LILLIE SYSTEMS

Dual strain gauge amplifier
with CAN bus

A2C-SG2-M12

Features

- Complete strain gauge signal conditioning
- Programable analogue gain
- Differential analogue frontend
- Selectable excitation voltage 2.5V / 5V
- Up to 4800 samples per second
- 24-bit ADC
- Periodic CAN messages with programmable data types
- 32 taps user programable FIR filter
- Low power consumption
- Durable aluminum / stainless steel housing
- Software upgradable via CAN bus
- Signal to noise ratio calculation

General Description

The A2C-SG2-M12 Strain gauge amplifier uses state of the art digital and analogue components to achieve exceptional sensitivity and high band width.

Specifications

- Selectable 2.5V / 5V Strain gauge supply
- 11-30V supply voltage
- 70mA supply current
- CAN interface (2.0A & B)
- CAN driver ISO 11898 compatible
- Logic output (versions made to order)
- Standard industrial M12 connector
- CNC machined aluminum / stainless steel housing

1 Ordering information

Part Number	Package	Interface	CAN Bus	Logic Output
A2C-SG2-M12-A	Black anodized aluminum	M12, 5pin Male connector for CAN Bus 2 x M12, 5pin Female for strain gauges	Yes	No
A2C-SG2-M12-S	Stainless steel 316	M12, 5pin Male connector for CAN Bus 2 x M12, 5pin Female for strain gauges	Yes	No

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

Specifications for

A2C-SG2-M12 Dual strain gauge amplifier to CAN bus

Version 1.12
8th May 2024

Document tracking control

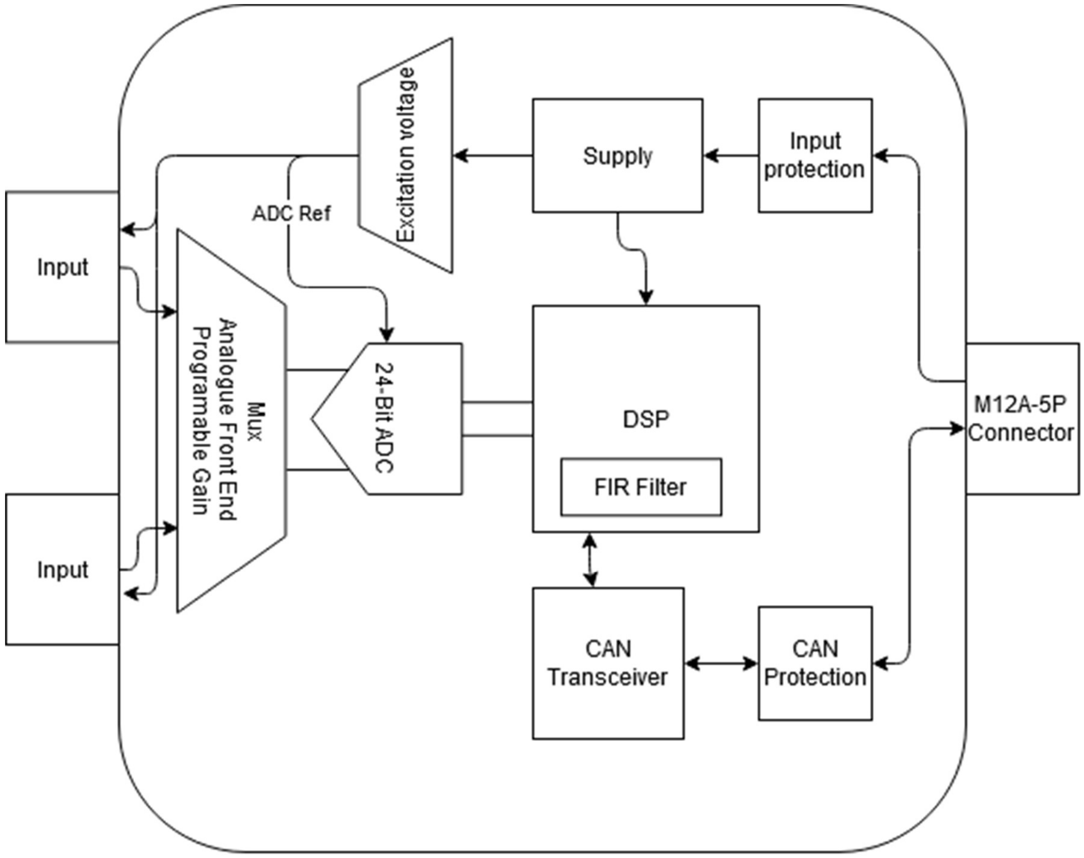
VERSION	SECTION	CHANGED BY	DATE	CHANGE
1.00	All	JL	03-08-2013	Initial Version
1.01		JL	12-02-2014	Updated error codes, baud rate, added SNR
1.02	All	JL	01-02-2015	Added periodic messages follow ADC. Released for public viewing
1.03	9	JL	26-06-2015	Added Integer Scalling section, Corrected CAN follow ADC wrong values for ints and floats
1.04	All	JL	12-05-2017	New commands for periodic messages from firmware
1.05	All	WN	15-01-2018	0x0107 and onwards. New examples.
1.06	17.1.1 12 2	JL	17-01-2019	Corrected wrong value to reset factory settings. Added more examples of float to byte array for calibration. Added data to specifications
1.07	12	JL	19-03-2019	Corrected float to array error
1.08	All	WN	20-03-2019	Gramma and typo corrections
1.09	2	JL	20-07-2019	Removed old A2C-SG drawing
1.10	All	JL	07-07-2020	Added FIR Filter and SNR sections
1.11	All	JL	29-07-2021	Corrected custom baud-rate calculations Added command for default calibration values Added section From measurement to CAN Value Updated mechanical drawing
1.12	2	JL	08-05-2024	Updated specifications

Contents

2 Specifications

Parameter	Condition	Values			Unit
SUPPLY VOLTAGE		Min	Typical	Max	
Operating Voltage (Vin)		11		32	V
Supply Current	Vin = 24V		22		mA
Power consumption	Vin = 24V		0.53		W
Turn-On Time			1500		ms
STRAIN GAUGE INPUTS					
Number of input channels			2 x differential		
Programmable gain		1		128	
Data rate				4800	Hz
Bandwidth (-3dB)		0.3		1445	Hz
Resolution			24		bits
RMS noise			8.5		nV
Differential input voltage range (Bipolar setup)			±Excitation voltage/gain		V
Analog input current					nA
Input current drift			±3		pA/°C
Excitation voltage 5V		4.95	5	5.05	V
Excitation voltage 2.5V		2.475	2.5	2.525	V
Excitation current – all excitation voltages				150	mA
Recommended strain gauge impedance (5V)			350		Ω
Recommended strain gauge impedance (2.5V)			150		Ω
NPN OUTPUT					
Maximum Collector Emitter Voltage			40		V
Maximum continuous current			75		mA
CAN BUS 2.0A & B					
Transceiver delay loop time				150	ns
Baud rate		10		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			Black anodized Aluminum		
Housing Body Material –S suffix			Stainless steel 316		
Lid Material - A suffix			Black anodized aluminum		
Lid Material - S suffix			Stainless steel 316		
CONNECTIVITY			M12-A Male 5 pin Connector		
Pin 1 = Shield					
Pin 2 = Vin					
Pin 3= Ground (both power and CAN ground)					
Pin 4= CAN High					
Pin 5 = CAN Low					
INPUTS x 2			M12-A Female 5 pin Connector		
Pin 1 = Positive Input					
Pin 2 = Negative Input					
Pin 3= Positive Excitation					
Pin 4= Negative Excitation					
Pin 5 = Shield					
DIMENSIONS (Including base plate)		Min	Typical	Max	
Length		80.2	80.6	81	mm
Width		59.6	60	60.4	mm
Height		26.2	27	27.8	mm
Weight			190		gram
TEMPERATURE					
Operating Temperature Range		-20		65	Deg
Housing temperature rise			3		Deg

2.1 *Functional block diagram*



2.2 *ADC performance*

2.2.1 **RMS noise (nV) vs. Gain and data rate**

Data Rate Filter value	Output data rate [Hz]	Settling time [ms]	Gain=1	Gain=8	Gain=16	Gain=32	Gain=64	Gain=128
1023	4.7	639.4	270	42	23	13.5	10.5	9
640	7.5	400	320	50	27	17	13	11.5
480	10	300	350	60	35	19	15	12.5
96	50	60	1000	134	86	50	35	29
80	60	50	1050	145	95	55	40	32
32	150	20	1500	225	130	80	58	50
16	300	10	1950	308	175	110	83	73
5	960	3.125	4000	590	330	200	150	133
2	2400	1.25	56,600	7000	3500	1800	900	490
1	4800	0.625	442,000	55,000	28,000	14,000	7000	3450

2.2.2 **Effective resolution**

Data Rate Filter value	Output data rate [Hz]	Settling time [ms]	Gain=1	Gain=8	Gain=16	Gain=32	Gain=64	Gain=128
1023	4.7	639.4	24	24	24	24	24	23
640	7.5	400	24	24	24	24	23.5	22.5
480	10	300	24	24	24	24	23.5	22.5
96	50	60	23.5	23	23	22.5	22	21.5
80	60	50	23	23	22.5	22.5	22	21
32	150	20	22.5	22.5	22	22	21.5	20.5
16	300	10	22.5	22	22	21.5	21	20
5	960	3.125	21.5	21	21	20.5	20	19
2	2400	1.25	17.5	17.5	17.5	17.5	17.5	17.5
1	4800	0.625	14.5	14.5	14.5	14.5	14.5	14.5

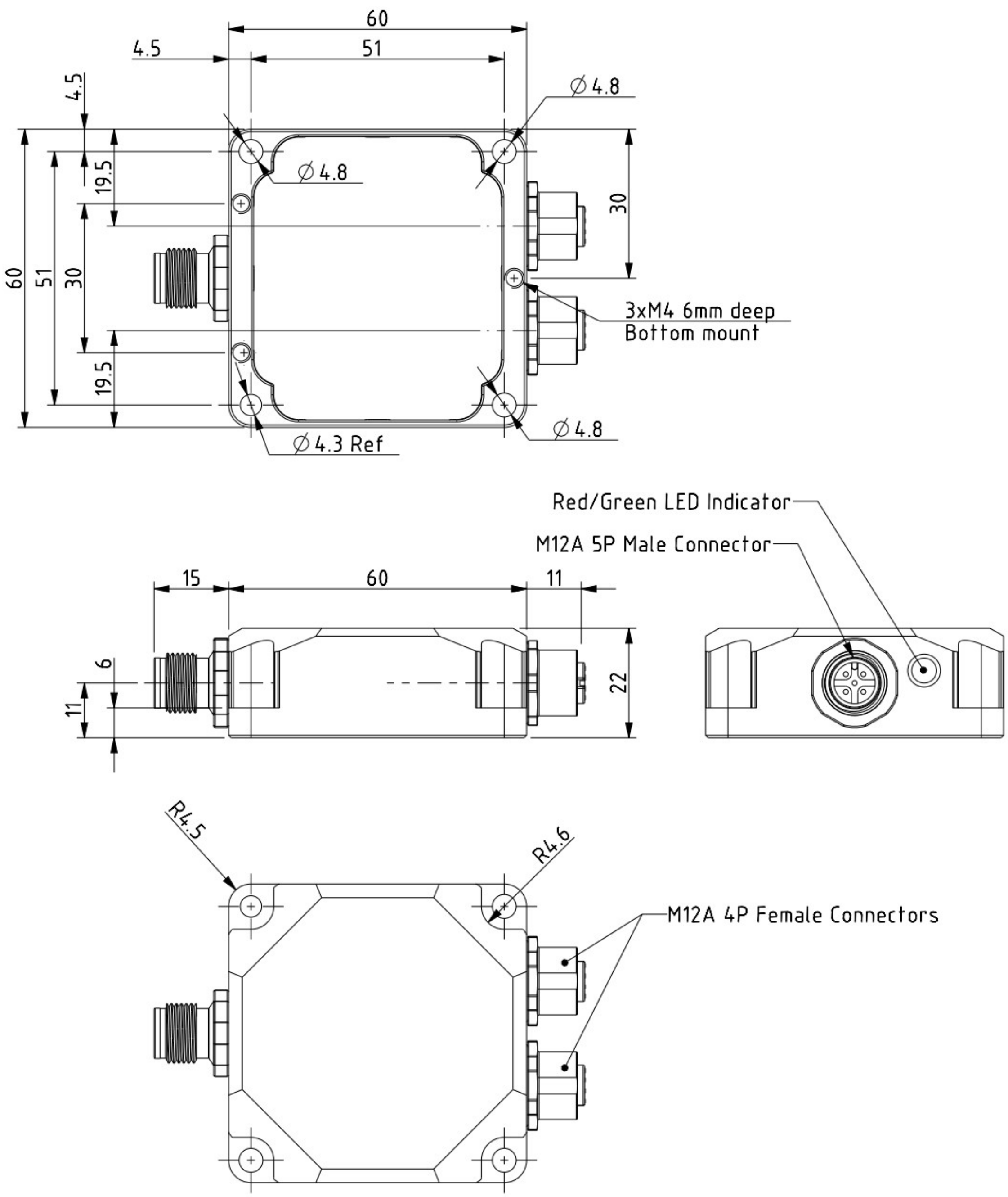
2.3 *Conformity*

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

2.4 *Strain gauge pin outs*

Please refer to the marking on the A2C-SG2 for how to connect it.

3 Mechanical Drawing



3.1 *Mounting*

Mount the sensor on an even surface. Use up to 4x M4 bolts to secure the sensor from the top or 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 1 - U2C Programming tool

The window GUI enables the user to easily set most parameters in the amplifier, and in real time see the data.

The A2C firmware must be version 1.0.3.4 or above.

5 Getting Started

Things you will need:

1. A2C-SG2-M12 - strain gauge amplifier with CAN Bus
2. U2C - Programming tool
3. Strain gauge bridge
4. Power supply
5. Internet connection

Start by downloading the latest version of this manual. Please go to www.lilliesystems.com or click here if you are using an electronic copy.

Attach your preferred strain gauge bridge by soldering the wires directly to the A2C-SG. Use a gauge with a minimum impedance of 350Ω, as the excitation voltage is 5V. Using a 120Ω gauge will cause excessive heat in the gauge, which can lead to inaccurate measurements.

Download the latest version of the A2C Sensor Utility software. Please go to www.lilliesystems.com/u2c/ or click here
If you have already installed the A2C Sensor Utility software, then this will update itself after being restarted. If you do not get any updates, please make sure that you have internet access and that you restart the program 2 times.

Connect the A2C to the T connector. Connect the power supply through the M12 connector which also contains a 120Ohm terminating resistor. Connect the U2C to the other side of the T connector. Turn on the power supply and watch the LED on the back of the A2C-SG2-M12 turn red and green.

Make sure the drivers for the U2C have been installed, as described in the separate manual. Connect the U2C to the PC and start the A2C Sensor Utitlity program. From the top menu bar, choose Sensor and click on the A2C-SG.

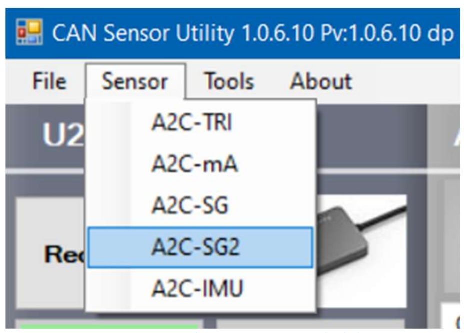


Figure 2 - Choose the A2C-SG

Next, press the Go On Bus button

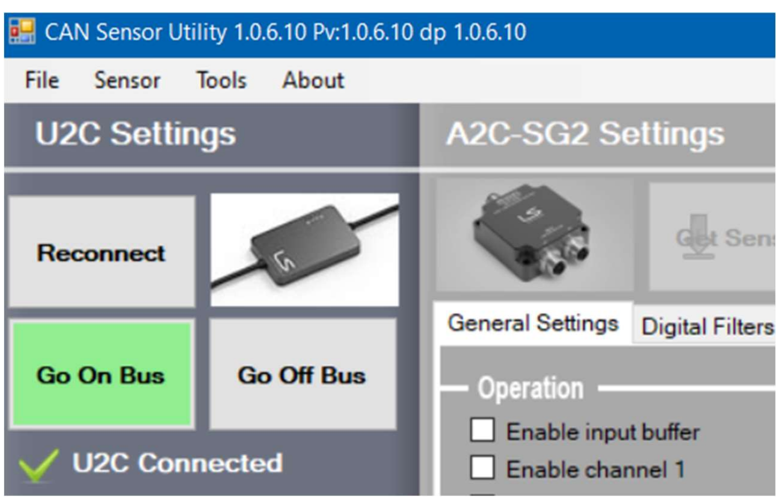


Figure 3 Go On Bus to enable communication

To parse all the parameters in the sensor to the A2C Import Utitlity program, press the Get Sensor Parameters

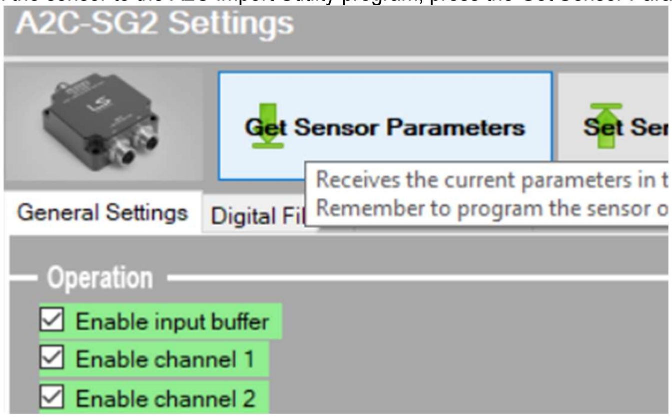


Figure 4 Get sensor parameters

If everything has been connected correctly, the parameters will be updated in the A2C Import Utility program.

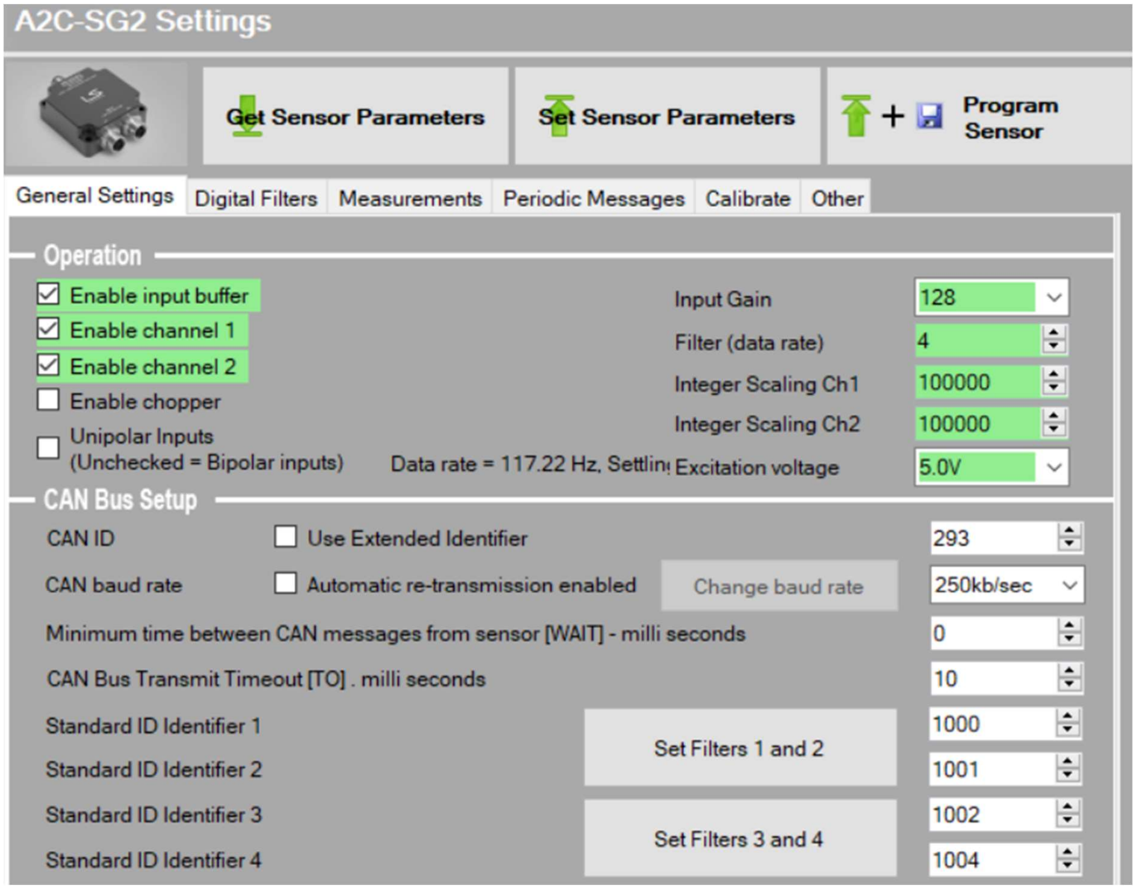


Figure 5 ADC setup

To change a value, first change the value, followed by pressing the "Set Sensor Parameters" button. All the values changed will be sent to the A2C-SG. To save the values, so that the A2C-SG uses the values at start up, press the "Program Sensor" button. If this button is not press, the A2C-SG will use the previously saved values after a restart (power cycle).

To view real time measurement values, click on the "Measurements" tab. Next click the "Start Receiving" button.

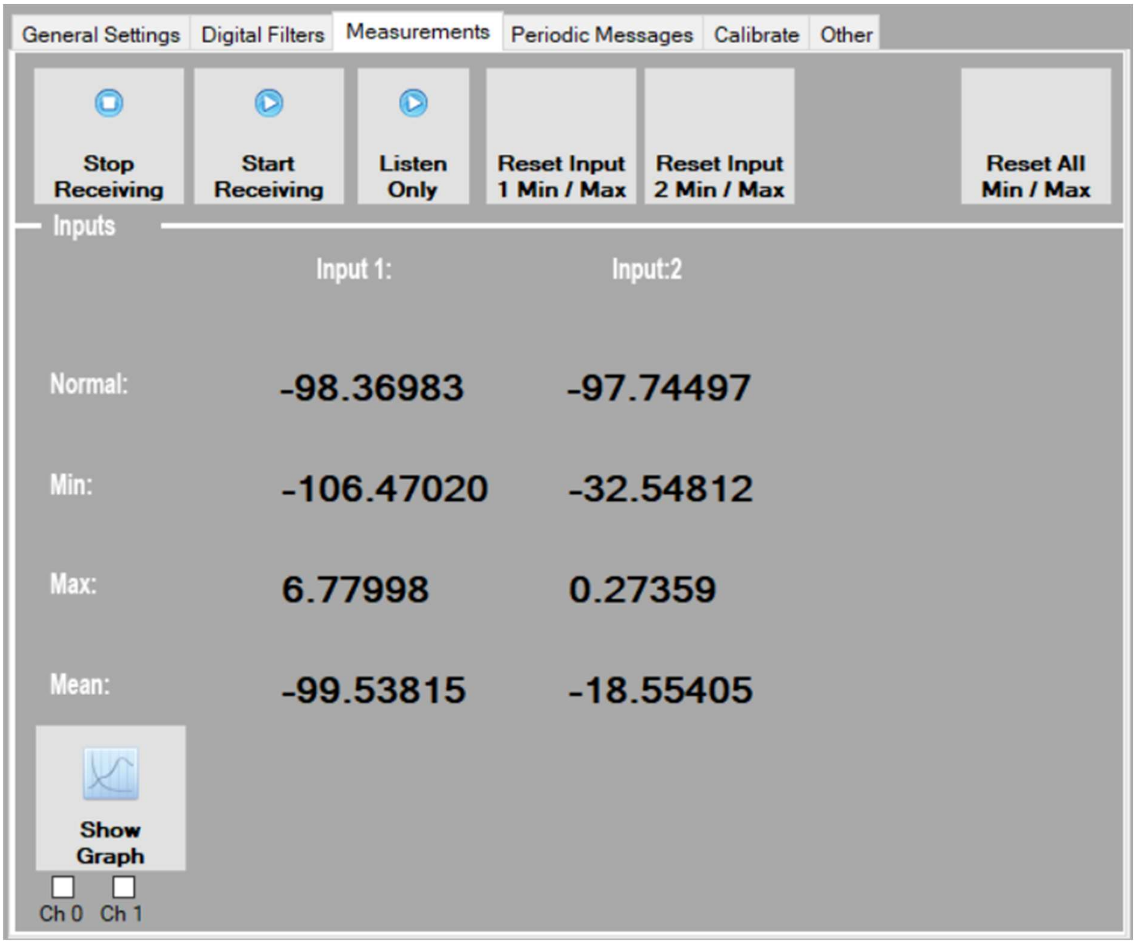


Figure 6 Measurements

The normal, min, max and mean values are displayed on the screen.
If the update rate seems slow, then change the update interval (Tools - Options - General). A value of 100ms or 50ms is suitable, but lower values are possible if the PC is powerful enough.

To view a graph of the data, press the Ch 0 or Ch 1 check boxes, followed by the "Show Graph" button. The values displayed are the normal values. Figure 7 shows example data from a load cell being compressed and tensioned by hand.

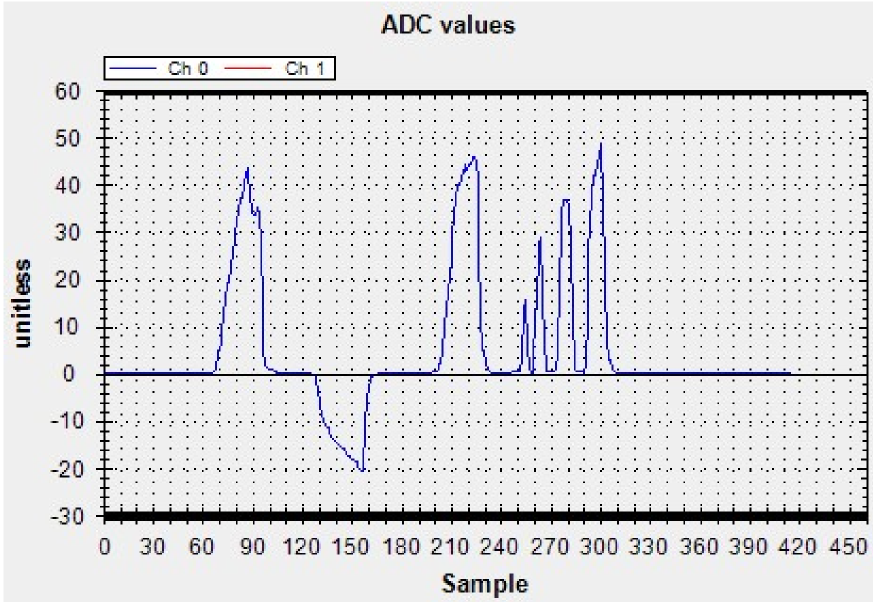
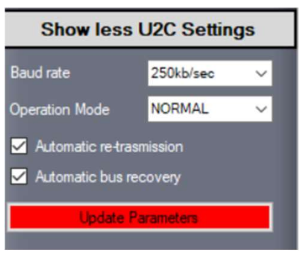


Figure 7 Graphing data

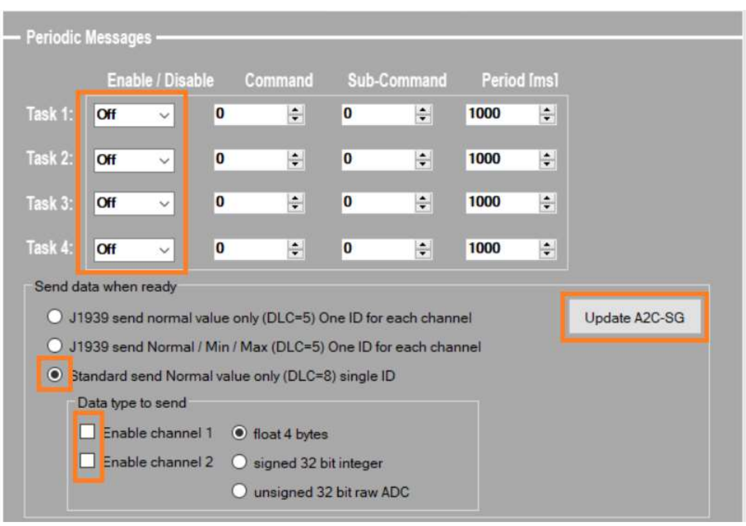
5.1 *Trouble shooting connection*

If the message “No Sensor found” appears try the following.

1. Make sure the U2C baud rate is correct. Press the Show more U2C Settings on the left pane to open up the settings for the U2C.



2. Disable periodic messages.
   a. Go to the Periodic Messages tab and disable all tasks.
   b. Press the Set Sensor Parameters – this will disable standard periodic messages
   c. Click the radio button “Standard send Normal value only”
   d. Unselect the check boxes “Enable channel 1” and “Enable channel 2”
   e. Click “Update A2C-SG”
3. Press the Get Sensor Parameters again to get the parameters.



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 0x125 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 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.

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 0x125.

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/s up to 100m, 250kbits/s up to 250m and 50kbits/s up to 1000m. These values can vary and depends on number of stubs etc.

7.2.1 Command: Set baud rate

The default baud rate from the factory is 500kbits/s. To change the baud rate to another value send the following message:

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

BAUD: See table below for valid values (firmware lower than 3.35 only support messages from 1-9)

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

7.2.2 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 7.2.1

7.3 *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 62.5kbits / second: We first select a prescale (PRES) value that creates an even T1 number. 36 is selected. The sample point is chosen to be 75%

$$\frac{36 \times 10^6}{62500 \times 32} = 16 = T1$$

$$16 \times 0.75 - 1 = 11 = BS1 - satisfy\ a\ value\ less\ than\ 16, ok!$$

$$16 - 11 - 1 = 4 = BS2 - satisfy\ a\ value\ less\ than\ 8, ok!$$

7.3.1 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:

- 0x01 = 1 time quantum
- 0x02 = 2 time quantums
- 0x03 = 3 time quantums
- 0x04 = 4 time quantums

BS1: Specifies the number of time quanta in Bit Segment 1. This parameter can be a value of

- 0x01 = 1 time quantum
- 0x02 = 2 time quantums
- …
- 0x10 = 16 time quantums

BS2: Specifies the number of time quanta in Bit Segment 2. This parameter can be a value of

- 0x01 = 1 time quantum
- 0x02 = 2 time quantums
- …
- 0x08 = 8 time quantums

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

7.3.2 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 7.3.1

7.4 *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

7.4.1 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							

7.4.2 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 7.4.1

8 ADC setup

The A2C-SG offers a very flexible frontend and ADC, that can be configures to various measurement applications.
The frontend contains a low noise, high precision 24bit sigma delta ADC with programmable gain amplifier and multiplexer.

8.1 *Bipolar / Unipoloar configuration*

The analog input to the A2C-SG can accept either unipolar or bipolar input voltage ranges. A bipolar input range does not imply that the part can tolerate negative voltages with respect to sense. For strain gauge applications the Bipolar mode should be used.

When the gain stage is enabled, the output from the buffer is applied to the input of the programmable gain array (PGA).
Even at high gain of 128 the A2C-SG still maintaining excellent noise performance. For example, when the gain is set to 128, the rms noise is 8.5 nV, typically, when the output data rate is 4.7 Hz. This is equivalent to 23 bits of effective resolution or 20.5 bits of noise-free resolution. The A2C-SG can be programmed to have a gain of 1, 8, 16, 32, 64, and 128. Therefore, the bipolar input voltage ranges are from ±19.53 mV to ±2.5 V with excitation voltage of 2.5V.

Mode	Buffer	Gain=1	Gain=8	Gain=16	Gain=32	Gain=64	Gain=128
2.5V excitation voltage							
Bipolar	Enabled	±1250mV	±156.25mV	±78.13mV	±39.06mV	±19.53mV	±9.76mV
	Disabled	±2.5V	±312.5mV	±156.2mV	±78.13mV	±39.06mV	±19.53mV
Unipolar	Enabled	+1250mV	+156.25mV	+78.13mV	+39.06mV	+19.53mV	+9.76mV
	Disabled						
5V excitation voltage							
Bipolar	Enabled	±3750mV	±468.75mV	±234.36mV	±117.19mV	±58.59mV	±29.29mV
	Disabled	±5.0V	±625mV	±312.5mV	±156.2mV	±78.13mV	±39.06mV
Unipolar	Enabled	+3750mV	+468.75mV	+234.36mV	+117.19mV	+58.59mV	+29.29mV
	Disabled						

Table 1

9 From Measurement to CAN value

From Figure 8 the flow of data is explained.
Once the signal is sampled the calibration constants are applied and from then on floating points are used. Min/Max/RMS calculations are done after the FIR filter.

The ADC has 24 bits making the highest value 0xFFFFFF. The maximum voltage is limited by the excitation voltage and also by the input buffer. See Table 1 for input ranges vs excitation and gain values.

In bipolar mode when the ADC inputs are shorted (0 input voltage), the ADC output is half the ADC range (0x80000). A negative differential voltage will result in values between 0- 0x7FFFFF, and positive voltages between 0x80000-0xFFFFFF.

After sampling the calibration constants are applied. From the factory the result is calibrated in such a way that the minimum ADC input results in -100, and maximum input = 100. This is useful in strain gauge and load cell applications as both tension and compression are output with zero strain resulting in 0.

After calibration the FIR filter can be enabled. The Minimum/Maximum and RMS values are calculated afterwards. Before being sent on the CAN Bus the integer scaling is applied.

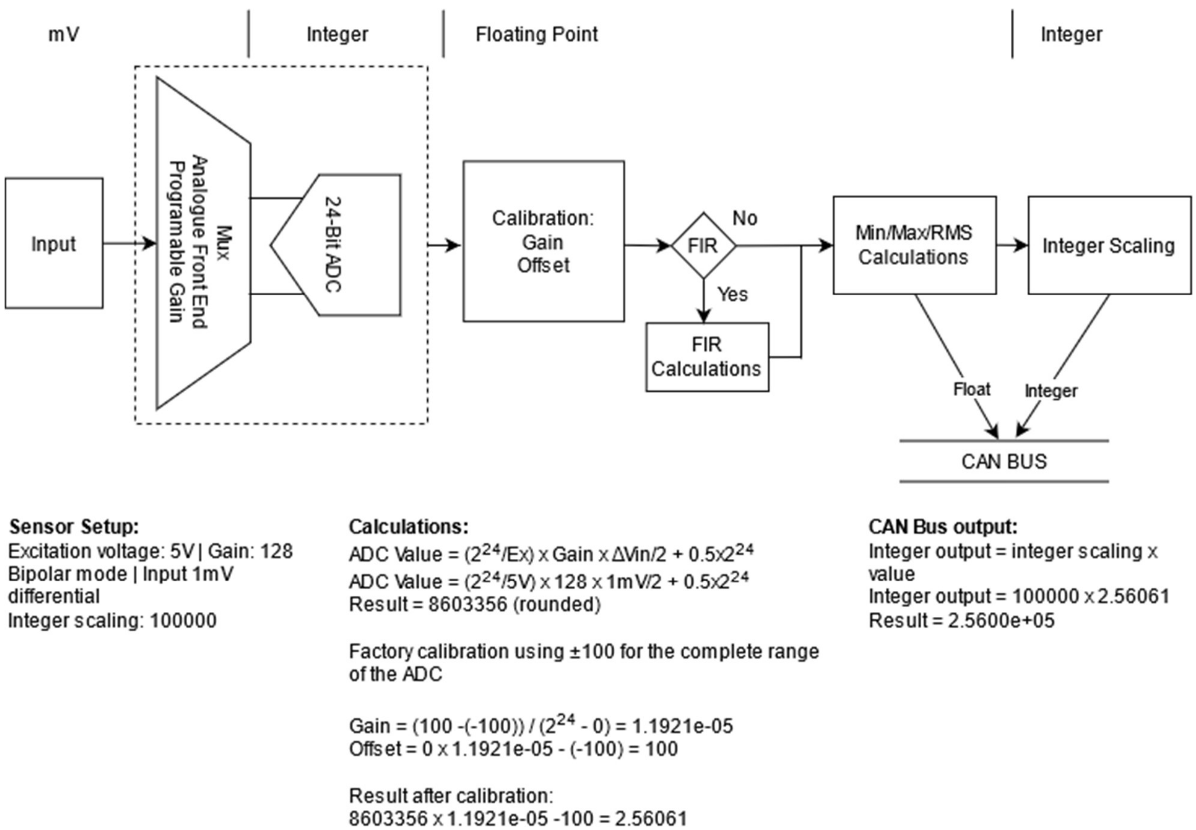


Figure 8 – data flow

Δ

9.1.1 Command: Setup ADC

The global minimum, maximum and mean values 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]
0x40	**Channels**	**Polarity**	**Gain**	**Data Rate Filter MSB**	**Data Rate Filter LSB**	**Chop Enabled**	**Buffer Enabled**
DLC = 0x08							

Channels:

- 0x01 = Use only channel 1
- 0x02 = Use only channel 2
- 0x03 = Use both channel 1 & 2

Polarity:

- 0x00 = Bipolar inputs (default for strain gauge bridges)
- 0x01 = Unipolar inputs

Gain:

- 0x01 = Analogue amplifier gain = 1
- 0x08 = Analogue amplifier gain = 8
- 0x10 = Analogue amplifier gain = 16
- 0x20 = Analogue amplifier gain = 32
- 0x40 = Analogue amplifier gain = 64
- 0x80 = Analogue amplifier gain = 128 (recommended for strain gauge bridges)

Data Rate Filter: Determines the digital filter and thereby the data rate of the A2C-SG. The lower the value the higher the data rate and consequently higher noise. When chopping is disabled, a data rate value of 1 in single channel mode will give an output rate of 4700 samples per second. In dual channel mode this falls to 437 samples per second for each channel. A value of 10 results in 533 and 52 samples per second respectfully.

- 0x0001 - 0x03FF.

Chop Enabled: chopping is a way to almost eliminate offset voltage errors on the expense of lower data rate. It also improves the RMS noise, as part of the chopping is done by averaging. For a single channel measurement and a data rate value of 96, the sample frequency becomes 12.5Hz, compared to 50Hz without chopping enabled.

- 0x00 = chopping disabled
- 0x01 = chopping enabled

Buffer Enabled: If enabled, the input channel feeds into a high impedance input stage of the buffer amplifier. Therefore, the input can tolerate significant source impedances and is tailored for direct connection to external resistive-type sensors such as strain gauges or resistance temperature detectors (RTDs).

- 0x00 = buffer disabled
- 0x01 = buffer enabled

Example 1
Setup ADC for a strain gauge bridge on both channels, with chop and buffer enabled and 10Hz sample rate for each channel. The data rate is set to 30 - 0x001E

Byte 0	***Byte 1***	***Byte 2***	***Byte 3***	***Byte 4***	***Byte 5***	***Byte 6***	***Byte 7***
0x40	*0x03 - both channels*	*0x00 - bipolar*	*0x80 - gain of 128*	*0x00*	*0x1E*	*0x01 chop enabled*	*0x01 buffer enabled*
DLC = 0x08							

Example 2
Setup ADC for a strain gauge bridge on both channels, with chop and buffer enabled and 0.5Hz sample rate for each channel. The data rate is set to 605 - 0x025D

Byte 0	***Byte 1***	***Byte 2***	***Byte 3***	***Byte 4***	***Byte 5***	***Byte 6***	***Byte 7***
0x40	*0x03 - both channels*	*0x00 - bipolar*	*0x80 - gain of 128*	*0x02*	*0x5D*	*0x01 chop enabled*	*0x01 buffer enabled*
DLC = 0x08							

9.1.2 Get ADC mode

To get the current ADC mode this command can be sent

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xC0							
DLC = 0x01							

Reply:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x0C	**Channels**	**Polarity**	**Gain**	**Data Rate MSB**	**Data Rate LSB**	**Chop Enabled**	**Buffer Enabled**
DLC = 0x08							

The format follows the same as for sending parameters by command 0x40

9.1.3 A note on data rate and data requested

If the data rate has a high value, this will result in longer sampling times. Data requested on the CAN Bus between samples will return the last sampled value. To overcome this problem, the periodic messages can be setup, so that a CAN message is sent when the ADC has a new sample ready. Changing the data rate then directly change the frequency of the CAN Bus messages.

10 Excitation voltage selection setup

The A2C-SG offers a selectable excitation voltage of 2.5V or 5V

10.1.1 Set excitation voltage:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x41	**Voltage**						
DLC = 0x02							

Voltage:

- 0x00 = Use 5V excitation
- 0x01 = Use 2.5V excitation
- 0x02 = Turns off Excitation voltage. This is useful for measuring the SN ratio.

10.1.2 Get excitation voltage:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xC6							
DLC = 0x01							

Reply:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xC6	**Voltage**						
DLC = 0x02							

The format follows the same as for sending parameters by command 0x41

11 Setting up FIR filter

The A2C-SG2 features a user programable FIR

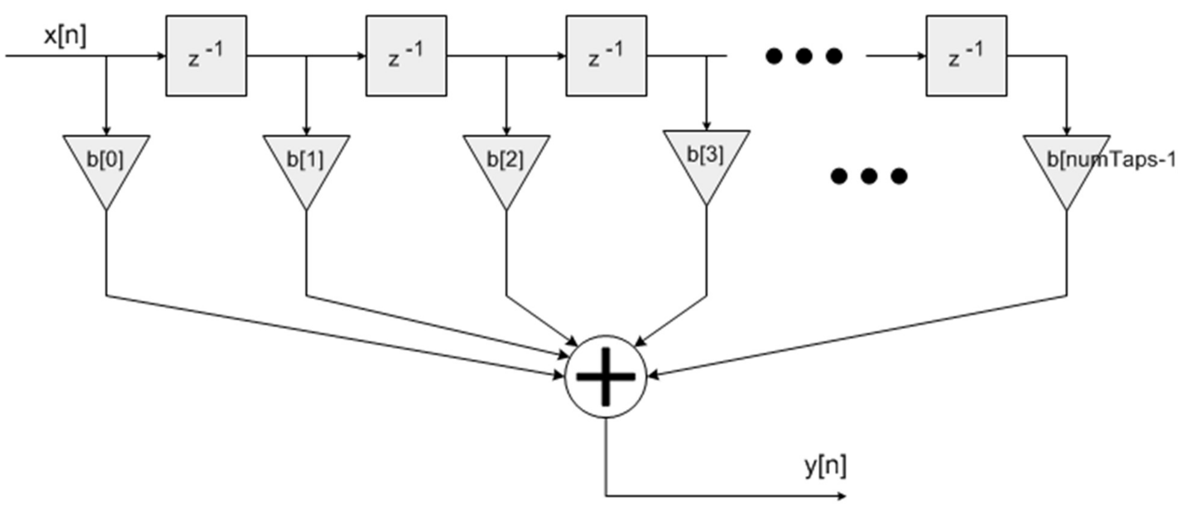
When the FIR filter is enabled it will process the calibrated values from the sensor.

FIR features

- Selectable number of taps, up to 32.
- Different coefficients and taps on both channels
- Floating point coefficients easily developed with MATLAB / Octave
- Floats are using Big Endian

The FIR filter algorithm is based upon a sequence of multiply-accumulate (MAC) operations. Each filter coefficient b[n] is multiplied by a state variable which equals a previous input sample `x[n]`.

```
y[n] = b[0]*x[n]+b[1]*x[n-1]+b[2]*x[n-2]+...+b[numTaps-1]*x[n-numTaps+1]
```



The coefficients points to a coefficient array of size numTaps. Coefficients are stored in ***time reversed order***.

```
{b[numTaps-1], b[numTaps-2], b[N-2], ..., b[1], b[0]}
```

11.1.1 Command: Set FIR parameters

Send this command:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x44	**Channel**	**EN**	**TAPS**				
DLC = 0x04 (values above 0x04 are also valid, but Data bytes are not used)							

Channel: is the channel number that is requested and must be one of the following values

- 0x00 = channel 1
- 0x01 = channel 2

EN: Enable filter

- 0x00 = Filter disabled (bypassed)
- 0x01 = Filter enabled

TAPS: Number of taps used. Number to be between 1 and 32

11.1.2 Command: Set FIR Coefficients

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x45	**Channel**	**INX**	**RESV**	**COEFF**	**COEFF**	**COEFF**	**COEFF**
DLC = 0x08							

Channel:

- 0x00 = Calibrate channel 1

- 0x01 = Calibrate channel 2

INX: Coefficient index. Value must be between 0 and 31.
RESV: Reserved byte for future use, and can have any value.
COEFF: The floating formatted using big endianness. Data[2] thus contains the sign bit and 7 exponent bits.

Example 1: Setting channel 1, coefficient index 1 to a value of 5000.0f

Byte 0	*Byte 1*	*Byte 2*	*Byte 3*	*Byte 4*	*Byte 5*	*Byte 6*	*Byte 7*
0x45	*0x00 (channel 1)*	*0x01*	*Reserved*	*0x45*	*0x9C*	*0x40*	*0x00*
DLC = 0x08							

Example 2: Setting channel 2, coefficient index 31 to a value of -5000.0f

Byte 0	*Byte 1*	*Byte 2*	*Byte 3*	*Byte 4*	*Byte 5*	*Byte 6*	*Byte 7*
0x45	*0x01 (channel 2)*	*0x1F Last Index 31*	*Reserved*	*0xC5*	*0x9C*	*0x40*	*0x00*
DLC = 0x08							

11.1.3 Command: Get FIR parameters

Send this command:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xD4	**Channel**						
DLC = 0x02 (values above 0x02 are also valid, but Data bytes are not used)							

Channel: is the channel number that is requested and must be one of the following values

- 0x00 = channel 1
- 0x01 = channel 2

Reply:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xD4	**Channel**	**EN**	**TAPS**				
DLC = 0x04							

EN: Enable filter

- 0x00 = Filter disabled (bypassed)
- 0x01 = Filter enabled

TAPS: Number of taps used.

11.1.4 Command: Get FIR Coefficients

Command		Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xD5		**Channel**	**INX**					
	DLC = 0x03							

Channel:

- 0x00 = Calibrate channel 1
- 0x01 = Calibrate channel 2

INX: Coefficient index. Value must be between 0 and 31.

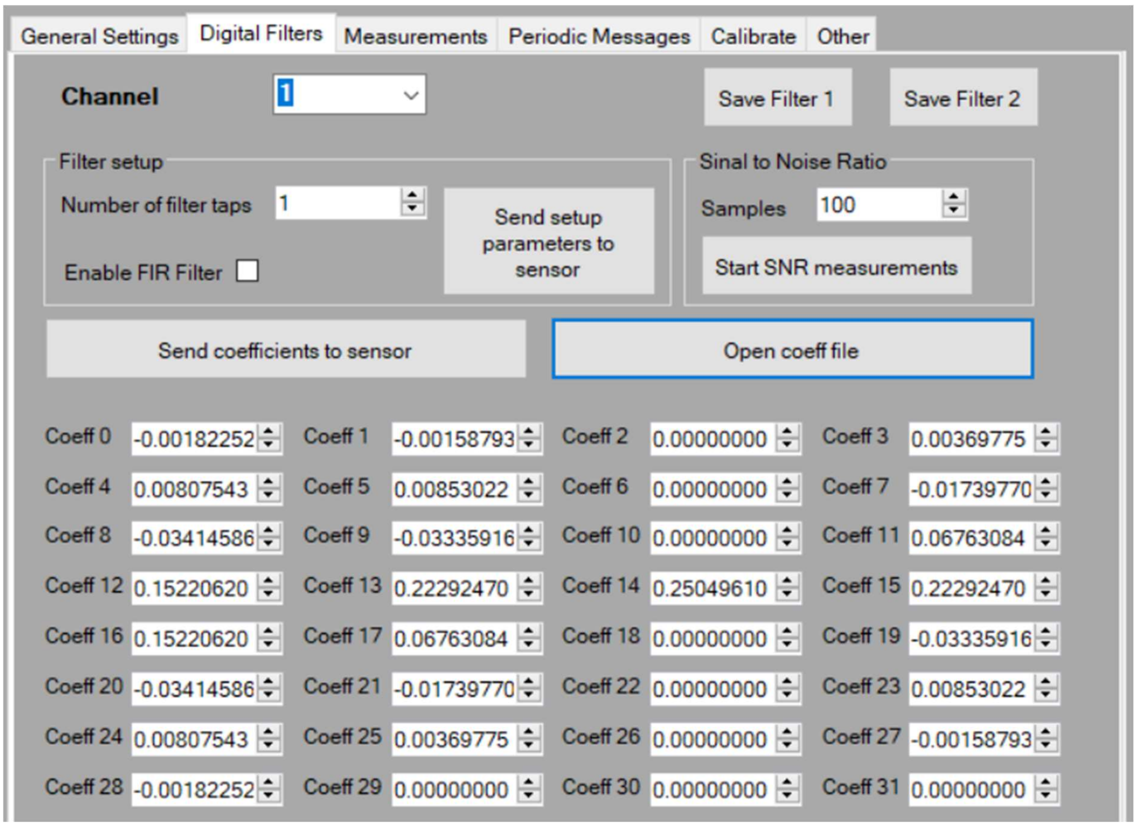
Reply:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xD5	**Channel**	**INX**	**0x00**	**COEFF**	**COEFF**	**COEFF**	**COEFF**
DLC = 0x08							

11.2 *Using the A2C-Sensor Utility*

The FIR parameters and coefficients can easily be changed using the graphical editor. To import coefficients, save them as a .coeff filetype. Each line in the file should have one single coefficient, with the first line being Coeff 0.

```
-0.0018225230
-0.0015879294
…
+0.0080754303
+0.0085302217
-0.0000000000
```



Press “Send coefficients to sensor” to update the A2C-SG2 coefficients. In the filter setup section set the number of taps and click the checkbox to enable the FIR filter. Press the “Send setup parameters to sensor”. For the A2C-SG2 to load the filter values at startup they must be saved first. Press the “Save Filter 1” and “Save Filter 2” to save the values to the memory, respectively for filter 1 & 2.

11.3 *Using MATLAB*

In MATLAB a FIR filter can be designed with the fir1 function:

```
h = fir1(28, 6/24);
```

The first argument is the filter order which must be one less than the desired length. So for this example the number of taps is 29, but we input 28. The second argument is the cutoff frequency in the range of 0 (DC) to 1.0 (Nyquist).

The A2C-SG FIR filter function requires the coefficients to be in time reversed order and this can be done using the ca

12 Getting measurement values

The measurements from the amplifier can be sent in different formats, either as 24bit integers, 32bit integers or 32 bit floating point.

There are currently 7 types of data available for each channel:
- Current value
- RMS value
- Mean value
- Minimum value
- Maximum value
- Sync value
- Sync RMS value

All types are continuously available. The mean, minimum and maximum values are calculated since startup or since the command: " Reset Global Minimum, Maximum and Mean values" is received - see section: 12.1.4.

12.1.1 Command: Get both

Send this command:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x0A	**RET**						
DLC = 0x02 (values above 0x02 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
- 0x01 = Get a previously synced value. Please see the Sync command in section
- 0x02 = Get the minimum values
- 0x03 = Get the maximum values
- 0x04 = Get the mean values
- 0x05 = Get the RMS values
- 0x06 = Get the previously synced RMS values
- 0x10

After the 0x0A and a sub command have been received, the sensor will return the measurements for both channels as two 24bit signed integers. The RET value applies to both channels at the same time.

Reply:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x0A	**Value sent is returned**	**Ch1 MSB**	**Ch1**	**Ch1 LSB**	**Ch2 MSB**	**Ch2**	**Ch2 LSB**
DLC = 0x08							

12.1.2 Command: Get configurable channel measurements

Send this command:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x0B	**Channel**	**Return type**	**Value type**				
DLC = 0x04							

Channel: is the channel number that is requested and must be one of the following values
- 0x00 = channel 1
- 0x01 = channel 2

Return type: is the data type returned:
- 0x00 = signed int - 4 bytes
- 0x01 = float - 4 bytes

Value type: is the requested value and must be one of the following values
- 0x00 = current measurement
- 0x01 = synced measurement
- 0x02 = minimum value
- 0x03 = maximum value

- 0x04 = mean value
- 0x05 = RMS value
- 0x06 = synced RMS value

Reply:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x0B	**Channel**	**Return type**	**Value type**	**Channel X MSB**	**Channel X**	**Channel X**	**Channel X LSB**
DLC = 0x08							

12.1.3 Command: Get current channel measurements with math functions

Send this command:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x0C	**Return type**	**Value type**	**MATH**				
DLC = 0x04 (values above 0x04 are also valid, but Data bytes are not used)							

Return type: is the data type returned:

- 0x00 = signed int - 4 bytes
- 0x01 = float - 4 bytes

Value type: is the requested value and must be one of the following values

- 0x00 = current measurement
- 0x01 = synced measurement
- 0x02 = minimum value
- 0x03 = maximum value
- 0x04 = mean value
- 0x05 = RMS value
- 0x06 = synced RMS value

MATH: is the requested math operation and must be one of the following values

- 0x00 = No math is performed
- 0x01 = Add channel 1 to channel 2
- 0x02 = Subtract channel 2 from channel 1
- 0x03 = Divide channel 2 with channel 1
- 0x04 = Multiply channel 1 with channel 2
- 0x05 = Subtract channel 1 from channel 2
- 0x06 = Divide channel 1 with channel 2

Reply:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x0C	**Return type**	**Value type**	**MATH**	**Math Result MSB**	**Math Result**	**Math Result**	**Math Result LSB**
DLC = 0x08							

12.1.4 Command: Reset Global Minimum, Maximum and Mean values

The global minimum, maximum and mean values 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	**VAL**						
DLC = 0x02 (values above 0x02 are also valid, but Data bytes are not used)							

VAL:

- 0x01 = Reset minimum, maximum and mean values for both input channels
- 0x02 = Reset minimum, maximum and mean values for input 1
- 0x03 = Reset minimum, maximum and mean values for input 2

12.1.5 Command: Set Integer Scaling

Since the calibration and all internal data types use floating point data types, getting integers requires scaling. The scaling can be any integer value, which is then multiplied to the internal floating point, truncated and eventually sent on the CAN bus. The default scaling value is 10. This means that if the actual value is 8.76, then this value is multiplied by 10 and truncated, thus the value sent on the CAN bus will be 87. There is no rounding, so a sufficient high scaling factor must be used.

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x1E	**Channel**	**Scaling MSB**	**Scaling**	**Scaling**	**Scaling LSB**		
DLC = 0x06 (values above 0x06 are also valid, but Data bytes are not used)							

Channel: is the channel number that is requested and must be one of the following values

- 0x00 = channel 1
- 0x01 = channel 2

Scaling: unsigned int – 4 bytes representing the scaling.

Example 1
Set scaling to 1000 for channel 1 - send the following command

Byte 0	***Byte 1***	***Byte 2***	***Byte 3***	***Byte 4***	***Byte 5***	***Byte 6***	***Byte 7***
0x1E	*0 (channel 1)*	*0x00*	*0x00*	*0x03*	*0xE8*		
DLC = 0x06							

As with all other parameters, the parameters must be saved if they are to be used automatically at startup.

12.1.6 Command: Get Integer Scaling

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x1F	**Channel**						
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]
0x1F	**Channel**	**Scaling MSB**	**Scaling**	**Scaling**	**Scaling LSB**		
DLC = 0x06							

Channel: is the channel number that is requested and must be one of the following values

- 0x00 = channel 1
- 0x01 = channel 2

Scaling: unsigned int – 4 bytes representing the scaling.

12.1.7 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.

12.1.8 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

12.1.9 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.

12.1.10 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

13 Setting up Periodic Messages

The strain gauge amplifier 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:

- Current measurements from both channels; Command: 0x0A
- Get configurable channel measurements; Command: 0x0B (sub commands 0x00, 0x01, 0x02)
- Get system mode (can be used as a heartbeat); 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.

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 (0x0A, 0x0B, 0xC0) and Byte 4 is the value of the subcommand. 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.**

If the data rate is set to a high value, data might not be ready to send. Please see 9.1.3.

13.1.1 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(periodic message 1)*	*0x01 (turn on)*	*0xC0 (command)*	*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 2 to send RMS values for all channels at a rate of 100 times per second

Byte 0	*Byte 1*	*Byte 2*	*Byte 3*	*Byte 4*	*Byte 5*	*Byte 6*	*Byte 7*
0x52	*0x02 (periodic message 2)*	*0x01 (turn on)*	*0x0A (command)*	*0x05 (sub command - RMS values)*	*0x00 (interval MSB)*	*0x0A(interval LSB = 10ms)*	
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 (off)*	*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.

13.1.2 Command: Set Periodic Messages Follow ADC

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x57	**VAL**						
DLC = 0x02 (values above 0x02 are also valid, but Data bytes are not used)							

VAL: Turns individual channels on/off:
- 0x00 = Turn off these periodic messages
- 0x01 = send a CAN message using a 32bit floating point type every time channel 1 finishes a conversion
- 0x02 = send a CAN message using a 32bit floating point type every time channel 2 finishes a conversion
- 0x03 = send a CAN message using a 32bit floating point type every time channel 1 or channel 2 finishes a conversion
- 0x04 = send a CAN message using a signed 32bit integer type every time channel 1 finishes a conversion
- 0x08 = send a CAN message using a signed 32bit integer type every time channel 2 finishes a conversion
- 0x0C = send a CAN message using a signed 32bit integer type every time channel 1 or channel 2 finishes a conversion
- 0x10 = send raw ADC data using a signed 32bit integer type every time channel 1 finishes a conversion
- 0x20 = send raw ADC data using a signed 32bit integer type every time channel 2 finishes a conversion
- 0x30 = send raw ADC data using a signed 32bit integer type every time channel 1 or channel 2 finishes a conversion

The returned data corresponds to data sent with command 0x0B - see section 12.1.2.

For single channel measurement with the lowest filter value of 1, the maximum number of messages is capped at 2400 messages per second, whereas the actual sample rate of the ADC is 4800 messages per second. This is to prevent CAN bus overload.

Note: Command 0x10, 0x20, 0x30 available from firmware version 0x00000118.

13.2 *J1939 type CAN Messages (only for 2 active channels)*

In this mode, each channel will have its own ID. Channel 1 will always be the same as the CAN Standard ID as set in 7.1.1, and channel 2 will always be one value higher. So, if the CAN Standard ID is set to 293, then channel 1 will output using this ID, and channel 2 will use 294.
Enabling J1939 type messages will override and the normal periodic messages that follows the ADC (command 0x57)
Only integer values are supported in this mode. The integer scaling is used as described in 12.1.5

13.2.1 Command: Set Periodic Messages to follow ADC - J1939 style

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 = send a single CAN message containing the normal value of the ADC, using a signed 32bit integer type every time channel 1 or 2 finishes a conversion
- 0x02 = send 3 CAN messages using a signed 32bit integer type every time channel 1 or 2 finishes a conversion. The first message contains the normal values, the second contains the minimum value, and the third contains the maximum value
- 0x03 =

Output:

Data[0]	Data[1]	Data[2]	Data[3]	Data[4]			
Ch1 MSB	**Ch1**	**Ch1**	**Ch1 LSB**	**Value type**			
DLC = 0x05							

Value type: is the requested value and must be one of the following values
- 0x00 = current measurement
- 0x02 = minimum value
- 0x03 = maximum value

13.2.2 Command: Get Periodic Messages to follow ADC - J1939 style

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 / 0x02 - as described for the Set command 0x6E

14 Signal to Noise calculations

The A2C-SG2 can be setup to output the SNR (signal to noise ratio) continuous in order to check the system for interference or noise. The output is calibrated in dB comparing the sampled data to the full swing of the ADC input voltage.
To evaluate the performance of the setup, the strain gauges must be held in a steady state. Alternatively, the input may be shorted at the gauge.

Both channels will be computed when the SNR is enabled. It is not possible to only enable one channel.

To setup SNR messages the number of samples must be specified. The value depends on the rate of SNR measurements required and on the data rate. If the data rate i.e. is set to 300 Hz, and the number of SNR measurements is set to 300, then the SNR will be outputted every second. The peak minimum and maximum values of those 300 samples are used to calculated the SNR.

To turn off the SNR calculations send 0 as the number of samples.

14.1.1 Command: Set SNR sample numbers

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x48	**RESV**	**Data MSB**	**Data LSB**				
DLC = 0x04 (values above 0x04 are also valid, but Data bytes are not used)							

- **RESV:** Reserved for future use
- **Data:** 16b unsigned integer containing the number of samples

When the SNR calculations are ready the A2C-SG2 will send the following message with the result which is identical to a standard message (value type mean):

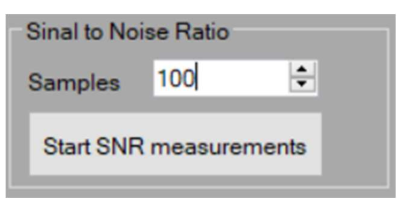
Reply:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x0B	**Channel**	**0x01**	**0x04**	**Data MSB**	**Data**	**Data**	**Data LSB**
DLC = 0x04							

Channel: is the channel number that is requested and must be one of the following values
- 0x00 = channel 1
- 0x01 = channel 2

Data: 32b float type, big endianness.

From the A2C-Sensor Utility you can directly enable the SNR function. Under the Digital Filters tap, set the Samples and click "Start SNR measurements"



15 Calibrating

The strain gauge amplifier is calculated with unitless values. A high and low value is required. This can be a strain gauge excitation, a millivolt signal etc.

The easiest way is to use the U2C and the A2C Sensor Utility. For most accurate calibration, the signal should be held steady for 1 second, before and after the measurement is taken by the sensor.

15.1.1 Command: Calibrate input channels using floating point data

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x20	**Channel**	**Data MSB**	**Data**	**Data**	**Data LSB**	**High/Low**	**0x80**
DLC = 0x08							

Channel:

- 0x00 = Calibrate channel 1
- 0x01 = Calibrate channel 2

High/Low:

- 0x00 = Low value of calibration data
- 0x01 = High value of calibration data

Example 1
Using 2-point calibration on channel 1, low value of 0.0f and high value of 500.0f
A float with a value of 0.0f will be all zeroes (data bytes 2-5)

Byte 0	***Byte 1***	***Byte 2***	***Byte 3***	***Byte 4***	***Byte 5***	***Byte 6***	***Byte 7***
0x20	*0x00 (channel 1)*	*0x00*	*0x00*	*0x00*	*0x00*	*0x00 (low)*	*0x80*
DLC = 0x08							

Now we need to set the high value. A float of 5000.0f will in bytes be: 0x45 0x9c 0x40 0x00

Byte 0	***Byte 1***	***Byte 2***	***Byte 3***	***Byte 4***	***Byte 5***	***Byte 6***	***Byte 7***
0x20	*0x00 (channel 1)*	*0x45*	*0x9C*	*0x40*	*0x00*	*0x01 (high)*	*0x80*
DLC = 0x08							

Other examples of float to byte conversions:

A float of 1000.12f will in bytes be: 0x44 0x7A 0x07 0xE6

Byte 0	***Byte 1***	***Byte 2***	***Byte 3***	***Byte 4***	***Byte 5***	***Byte 6***	***Byte 7***
0x20	*0x00 (channel 1)*	*0x44*	*0x7a*	*0x07*	*0xe6*	*0x01 (high)*	*0x80*
DLC = 0x08							

A float of - 123.987f will in bytes be: 0xC2 0xF7 0xF9 0x58

Byte 0	***Byte 1***	***Byte 2***	***Byte 3***	***Byte 4***	***Byte 5***	***Byte 6***	***Byte 7***
0x20	*0x00 (channel 1)*	*0xC2*	*0xF7*	*0xF9*	*0x58*	*0x01 (high)*	*0x80*
DLC = 0x08							

Compared to some converters from float to bytes, it may be required to reverse the array.

For an online converter please try: https://gregstoll.com/~gregstoll/floattohex/

15.1.2 Command: Calibrate input channels using integer data (signed 32bit integer)

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x19	**Channel**	**Data MSB**	**Data**	**Data**	**Data LSB**	**High/Low**	**0x80**
DLC = 0x08							

Channel:

- 0x00 = Calibrate channel 1
- 0x01 = Calibrate channel 2

High/Low:

- 0x00 = Low value of calibration data
- 0x01 = High value of calibration data

Example 1
Using 2 point calibration on channel 2, low value of 1000 and high value of 500000

Byte 0	*Byte 1*	*Byte 2*	*Byte 3*	*Byte 4*	*Byte 5*	*Byte 6*	*Byte 7*
0x19	*0x01 (channel 2)*	*0x00*	*0x00*	*0x03*	*0xE8*	*0x00 (low)*	*0x80*
DLC = 0x08							

Now we need to set the high value. A signed int of 500000 will in bytes be: 0x00 0x07 0xA1 0x20

Byte 0	*Byte 1*	*Byte 2*	*Byte 3*	*Byte 4*	*Byte 5*	*Byte 6*	*Byte 7*
0x19	*0x01 (channel 2)*	*0x00*	*0x07*	*0xA1*	*0x20*	*0x01 (high)*	*0x80*
DLC = 0x08							

Make sure to save the calibration constants using command 0x21. See 17.1.1

16 Set Default Calibration Values

To return to the default calibration constants send the below command. This will have no effect until the calibration data is saved and the sensor restarted. See command 17.1.1

16.1.1 Command: Set Default Calibration Constants

Issue the following command to set the default calibration constants:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x22	**0xFF**						
DLC = 0x02 (values above 2 are also valid, but Data bytes are not used)							

17 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 19
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.

17.1.1 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)							

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)
- 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							

19 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)							

20 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.

20.1.1 Command: Set factory settings

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x55	**0x01**	**0x53**	**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.

21 Recommended settings to get started

Example 1
Setup ADC for a strain gauge bridge on both channels, with chop and buffer enabled and 10Hz sample rate for each channel. The data rate is set to 30 - 0x001E and is sent on the CAN Bus every time an ADC finishes a sample

Set integer scaling to 10000 for channel 1

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x1E	0x00 (channel 1)	0x00	0x00	0x03	0xE8		
DLC = 0x06							

Set integer scaling to 10000 for channel 2

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x1E	0x01 (channel 2)	0x00	0x00	0x27	0x10		
DLC = 0x06							

Setup ADC

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x40	0x03 - both channels	0x00 - bipolar	0x80 - gain of 128	0x00	0x1E	0x01 chop enabled	0x01 buffer enabled
DLC = 0x08							

Setup periodic messages to follow ADCs and output a signed integer

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x57	0x0C - both channels						
DLC = 0x02							

Disable any J1939 periodic messages

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x6E	0x00channels						
DLC = 0x02							

Save settings

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x50	0xFF						
DLC = 0x02							

22 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.

23 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 follow

23.1.1 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

23.2 *Error message list:*

- 0x0001 = Error Baud Rate 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
- 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
- 0x001C = Error Get Incoming Filter ID out of range
- 0x001D = Error Get Sensor Information Sub Command out of range
- 0x0022 = Error Enter Bootloader 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.
- 0x0034 = Error Set Output Inverted out of range, must be 0x00 or 0x01 but was set to a different value.
- 0x0035 = Error J1939 Mode out of range
- 0x0036 = Error CAN set FIR filter coefficients channel
- 0x0037 = Error CAN set FIR filter control
- 0x0038 = Error CAN get FIR filter control
- 0x0039 = Error CAN get FIR filter coefficients channel out of range
- 0x003A = Error CAN get FIR filter coefficients out of range
- 0x003B = Error CAN set FIR filter coefficients out of range
- 0x003C = Error CAN save FIR filter parameters

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.