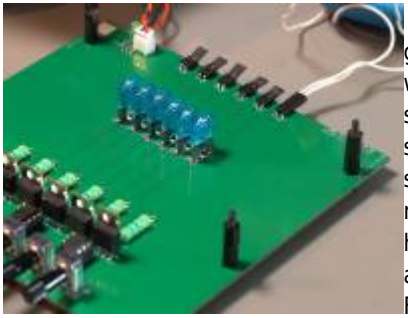
# Studentische Projekte

## Ströme messen mit Hallsensoren



Ströme richtig messen ist immer eine größere Herausforderung. Besonders dann, wenn man keinen direkten Zugang zum stromführenden Leiter hat. Hr. Sichler hat in seiner Bachelorarbeit untersucht ob sich mit sehr empfindlichen Hallsensoren Ströme messen lassen wenn diese über handelsüblichen Fahrzeugsicherungen angebracht werden. Die Idee dahinter: Fahrzeugkomponenten erkennen, welche bei einer Ruhestrommessung das Fahrzeug aufwecken. Das Problem: Das durch den Strom erzeugte Magnetfeld ist in der Größenordnung des Erdmagentfelds.
Videovorstellung

Poster zur Arbeit als Download

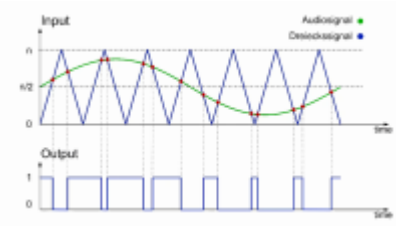
## Anpassung der Eingangsspannung über Tiefsetzsteller

Die Grundschaltung jeder Leistungselektronik ist der Tiefsetzsteller. Je wird nahezu auch auf jeder Schaltung benötigt um eine gemeinsame Eingangsspannung an die Bedürfnisse der einzelnen Komponenten wie zum Beispiel ein Mikrocontroller anzupassen. Besonders bei batteriebetriebenen Geräten kann nicht auf die beliebten Längsspannungsregler (LDO) aufgrund deren hohen Verlustleistung zurückgegriffen werden. In der Vorlesung Leistungselektronische Systeme haben die Teilnehmer recherchiert welche ICs am Markt verfügbar sind mit integriertem Schaltelement. Zum deren Betrieb werde typischerweise nur wenig externe Bauelemente wie die

Induktivität, die Freilaufdiode und ein Glättungskondensator benötigt. Download:

leistungselektronik_tiefsetzsteller_praesentation.pdf

## Unsere Motorsteuerung als digitale Audioendstufe




Jan Sperling hat in seiner Bachelorarbeit gezeigt, dass wir die Hardware der Motoransteuerung auch als digitale Class D Audio Endstufe einsetzen können. Unser TI-Launchpad tastet dazu die Audiosignal mit dem Analog-Digitalwandler zyklisch ab und übergibt die Werte dem PWM-Modul. Mit Hilfe einer T/2 versetzten Taktung wird dann der Lautsprecher angesteuert. Leider können die Audio-Signale nicht direkt auf den Mikrocontroller losgelassen werden. Audiosignale sind mittelwertfrei, das bedeutet sie schwingen um den Nullpunkt, mit einer maximalen Eingangsamplitude von ca. 1,2V. Die Analog- nach Digitalwandlung jedoch erwartet Pegel zwischen 0 und 5V. Dazu wurde von Hr. Sperling eine Pegelanpassung entwickelt welche in der Lage ist das Audiosignal exakt auf den Wertebereich des AD-Wandlers anzupassen. Für den perfekten Sound muss allerdings noch eine Nachfolgearbeit eine Störeinkopplung vom Leistungsteil in die Pegelpassung aufspüren und natürlich beseitigen. Posterdokumentation:

plakat_digitale_audioendstufe.pdf

Videodokumentation

## Displayansteuerung mit i2C Bus und Simulink Embedded Coder

Konstantin Lutz hat in seiner Bachelorarbeit eine Displayansteuerung implementiert für unsere universelle Hardwareplattform in der Leistungselektronik. Das Display kann nun für weitere Projekte komfortabel in die Simulink-Umgebung eingebunden werden. Ohne sich um die Details kümmern zu

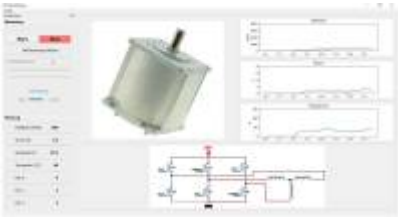
müssen können die einzelnen Zeilen ganz einach über Strings beschrieben werden. Als Schnittstelle wird der i2C Bus verwendet. Das Display kann sowohl im Stand-Alone als auch im Monitor&Tune Modus beschrieben werden. Im Video Video erklärt Hr. Lutz die Details zum Projekt und wie das Display angesprochen werden kann.
Die vollständige Dokumentation zum Download:

ansteuerung_eines_lcd-displays_mit_matlab_konstantin_lutz_2021.pdf

und die Simulink Demo-Files:

htwg_i2c_display_simulink.zip

## Ansteuerung eines BLDC Motors mit Visualisierung



Auch im „Corona-Semester“ haben wir in der Vorlesung Leistungselektronik unsere Projekte realisiert. Allerdings lag der Focus jetzt auf der Projektierung, Simulation und virtuellen Inbetriebnahme. Die Projektgruppe mit Christoph Casagranda, Johannes Huober und Lucas Flender hat in ihrem Projekt gezeigt wie sich ein BLDC - Motor ansteuern lässt mit einer Visualisierung der aktuellen Rotorposition auf einem externen PC. Nachfolgender Link zeigt das Projekt in einem kleinen Videobeitrag. Video

## Ein mit Simulink programmier- und regelbarer Buck-Boost Converter

Lukas Hölsch hat in seiner Projektarbeit zum ersten mal im Labor für Leistungselektronik ein TI Launchpad eingesetzt zur Programmierung und Regelung eines Buck-Boost Converters.
Das Besondere daran ist, dass die

Programmierung in Simulink rein graphisch erfolgt. Das bedeutet es sind keinerlei Programmierkenntnisse erforderlich. Weiterhin erlaubt die Anbindung der Schaltung an Simulink eine Variation aller Parameter im laufenden Betrieb. Somit ist es auf einfache Weise möglich Regelerparameter anzupassen und sich die Ergebnisse direkt visualisieren zu lassen.
Dokumentation zum Download.

projektarbeit_lukas_hoelsch.pdf

---

## 100W Klasse D Audio Verstärker

Im Porjektlabor Leistungselektronik haben wir auch dieses Semester eine 100W Audioendstufe entwickelt. Das Ergebnis von Fabian Sernatinger und Dennis Rutkowski kann sich hören lassen. Die erstmals eingesetzten Snubber über den Leistungshalbleitern haben einen deutlich positiven Effekt auf das Klangerlebnis. Die Endstufe generiert die Taktsignale mit Hilfe zweier analogen Referenzspannungen im T/2 Versatz. Dadurch können aufwendige Filerschaltungen zum Dämpfung des Trägersignals entfallen.
Überblick zum Projekt

le_rutkowskisernatinger_praesentation1.pdf

,

schaltplan_class_d_amp_rev1.pdf

Inbetriebnahmevideo

---

## Aktuelle Traktorgeneration

Auch dieses Semester gab es eine neu Version des vollelektrischen Traktors. Der Fokus liegt zunehmen auf Fahrspaß womit die Studierenden dem Motor immer mehr Drehmoment abverlangen. So konnte dieses Semester der Abzug noch einmal um 50% gesteigert werden. Die Obergrenze ist aber noch nicht erreicht. Ich bin gespannt ob wir in den kommenden Jahren noch schneller beschleunigen.

## Wechselrichter

Peter Schmidt hat in seiner Bachelorarbeit einen einen Sinuswechselrichter aufgebaut der zukünftig im Praktikum zu den elektrischen Maschinen zum Einsatz kommen wird. Der Wechselrichter kann sowohl im Stand-Alone als auch im Grid-Betrieb zur Energieeinspeisung in das öffenltiche Netz verwendet werden. Die Nachfolgearbeit wird die Frequenzerkennung verbessern und den Wechselrichter in einem kompakten Design aufbauen.
Dokumentation:

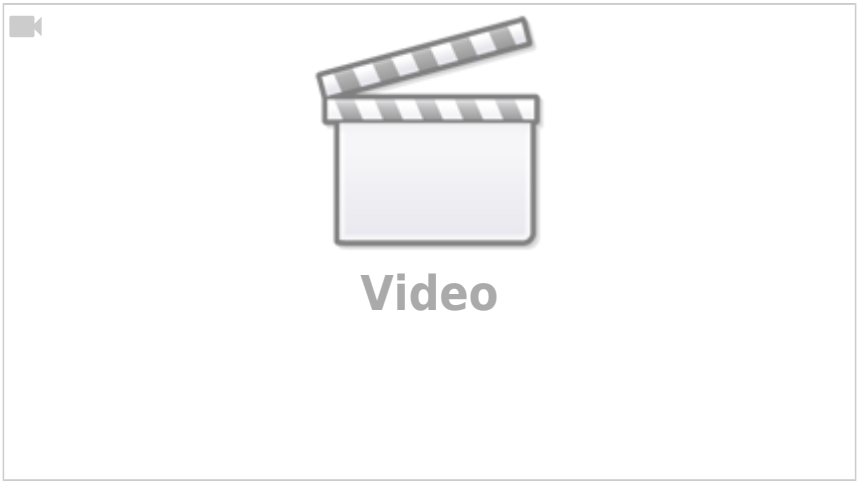
netzgefuehrter_sinuswechselrichter_schmidt.pdf

# Elektr. Trettraktor

Wer ist als Kind nicht gerne mit seinem Trettraktor oder dem Dreirad unterwegs gewesen? Wieviel besser wäre es dann erst mit einer elektrisch angetriebenen Version oder einer elektrischen Unterstützung? Im Sommersemester 2017 haben wir zum ersten mal für die lange Nacht der Wissenschaft die Traktoren elektrifiziert. Heute gibt es bereits mehrer Versionen: Als Pedelec, als Elektrofahrzeug, mit Gleichstrommotor oder BLDC Motor.
Die Traktoren werden ständig weiterentwickelt und stehen für Projekt- und Abschlussarbeiten zur Verfügung:

- Elektrische Lenkung
- Kollisionsvermeidung über Sensoren
- Autonom fahrende Traktoren
- Induktive Ladeplatte

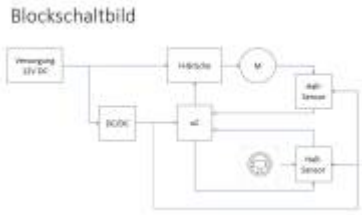
Die Systeme können auch im Rahmen des Projektlabors der Vorlesung Leistungselektronik aufgebaut werden.



# Steer-ByWire

Sollen unsere Traktoren bald autonom fahren können müssen noch einige Hardwareanforderungen dazu umgesetzt werden. Grundvoraussetzung ist natürlich die Möglichkeit die Fahrtrichtung elektronisch vorzugeben. Das bedeutet der Lenkwinkel / Fahrerwunsch muss elektronisch umgesetzt werden. Im PAE7 Praktikum haben sich Studenten der Aufgabe angenommen und einen ersten Umsetzungsvorschlag ausgearbeitet. Es geht hier nicht darum die perfekte Lösung für eine Fahrzeugintegration zu finden, sondern mit einfachen und vorhandenen Mitteln die Funktion darzustellen. Unsere Standartkomponente im Praktikum, welche auch hier zum Einsatz kommen:

- DC- Getriebemotor (Aus Akkuschrauber)
- H-Brücke aus vorhandenem Praktikumsversuch
- XC866 Trainingsboard




Die Lageerfassung des Lenkrads erfolgt über einen Hallsensor. Über einen zweiten Hallsensor wird die aktuelle Position der Lenkachse ermittelt. Die Software zur elektischen Lenkung hat nur eine Aufgabe: Der Fehlerwinkel zwischen Sollwinkel und Istwinkel muss ausgeglichen bzw. möglichst klein gehalten werden.
Das Projekt hat super funktioniert und es konnte gezeigt werden, dass auch mit einfachen Mitteln eine elektrische Lenkung oder ein Steer by Wire System aufgebaut werden kann. Für den Einsatz im Traktor müssen wir allerdings nach einem geigneten Motor ausschau halten, da:

- Das hohe Haltemomonte des Getriebes die Regelung ggf. sehr „nervös“ macht
- Das Getriebespiel der Akkuschraubereinheit sehr hoch ist

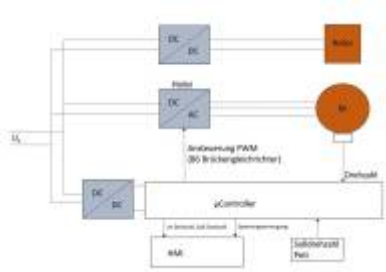
Das Ergebnis kann sich sehen lassen:

Video

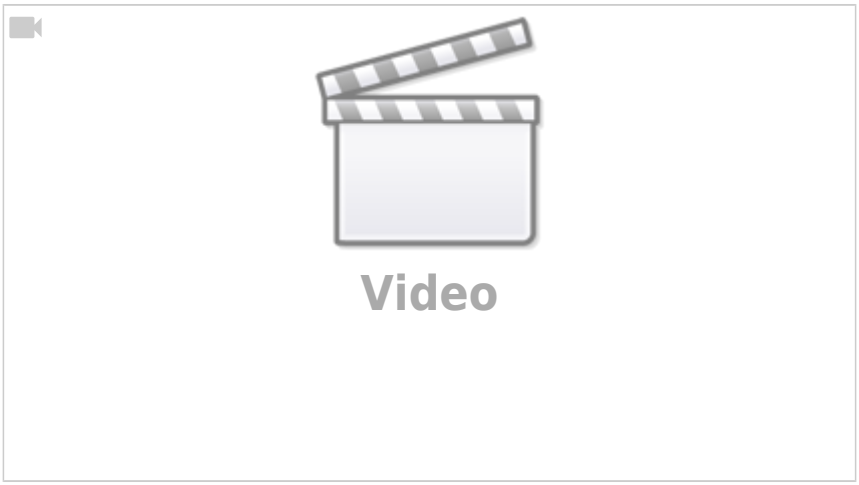
## Fahrzeuggenerator als Motor

Der Drehstromgenerator in modernen Fahrzeugen ist schon lange keine permanenterregte Gleichstrommaschine mehr, wie noch weithin angenommen. Besonders Fahrzeuge im Premmiumsegment haben einen Leistungsverbrauch im Bereich von 3-4kW auf der 12V- Seite mit steigender Tendenz. Für die Versorgung des 12V Bordnetz kommen heute Drehstromgeneratoren zum Einsatz. Genauer gesagt handelt es sich hierbei um fremderregte Synchronmaschinen.

Die Regelung der Ausgangsspannung bzw. des Ausgangsstromes erfolgt über einen Linearregler welcher den Erregerstrom entsrepchend beeinflusst.



Stabilisiert werden die Bordnetze über einen oder zwei Fahrzeugbatterien. Im ersten Schritt muss das hintere Lagerschild und die Gleichrichterdioden entfernt werden um an die einzelnen Wickelabgriffe zu kommen. Danach gilt es herauszufinden wie die Wicklung verschalten ist. Typischerweise sind die Maschinen im Dreieck verschalten. Da meist keine Informationen zur Verschaltung verfügbar sind kann über eine Spannungsmessung (Drehen von Hand +Erregerfeld anlegen) der Wicklungssinn ermittelt werden.




---

# TI Launchpad F28069

https://www.ti.com/tool/LAUNCHXL-F28069M#technicaldocuments

Das Launchpad F28069 bietet für die Leistungselektronik und elektrische Maschinen einen schnellen Einstieg

zur Steuerung und Regelung leistungselektronischer Schaltungen.
Zusätzlich ist es möglich die Hardware über Simulink zu programmieren, womit nur wenig Kontakt zum C-Code notwendig wird.
Die grafische Programmierung über Simulink ist in der Fahrzeugentwicklung und Autosar-Systemen weit verbreitet und bietet
auch ohne tiefes Wissen über die Hardware eine effiziente Programmierung.
Wie immer ist die erste Hürde (bis die Hardware mit Simulink kommuniziert) eine der Größten.

## Vorbereitende Maßnahmen:

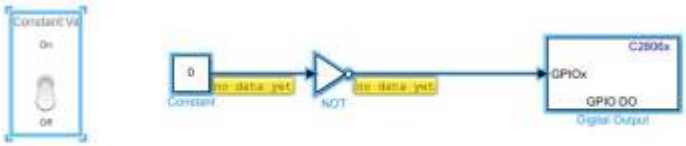
- Launchpad F28069 organisieren
- In Matlab die Pakete installieren: Embedded Coder Support Package for Texas Instruments C2000 Processors
- Notwendige Third Party Software wie CCS (Code Composer Studio) installieren

Anbei meine Fallstricke bei der Inbetriebnahme:

**Mit welchen LEDs kann ich spielen?**
LED Pins:

- GPIO34 D9 (Rot) (Enable Boot)
- GPIO39 D10 (Blau)

Zum Spielen bietet sich D10 an (Achtung, Ausgabe ist invertiert) ….



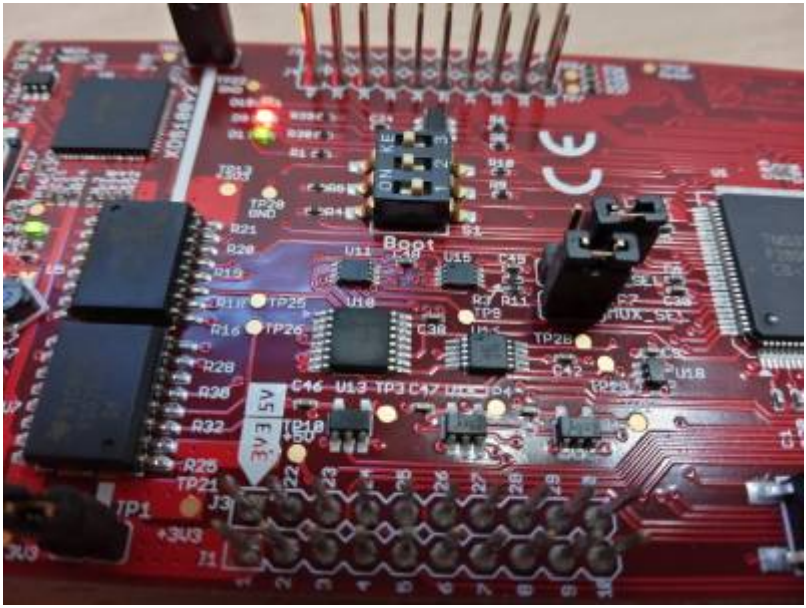

firstc2000_led_toggle.zip

Im Simulinkbeispiel lässt sich über den Schalter die LED D10 aktivieren und deaktivieren.
Dazu muss in Simulink unter Hardware- Run on Hardware - Monitor&Tune gewählt

werden.

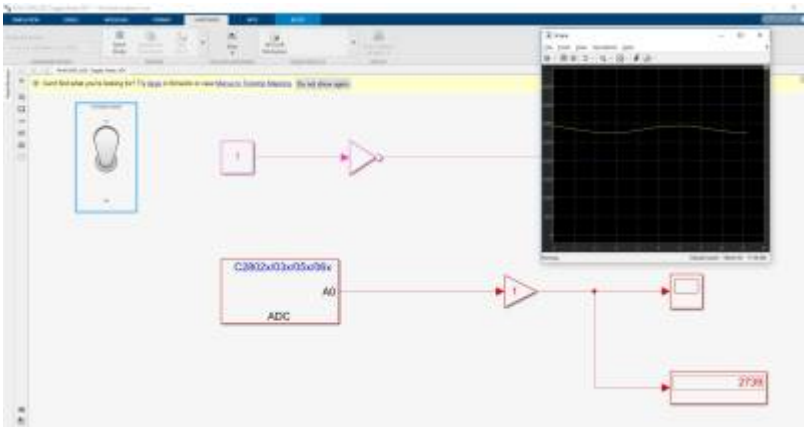
Falls der Download scheitert:

- Unter COM-Port im Gerätemanager muss XDS100 Class USB verfügbar sein, siehe https://e2e.ti.com/support/microcontrollers/c2000/f/171/t/670519
- Achtung: Schalter der Boot-Mode Switches beachten. Download aus Simulink mit Modus 2 Emulation Boot, siehe Launchpad Manual
- In Simulink muss der richtige COM-Port gewählt werden, siehe https://de.mathworks.com/help/supportpkg/texasinstrumentsc2000/ug/external-mode.html




## Daten vom ADC lesen

Über den Block ADC lässt sich der Analog- Digitalwandler konfigurieren und einlesen.
**To Do:** Wo ist der Unterschied zum Block AIOX?

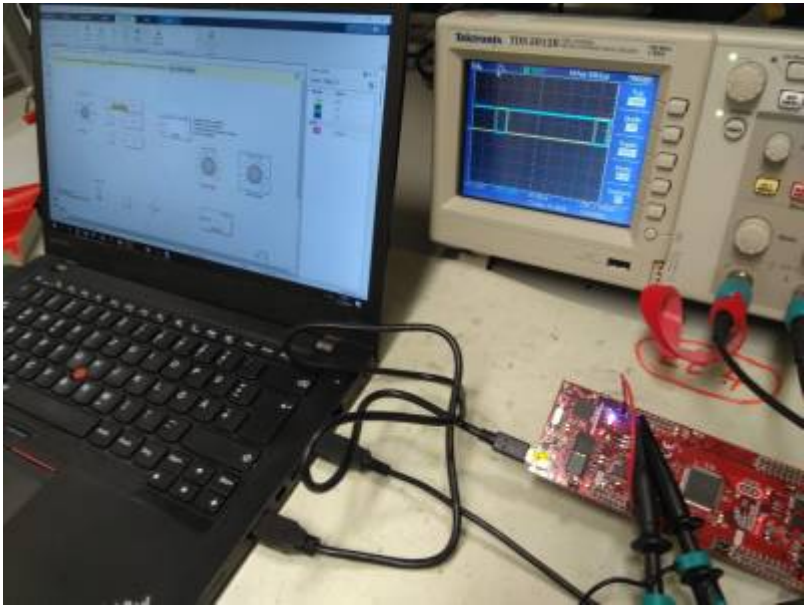



firstc2000_led_toggle_read_adc.zip

## PWM Ausgabe

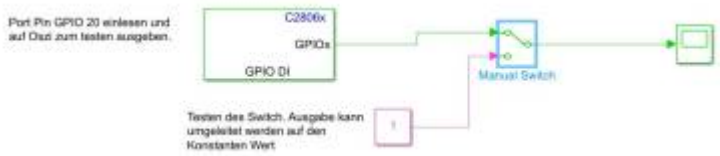
Die nächste Herausforderung besteht darin eine PWM zu generieren. Idealerweise gleich zusammen mit der invertierten Ausgabe
zur Ansteuerung einer Halbbrücke.

Folgendes Setup ist eine Erweiterung der bisherigen Versuch zur PWM Ausgabe.
Über die Dashboard-Potis lässt sich die Einschaltdauer ändern sowie die Todzeiten für die steigende und fallende Flanke.



firstc2000_led_toggle_read_adc_calcvalue_epwm.zip

## Digitalport einlesen




Der erste Versuch direkt einen Interrupt Routine auszulösen über den GPIO Pin 20, siehe unten ging leider schief. Die neue Strategie ist daher erst zu testen einen Digitalen Eingang auszulesen und damit auch zu prüfen ob die Logik-Pegel erkannt werden. Erzeugt werden die Impulse vom GPIO 0 welcher ein in Simulink einstellbares Rechtecksignal ausgibt. Erfreulich ist, dass sich die Signalquelle im Monitor&Tune Modus direkt ändern lassen, siehe

firstc2000_led_toggle_read_adc_calcvalue_epwm_readdigital.zip

## Interrupt

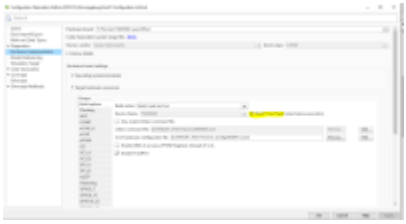
Das Launchpad besitzt eine Reihe an interruptfähigen Pins. In der Pin-Übersicht des Quick-Start-Guide mit (!) gekennzeichnet. Das fogende Beispiel löst durch einen Flankenwechsel am GPIO20 einen Interrupt aus. In der ISR wird zuerst Port 20 eingelesen, dann die LED ensprechend dem Port-Pin gesetzt. Achtung, das LED Signal ist invertiert. Zum testen lassen wir uns am GPIO0 ein Rechtecksignal erzeugen. Damit der Interrupt ausgelöst wird muss eine Drahtbrücke von GPIO 0 auf 20 gesetzt werden.
Wichtig ist, dass in Matlab unter Hardware-Settings → Target hardware resources → External Interrupt → der XINT3 ausgewählt wird für GPIO 20. Leider lassen sich nur drei externe Interruptquellen anwählen.

firstc2000_led_toggle_read_adc_calcvalue_epwm_interrupt.zip

## Stand-Alone

Um eine Stand-Alone Applikation (ohne Simulinkanbindung) zu generieren müssen folgende Änderungen vorgenommen werden.

- In Simulink unter Hardware Implementation „Boot From Flash“ aktivieren. Danach das Programm mit „Build, Deploy and Start“ auf den Controller flashen. Das Programm starte nach erfolgreichem Flashen auf dem Controller.



- Auf dem Lanchpad die Dipschalter (Mäuseklavier) auf die Stellung „Get Mode“ setzen: ON ON OFF. Damit wird die richtige Bootkonfiuration gesetzt, damit nach einem Spannungsreset das Programm geladen wird.

Dies ist der letzt Schritt bevor die Schaltung sich nicht mehr aus der USB verbindung versorgt. (Ausgangszustand: OFF ON ON)

- Jetzt können wir das Launchpad über die Zwischenkreisspannung (Batterie oder ähnliches versorgen). Dazu auf der Leiterkarte JP4 setzten. Auf dem **Launchpad** die Jumper JP1 und JP2 entfernen.

## Monitoring über Serielle Schnittstelle

In vielen Fällen stößt die Monitor&Tune Funktion an Ihre Grenzen. Insbesondere wenn es darum geht Interrupts mit hoher Wiederholrate zu übertragen. Hier ist es sinnvoller den µC ohne Tune-Funktion laufen zu lassen und sich die Werte über die serielle Schnittstelle an den PC (beliebiges Interface) übertragen zu lassen. Besonders einfach geht das natürlich wieder mit Simulink.
Vorgehensweise:

1. Code für µC in Simulink erstellen und über den Button Build,Deploy&Start auf den µC übertragen. Im Code müssen natürlich die zu übertragenden Signale an die Serielle Schnittstelle übergeben werden. (Siehe Beispielcode). Dieser Schritt muss bei jedem Power-Up des µC wiederholt werden. Außer, es werden die Jumper-Settings für den Stand-Alone Betrieb gesetzt, siehe oben.
2. Das Serial Communication Interface A ist für den Monitor&Tune Modus reserviert. Zur Übertragung zu einem Host-PC wird das SCI B Modul verwendet. Da der FTDI Serial-To-USB nur einen Kanal beinhaltet müssen wir die Jumper-Settings auf dem Board verändern entsprechend Tabelle 1 im Manual. JP 7 = offen, JP 6 = gesetzt/offen.
3. Simulink als Host und Block zum lesen/schreiben der Seriellen Schnittstelle öffnen. (Achtung, ggf. muss die COMport Nummer angepasst werden.

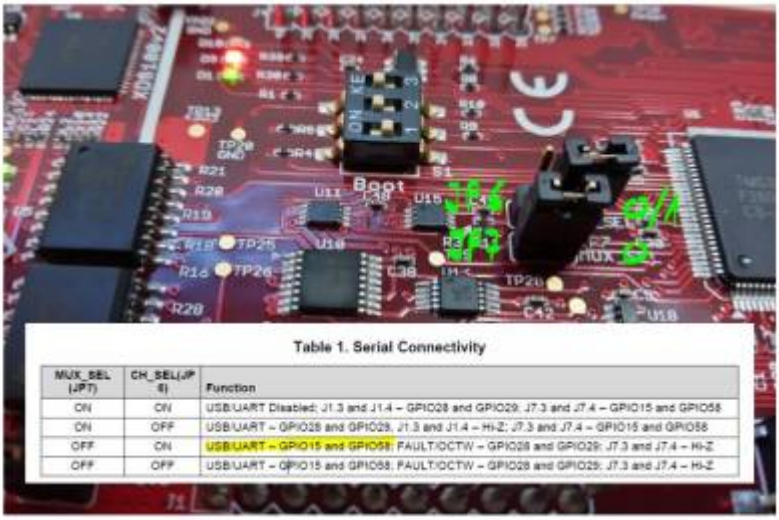



Table 1. Serial Connectivity

| MUX_SEL (JP7) | CH_SEL(JP6) | Function |
|---|---|---|
| ON | ON | USB:UART Disabled; J1.3 and J1.4 – GPIO28 and GPIO29; J7.3 and J7.4 – GPIO15 and GPIO58 |
| ON | OFF | USB:UART – GPIO28 and GPIO29, J1.3 and J1.4 – Hi-Z; J7.3 and J7.4 – GPIO15 and GPIO58 |
| OFF | ON | USB:UART – GPIO15 and GPIO58; FAULT/OCTW – GPIO28 and GPIO29; J7.3 and J7.4 – Hi-Z |
| OFF | OFF | USB:UART – GPIO15 and GPIO58; FAULT/OCTW – GPIO28 and GPIO29; J7.3 and J7.4 – Hi-Z |

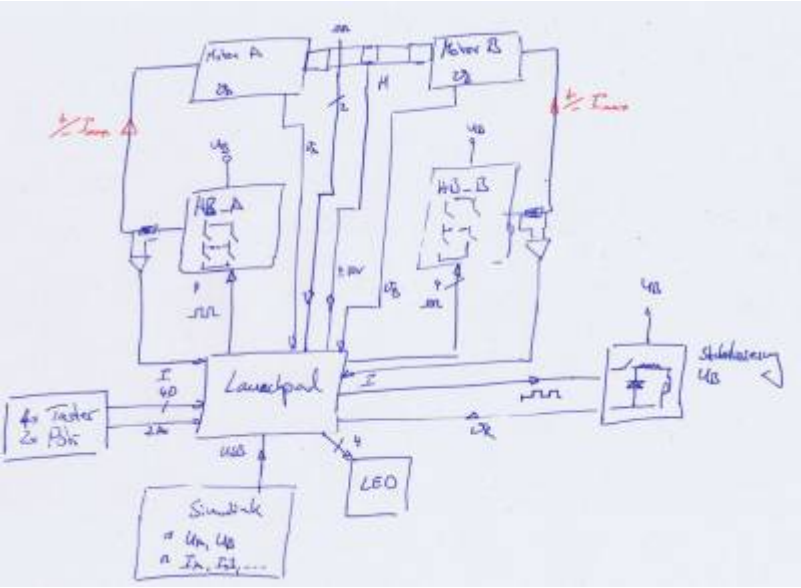
Demo-Download:

host-read.zip

---

## DC- Motor/Generator Prüfstand

Mit Hilfe von Simulink und dem TI-Launchpad werden zwei gekoppelte Gleichstrommaschinen getrennt voneinander angesteuert.




Hardware:

- Drehzahl / Drehmomentmessung an der Welle
- Strommessung (positiv, negativ) Motor A/B
- Spannung stellen Motor A/B
- Strom stellen Motor A/B (positiv, negativ), im Generatorbetrieb (Strom negativ) muss Leistung in Wärme umgesetzt werden
- Temperatur Motor A/B

Funktionsblöcke in Simulink:

- Konstante Spannung Motor (positiv, negativ) A/B (Strombegrenzung auf +/- Imax)
- Konstanter Motorstrom (positiv, negativ) A/B (Spannungsbegrenzung auf +/- Umax)
- Konstantes Drehmoment (positiv, negativ) A/B (Strombegrenzung)
- Lastsimulation, es können vorgegebene Drehmomentverläufe (z.B Fahrprofile) abgefahren werden
- Konstante Drehzahl A/B (Spannungsbegrenzung, Strombegrenzung)
- Konstante Position A/B (Spannungsbegrenzung, Strombegrenzung)

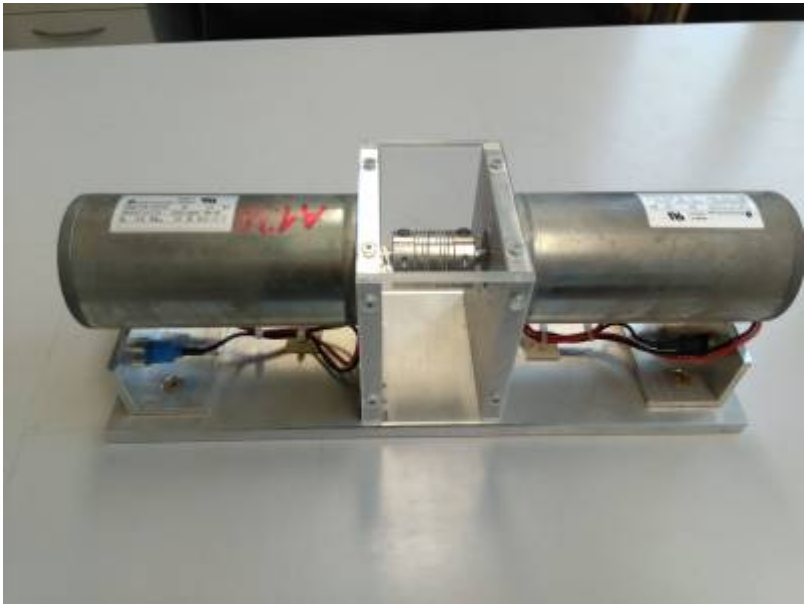
Versuchsdurchführung:

- Messung der Kennlinie n(M) Motor A, Parameter: Ausgangsspannung
- Messung der Kennlinie I(M) Motor A, Parameter: Ausgangsspannung
- Wirkungsgrad Motor A
- Hochlaufzeitzeit unter Last

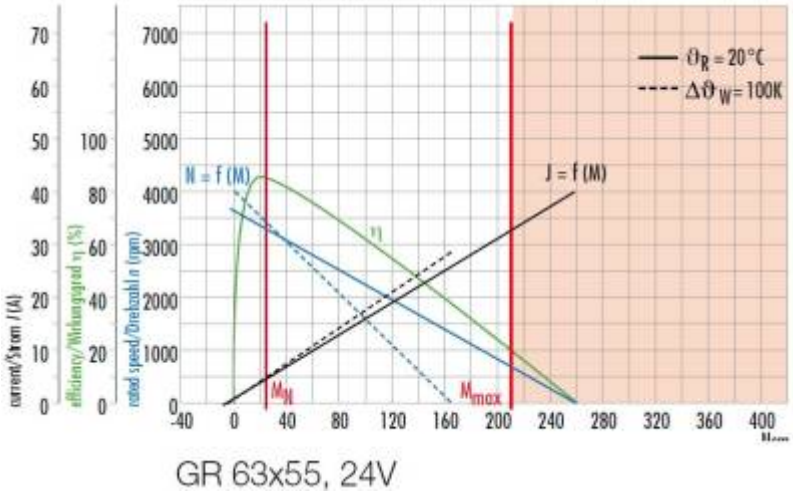
## Hardware-Blöcke

Parameter:

- Umax = 30V
- Imax = 20A



Das Bild zeigt den aktuellen Stand der gekoppelten Motoren. Datenblatt
Es fehlen noch der Encoder und die Drehmomentmessung.
Wellendurchmesser: 8mm, Technische Zeichnung
gr-63x55-8844201270.pdf




GR 63x55, 24V

Aus Datenblatt: Dunkermotoren.de Kenndaten eines Motors am Bemessungspunkt: Spannung: 24V Strom: 4,9A Drehzahl: 3350 1/min Drehmoment: 27Ncm Leistung 94,7W

Weitere Kenndaten: Leerlaufdrehzahl: 3650 1/min Leerlaufstrom: 0,4A Anlaufmoment (24V): 260Ncm Anlaufstrom 40A Entmagnetisierungsstrom 30A

Hardwarebegrenzung auf 20A ergibt ein maximales Drehmoment der Motoren von 128Ncm. Sicherheitsreserve für Impulsbelastung: Drehmomentmessung bis 300Ncm.

Nachfolgend ein Teil der notwendigen Hardwarekomponenten:

**Leistungswiderstand**

Der Widerstand wird verwendet damit im Genertorbetrieb die erzeugte Energie in Wärme umgesetzt werden kann, da unsere Netzteile im Allgemeinen nicht rückspeisefähig sind.
Beispiel für einzusetzenden Lastwiderstand:
https://de.farnell.com/arcol/hs100-r10-j/drahtwiderstand-0r1-5-axial/dp/2478150
Der Widerstand darf nicht zu groß gewählt werden, da ansonst nur für hohe Drehzahlen (große Uind)
ein außreichend großer Strom fließen kann.
Der Widerstand muss zusätzlich auf einem Kühlkörper montiert werden und wird mit einem Temperatursensor ausgestattet.

Die Ansteuerung des Widerstandes erfolgt über PWM (min. 50kHz) über das Launchpad, geregelt wird auf konstante Versorgungsspannung.

Der Leistungswiderstand wird in der ersten Version der Schaltung nicht weiter verfolgt, das stets zwei gekoppelte Motoren verwendet werden. Die Gefahr eines ungewollten Rückspeisens entsteht aufgrund der Verluste in der Leistungselektronik und den Motoren nicht.

**Spannungsversorgung**

Die Motoren müssen jeweils separat mit einer Spannung -Umax < = U < = +Umax angesteuert werden können.
Der maximale Laststrom wird auf +/-Imax begrenzt.
Die Ansteuerung der Spannungsregelung erfolgt über das Launchpad, geregelt wird auf konstante Ausgangsspannung.

**Drehmomentsensor**

Das an der Welle anliegende Drehmoment wird mittels einem Drehmomentsensor erfasst.
Analoger Messbereich +/-10V für z.B. Mmax = 1Nm
https://www.burster.de/de/drehmomentsensoren/p/detail/8661/

Nach den ersten Anfragen bei Herstellen von Drehmomentsensoren ist klar, dass sich die Messwellen im Bereich von > 2000€ absipielen → viel zu teuer.
Wir werden daher den Antriebsmotor frei lagern und über einen Hebelarm die Kraft auf einen Hebelarm messen.

**Encoder**

Zur Drehzahl und Drehrichtungserkennung muss ein Encoder zum Einsatz kommen.
Der Encoder RE30/500 ist mit 5V Impulsausgängen A, B und einem Indexausgang ausgestattet.

# Pinbelegung Launchpad Hardware V01

| **Funktion** | **Analog / Digital** | **Anpassung** | **Launchpad** | Launchpad Pin | Schaltplanlabel | **Sonstiges, Schutzbeschaltung** |
|---|---|---|---|---|---|---|
| PWM Out Motor A<br>HB_A_Links_HS<br>HalbBrücke_A_Links_HighSide | D | Logikpegel Treiber | 1PWM_AH<br>EPWM1A | 40 | M1B1H_IN | Todzeiteinstellung für jede Halbbrücken im µC ermöglichen |
| PWM Out Motor A<br>HB_A_Links_LS | D | Logikpegel Treiber | 1PWM_AL<br>EPWM1B | 39 | M1B1L_IN | |
| PWM Out Motor A<br>HB_A_Rechts_HS | D | Logikpegel Treiber | 1PWM_BH<br>EPWM2A | 38 | M1B2H_IN | |
| PWM Out Motor A<br>HB_A_Rechts_LS | D | Logikpegel Treiber | 1PWM_BL<br>EPWM2B | 37 | M1B2L_IN | |
| PWM Out Motor A*<br>HB_A*_HS | D | Logikpegel Treiber | 1PWM_CH<br>EPWM3A | 36 | M1B3H_IN | Vorhalt für BLDC Ansteuerung Motorport A |
| PWM Out Motor A*<br>HB_A*_LS | D | Logikpegel Treiber | 1PWM_CL<br>EPWM3B | 35 | M1B3L_IN | Vorhalt für BLDC Ansteuerung Motorport A |
| PWM Out Motor B<br>HB_B_Links_HS | D | Logikpegel Treiber | 2PWM_AH<br>EPWM4A | 80 | M2B1H_IN | |
| PWM Out Motor B<br>HB_B_Links_LS | D | Logikpegel Treiber | 2PWM_AL<br>EPWM4B | 79 | M2B1L_IN | |
| PWM Out Motor B<br>HB_B_Rechts_HS | D | Logikpegel Treiber | 2PWM_BH<br>EPWM5A | 78 | M2B2H_IN | |
| PWM Out Motor B<br>HB_B_Rechts_LS | D | Logikpegel Treiber | 2PWM_BL<br>EPWM5B | 77 | M2B2L_IN | |
| PWM Out Motor B*<br>HB_B*_HS | D | Logikpegel Treiber | 2PWM_CH<br>EPWM6A | 76 | M2B3H_IN | Vorhalt für BLDC Ansteuerung Motorport A |
| PWM Out Motor B*<br>HB_B*_LS | D | Logikpegel Treiber | 2PWM_CL<br>EPWM6B | 75 | M2B3L_IN | Vorhalt für BLDC Ansteuerung Motorport A |
| Strommessung Motorstrom A<br>Messbereich +/- 20A | A | ACS712 | ADCINA0 | 27 | Strom_M1_33 | |
| Strommessung Motorstrom B<br>Messbereich +/- 20A | A | ACS712 | ADCINB6 | 6 | Strom_M2_33 | |
| Versorgungspannung +15V | A | | ADCINB2 | 26 | +15V_33 | Überwachung der Versorgungsspannung, |
| Batteriespannung | A | | ADCINA2 | 25 | Vbat*_33 | |
| Versorgungsspannung +3,3V | A | | ADCINB1 | 24 | +3,3V | |
| Drehmomentmessung I<br>Kraftsensor 1 | A | Spannungsteiler | ADCINA6 | 2 | Moment_Pos_33 | |
| Drehmomentmessung II<br>Kraftsensor 1 | A | Spannungsteiler | ADCINA7 | 23 | Moment_Neg_33 | |
| Encoder Pulse | D<br>Interrupt 1 | Logikpegel anpassen | GPIO19<br>ECAP1 | 19 | Encoder-HallA_Interrupt_33 | Jumper BLDC Hall-Sensoren,<br>Wird auch als ECAP1 benutzt |
| Encoder Drehrichtung Rechts | D | Logikpegel anpassen | GPIO44 | 18 | Rechts_33 | |
| Encoder Drehrichtung Links | D | Logikpegel anpassen | GPIO14 | 47 | Links_33 | |
| Encoder Index | D<br>Interrupt 2 | Logikpegel anpassen | GPIO25<br>ECAP2 | 54 | Index_HallB_Interrupt_33 | Jumper |
| Vorhalt Hallsensor BLDC | D<br>Interrupt 3 | Logikpegel anpassen | GPIO26<br>ECAP3 | 58 | HallC_33 | BLDC Hall-Sensoren |
| Bafang Speed-Signal | D | Logikpegel anpassen | GPIO16 | 15 | Speed_33 | |
| Taster 1<br>für manuelle Ansteuerung | D | Pull-Up | GPIO12 | 5 | T1 | |
| Taster 2<br>für manuelle Ansteuerung | D | Pull-Up | GPIO22 | 8 | T2 | |

| **Funktion** | **Analog / Digital** | **Anpassung** | **Launchpad** | Launchpad Pin | Schaltplanlabel | **Sonstiges, Schutzbeschaltung** |
|---|---|---|---|---|---|---|
| Taster 3<br>für manuelle Ansteuerung | D | Pull-Up | GPIO13 | 34 | T3 | Achtung: Liegt auf Enable-Boot. Ggf. problematisch |
| Potentiometer A | A | | ADCINB7 | 63 | Poti 1 | Manuelle Drehzahlstellung Motor A<br>(frei Programmierbar in Simulink) |
| Potentiometer B | A | | ADCINB4 | 64 | Poti 2 | |
| Temperatur Motor A | A | LM355 mit Impedanzwandler | ADCINA5 | 65 | Temperatur1_33 | |
| Temperatur Motor B | A | LM355 mit Impedanzwandler | ADCINB5 | 66 | Temperatur2_33 | |
| LED-Anzeige | D | | GPIO51 | 12 | LED_Rot | |
| LED-Anzeige | D | | GPIO27 | 59 | LED_Grün | |
| Display i2C, SCL | D | | SCL<br>GPIO 33 | 9 | SCL | |
| Display i2C, SDA | D | | SDA<br>GPIO 32 | 10 | SDA | |
| Analoge Erweiterung 1 | A | | ADCINB0 | 28 | | |
| Analoge Erweiterung 2 | A | | ADCINA1 | 29 | | |

Aktualisiert 05.04.2020:
Die aktuelle Schaltung mit PCB zum Download in der ersten Version:

dcdc_v01_update.zip

Das Layout zur Inbetriebnahme der Schaltung in Simulink ermöglicht einen schnellen Test aller IOs.

Folgende Tests können durchgeführt werden:

- LEDs aktivieren
- Externe Schalterposition einlesen
- Pegel der Hall und Encodersignale einlesen
- Zeit zwischen zwei Hall-Flanken ermitteln
- Test aller Halbbrücken
- Analog-Werte einlesen und umwandeln

Download der Inbetriebnahme-Software

inbetriebnahme_v05.zip

Fazit und Erkenntnisse während des Aufbaus der Inbetriebnahmesoftware:

- Compiler und Flash-Vorgang dauert sehr lange.
- Dashboard Anzeigeelemente funktionieren leider nicht. Zur Anzeige stehen nur Display und Scopes zur Verfügung
- Das eCap Modul zur Messung der Periodendauer (Encoder) läuft nur alleine. Eine gleichzeitige Portabfrage über GPIO DI Pin ist nicht möglich

## Pinbelegung Launchpad Hardware V02

Die Umsetzung der ersten Hardware V01 hat nahezu problemlos funktioniert. Für die Hardware der zweiten Generation werden folgende Änderungen vorgenommen:

| Nr. | Änderungen zu V01 |
|---|---|
| 1 | Falsches Label zur Spannungsmessung 3,3V, daher keine Verbindung auf dem PCB |
| 2 | 3x Zusätzliche Stromsensoren zur Messung der Phasenströme Motor A |
| 3 | Wechsel auf IR2110 Treiber mit SD Eingang, da die PWM ansonsten nicht deaktiviert werden kann.<br>PWM 0% bedeutet die Low-Side Mosfets sind dauerhaft geschlossen womit kein<br>Freilauf möglich ist. SD Pin deaktiviert die Brücken. |
| 4 | Ausgabe der Encoder Impulse über Jumper optional auf Hall C.<br>Damit kann später auf Position geregelt werden. Simulink erlaubt keine doppelte Pinfunktion z.B. eCAP und gleichzeitig einen Interrupt |
| 5 | Neuer Drehmomentsensor mit 50N, da aktuelle Version mit 10N zu gering |
| 6 | Taster 1 und 2 werden auf ADC-Ports gesetzt, damit mehr GPIO freiwerden |
| 7 | Analoge Erweiterungenports entfallen |

Pinbelegung für V02

| Funktion | Analog / Digital | Anpassung | Launchpad | Launchpad Pin | Schaltplanlabel | Sonstiges, Schutzbeschaltung |
|---|---|---|---|---|---|---|
| PWM Out Motor A<br>HB_A_Links_HS<br>HalbBrücke_A_Links_HighSide | D | Logikpegel Treiber | 1PWM_AH<br>EPWM1A | 40 | M1B1H_IN | Todzeiteinstellung für jede Halbbrücken im µC ermöglichen |
| PWM Out Motor A<br>HB_A_Links_LS | D | Logikpegel Treiber | 1PWM_AL<br>EPWM1B | 39 | M1B1L_IN | |
| **SD A Links** | | Logikpegel Treiber | **GPIO50** | **13** | **M1B1_SD** | **Motor 1 Brücke 1 SD** |
| PWM Out Motor A<br>HB_A_Rechts_HS | D | Logikpegel Treiber | 1PWM_BH<br>EPWM2A | 38 | M1B2H_IN | |
| PWM Out Motor A<br>HB_A_Rechts_LS | D | Logikpegel Treiber | 1PWM_BL<br>EPWM2B | 37 | M1B2L_IN | |
| **SD A Rechts** | | Logikpegel Treiber | **GPIO24** | **55** | **M1B2_SD** | **Motor 1 Brücke 2 SD** |
| PWM Out Motor A*<br>HB_A*_HS | D | Logikpegel Treiber | 1PWM_CH<br>EPWM3A | 36 | M1B3H_IN | Vorhalt für BLDC Ansteuerung Motorport A |
| PWM Out Motor A*<br>HB_A*_LS | D | Logikpegel Treiber | 1PWM_CL<br>EPWM3B | 35 | M1B3L_IN | Vorhalt für BLDC Ansteuerung Motorport A |
| **SD A*** | | Logikpegel Treiber | **GPIO52** | **53** | **M1B3_SD** | **Motor 1 Brücke 3 SD** |
| PWM Out Motor B<br>HB_B_Links_HS | D | Logikpegel Treiber | 2PWM_AH<br>EPWM4A | 80 | M2B1H_IN | |
| PWM Out Motor B<br>HB_B_Links_LS | D | Logikpegel Treiber | 2PWM_AL<br>EPWM4B | 79 | M2B1L_IN | |
| **SD B Links** | D | Logikpegel Treiber | **GPIO53** | **52** | **B2B1_SD** | **Motor 2 Brücke 1 SD** |
| PWM Out Motor B<br>HB_B_Rechts_HS | D | Logikpegel Treiber | 2PWM_BH<br>EPWM5A | 78 | M2B2H_IN | |
| PWM Out Motor B<br>HB_B_Rechts_LS | D | Logikpegel Treiber | 2PWM_BL<br>EPWM5B | 77 | M2B2L_IN | |
| **SD B Rechts** | D | Logikpegel | **GPIO12** | **5** | **M2B2_SD** | **Motor 2 Brücke 2 SD** |
| PWM Out Motor B*<br>HB_B*_HS | D | Logikpegel Treiber | 2PWM_CH<br>EPWM6A | 76 | M2B3H_IN | Vorhalt für BLDC Ansteuerung Motorport A |
| PWM Out Motor B*<br>HB_B*_LS | D | Logikpegel Treiber | 2PWM_CL<br>EPWM6B | 75 | M2B3L_IN | Vorhalt für BLDC Ansteuerung Motorport A |

| **Funktion** | **Analog / Digital** | **Anpassung** | **Launchpad** | Launchpad Pin | Schaltplanlabel | **Sonstiges, Schutzbeschaltung** |
|---|---|---|---|---|---|---|
| **SD B*** | D | Logikpegel Treiber | **GPIO22** | **8** | **M2B3_SD** | **Motor 2 Brücke 3 SD** |
| Strommessung Motorstrom A Messbereich +/- 20A | A | ACS712 | ADCINA0 | 27 | Strom_M1_33 | |
| **Strommessung Motor 1 Brücke 1** | A | ACS712 | **ADCINA3** | **67** | **Strom_M1_B1_33** | **Phasenstrom Motor 1 B1** |
| **Strommessung Motor 1 Brücke 2** | A | ACS712 | **ADCINB3** | **68** | **Strom_M1_B2_33** | **Phasenstrom Motor 1 B2** |
| **Strommessung Motor 1 Brücke 3** | A | ACS712 | **ADCINA4** | **69** | **Strom_M1_B3_33** | **Phasenstrom Motor 1 B3** |
| Strommessung Motorstrom B Messbereich +/- 20A | A | ACS712 | ADCINB6 | 6 | Strom_M2_33 | |
| Versorgungspannung +15V | A | | ADCINB2 | 26 | +15V_33 | Überwachung der Versorgungsspannung, |
| Batteriespannung | A | | ADCINA2 | 25 | Vbat*_33 | |
| Versorgungsspannung +3,3V | A | | ADCINB1 | 24 | +3,3V | |
| Drehmomentmessung I Kraftsensor 1 | A | Spannungsteiler | ADCINA6 | 2 | Moment_Pos_33 | |
| Drehmomentmessung II Kraftsensor 1 | A | Spannungsteiler | ADCINA7 | 23 | Moment_Neg_33 | |
| Encoder Pulse ECPA Modul | D Interrupt 1 | Logikpegel anpassen | GPIO19 ECAP1 | 19 | Encoder-HallA_Interrupt_33 | Jumper BLDC A Hall-Sensor oder ECAP1 Modul |
| Encoder Drehrichtung Rechts | D | Logikpegel anpassen | GPIO44 | 18 | Rechts_33 | |
| Encoder Drehrichtung Links | D | Logikpegel anpassen | GPIO14 | 47 | Links_33 | |
| Encoder Index | D Interrupt 2 | Logikpegel anpassen | GPIO25 ECAP2 | 54 | Index_HallB_Interrupt_33 | Jumper BLDC B Hall-Sensor oder Encoder Index |
| **Vorhalt Hallsensor BLDC** | D Interrupt 3 | Logikpegel anpassen | GPIO26 ECAP3 | 58 | ~~HallC_33~~ **Encoder-HallC_Interrupt_33** | **Jumper** BLDC Hall-Sensoren oder **Encodersignal zur Positionsbestimmung** |
| Bafang Speed-Signal | D | Logikpegel anpassen | GPIO16 | 15 | Speed_33 | |
| **Taster 1 für manuelle Ansteuerung** | D | Pull-Up | ~~GPIO12~~ **ADCINB0** | ~~5~~ **28** | T1 | **Keine GPIO mehr übrig. Lösung über ADC Ports** |
| **Taster 2 für manuelle Ansteuerung** | D | Pull-Up | ~~GPIO22~~ **ADCINA1** | ~~8~~ **29** | T2 | **Keine GPIO mehr übrig. Lösung über ADC Ports** |
| Taster 3 für manuelle Ansteuerung | D | Pull-Up | GPIO13 | 34 | T3 | |
| Potentiometer A | A | | ADCINB7 | 63 | Poti 1 | Manuelle Drehzahlstellung Motor A (frei Programmierbar in Simulink) |
| Potentiometer B | A | | ADCINB4 | 64 | Poti 2 | |
| Temperatur Motor A | A | LM355 mit Impedanzwandler | ADCINA5 | 65 | Temperatur1_33 | |
| Temperatur Motor B | A | LM355 mit Impedanzwandler | ADCINB5 | 66 | Temperatur2_33 | |
| LED-Anzeige | D | | GPIO51 | 12 | LED_Rot | |
| LED-Anzeige | D | | GPIO27 | 59 | LED_Grün | |
| Display i2C, SCL | D | | SCL GPIO 33 | 9 | SCL | |
| Display i2C, SDA | D | | SDA GPIO 32 | 10 | SDA | |
| ~~Analoge Erweiterung 1~~ | A | | ~~ADCINB0~~ | ~~28~~ | | |
| ~~Analoge Erweiterung 2~~ | A | | ~~ADCINA1~~ | ~~29~~ | | |

## Hardware Version DCDC_V02 zum Download:

dcdc_v02.zip

Leider habe ich es in KiCad nicht geschafft alle Projektdateien umzubenennen da ansonsten ständig die Footprints verlorengehen.
Daher bleibt die Bezeichnung der Schaltpläne auf V01. KiCad ist hier leider nicht sehr benutzerfreundlich ..... (oder ich kenne mich zu wenig aus)
Schaltplan als pdf:

dcdc_v02.pdf

Software zur Inbetriebnahme der DCDC_V02 Hardware:

inbetriebnahme_v07_4_dcdc02.zip

## Pinbelegung Launchpad Hardware V03

Aktualisiert: 15.04.2021
Die Pinbelegung zu V02 hat sich nicht geändert. Allerdings wurden einige Fehler behoben und die Schaltung optimiert:

| Nr. | Änderungen zu V02 |
|---|---|
| 1 | 3,3V für Stand-Alone Betrieb wird über einen LM2574 generiert (alt Spannungsregler) |
| 2 | Kraftsensoren werden über 3,3V angesteuert (Entfall Spannungsteiler) |
| 3 | Temperatursensoren werden mit 3,3V versorgt (Entfall Spannungsteiler) |
| 4 | Wechsel der Stromsensoren auf ACS725 (Entfall Spannungsteiler) |
| 5 | Impedanzwandler werden mit 3,3V versorgt (Entfall Spannungsteiler) |

Hardware V02 war noch stark geprägt von der 5V Welt. Unnötigerweise mussten alle Signale zum µC dadurch über Spannungsteiler reduziert werden wodurch zusätzliche Bauteile und Messfehler entstehen.
V03 versucht daher möglichst viele Signale und Peripherie auf der 3,3V Ebene anzusteuern.

Schaltplan:

schaltplan_dcdc_v03.pdf

KiCAD-Projekt mit Bestellliste:

dcdc_v03_kicad.zip

Bekannte Fehler in V03:

- Fehler in der Temperaturmessung, muss überarbeitet werden

# Pinbelegung Launchpad Hardware V04

Aktualisiert: 15.01.2023

Die Hardware V04 beinhaltet eine Reihe an Änderungen welche die Schaltung deutlich robuster machen soll gegenüber ungewolltes Verpolen der Eingangsspannung sowie eventuell auftretenden Überspannungen. Die Spannungsversorgung wurde optimiert und eine visuelle Anzeige der Gate-Signale mit eingefügt.

Schaltplan:

schaltplan_dcdc_v04.pdf

KiCAD Projekt:

dcdc_v04_kicad.zip

| **Nr.** | **Änderungen zu V03** |
|---|---|
| 1 | Neue Bauteilbezeichnung |
| 2 | Separate Spannungsversorgung der Treiber möglich über VBat. Dadurch auch ZK-Spannungan an UBat < 15V möglich.<br>Falls die ZK-Spannung (UBat) unter VBat sinkt wird automatisch auf VBat gewechselt |
| 3 | |
| 4 | |
| 5 | |
| **Nr.** | **Fehler in V04** |
| 1 | Neue Bauteilbezeichnung |
| 2 | Separate Spannungsversorgung der Treiber möglich über VBat. Dadurch auch ZK-Spannungan an UBat < 15V möglich.<br>Falls die ZK-Spannung (UBat) unter VBat sinkt wird automatisch auf VBat gewechselt |
| 3 | |
| 4 | |
| 5 | |