# ICM-306xx Control Protocol & Architecture



InvenSense Inc.
1745 Technology Drive, San Jose, CA 95110 U.S.A
+1(408) 988–7339
www.invensense.com

Document Number: AN-000028
Revision: 1.20
Release Date: 01/20/2015

# TABLE OF CONTENTS





Document Number: AN-000028
Revision: 1.20
Release Date: 01/20/2015

## TABLE OF TABLES



## TABLE OF FIGURES

# FIFO Control Protocol and Architecture Specification

## 1. TOP LEVEL

### OVERVIEW

Three different processors are available as part of ICM-306XX— an ARM-M0, the DMP3, and the DMP4. The DMP3 is used for quick deployment, and based on the Android Lollipop ICM20645 solution. This DMP image has 6-axis (Accelerometer + Gyroscope) support including pedometer, Automatic Activity Recognition, Game Rotation Vector, gyroscope calibration and accelerometer calibration.

The DMP4 is a higher functioning processor with specialties in fixed point Q30 processing and 16-bit FFT. The DMP4 is register based; there are HW binary semaphores available on the DMP4 and the ARM for inner-process communication and OS support.

### 1.1 FRAMEWORK FEATURES

The framework running on the M0 will handle decoding the FIFO controls and sensor control.

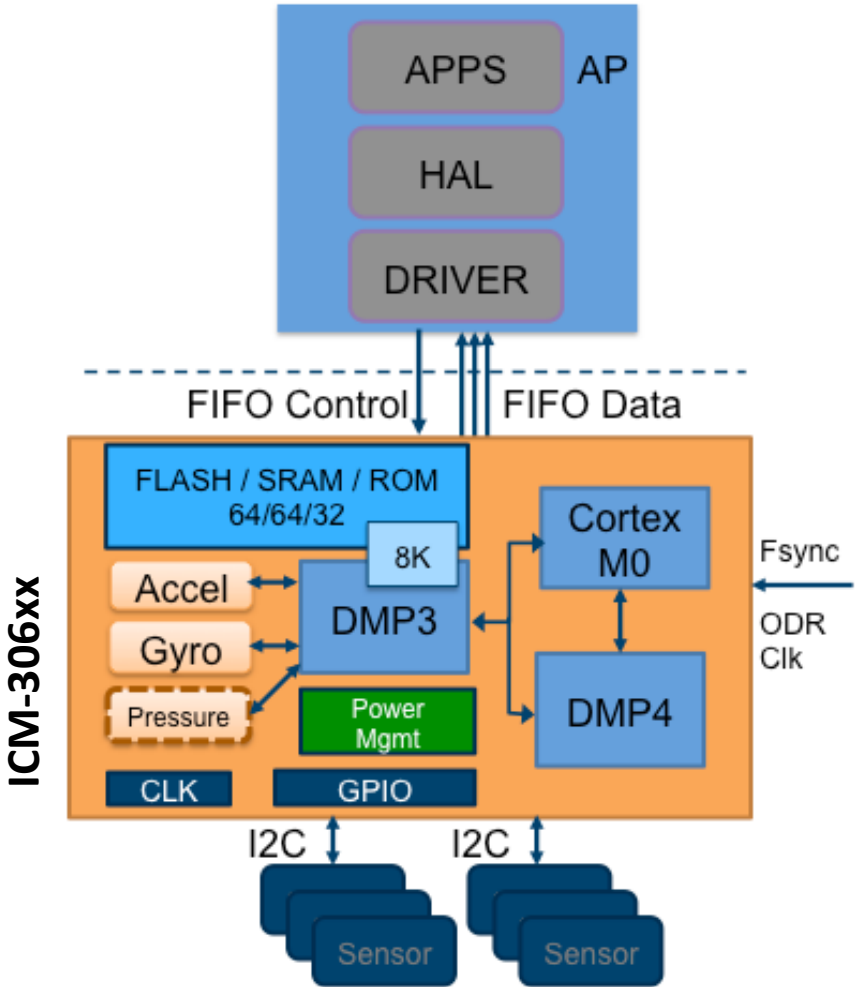


**Figure 1. ICM-306XX Architecture**



Document Number: AN-000028
Revision: 1.20
Release Date: 01/20/2015

## 1.2 RECOMMENDED HW LAYOUT

It is recommended that the AP communicates to the ICM-306xx through SPI. SPI provides a faster communication pathway compared to I2C. It is best to have an interrupt line from ICM-306XX connected to a wake-up pin on the AP. The interrupt line signals wake-up events from Android L sensors, such as significant motion detection and wake-up accelerometer. An interrupt line from ICM-306XX connected to a non-wake-up pin on the AP is also recommended. This line handles the non-wakeup sensors in Android L, and allows the AP to sleep if it is already asleep, as well as saving power. Other sensors connect to ICM-306XX using I2C. These sensors include, but are not limited to, magnetometer, proximity sensor, light sensor, and barometer. The system 32.768 KHz clock available on most phone platforms can connect directly to ICM-306xx, preventing clock drift, and allowing for accurate timestamps between the AP and ICM-306XX . This is strongly recommended and the default software is configured to work best with this connection. The ICM-306XX does not require a HW interrupt line from the AP, as it supports a Serial Interface (SPI) wake-up interrupt in the form of a Scratch Interrupt—the AP can wake the ICM-306XX device by performing a SPI/I2C write to the M0 Scratch Interrupt. The ICM-306XX supports three reset methods:

1. POR (Power On Reset): Requires a power cycle to reset the design.
2. RESETL: Active low reset input pin.
3. Soft Reset: Provided by the Host via a ICM-306XX serial interface write. Only resets a subset of the POR and RESETL resets.

For an external reset, it is preferred to use the RESETL (Active low reset input pin) as it provides the Host with the most control. If RESETL is not used, it should be pulled high and the internal POR will provide a HW reset upon application of power.

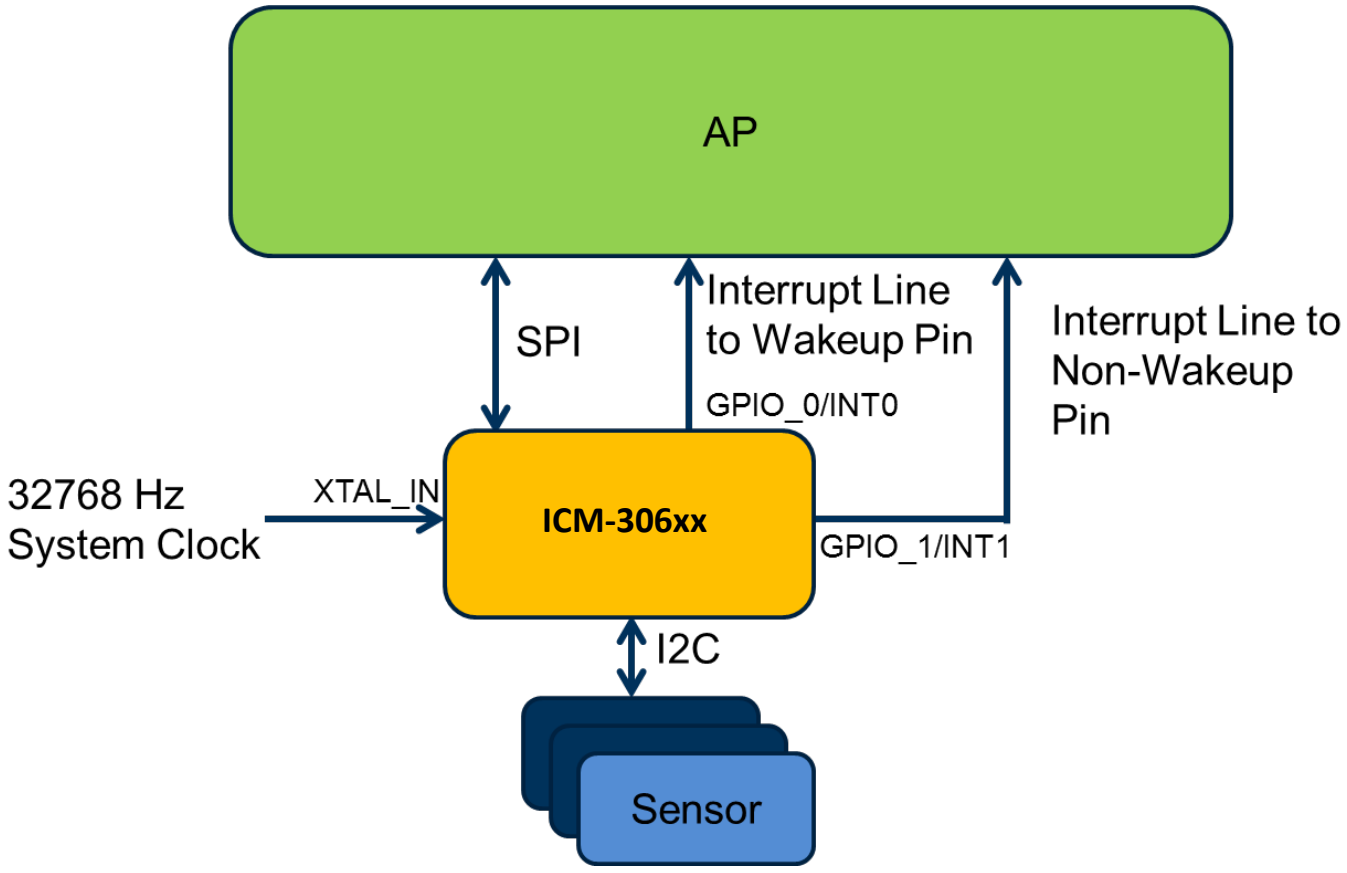


**Figure 2. Recommended HW Layout**



Document Number: AN-000028
Revision: 1.20
Release Date: 01/20/2015

### 1.3 MCU ARCHITECTURE

The main control for the MCU happens by sending commands into FIFO 1, and receiving output through FIFO 0.

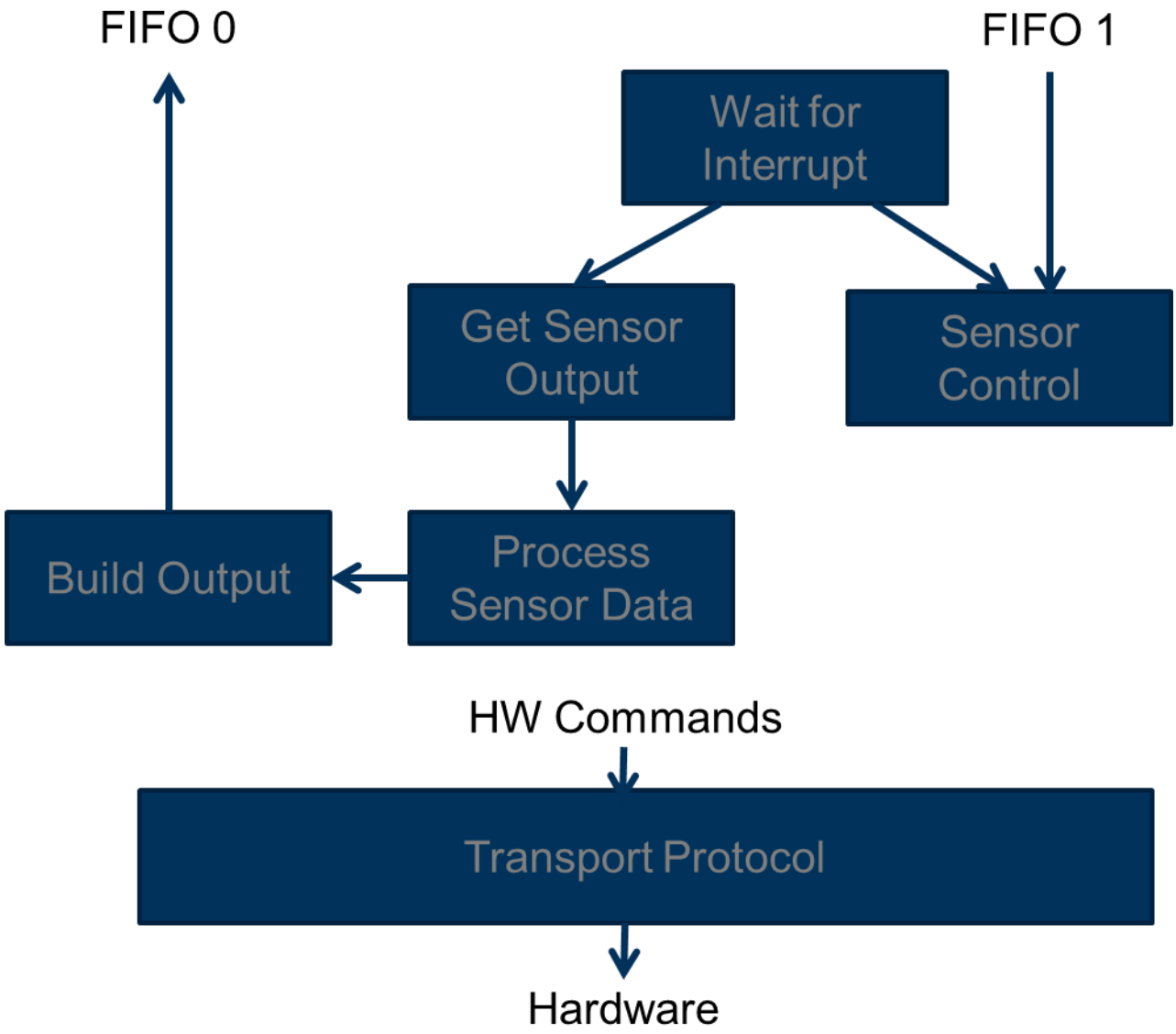


**Figure 3. Protocol Flow**



Document Number: AN-000028
Revision: 1.20
Release Date: 01/20/2015

## 1.4 TRANSPORT AND HARDWARE ABSTRACTION AND PORTING

This covers the "Transport Protocol" block in the Protocol Flow Figure 3. A hardware layer needs to be ported for a platform. There are basic functions to read/write registers on the MCU (ICM-306XX) and read/write registers on the MEMS part.

TBD : Add SPI read/write for single transactions. Add SPI read/write based upon a timer. Add SPI read/write based upon interrupt

| BASE FUNCTIONS FOR TRANSPORT. C | |
|---|---|
| **FUNCTION NAME** | **DESCRIPTION** |
| *inv_error_t inv_write_reg(unsigned short reg, int length, unsigned char *data)* | Write to DMP4/ARM (ICM-306XX) Silicon registers |
| *inv_error_t inv_read_reg(unsigned char reg, int length, unsigned char *data)* | Read from DMP4/ARM (ICM-306XX) Silicon registers |
| *inv_error_t inv_write_mems_reg(unsigned char reg, int length, unsigned char *data)* | Write to DMP3/ARM Silicon registers |
| *inv_error_t inv_read_mems_reg(unsigned char reg, int length, unsigned char *data)* | Read from DMP3/ARM Silicon registers |

**Table 1. Basic Functions for Transport.c**

When the above functions are transported, then other functions that build upon them can be built as needed. These include methods to read/write to MEMS memory. It also includes functions to sleep and wakeup the MEMS part. Functions to read/write the FIFO on the MCU are also needed.

| ADDITIONAL AND SPECIALIZED FUNCTIONS | |
|---|---|
| **FUNCTION NAME** | **DESCRIPTION** |
| *inv_error_t inv_write_mems(unsigned short reg, int length, unsigned char *data)* | Write to DMP3/MEMs SRAM |
| *inv_error_t inv_read_mems(unsigned short reg, int length, unsigned char *data)* | Read from DMP3/MEMs SRAM |
| *inv_error_t inv_sleep_mems()* | Puts DMP3 into the lowest power state. Assumes sensors are all off. |
| *inv_error_t inv_wakeup_mems()* | Wakes up DMP3 . |
| *inv_write_fifo(int fifo_num, int size, unsigned char *data)* | Writes to a ICM-306XX FIFO |
| *inv_read_fifo(int fifo_num, int size, unsigned char *data)* | Reads a ICM-306XX FIFO |

**Table 2. Additional and Specialized Functions**



Document Number: AN-000028
Revision: 1.20
Release Date: 01/20/2015

## 1.5 WAIT FOR INTERRUPT

This covers the "Wait for Interrupt" block in the Protocol Flow Figure 3. This block will wait for any interrupt and send off the appropriate action. Actions it can wait for include detecting an interrupt due to a command being sent to a FIFO. See 2.4 for details on the command FIFO protocol. It can wait on timers. It can also wait for a watchdog timer in case of errors. It can wait on interrupts from appropriate sensors and take appropriate action.

| FUNCTION NAME | DESCRIPTION |
|---|---|
| *int inv_wait_for_events(int event_mask)* | Waits in a low power state until given event happens. Waits for INV_INPUT_FIFO_EVENT, INV_DATA_READY_EVENT, INV_WATCHDOG, INV_CADENCE). Returns mask of which event(s) was triggered. |

**Table 3. Wait for Interrupt**

## 1.6 SENSOR CONTROL

This covers the "Sensor Control" block in the Architecture Figure 3. This section decodes the input control FIFO. See 2.4 for details on the command FIFO protocol. Its main job is to enable sensors including virtual sensors and gestures and to set data rates. Any newly defined virtual sensor would be added here.

Most sensors data rates work with a base clock and a divider. Because of issues with rounding, the lower control mechanisms also work with a divider. The lower level functions are:

| FUNCTION NAME | DESCRIPTION |
|---|---|
| *inv_error_t inv_set_odr(int sensor_number, int divide)* | Sets the output data rate for a sensor. This sets both the hardware and DMP firmware. In cases of conflicts between sensors, this function will determine resolution. Uses what has been logged as enabled to evaluate sensor rates. The sensor given in sensor_number is assumed to be enabled, even if the enable function has not been called.<br><br>**sensor_number** A number between 1 and 20 (inclusive) corresponding to Android sensors but excluding wake up sensors and meta data and excluding extended sensors.<br><br>**divide** A N+1 divider for the odr. 0=1125 Hz,1=1125/2 Hz,2=1125/3 Hz,3=1125/4 Hz, etc. -1 is a special case for 9000Hz. Note sometimes the data rate will be set higher.<br><br>**return** Non zero error code on error. |
| *inv_error_t inv_enable_sensor(int sensor_number, int enable)* | Enables or disables a given sensor.<br>**sensor_number** A number between 1 and 20 (inclusive) corresponding to Android sensors but excluding wake up sensors and meta data and excluding extended sensors.<br>**enable** Non-zero enables the given sensor. Zero disables given sensor.<br>**return** Non zero error code on error. |

**Table 4. Sensor Controls**

TBD A function that converts milliseconds to divider and splits off the wake up from the non-wakeup. Customer insert should probably go here also.



Document Number: AN-000028
Revision: 1.20
Release Date: 01/20/2015

# 2 AP AND M0 COMMUNICATION

## 2.1 OVERVIEW

The control of the IC is managed through a FIFO protocol instead of registers. The protocol is independent of the physical interface on the SOC and supports both SPI and $I^2C$. Using the FIFO for communication will remove race conditions and give a scalable interface. One of the goals is to have a protocol that is easy to implement on the AP's HAL and driver side. For this reason, the control protocol will mirror parts of the Android sensor definition.

## 2.2 ENDIANNESS

Data in the FIFO protocol is little endian. This endianness applies to all data (command and output) and to all FIFOs.



Document Number: AN-000028
Revision: 1.20
Release Date: 01/20/2015

## 2.3 SENSOR ID

The *Sensor ID* is an unsigned 8-bit value that defines the sensor to be controlled. These numbers should follow the Android specification for Sensor ID when possible.
For Wake-Up (WU) sensors, sensor ID correspond to their normal (N) sensor counterpart with the most significant bit set (0x80).
E.g.: If Normal sensor ID is ***N***, then Wake-Up sensor id is: ***N | 0x81***

- IDs between 0 and 31 are reserved for Android sensors.
- IDs between 32 and 35 are reserved for Customer extra sensors.
- IDs between 36 and 39 are reserved for INVN extra sensors.
- IDs above 128 are reserved for special system command.

| ID | SENSOR_TYPE | TRIGGER MODE | NORMAL / WAKE-UP |
|---|---|---|---|
| 0 | META_DATA | n/a | n/a |
| 1 | ACCELEROMETER | continuous | both |
| 2 | GEOMAGNETIC_FIELD / MAGNETIC_FIELD | continuous | both |
| 3 | ORIENTATION (*deprecated* \| compute in AP?) | continuous | both |
| 4 | GYROSCOPE | continuous | both |
| 5 | LIGHT | on-change | both |
| 6 | PRESSURE | continuous | both |
| 7 | TEMPERATURE (*deprecated*) | / | / |
| 8 | PROXIMITY | on-change | both |
| 9 | GRAVITY | continuous | both |
| 10 | LINEAR_ACCELERATION | continuous | both |
| 11 | ROTATION_VECTOR | continuous | both |
| 12 | RELATIVE_HUMIDITY | on-change | both |
| 13 | AMBIENT_TEMPERATURE | on-change | both |
| 14 | MAGNETIC_FIELD_UNCALIBRATED | continuous | both |
| 15 | GAME_ROTATION_VECTOR | continuous | both |
| 16 | GYROSCOPE_UNCALIBRATED | continuous | both |
| 17 | SIGNIFICANT_MOTION | one-shot | WU |
| 18 | STEP_DETECTOR | special | both |
| 19 | STEP_COUNTER | on-change | both |
| 20 | GEOMAGNETIC_ROTATION_VECTOR | continuous | both |
| 21 | HEART_RATE | on-change | both |
| 22 | TILT_DETECTOR | special | WU |
| 23 | WAKE_GESTURE | one-shot | WU |
| 24 | GLANCE_GESTURE | one-shot | WU |
| 25 | PICK_UP_GESTURE | one-shot | WU |
| 26-30 | Reserved for future Android sensors | - | - |
| 31 | ACTIVITY_CLASSIFIER | On-change | WU |
| 32-35 | Reserved for CUSTOMER specific sensors | - | - |
| 36-39 | Reserved for INVN custom sensors | - | - |
| TBD | Screen Rotation (not implemented) | TBD | TBD |
| 254 | Perform Self-Test | n/a | n/a |
| 255 | Platform Setup | n/a | n/a |

**Table 5. Sensors ID**



Document Number: AN-000028
Revision: 1.20
Release Date: 01/20/2015

## 2.4 FIFO INPUT PROTOCOL

### 2.4.1 Overview

This section covers how the AP communicates commands to the M0.
Everything is controlled through the input command FIFO. The AP writes to the command FIFO and the ARM-M0 on the sensor SOC will decode it. The command consists of 16-bits, broken up into 3 sections of a Sensor ID, Command, and Expansion.

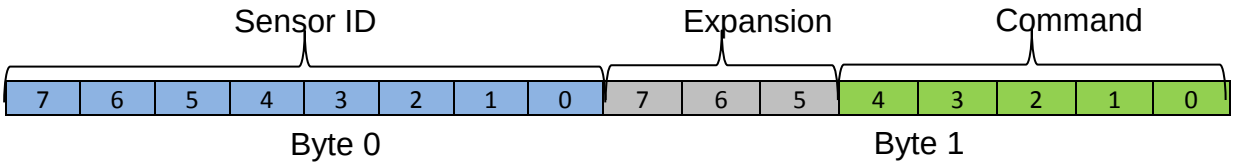


**Figure 4. Input Protocol Definition**

### 2.4.2 Platform Setup - Custom Codes

The *Platform Setup* command is a method to send implementation specific information to ICM-306XX. Examples of this would include how individual sensors or devices are mounted.
The data packet that follows is defined as:

| FEATURE | DESCRIPTION |
|---|---|
| TBD | TBD |

**Table 6. Platform Setup**



Document Number: AN-000028
Revision: 1.20
Release Date: 01/20/2015

### 2.4.3 Command

The *Command* is a 5-bit value that defines Sensor and Configuration commands. Some of these commands imply a data packet will follow.

| ID | COMMAND | DATA SIZE |
|---|---|---|
| 0 | Sensor off | 0 |
| 1 | Sensor On | 0 |
| 2 | Set Power | 1*uint16 = 2 bytes |
| 3 | Batch On | 1*uint16 = 2 bytes |
| 4 | Flush | 0 |
| 5 | Set Delay | 1*uint16 = 2 bytes |
| 6 | Set Calibration Gains | 9*int32 = 36 bytes |
| 7 | Get Calibration Gains | See 2.5.3 Data |
| 8 | Set Calibration Offsets | 3*int32 = 12 bytes |
| 9 | Get Calibration Offsets | See 2.5.3 Data |
| 10 | Set Reference Frame | 9* int32 = 36 bytes |
| 11 | Get Firmware Info | See 2.5.3 Data |
| 12 | Get data | See 2.5.3 Data |
| 13 | Get clock rate | See 2.5.3 Data |
| 14 | Ping | See 2.5.3 Data |
| 15 | Reset | 0 |
| 16-31 | Reserved | 0 |

**Table 7. Commands**

#### Sensor On / Off

Turns each sensor on or off.

| VALUE | MEANING |
|---|---|
| 0 | Sensor must be turned off |
| 1 | Sensor must be turned on |

**Table 8. Sensor On/Off**

#### Power

| VALUE | MEANING |
|---|---|
| 0 | Suspend |
| 2 | On |
| 3 | Idle |

**Table 9. Power**

**Batch On / Off**

Enables or disables sensor batching. Timeout in milliseconds, 0 means Off.



Document Number: AN-000028
Revision: 1.20
Release Date: 01/20/2015

**Delay**
Sampling rate in milliseconds.

**Flush**
Request an explicit flush for a sensor. A special data with this flush status flag needs to be placed in the output FIFO to signal when the flush command is over.

**Set Calibration Gains**
Calibration data used to adjust the sensor gains: for example, for a 3-axis Sensor, the command is followed by *9*int32* (3x3 fixed-point matrix in Q30 format).

**Set Calibration Offsets**
Data used to calibrate the Sensor offsets: for example, for a 3-axis Sensor, the command is followed by *3*int32* (3x1 fixed-point vector in Q16 format).

**Set Reference Frame**
Data used to rotate the Sensor to follow the platform axis specification after soldering. This is mainly useful for multi-axis sensors for which their outputs depends on sensor orientation.
This command is defined as a rotation matrix in Q30 format (9xint32).
Usually, the reference frame matrix will contain only 0, 1, or -1 in order to swap and reverse axis for the sensor data to be converted to sensor frame to the system frame. In case the mounting of the sensor is not perpendicular to system axis, any rotation can be applied using Q30 matrix data.

**Reset**
Software reset of a sensor. This is useful to manually reset a physical sensor or reset internal states of virtual sensors (e.g. for magnetometer sensor, this will reset in-use calibration states).

### 2.4.4 Expansion

Expansion can be described as:

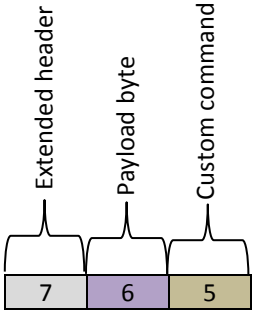


**Figure 5. Command Expansion bits Definition**

## Custom command

1-bit that defines Data packet type for interpreting frame data.

| VALUE | CUSTOM COMMAND |
|---|---|
| 0 | Command is standard and part of Table 7. Commands |
| 1 | Command is custom and not described in this document |

**Table 10. Custom Command**



Document Number: AN-000028
Revision: 1.20
Release Date: 01/20/2015

## Payload Byte

1-bit that defines Data packet type for interpreting frame data.

| VALUE | PAYLOAD BYTE |
|---|---|
| 0 | No byte is appended to the header |
| 1 | A byte is appended to the header to indicate command data payload size, before the command data (starting from Byte 3).<br>Payload bit must be set for if custom command bit is set. |

**Table 11. Payload Byte**

## Extended Command

1-bit that defines Data packet type for interpreting frame data.

| VALUE | EXTENDED COMMAND |
|---|---|
| 0 | Header is one byte |
| 1 | Header is two bytes (allowing adding more option in the future).<br>**Unused for now** |

**Table 12. Extended Command**



Document Number: AN-000028
Revision: 1.20
Release Date: 01/20/2015

## 2.5 FIFO OUTPUT PROTOCOL (HEADER AND DATA)

This section covers how the M0 communicates data to the AP.

Each frame contains the following elements:

- Header (16-bits)
  - Sensor ID (8-bits)
  - Status Flags (8-bits)
- Timestamp (32-bits see 2.5.2 ; this is overloaded by command ID in case the frame is an answer to a previous command)
- Data (variable size, see 2.5.3)

Data is Sensor specific. It may contain vectors, quaternions, activity id, etc.

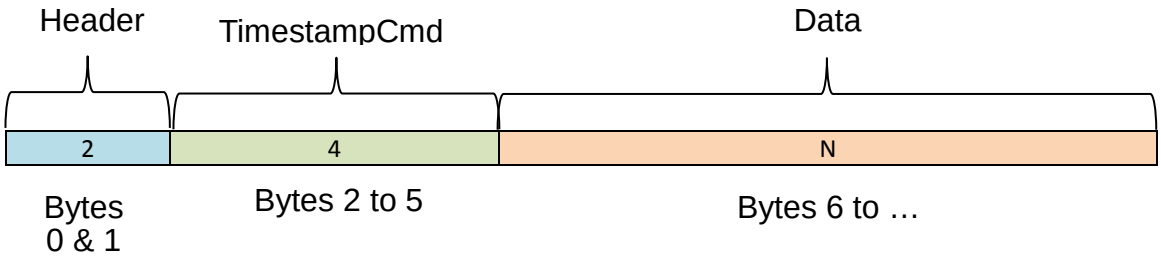


**Figure 6. Output Protocol Definition**

### 2.5.1 Header

Header can be described as:

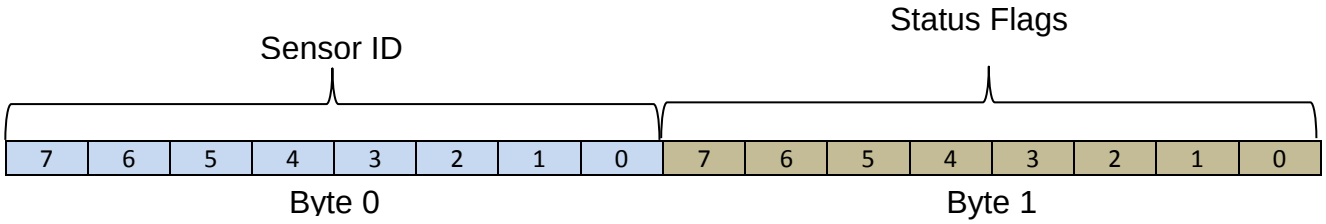


**Figure 7. Header Definition**



Document Number: AN-000028
Revision: 1.20
Release Date: 01/20/2015

**Sensor ID**
The sensor number is the same as in section 2.3, plus additional custom Sensors (TBD).

**Status Flags**
8-bit value defined as the following:

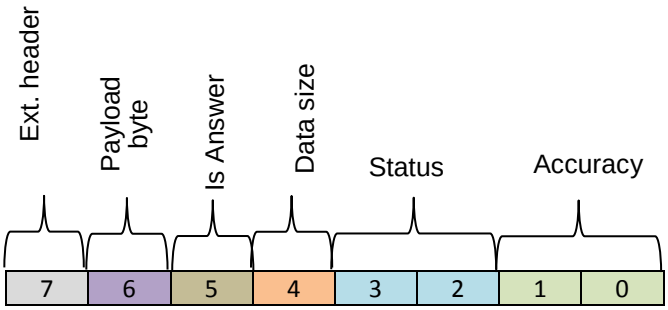


**Figure 8. Status Flag Definition**

## Accuracy

2-bit value that defines the reported accuracy of each sensor (from **Android 4.4** specifications):

| VALUE | ACCURACY |
|---|---|
| 0 | SENSOR_STATUS_UNRELIABLE |
| 1 | SENSOR_STATUS_ACCURACY_LOW |
| 2 | SENSOR_STATUS_ACCURACY_MEDIUM |
| 3 | SENSOR_STATUS_ACCURACY_HIGH |

**Table 13. Accuracy**

## Status

2-bits value indicating origin/reason for sending data:

| VALUE | STATUS |
|---|---|
| 0 | Data updated |
| 1 | State changed |
| 2 | Flush |
| 3 | Poll |

**Table 14. Status**

- Data updated: new Sensor data (normal)
- State changed: special values for initialization purposes (e.g. init timestamp)
- Flush: end of flush
- Poll: Data is a result of a poll and reflects latest measurement.

If status is 1 (State changed) or 2 (Flush), sensor data field will be empty (no data will be appended after the header) except if payload bit is set.
For status 0 (data updated) or 3(poll), sensor data will be appended to the header according to Table 20. Sensors Data Information.
If payload bit is set, a byte is appended after the header to precise length of data.



Document Number: AN-000028
Revision: 1.20
Release Date: 01/20/2015

## Data Size

1-bit that defines Data packet size for interpreting frame data.

| VALUE | DATA TYPE |
|---|---|
| 0 | 16,32-bit Integer (depends of the sensor) |
| 1 | 32-bit Float |

**Table 15. Data Size**

## Is Answer

1-bit that defines Data packet type for interpreting frame data.

| VALUE | IS ANSWER |
|---|---|
| 0 | Sensor data packet (timestamp is timestamp and data is sensor payload data) |
| 1 | Answer data packet (timestamp is command ID it is answering to and data is the answer itself) |

**Table 16. Is Answer**

## Payload Byte

1-bit that defines Data packet type for interpreting frame data.

| VALUE | PAYLOAD BYTE |
|---|---|
| 0 | No byte is appended to the header |
| 1 | A byte is appended to the header to indicate data payload length, before the data.<br>Payload bit must be set for all sensors id outside Android range. |

**Table 17. Payload Byte**

## Extended Header

1-bit that defines Data packet type for interpreting frame data.

| VALUE | IS ANSWER |
|---|---|
| 0 | Header is one byte |
| 1 | Header is two bytes (allowing adding more option in the future).<br>**Unused for now** |

**Table 18. Extended Header**

This bit is reserved to expand header in case we need to add more options. It is a 1-bit value that can be used to expand options. (Ex: Timestamp overflow flag, or extend the header).



Document Number: AN-000028
Revision: 1.20
Release Date: 01/20/2015

### 2.5.2 TimestampCmd

By default, when *IsAnswer* status flag is 0, this is the 32-bits *timestamp* value.
Its unit is *Clock Tick* (e.g. for ICM-306XX: 1/32768Hz=30.5xx micro-seconds). The clock frequency can be retrieved using the *Get Clock Rage* command.
When *IsAnswer* status flag is 1, the packet is then an answer to a previous command, and TimestampCmd contains related command ID. This allows knowing if M0 is providing some calibration gains or offsets for instance. When combined with sensor ID field, host can determine with certainty which command it is exactly answering to.

### 2.5.3 Data

This section describes additional data following the Header and Timestamp elements.
The size and meaning of the following data depends:

- On *IsAnswer* field: in case it is 1, it is the answer from the command requesting an answer. In case it is 0 it is the sensor data itself.
- On the *Sensor ID*. When possible, the data has the same meaning as the DMP3 data frame defined in chapter 4.
-

When *IsAnswer* status flag is 1, a data packet may follow:

| ID | COMMAND | DATA SIZE |
|---|---|---|
| 0 | Sensor off | *No answer for this command* |
| 1 | Sensor On | *No answer for this command* |
| 2 | Set Power | *No answer for this command* |
| 3 | Batch On | *No answer for this command* |
| 4 | Flush | *No answer for this command* |
| 5 | Set Delay | *No answer for this command* |
| 6 | Set Calibration Gains | *No answer for this command* |
| 7 | Get Calibration Gains | 9*int32 = 36 bytes |
| 8 | Set Calibration Offsets | *No answer for this command* |
| 9 | Get Calibration Offsets | 3*int32 = 12 bytes |
| 10 | Set Reference Frame | *No answer for this command* |
| 11 | Get Firmware Info | 3*uint8 + 8*char + 1*uint32 = 15 bytes |
| 12 | Get data | *variable* |
| 13 | Get clock rate | 1*uint32 = 4 bytes |
| 14 | Ping | 1*uint8 = 1 byte |
| 15 | Reserved | - |

**Table 19. Answers to Commands**



Document Number: AN-000028
Revision: 1.20
Release Date: 01/20/2015

**Get data**
For the host to manually request sensor data. This is useful when AP leaves SUSPEND mode, in order to retrieve last computed data for on-change sensors. E.g. when step counter is enabled and user walks while his phone is in SUSPEND mode, then the AP is supposed to get latest step count without waiting for next event, as it could never occur if user stops walking.
When receiving this command, the sensor hub will push a data packet with status flags set to 4 (POLL).
**Set / Get Calibration Gains**
Calibration data used to adjust the sensor gains: for example, for a 3-axis Sensor, the command is followed by *9*int32* (3x3 fixed-point matrix in Q30 format).
**Set / Get Calibration Offsets**
Data used to calibrate the Sensor offsets: for example, for a 3-axis Sensor, the command is followed by *3*int32* (3x1 fixed-point vector in Q16 format).
**Get FW info**
This command must return version and build information of the FW defined as:

- 3 bytes corresponding to the major, minor and patch version of the FW in binary format
- 8 bytes corresponding to the git commit hash in ASCII format
- The checksum of the generated and loaded binary in CRC32 (4 bytes) or SHA1 (20 bytes) (TBD)

**Get clock rate**
This command must return four bytes integer equals to the number of CPU ticks per µs. This value will be used to convert timestamp from data packets to µs or ms.
**Ping**
This command allows the AP to identify which Android sensors are currently supported by the sensor hub (allowing the AP HAL to not expose unsupported sensors).
When SH receives a PING command it must answer to the command with one byte data containing 1 if sensors is supported or 0 if not.
**Sensor data**
Sensor data is many times represented as a float in Android 4.4, but sent using the integer format. The conversion will be done by the AP, using a known scalar factor for each sensor.
For later evolution, it is possible to send data directly in Float format (IEEE754), using the DataType bit in the Status field.
All data is Calibrated & Scaled unless specified in the Comment column.
The following table is a summary of all Sensors (standard and wakeup) with the associated Data frame size, Data unit and comments.

Precision is expressed using *Qx* notation, where *x* is the number of bits reserved for the fractional parts.
Data are coded on 32btis signed integer or 16 bits signed integer when *h* prefix (*half*) is used. If *u* suffix (*unsigned*) is used, then underlying integer is unsigned.



Document Number: AN-000028
Revision: 1.20
Release Date: 01/20/2015

| ID | SENSOR NAME | SIZE | PRECISION | SCALE | ANDROID UNIT | COMMENT |
|---|---|---|---|---|---|---|
| **1** | Accelerometer | 3*int16 | hQ12 | $1/2^{12}$ | g | Calibrated, 3-axis |
| **2** | Magnetic Field | 3*int16 | hQ4 | $1/2^{4}$ | µTesla | Calibrated, 3-axis |
| **3** | Orientation | - | | $1/2^{6}$ | - | Deprecated |
| **4** | Gyroscope | 3*int16 | hQ4 | $1/2^{4}$ | dps | Calibrated, 3-axis |
| **5** | Light | 1*uint32 | Q0 | 1 | Lux | Todo: verify range and precision |
| **6** | Pressure | 1*uint32 | Q0 | 1 | Pa | Todo: verify range and precision |
| **7** | Temperature | - | | - | - | Deprecated |
| **8** | Proximity | 1*uint16 | hQ0 | 1 | mm | Not Scaled? |
| **9** | Gravity | 3*int16 | hQ12 | $1/2^{12}$ | g | |
| **10** | Linear Acceleration | 3*int16 | hQ12 | $1/2^{12}$ | g | |
| **11** | Rotation Vector | 4*int32 + 1*uint16 | Q30 hQ6 | $1/2^{30}$ ; $1/2^{6}$ | (accuracy in deg) | Quaternion (X,Y,Z,W)+ Accuracy |
| **12** | Relative Humidity | 1*uint16 | hQ9 | $1/2^{9}$ | Percent | Not Scaled? |
| **13** | Ambient Temperature | 1*int16 | hQ8 | $1/2^{8}$ | Celsius | Todo: verify range and precision |
| **14** | Magnetic Field Uncalibrated | 3*int16 + 3*int16 | hQ4 | $1/2^{4}$ | µTesla | Same as SensorID=2 but uncalibrated (Raw Data + Offsets) |
| **15** | Game Rotation Vector | 4*int32 + 1*uint16 | Q30 hQ6 | $1/2^{30}$ ; $1/2^{6}$ | deg | Same as SensorID=11 but based on Accerlometer + Gyroscope only |
| **16** | Uncalibrated Gyroscope | 3*int16 + 3*int16 | hQ4 | $1/2^{4}$ | dps | Same as SensorID=4 but uncalibrated (Raw Data + Offsets) |
| **17** | Significant Motion | 0 | | - | - | |
| **18** | Step Detector | 0 | | - | - | |



Document Number: AN-000028
Revision: 1.20
Release Date: 01/20/2015

| | | | | | | |
|---|---|---|---|---|---|---|
| **19** | Step Counter | 1*uint32 | Q0 | 1 | steps | Number of steps (since last boot or system reset) |
| **20** | Geomagnetic Rotation Vector | 4*int32 + 1*uint16 | Q30 hQ6 | $1 / 2^{30}$ ; $1/2^{6}$ | rad (accuracy) | Same as SensorID=11 but based on Acc + Mag only |
| **21** | Heart Rate | 1*uint16 | Q7 | $1 / 2^{7}$ | BPM | Todo: verify range and precision |
| **22** | Tilt Detector | 0 | | - | - | Emitted if: tilt_mvt > 35 degrees |
| | | | | | | |
| **31** | Activity Classification | Uint8_t | Q0 | 1 | Starting or ending activity Ending activity is indicated by MSBit set (eg: 0x01 = start 'In Vehicle' ; 0x81 = end 'In Vehicle') | 0x00: unknown<br>0x01: In Vehicle<br>0x02: Walking<br>0x03: Running<br>0x04: On Bicyle<br>0x05: Tilt<br>0x06: Still |

**Table 20. Sensors Data Information**



Document Number: AN-000028
Revision: 1.20
Release Date: 01/20/2015

# 3 DRIVER

## 3.1 OVERVIEW

Control of the IC is managed through a FIFO protocol instead of registers. Using the FIFO for communication will remove race conditions and give a scalable interface.

## 3.2 ENABLING SENSORS

To enable various sensors the function *void inv_enable_dmp_sensors(long bit_enable)* is used.
There are bit patterns to enable each sensor. The table below describes each feature. Some virtual sensors are derived from existing sensors. For example “Orientation Sensor” is derived from “Rotation Vector”.
It is recommended to compute “Orientation Sensor” on the AP to avoid harder processing. “Linear Acceleration” can be computed from “Rotation Vector” and “32-bit Accel”. It is easy to compute so it is recommended to be computed on ICM-306XX .

| bit_enable | PACKET SIZE (BYTES) | FEATURE |
|---|---|---|
| 0x8000_0000 | 12 | Accerlometer 32-bits |
| 0x4000_0000 | 6 | Gyroscope 16-bits |
| 0x2000_0000 | 6 | Mag 16-bits |
| 0x1000_0000 | 8 | ALS |
| 0x0800_0000 | 12 | Game Rotation Vector |
| 0x0400_0000 | 14 | Rotation Vector |
| 0x0200_0000 | 6 | Truncated Game Rotation Vector (Useful for batching) |
| 0x0100_0000 | 2 | Footer |
| 0x0080_0000 | 6 | Pressure data |
| 0x0040_0000 | 12 | Calibrated Gyroscope 32-bits |
| 0x0020_0000 | 12 | Calibrated Mag 32-bits |
| 0x0010_0000 | 4 | Step Detector |
| 0x0008_0000 | | Extended Header |
| 0x0004_0000 | | Pedometer Step Indicator Bit 2 |
| 0x0002_0000 | | Pedometer Step Indicator Bit 1 |
| 0x0001_0000 | | Pedometer Step Indicator Bit 0 |
| 0x0000_8000 | | TBD |
| 0x0000_4000 | 2 | Accerlometer Accuracy |
| 0x0000_2000 | 2 | Gyroscope Accuracy |
| 0x0000_1000 | | TBD |
| 0x0000_0800 | 2 | Drop Detection |
| 0x0000_0400 | | TBD |
| 0x0000_0200 | | Geomag Enable |
| 0x0000_0100 | | Batch Mode Enable |
| 0x0000_0080 | 6 | Activity Recognition |

**Table 21. Enabling Sensors**



Document Number: AN-000028
Revision: 1.20
Release Date: 01/20/2015

## SENSORS RATE

To set the Sensor data rate, the function *void inv_set_rate_dmp_sensors(int sensor_num, int rate_ms)* is used. The sensor number is defined as shown in the table below:

| sensor_num | RATE BASED ON | FEATURE |
|---|---|---|
| 0 | Accelerometer | Accelerometer 32-bits |
| 1 | Gyroscope | Gyroscope 16-bits |
| 2 | Gyroscope | Game Rotation Vector |
| 3 | Gyroscope | Calibrated Gyroscope 32-bits |
| 4 | Accelerometer | Step Counter (TBD) |
| 5 | Gyroscope | Game Rotation Vector + Step Indicator |
| 6 | Magnetometer | Magnetometer 16-bits |
| 7 | Mag | Calibrated Magnetometer 32-bits |
| 8 | Gyroscope (Accelerometer, Magnetometer) | Rotation Vector |
| 9 | Pressure | Pressure |
| 10 | ALS | ALS |
| | | |

**Table 22. Sensors Rate**

## 3.3 AUTOMATIC ACTIVITY RECOGNITION

Automatic Activity Recognition (AAR) is a new feature defined in Android 5.0 ( Lollipop). The purpose of AAR is to detect the activity of the device with an associated confidence. Detected activities include “in vehicle”, “walk”, “run”, “on bike”, “tilt” and “still”.
The inputs to AAR are accelerator data and gyro data. The accelerator has to be constantly turned on when AAR is running. The gyro does not have to be always on, as gyro consumes more power. The AAR module determines when to turn on the gyro when needed, and similarly turns off the gyro when it is not needed. The input to the AAR module is defined as shown in the table below:

| SENSOR | RATE | LENGTH | UNIT | AVAILABILITY |
|---|---|---|---|---|
| Accelerometer | 56.25 Hz | 16 bit signed | Gravity, scaled by 2^14 | Always on |
| Gyroscope | 56.25 Hz | 16 bit signed | dps | Needs-based |

**Table 23. Automatic Activity Recognition Input**



Document Number: AN-000028
Revision: 1.20
Release Date: 01/20/2015

It is worth mentioning that secondary sensors such as magnetometer and proximity sensor are also being considered to be used as inputs to the AAR module.
The outputs of the AAR module are the transitions between detected activities. As the user may move from one activity to another quickly, the AAR output is sent to FIFO, instead of using CIM to wake up the Application Processor (AP). The output of the AAR is 7-byte long. The format begins with a header (1 byte), followed by 1 byte to indicate the start of the detected activity, which is followed by 1 byte to indicate the end of the detected activity, and finally a 4-byte timestamp. Unit of the timestamp is DMP cycles.

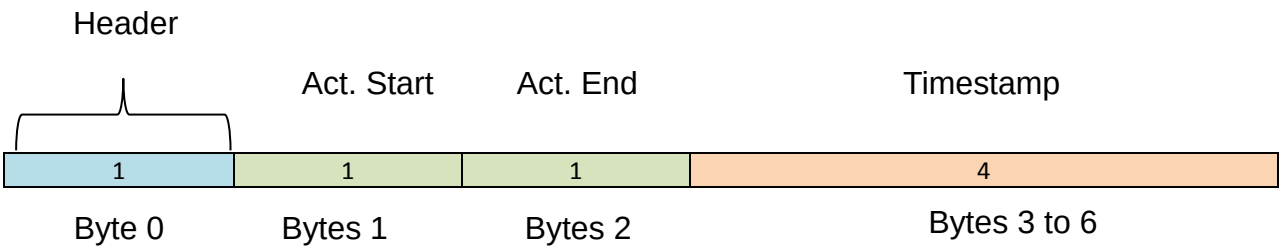


**Figure 9. AAR Definition**

## Detected Activity

The one byte of Activity Start could equal to one of the following values. The same is also true for Activity End:

| VALUE | ACTIVITY |
|---|---|
| 0x1 | In Vehicle |
| 0x2 | Walking |
| 0x4 | Running |
| 0x8 | On Bicycle |
| 0x10 | Tilt |
| 0x20 | Still |

**Table 24. Detected Activity**



Document Number: AN-000028
Revision: 1.20
Release Date: 01/20/2015

# 4 LOWER DRIVER

## 4.1 OVERVIEW

This covers the driver API from API to DMP4/DMP3 and external devices.

| FUNCTION NAME | DESCRIPTION |
|---|---|
| *inv_error_t inv_set_odr(int sensor_number, int divide)* | Sets the output data rate for a sensor. This sets both the hardware and DMP firmware. In cases of conflicts between sensors, this function will determine resolution. Uses what has been logged as enabled to evaluate sensor rates. The sensor given in sensor_number is assumed to be enabled, even if the enable function has not been called.<br><br>**sensor_number** A number between 1 and 20 (inclusive) corresponding to Android sensors but excluding wake up sensors and meta data and excluding extended sensors.<br><br>**divide** A N+1 divider for the odr. 0=1125 Hz,1=1125/2 Hz,2=1125/3 Hz,3=1125/4 Hz, etc. -1 is a special case for 9000Hz. Note sometimes the data rate will be set higher.<br><br>**return** Non zero error code on error. |
| *inv_error_t inv_enable_sensor(int sensor_number, int enable)* | Enables or disables a given sensor.<br>**sensor_number** A number between 1 and 20 (inclusive) corresponding to Android sensors but excluding wake up sensors and meta data and excluding extended sensors.<br>**enable** Non-zero enables the given sensor. Zero disables given sensor.<br>**return** Non zero error code on error. |
| *inv_error_t inv_write_mems(unsigned short reg, int length, unsigned char *data)* | Write to DMP3/MEMs SRAM |
| *inv_error_t inv_read_mems(unsigned short reg, int length, unsigned char *data)* | Read from DMP3/MEMs SRAMSRAM |
| *inv_error_t inv_write_reg(unsigned short reg, int length, unsigned char *data)* | Write to DMP4/ARM (ICM-306XX) Silicon registers |
| *inv_error_t inv_read_reg(unsigned char reg, int length, unsigned char *data)* | Read from DMP4/ARM (ICM-306XX) Silicon registers |
| *inv_error_t inv_write_mems_reg(unsigned char reg, int length, unsigned char *data)* | Write to DMP3/ARM Silicon registers |
| *inv_error_t inv_read_mems_reg(unsigned char reg, int length, unsigned char *data)* | Read from DMP3/ARM Silicon registers |
| *inv_error_t inv_initialize_lower_driver()* | Should be called once on power up. Loads DMP3, initializes internal variables needed for other lower driver functions. |
| *inv_error_t inv_sleep_mems()* | Puts DMP3 into the lowest power state. Assumes sensors are all off. |
| *inv_error_t inv_wakeup_mems()* | Wakes up DMP3 . |
| *int inv_wait_for_events(int event_mask)* | Waits in a low power state until given event happens. Waits for INV_INPUT_FIFO_EVENT, INV_DATA_READY_EVENT, INV_WATCHDOG, INV_CADENCE). Returns mask of which event(s) was triggered. |
| *inv_write_fifo(int fifo_num, int size, unsigned char *data)* | Writes to a ICM-306XX FIFO |
| *inv_read_fifo(int fifo_num, int size, unsigned char *data)* | Reads a ICM-306XX FIFO |
| *inv_error_t inv_mems_firmware_load()* | Load DMP3/MEMS Firmware |
| *inv_error_t inv_set_gyro_fullscale(int level)* | Sets the fullscale range of the gyro in hardware and on DMP. **level** 0=250 dps, 1=500 dps, 2=1000 dps, 3=2000 dps |



Document Number: AN-000028
Revision: 1.20
Release Date: 01/20/2015

| | |
|---|---|
| *inv_error_t inv_set_accel_fullscale(int level)* | Sets the fullscale range of the accel in hardware and on DMP. level 0=2g, 1=4g, 2=8g, 3=16g |
| *int inv_determine_accel_hw_odr(int sensor_number, int divide)* | Determines what the accel divider should be set to based upon sensors that are on and the given sensor. **sensor_number** A number between 1 and 20 (inclusive) corresponding to Android sensors but excluding wake up sensors and meta data and excluding extended sensors.**divide** A N+1 divider for the odr. 0=1125 Hz,1=1125/2 Hz,2=1125/3 Hz,3=1125/4 Hz, etc. -1 is a special case for 9000Hz. Note sometimes the data rate will be set higher. **return** what the accel hardware should be set to |
| *inv_error_t inv_save_calibration_data()* | Saves calibration data |

**Table 25. Lower Driver Functions**



Document Number: AN-000028
Revision: 1.20
Release Date: 01/20/2015

# 5 SELF-TEST

## 5.1 OVERVIEW

Self-Test will be run on the ARM-M0. The main reason for this is if there is an issue with self-test during production, it will need to be fixed within hours and therefore should have the most visibility.

| FUNCTION NAME | DMP4 | DESCRIPTION |
|---|---|---|
| *inv_pause_system(int enable)* | Y | *enable != 0*<br>Save current state. Stop I2C transactions, free up I2C channels. Turn off DMP. |
| | | *enable == 0*<br>Restore system to the state before the pause. |
| *inv_write_mpu(unsigned char reg, int length, unsigned char *data)* | N | Write to DMP3/MEMs Silicon registers |
| *inv_read_mpu(unsigned char reg, int length, unsigned char *data)* | N | Read from DMP3/MEMs Silicon registers |
| *inv_write_reg(unsigned char reg, int length, unsigned char *data)* | N | Write to DMP4/ARM (ICM-306XX) Silicon registers |
| *inv_read_reg(unsigned char reg, int length, unsigned char *data)* | N | Read from DMP4/ARM (ICM-306XX) Silicon registers |

**Table 26. Self Test**



Document Number: AN-000028
Revision: 1.20
Release Date: 01/20/2015

# 6 CODE SIZE TARGETS

Below is a list for code size targets:

| CODE SIZE | FUNCTION | MEMORY |
|---|---|---|
| 7K | DMP3 Image (Acc calib, Pedometer, Game Rotation Vector, Activity Classification) | Flash, SRAM*, ROM* |
| 2K | Mag calibration on DMP4 | SRAM |
| 1K | 9-axis Sensors fusion | SRAM |
| 1K | DMP Driver | FLASH |
| 5K | Framework | FLASH/SRAM |
| 2K | embOS or FreeRTOS (RT-OS) | SRAM |
| ? | ARM C library | ROM* |
| | | |

**Table 27. Code Size**



Document Number: AN-000028
Revision: 1.20
Release Date: 01/20/2015

# 7 DOCUMENT INFORMATION

## 7.1 REVISION HISTORY

| REVISION | DATE | DESCRIPTION | AUTHOR |
|---|---|---|---|
| 1.0 | Aug 13 2014 | Preview Initial version | Kerry Keal |
| | Aug 18 2014 | FIFO output control + more clarity | Kerry Keal |
| 1.1 | Sept 2 2014 | New FIFO Input and Output frame + New options | Etienne de Foras and Guillaume Aujay |
| 1.2 | Sept 3 2014 | Re-add 'wake-up' bit to Input frame + Redefine Output frame timestamp + Add details | Guillaume Aujay |
| 1.3 | Sept 3 2014 | Added M0 to AP frame definitions | Etienne de Foras |
| 1.4 | Sept 4 2014 | Update Input Command table + Update Output Status Flags definition + Update Output Data table | Etienne de Foras and Guillaume Aujay |
| 1.5 | Sept 8 2014 | - Update 'Sensor ID' table according to Android L -> no need for 'wake up' bit anymore.<br>- 'Set Reference Frame' use quaternion now.<br>- Update Data packet table<br>- Add 'embOS' code size | Guillaume Aujay |
| 1.6 | Sept 8 2014 | Add Lower Level driver functions | Kerry Keal |
| 1.7 | Sept 10 2014 | Added More Lower Level driver functions | Kerry Keal |
| 1.8 | Sept 16 2014 | Factorize Commands table<br>Add 'Set Power' command<br>Re-add 'Time mode' flag | Guillaume Aujay |
| 1.9 | Sept 16-25, 2014 | Merged in Gullaume 'Set Power' with James' cleanup.<br>Removed other 1.8 additions. | James Lim / Kerry Keal |
| 1.10 | | Preview arch – v1 | Kerry Keal |
| 1.11 | Oct 22, 2014 | Add ability to get an answer from FIFO output protocol in case a command requests it (introduce IsAnswer flag) | Julie Gonin |
| 1.12 | Dec 1, 2014 | Update data sizes and scaling for data output | Kerry Keal |
| 1.13 | Dec 3, 2014 | Specify expected data depending on STATUS flag<br>Add POLL_DATA status flag<br>Add Payload bit<br>Update sensor ID based on latest Android sensors.h from L release | Paul Besson |

| | | | |
|---|---|---|---|
| 1.14 | Dec 4, 2014 | Add dedicated command for CPU tick information<br>Specify get FW info command<br>Specify set/get calibration command<br>Specify set/get reference frame<br>Add PING command | Paul Besson |
| 1.15 | Dec 12, 2014 | Give more details on FIFO input/output data | Cyril Protat |
| 1.16 | Dec 17, 2014 | Add Kerry's section about AAR output in DMP3<br>Add FIFO Protocol ID and packet description for Activity Classifier | Paul Besson |
| 1.17 | Dec 17, 2014 | Change Set Reference Frame command definition to something easier to use and implement | Paul Besson |
| 1.18 | Dec 18, 2014 | Specified endianness for FIFO data<br>Detailed the expansion bit/byte for commands | Cyril Protat |
| 1.19 | Jan 12, 2015 | Reserved 5 bits for command instead of 4<br>Add RESET command<br>Split Expansion bits for command and add detailed description | Paul Besson |
| 1.20 | Jan 20, 2015 | Wrong version approved in Agile, need to upload new version and up rev number to 1.20 | Charlotte White |

**Table 28. Revision History**

InvenSense

©2015 InvenSense, Inc. All rights reserved.