PCBMotor Driver_version3 (Startkit version 3)

PCBMotor2018

In the following the theory and commands for PCBMotor's Motor Driver is explained with examples and suggestions for use.

Theory of operation:

All PCBMotors are designed to have a mechanical resonance in the 40–50 kHz range and the drive voltage must have a frequency close to the motor resonance to turn the motor. Since the resonance frequency changes with temperature and other external conditions, it is necessary to track the motor resonance by a control circuit in the driver.

The driver has a programmable step-down voltage supply to control the motor current while the frequency is set precisely to the resonance peak.

The driver control circuit is designed to measure the current when the motor is running and adjust the frequency to keep the current at the resonance peak.

If the motor runs continually self-heating will change the resonance towards a lower frequency (approx 80Hz per °C) and the driver must compensate. When the motor is stopped the oscillator frequency is stored in EPROM and used as starting point next time the motor is turned on.

The driver EPROM stores the individual settings for each motor so differences in resonance between motors will automatically be taken care of by the driver and when another motor is selected, the driver will change to the appropriate values. See more on this below in the discussion of the commands.

Installation

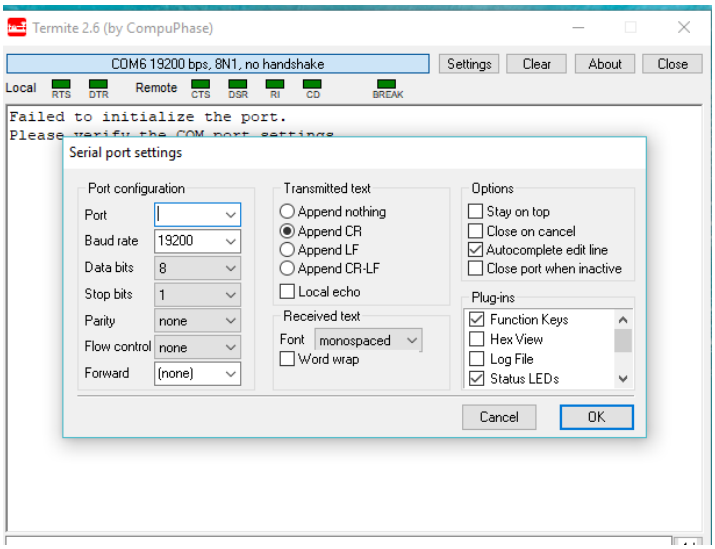
Start to download and install Termite
Download Termite – either from the Product tab “Software” in our shop, or directly from
https://www.compuphase.com/software_termite.htm

Go to the settings tab and click on the drop down for port to notices which COM port **Already in Use**.
In the screen print to the right there is no Com ported used.
Then check Port Configuration:

- Baud rate to be 19200
- Databit 8
- Stop bit 1
- Parity None
- Flow control none

For Transmitted text

- Only select: Append CR

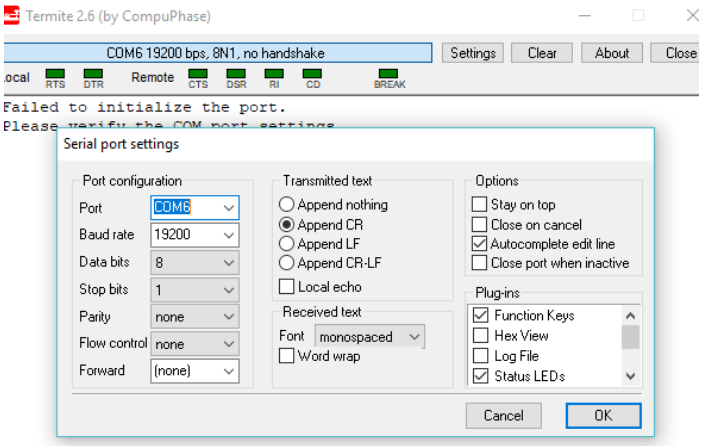


Then Plug the USB cable into the PC. Now the driver in the USB cable has been assign a Com port

When clicking on Port again you will see which new com port the device driver has been assigned. In this example the Com port is 6, and select port 6.

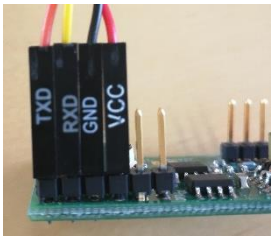
Keep Termite open and ready and now you can start to connect the wires from the USB cable to the driver, see next page

The other end of the cable with the 4 wires consist of, TXD, RXD, GND & VCC.
For the PC to communicate with the driver the TXD (a transmit signal) hall be connected to the driver receive pin, and similar for the RXD (a receive signal) and shall be connected to the driver transmit (TX)



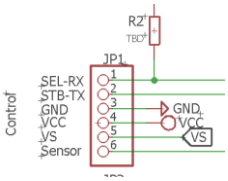
See the table below:

Cable



• TXD (Orange)	-> 1; RX
• RXD (Yellow)	-> 2; TX
• GND (Black)	-> 3; GND
• VCC (Red)	-> 4; VCC

Driver



It is possible to write custom communication software e.g. for LabView as long as all communication use ASCII characters.

It is a straight forward modem (RX/TX) communication. However, in other programs as Labview DTR is normal set to high:

The DTR modem signal is toggled low to do a software abort to stop a long sequence, so your LabView software should pull DTR inactive (high) when not used. Otherwise the motor will not turn (apart from the STB command!). There is a short note on this at the bottom of the help-txt: STOP commands D,S,G & U: Toggle DTR-signal on PC or short terminals DTR & GND

The modem setting should be:

- 19200
- 8
- 1
- N
- N

- Append CR

Special versions of the driver firmware also support SPI and I2C, but this has not been implemented in the current firmware.

Example of Available Commands and EEPROM memory:

The commands are listed by the driver in response to the 'h' command. The memory map for the system variables as well as for the individual motors is listed as well:

```
** StartKitv3 26-6-2018/PCBmotor **
COMMAND:  VALUE:  DESCRIPTION:
------------------------------
ADC>x?Stop        Stop repeats if (last measured mV/mA) > x. Also ADC<x?Stop
ADC>x?J=ADC       Set J (or K) = (last measured mV/mA) if > x. Also ADC<x?J=ADC
A/AI              Position/Index sensor voltage
BCR               BCR: Break for <CR>
BI/B+             BI: Search for Index, B+: Index break on
B-                B-: Break OFF
CW(CCW)   0..255  Motor voltage scaling (percent)
D(elay)   val     Delay (ms)
F(req)    val     VCO frequency (Hz, max 125000)
G/GN      +/-pos  Goto position. GN: No resonance tracking
I(nfo)     na     Frequency & current
J         val/P   Variable: j=10,sj-> run 10 steps. J=P sets J to current position
J>x?J=y           Set J to y if J > x. Also J<x?J=y, J>x?K=y, J<x?K=y
J>x?Stop          Stop repeat & set J=0. Also J<x?Stop
K                 Same as J
L(ed)     0..200  Set/query sensor current (x 0.1mA(
LS                Optimize sensor current
P90               Phase 90 dgr: Run mode
P0/P180           Static mode with phase 0/180 dgr. Use P90 to return to run mode
P         R/N/A   Resonance sweep: P, PR (reverse), PA (alternating)
Q/QS      +/-     Minimum messages. Q- turns info back on. QS shows string on SPI
R/RD/RL   adr     Reads EEPROM byte/integer/long from address 'adr' (0..255)
RS                RECOVERY: Restore Mem0-M38 from Mem216-mem255
RB                BACKUP: Store Mem0-M38 in Mem216-mem255
STB       +/-     Toggle motor on/off. Use - for CCW
S/SN      +/-step Steps (max +/-32768). SN: no resonance tracking
SW        +/-step Steps (max +/-32768) using sweep pulses
U/UP/UN    +/-val Micropulses. Use - for CCW. P/N to stop at p/n edge
US/UPS/UNS +/-val Same with sweep pulses
V(oltage) 0..5000 Motor voltage (mV)
Vmin      0..5000 Vmin: Min voltage to turn motor
Vmax      0..5000 Vmax: Max voltage
W/WD      val@adr Writes byte/int in address 'adr' (0..255). Note '@'
X/XI      val     Repeat 'val' times. XI => forever. Stop with <CR>
Z/ZI      +/-pos  Set/query Position or Index offset
H(elp)    na      This text

MEMORY: Read 2-byte positions with RD command, i.e. RD4:

0:     RX-TX timeout at power up (x 0.2s)
4+5:   Sweep pulse frequency step
6+7:   Sweep pulse frequency step duration (x 30us)
8:     Resonance range low (kHz)
9:     Resonance range high (kHz) (High>Low)
10:    Continue after stop (ms)
11:    Debounce multiplier (x 30us, std: 1)
12+13: Max step time (ms) (std 250)
```

```
14+15: Tracking stability delay (std:250)
16+17: Vs, motor voltage 0..5000mV. V-command to set/query
20:    Current calibration (std:30)
21:    Step size in sweep (x 10Hz)
22:    Step size in tracking (Hz)
26+27  Position
31     Vmin (x 100mV)
32     Vmax (x 100mV)
33     Sensor current (x 0.1mA)
34+35  Oscillator from sweep/tracking or with F-command
36     CW percent scaling of Vs (0..255) in S/G-commands
37     Same for CCW

STOP commands D,S,G & U: Type <CR>

NOTE !!! MAX RATED MOTOR CURRENT: Diameter x 16mA
```

The commands are explained in more detail below.

The driver has 256 Bytes of EEPROM memory which is assigned as follows:

```
mem0-mem21 for system memory,
mem22-mem32 Motor1 individual parameters
mem33-mem43 Motor2 individual parameters
mem44-mem54 ditto Motor3
mem55-mem65 ditto Motor4
mem66-mem76 ditto Motor5
.
.
.
Mem132-mem255 are free and available to the user with the R/W commands
```

It is not necessary to calculate the location of the parameters for each motor since the command will automatically select the right memory position once the motor is selected.

Note that is possible by mistake to overwrite the memory with serious consequences for the driver operation so a backup listing is recommended to be able to restore the contents of the EEPROM.

Command Interpreter:

When the LED is blinking on/off in a 1 second cycle, the driver is waiting for the next command from the user. This is indicated on the screen with '>>'.

When the driver receives a command line terminated by a <CR>, the driver starts executing the individual commands separated by ',' or ';' until all commands have been executed:

```
S720,d1000,s-720  => step 720 - delay 1000ms - step -720
```

Using a motor with a 1440 step codewheel the result of this line will be half a turn clockwise, wait a second and turn back.

It is possible to repeat part of the command line up to the next semicolon or end of the line by using the 'x' command

```
s100,d1000,s-100,x10;h
```

This command line will repeat the movement 10 times and then list the help text.

If a <CR> is received from the COM port the x-repeat will be stopped.

Commands:

ADC>x?S Stop repeats if (last measured mV/mA) > x. Also ADC<x?S

This will stop command line execution when the last measurement is above/below the limit given. Here is an example with microstepping ('u' command) until a given value from the position sensor ('ap' command) is reached:

```
>>u3,ap,ADC>2700?S,x100  => 3 microsteps, measure position sensor output, stop if >2700
```

Output from driver:

```
Position sensor: 1905 mV Value: 1
Position sensor: 2332 mV Value: 1
Position sensor: 2643 mV Value: 1
Position sensor: 2812 mV Value: 1
>>
```

ADC>x?J=ADC Set J (or K) = (last measured mV/mA) if > x. Also ADC<x?J=ADC

Transfer the measured value to one of the command line variables, J or K, which in turn could be stored in EEPROM for later evaluation.

A1,A2,A3 Analog values for Position and Index sensors

Not relevant in this application which uses a digital AEDR8500 sensor. Use the IS-command to see current logical values for the sensors.

BCR BCR: Break for <CR>

As default most commands can be interrupted from the keyboard by typing a <CR>. However, if this has been switched off by the 'b-' command it can be switched back on with this command.

BI/B+ BI: Search for Index, B+: Index break on

The motor codewheel has an index mark to set the "home" position within a revolution. BI will turn the motor on to search for the motor index position and then stop.

B+ just turns on the index sensor and let the user run the motor until the index is found. After the motor has stopped at the index, the user must turn off the index sensor with 'b-' and set the normal break on with 'bcr'. This is automatic with 'bi'.

B- B-: Break OFF

See above.

CW/CCW 0..255 Motor current scaling (percent)

Due to differences in loading and in the motor CW and CCW (clockwise and counter clockwise) allows the user to modify the motor current (Is) in the two directions. Default is 100 for both CW and CCW. By typing 'cw' (or 'ccw') without a value the driver will read out the current setting. The value can be 0..255:

```
>>cw

CW : 100

>>cw150

CW : 150

>>
```

D(elay) val Delay (msec)

The 'd' command will wait 0..999999 ms as set by the value given. Typing <CR> will abort the delay.

G/GN +/-pos Goto position. GN: No resonance tracking

Go to a specified absolute position within a ±30000 range. The 'z' command can set the zero point. Use the 's' command for steps relative to the current position:

```
>>z,g200,s50,g-200

Position set to 0 for motor 1
Start position 0 for Motor 1            => Go absolute to position 200
End position 200 for Motor 1
Motor current (mA): 452
Steps= 200
Steptime(us)= 3840
Start position 200 for Motor 1          => Move 50 steps relative to current position
End position 250 for Motor 1
Motor current (mA): 460
Steps= 50
Steptime(us)= 3904
Start position 250 for Motor 1          => Go absolute to position -200
End position -200 for Motor 1
Motor current (mA): 474
Steps= 450
Steptime(us)= 3840
```

The qualifier N as in 'GN' or 'SN' switches off the tracking software and keeps the oscillator frequency fixed. This can be useful for making various specialized tests e.g. the motor performance in the Power Consumption chart in the beginning of this text. Here the oscillator value ('o' command) is controlled thru the J-variable which is increased in steps of 1 for each repeat of the line from the semicolon to the end:

```
j=60;oj,sn200,i,j+1,x30

Oscillator: 60
Start position 30806 for Motor 1
End position 31006 for Motor 1
Motor current (mA): 105
Steps= 200
Steptime(us)= 14272
Osc,Hz,mA: 60,48488,  0
J: 61
29 repeats left
Oscillator: 61
Start position 31006 for Motor 1
End position 31206 for Motor 1
Motor current (mA): 107
Steps= 200
Steptime(us)= 13632
Osc,Hz,mA: 61,48392,  0
```

```
J: 62
.
.
.
```

Normally all the info from the command execution is not necessary and the 'q' command can be used to minimize reporting and speed up execution

I(nfo) Frequency & current

Reports oscillator setting, frequency and motor current. If the motor is not running the current is zero so it is necessary to turn the motor on first to get a valid reading of the current:

```
>>stb,d3000,i,stb-                    => Turn on the motor, wait 3 s, info readout, stop

Motor 1 ON
Waiting 3000 milliseconds
Osc,Hz,mA: 68,47672,142
>>
```

J val/P Variable: j=10,sj-> run 10 steps. J=P sets J equal to current position
J>x?J=y Set J to y if J > x. Also J<x?J=y
J>x?S Stop repeat & set J=0. Also J<x?S
K Same as J

J and K are numerical variables with a value that gets inserted everywhere J/K occurs in the command line. If the value changes the new value will be used – see the example under the 'g' command.

L 0..255 Set/query sensor LED drive
LS Optimize LED value

The driver has two comparators to convert analog position and index sensors to hi/lo values. Only with analog sensors (200 cpr).

M 1..10 Motor selection.

Selects the motor and stores the value in mem0 i.e. 'r1' will give the current motor number.

```
m1,s10,m0,d2000,x3

Start position 32608 for Motor 1
End position 32618 for Motor 1
Motor current (mA): 99
Steps= 10
Steptime(us)= 8768
Waiting 2000 milliseconds
2 repeats left
Start position 32618 for Motor 1
End position 32628 for Motor 1
Motor current (mA): 165
Steps= 10
Steptime(us)= 6848
Waiting 2000 milliseconds
1 repeats left
Start position 32628 for Motor 1
End position 32638 for Motor 1
Motor current (mA): 169
```

```
Steps= 10
Steptime(us)= 6656
Waiting 2000 milliseconds
```

O(sc)0..255 Oscillator setting, autoset with P command

Individual setting of the frequency for each motor. Once the motor is running the start frequency is determined by the O-value but the driver will continuously adjust the frequency to maximize current for the motor. When the motor is turned off the new O-value is stored in EEPROM.

P R/N/A Resonance sweep: P, PR (reverse), PA (alternating)

Resonance sweep of the oscillator, starting by the value in mem8 ending with the mem9 value:

```
>>p

Calibration sweep with motor 1
Osc, mA:
140, 70 ******************
141, 63 ****************
142, 63 ****************
143, 59 ***************
144, 59 ***************
145, 56 ***************
146, 56 ***************
147, 59 ***************
148, 59 ***************
149, 63 ****************
150, 73 ******************
151, 84 ***********************
152, 94 *************************
153,112 ******************************
154,133 ***********************************
155,154 ****************************************
156,178 **********************************************
157,199 ***************************************************
158,224 ************************************************************
159,255 ******************************************************************
160,280 ************************************************************************
161,294 ****************************************************************************
162,297 *****************************************************************************
163,287 ****************************************************************************
164,273 ************************************************************************
165,248 ******************************************************************
166,224 ************************************************************
167,196 ****************************************************
168,171 *********************************************
169,147 ***************************************
170,133 ***********************************
171,119 *******************************
172,105 ***************************
173, 94 *************************
174, 87 ***********************
175, 77 *********************
176, 70 ******************
177, 59 ***************
178, 52 **************
179, 45 ************
180, 42 ***********
Motor resonance: 45608 Hz, o: 162, max current: 297, voltage: 1500
```

Q +/- Minimum messages. Q- turns info back on

As default the driver reports a lot of statistics on the motor, but this can slow down execution due to the limited communication speed. The ‘q’-command turns of most of text.

R/RD/RL adr Reads EEPROM byte/integer/long from address 'adr' (0..255)

Used for reading EEPROM values. The memory map tells which parameters are using two Bytes.

STB +/- Toggle motor on/off. Use - for CCW

Debug start/stop of the motor without sensor and tracking.

S/SN +/-step Steps (max +/-32768). SN: no resonance tracking

Se the explanation for the 'G'-command.

U/UP/UN +/-val Micropulses. Use - for CCW. P/N to stop at p/n edge

Pulsed mode with ON period set by mem4+5 followed by an OFF period (mem6+7) repeated as many times as given by the value of val. Typically a pulse frequency of 1kHz can be achieved which means this operating mode will generate audible noise. The oscillator frequency is kept constant in this mode i.e. no tracking.

US/UPS/UPN +/-val Same using sweep pulses

Pulsed mode but with the driver frequency being swept while the motor is on. Contact factory before this mode is used.

V 0..5000 Motor voltage, Vs(mV)
Vmin,Vmax 0..5000 Vmin: Min voltage to turn motor, Vmax: Max voltage limit

Set the DC voltage for the output stage. Vmax is used to accidentally overloading the motor with too high a current - approx 500mA for an Ø30 motor.

W/WD val@adr Writes byte/integer in address 'adr' (0..255). Note '@'

Write a value to a specified EEPROM byte/integer.

```
wd40@4
Writing 40 to address 4(+1)
```

X val Repeat the command line 'val' times.

Specifies the number of repeats of a command line from a semicolon to the next semicolon

Z/ZI +/-pos Set/query Position or Index offset

Set or query the zero position and the offset for the index sensor.

H(elp)

The built-in driver help text

- O -