

A2C-TRI-C / A2C-TRI-M12

Features

- Selectable Bandwidth: 25Hz 50Hz 250Hz 340Hz
- Programmable number of sample averages
- Programmable alarms for acceleration on all axis
- Heartbeat CAN messages with programmable periods
- Periodic CAN messages with programmable data and time periods
- Logic alarm output (open drain) for standalone mode without the use of CAN bus
- Low power consumption
- Durable aluminum / stainless steel housing
- Software upgradable via CAN bus

Applications using CAN bus data

- Acceleration measurements on industrial machines
- Tilt measurements on industrial machines
- Low frequency machine vibration¹
- Crane boom angle measurements
- Vehicle accelerations measurements
- Mobile lift boom angle and acceleration alarms
- Wind turbine blade acceleration measurements
- Wind turbine blade movement expressed in mm (experimental)
- Rotating equipment speed measurement²

Application without CAN bus³

- Machine angle too high or low
- Machine vibration too high or low⁴
- Vehicle turn over alarm
- Mobile lift or crane chassis not level
- Wind turbine blade movement too large⁵
- Elevator vibration too high



General Description

The A2C-TRI sensor measures accelerations in 3 planes and communicates the measurements to a host via CAN Bus. The A2C-TRI contains a high-performance processor and can be programmed to report both accelerations and or tilt angles. Alarms can be programmed to trigger when a given acceleration or angle limit is passed. After initial programming the sensor can be used in a standalone mode without any CAN bus connected. On an alarm event the open drain line is activated, which can be used to drive a relay or plc input. An optional USB programmer simplifies programming and no knowledge of CAN bus communication

Specifications

- 3-Axis $\pm 2g$ MEMS sensor
- 7-30V supply voltage
- 30mA supply current
- CAN interface (2.0A & B)
- CAN driver ISO 11898 compatible
- Open drain output maximum current 500mA
- Cable or standard industrial M12 connector
- CNC machined aluminum / stainless steel housing
- Housing size 35x35x29.7mm (M12 version), 35x43x17mm (Cable version)

¹ Experimental. FFT analysis available in future firmware

² Experimental. Using FFT frequency analysis

³ The sensor must be programmed first with separate programmer

⁴ Experimental. FFT analysis available in future firmware

⁵ Experimental using AC distance calculations. Future firmware.

1 Ordering information

Part Number	Package	Interface	CAN Bus	Logic Output
A2C-TRI-M12-A-O	50x35x29mm (including connector) Anodized aluminum	M12A, 5pin Male connector.	Yes	Yes (pin 1)
A2C-TRI-M12-A-N	50x35x29mm (including connector) Anodized aluminum	M12A, 5pin Male connector.	Yes	No (pin 1 is connected to housing internally)
A2C-TRI-M12-S-O	50x35x29mm (including connector) Anodized aluminum	M12A, 5pin Male connector.	Yes	Yes (pin 1)
A2C-TRI-M12-S-N	50x35x29mm (including connector) Anodized aluminum	M12A, 5pin Male connector.	Yes	No (pin 1 is connected to housing internally)
A2C-TRI-C-A	35x43x17mm Anodized aluminum	1m cable, 4 wires and 1 shield	Yes	No
A2C-TRI-C-S	35x43x17mm 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-TRI-C / A2C-TRI-M12)
Smart Acceleration to CAN-Bus Sensors**

**Version 2.11
12th August 2019**

Document tracking control

VERSION	SECTION	CHANGED BY	DATE	CHANGE
1.00	All	JL	01-04-2012	Initial Version
1.01		JL	12-04-2013	Updated error codes, baud rate.
2.01		JL	27-05-2013	Added Sync Signal
2.02	9	JL	13-09-2013	Added Experimental LP Filter
2.03	9	JL	15-10-2013	Removed LP filter.
2.04	6+4	JL	10-04-2014	Added safety to baud rate settings. Added mounting instructions
2.05	11+19	DA	04-05-2014	New Get temperature function. Temperature calibration. Added Inverse output.
2.06	13	DA	13-07-2015	Added velocity / distance section
2.07	2.2	JL	01-09-2015	Inserted section 2.2 explaining the output
2.08	8.1	JL	12-04-2016	Added XY YZ XZ modes
2.09	1	JL	19-01-2017	A2C-TRI-C size change.
2.10	7	DA	18-08-2018	Added comments to changing baud rate
2.11	1	JL	12-08-2019	Updated drawings and added new parts numbers

Contents

1	Ordering information	3
	Document tracking control	5
2	Specifications	9
2.1	Input voltage range	10
2.2	Output switch	10
2.3	Temperature influence	10
2.4	Sensing Direction / co-ordinate system	11
3	Mechanical Drawing	12
4	Mounting	13
5	Programming tools	14
6	Protocol	15
6.1	Protocol format	15
7	Initial Setup	16
7.1	CAN Identifier	16
7.1.1	Command: Set CAN ID	16
7.1.2	Command: Get Standard CAN ID	16
7.2	Baud rate	16
7.3	Sample-point	16
7.3.1	Command: Set baud rate	16
7.3.2	Command: Get baud rate	17
7.4	Custom baud rate	17
7.4.1	Command: Set custom baud rate	17
7.4.2	Command: Get custom baud rate	18
7.5	CAN Filters	18
7.5.1	Command: Set CAN Filters	18
7.5.2	Command: Get CAN Filters	19
8	Sensor Modes	20
8.1	Acceleration and Angle mode	20
8.2	Velocity & Distance (experimental)	20
8.3	FFT mode (experimental)	20
8.4	AC mode (experimental)	20
8.4.1	Command: Set Sensor Mode	20
8.4.2	Command: Get System Mode	21
9	Setup Bandwidth	21
9.1.1	Command: Set System Bandwidth	21
9.1.2	Command: Get System Bandwidth	22
10	Syncing Data between many sensors in a network	22
10.1.1	Command: Sample Sync	22
11	Getting Accelerations	23
11.1.1	Command: Get All Accelerations	23
11.1.2	Command: Get single axis acceleration, including min / max values	23
12	Getting Angles	24

12.1.1	Command: Get All Angles	24
12.1.2	Command: Reset Global Minimum & Maximum Accelerations	24
13	Getting Velocity & Distance	25
13.1.1	Command: Get Velocity & Distance	25
13.1.2	Command: Reset Velocities & Distances	25
13.1.3	Command: Manually Set Velocity	25
14	Getting FFT Spectrum	27
14.1.1	Command: Request FFT spectrum from accelerations	27
14.1.2	Command: Set CAN timeout	27
14.1.3	Command: Get CAN timeout	27
14.1.4	Command: Set wait period between CAN messages	28
14.1.5	Command: Get wait period between CAN messages	28
15	Getting Sensor Information	29
15.1.1	Command: Get sensor information	29
16	Setting up Periodic Messages	30
16.1.1	Command: Set Periodic Messages	30
17	Setting Alarms	31
17.1	Accelerations	31
17.1.1	Command: Set acceleration alarms trip points / hysteresis	31
17.1.2	Command: Get acceleration alarms trip points / hysteresis	31
17.2	Angles	32
17.2.1	Command: Set angle alarms trip points / hysteresis	32
17.2.2	Command: Get angle alarms trip points / hysteresis	32
17.3	Distance	32
17.4	Velocity	32
17.5	AC distance	32
17.1	Enable / Disable checks	32
17.1.1	Command: Set Enable / disable checks	33
17.1.2	Command: Get Enable / disable checks	33
17.1.3	Command: Set Enable Alarms	33
17.1.4	Command: Get Enable Alarms	34
17.2	Alarm registers	35
17.2.1	Command: Get alarm register	35
17.2.2	Command: Set Alarm Logic signal minimum hold time (delayed off)	36
17.2.3	Command: Get Alarm Logic signal minimum duration (delayed off)	36
17.2.4	Command: Set delay between CAN messages on error	36
17.2.5	Command: Get delay between CAN messages on error	36
17.2.6	WAIT [0x00 – 0xFF] = Waiting time in milliseconds between CAN messages	36
18	Save Current Parameters in Sensor	37
19	Reset to Factory Settings	37
19.1.1	Command: Set factory settings	37
20	Calibrating Sensor	38
20.1.1	Command: Calibrate axis	38

21	Set Factory Calibration Values	38
21.1.1	Command: Set default calibration values	38
22	Save Current Calibration Constants	39
22.1.1	Command: Save Current Calibration Constants.....	39
23	Recovering Filter Settings	40
23.1.1	Command: Recover Filter Settings.....	40
24	Updating Sensor Firmware	41
25	Examples of Applications	42
25.1	Single crane boom inclination sensing	42
25.2	Industrial machine acceleration for stress analysis	42
25.3	Cars & Trucks acceleration analysis	42
25.4	Platform stabilization	42
25.5	Chassis leveling check	42
25.6	Motion picture track system leveling analysis	42
26	Error Codes	43
26.1.1	Command: Not Acknowledged	43
26.2	Error message list:	43

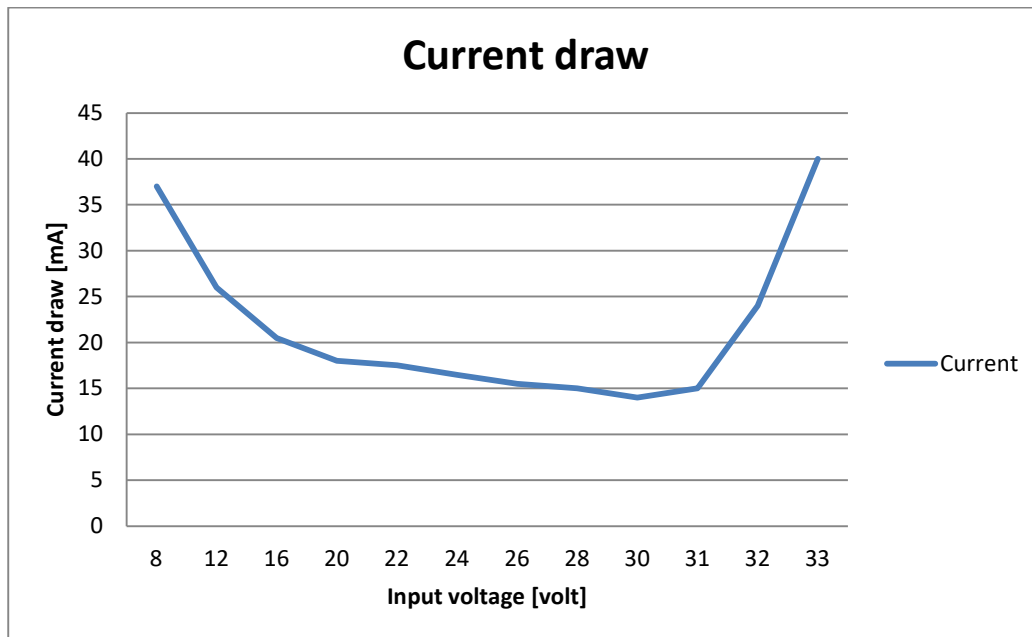
2 Specifications

Parameter	Condition	Values			Unit
SENSOR INPUT		Min	Typical	Max	
Measurement Range			± 2		g
Nonlinearity			0.005		g
Sensor Package Alignment Error			1		Deg
Inter-axis Alignment Error			1		Deg
Cross Axis Sensitivity			1		Deg
FREQUENCY RESPONSE					
Bandwidth (-3dB) ⁶		25		500	Hz
NOISE PERFORMANCE					
Noise Density All axis					
SUPPLY VOLTAGE		Min	Typical	Max	
Operating Voltage (Vin)		7		30	V
Supply Current	Vin = 24V		17		mA
Power consumption	Vin = 24V		0.4		W
Turn-On Time			500		ms
OPEN DRAIN OUTPUT					
Maximum collector emitter voltage			40		V
Maximum continuous current			500		mA
Maximum recommended inductance - including wires			10		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			0x123		
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			Chromated aluminum		
Lid Material - S suffix			Stainless steel 316		
CONNECTIVITY					
A2C-TRI-M12			M12-A Male 5 pin Connector		
A2C-TRI-C (cable length 1m)			2x2pair+1 wire + shield (PUR)		
Pin 1 / (Gray wire) = logic output					
Pin 2 / Red wire = Vin					
Pin 3 / Black wire = Ground					
Pin 4 / White wire = CAN High					
Pin 5 / Green wire = CAN Low					
Shield / Black heat shrink tubing			Connect to chassis		
DIMENSIONS		Min	Typical	Max	
Length		34,8	35	35,2	mm
Width		34,8	35	35,2	mm
Height (M12 version)		29,5	29,7	29,9	mm
Height (cable version)		12,5	12,7	12,9	mm
Weight (M12 version)			70		gram
Weight (cable version)			35		gram
TEMPERATURE					
Operating Temperature Range		-20		80	Deg
Housing temperature rise			10		Deg

⁶ Additional averaging for low frequency sensing available

User configurations	
PROGRAMMABLE ALARMS	Acceleration below set point – one per axis Acceleration above set point – one per axis Tilt angle below set point – one per axis Tilt angle above set point – one per axis
PERIODIC MESSAGES 4 selectable messages can be sent periodically at programmable periods	Acceleration from all axis (one signed 16bit value per axis) Angles from all axis (one signed 16bit value per axis with programmable resolution) Operation mode (used as simple heart beat from sensor)

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



2.1 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.

2.2 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.

2.3 Temperature influence

The sensor is calibrated at 26deg Celsius. Operating the sensor at other temperatures will cause small offsets on the measurements.

2.4 Sensing Direction / co-ordinate system

The positive acceleration directions can be seen in Figure 1. For the cable version of the A2C-TRI the positive X direction is the opposite direction of the cable entry.

The negative acceleration, act in the opposite direction as shown. Each axis is cable of measuring $\pm 2g$. The measurements include the acceleration by the earth gravity, so the axis pointing towards the earth will measure 1g. This also implies that only an additional acceleration of 1g 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.

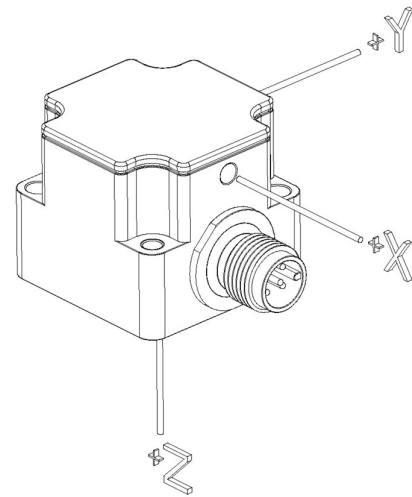


Figure 1

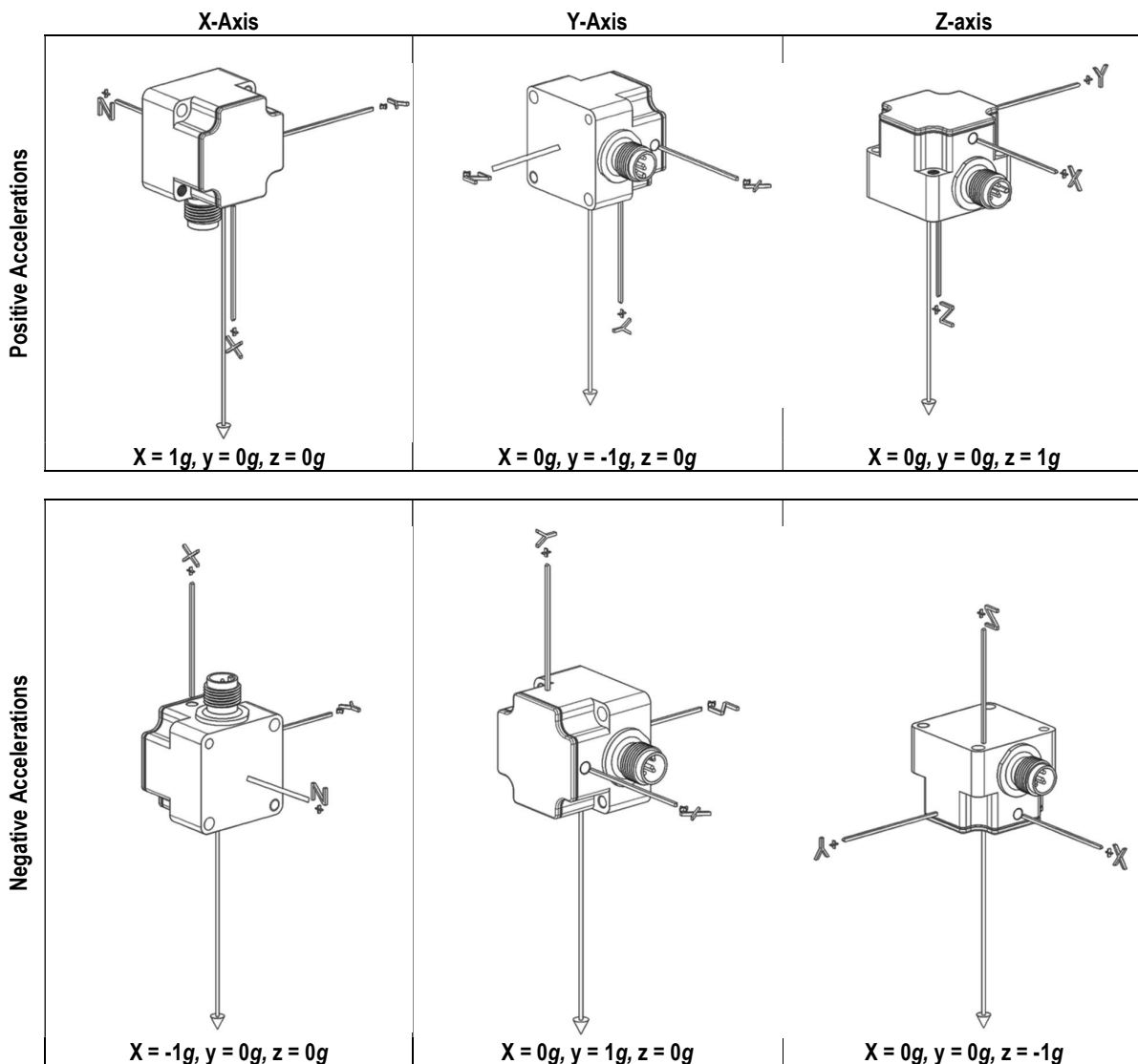
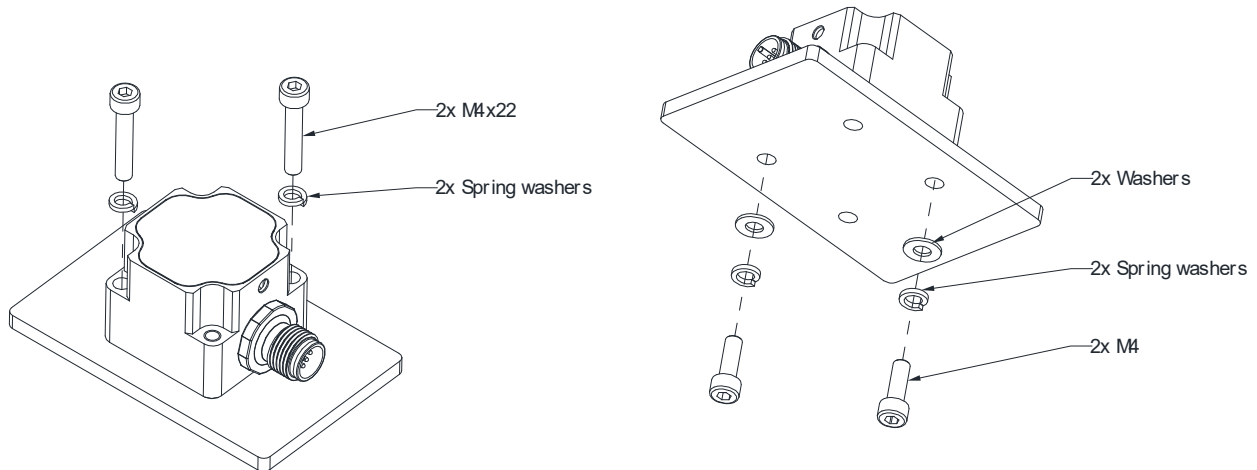


Figure 2



4 Mounting

Use two M4 bolts to secure the sensor from the top or the bottom.



5 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 3 - 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.

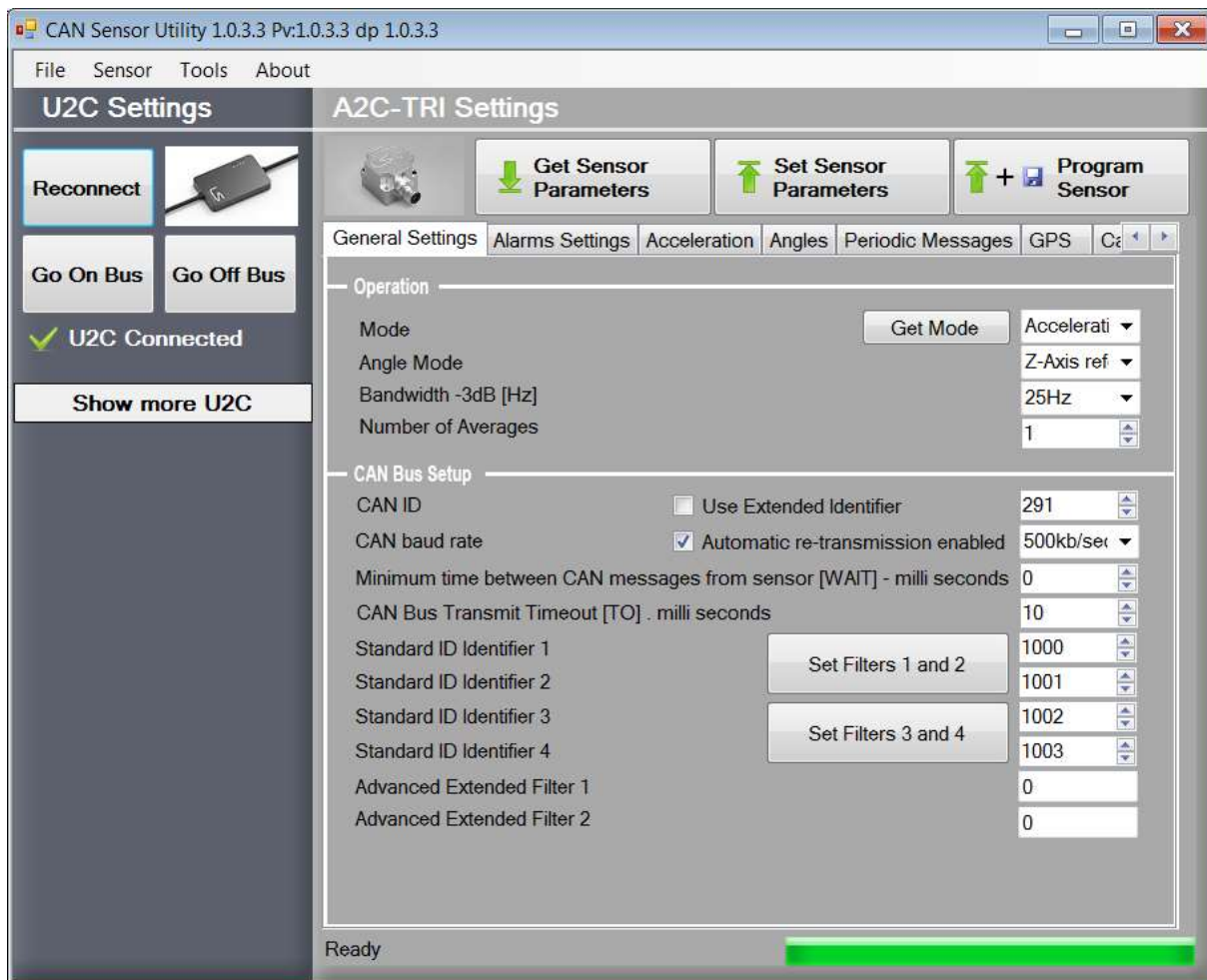


Figure 4 - Windows GUI

6 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.

6.1 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.

	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 7.1.1

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 command 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.

7 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.

7.1 CAN Identifier

The CAN Identifier is the identifier which the sensor sends when transmitting messages.
The factory default value is 0x123.

7.1.1 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)

7.1.2 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 7.1.1

7.2 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.

7.3 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 7.4

7.3.1 Command: Set baud rate

The default baud rate from the factory is 500kbits/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 but the baud rate setting is not yet saved in memory. To startup on the new baud rate, the please see the section 18. The save parameters command must be sent with the new baud rate.

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 Mega bits / second
- 0x02 = 500 Kilo bits / second
- 0x03 = 250 Kilo bits / second
- 0x04 = 125 Kilo bits / second
- 0x05 = 100 Kilo bits / second
- 0x06 = 50 Kilo bits / second
- 0x07 = Reserved for future use
- 0x08 = Reserved for future use
- 0x09 = Custom Baud rate.

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 7.3.1. This is to some degree prevent the baud rate to change and cause a CAN bus error if the filters are set incorrectly.

7.3.2 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 7.3.1

7.4 Custom baud rate

The following values must be calculated first.

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

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

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

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.

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

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

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

7.4.1 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 quanta
- 0x02 = 3 time quanta
- 0x03 = 4 time quanta

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 quanta
- ...
- 0x0F = 16 time quanta

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 quanta
- ...
- 0x07 = 8 time quanta

PRES_MSB: Specifies the MSB prescale value

PRES_LSB: Specifies the LSB prescale value

7.4.2 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 7.4.1

7.5 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

7.5.1 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							

7.5.2 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 7.5.1

8 Sensor Modes

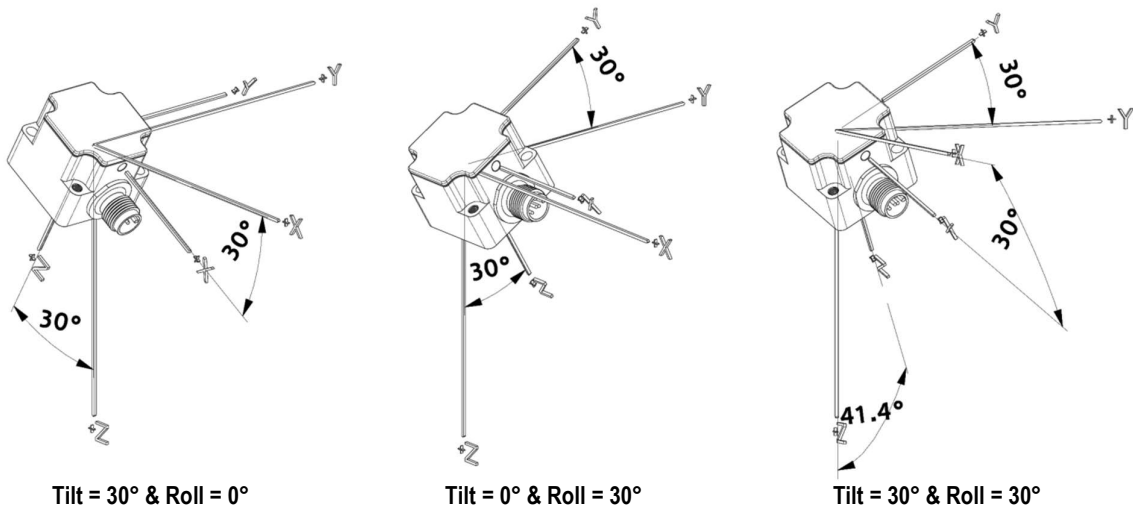
The sensor can operate in the following modes.

- Acceleration and Angle mode.
- Velocity & Distance mode
- FFT mode
- AC distance mode

8.1 Acceleration and Angle mode

In these modes the accelerations for all axes are continuously sampled and the angles continuously calculated. If enabled, the accelerations will continuously be compared to the alarm limits. Should a limit be exceeded this can be reported on the CAN bus, logic output or the alarm register can be requested on the CAN bus. Alternatively, the alarm register can be sent as a periodic message.

The global minimum and maximum accelerations are continuously updated and can be requested. When needed, the global minimum and maximum values can be reset, that is, set to the current value. This is useful if the host requesting CAN messages only want to see what maximum or minimum acceleration has been applied on the sensor since last requesting data. After the minimum & maximum accelerations have been read, the host can ask the sensor to reset the minimum & maximum values.



8.2 Velocity & Distance (experimental)

In this mode the velocity and distance are continuously calculated based on the current accelerations.

In order for this mode to work properly, the number of averages should be set to 16. See section 9.1.1.

As with most acceleration sensors, there will be some drift over time. This is due to noise in the complete signal chain, from sensor to analogue frontend, sampling and numeric rounding. To cope with this, the velocity and distance must be periodically reset.

An offset can be chosen to subtract 1g from the measurements, making it possible to detect speed and distance in an up and downwards motion. Without this offset, the sensor would calculate a free fall, as it measures 1g downwards.

8.3 FFT mode (experimental)

In FFT mode the FFT spectrum of the accelerations are continuously updated.

8.4 AC mode (experimental)

AC mode is used when measuring distance swing such as a crane boom or wind turbine blade. This mode continuously calculates the distances.

8.4.1 Command: Set Sensor Mode

To change the mode of operation send the following CAN message

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x40	MODE	SUBMODE					
DLC = 0x03 (values above 3 are also valid, but Data bytes are not used)							

MODE: Operating mode of the sensor

- 0x01 = Acceleration and Angle mode. The Sensor calculates the accelerations and angles for all axes for the chosen bandwidth.
- 0x02 = Acceleration and Angle mode. Same as mode 0x01.
- 0x03 = Velocity & Distance mode. Sensor calculates the velocity and distance for each axis.
- 0x04 = FFT Mode. The sensor calculates the FFT spectrum for each axis.
- 0x05 = AC distance mode.

SUBMODE:

When used in **Angle Mode** the following submodes are available

- 0x00 = (default) Z-axis is used as reference
- 0x01 = X-axis is used as reference
- 0x02 = Y-axis is used as reference
- 0x03 = XY Mode (Z-axis deactivated)
- 0x04 = YZ Mode (X-axis deactivated)
- 0x05 = XZ Mode (Y-axis deactivated)

When used in **Velocity & Distance Mode** the following submodes are available:

- 0x00 = (default) No acceleration offset on any axis
- 0x01 = -1.0g is subtracted from the X-axis measurement
- 0x02 = -1.0g is subtracted from the Y-axis measurement
- 0x03 = -1.0g is subtracted from the Z-axis measurement

8.4.2 Command: Get System Mode

To get the current sensor mode send the following CAN message:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xC0							
DLC = 0x01 (values above 1 are also valid, but Data bytes are not used)							

The sensor will reply:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xC0	MODE	SUBMODE	Not defined				
DLC = 0x04							

MODE & SUBMODE: See section 8.1

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]
0x64	BW	AVG MSB	AVG LSB				
DLC = 0x04 (values above 4 are also valid, but Data bytes are not used)							

BW: Bandwidth for all the accelerometers, (-3dB). All sensors are sampled simultaneously and the BW cannot be set individually for each axis. This byte must be set to one of the values given below:

- BW = 0-14 are reserved for future use, do not set to these values.

- BW = 15 will set bandwidth to 25Hz
- BW = 16 will set bandwidth to 50Hz
- BW = 17 will set bandwidth to 250Hz
- BW = 18 will set bandwidth to 340Hz

AVG: Number of sample averages. This increases sensor performance in low frequency angle / tilt measurement application.

- LP = 1-1024 is possible.

It should be noted that the 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.

Example.

"Set sensor to 25Hz Bandwidth, and use 4 number of averages". Send the following CAN Bus package:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x64	0x0F	0x00	0x04				
DLC = 0x4							

9.1.2 Command: Get System Bandwidth

To get the current sensor bandwidth send the following CAN message:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xE4							
DLC = 0x01 (values above 1 are also valid, but Data bytes are not used)							

The sensor will reply:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xE4	BW	AVG MSB	AVG LSB				
DLC = 0x04							

The BW and AVG being the same as when setting the bandwidth in section 9.1.1

10 Syncing Data between many sensors in a network

It is possible to send a Sync command to all sensors on the CAN network, which will then save their current value in memory. The value can then be requested from each sensor, and all values will be synchronized.

10.1.1 Command: Sample Sync

Send this command to sensor:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x10	VAL						
DLC = 0x02 (values above 2 are also valid, but Data bytes are not used)							

VAL: determines which values are synced

- 0x01 = Acceleration values are synced
- 0x02 = Angle values are synced
- 0x03 = Both Acceleration and Angle values are synced.

See section 11.1.1 and section 12.1.1 for commands that retrieve the synchronized values.

11 Getting Accelerations

To receive accelerations the mode must be set to acceleration or angle mode.

The accelerations from the sensor are sent as integer values multiplied with 10000. This means that a value received of 15520 must be divided by 10000, thus being 1.5520g

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 12.1.2.

11.1.1 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 if current values or a previous synced value is used

- 0x00 = Get current values, which are continuously updated
- 0x03 = Get a previously synced value. Please see the Sync command in section 10.1.1

After the 0x0A and a sub command have been received, the sensor will return the accelerations for all 3 accelerometers.

Reply:

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							

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

Send this command to sensor:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x0C	AXIS						
DLC = 0x02 (values above 2 are also valid, but Data bytes are not used)							

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

12 Getting Angles

To receive angles the mode must be set to acceleration or angle mode.

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

There are two commands to receive angles. The first command gets all 3 axes at the same time.

12.1.1 Command: Get All Angles

Send this command to sensor:

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

RET:

- 0x00 = Angle result is rounded to nearest integer value i.e. 85= 85deg
- 0x01 = Angle result is multiplied by 10 e.g. a value of 854 = 85,4deg
- 0x02 = Angle result is multiplied by 100 e.g. a value of 8542 = 85,42deg
- 0x03 = Previously Synced Angle result is rounded to nearest integer value i.e. 85= 85deg See section 10.1.1 for syncing.
- 0x04 = Previously Synced Angle result is multiplied by 10 e.g. a value of 854 = 85,4deg. See section 10.1.1 for syncing.
- 0x05 = Previously Synced Angle result is multiplied by 100 e.g. a value of 8542 = 85,42deg. See section 10.1.1 for syncing.

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

The sensor will reply with:

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							

12.1.2 Command: Reset Global Minimum & Maximum Accelerations

In angle and acceleration mode the global minimum and maximum accelerations 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.
- 0x05 = see velocity and distance mode
- 0x06 = see velocity and distance mode
- 0x07 = see velocity and distance mode

13 Getting Velocity & Distance

To receive Velocity and Distances the mode must be set to Velocity & Distance Mode.

The velocities / distances from the sensor are sent as signed 32bit integers multiplied. The actual values have been multiplied with 10000. The unit for velocity is in meters per second (m/s). The unit for distance is given in meters. This means that a velocity received of 15520 must be divided by 10000, thus being 1.5520m/s etc.

13.1.1 Command: Get Velocity & Distance

Send this command to sensor:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x0E	RET						
DLC = 0x02 (values above 2 are also valid, but Data bytes are not used)							

RET:

- 0x01 = Velocity X-axis
- 0x02 = Velocity Y-axis
- 0x03 = Velocity Z-axis
- 0x04 = Distance X-axis
- 0x05 = Distance Y-axis
- 0x06 = Distance Z-axis

After this command has been received, the sensor will return the requested data

The sensor will reply with:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x0E	RET	MSB Value	Value	Value	LSB Value		
DLC = 0x06							

13.1.2 Command: Reset Velocities & Distances

In Velocity & Distance mode the values can be reset (set to 0).

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:

- 0x05 = Reset velocity value for all axis
- 0x06 = Reset distance values for all axis
- 0x07 = Reset velocity and distance values for all axis

13.1.3 Command: Manually Set Velocity

In Velocity & Distance mode the velocity value can be set manually if it is known.

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x0F	VALUE	MSB Velocity	Velocity	Velocity	LSB Velocity		
DLC = 0x06 (values above 0x02 are also valid, but Data bytes are not used)							

VALUE:

- 0x0A = Set velocity X-axis
- 0x0B = Set velocity Y-axis
- 0x0C = Set velocity Z-axis

Velocity must be sent as a 32 bit signed integer multiplied with 10000.

Example.

"To set the velocity of the Z axis to -15m/s, Send the following CAN Bus package:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x0F	0x0C	0xFF	0xFD	0x86	0x10		
DLC = 0x06							

The velocity is sent at -150000.

Important information:

Please remember to set the number of averages to 16 before using the Velocity and Distance mode. A higher or lower number will also work, but has not been tested.

14 Getting FFT Spectrum

This section is currently still in experimental stages, meaning that there may still be bugs in the sensor software and this documentation. Please make sure you have the latest datasheet and firmware for your sensor.

In order to use the FFT functions of the sensor, the FFT mode must first be set, see 8.3.

14.1.1 Command: Request FFT spectrum from accelerations

Send this (these) command(s) to sensor

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x1A	AXIS						
DLC = 0x02 (values above 2 are also valid, but Data bytes are not used)							

AXIS: Only the FFT spectrum of a single axis can be requested per time. The sub command tells the sensor which axis is requested. It must be one of the following values

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

After this command has been received, the sensor will return the non aliased part of the spectrum i.e. 512 samples. Each sample is represented by an unsigned 16bit variable thus 1024 bytes are sent. The total transmission consists of 171 CAN messages, each with DLC of 8bytes. The last CAN message contains the two last FFT samples and (Data[0]->Data[3]) and the last two bytes are a 16bit CRC value, so the total FFT can be checked for errors.

Reply – PACN (CAN message) 0-169

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x1A	PACN	MSB sample a	LSB sample a	MSB sample b	LSB sample b	MSB sample c	LSB sample c
DLC = 0x08							

PACN: Package number. There are a total of 171 packages, each with 6bytes containing the FFT.

Reply – PACN (CAN message) = 170

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x1A	170	MSB sample a	LSB sample a	MSB sample b	LSB sample b	MSB CRC	LSB CRC
DLC = 0x08							

14.1.2 Command: Set CAN timeout

This setting only applies to FFT sending

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x66	TO						
DLC = 0x02 (values above 2 are also valid, but Data bytes are not used)							

TO: Timeout in ms. Default [0x20 = 32ms]

- 0x00 – 0xFF = timeout in ms that the sensor tries to send a message on the CAN Bus when sending the FFT. The timeout is independent of the baud rate. The timeout is also independent of the CAN bus fault system which will go off bus if the Transmit error counter reaches 255.

14.1.3 Command: Get CAN timeout

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xE6							
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]
0xE6	TO						
DLC = 0x02							

- **TO:**
[0x00 – 0xFF] = Timeout in milliseconds

14.1.4 Command: Set wait period between CAN messages

This setting only applies to FFT sending

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x65	WAIT						
DLC = 0x02 (values above 2 are also valid, but Data bytes are not used)							

WAIT: Time in milliseconds between CAN messages. Default [0x00 = 0ms]

- 0x00 – 0xFF = The time the sensor will wait between CAN messages. This is to make sure that the host has time to process the information from the sensor, especially when sending multiple packages such as the FFT requests.

NB: Using a large WAIT value will effectively halt the sensor from doing other work while it is transmitting. The lowest possible value should be used.

14.1.5 Command: Get wait period between CAN messages

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xE5							
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]
0xE5	WAIT						
DLC = 0x02							

WAIT [0x00 – 0xFF] = Waiting time in milliseconds between CAN messages

15 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

15.1.1 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)
- 0x30 = Internal Temperature (starting from firmware version 2.65)

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							

16 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:

- acceleration from all axis; Command: 0x0A
- tilt angles from all axis; Command: 0x0B
- average, minimum and maximum acceleration from a single axis; Command: 0x0C
- average, minimum and maximum angles from a single axis; Command: 0x0D
- velocity and distance from a single axis; Command: 0x0E
- Get system mode (can be used as a heart beat); Command: 0xC0

A maximum of 4 periodic messages can be setup at the same time – 4 tasks. Each message can be sent with an interval ranging from 2ms (500 messages per second) up to 65535ms.

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 1-4. 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.

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.**

16.1.1 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 1-4
- **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.

17 Setting Alarms

Alarms are used to monitor accelerations and angles. They can be setup to trigger in an event of a given acceleration or angle becoming too large or small. A total of 12 alarms can be set with different trigger points. For each alarm a hysteresis can be set, e.g. an alarm is set to trigger at 45deg angle on the X-axis. The hysteresis can then be set to e.g. 43deg. This means that the alarm will trigger at 45deg and only stop once the angle has been decreased to 43deg. For each alarm the minimum alarm output time can be set. Alarm output time is the time the alarm will keep sending data or hold the logic output. For instance, the sensor triggers an alarm which only lasts for 0.2seconds, however, setting the alarm output time to 0.5seconds keeps the output logic (and CAN bus alarm messages) on for 0.5seconds.

A minimum of 4 messages are required to setup the alarm; 1) trip point, 2) hysteresis value, 3) enable check, 4) enable alarm type.

- Trip point defines the value at which the alarm trips. Commands are **0x6B** and **0x6C**
- Hysteresis value defines the value at which the alarm turns off after tripping. Commands are **0x6B** and **0x6C**
- Enable checks defines which checks are performed. Command **0x6A**
- Enable alarm type defines what happens when an alarm is tripped: Logic output turns on / CAN bus message is sent. Command **0x53**

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. If a CAN message is sent this message is the same message as in

17.1 Accelerations

Acceleration trip points can be set between -1.999g and 1.999g.

17.1.1 Command: Set acceleration alarms trip points / hysteresis

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x6B	TYPE	X axis acceleration MSB	X axis acceleration LSB	Y axis acceleration MSB	Y axis acceleration LSB	Z axis acceleration MSB	Z axis acceleration LSB
DLC = 0x08							

TYPE:

- 0x01 = Accelerations More than
- 0x02 = Accelerations Less than
- 0x03 = Accelerations More than, hysteresis
- 0x04 = Accelerations Less than, hysteresis

NB: The acceleration alarm values must be sent as g multiplied with 1000

Example 1

Set the acceleration alarm to trigger for X-axis acceleration above 1.5g, Y-axis above 1.6g and Z-axis above 1.9g
For 1.5g send 1500 = 0x05DC, for 1.6g send 1600 = 0x0640 for 1.9g send 1900 = 0x076C

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x6B	0x01	0x05	0xDC	0x06	0x40	0x07	0x6C
DLC = 0x08							

17.1.2 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	TYPE						
DLC = 0x02							

The TYPE is the same as for setting the alarms as seen in 17.1.1

The sensor will reply with:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xEB	TYPE	X axis acceleration MSB	X axis acceleration LSB	Y axis acceleration MSB	Y axis acceleration LSB	Z axis acceleration MSB	Z axis acceleration LSB
DLC = 0x08							

The TYPE is the same as for setting the alarms as seen in 17.1.1

17.2 Angles

Angle trip points can be set between -89.9deg and 89.9deg.

17.2.1 Command: Set angle alarms trip points / hysteresis

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x6C	TYPE	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							

TYPE:

- 0x01 = Angle More than
- 0x02 = Angle Less than
- 0x03 = Angle More than, hysteresis
- 0x04 = Angle Less than, hysteresis

NB: The angle alarm values must be sent as *degrees* multiplied with 10

Example 1

Set the angle alarm to trigger for X-axis angle above 15deg, Y-axis above 20deg and Z-axis above 45deg

For 15deg send 150 = 0x0096, for 20deg send 200 = 0x00C8 for 45deg send 450 = 0x01C2

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x6C	0x01	0x00	0x96	0x00	0xC8	0x01	0xC2
DLC = 0x08							

17.2.2 Command: Get angle alarms trip points / hysteresis

Send this command to sensor:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xEC	TYPE						
DLC = 0x02							

The TYPE is the same as for setting the alarms as seen in 17.2.1

The sensor will reply with:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xEC	TYPE	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							

The TYPE is the same as for setting the alarms as seen in 17.2.1

17.3 Distance

The distance alarms will be available in future versions of the firmware

17.4 Velocity

The velocity alarms will be available in future versions of the firmware

17.5 AC distance

The AC distance alarms will be available in future versions of the firmware

17.1 Enable / Disable checks

This is the same as enabling / disabling the individual alarms. Alarm trip points can be setup beforehand and dynamically enabled / disabled with a short simple CAN message, as seen below.

17.1.1 Command: Set Enable / disable checks

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x6A	0x01	Data0	Data1				
DLC = 0x04 (values above 0x04 are also valid, but Data bytes are not used)							

	Bit	If bit is set then the alarm will be On
Data0	Bit 0	Check maximum acceleration: X-axis
	Bit 1	Check maximum acceleration: Y-axis
	Bit 2	Check maximum acceleration: Z-axis
	Bit 3	Check minimum acceleration: X-axis
	Bit 4	Check minimum acceleration: Y-axis
	Bit 5	Check minimum acceleration: Z-axis
	Bit 6	Check maximum angle: X-axis
	Bit 7	Check maximum angle: Y-axis
Data1	Bit 0	Check maximum angle: Z-axis
	Bit 1	Check minimum angle: X-axis
	Bit 2	Check minimum angle: Y-axis
	Bit 3	Check minimum angle: Z-axis
	Bit 4	Reserved for future functions
	Bit 5	Reserved for future functions
	Bit 6	Reserved for future functions
	Bit 7	Reserved for future functions

Example.

“Enable checks on angles on X-axis (min / max)” Send the following CAN package:

Byte 0	Byte 1	Byte 2	Byte 3
0x6A	0x01	0x40 (binary 01000000)	0x02 (binary 00000010)
DLC = 0x04			

17.1.2 Command: Get Enable / disable checks

To get the values sent in 17.1.1

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xEA							
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]
0xEA	0x02	Data0	Data1				
DLC = 0x04							

The received Data0 & Data 1 has the same format as in 17.1.1

17.1.3 Command: Set Enable Alarms

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x53	ALARM						
DLC = 0x02 (values above 0x02 are also valid, but Data bytes are not used)							

ALARM:

- 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

Example 1

Turn On CAN Bus and Logic Alarms

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
--------	--------	--------	--------	--------	--------	--------	--------

0x53	0x03						
DLC = 0x02							

17.1.4 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						

DLC = 0x02 (values above 0x02 are also valid, but Data bytes are not used)

ALARM being the same as in section 17.1.3

17.2 Alarm registers

This CAN message is sent from the sensor when alarm trips – if this is enabled in 17.2.1. Two bytes 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. It is also possible to set a periodic task to send this message. See section 16.1.1

17.2.1 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	Data0	Data1				
DLC = 0x04							

	Bit	If bit is set then the alarm is tripped!
Data0	Bit 0	Maximum acceleration: X-axis
	Bit 1	Maximum acceleration: Y-axis
	Bit 2	Maximum acceleration: Z-axis
	Bit 3	Minimum acceleration: X-axis
	Bit 4	Minimum acceleration: Y-axis
	Bit 5	Minimum acceleration: Z-axis
	Bit 6	Maximum angle: X-axis
	Bit 7	Maximum angle: Y-axis
Data1	Bit 0	Maximum angle: Z-axis
	Bit 1	Minimum angle: X-axis
	Bit 2	Minimum angle: Y-axis
	Bit 3	Minimum angle: Z-axis
	Bit 4	Reserved for future functions
	Bit 5	Reserved for future functions
	Bit 6	Reserved for future functions
	Bit 7	Reserved for future functions

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.

17.2.2 Command: Set Alarm Logic signal minimum hold time (delayed off)

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x51	VAR	OnOff / Time MSB Invert Output	Time LSB				
DLC = 0x04 (values above 0x04 are also valid, but Data bytes are not used)							

VAR:

- 0x01 = logic output test.
 - OnOff = 0x01 will turn on the alarm output for the durations specified below. Used for testing Logic output.
 - OnOff = 0x00 will turn off the alarm output. Used for testing Logic output.
- 0x02 = minimum hold time of Logic output in milliseconds
 - Time MSB & Time LSB. Values 0x0000 - 0xFFFF are valid. When a value of 0x0000 is used, the logic output will turn off as soon as the alarm event is over. For other values the output will turn off when the alarm event is over and the hold time has expired.
- 0x03 = reserved
- 0x04 = Invert logic alarm output.
 - Invert Output = 0x01 This will cause the output to be inverted, so when there is no alarm the output will be held low. When an alarm occurs it will release the output.
 - Invert Output = 0x00 Output is not inverted

17.2.3 Command: Get Alarm Logic signal minimum duration (delayed off)

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xC4	0x02						
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]
0xC4	0x02	Time MSB	Time LSB				
DLC = 0x04							

The received Time MSB and Time LSB is the minimum hold time in milliseconds, as explained in 0

17.2.4 Command: Set delay between CAN messages on error

If the Alarm in 17.1.3 is set to send a CAN bus message upon tripping then these messages will continue to be sent from the sensor. It is possible to set how often the messages are sent by sending the following CAN message

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x6D	0x01	DELAY_MSB	DELAY_LSB				
DLC = 0x04 (values above 4 are also valid, but Data bytes are not used)							

DELAY: Delay in milliseconds between CAN messages on alarm trip. Default [0x0A = 10ms]

- 0x00 – 0xFF = The time the sensor will wait between CAN messages. This is to make sure that the host has time to process the information from the sensor, especially when sending multiple packages such as the FFT requests.

17.2.5 Command: Get delay between CAN messages on error

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xED							
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]
0xED	WAIT						
DLC = 0x02							

17.2.6 WAIT [0x00 – 0xFF] = Waiting time in milliseconds between CAN messages

18 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 0

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)							

19 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.

19.1.1 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.

20 Calibrating Sensor






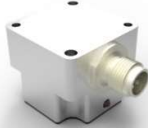
The sensor is calibrated using the earth gravity. The sensor is placed on all 6 sides and for each side a CAN message is sent to tell the sensor to sample the current position. Make sure that the sensor is completely stable and level when calibrating. The sensor must be stable 3 seconds before calibration and min. 3 seconds after issuing the calibration command has been issued.

20.1.1 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
- 0x10 = Calibration temperature. Data represents the ambient Celsius temperature of the sensor . For 26 deg send the value 0x0000001A.

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

21 Set Factory Calibration Values

Should a calibration have been performed, saved and the sensor for some reason be difficult to calibrate, then it is possible to go back to the factory calibration. The command can be sent and the sensor can be checked before sending the command to save the calibrations.

21.1.1 Command: Set default calibration values

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x22	0xFF						
DLC = 0x02							

It is necessary to issue the command to save calibration constants as seen in 0 if the factory values will be used.

22 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 does not include the other parameters in the sensor, which can be saved using the command in section 18

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.

22.1.1 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)							

23 Recovering Filter Settings

Should the CAN filters have been set, saved, and its values forgotten, it will be impossible to connect to the sensor. It is also not possible to set the default values. To rescue the filter settings, use the following procedure.

1. Turn off sensor
2. Setup a CAN bus device to transmit at a Baud rate of 75kbits/sec.
3. Turn on the sensor
4. Send the following message with a CAN ID of 0x7FF as soon as the sensor is powered up. The sensor will look for this message 50ms after power up.

23.1.1 Command: Recover Filter Settings

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
(char) 'R'	(char) 'e'	(char) 'c'	(char) 'o'	(char) 'v'	(char) 'e'	(char) 'r'	(char) '1'
DLC = 0x08							

The sensor will reply with the filter values:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x03	0xFF	CAN ID MSB	CAN ID LSB	STDFIL1MSB	STDFIL1LSB	STDFIL2MSB	STDFIL2LSB
DLC = 0x08							

- CAN ID is the CAN bus ID
- STDFIL1 is the standard filter number 1 value
- STDFIL2 is the standard filter number 2 value

The U2C and programming tool can be used to recover the filter values as seen in Figure 5. Press the "Recover Filter 1,2" button and then apply power to the sensor.

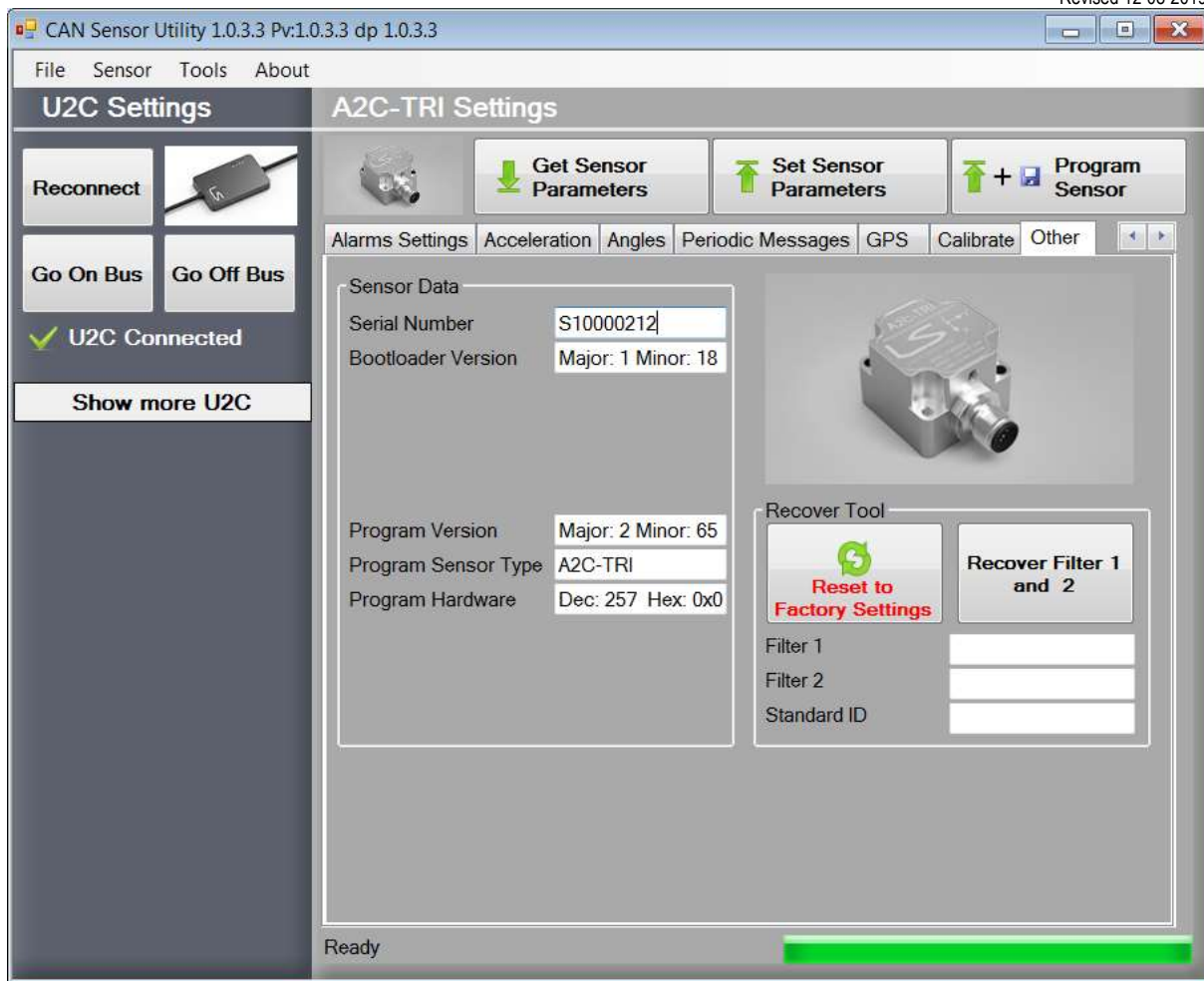


Figure 5

24 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.

25 Examples of Applications

- 25.1 *Single crane boom inclination sensing*
- 25.2 *Industrial machine acceleration for stress analysis*
- 25.3 *Cars & Trucks acceleration analysis*
- 25.4 *Platform stabilization*
- 25.5 *Chassis leveling check*
- 25.6 *Motion picture track system leveling analysis*

26 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 as follows

26.1.1 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 understand.

ERROR: Is the error message

26.2 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
- 0x0005 = Error Limit Acceleration Max Out Of Range
- 0x0006 = Error Limit Acceleration Min Out Of Range
- 0x0007 = Error Limit Sub Command Acceleration Out Of Range
- 0x0008 = Error Limit Angle Max Out Of Range
- 0x0009 = Error Limit Angle Min Out Of Range
- 0x000A = Error Limit Sub Command Angle Out Of Range
- 0x000B = Error Get Delay Between CAN Messages On Error Out Of Range
- 0x000C = Error Set Delay Between CAN Messages On Error Out Of Range
- 0x000D = Error Get Limits Accelerations Min Max Out Of Range
- 0x000E = Error Get Limits Angle Min Max 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
- 0x0022 = Error Enter Boot loader Data Not Valid
- 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 All is not 0x00 or 0x03.
- 0x0030 = Error Sub command CAN Send Angles All is not 0x00 to 0x05
- 0x0031 = Error Sub command Sample Sync Out Of Range. Should be 0x01 to 0x03.
- 0x0032= Error GPS Command Sub Cmd 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.