Workshop: Freescale Sensor Fusion Library for Kinetis MCUs
Lendo os movimentos da IoT
Alessandro Carmona | FAE
dec. 2014

Agenda

- Part 1: Motion Sensors Overview
  - Part 2: Movement and Orientation
  - Part 3: Introduction to Sensor Fusion
  - Part 4: Freescale Sensor Fusion Toolbox
  - Part 5: Lab #1 – Play with fusion options
- Part 6: Freescale Sensor Fusion Library
  - Part 7: Lab #2 – Build the embedded firmware
- Part 8: Optional Lab #3 – Make some changes
- Part 9: Odds & Ends and Wrap-up

Sensor Portfolio

- Pressure
- Accelerometer
- Magnetometer
- Gyroscope
- Sensing systems

Freescale Microcontrollers Overview

Kinetis Microcontrollers Family

- Ultra Low Power
- Cost
- Integration
- NXP
- Integration
- NXP
Ultra-low power/cost features in low pin-count packages.

- Freescale Bundled IDE, RTOS & Middleware - Rapid prototyping Platform - Broad ARM Ecosystem Support

MCU families with ARM Cortex-M0+ for use in high reliability applications.

- Kinetis M Series single chip smart meter conversion.
- Kinetis V Series advanced memory and feature integration

Kinetis Design Studio (KDS)

- No-cost integrated development environment (IDE) for Kinetis MCUs
- Eclipse and GCC-based IDE for C/C++ editing, compiling and debugging
- Customer Application, Middleware, Application Framework, Connectivity, Security, Power, Display, Audio, Video, etc.

Learn more at www.freescale.com/KDS (Jan/Feb 2014)

Freescale IDE Options (www.freescale.com/kids)

- Feature/IDEs:
  - Processor Expert
  - QorIQ Qonverge
  - QUICC Engine
  - Processor Expert with Kinetis SDK
  - FreeRTOS
  - MQX
  - GCC-based IDE for C/C++ editing, compiling and debugging

Support for SEGGER, P&E and OpenSource Linux (Ubuntu, Redhat, Centos)
Includes Processor Expert with Kinetis SDK
A free of charge and unlimited IDE for Kinetis
Based on Eclipse, GCC, GDB and other open-source environments
No-cost integrated and debugging (TCP/IP, USB)
Operating System
Application Specific Awareness (i.e. MQX, FreeRTOS)

Kinetis IDE Options (www.freescale.com/kids)
Kinetis IDE Comparison

<table>
<thead>
<tr>
<th>Feature</th>
<th>Freescale</th>
<th>ARM</th>
<th>ARMKEL</th>
<th>IAR</th>
<th>SEGGER</th>
<th>CMSIS-DAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>NXP</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Targeted Devices</td>
<td>NXP</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Tools</td>
<td>TrueSTUDIO, MDK Lite, MDK KickStart, Wearables</td>
<td>TrueSTUDIO, MDK Lite, MDK KickStart, Wearables</td>
<td>TrueSTUDIO, MDK Lite, MDK KickStart, Wearables</td>
<td>TrueSTUDIO, MDK Lite, MDK KickStart, Wearables</td>
<td>TrueSTUDIO, MDK Lite, MDK KickStart, Wearables</td>
<td>TrueSTUDIO, MDK Lite, MDK KickStart, Wearables</td>
</tr>
</tbody>
</table>

Austin Marathon – Freescale Survey

- 74% use wearables to train
- 88% of people surveyed said they rely on wearables for motivation similar to a coach
- 78% believe wearables give them a competitive edge
- 88% plan to use fitness wearables in the future

Wearable Market Forecast

- Fastest growing market over the next five years in both units and revenue
- CAGR 2013-17 > 50%
- 2017 Revenue = $50B
- 2017 unit forecast > 50 M

Wearable Market: Segmentation

<table>
<thead>
<tr>
<th>Category</th>
<th>Vertical</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Smart Watches</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Company</th>
<th>Full Feature OS</th>
<th>Function Specific OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Withings</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Samsung</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Fitbit</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Pebble</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Basis</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Markit</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Impulse</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Samsung Galaxy</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Samsung Gear</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Sony SmartWatch</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Samsung</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Garmin</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Kipros</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Coudos</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Alcatel Digital</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Samsung</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Motorola</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Casio</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Viva</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

17

External Use
Wearables is Not Just Smart Watches...

Wearable Reference Platform enabled by Freescale

Remote Patient Monitoring: Freescale Sensors Proposal

Wearable Market: Diverse Usage Models

Remote Patient Monitoring: Freescale Sensors Proposal

Enables Freescale Accelerometers, Gyrosopes, Sensing Platforms, Magnetic Sensors and Touch Sensors
- MMA9553L accelerometer/32 bit processor is the intelligent pedometer platform
- FXL09000 accelerometer/32 bit processor as a sensor hub and datalogger
- MAG3110 magnetometer and MMA9551 3 axis accelerometer combined in the FXOS0700, for orientation, motion, vibration, shock, fall, g-force, etc. are present
- MPL3115A digital pressure sensor for allometry
- MPY9121 for touch sensing
- FXAS21002 gyroscope provides the stability needed for a drift free readings, when taking accelerometer data, gyroscope bias.
Smart Watches Available NOW – SONY SWR10

Smart Watches Available in the future – SONY SWR30

Smart Watches Available in the future – SONY SWR50
Alessandro’s arm in the future

Wireless Charger

Wearebles in the future - http://youtu.be/-nVhBXuK-EI

WCT-SW1COILTX – EVALUATION BOARD

WCT-SW1COILTX Single-Coil Wireless Charger Block Diagram

Energy Harvesting
Energy Harvesting - Concept

Wireless sensor node

Harvester | Capacitor | PMIC | Capacitor Battery
--- | --- | --- | ---
Harvesting Energy | Storage, Convert the stable energy | Consuming Energy

Energy Harvesting - Application

Energy Source

- Light
- Vibration
- Thermal
- Radio Noise

Power Generation Device

- Power IC
- Solar Cell
- Piezoelectric
- Electromagnetic induction

MB89C821

MB89C821 (Boost)

Sensor

MCU Memory

Sensing & CPU

Power Generation part

Control & Wireless part

Energy Harvesting – Energy sources

- TDK
- IXYS
- YAMAHA
- Panasonic
- ADI
- TI
- STMicroelectronics

- Sensor Type Caveat Physical / Virtual
  - Accelerometer
    - Witth gravity
    - Without gravity
  - Linear Acceleration
    - With gravity
    - Without gravity
  - Magnetic Field
    - Uncalibrated
    - Calibrated
  - Gyroscope
    - Uncalibrated
    - Calibrated
  - Orientation
    - Rotation Matrix
    - Quaternion
    - Admittance, pitch, roll and rotation matrix
  - Ambient Temperature
  - Pressure
  - Humidity
  - Proximity
  - Light
  - Relative Humidity

Some Sensors are Physical, Some are “Virtual”

<table>
<thead>
<tr>
<th>Sensor Type</th>
<th>Covered</th>
<th>Physical / Virtual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accelerometer</td>
<td></td>
<td>Physical</td>
</tr>
<tr>
<td>Linear Acceleration</td>
<td></td>
<td>Virtual</td>
</tr>
<tr>
<td>Gravity</td>
<td></td>
<td>Virtual</td>
</tr>
<tr>
<td>Magnetic Field</td>
<td></td>
<td>Physical</td>
</tr>
<tr>
<td>Magneto Field</td>
<td></td>
<td>Virtual</td>
</tr>
<tr>
<td>Gyroscope</td>
<td></td>
<td>Physical</td>
</tr>
<tr>
<td>Orientation</td>
<td></td>
<td>Virtual</td>
</tr>
<tr>
<td>Ambient Temperature</td>
<td></td>
<td>Physical</td>
</tr>
<tr>
<td>Pressure</td>
<td></td>
<td>Physical</td>
</tr>
<tr>
<td>Proximity</td>
<td></td>
<td>Physical</td>
</tr>
<tr>
<td>Relative Humidity</td>
<td></td>
<td>Physical</td>
</tr>
</tbody>
</table>

Items in red are not supported by Freescale sensors.

Part 1: Motion Sensors Overview
Some Sensors are Physical, Some are “Virtual”

<table>
<thead>
<tr>
<th>Sensor / Type</th>
<th>Component</th>
<th>Physical / Virtual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rotation Vector</td>
<td>3-axis</td>
<td>Virtual</td>
</tr>
<tr>
<td>Gyro</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Geomagnetic Rotation Vector</td>
<td></td>
<td>VR</td>
</tr>
<tr>
<td>Chip/ChipMotion</td>
<td></td>
<td>VR</td>
</tr>
<tr>
<td>Stop Detector</td>
<td></td>
<td>VR</td>
</tr>
<tr>
<td>Stop Counter</td>
<td></td>
<td>VR</td>
</tr>
</tbody>
</table>

- The list above summarizes sensors & sensor fusion components that might be expected components for modern operating systems.
- All but the last 4 listed are supported by Android 4.3. “KitKat” offers support for the last four.
- Other OS’s continue to evolve in a similar fashion.
- The possible list of sensors and types of sensor fusion is virtually unlimited.

In this workshop…
- Because “Sensor Fusion” is an extremely broad topic, this course focuses on some specific examples:
  - Magnetic calibration
  - Electronic compass
  - Virtual gyro
  - Compute orientation
  - Compute linear acceleration sans gravity
- Sensors used include: Accelerometer + Magnetometer + Gyro
- For today’s session, we are ignoring: vibration analysis, gesture detection, contextual awareness, navigation / location, auto crash detection, auto stability control, etc.

Sensor Strengths & Weaknesses

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Strengths</th>
<th>Weaknesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accelerometer</td>
<td>• inexpensive • extremely low power • very linear • very low noise</td>
<td>• Measures the sum of gravity and acceleration. We need them separate.</td>
</tr>
<tr>
<td>Magnetometer</td>
<td>• the only sensor that can orient itself with regard to “North” • insensitive to linear acceleration</td>
<td>• Subject to magnetic interference • Not “spatially constant”</td>
</tr>
<tr>
<td>Gyro</td>
<td>• relatively independent of linear acceleration • Can be used to “gyro-compensate” the magnetometer</td>
<td>• Power hog • Long startup time • Zero rate offset drifts over time</td>
</tr>
<tr>
<td>Pressure Sensor</td>
<td>• the only stand-alone sensor that can give an indication of altitude</td>
<td>• Not well understood • A “relative” measurement • Subject to many interferences and environmental factors</td>
</tr>
</tbody>
</table>

An Accelerometer Measures Linear Acceleration plus Gravity

External Use

\[
\begin{align*}
\text{An accelerometer by itself is a “3 axis” system.} \\
\text{When any axis is vertical, we cannot detect rotation about that axis.} \\
\text{When horizontal, and accelerations at 1g in the direction of the arrow:} \\
X = 1g \\
Y = 0 \\
Z = 1g
\end{align*}
\]

Adding a gyroscope

This “6 axis” system is known as an Inertial Measurement Unit or “IMU”
This is a Right Hand System (RHR)

A 3-axis gyroscope measures angular velocity about each of the 3 axes.

What do we mean: Accelerometers measure linear acceleration plus gravity?

When horizontal, and at rest:
\[
\begin{align*}
X & = 0 \\
Y & = 0 \\
Z & = 1g
\end{align*}
\]

When horizontal, and accelerating at 1g in the direction of the arrow:
\[
\begin{align*}
X & = 1g \\
Y & = 0 \\
Z & = 1g
\end{align*}
\]
Adding a magnetometer

This “3 axis” system is known as a magnetic, angular rate & gravity (MARG) sensor. Add a processor and you have an attitude & heading reference system (AHRS).

A 3-axis magnetometer gives you the X/Y/Z components of the magnetic field.

As an aside...

In Grapevine Texas, during the week of FTF2014, almost 2/3 of the earth’s magnetic field is directed DOWN.

Adding a pressure sensor

This is a “10 axis” system.

Pressure can give you an estimate of altitude

Altitude = K1 X (1 - (P/P0)K2)
• K1 = 44330.77 meters
• K2 = 0.190263 (unitless)
• P0 = 101325 Pascals

Notice this is a log scale... (think in dB, ok???)

Typical Sensor Power in µW
Some observations

- Accelerometers are the most power efficient motion sensor you’ll find.
- They often include motion detection circuits – use those to power the system up/down for idle periods.
- Accelerometers are low power because they are usually “passive” devices. The proof mass moves only when the device is in motion.
- Gyros have continuously moving proof masses, requiring much higher currents to keep them in motion.
- TMR1-based magnetic sensors are arranged in a Wheatstone bridge formation – requiring DC biases.
- Another good sensor to “gate” others is an ambient light sensor.

Typical "Minimum" Sensor Complements / Application

<table>
<thead>
<tr>
<th>Application</th>
<th>Acc</th>
<th>Mag</th>
<th>Gyro</th>
<th>Pressure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pedometry, vibration analysis, tiltmeter</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>eCompass, pointing/remote control, augmented/virtual reality</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Virtual gyro</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gyro-compensated eCompass</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Activity monitors</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Motion capture</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>3D mapping &amp; localization</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Image stabilization, gesture recognition</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Movement

Any movement from point A to point B can be decomposed into a translation plus optional rotation.

We need at least 6 degrees of freedom (DOF) to describe a movement in 3 dimensions: \( \Delta X, \Delta Y, \Delta Z, \phi, \theta, \psi \).

Frames of Reference

- Most systems use a Cartesian frame of reference, but which one?

Aerospace uses NED

Windows 8 and Android use ENU

\[ \begin{align*}
\Delta X & = 10000 \\
\Delta Y & = 12000 \\
\Delta Z & = 14000 \\
\phi & = 16000 \\
\theta & = 2000 \\
\psi & = 6000
\end{align*} \]
There can be multiple, concurrent, frames of reference

Body or Device Reference Frame

North

North

Earth Frame

East

There are multiple representations for rotation

Options are:

- Euler Angles – intuitive (roll, pitch & yaw), but subject to gimbal lock
- Rotation Matrices – rotation as a matrix multiplication
- Axis / Angle – easy to understand, difficult to use
- Quaternions – similar to axis/angle, with a theoretical background that makes them useful

- Freescale sensor fusion libraries support all formats!

Euler Angle Rotation

Source: http://live.nxp.com/eng/3dAngularMotion.html

What is Sensor Fusion?

Sensor fusion encompasses a variety of techniques which:

- Trade off strengths and weaknesses of the various sensors to compute something more than can be calculated using the individual sensors;
- Improve the quality and noise level of computed results by taking advantage of:
  - Known data redundancies between sensors
  - Knowledge of system transfer functions, dynamic behavior and/or expected motion

Freescale Sensor Fusion Library

Full featured sensor fusion library including the award winning eCompass software

- Fully open source, eliminating proprietary constraints, increasing flexibility, and decreasing time-to-market

- Customer Application
  - Sensor Hub
  - Sensor Fusion
  - Quaternion

- Product Features
  - Functionality
    - 3-axis, 2-axis heading, 6-axis eCompass, 6-axis indirect Kalman filter, 3-axis relative rotation, and 6 axis indirect Kalman filter
    - Programmable sampling, fusion rates, and frame of reference
  - Included projects
    - Kineto X20, KL25Z, KL26Z, KL46Z, and X64F Freedom boards
    - Use of Freescale multi sensor boards
    - CodeWarrior and Kineto Design Studio
  - Additional commercial support and services available

Sensor Fusion Data Flow for Consumer Devices

Learn more at: www.freescale.com/sensorfusion
**Electronic Compass**

- Soft iron (in fixed spatial relationship to the sensor) distorts the measured field.
- Hard iron (permanent magnet) in fixed spatial relationship to the sensor with an offset.

Both are linear effects¹, and can be reversed – if you know what you are doing!

¹ Assuming there is no magnetic hysteresis present.

**Magnetic Calibration Variations**

\[ B_{m} = W(B_{o} - V) \]

Where:
- \( B_{m} \): Calibration magnetic vector
- \( B_{o} \): measured vector
- \( V \): Hard Iron Offset Vector

The 5-element calibration computes \( k_{x}, k_{y}, k_{z} \) and \( k_{h} \) hard iron offsets plus magnitudes of the geomagnetic vector \( M \) = identity matrix.

The 7-element calibration also computes \( k_{x}, k_{y}, k_{z} \) and \( k_{h} \). Diff diagonal components of \( W \) are:

\[ W_{x} = 1, \quad W_{y} = 1, \quad W_{z} = 1, \quad W_{h} = 1 \]

Everyone uses the same equation. The magic is in how you compute the coefficients.

**Freescale Magnetic Calibration Library**

- Now bundled into the sensor fusion library.
- 4 and 7 and now 10-element solvers are available in source form.
- As a virtual sensor in Freescale’s Intelligent Sensing Framework (ISF).
- Freescale’s eCompass software received the Electronic Products Magazine 2012 Product of the Year Award.

**Virtual Gyro**

If you calculate orientation from accel + mag, computing outputs for a virtual gyro is easy:

\[ \omega_{0} = -\Omega \Phi \theta \]

Angular rates = the time derivative of orientation

For rotation of fixed reference frame relative to body frame (equivalent to a gyro output), we have:

\[ \omega = \left[ \begin{array}{c} \omega_{x} \\ \omega_{y} \\ \omega_{z} \end{array} \right] = \Phi^{-1} \Omega \theta \]

where:

\[ \Phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & \sin \theta \\ 0 & -\sin \theta & \cos \theta \end{bmatrix} \]

**Implementing aTilt-Compensated eCompass with Magnetic Calibration**

- Once you have performed magnetic calibration, computing magnetic north is easy using cross products.
- The geomagnetic vector is produced by the sensor.

**Electronic Compass**

Once you have performed magnetic calibration, computing magnetic north is easy using cross products.

1. East\(_{mag} = B_{x} \times East\)
2. Normalize East = East\(_{mag} / |East\(_{mag}|\)
3. Normalize Up = A / |A|
4. Magnetic North = A \times East

**Virtual Gyro**

If you calculate orientation from accel + mag, computing outputs for a virtual gyro is easy:

\[ \omega = \left[ \begin{array}{c} \omega_{x} \\ \omega_{y} \\ \omega_{z} \end{array} \right] = \Phi^{-1} \Omega \theta \]

Angular rates = the time derivative of orientation

For rotation of fixed reference frame relative to body frame (equivalent to a gyro output), we have:

\[ \omega = \left[ \begin{array}{c} \omega_{x} \\ \omega_{y} \\ \omega_{z} \end{array} \right] = \Phi^{-1} \Omega \theta \]

where:

\[ \Phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & \sin \theta \\ 0 & -\sin \theta & \cos \theta \end{bmatrix} \]
Orientation can be thought of as a rotation from some standard reference (usually the global frame). For a set of sensors at rest, orientation can be considered to be the 3D rotation necessary to map magnetic north into calibrated magnetic field reading and gravity measured accelerometer reading.

\[
B = RM \begin{bmatrix} 0 \\ B_y \\ B_z \end{bmatrix} \quad A = RM \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix}
\]

- \(B\) = measured magnetic field after calibration
- \(B_0\) = magnetic north in the ENU frame of reference
- \(A\) = accelerometer reading (in gravities) at rest
- \(B_0\) = magnitude of the earth field
- \(RM\) = rotation matrix = orientation
- \(ENU\) = X=East, Y=North, Z=Up

Taking it up a notch

- The MagCal / eCompass example is nice because it can be explicitly calculated
- Other systems can be much more complex
- If we can model a system as a set of state variables, then we can use a Kalman filter to separate noise from desired system behavior
- A Kalman filter essentially does a linear regression between measured and expected system response.
- Results can be proved to be optimum in a least-squares sense.
Computing information is only half the puzzle.

You have to do something with it. Enter…

The Freescale Sensor Fusion Toolbox

• Provides visualization functions for the fusion library
• Allows you to experiment with different sensor/algorithmd choices
• Gives you access to raw sensor data
• Allows you to log sensor and fusion data for later use
• Works with demo and development versions of the Freescale Sensor Fusion Library

• Platforms
  • Android
  • Windows PC

Sensor output data is “fused” using Freescale-developed code running on Kinetis, and then “beamed” to a PC or Android device, where it drives the GUI.
The Freescale Sensor Fusion ToolBox Features by Platform

<table>
<thead>
<tr>
<th>Feature</th>
<th>Android</th>
<th>PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bluetooth wireless link</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>USB wireless link</td>
<td>On WiGig board only</td>
<td>✓</td>
</tr>
<tr>
<td>Support for sensors</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Device View</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Statistics View</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Data logging Capability</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Availability</td>
<td>Google Play</td>
<td>Freescale android</td>
</tr>
</tbody>
</table>

The PC is used for display only. All values are computed on the embedded board.

PC Version – Device View

1. Rotating 3D PCB display
2. Image align function
3. Navigation Tabs for:
   - Sensors Data Tab
   - Dynamics Tab
   - Kalman
   - Help
4. Packet information:
   - choice of PC comm port
   - packet activity indicator
   - # of packet errors
5. Roll/Pitch/Yaw & MagCal status
6. Choice of sensor set & algorithm
7. Sensor board run time and build parameters. Data logging on/off

This is the most intuitive way to confirm that your sensor fusion is working properly.

PC Version – Sensors Tab

1. Raw Accelerometer Values
2. Calibrated Magnetometer Values
3. Raw Gyroscope Values

The PC is used for display only. All values are computed on the embedded board.

PC Version – Dynamics Tab

1. Roll, pitch & compass heading
2. Current quaternion
3. Angular velocity
4. Linear Acceleration

The PC is used for display only. All values are computed on the embedded board.

PC Version – Magnetics Tab

You can use this display to view how the magnetic constellation evolves over time in response to changing magnetic environments.

PC Version – Kalman Tab

1. Error in orientation estimate (X,Y,Z)
2. Computed gyro offset
3. Error in gyro offset estimate (X,Y,Z)

Use this tab to view how well your sensor fusion “digests” changes in its environment.
Important Point

- The template programs contained in the Freescale Sensor Fusion Library for Kinetis MCUs assume that you are utilizing the FRDM-FXS-MULTI-B Bluetooth board.
- KL25Z, KL26Z and KL46Z projects can also be used via UART/USB wired interface by the simple expedient of removing jumper J7, which powers the Bluetooth module.
- This works because the same UART is drives the Bluetooth module and the OpenSDA UART interface.
- K20D50M and K64F use separate physical UARTS for Bluetooth and OpenSDA. You will need to reconfigure the Processor Expert UART/USB wired interface by the simple expedient of removing jumper J7, which powers the Bluetooth module.
- Library for Kinetis MCUs assume that you are utilizing the FRDM-FXS-MULTI-B Bluetooth board.
Stats Page

For mag / acccel / gyro and rotation, the “Statistics” Views displays:
• sensor description
• current sensor value
• min / mean / max values
• standard deviation
• noise / Hz

When used with the “local” sensor sources, this is a good way to gain insight into devices from the competition!

If you would like to try it...


Freescale Sensor Expansion Boards

Kinetis KL25Z and K20D50M compatible Freescale Sensor Expansion Boards

<table>
<thead>
<tr>
<th>Expansion Board</th>
<th>Part Number</th>
<th>Price</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRDM-FXS-MULTI-B +</td>
<td>FXAS21000</td>
<td>$10</td>
<td>Freescale Sensor Expansion board with only 2 sensors</td>
</tr>
<tr>
<td>FRDM-FXS-9AXIS*</td>
<td>FXAS21000</td>
<td>$125</td>
<td>Freescale Sensor Expansion board with only 2 sensors</td>
</tr>
<tr>
<td></td>
<td>FXAS21000</td>
<td>$30</td>
<td>Freescale Sensor Expansion board with only 2 sensors</td>
</tr>
</tbody>
</table>

Freedom Development Platform for Xtrinsic Sensors FRDM-FXS-MULTI-B

Freedom Xtrinsic: FRDM-FXS-development hardware

<table>
<thead>
<tr>
<th>FRDM-FXS-HC20</th>
<th>FRDM-FXS-HC20X</th>
<th>FRDM-FXS-HC20Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3 V Power Jumper</td>
<td>3.3 V Power Jumper</td>
<td>3.3 V Power Jumper</td>
</tr>
<tr>
<td>Bluetooth</td>
<td>Bluetooth</td>
<td>Bluetooth</td>
</tr>
<tr>
<td>NAD2012</td>
<td>NAD2012</td>
<td>NAD2012</td>
</tr>
<tr>
<td>FRDM6070CO</td>
<td>FRDM6070CO</td>
<td>FRDM6070CO</td>
</tr>
<tr>
<td>MPL3115A2</td>
<td>MPL3115A2</td>
<td>MPL3115A2</td>
</tr>
<tr>
<td>MPX5500AC</td>
<td>MPX5500AC</td>
<td>MPX5500AC</td>
</tr>
<tr>
<td>K400204000</td>
<td>K400204000</td>
<td>K400204000</td>
</tr>
</tbody>
</table>

Part 6: Freescale Sensor Fusion Library for Kinetis

Kinetis KL25Z and K20D50M compatible Freescale Sensor Expansion Boards

<table>
<thead>
<tr>
<th>Expansion Board</th>
<th>Part Number</th>
<th>Price</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRDM-FXS-MULTI-B +</td>
<td>FXAS21000</td>
<td>$10</td>
<td>Freescale Sensor Expansion board with only 2 sensors</td>
</tr>
<tr>
<td>FRDM-FXS-9AXIS*</td>
<td>FXAS21000</td>
<td>$125</td>
<td>Freescale Sensor Expansion board with only 2 sensors</td>
</tr>
<tr>
<td></td>
<td>FXAS21000</td>
<td>$30</td>
<td>Freescale Sensor Expansion board with only 2 sensors</td>
</tr>
</tbody>
</table>
PREÇO CIF NO DIA 02 / 12 / 14 (DÔLAR A R$ 2,5624)

<table>
<thead>
<tr>
<th>GS</th>
<th>Part Number</th>
<th>Fabricante</th>
<th>Preço Unitário (US$)</th>
<th>Quantidade</th>
<th>Preço de Entrega</th>
</tr>
</thead>
<tbody>
<tr>
<td>FG</td>
<td>FRDM-FXS-MULTI-B</td>
<td>Freescale / On Semiconductor</td>
<td>799,1128</td>
<td>2 pcs</td>
<td>Ent. USA entrega 2/3 semanas</td>
</tr>
<tr>
<td>FG</td>
<td>FRDM-FXS-MULTI-L</td>
<td>Freescale / On Semiconductor</td>
<td>815,6801</td>
<td>0 pcs</td>
<td>7 semanas</td>
</tr>
<tr>
<td>FG</td>
<td>FRDM-FXS-AX5S</td>
<td>Freescale / On Semiconductor</td>
<td>193,7875</td>
<td>2 pcs</td>
<td>Ent. USA entrega 2/3 semanas</td>
</tr>
<tr>
<td>FG</td>
<td>FRDM-K64Z</td>
<td>Freescale / On Semiconductor</td>
<td>32,2000</td>
<td>199 pcs</td>
<td>Ent. USA entrega 2/4 semanas</td>
</tr>
<tr>
<td>FG</td>
<td>KL16Z</td>
<td>Freescale / On Semiconductor</td>
<td>98,0000</td>
<td>50 horas</td>
<td>7 semanas</td>
</tr>
<tr>
<td>FG</td>
<td>KL46Z</td>
<td>Freescale / On Semiconductor</td>
<td>223,7220</td>
<td>13 pcs</td>
<td>Ent. USA entrega 2/3 semanas</td>
</tr>
</tbody>
</table>

Freescale Sensor Fusion Development Kits

- Optimized for the computation of orientation with respect to a global frame of reference as a function of sensor readings from:
  - accelerometer
  - and/or gyroscope
  - and/or magnetometer
- Along with orientation, also computes:
  - linear acceleration
  - magnetic interference and correction factors for same
  - magnetic inclination angle
  - gyroscope zero-rate offset
  - compass heading
  - virtual gyro from accelerometer / magnetometer

How to Engage with Sensor Fusion

- [http://www.freescale.com/sensorfusion](http://www.freescale.com/sensorfusion)
- Contains the latest sensor fusion information
- Downloadable SW and demos
- Blogs and app notes
- Sensor fusion development kits
  - Available November 2014
  - Combination of FRDM-MULTI-B and FRDM-K64F boards
- Part numbers
- FRDM-SFUSION-5 with 50 hours commercial support
- FRDM-SFUSION-5 with community support
- Factory contact: SFSW@Freescale.com
- Email alias includes sensor and MCU teams

Freescale Sensor Fusion Library for Kinetis MCUs

- Supplied in source form under license from Freescale
- Implemented as pure C-code sitting on top of device driver and MQX-lite implementations created via Processor Expert
- Shipped in the form of CodeWarrior projects compatible with the Freescale Sensor Fusion Toolbox
- Downloadable from [http://www.freescale.com/sensorfusion](http://www.freescale.com/sensorfusion)
- Community support available at [https://community.freescale.com/community/sensors/sensorfusion](https://community.freescale.com/community/sensors/sensorfusion)
- Contract support services offered by Freescale. Contact: sfus@freescale.com for details.

Ordering Details

<table>
<thead>
<tr>
<th>Component</th>
<th>Price</th>
<th>Location</th>
</tr>
</thead>
</table>

Prices are current as of 6 Sept, 2014. They may vary in the future.
Part 5: Play with fusion options

Experiment with each of the following options

1. Accelerometer
2. Gyroscope
3. Accelerometer + Magnetometer (Compass)
4. Accelerometer + Gyroscope
5. 9-Axis Accelerometer + Gyroscope + Magnetometer

Experiment with each tab function on the fusion toolbox

For this demo

You need

- Freescale Freedom boards shown
- USB cable
- Freescale Sensor Fusion Toolbox running on a Windows Laptop
- Freescale Sensor Fusion Library for Kinetis MCUs

Make sure the switch on the top sensor board is “on”.

If you have a MULTI-B board, remove jumper J7
Development Requirements

- You must have either Kinetis Design Studio 1.1.1 or CodeWarrior 10.6 and Processor Expert to build sensor fusion applications using the Freescale Sensor Fusion Toolbox.
- Processor Expert can be downloaded from http://www.freescale.com/pe.
- In order to experiment with the demo program, you will need an Android 3.0 or higher device running the Freescale Sensor Fusion Toolbox OR the PC-based variant of the toolbox. Details are available at http://www.freescale.com/sensortoolbox.
- Fusion libraries and example projects supplied by the Freescale Sensor Solutions Division.
- Try the included demos to see the functionality of the toolbox.

Easy to use...

- Pre-built templates are targeted at specific Freedom boards.
- User code easily added to a single `.c` file within any of the following functions:
  - void UserStartup(void); // runs once, the first time through the high frequency task
  - void UserHighFrequencyTaskInit(void); // runs once, the first time through the high frequency task
  - void UserMediumFrequencyTaskInit(void); // runs once, the first time through the medium frequency task
  - void UserLowFrequencyTaskInit(void); // runs once, the first time through the low frequency task
- Sensor and fusion values are simply read from predefined structures.

User Tasks Page 1 of 3

```c
void UserStartup(void) {
  // PUT YOUR CODE HERE
  return;
}
```

```
void UserHighFrequencyTaskRun(void) {
  // PUT YOUR CODE HERE
  return;
}
```

```
void UserLowFrequencyTaskRun(void) {
  // PUT YOUR CODE HERE
  return;
}
```

User Tasks Page 2 of 3

```c
void UserMediumFrequencyTaskInit(void) {
  // PUT YOUR CODE HERE
  return;
}
```

```
void UserMediumFrequencyTaskRun(void) {
  // PUT YOUR CODE HERE
  return;
}
```

```
void UserLowFrequencyTaskRun(void) {
  // PUT YOUR CODE HERE
  return;
}
```

User Tasks Page 3 of 3

```c
void UserHighFrequencyTaskRun(void) {
  // PUT YOUR CODE HERE
  return;
}
```

```
void UserLowFrequencyTaskRun(void) {
  // PUT YOUR CODE HERE
  return;
}
```

Access Fusion Inputs & Outputs Via a Standard Set of Global Data Structures

Input Global Data Structures defined in build.h

```
// Pointer Function Structure Name Structure Type
Accelerometer thisAccel Accelerometer
Magnetometer thisMag Magnetometer
Gyroscope thisGyro Gyroscope
```

Output Global Data Structures defined in tasks.h

```
// Pointer Function Structure Name Structure Type
Accelerometer thisAccel.Accel Accelerometer
Magnetometer thisMag.Accel Magnetometer
Gyroscope thisGyro.Gyro Gyroscope
```

Steps to use:
1. Import project into CodeWarrior
2. Add your code as shown above
3. Build
4. Download and run
Location of Variables Within the Global Structures

Example: Reading Euler Angles

Using 3-axis model:
- float roll = thisSV_9DOF_GBY_KALMAN.fLPR[3][3];
- float pitch = thisSV_9DOF_GBY_KALMAN.fLPR[3][3];
- float yaw = thisSV_9DOF_GBY_KALMAN.fLPR[3][3];

Using 6-axis acceleration + gyro model:
- float roll = thisSV_6DOF_GY_KALMAN.fLPR[3][3];
- float pitch = thisSV_6DOF_GY_KALMAN.fLPR[3][3];
- float yaw = thisSV_6DOF_GY_KALMAN.fLPR[3][3];

Using 6-axis calibration model:
- float roll = thisSV_6DOF_GB_BASIC.fLPR[3][3];
- float pitch = thisSV_6DOF_GB_BASIC.fLPR[3][3];
- float yaw = thisSV_6DOF_GB_BASIC.fLPR[3][3];

Using 9-axis Kalman filter model:
- float roll = thisSV_9DOF_GB_KALMAN.fLPR[3][3];
- float pitch = thisSV_9DOF_GB_KALMAN.fLPR[3][3];
- float yaw = thisSV_9DOF_GB_KALMAN.fLPR[3][3];

Using 3-axis Kalman filter model:
- float roll = thisSV_3DOF_GB_BASIC.fLPR[3][3];
- float pitch = thisSV_3DOF_GB_BASIC.fLPR[3][3];
- float yaw = thisSV_3DOF_GB_BASIC.fLPR[3][3];

Here is an Example of Grabbing Quaternion Values

- float quaternion[4];
- float q0 = quaternion[0];
- float q = quaternion[1];
- float q1 = quaternion[2];
- float q2 = quaternion[3];

The Development Kit provides:

- Access to raw fusion and magnetic calibration functions
- Control over sampling and fusion rates
- Ability to add custom Hardware Abstraction Layer (HAL)
- Access to MQX-Lite customization via Processor Expert

Product Development Kit Structure

Files in bold red are meant to be customized on a per-project basis.

3.2 Project Overview
Part 7: Explore the Sensor Fusion Library

Fusion Options Are Controlled Via build.h

External Use

Fusion Options Are Controlled Via build.h

External Use

For this lab

You need:

- Freescale Freedom boards shown
- USB cable
- Freescale Sensor Fusion Toolbox running on a Windows Laptop
- FSPK, KLS25Z project template

You will install updated software images on your board.

Make sure the KLS25 switch is “on”

Note: The same process described here works for any of the fusion library template projects. You can use any of KLS25Z, KL26Z, KL46Z, K64F05M and K64F Freedom boards.
IF your PC has the template pre-installed...

- SKIP to Step 8
- Otherwise, repeat Steps 1 through 7 on the following pages

Installation Step 1
a. Copy installer into your working directory
b. Double-click FreescaleSensorFusionInstaller.exe

Installation Step 2
Read the license terms, click "I Agree"

Installation Step 3
a. Review the system requirements.
b. Click "Next"

Installation Step 4
a. Select the destination folder (automatically defaults to the folder in which you placed the installer).
b. Click "Next"

Installation Step 5
a. Select your choice of kits (defaults to CodeWarrior Fusion Projects and documentation).
b. Click "Install"
Installation Step 6  
a. Click “Close” to complete installation

Installation Step 7  
a. Confirm presence of project template, tools and docs directory

Installation Step 8  
a. Expand the template folder down into the FSPK_KL25Z/Sources directory  
b. Confirm that the set of files shown below is present

Installation Step 9  
a. Start CodeWarrior 10.6  
b. Select c:/Temp (or whatever directory you used) as your workspace  
c. Click “OK”

Installation Step 10  
a. From CodeWarrior, Select File->Import->General->Existing Projects into Workspace  
b. Click “Next”

Installation Step 11  
a. Select the proper root directory  
b. Check the project to be imported  
c. Click Finish
Installation Step 12
a. Close the CodeWarrior “Welcome Screen” if present
b. Expand the project folder to view contents

Your project has been successfully installed.

Lab 2, Step 1
a. Double-click on ProcessorExpert.pe. This will bring up the components browser
b. Click on “Generate Processor Expert Code” icon to run Processor Expert

c. Expand CodeWarrior-External Use-150

Lab 2, Step 2
a. Select the project name
b. Click on the “Build” icon

c. Click “Run”

d. Click “Run Configurations

Lab 2, Step 3
a. Plug your board in if it is not plugged in
b. Run->Run Configurations

Status check
• You should see the green LED blinking steadily, with a red flash a couple of times per second
• You have just successfully reprogrammed your board with the same application we’ve already experimented with
• Open up the Freescale Sensor Fusion Toolbox on your PC and confirm that operation is unchanged
• Open Sources/drivers.c and review function CreateAndSendBluetoothPacketsViaUART(). This function pulls virtually all fusion results from fusion output structures for transmission back to the Sensor Fusion Toolbox.
• This completes Lab2.

Optional Lab 3, Step 1: Let’s modify a few things
In Sources/drivers.c
Lab 3, Step 2: Rebuild & experiment

What should be the effect of the changes on the prior page?

Hint: iChi is tilt angle in degrees

a) Rebuild the project
b) Download and experiment with changes via the “Dynamics” tab in the Freescale Sensor Fusion Toolbox running on your PC

Don’t forget to refer to the slides which specify available fusion outputs.

This concludes the 3rd lab.

Reminder: Global Data Structures

<table>
<thead>
<tr>
<th>Pointer Function</th>
<th>Structure Name</th>
<th>Structure Type</th>
<th>defined in include file</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accelerometer</td>
<td>XYZ_Accel</td>
<td>AccelSensor</td>
<td>prog_config.h</td>
</tr>
<tr>
<td>Magnetometer</td>
<td>XYZ_Mag</td>
<td>MagSensor</td>
<td></td>
</tr>
<tr>
<td>Gyroscope</td>
<td>XYZ_Gyro</td>
<td>GyroSensor</td>
<td></td>
</tr>
<tr>
<td>9-axis results</td>
<td>XYZ_9DOF_GY_KALMAN</td>
<td>9-axis filter</td>
<td></td>
</tr>
<tr>
<td>xCompass results</td>
<td>XYZ_9DOF_GB_KALMAN</td>
<td>9-axis filter</td>
<td></td>
</tr>
<tr>
<td>accel+gyro results</td>
<td>XYZ_9DOF_GY_KALMAN</td>
<td>9-axis filter</td>
<td></td>
</tr>
</tbody>
</table>

Reminder: Location of variables within the global structures

<table>
<thead>
<tr>
<th>Description</th>
<th>Data Type</th>
<th>Structure Name</th>
<th>Global Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>tilt angle in degrees</td>
<td>float</td>
<td>fLPChi</td>
<td>fChiPl</td>
</tr>
<tr>
<td>linear acceleration in the global</td>
<td>float</td>
<td>fLPDelta</td>
<td>GB</td>
</tr>
<tr>
<td>magnetic inclination angle in tilt angle in degrees</td>
<td>float</td>
<td>fLPChi</td>
<td>fChiPl</td>
</tr>
<tr>
<td>acceleration vector (global)</td>
<td>float</td>
<td>fRPl[3][3]</td>
<td>fRPl[3][3]</td>
</tr>
<tr>
<td>sensor fusion in the global</td>
<td>float</td>
<td>fLPaGlPl[3]</td>
<td>9-axis</td>
</tr>
<tr>
<td>yaw angle</td>
<td>float</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>pitch angle</td>
<td>float</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>roll angle</td>
<td>float</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>angular velocity in earth</td>
<td>float</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>angular acceleration in earth</td>
<td>float</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>angular velocity in earth</td>
<td>float</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>angular acceleration in earth</td>
<td>float</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

In summary

Freescale offers the lowest cost, most complete, sensor fusion solution available anywhere, with:

- Free when used with Freescale sensors (see license file for details)
- 3, 6 and 9-axis sensor fusion options
- Source code for all functions
- Working template programs
- Low cost hardware options
- Extensive documentation (data sheet, user manual and multiple app notes, training slides and videos)
- Free Windows and Android applications to visualize fusion results
- Freescale community support at https://community.freescale.com/community-sensors/sensorfusion
- Paid support available from Freescale’s Software Services team (dstw@freescale.com)
- For more details, please visit http://www.freescale.com/sensorfusion

Free Windows and Android applications to visualize fusion results

Extensive documentation (data sheet, user manual and multiple app notes, training slides and videos)

Free Windows and Android applications to visualize fusion results

Freescale offers the lowest cost, most complete, sensor fusion solution available anywhere, with:

- Free when used with Freescale sensors (see license file for details)
- 3, 6 and 9-axis sensor fusion options
- Source code for all functions
- Working template programs
- Low cost hardware options
- Extensive documentation (data sheet, user manual and multiple app notes, training slides and videos)
- Free Windows and Android applications to visualize fusion results
- Freescale community support at https://community.freescale.com/community-sensors/sensorfusion
- Paid support available from Freescale’s Software Services team (dstw@freescale.com)
- For more details, please visit http://www.freescale.com/sensorfusion
More Information on Freescale Sensor Solutions

- [http://www.freescale.com/freedom](http://www.freescale.com/freedom)
- [http://www.freescale.com/gyro](http://www.freescale.com/gyro)
- [http://www.freescale.com/sensors](http://www.freescale.com/sensors)
- [http://www.freescale.com/sensorstoolbox](http://www.freescale.com/sensorstoolbox)
- [http://twitter.com/Sensorfusion](http://twitter.com/Sensorfusion)
- Blogs: Smart Mobile Devices
- Android App available on Google Play
  - Freescale Sensor Fusion Toolbox
  - [http://www.freescale.com/sensorfusion](http://www.freescale.com/sensorfusion)

Additional Resources

- Orientation Representations: Part 1
- Orientation Representations: Part 2
- Hard and soft iron magnetic compensation explained
- Freescale E-Compass Software

Wrap-up

In this course, we have:
- Learned some motion sensor basics
- Learned what “orientation” is
- Reviewed a basic introduction to motion sensor fusion
- Learned about Freescale’s Freescale Sensor Fusion Library, and how we might use it to create our own custom functions
- Experimented with the Freescale Sensor Fusion Toolbox
- Learned where to look for more information

Thank you for your time and interest.
Use the right rotation representation at each stage of your calculation

<table>
<thead>
<tr>
<th>Storage</th>
<th>Requires 11 bytes of storage (in triple precision: 6 elements at 3 bytes each)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computation for an orthogonal rotation</td>
<td>Requires 8 bytes of storage (6 elements at 4 bytes each)</td>
</tr>
<tr>
<td>Vector rotation</td>
<td>Rotating a vector by a given angle requires multiplication and 6 additions + 6 operations</td>
</tr>
<tr>
<td>Derivatives</td>
<td>Rotating a vector via rotation matrix requires 15 operations (6 elements each requiring 3 multiplications and 2 additions)</td>
</tr>
<tr>
<td>Understanding</td>
<td>Usually not easy to understand, with the details from rotation matrices</td>
</tr>
</tbody>
</table>
| Summary | Easily understood by most engineers

From rotation matrix = \( RM = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \) and \( \theta = \frac{a \cdot b}{|a||b|} \)

A couple of really useful math identities

- If \( \mathbf{a} \) and \( \mathbf{b} \) are 3x1 vectors, then
  - The dot product \((\mathbf{a} \cdot \mathbf{b})\) is a scalar:
    \[ \mathbf{a} \cdot \mathbf{b} = a_1b_1 + a_2b_2 + a_3b_3 = |\mathbf{a}| |\mathbf{b}| \cos \theta \]
  - \( \theta \) is the angle between the two vectors \( \cos \theta = \frac{(\mathbf{a} \cdot \mathbf{b})}{|\mathbf{a}||\mathbf{b}|} \)

- The cross product \((\mathbf{a} \times \mathbf{b})\) is another vector:
  \[ \mathbf{a} \times \mathbf{b} = |\mathbf{a}||\mathbf{b}| \sin \theta \mathbf{n} \]
where \( \mathbf{n} \) is a unit vector perpendicular to the plane containing \( \mathbf{a} \) and \( \mathbf{b} \)

tasks.c

- Defines the following functions:
  - RDAccelData_Init (void)
  - RDAccelData_Run (void)
  - Fusion_Init (void)
  - Fusion_Run (void)
  - MagCal_Run (void)
  - ApplyHybrid

- Compile options for tasks.c are responsible for binding in various algorithms into the final application

Project Configuration

- build.h contains standard defines to control the build process:
  - THISCOORDINESTATION = NED | ANDROID | WIN8
  - Boolean controls ( uncomment #define to enable):
    - UART_LIB = (enable UART communication for power management)
    - COMPUTE_3DOF_G_BASIC = Enable 3-axis accelerometer tilt algorithm
    - COMPUTE_3DOF_G_IBASIC = Enable 3-axis accelerometer Kalman algorithm
    - COMPUTE_3DOF_G_KALMAN = Enable 3-axis Kalman algorithm

Other configuration file changes are best made by the Freescale software and services team

Project Configuration

- #define SENSORFS 200 \( \times 1000 \) frequency (Hz) of sensor sampling process
- #define OVERSAMPLE_RATIO 8 \( \times 100 = \text{SENSORS} / \text{OVERSAMPLE} \)

Events.c

- NMI interrupt handlers (not used)
- Low frequency counter restart
- UART control functions
  - UART_On-BlockReceived() is where the application command interpreter is located
  - This is example code only, not a formal part of the fusion library
drivers.c major functions

- FXOS8700_Init() initializes the FXOS87000CO combo sensor
- FXAS21000_Init() initializes the FXAS21000 gyro
- MMA8652_Init() initializes the MMA8652 accelerometer
- MAG3110_Init() initializes the MAG3110 magnetometer

fx AS21000_ReadData() initializes the FXAS21000 gyro
MMA8652_Init() initializes the MMA8652 accelerometer
MAG3110_Init() initializes the MAG3110 magnetometer

CreateAndSendBluetoothPacketsViaUART() sends data packets via Bluetooth

mqx_tasks.c

- Main_task() sets up periodic tasks then exits
- RdSensData_task() is the high frequency sample task
- Fusion_task() is the medium frequency fusion task
  - flash green LED
  - calls Fusion_Run()
- send new packet via Bluetooth via CreateAndSendBluetoothPacketsViaUART()
- set MagCal event as necessary
- MagCal_task()
  - flash red LED
  - run MagCal_Run(), which is part of the fusion library

main.c

- "C" main()
  - PE_low_level_init()