microComponents

# Operating Manual for the Micropump Driver mp6-QuadKEY

Rev. 1.6 (01.2021)

Content

# General Description

The mp6-QuadKEY is an evaluation board that allows to control up to four mp6 micropumps simultaneously with various waveforms, frequencies ranging from 50 Hz up to 800 Hz and amplitudes from 10 Vpp to 250 Vpp. The mp6-QuadKEY has the mp6-QuadOEM and an Arduino nano compatible microcontroller onboard. Both can be fully configurated by the user to fit the required application demands. The pump driver outputs of the mp6-QuadOEM are wired to four Molex flex cable connectors so that Bartels Mikrotechnik micropumps (mp6-liq/mp6-gas/mp6-PI/mp6-PP) can be directly attached. The mp6-QuadOEM is connected to the microcontroller through the $I^2C$ bus. Most of the microcontroller I/O pins are externally connectable through a pin header and/or the prototyping area on the bottom side of the board. A mini-USB connector on the microcontroller can be used to supply power, upload software and for serial communication. An external power supply connector is also available to power the board from an external source.

# Proper Use

### Intended Purpose

The mp6-QuadOEM is designed as a next step from the mp6-QuadEVA board to control up to four micropumps for gas pumping, i.e. four pieces of mp6-gas micropumps. Nevertheless, it is also possible to pump liquids, with either the mp6-gas, mp6-pp or the standard mp6-liq pump; though the higher frequencies will not result in a performance boost.

If liquids should be pumped, please regard the following:

The micropump is intended for pumping liquids or gases with varying flow rates controlled by the electronics. The mp6-QuadOEM is intended as a pump driver for mp6-gas/mp6-liq/mp6-pi/mp6-pp.

Any other use of the micropump or controller unit is deemed improper.

Do not make any modifications or extensions to the pump or controller without the prior written consent of the manufacturer. Such modifications may impair the safety of the unit and are prohibited! Bartels Mikrotechnik GmbH rejects any responsibility for damage to the unit caused by unauthorized modifications to the pump and risk and liability are automatically transferred to the operator.

### Misuse

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

### Staff Selection and Qualification

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.

Safety Notice

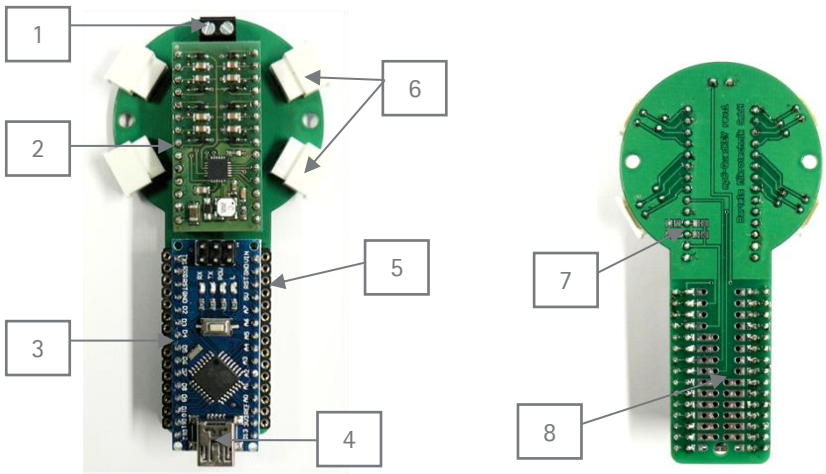
**The mp6-QuadOEM generates voltages of up to 250 Vpp.** All parts of the controller can carry voltages in this range. Therefore, the board should only be used by qualified personnel. Although the output power of the module is very low, **proper insulation according to the application conditions needs to be considered by the customer.** This especially applies to the bottom side of the PCB. Contact with water or other liquids needs to be prevented. The pump must not be unplugged while the board is active.

| ⚠ DANGER |
| --- |
| THE DEVICE CAN CARRY HIGH VOLTAGE!<br><br>BE CAREFUL, WHILE CONNECTING AND HANDLING THE BOARD! |

# Overview

The mp6-QuadKEY is available as set of the following components:

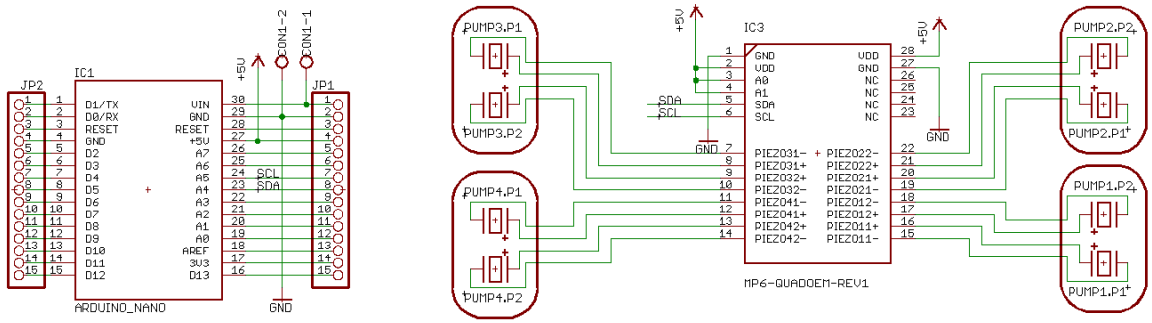
- mp6-QuadKEY board
- Mini-USB cable
- 4x micropump from mp6-series (4 mp6-gas or 4 mp6-liq or 4 mp6-pi or 4 mp6-pp)
- CD with Arduino demo Sketch and USB-driver



In the figures above the components are:

1. Power Supply Terminal
2. mp6-QuadOEM
3. Arduino Nano or compatible Microcontroller
4. USB connector
5. Pin headers
6. Flex cable connectors for Micropumps (mp6)
7. I²C address selector
8. Prototyping area

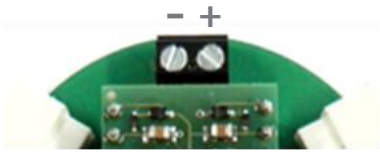
# Board Schematic




1 Board Schematic

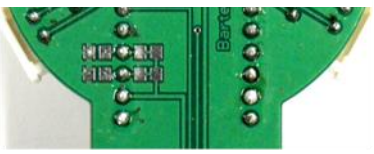
# Electrical Characteristics

The power supply connector is directly wired to the VIN and GND pins of the microcontroller and tied to its onboard voltage regulator. Recommended input voltage for this connector is 7-12V and absolute voltage limits are 6-20V.




2 Connector Polarity

The address pins (A0/A1) of the mp6-QuadOEM are tied to +5V. So the I²C address defaults to 0x7B. If necessary this can be changed by removing the bridge between A0/A1 pins and +5V and connecting one or both pins to GND.



3 Address pins

For all other characteristics of microcontoller or mp6-QuadOEM please refer to the corresponding manual or datasheed. Information about the Arduino nano can be found at:
https://www.arduino.cc/en/Main/ArduinoBoardNano

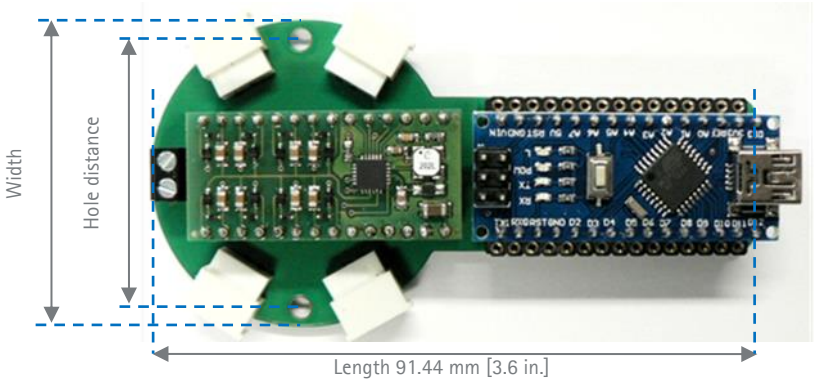
Information about the mp6-QuadOEM and the mp6-series micropumps is available at:
http://www.bartels-mikrotechnik.de/index.php/en/products/electronic

# Arduino Demo Sketch

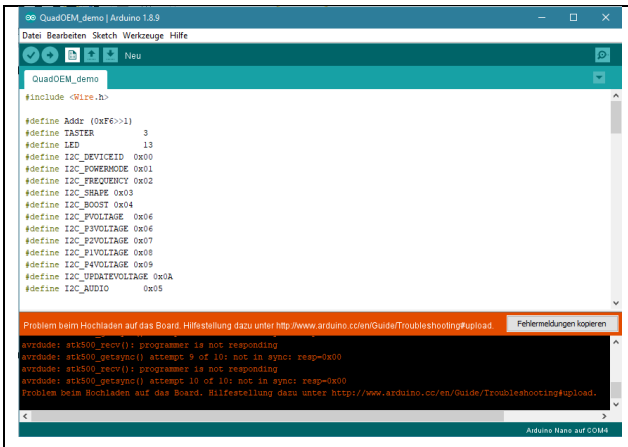
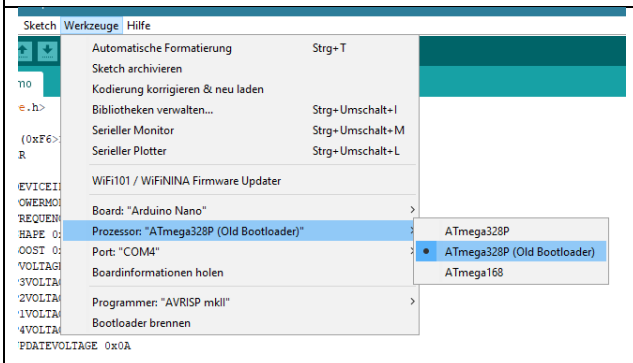
The following code shows the setup routine implemented in the demo sketch delivered with the mp6-QuadKEY. Also enabling/disabling pumps and changing the frequency is shown and demonstrates the ease of controlling the mp6-QuadOEM with just a few commands.

```
#include<Wire.h>
#define Addr 0x7B
#define TASTER          3
#define LED             13
#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

int debounce = 0 ;

int mode = 0;
#define MODE_OFF    0
#define MODE_50     1
#define MODE_100    2
#define MODE_200    3
#define MODE_400    4
#define MODE_800    5
#define MODE_MAX    6

// the setup routine runs once when you press reset:
void setup() {
Wire.begin();
Serial.begin(9600);
Serial.println();
pinMode(LED, OUTPUT);
pinMode (TASTER,INPUT_PULLUP);
Wire.beginTransmission(Addr);
Wire.write(I2C_POWERMODE); // start address = 0x01
Wire.write(0x00);       //Adress 0x01 = 0x01 (enable)
Wire.write(0x40);       //Adress 0x02 = 0x40 (100Hz)
Wire.write(0x00);       //Adress 0x03 = 0x00 (sine wave)
Wire.write(0x00);       //Adress 0x04 = 0x00 (800KHz)
Wire.write(0x00);       //Adress 0x05 = 0x00 (audio off)
Wire.write(0x1F);       //Adress 0x06 = 0x00 (VO1)
Wire.write(0x1F);       //Adress 0x07 = 0x00 (VO2)
Wire.write(0x1F);       //Adress 0x08 = 0x00 (VO3)
Wire.write(0x1F);       //Adress 0x09 = 0x00 (VO4)
Wire.write(0x01);       //Adress 0x0A = 0x00 (update)
Wire.endTransmission();
}

// the main loop runs repeatedly:
void loop() {
if (digitalRead(TASTER)==LOW) {
if (!(debounce&0x01)) {
mode=(mode+1)%MODE_MAX;
switch (mode) {
case MODE_OFF:
Wire.beginTransmission(Addr);
Wire.write(I2C_POWERMODE);    // start adress
Wire.write(0x00);       // disable pumps
Wire.endTransmission();
digitalWrite(LED,LOW);
break;
case MODE_50:
Wire.beginTransmission(Addr);
Wire.write(I2C_POWERMODE);   // start adress
Wire.write(0x01);           // enable pumps
Wire.write(0x00);           // frequency 50 Hz
Wire.endTransmission();
digitalWrite(LED,HIGH);
break;
case MODE_100:
Wire.beginTransmission(Addr);
Wire.write(I2C_FREQUENCY);      // start adress
Wire.write(0x40);       // frequency 100 Hz
Wire.endTransmission();
break;
case MODE_200:
Wire.beginTransmission(Addr);
Wire.write(I2C_FREQUENCY);      // start adress
Wire.write(0x80);       // frequency 200 Hz
Wire.endTransmission();
break;
case MODE_400:
Wire.beginTransmission(Addr);
Wire.write(I2C_FREQUENCY);      // start adress
Wire.write(0xC0);       // frequency 400 Hz
Wire.endTransmission();
break;
case MODE_800:
Wire.beginTransmission(Addr);
Wire.write(I2C_FREQUENCY);      // start adress
Wire.write(0xFF);       // frequency 800 Hz
Wire.endTransmission();
break;
}
debounce|=0x01;
}
} else {
if (debounce&0x01) {
debounce&=~0x01;
}
}
delay(100);
}

```

# Package Dimensions




# Troubleshooting

| | |
|---|---|
|  | Since version 1.8.5 of the Arduino IDE a new booloader has to be selected from the tools menu for the upload to work.<br><br>If your IDE shows the error<br><br>`avrdude: stk500_getsync() attempt 1 of 10: not in sync: resp=0x00`<br><br>during upload, click on the Tools menu, submenu Processor and select **ATmega328P (Old Bootloader)**. |
|  | Now the upload process should finish without error. |

All values are approximate and no guarantee of specific technical properties.

Changes in the course of technical progress are possible without notice.

**Contact Data:**

Bartels Mikrotechnik GmbH
Konrad-Adenauer-Allee 11
44263 Dortmund Germany
www.bartels-mikrotechnik.de
info@bartels-mikrotechnik.de
Tel: +49-231-47730-500
Fax: +49-231-47730-501

**Visit our Website**

www.bartels-mikrotechnik.de

**for further information on applications.**

**Tutorials and helpful answers to frequently asked questions can be found in our FAQ**

www.bartels-mikrotechnik.de/en/faq-english/

**or on our YouTube channel**

https://www.youtube.com/user/BartelsMikrotechnik

**Social Media:**