Embedded Motion Driver 20x48 – Users Guide

Table of Contents

1 Revision History

Revision Date	Revision	Description
02/01/2017	1.0	Initial Release

2 Purpose

The Embedded Motion Driver is an embedded software stack of the sensor driver layer that easily configures and leverages many of the features of InvenSense motion tracking solutions. Invensense release of the eMDs implements several sensor fusion algorithms for customers to use such as Quaternion generations and gesture detections. Detailed descriptions of the features are in section 7 Features Overview.

The eMD is designed as a solution which can be easily ported to most MCUs. With the release of the eMD 20X48 it includes a complete 9-axis solution with sensor fusion, calibration, and common gesture detection all running on the DMP. The eMD release package contains an example project built on top of a bare metal ST Discovery evaluation aboard with an ARM M4.

This document details how to set up the hardware and get the default projects up and running. It is recommended as a good way to understand the Motion Driver algorithms, DMP, and hardware features.

3 Release Package

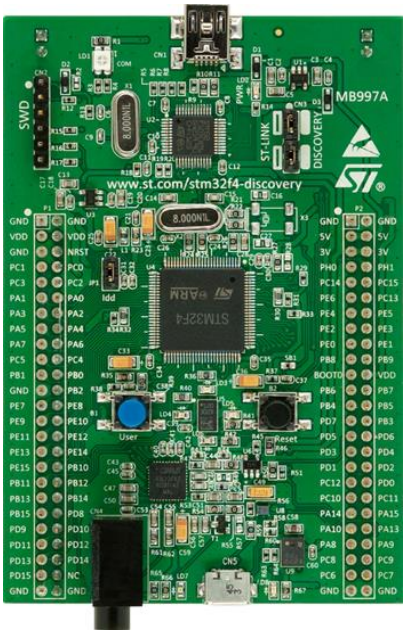
eMD 20x48 release package contains an example project of the ICM20648 and ICM20948 using an STM ARM Cortex M4 core. It was developed using a STM32F4 Discovery Board and IAR ARM Workbench Toolchain and compiler. The package's more important contents are highlighted in the following….

- **…\emd-ICM20648-20948.rc1\embedded-motion-driver\EWARM\STM32F4_eMD-VDM.eww :** IAR ARM Project file
- **…\emd-ICM20648-20948.rc1\embedded-motion-driver\User** : Contains the main.c and main.h
- **…\emd-ICM20648-20948.rc1\embedded-motion-driver\Mems\ :** Invensense specific driver codes
- **...\emd-ICM20648-20948.rc1\embedded-motion-driver\EWARM\ :** example Project file using IAR tool ARM embedded workbench toolchain
- **…\emd-ICM20648-20948.rc1\\pyConsoleScipts-rolldice\ :** A python script which enables a 3D model of a cube that takes in quaternions. The cube is used as a way to help validate the quaternions generated from the system.

4 STM32F4 (Cortex-M4) Discovery Board Project

4.1 Hardware Requirements

- STM32F4-Discovery Board Evaluation Board Kit with STM32F407VG MCU (purchasable through DigiKey, Mouser, etc…)



- FTDI UART module x 2 (we used UB232R this one)
  - http://www.digikey.com/product-detail/en/ftdi-future-technology-devices-international-ltd/UB232R/768-1022-ND/1836400?WT.srch=1&gclid=CKOzzIeq_NECFUlNfgods1MJsw

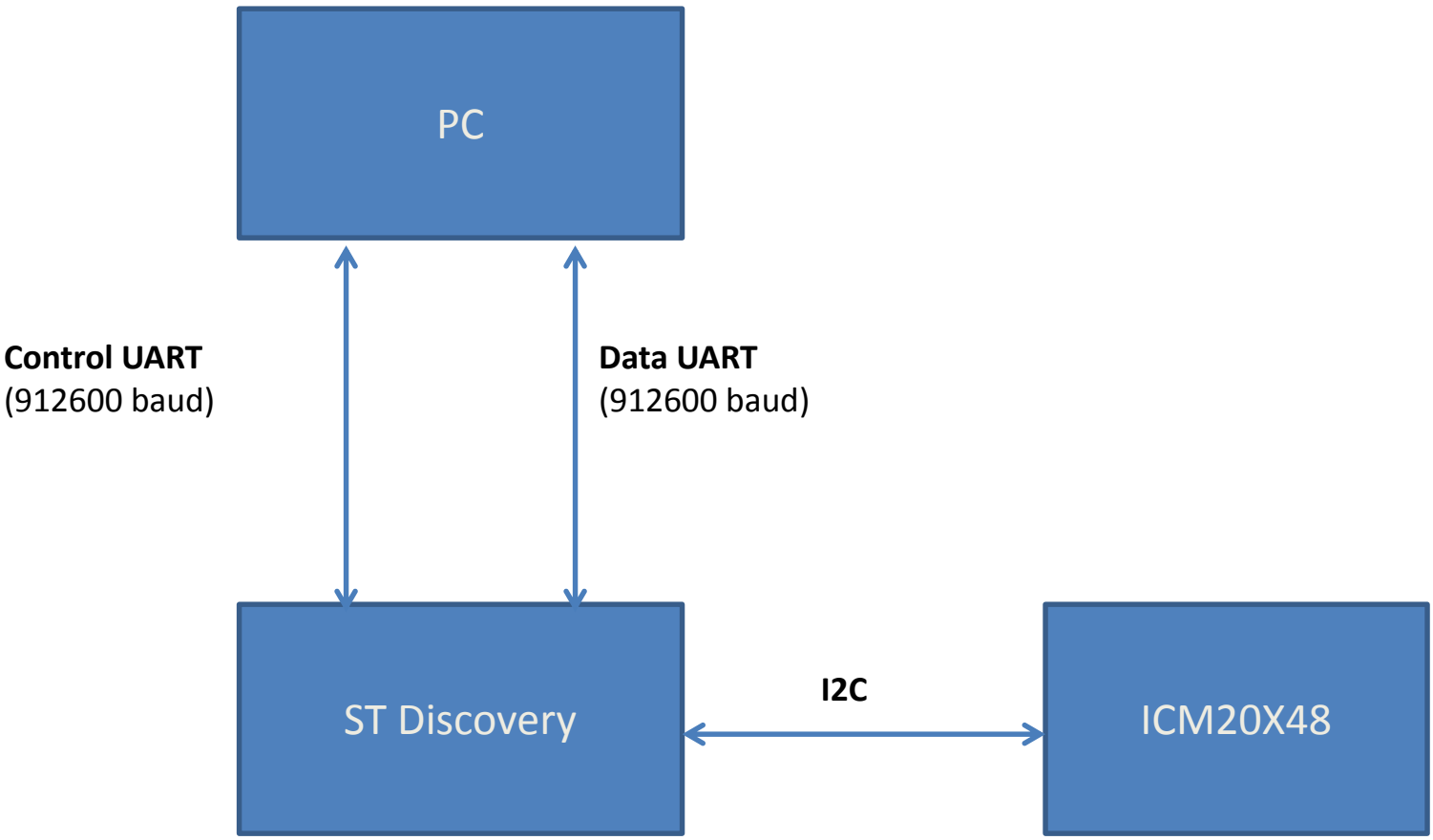
- InvenSense evaluation boards for ICM20648 (6-axis accel and gyro) or ICM20948 (9-axis accel, gyro, and mag)

- ICM20948 1.8V power supply – ICM20948 VDDIO requires 1.8V which Discovery Board does not supply (later ST MCU boards like the Nucleo does supply this current). Unfortunately customers will need an external supply in order to use the full 9-axis

4.2 Software Requirements

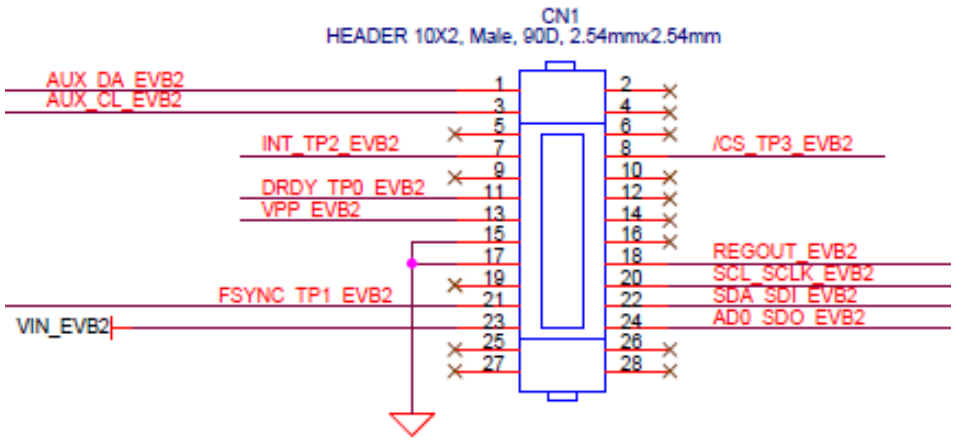
- IAR Embedded Workbench 7.3 or higher version
- ST-LINK USB Driver for debugging
- Terminal Application like TeraTerm – used for inputting commands and seeing output logs like sensor data
- Python 2.7 + pySerial + pyGame – a PC 3D model of a cube that takes in quaternions from the Invensense eMD project through the UART/USB output of the ST Discovery Board

4.3 Connecting the Hardware



- **InvenSense ICM20648/ICM20948 Eval Board connection to Discovery Board**

The connection from the InvenSense eval board to the discovery board will require wiring between the two PCB boards. The InvenSense eval board pin outs are all similar



To connect to the Discovery Board you will need to connect these 5 pins on the Invensense EVB. Most Invensense EVBs do not populate the pins 1-4 and 25-28.

EVB Header Pin Number	Description	Discovery Board GPIO Pin Number
7	INT output	PA1
17 or 15	GND	GND
23	VCC_IN	EXT_3V
20	I²C SCL	PB10
22	I²C SDA	PB11

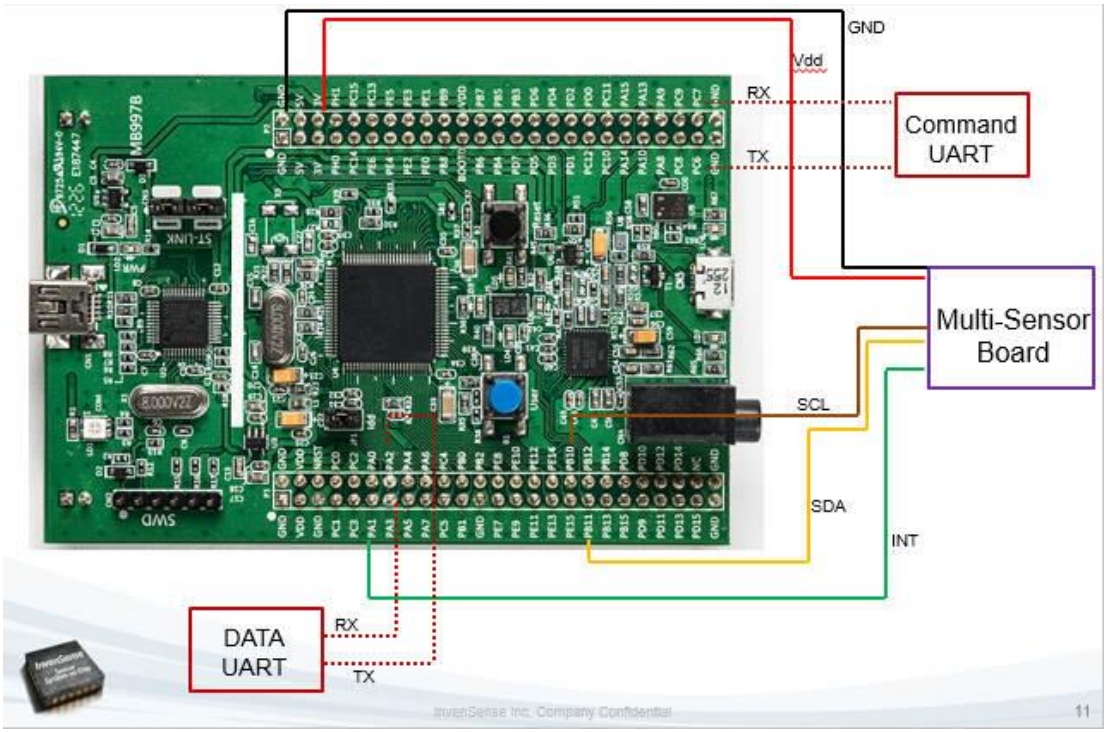
In addition, if you are using the ICM20948 you need an external 1.8V supply

5	1.8V supply	NA - needs external source

- **Discovery Board UART Output**

The eMD outputs data through 2 UART outputs. 1 UART is the Command UART which takes in command inputs, and outputs debug logs and data. The 2nd UART is the Data UART and primary used to output quaternions for the Python Cube. The pins are

Discovery Board UART Out Pin Number	Description
PA2	Data UART Tx (pin 8 if using UB232R)
PA3	Data UART Rx (pin 7 if using UB232R)
PC6	Command UART Tx (pin 8 if using UB232R)
PC7	Command UART Rx (pin 7 if using UB232R)



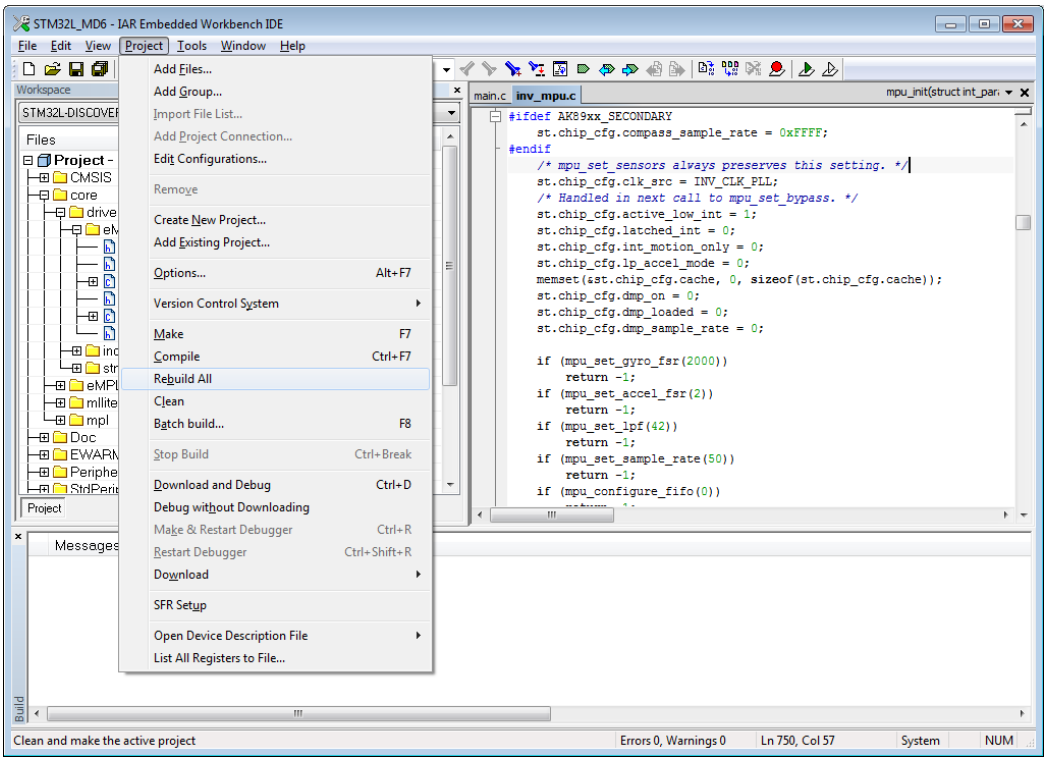
4.4 Setup

- **Open IAR IDE and load project -** Double click on the IAR ARM project file to automatically open the workspace in IAR ARM compiler. Project file is under the directory

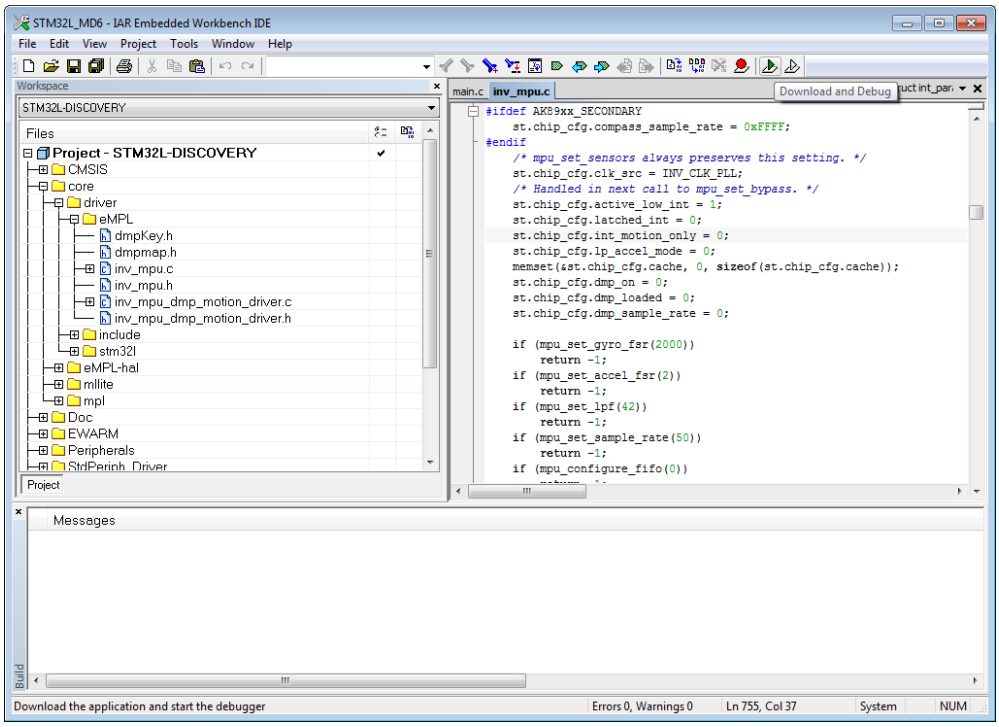
  …\embedded-motion-driver\EWARM\STM32F4_eMD-VDM.eww

- **Configure Invensense target chip in Compiler Options in the project**
  - Right click on the project and goto “C/C++Compiler Category” and “Preprocessor” tab

- Under “Defined symbols” box the “MEMS_20648” is defined. Do not change this define even if you are using ICM20948. The ICM20948 is the ICM20648 with a compass attached to its auxiliary I2C.
- If ICM20948 define the following compiler symbols, if ICM20648 please remove. (Technically if you have external AKM compass connected to the ICM20648 AUX I2C you can define these and it will act like a ICM20948, however for simplicity right now we will assume it is standalone ICM20648)
  - “MEMS_SECONDARY_DEVICE”
  - “MEMS_AUGMENTED_SENSORS”
- Configure Basic Activity Classification (BAC) to either wearable mode (on wrist) or mobile (phone) if using this feature
  - If device is more like wrist worn, then define “MEMS_WEARBLE_DEVICE” in compiler symbols
- Configure Compass Device and I2C Address is needed.
  - In main.c search for “COMPASS_SLAVE_ID” and select the appropriate device. For ICM20948 change to “AK09916” since it is using an internal AKM9916 part.
  - In main.c search for “COMPASS_CHIP_ADDR”. If ICM20948 or AK09916 change to “0x0C”. Other AKM compass devices use address “0x0E”.
- Configure Mounting Matrix for accel/gyro and compass in main.c
  - The mounting matrix is the chip frame to body frame transformation. There is a mounting matrix for accel/gyro and a separate mounting matrix for the compass.
  - Search “ACCEL_GYRO_ORIENTATION” for accel and gyro
  - Search “COMPASS_ORIENTATION” for compass
- Select the ‘Project’ pull down menu and ‘Rebuild All’

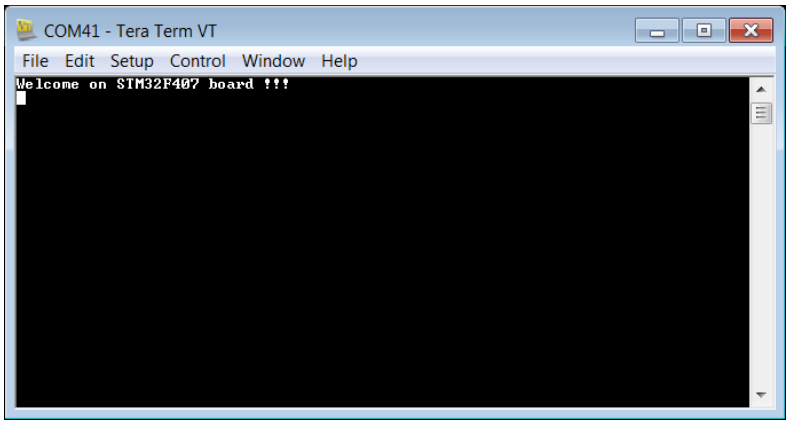


- With the hardware connected, hit the ‘Download and Debug’ ICON

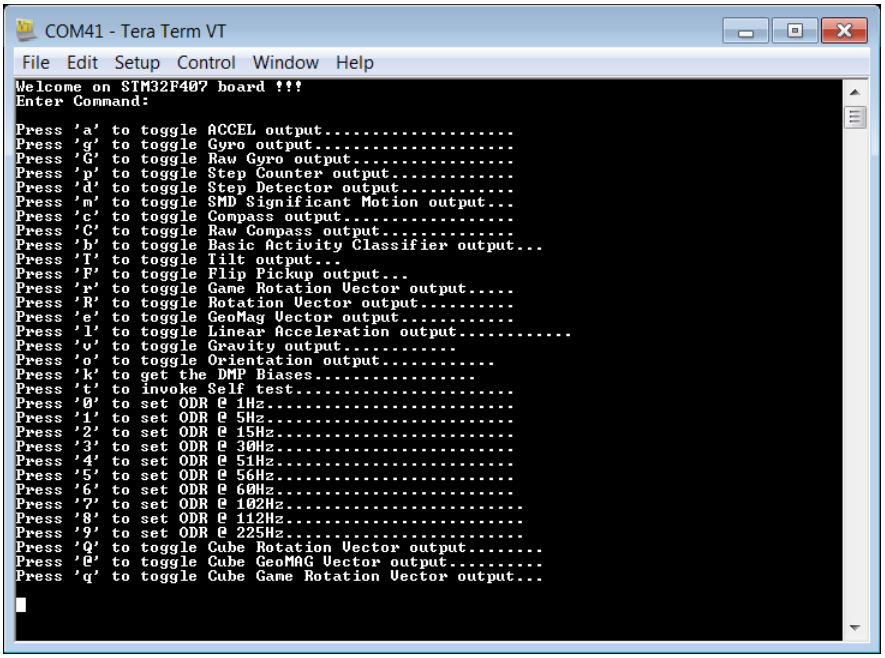


5 Setting up the UARTs

- Connect Data and Command UART to PC
- Let UART enumerate and note COM port numbers
- Open up TeraTerm or similar serial terminal application. The serial port setup should be this
  - Baud rate : 912600
  - Data : 8 bit
  - Parity : None
  - Stop : 1 bit
  - Flow Control : None
- If everything is good, when you reset the ST Discovery Board you should see this message on the Data UART output



- If you enter 'h' on the Data UART you will get a list of possible input commands



- You can then enter different commands into the DATA UART to get the data you want
- If you enter command 'Q', '@', and 'q'…these commands will output quaternion through the Command UART which customers can use the python cube.

6 Python Client

A python client is included with the release package to test the performance of the quaternions generated by the Invensense driver. The client can be found in the release package under the directory

..\pyConsoleScipts-rolldice\

The python client takes in quaternions through the Data UART and rotates the 3D cube model according to how the ICM20X48 moves. To use the pycube you would need to install Python, pyserial, and pygame for the python script to execute.

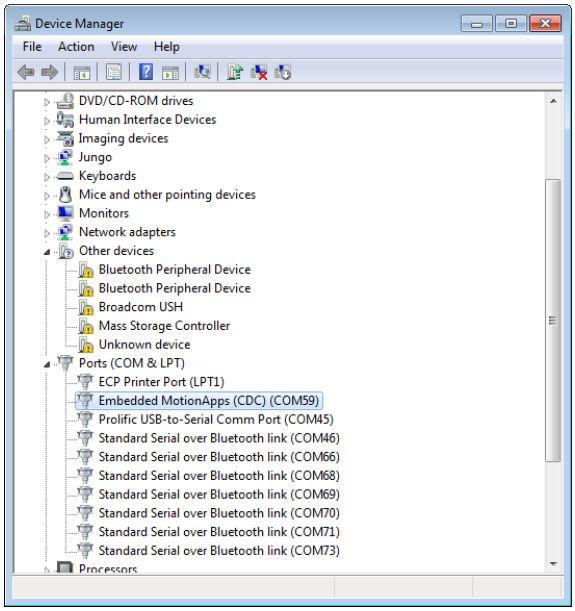
- Installing Python 2.7 (32-bits version) or above, pyserial, and pygame

  Python: https://www.python.org/downloads/
  Pyserial: https://pypi.python.org/pypi/pyserial
  Pygame: http://www.pygame.org/download.shtml

  For your convenience we downloaded the required executables and also install instructions in the folder

  ..\ pycube_install

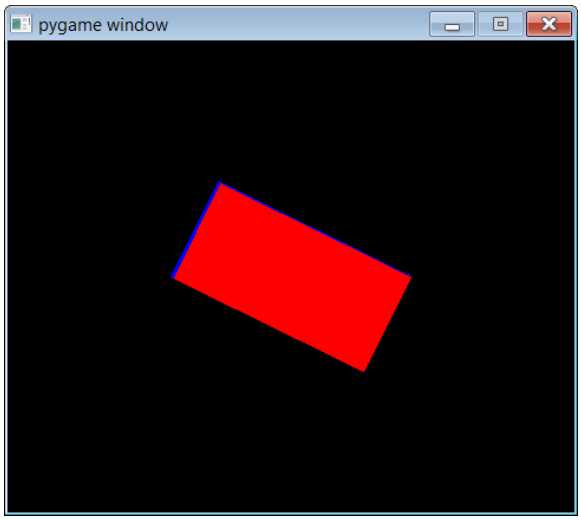
  Connect your flashed and working hardware to your PC and find the COM port in the device manager



- Start the python client by opening up a command prompt window and browse to the python client directly and enter the following command

  - python motion-driver-client.py<COM PORT NUMBER>

- A window will appear that displays a 3D Cube which corresponds with the quaternion angles outputted from the device.



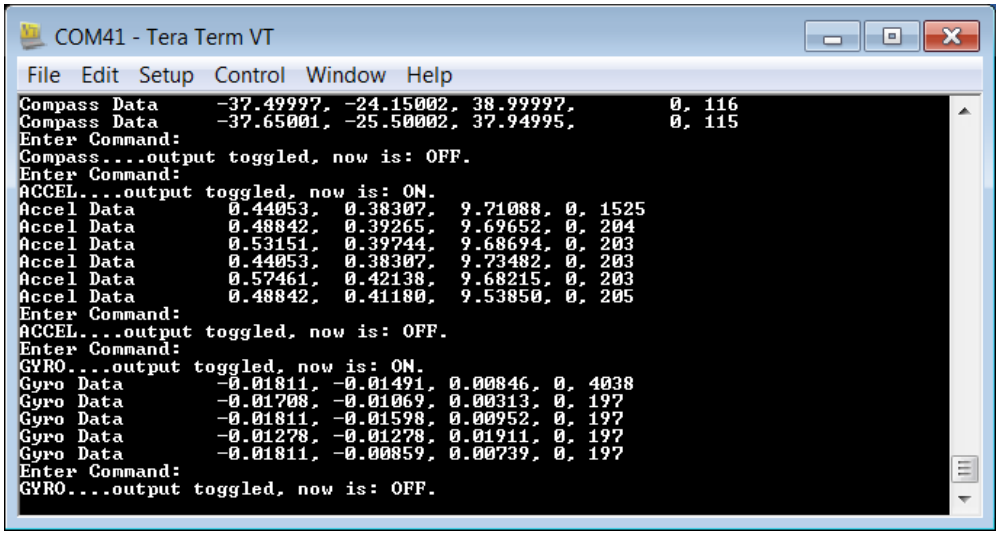
7 eMD 20X48 Features Overview

The feature menu for ICM20648/20698 is as follows:

- Press 'a' to toggle ACCEL output.....
  - Outputs calibrated accel data in m/s^2
- Press 'g' to toggle Gyro output.....
  - Outputs calibrated gyro data in radians
- Press 'G' to toggle Raw Gyro output.....
  - Outputs non-calibrated gyro data (raw) in radians
- Press 'p' to toggle Step Counter output.....
  - Prints out a pedometer step count. When enable for first time after power on, requires 8 continuous steps registered before outputting
- Press 'd' to toggle Step Detector output.....
  - Prints “Step Detected>>>>>>” is a step motion is triggered
- Press 'm' to toggle SMD Significant Motion output...
  - Prints “>> SMD Interrupt *********” if a significant motion of the device is detected
- Press 'c' to toggle Compass output.....
  - Outputs calibrated compass data (if applicable) in Tesla units
- Press 'C' to toggle Raw Compass output.....
  - Outputs non-calibrated compass data along with compass calculated biases
- Press 'b' to toggle Basic Activity Classifier (BAC) output...
  - Detect the following activities and prints it to the output
    - Standing, walking, running, biking and drive

  - Activity detection can be more accurate if configured to be a wearable device (on wrist) or mobile (in hand) … see section 4.4 Setup on how to do this
- Press 'T' to toggle Tilt output...
  - Outputs when a tilt is detected and when a tile detection ends
- Press 'F' to toggle Flip Pickup output...
  - Detects and outputs a flip or a pickup motion
- Press 'r' to toggle Game Rotation Vector output.....
  - Outputs 6-axis quaternions based on accel and gyro to the Command UART
- Press 'R' to toggle Rotation Vector output.....
  - Outputs 9-axis quaternions based on accel, gyro, and compass to the Command UART
- Press 'e' to toggle GeoMag Vector output.....
  - Outputs 6-axis quaternions based on accel and compass to the Command UART
- Press 'l' to toggle Linear Acceleration output.....
  - Outputs device linear acceleration (acceleration minus gravity) in m/s^2
- Press 'v' to toggle Gravity output.....
  - Outputs on which accel axis the gravity is affecting the device in m/s^2
- Press 'o' to toggle Orientation output.....
  - Outputs Euler angles
- Press 'k' to get the DMP Biases.....
  - Prints out the accel and gyro most recent biases calculated from the DMP
- Press 't' to invoke Self test.....
  - Invokes a test on the sensors to determine if MEMS hardware is still functioning correctly. The Self Test is mainly for production line testing to determine before final assembly if sensors are still working. The self test will
    - Give a “PASS” or “FAIL” on accel, gyro, and compass of the self test
    - Executes a factory calibration of the accel and gyro and apply the biases to the sensor registers
- Press '0' to set ODR @ 1Hz.....
- Press '1' to set ODR @ 5Hz.....
- Press '2' to set ODR @ 15Hz.....
- Press '3' to set ODR @ 30Hz.....
- Press '4' to set ODR @ 51Hz.....
- Press '5' to set ODR @ 56Hz.....
- Press '6' to set ODR @ 60Hz.....
- Press '7' to set ODR @ 102Hz.....
- Press '8' to set ODR @ 112Hz.....

- Press '9' to set ODR @ 225Hz.....
    - Commands 0-9 sets the ODR (Output Data Rate) to be the selected frequency
- Press 'Q' to toggle Cube Rotation Vector output.....
- Press '@' to toggle Cube GeoMAG Vector output.....
- Press 'q' to toggle Cube Game Rotation Vector output...
    - Commands ‘Q’, ‘@’, ‘q’ are commands which outputs the quaternions to the DATA UART. This is mainly for the pycube or other applications to use.



- Sensor outputs will contain the sensor data but most will also contain an Accuracy Flag and also Timestamp
    - Accuracy Flag – single digit number after the sensor data. Ranges are usually from 0 to 3 where 3 means sensor is calibrated and most accurate and 0 means not calibrated and strictly raw data
    - Timestamp – the millisec from the previous request for data

8 The DMP

The Digital Motion Processor, or DMP, is Invensense’s proprietary embedded lightweight processor designed specifically to handle sensor fusion and advanced gesture recognition. The DMP offloads functionality from the main processor to decrease overall system power.

The DMP is the main core of the features of the ICM20x48. All sensor fusion data is generated on board the ICM20x48 through the DMP as well as gesture recognition. The DMP also provides dynamic calibration of all the sensors including compass.

The eMD ICM20x48 release provides a DMP image for customers. The DMP image is about 16Kb and can be easily downloaded through to the ICM20x48 through I2C or SPI. The features of the ICM20x48 DMP includes

ICM20X48 DMP Features	Description
Pedometer	Step Count and Step Detection
Significant Motion Detection	Used for wake up functionality
Pick Up	Detects when device is picked up
Bring To See	Detects motion of bringing a watch-like device to see
Tilt	Detects when device tilts past a threshold
Basic Activity Classifiers	Detects Run, Walk, Bike, Stand, and Drive activity
6-Axis Sensor Fusion	Accel and Gyro quaternion generation
9-Axis Sensor Fusion	Accel and Gyro and Compass quaternion generation
Dynamic Accel Cal	A constant and in use calibration of accel. Reduces errors due to wear and tear
Dynamic Gyro Cal	A constant and in use calibration of gyro. Reduces errors due to temperature and reference drift
Dynamic Compass Cal	A constant and in use calibration of compass. Reduces errors due to magnetic variations

9 Porting the Code

When customers are to the point in which they want to port the eMD ICM20x48 into their own processor and system, the 4 basic functions which requires porting are

- I2C Serial Communication to the ICM20x48
- Timer (1ms resolution) – for timestamp and wait function
- Interrupt Handling – callback function needs to be called when interrupt comes in from the chip
- UART – host interface, data output, debug

9.1 Serial Communication

The example project uses I2C to communicate from the MCU to the ICM20x48. When porting the code customers will need to establish their own communications between the 2 components. The I2C interface hook between the Invensense drivers to the MCU is modularized in these files and functions…

- Files
  - \embedded-motion-driver\Mems\common\inv_mems_drv_hook.h
  - \embedded-motion-driver\Mems\common\inv_mems_drv_hook.c
- Write
  - int inv_serial_interface_write_hook(uint16_t reg, uint32_t length, uint8_t *data)
- Read
  - int inv_serial_interface_read_hook(uint16_t reg, uint32_t length, uint8_t *data)

9.2 Timer

The timer is mainly used for timestamps and to add delays in the firmware.

- Files
  - \embedded-motion-driver\Mems\common\inv_mems_drv_hook.h
  - \embedded-motion-driver\Mems\common\inv_mems_drv_hook.c
- Tick (1ms resolution)
  - long inv_get_tick_count()
- Wait function (1ms resolution)
  - void inv_sleep(unsigned long nTime)
  - void inv_sleep_100us(unsigned long nHowMany100MicroSecondsToSleep)

9.3 UART

If customers will be using the UART, the UART will need to be initialized by the toolchain customers use. Once initialized, the interface to output to the UART are in the following functions.

- Files
  - embedded-motion-driver/User/inc/main.h
  - embedded-motion-driver/User/src/main.c
- Command console output
  - void print_command_console(char * str)
- Data console output
  - void print_data_console(char * str)

9.4 Interrupt Handling

The example project uses interrupts as a way for the ICM20x48 to indicate there are sensor data ready or a gesture was detected. It is recommended that customer implement an interrupt handler to process the data coming out of the DMP in a timely matter. The interrupt callback function is encapsulated here…

- Callback function
  - void gyro_data_ready_cb()

  - This needs to be called when interrupt comes in
- Files
  - embedded-motion-driver/User/src/main.c (callback function)

X-ON Electronics

Largest Supplier of Electrical and Electronic Components

Click to view similar products for Acceleration Sensor Development Tools ***category:***

Click to view products by TDK InvenSense ***manufacturer:***

Other Similar products are found below :

EVAL-ADXL350Z-S ADIS16201/PCBZ ADIS16260/PCBZ BRKOUT-FXLN8372Q BRKTSTBC-A8471 2019 EVAL-ADXL313-Z EVAL-ADXL343Z-M EVAL-ADXL343Z-S EVAL-ADXRS622Z BRKOUT-FXLN8362Q BRKOUT-FXLN8371Q ADISEVALZ EVAL-ADXL346Z EVAL-ADXL346Z-S STEVAL-MKI151V1 EVAL-ADXL350Z FRDM-K64F-AGM04 BRKTSTBC-A8491 FRDMKL25-A8491 FRDMKL25-A8471 FRDM-STBC-AGM04 KX224-I2C-EVK-001 FRDMSTBC-A8471 EVAL-ADXL372-ARDZ 101990281 1018 EVAL-ADXL362-ARDZ EVAL-KXCJ9-1008 1120 1231 1247 1413 DEV-13629 2020 ADXL213EB EVAL-ADXL343Z-DB EVAL-ADXL344Z-M EVAL-ADXL345Z-M EVAL-ADXL363Z EVAL-ADXL375Z-S EVAL-ADXRS623Z EVAL-ADXRS652Z EVAL-CN0274-SDPZ EV-BUNCH-WSN-1Z EV-CLUSTER-WSN-2Z STEVAL-MKI033V1 EVAL-ADXL344Z-DB EVAL-ADXL346Z-DB EVAL-ADXL363Z-MLP