

Software User Guide

For

**DK-20680/
DK-20680HP/
DK-20680HT
Dev Kit**

CONTENTS

Useful Links	3
1 Overview	4
1.1 Introduction	4
1.2 Architecture	4
2 Hardware Platform	5
2.1 DK-20680/DK-20680HP/DK-20680HT Overview.....	5
2.2 DK-20680/DK-20680HP/DK-20680HT Board Setup	6
2.2.1 Plugging-in Sensor EVB.....	6
2.2.2 Powering the SAMG55 Dev Kit.....	6
2.2.3 Debugging on the SAMG55 Dev Kit.....	6
3 Software Environment	7
3.1 Prerequisite.....	7
3.2 TDK eMD Developer Kit Packages.....	7
4 Example Applications	8
4.1 Overview	8
4.2 Building and Running Example Applications.....	8
4.2.1 Building the Example Application	8
4.2.2 Running the Example Application	9
4.2.3 Default Application Behavior	10
4.2.4 Expected Console Output	10
4.3 User Configurable Settings	11
4.3.1 Selecting IAM20680/IAM20680HP/IAM20680HT Product	11
4.3.2 System Settings	11
4.3.3 Sensor Settings.....	12
5 Porting Guide	14
5.1 In-depth Description of the SW Package	14
5.1.1 EMD-Core.....	14
5.1.2 EMD-App	14
5.2 Step-by-step Instructions for Driver Integration	15
5.2.1 Step 1 – Add files to your project.....	15
5.2.2 Step 2 – Select Correct Product	17
5.2.3 Step 3 – Implement MSG_PRINTER function	17
5.2.4 Step 4 – Implement READ/WRITE/SLEEP functions	18
5.2.5 Step 5 – Implement the system initialization flow.....	19
5.3 Integration Checklist	19
6 Document Information	20
6.1 Revision History	20

TABLE OF FIGURES

<i>Figure 1 Software Architecture Diagram</i>	<i>4</i>
<i>Figure 2 TDK SAMG55 Dev Kit – Onboard IAM20680</i>	<i>5</i>
<i>Figure 3 TDK SAMG55 Dev Kit – External IAM20680</i>	<i>5</i>
<i>Figure 4 Example Serial Console Configuration</i>	<i>6</i>
<i>Figure 5 About Atmel Studio</i>	<i>7</i>
<i>Figure 6 How to build using Atmel Studio</i>	<i>9</i>
<i>Figure 7 How to load and debug using Atmel Studio</i>	<i>9</i>
<i>Figure 8 Board reference frames for built-in sensor and external sensor boards</i>	<i>13</i>

USEFUL LINKS

TDK website:

<http://www.InvenSense.com/>

Atmel website:

<http://www.atmel.com>

<http://www.atmel.com/tools/atsamg55-xpro.aspx>

<http://www.atmel.com/tools/atmelstudio.aspx>

1 OVERVIEW

The purpose of this document is to give an overview of the IAM20680/IAM20680HP/IAM20680HT 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 IAM20680 package and its elements, including setup use of the sample applications.

1.1 INTRODUCTION

The IAM20680/IAM20680HP/IAM20680HT 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 by using a standalone microcontroller. The IAM20680/IAM20680HP/IAM20680HT ATSAMG55 solution is an embedded sensors combo (accelerometer & gyroscope) on chip, easy to integrate for users developing within the automotive space. The Dev Kit includes console example sensor software solutions.

1.2 ARCHITECTURE

The IAM20680/IAM20680HP/IAM20680HT chip contains accelerometer and gyroscope sensors and is accessible through SPI or I2C interface.

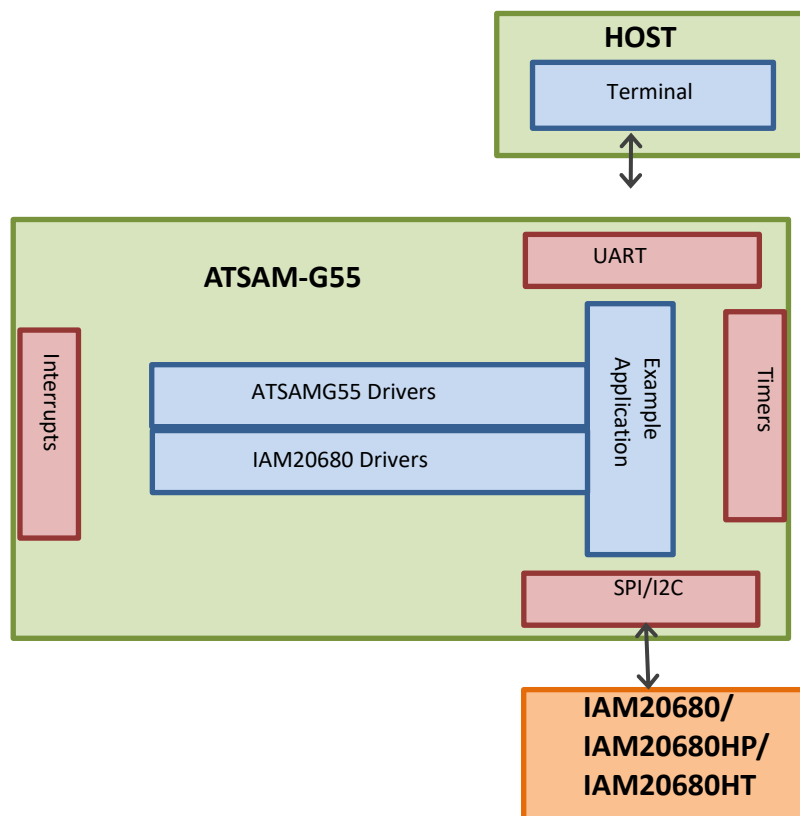


Figure 1 Software Architecture Diagram

2 HARDWARE PLATFORM

The TDK Dev Kit platform for IAM20680 family consists one of the following development platforms:

- DK-20680, TDK SAMG55 Dev Kit with onboard IAM20680 as an IAM20680 development platform.
- DK-20680HP, TDK SAMG55 Dev kit with onboard IAM20680HP as an IAM20680HP development platform.
- DK-20680HT, TDK SAMG55 Dev kit with onboard IAM20680HT as an IAM20680HT development platform.

2.1 DK-20680/DK-20680HP/DK-20680HT OVERVIEW

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

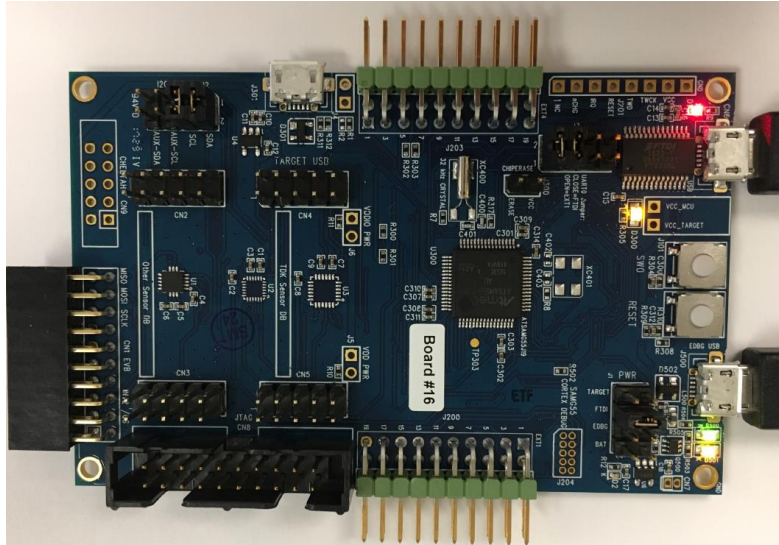


Figure 2 TDK SAMG55 Dev Kit – Onboard IAM20680

Figure 2 shows a TDK SAMG55 Dev Kit with an onboard IAM20680 six-axis sensor.

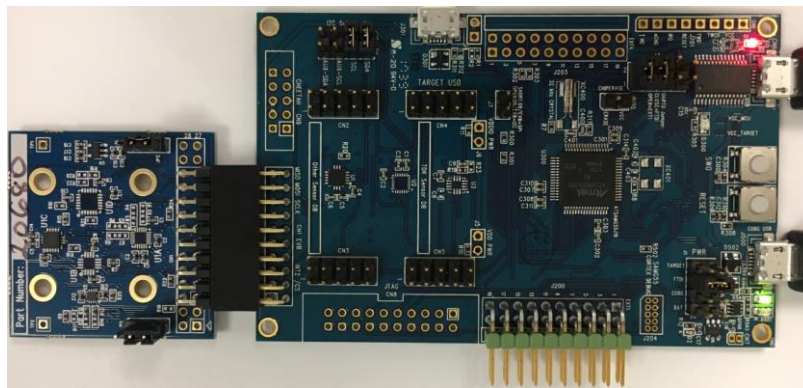


Figure 3 TDK SAMG55 Dev Kit – External IAM20680

Figure 3 shows a TDK SAMG55 Dev Kit which does not include an onboard sensor, connected with an external IAM20680 EVB board.

Note: The external IAM20680/IAM20680HP must be mounted on the slot labeled “CN1”.

2.2 DK-20680/DK-20680HP/DK-20680HT BOARD SETUP

The systems pictured above are configured for I2C communication between the ATSAMG55 and the IAM20680.

The jumper configuration is the same for both systems (onboard or external IAM20680/IAM20680HP/IAM20680HT).

- Power (J1):
 - o To power the Dev Kit via the EDBG USB port (J500), connect a jumper across pins 3 & 4.
 - o To receive power via the FTDI USB port (CN6), connect a jumper across pins 5 & 6.
- SPI/I2C configuration:
 - o For communication between the ATSAMG55 and the IAM20680 via I2C,
 - Add jumpers between pins 1 & 2 and pins 3 & 4 on J2.
 - Ensure J7 jumper pins are open on TDK SmartMotion Board RevB (only required if using external EVB).
 - o For communication between the ATSAMG55 and the IAM20680 via SPI,
 - Remove the jumpers between pins 1 & 2 and pins 3 & 4 on J2.
 - Ensure J7 jumper pins are closed on TDK SmartMotion Board RevB (only required if using external EVB).
 - o Note: By default, the firmware and hardware are setup for SPI communication. To use the I2C interface, both the hardware (as described above) and the firmware (as described below in section 4.3.2.1) must be changed.
- UART (J3)
 - o For UART communication over FTDI, connect pins 1 & 2 and pins 3 & 4.

2.2.1 Plugging-in Sensor EVB

The default configuration is for DK-20680/DK-20680HP/DK-20680HT 9-14-2017 Ver. B board. There is provision to use older board revisions and/or external IAM20680 sensor boards.

The external IAM20680/IAM20680HP/IAM20680HT EVB must be connected on the slot labelled "CN1". Set the following compile-time macros for SmartMotion board and sensor configurations. The default configuration is set for SmartMotion RevB Board with built-in IAM20680/IAM20680HP/IAM20680HT sensor. (refer to EMD-App\src\IAM20680\system.h)

```
/* Set 0 for external sensor board on TDK SmartMotion Board *
 * Set 1 for in-built sensor board on TDK SmartMotion Board */
#define TDK_BOARD_INBUILT_SENSOR 1

/* Set 0 for TDK SmartMotion RevA *
 * Set 1 for TDK SmartMotion RevB */
#define TDK_BOARD_REVISION 1
```

2.2.2 Powering the SAMG55 Dev Kit

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

2.2.3 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. A serial terminal emulator can be used to read the messages at baud-rate 921600 by selecting the correct COM port.

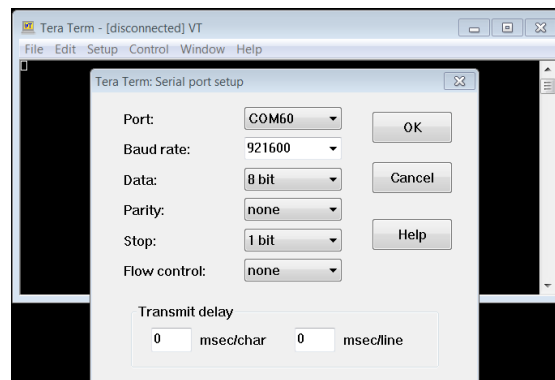


Figure 4 Example Serial Console Configuration

3 SOFTWARE ENVIRONMENT

3.1 PREREQUISITE

To build and use the samples application provided as part for the DK-20680 Developer Kit packages, the following software is required:

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

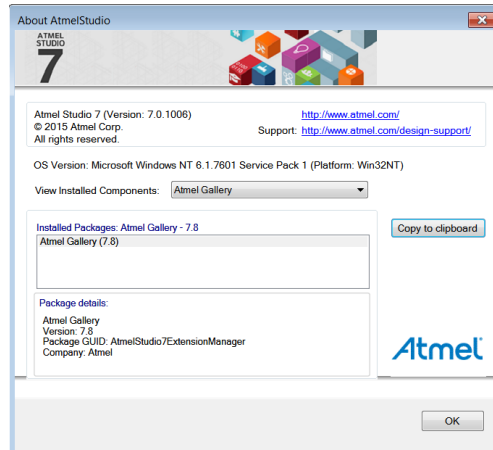


Figure 5 About Atmel Studio

3.2 TDK EMD DEVELOPER KIT PACKAGES

The release packages are available:

- o *eMD-SmartMotion-IAM20680-x.y.z.zip*

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

Note: *x, y, z stands for the release version*

The eMD-SmartMotion-IAM20680 package includes all the necessary files to create a custom application using an IAM20680/IAM20680HP/IAM20680HT device.

The package is organized as follows:

- **doc:** Documents describing the use of this firmware development platform.
- **EMD-App:** Contains sample firmware source and project files.
 - o **src:**
 - **At the top level:** Shared .c & .h files.
 - **ASF:** Shared Atmel system files.
 - **config:** Shared config files.
 - **IAM20680:** Sensor specific files, main.[c,h], sensor.[c,h] and system.[c,h] for controlling the motion sensor (IAM20680/IAM20680HP/IAM20680HT)
 - o ***.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-IAM*.a. Each supported sensor has its own .a file.
 - o **config :** The Makefiles used to create the sensor driver archives.
 - o **sources/Inven:** TDK libraries source files.
 - o ***.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.
- **EMD-G55-IAM20680-xxx.atsln** – Atmel Studio solution files for each example.
- **release\IAM20680:** contains precompiled binary and elf files for the applications.

4 EXAMPLE APPLICATIONS

4.1 OVERVIEW

The following example solutions are available in the package -

- **example-raw-ag**
 - An example that demonstrates raw accelerometer and raw gyroscope data streaming. The default configuration is low-noise mode and ODR is 50Hz.
- **example-wom**
 - An example that demonstrates the configuration and use of the WOM (Wake-on-Motion) feature as well as full FIFO read when WOM interrupt is received. The default configuration is low-power and ODR is 500Hz. The default FIFO size for IAM20680HP/IAM20680HT is set to 4 KB. For IAM20680HT, this example also shows how to enable INT2 pin. User can toggle the compile-time macro **#define ENABLE_INT2_PIN** in EMD-App\src\IAM20680\system.h to enable/disable INT2 pin.
- **example-selftest**
 - An example that performs the accel and gyro self-test and provides PASS/FAIL results as well as gyro/accel low noise/low power biases.

For each of the above solutions, the following two projects have been included -

- **EMD-App** – This application project demonstrates how to use low-level drivers to control and retrieve data from IAM devices. It sends data over the UART interface to be displayed on host console. The application uses the Core library 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 AND RUNNING EXAMPLE APPLICATIONS

The following ready-to-use Atmel Studio projects are available in the root directory.

- EMD-G55-IAM20680-raw-ag.atsln
- EMD-G55-IAM20680-selftest.atsln
- EMD-G55-IAM20680-wom.atsln

Refer to Atmel Studio website for details on how to install Atmel Studio, building the FW and loading elf/binary to Atmel SAM-G55 Dev Kit.

4.2.1 Building the Example Application

Atmel Studio can be used to compile both the EMD-APP and EMD-CORE projects. It is required to do a “Rebuild Solution” whenever there are changes made in the EMD-CORE project.

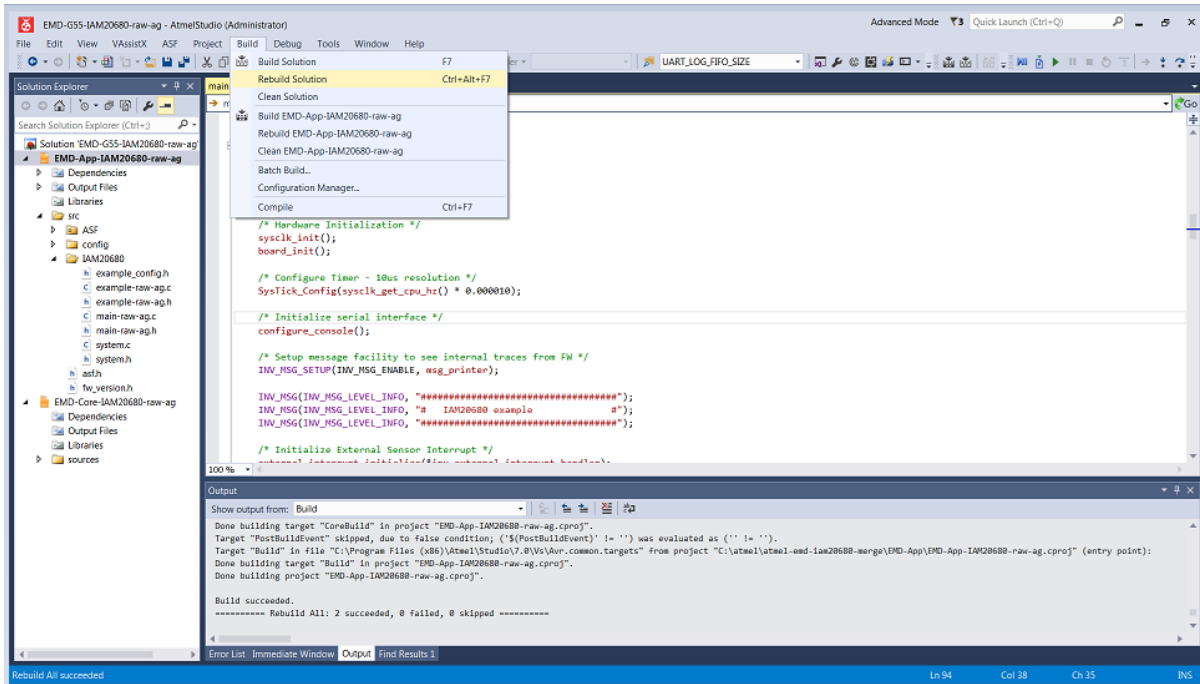


Figure 6 How to build using Atmel Studio

4.2.2 Running the Example Application

This application targets compatibility with low performances microcontroller (Cortex-M0, M3, ...). The application instantiates directly the IAM driver and communicates through low-level APIs. The algorithms are called in the application at the frequency specified. The data is reported through the UART interface.

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

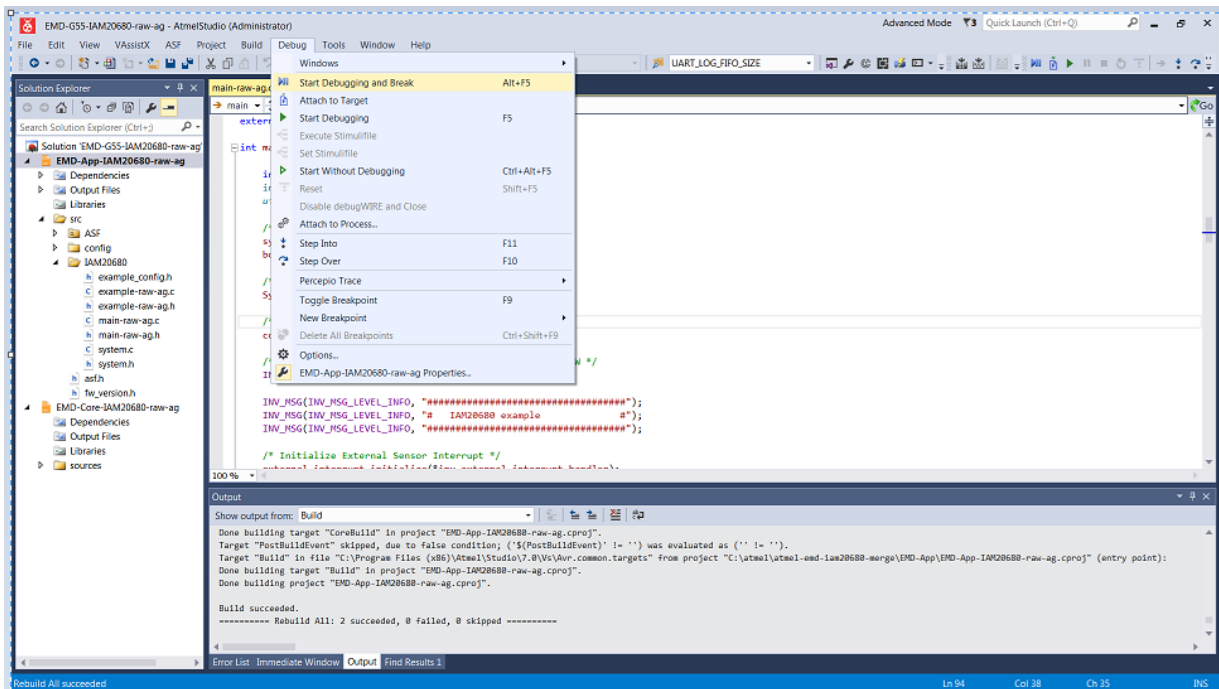


Figure 7 How to load and debug using Atmel Studio

4.2.3 Default Application Behavior

Once flashed, these applications will automatically start running at power on. Data output will automatically be sent to the carrier board UART/USB connector.

At startup, the sample applications will:

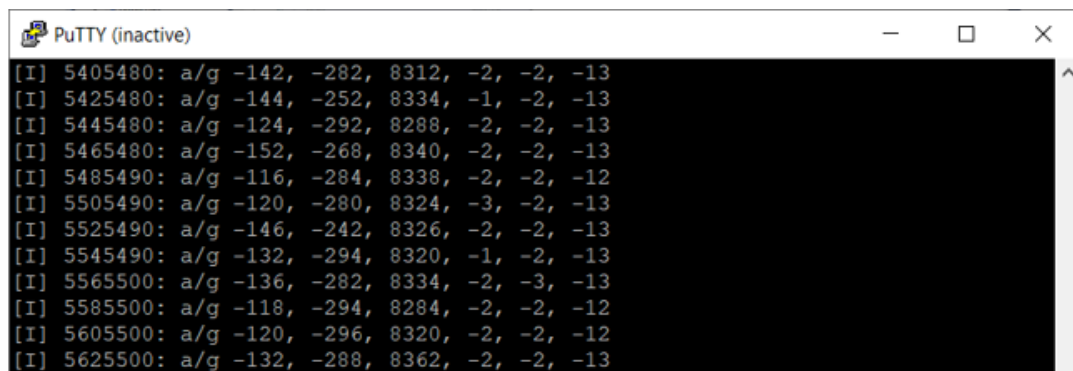
- Initialize ATSAMG55 peripherals (IRQ, TIMER, SPI/I2C)
- Configure serial interfaces (UART/USART) to get traces on COM port.
- Configure IAM/Sensor specific registers

4.2.4 Expected Console Output

➤ *example-raw-ag*

If both accel and gyro are enabled, accel x/y/z and gyro x/y/z data will be output. The unit of the data is in LSB. The output format is:

timestamp(us): a/g AX, AY, AZ, GX, GY, GZ



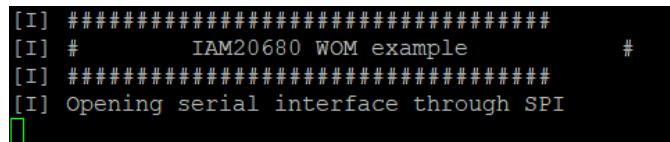
```

PuTTY (inactive)
[I] 5405480: a/g -142, -282, 8312, -2, -2, -13
[I] 5425480: a/g -144, -252, 8334, -1, -2, -13
[I] 5445480: a/g -124, -292, 8288, -2, -2, -13
[I] 5465480: a/g -152, -268, 8340, -2, -2, -13
[I] 5485490: a/g -116, -284, 8338, -2, -2, -12
[I] 5505490: a/g -120, -280, 8324, -3, -2, -13
[I] 5525490: a/g -146, -242, 8326, -2, -2, -13
[I] 5545490: a/g -132, -294, 8320, -1, -2, -13
[I] 5565500: a/g -136, -282, 8334, -2, -3, -13
[I] 5585500: a/g -118, -294, 8284, -2, -2, -12
[I] 5605500: a/g -120, -296, 8320, -2, -2, -12
[I] 5625500: a/g -132, -288, 8362, -2, -2, -13
  
```

If only one sensor is enabled, then the 3-axis data of the enabled sensor will be output.

➤ *example-wom*

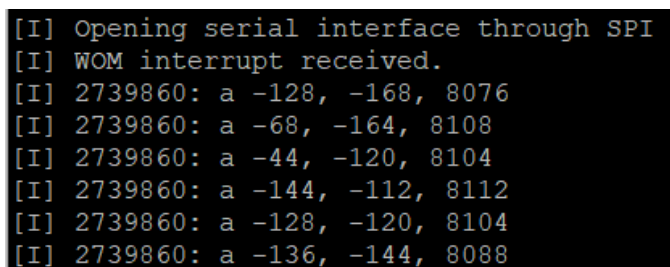
Upon power up, the chip will be in accel only low power mode, waiting for a motion. No data is output at this moment.



```

[I] #####
[I] #           IAM20680 WOM example           #
[I] #####
[I] Opening serial interface through SPI
  
```

Once a motion is detected, the chip wakes up and accel data will be output.



```

[I] Opening serial interface through SPI
[I] WOM interrupt received.
[I] 2739860: a -128, -168, 8076
[I] 2739860: a -68, -164, 8108
[I] 2739860: a -44, -120, 8104
[I] 2739860: a -144, -112, 8112
[I] 2739860: a -128, -120, 8104
[I] 2739860: a -136, -144, 8088
  
```

➤ *example-selftest*

Upon power up, the accel and gyro self-test will be performed, and PASS/FAIL results will be printed. Then the factory calibration will run and collect gyro and accel bias in low noise and low power mode. The bias values will be printed.

```
[I] #####
[I]     IAM20680 Self Test example
[I]     Ver: 0.0.0
[I] #####
[I] Opening serial interface through SPI
[I] ACC Trim offset (g): x=0xe71a, y=0xe312, z=0x13d8
[I] Gyro Selftest PASS
[I] Accel Selftest PASS
[I] Factory Calib Collecting LN/LP bias
[I] GYR LN bias (FS=250dps) (dps): x=0.373840, y=0.030518, z=0.038147
[I] GYR LP bias (FS=250dps) (dps): x=0.137329, y=-0.205994, z=-0.213623
[I] ACC LN bias (FS=2g) (g): x=-0.032898, y=-0.149597, z=-0.025146
[I] ACC LP bias (FS=2g) (g): x=-0.034729, y=-0.151794, z=-0.018555
[I]
```

4.3 USER CONFIGURABLE SETTINGS

4.3.1 Selecting IAM20680/IAM20680HP/IAM20680HT Product

Sample applications can be used for any one of the three products. User can select which product to use by setting the corresponding macro to 1 in file EMD-Core\sources\Invn\Devices\Drivers\Iam20680\Iam20680Product.h.

```
#define IAM20680          0 /* Set to 1 if sensor connected is IAM-20680, else 0 */
#define IAM20680_HP      0 /* Set to 1 if sensor connected is IAM-20680HP, else 0 */
#define IAM20680_HT      1 /* Set to 1 if sensor connected is IAM-20680HT, else 0 */
```

4.3.2 System Settings

All the user configurable system settings are defined in EMD-App\src\IAM20680\system.h.

4.3.2.1 Serial Interface Options

By default, SPI is used to communicate between ATSAMG55 and IAM device. This can be changed by setting

```
#define SERIF_TYPE_SPI    0
#define SERIF_TYPE_I2C    1
```

and by adding the jumpers between pins 1 & 2 and pins 3 and 4 of J2 (as described in Section 2.2).

4.3.2.2 DK Board Type and Revision

The default configuration is set for SmartMotion RevB Board with built-in IAM20680/IAM20680HP/IAM20680HT sensor.

There is provision to use older board revisions and/or external IAM20680 sensor boards. The external sensor EVB must be connected on the slot labelled "CN1".

Set the following compile-time macros for SmartMotion board and sensor configurations.

```
/* Set 0 for external, 1 for in-built sensor on TDK SmartMotion Board */
#define TDK_BOARD_INBUILT_SENSOR    1
/* Set 0 for TDK SmartMotion RevA, 1 for TDK SmartMotion RevB/B+/C */
#define TDK_BOARD_REVISION          1
```

4.3.2.3 Enable/Disable INT2 Pin

For IAM20680HT product, user can set the following compile-time macro to enable/disable INT2 pin:

```
/* Set 1 to enable INT2 so that all interrupts except for data ready appear on the INT2 pin, and
data ready interrupt appears on the INT interrupt pin. */
/* Set 0 to disable INT2 so that all of the interrupts appear on the INT pin, and INT2 interrupt pin
is unused. */
#define ENABLE_INT2_PIN            1
```

4.3.3 Sensor Settings

All the user configurable sensor settings are defined in EMD-App\src\IAM20680\example_config.h. The usage of these settings is only demonstrated in the example-raw-ag application.

4.3.3.1 Enable/Disable Accel/Gyro

Accel and Gyro are both enabled by default in the sample application example-raw-ag. They can be disabled individually by changing the build flags `"#define ENABLE_ACCEL"` or `"#define ENABLE_GYRO"` to 0 in example_config.h.

4.3.3.2 Low-Power or Low-Noise mode

The sample application example-raw-ag can be run in either low-power or low-noise mode. By default, the application is in low-noise mode. But this can be changed by setting `#define LOW_NOISE_MODE` to 0 and `#define LOW_POWER_MODE` to 1 in example_config.h.

Accelerometer only low power mode allows ODR to be configurable within small subset of ODR's. Refer IAM20680/IAM20680HP/IAM20680HT data sheet for detailed explanation.

The sample application will configure accelerometer low power ODR based on input ODR as per following table:

Input ODR in Hz	Configured Accel Only LP ODR in Hz
Input_ODR < 5.85	3.9
Input_ODR >= 5.85 && Input_ODR < 11.7	7.8
Input_ODR >= 11.7 && Input_ODR < 23.45	15.6
Input_ODR >= 23.45 && Input_ODR < 46.9	31.3
Input_ODR >= 46.9 && Input_ODR < 93.75	62.5
Input_ODR >= 93.75 && Input_ODR < 187.5	125
Input_ODR >= 187.5 && Input_ODR < 375	250
Input_ODR >= 375	500

Table 1 Accel Only Low Power Mode ODR

4.3.3.3 Configuring the device

ODR and Full-Scale Range (FSR) can be configured for example-raw-ag.

- **ODR**
ODR can be configured by setting `#define ODR_US` to desired data rate (in microseconds) in example_config.h.
- **FSR**
FSR can be configured by setting `#define FSR_ACC_G` and `#define FSR_GYR_DPS` to desired full scale range for accel (in g) and gyro (in dps) in example_config.h. 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
- **Board reference frame**

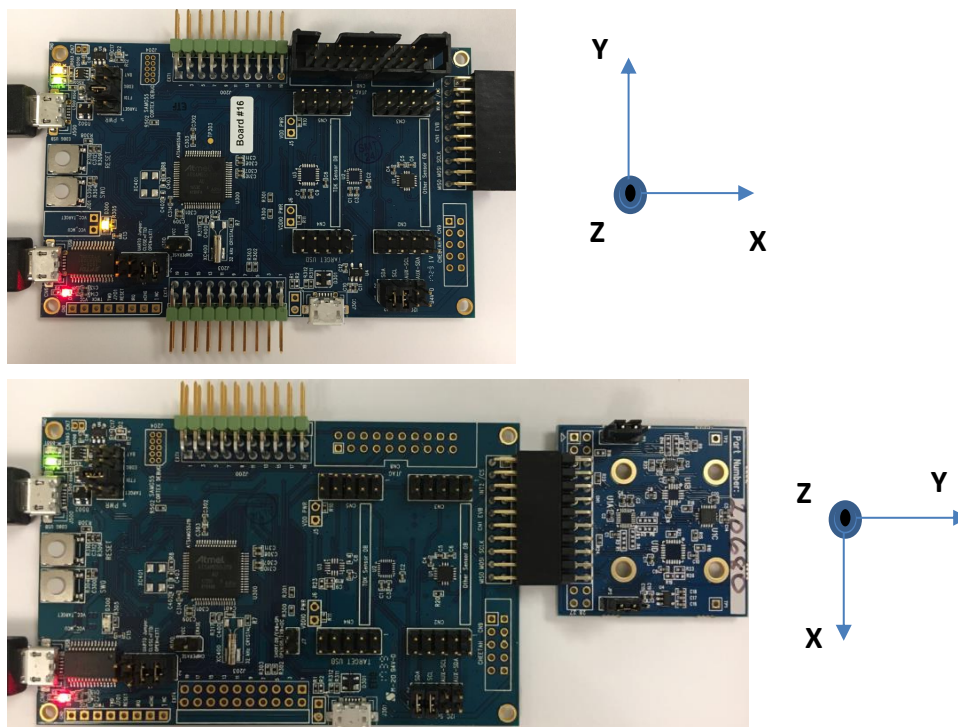


Figure 8 Board reference frames for built-in sensor and external sensor boards

5 PORTING GUIDE

5.1 IN-DEPTH DESCRIPTION OF THE SW PACKAGE

5.1.1 EMD-Core

The EMD-Core directory contains device driver for IAM20680 (EMD-Core\sources\Invn\Devices\Drivers\IAM20680) and embedded utility functions (EMD-Core\sources\Invn\EmbUtils).

All the files under EMD-Core folder are platform independent therefore do not require modifications except for selecting the correct product in `Iam20680Product.h`.

Here are detailed descriptions of the device driver files:

- **Iam20680Defs.h**
 - contains definitions of registers, bit-fields and declarations of low-level driver APIs.
 - **Iam20680Driver.c**
 - contains implementations of low-level driver APIs (read/write a certain bit field of a certain register).
 - **Iam20680_HL.c/h**
 - contains high level driver APIs, which calls low-level APIs. These functions implement certain sequences or perform some logic to provide user convenience in controlling the device.
 - **Iam20680ExtFunc.h**
 - contains declarations of hooks to the system-dependent functions. These functions need to be implemented by upper layer.
 - **Iam20680Product.h**
 - contains macro definitions for the 3 products this software package supports.
 - **Iam20680SelfTest.c/h**
 - contains functions to perform accel and gyro self-test.
 - **Iam20680Transport.c/h**
 - contains functions to read/write registers.
 - The serial interface transport functions – SPI/I2C read/write, need to be implemented in the upper layer. The function pointers need to be assigned to `inv_iam20680_serif` structure, which is part of the device driver states structure `inv_iam20680_t`.
 - The example code to pass these function pointers can be found in `SetupInnvDevice` function in the EMD-App\src\IAM20680\example-xxx.c files.
- ```

/*!
 * \brief Set up the InvenSense device
 * \param[in] read_reg Function pointer for reading register
 * \param[in] write_reg Function pointer for writing register
 * \param[in] isSPI 1: using SPI interface; 0: using I2C interface
 */
int SetupInnvDevice(int (*read_reg)(void * context, uint8_t reg, uint8_t * buf, uint32_t len),
 int (*write_reg)(void * context, uint8_t reg, const uint8_t * buf, uint32_t len),
 inv_bool_t isSPI);

```

#### 5.1.2 EMD-App

The EMD-App directory contains platform dependent functions and example code to interface the driver for configuring the sensor and streaming data, etc.

Here are detailed descriptions of the application files (EMD-App\src\IAM20680):

- **system.c/h**
  - contains system functions that are dependent on Atmel SAMG55 platform, including:
    - SPI/I2C serial interface functions, for accessing device registers
    - UART console configuration function, for outputting debug traces
    - External interrupt initialization, for receiving interrupts, like data ready interrupt and WoM interrupt
    - SysTick Timer handler to get accurate timestamps and a ring buffer to store the timestamps for sensor data
  - Users need to implement these functions based on their own platform
- **example-xxx.c/h**



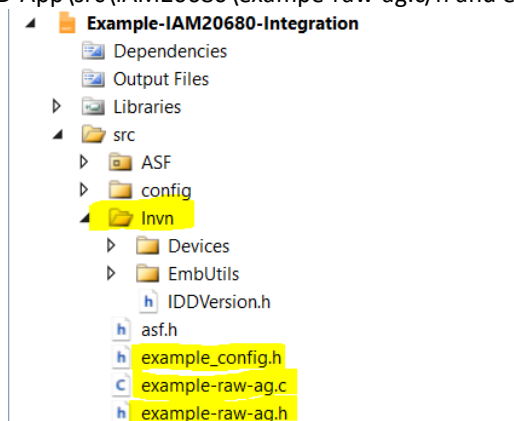
- contains example code to interface the driver for device initialization, sensor configuration, data collection, etc. The example functions included in each example may be different in order to showcase specific usages.
- example functions include:
  - *SetupInvDevice* (included in all the examples)  
This function does device initialization:
    - Setting serial interface function pointers
    - Initialize driver state structure `inv_iam20680_t`
    - Check WHOAMI
    - Initialize device
    - Plus any init functions specific to each example
  - *ConfigureInvDevice* (included in example-raw-ag)  
This function shows how to configure various sensor settings, including sensor enabling, power mode, full scale range (fsr), output data rate (odr).
  - *GetDataFromFIFO* (included in example-raw-ag)  
This function checks interrupt status to ensure data ready status, then reads FIFO to get sensor data. Sensor data is output to UART DBG port.
  - *ConfigureInvDeviceForWOM* (included in example-wom)  
This function configures the device to work in wake-on-motion mode
  - *ProcessIAM20680Irq* (included in example-wom)  
This function shows how to check for wom interrupt, then configures the device back to normal mode (enables data ready interrupt) upon receiving wom interrupt, and then get sensor data.
  - *RunSelfTest* (included in example- selftest)  
This function performs selftest on accel and gyro and get PASS/FAIL results.
  - *RunFactoryCalib* (included in example- selftest)  
This function collects gyro and accel bias in both low noise and low power modes.
  - *ApplyOffset* (included in example-selftest)  
This function applies the offset that is computed from RunFactoryCalib function to accel and gyro.
- **main-xxx.c/h**
  - The main function implements the entire workflow.

## 5.2 STEP-BY-STEP INSTRUCTIONS FOR DRIVER INTEGRATION

This chapter shows the integration of raw-ag example on Atmel studio platform as an example. For other platforms, apply the platform specific method for below steps to implement the integration.

### 5.2.1 Step 1 – Add files to your project

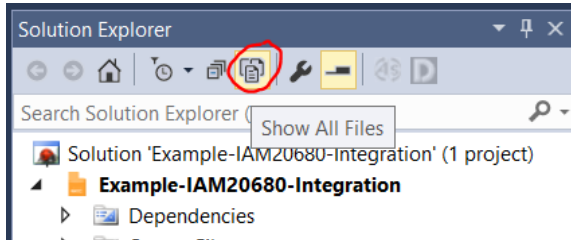
1. From the sw release package, copy the following Invn driver folder and example files to your own project folder and add them to your project:
  - EMD-Core\sources\Invn
  - EMD-App\src\IAM20680\exampe-raw-ag.c/h and example\_config.h



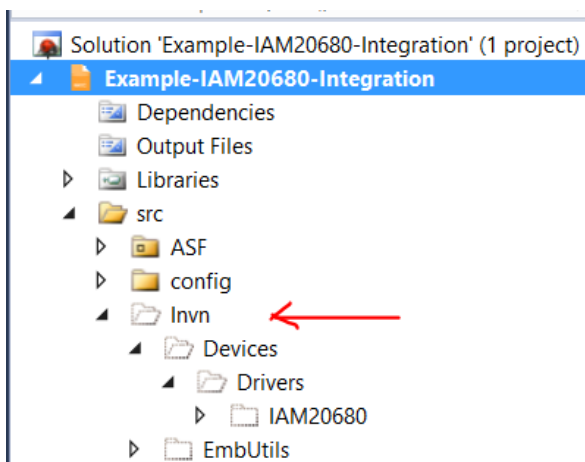


In order to add the entire Invn folder to your project in its original folder structure, you can follow below steps:

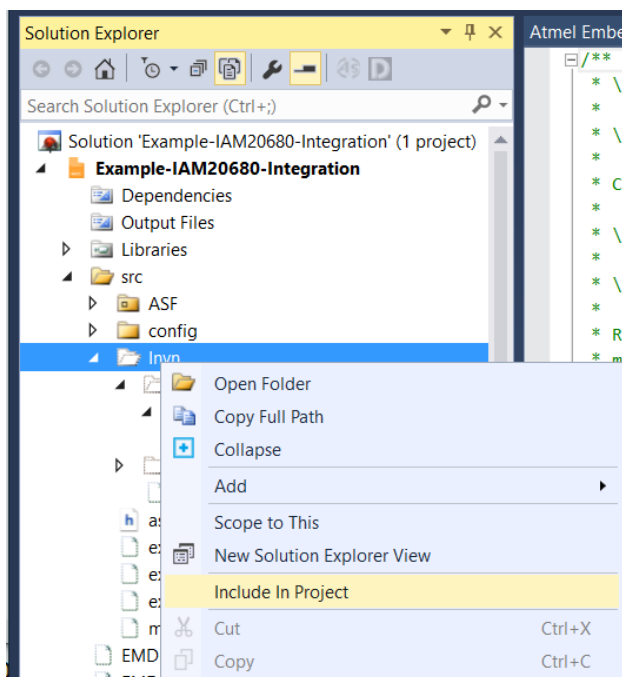
- a. Make sure your project is selected as "Startup Project"
- b. Click on "Show All Files".



The source code directory you copied into the project directory should show up here as a "dashed folder".

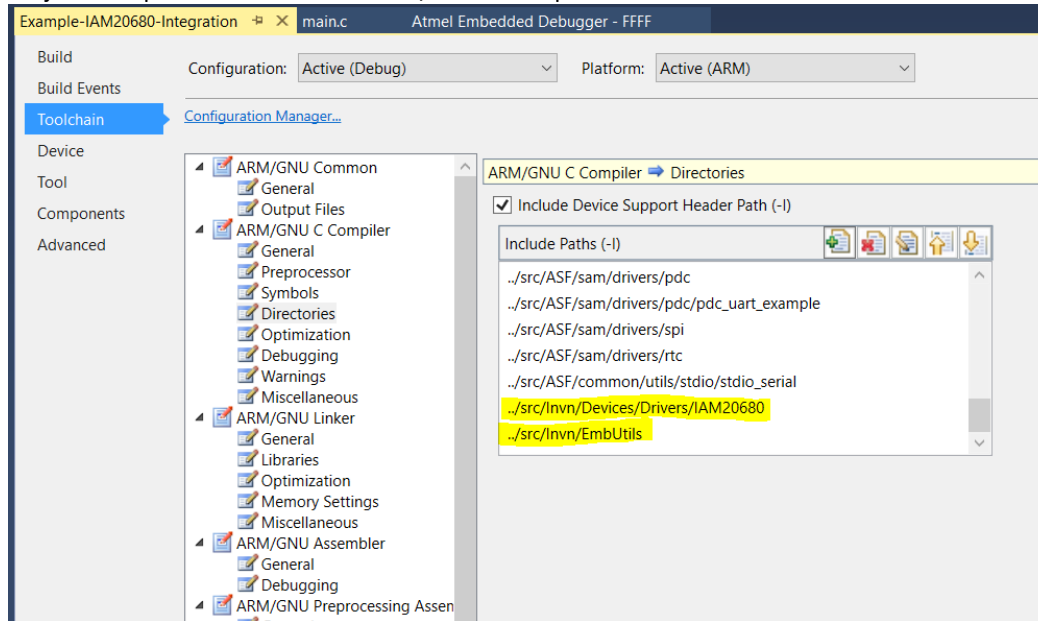


- c. Right click on the folder and select "Include In Project"



2. Add the directory of the header files to the compiler directories.

Project->Properties->Toolchain->ARM/GNU C Compiler->Directories->Include Paths



## 5.2.2 Step 2 – Select Correct Product

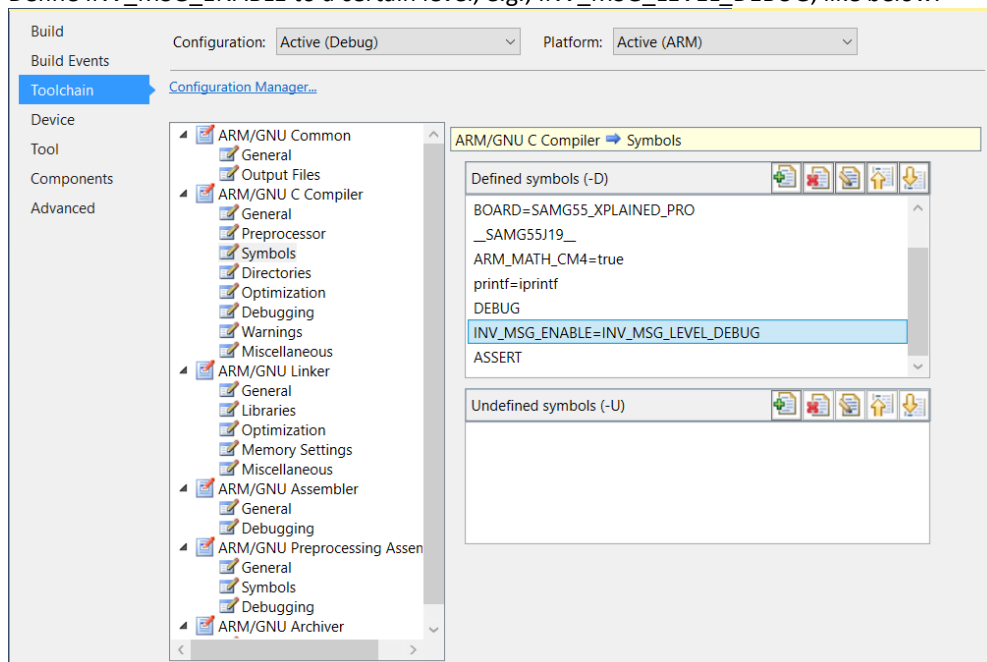
Set the corresponding IAM-20680/IAM-20680HT/IAM-20680HP macros to 1 in lam20680Product.h:

```
#define IAM20680 0 /* Set to 1 if sensor connected is IAM-20680, else 0 */
#define IAM20680_HP 0 /* Set to 1 if sensor connected is IAM-20680HP, else 0 */
#define IAM20680_HT 1 /* Set to 1 if sensor connected is IAM-20680HT, else 0 */
```

## 5.2.3 Step 3 – Implement MSG\_PRINTER function

The example application uses INV\_MSG helper functions to output debug traces to UART console. If you would like to use this utility, you need to:

1. Define INV\_MSG\_ENABLE to a certain level, e.g., INV\_MSG\_LEVEL\_DEBUG, like below:



2. Implement msg\_printer function.

```
static void msg_printer(int level, const char * str, va_list ap)
```

3. Call INV\_MSG\_SETUP in your init sequence in main().

```
INV_MSG_SETUP(INV_MSG_ENABLE, msg_printer);
```

## 5.2.4 Step 4 – Implement READ/WRITE/SLEEP functions

These functions are needed by the driver and must be implemented based on your platform.

Below are examples of these function implementations provided in this sw release package:

```
static int idd_io_hal_read_reg(void * context, uint8_t reg, uint8_t * rbuffer, uint32_t rlen) {
 (void)context;

 #if (SERIF_TYPE_SPI == 1)
 return spi_master_read_register(NULL, reg, rbuffer, rlen);
 #elif (SERIF_TYPE_I2C == 1)
 return i2c_master_read_register(I2C_Address, reg, rlen, rbuffer);
 #endif
}

static int idd_io_hal_write_reg(void * context, uint8_t reg, const uint8_t * wbuffer, uint32_t wlen) {
 (void)context;

 #if (SERIF_TYPE_SPI == 1)
 return spi_master_write_register(NULL, reg, wbuffer, wlen);
 #elif (SERIF_TYPE_I2C == 1)
 return i2c_master_write_register(I2C_Address, reg, wlen, wbuffer);
 #endif
}

/* Sleep implementation for IAM20680 */
void inv_iam20680_sleep_us(int us) {
 delay_us(us);
}

/* Sleep implementation for IAM20680 */
void inv_iam20680_sleep_ms(int ms) {
 delay_us(ms);
}
```

The serial interface read/write function pointers need to be registered with driver by being assigned to `inv_iam20680_serif` structure, which is part of the device driver states structure `inv_iam20680_t`.

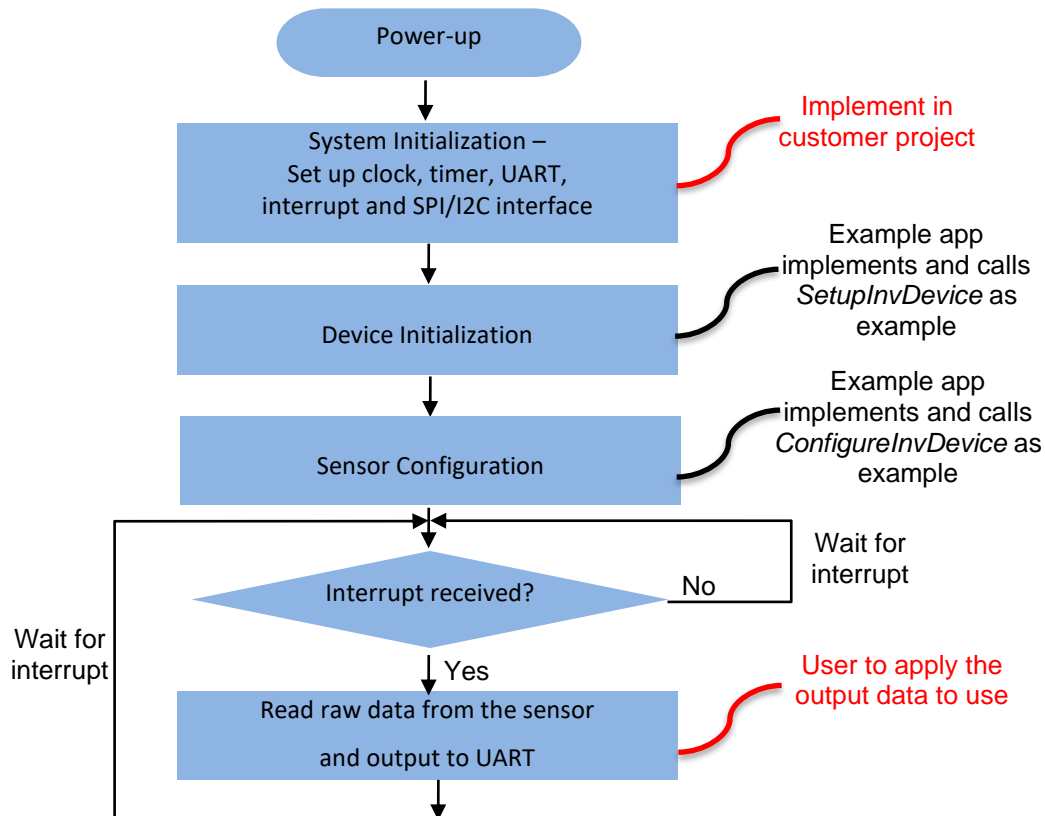
The example code to pass these function pointers can be found in SetupInvDevice function in the EMD-App\src\IAM20680\example-xxx.c files.

```
/*!
 * \brief Set up the InvenSense device
 * \param[in] read_reg Function pointer for reading register
 * \param[in] write_reg Function pointer for writing register
 * \param[in] isSPI 1: using SPI interface; 0: using I2C interface
 */
int SetupInvDevice(int (*read_reg)(void * context, uint8_t reg, uint8_t * buf, uint32_t len),
 int (*write_reg)(void * context, uint8_t reg, const uint8_t * buf, uint32_t len),
 inv_bool_t isSPI);
```

The sleep functions do not need to be registered. They will be used by the driver directly to implement delays.

### 5.2.5 Step 5 – Implement the system initialization flow

The diagram below shows the entire workflow for example-raw-ag application. User will need to implement all the system initialization functions for their platform.



### 5.3 INTEGRATION CHECKLIST

A list is provided here to make it easy for user to checkoff what is done and what needs to be done:

- ☐ Select product in `Iam20680Product.h`
- ☐ Implement `msg_printer` function if `INV_MSG` helper functions are needed
- ☐ Implement serial interface read/write functions and register with driver
- ☐ Implement sleep functions
- ☐ Implement system functions:
  - ☐ UART console configuration function, for outputting debug traces
  - ☐ External interrupt initialization, for receiving interrupts, like data ready interrupt and WoM interrupt
  - ☐ SysTick Timer handler to get accurate timestamps and a ring buffer to store the timestamps for sensor data
- ☐ Device/sensor initialization and configuration

## 6 DOCUMENT INFORMATION

### 6.1 REVISION HISTORY

| REVISION | DATE             | DESCRIPTION                                                                                                        | AUTHOR                 |
|----------|------------------|--------------------------------------------------------------------------------------------------------------------|------------------------|
| 0.1      | October 13, 2017 | Initial version.                                                                                                   | Rajesh Bisoi           |
| 0.2      | October 17, 2017 | Revised version.                                                                                                   | Rajesh Bisoi/Qing Wang |
| 0.3      | October 24, 2017 | Revised version.                                                                                                   | Rajesh Bisoi/Qing Wang |
| 0.4      | October 25, 2017 | Revised version - SPI support.                                                                                     | Rajesh Bisoi           |
| 0.6      | July 16, 2019    | Revised version – Fusion algorithm support                                                                         | Sai Gummitha           |
| 0.7      | October 2, 2019  | Revised version – Creating new document with algorithm support and removing algorithm reference from this document | Sai Gummitha           |
| 0.8      | July 24, 2020    | Added 20680HP support                                                                                              | Andrew Muir            |
| 0.9      | Aug 12, 2021     | Added 20680HT support                                                                                              | Qing Wang              |
| 0.10     | Aug 17, 2021     | Added porting guide                                                                                                | Qing Wang              |

Table 2 Revision History