A2C-IMU-C / A2C-IMU-M12

Features

- 1kHz maximum data rate
- CAN 2.0A and 2.0B compatible.
- Ultra-low noise accelerometer of 100µg/√Hz
- Selectable full-scale range of ±2 / 4 / 8 / 16g
- Selectable BW of 420/220/100/45/21/10/5Hz
- Ultra-low drift gyro of 4mdsp//√Hz
- Selectable full-scale range of ±250 / ±500 / ±1000 / ±2000dps
- Selectable BW of 176 / 92 / 41 / 20 / 10 / 5Hz
- Sensor fusion for linear accelerations, tilt and roll angles, predictive quaternion.
- Programmable alarms
- Programmable heartbeat CAN messages
- Periodic CAN messages with programmable data and time periods
- Logic alarm output (open drain) for standalone mode without the use of CAN bus (M12 version)
- Software upgradable via CAN bus

A2C-IMU-C-A

A2C-IMU-M12-A

Applications

- Ground based robotics
- Underwater robotics
- Electrical carts
- Mining vehicles
- Vehicle developments
- Acceleration measurements on industrial machines
- Tilt measurements on industrial machines
- Low frequency machine vibration
- Crane boom angle measurements
- Vehicle accelerations measurements
- Mobile lift boom angle and acceleration alarms
- Wind turbine blade acceleration measurements

General Description

The A2C-IMU sensor is a 6 DOF sensor with a 3-axis accelerometer and a 3-axis gyro. It contains a high-performance DSP processor to combine the data with sensor fusion to provide many types of measurements which are sent to a host via CAN Bus.

Specifications

- 3-Axis ±2*g-16g MEMS sensor*
- 7-30V supply voltage
- 10mA supply current @ 24V
- CAN interface (2.0A, 2.0B)
- Cable or standard industrial M12 connector
- CNC machined aluminum / stainless steel housing
- Housing size 55x55x17mm (Cable version)
- Housing with M12 50x50x19.5mm

1 Ordering information

Part Number	Package	Interface	CAN Bus	Logic Output
A2C-IMU-M12-A-O	50x65x19.5mm Anodized aluminum	M12A, 5pin Male connector.	Yes	Yes (pin 1)
A2C-IMU-M12-A-N	50x65x19.5mm Anodized aluminum	M12A, 5pin Male connector.	Yes	No (pin 1 is connected to housing internally)
A2C-IMU-M12-S-O	50x65x19.5mm Stainless steel 316	M12A, 5pin Male connector.	Yes	Yes (pin 1)
A2C-IMU-M12-S-N	50x65x19.5mm Stainless steel 316	M12A, 5pin Male connector.	Yes	No (pin 1 is connected to housing internally)
A2C-IMU-C-A-N	55x55x17mm Anodized aluminum	1m cable, 4 wires and 1 shield	Yes	No
A2C-IMU-C-S-N	55x55x17mm Stainless steel 316	1m cable, 4 wires and 1 shield	Yes	No

For a customer specific package please contact us. We have other materials / coatings available not listed here.

Specifications for (A2C-IMU-C / A2C-IMU-M12) Smart IMU to CAN-Bus Sensors

Version 1.9
April 2026

Document tracking control

VERSION	SECTION	CHANGED BY	DATE	CHANGE
1.0	All	JL	01-10-2019	Initial Version
1.1	All	NP	23-12-2019	Updated all sections
1.2	All	NP	13-09-2020	Updated 8 and up.
1.3	12-15	JL	04-12-2020	Added new features
1.4	2	JL	03-11-2021	Specifications updated
1.5	8-12	JL	17-11-2021	Added additional info / features
1.6	2/4/20	JL	26-9-2022	Added Alarm section. Added M12 version
1.7	4	JL	26-10-2023	Added M12 version mechanical drawing
1.8	9	JL	06-01-2026	Clarified new avg values
1.9	7/10/18	JL	20-04-2026	System Modes added. Get info updated. Heart beat message remove. List of supported CMDs for periodic messages added

Contents

Specifications

Parameter	Condition	Values			Unit
SENSOR INPUT		Min	Typical	Max	
Measurement Range		±2		±16	*g*
Nonlinearity			0.005		*g*
Sensor Package Alignment Error			±0.5		*Deg*
Inter-axis Alignment Error			±0.5		*Deg*
Cross Axis Sensitivity			±0.5		*Deg*
FREQUENCY RESPONSE					
Accelerometer bandwidth (-3dB)[1]		5		420	Hz
Gyro bandwidth (-3dB)[2]		5		176	Hz
NOISE PERFORMANCE					
Accelerometer Power Spectral Density - all axis	@ 10Hz		100		µg/√Hz
Accelerometer RMS Noise – all axis	BW = 100Hz		1.0		mg - rms
Gyro Rate Noise Spectral Density - all axis	@ 10Hz		0.004		dps /√Hz
Gyro Total RMS Noise – all axis	BW = 100Hz		0.04		dps - rms
SUPPLY VOLTAGE		Min	Typical	Max	
Operating Voltage (Vin)		7		30	V
Supply Current	Vin = 24V		10		mA
Power consumption	Vin = 24V		0.24		W
Turn-On Time			1000		ms
OPEN DRAIN OUTPUT (-O versions)					
Maximum collector emitter voltage			40		V
Maximum continuous current			100		mA
Maximum recommended inductance - including wires			5		mH
Shutdown current		4	5.5	7.5	A
CAN BUS 2.0A & B					
Transceiver delay loop time				150	ns
Baudrate		50		1000	kBits/sec
Default device standard ID			0x125		
Default device filters			0x3E8-0x3EB		
Software Protocol			Proprietary		
Hardware Protocol			2.0A / 2.0B		
HOUSING					
Housing Body Material –A suffix			Anodized Aluminum		
Housing Body Material –S suffix			Stainless steel 316		
Lid Material - A suffix			Aluminum		
Lid Material - S suffix			Stainless steel 316		
CONNECTIVITY					
A2C-IMU-M12			M12-A Male 5 pin Connector		
Pin 1 = N-Version = Shield / O-Version = Output					
Pin 2 = Vin					
Pin 3 = Ground					
Pin 4 = CAN High					
Pin 5 = CAN Low					
A2C-IMU-C (cable length 1m)			2x2pair+shield		
Red wire = Vin					
Black wire = Ground					
White wire = CAN High					
Blue wire = CAN Low					
Yellow heat shrink tube = Shield					
WEIGHT		Min	Typical	Max	
-M12-A			---		gram
-C-A (1m cable)			165		gram
TEMPERATURE		Min	Typical	Max	
Operating Temperature Range		-20		80	Deg
Housing temperature rise				5	Deg

[1] Additional averaging for low frequency sensing available
[2] Additional averaging for low frequency sensing available

User configurations	
PROGRAMMABLE ALARMS	See alarm sections for more details
PERIODIC MESSAGES Messages can be sent periodically at programmable periods	Acceleration from all axis (one signed 16bit value per axis) Inclination from all axis (one signed 16bit value per axis with programmable resolution) Rotations

Conformity	
IEC 60721-3-5 Climate Biological Chemically active substances Mechanically active substances Contaminating fluids Mechanical conditions	Tests are ongoing

1.1 *Input voltage range*

The input voltage must be kept below 30V. Above 31V the input protection clamp begins to conduct, causing internal heat built up. This clamp is designed to protect against voltage spikes of very short durations.

1.2 *Output switch*

The output switch is designed to drive inductive loads such as a relay or coil. The inductance must be less than 10mH including all line inductances. It is recommended to place a freewheeling diode in parallel with the output, close to the inductive source.

1.3 *Sensing Direction / co-ordinate system*

The positive acceleration and rotations can be seen in **Error! Reference source not found.**.

The negative acceleration, act in the opposite direction as shown. The measurements include the acceleration by the earth gravity, so the axis pointing towards the earth will measure 1*g*. This also implies that only an additional acceleration of 1*g* in the z direction is possible before saturating the accelerometer.

Figure 2 shows how the sensor will respond to different orientations – without any additional vibration and rotations.

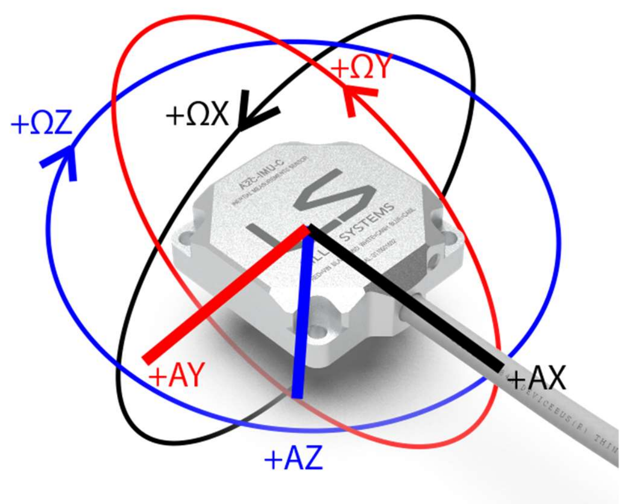


Figure 1

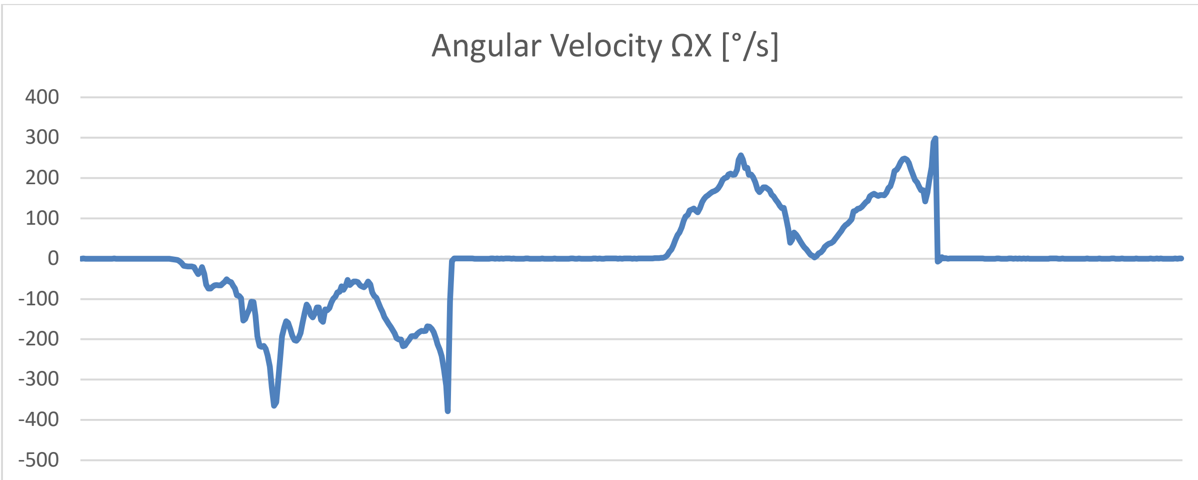
	X-Axis	**Y-Axis**	**Z-axis**
Positive Accelerations	 **X = 1*g*, y = 0*g*, z = 0*g***	 **X = 0*g*, y = 1*g*, z = 0*g***	 **X = 0*g*, y = 0*g*, z = 1*g***
Negative Accelerations	 **X = -1*g*, y = 0*g*, z = 0*g***	 **X = 0*g*, y = -1*g*, z = 0*g***	 **X = 0*g*, y = 0*g*, z = -1*g***

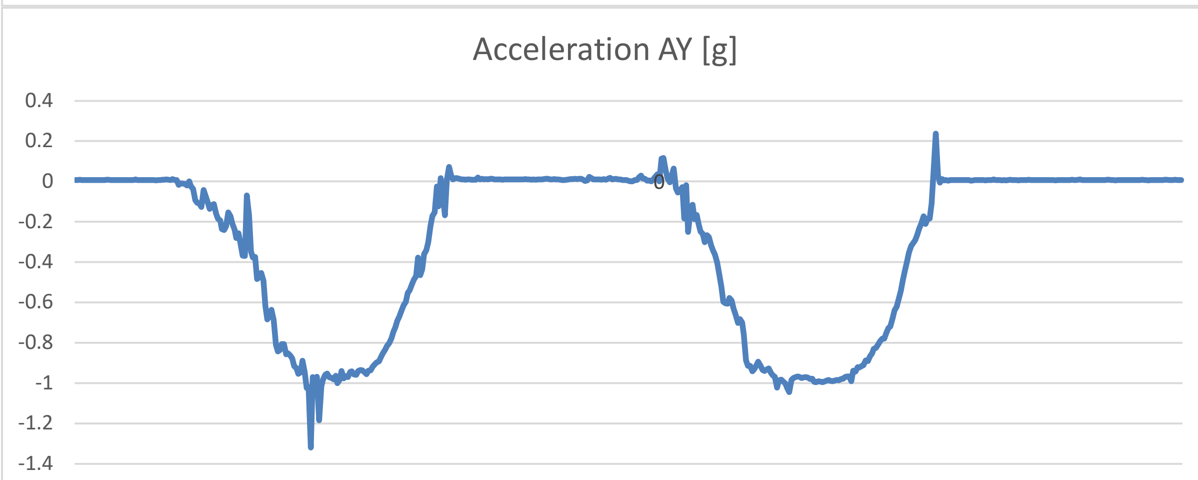
Figure 2

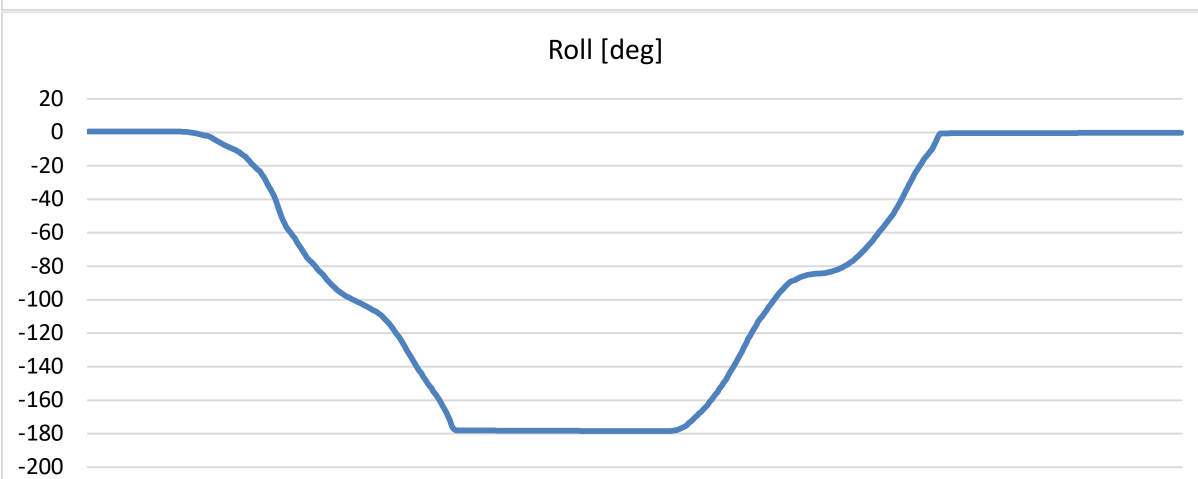
2 Sensor Fusion

Sensor fusion of accelerometer and gyro enables precise inclination angles and linear accelerations to be calculated. The A2C-IMU includes advanced Kalman filtering with temperature compensation, which guarantees consistently high performance.
As of firmware 1.26 the sensor fusion must have the gyro full scale range set to 2000 dps to work correctly.

Below is an example of sensor fusing taking the Acceleration and gyro data to calculate the precise roll angle.







Mechanical Drawing

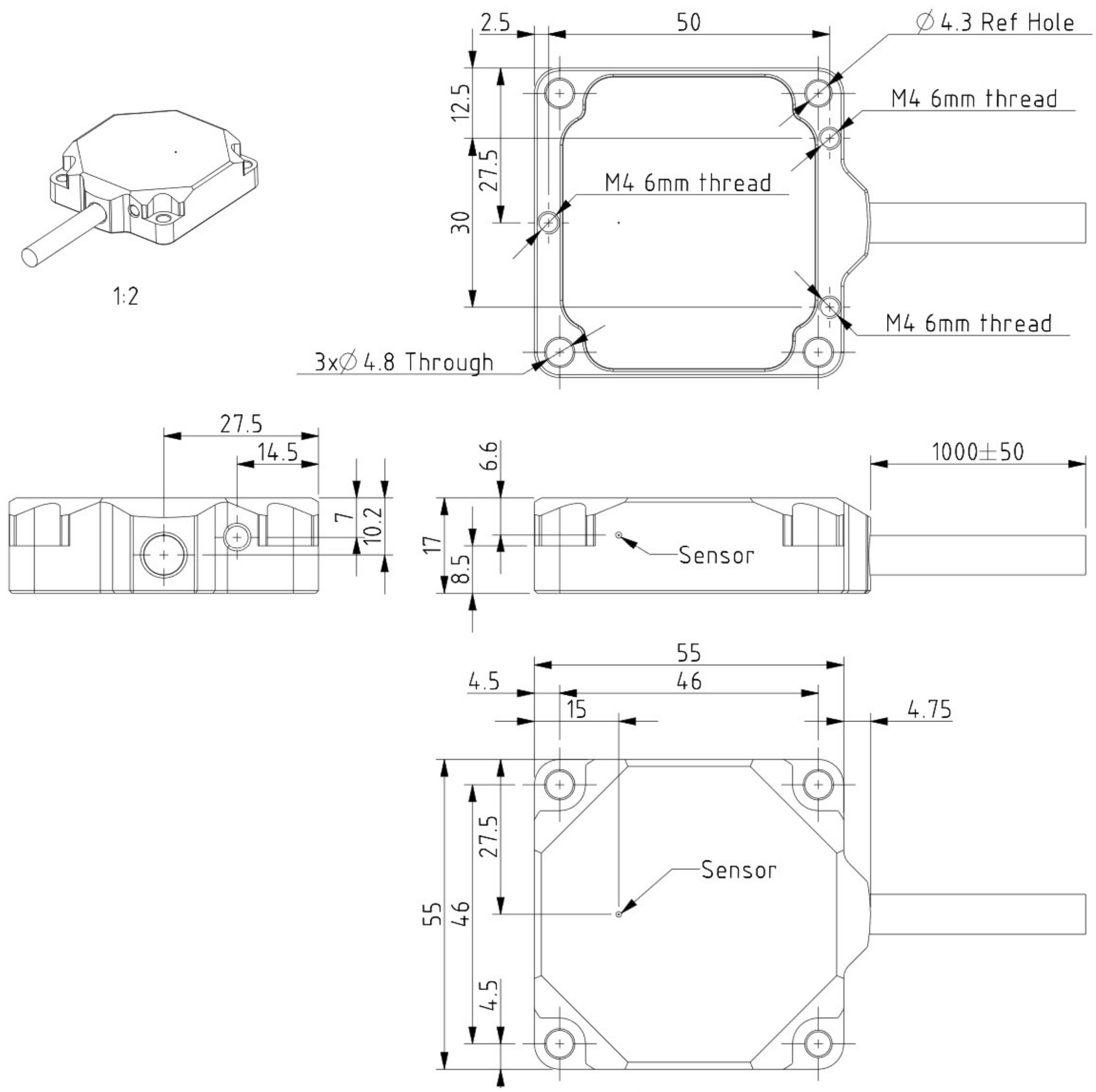


Figure 3 - A2C-IMU-C

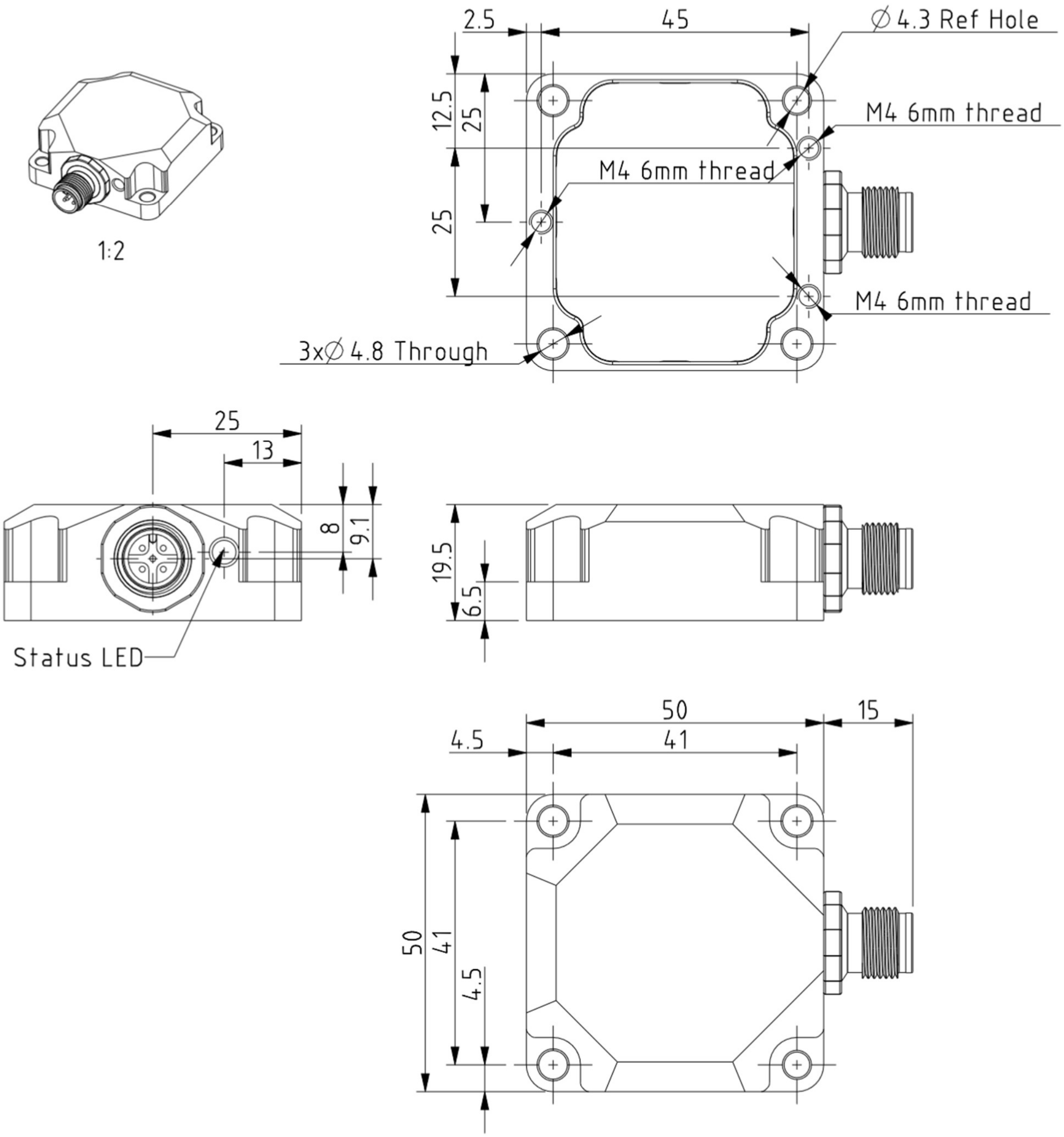
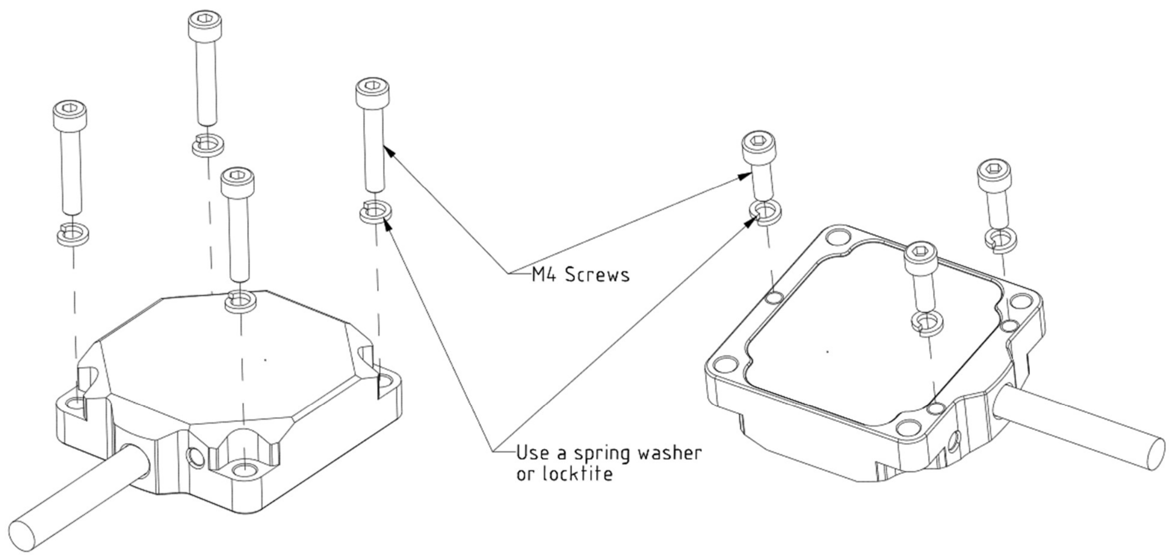


Figure 4 - A2C-IMU-M12

3 Mounting

Use two to 2-4 x M4 bolts to secure the sensor from the top or 3 x M4 screws from the bottom.



4 Programming tools

In order to simplify programming and to test sensors from Lillie System, a programming tool and accompanying software may be purchased to speed up development. If the sensor will be used in standalone mode, these tools are essential. No understanding of CAN bus and programming is required.

The U2C is the programming tool which connects to the USB port of a windows PC in one end, and the sensor CAN bus on the other end. The U2C also functions as a general USB to CAN Bus adaptor / bus monitor.

Figure 5 - U2C Programming tool

The window GUI enables the user to easily set all parameters in the sensor, and in real time see the accelerations and angles.

5 Protocol

It is important to understand the simple protocol before reading further. The first CAN byte is always the command. The second CAN byte is a sub-command. The remaining 6 CAN bytes are Data.

5.1 *Protocol format*

Communication takes place over a CAN bus Interface
The communication can use both 11-bit or 29bit frame format – CAN 2.0A / 2.0B / FD 1.0.

	RTR	DLC	Command	Sub command	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5
Bit length	1	4	8	8	8	8	8	8	8	8
Range	0 Always 0	1 - 8	0x00 – 0xff	0x00 – 0xff	0x00 – 0xff	0x00 – 0xff	0x00 – 0xff	0x00 – 0xff	0x00 – 0xff	0x00 – 0xff

Identifier: Default Identifier is set to 0x123 from factory, but can be changed as shown in 6.1.1
RTR: RTR is not used, so it must always be 0.
DLC: DLC should be between 1 and 8. There is always at least one data byte as they are used as a commando word.
Command: Command byte
Sub Command: Sub command byte
When data bytes are combined to form 16 or 32bit variables the big endian system is used.

6 Initial Setup

The sensor comes with factory setting such as CAN filters, CAN Identifiers, bandwidth, mode etc. To prepare the sensor for operation some CAN settings might need to be changed. This is described in this section.

6.1 *CAN Identifier*

The CAN Identifier is the identifier which the sensor sends when transmitting messages.
The factory default value is 0x125.

6.1.1 Command: Set CAN ID

To change the CAN ID to other values send the following message:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x68	**STD_EXT**	**ID MSB**	**ID**	**ID**	**ID LSB**		
DLC = 0x06 (values above 0x06 are also valid, but Data bytes are not used)							

STD_EXT:

- 0x01 = CAN Standard ID (11bit identifier)
- 0x02 = CAN Extended ID (29bit identifier)

ID: This is the CAN Identifier that the sensor uses when transmitting data, sent at an unsigned 32bit integer

- [0x000 – 0x7FF] for CAN Standard ID (11bit identifier). **Data[0] and Data[1] must both be 0x00!**
- [0x00000000 – 1FFFFFFF] for CAN Extended ID (29bit identifier)

6.1.2 Command: Get Standard CAN ID

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xE8	**Any value**						
DLC = 0x02 (values above 0x02 are also valid, but Data bytes are not used)							

Reply

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xE8	**STD_EXT**	**ID MSB**	**ID**	**ID**	**ID LSB**		
DLC = 0x06							

The reply format follows the same format as setting the CAN ID as seen in 6.1.1

6.2 *Baud rate*

The Baud rate is the communication speed on the CAN bus. It can be set to predefined values or to a custom value. The maximum CAN bus cable length is dependent on the baud rate. In general, bus speed of 1 Mega bits is used up to 40m, 500kbits/sec up to 100m, 250kbits/sec up to 250m and 50kbits/sec up to 1000m. These values can vary. Please read additional information on the internet about CAN bus speed and cable lengths.

6.3 *Sample-point*

For all standard baud rates, the sample-point is 75%. If you need a different sample point, then you must use a custom baud rate as is shown in 6.4

6.3.1 Command: Set baud rate

The default baud rate from the factory is 250kbits/s, but we may pre-program the baud rate for customers, which is indicated in the inlay with the sensor. To change the baud rate to another value send the following message:
NB: Be aware that changing the baud rate will take effect immediately, the IMU will save the value and restart. This also implicates that other unsaved changes will be saved.

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x67	**BAUD**	**AUTOTRANS**	**Any value**	**(char) 'S' 0x53**	**(char) 'A' 0x41**	**(char) 'F' 0x46**	**(char) 'E' 0x45**
DLC = 0x08							

BAUD: See table below for valid values

- 0x01 = 1 Mb/s – sampling point 87.5%
- 0x02 = 500 kb/s – sampling point 87.5%
- 0x03 = 250 kb/s – sampling point 87.5%

- 0x04 = 125 kb/s – sampling point 87.5%
- 0x05 = 100 kb/s – sampling point 87.5%
- 0x06 = 50 kb/s – sampling point 87.5%
- 0x07 = Reserved for future use
- 0x08 = Reserved for future use
- 0x09 = Custom Baud rate.
- 0x0A = 1 Mb/s – sampling point 75%
- 0x0B = 500 kb/s – sampling point 75%
- 0x0C = 250 kb/s – sampling point 75%
- 0x0D = 125 kb/s – sampling point 75%
- 0x0E = 100 kb/s – sampling point 75%
- 0x0F = 50 kb/s – sampling point 75%

AUTOTRANS: Enable / disable automatic re-transmission on CAN bus

- 0x00 = No automatic retransmission
- 0x01 = Automatic retransmission

In addition of the **BAUD** and **AUTOTRANS** values, the data bytes 2 to 5 must contain the chars as shown in 6.3.1. This is to some degree prevent the baud rate to change and cause a CAN bus error if the filters are set incorrectly.

6.3.2 Command: Get baud rate

To get the current baud rate

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xE7							
DLC = 0x01 (values above 1 are also valid, but Data bytes are not used)							

Reply

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xE7	**BAUD**	**AUTOTRANS**	**Not defined**				
DLC = 0x04							

The reply format follows the same format as seen in 6.3.1

6.4 *Custom baud rate*

The following values must be calculated first.

$$T1 = \frac{36 \times 10^6}{Baud\ rate \times PRES}$$

$$\boldsymbol{BS1} = T1 \times Sample\ Point - 1, must\ be\ less\ than\ 16$$

$$\boldsymbol{BS2} = T1 - BS1 - 1, must\ be\ less\ than\ 8$$

Example: Generate baud rate of 83.33kbits / second with a sample point of 87.5%: We first select a prescale (PRES) value that creates an even T1 number. 27 is selected.

$$\frac{36 \times 10^6}{83333 \times 27} = 16 = T1$$

$$16 \times 0.875 - 1 = 13 = BS1 - satisfy\ a\ value\ less\ than\ 16, ok!$$

$$16 - 13 - 1 = 2 = BS2 - satisfy\ a\ value\ less\ than\ 8, ok!$$

6.4.1 Command: Set custom baud rate

From the above calculations we can now send the following message.

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x54	**0x01**	**SJW**	**BS1**	**BS2**	**PRES_MSB**	**PRES_LSB**	
DLC = 0x07 (values above 7 are also valid, but Data bytes are not used)							

SJW: Resynchronization Jump Width, Specifies the maximum number of time quanta the CAN hardware is allowed to lengthen or shorten a bit to perform resynchronization. This parameter can be a value of:

- 0x00 = 1 time quantum
- 0x01 = 2 time quantums
- 0x02 = 3 time quantums
- 0x03 = 4 time quantums

BS1: Specifies the number of time quanta in Bit Segment 1. This parameter can be a value of
- 0x00 = 1 time quantum
- 0x01 = 2 time quantums
- …
- 0x0F = 16 time quantums

BS2: Specifies the number of time quanta in Bit Segment 2. This parameter can be a value of
- 0x00 = 1 time quantum
- 0x01 = 2 time quantums
- …
- 0x07 = 8 time quantums

PRES_MSB: Specifies the MSB prescale value
PRES_LSB: Specifies the LSB prescale value

6.4.2 Command: Get custom baud rate

To get the current baud rate

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xC3	**Any value**	**SJW**	**BS1**	**BS2**	**PRES_MSB**	**PRES_LSB**	
DLC = 0x01 (values above 1 are also valid, but Data bytes are not used)							

Reply

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xC3	**Value sent**	**SJW**	**BS1**	**BS2**	**PRES_MSB**	**PRES_LSB**	
DLC = 0x02							

The reply format follows the same format as seen in 6.4.1

6.5 *CAN Filters*

The sensor will only respond to values which have passed through its CAN message filters. This means that many similar sensors can be attached to the same CAN network and by defining filters, only the sensor nodes which filter matches the CAN ID will interpret the message.
There are two types of filters; standard filters which are unsigned 16bit integers and used for 11 bit identifiers, and extended filters which are unsigned 32 bit integers and used for 29bit identifiers. There are 4 standard filters and 2 extended filters. The standard filters only allow a message with the same ID as the filter value to pass through.

From the factory settings the filters are configured as follows:
- Standard Filter 1 = 1000
- Standard Filter 2 = 1001
- Standard Filter 3 = 1002
- Standard Filter 4 = 1003
- Extended Filter 1 = 0
- Extended Filter 2 = 0

6.5.1 Command: Set CAN Filters

To change the filters to other values send the following message:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]
0x69	**FILT**	**MSB Std Filter 1&3** **MSB Ext Filter 1&2**	**LSB Std Filter 1&3**	**MSB Std Filter 2&4**	**LSB Std Filter 2&4** **LSB Ext Filter 1&2**
DLC = 0x06 (values above 6 are also valid, but Data bytes are not used)					

FILT: Filter Number
- 0x01 = Standard Filter 1&2
- 0x02 = Standard Filter 3&4

- 0x03 = Extended Filter 1
- 0x04 = Extended Filter 2

MSB Std Filter 1&3: Most significant bit of standard filters 1 & 3. [0x00-0x07]
LSB Std Filter 1&3: Least significant bit of standard filters 1 & 3. [0x00-0xFF]

MSB Std Filter 2&4: Most significant bit of standard filters 2 & 4. [0x00-0x07]
LSB Std Filter 2&4: Least significant bit of standard filters 2 & 4. [0x00-0xFF]

MSB Ext Filter 1&2: Most significant bit of extended filter 1. [0x00-0x1F]
LSB Ext Filter 1&2: Least significant bit of extended filter 1. [0x00-0xFF]

Example 1
Set standard filters 1&2 to 0x0123 and 0x01C1 respectively

Byte 0	***Byte 1***	***Byte 2***	***Byte 3***	***Byte 4***	***Byte 5***	***Byte 6***	***Byte 7***
0x69	*0x01*	*0x01*	*0x23*	*0x01*	*0xC1*		
DLC = 0x06							

Example 2
Set standard filters 3&4 to 0x0100 and 0x0734 respectively

Byte 0	***Byte 1***	***Byte 2***	***Byte 3***	***Byte 4***	***Byte 5***	***Byte 6***	***Byte 7***
0x69	*0x02*	*0x01*	*0x00*	*0x07*	*0x34*		
DLC = 0x06							

Example 3
Set extended filter 1 to 0x01020304

Byte 0	***Byte 1***	***Byte 2***	***Byte 3***	***Byte 4***	***Byte 5***	***Byte 6***	***Byte 7***
0x69	*0x03*	*0x01*	*0x02*	*0x03*	*0x04*		
DLC = 0x06							

6.5.2 Command: Get CAN Filters

To get the current sensor filter settings send the following CAN message:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xE9	**FILT**						
DLC = 0x02 (values above 2 are also valid, but Data bytes are not used)							

FILT: Filter Number

- 0x01 = Get Standard Filter 1&2
- 0x02 = Get Standard Filter 3&4
- 0x03 = Get Extended Filter 1
- 0x04 = Get Extended Filter 2

The sensor will reply:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]
0xE9	**FILT**	**MSB Std Filter 1&3** **MSB Ext Filter 1&2**	**LSB Std Filter 1&3**	**MSB Std Filter 2&4**	**LSB Std Filter 2&4** **LSB Ext Filter 1&2**
DLC = 0x06					

The reply format follows the same format as setting the filters. See 6.5.1

7 Setup System Mode

From firmware versions starting at 290 the sensor can operate in different modes: Sensor Fusion mode (default) or RMS vibration mode. The default mode is mode 1, sensor fusion. Other modes offer RMS calculations of individual axis or combined into a vector.

7.1.1 Command: Set System Mode

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x40	**MODE**						
DLC = 0x04 (values above 4 are also valid, but Data bytes are not used)							

MODE	Description
0	RESERVED
1 (Default)	Sensor Fusion. All accelerations, angular velocity and pitch, yaw, roll available
2	RMS vibration mode. Individual RMS calculation for each axis. No sensor fusion.
3	RMS vibration mode. X-Y Vector RMS calculation. No sensor fusion.
4	RMS vibration mode. Y-Z Vector RMS calculation. No sensor fusion.
5	RMS vibration mode. X-Z Vector RMS calculation. No sensor fusion.
6	RMS vibration mode. X-Y-Z Vector RMS calculation. No sensor fusion.

Example.
“Set the sensor system mode to RMS vibration to return one vector from all axis:

Byte 0	*Byte 1*	*Byte 2*	*Byte 3*	*Byte 4*	*Byte 5*	*Byte 6*	*Byte 7*
0x40	*0x06*						
DLC = 0x2							

8 Setup Full Scale Range

The full-scale range determines the maximum range of the sensor for both the accelerometer and the gyro.

8.1.1 Command: Set Full Scale Range

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x59	**A_G**	**FSR**					
DLC = 0x04 (values above 4 are also valid, but Data bytes are not used)							

FSR	A_G = 1 (Configure Accelerometer)	A_G = 2 (Configure Gyro)
0	±2g	±250dps
1	±4g	±500dps
2	±8g	±1000dps
3	±16g	±2000dps

After changing the FSR a save parameters command must be executed and the sensor restarted.
Note: For sensor fusion to correctly calculate the inclinations, the full-scale gyro range of 2000dps must be used. This applies to linear accelerations, inclination, linear vector sum.

Example.
“Set the sensor gyro to have a full scale range of 2000dps:

Byte 0	*Byte 1*	*Byte 2*	*Byte 3*	*Byte 4*	*Byte 5*	*Byte 6*	*Byte 7*
0x59	*0x02*	*0x03*					
DLC = 0x3							

To read the full scale range back from the sensor, refer to 10.1.2

9 Setup Bandwidth

The bandwidth determines what frequencies the sensor can sense. The higher the bandwidth, the higher the frequencies, lower response time but higher noise. The lower the bandwidth the lower frequencies the sensor can sense, but the noise will be lower too.

9.1.1 Command: Set System Bandwidth

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x58	**A_G**	**BW**					
DLC = 0x04 (values above 4 are also valid, but Data bytes are not used)							

BW	A_G = 1 (Configure Accelerometer)	A_G = 2 (Configure Gyro)
7	420Hz	-
1	220Hz	176Hz
2	100Hz	92Hz
3	45Hz	41Hz
4	21Hz	20Hz
5	10Hz	10Hz
6	5Hz	5Hz

Example.
"Set accelerometer sensor to 20Hz Bandwidth

Byte 0	*Byte 1*	*Byte 2*	*Byte 3*	*Byte 4*	*Byte 5*	*Byte 6*	*Byte 7*
0x58	*0x01*	*0x04*					
DLC = 0x3							

To read the Bandwidth back from the sensor, refer to 10.1.2

10 Setup Averaging

The sensor can be setup to average the data using a simple moving average filter. Sensor noise is reduced when increasing the number of samples, so for low frequency measurements it is better to set the highest possible value that still meets performance criteria. However, it should be checked with the application that the response is fast enough when setting higher values, as the phase shift (delay) is also increased.

The number of samples can be set from 0 to 16.
There are two separate averaging values, one for System Mode 1 (Sensor Fusion) and one for RMS vibration modes.
For the averaging in Sensor Fusion mode, this occurs after sensor fusion is calculated and affects both accelerations, inclinations and rotations. It is not possible at this time to set a different averaging for each measurement.

10.1.1 Command: Set System Averaging

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x60	**0x00**	**STD_AVG**	**RMS_AVG**				
DLC = 0x04 (values above 4 are also valid, but Data bytes are not used)							

- **STD_AVG**: Number of averaging for sensor fusion data; Acceleration, Linear acceleration, angular velocity, pitch, yaw, roll will all be averaged. This number must be from 0-16.
- **RMS_AVG**: Number of averaging for RMS vibration samples

10.1.2 Command: Get All Setup Info

A single command is issued to get the entire setup of the sensor.

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x62							
DLC = 0x02 (values above 4 are also valid, but Data bytes are not used)							

The sensor will reply with 6 CAN messages:
Message 1 & 2: Full Scale Range of Accelerometer / Gyro

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]
0x59	**A_G**	**FSR**			
DLC = 0x08					

FSR	**A_G = 1 (Accelerometer)**	**A_G = 2 (Configure Gyro)**
0	±2g	±250dps
1	±4g	±500dps
2	±8g	±1000dps
3	±16g	±2000dps

Message 3 & 4: Bandwidth of Accelerometer / Gyro

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]
0x58	**A_G**	**BW**			
DLC = 0x08					

BW	**A_G = 1 (Configure Accelerometer)**	**A_G = 2 (Configure Gyro)**
7	420Hz	-
1	220Hz	176Hz
2	100Hz	92Hz
3	45Hz	41Hz
4	21Hz	20Hz
5	10Hz	10Hz
6	5Hz	5Hz

Message 5: Averaging of Accelerometer / Gyro

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]
0x60	**0**	**STD_AVG**	**RMS_AVG**		
DLC = 0x08					

- **STD_AVG:** Number of samples which are averaged (Sensor Fusion mode)
- **RMS_AVG:** Number of samples which are averaged in any of the RMS vibration modes.

Message 6: Get System Mode

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]
0xC0	**0**	**MODE**			
DLC = 0x08					

MODE:

0	RESERVED
1 (Default)	Sensor Fusion. All accelerations, angular velocity and pitch, yaw, roll available
2	RMS vibration mode. Individual RMS calculation for each axis. No sensor fusion.
3	RMS vibration mode. X-Y Vector RMS calculation. No sensor fusion.
4	RMS vibration mode. Y-Z Vector RMS calculation. No sensor fusion.
5	RMS vibration mode. X-Z Vector RMS calculation. No sensor fusion.
6	RMS vibration mode. X-Y-Z Vector RMS calculation. No sensor fusion.

11 J1939 Mode CAN ID when sending data

The A2C-IMU can operate in two different modes when sending data.

The CAN ID of the sensor is normally a fixed value and all messages sent use this same CAN ID as described in 6.1.
To identify the message, the first and second data bytes are used (command, sub-command). An example is getting the acceleration. Data[0] is 0x0A and Data[1] could be any value between 0-8 to indicate which axis etc.

Many systems however use the CAN ID itself to identify what kind of data is being sent. In order to accommodate these systems, the A2C-IMU can be set to run in J1939 mode. In this mode a CAN ID offset is added to original CAN ID so that each message is sent with its own ID. To enable the J1939 mode send the command below.

11.1.1 Command: Set J1939 CAN ID Format

Send this command to sensor:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x6E	**MODE**						
DLC = 0x02 (values above 0x02 are also valid, but Data bytes are not used)							

MODE:

- 0x00 = turn OFF J1939 type messages
- 0x01 = turn ON J1939 type messages for some messages.

11.1.2 Command: Get J1939 CAN ID Format Send this command:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x6F							
DLC = 0x01 (values above 0x01 are also valid, but Data bytes are not used)							

Reply:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x6F	**VAL**						
DLC = 0x02							

VAL returns 0x00 / 0x01 - as described for the Set command 0x6E

12 Getting Accelerations

The accelerations from the sensor are sent as integer values. Depending on the full-scale range (FSR), two different multipliers are used. For FSR of ±2g the scaling is 10000. For all other FSR the multiplier is 1000.

There are two commands to receive accelerations. The first command gets all 3 axes at the same time. For the second command the user must specify which axis to receive and this returns the current acceleration, the minimum acceleration and the maximum accelerations. The minimum and maximum values can be reset by sending the "Set Values To Zero" command see 0.
Some of the commands support the J1939 mode. The CAN ID offset is written in parenthesis.

12.1.1 Command: Get All Accelerations

Send this command to sensor:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x0A	**RET**						
DLC = 0x02 (values above 2 are also valid, but Data bytes are not used)							

RET: determines the returned value.

- 0x00 = Get current normal acceleration values for all axis, which are continuously updated
- 0x01 = Get current linear acceleration values for all axis, which are continuously updated
- 0x02 = Planned future functionality.
- 0x03 = Get X-axis acceleration, current, min, max (J1939 CAN ID offset = 7)
- 0x04 = Get Y-axis acceleration, current, min, max (J1939 CAN ID offset = 8)
- 0x05 = Get Z-axis acceleration, current, min, max (J1939 CAN ID offset = 9)
- 0x06 = Get X-axis linear acceleration, current, min, max (J1939 CAN ID offset = 13)
- 0x07 = Get Y-axis linear acceleration, current, min, max (J1939 CAN ID offset = 14)
- 0x08 = Get Z-axis linear acceleration, current, min, max (J1939 CAN ID offset = 15)

After the 0x0A and a sub command have been received, the sensor will return the accelerations

Reply when RET = 0x00 or 0x01

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x0A	**Value sent is returned**	**X axis MSB**	**X axis LSB**	**Y axis MSB**	**Y axis LSB**	**Z axis MSB**	**Z axis LSB**
DLC = 0x08							

Reply when RET = 0x02-0x08

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x0A	**Value sent is returned**	**Current Acceleration MSB**	**Current Acceleration LSB**	**Min. Acceleration MSB**	**Min. Acceleration LSB**	**Max. Acceleration MSB**	**Max. Acceleration LSB**
DLC = 0x08							

12.1.2 Command: Get single axis acceleration, including min / max values

This command is only to support backwards compatibility with the A2C-TRI sensor. Use command 0x0A instead

Send this command to sensor:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x0C	**AXIS**						

AXIS: must be one of the following values

- 0x00 = X-axis
- 0x01 = Y-axis
- 0x02 = Z-axis

Reply:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x0C	**AXIS**	**Current Acceleration MSB**	**Current Acceleration LSB**	**Min. Acceleration MSB**	**Min. Acceleration LSB**	**Max. Acceleration MSB**	**Max. Acceleration LSB**

DLC = 0x08

AXIS:

- 0 = X-axis
- 1 = Y-axis
- 2 = Z-axis

13 Getting Inclination

The inclination from the sensor are sent as integer values multiplied with a specified number given by the subcommand.

13.1.1 Command: Get All Inclinations

Send this command to sensor:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x0B	**RET**						
DLC = 0x02 (values above 2 are also valid, but Data bytes are not used)							

RET:

- 0x00 = All 3 inclinations sent and result is rounded to nearest integer value i.e. 85= 85deg
- 0x01 = All 3 inclinations sent and result is multiplied by 10 e.g. a value of 854 = 85,4deg
- 0x02 = All 3 inclinations sent and result is multiplied by 100 e.g. a value of 8542 = 85,42deg
- 0x03 = Get only Pitch angle multiplied with 100, current, min, max (J1939 CAN ID offset = 19)
- 0x04 = Get only Roll angle multiplied with 100, current, min, max (J1939 CAN ID offset = 20)
- 0x05 = Get only Yaw angle multiplied with 100, current, min, max (J1939 CAN ID offset = 21)

.

After this command has been received, the sensor will return the angles for all 3 accelerometers.

Reply when RET = 0x00-0x02

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x0B	**RET**	**X axis angle MSB**	**X axis angle LSB**	**Y axis angle MSB**	**Y axis angle LSB**	**Z axis angle MSB**	**Z axis angle LSB**
DLC = 0x08							

Reply when RET = 0x03-0x05

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x0B	**RET**	**Current Inclination MSB**	**Current Inclination LSB**	**Min. Inclination MSB**	**Min. Inclination LSB**	**Max. Inclination MSB**	**Max. Inclination LSB**
DLC = 0x08							

Note: For sensor fusion to correctly calculate the inclinations, the full scale gyro range of 2000 dps must be used.

14 Getting angular velocity

The angular velocity from the sensor are sent as integer values multiplied by a factor which is dependent on the full-scale range. There are two commands to receive angular velocity. The first command gets all 3 axes at the same time.

14.1.1 Command: Get All Angles

Send this command to sensor:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x0D	**RET**						
DLC = 0x02 (values above 2 are also valid, but Data bytes are not used)							

RET: determines if current values or a previous synced value is used

- 0x00 = Get current values, which are continuously updated
- 0x03 = Get only ΩY (pitch axis), current, min, max (J1939 CAN ID offset = 25)
- 0x04 = Get only ΩX (roll axis), current, min, max (J1939 CAN ID offset = 26)
- 0x05 = Get only ΩZ (yaw axis), current, min, max (J1939 CAN ID offset = 27)

After this command has been received, the sensor will return the angular velocity from all 3 gyros.

Reply when RET = 0x00

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x0D	**RET**	**X axis rotation MSB**	**X axis rotation LSB**	**Y axis rotation MSB**	**Y axis rotation LSB**	**Z axis rotation MSB**	**Z axis rotation LSB**
DLC = 0x08							
Scaling (Quantization)							
Full-scale range ±2g: 0.0001 [g/digit] Full-scale range ±4/8/16g: 0.001 [g/digit]		**Full-scale range ±2g: 0.0001 [g/digit] Full-scale range ±4/8/16g: 0.001 [g/digit]**		**Full-scale range ±2000°/s: 0.06250 [°/s/digit] Full-scale range ±1000°/s: 0.03125 [°/s/digit] Full-scale range ±500°/s: 0.015625 [°/s/digit] Full-scale range ±250°/s: 0.0078125 [°/s/digit]**		**0.01 [°/digit]**	

Reply when RET = 0x03-0x05

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x0D	**RET**	**Current rotation MSB**	**Current rotation LSB**	**Min. rotation MSB**	**Min. rotation LSB**	**Max. rotation MSB**	**Max. rotation LSB**
DLC = 0x08							

15 Getting RMS data (beta)

For use in vibration sensing mode the RMS vibrations can be calculated.

15.1.1 Command: Get RMS data

Send this command to sensor:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x10							
DLC = 0x01 (values above 1 are also valid, but Data bytes are not used)							

After this command has been received, the sensor will return data, which depends on the System Mode setting

System Mode	DLC	Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]
2	**7**	**0x10**	**X axis RMS MSB**	**X axis RMS LSB**	**Y axis RMS MSB**	**Y axis RMS LSB**	**Z axis RMS MSB**	**Z axis RMS LSB**
3	**3**	**0x10**	**X-Y Vector RMS MSB**	**X-Y Vector LSB**				
4	**3**	**0x10**	**Y-Z Vector RMS MSB**	**Y-Z Vector LSB**				
5	**3**	**0x10**	**X-Z Vector RMS MSB**	**X-Z Vector LSB**				
6	**3**	**0x10**	**X-Y-Z Vector RMS MSB**	**X-Y-Z Vector LSB**				
DLC = 0x07								
Full-scale range ±2g: **0.0001 [g/digit]** **Full-scale range ±4/8/16g:** **0.001 [g/digit]**								

16 Getting combined data – J1939 style

The combined Linear Acceleration, Normal Acceleration, Angular Velocity and Inclination data are sent in one CAN-message for each axis.

Each CAN-message has its own CAN ID offset. From the factory the sensor is programmed with a CAN ID 0x0125. To this there is an offset for each axis.

Note: The J1939 CAN ID mode described in section 11 has no influence on this command.

16.1.1 Command: Get Combined Data

Send this command to sensor:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x09	**Axis**						
DLC = 0x02 (values above 2 are also valid, but Data bytes are not used)							

Axis: determines the returned value.

- 0x00 = Get X-axis combined data
- 0x01 = Get Y-axis combined data
- 0x02 = Get Z-axis combined data

After the 0x09 and a sub command have been received, the sensor will return the data.
Reply:

Axis	**CAN ID Offset**	**CAN ID Factory Default**
X-Axis	+ 0x01	0x0126
Y-Axis	+ 0x02	0x0127
Z-Axis	+ 0x03	0x0128

CAN-Message content

Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
Linear Acceleration MSB	**Linear Acceleration LSB**	**Normal Acceleration MSB**	**Normal Acceleration LSB**	**Angular Velocity MSB**	**Angular Velocity LSB**	**Inclination MSB**	**Inclination LSB**
DLC = 0x08							

Scaling (Quantization)

Linear Acceleration	Normal Acceleration	Angular Velocity	Inclination
Full-scale range ±2g: **0.0001 [g/digit]** **Full-scale range ±4/8/16g:** **0.001 [g/digit]**	**Full-scale range ±2g:** **0.0001 [g/digit]** **Full-scale range ±4/8/16g:** **0.001 [g/digit]**	**Full-scale range ±2000°/s:** **0.06250 [°/s/digit]** **Full-scale range ±1000°/s:** **0.03125 [°/s/digit]** **Full-scale range ±500°/s:** **0.015625 [°/s/digit]** **Full-scale range ±250°/s:** **0.0078125 [°/s/digit]**	**0.01 [°/digit]**

17 Command: Reset Global Minimum & Maximum

The minimum and maximum accelerations, inclinations and angular velocities are continuously updated. They can be set to the current value i.e. reset by sending the following message:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x0F	**VALUE**						
DLC = 0x02 (values above 0x02 are also valid, but Data bytes are not used)							

VALUE:

- 0x01 = Reset acceleration value for all 3-axis.
- 0x02 = Reset acceleration value for all X-axis.
- 0x03 = Reset acceleration value for all Y-axis.
- 0x04 = Reset acceleration value for all Z-axis.

18 Getting Sensor Information

The sensor information can be requested at any time.
The following information can be sent from the sensor:

- Sensor Serial Number
- Firmware Number
- Hardware Revision
- Sensor Type
- Firmware Number - Bootloader

18.1.1 Command: Get sensor information

Send this command to sensor:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xEF	**INFOTYPE**						
DLC = 0x02 (values above 2 are also valid, but Data bytes are not used)							

INFOTYPE:

- 0x01 to 0x03 = reserved
- 0x04 = Firmware Number, is the version of the software in the sensor
- 0x05 = reserved
- 0x06 = Sensor Type, a number specifying the type of sensor.
- 0x14 = Serial number (same as is laser engraved on the sensor)

After this command has been received, the sensor will return requested information as an unsigned 32bit integer

Reply:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xEF	**INFOTYPE**	**INFO_MSB**	**INFO**	**INFO**	**INFO_LSB**		
DLC = 0x06							

19 Setting up Periodic Messages

The sensor can be configured to send periodic CAN messages at a user specified time interval for each message. The messages that can be sent periodically are:

- Linear Acceleration, Normal Acceleration, Angular velocity and inclination for a single axis; command: 0x09
- Acceleration from all axis; command: 0x0A
- Inclination angles from all axis; command: 0x0B
- Average, minimum and maximum acceleration from a single axis; command: 0x0C
- Gyro rotations, command: 0x0D

A maximum of 8 periodic messages can be setup at the same time – 6 tasks. Each message can be sent with an interval ranging from 2ms (500 messages per second) up to 65535ms.

The baud rate of the sensor will have an influence on the maximum messages per second that can be transmitted.

- 1000kbit/s = 2500 messages /s
- 500kbit/s = 2000 messages /s
- 250kbit/s = 1000 messages /s
- 150kbit/s = 500 messages /s

To setup periodic messages the mode must be set to acceleration or angle mode. The CAN message that controls the periodic messages starts with the command byte (byte 0) which must be 0x52 followed by the subcommand (byte 1) which represents the message number. The message number is currently limited to a value of 1-8. Byte 2 is the state of the period message which can be 0x00 for Off or 0x01 for On. Byte 3 is the value of the command and Byte 4 is the value of the subcommand which would normally be requested e.g. 0x0C and 0x01 respectively for receiving maximum and minimum acceleration from the Y-axis. The bytes 5 & 6 are the periodic interval. This is sent as an unsigned 16bit integer with a value between 2 and 65535.
The list of commands (data) which can be sent periodically is limited to the following commands:

- 0x09 - Get Combined Data
- 0x0A - Get All Accelerations
- 0x0B - Get All Inclinations
- 0x0C - Get single axis acceleration, including min / max values
- 0x0D - Get All Angles
- 0x0F - Reset Global Minimum & Maximum
- 0x10 – Get Vibration RMS Accelerations (beta)

To turn Off a periodic message byte 2 must be set to 0x00. **Note that when a periodic message is turned off, the command, sub-command and period values will not be updated and the values sent will be ignored.**

19.1.1 Command: Set Periodic Messages

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x52	**MsgNbr**	**State**	**Cmd**	**SubCmd**	**TimeMSB**	**TimeLSB**	
DLC = 0x07 (values above 0x07 are also valid, but Data bytes are not used)							

- **MsgNbr:** Message number currently 1-8
- **State:** 0x01 for ON, 0x00 for OFF
- **Cmd:** Command number
- **SubCmd:** Sub-command number
- **TimeMSB:** MSB of time period [0x00 – 0xFF]
- **TimeLSB:** LSB of time period [0x02 – 0xFF]

Example 1
Set periodic message number 1 to send heart beat every second

Byte 0	*Byte 1*	*Byte 2*	*Byte 3*	*Byte 4*	*Byte 5*	*Byte 6*	*Byte 7*
0x52	*0x01*	*0x01*	*0xC0*	*N/A*	*0x03*	*0xE8*	
DLC = 0x07							

An interval of 1 second is 1000ms = 0x03E8. MSB is sent first, then LSB.

Example 2
Set periodic message number 3 to send average, minimum and maximum accelerations 100 times per second for the Z-axis

Byte 0	***Byte 1***	***Byte 2***	***Byte 3***	***Byte 4***	***Byte 5***	***Byte 6***	***Byte 7***
0x52	*0x02*	*0x01*	*0x0C*	*0x02*	*0x00*	*0x0A*	
DLC = 0x07							

An interval of 100 times per second is a message every 10ms = 0x000A. MSB is sent first, then LSB.

Example 3
Set periodic message number 3 to Off – stop sending this periodic message

Byte 0	***Byte 1***	***Byte 2***	***Byte 3***	***Byte 4***	***Byte 5***	***Byte 6***	***Byte 7***
0x52	*0x03*	*0x00*	*0x0C*	*0x02*	*0x00*	*0x0A*	
DLC = 0x07							

If the above is sent, then bytes 3 to 6 will be discarded and not be changed in the sensor. To change these bytes, byte 2 must be set to 0x01.

19.1.2 Command: Get Periodic Messages

Send this command to sensor:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xC1	**MsgNbr**						
DLC = 0x02							

The sensor will reply with:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xC1	**MsgNbr**	**State**	**Cmd**	**SubCmd**	**TimeMSB**	**TimeLSB**	
DLC = 0x07							

20 Setting Alarms

Alarms are used to monitor accelerations, inclinations, linear accelerations and angular rate. They can be setup to trigger in an event of a given limit exceeding a threshold. A total of 4 alarms can be set with different trigger values. For each alarm a hysteresis can be set to keep the alarm On until the value is lower than the hysteresis. For each alarm the minimum alarm output time can be set. This is especially useful for the logic output to hold the output for a given minimum time.

A minimum of 2 messages are required to setup the alarms;

- Command 0x6B sets the alarm number from 0-3, enables the alarm with more or less logic, sets which measurement to trigger on and the trigger value and hysteresis
- Command 0x53 global parameters of all alarms. These parameters define how an alarm is reported, the logic hold time, delay between CAN messages etc. See section20.1.4

20.1.1 Calculating Trigger / Hysteresis

The Trigger point follows the data format used for receiving accelerations etc. Hysteresis is calculated the same way.

Acceleration Full-Scale Range	Acceleration Trigger Range	Angular Full-Scale Range	Angular Trigger Range	Inclination Trigger Range
±2g	±20000 0.0001 [g/digit]	±2000°/s	±32000 0.06250 [°/s/digit]	±17999 0.01 [°/digit]
±4g	±4000 0.001 [g/digit]	±1000°/s	±32000 0.03125 [°/s/digit]	
±8g	±8000 0.001 [g/digit]	±500°/s	±32000 0.015625 [°/s/digit]	
±16g	±16000 0.001 [g/digit]	±250°/s	±32000 0.0078125 [°/s/digit]	

Example 1. Calculate trigger point for positive 1.55g acceleration when full scale range is ±2g: 1.55g/0.0001 = 15500.
Example 2. Calculate trigger point for negative angular velocity of -300°/s when full scale range is ±2000°/s: -300°/s /0.0625 = -4800.

N.B. The linear vector sum uses the same trigger values as any other acceleration. This result is always positive.

20.1.2 Command: Setup Individual Alarms

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x6B	**ALARM NO**	**ENABLE ALARM / LOGIC**	**TYPE**	**Trigger value MSB**	**Trigger value LSB**	**Hysteresis value MSB**	**Hysteresis value LSB**
DLC = 0x08							

ALARM NO:

- 0x00 – 0x03 Alarm number to be configured

ENABLE ALARM / LOGIC:

- 0x00 = Disable this alarm
- 0x01 = Enable this alarm, logic = Less than
- 0x02 = Enable this alarm, logic = More than

TYPE:

- 0x01 = Accelerations X-Axis
- 0x02 = Accelerations Y-Axis
- 0x03 = Accelerations Z-Axis
- 0x04 = Angle X-Axis
- 0x05 = Angle Y-Axis
- 0x06 = Angle Z-Axis
- 0x07 = Linear accelerations X-Axis
- 0x08 = Linear accelerations Y-Axis
- 0x09 = Linear Accelerations Z-Axis
- 0x0A = Angular rate X-Axis
- 0x0B = Angular rate Y-Axis
- 0x0C = Angular rate Z-Axis
- 0x0D = Linear vector sum

Example 1
Set Alarm 1 to trigger for on X-axis acceleration above 1.55g with Full-Scale range of ±2g. Hysteresis will be set to 1.10g

For 1.55g send 15500 = 0x3C8C, for 1.10g send 11000 = 0x2AF8.

Byte 0	*Byte 1*	*Byte 2*	*Byte 3*	*Byte 4*	*Byte 5*	*Byte 6*	*Byte 7*
0x6B	*0x00 (Alarm 1)*	*0x02 (Enable & More Than)*	*0x01 (Type Acceleration X – Axis)*	*0x3C (MSB Trigger)*	*0x8C (LSB Trigger)*	*0x2A (MSB Hysteresis)*	*0xF8 (LSB Hysteresis)*
DLC = 0x08							

20.1.3 Command: Get acceleration alarms trip points / hysteresis

Send this command to sensor:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xEB	**ALARM NO**						
DLC = 0x02							

The sensor will reply with:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xEB	**ALARM NO**	**ENABLE ALARM / LOGIC**	**TYPE**	**Trigger value MSB**	**Trigger value LSB**	**Hysteresis value MSB**	**Hysteresis value LSB**
DLC = 0x08							

20.1.4 Command: Set Alarms Global Parameters

Enables the alarms globally, and sets weather or not the logic alarm is enabled. When the alarm is setup and it trips it can either send CAN messages, enable the Logic output, do both or if the alarm is turned off – do nothing.
The logic output can be inverted by setting the value to 0x01 so that the output is normally On until an alarm is triggered and it turns Off.
When an alarm is triggered, the output will by default stay active for 1000ms after the alarm is cleared. This is to prevent alarm pulses which are too short to trigger external equipment. This hold time can be changed by writing the Logic Hold value. If CAN bus reporting is enabled, CAN frames will be continuously sent. As with the logic output, CAN frames will continue to be sent until the alarm is cleared and the logic hold time has passed. The message frequency can be set by the CAN Delay value. By default, this is 1000ms, meaning a CAN frame will be sent each second.

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x53	**ALARM REPORTING**	**LOGIC INVERTED**	**0x00**	**Logic Hold value MSB**	**Logic Hold value LSB**	**CAN Delay value MSB**	**CAN Delay value LSB**
Default Values	**0x03**	**0x00**	**0x00**	**0x03**	**0xE8**	**0x03**	**0xE8**
DLC = 0x08 (values above 0x08 are also valid, but Data bytes are not used)							

ALARM REPORTING:

- 0x00 = Turn Off all alarms
- 0x01 = Turn On CAN Bus alarm only
- 0x02 = Turn On Logic alarm only
- 0x03 = Turn On CAN Bus & Logic alarm

LOGIC INVERTED:

- 0x00 = Logic output not inverted
- 0x01 = Logic output inverted

Example 1
Turn On CAN Bus and Logic Alarms, Logic output inverted, 1000ms hold time, and CAN frames at 10Hz (100ms delay)

Byte 0	*Byte 1*	*Byte 2*	*Byte 3*	*Byte 4*	*Byte 5*	*Byte 6*	*Byte 7*
0x53	*0x03*	*0x01*	*0x00*	*0x03*	*0xE8*	*0x00*	*0x64*
DLC = 0x02							

20.1.5 Command: Get Enable Alarms

Get the alarm setting from the sensor

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xC2							
DLC = 0x01 (values above 0x01 are also valid, but Data bytes are not used)							

Reply from sensor

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xC2	**ALARM REPORTING**	**LOGIC INVERTED**	**0x00**	**Logic Hold value MSB**	**Logic Hold value LSB**	**CAN Delay value MSB**	**CAN Delay value LSB**
DLC = 0x08							

20.1 *Alarm registers*

This CAN message is sent from the sensor when alarm trips – if this is enabled 20.1.4. One byte represents the alarm state. If the bit for the given alarm is set then the alarm is currently tripped. This register can also be requested at any given time. It is possible to turn off the alarm and manually request these bytes.

20.1.1 Command: Get alarm register

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xEE	**0x01**						
DLC = 0x04 (values above 0x04 are also valid, but Data bytes are not used)							

Reply from sensor when issuing the 0xEE command or when alarm trips and the CAN message alarm is turned On.

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xEE	**0x00**	**0x00**	**Data**				
DLC = 0x04							

	Bit	**If bit is set then the alarm is tripped**
Data0	Bit 0	Alarm 1
	Bit 1	Alarm 1
	Bit 2	Alarm 1
	Bit 3	Alarm 1
	Bit 4	N/A
	Bit 5	N/A
	Bit 6	N/A
	Bit 7	N/A

Example.
The following message is received when requesting the alarm register

Byte 0	***Byte 1***	***Byte 2***	***Byte 3***
0xEE	*0x00*	*0x40 (binary 01000000)*	*0x08 (binary 00001000)*
DLC = 0x04			

This indicates that the maximum angle on X-axis and minimum angle on Z-axis has tripped alarms.

21 Save Current Parameters in Sensor

After changing any parameter in the sensor these settings will remain unchanged until the sensor is reset. By saving the current parameters to the sensor, these parameters will be loaded at start-up.

The saved values does not include the calibration values which can be saved using command in section 24
NB: Since the parameters are stored in FLASH memory which have a limited number of erase / write cycles, the user must ensure that this command is not called more than 10.000 times within the sensor's lifetime.

Issue the following command to save the current parameters:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x50	**0xFF**						
DLC = 0x02 (values above 2 are also valid, but Data bytes are not used)							

22 Reset to Factory Settings

This will reset the sensor to its factory settings. The calibrations values will not be affected. The settings are automatically saved in memory, so it is not required to use the save command.

22.1.1 Command: Set factory settings

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x55	**0x01**	**0x52**	**0x65**	**0x74**	**0x66**	**0x61**	**0x63**
DLC = 0x08							

After sending this command, the sensor will reset and re-start itself. During this time the sensor will be unresponsive.

23 Calibrating Sensor

The sensor is calibrated using the earth gravity. The Full-scale range of ±16g must be set first. The sensor is then placed on each side and a CAN message is sent to tell the sensor to sample the current position. Only one position of each axis is required, as the scale is a fixed precision value that is not required to calibrate.

The sensor must be stable 5 seconds before calibration and min. 5 seconds after issuing the calibration command has been issued. Effects of a new calibration can be immediately seen in the data. To save the calibration data see section 24.

23.1.1 Command: Calibrate axis

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x20	**AXIS**	**Data MSB**	**Data**	**Data**	**Data LSB**		
DLC = 0x06							

AXIS:

- 0x01 = Calibrate positive X axis, ignore the Data bytes
- 0x02 = Calibrate positive Y axis, ignore the Data bytes
- 0x03 = Calibrate positive Z axis, ignore the Data bytes
- 0x04 = Calibrate negative X axis, ignore the Data bytes
- 0x05 = Calibrate negative Y axis, ignore the Data bytes
- 0x06 = Calibrate negative Z-axis, ignore the Data bytes

	X-axis	**Y-axis**	**Z-axis**
Positive	 **AXIS = 0x01**	 **AXIS = 0x02**	 **AXIS = 0x03**
Negative	 **AXIS = 0x04**	 **AXIS = 0x05**	 **AXIS = 0x06**

24 Save Current Calibration Constants

After changing calibration constants in the sensor, these constants will remain unchanged until the sensor is reset. By saving the current calibration constants to the sensor, these will be loaded at start-up.

The saved values do not include the other parameters in the sensor, which can be saved using the command in section 20.1
NB: Since the calibration constants are stored in FLASH memory which have a limited number of erase / write cycles, the user must ensure that this command is not called more than 10.000 times within the sensor's lifetime.

24.1.1 Command: Save Current Calibration Constants

Issue the following command to save the current calibration constants:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x21	**0xFF**						
DLC = 0x02 (values above 2 are also valid, but Data bytes are not used)							

Figure 6

25 Updating Sensor Firmware

The sensor firmware can be updated over the CAN bus by using the U2C programmer. New updates are continuously made available when new functionality is added or improvements are made.

Please visit www.lilliesystems.com to check for updates and the latest version of this manual.

Should you be interested in updating the firmware using your own CAN device, please contact us for a description of the protocol, and NDA, which must be signed prior to receiving the protocol.

26 Examples of Applications

26.1 *Single crane boom inclination sensing*

26.2 *Industrial machine acceleration for stress analysis*

26.3 *Cars & Trucks acceleration analysis*

26.4 *Platform stabilization*

26.5 *Chassis leveling check*

26.6 *Motion picture track system leveling analysis*

27 Error Codes

If the sensor does not understand the command it receives or if some parameter is out of range it will respond with a "Not acknowledged" message. This message cannot be sent to the sensor, only received.

The message format is a follows

27.1.1 Command: Not Acknowledged

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xFE	**CMD**	**SUB-CMD**	**ERROR_MSB**	**ERROR_LSB**			
DLC = 0x05							

CMD: returns the same command which was sent, and which the sensor does not understand
SUB-CMD: returns the same sub-command which was sent, and which the sensor does not undertand.
ERROR: Is the error message

27.2 *Error message list:*

- 0x0001 = Error Baud Rate Out Of Range.
- 0x0002 = Error Mode Out Of Range
- 0x0003 = Error Band Width Out OF Range
- 0x0004 = Error Axis Selection Out OF Range
- 0x000F = Error Must be in Angle Or Acceleration Mode To Send Angles
- 0x0010 = Error Axis Selection Velocity Distance Out Of Range
- 0x0011 = Error Set Values To Zero Out Of Range
- 0x0012 = Error Set Periodic Task Sub Command Out Of Range, // must be 0 for general or 1-4 (max nbr tasks)
- 0x0013 = Error Set Periodic Task Not Valid
- 0x0014 = Error Set Periodic Task Interval Below 2ms
- 0x0015 = Error Get Periodic Task Out Of Range
- 0x0016 = Error Set Alarm Modes Error Mode Out Of Range
- 0x0017 = Error Set CAN Custom Baud Error Mode Out Of Range
- 0x0018 = Error Set Std ID Out Of Range
- 0x0019 = Error Set IncomingFilterID1_2ID Out Of Range
- 0x001A = Error Set IncomingFilterID3_4ID Out Of Range
- 0x001B = Error Set Limits To Be Checked Out Of Range
- 0x001C = Error Get Incoming Filter ID Out Of Range
- 0x001D = Error Get Sensor Information Sub Command Out Of Range
- 0x001E = Error Save Calibration Values To Flash Sub Command Not 0xFF
- 0x001F = Error Calibrate Using Earth Gravity Sub Command Out Of Range
- 0x0020 = Error Set Default Calibration Values Sub Command Not 0xFF
- 0x0021 = Error Set Save Parameters To Flash Sub Command Not 0xFF
- 0x0023 = Error Set Output On Off Data Out Of Range
- 0x0024 = Error Command Not Valid
- 0x0025 = Error Set Factory Settings Wrong Data
- 0x0026 = Error Set Ext ID Out Of Range
- 0x0027 = Error Set CAN ID Sub command Out Of Range
- 0x0028 = Error Set Logic Output Parameters Sub Command Out Of Range
- 0x0029 = Error Limit Angle Max Hysteresis Out Of Range
- 0x002A = Error Limit Angle Min Hysteresis Out Of Range
- 0x002B = Error Limit Acceleration Min Hysteresis Out Of Range
- 0x002C = Error Limit Acceleration Max Hysteresis Out Of Range
- 0x002D = Error Set Add Subtract Tilt Sub CmdNot0x01
- 0x002E = Error Get Add Subtract Tilt Sub CmdNot0x01
- 0x002F = Error Sub command CAN Send Acceleration Out Of Range
- 0x0030 = Error Sub command CAN Send Angles All Out Of Range
- 0x0034 = Error Set Output Inverted Out Of Range, must be 0x00 or 0x01 but was set to a different value.

IMPORTANT NOTICE

Lillie Systems reserve the right to make corrections, enhancements, improvements and other changes to its products (sometimes referred to as components) and services without prior notice. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to Lillie Systems' terms and conditions of sale supplied at the time of order acknowledgment.
Lillie Systems warrants performance of its products (components) to the specifications applicable at the time of sale, in accordance with the warranty in Lillie System's terms and conditions of sale. Testing and other quality control techniques are used to the extent that Lillie Systems deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.
Lillie Systems assumes no liability for applications assistance or the design of Buyers' products. Buyers are responsible for their products and applications using Lillie System components. To minimize the risks associated with Buyers' products and applications, Buyers should provide adequate design and operating safeguards.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of Lillie Systems' components in its applications, notwithstanding any applications-related information or support that may be provided by Lillie Systems. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify Lillie Systems and its representatives against any damages arising out of the use of any Lillie Systems components in safety-critical applications.

Lillie Systems products may be promoted specifically to facilitate safety-related applications. With such components, Lillie Systems' goal is to help customers design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.