Firmware /PWMIO18

The /PWMIO18 firmware works PC interface with 18 digital I/O lines with adjustable data direction (18 GPIO). All I/O lines can generate a 8 bit / 50 Hz PWM signal, as well as blink an pulse signals. Port D can be used to control servo drives. It generates a 50 Hz-PWM-signal with a pulse duration of 1..2 ms for that purpose.
Servo and PWM outputs are controlled by software, by ADC inputs or by an internal ramp generator. Ramps and pulses can be triggered with Port B signals.
I/O lines are grouped to six ports (Port B, Port C, Port D). Each I/O line can be configured for alternative purposes:

Port B0…Port B5	Digital I/O, PWM, Blink/Pulse output, 6x Trigger input
Port C0…Port C5	Digital I/O, PWM, Blink/Pulse output, 6x ADC
Port D2…Port D7	Digital I/O, PWM, Blink/Pulse output, 6x Servo

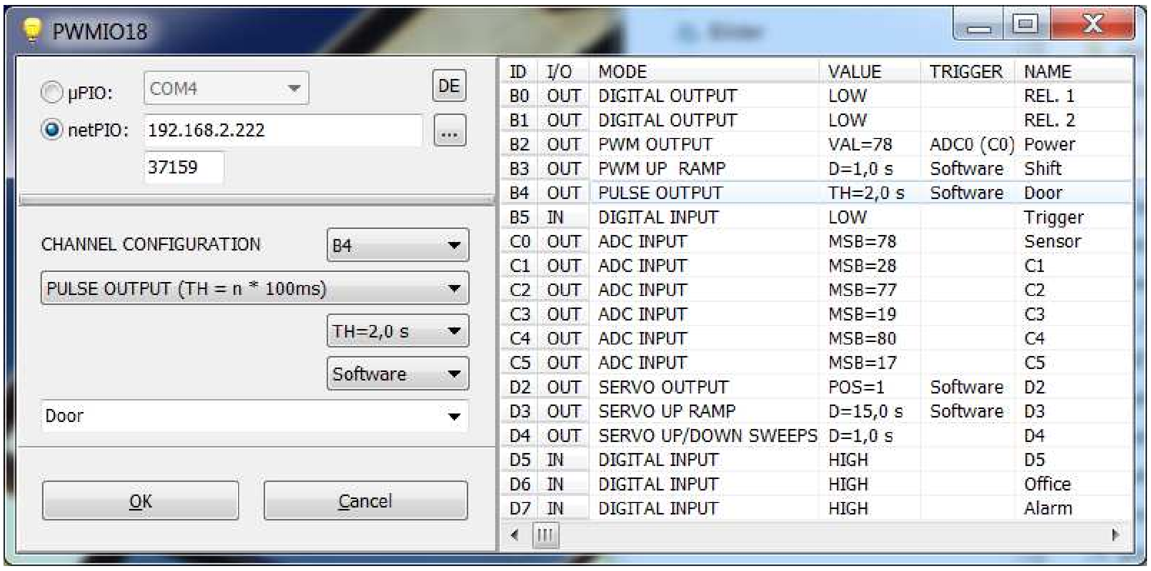
In addition Port C voltages can be read with a 10 bit ADC, using three different reference options:

External reference voltage: External Vref is supplied through VREF screw terminal.
0 < Vref < Vcc must not be overdriven.

Internal reference VCC: Positive 5V supply voltage is used as reference.

Internal reference 1.1 V: An internal reference voltage of 1.1 V is used.

Input channels can be configured to be "pulled high" by an internal pull-up resistor, to have a defined high level that could be pulled low by switch contacts.
The module powers up with an individual configuration storable in the modules EEPROM, so it is useful for some small stand-alone control applications as well.
User- defined friendly names (17 chars / channel) can be assigned to each channel.



192.168.2.222
37159
B4
PULSE OUTPUT (TH = n * 100ms)
TH=2,0 s
Software
Door

ID	I/O	MODE	VALUE	TRIGGER	NAME
B0	OUT	DIGITAL OUTPUT	LOW		REL. 1
B1	OUT	DIGITAL OUTPUT	LOW		REL. 2
B2	OUT	PWM OUTPUT	VAL=78	ADC0 (C0)	Power
B3	OUT	PWM UP RAMP	D=1,0 s	Software	Shift
B4	OUT	PULSE OUTPUT	TH=2,0 s	Software	Door
B5	IN	DIGITAL INPUT	LOW		Trigger
C0	OUT	ADC INPUT	MSB=78		Sensor
C1	OUT	ADC INPUT	MSB=28		C1
C2	OUT	ADC INPUT	MSB=77		C2
C3	OUT	ADC INPUT	MSB=19		C3
C4	OUT	ADC INPUT	MSB=80		C4
C5	OUT	ADC INPUT	MSB=17		C5
D2	OUT	SERVO OUTPUT	POS=1	Software	D2
D3	OUT	SERVO UP RAMP	D=15,0 s	Software	D3
D4	OUT	SERVO UP/DOWN SWEEPS	D=1,0 s		D4
D5	IN	DIGITAL INPUT	HIGH		D5
D6	IN	DIGITAL INPUT	HIGH		Office
D7	IN	DIGITAL INPUT	HIGH		Alarm

A configuration software helps you to setup the desired configuration.

Data protocol
USB-PC interface uses virtual COM (VCP) driver.

Serial (USB) interface parameters are: 57600, 8 N, 1

Baud 57600
8 Data bits
No parity
1 Stop bit

Request / response communication is being used. A request is 20 bytes long in any case. A response with same length and format will be returned on any request.
The sample rate is about 10 samples/sec..

Request / Response: 20 bytes
The first of the 20 request bytes defines a command to be executed. The second request byte addresses a channel and is followed by 18 parameter bytes,

/ Byte1 = command // Byte 2 = channel // Byte 3..20 = parameters /

Each command triggers a response with same data structure.

Command (Byte 1):

The following commands are defined:

Mode Read/Write
```
    cmd_WriteChannelModes = 'D';
    cmd_ReadChannelModes = 'E';
```

Status Read/Write
```
    cmd_WriteChannelStats = 'S';
    cmd_ReadChannelStats = 'T';
```

Trigger/Source select
```
    cmd_WriteChannelSources = 'F';
    cmd_ReadChannelSources = 'G';
```

Namen Read/Write
```
    cmd_WriteChannelname = 'N';
    cmd_ReadChannelname = 'O';
```

Reference voltage and ADC
```
    cmd_WriteVref = 'V';
    cmd_ReadVref = 'W';
    cmd_ReadADCs = 'A';
```

Memory functions
```
    cmd_ConfigFromEeprom = 'X';
    cmd_ConfigToEeprom = 'Y';
```

Channel number (Byte 2):
The channel number is located in Byte 2 of the request.
The channel assignment is as follows (decimal):

ch_ALL = 0; (All channels addressed, if supported by command)

ch_B0 = 1;
ch_B1 = 2;
ch_B2 = 3;
ch_B3 = 4;
ch_B4 = 5;
ch_B5 = 6;

ch_C0 = 7;
ch_C1 = 8;
ch_C2 = 9;
ch_C3 = 10;
ch_C4 = 11;
ch_C5 = 12;

ch_D2 = 13;
ch_D3 = 14;
ch_D4 = 15;
ch_D5 = 16;
ch_D6 = 17;
ch_D7 = 18;

Parameters (Bytes 3..20)
The 18 parameter bytes are assigned to the channels B0 to D7 one after each other.

Examples:
The first parameter byte (P1 = Byte 3 of requests) is assigned to channel B0.
The last parameter byte (P18 = Byte 20 of requests) is assigned to channel D7.
The 8th parameter byte (P8 = Byte 10 of request) is assigned to channel C1.

```
// 1 /    2   / 3/ 4/ 5/ 6/ 7/ 8/ 9/10/11/ 12/ 13/ 14/ 15/ 16/ 17/ 18/ 19/ 20//
//CMD/CHANNEL/P1/P2/P3/P4/P5/P6/P7/P8/P9/P10/P11/P12/P13/P14/P15/P16/P17/P18//
//CMD/CHANNEL/B0/B1/B2/B3/B4/B5/C0/C1/C2/ C3/ C4/ C5/ D2/ D3/ D4/ D5/ D6/ D7//
```

cmd_WriteChannelModes = 'D'
This command selects the desired channel mode (input, output, blink, etc.)
All 18 channels could be set in one go, or one channel can be selected individually.
(see above "channel number")

The byte values of the 18 parameter bytes (bytes 3..20 of request) specify the channel mode for each channel. Mode for channel B0 is located in first parameter byte (Byte 3 of request), while last parameter byte refers to channel D7. If a certain channel is addressed parameter bytes for other channels are ignored.

The module will echo the request as soon it received it.

The following modes are selectable with the parameter bytes:

Pm_disabled = 0
The channel is deactivated (high impedance) .

Pm_adc_input = 1
Data direction is set to input for use as ADC input channel. This mode can be used on Port C channels only!

Pm_open_input = 2
Data direction is set to input. Channel work as 5V TTL logic input.
(level floating if left open).

Pm_pullup_input = 3
Data direction is set to input an channel is pulled high with internal pullup resistor.
Input may be pulled down to ground with switch contacts or similar.

Pm_output = 4
Data direction is set to output. Output acts as general purpose logic output with 5V high level.

Pm_pwm = 5
Channel mode is set to PWM output.

Pm_servo = 6
Channel mode is set to servo output.
Only possible on Port D channels!

Pm_blink100ms = 7
Channel mode is programmed to output blink/clock signals, with intervals ranging from
1 * 100ms (0,1 sec.) to 255 * 100 ms (25,5 sec.).

Pm_blink1s = 8
Channel mode is programmed to output blink/clock signals, with intervals ranging from
1 * 1s (1 sec.) to 255 * 1s (4min. 15sec.).

Pm_pulse100ms = 9
Channel mode is programmed to output single pulses, ranging from
1 * 100ms (0,1 sec.) to 255 * 100 ms (25,5 sec.).
(Triggered by software or Port B channels)

Pm_pulse1s = 10
Channel mode is programmed to output single pulses, ranging from
1 * 1s (1 sec.) to 255 * 1s (4min. 15sec.).
(Triggerable by software or Port B channels)

Pm_pwmRampUp = 11
Output mode is set to PWM. The PWM gets modulated by a (single shot) ramp (0...100%).
(Triggered by software or Port B channels)

Pm_pwmSweepUp = 12
Output mode is set to PWM. The PWM gets modulated by a (continious) ramp (0...100%).

Pm_ServoRampUp = 13
Only available on Port D!
Output mode is set to servo. The servo position gets shifted by a (single shot) ramp (0...100%).
(Triggered by software or Port B channels)

Pm_ServoSweepUp = 14
Only available on Port D!
Output mode is set to servo. The servo position gets shifted by a (continuous) ramp (0...100%).
(Triggered by software or Port B channels)

Pm_pwmRampDown = 15
Pm_pwmSweepDown = 16
Pm_servoRampDown = 17 Only available on Port D!
Pm_servoSweepDown = 18 Only available on Port D!
Same modes as 11,12,13,14, but opposite ramp direction (down).

Pm_pwmRampUpDown = 19
Pm_pwmSweepUpDown = 20
Pm_servoRampUpDown = 21 Only available on Port D!
Pm_servoSweepUpDown= 22 Only available on Port D!
Same as above,but with alternating ramp directions (up/down).

cmd_ReadChannelModes = 'E'
This command request channel modes being set for all channels. Channel no and parameters are ignored.
Modes are returned as described under **cmd_WriteChannelModes.**

cmd_WriteChannelStats = 'S'
This command is used to change channel conditions. Depending on the channel mode the parameter bytes will affect individual channel conditions, like output state, blink interval, etc.
All 18 channels could be set in one go, or one channel can be selected individually.
(see above "channel number") A response with conditions of all channels will be returned

Parameter meanings depend on channel modes as follows:

Pm_output:

Parameter byte = 0	=> Output goes LOW
Parameter byte <> 0	=> Output goes HIGH

Pm_blink100ms,
Pm_pulse100ms:

Parameter byte = 0	=> Output goes LOW
Parameter byte = 1 ...255	=> Sets the duration (Parameter * 100ms)

Pm_blink1s,
Pm_pulse1s:

Parameter byte = 0	=> Output goes LOW
Parameter byte = 1 ...255	=> Sets the duration (Parameter * 1s)

Pm_pwm:

Parameter byte = 0 ...255	=> Sets the PWM amount (0..100%)

Pm_servo:

Parameter byte = 0 ...255	=> Adjusts the servo position

Pm_xxRampxx,
Pm_xxSweepxx:

Parameter byte = 0	=> Ramp off
Parameter byte = 1 ...255	=> Adjusts ramp duration (1...255 s)

Pulses and ramps are triggered by software writing the duration value, but just in case a certain channel is addressed.

cmd_ReadChannelStats = 'T'
Reads condition parameters of all channels. Channel and parameter bytes are ignored.
The response contains the channel conditions as described for **cmd_WriteChannelStats.** Conditions of input channels are reported as well:

Pm_adc_input :

Parameter byte	=>	Byte value read by ADC (MSB)

Pm_open_input ,
Pm_pullup_input = 3

Parameter byte = 0	=>	Input is LOW
Parameter byte <> 0	=>	Input is HIGH

cmd_WriteChannelSources = 'F'
Some modes can be controlled by other channels instead of controlling these by software.
This is done by assigning a source channel to the channel being controlled, using this command.

Pm_pwm, Pm_servo:
PWM amounts and servo positions can be controlled by a ADC channel (available on Port C). The control source for a channel is selected by one of the following values, to be placed in the parameter bytes.

```
src_sw    = 0; // software controlled
src_adc0 = 1; // controlled by ADC channel
src_adc1 = 2;
src_adc2 = 3;
src_adc3 = 4;
src_adc4 = 5;
src_adc5 = 6;
```

Pm_pulse100ms, Pm_pulse1s,
Pm_pwmrampup, Pm_servorampup,
Pm_pwmrampdown,Pm_servorampdown,
Pm_pwmrampupdown, Pm_servorampupdown,
Pulses and ramps can be triggered by port B channels (falling edge). The trigger source is selected by the following values of the parameter bytes.

```
src_sw = 0; // Software trigger
src_B0 = 1; // triggered by falling edge on port B
src_B1 = 2;
src_B2 = 3;
src_B3 = 4;
src_B4 = 5;
src_B5 = 6;
```

cmd_ReadChannelSources = 'G'
Some modes can be controlled by other channels instead of controlling these by software.
This commands reads the current control source settings of all channels.
(refer to cmd_WriteChannelSource)

cmd_WriteVref = 'V'

Thos command selcects one of three possible voltage reference options. Reference voltage is common for all ADC channels.

VrefExt	= 0; // supply external reference voltage
VrefVcc	= 1; // reference voltage = 5V (internal supply voltage)
Vref1V1	= 3; // reference voltage = 1,1V (internal)

This command does not need any channel number, so the option byte replaces the channel byte in the request.

cmd_ReadVref = 'W'

Reads the selected reference voltage option from the board.
(see cmd_WriteVref)

cmd_ReadADCs = 'A'

This commands reads the ADC raw values from Port C, whatever channel mode is set for these channels. It does not change the data direction or channel mode in any way.
The leading twelve parameter bytes of the response contain the ADC raw values (words) of the six ADC channels. The words bits are left-aligned, so bit 0..5 are not valid, while bits 6...15 represent the 10 bit ADC result.

Pairs of two bytes are combined to one word value.

First parameter byte is LByte of ADC0.
2nd parameter byte is HByte of ADC0.
3rd parameter byte is LByte von ADC1.
...
12th parameter byte is HByte of ADC5.

cmd_WriteChannelName = 'N'

Command assigns a friendly name to a channel. The channel name is a null-terminated ASCII string, the must be placed in the parameter bytes. 17 character + CHR(0) = 18 bytes.
Unused characters at the end should be filled with CHR(0).
The response echoes the request.

cmd_ReadChannelName = 'O'

Reads the channel name of a channel. A certain channel no must be specified. Parameter bytes are ignored. The response contains a null terminated ASCII string in the parameter bytes.

cmd_ConfigToEeprom = 'Y'
All configuration settings (modes, states, names, etc.) that are made with above commands are temporary stored in the modules RAM memory. For permanent storage current settings can be transferred to the modules EEPROM memory. After power-up the module recalls the settings stored permanently in the EEPROM.
The response echoes the request.

cmd_ConfigFromEeprom = 'X'
Recalls the start-up settings stored in the devices EEPROM memory (see ***cmd_ConfigToEeprom***).
The response echoes the request.

Appendix

Commands (Byte 1)

```
cmd_Writechannelname = 'N';
cmd_Readchannelname = 'O';
cmd_Writechannelmodes = 'D';
cmd_Readchannelmodes = 'E';
cmd_Writechannelstats = 'S';
cmd_Readchannelstats = 'T';
cmd_WritechannelSources = 'F';
cmd_ReadchannelSources = 'G';
cmd_Writevref = 'V';
cmd_Readvref = 'W';
cmd_Readadcs = 'A';
cmd_Configfromeeprom = 'X';
cmd_Configtoeeprom = 'Y';
```

Channel address (Byte 2)

```
ch_ALL = 0;
ch_B0 = 1;
ch_B1 = 2;
ch_B2 = 3;
ch_B3 = 4;
ch_B4 = 5;
ch_B5 = 6;
ch_C0 = 7;
ch_C1 = 8;
ch_C2 = 9;
ch_C3 = 10;
ch_C4 = 11;
ch_C5 = 12;
ch_D2 = 13;
ch_D3 = 14;
ch_D4 = 15;
ch_D5 = 16;
ch_D6 = 17;
ch_D7 = 18;
```

ADC source

```
src_sw   = 0;
src_adc0 = 1;
src_adc1 = 2;
src_adc2 = 3;
src_adc3 = 4;
src_adc4 = 5;
src_adc5 = 6;
```

Trigger source

```
    src_sw = 0;
    src_B0 = 1;
    src_B1 = 2;
    src_B2 = 3;
    src_B3 = 4;
    src_B4 = 5;
    src_B5 = 6;
```

USB serial protocol structure

```
RequestSize = 20;
TRequest =  packed record
                Command: AnsiChar;
                ChannelNo: Byte;
                Params: array[0..17] of Byte;
            end;

ResponseSize = 20;
TResponse = TRequest;
```