|  |  |  |  |
| --- | --- | --- | --- |
|  | |  | |
| smartNIV  Sensor-Patch Schnittstellen Spezifikation | |
|  | |  | |
| Version  **Dokument-Tags** | | 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 |
|  |  |  |  |  |

Inhalt

[1 Ziel des Dokuments 3](#_Toc62672318)

[2 Einleitung 3](#_Toc62672319)

[3 Schnittstellen Konfiguration 3](#_Toc62672320)

[4 Frame 3](#_Toc62672321)

[5 Befehle 4](#_Toc62672322)

[5.1 Test Übertragung 4](#_Toc62672323)

[5.1.1 Befehl-Argumente 4](#_Toc62672324)

[5.1.2 Antwort 4](#_Toc62672325)

[5.2 LED an-/Ausschalten 4](#_Toc62672326)

[5.2.1 Befehl-Argumente 4](#_Toc62672327)

[5.2.2 Antwort 4](#_Toc62672328)

[5.3 Sensordaten einmalig abfragen 4](#_Toc62672329)

[5.3.1 Befehl-Argumente 4](#_Toc62672330)

[5.3.2 Antwort 5](#_Toc62672331)

[5.4 Sensordaten kontinuierlich abfragen 5](#_Toc62672332)

[5.4.1 Befehl-Argumente 5](#_Toc62672333)

[5.4.2 Antwort 5](#_Toc62672334)

[5.5 Sensor-Offset automatisch setzen 5](#_Toc62672335)

[5.5.1 Befehl-Argumente 5](#_Toc62672336)

[5.5.2 Antwort 5](#_Toc62672337)

[5.6 Sensor-Offset manuell setzen 5](#_Toc62672338)

[5.6.1 Befehl-Argumente 6](#_Toc62672339)

[5.6.2 Antwort 6](#_Toc62672340)

[5.7 Voltage Gain setzen 6](#_Toc62672341)

[5.7.1 Befehl-Argumente 6](#_Toc62672342)

[5.7.2 Antwort 6](#_Toc62672343)

[5.8 SensorDaten Kontinuierlich abfragen Stoppen 6](#_Toc62672344)

[5.8.1 Befehl-Argumente 6](#_Toc62672345)

[5.8.2 Antwort 6](#_Toc62672346)

[5.9 Messfrequenz abfragen 6](#_Toc62672347)

[5.9.1 Befehl-Argumente 6](#_Toc62672348)

[5.9.2 Antwort 7](#_Toc62672349)

[5.10 Timestamp abfragen 7](#_Toc62672350)

[5.10.1 Befehl-Argumente 7](#_Toc62672351)

[5.10.2 Antwort 7](#_Toc62672352)

[6 Formeln 7](#_Toc62672353)

[A Glossary 8](#_Toc62672354)

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 | 0x00 – 0xFF | m\_values | uint8 | 1 |
| 5..6 | 0x0000 – 0xFFFF | delay\_switch | uint16 | 2 |
| 7..8 | 0x0000 – 0xFFFF | delay\_meas | uint16 | 2 |

* + 1. Antwort

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Byte #** | **Byte-Format** | **Name** | **Typ** | **Länge**  **(Bytes)** |
| 0..(n\*2) | 0x0000 – 0xFFFF | Sensor Werte | 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 | 0x00 – 0xFF | m\_values | uint8 | 1 |
| 5..6 | 0x0000 – 0xFFFF | delay\_switch | uint16 | 2 |
| 7..8 | 0x0000 – 0xFFFF | delay\_meas | uint16 | 2 |

* + 1. Antwort

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Byte #** | **Byte-Format** | **Name** | **Typ** | **Länge**  **(Bytes)** |
| 0..(n\*2) | 0x0000 – 0xFFFF | Sensor Werte | 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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Byte #** | **Byte-Format** | **Name** | **Typ** | **Länge**  **(Bytes)** |
| 0..(n\*2) | 0x0000 – 0xFFFF | Offset Werte | uint16 | n\*2 |

* 1. Sensor-Offset manuell setzen

Setzt die Offset-Werte pro Sensor manuell. Damit kann das Sensor-Patch „programmiert“ werden.

* **Befehl-Byte:** 0x06
  + 1. Befehl-Argumente

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Byte #** | **Byte-Format** | **Name** | **Typ** | **Länge**  **(Bytes)** |
| 0..(n\*2) | 0x0000 – 0xFFFF | Offset Werte | uint16 | n\*2 |

* + 1. Antwort

-

* 1. Voltage Gain setzen

Das ist die Verstärkung von der Brücke zum ADC. Auf der Platine wird nur die Brückenspannung gemessen (in Millivolt) und diese muss verstärkt werden. Das ADC hat trotzdem eine feste Spannung. Das Voltage Gain schiebt die Spannung in einen Bereich, der vom ADC wahrgenommen wird.

* **Befehl-Byte:** 0x07
  + 1. Befehl-Argumente

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Byte #** | **Byte-Format** | **Name** | **Typ** | **Länge**  **(Bytes)** |
| 0 | 0x00 – 0x08 | Voltage Gain | uint8 | 1 |

* + 1. Antwort

-

* 1. SensorDaten Kontinuierlich abfragen Stoppen

Stoppt die kontinuierliche Rückgabe von Sensorwerten.

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

-

* + 1. Antwort

-

* 1. Messfrequenz abfragen

Die echte Frequenz (hardwaretechnisch). Es wird die Zahl geschickt, die der Mikrocontroller gebraucht hat für die Messung.

* **Befehl-Byte:** 0x09
  + 1. Befehl-Argumente

-

* + 1. Antwort

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Byte #** | **Byte-Format** | **Name** | **Typ** | **Länge**  **(Bytes)** |
| 0 | 0x0000 – 0xFFFF | Hardwareseitige Frequenz | uint16 | 2 |

* 1. Timestamp abfragen

TODO: Realisierbarkeit mit dem Mikrokontroller klären.

* **Befehl-Byte:** 0x0A
  + 1. Befehl-Argumente

-

* + 1. Antwort

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Byte #** | **Byte-Format** | **Name** | **Typ** | **Länge**  **(Bytes)** |
| 0 | 0x00000000 – 0xFFFFFFFF | Zeitstempel | uint32 | 4 |

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):**

* 3 / 0 / 5 / 0 / 5 / 10 / 200 / 5

**Anfrage (Hex):**

* 0x81 0x03 0x30 0x00 0x50 0x00 0x50 0xa0 0x00 0xc8 0x00 0x05 0x7E
  1. Antwort

**Antwort gekürzt (Parameter):**

* 1482 / 2076 / .. / 1446

**Anfrage gekürzt (Hex):**

* 0x81 0x03 0x50 0xca 0x80 0x1c .. 0x50 0xa6 0x7E

1. Formeln

Allgemeine Formeln zum Umrechnen einzelner Werte:

|  |  |
| --- | --- |
| **Berechnung** | **Formel** |
| dac\_ref\_offset | dac\_ref / 2 |
| ADC- zu Sensor-Wert | ((adc\_value / dac\_nmax) \* dac\_ref) – dac\_ref\_offset |
| Sensor- zu ADC-Wert | int((sensor\_value + dac\_ref\_offset) / dac\_ref \* dac\_nmax) |
| Digital- zu Analog-Wert | digital\_value / dac\_nmax \* dac\_ref – dac\_ref\_offset |
| Analog- zu Digital-Wert | round((analog\_value + dac\_ref\_offset) / dac\_ref) \* dac\_nmax |
| Frequenz | 78125 / (Hardwareseitige Frequenz aus Befehl 0x09) |

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. |
| m\_values | Wie viele Messungen genutzt werden, um einen Mean für einen einzelnen Sensor zu berechnen. |
| delay\_switch | Wartet x Mikrosekunden zwischen den einzelnen Sensormessungen. |
| delay\_meas | Wartet x Millisekunden zwischenzwei Gesamtmessungen (alle Sensoren). |
| sensor\_offsets | Offset-Werte, die bei der Messung jedes Sensors benutzt werden. Größe der Liste muss der Gesamtanzahl der Sensoren entsprechen. |
| voltage\_gain | Verstärkung von der Brücke zum ADC. Auf der Platine wird nur die Brückenspannung gemessen (in mV). Diese wird verstärkt. ADC hat eine feste Spannung.  Die Werte [1..8] mappen auf [2, 50, 100, 200, 400, 600, 800, 100]. Der Standard-Wert ist 4. |
| ADC-Wert | Der übertragene Sensor-Wert zwischen 0 und 4095 |
| Sensor-Wert | Der in Volt umgerechnete Sensor-Wert zwischen -1.25 und 1.25 V |
| Analog-Wert | Auf dem Sensorpatch gemessene und verstärkte Brückenspannung |
| Offset-Wert | Der Wert zwischen 0 und 4095 mit dem der Sensor ausgeglichen wird, um einen neutralen Sensor-Wert von 0 zu bekommen, wenn das Sensor-Patch nicht beansprucht wird. |
| dac\_ref | Die Range zwischen dem kleinsten und größten messbaren Sensor-Werten (-1.25 bis 1.25 🡪 2.5). |
| dac\_nmax | Der maximal messbare Sensor-Wert in Volt des Digital-Analog-Konverters (3.3 / 2V). |