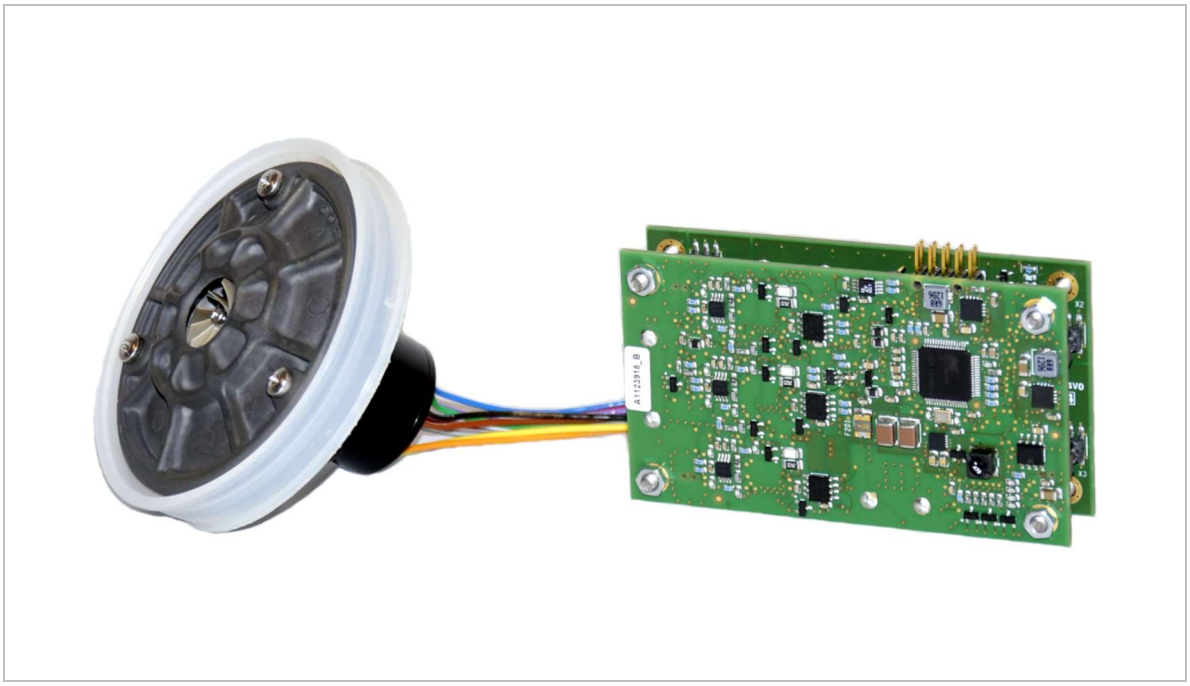
DEMCON | macawi respiratory systems

User Manual

Manual

1008 - Panoramix



Blower Controller Module

Protocol version 1

Software version 1.1 and 1.2

Document Identification and Filing		
Product:	Version:	Status:
1008	**6.0**	**Released**
Project:	Dossier:	Doc. ID:
563.01	**DHF**	**PRD1008-6-8**

Template:

R 4.2F Basic User Manual

WARNING

For a full understanding of the performance characteristics of this module, the user should carefully read these Instructions for Use before use of the module

1. WORKING WITH THESE INSTRUCTIONS FOR USE

The Title of the main chapter in the header line helps with orientation and navigation.

The instructions for the user combine text and illustrations, providing a comprehensive overview of the module.

Definitions

WARNING

A WARNING statement provides important information about a potentially hazardous situation which, if not avoided, could result in death or serious injury.

CAUTION

A CAUTION statement provides important information about a potentially hazardous situation, if not avoided, may result in minor or moderate injury to the user of patient or in damage to the module or other property.

NOTE

A Note provides additional information intended to avoid inconvenience during operation.

2. TABLE OF CONTENTS

3. INTRODUCTION

This **RESPIRATORY BLOWER** represents the latest generation of blower controller technology.

The COMPLETE SYSTEM consists of a controller printed circuit board and a centrifugal blower.

Recent technological advances on high speed turbines have been used allowing a performance only previously available from equipment of much greater size, higher noise level and higher cost.

The **SYSTEM** can be controlled by a Serial 3V3 interface with a specific serial protocol or analog signal 0-5 Volt

3.1 RoHS compliancy

Based on analyses, manufacturer statements and CE markings, the Panoramix unit is RoHS compliant as far as Demcon Macawi respiratory systems B.V. can check.

4. MODULE INTEGRATION

4.1 Mounting

Blower part

The Macawi Respiratory blower is designed to be integrated in the gas-conduit of a medical ventilator. The motor cooling is based on the respiratory flow through the blower housing along the motor.

CAUTION

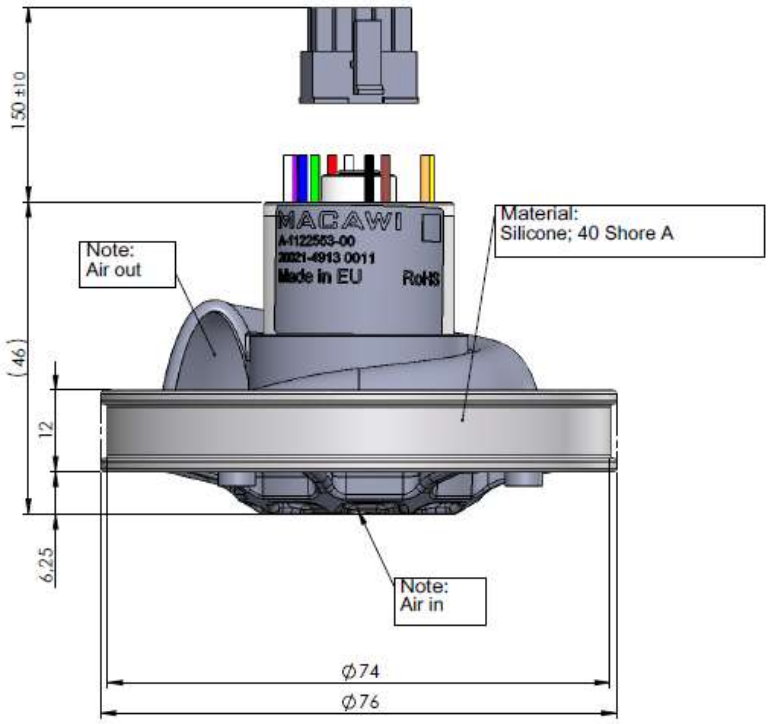
It is important to construct the housing for the blower that it enables the flow from the Air out to pass the motor.

Around the spiral housing of the blower a silicon seal is mounted. This seal has a double functionality;

- To seal the “low pressure” inlet side from the “high pressure” outlet side.
- To mechanically separate the blower from the housing to adsorb eventual vibrations so that these are not conducted to the outer housing of the respiratory blower.

The construction of the housing around the blower is important in the fulfillment of these functionalities.

Below you’ll find the drawing with the important dimensions of the respiratory blower



CAUTION

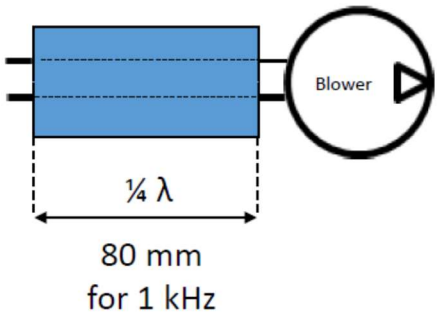
The temperature of the motor is mainly influenced by the peak current needed for the acceleration speed and the braking speed of the motor. Take care not to set an unnecessary short ramp-up and/or ramp-down time for the pressure or flow ramp.

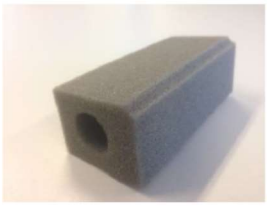
4.2 Examples or Noise reduction measures

4.2.1 Solution way 1 for Blower/Flow noise absorption.

Inlet labyrinth based on noise absorption over ¼ λ.

Irritating frequencies are from 1 kHz what means an adsorption path length of 80mm with open cell noise adsorption foam at the entrance of the blower. This means that you have to construct a chamber with foam with one or more holes which have a length of at least 80 mm. Use enough thickness of the foam, preferable about 20 – 30 mm.



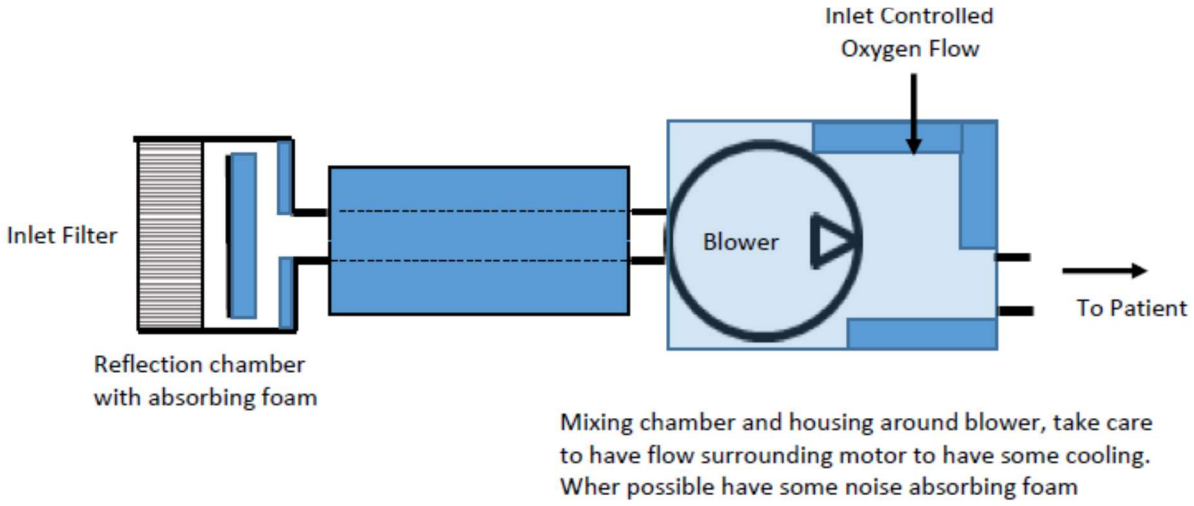


OR

In the path between the absorbtion chamber and the inlet of the blower it is prefered to have noise absorbing foam also, like in the picture.

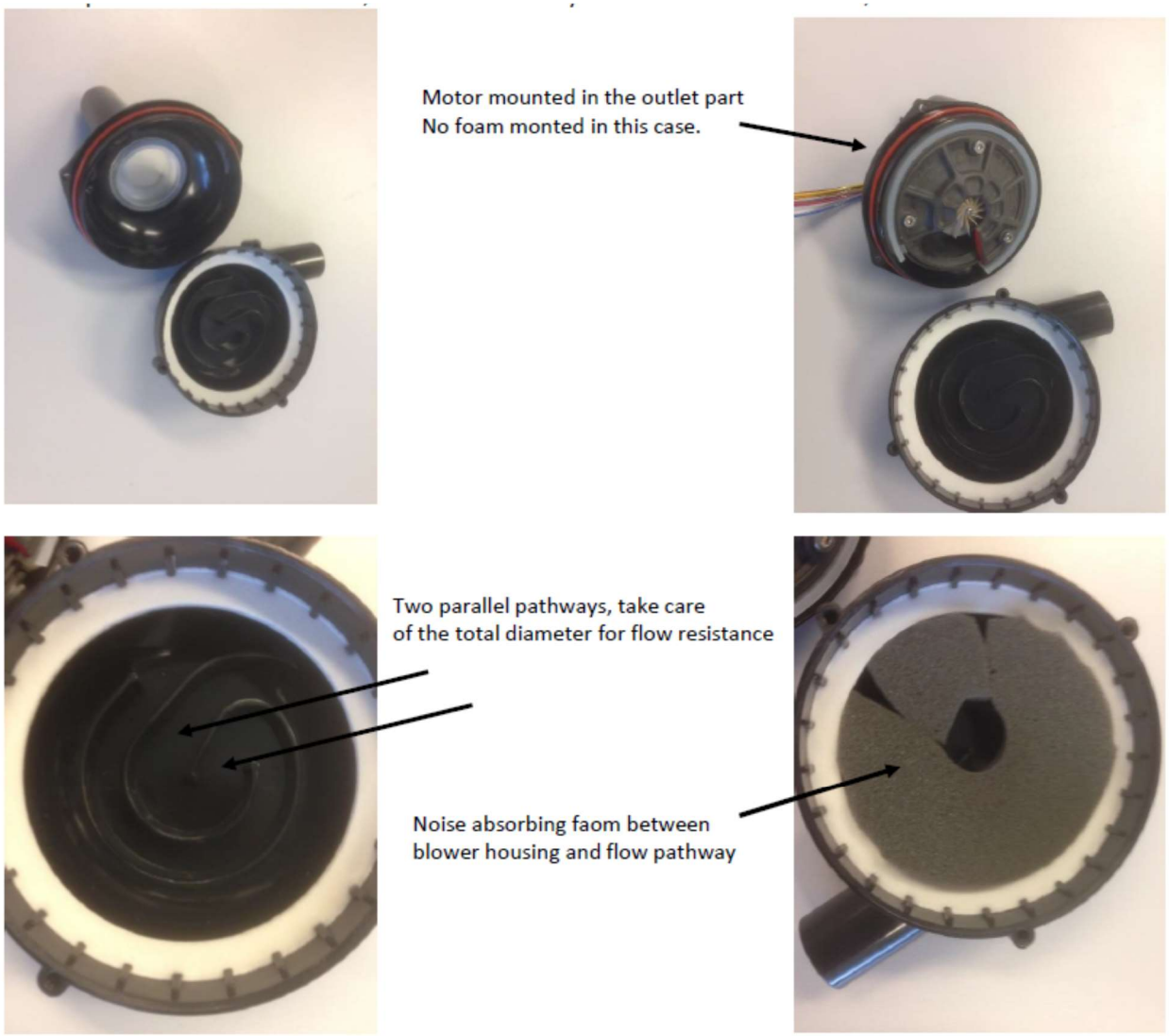
4.2.2 Inlet Filter construction

Very important to configure the inlet to the labyrinth from the inlet filter, Beware of the *Trumpet effect.*



4.2.3 Solution way 2 for Blower/Flow Noise absorption

Easier way of construction is to create path length by a labyrinth and bring adsorbing foam along the path. The blower is easier to mount in this 2 scale construction. The noise elimination is less effective as described in the first solution. On the inlet site, again take care of the Trumpet effect at the filter holder.



4.2.4 Elimination of Mechanical vibration from respiratory part to main housing

The Macawi high dynamic respiratory blower is very well balanced, but nevertheless it is possible that at certain blower speed some vibration is generated by the blower. For that purpose we advise the use of mechanical disconnection between the part where the blower is integrated in and the housing of your ventilator. This can be done with different available solutions like swing-adsorbers or flexible rings between the mounting screws and the brackets where the system is mounted to.

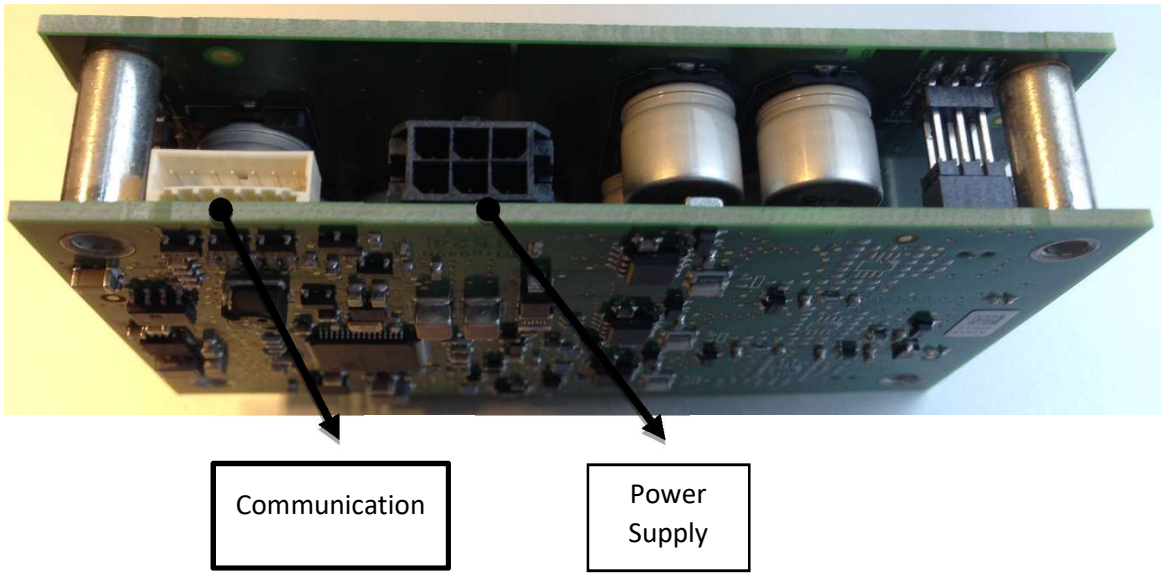
4.2.5 Module Controller part

The Module Controller has 4 mounting holes for mounting in a device.

CAUTION

We strongly advice to take measures to prevent from emission of lose particles out of your final assembled device. The Macawi blower which is constructed with the greatest precision can together with all the other elements you mount in the gas-pathway be a source of emission of lose particles. Our strict advice for the final test you do on your device is to set the blower at high speed with an open device for some time in order to have a very high flow through the complete device that can blow out the eventual lose particles from your device.

5. ELECTRICAL ARRANGEMENT



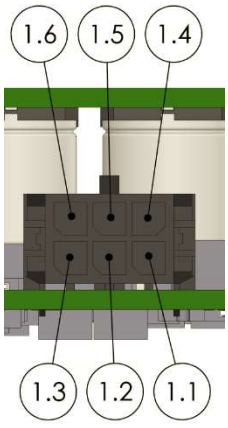
5.1 Power Supply (to be provided by the OEM)

The Controller requires an external power supply as specified in Technical Data

Connection to the Controller for the power supply is realized with a 6 way connector onto the PCB.

Mating crimp housing required Molex micro-fit 3.0 receptacle housing, 2row, 6 way part number 43025-0600.

Actuator power disable; is a control line to be able to stop the motor immediately by pulling this line down. When this line is made “0” the Motor Voltage will be disabled on the board and the microprocessor gets the information that the motor is **S**topped. The microprocessor will continue communication and will have the Status of **S**topped available on the communication interface.



1.1 GND

1.2 GND

1.3 GND

1.4 Actuator power disable

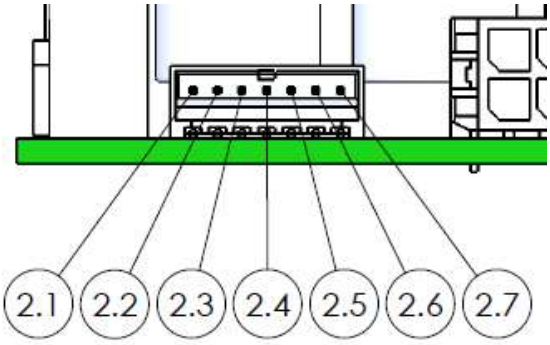
1.5 +24V DC

1.6 +24V DC

5.2 Communication

Connection to the Controller for the communication is realized with a 7 way connector onto the PCB.

Mating crimp housing required Molex Pico-Spox 1.5 mm receptacle housing, 1row, 7 way part number 87439-0700 with crimp terminal 87421-0000.



2.1 UART - TXD
2.2 UART – RXD
2.3 Tacho-out
2.4 Analog Speed Control + in
2.5 Analog Speed Control GND
2.6 GND
2.7 Shielding

UART – TXD : This line is the transmit line from the UART on the Panoramix PCBA. The line control voltage is 3.3V. The line is filtered with a π filter of 1 MHz and the processor output is safeguarded with clamp diodes between 3.3V and GND.

UART – RXD : This line is the receive line from the UART on the Panoramix PCBA. The line control voltage is 3.3V. The line is filtered with a π filter of 1 MHz and the processor input is safeguarded with clamp diodes between 3.3V and GND.

Tacho - out : This is an output line of the motor speed with 1 pulse per rotation. The line is configured as an open collector output and filtered with a π filter of 1 MHz the output transistor is safeguarded with a current limiter for max. current of 40 mA.

Analog Speed Control + in : This is an Analog control line, which in combination with the Analog Speed Control GND, can be used to have an accurate control of the motor speed. The input impedance between the Analog speed control + line and the Analog speed control GND line is >100 kΩ. The control is 0 – 5V which is 60 – 100.000 RPM. The line is filtered with a π filter of 1 MHz and the Amplifier input is safeguarded with clamp diodes between 5V and GND.

Analog Speed Control GND: This is the GND line for the Analog Speed control. This line shall be connected to GND at the source of the Analog control signal**.** This to prevent from offsets in the Analog Speed control signal caused by GND currents. The line is filtered with a π filter of 1 MHz and the Amplifier input is safeguarded with clamp diodes between 5V and GND.

GND : Is the connection to connect the GND of the board to the source of the communication.

Shielding : may be used to connect the shielding of the communication connection, caution shall be taken to prevent against ground loops.

6. PROTOCOL SPECIFICATION

This chapter defines the Respiratory Blower Protocol. Which type of messages the Respiratory Blower can send and receive is described in Chapter Message definitions.

Every packet is encoded in the same way. The classification of a packet – real-time or best-effort – only influences the scheduling of the packet transmission time by the Respiratory Blower. From the Client's perspective, there is no difference in encoding.

The protocol uses a layered approach, which is in a top-down fashion:

- *Message: application-specific information*
  For example, a request to control the motor speed towards a given set point. The semantics of the messages are defined in Chapter Message definitions.
- *Packet: a (retransmitted) best-effort request / best-effort response / real-time status*
  This layer implements flow control, and retransmissions of lost packets. The Respiratory Blower schedules packets based on whether the packet is either best-effort or real-time.
- *Frames: a packet carrier*
  This layer adds packet boundary synchronization, packet integrity checks, and payload encoding. A frame contains exactly one packet.
- *UART: a digital noisy communication channel*
  This layer allows transmission on byte-level.

The rest of this chapter defines the layers in a bottom-up fashion.

6.1 UART

The UART is by default configured using the following parameters:

- baud rate: 115200
- 8 data-bits
- 1 stop-bit
- no parity bits
- no flow-control

In this configuration, the channel has a bandwidth of roughly 115 bytes per 10 millisecond interval.

6.2 Frame

A frame has the following format:

1 to 253 bytes	1 byte	1 byte	1 byte
payload: encoded packet	CRC-8 (MSB)	CRC-8 (LSB)	ETB

Frames have a fixed maximum length of 256 bytes, including the CRC and end byte.

6.2.1 Payload

Frames carry encoded packets in their payload. At the frame level, the ASCII control characters (all characters with a hex value of 0x00 to 0x1F) are reserved. For the frame payload the characters with a hex value of 0x20 to 0xFF can be used.

6.2.2 CRC

The encoded payload is protected against bit-flips by means of a CRC-8 C2 with the following properties:

- The used polynomial is 0x97.
- Initialization set to 0, no reversal or final XOR.

The CRC is appended to the payload in a textual hex representation. To this extend, two bytes are used: the four MSB are encoded in the first byte using the corresponding ASCII character ('0'-'9', 'A'-'F'), the four LSB in the second. For example, the packet byte sequence ‘T’ ‘ !’ (0x54 0x21) has a computed CRC-8 of 0x75. The textual representation is '7' '5', which corresponds to the ASCII values 0x37 0x35.

The receiver shall verify the CRC of the frame's payload.

NOTE	
CRC-8 C2 can be checked on http://www.zorc.breitbandkatze.de/crc.html or http://dk.bg-elektronik.dk/support/crc/	
Settings :	
Crc order	**: 8**
Crc polynom	**: 0x97**

6.2.3 Frame's end

The frame is terminated by a byte with the value 0x17 (ETB). A frame is defined as all bytes between two ETB characters. ETB cannot occur as part of the payload, since such a payload byte is prohibited. Frames can be sent back-to-back; there is no need to insert an inter-frame gap after sending the ETB.

Frames always contain a payload and CRC. Therefore, at least three bytes are transmitted between ETBs. If the receiver gets fewer bytes, the frame can be considered invalid and shall be discarded. Moreover, when the receiver gets more than 256 bytes, without an intermediate ETB, the frame is invalid and shall be discarded. All bytes up to the next ETB shall be discarded as well.

A sender may send just an ETB byte (without previous payload or CRC) to synchronize with the receiver. This can be used after power-up, to make sure that no previously generated noise or garbage will be seen as part of the first transmitted frame by the receiver.

For example, the full transmission of the payload ‘T’ ‘!’ will be the sequence 0x54 0x21 0x37 0x35 0x17.

6.3 Packet

The frame's payload is a packet. A packet is defined as follows:

1 byte			0 or more bytes
1 bit R	2 bit T	5 bit ID	payload: packet-specific data

The first byte of the packet is the 'type byte'. It contains three fields:

- R-field: indicates that this packet is a retransmission, where the T, ID and payload is identical to one or more earlier transmissions.
- T-field: the type of this packet, which is defined as:
  - 0: reserved
  - 1: (real-time) status packet
  - 2: request packet, which shall be answered by the receiver by means of a response packet
  - 3: response packet, which is a response to an earlier received request packet
- ID-field: a message-specific ID, which is defined in Chapter Message definitions.

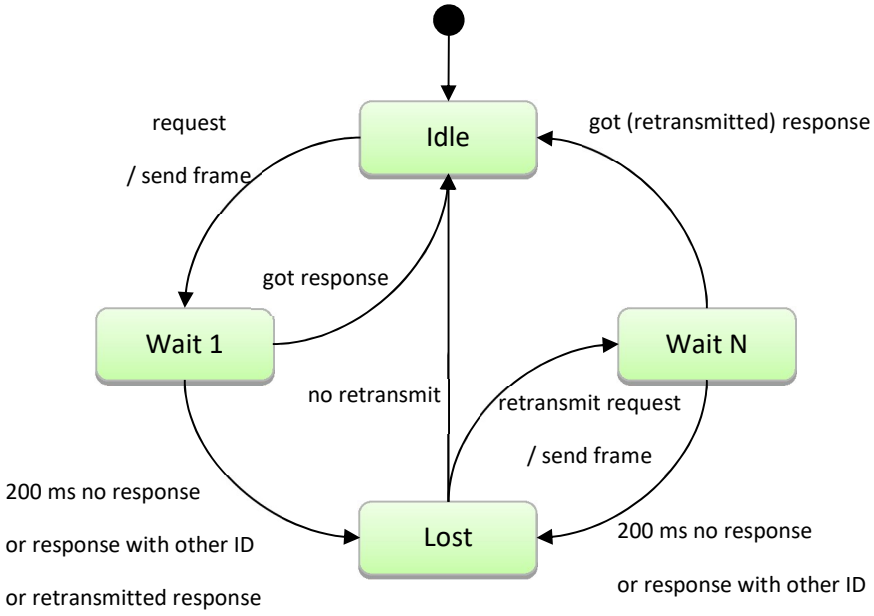
All other bytes in the packet are the payload. It contains packet-specific data, which is defined in Chapter Message definitions.

A response to a request shall have the same ID. The IDs of status packets are unrelated to those of requests/responses.

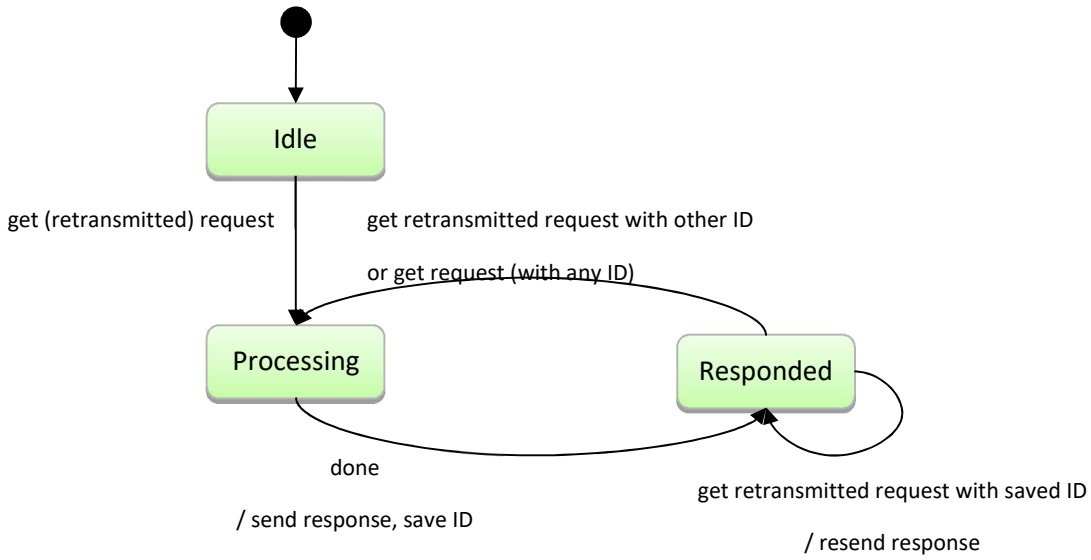
The format of the type byte is chosen such that its ASCII representation is easily distinguishable and recognizable. For example, the type of a request packet with ID 1 is the byte value 0x41, which is the character 'A'. A response to this request is the response with ID 1, which is the type byte 0x61, which in turn is the character 'a'. Status packets have a type byte, which corresponds to characters like '#', '$', and '='.

6.3.1 Response time and Retransmissions

The sender of a request shall receive a response within 200 milliseconds after transmission of the full frame, including the ETB byte. When the response does not arrive in time, the sender shall assume that either the request or the response is lost. Then, the sender may retransmit the request. The sender may also decide not to retransmit at all. The intended state machine of the sender is depicted below.



For the sender, it is unknown whether the request or the response got lost. Therefore, the receiver may get an already processed request again. In that case, it shall resend the response, without reprocessing the request. The intended state machine of the receiver is depicted below.



Retransmission can be used for requests that shall be executed exactly once. There is a sequence of packets and states, which will violate this. Consider the following example communication sequence, where the 'client' tries to execute a command exactly twice by the 'server':

1 client -> server: send request
2 server -> client: send response to the request (but it gets lost)

The client does not receive a response in time and decides to retransmit.

3 client -> server: retransmit request

The server receives a retransmitted request, which it already sent a response to. It resends the response without executing the request again.

4 server -> client: retransmit response
Now, the client sees correct execution of the request. Next, the client wants to issue the same request for a second execution.

5 client -> server: send another request with the same ID as the previous one (but it gets lost)

The client does not receive a response in time and decides to retransmit.
6 client -> server: retransmitted request
Now the server sees a second retransmission of the first request. Like before, it resends the response without executing the request again.

7 server -> client: retransmitted response

The client now receives a retransmitted response, which can mean that operation 5 got lost or the response to 5 got lost. The client cannot determine whether the request has been executed once or twice.

6.3.2 Flow control

To prevent excessive buffering, the implementation shall adhere to the following rules:

- A (retransmitted) response packet shall only be sent as an answer to a (retransmitted) request packet; responses cannot be sent 'out of the blue'.
- A (retransmitted) request packet shall only be responded once by either a response or a retransmitted response.
- No request shall be sent during the 200 ms timeout period of the previous request, unless a valid response to the previous request has been received.
- Status packets may always be sent, regardless of the state of concurrent requests/responses.

Hence, multiple concurrent requests are not guaranteed.

NOTE

The client can send requests without waiting for a response, the Respiratory Blower will execute the request. It is possible in this case that the Respiratory Blower will drop requests or responses.

6.4 Timeout

When the UART interface is used to control the system, it is essential that the connection between the Client and Respiratory Blower remains stable. To this extend, the Respiratory Blower keeps track of the time since the last request it received. If no request is received within a specific timeout period, the connection is presumed to be lost and the motor is stopped. This timeout is set to 500 ms, which is 2.5 times the packet response timeout, so packets can be retransmitted twice before the connection timeout expires. The Client can prevent auto-disconnection by sending any request. Among others, the Echo request (without payload, see Section Echo ('E')) can be used to generate dummy requests.

The motor can only be controlled by one input at the time. When one is selected, all other inputs are ignored. By default the system starts in analog ('A') mode.

6.5 Message definitions

The Respiratory Blower always sends real-time status updates, and can respond to requests. The Respiratory Blower will never send a request to the Client.

Messages that are malformed should be considered invalid and discarded. Therefore, an invalid request shall not be responded to.

For many messages, an '*n*-byte textual representation of an *m*-bit value' is used, where *n* equals *m*/4. This shall be interpreted as follows. Let us assume that *n* is 4 and *m* is 16. These bytes are ordered as depicted below:

4 bytes			
w	*x*	*y*	*z*

Every byte, *w*, *x*, *y*, and *z*, is an ASCII value in the range '0' (0x30) to '9' (0x39), or 'A' (0x41) to 'F' (0x46). Combined they form the hexadecimal value 0x*wxyz*. Intuitively, when one sees the raw packet on the screen, the 16-bit hex value is human-readable. For example, the ASCII character string "6A3C" is the byte sequence 0x36 0x41 0x33 0x43, which represents the hex value 0x6A3C and the decimal value 27196. Values are always unsigned, unless explicitly stated. In case of signed values, 2-complements representation is used. All values are big-endian.

6.6 Requests

Every request and response starts with a byte which contains the ID. This is the same byte as the first byte of the packet (as discussed in Section Packet) except for the retransmission bit, which is specific to the packet layer. For example, if the first byte is 'V' (ASCII value 0x56), the ID is actually 0x16 and the packet's T-field is set to 'request packet'. Still, a type byte of 0x56 | 0x80 = 0xD6 is used upon retransmission.

6.7 Version information ('V')

6.7.1 Request:

1 byte							
7	6	5	4	3	2	1	0
R	T		ID				
0	b10		b10110				
'V'							

6.7.2 Response:

1 byte								1 byte	4 bytes	4 bytes	4 bytes	4 bytes
7	6	5	4	3	2	1	0	Protocol version	SW major	SW minor	HW major	HW minor
R	T		ID									
0	b11		b10110					0x31				
'v'								1				

The Protocol version is the ASCII character '1' (0x31), which corresponds to the definition in this document. Other values are reserved for future use.

The software version (SW) and hardware version (HW) both consist of a major and minor version, which all are a 4-byte textual representation of a 16-bit value. They are form the version number as: *major.minor*

6.8 Part information ('P')

6.8.1 Request:

1 byte							
7	6	5	4	3	2	1	0
R	T		ID				
0	b10		b10000				
'P'							

This request is reserved for future application.

6.8.2 Response:

1 byte								4 bytes	4 bytes	4 bytes	4 bytes
7	6	5	4	3	2	1	0	part major	part minor	serial major	serial minor
R	T		ID								
0	b11		b10000								
'p'											

6.9 Echo ('E')

6.9.1 Request:

1 byte								(any length)
7	6	5	4	3	2	1	0	payload
R	T		ID					
0	b10		b00101					
'E'								

This request echo's the payload back to the Client. The payload can be of any length and does not have a specific meaning. The command can be used to check the communication link, like sending a ping or keep-alive command.

6.9.2 Response:

1 byte								(any length)
7	6	5	4	3	2	1	0	payload
R	T		ID					
0	b11		b00101					
'e'								

The payload of the response is identical to the payload in the request packet.

6.10 Motor control input mode ('C')

6.10.1 Request:

1 byte								1 byte
7	6	5	4	3	2	1	0	mode
R	T		ID					
0	b10		b00011					
'C'								

The mode byte shall be either the ASCII character 'U' (0x55) or 'A' (0x41), where 'U' means that the motor speed is controlled over commands over the UART interface, and 'A' means control by the analog input to Respiratory Blower.

See section Timeout for timeout restrictions when using uart mode.

6.10.2 Response:

1 byte								2 byte	1 byte
7	6	5	4	3	2	1	0	Error code	mode
R	T		ID						
0	b11		b00011						
'c'									

The response contains the mode byte that the system switched to. In normal conditions, this will be the same mode as requested. However, when an error occurred, Respiratory Blower may choose a different mode. The error code is a 2-byte textual representation of an 8-bit value which contains the error code (if any). When the error is 0, the requested mode is always selected.

6.11 Set motor speed ('R')

6.11.1 Request:

1 byte								6 byte
7	6	5	4	3	2	1	0	RPM
R	T		ID					
0	b10		b10010					
'R'								

The RPM is a 6-byte textual representation of a 24-bit value which is the requested revolutions per minute. Respiratory Blower accepts values from 0 to 150000. When a value outside of the supported range is requested, the closest valid value is automatically selected.

This command is only valid when the UART input mode (see Section Motor control input mode ('C')) is selected.

6.11.2 Response:

1 byte								2 byte
7	6	5	4	3	2	1	0	Error code
R	T		ID					
0	b11		B10010					
'r'								

The response only contains the error code while executing the request. When the request has been executed successfully, the motor may not run at the requested speed (yet). To retrieve the current speed, see Section Get tag ('T'). The error EPERM is returned when another motor control input mode than UART has been selected.

6.12 Get tag ('T')

6.12.1 Request:

1 byte								1 byte
7	6	5	4	3	2	1	0	tag
R	T		ID					
0	b10		b10100					
'T'								

This requests the current value of the specified tag. The tag and values are defined in Section Status. For example, to request the current motor speed, the request T= can be issued.

6.12.2 Response:

1 byte								1 byte	(any length)
7	6	5	4	3	2	1	0	tag	tag-specific payload
R	T		ID						
0	b11		b10100						
't'									

The tag reflects the tag in the request. The payload contains the value, equivalent to as defined in Section Status. For example, when the motor speed is requested, the response can be t=00E652, which indicates a current speed of 58962 RPM. When the requested tag is not supported, the payload is empty.

6.13 Status update configuration ('S')

6.13.1 Request:

1 byte								4 byte	(any length)
7	6	5	4	3	2	1	0	interval	tags
R	T		ID						
0	b10		b10011						
'S'									

The interval and content of the real-time status update packets can be configured. The interval is a 4-byte textual representation of a 16-bit value, which indicates the interval in milliseconds in which status updates are transmitted by the Respiratory Blower. The tags are a sequence of measurements that should be reflected in the next status interval. See Section Status for details. Every successive status packet contains the (supported) tags in the order in which they are requested.

The Respiratory Blower may reflect the requested changes to the status packets before the response packet is sent back to the Client.

The default interval is 10 ms and no additional tags.

Real-time response is dropped when a pending real-time response still is in progress. This occurs when the real-time response doesn't fit in the interval time because of baud rate setting and number of tags.

An example request for the current temperature and speed is: S000A!=

6.13.2 Response:

1 byte								2 byte
7	6	5	4	3	2	1	0	Error code
R	T		ID					
0	b11		b10011					
's'								

The response only contains the error code while executing the request. The error E2BIG is returned when not all specified tags can be supported in a single status message. Any superfluous tags may be ignored. The error ENOENT is returned when one or more tags are not supported. The other tags will be reflected in successive status messages.

6.14 Switch motor control state ('Z')

6.14.1 Request:

1 byte								1 byte
7	6	5	4	3	2	1	0	state
R	T		ID					
0	b10		b11010					
'Z'								

The state byte shall be one of the following values:

- 'A' (0x41): active
  Drive motor at a given speed. The motor may be running at least at a certain speed to let it quickly respond when a different speed is requested.
- 'I' (0x49): idle
  The motor may be stopped completely. It may take some time to get active again when a new speed set point it given.
- 'R' (0x52): reboot
  Reboot Respiratory Blower.

6.14.2 Response:

1 byte								2 byte	1 byte
7	6	5	4	3	2	1	0	Error code	state
R	T		ID						
0	b11		b11010A						
'z'									

The response contains the state byte that the system switched to. In normal conditions, this will be the same mode as requested. However, when an error occurred, Respiratory Blower may choose a different state. The error code is a 2-byte textual representation of an 8-bit value which contains the error code (if any). When the error is 0, the requested state is always selected. The error EINVAL indicates that the requested state is invalid. When another control mode than UART is chosen, trying to set the mode to active or idle will result in the error EPERM. When Respiratory Blower cannot comply because it is running other tasks, such as a self-test, EBUSY is returned.

When a reboot is requested, the system may not respond to the request and may reboot immediately. When a reboot request is the first request Respiratory Blower receives after it is booted, it is responded to with error code 0, without rebooting again. So, the Client has to retransmit the reboot request at least once in order to get a proper response.

6.15 Firmware update ('F')

6.15.1 Request:

1 byte								2 byte	(any length)
7	6	5	4	3	2	1	0	sequence	Part of xml file
R	T		ID						
0	b10		b00110						
'F'									

The firmware can be updated using this request. The sequence is a 2-byte textual representation of an 8-bit value. The remaining payload is a part of a firmware image (XML) file. The Client shall use this command as follows:

- To request to start updating the firmware, the Client sends a request with sequence number 0 and no additional payload. The Respiratory Blower responds with sequence and error code 0.
- Next, the Client sends the complete XML file in chunks of any length (up to the allowable frame size per request). Every request shall be an 'F' request with an increasing sequence number. The Respiratory Blower responds with the same sequence number and error code. The sequence number shall restart from the value 1 after the value of 254 is used.
- The last (possibly empty) part of the image shall have the sequence number 255. The Respiratory Blower responds with the same sequence number and error code. Then, the Respiratory Blower installs the firmware and reboots.

6.15.2 Response:

1 byte								2 byte	2 byte
7	6	5	4	3	2	1	0	sequence	Error code
R	T		ID						
0	b11		b00110						
'f'									

The sequence number is identical to the one used in the request message. The error code indicates the result of processing the last part of the image. When the Respiratory Blower responds with error 0 to the request with sequence 0 (start of firmware update), all update message may be suppressed. When another error is returned, the state of the system remains unchanged.

When after the initial request an error occurs, the Respiratory Blower may remain in a dysfunctional state. Any other request than the Echo and Firmware update may not be responded to.

The error EINVAL is returned when the sequence number is invalid. The error EBADMSG indicates an invalid XML file. EIO indicates an error while writing to the flash memory. The last request may be responded to with error ENODATA to indicate that the XML file did not contain any usable software for this device. ENOENT indicates that the uploaded software is not bootable.

Before the software upload can start, the default baud rate of 115200 must be selected before. If the UART interface is running at another speed, the error EPERM is returned.

6.16 Status

The real-time status update packet contains a sequence of tags. Which tags are reflected in the status can be configured (see Section Status update configuration ('S')). The Respiratory Blower will always send tag $ first. Every tag has one payload field. The table below list the tags, payload size and description.

tag	**payload**	**description**
$	1-byte [a]	Actual system state: 'B' (booting), 'A' (active), 'I' (idle), 'S' (stopped), 'U' (firmware upload in progress)
#	4-byte/16-bit [b]	Actual event code
=	signed 6-byte/24-bit [b]	Actual motor speed, in RPM
!	signed 2-byte/8-bit [b]	Actual motor temperature: in range -78 °C – + 177 °C (**using offset of 50 °C**)
>	signed 4-byte/16-bit [b]	Peek current through motor during last rotation, in mA
<	signed 4-byte/16-bit [b]	Motor voltage, in mV
?	4-byte/16-bit [b]	Status message counter

[a] single character payload

[b] *n*-byte textual representation of an *m*-bit value

For example, when the status is configured using S000A!=, the status packet $A!12=00E652 indicates an active state, a current temperature of 18 °C + 50 °C offset = 68 °C, and a speed of 58962 RPM.

6.16.1 Event codes

The 32-bit event code value has the following format:

bit 31-19	bit 22-20	bit 19	bit 18-16	bit 15-14	bit 13-8	bit 7-6	bit 5-4	bit 3	bit 2	bit 1-0
reserved	CR	reserved	WE	*reserved*	HE	SV	MT	PD	MB	ST

- *CR, Capacitor board revision:* the revision, where an unconnected board is reported as revision 7
- *WE, Winding Error state:* combination of the following flags:
  - bit 16: when 1, the system detected a failure in motor winding 1
  - bit 17: when 1, the system detected a failure in motor winding 2
  - bit 18: when 1, the system detected a failure in motor winding 3

- *HE, Hall sensor Error state:* combination of the following flags:
  - bit 8: when 1, the Hall sensor 1 is stuck at a high value
  - bit 9: when 1, the Hall sensor 2 is stuck at a high value
  - bit 10: when 1, the Hall sensor 3 is stuck at a high value
  - bit 11: when 1, the Hall sensor 1 is stuck at a low value
  - bit 12: when 1, the Hall sensor 2 is stuck at a low value
  - bit 13: when 1, the Hall sensor 3 is stuck at a low value
- *SV, Motor Voltage state:* contains one of the following values:
  - 0: no voltage measurement available
  - 1: voltage is OK
  - 2: voltage is low
  - 3: voltage is too low
- *MT, Motor Temperature state:* contains one of the following values:
  - 0: no temperature measurement available, so the !-tag contains invalid data
  - 1: temperature is OK
  - 2: motor is too hot
  - 3: motor is overheated
- *PD, motor Power Disabled*: when 1, the motor power is disabled by external pin
- *MB, Motor Blocked state:* when 0, the rotor is free; when 1, the rotor is blocked
- *ST, Self-Test state:* contains one of the following values:
  - 0: not tested
  - 1: test in progress
  - 2: test failed
  - 3: test succeeded

6.17 Error codes

error code	**description**
0x00	No error
0x01	EPERM
0x02	ENOENT
0x05	EIO
0x07	E2BIG
0x10	EBUSY
0x16	EINVAL
0x3D	ENODATA
0x4D	EBADMSG
0x58	ENOSYS
0xFF	Unknown error

7. TECHNICAL DATA

7.1 Motor Controller

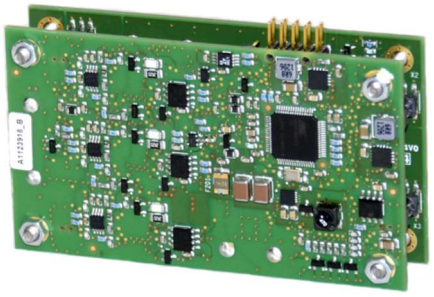
ELECTRICAL OPERATING CONDITIONS		
Property	Value	Remark
Power supply voltage	24V DC ±10%	
Peak current	up to 6A	at 24V max. during max. 200 msec. at maximum inspiratory pressure setting and restricted flow condition.
Continuous current	up to 2.5A	at 24V during inspiration at maximum inspiratory pressure setting and restricted flow condition
Nominal power consumption	5 - 30 VA	at 24V depending on ventilation settings and patient

ENVIRONMENTAL OPERATING CONDITIONS		
Property	Value	Remark
Temperature	-20 …+ 60 °C	-
Relative air humidity	5 … 95% R.H.	-
Ambient Pressure	600 … 1060 hPa	

ENVIRONMENTAL TRANSPORT AND TRANSPORT CONDITIONS		
Property	Value	Remark
Temperature	-20 … + 70 °C	-
Relative air humidity	5 … 95% R.H.	Non Condensing
Ambient Pressure	600 … 1060 hPa	

DIGITAL INTERFACE		
Property	Value	Remark
Serial interface	UART compatible	3.3V
Commands	• Speed control mode • Set speed • Get speed and status info • Firmware update	Speed control Digital or Analog - - -
Digital speed control	0 – ~100.000 RPM	no exact speed control depends on load
Analog speed control	0 – 5V DC = 0 -~100.000 RPM	no exact speed control. Standard active from start up
Digital output tacho signal	1 pulse per rotation	open collector
Digital input motor disable	0 V	Switch this input to gnd. This line is part of Power connector

DIMENSIONS		
Property	Value	Remark
Controller	86mm x 50mm x 21mm	2 boards as piggyback connected



7.2 Blower Performance

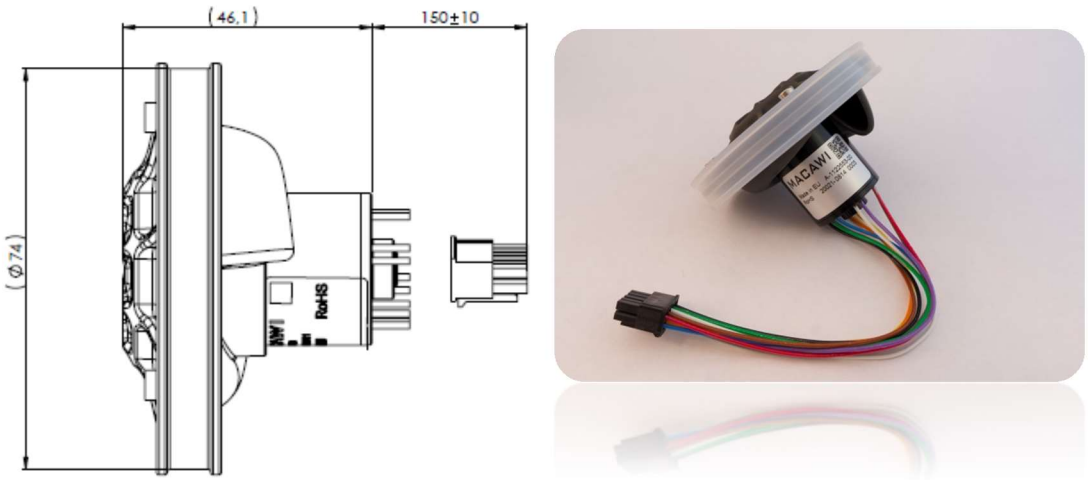
Motor: electronically commutated motor with hall sensors. External motor driver required.

BLOWER PROPERTIES	
Property	Value
U_N	24 V_{DC}
$P_{el.}$ at 2 kPa at 0 flow	~4 W (1
$P_{el.}$ at 6 kPa at 0 flow	~17 W (1
Max. Pressure at 0 flow	>10 kPa(2
Max. output flow	>300L/min. (1
Max. speed	90.000 RPM
Pressure response	>5 kPa / 100 msec. (1 (dependent of resp. circuit compliance)
Operating temperature	-20 … + 60 °C
Operating relative air humidity	5 … 95% R.H.
Operating ambient pressure	600 … 1060 hPa
Temperature during storage and transport	-20… + 70 °C
Relative air humidity during storage and transport	5 … 95% R.H. (non Condensing)
Ambient pressure during storage and transport	600 … 1060 hPa
Expected Life time at moderate ventilation loads	> 40.000 hrs.(B10)
Gas media compliancy	Blower design for use in Oxygen enriched environment
Dimensions	Ø 75 mm x 55 mm
Blower mass excl. electronics	< 150 gr.

(1 at 101 kPa environmental pressure

(2 not continuously, depends on thermal management

7.3 Mechanical Data



Dimensions in mm

Grommet = Silicon ring as seal, mechanical vibration damper and mounting aid.

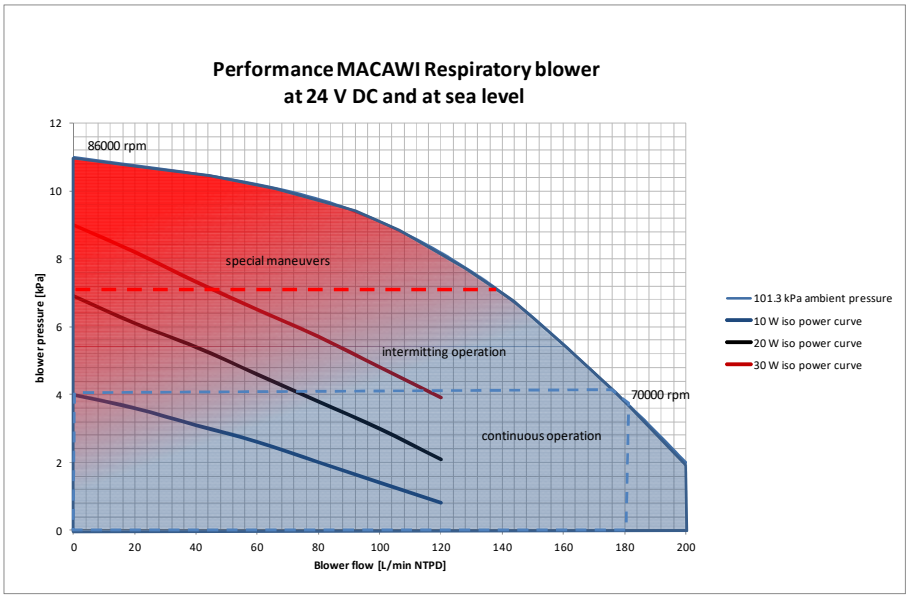


Figure 1 Maximum performance pressure / flow

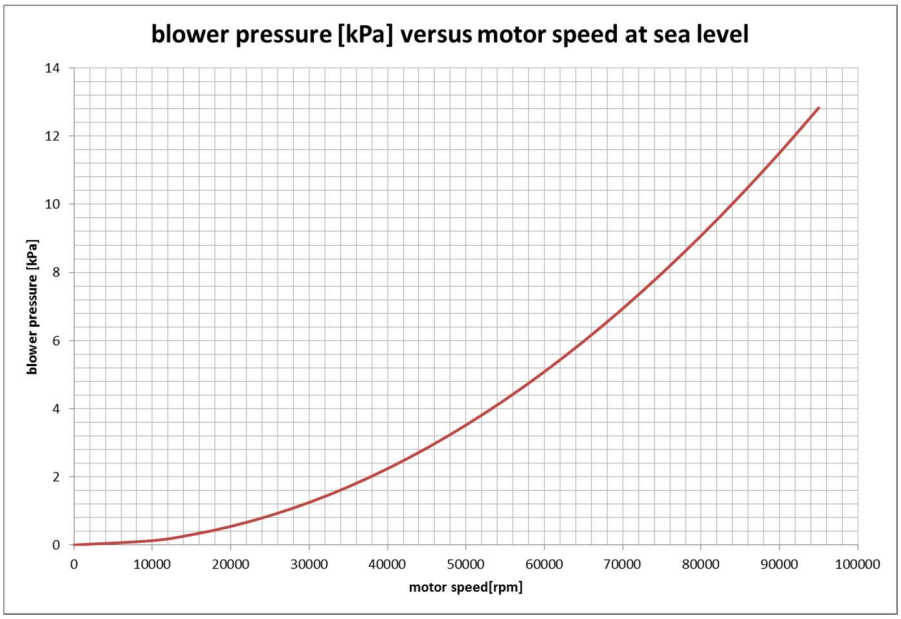


Figure 2 Pressure - Motor speed characteristic

Note:

The shown data is only applicable for dynamic use of the blower. The design is not meant for static high pressure high flow situations because of the maximum motor power of the chosen BLDC Motor.

8. CHANGE LOG

CHANGE LOG		
Revision	Date (yyyy-mm-dd)	Description of change(s)
1.0	2014-10-29	First release
2.0	2014-12-05	Second release
3.0	2016-07-18	Third release
4.0	2017-01-03	Fourth release
5.0	2018-02-16	Fifth release
6.0	2019-06-25	Sixth release