



# **SDLog for Shimmer3**

## **Firmware User Manual**

### **Rev 0.19a**

## Legal Notices and Disclaimer

*Redistribution IS permitted provided that the following conditions are met:*

*Redistributions must retain the copyright notice, and the following disclaimer. Redistributions in electronic form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the document.*

*Neither the name of Shimmer Research, or Realtime Technologies Ltd. nor the names of its contributors may be used to endorse or promote products derived from this document without specific prior written permission.*

**THIS DOCUMENT IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS DOCUMENT, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.**

## Table of Contents

<b>1. Introduction .....</b>	<b>4</b>
<b>2. Changes in latest release .....</b>	<b>5</b>
2.1. Bug Fixes .....	5
2.2. New Features .....	5
<b>3. Scope of this User Manual .....</b>	<b>5</b>
<b>4. Pre-Requisites .....</b>	<b>5</b>
<b>5. Installation .....</b>	<b>6</b>
<b>6. Firmware features .....</b>	<b>7</b>
6.1. Configuration .....	7
6.2. Calibration parameters .....	14
6.3. Real-world clock .....	17
6.4. Time synchronisation .....	17
6.5. Start/Stop logging .....	21
6.6. LED indicators .....	23
<b>7. Reading the SD card Data .....</b>	<b>26</b>
7.1. SD card directory structure .....	26
7.2. Parsing the data .....	26
<b>8. Synchronising the data .....</b>	<b>35</b>
8.1. Aligning data to a common clock .....	35
<b>9. Appendices .....</b>	<b>39</b>
9.1. Known bugs .....	39
9.2. InfoMem contents .....	39
9.3. SD data file configuration header - sensor calibration bytes .....	45
9.4. Shimmer3 sensor conflicts .....	50
9.5. Example sdlog.cfg file for Shimmer3 .....	51
9.6. MPL calibration file contents .....	53
9.7. Legacy Firmware: SDLog v0.12.0 (or earlier): Configuration header .....	55
9.8. As with the normal calibration Legacy Firmware: SDLog v0.12.0 (or earlier): InvenSense MPL calibration .....	55
9.9. Legacy Firmware: SDLog v0.12.0 (or earlier): Writing calibration parameters to file .....	55


9.10.	Legacy Firmware: SDLog v0.11.0 (or earlier) data log.....	57
9.11.	Legacy Firmware: SDLog v0.8.0 (or earlier) synchronisation.....	57
9.12.	Legacy Firmware: SDLog v0.5.0 file contents .....	60
9.13.	Troubleshoot.....	68

## 1. Introduction

This document is an accompaniment to the *SDLog Firmware v0.19.0* image and later for *Shimmer3*. No previous development experience is required.

*SDLog Firmware* is a firmware image which allows logging of data from a Shimmer to the on-board SD card. The firmware allows full user configuration of the Shimmer via a configuration file, stored on the SD card. Many useful features, such as time synchronisation among multiple devices, start/stop logging on one or more devices by a single button press and user-defined naming of devices are enabled by this firmware image.

It is recommended that *SDLog Firmware* be used in conjunction with the *ConsensysBASIC* and/or *ConsensysPRO* software applications, which are available for download from the Shimmer [website](http://www.shimmersensing.com/products/consensys)<sup>1</sup>. The release of *SDLog Firmware v0.19.0* coincides with the latest release of the *ConsensysBASIC* and *ConsensysPRO* applications (greater than *Consensys v1.4.0*).

 Whenever the warning symbol appears throughout this document, it denotes a new, modified or deprecated feature in v0.9.0 (or later).

**Note:** There was a significant change to the *SDLog* configuration header and the data format in *SDLog Firmware* from v0.6.0 onwards. Users of *SDLog Firmware v0.5.0* should refer to the Appendices section of this document for parsing information. Users of *SDLog Firmware v0.5.0* are advised to upgrade to the latest version of the *SDLog Firmware* to ensure support for future Shimmer3 offerings. Versions of *SDLog Firmware* prior to v0.5.0 are unsupported by this document and are unsupported in applications provided by Shimmer.

**Note:** See section 9.1 for a number of known, outstanding bugs with *SDLog Firmware* that will be fixed in a future revision of the firmware.

---

<sup>1</sup> <http://www.shimmersensing.com/products/consensys>

## 2. Changes in latest release

This release is a major update to v0.18.0 with the following updates/improvements:

### 2.1. Bug Fixes

None

### 2.2. New Features

New SR47-4 (ExG) devices now supported. ADS1292R clock lines are reconfigured for this device such that both ADS chips share the same clock line. Interrupt for Chip 2 is disabled such that both chips are serviced by chip 1's data ready pin interrupt routine alone. Default configuration bytes now also cater for SR47-4 (and newer) Shimmer ExG devices. Checks EEPROM info and applies ADS clock chip configuration bit accordingly.

An additional 1 second time-out is now applied in firmware- see page 63 of ADS1292R datasheet for more info.

#### NOTE

An issue was found in SDLog (also spanning previous firmware versions) whereby a Shimmer configured to record ExG data (test signal used) would present aperiodic spikes in the output data if the Shimmer was not docked before logging began. After debugging and cross-comparison with LogAndStream (where the issue is not present), no immediate resolution could be made.

## 3. Scope of this User Manual

The purpose of this *User Manual* is to guide the user through the key features of the *SDLog Firmware* image and to provide the required instructions to configure the data logging options and to parse the recorded data. The *User Manual* does not provide an extensive explanation of the source code for the firmware.

## 4. Pre-Requisites

The *SDLog Firmware* can be used with any *Shimmer3* device that has a connected microSD card. For *Shimmer3* compatibility, the chosen microSD card must be less than 32 GB in capacity, not be an XC (eXtended Capacity) card and it must support 1-bit SPI mode. Please refer to the Shimmer3 MicroSD Media Guide on our [website](http://www.shimmersensing.com)<sup>2</sup> for more information.

**Note:** Please see [SDLog\\_Shimmer3\\_v0.19.0](http://www.shimmersensing.com/support/wireless-sensor-networks-documentation/category/12) for full release notes and accompanying firmware.

---

<sup>2</sup> <http://www.shimmersensing.com/support/wireless-sensor-networks-documentation/category/12>

A *Shimmer Dock* or *Consensys Base* is required to allow PC access to the Shimmer's SD card for either configuration of logging preferences or data transfer. Please note that legacy (black) Shimmer docks, sold with Shimmer2r and earlier hardware, are not suitable for this purpose.

## 5. Installation


Install the *SDLog* v0.19.0 Firmware image (SDLog\_Shimmer3\_v0.19.0.txt) onto a *Shimmer3* device, using the *ConsensysBASIC* or *ConsensysPRO* software applications, available on our [website](#)<sup>1</sup>.

## 6. Firmware features


This section outlines the important information required for using *SDLog Firmware*, including how to configure the device and experiment settings, configure calibration parameters, how the synchronisation procedure works, how to start and stop logging and what the LED indicators mean. Details about how the data is stored and about how to interpret the synchronisation data are provided in Sections 7 and 8, respectively.

### 6.1. Configuration

To use this firmware image, the user must provide the desired configuration parameters of each Shimmer. It is recommended that Shimmer software applications (e.g. *ConsensysBASIC* or *ConsensysPRO*) are always used for configuration of sensors for logging, in order to avoid errors introduced in the development of third party software. However, the configuration parameters can be sent to the Shimmer outside of the provided applications, by following the guidelines throughout this section.

 Since v0.9.0, there are two options available for configuring the Shimmer, as outlined below.

#### 6.1.1. Configuration via UART

 Configuration via UART was a new feature introduced in v0.9.0 (or later).

Configuration parameters can be written directly to the non-volatile memory (InfoMem) on the Shimmer3 via the UART interface, using a *Shimmer Dock* or *Consensys Base*.

To write the configuration parameters via UART, the following must be specified:

- The memory address of the first byte.
- The number of bytes to be written.
- The values to be written to the relevant bytes.

Please refer to Section 9.2 in the Appendices of this document for a description of InfoMem contents and refer to the information in the following subsections for more details regarding the valid values and descriptions of each parameter.

If the configuration parameters are modified via UART, the 'InfoMem changed' flag should be set to true to notify the firmware that the configuration file on the SD card (see Section 6.1.2 below) and the InfoMem do not hold the same configuration parameters. Refer to Section 9.2 for a description of which bit in InfoMem holds the value of this flag.

#### 6.1.2. Configuration by file

The configuration parameters can be saved in a configuration file, named *sdlog.cfg*, on the Shimmer's on-board SD card, from which they will be loaded by the Shimmer upon initialisation. Configuration parameters are read from the *sdlog.cfg* file or from the InfoMem according to the following:

- If the 'InfoMem changed' flag (see Section 6.1.1 above) is set, the firmware will read the configuration from InfoMem and will replace any existing *sdlog.cfg* file.



- If the 'InfoMem changed' flag is not set, the firmware will look for a configuration file on the SD card. Assuming that the file exists, the configuration will be read and copied to InfoMem.
- If the 'InfoMem changed' flag is not set and no configuration file exists on the SD card, the firmware will check that there is a valid configuration on the InfoMem. Assuming that there is a valid configuration, a new configuration file will be written with the configuration parameters from InfoMem.
- If the 'InfoMem changed' flag is not set, no configuration file exists on the SD card, and there is not a valid configuration on the InfoMem, the firmware will indicate an error (see Section 6.6 for LED indicators).

Thus, to change the configuration via the *sdlog.cfg* file, simply modify the *sdlog.cfg* file and reboot the Shimmer (there is no need to reprogram).

The configuration file must be saved in the top level directory of the SD card (i.e. not within a subfolder). This file allows configuration of all firmware features, as described throughout the following subsections. Each new line of the configuration file contains a single configuration parameter command. Users should note that the commands should be written **exactly** as they appear in the following sections so that they will be successfully parsed by the firmware. Any extra whitespaces will cause parsing errors.

If the configuration file has a glitch, such that the firmware cannot read it, the red/yellow LED indicators on the Shimmer will intermittently flash to warn the user of the error (see Section 6.6 for LED indicators). Note that this error will not occur in the event of parsing errors (e.g. typographical errors, extra whitespace, and repeated commands); it is the user's responsibility to ensure that the *sdlog.cfg* file is well-written. Commands with typographical errors and similar will be ignored by the firmware.

**Note:** The above SD error LED sequence is present in SDLog\_v0.17.0 and later. Older versions of SDLog couple both SD and RTC errors into a 0.1s Blue/0.1s Green LED flashing sequence. See Section 6.6 for full details.

It is recommended to use the *ConsensysBASIC* or *ConsensysPRO* application for configuration of Shimmer devices, to avoid errors in writing the configuration file. However, the configuration file can be written in any text editor and saved to the SD card without using Shimmer software, by following the guidelines throughout this section. ConsensysBASIC and ConsensysPRO are available for download from the Shimmer [website](http://www.ShimmerSensing.com)<sup>1</sup>.

### 6.1.3. Enabling sensors

#### Standard Shimmer sensors

The user may choose which sensors are enabled and disabled on the Shimmer for a particular experiment. These can be enabled or disabled by setting the value of the appropriate bits in InfoMem or by modifying the configuration file. The configuration file should contain a new line for each sensor enable/disable command. If any sensor is not included in the configuration file, it will be assumed to be disabled.

The currently available sensors and their corresponding *sdlog.cfg* entries are listed in Table 6-1 for *Shimmer3*.

Sensor		Enable command	Disable command
Low Noise Accelerometer		accel=1	accel=0
Gyroscope		gyro=1	gyro=0
Magnetometer		mag=1	mag=0
Wide Range Accelerometer		accel_d=1	accel_d=0
Battery voltage		vbat=1	vbat=0
External Expansion Channel 7		extch7=1	extch7=0
External Expansion Channel 6		extch6=1	extch6=0
External Expansion Channel 15		extch5=1	extch15=0
Internal Expansion Channel 1		intch1=1	intch1=0
Internal Expansion Channel 12		intch12=1	intch12=0
Internal Expansion Channel 13		intch13=1	intch13=0
Internal Expansion Channel 14		intch14=1	intch14=0
GSR		gsr=1	gsr=0
Pressure & Temperature		pres_bmp180=1	pres_bmp180=0
Pulse/PPG		intch13=1 and exp_power=1	intch13=0 and exp_power=0
ECG	24-bit	exg1_24bit=1 exg2_24bit=1	exg1_24bit=0 exg2_24bit=0
	16-bit	exg1_16bit=1 exg2_16bit=1	exg1_16bit=0 exg2_16bit=0
EMG	24-bit	exg1_24bit=1	exg1_24bit=0
	16-bit	exg1_16bit=1	exg1_16bit=0
Bridge Amplifier		br_amp=1	br_amp=0

Table 6-1: Sensor enable/disable commands for Shimmer3

**Note:** To enable the ExG Test Signal to verify the correct operation of the *Shimmer3* ECG or EMG module see the *ECG User Guide* or the *EMG User Guide*, available on our [website](http://www.shimmersensing.com/support/wireless-sensor-networks-documentation/category/12)<sup>3</sup>.

Certain combinations of sensors cannot be enabled simultaneously due to hardware or firmware restrictions. A table listing the *Shimmer3* sensor conflicts is given in Table 9- in Appendix 9.4.

<sup>3</sup> <http://www.shimmersensing.com/support/wireless-sensor-networks-documentation/category/12>

## 6.1.4. Sensor parameters

### Standard Shimmer sensors

See Table 6-2 for sensor configuration parameters and corresponding configuration file commands.

Parameter	Example command	Valid options		Default
Accel range	acc_range=0	<b>LSM303DLHC</b>	<b>LSM303AHTR</b>	0 (± 2.0 g)
		0 (± 2.0 g)	0 (± 2.0 g)	
		1 (± 4.0 g)	2 (± 4.0 g)	
		2 (± 8.0 g)	3 (± 8.0 g)	
		3 (± 16.0 g)	1 (± 16.0 g)	
Gyro range	gyro_range=0	<b>MPU9150</b>	<b>MPU9250</b>	0 (±250 dps)
		0 (±250 dps) 1 (±500 dps) 2 (±1000 dps) 3 (±2000 dps)		
Mag range	mg_range=1	<b>LSM303DLHC</b>	<b>LSM303AHTR</b>	1 (±1.3 Ga) or 0 (± 49.152Ga)  depending on board type
		1 (± 1.3 Ga)	0 (± 49.152Ga)	
		2 (± 1.9 Ga)		
		3 (± 2.5 Ga)		
		4 (± 4.0 Ga)		
		5 (± 4.7 Ga)		
		6 (± 5.6 Ga)		
		7 (± 8.1 Ga)		
Accel internal data rate	acc_internal_rate=1	<b>LSM303DLHC</b>	<b>LSM303AHTR</b>	5 (100 Hz) or 4 (100Hz)  depending on board type
		0 (power down)	0 (power down)	
		1 (1.0 Hz)	1 (12.5 Hz)	
		2 (10 Hz)	2 (25 Hz)	
		3 (25 Hz)	3 (50 Hz)	
		4 (50 Hz)	4 (100 Hz)	
		5 (100 Hz)	5 (200 Hz)	
		6 (200 Hz)	6 (400 Hz)	
		7 (400 Hz)	7 (800 Hz)	
		8 (1.620 kHz)	8 (1.600 kHz)	
		9 (1.344 kHz)	9 (3.200 kHz)	
			10 (6.400 kHz)	
Gyro internal data rate	gyro_samplingrate=0	<b>MPU9150</b>	<b>MPU9250</b>	155 (51.28 Hz)
		0 – 255 (8000/value -1 Hz)		
Mag internal data rate	mg_internal_rate=0	<b>LSM303DLHC</b>	<b>LSM303AHTR</b>	6 (75 Hz) Or 3 (100.0 Hz)  depending on board type
		0 (0.75 Hz)	0 (10.0 Hz)	
		1 (1.5 Hz)	1 (20.0 Hz)	
		2 (3.0 Hz)	2 (50.0 Hz)	
		3 (7.5 Hz)	3 (100.0 Hz)	
		4 (15.0 Hz)		
		5 (30.0 Hz)		
		6 (75.0 Hz)		
	7 (220.0 Hz)			
Accel Low Power Mode	acc_lpm=0	0 (disabled) 1 (enabled)		0 (disabled)
Accel High Res Mode	acc_hrm=0	0 (disabled)		0 (disabled)

		1 (enabled)	
<b>GSR range</b>	<code>gsr_range=0</code>	0 (10 kΩ – 56 kΩ) 1 (56 kΩ – 220 kΩ) 2 (220 kΩ – 680 kΩ) 3 (680 kΩ – 4.7 MΩ) 3 (Auto Range)	4 (Auto Range)
<b>Pressure Resolution</b>	<code>pres_bmp180_prec=0</code>	0 (Low) 1 (Standard) 2 (High) 3 (Very High)	0 (Low)

Table 6-2: Sampling configuration options for Shimmer3

**Note:** See *ECG User Guide* or *EMG User Guide* for ExG gain and rate settings.

### 6.1.5. Experiment settings

In addition to the sensor-specific parameters listed in the previous subsections, configuration parameters for an experiment or data collection can be configured using the commands in Table 6-3. Further details describing these parameters are provided below the table.

Parameter	Example command	Valid options	Default
<b>Sampling rate (in Hz)</b>	<code>sample_rate=51.2</code>	0, ..., 1024	51.2
<b>Number of Shimmers</b>	<code>Nshimmer=1</code>	positive integer value: 1, ..., 255	1
<b>Shimmer ID number</b>	<code>myid=12</code>	positive integer value: 1, ..., Nshimmer	1
<b>Shimmer name</b>	<code>shimmername=shimmer1</code>	string with up to 11 characters	IDxxxx
<b>Experiment ID</b>	<code>experimentid=expid</code>	string with up to 11 characters	-
<b>Configuration ID</b>	<code>configtime=1234567890</code>	any 32-bit signed integer value	0
<b>User button</b>	<code>user_button_enable=1</code>	0 (disabled) 1 (enabled)	1
<b>Single-touch</b>	<code>singletouch=0</code>	0 (disabled) 1 (enabled)	0
<b>Synchronisation</b>	<code>sync=0</code>	0 (no time synchronisation) 1 (master-slave synchronisation)	0
<b>Master</b>	<code>iammaster=0</code>	0 (slave) 1 (master)	0
<b>Master ID</b>	<code>center=0006664c34ef</code>	12-digit MAC address of the Bluetooth radio on the master device.	-
<b>Slave Node ID</b>	<code>node=0006664c36ad</code>	12-digit MAC address of the Bluetooth radio on the slave device (node). Include a new line for each slave node.	-
<b>Estimated Experiment Length</b>	<code>est_exp_len=0</code>	positive integer value: 0, ..., 65535	0
<b>Maximum Experiment Length</b>	<code>max_exp_len=0</code>	positive integer value: 0, ..., 65535	0

Table 6-3: Logging configuration options.

The **Sampling rate** is the frequency at which sampling should be performed. It should be noted that the actual sampling rate may not be exactly equal to this parameter because the firmware requires the sampling period to be an integer multiple of 1/32768 s. For example, a desired sampling

frequency of 500 Hz requires a sampling period of 2 ms which is equal to  $65.536 \times (1/32768)$  s. In the firmware, this sampling period will be rounded up to  $66 \times (1/32768) = 2.014$  ms, so the true sampling rate will be 496.48 Hz. The actual sampling rate that will be used (*True Fs*) can be calculated from the specified sampling rate (*Fs*) using the following formula:

$$\text{True } Fs = 32768 / (\text{round}(32768 / Fs)),$$

where the round() function means rounding to the nearest integer.

**Number of Shimmers** and **Shimmer ID number** are used to describe the total number of Shimmers in an experiment and the ID number of each individual Shimmer. These parameters do not have any explicit function in firmware.

The **Shimmer name** parameter allows the user to specify a meaningful name for each device. It is used by the firmware to name the data directories on the SD card. For more information on the directory structure, see Section 7.1.

The **Experiment ID** parameter allows the user to specify a meaningful name for an experiment. It is used by the firmware to name the data directories on the SD card. For more information on the directory structure, see Section 7.1.

The **Configuration ID** is a numeric identifier for the experiment. It typically refers to the date and/or time at which the configuration file was created (but users may choose any valid numeric value that is meaningful to them<sup>4</sup>). It is appended to the experiment ID name in the data directories to allow the user to identify different experiments in the case that the same experiment ID name is used multiple times. It is vital that this parameter is equal for all Shimmers in the experiment if synchronisation or single-touch start is enabled.

The **User button** refers to the button on the baseboard which is accessible by pressing the circular orange button on the *Shimmer3* enclosure. If the button is enabled, then logging will begin when the user button is pressed. Logging will finish when any one of the following happens: the user button is pressed again; the Shimmer battery runs out; the Shimmer is placed on the *Shimmer Dock*.

**Single-touch start** allows logging to be started on multiple Shimmers, in response to a button press on a single Shimmer. If this setting is enabled, exactly one Shimmer must be designated as the *Master* and all others are slaves and the user button must be enabled. It is vital that the *Configuration ID* parameter (described below) is equal for all Shimmers in the experiment if single-touch start is enabled. For more details describing this feature, see Section 6.5.

The **Synchronisation** setting allows samples from multiple Shimmers to be synchronised in time. If this setting is enabled, exactly one Shimmer must be designated as *Master* and the others as *Slaves* (see *Master* setting below). It is vital that the *Configuration ID* parameter (described below) is equal

---

<sup>4</sup> Note that using a value other than the *Configuration ID* automatically generated by Shimmer software applications will likely cause some confusion if the logged data is subsequently imported into Shimmer software, like *Consensus*, which assume that the value is related to the configuration date and time.

for all Shimmers in the experiment if synchronisation is enabled. For more information on this feature, see Section 6.4.


If **Master** is enabled, the device will assume the role of master for synchronisation of timestamps and/or for the single-touch start feature (described below). Otherwise, the device will assume the role of slave.

**Note:** if a Shimmer software application (e.g. *Consensus*) is not used to configure the experiment, it is the user's responsibility to ensure that there is exactly one master and all other devices are slaves. If multiple Shimmers are acting as master, behaviour of the slaves will not be as expected.


The **Master ID** parameter is used to specify the MAC Address of the Bluetooth radio on the master device (center). This parameter does not have any specific functionality in firmware but is used by Shimmer software applications to identify which device was the master for a given trial.

The **Slave Node ID** parameter is used to specify the MAC Address of the Bluetooth radio on a slave device (node). In the configuration file, this command should be repeated on a new line for each slave node. A maximum of twenty (20) nodes may be synchronised by a single master.

**Note:** if a Shimmer software application (e.g. *Consensus*) is not used to configure the experiment, it is the user's responsibility to ensure that the master has the MAC addresses of all of the slaves' Bluetooth modules stored as the value for the *Slave Node ID* parameters in its *sdlog.cfg* file.

 In *SDLog Firmware* prior to v0.9.0, each slave saved the MAC address of the Master Shimmer device in the *Master MAC Address* parameter. This has been deprecated since v0.9.0 and replaced by the *Slave Node ID* parameters.

The **Estimated Experiment Length** parameter is used to determine how often communication between master and slaves will be carried out for synchronisation. It should be set to the expected length of the logging session, in minutes. Master-slave communication will be attempted at three regularly-spaced intervals throughout the estimated experiment length, such that the interval between communication attempts will be the value of this parameter divided by three (3) or one hour, whichever is smaller. If the duration of the logging session exceeds the estimated length, communications will continue at the same interval. If unsure of the total experiment length, it is better to underestimate, rather than overestimate. If the value of this parameter is less than ten (10) minutes, communication will be attempted on a continuous basis; i.e. in this case, the interval is effectively zero.

 In *SDLog Firmware* prior to v0.9.0, the *Broadcast Interval* parameter was used to determine how often the master Shimmer sends a timestamp update broadcast. This has been deprecated since v0.9.0 and replaced by the *Estimated Experiment Length* parameter.

The **Maximum Experiment Length** parameter is used to automatically stop all sensors from logging data after the defined logging duration has been reached. It should be set to the maximum desired length of the logging session, in minutes. A value of zero (0) means that logging will continue indefinitely until manually terminated by the user (see Section 6.5 for ways to terminate logging).



## 6.2. Calibration parameters

There are two sets of calibration parameters that the *Shimmer3* uses - one set for the MPU9150 sensors when the *DMP™* is enabled and another set for all other sensors (e.g., wide-range accelerometer, low-noise accelerometer etc.). Calibration parameters for all sensors other than the *DMP™* related sensors are not used in this firmware image. They are saved in the configuration header along with the data, so that these data can be calibrated afterwards.

### 6.2.1. Providing calibration parameters

As with configuration parameters, there are two ways to provide calibration parameters in v0.19.0: by file *or* by writing to the InfoMem. Calibration parameters can also be provided via Bluetooth, through the use of *LogAndStream* firmware.

#### ***Writing calibration parameters to InfoMem via Bluetooth***

The recommended method of writing the calibration parameters to the Shimmer is to program the device with *SDlog\_v0.19.0* firmware and perform calibration using the *Shimmer 9DoF Calibration* application, which can be downloaded from the Shimmer [website](#)<sup>5</sup>. Assuming that *SDlog\_v0.19.0* firmware is programmed on the devices during calibration<sup>6</sup>, the new calibration parameters will automatically be written to appropriate files on the SD card, following the file structure outlined later in this section.



#### ***Writing calibration information to file***

Users should note that this is not a recommended method of providing calibration parameters for the Shimmer when using *SDLog v0.19.0*.

To supply calibration information by file, the user should create a folder called "Calibration" or "calibration" in the top-level SD card directory and should save the calibration file within that folder. The format of the file should match the output given by the *Shimmer 9DoF Calibration* application's "Save to file" function.

Since *SdLog v0.13.0* onwards, the Shimmer firmware uses a new format of calibration file: the "Calibration Dump". This file stores all calibration information, including all ranges of all kinematic sensors, in the same file in hex format. The file name is "calib\_xxxx" where xxxx is the ID of the Shimmer device. Backwards compatibility has NOT been included, i.e. the users have to re-calibrate the shimmers before the new calibration system can be functional.

Calibration information is read from the calibration folder or from the InfoMem according to the following:

- If the calibration dump file is available, the firmware will read the calibration information from the file, and updates the RAM, the InfoMem and the data header.

---

<sup>5</sup> [www.shimmersensing.com/support/wireless-sensor-networks-download/category/21](http://www.shimmersensing.com/support/wireless-sensor-networks-download/category/21)

<sup>6</sup> If an earlier version of *LogAndStream* firmware is programmed on the Shimmer during calibration, the calibration parameters will likely be lost during reprogramming with *SDLog* firmware. This situation should be avoided at all times.

- If the calibration dump file is not available, the firmware will generate a default calibration dump in the RAM. If the InfoMem is available, the firmware will replace the corresponding ranges of the sensors in the RAM with the InfoMem values. Then the firmware will create a new calibration dump file "calibration\calib\_XXXX" and update the data header with the RAM values.

The calibration timestamp will be all 0 by default, the calibration parameters are given in *Table 6-4*.

	Offset (all axes)		Sensitivity (all axes)		Alignment Matrix
<b>LN Accelerometer</b>	Kionix KXRB5-2042	Kionix KXTC9-2050	Kionix KXRB5-2042	Kionix KXTC9-2050	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}$
	2047 LSB	2253 LSB	$\pm 2 \text{ g} \rightarrow 83 \text{ LSB}/(\text{m/s}^2)$	$\pm 2 \text{ g} \rightarrow 92 \text{ LSB}/(\text{m/s}^2)$	
<b>Gyroscope</b>	0		$\pm 250 \text{ dps} \rightarrow 131 \text{ LSB/dps}$ $\pm 500 \text{ dps} \rightarrow 65.5 \text{ LSB/dps}$ $\pm 1000 \text{ dps} \rightarrow 32.8 \text{ LSB/dps}$ $\pm 2000 \text{ dps} \rightarrow 16.4 \text{ LSB/dps}$		$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}$
<b>WR Accelerometer</b>	0		<b>LSM303DLHC</b>	<b>LSM303AHTR</b>	<b>LSM303DLHC</b>
			$\pm 2 \text{ g} \rightarrow 1631 \text{ LSB}/(\text{m/s}^2)$	$\pm 2 \text{ g} \rightarrow 1671 \text{ LSB}/(\text{m/s}^2)$	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & +1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$
			$\pm 4 \text{ g} \rightarrow 815 \text{ LSB}/(\text{m/s}^2)$	$\pm 4 \text{ g} \rightarrow 836 \text{ LSB}/(\text{m/s}^2)$	
			$\pm 8 \text{ g} \rightarrow 408 \text{ LSB}/(\text{m/s}^2)$	$\pm 8 \text{ g} \rightarrow 418 \text{ LSB}/(\text{m/s}^2)$	<b>LSM303AHTR</b>
			$\pm 16 \text{ g} \rightarrow 135 \text{ LSB}/(\text{m/s}^2)$	$\pm 16 \text{ g} \rightarrow 209 \text{ LSB}/(\text{m/s}^2)$	$\begin{bmatrix} 0 & -1 & 0 \\ +1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}$
<b>Magnetometer</b>	0		<b>LSM303DLHC</b>	<b>LSM303AHTR</b>	<b>LSM303DLHC</b>
			$\pm 1.3 \text{ Ga} \rightarrow 1100 \text{ LSB/Ga}$	$\pm 49.152 \text{ Ga} \rightarrow 667 \text{ LSB/Ga}$	
			$\pm 1.9 \text{ Ga} \rightarrow 855 \text{ LSB/Ga}$		$\begin{bmatrix} -1 & 0 & 0 \\ 0 & +1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$
			$\pm 2.5 \text{ Ga} \rightarrow 670 \text{ LSB/Ga}$		
			$\pm 4.0 \text{ Ga} \rightarrow 450 \text{ LSB/Ga}$		
			$\pm 4.7 \text{ Ga} \rightarrow 355 \text{ LSB/Ga}$		<b>LSM303AHTR</b>
			$\pm 5.6 \text{ Ga} \rightarrow 330 \text{ LSB/Ga}$		
			$\pm 8.1 \text{ Ga} \rightarrow 230 \text{ LSB/Ga}$		

*Table 6-4 Default calibration parameters set in firmware if no available calibration information*

Table's 5-8 and 5-9 help briefly introduce the structure of the calibration dump file. A default calibration dump file is of 538 bytes for *SDLog v0.19.0*, the included ranges of sensors follow the sequence in *Table 6-4*. The file structure supports more calibration parameters if necessary. The



structure of the calibration dump file is shown as in *Table 6-5*. The first two bytes are the Total\_Length bytes. This length is the total data length except the "Total\_Length" itself. i.e. if the file is 538 bytes long, including the first two Total\_Length bytes and 536 data bytes, then there should be Total\_Length=536 in this case. Following the Total\_Length bytes are 8 version info bytes. The sensor calibration timestamp and parameters per range are stored after the first 10 bytes in sequence.

Byte	content		
0	sys info	Total_Length	lsb
1			msb
2		Version Info (8 bytes)	hw_id_lsb
3			hw_id_msb
4			fw_id_lsb
5			fw_id_msb
6			fw_ver_major_lsb
7			fw_ver_major_msb
8			fw_ver_minor
9			fw_ver_internal
10	sensor1, range1	sensor ID	lsb
11			msb
12		range	
13		Length=21	
14		Timestamp (8 bytes)	lsb
...			...
21			msb
22		parameters (21 bytes for example)	Byte 1
23			Byte 2
...			...
42			Byte 21
43	sensor1, range2	sensor ID (2 bytes)	lsb
44			msb
45		range	
...		...	

*Table 6-5 Calibration Dump file structure*

## 6.3. Real-world clock

In *SDLog v0.8.0*, the capability to save the 'real-world' time from a PC to the Shimmer was introduced. This feature allows simple synchronisation with external events or third party devices by providing a common clock timestamp at the beginning of a logging session.

The recommended method to set the real-world timestamp from a PC to a Shimmer device is to use Shimmer *Consensus* software. The time is set via the UART interface with *Shimmer Dock* or *Consensus Base*. In the case of a user who wishes to set the time outside of a Shimmer application, more details should be sought in the source code for *SDLog Firmware* on the [Shimmer github repository](https://github.com/ShimmerResearch/shimmer3/tree/master/SDLog)<sup>7</sup>.

### 6.3.1. Important considerations for real-world clock

Please note the following factors that will affect the reliability of the real-world clock timestamps.

1. The real-world time is saved in volatile memory. This means that, if the Shimmer is powered off or reset, the real-world time will be lost.
2. The real-world time setting is implemented as a once-off operation that relates the timestamp on the Shimmer's local clock to the timestamp on the PC. This setting cannot measure or correct for drift of the Shimmer clock (see Sections 6.4 and 8.1 for details on time synchronisation, which corrects for clock drift). It is recommended that a logging session should begin as soon as possible after the real-world clock has been set on the Shimmer. Similarly, if the PC time is being set on multiple devices (Shimmer or third party devices), this should be done on all devices within as short an interval as possible to reduce the effect of clock drift.
3. The firmware indicates if the real-world time has been set on the Shimmer. A blue/green LED indicator on the Shimmer will intermittently flash to indicate to the user that the real world time has not been set (see Section 6.6 for LED indicators).

**Note:** It is not critical for the real-world time setting to be present on the Shimmer in order to record data to the Shimmer's SD card. The LED sequence will change to the "logging" mode after undock start or push button start and will return to the error LED sequence when logging ceases (see Section 6.6 for LED indicators).

## 6.4. Time synchronisation

Each Shimmer device has its own independent on-board clock which is reset to zero whenever the device is powered on or reset and which is subject to drift, caused by component-specific differences and environmental factors, like temperature. Thus, in order to enable valid comparisons and joint processing methods for the samples logged on multiple shimmer devices simultaneously, the local timestamps from each device must be related to a common reference timeframe. The time synchronisation procedure in *SDLog Firmware* allows all samples to be referred to the clock of one single Shimmer - the "master". If the real-world' clock has been set on the master Shimmer (see

---

<sup>7</sup> <https://github.com/ShimmerResearch/shimmer3/tree/master/SDLog>

Section 6.3), the reference time will be related to the real-world clock. Otherwise, the reference time will be related to the master's local clock.

**⚠** In *SDLog Firmware v0.9.0 (or later)*, the synchronisation method has been significantly modified, relative to earlier firmware versions, in order to improve usability and robustness and to increase the battery life of the slave devices. This section describes the synchronisation procedure for v0.9.0 (or later). For earlier firmware versions, legacy information is contained in Appendix 9.11.

The important improvements in the synchronisation method are summarised below, along with their main benefits:

- Master device initiates communication with each slave in turn.
  - Avoid collisions among simultaneous communication attempts from multiple slaves.
  - Increase battery life for slaves by reducing the number of failed connection attempts.
  - Improve usability by handing control of communication success to master device, rather than user being responsible for coordinating 'radio ON' epochs on all devices.
- For each successful master-slave connection, multiple timestamps are shared.
  - Improve efficiency by maximising the amount of useful information gained from each successful connection.
  - Improve accuracy by gathering many samples of randomly varying communication latency and minimising error in recorded offsets.
  - Increase battery life by requiring less frequent communications.
- "Real-world time" can be used for synchronisation, instead of local time on the master device's clock (see Section 6.3).


Synchronisation requires exactly one Shimmer to be designated as the “master” Shimmer and any other Shimmers to be designated as “slaves”. Enabling time synchronisation allows samples from multiple Shimmers to be converted to a common reference clock (the clock of the master Shimmer).

It is important to note that, in this synchronisation method, no effort is made to alter the clock frequency of any Shimmer in the firmware. Instead, each clock runs at its own frequency throughout logging and the alignment of samples to a common reference clock is done in post-processing. The basis of the alignment is a log of offsets between each Shimmer’s local timestamp and master timestamps. On the *Shimmer3*, the master connects with each slave via Bluetooth (BT) at regular intervals and the master shares its timestamp at that instant with the connected slave. The slaves must be within radio range (approximately 10m) of the master for successful communication.

One important synchronisation parameter is the communication interval (denoted CI in the following). This is the interval at which master-slave communication attempts will be made. See Section 6.1.5 for more details on this parameter, which is derived from the *Estimated Experiment Length* configuration parameter.

#### 6.4.1. Procedure

1. When **each device** is powered on or reset, its on-board 32 kHz clock will begin counting at zero.

- a. If the 'real-world time' **is not set** on the master device, the master's local 32kHz clock's time will be the common reference time for all samples in the experiment.
  - b. If the 'real-world time' **is set** on the master device, the master's real-world time will be the common reference time for all samples in the experiment.
  - c. For synchronisation purposes, it is recommended but not necessary that the 'real-world time' be set on the slave devices.
2. The **master** maintains a list of all slave devices in the experiment (using the MAC addresses from the configuration) and uses this list as the synchronisation queue for each communication attempt.
3. When **each device** starts logging, it will save its initial timestamp with a resolution of 5 bytes (40 bits) - see Section 7.2.2 for a description on where this initial timestamp is saved.  
 In *SDLog Firmware v0.12.0 (or earlier)*, the initial timestamp was 4 bytes (32 bits)
4. When the **master** turns on its radio, it will generate the synchronisation queue from the list of all configured slave devices.
5. The **master** will attempt to connect with the first device in the synchronisation queue.
6. If the **master** fails to make a connection with a slave device, it moves that slave to the end of its synchronisation queue and continues to attempt connection with the next slave device.
7. When a connection is successfully made with a slave device, the **master** will send its timestamp with a resolution of 64 bits.
8. The **slave** will acknowledge receipt of the timestamp from the master and immediately record the time on its own clock. The difference between the slave's own timestamp and the received master timestamp (slave time - master time) is calculated, as follows:
  - a. The *offset sign* (one byte) represents the sign of the difference - 0 (positive) if slave time  $\geq$  master time and 1 (negative) otherwise.
  - b. The magnitude of the difference ( $|\text{local time} - \text{master time}|$ ), with a resolution of 64 bits, is represented by an 8-byte unsigned integer.
9. Steps 7 - 8 will be repeated multiple times until the slave has a sufficiently large sample set from which to accurately estimate the offset between its own clock and the master's clock, minimising the error due to randomly varying Bluetooth transmission latency. The **slave** saves the estimated offset to a memory location, from which it will be written to the SD card on the next data writing epoch.
10. When the **slave** indicates that it does not need any more samples, the **master** terminates connection with the slave device and removes that device from its synchronisation queue. The **slave** turns off its radio.

11. The **master** continues to attempt connection with slave devices, by repeating Steps 5 - 10 until either all devices have been removed from the synchronisation queue or 400 seconds have elapsed since it turned on its radio, whichever happens soonest.
12. The **master** turns off its radio.
13. Steps 0 - 12 are repeated until logging terminates.

#### **6.4.2. Important considerations for Synchronisation**

Please note the following factors that will affect the reliability and accuracy of master-slave synchronisation.

1. The range of the Bluetooth radio on the Shimmer is approximately 10 meters. All slave devices must be within this range of the master device.
2. Bluetooth requires a line-of-sight between sender and receiver. Ensure that there are no obstructions between master and slave devices. Human tissue is especially problematic for Bluetooth signals so avoid 'body-blocking' at all times.
3. The maximum number of devices that can be synchronised by a single master is 20.

## 6.5. Start/Stop logging

### 6.5.1. Dock (automatic start)

This is the default way for the Shimmer to start and stop logging data to the SD card and is enabled if none of the other options below are chosen. With this configuration, if the Shimmer is powered on or reset while it is **not** on the dock, the following process will begin immediately:

1. The Shimmer unit will go into standby mode for up to 3 seconds.
2. The Shimmer unit will read the configuration settings and create the required directories on the SD card.
3. The Shimmer unit will start logging.

Alternatively, if the Shimmer is powered on or reset while it **is** on the dock, the three steps above will begin as soon as the Shimmer is removed from the dock.

In either case, logging will continue until the Shimmer unit is reset, powered off, replaced in the dock or the battery runs out, whichever happens soonest.

Repeatedly resetting the Shimmer unit will result in multiple logging sessions on the SD card.

**Warning:** Please note that it is not recommendable to dock the Shimmer unit while it is being configured. Ideally, the Shimmer unit should either be powered off or in standby mode whenever it is being placed on the dock.

### 6.5.2. User button (manually start/stop logging)

This option relies on the orange user button on the *Shimmer3* enclosure to start/stop logging.

With this setting, undocking the Shimmer unit or powering it on and off will trigger the same steps 1 and 2 as above (i.e. standby for 3 seconds followed by configuration). However, logging will not start until the user button is pressed by the user.

If the user button is pressed while the Shimmer unit is on the dock, it will have no effect on logging.


If the user button is pressed while the Shimmer unit is undocked (and not logging), logging will start immediately.

Logging will continue until one of the following occurs: the user button is pressed, the Shimmer unit is reset, powered off, replaced in the dock or the battery runs out, whichever happens soonest.

Repeatedly pressing the user button and/or docking the Shimmer unit (to stop logging) will result in multiple logging sessions on the SD card.

### 6.5.3. Single-touch start/stop

This option requires exactly one Shimmer to be designated as the “master” Shimmer and any other Shimmers to be designated as “slaves”. This option allows logging to be started on all Shimmers by a single press of the user button on the master Shimmer.

 In *SDLog Firmware v0.9.0 (or later)*, the ability to stop logging on all Shimmers by a single press of the user button on the master Shimmer has been removed. This is due to incompatibility of this feature with the new synchronisation method implemented and, also, due to inconsistency in the success of the feature, due to the long interval between master-slave communication attempts.

With this configuration, if the Shimmer is powered on or reset while it is **not** on the dock, the following process will begin immediately:

1. The Shimmer unit will go into standby mode for up to 3 seconds.
2. The Shimmer unit will read the configuration settings and create the required directories on the SD card.
3. The slave devices will turn on their Bluetooth radios and await a command from the master device to start logging.
4. The master device will remain in 'standby' mode until the user presses its button.
5. When the master user button is pressed, the master will immediately start logging. Furthermore, it will initiate communication with all slave devices (see Section 6.4.1 for a description of how communication is controlled by the master) and send a "start-logging" message in all further Bluetooth communications with slaves, telling them to start logging.
6. When a slave device receives a "start-logging" command from the master, it will automatically start logging data.
7. Data will continue to be logged on all devices until its user button is pressed, the device is docked or the battery runs out, whichever happens soonest.

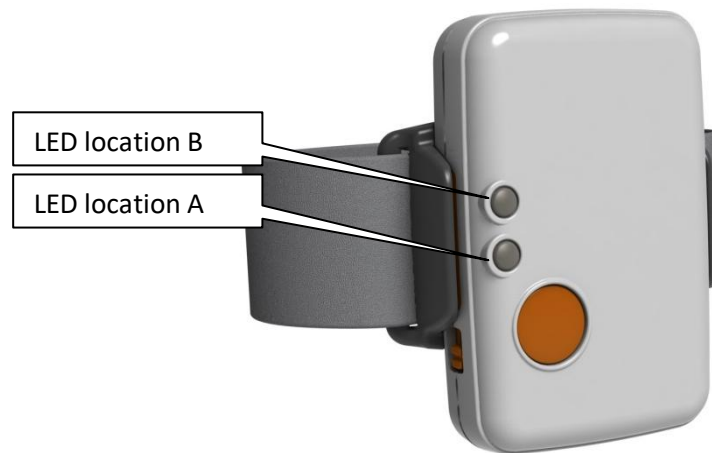
There are a number of important points to remember with this setting, as outlined below:

- If the **master** user button is pressed while the master Shimmer is on the dock, it will have no effect on logging for any device.
- If the **master** user button is pressed while the master Shimmer is undocked and logging, the master will immediately stop logging. No further communications will be made with slave devices.
- The user button is always enabled on the **slave** Shimmers in single-touch start configuration. This feature allows the user to manually start logging on the slave Shimmers, either to override the single-touch start option or as a backup, in case the "start-logging" messages from the master are missed by the slave.
- If a "start-logging" message is received by the **slave**, while it is already logging, the message will be ignored.
- Synchronisation is always enabled when single-touch start is enabled.



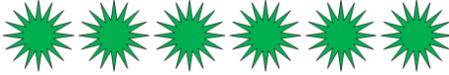




## 6.6. LED indicators

The *Shimmer3* has five LEDs in two locations: lower location A (green, yellow and red); upper location B (green , blue), as shown in *Figure 6-1*.



*Figure 6-1 Shimmer3 LED Locations*

The LEDs in Location A are used to indicate battery charge status, as outlined in Table 6-6.

		LED Pattern	Description
Docked or in Multi Charger	Full Charge		Green Solid ON
	Charging		Yellow Solid ON
Undocked	Full Charge		Green 0.1s ON/5s OFF
	Medium Charge		Yellow 0.1s ON/5s OFF
	Low Charge		Red 0.1s ON/5s OFF

*Table 6-6 SDLog Firmware Battery Charge Status Indication*



The LEDs in Location B are used to indicate operation status, as outlined in Table 6-7.

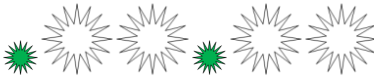
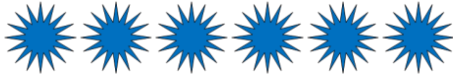






		LED Pattern	Description
Docked or Undocked	Standby		Green 0.1s ON/2s OFF
	Radio On		Blue Solid ON
	Slave has not synchronised with Master		Blue Solid ON
	Synchronising with another Shimmer		0.1s Blue/0.1s Green
	RTC not set		0.1s Blue/0.1s Green
Undocked only	Configuring		Green 0.1s ON/ 0.1s OFF
	SD error		0.1s Red/0.1s Yellow
	Logging		Green 1s ON / 1s OFF

Table 6-7 SDLog Firmware Operation Status Indication


**Note:** The Shimmer should never be placed in the Dock while the operation LEDs indicate that it is configuring as this may cause a file-system error. Once configuration has begun, you must power off or reset the Shimmer before docking.

**Note:** It is not advisable to place the Shimmer in the Dock while it is logging data as this can cause SD card access problems. Make sure that logging has stopped before docking the device.

**Note:** Logging and other SD card related operations are not carried out while the device is on the dock.

**Note:** The above SD error LED sequence is present in SDLog\_v0.17.0 and later. Older versions of SDLog couple both SD and RTC errors into a 0.1s Blue/0.1s Green LED flashing sequence.

**Note:** There are a couple of reasons for the Shimmer to enter "Error" mode.

1. If the real-world time has not been set on the Shimmer.
  - a. It is not critical for the real-world time setting to be present on the Shimmer in order to record data to the Shimmer's SD card. The LED sequence will change to the "logging" mode after undock start or push button start.  LED sequence for real-world time status only in SDLog v0.12.0 or later.
2. If no configuration file is present on the SD card.

3. If the Shimmer is unable to access the SD card for some reason e.g. if the SD card is corrupt (try reformatting the SD card but note that this will delete all of the data from the SD card)

## 7. Reading the SD card Data

It is recommended that the *Consensys* software, which can be downloaded from the Shimmer [website](http://www.shimmersensing.com/support/wireless-sensor-networks-download/category/19)<sup>8</sup> be used to read the data from the SD card. However, using the following guidelines, the user may use their platform of choice to read and parse the data.

### 7.1. SD card directory structure

The data directory structure on the SD card is as follows:

- A folder called "*data*" is created by the firmware in the SD card top-level directory if it does not already exist.
- A subfolder is created within the *data* folder each time a new "*experiment id*" is encountered in the *sdlog.cfg* file (default is "*default\_exp*").
- A new subfolder is created within the appropriate experiment folder each time logging starts. The naming convention for this folder is the Shimmer name specified in the *sdlog.cfg* file (default is the abbreviated form of the *Shimmer3* id number (e.g. *IDbf5e*)), followed by a three digit number which is sequentially incremented at new logging session.

For example, the first time a *Shimmer3* with name "*device1*" in experiment "*experiment1*" logs data to the SD card, it will create the folder *data/experiment\_\*/device1-000/*. The next time it stops and subsequently restarts logging, it will create *data/experiment\_\*/device1-001/*, etc. The \* in the filename represents the time-code at which the Shimmer was last configured through *ConsensysBASIC* or *ConsensysPRO*.

- Within this subfolder, data is logged in files which are named sequentially with a three digit number indicating the order of logging, starting with *000*. The file is closed after 1 hour of continuous data logging and a new file is opened in the same subfolder, such that the second file is called *001* and so on. The last file is closed when logging stops and a new file (in a different folder) is created the next time logging begins.

### 7.2. Parsing the data

#### 7.2.1. Configuration header

Each RAW data file (e.g. *data/experiment1/device1-000/000*), begins with a header which contains the *Shimmer3* configuration information. The structure of this header is outlined in the tables below, where each row of each table represents one byte and the individual bits are separated, where appropriate, depending on whether each bit represents an independent binary value or the entire byte contains a single value. Empty cells and cells with constant values represent bits that are reserved for future use. For *Shimmer3*, there are a total 256 bytes in the header.

---

<sup>8</sup> <http://www.shimmersensing.com/support/wireless-sensor-networks-download/category/19>

### Byte # 0 – 9: Enabled Sensors

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	ADC Sample Rate (LSB)							
1	ADC Sample Rate (MSB)							
2								
3	Acc (LN) <sup>9</sup>	Gyr	Mag	EXG1_24BIT	EXG2_24BIT	GSR	ExtCh 7	ExtCh 6
4	br_amp		Battery	Acc (WR) <sup>10</sup>	ExtCh 15	IntCh 1	IntCh 12	IntCh13
5	IntCh 14	MPU9150_A CCEL	MPU9150_ MAG	EXG1_16BIT	EXG2_16BIT	Pressure	MPU9150_ TEMP	
6	MPL_QUAT_ 6DOF							
7	MPU9150_G YRO_CAL	MPU9150_A CCEL_CAL	MPU9150_ MAG_CAL					
8	LSM303 Digital Accel Rate				WR Accel Range		WR Accel LPM	WR Accel HRM
9	MPU9150 Gyro Rate							

### Byte # 10 – 29: Trial Configuration

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
10	LSM303 Mag Range			LSM303 Mag Rate			MPU9150 Gyro Range	
11	MPU_ACCEL_RANGE		Pressure Precision		GSR Range			EXP_PWR
12	MPU9150_D MP		MPU9150_LFP			MPU9150_MOT_CAL_CFG		
13	MPU9150_MPL_SAMPLING_RATE			MPU9150_MAG_SAMPLING_RATE				
14	MPL_sensor _fusion	MPL_gyro_c al_tc	MPL_vect_c omp_cal	MPL_mag_di st_cal	MPL_ENABL E			
15								
16	rtc_set_by _bt	0	User button	rtc_error	1	Sync	Master	
17	Single touch							
18	Broadcast interval							
19								
20	EST_EXP_LEN (MSB)							
21	EST_EXP_LEN (LSB)							
22	MAX_EXP_LEN (MSB)							
23	MAX_EXP_LEN (LSB)							
24	mac 1-2							
25	mac 3-4							
26	mac 5-6							
27	mac 7-8							
28	mac 9-10							
29	mac 11-12							

<sup>9</sup> There are multiple accelerometers on the Shimmer3 mainboard. The low noise (LN) analog accelerometer on the KXRBS-2042 chip is enabled via Byte 3, Bit 7 (Acc (LN)), whilst the wide range digital accelerometer on the LSM303DLHC chip is enabled via Byte 4, Bit 4 (Acc (WR)).

### Byte # 30 – 39: Firmware and Shimmer Parameters

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
30	ShimmerVersion (MSB)							
31	ShimmerVersion (LSB)							
32	MyTriald							
33	Nshimmer							
34	FW Version - Type (MSB)							
35	FW Version - Type (LSB)							
36	FW Version - Major (MSB)							
37	FW Version - Major (LSB)							
38	FW Version - Minor							
39	FW Version - Release							

*Shimmer Version* indicates the hardware version for which the code was compiled. *Shimmer1* is denoted by 0, *Shimmer2* by 1, *Shimmer2r* by 2 and *Shimmer3* by 3.

The *FW Version* bytes define the firmware version. The *Type* field will always be 2 for *SDLog* firmware images. *Major* and *Minor* versions are indicated by a two- and one-byte value, respectively (e.g. for *SDLog v1.2*, *Major* = 1 and *Minor* = 2). The *Release* field can be ignored by users; it will have a value of 0.

### Byte # 40: Derived Sensors

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
40				PPG2_1.14	PPG1_12.13	PPG1_12.13	Skin_Temp	Res_Amp

Used to enable derived sensors from the native Shimmer sensors *in Shimmer data acquisition software applications, specifically Consensys*.

### Byte # 41 – 43: Reserved for future use

### Byte # 44 – 51: Real Time Clock Difference

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
44	Real Time Clock Difference 0 (MSB)							
45	Real Time Clock Difference 1							
46	Real Time Clock Difference 2							
47	Real Time Clock Difference 3							
48	Real Time Clock Difference 4							
49	Real Time Clock Difference 5							
50	Real Time Clock Difference 6							
51	Real Time Clock Difference 7 (LSB)							

This value is the difference between the local PC time (using Unix epoch) and the time on the Shimmer internal clock. It is used primarily to enable real world timestamps in Shimmer data acquisition software applications, like *Consensys*. See Section 6.3 for more information.

### Byte # 52 - 55: Configuration Time

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
52	Config Time 0 (MSB)							
53	Config Time 1							

54	Config Time 2
55	Config Time 3 (LSB)

This value comes from the *Configuration ID* parameter from Section 6.1.5.

### Byte # 56-76: ExG Configuration

For *Shimmer3*, these parameters refer to the ExG module which is made up of two ADS1292R chips.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
56	NV_EXG_ADS1292R_1_CONFIG1							
57	NV_EXG_ADS1292R_1_CONFIG2							
58	NV_EXG_ADS1292R_1_LOFF							
59	NV_EXG_ADS1292R_1_CH1SET							
60	NV_EXG_ADS1292R_1_CH2SET							
61	NV_EXG_ADS1292R_1_RLD_SENS							
62	NV_EXG_ADS1292R_1_LOFF_SENS							
63	NV_EXG_ADS1292R_1_LOFF_STAT							
64	NV_EXG_ADS1292R_1_RESP1							
65	NV_EXG_ADS1292R_1_RESP2							
66	NV_EXG_ADS1292R_2_CONFIG1							
67	NV_EXG_ADS1292R_2_CONFIG2							
68	NV_EXG_ADS1292R_2_LOFF							
69	NV_EXG_ADS1292R_2_CH1SET							
70	NV_EXG_ADS1292R_2_CH2SET							
71	NV_EXG_ADS1292R_2_RLD_SENS							
72	NV_EXG_ADS1292R_2_LOFF_SENS							
73	NV_EXG_ADS1292R_2_LOFF_STAT							
74	NV_EXG_ADS1292R_2_RESP1							
75	NV_EXG_ADS1292R_2_RESP2							

### Byte # 76-181: Calibration parameters

Bytes 76 to 181 store the calibration data for individual *Shimmer3* sensors. These parameters are used to calibrate the RAW data during post-experimentation analysis. The layout and description of these bytes can be found in the Appendix section of this document (*i.e.* Section 9).

### ⚠ Byte # 182-213: Calibration Timestamps

Calibration Timestamps was a new feature introduced in v0.13.0.

Bytes 182 to 213 store the calibration timestamps for individual *Shimmer3* sensors. Such timestamps are used to calibrate the RAW data during post-experimentation analysis. The layout and description of these bytes can be found in the Appendix section of this document (*i.e.*, Section 9).

### ⚠ Byte # 214-216: Daughter Card ID bytes

Daughter Card ID bytes were a new feature introduced in v0.13.0.

Bytes 214 to 216 store the Daughter Card ID bytes. These three bytes are read from the daughter card, representing the version numbers of the card. If no expansion cards are appended to the *Shimmer3* unit, the values of these three bytes are 0xFF.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
214	Daughter_Card_ID_byte0							
215	Daughter_Card_ID_byte1							
216	Daughter_Card_ID_byte2							

**Byte # 217 - 250: Reserved for future use**

### 7.2.2. Initial Timestamp

Immediately after the configuration header, there is a 5-byte timestamp, whose value is the time on the local *Shimmer3* clock (in units of ticks of the 32 kHz clock) when the first sample in that file was recorded. It is used for aligning the data from multiple Shimmer devices to a common clock (see Section 8.1).

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
251	Initial time stamp (MSB) (1 <sup>st</sup> byte)							
252	Initial time stamp (LSB) (5 <sup>th</sup> byte)							
253	Initial time stamp (4 <sup>th</sup> byte)							
254	Initial time stamp (3 <sup>rd</sup> byte)							
255	Initial time stamp (2 <sup>nd</sup> byte)							

### 7.2.3. Data log

Following these 256 bytes, the logged sensor data begins (i.e. at the 257th byte in the file). The data is written in blocks of up to 512 bytes. These blocks are continuously written until data has been logged to a given file for one hour, at which time the current file is closed and a new file is opened, with the same format as the current one (i.e. the configuration header is repeated in every file).

**Note:** The block size in *SDLog* v0.3.0 (and earlier) is 1024 bytes as opposed to 512 bytes so any formula referencing 512 bytes should be replaced with 1024 if *SDLog* v.0.3.0 or earlier was used.

The exact number of bytes per block and the interpretation of those bytes of data depend on the enabled sensors and whether or not synchronisation is enabled. Introducing the variable,  $S$ , such that  $S = 1$  if master-slave synchronisation is enabled and 0 otherwise, the total number of bytes written per block,  $Bp$ , is given by:

$$Bp = N * (Bs + 3) + 9 * S$$

where,

$Bs = Nc3 * 3 + Nc2 * 2 + Nc1$ , is the number of bytes per sample, which depends on the number of 3byte channels ( $Nc3$ ), number of 2byte channels ( $Nc2$ ) and the number of 1-byte channels ( $Nc1$ ) required by the enabled sensors;

$N = \text{floor}((512 - 9 * S) / (Bs + 3))$ , is the integer number of samples per block, with a buffer of 512 bytes and the extra 3 bytes per sample are the lower three bytes of the 64-bit timestamp from the Shimmer's local 32 kHz clock.

There are two possible cases, as outlined in the examples below.



### **Case 1: Synchronisation is enabled.**

The first byte represents the sign (0 if positive, 1 if negative) of the time offset between the local slave timestamp and the broadcasted master timestamp. The next eight bytes represent the 64-bit magnitude of the offset, such that applying the sign byte results in:  $\text{offset} = \text{local timestamp} - \text{master timestamp}$ . The remainder of the bytes in the block are interleaved local timestamps (lower three bytes of the 32-bit timer) and the associated sensor samples. For example, with the accelerometer and the gyroscope enabled, there are three 2-byte channels for the accelerometer and three 2-byte channels for the gyroscope in each sample and the data are as follows:

Byte 0:	offset sign	
Byte 1:	offset magnitude byte 0 (LSB)	
Byte 2:	offset magnitude byte 1	
Byte 3:	offset magnitude byte 2	
Byte 4:	offset magnitude byte 3	
Byte 5:	offset magnitude byte 4	
Byte 6:	offset magnitude byte 5	
Byte 7:	offset magnitude byte 6	
Byte 8:	offset magnitude byte 7 (MSB)	
Byte 9:	timestamp 0 byte 0 (LSB)	
Byte 10:	timestamp 0 byte 1	
Byte 11:	timestamp 0 byte 2 (MSB)	
Byte 12:	sample 0 channel 0 byte 0 (MSB)	(XAccel)
Byte 13:	sample 0 channel 0 byte 1 (LSB)	(XAccel)
Byte 14:	sample 0 channel 1 byte 0 (MSB)	(YAccel)
Byte 15:	sample 0 channel 1 byte 1 (LSB)	(YAccel)
Byte 16:	sample 0 channel 2 byte 0 (MSB)	(ZAccel)
Byte 17:	sample 0 channel 2 byte 1 (LSB)	(ZAccel)
Byte 18:	sample 0 channel 3 byte 0 (MSB)	(XGyro)
Byte 19:	sample 0 channel 3 byte 1 (LSB)	(XGyro)
Byte 20:	sample 0 channel 4 byte 0 (MSB)	(YGyro)
Byte 21:	sample 0 channel 4 byte 1 (LSB)	(YGyro)
Byte 22:	sample 0 channel 5 byte 0 (MSB)	(ZGyro)
Byte 23:	sample 0 channel 5 byte 1 (LSB)	(ZGyro)
Byte 24:	timestamp 1 byte 0 (LSB)	
Byte 25:	timestamp 1 byte 1	
Byte 26:	timestamp 1 byte 2 (MSB)	
Byte 27:	sample 1 channel 0 byte 0 (MSB)	(XAccel)
Byte 28:	sample 1 channel 0 byte 1 (LSB)	(XAccel)

and so on, until a total of  $Bp = 504$  bytes is reached ( $Bs = 3*2 + 3*2 = 12$ ;  $N = \text{floor}((512 - 9)/(12 + 3)) = 33$ ).

The 9 bytes of the offset (sign and magnitude) are written to the SD card at the start of every write operation. There will be many write operations for each time offset update; for example, a write operation might occur many times every second, whilst the value for the synchronisation



communication interval can be up to one hour. At any given write operation, if a new offset has not been calculated since the previous write operation, then the value 0xFFFFFFFFFFFFFFFF is written for the offset magnitude, with a sign byte of 0. This value must be recognised in software to denote an “invalid” value and be removed before processing the offsets.

**!** In *SDLog v0.11.0* or earlier the Shimmer timestamp channel was only two bytes (16-bit), the offset magnitude was only 4 bytes.

### **Case 2: Synchronisation is not enabled.**

All of the bytes in the block are interleaved local timestamps (lower three bytes of the 64-bit timer) and the associated sensor samples. For example, with the accelerometer and the gyroscope enabled, there are three 2-byte channels for the accelerometer and three 2-byte channels for the gyroscope in each sample and the data are as follows:

Byte 0:	timestamp 0 byte 0 (LSB)	
Byte 1:	timestamp 0 byte 1	
Byte 2:	timestamp 0 byte 2 (MSB)	
Byte 3:	sample 0 channel 0 byte 0 (MSB)	(XAccel)
Byte 4:	sample 0 channel 0 byte 1 (LSB)	(XAccel)
Byte 5:	sample 0 channel 1 byte 0 (MSB)	(YAccel)
Byte 6:	sample 0 channel 1 byte 1 (LSB)	(YAccel)
Byte 7:	sample 0 channel 2 byte 0 (MSB)	(ZAccel)
Byte 8:	sample 0 channel 2 byte 1 (LSB)	(ZAccel)
Byte 9:	sample 0 channel 3 byte 0 (MSB)	(XGyro)
Byte 10:	sample 0 channel 3 byte 1 (LSB)	(XGyro)
Byte 11:	sample 0 channel 4 byte 0 (MSB)	(YGyro)
Byte 12:	sample 0 channel 4 byte 1 (LSB)	(YGyro)
Byte 13:	sample 0 channel 5 byte 0 (MSB)	(ZGyro)
Byte 14:	sample 0 channel 5 byte 1 (LSB)	(ZGyro)
Byte 15:	timestamp 1 byte 0 (LSB)	
Byte 16:	timestamp 1 byte 1	
Byte 17:	timestamp 1 byte 2 (MSB)	
Byte 18:	sample 1 channel 0 byte 0 (MSB)	(XAccel)
Byte 19:	sample 1 channel 0 byte 1 (LSB)	(XAccel)

and so on, until a total of  $B_p = 510$  bytes is reached ( $B_s = 12$ ;  $N = \text{floor}((512)/(12 + 3)) = 34$ ).

**!** In *SDLog v0.11.0* or earlier the Shimmer timestamp channel was only two bytes (16-bit)

### **Order of data channels**

The order in which the sensors appear in the data channels depends on which sensors are enabled. The firmware will assign the channels in the order outlined in Table 7-1 (top to bottom). Note that “Accel (LN)” refers to the low noise accelerometer, whilst “Accel (WR)” refers to the wide range accelerometer. Each row contains the sensor name, the number of channels per sensor, and the

number of bytes per channel. The datatype and endianness of the bytes for each channel are also specified.

Channel type	Channel contents		Number of channels	Bytes per channel	Endian		Datatype
Analog channels	Accel (LN)		3	2	little		u12
	Battery		1	2	little		u12
	Ext Exp A7		1	2	little		u12
	Ext Exp A6		1	2	little		u12
	Ext Exp A15		1	2	little		u12
	Int Exp A12		1	2	little		u12
	Int Exp A13		1	2	little		u12
	Int Exp A14		1	2	little		u12
	Bridge Amplifer High		1	2	little		u12
	Bridge Amplifer Low		1	2	little		u12
	Int Exp A1		1	2	little		u12
	GSR		1	2	little		u12
Digital channels	Gyro_mpu*		3	2	big		i16
	Accel_lsm (WR)		3	2	little		i16
	Mag_lsm		3	2	LSM303DLHC	LSM303AHTR	i16
					big	little	
	Accel_mpu*		3	2	big		i16
	Mag_mpu*		3	2	little		i16
	Temperature_bmpX80 where X is 1 or 2 depending on board version		1	2	big		u16
	Pressure_bmpX80 where X is 1 or 2 depending on board version		1	3	big		u24
	EXG1(24-bit/16-bit)		1	7/5	big		1 x u8 2 x i24/i16
	EXG2(24-bit/16-bit)		1	7/5	big		1 x u8 2 x i24/i16
	DMP / MPL Channels	Quat_6DOF	4	4	big		i32**
		Quat_9DOF	4	4	Big		i32**
		Euler_6DOF	3	4	Big		i32*
		Euler_9DOF	3	4	Big		i32*
		MPL_heading	1	4	Big		i32*
		MPU_temp	1	4	Big		i32*
		MPL_pedom_cnt	1	4	Big		u8
		MPL_pedom_time	1	4	Big		u8
		TapDirAndTapCnt	1	1	Big		u8
		MotionAndOrient	1	1	Big		u8
		MPL_gyro_cal	3	4	Big		i32*

		MPL_accel_cal	3	4	Big	i32*
		MPL_mag_cal	3	4	Big	i32*
		Quat_6DOF_raw	4	4	Big	i32*

*Table 7-1: Order of data channels for Shimmer3. Note that the channels shaded in grey are reserved but are not supported in the current SDLog firmware release. Note: \* denotes data scaled by  $2^{16}$  while \*\* denotes data scaled by  $2^{30}$ .*


## 8. Synchronising the data

### 8.1. Aligning data to a common clock

If synchronisation has been enabled, the clock of the master Shimmer can be taken as a reference clock and data from all slave Shimmers can be aligned with this reference clock by manipulating the timestamps associated with the slave data. It is important to note that this alignment is **not** done by the firmware and must be done by post-processing in software, i.e. after data logging has ended.

Shimmer offers software applications to automate the process of parsing and aligning the synchronised data; it is highly recommended that these applications are used, in order to avoid errors in aligning the logged data to a common clock. The *Consensus* application should be used along with *Consensus Base* hardware or multiple *Shimmer Dock* units. Customers who have purchased a license for the (deprecated<sup>10</sup>) *Multi Shimmer Sync for SD* may continue to use this software; however, it is recommended to update to *Consensus* in order to benefit from the most up-to-date feature set and future updates. Check our [website](#) for more information.

Alternatively, the user may choose to write their own software, following the procedure outlined below.

1. Calculate the value of the slave timestamps:
  - a. Convert the logged 24-bit (16-bit in SDLog v0.11.0 or earlier) raw timestamps to a continuous value (i.e. one that does not return to zero when it reaches 16777216 or 65536 in SDLog v0.11.0 or earlier).
-  In SDLog v0.11.0 or earlier the Shimmer timestamp channel was only two bytes (16-bit)
- b. Note that these timestamps are relative to when logging started and must later be converted to a value relative to when the Shimmer clock was reset to zero (see Step 6).
  - c. Note that these timestamps are in units of ticks of the 32,768 Hz clock.
2. Remove invalid offsets<sup>11</sup> from the data, leaving only valid offsets, as shown in Figure 8-1:
  - a. Invalid offsets will have a value of  $2^{64}-1$  in decimal (0xFFFFFFFFFFFFFFFF in hexadecimal format) in the 8-byte *offset magnitude*.
3. Find the time on the slave clock at which the valid offset messages were received:
  - a. Use the first timestamp of the block containing the valid offset.
  - b. This step will generate a set of {*message time*, *offset*} pairs, as shown in Figure 8-1.
4. Apply the *offset sign* to the *offset magnitude*:

$$i. \text{ Offset} = (1 - 2 * \text{offset sign}) * \text{offset magnitude}$$

<sup>10</sup> MSS for SD is due to be deprecated in April 2015.

<sup>11</sup> See Section 7.2.3 for details on "invalid" offsets.

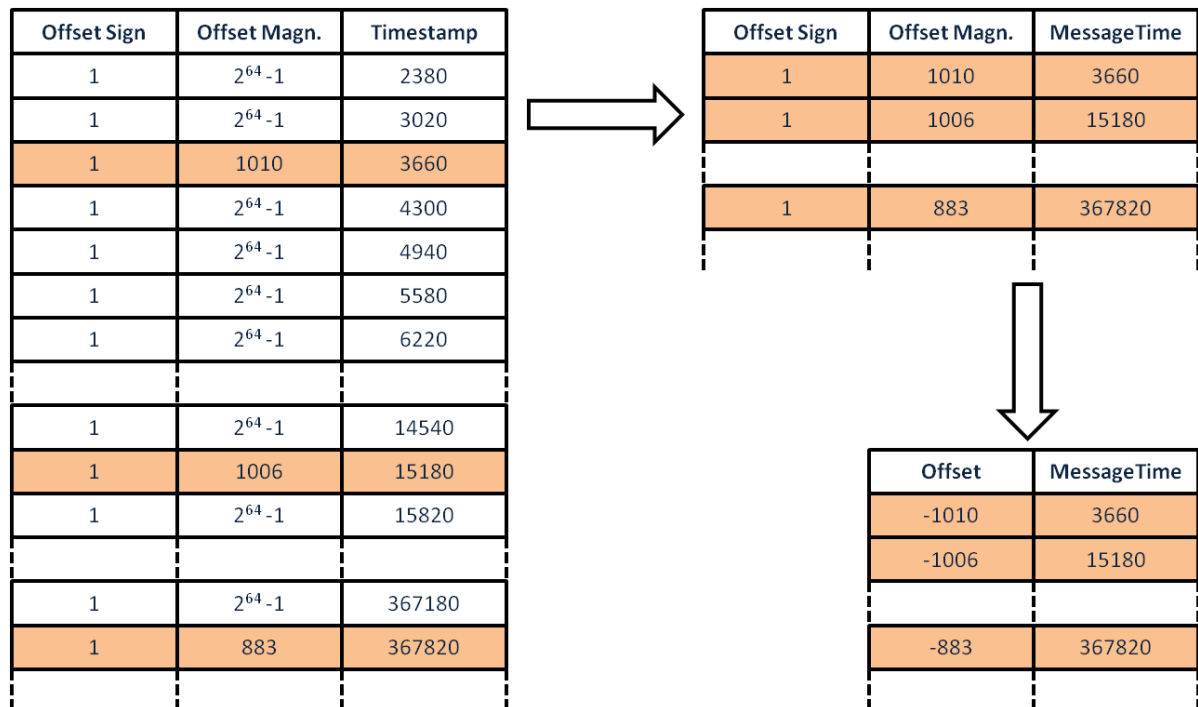


Figure 8-1: Steps 1 - 4

5. Interpolate the {message time, offset} pairs using the slave timestamps as shown in Figure 8-3:

a. Use linear interpolation.

i. A minimum mean square error fit or similar will provide reasonable results. However, the ideal fit would produce a lower bound on the estimated offset values, as illustrated in Figure 8-2.

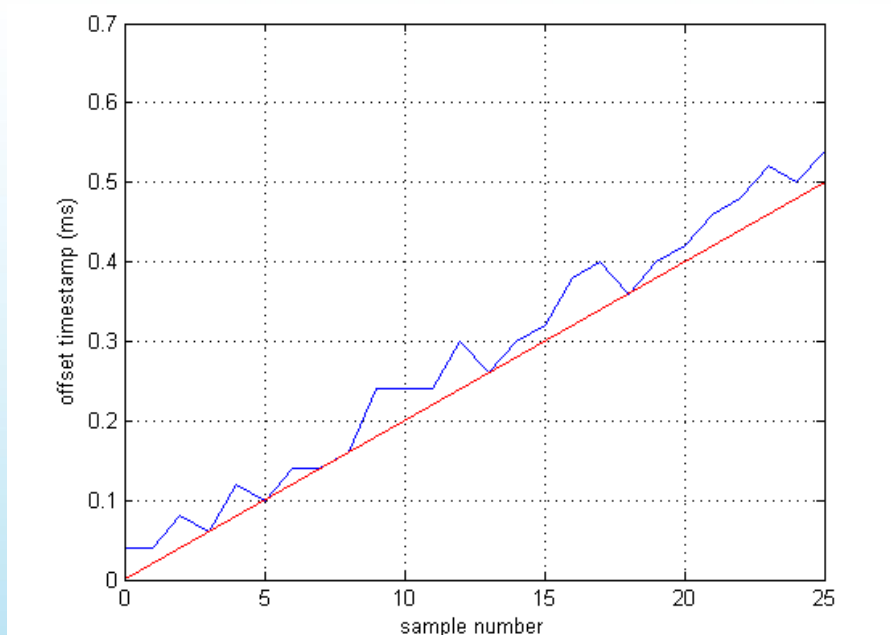


Figure 8-2: Step 5a

- b. This step will generate an offset estimate for every sample, as shown in Figure 8-3.

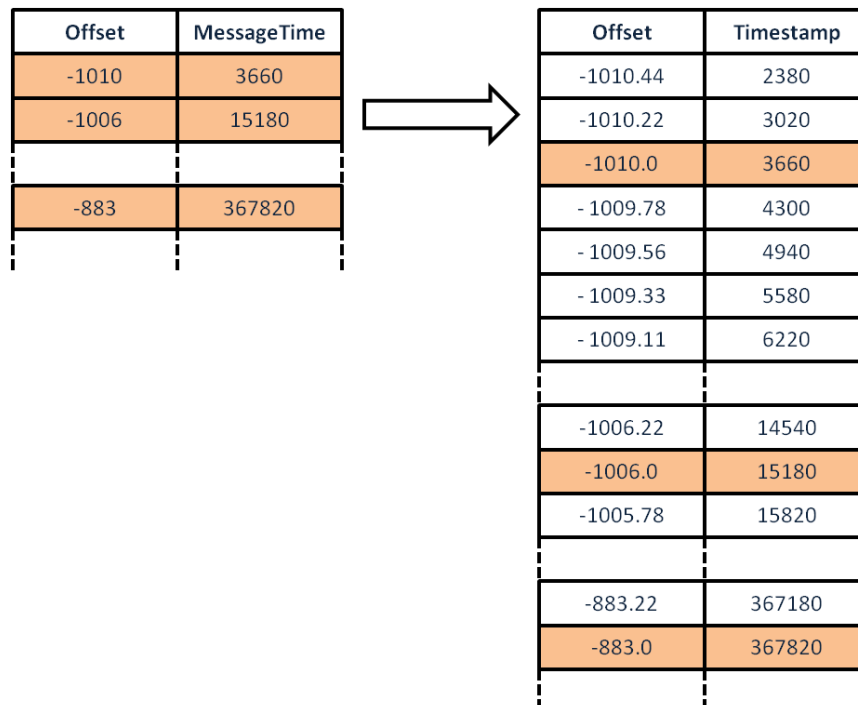



Figure 8-3: Step 5b

6. Convert the timestamps to the 32-bit slave clock value, relative to when the slave was rebooted (and its local clock reset to zero):

- a. The initial timestamp (bytes 251 - 255 of the data file) is the time on the slave clock in 40-bit resolution when the first sample of that file was recorded.

 In *SDLog v0.12.0* or earlier the initial timestamp (bytes 252 - 255 of the data file) had 32-bit resolution.

- b. The timestamp for the first logged data sample (sample 0) will be the 24-bit (or 16-bit equivalent in *SDLog v0.11.0* or earlier) equivalent of this 32-bit value and can be used to calculate the constant value that should be added to all subsequent samples.

 In *SDLog v0.11.0* or earlier the Shimmer timestamp channel was only two bytes (16-bit)

7. Convert the timestamps to the 'real time clock' value, if this value was set for the data logging session:

- a. The RTC offset (bytes 252 - 255 of the data file) is the offset between the Shimmer device's local clock in 32-bit resolution and the 'real world' clock, sent to the device before logging started. The value will be zero if no 'real world' time was sent to the device or if the device was powered off or reset before logging started.

- b. The constant RTC offset should be added to all slave timestamps.
8. Subtract the offset estimates from the slave timestamps to give the estimated slave timestamps relative to the master clock:
- a. *Aligned timestamp = Slave timestamp - Offset.*

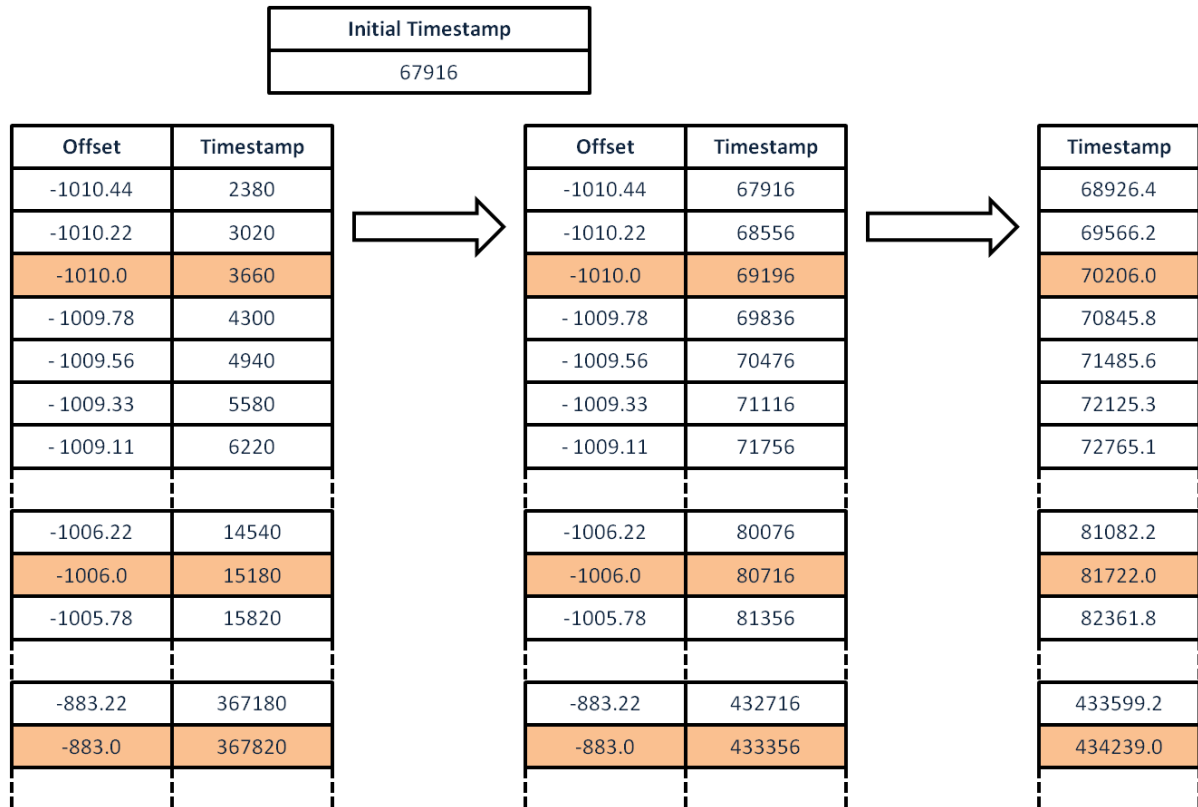


Figure 8-4: Steps 6 – 8

## 9. Appendices

### 9.1. Known bugs

#### 9.1.1. Invensense MPL

There is a known bug in *SDLog Firmware* associated with the Invensense MPL sensors which means users should avoid using the MPL Accel, MPL Gyro and MPL Mag in SDLog Firmware v0.10.0 (or earlier).

**Note:** SDLog\_v0.19.0 does not use the Invensense MPL library and so this bug does not apply.

#### 9.1.2. ExG signal issue

An issue was found in SDLog (also spanning previous firmware versions) whereby a Shimmer configured to record ExG data (test signal used) would present aperiodic spikes in the output data if the Shimmer was not docked before logging began. After debugging and cross-comparison with LogAndStream (where the issue is not present), no immediate resolution could be made.

### 9.2. InfoMem contents

⚠ The information in this section applies to *SDLog Firmware v0.9.0* (or later) only.

The configuration and calibration parameter values are stored by the *Shimmer3* in the InfoMem, which is the part of the *Shimmer3* memory that survives a reset or power cycle but is overwritten when the *Shimmer3* is reprogrammed. The format of the configuration data stored in InfoMem is as follows:

InfoMem Byte	Contents
0 - 1	Sampling rate
2	Buffer size
3 - 5	Selected sensors
6 - 9	Config bytes (Allows for 56 individual boolean settings)
10 - 29	ExG configuration bytes
30	Bluetooth Communication baud rate
31 - 33	Derived Channels
34 - 54	Low Noise Accelerometer calibration values
55 - 75	Gyroscope calibration values
76 - 96	Magnetometer calibration values
97 - 117	Wide Range Accelerometer calibration values
118 - 127	Reserved for future use
128 - 129	MPL sensors
130 - 132	MPL config bytes
133 - 153	MPL Accelerometer calibration values
154 - 174	MPL Magnetometer calibration values
175 - 186	MPL Gyroscope calibration values
187 - 229	SD Logging configuration parameters
230	InfoMem contents changed flags
256 - 381	Slave Node IDs



*Table 9-1 InfoMem layout overview.*

## Selected Sensors - InfoMem Bytes 3 to 5

The *Selected sensors* bytes have a single bit assigned to each sensor as follows:

	Bit	Property
InfoMem Byte 3	7	Low Noise Accelerometer.
	6	Gyroscope.
	5	Magnetometer.
	4	ExG1_24BIT.
	3	ExG2_24BIT.
	2	GSR.
	1	External Expansion ADC Channel 7.
	0	External Expansion ADC Channel 6.
InfoMem Byte 4	7	Bridge Amplifer.
	6	Not yet assigned.
	5	Battery Monitor.
	4	Wide Range Accelerometer.
	3	External Expansion ADC Channel 15.
	2	Internal Expansion ADC Channel 1.
	1	Internal Expansion ADC Channel 12.
	0	Internal Expansion ADC Channel 13.
InfoMem Byte 5	7	Internal Expansion ADC Channel 14.
	6	MPU9X50 Accelerometer where X is 1 or 2 depending on board version
	5	MPU9X50 Magnetometer where X is 1 or 2 depending on board version
	4	ExG1_16BIT.
	3	ExG2_16BIT.
	2	BMPX80 Pressure where X is 1 or 2 depending on board version
	1	BMPX80 Temperature where X is 1 or 2 depending on board version
	0	MSP430 Temperature.

Table 9-2 Selected Sensor Bytes

## Sensor Config Bytes - InfoMem Bytes 6 to 9

The *Sensor Config* bytes contain the following parameters:

InfoMem Byte 6 - Config Setup Byte 0	
Bits 7 – 4	Wide Range (LSM303DLHC or LSM303AHTR) Accelerometer Data Rate.
Bits 3 – 2	Wide Range (LSM303DLHC or LSM303AHTR) Accelerometer Range.
Bit 1	Wide Range (LSM303DLHC or LSM303AHTR) Accelerometer Low Power Mode.
Bit 0	Wide Range (LSM303DLHC or LSM303AHTR) Accelerometer High Resolution Mode.
InfoMem Byte 7 - Config Setup Byte 1	
Bits 7 – 0	MPU9X50 Data Rate where X is 1 or 2 depending on board version
InfoMem Byte 8 - Config Setup Byte 2	
Bits 7 – 5	(LSM303DLHC or LSM303AHTR) Magnetometer Range.
Bits 4 – 2	(LSM303DLHC or LSM303AHTR) Magnetometer Data Rate.
Bit 1 - 0	MPU9X50 Gyroscope Range where X is 1 or 2 depending on board version

InfoMem Byte 9 - Config Setup Byte 3	
Bits 7 – 6	MPU9X50 Accelerometer Range where X is 1 or 2 depending on board version
Bits 5 – 4	BMPX80 Pressure Resolution where X is 1 or 2 depending on board version
Bit 3 - 1	GSR Range
Bit 0	Internal Expansion Power Enable

Table 9-3 Sensor Config bytes

### ExG Configuration Bytes - InfoMem Bytes 10 to 29

These bytes store the configuration bytes which are sent to the *ECG/EMG Expansion Board* if one is connected and enabled. For detailed information on these bytes please refer to either the *ECG User Guide* or the *EMG User Guide* - both of which are available for download from the members section of the Shimmer website.

### BT Communication Baud Rate - InfoMem Byte 30

This byte stores the baud rate at which the Shimmer's microcontroller communicates with the on-board Bluetooth module and consequently, back to a base device. There are 11 allowable options, as listed in Table 9-4:

Value (decimal)	Baud
0	115200
1	1200
2	2400
3	4800
4	9600
5	19200
6	38400
7	57600
8	230400
9	460800 (default)
10	921600

Table 9-4 BT Communication Baud Rate byte options

### Derived Channels - InfoMem Bytes 31 to 34

These bytes contain flags to indicate the type of peripheral that is attached to the analog channels. These bytes have no explicit function in firmware and are included to allow software applications, specifically *Consensus*, to correctly label the data. In custom applications, they may be used as the developer sees fit.

### Calibration Parameters - InfoMem Bytes 34 to 117

The calibration parameters for the inertial measurement units (accelerometer, gyroscope and magnetometer) consist of a three-element offset bias vector, a three-element sensitivity vector and

a 3x3-element alignment matrix<sup>12</sup>. The structure of these values when they are sent to/from the *Shimmer3* and stored in InfoMem is as follows:

- Each of the 3 offset bias vector values are stored as 16-bit signed integers (big endian) and are contained in bytes 0-5.
- Each of the 3 sensitivity vector values are stored as 16-bit signed integers (big endian) and are contained in bytes 6-11.
- Each of the 9 alignment matrix values are stored as 8-bit signed integers and are contained in bytes 12-20.

### ***MPL Parameters - InfoMem Bytes 118 to 186***

These bytes contain the configuration and calibration parameters for the Invensense MPL features which have been described throughout this document.

- The contents of bytes 128 - 129 replicate the contents of bytes 6 - 7 of the Configuration header described in Section 7.2.1.
- The contents of bytes 130 - 132 replicate the contents of bytes 12 - 14 of the Configuration header described in Section 7.2.1.
- The contents of bytes 133 - 186 replicate the contents of bytes 182 - 235 of the Configuration header described in Section 9.3.

### ***SD Logging - Experiment parameters - InfoMem Bytes 187 to 229***

<b>InfoMem Byte 187 - 198</b>	Shimmer name
<b>InfoMem Byte 199 - 210</b>	Experiment ID name
<b>InfoMem Byte 211 - 214</b>	Configuration time
<b>InfoMem Byte 215</b>	Trial ID
<b>InfoMem Byte 216</b>	Reserved
<b>InfoMem Byte 217 - 218</b>	Reserved
<b>InfoMem Byte 219</b>	Reserved
<b>InfoMem Byte 220 - 221</b>	Estimated Trial Length
<b>InfoMem Byte 222 - 223</b>	Maximum Trial Length
<b>InfoMem Byte 224 - 229</b>	Multi-Shimmer Sync - Master Shimmer MAC address

*Table 9-5 Experiment Settings*

### ***InfoMem contents changed flags - InfoMem Byte 230***

This byte contains the 'InfoMem changed' (bit 0) and 'calibration changed' (bit 1) flags, which indicate to the firmware that the InfoMem contents have been updated and that the new parameters should be written to the configuration file and/or the calibration files on the SD card. See Sections 6.1.1 and 6.2.1 for more details.

<sup>12</sup> For a more detailed description of IMU calibration parameters, refer to the *Shimmer 9DoF Calibration User Manual* and the *Shimmer IMU User Guide*.

### ***Slave Node IDs - InfoMem Bytes 256 - 381***

These bytes contain the mac addresses of the slave nodes, with six (6) bytes used per node.

### 9.3. SD data file configuration header - sensor calibration bytes

#### **Byte # 76-96: Digital Accelerometer Calibration**

For *Shimmer3*, these parameters refer to the LSM303DLHC accelerometer. There are other accelerometers also populated on the *Shimmer3* circuit board; the associated calibration parameters for the other devices are stored in a later section of the configuration header.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
76	Digital Accel Calibration Offset X (MSB)							
77	Digital Accel Calibration Offset X (LSB)							
78	Digital Accel Calibration Offset Y (MSB)							
79	Digital Accel Calibration Offset Y (LSB)							
80	Digital Accel Calibration Offset Z (MSB)							
81	Digital Accel Calibration Offset Z (LSB)							
82	Digital Accel Calibration Gain X (MSB)							
83	Digital Accel Calibration Gain X (LSB)							
84	Digital Accel Calibration Gain Y (MSB)							
85	Digital Accel Calibration Gain Y (LSB)							
86	Digital Accel Calibration Gain Z (MSB)							
87	Digital Accel Calibration Gain Z (LSB)							
88	Digital Accel Calibration Align XX							
89	Digital Accel Calibration Align XY							
90	Digital Accel Calibration Align XZ							
91	Digital Accel Calibration Align YX							
92	Digital Accel Calibration Align YY							
93	Digital Accel Calibration Align YZ							
94	Digital Accel Calibration Align ZX							
95	Digital Accel Calibration Align ZY							
96	Digital Accel Calibration Align ZZ							

#### **Byte # 97 - 117: Gyroscope Calibration**

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
97	Gyro Calibration Offset X (MSB)							
98	Gyro Calibration Offset X (LSB)							
99	Gyro Calibration Offset Y (MSB)							
100	Gyro Calibration Offset Y (LSB)							
101	Gyro Calibration Offset Z (MSB)							
102	Gyro Calibration Offset Z (LSB)							
103	Gyro Calibration Gain X (MSB)							
104	Gyro Calibration Gain X (LSB)							
105	Gyro Calibration Gain Y (MSB)							
106	Gyro Calibration Gain Y (LSB)							
107	Gyro Calibration Gain Z (MSB)							
108	Gyro Calibration Gain Z (LSB)							
109	Gyro Calibration Align XX							
110	Gyro Calibration Align XY							

111	Gyro Calibration Align XZ
112	Gyro Calibration Align YX
113	Gyro Calibration Align YY
114	Gyro Calibration Align YZ
115	Gyro Calibration Align ZX
116	Gyro Calibration Align ZY
117	Gyro Calibration Align ZZ

**Byte # 118 - 138: Magnetometer calibration**

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
118	Mag Calibration Offset X (MSB)							
119	Mag Calibration Offset X (LSB)							
120	Mag Calibration Offset Y (MSB)							
121	Mag Calibration Offset Y (LSB)							
122	Mag Calibration Offset Z (MSB)							
123	Mag Calibration Offset Z (LSB)							
124	Mag Calibration Gain X (MSB)							
125	Mag Calibration Gain X (LSB)							
126	Mag Calibration Gain Y (MSB)							
127	Mag Calibration Gain Y (LSB)							
128	Mag Calibration Gain Z (MSB)							
129	Mag Calibration Gain Z (LSB)							
130	Mag Calibration Align XX							
131	Mag Calibration Align XY							
132	Mag Calibration Align XZ							
133	Mag Calibration Align YX							
134	Mag Calibration Align YY							
135	Mag Calibration Align YZ							
136	Mag Calibration Align ZX							
137	Mag Calibration Align ZY							
138	Mag Calibration Align ZZ							

**Byte # 139 - 159: Analog Accelerometer Calibration**

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
139	Analog Accel Calibration Offset X (MSB)							
140	Analog Accel Calibration Offset X (LSB)							
141	Analog Accel Calibration Offset Y (MSB)							
142	Analog Accel Calibration Offset Y (LSB)							
143	Analog Accel Calibration Offset Z (MSB)							
144	Analog Accel Calibration Offset Z (LSB)							
145	Analog Accel Calibration Gain X (MSB)							
146	Analog Accel Calibration Gain X (LSB)							
147	Analog Accel Calibration Gain Y (MSB)							
148	Analog Accel Calibration Gain Y (LSB)							
149	Analog Accel Calibration Gain Z (MSB)							
150	Analog Accel Calibration Gain Z (LSB)							



<b>151</b>	Analog Accel Calibration Align XX
<b>152</b>	Analog Accel Calibration Align XY
<b>153</b>	Analog Accel Calibration Align XZ
<b>154</b>	Analog Accel Calibration Align YX
<b>155</b>	Analog Accel Calibration Align YY
<b>156</b>	Analog Accel Calibration Align YZ
<b>157</b>	Analog Accel Calibration Align ZX
<b>158</b>	Analog Accel Calibration Align ZY
<b>159</b>	Analog Accel Calibration Align ZZ

**Byte # 160 - 181: Temperature (BMP180) and Pressure Calibration**

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>160</b>	Temp & Pres Calibration AC1_MSB							
<b>161</b>	Temp & Pres Calibration AC1_LSB							
<b>162</b>	Temp & Pres Calibration AC2_MSB							
<b>163</b>	Temp & Pres Calibration AC2_LSB							
<b>164</b>	Temp & Pres Calibration AC3_MSB							
<b>165</b>	Temp & Pres Calibration AC3_LSB							
<b>166</b>	Temp & Pres Calibration AC4_MSB							
<b>167</b>	Temp & Pres Calibration AC4_LSB							
<b>168</b>	Temp & Pres Calibration AC5_MSB							
<b>169</b>	Temp & Pres Calibration AC5_LSB							
<b>170</b>	Temp & Pres Calibration AC6_MSB							
<b>171</b>	Temp & Pres Calibration AC6_LSB							
<b>172</b>	Temp & Pres Calibration B1_MSB							
<b>173</b>	Temp & Pres Calibration B1_LSB							
<b>174</b>	Temp & Pres Calibration B2_MSB							
<b>175</b>	Temp & Pres Calibration B2_LSB							
<b>176</b>	Temp & Pres Calibration MB_MSB							
<b>177</b>	Temp & Pres Calibration MB_LSB							
<b>178</b>	Temp & Pres Calibration MC_MSB							
<b>179</b>	Temp & Pres Calibration MC_LSB							
<b>180</b>	Temp & Pres Calibration MD_MSB							
<b>181</b>	Temp & Pres Calibration MD_LSB							

### Byte # 160 – 181 and Byte # 222-223: Temperature (BMP280) and Pressure Calibration

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
160	Temp & Pres Calibration DIG_T1_MSB							
161	Temp & Pres Calibration DIG_T1_LSB							
162	Temp & Pres Calibration DIG_T2_MSB							
163	Temp & Pres Calibration DIG_T2_LSB							
164	Temp & Pres Calibration DIG_T3_MSB							
165	Temp & Pres Calibration DIG_T3_LSB							
166	Temp & Pres Calibration DIG_P1_MSB							
167	Temp & Pres Calibration DIG_P1_LSB							
168	Temp & Pres Calibration DIG_P2_MSB							
169	Temp & Pres Calibration DIG_P2_LSB							
170	Temp & Pres Calibration DIG_P3_MSB							
171	Temp & Pres Calibration DIG_P3_LSB							
172	Temp & Pres Calibration DIG_P4_MSB							
173	Temp & Pres Calibration DIG_P4_LSB							
174	Temp & Pres Calibration DIG_P5_MSB							
175	Temp & Pres Calibration DIG_P5_LSB							
176	Temp & Pres Calibration DIG_P6_MSB							
177	Temp & Pres Calibration DIG_P6_LSB							
178	Temp & Pres Calibration DIG_P7_MSB							
179	Temp & Pres Calibration DIG_P7_LSB							
180	Temp & Pres Calibration DIG_P8_MSB							
181	Temp & Pres Calibration DIG_P8_LSB							
222	Temp & Pres Calibration DIG_P9_MSB							
223	Temp & Pres Calibration DIG_P9_LSB							

### ! Byte # 182 - 213: Calibration Timestamps

Calibration Timestamps was a new feature introduced in v0.13.0. *DMP™* and *InvenSense MPL* calibration parameters are removed.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
182	Digital Accel Calibration TimeStamp (LSB) byte0							
183	byte1							
184	byte2							
185	byte3							
186	byte4							
187	byte5							
188	byte6							
189	Digital Accel Calibration TimeStamp (MSB) byte7							
190	Digital Gyro Calibration TimeStamp (LSB) byte0							
191	byte1							
192	byte2							
193	byte3							

194	byte4
195	byte5
196	byte6
197	Digital Gyro Calibration TimeStamp (MSB) byte7
198	Digital Mag Calibration TimeStamp (LSB) byte0
199	byte1
200	byte2
201	byte3
202	byte4
203	byte5
204	byte6
205	Digital Mag Calibration TimeStamp (MSB) byte7
206	Analogue Accel Calibration TimeStamp (LSB) byte0
207	byte1
208	byte2
209	byte3
210	byte4
211	byte5
212	byte6
213	Analogue Accel Calibration TimeStamp (MSB) byte7

## 9.4. Shimmer3 sensor conflicts

The *Shimmer3* sensors conflicts are shown below in Table 9-. Note that all Accelerometer, Magnetometer, Gyroscope, Pressure and Vbattery sensor channels are compatible with any other sensors.

	Int A14	Int A1	Int A12	Int A13	GSR	ExG1 24 Bit	ExG2 24 Bit	ExG1 16 Bit	ExG2 16 Bit	Br. Amp.
Int A14	-	OK	OK	OK	✗	✗	✗	✗	✗	✗
Int A1	OK	-	OK	OK	✗	✗	✗	✗	✗	OK
Int A12	OK	OK	-	OK	OK	✗	✗	✗	✗	✗
Int A13	OK	OK	OK	-	OK	✗	✗	✗	✗	✗
GSR	✗	✗	OK	OK	-	✗	✗	✗	✗	✗
ExG1 24 Bit	✗	✗	✗	✗	✗	-	OK	✗	OK <sup>13</sup>	✗
ExG2 24 Bit	✗	✗	✗	✗	✗	OK	-	OK <sup>14</sup>	✗	✗
ExG1 16 Bit	✗	✗	✗	✗	✗	✗	OK <sup>14</sup>	-	OK	✗
ExG2 16 Bit	✗	✗	✗	✗	✗	OK <sup>14</sup>	✗	OK	-	✗
Br. Amp.	✗	OK	✗	✗	✗	✗	✗	✗	✗	-

Table 9-6: Shimmer3 sensor conflicts.

<sup>13</sup> Combining one ExG 24-bit channel with one ExG 16-bit channel will not cause any conflict in *SDLog* firmware but is not supported in any Shimmer software applications.

## 9.5. Example sdlog.cfg file for Shimmer3

```
accel=1
gyro=1
mag=1
exg1_24bit=0
exg2_24bit=0
gsr=0
extch7=0
extch6=0
br_amp=0
vbat=1
accel_d=0
extch15=0
intch1=0
intch12=0
intch13=0
intch14=0
accel_mpu=0
mag_mpu=0
exg1_16bit=0
exg2_16bit=0
pres_bmp180=0
sample_rate=51.20
mg_internal_rate=6
mg_range=1
acc_internal_rate=0
accel_mpu_range=0
pres_bmp180_prec=0
gsr_range=4
exp_power=0
gyro_range=1
gyro_samplingrate=155
acc_range=0
acc_lpm=0
acc_hrm=1
rtc_error_enable=1
user_button_enable=1
iammaster=0
sync=0
low_battery_autostop=0
singletouch=0
tcxo=0
interval=0
est_exp_len=200
max_exp_len=12
myid=1
Nshimmer=1
shimmername=Shimmer_D336
experimentid=DefaultTrial
configtime=1495705715
baud_rate=9
derived_channels=0
EXG_ADS1292R_1_CONFIG1=0
EXG_ADS1292R_1_CONFIG2=128
EXG_ADS1292R_1_LOFF=16
EXG_ADS1292R_1_CH1SET=0
EXG_ADS1292R_1_CH2SET=0
EXG_ADS1292R_1_RLD_SENS=0
EXG_ADS1292R_1_LOFF_SENS=0
EXG_ADS1292R_1_LOFF_STAT=0
EXG_ADS1292R_1_RESP1=2
EXG_ADS1292R_1_RESP2=1
EXG_ADS1292R_2_CONFIG1=0
EXG_ADS1292R_2_CONFIG2=128
EXG_ADS1292R_2_LOFF=16
EXG_ADS1292R_2_CH1SET=0
EXG_ADS1292R_2_CH2SET=0
```

```
EXG_ADS1292R_2_RLD_SENS=0
EXG_ADS1292R_2_LOFF_SENS=0
EXG_ADS1292R_2_LOFF_STAT=0
EXG_ADS1292R_2_RESP1=2
EXG_ADS1292R_2_RESP2=1
DMP=0
MPL_use_LSM_mag=0
MPL_LPF=0
MPL_mot_cal=0
MPL_rate=0
MPU_mag_rate=0
MPL_mag_mix=0
MPL_QUAT_6DOF=0
MPL_QUAT_9DOF=0
MPL_Euler_6DOF=0
MPL_Euler_9DOF=0
MPL_heading=0
MPU_temp=0
MPL_pedometer=0
MPL_tap=0
MPL_motion_orient=0
MPL_gyro_cal=0
MPL_accel_cal=0
MPL_mag_cal=0
MPL_QUAT_6DOF_RAW=0
MPL_sensor_fusion=0
MPL_gyro_cal_tc=0
MPL_vect_comp_cal=0
MPL_mag_dist_cal=0
MPL=0
```

## 9.6. MPL calibration file contents

### 9.6.1. Example MPUcalib.ini

```
[MPL Accel 2g]
b0 = 5439488.000000
b1 = 10027008.000000
b2 = -44269568.000000
k0 = 16384
k1 = 16384
k2 = 16384
r00 = 0.000000
r01 = -1.000000
r02 = 0.000000
r10 = -1.000000
r11 = 0.000000
r12 = 0.000000
r20 = 0.000000
r21 = 0.000000
r22 = -1.000000
```

```
[MPL Gyro 2000dps]
b0 = -8488222.000000
b1 = -432537.000000
b2 = -1599078.000000
k0 = 16.400000
k1 = 16.400000
k2 = 16.400000
r00 = 0.000000
r01 = -1.000000
r02 = 0.000000
r10 = -1.000000
r11 = 0.000000
r12 = 0.000000
r20 = 0.000000
r21 = 0.000000
r22 = -1.000000
```

```
[MPL Mag 1.2Ga]
b0 = 16147553.000000
b1 = 6289477.000000
b2 = 105146528.000000
k0 = 341.000000
k1 = 341.000000
k2 = 341.000000
r00 = -1.000000
r01 = 0.000000
r02 = 0.000000
r10 = 0.000000
r11 = -1.000000
r12 = 0.000000
r20 = 0.000000
r21 = 0.000000
r22 = 1.000000
```



## 9.6.2. MPLcalibBytes.dat

Byte number (decimal)	Data	Datatype
0 to 3	Calibration size	long
4 to 7	Checksum	uint32
8 to 9	Key	uint16
10 to 19		
20 to 23	Compass offset - X axis	long
24 to 27	Compass offset - Y axis	long
28 to 31	Compass offset - Z axis	long
32 to 35	Gyroscope offset - X axis	long
36 to 39	Gyroscope offset - Y axis	long
40 to 43	Gyroscope offset - Z axis	long
44 to 47	Gyroscope Temperature	long
48 to 51	Accelerometer offset - X axis	long
52 to 55	Accelerometer offset - Y axis	long
56 to 59	Accelerometer offset - Z axis	long
60 to 63	Accelerometer Temperature	long
64 to 75		
76 to 77	Gyroscope accuracy	int16
78 to 79	Accelerometer accuracy	int16
80 to 81	Compass accuracy	int16
82 to 105		

*Table 9-7: The deciphered contents of the InvenSense MPL calibration values. All data is in little endian format.*

## 9.7. Legacy Firmware: SDLog v0.12.0 (or earlier): Configuration header

### **Byte # 76-235: Calibration parameters**

Bytes 76 to 235 store the calibration data for individual *Shimmer3* sensors. These parameters are used to calibrate the RAW data during post-experimentation analysis. The layout and description of these bytes can be found in the Appendix section of this document (*i.e.* Section 8.3).

### **Byte # 236 - 251: Reserved for future use**

## 9.8. As with the normal calibration Legacy Firmware: SDLog v0.12.0 (or earlier): InvenSense MPL calibration

As with the normal calibration method, on commencement of logging, if a calibration.ini file is found by the firmware, the appropriate parameters from the file will be stored in the configuration header for the enabled sensors. If no calibration file is found, then default calibration values will be stored for all sensors. If a calibration file is found but it does not contain parameters for a given enabled sensor, or for the configured sensor range, then default calibration values are stored for that sensor.

## 9.9. Legacy Firmware: SDLog v0.12.0 (or earlier): Writing calibration parameters to file

Users should note that this is not a recommended method of providing calibration parameters for the Shimmer when using *SDLog v0.9.0* (or later).

To supply calibration parameters by file, the user should create a folder called "Calibration" or "calibration" in the top-level SD card directory and should save calibration files within that folder. The format of the file should match the output given by the *Shimmer 9DoF Calibration* application's "Save to file" function. There should be a separate file for each sensor and range (as required) and the filenames should follow the defined convention, in order to identify the sensor and range for which the parameters apply, as listed in Table 9-6 below.

Sensor	Range	Filename
Low noise accelerometer	2 g	<i>calib_accel_ln_2g.ini</i>
	4 g	<i>calib_accel_wr_4g.ini</i>
Wide range accelerometer	8 g	<i>calib_accel_wr_8g.ini</i>
	16 g	<i>calib_accel_wr_16g.ini</i>
Gyroscope	250 dps	<i>calib_gyro_250dps.ini</i>
	500 dps	<i>calib_gyro_500dps.ini</i>
	1000 dps	<i>calib_gyro_1000dps.ini</i>
	2000 dps	<i>calib_gyro_2000dps.ini</i>
Magnetometer	1.3 Ga	<i>calib_mag_13ga.ini</i>
	1.9 Ga	<i>calib_mag_19ga.ini</i>
	2.5 Ga	<i>calib_mag_25ga.ini</i>
	4.0 Ga	<i>calib_mag_40ga.ini</i>
	4.7 Ga	<i>calib_mag_47ga.ini</i>

	5.6 Ga	<i>calib_mag_56ga.ini</i>
	8.1 Ga	<i>calib_mag_81ga.ini</i>

Table 9-6 Calibration file name and keyword

The provision of a separate calibration file for each sensor and range was a new feature in *SDLog v0.9.0* (or later) and it was implemented to enable the simultaneous storage of calibration parameters for multiple ranges of a given sensor type, which was previously not possible. Backwards compatibility has been included, such that any existing legacy calibration files, with filename "*calibParams.ini*" (case sensitive), may be used with *SDLog v0.9.0* (or later).

Calibration parameters are read from the calibration folder or from the InfoMem according to the following:

- If the 'calibration changed' flag is set, the firmware will read the calibration parameters from InfoMem and will replace any existing calibration file(s).
- If the 'calibration changed' flag is not set, the firmware will check the range settings of the enabled sensors and look for an appropriate calibration file on the SD card. Assuming that the file exists, the calibration will be read and copied to InfoMem.
- If the 'calibration changed' flag is not set and no appropriate calibration file exists on the SD card, the firmware will check that there is a valid set of calibration parameters on the InfoMem. Assuming that there is a valid set, (a) new calibration file(s) will be written with the calibration parameters from InfoMem.
- If the 'calibration changed' flag is not set, no appropriate calibration file exists on the SD card, and there is not a valid set of calibration parameters in the InfoMem, the firmware will assume default calibration parameters, as given in Table 9-7. No file will be created in this case.

	Offset (all axes)	Sensitivity (all axes)	Alignment Matrix
<b>WR Accelerometer</b>	0	$\pm 2 \text{ g} \rightarrow 1631 \text{ LSB}/(\text{m/s}^2)$ $\pm 4 \text{ g} \rightarrow 815 \text{ LSB}/(\text{m/s}^2)$ $\pm 8 \text{ g} \rightarrow 408 \text{ LSB}/(\text{m/s}^2)$ $\pm 16 \text{ g} \rightarrow 135 \text{ LSB}/(\text{m/s}^2)$	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & +1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$
<b>LN Accelerometer</b>	2047 LSB	$\pm 2 \text{ g} \rightarrow 83 \text{ LSB}/(\text{m/s}^2)$	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}$
<b>Gyroscope</b>	0	$\pm 250 \text{ dps} \rightarrow 131 \text{ LSB}/\text{dps}$ $\pm 500 \text{ dps} \rightarrow 65.5 \text{ LSB}/\text{dps}$ $\pm 1000 \text{ dps} \rightarrow 32.8 \text{ LSB}/\text{dps}$ $\pm 2000 \text{ dps} \rightarrow 16.4 \text{ LSB}/\text{dps}$	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}$
<b>Magnetometer</b>	0	$\pm 1.3 \text{ Ga} \rightarrow 1100 \text{ LSB}/\text{Ga}$ $\pm 1.9 \text{ Ga} \rightarrow 855 \text{ LSB}/\text{Ga}$ $\pm 2.5 \text{ Ga} \rightarrow 670 \text{ LSB}/\text{Ga}$ $\pm 4.0 \text{ Ga} \rightarrow 450 \text{ LSB}/\text{Ga}$ $\pm 4.7 \text{ Ga} \rightarrow 355 \text{ LSB}/\text{Ga}$ $\pm 5.6 \text{ Ga} \rightarrow 330 \text{ LSB}/\text{Ga}$ $\pm 8.1 \text{ Ga} \rightarrow 230 \text{ LSB}/\text{Ga}$	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & +1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$

Table 9-7: Default calibration parameters set in firmware if no calibration file present on microSD card.

## 9.10. Legacy Firmware: SDLog v0.11.0 (or earlier) data log

Byte 0:	offset sign	
Byte 1:	offset magnitude byte 0 (MSB)	
Byte 2:	offset magnitude byte 1	
Byte 3:	offset magnitude byte 2	
Byte 4:	offset magnitude byte 3 (LSB)	
Byte 5:	timestamp 0 byte 0 (MSB)	
Byte 6:	timestamp 0 byte 1 (LSB)	
Byte 7:	sample 0 channel 0 byte 0 (MSB)	(XAccel)
Byte 8:	sample 0 channel 0 byte 1 (LSB)	(XAccel)
Byte 9:	sample 0 channel 1 byte 0 (MSB)	(YAccel)
Byte 10:	sample 0 channel 1 byte 1 (LSB)	(YAccel)
Byte 11:	sample 0 channel 2 byte 0 (MSB)	(ZAccel)
Byte 12:	sample 0 channel 2 byte 1 (LSB)	(ZAccel)
Byte 13:	sample 0 channel 3 byte 0 (MSB)	(XGyro)
Byte 14:	sample 0 channel 3 byte 1 (LSB)	(XGyro)
Byte 15:	sample 0 channel 4 byte 0 (MSB)	(YGyro)
Byte 16:	sample 0 channel 4 byte 1 (LSB)	(YGyro)
Byte 17:	sample 0 channel 5 byte 0 (MSB)	(ZGyro)
Byte 18:	sample 0 channel 5 byte 1 (LSB)	(ZGyro)
Byte 19:	timestamp 1 byte 0 (MSB)	
Byte 20:	timestamp 1 byte 1 (LSB)	
Byte 21:	sample 1 channel 0 byte 0 (MSB)	(XAccel)
Byte 22:	sample 1 channel 0 byte 1 (LSB)	(XAccel)

## 9.11. Legacy Firmware: SDLog v0.8.0 (or earlier) synchronisation

This option requires exactly one Shimmer to be designated as the “master” Shimmer and any other Shimmers to be designated as “slaves”. Enabling time synchronisation allows samples from multiple Shimmers to be converted to a common reference clock (the clock of the master Shimmer).

It is important to note that, in this synchronisation method, no effort is made to alter the clock frequency of any Shimmer in the firmware. Instead, each clock runs at its own frequency throughout logging and the alignment of samples to a common reference clock is done in post-processing. The basis of the alignment is a log of offsets between each Shimmer’s local timestamp and master timestamps. On the *Shimmer3*, the slaves each connect with the master via Bluetooth (BT) at regular intervals and the master sends its timestamp at that instant to the connected slave. The slaves must be within radio range (approximately 10m) of the master for successful communication.

One important synchronisation parameter is determined by the user in the *sdlog.cfg* file: the broadcast interval (denoted BI in the following). If the user does not provide a value for BI, it defaults to 120 seconds.

### **Master:**

- When the master Shimmer is powered on or reset, its on-board 32 kHz clock will begin counting at zero. This clock will be the common reference clock for all samples in the experiment.
- When the master starts logging, it will save its initial timestamp with a resolution of 32 bits (see Section 7.2.2 for a description on where this initial timestamp is saved).
- On *Shimmer3*:
  - Throughout logging, the master will periodically turn on its BT radio at regular intervals of BI seconds. When a connection is successfully made with the Bluetooth from a slave device, the master will send its timestamp with a resolution of 32 bits.
  - The master turns on its radio for a pre-determined amount of time and waits for connections from slave devices. After the pre-determined length of time has elapsed, the master turns off its radio.

### **Slave:**

- When a slave Shimmer is powered on or reset, its on-board 32 kHz clock will begin counting at zero.
- When the slave starts logging, it will save its initial timestamp with a resolution of 32 bits (see Section 7.2.2 for a description on where this initial timestamp is saved).
- On *Shimmer3*:
  - The slave will periodically turn on its BT module and connect with the master device, according to the MAC address provided in the *sdlog.cfg* file.
  - Upon successful connection, the slave will receive a message from the master containing the master's timestamp.
  - The slave Shimmer will turn off its radio immediately after receiving a master timestamp message or if the connection attempt is unsuccessful.
  - If the previous connection attempt was unsuccessful, the slave will repeat the above steps at a short interval; otherwise, the slave will repeat the communication steps every BI seconds.
  - This sequence repeats throughout logging.
- Whenever the slave receives a timestamp, it immediately records the time on its local 32 kHz clock.
- The difference between the slave's local time and the received master time (local time - master time) is calculated, as follows:

- The *offset sign* (one byte) represents the sign of the difference (0 (positive) if local time  $\geq$  master time and 1 (negative) otherwise).
  - The magnitude of the difference ( $|\text{local time} - \text{master time}|$ ), with a resolution of 32 bits, is represented by a 4-byte unsigned integer.
- The 5 bytes of the offset (sign and magnitude) are written to the SD card at the start of every write operation. There will be many write operations for each time offset update; for example, a write operation might occur many times every second, whilst the default value for the broadcast interval is 120 seconds. At any given write operation, if a new master timestamp has not been received since the previous write operation, then the value 0xFFFFFFFF is written for the offset magnitude, with a sign byte of 0. This value must be recognised in software to denote an “invalid” value and be removed before processing the offsets.

## 9.12. Legacy Firmware: SDLog v0.5.0 file contents

### 9.12.1. Configuration header

Each data file (e.g. *data/experiment1/device1-000/000*), begins with a header with the configuration information. The structure of this header is outlined in the tables below, where each row of each table represents one byte and the individual bits are separated, where appropriate, depending on whether each bit represents an independent binary value or the entire byte contains a single value. Empty cells and cells with constant values represent bits that are reserved for future use. For *Shimmer3*, there are a total 178 bytes in the header.

#### Byte # 0 – 9: Enabled Sensors

Byte #	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0	Acc (LN)*	Gyr	Mag	N/A	N/A	GSR	ExtCh 7	ExtCh 6
1	N/A	N/A	Battery	Acc (WR)*	ExtCh 15	IntCh 1	IntCh 12	IntCh 13
2	IntCh 14					pres	N/A	exp_pwr
3								
4								
5								
6								
7								
8								
9								

\* There are multiple accelerometers on the *Shimmer3* circuit board. The low noise (LN) analog accelerometer on the KXRB5-2042 chip is enabled via Byte 0, Bit 7 (Acc (LN)), whilst the wide range digital accelerometer on the LSM303DLHC chip is enabled via Byte 1, Bit 4 (Acc (WR)).

#### Byte # 10 – 19: Trial Configuration

Byte #	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
10	SV Reg	PMUX	User button	Gyro button	1	Sync	Master	0
11	Single touch Start	Accel LPM	Accel HRM	txxo				
12								
13								
14								
15	Broadcast Interval							
16								
17								
18								
19								



### Byte # 20 – 29: Sensor Configuration

Byte #	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
20	Sample Rate (MSB)							
21	Sample Rate (LSB)							
22	Digital Accel Range							
23	GSR Range							
24	Mag Range							
25	Mag Rate							
26	Digital Accel Rate							
27	Gyro Rate							
28	Gyro Range							
29	Pressure Precision							

### Byte # 30 – 39: Firmware and Shimmer Parameters

Byte #	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
30	ShimmerVersion (MSB)							
31	ShimmerVersion (LSB)							
32	MyTrialId							
33	Nshimmer							
34	FW Version - Type (MSB)							
35	FW Version - Type (LSB)							
36	FW Version - Major (MSB)							
37	FW Version - Major (LSB)							
38	FW Version - Minor							
39	FW Version - Release							

*Shimmer Version* indicates the hardware version for which the code was compiled. *Shimmer1* is denoted by 0, *Shimmer2* by 1, *Shimmer2r* by 2 and *Shimmer3* by 3.

The *FW Version* bytes define the firmware version. The *Type* field will always be 2 for *SDLog* firmware images. *Major* and *Minor* versions are indicated by a two- and one-byte value, respectively (e.g. for *SDLog v1.2*, *Major* = 1 and *Minor* = 2). The *Release* field can be ignored by users; it will have a value of 0.

### Byte # 40 – 51: Reserved for future use

### Byte # 52 - 55: Configuration Time

Byte #	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
52	Config-Time 0 (MSB)							
53	Config-Time 1							
54	Config-Time 2							
55	Config-Time 3 (LSB)							

This value comes from the *Configuration ID* parameter from Section 6.1.5.



### Byte # 56-76: Digital Accelerometer Calibration

For *Shimmer3*, these parameters refer to the LSM303DLHC accelerometer. There are other accelerometers also populated on the *Shimmer3 mainboard*; the associated calibration parameters for the other devices are stored in a later section of the configuration header.

Byte #	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
56	Digital Accel Calibration Offset X (MSB)							
57	Digital Accel Calibration Offset X (LSB)							
58	Digital Accel Calibration Offset Y (MSB)							
59	Digital Accel Calibration Offset Y (LSB)							
60	Digital Accel Calibration Offset Z (MSB)							
61	Digital Accel Calibration Offset Z (LSB)							
62	Digital Accel Calibration Gain X (MSB)							
63	Digital Accel Calibration Gain X (LSB)							
64	Digital Accel Calibration Gain Y (MSB)							
65	Digital Accel Calibration Gain Y (LSB)							
66	Digital Accel Calibration Gain Z (MSB)							
67	Digital Accel Calibration Gain Z (LSB)							
68	Digital Accel Calibration Align XX							
69	Digital Accel Calibration Align XY							
70	Digital Accel Calibration Align XZ							
71	Digital Accel Calibration Align YX							
72	Digital Accel Calibration Align YY							
73	Digital Accel Calibration Align YZ							
74	Digital Accel Calibration Align ZX							
75	Digital Accel Calibration Align ZY							
76	Digital Accel Calibration Align ZZ							

### Byte # 77 - 97: Gyroscope Calibration

Byte #	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
77	Gyro Calibration Offset X (MSB)							
78	Gyro Calibration Offset X (LSB)							
79	Gyro Calibration Offset Y (MSB)							
80	Gyro Calibration Offset Y (LSB)							
81	Gyro Calibration Offset Z (MSB)							
82	Gyro Calibration Offset Z (LSB)							
83	Gyro Calibration Gain X (MSB)							
84	Gyro Calibration Gain X (LSB)							
85	Gyro Calibration Gain Y (MSB)							
86	Gyro Calibration Gain Y (LSB)							
87	Gyro Calibration Gain Z (MSB)							
88	Gyro Calibration Gain Z (LSB)							
89	Gyro Calibration Align XX							
90	Gyro Calibration Align XY							
91	Gyro Calibration Align XZ							
92	Gyro Calibration Align YX							
93	Gyro Calibration Align YY							
94	Gyro Calibration Align YZ							
95	Gyro Calibration Align ZX							
96	Gyro Calibration Align ZY							
97	Gyro Calibration Align ZZ							

### **Byte # 98 - 118: Magnetometer calibration**

Byte #	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
98	Mag Calibration Offset X (MSB)							
99	Mag Calibration Offset X (LSB)							
100	Mag Calibration Offset Y (MSB)							
101	Mag Calibration Offset Y (LSB)							
102	Mag Calibration Offset Z (MSB)							
103	Mag Calibration Offset Z (LSB)							
104	Mag Calibration Gain X (MSB)							
105	Mag Calibration Gain X (LSB)							
106	Mag Calibration Gain Y (MSB)							
107	Mag Calibration Gain Y (LSB)							
108	Mag Calibration Gain Z (MSB)							
109	Mag Calibration Gain Z (LSB)							
110	Mag Calibration Align XX							
111	Mag Calibration Align XY							
112	Mag Calibration Align XZ							
113	Mag Calibration Align YX							
114	Mag Calibration Align YY							
115	Mag Calibration Align YZ							
116	Mag Calibration Align ZX							
117	Mag Calibration Align ZY							
118	Mag Calibration Align ZZ							

**Byte # 119 - 130: Reserved for future use**

**Byte # 131 - 151: Analog Accelerometer Calibration**

Byte #	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
131	Analog Accel Calibration Offset X (MSB)							
132	Analog Accel Calibration Offset X (LSB)							
133	Analog Accel Calibration Offset Y (MSB)							
134	Analog Accel Calibration Offset Y (LSB)							
135	Analog Accel Calibration Offset Z (MSB)							
136	Analog Accel Calibration Offset Z (LSB)							
137	Analog Accel Calibration Gain X (MSB)							
138	Analog Accel Calibration Gain X (LSB)							
139	Analog Accel Calibration Gain Y (MSB)							
140	Analog Accel Calibration Gain Y (LSB)							
141	Analog Accel Calibration Gain Z (MSB)							
142	Analog Accel Calibration Gain Z (LSB)							
143	Analog Accel Calibration Align XX							
144	Analog Accel Calibration Align XY							
145	Analog Accel Calibration Align XZ							
146	Analog Accel Calibration Align YX							
147	Analog Accel Calibration Align YY							
148	Analog Accel Calibration Align YZ							
149	Analog Accel Calibration Align ZX							
150	Analog Accel Calibration Align ZY							
151	Analog Accel Calibration Align ZZ							

### Byte # 152 - 173: Temperature (BMP180) and Pressure Calibration

Byte #	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
152								Temp & Pres Calibration AC1_MSB
153								Temp & Pres Calibration AC1_LSB
154								Temp & Pres Calibration AC2_MSB
155								Temp & Pres Calibration AC2_LSB
156								Temp & Pres Calibration AC3_MSB
157								Temp & Pres Calibration AC3_LSB
158								Temp & Pres Calibration AC4_MSB
159								Temp & Pres Calibration AC4_LSB
160								Temp & Pres Calibration AC5_MSB
161								Temp & Pres Calibration AC5_LSB
162								Temp & Pres Calibration AC6_MSB
163								Temp & Pres Calibration AC6_LSB
164								Temp & Pres Calibration B1_MSB
165								Temp & Pres Calibration B1_LSB
166								Temp & Pres Calibration B2_MSB
167								Temp & Pres Calibration B2_LSB
168								Temp & Pres Calibration MB_MSB
169								Temp & Pres Calibration MB_LSB
170								Temp & Pres Calibration MC_MSB
171								Temp & Pres Calibration MC_LSB
172								Temp & Pres Calibration MD_MSB
173								Temp & Pres Calibration MD_LSB

#### 9.12.2. Initial Timestamp

Immediately after the configuration header, there is a 4-byte timestamp, whose value is the time on the local Shimmer clock (in units of ticks of the 32 kHz clock) when the first sample in that file was recorded. It is used for aligning the data from multiple Shimmers to a common clock.

174	initial time stamp
175	initial time stamp
176	initial time stamp
177	initial time stamp

#### 9.12.3. Data log

Following these 178 bytes, the logged sensor data begins (i.e. at the 179th byte in the file). The data is written in blocks of up to 512 bytes. These blocks are continuously written until data has been logged to a given file for one hour, at which time the current file is closed and a new file is opened, with the same format as the current one (i.e. the configuration header is repeated in every file).

## Order of data channels

The order in which the sensors appear in the data channels depends on which sensors are enabled. The firmware will assign the channels in the order outlined below (left to right). Note that “Accel (LN)” refers to the low noise accelerometer, whilst “Accel (WR)” refers to the wide range accelerometer. The number of channels per sensor is listed below the channel name. All channels are 2 bytes each except where otherwise stated i.e. Pressure\_bmp180 and all channels are little endian.

Analog channels										Digital channels						
Accel (LN)	Battery	Ext Exp A7	Ext Exp A6	Ext Exp A15	Int Exp A1	Int Exp A12	Int Exp A13	Int Exp A14	mcp430_Temperature	Gyro_mpu	Accel (WR)_Ism	Mag_Ism	Accel_mpu	Mag_mpu	Temperature_bmp180	Pressure_bmp180
3	1	1	1	1	1	1	1	1	1	3	3	3	3	3	1	1
																3bytes

*Table 9-8: Order of data channels for legacy versions of SDLog firmware (i.e., less than or equal to SDLog v0.5.0).*

## 9.13. Troubleshoot

### ***Red and yellow LEDs flash when I undock or reboot my Shimmer.***

The red and yellow LEDs flashing on the Shimmer indicate that the firmware either cannot create the required directories or cannot write to the required files on the SD card. Try the following steps to rectify the problem:

1. Check that the SD card is correctly inserted.
2. Ensure that the microSD card is less than 32GB in capacity, is not an XC (eXtended Capacity) card and that it supports 1-bit SPI mode.
3. Ensure that the SD card memory is not full.
4. Ensure that the *experimentid* and *shimmername* parameters, specified in the *sdlog.cfg* file contain only alphanumeric characters (a,..., z, A,..., Z, 0,..., 9), dash ('-') and underscore ('\_').

**Note:** The above SD error LED sequence is present in SDLog\_v0.17.0 and later. Older versions of SDLog couple both SD and RTC errors into a 0.1s Blue/0.1s Green LED flashing sequence.

### ***When I undock or power on my Shimmer, the green LED continuously flashes at a rate of 5Hz.***

Try power cycling the Shimmer. The problem may be due to an error in Bluetooth initialisation.

If the problem persists after power cycling, try changing the SD card for a newer one - if the SD card is corrupt, the firmware may fail to correctly read the configuration file.

### ***The data file is empty after logging.***

Ensure that you log for a minimum of one minute in order for data to be written to the SD card.

### ***The configuration does not match the parameters in the sdlog.cfg file.***

The *sdlog.cfg* file is read once each time the Shimmer is rebooted or undocked so the configuration will always match the most recent configuration file at the time of logging. If the parameters in the configuration header of your data files do not match those in the *sdlog.cfg* file, it is likely that you have changed the *sdlog.cfg* file contents since logging the data in question.

### ***The calibration parameters in the configuration header do not match the Calibration/calibParams.ini file.***

Calibration parameters will only be loaded for sensors that are enabled; calibration parameters for disabled sensors will all have zero value.

If a sensor is enabled and the calibration parameters do not match the calibration file, try the following steps to rectify the problem:

1. Ensure that the calibration file is stored in the correct file location from the SD card top level directory, according to the following (case-sensitive) options:
  - /Calibration/calibParams.ini

- /calibration/calibParams.ini

No other file path will be recognised by the firmware.

2. Ensure that you have implemented the correct byte-order and endianness when you read the calibration parameters from the configuration header, according to the information in Section 7.2.

***How do I interpret the files on the SD card?***

The files on the SD card need to be converted from binary format.

***How do I convert the files on the SD card from binary format?***

It is recommended to use *ConsensysBASIC* or *ConsensysPRO* software, which is available for download from our [website](#)<sup>1</sup>.



**Shimmer International Offices:**

Europe – Dublin, Ireland.

USA – Boston, MA.

Asia – Kuala Lumpur, Malaysia.

**Web:** [www.ShimmerSensing.com](http://www.ShimmerSensing.com)

**Email:** [info@ShimmerSensing.com](mailto:info@ShimmerSensing.com)

 [www.Shimmersensing.com](http://www.Shimmersensing.com)

 [/ShimmerResearch](https://www.facebook.com/ShimmerResearch)

 [@ShimmerSensing](https://twitter.com/ShimmerSensing)

 [/company/Shimmer](https://www.linkedin.com/company/Shimmer)

 [/ShimmerSensing](https://www.youtube.com/ShimmerSensing)

 [/ShimmerResearch](https://www.instagram.com/ShimmerResearch)