Application Note

Programming Sequence for ICM-20648 DMP Hardware Function

Contents

1 Revision History

Revision Date	Revision	Description
07/10/2015	0.1	Initial Release for ICM-20648

2 Overview

InvenSense MotionTracking Devices include advanced hardware features that can be enabled and disabled through simple hardware register settings described in this document. These advanced hardware features enable the following motion-based functions without using an external microprocessor: Batch Mode (BM) operation, Significant Motion Detection (SMD), Step Detect/Step Count, Pick-up gesture detection, Fsync detection, Activity Recognition (Basic Activity Classifier (BAC)), sensor fusion ("quaternion/GRV-Gaming rotation vector"), and dynamic calibration.

This document covers the following areas:

1. ICM 20648 registers used to set up the DMP.
2. Set up and configuration procedures for each of the advanced hardware features
   - Prepare the chip settings for use of advanced hardware features
   - Setup and configure the advanced hardware features
   - Enable operation of advanced hardware features
   - Read and interpret output data

Invensense ICM-20648 devices support Android L features in DMP.
The advance hardware features considered in this application note are as follows:

- Significant Motion Detection
- Batch mode
- 6-Axis Game Rotation Vector
- 9-Axis Quaternion
- Geo-magnetic quaternion
- Multiple ODR (Output Data Rate) support.
- Dynamic calibration (gyro/accel/compass)
- Step Count/Step detect
- Tilt detect
- Pick up detect
- Fsync detect
- Activity Recognition

In addition to these advanced hardware features, the InvenSense embedded Digital Motion Processor (DMP) can output the following data to the FIFO:

- Gyroscope raw data
- Accelerometer raw data
- Compass raw data
- Pressure raw data
- ALS raw data

Please note that the pedometer data is directly read from the MotionTracking device advanced hardware feature registers.

3 HW setup

ICM should be setup by using the main ICM HW registers below before using the advanced hardware features. Please refer to InvenSense Register Map documents for detailed information about HW registers.

3.1 Initializing the ICM

- Configure clock source through `PWR_MGMT_1`.
- Enable accel and gyro sensors through `PWR_MGMT_2`.
- Configure Gyro/Accel in Low power mode with `LP_CONFIG`.
- Set Gyro FSR (Full scale range) to 2000dps through `GYRO_CONFIG_1`.
- Set Accel FSR (Full scale range) to 4g through `ACCEL_CONFIG`.

3.2 Configuring FIFO and Interrupts

- Enable interrupt for FIFO overflow from FIFOs through `INT_ENABLE_2`.
- Turn off what goes into the FIFO through `FIFO_EN, FIFO_EN_2`.
- Turn off data ready interrupt through `INT_ENABLE_1`.

3.3 Reset the FIFO

- Reset FIFO through `FIFO_RST`.

3.4 Configuring Sensor Sample Rate

- Set gyro sample rate divider with `GYRO_SMPLRT_DIV`.
- Set accel sample rate divider with `ACCEL_SMPLRT_DIV_2`.

3.5 Configuring the DMP start address

- Setup DMP start address through `PRGM_STRT_ADDRH`/`PRGM_STRT_ADDRL`.

3.6 Loading DMP Firmware

The DMP firmware size is >12Kbytes. DMP program code may be loaded one byte at a time or in burst mode as explained below.

3.6.1 Load Firmware One Byte at a Time

Starting from byte [0] of the firmware image provided, perform the following steps to load each byte of image to Nth byte of DMP address. N must start from address (0x90) and increase accordingly:

- Write the value (N >> 8) to `MEM_BANK_SEL`
- Write the value (N & 0xFF) to `MEM_START_ADDR`
- Write the data to `MEM_R_W` starting from the 0x90th byte of DMP firmware image.

Repeat the steps above for each byte of the DMP firmware image.

3.6.2 Load Firmware up to 256 Bytes at a Time

Up to 256 bytes can be burst written. Please follow the steps outlined in previous section. However, the maximum number of bytes that can be written at once is not always 256, but is given by (256 – (N & 0xFF)).

3.6.3 Load Firmware Start Value and start DMP

Please perform the following once the DMP firmware has been loaded.

- Write the 2 byte Firmware Start Value to ICM `PRGM_STRT_ADDRH`/`PRGM_STRT_ADDRL`

The Firmware Start Value is provided by InvenSense. If you do not have this value, please contact your field representative.

3.7 Configure Accel scaling to DMP

The DMP scales accel raw data internally to align 1g as 2^25.

To do this and output hardware unit again as configured FSR, write 0x4000000 to ACC_SCALE DMP register, and write 0x40000 to ACC_SCALE2 DMP register.

ACC_SCALE

Type: Read/Write

Register Name	Bits[31:0]
ACC_SCALE	Write accel scaling value for internal use

Description:

In order to align internal accel raw data 2^25 = 1g write 0x4000000 when FSR is 4g.

ACC_SCALE2

Type: Read/Write

Register Name	Bits[31:0]
ACC_SCALE2	Write accel scaling down value

Description:

In order to output hardware unit data as configured FSR write 0x40000 when FSR is 4g.

3.8 Configure Compass mount matrix and scale to DMP

The mount matrix write to DMP register is used to align the compass axes with accel/gyro. This mechanism is also used to convert hardware unit to uT. The value is expressed as 1uT = 2^30.

CPASS_MTX_xx

Type: Read/Write

Register Name	Bits[31:0]
CPASS_MTX_00	Compass mount matrix and scale
CPASS_MTX_01	Compass mount matrix and scale
CPASS_MTX_02	Compass mount matrix and scale
CPASS_MTX_10	Compass mount matrix and scale
CPASS_MTX_11	Compass mount matrix and scale
CPASS_MTX_12	Compass mount matrix and scale
CPASS_MTX_20	Compass mount matrix and scale

Register Name	Bits[31:0]
CPASS_MTX_21	Compass mount matrix and scale
CPASS_MTX_22	Compass mount matrix and scale

Description:

Each compass axis will be converted as below.

X = raw_x * ***CPASS_MTX_00*** + raw_y * ***CPASS_MTX_01*** + raw_z * ***CPASS_MTX_02***

Y = raw_x * ***CPASS_MTX_10*** + raw_y * ***CPASS_MTX_11*** + raw_z * ***CPASS_MTX_12***

Z = raw_x * ***CPASS_MTX_20*** + raw_y * ***CPASS_MTX_21*** + raw_z * ***CPASS_MTX_22***

3.9 Reset FIFO and DMP

- Reset FIFO with `FIFO_RST`.
- Reset DMP with `USER_CTRL`.

3.10 Enable DMP interrupt

- Enable DMP interrupt with `INT_ENABLE`.

4 Advanced Hardware Features

The advanced hardware features are not initially enabled after device power up. These features must be individually enabled through programming the DMP registers (not through direct hardware registers).

DMP register addresses may change from release to release. For the addresses of these registers in the current release, please refer to our specific release information.

4.1 Writing to DMP Registers

To write to advanced hardware register 0xABC with data {0x1, 0x2, 0x3, 0x4}, please follow the procedure below:

1. Write 0xA to `MEM_BANK_SEL`
2. Write 0xBC to `MEM_START_ADDR`
3. Write {0x1, 0x2, 0x3, 0x4} to `MEM_R_W`

Please note that when we say “write {0x1, 0x2, 0x3, 0x4} to 0xABC”, we imply writing {0x1, 0x2, 0x3, 0x4} to registers 0xABC – 0xABF. This can be achieved by writing the registers one at a time or by burst writing.

4.2 Configuring DMP to output data to FIFO

DMP can be configured to stream sensor data to the FIFO. The following register is used to configure DMP to output required data.

DATA_OUT_CTL1

Type: Read/Write

Register Name	Bits[15:0]
DATA_OUT_CTL1	Write the required bit map

Description:

In order to turn on data streaming to the FIFO, write the correct bit map as per the following table.

Data	Bits[15:0]
16-bit accel	0x8000
16-bit gyro	0x4000
16-bit compass	0x2000
16-bit ALS	0x1000
32-bit 6-axis quaternion	0x0800
32-bit 9-axis quaternion + heading accuracy	0x0400
16-bit pedometer quaternion	0x0200
32-bit Geomag rv + heading accuracy	0x0100
16-bit Pressure	0x0080
32-bit calibrated gyro	0x0040
32-bit calibrated compass	0x0020
Pedometer Step Detector	0x0010
Header 2	0x0008
Pedometer Step Indicator Bit 2	0x0004
Pedometer Step Indicator Bit 1	0x0002
Pedometer Step Indicator Bit 0	0x0001

Table 1. Data output control 1 register bit definition

DATA_INTR_CTL

Type: Read/Write

Register Name	Bits[15:0]
DATA_INTR_CTL	Write the required bit map

Description:

Determines which sensors can generate interrupt according to bit map defined for ***DATA_OUT_CTL1*** in Table 1 above.

DATA_RDY_STATUS

Type: Read/Write

Register Name	Bits[15:0]
DATA_RDY_STATUS	Write the required bit map

Description:

Indicates which sensor is available.

Sensor available	Bits[15:0]
Secondary samples available	0x0008
Accel samples available	0x0002
Gyro samples available	0x0001

Table 2. Data ready status register bit definition

4.3 Configuring DMP to output data at multiple ODRs

DMP is capable of outputting multiple sensor data at different rates to FIFO.

The following table gives the key definitions ODRs for each of the data features described in the previous section.

ODR register	Definition
ODR_ACCEL	Register for accel ODR
ODR_GYRO	Register for gyro ODR
ODR_CPASS	Register for compass ODR
ODR_ALS	Register for ALS ODR
ODR_QUAT6	Register for 6-axis quaternion ODR
ODR_QUAT9	Register for 9-axis quaternion ODR
ODR_PQUAT6	Register for 6-axis pedometer quaternion ODR
ODR_GEOMAG	Register for Geomag rv ODR
ODR_PRESSURE	Register for pressure ODR
ODR_GYRO_CALIBR	Register for calibrated gyro ODR
ODR_CPASS_CALIBR	Register for calibrated compass ODR

In order to set an ODR for a given sensor data, write 2-byte value to DMP using key defined above for a particular sensor.

Setting value can be calculated as follows:

Value = (DMP running rate (225Hz) / ODR) - 1

E.g. For a 25Hz ODR rate, value= (225/25) -1 = 8.

During run-time, if an ODR is changed, the corresponding rate counter must be reset. To reset, write 2-byte {0,0} to DMP using keys below for a particular sensor:

ODR counter register	Definition
ODR_CNTR_ACCEL	Register for accel ODR counter
ODR_CNTR_GYRO	Register for gyro ODR counter
ODR_CNTR_CPASS	Register for compass ODR counter
ODR_CNTR_ALS	Register for ALS ODR counter
ODR_CNTR_QUAT6	Register for 6-axis quaternion ODR counter
ODR_CNTR_QUAT9	Register for 9-axis quaternion ODR counter
ODR_CNTR_PQUAT6	Register for 6-axis pedometer quaternion ODR counter
ODR_CNTR_GEOMAG	Register for Geomag rv ODR counter
ODR_CNTR_PRESSURE	Register for pressure ODR counter
ODR_CNTR_GYRO_CALIBR	Register for calibrated gyro ODR counter
ODR_CNTR_CPASS_CALIBR	Register for calibrated compass ODR counter

4.4 Additional FIFO output control

In addition to the above data configurations, DMP can be configured to output data in batch mode.

Use the following register to configure the BM, accel/gyro/compass accuracy and gesture such as Pick-up.

DATA_OUT_CTL2

Type: Read/Write

Register Name	Bits[15:0]
DATA_OUT_CTL2	Write the required bit map

Description:

In order to configure the BM and other features, use the following bit map to write to this register.

Data	Bits[15:0]
Accel Accuracy	0x4000
Gyro Accuracy	0x2000
Compass Accuracy	0x1000
Fsync detection	0x0800
Pick-up	0x0400
Batch Mode Enable	0x0100
Activity Recognition (BAC)	0x0080
Secondary On/Off	0x0040

4.4.1 Batch mode configuration

FIFO_WATERMARK

Type: Read/Write

Register Name	Bits[16:0]
FIFO_WATERMARK	FIFO watermark

Description:

DMP will send FIFO interrupt if FIFO count > FIFO watermark. FIFO watermark is set to 80% of actual FIFO size by default.

BM_BATCH_MASK

Type: Read/Write

Register Name	Bits[16:0]
BM_BATCH_MASK	Write the required bit map

Description:

Batch counter will increment only if batch mode mask is set according to bit definition for ***DATA_RDY_STATUS*** as shown in Table 2.

BM_BATCH_CNTR

Type: Read/Write

Register Name	Bits[31:0]
BM_BATCH_CNTR	BM counter

Description:

Set this register to zero before batch mode is enabled.

4.4.2 Batch mode threshold

BM_BATCH_THLD

Type: Read/Write

Register Name	Bits[31:0]
BM_BATCH_THLD	BM threshold

Description:

Set the batch mode threshold in terms of number ticks, each tick is (1/225) msec.

4.5 Configuring DMP features

DMP can be configured for Android L and Invensense specific features by the following register.

MOTION_EVENT_CTL

Type: Read/Write

Register Name	Bits[15:0]
MOTION_EVNT_CTL	Write the required bit map

Description:

This register is used as a bit map to configure the following features.

Feature	Bits[15:0]
Activity Recognition / Pedometer	0x4000
Pedometer Interrupt	0x2000
Tilt Interrupt	0x1000
Significant Motion Detection	0x0800
Accel Calibration	0x0200
Gyro Calibration	0x0100
Compass Calibration	0x0080
9-axis	0x0040
Pick-up	0x0010
Geomag rv	0x0008
Activity Recognition / Pedometer accel only	0x0002

4.5.1 Reading number of steps out of DMP registers

PED_STD_STEPCTR

Type: Read

Register Name	Bits[31:0]
PEDSTD_STEPCTR	Number of steps in big endian

Description:

This register is used to read the number of steps when pedometer is activated.

4.5.2 Reading pedometer walk time

PED_STD_TIMECTR

Type: Read

Register Name	Bits[31:0]
PEDSTD_TIMECTR	Walk time in big endian

Description:

This register is used to read the walk time once pedometer is activated.

4.5.3 Enabling Activity Recognition (BAC) feature

In order to enable DMP Activity Recognition feature write 0x4000 to the ***MOTION_EVENT_CTL*** DMP register. Detected activities will be sent to FIFO when write 0x0008 to ***DATA_OUT_CTL1*** and 0x00c0 to ***DATA_OUT_CTL2***.

The DMP Activity Recognition feature on ICM20648 uses both accelerometer and gyroscope data. The minimal ODR for accelerometer and gyroscope is 56.25 Hz. If the accelerometer or gyroscope is running at a higher rate, configure the following DMP register to downsample to 56.25 Hz.

BAC_RATE

Type: Read/Write

Register Name	Bits[15:0]
BAC_RATE	Write {0x00, 0x00} if accelerometer runs at 56.25 Hz Write {0x00, 0x01} if accelerometer runs at 112.5 Hz Write {0x00, 0x03} if accelerometer runs at 225 Hz

Description:

This register is used to downsample accelerometer ODR to 56.25 Hz.

To save power, the DMP Activity Recognition feature may request through FIFO data to turn off gyroscope to save power when the gyroscope is not needed.

4.5.4 Enabling Significant Motion Detect (SMD) feature

In order to enable SMD feature write 0x4800 to the ***MOTION_EVENT_CTL*** DMP register. When an SMD event is detected, it will appear on bit 1 of `DMP_INT1_STATUS` register.

4.5.5 Enabling Tilt Detector feature

In order to enable DMP Tilt detection feature write 0x4000 to the ***MOTION_EVENT_CTL*** DMP register. The tilt event will be reported as FIFO data when write 0x0008 to ***DATA_OUT_CTL1*** and 0x00c0 to ***DATA_OUT_CTL2***.

To enable tilt interrupt, write 0x5000 to the ***MOTION_EVENT_CTL*** DMP register. When a tilt event is detected, it will appear on bit 3 of `DMP_INT1_STATUS` register.

4.5.6 Enabling Pick Up Gesture feature

In order to enable DMP Pick up gesture feature write 0x0010 to the ***MOTION_EVENT_CTL*** DMP register. When a Pick up event is detected, it is sent to FIFO when write 0x0008 to ***DATA_OUT_CTL1*** and write 0x0400 to ***DATA_OUT_CTL2***.

4.5.7 Enabling Fsync detection feature

In order to enable DMP Fsync detection feature write 0x0008 to ***DATA_OUT_CTL1*** and write 0x0800 to ***DATA_OUT_CTL2***.

The Fsync detection will be reported as FIFO data.

4.5.8 Configuring Accel calibration

There are DMP registers regarding accel calibration and it needs to be set according to sample rate.

Register Name	Bits[15:0]
ACCEL_CAL_RATE	0 (225Hz, 112Hz, 56Hz)

Register Name	Bits[31:0]
ACCEL_ALPHA_VAR	1026019965 (225Hz) 977872018 (112Hz) 882002213 (56Hz)
ACCEL_A_VAR	47721859 (225Hz) 95869806 (112Hz) 191739611 (56Hz)

4.5.9 Configuring Compass calibration

There is DMP register regarding compass calibration and it needs to be set according to sample rate.

Register Name	Bits[15:0]
CPASS_TIME_BUFFER	Write a number of corresponding to the running rate of compass. e.g. 70 (70Hz)

4.5.10 Configuring Gyro gain

There is DMP register need to be configured according to `TIMEBASE_CORRECTION_PLL` register.

Register Name	Bits[31:0]
GYRO_SF	Write a value calculated following Appendix II

4.5.1 Configuring Accel gain

There is DMP register need to be configured according to sample rate of accel.

Register Name	Bits[31:0]
ACCEL_ONLY_GAIN	15252014 (225Hz) 30504029 (112Hz) 61117001 (56Hz)

4.6 Biases

Biases can be set by an outside control. If a bias algorithm is running on the DMP it will over-write these values. The biases are 32-bits in chip frame in hardware unit scaled by 2^12 (FSR 4g) for accel, 2^15 for gyro, in uT scaled by 2^16 for compass.

Register Name	Bits[31:0]
GYRO_BIAS_X	Gyro X bias
GYRO_BIAS_Y	Gyro Y bias
GYRO_BIAS_Z	Gyro Z bias
ACCEL_BIAS_X	Accel X bias
ACCEL_BIAS_Y	Accel Y bias
ACCEL_BIAS_Z	Accel Z bias
CPASS_BIAS_X	Compass X bias
CPASS_BIAS_Y	Compass Y bias
CPASS_BIAS_Z	Compass Z bias

5 FIFO Output summary

The procedures described in the previous sections outlined how to setup, configure and start the DMP hardware mode of operation. At this point, the hardware should be receiving samples at a rate of 225 Hz, and writing sets of outputs to the FIFO at the configured output rate.

5.1 DMP Interrupt

In normal mode, each time a set of data is written to the FIFO, it raises an interrupt. In batch mode, interrupt is raised based on batch settings.

Since the interrupt can come from multiple sources, the user should confirm that it is related to one of the advanced hardware features described in this document by reading the `DMP_INT1_STATUS`.

The contents of the FIFO can be read at this point. To learn the procedure to read the MPU FIFO, refer to InvenSense Product Specification and Register Map documents.

5.2 Content of DMP Output to FIFO

The exact contents of the data output to the FIFO depend on the features that were enabled in previous section.

Data for each feature activated will be in the following order with the size indicated below. These data packet will be after a 2 byte general header. When a general header indicates the existence of header2, header2 is stored just after a general header.

- Raw ACCEL sensor (6 bytes of data, 2 bytes for each axis, X, Y, Z).
- Raw Gyro (6 bytes of data, 2 bytes for each axis, X, Y, Z).
- Compass (6 bytes of data, 2 bytes for each axis, X, Y, Z).
- ALS (8 bytes of data)
- 6-axis Quaternion (12 bytes of data)
- 9-axis Quaternion (14 bytes of data)
- 6-axis pedometer Quaternion (6 bytes of data)
- Geomag rv (14 bytes of data)
- Pressure (6 bytes of data).
- Calibrated Gyro (12 bytes of data, 4 bytes for each axis, X, Y, Z)
- Calibrated Compass (12 bytes of data, 4 bytes for each axis, X, Y, Z)
- Pedometer step detector (4 bytes for timestamp)
- Accel Accuracy (2 bytes of data)
- Gyro Accuracy (2 bytes of data)
- Compass Accuracy (2 bytes of data)
- Fsync Delay Time (2 bytes of data)
- Pick up gesture (2 bytes of data)
- Activity Recognition (6 bytes of data)
- Secondary On/Off (2 bytes data)

If a particular feature is not enabled, its output will not be present in the FIFO data. However, the above order is maintained.

The data will always be preceded with a 2-byte header data and additionally 2-byte header2. The header is a bitmap and the bits will be set based on the activated features.

Two bytes of header bitmap will be as follows:

Header2	0x0008
Accel	0x8000
Gyro	0x4000
Compass	0x2000
ALS	0x1000
6-Axis Quaternion	0x0800
9-Axis Quaternion	0x0400
6-Axis Pedometer Quaternion	0x0200
Geomag rv	0x0100
Pressure	0x0080
Calibrated Gyro	0x0040
Calibrated Compass	0x0020
Step Detector	0x0010

Accel Accuracy	0x4000 (header2)
Gyro Accuracy	0x2000 (header2)
Compass Accuracy	0x1000 (header2)
Fsync	0x0800 (header2)
Pick up	0x0400 (header2)
Activity Recognition	0x0080 (header2)
Secondary On/Off	0x0040 (header2)

5.2.1 Accel data

Hardware unit, 2 bytes for each axis, X, Y and Z in order.

5.2.2 Gyro data

Hardware unit, 2 bytes for each axis, X, Y and Z in order.

5.2.3 Compass data

Hardware unit, 2 bytes for each axis, X, Y and Z in order.

5.2.4 ALS data

The information here is only about particular supported ALS.

Byte[0]: Dummy, Byte[2:1]: Ch0DATA, Byte[4:3]: Ch1DATA, Byte[6:5]: PDATA, Byte[7]: Dummy

Please contact your field representative as for supported ALS chip.

5.2.5 6-Axis quaternion data

5.2.6 9-Axis quaternion data

5.2.7 6-Axis pedometer quaternion data

5.2.8 Geomag rv data

The 6-Axis and 9-axis Quaternion outputs each consist of 12 bytes of data. These 12 bytes in turn consists of three 4-byte elements. 9-axis quaternion data and Geomag rv is always followed by 2-bytes of heading accuracy, hence the size of Quat9 and Geomag data size in the FIFO is 14 bytes. Quaternion data for both cases is cumulative/integrated values. For a given quaternion Q, the ordering of its elements is {Q_1, Q_2, Q_3}. Each element is represented using Big Endian byte order. Q_0 value is computed from this equation: $Q^2_0 + Q^2_1 + Q^2_2 + Q^2_3 = 1$. In case of drift, the sum will not add to 1, therefore, quaternion data need to be corrected with right bias values. The quaternion data is scaled by 2^30.

5.2.9 Pressure data

The information here is only about particular supported pressure sensor.

Byte [2:0]: Pressure data, Byte [5:3]: Temperature data

Please contact your field representative as for supported pressure sensor chip.

5.2.10 Calibrated Gyro data

Hardware unit scaled by 2^15, 4 bytes for each axis, X, Y and Z in order.

5.2.11 Calibrated Compass data

The unit is uT scaled by 2^16, 4 bytes for each axis, X, Y and Z in order.

5.2.12 Step Detector data

The data is 4 bytes timestamp as DMP cycle.

5.2.13 Accel Accuracy data

The accuracy is expressed as 0~3. The lowest is 0 and 3 is the highest.

5.2.14 Gyro Accuracy data

The accuracy is expressed as 0~3. The lowest is 0 and 3 is the highest.

5.2.15 Compass Accuracy data

The accuracy is expressed as 0~3. The lowest is 0 and 3 is the highest.

5.2.16 Fsync data

The data is delay time between Fsync event and the 1st ODR event after Fsync event.

5.2.17 Pick up data

The value “2” indicates pick up is detected.

5.2.18 Activity Recognition data

The data include Start and End states, and timestamp as DMP cycle.

Byte [0]: State-Start, Byte [1]: State-End, Byte [5:2]: timestamp.

The states are expressed as below.

Drive:	0x01
Walk:	0x02
Run:	0x04
Bike:	0x08
Tilt:	0x10
Still:	0x20

5.2.19 Secondary On/Off data

BAC algorithm requires sensors on/off through FIFO data to detect activities effectively and save power.

The driver is expected to control sensors accordingly.

The data indicates which sensor and on or off as below.

Gyro Off:	0x01
Gyro On:	0x02
Compass Off:	0x04
Compass On:	0x08
Proximity Off:	0x10
Proximity On:	0x20

6 Appendix I: DMP register addresses

The above DMP related configuration register settings may change from release to release. Please refer to this appendix section for the current definitions.

```
// data output control
#define DATA_OUT_CTL1                  (4 * 16)
#define DATA_OUT_CTL2                  (4 * 16 + 2)
#define DATA_INTR_CTL                  (4 * 16 + 12)
#define FIFO_WATERMARK                 (31 * 16 + 14)

// motion event control
#define MOTION_EVENT_CTL          (4 * 16 + 14)

// indicates to DMP which sensors are available
#define DATA_RDY_STATUS                (8 * 16 + 10)

// batch mode
#define BM_BATCH_CNTR                  (27 * 16)
#define BM_BATCH_THLD                  (19 * 16 + 12)
#define BM_BATCH_MASK                  (21 * 16 + 14)

// sensor output data rate
#define ODR_ACCEL                      (11 * 16 + 14)
#define ODR_GYRO                       (11 * 16 + 10)
#define ODR_CPASS                      (11 * 16 +  6)
#define ODR_ALS                        (11 * 16 +  2)
#define ODR_QUAT6                      (10 * 16 + 12)
#define ODR_QUAT9                      (10 * 16 +  8)
#define ODR_PQUAT6                     (10 * 16 +  4)
#define ODR_GEOMAG                     (10 * 16 +  0)
#define ODR_PRESSURE                   (11 * 16 + 12)
#define ODR_GYRO_CALIBR                (11 * 16 +  8)
#define ODR_CPASS_CALIBR               (11 * 16 +  4)

// sensor output data rate counter
#define ODR_CNTR_ACCEL                 (9 * 16 + 14)
```

```
#define ODR_CNTR_GYRO                  (9 * 16 + 10)
#define ODR_CNTR_CPASS                 (9 * 16 +  6)
#define ODR_CNTR_ALS                   (9 * 16 +  2)
#define ODR_CNTR_QUAT6                 (8 * 16 + 12)
#define ODR_CNTR_QUAT9                 (8 * 16 +  8)
#define ODR_CNTR_PQUAT6                (8 * 16 +  4)
#define ODR_CNTR_GEOMAG                (8 * 16 +  0)
#define ODR_CNTR_PRESSURE              (9 * 16 + 12)
#define ODR_CNTR_GYRO_CALIBR    (9 * 16 +  8)
#define ODR_CNTR_CPASS_CALIBR   (9 * 16 +  4)


// mounting matrix
#define CPASS_MTX_00        (23 * 16)
#define CPASS_MTX_01        (23 * 16 + 4)
#define CPASS_MTX_02        (23 * 16 + 8)
#define CPASS_MTX_10        (23 * 16 + 12)
#define CPASS_MTX_11        (24 * 16)
#define CPASS_MTX_12        (24 * 16 + 4)
#define CPASS_MTX_20        (24 * 16 + 8)
#define CPASS_MTX_21        (24 * 16 + 12)
#define CPASS_MTX_22        (25 * 16)


// bias calibration
#define GYRO_BIAS_X         (139 * 16 +  4)
#define GYRO_BIAS_Y         (139 * 16 +  8)
#define GYRO_BIAS_Z         (139 * 16 + 12)
#define ACCEL_BIAS_X        (110 * 16 +  4)
#define ACCEL_BIAS_Y        (110 * 16 +  8)
#define ACCEL_BIAS_Z        (110 * 16 + 12)
#define CPASS_BIAS_X        (126 * 16 +  4)
#define CPASS_BIAS_Y        (126 * 16 +  8)
#define CPASS_BIAS_Z        (126 * 16 + 12)
```

```
// Accel FSR
#define ACC_SCALE             (30 * 16 + 0)
#define ACC_SCALE2            (79 * 16 + 4)


// pedometer
#define PEDSTD_STEPCTR        (54 * 16)
#define PEDSTD_TIMECTR        (60 * 16 +  4)


// Activity Recognition
#define BAC_RATE              (48  * 16 + 10)


// parameters for accel calibration
#define ACCEL_CAL_RATE        (94 * 16 + 4)
#define ACCEL_ALPHA_VAR       (91 * 16)
#define ACCEL_A_VAR           (92 * 16)


// parameters for compass calibration
#define CPASS_TIME_BUFFER        (112 * 16 + 14)


// gains
#define ACCEL_ONLY_GAIN        (16 * 16 + 12)
#define GYRO_SF                (19 * 16)
```

7 Appendix II: GYRO_SF calculation

Read `TIMEBASE_CORRECTION_PLL` register, and correct the sing following InvenSense Register Map documents.

Utilize the function below with a parameter *pll* collected above.

The return value of this function is to be written to ***GYRO_SF***.

```
#define BASE_SAMPLE_RATE        1125
#define MPU_DEFAULT_DMP_FREQ    225
#define DMP_DIVIDER             (BASE_SAMPLE_RATE / MPU_DEFAULT_DMP_FREQ)

static int inv_calc_gyro_sf(s8 pll)
{
    int a, r;
    int value, t;

    t = 102870L + 81L * pll;
    a = (1L << 30) / t;
    r = (1L << 30) - a * t;
    value = a * 797 * DMP_DIVIDER;
    value += (s64) ((a * 1011387LL * DMP_DIVIDER) >> 20);
    value += r * 797L * DMP_DIVIDER / t;
    value += (s32) ((s64) ((r * 1011387LL * DMP_DIVIDER) >> 20)) / t;
    value <<= 1;

    return value;
}
```