXTH structure. To ensure that the function AKEC_METSCal operates correctly, the AKEC_EXTH structure must be always initialized by this function.

Definition :

```
void AKSC_InitMETSCal(
        int16           pMEST[],
        int32           n[],
        int32           p[],
        AKSC_EXTH       *exth,
        AKSC_EXTH       *exthp
```

Return value :

None.

Argument :

pMEST [ ]

(Output) Pointer to the int16 type array which stores registers value (from MEST1 to MEST9). The number of elements must be 9.

n[ ]

(Output) Pointer to the int32 type array which stores data classified based on the activity level. The number of elements must be AKSC_ME_LVNUM.

p[ ]

(Output) Pointer to the int32 type array which stores summation of difference between levels. The number of elements must be AKSC_ME_LVNUM.

**exth*

(Output) Pointer to the AKEC_EXTH structure. The structure includes accumulation of intensity of physical activity classified based on activity level (not include Basal Metabolic Rate).

**exthp*

(Output) Pointer to the AKEC_EXTH structure. The structure includes accumulation of intensity of physical activity classified based on activity level (include Basal Metabolic Rate).

9.3.2. AKEC_METSCal

Functionality :

This function calculates accumulation of intensity of physical activity.

Definition :

```
int16 AKSC_METSCal(
    const   int16           MEST[],
            int16           pMEST[],
            int32 n[],
            int32 p[],
            AKSC_EXTH *exth,
            AKSC_EXTH *exthp
);
```

Return value :

0: Calculation result overflowed.

1: Completed normally.

Argument :

MEST [ ]

(Input) A pointer to the int16 type array which stores the latest registers value (from MEST1 to MEST9). The number of elements must be 9.

pMEST [ ]

(Input/Output) A pointer to the int16 type array which stores the previous MEST registers value. The number of elements must be 9. When this function finishes, the value stored in MEST array is copied to this array.

n[ ]

(Input/Output) A pointer to the int32 type array which stores data classified based on the activity level. The number of elements must be AKSC_ME_LVNUM.

p[ ]

(Input/Output) Pointer to the int32 type array which stores summation of difference between levels. The number of elements must be AKSC_ME_LVNUM.

**exth*

(Output) Pointer to the AKEC_EXTH structure. The structure includes accumulation of intensity of physical activity classified based on activity level (not include Basal Metabolic Rate).

**exthp*

(Output) Pointer to the AKEC_EXTH structure. The structure includes accumulation of intensity of physical activity classified based on activity level (include Basal Metabolic Rate).

10. AKEC_VNorm.h File

10.1. Function

10.1.1. AKEC_VbNorm

Functionality:

Substitutes offset data from measured data and then divide by sensitivity data and multiplied by normalized sensitivity data (This operation is called as Normalization on this document.). In other words, execute the following calculation formula.

$$vvec = \frac{vdata - o}{s} \times tgt$$

This function applies the normalization operation to the first nbuf data in vdata[ ] buffer.

Definition:

```
int16 AKEC_VbNorm(
    const   int16       ndata,
    const   AKFVEC      vdata[],
    const   int16       nbuf,
    const   AKFVEC*     o,
    const   AKFVEC*     s,
    const   AKFLOAT     tgt,
    const   int16       nvec,
            AKFVEC      vvec[]
);
```

Return value:

AKEC_ERROR: The measured data can't be normalized.

AKEC_SUCCESS: Completed normally.

Argument:

ndata

(Input) The size of the array vdata[ ].

vdata[ ]

(Input) Pointer to the structure array that holds the measured data.

nbuf

(Input) The number of vector data to be normalized.

**o*

(Input) Pointer to the structure variable that holds the offset data.

**s*

(Input) Pointer to the structure variable that holds the sensitivity data of the sensor.

tgt

(Input) Specify the normalized sensitivity data. In a typical case, specify AKEC_HSENSE_TARGET or AKEC_ASENSE_TARGET.

nvec

The size of the array vvec[ ].

vvec[ ]

(Output) Pointer to the structure array that holds the normalized vector data. The data scale of magnetic data acquired from AS9888 is 0.3 μT/LSB. Therefore, the value vdata can be converted to the magnetic flux density unit μT by following equation. b refers to the converted value.

$$b = \frac{vdata}{1} \times 0.3$$

10.1.2. AKEC_VbAve

Functionality:

Calculate average from a vector buffer.

Definition:

```
int16 AKEC_VbAve(
    const   int16       nvec,
    const   AKFVEC      vvec[],
    const   int16       nave,
            AKFVEC*     vave
);
```

Return value:

AKEC_ERROR: The average vector can't be calculated.

AKEC_SUCCESS: Completed normally.

Argument:

nvec

(Input) The size of the array vvec[ ].

vvec[ ]

(Input) Pointer to the structure array that holds the normalized vector.

nave

(Input) The number of vector data to be averaged.

**vave*

(Output) Pointer to the structure variable that holds the averaged vector data.

11. libFST_AS9888.h File

11.1. Constant

11.1.1. FST_TEST_FAIL

It represents the test has failed.

11.1.2. FST_TEST_PASS

It represents the test has been succeeded.

11.1.3. FST_ERROR

It represents system error was occurred.

11.1.4. FST_NUMOF_STEP

It represents the total number of steps of function tests.

11.1.5. FST_MODE_CONTINUE

When this mode is selected, the function test program will continue to the end regardless of the result of an each test item.

11.1.6. FST_MODE_STOP_ON_FAIL

When this mode is selected, the function test program will return immediately if any test item has failed.

11.1.7. FST_DMT_TRT

It represents normal temperature of factory shipment test.

11.1.8. FST_DMT_THT

It represents high temperature of factory shipment test.

11.2. Structure

11.2.1. DEV_HANDLE

Structure that holds function pointers to access a device driver.

```
typedef struct _DEV_HANDLE {
    DEV_TX_DATA     txData;
    DEV_RX_DATA     rxData;
    DEV_SET_MODE    setMode;
    DEV_GET_DATA    getData;
    DEV_RESET       reset;
} DEV_HANDLE;
```

11.2.2. FST_RESULT

Structure that holds a result of each test item.

```
typedef struct _FST_RESULT {
    int8        status;
    uint8       step;
    uint8       no;
    int16       data;
} FST_RESULT;
```

11.2.3. FST_COMP

Structure that holds parameters to calculate sensitivity and offset from current temperature.

```
typedef struct _FST_COMP{
    AKFLOAT     t9eRt;
    AKFLOAT     t9eLv;
```

```
    AKFVEC      kst;
    AKFVEC      st;
    AKFVEC      kot;
    AKFVEC      ot;
} FST_COMP;
```

11.3. Function Pointer

11.3.1. DEV_TX_DATA

Functionality:

Write data to the device.

Definition:

```
typedef int16(*DEV_TX_DATA)(const uint8, const uint8 *, const uint16);
```

Return value:

When function succeeded, 1 is returned. When it failed, 0 is returned.

Argument:

1st

(Input) Specifies the address of the first byte to be written.

2nd

(Input) Specifies a pointer of an array, which stores data to be written.

3rd

(Input) Specifies the number of bytes to be written (n).

11.3.2. DEV_RX_DATA

Functionality:

Read data from the device.

Definition:

```
typedef int16(*DEV_RX_DATA)(const uint8, uint8 *, const uint16);
```

Return value:

When function succeeded, 1 is returned. When it failed, 0 is returned.

Argument:

1st

(Input) Specify the address of the first byte to be obtained.

2nd

(Output) Specifies a pointer of an array, to which data obtained from specified address are stored.

3rd

(Input) Specify the number of bytes to be (n) obtained.

11.3.3. DEV_SET_MODE

Functionality:

Set operation mode of the device. If invalid value is specified to the argument, this function fails.

Definition:

```
typedef int16(*DEV_SET_MODE)(const uint8);
```

Return value:

When function succeeded, 1 is returned. When it failed, 0 is returned.

Argument:

1st

(Input) Specify a operation mode. The value must be a valid number, which can be set to the MS (62H) register.

11.3.4. DEV_GET_DATA

Functionality:

This function obtains a measurement data (from INT1ST to ST4) when DRDY pin is changed to HIGH. When a measurement data is not the latest, this function blocks until measurement is done. If DRDY pin does not change for the fixed period of time, this function fails.

Definition:

```
typedef int16(*DEV_GET_DATA)(uint8*);
```

Return value:

When function succeeded, 1 is returned. When it failed, 0 is returned.

Argument:

1st

(Input) Specifies a pointer of an array, to which the obtained data are stored. The size of the array must be SENSOR_DATA_SIZE or more.

11.3.5. DEV_RESET

Functionality:

Reset the device.

Definition:

```
typedef int16(*DEV_RESET)(void);
```

Return value:

When function succeeded, 1 is returned. When it failed, 0 is returned.

11.4. Function

11.4.1. AKEC_GetAccelerometerParam

Functionality:

Calculate accelerometer parameters, i.e. offset and sensitivity of accelerometer, from FST_COMP parameter and current temperature. FST_COMP parameter should be obtained by calling AKEC_Test_And_Compensate function.

Definition:

```
void AKEC_GetAccelerometerParam(
    const    FST_COMP*    cmp,
    const    AKFLOAT      curTemperature,
             AKFVEC*      as,
             AKFVEC*      ao
```

```
);
```

Return value:

None

Argument:

**cmp*

(Input) Specify a pointer to FST_COMP structure.

curTemperature

(Input) Specify a current temperature in Celsius scale.

**as*

(Output) Specify a pointer to AKFVEC structure. When this function returns, sensitivity of accelerometer is set to the structure.

**ao*

(Output) Specify a pointer to AKFVEC structure. When this function returns, offset of accelerometer is set to the structure.

11.4.2. AKEC_Test_And_Compensate

Functionality:

This function executes function test and calculate parameters, which are used to calculate accelerometer parameters. If this function fails, the FST_COMP parameter cannot be filled with valid value, so that the parameter cannot be used for AKEC_GetAccelerometerParam function.

Definition:

```
int16 AKEC_Test_And_Compensate(
    const    DEV_HANDLE*    handle,
    const    int16          mode,
    const    int16          nave,
    const    uint8          filter,
             FST_RESULT     result[FST_NUMOF_STEPS],
             FST_COMP*      cmp
);
```

Return value:

FST_ERROR: handle is invalid.

FST_TEST_FAIL: One or more test item has failed.

FST_TEST_PASS: Completed normally.

Argument:

**handle*

(Input) Specify a pointer to a DEV_HANDLE structure. Valid function pointers should be set to the each element of the structure.

mode

(Input) Specify a test mode, i.e. FST_MODE_CONTINUE or FST_MODE_STOP_ON_FAIL.

nave

(Input) Specify a number of averages. This value should be within 1 and 32.

filter

(Input) Specify a filter enable. If 0 is specified, filter function is disabled. Otherwise the function is enabled.

result[ ]

(Output) Specify a pointer to a FST_RESULT structure array. The size of this array should be larger than FST_NUMOF_STEPS.

**cmp*

(Output) Specify a pointer to a FST_COMP structure. When this function succeeds, this structure is filled with parameters of accelerometer.