LILLIE SYSTEMS

3-Channel 0-20mA analyzer with CAN bus and logic output.

A2C-mA-M12-X

General Description

The A2C-mA-M12 is a 0-20mA analyzer that can measure 3 x mA signals simultaneously. The analyzer can calculate True RMS values, minimum, maximum values, average values and combine multiple inputs with mathematical calculations. For instance, the difference between two inputs can be calculated (can be used for differential measurements). The results calculated can be used to trigger up to 6 different alarms with programmable thresholds.
The analyzer can communicate the measurements to a host via CAN Bus or it can be used in a standalone mode without any CAN bus connected after initial programming with the U2C accessory. In the event of an alarm triggering, the logic output is activated, which can be used to drive a relay / buzzer / lamp or plc input.

The optional USB programmer (U2C) with free software, simplifies programming, and no knowledge of CAN bus communication is required.

Features

- 3 channels 0-20mA inputs with over current protection
- Selectable Bandwidth
- Programmable number of sample averages
- Programmable alarms for mA on all inputs
- Heartbeat CAN messages with programmable periods
- Periodic CAN messages with programmable data and time periods
- Logic alarm output for standalone mode without the use of CAN bus
- Low power consumption
- Durable aluminum / stainless steel housing
- Software upgradable via CAN bus

Specifications

- 10-30V supply voltage
- 30mA supply current
- CAN interface (2.0A & B)
- CAN driver ISO 11898 compatible
- Open collector output with over current protection
- Industry standard M12 connectors
- CNC machined aluminum / stainless steel housing
- Small size

1 Ordering information

Part Number	Package	Interface	CAN Bus	Logic Output
A2C-mA-M12-A	35x35x29.7mm Anodized aluminum	M12, 5pin Male connector.	Yes	Yes (pin 1)
A2C-mA-M12-S	35x35x29.7mm Stainless steel 316	M12, 5pin Male connector.	Yes	Yes (pin 1)

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

Specifications for

A2C-mA-M12

3 Channel, 0-20 mill ampere analyzer with CAN-Bus and logic output

Version 1.03

Document tracking control

VERSION	SECTION	CHANGED BY	DATE	CHANGE
1.00	All	JL	04-06-2013	Initial Version
1.01		JL	12-07-2013	Updated error codes, sampling rate function
1.02		JL	04-04-2014	Sync function explanation updated
1.03	8+10	JL	28-08-2014	Reset Min Max Mean function added

Contents

2 Specifications

Parameter	Condition	Values			Unit
SENSOR INPUT		Min	Typical	Max	
Measurement Range		0.5		20	*mA*
Resolution			0.001		*mA*
FREQUENCY RESPONSE					
Bandwidth (-3dB)[1]		25		340	Hz
NOISE PERFORMANCE					
Noise Density all channels			5		µA
SUPPLY VOLTAGE		Min	Typical	Max	
Operating Voltage (Vin)		10		30	V
Supply Current	Vin = 24V		35		mA
	Vin = 12V		28		mA
Power consumption	Vin = 24V		0.34		W
	Vin = 12V		0.34		W
Turn-On Time			50		ms
Open collector output					
Maximum Collector Emitter Voltage			40		V
Maximum continuous current			250		mA
CAN BUS 2.0A & B					
Transceiver delay loop time				150	ns
Baud rate		10		1000	kBits/sec
Default device standard ID			0x124		
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			Anodized aluminum		
Lid Material - S suffix			Stainless steel 316		
CONNECTIVITY CAN Bus & POWER			M12-A Male 5 pin Connector		
Pin 1 = logic output					
Pin 2 = Vin					
Pin 3= Ground (both power and CAN ground)					
Pin 4= CAN High					
Pin 5 = CAN Low					
CONNECTIVITY INPUTS			M12-A Female 5 pin Connector		
Pin 1 = V+ (Vin - 0.5V)[2]			Sensor positive power		
Pin 2 = Ground			Sensor ground (negative)		
Pin 3= mA Input 1					
Pin 4= mA Input 2					
Pin 5 = mA Input 3					
DIMENSIONS		Min	Typical	Max	
Length			62.5		mm
Width			35		mm
Height (M12 version)			29,7		mm
Weight (M12 version)			70		gram
TEMPERATURE					
Operating Temperature Range		-20		75	Deg
Housing temperature rise				10	Deg

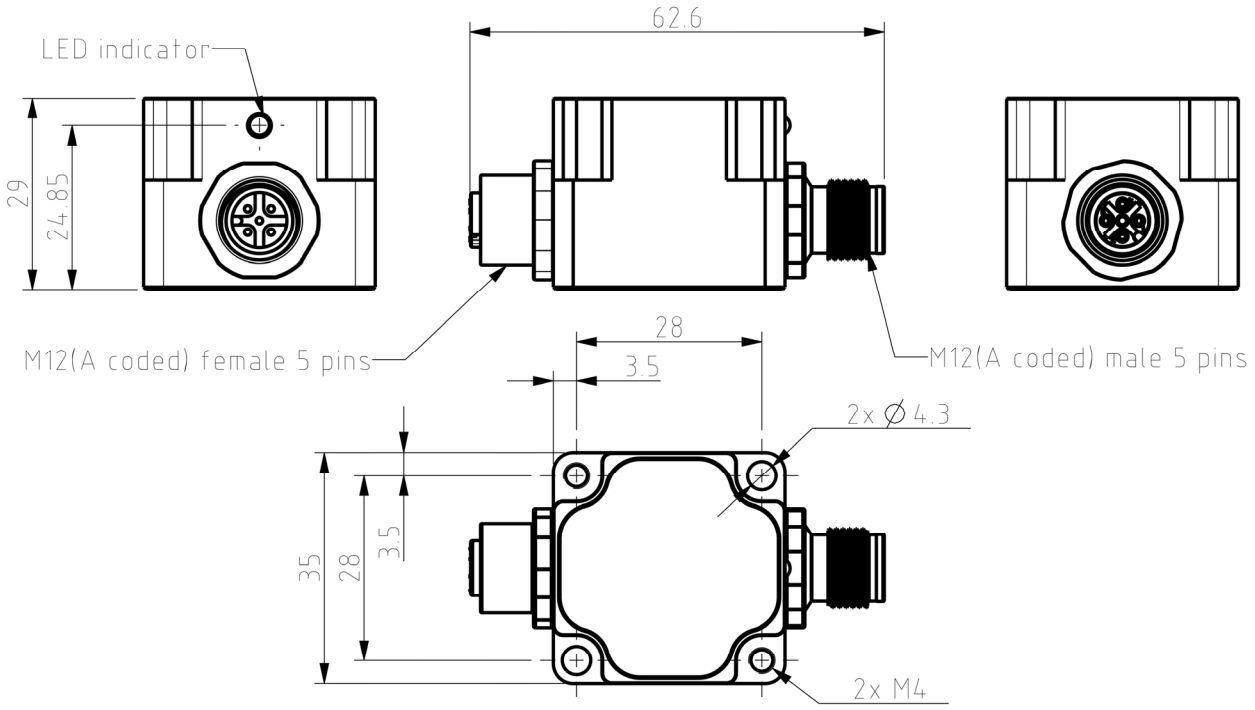
[1] Additional averaging for low frequency sensing available
[2] Maximum 200mA current draw. This output it not protected against short circuit!

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

2.1 *Temperature influence*

The analyzer is calibrated at 26deg Celsius. Operating the analyzer at other temperatures will cause offsets on the measurements. Lillie Systems plans to release a firmware update with temperature compensation in September 2014.

3 Mechanical Drawing



4 Mounting

Use two M4 bolts to secure the analyzer 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 analyzer 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 analyzer 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 all parameters in the analyzer, and in real time see the measurements

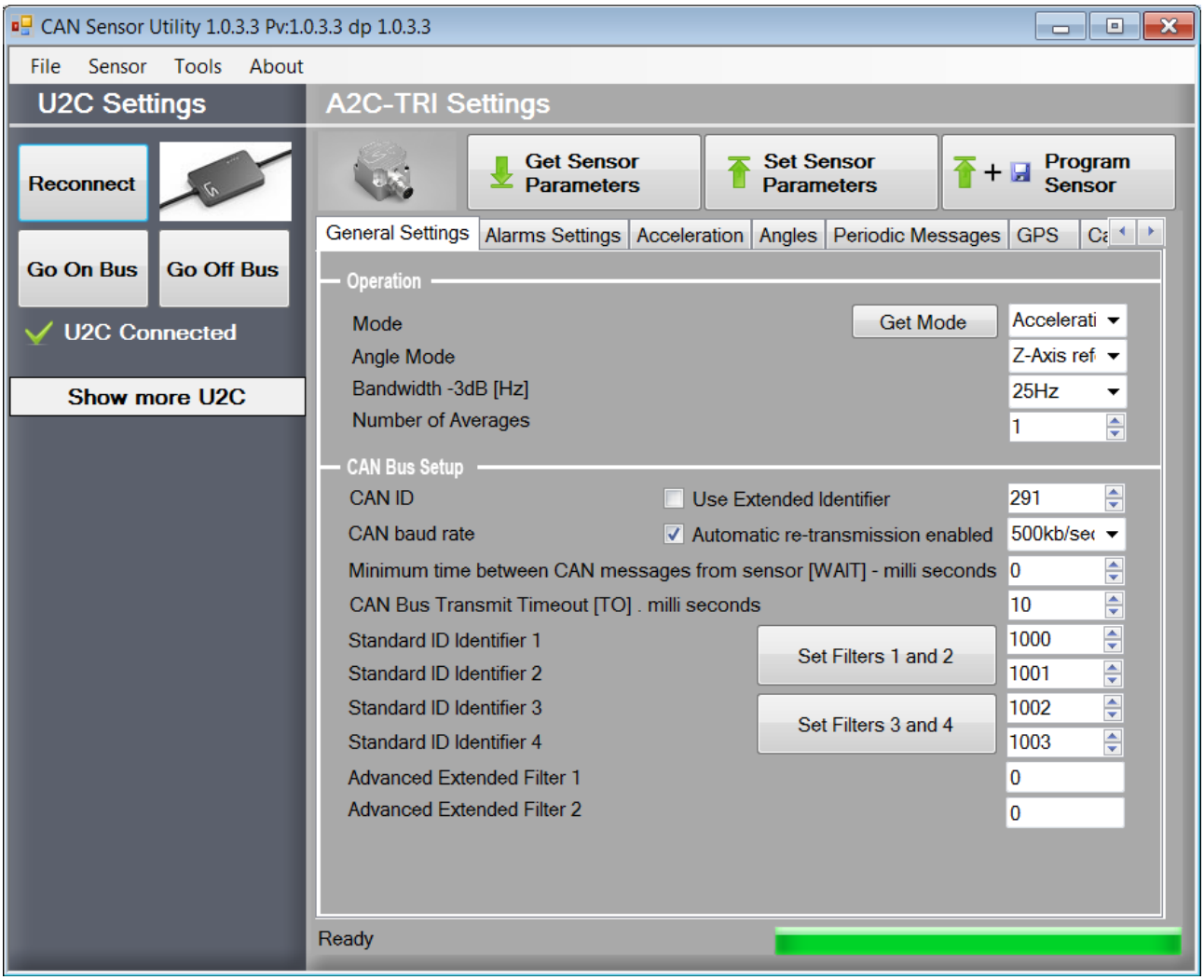


Figure 2 - Windows GUI

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 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 analyzer comes with factory setting such as CAN filters, CAN Identifiers, bandwidth, mode etc. To prepare the analyzer 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 analyzer sends when transmitting messages.
The factory default value is 0x124.

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 analyzer 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.

7.2.1 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'**	**(char) 'A'**	**(char) 'F'**	**(char) 'E'**
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 7.2.1. 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.

$$T1 = \frac{32 \times 10^6}{Baud\ rate \times PRES}$$

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

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

Example: Generate baud rate of 62.5kbits / second: We first select a prescale (PRES) value that creates an even T1 number. 32 is selected. The sample point is chosen to be 75%

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

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

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

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:

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

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

- 0x00 = 1 time quantum
- 0x01 = 2 time quantums
- …

- 0x0F = 16 time quantums

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

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

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 analyzer will only respond to values which have passed through its CAN message filters. This means that many similar analyzers can be attached to the same CAN network and by defining filters, only the analyzer 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 analyzer 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 analyzer 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 Setup Bandwidth

The bandwidth determines what frequencies the analyzer can sense. The higher the bandwidth, the higher the frequencies, lower response time but higher noise. The lower the bandwidth the lower frequencies the analyzer can sense, but the noise will be lower too. The inner workings of the sensor electronics and DSP can be seen in Figure 3. The DSP uses a fixed high frequency ADC which is oversampling the signals. A digital filter is used to change the bandwidth, providing better signal to noise levels. After the selected Bandwidth, an additional averaging can be applied.

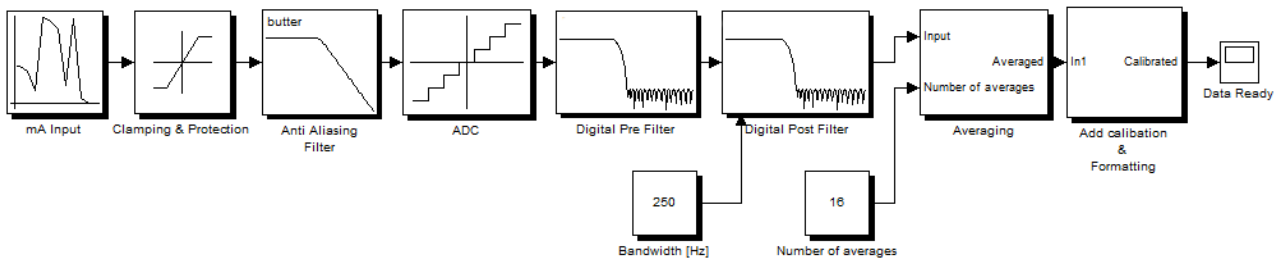


Figure 3 - Inner workings

8.1.1 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 channels, (-3dB). All channels are sampled simultaneously and the BW cannot be set individually for each channel. 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 analyzer performance in low frequency measurement application.

- LP = 1-1024 is possible.

It should be noted that the analyzer 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 Bandwidth to 25Hz, 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							

8.1.2 Command: Get System Bandwidth

To get the current 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 analyzer 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 8.1.1

9 Syncing Data between many devices in a CAN bus network

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

9.1.1 **Command: Sample Sync**

Send this command:

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 = Normal instantaneous values are saved for later retrieval
- 0x02 = RMS values are saved for later retrieval

See section 10.1.1 for commands that retrieve the synchronized data.

10 Getting mA values

The mA measurements from the analyzer are sent as integer values multiplied with 1000. This means that a value received of 15520 must be divided by 1000, thus being 15.520mA

There are currently 5 types of data available for each channel:

- Current value
- RMS value
- Mean value
- Minimum value
- Maximum 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: 10.1.5.

There are two commands to receive measurements. The first command gets all 3 channels at the same time. For the second command the user must specify which channel to receive but this can return more information in one CAN package.

10.1.1 Command: Get all channels

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 9.1.1
- 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

After the 0x0A and a sub command have been received, the sensor will return the measurements for all 3 channels

Reply:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x0A	**Value sent is returned**	**Ch1 MSB**	**Ch1 LSB**	**Ch2 MSB**	**Ch2 LSB**	**Ch3 MSB**	**Ch3 LSB**
DLC = 0x08							

10.1.2 Command: Get configurable channel measurements

Send this command:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x0B	**0x00**	**Channel X**	**Value X**	**Channel Y**	**Value Y**	**Channel Z**	**Value Z**
DLC = 0x08							

Channel X/Y/Z: is the channel number that is requested and must be one of the following values

- 0x00 = channel 1
- 0x01 = channel 2
- 0x02 = channel 3

Value X/Y/Z: is the requested value and must be one of the following values

- 0x00 = current mA measurement
- 0x01 = synced mA measurement
- 0x02 = minimum value
- 0x03 = maximum value
- 0x04 = mean value

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

Example.
"To get channel 1 RMS value, Channel 1 minimum value, and Channel 3 maximum value. Send the following CAN Bus package:

Byte 0	***Byte 1***	***Byte 2***	***Byte 3***	***Byte 4***	***Byte 5***	***Byte 6***	***Byte 7***
0x0B	*0x00*	*0x00 (channel 1)*	*0x05 (RMS value)*	*0x00 (channel 1)*	*0x02 (min value)*	*0x02 (channel 3)*	*0x03 (max value)*
DLC = 0x8							

Reply:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x0B	**0x00**	**Channel X MSB**	**Channel X LSB**	**Channel Y MSB**	**Channel Y LSB**	**Channel Z MSB**	**Channel Z LSB**
DLC = 0x08							

10.1.3 **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]
0x0B	**0x01**	**Channel X**	**Channel Y**	**MATH**			
DLC = 0x05 (values above 0x05 are also valid, but Data bytes are not used)							

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

- 0x00 = channel 1
- 0x01 = channel 2
- 0x02 = channel 3

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

- 0x00 = No math is performed
- 0x01 = Add channel X to channel Y
- 0x02 = Subtract channel X from channel Y
- 0x03 = Divide channel X with channel Y
- 0x04 = Multiply channel X with channel Y

Example.
"To subtract channel 2 from channel 1, Send the following CAN Bus package:

Byte 0	***Byte 1***	***Byte 2***	***Byte 3***	***Byte 4***	***Byte 5***	***Byte 6***	***Byte 7***
0x0B	*0x01*	*0x01 (channel 2)*	*0x00 (channel 1)*	*0x02 (subtract)*			
DLC = 0x05							

Reply:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x0B	**0x01**	**Channel X**	**Channel Y**	**MATH**	**Result LSB**	**Result MSB**	
DLC = 0x08							

The result is a signed 16bit integer, so negative values are represented.

10.1.4 **Command: Get RMS channel measurements with math functions**

Send this command:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x0B	**0x02**	**Channel X**	**Channel Y**	**MATH**			
DLC = 0x05 (values above 0x05 are also valid, but Data bytes are not used)							

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

- 0x00 = channel 1
- 0x01 = channel 2
- 0x02 = channel 3

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

- 0x00 = No math is performed
- 0x01 = Add channel X to channel Y
- 0x02 = Subtract channel X from channel Y
- 0x03 = Divide channel X with channel Y
- 0x04 = Multiply channel X with channel Y

Example.
"To subtract channel 2 RMS value from channel 1 RMS value, Send the following CAN Bus package:

Byte 0	***Byte 1***	***Byte 2***	***Byte 3***	***Byte 4***	***Byte 5***	***Byte 6***	***Byte 7***
0x0B	*0x02*	*0x01 (channel 2)*	*0x00 (channel 1)*	*0x02 (subtract)*			
DLC = 0x05							

Reply:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x0B	**0x02**	**Channel X**	**Channel Y**	**MATH**	**Result LSB**	**Result MSB**	
DLC = 0x08							

The result is a signed 16bit integer, so negative values are represented.

10.1.5 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 all 3 input channels
- 0x02 = Reset minimum, maximum and mean values for input 1
- 0x03 = Reset minimum, maximum and mean values for input 2
- 0x04 = Reset minimum, maximum and mean values for input 3

11 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 - Boot loader

11.1.1 Command: Get sensor information

Send this command:

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							

12 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:

- Current measurements from all channels; Command: 0x0A
- Get configurable channel measurements; Command: 0x0B (sub commands 0x00, 0x01, 0x02)
- Get system mode (can be used as a heart beat); Command: 0xC0
- Get Alarms Register: Command: 0xEE

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 which would normally be 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.**

12.1.1 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 Setting Alarms

Alarms are used to monitor each channel. They can be setup to trigger in an event of a given measurement becoming too large or small. A total of 6 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 6.550mA. The hysteresis can then be set to e.g. 6.450mA. This means that the alarm will trigger at 6.550mA and only stop once the measurement has been reduced to 6.450mA. 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.

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

Trip points can be set between 0.5mA and 20mA.

13.1.1 Command: Set alarms trip points / hysteresis

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x6B	**Alarm number**	**Channel**	**Logic**	**Threshold MSB**	**Threshold LSB**	**Hysteresis MSB**	**Hysteresis LSB**
DLC = 0x08							

Alarm number:

- Value between 0x00 and 0x05

Channel:

- 0x00 = channel 1
- 0x01 = channel 2
- 0x02 = channel 3

Logic:

- 0x00 = Alarm will be turned off
- 0x01 = Less than or equal
- 0x02 = More than

Threshold:

- Value between 500 (0.5mA) and 20000 (20mA); 0x01F4 to 0x4E20

Hysteresis:

- Value between 500 (0.5mA) and 20000 (20mA); 0x01F4 to 0x4E20

Example
Set alarm number 1to trigger if the measurements are above 10.5mA on channel 1, set hysteresis to 10.0mA

Byte 0	***Byte 1***	***Byte 2***	***Byte 3***	***Byte 4***	***Byte 5***	***Byte 6***	***Byte 7***
0x6B	*0x00 (alarm number 1)*	*0x00(channel 1)*	*0x02 (trigger if more than)*	*0x29*	*0x04*	*0x27*	*0x10*
DLC = 0x08							

13.1.2 Command: Get alarms settings

Send this command::

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0xEB	**Alarm number**						
DLC = 0x02							

The analyzer will reply with:

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x6B	**Alarm number**	**Channel**	**Logic**	**Threshold MSB**	**Threshold LSB**	**Hysteresis MSB**	**Hysteresis LSB**
DLC = 0x08							

13.1.3 **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							

13.1.4 **Command: Get Enable Alarms**

Get the alarm setting from the analyzer

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 analyzer

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 13.1.3

13.1 *Alarm registers*

This CAN message is sent from the analyzer when alarm trips – if this is enabled in **Error! Reference source not found.**. One byte represents the alarm state. If the bit for the given alarm is set then the alarm is currently tripped. This register can 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 12.1.1

13.1.1 **Command: Get alarm register**

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

Reply 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**					
DLC = 0x04							

	Bit	**If bit is set then the alarm is tripped!**
Data0	Bit 0	Alarm 0
	Bit 1	Alarm 1
	Bit 2	Alarm 2
	Bit 3	Alarm 3
	Bit 4	Alarm 4
	Bit 5	Alarm 5

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

Byte 0	***Byte 1***	***Byte 2***	
0xEE	*0x00*	*0x01 (binary 00000001)*	
DLC = 0x04			

This indicates that alarm 1 has tripped

13.1.2 **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

13.1.3 **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

13.1.4 **Command: Set delay between CAN messages on error**

If the Alarm in 13.1.3 is set to send a CAN bus message upon tripping then these messages will continue to be sent. 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 analyzer will wait between CAN messages. This is to make sure that the host has time to process the information, especially when sending multiple packages.
-

13.1.5 **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							

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

14 Save Current Parameters

After changing any parameter, these will immediately take effect. However, they will be lost after a power cycle unless they are saved in memory.
The saved values does not include the calibration values which can be saved using command in section 18.1.1
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.

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)							

15 Reset to Factory Settings

This will reset the analyzer 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.

15.1.1 **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 analyzer will reset and re-start itself. During this time it will be unresponsive.

16 Calibrating the analyzer

The analyzer is factory calibrated but can be calibrated again using a mA calibrator.
Each channel is calibrated using two known mA currents. A linear approximation is then used for the whole input range.
Normally the calibration inputs are 4 and 16mA.

16.1.1 **Command: Calibrate axis**

Command	Sub Command	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
0x20	**Channel High / Low**	**Value MSB**	**Value LSB**				
DLC = 0x06							

Channel High / Low:

- 0x01 = Calibrate channel 1, high value (e.g. 16mA)
- 0x02 = Calibrate channel 2, high value (e.g. 16mA)
- 0x03 = Calibrate channel 3, high value (e.g. 16mA)
- 0x04 = Calibrate channel 1, low value (e.g. 4mA)
- 0x05 = Calibrate channel 2, low value (e.g. 4mA)
- 0x06 = Calibrate channel 3, low value (e.g. 4mA)

For each channel, the low value (e.g. 4mA) must be calibrated first, followed by the high value. When the high value has been calibrated, the new calibrations take effect immediately, but they have not yet been saved. To save them, see section 18.1.1
If for some reason the calibrations are not wanted, the analyzer can be power cycled before saving the calibration values.

17 Set Factory Calibration Values

Should a calibration have been performed, saved and the analyzer 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.

17.1.1 **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.

18 Save Current Calibration Constants

After changing calibration constants, these will immediately take effect. However, they will be lost after a power cycle unless they are saved in memory.
The saved values does not include the other parameters in the analyzer, which can be saved using the command in section 14
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.

18.1.1 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)							

19 Recovering Filter Settings

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

1. Turn off analyzer
2. Setup a CAN bus device to transmit at a Baud rate of 75kbits/sec.
3. Turn on the analyzer
4. Send the following message with a CAN ID of 0x7FF 50ms after it is powered up.

19.1.1 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 analyzer 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 4. Press the "Recover Filter 1,2" button and then apply power to the analyzer.

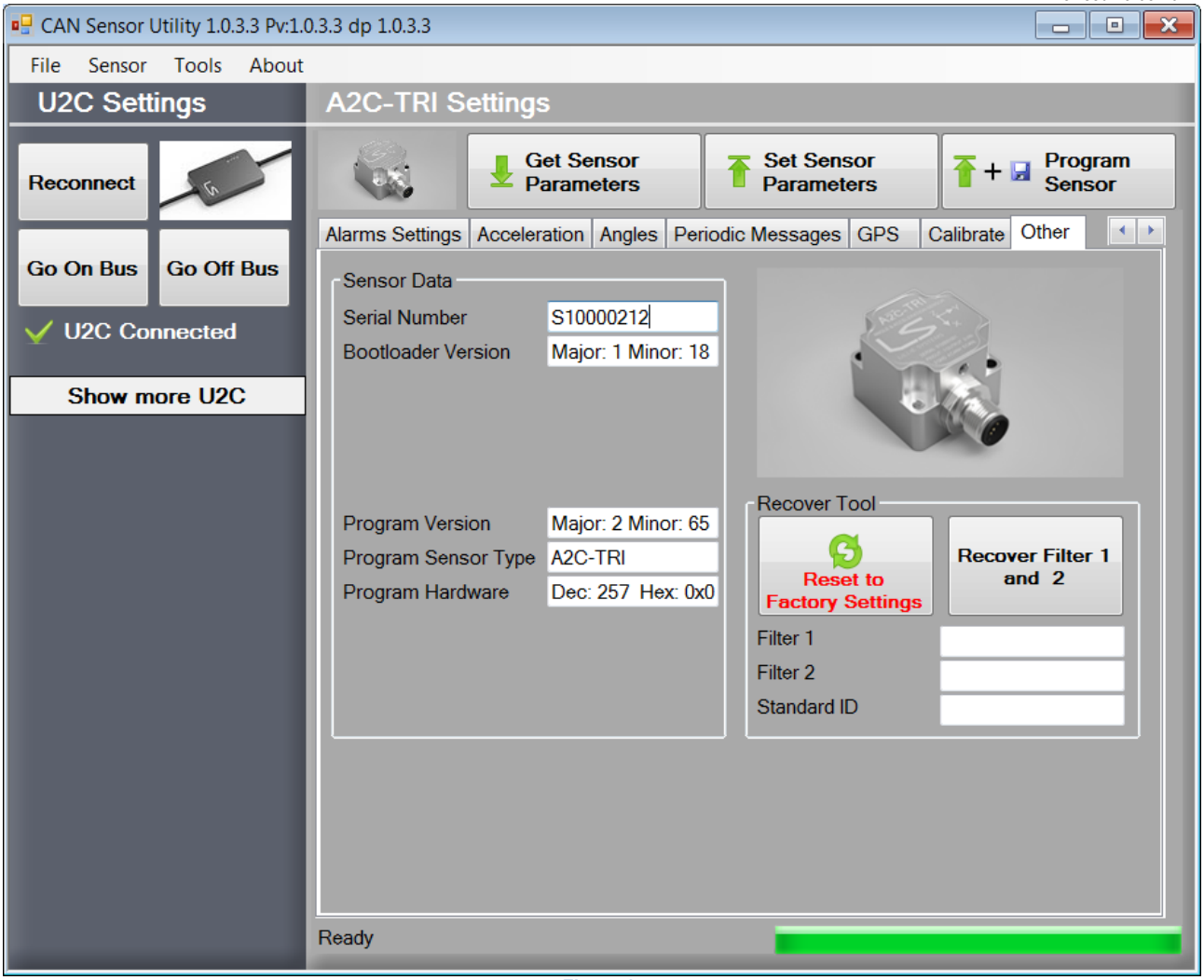


Figure 4

20 Updating Sensor Firmware

The analyzer 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.

21 Error Codes

If the analyzer 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; only received.

The message format is a follows

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

21.2 *Error message list:*

- 0x0001 Baud Rate Out Of Range.
- 0x0002 Mode Out Of Range
- 0x0003 Band Width Out OF Range
- 0x0004 Channel Selection Out OF Range
- 0x0005 = Limit Max Out Of Range
- 0x0006 = Limit Min Out Of Range
- 0x0007 = Limit Sub Command Out Of Range
- 0x0008 = Limit Angle Max Out Of Range
- 0x0009 = Set Alarms Math Out Of Range
- 0x000A = Set Alarm Number To Be Set Out Of Range
- 0x000B = Get Delay Between CAN Messages On Error Out Of Range
- 0x000C = Set Delay Between CAN Messages On Error Out Of Range
- 0x000D = Get Alarms To Be Checked Out Of Range
- 0x000E = Get Alarms Math Out Of Range
- 0x0011 = Set Values To Zero Out Of Range
- 0x0012 = Set Periodic Task Sub Command Out Of Range, // must be 0 for general or 1-4 (max number of tasks)
- 0x0013 = Set Periodic Task Not Valid
- 0x0014 = Set Periodic Task Interval Below 2ms
- 0x0015 = Get Periodic Task Out Of Range
- 0x0016 = Set Alarm Modes Error Mode Out Of Range
- 0x0017 = Set CAN Custom Baud Error Mode Out Of Range
- 0x0018 = Set Std ID Out Of Range
- 0x0019 = Set IncomingFilterID1_2ID Out Of Range
- 0x001A = Set IncomingFilterID3_4ID Out Of Range
- 0x001B = Set Limits To Be Checked Out Of Range
- 0x001C = Get Incoming Filter ID Out Of Range
- 0x001D = Get Sensor Information Sub Command Out Of Range
- 0x001E = Save Calibration Values To Flash Sub Command Not 0xFF
- 0x001F = Calibrate Using Earth Gravity Sub Command Out Of Range
- 0x0020 = Set Default Calibration Values Sub Command Not 0xFF
- 0x0021 = Set Save Parameters To Flash Sub Command Not 0xFF
- 0x0022 = Enter Boot loader Data Not Valid
- 0x0023 = Set Output On Off Data Out Of Range
- 0x0024 = Command Not Valid
- 0x0025 = Set Factory Settings Wrong Data
- 0x0026 = Set Ext ID Out Of Range
- 0x0027 = Set CAN ID Sub command Out Of Range
- 0x0028 = Set Logic Output Parameters Sub Command Out Of Range
- 0x002B = Limit Min Hysteresis Out Of Range

- 0x002C = Limit Max Hysteresis Out Of Range
- 0x002F = Sub command CAN Send all measurements is not between 0x00 and 0x06.
- 0x0030 = Sub command CAN Send all RMS measurements is not between 0x00 and 0x06.
- 0x0031 = Sub command Sample Sync Out Of Range. Should be 0x01 to 0x03.
- 0x0033 = Math Parameters Out Of Range Send All measurements
- 0x0034 = Set Output Inverted Out Of Range, must be 0x00 or 0x01 but was set to a different value.
- 0x0035 = Calibration Data Out Of Range

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.