Software User Guide

SmartMotion ICM-42688-P

March 03, 2020

Version 2.0

TABLE OF CONTENTS

1. USEFUL LINKS

TDK-InvenSense website: http://www.InvenSense.com/

SmartMotion Platform: https://www.invensense.com/smartmotion-platform/

Microchip website: https://www.microchip.com/mplab/avr-support/atmel-studio-7

2. OVERVIEW

The purpose of this document is to give an overview of the ICM-426xx SmartMotion Development Kit that will allow users to run a predefined application based on motion sensors. This document may also serve as a quick start guide for the ICM-426xx package and its elements, including setup and how to use the sample applications provided.

2.1. INTRODUCTION

The ICM-426xx SmartMotion solution relies on the DK-426xx platform based on an ATSAMG55J19 microcontroller. The supported development tool is Atmel Studio. This solution aims at showcasing how to manage sensors and how to process algorithm, using a standalone microcontroller.

2.2. ICM-426XX BASICS

The ICM-426xx chip contains an Accelerometer and a Gyroscope accessible via the following interfaces:

- SPI 4 wires (full-duplex)
- SPI 3 wires (half-duplex)
- I2C

To support EIS (Electronical Image Stabilization), a pin can be dedicated to the FSYNC signal.

Two secondary SPI are dedicated to OIS (Optical Image Stabilization) support. When enabled, Gyroscope data can be retrieved from these interfaces on order to drive a camera module.

To help the synchronization of the timestamp, a pin can also be used to inject a clock signal (CLKIN).

⚠ The MCU embedded into the SmartMotion platform doesn’t support SPI 3 wires

3. Hardware Platform

The SmartMotion platform for ICM-426xx consists of the following components:

- SmartMotion Platform with Atmel SAMG55J19 MCU (DK-426xx)
- Optionally, sensor Daughter Boards for AK09915 magnetometer. *This daughter board is not included in the kit. Please contact TDK-Invensense sales if needed.*

3.1. SmartMotion Platform Connectivity

The SmartMotion platform is powered by its micro USB connectors (EDBG USB or FTDI USB). Please connect a micro-USB cable to your computer. In this demo software, both USB connector will be used, therefore, we recommend connecting both to your computer.

3.1.1. Jumpers configuration

J2 is used to select the source of the power. Using FTDI is recommended:

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

J1 is used to select the interface use to communicate with sensors. When using the ICM in I2C mode, it will use the “SDA” and “SCL” lines. When using the ICM in SPI mode, the auxiliary sensor (magnetometer) will use the “AUX-SCL” and “AUX-SDA” lines:

Config	ICM in SPI	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

** Using the magnetometer is optional. If not used, and if ICM is in SPI, then J1 can be completely open. Nevertheless, shorting AUX-SCL and AUX-SDA is harmless in this case.*

3.1.2. Daughter boards

The magnetometer daughter board is not included in the kit. Please contact TDK-Invensense sales if needed.

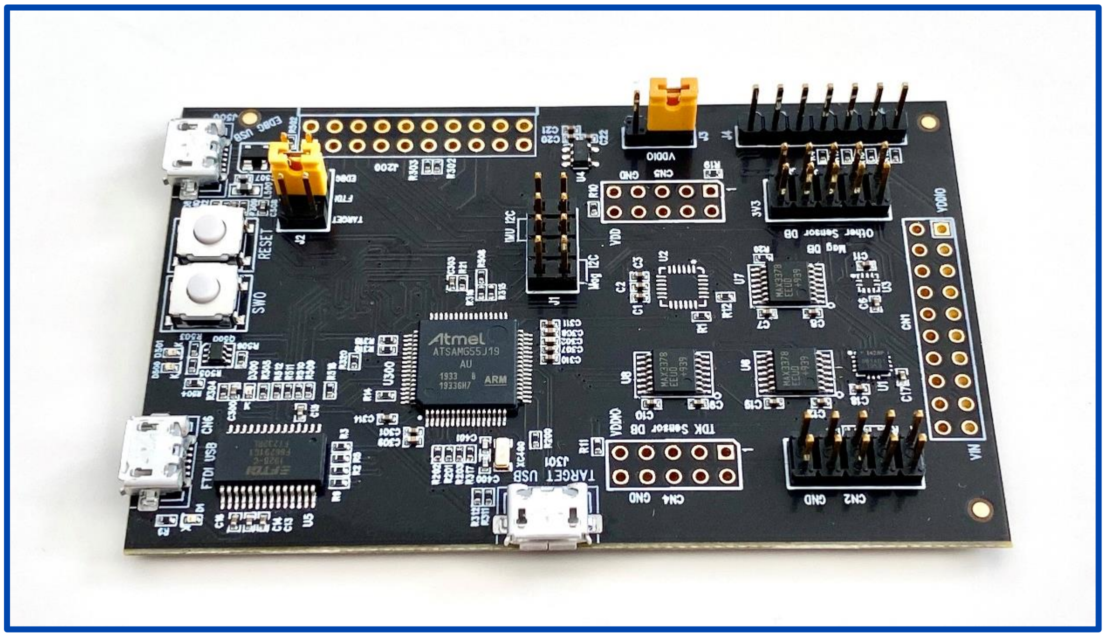
The magnetometer daughter boards must be connected on “Other Sensor DB”. The connector CN1 on DBs must be connected to CN3 on SmartMotion.

On the magnetometer daughter board, the jumper **J1** is used to select how it communicates with the MCU (aux interface or regular interface). When the ICM is in SPI mode, it uses the auxiliary interface, whereas when ICM is in I2C mode, it uses the regular interface:

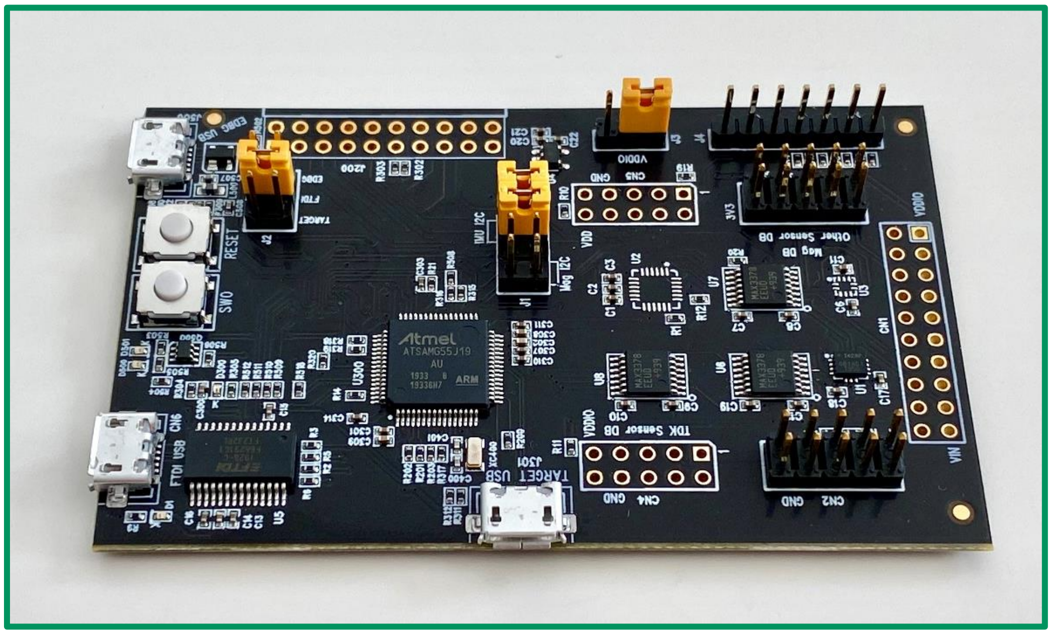
Pins	**J1 (AKM DB) ICM in SPI**	**J1 (AKM DB) ICM in I2C**
1 and 2	**Short**	Open
3 and 4	**Short**	Open
5 and 6	Open	**Short**
7 and 8	Open	**Short**

3.1.3. Illustration of the configuration for SPI and I2C

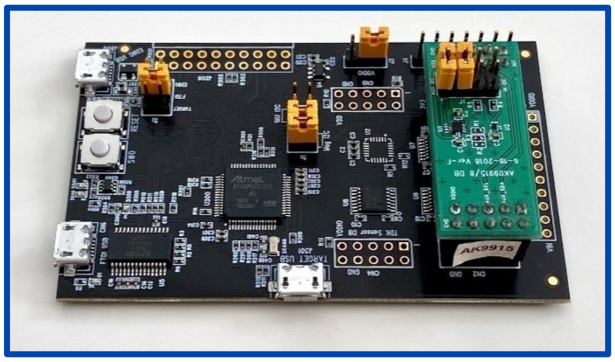
Platform configuration when communicating with ICM through **SPI**:

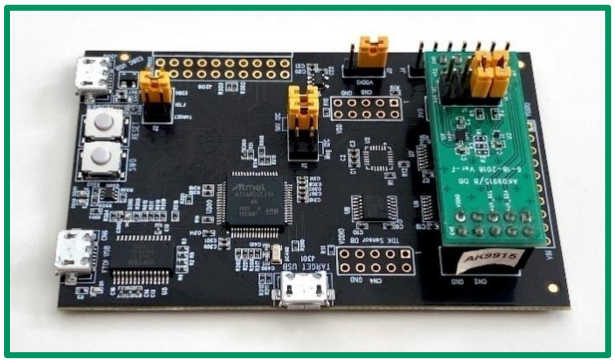


Platform configuration when communicating with ICM through **I2C**:



Platform configuration when magnetometer daughter board is connected:





3.1.4. Getting traces on the SmartMotion platform

First, please install FTDI drivers on your computer by following this link:
http://www.ftdichip.com/Drivers/VCP.htm

You can then retrieve the firmware traces by opening a terminal emulator and configure it as followed:

Speed	921600 bauds
Data bits	8
Stop bits	1
Parity	None
Flow control	None

4. SOFTWARE ENVIRONMENT

4.1. PREREQUISITE

To build and use standalone example provided in this package, please install the following software.

- Atmel Studio 7 (or above) IDE: https://www.microchip.com/mplab/avr-support/atmel-studio-7
- A RS232 terminal emulator
    - For instance, Putty: http://www.putty.org/

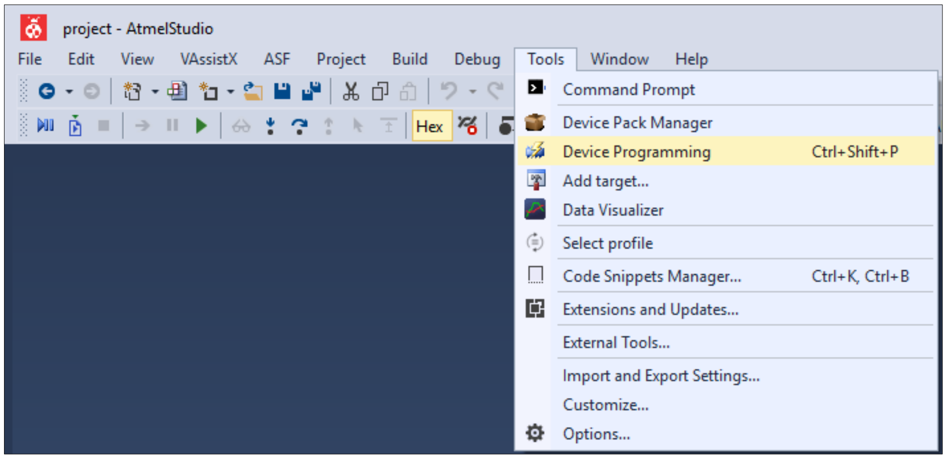
4.2. PACKAGE DESCRIPTION

This package is organized as followed:

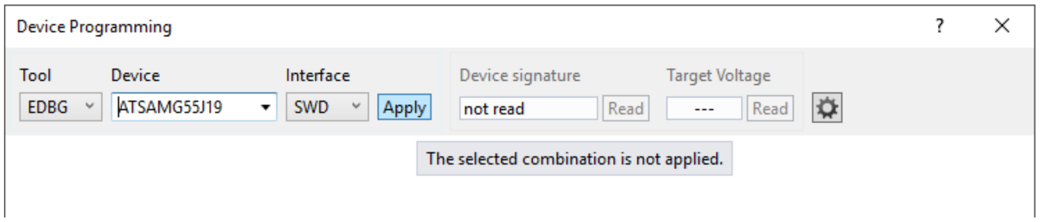
- ***demo/***
    - An executable to visualize the data from the example-algo
- ***doc/***
    - This documentation as well as the ICM driver documentation
- ***prebuilt/lib/***
    - The algorithm library built for Cortex-M4 with FPU
- ***release/bin/***
    - Firmware binaries of the examples
- ***sources/board-hal/***
    - Low-level driver for the SmartMotion platform used in examples
- ***sources/examples/***
    - Sample code for each example with a ready-to-use Atmel Studio project
- ***sources/Invn/***
    - Driver and various

4.3. FLASHING BINARIES

The tool "Device Programming" from Atmel Studio is used to flash the firmware. It is available from the "Tools" tab:

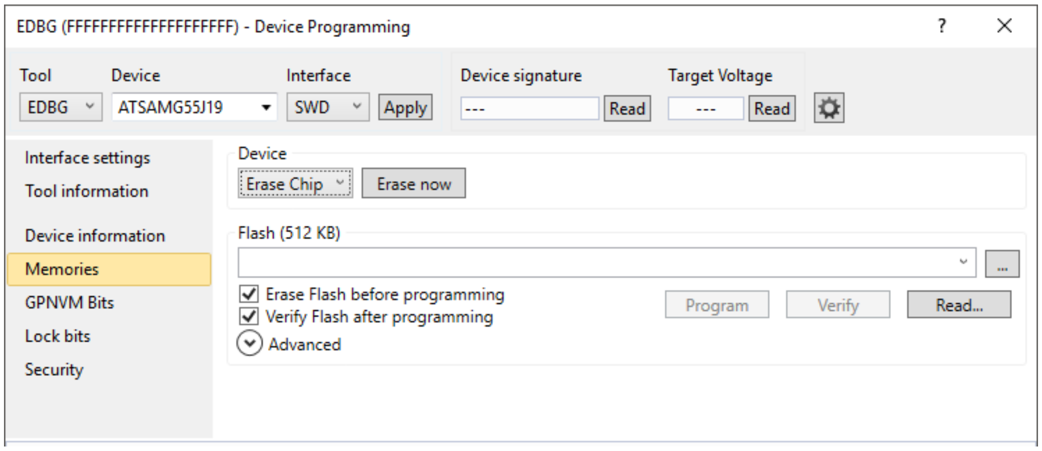

project - AtmelStudio
File Edit View VAssistX ASF Project Build Debug Tools Window Help
Hex
Command Prompt
Device Pack Manager
Device Programming Ctrl+Shift+P
Add target...
Data Visualizer
Select profile
Code Snippets Manager... Ctrl+K, Ctrl+B
Extensions and Updates...
External Tools...
Import and Export Settings...
Customize...
Options...

Select “EDBG” and “ATSAMG55J19” in the device field. Then click on “Apply”.



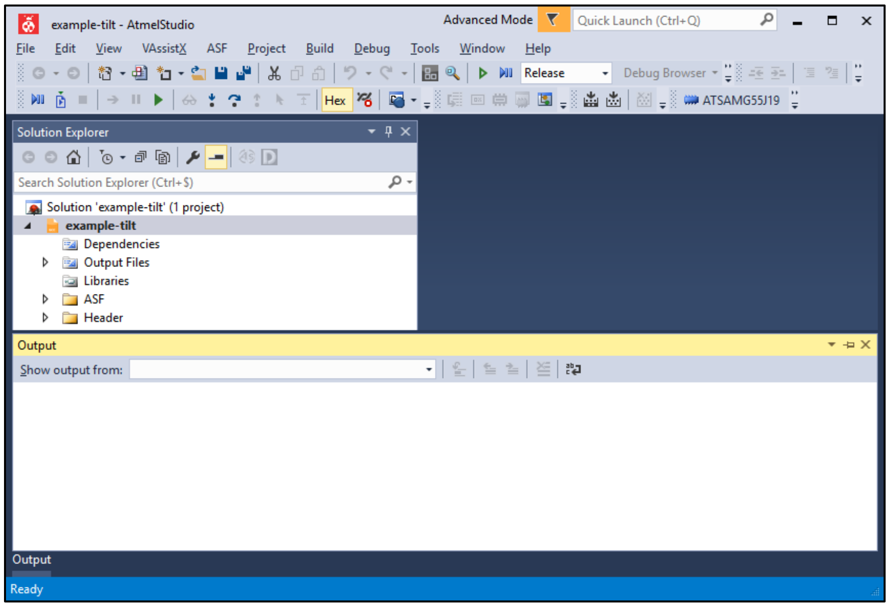
The selected combination is not applied.

Few tabs should show up on the left. On the “Memories” tab, select “Erase Chip” then click on “Erase now”. Select the binary you wish to flash and then click on “Program” then “Verify”.



4.4. OPENING EXAMPLES PROJECTS

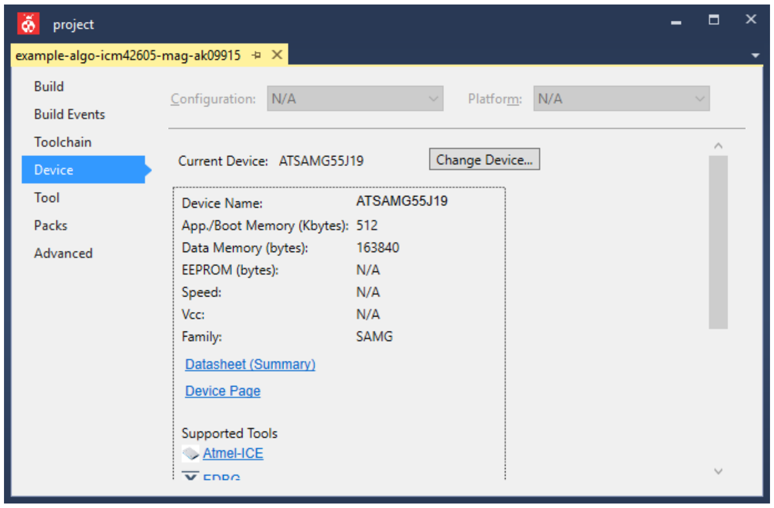
Open the desired **.cproj* file and AtmelStudio will open a new window.


example-tilt - AtmelStudio
Advanced Mode
Quick Launch (Ctrl+Q)
File Edit View VAssistX ASF Project Build Debug Tools Window Help
Release
Debug Browser
Hex
ATSAMG55J19
Solution Explorer
Search Solution Explorer (Ctrl+$)
Output
Show output from:
Output
Ready

Solution 'example-tilt' (1 project)
example-tilt
Dependencies
Output Files
Libraries
ASF
Header

On the left side you will find the workspace with all the sources. The project configuration can be reviewed by left clicking on “project options” in *Project>Properties.*

The device should be set to “*ATSAMG55J19”* and “*Use GDB”* should be selected under *Advanced* category.

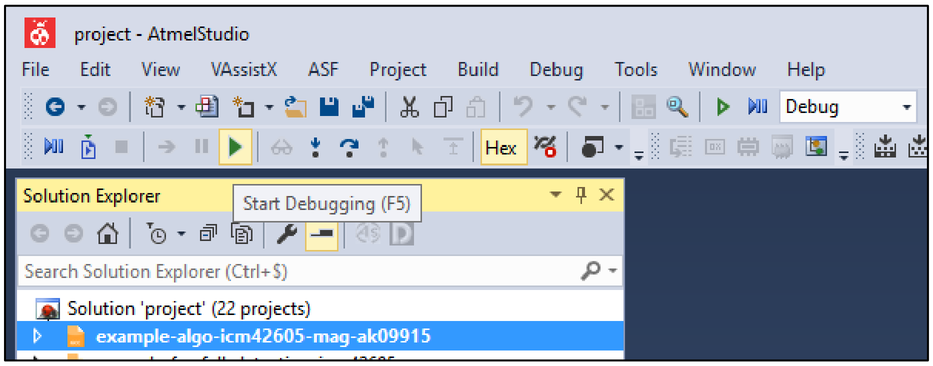


Configuration: N/A
Platform: N/A

Current Device: ATSAMG55J19

Device Name:	ATSAMG55J19
App./Boot Memory (Kbytes):	512
Data Memory (bytes):	163840
EEPROM (bytes):	N/A
Speed:	N/A
Vcc:	N/A
Family:	SAMG

To build and flash the project with a debug session, click on the “*Start Debugging”* button:


project - AtmelStudio
File Edit View VAssistX ASF Project Build Debug Tools Window Help
Debug
Hex
Solution Explorer
Start Debugging (F5)
Search Solution Explorer (Ctrl+$)

Solution 'project' (22 projects)
example-algo-icm42605-mag-ak09915

5. Example Applications

5.1. Commonalities

Once flashed, these applications will automatically start running when powering up the board. Data output will automatically be sent through the FTDI UART/USB connector.

5.1.1. Interfaces

By default, SPI is selected for the communication with the ICM. This can be changed by setting **`#define SERIF_TYPE`** to:

- **`ICM426XX_UI_SPI4`** for SPI4 communication
- **`ICM426XX_UI_I2C`** for I2C communication

Please refer to *3 Hardware Platform* section for proper wiring depending on your setup.

5.1.2. DMP configuration

Some examples will make use of the DMP unit. For those who do, you might find configuration option to change the "DMP ODR" or the "Power Save Mode".

5.1.2.1. DMP ODR

The DMP ODR corresponds to the frequency at which the DMP will process the accel data. The define used to configure this parameter is based on the "ICM426XX_APEX_CONFIG0_DMP_ODR_t" type.

It usually shows up as followed (example taken from Pedometer example):

```
/*
 * Pedometer frequency
 * Use type ICM426XX_APEX_CONFIG0_DMP_ODR_t to define pedometer frequency
 * These types are defined in Icm426xxDefs.h.
 *
 * \note The frequency modes to run the Pedometer are :
 * ICM426XX_APEX_CONFIG0_DMP_ODR_25Hz  (Low Power mode),
 * ICM426XX_APEX_CONFIG0_DMP_ODR_50Hz  (Normal mode)
 */
#define ICM_PEDOMETER_FREQUENCY_MODE ICM426XX_APEX_CONFIG0_DMP_ODR_25Hz
```

Please refer to the comment associated with the define for possible values (here, 25 Hz or 50 Hz).

Please also note that not all DMP related example exposes this configuration option.

5.1.2.2. Power Save Mode

The Power Save mode allows the DMP to sleep while no motion is detected. This feature relies on the WOM to wake up the DMP when motion is detected.

It usually shows up as followed (example taken from Pedometer example):

```
/*
 * Pedometer power save mode
 * Use type ICM426XX_APEX_CONFIG0_DMP_POWER_SAVE_t to define pedometer power save mode
 * These types are defined in Icm426xxDefs.h.
 */
#define ICM_PEDOMETER_POWER_SAVE_MODE ICM426XX_APEX_CONFIG0_DMP_POWER_SAVE_EN
```

Please note that not all DMP related example exposes this configuration option.

5.2. ALGO EXAMPLE

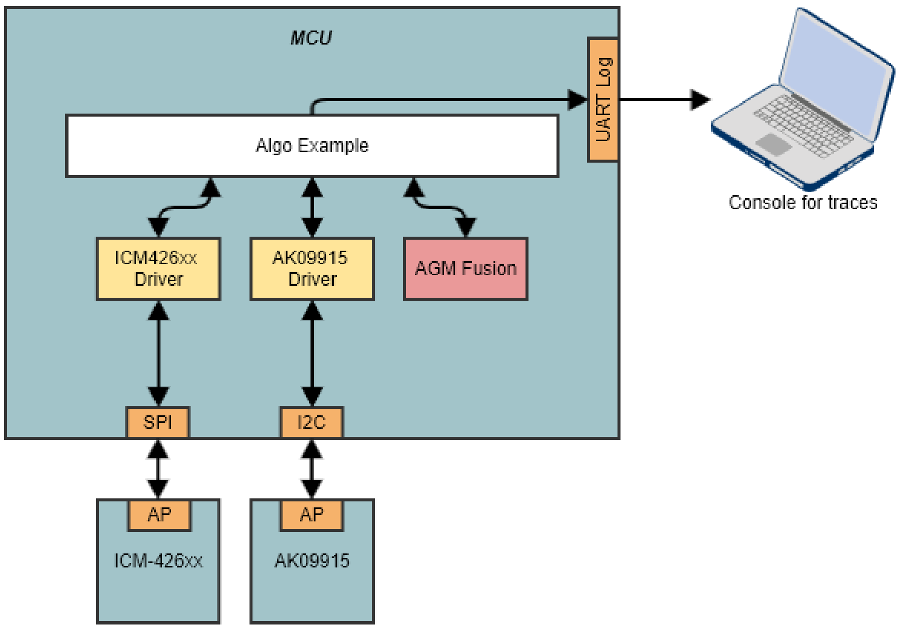
This project is an example of how to integrate the algorithms with ICM-426xx. It also demonstrates how to integrate the algorithms with a magnetometer (AK09915) that is connected to the main processor.

The external magnetometer is not included in the kit. Please contact TDK-InvenSense sales for example AK09915 daughter board for DK-42688-P if needed.

Algorithms outputs are printed on the terminal.

To get good accuracy level, it is recommended to proceed to basic calibration at startup:

- To calibrate the gyroscope, keep the board still for ~1 second (until cal gyro accuracy reaches 3)
- To calibrate the magnetometer, do 8-shape movement (the board should rotate around all 3 axis).
- To calibrate the accelerometer, keep the board still on 4 different faces for ~1 second.



Data are printed on the terminal as follow:

timestamp: INPUT/OUTPUT sensor_name1=[data], sensor_name2=[data]…, accuracy, temp

timestamp	Time in microsecond read from FIFO and synchronized with MCU clock
INPUT/OUTPUT	Input: data to be process by the AGM_FUSION algo.

	Output: data processed by the AGM_FUSION algo.
Sensor_name	String identifying the sensor data.
data	Output of the algorithm (depends on sensor)
accuracy	Indicator of the accuracy of the algorithm output. Can be: - None: No accuracy information - Accuracy flag: A value between 0 (very low accuracy) to 3 (very high accuracy) - Heading accuracy: Only applies to Rotation Vector and corresponds to the heading error estimate, in degrees.
temp	Temperature (LSB or °C)

Refer to this table for the corresponding data and accuracy per sensor:

Sensor	**INPUT/OUTPUT**	**Data type**

Raw accelerometer
Raw gyroscope
Raw temperature
(as read from sensor)
	INPUT	RAcc=[raw_acc_x, raw_acc _y, raw_acc _z]
		RGyr=[raw_gyr_x, raw_gyr_y, raw_gyr_z]
		Rtemp=[raw_temp]

Raw magnetometer *(as read from sensor)*	*INPUT*	RMag=[raw_mag_x, raw_mag_y, raw_mag_z]

Calibrated accelerometer,
Accelerometer biases
	OUTPUT	Acc=[cal_acc_x, cal_acc_y, cal_acc_z]
		AccBias= [acc_bia_x, acc_bias_y, acc_bias_z]
		Accuracy=[acc_accuracy]

Calibrated gyroscope
Gyroscope biases
Calibrated Temperature
	OUTPUT	Gyr=[cal_gyr_x, cal_gyr_y, cal_gyr_z]
		GyrBias=[gyr_bias_x, gyr_bias_y, gyr_bias_z]
		Temp=[cal_temp]
		Accuracy=[gyr_accuracy]

Calibrated magnetometer
Magnetometer biases
	OUTPUT	Mag=[cal_mag_x, cal_mag_y, cal_mag_z]
		MagBias=[mag _bias_x, mag _]bias_y, mag_bias_z]
		Accuracy=[mag_accuracy

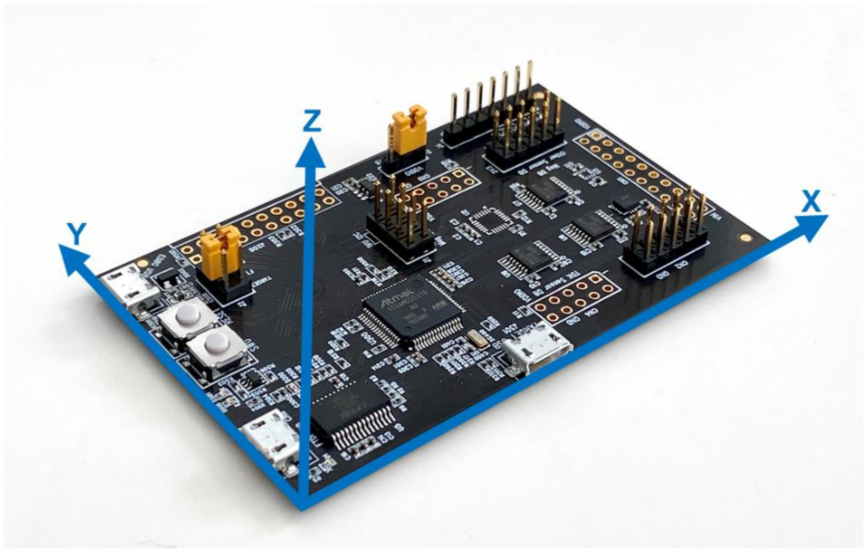
Rotation Vector	*OUTPUT*	9Axis=[rv_quat_w, rv_quat_x ,rv_ quat_y, rv_quat_z] 9AxisAccuracy=[rv_accuracy]
Euler angles (from rv quaternion)	*OUTPUT*	Euler9Axis=[raw, pitch, yaw] 9AxisAccuracy=[rv_accuracy]
Game rotation vector	*OUTPUT*	6Axis=[grv_quat_w, grv_quat_x , grv_ quat_y, grv_quat_z]
Euler angles (from grv quaternion)	*OUTPUT*	6Axis=[raw, pitch, yaw]

Gravity	*OUTPUT*	Gravity=[grav_x, grav_y, grav_z]
		Accuracy=[acc_accuracy]

Linear Acceleration	*OUTPUT*	LinearAcc =[linacc_x, linacc_y, linacc_z]

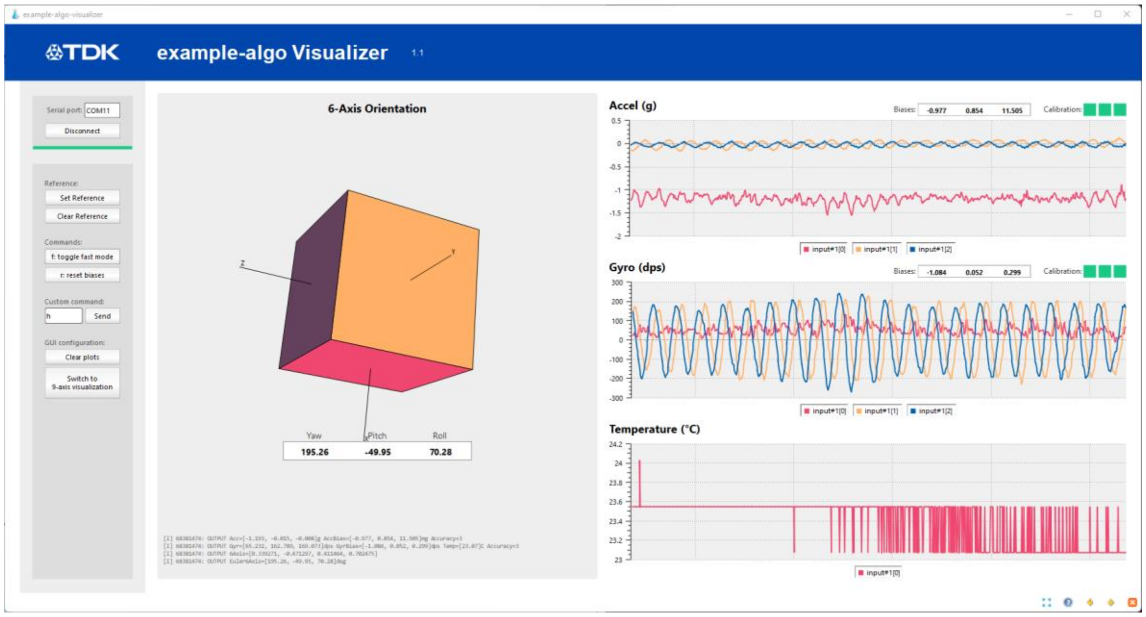
Mounting Matrix

Mounting matrix can be configured by setting the **`icm_mounting_matrix[9]`** array in *example-algo.c.* Default mounting matrix is set to identity which corresponds to the following reference frame:



Algo visualizer

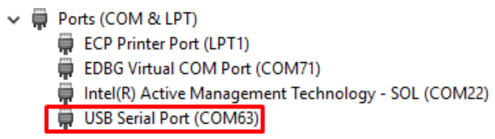
To visualize the algorithm output, an executable is provided. Ensure that your terminal is not connected and run the "*demo/example-vizualizer.exe*" application.



Yaw	Pitch	Roll
195.26	-49.95	70.28

```
[I] 68381474: OUTPUT Acc=[-1.195, -0.015, -0.008]g AccBias=[-0.977, 0.854, 11.505]mg Accuracy=3
[I] 68381474: OUTPUT Gyr=[65.232, 162.789, 169.073]dps GyrBias=[-1.088, 0.052, 0.209]dps Temp=[23.07]C Accuracy=3
[I] 68381474: OUTPUT 6Axis=[0.339271, -0.471297, 0.411464, 0.702475]
[I] 68381474: OUTPUT Euler6Axis=[195.26, -49.95, 70.28]deg
```

Once started, set the COM port of the FTDI on the “Serial port” field (top left corner) then click “Connect”. To find the FTDI port number on Windows, 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.



Ports (COM & LPT)
ECP Printer Port (LPT1)
EDBG Virtual COM Port (COM71)
Intel(R) Active Management Technology - SOL (COM22)
USB Serial Port (COM63)

The application will receive the firmware traces and parse them to display data. Each feature can be disabled or enabled 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 6-axis and 9-axis fusion to an initial position. It will apply to both the quaternion (visualized by the cube) and the Euler angles.

COM port output

```
[I] ###################
[I] #   Example AGM   #
[I] ###################
[I] Initializing ICM device...
[I] OK
[I] Initializing algorithm...
[I]    No bias values retrieved
[I] OK
[I] Configuring ICM device...
[I] OK
[I] Initializing Mag device...
[I] Initialize Ak09915
[I] OK
[I] ##########################
[I] #   Help - Example Algo  #
[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]     '9' : print rv quaternion data and eulers angles (9axis fusion)
[I]     '6' : print grv quaternion data and eulers angles (6axis fusion)
[I]     'G' : print gravity estimation in sensor frame
[I]     'l' : print linear acceleration estimation in sensor frame
[I]     'r' : reset biases and accuracies (will also reinit algorithm)
[I]     'f' : toggle fast-mode (data printed every 20 ms or every 1 s)
[I]     'h' : print this helper
[I] Start processing
[I] 15859525850: OUTPUT Acc=[-0.026, 0.006, 1.021]g AccBias=[0.000, 0.000, 0.000]mg Accuracy=0
[I]  15859525850:  OUTPUT  Gyr=[-0.046,  -0.027,  -0.032]dps  GyrBias=[0.290,  -0.156,  0.643]dps  Accuracy=3
Temp=19.20 C
[I] 15859524953: OUTPUT Mag=[23.999, -7.800, -17.849]uT MagBias=[0.000, 0.000, 0.000]uT Accuracy=0
[I]
[I] 15860502394: OUTPUT Acc=[-0.025, 0.006, 1.019]g AccBias=[0.000, 0.000, 0.000]mg Accuracy=0
[I] 15860502394: OUTPUT Gyr=[0.061, 0.019, 0.033]dps GyrBias=[0.244, -0.202, 0.638]dps Accuracy=3 Temp=19.20
C
[I] 15860495046: OUTPUT Mag=[25.199, -7.650, -16.799]uT MagBias=[0.000, 0.000, 0.000]uT Accuracy=0
[I]
[I] 15861478938: OUTPUT Acc=[-0.022, 0.005, 1.019]g AccBias=[0.000, 0.000, 0.000]mg Accuracy=0
[I]  15861478938:  OUTPUT  Gyr=[0.054,  0.027,  -0.095]dps  GyrBias=[0.251,  -0.210,  0.645]dps  Accuracy=3
Temp=19.20 C
```

```
[I] 15861475027: OUTPUT Mag=[23.549, -6.750, -17.849]uT MagBias=[0.000, 0.000, 0.000]uT Accuracy=0
```

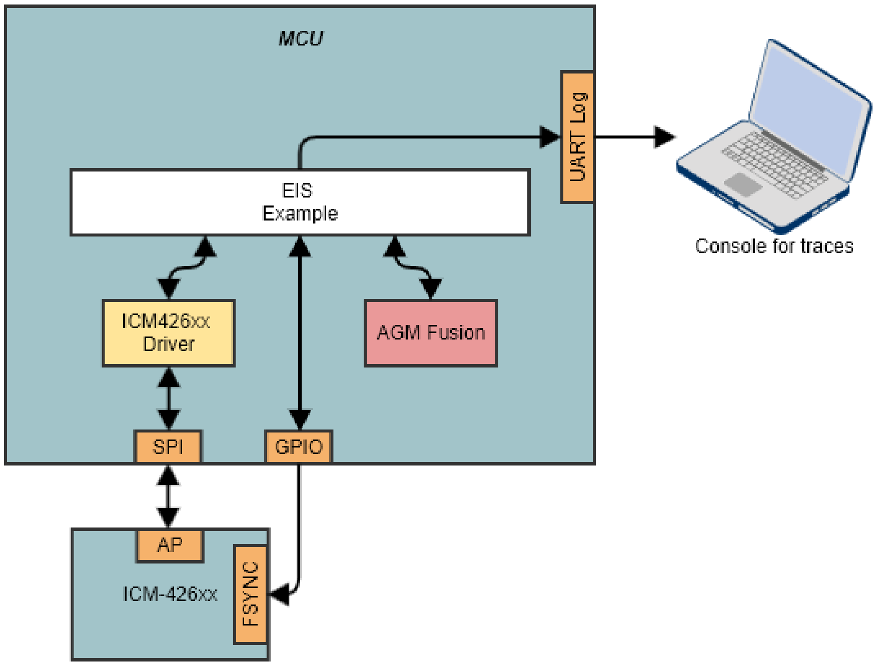
After calibration process done:

```
[I] 15951321015: OUTPUT Acc=[0.004, 1.003, -0.016]g AccBias=[-1.831, 0.763, 19.592]mg Accuracy=3
[I]  15951321015:  OUTPUT  Gyr=[-0.293,  -0.038,  0.208]dps  GyrBias=[0.232,  -0.206,  0.647]dps  Accuracy=3
Temp=19.69 C
[I] 15951318353: OUTPUT Mag=[26.949, -37.533, 1.000]uT MagBias=[3.050, 18.784, 1.400]uT Accuracy=3
[I]
[I] 15952297559: OUTPUT Acc=[0.003, 1.001, -0.020]g AccBias=[-1.831, 0.763, 19.592]mg Accuracy=3
[I]  15952297559:  OUTPUT  Gyr=[0.379,  0.145,  -0.158]dps  GyrBias=[0.232,  -0.206,  0.647]dps  Accuracy=3
Temp=19.69 C
[I] 15952298364: OUTPUT Mag=[28.749, -37.533, 1.750]uT MagBias=[3.050, 18.784, 1.400]uT Accuracy=3
```

5.3. EIS EXAMPLE

This application demonstrates how to get tagged gyroscope data from a FSYNC signal. This signal usually comes from a camera module that will be emulated here for the example. Here, the FSYNC signal is a 30Hz square signal and is connected to the ICM through the FSYNC pin.

Gyroscope data will be reported with a FSYNC flag being set for the event happening next to the FSYNC rising edge (otherwise, the flag will be 0). The FSYNC delay counter represents the time between the FSYNC rising edge and the sampling.



Data are printed on the terminal as follow:

timestamp : gyrcal_x, gyrcal_y, gyrcal_z accuracy_flag FSYNC event fsync_delay_cnt

timestamp	Time in microsecond read from FIFO and synchronized with MCU clock

gyrcal_x	Calibrated gyroscope X value
gyrcal_y	Calibrated gyroscope Y value
gyrcal_z	Calibrated gyroscope Z value
accuracy_flag	Accuracy level (from 0 to 3)
FSYNC event	Displayed only when the FSYNC is detected
fsync_delay_cnt	Delay in microsecond between the data and the FSYNC signal detected

COM port output

```
[I] ###################################
[I] #  ICM426xx streamlined example   #
[I] ###################################
[I] #######################
[I] #       EIS example   #
[I] #######################
[I] Initialize Icm426xx
[I] Check Icm426xx whoami value
[I] 468662: -0.183105, -0.549316, -0.244141  0
[I] 473678: -0.183105, -0.549316, -0.244141  0
[I] 478710: -0.183105, -0.549316, -0.244141  0
[I] 483735: -0.183105, -0.549316, -0.244141  0 FSYNC event 379
[I] 488759: -0.183105, -0.610352, -0.183105  0
[I] 493775: -0.244141, -0.610352, -0.244141  0
[I] 498807: -0.183105, -0.549316, -0.244141  0
[I] 503823: -0.122070, -0.549316, -0.244141  0
[I] 508855: -0.183105, -0.549316, -0.244141  0
[I] 513872: -0.183105, -0.549316, -0.244141  0
[I] 518905: -0.183105, -0.549316, -0.244141  0 FSYNC event 2208
[I] 523929: -0.183105, -0.549316, -0.244141  0
[I] 528961: -0.183105, -0.610352, -0.183105  0
```

We can see the accuracy level reached 3 quickly when the board is still on table.

```
[I] 2106565: 0.057220, -0.030518, 0.007629  3
[I] 2111581: -0.003815, -0.030518, 0.007629  3
[I] 2116614: 0.057220, -0.091553, 0.007629  3
[I] 2121630: 0.057220, -0.030518, 0.007629  3 FSYNC event 4983
[I] 2126654: 0.118256, -0.030518, -0.053406  3
[I] 2131670: 0.057220, -0.030518, -0.053406  3
[I] 2136702: 0.057220, -0.091553, 0.007629  3
[I] 2141718: 0.118256, -0.030518, 0.007629  3
[I] 2146750: 0.118256, -0.091553, 0.007629  3
[I] 2151775: 0.118256, -0.030518, 0.007629  3 FSYNC event 1812
```

We can also verify the FSYNC signal is detected at 30Hz (based on the last two FSYNC event detected above:

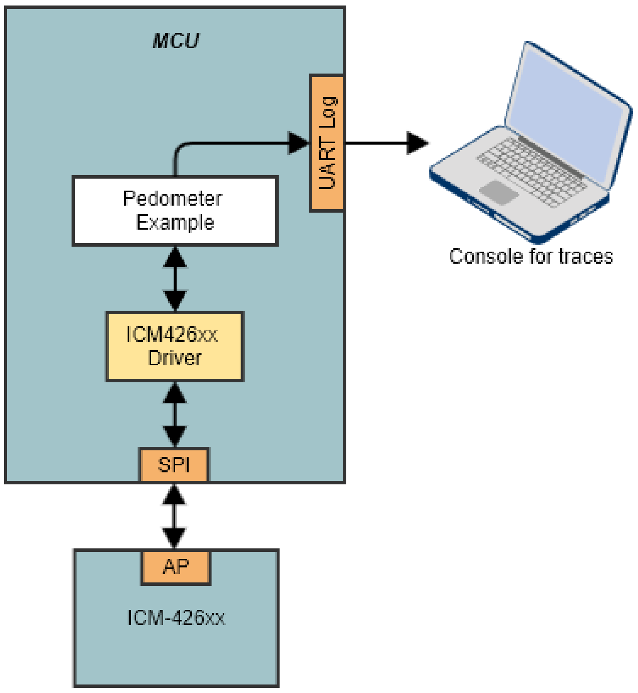
$$fsync\ delta\ time = (t_1 - delay_1) - (t_0 - delay_0)$$

$$fsync\ delta\ time = (2151775 - 1812) - (2121630 - 4983)$$

$$\boldsymbol{fsync\ delta\ time = 33316\ us = 30.0156\ Hz}$$

5.4. PEDOMETER EXAMPLE

The pedometer example demonstrates how to configure the hardware to detect steps and get the total count.



Seven parameters are configurable:

- DMP Frequency (refer to 5.1.2.1)
- DMP Power Save Mode (refer to 5.1.2.2)
- Pedometer Peak Threshold
- Minimum number of steps for step count detection
- Pedometer "non-walk" duration
- Minimum number of steps for step detector detection
- Pedometer sensitivity mode

```
/*
 * Peak threshold value to be considered as a valid step (mg)
 * Use type ICM426XX_APEX_CONFIG2_PEDO_AMP_TH_t to define the valid step peak threshold
 * These types are defined in Icm426xxDefs.h
 */
#define ICM_PEDOMETER_VALID_STEP_THRESHOLD ICM426XX_APEX_CONFIG2_PEDO_AMP_TH_2080374_MG

/*
 * Minimum number of steps that must be detected before the pedometer step count begins
incrementing
```

```
 */
#define ICM_PEDOMETER_STEP_COUNTER_THRESHOLD 5

/*
 * Duration of non-walk in number of samples to exit the current walk mode.
 * ICM_PEDOMETER_STEP_COUNTER_THRESHOLD number of steps above must again be detected before
step count starts to increase
 * Use type ICM426XX_APEX_CONFIG3_PEDO_SB_TIMER_TH_t to define the non-walk duration
 * These types are defined in Icm426xxDefs.h
 *
 * \note The recommended values according to the frequency mode selected are :
 * ICM426XX_APEX_CONFIG3_PEDO_SB_TIMER_TH_100_SAMPLES in Low Power mode
 * ICM426XX_APEX_CONFIG3_PEDO_SB_TIMER_TH_150_SAMPLES in Normal mode
 */
#define ICM_PEDOMETER_NON_WALK_DURATION ICM426XX_APEX_CONFIG3_PEDO_SB_TIMER_TH_100_SAMPLES

/*
 * Minimum number of low latency steps that must be detected before the pedometer step count
begins incrementing
 */
#define ICM_PEDOMETER_STEP_DETECTOR_THRESHOLD 2

/*
 * Sensitivity mode: Normal
 * Use type ICM426XX_APEX_CONFIG9_SENSITIVITY_MODE_t to define sensitivity mode
 * These types are defined in Icm426xxDefs.h.
 */
#define ICM_PEDOMETER_SENSITIVITY_MODE ICM426XX_APEX_CONFIG9_SENSITIVITY_MODE_NORMAL
```

COM port output

```
[I] ###################################
[I] #  ICM426xx streamlined example   #
[I] ###################################
[I] #########################
[I] #   Pedometer example   #
[I] #########################
[I] Initialize Icm426xx
[I] Check Icm426xx whoami value
```

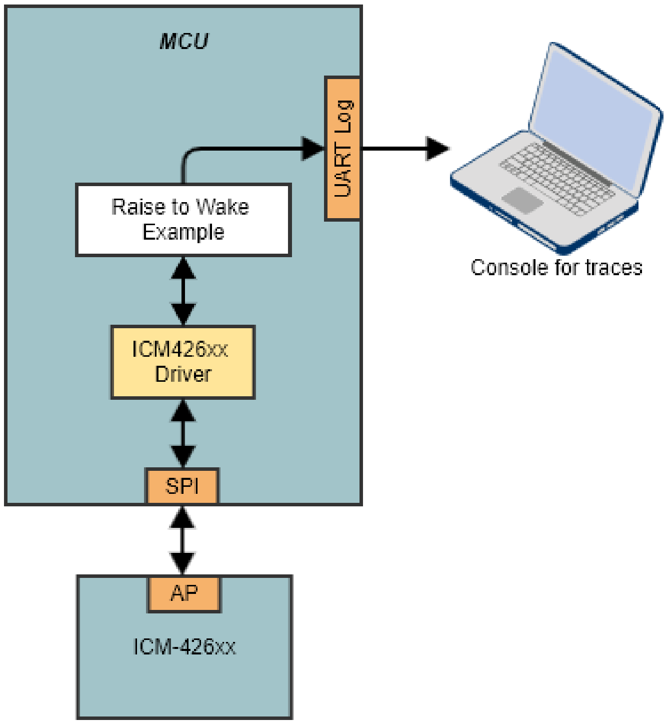
When steps are detected, the step count is printed with the cadency and the activity detected.

```
[I] ###################################
[I] #  ICM426xx streamlined example   #
[I] ###################################
[I] #########################
[I] #   Pedometer example   #
[I] #########################
[I] Initialize Icm426xx
[I] Check Icm426xx whoami value
[I] 7620377: 6 steps - cadence: 2.00 steps/sec
[I] 8023110: 7 steps - cadence: 2.17 steps/sec - WALK
[I] 8506386: 8 steps - cadence: 2.13 steps/sec - WALK
[I] 8949416: 9 steps - cadence: 2.17 steps/sec - WALK
[I] 9432715: 10 steps - cadence: 2.17 steps/sec - WALK
[I] 10439593: 11 steps - cadence: 1.56 steps/sec - WALK
[I] 12453228: 12 steps - cadence: 0.92 steps/sec - WALK
[I] 12896222: 13 steps - cadence: 1.15 steps/sec - WALK
[I] 13983515: 15 steps - cadence: 1.54 steps/sec - WALK
[I] 14748637: 17 steps - cadence: 2.13 steps/sec - WALK
```

```
[I] 15473575: 19 steps - cadence: 2.56 steps/sec - RUN
[I] 16117958: 21 steps - cadence: 2.94 steps/sec - RUN
[I] 16762305: 23 steps - cadence: 3.13 steps/sec - RUN
[I] 17446925: 25 steps - cadence: 3.03 steps/sec - RUN
[I] 18091251: 27 steps - cadence: 3.13 steps/sec - RUN
[I] 18775856: 29 steps - cadence: 3.03 steps/sec - RUN
[I] 19460432: 31 steps - cadence: 3.03 steps/sec - RUN
[I] 20104784: 33 steps - cadence: 3.13 steps/sec - RUN
[I] 20708859: 35 steps - cadence: 3.33 steps/sec - RUN
[I] 21433752: 37 steps - cadence: 2.94 steps/sec – RUN
```

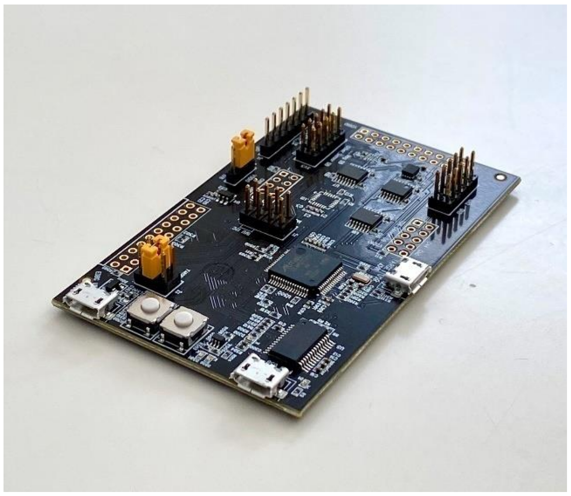
5.5. RAISE TO WAKE EXAMPLE

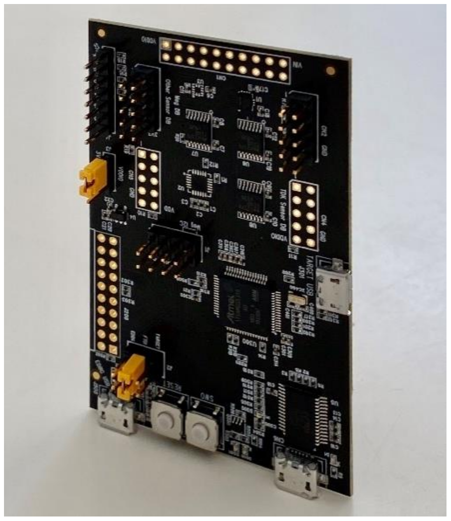
The raise to wake example demonstrates how to configure hardware to detect “wake” and “sleep” events.



Once the example is running, the user shall perform a “raise to wake” movement to detect it. A “sleep” event is detected if the user performs the “sleep” movement or if the sleep time out is reached.

The gestures to make a “raise to wake” or a “sleep” is a rotation around the Y-axis:


Initial position: board laying in the table.


Final position: board is facing the user. USB cable are connected at the bottom of the board.

Four configuration options are available:

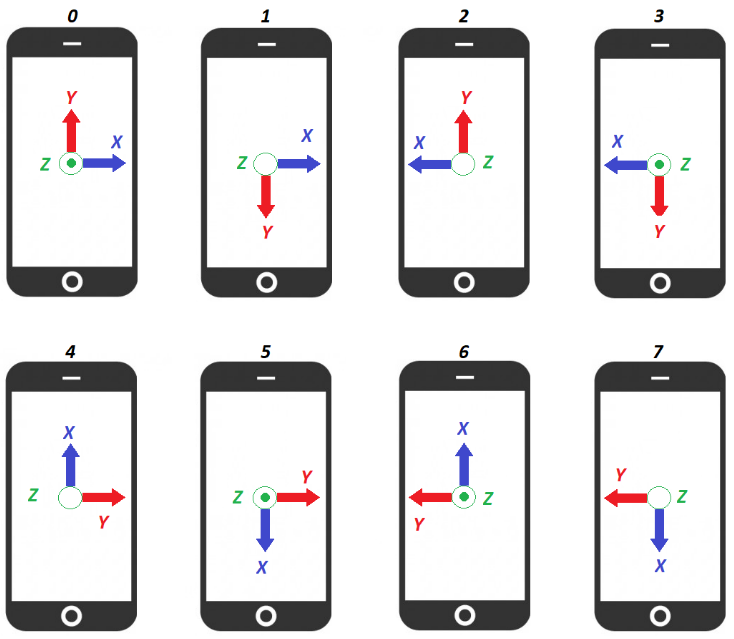
- DMP Power Save Mode (refer to 5.1.2.2)
- R2W Sleep Time-out
- R2W Mounting Matrix
- R2W Sleep delay

```
/*
 * Raise to wake sleep time
 * Use type ICM426XX_APEX_CONFIG4_R2W_SLEEP_TIME_OUT_t to define the time out for a sleep
detection.
 * Warning : the ICM426XX_APEX_CONFIG1_DMP_POWER_SAVE_TIME_t value must be superior to the
ICM426XX_APEX_CONFIG4_R2W_SLEEP_TIME_OUT_t value.
 * These types are defined in Icm426xxDefs.h.
 */
#define ICM_R2W_SLEEP_TIME ICM426XX_APEX_CONFIG4_R2W_SLEEP_TIME_OUT_6_4S

/*
 * Raise to wake mounting matrix
 * Use type ICM426XX_APEX_CONFIG5_R2W_MOUNTING_MATRIX_t to define mounting matrix, chip to
device frame.
 * These types are defined in Icm426xxDefs.h.
 */
#define ICM_R2W_MOUNTING_MATRIX ICM426XX_APEX_CONFIG5_R2W_MOUNTING_MATRIX_6

/*
 * Raise to wake sleep gesture delay
 * Use type ICM426XX_APEX_CONFIG6_R2W_SLEEP_GEST_DELAY_t to define the detection window for a
sleep gesture detection.
 * These types are defined in Icm426xxDefs.h.
 */
#define ICM_R2W_SLEEP_GEST_DELAY ICM426XX_APEX_CONFIG6_R2W_SLEEP_GEST_DELAY_0_96S
```

The following picture describes how to configure the mounting matrix so the “wake” event triggers when the user looks at the screen.



COM port output

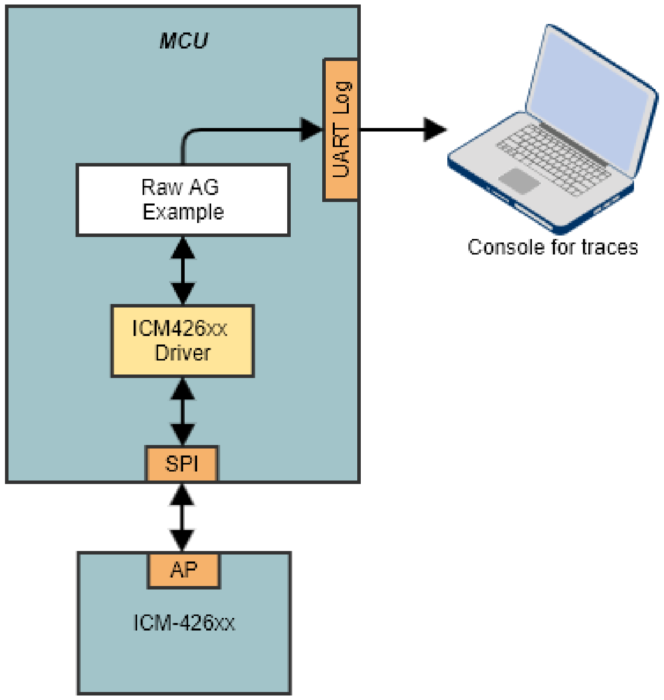
```
[I] ###################################
[I] #  ICM426xx streamlined example   #
[I] ###################################
[I] ###########################
[I] #  RAISE TO WAKE example  #
[I] ###########################
[I] Initialize Icm426xx
[I] Check Icm426xx whoami value
```

When an event is detected, the timestamp and an event notification are displayed:

```
[I] ###################################
[I] #  ICM426xx streamlined example   #
[I] ###################################
[I] ###########################
[I] #  RAISE TO WAKE example  #
[I] ###########################
[I] Initialize Icm426xx
[I] Check Icm426xx whoami value
[I] 1453048: SLEEP event detected
[I] 10963290: WAKE event detected
[I] 14093239: SLEEP event detected
[I] 15898965: WAKE event detected
```

5.6. RAW AG EXAMPLE

The Raw AG example uses UI data path from ICM426XX. It configures the full-scale range and the ODR to 1kHz for both accelerometer and gyroscope. It also enables and uses FIFO to store sensor data.



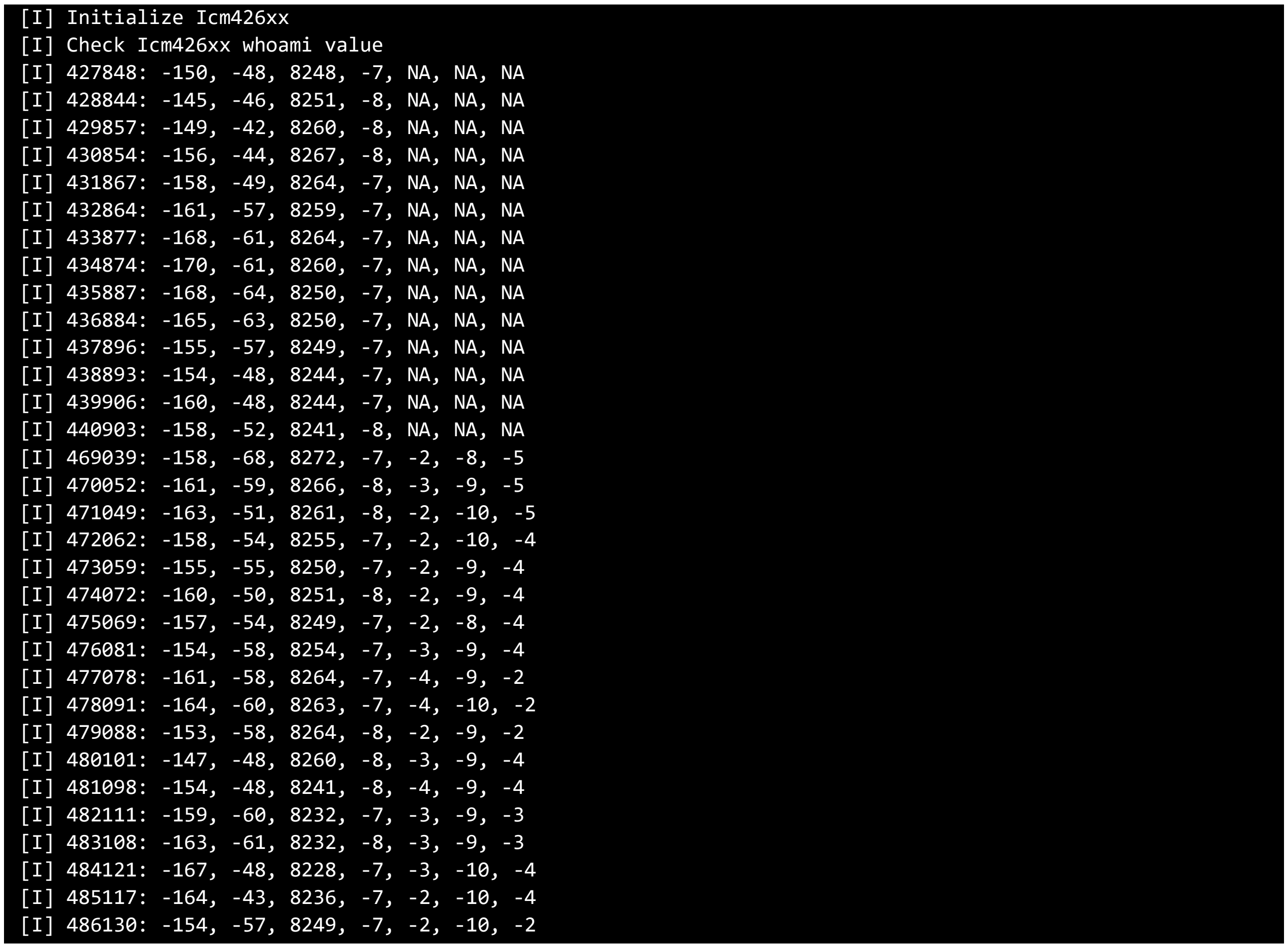
Data are printed on the terminal as follow:

timestamp : racc_x racc_y racc_z raw_temperature rgyr_x rgyr_ rgyr_z

timestamp	Time in microsecond read from FIFO and synchronized with MCU clock
racc_x	Raw accelerometer X value read from FIFO
racc_y	Raw accelerometer Y value read from FIFO
racc_z	Raw accelerometer Z value read from FIFO
raw_temperature	Raw temperature read from FIFO
rgyr_x	Raw gyroscope X value read from FIFO
rgyr_y	Raw gyroscope Y value read from FIFO
rgyr_z	Raw gyroscope Z value read from FIFO

COM port output:

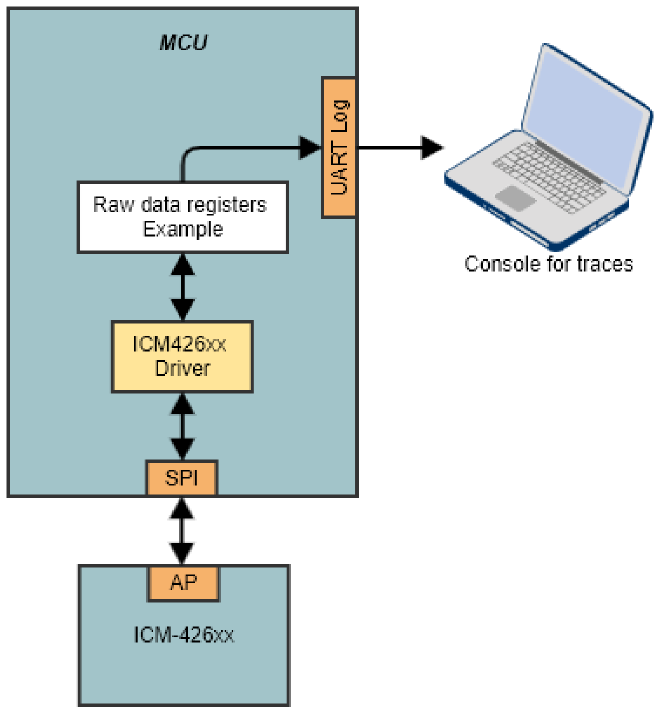
```
[I] ###################################
[I] #  ICM426xx streamlined example   #
[I] ###################################
[I] #######################
[I] #   RAW A+G example   #
[I] #######################
```



```
[I] Initialize Icm426xx
[I] Check Icm426xx whoami value
[I] 427848: -150, -48, 8248, -7, NA, NA, NA
[I] 428844: -145, -46, 8251, -8, NA, NA, NA
[I] 429857: -149, -42, 8260, -8, NA, NA, NA
[I] 430854: -156, -44, 8267, -8, NA, NA, NA
[I] 431867: -158, -49, 8264, -7, NA, NA, NA
[I] 432864: -161, -57, 8259, -7, NA, NA, NA
[I] 433877: -168, -61, 8264, -7, NA, NA, NA
[I] 434874: -170, -61, 8260, -7, NA, NA, NA
[I] 435887: -168, -64, 8250, -7, NA, NA, NA
[I] 436884: -165, -63, 8250, -7, NA, NA, NA
[I] 437896: -155, -57, 8249, -7, NA, NA, NA
[I] 438893: -154, -48, 8244, -7, NA, NA, NA
[I] 439906: -160, -48, 8244, -7, NA, NA, NA
[I] 440903: -158, -52, 8241, -8, NA, NA, NA
[I] 469039: -158, -68, 8272, -7, -2, -8, -5
[I] 470052: -161, -59, 8266, -8, -3, -9, -5
[I] 471049: -163, -51, 8261, -8, -2, -10, -5
[I] 472062: -158, -54, 8255, -7, -2, -10, -4
[I] 473059: -155, -55, 8250, -7, -2, -9, -4
[I] 474072: -160, -50, 8251, -8, -2, -9, -4
[I] 475069: -157, -54, 8249, -7, -2, -8, -4
[I] 476081: -154, -58, 8254, -7, -3, -9, -4
[I] 477078: -161, -58, 8264, -7, -4, -9, -2
[I] 478091: -164, -60, 8263, -7, -4, -10, -2
[I] 479088: -153, -58, 8264, -8, -2, -9, -2
[I] 480101: -147, -48, 8260, -8, -3, -9, -4
[I] 481098: -154, -48, 8241, -8, -4, -9, -4
[I] 482111: -159, -60, 8232, -7, -3, -9, -3
[I] 483108: -163, -61, 8232, -8, -3, -9, -3
[I] 484121: -167, -48, 8228, -7, -3, -10, -4
[I] 485117: -164, -43, 8236, -7, -2, -10, -4
[I] 486130: -154, -57, 8249, -7, -2, -10, -2
```

5.7. RAW DATA REGISTERS EXAMPLE

Accelerometer and gyroscope data are read from register using Data Ready interrupt. It configures the full-scale range and the ODR to 1kHz for both accelerometer and gyroscope. In this example, both accel and gyro data are printed at the fastest ODR.



Data are printed on the terminal as follow:

timestamp : racc_x racc_y racc_z raw_temperature rgyr_x rgyr_y rgyr_z

timestamp	Time in microsecond read from MCU clock
racc_x	Raw accelerometer X value read from register
racc_y	Raw accelerometer Y value read from register
racc_z	Raw accelerometer Z value read from register
raw_temperature	Raw temperature read from register on 16 bits (1/256 degC format)
rgyr_x	Raw gyroscope X value read from register
rgyr_y	Raw gyroscope Y value read from register
rgyr_z	Raw gyroscope Z value read from register

COM port output

```
[I] ###################################
[I] #  ICM426xx streamlined example   #
[I] ###################################
[I] #################################
[I] #   RAW DATA register example   #
[I] #################################
[I] Initialize Icm426xx
[I] Check Icm426xx whoami value
[I] 7162: -129, 55, 8170, -296, 0, -3, -1
[I] 8148: -138, 52, 8178, -316, 0, -2, -1
[I] 9134: -147, 50, 8183, -280, 0, -1, -1
[I] 10120: -145, 49, 8173, -294, -1, -2, -1
[I] 11107: -133, 44, 8167, -290, -2, -3, -2
[I] 12093: -133, 43, 8178, -312, -2, -2, -3
[I] 13079: -140, 52, 8189, -312, -1, -2, -3
[I] 14065: -141, 60, 8184, -322, -2, -2, -3
[I] 15051: -146, 65, 8172, -326, -2, -3, -2
[I] 16038: -153, 63, 8167, -288, 0, -2, -1
[I] 17024: -152, 49, 8164, -324, 1, -2, -1
[I] 18010: -142, 46, 8154, -306, -1, -3, -2
[I] 18997: -136, 59, 8145, -308, 0, -4, -3
[I] 19983: -133, 61, 8151, -304, 1, -4, -3
[I] 20969: -132, 55, 8160, -310, 1, -3, -3
[I] 21955: -133, 50, 8158, -312, -1, -3, -2
[I] 22942: -131, 52, 8163, -316, -2, -2, -2
[I] 23928: -140, 50, 8174, -306, -1, -2, -1
[I] 24914: -149, 39, 8171, -310, 0, -2, -1
[I] 25901: -140, 36, 8161, -302, -2, -2, -1
[I] 26887: -137, 48, 8162, -312, -2, -2, -1
[I] 27873: -143, 53, 8169, -290, -1, -2, -1
[I] 28860: -142, 49, 8161, -312, 0, -2, -2
[I] 29846: -138, 45, 8149, -324, 0, -2, -2
[I] 30832: -136, 42, 8157, -300, 0, -3, -2
[I] 31818: -136, 46, 8162, -292, 0, -3, -1
```

5.8. SANITY EXAMPLE

This example aims at verifying the integrity of the algorithm library. Its main purpose is to be ported to a custom platform to allow user to verify the library built with a different toolchain.

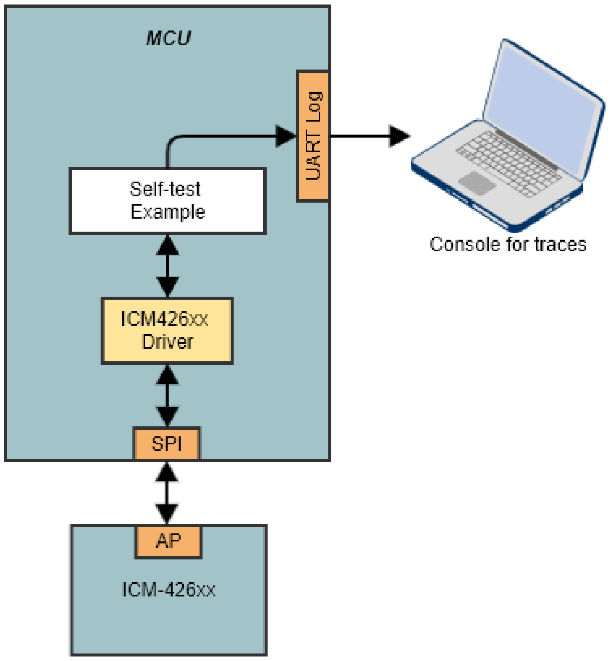
COM port output

```
[I] #############################
[I] #   AgmFusion 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_agm_1
[I]    > SUCCESS
[I]
[I] [SUCCESS] AgmFusion sanity test is passing
```

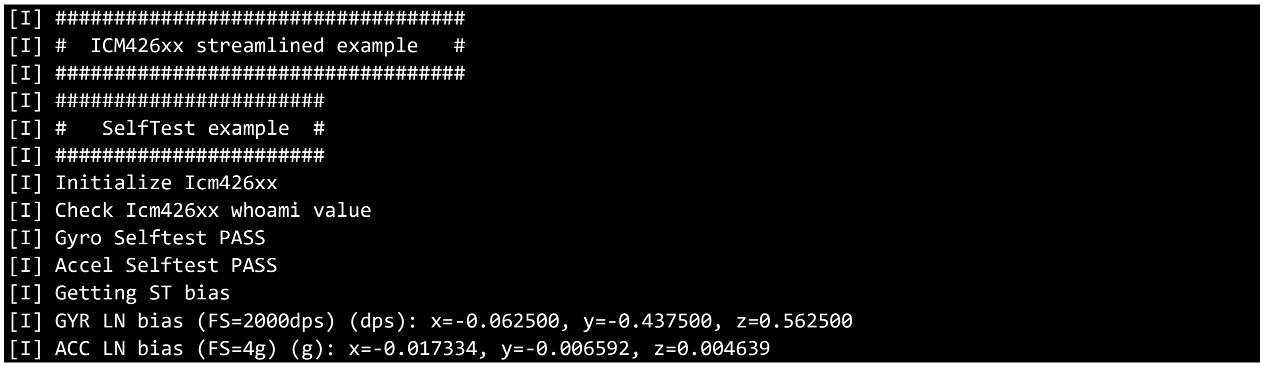
5.9. Self-test Example

This application demonstrates how to execute the Self-test. Accelerometer and 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.



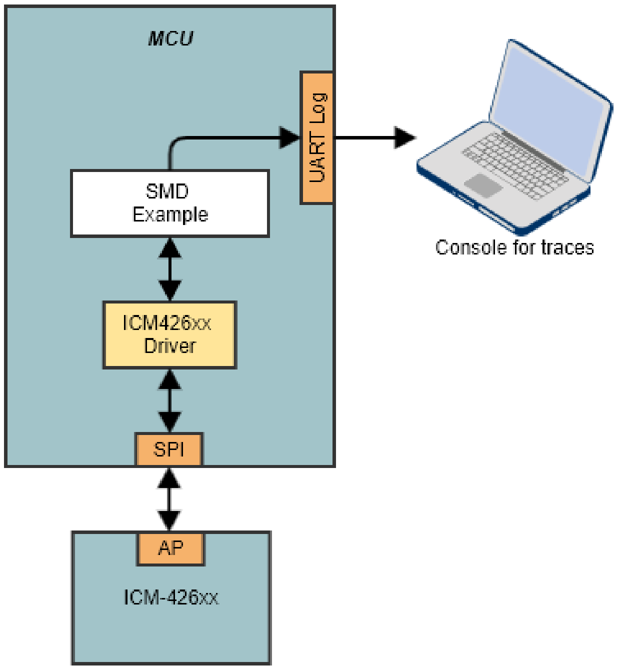
COM port output



```
[I] ###################################
[I] #  ICM426xx streamlined example   #
[I] ###################################
[I] #######################
[I] #   SelfTest example  #
[I] #######################
[I] Initialize Icm426xx
[I] Check Icm426xx whoami value
[I] Gyro Selftest PASS
[I] Accel Selftest PASS
[I] Getting ST bias
[I] GYR LN bias (FS=2000dps) (dps): x=-0.062500, y=-0.437500, z=0.562500
[I] ACC LN bias (FS=4g) (g): x=-0.017334, y=-0.006592, z=0.004639
```

5.10. SMD Example

The SMD example demonstrates how to configure hardware to detect SMD events.



Once the example is running, the user shall simulate a displacement to generate a SMD. A displacement is typically simulated by doing “steps” (moving the board up and down for ~8 sec).

COM port output

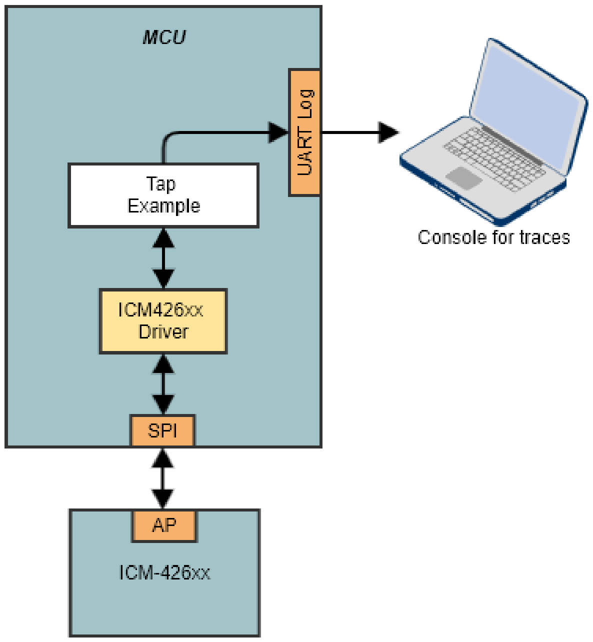
```
[I] ###################################
[I] #  ICM426xx streamlined example   #
[I] ###################################
[I] ###############
[I] #   TAP example   #
[I] ###############
[I] Initialize Icm426xx
[I] Check Icm426xx whoami value
```

When the SMD event is detected, the timestamp and an event notification are printed. Then the SMD is disabled (example has to be restart to detect another one).

```
[I] ###################################
[I] #  ICM426xx streamlined example   #
[I] ###################################
[I] #################
[I] #     SMD example   #
[I] #################
[I] Initialize Icm426xx
[I] Check Icm426xx whoami value
[I] 5291062: SMD detected, disabling it
```

5.11. Tap Example

The tap example demonstrates how to configure hardware to detect tap events.



Once example is running, user should try to make a tap or double tap on the board.

COM port output

```
[I] ###################################
[I] #  ICM426xx streamlined example   #
[I] ###################################
[I] ###################
[I] #   TAP example   #
[I] ###################
[I] Initialize Icm426xx
[I] Check Icm426xx whoami value
```

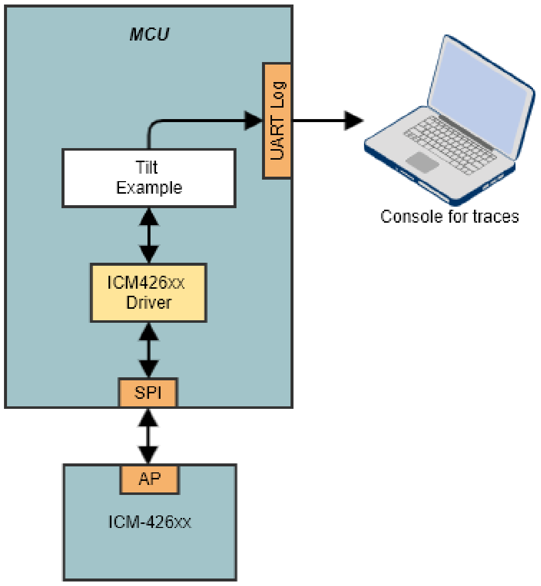
When a tap event is detected, the timestamp and an event notification are displayed:

```
[I] ###################################
[I] #  ICM426xx streamlined example   #
[I] ###################################
[I] ###################
[I] #   TAP example   #
[I] ###################
[I] Initialize Icm426xx
[I] Check Icm426xx whoami value
[I] 358959857: Single Tap detected on X-axis (positive direction)
[I] 854090832: Single Tap detected on X-axis (positive direction)
[I] 986873986: Single Tap detected on X-axis (positive direction)
[I] 988347459: Double Tap detected on X-axis (negative direction) with 464 ms between tap
```

```
[I] 989870255: Single Tap detected on X-axis (negative direction)
[I] 995898564: Double Tap detected on X-axis (positive direction) with 336 ms between tap
[I] 997528776: Double Tap detected on X-axis (positive direction) with 368 ms between tap
[I] 999140998: Double Tap detected on X-axis (positive direction) with 368 ms between tap
```

5.12. Tilt Example

The tilt example demonstrates how to configure and detect Tilt events.



Once the example is running, tilting the board with an angle of 30 degrees or more will generate a tilt. By default, a tilt event is generated when the position is held for 4 seconds.

Three configuration options are exposed for this example:

- DMP Frequency (refer to 5.1.2.1)
- DMP Power Save Mode (refer to 5.1.2.2)
- Tilt wait time

The configuration of the Tilt wait time can be changed with this define (from main.c):

```
/*
 * Tilt wait time
 * Use type ICM426XX_APEX_CONFIG4_TILT_WAIT_TIME_t to define the number of accelerometer
samples
 * to wait before triggering tilt event.
 * These types are defined in Icm426xxDefs.h.
 */
#define ICM_TILT_WAIT_TIME ICM426XX_APEX_CONFIG4_TILT_WAIT_TIME_4S
```

COM port output

```
[I] ###################################
[I] #  ICM426xx streamlined example   #
```

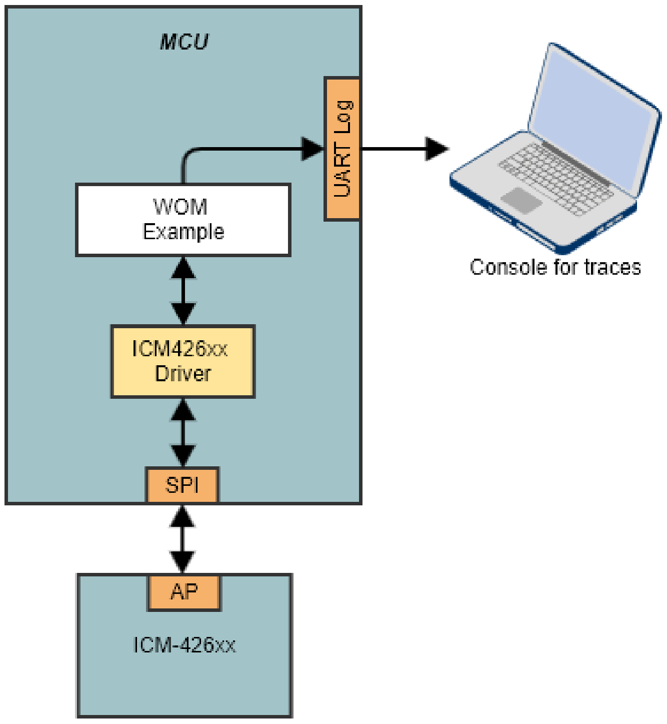
```
[I] ###################################
[I] ####################
[I] #   TILT example   #
[I] ####################
[I] Initialize Icm426xx
[I] Check Icm426xx whoami value
```

When a tilt event is detected, the timestamp and a tilt event notification are displayed:

```
[I] ###################################
[I] #  ICM426xx streamlined example   #
[I] ###################################
[I] #########################
[I] #      TILT example     #
[I] #########################
[I] Initialize Icm426xx
[I] Check Icm426xx whoami value
[I] 20988503: Tilt event detected
[I] 26505311: Tilt event detected
[I] 33390774: Tilt event detected
[I] 40114883: Tilt event detected
[I] 491206000: Tilt event detected
```

5.13. WOM EXAMPLE

The WOM example demonstrates how to collect WOM interrupt.



Once the example is running, moving the board will generate WOM events. The axis on which the WOM has been detected will then be printed on the terminal.

The default configuration will lead to WOM event being generated when a 200 mg acceleration is measured. This configuration can be modified with this define:

```
/* Initial WOM threshold to be applied to ICM in mg */
#define WOM_THRESHOLD_INITIAL_MG 200
/* WOM threshold to be applied to ICM, ranges from 1 to 255, in 4mg unit */
static uint8_t wom_threshold = WOM_THRESHOLD_INITIAL_MG/4;
```

COM port output

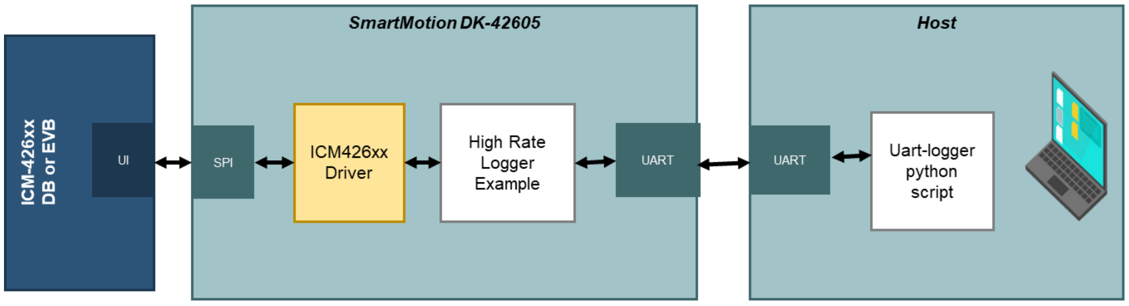
```
[I] ###################################
[I] #  ICM426xx streamlined example   #
[I] ###################################
[I] #######################
[I] #      WOM example    #
[I] #######################
[I] Initialize Icm426xx
[I] Check Icm426xx whoami value
[I] Waiting for detection...
[I] WoM interrupt at 8932 (X, Y, Z): 0, 0, 1
[I] WoM interrupt at 8744445 (X, Y, Z): 0, 0, 1
[I] WoM interrupt at 8764525 (X, Y, Z): 0, 0, 1
[I] WoM interrupt at 23605041 (X, Y, Z): 0, 0, 1
[I] WoM interrupt at 23625120 (X, Y, Z): 1, 0, 0
[I] WoM interrupt at 24930539 (X, Y, Z): 1, 0, 0
[I] WoM interrupt at 24950619 (X, Y, Z): 0, 1, 0
[I] WoM interrupt at 24970697 (X, Y, Z): 0, 1, 0
[I] WoM interrupt at 24990776 (X, Y, Z): 0, 0, 1
[I] WoM interrupt at 25030938 (X, Y, Z): 1, 1, 1
```

5.14. High Rate Logger Example

The example demonstrates how to get raw accel and gyro data from registers at high rate (8/16/32 kHz). This example has a specific interface configuration:

- **SPI4 only is supported at 24Mhz.** To maximize the register read transaction speed.
- **Log Console COM port is EDBG** at 921600 bauds
- **USB Serial Port is used to stream Uart data** at 3M baudrate

In addition of the FW, a python script is provided to log/parse Uart data on host side.



5.14.1. Firmware

The ODR can be configured using the SENSOR_ODR macro.

```
typedef enum {
       HIGH_RATE_32KHZ = 0, /* Warning: at 32 KHz, only one axis at a time can be logged */
       HIGH_RATE_16KHZ = 1,
       HIGH_RATE_8KHZ  = 2,
} SENSOR_ODR_t;
/*
 * Select sensor ODR (32, 16 or 8 KHz)
 * Please refer to the datasheet of your device to know the maximum ODR supported
 */
#define SENSOR_ODR   HIGH_RATE_32KHZ
```

When 32kHz is selected, only one axis can be read and reported at a time. This is due to very short time allowed to process and send data (<31µs). In this case, you can select which axis you would like to log using the AXIS_TO_LOG macro.

```
typedef enum {
       ACCEL_X_AXIS = 0,
       ACCEL_Y_AXIS = 1,
       ACCEL_Z_AXIS = 2,
       GYRO_X_AXIS  = 3,
       GYRO_Y_AXIS  = 4,
       GYRO_Z_AXIS  = 5,
} AXIS_TO_LOG_t;
/*
 * /!\ Only apply if HIGH_RATE_32KHZ is selected as SENSOR_ODR
 * Select which axis to log when ODR is 32 KHz
 */
#define AXIS_TO_LOG  ACCEL_X_AXIS
```

In order to log reliably, the firmware has to be built with optimization enabled. To do so, within Microchip Studio, right click on the project and open properties. Then, go to *Toolchain > ARM/GNU C Compiler Optimization* and set *Optimization level* to `Optimize most (-O3)` (default value should be `None (-O0)`).

COM port output

At 32 KHz:

```
[I] ##################################
[I] #   Example High Rate Logger     #
[I] ##################################
[I] Initialize Icm426xx
[I] Check Icm426xx whoami value
[I] Start streaming: Raw Accel x-axis at 32 KHz
```

`Frame_counter : data`

Frame_counter	Simple counter coded on 8bits, which is incrementing at each frame
data	Value read from register

At 16 or 8 KHz:

```
 [I] ##################################
[I] #   Example High Rate Logger     #
[I] ##################################
[I] Initialize Icm426xx
[I] Check Icm426xx whoami value
[I] Start streaming : Raw Accel x,y,z and Raw Gyro x,y,z at 16kHz
```

Frame_counter : racc_x racc_y racc_z raw_temperature rgyr_x rgyr_y rgyr_z

Frame_counter	Simple counter coded on 8bits, which is incrementing at each frame
racc_x	Raw accelerometer X value read from register
racc_y	Raw accelerometer Y value read from register
racc_z	Raw accelerometer Z value read from register
rgyr_x	Raw gyroscope X value read from register
rgyr_y	Raw gyroscope Y value read from register
rgyr_z	Raw gyroscope Z value read from register

5.14.2. Host script

A Python script is provided to collect and process the data. Please note that the processing is done once the collect is completed to prevent any slow-down which could cause data loss.

The following processing are available:

- **Printer**: Print all data on the terminal (warning: there will be a lot of lines)
- **Checker**: Check counter value and report error if a data is lost (delta > 1)
- **Logger**: Log all data into a file
- **Plotter**: Plot data using Matplotlib

To use this script, please install Python 3.8 or newer. For UART communication, Pyserial is used. For plotting, Matplotlib is used.

You can install them with the following command:

```
python -m pip install -U pip
python -m pip install -U pyserial
python -m pip install -U matplotlib
```

The help screen is printed with the following command:

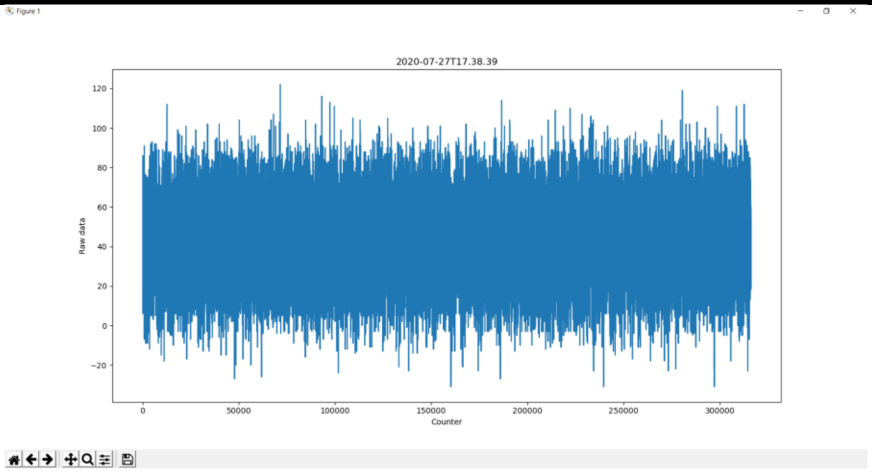
Here is help displayed on console :

```
>python uart_logger.py -h
Usage: uart_logger.py [options]
```

```
Options:
  -h, --help            show this help message and exit
  -p PORT, --port=PORT  USB Serial COM Port ID
  -s SPEED, --speed=SPEED
                        Uart baudrate
  -d DURATION, --duration=DURATION
                        Logging duration in second
  -r, --printer         Once record is completed, print data
  -c, --checker         Once record is completed, check counter for data
                        losses
  -l, --logger          Once record is completed, write data into a file
  -o, --plotter         Once record is completed, plot data using matplotlib
```

Example of usage (10 sec on port COM23, data will be logged and plotted):

```
> python uart_logger.py --port COM23 --duration 10 --logger --plotter
COM port: COM23
UART speed: 3000000
test duration (s): 10
print data: False
check counter: False
log data: True
plot data: True
Logging started. Ctrl-C to stop.
Parsing started. Ctrl-C to stop.
```



6. CODE SIZE

The code size of the driver and algorithm for each example is summarize below.

Example	Module	Code size (Bytes)	Data size (Bytes)
Algo	Driver	6.5K	0
	Algorithm	15.6K	2K
EIS	Driver	6.4K	0
Pedometer	Driver	4.9K	0
Raise to Wake	Driver	4.8K	0
Raw AG	Driver	7.1K	0
Raw Data Registers	Driver	5.1K	0
Sanity	Driver	2.9K	0
	Algorithm	15.6K	2K
Self-test	Driver	5.6K	0
SMD	Driver	4.6K	0
Tap	Driver	3.9K	0
Tilt	Driver	4.8K	0
WOM	Driver	4.6K	0