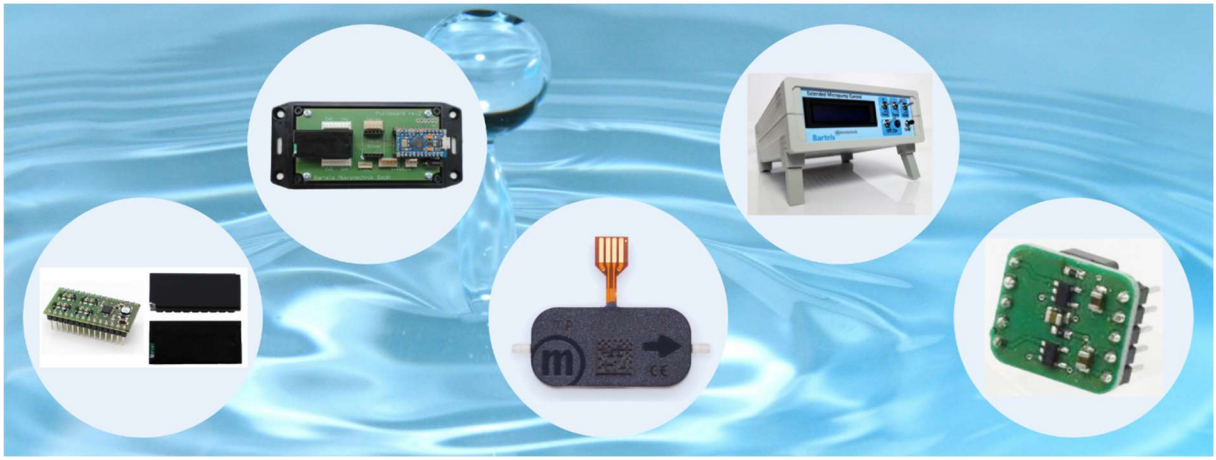
Datasheet mp6 electronics



Summary:
This document contains detailed information on all electronic by Bartels Mikrotechnik for the mp6-micropump series.

Bartels electronics comprises of laboratory solutions with access to the full range of driving parameters down to smaller solutions for smart micro controller applications.

The evaluation board allows quick and flexible setups with our three integrable pumpdriver options with various flow and pressure sensors. Furthermore, we make an easy and user-friendly access possible with user-interfaces and Open-Source Codes for easier tinkering and prototyping for our customers.

1 Contents

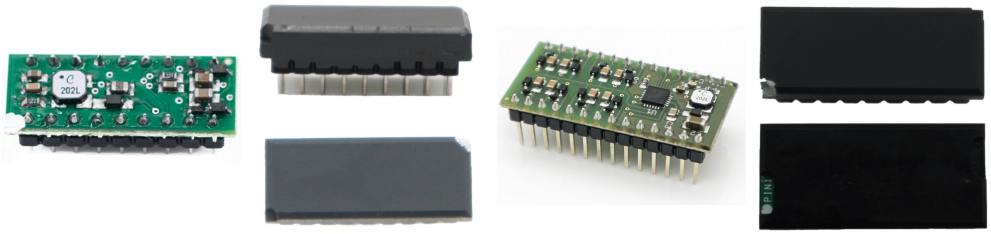
2 Electronic drivers for mp6 series: Highdriver and Lowdriver platforms

2.1 Highdriver platforms: mp-Highdriver/-Highdriver4

The mp-Highdriver is a small, easy to use driving circuit developed for the micropumps of the mp6-series especially the mp-Highdriver4 as it can drive up to four mp6 pumps simultaneously. It generates amplitudes up to 250 Vpp from a 3-5 V DC supply.

Its low power consumption makes it ideal for battery powered handheld devices or even solar powered devices.
The build-in interface allows the user to adapt frequency, amplitude and signal-shape to its application by the use of a few additional components or a microcontroller.

In order to locate Pin 1, please refer to the following figure. The pin is marked with a colored spot or triangular marking on the corner of the PCB.



Picture 1 mp-Highdriver and mp-Highdriver4

The Highdriver platform offers a high-voltage DC-AC converter that drives mp6 pumps. The device features a 2.7 V to 5.5 V input range that allows the device to accept a variety of sources such as single-cell Li+ batteries. The outputs of the device generate up to 250 VPP for maximum pump performance. The Highdriver platform utilizes a high-frequency spread-spectrum boost converter that reduces the amount of EMI/EFI generated by the circuit. The boost-converter switching frequency is set with an 8-bit register through the I2C interface. The mp-Highdriver4 uses a high-voltage full-bridge output stage to convert the high voltage generated by the boost converter to an AC waveform suitable for driving a piezo membrane pump. An internal register controlled through the I2C interface sets the shape of the output waveform. The output switching frequency for all outputs is set with an 8-bit register through the I2C interface. The Highdriver platform provides a serial digital interface that allows the user to set the peak voltage of their outputs independently with 5 bits of resolution. Highdriver platform) also provides an adjustable automatic ramping feature that slowly increases or decreases the peak output voltage when the set value is changed. The slew rate of the ramp is set with 3 bits of resolution through the I2C interface and it is independent for each channel. The high-voltage outputs are ESD protected up to ±15 kV Human Body Model, ±8 kV Air Gap Discharge, and ±6 kV Contact Discharge, as specified in IEC 61000-4-2.

2.1.1 Output Voltage

The shape, slope, frequency, ramp-on/-off times, and peak-to-peak voltage of the Highdriver platform lamp outputs are programmed using internal registers. The Highdriver platform is capable of producing output waveforms with varying shapes and slew rates. The user sets the shape and slew rate of the output using bits in the shape registers. The Highdriver platform output frequency uses an internal oscillator to set the desired frequency. The output frequency is adjusted by the FO[7:0] bits of the output frequency register. The frequency increases and decreases linearly with FO[7:0]. The peak-to-peak voltage of the output is varied from zero to 250 VPP by programming the VO_ _[4:0] bits of the ramping time and output peak voltage registers. The peak-to-peak voltage increases and decreases linearly with VO_ _[4:0]. The Highdriver platform also features a slow fade-on and slow fade-off time feature, it is programmed by the RT_ _ [2:0] bits of the ramping time and output peak voltage registers. This slow fade-on/-off feature causes the peak-to-peak voltage of the outputs to rise slowly from the previously set value to the maximum set value. This feature also causes the peak-to-peak voltage of the outputs to fall from the maximum

set value to zero when the device is placed into shutdown. The slow rise and fall of the peak-to-peak output voltage creates a soft fade-on and fade-off of the mp6 pumps.

2.1.2 Boost Converter

The Highdriver platform boost converter consists of an external-tapped inductor from VDD to the LX input, an internal DMOS switch, an external diode from the secondary of the tapped inductor to the CS output, an external capacitor from the CS output to GND, and a mp6 pump connected to each output. When the DMOS switch is turned on, LX is connected to GND, and the inductor is charged. When the DMOS switch is turned off, the energy stored in the inductor is transferred to the capacitor CCS and the mp6. Note: The Highdriver platform exhibits high-voltage spikes on the LX node. The addition of a snubber circuit to the LX node protects the device by suppressing the high voltage spikes. The values of RSN and CSN should be optimized for the specific tapped inductor used. Typical values are RSN = 20Ω and CSN = 330pF. The Highdriver platform boost-converter frequency uses an internal oscillator to set the frequency of the boost converter. The oscillator frequency is adjusted by the FSW[4:0] bits of the boost-converter frequency register. The boost converter increases and decreases linearly with FSW[3:0]. To further reduce the amount of EMI/EFI generated by the circuit, the boost-converter frequency can be modulated (see the SS[1:0] bits of the boost-converter frequency register). Enabling modulation spreads the switching energy of the oscillator in the frequency domain, thus decreasing EMI.

2.1.3 Independent Dimming Control

The performance of the mp6-series pumps is proportional to the peak-to-peak voltage applied. The Highdriver platform provides four registers to control the peak-to-peak voltage of each output using the VO_ _[4:0] bits of the ramping time and output peak voltage registers.

2.1.4 Signal Output Waveshape

The Highdriver platform can produce sine-wave to square-wave waveshapes on the outputs by varying the slope of the outputs. This is achieved by using bits SL[1:0] of the shape register. If the shape configuration is set to sine and if all outputs have the same amplitude settings, then each output has a sinusoidal waveshape. If the outputs have different amplitude settings, then the output with the highest setting has a sine waveshape while the remaining outputs have a clamped sine waveshape.

2.1.5 Shutdown

The Highdriver platform can be placed in shutdown by writing a '0' to the EN bit of the system register. When activating shutdown, the pump outputs are shut down; however, the register contents remain unchanged.

2.1.6 Undervoltage Lockout (UVLO)

The Highdriver platform has a UVLO threshold of +2.0V (typ). When VDD falls below +2.0V (typ), the device enters a non-operative mode. The contents of the I2C registers are not guaranteed to remain unchanged below UVLO.

2.1.7 Thermal Protection

The Highdriver platform enters a non-operative mode if the internal die temperature of the device reaches or exceeds +160°C (typ). The mp-Highdriver4 is latched, and only cycling the power supply of the Highdriver platform resets the thermal protection bit as well as all registers.

2.1.8 I2C Registers and bit descriptions

Ten internal registers program the mp-Highdriver4. Table 1 lists all the registers, their addresses, and power-on reset states. All registers are read/write. Register 0x0A is reserved as a command to update all signal peak voltage output registers. Register 0x0B is reserved and should not be used.

Table 1 Register map mp-Highdriver

REGISTER	B7	B6	B5	B4	B3	B2	B1	B0	REGISTER ADDRESS	POWER-ON RESET STATE
SYSTEM										
Device Id	DEVID3	DEVID2	DEVID1	DEVID0	REV3	REV2	REV1	REV0	0x00	0xB2
Power Mode	OVR TEMP[1]	X	X	X	X	X	X	EN	0x01	0x00
OUTPUT FREQUENCY										
Output Frequency	FO7	FO6	FO5	FO4	FO3	FO2	FO1	FO0	0x02	0x00
Signal Shape										
Slope/Shape	X	ENDAMP	X	X	SHAPE1	SHAPE0	SL1	SL0	0x03	0x00
BOOST-CONVERTER FREQUENCY										
Boost-Converter Frequency	SS1	SS0	X	FSW4	FSW3	FSW2	FSW1	FSW0	0x04	0x00
AUDIO[2]										
Audio Effects	FR_AM	NO_SAMPLE	AUXDIV1	AUXDIV0	AU4	AU3	AU2	AU1	0x05	0x00
SIGNAL RAMPING TIME AND PEAK VOLTAGE										
Ramping Time and Peak Output Voltage CH1[3]	RT1_2	RT1_1	RT1_1	VO1_4	VO1_3	VO1_2	VO1_1	VO1_0	0x06	0x00
Ramping Time and Peak Output Voltage CH2[3]	RT2_2	RT2_1	RT2_1	VO2_4	VO2_3	VO2_2	VO2_1	VO2_0	0x07	0x00
Ramping Time and Peak Output Voltage CH3[3]	RT3_2	RT3_1	RT3_1	VO3_4	VO3_3	VO3_2	VO3_1	VO3_0	0x08	0x00
Ramping Time and Peak Output Voltage CH4[3,4]	RT4_2	RT4_1	RT4_1	VO4_4	VO4_3	VO4_2	VO4_1	VO4_0	0x09	0x00

X = Don't Care
[1]Read back only
[2]Audio functions are not supported, as the audio input pin is not available. Keep the register value at the default (0x00)
[3]Send command 0Ah (update all signal ramping time and peak voltage registers) to have the programmed voltage effectively applied to the pumps
4Only this channel output is used for mp-Highdriver.

2.1.9 Slave address

The Highdriver platform device address is set through external inputs. The slave address consists of five fixed bits (B7–B3, set to 11110) followed by two input programmable bits (A1 and A0). For example: If A1 and A0 are hardwired to ground, then the complete address is 1111000. The full address is defined as the seven most significant bits followed by the read/write bit. Set the read/write bit to 1 to configure the mp-

Highdriver4 to read mode. Set the read/write bit to 0 to configure the Highdriver platform to write mode. The address is the first byte of information sent to the mp-Highdriver4 after the START condition.

2.1.10 System Registers (0x00, 0x01)

Table 2 System Registers

Device (DEVID3/DEVID2/DEVID1/DEVID0)	ID	DEVID[3:0] is preprogrammed to 1011 to identify the mp-Highdriver4; see Table 3.
Revision (REV3/REV2/REV1/REV0)		REV[3:0] is preprogrammed to the current revision of the mp-Highdriver4 and is REV[3:0] = 0010.
System Over temperature (OVRTEMP)		1 = Thermal shutdown temperature exceeded. 0 = Analog circuitry operating properly. OVRTEMP = 1 turns the outputs off. To set OVRTEMP to 0 and restart in default condition (all register reset), cycle the power supply of the mp-Highdriver4.

Table 3 Device identification, status and enable

REGISTER	B7	B6	B5	B4	B3	B2	B1	B0
0x00	DEVID3	DEVID2	DEVID1	DEVID0	REV3	REV2	REV1	REV0
0x01	OVRTEMP	X	X	X	X	X	X	EN

X = Don't Care

2.1.11 Output Frequency Register (0x02)

FO[7:6] sets the output frequency range of all output channels and FO[5:0] sets the output frequency within the frequency range; see Table 4. FO[5:0] = 000000 sets the frequency to the minimum value of the frequency range. FO[5:0] = 111111 sets the frequency to the maximum value of the frequency range. Output frequency increases linearly with FO[5:0]; see Table 5.

Table 4 Signal output frequency

REGISTER	B7	B6	B5	B4	B3	B2	B1	B0
0x02	FO7	FO6	FO5	FO4	FO3	FO2	FO1	FO0

Table 5 Signal Frequency Range

FO[7:6]	FREQUENCY RANGE (Hz)
00	50-100
01	100-200
10	200-400
11	400-800

2.1.12 Output Shape Register (0x03)

Table 6 Output shape register

Damping Enable (ENDAMP)	1 = Active damping on LX node enabled. 0 = Active damping on LX node disabled. ENDAMP = 1 actively damps the oscillation on the LX pin and could reduce EMI.
Shape (SHAPE1/SHAPE0)	SHAPE[1:0] sets the desired output waveform; see Table 7 and Table 8.
Slew Rate (SL1/SL0)	SL[1:0] sets the slope of the output; see Table 9.

Table 7 Signal shape configuration

REGISTER	B7	B6	B5	B4	B3	B2	B1	B0
0x03	X	ENDAMP	X	X	SHAPE1	SHAPE0	SL1	SL0

X = Don't Care

Table 8: Signal Output Shape Configuration

SHAPE[1:0]	OUTPUT SHAPE
0X	Full Wave
10	Half Wave
11	Half Wave

X = Don't Care

Table 9: Signal Slope Configuration

SL[1:0]	OUTPUT SLOPE
00	Sine
01	Fast Slope
10	Faster Slope
11	Fastest Slope (Square Wave)

2.1.13 Boost-Converter Frequency Register (0x04)

Table 10 Boost-converter frequency register

Spread Spectrum (SS1/SS0)	SS[1:0] sets the spread-spectrum modulation frequency to a fraction of the boost-converter frequency; see Table 11 and Table 12.
Boost-Converter Switching Frequency (FSW[4:0])	FSW4 sets the switching frequency range of the boost converter and FSW[3:0] sets the switching frequency within the frequency range; see Table 13. The frequency range for FSW4 = 0 is 800 kHz – 1600 kHz. The frequency range for FSW4 = 1 is 400 kHz–800kHz. FSW[3:0] = 0000 sets the frequency to the minimum value of the frequency range. FSW[3:0] = 1111 sets the frequency to the maximum value of the frequency range. Boost-converter switching frequency increases linearly with FSW[3:0].

Table 11: Boost-Converter Configurations

REGISTER	B7	B6	B5	B4	B3	B2	B1	B0
0x04	SS1	SS0	X	FSW4	FSW3	FSW2	FSW1	FSW0

Table 12: Spread-Spectrum Configuration

SS[1:0]	SPREAD-SPECTRUM
00	Disabled
01	1/8
10	1/32
11	1/128

Table 13: Boost-Converter Frequency Range

FSW3	FSW2	FSW1	FSW0	BOOST-CONVERTER SWITCHING FREQUENCY (kHz)	
				FSW4 = 0	FSW4 = 1
0	0	0	0	800	400
0	0	0	1	853	427
0	0	1	0	907	453
0	0	1	1	960	480
0	1	0	0	1013	507
0	1	0	1	1067	533
0	1	1	0	1120	560
0	1	1	1	1173	587
1	0	0	0	1227	613
1	0	0	1	1280	640
1	0	1	0	1333	667
1	0	1	1	1387	693
1	1	0	0	1440	720
1	1	0	1	1493	747
1	1	1	0	1547	773
1	1	1	1	1600	800

2.1.14 Ramping Time and Peak Output Voltage Register (0x06, 0x07, 0x08, 0x09)

Table 14 Ramping time and peak output voltage register

Ramping Time (RT4_ _/RT3_ _/RT2_ _/RT1_ _)	RT_ _[2:0] sets the ramp time of each output; see Table 16.
Output Peak-to-Peak Voltage (VO1_ _/VO2_ _/VO3_ _/VO4_ _)	VO_ _[4:0] controls the peak-to-peak voltage of each output. When VO_ _[4:0] = 00000, the output follows COM. When VO_ _[4:0] = 11111, the output has a 250V peak with respect to COM. The output voltage rises linearly with VO_ _[4:0].

Table 15: Output Configuration

REGISTER	B7	B6	B5	B4	B3	B2	B1	B0
0x06	RT1_2	RT1_1	RT1_1	VO1_4	VO1_3	VO1_2	VO1_1	VO1_0
0x07	RT2_2	RT2_1	RT2_1	VO2_4	VO2_3	VO2_2	VO2_1	VO2_0
0x08	RT3_2	RT3_1	RT3_1	VO3_4	VO3_3	VO3_2	VO3_1	VO3_0
0x09	RT4_2	RT4_1	RT4_1	VO4_4	VO4_3	VO4_2	VO4_1	VO4_0

Table 16: Ramping Time Configuration

RT_ _[2:0]	Ramping Time (ms)
000	<0.1
001	62.5
010	125
011	250
100	500
101	750
110	1000
111	2000

2.1.15 I²C Interface

The Highdriver platform features an I²C-compatible as a slave device, 2-wire serial interface consisting of a serial data line (SDA) and a serial-clock line (SCL). SDA and SCL facilitate communication to the device at clock rates up to 400 kHz. Figure 9 shows the 2-wire interface-timing diagram. The master generates SCL and initiates data transfer on the bus. A master device writes data to the mp-Highdriver4 by transmitting the proper slave address followed by the register address and then the data word. Each transmit sequence is framed by a START (S) or REPEATED START (Sr) condition and a STOP (P) condition. Each word transmitted to the mp-Highdriver4 is 8 bits long and is followed by an acknowledge clock pulse. A master reading data from the mp-Highdriver4 transmits data on SDA in sync with the master-generated SCL pulses. The master acknowledges receipt of each byte of data. Each read sequence is framed by a START or REPEATED START condition, a not acknowledge, and a STOP condition. SDA operates as both an input and an open-drain output. A pull-up resistor, typically greater than 500Ω, is required on SCL if there are multiple masters on the bus, or if the master in a single-master system has an open-drain SCL output. Series resistors in line with SDA and SCL are optional. Series resistors protect the digital inputs of the mp-Highdriver4 from high-voltage spikes on the bus lines, and minimize crosstalk and undershoot of the bus signals.

2.1.16 Bit Transfer

One data bit is transferred during each SCL cycle. The data on SDA must remain stable during the high period of the SCL pulse. Changes in SDA while SCL is high are control signals (see the START and STOP Conditions section). SDA and SCL idle high when the I2C bus is not busy.

2.1.17 START and STOP Conditions

SDA and SCL idle high when the bus is not in use. A master initiates communication by issuing a START condition. A START condition is a high-to-low transition on SDA with SCL high. A STOP condition is a low-to-high transition on SDA while SCL is high (Figure 1). A START condition from the master signals the beginning of a transmission to the mp-Highdriver4. The master terminates transmission and frees the bus by issuing a STOP condition. The bus remains active if a REPEATED START condition is generated instead of a STOP condition.

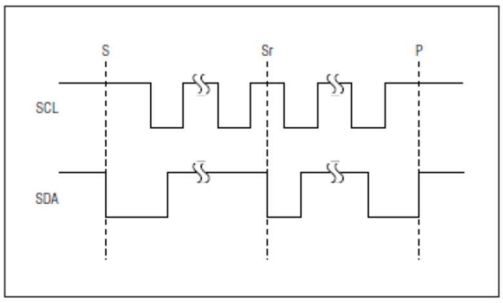


Figure 1: START, STOP and REPEATED START Conditions

2.1.18 Early STOP Conditions

The Highdriver platform recognizes a STOP condition at any point during data transmission, except if the STOP condition occurs in the same high pulse as a START condition. For proper operation, do not send a STOP condition during the same SCL high pulse as the START condition.

2.1.19 Acknowledge

The acknowledge bit (ACK) is a clocked 9.th bit that the mp-Highdriver4 uses to handshake receipt each byte of data when in write mode (see Figure 2). The Highdriver platform pulls down SDA during the entire master generated 9th clock pulse if the previous byte is successfully received. Monitoring ACK allows for detection of unsuccessful data transfers. An unsuccessful data transfer occurs if a receiving device is busy or if a system fault had occurred. In the event of an unsuccessful data transfer, the bus master may retry communication. The master pulls down SDA during the 9.th clock cycle to acknowledge receipt of data when the Highdriver platform is in read mode. An acknowledge is sent by the master after each read byte to allow data transfer to continue. A not-acknowledge is sent when the master reads the final byte of data from the Highdriver platform followed by a STOP condition.

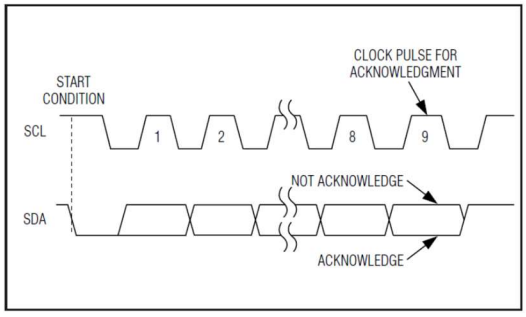


Figure 2: Acknowledge

2.1.20 Write Data Format

A write to the Highdriver platform includes transmission of a START condition, the slave address with the R/W bit set to 0, one byte of data to configure the internal register address pointer, one or more bytes of data, and a STOP condition. Figure 3 illustrates the proper frame format for writing one byte of data to the Highdriver platform. Figure 4 illustrates the frame format for writing n-bytes of data to the Highdriver platform. The slave address with the R/W bit set to 0 indicates that the master intends to write data to the Highdriver platform. The Highdriver platform acknowledges receipt of the address byte during the master-generated 9.th SCL pulse. The second byte transmitted from the master configures the mp-Highdriver4 internal register address pointer. The pointer tells the mp-Highdriver4 where to write the next byte of data. An acknowledge pulse is sent by the Highdriver platform upon receipt of the address pointer

data. The third byte sent to the Highdriver platform contains the data that will be written to the chosen register. An acknowledge pulse from the Highdriver platform signals receipt of the data byte. The address pointer automatically increments to the next register address after each received data byte. This auto increment feature allows a master to write to sequential registers within one continuous frame. Attempting to write to register addresses higher than 0x0B results in repeated writes of 0x0B. Figure 4 illustrates how to write to multiple registers with one frame. The master signals the end of transmission by issuing a STOP condition.

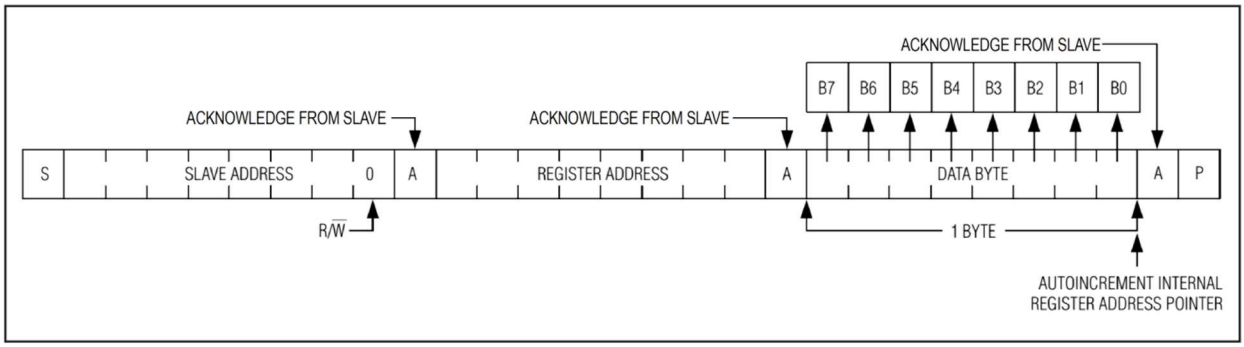


Figure 3: Writing One Byte of Data to the mp-Highdriver4

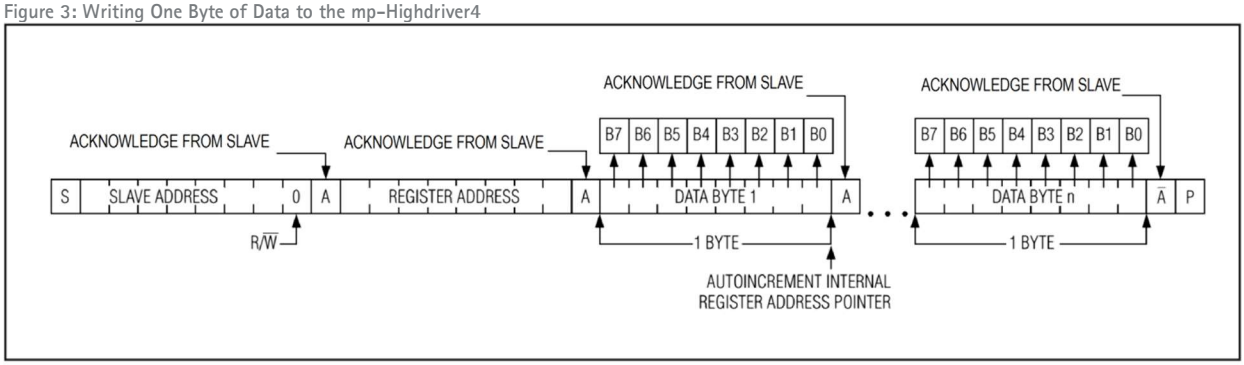


Figure 4: Writing n-Bytes of Data to the mp-Highdriver4

2.1.21 Read Data Format

Send the slave address with the R/W set to 1 to initiate a read operation. The Highdriver platform acknowledges receipt of its slave address by pulling SDA low during the 9th SCL clock pulse. A START command followed by a read command resets the address pointer to register 0x00. The first byte transmitted from the Highdriver platform will be the contents of register 0x00. Transmitted data is valid on the rising edge of the master-generated serial clock (SCL). The address pointer automatically increments after each read data byte. This auto increment feature allows all registers to be read sequentially within one continuous frame. A STOP condition can be issued after any number of read data bytes. If a STOP condition is issued followed by another read operation, the first data byte to be read will be from register 0x00 and subsequent reads will auto increment the address pointer until the next STOP condition. The address pointer can be preset to a specific register before a read command is issued. The master presets the address pointer by first sending the Highdriver platform's slave address with the R/W bit set to 0 followed by the register address. A REPEATED START condition is then sent, followed by the slave address with the R/W set to 1. The Highdriver platform transmits the contents of the specified register. The address pointer automatically increments after transmitting the first byte. Attempting to read from register addresses higher than 0x0B results in repeated reads of 0x0B. The master acknowledges receipt of each read byte during the acknowledge clock pulse. The master must acknowledge all correctly received bytes except the last byte. The final byte must be followed by a not acknowledge from the master and then a STOP condition. Figure 5 illustrates the frame format for reading one byte from the mp-Highdriver4. Figure 6 illustrates the frame format for reading multiple bytes from the Highdriver platform.

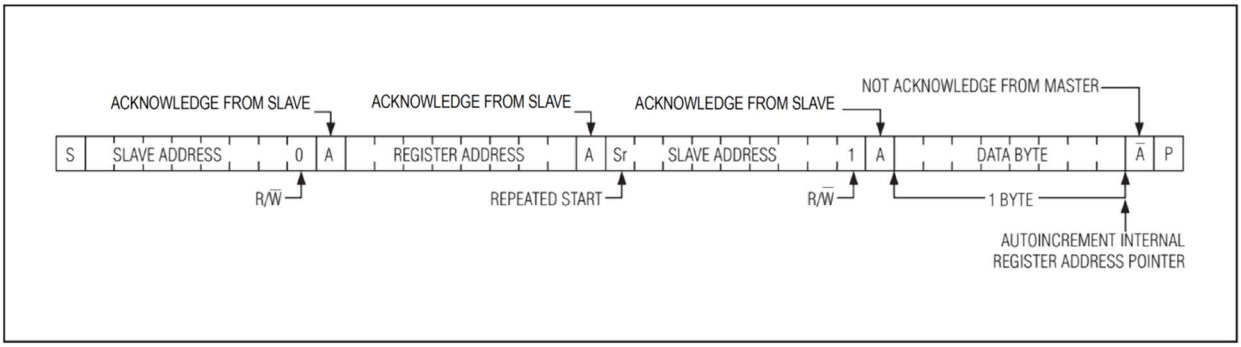


Figure 5: Reading One Indexed Byte of Data from the mp-Highdriver4

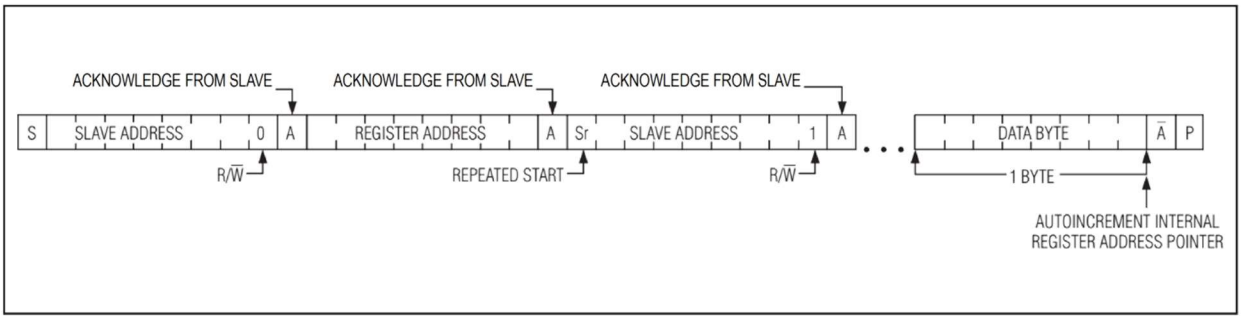
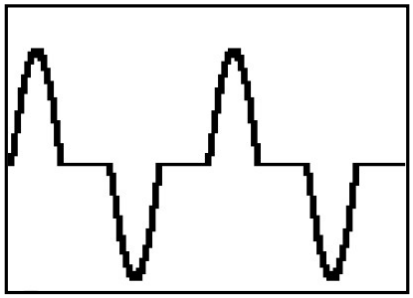


Figure 6: Reading n-Bytes of Indexed Data from the mp-Highdriver4

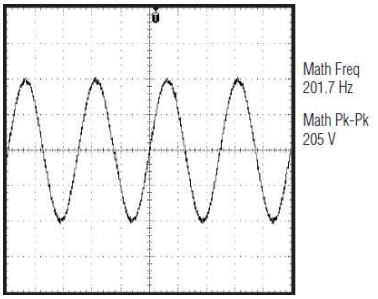
2.1.22 Signal Shapes

Signal Shape with SHAPE[1:0] = 10



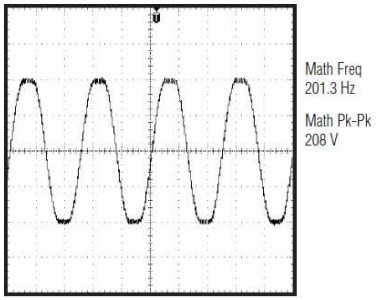
and SL[1:0] = 00

Signal Shape with SL[1:0] = 00



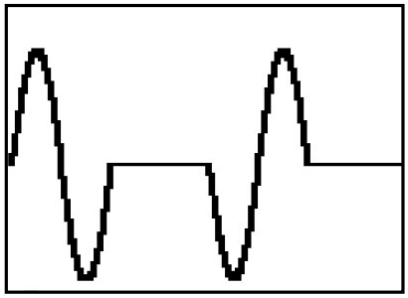
and SHAPE[1:0] = 0X

Signal Shape with SL[1:0] = 01



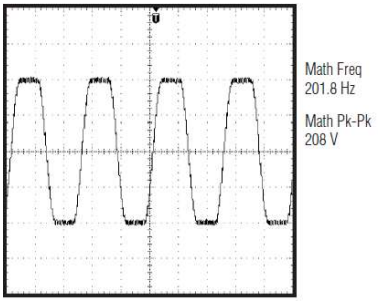
and SHAPE[1:0] = 0X

Signal Shape with SHAPE[1:0] = 11



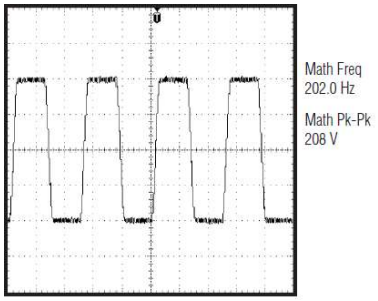
and SL[1:0] = 00

Signal Shape with SL[1:0] = 10



and SHAPE[1:0] = 0X

Signal Shape with SL[1:0] = 11



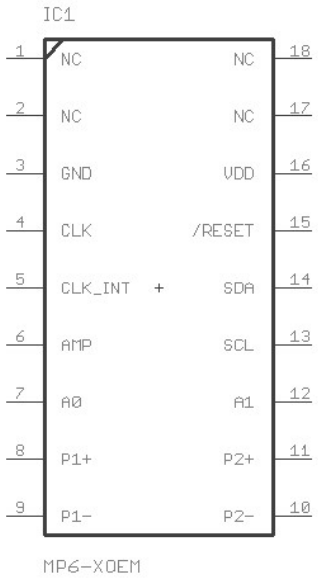
and SHAPE[1:0] = 0X

Figure 7: Signal shapes

2.2 mp-Highdriver Specifics

2.2.1 Pinout Highdriver

Pin	Name	Function
1,2,17,18	NC	This pins should not be connected and left floating
3	GND	Ground
4	CLK	Predefined clock signal. Frequency will be set to the nominal 200 Hz, when this pin is connected to CLK_INT (see image "Schematic 1" in chapter **Fehler! Verweisquelle konnte nicht gefunden werden.**). It is also possible to connect an external clock signal with a **quadruplicated** frequency of the micropumps frequency.
5	CLK_INT	When connected to CLOCK the frequency is set to 200 Hz.
6	AMP	The amplitude can be set with an analogue voltage between 0.5 V to 1.3 V.
7	A0	Address Input 0. Address inputs allow up to four connections on one common bus. Connect A0 to GND or VDD.
8	P1+	Piezo actuator 1, positive electrode (see page **Fehler! Textmarke nicht definiert.**)
9	P1-	Piezo actuator 1, negative electrode (see page **Fehler! Textmarke nicht definiert.**)
10	P2-	Piezo actuator 2, negative electrode (see page **Fehler! Textmarke nicht definiert.**)
11	P2+	Piezo actuator 2, positive electrode (see page **Fehler! Textmarke nicht definiert.**)
12	A1	Address Input 1. Address inputs allow up to four connections on one common bus. Connect A1 to GND or VDD.
13	SCL	Serial-Clock Input. SCL requires an external pullup resistor.
14	SDA	Open-Drain, Serial Data Input/Output. SDA requires an external pullup resistor.
15	/RESET	If the I2C-Bus is to be used, the internal processor has to be disabled by applying GND to this pin. Otherwise connect this pin to VDD for normal operation.
16	VDD	Input Supply voltage



2.2.2 Technical specifications mp-Highdriver

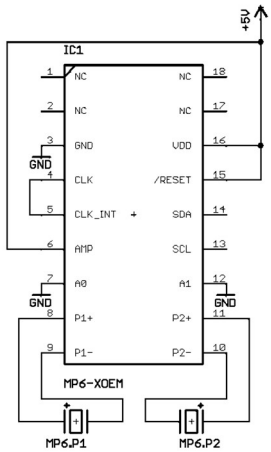
mp-Highdriver controller	**Order code: mp-Highdriver**
The mp-Highdriver controller drives the micropump at adjustable performance in a package similar to an integrated circuit. It enables integration into system electronics or on a PCB.	
Dimensions	10,16 x 25,40 x 2.82 mm 0.4 x 1.0 x 0.11 in
Adjustable parameters	amplitude, frequency, wave form
Amplitude range	10 – 250 Vpp
Frequency range	50 - 800 Hz
Signal form	sine, rectangular, trapezoid
Power supply	2.7 – 5.5 VDC
Pin arrangement	DIL 18; horizontal 2.54 mm, vertical 7.62 mm

2.2.3 Electrical Characteristics mp-Highdriver

Parameter	Symbol	Conditions	Min	Typ.	Max	Unit
Supply voltage	VDD		2.7		5.5	V
Average current consumption	IDD	VDD = 5 V		40		mA
Setting range AMPLITUDE			0.5		1.3	VDC
min. voltage at pump	Vpump	AMPLITUDE = 0.5 VDC		100		Vpp
max. voltage at pump (1)	Vpump	AMPLITUDE = 1.25 VDC	250	260	270	Vpp
Frequency output	F	VDD = 5 V (Default)		200		Hz
Digital Low-Signal					0	V
Digital High-Signal			2			V
Input current AMPLITUDE			1		3	µA
Operating current during Shutdown Mode				1.6		µA

2.2.4 Ex. of circuiting the mp-Highdriver: Operation with fixed settings

The mp-Highdriver can operate the micropumps of the mp6-series without further external components. In this case, frequency and amplitude to the micropump are predefined to 250 Vpp and 200 Hz by the internal circuit.



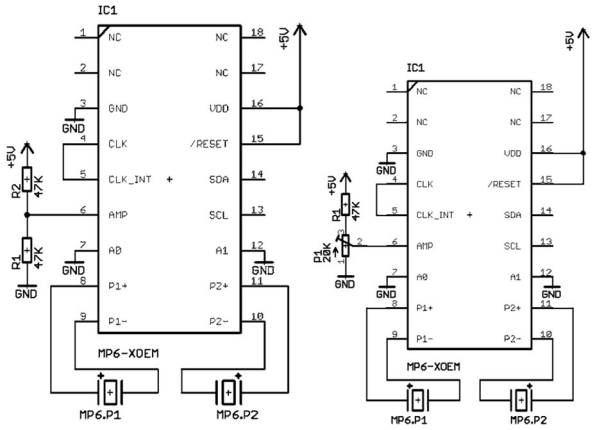
Schematic 1 Predefined amplitude of 250 Vpp and frequency of 200 Hz by internal circuit

2.2.5 Ex. of circuiting the mp-Highdriver: Operation with variable settings & external components

In this example, amplitude to the micropump will be defined by external components. The amplitude can be varied in the range from 0 Vpp to 250 Vpp.
Using a voltage divider, it is possible to set the amplitude with only two additional resistors. See image Schematic 2 below for an example. First picture shows two resistors generating the reference voltage for the AMP-pin. The voltage at the AMP-pin can be calculated with this formula:

$$V_{AMPLITUDE} = V_{DD} \cdot \frac{R1}{R1 + R2}$$

As an alternative an appropriate potentiometer can be used to adjust the reference voltage. On the second picture a potentiometer and an offset resistor are used to generate the reference voltage for the AMP-pin.



Schematic 2 Defining amplitude with external components

The relation of the voltage at the AMP-pin vs. the output voltage is shown in the picture below. It shows a gain of approx. 200:1 between reference voltage and output voltage.

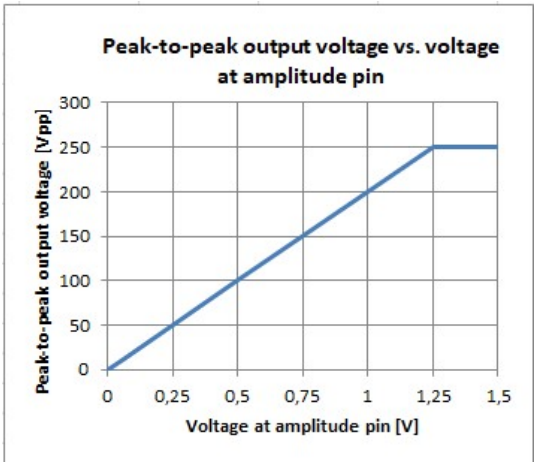


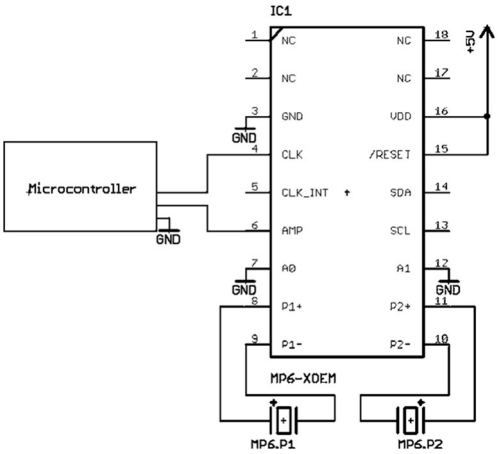
Figure 8 OEM output voltage changed with external components

Unlike the mp6-OEM, the frequency on the mp-Highdriver cannot be selected via external capacitor. This can only be done my supplying a clock signal to the CLK-pin witch is shown in the next chapter.

2.2.6 Ex. of circuiting the mp-Highdriver: Operation with variable settings via microcontroller

Using a microcontroller to operate the micropump, a square wave signal (0V-VDD; 50% duty-cycle) with four times the desired output frequency has to be supplied to the CLK Pin.
The amplitude can be set with an analog voltage between 0 V and 1.25 V. Voltages above 1.25V up to VDD are allowed and keep the output voltage at maximum.

For minimizing the power consumption, the mp-Highdriver can be switched off. To do this, apply a 0V signal to the AMP-pin.

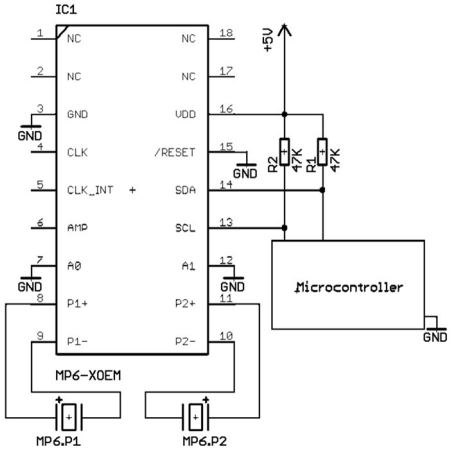


Schematic 3 Controlling settings via microcontroller

2.2.7 Ex. of circuiting the mp-Highdriver: Operation with variable settings via I²C-Interface

The mp-Highdriver can also be operated using the I²C-Interface. Using this interface all the features of the chip can be accessed. The functionality and procedures are identical to the mp6-Highdriver4. A complete interface and protocol description can be found in the I²C section of this document.

To activate the I²C-Interface the onboard processor has to be deactivated by tying the reset pin to Ground. Select a desired slave-address connecting pins A0 and A1 to either VDD or GND and interface the chip through the I²C-pins SDA and SCL. Both need an external pullup resistor (4,7K recommended) tied to VDD. (There are no pullup resistors onboard the mp-Highdriver)



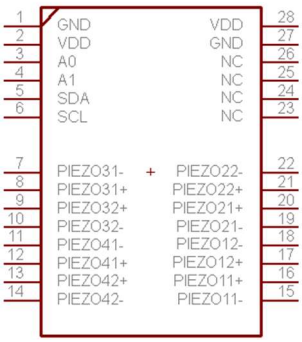
Schematic 4 Controlling settings via I²C

2.3 mp-Highdriver4 Specifics

2.3.1 Pinout mp-Highdriver4

PIN	NAME	FUNCTION
1	GND	Ground
2	VDD	Input Supply Voltage
3	A0	Address Input 0. Address inputs allow up to four connections on one common bus. Connect A0 to GND or VDD.
4	A1	Address Input 1. Address inputs allow up to four connections on one common bus. Connect A1 to GND or VDD.
5	SDA	Open-Drain, Serial Data Input/Output. SDA requires an external pull-up resistor

6	SCL	Serial-Clock Input. SCL requires an external pull-up resistor.
7	COM[1] PIEZO31-	High-Voltage common Output. Connect to negative input of piezo 1 of pump 3
8	PIEZO31+	High-Voltage Output 31+. Connect to positive input of piezo 1 of pump 3
9	PIEZO32+	High-Voltage Output 32+. Connect to positive input of piezo 2 of pump 3
10	PIEZO32-	High-Voltage Output 32-. Connect to negative input of piezo 2 of pump 3
11	COM[1] PIEZO41-	High-Voltage common Output. Connect to negative input of piezo 1 of pump 4
12	PIEZO41+	High-Voltage Output 41+. Connect to positive input of piezo 1 of pump 4
13	PIEZO42+	High-Voltage Output 42+. Connect to positive input of piezo 2 of pump 4
14	PIEZO42-	High-Voltage Output 42-. Connect to negative input of piezo 2 of pump 4
15	COM[1] PIEZO11-	High-Voltage common Output. Connect to negative input of piezo 1 of pump 1
16	PIEZO11+	High-Voltage Output 11+. Connect to positive input of piezo 1 of pump 1
17	PIEZO12+	High-Voltage Output 12+. Connect to positive input of piezo 2 of pump 1
18	PIEZO12-	High-Voltage Output 12-. Connect to negative input of piezo 2 of pump 1
19	COM[1] PIEZO21-	High-Voltage common Output. Connect to negative input of piezo 1 of pump 2
20	PIEZO21+	High-Voltage Output 21+. Connect to positive input of piezo 1 of pump 2
21	PIEZO22+	High-Voltage Output 22+. Connect to positive input of piezo 2 of pump 2
22	PIEZO22-	High-Voltage Output 22-. Connect to negative input of piezo 2 of pump 2
23	NC	Not connected (do not connect)
24	NC	Not connected (do not connect)
25	NC	Not connected (do not connect)
26	NC	Not connected (do not connect)
27	GND	Ground
28	VDD	Input Supply Voltage



2.3.2 Electrical Characteristics mp-Highdriver4

Table 17 Absolute Maximum Ratings

VDD	-0.3V to +6.0V	A0, A1	-0.3V to +6.0V
PIEZOXX, COM	-0.3V to +160V	SCL, SDA	-0.3V to (VDD + 0.3V)

Table 18 Electrical Characteristics

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Input Voltage	V_{DD}		2.7		5.5	V
Input Supply Current	I_{DD}	All channels on, 300V_{PP}, f = 100Hz, sinewave output shape		75		mA
Shutdown Supply Current	I_{SHDN}	A0, A1 = 0V or V_{DD}; SCL = SDA = GND or V_{DD}; not toggling		25	100	nA
Undervoltage Lockout	VUV	V_{DD} rising	1.6	2.0	2.5	V
PIEZO OUTPUTS						
Peak-to-Peak Output Voltage	V_{PP}	PIEZO_ _+ - PIEZO_ _-; VO_ _[4:0]=01000; V_{DD}=5.0V		75		V
		PIEZO_ _+ - PIEZO_ _-; VO_ _[4:0]=10000; V_{DD}=5.0V		150		
		PIEZO_ _+ - PIEZO_ _-; VO_ _[4:0]=11111; V_{DD}=5.0V		250		
Switching Frequency	f_{LR}	FO[7:0]=10000000; V_{DD}=5.0V	194	200	206	Hz
	f_{HR}	FO[7:0]=10111111; V_{DD}=5.0V	388	400	412	
BOOST CONVERTER						
Peak Output Voltage	V_{CS}	VO_ _[4:0] = 01000; V_{DD} = 5.0V		40		V
		VO_ _[4:0] = 10000; V_{DD} = 5.0V		75		
		VO_ _[4:0] = 11111; V_{DD} = 5.0V		150		
Tapped-Inductor Center Switching Frequency	f_{SW}	FSW[4:0] = 10000	400			kHz
		FSW[4:0] = 11111	800			
		FSW[4:0] = 00000 (default)	800			
		FSW[4:0] = 01111	1600			
Tapped-Inductor Switching Frequency Spreading Factor	S_F	SS[1:0] = 01, 10, or 11	8			%
I²C INTERFACE LOGIC (SDA, SCL, A1, AND A0)						
Input Logic-Low Voltage	V_{IL}				0.5	V
Input Logic-High Voltage	V_{IH}		1.5			V
Input Hysteresis	I_{HYS}			130		mV
Input Leakage Current	I_{LKG}		-1		+1	µA
Output Low Voltage	V_{OL}	ISINK = 3mA			0.4	V
Input/Output Capacitance	$C_{I/O}$		10			pF
Serial-Clock Frequency	f_{SCL}				400	kHz
Clock Low Period	t_{LOW}		1.3			µs
Clock High Period	t_{HIGH}		0.6			µs
Bus Free Time	t_{BUF}		1.3			µs
START Setup Time	$t_{SU,STA}$		0.6			µs
START Hold Time	$t_{HD,STA}$		0.6			µs
STOP Setup Time	$t_{SU,STO}$		0.6			µs
Data In Setup Time	$t_{SU,DAT}$		100			ns
Data In Hold Time	$t_{HD,DAT}$		0		900	ns

Receive SCL/SDA Minimum Rise Time	t_R			$20 + 0.1C_B$		ns
Receive SCL/SDA Maximum Rise Time	t_R			300		ns
Receive SCL/SDA Minimum Fall Time	t_F			$20 + 0.1C_B$		ns
Receive SCL/SDA Maximum Fall Time	t_F			300		Ns
Transmit SDA Fall Time	t_F	C_B = 400pF	$20 + 0.1C_B$		300	ns
SCL/SDA Noise Suppression Time	t_I			50		ns
ESD PROTECTION						
PIEZO_ _+ to PIEZO_ _-		Human Body Model	±15			kV
		IEC 61000-4-2 Contact Discharge	±6			
		IEC 61000-4-2 Air Gap Discharge	±8			
THERMAL PROTECTION						
Thermal Shutdown	T_{SHDN}		160			°C
Thermal Shutdown Hysteresis	T_{HYST}		12			°C

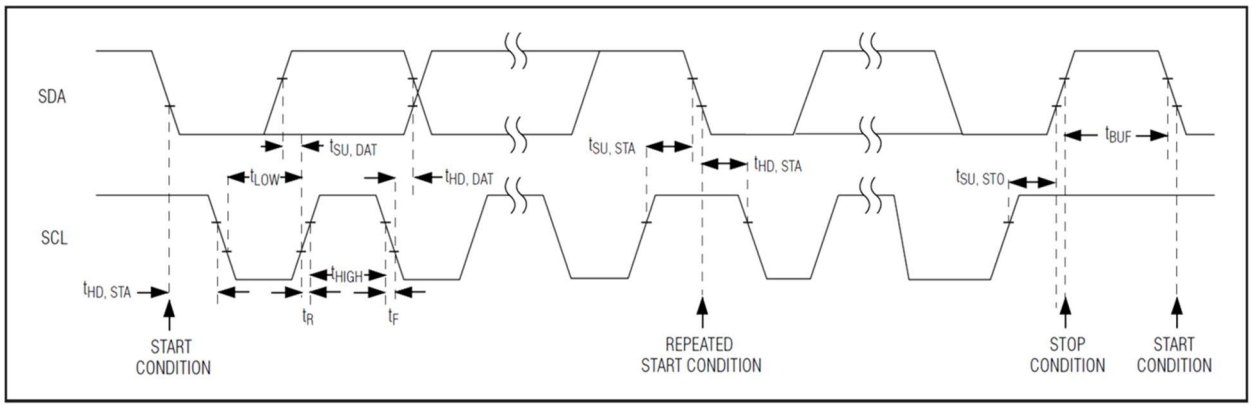
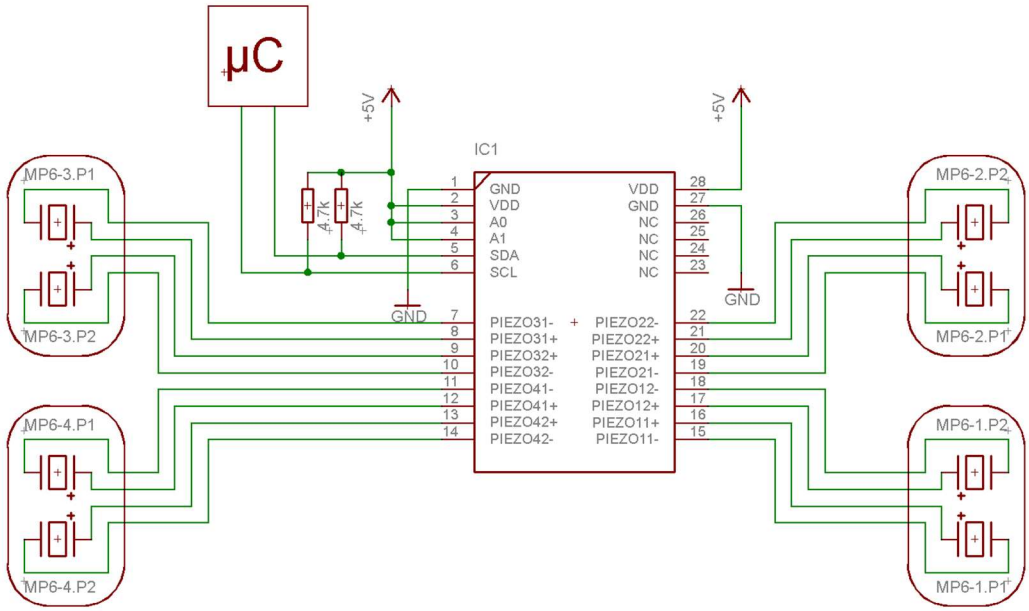


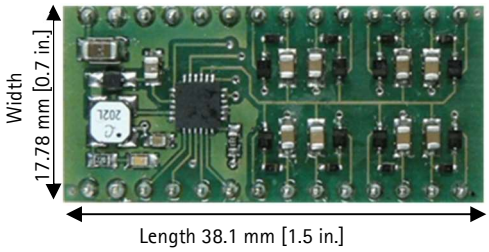
Figure 9: I2C Timing Specifications

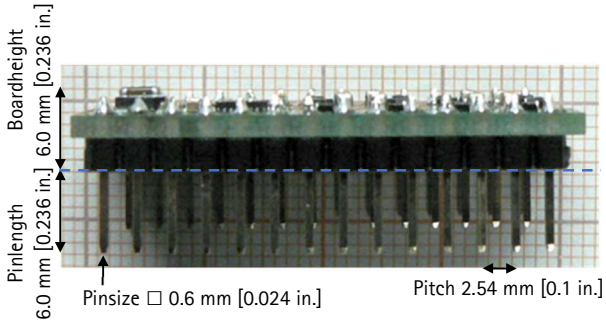
2.3.3 Examples of circuiting the mp-Highdriver4



Schematic 5 Typical Operating circuit for the mp-Highdriver4

2.3.4 Packaging Dimensions





All pins have a square cross section of 0.6 mm (0.024 in.)

2.3.5 Example code for mp-Highdriver

```
#define I2C_HIGHDRIVER_ADRESS  (0x78)   //Default adress for mp-Highdriver
#define I2C_DEVICEID       0x00
#define I2C_POWERMODE      0x01
#define I2C_FREQUENCY      0x02
#define I2C_SHAPE          0x03
#define I2C_BOOST          0x04
#define I2C_PVOLTAGE       0x06
#define I2C_P1VOLTAGE      0x06
#define I2C_P2VOLTAGE      0x07
#define I2C_P3VOLTAGE      0x08
#define I2C_P4VOLTAGE      0x09
#define I2C_UPDATEVOLTAGE  0x0A
#define I2C_AUDIO          0x05

extern boolean bPumpState[4];
extern uint8_t nPumpVoltageByte[4];
extern uint8_t nFrequencyByte;

void Highdriver_init(void) {
  Wire.beginTransmission(I2C_HIGHDRIVER_ADRESS);
```

```
  Wire.write(I2C_POWERMODE);   // Start Register 0x01
  Wire.write(0x01);                    // Register 0x01 = 0x01 (enable)
  Wire.write(nFrequencyByte);     // Register 0x02 = 0x40 (100Hz)
  Wire.write(0x00);      // Register 0x03 = 0x00 (sine wave)
  Wire.write(0x00);      // Register 0x04 = 0x00 (800KHz)
  Wire.write(0x00);      // Register 0x05 = 0x00 (audio off)
  Wire.write(0x00);      // Register 0x06 = Amplitude1
  Wire.write(0x00);      // Register 0x07 = Amplitude2
  Wire.write(0x00);      // Register 0x08 = Amplitude3
  Wire.write(0x00);      // Register 0x09 = Amplitude4
  Wire.write(0x01);       // Register 0x0A = 0x01 (update)
  Wire.endTransmission();
  bPumpState[0] = false;
  nPumpVoltageByte[0] = 0x1F;
}

void Highdriver_setvoltage(uint8_t _voltage) {     // Amplituden uebernehmen
  float temp = _voltage; temp*=31.0f; temp/=250.0f;   //150Vpp = 0xFF
  nPumpVoltageByte[0] = constrain(temp,0,31);
  Wire.beginTransmission(I2C_HIGHDRIVER_ADRESS);
  Wire.write(I2C_PVOLTAGE);
  Wire.write(0);
  Wire.write(0);
  Wire.write(0);
  Wire.write((bPumpState[0] ? nPumpVoltageByte[0] : 0));
  Wire.write(0x01);           // update register values
  Wire.endTransmission();
}

void Highdriver_setvoltage(void) {      // Amplituden uebernehmen
  Wire.beginTransmission(I2C_HIGHDRIVER_ADRESS);
  Wire.write(I2C_PVOLTAGE);
  Wire.write(0);
  Wire.write(0);
  Wire.write(0);
  Wire.write((bPumpState[0] ? nPumpVoltageByte[0] : 0));
  Wire.write(0x01);           // Register Werte uebernehmen
  Wire.endTransmission();
}

void Highdriver_setfrequency(uint16_t _frequency) {
  if (_frequency>=800) {
    nFrequencyByte=0xFF;
  } else if (_frequency>=400) {         // 400-800 Hz
    _frequency-=400;
    _frequency*=64;
    _frequency/=400;
    nFrequencyByte = _frequency|0xC0;
  } else if (_frequency>=200) {         // 200-400 Hz
    _frequency-=200;
    _frequency*=64;
    _frequency/=200;
    nFrequencyByte = _frequency|0x80;
  } else if (_frequency>=100) {         // 100-200 Hz
    _frequency-=100;
    _frequency*=64;
    _frequency/=100;
    nFrequencyByte = _frequency|0x40;
  } else if (_frequency>=50) {          // 50-100 Hz
    _frequency-=50;
    _frequency*=64;
    _frequency/=50;
    nFrequencyByte = _frequency|0x00;
```

```
  } else {              // outside of valid area
nFrequencyByte=0x00;
  }
  Wire.beginTransmission(I2C_HIGHDRIVER_ADRESS);
  Wire.write(I2C_FREQUENCY);
  Wire.write(nFrequencyByte);
  Wire.endTransmission();
}
```

2.3.6 Example code for the mp-Highdriver4

```
#define I2C_HIGHDRIVER_ADRESS  (0x78)   //Default adress for mp-Highdriver4 11110XX

#define I2C_DEVICEID       0x00
#define I2C_POWERMODE      0x01
#define I2C_FREQUENCY      0x02
#define I2C_SHAPE          0x03
#define I2C_BOOST          0x04
#define I2C_PVOLTAGE       0x06
#define I2C_P1VOLTAGE      0x06
#define I2C_P2VOLTAGE      0x07
#define I2C_P3VOLTAGE      0x08
#define I2C_P4VOLTAGE      0x09
#define I2C_UPDATEVOLTAGE  0x0A
#define I2C_AUDIO          0x05

extern boolean bPumpState[4];
extern uint8_t nPumpVoltageByte[4];
extern uint8_t nFrequencyByte;

void Highdriver4_init(void) {              // Initialize mp-Highdriver
  Wire.beginTransmission(I2C_HIGHDRIVER_ADRESS);
  Wire.write(I2C_POWERMODE);         // Start Register 0x01
  Wire.write(0x01);                  // Register 0x01 = 0x01 (enable)
  Wire.write(nFrequencyByte);    // Register 0x02 = 0x40 (100Hz)
  Wire.write(0x00);       // Register 0x03 = 0x00 (sine wave)
  Wire.write(0x00);       // Register 0x04 = 0x00 (800KHz)
  Wire.write(0x00);       // Register 0x05 = 0x00 (audio off)
  Wire.write(0);           // Register 0x06 = Amplitude1
 Wire.write(0);           // Register 0x07 = Amplitude2
  Wire.write(0);           // Register 0x08 = Amplitude3
  Wire.write(0);           // Register 0x09 = Amplitude4
  Wire.write(0x01);       // Register 0x0A = 0x01 (update)
  Wire.endTransmission();
  bPumpState[0] = false;
  bPumpState[1] = false;
  bPumpState[2] = false;
  bPumpState[3] = false;
  nPumpVoltageByte[0] = 0x1F;
  nPumpVoltageByte[1] = 0x1F;
  nPumpVoltageByte[2] = 0x1F;
  nPumpVoltageByte[3] = 0x1F;
}

void Highdriver4_setvoltage(uint8_t _pump, uint8_t _voltage) {   // Amplituden uebernehmen
  float temp = _voltage; temp*=31.0f; temp/=250.0f;   //250Vpp = 0x1F
  if (_pump>=1 &t&t _pump<=4) nPumpVoltageByte[_pump-1]=constrain(temp,0,31);
  Wire.beginTransmission(I2C_HIGHDRIVER_ADRESS);
  Wire.write(I2C_PVOLTAGE);
  Wire.write((bPumpState[0] ? nPumpVoltageByte[0] : 0));
  Wire.write((bPumpState[1] ? nPumpVoltageByte[1] : 0));
  Wire.write((bPumpState[2] ? nPumpVoltageByte[2] : 0));
  Wire.write((bPumpState[3] ? nPumpVoltageByte[3] : 0));
```

```
  Wire.write(0x01);                                  // Register Werte uebernehmen
  Wire.endTransmission();
}

void Highdriver4_setvoltage(void) {                          // Amplituden uebernehmen
  Wire.beginTransmission(I2C_HIGHDRIVER_ADRESS);
  Wire.write(I2C_PVOLTAGE);
  Wire.write((bPumpState[0] ? nPumpVoltageByte[0] : 0));
  Wire.write((bPumpState[1] ? nPumpVoltageByte[1] : 0));
  Wire.write((bPumpState[2] ? nPumpVoltageByte[2] : 0));
  Wire.write((bPumpState[3] ? nPumpVoltageByte[3] : 0));
  Wire.write(0x01);              // update register values
  Wire.endTransmission();
}

void Highdriver4_setfrequency(uint16_t _frequency) {
  if (_frequency>=800) {
    nFrequencyByte=0xFF;
  } else if (_frequency>=400) {        // Bereich 400-800
    _frequency-=400;
    _frequency*=64;
    _frequency/=400;
    nFrequencyByte = _frequency|0xC0;
  } else if (_frequency>=200) {        // Bereich 200-400
    _frequency-=200;
    _frequency*=64;
    _frequency/=200;
    nFrequencyByte = _frequency|0x80;
  } else if (_frequency>=100) {        // Bereich 100-200
    _frequency-=100;
    _frequency*=64;
    _frequency/=100;
    nFrequencyByte = _frequency|0x40;
  } else if (_frequency>=50) {         // Bereich 50-100
    _frequency-=50;
    _frequency*=64;
    _frequency/=50;
    nFrequencyByte = _frequency|0x00;
  } else {                             // outside of valid area
    nFrequencyByte=0x00;
  }
  Wire.beginTransmission(I2C_HIGHDRIVER_ADRESS);
  Wire.write(I2C_FREQUENCY);
  Wire.write(nFrequencyByte);
  Wire.endTransmission();
}
```

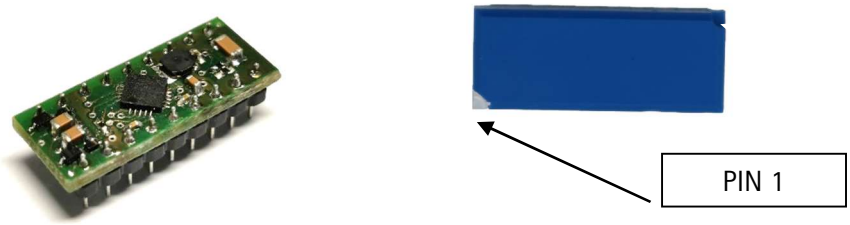
2.4 Lowdriver platform: mp-Lowdriver

The mp-Lowdriver is a small, easy to use driving circuit developed for the micropumps of the mp6-series especially for low-flowrate applications as it has a high amplitude resolution and is thus ideally suited for controlled loop flow regulation systems. It generates amplitudes up to 150 Vpp from a 3-5 V DC supply. The mp-Lowdriver comes with a high range of frequencies, but take into account that we only guarantee proper working conditions of the mp6 micropump inbetween 0 – 800 Hz.

Its low power consumption makes it ideal for battery powered handheld devices or even solar powered devices.

The module can be integrated into a PCB design as a 18 pin DIL package.
The I²C interface allows the user to adapt frequency, amplitude and signal-shape to its application by the use of any simple microcontroller capable of I²C communication.

In order to locate Pin 1, please refer to the following figure. The pin is marked with a colored spot or triangular marking on the corner of the PCB.



2.4.1 Technical specifications mp-Lowdriver

Tabelle 1 mp6 micropump series has only been tried and tested up to 800 Hz. The use of frequencies above 800 Hz is not advised

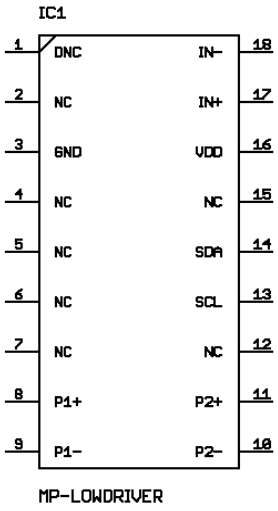
mp-Lowdriver controller	Order code: mp-Lowdriver
Dimensions	10.16 x 25.4 x 2.64 mm 0.4 x 1.0 x 0.10 in
Adjustable parameters	amplitude, frequency, signal shape
Amplitude range	0 – 150 Vpp in 255 steps
Frequency range	8 – 2000 Hz[1] in 7.8125 Hz steps
Signal form	sine, custom
Power supply	3 – 5.5 V DC (6.0V absolute maximum rating)
Pin arrangement	DIL 18; horizontal 2.54 mm, vertical 7.62 mm

2.4.2 Electrical Characteristics

Parameter	Symbol	Conditions	Min	Typ.	Max	Unit
Supply voltage	VDD		3.0		5.5	V
Average current consumption	IDD	VDD = 5 V		40		mA
Voltage at pump	Vpump	VDD = 5 V	0		150	Vpp
Frequency range	Fpump	VDD = 5 V	7.8125		1992.1875	Hz
Digital Low-Signal					0.5	V
Digital High-Signal			1.4			V
Digital input current					1	µA
Shutdown current		VDD = 3.6V STANDBY=1		10		µA

2.4.3 Pin description

Pin	Name	Function
1, 2, 4, 5, 6, 7, 12, 15	NC	These pins should not be connected and left floating
3	GND	Ground
8	P1+	Piezo actuator 1, positive electrode (see schematic 1)
9	P1-	Piezo actuator 1, negative electrode (see schematic 1)
10	P2-	Piezo actuator 2, negative electrode (see schematic 1)
11	P2+	Piezo actuator 2, positive electrode (see schematic 1)
13	SCL	Serial-Clock Input. SCL requires an external pullup resistor.
14	SDA	Open-Drain, Serial Data Input/Output. SDA requires an external pullup resistor.
16	VDD	Input Supply voltage
17	IN+	Analog input voltage positive (WIP[1])
18	IN-	Analog input voltage negative (WIP[1])



[1]WIP: work in progress. Analog inputs are not implemented yet. There is no documentation currently available for the use of this feature.

2.4.4 Register Map

Table 19 mp-Lowdriver register map

REG NO.	DEFAULT	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
0x00	0x02	Reserved					ILLEGAL_ADDR	FIFO_EMPTY	FIFO_FULL
0x01	0x38	Reserved					INPUT_MUX	GAIN[1:0]	
0x02	0x40	DEV_RST	STANDBY	Reserved		TIMEOUT[1:0]		EN_OVERRIDE	GO
0x03	0x00	WAVEFORM0[7:0]							
0x04	0x00	WAVEFORM1[7:0]							
0x05	0x00	WAVEFORM2[7:0]							
0x06	0x00	WAVEFORM3[7:0]							
0x07	0x00	WAVEFORM4[7:0]							
ox08	0x00	WAVEFORM5[7:0]							
0x09	0x00	WAVEFORM6[7:0]							
0x0A	0x00	WAVEFORM7[7:0]							
0x0B	0x00	FIFO[7:0]							
0xFF	0x00	PAGE[7:0]							

Table 20 mp-Lowdriver register 0x00

Bit	Field	Type	Default	Value	Description
7-3	Reserved				
2	ILLEGAL_ADDR	RO	0		Indicates that the waveform generator attempted to perform an illegal operation. This usually means that the user entered improper header information in the waveform memory.
				0	Normal operation
				1	Illegal address attempted
1	FIFO_EMPTY	RO	1		Indicates that the internal 100-byte FIFO is empty.
				0	FIFO is not empty
				1	FIFO is empty

0	FIFO_FULL	RO	0		Indicates that the internal 100-byte FIFO is full and cannot accept data until another byte has played through the internal DAC.
				0	FIFO not full
				1	FIFO is full

Table 21 mp-Lowdriver register 0x01

Bit	Field	Type	Default	Value	Description
7-3	Reserved				
2	INPUT_MUX[0]	RW	0		Selects the source to be played.
				0	Digital input source
				1	Analog input source
1-0	GAIN[1:0]	RW	0		Selects the gain for the amplifier.
				0	25 V (Digital) - 28.8 dB (Analog)
				1	50 V (Digital) - 34.8 dB (Analog)
				2	75 V (Digital) - 38.4 dB (Analog)
				3	100 V (Digital) - 40.7 dB (Analog)

Table 22 mp-Lowdriver register 0x02

Bit	Field	Type	Default	Value	Description
7	DEV_RST	RW	0		When asserted, the device will immediately stop any transaction in process, reset all of its internal register to their default values, and enters standby mode.
				0	Normal Operation
				1	Reset device
6	STANDBY	RW	1		Low-power standby
				0	Device is active and ready to receive a signal
				1	Device is in low-power standby mode
5-4	Reserved				
3-2	TIMEOUT[1:0]	RW	0		Time period when the FIFO runs empty and the device goes into idle mode, powering down the boost converter and amplifier.
				0	5 ms
				1	10 ms
				2	15 ms
				3	20 ms
1	EN_OVERRIDE	RW	0		Override bit for the boost converter and amplifier enables.
				0	Boost converter and amplifier enables are controlled by device logic
				1	Boost converter and amplifier are enabled indefinitely
0	GO	RW	0		Starts waveform playback, as indicated by the sequence registers 0x03 through 0x0A. This bit remains high during the execution of waveform playback, and self-clears upon completion of playback. The user may optionally clear this bit to cancel waveform playback.
				0	No waveform playing
				1	Play waveform

Table 23 mp-Lowdriver register 0x03 – 0x0A

Bit	Field	Type	Default	Description
7-0	WAVEFORM[7:0]	RW	0	When the GO bit is asserted, the waveform processing engine will go to register address 0x03 and play the waveform ID that is indicated there. After completion of that waveform, the engine proceeds to register address 0x04 to play that waveform ID. If the ID value is zero, the playback process terminates. Otherwise, this process repeats until it finds a waveform ID of zero, or all 8 waveforms are played.

Table 24 mp-Lowdriver register 0x0B

Bit	Field	Type	Default	Description
7-0	FIFO[7:0]	RW	0	Entry point for FIFO data. The user repeatedly writes this register with continuous waveform data.

Table 25 mp-Lowdriver register 0xFF

Bit	Field	Type	Default	Description
7-0	PAGE[7:0]	RW	0	Page register for memory interface. Write this register with the memory page to be accessed.

Because the device addresses are only 8-bits, a special exception exists to distinguish whether the user is trying to write the page register on the control page at address 0xFF or the memory page at location 0x1FF. In order to access the page register, the programmer must use a Single-Byte I2C protocol to perform a single-byte write to memory location 0xFF.

2.4.5 Waveform Synthesis

The mp-Lowdriver possess a waveform synthesizer for sinusoid signal. This allows the microcontroller to drive a pump with minimum amount of workload. A package of four data bytes representing amplitude, frequency, cycles and envelope need to be stored inside the mp-Lowdriver.

Amplitude

The amplitude byte refers to the magnitude of the synthesized sinusoid. 0xFF produces a full-scale sinusoid, 0x80 produces a half-scale sinusoid, and 0x00 does not produce any signal. To calculate the absolute peak voltage, use the following equation, where amplitude is a single-byte integer:
Peak voltage = amplitude / 255 x full-scale peak voltage

Frequency

The frequency byte adjusts the frequency of the synthesized sinusoid. The minimum frequency is 7.8125 Hz. A value of zero is not allowed. The sinusoidal frequency is determined with the following equation, where frequency is a single-byte integer:
Sinusoid frequency (Hz) = 7.8125 x frequency

Cycles

The number of sinusoidal cycles to be played by the synthesizer. A convenient way to specify the duration of a coherent sinusoid is by inputting the number of cycles. This method ensures that the waveform chunk will always begin and end at zero amplitude, thus avoiding discontinuities. The actual duration in time given by this value may be calculated through the following equation, where # of cycles and frequency are both single-byte integers.
Duration (ms) = 1000 x # of cycles / (7.8125 x frequency)

Envelope

The envelope byte is divided into two nibbles. The upper nibble, bits [7:4], sets the ramp-up rate at the beginning of the synthesized sinusoid, and the lower nibble, bits [3:0], sets the ramp-down rate at the end of the synthesized sinusoid. The user must note that the ramp-up time is included in the duration parameter of the waveform, and the

ramp-down time is appended to the duration parameter of the waveform. As such, if a ramp-up time is used, the ramp-up time must be less than the duration time as programmed in byte 3. ramp to full-scale amplitude (amplitude = 0xFF). Ramps to a fraction of full-scale have the same Also note that the Total Ramp Time is for a fraction of the Total Ramp Time.

Table 26 Envelope ramp-up times

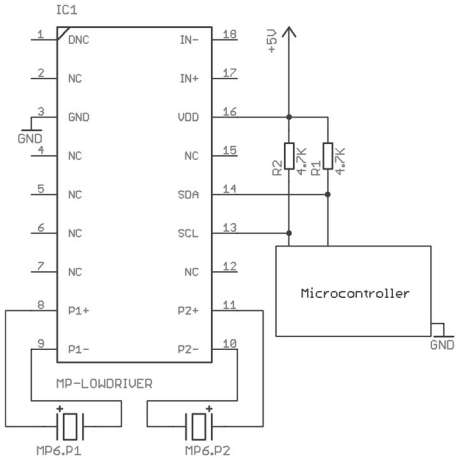
Nibble Value	Total Ramp Time
0	No Envelope
1	32 ms
2	64 ms
3	96 ms
4	128 ms
5	160 ms
6	192 ms
7	224 ms
8	256 ms
9	512 ms
10	768 ms
11	1024 ms
12	1280 ms
13	1536 ms
14	1792 ms
15	2048 ms

The Example code shows how to initialize the mp-Lowdriver as well as a way to change the amplitude and frequency.

2.4.6 Operation with variable settings via I²C-Interface

The mp-Lowdriver can only be operated using the I²C-Interface. Using this interface all the features of the chip can be accessed. A complete interface and protocol description will be released on our website in the future. At the moment the mp-Multiboard source-code is the best source for initializing and changing parameters of the mp-Lowdriver using the I²C-Interface. The source-code is available for download on our website. An excerpt is included in the last chapter of this document.

Use external pullup resistors for the SDA and SCL signals to set the logic-high level for the bus. Pullup resistors with values between 660Ω and 4.7 kΩ are recommended. Do not allow the SDA and SCL voltages to exceed the mp-Lowdriver supply voltage VDD. The mp-Lowdriver device operates as an I2C-slave with 1.8-V logic thresholds, but can operate up to the VDD voltage.



Schematic 6 Controlling the mp-Lowdriver via I²C

2.4.7 Slave address

The slave address for the device is 0x59 (7-bit), or 1011001 in binary, which is equivalent to 0xB2 (8-bit) for writing and 0xB3 (8-bit) for reading.

2.4.8 Example code for mp-Lowdriver

```
#define I2C_LOWDRIVER_ADRESS (0x59)   //Defaultadress for Lowdriver 1011001

extern boolean bPumpState[4];
extern uint8_t nPumpVoltageByte[4];
extern uint8_t nFrequencyByte;

void selectControlRegisters() {
Wire.beginTransmission(I2C_LOWDRIVER_ADRESS);
Wire.write(0xFF);
Wire.write(0x00);
Wire.endTransmission();
}
void selectMemoryRegisters() {
Wire.beginTransmission(I2C_LOWDRIVER_ADRESS);
Wire.write(0xFF);
Wire.write(0x01);
Wire.endTransmission();
}

void Lowdriver_init() {
selectControlRegisters();
Wire.beginTransmission(I2C_LOWDRIVER_ADRESS);
Wire.write(0x01); //Select (Control) Register 0x01
Wire.write(0x02); //Set Gain 0-3 (0x00-0x03 25v-100v)
Wire.write(0x00); //Take device out of standby mode
Wire.write(0x01); //Set sequencer to play WaveForm ID #1
Wire.write(0x00); //End of sequence
Wire.endTransmission();
selectMemoryRegisters();
Wire.beginTransmission(I2C_LOWDRIVER_ADRESS);
Wire.write(0x00); //Select Register 0x00
Wire.write(0x05); //Header size -1
Wire.write(0x80); //Start address upper byte (page), also indicates Waveform Synthesis Mode
Wire.write(0x06); //Start address lower byte (in page address)
Wire.write(0x00); //Stop address upper byte
Wire.write(0x09); //Stop address Lower byte
Wire.write(0x00); //Repeat count, 0 = infinite loop
Wire.write((bPumpState[2] ? nPumpVoltageByte[2] : 0)); //Amplitude
Wire.write(0x0C); //Frequency. (100Hz)
Wire.write(100);  //cycles
Wire.write(0x00); //envelope
Wire.endTransmission();
delay(10);
```

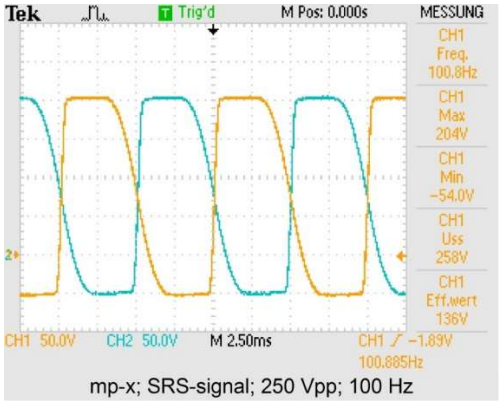
```
selectControlRegisters();
 Wire.beginTransmission(I2C_LOWDRIVER_ADRESS);
 Wire.write(0x02); //Set page register to control space
 Wire.write(0x01); //Set GO bit (execute WaveForm sequence)
 Wire.endTransmission();
}

void Lowdriver_setvoltage(uint8_t _voltage) {
nPumpVoltageByte[2]=_voltage;
Wire.beginTransmission(I2C_LOWDRIVER_ADRESS);
Wire.write(0x02); //Stop Waveform playback
Wire.write(0x00);
Wire.endTransmission();
selectMemoryRegisters();
Wire.beginTransmission(I2C_LOWDRIVER_ADRESS);
Wire.write(0x06); //Set page register to control space
Wire.write((bPumpState[2] ? nPumpVoltageByte[2] : 0));    //0-255
Wire.endTransmission();
delay(10);
selectControlRegisters();
Wire.beginTransmission(I2C_LOWDRIVER_ADRESS);
Wire.write(0x02); //Start Waveform playback
Wire.write(0x01);
Wire.endTransmission();
}

void Lowdriver_setfrequency(uint16_t _frequency) {
float temp = _frequency; temp/=7.8125;
nFrequencyByte = temp;
if (nFrequencyByte==0) nFrequencyByte=1;
Wire.beginTransmission(I2C_LOWDRIVER_ADRESS);
Wire.write(0x02); //Stop Waveform playback
Wire.write(0x00);
Wire.endTransmission();
selectMemoryRegisters();
Wire.beginTransmission(I2C_LOWDRIVER_ADRESS);
Wire.write(0x07); //Set page register to control space
Wire.write(nFrequencyByte);    //0-255
Wire.endTransmission();
delay(10);
selectControlRegisters();
Wire.beginTransmission(I2C_LOWDRIVER_ADRESS);
Wire.write(0x02); //Start Waveform playback
Wire.write(0x01);
Wire.endTransmission();
}
```

2.5 mp-Labtronix controller

mp-Labtronix controller	Order code: mp-Labtronix
Access to the full range of driving parameters. A system for the professional evaluation of the micropumps.	
Dimensions	7.5 x 16 x 20 cm 2.983 x 6.299 x 7.874 in.
Weight	ca. 800 g
Adjustable parameters	amplitude, frequency, signal form
Amplitude range	0 – 250 V
Frequency range	0 - 300 Hz
Signal form	SRS, rectangular, sine
Power supply	mains adaptor
Current consumption	750 mA at 7.5 V
USB-Port	one
connectable micropumps	mp6-liq, mp6-gas, mp6-gas+, mp6-pi, mp6-pp: 1x

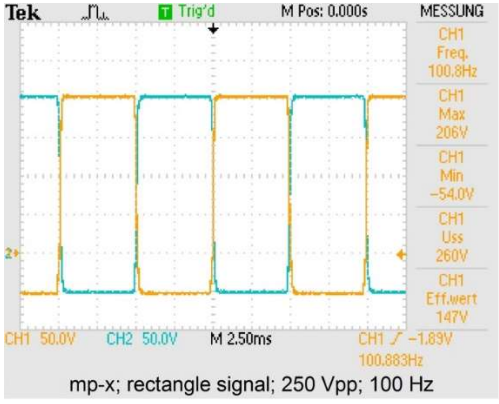
2.5.1 Electrical signal form

SRS



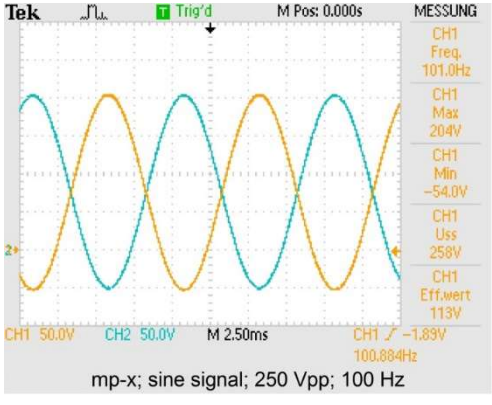
mp-x; SRS-signal; 250 Vpp; 100 Hz

Rectangular



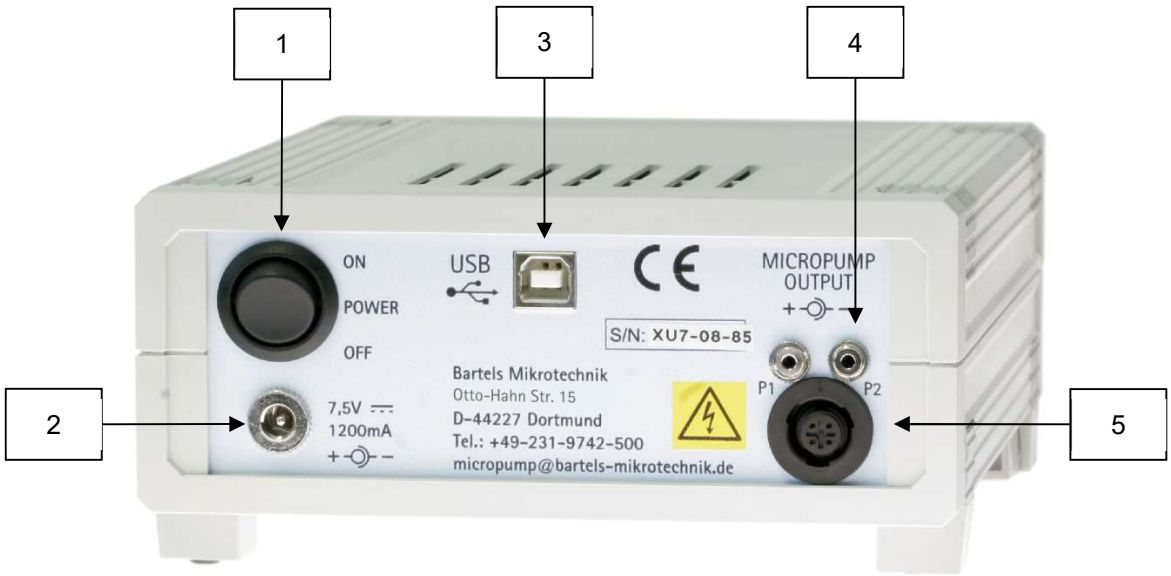
mp-x; rectangle signal; 250 Vpp; 100 Hz

Sine



mp-x; sine signal; 250 Vpp; 100 Hz

2.5.2 Connecting the pump to the mp-Labtronix



1 Main power switch
2 Power supply connector
3 USB interface for computer connection
4 Connections port for one or two mp5
5 Connection port for one micropumps of the mp6-series

Please note that it is only possible to connect either one micropump of the mp6-series or a maximum of two mp5 to the mp-Labtronix, otherwise a maximum voltage drop is possible!

Step 1: Plug the micropump control cable into the corresponding micropump connector.

Step 2: Check the mains adaptor plug polarity. It is pictured next to the power supply connector at the back of the controller. If the plug polarity is wrong, the controller cannot work.
Please make sure that the setting on the included connector is attuned to 7.5 V.

Step 3: Connect the mains adaptor with the power supply connection.

Step 4: Plug the mains adaptor into a mains socket.

Step 5: Now you can start the control unit with the main power switch.

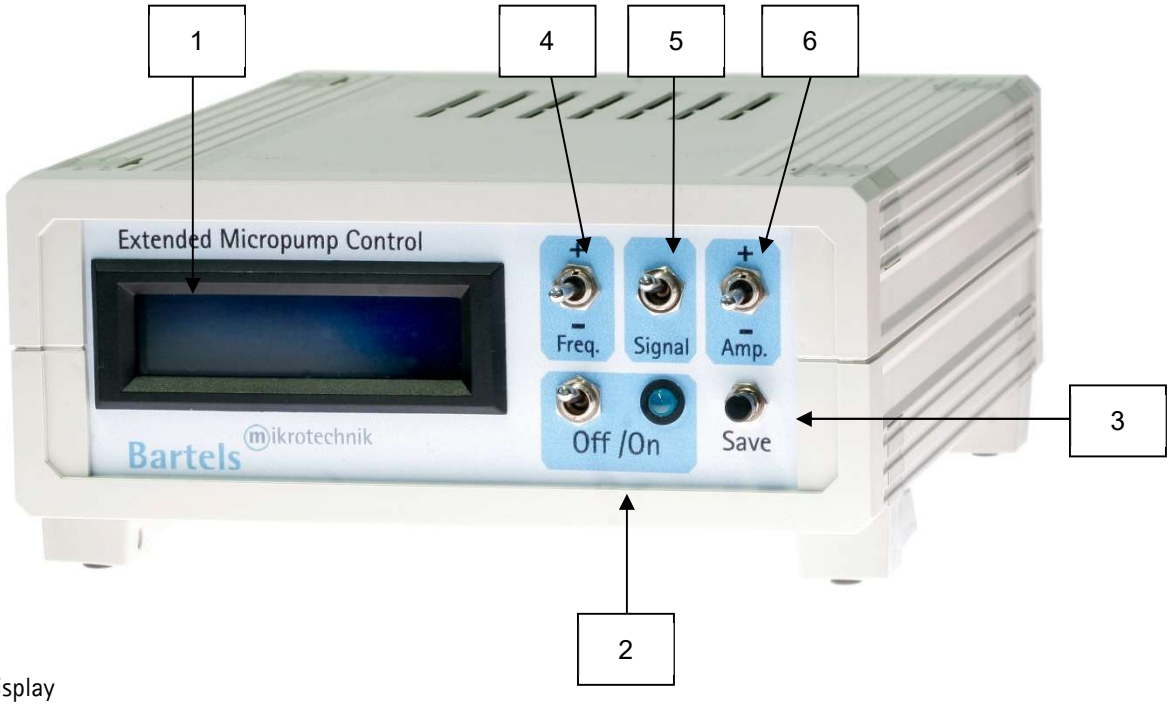
⚠ DANGER
THE "MICROPUMP OUT" CONNECTOR CAN CARRY HIGH VOLTAGE! PLUG IN THE MICROPUMP CONTROL CABLE ONLY WHEN THE MP-LABTRONIX CONTROLLER IS SWITCHED OFF!

2.5.3 Operation of the mp-Labtronix

The mp-Labtronix provides three parameters to be selected independently of each other to control the micropumps:

- Frequency
- Amplitude
- Signal form

It is possible to change the settings while the pump is operating. However, to extend the lifetime it is advisable to turn off the micropump first before changing the settings.



LCD Display

On/Off-switch: Push the on/off switch to turn the pump and control diode on or off.

Save: Push the save switch to save the present settings.

Freq.: Push the switch up to raise and down to reduce the frequency.

Signal: Push the switch once to changes between SRS, rectangular or sine.

Amp.: Push the switch up to raise and down to reduce the amplitude.

To operate the micropump, prepare the controller as described in point 2.5.2 and follow the steps below:

Step 1: Choose a frequency by pushing the frequency switch up and down.

Step 2: Choose one of the signal forms by pushing the signal switch.

Step 3: Set the amplitude by pushing the amplitude switch up and down.

Step 4: Push the on/off switch to turn on the micropump and the control diode.

Step 5: Push the on/off switch again and the micropump will stop while the control diode turns off.

The Save-switch will store the current settings.

For shutdown of the controller please switch off the mp-Labtronix first and then the main power-switch on the backside of the controller. Then disconnect the power plug. Do not unplug the micropump before switching off the controller.

2.5.4 Operation via USB port (after installation of the drivers)

The driving parameters can be set via your PC. For this purpose you can use software (or programming language) of your choice that is capable of sending commands to serial COM-Ports. In the example below the Windows software Hyperterminal is used. As the Hyperterminal is not available in Windows 7 anymore, we recommend freeware terminal software like PuTTY (http://www.putty.org).

Please note that the mp-Labtronix requires a time of approx. 100 ms to process a command. Therefore real-time flow control is not possible. Regardless of the time for processing a command, every change (amplitude, frequency or signal form) results in a jump in flow.

Hyperterminal example:

Step 1: Connect the control unit to your computer and turn it on.
Step 2: Start Windows Hyperterminal. Every new session has to be titled.
Step 3: Choose the COM-port specified in the device manager.
Step 4: The connection-settings have to be:
Bits per second: 9600; Data bits: 8; Parity: none; Stop bits: 1; Flow control: none

2.5.5 Possible commands (followed by the enter key)

bon	turns the micropump on
boff	turns the micropump off
F(1-300) F100	sets the required frequency between 1 and 300 Hz here as an example 100 Hz.
A(1-250) A100	sets the required frequency between 1 and 250 Vpp here as an example 100 Vpp
MS	sets signal form modus (S)ine
MR	sets signal form modus (R)ectangular
MC	sets signal form modus SRS
(enter key)	displays present settings of the control unit

The control unit can also be used via LabView, Matlab or other programs. Using LabView, please ensure that the "NI-Serial" package is installed. This will be installed in regular cases together with LabView, but sometimes this option is skipped during install.

It can be downloaded on the National Instruments website as the "NI-Serial" package:
http://joule.ni.com/nidu/cds/view/p/id/2316/lang/en

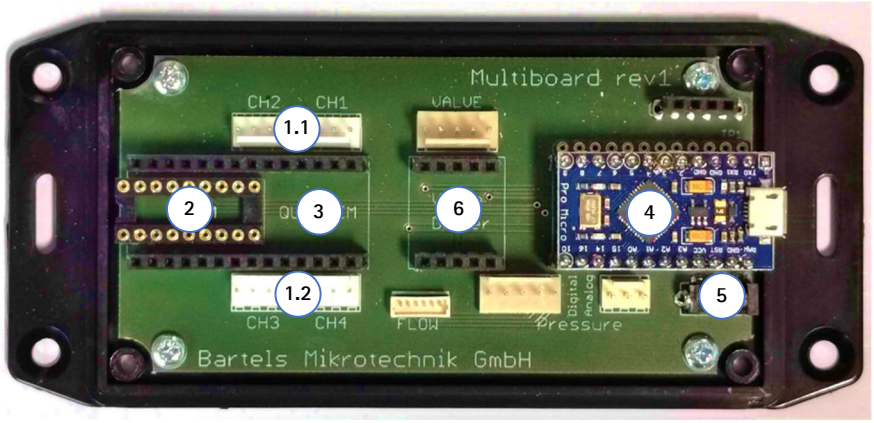
Afterwards it is possible to select the right COM-port for your mp-Labtronix in the VISA resource name.
On request, we can send a package of LabView-Routines as an example of implementation.

We also offer to develop customer specific software for the application of the controller units.

2.6 Evaluation Board: mp-Multiboard

The mp-Multiboard is available in different combinations and has the following standard components included:

- mp-Multiboard board
- micropump mp6
- Micro-USB cable
- **selective**: pump driver mp-Lowdriver, mp-Highdriver, mp-Highdriver4



The figure above shows the following components:

1.1 & 1.2 Pump cable harness connector
3. mp-Highdriver4 socket
5. Power supply terminal
2. mp-Lowdriver / Highdriver socket
4. Arduino microcontroller
6. Socket for valve driver

2.6.1 Setup instructions

Make sure pump driver, pump cable harness and pumps are plugged in correctly; install the Arduino Pro Micro USB driver; connect the Multiboard via USB cable to the computer; start the Multiboard App
Do not disconnect pumps or driver while Multiboard is powered (via USB or otherwise)!

2.6.2 Electrical Characteristics

The Multiboard is powered via USB connection of the microcontroller. It does comply with the USB standard and does not require more than 500mA to operate under normal conditions. If in any case more current is needed on the board, an external power supply can be connected to the power supply connector labeled "POWER".

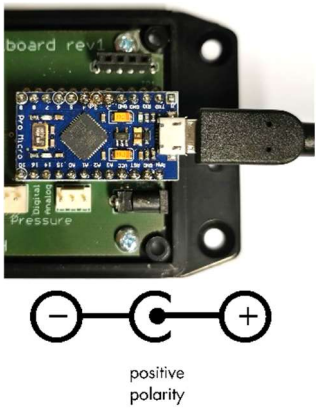


Figure 10: Connector Polarity

The power supply connector is directly wired to the RAW and GND pins of the microcontroller and tied to its onboard voltage regulator. Recommended input voltage for this connector is 7,5-12V. The mate plug should have an outer diameter of 3,5 mm and an inner diameter to be able to accept the 1 mm pin. See Figure 10 for connector polarity. A suited power supply is available in our store.

For all other characteristics of the microcontoller or the pump driver please refere to the corresponding manual or datasheet.

Information about the pump drivers and the mp6 micropumps is available on our download-page:
https://www.bartels-mikrotechnik.de/downloads/

2.6.3 Pump drivers

The Multiboard is compatible to our mp-Lowdriver, mp-Highdriver and mp-Highdriver4. Depending on the driver attached to the Multiboard, capabilities and ranges of amplitude and frequency can change. Please refer to the according manual for details. When attaching the pump driver make sure to follow the correct orientation as shown in the pictures below (notice the white dot always pointing outwards).

Use caution when removing the pump driver from the board in order to not bend the pins of the chip. IC pliers are recommended to be used for extraction.

2.6.4 Valve driver

The socket labeled "Valve driver" accepts our current dual valve driver that is able to drive two Takasago valves. The valves are current driven and the driver takes care of generating and supplying a constant current on two separate outputs on demand. The two outputs are digitally controlled via the Multiboard.

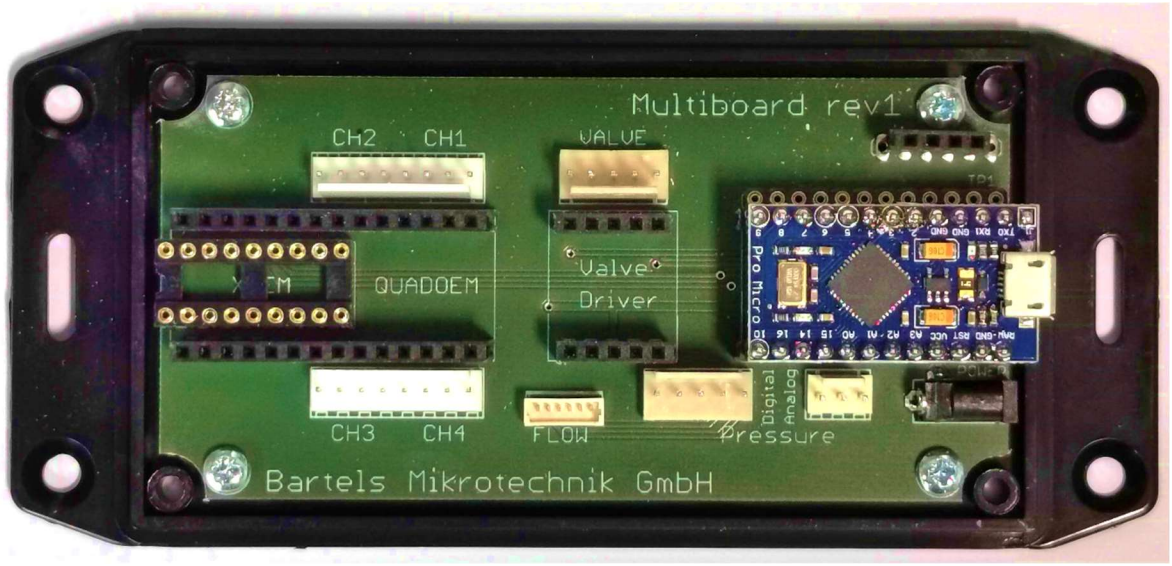
Attention: Since the valve driver needs a minimum of 4.5V to operate correctly, an external power supply needs to be attached to the Multiboard. The 5V rail of the Multiboard supplied by the USB is not reliable enough, to guarantee operation. See chapter

Electrical Characteristics **for more details.**

2.6.5 Sensors

Since App version 1.5 (and Firmware 20210428) the Multiboard supports readout of digital flow and pressure sensors. The liquid flow-sensors of type SLF3S-0600F and SLD3S-1300F from the manufacturer Sensirion can be attached to the connector labeled "Flow" via a straight 6-pin ribbon cable. Both sensors can be bought form our store with the ribbon cable included. For measuring pressure, we have implemented the readout for Honeywell ABP-series 15PSI digital pressure sensor. This one is sold on our store with a cable and 5-pin connector already attached, fitting the pinheader labeled "Pressure digital".

2.6.6 Auxiliary connectors



The Multiboard has multiple auxiliary connectors available. Some of these are not in use at the moment and will be documented in this manual once fully implemented. Nevertheless, here is a list of the connectors and their (future) use:

CH2 / CH1, CH3 / CH4	Two 8-pin connectors for pump cable harness. Each cable harness has two FFC connectors on the other end to connect to the micropump flex cable
Valve	5-pin connector for two active valves
TP1	12-pin header for custom purposes
Flow	6-pin connector for an I²C flow-sensor
Pressure digital	5-pin connector for an I²C pressure-sensor
Pressure analog	3-pin connector for an analog pressure-sensor
Power	Barrel power supply connector

2.6.7 USB-driver

The most current USB-driver is commonly installed with the Arduino IDE. If you don't want to install the whole IDE and just need the driver, you can download the IDE including the driver as ZIP archive from the Arduino download site (https://www.arduino.cc/en/software) and install just the USB driver.
A driver-package is also included in the Multiboard App download.

2.6.8 Multiboard App

The Multiboard App is a Software tool that enables you to control various features of the Multiboard remotely via USB. It interfaces with the Arduino via the USB-Serial interface and calls the protocol commands of the Firmware. A description of the protocol and list of commands will be available in the future, allowing to use the Multiboard using other software like LabVIEW, Matlab and python.

After executing the app, the following screen opens up:

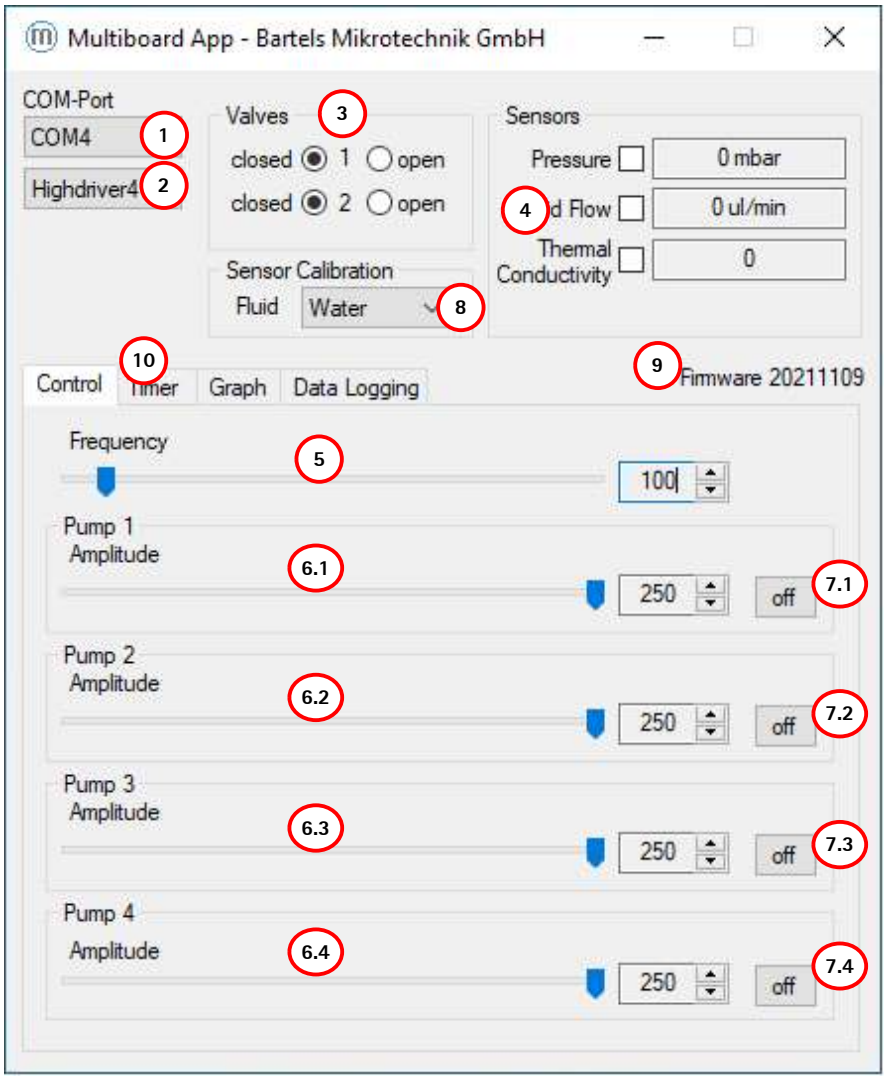


Figure 11: Multiboard App GUI

Select the COM-Port of your Multiboard in the list-box (1), followed by the pump driver currently inserted on the Multiboard in the list-box below (2).
Now you are able to adjust each pump's amplitude separately (6.1 - 6.4), switch every pump on and off (7.1 – 7.4) and change the frequency for all pumps globally (5).
The radio-buttons in the Valves-group (3) can be used to open and close active valves connected to the Multiboard, when a valve driver is plugged onto the Multiboard.
The checkboxes in the Sensors-group (4) can be used to enable & disable sensor readout when sensors are plugged into the Multiboard.
To change the calibration of the liquid sensor a fluid can be selected in the listbox (8).
The Label (9) shows the current Firmware version on the Multiboard. Always check if newer Software is available on our Website to get the latest features of the Multiboard unlocked. The App and Firmware should be both up to date to make sure, there are no compatibility issues.
By selecting the Timer-tab (10) you switch over to the Timer-mode user interface, showing settings to setup a timed on/off cycle of each pump:

6

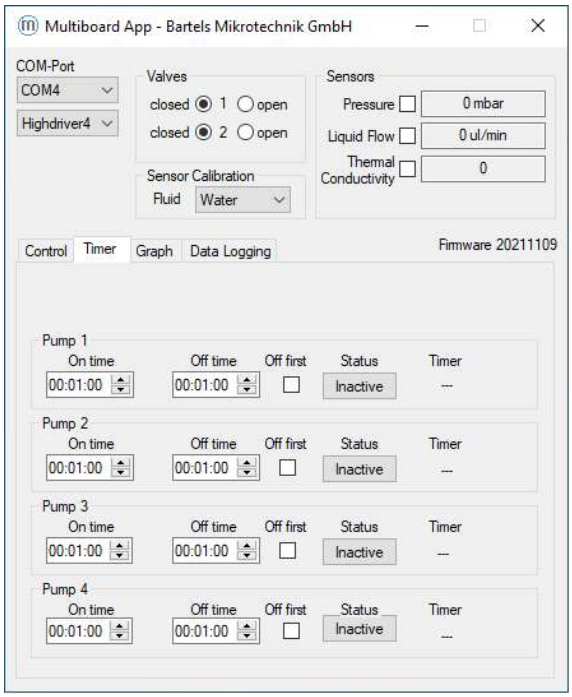
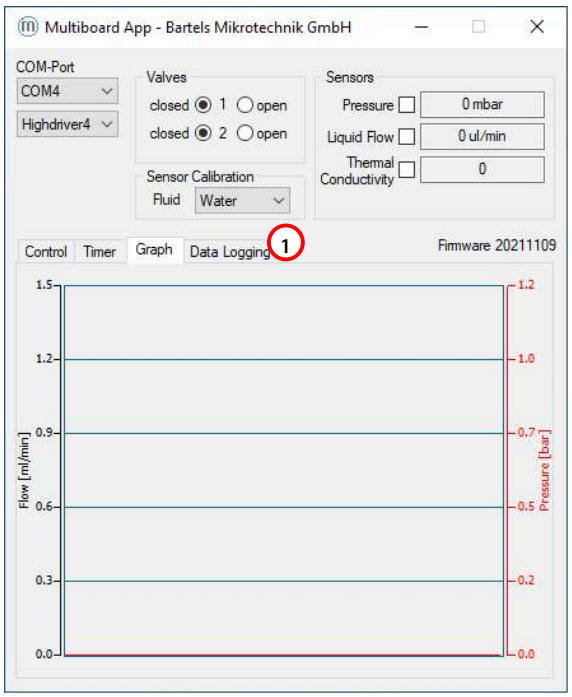


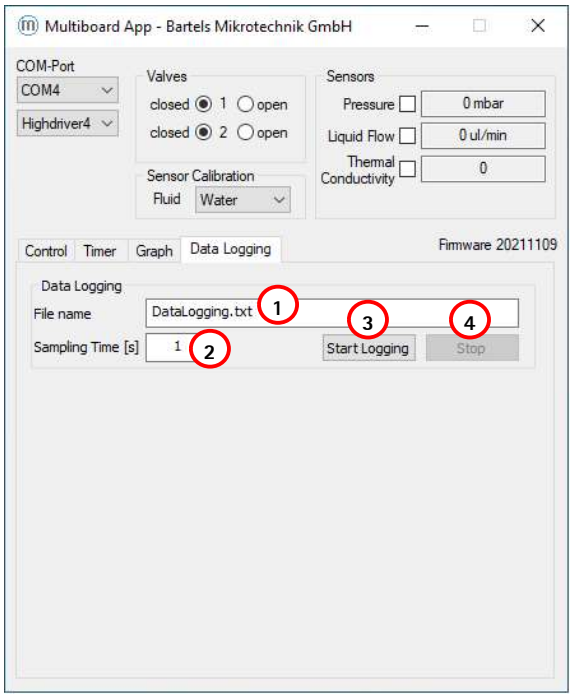
Figure 12: Multiboard App GUI in "Timer mode"

The "On time" (1) defines how long the pump is to stay activated (pumping), while the "Off time" (2) defines how long the pump is to stay deactivated before starting a new cycle. When "Off first" (3) is ticked, the cycle starts with the pump deactivated. By pressing the "Status" button (4) this periodic pumping can be activated/deactivated for each pump. The indicator "Timer" (5) shows the current state of the pump (on/off) and the time remaining for this part of the cycle.

The on/off button on the Control-tab is disabled and greyed while timer mode is active. The frequency and amplitude controls are not disabled since they still take effect on the pumps in timer mode.
By selecting the Graph-tab (6) you switch over to a Graph showing the current sensor values of the attached (and enabled sensors):



The Data Logging-tab (1) allows the recording of your selected sensors.



A file name can be set inside the dialog field (1) and the sampling time can be changed in in dialog field (2). The "Start Logging"-button begins the logging of the sensor data and the "Stop"-button ends the measurement. The file format will be a .txt that can be easily imported to excel if needed.

2.6.9 USB/Serial Communication Protocol

The Multiboard is connected to the computer via the USB connector of the Arduino pro micro. It acts as a USB-Serial interface so the Multiboard appears on the computer as a COM-Port inside the device manager. To communicate with the Multiboard any software capable of sending and receiving data to/from a serial port can be used. Programs like Hyperterminal and PuTTY can be used to send commands directly, while programs like LabVIEW, Matlab and Python can be used to write software that interacts with the user and the Multiboard allowing for more automation and ease of use. A list of commands used to control the Multiboard can be found in the table below:

SELECTLOWDRIVER SELECTHIGHDRIVER SELECTQUADDRIVER SELECTNONE	Tells the Multiboard which driver is currently on the board and initializes the driver
PON	turns all pumps on
POFF	turns all pumps off
P<p>ON P1ON	Turns the selected pump <p> on Example: Turns pump #1 on
P<p>OFF P1OFF	Turns the selected pump <p> off Example: Turns pump #1 on
P<p>V<a> P1V250	Sets the amplitude for the selected pump <p> to the chosen value <a> Example: Pump #1 set to 250 Vpp
P<p>V? P1V?	Gets the amplitude for the selected pump <p> Example: Returns "250" meaning 250 Vpp for pump #1
F<f> F100	Sets the frequency (for all pumps) to the chosen value <f> Example: Sets the frequency to 100 Hz.
V1ON V2ON	Turns valve 1/2 on
V1OFF V2OFF	Turn valve 1/2 off

DPON DPOFF	Enables/disables pressure readout
DFON DFOFF	Enables/disables flow readout
MC	sets signal form modus SRS
V	Displays the current firmware version
(enter key)	displays current settings of the Multiboard

2.6.10 Package Dimensions

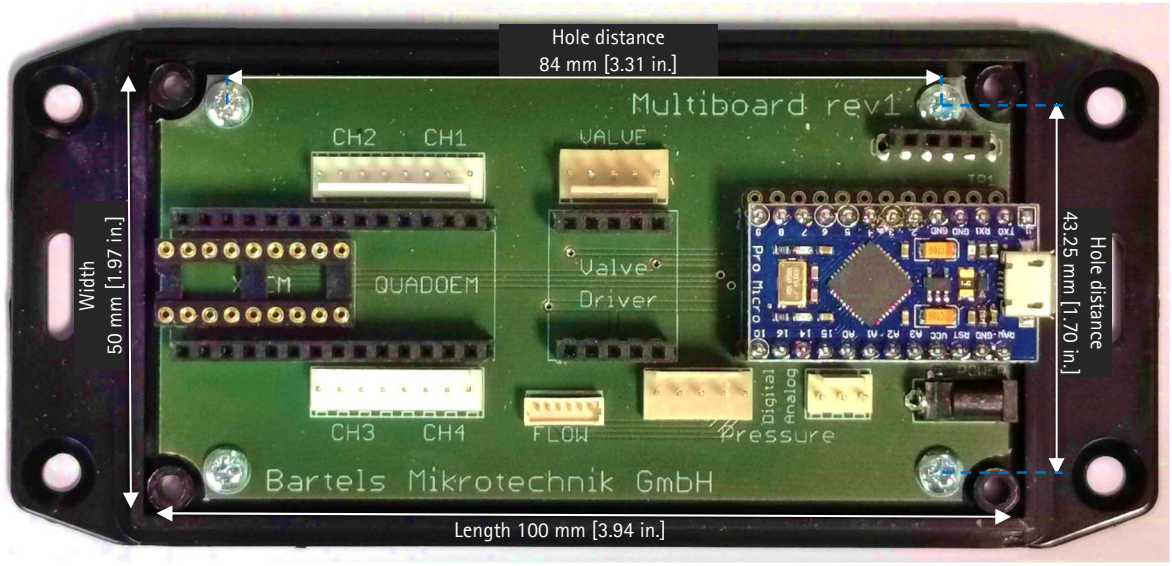
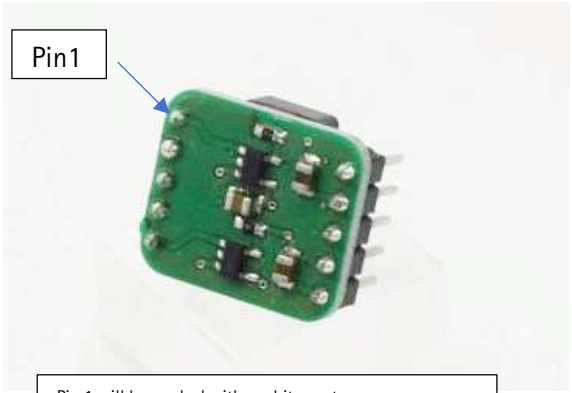


Figure 13: Package Dimensions

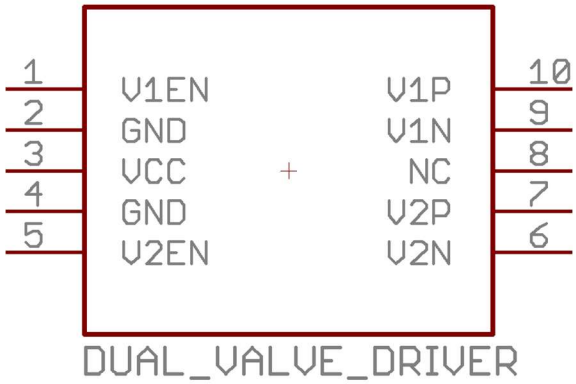
2.7 mp-valvedriver

The valve driver is a small, easy to use driving circuit developed for the Takasago SMV-2R-BN1F shape memory alloy valve as an accessory for the micropumps of the mp6-series.

2.7.1 Pin assignment



Pin 1 will be marked with a white spot.



Pin	Name	Function	Pin	Name	Function
1	V1EN	Enable pin. Float to enable	10	V1P	Valve 1 output pin+ *
2	GND	Ground	9	V1N	Valve 1 output pin- *
3	VCC	Input supply voltage	8	NC	not connected pin
4	GND	Ground	7	V2P	Valve 2 output pin+ *
5	V2EN	Enable pin. Float to enable	6	V2N	Valve 2 output pin- *

* Note: The Takasago valves do not have polarity. So positive and negative valve output pins can be swapped.

2.7.2 Technical specifications valve driver

Valve driver	Order code: mp-valve driver
The valve driver enables/disables up to two Takasago SMV-2R-BN1F shape memory alloy valves. It is meant to be used in combination with a micro controller. The valve is a normally-closed valve, so it is closed when there is no power supplied.	
Dimensions	17,78 x 15,24 x 1,6 mm
Power supply	4,5 – 17 VDC (5 V recommended for optimized performance)
Current consumption (one/two valves)	85 mA/170 mA at 5 V

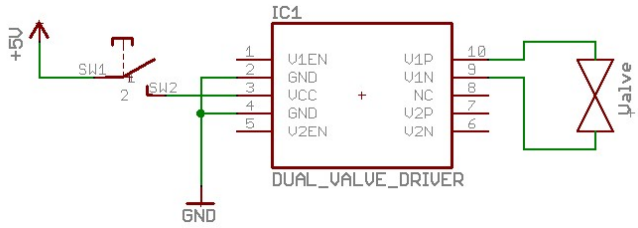
2.7.3 Electrical Characteristics

Parameter	Symbol	Conditions	Min	Typ.	Max	Unit
Supply voltage	VCC		4.5	5	17	V
Input current	IDD	Typical VCC = 5 V		85		mA
Source current	V1EN V2EN				±100	µA
VCC Shutdown supply current		EN = 0 V, VCC = 12 V	2.0	3.7	9	µA

2.7.4 Examples of circuiting the mp-valve driver

Manual operation

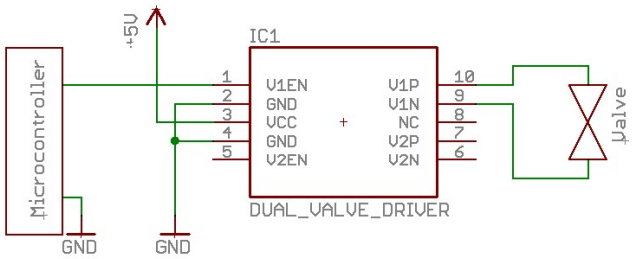
The first example shows the valves being operated manually by switching the power supply on and off. The valve driver is enabled by default, so the enable pins can be left floating.



Schematic 14: Manual operation

Operation via microcontroller

The second example shows the valves being operated by a microcontroller. The enable pins are controlled by the micro and the power supply is connected permanently.



Schematic 15: Operation via microcontroller

3 Important Notices

This operating manual contains necessary instructions for the installation, commissioning, operation and maintenance of the mp6-series. The manual is intended to help you achieving optimal results in a short time and shall also assist avoiding possible sources of errors. The operating manual of the controllers and the accessories are available separately.

The products have been designed with state-of-the-art technology and in accordance with all relevant safety regulations. However, a risk of damage to the units, other property, the operator and/or other persons cannot be fully excluded.

Always ensure that specialized and trained personnel will comply with the following general instructions. Therefore, please keep this manual and hand out copies as required.

All values are approximate and no guarantee of specific technical properties.
Changes in the course of technical progress are possible without notice.

3.1 Warranty

The mp6 micropumps have been developed for the transport of gases or liquids. The controllers have been developed for operating the mp6 micropumps. Bartels Mikrotechnik can assume no liability for damages resulting from the pump media. This applies especially for hazardous fluids.

The mp6 micropumps must be operated with Bartels Mikrotechnik electronics. Bartels Mikrotechnik GmbH cannot guarantee the proper work of the units with customer specific electronics. If other controllers than the ones from Bartels Mikrotechnik are used, Bartels Mikrotechnik disclaims any warranty.

Bartels Mikrotechnik assumes no liability for abnormal handling, improper or negligent use of the mp6 micropumps and the controller that is not conform to the specified purpose of the system. This applies especially for micropump controllers, components and systems of other manufacturers, which have not been certified by Bartels Mikrotechnik.

We guarantee that the mp6 micropumps comply with the actual state of scientific and technical knowledge and due to this, the operational risks are limited to a minimum.

Bartels Mikrotechnik GmbH warrants solely to the original purchaser of this product for a period of 12 months (one year) from the date of delivery that this product shall be of the quality, material and workmanship defined in Bartels Mikrotechnik GmbH published specifications of the product. Within such period, if proven to be defective, Bartels Mikrotechnik GmbH shall repair and/or replace this product, in Bartels Mikrotechnik GmbH's discretion, free of charge to the Buyer, provided that:

- notice in writing describing the defects shall be given to Bartels Mikrotechnik GmbH within fourteen (14) days after their appearance
- such defects shall be found, to Bartels Mikrotechnik GmbH's reasonable satisfaction, to have arisen from Bartels Mikrotechnik GmbH's faulty design, material or workmanship
- the defective product shall be returned to Bartels Mikrotechnik's factory at the Buyer's expense
- the warranty period for any repaired or replaced product shall be limited to the unexpired portion of the original period.

This warranty does not apply to any equipment which has not been installed and used within the specifications recommended by Bartels Mikrotechnik GmbH for the intended and proper use of the equipment.

EXCEPT FOR THE WARRANTIES EXPRESSLY SET FORTH HEREIN, Bartels Mikrotechnik GmbH MAKES NO WARRANTIES, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THE PRODUCT. ANY AND ALL WARRANTIES, INCLUDING WITHOUT LIMITATION, WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, ARE EXPRESSLY EXCLUDED AND DECLINED. Bartels Mikrotechnik GmbH is only liable for defects of this product arising under the conditions of operation provided for in the data sheet and proper use of the goods. Bartels Mikrotechnik GmbH explicitly disclaims all warranties, express or implied, for any period during which the goods are operated or stored not in accordance with the technical specifications. Bartels Mikrotechnik GmbH does not assume any liability arising out of any application or use of any product or circuit and specifically disclaims any and all liability, including without limitation consequential or incidental damages. All operating parameters, including without limitation recommended parameters, must be validated for each customer's applications by customer's technical experts. Recommended parameters can and do vary in different applications. Bartels Mikrotechnik GmbH reserves the right, without further notice, (i) to change the product specifications and/or the information in this document and (ii) to improve reliability, functions and design of this product.

3.2 Warning, Personal Injury

Bartels Mikrotechnik GmbH rejects any responsibility for damages to persons or property resulting from non-compliance with the instructions in this manual. In this case all warranties shall be void.

Do not use this product as safety or emergency stop devices or in any other application where failure of the product could result in personal injury. Do not use this product for applications other than its intended and authorized use. Before installing, handling, using or servicing this product, please consult the data sheet and application notes. Failure to comply with these instructions could result in death or serious injury. If the Buyer shall purchase or use Bartels Mikrotechnik GmbH products for any unintended or unauthorized application, Buyer shall defend, indemnify and hold harmless Bartels Mikrotechnik GmbH and its officers, employees, subsidiaries, affiliates and distributors against all claims, costs, damages and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if Bartels Mikrotechnik GmbH shall be allegedly negligent with respect to the design or the manufacture of the product.
Moreover, please note that components of the controller and pump are operating with high voltage. Therefore, persons wearing pacemakers are recommended to avoid the operating system. **Do not open the housing of the micropump and the controllers.**

The use of liquids, which may alone or in combination create explosive or otherwise health-endangering conditions (including vapors) is not permitted.

All work in connection with the installation, assembly, commissioning/decommissioning, disassembly, operation, servicing, cleaning and repairing of the pump and the controller must be carried out by qualified, suitably trained and instructed personnel. Work on electrical components and assemblies must be carried out by personnel with the necessary qualifications and skills.

We guarantee that the micropumps comply with the actual state of scientific and technical knowledge and due **to this, t**he operational risks are limited to a minimum.

ESD Precautions: The inherent design of the active electronic components, i.e. mp-Lowdriver, -Highdriver, -Highdriver4, Valvedriver and -Labtronix causes it to be sensitive to electrostatic discharge (ESD). To prevent ESD-induced damage and/or degradation, take customary and statutory ESD precautions when handling this product.

3.3 Declaration of conformity

Bartels Mikrotechnik GmbH declares that the products are compliant to the RoHS directive 2011/65/EU. The controllers comply with the requirements of EMV 2014/30/EU and CE markings have been affixed to the devices. Additionally, the controllers are also compliant to the EU Low Voltage Directive 2014/35/EU.

4 Company information

Bartels Mikrotechnik is a globally active manufacturer and development service provider in the field of microfluidics. In the microEngineering division, the company supports industrial customers in the modification, adaptation and new development of high-performance and market-oriented product solutions through the innovative means of microsystems technology. The second division, microComponents, produces and distributes microfluidic products and systems, especially for miniaturized and portable applications. Our key products are micropumps that convey smallest quantities of gases or liquids and are used in a variety of ways in biotechnology, pharmaceuticals, medical technology and numerous other applications.

Bartels Mikrotechnik with passion for microfluidics!

Contact us:

Bartels Mikrotechnik GmbH
Konrad-Adenauer-Allee 11
44263 Dortmund Germany

www.bartels-mikrotechnik.de
info@bartels-mikrotechnik.de
Tel: +49-231-47730-500

Follow Bartels: