Bachelorarbeit

Benedikt Johannsen

Design und Implementierung einer extern und dynamisch konfigurierbaren ECU auf Basis der MegaSquirt

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Benedikt Johannsen

Design und Implementierung einer extern und dynamisch konfigurierbaren ECU auf Basis der MegaSquirt

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Technische Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. Franz Korf
Zweitgutachter : Prof. Dr. rer. nat. Stephan Pareigis

Abgegeben am 7. Juni 2009

Benedikt Johannsen

Thema der Bachelorarbeit

Design und Implementierung einer extern und dynamisch konfigurierbaren ECU auf Basis der MegaSquirt

Stichworte

Motorsteuerung, Engine Control Unit, ECU, Verbrennungsmotor, Zündung, Einspritzung; MegaSquirt; CAN, TTCAN, Time Triggered Bus; Echtzeit; Mikrokontroller, Freescale, MC9S12; dynamische Konfiguration; Hawks Racing, Formula Student

Kurzzusammenfassung

Gegenstand dieser Bachelorarbeit ist die Entwicklung einer Motorsteuerung für einen Verbrennungsmotor, die im Hawks Racing Team eingesetzt werden soll. Als Basis für die Entwicklung dient die MegaSquirt-Motorsteuerung. Diese wird auf ihre Eignung hinsichtlich der Anforderungen analysiert und entsprechend angepasst werden. Desweiteren wird die Motorsteuerung um eine dynamische Konfiguration über einen Bus erweitert, sodass während des Betriebs direkt Einfluss auf Betriebsparameter genommen werden kann. Dies kann perspektivisch für den Einsatz von Fahrerassistenzsystemen genutzt werden.

Benedikt Johannsen

Title of the paper

Design and implementation of an ECU with external and dynamic configuration based on MegaSquirt

Keywords

ECU, Engine Control Unit, combustion engine, ignition, injection; MegaSquirt; CAN, TTCAN, time triggered Bus; realtime; Microcontroller, Freescale, MC9S12; dynamic configuration; Hawks racing, Formula Student

Abstract

Subject of this bachelor thesis is the development of an engine control unit (ECU) for a combustion engine, to be applied by the Hawks Racing Team. The development is based on the MegaSquirt-ECU, which is analized for its suitability and adjusted according to the requirements. Furthermore, the ECU is extended by a dynamic configuration via bus system, so that operating parameters can be manipulated directly during operation. This can be used for future applications of driver assistance systems.

Inhaltsverzeichnis

1. Einführung

1.1. Motivation

Die Motorsteuerung, kurz ECU[1], stellt bei heutigen Kraftfahrzeugen eine zentrale Komponente dar. Sie ist elementarer Dreh- und Angelpunkt für die Bestrebungen, einen Verbrennungsmotor effizienter und ökonomischer nutzen zu können. Die Steuerung und Kontrolle der Betriebsparameter hat sich stark weiterentwickelt. Anfangs wurden diese durch mechanische Komponenten regelt, doch mit dem Aufkommen der Elektronik übernahmen mehr und mehr elektronische Baugruppen diese Funktion, da sie diese genauer und schneller erledigen konnten.

Abbildung 1.1.: Hawks H04 in Hockenheim 2008
Hanselmann, FSG

Besonders für einen Rennwagen ist die Motorsteuerung ein bedeutsames Thema. Sie ermöglicht es den Motortechnikern, den Motor optimal auf die maximale Leistung bei möglichst geringem Verbrauch abzustimmen und zu konfigurieren.

Im Rahmen dieser Arbeit wird, basierend auf einer OpenSource-ECU, eine Motorsteuerung für das Formula Student Team „Hawks-Racing“ entwickelt.

[1] Engine Control Unit

1.2. Hawks-Racing und Formula Student

Die Formula Student[2] ist eine Möglichkeit für studentische Teams, mit selbstkonstruierten Rennwagen in Wettbewerbsveranstaltungen gegeneinander anzutreten. Ziel ist es, den Studenten neben dem Studium die Möglichkeit zu bieten, Praxiserfahrung in Konstruktion und Fertigung zu sammeln und Kontakte in die Automobilindustrie zu knüpfen.

Die Wettbewerbe bewerten neben den Fahrleistungen in sogenannten „dynamischen“ Disziplinen auch Aspekte wie das Designkonzept und die Entwicklungskosten. Diese sogenannten „statischen“ Disziplinen gehen ebenso in die Wertung mit ein.

Abbildung 1.2.: Logo Hawks Racing

Abbildung 1.3.: Logo Formula Student Germany
Quelle: http://www.formulastudent.de

Das Team Hawks-Racing besteht seit 2003 und besteht derzeit aus 40 Studenten unterschiedlichster Fachrichtungen. Bisher wurden vier Rennwagen konstruiert, gefertigt und bei Wettbewerben der Formula Student erfolgreich eingesetzt. Während der Entstehung dieser Arbeit lief die Entwicklung des fünfte Wagens. Hier wird wieder eine ECU des italienischen Herstellers Walbro eingesetzt. Diese läuft zwar zuverlässig, lässt sich aber nur bedingt in das umfangreiche Telemetrie-System integrieren.

1.3. Zielsetzung

Diese Arbeit soll diesen Umstand ändern und die zentrale elektronische Komponente vollständig in das Telemetriesystem integrieren. Hierzu wird die sogenannte „MegaSquirt“-ECU untersucht um einen Eindruck zu gewinnen, wie eine Motorsteuerung aufgebaut sein kann. Es wird weiterhin geprüft, inwieweit diese ECU

[2] http://www.formulastudent.de/

den Ansprüchen des Hawks-Teams genügt und wo Anpassungsbedarf besteht.
Die ECU soll dynamisch konfigurierbar aufgebaut sein. Dies bedeutet, dass während des Betriebs Einfluss auf die Funktionen und die Betriebsparameter genommen werden kann, um die Motorsteuerung besser an die jeweilige Fahrsituation abpassen zu können. So kann direkt Einfluss auf die Motorleistung oder auch den Kraftstoffverbrauch genommen werden, je nachdem, welche Ansprüche Priorität haben.

So lassen sich auch Fahrassistenzsysteme wie eine Antriebsschlupfregelung (ASR) implementieren. Eine ASR würde so auf Basis der gemessenen Raddrehzahlen erkennen, wenn die Antriebsräder durchdrehen und so Motorleistung verschenkt wird. In diesem Fall würde die Regelung die Motorleistung reduzieren. Der einfachste Weg, dies zu erreichen, wäre ein Eingriff in die ECU [vgl. Kolbe, 2008].

Zudem würde eine eigenentwickelte ECU im sich sogenannten Design-Report der Formula Student positiv bemerkbar machen und so wertvolle Punkte für das Hawks-Team gewinnen.

1.4. Gliederung der Arbeit

Im Kapitel Grundlagen der Motortechnik werden die elementaren Konzepte der Motortechnik dargestellt, die zum Entwurf einer ECU unentbehrlich sind.
Das darauffolgende Kapitel behandelt die Anforderungen und Rahmenbedingungen an diese Arbeit und an die neue Motorsteuerung. In der Analyse der MegaSquirt-ECU wird aufgezeigt inwieweit die MegaSquirt-ECU diese Anforderungen bereits erfüllt und wo Handlungsbedarf besteht.
Die Lösungsmöglichkeiten werden in der Konzeption erläutert und diskutiert. Ihre Umsetzung ist im Kapitel Realisierung beschrieben.

2. Grundlagen der Motortechnik

Zur Entwicklung einer Motorsteuerung für einen Verbrennungsmotor sind Kenntnisse in Motor- und Fahrzeugtechnik unabdingbar. In diesem Kapitel werden die elementaren Grundlagen erläutert.

Der zurzeit im Hawks-Projekt verwendete Motor ist ein 4-Zylinder Viertakt-Ottomotor mit Saugrohreinspritzung[1]. Da dies zudem ein gebräuchliches Motorkonzept ist, wird der Schwerpunkt auf dieser Technik liegen und alternative Möglichkeiten werden nicht tiefergehend behandelt.

Zur Begriffsklärung dient die Definition eines Ottomotors[2]:

Definition Ottomotor laut Robert Bosch GmbH [2005]

> *Ein Ottomotor ist ein fremdgezündeter Verbrennungsmotor, der ein Luft-Kraftstoff-Gemisch verbrennt und damit die im Kraftstoff enthaltene chemische Energie in kinetische Energie umwandelt.*

Fremdzündung bedeutet hierbei, dass das Gemisch nicht durch Selbstentzündung gezündet wird, wie etwa bei einem Dieselmotor, sondern durch einen Zündfunken. Die Verbrennung des Gemischs treibt einen Kolben in einem Zylinder in eine periodische Auf- und Abwärtsbewegung. Diese Bewegung wird über Pleuelstangen in die Rotationsbewegung einer Welle umgesetzt. Daher auch der Name Hubkolbenmotor. [vgl. Robert Bosch GmbH, 2005, S.16ff.]

2.1. Aufbau des Motors

Im Zentrum eines 4-Takt-Motors steht die Brennkammer des Zylinders. Sie wird von einem beweglichen Koblen abgeschlossen. Der Kolben ist über eine Pleuelstange mit der Kurbelwelle verbunden und setzt so seine Auf- und Abbewegung in eine Drehbewegung der Kurbelwelle um.

[1] Kawasaki ZX600J

[2] benannt nach seinem Entwickler Nikolaus August Otto, erfunden 1878

Die Zuführung von Kraft- und Sauerstoff erfolgt als Gas-Gemisch über den Einlasskanal. Dessen Zugang zur Brennkammer wird durch das Einlassventil geregelt. Das Ausstoßen von Abgasen erfolgt durch den Abgaskanal, der durch das Auslassventil von der Brennkammer getrennt ist. Die Ventile werden durch eine Nockenwelle betätigt.
Die Zündung des Gemischs erfolgt über eine elektrische Zündkerze. Der Aufbau wird in Abbildung 2.1 veranschaulicht.

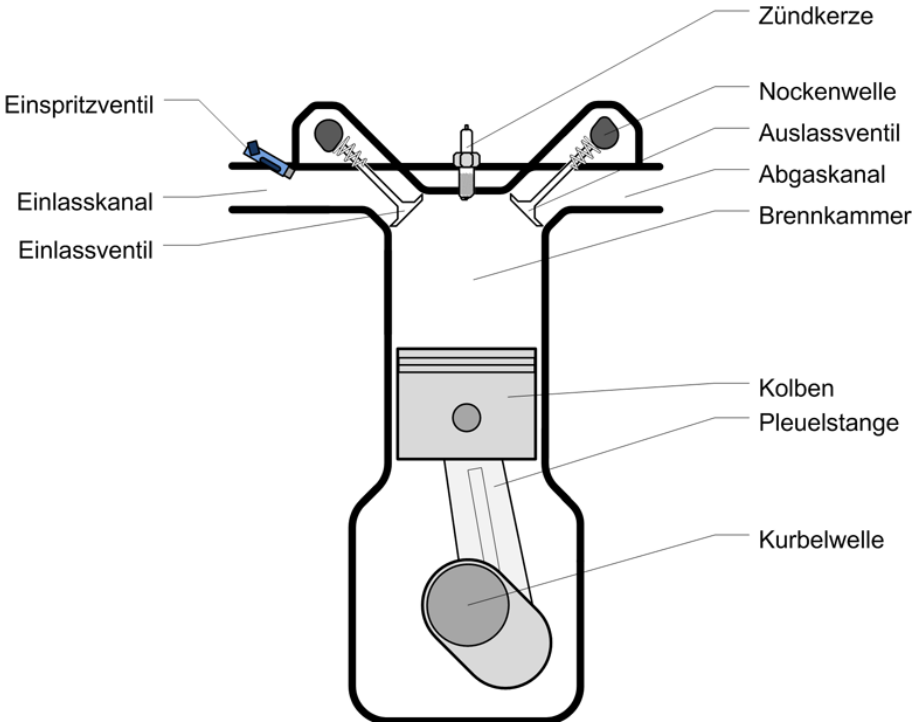

Abbildung 2.1.: Aufbau 4-Takt-Motor

Zum Verständnis der folgenden Kapitel sind folgende Begriffe zu definieren:

Oberer Totpunkt: Dies bezeichnet die Position der Kurbelwelle, bei der der Kolben des ersten Zylinders seine maximale Höhe erreicht hat und von der Aufwärts- in die Abwärtsbewegung übergeht. Der Begriff wird ebenfalls verwendet, um die Position der Kurbelwelle für den gesamten Motor zu beschreiben. Auch hierbei bezieht er sich auf den ersten Zylinder.

Grad Kurbelwelle: Die Stellung der Kurbelwelle wird „°KW“, Grad Kurbelwelle, angegeben und bezieht sich auf den Oberen Totpunkt.

Nockenwelle: Die Nockenwelle dreht sich mit der halben Geschwindigkeit der Kurbelwelle. Eine Umdrehung der Nockenwelle entspricht folglich 720°KW.

2.2. 4-Takt-Zyklus

Das Arbeiten eines Ottomotors ist in mehrere Takte unterteilt. Hierbei unterscheidet man zwischen Zweitakt- und Viertakt-Motoren. Der Zweitaktmotor wird heute nur noch in sehr kleinen Motoren wie beispielweise für Rasenmäher oder ähnliche Arbeitsgeräte oder aber sehr große, langsamdrehende Motoren wie zum Beispiel Schiffsdiesel verwendet und spielt in der heutigen Fahrzeugtechnik daher nur noch eine untergeordnete Rolle. Ein Grund hierfür sind die strengen Abgasnormen, die im 2-Takt Betrieb nur mit sehr großem Aufwand einzuhalten sind. [vgl. Robert Bosch GmbH, 2003, S.472]

Einen kompletten Zyklus durch alle Takte bezeichnet man als „Arbeitsspiel“. Jeder Takt braucht eine halbe Umdrehung der Kurbelwelle, sprich 180°KW. Somit benötigt ein komplettes Arbeitsspiel im Vier-Takt-Betrieb 720°KW, zwei Umdrehungen der Kurbelwelle. Die Umdrehungen bezeichnet man hierbei als „Phase“. Die erste Phase umfasst die ersten beiden, die zweite Phase die letzten beiden Takte. Um also den aktuellen Takt eines Zylinders eindeutig zu identifizieren, reicht die Kurbelwellen-Stellung alleine nicht aus. Hierzu muss ein zusätzlicher Nockenwellensensor genutzt werden. [vgl. Brandt, 2008, S.25]

Im Kraftfahrzeugbereich werden hauptsächlich Mehr-Zylinder-Motoren genutzt. Dabei befinden sich die einzelnen Zylinder zu jedem Zeitpunkt in unterschiedlichen Takten des Arbeitsspiels, um eine gleichmäßigere Bewegung und Kraftverteiltung sowie eine bessere Laufruhe zu erzielen.

Die auf der nächsten Seite folgende Beschreibungen der Takte sind idealisiert. Einflüsse wie die Verbrennungsgeschwindigkeit des Gemisches werden hier nicht berücksichtigt.

Die Arbeitstakte eines Viertakt-Motors sind:

Takt 1: Ansaugen
Sobald der Kolben den oberen Totpunkt erreicht hat, wird das Einlassventil geöffnet. Durch die Abwärtsbewegung des Kolbens entsteht ein Unterdruck, der das im Einlasskanal befindliche Gemisch in den Zylinder saugt.
Sobald der Kolben den unteren Totpunkt erreicht, wird das Einlassventil geschlossen.

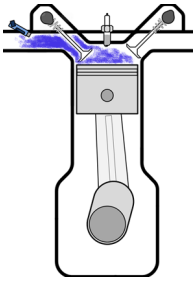

Abbildung 2.2.: Takt 1: Ansaugen

Takt 2: Verdichten
Hat der Kolben den unteren Totpunkt erreicht, beginnt der Verdichtungstakt. Hier sind die Ventile geschlossen und das Gemisch wird durch die Aufwärtsbewegung des Kolbens verdichtet. Dies dient der Erhöhung der Verbrennungsgeschwindigkeit im folgenden Takt.

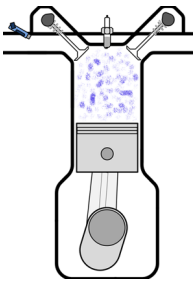

Abbildung 2.3.: Takt 2: Verdichten

Takt 3: Arbeiten
Sobald der Kolben den oberen Totpunkt überschritten hat, wird das Gemisch durch einen Zündfunken der Zündkerze gezündet. Der entstehende Druck treibt den Kolben nach unten.

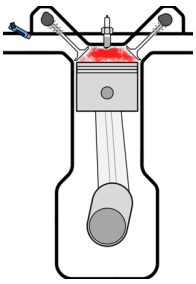

Abbildung 2.4.: Takt 3: Arbeiten

Takt 4: Ausstoßen
Ist der untere Totpunkt erreicht, wird das Auslassventil geöffnet und die Abgase durch die Aufwärtsbewegung des Kolbens aus dem Zylinder gepresst.

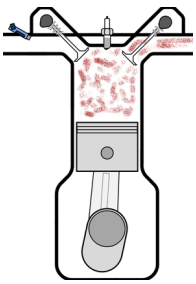

Abbildung 2.5.: Takt 4: Ausstoßen

2.3. Ladungswechsel

Als Ladungswechsel bezeichnet man in der Motortechnik das Wechselspiel zwischen „Befüllen“ und „Entleeren“ des Zylinders. Zu Beginn des Arbeitsspiels muss der Zylinder mit dem Gemisch „befüllt“ werden. Hier spielt die Steuerzeit für das Einlassventil die entscheidende Rolle. Wird es später geöffnet, so entsteht ein größerer Druckunterschied zwischen Saugrohr und Zylinder und das Gemisch wird schneller angesogen. Wird es zu spät geöffnet, wird möglicherweise nicht genug Gemisch angesaugt.
Beim „Entleeren“ des Zylinders werden die Abgase durch das Auslassventil hinausgesaugt. Hier spielen die Geometrie der Abgasanlage sowie die Ventilsteuerzeiten eine große Rolle. [vgl. Robert Bosch GmbH, 2003]

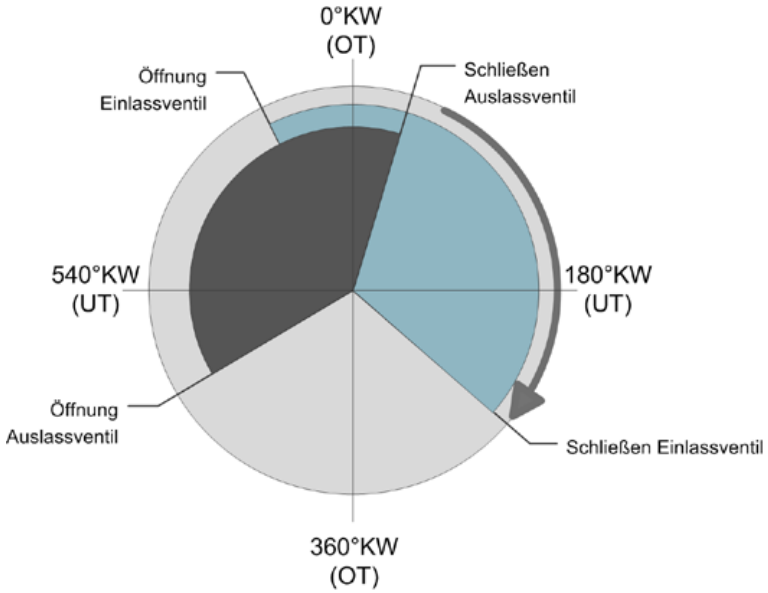

Abbildung 2.6.: Beispiel Steuerzeiten der Ventile

Im Detail wird dies hier nicht erläutert, da die Ventile im Hawks-Projekt nicht elektronisch gesteuert werden und daher kein Einfluss auf diese Größen genommen werden kann.

2.4. Gemischbildung und Einspritzung

Die Bildung eines zündfähigen Gemisches aus Luft und Kraftstoff ist eine der Hauptaufgaben des Motorenmanagements. Ziel ist es, der angesaugten Menge an Luft die optimale Menge an Kraftstoff beizumischen. Entscheidende Kriterien sind hierbei das Verhältnis von Luft zu Kraftstoff und eine möglichst gute Vermischung. Seitens der Motorsteuerung kann nur auf das Verbrennungsluftverhältnis Einfluss genommen werden. Diese Begriffe werden im folgenden Kapitel erläutert.

2.4.1. Verbrennungsluftverhältnis

Das Verhältnis von Luftmasse zur Kraftstoffmenge wird als λ oder Verbrennungsluftverhältnis bezeichnet.

$$\lambda = \frac{Luftmasse}{Kraftstoffmenge}$$

Wenn eine vollständige Verbrennung eintritt, ist $\lambda = 1$. Dieses Verhältnis ist allerdings nicht zwingend erstrebenswert. Ein Verbrennungsmotor entwickelt seine maximale Leistung bei einem leichten Luftmangel, also $\lambda < 1$. Hier ist der Wirkungsgrad am größten.
Wenn der Motor nicht unter Volllast betrieben wird, wird er mit einem mageren Gemisch betrieben, um Emissionen und Kraftstoffverbrauch gering zu halten. Diese Zusammenhänge werden in Abbildung 2.7 veranschaulicht.

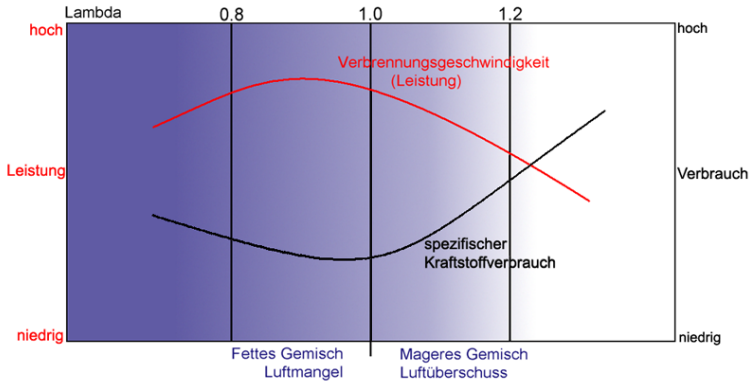

Abbildung 2.7.: Relation zwischen Lambda und Verbrennungsgeschwindigkeit

Die benötigte Kraftstoffmenge resultiert aus der angesaugten Luftmasse, die sich aus dem Luftdruck im Ansaugrohr und deren Temperatur ergibt, und dem Fahrerwunsch nach Leistung, gemessen durch die Veränderung und Stellung der Drosselklappe. Diese Abhängigkeiten werden in Abbildung 2.8 dargestellt.

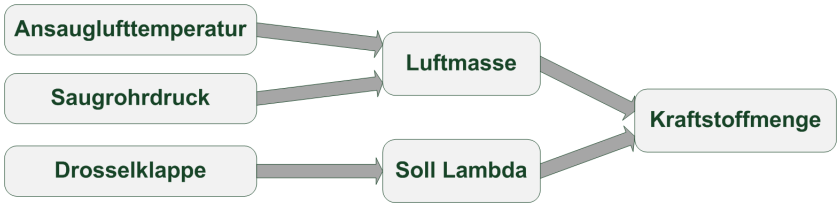

Abbildung 2.8.: Abhängigkeiten der Kraftstoffmenge

2.4.2. Gemischbildung

Das Erzeugen dieses Gemisches wurde früher durch einen mechanischen Vergaser übernommen, der den Kraftstoff durch den im Saugrohr herrschenden Unterdruck zerstäubte. Die Kraftstoffmenge wurde hierbei durch den Druckunterschied, also durch die Menge der angesaugten Luft, ermittelt. Allerdings ist diese Lösung unflexibel und ungenau, weshalb heutzutage fast ausschließlich elektronisch gesteuerte Einspritzanlagen verwendet werden.

Einspritzung

Zur Einspritzung werden elektromagnetische Einspritzventile genutzt. Diese werden durch Anlegen einer Spannung geöffnet und injizieren den Kraftstoff. Die Kraftstoffmenge lässt sich über die Öffnungszeit der Ventile exakt bemessen, wenn die Durchflussrate der Ventile sowie der Benzindruck bekannt sind.
Die Vermischung von Luft und Kraftstoff entsteht hierbei durch den im Einlasskanal herrschenden Unterdruck sowie durch die Geometrie der Einspritzentile, die den Kraftstoff zerstäuben.

Aufbau der Kraftstoffversorgung

Der Kraftstoff wird mittels einer Kraftstoffpumpe vom Tank mit einem definierten Druck zu den Einspritzventilen geführt. Da dieser Druck bei Öffnung der Ventile abfällt, wird dieser durch einen Druckregler reguliert um ein konstantes und berechenbares Druckniveau zu erhalten. Dieser Aufbau wird in Abbildung 2.9 dargestellt.

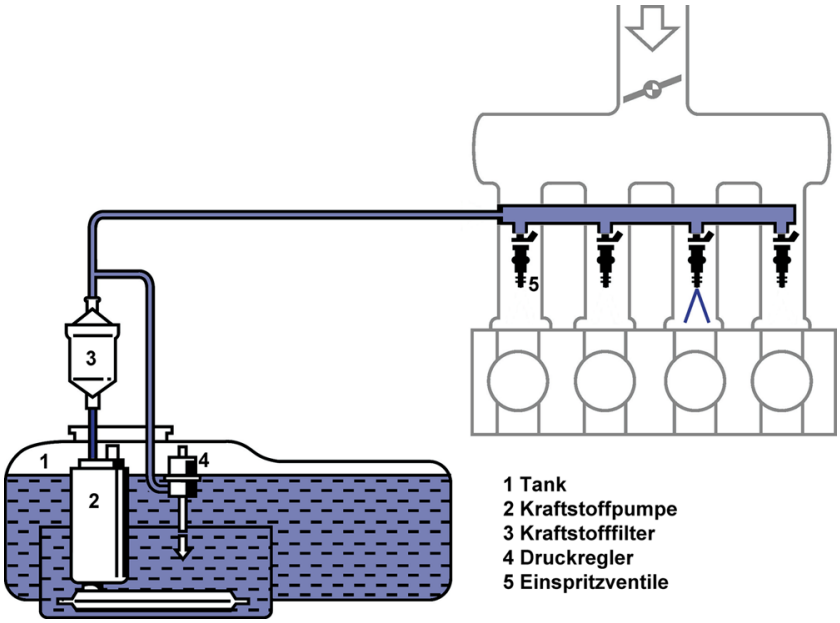

Abbildung 2.9.: Aufbau der Kraftstoffversorgung [basierend auf Robert Bosch GmbH, 2005]

2.4.3. Möglichkeiten der Einspritzung

Die Einspritzung kann auf unterschiedliche Arten erfolgen. Diese lassen sich anhand zweier Merkmale charakterisieren: Dem Zeitpunkt der Einspritzung in Bezug auf die Einlassventilstellung und die Organisation der Einspritzung bei Mehrzylindermotoren.

Zeitpunkt der Einspritzung

Es gibt zwei mögliche Zeitpunkte für die Einspritzung. Um eine gleichmäßige und kalkulierbare Gemischbildung zu gewährleisten dürfen diese beiden Möglichkeiten nicht vermischt werden.

Einspritzung bei geschlossenen Ventilen

Hierbei wird nur bei geschlossenen Einlassventilen eingespritzt. Dies wird auch als vorgelagerte Gemischbildung bezeichnet. Der Vorteil dieser Lösung besteht darin, dass das Zeitfenster zum Einspritzen größer ist, da die komplette Zeit mit geschlossenen Ventilen genutzt werden kann. Das Zeitfenster ist in der folgenden Abbildung 2.10 grün dargestellt.

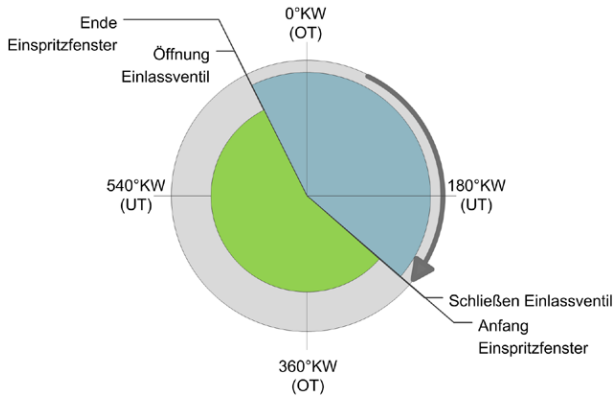

Abbildung 2.10.: Zeitfenster für die Einspritzung bei geschlossenem Einlassventil

Einspritzung in offene Ventile

Es wird nur in geöffnete Einlassventile eingespritzt. Dies hat den Vorteil, dass der Kraftstoff durch den starken Druckunterschied zwischen Zylinder und Ansaugrohr angesaugt wird und dadurch gut verteilt wird. Der Nachteil ist, dass die maximal einspritzbare Kraftstoffmenge direkt von der Öffnungszeit der Ventile abhängt. Bei hohen Drehzahlen ist ein hoher Durchsatz der Einspritzventile und dafür ein hoher Benzindruck nötig. Das Einspritzzeitfenster ist in Abbildung 2.11 grün markiert.

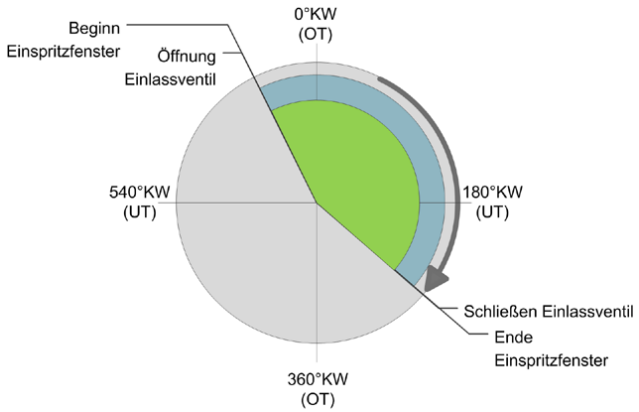

Abbildung 2.11.: Zeitfenster für die Einspritzung bei offenem Einlassventil

Organisation der Einspritzung bei Mehrzylinder-Motoren

Da sich bei einem Mehrzylindermotor die Zylinder in unterschiedlichen Takten befinden, müssen diese theoretisch auch zu unterschiedlichen Zeiten befüllt werden. Da dies sehr aufwändig ist, gibt es folgende Ansätze um den Aufwand zu senken:

Simultane Einspritzung

Hierbei werden alle Zylinder gleichzeitig befüllt, Es werden alle Einspritzventile parallel angesteuert. Da sich die Zylinder in unterschiedlichen Takten befinden hat dies hat den Nachteil, dass jeder Zylinder ein leicht abweichendes Gemisch erhält, da das Gemisch unterschiedlich lange vor dem Zylinder verweilt und damit das Verdampfen unterschiedlich weit fortgeschritten ist.
Des Weiteren ist eine Trennung zwischen der Einspritzung in offene und geschlossene Einlassventile (vgl. S.19) nicht möglich, da immer mindestens ein Einlassventil geöffnet und immer mindestens ein Einlassventil geschlossen ist.
Diese Art der Einspritzung ist sehr leicht umzusetzen, da hier kein Nockenwellensensor benötigt wird und nur ein Ausgang zum Ansteuern aller Einspritzventile benötigt wird. Allerdings ist diese Lösung sehr ungenau. [vgl. Robert Bosch GmbH, 2005]

Vollsequentieller Betrieb

Bei einer vollsequentiellen Einspritzung wird jedes Einspritzventil für jeden Zylinder individuell angesteuert. Damit kann gewährleistet werden, dass jeder Zylinder ein identisches Gemisch erhält. So kann eine höhere Laufruhe erreicht werden.
Ein Nachteil ist der höhere Aufwand, da hier zur eindeutigen Identifikation der Phase ein Nockenwellensensor erforderlich ist. (vgl. Kapitel 2.2 4-Takt-Zyklus, S. 14)

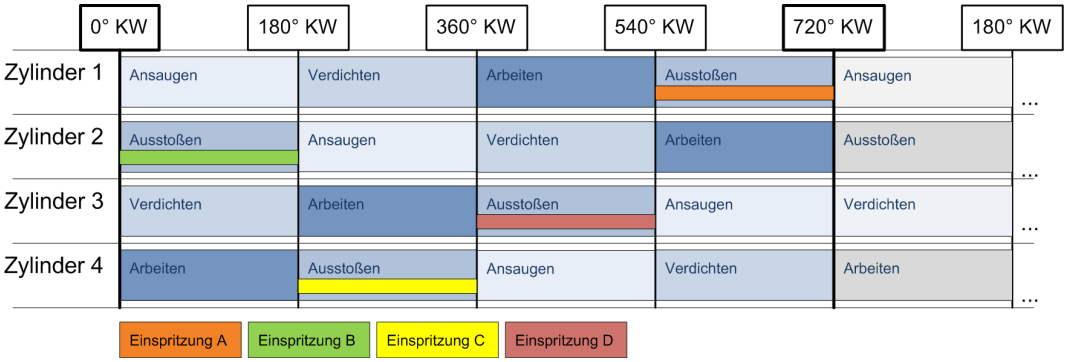

Abbildung 2.12.: Vollsequentielle Einspritzung bei geschlossenen Ventilen

Halbsequentieller Betrieb

Als Kompromiss zwischen der vollsequentiellen und der simultanen Einspritzung wurde die halbsequentielle Einspritzung entwickelt. Es werden immer paarweise Zylinder befüllt. (vgl. Abbildung 2.13). Die Befüllung eines Zylinders wird auf zwei Einspritz-Pulse verteilt. Der erste Puls verbleibt eine Umdrehung im Einlasskanal bevor der zweite Puls direkt vor Öffnung des Einlassventils erfolgt.
Gegenüber der vollsequentiellen Einspritzung bietet sie den Vorteil, dass hier kein Nockenwellensensor benötigt wird, da eine Identifikation von einer Umdrehung, 360°KW, ausreicht.

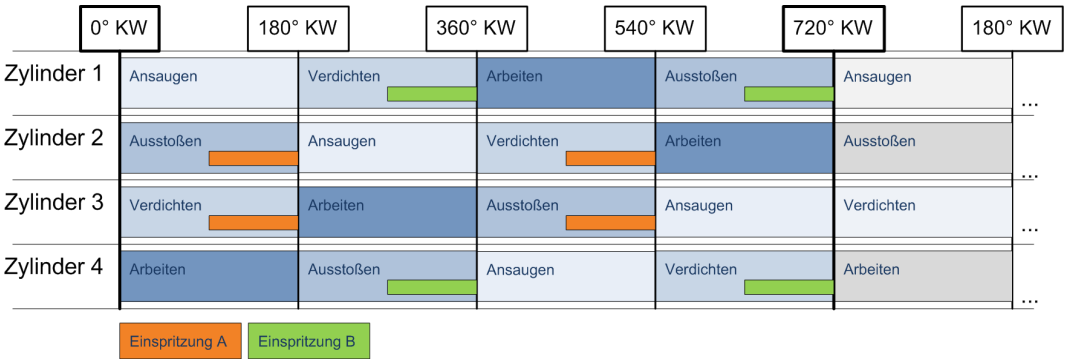

Abbildung 2.13.: Halbsequentielle Einspritzung bei geschlossenen Ventilen

2.5. Zündung

Das Gemisch wird bei einem Ottomotor fremdgezündet, also durch einen externen Impuls und nicht durch Selbstentzündung wie beispielsweise bei einem Dieselmotor.
Die Zündung erfolgt über den Zündfunken einer elektrischen Zündkerze. Die dafür nötige Hochspannung wird in einer Zündspule aufgebaut. Diese hat zwei Wicklungen, eine Primär- und eine Sekundärwicklung. Die Sekundärwicklung ist direkt an die Zündkerze angeschlossen und hat eine deutlich größere Wicklungszahl im Vergleich zur Primärwicklung.
Legt man an die Primärwicklung eine realtiv niedrige Spannung an, baut sich ein Magnetfeld auf. Fällt die Spannung ab, bricht das Magnetfeld zusammen und erzeugt auf der Sekundärwicklung durch Induktion eine hohe Spannung. Diese reicht

aus um den Zündfunken zu erzeugt. Die Schaltung einer Zündspule ist in Abbildung 2.14 auf Seite 23 dargestellt. [vgl. Robert Bosch GmbH, 2003, S.621]

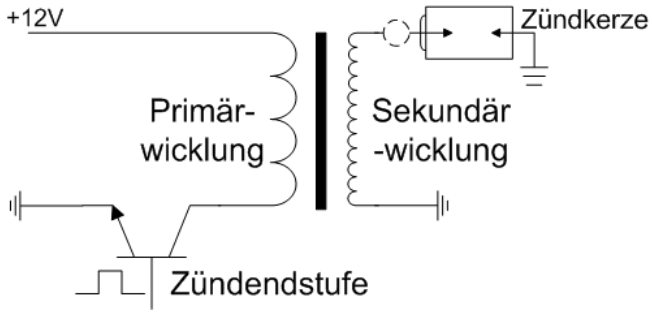

Abbildung 2.14.: Schaltplan einer Zündspule

Das Aufladen der Zündspule benötigt Zeit, die als „Schließzeit“ (engl. „dwelltime“) bezeichnet wird. Sie hat direkten Einfluss auf die Zündenergie und ist von der Batteriespannung abhängig. Eine zu niedrige Zündenergie führt zu Fehlzündungen, da die nötige Spannung nicht erreicht wird. Eine zu hohe Zündenergie kann die Zündspule überhitzen und somit zerstören.

2.5.1. Zündwinkel

Der Zeitpunkt der Zündung hat entscheidenen Einfluss auf die Leistung des Motors. Er wird als Zündwinkel bezeichnet und bezieht sich auf den oberen Totpunkt des jeweiligen Zylinders. (vgl. Abbildung 2.15)
Die Verbrennung des Gemischs geschieht mit endlicher Geschwindigkeit und braucht somit Zeit. Der durch die Verbrennung entstehende Druck steigt mit dem Fortschritt der Verbrennung. Folglich sollte der maximale Druck kurz nach Überschreiten des oberen Totpunkts (OT) erreicht werden, um ihn möglichst effektiv nutzen zu können.

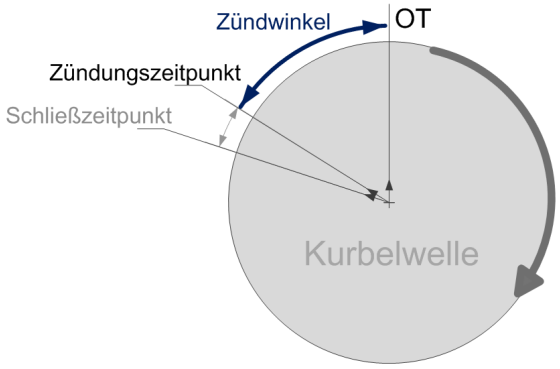

Abbildung 2.15.: Zündwinkel

Der Zündwinkel ist von der Motordrehzahl und der Verbrennungsgeschwindigkeit des Gemischs abhängig. Dreht der Motor schneller, ist die Zeit bis zum Erreichen des OT kürzer. Da allerdings die Verbrennungsgeschwindigkeit davon unabhängig ist, muss früher gezündet werden.
Die Verbrennungsgeschwindigkeit hängt vom λ-Wert (vgl. Verbrennungsluftverhältnis, S.17) und der Befüllung des Zylinders ab. Die Verbrennungsgeschwindigkeit erreicht bei leichtem Luftmangel ihr Maximum und steigt mit zunehmender Befüllung des Zylinders. Ist die Verbrennungsgeschwindigkeit geringer, muss der Zündwinkel verringert werden.
Die Einflussgrößen auf den Zündwinkel sind in Abbildung 2.16 dargestellt.

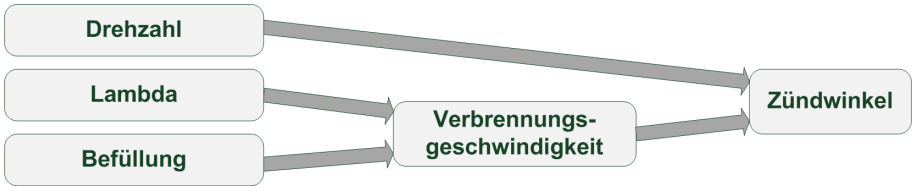

Abbildung 2.16.: Abhängigkeiten des Zündwinkels

Ist der Zündzeitpunkt zu früh, kommt es zu sogenanntem Klopfen. Hierbei kommt es durch den Druck- und Temperaturanstieg im Brennraum zur Selbstentzündung des noch nicht von der eigentlichen Verbrennung erfassten Gemisches. Somit verläuft die Verbrennung schlagartig und unkontrolliert. Dies setzt den Motor extremen Belastungen aus und kann diesen stark beschädigen.
Der Zeitpunkt, ab dem dieser Effekt auftritt, wird als „Klopfgrenzen“ bezeichnet. Sie hängt von der Temperatur des Motors, sowie vom verwendetem Kraftstoff ab.

Liegt der Zündzeitpunkt zu spät, verliert der Motor an Leistung und die Abgastemperatur steigt stark an. Letzteres kann auf Dauer die Abgassanlage überhitzen und zerstören. [vgl. Robert Bosch GmbH, 2003, S.623ff]

2.5.2. Zündverteilung

Bei Mehr-Zylinder-Motoren befinden sich die einzelnen Zylinder in unterschiedlichen Takten des Arbeitsspiels. (vgl. 2.2 4-Takt-Zyklus, S.14). Die Reihenfolge, in der die Zylinder arbeiten, wird durch die „Zündreihenfolge“ beschrieben. Sie ist durch die Bauart des jeweiligen Motors festgelegt. [vgl. Robert Bosch GmbH, 2003, S.455]

Bei einem 4-Zylinder-Motor muss jediglich ein Zylinder pro Takt gezündet werden. Die Aufgabe, die Zündung entsprechend der Zündreihenfolge zu verteilen, übernahm früher ein mechanischer Zündverteiler, der durch rotierende Kontakte die Spannung einer Zündspule auf den jeweiligen Zylinder leitete. Hier spricht man von „rotierender Verteilung“ (ROV).

Heute wird dies elektronisch geregelt, was als „ruhende Verteilung“ (RUV) bezeichnet wird. Vorteile sind hierbei der Wegfall von mechanischen Verschleißteilen sowie die höhere Genauigkeit, die durch eine elektronische Regelung erreicht werden kann. Nachteil ist, dass bei der RUV pro Zylinder eine eigene Zünspule benötigt wird, bei einem mechanischen Verteiler hingegen nur eine. [vgl. Robert Bosch GmbH, 2003, S.652]
Für die elektronische Zündverteilung gibt es zwei Verfahren:

Vollsequentielle Zündung

Es wird pro Takt nur der Zylinder gezündet, der sich auch tatsächlich im Arbeitstakt befindet. Dieses Verfahren ist das Pendant zur vollsequentiellen Einspritzung (vgl. 2.4.3 Organisation der Einspritzung bei Mehrzylinder-Motoren), da hier auch das volle Arbeitsspiel eines Zylinders mittels eines Nockenwellensensors identifiziert werden muss.
Dieses Verfahren wird fälschlicherweise auch als „Coil-on-Plug“ (COP) bezeichnet. Dieser Begriff bezieht sich auf die Bauform der Zündspulen, die üblicherweise direkt auf der Zündkerze stecken. Dies ist jedoch auch beim Wasted Spark-Verfahren der Fall. In Abbildung 2.17 ist das Arbeitsspiel mit einer vollsequentiellen Zündung abgebildert. Die Zündungs-Ereignisse sind als gelbe Blitze dargestellt.

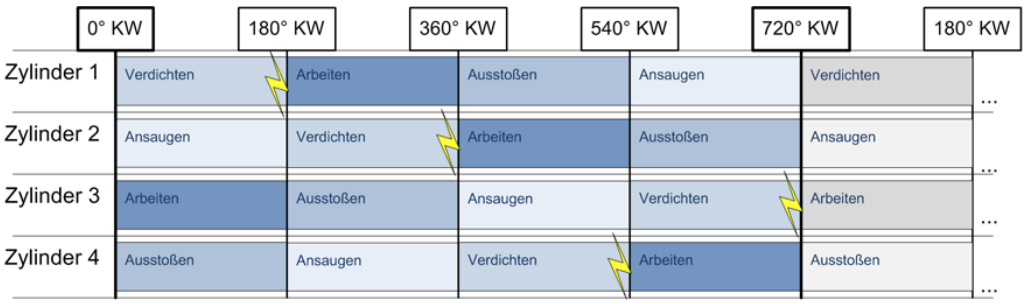

Abbildung 2.17.: Arbeitsspiel eines 4-Zylinder Motors mit vollsequentieller Zündung und Zündreihenfolge 1-2-4-3

Wasted Spark

Eine Möglichkeit, die Zündung einfacher zu gestalten, besteht im sogenannten „Wasted Spark“-Verfahren. Hierbei werden jeweils 2 Zylinder zeitgleich gezündet. Einmal der reguläre Zylinder im Arbeitstakt und zusätzlich der Zylinder im Ausstoßtakt. In diesem hat der Zündfunke keine Auswirkungen, da sich zu dem Zeitpunkt kein zündfähiges Gemisch mehr im Zylinder befindet. So kann auf einen Nockenwellensonser verzichtet werden. Nachteil ist allerdings, dass die Zündspulen stärker belastet werden, da sie nun jede Umdrehung geladen weren müssen, also doppelt so oft, wie im vollsequentiellen Betrieb. [vgl. Robert Bosch GmbH, 2003, S.627]
Abbildung 2.18 zeigt den Ablauf des Arbeitsspiels. Die relevanten Zündungen sind gelb, die zusätzlichen Zündfunken im Ausstoßtakt in orange dargestellt.

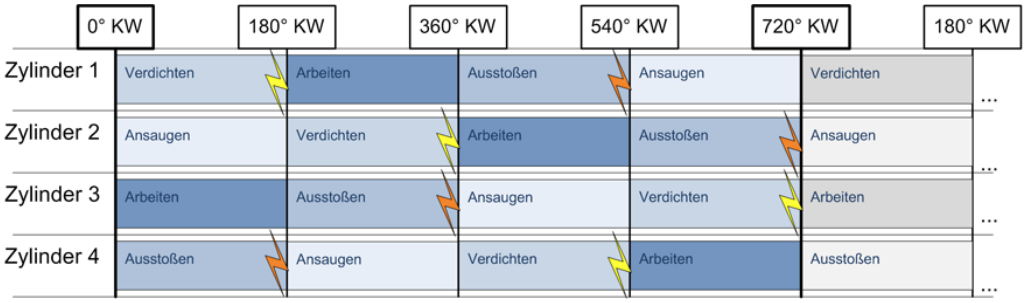

Abbildung 2.18.: Arbeitsspiel bei einem 4-Zylinder-Motor mit Wasted Spark

2.6. Motorsteuerung

Die Motorsteuerung hat die Aufgabe, die Betriebsparameter des Motors dem Fahrerwunsch und der aktuellen Fahrsituation anzupassen und dabei flexibler und genauer zu sein, als bisherige, mechanische Lösungen.
Üblicherweise basiert sie auf einem Mikrocontroller und vereint die Steuerung von Einspritzung und Zündung in einem Gerät.
Sie wird auch als „Engine Control Unit“ (ECU) bezeichnet.

Zudem bietet eine elektronische Motorsteuerung Möglichkeiten, flexibel auf unterschiedliche Fahrsituationen zu reagieren.
Typische Funktionen sind beispielsweise:

Aufwärm-Anreicherung:

Während der Aufwärmphase benötigt der Motor mehr Kraftstoff, da dieser teilweise an den noch kalten Zylinderwänden kondensiert und so nicht am Verbrennungsprozess teilnimmt. Durch eine Anpassung der Kraftstoffmenge in Abhängigkeit von der Motortemperatur lässt sich dies ausgleichen und den Motor schneller auf Betriebstemperatur bringen.

Beschleunigungs-Anreicherung:

Wenn der Fahrer beschleunigen will und sich die Stellung des Gaspedals stark verändert, kann die ECU für diesen kurzen Moment das Gemisch anfetten um so das Ansprechen des Motors zu beschleunigen. (vgl. Kapitel 2.4.1 Verbrennungsluftverhältnis, S.17).

Schubabschaltung:

Wenn der Motor im Schleppbetrieb arbeitet, also die Drosselklappe komplett geschlossen ist und die Drehzahl über einem einstellbaren Schwellenwert liegt, wird die Einspritzung unterbrochen, um Kraftstoff zu sparen. In Kombination mit Aufwärm-Anreicherung kann diese Funktion temperaturabhängig auch abgeschaltet werden. Dies macht Sinn, falls der Motor zu kalt ist.[vgl. Robert Bosch GmbH, 2005]

Drehzahlbegrenzer:

Die Drehzahl wird abhängig von der Motortemperatur begrenzt. Ist der Motor zu kalt, wird die Drehzahl begrenzt, um zu hohen Verschleiß zu vermeiden. Ist er zu heiß, wird ebenfalls reduziert, um eine Überhitzung des Motors zu verhindern.

2.6.1. Typische Peripherie

Um diese Aufgaben zu erfüllen benötigt die ECU eine Vielzahl von Sensoren und Aktoren. Die typische Peripherie einer modernen ECU ist in Abbildung 2.19 dargestellt.

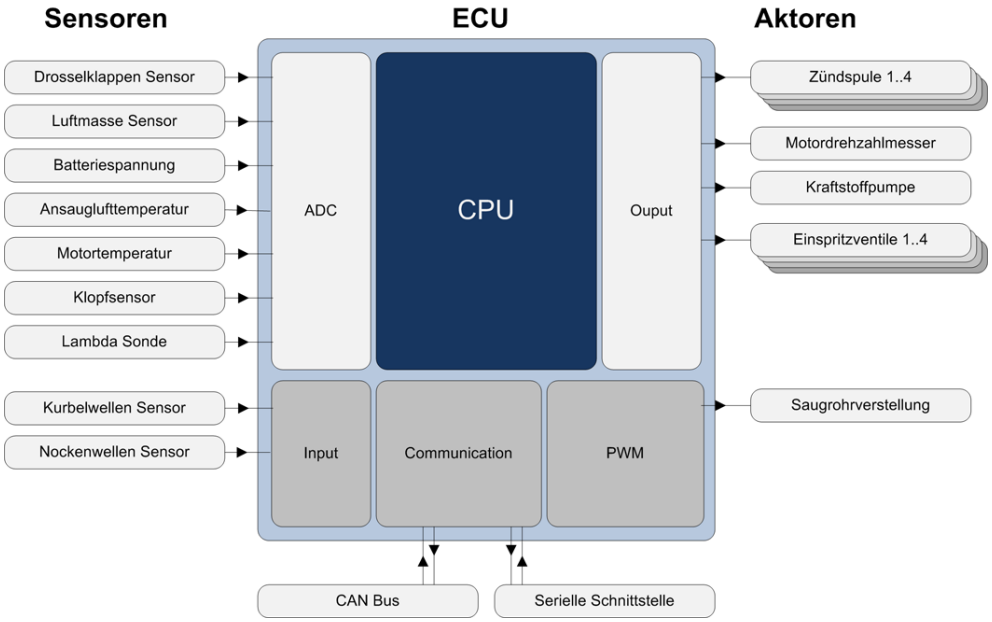

Abbildung 2.19.: Blockbild der Struktur einer typischen ECU

Die Aktoren wie Zündspulen und Einspritzventile wurden in Kapitel 2.5 beziehungsweise 2.4.2 bereits erläutert. Hier wird nun die Sensorik im einzelnen erläutert:

Sensorik

Kurbelwellensensor:

Ein auf der Kurbelwelle angebrachtes Zahnrad wird mit einem induktiven Sensor oder einem Hall-Sensor abgetastet. So entsteht ein Puls bei jedem „Zahn“. Somit kann die aktuelle Drehzahl gemessen werden. Um nicht nur die Drehzahl, sondern auch die Stellung der Kurbelwelle auslesen zu können, fehlt typischerweise mindestens ein Zahn.
Das Zahnrad wird auch als „Flywheel“ bezeichnet.

Nockenwellensensor:

Ähnlich dem Kurbelwellensensor befindet sich auf einer Nockenwelle ein induktiver Sensor bzw ein Hall-Sensor, der als Phasen-Geber für den vollsequentiellen Betrieb dient. (vgl. Kapitel 2.2 4-Takt-Zyklus, S. 14).

Kühlwassertemperatur:

Aus der Kühlwassertemperatur lässt sich die Temperatur des Motors ermitteln. Diese Information wird zum berechnen der Kraftstoffmenge. Zudem kann so eine mögliche Überhitzung rechtzeitig erkannt und verhindert werden.

Luftmassesensor:

Über den Druck im Saugrohr lässt sich die aktuell angesaugte Luftmasse ermitteln. Dies ist wichtig für die Ermittlung der optimalen Kraftstoffmenge.

Ansauglufttemperatur:

Die Temperatur der angesaugten Luft hat ebenfalls Einfluss auf die Luftmasse und wird daher zur korrekten Berechnung ebenfalls benötigt.

Drosselklappensensor:

Anhand der Drosselklappenstellung kann die Stellung des Gaspedals und damit der Wunsch des Fahrers ermittelt werden.

Lambdasonde:

Dieser Sensor misst den Sauerstoffanteil in den Abgasen. Aus dieser Information können Rückschlüsse auf den λ-Wert gezogen werden und die Kraftstoffmenge dementsprechend nachgeregelt werden. (vgl. Kapitel 2.4.1 Verbrennungsluftverhältnis, S.17)

2.6.2. Kennfelder

Die Zusammenhänge zwischen den Sensorwerten und den Ausgangsparametern sind sehr komplex und daher nur schwer als mathematische Funktionen implementierbar. Da diese Zusammenhänge zudem Bestandteil der Konfiguration einer Motorsteuerung sind, sollen diese angepasst werden können. Hierfür müssten die Funktionen modifiziert werden, was einen ungleich höheren Aufwand bedeuten würde.

Stattdessen hat sich das Verwenden von sogenannten „Kennfeldern“ durchgesetzt. Kennfelder sind Tabellen, die mehrere Eingangsgrößen in Bezug zu einer Ausgangsgröße setzen. Die einzelnen Spalten und Zeilen sind sogenannten "Stützstelle“ zugeordnet. Diese bilden eine kleine Teilmenge des Wertebereichs der Eingangsgrößen. Die Tabellenzellen sind mit den Ergebniswerten gefüllt, die für die jeweiligen Stützstellen gelten sollen:

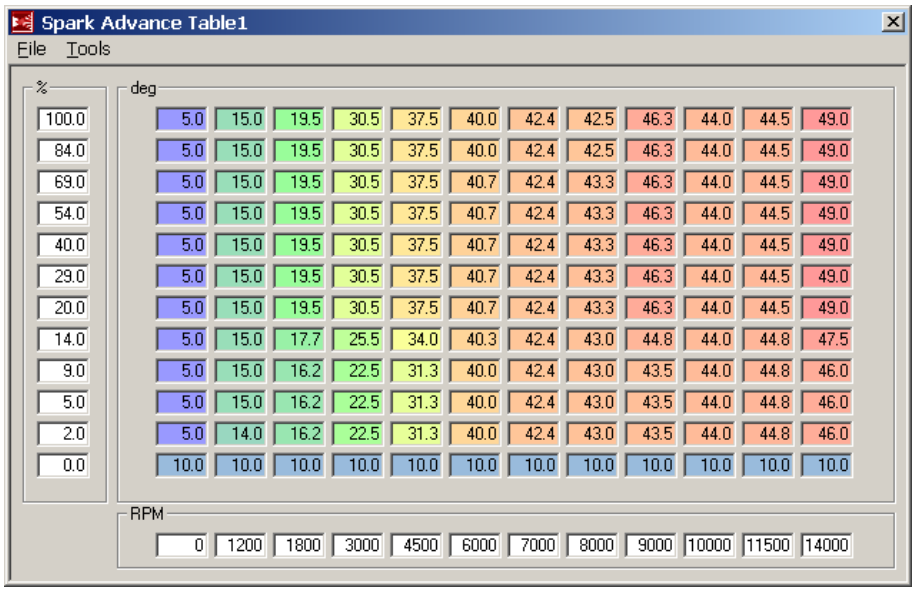

Abbildung 2.20.: 12x12 Kennfeld des Zündwinkels in Abhängigkeit von Drehzahl und Drosselklappenstellung

Wird nun eine Größe in Abhängigkeit von zwei gegebenen Werten gesucht, werden für jeden Eingangswert die benachbarten Stützstellen ermittelt und die Funktion der Grade zwischen diesen beiden Punkten ermittelt. Mit Hilfe der ermittelten Funktion wird die Ausgangsgröße berechnet.

Die Vorteile dieser Methode sind die Flexibilität in der Konfiguration und die einfache Implementierbarkeit. Nachteil ist der hohe Speicherbedarf für die Tabellen.

Neben der Darstellungsform in Tabellenform (vgl. Abbildung 2.20) ist es auch üblich, Kennfelder dreidimensional darzustellen.

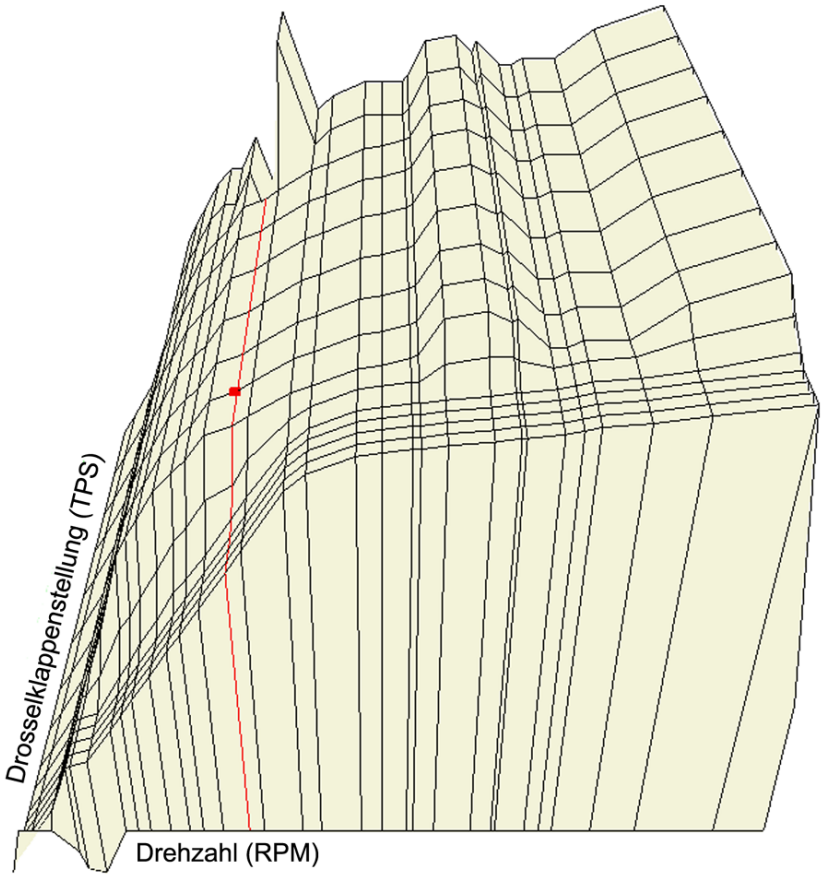

Abbildung 2.21.: Dreidimensionale Darstellung eines Kennfeldes

3. Anforderungen

In diesem Kapitel werden die verschiedenen Anforderungen erläutert, denen die im Rahmen dieser Arbeit entwickelte ECU genügen muss. Es wird jeweils die Grundlage der jeweiligen Anforderung und die dazugehörige Problematik beschrieben sowie ein Abnahmekriterium definiert.

3.1. Anforderungen seitens Hawks-Racing

In einem Rennfahrzeug gelten spezielle Anforderungen. Diese werden seitens des Hawks-Teams und durch die Formula Student gestellt und hier beschrieben:

3.1.1. Kompatibilität

Beschreibung

Die entwickelte Motorsteuerung soll die bisher verwendete ECU direkt ersetzen können. Es soll direkt zwischen den Steuergeräten gewechselt werden können, ohne weitere Umbauten oder Konfigurationen vornehmen zu müssen.
Hierfür müssen sowohl die Hardware- als auch die Software-Schnittstellen entsprechend angepasst werden.

Problembeschreibung

Für die Entwicklungsphase der ECU sind auch Tests im Fahrzeug vorgesehen. Um den Aufwand gering zu halten und die Leistungsfähigkeit des Fahrzeugs nicht einzuschränken, soll ein direkter Austausch möglich sein.

Abnahmekriterium

Die Anforderung gilt als erfüllt, wenn die Steuergeräte direkt getauscht werden können ohne weitere Einstellungen am Fahrzeug vornehmen zu müssen.

3.1.2. Baugröße und Gewicht

Beschreibung

Die entwickelte ECU soll mit minimalem Bauraum auskommen und möglichst leicht gebaut sein.

Problembeschreibung

Bei der Entwicklung eines Rennfahrzeugs sind Bauraum und Gewicht kritische Faktoren. Das Gewicht des Gesamtfahrzeugs sollte so niedrig wie möglich gehalten werden, um die Fahreigenschaften optimal zu gestalten. Das zu entwickelnde Steuergerät soll seinen Beitrag dazu leisten und ein möglichst geringes Gewicht aufweisen.

Abnahmekriterium

Die Anforderung gilt als erfüllt, wenn sich die neue ECU problemlos im Fahrzeug unterbringen lässt und das Gewicht nicht das des bisherigen Steuergerätes überschreitet.

3.1.3. Konfigurations-Software

Beschreibung

Es soll eine einfach zu bedienende Software geben, über die die vollständige Konfiguration der ECU erledigt werden kann.

Problembeschreibung

Die Konfiguration einer ECU ist ein komplexer Vorgang und erfordert einen hohen Grad an Kenntnissen der Fahrzeugtechnik. Daher wird die Konfiguration von Kraftfahrzeugtechnikern übernommen werden. Damit diese sich auf ihre Aufgabe konzentrieren können, soll die Software übersichtlich, zuverlässig und einfach zu bedienen sein.

Abnahmekriterium

Die Anforderung gilt als erfüllt, wenn sich die neue ECU mittels der Software zuverlässig konfigurieren lässt.

3.1.4. Reglement der Formula Student

Beschreibung

Die entwickelte ECU muss dem Reglement der Formula Student entsprechen.

Problembeschreibung

Das Reglement schränkt einige technische Möglichkeiten ein. Damit das Fahrzeug für die Wettbewerbe der Formula Student zugelassen wird, muss es diesen Einschränkungen entsprechen.

Abnahmekriterium

Die Anforderung gilt als erfüllt, wenn das Reglement der Formula Student eingehalten wird.

3.2. Anforderungen des Motors

Neben den Anforderungen durch das Hawks-Projekt gibt es eine Reihe von Anforderungen seitens des zusteuernden Motors. Diese wurden in Absprache mit der Motorbaugruppe erarbeitet.

3.2.1. Vollsequentielle Einspritzung

Beschreibung

Der verwendete Motor benötigt eine vollsequentielle Einspritzung.

Problembeschreibung

Der Kawasaki Motor ist ein sehr hochdrehendes Modell, wodurch die Zeitfenster für die Einspritzung sehr kurz werden können. (vgl. 2.4.3 Organisation der Einspritzung bei Mehrzylinder-Motoren, S.21). Frühere Versuche, den Motor mit einer halbsequentiellen Einspritzung zu betreiben, führten zu sogenannten Rückzündungen, sodass der Motor nur vollsequentiell betrieben werden sollte.[vgl. Brandt, 2008]

Abnahmekriterium

Die Anforderung gilt als erfüllt, wenn die neue ECU die Einspritzung vollsequentiell betreibt.

3.2.2. Uneingeschränkte Leistung/Drehzahl

Beschreibung

Die Motorsteuerung darf die Drehzahl und damit die Leistung des Motors nicht einschränken und muss bis zur theoretischen Maximaldrehzahl sauber arbeiten.

Problembeschreibung

Die Auslastung der Motorsteuerung hängt direkt von der jeweiligen Drehzahl ab. Je höher der Motor dreht, desto weniger Zeit verbleibt der ECU um die nötigen Routinen abzuarbeiten. Ist sie nicht mehr in der Lage, die Aktoren korrekt zu steuern, gefährdet dies den Motor und schränkt so dessen Lebensdauer und Leistung ein.

Abnahmekriterium

Die Anforderung gilt als erfüllt, wenn die neue ECU die Leistung des Motors nicht einschränkt und auch bei maximaler Drehzahl korrekt arbeitet.

3.2.3. Genauigkeit

Beschreibung

Die Motorsteuerung muss Zündung und Einspritzung hinreichend genau ansteuern können.

Problembeschreibung

Die genau Konfiguration der ECU ist nur dann möglich und sinnvoll, wenn die ECU die konfigurierten Werte auch genau umsetzt. Hierfür wurden nach Rücksprache mit der Motorbaugruppe Toleranzen festgesetzt.

Abnahmekriterium

Die Anforderung gilt als erfüllt, wenn die entwickelte ECU die Einspritzung mit einer zeitlichen Auflösung von $1\mu s$ und den Zündwinkel mit einer Genauigkeit von $0.1°\mathrm{KW}$ regelt.

3.2.4. Schadensfreiheit

Beschreibung

Die Motorsteuerung darf in keinster Weise die Funktionalität des Motors oder des Gesamtfahrzeuges einschränken oder gefährden.

Problembeschreibung

Ein Verbrennungsmotor kann durch das Setzen falscher Parameter seitens der ECU beschädigt werden. Wird beispielsweise im falschen Moment gezündet, wird der Motor starken Belastungen ausgesetzt, die ihn auf Dauer beschädigen würden.

Abnahmekriterium

Die Paramter für Zündung und Einspritzung entsprechen in jeder Situation den in der Konfiguration vorgegebenen Werten.

4. Analyse der Auto-Infrastruktur

Ein Überblick über die Infrastruktur im Auto ist für die Anpassung und Integration einer ECU essentiell. Hier wird neben der kraftfahrzeugtechnischen Ebene auch das bestehende Telemetrie-System erläutert.

4.1. Motor

Im Hawks Projekt wird aktuell ein Kawasaki ZX600J Motor eingesetzt. Dieser ist im H04 verbaut worden und wird auch im H05 zum Einsatz kommen.

Der Motor stammt von einem Rennmotorrad und erreicht eine maximale Drehzahl von bis zu $16000\frac{\mathrm{U}}{\mathrm{min}}$. Durch den in der Formula Student vorgeschriebenen Restriktor mit einem Durchmesser von nur $20\mathrm{mm}$ wird allerdings die maximal mögliche Luftmenge begrenzt, sodass der Motor im Hawks-Projekt nur noch $13000\frac{\mathrm{U}}{\mathrm{min}}$ erreicht. [vgl. Formula Student Germany, 2008]

Abbildung 4.1.: Montierter Motor im H04

Einen Überblick der technischen Daten des ZX600J Motors bietet die folgende Tabelle:

Funktionsweise	Ottomotor
Bauart	4 Zylinder in Reihe
Hubraum	$599\mathrm{cm}^3$
Ventilanzahl	4 Ventile
Nockenwellen	Doppelte obenliegende Nockenwellen
Verdichtungsverhältnis	12.8 : 1
Zündreihenfolge	1-2-4-3
Steuerzeiten Einlassventile	EÖ: $56°\mathrm{KW}$ vor OT ES: $80°\mathrm{KW}$ nach UT Maximaler Ventilhub: 9.2mm
Steuerzeiten Auslassventile	AÖ: $61°\mathrm{KW}$ vor UT, AS: $33°\mathrm{KW}$ nach OT Maximaler Ventilhub: 8.4mm
Kühlanlage	Flüßigkeitsgekühlt, mechanisch angetrieben Kühlmittelpumpe
Generator	Integrierte 3-Phasen Wechselstrom Lichtmaschine, Nennleistung 308W
Startanlage	Elektromotor und Freilaufkupplung

Tabelle 4.1.: Daten des Kawasaki ZX600J Motors [Quelle: Brandt, 2008]

4.1.1. Sensoren

Die verwendete Motor-Sensorik wird anhang der verbauten Sensoren im Hawks-H04 vorgestellt. Die Datenblätter der aufgeführten Typen befinden sich im elektronischen Anhang (vgl. Anhang C Inhalt der CD)

Kurbelwelleninkrementalgeber

Als Geberrad ist ein sogenanntes „24-2“-Rad verbaut, ein Zahnrad mit 24 Zähnen, wovon 2 fehlen. Die fehlenden Zähne sind benachbart. Diese Konfiguration ist von der verwendeten Walbro-ECU vorgegeben.
Die Frage, warum die Lücke 2 Zähne groß sein muss, bleibt unklar. Theoretisch reicht ein fehlender Zahn um den Winkel der Kurbelwelle zu erkennen (vgl. 2.6.1 Sensorik, S.28).

Abbildung 4.2.: Geberrad und induktiver Sensor am Motor montiert (Quelle:Brandt [2008])

Es wird der induktive Sensor verwendet, der serienmäßig am Kawasaki Motor verbaut ist.

Nockenwellensensor

Für einen vollsequentiellen Betrieb muss die Umdrehung der Nockenwelle erfasst werden, um die aktuelle „Phase“ zu identifizieren (vgl Grundlagen). Dieser Sensor ist beim verwendeten Motor nicht vorgesehen und wurde nachträglich ergänzt. Es wird ein Hallgeber vom Typ ATS616LSGTNT verwendet. [vgl. Brandt, 2008]

Drosselklappenstellung

Die Stellung der Drosselklappe wird mittels eines Potentiometers ermittelt. Der Spannungsabfall einer definierten Referenzspannung wird gemessen und so der Winkel der Drosselklappe errechnet. Im Reglement der Formula Student ist eine feste, mechanische Verbindung zwischen Gaspedal und Drosselklappe vorgeschrieben. Daher kann der Drosselklappenwinkel direkt als der Wunsch des Fahrers nach Leistung interpretiert werden.

Temperatursensoren

Als Temperatursensoren werden Heißleiterelemente verwendet. Hier sind Sensoren für die Ansaugluft und die Kühlmitteltemperatur verbaut.

Lambdasonde

Es ist eine Breitbandsonde vom Typ NGKLZA03 – E1 verwendet. Diese wird mittels des dazugehörigen Steuergeräts NGKTC6300D kontrolliert und gibt den gemessenen Wert als analogen Spannungspegel aus. [vgl. Brandt, 2008]

4.1.2. Aktoren

Verwendete Einspritzventile

Es werden Einspritzventile vom Typ Bosch EV-14 verwendet. Diese sind hochohmig und können daher direkt mit $12\mathrm{V}$ geschaltet werden. Sie werden direkt von den integrierten Einspritzungstreibern der Walbro ECU angesteuert. Das dazugehörige Datenblatt befindet sich im elektronischen Anhang (vgl Anhang C Inhalt der CD, S.125).

Verwendete Zündanlage

Es werden vier separate Zündspulen verwendet, mit denen der Motor serienmäßig bestückt wird. Diese werden direkt durch die in der Walbro integrierten Zündendstufe angesteuert.

4.1.3. Konfiguration der Walbro ECU

Die Walbro ist für einen vollsequentiellen Betrieb konfiguriert. Dies gilt für die Einspritzung wie auch für die Zündung (vgl. 2.4.3 Organisation der Einspritzung bei Mehrzylinder-Motoren, S.21). Die Einspritzung wird bei geschlossenen Ventilen durchgeführt, um bei den hohen Drehzahlen des Kawasaki-Motors ein größeres Zeitfenster zur Verfügung zu haben (vgl 2.4.3 Zeitpunkt der Einspritzung, S.20).

4.2. Telemetrie-System

Das Telemetrie-System des Hawks-Teams umfasst derzeit rund 20 Mikrocontroller. Diese dienen primär der Auswertung und Aufbereitung von Sensorikdaten wie beispielweise Lenkwinkel oder Raddrehzahl. Die Kommunikation zwischen den Controllern läuft über einen Time-Triggered CAN-Bus, kurz TTCAN.

4.2.1. TTCAN

TTCAN ist ein zeitgesteuerter Bus, der auf dem CAN-Bus basiert. Zeitgesteuert bedeutet, dass jeder Busteilnehmer definierte Zeitfenster (engl. „Timeslot“) hat, in denen er auf den Bus senden darf. Der Ablauf dieser Zeitfenster ist dabei zyklisch und wiederholt sich regelmäßig. Außerhalb dieser Zeitfenster darf der Teilnehmer nicht senden [vgl. Kolbe, 2008, S.22].

TimeMaster

Damit jeder Busteilnehmer eindeutig weiß, wann er senden darf, muss eine globale Zeit definiert und laufend synchronisiert werden. Die Aufgabe, diese Zeit bereitzustellen und zu übermitteln übernimmt der sogenannte „TimeMaster“. Dieser sendet in regelmäßigen Abständen eine Referenznachricht. Die Busteilnehmer vergleichen die Zeitdifferenz zwischen zwei empfangenen Referenznachrichten mit einem bekannten Sollwert. Weichen diese von einander ab, so muss die Zeit des entsprechenden Clients entsprechend nachgestellt werden.

Matrixzyklus

Bei TTCAN wird der zyklische Ablauf der Zeitfenster als „Matrixzyklus“ bezeichnet. Durch die Länge des Matrixzyklus wird die Frequenz festgelegt, mit der eine Nachricht gesendet werden kann. Um die mögliche Anzahl Zeitfenster zu erhöhen, ohne ein Risiko bei der Synchronisation einzugehen kann der Matrixzyklus in mehrere „Basiszyklen“ gegliedert werden. Dabei hat jeder Basiszyklus die gleiche Anzahl Zeitfenster und eine TimeMaster-Nachricht. Auf diese Weise sind dann unterschiedliche Frequenzen möglich.

Im Hawks-Projekt wird ein Matrixzyklus mit einer Länge von $40\mathrm{ms}$ verwendet. Dieser ist in 4 Basiszyklen unterteilt. Somit ergeben sich als mögliche Frequenzen für Nachrichten $100\mathrm{Hz}$, $50\mathrm{Hz}$ und $25\mathrm{Hz}$, vorrausgesetzt, eine Nachricht wird maximal einmal pro Basiszyklus übertragen. Die Struktur des Zyklus ist in Abbildung 4.3 auf der folgenden Seite dargestellt.

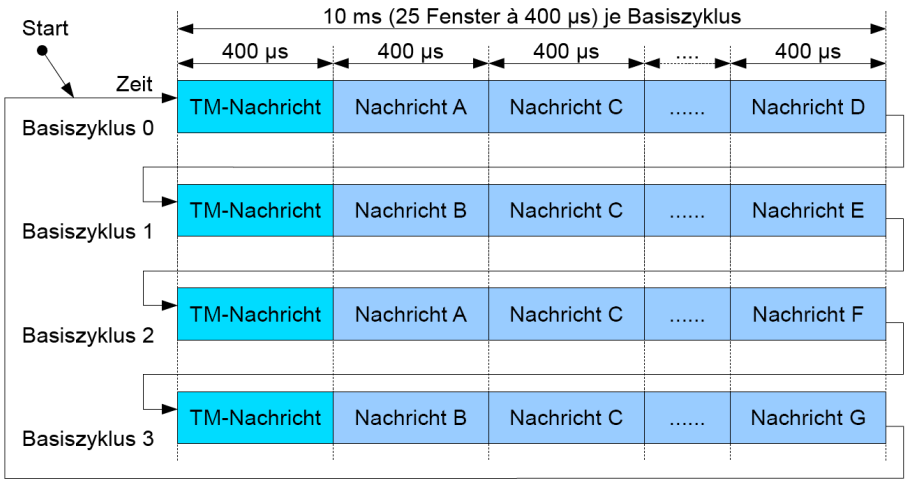

Abbildung 4.3.: Struktur des Matrixzyklus im Hawks-Telemetrie-System
Quelle: [Kolbe, 2008]

4.2.2. Scheduler

Um die gemeinsame Zeit für den TTCAN-Bus festzulegen, wird lokal auf den einzelnen Teilnehmern ein Scheduler eingesetzt. Die Zeiteinheiten des Schedulers werden als “Ticks“ bezeichnet und stellen die zu synchronisierende Zeit für den Bus dar.

Die Funktion des Schedulers ist es, verschiedene Aufgaben (engl. Tasks) zu verwalten und deren Ausführungszeit zu steuern. Somit spielt er die entscheidende Rolle, wenn es darum geht, einen Task zu einem genau bestimmten Zeitpunkt auszuführen, wie beispielsweise das Senden auf einen Time-Triggered-Bus. Der Scheduler bietet zudem die Möglichkeit eines regelmäßigen Tasks, der bei jedem „Tick“ ausgeführt wird. Dieser wird als „regtask“ bezeichnet und dient dazu, häufig anfallende kleine Aufgaben abzuarbeiten.

Dieser Ansatz bietet sich an, da alle Tasks, die auf den TTCAN-Bus senden müssen, periodisch abgearbeitet werden müssen. Somit kann dieser Ablauf über einen Scheduler kontrolliert werden. Die Funktionalität des implementierten Schedulers ist in [Schuckert, 2007] dokumentiert.

4.2.3. ECU-Gateway

Die Kommunikation mit der jetzigen ECU vom Hersteller Walbro läuft über die serielle Schnittstelle. Damit die Telemetriedaten der ECU auf dem TTCAN-Bus verfügbar sind, wird ein zusätzlicher Mikrocontroller eingesetzt. Dieser kommuniziert über die serielle Schnittstelle mit der Walbro und empfängt so die Telemetriedaten. Danach werden diese auf den TTCAN-Bus gesendet.

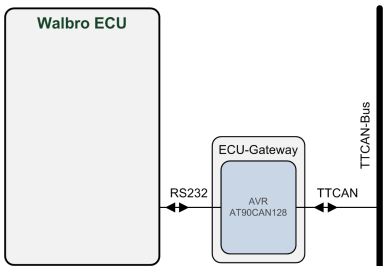

Abbildung 4.4.: Struktur der Kommunikation über das ECU-Gateway

Kommunikation mit der Walbro ECU

Das Kommunikationsprotokoll der Walbro ist nicht vollständig bekannt. Im Rahmen der Entwicklung des ECU-Gateways wurde es jedoch teilweise rekonstruiert. Es zeigte sich, dass ein Handshake zum Initialisieren des Datenstroms erfolgt. Danach sendet die Walbro ununterbrochen Daten. Das Ende eines Datenpakets wird mittels eines Trennzeichens („:“) signalisiert. Der genaue Ablauf das Handshakes ist in Abbildung 4.5 festgehalten.

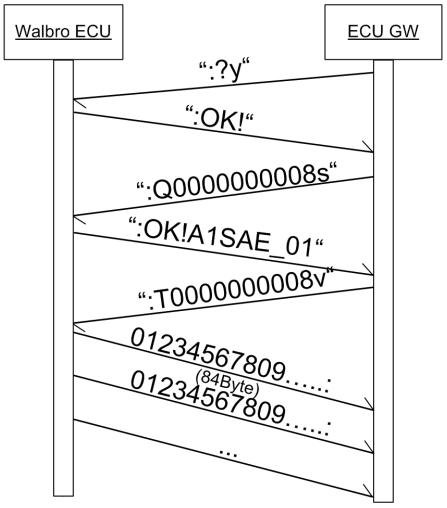

Abbildung 4.5.: Kommunikationsprotokoll der Walbro ECU zum Senden der Telemetriedaten

Die Daten werden als Hexadezimalwert übermittelt. Dies bedeutet, dass beispielsweise bei einer Drehzahl von $10000\frac{\mathrm{U}}{\mathrm{min}}$ der Wert als „2710“ übertragen werden würde. Dies bedarf damit $32\mathrm{bit}$ obwohl der Dezimalwert problemlos in $16\mathrm{bit}$ codiert werden könnte.

Gesendetes Datenformat

Damit nicht alle 84 empfangenen Bytes auf den CAN-Bus gesendet und so 12 Nachrichten benötigt würden, werden sie als Binärwert übertragen. So kann genau die Hälfte des Speicherbedarfs eingespart werden und es reichen 6 CAN-Nachrichten aus. Diese werden einmal pro Matrixzyklus versendet. Das entspricht einer Frequenz von $25\mathrm{Hz}$.

5. Analyse der MegaSquirt-ECU

Die Analyse der MegaSquirt-ECU gibt einen Einblick in das Konzept und die Funktionsweise der MegaSquirt. Es wird ermittelt, inwieweit die an die ECU gestellten Anforderungen erfüllt werden und welche Anpassungen und Modifikationen nötig sind.

5.1. Überblick

Die MegaSquirt-ECU (kurz MS) ist eine frei erhältliche Motorsteuerung, die von zwei amerikanischen Ingenieuren[1] entwickelt wurde. Das Ziel der Entwicklung war es, eine einfache ECU anzubieten, die günstig erworben werden und ohne fundierten technischen Hintergrund in ein Fahrzeug integriert werden kann. Hierbei wurde zudem darauf Wert gelegt, dass möglichst viele Motorenkonfigurationen unterstützt werden.
Es werden sämtliche Schaltpläne und Quellcodes frei angeboten, sodass nur noch die Hardware gekauft oder aufgebaut werden muss.

5.1.1. Entstehung

Ursprünglich wurde die MS als reine Electronic Fuel Injection (EFI) konzipiert und basierte auf einem Motorola 68HC908 8bit-Mikrocontroller. Es wurde lediglich die Einspritzung gesteuert.
Mit dem Fortschritt der technischen Entwicklung und den wachsenden Erwartungen an eine ECU wurde die MegaSquirt2 (MS2) entwickelt. Diese verwendet einen Motorola MC9S12C64 16bit-Mikrocontroller, der deutlich mehr Leistung bietet (vgl. Anhang A.2 Vergleich MegaSquirt1 und 2, S.115).
Ein Ziel bei der Entwicklung dieser neuen Version war es, abwärtskompatibel zu bleiben. Bisherige Besitzer der ersten Version sollten möglichst einfach aufrüsten

[1] Al Grippo und Bruce Bowling, 2003

können. Hierzu ist es nötig, dass die Hardwarebasis identisch tbleib und nur ein anderer Controller zum Einsatz kommt. Die Auswirkungen dieser Entscheidung werden in Kapitel 5.2.1 „Struktur der Hardware“ deutlich. [vgl. Murrey u. a., 2008]

5.2. Analyse der Hardware

Im Rahmen dieser Arbeit wird die Hardware-Version V3.57 genutzt. Diese ist in einem Aluminium-Gehäuse mit den Maßen $181\text{mm} * 105\text{mm} * 44\text{mm}$ (Länge$*$Höhe$*$Breite) verbaut. Das Gehäuse ist in Abbildung 5.1 dargestellt.

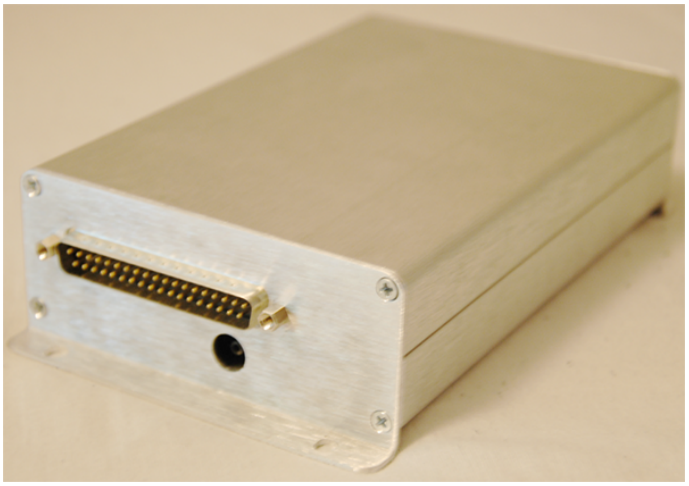

Abbildung 5.1.: MegaSquirt Gehäuse

Das Gehäuse bietet folgende Anschlüsse:

Steckerformat	Funktion
Sub-D37 male	Stromversorgung, Sensoren, Aktoren
Sub-D15 male	nicht beschaltet, für Modifikation und Erweiterungen vorgesehen
Sub-D9 female	Serielle Schnittstelle
Drucksensor, 4.5mm	MAP Drucksensor Anschluss

Tabelle 5.1.: MegaSquirt-Gehäuse-Schnittstellen

Die Belegung des Sub-D37 Steckers befindet sich im Anhang A.3 auf Seite 116.

5.2.1. Struktur der Hardware

Die Hauptplatine ist mit SMD-Elementen bestückt und beherbergt die komplette Peripherie. Es befindet sich ein DIP40 Sockel auf der Platine, der die von den Entwicklern gewünschte Kompatibilität zwischen der MegaSquirt und der MegaSquirt2 herstellt.(vgl. 5.1.1 Entstehung)
Dieser Sockel kann entweder mit dem Mikrocontroller für die MegaSquirt bestückt werden oder mit dem sogenannten „Daughterboard“ der MegaSquirt 2.
Dies ist eine zusätzliche Platine, die den Mikrocontroller der MS2 sowie einen CAN-Treiber und ein NAND-Gatter beherbergt. Das NAND-Gatter dient zur Realisierung einer sogenannten Peak and Hold- Schaltung, die zur Ansteuerung von niederohmigen Einspritzventilen dient. Abbildung 5.2 zeigt das Daughterboard.

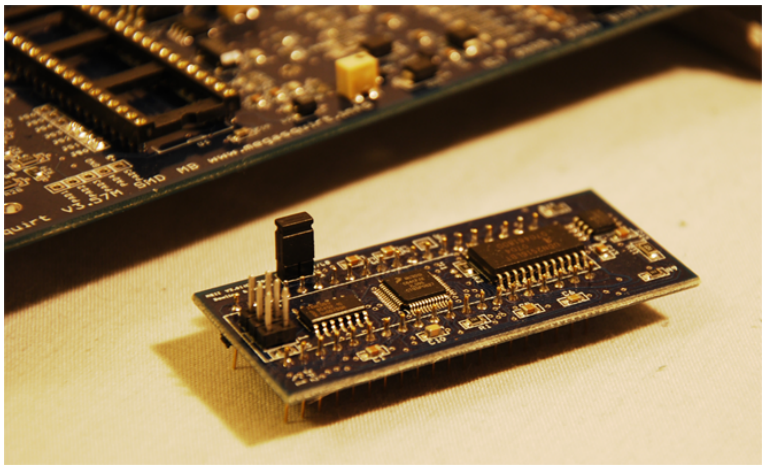

Abbildung 5.2.: MegaSquirt2 Daughterboard

Da dieses Daughterboard über den gleichen Sockel mit der Peripherie verbunden ist wie der Controller der MegaSquirt1, bildet sich dort ein Engpass. Es stehen nur 40 Pins zwischen Daughterboard und Hauptplatine zur Verfügung. Somit wird die Anzahl möglicher IO-Ports eingeschränkt.

5.2.2. Freescale MC9S12C64

Der in der MS2-Hardware verwendete Mikrocontroller bietet folgende Eigenschaften:

	Freescale (Motorola) MC9S12C64
Package	LQFP48
Architektur	16bit
Takt	24MHz
Flash	64KB
RAM	2KB
HW-Timer	8 * 16bit, gemeinsamer Takt
PWM-Kanäle	6 * 16bit
ADCs	8 * 10bit
I/Os	31, 29 shared
Kommunikation	SPI, SCI, CAN

Tabelle 5.2.: Überblick MC9S12C64
[siehe Freescale, 2007]

Von besonderem Interesse sind hier das Timer-Modul, das CAN-Modul sowie das Interrupt-Handling.

Das Timer-Modul verfügt über 8 separate Kanäle die auf einer gemeinsamen Taktquelle und einem gemeinsamen Prescaler basieren. Jeder Kanal kann entweder im Input-Capture (IC) oder Output-Compare (OC) betrieben werden, jeweils getrennt oder verbunden mit einem fest zugeordnetem Pin. Die Pins teilt sich das Timer-Modul mit dem PWM-Modul.

Das sogenannte „Freescale Scaleable CAN“-Interface bietet eine vollständige Unterstützung das CAN 2.0A/B Standards. Es beinhaltet einen umfangreichen Acceptance-Filter und ist somit für die Verwendung im TTCAN-Bus geeignet, da nur Nachrichten Interrupts auslösen, die auch wirklich benötigt werden. Ein „Resend-on-Error“-Verhalten ist nicht implementiert, sodass hier auch keine Probleme im TTCAN-Betrieb zu erwarten sind. Diese Methode wäre problematisch, da hier die Nachricht beim Aufreten eines Fehlers automatisch ein weiteres mal versendet werden würde. Dies wäre dann vermutlich außerhalb des vorgesehenen TTCAN-Zeitfensters und würde damit den Bus lahmlegen (vgl. 4.2.1 TTCAN, S.41).

Das Interrupt-Handling des MC9S12C64 Controllers ist simpel und berrscht kein preemtives, sprich unterbrechbares Handling und damit auch keine eingebetteten

Interrupts. Eine Priorisierung ist nur durch die Adressen der Interrupt-Vektoren der Interrupt-Quellen vorhanden. Diese ist allerdings auch nur relevant, wenn mehrere Interrupts zeitgleich anliegen. Dann erhält die ISR den Vorrang, die niedrigste Adresse aufweisen kann. [vgl. Freescale, 2007]

Die Pin-Belegung und die Auflistung von Pin- und Signal-Namen aus den Schaltplänen befindet sich im Anhang unter A.1 Pin Belegung HC9S12C64.

5.2.3. Peripherie

Die MegaSquirt bietet eine Vielzahl an Möglichkeiten um die im Auto typischen Sensoren und Aktoren zu betreiben. Die dafür nötige Elektronik befindet sich auf der Hauptplatine. Die Schaltpläne dazu befinden sich im elektronischen Anhang (siehe C Inhalt der CD, S.125).
Im Folgenden wird ein Überblick über die unterstützten Sensoren und Aktoren gegeben:

Sensoren

Es sind die folgenden Sensoren vorgesehen:

Kurbelwellensensor

Als Kurbelwellensensor werden sowohl Hall- als auch induktive Sensoren unterstützt (vgl. 2.6.1 Sensorik, S.28). In der entsprechenden Schaltung werden die Nulldurchgänge des Eingangssignals ermittelt und dementsprechend Pulse erzeugt.

Drosselklappenstellung

An der Achse der Drosselklappe ist ein Potentiometer angebracht. An dieses wird eine $5\mathrm{V}$-Referenzspannung angelegt und der entsprechende Spannungsabfall am Potentiometer über einen ADC gemessen (vgl. 2.6.1 Sensorik, S.28).

Saugrohrdruck

Der Drucksensor für den Saugrohrdruck ist direkt auf der Hauptplatine verbaut.

Temperaturen

Für Die Temperaturmessung von Kühlmittel und Ansaugluft sind passive Sensoren vorgesehen. Diese werden gegen Masse geschaltet und dann mittels einer Referenzspannung der Spannungsabfall über einen ADC gemessen.

Lambdasonde

Es wird die Ausgangsspannung eines externen Lambdasonden-Steuergeräts gemessen. Es werden sowohl Breit- als auch Schmalbandsonden unterstützt.

Batteriespannung

Die Betriebsspannung die an der ECU anliegt wird gemessen. Sie entspricht üblicherweise der Bordnetzspannung.

Eine tabellarische Zusammenfassung der Sensorik zeigt Tabelle 5.3.

Sensor	Abkürzung	Signal
Kurbelwellensensor	RPM	Pulse eines induktiven Sensors oder eines Hallgebers
Saugrohrdruck	MAP	Spannungspegel
Drosselklappenstellung	TPS	Spannungspegel
Ansaugluftemperatur	MAT	Spannungspegel
Kühlungstemperatur	CLT	Spannungspegel
Lambdasonde	Lambda	Spannungspegel
Batteriespannung	BAT	Spannungspegel

Tabelle 5.3.: Übersicht über die Sensor-Eingänge der MegaSquirt2, [vgl. Bowling und Grippo, 2008]

Der wichtigste Aspekt der Sensoren-Analyse ist hier das Fehlen eines Nockenwellensensors. Somit beherrscht die MegaSquirt2 weder eine vollsequentielle Einspritzung noch eine vollsequentielle Zündung (vgl. 2.4.3 Organisation der Einspritzung bei Mehrzylinder-Motoren, S.21).

Aktoren

Folgende Aktoren sind seitens der MegaSquirt-Hardware vorgesehen:

Einspritzung

Es sind zwei Einspritzkanäle inklusive Treiberbausteinen vorhanden. Die Einspritzung ist in einer sogenannten „Peak and Hold“-Schaltung aufgebaut. Diese dient dazu, auch niederohmige Einspritzventile, die nicht mit $12\mathrm{V}$ angesteuert werden können, verwenden zu können.
Hierzu wird kurzfristig die volle Spannung angelegt, um ein schnelles Ansprechen des Einspritzventils zu gewährleisten, und danach auf ein begrenzte Spannung reduziert.
Diese Schaltung wird durch Verwendung von jeweils einem PWM- und einem GPIO-Pin realisiert, die beide auf ein NAND-Gate gelegt sind. Die Funktion ist in Abbildung 5.3 dargestellt.

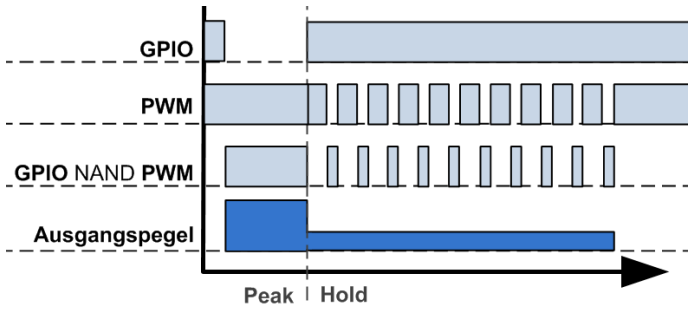

Abbildung 5.3.: Peak and Hold Prinzip

Diese Schaltung ist allerdings bei Hawks nicht nötig, da hier lediglich hochohmige Einspritzventile verwendet werden (vgl. 4.1.2 Verwendete Einspritzventile, S.40).
Wichtig ist hier der Umstand, dass nur zwei Einspritzkanäle zur Verfügung stehen und somit bei einem 4-Zylinder-Motor keine vollsequentielle Einspritzung möglich ist.

Zündung

Es ist lediglich ein Zündausgang seitens des Mikrocontrollers vorhanden. Auf der Hauptplatine ist ein Platz für einen Zündtreiber im TO220 vorgesehen, allerdings ist er nicht verbaut worden.
Diese Struktur dient dazu, entweder eine zentrale Zündspule mit externem Zündverteiler anzusteuern, oder ein externes elektronisches Zündmodul zu

kontrollieren. Eine vollsequentielle Zündung, wie sie im Hawks-Projekt verwendet wird (vgl 4.1.2 Verwendete Zündanlage, S.page4.1.2), ist hier nicht vorgesehen.

Kraftstoffpumpe

Das Kraftstoffpumpenrelais wird von der ECU gesteuert um bei Stillstand des Motors keinen Druck auf der Kraftoffverteilung zu haben.

Die vorgesehene Ansteuerung für die Aktoren für Einspritzung und Zündung ist unzureichend und so nicht für das Hawks Projekt brauchbar.

5.2.4. MegaStim - Stimuliboard

Um nicht zwingend auf einen Motor und dessen Sensorik angewiesen zu sein bieten die Entwickler der MegaSquirt eine zusätzliche Platine an, die die Sensorik eines Fahrzeugs simuliert. Diese wird als MegaStim bezeichnet.

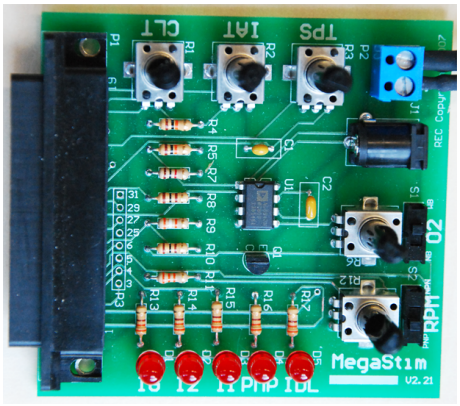

Abbildung 5.4.: MegaStim Platine

Die Platine verfügt über einen DB37-Stecker und kann direkt an den entsprechenden Stecker der MegaSquirt gesteckt werden. Das MegaStim simuliert folgende Sensoren:

- Drehzahl, Pulse eines 32-0-Geberrads
- Drosselklappen-Potentiometer
- Ansauglufttemperatur
- Kühlungstemperatur
- Lambda-Sonde

Leider besteht hier keine Möglichkeit das Drehzahl-Signal auf einfache Weise anzupassen. Es würden die Pulse eines 24-2 Geberrades benötigt sowie ein Puls der Nockenwelle. Hier muss eine eigene Signalquelle konzipiert werden. Die restlichen Stimuli können aber zur Simulation genutzt werden.

5.3. Analyse der Embedded-Software

Der Embedded-Code der MegaSquirt2 ist in der Programmiersprache C geschrieben und hat einen Umfang von etwa 10000 Zeilen. Er ist leider unstrukturiert und lässt einen eindeutigen Codingstyle und oder eine Namenskonvention vermissen. Dies erschwert die Analyse und das Arbeiten mit diesem Code.
Eine Recherche ergab, dass es ein geeigneteres Code-Derivat mit dem Namen „MegaSquirt Extra“ existiert. Ein Vergleich der Eigenschaften beider Versionen ist in Tabelle 5.4 festgehalten.

	MegaSquirt II	**MegaSquirt II Extra**
Genauigkeit Einspritzung	$0.67\mu s$	$0.67\mu s$
Einspritzungs-Kennfeld	12*12	16*16
Anzahl Einspritzkanäle	2	2
Genauigkeit Drehzahl	$1\frac{U}{min}$	$1\frac{U}{min}$
Größe Zündwinkel-Kennfeld	12*12	12*12
Zündwinkel-Auflösung	$0.1°KW$	$0.1°KW$
Anzahl Zündausgänge	1	4
Nockenwellen-Sensor	Nein	Ja
Kennfeld-Wechsel	Nein	Ja

Tabelle 5.4.: Vergleich des Funktionsumfangs von MS2 und MS2extra
[vgl. Murrey u. a., 2008]

Der markante Vorteil der MegaSquirt Extra liegt im Vorhandensein von bis zu 4 Zündkanälen und der Unterstützung eines Nockenwellensensors. Diese beiden Punkte sind ausschlaggebend dafür, dass sich hier für die Verwendung dieses Code-Derivats entschieden wird. Unter dem Namen „MegaSquirt Extra“ (MSextra) haben drei britische Ingenieure den MS2 Code umstrukturiert und auf den freien GCC-Compiler portiert. Ihre Zielsetzung war es, die MS ihren Ansprüchen anzupassen und den Funktionsumfang zu erweitern.
Der Quellcode ist besser dokumentiert und strukturiert, was sich beispielsweise

schon durch die triviale Maßnahme äußert, dass hier der Code auf mehrere Dateien verteilt wurde.
Der Quellcode nutzt die gleiche Hardware-Basis. Diese muss allerdings modifiziert werden, wenn man beispielsweise die vollsequentielle Zündung nutzen will. Diese Modifikationen sind auf der Website der Entwickler gut und detailliert dokumentiert. [vgl. Murrey u. a., 2008]

5.3.1. Struktur des Embedded-Codes

Es werden die wichtigsten Komponenten vorgestellt und deren Zusammenspiel dargestellt. Es wird nicht im Detail jede Funktion erläutert, da hier das Verständnis des Konzepts im Vordergrund steht, das in Form der MegaSquirt umgesetzt wurde.

Die wesentlichen Elemente der MegaSquirt Extra sind die folgenden:

Super Loop

Eine Schleife, die endlos durchlaufen wird und damit zeitunkritische Aufgaben abarbeitet. Sie wird nur von Interrupt Service Routinen unterbrochen. Die Häufigkeit ihrer Abarbeitung hängt dabei davon ab, wie oft ISRs aufgerufen werden und wie lange deren Abarbeitung dauert.

Real Time Clock

Die Real Time Clock (RTC) ist ein spezieller Timer, der einen hochpriorisierten Interrupt in einem regelmäßigen Intervall auslöst. Im Falle der MSextra ist dieses Intervall $128\mu s$ lang. Diese ISR wird vorrangig für Software-Timer genutzt. Durch diese SW-Timer können Eingangsgrößen in Bezug zur Zeit gesetzt werden.

Input Capture ISR für Kurbel- und Nockenwellensensor

Um die Pulse des Kurbelwellen- und des Nockenwellensensors zu erkennen werden zwei Timer im Input Capture-Modus verwendet. Diese stoßen bei einem Puls an einem I/O-Pin eine ISR an. Somit kann das exakte Zeitintervall zwischen zwei Pulsen bestimmt und somit die Drehzahl gemessen werden.

Timer ISRs für Aktoren

Für alle zeitkritischen Aktoren wie beispielsweise Zündung und Einspritzung werden Timer im Output Compare-Modus genutzt. Diese stoßen nach Ablauf einer definierten Zeit eine ISR an. Diese wird in hier lediglich zum Setzen eines IO-Pins zu einem definierten Zeitpunkt genutzt.

5.3.2. Analyse der Funktionalitäten

Um die wesentliche Funktionsweise der MegaSquirt Extra zu verstehen, wird der Code näher untersucht. Hierzu wird erläutert, wie die MSextra folgende essentiellen Aufgaben löst.

- **Erkennung der Drehzahl und der Kurbelwellenstellung**
- **Berechnung von Parametern**
- **Timing und Ansteuerung von Aktoren**

Erkennung der Drehzahl und der Kurbelwellenstellung

Die MegaSquirt Extra verwendet Timer im Input-Capture Modus um die Pulse des Kurbelwelleninkrementalgebers auszuwerten. Diese ISR, die als ISR_Ign_TimerIn bezeichnet wird, misst dabei die Zeit zwischen den Pulsen und ermittelt so die Drehzahl. Um den Winkel der Kurbelwelle zu ermitteln wird ein Bezugspunkt benötigt. Dieser wird durch die Lücke im Geberrad der Kurbelwelle dargestellt (vgl. 2.6.1 Sensorik, S.28).

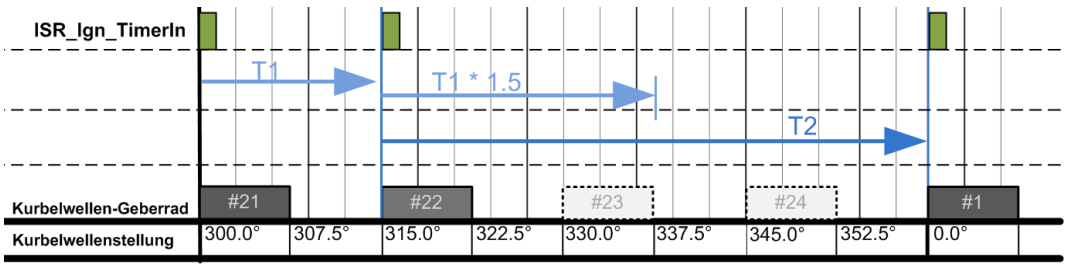

Abbildung 5.5.: Erkennung des fehlenden Zahns des Geberrads

Die ISR erkennt diese Lücke durch den Vergleich der Zeiten zwischen den Pulsen. Das Verfahren ist in Abbildung 5.5 dargestellt. Wenn die aktuelle Zeitdifferenz $T2$ zwischen zwei Pulsen um den Faktor 1.5 länger ist als die vorangegangene Differenz $T1$, wird der aktuelle Zahn als erster Zahn nach der Lücke angesehen. Dieses Verfahren ist einfach aber effektiv, da sich die Winkelgeschwindigkeit der Kurbelwelle durch die Massenträgheit des Motors nicht derart schlagartig ändern kann.
Ist der erste Zahn erkannt, so werden die weiteren Pulse durchgezählt. Durch die bekannte Anzahl an Zähnen kann so auf die aktuelle Stellung der Kurbelwelle geschlossen werden.

Die Pulse des Nockenwellensensors werden ebenfalls durch einen Timer im Input-Capture Modus abgefragt. Deren ISR setzt allerdings nur ein Statusflag und führt dann einen Sprung in die ISR der Kurbelwelle aus. Der exakte Zeitpunkt des Nockenwellenpulses spielt hierbei keine Rolle. Es wird lediglich überprüft, dass der Puls in jeder zweite Umdrehung erfasst wird.

Berechnung von Parametern

Die Berechnung der Betriebsparameter für die Aktoren findet in der Super Loop statt. Hier werden die ADCs ausgewertet deren Werte als Stützstellen für die jeweiligen Kennfelder dienen (vgl. 2.6.2 Kennfelder, S.29).
Der Vorteil dieses Ansatzes ist, dass jede Zeit außerhalb der Interrupt Service Routinen zum Berechnung der Parameter verwendet werden kann. Nachteil ist, dass so keine definitive Aussage darüber möglich ist, wann die aktuell verwendeten Parameter berechnet wurden. Es sind nur Schätzungen möglich, die auf der Zeit basieren, die benötigt wird um die Super Loop zu durchlaufen. Diese wird im folgenden Kapitel 5.5 „Analyse von Timing und Genauigkeit“ auf Seite 62 gemessen und diskutiert.

Timing und Ansteuerung von Aktoren

Wenn die Berechnung der Parameter abgeschlossen ist, werden die dazugehörigen Kurbelwellenwinkel berechnet. Es wird der Zahn des Geberrads ermittelt, der am dichtesten vor dem errechneten Winkel liegt. Wenn dieser „Trigger“-Zahn die Kurbelwellen-ISR auslöst, wird die restliche Wartezeit mittels eines Hardware-Timers abgearbeitet. Das letzendliche Setzen der Ausgangspins zum Ansteuern der Aktoren geschieht dann in den entsprechenden Service Routinen.

Im Folgenden wird der genaue Ablauf eines Zündungs- und eines Einspritzungsereignisses betrachtet.

Organisation der Zündung

Die entscheidenden Faktoren der Zündung sind der Zündwinkel und die Schließzeit (vgl. 2.5.1 Zündwinkel, S.23). Folglich sind beide Ereignisse, das Schließen und das Öffnen des Zündtreibers, auf einen möglichst exakten Zeitpunkt angewiesen.
Die MSextra organisiert beide Ereignisse mittels eines Hardware-Timers. Um Timer zu sparen, sind diese Timer nicht direkt mit ihrem zugewiesenen OC-Pin verbunden sondern steuern GPIO-Pins an. Dies ist sinnvoll, da sonst mindestens ein Timer pro verwendetem Zündkanal nötig wäre.

In Abbildung 5.6 ist der zeitliche Ablauf dargestellt. Hier wurde eine Drehzahl von $1000\frac{\mathrm{U}}{\mathrm{min}}$, ein Zündwinkel von $25°\mathrm{KW}$ und eine Schließzeit von $2\mathrm{ms}$ angenommen.

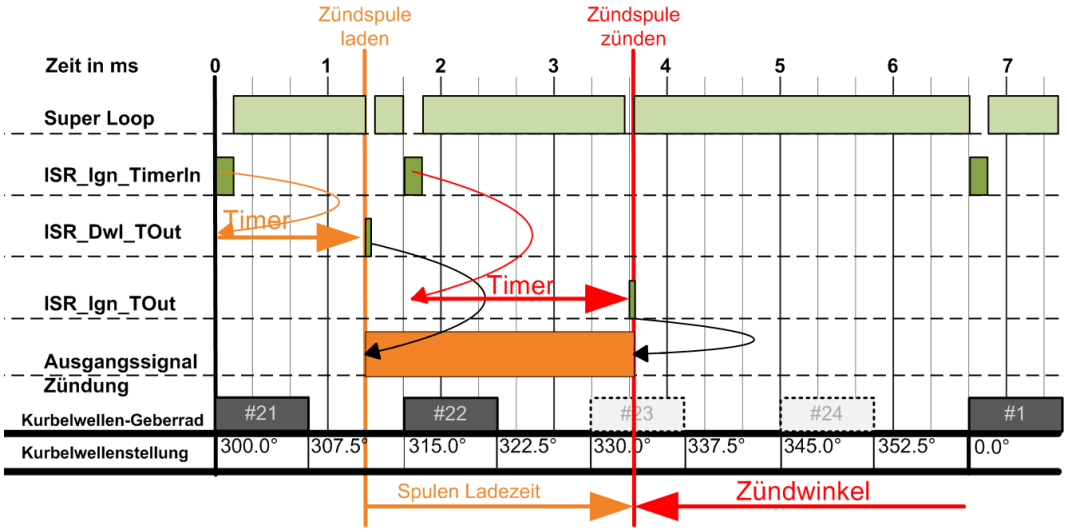

Abbildung 5.6.: Softwareseitige Organisation der Zündung

Organisation der Einspritzung

Im Gegensatz zur Zündung spielt bei der Einspritzung nicht der exakte Start- und Endzeitpunkt die entscheidende Rolle sondern das exakte Zeitintervall zwischen diesen Ereignissen. Die MegaSquirt nutzt diesen Umstand aus und legt das Öffnen der Einspritzventile auf einen „Zahn“ der Kurbelwelle. Es wird der Zahn ausgewählt, der möglichst dicht an $350°\mathrm{KW}$ liegt, also $10°\mathrm{KW}$ vor

dem OT. Seitens der Konfiguration kann auf diesen Wert kein Einfluss genommen werden.
So übernimmt die ISR_Ign_TimerIn die Aufgabe, das Einspritzventil zu öffnen. Folglich wird hier kein weiterer Timer benötigt. Das Schließen der Einspritzventile übernimmt eine Timer-ISR. Über die Laufzeit dieses Timers kann folglich die Öffnungszeit der Ventile genau bestimmt werden.
Dieser Ablauf ist in Abbildung 5.7 dargestellt.

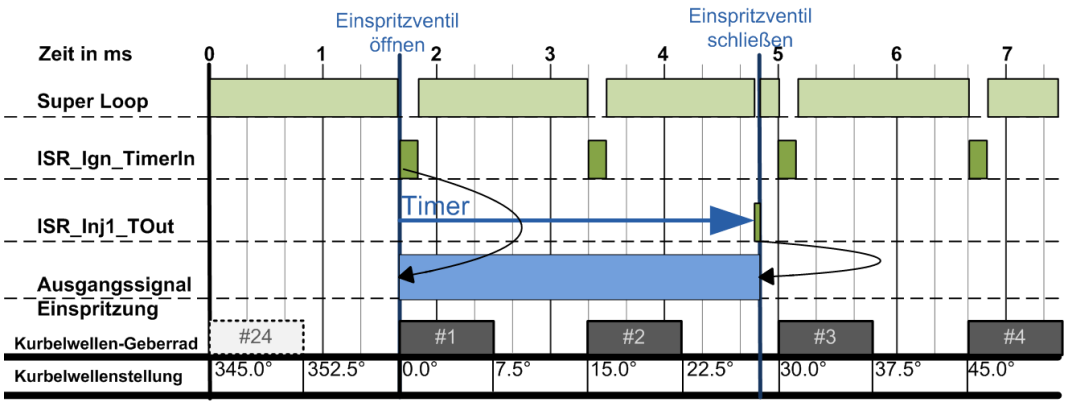

Abbildung 5.7.: Softwareseitige Organisation der Einspritzung

Das System der MegaSquirt Extra kann als „ereignisorientiert“ bezeichnet werden. Es wird nicht abgefragt, ob ein Ereignis eingetroffen ist, sondern es wird auf das Ereignis direkt gewartet. Diese Wartezeit kann für die Superloop genutzt werden um zeitunkritische Aufgaben abzuarbeiten. [vgl. Pont, Michael J., 2001]

Ein alternativer Ansatz ist hier kaum zu finden, da der Bezugspunkt für alle zeitrelevanten Ereignisse die Umdrehung der Kurbelwelle und damit die Pulse des Kurbelwellensensors sind. Da sich das Zeitintervall zwischen dessen Pulsen mit der Motordrehzahl verändert, ist ein „Time-Triggered“-Ansatz auf Basis eines Schedulers nur schwer möglich. Hier müssten Start- und Endzeiten der Tasks dynamisch mit der Drehzahl variiert werden, was einen immensen Verwaltungsaufwand mit sich bringen würde. Zudem müsste die Drehzahl durch sogenanntes „Polling“ erkannt werden. Hierfür müsste die Frequenz des Schedulers deutlich über der maximalen Frequenz der Pulse liegen.

Bei dem im Hawks-Projekt verwendeten Flywheel mit 24 Zähnen und einer theoretischen maximalen Drehzahl von $16000\frac{\mathrm{U}}{\mathrm{min}}$ entspricht dies einer Frequenz von $6.4\mathrm{kHz}$. Beim verwendeten Motorola Mikrocontroller mit einem Takt von $24\mathrm{MHz}$ ist es so kaum zuverlässig möglich, die Pulse ohne eine Input-Capture-Funktion zu erkennen.

5.4. Konfigurations-Software

Zur Konfiguration dient die Windows-Software „MegaTune“. Diese kann Echtzeitwerte der ECU anzeigen sowie die komplette Konfiguration vornehmen. Die Kommunikation zwischen ECU und Software läuft über die serielle Schnittstelle.

Abbildung 5.8.: MegaTune Konfigurationssoftware

Die Software kann komplett über Konfigurationsdateien angepasst und modifiziert werden. Dies betrifft sowohl die Form der Darstellung als auch die Menüstruktur der Konfigurations-Einstellungen. Es kann also die Konfiguration von Modifikationen ebenfalls über MegaTune erfolgen.

5.4.1. Funktionsweise

Die Serielle Schnittstelle der MS2 läuft standardmäßig auf $115200\mathrm{baud}$, 8 Datenbits ohne Parität mit Stopbit. Die beiden wesentlichen Funktionen sind das Auslesen von Variablen zur Laufzeit und das Schreiben von Konfigurations-Daten.

Lesen der Laufzeit-Variablen

Die Laufzeitvariablen müssen mit einem Kommando von der ECU angefordert werden. Die ECU sendet daraufhin alle Laufzeitvariablen ohne Trennzeichen und ohne Pause. Im Gegensatz zur Walbro muss hier jedes Datenpaket einzeln angefordert werden. (vgl. Abbildung 5.9)

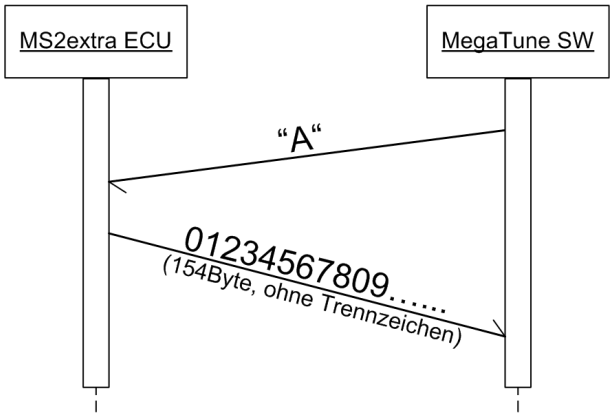

Abbildung 5.9.: MegaSquirt2 Extra Telemetrieprotokoll

Die MegaTune-Software sendet zum Anzeigen der Variablen alle 200ms das entsprechende Kommando. Die Laufzeitvariablen umfassen 154 Byte. Das Senden bzw. Empfangen nimmt also folgende Zeit in Anspruch:

$$t = \frac{154\text{B} * (8\text{bit} + 1\text{bit})}{115200\text{baud}} = \underline{12.03\text{ms}}$$

Die vollständige Auflistung des Inhalts der Telemetriedaten befindet sich in Anhang A.4 „Telemetriedaten der MegaSquirt Extra“ auf Seite 117.

Setzen von Konfigurationsoptionen

Um die Konfiguration der MSextra ohne ein erneutes Flashen des Mikrocontrollers zu ermöglichen, wird folgender Ansatz verwendet:
Alle Konfigurationskonstanten liegen an festgelegten Speicheradressen im Flash-Speicher des Controllers. Wird nun seitens des Konfigurationsprogramms ein Wert verändert, so wird dieser an die entsprechende Stelle in den Speicher des Mikrocontrollers geflasht.

Dafür müssen folgende Vorraussetzungen erfüllt sein:

- **Konfigurations-Software muss Speicheradressen kennen**

  Die Konfigurations-Software muss die Speicheradressen der jeweiligen Konstante kennen um den neuen Wert an die entsprechende Stelle schreiben zu können.

- **Der Quellcode enthält Werte der Konfiguration**

  Der Compiler benötigt für Konstanten wie die Konfigurationsoptionen einen zugewiesenen Wert. Somit steht im Quellcode eine vollständige Konfiguration. Somit muss nach jedem Aufspielen einer neuen Programmversion die Konfiguration geprüft werden, da sie überschrieben wurde.

- **Der Compiler darf Konfigurationskonstanten nicht optimieren**

  Der Compiler darf keine Optimierungen vornehmen, die auf einer Konfigurationsoption basieren. Auch wenn diese Werte zur Compile-Zeit bekannt sind, muss der Compiler sie als unbekannt annehmen.
  Durch die große Anzahl an Konfigurationsmöglichkeiten resultieren dementsprechend viele Entscheidungen im Quellcode, die allesamt immer durchlaufen werden. Die Auswirkungen werden in Kapitel 5.5.2 „Auswirkung der fehlenden Compileroptimierung“ auf Seite 64 näher diskutiert.

5.5. Analyse von Timing und Genauigkeit

Die Auslastung der ECU wird durch Messung der Ausführungsdauer und der Intervalle der einzelnen ISRs ermittelt. Anhand der Werte kann ermittelt werden, wieviel Prozent der Rechenzeit der Controller in der Super Loop verbringt. Misst man die Dauer eines Loop-Durchlaufs, lässt sich zudem ermitteln, wie viele Durchläufe pro Motor-Umdrehung erreicht werden.
Dies lässt Rückschlüsse auf die Genauigkeit und die Regelgeschwindigkeit der ECU zu.

5.5.1. Messbedingungen

Um der angestrebten Konfiguration im Fahrzeug möglichst nahe zu kommen wird die ECU für eine halbsequentielle Einspritzung und eine vollsequentielle Zündung mit einem 24-2 Geberrad konfiguriert (vgl. 2.6.1 Sensorik, S.28. Als Drehzahl wurden 16000$\frac{\mathrm{U}}{\mathrm{min}}$ gewählt, das theoretische Maximum des verwendeten Kawasaki Motors (vgl. 4.1 Motor, S.37).

Gemessen wurde mit einem Tektronix DPO300 Oszilloskop. Es wurde jeweils am Anfang der entsprechenden ISR in GPIO-Pin auf High und am Ende der ISR wieder auf Low gesetzt. Die Länge des Pulses wurde gemessen. Die Latenz des Ports hat keinen Einfluss auf die Messung, da diese sowohl am Anfang als auch am Ende auftritt und somit die Zeitdifferenz nicht beeinflusst.
Es wurden jeweils 5 Messungen gemacht um einen aussagekräftigen Durchschnittswert zu erhalten.

5.5.2. Ergebnisse Timinganalyse

Nach der in den Messbedingungen beschriebenen Methode wurden folgende Zeiten gemessen:

Auslastung durch die ISRs

Die Ergebnisse der Zeitmessungen der ISRs sind in Tabelle 5.5 festgehalten. Zudem wird hier angegeben, wie oft die entsprechende ISR ausgeführt während einer Umdrehung ausgeführt wird.

Bei der angenommenen Drehzahl von $16000\frac{\mathrm{U}}{\mathrm{min}}$ braucht eine Umdrehung folgende Zeit:

$$t_U = \frac{60\mathrm{s}}{16000\frac{\mathrm{U}}{\mathrm{min}}} = \underline{3.75\mathrm{ms}}$$

Von dieser Zeit wird folgender Anteil mit der Abarbeitung der ISRs verbracht:

$$t_{ISR} = \frac{3.75\mathrm{ms}}{0.128\mathrm{ms}} * 11.4\mu\mathrm{s} + 22 * 29.2\mu\mathrm{s} + \frac{1}{2} * 15.3\mu\mathrm{s} + 2 * 4\mu\mathrm{s} + 2 * 3.2\mu\mathrm{s} + 1 * 1.4\mu\mathrm{s}$$

$$t_{ISR} = \underline{1023.15\mu\mathrm{s}}$$

Während einer Umdrehung werden also 28% der Rechenzeit für die Abarbeitung der ISRs benötigt. Es bleiben folglich 72% für die Super Loop.

Minimale Regelfrequenz

Die Berechnungen der Parameter finden in der Super Loop statt. Da auf diese Weise keine eindeutige Aussage über den exakten Zeitpunkt der Berechnungen gemacht werden kann, wird hier gemessen, wie lange ein Durchlauf durch die Super Loop dauert. Somit kann die minimal mögliche Regelfrequenz, die bei maximaler Auslastung der ECU auftritt, bestimmt werden.

$$f = \frac{1}{1108\mu\mathrm{s}} = \underline{902.5\mathrm{Hz}}$$

Die Regelfrequenz liegt also deutlich über der maximal möglichen Frequenz an Steuernachrichten des TTCAN-Bus. (vgl. 4.2.1 Matrixzyklus, S.41).

ISR	Anmerkung	Dauer	Ausführung
ISR_RTC	Real Time Clock	$11.4\mu\mathrm{s}$	alle $0.128\mathrm{ms}$
ISR_Ign_In	Input Capture Kurbelwelle	$29.2\mu\mathrm{s}$	$\frac{22}{\mathrm{U}}$
ISR_Ign_In	Input Capture Nockenwelle	$15.3\mu\mathrm{s}$	$\frac{1}{2*\mathrm{U}}$
ISR_Dwl_TimerOut	Schließzeit Timer	$4.0\mu\mathrm{s}$	$\frac{2}{\mathrm{U}}$
ISR_Ign_TimerOut	Zündungstimer	$3.2\mu\mathrm{s}$	$2\frac{1}{\mathrm{U}}$
ISR_Inj1_TimerOut	Einspritzungs Ende 1	$1.4\mu\mathrm{s}$	$\frac{1}{2*\mathrm{U}}$
ISR_Inj2_TimerOut	Einspritzungs Ende 2	$1.4\mu\mathrm{s}$	$\frac{1}{2*\mathrm{U}}$

Tabelle 5.5.: Zeitmessungen der ISRs

	Anmerkung	Dauer
Super Loop	bei $16000\frac{\mathrm{U}}{\mathrm{min}}$	$1108\mu\mathrm{s}$

Tabelle 5.6.: Zeitmessungen der Superloop

Auswirkung der fehlenden Compileroptimierung

In Kapitel 5.4.1 „Funktionsweise“ wurde beschrieben, dass der Compiler keine Optimierung auf Basis der als static definierten Konfigurationsvariablen vornimmt. Die Auswirkungen dieser Maßnahme werden anhand eines Beispiels erläutert.

Um die Auswirkungen zu analysieren wird die Ausführungszeit der ISR der Real Time Clock (vgl. 5.3.1 Struktur des Embedded-Codes) gemessen. Diese wurde für dieses Beispiel gewählt, da sie die am häufigsten aufgerufende Methode der MegaSquirt ist. Nach der Zeitmessung im ursprünglichen Zustand wird die Service Routine dahingehend optimiert, dass alle Entscheidungen und bedingten Sprünge, deren Ergebnisse zur Compilezeit schon bekannt sind, optimiert werden. So werden unnötige Abfragen vermieden.

Das Ergebnis ist in der folgenden Tabelle 5.7 festgehalten.

	Anmerkung	Dauer
ISR_RTC	original	$11.4\mu\mathrm{s}$
ISR_RTC	optimiert	$8.8\mu\mathrm{s}$
	Differenz	$2.6\mu\mathrm{s}$

Tabelle 5.7.: Messung des Einflusses der Compileroptimierung

Es würden bei der gewählten Konfiguration $2.6\mu\mathrm{s}$ pro Aufruf gespart. Da diese ISR in einem Intervall von $0.128\mathrm{ms}$ aufgerufen wird, würde dies bei der Maximaldrehzahl pro Umdrehung eine Ersparnis von $78\mu\mathrm{s}$ beziehungsweise von 22.8% bringen. Bezogen auf die Summe aller Service Routinen entspräche dies 7.6%.

Diese Variante der Konfiguration geht also deutlich zu Lasten der Leistung.

5.5.3. Genauigkeit

Anhand der Software-Konfiguration wird eine Betrachtung der möglichen Genauigkeit vorgenommen. Hier wird ausschließlich die maximale Genauigkeit der Software betrachtet, Einflüsse der Hardware werden nicht berücksichtigt.

Zündung

Das Timer-Modul der MegaSquirt läuft mit einem Takt von $1.5\mathrm{MHz}$, also einer Auflösung von $0.66\mu\mathrm{s}$. Bei der Maximaldrehzahl von $16000\frac{\mathrm{U}}{\mathrm{min}}$ lässt sich der in der Zeit zurückgelegte Winkel α der Kurbelwelle errechnen:

$$\alpha = \frac{t * 360° * rpm}{60\mathrm{s}}$$

$$\alpha = \frac{0.66\mu\mathrm{s} * 360° * 16000\frac{\mathrm{U}}{\mathrm{min}}}{60\mathrm{s}} = \underline{\mathbf{0.0641}°}$$

Die Genauigkeit der Software liegt also selbst bei der Maximaldrehzahl deutlich unterhalb dem in den Anforderungen festgelegten Wert vom $\frac{1}{10}tel$ Grad. Seitens der Software gibt es also keinen Optimierungsbedarf um die Anforderungen zu erfüllen (vgl. 3.2.3 Genauigkeit, S.35).

Einspritzung

Die Genauigkeit der Einspritzung kann in Form der Öffnungszeit der Einspritzventile angegeben werden. Bei der Timer-Frequenz von $1.5\mathrm{MHz}$ beträgt die Genauigkeit seitens der Software $0.66\mu\mathrm{s}$. Auch dieser Wert genügt den gestellten Anforderungen.

6. Konzeption

Nach der Analyse der MegaSquirt-ECU und der Analyse der Auto-Infrastruktur ergaben sich folgende Punkte, die für den Betrieb im Hawks-Projekt angepasst werden müssen:

- Einspritzung
- Zündung
- Senden der Telemetriedaten

Hinzu kommt die Ergänzung der dynamischen Konfiguration. Es werden nun die Möglichkeiten für diese Anpassungen und Erweiterungen diskutiert.

6.1. Modifikation der Zündung

Ab Werk verfügt die MS2 über nur einen Zündkanal. Dieser kann zur Ansteuerung eines externen Zündverteilers genutzt werden. Für eine ruhende Verteilung (vgl. S.25) bei einem 4-Zylinder-4-Takt-Motor sind jedoch mindestens zwei Kanäle nötig um eine Wasted Spark -Zündung zu realisieren.
Für eine komplett sequentielle Zündung sind vier separate Ansteuerungen und Treiber nötig.

6.1.1. Lösungsansätze für die Zündung

Sowohl die Möglichkeit der Wasted Spark-Zündung als auch die Möglichkeit der vollsequentiellen Zündung sind seitens MSextra softwareseitig implementiert. Die daraus resultierenden Hardware-Modifikationen sind ebenfalls dokumentiert. [vgl. Murrey u. a., 2008]

Ansatz 1: Wasted Spark

Hierzu wird ein Port der drei Status-LEDs der MegaSquirt direkt abgegriffen und zum Ansteuern eines zusätzlichen Treibers verwendet. Der Vorteil dieser Lösung ist, dass sie keinen Nockenwellengeber benötigt und daher einfach umzusetzen ist.

Ansatz 2: Vollsequentielle Zündung

Es werden alle drei Ports der Status-LEDs direkt abgegriffen und zusätzlich zum schon bestehenden Port zur Ansteuerung der Zündtreiber verwendet. In Abbildung 6.1 ist dargestellt, wo im Schaltplan die Zündendstufe angeschlossen wird:

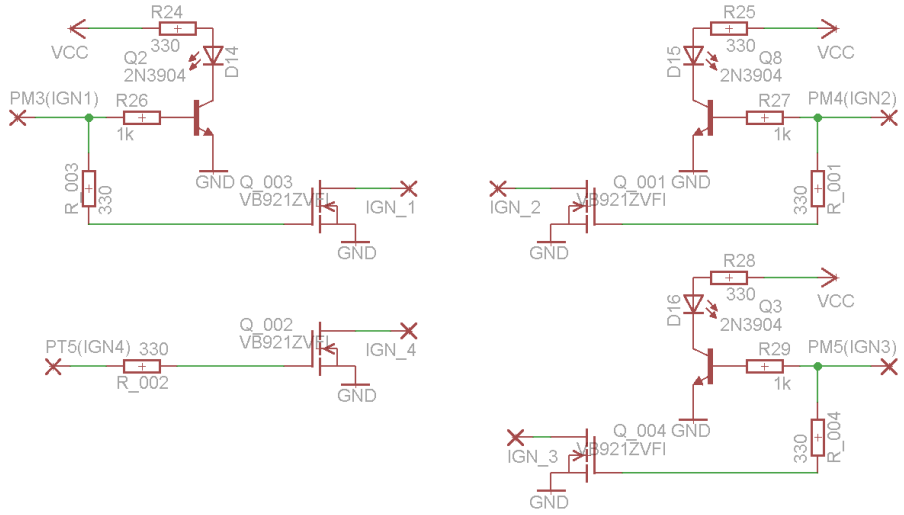

Abbildung 6.1.: Schaltplan für die vollsequentielle Zündung

6.1.2. Entscheidung für die vollsequentielle Zündung

Da für die vollsequentielle Einspritzung ohnehin ein Nockenwellensignal benötigt wird und der sonstige Mehraufwand überschaubar ist, wird hier die vollsequentuelle Einspritzung implementiert. Der Mehrbedarf an GPIOs ist hier relativ zu sehen, da diese bisher nur dafür genutzt wurden simple Debuginformationen über den Status der ECU anzuzeigen.
Diese Funktionalität soll zukünftig von der Telemetrie übernommen werden. (siehe 6.3 Telemetriekonzept, S.70)

6.2. Modifikationen der Einspritzung

Die MegaSquirt verfügt im Auslieferungszustand nur über zwei separate Treiber für die Einspritzung. Damit wäre eine vollsequientielle Einspritzung (vgl. Organisation der Einspritzung bei Mehrzylinder-Motoren, S.21) nicht möglich.
Da diese aber eine der Anforderungen an diese Arbeit ist, muss die ECU modifiziert werden. (vgl. Vollsequentielle Einspritzung, S.34)

6.2.1. Lösungsansätze für die Einspritzung

Um vier Einspritzventile getrennt anzusteuern sind vier Treiberbausteine notwendig. Es werden folglich zwei zusätzliche Treiber benötigt. Dabei sollte es sich möglichst um den gleichen Treiber-Baustein handeln, um Unterschiede in der Belastbarkeit und der Schaltzeit auszuschließen.
Für die Ansteuerung der Treiber gibt es zwei Lösungsansätze:

Ansatz 1: Lösung durch Hardware Logik-Gatter

Es wird ein zusätzlicher GPIO-Pin benötigt, der als Select-Signal fungiert und mit der Phase des Motors (vgl. 2.2 4-Takt-Zyklus, S.14) seinen Pegel wechselt. Dieses Signal wird mit dem Signal der Einspritzungs-Ansteuerung mittels eines AND-Gatters verknüpft (vgl. Abbildung 6.2).
Der Vorteil dieser Lösung wäre, das die „Peak&Hold“-Schaltung der MegaSquirt weiterhin verwendet werden könnte. Dadurch wäre zudem auch die Verwendung von niederohmigen Einspritzventilen möglich (vgl. 5.2.3 Aktoren, S.51).

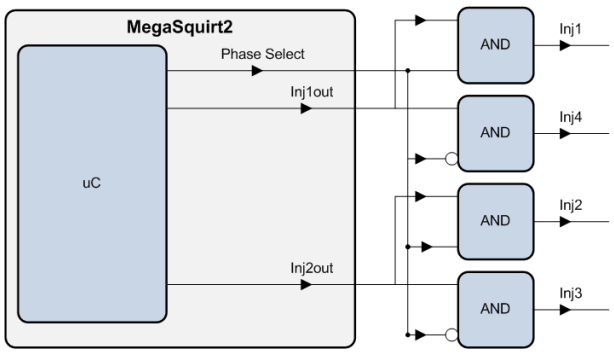

Abbildung 6.2.: Struktur des Hardware-Lösungs-Ansatz

Der Nachteil ist allerdings, dass hierbei nur ein Einspritzfenster von maximal 360°KW möglich ist, da danach die Phase und damit auch das „Select“-Signal wechselt. Würde dieser Punkt überschritten, würde in der Konsequenz ein falscher Treiber angesteuert und somit in den falschen Zylinder zum falschen Zeitpunkt eingespritzt.
Mit der im Hawks-Projekt verwendeten Einspritzung bei geschlossenen Ventilen (vgl. 2.4.3 Zeitpunkt der Einspritzung, S.20), würde dies die mögliche Einspritzzeit und damit die maximal mögliche Kraftstoffmenge reduzieren.

Ansatz 2: Software-Lösung durch zusätzliche GPIOs

Hierbei werden alle vier Treiber separat durch jeweils einen eigenen GPIO-Pin angesteuert. Hierfür werden allerdings zusätzlich zu den zwei schon bestehenden Pins zwei weitere GPIO-Pins benötigt. (vgl. Abbildung 6.3)

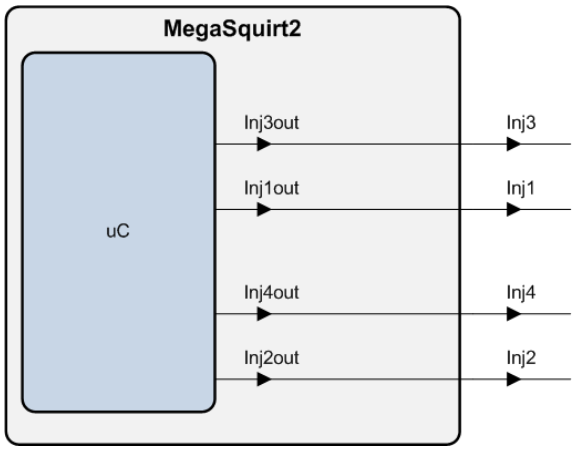

Abbildung 6.3.: Struktur des Software-Lösungsansatzes

Diese Lösung bietet zudem theoretisch die Möglichkeit, für jeden Zylinder eine individuelle Konfiguration beziehungsweise ein individuelles Kennfeld zu nutzen. Diese Option wird im Hawks-Projekt zur Zeit nicht genutzt, ist aber im Profirennsport durchaus üblich um minimale Fertigungstoleranzen oder unterschiedliche Verschleißerscheinungen auszugleichen.

6.2.2. Entscheidung für die Software-Lösung

Aufgrund der im Hawks-Projekt verwendeten Einspritzungsart wird im Rahmen dieser Arbeit die Software-Lösung implementiert. Hauptgrund hierfür ist die Nichtbeeinflussung des Zeitfensters für die Einspritzung, sodass hier keine Einschränkung der maximal möglichen Kraftstoffmenge geschieht.
Durch Wegfallen der Peak&Hold-Schaltung die können die bisherigen PWM-Pins als zusätzliche GPIOs genutzt werden können.(vgl. A.1 Pin Belegung HC9S12C64, S.113).

6.3. Telemetriekonzept

Das Senden der Telemetriedaten geschieht standardmäßig über die RS232-Schnittstelle (vgl. 5.4.1 Lesen der Laufzeit-Variablen, S.60). Das Telemetriekonzept des Gesamtfahrzeugs beruht jedoch auf einem TTCAN-Bus. Folglich muss hier die ECU angepasst werden. Dabei sind folgende Anforderungen zu erfüllen:

- Das Datenformat muss mit der Walbro überstimmen
- Die Daten werden über einen TimeTriggered CAN-Bus gesendet

6.3.1. Lösungsansätze für das Senden der Telemetriedaten

Um diese Anforderungen zu erfüllen, gibt es zwei Möglichkeiten, die jeweils eine unterschiedliche Schnittstelle der MegaSquirt nutzen. Zum einen die RS232-Schnittstelle und zum anderen den CAN-Bus. Die Ansätze werden hier diskutiert.

Ansatz 1: Senden via RS232

Es wird standardmäßig das von der MegaSquirt genutzte Datenformat über die RS232-Schnittstelle an einen externen Mikrocontroller gesendet. Dieser Controller wird als „ECU-Gateway“ bezeichnet und übernimmt das Senden der Telemetriedaten auf den TTCAN-Bus.
Diese Lösung wäre analog zum bestehenden System mit der Walbro. (vgl. 4.2.3 ECU-Gateway, S.43)

Das Umrechnen in das Format der Walbro würde hierbei ebenfalls das ECU-Gateway übernehmen.

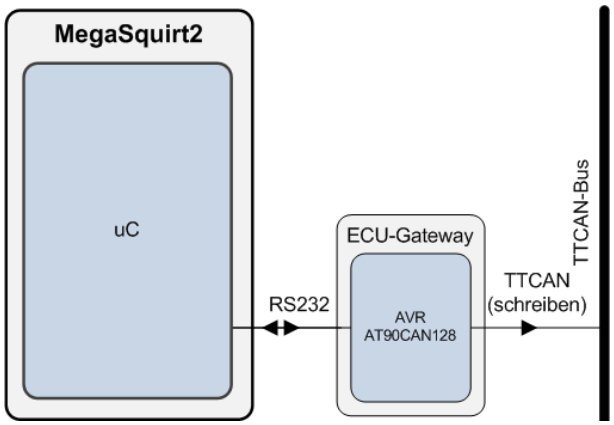

Abbildung 6.4.: Telemetriekonzept via RS232

Der Vorteil dieses Ansatzes liegt in der Abwärtskompatibilität zur momentan verwendeten Lösung mit einem ECU-Gateway für die Walbro-ECU. Es kann dieselbe Hardware-Basis des schon vorhandenen ECU-Gateways genutzt werden, lediglich die Software muss angepasst werden. (vgl. 4.2.3 ECU-Gateway, S.43)
Zudem ist der Ansatz überschaubar, da hier die Problematik, zwei zeitkritische Aufgaben auf einem Mikrocontroller zu verknüpfen, umgangen wird. Die Kommunikation via TTCAN wäre hierbei von der Steuerung des Verbrennungsmotors getrennt und so würden sich beide Aufgaben nicht gegenseitig behindern.

Ansatz 2: Senden über die CAN-Schnittstelle

Die Alternative zum Senden über die serielle Schnittstelle wäre die direkte Anbindung an den TTCAN-Bus. Hierfür muss die MegaSquirt softwareseitig sowohl das Umformatieren der Telemetriedaten als auch das Synchronisieren mit dem Timemaster (TM) des TTCAN-Bus implementieren.

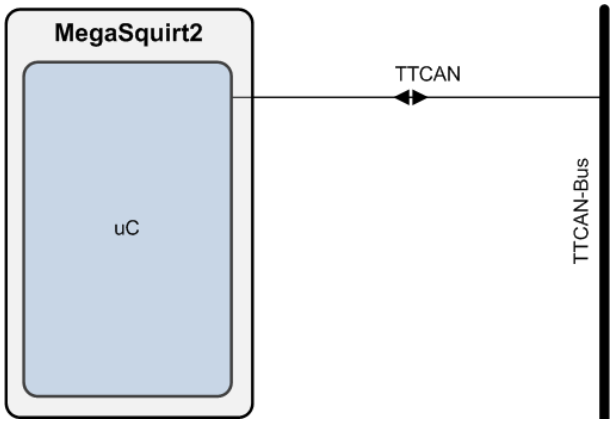

Abbildung 6.5.: Telemetriekonzept via CAN

Zum Umsetzen des TTCAN-Bus sind folgende Punkte nötig:

- **Empfangen der TM-Referenznachrichten**

  Es müssen die Nachrichten des TM empfangen werden und ihre Empfangszeiten gespeichert werden.

- **Zeitmessung**

  Es muss die Zeit zwischen den Timemaster-Referenznachrichten ermittelt werden. Da der Timer des Mikrocontrollers der MegaSquirt2 mit einer Frequenz von $1.5\mathrm{MHz}$ läuft und dies deutlich über der Taktrate des in der Hawks-Telemetrie verwendeten Scheduler-Takts von $5\mathrm{kHz}$ liegt, ist es möglich, den Timer-Stand in der CAN-Receive-ISR abzufragen und mit der Sollzeit zu vergleichen. Dementsprechend muss der Compare-Wert des Timers zum Senden von Daten korrigiert werden.

- **Zeit bis zum Timeslot warten**

  Es müssen die konkreten Start- und Endzeiten des eigenen Timeslots ermittelt werden. Da die Position im TTCAN-Matrixzyklus bekannt ist, muss ein Timer die Anzahl vorhergehender Slots abwarten und löst zum richtigen Zeitpunkt eine ISR zum Senden der CAN-Daten aus.

- **CAN-Message vorbereiten**

  Die zu sendende Nachricht sollte vor dem eigentlichen Sendezeitpunkt schon vorbereitet sein, damit die ISR zum Senden möglichst kurz gehalten werden kann. Dies kann in der Super Loop geschehen.

Dieser Ansatz braucht zwar keinen zusätzlichen Mikrocontroller, wie er in Ansatz 1: Senden via RS232 benötigt wird. Allerdings wäre er nicht kompatibel zum jetzigen Telemetriesystem, da beispielsweise in den jetzigen Hawks-Fahrzeugen das ECU-Gateway und die ECU nicht am selben Ort verbaut sind und die vorhandene Verkabelung keinen Anschluss der ECU an den TTCAN-Bus vorsieht.
Somit müsste hier die Verkabelung modifiziert werden um die MegaSquirt im Fahrzeug zu testen.

6.3.2. Entscheidung für das Senden via RS232

Um die Kompatibilität zur Walbro zu erhalten und um das System übersichtlich zu gestalten, wird sich für das Senden via RS232 entschieden. Ohne preemptive Interrupts und ohne einen globalen Sheduler sind der TTCAN-Bus und die Funktion der ECU nicht kollisionsfrei auf einem System zu implementieren.

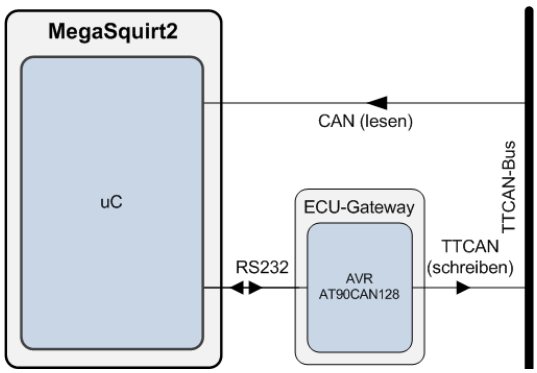

Abbildung 6.6.: Telemetriekonzept für die MegaSquirt

6.4. Konzept der dynamischen Konfiguration

Seitens der MegaSquirt stehen zwei Schnittstellen für die Kommunikation zur Verfügung: Die RS232-Schnittstelle und der CAN-Bus. Werkseitig wird der CAN-Bus nicht genutzt. Die Hardware ist allerdings soweit darauf vorbereitet, das nur zwei Signalleitungen mit entsprechenden Ausgangspins verbunden werden müssen, um den CAN-Controller in Betrieb zu nehmen. Daher bietet sich zum Empfangen von Steuerbefehlen und Konfigurationsdaten zur Laufzeit die CAN-Schnittstelle an.
Das reine Lesen von einem TTCAN-Bus ist zeitunkritisch, da hier keine Synchronisierung mit dem TimeMaster nötig ist.

6.4.1. Konzeption der Kommandos

In einem CAN-Paket sind 8 Daten-Bytes verfügbar. Das Erste davon ist im Telemetrie-System im Hawks-Projekt für TTCAN-Protokolldaten reserviert. Somit bleiben 7 Bytes für Nutzdaten (vgl. 4.2.1 TTCAN, S.41).
Um Timeslots im TTCAN-Matrixzyklus zu sparen und dennoch mit einer hohen Frequenz von Nachrichten arbeiten zu können, sollten alle nötigen Daten in einem Paket übertragen werden und dieses in jedem Basiszyklus gesendet werden.

Es gibt zwei Arten von Konfigurationsparametern: Steuerbits und Steuerdaten. Steuerbits sind Kommandos, die in einem einzelnen Bit kodiert sind. Auf diesem Bit basiert eine binäre Entscheidung innerhalb der ECU. Somit hat das Steuerbit direkten Einfluss auf deren Funktion.
Steuerdaten hingegen werden seitens der ECU direkt zur Beeinflussung von Aktoren genutzt und mit in deren Berechnung einbezogen indem sie als dezimale Werte interpretiert werden.

Im folgenden werden die Möglichkeiten der externen Konfiguration einer ECU erläutert.

Steuerbits

- **Zündunterbrechung (Ignition Cut)**

  Die Zündung ist zu unterbrechen, solange dieses Flag gesetzt ist. Dies bedeutet, dass keine Zündspule mehr geladen werden darf. Befindet sich eine Spule bereits im Ladezustand wenn das Flag gesetzt wird, so wird diese noch gezündet. Dies dient dazu, einen unerwünschten Zündfunken zu einem unkontrollierten Zeitpunkt zu vermeiden. Würde das Laden der Spule unterbrochen und die gesammelte Zündenergie bereits ausreichen um einen Funken zu bilden, so könnte ein Zündfunke entstehen. Im ungünstigsten Fall befände sich der betroffene Zylinder dabei im Verdichtungstakt und würde eine Verbrennung des Gemischs initiieren. Dies würde den Motor stark belasten.

- **Kraftstoffunterbrechung (Fuel Cut)**

  Die Einspritzung ist zu unterbrechen, solange dieses Flag gesetzt ist. Äquivalent zur Zündunterbrechung sollte hierbei ein bereits laufender Einspritzvorgang nicht unterbrochen werden, um kein unkontrolliertes Gemisch zu erzeugen. Sobald die Einspritzung unterbrochen würde, wäre die Kraftstoffmenge zu gering für die Luftmasse und ein sehr mageres Gemisch wäre die Folge.

Im schlimmsten Falle wäre dieses Gemisch nicht mehr zündfähig und würde später in der Abgasanlage verbrennen.

- **Kennfeldwechsel (Change Map)**

  Mit diesem Flag werden die Kennfelder für Kraftstoff und Zündwinkel gewechselt. Das zweite Kennfeld kann für einen Kraftstoffsparbetrieb oder auch für einen leistungsstärkeren Betrieb ausgelegt sein. Das Kennfeld kann zu jeder Zeit gewechselt werden, da es keinen Einfluss auf bereits laufende Ereignisse hat, sondern sich nur auf neu beginnende Ereignisse auswirkt.

- **Soft-Notaus**

  Durch dieses Flag werden alle Aktoren der ECU abgeschaltet. Diese Möglichkeit dient dazu, den Motor im Gefahrenfall abzuschalten ohne den manuellen Notaus betätigen zu müssen. Der Vorteil liegt hierbei darin, dass das Telemetriesystem und damit die Datenaufzeichnung weiterläuft und somit im Nachhinein die Fehlersuche erleichtert wird.

Steuerdaten

- **Zündwinkelanpassung, relativ**

  Der übertragene Wert wird vom errechneten Zündwinkel abgezogen um ihn so „Richtung Spät“ zu verstellen und so die Motorleistung zu reduzieren. Eine Verstellung „Richtung früh“ ist nicht sinnvoll, da der errechnete Zündwinkel üblicherweise möglichst nah an der Klopfgrenze liegt bzw liegen sollte (vgl. 2.5.1 Zündwinkel, S.24).
  Ein sinnvoller Wertebereich wäre hier von 0-25°KW, was fast dem kompletten Wertebereich des normalen Kennfelds entspricht. Dieser Wert ließe sich in 8bit mit einer Auflösung von 0.1°KW/bit kodieren.

- **Kraftstoffmenge anpassen, relativ**

  Ein übertragener Wert wird als Prozent-Wert interpretiert um die eigentliche Kraftstoffmenge um den gegebenen Wert zu reduzieren. Bei der Kraftstoffmenge sollte der Wert sowohl nach oben als auch nach unten anpassbar sein. Daher wird der übertragene Wert als signed interpretiert. Bei $8\mathrm{bit}$ bleibt so ein Wertebereich von -128% bis +127%. In wieweit dies sinnvoll ist, muss die Praxis zeigen, denn eine Reduzierung des Kraftstoffs um 100% oder mehr kommt einer Unterbrechung der Einspritzung gleich (vgl. 6.4.1 Steuerbits, S.74).

- **Leistungsreduzierung, relativ**

  Die Leistung des Motors soll um einen gegebenen Prozentsatz reduziert werden. Auf Basis dieses Werts werden Zündwinkel und Kraftstoffmenge angepasst um den gewünschten Effekt zu erzielen. Die Kennlinien hierfür liegen auf der ECU. Hier kann davon ausgegangen werden, dass nur eine Leistungsreduzierung von außen sinnvoll ist. Schließlich soll der Motor im Regelfall immer die maximale Leistung liefern.
  Als Wertebereich würden sich hier 0-100% anbieten, was in 8bit mit einer Auflösung von 0.5%/bit kodierbar ist.

- **Drehzahlbegrenzer**

  Die Maximale Drehzahl wird auf einen gegebenen Wert festgesetzt. Ist hier ein Wert eingetragen, so darf diese Drehzahl nicht überschritten werden. Ist der Wert 0, so ist der Begrenzer deaktiviert.
  Der Wertebereich sollte das volle Spektrum der Drehzahl umfassen, von der Anlasserdrehzahl bis zum theoretischen Maximum. Da hier keine exakte Regelung des Wertes erfolgt, kann die Auflösung des Wertes gröber sein. Auch hier bietet sich ein 8bit-Wert an. Mit einer Auflösung von $65\frac{\mathrm{U}}{\mathrm{min}}$/bit wäre der Regelbereich von $0 - 16500\frac{\mathrm{U}}{\mathrm{min}}$ möglich.

Auswahl für eine Testapplikation

Im Hawks-Umfeld ist es vorrangig nötig, die Leistung des Motors im Betrieb abhängig von der jeweiligen Fahrsituation zu drosseln. Eine Erhöhung der Leistung ist nicht sinnvoll, da der Motor von seiner Basisabstimmung her immer für ein Maximum an Leistung ausgelegt ist und somit kaum Spielraum für Mehrleistung seitens der ECU besteht.
Daher werden im Rahmen dieser Arbeit folgende der in 6.4 Konzept der dynamischen Konfiguration genannten Optionen implementiert, da diese dafür genutzt werden können die Leistung des Motors zu reduzieren:

- Zündunterbrechung
- Kraftstoffunterbrechung
- Kennfeldwechsel
- Anpassung des Zündwinkels

6.4.2. Sicherheitskonzepte

Um die Anforderung in Hinblick auf Zuverlässigkeit und Schadensfreiheit (vgl 3.2.4 Schadensfreiheit, S.36) zu erfüllen, sind folgende Sicherheitskonzepte im Bezug auf die externe Konfiguration nötig:

Bus-Timeout

Sollte der CAN-Bus ausfallen, muss die ECU autark weiterarbeiten, wie sie es ohne Bus täte. Sobald wieder reguläre Nachrichten empfangen werden, sollen die empfangenen Parameter wieder verwendet werden.

Erkennen von Übertragungsfehlern

Auf eine Checksumme oder ähnliche Mechanismen, die die Korrektheit der empfangenen Nachricht überprüfen, wird verzichtet, da der CAN-Bus dies in Form einer 16bit CRC-Prüfsumme bereits ausreichend implementiert.

Prüfung der Parameter

Um sicherzustellen, dass der Einflusss den die übermittelten Parameter auf die Aktoren haben zulässig ist, wird der errechnete Wert mit den jeweiligen Minima und Maxima aus den Kennfeldern verglichen. Diese dürfen nicht über- bzw unterschritten werden.

7. Realisierung

In diesem Kapitel werden die Umsetzungen der im Kapitel 66 Konzeption diskutierten Lösungsansätze beschrieben.
Hierzu wird zuerst die Testumgebung vorgestellt um einen Gesamtüberblick zu vermitteln.

Anschließend wird danach die Umsetzung der einzelnen Punkte in folgender Reihenfolge erörtert:

- Einspritzung
- Zündung
- Dynamische Konfiguration
- Telemetriekonzept

7.1. Testumgebung

Um die ECU zu testen und die Modifikationen durchzuführen wird die folgende Testumgebung aufgebaut. Diese umfasst neben der ECU, dem MegaStim und der nötigen Verkabelung folgende Teile:

- RPM-Stim Simulation der Signale für Kurbel- und Nockenwelle.
- Adapter-Board Hardware-Adapter für Messpunkte und die Möglichkeit unterschiedliche Hardware-Quellen für Stimuli zu wählen.

Einen Überblick über die Struktur des Aufbaus gibt Abbildung 7.1 auf Seite 79.

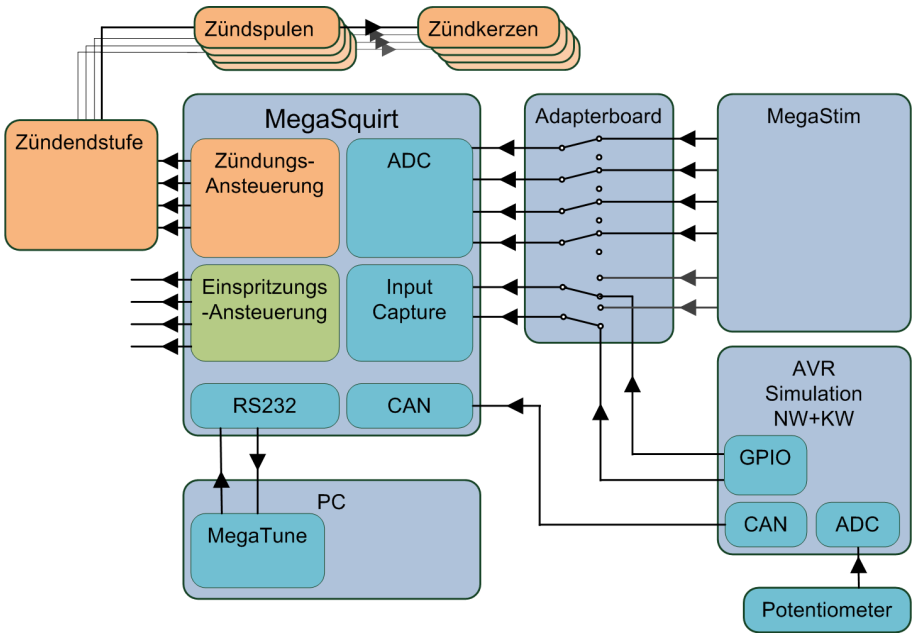

Abbildung 7.1.: Struktur des Testaufbaus

7.1.1. RPM-Simulation

Das MegaStim-Stimuli-Board simuliert für die Drehzahl lediglich Pulse eines Sensors ohne fehlenden Zahn. Es ist nicht möglich dies anzupassen oder auf einfache Weise zu modifizieren. Daher wurde eine eigene Simulation der Signale von Kurbel- und Nockenwelle auf Basis eines AVR AT90CAN128 Mikrocontrollers in C entwickelt.

Der Mikrocontroller simuliert die Signale durch Pulse. Die gewünschte Drehzahl wird über ein Potentiometer stufenlos zwischen 350 und 16000 $\frac{\mathrm{U}}{\mathrm{min}}$ geregelt.

Funktionsweise

Es wird eine konfigurierbare Anzahl Events pro Umdrehung simuliert. Ein Event kann hierbei eine steigende oder eine fallende Flanke eines simulierten „Zahns“ des Kurbelwellen-Inkrementalgebers sein. Der zeitliche Abstand dieser Events wird über einen Timer geregelt, dessen Compare-Wert abhängig vom Wert des ADCs errechnet wird.
Die einzige Einschränkung bei der Konfiguration ist der Umstand, dass ein „Zahn“

genauso breit ist wie die darauf folgende „Lücke“. Dies ist aber unerheblich, da die MegaSquirt die Flanken erkennt und nicht auf die Dauer der Pulse reagiert.
Die Winkel für Beginn und Ende des Pulses der Nockenwelle können ebenfalls konfiguriert werden. Es wird das nächst frühere „Event“ ermittelt und der dazugehörige Offset über einen zusätzlichen Timer abgewartet.
Alle Parameter sind durch defines im Quellcode definierbar.

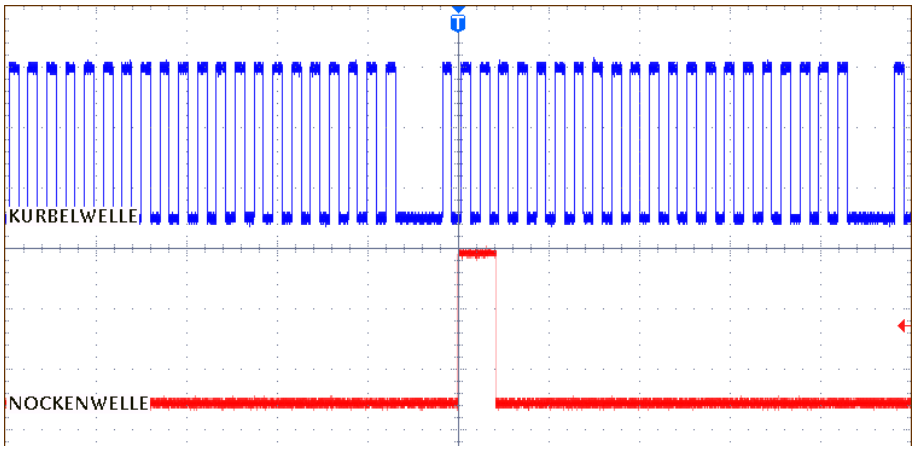

Abbildung 7.2.: Signale der Drehzahl-Simulation

In Abbildung 7.2 sind die so erzeugten Signale zu sehen. Das Kurbelwellen-Signal ist blau, das Nockenwellen-Signal rot dargestellt.

7.1.2. Adapterboard

Da das MegaStim-Board nicht alle benötigten Signal-Arten simulieren kann, werden weitere Stimuliquellen benötigt. Ein Beispiel hierfür ist die RPM-Simulation (siehe 7.1.1 RPM-Simulation). Folglich bedarf es einer Möglichkeit, die Signalquelle wählen zu können. Hierfür wurde das Adapterboard entwickelt (vgl. 5.2.4 MegaStim - Stimuliboard, S.52).
Das Adapterboard ist eine passive Platine, die zwischen MegaSquirt und MegaStim geschaltet wird. Sie ermöglich es, die Signalleitungen der Sensorik über DIP-Switches umschalten zu können. So kann für jedes Signal separat gewählt werden, aus welcher Quelle es stammen soll. (vgl. Abbildung 7.1 auf Seite 79)

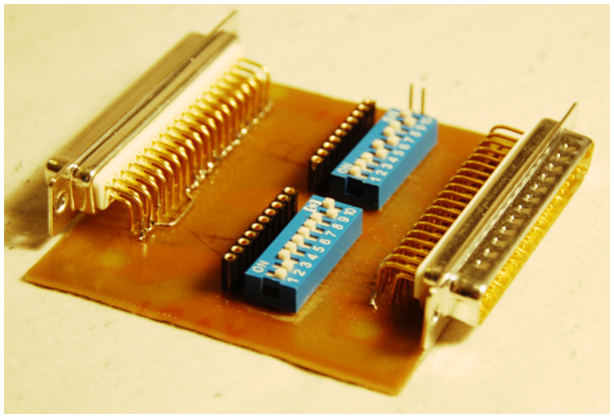

Abbildung 7.3.: Adapterboard

Das dazugehörige Eagle-Layout befindet sich im elektronischen Anhang C, der Schaltplan in Anhang B.2.3.

7.2. Realisierung der vollsequentiellen Zündung

Zur Umsetzung der vollsequentiellen Zündung wurde die Anleitung von msextra.com [Murrey u. a., 2008] umgesetzt. Hierzu muss eine Zündendstufe aufgebaut und die MegaSquirt entsprechend konfiguriert werden.

7.2.1. Zündendstufe

Die Zündendstufe wird als Testaufbau auf einer Lochrasterplatine entworfen und getestet. Hierzu wurden die seitens der MegaSquirt empfohlenen Zündtreiber vom Typ VB921ZVFI verwendet. Diese werden allerdings nicht mehr hergestellt, sodass die Beschaffung sehr zeit- und kostenintnesiv ist. Im Rahmen dieser Arbeit wird mit diesem Treiber getestet. Zukünftig sollte jedoch ein anderer Treibertyp gewählt werden.

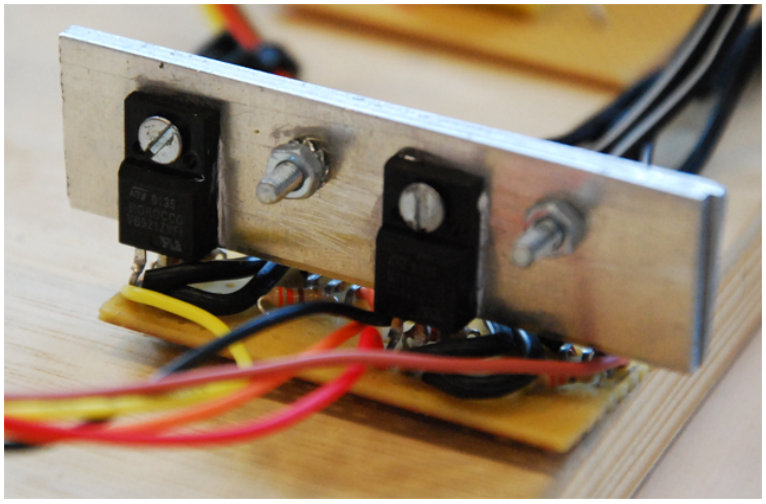

Abbildung 7.4.: Zündtreiberstufe auf Basis von VB921ZVFI Treibern

7.2.2. Konfiguration der ECU

Damit die ECU die vollsequentielle Zündung umsetzt, müssen die in Tabelle 7.1 aufgelisteten Einstellungen in MegaTune vorgenommen werden. Alle weiteren Einstellungen wurden auf ihren Standardeinstellungen belassen.

Option	**Menü**	**Einstellung**
Spark Output	Ignition Options	Going High (Inverted)
Number of Coils	Ignition Options	Coil on Plug
Spark A Output Pin	Ignition Options	D14
Fixed Advance	More Ignition Options	Use Table
Dwell Type	More Ignition Options	Standard Dwell

Tabelle 7.1.: Konfiguration der Zündung in MegaTune

Diese Einstellungen bewirken, dass die GPIO-Pins der Onboard-Status-LEDs für die Ansteuerung der Treiber genutzt werden. Es ergibt sich die in Tabelle 7.2, S. 83 dargestellte Belegung der GPIOs:

GPIO-Port	LED	Zündungskanal
PM3	Injection LED (D14)	Spark A
PM4	Warm Up LED (D15)	Spark B
PM5	Acceleration LED (D16)	Spark C
PT5	—	Spark D

Tabelle 7.2.: Belegung GPIO-Pins und Zündkanäle

7.2.3. Testen der Zündung

Um die Funktion der Zündung sicherzustellen wurde sie auf Zuverlässigkeit und Genauigkeit getestet. Die Genauigkeit wurde durch Kontrolle des tatsächlichen Zündwinkels überprüft. Die Zuverlässigkeit wird durch eine Messung der Temperatur der Treiber unter Last getestet.

Überprüfung des Zündwinkels

Mittels eines Speicheroszilloskops wird gemessen, ob der von MegaTune angezeigte Zündwinkel auch dem Tatsächlichem enspricht. Als Referenz dient hierbei das Kurbelwellen-Signal der RPM-Simulation. Der Winkel lässt sich über die Zeit in Bezug auf die Motordrehzahl berechnen.
Dieses Vorgehen wird hier erläutert und exemplarisch an einer Messung verdeutlicht.

Die RPM-Simulation wird auf eine feste Drehzahl eingestellt, ebenso der Drosselklappenwinkel. Dies ist notwendig, da das Zündwinkel-Kennfeld auf diesen Größen basiert und der Wert während der Messung möglichst stabil sein sollte. Zudem muss der Puls der Kurbelwelle, den die ECU als oberen Totpunkt interpretiert, bekannt sein. Da die MegaSquirt auf einen Versatz von $0°\mathrm{KW}$ konfiguriert wurde, ist dies die steigende Flanke des ersten Zahns des Kurbelwellengebers.
Mit Hilfe des Oszilloskops werden nun das Kurbelwellensignal und der erste Zündkanal dargestellt.
Sollte ein anderer Zündkanal gewählt werden, so muss dessen Versatz zum OT im Rahmen der Zündreihenfolge mit berücksichtigt werden. Nun wird die Zeit zwischen der steigenden Flanke des ersten Zahns (OT) und der fallenden Flanke des Zündkanals gemessen. Die gemessene Zeit wird dann in °KW umgerechnet und mit dem von MegaTune angezeigten Wert verglichen.

Als Beispiel wird eine Drehzahl von 1000 $\frac{\mathrm{U}}{\mathrm{min}}$ und eine Drosselklappenstellung von 1% angenommen. Dies ergibt nach dem in Abbildung 7.5 gezeigten Kennfeld einen Zündwinkel von $14.6°\mathrm{KW}$.

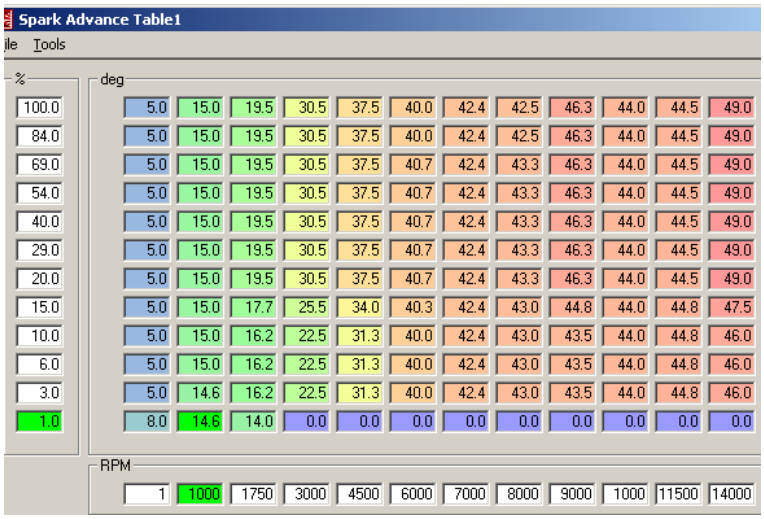

Abbildung 7.5.: Das aktive Kennfeld während der Messung

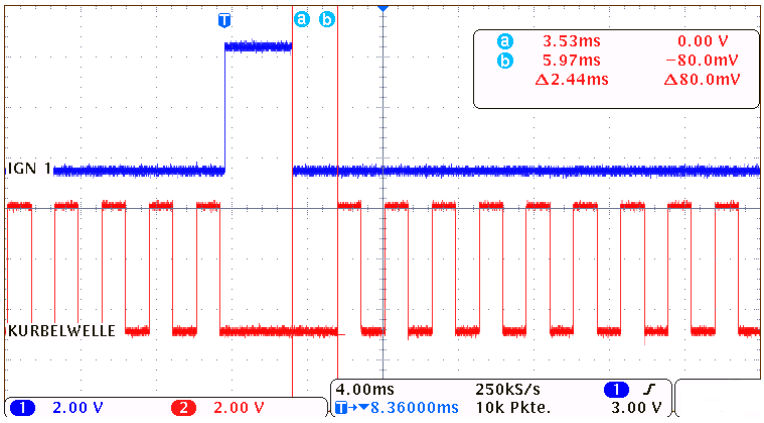

Abbildung 7.6.: Messung des Zündwinkels bei 1000U/min

Die in Abbildung 7.6 gezeigte Messung ergibt eine Zeit von $t = 2.44\mathrm{ms}$. Dies entspricht einer Drehzahl von 1000 $\frac{\mathrm{U}}{\mathrm{min}}$ folgendem Zündwinkel ZW:

$$ZW = \frac{t * \frac{\mathrm{U}}{\mathrm{min}} * 360°}{60\mathrm{s}}$$

$$ZW = \frac{2.44\text{ms} * 1000\frac{\text{U}}{\text{min}} * 360°}{60\text{s}}$$
$$ZW = \underline{14.64°\text{KW}}$$

Die Abweichung liegt also im Bereich von unter $0.1°\text{KW}$ und erfüllt damit die an die Genauigkeit gestellte Anforderung. (vgl. 3.2.3 Genauigkeit, S. 35).

Lasttest

Um die Funktionalität der Zündung unter Last zu gewährleisten wird die Temperatur der Zündtreiber bei maximaler Drehzahl über einen Zeitraum von 10 Minuten gemessen. Dieser Zeitraum hat sich als ausreichend erwiesen, da hier kein Temperaturanstieg mehr zu verzeichnen war.

Die Zündtreiber erreichten so eine maximale Temperatur von $80°\text{C}$. Diese liegt deutlich unter der im Datenblatt angegebenen Höchsttemperatur von $150°\text{C}$. Die Tests wurden allerdings bei Raumtemperatur durchgeführt. Im Fahrzeug wird die Temperatur durch die Abwärme des Motors deutlich steigen. Abhilfe sollte ein vergrößerter Kühlkörper schaffen. Für die im Rahmen dieser Arbeit durchgeführten Tests und Messungen ist die Kühlung aber ausreichend.

7.2.4. Probleme der Zündung

Sobald eine oder mehr Zündspulen angeschlossen sind, springt die Drehzahlanzeige der MegaSquirt unkontrolliert in unregelmäßigen Abständen. Als Konsequenz daraus laufen Zündung und Einspritzung ebenso unregelmäßig wie unkontrolliert. Messungen der Spannungsversorgung der ECU ergaben, dass die Zündung starke Spannungsschwankungen in der Spannungsversorgung erzeugten. Das Ergebnis der Messung mittels Oszilloskop ist in Abbildung 7.7 auf der folgenden Seite dargestellt. Es wird zwischen der Masse der Spannungsversorgung der ECU und der Masse des Motorblocks gemessen. Folglich sollte die gemessene Spannung keine bis minimale Schwankungen aufweisen:

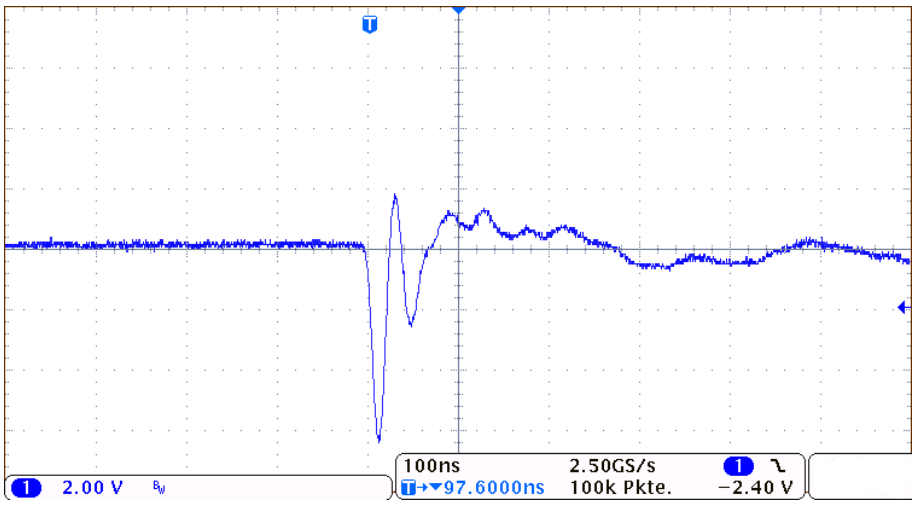

Abbildung 7.7.: Messung zwischen Masse der ECU und Masse des Motorblocks

Die Messung ergibt allerdings, dass Schwankungen von bis zu $6\mathrm{V}$ auf der Masseleitung anliegen. Welche Folgen diese Schwankungen haben wird nun näher analysiert.

Analyse

Da die gemessene Drehzahl springt und somit scheinbar der Auslöser der Problematik ist, werden die simulierten Signale von Kurbel- und Nockenwelle kontrolliert. Um die mögliche Ursache noch weiter einzugrenzen wird die ECU für einen Wasted Spark-Betrieb und eine halbsequentielle Einspritzung konfiguriert, um zu überprüfen, ob die Problematik ohne ein Nockenwellensignal auch vorhanden ist.
Im halbsequentiellen Betrieb blieb die Störung aus. Folglich liegt durch die Zündung eine Störung des Nockenwellensignals vor. Die Messung des Signals ist in Abbildung 7.8 rot zu sehen. Zum Vergleich ist die zuvor in Abbildung 7.7 gemessene Spannungsschwankung hier nochmals blau dargestellt.

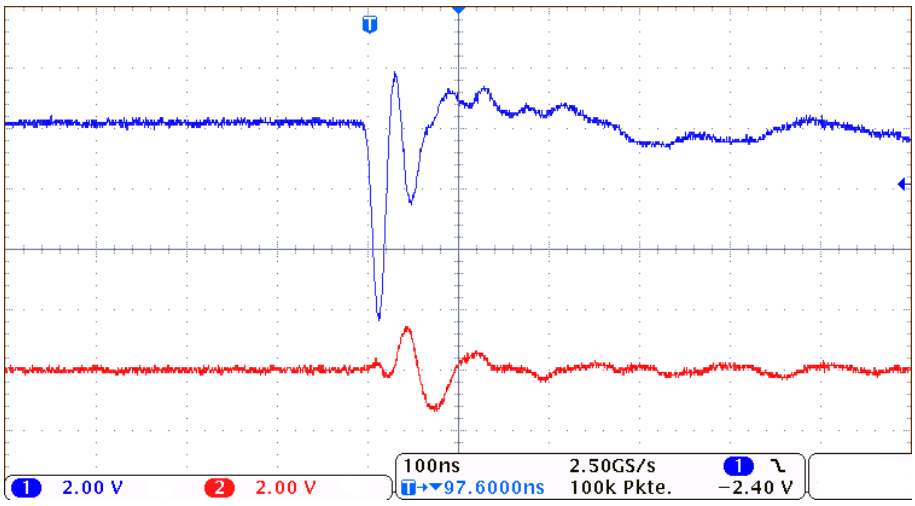

Abbildung 7.8.: Kontrolle des Nockenwellen-Signals

Es liegt die Vermutung nahe, dass diese Pegelschwankung des Nockenwellensignals die Input Capture-Interrupt Service Routine der Nockenwelle anstößt und somit der ECU einen Nockenwellenpuls zu einem völlig falschen Zeitpunkt vermittelt. Die ECU erkennt daraufhin eine entsprechend falsche Drehzahl. Da diese Störungen scheinbar nicht regelmäßig eine ISR auslösen, ist die Drehzahl dementsprechend ungleichmäßig.
Eine weitere Messung bestätigt diese Vermutung. Hierzu wurde ein GPIO-Pin angesprochen, solange die Nockenwellen-ISR aktiv ist. In Abbildung 7.9 ist das Kurbelwellensignal blau dargestellt und das Signal der ISR in rot.

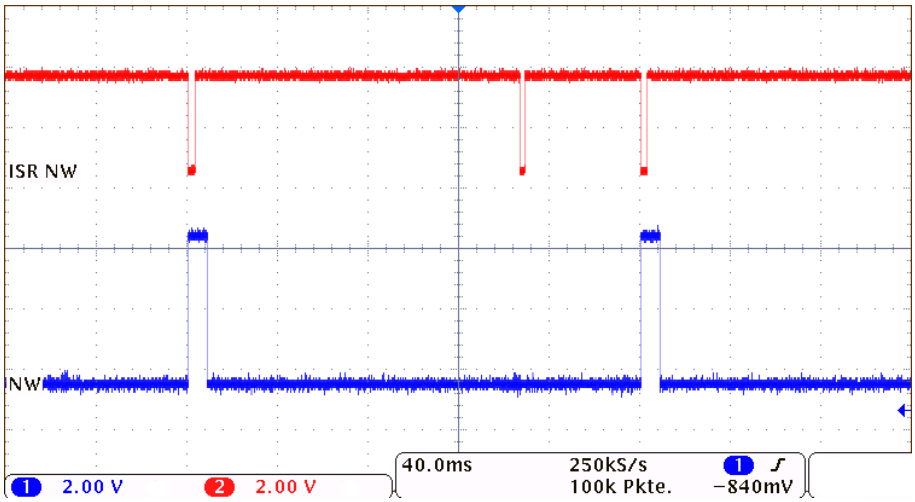

Abbildung 7.9.: Nockenwellensignal und Nockenwellen-ISR

Die Störungen des Nockenwellensignals sind hier nicht sichtbar, da die zeitliche Auflösung deutlich gröber ist. Dennoch ist deutlich erkennbar, dass die ISR zwischen den beiden regulären Nockenwellenpulsen ein weiteres Mal abgearbeitet wird, obwohl kein regulärer Puls vorliegt. Eine Messung mit feinerer Zeitauflösung bestätigte die Vermutung, dass hier die Störung der Zündung auf dem Nockenwellensignal vorliegt und die ISR anstößt.

Lösungsansätze

Zur Beseitigung dieser Störungen wurden folgende Lösungsansätze erarbeitet und umgesetzt:

- **RC-Glied (Tiefpass)**

  Da die Störung eine sehr hochfrequente Schwingung ist und ihre Frequenz um einige Größenordnungen höher liegt als die des Nockenwellensignals, wird ein passiver Tiefpass erster Ordnung aufgebaut. Dieser wurde in das Signal der Nockenwelle geschaltet.
  Die Grenzfrequenz wurde mittels der maximalen Drehzahl von 16000 $\frac{\mathrm{U}}{\mathrm{min}}$ und der Anzahl Pulse des Kurbelwellensensors auf $7\mathrm{kHz}$ festgelegt.
  Die Störung hat eine Frequenz von ungefähr $20\mathrm{MHz}$, liegt also um den Faktor 3000 höher als die Grenzfrequenz. Das Störsignal sollte also um mehr als $40\mathrm{dB}$ gedämpft werden. Dies entspricht einem Prozentwert von über 99%.
  Dennoch ließ sich das Verhalten durch diese Maßnahme nicht beseitigen.

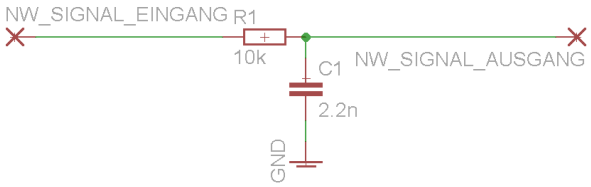

Abbildung 7.10.: Schaltplan für einen Tiefpass 1. Ordnung

  Da dieser Ansatz nicht die Ursache der Störung beseitigt, sondern nur seine Wirkung an einem Punkt dämpft, liegt der Schluss nahe, dass die Störung weitere Auswirkungen auf das System hat.

- **Freilaufdioden**

  Freilaufdioden sind schnell schaltende Dioden, die umgekehrt zur eigentlichen Stromrichtung geschaltet werden. So liegt ihre Sperrrichtung entgegen der Stromrichtung und sie sperren Spannungen, die entgegen der gewollten Richtung anliegen.
  Es wurden Freilaufdioden parallel zu den Zündspulen geschaltet.

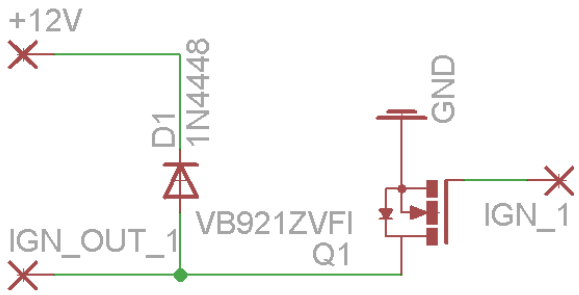

Abbildung 7.11.: Schaltplan für eine Freilaufdiode über einen Zündtreiber

  Dies sollte den negativen Teil der Störung filtern und somit auch das sichtbare Schwingen der Störung dämpfen. Es wurden Dioden vom Typ 1N4448 verwendet, die mit einer RecoveryTime von 2ns schnell genug sind um die Störung zu unterdrücken.
  Hier stellte sich keine messbare Verbesserung ein.

Leider brachte keine der umgesetzten Lösungsansätze einen überzeugenden Erfolg, sodass aus Zeitgründen die Entwicklung hier eingestellt werden musste, da der Schwerpunkt dieser Arbeit nicht auf elektrotechnischen Aspekten liegt. Es wurde daher ohne angeschlossene Zündung weitergearbeitet.

7.3. Umsetzung der vollsequentiellen Einspritzung

Entsprechend der Konzeption der Einspritzung (vgl. 6.1.2 Entscheidung für die vollsequentielle Zündung, S.67) wird eine Erweiterung der Einspritzung vorgenommen. Hierfür muss ebenfalls eine eigene Treiberstufe entwickelt werden. Zu deren Ansteuerung wird die auf Seite 69 vorgestellte Lösung umgesetzt.

7.3.1. Treiberstufe der Einspritzung

Der Treiberbaustein vom Typ IXDI404PI, der auf der MegaSquirt als Treiber für die Einspritzung genutzt wird, war nicht zu beschaffen. Daher wurde ein Ersatztyp mit ähnlichen elektrischen Eigenschaften gewählt. Die Wahl fiel dabei auf einen MosFET vom Typ IRFP240, da dieser mit bis zu 20A ausreichend starke Ströme schalten kann und daher den Anforderungen auf jeden Fall gewachsen ist.
Der vollständige Schaltplan der Treiberstufe ist in Abbildung 7.12 dargestellt. Dabei die MosFET (Q1-Q4 im Schaltplan) sicher angesteuert werden, wird jeweils ein weiterer Schalttransistor (T1-T4) verwendet damit derentsprechende Pin des Mikrocontrollers nicht überlastet wird. Als zusätzlicher Schutz für den Mikrocontroller wird jeweils eine Zehner-Diode (Z1-Z4) gegen Masse geschaltet um im Fall eines defekten MosFETs Spannungen in Richtung Controller abzuleiten.

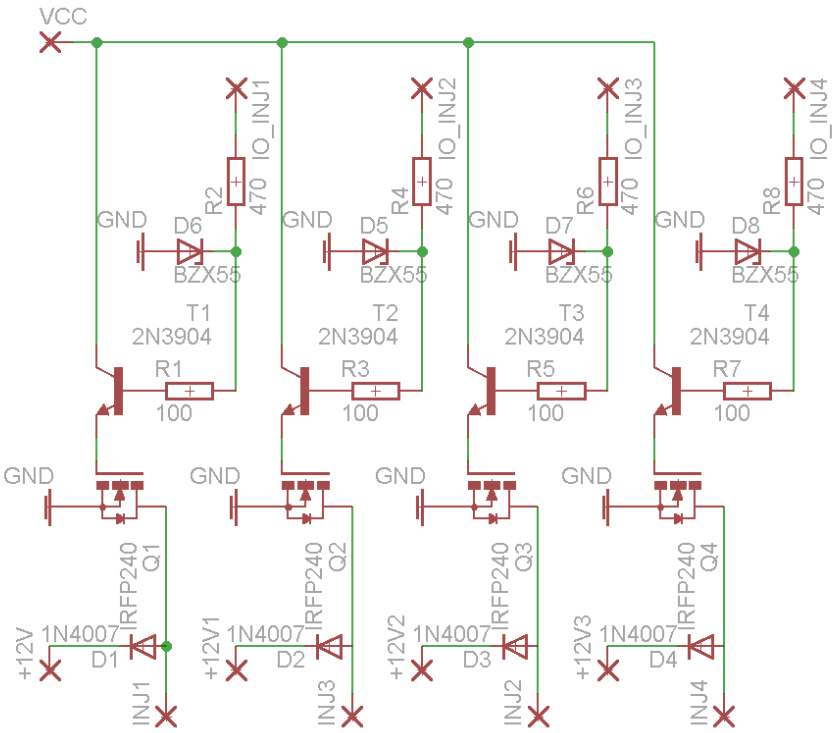

Abbildung 7.12.: Schaltplan für die Einspritzungs-Treiberstufe

Aus Zeitgründen konnte die entworfene Treiberstufe nicht aufgebaut werden, da die Problematik mit der Zündung (vgl. 7.2.4 Probleme der Zündung, S.85) die Entwicklung stark verzögert hat.

7.3.2. Ansteuerung der Treiberstufe

Die Ansteuerung der Treiber wird durch vier GPIO-Pins realisiert. Da auf der ECU zwei ungenutzte Timer verfügbar sind, wurden diese zusätzlich zu den beiden schon genutzten Timern verwendet. Somit verfügt jeder Einspritzkanal über seinen eigenen Timer. Jedoch steht nicht jedem dieser Timer sein zugeordneter Output-Compare-Pin (OC) zur Verfügung, da diese von anderen Funktionen belegt sind.
Da es wenig sinnvoll ist, zwei Einspritzkanäle via OC-Pin und zwei via GPIO laufen zu lassen, wurden die beiden bestehenden Timer auch von ihrer OC-Funktionalität getrennt und die Pins auch via GPIO anzusteuern.
Üblicherweise schaltet ein OC-Pin schneller als ein GPIO-Pin. Um dies zu überprüfen wird diese Zeitdifferenz gemessen und die Ventilöffnungszeit um diese Differenz erhöht. Dies ist nötig da sonst die errechnete Kraftstoffmenge nicht vollständig eingespritzt wird. Die Messung ist in Abbildung 7.13 veranschaulicht. Das Setzen eines GPIO-Pins benötigt $2.51\mu s$ länger. Dieser Wert muss folglich auf die errechnete Öffnungszeit aufaddiert werden, um diese Verzögerung zu kompensieren.

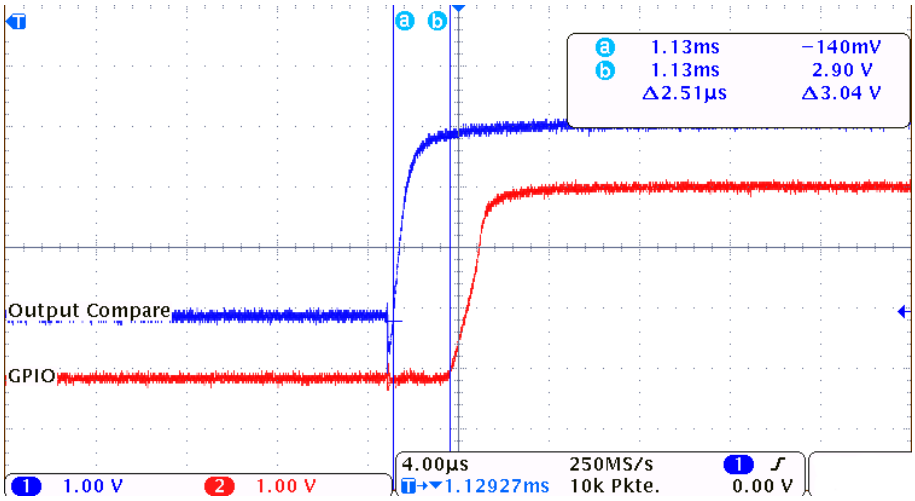

Abbildung 7.13.: Vergleich von Output-Compare und GPIO

7.4. Umsetzung des Telemetriekonzepts

Als Basis dient die Version des ECU-Gateways die im H04 verwendet wird. Hier muss das Kommunikationsprotokoll an die MegaSquirt angepasst werden und die empfangenen Telemetriedaten in das Format der Walbro umgerechnet werden. (vgl. 6.3 Telemetriekonzept, S.70)

7.4.1. Anpassung des ECU-Gateways

Damit die Anforderung erfüllt werden, muss das Gateway die folgende Aufgaben erfüllen:

- **Anfrage schicken**

  Das Kommunikationsprotokoll der MegaSquirt sieht vor, dass die Daten angefordert werden müssen. Hierzu muss die Anforderung „A“ gesendet werden. Eine Initialisierung ist nicht nötig.

- **Daten empfangen (req_task)**

  Die Antwort der MegaSquirt umfasst im Auslieferungszustand $152\mathrm{Byte}$ und wird als Stream ohne Trennzeichen gesendet. Sie enthält neben den relevanten Daten auch sehr viele MegaSquirt-spezifische Daten. Diese sind so im Hawks-Projekt bisher nicht vorgesehen und werden daher auch nicht auf den TTCAN-Bus gesendet.

- **Daten umrechnen**

  Die erhaltenen Daten müssen in das entsprechende Format der Walbro umgerechnet werden. Die entsprechenden Umrechnungsfunktionen sind im Anhang B.1 „Telemetriedaten im Vergleich“ dargestellt.

- **Daten auf CAN**

  Sobald die Daten umgerechnet worden sind, müssen die Daten auf den CAN-Bus gesendet werden. Ziel ist es, dass das Umrechnen der Daten sehr zeitnah vor dem TTCAN-Timeslot des ECU-Gateways liegt damit die Aktualität der Daten gewährleistet werden kann.

Empfangen der Daten via RS232

Die Baudrate der Walbro liegt mit 38400baud relativ niedrig. Die standardmäßige Geschwindigkeit der Schnittstelle der MegaSquirt liegt bei 115200baud. Beide sind auf einen 8-N-1-Betrieb konfiguriert. Dies heißt: 8-Datenbits, keine Parität und ein Stopbit.
Da der Datenstrom weder Trennzeichen noch Anfang oder End-Markierung hat, muss das Paket unterbrechungsfrei empfangen werden. Das Gateway empfängt alle 10ms die TM-Referenznachricht. Folglich sollte das Übertragen der Daten über die Serielle Schnittstelle zeitlich zwischen diese Referenznachrichten passen. Es bleibt also ein Zeitfenster von 9.6ms in denen die Daten vollständig empfangen werden sollen.
Wie lange die Übertragung der Daten bei einer Baudrate von 115200 dauert, zeigt die folgende Rechnung:

$$t = \frac{152Bytes * (8Datenbits + 1Stopbit)}{115200\frac{\text{bit}}{\text{s}}} = \underline{11.875\text{ms}}$$

Selbst bei einer Baudrate von 115200baud würde das Zeitfenster nicht ausreichen, die Daten an einem Stück innerhalb eines Basiszyklus zu empfangen.
Das Datenpaket der MegaSquirt enthält neben den für die Hawks-Telemetrie relevanten Daten auch viele MegaSquirt-spezifische Informationen. Daher kann das Datenpaket gekürzt werden. Allerdings nutzt die Konfigurationssoftware MegaTune diese Kommunikationsschnittstelle ebenfalls und würde dann einen Teil ihrer Funktionalität verlieren.
Um diese Problematik zu lösen wird das Kommunikationsprotokoll erweitert und eine Anforderung für ein Daten-Paket implementiert, das nur noch die für die Hawks-Telemetrie relevanten Informationen enthält. Der Aufbau dieses Datenpakets ist in Tabelle 7.3 auf Seite 94 dargestellt.

Es werden nur noch 24Bytes zwischen ECU und Gateway übertragen. Damit verkürzt sich die Übertragungszeit deutlich. In Tabelle 7.4 sind die Zeiten in Abhängigkeit zur Baudrate aufgelistet. Die Abweichungen des AVR-Takt von "16MHz zur Baudrate ist ebenfalls dargestellt. Hier zeigt sich, dass die Abweichung bei 115200baud deutlich höher liegt als bei 76800baud. Um hier kein Risiko in Hinblick auf die Zuverlässigkeit einzugehen, wird sich für eine Baudrate von 76800baud entschieden.

Diese Geschwindigkeit bringt es aber mit sich, das das Empfangen der Daten nicht mehr durch die RegTask geschehen kann, da diese nur alle $200\mu\text{s}$ ausgeführt

	Bezeichnung	Format	Wertebereicht	Offset
RPM	Drehzahl	16bit, unsigned	$0 - 65535\frac{\mathrm{U}}{\mathrm{min}}$	0x00
PW1	Einspritzungszeit 1&2	16bit, unsigned	$0 - 43253\mu\mathrm{s}$	0x02
PW2	Einspritzungszeit 3&4	16bit, unsigned	$0 - 43253\mu\mathrm{s}$	0x04
adv_deg	Zündwinkel	16bit, unsigned	$0 - 6553.5°\mathrm{KW}$	0x06
MAP	Ansaugrohrdruck	16bit, unsigned	$0 - 6553.5\mathrm{kPa}$	0x08
MAT	Ansauglufttemperatur	16bit, unsigned	$0 - 6553.5°\mathrm{C}$	0x0A
CLT	Kühlungstemperatur	16bit, unsigned	$0 - 6553.5°\mathrm{C}$	0x0C
baro	Umgebungsdruck	16bit, unsigned	$0 - 6553.5\mathrm{kPa}$	0x0E
TPS	Drosselklappenstellung	16bit, unsigned	$0 - 6553.5\%$	0x10
batt	Versorgungsspannung	16bit, unsigned	$0 - 6553.5\mathrm{V}$	0x12
EGO1	Lambdawert	16bit, unsigned	$0 - 6553.5$	0x14
afrtgt	Soll Lambdawert	8 bit, unsigned	$0 - 255$	0x16

Tabelle 7.3.: Aufbau der modifizierten Telemetriedaten seitens der MegaSquirt

Baudrate	**Abweichung** zum AVR-Takt	**Übertragungsdauer** für 24Byte	**Zeit pro Byte**
38400baud	0.2%	5.62ms	$234\mu\mathrm{s}$
76800baud	0.2%	2.82ms	$117\mu\mathrm{s}$
115200baud	3.5%	1.88ms	$87\mu\mathrm{s}$

Tabelle 7.4.: Vergleich der Übertragungszeiten in Abhängigkeit der Baudraten

wird und somit nicht jedes Byte lesen könnte. Da aber bei der MegaSquirt jedes Datenpaket einzelnd angefordert werden muss, ist dies unproblematisch. Es kann nach der Anforderung der Daten ein ausreichend langer Task zum Empfangen genutzt werden (vgl. 4.2.2 Scheduler, S.42).

Das vorgestellte Konzept konnte im Rahmen dieser Arbeit allerdings nicht komplett umgesetzt werden.

7.5. Implementierung der dynamischen konfiguration

Um die konzipierte dynamische Konfiguration via CAN umzusetzen bedarf es folgender Schritte:

- Festlegung des Datenformats
- Empfangen der Daten via CAN
- Einbindung der empfangenen Parameter in die Berechnungen

7.5.1. Aufbau der CAN-Nachricht

Die in Kapitel 6.4.1 ausgewählten Punkte sollen zu Testzwecken implementiert werden. Die Codierung der geforderten Eigenschaften ist in Tabelle 7.5 aufgelistet:

Byte	Bezeichnung	Erläuterung	Format	Wertebereich
0	TTCAN	Protokolldaten	-	-
1	IGN_ADV_COR	Zündwinkel	8bit unsigned	0.0 - 25.4°KW
2	ungenutzt	ungenutzt	-	-
3	ungenutzt	ungenutzt	-	-
4	ungenutzt	ungenutzt	-	-
5	ungenutzt	ungenutzt	-	-
6	ungenutzt	ungenutzt	-	-
7	FLAGS	Steuerbits	-	-

Tabelle 7.5.: Aufbau der CAN-Nachricht zur dynamischen Konfiguration

Es passen also wie gefordert alle Kommandos in eine CAN-Nachricht. Zudem ist hier noch ausreichend Kapazität für Erweiterungen vorhanden, sodass davon ausgegangen werden kann, dass eine Nachricht auch in Zukunft ausreichen wird.

Tabelle 7.6 zeigt den Aufbau des FLAGS-Byte:

Alle Steuerbits sind High-Aktiv ausgelegt, das heißt, das sie bei einem Wert von 0 keinen Einfluss auf die ECU haben. Dies dient der Umsetzung der Sicherheitsaspekte aus 6.4.2 „Sicherheitskonzepte“.

Durch die schnelle Reaktionszeit der ECU bietet es sich an, die entsprechende Nachricht in jedem Basiszyklus zu senden, also mit einer Frequenz von 100Hz (vgl. 4.2.1 Matrixzyklus, S.41).

Bit	Bezeichnung	Funktion
0	IGN_MAP_CHANGE	Kennfeldwechsel Zündung
1	FUEL_MAP_CHANGE	Kennfeldwechsel Einspritzung
2	IGN_CUT	Zündunterbrechung
3	FUEL_CUT	Unterbrechung der Einspritzung
4	ungenutzt	ungenutzt
5	ungenutzt	ungenutzt
6	ungenutzt	ungenutzt
7	ungenutzt	ungenutzt

Tabelle 7.6.: Aufbau des FLAGS-Byte der CAN-Nachricht zur dynamischen Konfiguration

7.5.2. Implementierung auf der ECU

Zum Empfangen der CAN-Daten seitens der MegaSquirt wird die bisher vorhandene CAN-Implementierung komplett umgeschrieben. Die Initialisierung wird so angepasst, dass nur Nachrichten mit der ID $0x42$ akzeptiert werden. Dies entspricht der nächsten freien ID im Hawks-Telemetrie-System.

In der Empfangs-ISR werden alle Datenbytes in ein entsprechendes globales Array geschrieben, sodass die Parameter global und sofort verfügbar sind.

Einbinden der Parameter in die Berechnungen

Zündunterbrechung

Die Zündunterbrechung wird dadurch realisiert, dass in der ISR für das Laden der Zündspulen das empfangene Steuerbit abgefragt und im Falle einer Zündunterbrechung die ISR verlassen wird.
So wird verhindert, das ein neuer Zündspulen-Ladevorgang beginnen kann (vgl. 6.4.1 Steuerbits, S.74 & 5.3.2 Timing und Ansteuerung von Aktoren, S.57).

Unterbrechung der Einspritzung

Hierbei wird der Beginn eines neuen Einspritzvorgangs dadurch unterbunden, dass in der ISR_Ign_TimerIn die Ventile nicht geöffnet werden, wenn das entsprechende Steuerbit anliegt. (5.3.2 Timing und Ansteuerung von Aktoren, S.57).

Kennfeldwechsel

Basierend auf dem entsprechenden Steuerbit wird in der Super Loop gewählt, welches Kennfeld als Basis zur Berechnung verwendet werden soll. Hierzu wird das von der MSextra vorgesehene, zweite Kennfeld genutzt. Dieses ist ursprünglich dazu gedacht, ab einem definierbarem Grenzwert eines Sensors eingesetzt zu werden, um so beispielsweise in einem sehr niedrigen Drehzahlbereich ein alternatives Kennfeld nutzen zu können.

Diese Eigenschaft wird hier ausgenutzt indem die Bedingung für den Kennfeldwechsel angepasst und das entsprechende Steuerbit abgefragt wird.

Anpassung des Zündwinkels

Bei der Berechnung des Zündwinkels wird der übertragene Wert von dem errechneten Winkel abgezogen, bevor er in den entsprechenden Timerwert umgerechnet wird (vgl. 5.3.2 Timing und Ansteuerung von Aktoren, S.57).

Sicherheitsaspekte

Um die in der Konzeption diskutierten Sicherheitsaspekte umzusetzen wird ein Software-Timer eingesetzt. Dieser wird in der ISR der RTC implementiert und zählt somit in einem Takt von $0.128\mu s$.

Wird eine CAN-Nachricht empfangen, wird der Timer auf einen Startwert gesetzt. Dieser wird in der $\mathrm{ISR\_RTC}$ dekrementiert. Erreicht er den Wert 0, werden alle empfangenen Steuerbits und Steuerdaten auf 0 gesetzt. So haben sie keinerlei Einfluss mehr auf die ECU.

Sobald wieder eine gültige Nachricht empfangen wird, wird der Timer neu gestartet und entsprechend die Konfiguration wieder angenommen.

Die Dimensionierung dieses „Timeouts“ orientiert sich an der Länge eines Matrixzyklus. Wird für mehr als $40\mathrm{ms}$ keine Nachricht empfangen, bedeutet dies, dass die Steuernachricht für die ECU über einen kompletten Matrixzyklus ausgeblieben ist. Bei der angestrebten Nachrichten-Frequenz von $100\mathrm{Hz}$ entspricht dies vier fehlenden Nachrichten.

7.5.3. Tests

Zum Testen der dynamischen Konfiguration wurde die in 7.1.1 „RPM-Simulation“ vorgestellte RPM-Simulation erweitert. Sie sendet mit einer Frequenz von $100\mathrm{Hz}$ die Steuer-Nachricht auf den CAN-Bus.

Um eine Ausgabe und einen Status zu erhalten, wird das Kommunikationsprotokoll zwischen MegaSquirt und der Konfigurationssoftware MegaTune erweitert. Es werden alle empfangenen CAN-Daten übertragen sowie der Status der Betriebsmodi, die durch die Steuerbits gesetzt werden können.

Test der Funktionalitäten

Die Funktionalität der dynamischen Konfiguration wurde anhand der unter 7.2.3 „Überprüfung des Zündwinkels“ beschriebenen Methode überprüft. Es wurde gezeigt, dass auch hier die in MegaTune angezeigten Werte einwandfrei umgesetzt werden. So konnte die Funktionalität der dynamischen Konfiguration getestet werden.

Unterbrechung der CAN-Verbindung

Die Zuverlässigkeit des Systems wird durch die Unterbrechung der CAN-Verbindung getestet. Hier sollte der unter 7.5.2 „Sicherheitsaspekte“ angesprochene TimeOut greifen und die ECU normal weiterarbeiten.
Sobald die Verbindung wieder besteht, soll die dynamische Konfiguration wieder möglich sein.

Die Tests verliefen einwandfrei und zuverlässig.

8. Fazit

Diese Arbeit zeigt, dass die Motorsteuerung eines Verbrennungsmotors eine umfangreiche und komplexe Thematik ist, die sehr stark von Interdisziplinarität geprägt wird. Hier sind Informatik, Elektro- und Fahrzeugtechnik eng miteinander verbunden.

Die Entwicklung der ECU erforderte vor allem die Auseinandersetzung mit Problemen aus dem Bereich der Elektrotechnik (vgl. 7.2.4 Probleme der Zündung). Diese führten dazu, dass die ECU nicht auf dem Motorprüfstand getestet werden konnte da die dafür nötigen Hardware-Vorraussetzungen nicht gegeben waren.

Durch diese unerwartete Problematik wurde die eigentliche Kernthematik, die dynamische Konfiguration, relativ kurz behandelt. Daher ist diese Arbeit als eine Analyse zu betrachen, die eine spätere Konzeption einer ECU ermöglicht.

So lassen sich abschließend folgende Schlüsse ziehen:

Eignung der MegaSquirt für das Hawks-Projekt

Die MegaSquirt ist keine geeignete Basis für eine ECU für das Hawks-Projekt. Ohne Modifikationen werden die von Hawks gestellten Anforderungen nur sehr begrenzt erfüllt, sodass hier Entwicklungsbedarf an der Software und der Hardware besteht.

Die Struktur des Quellcodes ist undurchsichtig und schlecht erweiterbar. Hier macht es sich bemerkbar, dass die Software vom MegaSquirt Projekt konzipiert und dann vom MegaSquirt Extra-Projekt modifiziert wurde. Hier haben eindeutig unabhängig voneinander verschiedene Entwickler aufeinander aufgebaut ohne saubere Schnittstellen zu definieren.
Dennoch können hier einige der Software-Konzepte als Vorbilder für eine eventuelle Eigenentwicklung dienen. So ist beispielsweise die Erkennung der Drehzahl, wie auch das Timing der Aktoren klar strukturiert und sinnvoll umgesetzt worden (vgl. 5.3.2 Analyse der Funktionalitäten, S.55).

Auch die Hardware ist nur bedingt geeignet, da sie sehr aufwendig modifiziert werden müsste, wie in Kapitel 6 „Konzeption“ beschrieben wurde. Um hier

Bauraum zu sparen, müssten alle Platinen neu entwickelt und gegebenfalls zusammengefasst werden, damit man alle nötigen Bauteile in einem kompakten Gehäuse zusammenfassen kann.

Letztendlich müsste also die MegaSquirt komplett umgestaltet werden, was dem Aufwand einer kompletten Eigenentwicklung in etwa gleichkommt.

Aussicht auf eine komplette Eigenentwicklung

Der Aufwand, eine ECU von Grund auf selbst zu entwickeln ist nicht zu unterschätzen. Hierzu sind im Groben drei Arbeitspakete abzuarbeiten:

- **Embedded Software**

  Es muss eine klare Software-Architektur entworfen werden, die auf Übersicht und Flexibilität optimiert ist, um den stetig wachsenden Anforderungen und dem sich ändernden Umfeld gewachsen zu sein. Eine interessante Option wäre auch die Verwendung von FPGAs, um einige Optionen auszulagern. Hier könnte beispielweise die Erkennung der Drehzahl und der Kurbelwellenstellung in Hardware realisiert werden um die Software zu entlasten.

- **Hardware**

  Eine solide Hardware-Plattform, die der Umgebung im Kraftfahrzeug gewachsen ist und deren Einflüsse von der Software fernhält, ist eine Grundvorrausetzung für die erfolgreiche Entwicklung. Hier sollte für die umfangreichen Entwicklungsarbeiten zuerst eine angemessene Evaluations-Umgebung geschaffen werden, die erst später in Hinblick auf Bauraum und Gewicht optimiert wird.

- **Konfigurations-Software**

  Es muss eine angemessene Konfigurationsmöglichkeit für die Kraftfahrzeugtechniker geben, wie in den Anforderungen festgehalten. Diese Software sollte auch die Möglichkeit bieten, Daten zur Laufzeit anzuzeigen und zu Diagnosezwecken aufzuzeichnen.

Möglichkeiten einer selbstentwickelten ECU

Eine vollständig selbstentwickelte Motorsteuerung brächte eine Vielzahl von Vorteilen. Hierzu zählen zum einen die Transparenz und die Unabhängigkeit von anderen Herstellern. Der Aufbau und das Verhalten der ECU wäre in jeder Situation vollständig bekannt, was das exakte Timing und die Abstimmung sehr erleichtern würde.

Zudem könnte man die ECU komplett integrieren und beispielsweise genau die Daten übertragen, die für Hawks relevant sind und ist nicht auf die Daten festgelegt, die einem eine kommerzielle Motorsteuerung anbietet.

Zu guter letzt würde eine Eigenentwicklung im Design Report der Formula Student wichtige Pluspunkte sammeln und damit den Erfolg des Teams auch abseits der Rennstrecke fördern.

8.1. Schlusswort

Das Thema Motorsteuerung wird für das Hawks-Projekt aktuell bleiben. Die bisher verwendete Walbro funktioniert zwar zuverlässig, ist aber bezüglich der Konfigurierbarkeit sehr unflexibel, sodass hier auch seitens der Motorbaugruppe der Wunsch besteht, geeigneteren Ersatz zu finden.
Zudem sucht die Telemetriebaugruppe einen Weg, in die Steuerung des Motors eingreifen zu können und so diese zentrale Komponente des Fahrzeugs besser in das Telemetriesystem integrieren zu können. Hier könnte eine Eigenentwicklung die Wünsche von Motor- und Telemetriebaugruppe vereinen.

Tabellenverzeichnis

Abbildungsverzeichnis

Literaturverzeichnis

[Freescale 2007] Freescale Semiconductors (Hrsg.): *MC9S12C Family Datasheet*. 2007. – URL http://www.freescale.com/files/microcontrollers/doc/data_sheet/MC9S12C128V1.pdf. – Zugriffsdatum: 09.12.2008

[Bowling und Grippo 2008] Bowling, Bruce ; Grippo, Al ; MegaManual.com (Hrsg.): *MegaManual.com*. 2008. – URL http://www.megamanual.com/. – Zugriffsdatum: 10.12.2008

[Brandt 2008] Brandt, Johannes: *Applikation des HAWK07 Motorsteuergerätes*. 2008. – Studienarbeit an der HAW Hamburg im Studiendepartment Maschinenbau und Produktion

[Eichlseder u. a. 2008] Eichlseder, Helmut ; Klüting, Manfred ; Piock, Walter F.: *Grundlagen und Technologien des Ottomotors*. SpringerWienNewYork, 2008. – ISBN 978-3-211-25774-6

[Formula Student Germany 2008] Formula Student Germany ; FSG (Hrsg.): *Formula Student Germany Rules 2009*. 2008. – URL http://www.formulastudent.de/uploads/media/FSG_RULES_2009_Version20081107_01.pdf. – Zugriffsdatum: 08.12.2008

[Haase 2007] Haase, Sebastian: *Telemetrie im Formula Student Rennwagen auf Basis von CAN Bus, Datenspeicherung und Wireless LAN Technologien*. 2007. – Bachelor Arbeit an der HAW Hamburg im Department Informatik

[Kolbe 2008] Kolbe, Felix: *Redundanzkonzept eines Time-Triggered Bussystems in einem Formula Student Rennwagen: Modellierung, Implementierung und Anwendung in der Antriebsschlupfregelung*. 2008. – Bachelor Arbeit an der HAW Hamburg im Department Informatik

[Murrey u. a. 2008] Murrey, James ; Culver, Ken ; Ringwood, Philip ; MS-Extra.com (Hrsg.): *MegaSquirt2 Extra*. 2008. – URL http://www.msextra.com/ms2extra/. – Zugriffsdatum: 10.12.2008

[Pont, Michael J. 2001] Pont, Michael J.: *Patterns for Time-Triggered Embedded Systems*. Addison-Wesley, 2001. – ISBN 0-201-33138-1

[Robert Bosch GmbH 2003] Robert Bosch GmbH (Hrsg.): *Kraftfahrtechnisches Handbuch*. Friedrich Vieweg & Sohn Verlag, 2003. – ISBN 3-528-23876-3

[Robert Bosch GmbH 2005] Robert Bosch GmbH (Hrsg.): *Ottomotor-Managment*. Friedrich Vieweg & Sohn Verlag, 2005. – ISBN 3-8348-0037-6

[Schuckert 2007] Schuckert, Simon: *Mikrocontrollerbasierte Telemetrie und Echtzeitauswertung von Sensordaten im Formula Student Rennwagen*. 2007. – Bachelor Arbeit an der HAW Hamburg im Department Informatik

Glossar

Arbeitsspiel
Als Arbeitsspiel wird ein kompletter Durchlauf aller Motortakte bezeichnet. Siehe 2.2 4-Takt-Zyklus, S.14.

Basiszyklus
Der Matrixzyklus kann in mehrere Basiszyklen unterteilt sein. Siehe 4.2.1 Matrixzyklus, S.41.

Daughterboard
Eine Platine, die den μController der MegaSquirt2 beherbergt und in den DIP40-Sockel der MegaSquirt gesteckt werden kann. (vgl. 5.2.1 Struktur der Hardware, S.47)

Flywheel
Bezeichnet das Zahnrad auf der Kurbelwelle, das der Kurbelwellensensor induktiv ausliest. (vgl. 2.6.1 Sensorik, S.28)

GCC
Freie Sammlung von Compilern, http://gcc.gnu.org/

Gemisch
Gasgemisch, das aus Luft bzw Sauerstoff und Kraftstoff besteht

Kennfeld
Mehrdimensionales Array das zur Abstraktion einer komplexen Funktion dient. Englisch auch als „Map“ bezeichnet, nicht zur verwechseln mit dem Ansaugrohrdruck „MAP“. Siehe auch 2.6.2 Kennfelder, S.29.

Klopfen
schlagartige, unkontrollierte Verbrennung bei zu frühem Zünden. (vgl. 2.5.1 Zündwinkel, S.24)

Klopfgrenzen
Der Zünndwinkel ab dem Klopfen Auftritt. (vgl. 2.5.1 Zündwinkel, S.24)

Kurbelwelle
Rotierende Welle, die die Hubbewegung der Kolben über die Pleuelstange in eine Rotation umsetzt.

Matrixzyklus
Definierte Folge von Zeitfenstern. Siehe 4.2.1 Matrixzyklus auf Seite 41.

MegaStim
Eine seitens der MegaSquirt-Entwickler entwickeltes Stimuli-Platine. Diese simuliert die wesentliche Sensorik wie Temperatur, Drehzahl oder Drosselklappenstellung.

MegaTune
MegaTune ist die zur MegaSquirt gehörige Konfigurations-Software.

Ottomotor
Fremdgezündeter Verbrennungsmotor. Benannt nach seinem Entwickler Nikolaus August Otto, präsentiert auf der Pariser Weltausstellung 1878. Siehe auch 2 Grundlagen der Motortechnik, S.12.

Peak and Hold
Eine Methode, die zur Ansteuerung von Aktoren dient, die schnell ansprechen müssen, aber eine limitierte Spannung benötigen. Es wird am Anfang eine höhere Spannung genutzt (Peak), um den Aktor schnell in Bewegung zu versetzen, und dann auf die Arbeitsspannung (Hold) reduziert.

Phase
Bezeichnet eine Umdrehung innerhalb eines Arbeitsspiels. Siehe 2.2 4-Takt-Zyklus, S.14.

Restriktor
Ein Rohr mit definierter Länge und definiertem Durchmesser, das hinter der Drosselklappe im Ansaugtrakt sitzt. Es dient der Limitierung der Motorleistung durch begrenzung der maximalen Luftmenge.

Rückzündung
Bezeichnet ein übergreifen des Verbrennungsprozesses in den Ansaugtrakt. Dies kann durch fehlerhafte Einlassventile oder auch durch zu hohe Temperaturen verursacht werden. Rückzündungen können zur Zerstörung des Ansaugtraks führen.

Scheduler
Organisiert die Abarbeitung von unterschiedlichen Tasks auf einem Betriebssystem

Schließzeit
Die Zeit, die die Zündspule zum Laden benötigen. Der Begriff bezieht sich darauf, das in dieser Zeit der Stromkreis über die Zündspule geschlossen ist.

Stützstelle
Eine Stützstelle ist ein Wert einer Eingangsgröße eines Kennfeldes. Auf diesen Wert basieren die konfigurierten Werte des Kennfelds. (vgl. 2.6.2 Kennfelder, S.29)

Super Loop
Bezeichnet eine Software-Struktur, die endlos durchlaufen wird um zeitunkritische Aufgaben abzuarbeiten. (siehe 5.3.1 Struktur des Embedded-Codes, S.54) [vgl. Pont, Michael J., 2001]

Telemetrie
Als Telemetrie bezeichnet man das Erfassen von Messwerten und deren Übertragung an eine entfernten Ort.

Umdrehung
Bezeichnet eine vollständige Drehung der Kurbelwelle.

Walbro
Walbro ist ein italienischer Hersteller von Steuergeräten. Der Begriff wird in dieser Arbeit als Synonym für die im Hawks-Projekt verwendete ECU vom Typ Walbro HPUH-1 verwendet.

Wasted Spark
Ein Zündungsverfahren, beidem jeweils zwei Zylinder gezündet werden allerdings befindet sich nur ein Zylinder dabei im Arbeitstakt. (siehe 2.5.2 Wasted Spark, S.26)

Zündreihenfolge
Gibt die Reihenfolge an, in der die Zylinder von Mehrzylindermotoren gezündet werden. Sie ist durch die Bauart des Motors festgelegt. (vgl. 2.5.2 Zündverteilung, S.25)

Zündwinkel
Zündzeitpunkt in Relation zur Stellung der Kurbelwelle

Abkürzungsverzeichnis

ADC	Analog Digital Converter
ADV	Advance, Abkürzung für die englische Bezeichnung für Zündwinkel
ASR	Antriebsschlupfregelung
CAN	Controller Area Network
CLT	Cooling Temperature, Kühlertemperatur
COP	Coil on Plug, siehe Siehe Glossar
ECU	Engine Control Unit
EFI	Electronic Fuel Injection
EGO	Exhaust gas oxygen sensor, Lambda-Sonde
FP	Fuel Pump, Kraftstoffpunpe
FPGA	Field Programmable Gate Array, programmierbarer IC
GND	Ground, elektronische Masse
GPIO	General Purpose Input/Output
H04	Interne Bezeichnung für den Hawks Rennwagen aus der Saison 2008
HW	Hardware
IAC	Idle Air Control
IC	Input Capture
IGN	Ignition, Zündung
INJ	Injection, Einspritzung
ISR	Interrupt Service Routine

KW	Kurbelwelle
MAP	Manifold Absolute Pressure, Ansaugrohrdruck
MAT	Manifold Air Temperatur, Ansauglufttemperatur
MS	MegaSquirt
MS2	MegaSquirt 2
MSextra	MegaSquirt 2 Extra
NW	Nockenwelle
OC	Output Compare
OT	Oberer Totpunkt
PW	Pulsewidth, Pulsweite im Bezug auf die Einspritzung
PWM	Pulse Width Modulation
ROV	Rotierende Verteilung, bezogen auf Zündverteilung
RPM	Rounds per Minute, Umdrehungen pro Minute
RTC	Real Time Clock
RUV	Ruhende Verteilung, bezogen auf die Zündverteilung
TM	Timemaster
TPS	Throttle Position Sensor, Drosselklappenstellung
TTCAN	Time-Triggered CAN

A. Dokumentation MegaSquirt2extra

Die hier dargestellten Tabellen und Abbildungen beziehen sich auf die MegaSquirt2 unter Verwendung des MegaSquirt2extra Quellcodes (vgl. 5.3 Analyse der Embedded-Software, S.53). Es wird die Konfiguration im Auslieferungszustand dokumentiert.

A.1. Pin Belegung HC9S12C64

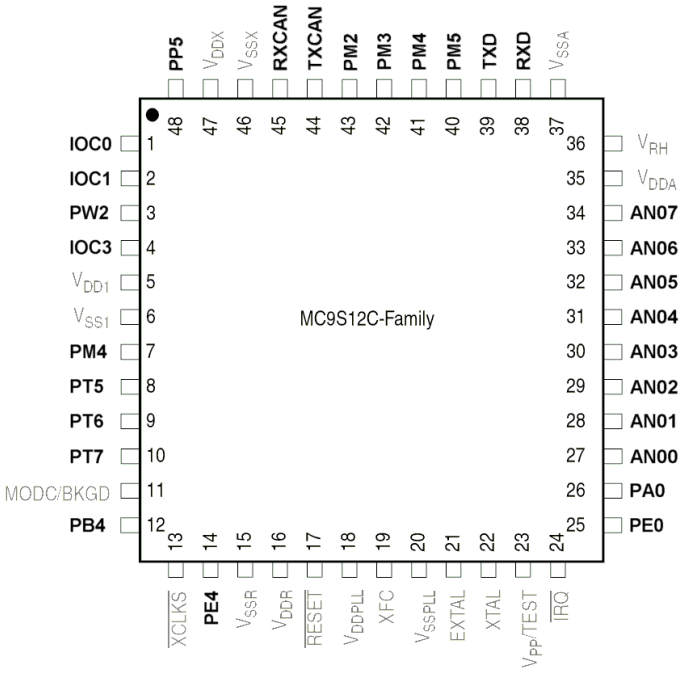

Abbildung A.1.: Pin Belegung HC9S12C64

´

Eine tabellarische Aufstellung über die Pins und ihre Verwendung befindet sich auf der folgenden Seite.

Pin	Bez.	Signal	IO	Erläuterung
1	IOC0	IRQ	Input	IC Timer0, KW-Ssensor
2	IOC1	INJ1	Output	OC T1, Einspritzung Bank1
3	PM2	PWM1	Output	PWM Einspritzung Bank1
4	IOC3	INJ2	Output	OC T2, Einspritzung Bank2
7	PM4	PWM2	Output	PWM Einspritzung Bank2
8	PT5	IGN	Output	Ansteuerung des Zündmoduls
9	PT6	IAC1	Output	Ansteuerung des IAC Treibers
10	PT7	IAC2	Output	Ansteuerung des IAC Treibers
12	PB4	IACenbl	Output	Ansteuerung des IAC Treibers
14	PE4	FP	Output	Ansteuerung Treibstoffpumpe
27	AN0	AD0-1	Input	ADC, Saugrohrdrucksensor
28	AN1	AD1-1	Input	ADC, Ansauglufttemperatur
29	AN2	AD2-1	Input	ADC, Kühlungstemperatur
30	AN3	AD3-1	Input	ADC, Drosselklappenstellung
31	AN4	AD4-1	Input	ADC, Batteriespannung
32	AN5	AD5-1	Input	ADC, Lambda-Sonde
33	AN6	AD6-1	Input	ADC, Luftdrucksensor
34	AN7	AD7-1	Input	ADC, Klopfsensor
38	RXD	Rx	SCI	Serielle Schnittstelle, RX
39	TXD	Tx	SCI	Serielle Schnittstelle, TX
40	PM5	WarmLED	Output	Ansteuerung Warm-Up LED
41	PM4	AccelLED	Output	Ansteuerung Acceleration LED
42	PM3	InjLED	Output	Ansteuerung Injection LED
43	PM2	Idle	Output	Ansteuerung „Fast Idle“
44	TXCAN	CANTx	MSCAN	CAN-TX
45	RXCAN	CANRx	MSCAN	CAN-RX

Tabelle A.1.: MegaSquirtII Pin Belegung des HC9S12C64

A.2. Vergleich MegaSquirt1 und 2

	MegaSquirt I	**MegaSquirt II**
Controller	Motorola 68HC908GP32	Motorola HC9S12C64
Package	40Pin PDIP	48Pin LQFP
Architektur	8bit	16bit
Takt	8MHz	24MHz
Flash	32KB	64KB
RAM	512B	2KB
HW-Timer	2x16bit	8x16bit
ADCs	8*10bit	8*10bit
GPIOs	33	31
Kommunikation	SPI, SCI	SPI, SCI, CAN

Tabelle A.2.: Vergleich der uController ins MS1 und MS2[vgl. Bowling und Grippo, 2008]

A.3. Belegung Sub-D37 Stecker

Die Belegung des Sub-D37-Steckers der MegaSquirt ist die folgende:

Pin	Bezeichnung	Beschreibung
1,2	GND	Masse
3	SPR1	Spare 1, frei
4	SPR2	Spare 2, frei
5	SPR3	Spare 3, frei
6	SPR4	Spare 4, frei
7-19	GND	Masse
20	MAT	Eingang, Ansauglufttemperatur
21	CLT	Eingang, Kühlungstemperatur
22	TPS	Eingang, Drosselklappenpotentiometer
23	EGO	Eingang, Lambdasondensteuergerät
24	KW-Sensor	Eingang, Kurbelwellen-Sensor
25	IAC1A	Ausgang, Ansteuerung IAC
26	Vref	Ausgang, +5V Referenzspannung
27	IAC1B	Ausgang, Ansteuerung IAC
28	+12V	Eingang, Spannungsversorgung ECU
29	IAC2A	Ausgang, Ansteuerung IAC
30	IDL	Ausgang, Ansteuerung Fast Idle Valve
31	IAC2B	Ausgang, Ansteuerung IAC
32,33	INJ1	Ausgang, Einspritzung Kanal 1
34,35	INJ2	Ausgang, Einspritzung Kanal 2
36	IGN_OUT	Ausgang, Zündmodul
37	FP1	Ausgang, Kraftstoffpumpenrelais

Tabelle A.3.: Pin Belegung der MegaSquirt

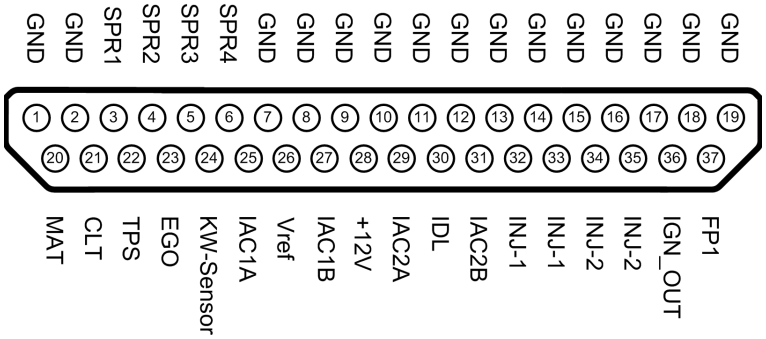

Abbildung A.2.: Pin-Belegung Sub-D37

A.4. Telemetriedaten der MegaSquirt Extra

Offset	Name	Format	Einheit
0	seconds	16bit uint	$1\mathrm{s}/bit$
2	pw1	16bit uint	$0.66\mu\mathrm{s}/bit$
4	pw2	16bit uint	$0.66\mu\mathrm{s}/bit$
6	rpm	16bit uint	$1\mathrm{U}/bit$
8	spk_adv	16bit int	$0.1°\mathrm{KW}/bit$
10	squirt	8bit char	–
11	engine	8bit char	–
12	afrtgt1	8bit char	–
13	afrtgt2	8bit char	–
14	wbo2_en1	8bit char	–
15	wbo2_en2	8bit char	–
16	baro	16bit int	$0.1\mathrm{kPa}/bit$
18	map	16bit int	$0.1\mathrm{kPa}/bit$
20	mat	16bit int	$0.1°\mathrm{C}/bit$
22	clt	16bit int	$0.1°\mathrm{C}/bit$
24	tps	16bit int	$0.1\%/bit$
26	batt	16bit int	$0.1\mathrm{V}/bit$
28	ego1	16bit int	$0.1/bit$
30	ego2	16bit int	$0.1/bit$
32	knock	16bit int	$0.01\mathrm{V}/bit$
34	egocor1	16bit int	$1\%/bit$
36	egocor2	16bit int	$1\%/bit$
38	aircor	16bit int	$1\%/bit$
40	warmcor	16bit int	$1\%/bit$
42	tpsaccel	16bit int	$1\%/bit$
44	tpsfuelcut	16bit int	$1\%/bit$
46	barocor	16bit int	$1\%/bit$
48	gammae	16bit int	$1\%/bit$
50	vecurr1	16bit int	$1\%/bit$
52	vecurr2	16bit int	$1\%/bit$
54	iacstep	16bit int	$1\mathrm{step}/bit$
56	cold_adv_deg	16bit int	$0.1°\mathrm{KW}/bit$
58	tpsdot	16bit int	$\frac{0.1\%}{0.1\mathrm{s}}/bit$
60	mapdot	16bit int	–
62	coil_dur	16bit uint	$0.1\mathrm{s}/bit$
64	maf	16bit int	–
66	fuelload	16bit int	–

68	fuelcor	16bit int	$1\%/bit$
70	port_status	8bit uchar	–
71	knk_rtd	8bit uchar	$0.1°\mathrm{KW}/bit$
72	EAEfcor1	16bit uint	–
74	egoV1	16bit int	$0.01\mathrm{V}/bit$
76	egoV2	16bit int	$0.01\mathrm{V}/bit$
78	status1	8bit uchar	–
79	status2	8bit uchar	–
80	status3	8bit uchar	–
81	status4	8bit uchar	–
82	looptime	16bit uint	$0.66\mu\mathrm{s}/bit$
84	istatus5	16bit uint	–
86	tpsadc	16bit uint	–
88	fuelload2	16bit int	–
90	ignload	16bit int	–
92	ignload2	16bit int	–
94	spare	16bit int[5]	–
104	synccnt	8bit uchar	–
105	timing_err	8bit char	–
106	dt3	32bit ulong	–
110	wallfuel1	32bit ulong	–
114	gpioadc	16bit uint[8]	–
130	gpiopwmin	16bit uint[4]	–
138	gpioport	8bit uchar[3]	–
141	adc6	16bit uint	–
143	adc7	16bit uint	–
145	wallfuel2	32bit ulong	–
149	EAEfcor2	16bit uint	–
151	boostduty	8bit uchar	–
152	—	16bit uint	Paritätsdaten

Tabelle A.4.: Telemetriedaten der MegaSquirt Extra

B. Dokumentation der Realisierung

B.1. Telemetriedaten im Vergleich

Falls die Bezeichnung für eine Information zwischen Walbro und MegaSquirt abweicht ist die Bezeichnung der MegaSquirt in Klammern ergänzt. Die Umrechnungsfunktionen beziehen sich auf „W“ (Walbro) und „MS2“ (MegaSquirt2).

Bezeichnung Walbro (MS2)	**Format Walbro** Einheit, Datentyp	**Format MS2** Einheit, Datentyp	**Umrechnungsfunktion** Walbro = MS2 * Faktor
MAP	$0.1\text{mbar}/bit$, 2B	$0.1\text{kPA}/bit$, 2B	W = MS2
TAir (MAT)	$0.6°\text{C}/bit$, 1B	$0.1°\text{C}/bit$, 2B	W = MS2 * $\frac{1}{6}$
TEngine (CLT)	$0.6°\text{C}/bit$, 1B	$0.1°\text{C}/bit$, 2B	W = MS2 * $\frac{1}{6}$
TPS	$0.5\%/bit$, 1B	$0.1\%/bit$, 2B	W = MS2 * $\frac{1}{5}$
RPM	$1\frac{\text{U}}{\text{min}}/bit$, 2B	$1\frac{\text{U}}{\text{min}}/bit$, 2B	W = MS2
Inj1 (pw1)	$1\mu\text{s}/bit$, 2B	$0.66\mu\text{s}/bit$, 2B	W = MS2 * $\frac{2}{3}$
Inj2 (pw1)	$1\mu\text{s}/bit$, 2B	$0.66\mu\text{s}/bit$, 2B	W = MS2 * $\frac{2}{3}$
Inj3 (pw2)	$1\mu\text{s}/bit$, 2B	$0.66\mu\text{s}/bit$, 2B	W = MS2 * $\frac{2}{3}$
Inj4 (pw2)	$1\mu\text{s}/bit$, 2B	$0.66\mu\text{s}/bit$, 2B	W = MS2 * $\frac{2}{3}$
Spark1 (adv_deg)	$0.1°\text{KW}/bit$, 2B	$0.1°\text{KW}/bit$, 2B	W = MS2
Spark2 (adv_deg)	$0.1°\text{KW}/bit$, 2B	$0.1°\text{KW}/bit$, 2B	W = MS2
Spark3 (adv_deg)	$0.1°\text{KW}/bit$, 2B	$0.1°\text{KW}/bit$, 2B	W = MS2
Spark4 (adv_deg)	$0.1°\text{KW}/bit$, 2B	$0.1°\text{KW}/bit$, 2B	W = MS2
Lambda (EGO1)	$0.01/bit$, 1B	$0.1/bit$, 2B	W = MS2 * $\frac{1}{10}$
LambdaTarget (afrtgt)	$0.01/bit$, 1B	$0.1/bit$, 1B	W = MS2 * $\frac{1}{10}$
VBatt (batt)	$0.07\text{V}/bit$, 1B	$0.1\text{V}/bit$, 2B	W = MS2 * $\frac{7}{10}$

Tabelle B.1.: Vergleich der Telemetriedaten von MegaSquirt2 (MS2) und Walbro

B.2. Schaltpläne

Hier befindet sich die Schaltpläne aller im Rahmen dieser Arbeit gefertigten und entworfenen Schaltungen. Diese befinden sich ebenfalls im elektronischen Anhang im Format der Layout-Software „Eagle“ (siehe C Inhalt der CD, S.125).

B.2.1. Einspritzungs-Treiberstufe

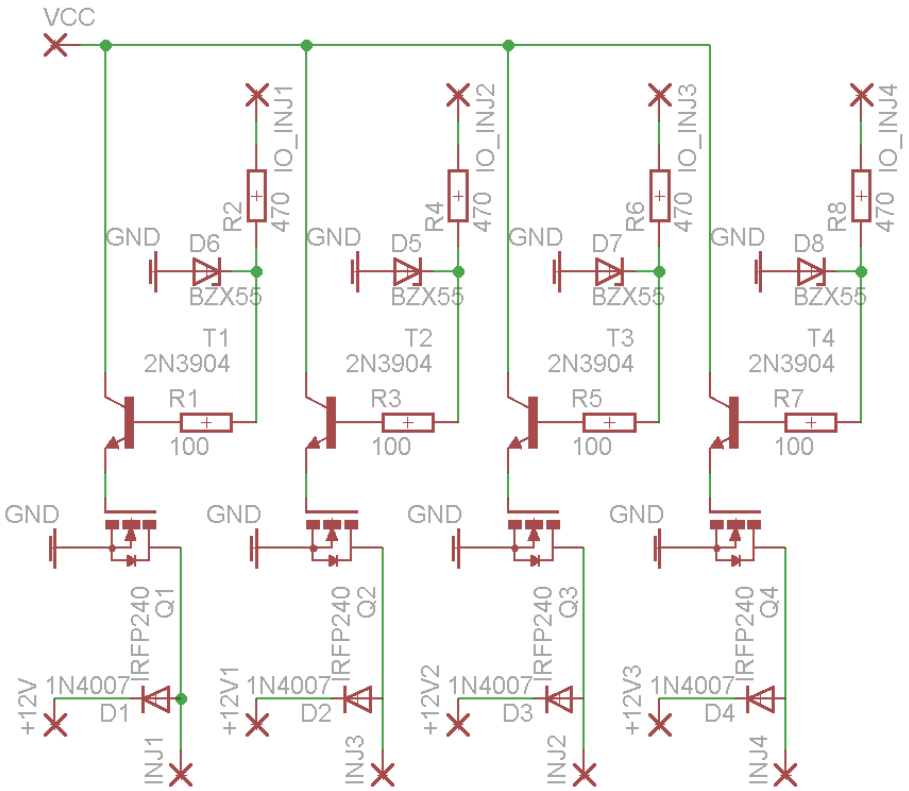

Abbildung B.1.: Schaltplan für die Einspritzungs-Treiberstufe

B.2.2. Zündendstufe

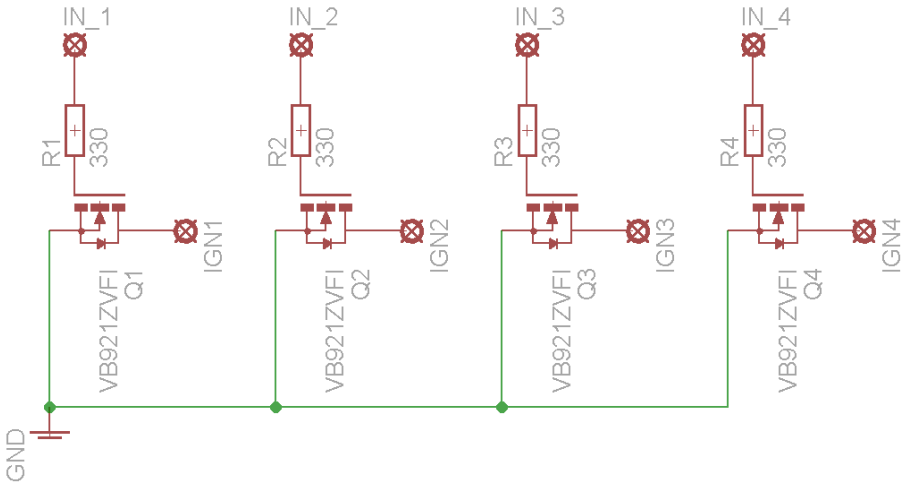

Abbildung B.2.: Schaltplan der Zündendstufe

B.2.3. Adapterboard

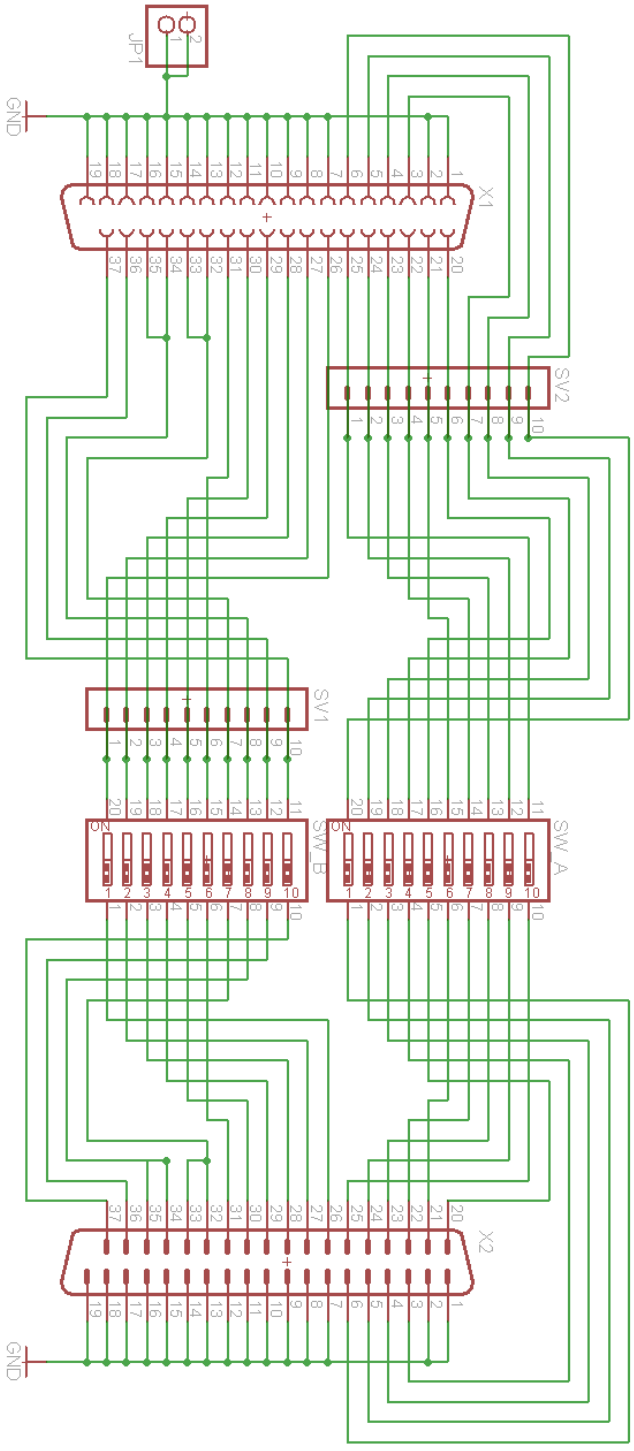

Abbildung B.3.: Schaltplan des Adapterboards

B.3. Pin-Belegungen

B.3.1. Belegung Sub-D37

Pin	Bezeichnung	Beschreibung
1,2,7-19	GND	Masse
3	CANL	CANL
4	CANH	CANH
5	GND	GND
6	NW-Sensor	Nockenwellensensor
20	MAT	Eingang, Ansauglufttemperatur
21	CLT	Eingang, Kühlungstemperatur
22	TPS	Eingang, Drosselklappenpotentiometer
23	EGO	Eingang, Lambdasondensteuergerät
24	KW-Sensor	Eingang, Kurbelwellen-Sensor
25	INJ3	Ausgang, Einspritzung Kanal 3
26	Vref	Ausgang, +5V Referenzspannung
27	INJ4	Ausgang, Einspritzung Kanal 4
28	+12V	Eingang, Spannungsversorgung ECU
29	frei	frei
30	IDL	Ausgang, Ansteuerung Fast Idle Valve
31	frei	frei
32,33	INJ1	Ausgang, Einspritzung Kanal 1
34,35	INJ2	Ausgang, Einspritzung Kanal 2
36	frei	frei
37	FP1	Ausgang, Kraftstoffpumpenrelais

Tabelle B.2.: Modifizierte Pin Belegung DB37

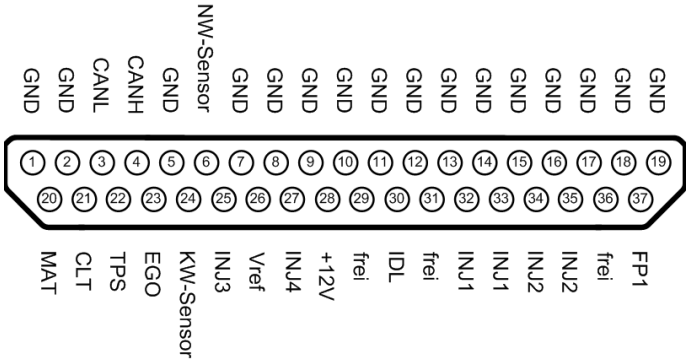

Abbildung B.4.: Modifizierte Pin-Belegung Sub-D37

B.3.2. Belegung Sub-D15

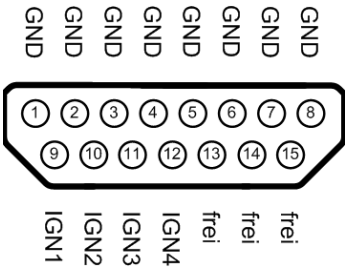

Abbildung B.5.: Modifizierte Pin-Belegung Sub-D15

B.3.3. Belegung Adapterboard

Belegung der DIP-Switches des Adapterboards:

DIP-Switch	Pin DB37	Beschreibung
A1	6	SPARE4 (NW-Sensor)
A2	5	SPARE3 (GND)
A3	4	SPARE2 (CANH)
A4	3	SPARE1 (CANL)
A5	20	MAT, Ansauglufttemperatur
A6	21	CLT, Kühlungstemperatur
A7	22	TPS, Drosselklappenpotentiometer
A8	23	EGO, Lambdasondensteuergerät
A9	24	KW-Sensor
A10	25	INJ3, Einspritzung Kanal 3
B1	26	Vref, +5V Referenzspannung
B2	27	INJ4, Einspritzung Kanal 4
B3	28	+12V, Spannungsversorgung ECU
B4	29	frei
B5	30	IDL, Ansteuerung Fast Idle Valve
B6	31	frei
B7	32,33	INJ1, Einspritzung Kanal 1
B8	34, 25	INJ2, Einspritzung Kanal 2
B9	36	frei
B10	37	FP1, Kraftstoffpumpenrelais

Tabelle B.3.: Belegung der DIP Switches des Adapterboards

C. Inhalt der CD

Dieser Arbeit liegt eine CD-ROM mit folgendem Inhalt bei:

Bachelorarbeit_Johannsen.pdf - Diese Arbeit im PDF-Format.

Datenblätter

- EV_14.pdf - Datenblatt der verwendeten Einspritzventile
- VB921ZVFI.pdf - Datenblatt der verwendeten Zündtreiber
- IXDD404SY.pdf - Datenblatt des Einspritzungstreibers der MegaSquirt
- IRFP240.pdf - Datenblatt der verwendeten Einspritz-Treiber
- MC9S12C128V1.pdf - Datenblatt des Motorola μController der MegaSquirt

Schaltpläne

- Adapterboard
  - AdapterBoard.brd - Layout des Adapterboards im Eagle-Format
  - AdapterBoard.shm - Schaltplan des Adapterboards im Eagle-Format
  - AdapterBoard.pdf - Schaltplan des Adapterboard als PDF
- MegaSquirt
  - Daughterboard - Schaltpläne des Daughterboards
  - MegaSquirt - Schaltpläne der Hauptplatine
- Zündendstufe
  - Zündung-Schaltplan.pdf - Schaltplan der Zündendstufe
  - Zündung-Schaltplan.shm - Schaltplan im Eagle-Format
- Einspritzungs-Treiberstufe
  - Einspritzungs-Treiberstufe.pdf - Schaltplan der Einspritzungsendstufe

  - Einspritzungs-Treiberstufe.shm - Schaltplan im Eagle-Format

Quellcode

- ms2extra - Quellcode der Modifizierten MegaSquirt2 extra
- RPM-Simulation - Quellcode der RPM-Simulation

Konfigurationsdateien

- megasquirt-ii.ini - Konfigurationsdatei für MegaTune
- MegaSquirt-Settings.msq - Konfiguration der MegaSquirt

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 7. Juni 2009	
Ort, Datum	Unterschrift