Software User Guide

For

DK-20789

Dev Kit

TABLE OF CONTENTS

USEFUL LINKS

TDK website:

https://www.invensense.com

https://www.microchip.com

http://www.microchip.com/developmenttools/productdetails.aspx?partno=atsamg55-xpro

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

1 OVERVIEW

The purpose of this document is to give an overview of the ICM207xx/SAMG55 Developer Kit that will allow users to create an application based on motion sensors. This document may also serve as a quick start guide for the ICM207xx package and its elements, including setup use of the sample applications.

1.1 INTRODUCTION

The ICM207xx/SAMG55 Dev Kit is compatible with the Atmel's ATSAMG55-XPRO evaluation kit based on a SAM G55 Cortex™-M4 processor-based microcontrollers. The supported development tools are Atmel Studio and Embedded Debugger. The purpose of this solution is to allow sensor management and algorithm processing by using a standalone microcontroller. The ICM207xx/SAMG55 solution is an embedded sensors combo (accelerometer, gyroscope & pressure) on chip, easy to integrate for users developing within the wearable and IoT space. The Dev Kit includes a full sensor software solution.

2 HARDWARE PLATFORM

The TDK Dev Kit platform for ICM207xx consists of the following components:

- TDK SAMG55 Dev Kit with onboard ICM207xx.

2.1 ATSAMG55-XPRO SETUP

The TDK SAMG55 Dev Kit includes a SAMG55J19A microcontroller. For more information on this MCU, please refer to Atmel/Microchip website (see *Useful Links* section above.)

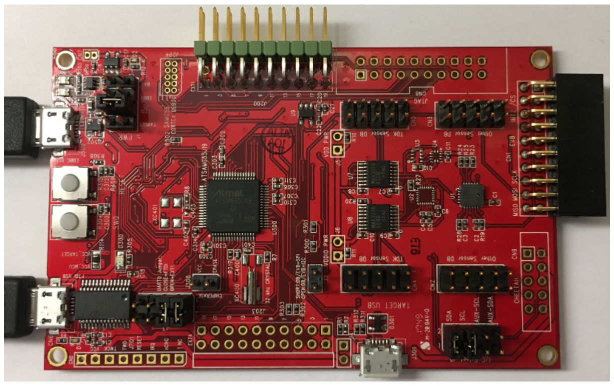


Figure 1 – TDK SAMG55 Dev Kit– Onboard 207xx

Figure 1 shows a TDK SAMG55 Dev Kit with an onboard ICM20789 seven axis sensor. The default jumper configuration is set for I2C interface.

2.2 ATSAMG55 BOARD SETUP

The systems pictured above are configured for I2C communication between the ATSAMG55 and the ICM207xx.

Jumper configurations -

- Power (J1):
    - To power the Dev Kit via the EDBG USB port (J500), connect a jumper across pins 3 & 4.
    - To receive power via the FTDI USB port (CN6), connect a jumper across pins 5 & 6 (default configuration)

  Note: The default configuration (power over the FTDI port) is useful when the firmware has already been flashed and no debugging is required. This configuration allows only for only one USB connection, via the FTDI port.

- I2C/SPI configuration (J2):
    - For communication between the ATSAMG55 and the ICM207xx via I2C (default), add jumpers between pins 1 & 2 and pins 3 & 4
    - For communication between the ATSAMG55 and the ICM207xx via SPI, remove the jumpers between pins 1 & 2 and pins 3 & 4.
    - For communication between the ATSAMG55 and the ICM207xx-pressure sensor, add jumpers between I2C auxiliary pins, 5 & 6 and 7 & 8

Note: By default, the firmware and hardware are setup for I2C communication. To use the SPI interface, both the hardware (as described above) and the firmware (as described below) must be changed.

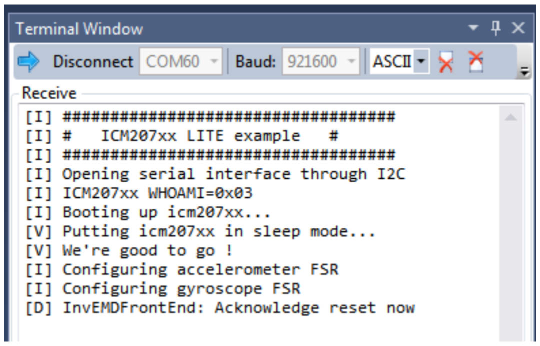
- UART (J3)
  - For UART communication over FTDI, connect pins 1 & 2 and pins 3 & 4.

2.2.1 Powering the SAMG55 Dev Kit

To power the platform, connect either the J500 or the CN6 port (based on the J1 jumper setting) to a PC using a micro-USB cable.

2.2.2 Debugging on the SAMG55 Dev Kit

To debug or flash the firmware, the EDBG USB port needs to be connected to a host PC using a micro-USB cable. There is also a provision for the firmware to print debug traces on the EDBG UART/USB connector. AtmelStudio Terminal Window can be used to read the messages at baud-rate 921600 by selecting the correct COM port. The serial line configurations are mentioned below. In order to disable data logging across the UART, the INV_MSG_ENABLE macro must be undefined and the firmware rebuilt. If the verbose level needs to be changed, the INV_MSG_ENABLE macro needs to be overloaded with the new desired verbose level.



```
[I] ####################################
[I] #    ICM207xx LITE example    #
[I] ####################################
[I] Opening serial interface through I2C
[I] ICM207xx WHOAMI=0x03
[I] Booting up icm207xx...
[V] Putting icm207xx in sleep mode...
[V] We're good to go !
[I] Configuring accelerometer FSR
[I] Configuring gyroscope FSR
[D] InvEMDFrontEnd: Acknowledge reset now
```

Figure 2 – Atmel Studio Terminal Window settings

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

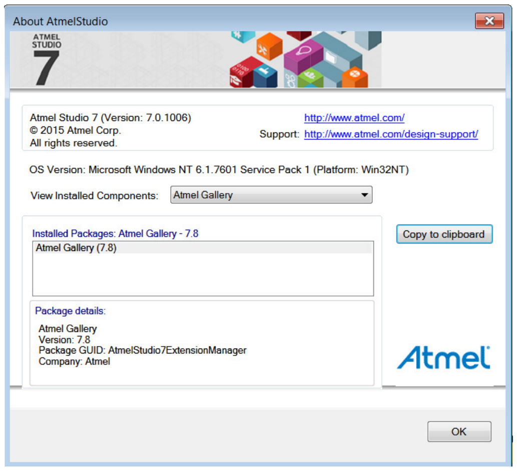
Figure 3 - Serial line configuration

3 SOFTWARE ENVIRONMENT

3.1 PREREQUISITE

To build and use the samples application provided as part for the DK-20789 Developer Kit packages, the following software is required:

- An RS232 terminal emulator (such as Putty: http://www.putty.org/)
  - To retrieve traces from provided FW application
- Atmel Studio: http://www.atmel.com/tools/atmelstudio.aspx
  - To load FW binaries and access to the USB EDBG port of the SAMG55 Dev Kit



Atmel Studio 7 (Version: 7.0.1006)
© 2015 Atmel Corp.
All rights reserved.

http://www.atmel.com/
Support: http://www.atmel.com/design-support/

OS Version: Microsoft Windows NT 6.1.7601 Service Pack 1 (Platform: Win32NT)

View Installed Components: Atmel Gallery

Installed Packages: Atmel Gallery - 7.8
Atmel Gallery (7.8)

Package details:
Atmel Gallery
Version: 7.8
Package GUID: AtmelStudio7ExtensionManager
Company: Atmel

Figure 4 - About Atmel Studio

3.2 EMD DEVELOPER KIT TDK PACKAGES

The following package is available:

- *eMD-SmartMotion_ICM207xx_x.y.z.zip*

This example targets low performance microcontrollers with a very simple sensor application.

Tools running on the PC for data display are available within the delivered package.

3.3 FW PACKAGE DESCRIPTION

The DK-20789 package includes all the necessary files to create a custom application using an ICM207xx device.

The package is organized as follow

- ***doc***: Document(s) describing the use of this firmware development platform.
- ***EMD-App***: contains sample firmware source and project files.
  - ***src:***
    - **At the top level:** Shared .c & .h files.
    - **ASF:** Shared Atmel system files.
    - **config:** Shared config files.
    - **ICM*:** Sensor specific files, main.[c,h], sensor.[c,h] and system.[c,h].
  - ****.cproj:*** AtmelStudio project files for each of the supported sensors.
- ***EMD-Core:*** Contains TDK driver files. These files are built into an archive libEMD-Core-ICM*.a. Each supported sensor has it's own .a file.
  - ***config :*** The Makefiles used to create the sensor driver archives.
  - ***Prebuilt/lib***: contains prebuilt TDK algorithm and math libraries (for Cortex-M4 with FPU)
  - ***sources/Invn:*** TDK libraries source files.
  - ****.cproj:*** AtmelStudio project files for each of the supported sensors.
- ***scripts* –** Batch files for building and flashing release versions of the firmware for each sensor.
- ***tools* –** The files required to run the host application sensor-cli.
- **EMD-G55-ICM*.atsln** – Atmel Studio solution files for each of the supported sensors.
- ***release*** – contains precompiled elf and binary files

4 BUILDING AND RUNNING SAMPLES APPLICATIONS

4.1 OVERVIEW

The following two projects are available:

➢ ***EMD-App* –** This application project demonstrates how to use TDK-InvenSense's low-level drivers to control and retrieve data from ICM devices. It encodes sensor events and sends them over the UART interface to be displayed by *sensor-cli*. The application uses the Core library and Algo libraries to generate a loadable binary.

➢ ***EMD-Core –*** This project includes low-level drivers and firmware code and generates the eMD Core library used by the *EMD-APP.*

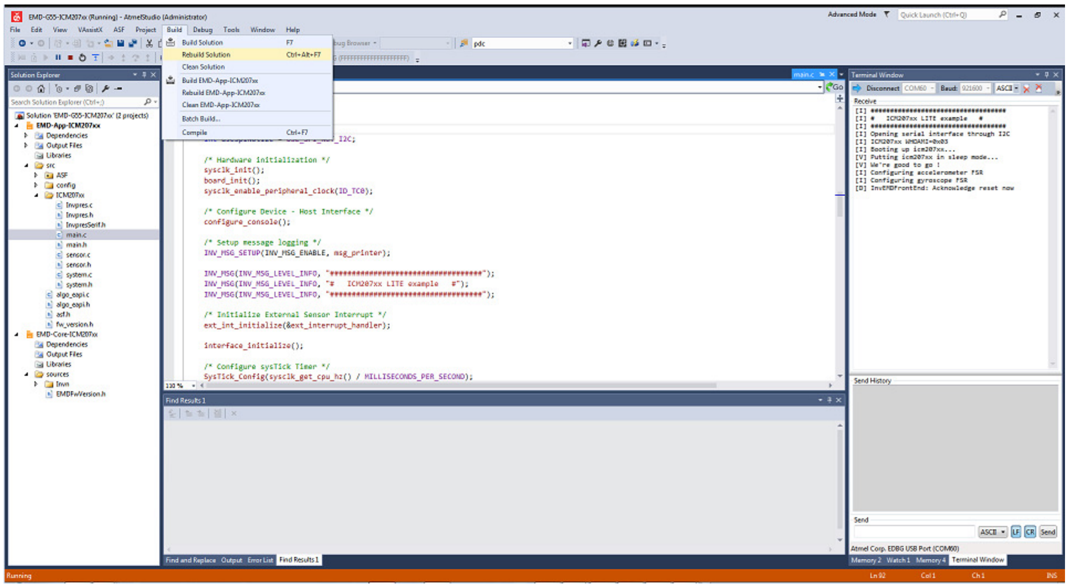
4.2 BUILDING EXAMPLE APPLICATIONS

A ready to use Atmel Studio project (EMD-G55-ICM207xx.atsln) is available in the root directory.

Refer to Atmel Studio website for details on how to install Atmel Studio, building the FW and loading it to the SAMG55 Dev Kit.

Additional information is available on this document Appendix section.

Atmel Studio can be used to compile both the EMD-APP and EMD-CORE projects.



```
    /* Hardware initialization */
    sysclk_init();
    board_init();
    sysclk_enable_peripheral_clock(ID_TC0);

    /* Configure Device - Host Interface */
    configure_console();

    /* Setup message logging */
    INV_MSG_SETUP(INV_MSG_ENABLE, msg_printer);

    INV_MSG(INV_MSG_LEVEL_INFO, "###################################");
    INV_MSG(INV_MSG_LEVEL_INFO, "#   ICM207xx LITE example   #");
    INV_MSG(INV_MSG_LEVEL_INFO, "###################################");

    /* Initialize External Sensor Interrupt */
    ext_int_initialize(&ext_interrupt_handler);

    interface_initialize();

    /* Configure sysTick Timer */
    SysTick_Config(sysclk_get_cpu_hz() / MILLISECONDS_PER_SECOND);
```

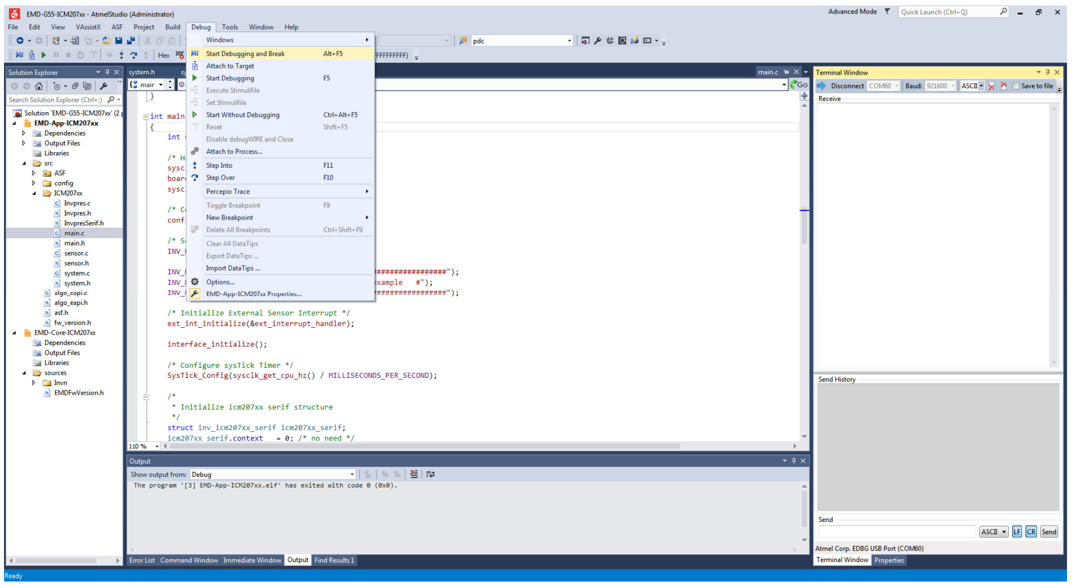
```
[I] ###################################
[I] #   ICM207xx LITE example   #
[I] ###################################
[I] Opening serial interface through I2C
[I] ICM207xx WHOAMI=0x03
[I] Booting up icm207xx...
[V] Putting icm207xx in sleep mode...
[V] We're good to go !
[I] Configuring accelerometer FSR
[I] Configuring gyroscope FSR
[D] InvEMDFrontEnd: Acknowledge reset now
```

Figure 5 - Building application

4.2.1 Running IDD-ICM207xx example application

This application targets compatibility with low performances microcontroller (Cortex-M0, M3, …). The application instantiates directly the ICM driver and communicates through low-level APIs. The algorithms are called in the application at the frequency specified. Data is reported through the UART interface using the same protocol as described above for IDD Wrapper case.

Atmel Studio can be used to download the compiled binary to the board via Embedded Debugger “EDBG USB” port.



```
    /* Initialize External Sensor Interrupt */
    ext_int_initialize(&ext_interrupt_handler);

    interface_initialize();

    /* Configure sysTick Timer */
    SysTick_Config(sysclk_get_cpu_hz() / MILLISECONDS_PER_SECOND);

    /*
     * Initialize icm207xx serif structure
     */
    struct inv_icm207xx_serif icm207xx_serif;
    icm207xx_serif.context   = 0; /* no need */
```

Show output from: Debug

```
The program '[3] EMD-App-ICM207xx.elf' has exited with code 0 (0x0).
```

Figure 6 - Image downloading, debugging and running application

The example supports a large set of features without using a sensor framework. Some simplifications have been made to streamline the code:

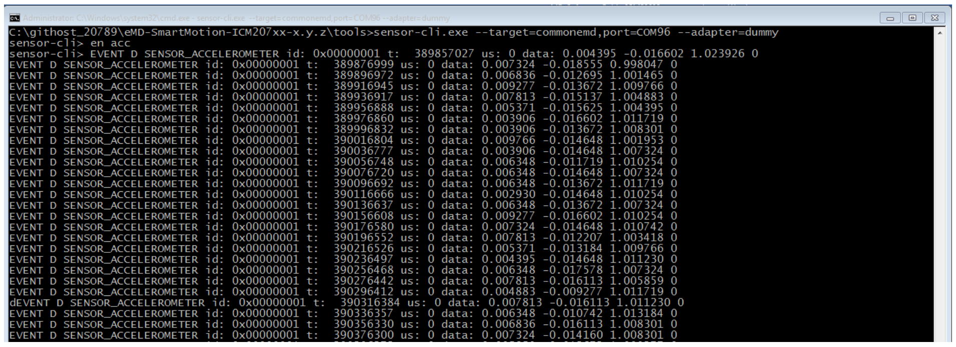
- At initialization, all algorithms are initialized and executed continuously at the default rate (50Hz)
- When a *set_sensor_period* command is received, **the ODR is changed for all sensors.** This ODR is checked to ensure it is within the valid ranges for all the enabled sensors.
- When an *enable_sensor* command is received, the data report is enabled (through UART)
- When a *disable_sensor* command is received, the data report is disabled but algorithms continue to run

To display data, run ***sensor-cli*** on the Windows PC from a console with the following argument.

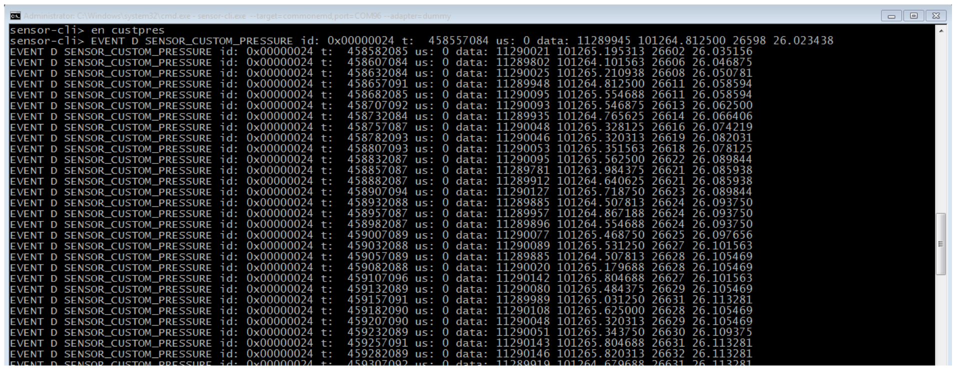
If only one SAMG55 system is connected to the PC, sensor-cli does not require any command line arguments. If there are multiple SAMG55 systems attached, the command should look like the following:

```
sensor-cli --target=commonemd,port=\\.\COMXX
```

Where COMxx is the comm port associated with the SAMG55 to be run.


Administrator: C:\Windows\system32\cmd.exe - sensor-cli.exe --target=commonemd,port=COM96 --adapter=dummy

```
C:\githost_20789\eMD-SmartMotion-ICM207xx-x.y.z\tools>sensor-cli.exe --target=commonemd,port=COM96 --adapter=dummy
sensor-cli> en acc
sensor-cli> EVENT D SENSOR_ACCELEROMETER id: 0x00000001 t:  389857027 us: 0 data: 0.004395 -0.016602 1.023926 0
EVENT D SENSOR_ACCELEROMETER id: 0x00000001 t:  389876999 us: 0 data: 0.007324 -0.018555 0.998047 0
EVENT D SENSOR_ACCELEROMETER id: 0x00000001 t:  389896972 us: 0 data: 0.006836 -0.012695 1.001465 0
EVENT D SENSOR_ACCELEROMETER id: 0x00000001 t:  389916945 us: 0 data: 0.009277 -0.013672 1.009766 0
EVENT D SENSOR_ACCELEROMETER id: 0x00000001 t:  389936917 us: 0 data: 0.007813 -0.015137 1.004883 0
EVENT D SENSOR_ACCELEROMETER id: 0x00000001 t:  389956888 us: 0 data: 0.005371 -0.015625 1.004395 0
EVENT D SENSOR_ACCELEROMETER id: 0x00000001 t:  389976860 us: 0 data: 0.003906 -0.016602 1.011719 0
EVENT D SENSOR_ACCELEROMETER id: 0x00000001 t:  389996832 us: 0 data: 0.003906 -0.013672 1.008301 0
EVENT D SENSOR_ACCELEROMETER id: 0x00000001 t:  390016804 us: 0 data: 0.009766 -0.014648 1.001953 0
EVENT D SENSOR_ACCELEROMETER id: 0x00000001 t:  390036777 us: 0 data: 0.003906 -0.014648 1.007324 0
EVENT D SENSOR_ACCELEROMETER id: 0x00000001 t:  390056748 us: 0 data: 0.006348 -0.011719 1.010254 0
EVENT D SENSOR_ACCELEROMETER id: 0x00000001 t:  390076720 us: 0 data: 0.006348 -0.014648 1.007324 0
EVENT D SENSOR_ACCELEROMETER id: 0x00000001 t:  390096692 us: 0 data: 0.006348 -0.013672 1.011719 0
EVENT D SENSOR_ACCELEROMETER id: 0x00000001 t:  390116666 us: 0 data: 0.002930 -0.014648 1.010254 0
EVENT D SENSOR_ACCELEROMETER id: 0x00000001 t:  390136637 us: 0 data: 0.006348 -0.013672 1.007324 0
EVENT D SENSOR_ACCELEROMETER id: 0x00000001 t:  390156608 us: 0 data: 0.009277 -0.016602 1.010254 0
EVENT D SENSOR_ACCELEROMETER id: 0x00000001 t:  390176580 us: 0 data: 0.007324 -0.014648 1.010742 0
EVENT D SENSOR_ACCELEROMETER id: 0x00000001 t:  390196552 us: 0 data: 0.007813 -0.012207 1.003418 0
EVENT D SENSOR_ACCELEROMETER id: 0x00000001 t:  390216526 us: 0 data: 0.005371 -0.013184 1.009766 0
EVENT D SENSOR_ACCELEROMETER id: 0x00000001 t:  390236497 us: 0 data: 0.004395 -0.014648 1.011230 0
EVENT D SENSOR_ACCELEROMETER id: 0x00000001 t:  390256468 us: 0 data: 0.006348 -0.017578 1.007324 0
EVENT D SENSOR_ACCELEROMETER id: 0x00000001 t:  390276442 us: 0 data: 0.007813 -0.016113 1.005859 0
EVENT D SENSOR_ACCELEROMETER id: 0x00000001 t:  390296412 us: 0 data: 0.004883 -0.009277 1.011719 0
dEVENT D SENSOR_ACCELEROMETER id: 0x00000001 t:  390316384 us: 0 data: 0.007813 -0.016113 1.011230 0
EVENT D SENSOR_ACCELEROMETER id: 0x00000001 t:  390336357 us: 0 data: 0.006348 -0.010742 1.013184 0
EVENT D SENSOR_ACCELEROMETER id: 0x00000001 t:  390356330 us: 0 data: 0.006836 -0.016113 1.008301 0
EVENT D SENSOR_ACCELEROMETER id: 0x00000001 t:  390376300 us: 0 data: 0.007324 -0.014160 1.008301 0
```


Administrator: C:\Windows\system32\cmd.exe - sensor-cli.exe --target=commonemd,port=COM96 --adapter=dummy

```
sensor-cli> en custpres
sensor-cli> EVENT D SENSOR_CUSTOM_PRESSURE id: 0x00000024 t:  458557084 us: 0 data: 11289945 101264.812500 26598 26.023438
EVENT D SENSOR_CUSTOM_PRESSURE id: 0x00000024 t:  458582085 us: 0 data: 11290021 101265.195313 26602 26.035156
EVENT D SENSOR_CUSTOM_PRESSURE id: 0x00000024 t:  458607084 us: 0 data: 11289802 101264.101563 26606 26.046875
EVENT D SENSOR_CUSTOM_PRESSURE id: 0x00000024 t:  458632084 us: 0 data: 11290025 101265.210938 26608 26.050781
EVENT D SENSOR_CUSTOM_PRESSURE id: 0x00000024 t:  458657091 us: 0 data: 11289948 101264.812500 26611 26.058594
EVENT D SENSOR_CUSTOM_PRESSURE id: 0x00000024 t:  458682085 us: 0 data: 11290095 101265.554688 26611 26.058594
EVENT D SENSOR_CUSTOM_PRESSURE id: 0x00000024 t:  458707092 us: 0 data: 11290093 101265.546875 26613 26.062500
EVENT D SENSOR_CUSTOM_PRESSURE id: 0x00000024 t:  458732084 us: 0 data: 11289935 101264.765625 26614 26.066406
EVENT D SENSOR_CUSTOM_PRESSURE id: 0x00000024 t:  458757087 us: 0 data: 11290048 101265.328125 26616 26.074219
EVENT D SENSOR_CUSTOM_PRESSURE id: 0x00000024 t:  458782093 us: 0 data: 11290046 101265.320313 26619 26.082031
EVENT D SENSOR_CUSTOM_PRESSURE id: 0x00000024 t:  458807093 us: 0 data: 11290053 101265.351563 26618 26.078125
EVENT D SENSOR_CUSTOM_PRESSURE id: 0x00000024 t:  458832087 us: 0 data: 11290095 101265.562500 26622 26.089844
EVENT D SENSOR_CUSTOM_PRESSURE id: 0x00000024 t:  458857087 us: 0 data: 11289781 101263.984375 26621 26.085938
EVENT D SENSOR_CUSTOM_PRESSURE id: 0x00000024 t:  458882087 us: 0 data: 11289912 101264.640625 26621 26.085938
EVENT D SENSOR_CUSTOM_PRESSURE id: 0x00000024 t:  458907094 us: 0 data: 11290127 101265.718750 26623 26.089844
EVENT D SENSOR_CUSTOM_PRESSURE id: 0x00000024 t:  458932088 us: 0 data: 11289885 101264.507813 26624 26.093750
EVENT D SENSOR_CUSTOM_PRESSURE id: 0x00000024 t:  458957087 us: 0 data: 11289957 101264.867188 26624 26.093750
EVENT D SENSOR_CUSTOM_PRESSURE id: 0x00000024 t:  458982087 us: 0 data: 11289896 101264.554688 26624 26.093750
EVENT D SENSOR_CUSTOM_PRESSURE id: 0x00000024 t:  459007089 us: 0 data: 11290077 101265.468750 26625 26.097656
EVENT D SENSOR_CUSTOM_PRESSURE id: 0x00000024 t:  459032088 us: 0 data: 11290089 101265.531250 26627 26.101563
EVENT D SENSOR_CUSTOM_PRESSURE id: 0x00000024 t:  459057089 us: 0 data: 11289885 101264.507813 26628 26.105469
EVENT D SENSOR_CUSTOM_PRESSURE id: 0x00000024 t:  459082088 us: 0 data: 11290020 101265.179688 26628 26.105469
EVENT D SENSOR_CUSTOM_PRESSURE id: 0x00000024 t:  459107096 us: 0 data: 11290142 101265.804688 26627 26.101563
EVENT D SENSOR_CUSTOM_PRESSURE id: 0x00000024 t:  459132089 us: 0 data: 11290080 101265.484375 26629 26.105469
EVENT D SENSOR_CUSTOM_PRESSURE id: 0x00000024 t:  459157091 us: 0 data: 11289989 101265.031250 26631 26.113281
EVENT D SENSOR_CUSTOM_PRESSURE id: 0x00000024 t:  459182090 us: 0 data: 11290108 101265.625000 26628 26.105469
EVENT D SENSOR_CUSTOM_PRESSURE id: 0x00000024 t:  459207090 us: 0 data: 11290048 101265.320313 26629 26.105469
EVENT D SENSOR_CUSTOM_PRESSURE id: 0x00000024 t:  459232089 us: 0 data: 11290051 101265.343750 26630 26.109375
EVENT D SENSOR_CUSTOM_PRESSURE id: 0x00000024 t:  459257091 us: 0 data: 11290143 101265.804688 26631 26.113281
EVENT D SENSOR_CUSTOM_PRESSURE id: 0x00000024 t:  459282089 us: 0 data: 11290146 101265.820313 26632 26.113281
EVENT D SENSOR_CUSTOM_PRESSURE id: 0x00000024 t:  459307092 us: 0 data: 11289919 101264.679688 26631 26.113281
```

Figure 7 – Sensor-cli console screen-shot

4.2.1.1 Default application behavior

At initialization, the application will:

- Initialize Atmel SAM G55 peripherals (IRQ, TIMER, SPI/I2C)
- Configure the UART (Baud-rate 2M)
- Initialize the drivers for the selected ICM device
- Setup and initialize the ICM device

4.2.1.2 Choosing between SPI and I2C

By default, I2C is used to communicate between ATSAMG55 and ICM device. This can be changed by setting #***define*** `USE_SPI_NOT_I2C` define to **1** (can be found in *system.h*) and by removing the jumpers between pins 1 & 2 and pins 3 and 4 of J2 (as described above).

4.2.1.3 Configuring the device

Full Scale Range (FSR), Mounting Matrix, can be changed by updating the value of the corresponding variables in *sensor.c*.

Default FSR value are **+/- 4g** for accelerometer and **+/- 2000dps** for gyroscope.

Supported FSR values are:

- Gyroscope: 250dps, 500dps, 1000dps and 2000dps
- Accelerometer: 2g, 4g, 8g and 16g
    - *Note:* Accelerometer FSR is expressed in **mg** in the driver stack and application.

The array cfg_mounting_matrix (for acc and gyro) is defined in *sensor.c*. Modifying the elements of these arrays will reconfigure the mounting matrix for the associated sensors.

Default mounting matrix is set to identity which corresponds to the following reference frame:

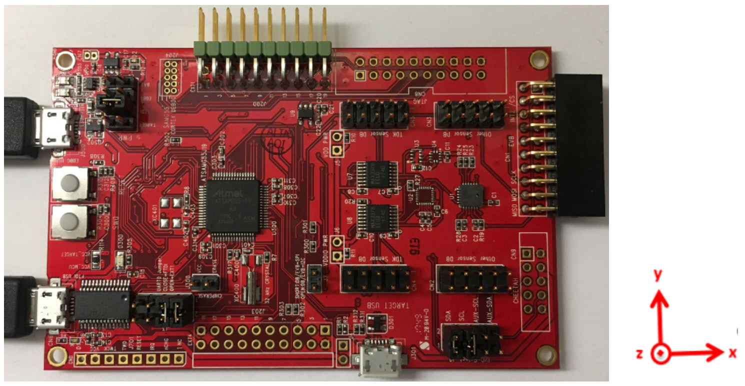


Figure 8 – board reference frame

4.2.1.4 Storing self-test and algorithm bias in NV memory

Algorithm bias storage to NV memory has not been supported.

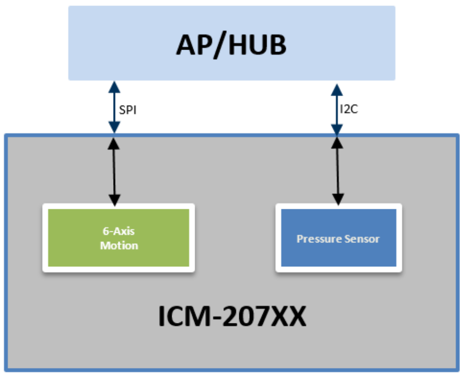


Figure 9 – ICM207XX Architecture diagram

Sensor features supported:

- Accelerometer
- Gyroscope
- Gravity
- Linear Acceleration
- Game rotation vector
- Uncalibrated gyroscope
- Raw accelerometer
- Raw Gyroscope
- Custom Pressure - output the following data:
    - Raw Pressure
    - Calibrated Pressure (in Pascal)
    - Raw Pressure Temperature
    - Pressure Temperature (in degree Celsius)

Supported command-set:

- Ping a sensor to check if it is supported by the device
- Enable/Disable sensor
- Set sensor period
- Self-tests

Operating range for sensors:

- Accel, Gyro, Gravity, Linear accel, Uncal gyro, Raw accel, Raw gyro: 4Hz – 200Hz
- Game rotation vector: 50Hz - 200Hz
- Custom Pressure: 1Hz – 40Hz

All sensors run at the same rate in the application.

All sensors are turned on at startup, but no data is reported across the UART until the sensor is enabled.

Note: Per design at start-up, all sensors and algorithms are started. The GRV orientation may drift until the gyroscope is calibrated. Once calibrated, the position is kept as the initial reference. The user may use this orientation as reference and only use the relative changes. To do so, you can refer to the following formula/code:

$$\boldsymbol{quat_{out} = quat_{grv}\,.\,conjugate(quat_{ref})}$$

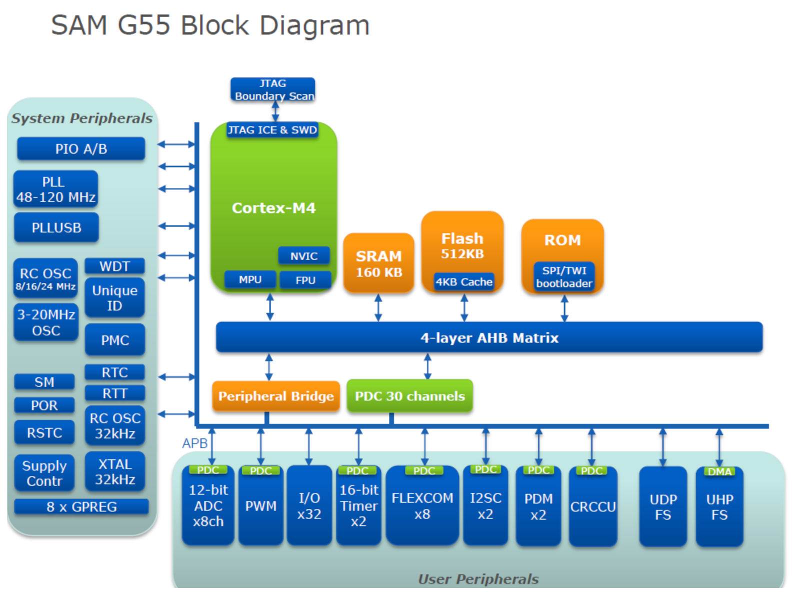
With:

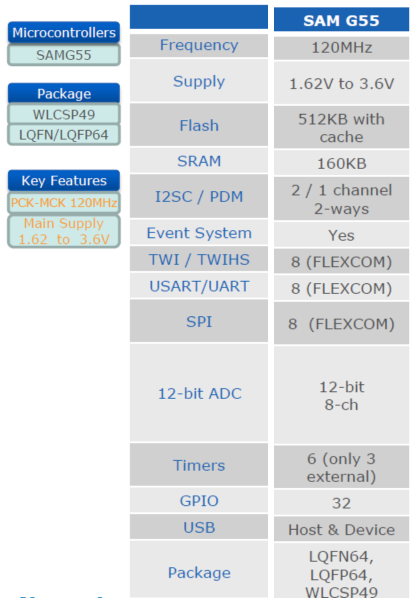
- $quat_{out}$ the result quaternion
- $quat_{grv}$ the quaternion obtained with the GRV sensor
- $quat_{ref}$, the quaternion that represents the position of reference

```
static void applyReferenceQuat(const float qin[4], const float q0[4], float qout[4])
{
          float q0c[4];
          // Conjugate
          q0c[0] = q0[0];
          q0c[1] = -q0[1];
          q0c[2] = -q0[2];
          q0c[3] = -q0[3];
          // Apply Compensation
          qout[0] = q0c[0]*qin[0] - q0c[1]*qin[1] - q0c[2]*qin[2] - q0c[3]*qin[3];
          qout[1] = q0c[0]*qin[1] + q0c[1]*qin[0] + q0c[2]*qin[3] - q0c[3]*qin[2];
          qout[2] = q0c[0]*qin[2] + q0c[2]*qin[0] + q0c[3]*qin[1] - q0c[1]*qin[3];
          qout[3] = q0c[0]*qin[3] + q0c[3]*qin[0] + q0c[1]*qin[2] - q0c[2]*qin[1];
          // Normalize
          float tmp = sqrtf(qout[0]*qout[0] + qout[1]*qout[1] + qout[2]*qout[2] + qout[3]*qout[3]);
          if (tmp > 0 ) {
                    qout[0] /= tmp;
                    qout[1] /= tmp;
                    qout[2] /= tmp;
                    qout[3] /= tmp;
          }
}
```

5 APPENDIX

5.1 ATMEL SAM G55-J19 ARCHITECTURE AND SPECIFICATIONS





Microcontrollers
SAMG55

Package
WLCSP49
LQFN/LQFP64

Key Features
PCK-MCK 120MHz
Main Supply 1.62 to 3.6V

	SAM G55
Frequency	120MHz
Supply	1.62V to 3.6V
Flash	512KB with cache
SRAM	160KB
I2SC / PDM	2 / 1 channel 2-ways
Event System	Yes
TWI / TWIHS	8 (FLEXCOM)
USART/UART	8 (FLEXCOM)
SPI	8 (FLEXCOM)
12-bit ADC	12-bit 8-ch
Timers	6 (only 3 external)
GPIO	32
USB	Host & Device
Package	LQFN64, LQFP64, WLCSP49

Figure 10 (SAMG55 block diagram)

Refer to Microchip's official site for further information on SAM-G55 architecture and system specifications.

6 DOCUMENT INFORMATION

6.1 REVISION HISTORY

REVISION	DATE	DESCRIPTION	AUTHOR
1.0	June 27, 2017	Initial version.	Rajesh Bisoi
1.1	September 1, 2017	Updated for new hardware.	Andrew Muir
1.2	October 2, 2017	Updated included file descriptions	Andrew Muir
1.3	October 6, 2017	General cleanup	Andrew Muir
1.4	March 19, 2018	Revised.	Rajesh Bisoi
1.5	March 23, 2018	Revised.	Rajesh Bisoi

Table 1. Revision History