Workshop: Freescale **Sensor Fusion Library** for Kinetis MCUs

Lendo os movimentos da IoT

Alessandro Cunha | FAE

d e z . 2 0 1 4

External Use

Agenda

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

External Use | 1

External Use | 2

Sensor Portfolio

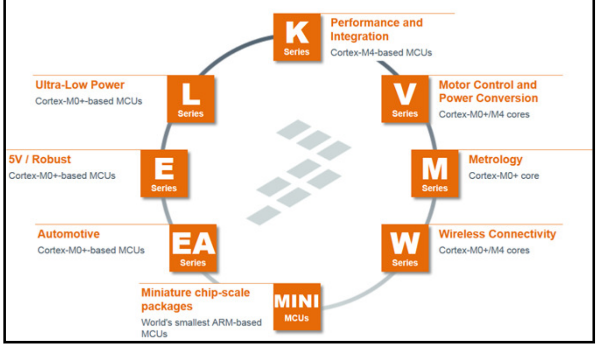
Pressure	**Automotive, industrial, medical** and **consumer** absolute and differential sensors *Flow, comfort management, HVAC, medical, engine control*
Accelerometer	**Consumer and industrial** low-g sensors and tilt sensors **Automotive** medium- and high-g crash sensors *Vehicle stability, airbag, vibration monitor, tilt alignment*
Magnetometer	**Consumer and industrial** magnetic field sensor and 3D compass *Orientation alignment, proximity detection, magnetic switch*
Gyroscope	**Consumer and industrial** angular rate sensors and 6/9-DOF IMU **Automotive** roll sensor and IMU *Stabilization, motion and gesture HMI, inertial navigation, gaming*
Sensing systems	**Consumer and industrial** MCU and sensor integrated platforms **Automotive** tire pressure monitoring system *Smart sensors, pedometer, anti-tamper, fault prognostication*

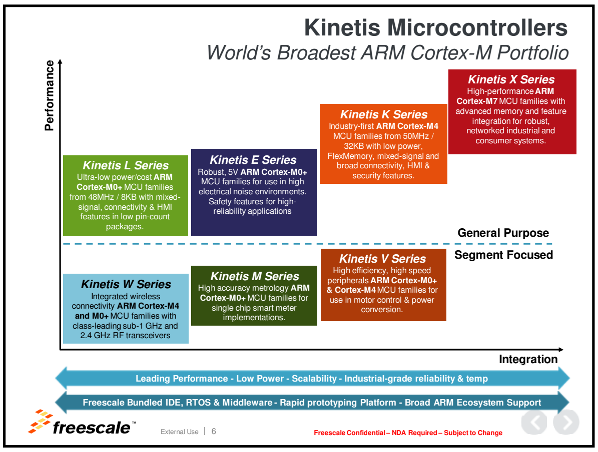
External Use | 3

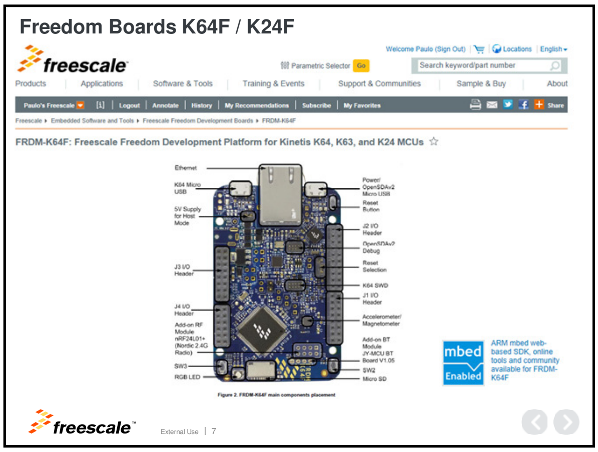
External Use | 4

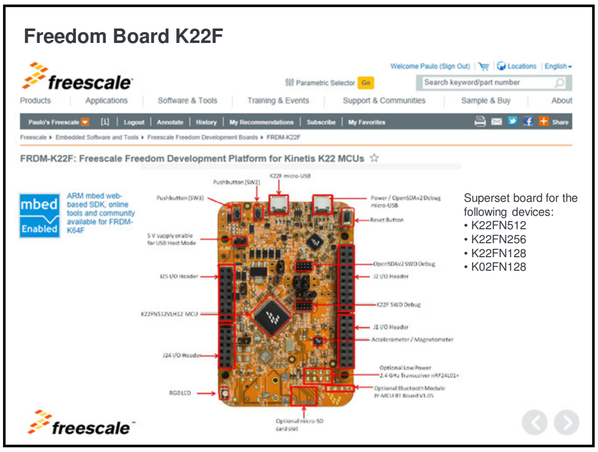
Kinetis Microcontrollers Family

Kinetis Microcontrollers (MCUs) consist of multiple hardware- and software-compatible ARM® Cortex®-M0+ and -M4-based MCU series with exceptional low-power performance, scalability and feature integration.

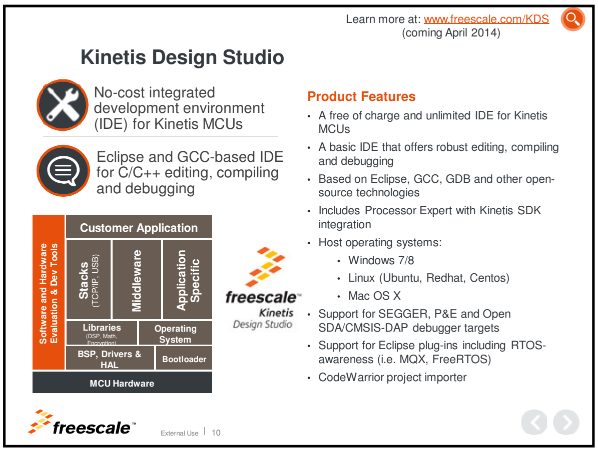


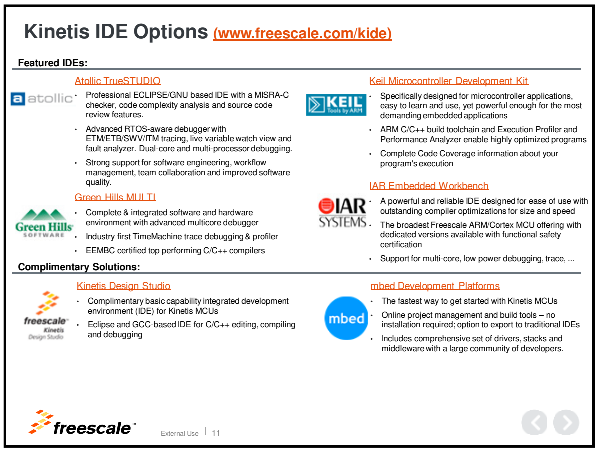

Kinetis Microcontrollers
World's Broadest ARM Cortex-M Portfolio
Performance
Kinetis X Series
High-performance ARM Cortex-M7 MCU families with advanced memory and feature integration for robust, networked industrial and consumer systems.
Kinetis K Series
Industry-first ARM Cortex-M4 MCU families from 50MHz / 32KB with low power, FlexMemory, mixed-signal and broad connectivity, HMI & security features.
Kinetis E Series
Robust, 5V ARM Cortex-M0+ MCU families for use in high electrical noise environments. Safety features for high-reliability applications
Kinetis L Series
Ultra-low power/cost ARM Cortex-M0+ MCU families from 48MHz / 8KB with mixed-signal, connectivity & HMI features in low pin-count packages.
General Purpose
Segment Focused
Kinetis V Series
High efficiency, high speed peripherals ARM Cortex-M0+ & Cortex-M4 MCU families for use in motor control & power conversion.
Kinetis M Series
High accuracy metrology ARM Cortex-M0+ MCU families for single chip smart meter implementations.
Kinetis W Series
Integrated wireless connectivity ARM Cortex-M4 and M0+ MCU families with class-leading sub-1 GHz and 2.4 GHz RF transceivers
Integration
Leading Performance - Low Power - Scalability - Industrial-grade reliability & temp
Freescale Bundled IDE, RTOS & Middleware - Rapid prototyping Platform - Broad ARM Ecosystem Support
freescale
External Use | 6
Freescale Confidential – NDA Required – Subject to Change


Freedom Boards K64F / K24F
freescale
FRDM-K64F: Freescale Freedom Development Platform for Kinetis K64, K63, and K24 MCUs
mbed Enabled
ARM mbed web-based SDK, online tools and community available for FRDM-K64F
freescale
External Use | 7


Freedom Board K22F
freescale
FRDM-K22F: Freescale Freedom Development Platform for Kinetis K22 MCUs
mbed Enabled
ARM mbed web-based SDK, online tools and community available for FRDM-K64F
Superset board for the following devices:
• K22FN512
• K22FN256
• K22FN128
• K02FN128
freescale

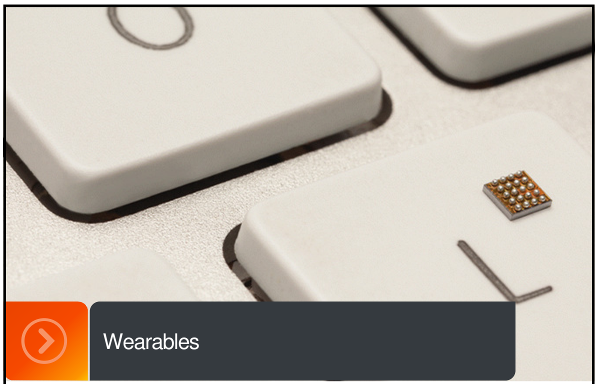

Kinetis Design Studio (KDS)
freescale
External Use | 9


Learn more at: www.freescale.com/KDS
(coming April 2014)
Kinetis Design Studio
No-cost integrated development environment (IDE) for Kinetis MCUs
Eclipse and GCC-based IDE for C/C++ editing, compiling and debugging
Product Features
• A free of charge and unlimited IDE for Kinetis MCUs
• A basic IDE that offers robust editing, compiling and debugging
• Based on Eclipse, GCC, GDB and other open-source technologies
• Includes Processor Expert with Kinetis SDK integration
• Host operating systems:
• Windows 7/8
• Linux (Ubuntu, Redhat, Centos)
• Mac OS X
• Support for SEGGER, P&E and Open SDA/CMSIS-DAP debugger targets
• Support for Eclipse plug-ins including RTOS-awareness (i.e. MQX, FreeRTOS)
• CodeWarrior project importer
Customer Application
Software and Hardware Evaluation & Dev Tools
Stacks (TCP/IP, USB)
Middleware
Application Specific
Libraries
Operating System
BSP, Drivers & HAL
Bootloader
MCU Hardware
freescale Kinetis Design Studio
freescale
External Use | 10


Kinetis IDE Options (www.freescale.com/kide)
Featured IDEs:
Atollic TrueSTUDIO
• Professional ECLIPSE/GNU based IDE with a MISRA-C checker, code complexity analysis and source code review features.
• Advanced RTOS-aware debugger with ETM/ETB/SWV/ITM tracing, live variable watch view and fault analyzer. Dual-core and multi-processor debugging.
• Strong support for software engineering, workflow management, team collaboration and improved software quality.
Green Hills MULTI
• Complete & integrated software and hardware environment with advanced multicore debugger
• Industry first TimeMachine trace debugging & profiler
• EEMBC certified top performing C/C++ compilers
Keil Microcontroller Development Kit
• Specifically designed for microcontroller applications, easy to learn and use, yet powerful enough for the most demanding embedded applications
• ARM C/C++ build toolchain and Execution Profiler and Performance Analyzer enable highly optimized programs
• Complete Code Coverage information about your program's execution
IAR Embedded Workbench
• A powerful and reliable IDE designed for ease of use with outstanding compiler optimizations for size and speed
• The broadest Freescale ARM/Cortex MCU offering with dedicated versions available with functional safety certification
• Support for multi-core, low power debugging, trace, ...
Complimentary Solutions:
Kinetis Design Studio
• Complimentary basic capability integrated development environment (IDE) for Kinetis MCUs
• Eclipse and GCC-based IDE for C/C++ editing, compiling and debugging
mbed Development Platforms
• The fastest way to get started with Kinetis MCUs
• Online project management and build tools – no installation required; option to export to traditional IDEs
• Includes comprehensive set of drivers, stacks and middleware with a large community of developers.
freescale
External Use | 11

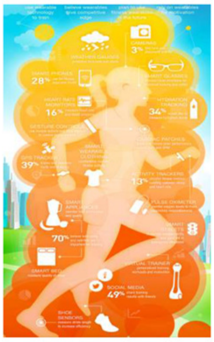
Kinetis IDE Comparison

	Atollic TrueStudio Pro	Green Hills MULTI	IAR Embedded Workbench for ARM (EWARM)	Keil PRO Edition Microcontroller Development Kit (MDK)	Kinetis Design Studio
Free version / Limitations	TrueSTUDIO Lite: 32KB 8KB for Cortex-M0(+)	Evaluation: 30 days	Evaluation: 30-days KickStart Edition: 32KB	MDK Lite: 32KB	Unlimited
Processor Expert support	Yes	Yes	Yes	Yes	Yes
IDE Framework	Improved/simplified Eclipse	Proprietary	Proprietary/Eclipse	Proprietary	Eclipse
Debugger	GDB + proprietary extensions	Multi	IAR C-SPY	uVison	GDB
Compiler	Atollic GNU gcc v4.7.3	Multi	IAR icc/c++	armcc	GNU gcc 4.8
Standard Libraries	newlib v1.19 newlib-nano v1.0 libstdc++ v6.0.17	Multi	IAR DLIB/CMSIS	ARM MicroLib ARM Standard	newlib 1.19 newlib-nano 1.0
Run Control Interfaces	P&E, SEGGER, CMSIS-DAP (coming soon), gdbserver compatible probes	GHS Probe, GHS SuperTrace Probe, OpenOCD, CMSIS-DAP (coming soon)	I-jet, P&E, SEGGER, OpenOCD, CMSIS-DAP	ULINK, ULINKpro, CMSIS-DAP, P&E, SEGGER	P&E, SEGGER, OpenOCD/CMSIS-DAP
Trace/Profiling Support	Yes	Yes	Yes	Yes	No
Kinetis SDK Support	1.0 GA (Summer 2014)	-	1.0 Beta (April 2014)	1.0 GA (Summer 2014)	1.0 GA (Summer 2014)
Freescale MQX Kernel / Task Awareness	Yes	-	Yes	Yes	Coming Soon
Other RTOS Support Includes	FreeRTOS, uC/OS	uvelOSity	FreeRTOS, uCos	FreeRTOS, uCOS, Keil RTX	FreeRTOS, uCos



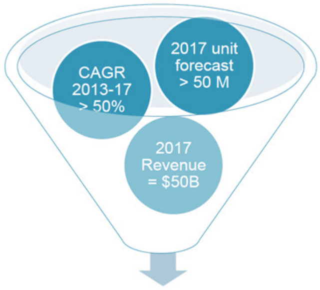
Wearables

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

Sources: IHS Research, ABI Research, Credit Suisse Equity Research, Berg Insight, Juniper

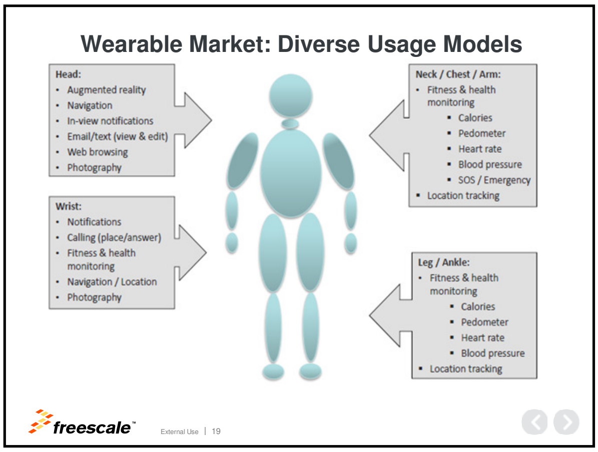
Smart Watches Available NOW

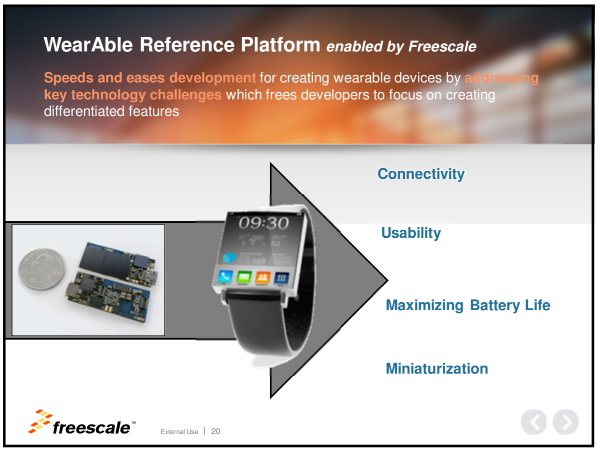
Full Feature OS | Function Specific OS

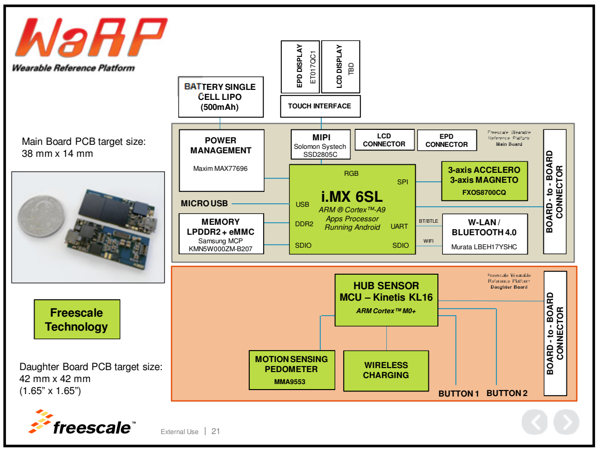
Wearable Market: Segmentation

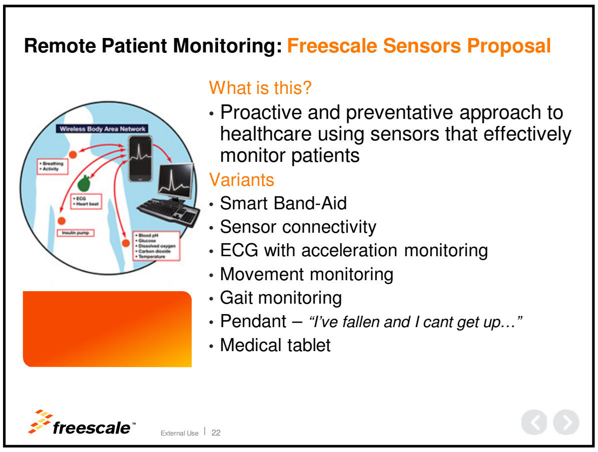
Vertical	Categories
Fitness & Wellness	**Sports & Heart Rate Monitors** **Pedometers, Activity Monitors** Smart Sport Glasses Smart Clothing Sleep Monitors Emotional Measurements
Healthcare & Medical	CGM (Continuous Glucose Monitoring) ECG Monitoring Pulse Oximetry Blood Pressure Monitors Drug Delivery (Insulin Pumps) **Wearable Patches** (ECG, HRM, SpO2)
Infotainment	**Smart Watches** Augmented Reality Headsets **Smart Glasses** Wearable Imaging Devices
Industrial & Military	Hand-worn Terminals Augmented Reality Headsets Smart Clothing

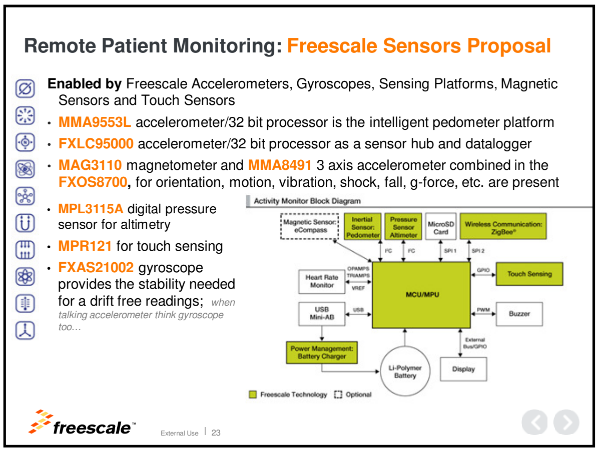
12/19/2014 11:25:38 AM



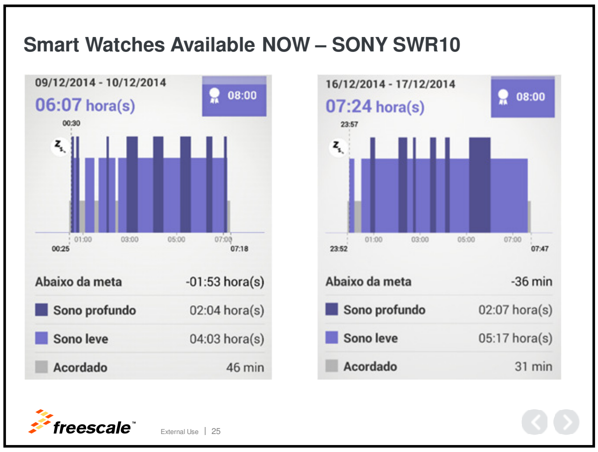


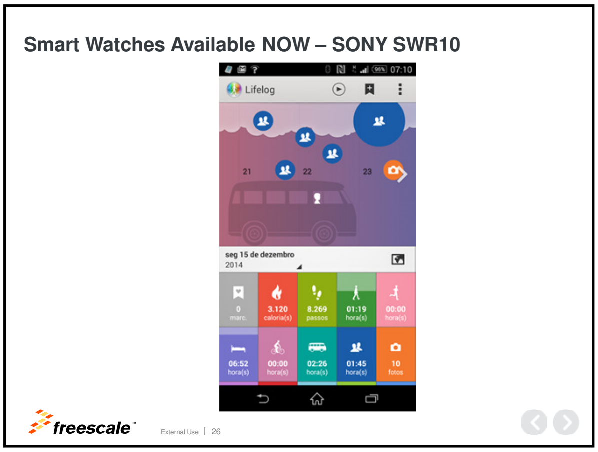







Smart Watches Available NOW – SONY SWR10
SONY
freescale
External Use | 24


Smart Watches Available NOW – SONY SWR10
09/12/2014 - 10/12/2014
06:07 hora(s)
08:00
Abaixo da meta -01:53 hora(s)
Sono profundo 02:04 hora(s)
Sono leve 04:03 hora(s)
Acordado 46 min
16/12/2014 - 17/12/2014
07:24 hora(s)
08:00
Abaixo da meta -36 min
Sono profundo 02:07 hora(s)
Sono leve 05:17 hora(s)
Acordado 31 min
freescale
External Use | 25

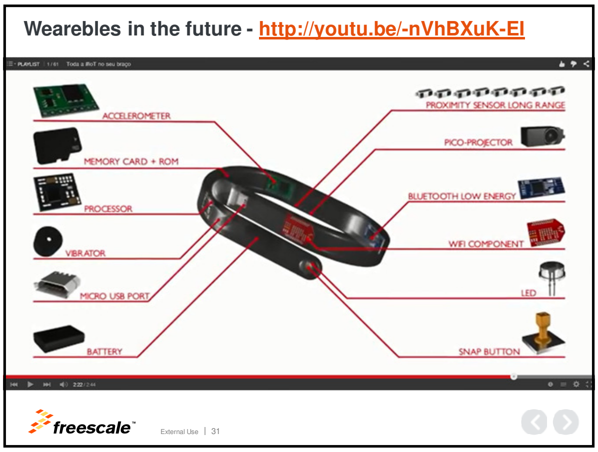

Smart Watches Available NOW – SONY SWR10
Lifelog
seg 15 de dezembro 2014
3.120 calorias
8.269 passos
freescale
External Use | 26


Smart Watches Available in the future – SONY SWR30
10
35
freescale
External Use | 27

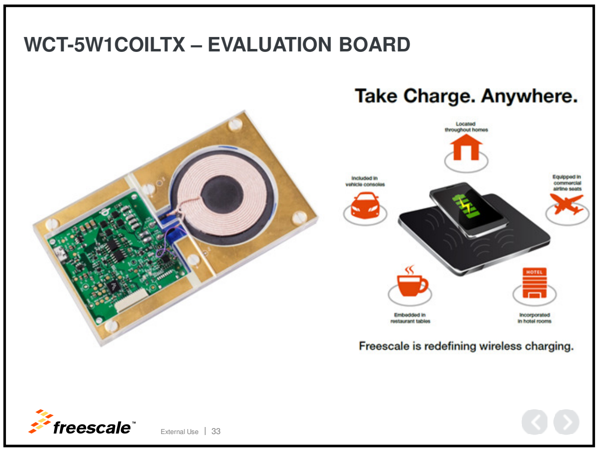

Smart Watches Available in the future – SONY SWR50
10
35
freescale
External Use | 28

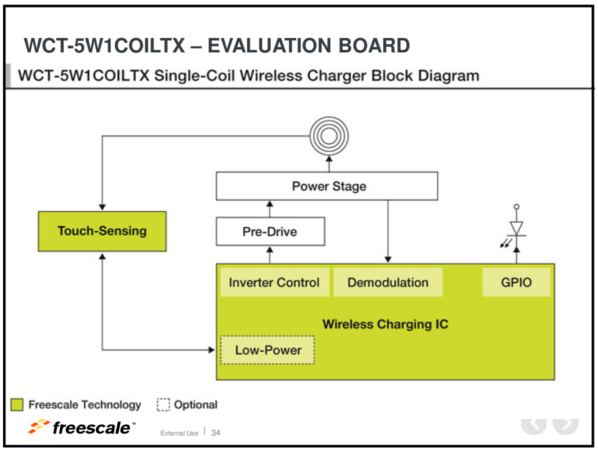

Smart Watches Available in the future – SONY SWR50
SONY
Smartwatch 3 SWR50
maxabout.com
1.6 Inch
Transflective Display
320 x 320 pixels
IP68 rated Water Protected
Voice, touch
Gesture input
Microphone
On/off/wake up key
420 mA Battery
Normal Use Upto 2 days
1.2 GHz Quad
ARM A7 processor
SmartWatch 3 is optimised
for devices running on
Android 4.3 and later
Accelerometer
Compass Sensor
Ambient light sensors
Gyro Sensor
GPS Sensor
45 Grams
Weight
Black
Yellow
v4.0 Bluetooth
NFC, Micro USB
512 MB RAM
4 GB eMMC
Price
$249.99
₹15400
freescale
External Use | 29


Alessandro's arm in the future
10
35
Steps goal reached!

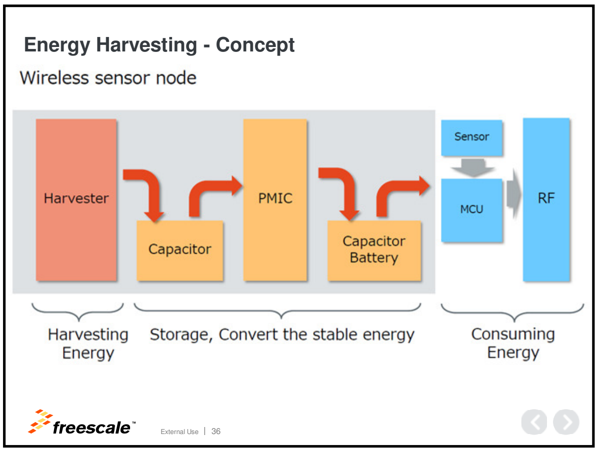

Wearebles in the future - http://youtu.be/-nVhBXuK-EI
ACCELEROMETER
MEMORY CARD + ROM
PROCESSOR
VIBRATOR
MICRO USB PORT
BATTERY
PROXIMITY SENSOR LONG RANGE
PICO-PROJECTOR
BLUETOOTH LOW ENERGY
WIFI COMPONENT
LED
SNAP BUTTON
freescale
External Use | 31

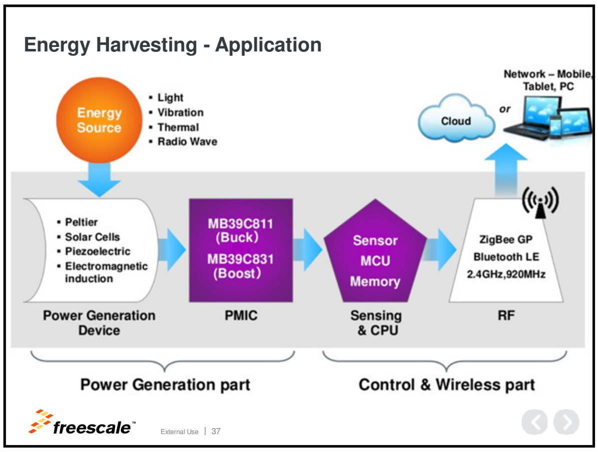

Wireless Charger
freescale
External Use | 32

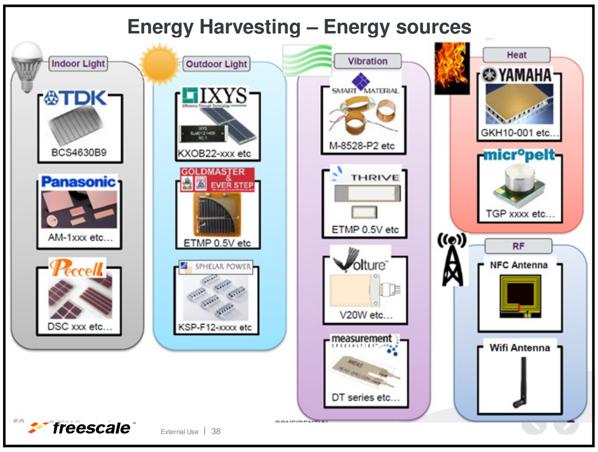

WCT-5W1COILTX – EVALUATION BOARD
Take Charge. Anywhere.
Freescale is redefining wireless charging.
freescale
External Use | 33

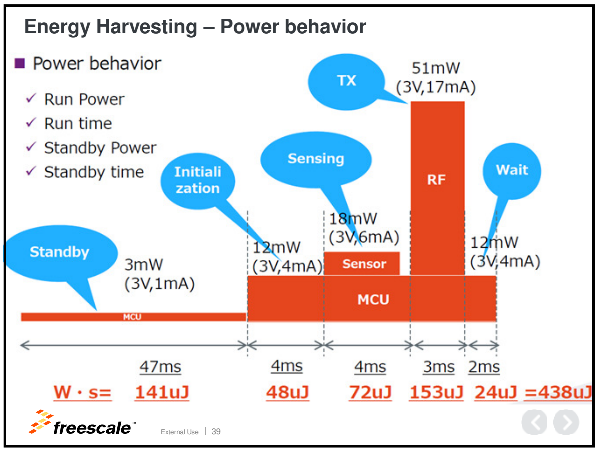

WCT-5W1COILTX – EVALUATION BOARD
WCT-5W1COILTX Single-Coil Wireless Charger Block Diagram
Power Stage
Touch-Sensing
Pre-Drive
Inverter Control
Demodulation
GPIO
Wireless Charging IC
Low-Power
Freescale Technology
Optional
freescale
External Use | 34


Energy Harvesting
freescale
External Use | 35









Some Sensors are Physical, Some are "Virtual"

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

Items in red are not supported by Freescale sensors.

freescale External Use | 41

Some Sensors are Physical, Some are "Virtual"

Sensor Type	Caveat	Physical / Virtual
Rotation Vector	9-axis	Virtual
Game Rotation Vector	Accel/gyro only	Virtual
Geomagnetic Rotation Vector	*Accel/mag only*	*Virtual*
Significant Motion		*Virtual*
Step Detector		*Virtual*
Step Counter		*Virtual*

- 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.

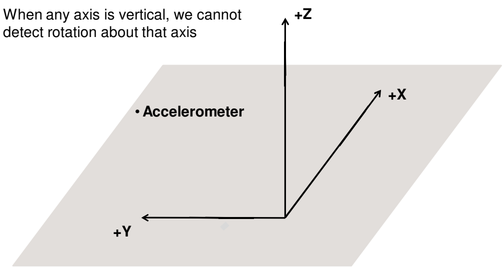
Sensor Strengths & Weaknesses

Sensor	Strengths	Weaknesses
Accelerometer	• Inexpensive • Extremely low power • Very linear • Very low noise	• Measures the sum of gravity and acceleration. We need them separate.
Magnetometer	• The only sensor that can orient itself with regard to "North" • Insensitive to linear acceleration	• Subject to magnetic interference • Not "spatially constant"
Gyro	• Relatively independent of linear acceleration • Can be used to "gyro-compensate" the magnetometer	• Power hog • Long startup time • Zero rate offset drifts over time
Pressure Sensor	• The only stand-alone sensor that can give an indication of altitude	• Not well understood • A "relative" measurement • Subject to many interferences and environmental factors

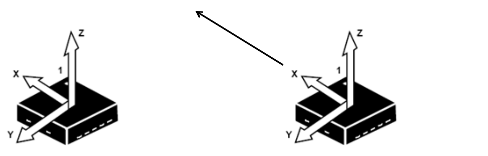
An Accelerometer Measures Linear Acceleration plus Gravity

An accelerometer by itself is a "3 axis" system

When any axis is vertical, we cannot detect rotation about that axis



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

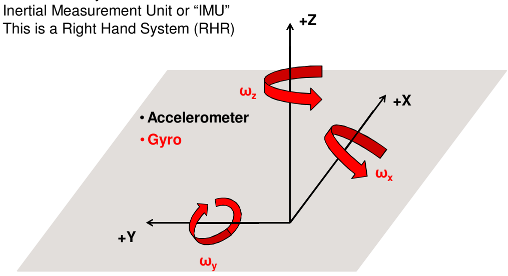


When horizontal, and at rest:
X = 0
Y = 0
Z = 1g

When horizontal, and accelerating at 1g in the direction of the arrow:
X = 1g
Y = 0
Z = 1g

Adding a gyroscope

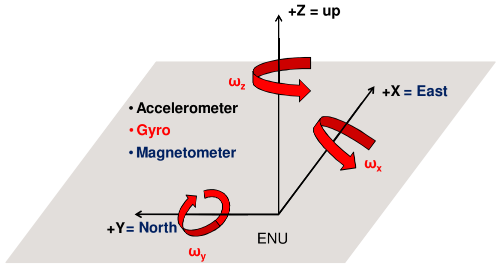
This "6 axis" system is known as an Inertial Measurement Unit or "IMU"
This is a Right Hand System (RHR)



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

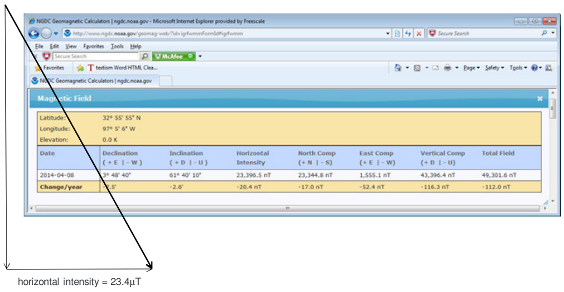
Adding a magnetometer

This "9 axis" system is known as a magnetic, angular rate & gravity (MARG) sensor
Add a processor and you have an attitude & heading reference system (AHRS)



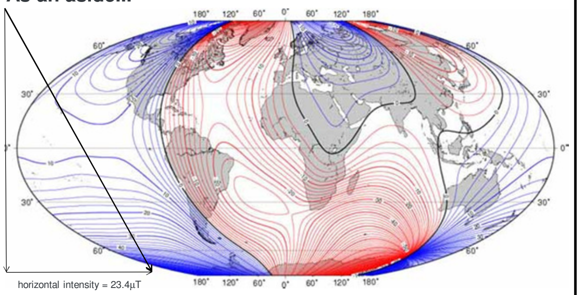
A 3-axis magnetometer gives you the X/Y/Z components of the magnetic field.

As an aside...



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

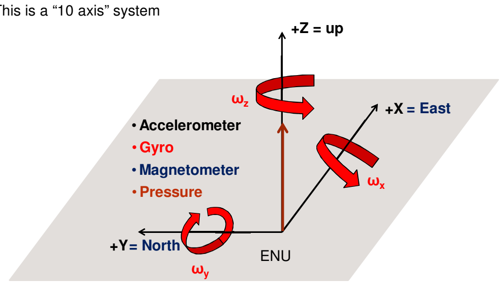
As an aside...



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

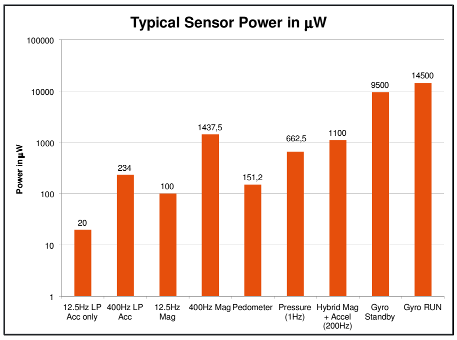


Pressure is a scalar (versus vector) quantity

Pressure can give you an estimate of altitude

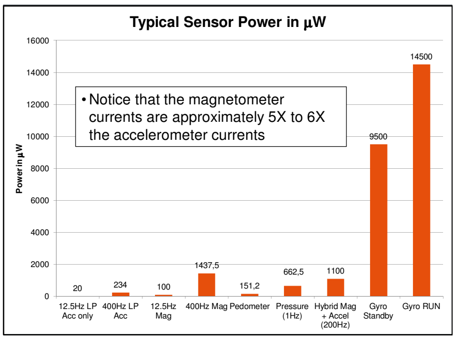
Altitude = K1 X (1 - $(P/P0)^{K2}$)
- K1 = 44330.77 meters
- K2 = 0.190263 (unitless)
- P0 = 101325 Pascals

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



This chart was created 2013, you can expect numbers to decrease over time.

Gyro Ready to active = 2/ODR + 10ms (now, think linear)



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

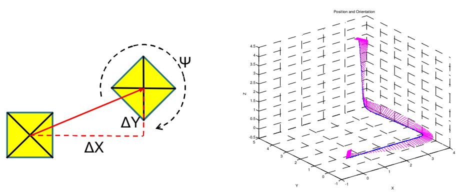
[1] TMR = Tunneling MagnetoResistive

Typical "Minimum" Sensor Complements / Application

Application	Acc	Mag	Gyro	Pressure
Portrait/landscape, tap detect, fall detection	X			
Pedometry, vibration analysis, tiltmeter	X			
eCompass, pointing/remote control, augmented/virtual reality	X	X		
Virtual gyro	X	X		
Gyro-compensated eCompass	X	X	X	
Activity monitors	X	X		
	X		X	
Motion capture	X	X	X	
3D mapping & localization	X	X	X	X
Image stabilization, gesture recognition	X		X	

Movement

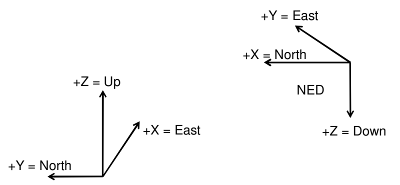
Any movement from point A to point B can be decomposed into a <u>translation</u> plus optional <u>rotation</u>



We need <u>at least</u> 6 degrees of freedom (DOF) to describe a movement in 3 dimensions: ΔX, ΔY, ΔZ, Φ, θ, Ψ

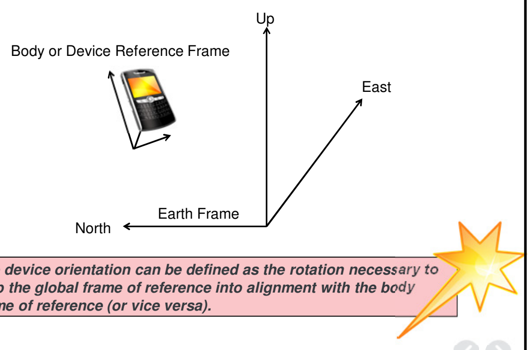
Frames of Reference

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



Aeronautics uses NED
Windows 8 and Android use ENU

There can be multiple, concurrent, frames of reference

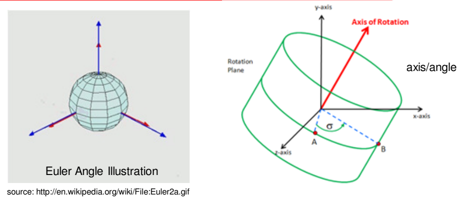


The device orientation can be defined as the rotation necessary to map the global frame of reference into alignment with the body frame of reference (or vice versa).

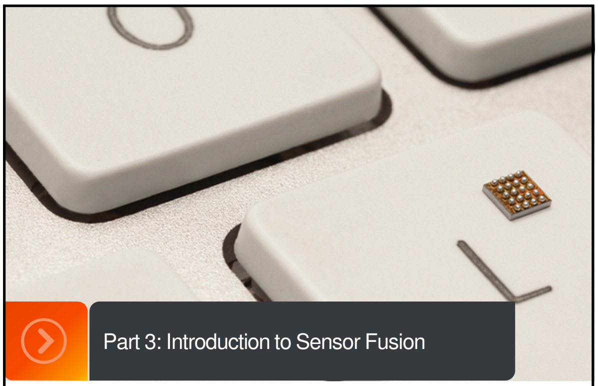
There are multiple representations for rotation

Options are:

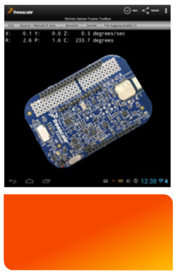
- **Euler Angles** – intuitive (roll, pitch & yaw), but subject to gimbal lock
- **Rotation Matrices** – rotation as a matrix multiplication
- **Axis / Angle** – easy to understand, difficult to use
- **Quaternions** – similar to axis/angle, with a theoretical background that makes them useful
- **Freescale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



source: http://en.wikipedia.org/wiki/File:Euler2a.gif



What is Sensor Fusion?



Sensor fusion encompasses a variety of techniques which:

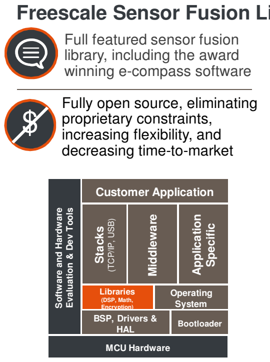
- Trade off strengths and weaknesses of the various sensors to compute something more than can be calculated using the individual sensors;
- Improve the quality and noise level of computed results by taking advantage of:
  - Known data redundancies between sensors
  - Knowledge of system transfer functions,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

Learn more at: www.freescale.com/sensorfusion

Freescale Sensor Fusion Library

Full featured sensor fusion library, including the award winning e-compass software

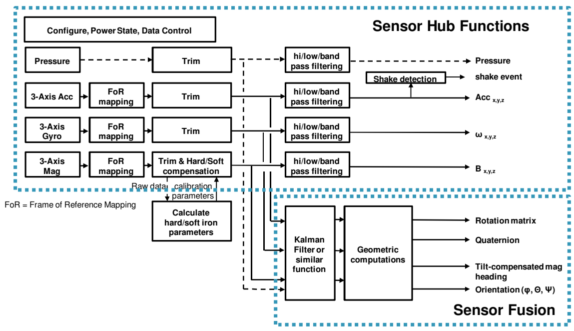
Fully open source, eliminating proprietary constraints, increasing flexibility, and decreasing time-to-market



Product Features

- Functionality
  - 3-axis, 2-axis heading, 6-axis eCompass,6-axis indirect Kalman filter, 3-axis relative rotation, and 9-axis indirect Kalman filter
  - Programmable sampling, fusion rates, and frame of reference,
- Included projects
  - Kinetis K20, KL25Z, KL26Z, KL46Z, and K64F Freedom boards
  - Use of Freescale Multi sensor boards
  - CodeWarrior and Kinetis Design Studio
- Additional commercial support and services available

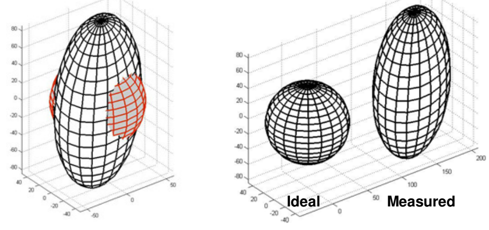
Sensor Fusion Data Flow for Consumer Devices



Magnetic Calibration

Soft Iron *in fixed spatial relationship to the sensor* distorts the measured field. The sphere is distorted into an ellipsoid.

Hard Iron (permanent magnet) *in fixed spatial relationship to the sensor* adds an offset.



Both are linear effects[1], and can be reversed – if you know what you are doing!

[1] Assuming there is no magnetic hysteresis present

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.

Magnetic Calibration Variations

$$B_c = W^{-1}(B_p - V) \Rightarrow \begin{bmatrix} B_{cx} \\ B_{cy} \\ B_{cz} \end{bmatrix} = \begin{bmatrix} s_1 & s_2 & s_3 \\ s_2 & s_4 & s_5 \\ s_3 & s_5 & s_6 \end{bmatrix} \begin{bmatrix} B_{px} - V_x \\ B_{py} - V_y \\ B_{pz} - V_z \end{bmatrix}$$

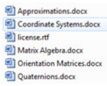
where:

- B_c Calibrated magnetic vector
- W^{-1} Inverse Soft Iron Matrix
- B_p Physical magnetic measurement
- V Hard Iron Offset Vector

The 4-element calibration computes V_x, V_y and V_z hard iron offsets plus magnitude of the geomagnetic vector . W^{-1} = identity matrix

The 7-element calibration also computes s_1, s_4 and s_6. Off diagonal components of W^{-1} are 0.

$$W^{-1} = \begin{bmatrix} s_1 & 0 & 0 \\ 0 & s_4 & 0 \\ 0 & 0 & s_6 \end{bmatrix}$$



The 10-element calibration computes all elements of W^{-1}, including s_2, s_3, and s_5

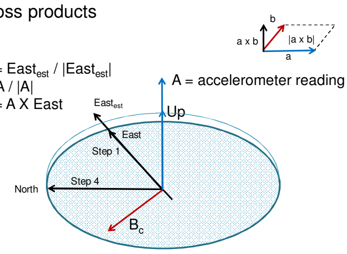
$$W^{-1} = \begin{bmatrix} s_1 & s_2 & s_3 \\ s_2 & s_4 & s_5 \\ s_3 & s_5 & s_6 \end{bmatrix}$$

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

Electronic Compass

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

Step 1: $East_{est} = B_c \times A$
Step 2: Normalize $East = East_{est} / |East_{est}|$
Step 3: Normalize $Up = A / |A|$
Step 4: Magnetic North = A X East



See getRotationMatrix function at:
http://developer.android.com/reference/android/hardware/SensorManager.html

eCompass – Virtual Gyro

Freescale Semiconductor
User's Guide

Document Number: MAGCALSWUG
Rev. 0, 02/2012

Implementing aTilt-Compensated eCompass with Magnetic Calibration

Software User's Guide

by: Mark Pedley
Freescale Semiconductor, Tempe, AZ

Virtual Gyro

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

angular rates = the time derivative of orientation

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

$$\text{Small signal rotation matrix} = R = R_\Phi R_\theta R_\Psi = \begin{bmatrix} 1 & -\Psi & \theta \\ \Psi & 1 & -\Phi \\ -\theta & \Phi & 1 \end{bmatrix}$$

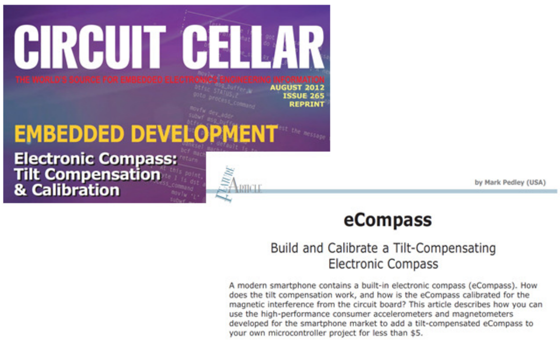
$$dR/dT = d(R_\Phi R_\theta R_\Psi)/dT = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} = (1/\Delta t)(R_{t+1} R_t^T - I_{3x3}) = \begin{bmatrix} 0 & \Omega_{1,2} & \Omega_{1,3} \\ \Omega_{2,1} & 0 & \Omega_{2,3} \\ \Omega_{3,1} & \Omega_{3,2} & 0 \end{bmatrix}$$

$$\omega_x = (2\Delta t)^{-1} (\Omega_{3,2} - \Omega_{2,3})$$
$$\omega_y = (2\Delta t)^{-1} (\Omega_{1,3} - \Omega_{3,1})$$
$$\omega_z = (2\Delta t)^{-1} (\Omega_{2,1} - \Omega_{1,2})$$

This derivation utilizes small angle approximations. See https://community.freescale.com/community/the-embedded-beat/blog/2013/03/12/building-a-virtual-gyro for derivation details.

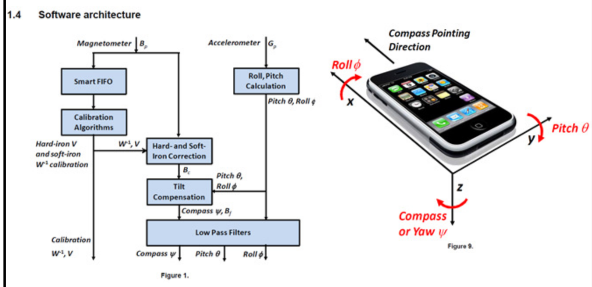
12/19/2014 11:25:38 AM

eCompass – Virtual Gyro



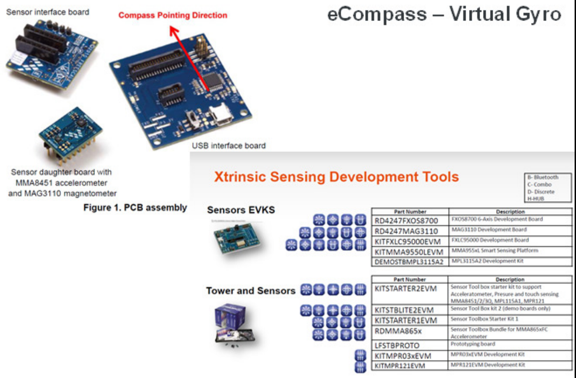
freescale External Use | 72

eCompass – Virtual Gyro



freescale External Use | 73

eCompass – Virtual Gyro



freescale External Use | 74

eCompass – Virtual Gyro

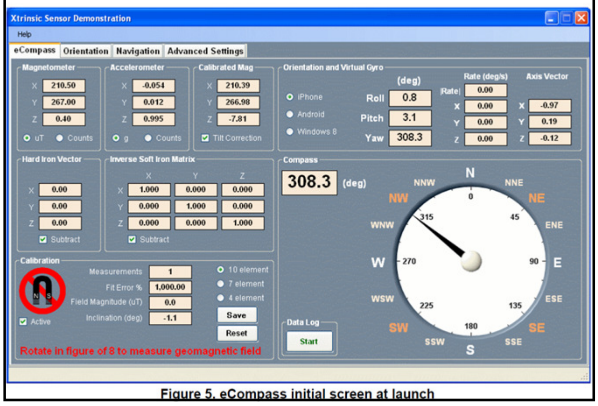


Figure 5. eCompass initial screen at launch

Orientation

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 to measured accelerometer reading.

$$B = RM \begin{bmatrix} 0 \\ B_N \\ B_Z \end{bmatrix} \qquad A = RM \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

gravity in the ENU[1] frame of reference

magnetic north in the ENU frame of reference. B_N is the horizontal component of the earth field, B_Z is the vertical.

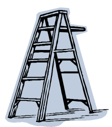
A = accelerometer reading (in gravities) at rest
B = measured magnetic field after calibration
|B| = magnitude of the earth field
RM = rotation matrix = orientation
ENU = X=East, Y=North, Z=Up

[1] Use $[0, 0, 1]^T$ Windows 8. Use $[0, 0, -1]^T$ for Android.

freescale External Use | 76

Taking it up a notch

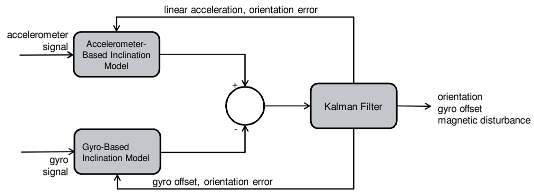
- The MagCal / eCompass example is nice because it can be explicitly calculated
- Other systems can be much more complex
- If we can model a system as a set of state variables, then we can use a Kalman filter to separate noise from desired system behavior
- A Kalman filter essentially does a linear regression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



freescale External Use | 77

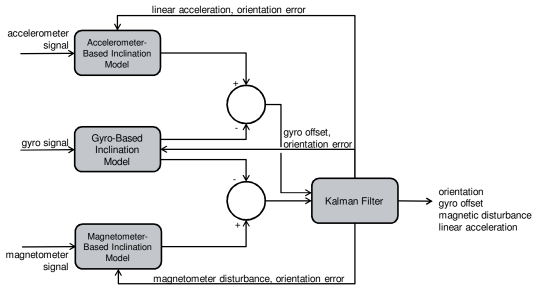
12/19/2014 11:25:38 AM

6-Axis Accel + Gyro Indirect Kalman Filter



- This algorithm has no sense of magnetic north
- The output orientation may drift about the gravity vector as a result of uncorrected gyro gain errors

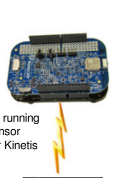
9-axis accel + mag + gyro Indirect Kalman Filter



Computing information is only half the puzzle.

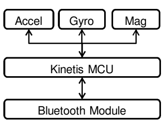
You have to do something with it. Enter…

The Freescale Sensor Fusion Toolbox



Embedded board running the Freescale Sensor Fusion Library for Kinetis

=



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

Phone/tablet running the Freescale Sensor Fusion Toolbox for Android

The Freescale Sensor Fusion Toolbox

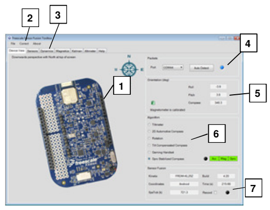
- Provides visualization functions for the fusion library
- Allows you to experiment with different sensor/algorithm choices
- Gives you access to raw sensor data
- Allows you to log sensor and fusion data for later use
- Works with demo and development versions of the Freescale Sensor Fusion Library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

The Freescale Sensor Fusion Toolbox Features by Platform

Feature	Android	PC
Bluetooth wireless link	✓	Requires BT on PC (built-in or dongle)
Ethernet wireless link	On WiGo board only	-
UART over USB	-	✓[1]
OS requirements	>= Android 3.0	>= Windows 7.0
Support for native sensors	✓	-
Device View	✓	✓
Panorama View	✓	-
Statistics View	✓	-
Canvas View	✓	-
Orientation XY Plots	-	✓
Inertial XY Plots	-	✓
Magnetics	-	✓
Kalman	-	✓
Altimeter XY Plots	-	✓
Data Logging Capability	✓	✓
Integrated documentation	✓	✓
Availability	Google Play	Freescale website
Price	Free	Free

[1] FRDM_K64F and FRDM_K20D50M projects require a Processor Expert configuration change to run in wired mode.

PC Version – Device View

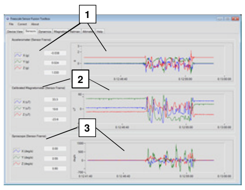


Figures are from 28 August 2014 build of the application. Appearance may vary for other versions.

1. Rotating 3D PCB display
2. Image align function
3. Navigation Tabs for:
   - Sensors Data Tab
   - Dynamics Tab
   - Magnetics
   - Kalman
   - Altimeter
   - 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.

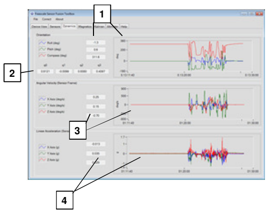
PC Version – Sensors Tab



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.

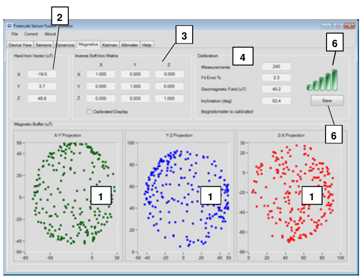
PC Version – Dynamics Tab



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.

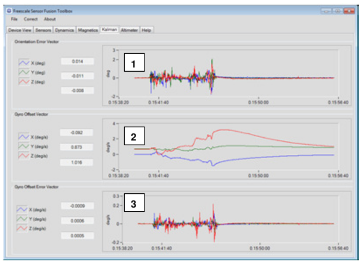
PC Version – MagneticsTab



1. 2D representation of the data point "cloud" used for hard/soft iron compensation
2. Computed hard iron vector
3. Soft iron matrix
4. Statistics
5. Calibration status light
6. Save to text file

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

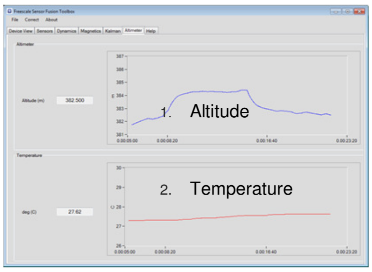
PC Version – Kalman Tab



Use this tab to view how well your sensor fusion "digests" changes in its environment.

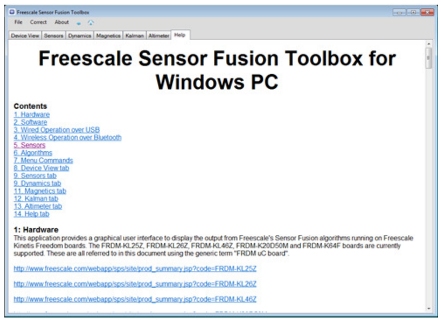
1. Error in orientation estimate (X,Y,Z)
2. Computed gyro offset
3. Error in gyro offset estimate (X,Y,Z)

PC Version – Altimeter Tab



Not available when using FRDM-FXS-9AXIS board

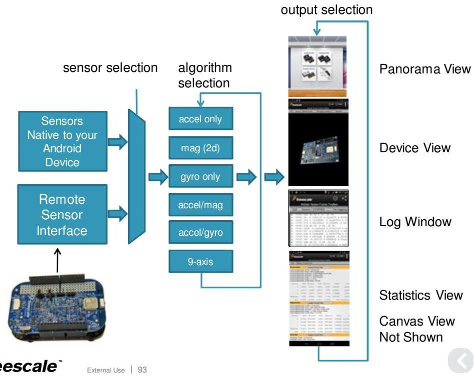
PC Version – Help tab



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 component in these projects if you wish to use a wired UART/USB interface. Additional detail is in the user manual.

Android Version Program Operation

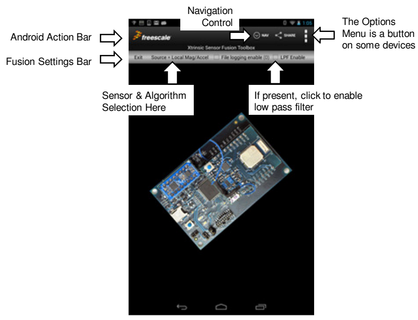


ENU Frame of Reference

X = East
Y = North
Z = up

+Z is out of page

Application Controls

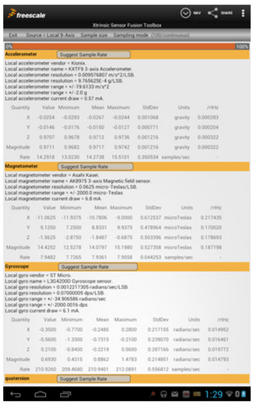


freescale

Stats Page

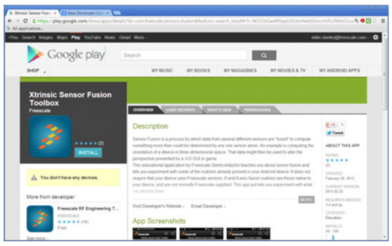
For mag / accel / gyro and rotation, the “Statistics” Views displays:
- sensor description
- current sensor value
- min / mean / max values
- standard deviation
- noise / √Hz

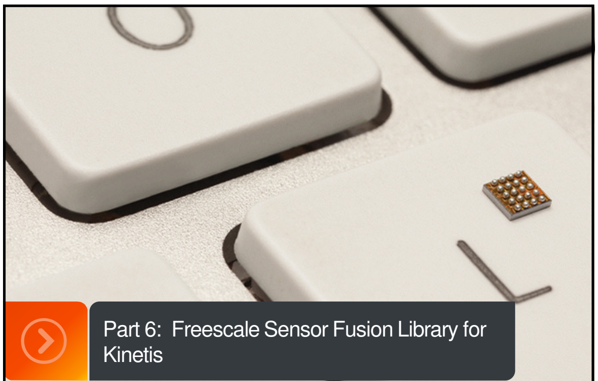
When used with the “local” sensor sources, this is a great way to gain insight into devices from the competition!



If you would like to try it…

http://play.google.com/store/apps/details?id=com.freescale.sensors.sfusion

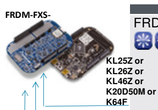




Part 6: Freescale Sensor Fusion Library for Kinetis

Freescale Sensor Expansion Boards

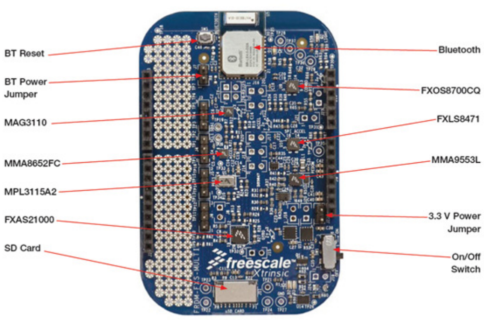
Kinetis KL25Z and K20D50M compatible Freescale Sensor Expansion Boards



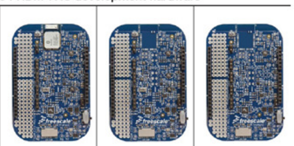
Part Number	Description	Pricing	Availability
FRDM-FXS-MULTI*	Freescale Sensor Expansion board MPL3115A2 MMA8652 FXAS21000 FXOS8700 FXLS8471 MMA955x MAG3110	$50	Now
FRDM-FXS-MULTI-B*	Freescale Sensor Expansion board with Bluetooth and Battery MPL3115A2 MMA8652 FXAS21000 FXOS8700 FXLS8471 MMA955x MAG3110	$125	Now
FRDM-FXS-9AXIS*	Freescale Sensor Expansion board with only 2 sensors FXAS21000 FXOS8700	$30	Now

Freescale Sensor Expansion Boards

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



Freedom Xtrinsic FRDM-FXS development hardware



	FRDM-FXS-MULTI-B	FRDM-FXS-MULTI	FRDM-FXS-9AXIS
Compatible Freedom Development Hardware	FRDM-KL25Z FRDM-K20D50M	FRDM-KL25Z FRDM-K20D50M	FRDM-KL25Z FRDM-K20D50M
Arduino R3-compatible board	√	√	√
FXAS21000 Gyroscope	√	√	√
FXOS8700CQ	√	√	√
MMA8652FC Accelerometer	√	√	
MPL3115A2 Altimeter/ Barometer Sensor	√	√	
FXLS8471 Accelerometer	√	√	
MMA9553L Pedometer	√	√	
MAG3110 Magnetometer	√	√	
Bluetooth Module and Battery	√		

PREÇO CIF NO DIA 02 / 12 / 14 (DÓLAR A R$ 2,5624)

DS	Part Number	Fabricante	Preço Unitario (R$)	Estoque (EUA)	Prazo de Entrega
	FRDM-FXS-MULTI-B	Freescale / On Semi	$ 799,1129	2 pçs	Est. USA entrega 2/3 semanas
	FRDM-FXS-MULTI	Freescale / On Semi	$ 319,6452	0 pçs	7 semanas
	FRDM-FXS-9AXIS	Freescale / On Semi	$ 191,7871	2 pçs	Est. USA entrega 2/3 semanas
	FRDM-KE06Z	Freescale / On Semi	$ 82,8069	89 pçs	Est. USA entrega 2/3 semanas
	FRDM-KL25Z	Freescale / On Semi	$ 68,2160	0 pçs	7 semanas
	FRDM-K64F	Freescale / On Semi	$ 223,7516	13 pçs	Est. USA entrega 2/3 semanas

Ordering Details

Component	Price	Location
Sensor Fusion Library for Kinetis MCUs	Free	http://www.freescale.com/sensorfusion
Freescale Freedom Development Platform	KL25Z = $12.95 KL26Z = $15.00 KL46Z = $15.00 K20D50M = $18.00 K64F = $29.00	http://www.freescale.com/freedom
Freescale Freedom Development Platform for Multiple Freescale Sensors	$30 $50 $125	http:www.freescale.com/FRDM-FXS-9AXIS http:www.freescale.com/FRDM-FXS-MULTI http:www.freescale.com/FRDM-FXS-MULTI-B
Freescale Sensor Fusion Toolboxes For PC	Free	http://www.freescale.com/sensorfusion
Freescale Sensor Fusion Toolboxes Android	Free	https://play.google.com/store/apps/details?id=com.freescale.sensors.sfusion
Freescale Sensors	Various	http://www.freescale.com/sensors

Prices are current as of 6 Sept, 2014. They may vary in the future.

Sensor Fusion Development Kit

Development Kit
- Enables quick development and prototype of sensor fusion applications
- Includes
  - Kinetis FRDM-K64F Freedom board
  - Freedom Development Platform for Freescale Sensors with Bluetooth®
- Part numbers
  - FRDM-SFUSION with community support ($170)
  - FRDM-SFUSION-S with 50 hours commercial support ($10K)

Commercial Support
- Reduces project risk, accelerates time to market
- Prioritized and dedicated access
- Guaranteed response time
- Senior level developer access
- Private portal with customer reporting and dedicated escalation path
- Annual Subscription

Freescale Sensor Fusion Library for Kinetis MCUs

- 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
  - 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-S** with 50 hours of commercial support
    - **FRDM-SFUSION** 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
- Community support available at https://community.freescale.com/community/sensors/sensorfusion
- Contract support services offered by Freescale. Contact: sfsw@freescale.com for details.

Features vs. Sensor Set

Feature	Accel only	Accel + gyro	Accel + mag	Accel + mag + gyro
Filter Type	Low Pass	Indirect Kalman	Low Pass	Indirect Kalman
Roll / Pitch / Tilt in degrees	Yes	Yes	Yes	Yes
Yaw in degrees	No	No	Yes	Yes
Angular Rate[1] in degrees/second	virtual 2 axis[2]	Yes	virtual 3 axis	Yes
Compass heading (magnetic north) in degrees	No	No	Yes	Yes
Quaternion and rotation vector	Yes	Yes	Yes	Yes
Rotation matrix	Yes	Yes	Yes	Yes
Linear acceleration separate from gravity	No	Yes	No	Yes
NED (North-East-Down Frame of Reference	Yes[3]	Yes[3]	Yes	Yes
ENU (Windows 8 variant) Frame of Reference	Yes[3]	Yes[3]	Yes	Yes
ENU (Android variant) Frame of Reference	Yes[3]	Yes[3]	Yes	Yes
Magnetic calibration included	No	No	Yes	Yes
Gyro offset calibration included	N/A	Yes	N/A	Yes
FRDM-KL25Z board support	Yes	Yes	Yes	Yes
FRDM-KL26Z board support	Yes	Yes	Yes	Yes
FRDM-KL46Z board support	Yes	Yes	Yes	Yes
FRDM-K20D50M board support	Yes	Yes	Yes	Yes
FRDM-K64F board support	Yes	Yes	Yes	Yes

1. Angular rate for configurations with a gyro include corrections for gyro offset
2. Subject to well-known limitation of being blind to rotation about axes aligned with gravity
3. These solutions do not include a magnetometer, therefore there is no sense of compass heading

freescale External Use | 108

Option Details

Feature	Details
License	Free when used with Freescale sensors (see license file for details)
CPU selection	The ANSI C99 source code was optimized on Freescale Kinetis MCUs based upon ARM™ Cortex M0+, M4 and M4F processors, but should be portable to any CPU.
Board customizable	Yes[1]
Sensor sample rate	Programmable
Fusion rate	Programmable, typically = sample rate/N
Frame of Reference	Programmable (NED, Android, or Windows 8)
Algorithms Executing	Any combination of those shown in the prior slide
Sleep mode enabled between samples/calculations	Programmable
RTOS	MQX-Lite
Code flexibility	All code is supplied in source form
Access to Processor Expert	Yes
Product Deliverables	* Datasheet, User guide, Application Notes * Template CodeWarrior projects * Pre-compiled s-record files

[1] FRDM_KL25Z, KL26Z, KL46Z, K20D50M and K64F are supported "out of the box" and may be used as templates for other board/MCU combinations..

freescale External Use | 109

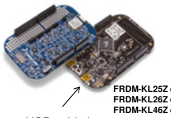
Part 5: Play with fusion options

freescale External Use | 110

For this demo

You need
- Freescale Freedom boards shown
- USB cable
- Freescale Sensor Fusion Toolbox running on a Windows Laptop
  (C:\Program Files\Freescale\Freescale Sensor Fusion Toolbox\SensorFusion.exe)
- Freescale Sensor Fusion Library for Kinetis MCUs

FRDM-FXS-9AXIS or
FRDM-FXS-MULTI or
FRDM-FXS-MULTI-B



FRDM-KL25Z or
FRDM-KL26Z or
FRDM-KL46Z or

Plug your USB cable in this connector

Make sure the switch on the top sensor board is "on".
If you have a MULTI-B board, remove jumper J7

freescale External Use | 111

Experiment with each of the following options

#	Option	Comments
1	Accelerometer	Roll & Pitch only, no yaw
2	Gyroscope	Roll, Pitch & Yaw, but no absolute reference
3	Accelerometer + Magnetometer (eCompass)	Roll, Pitch & Yaw relative to earth frame, but sensitive to magnetic interference and linear acceleration
4	Accelerometer + Gyroscope	Roll, Pitch with respect to horizontal plane, yaw is relative
5	9-Axis Accelerometer + Gyroscope + Magnetometer	Roll, Pitch & Yaw relative to earth frame, relatively independent of magnetic interference and linear acceleration

Experiment with each tab function on the fusion toolbox

freescale External Use | 112

Sensor Fusion Library Details

freescale External Use | 113

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 project templates.
  - CodeWarrior can be downloaded from http://www.freescale.com/codewarrior.
  - Kinetis Design Studio can be downloaded from http://www.freescale.com/kds
- 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/sensorfusion
- Fusion libraries and example projects supplied by the Freescale Sensor Solutions Division
- Development board(s)[1] with:
  - Kinetis Cortex-M0+, M4 or M4F MCU
  - Freescale FXOS8700CQ 3-axis magnetometer + 3 axis accelerometer
  - Freescale FXAS21000 3-axis gyroscope

[1] See details on "Freescale Sensor Expansion Boards". Additional sensor combinations are supported in build.h. And of course, you can add your own! Future expansion boards may replace the FXAS21000 with the FXAS21002, which is also supported.

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);
  - void UserHighFrequencyTaskInit(void) ; // runs once, the first time through the 200Hz task
  - void UserHighFrequencyTaskRun(void); // runs each time the 200Hz task runs
  - void UserMediumFrequencyTaskInit(void); // runs once, the first time through the 25Hz task
  - void UserMediumFrequencyTaskRun(void); // runs each time the 25Hz task runs
- Sensor and fusion values are simply read from predefined global structures

user_tasks.c Template Page 1 of 3

```
#include "Cpu.h"
#include "Events.h"
#include "mqx_tasks.h"
#include "UART.h"
#include "include_all.h"

void UserStartup(void) {
        // The following UART function call initializes Bluetooth communications used by the
        // Freescale Sensor Fusion Toolbox.  If the developer is not using the toolbox,
        // these can be removed.
        //
        // initialize BlueRadios Bluetooth module
        BlueRadios_Init(UART2_DeviceData);

        // put code here to be executed at the end of the RTOS startup sequence.
        //
        // PUT YOUR CODE HERE
        //

        return;
```


user_tasks.c Template Page 2 of 3

```
void UserHighFrequencyTaskInit(void) {
    // User code to be executed ONE TIME the first time the high frequency task is run.
    //
    // PUT YOUR CODE HERE
    //
    return;
}

void UserMediumFrequencyTaskInit(void) {
    // User code to be executed ONE TIME the first time the medium frequency task is run
    //
    // PUT YOUR CODE HERE
    //
    return;
}

void UserHighFrequencyTaskRun(void) {
    // The default frequency at which this code runs is 200Hz.
    // This code runs after sensors are sampled.
    // In general, try to keep "high intensity" code out of UserHighFrequencyTaskRun.
    // The high frequency task also has highest priority.
    //
    // PUT YOUR CODE HERE
    //
    return;
}
```


user_tasks.c Template Page 3 of 3

```
void UserMediumFrequencyTaskRun(void) {
        // This code runs after the Kalman filter loop
        // The default frequency at which this code runs is 25Hz.

        // The following UART function constructs and sends Bluetooth packets used by the
        // Freescale Sensor Fusion Toolbox.  If the developer is not using the toolbox,
        // it can be removed.
        // transmit orientation over the radio link
        CreateAndSendBluetoothPacketsViaUART(UART2_DeviceData);

        //
        // PUT YOUR CODE HERE
        //
        return;
}
```

Steps to use:
1. Import project into CodeWarrior
2. Add your code as shown above
3. Build
4. Download and run

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	AccelSensor
Magnetometer	thisMag	MagSensor
Gyroscope	thisGyro	GyroSensor

Output Global Data Structures defined in tasks.h

Pointer Function	Structure Name	Structure Type
Altimeter results	thisSV_1DOF_P_BASIC	SV_1DOF_P_BASIC
3-axis Accelerometer results	thisSV_3DOF_G_BASIC	SV_3DOF_G_BASIC
2D Magnetic-only eCompass results	thisSV_3DOF_B_BASIC	SV_3DOF_B_BASIC
Gyro-only orientation	thisSV_3DOF_Y_BASIC	SV_3DOF_Y_BASIC
eCompass results	thisSV_6DOF_GB_BASIC	SV_6DOF_GB_BASIC
accel+gyro results	thisSV_6DOF_GY_KALMAN	SV_6DOF_GY_KALMAN
9-axis results	thisSV_9DOF_GBY_KALMAN	SV_9DOF_GBY_KALMAN

Location of Variables Within the Global Structures

Description	Data Type	Fusion Algorithm Options			
		G (accel)	GB (eCompass)	GY (accel + gyro)	GBY 9-axis
roll in degrees	float	fLPPhi	fLPPhi	fPhiPl	fPhiPl
pitch in degrees	float	fLPThe	fLPThe	fThePl	fThePl
yaw in degrees	float	fLPPsi	fLPPsi	fPsiPl	fPsiPl
compass heading in degrees	float	fLPRho	fLPRho	fRhoPl	fRhoPl
tilt angle in degrees	float	fLPChi	fLPChi	fChiPl	fChiPl
magnetic inclination angle in degrees	float	N/A	fDelta fLPDelta	N/A	fDeltaPl
geomagnetic vector (microTeslas, global frame)	float	N/A	N/A	N/A	fmGl[3]
gyro offset in degrees/sec	float	N/A	N/A	fbPl[3]	fbPl[3]
linear acceleration in the sensor frame in gravities	float	N/A	N/A	faSePl[3]	faSePl[3]
linear acceleration in the global frame in gravities	float	N/A	N/A	N/A	faGlPl[3]
quaternion (unitless)	fquaternion	fq fLPq	fq fLPq	fqPl	fqPl
angular velocity in dps	float	fOmega[3][1]	fOmega[3]	fOmega[3][2]	fOmega[3] [2]
orientation matrix (unitless)	float	fR[3][3] fLPR[3][3]	fR[3][3] fLPR[3][3]	fRPl[3][3]	fRPl[3][3]
rotation vector	float	fLPRVec[3]	fLPRvec[3]	fRVecPl[3]	fRVecPl[3]
time interval in seconds	float	fdeltat	fdeltat	fdeltat	fdeltat

Data elements for altimeter, 2D eCompass, and gyro only are not shown.

Here is an Example of Grabbing Quaternion Values

```
struct fquaternion fq;          // quaternion
float q0, q1, q2, q3;

//fq = thisSV_3DOF_G_BASIC.fLPq;  // OR
//fq = thisSV_6DOF_GB_BASIC.fLPq;  // OR
//fq = thisSV_6DOF_GY_KALMAN.fqPl; // OR
fq = thisSV_9DOF_GBY_KALMAN.fqPl;

q0 = fq.q0;
q1 = fq.q1;
q2 = fq.q2;
q3 = fq.q3;

// more details/examples are presented in the following section
```


Example: Reading Euler Angles

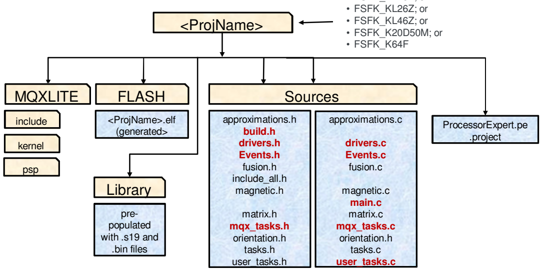
```
Using 3-axis model:
  float roll = thisSV_3DOF_G_BASIC.fLPPhi;
  float pitch = thisSV_3DOF_G_BASIC.fLPThe;
  float yaw = thisSV_3DOF_G_BASIC.fLPPsi;

Using 6-axis accel + mag (eCompass) model:
  float roll = thisSV_6DOF_GB_BASIC.fLPPhi;
  float pitch = thisSV_6DOF_GB_BASIC.fLPThe;
  float yaw = thisSV_6DOF_GB_BASIC.fLPPsi;

Using 6-axis accel + gyro Kalman filter model:
  float roll = thisSV_6DOF_GY_KALMAN.fPhiPl;
  float pitch = thisSV_6DOF_GY_KALMAN.fThePl;
  float yaw = thisSV_6DOF_GY_KALMAN.fPsiPl;

Using 9-axis Kalman filter model:
  float roll = thisSV_9DOF_GBY_KALMAN.fPhiPl;
  float pitch = thisSV_9DOF_GBY_KALMAN.fThePl;
  float yaw = thisSV_9DOF_GBY_KALMAN.fPsiPl;
```


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

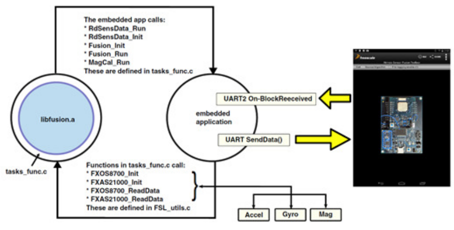
Product Development Kit Structure

As shipped:
- FSFK_KL25Z; or
- FSFK_KL26Z; or
- FSFK_KL46Z; or
- FSFK_K20D50M; or
- FSFK_K64F



Files in **bold red** are most likely to be customized on a per project basis.

3.2 Project Overview



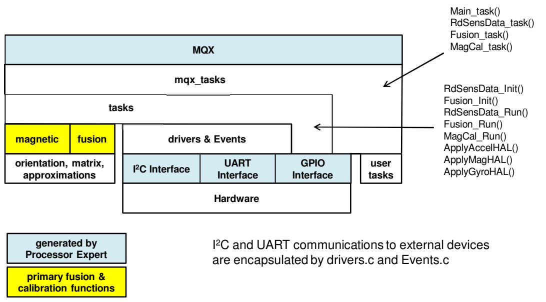
Source File Descriptions

Files	Description
approximations.c approximations.h	Reduced accuracy/power trig functions
build.h	Build options consolidated into a single file
drivers.c drivers.h	Initialization of hardware timers and I²C drivers for inertial and magnetic sensors. Contains **CreateAndSendBluetoothPacketsViaUART**().
Events.c Events.h	Callback functions for hardware events. Contains **UART_OnBlockReceived**()
fusion.c fusion.h	This is where the primary sensor fusion routines reside. All 3, 6 and 9-axis fusion routines are here.
include_all.h	A catchall for all the other .h files
magnetic.c magnetic.h	Magnetic calibration functions

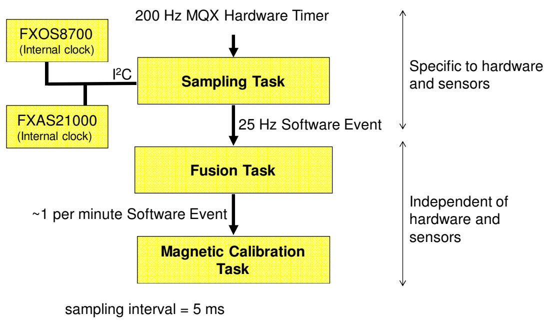
Source File Descriptions

Files	Description
main.c	Initializes and executes MQX
matrix.c matrix.h	Optimized matrix manipulation functions
mqx_tasks.c mqx_tasks.h	Creates and runs the Sampling, Fusion and Calibration tasks which in turn call functions in tasks.c
orientation.c orientation.h	This file contains functions designed to operate on, or compute, orientations. These may be in rotation matrix form, quaternion form, or Euler angles. It also includes functions designed to operate with specific reference frames (Android, Windows 8, NED).
tasks.c tasks.h	tasks.c provides the high level fusion library interface. It also includes the option to apply a Hardware Abstraction Layer (HAL). With proper attention to sensor orientations during PCB design, tasks.c may never need modification.
user_tasks.c user_tasks.h	Placeholder functions for // Put your code here

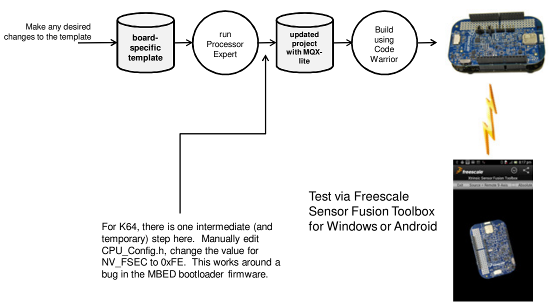
High Level Architecture



Our Sensor Fusion is Partitioned Into 3 Tasks



The Build Process



MCU Resources Used by the Template Projects

Function	FSFK_KL25Z	FSFK_KL26Z	FSFK_KL46Z	FSFK_ K20D50M	FSFK_K64F	Description
Cpu	MKL25Z128VLK4	MKL26Z128VLH4	MKL46Z256VMC4	MK20DX128VLH5	MK64FN1M0VLL12	
LED_RED	PTB18	PTE29	PTE29	PTC3	PTB22	Illuminated when a magnetic calibration is in progress
LED_GREEN	PTB19	PTE31	PTD5	PTD4	PTE26	Flickers when fusion algorithms are running
LED_BLUE	PTD1	PTD5	PTE31	PTA2	PTB21	Currently unused
FTM	LPTMR0	LPTMR0	LPTMR0	LPTMR0	LPTMR0	Low frequency timer drives the 200 Hz sensor read process
UART	UART0 on PTA2:1	UART0 on PTA2:1	UART0 on PTA2:1	UART1 on PTE1:0	UART3 on PTC17:16	Used for Bluetooth communications
I2C	I2C1on PTC2:1	I2C1 on PTC2:1	I2C1 on PTC2:1	I2C0 on PTB1:0	I2C1 on PTC11:10	Communicates to sensors
TestPin_KF_Time	PTC10	PTC10	PTC10	PTC10	PTC7	Output lines used for debug purposes
TestPin_MagCal_ Time	PTC11	PTC11	PTC11	PTC1	PTC5	

Fusion Options Are Controlled Via build.h

```
#ifndef BUILD_H
#define BUILD_H

// PCB HAL options
#define BOARD_WIN8_REV05      0                  // with sensor shield
#define BOARD_FRDM_KL25Z      1                  // with sensor shield
#define BOARD_FRDM_K20D50M    2                  // with sensor shield
#define BOARD_FXLC95000CL     3
#define BOARD_FRDM_KL26Z      4                  // with sensor shield
#define BOARD_FRDM_K64F       5                  // with sensor shield
#define BOARD_FRDM_KL16Z      6                  // with sensor shield
#define BOARD_FRDM_KL46Z      7                  // with sensor shield
#define BOARD_FRDM_KL46Z_STANDALONE  8           // without sensor shield

// enter new PCBs here with incrementing values
// C Compiler Preprocessor define in the CodeWarrior project will choose which board to use
#ifdef REV05
#define THIS_BOARD_ID              BOARD_WIN8_REV05
#endif
#ifdef KL25Z
#define THIS_BOARD_ID              BOARD_FRDM_KL25Z
#endif
```


Fusion Options Are Controlled Via build.h

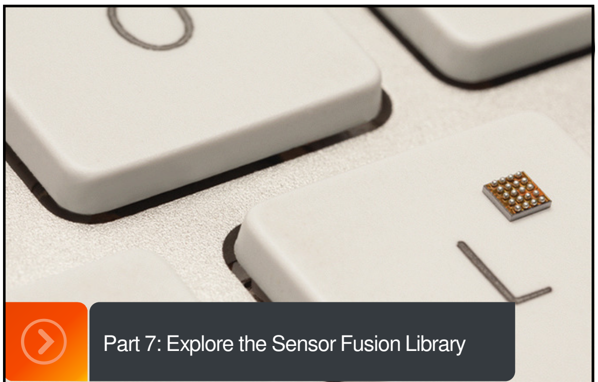
```
#ifdef K20D50M
#define THIS_BOARD_ID              BOARD_FRDM_K20D50M
#endif
#ifdef FXLC95000CL
#define THIS_BOARD_ID              BOARD_FRDM_FXLC95000CL
#endif
#ifdef KL26Z
#define THIS_BOARD_ID              BOARD_FRDM_KL26Z
#endif
#ifdef K64F
#define THIS_BOARD_ID              BOARD_FRDM_K64F
#endif
#ifdef KL16Z
#define THIS_BOARD_ID              BOARD_FRDM_KL16Z
#endif
#ifdef KL46Z
#define THIS_BOARD_ID              BOARD_FRDM_KL46Z
#endif
#ifdef KL46Z_STANDALONE
#define THIS_BOARD_ID              BOARD_FRDM_KL46Z_STANDALONE
#endif
// coordinate system for the build
#define NED 0                      // identifier for NED angle output
#define ANDROID 1                  // identifier for Android angle output
#define WIN8 2                     // identifier for Windows 8 angle output
#define THISCOORDSYSTEM ANDROID    // the coordinate system to be used
```


Fusion Options Are Controlled Via build.h

```
// sensors to be enabled: compile errors will warn if the sensors are not compatible with the algorithms.
// avoid enabling FXOS8700 plus MMA8652 and MAG3110 which will result in sensor read from all sensors
// with the data read first from FXOS8700 and then over-written by data from MMA8652 and MAG3110.
// it will still work but it's a waste of clock cycles.
#define USE_MPL3115
#define USE_FXOS8700
#define USE_FXAS21000
//#define USE_FXAS21002
//#define USE_MMA8652
//#define USE_MAG3110

// enforce a fatal compilation error if the K20D50M board is used with MMA8652
#if (THIS_BOARD_ID == BOARD_FRDM_K20D50M) && defined USE_MMA8652
#error This build creates an I2C conflict between MMA8451 on K20D50M board and MMA8652 on sensor board
#endif
. . .
// normally all enabled: degrees of freedom algorithms to be executed
#define COMPUTE_1DOF_P_BASIC        // 1DOF pressure (altitude) and temperature: (1x pressure)
#define COMPUTE_3DOF_G_BASIC        // 3DOF accel tilt: (1x accel)
#define COMPUTE_3DOF_B_BASIC        // 3DOF mag eCompass (vehicle): (1x mag)
#define COMPUTE_3DOF_Y_BASIC        // 3DOF gyro integration: (1x gyro)
#define COMPUTE_6DOF_GB_BASIC       // 6DOF accel and mag eCompass: (1x accel + 1x mag)
#define COMPUTE_6DOF_GY_KALMAN      // 6DOF accel and gyro (Kalman): (1x accel + 1x gyro)
#define COMPUTE_9DOF_GBY_KALMAN     // 9DOF accel, mag and gyro (Kalman): (1x accel + 1x mag + 1x gyro)
```


Fusion Options Are Controlled Via build.h

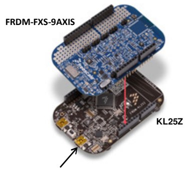
```
// int16 build number sent in Bluetooth debug packet
#define THISBUILD  420

// sampling rate and kalman filter timing
#define FTM_INCLK_HZ                1000000 // int32: 1MHz FTM timer frequency set in PE: do not change
#define SENSORFS                    200     // int32: 200Hz: frequency (Hz) of sensor sampling process
#define OVERSAMPLE_RATIO            8     // int32: 8x: 3DOF, 6DOF, 9DOF run at SENSORFS / OVERSAMPLE_RATIO Hz

// power saving deep sleep
//#define DEEPSLEEP                              // define to enable deep sleep power saving

// UART (Bluetooth) serial port control
//#define UART_OFF                               // define to measure MCU+algorithm current only
```

Part 7: Explore the Sensor Fusion Library

For this lab

You need:
- Freescale Freedom boards shown
- USB cable
- Freescale Sensor Fusion Toolbox running on a Windows Laptop
  (C:\Program Files\Freescale\Freescale Sensor Fusion Toolbox/SensorFusion.exe)
- FSFK_KL25Z project template
  (pre-installed on FTF laptops at C:/Temp)



Plug your USB cable in this connector

You will install updated software images on your board.

Make sure the KL25Z switch is "on"

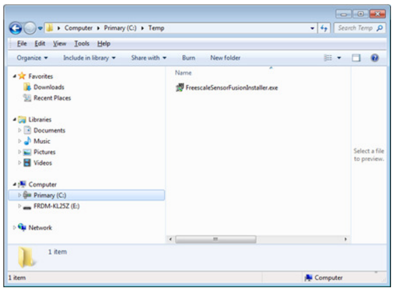
Note: The same process described here works for any of the fusion library template projects. You can use any of KL25Z, KL26Z, KL46Z, K20D50M and K64F Freedom boards.

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

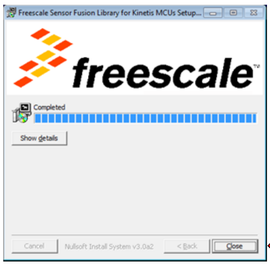
a. Select the destination folder (automatically defaults to the folder in which you placed the installer).
b. Click "Next"

Installation Step 5

a. Select your choice of kits (defaults to CodeWarrior Fusion Projects and documentation).
b. Click "Install"

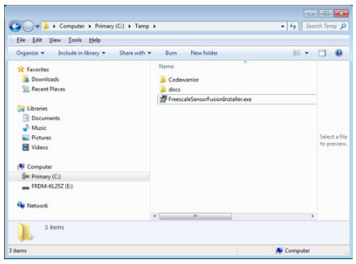
Installation Step 6

a. Click “Close” to complete installation



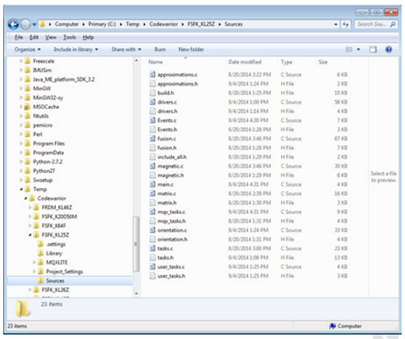
Installation Step 7

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



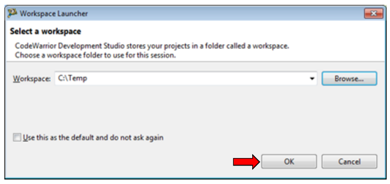
Installation Step 8

a. Expand the template folder down into the FSFK_KL25Z/Sources directory
b. Confirm the that the set of files shown below is present



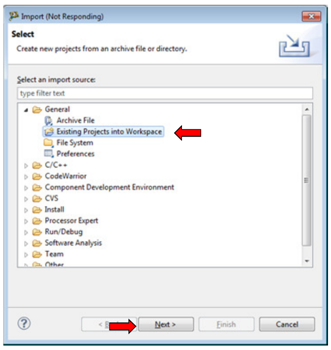
Installation Step 9

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



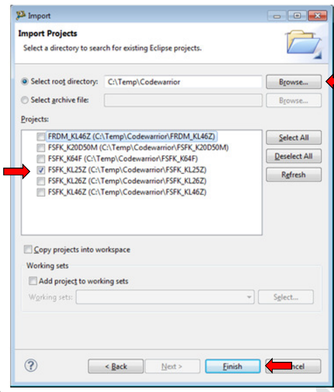
Installation Step 10

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



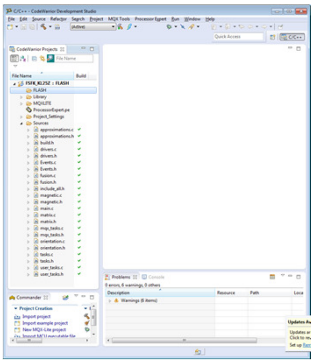
Installation Step 11

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

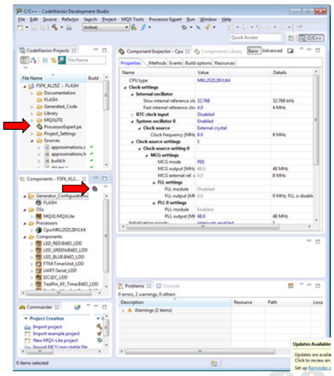


Your project has been successfully installed.

Lab 2, Step 1

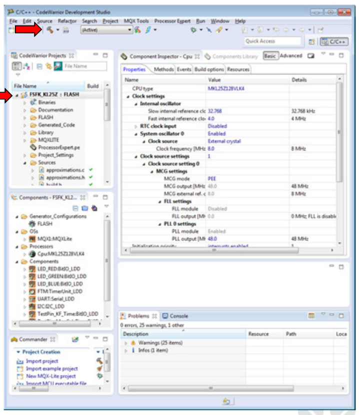
a. Double-click on ProcessorExpert.pe. This will bring up the components browser

b. Click on "Generate Processor Expert Code" icon to run Processor Expert



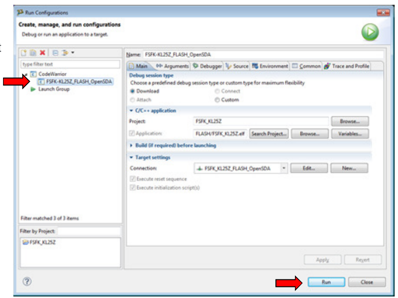
Lab 2, Step 2

a. Select the project name
b. Click on the "Build" icon



Lab 2, Step 3

a) Plug your board in if it is not plugged in
b) Run->Run Configurations
c) Expand CodeWarrior->FSFK-KL25Z_FLASH_OpenSDA (run configuration name may vary)
d) Click "Run"



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

Add:

```
int16 iChi;  // tilt angle
```

at the top of function CreateAndSendBlueToothPacketsViaUART()

Append statements to look up iChi to each of the case options of switch(globals.QuaternionPacketType). The 7 statements needed are:

```
iChi = (int16) (10.0F * thisSV_3DOF_G_BASIC.fLPChi);   // Q3
iChi = (int16) (10.0F * thisSV_3DOF_B_BASIC.fLPChi);  // Q3M
iChi = (int16) (10.0F * thisSV_3DOF_Y_BASIC.fChi);  //  Q3G
iChi = (int16) (10.0F * thisSV_6DOF_GB_BASIC.fLPChi);  //Q6MA
iChi = (int16) (10.0F * thisSV_6DOF_GY_KALMAN.fChiPl); // Q6AG
iChi = (int16) (10.0F * thisSV_9DOF_GBY_KALMAN.fChiPl);  // Q9
iChi = 0; // NOT IMPLEMENTED.  THIS IS A PLACEHOLDER  // QCC
```

In the "if (globals.RPCPacketOn) section, replace:

```
sBufAppendItem(sUARTOutputBuf, &iIndex, (uint8*)&iPhi, 2);
sBufAppendItem(sUARTOutputBuf, &iIndex, (uint8*)&iThe, 2);
sBufAppendItem(sUARTOutputBuf, &iIndex, (uint8*)&iRho, 2);
```

with

```
int16 zero, compassPoint;
// Use iChi instead of iPhi
// Convert compass heading to a cruder N, NE, E, SE, S, SW, W, NW heading
// [12-7]: add the angles (resolution 0.1 deg per count) to the transmit buffer
zero = 0;
compassPoint = iRho-22.5;
compassPoint = compassPoint/450;
compassPoint = compassPoint * 450;
sBufAppendItem(sUARTOutputBuf, &iIndex, (uint8*)&iChi, 2);
sBufAppendItem(sUARTOutputBuf, &iIndex, (uint8*)&zero, 2);
sBufAppendItem(sUARTOutputBuf, &iIndex, (uint8*)&compassPoint, 2);
```

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.

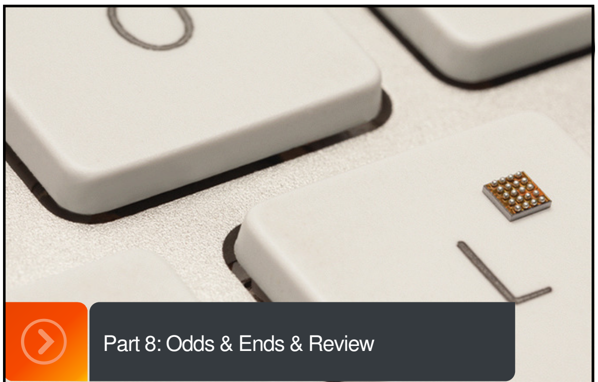
Reminder: Global Data Structures

Pointer Function	Structure Name	Structure Type	defined in include file
Accelerometer	thisAccel	AccelSensor	proj_config.h
Magnetometer	thisMag	MagSensor	
Gyroscope	thisGyro	GyroSensor	
3-axis results	thisSV_3DOF_G_BASIC	SV_3DOF_G_BASIC	tasks_func.h
eCompass results	thisSV_6DOF_GB_BASIC	SB_6DOF_GB_BASIC	
accel+gyro results	thisSV_6DOF_GY_KALMAN	SV_6DOF_GY_KALMAN	
9-axis results	thisSV_9DOF_GBY_KALMAN	SV_9DOF_GBY_KALMAN	

Reminder: Location of variables within the global structures

Description	data type	Fusion Algorithm Options			
		G (accel)	GB (eCompass)	GY (accel + gyro)	GBY 9-axis
roll in degrees	float	fLPPhi	fLPPhi	fPhiPl	fPhiPl
pitch in degrees	float	fLPThe	fLPThe	fThePl	fThePl
yaw in degrees	float	fLPPsi	fLPPsi	fPsiPl	fPsiPl
compass heading in degrees	float	fLPRho	fLPRho	fRhoPl	fRhoPl
tilt angle in degrees	float	fLPChi	fLPChi	fChiPl	fChiPl
magnetic inclination angle in degrees	float	N/A	fDelta fLPDelta	N/A	fDeltaPl
geomagnetic vector (microTeslas, global frame)	float	N/A	N/A	N/A	fmGl[3]
gyro offset in degrees/sec	float	N/A	N/A	fbPl[3]	fbPL[3]
linear acceleration in the sensor frame in gravities	float	N/A	N/A	faSePl[3]	faSePl[3]
linear acceleration in the global frame in gravities	float	N/A	N/A	N/A	faGlPl[3] fLPaGlPl[3]
quaternion (unitless)	fquaternion	fq fLPq	fq fLPq	fqPl	fqPl
angular velocity in dps	float	fOmega[3] [1]	fOmega[3]	fOmega[3] [2]	fOmega[3] [2]
orientation matrix (unitless)	float	fR[3][3] fLPR[3][3]	fR[3][3] fLPR[3][3]	fRPl[3][3]	fRPl[3][3]
rotation vector	float	fLPRVec[3]	fLPRvec[3]	fRVecPl[3]	fRVecPl[3]
time interval in seconds	float	fdeltat	fdeltat	fdeltat	fdeltat

Q & A Opportunity



Part 8: Odds & Ends & Review

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 (sfsw@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/gyro
- http://www.freescale.com/sensors
- http://www.freescale.com/sensortoolbox
- www.twitter.com/Sensorfusion

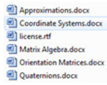
- Blogs: Smart Mobile Devices
  - http://blogs.freescale.com/author/michaelestanley/

- Android App available on Google Play
  - Freescale Sensor Fusion Toolbox

http://www.freescale.com/sensorfusion

freescale External Use | 162

Additional Resources



- Orientation Representations: Part 1
- Orientation Representations: Part 2
- Hard and soft iron magnetic compensation explained
- Freescale E-Compass Software
- "Euler Angles" at http://en.wikipedia.org/wiki/Euler_Angles
- "Introduction to Random Signals and Applied Kalman Filtering", 3rd edition, by Robert Grover brown and Patrick Y.C. Hwang, John Wiley & Sons, 1997.
- "Quaternions and Rotation Sequences", Jack B. Kuipers, Princeton University Press, 1999.
- Matlab computer software by MathWorks - http://www.mathworks.com/products/matlab/

freescale External Use | 163

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

freescale External Use | 164

Thank you for your time and interest.

freescale External Use | 165

freescale™

www.Freescale.com

Auxiliary Slides

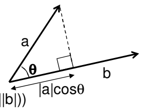
freescale External Use | 167

Use the right rotation representation at each stage of your calculation

Topic	Quaternion	Rotation Matrix
Storage	Requires **16 bytes** of storage in single precision floating point (4 elements at 4 bytes each)	Requires **36 bytes** of storage (9 elements at 4 bytes each)
Computation (for 2 sequential rotations)	4 elements each requiring 4 multiplies and 3 additions = **28 operations**	9 elements, each requiring 3 multiplies and 2 additions = **45 operations**
Vector rotation	Rotating a vector by pre- and post-multiplication of quaternion requires **52 operations**	Rotating a vector via rotation matrix requires **15 operations** (3 elements each requiring 3 multiplies and 2 additions)
Discontinuities	α° about any axis of rotation XYZ is equivalent to -α° about axis of rotation -XYZ .	None
Ease of Understanding	Generally takes a lot of study to understand the details	Easily understood by most engineers
Conversion	From rotation matrix = $\begin{bmatrix} m11 & m12 & m13 \\ m21 & m22 & m23 \\ m31 & m32 & m33 \end{bmatrix}$ we have: $q_0 = 0.5\ sqrt(m_{11} + m_{22} + m_{33} + 1)$ $q_1 = (m_{32} - m_{23}) / (4q_0)$ $q_2 = (m_{13} - m_{31}) / (4q_0)$ $q_3 = (m_{21} - m_{12}) / (4q_0)$	RM = $\begin{bmatrix} 2q_0^2 - 1 + 2q_1^2 & 2q_1q2 - 2q0q3 & 2q_1q_3 + 2q_0q_2 \\ 2q_1q_2 + 2q_0q_3 & 2q_0^2 - 1 + 2q_2^2 & 2q_2q_3 - 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_2q_3 + 2q_0q_1 & 2q_0^2 - 1 + 2q_3^2 \end{bmatrix}$

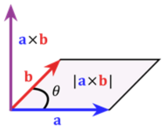
A couple of really useful math identities

If a and b are 3x1 vectors, then



- The **dot product** (a·b) is a scalar:
  - $a \cdot b = a_1b_1 + a_2b_2 + a_3b_3 = |a||b| \cos \theta$
  - θ is the angle between the two vectors = $\cos^{-1}(a \cdot b / (|a||b|))$
- The **cross product** (a X b) is another vector:
  - a X b = |a||b| sin θ n, where n is a unit vector perpendicular to the plane containing a and b

$$a\ X\ b = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} X \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} a_2b_3 - a_3b_2 \\ a_3b_1 - a_1b_3 \\ a_1b_2 - a_2b_1 \end{bmatrix} = \begin{vmatrix} i & j & k \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix}$$



tasks.c

- Defines the following functions:
  - RdSensData_Init (void)
  - RdSensData_Run (void)
  - Fusion_Init (void)
  - Fusion Run (void)
  - MagCal_Run (void)
  - ApplyHal (struct AccelSensor *pthisAccel, struct MagSensor *pthisMag, struct GyroSensor *pthisGyro, int32 irow)

  These are the main functions called from MQX
- 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
  - THISCOORDINATESYSTEM = NED | ANDROID | WIN8
  - Boolean controls (uncomment #define to enable):

#define name	Function
DEEPSLEEP	Enable deep sleep in idle task()
UART_OFF	Disables UART communication for power measurements
COMPUTE_3DOF_G_BASIC	Enable 3-axis accelerometer tilt algorithm
COMPUTE_6DOF_GB_BASIC	Enable 6-axis accel/mag eCompass algorithm
COMPUTE_6DOF_GY_KALMAN	Enable 6-axis accel/gyro Kalman algorithm
COMPUTE_9DOF_GBY_KALMAN	Enable 9-axis Kalman algorithm

Project Configuration

```
#define SENSORFS 200          // int32: frequency (Hz) of sensor sampling process
#define OVERSAMPLE_RATIO 8    // ODR = SENSORFS/OVERSAMPLE_RATIO
```

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

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 FXOS87000CQ combo sensor
FXAS21000_Init() initializes the FXAS21000 gyro
MMA8652_Init() initializes the MMA8652 accelerometer
MAG3110_Init() initializes the MAG3310 magnetometer

FXAS21000_ReadData()
FXOS8700_ReadData()
MMA8652_ReadData()
MAG3110_ReadData()

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()
  - PEX_RTOS_START()

Dependencies Between Project & Fusion Library/Source

Calling Function	Calling Function File	Calls	From
RdSensData_Init	tasks.c	**MPL3115_Init** **FXOS8700_Init** **FXAS21000_Init** **MMA8652_Init** **MAG3110_Init**	drivers.c
RdSensData_Run		**MPL3115_ReadData** **FXOS8700_ReadData** **FXAS21000_ReadData** **MMA8652_ReadData** **MAG3110_ReadData**	
RdSensData_task	mqx_tasks.c	**RdSensData_Run** **RdSensData_Init**	tasks.c
Fusion_task		**Fusion_Init** **Fusion_Run**	
MagCal_task		**MagCal_Run**	

freescale™

www.Freescale.com