Nehring PC-Messtechnik • Hauptstraße18 • D-56281 Dörth

Nehring PC Messtechnik

NeUSB

Beschreibung Softwareschnittstelle

Version 1.03 vom 26.01.2021

Inhaltsverzeichnis

Versionshistorie

Version	Datum	Bearbeiter	Beschreibung
1.00	2015-08-18	C. Kremp	Aus Dokumentation NeUSB-digIO extrahiert
1.01	2018-09-19	A. Nehring	Kleinere Formatierungsänderungen
1.02	2019-03-13	A. Nehring	Abschnitt „Softwarebeispiele“ hinzugefügt
1.03	2021-01-26	C. Kremp	Deckblatt aktualisiert

1 Beschreibung

Die Kommunikation des NeUSB-Moduls mit dem PC erfolgt über eine virtuelle serielle Schnittstelle.

Je nach Betriebssystem ist die Installation der „Microchip CDC-Treiber“ erforderlich. Eine Installationsanleitung finden sie in Abschnitt 4.

2 Kommunikation Protokoll

2.1 Schnittstellen Einstellungen

Es werden folgenden Einstellungen verwendet:

- Baudrate 2400 bis 195600; 8 Bits; kein Stoppbit; Parität: keine

Die Baudrate kann beliebig eingestellt werden, da der CDC-Treiber diese auf den USB-Port umsetzt.

2.2 Grundlegende Eigenschaften der Kommunikation

- Um die Lesbarkeit des Protokolls zu verbessern, werden alle Befehle und Daten im ASCII-Format als Hex-Zahl übertragen: d.h.: aus 1 BYTE werden 2 ASCI-Bytes
- PC-Startzeichen: Jeder Befehl vom PC fängt mit einem ‘#’ an
- Controller-Startzeichen: Jede Befehl vom Controller beginnt mit einem ‘!’
- Der Controller antwortet auf jeden Befehl vom PC mit einer Quittung, bestehend auf Controller-Startzeichen gefolgt vom Befehl, ggf. gefolgt von Daten
- Jeder Befehl wird mit einem ‘\r\n’ abgeschlossen

2.3 Befehlssatz

Um die Anzahl der möglichen Befehl zu erweitern, ist der Befehlssatz ist in nachfolgende Modulbereiche unterteilt:

Jedes Modul erhält eine „SUB-Modul-Kennung“, über die das Modul adressiert wird.

SUB-Modulbereich	Befehlszeichen
Allgemeine Befehle	#A, #Y, #Z
Befehle für SUB-Modul NeUSB-dig/O	#B,….. Unterbefehl für Modul
Befehle für SUB-Modul NeUSB-DMS	#C,…. Unterbefehl für Modul
Befehle für SUB-Modul NeUSB-MEMs	#D,…. Unterbefehl für Modul
Befehle für SUB-Modul NeAcustik-RPM	#E,…. Unterbefehl für Modul

Für C++ ist der Befehlssatz ist in der Datei „NeUSB_Command.h“ definiert.

Für C# ist der Befehlssatz ist in der Datei „GeraeteManagerNeUSB.cs“ definiert.

2.4 Allgemeiner Befehlssatz

Befehl	Beschreibung	Antwort des Controllers
„A“	Modulinformationen einlesen #A\r\n ab Software Version 1.20	!A,HS:xx,MK:xx,SV:xx,HV:xx, SN:xx,DI:xx, DO:xx, AI:xx,\r\n xx = Anzahl der Bytes, HB -> LB HS: Hersteller [String] MK: Modulkennung [String] SV: Softwareversion [String] SN: Seriennummer [String] HV: Hardware-Variante [String] DI: Info digitale Eingänge [String] DO: Info digitale Ausgänge [String] AI: Info analoge Eingänge [String] BV: Brückenverstärker [String] ME: MEMs Sensor [String] AD: ADC24 Bit Modul siehe hierzu Absatz: **Fehler! Verweisquelle konnte nicht gefunden werden.**
„Y“	unbekannter Befehl empfangen	!Y,CMD\r\n CMD: als Befehl erkanntes Zeichen [WORD]
„Z“	Modulstatus einlesen #Z\r\n	!Z,MS\r\n MS: Modulstatus [WORD] siehe hierzu Absatz: **Fehler! Verweisquelle konnte nicht gefunden werden.**

2.5 Befehlssatz für NeUSB-digIO

SUB-Modul-Kennung: „B“

Befehl	Beschreibung	Antwort des Controllers
„A“	Status der digitalen Eingänge einlesen #BA\r\n	!BA,FD\ r\n FD digitale Eingänge [WORD]
„B“	Digitale Ausgänge setzen #BB,FF\r\n FF Statusflag der digitalen Ausgänge [WORD]	!BB\r\n
„C“	Status digitale Ausgänge einlesen #BC,FF\r\n	!BC,FF\r\n FF Statusflag der digitalen Ausgänge [WORD]
„D“	Enable Auto Send, wenn DI sich ändert #BD,Enable Mit Enable= 00 oder 01	!B,D\r\n Vor 20.10.2017 war 0 und 1 in der Bedeutung vertauscht

2.6 Befehlssatz für DMS-Brückenverstärker

SUB-Modul-Kennung: „C“

Befehl	Beschreibung	Antwort des Controllers
„A“	Konfiguration setzen #CA\r\n	!CA\r\n Wird zur Zeit nicht unterstützt
„B“	Konfiguration auslesen #CB\r\n	!CB,ST:xx,MO:xx,CF:xx,ID:xx,OF:xx, FS:xx, CM:xx,\r\n xx = Anzahl der Bytes, HB -> LB ST: Status-Register [1 BYTE] MO: Mode-Register [2 BYTE] CF: Configuration-Register [2 BYTE] ID: ID-Register [1 BYTE] OF: Offset-Register [3 BYTE] FS: Fullscale-Register [3 BYTE] CM: Command-Register [1 BYTE]
„C“	DMS-Daten auslesen #CC\r\n	!C,C,X1X2X3,RC,CS\r\n X1X2X3: DMS-Daten [6 BYTE] RC: Rolling Counter [2 BYTE] CS: Checksumme [2 BYTE] Reihenfolge der Bytes: HB -> LB
„D“	Automatisches Senden der DMS-Daten aktivieren #CD,Flag\r\n Flag:: boolsches Flag [1 BYTE] CMD = 01: enabled CMD = 00: disabled	!C,D\r\n
„E“	Chip Temperatur einlesen #CE\r\n	!C,E,X1X2X3\r\n X1X2X3: Chip-Temperatur [6 BYTE] Reihenfolge der Bytes: HB -> LB
„F“	Versorgungsspannung einlesen #CF\r\n	!C,F, X1X2X3\r\n X1X2X3: Versorgungsspannung [6 BYTE] Reihenfolge der Bytes: HB -> LB
„G“	Reset Brückenverstärker #CG\r\n	!C,G,\r\n
„H“	Kalibrierung internal Offset #CH\r\n	!C,H,\r\n
„I“	Kalibrierung Systemoffset #CI\r\n	!C,I,\r\n
„J“	Kalibrierung Full-Scale #CJ\r\n	!C,J,\r\n
„M“	Abtastrate	!C,M,X1\r\n X1: Abtastraten-Enum [2 BYTE]
„Z“	Unknown Kommand #CZ\r\n	!C,Z\r\n

2.7 Befehlssatz für MEMs-Sensor

SUB-Modul-Kennung: „D“

Befehl	Beschreibung	Antwort des Controllers
„A“	Konfiguration setzen #DA\r\n	!D,A\r\n
„B“	Konfiguration auslesen #DB\r\n	!D,B,\r\n xx = Anzahl der Bytes, HB -> LB
„C“	Sensor Daten auslesen #DC\r\n	!D,C,X1X2X3\r\n X1X2X3: DMS-Daten [6 BYTE] Reihenfolge der Bytes: HB -> LB
„D“	Automatisches Senden der Sensor-Daten aktivieren #DD,Flag\r\n Flag:: boolsches Flag [1 BYTE] CMD = 1: enabled CMD = 0: disabled	!D,D\r\n
„Z“	Unknown Kommand Sendet der Controller, wenn er einen unbekannten Befehl erhalten hat	!Z,SubModul,Command\r\n SubModul Kennung Submodul [1 BYTE] *1 Command Received unknown Command [1 BYTE] *1

2.8 Befehlssatz für NeAcoustik-RPM

SUB-Modul-Kennung: „E“

Befehl	Beschreibung	Antwort des Controllers
„A“	Get Configuration #EA\r\n	!E,A,P1,xP2,P3,P4,P5,P6,\r\nP7\r\n P1:: Flag: Auto Send ADC-Data [2 BYTE] *1 P2:: Flag: Auto Send Frequenz [2 BYTE] *1 P3:: Flag: Debug Auto Gain [2 BYTE] *1 P4:: Flag: AutoGain enabled [2 BYTE] *1 P5:: BYTE, aktuelles PGA Gain [2 BYTE] *1 P6:: BYTE, sizeof Rechenparameter [2 BYTE] *1 \r\n Trenner: ab jetzt Binärdaten P7 P6 x Binärdaten für Rechneparameter
„B“	Noch nicht implementiert Set Configuration #E,B,P1,xP2,P3,P4,P5,P6,\r\nP7\r\n P1:: Flag: Auto Send ADC-Data enabeld P2:: Flag: Auto Send Frequenz enabled P3:: Flag: Debug Auto Gain enabled P4:: Flag: AutoGain enabled P5:: BYTE, PGA Gain P6:: BYTE, size of Rechenparameter \r\n Trenner: ab jetzt kommen P6 Binärdaten P7 P6 x Binärdaten für Rechneparameter	Noch nicht implementiert !E,B\r\n
„C“	Automatisches Senden der ADC-Daten aktivieren #EC,Flag\r\n Flag::boolsches Flag [2 BYTE] *1 = 1: enabled = 0: disabled	!E,C,Flag\r\n Flag:: boolsches Flag [2 BYTE] *1 = 01: enabled = 00: disabled

„D“	Automatisches Senden der Frequenz aktivieren #ED,Flag\r\n Flag::boolsches Flag [2 BYTE] *1 = 1: enabled = 0: disabled	!E,D,Flag\r\n Flag:: boolsches Flag [2 BYTE] *1 = 1: enabled = 0: disabled
„E“	Aktuelle Drehzahl abfragen #EE\r\n	!E,E,PRM\r\n PRM: WORD in U/Min [4 BYTE] *1
„F“	ADC-Daten (zyklisch) #EF\r\n Die ADC-Daten werden zyklisch abwechselnd in 2 Buffern übertragen.	!E,F,BufferNr,AnzDat\r\nADC-Data BufferNr BYTE Nummer des Buffers [2 BYTE] *1 AnzDat WORD Anzahl Byte ADC-Daten [4 BYTE] *1 ADC-Data AnzDat Byte als WORD [LB, HB] [AnzDat BYTE] *2
„G“	Gain für PGA einstellen #EG,Gain\r\n Gain: Beschreibung = siehe Antwort	!E,G,Gain\r\n Gain: Gain 0 .. 7: 0 = aus, 1 = min, 7 = max, FF = invalid Gian [2 BYTE] *1
„H“	Erkanntes Maximum am PGA auslesen #EH\r\n	!E,H,Max,Gain\r\n Max: Erkanntes Maximum am PGA-Ausgang [4 BYTE] *1 Gain: aktuelles Gain des PGA [2 BYTE] *1
„I“	Debug Autogain aktivieren #EI,Flag\r\n Flag::boolsches Flag [2 BYTE] *1 = 1: enabled = 0: disabled - Botschaft ‚H‘ wird zyklisch gesendet - Botschaft „K“ sendet die ADC-Daten des PGA-Ausgangs	!E,I,Flag\r\n Flag:: boolsches Flag [2 BYTE] *1 = 1: enabled = 0: disabled
„J“	Disable, Enable Autogain #EJ,Flag\r\n Flag::boolsches Flag [2 BYTE] *1 = 1: enabled = 0: disabled	!E,J,Flag\r\n Flag:: boolsches Flag [2 BYTE] *1 = 1: enabled = 0: disabled

„K“	ADC-Daten PGA-Ausgang (zyklisch) #EK\r\n Die ADC-Daten des PGA-Ausgangs werden zyklisch abwechselnd in 2 Buffern übertragen.	!E,K,BufferNr,AnzDat\r\nADC-Data BufferNr BYTE Nummer des Buffers [2 BYTE] *1 AnzDat WORD Anzahl Byte ADC-Daten [4 BYTE] *1 ADC-Data AnzDat Byte als WORD [LB, HB] [AnzDat BYTE] *2
„Z“	Unknown Kommand Sendet der Controller, wenn er einen unbekannten Befehl erhalten hat	!Z,SubModul,Command\r\n SubModul Kennung Submodul [1 BYTE] *1 Command Received unknown Command [1 BYTE] *1

Legende:

*1 im ASCII Format

*2 im Binär Format

2.9 Modulinformationen einlesen

Je nach Hardwarevariante und Softwareversion, ist das NeUSB-Modul in unterschiedlichen Konfigurationen erhältlich.

Um herauszufinden, mit welcher Konfigurationen das Modul ausgestattet ist, kann mit dem Befehl ‚#E' die erweiterte Modulinformation abgefragt werden.

Die Modulinformation beinhaltet mehrere verkettete Abschnitte. Jeder Abschnitt beginnt mit einem Suchkriterium - bestehend aus zwei Großbuchstaben und einem ‚:'. – gefolgt von der eigentlichen Information. Der Abschnitt wird mit einem Komma abgeschlossen.

Die erweiterte Modulinformation beinhaltet nachfolgende Informationen:

<table>
<tr><th>Modulfeature</th><th>Suchkriterium</th><th>Inhalt</th></tr>
<tr><td>Hersteller</td><td>HS:</td><td>Nehring PC Messtechnik</td></tr>
<tr><td>Modulkennung</td><td>MK:</td><td>NeUSB-digI/O</td></tr>
<tr><td>Hardware-Variante</td><td>HV:</td><td>Mögliche Varianten:
Stiftleiste, SUB-D, Optokoppler, Optokoppler Push-Pull-Treiber</td></tr>
<tr><td>Digitale Eingänge</td><td>DI:</td><td>Je nach Hardware-Varianten:
<table>
<tr><th>Hardware-Varianten</th><th>Information</th></tr>
<tr><td>SUB-D</td><td>TTL</td></tr>
<tr><td>Stiftleiste</td><td>TTL</td></tr>
<tr><td>Optokoppler</td><td>Optokoppler – Schaltschwelle in V</td></tr>
<tr><td>Optokoppler Push-Pull</td><td>Optokoppler – Schaltschwelle in V</td></tr>
</table></td></tr>
<tr><td>Digitale Ausgänge</td><td>DO:</td><td>Je nach Hardware-Varianten:
<table>
<tr><th>Hardware-Varianten</th><th>Information</th></tr>
<tr><td>SUB-D</td><td>TTL</td></tr>
<tr><td>Stiftleiste</td><td>TTL</td></tr>
<tr><td>Optokoppler</td><td>Open Kollektor</td></tr>
<tr><td>Optokoppler Push-Pull</td><td>Optokoppler mit Push-Pull-Treiber</td></tr>
</table></td></tr>
<tr><td>Analoge Eingänge</td><td>AI:</td><td>0..5V (keine gal. Trennung)</td></tr>
<tr><td>DMS-Brückenverstärker</td><td>BV:</td><td>Typenbezeichnung des verbauten ICs
AD7790</td></tr>
<tr><td>MEMs-Sensoren</td><td>ME:</td><td>Typenbezeichnung des verbauten ICs
z.B.: MPSxxxx</td></tr>
</table>

3 Softwarebeispiele

3.1 NeUSB-DMS-light

Beispielprogramm mit vollständigem Quellcode zur Benutzung des NeUSB DMS. Dieses Beispielprogramm zeigt die kalibrierten Werte eines DMS-Sensors an.

Compiler	Microsoft Visual Studio 2017
Sprache	C#
.NET Framework	4

3.2 Beispiele in C++

Auf Anfrage.

4 Installation Microchip CDC-Treiber

========== TODO ==========

Den „Microchip CDC-Treiber“ finden sie in Ordner: C:\Programm (x86)\L:\PROJEKTE\NeUSB\Microchip CDC-Treiber

Die Installation des Treibers erfolgt durch das Ausführen der Datei „MCP2200DriverInstallationTool.exe“ aus dem entsprechenden Unterordner - x86 bei 32 Bit oder x64 64Bit Betriebssystem.

Der „Microchip CDC-Treiber“ unterstützt nachfolgende Betriebssysteme:

- Windows XP SP3,
- Vista
- WIN7 (64 und 32 Bit) – Administrationsrechte sind erforderlich

5 USB Spezifikation

5.1 Controller

- Controller 16F1455
- Microchip MLA Version v2014_07_22
- CDC-Service

5.2 Verwendeter Treiber

- INF-Datei: mchpcdc.inf
- Gerätetreiber