# Software User Guide
# Advanced Sensor Fusion
# For DK-42688-V



**InvenSense, a TDK Group Company**
1745 Technology Drive, San Jose, CA 95110 U.S.A
+1(408) 988–7339
invensense.tdk.com

Document Number: AN-000264
Revision: 1.2
Date: 02/01/2023

# TABLE OF CONTENTS

# 1 OVERVIEW

The purpose of this document is to give an overview of the DK-42688-V Development Kit and to guide the user with the hardware and software setup.

The DK-42688-V provides a platform for evaluation and development of TDK's Advanced Motion Tracking solution using ICM-42688-V.

The evaluation kit comes with a dedicated motion sensor (ICM-42688-V) and an advanced sensor fusion library. This combination provides an accurate out-of-box motion tracking solution for a wide range of applications such as Virtual Reality, Augmented Reality, Hearables, Gaming, Wearables, smartphones, tablets, and robotics.

This document will provide details on how to evaluate the solution using DK-42688-V to get high orientation accuracy.

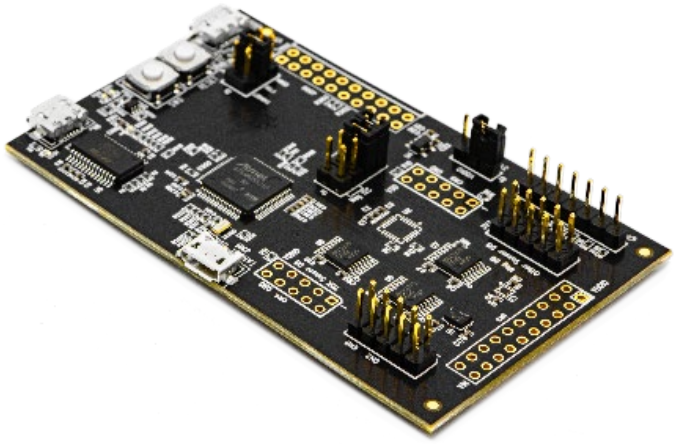


**Figure 1. DK-42688-V main board**

# 2 HARDWARE PLATFORM

The DK-42688-V board is designed around the **Microchip ATSAMG55J19 microcontroller**. A **6-axis ICM-42688-V device** is soldered on the main board to provide accelerometer and gyroscope data. This sensor can be connected either in SPI or $I^2C$ (depending on jumper configuration). An external **AK-09915 magnetometer daughterboard** (aka DB) is also provided and is always connected in $I^2C$.

## 2.1. CONNECTING AND POWERING THE BOARD

First connect the magnetometer daughterboard on the *Other Sensor DB* slot. The connector CN1 on DB (label is written under the PCB) must be on connector CN3 from main board as depicted in Figure 2.

The DK-42688-V is powered by its micro USB connectors (EDBG USB or FTDI USB). Start by connecting a micro-USB cable to one of these ports and to your computer and refer to the next section for jumper configuration.

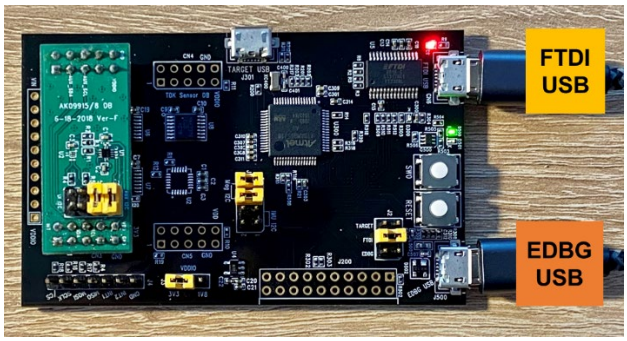



**Figure 2. Board connection**

## 2.2. JUMPERS CONFIGURATION FOR MAIN BOARD

**J2** is used to select the source of the power. Default jumper configuration selects FTDI as power source.

| PINS | J2 (PWR) |
|---|---|
| **1 and 2 (EDBG)** | Open |
| **3 and 4 (FTDI)** | **Short** |
| **5 and 6 (TARGET)** | Open |

**J1** is used to select the sensor communication interface. Refer to the next tables to configure jumpers for the relevant connection. Default configuration is set to SPI for the sensor; the jumper is set accordingly.

| PINS | J1<br>ICM IN SPI | J1<br>ICM IN I2C |
|---|---|---|
| **1 and 2 (SDA)** | Open | **Short** |
| **3 and 4 (SCL)** | Open | **Short** |
| **5 and 6 (AUX-SCL)** | **Short** | Open |
| **7 and 8 (AUX-SDA)** | **Short** | Open |

**J3** is used to select VDDIO voltage. Default jumper configuration selects 3V3.

| PINS | J3<br>VDDIO<br>3V3 |
|---|---|
| 1 and 2 | Short |
| 2 and 3 | Open |

## 2.3. JUMPER CONFIGURATION FOR DAUGHTERBOARD

J1 is used to select how the magnetometer communicates with the MCU (aux interface or regular interface). When the ICM is in SPI mode, it must use the auxiliary interface. When the ICM is in $I^2C$ mode, it must use the regular interface:

| PINS | J1 (AKM DB)<br>ICM IN SPI | J1 (AKM DB)<br>ICM IN I2C |
|---|---|---|
| 1 and 2 | Short | Open |
| 3 and 4 | Short | Open |
| 5 and 6 | Open | Short |
| 7 and 8 | Open | Short |

## 2.4. ILLUSTRATION OF THE BOARD CONFIGURATION FOR SPI AND $I^2C$

Platform configuration when ICM is in SPI (default configuration)

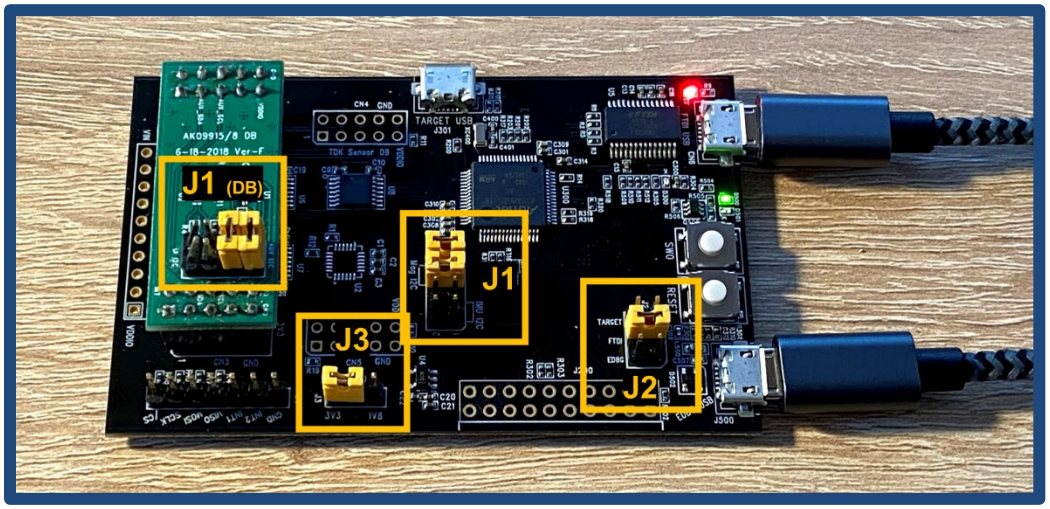



**Figure 3. Platform configuration when ICM is in SPI**

Platform configuration when ICM is in I²C

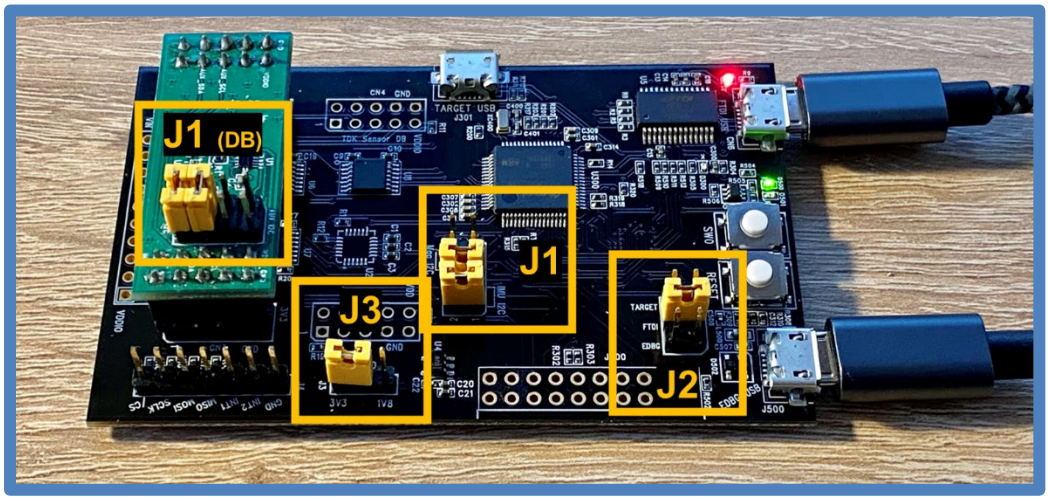



**Figure 4. Platform configuration when ICM is in I²C**

## 2.5. BOARD REFERENCE FRAME

The reference frame of the board is as depicted on the picture below.

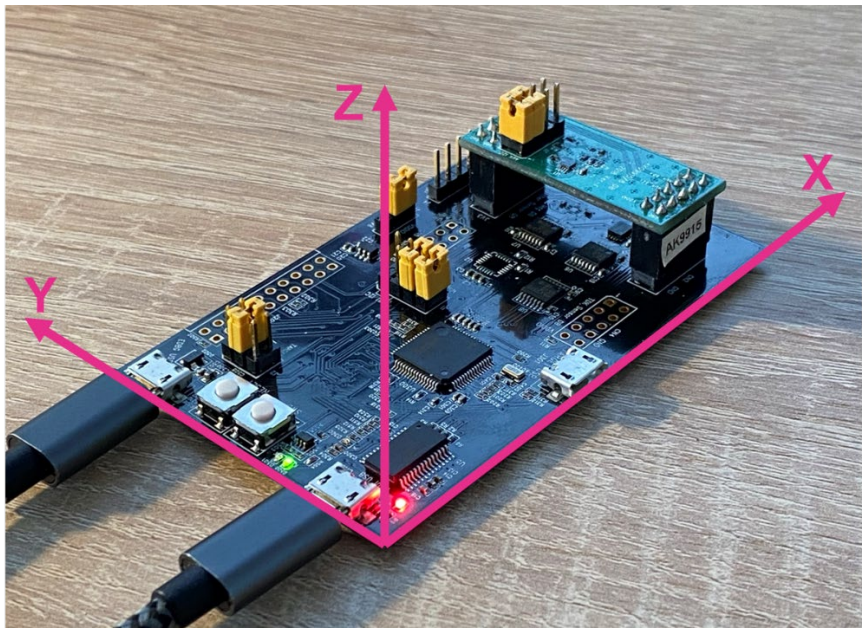



**Figure 5. Board reference frame**

# 3 SOFTWARE ENVIRONMENT

The software package relies on Microchip Studio development suite.

## 3.1. PREREQUISITE

To build and debug the software provided in this package, the following tools are required:

- **Microschip Studio 7** (or above) IDE: https://www.microchip.com/en-us/development-tools-tools-and-software/microchip-studio-for-avr-and-sam-devices#
- **RS232 terminal emulator**: for instance, Putty: http://www.putty.org/
- **FTDI drivers** to retrieve data from FTDI port: https://ftdichip.com/drivers/vcp-drivers/

## 3.2. PACKAGE DESCRIPTION

This package is organized as followed:

- ***doc/*** → Documentation, including this user guide as well as algorithm API details
- ***prebuilt/lib/*** → Algorithm library built for Cortex-M4-FPU with microchip toolchain
- ***release/bin/*** → Firmware binaries
- ***sources/board-hal/*** → Low-level driver for the DK-42688-V platform
- ***sources/examples/*** → Sample code with a ready-to-use Microchip Studio project
- ***sources/Invn/*** → Driver and various utils

## 3.3. OPENING EXAMPLE PROJECTS

There are several projects into the package. Each project is related to one example and contains a dedicated *cproj* file. Here is the list of the available projects:

- **example-asf.cproj** : showcase Advanced Sensor Fusion library
- **example-selftest.cproj** : showcase sensor self-test procedure
- **example-sanity.cproj** : showcase Advanced Sensor Fusion library porting into the current target

Open the desired example by double clicking on the corresponding *cproj* file.

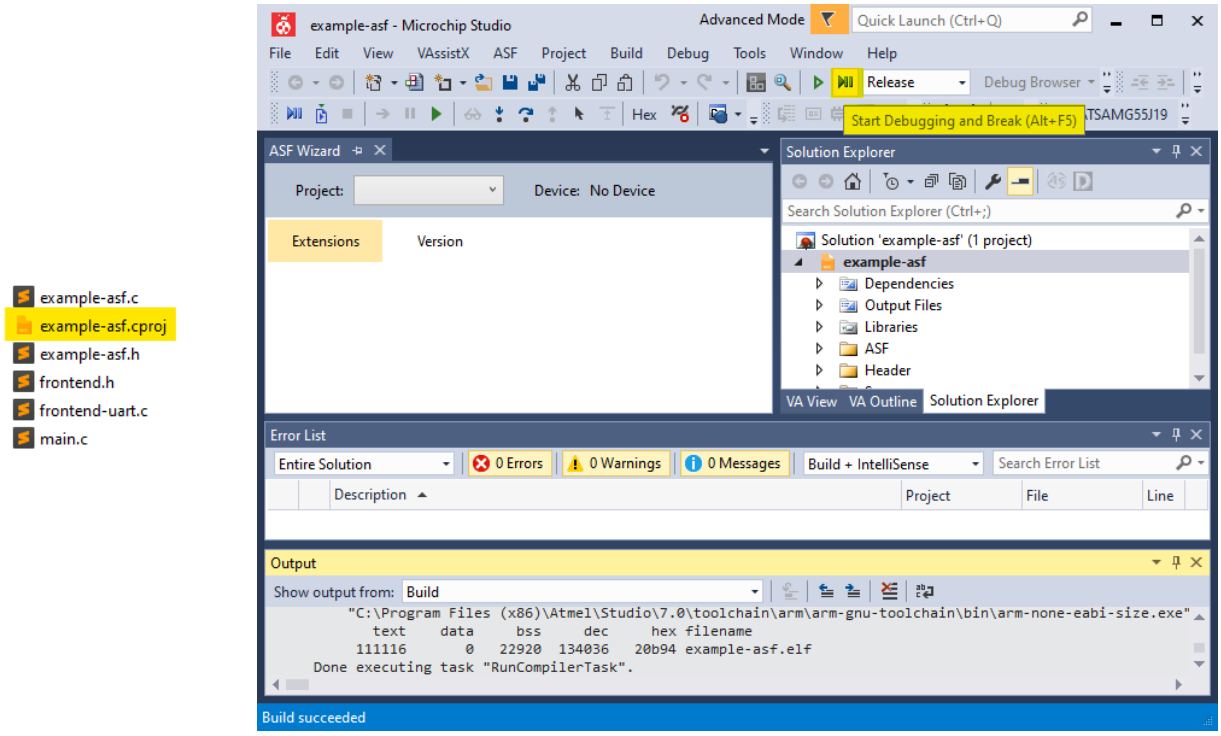



**Figure 7. Opening the example**

## 3.4. FLASHING BINARIES

Please use the *Device Programming* tool from Microchip Studion to flash binaries to your DK-42688-V. It is available from the “Tools” tab:

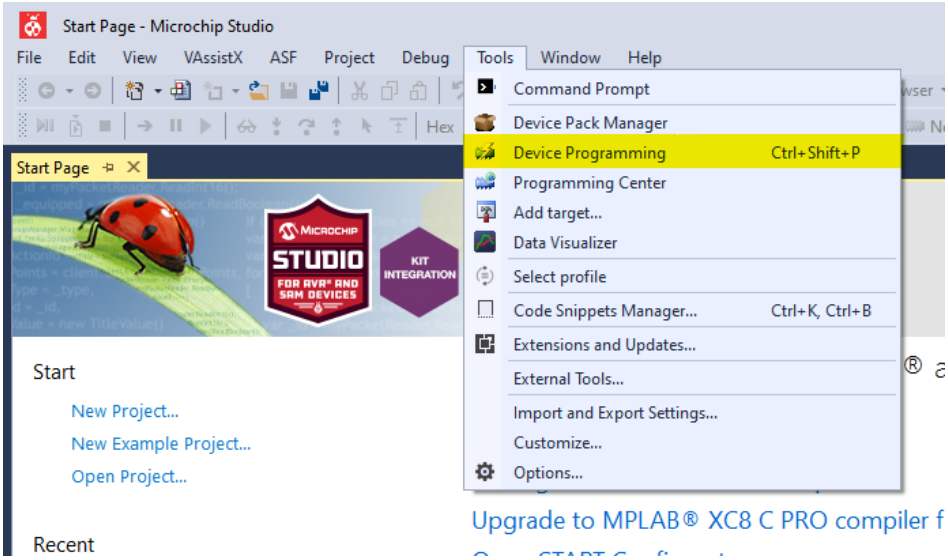



**Figure 8. Device programming**

- Select *EDBG* and *ATSAMG55J19* in the device field. Then hit *Apply*.
- On the left side, select *Memories* tab, then select *Erase Chip* then hit *Erase now*.

- Select the binary you want to flash and then hit *Program* then *Verify*.

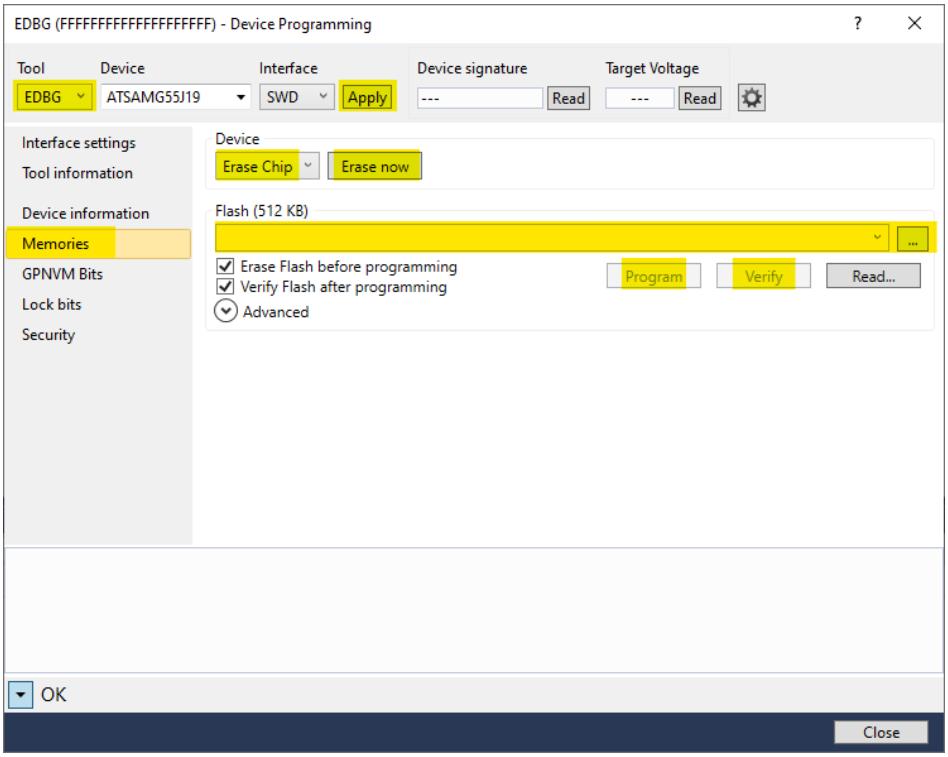



**Figure 6. Device programming continued**

## 3.5. ADVANCED SENSOR FUSION LIBRARY

### 3.5.1. Highlighted features

The Advanced Sensor Fusion library is a TDK InvenSense Prioprietary ICM-42688-V sensor algorithm library. The main features included in the library are:

- **Fast Cal Accel and Mag:** Calibrate accel and mag offsets with a limited amount of rotations.
- **Fast No Motion:** Calibrate gyro offset on continuous gestures (such as device is on the head), the algorithm can track the Gyro's offset, even during small gestures.

This algorithm can operate in two modes:

- High Accuracy Mode: This mode focuses on providing high orientation accuracy for applications that do not involve fast translational shakes or rigorous movement. Example applications include Head Motion detection for VR, AR and TWS, and Robotics.
- Fast Motion Mode: This mode focuses on fast motion tracking for applications that involve fast motion and translational shakes. Example applications include HHD controllers, Gaming and wearables.

### 3.5.2. Supported compiler

The Advanced Sensor Fusion library is provided in this package built for CM4-FPU with Microchip toolchain built with the following flags:

| | |
|---|---|
| **Compiler flags** | -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpu=fpv4-sp-d16 -fdata-sections -ffunction-sections |
| **Linker flags** | -Wl,--gc-sections |

The library is also available for:

- IAR 7.70
- Keil uVision 5.28 using arm6 (cclang)
- GCC 9.2-2019-q4

#### 3.5.2.1. Flags for IAR 7.70

| | | COMPILER FLAGS | LINKER FLAGS |
|---|---|---|---|
| **IAR** 7.70 | **CM4 FPU** | `--cpu=Cortex-M4F --fpu=VFPv4_sp --D__inline=inline --dlib_config \"${_IAR_HOME}/arm/INC/c/DLib_Config_Normal.h\" -D IAR_COMPILER --silent --endian=little --char_is_signed -e --diag_suppress Pa050,Go004,Go005 -DNDEBUG -Oh` | `--entry __iar_program_start` |
| | **CM3** | `--cpu=Cortex-M3 --D__inline=inline --dlib_config \"${_IAR_HOME}/arm/INC/c/DLib_Config_Normal.h\" -D IAR_COMPILER --silent --endian=little --char_is_signed -e --diag_suppress Pa050,Go004,Go005 -DNDEBUG -Oh` | `--entry __iar_program_start` |
| | **CM0** | `--cpu=Cortex-M0 --D__inline=inline --dlib_config \"${_IAR_HOME}/arm/INC/c/DLib_Config_Normal.h\" -D IAR_COMPILER --silent --endian=little --char_is_signed -e --diag_suppress Pa050,Go004,Go005 -DNDEBUG -Oh` | `--entry __iar_program_start` |

**Table 1. Flags for IAR 7.70**

#### 3.5.2.2. Flags for Keil uVision 5.28 using arm6 (cclang)

| | | COMPILER FLAGS | LINKER FLAGS |
|---|---|---|---|
| **Keil uVision** 5.28 using arm6 (cclang) | **CM4 FPU** | `--target=arm-arm-none-eabi -mcpu=cortex-m4+fp16 -mthumb -mfpu=fpv4-sp-d16 -mlittle-endian --cpu=Cortex-M4.fp --fpu=fpv4-sp -xc -fsigned-char -std=c99 -ffunction-sections -O2 -DNDEBUG` | `--datacompressor=off` |
| | **CM3** | `--target=arm-arm-none-eabi -mcpu=cortex-m3 -mthumb -mfpu=softvfp -mlittle-endian --cpu=Cortex-M3 --fpu=softvfp -xc -fsigned-char -std=c99 -ffunction-sections -O2 -DNDEBUG` | `--datacompressor=off` |
| | **CM0** | `--target=arm-arm-none-eabi -mcpu=cortex-m0 -mthumb -mfpu=softvfp -mlittle-endian --cpu=Cortex-M0 --fpu=softvfp -xc -fsigned-char -std=c99 -ffunction-sections -O2 -DNDEBUG` | `--datacompressor=off` |

**Table 2. Flags for Keil uVision 5.28 using arm6 (cclang)**

#### 3.5.2.3. Flags for GCC 9.2-2019-q4

| | | COMPILER FLAGS | LINKER FLAGS |
|---|---|---|---|
| **GCC** 9.2-2019-q4 | **CM4 FPU** | `-march=armv7e-m -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpu=fpv4-sp-d16 -fdata-sections -ffunction-sections -fsigned-char -fomit-frame-pointer -falign-functions=16 -fno-common -Wall -Wextra -std=c99 -O2 -DNDEBUG -finline-functions --specs=nosys.specs` | `-Wl,--gc-sections` |
| | **CM3** | `-march=armv7-m -mcpu=cortex-m3 -mthumb -fdata-sections -ffunction-sections -fsigned-char -fomit-frame-pointer -falign-functions=16 -fno-common -Wall -Wextra -std=c99 -O2 -DNDEBUG -finline-functions --specs=nosys.specs` | `-Wl,--gc-sections` |
| | **CM0** | `-march=armv6-m -mcpu=cortex-m0 -mthumb -fdata-sections -ffunction-sections -fsigned-char -fomit-frame-pointer -falign-functions=16 -fno-common -Wall -Wextra -std=c99 -O2 -DNDEBUG -finline-functions --specs=nosys.specs` | `-Wl,--gc-sections` |

**Table 3. Flags for GCC 9.2-2019-q4**

### 3.5.3. MCU specification requirements

The metrics below are computed from the library provided on the software package, built with Microchip toolchain compiler for a CM4-FPU.

| | |
|---|---|
| Code size – RAM | 2.9 KB |
| Code size – Flash | 28 KB |

The MIPS measurement is done as followed:

- Measure the duration of the algorithm processing for each new IMU sample and each new mag sample and average this value over 1 minute (dimu and dmag)
- Multiply this duration by the number of samples received during 1 second (typically 1024 for IMU and 100 for mag)
- The measurement is made on ATSAMG55J19 CM4-FPU platform running at 120 MHz. Typical DMIPS for CM4-FPU is 1.27 DMIPS/MHz. So total DMIPS available for this platform is 120*1.27=152.4 DMIPS. Multiply the total duration by this value to get the DMIPS usage of the algorithm.

$$DMIPS_{VR\_three\_dof} = \left(d_{IMU,s} \times ODR_{IMU} + d_{Mag,s} \times ODR_{Mag}\right) \times 120 \times 1.27$$

| | |
|---|---|
| $d_{imu}$ | 159 us (max = 357 µs) |
| $d_{mag}$ | 126 us (max = 315 µs) |
| **DMIPS for Advanced Sensor Fusion**<br>IMU at 1 KHz, Mag at 100 Hz | **26.73 DMIPS** |
| **DMIPS for Advanced Sensor Fusion**<br>IMU at 100 Hz, Mag at 100 Hz | **4.39 DMIPS** |

# 4 EXAMPLE APPLICATIONS

All these examples use the EDB USB port to flash and debug. To read traces and send commands to the firmware, use the FTDI USB port.

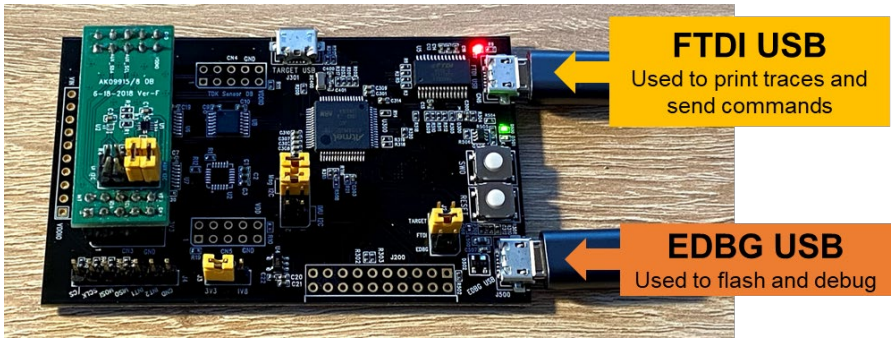



**Figure 7. Example applications**

The terminal emulator shall be connected to the *USB Serial Port* in Device Manager. Please use the following settings:

| | |
|---|---|
| **SPEED** | 921600 bauds |
| **DATA BITS** | 8 |
| **STOP BITS** | 1 |
| **PARITY** | None |
| **FLOW CONTROL** | None |

## 4.1. ASF EXAMPLE

### 4.1.1. Overview

This example demonstrates how to use the Advanced Sensor Fusion algorithm, showcasing how to configure it and how to process the output of the library.

The software architecture is depicted in Figure 8.

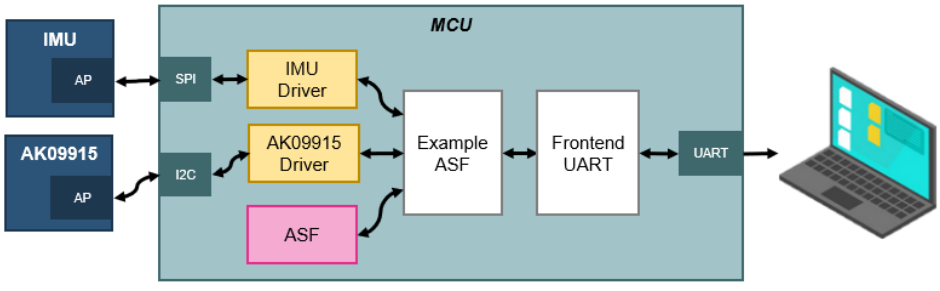



**Figure 8. Software architecture**

At startup, the firmware will start by configuring the IMU and mag. It will then proceed with the algorithm configuration and will first look in the flash memory if previously computed offsets exist. If so, it will load them in the algorithm to speed up calibration. Otherwise, it will just use value of 0 for the configuration. Once done, the IMU will start providing data, which will be processed by the algorithm. If the outputs of the algorithm show that all sensors (accel, gyro, and mag) are fully calibrated, meaning all accuracy flags are set to 3, it will save these offsets if flash for the next execution. The saving process will occur only once per run. Then data are sent to the frontend module where it will be printed on the terminal.

The state machine of the example is described in Figure 9.

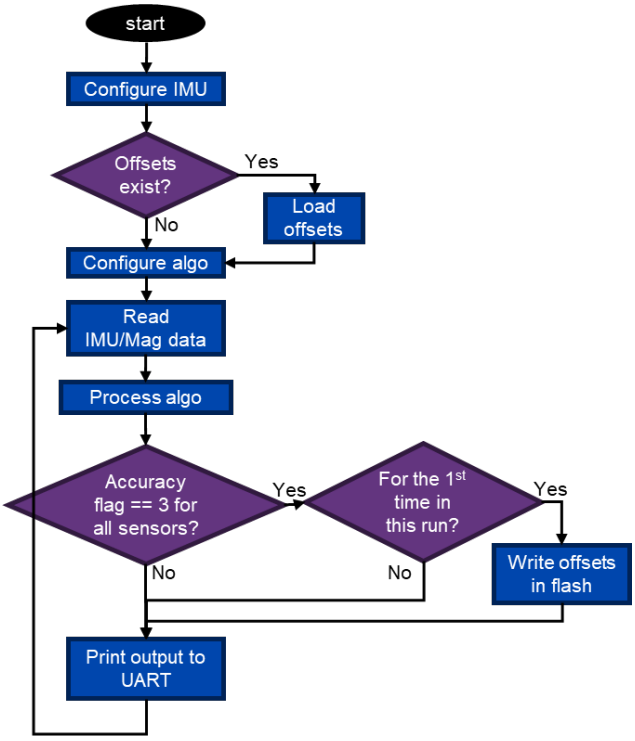



**Figure 9. State of the machine example**

### 4.1.2. UART output

Data are printed on the terminal as follow:

```
[I] ###################
[I] #   Example ASF   #
[I] ###################
[I] Initializing ICM device...
[I] OK
[I] Initializing algorithms...
[I]    Biases loaded from flash:
[I]     - Accel: [-0.003143 -0.003052 0.018402]g
[I]     - Gyro:  [1.159393 0.613205 -0.405075]dps
[I]     - Mag:   [20.451660 -14.824707 -10.238770]uT
[I] OK
[I] Configuring ICM device...
[I] OK
[I] Initializing Mag device...
[I] OK
```

```
[I] ##########################
[I] #   Help - Example ASF   #
[I] ##########################
[I]   'i' : print input data (raw accel, raw gyro and raw mag)
[I]   'a' : print accel data
[I]   'g' : print gyro data
[I]   'm' : print mag data
[I]   'q' : print quaternion data (9axis fusion)
[I]   'e' : print Euler angles for quaternion
[I]   'p' : print predictive quaternion data (predicted 9axis fusion)
[I]   'E' : print Euler angles for predictive quaternion
[I]   'r' : reset biases and accuracies for accel, gyro and mag
[I]   'f' : toggle fast-mode (data printed every 20 ms or every 1 s)
[I]   'h' : print this helper
[I] Start processing
[I] 2390373406: OUTPUT Acc=[-0.018, 0.000, 1.004]g Accuracy=3
[I] 2390373406: OUTPUT Gyr=[0.004, 0.002, 0.043]dps Accuracy=3 Temp=27.69 C
[I] 2390372612: OUTPUT Mag=[-22.702, -9.324, -12.410]uT MagAccuracy=1
[I] 2390373406: OUTPUT Euler9Axis=[110.97, -0.02, 1.00]deg 9AxisAccuracy=[46.480515]deg
[I] 2390373406: OUTPUT EulerPredQuat=[110.97, -0.02, 1.00]deg 9AxisAccuracy=[46.480515]deg
```

### 4.1.3. Supported commands

Sending characters to the UART allows you to send command to the firmware, to hide or display the various output of the algorithm, or to execute a specific action.

| | |
|---|---|
| **h** | Print help screen:<br>`[I] ##########################`<br>`[I] #   Help - Example ASF   #`<br>`[I] ##########################`<br>`[I]   'i' : print input data (raw accel, raw gyro and raw mag)`<br>`[I]   'a' : print accel data`<br>`[I]   'g' : print gyro data`<br>`[I]   'm' : print mag data`<br>`[I]   'q' : print quaternion data (9axis fusion)`<br>`[I]   'p' : print predictive quaternion data (predicted 9axis fusion)`<br>`[I]   'e' : print Euler angles`<br>`[I]   'r' : reset biases and accuracies for accel, gyro and mag`<br>`[I]   'f' : toggle fast-mode (data printed every 20 ms or every 1 s)`<br>`[I]   'h' : print this helper` |
| **l, a, g, m, q, p, e** | Hide or display corresponding output:<br>• 'i' for input data (raw accel, raw gyro and raw mag)<br>• 'a' for calibrated accel data<br>• 'g' for calibrated gyro data<br>• 'm' for calibrated mag data<br>• 'q' for quaternion data (9axis fusion)<br>• 'p' for predictive quaternion data (predicted 9axis fusion)<br>• 'e' for Euler angles |

| | |
|---|---|
| | At startup, only accel, gyro, mag, and Euler angles are displayed. |
| r | Reset biases and accuracies for accel, gyro, and mag |
| f | To toggle fast-mode: data are either printed every 1s (default) or 20 ms. Use for Visualizer app (see section below). |

### 4.1.4. Visualizer

To visualize the algorithm output, an executable is provided on the `demo\win\` folder. Ensure that your terminal is not connected and run the *example-asf-visualizer.exe* application.

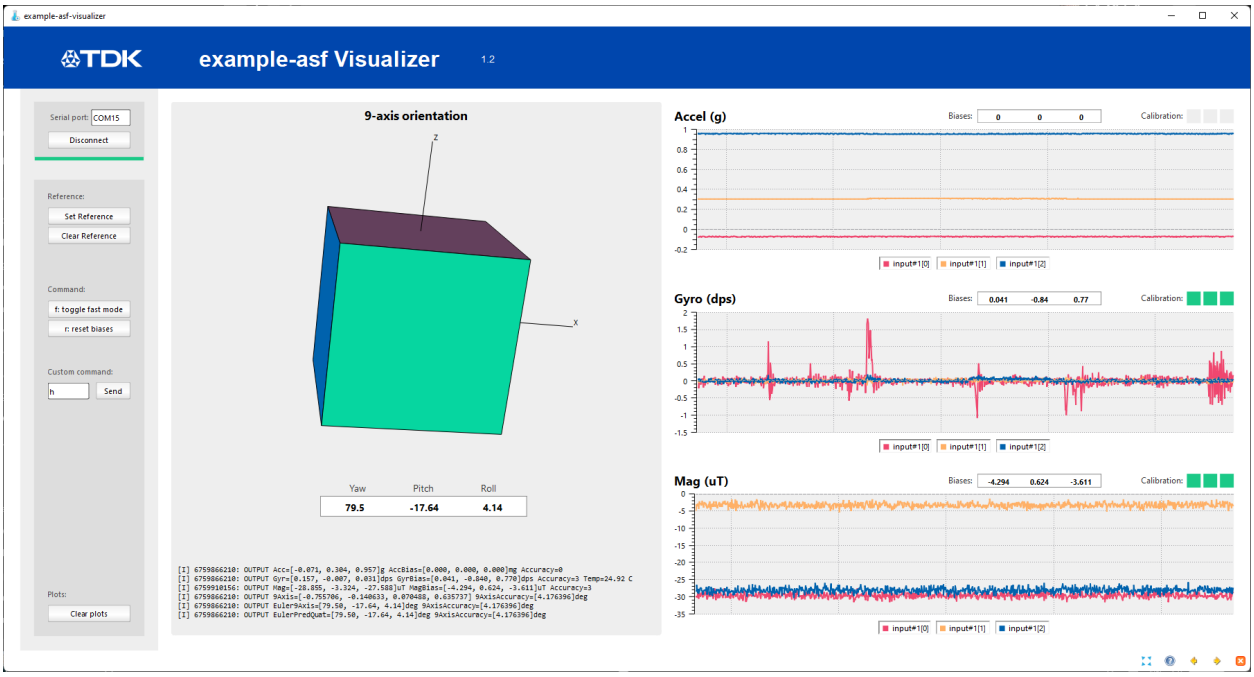
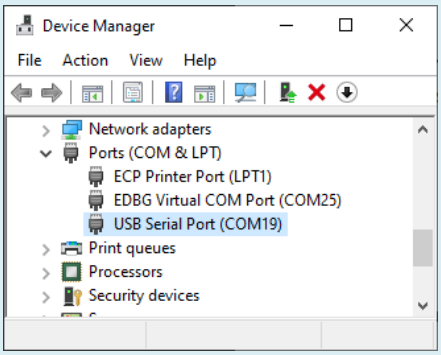



**Figure 10. Visualizer**

On the top-left corner, set the FTDI port number on the *Serial port* field then click *Connect*.

To identify the FTDI port number, open *Device Manager* and go to *Port (COM & LPT)* section. Look for *USB Serial Port* as highlighted below and use the associated COM port value in the Serial Port input box.




The application receives the firmware traces and parses them to display data. Features can be enabled or disabled by sending letters to the firmware. The *UART Input* item is used for that purpose.

The *Set Reference* and *Clear Reference* buttons can be used to align the orientation to an initial position. It will apply to both the quaternion (visualized by the cube) and the Euler angles.

Please note that the Euler angles represented on this application are calculated from the quaternion and can slightly differ from the value reported by the firmware.

On the top-right corner of the plots (accel, gyro, and mag), 3 squares represent the quality of the calibration. For a given sensor, the calibration is considered as completed when all 3 squares are green.

### 4.1.5. Available configurations

Available configurations are shown in this section and can be found in *example-asf.h* file.

**Algo mode**

The algo mode allows you to choose whether the algorithm will be optimized for **High Accuracy** or for **Fast Motion**.

Please refer to the following define:

```
/*
 * Configure the algorithm mode
 * Possible values:
 * - INVN_ALGO_ASF_HIGH_ACCURACY
 * - INVN_ALGO_ASF_FAST_MOTION
 */
#define ALGO_MODE INVN_ALGO_ASF_HIGH_ACCURACY
```

| | |
|---|---|
| **Use mag** | In case you don't want to use magnetometer, it can be disabled by setting the following define to 0.<br>`/*`<br>` * Set this define to 0 to disable mag support`<br>` * Recommended value: 1`<br>` */`<br>`#define USE_MAG 1` |
| **IMU ODR** | The accelerometer and gyroscope ODR are set by default to 1 KHz. You can change the frequency by modifying the following define. It will apply to both accel and gyro.<br>`/*`<br>` * Accelerometer and gyroscope frequencies.`<br>` * Recommended value: ICM426XX_GYRO_CONFIG0_ODR_1_KHZ (1000 Hz)`<br>` * Possible values:`<br>` * - ICM426XX_GYRO_CONFIG0_ODR_1_KHZ  (1000 Hz)`<br>` * - ICM426XX_GYRO_CONFIG0_ODR_500_HZ (500 Hz)`<br>` * - ICM426XX_GYRO_CONFIG0_ODR_200_HZ (200 Hz)`<br>` * - ICM426XX_GYRO_CONFIG0_ODR_100_HZ (100 Hz)`<br>` */`<br>`#define ICM_FREQ ICM426XX_GYRO_CONFIG0_ODR_1_KHZ` |
| **Mag ODR** | The magnetometer ODR must be 10000us (100Hz). |
| **FSR** | In this example, the IMU is configured in high resolution mode, allowing to have data over 20 bits. This mode enforces FSR to be **16 G for accelerometer** and **2000 dps for gyroscope**. |
| **SPI or $I^2C$** | By default, SPI is selected for the IMU to MCU communication. This can be changed with the following define. Please refer to the hardware section for proper jumper configuration.<br>`/*`<br>` * Select communication link between Smartmotion and ICM426xx`<br>` * Possible values:`<br>` * - ICM426XX_UI_SPI4`<br>` * - ICM426XX_UI_I2C`<br>` * Warning: With I2C interface, due to bandwidth limitation, maximum ODR is 500 Hz.`<br>` */`<br>`#define SERIF_TYPE ICM426XX_UI_SPI4`<br>Warning: With I2C interface, due to bandwidth limitation, maximum ODR is 500 Hz. |

**Table 4. Available configurations**

## 4.2. SANITY EXAMPLE

### 4.2.1. Overview

This application allows to check if the library integration is correct once ported to a new target. It will replay a test vector and compare the output of the library with the gold from the test vector.

The ICM driver needs to be ported and working for the test to pass. If the result is Success, it means the integration is correct, otherwise the library is not properly ported. Basically, this this project can be run during first steps of integration before running ASF example on a new target.

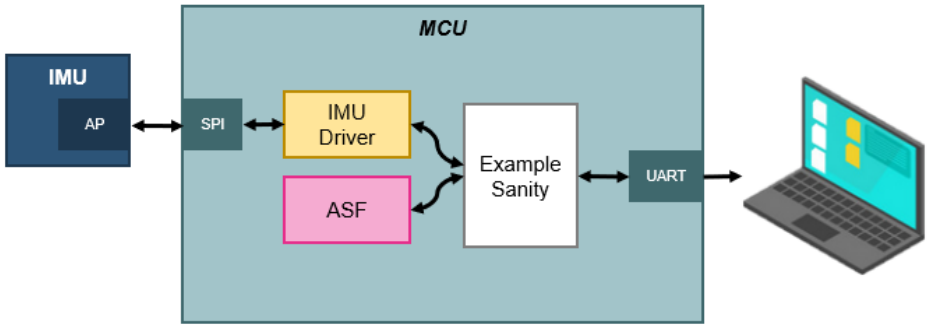



**Figure 11. Sanity example**

### 4.2.2. UART output

Data are printed on the terminal as follow:

```
[I] ########################
[I] #   ASF Sanity Test    #
[I] ########################
[I] Initializing ICM device...
[V] Initialize Icm426xx
[V] Check Icm426xx whoami value
[I] OK
[I] Initializing algorithms...
[I] OK
[I] Start processing
[I]
[I]    > Test case: out_invn_asf_1
[I]    > SUCCESS
[I]
[I] [SUCCESS] ASF sanity test is passing
```

## 4.3. SELF-TEST EXAMPLE

### 4.3.1. Overview

This application demonstrates how to execute the self-test. The self-test is a factory test that tests the MEMS structure and its functionality. Please read the datasheet for more details. In the software project example, the self-test is executed and the accelerometer, gyroscope offsets are then extracted from the procedure and printed on the terminal.

The board needs to be static during the execution of the self-test example.

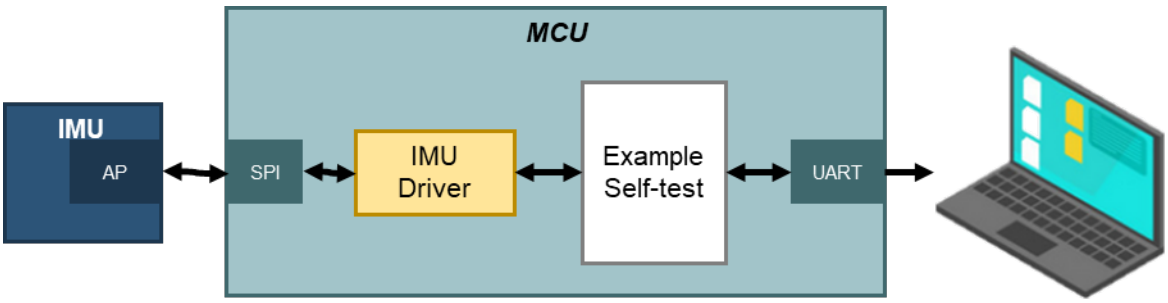



**Figure 12. Self-test example**

### 4.3.2. UART output

Data are printed on the terminal as follow:

```
[I] ######################
[I] #  Example SelfTest  #
[I] ######################
[I] Initializing ICM device...
[I] OK
[I] Check Icm426xx whoami value...
[I] OK
[I] Start Selftest...
[I] Gyro Selftest PASS
[I] Accel Selftest PASS
[I] Getting ST bias...
[I] GYR LN bias (dps): x=-0.167847, y=0.114441, z=0.228882
[I] ACC LN bias (g): x=-0.010498, y=0.003662, z=0.015259
```

# 5 REVISION HISTORY

| REVISION DATE | REVISION | DESCRIPTION |
|---|---|---|
| 01/29/2021 | 1.0 | Initial release |
| 03/24/2021 | 1.1 | Update examples names |
| 02/01/2023 | 1.2 | Update visuals and add warning for max ODR when in I2C |

©2021—2023 InvenSense. All rights reserved.