

|  |
| --- |
| **HiL-Interface** |
| Projektarbeit IPA |

Eine Dokumentation über die Entwicklung eines Interface-Systems zur Automatisierung der Tests für die neu entwickelte DismessA-Steuerung. Die Arbeit wurde im Rahmen der Lehrabschlussprüfung (IPA) durchgeführt.

Autor: Yannick Burkhalter  
Fachvorgesetzte: Simon Grossenbacher  
 Lukas Röthlisberger  
Experten: Stefan Schneider  
 Andreas Kistler  
Auftraggeber: Biral AG 05.04.2018

**Abstract**

Die im nachfolgenden Dokument beschriebene Arbeit wurde im Ramen einer individuellen Projektarbeit für die Lehrabschlussprüfung durchgeführt:

Die Aufgabe lautete, ein Hardware in the Loop Interface zu entwickeln, dass den Prüfvorgang der neu Entwickelten DismessA-Steuerung automatisieren und somit beschleunigen soll. Daher muss die Hardware alle Ein- und Ausgänge der DismessA-Steuerung stimulieren und muss mit einem Personal Computer angesteuert werden können. Zu der Hardware soll auch die Entsprechende Software für das Interface und die Software für die Ansteuerung über einen Computer entwickelt werden.

Als Vorarbeit zu dieser Aufgabe wurde ein Simulator entwickelt. Dieser kann bereits einige der Eingänge und Ausgänge stimulieren und kann vom Computer aus angesteuert werden. Für die neue Hardware und Gerätefirmware für das Interface können einige Teile des Vorprojektes übernommen werden. Ein Grossteil muss jedoch neu Entwickelt oder weiterentwickelt werden.

Die entwickelte Hardware kann alle Ein- und Ausgänge der DismessA-Steuerung ansteuern und stimulieren. Auch die Ansteuerung von einem Computer funktioniert mit Hilfe eines in der Firmware programmierten Parser. Durch den Parser können Befehle an das Interface gesendet werden, welche von Menschen gelesen werden können. Dazu wurde auch ein Demo-Programm, welches zur Demonstration der Fähigkeiten des Interfaces gedacht ist, auf Basis der Programmiersprache C# entwickelt.

Am Ende der Arbeit wurden alle grundlegenden Ziele erreicht und es konnten sogar einige optionale Ziele erfüllt werden. Arbeiten die noch weitergeführt werden könnten, wären die Entwicklung eines Messe-Demo-Programms, welches eine schönere visuelle Präsentation der DismessA-Steuerung bietet als das aktuelle Demo-Programm, und einer Unit-Test Library. Die Unit-Test-Library wird dazu verwendet, dass das Interface über ein Raspberry-Pi angesteuert wird, welches von einem zweiten Computer kontrolliert wird.

**Inhaltsverzeichnis**

[1 Einleitung 7](#_Toc510694377)

[1.1 Aufgabenstellung 7](#_Toc510694378)

[1.2 Vorgehen 7](#_Toc510694379)

[1.2.1 Vorarbeiten 7](#_Toc510694380)

[1.2.2 IPA 8](#_Toc510694381)

[1.3 Zielsetzung 9](#_Toc510694382)

[1.3.1 Grundfunktionen 9](#_Toc510694383)

[1.3.2 Optional 9](#_Toc510694384)

[2 Hardware 10](#_Toc510694385)

[2.1 Konzept 10](#_Toc510694386)

[2.1.1 Mikrocontroller 10](#_Toc510694387)

[2.1.2 Clock 11](#_Toc510694388)

[2.1.3 Serielle Schnittstelle 11](#_Toc510694389)

[2.1.4 Anzeige 11](#_Toc510694390)

[2.1.5 Digitale Outputs 12](#_Toc510694391)

[2.1.6 Digitale Inputs 230VAC 13](#_Toc510694392)

[2.1.7 Stromschnittstelle 13](#_Toc510694393)

[2.1.8 Anschlüsse 13](#_Toc510694394)

[2.1.9 Speisung 14](#_Toc510694395)

[2.2 Realisierung 16](#_Toc510694396)

[2.2.1 Blockschaltbild 16](#_Toc510694397)

[2.2.2 Mikrocontroller 16](#_Toc510694398)

[2.2.3 Serielle Schnittstelle 19](#_Toc510694399)

[2.2.4 Inputs 21](#_Toc510694400)

[2.2.5 Outputs 22](#_Toc510694401)

[2.2.6 Anschlüsse 24](#_Toc510694402)

[2.2.7 Speisung 24](#_Toc510694403)

[3 Software 25](#_Toc510694404)

[3.1 Firmware 25](#_Toc510694405)

[3.1.1 Aufbau 25](#_Toc510694406)

[3.1.2 Variablen/Defines 25](#_Toc510694407)

[3.1.3 Ablauf 25](#_Toc510694408)

[3.1.4 Module 28](#_Toc510694409)

[3.2 GUI 34](#_Toc510694410)

[3.2.1 Aufbau 34](#_Toc510694411)

[3.2.2 Ablauf 35](#_Toc510694412)

[4 Bedienungsanleitung 38](#_Toc510694413)

[4.1 Bedienung und Anzeige 38](#_Toc510694414)

[4.1.1 Anzeige 38](#_Toc510694415)

[4.1.2 Anschluss 38](#_Toc510694416)

[4.1.3 Start 39](#_Toc510694417)

[4.1.4 Bedienung durch Hterm 39](#_Toc510694418)

[4.1.5 Bedienung mit Demonstrationssoftware 41](#_Toc510694419)

[5 Schlussbetrachtung 43](#_Toc510694420)

[5.1 Fazit 43](#_Toc510694421)

[5.2 Verbesserungsmöglichkeiten 43](#_Toc510694422)

[5.2.1 Hardware 43](#_Toc510694423)

[5.2.2 Software 43](#_Toc510694424)

[5.3 Ausblicke 44](#_Toc510694425)

[6 Glossar 44](#_Toc510694426)

[7 Literatur- und Quellenverzeichnis 45](#_Toc510694427)

[7.1 Internet 45](#_Toc510694428)

[8 Anhang 46](#_Toc510694429)

[8.1 Datenträgerstruktur 46](#_Toc510694430)

**Abbildungsverzeichnis**

Abbildung 1: DismessA-Simulator 7

Abbildung 2: Blockschaltbild 16

Abbildung 3: Speisung Mikrocontroller 16

Abbildung 4: Pin-Belegung am Mikrocontroller 17

Abbildung 5: SWD-Programmierschnittstelle 19

Abbildung 6: Reset-Taster 19

Abbildung 7: Beschaltung FT232RL 20

Abbildung 8: Beschaltung der Stiftleiste 21

Abbildung 9: 230 VAC Spannungsauswertung 21

Abbildung 10: Einlesen der Relais 22

Abbildung 11: Beschaltung ULN2803A 22

Abbildung 12: Relais-Schaltung 22

Abbildung 13: Heartbeat-LED 23

Abbildung 14: 4-20 mA Schnittstelle 23

Abbildung 15: Speisung 24

Abbildung 16: Flussdiagramm 26

Abbildung 17: Messresultate Stromschnittstelle (nicht-linearisiert) 28

Abbildung 18: Linearisierungsfunktion 29

Abbildung 19: Umrechnungsfunktion 30

Abbildung 20: State-Event-Diagramm Parsing 31

Abbildung 21: Befehls-Syntax 31

Abbildung 22: Entprellung 33

Abbildung 23: GUI-Aufbau 34

Abbildung 24: GUI-Events 36

Abbildung 25: Thread-Interaction 37

Abbildung 26: Anschluss HiL-Interface 38

Abbildung 27: Einstellungen Hterm 39

Abbildung 28: Hterm Kommunikation 40

Abbildung 29: Verbinden der Demo-Software 41

Abbildung 30: Bedienoptionen Demo-Software 41

Abbildung 31: Ordnerstruktur 46

**Tabellenverzeichnis**

Tabelle 1: Strombedarf 24 VDC Speisung 14

Tabelle 2:Strombedarf 3.3 VDC Speisung 15

Tabelle 3: Pinbelegung uC 17

**Abkürzungsverzeichnis**

**IPA** Individuelle Projektarbeit  
**HiL** Hardware in the Loop  
**PC** Personal Computer  
**DC** Direct Current  
**AC** Alternating Current  
**FTDI** Future Technology Devices International **UART** Universal Asynchronous Receiver Transmitter  
**DAC** Digital-Analog-Converter  
**PLL** Phase Locked Loop  
**USB** Universal Serial Bus  
**I/O** Inputs/Outputs  
**LED** Light Emitting Diode  
**uC** Mikrocontroller  
**GND** Ground  
**V** Volt  
**A** Ampere  
**IC** Integrated Circuit  
**F** Farad  
**GUI** Graphical User Interface  
**RC** Widerstand **+** Kondensator

# Einleitung

Im Rahmen einer Lehrabschlussprüfung soll eine individuelle Projektarbeit (IPA) durchgeführt werden.

## Aufgabenstellung

In der Biral AG wird, wie in vielen anderen Firmen auch, der Testablauf der eigenen Produkte weiterentwickelt. Aus dem Grund, dass dieser Vorgang immer schneller erfolgen muss, soll ein Gerät entwickelt werden, welches das Testen der neuen DismessA-Steuerung automatisiert. Dies kann durch ein so genanntes HiL-Interface (Hardware in the Loop Interface) realisiert werden.  
  
Die Aufgabe besteht darin, ein HiL-Interface zu entwickeln, welches die Ein- und Ausgänge des Abwassersteuergerätes stimuliert und auch beabsichtigte Fehlerinjektionen zulässt. Dieses Interface soll mit einem Personal Computer (PC) kommuniziere, damit die Testabläufe überwacht und gesteuert werden können.

## Vorgehen

### Vorarbeiten

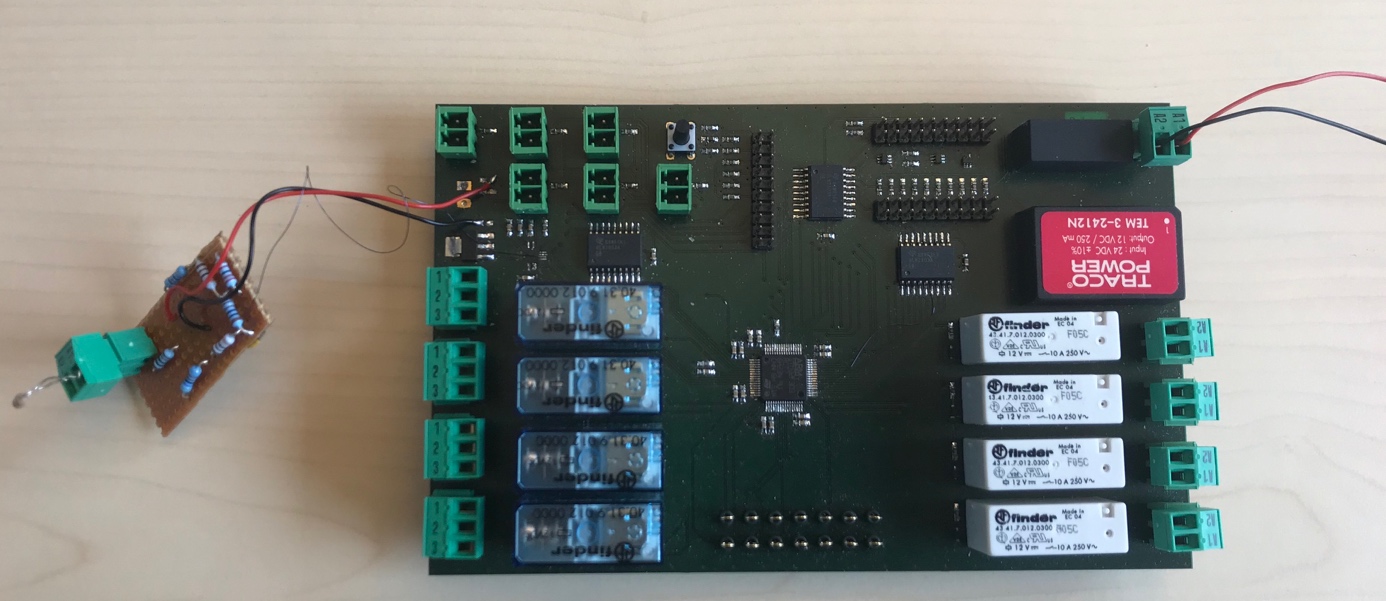
Im Vorfeld zur IPA wurde bereits ein Simulator entwickelt, welcher auch für die DismessA-Steuerung gebaut wurde. Dadurch wurden viele Schnittstellen der Steuerung bereits abgedeckt und können so weiterverwendet werden. In der IPA muss die grundlegende Hardware noch erweitert werden, damit alle Schnittstellen komplett steuerbar sind.

Abbildung 1: DismessA-Simulator

### IPA

Der erste Schritt für das Projekt war die Produktion der Leiterplatte und dessen Bestückung. Da, wie bereits im Abschnitt 1.2.1 beschrieben wurde, einige Teile der Schaltung übernommen werden konnten und noch Erweiterungen gemacht werden mussten, konnte dieser Arbeitsschritt gemäss Zeitplan erledigt werden. Nur ein kleiner Schemafehler, welcher übersehen wurde führte zu einer zeitlichen Verschiebung von einem halben Tag.

Als nächster Schritt musste die Software für das HiL-Interface generiert werden. Dabei konnte ein kleiner Teil der Software vom vorhergegangenen Projekt verwendet werden. Die Software musste nun noch so angepasst werden, dass das HiL-Interface die korrekten Befehle entgegennehmen kann. Des weiteren musste der Softwareteil, welcher die Ansteuerung der Hardware des Interfaces steuert, neu geschrieben werden, damit die zusätzliche hinzugefügte Hardware auch korrekt funktioniert. Da die Hardware 24/7 in Betrieb sein soll, wurde noch ein Watchdog-Timer implementiert.

Als letztes sollte noch eine PC-Software entwickelt werden, mit welcher das HiL-Interface angesteuert werden kann. Dabei ging es in einem ersten Teil darum, mit der Software die Möglichkeiten des HiL-Interfaces aufzuzeigen und mit Hilfe eines „Messe“-Modus diese Software so zu gestalten, dass sie Kunden präsentiert werden kann. Da jedoch die Zeit nicht mehr ausreichte, wurde eine Demoversion erstellt, die keinen Messemodus miteingebaut hat und lediglich zur Demonstration der Funktionen der Hardware dient.  
Auch für das Erstellen einer Unit-Test Library hat es leider nicht gereicht.

## Zielsetzung

Die IPA wurde in zwei Zielbereiche unterteilt. In einen grundlegenden und in einen optionalen Zielbereich.

### Grundfunktionen

Die Grundfunktionen beinhalten alle Hardwareaspekte der Arbeit. Diese sind notwendig, damit die Hardware funktionieren kann.

* Speisung 24V Direct Current (DC), Speisung muss für 24/7 Betrieb ausgelegt werden.
* Abdecken der DismessA-Schnittstellen welche beinhalten:
  + 12 digitale Inputs
  + 4 digitale Outputs
  + 3 Outputs mit 230V Alternating Current (AC) -> galvanisch getrennte Auswertung!
  + 4-20 mA Analog Input
* Digitale Eingänge müssen potentialfrei geschaltet werden.
* Relais-Strom 5 mA bei geschlossenen Kontakten (Frittstrom)
* Steuerung der Stromversorgung der Abwassersteuerung
* Fehlerinjektion in alle Schnittstellen möglich
* Dient nur als Schnittstelle -> Keine logischen Testabläufe mit einbauen

### Optional

Die optionalen Ziele sollen erst realisiert werden, sobald die Grundfunktionen alle einwandfrei funktionieren. Die optionalen Ziele sind:

* Erstellen einer Firmware mit folgenden Aufgaben:
  + Auswerten der Befehle des Clients
  + Bedienen der Schnittstellen zum Abwassersteuergerät
  + Erweiterung der Softwareschnittstellen zum Client nachträglich möglich
  + Betriebszeit 24/7 -> Watchdog Timer einsetzen
* Kommunikationsschnittstelle mit PC über USB mit einem Future Technology Devices International-Chip (FTDI-CHIP) und Universal Asynchronous Receiver Transmitter (UART)
* Alternativ zu USB muss das UART-Signal auf eine Stiftleiste geführt werden, und Kompatibel mit gängigen USB zu UART- Kabel sein.
* Implementierung eines einfachen Protokolls mit Getter/Setter-Funktionen

# Hardware

## Konzept

In diesem Abschnitt sollen die einzelnen Schaltungsteile analysiert und evaluiert werden. Hierbei ist darauf zu Achten, dass einige Teile der Schaltung von einem vorhergegangenen Projekt übernommen wurden und daher nicht überall neue Evaluationen durchgeführt wurden.

### Mikrocontroller

Der Mikrocontroller dient in dieser Anwendung zur Ansteuerung der einzelnen Hardwarekomponenten und soll auch die Kommunikation mit dem PC übernehmen.

#### Vorgaben

Der Kontroller muss folgende Punkte erfüllen:

* Um Daten mit dem PC auszutauschen, wird eine serielle Schnittstelle benötigt. Diese muss Hardwaremässig vorhanden sein.
* Ein Digital-Analog-Wandler (DAC) muss vorhanden sein, damit eine Stromschnittstelle realisiert werden kann. Dabei sollte eine genug hohe Auflösung (12-bit) vorhanden sein, damit sehr genaue Stromwerte eingestellt werden können.
* Es muss ein Watchdog-Timer vorhanden sein, damit eine Laufzeit von 24/7 realisiert werden kann.
* Es werden insgesamt 22 I/Os (Inputs/Outputs) benötigt um die Hardwareteile anzusteuern.
* Genügend Flash-Speicher und Arbeitsspeicher für Berechnungen mit FLOAT-Variablen und für die Verarbeitung der Kommandos des Benutzers.

Alle genannten Punkte sollen so schnell wie möglich verarbeitet werden.

#### Evaluation

Alle in Abschnitt 2.1.1.1 erwähnten Punkte werden vom STM32F303CCT6[[1]](#footnote-1) erfüllt. Dieser Mikrocontroller gehört auch zu derselben Mikrocontrollerfamilie wie der im vorhergegangenen Projekt angewendete Controller. Dies vereinfacht die Übernahme einzelner Codeausschnitte vom Vorprojekt.

Der gewählte Kontroller besitzt folgende Peripherie:

* 256 kBytes Flash-Speicher, 40 kBytes SRAM
* Bis zu 3 verwendbare UART-Schnittstellen
* 1 Digital-Analog-Converter
* 1 Window-Watchdog-Timer und 1 Independent-Watchdog-Timer
* 27 I/Os
* 8 Timer

### Clock

Der in Abschnitt 2.1.1 evaluierte Mikrocontroller besitzt einen eingebauten 8 MHz RC-Oszillator mit 16facher Phase Locked Loop Option (PLL) was eine Taktfrequenz von bis zu 72 MHz erlaubt. Die Ungenauigkeiten des RC-Oszillators sind nicht ideal für die Anwendung mit seriellen Schnittstellen. Da die Kommunikation über UART mit einer Baudrate von 115200 betrieben wird, ist die Taktfrequenz aber bei weitem genug hoch, um davon auszugehen, dass keine Probleme bei der Übertragung aufgrund der Taktfrequenz generiert werden.

### Serielle Schnittstelle

#### USB

«USB (Universal Serial Bus) ist eine serielle Schnittstelle, die sehr weit verbreitet ist und sich durch seine einfache Handhabung empfiehlt. Die USB-Schnittstelle ist eine serielle Schnittstelle, bei der die Daten differenziell über zwei verdrillte Datenleitungen übertragen werden. Alle Teilnehmer an einem Bus (Slaves) müssen sich an einem Hostkontroller anmelden. Auf diese Art können theoretisch bis zu 127 Geräte an einen Hostcontroller angeschlossen werden.»[[2]](#footnote-2)  
In diesem Fall wird die Kommunikation über USB mit Hilfe eines FTDI-Chips (FT232RL[[3]](#footnote-3)) geregelt. Dieser wandelt TTL-Signale in differentielle Signale um und regelt die Kommunikation mit dem Hostcontroller. Am PC wird das Gerät als normaler COM-Port angezeigt.

### Anzeige

#### Kommunikationsanzeige

Beim FT232RL gibt es die Möglichkeit, das Empfangen und Senden von Daten über zwei I/Os anzuzeigen. Für die Auswahl der Anzeige-LEDs (Light Emitting Diode) wurde eine Grüne und eine rote LED evaluiert. Aufgrund dessen, dass die LEDs verschiedene Lichtstärken besitzen (Rot ist stärker als Grün), werden die LEDs auch mit unterschiedlichen Strömen betrieben.

Die einzige Vorgabe, welche die 2 LEDs einhalten müssen, ist der maximale Strom von 35 mA, welcher nicht überschritten werden darf. Dies ist der maximale Strom, der vom FT232RL geliefert werden kann.

Daher wurden die folgenden LEDs evaluiert:

* Rot: SML-LXFP0603SICATR[[4]](#footnote-4)
* Grün: SML-LX0603SUGW-TR[[5]](#footnote-5)

#### Heartbeat-LED

Das Heartbeat-LED dient zur Anzeige der Funktionstüchtigkeit des Gerätes. Dieses LED soll später durch Software zum Blinken gebracht werden. Solange dieses LED blinkt, ist das Gerät in Betrieb und ist nicht softwaremässig steckengeblieben.  
Als Heartbeat LED wurde dieselbe LED verwendet, wie die rote LED beim FT232RL (Abschnitt 2.1.4.1).

### Digitale Outputs

Die digitalen Outputs dieser Hardware entsprechen grundsätzlich normalen Relais-Schaltungen. Diese Schaltungen können bereits vom vorhergegangenen Projekt übernommen werden. Bei der Relaisauswahl gab es jedoch noch eine neue Evaluation. Dies vor allem auf Grund der Grösse der vorher ausgewählten Relais.  
Für die Ansteuerung der Relais wird der ULN2803A[[6]](#footnote-6) von der vorgegangenen Schaltung übernommen. Dies ist ein IC (Integrated Circuit), welches einem Darlington-Transistor-Array entspricht. Dies vereinfacht die Ansteuerung der Relais stark. Diese ICs können ebenso wie die Relais mit 24 VDC betrieben werden.

#### Vorgaben

Die Vorgaben für die Relais sind wie folgt:

* Frittstrom 5 mA
* Möglichst klein
* Spulenspannung 24VDC
* Ausgang hält 230VAC aus

#### Evaluation

Das evaluierte Relais (Typ G6DN-1A DC24[[7]](#footnote-7)) wird bereits auf der DismessA-Steuerung verwendet. Die Angaben, welche unter Punkt 2.1.5.1 gemacht wurden, werden vom Relais alle erfüllt.

### Digitale Inputs 230VAC

Um die Signale von den Motoren und vom Alarmhorn auszuwerten, wird eine andere Schaltung, als die in Abschnitt 2.1.6 erwähnte Schaltung, angewendet. Das Signal wird durch das „Anzapfen“ der Speisung der Motoren und des Alarmhornes ausgewertet. Falls diese Eingeschalten sind, liegt eine Spannung von 230 VAC an. Diese Spannung wird auch auf das HiL-Interface geführt. Die Auswertung erfolgt über ein Relais.

#### Vorgaben

Das Relais muss folgende Angaben erfüllen:

* Frittstrom 5 mA
* Spulenspannung 230 VAC
* Galvanische Trennung

#### Evaluation

Das Ausgewählte Relais (Typ 40.31.8.230.0000[[8]](#footnote-8)) von Finder erfüllt alle in Abschnitt 2.1.7.1 vorgegebenen Angaben. Dieses Relais wurde bereits in einer DC-Version beim Simulator-Projekt eingesetzt.

### Stromschnittstelle

Die Stromschnittstelle soll komplett vom vorherigen Projekt übernommen werden. Der LM358[[9]](#footnote-9) ist kein Rail-to-Rail-Operationsverstärker, was in diesem Fall jedoch keine grosse Rolle spielt. Die Vorgaben für die Stromschnittstelle sind, dass sie mit der 24 VDC Speisung gespiesen werden soll. Da der LM358 eine Speisung von bis zu 32V aushält ist er gut für die Anwendung geeignet. Die Nichtlinearität der Bauteile wird später mit Hilfe der Software auskorrigiert.  
Die Angaben des LM358 sind:

* Versorgungsspannung: 3-32 VDC
* Ausgangsstrom pro Kanal: 30 mA
* 2-Channel
* Eingangsoffsetspannung: 3 mV

### Anschlüsse

Die Speisung, sowie die 230 VAC und die Verschiedenen Relais Inputs und Outputs, sowie auch die Stromschnittstelle müssen über Stecker mit der DismessA-Schaltung verbunden werden.  
Da für die Stecker keine speziellen Vorgaben benötigt werden, werden die Stecker, welche auf dem Simulator, sowie auch auf der DismessA-Steuerung Verwendung finden, eingesetzt.  
Angaben zum Stecker (Typ CM020PT[[10]](#footnote-10)):

* 2-Polig
* Nennstrom 11 A (Ampere)
* Nennstrom 300 V

### Speisung

Für die Speisung soll ein Schaltnetzteil eingesetzt werden, welches 230 VAC auf 24 VDC hinunterregelt. Ausserdem muss noch ein DC-DC Wandler auf die Leiterplatte montiert werden, um eine Speisung von 3.3 VDC für den uC und diverse andere Bauteile zu bieten. Für die Speisung des DismessA-Simulators (Vorarbeit) wurden bereits ein Schaltnetzteil (Typ UNO-PS/1AC/ 5DC/ 25W[[11]](#footnote-11)) sowie ein DC/DC-Wandler (Typ PG02S2403A[[12]](#footnote-12)) eingesetzt. Diese sollen auch auf dieser Schaltung eingesetzt werden. Im folgenden Abschnitt (2.1.9.1) werden jedoch noch die Berechnungen für den Strombedarf der einzelnen Spannungsquellen durchgeführt, damit überprüft werden kann, ob die Speisungen genügend Leistung zur Verfügung stellen.

#### Strombedarf

Zuerst wird die Stromaufnahme der 24 VDC Speisung überprüft. Diese liefert hauptsächlich Strom für die Relais und deren Ansteuerung. Die Speisung wurde jedoch beim Vorgängerprojekt so ausgelegt, dass die Speisung auch von anderen Geräten zusätzlich verwendet werden kann.  
Alle Teile, welche Strom von der 24 VDC Speisung beziehen, werden in der Tabelle 1 aufgeführt.

Tabelle 1: Strombedarf 24 VDC Speisung

|  |  |  |  |
| --- | --- | --- | --- |
| Anzahl | Bauteil | Strom/Bauteil | Strom Total |
| 1 | Speisung 3.3 VDC | Max. 96 mA | 96 mA |
| 14 | Relais 24 VDC | Ca. 5 mA | 70 mA |
| 1 | LM358 | Max 50 mA | 50 mA |
| 1 | Heartbeat LED | Ca. 5 mA | 5 mA |
| 2 | ULN2803A | Ca. 20 mA | 40 mA |

Die 24VDC Speisung bietet eine maximale Leistung von 30 Watt, was einem Strom von maximal 1.25 Ampere entspricht. Dies ist weitaus genügend für unsere Speisung. Zu beachten ist noch, dass die maximalen Werte aus dem Datenblatt entnommen wurden und daher zu hoch angesetzt sind. In der realen Anwendung verbrauchen die Bauteile weniger Strom. Es sollte in diesen Berechnungen jedoch der Worst-Case betrachtet werden.  
  
Bei der 3.3 VDC Speisung wird hauptsächlich vom uC Strom bezogen. Die anderen Schaltungsteile, welche 3.3 VDC benötigen, benötigen Strom im uA-Bereich, weshalb sie nicht aufgelistet wurden.  
Alle Anwendungsbereiche, welche viel Strom benötigen, sind in der Tabelle 2 aufgelistet.

Tabelle 2:Strombedarf 3.3 VDC Speisung

|  |  |  |  |
| --- | --- | --- | --- |
| Anzahl | Bauteil | Strom/Bauteil | Strom Total |
| 1 | STM32F303CCT6 | Max. 160 mA | 160 mA |

Auch die 3.3 VDC Spannungsquelle kann mit einer Leistung von 2 W genügen Strom für unser Gerät liefern.

## Realisierung

In diesem Abschnitt wird auf die Realisierung der Hardware genauer eingegangen und die einzelnen Schaltungsblöcke werden genauer erklärt.

### Blockschaltbild

Abbildung 2: Blockschaltbild

### Mikrocontroller

Der Mikrocontroller ist der Kern der Hardware und steuert alle anderen Baugruppen.

#### Speisung

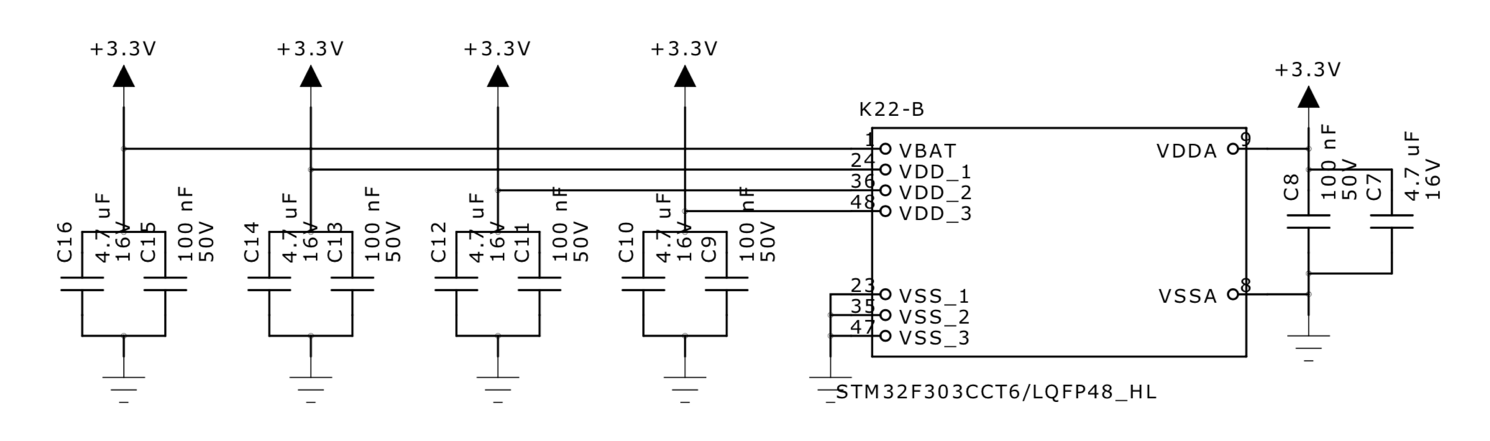
Bei der Speisung gibt es nicht viel zu beachten. Es muss lediglich vor jedem Speisungseingang ein 100 nF und ein 4.7 uF Kondensator geschaltet werden.  
Diese dienen als so genannte Abblockkondensatoren wobei der 100 nF für das Abfangen von schnellen und kleinen Störungen gedacht ist und der 4.7 uF für grössere Störungen. Da sie für das Entstören der Speisespannung zuständig sind, ist auch beim Layout darauf zu achten, dass diese Kondensatoren sehr nahe am uC platziert werden.

Abbildung 3: Speisung Mikrocontroller

#### Pin-Belegung

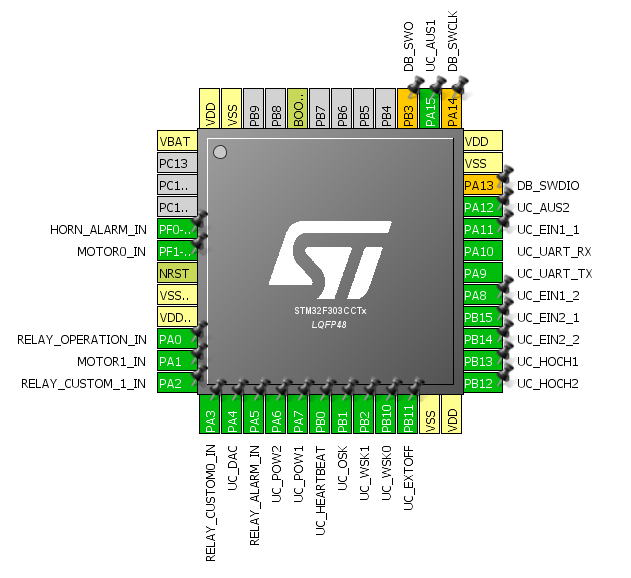
Bei der Pin-Belegung wurde Hauptsächlich in einem ersten Schritt auf die Funktion der Pins geachtet, da es Funktionen gibt, welche einen bestimmten Pin benötigen.  
Danach wurde die Pin-Belegung so angepasst, dass das Erstellen des Layouts vereinfacht wird (Anpassung erst während dem Erstellen des Layouts).

Abbildung 4: Pin-Belegung am Mikrocontroller

Tabelle 3: Pinbelegung uC

|  |  |  |
| --- | --- | --- |
| Pin nr. | bezeichnung | funktion |
| 1 | VBAT | Speisung 3.3 VDC |
| 2 | N.C. | Not Connected (Als Output konfiguriert) |
| 3 | N.C. | Not Connected (Als Output konfiguriert) |
| 4 | N.C. | Not Connected (Als Output konfiguriert) |
| 5 | HORN\_ALARM\_IN | Input: Liest ein, ob das Alarmhorn eingeschaltet ist oder nicht. |
| 6 | MOTOR0\_IN | Input: Liest ein, ob Motor 0 eingeschaltet wurde oder nicht. |
| 7 | RESET | Resettet den Controller und wird ebenfalls für die Programmierung verwendet. |
| 8 | VSS | GND |
| 9 | VDD | Speisung 3.3 VDC |
| 10 | RELAY\_OPERATION\_IN | Input: Liest ein, ob das „Operation“-Relais eingeschaltet ist oder nicht. |
| 11 | MOTOR1\_IN | Input: Liest ein, ob Motor1 eingeschaltet ist oder nicht. |
| 12 | RELAY\_CUSTOM1\_IN | Input: Liest ein, ob das „Custom1“-Relais eingeschaltet ist oder nicht. |
| 13 | RELAY\_CUSTOM0\_IN | Input: Liest ein, ob das „Custom0“-Relais eingeschaltet ist oder nicht. |
| 14 | UC\_DAC | Digital-Analog-Converter-Ausgang für die 4-20 mA Schnittstelle |
| 15 | RELAY\_ALARM\_IN | Input: Liest ein, ob das „Alarm“-Relais eingeschaltet ist oder nicht. |
| 16 | UC\_POW2 | Output: Schaltet Relais 2 für die DismessA-Speisung ein/aus |
| 17 | UC\_POW1 | Output: Schaltet Relais 1 für die DismessA-Speisung ein/aus |
| 18 | UC\_HEARTBEAT | Output: Heartbeat-LED blinkt im Betrieb |
| 19 | UC\_OSK | Output: Schaltet das OSK-Relais ein/aus |
| 20 | UC\_WSK1 | Output: Schaltet das WSK1-Relais ein/aus |
| 21 | UC\_WSK0 | Output: Schaltet das WSK2-Relais ein/aus |
| 22 | UC\_EXTOFF | Output: Schaltet das EXTOFF-Relais ein/aus |
| 23 | VSS | GND |
| 24 | VDD | Speisung 3.3 VDC |
| 25 | UC\_HOCH2 | Output: Relais 2 für die Simulation des Schwimmerschalters HOCH |
| 26 | UC\_HOCH1 | Output: Relais 1 für die Simulation des Schwimmerschalters HOCH |
| 27 | UC\_EIN2\_2 | Output: Relais 2 für die Simulation des Schwimmerschalters EIN2 |
| 28 | UC\_EIN2\_1 | Output: Relais 1 für die Simulation des Schwimmerschalters EIN2 |
| 29 | UC\_EIN1\_2 | Output: Relais 2 für die Simulation des Schwimmerschalters EIN1 |
| 30 | UC\_UART\_TX | Sendet Daten via FT232RL an PC |
| 31 | UC\_UART\_RX | Empfängt Daten via FT232RL von PC |
| 32 | UC\_EIN1\_1 | Output: Relais 1 für die Simulation des Schwimmerschalters EIN1 |
| 33 | UC\_AUS2 | Output: Relais 2 für die Simulation des Schwimmerschalters AUS |
| 34 | DB\_SWDIO | Bidirektionaler Daten-Pin für die Programmierung |
| 35 | VSS | GND |
| 36 | VDD | Speisung 3.3 VDC |
| 37 | DB\_SWCLK | Clock-Signal für die Programmierung des uC |
| 38 | UC\_AUS1 | Output: Relais 1 für die Simulation des Schwimmerschalters AUS |
| 39 | DB\_SWO | Serial Wire Output |
| 40 | N.C. | Not Connected (Als Output konfiguriert) |
| 41 | N.C. | Not Connected (Als Output konfiguriert) |
| 42 | N.C. | Not Connected (Als Output konfiguriert) |
| 43 | N.C. | Not Connected (Als Output konfiguriert) |
| 44 | BOOT0 | Connected to GND |
| 45 | N.C. | Not Connected (Als Output konfiguriert) |
| 46 | N.C. | Not Connected (Als Output konfiguriert) |
| 47 | VSS | GND |
| 48 | VDD | Speisung 3.3 VDC |

#### Programmierschnittstelle

Für die Programmierung des Mikrocontrollers wird die SWD-Programmierschnittstelle eingesetzt. Die Beschaltung wurde von der DismessA-Steuerung übernommen. Die SWD-Programmierschnittstelle bietet den Vorteil, dass der Controller direkt auf der Hardware Programmiert werden kann. Als Programmer dient der Segger J-Link.

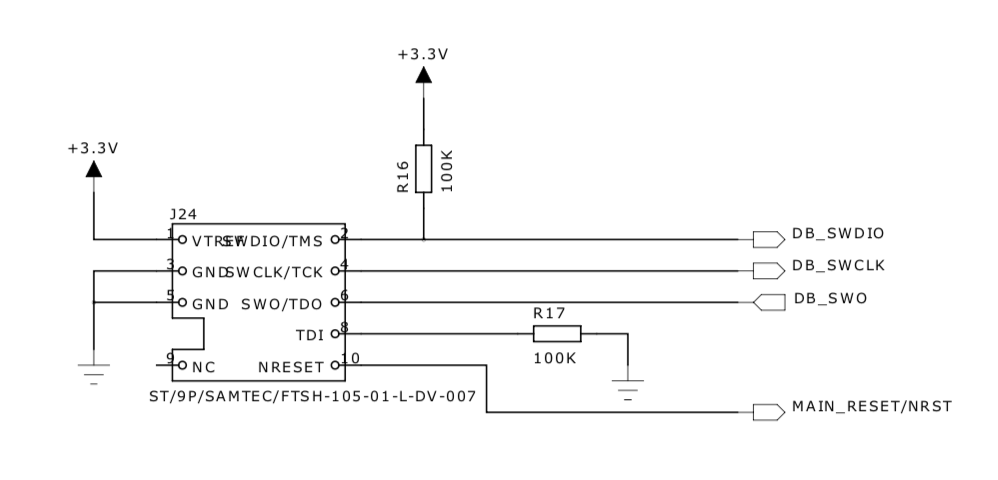
Verbunden mit dem Controller wird der Programmer via 10-poliger zweireihiger Stiftleiste. Für das spätere Testen der Software zu vereinfachen wird zusätzlich noch ein Taster eingebaut, welcher einen manuellen Reset des Controllers ermöglicht.

Abbildung 5: SWD-Programmierschnittstelle

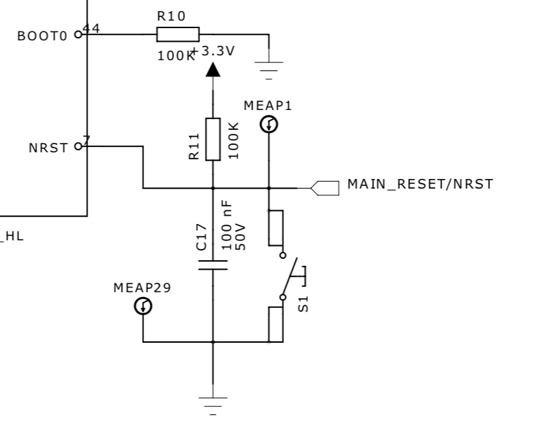


Abbildung 6: Reset-Taster

### Serielle Schnittstelle

Für die Kommunikation über den USB-Anschluss wird wie bereits erwähnt der FT232RL[[13]](#footnote-13) verwendet. Die Beschaltung wurde aus dem Datenblatt entnommen und entspricht der „Typical Application“ mit der „USB to MCU UART Interface“ Konfiguration. Es gibt jedoch kleine Abweichungen zu der im Datenblatt beschriebenen Schaltung. Die CTS- und RTS-Leitungen (Flow-Control) wurden nicht verbunden, da sie nicht benötigt werden. Auch CLKIN und PWREN wurden nicht, wie im Datenblatt beschrieben, verwendet. Man kann ausserdem erkennen, dass sich vor dem USB-Stecker noch eine TVS-Diode (Typ SN65220DBV[[14]](#footnote-14)) befindet.

Diese ist für USB-Schnittstellen geeignet und soll den FT232RL vor Überspannung schützen, welche über den USB-Stecker übermittelt werden könnte.

Um die Datenübermittlung visuell sichtbar zu machen, was das Debugging vereinfacht, wurden noch 2 LEDs mit eingebaut. Diese werden vom FT232RL selbst gespiesen und der gemeinsame Strom darf 35 mA nicht überschreiten. Da die LEDs nicht sehr hell leuchten müssen, da sie nicht zu beleuchtungszwecken verwendet werden, wird mit einem Strom von 5 mA gerechnet.

Berechnung der Vorwiderstände für die Signal-LEDs:

Rote LED[[15]](#footnote-15):

Grüne LED[[16]](#footnote-16):

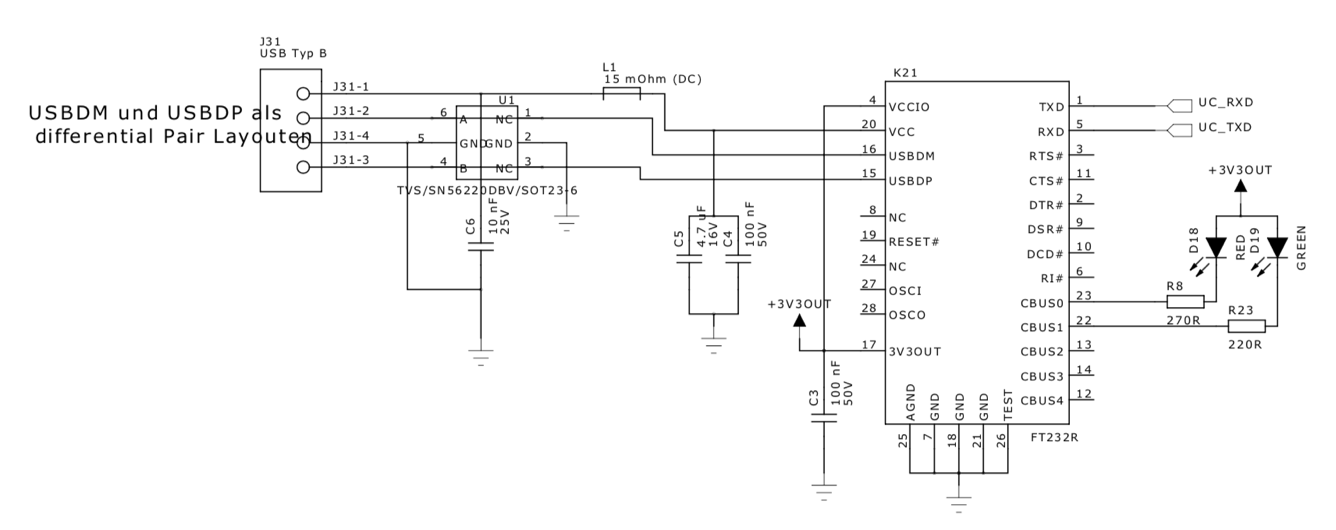
Da die grüne LED, bei gleichem Strom, dunkler als die rote LED leuchten wird, wurde der Widerstand nach unten abgerundet, damit ein wenig mehr Strom fliesst. Die Widerstandswerte entsprechen er E12-Reihe.

Abbildung 7: Beschaltung FT232RL

Das Rx und Tx Signal wurde ausserdem noch über eine Stiftleiste zugänglich gemacht. Im Falle, dass der FT232RL zerstört würde, könnte man über die Stiftleiste (dessen Pin-Out mit gängigen USB zu UART Kabel übereinstimmt) immer noch mit der Hardware kommunizieren. Auch dort sind TVS-Dioden (Typ SMF05C[[17]](#footnote-17)) zum Schutz der Schaltung mit eingebaut.

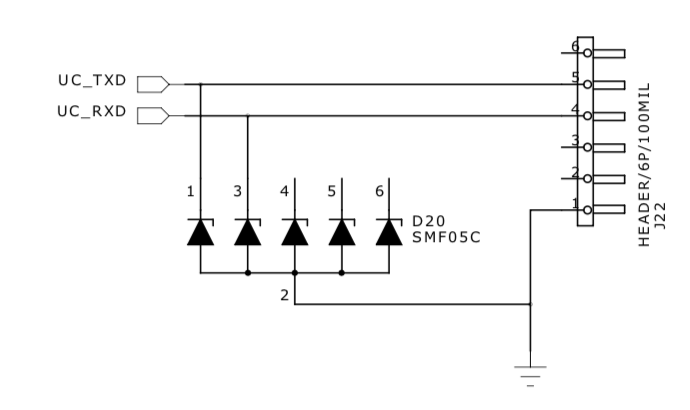


Abbildung 8: Beschaltung der Stiftleiste

**Hinweis:** Beim Layouten der USB-Verbindungen ist darauf zu achten, dass USBDM und USBDP als differential Pair gelayoutet werden. Das heisst sie müssen dicht nebeneinander verlaufen und die Leiterbahnlängen sollten so identisch wie möglich sein.

### Inputs

#### 230 VAC Spannungsauswertung

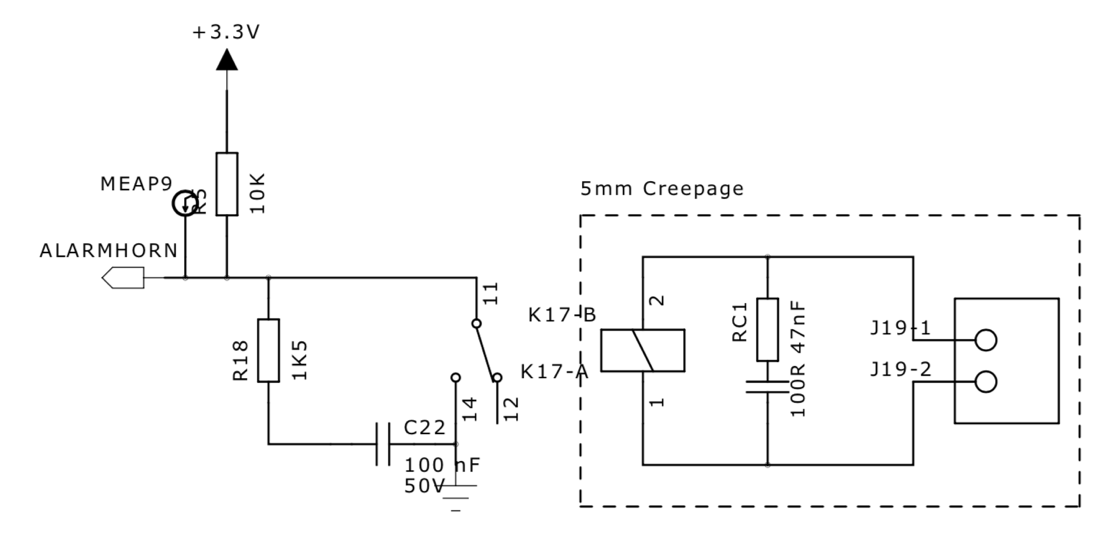
Bei der Auswertung der 230 VAC, welche beispielsweise von den Motoren kommen, wird ein Relais eingesetzt. Dieses Relais (Typ 40.31.8.230.0000[[18]](#footnote-18)) ist für 230 VAC ausgelegt und Schaltet auf der Schaltkontaktseite einen 10 kOhm auf GND, wenn Spannung anliegt. Das heisst, wenn keine Spannung Anliegt, liegt der Einlese-Pin am uC auf 3.3 VDC. Schaltet das Relais wird dieser Pin auf GND gezogen. Im Schema ist ersichtlich, dass parallel zum Relais ein Snubber-Glied (R und C) geschaltet ist.

Abbildung 9: 230 VAC Spannungsauswertung

Dieses Snubber-Glied soll die Spannungsspitzen, welche beim Ausschalten des Relais entstehen können, neutralisieren, damit keine Bauteile zerstört werden. Auf der Seite des Einlese-Pins ist auch ein Snubber-Glied dazu geschaltet. Dort dient es dazu, den Schaltkontakt des Relais zu entprellen.

#### Relais einlesen

Das Einlesen der Relais erfolgt mit Hilfe eines 10 kOhm Widerstandes. Solange der Schaltkontakt des Relais nicht durchkontaktiert ist, ist das Eingangssignal am uC 3.3 VDC. Sobald das Relais Durchgeschaltet wird, wird der Pin am uC auf GND gezogen. Das Snubber-Glied, welches parallel zum Signalstecker geschaltet ist, dient dazu den Schaltkontakt des Relais zu entprellen.

### Outputs

Abbildung 10: Einlesen der Relais

#### Relais-Ansteuerung

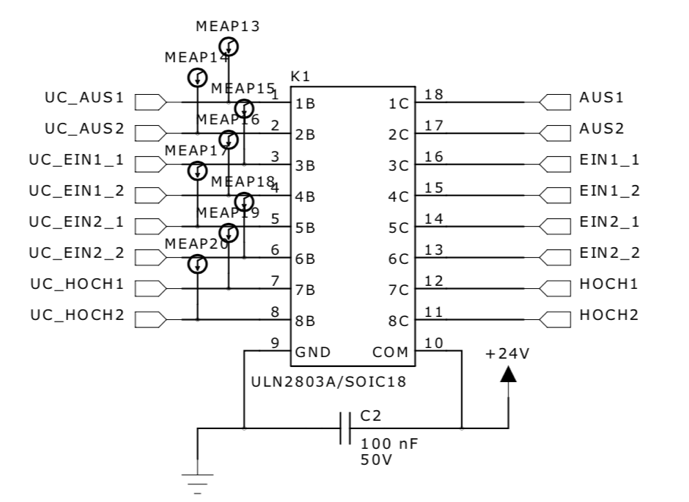
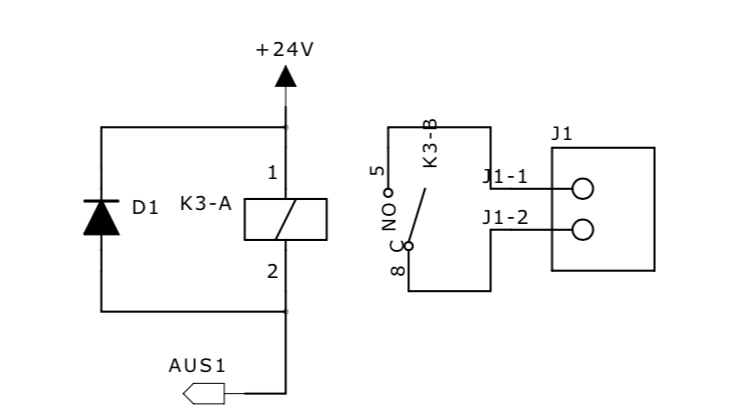
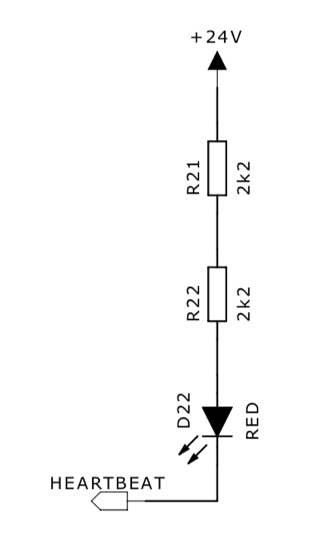
Die Ansteuerung der Relais erfolgt über den ULN2803A[[19]](#footnote-19). Dies ist ein Darlington-Transistor-Array mit total 8 Ein- und Ausgängen. Die Ausgänge werden Low-Aktiv betrieben. Um die Relais durchzuschalten muss der Eingang auf 3.3 VDC gesetzt werden, was intern die Darlington-Transistor-Schaltung durchschaltet und das Relais auf GND geschaltet. Der ULN2803A besitzt intern bereits eingebaute Freilaufdioden. Im Schema ist zu erkennen, dass trotzdem noch Freilaufdioden eingebaut wurden. Diese sind lediglich zur Absicherung da und müssen schlussendlich nicht zwingend aufgelötet werden. Die Freilaufdiode soll verhindern, dass die beim Ausschalten eines Relais entstehenden Spannungsspitzen den ULN2803A zerstören.

Abbildung 11: Beschaltung ULN2803A

Abbildung 12: Relais-Schaltung

#### Heartbeat-LED

Die Ansteuerung der Heartbeat-LED wird auch über den ULN2803A geregelt. Daher entspricht die Schaltung fast der, der Relais. Die LED wird ebenso wie die Relais Low-Aktiv betrieben und wird durch das Schalten von 3.3 VDC an den Eingang durchgeschaltet. Da die gleiche rote LED wie die LED beim FT232RL verwendet wird, können die Werte übernommen werden. Die LED wird allerdings an 24 VDC mit 5 mA betrieben.

Berechnung des Vorwiderstandes der Heartbeat-LED:

Abbildung 13: Heartbeat-LED

Wie man im Schema sieht wurde der Vorwiderstand auf 2 Vorwiderstände aufgeteilt. Dies wurde aufgrund der Leistung, welche die Widerstände aushalten, so geregelt. Diese halten lediglich eine Leistung von 100 mW aus, was genau der Leistung entspricht, welche über einen 4.4 kOhm Widerstand wirken würde. Durch die Aufteilung auf 2 Widerstände kann die Leistung auf 50 mW pro Widerstand reduziert werden.

#### 4-20 mA Schnittstelle

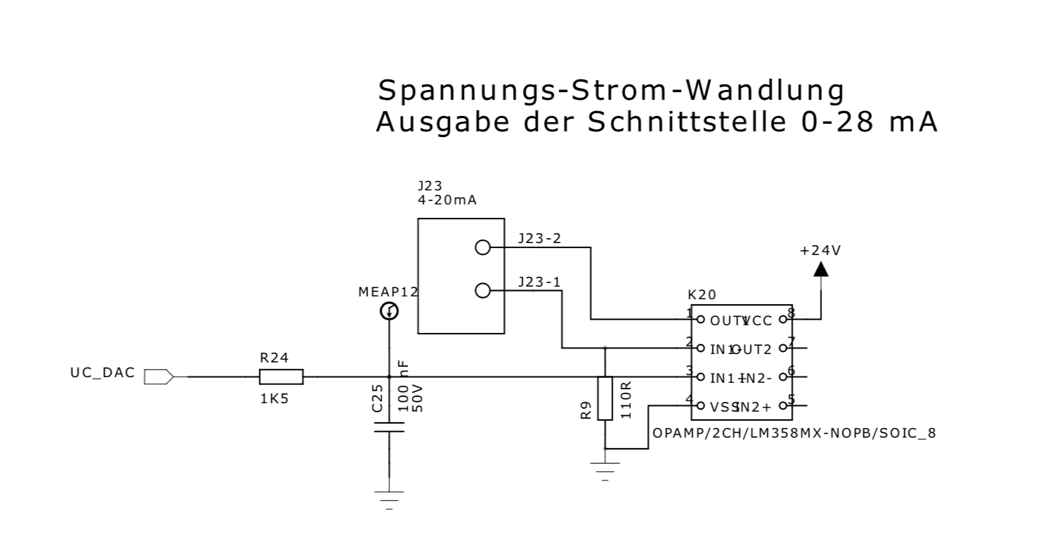
Die 4-20 mA Schnittstelle wurde vom Vorgängerprojekt übernommen. Daher wird der LM358[[20]](#footnote-20) verwendet. Die Schaltung für die Stromschnittstelle entspricht der eines Nicht-Invertierenden-Verstärkers.

Abbildung 14: 4-20 mA Schnittstelle

Der Lastwiderstand entspricht bei der DismessA-Steuerung 150 Ohm und wird zwischen den Ausgang und den negativen Eingang des Verstärkers geschaltet. Der Strom welcher durch den Lastwiderstand fliesst wird durch den Widerstand R9 bestimmt, denn die Spannung, welche vom DAC des uC ausgegeben wird (0 – ca. 3.3 VDC) herrscht auch über dem Widerstand R9.   
Die Stromschnittstelle wurde so ausgelegt, dass nicht 4-20 mA sondern 0-30 mA möglich sind. Später wird dann die Linearität der Stromkurve und die Strombegrenzung von 0-25 mA durch die Software geregelt.  
Berechnung R9:

### Anschlüsse

Die Anschlüsse der Speisung, der 230 VAC Relais, der 24 VDC Relais, der Programmierschnittstelle und der Stiftleiste sind alle bereits in den einzelnen Schaltungen ersichtlich und benötigen aus diesem Grund keine spezielle Erklärung.

### Speisung

Die Schaltung der Speisung ist einfach erklärt. Die 24 VDC Spannung wird über einen Stecker auf die Schaltung geführt, wo sie dann als Speisung für den DC/DC-Wandler sowie auch für die Relais und diverse andere Bauteile dient. Bei der Beschaltung des PG02S2403A[[21]](#footnote-21) ist darauf zu achten, dass der Remote-Pin auf GND gehängt ist. Ansonsten funktioniert der DC/DC-Wandler nicht. Vor Empfindlichen Bauteilen wie beispielsweise dem uC oder auch dem ULN2803A wurden jeweils 100 nF Kondensatoren als Abblockkondensatoren in die Speisung gehängt.

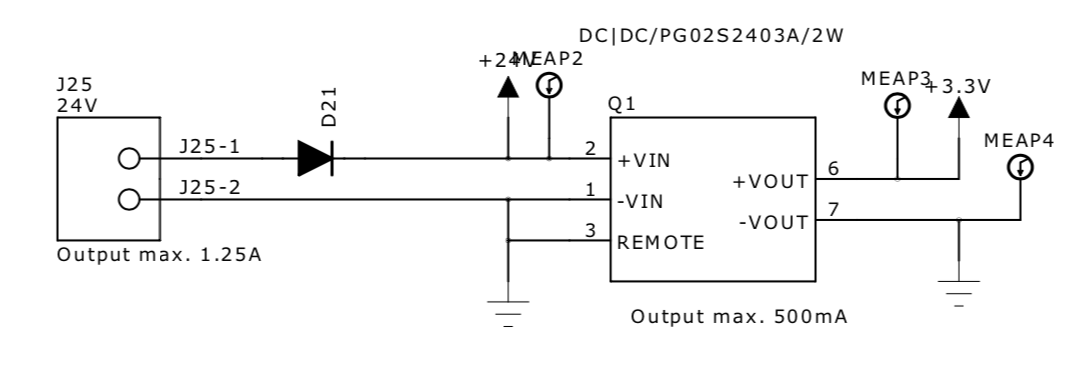


Abbildung 15: Speisung

# Software

In diesem Abschnitt der Dokumentation sollen die einzelnen Teile der Firmware analysiert und erklärt werden.

## Firmware

### Aufbau

Die Firmware für das HiL-Interface wurde modular strukturiert. Das bedeutet, dass jede Peripherie ein eigenes Modul darstellt. Die einzelnen Module entsprechen eigenen C-Files und können dann im Programm mit Hilfe der Headerdateien eingebunden werden. Diese Module enthalten alle Funktionen, welche für die Ansteuerung der jeweiligen Peripherie benötigt werden. Lediglich alle Inputs wurden zusammengefasst in ein C-File, da dies das Einlesen wesentlich vereinfacht.

### Variablen/Defines

Damit die Software übersichtlicher bleibt wurden die Variablen nach einem bestimmten Prinzip benannt. Alle Variablen und Defines eines Moduls beginnen immer mit dem Modulnamen. Nur die Variablen, welche nur in einer Funktion und nicht im ganzen Modul verwendet werden, müssen dieser Syntax nicht folgen.  
Alle Variablen und Defines wurden auf Englisch benannt.  
Beispiele:

* mod\_dac\_current
* MOD\_DAC\_START\_VALUE

### Ablauf

Der Softwareablauf entspricht grundsätzlich dem Modell eines Flussdiagramms. Dies liegt daran, dass solange keine Befehle über UART empfangen werden, das Programm immer denselben Ablauf durchführt. Erst sobald ein Interrupt von der UART-Schnittstelle ausgelöst wird, wird das Parsing des erhaltenen Befehls durchgeführt. Aus diesem Grund wird die Software mit Hilfe eines Flussdiagramms und eines Sequenzdiagramms dargestellt. Der Ablauf des Parsing wird zu einem späteren Zeitpunkt noch durch ein State-Event-Diagramm erläutert.

#### Erklärung Flussdiagramm

In Abbildung 16 wird das Flussdiagramm des HiL-Interfaces dargestellt. Sobald das HiL-Interface gestartet wird, so wird die Hardware zuerst initialisiert. Das bedeutet, dass alle Outputs in ihren Startzustand versetzt werden, sowie dass die UART-Schnittstelle und auch der Watchdog-Timer und der Digital-Analog-Converter initialisiert werden. Danach beginnt der «Loop» des Flussdiagramms. Das Programm wird die im «Loop» enthaltenen «Aufgaben» so lange lösen, bis das Gerät neu gestartet wird oder ein Interrupt geschieht. Der Loop besitzt ein Delay von 1 ms. Dies ist auf der einen Seite nötig wegen der Entprellung der Inputs und andererseits für das Hochzählen der verschiedenen Counter.

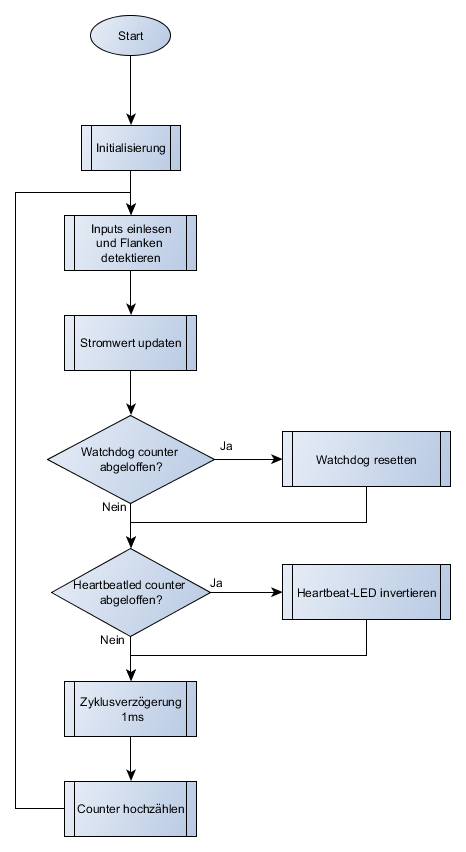


Abbildung 16: Flussdiagramm

Das Einlesen der Taster wird später, wenn die einzelnen Module genauer erklärt werden, genauer betrachtet. Interessant sind vor allem der Watchdog-Counter und der Heartbeat-LED-Counter.

#### Heartbeat-LED-Counter

Der Heartbeat-LED-Counter ist, wie aus dem Flussdiagramm zu entnehmen ist, ein Zähler, welcher dazu da ist, die Heartbeat-LED zum Blinken zu bringen. Die Heartbeat-LED signalisiert dem Benutzer, dass das HiL-Interface in Betrieb ist und dass die Software nicht hängengeblieben ist. Die Blinkfrequenz des Heartbeat-LEDs muss also nicht extrem genau sein und beträgt 5 Hz. Dies wird realisiert, indem der Counter auf 100 zählt (=100 ms) und danach das Heartbeat-LED invertiert.  
Zählt man die aus- und die eingeschaltete Zeit zusammen erhält man 200 ms als Periodendauer (T).

#### Watchdog-Counter

Der Watchdog-Counter soll verhindern, dass der Watchdog-Timer abläuft und somit der Mikrocontroller neu gestartet wird. Beim Window Watchdog Timer, welcher in dieser Firmware eingesetzt wird, ist die Berechnung für das Counter-Timeout jedoch etwas komplizierter. Denn der Watchdog Timer muss innerhalb eines bestimmten Zeitfensters zurückgesetzt werden, damit kein Fehler entsteht und der Mikrocontroller zurückgesetzt wird.

Berechnung des Zeitfensters:

= 32768

WDGTB = Prescaler = 3

Counter Value = 0x7F => Reset value = 0x7F-0x40 = 0x3F

(0x3F+1) = 32.768 ms

tRefreshMin = 32.768 ms – 8.704 ms = 24.064 ms

Die Berechnungen ergeben, dass das Zeitfenster zwischen 24.064 ms und 32.768 ms liegt. Wird der Watchdog Timer vorher oder nach diesem Zeitfenster zurückgesetzt, so wird der Mikrocontroller zurückgesetzt. Aus diesem Grund Zählt der Watchdog-Counter bis 25 (=25 ms) damit der Timer rechtzeitig zurückgesetzt werden kann.  
Falls das Programm hängen bleibt, läuft der Timer ab und die Firmware wird neu gestartet. Dies ermöglicht das Einhalten einer Laufzeit von 24 Stunden 7 Tage lang.

### Module

In diesem Kapitel werden die einzelnen Module der Firmware genauer analysiert und erklärt. Es werden jedoch nur die Module genauer betrachtet, welche Berechnungen oder komplizierten Code enthalten. Die restlichen Module werden kurz erklärt.

#### Config

Das Modul Config besteht lediglich aus einem Header-File. Dieses enthält einige Angaben, die vom Benutzer einfach und schnell angepasst werden können. Beispielsweise wird dort die Versionsangabe gemacht.

#### DAC

Das Digital-Analog-Converter-Modul ist eines der Anspruchsvolleren Module. Dies nicht aufgrund dessen, dass es viele Funktionen beinhalten würde, sondern weil darin die Strom-Linearisierung geregelt wird. Bevor jedoch die Linearisierung erfolgt, wird der eingegeben Stromwert in einen Digitalen Wert umgewandelt, der diesem Strom entspricht.  
Betrachten wir zuerst die Linearisierung.  
Um die Stromkurve zu linearisieren wurde zuerst eine Messreihe erstellt und in ein Excel-Dokument[[22]](#footnote-22) geschrieben. Dabei wurden die Daten in Abbildung 17 gemessen.

Der Soll-Wert wird berechnet, indem man den maximalen Strom durch die maximale Auflösung des DAC rechnet und diesen Wert mit dem eingestellten DAC-Wert multipliziert. Beispiel:

Abbildung 17: Messresultate Stromschnittstelle (nicht-linearisiert)

Wie man sehen kann, ist die gemessene Ist-Stromkurve viel steiler als die Soll-Kurve. In einem nächsten Schritt wird die Differenz jedes Messpunktes in einen Korrekturwert umgerechnet. Dazu muss zuerst die Differenzspannung berechnet werden.

=>

Alle Differenzspannungen werden in einen Digitalen Korrekturwert umgerechnet.

Mit diesen Korrekturwerten kann nun eine Korrekturfunktion gezeichnet werden. Wichtig ist hierbei, dass die Werte DAC-Werte unter 200 und über 4000 nicht mit in die Linearisierungsfunktion genommen werden, da sie, wie man in Abbildung 17 gut sehen kann, extrem nichtlinear sind.

Abbildung 18: Linearisierungsfunktion

Nun kann man in Excel bei der Funktion eine «Trendlinie» hinzufügen, wodurch die mathematische Funktion angegeben wird. Diese Funktion muss nun zum ursprünglich eingegebenen DAC\_Wert hinzugerechnet werden. Man erhält also die Korrekturfunktion:

663\*DACwert+1.9906

Nun kommen wir noch zur Umwandlung des ursprünglich eingegebenen Stromwertes in den DAC\_Wert, welcher dann mit der Korrekturfunktion korrigiert wird. Um die Umrechnungsfunktion zu erhalten, wurde wieder eine Testreihe durchgeführt, wobei immer ein bestimmter Stromwert eingestellt wurde und der dazu eingegeben DAC\_Wert in das Excel-Dokument gespeichert wurde. Abbildung 19 zeigt die erhaltene Umrechnungsfunktion.  
Der Stromwert wird also zuerst mit dieser Umrechnungsfunktion in einen digitalen DAC\_Wert umgerechnet und danach noch durch die Strom-Linearisierung auskorrigiert.

Abbildung 19: Umrechnungsfunktion

#### Levelswitch

Das Levelswitch-Modul besteht grundsätzlich lediglich aus einer Getter- und einer Setter-Funktion. Diese Funktionen ermöglichen die einzelnen Levelswitches in die 4 Zustände (Inaktiv, Aktiv, Kabel und Schalter) zu setzen und auch diesen Zustand auszulesen.

#### Parse

Wie bereits unter Punkt 3.1.3 erwähnt, wird da Parsen anhand eines State-Event-Diagramms erklärt. Dies ist nicht zu 100 Prozent korrekt, denn das State-Event-Diagramm wird eigentlich nur verwendet, wenn es zu State-Wechseln kommt. Diese Wechsel werden durch Events (zum Beispiel Tastendruck) ausgelöst. Dies ist im Fall des Parsens nicht ganz Korrekt, denn es gibt keine State-Wechsel oder Events. Jedoch eignet sich diese Art von Diagramm für eine Erklärung am meisten. In Abbildung 20 ist das State-Event-Diagramm vom Parsing ersichtlich.

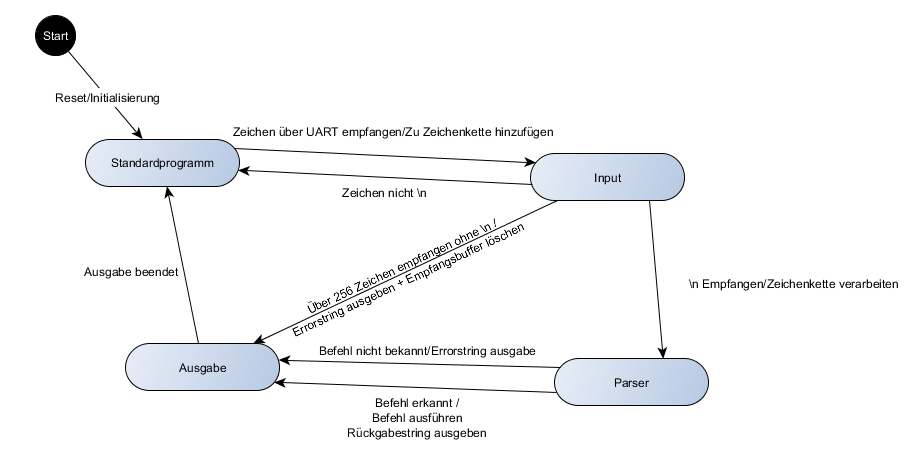
Die Funktionen des Parse-Moduls werden lediglich dann aufgerufen, wenn Daten über die UART-Schnittstelle empfangen werden. Sobald dies geschieht, wechselt das Programm vom «Standartprogramm» zum State «Input». Entspricht das Empfangene Zeichen nicht Line Feed (\n), so wechselt das Programm wieder zurück zum Standartprogramm. Werden über 256 Zeichen empfangen bevor Line Feed empfangen wird, so wird der Empfangs-Buffer gelöscht und eine Fehlermeldung ausgegeben. Sobald Line Feed empfangen wurde, wird der Befehl im State «Parser» verarbeitet. Ist der Befehl nicht bekannt, wird eine Fehlermeldung ausgegeben. Ansonst wird der Befehl Ausgeführt und ein Rückgabestring an den PC gesendet.

Abbildung 20: State-Event-Diagramm Parsing

##### Befehls-Syntax

Damit das HiL-Interface die Befehle erkennen kann, müssen diese einer bestimmten Syntax folgen. Dieser ist in Abbildung 21 zu sehen.

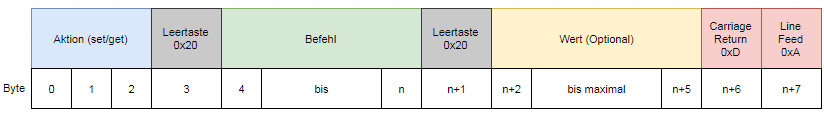


Abbildung 21: Befehls-Syntax

Die Eingabe der Befehle muss der Syntax entsprechen und die totale Zeichenlänge (inklusive Carriage Return und Line Feed) darf nicht 256 Zeichen übersteigen, da sonst der Empfangs-Buffer auf dem HiL-Interface gelöscht wird.  
Beispielbefehle:

* set power 0\r\n
* set level\_switch1 3\r\n

Ausnahme von der Befehlssyntax gibt es nur für die Befehle «help», «version» und «reset». Diese benötigen keine Set-/Get-Aktion oder einen Übergabewert.

Beispielbefehl:

* help\r\n
* version\r\n

Eine komplette Liste mit allen Befehlen befindet sich auf den Seiten 9 und 10 im Pflichtenheft[[23]](#footnote-23).

#### Power

Das Modul Power besitzt wenige Funktionen. Es beinhaltet eine Funktion um die Speisung der DismessA-Steuerung ein- und auszuschalten und eine Funktion, um den Zustand der Speisung auszulesen.

#### Resources

Das Resource-Modul besteht wie das Config Modul nur aus einem Header-File und beinhaltet alle Rückgabewerte, die das HiL-Interface ausgeben kann. Die Rückgabewerte werden direkt als String gespeichert, damit sie für die Übertragung bereits das richtige Format besitzen.

#### Switch

Das Modul Switch hat auch nur die Getter/Setter-Funktionen wie das Modul Power, aber es Steuert EXTOFF, WSK0+1 und OSK.

#### UART

Das UART-Modul besteht nur aus der init-Methode und zwei Funktionen. Zum einen enthält das Modul eine Callback Funktion für das Empfangen von Daten. Diese wird aufgerufen, sobald Daten über die UART-Schnittstelle empfangen wurden. Die Funktion startet ausserdem das Parsing.  
Die zweite Funktion ist eine simple Transmit-Funktion, welche den übergebenen String über UART sendet.

#### Update

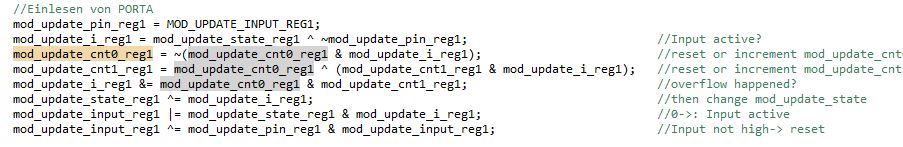
Im Modul Update werden alle Inputs eingelesen. Das heisst die Entprellung wird ebenfalls durchgeführt. Diese wollen wir etwas genauer betrachten. In Abbildung 22 wird die Entprellung der Eingänge dargestellt.

Abbildung 22: Entprellung

Die Entprellung funktioniert einfacher, als es aussieht. Der in Abbildung 22 dargestellte Code entspricht einem 2-Bit-Counter wobei die einzelnen Bit auf 2 verschiedene Variablen verteilt sind. Da an den Eingangs-Pins des Mikrocontrollers bei durchgeschaltetem Relais eine 0 eingelesen wird, sind zu Beginn alle Variablen auf 0xFFFF gesetzt. Sobald nun beispielsweise am Eingangs-Pin1 eine 0 erscheint wird das Bit 1 in der Variable cnt0 auch auf 0 gesetzt. Ist beim nächsten durchlauf dieser Eingang immer noch auf 0, so wird cnt0 wieder auf 0xFFFF gesetzt und bei cnt1 wird das Bit 1 auf 0 gesetzt. Folgt noch einmal eine 0 an demselben Eingang, so wird auch bei cnt0 das Bit 1 auf 0 gesetzt. Ist auch beim vierten Durchlauf derselbe Pin auf 0, so gibt es einen überlauf. Beide count-Variablen werden auf 0xFFFF gesetzt und bei der Variable update\_state wird das Bit 1 auf 0 gesetzt. Diese Variable kann dann verwendet werden, um auszulesen ob ein Eingang high oder low ist. Für die Entprellung werden also insgesamt 4 Durchläufe benötigt, was 4 ms entspricht, da die Entprellfunktion einmal pro Sekunde aufgerufen wird. Für das Löschen des High-Zustandes aus der update\_state-Variable werden ebenfalls wieder 4 ms benötigt. Wird während dem Entprellvorgang (beispielsweise beim dritten Durchlauf) am Eingangs-Pin wieder eine 1 eingelesen, so werden die count-Variablen zurückgesetzt und der Entprellvorgang muss von vorne begonnen werden.

## GUI

Für Demonstrationszwecke wurde ein GUI (Graphical User Interface) erstellt. Dieses GUI dient dazu, alle Funktionen der Hardware abzudecken. Die Software wurde mit Hilfe des Editors „Visual Studio 2017“ in der Programmiersprache C# programmiert.

### Aufbau

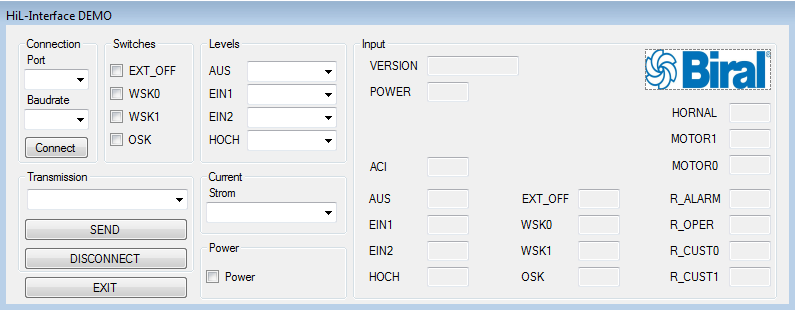
Da die Software rein zu Demonstrationszwecken verwendet wird, soll diese möglichst einfach und übersichtlich gestaltet sein. Die bedienende Person sollte die Software ohne grosse Anweisungen selber steuern können.

Abbildung 23: GUI-Aufbau

Wie in Abbildung 23 ersichtlich ist, ist das GUI in 7 Hauptteile unterteilt.

* Connection Dient zum Verbinden mit dem HiL-Interface
* Switches Steuert WSK0+1, OSK und EXTOFF
* Levels Einstellen der Levelswitch-Zustände
* Transmission Übertragen von vorbestimmten Befehlen und Beenden der   
   Verbindung mit dem HiL-Interface
* Current Einstellen des Stromes für die 4-20 mA Schnittstelle
* Power Steuern der Speisung der DismessA-Steuerung
* Input Darstellen der Rückgabewerte des HiL-Interfaces

Hinzu kommt noch ein EXIT-Button, der für das Beenden der Software zuständig ist.

### Ablauf

Die Programmierung auf einem PC unterscheidet sich sehr stark von der Programmierung eines Mikrocontrollers. Das Programm hat keinen bestimmten Programmablauf und führt Aktionen nur nach einem „Event“ aus. Ein Event entspricht hierbei der Eingabe durch einen Benutzer oder das Ablaufen eines erstellten Timers.  
Die Aufgabe des GUIs ist es, die vom Benutzer eingegebenen Befehle an das HiL-Interface zu senden und die erhaltenen Daten visuell darzustellen.

#### Timer

In der Software wurden insgesamt 2 Timer eingebaut, welche Relevant für die Funktion des GUIs sind. Aus diesem Grund werden die Timer und ihre Funktion in diesem Abschnitt kurz erläutert.  
Die Abbildung 24 zeigt die einzelnen «Events» des GUIs. Sie sind nach dem anzunehmenden Softwareablauf angeordnet. Zu beachten ist auch Abbildung 25. Diese zeigt die Verständigung zwischen dem Hauptthread und dem Nebenthread, welcher für das Senden und Empfangen über die serielle Schnittstelle zuständig ist.

**Update-Timer**

Der Update-Timer wird nur verwendet solange eine Verbindung mit dem HiL-Interface hergestellt ist. Das heisst, er wird eingeschaltet sobald der Button «CONNECT» gedrückt wurde und wieder ausgeschaltet, wenn «DISCONNECT» oder «EXIT» gedrückt wird.

Danach werden alle 50 ms Teile der Hardwarezustände abgefragt. Da das Aktualisieren der Zustände aller Hardwarekomponenten auf einmal die GUI-Software blockieren würde und somit einen flüssigen Ablauf der Software verhindert wird, müssen die Abfragen auf mehrere Blöcke aufgeteilt werden. Beispielsweise werden also immer nach 50 ms zwei andere Hardwarezustände abgefragt.

**Reception-Timer**

Der Reception-Timer dient dazu einen Timeoutfehler auszugeben, wenn zu viel Zeit vergeht ohne dass Daten empfangen wurden. Der Timer funktioniert so dass immer, wenn etwas über die serielle Schnittstelle (UART) gesendet wird, wird der Timer aktiviert.

Wird die Antwort des HiL-Interface innerhalb von 50 ms empfangen, so wird der Timer wieder deaktiviert und die Software arbeitet weiter. Sollte es jedoch dazu kommen, dass innerhalb der 50 ms nichts empfangen wird, so wird eine Fehlermeldung an den Benutzer ausgegeben, welche quittiert werden muss.

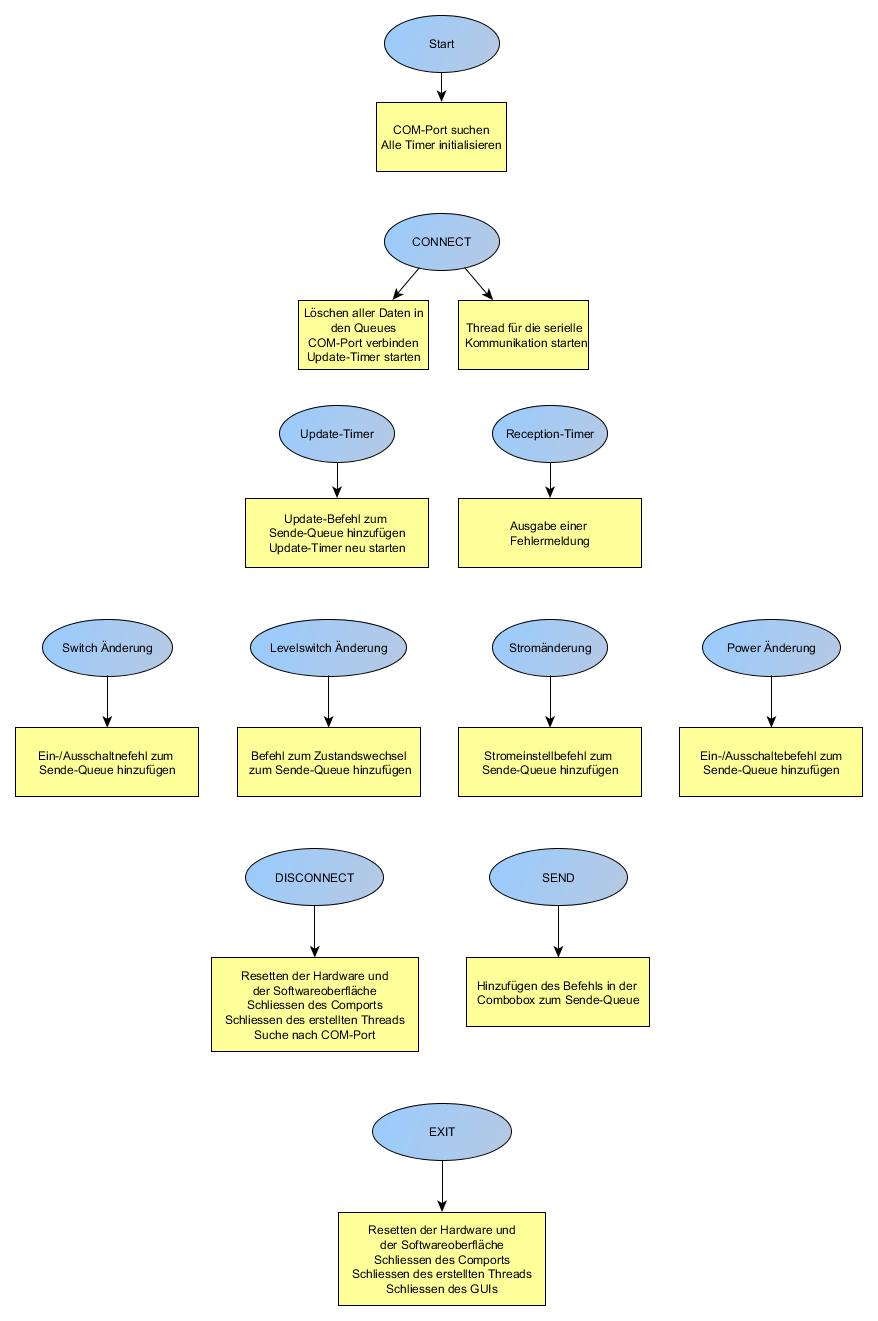


Abbildung 24: GUI-Events

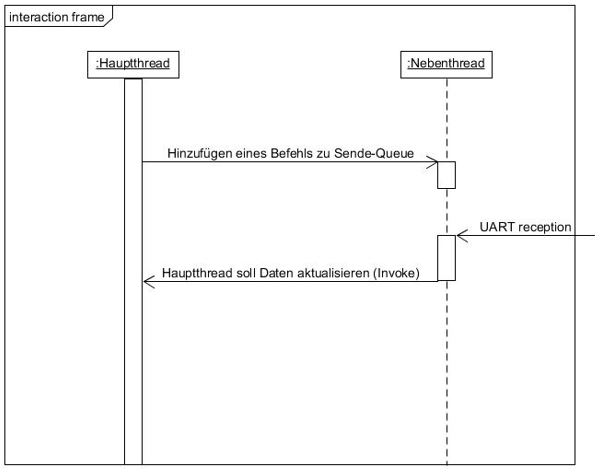
Abbildung 25 zeigt die Interaktion zwischen dem Haupt und dem Nebenthread, welcher für das Senden und Empfangen über die serielle Schnittstelle zuständig ist.  
Aus dem Hauptthread werden dauernd Sendebefehle in den Sende-Queue hinzugefügt. Sobald dies geschieht sendet der Nebenthread ein Befehl nach dem anderen an das HiL-Interface. Der nächste Befehl kann aber immer erst dann gesendet werden, wenn die Antwort des vorhergegangenen Befehls eingetroffen und verarbeitet ist. Sobald die Daten eingetroffen sind werden sie vom Nebenthread auch gerade verarbeitet. Handelt es sich um Daten, welche im User Interface aktualisiert werden müssen, so «teilt» der Nebenthread dem Hauptthread mit, dass er die Daten aktualisieren soll.

Abbildung 25: Thread-Interaction

# Bedienungsanleitung

Die folgende Kurzanleitung soll dem Benutzer die Funktion und die Bedienung des HiL-Interface erklären.

## Bedienung und Anzeige

Das HiL-Interface kann entweder mit Hilfe der Demonstrationssoftware oder mit Hilfe eines Terminals (zum Beispiel Hterm oder TeraTerm) verwendet werden. Beide Bedienungsmöglichkeiten werden kurz erklärt.

### Anzeige

Auf der Hardware des HiL-Interface gibt es nicht sehr viele Anzeigeelemente. Das Wichtigste ist das Heartbeat-LED. Dies befindet sich direkt neben der Speisung und signalisiert, dass das Gerät betriebsbereit ist.   
  
Die beiden anderen LEDs sind die Rx- und Tx-LEDs. Diese LEDs blinken auf, sobald der USB-Stecker verbunden wird und zeigen während dem Betrieb lediglich an, dass Daten zwischen dem PC und dem Gerät ausgetauscht werden.

### Anschluss

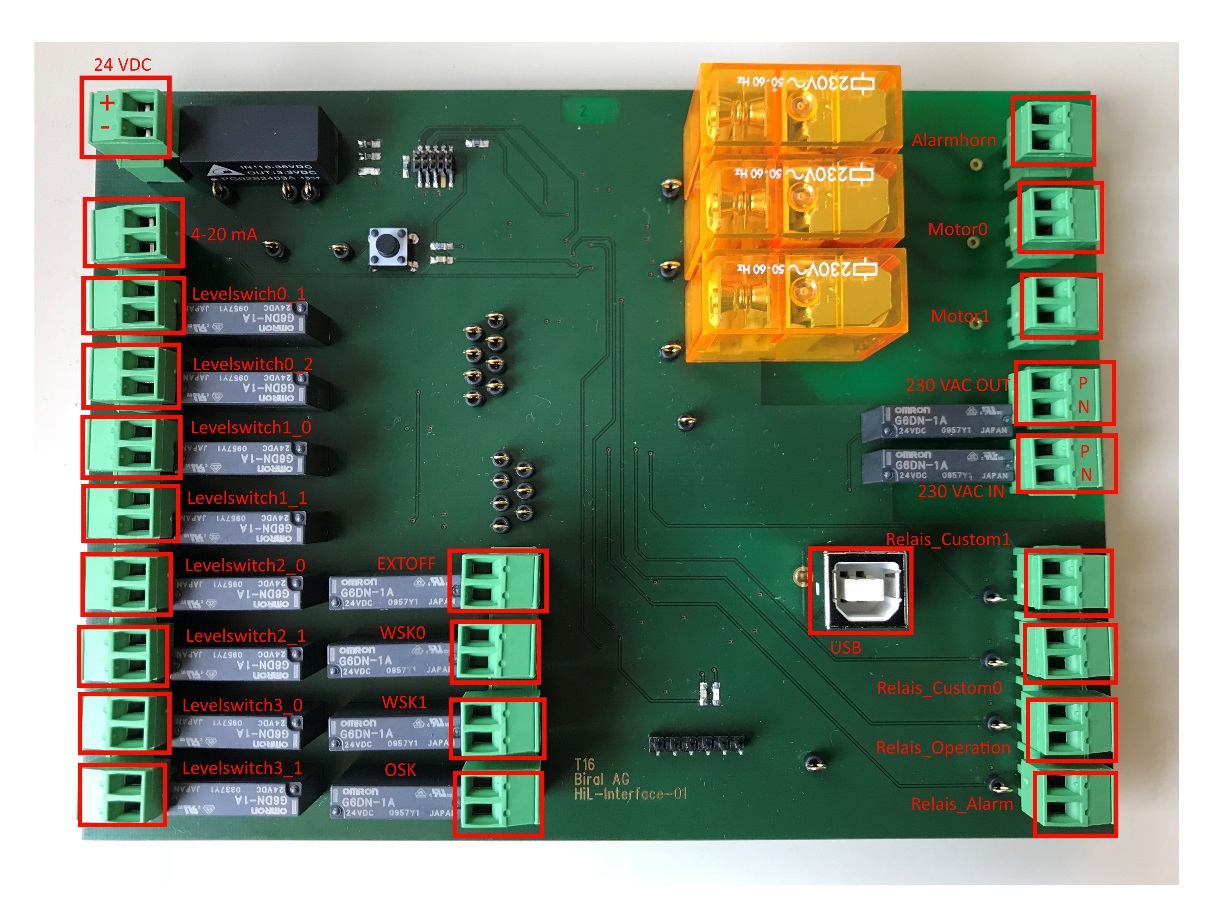
Bevor das Gerät gestartet wird, sollten noch alle Anschlüsse mit den Inputs/Outputs der DismessA-Steuerung verbunden werden, Dies kann auch nach dem Einschalten noch erfolgen, ohne dass das HiL-Interface Schäden davonträgt.

Abbildung 26: Anschluss HiL-Interface

### Start

Sobald das Gerät eingeschaltet wird (an eine 24 VDC Speisung gehängt wird), beginnt die Heartbeat-LED zu blinken. Das Gerät ist nun betriebsbereit und kann mit einem PC verbunden werden.

### Bedienung durch Hterm

#### Verbinden des HiL-Interface

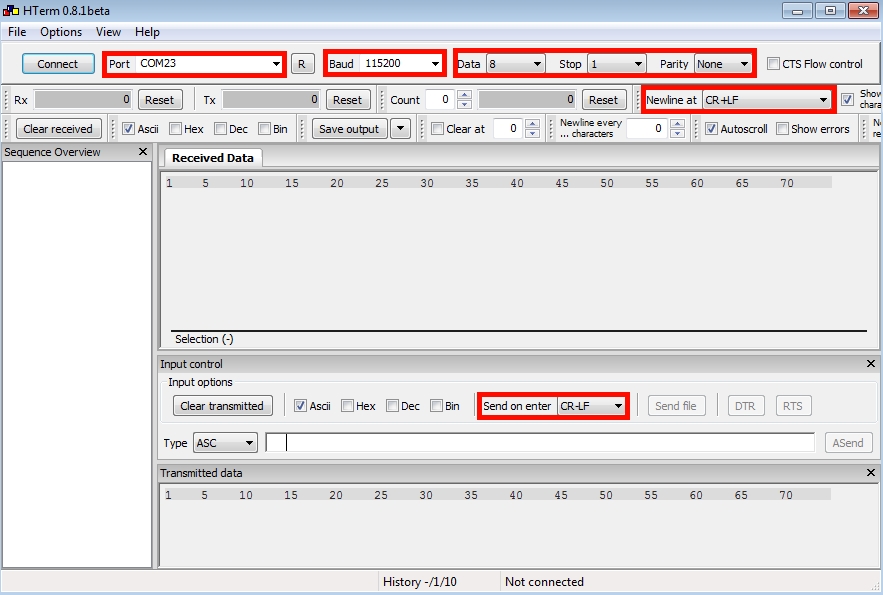
Um die Verbindung zwischen der Hardware und der Software herzustellen, müssen zuerst einige Einstellungen in Hterm gemacht werden. Diese sind auch notwendig, damit die Kommunikation überhaupt korrekt funktionieren kann. Im Beispiel auf Abbildung 27 ist zu sehen, wie die Einstellungen für die Verbindung aussehen müssen. Wichtig ist, dass beim Senden eines Befehls immer Carriage Return (CR) und Line Feed (LF) am Ende angehängt sein muss, damit die Hardware den Befehl erkennen kann. Wenn alle Einstellungen mit den Einstellungen in Abbildung 27 übereinstimmen, kann der Connect-Button gedrückt werden und das HiL-Interface ist mit dem PC verbunden.

Abbildung 27: Einstellungen Hterm

#### Kommunikation

Die Kommunikation mit dem HiL-Interface wird über ein Sendefeld (rot markiert in Abbildung 28) geregelt. Dort kann der Befehl gemäss der in Abschnitt 3.1.4.4.1 beschriebenen Syntax eingegeben werden. Der Befehl wird dann mit drücken der Enter-Taste an das HiL-Interface gesendet. Die Rückgabe vom HiL-Interface wird dann im Received Data Abschnitt dargestellt.

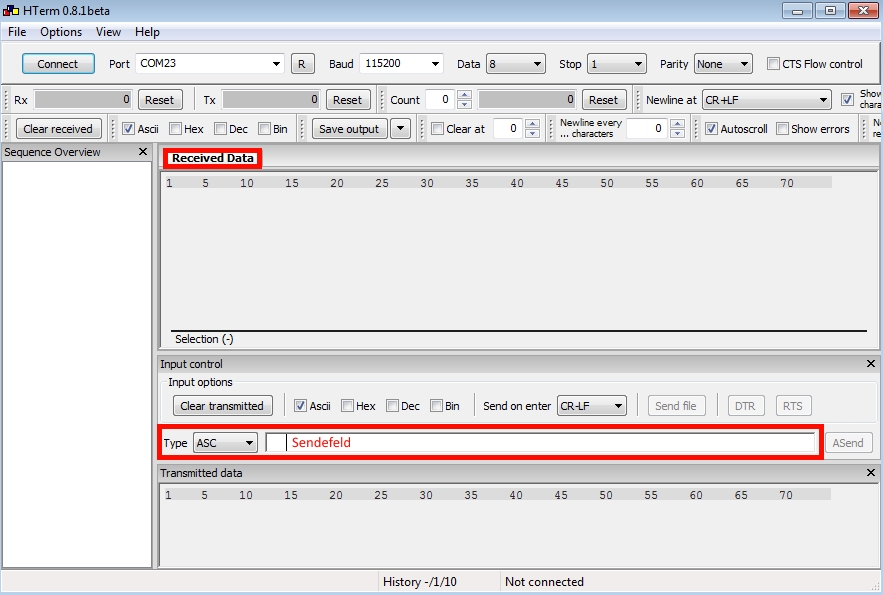


Abbildung 28: Hterm Kommunikation

### Bedienung mit Demonstrationssoftware

#### Verbinden des HiL-Interface

Beim Starten der Demonstrationssoftware sind die nicht mit rot gekennzeichneten Abschnitte gesperrt und nur der Verbindungsbereich (rot markiert in Abbildung 29), sowie der EXIT-Button können verwendet werden. Um den PC mit dem HiL-Interface zu verbinden, müssen die Einstellungen für den COM-Port und die Baudrate gemacht werden. Danach kann über den Button „CONNECT“ die Verbindung hergestellt werden. Die Einstellungen werden in Abbildung 29 angezeigt.

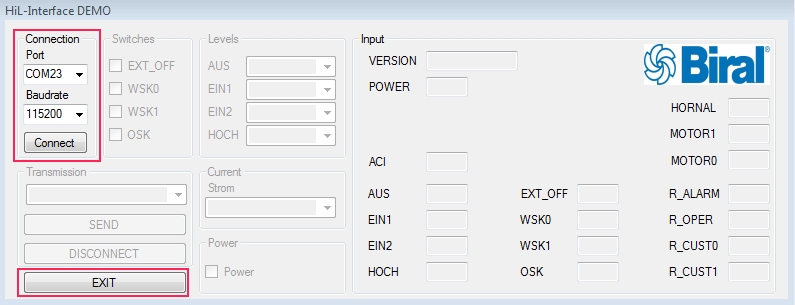


Abbildung 29: Verbinden der Demo-Software

#### Steuerung

Direkt nach dem Verbinden der Software mit der Hardware werden die Bedienelemente freigegeben und die Verbindungselemente gesperrt. In Abbildung 30 können die einzelnen Einstellungsblöcke betrachtet werden.

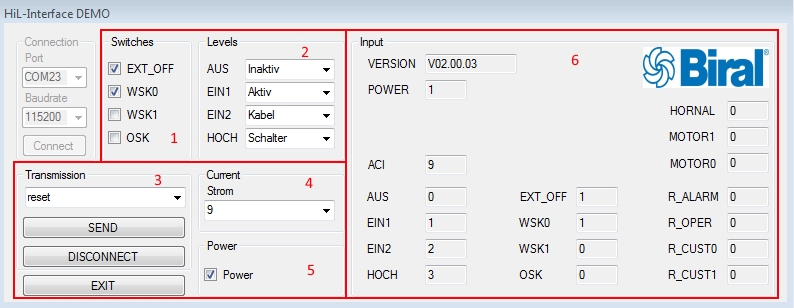


Abbildung 30: Bedienoptionen Demo-Software

Mit den Checkboxen in der Gruppe 1 können EXTOFF, WSK0 und 1 sowie OSK ein- und ausgeschaltet werden. Auf der rechten Seite befindet sich die Gruppe 2 mit den Levelswitches.

Bei jedem Levelswitch kann man individuell mit Hilfe des Drop-Down-Menus den Zustand des Levelswitches wählen. Hierbei kann zwischen den Zuständen Inaktiv, Aktiv, Kabel und Stecker gewählt werden. Mit der Gruppe 3 können die Befehle «reset» und «version» mit dem SEND-Button übermittelt werden. Drückt man den DISCONNECT-Button, so wird die Hardware resettet und man gelangt wieder zu Punkt 4.1.5.1. Mit dem Dropdown-Menu in Gruppe 4 kann der Strom für die 4-20 mA Schnittstelle eingestellt werden und mit Gruppe 5 kann die Speisung für die DismessA-Schaltung ein und ausgeschaltet werden.

#### Ausgabe

In der Abbildung 30 kann man sehen, dass noch eine 6. Gruppe eingezeichnet ist. Dies ist mit Abstand die Grösste Box und stellt die Zustände der Elemente auf der Hardware dar. Wenn man in den anderen 5 Gruppen einen Zustand ändert, so kann man mit einer kleinen Verzögerung die Änderung des Zustandes in Gruppe 6 beobachten. Die Verzögerung kommt daher, dass die Zustände in einem gewissen Zeitabstand gepollt werden und die Aktualisierungszeit somit vorgegeben ist.

#### Beenden der Software

In der unteren linken Ecke des Programms gibt es einen EXIT-Button (Gut ersichtlich in Abbildung 29). Dieser dient dazu die Hardware zu resetten und die Software zu beenden.

# Schlussbetrachtung

## Fazit

Das HiL-Interface erfüllt alle grundlegenden Ziele, welche im Pflichtenheft angegeben wurden. Es konnte auch die Hälfte der optionalen Ziele erreicht werden. Das Hil-Interface kann alle Inputs und Outputs der DismessA-Steuerung stimulieren und kann von einem PC aus gesteuert werden. Zur Demonstration der Funktionstüchtigkeit des Gerätes und seinen Funktionen wurde ein Demo-GUI erstellt, welches alle wichtigen Funktionen der Hardware abdeckt.

Alles in allem würde ich die Arbeit als gelungen betrachten. Ich konnte das Wissen, welches ich aus der Vorarbeit erarbeitet habe gut anwenden und habe auch noch einiges an neuem Wissen dazugewonnen. Vor allem im Bereich der GUI-Programmierung konnte ich noch einiges dazulernen. Trotz dem, dass ich nicht ein messereifes Demonstrations-GUI erstellen konnte, war die Arbeit mit viel Spass und abwechslungsreicher Arbeit verbunden.

## Verbesserungsmöglichkeiten

Trotz sauberem Vorgehen während der Arbeit konnten noch einige kleine Fehler entdeckt werden, welche bei späteren Versionen auskorrigiert werden sollten.

### Hardware

* Bei der 4-20 mA Schnittstelle herrscht ein kleiner Stromoffset von 550 uA. Dieser Offsetstrom könnte durch den Einsatz eines zweiten Operationsverstärkers auskorrigiert werden.
* Anstatt der aktuellen Rx-/TX-LEDs könnten andere LEDs (beispielsweise die der DismessA-Steuerung) eingesetzt werden, da die Helligkeit keine Rolle spielt, da sie ja nur die Übertragung anzeigen und nicht für Beleuchtungszwecke verwendet werden.

### Software

* Die Demonstrations-GUI verbraucht im Moment sehr viel Arbeitsspeicher vom PC. Dies liegt daran, dass das Senden und das Empfangen über die UART-Schnittstelle mit Hilfe von Pollen geregelt wird. Dies könnte verbessert werden, indem man anstatt ConcurrentQueues, BlockingCollections verwendet.
* Es wäre noch denkbar, dass vor allem bei einem messereifen Demonstrations-GUI noch ein Help-Button eingebaut werden sollte, damit die Funktionen des Programms angezeigt werden.

## Ausblicke

Es ist fraglich, ob noch einmal eine Zweitversion von dem HiL-Interface entwickelt wird, da es lediglich für die Automatisierung der Teststrecke der DismessA-Steuerung dient. Es wäre jedoch gut denkbar, dass die Hardware als Vorlage für zukünftige Testgeräte dienen könnte. Die Hardware wird aber möglicherweise für die Demonstration der DismessA-Steuerung an Messen verwendet. Daher wäre es schlau, eine neue Demonstrationssoftware am PC zu erstellen, welche visuell etwas hergibt.

# Glossar

|  |  |
| --- | --- |
| Wort | Beschreibung |
| Baudrate | Baud ist eine Einheit aus der Nachrichtentechnik. Die Baudrate beschreibt die Anzahl Zeichen, die pro Sekunde übertragen werden können. |
| COM-Port | Communication Port -> UART |
| Evaluation | Beurteilung, Bewertung, ob sich ein Bauteil für den vorgegebenen Zeck eignet oder nicht. |
| Freilaufdiode | Freilaufdioden sind Bauteile, welche zum Schutz vor Überspannung oder unerlaubten Spannungen eingesetzt werden. |
| Galvanische Trennung | Elektrische Trennung zweier Stromkreise. |
| interrupt | Kurze Unterbrechung des Programmes, um eine andere zeitkritische Verarbeitung durchzuführen. |
| oszillator | Ein Oszillator ist ein *schwingungsfähiges System*. Oszillation bedeutet, dass eine fortwährende Veränderung zwischen zwei Zuständen, oder um einen zentralen Punkt stattfindet, der meist der Ruhelage des Systems entspricht.[[24]](#footnote-24) |
| Peripherie | In einem elektronischen Gerät, werden alle zusätzlichen Schaltungen neben dem Controller als Peripherie bezeichnet. |
| Snubber-glied | Als Snubber-Glied bezeichnet man eine elektrische Schaltung, die störende Hochfrequenzen oder Spannungsspitzen neutralisieren soll, die meist beim Schalten induktiver Lasten auftreten, wenn der Stromfluss abrupt unterbrochen wird.[[25]](#footnote-25) |
| Concurrent Queues / BlockingCollections | Die ConcurrentQueues werden in der Programmierung mit der Sprache C verwendet. Sie funktionieren wie ein FIFO (Datenregister: Was als erstes abgespeichert wird, wird auch als erstes ausgelesen) und sind Thread-Sicher. Das heisst es kann nicht aus mehreren Threads gleichzeitig auf den Inhalt zugegriffen werden. BlockingCollections funktionieren gleich, besitzen aber noch andere Funktionen. |

# Literatur- und Quellenverzeichnis

Auf den folgenden Seiten werden alle verwendeten Quellen angegeben.

## Internet

**Snubber-Glied**Autor: Unbekannt  
Link: [https://de.wikipedia.org/wiki/Snubber](https://de.wikipedia.org/wiki/Snubber%20) (Stand:03.04.2018)

**USB**Autor: Unbekannt  
Link: <https://de.wikipedia.org/wiki/Universal_Serial_Bus> (Stand:03.04.2018)

**Windows Watchdog Timer Calculation**Autor: Robert Thomas  
Link: <https://www.youtube.com/watch?v=Tx5ewDyjmM0> (Stand:03.04.2018)

**Partial Classes in C#**Autor: Bill WagnerLink: <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/partial-classes-and-methods> (Stand:03.04.2018)

**System Timer**Autor: Unbekannt  
Link: <https://msdn.microsoft.com/de-de/library/system.timers.timer(v=vs.110).aspx> (Stand:03.04.2018)

**Threading**Autor: Unbekannt  
Link: <https://www.tutorialspoint.com/csharp/csharp_multithreading.htm> (Stand:03.04.2018)

**ConcurrentQueues**Autor: Unbekannt  
Link: <http://dotnetpattern.com/csharp-concurrentqueue> (Stand:03.04.2018)

**BlockingCollections**Autor: Unbekannt  
Link: <https://msdn.microsoft.com/en-us/library/dd267312.aspx> (Stand:03.04.2018)

**Invoke**Autor: GETah  
Link: <https://stackoverflow.com/questions/11211533/update-a-control-in-ui-with-running-background-thread-in-winforms> (Stand:03.04.2018)

# Anhang

Aus ökologischen Gründen beschränkt sich der Anhang auf einen USB-Stick

**USB-Stick**

Ordner: IPA

Beschreibung: Alle Datenblätter, Dateien und Sourcecodes

## Datenträgerstruktur

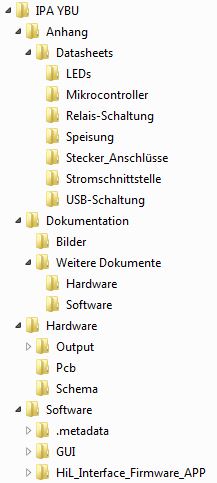


Abbildung 31: Ordnerstruktur

1. [..\Anhang\Datasheets\Mikrocontroller\STM32F303CCT6.pdf](../Anhang/Datasheets/Mikrocontroller/STM32F303CCT6.pdf) [↑](#footnote-ref-1)
2. <https://de.wikipedia.org/wiki/Universal_Serial_Bus> [↑](#footnote-ref-2)
3. [..\Anhang\Datasheets\USB-Schaltung\FT232RL.pdf](../Anhang/Datasheets/USB-Schaltung/FT232RL.pdf) [↑](#footnote-ref-3)
4. [..\Anhang\Datasheets\LEDs\SML-LXFP0603SICATR.pdf](../Anhang/Datasheets/LEDs/SML-LXFP0603SICATR.pdf) [↑](#footnote-ref-4)
5. [..\Anhang\Datasheets\LEDs\SML-LX0603SUGW-TR.pdf](../Anhang/Datasheets/LEDs/SML-LX0603SUGW-TR.pdf) [↑](#footnote-ref-5)
6. [..\Anhang\Datasheets\Relais-Schaltung\ULN2803A.pdf](../Anhang/Datasheets/Relais-Schaltung/ULN2803A.pdf) [↑](#footnote-ref-6)
7. [..\Anhang\Datasheets\Relais-Schaltung\G6DN-1A DC24.pdf](../Anhang/Datasheets/Relais-Schaltung/G6DN-1A%20DC24.pdf) [↑](#footnote-ref-7)
8. [..\Anhang\Datasheets\Relais-Schaltung\RELAY 230VAC.pdf](../Anhang/Datasheets/Relais-Schaltung/RELAY%20230VAC.pdf) [↑](#footnote-ref-8)
9. [..\Anhang\Datasheets\Stromschnittstelle\LM385-N.pdf](../Anhang/Datasheets/Stromschnittstelle/LM385-N.pdf) [↑](#footnote-ref-9)
10. [..\Anhang\Datasheets\Stecker\_Anschlüsse\CM020PT.pdf](../Anhang/Datasheets/Stecker_Anschlüsse/CM020PT.pdf) [↑](#footnote-ref-10)
11. [..\Anhang\Datasheets\Speisung\UNOPOWER.pdf](../Anhang/Datasheets/Speisung/UNOPOWER.pdf) [↑](#footnote-ref-11)
12. [..\Anhang\Datasheets\Speisung\PG02S2403A.pdf](../Anhang/Datasheets/Speisung/PG02S2403A.pdf) [↑](#footnote-ref-12)
13. [..\Anhang\Datasheets\USB-Schaltung\FT232RL.pdf](../Anhang/Datasheets/USB-Schaltung/FT232RL.pdf) [↑](#footnote-ref-13)
14. [..\Anhang\Datasheets\USB-Schaltung\SN65220.pdf](../Anhang/Datasheets/USB-Schaltung/SN65220.pdf) [↑](#footnote-ref-14)
15. [..\Anhang\Datasheets\LEDs\SML-LXFP0603SICATR.pdf](../Anhang/Datasheets/LEDs/SML-LXFP0603SICATR.pdf) [↑](#footnote-ref-15)
16. [..\Anhang\Datasheets\LEDs\SML-LX0603SUGW-TR.pdf](../Anhang/Datasheets/LEDs/SML-LX0603SUGW-TR.pdf) [↑](#footnote-ref-16)
17. [..\Anhang\Datasheets\USB-Schaltung\SMF05C.pdf](../Anhang/Datasheets/USB-Schaltung/SMF05C.pdf) [↑](#footnote-ref-17)
18. [..\Anhang\Datasheets\Relais-Schaltung\RELAY 230VAC.pdf](../Anhang/Datasheets/Relais-Schaltung/RELAY%20230VAC.pdf) [↑](#footnote-ref-18)
19. [..\Anhang\Datasheets\Relais-Schaltung\ULN2803A.pdf](../Anhang/Datasheets/Relais-Schaltung/ULN2803A.pdf) [↑](#footnote-ref-19)
20. [..\Anhang\Datasheets\Stromschnittstelle\LM385-N.pdf](../Anhang/Datasheets/Stromschnittstelle/LM385-N.pdf) [↑](#footnote-ref-20)
21. [..\Anhang\Datasheets\Speisung\PG02S2403A.pdf](../Anhang/Datasheets/Speisung/PG02S2403A.pdf) [↑](#footnote-ref-21)
22. [Weitere Dokumente\Hardware\Linearisierung 4-20 mA Schnittstelle.xlsx](Weitere%20Dokumente/Hardware/Linearisierung%204-20%20mA%20Schnittstelle.xlsx) [↑](#footnote-ref-22)
23. [Weitere Dokumente\IPA\_YBU\_Pflichtenheft.pdf](Weitere%20Dokumente/IPA_YBU_Pflichtenheft.pdf) [↑](#footnote-ref-23)
24. <https://de.wikipedia.org/wiki/Oszillator> [↑](#footnote-ref-24)
25. <https://de.wikipedia.org/wiki/Snubber> [↑](#footnote-ref-25)