LPMS-CU

Reference Manual

Version 1.1.0

www.lp-research.com

I. INTRODUCTION

Welcome to the LP-RESEARCH Motion Sensor CAN bus and USB version (LPMS-CU) User's Manual!

In this manual we will try to explain everything you need to know to set up the LPMS-CU hardware, install and use its software, as well as getting started with integrating the sensor within your own software project. We have put a lot of effort into making the LPMS-CU a great product, but we are always eager to improve and work on new developments. So, if you have any further question or have any comment regarding this manual please feel free to contact us anytime.

For more information on the LPMS-CU or other product series, please refer to the flyers, datasheets or user manuals, available from the LP-RESEARCH Inc. website at the following address: http://www.lp-research.com.

II. TABLE OF CONTENTS

I. INTRODUCTION **2**

II. TABLE OF CONTENTS **3**

III. REVISION HISTORY **6**

IV. DEVICE OVERVIEW **7**

Measurement Output 7

Technical Background 7

Communication Methods 8

Calibration 8

Magnetic Field Distortion Compensation 8

Size and Run-times 8

Application Areas 9

V. DEVICE SPECIFICATIONS **10**

VI. CONNECTOR CONFIGURATION **11**

Plastic casing LPMS-CU 11

Aluminum casing LPMS-CU 11

VII. OPERATION **12**

Powering Up and Operation Modes 12

Host Device Communication 13

Communication through USB Interface 13

Communication through CAN Bus Interface 13

Data Acquisition 13

Raw Sensor Data 13

Orientation Data 14

Filter Settings 14

Filter Modes 14

Magnetometer Correction Setting 15

Acceleration Compensation Setting 16

Gyroscope Threshold 17

Gyroscope Auto-calibration Function 17
Low Pass Filter Setting 17
Trade-offs and Limitations 18
Calibration Methods 18
Gyroscope Calibration 18
Magnetometer Calibration 19
Accelerometer Calibration 19
VIII. COMMUNICATION PROTOCOL 21
Establishing a Connection 21
In this manual the LpBus and LpCAN protocol will be explained first. Please skip the next two sections, if you are just interested in reading sensor data via the CAN bus. 21
LpBUS Protocol 21
Basic Command Introduction 21
Packet Format 22
Data Format in a Packet Data Field 23
LpCAN Protocol 24
CANOpen Protocol 25
Protocol Commands List 26
Acknowledged and Not-acknowledged Identifier 26
Firmware Update and In-Application-Programmer Upload Command 26
Configuration and Status Command 27
Mode Switching Command 28
Data Transmission Command 29
Register Value Save and Reset Command 31
Reference Setting and Offset Reset Command 32
Self-Test Command 32
IMU ID Setting Command 32
Gyroscope Settings Command 33
Accelerometer Settings Command 34
Magnetometer Settings Command 35
Filter Settings Command 37
CAN Bus Settings Command 38
Additional Settings 39
Example Communication 42

Request Sensor Configuration 42
Request Gyroscope Range 43
Set Accelerometer Range 44
Read Sensor Data 45
IX. OpenMAT 47
Overview 47
Introduction 47
Application Installation 47
LpmsControl Software Operation 48
Overview 48
GUI Elements 49
Device Discovery 52
Connecting and Disconnecting a Device 52
Sensor Parameter Adjustment 52
Reset of Orientation and Reference Vectors 53
How to Upload New Firmware 54
The LpSensor Library 54
Building Your Application 54
Important Classes 55
Example Code 62
X. MECHANICAL INFORMATION 65
LPMS-CU Dimension 65

III. REVISION HISTORY

Date	Revision	Changes
01-May-2012	1.0	Initial release.
01-Sep-2012	1.0.11	The introduction part of LPMS-B has been removed, and summarized into another separated document. The whole manual includes only the information of LPMS-CU.
17-Sep-2012	1.0.12	- Updates to reflect the latest changes in the firmware command set. - OpenMAT library section contains more details on how to use the binary LpSensor library. - Section on how to compile LpmsControl was removed.
25-Feb-2013	1.1.0	- Correction of some bugs of system sampling timing. - GUI has been optimized by adding more tool bars. - Up to 4 sensor 3 D visualization view mode is added. - Altitude calculation by using pressure sensor is included. - Another two Euler filter modes are added. - Low pass filter setting is added. - CANopen protocol has been optimized.

IV. DEVICE OVERVIEW

Measurement Output

The LP-RESEARCH Motion Sensor CAN bus and USB version (LPMS-CU) is a wired inertial measurement unit. We designed the unit to be as small as possible so that it can be used in a wide range of applications from measuring the human motion to the stabilization of ground vehicles or air-planes. The unit can measure orientation in 360 degrees about all three global axes. Measurements are taken digitally and transmitted to a data analysis system in the form of orientation quaternion or Euler angles. Whereas Euler angles are the traditional way of describing the orientation of an object, quaternion allow orientation measurement without encountering the so-called Gimbal's lock by using a four-element vector to express orientation around all axes without being limited by singularities. A more in-depth explanation of the quaternion output of the LPMS-CU will follow later on in this manual.

In addition, a pressure sensor is selectable on the LPMS-CU for some specific applications, such as GPS navigation enhancement, indoor and outdoor navigation, vertical velocity indication etc.

Technical Background

To measure the orientation of an object, the sensor internally uses three different sensing units. These units are micro-electro-mechanical system (MEMS) sensors that integrate complex mechanical and electronic capabilities on a miniaturized device. The units used in the LPMS-CU for orientation determination are a 3-axis gyroscope (detecting angular velocity), a 3-aixs accelerometer (detecting the directing of the earth's gravity field) and a 3-axis magnetometer to measure the direction of the earth magnetic field. In principle orientation data about all three room axes can be determined by integrating the angular velocity data from the gyroscope. However through the integration step the error from the gyroscope measurements, although it might be very small, has an exponential influence on the calculation result. Therefore we correct the orientation data from the gyroscope with information from the accelerometer (roll and pitch angles) and magnetometer (yaw angle) to calculate orientation information of high accuracy and stability while guaranteeing fast sampling rates. We combine the orientation information from the three sensing units using a complementary filter in conjunction with an extended Kalman filter (EKF). The Kalman filter allows us to reduce the measurement error especially in case of regular movements (e.g. human gait analysis, vehicle vibration analysis etc.). Sampling rates of the sensor can be adjusted to up to 300 Hz internal measurement frequency.

Communication Methods

Data can be transferred either using a CAN Bus network or a universal serial bus (USB) connection. For communication protocols we rely on commonly used open standard protocols: In case of USB interface we use a modified ModBus protocol (LpBUS) and in case of the CAN Bus interface we offer communication through a simplified CANOpen implementation or our proprietary LpCAN protocol.

Calibration

For accurate operation the sensor needs to be calibrated. The calibration procedure includes the determination of the gyroscope data offset, gyroscope movement threshold, accelerometer misalignment, accelerometer offset, and magnetometer interference bias and gain. As the earth magnetic field can be distorted by metal or electromagnetic sources within the vicinity of the sensor, the re-calibration of the magnetic sensor and re-calculation of the magnetic reference vector of the sensor might be necessary when using the sensor in different location or under varying experiment environments. Later in this manual we will describe in detail the necessary calibration procedures and measures to be taken to guarantee the accuracy of the measurements taken by the sensor. We tried to automate the calibration procedures as far as possible inside the firmware of the sensor to make the usage as convenient as possible for the users.

Magnetic Field Distortion Compensation

Additionally to the established method of compensating a distorted earth magnetic field by re-adjusting the magnetometer bias and gain, the LPMS-CU offers either completely switching off the magnetometer compensation of the gyroscope data or selectively switching the compensation modes between: dynamic, weak, medium and strong magnetometer correction, in places where an earth magnetic field outside the normal limits is being detected. We implemented a special algorithm that allows switching between operation with different modes of magnetometer compensation and without magnetometer compensation without any inconsistencies in the orientation detection. For further adjustment of the calibration parameters to the sensor environment a temperature sensor and pressure sensor have been integrated on the LPMS-CU. Data from these indicators can be utilized by the user to correct raw data measurements from the LPMS-CU sub-sensors.

Size and Run-times

During the development of the LPMS-CU we tried to make the unit as small as possible to allow a large variety of application areas. For size reduction the actual sensing units and microcontroller hardware are integrated into one main-board with 6-layers PCB design. The communication hardware interface is implemented on an extension-board, which is stacked above the main-board. Each LPMS-CU consists of these two boards as a whole unit. The main-board contains the actual sensor devices and manages the sensor data acquisition. The extension-board contains the CAN Bus and USB hardware to communicate with data logging devices.

Application Areas

The LPMS-CU is suitable for a wide range of applications. One application focus for a small scale motion sensor is the measurement of human movement for injury rehabilitation, gait cycle analysis, surgical skill training and evaluation etc. The sensor can also be effectively used in the field of virtual reality, navigation, robotics, or for measuring vehicle dynamics. If more than one sensor is used for a sensor network the motion of complex objects as necessary in cinematic motion capturing or animation movie production is possible.

V. DEVICE SPECIFICATIONS

Currently two packaged versions of the LPMS-CU are available; one with plastic casing, and another one with aluminum casing. The plastic casing LPMS-CU is implemented for both CAN Bus and USB communication, but the aluminum casing is implemented only for CAN Bus communication. Please see the below table of the summary of sensor specification. Please refer to the section "*X. MECHNICAL INFORMATION*" for detail introduction of package layout.

Wired Interface	CAN Bus	USB 2.0
Maximum baudrate	1Mbit/s	921.6Kbit/s
Communication protocol	LpCAN / CANOpen	LpBUS
Size	Plastic case: 37 x 28 x 17 mm Aluminum case: 49 x 39 x 25 mm	
Weight	Plastic case: 12.8 g Aluminum case: 70.4 g	
Orientation	360^{O} about all axes	
Resolution	< 0.05°	
Accuracy	< 2° RMS (dynamic), < 0.5°(static)	
Accelerometer	3-axis, ±20 / ±40 / ±80 / ±160 m/s², 16 bits	
Gyroscope	3-axis, ±250 / ±500 / ±2000 ^{O}/s, 16 bits	
Magnetometer	3-axis, ±130 ～ ±810 uT, 16 bits	
Pressure sensor	300 ～ 1100 hPa *	
Data output format	Raw data / Euler angle / Quaternion	
Sampling rate	0 ~ 300 Hz.	
Latency	5ms	
Power consumption	165 mW	
Supply voltage (Vcc)	5 ～ 18 V DC	
Connector	Plastic case: Micro USB, type B Aluminum case: ASX002-05SN-HE	
Temperature range	- 40 ～ +80 ^{O}C	
Software	C++ library for Windows, Java library for Android, LpmsControl utility software for Windows, Open Motion Analysis Toolkit (OpenMAT) for Windows	

* The pressure sensor is optional and can be added on LPMS-CU, which depends on the requirement from users. Please contact us for more information about this.

VI. CONNECTOR CONFIGURATION

Plastic casing LPMS-CU

There are two connectors on the plastic casing LPMS-CU marked as “CAN” and “USB” on the top of the sensor case. Please see the pin-out for both connectors below.

Pin description:

Pin (CAN port)	**1**	**2**	**3**	**4**	**5**
Function	*CAN_V+*	*CAN_L*	*CAN_H*	*Reset*	*CAN_GND*

Pin (USB port)	**1**	**2**	**3**	**4**	**5**
Function	*+5V*	*D-*	*D+*	*None*	*GND*

Connector type: Micro-USB type B female

Remark: A 120 Ohm CAN Bus termination resistor is NOT integrated in the LPMS-CU. Should it be necessary to add the termination resistor to the sensor for your specific system please contact us for detailed instructions. Under normal circumstances your system should be operable by connecting the sensor to your system without including the termination resistor.

IMPORTANT: The two connectors cannot be used at the same time.

Aluminum casing LPMS-CU

There is only one connector marked as “CAN” on the top of aluminum casing LPMS-CU. Please see the connector pin-out as below.

Pin description:

Pin no.	**1**	**2**	**3**	**4**	**5**
Function	*CAN_SHLD*	*CAN_V+*	*CAN_GND*	*CAN_H*	*CAN_L*

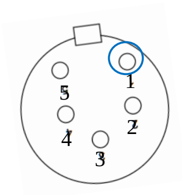
CAN_SHLD: Optional CAN shield

CAN_V+: External positive power supply

CAN_GND: Ground

CAN_H: CAN_H bus line (dominant high)

CAN_L: CAN_L bus line (dominant low)



Connector type: ASX connector, part number: ASX002-05SN-HE

VII. OPERATION

Powering Up and Operation Modes

The LPMS-CU sensor is switched on by connecting the sensor with a power source, either by USB or via the power lines of the CAN bus connector. In the case of plastic casing LPMS-CU, the red and green LEDs visible on the top of the sensor light up when operation power is supplied to the device. After about 3 seconds, the green color status LED will start blinking with an interval of 1s, which means the sensor is ready for connection. There are 3 different modes for operation:

Mode	Description
Command mode	In command mode the functionality of the sensor is accessed command-by-command. Also data is transferred from the sensor to the user by a special command. This mode is suitable for making adjustments to the parameter settings of the sensor and synchronized data-transfer.
Streaming mode	In streaming mode data is continuously sent from the sensor to the host. This mode is suitable for simple and high-speed data acquisition. Sensor parameters cannot be set in this mode.
Sleep mode	Sleep mode is the power-saving state of the sensor. The sensor can be woken up by switching into streaming mode or command mode. In this mode no data can be read from the sensor.

After powering up, the sensor switches automatically between CAN protocols and USB connectivity. Please see the table below for the available options depending on the user's actions:

User action	Description
Sensor power-on	The sensor is now in CANopen streaming mode and continuously sends data over the CAN bus using the CANopen protocol.
User sends data to sensor using LpCan protocol	Sensor will switch to LpCan protocol communication mode and wait for command mode instructions over the CAN port. This is the way the LpmsControl application communicates with the sensor, if connected via the CAN bus.
User sends data to sensor	Sensor will switch to USB communication mode and wait for

over USB port using LpBus protocol	instructions over USB port. This is the way the LpmsControl application communicates with the sensor, if connected via the USB bus.

Host Device Communication

Communication through USB Interface

The USB interface of the LPMS-CU uses a serial-to-USB interface IC by the company FTDI. Drivers for this IC for all major operating systems can be downloaded from their website: http://www.ftdichip.com/FTDrivers.htm. Generally there are two options for communicating with the FTDI chip.

1. By downloading a virtual com port driver (VCP): This driver allows you to see the LPMS-CU as COM port in your operating system. All communication is done using standard COM port access procedures. The default connection baudrate is 912.6Kbit/s.
2. By accessing the FTDI chip directly using a DLL library: FTDI offers a convenient library that allows users to communicate with their USB interface ICs.

Communication through CAN Bus Interface

Users should be able to communicate with LPMS-CU using any standard CAN interface. The CAN message uses standard 11 bits identifier and 8 bytes of data. The default connection baudrate is 1Mbit/s.

Data Acquisition

For data acquisition, all the communications with the device needs to be according to the LpBUS or LpCAN protocol, which is introduced in section “*VIII. COMMUNICATION PROTOCOL*”.

Raw Sensor Data

The LPMS-CU IMU contains three MEMS sensors: A gyroscope, an accelerometer and a magnetometer. The raw data from all three of these sensors can be accessed by the host system based on the LpBUS protocol. This data can be used to check if the current acquisition range of the sensors is sufficient and if the different sensors generate correct output. Users can also implement their own sensor fusion algorithms using the raw sensor data values. Sensor range and data sampling speed can be set by sending commands to the firmware. Details will be explained later on in this manual at section of “*VIII. COMMUNICATION PROTOCOL*”.

The LPMS-CU is calibrated by default, but it might be necessary to recalibrate the sensors if the measurement environment changes (e.g. different ambient electromagnetic field, strong temperature changes). Please refer to the following sections for a detailed introduction of sensor calibration methods.

Orientation Data

The LPMS-CU has two orientation output formats: quaternion and Euler angle. As the Euler angle representation of orientation is subject to the Gimbal lock, we strongly recommend users to use the quaternion representation for the orientation calculation where possible.

Filter Settings

Data from the three MEMS sensors is combined using an extended complementary Kalman filter (LP-Filter) to calculate the orientation data (orientation quaternion and Euler angles). To make the filter operate correctly, its measurement parameters need to be set in an appropriate way.

Filter Modes

First, the mode of the filter needs to be selected, which can be set by LpmsControl software or firmware commands. The following filter modes are available:

Filter mode	Description
Gyroscope only	Only the data from the gyroscope is used to calculate the orientation data output from the sensor. In this mode the orientation data can be calculated very quickly and with little noise. However, a strong drift of the acquired values can occur due to the inherent bias problem of gyroscope. This mode should therefore be only used for cases in which a frequent reset of the zero-angle position is allowed.
Gyroscope + accelerometer	The orientation data that is calculated from the gyroscope is corrected by the accelerometer data based on quaternion representation. The accelerometer acquires accurate information about the roll and pitch orientation regarded with the earth gravity vector. The result of the correction is therefore orientation information that has very little error on the roll and pitch axis, the yaw axis however is still affected by the drift of the un-corrected gyroscope data. This mode might be significant useful when there is a strong magnetic interference that can hardly be efficiently calibrated exiting around the sensor and only the roll and pitch information is interest to the users.
Gyroscope + accelerometer +	Orientation data from the gyroscope that has been corrected by the accelerometer output as previously described is additionally modified by the

magnetometer (default mode)	direction of the earth magnetic field. This results in accurate orientation information for all three axes. This mode delivers good speed and accuracy for roll, pitch and yaw. In this mode, (un-calibrated) distortions of the earth magnetic will affect the accuracy of the orientation measurement.
Accelerometer + magnetometer (Euler only)	Orientation is directly calculated by the combination of the data from accelerometer and magnetometer using Euler representation. Therefore it has the singularity problem at certain orientations. Based on the information of gravity in the vertical frame and the geomagnetic field vector in horizontal frame, the roll, pitch and yaw angle can be achieved based on the readings from accelerometer and magnetometer. This mode is suitable for the application of small motion and limited magnetic distortion.
Gyroscope + accelerometer (Euler only)	The orientation data that is calculated from the gyroscope is corrected by the accelerometer data based on Euler representation. Therefore it has the singularity problem at certain orientations. The accelerometer acquires accurate information about the roll and pitch orientation regarded with the earth gravity vector. The result of the correction is therefore orientation information that has very little error on the roll and pitch axis, the yaw axis however is still affected by the drift of the un-corrected gyroscope data. This mode might be significant useful when there is a strong magnetic interference that can hardly be efficiently calibrated exiting around the sensor and only the roll and pitch information is interest to the users.

Magnetometer Correction Setting

The amount by which the magnetometer corrects the orientation output of the sensor can be controlled by the magnetic correction settings. The following options are selectable through LpmsControl or directly through the firmware commands.

Parameter presets	Description
Dynamic (default)	The value "Dynamic" means the magnetic correction inside the filter is performed dynamically together with the acceleration data according to the variance of magnetic interference. This parameter set is suitable for the situation when the magnetic interference keeps changing.
Weak	The value "weak" means the magnetic correction inside the filter has little impact on the orientation output. Sensor orientation is

	calculated mainly from the acceleration / gyroscope data. This parameter set is suitable for situations when strong magnetic interference that cannot be compensated through calibration appear regularly.
Medium	With the “medium” correction setting the impact of the magnetometer readings is still relatively weak, but stronger than in “weak” mode. This mode should be suitable for environments with occasional irregular field distortions.
Strong	In this mode the magnetometer readings have a strong direct impact on the orientation output. It can be used in environments with a calibrate-able constant field distortion or in “clean” fields (outside buildings with no metal parts or strong power sources in the vicinity of the sensor). Yaw orientation measurement in world coordinates will be most accurate in this mode.

Acceleration Compensation Setting

The amount by which the accelerometer corrects the orientation output of the sensor can be controlled by both linear acceleration and centripetal acceleration settings. The following options are selectable through LpmsControl or directly through the firmware commands.

Linear Acceleration Correction Settings

Parameter presets	Description
Off	There is no linear acceleration compensation for the sensor fusion in this mode. This parameter set is suitable for situations when there is no linear acceleration appears.
Weak	The value “weak” means the linear acceleration correction inside the filter has little dynamic impact on the orientation output. This parameter set is suitable for situations when linear acceleration appears regularly and slightly.
Strong (default)	The value “Strong” means the linear acceleration correction inside the filter has strong dynamic impact on the orientation output. This parameter set is suitable for situations when linear acceleration appears regularly and strongly.

Rotational Acceleration Correction Settings

Parameter presets	Description
Disable	There is no rotational acceleration compensation for the sensor fusion in this mode.
Enable (default)	There is dynamic rotational acceleration compensation for the sensor fusion in this mode.

Gyroscope Threshold

The input from the gyroscope can be thresholded so that the sensor orientation data is only updated when the sensor is moved. This threshold is automatically determined during gyroscope calibration.

Parameter preset	Description
Enable	Switches gyroscope threshold on.
Disable (default)	Switches gyroscope threshold off.

Gyroscope Auto-calibration Function

The selection of the following parameter values allows the users to enable or disable the gyroscope auto calibration function. In auto calibration mode the filter is automatically detects if the sensor is moving or not. If the sensor stays still for a certain time, the currently sampled gyroscope data will be used to re-calculate the gyroscope offset. This function is significant useful when the user is using the "Gyroscope only" filter mode, and most the time of the system stays still. Using this function will reduce the drift problem of the gyroscope.

Parameter preset	Description
Enable	Switch gyroscope auto-calibration on.
Disable (default)	Switch gyroscope auto-calibration off.

Low Pass Filter Setting

The selection of the following parameter values allows the users to further implement a simple low pass filter for smoothing the output data after the sensor fusion algorithm. The low pass filter is based on the following formula: $X_i = (1-a)*X_{i-1} + a*U_i$, where a is the coefficient listed in the following table, U is the input.

Parameter preset	Description

Off (default)	No filter implemented.
0.1	$a = 0.1$
0.05	$a = 0.05$
0.01	$a = 0.01$
0.005	$a = 0.005$
0.001	$a = 0.001$

Trade-offs and Limitations

Although we have put (and still do) a lot of effort into the design of the LPMS-CU, there are a few limitations of the sensor that need to be taken into account when using the device. The accuracy of the sensor is limited by the electronic noise level of the MEMS sensors used in the LPMS-CU. Although the sensor data acquisition speeds for gyroscope, accelerometer and magnetometer are more than 500Hz, but the data output frequency of the whole system is limited to a certain frequency (up to 300Hz). The parameters of the filter that fuses the data from the gyroscope, magnetometer and accelerometer need to be adjusted well, in order to achieve measurements with maximum accuracy. Furthermore, in case the sensor is used in changing environments, the sensor occasionally might need to be re-calibrated. The greatest drawback of the measurement principle of the sensor certainly is its affectability by a deformed earth magnetic field (in the vicinity of hard / soft iron, electric motors etc.). In such situations the use of the filter mode and parameters of the filter must be well considered. In case of LPMS-CU, battery run-times should be taken into account when considering the sensor for a new application. Furthermore, the wireless Bluetooth connection puts a limit on the maximum range and the maximum data update frequency.

Calibration Methods

Gyroscope Calibration

When the sensor is resting the output data of the gyroscope should be around zero. The raw data from the gyroscope sensor has a constant bias of a certain value. To determine this value please follow the following calibration procedure:

Step	Description
1	If it is not already switched on, power up the LPMS-CU device.
2	Put the sensor in a resting (non-moving) position.
3	Connect to the sensor.
4	Trigger the gyroscope calibration procedure either through a firmware command or using the LpmsControl software.

5	The gyroscope calibration will take around 5s. After that the gyroscope is calibrated, normal operation can be resumed.

Additionally to the gyroscope bias, the gyroscope threshold value will be adjusted during this calibration procedure. By default the use of the gyroscope threshold is disabled. It can be enabled by sending a firmware command or using the LpmsControl software.

Magnetometer Calibration

During the magnetometer calibration procedure several parameters are to be determined: magnetometer bias and gain on the X, Y and Z-axis; length and direction of the geomagnetic field vector. In most environments the earth magnetic field is influenced by electromagnetic noise sources such as power lines, metal etc. As a result the magnetic field becomes de-centered and deformed. During the magnetometer calibration the amount of de-centering and deformed as well as the average length of the magnetic field vector is calculated. These parameters are tuned automatically using the calibration procedures in the LpmsControl software:

Step	Description
1	If it is not already switched on, power up the LPMS-CU device.
2	If it is not already connected, connect to the sensor.
3	Start the magnetometer calibration using the LpmsControl software.
4	Rotate the sensor around its yaw axis for 2~3 rotations.
5	Rotate the sensor around its pitch axis for 2~3 rotations.
6	Rotate the sensor around its roll axis for 2~3 rotations.
7	Rotate the sensor randomly to acquire data as much as possible from different directions.
8	The calibration procedure finished automatically after 30 seconds. After that the magnetometer has been calibrated.

Accelerometer Calibration

The misalignment of the accelerometer relative to the casing of the LPMS-CU device is expressed by the so called misalignment matrix. Using the LpmsControl software this misalignment matrix can be calibrated by the user. In the mean time, the offsets of the accelerometer can be also evaluated. Whereas the usage of the LpmsControl software is explained in more detail in the "*LpmsControl Software Operation*" section, the calibration procedure consists of the following steps:

Step	Description
1	If it is not already switched on, power up the LPMS-CU device.
2	If it is not already connected, connect to the sensor.
3	Start the accelerometer misalignment calibration using the LpmsControl software. See "LpmsControl Software Operation" section.

4	Fix the sensor to a horizontal surface with the Z-axis pointing upwards.
5	Fix the sensor to a horizontal surface with the Z-axis pointing downwards.
6	Fix the sensor to a horizontal surface with the X-axis pointing upwards.
7	Fix the sensor to a horizontal surface with the X-axis pointing downwards.
8	Fix the sensor to a horizontal surface with the Y-axis pointing upwards.
9	Fix the sensor to a horizontal surface with the Y-axis pointing downwards.
10	After finishing the above procedures the accelerometer misalignment matrix and offset values will be re-calculated. This finishes the accelerometer calibration.

VIII. COMMUNICATION PROTOCOL

Establishing a Connection

There are two ways to communicate with LPMS-CU:

1. After powering up the sensor by default continuously streams measurement data over the CAN bus using the CANopen protocol. For a short explanation of our CANopen implementation please read further below. In this mode it is not necessary to send any commands to the sensor. The sensor will just send the measurement values non-stop over the CAN bus. Which values are sent, as well as the other measurement parameters can be set using the LpmControl application and then saved to the flash memory of the sensor. Please don't forget that the sensor needs to be powered down once after using LpmsControl to be returned into its default data streaming mode. Use this method, if you simply want to read data from the sensor.

2. More complex communication can be achieved with the LpBus or LpCAN protocol. This protocol allows the user not only to read data from the sensor, but also access the sensors parameter registers and settings. Use this method, if switching between filter modes, ranges etc. is required for your application.

In this manual the LpBus and LpCAN protocol will be explained first. Please skip the next two sections, if you are just interested in reading sensor data via the CAN bus.

LpBUS Protocol

Basic Command Introduction

The communication packet has two basic command types, GET and SET, that are sent from a host (PC, mobile data logging unit etc.) to a client (LPMS-CU device). Later in this manual we will show a description of all supported commands to the sensor, their type, contained data etc.

GET Commands

Data from the client is read using GET requests. A GET request usually contains no data. The answer from the client to a GET request contains the requested data.

SET Commands

Data registers of the client are written using SET requests. A SET command from the host contains the data to be set. The answer from the client is an ACK command feedback for a successful write, or NACK command feedback for a failure to set the register occurred.

Packet Format

All communication with the USB interface of LPMS-CU works with a common protocol called LpBUS. The protocol is based on the industry standard MODBUS that we slightly adapted to be most suitable for our purpose. Each packet sent during the communication is based on this protocol, which is described in the following table:

Byte no.	Name	Description
0	Packet start (3Ah)	Mark of the beginning of a data packet.
1	OpenMAT ID byte 1	Contains the low byte of the OpenMAT ID of the sensor to be communicated with. The default value of this ID is 1. The host sends out a GET / SET request to a specific LPMS-CU sensor by using this ID, and the client answers to request also with the same ID. This ID can be adjusted by sending a SET command to the sensor firmware.
2	OpenMAT ID byte 2	High byte of the OpenMAT ID of the sensor.
3	Command no. byte 1	Contains the low byte of the command to be performed by the data transmission.
4	Command no. byte 2	High byte of the command number.
5	Packet data length byte 1	Contains the low byte of the packet data length to be transmitted in the packet data field.
6	Packet data length byte 2	High byte of the data length to be transmitted.
x	Packet data (***n*** bytes)	If data length ***n*** not equal to zero, ***x*** = 6+1, 6+2…6+***n***. Otherwise ***x*** = none. This data field contains the packet data to be transferred with the transmission if the data length not equals to zero, otherwise the data field is empty.
7+*n*	LRC byte 1	The low byte of LRC check-sum. To ensure the integrity of the transmitted data the LRC check-sum is used. It is calculated in the following way:

		LRC = sum(packet byte no. 1 to no. ***x***) The calculated LRC is usually compared with the LRC transmitted from the remote device. If the two LRCs are not equal, and error is reported.
8+n	LRC byte 2	High byte of LRC check-sum.
9+n	Termination byte 1	0Dh
10+n	Termination byte 2	0Ah

Data Format in a Packet Data Field

Generally data is sent in little-endian format, low order byte first, high order byte last. Data in the data fields of a packet can be encoded in several ways, depending on the type of information to be transmitted. In the following we list the most common data types. Other command-specific data types are explained in the command reference.

32-bit integer values (LENGTH = 4 bytes)

Transmission order	**0**	**1**	**2**	**3**
Integer word, byte number	0 (LSB)	1	2	3 (MSB)

32-bit integer 3-component vector (LENGTH = 12 bytes)

Transmission order	**0**	**1**	**2**	**3**
Vector component 1, byte number	0 (LSB)	1	2	3 (MSB)
Transmission order	**4**	**5**	**6**	**7**
Vector component 2, byte number	0 (LSB)	1	2	3 (MSB)
Transmission order	**8**	**9**	**10**	**11**
Vector component 3, byte number	0 (LSB)	1	2	3 (MSB)

32-bit float value encoded as integer (LENGTH = 4 bytes)

Transmission order	**0**	**1**	**2**	**3**
Integer-encoded float, byte number	0 (LSB)	1	2	3 (MSB)

32-bit float 3-component vector (LENGTH = 12 byte)

Transmission order	**0**	**1**	**2**	**3**
Vector component 1, byte number	0 (LSB)	1	2	3 (MSB)
Transmission order	**4**	**5**	**6**	**7**
Vector component 2, byte number	0 (LSB)	1	2	3 (MSB)
Transmission order	**8**	**9**	**10**	**11**

Vector component 3, byte number	0 (LSB)	1	2	3 (MSB)

LpCAN Protocol

To exchange data with LPMS-CU through the CAN bus interface, the serial LpBUS protocol is split into CAN bus messages. We call this CAN bus wrapper for the LpBUS protocol: LpCAN.

A regular LpCAN message is structured as shown below:

11-bit CAN identifier	The CAN identifier of a CAN message. This identifier is set to the value **514h** for all LpCAN transmissions.
8 data bytes	Contains the actual data to be transmitted in a CAN message.

An example packet with 4 data bytes wrapping from LpBUS to LpCAN results in the following CAN messages:

CAN Message #1:

Byte no.	Name	Description
0	Packet start (3Ah)	Mark of the beginning of a data packet.
1	OpenMAT ID byte 1	Contains the low byte of the OpenMAT ID of the sensor to be communicated with. The default value of this ID is 1. The host sends out a GET / SET request to a specific LPMS-CU sensor by using this ID, and the client answers to request also with the same ID. This ID can be adjusted by sending a SET command to the sensor firmware.
2	OpenMAT ID byte 2	High byte of the OpenMAT ID of the sensor.
3	Command no. byte 1	Contains the low byte of the command to be performed by the data transmission.
4	Command no. byte 2	High byte of the command number.
5	Packet data length byte 1	Contains the low byte of the packet data length to be transmitted in the packet data field (in this example 4)
6	Packet data length byte 2	High byte of the data length to be transmitted (in this example 0)
7	Packet data	Packet data byte 0

CAN Message #2:

Byte no.	Name	Description
0	Packet data	Packet data byte 1
1	Packet data	Packet data byte 2
2	Packet data	Packet data byte 3
3	LRC byte 1	The low byte of LRC check-sum.
4	LRC byte 2	High byte of LRC check-sum.
5	Termination byte 1	0Dh
6	Termination byte 2	0Ah
7	Not used	0

The number of messages needed to contain the data depends on the length of the data to be transmitted. The last message of a set is truncated to be just long enough to transport all of the remaining wrapped LpBUS data (in the example 7 bytes). The unused bytes of a message are filled up with 0.

CANOpen Protocol

After the sensor is powered-up, it will by default start streaming sensor data in CANopen format. Our CANopen implementation consists of 4 TPDO (Transmission Data Process Object) messages and a heartbeat message that are transmitted over the CAN bus. Sensor data can be assigned to specific messages using the LpmsControl application. The frequency of the CANopen heartbeat message is adjustable between 0.1 Hz and 2 Hz. For details on how to adjust parameters using LpmsControl, please see the next chapter.

CANOpen data is continuously sent from the sensor to the host with the streaming frequency selected in the LpmsControl application at the selected baudrate. The data to be transmitted can be selected to adjust the bus bandwidth used by the LPMS system. All transmitted values are in IEEE754 32-bit integer encoded floating point format. Please see an overview of the CANopen messages below:

Message ID	Description
180h + IMU ID	CANopen TPDO 1. Freely assignable. Data in IEEE754 format.
280h + IMU ID	CANopen TPDO 2. Freely assignable. Data in IEEE754 format.
380h + IMU ID	CANopen TPDO 3. Freely assignable. Data in IEEE754 format.
480h + IMU ID	CANopen TPDO 4. Freely assignable. Data in IEEE754 format.
700h + IMU ID	If the sensor is in operational state, this message contains one

	byte with value 5h. If the sensor is stopped due to an error, the value is 4h.

Protocol Commands List

If a user connects to LPMS-CU either using the LpBus protocol or the LpCAN protocol, he can access the sensor functionality using the commands in the list below.

Acknowledged and Not-acknowledged Identifier

Command No. (decimal values)	**Command description**
0	REPLY_ACK (acknowledged). Confirms a successful SET command.
1	REPLY_NACK (not-acknowledged) Reports an error during processing a SET command.

Firmware Update and In-Application-Programmer Upload Command

2	Start the firmware update process. IMPORTANT: By not correctly uploading a firmware file the sensor might become in-operable. In normal cases please use the LpmsControl software to upload new firmware. Also please only use firmware packages that have been authorized by LP-RESEARCH. **Packet data:** Firmware data **Data format** Firmware binary file separated into 256 byte chunks for each update packet. **Macro name:** UPDATE_FIRMWARE **Response:** ACK (success) or NACK (error) for each transmitted packet.
3	“**RESERVED**” This command is reserved by LP-RESEARCH. Start the in-application programmer (IAP) update process. **Packet data:** IAP data **Data format** IAP binary file separated into 256 byte chunks for each update packet. **Macro name:** UPDATE_IAP **Response:** ACK (success) or NACK (error) for each

	transmitted packet.

Configuration and Status Command

4	Get the current value of the configuration register of the sensor. The configuration word can ONLY be read. The different parameters are set by their respective SET commands. E.g. SET_TRANSMIT_DATA for defining which data is transmitted from the sensor. **Packet data:** Configuration word. Each bit represents the state of one configuration parameter. **Return format:** 32-bit integer

Bit	Reported State / Parameter
0 - 2	Stream frequency setting (see SET_STREAM_FREQ)
3 - 8	Reserved
9	Pressure data transmission enabled
10	Magnetometer data transmission enabled
11	Accelerometer data transmission enabled
12	Gyroscope data transmission enabled
13	Temperature output enabled
14	Heave motion output enabled
15	Reserved
16	Angular velocity output enabled
17	Euler angle data transmission enabled
18	Quaternion orientation output enabled
19	Output enabled
20	Dynamic magnetometer correction enabled
21	Linear acceleration output enabled
22	Reserved
23	Gyroscope threshold enabled
24	Magnetometer compensation enabled
25	Accelerometer compensation enabled
26	Reserved
27	Reserved
28	Reserved
29	Reserved
30	Gyroscope auto-calibration enabled

<table>
<tr><td></td><td>
<table><tr><td>31</td><td>Reserved</td></tr></table>
Macro name: GET_CONFIG
</td></tr>
<tr><td>5</td><td>
Get the current value of the status register of the LPMS-CU device. The status word can ONLY be read.
Packet data: Status indicator. Each bit represents the state of one status parameter.
Return format: 32-bit integer
<table>
<tr><th>Bit</th><th>Indicated state</th></tr>
<tr><td>0</td><td>COMMAND mode enabled</td></tr>
<tr><td>1</td><td>STREAM mode enabled</td></tr>
<tr><td>2</td><td>SLEEP mode enabled</td></tr>
<tr><td>3</td><td>Gyroscope calibration on</td></tr>
<tr><td>4</td><td>Reserved</td></tr>
<tr><td>5</td><td>Gyroscope initialization failed</td></tr>
<tr><td>6</td><td>Accelerometer initialization failed</td></tr>
<tr><td>7</td><td>Magnetometer initialization failed</td></tr>
<tr><td>8</td><td>Pressure sensor initialization failed</td></tr>
<tr><td>9</td><td>Gyroscope unresponsive</td></tr>
<tr><td>10</td><td>Accelerometer unresponsive</td></tr>
<tr><td>11</td><td>Magnetometer unresponsive</td></tr>
<tr><td>12</td><td>Flash write failed</td></tr>
<tr><td>13</td><td>Reserved</td></tr>
<tr><td>14</td><td>Set broadcast frequency failed</td></tr>
<tr><td>15-31</td><td>reserved</td></tr>
</table>
Macro name: GET_STATUS
</td></tr>
</table>

Mode Switching Command

<table>
<tr><td>6</td><td>
Switch to streaming mode. In this mode data is continuously streamed from the sensor, and all other commands cannot be performed until the sensor receives the GOTO_COMMAND_MODE command.
Packet data: none
Macro name: GOTO_STREAM_MODE
Response: ACK (success) or NACK (error)
</td></tr>
<tr><td>7</td><td>Switch to command mode. In command mode the user can issue commands to the</td></tr>
</table>

	firmware to perform calibration, set parameters etc. **Packet data:** none **Macro name:** GOTO_COMMAND_MODE **Response:** ACK (success) or NACK (error)
8	Switch to sleep mode. The purpose of the sleep mode is to reduce the power consumption of the sensor. Once in sleep mode, no commands can be issued to the sensor until it is woken up by switching back into command mode or streaming mode. **Packet data:** none **Macro name:** GOTO_SLEEP_MODE **Response:** ACK (success) or NACK (error)

Data Transmission Command

9	Get the latest set of sensor data. The format of the sensor data depends on the transmission settings (SET_TRANSMIT_DATA). The currently set format can be retrieved with the sensor configuration word. IMPORTANT: In the current version of the firmware calibrated accelerometer data as well as calibrated magnetometer data will always be transmitted. As these values are necessary for the calibration of the sensor, they can at the moment not be switched off. This format is also used in streaming mode to continuously send data from the sensor to the host. **Packet data:** Sensor data. The data always has the same order. Depending on the enabled transmission data, chunks are inserted or left out. **Return format:** Raw sensor data chunk

Chunk number	Data type	Sensor data
1	Float	Timestamp (ms)
2	Float 3-vector	Calibrated gyroscope data [deg/s]
3	Float 3-vector	Calibrated accelerometer data [m/s^2]
4	Float 3-vector	Calibrated magnetometer data [μT]
5	Float 3-vector	Angular velocity [deg/s]
6	Float 4-vector	Orientation quaternion [normalized]

	<table><tr><td>7</td><td>Float 3-vector</td><td>Euler angle data [deg.]</td></tr><tr><td>8</td><td>Float 3-vector</td><td>Linear acceleration data [m/s^2]</td></tr><tr><td>9</td><td>Float</td><td>Barometric pressure [mPa]</td></tr><tr><td>10</td><td>Float</td><td>Heave motion [m] (if enabled)</td></tr></table> **Macro name:** GET_SENSOR_DATA
10	Set the data that is transmitted from the sensor in streaming mode or when retrieving data through the GET_SENSOR_DATA command. **Packet data:** Data selection indicator **Data format:** 32-bit integer. The flags to switch data chunks on (set the bit to 1) and off (set the bit to 0) are the same as in the configuration word (see SET_CONFIG). <table><tr><th>Bit</th><th>Reported State / Parameter</th></tr><tr><td>**9**</td><td>Pressure data transmission enabled</td></tr><tr><td>**10**</td><td>Magnetometer data transmission enabled</td></tr><tr><td>**11**</td><td>Accelerometer data transmission enabled</td></tr><tr><td>**12**</td><td>Gyroscope data transmission enabled</td></tr><tr><td>**13**</td><td>Temperature output enabled</td></tr><tr><td>**14**</td><td>Heave motion output enabled</td></tr><tr><td>**16**</td><td>Angular velocity output enabled</td></tr><tr><td>**17**</td><td>Euler angle data transmission enabled</td></tr><tr><td>**18**</td><td>Quaternion orientation output enabled</td></tr><tr><td>**19**</td><td>Altitude output enabled</td></tr><tr><td>**21**</td><td>Linear acceleration output enabled</td></tr></table>**Macro name:** SET_TRANSMIT_DATA **Response:** ACK (success) or NACK (error) **Default value:** Gyroscope, accelerometer, magnetometer and quaternion data.
11	Set the timing in which streaming data is sent to the host. Please note that high frequencies might be not practically applicable due to limitations of the communication interface. Check the current baudrate before setting this parameter. **Packet data:** Update frequency identifier **Format:** 32-bit integer <table><tr><td>Frequency (Hz)</td><td>Identifier</td></tr></table>

<table>
<tr><td></td><td><table><tr><td>5</td><td>5</td></tr><tr><td>10</td><td>10</td></tr><tr><td>30</td><td>30</td></tr><tr><td>50</td><td>50</td></tr><tr><td>100</td><td>100</td></tr><tr><td>200</td><td>200</td></tr><tr><td>300</td><td>300</td></tr><tr><td>500</td><td>500</td></tr></table>
Macro name: SET_STREAM_FREQ
Response: ACK (success) or NACK (error)
Default value: 100 Hz</td></tr>
<tr><td>12</td><td>Get the current roll angle in radians.
Packet data: Roll angle
Return format: 32-bit integer coded float value.
Macro name: GET_ROLL</td></tr>
<tr><td>13</td><td>Get the current pitch angle in radians.
Packet data: Pitch angle
Return format: 32-bit integer coded float value.
Macro name: GET_PITCH</td></tr>
<tr><td>14</td><td>Get the current yaw angle in radians.
Packet data: Yaw angle
Return format: 32-bit integer coded float value.
Macro name: GET_YAW</td></tr>
</table>

Register Value Save and Reset Command

15	Write the currently set parameters to flash memory. **Packet data:** None **Macro name:** WRITE_REGISTERS **Response:** ACK (success) or NACK (error)
16	Reset the LPMS parameters to factory default values. Please note that upon issuing this command your currently set parameters will be erased. **Packet data:** none **Macro name:** RESTORE_FACTORY_VALUE

	Response: ACK (success) or NACK (error)

Reference Setting and Offset Reset Command

17	Set the accelerometer and magnetometer reference vectors. **Packet data:** None **Macro name:** RESET_REFERENCE **Response:** ACK (success) or NACK (error)
18	Set the orientation offset (the value that is subtracted from the acquired orientation value after a measurement) to the currently measured orientation. This effectively resets the zero orientation of the sensor to the current orientation. **Packet data:** none **Macro name:** SET_OFFSET **Response:** ACK (success) or NACK (error)

Self-Test Command

19	Initiate the self-test. During the self test the sensor automatically rotates about the three room axes. To simulate realistic circumstances an artificial offset is applied to the magnetometer and the gyroscope values. **Packet data:** none **Macro name:** SELF_TEST **Response:** ACK (success) or NACK (error)

IMU ID Setting Command

20	Set the OpenMAT ID of the LPMS-CU. **Packet data:** OpenMAT ID **Data format:** 32-bit integer **Macro name:** SET_IMU_ID **Response:** ACK (success) or NACK (error) **Default value:** 1
21	Get the ID (OpenMAT ID) of the device **Packet data:** The ID of the IMU device **Return format:** 32-bit integer **Macro name:** GET_IMU_ID

Gyroscope Settings Command

<table>
<tr><td>22</td><td>Start the calibration procedure of the gyroscope sensor. Details of the gyroscope calibration procedure are described in the Operation – Calibration Methods section of this manual. The calibration takes about 5s.
Packet data: none
Macro name: START_GYR_CALIBRATION
Response: ACK (success) or NACK (error)</td></tr>
<tr><td>23</td><td>Enable or disable auto-calibration of the gyroscope.
Packet data: Gyroscope auto-calibration enable / disable identifier
Format: 32-bit integer<table><tr><td>State</td><td>Value</td></tr><tr><td>Disable</td><td>0x00000000</td></tr><tr><td>Enable</td><td>0x00000001</td></tr></table>Macro name: ENABLE_GYR_AUTOCAL
Response: ACK (success) or NACK (error)
Default value: Disable</td></tr>
<tr><td>24</td><td>Enable or disable gyroscope threshold.
Packet data: Gyroscope threshold enable / disable identifier
Format: 32-bit integer<table><tr><td>State</td><td>Value</td></tr><tr><td>Disable</td><td>0x00000000</td></tr><tr><td>Enable</td><td>0x00000001</td></tr></table>Macro name: ENABLE_GYR_THRES
Response: ACK (success) or NACK (error)
Default value: Disable</td></tr>
<tr><td>25</td><td>Set the current range of the gyroscope.
Packet data: Gyroscope range identifier
Format: 32-bit integer<table><tr><td>Range (deg/s)</td><td>Identifier</td></tr><tr><td>250</td><td>250</td></tr><tr><td>500</td><td>500</td></tr><tr><td>2000</td><td>2000</td></tr></table>Macro name: SET_GYR_RANGE</td></tr>
</table>

	Response: ACK (success) or NACK (error) **Default value:** 2000 deg/s
26	Get current gyroscope range. **Packet data:** Gyroscope range indicator **Return format:** 32-bit integer **Macro name:** GET_GYR_RANGE

Accelerometer Settings Command

27	Set the accelerometer bias. **Packet data:** Accelerometer bias (X, Y, Z-axis) **Format:** 32-bit integer encoded float 3-component vector **Macro name:** SET_ACC_BIAS **Response:** ACK (success) or NACK (error) **Default value:** (0.0, 0.0, 0.0)
28	Get the current accelerometer bias vector. **Packet data:** Accelerometer bias vector **Return format:** 32-bit integer encoded float 3-component vector **Macro name:** GET_ACC_BIAS
29	Set the accelerometer alignment matrix. **Packet data:** Alignment matrix **Format:** 32-bit integer encoded float 3 x 3 matrix **Macro name:** SET_ACC_ALIG **Response:** ACK (success) or NACK (error) **Default value:** 3x3 Identity matrix
30	Get the current accelerometer alignment matrix. **Packet data:** Accelerometer alignment matrix **Return format:** 32-bit integer encoded float 3 x 3 matrix **Macro name:** GET_ACC_ALIG
31	Set the current range of the accelerometer. **Packet data:** Accelerometer range identifier **Format:** 32-bit integer

<table>
<tr><td></td><td>
<table>
<tr><td>Range (g: 1 gravity)</td><td>Identifier</td></tr>
<tr><td>2g</td><td>2</td></tr>
<tr><td>4g</td><td>4</td></tr>
<tr><td>8g</td><td>8</td></tr>
<tr><td>16g</td><td>16</td></tr>
</table>
Macro name: SET_ACC_RANGE
Response: ACK (success) or NACK (error)
Default value: 2g
</td></tr>
<tr><td>32</td><td>Get current accelerometer range.
Packet data: Accelerometer range indicator
Return format: 32-bit integer
Macro name: GET_ACC_RANGE
</td></tr>
</table>

Magnetometer Settings Command

<table>
<tr><td>33</td><td>Set the current range of the magnetometer.
Packet data: Magnetometer range identifier
Format: 32-bit integer
<table>
<tr><td>Range</td><td>Identifier</td></tr>
<tr><td>130 uT</td><td>130</td></tr>
<tr><td>190 uT</td><td>190</td></tr>
<tr><td>250 uT</td><td>250</td></tr>
<tr><td>400 uT</td><td>400</td></tr>
<tr><td>470 uT</td><td>470</td></tr>
<tr><td>560 uT</td><td>560</td></tr>
<tr><td>810 uT</td><td>810</td></tr>
</table>
Macro name: SET_MAG_RANGE
Response: ACK (success) or NACK (error)
Default value: 250 uT
</td></tr>
<tr><td>34</td><td>Get current magnetometer range.
Packet data: Magnetometer range indicator (same as above)
Return format: 32-bit integer
Macro name: GET_MAG_RANGE
</td></tr>
<tr><td>35</td><td>Set the current hard iron offset vector.
Packet data: Hard iron offset values in uT
</td></tr>
</table>

	Format: 32-bit integer encoded 3-element float vector **Macro name:** SET_HARD_IRON_OFFSET **Response:** ACK (success) or NACK (error) **Default value:** (0.0, 0.0, 0.0)
36	Get current hard iron offset vector. **Packet data:** Hard iron offset values in uT **Return format:** 32-bit integer encoded 3-element float vector **Macro name:** GET_HARD_IRON_OFFSET
37	Set the current soft iron matrix. **Packet data:** Soft iron matrix values in uT **Format:** 32-bit integer encoded 9-element (3x3) float matrix **Macro name:** SET_SOFT_IRON_MATRIX **Response:** ACK (success) or NACK (error) **Default value:** (1, 0, 0) (0, 1, 0) (0, 0, 1)
38	Get the current soft iron matrix. **Packet data:** Soft iron matrix values in uT **Return format:** 32-bit integer encoded 9-element (3x3) float matrix **Macro name:** GET_SOFT_IRON_MATRIX
39	Set the current earth magnetic field strength estimate. **Packet data:** Field estimate value in uT **Format:** 32-bit integer encoded float **Macro name:** SET_FIELD_ESTIMATE **Response:** ACK (success) or NACK (error) **Default value:** 50 uT
40	Get the current earth magnetic field strength estimate. **Packet data:** Field estimate value in uT **Return format:** 32-bit integer encoded float **Macro name:** GET_FIELD_ESTIMATE

Filter Settings Command

<table>
<tr><td>41</td><td>Set the sensor filter mode.
Packet data: Mode identifier
Format: 32-bit integer
<table>
<tr><td>Mode</td><td>Value</td></tr>
<tr><td>Gyroscope only</td><td>0x00000000</td></tr>
<tr><td>Accelerometer + gyroscope</td><td>0x00000001</td></tr>
<tr><td>Accelerometer + gyroscope + magnetometer</td><td>0x00000002</td></tr>
<tr><td>Accelerometer + Magnetometer (Euler angle based filtering)</td><td>0x00000003</td></tr>
<tr><td>Accelerometer + Gyroscope (Euler angle-based filtering)</td><td>0x00000004</td></tr>
</table>
Macro name: SET_FILTER_MODE
Response: ACK (success) or NACK (error)
Default value: Accelerometer + gyroscope + magnetometer</td></tr>
<tr><td>42</td><td>Get the currently selected filter mode.
Packet data: Filter mode identifier
Return format: 32-bit integer
<table>
<tr><td>Mode</td><td>Value</td></tr>
<tr><td>Gyroscope only</td><td>0x00000000</td></tr>
<tr><td>Accelerometer + gyroscope</td><td>0x00000001</td></tr>
<tr><td>Accelerometer + gyroscope + magnetometer</td><td>0x00000002</td></tr>
</table>
Macro name: GET_FILTER_MODE</td></tr>
<tr><td>43</td><td>Set one of the filter parameter presets.
Packet data: Magnetometer correction strength preset identifier
Format: 32-bit integer
<table>
<tr><td>Preset</td><td>Value</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td>Dynamic</td><td>0x00000000</td></tr>
<tr><td>Strong</td><td>0x00000001</td></tr>
<tr><td>Medium</td><td>0x00000002</td></tr>
<tr><td>Weak</td><td>0x00000003</td></tr>
</table>
Macro name: SET_FILTER_PRESET
Response: ACK (success) or NACK (error)
Default value: Dynamic
</td></tr>
<tr><td>44</td><td>
Get the currently magnetometer correction strength preset
Packet data: Magnetometer correction strength preset identifier
Return format: 32-bit integer
<table>
<tr><td>Correction strength</td><td>Value</td></tr>
<tr><td>Dynamic</td><td>0x00000000</td></tr>
<tr><td>Strong</td><td>0x00000001</td></tr>
<tr><td>Medium</td><td>0x00000002</td></tr>
<tr><td>Weak</td><td>0x00000003</td></tr>
</table>
Macro name: GET_FILTER_PRESET
</td></tr>
</table>

CAN Bus Settings Command

<table>
<tr><td>45</td><td>Set CAN stream format. This command has been deprecated.</td></tr>
<tr><td>46</td><td>
Set the CAN baudrate
Packet data: CAN communication baudrate
Format: 32-bit integer
<table>
<tr><td>Correction strength</td><td>Value</td></tr>
<tr><td>10Kbit/s</td><td>0x00000000</td></tr>
<tr><td>20Kbit/s</td><td>0x00000008</td></tr>
<tr><td>50Kbit/s</td><td>0x00000010</td></tr>
<tr><td>125Kbit/s</td><td>0x00000018</td></tr>
<tr><td>250Kbit/s</td><td>0x00000020</td></tr>
<tr><td>500Kbit/s</td><td>0x00000028</td></tr>
<tr><td>800Kbit/s</td><td>0x00000030</td></tr>
<tr><td>1Mbit/s</td><td>0x00000038</td></tr>
</table>
Macro name: SET_CAN_BAUDRATE
Response: ACK (success) or NACK (error)
Default value: 1Mbit/s
</td></tr>
</table>

Additional Settings

47	Set CAN stream format. This command has been deprecated.
48	Set gyroscope alignment bias **Packet data:** Gyroscope alignment bias **Format:** Float 3-vector **Macro name:** SET_GYR_ALIGN_BIAS **Response:** ACK (success) or NACK (error) **Default value:** 1Mbit/s
49	Get gyroscope alignment bias **Packet data:** Gyroscope alignment bias **Return format:** Float 3-vector **Macro name:** GET_GYR_ALIGN_BIAS
50	Set gyroscope alignment matrix **Packet data:** Gyroscope alignment matrix **Format:** Float 3x3 matrix **Macro name:** SET_GYR_ALIGN_MATRIX **Response:** ACK (success) or NACK (error) **Default value:** (1, 0, 0) (0, 1, 0) (0, 0, 1)
51	Get gyroscope alignment matrix **Packet data:** Gyroscope alignment matrix **Return format:** Float 3x3 matrix **Macro name:** GET_GYR_ALIGN_MATRIX
52	Reserved
53	Reserved
54	Reserved
55	Reserved
56	Reserved
57	Reserved
58	Reserved
59	Reserved
60	Set raw data low-pass **Packet data:** Low pass strength (1.0 is weakest / disabled) **Format:** Float **Macro name:** SET_RAW_DATA_LP

	Response:	ACK (success) or NACK (error)
	Default value:	1.0
61	Get raw data low-pass	
	Packet data:	Low pass strength (1.0 is weakest / disabled)
	Return format:	Float
	Macro name:	GET_RAW_DATA_LP
62	Set CANopen mapping	
	Packet data:	CANopen mapping
	Format:	The mapping data consists of 8 integer words. Each of these words represents the assignment of half a CANopen transmission object / message (TPDO) to specific sensor data. In more detail:

Message name	**Position in configuration mapping message**
TPDO 1 (msg. 0x180, bytes 0-3)	0
TPDO 1 (msg. 0x180, bytes 5-7)	1
TPDO 2 (msg. 0x280, bytes 0-3)	2
TPDO 2 (msg. 0x280, bytes 5-7)	3
TPDO 3 (msg. 0x380, bytes 0-3)	4
TPDO 3 (msg. 0x380, bytes 5-7)	5
TPDO 4 (msg. 0x480, bytes 0-3)	6
TPDO 4 (msg. 0x480, bytes 5-7)	7

Assignments work according to the following table:

Sensor data	Assignment index
Angular velocity X	0
Angular velocity Y	1
Angular velocity Z	2
Euler angle X	3
Euler angle Y	4
Euler angle Z	5
Lin. acceleration X	6
Lin. acceleration Y	7

<table>
<tr><td></td><td>
<table>
<tr><td>Lin. acceleration Z</td><td>8</td></tr>
<tr><td>Magnetometer X</td><td>9</td></tr>
<tr><td>Magnetometer Y</td><td>10</td></tr>
<tr><td>Magnetometer Z</td><td>11</td></tr>
<tr><td>Quaternion W</td><td>12</td></tr>
<tr><td>Quaternion X</td><td>13</td></tr>
<tr><td>Quaternion Y</td><td>14</td></tr>
<tr><td>Quaternion Z</td><td>15</td></tr>
<tr><td>Accelerometer X</td><td>16</td></tr>
<tr><td>Accelerometer Y</td><td>17</td></tr>
<tr><td>Accelerometer Z</td><td>18</td></tr>
</table>
Macro name: SET_CAN_MAPPING
Response: ACK (success) or NACK (error)
Default value: 0x00000007 00000006 00000005 00000004 00000003 00000002 00000001 00000000
</td></tr>
<tr><td>63</td><td>Get CANopen mapping
Packet data: CANopen mapping
Return format: See command 62
Macro name: GET_CAN_MAPPING
</td></tr>
<tr><td>64</td><td>Set CANopen heartbeat frequency
Packet data: CANopen heartbeat frequency
Format: Integer. In detail:
<table>
<tr><td>Heartbeat frequency</td><td>Identifier</td></tr>
<tr><td>5Hz</td><td>0x00000000</td></tr>
<tr><td>1Hz</td><td>0x00000001</td></tr>
<tr><td>0.5Hz</td><td>0x00000002</td></tr>
<tr><td>0.2Hz</td><td>0x00000003</td></tr>
<tr><td>0.1Hz</td><td>0x00000004</td></tr>
</table>
Macro name: SET_CAN_HEARTBEAT
Response: ACK (success) or NACK (error)
Default value: 0x00000000
</td></tr>
<tr><td>65</td><td>Get CAN heartbeat
Packet data: CANopen heartbeat frequency
Return format: See command 64
Macro name: GET_CAN_HEARTBEAT
</td></tr>
</table>

66	Reset sensor data timestamp to 0 **Packet data:** none **Format:** none **Macro name:** RESET_TIMESTAMP **Response:** ACK (success) or NACK (error) **Default value:** none

Example Communication

In this section we will show a practical example of how a communication sequence could be structured. A similar sequence is also used in the LpmsControl software to poll data from the LPMS-CU. Our standard LpBUS protocol is used.

Request Sensor Configuration

GET request (HOST -> SENSOR)

Packet byte no.	Content	Meaning
0	3Ah	Packet start
1	01h	OpenMAT ID LSB (ID = 1)
2	00h	OpenMAT ID MSB
3	04h	Command no. LSB (4d = GET_CONFIG)
4	00h	Command no. MSB
5	00h	Data length LSB (GET command = no data)
6	00h	Data length MSB
7	05h	Check sum LSB
8	00h	Check sum MSB
9	0Dh	Packet end 1
10	0Ah	Packet end 2

Reply data (SENSOR -> HOST)

Packet byte no.	Content	Meaning
0	3Ah	Packet start
1	01h	OpenMAT LSB (ID = 1)
2	00h	OpenMAT MSB
3	04h	Command no. LSB (4d = GET_CONFIG)
4	00h	Command no. MSB
5	04h	Data length LSB (32-bit integer = 4 bytes)

6	00h	Data length MSB
7	xxh	Configuration data byte 1 (LSB)
8	xxh	Configuration data byte 2
9	xxh	Configuration data byte 3
10	xxh	Configuration data byte 4 (MSB)
11	xxh	Check sum LSB
12	xxh	Check sum MSB
13	0Dh	Packet end 1
14	0Ah	Packet end 2

Request Gyroscope Range

GET request (HOST -> SENSOR)

Packet byte no.	Content	Meaning
0	3Ah	Packet start
1	01h	OpenMAT ID LSB (ID = 1)
2	00h	OpenMAT ID MSB
3	1Ah	Command no. LSB (26d = GET_GYR_RANGE)
4	00h	Command no. MSB
5	00h	Data length LSB (GET command = no data)
6	00h	Data length MSB
7	1Bh	Check sum LSB
8	00h	Check sum MSB
9	0Dh	Packet end 1
10	0Ah	Packet end 2

Reply data (SENSOR -> HOST)

Packet byte no.	Content	Meaning
0	3Ah	Packet start
1	01h	OpenMAT ID LSB (ID = 1)
2	00h	OpenMAT ID MSB
3	1Ah	Command no. LSB (26d = GET_GYR_RANGE)
4	00h	Command no. MSB
5	04h	Data length LSB (32-bit integer = 4 bytes)
6	00h	Data length MSB
7	xxh	Range data byte 1 (LSB)

8	xxh	Range data byte 2
9	xxh	Range data byte 3
10	xxh	Range data byte 4 (MSB)
11	xxh	Check sum LSB
12	xxh	Check sum MSB
13	0Dh	Packet end 1
14	0Ah	Packet end 2

Set Accelerometer Range

SET request (HOST -> SENSOR)

Packet byte no.	Content	Meaning
0	3Ah	Packet start
1	01h	OpenMAT ID LSB (ID = 1)
2	00h	OpenMAT ID MSB
3	1Fh	Command no. LSB (31d = SET_ACC_RANGE)
4	00h	Command no. MSB
5	04h	Data length LSB (32-bit integer = 4 bytes)
6	00h	Data length MSB
7	08h	Range data byte 1 (Range indicator 8g = 8d)
8	00h	Range data byte 2
9	00h	Range data byte 3
10	00h	Range data byte 4
11	2Bh	Check sum LSB
12	00h	Check sum MSB
13	0Dh	Packet end 1
14	0Ah	Packet end 2

Reply data (SENSOR -> HOST)

Packet byte no.	Content	Meaning
0	3Ah	Packet start
1	01h	OpenMAT ID LSB (ID = 1)
2	00h	OpenMAT ID MSB
3	00h	Command no. LSB (0d = REPLY_ACK)
4	00h	Command no. MSB
5	00h	Data length LSB (ACK reply = no data)

6	00h	Data length MSB
11	01h	Check sum LSB
12	00h	Check sum MSB
13	0Dh	Packet end 1
14	0Ah	Packet end 2

Read Sensor Data

Get request (HOST -> SENSOR)

Packet byte no.	Content	Meaning
0	3Ah	Packet start
1	01h	OpenMAT ID LSB (ID = 1)
2	00h	OpenMAT MSB
3	09h	Command no. LSB (9d = GET_SENSOR_DATA)
4	00h	Command no. MSB
5	00h	Data length LSB (GET command = no data)
6	00h	Data length MSB
7	0Ah	Check sum LSB
8	00h	Check sum MSB
9	0Dh	Packet end 1
10	0Ah	Packet end 2

Reply data (SENSOR -> HOST)

In this example the selected transmission data is: Raw gyroscope, raw accelerometer, raw magnetometer and orientation quaternion.

Packet byte no.	Content	Meaning
0	3Ah	Packet start
1	01h	OpenMAT ID LSB (ID = 1)
2	00h	OpenMAT ID MSB
3	09h	Command no. LSB (9d = GET_SENSOR_DATA)
4	00h	Command no. MSB
5	34h	Data length LSB (56 bytes)
6	00h	Data length MSB
7-10	xxxxxxxxh	Timestamps
11-14	xxxxxxxxh	Gyroscope data x-axis
15-18	xxxxxxxxh	Gyroscope data y-axis

19-22	xxxxxxxxh	Gyroscope data z-axis
23-26	xxxxxxxxh	Accelerometer x-axis
27-30	xxxxxxxxh	Accelerometer y-axis
31-34	xxxxxxxxh	Accelerometer z-axis
35-38	xxxxxxxxh	Magnetometer x-axis
39-42	xxxxxxxxh	Magnetometer y-axis
43-46	xxxxxxxxh	Magnetometer z-axis
47-50	xxxxxxxxh	Orientation quaternion q0
51-54	xxxxxxxxh	Orientation quaternion q1
55-58	xxxxxxxxh	Orientation quaternion q2
59-62	xxxxxxxxh	Orientation quaternion q3
63	xxh	Check sum LSB
64	xxh	Check sum MSB
65	0Dh	Message end byte 1
66	0Ah	Message end byte 2

IX. OpenMAT

Overview

Introduction

OpenMAT is the software package delivered with a LPMS device. The package contains the basic hardware device drivers for the sensors, a C++ library to easily access the functionality of the IMUs and also a network interface (OpenMAT network) that allows applications to communicate with each other to exchange sensor information. OpenMAT consists of the following components:

1. **LpSensor library**: OpenMAT applications above are based on the LpSensor library. This library contains classes that allow easy access to the functionality of the LPMS devices. Contained classes and their most important methods as well as usage examples are described further on in this chapter.

2. **LpmsControl application**: This application is used to control the basic LPMS device functionality. It can be used to connect to multiple sensors, adjust parameters and record sample data. Data is graphically represented as line graphs or as a 3D cube that changes orientation according to the data received from a sensor.
   PLEASE NOTE: LpmsControl is also used to do updates of the LPMS firmware. We will explain further details below. IMPORTANT: We recommend the users to use the high performance mode of a PC in order to guarantee the LpmsControl application performance.

3. **OpenMAT server**: The OpenMAT server manages the communication of applications on the OpenMAT network. Please contact LP-Research for examples of how to use the OpenMAT network.

OpenMAT is available as binary release and as source code release. If you would like to use the included applications as is, please use the binary release. This is suggested as the easiest way to start as it allows you to test the functionality of your sensor.

We also offer a source code release that allows you to re-compile or modify the code. In case you would like to include OpenMAT with your own applications it is recommended to take a look at the source code release.

Application Installation

Please follow the steps below to install the OpenMAT binary release. The binary release also includes the OpenMAT API pre-compiled for Windows 32-bit.

1. When you purchase one of our sensors the latest version of the library at the time is also contained on the included CD. Please be aware that development on OpenMAT is ongoing and therefore the version on the CD might become outdated. Therefore please check on our website for updates.
2. Start OpenMAT-x.x.x-Setup.exe (x.x.x being the latest version number).
3. Follow the displayed installation instructions.
4. Switch the LPMS device on.
5. Start LpmsControl from the OpenMAT entry in the start menu.
6. Check if your device is listed in the ‘Discovered devices’ list.
7. Mark the device you would like to connect to by clicking on it in the list and push the connect button.
8. After a few seconds you should be seeing data being streamed from your sensor.

LpmsControl Software Operation

Overview

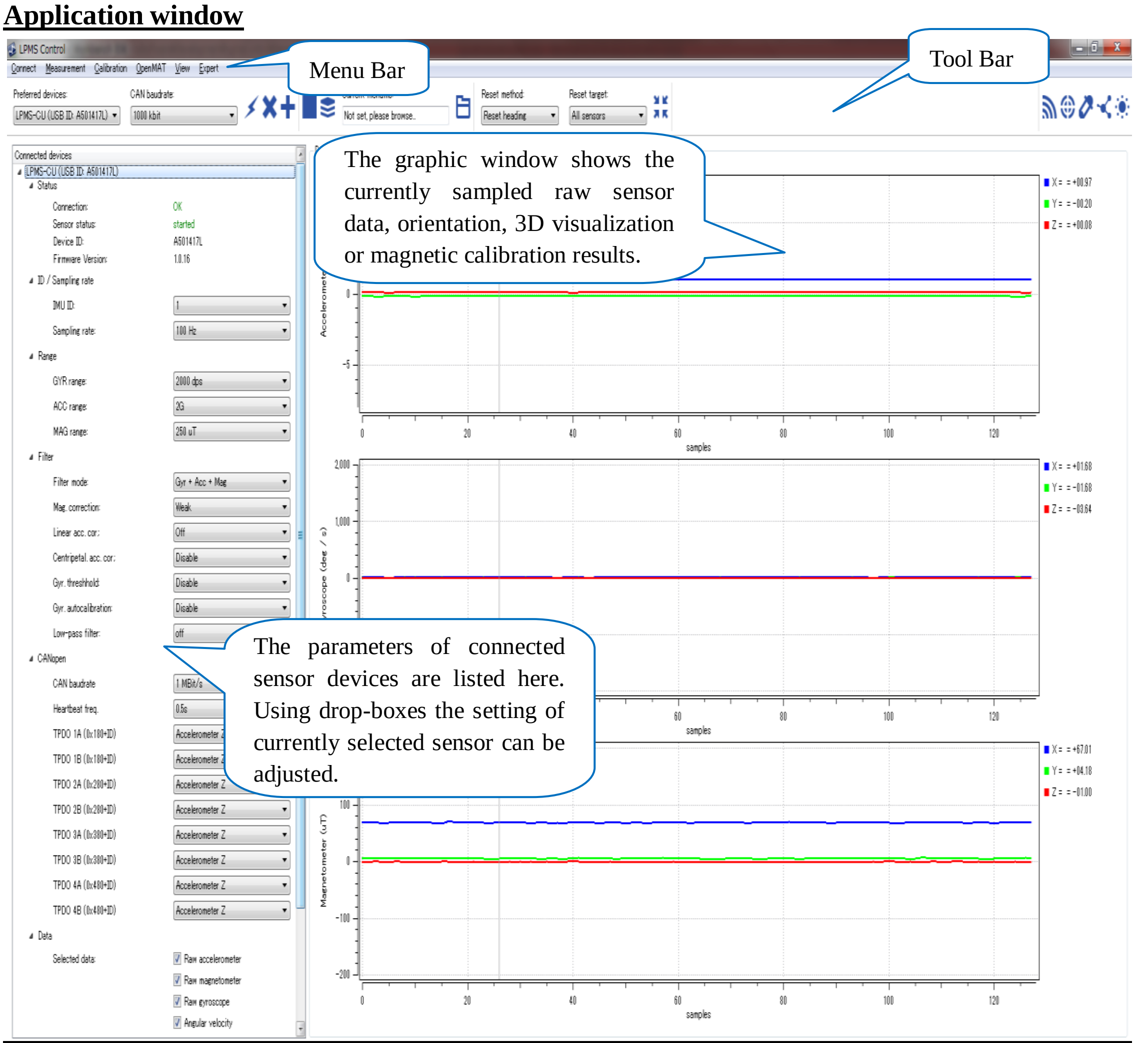
The LpmsControl application allows users to control various aspects of the LPMS-CU sensor that can be used to:

- List all LPMS-CU devices that are discovered in the system.
- Connect to multiple sensors simultaneously over USB interface.
- Adjust the sensor parameters (sensor range etc.).
- Reset orientation and reference vectors.
- Initiate gyroscope and magnetometer calibration.
- Adjust the accelerometer misalignment matrix.
- Display the acquired data in real-time either as line graphs or a 3D cube.
- Record data from the sensors to a CSV data file.
- Upload new firmware and in-application-programming software to the sensor.

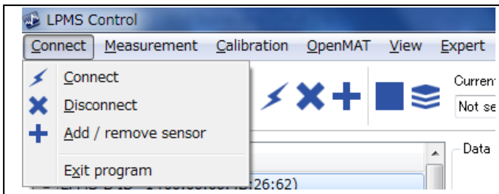
As LpmsControl is part of the open-source OpenMAT package its source code is available and can be modified by the user. Most parts of the code are documented, so that a user can also use parts of LpmsControl to write their own sensor control code.

GUI Elements

Application window



Menu Bar



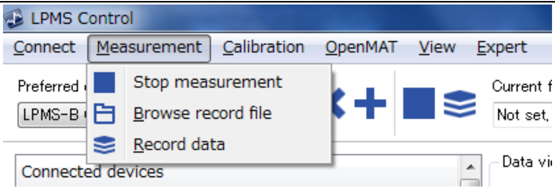
Connect menu

Connect - Connects to sensor selected in 'Preferred devices' list.

Disconnect - Disconnects sensor currently selected in the 'Connected devices' list.

Add / remove sensor – add new discovered sensor from the "Discovered devices" list to "Preferred devices" list, or remove the currently selected sensor from "Preferred devices" list.

Exit program - Exits the application.

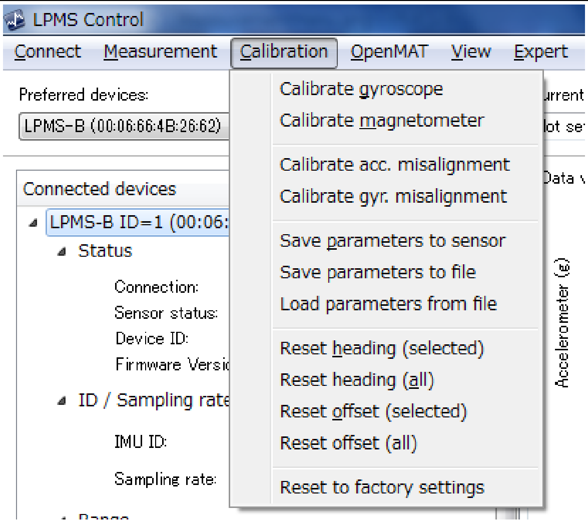


Measurement menu

Stop measurement - Starts or stops a measurement (depending if measurement is already in progress or not).

Browse record file – locates or create a csv format file for saving the recorded data.

Record data - Starts or stops data recording (depending if recording is already in progress or not).



Calibration menu

Calibrate gyroscope – Starts the gyroscope calibration (users should follow the insturctions introdued in section "*Calibration Methods*")

Calibrate magnetometer – Starts the magnetic calibration (users should follow the insturctions introdued in section "*Calibration Methods*"). IMPORTANT: Euler angle transmission must be turned on for the magnetometer calibration to succeed.

Calibrate acc. misalignment – Starts the accelerometer calibration. (users should follow the insturctions introdued in section "*Calibration Methods*").

Calibrate gyr. misalignment – reserved by LP-RESEARCH.

Save parameters to sensor – Saves the current parameter settings and calibration results into the sensor flash.

Save parameters to file – Saves the current parameter settings and calibration results into a .txt file in your local host system.

Load parameters from file – Loads the previously saved calibration results in a local txt file into the sensor flash.

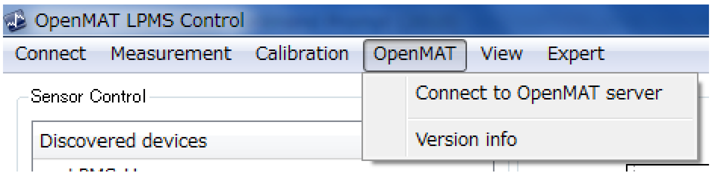
Reset heading (selected) – Sets the magnetometer and accelerometer reference of the LP-Filter of the currenly selected sensor in the "Connected devices" list to the current measured magnetic and acceleration vector. This function should be used after calibrating the magnetometer.

Reset heading (all) – Sets the magnetometer and accelerometer reference of the LP-Filter of all the sensors in the "Connected devices" list to the current measured magnetic and acceleration vector. This function should be used after calibrating the magnetometer.

Reset offset (selected) - Resets the current orientation of the selected sensor in the "Connected devices" list as zero-orientation. Further rotations will be the difference rotation between the zero-orientation and the currently measured orientation.

Reset offset (all) - Resets the current orientation of all the sensors in the "Connected devices" list as zero-orientation. Further rotations will be the difference rotation between the zero-orientation and the currently measured orientation.

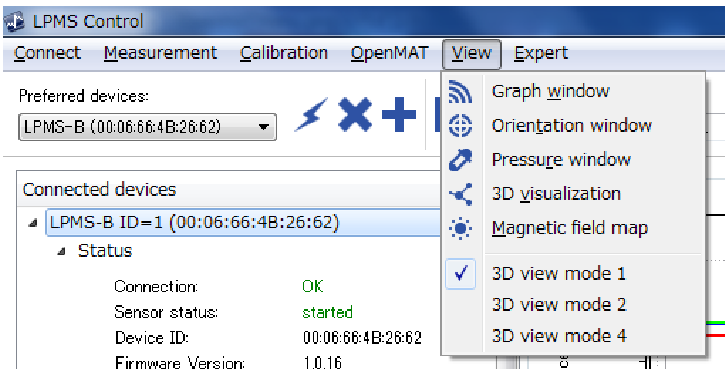
Reset to factory settings – Recovers the settings of all the connected sensors to the factory default values.



OpenMAT menu

Connect to OpenMAT server - This is used for human model simulator. The human model simulator allows the construction of 3D models with links and joints that can be associated with orientation sensors on the OpenMAT network. Momentarily this application is still in an experimental state. PLEASE NOTE: This function is reserved by LP-RESEARCH.

Version info – Version information of the LpmsControl software.



View menu

Graph window- Switches the middle graph window to show the raw sensor data.

Orientation window – Switches the middle graph window to show the orientation data.

Pressure window - Switches the middle graph window to show the pressure data.

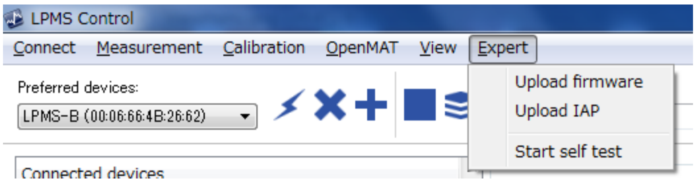
3D visualization - Switches the middle graph window to 3D cube view.

Magnetic field map – Switches the middle graph window to magnetic field map view.

3D view mode 1 - Switches the middle graph window to 1 window for one sensor 3D visualization.

3D view mode 2 - Switches the middle graph window to 2 windows for 2 sensors 3D visualization.

3D view mode 4 - Switches the middle graph window to 4 windows for 4 sensors 3D visualization.



Expert menu

Upload firmware - Starts firmware upload. IMPORTANT: Only upload authorized firmware that you received from LP-RESEARCH. Uploading a wrong firmware file can make the sensor in-operable.

Upload IAP - Uploads a new in-application programmer. PLAESENOTE: This is reserved by LP-RESEARCH and should not be used by user.

Start self test - Starts a self test for checking the basic functionalities of the sensor.

Device Discovery

Discovery can be re-started by clicking on the "Scan devices" button. If the LPMS-CU device cannot be discovered by the initial discovering, please try to push the "Scan devices" button and search again.

IMPORTANT: After you plug-in the LPMS-CU to your USB port, it takes a few seconds until Windows will correctly recognize the device. During that period the device might not be discoverable by the LpmsControl application.

Connecting and Disconnecting a Device

To connect a LPMS-CU sensor click on the corresponding item in the "Preferred devices" list and click the "Connect" button. The sensor should now become listed in the "Connected devices" list. While establishing the connection, the 'Connection status' indicator shows 'connecting…'. Once a connection has been successfully established, the connection status will change to 'connected'. The sensor will start measuring automatically after connecting. Should the connection procedure fail for some reason, 'failed' will be displayed. If a successful connection is interrupted the connection status will change to 'connection interrupted'.

Sensor Parameter Adjustment

Sensor parameters can be adjusted using the item in the connected sensors list that corresponds to the target device. Using the drop down lists the following parameters can be set:

- **IMU ID**: The device OpenMAT ID.

- **Sampling rate**: System sampling frequency
- **GYR range**: Gyroscope measurement range
- **ACC range**: Accelerometer measurement range
- **MAG range**: Magnetometer measurement range
- **Filter mode**: The filter mode setting (see also the previous section "*Filter Settings*")
- **Mag. correction**: The magnetic correction setting (see also the previous section "*Filter Settings*")
- **Linear acc cor**.: The linear acceleration correction setting (see also the previous section "*Filter Settings*")
- **Rotational acc cor**.: The linear rotational correction setting (see also the previous section "*Filter Settings*")
- **Gyr. threshold**: To enable or disable the gyroscope threshold function (see also the previous section "*Filter Settings*")
- **Gyr. autocalibration**: To enable or disable the gyroscope auto calibration function (see also the previous section "*Filter Settings*")
- **Low-pass filter**: To set up the coefficient of the low pass filter (see also the previous section "*Filter Settings*")
- **CAN baudrate**: Selects the baudrate used for CAN bus communication.
- **CANopen heartbeat**: Selects the frequency with which the CANopen heartbeat is transmitted from the sensor.
- **TPDO 0 – 4 data setting**: Selects the data to be transmitted via the CANopen TPDOs. Angular velocity, orientation in Euler angles, orientation quaternion, linear acceleration and raw accelerometer and magnetometer data can be sent.
- **Selected data**: check the data types you want to acquire.

Parameter adjustments are normally only persistent until the sensor is switched off. You can permanently save the newly adjusted parameters to the LPMS flash memory by selecting "Save parameters to sensor" in the "Calibration" menu of LpmsControl.

Reset of Orientation and Reference Vectors

The offset of the orientation measured by the sensor can be set to the currently acquired orientation by clicking on the "Reset offset" functions of LpmsControl. The newly reported orientation data will be the orientation difference between this zero-orientation and the un-adjusted (raw) orientation measurement.

The accelerometer and magnetometer reference vector is reset by clicking on the "Reset heading" function of LpmsControl. Before resetting the heading reference, PLEASSE DO complete the magnetic calibration. While initiating the heading reference reset, point the y axis of the sensor roughly in north direction and hold the x-y plane of the sensor parallel to the ground.

IMPORTANT: The adjustment of the heading reference vectors is very important for accurate orientation measurements. The sensor will be delivered to you in a pre-calibrated state. However, as the direction of the earth magnetic field slightly varies at different place, it might be necessary to reset the reference. To save the new heading reference after a successful reset, select "Save parameters to sensor" function of LpmsControl. Normally the setting of the heading reference vectors when done accurately only needs to be done once.

How to Upload New Firmware

IMPORTANT: Please follow the following steps carefully when you are updating the sensor firmware. Any mistake operation might result in a failure of firmware update and disable sensor functionality.

1. Start your current LpmsControl software.
2. Connect to the sensor you would like to update.
3. Choose the "Save parameters to file" function from the calibration menu of LpmsControl to save the current sensor calibration results into a .txt file in your local host system.
4. Select "Upload firmware" function in the "Expert" menu.
5. Click OK and select the new firmware file. Be careful that you select the right file which should be named as LpmsBFirmwareX.X.X.bin by LP-RESEARCH.
6. Wait for the upload process to finish. It should take around 30 seconds. At around 15s the green LED on the sensor should begin to blink rapidly.
7. Disconnect from the sensor and exit LpmsControl.
8. Now install the new LpmsControl application. The previous LpmsControl application does not need to be un-installed.
9. Start LpmsControl and connect to your sensor.
10. Choose the "Load parameters from file" function from the calibration menu of LpmsControl to recover the previous sensor calibration results.
11. Choose the "Save parameters to sensor" function from the calibration menu of LpmsControl to save the previous sensor calibration results into sensor flash.
12. The whole procedure is done. Make sure everything works as expected. If there is anything unexpected, please contact LP-RESEARCH by Email: info@lp-research.com.

The LpSensor Library

Building Your Application

The LpSensor library contains classes that allow a user to integrate LPMS devices into their own applications. The library is a Windows 32-bit C++ library for MS Visual C++ (express) 2010. Should you require a binary for the library for other operating systems or 64-bit applications, please contact LP-RESEARCH. Compiling applications that use the LpSensor library requires the following components:

Header files (usually in C:/OpenMAT/include)

LpmsSensorManagerI.h	Contains the interface for the LpmsSensorManager class.
LpmsSensorI.h	Contains the interface for the LpmsSensor class
ImuData.h	Structure for containing output data from a LPMS device
LpmsDefinitions.h	Macro definitions for accessing LPMS
DeviceListItem.h	Contains the class definition for an element of a LPMS device list

LIB files (usually in C:/OpenMAT/lib/x86)

LpSensorD.lib	LpSensor library (Debug version)
LpSensor.lib	LpSensor library (Release version)

DLL files (usually in C:/OpenMAT/lib/x86)

LpSensorD.dll	LpSensor library (Debug version)
LpSensor.dll	LpSensor library (Release version)

PCANBasic.dll PeakCAN library DLL for CAN interface communication. This file is only needed, if you use a PeakCAN interface to communicate with LPMS-CU.

ftd2xx.dll The FTDI library to communicate with an LPMS over USB.

To compile the application please do the following:

1. Include LpmsSensorManagerI.h before you access any LpSensor classes.
2. Add LpSensor.lib (or LpSensorD.lib if you are compiling in debug mode) to the ist of linked libraries for your application.
3. Make sure that you set a path to LpSensor.dll / LpSensorD.dll, PCANBasic.dll and ftd2xx.dll reside so that the runtime file of your application can access them.

Important Classes

SensorManager

The sensor manager class wraps a number of LpmsSensor instances into one class, handles device discovery and device polling. For user applications the following methods are most commonly used. Please refer to the interface file SensorManagerI.h for more information.

IMPORTANT: An instance of LpmsSensor is returned by the static function **LpmsSensorManagerFactory()**. See the example listing in the next section for more information how to initialize a LpmsSensorManager object.

Method name	**SensorManager(void)**
Parameters	none
Returns	SensorManager object
Description	Constructor of a SensorManager object.

<table>
<tr><td>Method name</td><td colspan="2">LpSensor* addSensor(int mode, string deviceId)</td></tr>
<tr><td rowspan="2">Parameters</td><td>mode</td><td>The device type to be connected. The following device types are available:
<table>
<tr><td>Macro</td><td>Device type</td></tr>
<tr><td>DEVICE_LPMS_B</td><td>LPMS-B</td></tr>
<tr><td>DEVICE_LPMS_C</td><td>LPMS-CU (CAN mode)</td></tr>
<tr><td>DEVICE_LPMS_U</td><td>LPMS-CU (USB mode)</td></tr>
</table></td></tr>
<tr><td>deviceId</td><td>Device ID of the LPMS device. The ID is equal to the OpenMAT ID (initially set to 1, user definable).</td></tr>
<tr><td>Returns</td><td colspan="2">Pointer to LpSensor object.</td></tr>
<tr><td>Description</td><td colspan="2">Adds a sensor device to the list of devices adminstered by the SensorManager object.</td></tr>
<tr><td>Method name</td><td colspan="2">void removeSensor(LpSensor *sensor)</td></tr>
<tr><td>Parameters</td><td>sensor</td><td>Pointer to LpSensor object that is to be removed from the list of sensors. The call to removeSensor frees the memory associated with the LpSensor object.</td></tr>
<tr><td>Returns</td><td colspan="2">none</td></tr>
<tr><td>Description</td><td colspan="2">Removes a device from the list of currently administered sensors.</td></tr>
</table>

Method name		**void listDevices(std::vector<DeviceListItem> *v)**
Parameters	***v**	Pointer to a vector containing DeviceListItem objects with information about LPMS devices that have been discovered by the method.
Returns		None
Description		Lists all connected LPMS devices. The device discovery runs in a

	seperate thread.For Bluetooth devices should take several seconds to be added to the devicelist. CAN bus and USB devices should be added after around 1s.

LpmsSensor

This is a class to access the specific functions and parameters of an LPMS. The most commonly used methods are listed below. Please refer to the interface file LpmSensorI.h for more information.

Method name	`void run(void)`
Parameters	None
Returns	None
Description	Starts the data acquisition procedure.

Method name	`void pause(void)`
Parameters	None
Returns	None
Description	Pauses the data acquisition procedure.

<table>
<tr><th>Method name</th><th><code>int getSensorStatus(void)</code></th></tr>
<tr><td>Parameters</td><td>None</td></tr>
<tr><td>Returns</td><td>Sensor state identifier:
<table>
<tr><th>Macro</th><th>Sensor state</th></tr>
<tr><td>SENSOR_STATUS_PAUSED</td><td>Sensor is currently paused.</td></tr>
<tr><td>SENSOR_STATUS_RUNNING</td><td>Sensor is currently acquiring data.</td></tr>
<tr><td>SENSOR_STATUS_CALIBRATING</td><td>Sensor is currently calibrating.</td></tr>
<tr><td>SENSOR_STATUS_ERROR</td><td>Sensor has detected an error.</td></tr>
<tr><td>SENSOR_STATUS_UPLOADING</td><td>Sensor is currently receiving new firmware data.</td></tr>
</table></td></tr>
<tr><td>Description</td><td>Retrieves the current sensor status.</td></tr>
</table>

Method name	`int getConnectionStatus(void)`
Parameters	None

<table>
<tr><td>Returns</td><td>Connection status identifier:
<table>
<tr><td>Macro</td><td>Sensor state</td></tr>
<tr><td><code>SENSOR_CONNECTION_CONNECTED</code></td><td>Sensor is connected.</td></tr>
<tr><td><code>SENSOR_CONNECTION_CONNECTING</code></td><td>Connection is currently being established.</td></tr>
<tr><td><code>SENSOR_CONNECTION_FAILED</code></td><td>Attempt to connect has failed.</td></tr>
<tr><td><code>SENSOR_CONNECTION_INTERRUPTED</code></td><td>Connection has been interrupted.</td></tr>
</table></td></tr>
<tr><td>Description</td><td>Retrieves the current connection status.</td></tr>
</table>

Method name	**`void startResetReference(void)`**
Parameters	None
Returns	None
Description	Resets the current accelerometer and magnetometer reference. Please see the ‘Operation’ chapter for details on the reference vector adjustment procedure.

Method name	**`void startCalibrateGyro(void)`**
Parameters	None
Returns	None
Description	Starts the calibration of the sensor gyroscope.

Method name	**`void startCalibrateMag(void)`**
Parameters	None
Returns	None
Description	Starts the calibration of the LPMS magnetometer.

Method name	**`CalibrationData* getConfigurationData(void)`**
Parameters	None
Returns	Pointer to CalibrationData object.
Description	Retrieves the CalibrationData structure containing theconfigurationparameters ofthe connected LPMS.

Method name	**`bool setConfigurationPrm(int parameterIndex, int`**

	parameter)
Parameters	**parameterIndex** The parameter to be adjusted. **parameter** The new parameter value. Supported parameterIndex identifiers:

Macro	Description
PRM_OPENMAT_ID	Sets the current OpenMAT ID.
PRM_FILTER_MODE	Sets the current filter mode.
PRM_PARAMETER_SET	Changes the current filter preset.
PRM_GYR_THRESHOLD_ENABLE	Enables / diables the gyroscope threshold.
PRM_MAG_RANGE	Modifies the current magnetometer sensor range.
PRM_ACC_RANGE	Modifies the current accelerometer sensor range.
PRM_GYR_RANGE	Modifies the current gyroscope range.

Supported parameter identifiers for each parameter index:

PRM_OPENMAT_ID

`Integer ID number between 1 and 255.`

PRM_FILTER_MODE

Macro	Description
FM_GYRO_ONLY	Only gyroscope
FM_GYRO_ACC	Gyroscope + accelerometer
FM_GYRO_ACC_MAG_NS	Gyroscope + accelerometer + magnetometer

PRM_PARAMETER_SET

Macro	Description
LPMS_FILTER_PRM_SET_1	Magnetometer correction “dynamic” setting.
LPMS_FILTER_PRM_SET_2	Strong

	LPMS_FILTER_PRM_SET_3 | Medium LPMS_FILTER_PRM_SET_4 | Weak **PRM_GYR_THRESHOLD_ENABLE** **Macro** | **Description** IMU_GYR_THRESH_DISABLE | Enable gyr. threshold IMU_GYR_THRESH_ENABLE | Disable gyr. thershold **PRM_GYR_RANGE** **Macro** | **Description** GYR_RANGE_250DPS | Gyr. Range = 250 deg./s GYR_RANGE_500DPS | Gyr. Range = 500 deg./s GYR_RANGE_2000DPS | Gyr. Range = 2000 deg./s **PRM_ACC_RANGE** **Macro** | **Description** ACC_RANGE_2G | Acc. range = 2g ACC_RANGE_4G | Acc. range = 4g ACC_RANGE_8G | Acc. range = 8g ACC_RANGE_16G | Acc. range = 16g **PRM_MAG_RANGE** **Macro** | **Description** MAG_RANGE_130UT | Mag. range = 130uT MAG_RANGE_190UT | Mag. range = 190uT MAG_RANGE_250UT | Mag. range = 250uT MAG_RANGE_400UT | Mag. range = 400uT MAG_RANGE_470UT | Mag. range = 470uT MAG_RANGE_560UT | Mag. range = 560uT MAG_RANGE_810UT | Mag. range = 810uT
Returns	None
Description	Sets a configuration parameter.

Method name	**`bool getConfigurationPrm(int parameterIndex, int *parameter)`**

Parameters	**parameterIndex** The parameter to be adjusted. **parameter** Pointer to the retrieved parameter value. See setConfigurationPrm method for an explanation of supported paramer indices and parameters.
Returns	None
Description	Retrieves a configuration parameter.

Method name	`void resetOrientation(void)`
Parameters	None
Returns	None
Description	Resets the orientation offset of the sensor.

Method name	`void saveCalibrationData(void)`
Parameters	None
Returns	None
Description	Starts saving the current parameter settings to the sensor flash memory.

Method name	`virtual void getCalibratedSensorData(float g[3], float a[3], float b[3])`
Parameters	**g[0..2]** Calibrated gyroscope data (x, y, z-axis). **a[0..2]** Calibrated accelerometer data (x, y, z-axis). **b[0..2]** Calibrated magnetometer data (x, y, z-axis).
Returns	None
Description	Retrieves calibrated sensor data (gyroscope, accelerometer, magnetometer).

Method name	`virtual void getQuaternion(float q[4])`
Parameters	**q[0..3]** Orientation quaternion (qw, qx, qy, qz)
Returns	None
Description	Retrieves the 3d orientation quaternion.

Method name	`virtual void getEulerAngle(float r[3])`
Parameters	**r[0..2]** Euler angle vector (around x, y, z-axis)
Returns	None
Description	Retrieves the currently measured 3d Euler angles.

Method name	**`virtual void getRotationMatrix(float M[3][3])`**
Parameters	**M[0..2][0..2]** Rotations matrix (row i=0..2, column j=0..2)
Returns	None
Description	Retrievs the current rotation matrix.

Example Code

Connecting to the an LPMS-CU device

```
#include "LpmsSensorI.h"
#include "LpmsSensorManagerI.h"

main()
{
	// Get a LpmsSensorManager instance
	LpmsSensorManagerI* manager = SensorManagerFactory();

	// Connect to LPMS-CU sensor with address A123456
	LpmsSensorI* lpms = manager->addSensor(DEVICE_LPMS_U, "A123456");

	while(1) {
		float q[4];

		// Read quaternion data
		lpms->getQuaternion(q);

		// Do something with the data
		// ..
	}

	// After doing the work, remove the initialized sensor
	sm->removeSensor(lpms);

	// Delete LpmsSensorManager object
	delete manager;
```

```
}
```

Setting and Retrieval of Sensor Parameters

```
/* Setting a sensor parameter. */
lpmsDevice->setParameter(PRM_ACC_RANGE, LPMS_ACC_RANGE_8G);

/* Retrieving a sensor parameter. */
int p;
lpmsDevice->setParameter(PRM_ACC_RANGE, &p);
```

Sensor and Connection Status Inquiry

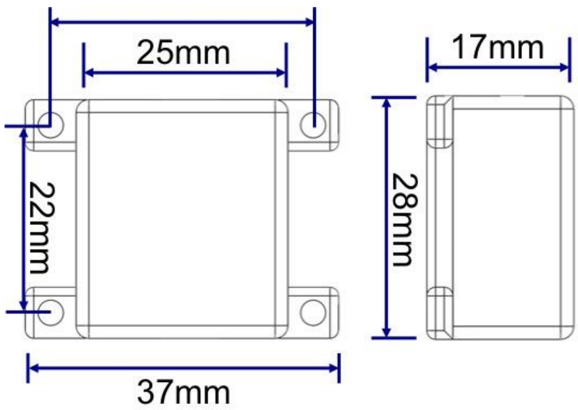
```
/* Retrieves current sensor status */
int status = getSensorStatus();

switch (status) {
case SENSOR_STATUS_RUNNING:
        std::cout << "Sensor is running." << std::endl;
break;

case SENSOR_STATUS_PAUSED:
        std::cout << "Sensor is paused." << std::endl;
break;
}

status = lpmsDevice->getConnectionStatus();

switch (status) {
case SENSOR_CONNECTION_CONNECTING:
        std::cout << "Sensor is currently connecting." << std::endl;
break;

case SENSOR_CONNECTION_CONNECTED:
        std::cout << "Sensor is connected." << std::endl;
break;
}
```

In case you have any further questions regarding the programming interface please contact LP-RESEARCH directly.

X. MECHANICAL INFORMATION

LPMS-CU Dimension

Plastic casing:



Aluminum casing:

Please Read Carefully:

Information in this document is provided solely in connection with LP-RESEARCH products. LP-RESEARCH reserves the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All LP-RESEARCH products are sold pursuant to LP-RESEARCH's terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the LP-RESEARCH products and services described herein, and LP-RESEARCH assumes no liability whatsoever relating to the choice, selection or use of the LP-RESEARCH products and services described herein.

UNLESS OTHERWISE SET FORTH IN LP-RESEARCH'S TERMS AND CONDITIONS OF SALE LP-RESEARCH DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF LP-RESEARCH PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

LP-RESEARCH PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. LP-RESEARCH PRODUCTS WHICH ARE NOT SPECIFIED AS "AUTOMOTIVE GRADE" MAY ONLY BE USED IN AUTOMOTIVE APPLICATIONS AT USER'S OWN RISK.

Japan – China – Germany – Korea – Egypt
www.lp-research.com