Firmware /PWMIO18

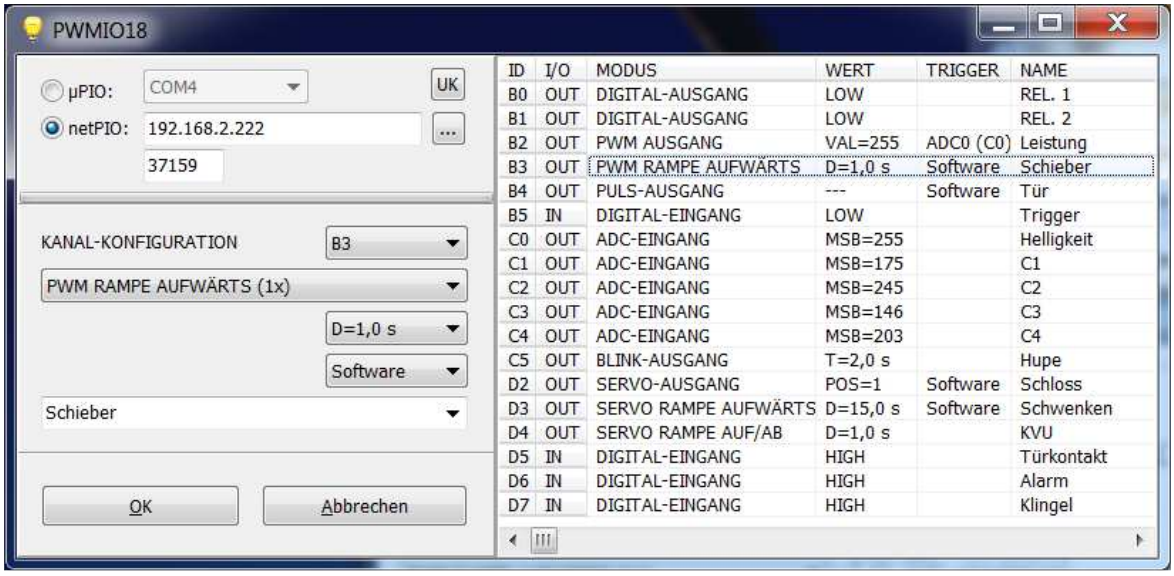
Die /PWMIO18-Firmware arbeitet als PC-Interface mit 18 digitalen I/O-Kanälen deren Datenrichtung umschaltbar ist (18 GPIO). Alle I/O-Leitungen erlauben alternativ die Ausgabe eines 8-Bit / 50Hz-PWM-Signals, sowie eines Blink- oder Puls-Signals. Port D kann für die Ansteuerung von (Modellbau-)Servos verwendet werden und gibt bei Bedarf ein 50Hz-PWM-Signal mit einer Pulsdauer von 1..2 ms aus. Servo- und PWM-Ausgänge können per Software, per ADC-Eingangsspannung oder von einem internen Rampengenerator gesteuert werden. Rampen und Puls-Ausgänge lassen sich durch Triggersignale am Port B auslösen. Die I/O-Leitungen sind zu drei Ports zusammengefasst (Port B, Port C, Port D).
Jeder Port besitzt sechs I/O-Leitungen, die alternativ folgende Funktionen übernehmen können:

Port B0…Port B5	Digital-I/O, PWM, Blink/Puls-Ausgang, 6x Triggereingang
Port C0…Port C5	Digital-I/O, PWM, Blink/Puls-Ausgang, 6x ADC
Port D2…Port D7	Digital-I/O, PWM, Blink/Puls-Ausgang, 6x Servo

Die sechs Spannungen am Port C können darüber hinaus mit dem internen 10-Bit-A/D-Wandler (ADC) eingelesen werden. Die Referenzspannung für den ADC kann dabei aus drei umschaltbaren Quellen bezogen werden:

Externe Referenzspannung: Die Referenzspannung wird über den Anschluss VREF eingespeist. 0<Vref<Vcc ist dabei einzuhalten.
Interne Referenz VCC: Als Referenzspannung dient die +5V Versorgungsspannung.
Interne Referenz 1.1 V: Eine Referenzspannung von 1.1V wird intern erzeugt.

Für Kanäle die als Eingang arbeiten kann per Software ein interner Pullup-Widerstand aktiviert werden, der den jeweiligen Eingang auf das Potential der positiven Versorgungsspannung (Vcc) legt, um z.B. offenen Eingänge ein definiertes HIGH-Potential zu geben, welches von Schaltkontakten nach Masse gezogen werden kann. Mit dem Anlegen der Versorgungsspannung stellt die Firmware eine individuelle, auf dem Modul speicherbare Kanalkonfiguration selbstständig her, so dass das Modul auch ohne PC kleine Steuerungsaufgaben übernehmen kann. Zusammen mit der Konfiguration speichert das Modul für jeden Kanal einen benutzerdefinierbaren Kanalnamen (max. 17 Zeichen/pro Kanal).



Für die Kanalkonfiguration steht ein Software-Tool zur Verfügung, mit man die Grundkonfiguration des Moduls vornehmen kann.

Datenprotokoll
Die Kommunikation mit dem PC erfolgt per USB. Der USB-Treiber installiert einen virtuellen COM-Port. Der Datenaustausch erfolgt somit wie über eine ‚echte' RS232-Schnittstelle.

Die Schnittstellenparameter sind: 57600, 8 N, 1

Baud 57600
8 Datenbits
Keine Parität
1 Stoppbit

Die Kommunikation erfolgt nach dem Frage-Antwort-Prinzip (Request/Response). Dabei besteht ein Request immer aus genau 20 Datenbytes. Nachdem Empfang dieser 20 Datenbytes sendet die Firmware ebenfalls 20 Bytes als Antwort an den PC zurück.
Die erzielbare Abfragerate (sample rate) liegt bei etwa 10 Samples/Sek.

Request / Response: 20 Bytes

Ein Request besteht immer aus 20 Bytes. Das erste Byte dient als Kommando-Byte, das zweite Byte enthält die Kanalnummer. Es folgen 18 Parameter-Bytes.

/ Byte1 = Kommando // Byte 2 = Kanal // Byte 3..20 = Parameter /

Jedes Kommando wird mit einer Antwort von 20 Bytes vom Modul beantwortet. Die Antwort hat den gleichen Aufbau wie der Request.

Kommando (Byte 1):

Folgende Kommandos sind definiert:

Modus Lesen/Schreiben
    cmd_WriteChannelModes = 'D';
    cmd_ReadChannelModes = 'E';

Status Lesen/Schreiben
    cmd_WriteChannelStats = 'S';
    cmd_ReadChannelStats = 'T';

Trigger/Source wählen
    cmd_WriteChannelSources = 'F';
    cmd_ReadChannelSources = 'G';

Namen Lesen/Schreiben
    cmd_WriteChannelname = 'N';
    cmd_ReadChannelname = 'O';

Referenzspannung unf ADC
    cmd_WriteVref = 'V';
    cmd_ReadVref = 'W';
    cmd_ReadADCs = 'A';

Speicherfunktionen
    cmd_ConfigFromEeprom = 'X';
    cmd_ConfigToEeprom = 'Y';

Kanalnummer (Byte 2):

Die Kanalnummer wird in **Byte 2** des Request abgelegt.
Es gilt folgende Kanalzuordnung (dezimal):

ch_ALL = 0; (Alle Kanäle ändern, sofern vom Kommando unterstützt)

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;

Parameter (Byte 3..20)

Die 18 Parameter-Bytes sind den Kanälen B0 bis D7 der Reihe nach zugeordnet.

Beispiele:
Das erste Parameter-Byte (P1 = Byte 3 des Requests) beeinflusst Kanal B0.
Das letzte Parameter-Byte (P18 = Byte 20 des Requests) beeinflusst Kanal D7.
Das achte Parameter-Byte (P8 = Byte 10 des Requests) beeinflusst Kanal 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'
Dieses Kommando stellt den gewünschten Kanal-Modus (Eingang, Ausgang, Blinken, etc) ein.

Wahlweise können die Modi aller 18 Kanäle auf einmal gesetzt, oder nur ein bestimmter Kanal verändert werden. (Siehe Abschnitt "Kanalnummer").

Die Bytewerte der 18 Parameter-Bytes (Byte 3..20 des Requests) bestimmen jeweils den Modus eines Kanals. Der Modus für Kanal B0 wird also z.B. im ersten Parameter-Byte (Byte 3 des Requests) abgelegt, der Modus für Kanal D7 im letzten Parameter-Byte (= Byte 20 des Requests). Wird ein bestimmter Kanal adressiert, werden die Parameter-Bytes aller anderen Kanäle ignoriert.

Nach dem Empfang des Kommandos sendet das Modul den Request unverändert als Response zurück.

Folgende Modi können durch die Parameterbytes eingestellt werden.

Pm_disabled = 0
Der Kanal wird deaktiviert und hochohmig geschaltet.

Pm_adc_input = 1
Kanal wird für die Verwendung des ADC (Spannungseingang) auf Eingang programmiert. Dieser Modus ist nur an Port C möglich!

Pm_open_input = 2
Kanal wird für die Verwendung als Digital-Eingang (5V-Logik) auf Eingang programmiert. (Ohne externe Beschaltung ist der Zustand des hochohmigen Eingangs nicht definiert).

Pm_pullup_input = 3
Kanal wird für die Verwendung als Digital-Eingang auf Eingang programmiert und über einen interner Pullup-Widerstand auf HIGH-Potential gelegt, so dass dieser extern (z.B. durch einen Schaltkontakt) nach Masse gezogen werden kann.

Pm_output = 4
Kanal wird für die Verwendung als Digital-Ausgang (5V-Logik) auf Ausgang programmiert.

Pm_pwm = 5
Kanal wird für die Verwendung als PWM-Ausgang (5V-Logik) auf Ausgang programmiert.

Pm_servo = 6
Kanal wird für die Verwendung als Servo-Ausgang (5V-Logik) auf Ausgang programmiert. Dieser Modus ist nur an Port D möglich!

Pm_blink100ms = 7
Kanal wird für die Verwendung als Blink-(Takt)-Ausgang programmiert. Dieser Modus ermöglicht ganzzahlige Intervalle von 1 * 100ms (0,1 Sekunde) bis 255 * 100 ms (25,5 Sekunden).

Pm_blink1s = 8
Kanal wird für die Verwendung als Blink-(Takt)-Ausgang programmiert. Dieser Modus ermöglicht ganzzahlige Intervalle von 1 * 1s (1 Sekunde) bis 255 * 1s (4Min,15Sekunden).

Pm_pulse100ms = 9
Kanal wird für die Verwendung als Puls-Ausgang programmiert. Dieser Modus ermöglicht ganzzahlige Pulsdauern von 1 * 100ms (0,1 Sekunde) bis 255 * 100 ms (25,5 Sekunden). Dieser Modus ermöglicht die Triggerung (Auslösung) per Software oder durch eine fallende Flanke eines Kanals am Port B.

Pm_pulse1s = 10
Kanal wird für die Verwendung als Puls-Ausgang programmiert. Dieser Modus ermöglicht ganzzahlige Pulsdauern von 1 * 1s (1 Sekunde) bis 255 * 1s (4Min,15Sekunden). Dieser Modus ermöglicht die Triggerung (Auslösung) per Software oder durch eine fallende Flanke eines Kanals am Port B.

Pm_pwmRampUp = 11
Kanal wird für die Verwendung als PWM-Ausgang (5V-Logik) auf Ausgang programmiert. Die PWM-Modulation wird (einmalig) von einer ansteigenden Rampenfunktion des internen Rampengenerators moduliert (0...100%).
Dieser Modus ermöglicht die Triggerung (Auslösung) per Software oder durch eine fallende Flanke eines Kanals am Port B.

Pm_pwmSweepUp = 12
Kanal wird für die Verwendung als PWM-Ausgang (5V-Logik) auf Ausgang programmiert. Die PWM-Modulation wird (fortlaufend) von einer ansteigenden Rampenfunktion des internen Rampengenerators moduliert (0...100%).

Pm_ServoRampUp = 13
Kanal wird für die Verwendung als Servo-Ausgang (5V-Logik) auf Ausgang programmiert. Die Servo-Position wird (einmalig) von einer ansteigenden Rampenfunktion des internen Rampengenerators moduliert.
Dieser Modus ermöglicht die Triggerung (Auslösung) per Software oder durch eine fallende Flanke eines Kanals am Port B. Dieser Modus ist nur an Port D möglich!

Pm_ServoSweepUp = 14
Dieser Modus ist nur an Port D möglich!
Kanal wird für die Verwendung als Servo-Ausgang (5V-Logik) auf Ausgang programmiert. Die PWM-Modulation wird (fortlaufend) von einer ansteigenden Rampenfunktion des internen Rampengenerators moduliert (0...100%).

Pm_pwmRampDown = 15
Pm_pwmSweepDown = 16
Pm_servoRampDown = 17 Dieser Modus ist nur an Port D möglich!
Pm_servoSweepDown = 18 Dieser Modus ist nur an Port D möglich!
Diese Modi entsprechen den Modi 11,12,13,14, jedoch wird dabei die Rampenfunktion in abfallender Richtung durchlaufen.

Pm_pwmRampUpDown = 19
Pm_pwmSweepUpDown = 20
Pm_servoRampUpDown = 21 Dieser Modus ist nur an Port D möglich!
Pm_servoSweepUpDown= 22 Dieser Modus ist nur an Port D möglich!
Diese Modi entsprechen den Modi 11,12,13,14, jedoch wird dabei die Rampenfunktion abwechselnd in steigender und fallender Richtung durchlaufen.

cmd_ReadChannelModes = 'E'
Das Kommando ruft den eingestellten Kanal-Modus (Eingang, Ausgang, Blinken, etc)
für alle Kanäle des Moduls gleichzeitig ab. Die Kanalnummer und Parameter-Bytes dieses Requests werden nicht ausgewertet (don´t care). Die Parameter-Byte der Antwort enthalten die Kanal-Modi, wie zuvor unter **cmd_WriteChannelModes**.

cmd_WriteChannelStats = 'S'

Dieses Kommando überträgt Zustandsparameter an die Kanäle. Je nach eingestelltem Kanalmodus beeinflussen die Zustandsparameter das individuelle Kanalverhalten (wie z.B. Blinkintervall, Servoposition, etc.). Wahlweise können alle 18 Kanäle auf einmal gesetzt, oder nur ein bestimmter Kanal verändert werden. (Siehe Abschnitt "Kanalnummer").
Nach dem Empfang des Kommandos sendet das Modul die Zustandsparameter aller Kanäle als Response zurück. Die Aufgabe des Zustandsparameters ist vom eingestellten Modus des Kanals abhängig. Nachfolgende Modi werten das Schreiben von Zustandsparametern aus:

Pm_output:
Parameter-Byte = 0 => Ausgang geht auf LOW-Pegel
Parameter-Byte <> 0 => Ausgang geht auf HIGH-Pegel

Pm_blink100ms,
Pm_pulse100ms:
Parameter-Byte = 0 => Ausgang geht auf LOW-Pegel
Parameter-Byte = 1 ...255 => Stellt die Dauer ein (Parameter * 100ms)

Pm_blink1s,
Pm_pulse1s:
Parameter-Byte = 0 => Ausgang geht auf LOW-Pegel
Parameter-Byte = 1 ...255 => Stellt die Dauer ein (Parameter * 1s)

Pm_pwm:
Parameter-Byte = 0 ...255 => Stellt die PWM ein (0..100%)

Pm_servo:
Parameter-Byte = 0 ...255 => Stellt die Servoposition ein

Pm_xxRampxx,
Pm_xxSweepxx:
Parameter-Byte = 0 => Rampe aus
Parameter-Byte = 1 ...255 => Stellt die Rampendauer ein (1...255 s)

Die Puls bzw. Rampen-Auslösung per Software erfolgt durch das Schreiben der Dauer und nur dann wenn ein Kanal einzeln adressiert wurde.

cmd_ReadChannelStats = 'T'

Liest die Zustandsparameter aller Kanäle. Die Kanalnummer und Parameter-Bytes dieses Requests werden nicht ausgewertet (don´t care). Die Antwort enthält die Kanalzustände wie zuvor unter **cmd_WriteChannelStats** bechrieben. Darüber hinaus werden auch die Zustände von Eingangskanälen gelesen:

Pm_adc_input :
Parameter-Byte => Bytewert des ADC (MSB)

Pm_open_input ,
Pm_pullup_input = 3
Parameter-Byte = 0 => Eingang ist LOW
Parameter-Byte <> 0 => Eingang ist HIGH

cmd_WriteChannelSources = 'F'

Einige Modi können nicht nur per Software gesteuert werden, sondern erlauben alternativ eine Steuerung durch einen anderen, zuweisbaren Kanal (Source). Dieses Kommando wählt die gewünschte Steuerquelle für einen steuerbaren Kanal.

Pm_pwm, Pm_servo:
PWM- und Servokanäle können so konfiguriert werden, dass die PWM-Modulation bzw. Servoposition durch einen (an Port C verfügbaren) ADC-Kanal (d.h. spannungsabhängig) gesteuert wird. Dazu übergibt man in den Parameterbytes dieses Kommandos einen der nachstehenden Werte für den zu steuernden Kanal.

```
src_sw    = 0; // Steuerung per Software
src_adc0 = 1; // Steuerung durch einen ADC-Kanal
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,
Kanäle mit Puls- oder Rampenfunktion können so konfiguriert werden, dass die Auslösung (Triggerung) statt per Software, durch eine fallende Flanke an einem der Kanäle des Port B erfolgt. Dazu übergibt man in den Parameterbytes dieses Kommandos einen der nachstehenden Werte für den zu steuernden Kanal.

```
src_sw = 0; // Auslösung per Software
src_B0 = 1; // Auslösung durch fallende Flanke am Port B
src_B1 = 2;
src_B2 = 3;
src_B3 = 4;
src_B4 = 5;
src_B5 = 6;
```

cmd_ReadChannelSources = 'G'

Einige Modi können nicht nur per Software gesteuert werden, sondern erlauben alternativ eine Steuerung durch einen anderen, zuweisbaren Kanal (Source). Dieses Kommando liest die eingestellte Steuerquelle eines einen steuerbaren Kanals.
(siehe cmd_WriteChannelSource)

cmd_WriteVref = 'V'
Diese Kommando wählt eine von drei möglichen Optionen der Referenzspannung aus, die für die alle ADC-Kanäle des Port C gemeinsam verwendet wird.

```
VrefExt     = 0; // Referenzsspannung extern eingespeisen
VrefVcc     = 1; // Referenzsspannung = 5V Betriebsspannung
Vref1V1     = 3; // Referenzsspannung = 1,1V (wird intern erzeugt)
```

Der Wert für die gewünschte Funktion, wird im Kanalbyte des Request übergeben, da dieses Kommando keine Kanalnummer benötigt.

cmd_ReadVref = 'W'
Dieses Kommando liest die gewählte Option für die Referenzspannung.
(siehe cmd_WriteVref)

cmd_ReadADCs = 'A'
Dieses Kommando liest die ADC-Werte (Port C) unabhängig davon welcher Modus für den Kanal konfiguriert ist. Das Kommando hat keinen Einfluss auf die Datenrichtung oder den Modus der Port C-Kanäle. Die ersten zwölf Parameter-Bytes der Antwort anhalten die RAW-Werte (Words) aller sechs ADC-Kanäle. Die Words sind "left-aligned", d.h. die oberen 10-Bit des Words (MSB) enthalten den 10-Bit-Wandler-Wert, während die unteren sechs Bit (LSB) undefiniert sind. Je zwei Parameter-Bytes bilden paarweise das Datenwort.

Das erste Parameterbyte enthält das LByte von ADC0.
Das zweite Parameterbyte enthält das HByte von ADC0.
Das dritte Parameterbyte enthält das LByte von ADC1.
usw.
...
Das zwölfte Parameterbyte enthält das HByte von ADC5.

cmd_WriteChannelName = 'N'
Dieses Kommando gibt einem Kanal einen benutzerdefinierten Namen.
Der zu übertragene ASCII-String (Kanalname) wird als NULL-terminierter String in den Parameter-Bytes (Byte 3..20) des Request abgelegt. Für jeden Kanal sind somit 17 ASCII-Zeichen nutzbar, die mit einem CHR(0) abgeschlossenen sein müssen. Nicht verwendete Zeichen am Ende sollten mit CHR(0) aufgefüllt werden.

Nach dem Empfang des Kommandos sendet das Modul den Request unverändert als Response zurück.

cmd_ReadChannelName = 'O'
Liest den Kanalnamen eines Kanals. Die Kanalzuordnung erfolgt wie zuvor beschrieben mit Byte 2. Die Parameterbytes (3..20) des Request werden nicht ausgewertet (don´t care bytes).

Die Antwort enthält den angeforderten Kanalname in den Parameter-Bytes (3..20) als null-terminierten ASCII-String.

cmd_ConfigToEeprom = 'Y'
Alle Konfigurationsdaten, wie Modus, Status und Name der Kanäle, usw., die mit den zuvor beschrieben Kommandos zum Modul übertragen werden, gelangen zunächst nur "vorläufig" in den flüchtigen RAM-Speicher des Moduls, und gehen somit beim Abschalten der Versorgungsspannung verloren.
Dieses Kommando übernimmt daher die aktuellen Einstellungen in den EEPROM-Speicher des Moduls und legt sie dort dauerhaft ab.
Nach dem Einschalten der Versorgungsspannung verwendet das Modul die Einstellungen, die mit diesem Kommando zuvor im EEPROM als Startkonfiguration abgelegt wurden.
Nach dem Empfang des Kommandos sendet das Modul den Request unverändert als Response zurück.

cmd_ConfigFromEeprom = 'X'
Stellt die Startkonfiguration wieder her, die im EEPROM-Speicher des Moduls abgelegt wurde (siehe ***cmd_ConfigToEeprom***). Nach dem Empfang des Kommandos sendet das Modul den Request unverändert als Response zurück.

Anhang

Kommmandos (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';
```

Kanaldressierung (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;
```

Protokollstruktur (USB VCP)

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

ResponseSize = 20;
TResponse = TRequest;
```