Freescale Semiconductor
Application Note

Document Number:AN4706
Rev 2.0, 5/2013

Accelerometer and Magnetometer Selection and Configuration

by: Mark Pedley

Contents

1 Introduction

This Application Note contains recommendations for the selection and configuration of accelerometer and magnetometer sensors for use with Freescale's Xtrinsic eCompass and magnetic calibration software provided under license at www.freescale.com/ecompass. This Application Note is part of the technical documentation for that software and its use and distribution are controlled by that license agreement.

1.1 Summary

This document provides recommendations for the selection and configuration of the accelerometer and magnetometer sensors in tilt-compensated eCompass applications.

2 Recommended Devices

2.1 Accelerometer

Feature	MMA8451Q	MMA8452Q	MMA8453Q	MMA8652FC	MMA8653FC
Package	3mm x 3mm x 1mm QFN16	3mm x 3mm x 1mm QFN16	3mm x 3mm x 1mm QFN16	2mm x 2mm x 1mm DFN10	2mm x 2mm x 1mm DFN10
Range	±2 *g*, ±4 *g*, ±8 *g*	±2 *g*, ±4 *g*, ±8 *g*	±2 *g*, ±4 *g*, ±8 *g*	±2 *g*, ±4 *g*, ±8 *g*	±2 *g*, ±4 *g*, ±8 *g*
Resolution	14 bits	12 bits	10 bits	12 bits	10 bits
Rate	1.56 to 800 Hz	1.56 to 800 Hz	1.56 to 800 Hz	1.56 to 800 Hz	1.56 to 800 Hz
0 *g* offset post board mount	±30 m*g*	±30 m*g*	±30 m*g*	±33.5 m*g*	±60 m*g*

2.2 Magnetometer

Feature	MAG3110FC
Package	2 mm x 2 mm x 0.85 mm DFN10
Range	±1000 µT
Resolution	16 bits, 0.1 µT
Rate	80 Hz
Zero flux offset	±1000 µT

2.3 Integrated accelerometer and magnetometer

Feature	FXOS8700CQ (Accelerometer)	FXOS8700CQ (Magnetometer)
Package	3 mm x 3 mm x 1.2 mm QFN16	
Range	±2 g, ±4 g, ±8 g	±1200 uT
Rate	400 Hz	400 Hz
Resolution	14 bits	16 bits, 0.1 uT
Offset	±30 mg zero g offset post board mount	±10 uT zero flux offset

3 Configuration

3.1 Data rate

Output data rates for handheld eCompass applications are typically 20 Hz to 50 Hz. Data rates above this offer little advantage since the mathematics of the tilt-compensated eCompass assume that the eCompass is held steadily with no linear acceleration present so that the accelerometer reading is only a function of the orientation angles.

The compile time constant `DELTAT` is set for a 50 Hz sampling rate (0.02 s sampling interval) in the reference software but should be set to the sampling rate in the final system. The low pass filter on the output orientation angles has a cutoff defined by `ANGLE_LPF_FPU`. This is set to a default all-pass value of 1.0 for use with the sensor simulation function but will typically have a value of Nyquist/16 or approximately 16 samples when used with real sensors. Increasing this value increases the level of low pass filtering of the estimated orientation at the expense of increased latency.

```
// orientation LPF cutoff = Nyquist / ANGLE_LPF_FPU
// so about ANGLE_LPF_FPU samples response
// ANGLE_LPF_FPU equal to 1.0F is valid and corresponds to all pass.
#define ANGLE_LPF_FPU 1.0F
// sampling interval (secs)
#define DELTAT 0.02F
```

3.2 Accelerometer

The accelerometer should be configured into the ±2 *g* range for handheld applications. The compile time constant `FCOUNTSPERG` in the file include.h should be matched to the part and range selected.

```
// MMA8451 and FXOS8700 provides 4096 counts per g in 2g mode
#define FCOUNTSPERG 4096.0
```

3.3 Magnetometer

The MAG3110FC and FXOS8700CQ magnetometers have a fixed range and fixed resolution. The compile time constants below in the include.h need not be changed for these devices.

```
// MAG3110 and FXOS8700 provide 10 counts / uT
#define FCOUNTSPERUT 10.0F
#define FUTPERCOUNT 0.1F
```

It is extremely important to configure the magnetometer sensor to minimize internal low pass filtering (over-sampling) for a given noise level. The magnetometer readings are distorted by soft iron effects caused by ferromagnetic components on the circuit board and will be distributed over the surface on the soft iron ellipsoid. Enabling excessive averaging of the magnetometer readings within the magnetometer sensor will average the distribution ellipsoid towards a sphere and result in an inaccurate soft iron estimate. In general, the highest ADC setting should be used in the magnetometer sensor. Recommendations are given for the MAG3110FC and FXOS8700CQ below.

3.4 MAG3110FC specific recommendations

- The AUTO_MRST_EN bit in CTRL_REG2 must be explicitly enabled by the user's software to enable periodic degaussing of the MAG3110FC magnetic sensor. The MAG3110FC powers up with this bit disabled.
- Table 30 in the MAG3110FC data sheet lists the possible configurations of the sensor for over-sampling ratios, data rates and noise levels. The settings corresponding to an ADC rate of 1280Hz are recommended (first four rows) to reduce smearing of the soft iron ellipsoid.
- The first magnetic reading output from the MAG3110FC sensor after power up is zero. To avoid corruption of the magnetometer measurement buffer, cautious users of the Xtrinsic software may wish to add the test below to the control loop code. In practice, this is unlikely ever to be an issue since the second measurement will almost certainly correspond to the same magnetometer data buffer bin and immediately over-write the earlier measurement.

```
// update the magnetometer measurement buffer integer magnetometer data
if (loopcounter >= 1)
  fUpdateMagnetometerBuffer(&thisMagneticBuffer, &thisMag, &thisAccel, loopcounter);
```

3.5 FXOS8700CQ specific recommendations

- Table 96 in the FXOS8700CQ datasheet lists the internal oversampling ratios used to average the magnetometer measurements. Higher over-sampling ratios will offer lower measurement noise but large ratios may average the soft iron ellipsoid towards a sphere and should be avoided.
- The first magnetic reading from the FXOS8700CQ after power up is, like the MAG3110FC, zero. The same test as described above may also be applied to FXOS8700CQ software:

```
// update the magnetometer measurement buffer integer magnetometer data
if (loopcounter >= 1)
  fUpdateMagnetometerBuffer(&thisMagneticBuffer, &thisMag, &thisAccel, loopcounter);
```

4 Example FXOS8700CQ Driver Code

4.1 Introduction

It is very straightforward to configure the FXOS8700CQ and start receiving data from the three accelerometer and three magnetometer channels. Unfortunately, since every hardware platform will be different, it is not possible to provide completely portable software drivers. This section therefore provides real FXOS8700CQ driver code for a Kinetis uC board running under the MQX operating system. The I^2C functions `s_i2c_read_regs` and `s_i2c_write_regs` are not provided here and should be replaced with the corresponding low level I^2C driver code on the development platform.

4.2 FXOS8700CQ Addresses

This section lists the I^2C address of the FXOS8700CQ. The I^2C address depends on the logic level of FXOS8700CQ pins SA0 and SA1, therefore the I^2C address may be 0x1C, 0x1D, 0x1E or 0x1F. See the FXOS8700CQ datasheet for further details.

```
// FXOS8700 I2C address
#define FXOS8700CQ_SLAVE_ADDR     0x1E    // with pins SA0=0, SA1=0
```

Some of the key FXOS8700CQ internal register addresses are listed below.

```
// FXOS8700 internal register addresses
#define FXOS8700CQ_STATUS         0x00
#define FXOS8700CQ_WHOAMI         0x0D
#define FXOS8700CQ_XYZ_DATA_CFG   0x0E
#define FXOS8700CQ_CTRL_REG1      0x2A
#define FXOS8700CQ_M_CTRL_REG1    0x5B
#define FXOS8700CQ_M_CTRL_REG2    0x5C
#define FXOS8700CQ_WHOAMI_VAL     0xC7
```

4.3 FXOS8700CQ Configuration Function

This function configures the FXOS8700CQ into 200 Hz hybrid mode, meaning that both accelerometer and magnetometer data are provided at the 200 Hz rate. The code is self-explanatory and can be easily customized for different settings.

```
// function configures FXOS8700 combination accelerometer and magnetometer sensor
int16 s_FXOS8700CQ_start(MQX_FILE_PTR aFP)
{
 uint8_t databyte;

 // read and check the FXOS8700 WHOAMI register
 if (s_i2c_read_regs(aFP, FXOS8700CQ_SLAVE_ADDR, FXOS8700CQ_WHOAMI, &databyte, (uint8_t) 1) !
= 1)
 {
     return (I2C_ERROR);
 }
 if (databyte != FXOS8700CQ_WHOAMI_VAL)
 {
     return (I2C_ERROR);
 }

 // write 0000 0000 = 0x00 to accelerometer control register 1 to place FXOS8700 into standby
 // [7-1] = 0000 000
 // [0]: active=0
 databyte = 0x00;
 if (s_i2c_write_regs(aFP, FXOS8700CQ_SLAVE_ADDR, FXOS8700CQ_CTRL_REG1, &databyte, (uint8_t)
1) != 1)
 {
     return (I2C_ERROR);
 }

 // write 0001 1111 = 0x1F to magnetometer control register 1
 // [7]: m_acal=0: auto calibration disabled
 // [6]: m_rst=0: no one-shot magnetic reset
 // [5]: m_ost=0: no one-shot magnetic measurement
 // [4-2]: m_os=111=7: 8x oversampling (for 200Hz) to reduce magnetometer noise
 // [1-0]: m_hms=11=3: select hybrid mode with accel and magnetometer active
 databyte = 0x1F;
 if (s_i2c_write_regs(aFP, FXOS8700CQ_SLAVE_ADDR, FXOS8700CQ_M_CTRL_REG1, &databyte,
(uint8_t) 1) != 1)
 {
     return (I2C_ERROR);
 }
```

Example FXOS8700CQ Driver Code

```
 // write 0010 0000 = 0x20 to magnetometer control register 2
 // [7]: reserved
 // [6]: reserved
 // [5]: hyb_autoinc_mode=1 to map the magnetometer registers to follow the accelerometer
registers
 // [4]: m_maxmin_dis=0 to retain default min/max latching even though not used
 // [3]: m_maxmin_dis_ths=0
 // [2]: m_maxmin_rst=0
 // [1-0]: m_rst_cnt=00 to enable magnetic reset each cycle
 databyte = 0x20;
 if (s_i2c_write_regs(aFP, FXOS8700CQ_SLAVE_ADDR, FXOS8700CQ_M_CTRL_REG2, &databyte,
(uint8_t) 1) != 1)
 {
     return (I2C_ERROR);
 }

 // write 0000 0001= 0x01 to XYZ_DATA_CFG register
 // [7]: reserved
 // [6]: reserved
 // [5]: reserved
 // [4]: hpf_out=0
 // [3]: reserved
 // [2]: reserved
 // [1-0]: fs=01 for accelerometer range of +/-4g range with 0.488mg/LSB
 databyte = 0x01;
 if (s_i2c_write_regs(aFP, FXOS8700CQ_SLAVE_ADDR, FXOS8700CQ_XYZ_DATA_CFG, &databyte,
(uint8_t) 1) != 1)
 {
     return (I2C_ERROR);
 }

 // write 0000 1101b = 0x0D to accelerometer control register 1
 // [7-6]: aslp_rate=00
 // [5-3]: dr=001=1 for 200Hz data rate (when in hybrid mode)
 // [2]: lnoise=1 for low noise mode
 // [1]: f_read=0 for normal 16 bit reads
 // [0]: active=1 to take the part out of standby and enable sampling
 databyte = 0x0D;
 if (s_i2c_write_regs(aFP, FXOS8700CQ_SLAVE_ADDR, FXOS8700CQ_CTRL_REG1, &databyte, (uint8_t)
1) != 1)
 {
     return (I2C_ERROR);
 }

 // normal return
 return (I2C_OK);
}
```

4.4 FXOS8700CQ Data Read Function

This function performs a block read of 13 bytes comprising the status byte, the three 16 bit accelerometer channels and the three 16 bit magnetometer channels and places the measurements into the `thisAccel` and `thisMag` structures.

Note that this function assumes that the `hyb_autoinc_mode` bit has been set to map the magnetometer registers to follow the accelerometer registers to permit a single block read.

```
// read status and the three channels of accelerometer and magnetometer data from FXOS8700CQ
(13 bytes)
int16 ReadFXOS8700CQData(struct AccelSensor *pthisAccel, struct MagSensor *pthisMag)
{
    MQX_FILE_PTR fp;            // I2C file pointer
    uint8_t Buffer[13];         // read buffer for the 13 bytes read

    // read 13 bytes (status byte and the six channels of 16 bit data)
    if (s_i2c_read_regs(fp, FXOS8700CQ_SLAVE_ADDR, FXOS8700CQ_STATUS, Buffer, 13) == 13)
    {
        pthisAccel->iGpx = (Buffer[1] << 8) | Buffer[2];
        pthisAccel->iGpy = (Buffer[3] << 8) | Buffer[4];
        pthisAccel->iGpz = (Buffer[5] << 8) | Buffer[6];
        pthisMag->iBpx = (Buffer[7] << 8) | Buffer[8];
        pthisMag->iBpy = (Buffer[9] << 8) | Buffer[10];
        pthisMag->iBpz = (Buffer[11] << 8) | Buffer[12];

        // normal return
        return (I2C_OK);
    }
    else
    {
        // return with error
        return (I2C_ERROR);
    }
}
```

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number AN4706
Rev 2.0, 5/2013
21 May 2013