|  |  |  |  |
| --- | --- | --- | --- |
|  | |  | |
| smartNIV  Sensor-Patch Schnittstellen Spezifikation | |
|  | |  | |
| Version  **Dokument-Tags** | | 2.0 smartNIV | |
| Content | | Spezifikation des Kommunikationprotokolls zwischen der smartNIV software und dem Sensor-Patch Controller-PCB | |

|  |  |  |
| --- | --- | --- |
| Zuständigkeit | Name | Funktion |
|  | msg DAVID GmbH | Implementierung SmartNIV Software |
|  | Eugen Koch | Implementierung Controller-PCB |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Änderungen | Author | Änderung | Datum | Version |
|  | Dmitrij Drandarov | Initial | 27. Januar 2021 | 1.0 |
|  | Yang Wang | Protocol for new patch | 07. March 2022 | 2.0 |
|  |  |  |  |  |

Content

[1 Ziel des Dokuments 3](#_Toc97556331)

[2 Einleitung 3](#_Toc97556332)

[3 Schnittstellen Konfiguration 3](#_Toc97556333)

[4 Frame 3](#_Toc97556334)

[5 Befehle 4](#_Toc97556335)

[5.1 Test Übertragung 4](#_Toc97556336)

[5.1.1 Befehl-Argumente 4](#_Toc97556337)

[5.1.2 Antwort 4](#_Toc97556338)

[5.2 LED an-/Ausschalten 4](#_Toc97556339)

[5.2.1 Befehl-Argumente 4](#_Toc97556340)

[5.2.2 Antwort 4](#_Toc97556341)

[5.3 Sensordaten einmalig abfragen 4](#_Toc97556342)

[5.3.1 Befehl-Argumente 4](#_Toc97556343)

[5.3.2 Antwort 5](#_Toc97556344)

[5.4 Sensordaten kontinuierlich abfragen 5](#_Toc97556345)

[5.4.1 Befehl-Argumente 5](#_Toc97556346)

[5.4.2 Antwort 5](#_Toc97556347)

[5.5 Sensor-Offset automatisch setzen 5](#_Toc97556348)

[5.5.1 Befehl-Argumente 5](#_Toc97556349)

[5.5.2 Antwort 5](#_Toc97556350)

[5.6 SensorDaten Kontinuierlich abfragen Stoppen 5](#_Toc97556351)

[5.6.1 Befehl-Argumente 6](#_Toc97556352)

[5.6.2 Antwort 6](#_Toc97556353)

[6 Beispiel 6](#_Toc97556354)

[6.1 Anfrage 6](#_Toc97556355)

[6.2 Antwort 6](#_Toc97556356)

[A Glossary 7](#_Toc97556357)

1. Ziel des Dokuments

Das Ziel des Dokuments ist es eine einheitliche und klare Spezifikation für die Kontrolle und das Abfragen des Sensor-Patches zu definieren.

1. Einleitung

Die Kommunikation zwischen den beiden Geräten findet über eine serielle Schnittstelle statt. Jede Nachricht wird von einem Frame ummantelt, der einen Header mit dem Start-Byte und der Identifikation des Befehls und einen Footer mit dem End-Byte beinhaltet. Auf Checksummen wurde verzichtet, um die hochfrequente Kommunikation möglichst performant zu halten.

1. Schnittstellen Konfiguration

Die serielle Schnittstelle nutzt folgende Einstellungen:

* Data bits: 8
* Stop bits: 1
* Flow control: -
* Parity: 1
* Baud rate: 230400

1. Frame

Jede Nachricht ist in einem Frame beinhaltet. Der Header beinhaltet den Start-Byte und die Identifikation des Befehls. Der Footer enthält das End-Byte. Für Daten, die mehrere Bytes spannen benutzen wir **Big Endian**.

|  |  |  |
| --- | --- | --- |
| **Byte #** | **Name** | **Länge**  **(Bytes)** |
| *Header* | | |
| 0 | Start-Byte (0x81) | 1 |
| 1 | Befehl-Byte | 1 |
| *Payload* | | |
| 2..n+1 | Befehl-Argumente / Antwort | N |
| *Footer* | | |
| n+2 | End-Byte (0x7E) | 1 |

|  |  |
| --- | --- |
| **Datentyp** | **Bedeutung** |
| Uint8 | Einzelne Bytes sollen als unsigned 8-bit Integer interpretiert werden. |
| Uint16 | Zwei zusammenhängende Bytes sollen als Big Endian unsigned 16-bit Integer interpretiert werden. |
| Uint32 | Vier zusammenhängende Bytes sollen als Big Endian unsigned 32-bit Integer interpretiert werden. |
| ASCII | Einzelne Bytes sollen als ASCII Charaktere interpretiert werden. |

1. Befehle

Jeder Befehl besitzt ein Befehl-Byte, das im Header des Frames der Anfrage und der Antwort beinhaltet werden muss. Der Befehl-Inhalt und die Antwort sind folgend in zwei Tabellen aufgeteilt. Zudem wird die Anzahl der Sensoren die Variable „n“ benutzt.

* 1. Test Übertragung

Es wird eine Testübertragung von der Hardware angefragt. Die Hardware antwortet mit einer ASCII Nachricht.

* **Befehl-Byte:** 0x01
  + 1. Befehl-Argumente

-

* + 1. Antwort

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Byte #** | **Byte-Format** | **Name** | **Typ** | **Länge**  **(Bytes)** |
| 0..4 | 0x54 0x65 0x73 0x74 0x00 | Test\0 (ASCII) | ASCII | 5 |

* 1. LED an-/Ausschalten

Es wird die LED, die auf dem Controller-PCB angebracht ist, an- oder ausgeschaltet.

* **Befehl-Byte:** 0x02
  + 1. Befehl-Argumente

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Byte #** | **Byte-Format** | **Name** | **Typ** | **Länge**  **(Bytes)** |
| 0 | 0x00 / 0x01 | LED aus / LED an | uint8 | 1 |

* + 1. Antwort

-

* 1. Sensordaten einmalig abfragen

Fragt mit den angegebenen Parametern einmalig Sensorwerte ab.

* **Befehl-Byte:** 0x03
  + 1. Befehl-Argumente

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Byte #** | **Byte-Format** | **Name** | **Typ** | **Länge**  **(Bytes)** |
| 0 | 0x00 – 0x05 | x\_min | uint8 | 1 |
| 1 | 0x00 – 0x05 | x\_max | uint8 | 1 |
| 2 | 0x00 – 0x05 | y\_min | uint8 | 1 |
| 3 | 0x00 – 0x05 | y\_max | uint8 | 1 |
| 4..5 | 0x0000 – 0xFFFF | delay\_switch (Big endian) | uint16 | 2 |
| 6..7 | 0x0000 – 0xFFFF | delay\_meas (Big endian) | uint16 | 2 |

* + 1. Antwort

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Byte #** | **Byte-Format** | **Name** | **Typ** | **Länge**  **(Bytes)** |
| 0..3 | 0x00000000 – 0xFFFFFFFF | Timestamp (Big endian) | Uint32 | 4 |
| 4..(4+n\*2) | 0x0000 – 0xFFFF | Sensor Werte (Big endian) | uint16 | n\*2 |

* 1. Sensordaten kontinuierlich abfragen

Fragt mit den übergebenen Parametern kontinuierlich Sensorwerte ab.

* **Befehl-Byte:** 0x04
  + 1. Befehl-Argumente

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Byte #** | **Byte-Format** | **Name** | **Typ** | **Länge**  **(Bytes)** |
| 0 | 0x00 – 0x05 | x\_min | uint8 | 1 |
| 1 | 0x00 – 0x05 | x\_max | uint8 | 1 |
| 2 | 0x00 – 0x05 | y\_min | uint8 | 1 |
| 3 | 0x00 – 0x05 | y\_max | uint8 | 1 |
| 4..5 | 0x0000 – 0xFFFF | delay\_switch (Big endian) | uint16 | 2 |
| 6..7 | 0x0000 – 0xFFFF | delay\_meas (Big endian) | uint16 | 2 |

* + 1. Antwort

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Byte #** | **Byte-Format** | **Name** | **Typ** | **Länge**  **(Bytes)** |
| 0..3 | 0x00000000 – 0xFFFFFFFF | Timestamp (Big endian) | Uint32 | 4 |
| 4..(4+n\*2) | 0x0000 – 0xFFFF | Sensor Werte (Big endian) | uint16 | n\*2 |

* 1. Sensor-Offset automatisch setzen

Offset-Werte, wie sie zum Zeitpunkt der Messung sind, werden als Rückgabe-Wert zurückgegeben. Gibt den Spannungszustand des Patches zurück. Und gibt auch die Werte, also die Differenz an, die notwendig war, um die Spannung auf 0 zu setzen. D.h. danach sind sie 0. Wenn man also direkt danach nochmal messen würde, wäre das Ergebnis 36 mal 0.

* **Befehl-Byte:** 0x05
  + 1. Befehl-Argumente

-

* + 1. Antwort

-

* 1. SensorDaten Kontinuierlich abfragen Stoppen

Stoppt die kontinuierliche Rückgabe von Sensorwerten.

* **Befehl-Byte:** 0x08
  + 1. Befehl-Argumente

-

* + 1. Antwort

-

1. Beispiel

Im Folgenden wir ein Beispiel für die erwartete Kommunikation beim Ausführen von Befehl „Sensordaten einmalig abfragen“ (0x03) gezeigt.

**Legende:**

* Frame: 0x81 0x01:
* Nachrichten-Inhalt: 0x30 0x01
  1. Anfrage

**Anfrage (Parameter):**

* xmin = 0
* xmax = 3
* ymin = 0
* ymax = 3
* delay\_swtich = 300
* delay\_meas = 10

**Anfrage (Hex):**

* 0x81 0x03 0x00 0x03 0x00 0x03 0x01 0x2c 0x00 0x0A 0x7E
  1. Antwort

**Antwort gekürzt (Parameter):**

* 1482 / 2076 / .. / 1446

**Anfrage gekürzt (Hex):**

* 0x81 0x03 0xCA 0x05 0xC1 0x08 .. 0x6A 0x05 0x7E

1. Glossary

|  |  |
| --- | --- |
| Begriff | Beschreibung |
| x\_min | Die untere Grenze der Range der Sensoren, die auf der x-Achse gemessen werden. |
| x\_max | Die obere Grenze der Range der Sensoren, die auf der x-Achse gemessen werden. |
| y\_min | Die untere Grenze der Range der Sensoren, die auf der y-Achse gemessen werden. |
| y\_max | Die obere Grenze der Range der Sensoren, die auf der y-Achse gemessen werden. |
| delay\_switch | Wartet x Mikrosekunden zwischen den einzelnen Sensormessungen. |
| delay\_meas | Wartet x Millisekunden zwischenzwei Gesamtmessungen (alle Sensoren). |