InvenSense

InvenSense Inc.
1197 Borregas Ave., Sunnyvale, CA 94089 U.S.A.
Tel: +1 (408) 988-7339 Fax: +1 (408) 988-8104
Website: www.invensense.com

Document Number: AN-MPU-9150IMF-03
Revision: 1.0
Release Date: 04/01/2013

InvenSense Contextual Awareness-SDK™ 5.1.1 User Guide

Release 1.0

A printed copy of this document is
NOT UNDER REVISION CONTROL
unless it is dated and stamped in red ink as,
"REVISION CONTROLLED COPY."

Table of Contents

1 Purpose and Scope

This document describes the InvenSense CA-SDK™ 5.1.1 User Guide. It is intended as a guide for the user to run the CA-SDK™ with the PC or Android application.

2 Revision History

Revision Date	Revision	Description
04/01/2013	1.0	InvenSense CA-SDK™ 5.1.1 User Guide

3 Reference Documents and Software Packages

1. MPU-9250™ CA-SDK™ Reference Board User Guide : General hardware guidelines for CA-SDK board
2. Embedded Motion Apps (eMA) v5.1.1 APIs Specification: Document describes eMA v5.1.1 APIs
3. CA-SDK application: A sample application running on CA-SDK board showing how to use and configure eMA v5.1.1
4. Python eMA client: A PC application to interface and configure CA-SDK application
5. Android CA-SDK client: A android application to interface and configure CA-SDK application

4 CA-SDK™ SDK

4.1 Hardware

Hardware Needed for Programming Using JTAG

- CA-SDK™ SDK (http://www.invensense.com/developers)

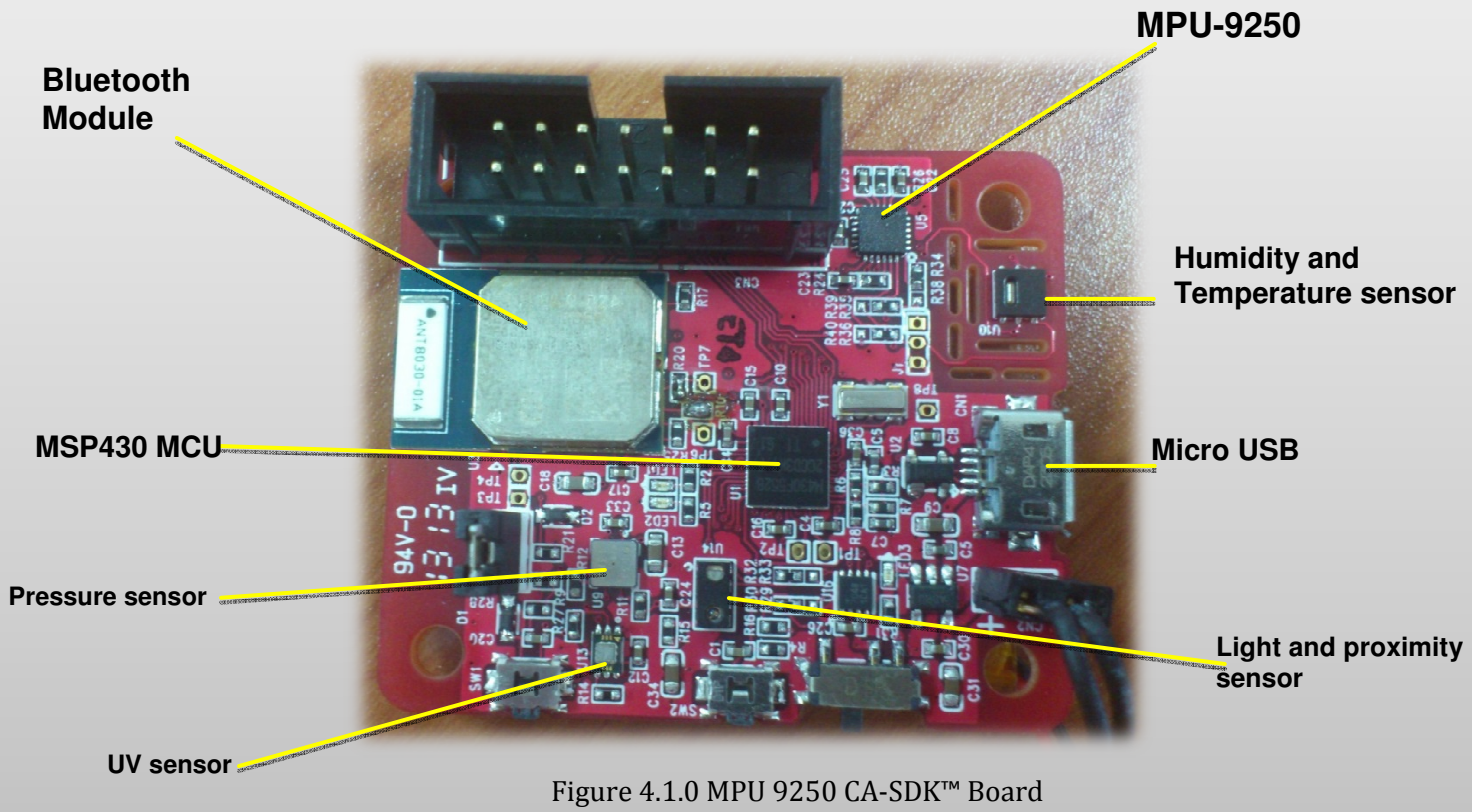
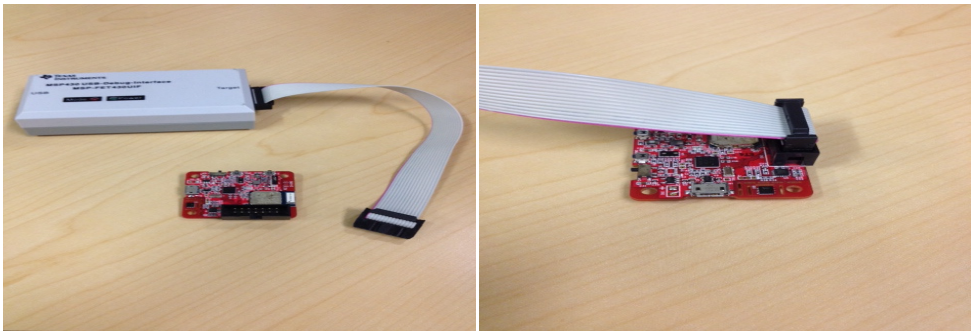


Figure 4.1.0 MPU 9250 CA-SDK™ Board

Please refer to the MPU-9250™ CA-SDK™ Reference Board User Guide for details on each of these sensors and hardware schematics.

- MSP-FET430UIF flash tool (http://www.ti.com/tool/msp-fet430uif)

Figure 4.1.1: Flash tool required for programming



4.2 Software Components Needed to Program the SDK

Loading Software on MCU

Option 1 – PC Program to load the software (that loads firmware on the MCU): Please install the software (http://www.ti.com/tool/flash-programmer) on your PC. For questions on installation or usage of this program please contact TI.

Option 2 - Code Composer Studio: The Code Composer Studio from TI can be used for application development on the SDK. Information on CCS can be obtained from http://www.ti.com/tool/ccstudio or http://processors.wiki.ti.com/index.php/Download_CCS.

Option 3- Boot Strap Loading via USB

BSL will comply with the standards from TI's application note as below.

http://www.ti.com/lit/an/slaa452b/slaa452b.pdf

The option to load the program through BSL requires the existing software on the CA-SDK board to have a BSL jump with a user interface. The software that is pre flashed on all CA-SDK boards that are shipped does come with the option of pressing the button to put the CA-SDK in BSL where the user can update their software using USB interface. A similar interface is recommended for any software updates from the developer to make this feature available.

For the software shipped with CA-SDK, holding the button for two seconds with USB connected allows for BSL jump. To be able to allow it in your program**;** use this line of code:

((void (*)())0x1000)();

For more details refer to checkClick() function in the source code and refer to this document:

4.3 Software

Firmware Embedded MotionApps-5.1.1

1. The CA-SDK™ board comes with preloaded software which interfaces with eMA and outputs quaternion via a Bluetooth link. The code is implemented using Code Composer Studio from TI. The link below provides the documentation for Code Composer.

   http://processors.wiki.ti.com/index.php/Category:Code_Composer_Studio_v5 .

2. The default software settings for the CA-SDK application can be changed according to the eMA ™v5.1.1 API Specification document. Please check the mllite_test.c file in the application release package for details of the implementation.

Feature	Settings
Quaternion Update Rate	20Hz
Quaternion Resolution	32 Bits
Quaternion Range	Full 360º

CA-SDK™ Software

To integrate eMA 5.1.1 you will need to locate the *.Hex file which is included in the CA-SDK software package. Refer to the eMA™ v5.1.1 API Specification for supported API calls needed to build your application. Except for the MPL library the entire software is available in source code for developers to showcase on how to interface the APIs. eMA™ v5.1.1 covers the MPU-9250 sensors which are the gyroscope, accelerometer, and magnetometer.

4.4 InvenSense Sensor

- **InvenSense MPU-9250:** A 3-axis gyroscope, 3-axis accelerometer, and 3-axis magnetometer integrated into a single package**.** The eMA™ v5.1.1 API covers the integration and calibrations for the MPU-9250.

4.5 External Sensors. Datasheets can be found inside the CA-SDK zip file available for download in the Developers Corner. http://www.invensense.com/developers/index.php?_r=default

- **Sensirion SHT21**: Temperature and humidity sensors. CA-SDK software provides the driver for retrieving the raw sensor data from the sensor.
- **Capella CM36682:** Light and proximity sensors. CA-SDK software provides the driver for retrieving the raw sensor data from the sensors. The light sensor is provided as an interface example and the proximity sensor is disabled by default for power concerns.
- **Capella CM3512**: UVI (ultra violet index) sensor. CA-SDK software provides the driver for retrieving the UV index from the sensor.
- **ALSP HSPPA032A**: Pressure sensor. CA-SDK software provides the driver for retrieving the pressure values.

The sensor_probe.c file in the CA-SDK software provides the needed APIs to retrieve the raw sensor data from the external sensors. In the main function, probe_setup_all_external_sensor () will probe, and if any setup is needed will setup the sensors. The functions get_<sensor>_data (data) retrieves the sensor data. Function make_protocol () packages these sensors to be sent out using USB or UART. For further information or other configuration please refer to the datasheet for the external sensors.

Macronix Serial Flash Driver for MX25L25635E

For information on the serial flash, please refer to MX25L25635E, LLD, v0.3.zip on the Macronix website. Also, an example is given in test_flash() function .

Demos

Android client

1. This android application provides a way to evaluate the CA-SDK board. The example application can be downloaded from the play store. Please click the following link. The application works with Bluetooth connectivity.

Figure 4.3.0 is the startup menu in the android application.
It gives an overview of present sensors.

The sensor values are collected from the CA-SDK board, the methods for calculating these values will be given shortly.

Euler angles are updated using quaternion packets received from eMA 5.1.1.

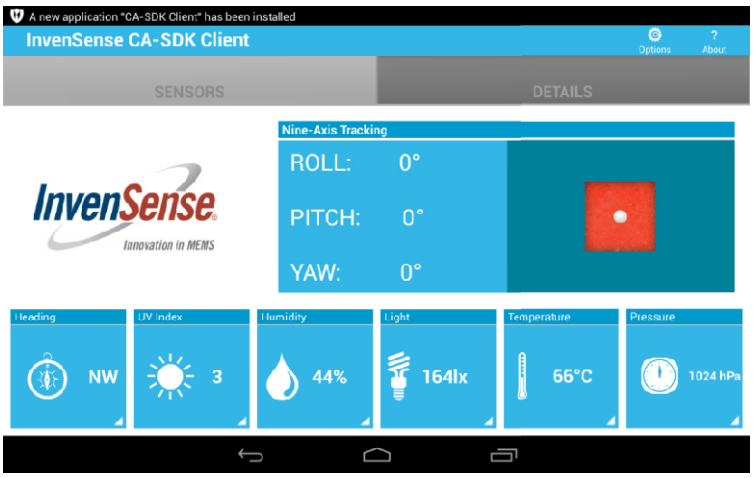


Figure 4.3.0

For more information on external sensors and how to calculate them, please check BluethoothDataReader.java class.

Temperature Sensor

```
Temperature = temperature [0] *256 + temperature [1]
Temperature= (float) (-46.85 + 175.72 * (Temperature / (2.0^16.0)))
```

Humidity Sensor

```
Humidity = humidity[0] *256 + humidity[1]
1- First remove the last three bits
     For Android: float hex = Integer.parseInt("FFF8", 16);
      Humidity = Float.intBitsToFloat(Float.floatToRawIntBits(Humidity)&
      Float.floatToRawIntBits(hex));
2- Humidity = (float) (-6 + 125 * (Humidity /(2 ^ 16)))
```

Light Sensor

```
  currLight = light[0] *256 + light[1]
  currLight = currLight * 0.06103f;
  Light = (currLight * .8f) + (Light * .2f)
```

UV sensor

```
      UV = uv[0] *256 + uv[1]
      UV = (UV * 0.022f) * 10
```

Figure 4.3.1 section consists of all commands the user can send to the CA-SDK board.
On the right, the cube uses quaternion data to rotate. In the middle debugging information will be shown.

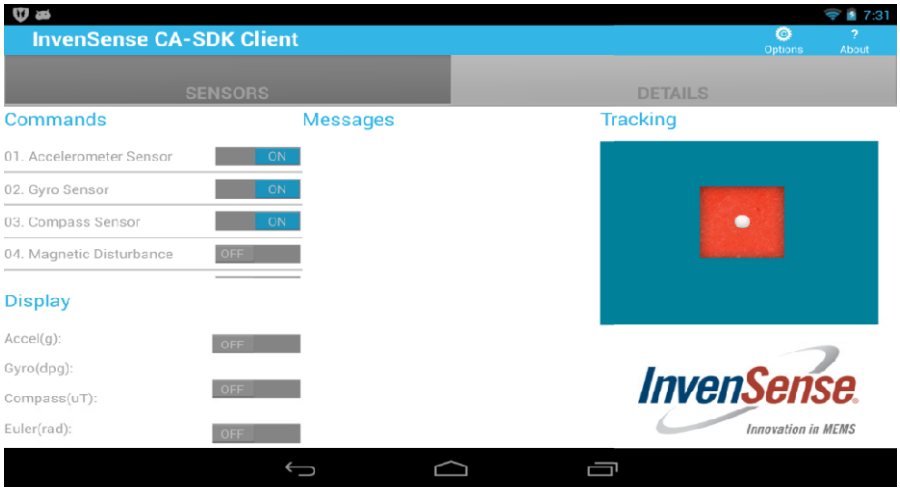


Figure 4.3.1

1- Enable some Display options, and then click on Options-> start Logging.
2- When logging, the Display commands will be disabled. The files will be saved in sdcard/CA-SDKClient.

Three log files will be created, one for external sensors (Pressure, Humidity, Light, UV, Temperature) which is named sensors_TIME_external.csv and another for other sensors, named sensors_TIME.csv. The third file debug_TIME.txt contains debugging Messages.

Python cube debug console app

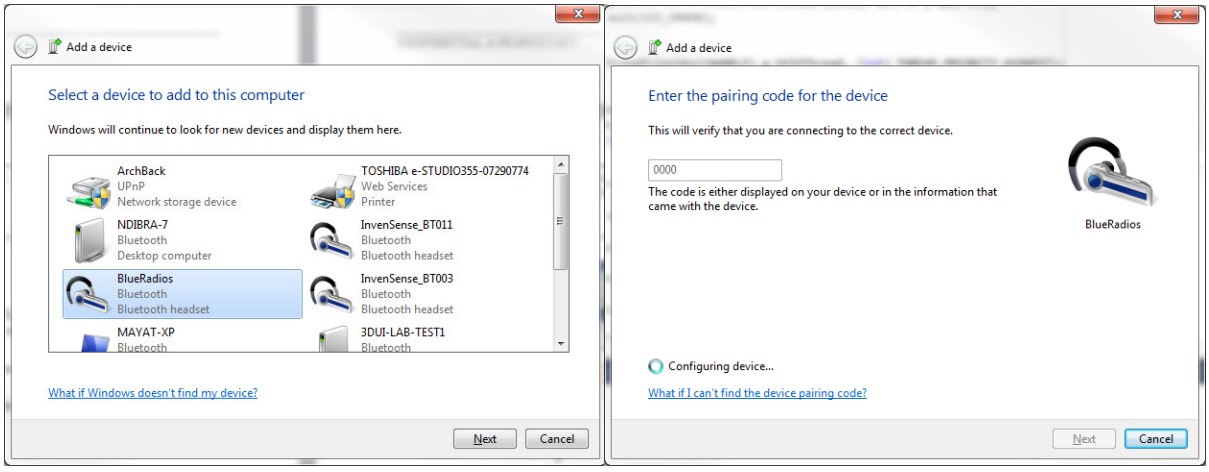
2. The cube demo is written in python script and needs python installed on the computer. This cube demo lets users do more than just observe the orientation. For a complete set of functionality please look at Software Setup to Run the Demo Application in section 5.2 of this document, or the python eMA-client.py script

5 Hardware and Software Setup

5.1 Hardware Setup

Charge the unit via USB connector for 2 hours. Turn on the SDK using the DIP switch on the board.

1. Configure the Bluetooth interface on the PC. Go to "*control Panel\All Control Panel Items\Devices and Printers\"* then choose *"Add a device"* and choose blue "BlueRadios" device. (figure 4.1.0)
2. If the computer asks for the password, enter "default". (Zeros in numerical)
3. The PC will add the required driver to successfully add the device to the PC.
4. Go to the Control Panel\All Control Panel Items\Devices and Printers\ to properties and check the port number of the Bluetooth serial port (figure 5.1.1.1). In this example the port number is 66, but we will call it just COM# from now on. After successfully pairing, power cycle the CA-SDK™ once by flipping the DIP switch OFF and ON.



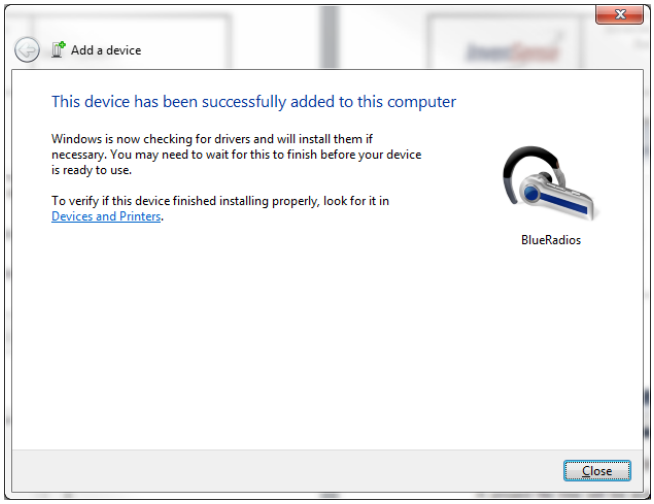


Figure 5.1.1.0 Adding BlueRadios to the PC (Steps 1, 2, 3)

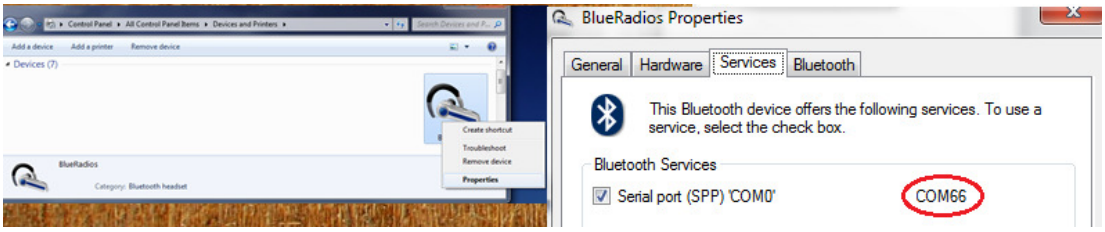


Figure 5.1.1.1 Right click the added device to go to properties and check the port number (Step 4)

5.2 Software Setup to Run the Demo Application

Debug Console App (Optional demo to run if needed)

(eMA-client.py)

A debugging console application is provided to aid developers using the CA-SDK™. This application is provided as Python source code. It has the following functionality:

1. Provides a console for MPL_LOG messages.
2. Provides a simple cube demo for visualizing sensor fusion output.

Because the debug console is provided as Python source code, the user will need to install the following components to have a working Python environment.

1. Python 2.7: The debug console was tested with Python 2.7 on Windows 7 and Windows 8. Download it from http://www.python.org/download/releases/2.7.4/
2. Pyserial 2.6: The debug console requires pyserial to communicate on the serial port. Download the Python 2.6 source code from https://pypi.python.org/pypi/pyserial

3. Pygame 1.9.1: The debug console uses the pygame library to render the cube demo. Release for 86x: http://pygame.org/ftp/pygame-1.9.1.win32-py2.7.msi / Pygame 1.9.2. windows binaries for 64x release from http://www.lfd.uci.edu/~gohlke/pythonlibs/#pygame
4. You must install Python before installing the pyserial and pygame libraries. After installing python use the command prompt to install pyserial (command prompt needs to point to location of pyserial), "python install pyserial.py".
5. You will want to add Python to your PATH for convenience. In Windows, right click on My Computer, select Properties, then in the advanced tab, and select Environment Variables. In the user variables (top section), select Path, click Edit, and add the directory C:\Python27 to the end of the list.
6. Once you have a working Python environment, you can run the debug console application from the command line with the command. To accept data one has to give the python location of the python cube relative to the release folder as below.
7. CA-SDK_With_eMA511_Release.zip\CA-SDK_With_eMPA511_Release\core\eMA-hal\eMA-Client
8. python eMA-client.py <comport>

Where <comport> is the COM# as described in figure 5.1.1.1

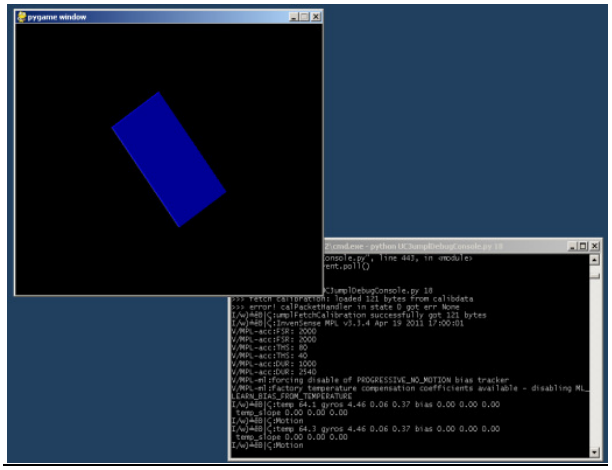


Figure 5.2.2: Cube Demo Application running on console

5.3 User Input

To enable or disable certain functionalities, first enter "**inv**" followed by following case:

case '8': Toggles Accel sensor

case '9': Toggles Gyro sensor

case '0': Toggles Compass sensor

case 'a': Prints Accel sensor

case 'g': Prints Gyro sensor

case 'c': Prints Compass sensor

case 'e': Prints Eular data in radius

case 'r': Prints Rotational Matrix data

case 'q': Prints Quaternion

case 'h': Prints Heading in degrees

case 'k': Prints Pressure, Temperature, UV, Light, Humidity

case 'm': Start/Stop Magnetic disturbance algorithm

case 'n': Get eMA version

case 'w': Get Compass status and Accuracy

case 't' Run Calibration

case '1' to case '4' : Slow down or speed up the rate of sensors

case 'x': reset msp430

case 'v': Toggle LP quaternion

For more cases refer to eMA 5.1.1. Source code in handle_user() function

CTLR + S: Start logging
CTLR + T: Stop logging
The files will be saved in Logs folder created in your python script location.

To Note:

1- For logging, to start user needs to have at least one sensor display enabled.
2- In case a sensor is running on startup of eMA-Client.py, first use 'invx' to go to default setting. It is needed for logging data.
3- Sensors need to be enabled on start of python script and if they are running already, the program is not in default setting, and their data will not be saved in logs.
4- When logging has started, user cannot disable/enable any sensor display or toggle any sensors.
    a. Cases: 8, 9, 0, a, g, c, e, q, r, h, k will not be recognized
5- Two log files will be created, one for external sensors (Pressure, Humidity, Light, UV, Temperature) which is named External_TIME.csv and another named all_TIME.csv

There are three packages that python script accepts,

1- debug_packet: used to show any debugging messages
2- quat_packet: used for rotating cube
3- data_packet: contains sensors information
    a. All conversion formulas which reads data from eMA 5.1.1 and translate it to meaningful numbers can be found in __init__() function of data_packet class.

6 Programming the SDK

6.1 The CA-SDK™ comes pre-flashed with software needed to generate calibrated quaternion. The remainder of this section is needed only if the user would need to modify the embedded firmware.

6.2 How to use eMA 5.1.1 Features and Integrate with an Application

The CA-SDK™ package comes with an example application (that instantiates eMA v5.1.1 API calls) to transmit sensor fusion data via serial UART. The next sections cover details about the eMA architecture and example application code.

6.3 Embedded MotionApps™ Platform Overview

Embedded MotionApps 5.1.1 has been designed with a new architecture as explained below. The new architecture provides customers with a much more modular MotionTracking solution that consumes much less application processor resources, improving the overall power consumption.

There are three basic components in this architecture:

- Embedded Hardware Abstraction Layer (HAL):
  a. This is the central point of communication between eMA, device drivers, and the applications.
  b. HAL provides function calls to initialize, enable, start, stop, and control all features in eMA. Please refer to [2] for further details.
- Embedded MotionProcessing Library (eMPL):
  a. eMA provides a plug-in framework for single function features (e.g. gyro bias calibration, 9-axis MotionFusion, etc.).
  b. eMA provides a well-defined flow to register/unregister, initialization, start, stop, and generate features. Please refer to [2] for further details.
- Embedded Motion Driver:
  a. Device drivers for all supported InvenSense sensors.

6.4 Calibration and Self-test

Embedded MotionApps consists of the following key calibration and self-test features:

Calibration:

- Calibration for gyro bias
  - Fast no motion (<1 second convergence time)
  - Standard no motion (8 second convergence time)
  - SW temperature compensation
- InvenSense proprietary compass calibration
  - Standard fit
  - Vector

- Small Motion using gyro

Self-test:

- Integrated MotionTracking device based self-test for gyroscope in the MPU-9250 (Refer to [2] for further details).
- Integrated MotionTracking device-based self-test for accelerometer in the MPU-9250 (Refer to [2] for further details).
- Integrated MotionTracking device-based self-test for compass in the MPU-9250 (Refer to [2] for further details).
- Self-test is integrated into the respective device drivers.

Please refer to [2] for a brief description of individual calibration functions and recommended settings.

6.5 Ultra-Low Power Features

The ultra-low power features introduced in this release and listed below support fully autonomous operation, completely independent from the application processor:

- DMP-based pedometer
- DMP-based gestures
  - Directional Tap
  - Orientation (including Flip)
  - Screen Orientation
- Low Power Quaternion
- Low Power 6-Axis Sensor Fusion

In general, please refer to [2] on the description, usage, and integration of each of the above features.

6.6 Compilation Setup

The Embedded MotionApps™ Platform requires Code Composer Studio v4.0 or later and the default chip selected is MSP430F5528 which is the one on the CA-SDK board.

6.7 Steps to Bring Up the Project in Code Composer Studio

- Download and install the Code Composer Studio from the TI website (http://www.ti.com/tool/ccstudio). Please note that TI offers a free version of CCS which has restrictions on code size that makes it unusable with the CA-SDK™ hence you would need the full featured version. TI offers a trial version of the software which can be used for configuring your CA-SDK application.
- Download the release package from the Developers Corner and unzip (or 7zip) the folder to extract the Embedded MotionApps eMA-5.1.1 software.
- Open the Code Composer, go to File -> Imports as in figure below.

- Open Existing CCS/CCE Eclipse Projects.
- Browse the folder where the eMA is located and open the project.
- Now the project opens up and it is ready to compile, look for the main.c file to see how the example project is being integrated into eMA.

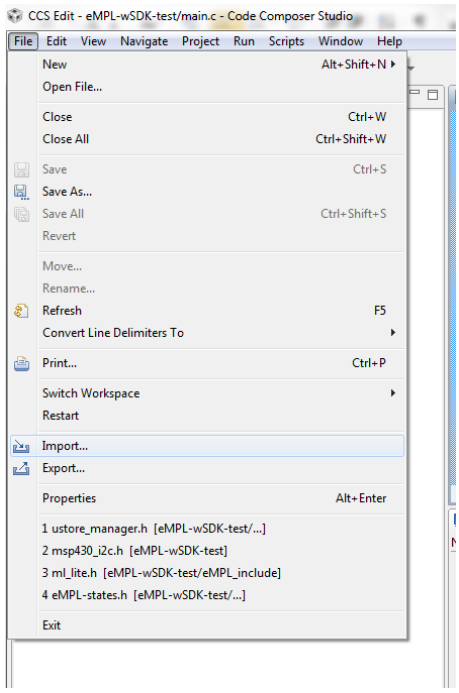


Figure 6.9.0: Select import to import the project

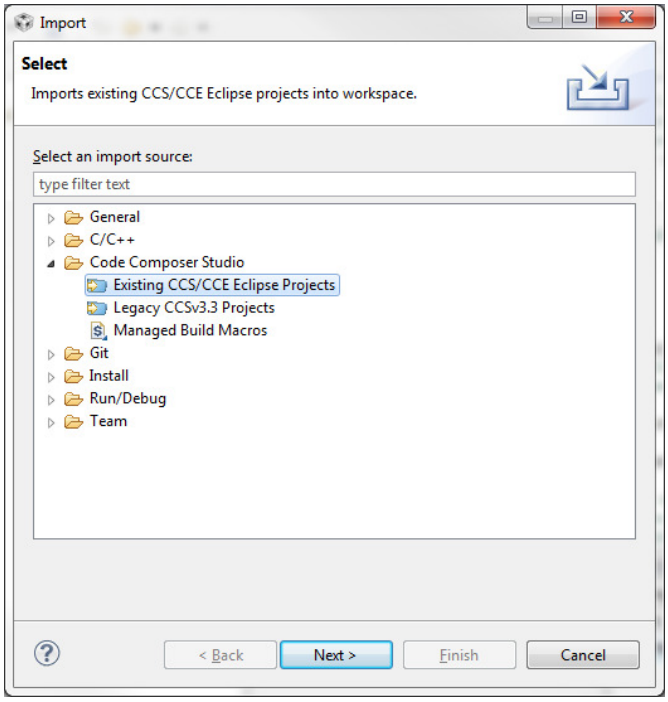


Figure 6.9.1: Open the existing project and put it in the project

6.8 Review of the mllite_test.c File to Build a Sample Application with Embedded MotionApps 5.1.1

The file is self-explanatory in terms of inline documentation and therefore we will go through the important steps only. This example application program is designed to continuously transfer quaternion data at 50Hz update rate. The value of the quaternion is transmitted to the UART port and then transferred via Bluetooth module. The previous section shows how to use the program with PC applications.

- msp430_clock_init(12000000L, 2); msp430_i2c_enable(); msp430_int_enable(); msp430_uart_init(); is needed to setup the system layer functions. This will enable the I2C, clocks, and UART functionality in the project. Initial_Spi(); to enable flash read/write.
- It is very important to initialize using inv_init_mpl(). It should be called first and once.
- Starting eMA using inv_start_mpl() will enable eMA. Here are the major default values which are initialized. There are other defaults sets as well.
    - 20 Hz fifo output rate (This is the default and can be changed)
    - Range for accelerometer can be set using gyro_set_accel_fsr();
    - Gyro range can be set using gyro_set_gyro_fsr()
- In this example application the MCU continuously transmits the data and is set in a loop. Based on the application one can change the use case of the application.
- inv_execute_on_data() will process the data it has received and update all the internal states and features that have been turned on. This function should be called after at least one of inv_build_gyro(), inv_build_accel(), or inv_build_compass() has been called.
- eMA_send_quat(data); this function packages the quaternion and sends it out using the UART module at 115200 baud rate.

7. Known Issues

7.1 For MSP430 security Fuse blown:

http://e2e.ti.com/support/microcontrollers/msp430/f/166/t/166038.aspx

http://e2e.ti.com/support/microcontrollers/msp430/f/166/t/121347.aspx

7.1 Default Frequency is 20 Hz; frequencies above this with compass enabled will cause interruptions. Future versions will address this issue.