Inhaltsverzeichnis

1. Einleitung

Durchschnittlich 800 Mal täglich steht in Deutschland ein Gebäude in Flammen. Die meisten Brandopfer verbrennen nicht, sie ersticken durch den entstehenden giftigen Brandrauch. Der Rauch ist bei einem Brand die größte Gefahr für die Menschen, denn innerhalb von weniger als drei Minuten sinkt durch den entstehenden Brandrauch die Sichtweite so weit ab, dass betroffene Personen die Orientierung verlieren und sich nicht mehr in Sicherheit bringen können [5]. Daher ist es sehr wichtig dass:

- Ein Brand sehr schnell erkannt wird
- Durch Warnsysteme, Menschen gewarnt werden
- Der Rauch am Brandort sehr schnell abgezogen werden

Um diese Anforderungen an einem Gebäude umsetzen zu können, werden die sogenannten Rauch- und Wärmeabzug-Systeme eingesetzt. Diese können selbstständig einen Brand erkennen, melden, und Menschenleben retten. Weiterhin lassen sich Rauch und Wärmeabzugssysteme meist mit einer natürlichen Belüftung kombinieren. Das bietet sich auch an, weil Fenster, die auf einen Alarm reagieren, schon mit Fensterantrieben ausgerüstet sind. Da es nichts Wichtigeres gibt als Menschenleben zu retten, müssen die Antriebe, die diese Fenstern auf und zu schließen zuverlässig arbeiten. Es gibt mehrere Arten von Antrieben, im Bild 1.1 und Bild 1.2 ist ein Kettenantrieb und ein Zahnstangenantrieb zu sehen.

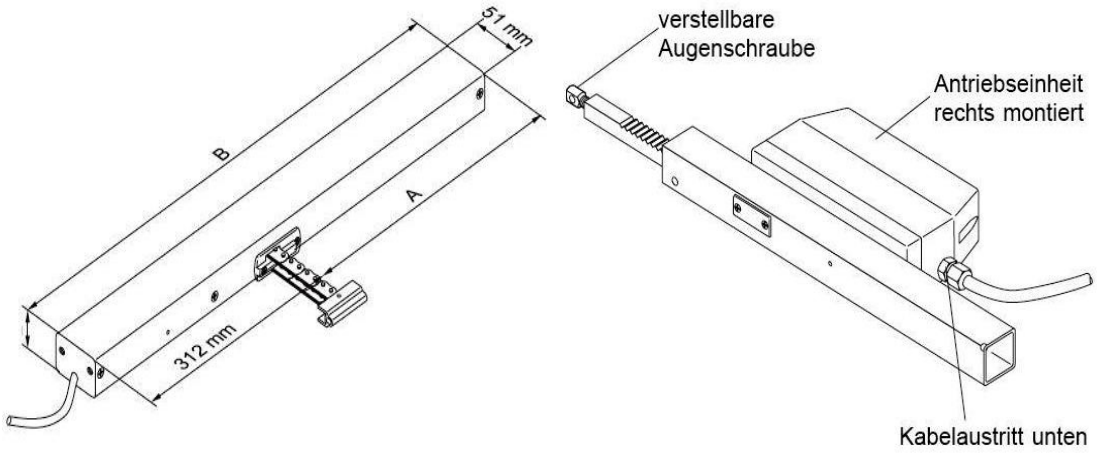


Bild 1.1 Ketten-und Zahnstangenantrieb

Vor der Installation der Fensterantriebe in das Gebäude, müssen diese aufwendige Tests durchlaufen. Während der verschiedenen Testphasen, müssen die Messgrößen Strom(I), Spannung(U), Weg(S), Kraft(F), Farben von LEDs[1] und deren Intensität mit einer hohen Genauigkeit gemessen werden. Auch der korrekte Ablauf der Prüfschritte zur Aufnahme der oben genannten Messgrößen ist sehr wichtig. Um alle diese Aufgaben schnell und sicher zu erledigen, wurde eine Prüfeinheit mit dem Namen „MPS2010“ entwickelt. Diese Prüfeinheit besteht aus fünf Modulen. Jedes Modul übernimmt eine definierte Aufgabe. Anhand des nachfolgenden Bildes (1.3) wird kurz dargestellt, aus welchen Modulen die Prüfeinheit zusammengesetzt ist und welche Funktionen sie besitzen.

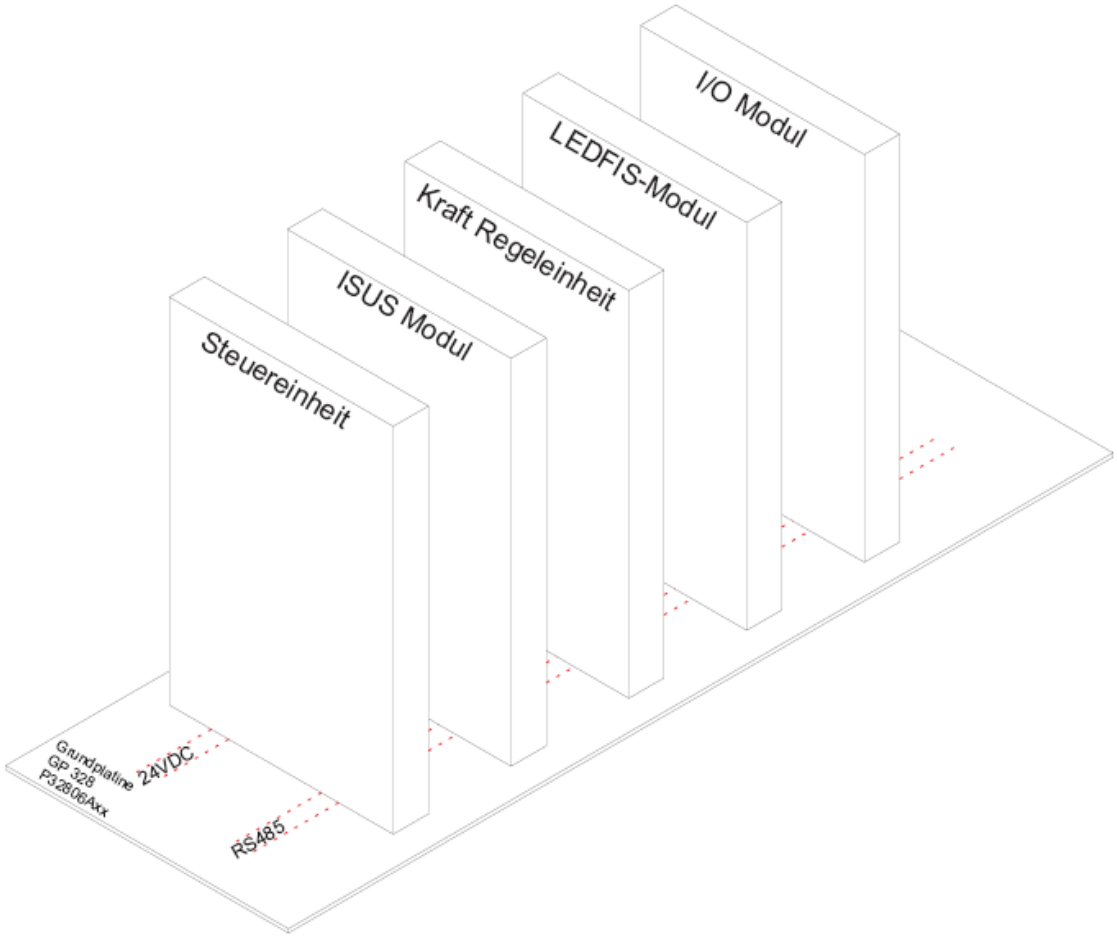


Bild 1.2 MPS2010

[1] LED steht für Light Emitting Diode

Das MPS2010 besteht aus folgenden Modulen:

- **Steureineheit :**

  Die Steuereinheit in diesem System ist als Master ausgeführt. Sie sendet den übrigen Modulen, welche als Slaves agieren, Befehle zu. Die Slaves können nicht untereinander kommunizieren. Sie führen die, von der Steuereineheit geforderten Aufgaben aus und warten anschließend auf neue Befehle.

- **ISUS-Modul :**

  Das ISUS-Modul ist ein multifunktionales Messmodul und hat unteranderem die Aufgabe Ströme, die durch die unterschiedlichen Motoren fließen zu messen. Weitere Funktionen des ISUS-Modul sind Spannungen und Wegmessungen mit einer hohen Auflösung durchzuführen.

- **Kraft Regeleinheit :**

  Die zu testenden Antriebe werden mit Ihrer Nennkraft belastet. Dazu ist die Reglung(Messung Istwert, Vergleich mit Sollwert, Korrektur)der Kraft sehr wichtig. Diese Aufgabe erledigt die Kraft Regeleinheit.

- **LEDFIS-Modul :**

  Das LEDFIS[2]- Modul hat die Aufgabe, LED-Farben und deren Intensität zu erkennen. Dies wird ausgenutzt um während der Prüfabläufe z.B Störungen zu erkennen.

- **I/O-Modul :**

  Das I/O-Modul hat die Aufgabe, eine Vielzahl von Ein- und Ausgangsschaltkombination zu realisieren.

  Die Kommunikation zwischen der Steuereinheit und allen anderen Modulen erfolget über den RS485-Bus.

[2] LEDFIS steht für Light Emitting Diode Farben Intensität System

2. Aufgabenstellung

Das MPS2010 ist eine Prüfeinheit die aus fünf verschiedenen Modulen, welche je nach Bedarf kombiniert werden können, besteht. Diese Prüfeinheit soll dazu genutzt werden, um die Endprüfung der verschiedenen Antriebe auf einem flexiblen Prüfgerät zu ermöglichen.

Die Aufgabe dieser Diplomarbeit ist die Entwicklung des ISUS-Moduls. Das ISUS-Modul muss in der Lage sein Ströme, die durch die Motoren der unterschiedlichen Antriebe fließen zu messen. Es sind Ströme im Bereich von 0A bis 6A spezifiziert. Darüberhinaus muss das ISUS-Modul die Spannungen, die für die Steuerung der Antriebe nötig sind messen können. Dazu werden je nach Antrieb Spannungen von 0V bis 30V spezifiziert. Zu den Aufgaben des ISUS-Moduls gehört auch eine Wegmessung. Dies ermöglicht während der Prüfung, z. B das Erkennen eines offenen oder geschossenen Fensters.

Die gemessenen Werte müssen an die Steuereinheit über einen RS485-Bus übermittelt werden. Ein einheitliches Protokoll muss hierfür festgelegt werden. Die Kommunikation zwischen ISUS und der Steuereinheit wird in Kapitel 6 Konzept und Implementierung ausführlicher beschrieben. Für das ISUS-Modul muss sowohl die Software als auch die Hardware entwickelt werden. Die Software wird in der Hochsprache C geschrieben. Für die Hardwarerealisierung muss ein Mikrokontroller eingesetzt werden. Eine Platine, welche den Mikrokontroller und die notwendigen Peripherie enthält ist ebenso zu entwickeln. Die Maßstäbe für die Platinengröße sind von der Firma D+H festgelegt und betragen 60mm x 90mm groß. Für die messenden Größen ist eine Fehlertoleranz von bis zu 2% erlaubt. Bei der Entwicklung des ISUS-Moduls soll je Messgröße je eine LED vorgesehen werden, welche aufleuchtet wenn ein Messwert an die Steuereinheit gesendet wird.

Die Schaltung für das Modul muss sicher gegen Störungen sein und der RS485-Bus muss sich in einem definierten Zustand befinden auch wenn keine Nachrichten am Bus anliegen.

3. Projektplan

In diesem Kapitel wird die, die in der Aufgabenstellung erwähnte Anforderungen des ISUS-Moduls in einzelne Teilaufgaben unterteilt. Diese werden nach einer zeitlichen Abschätzung in einem Projektplan eingetragen. Bei drei Monaten Bearbeitungszeit wurde folgender Zeitplan(Bild 3.1) erstellt.

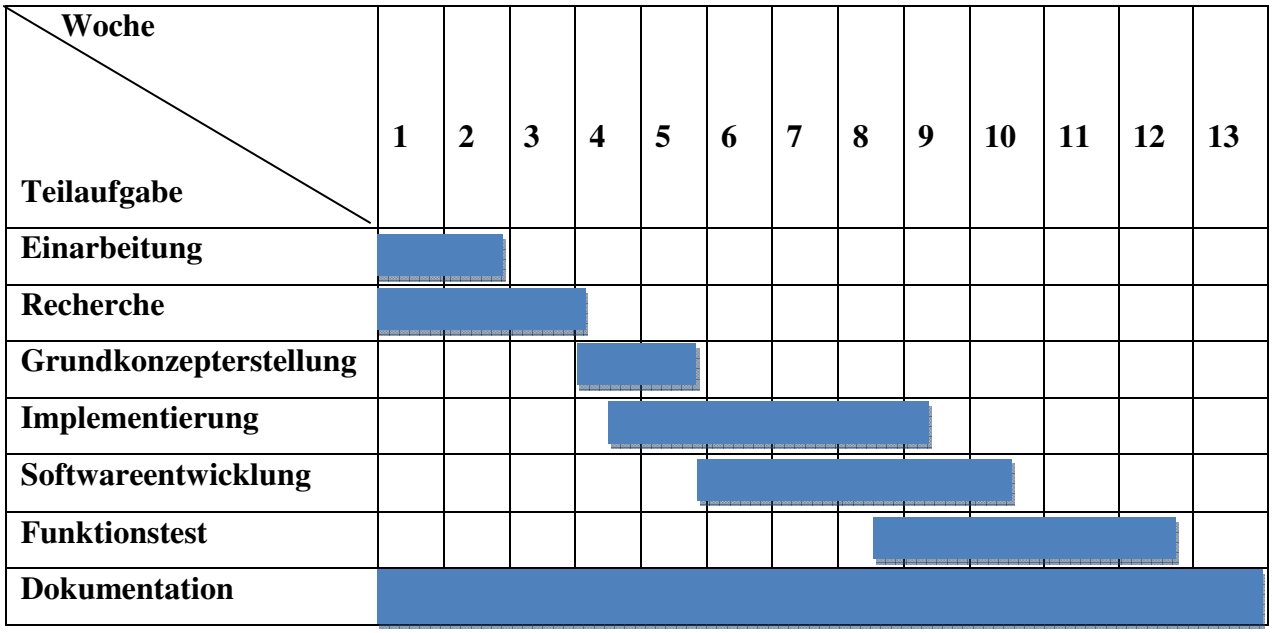


Bild 3.1 Projektplan

In der Einarbeitungsphase wurde der Umfang der Arbeit so untersucht, dass eine Einteilung das gesamte Projekt in Teilaufgaben und deren zeitlichen Abschätzung durchgeführt werden konnte. Während dieser Zeit wurde bereits recherchiert und Informationen zu den eventuell benötigten Hardwarekomponenten gesammelt. Zu dem Bereich Recherche gehörte auch eine Sammlung der Anforderungen an das gesamte System. In der Konzepterstellungsphase wurde festgelegt, welche Hardwarekomponenten eingesetzt werden. Auch die Fragen welcher Mikrokontroller und welche Programmiersprache eingesetzt werden soll, konnten geklärt werden. Zuerst wurde die Umsetzung der Wegmessung konzipiert und zwischendurch getestet. Danach erfolgte die Umsetzung der Strommessungen und der Spannungsmessungen. Während der Konzeptphase wurde bereits mit der Implementierung begonnen. Wie aus dem Projektplan ersichtlich wird, sind die Implementierungsphase und die Softwareentwicklungsphase nahezu parallel verlaufen. Dies liegt daran, dass diese Arbeit auf einem Mikrokontroller ausgelegt ist und dieser ohne Software nicht funktionsfähig ist. Für

jede Teilaufgabe des Projekts wurde ein Softwareteil geschrieben. Erst in der Testphase sind alle Hardwarekomponenten auf eine Platine zusammengefasst worden und die Software wurde aus allen Testprogrammen zusammengestellt. Da es wichtig war, jede gewonnene Erkenntnisse festzuhalten, wurde die Dokumentation der Arbeit parallel während des gesamten Projektablaufs durchgeführt.

4. Hardwarebeschreibung

Dieses Kapitel beschreibt die Hardwarekomponenten, die für diese Diplomarbeit eingesetzt wurden. Es werden die Eigenschaften des Mikrokontrollers und die Funktionsweisen der einzelnen Hardwarekomponenten erklärt.

4.1 µ- Controller PIC16F886

Der Mikrokontroller spielt in dieser Diplomarbeit die größte Rolle. Außer der Anforderung, der Rechenoperationen bearbeiten zu können, musste er auch Möglichkeiten für serielle Kommunikation, externe Interrupts sowie A/D- Umsetzung zur Verfügung stellen. Nach einiger Zeit Recherchen und Empfehlungen der Arbeitskollegen fiel die Wahl auf den PIC16F886. Der PIC 16F886 ist ein Produkt der Firma Microchip und gehört zu PIC16F-Familie. Der Controller besitzt 28-Pins und bietet dadurch eine Vielzahl von peripheren und Anschlussmöglichkeiten.

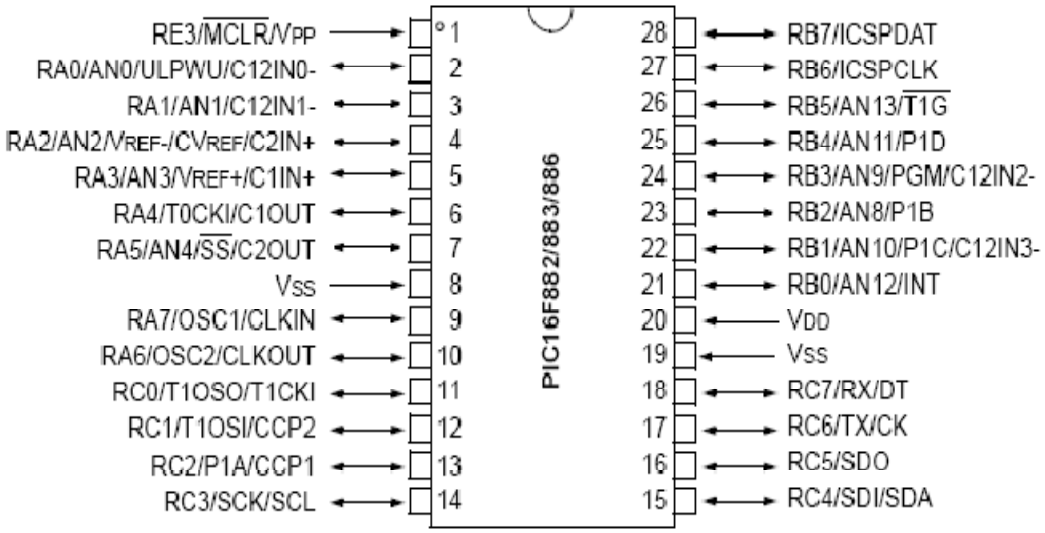


Bild 4.1.1 Pinbelegung des PIC16F886

Nachstehend sind die wichtigsten Eigenschaften des Controllers zusammengetragen:

- 8-Bit RISC Architektur
- PIC16F Serie
- Memory Size, Flash : 14KB
- EEPROM-Größe : 256Byte
- Speichergröße RAM : 368Byte

- 24 I/O Leitungen mit einstellbarer Datenrichtung (im Multiplex mit anderen Funktionen)
- Analog/Digital-Umsetzer mit 11 Kanälen und bis zu10-Bit Auflösung
- 3 Timers (2x 8-Bit und 1x 16-Bit)
- 2 Pulsweiten-Modulation(PWM) mit einer Auflösung bis zu 10-Bit
- Bis zu 20MHz Taktfrequenz
- Schnittstellentyp : EUSART, I2C, SPI
- Temperatur Arbeitsbereich : -40°C bis +85°C
- Betriebsspannung +2V bis 5.5V

Im nachstehenden Bild 4.1.2 ist ein vereinfachtes Blockschaltbild der Controllerarchitektur zu sehen.

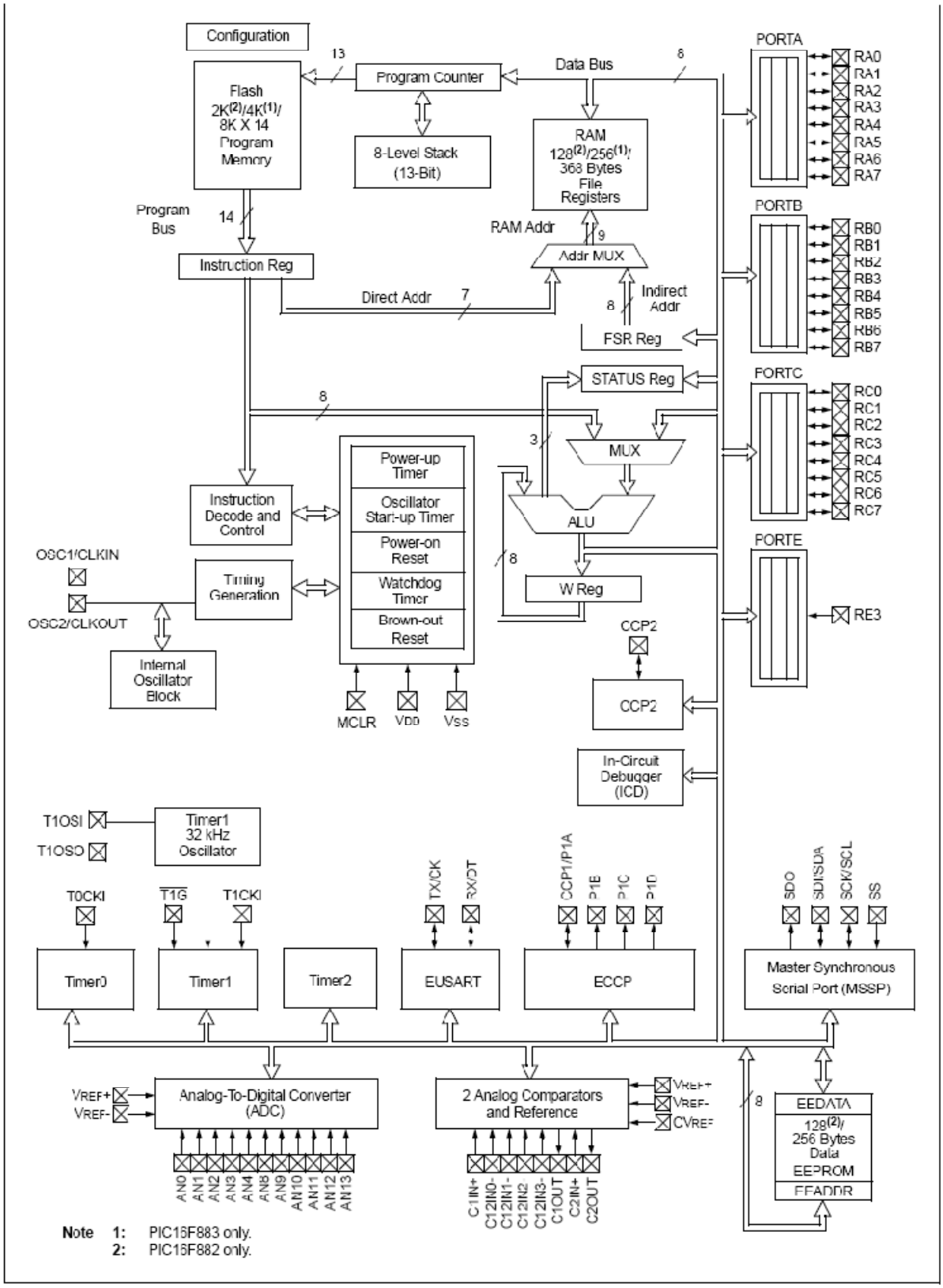


Bild 4.1.2 Blockschaltbild der Controllerarchitektur

4.2 Peripherie- Register und I/O Ports

Der PIC16F886 besitzt insgesamt 3 I/O-Ports (A, B, C). Für alle 3 Ports existieren Datenrichtungsregister (Data Direction Register) um die einzelnen Pins des jeweiligen Ports als Ein-bzw. Ausgang zu definieren. Schreibt man in das Datenrichtungsregister eines Ports eine „1“, wird der dazugehörige Pin als Eingang definiert. Alle Ports des Controllers sind mit anderen Funktionen mehrfach belegt, als Beispiel Port A kann neben der Port-I/O-Funktion auch als Analoge Eingang benutzt werden.

4.3 Analog/Digital- Umsetzer

Der ADU spielt eine sehr große Rolle bei der Entwicklung des ISUS-Moduls. Denn mit dessen Hilfe, wurden die Ströme und Spannungen gemessen. Bild 4.3.1 zeigt den schematischen Aufbau des ADU in der PIC16F- Familie. Wie aus dem Bild zu entnehmen ist, besteht die Eingangsschaltung des ADU aus einem Multiplexer mit elf (PIC16F886 besitzt nur elf Kanäle) analogen Eingangskanälen. Der ADU hat eine Auflösung von10-Bit und arbeitet nach der Methode der sukzessiven Approximation

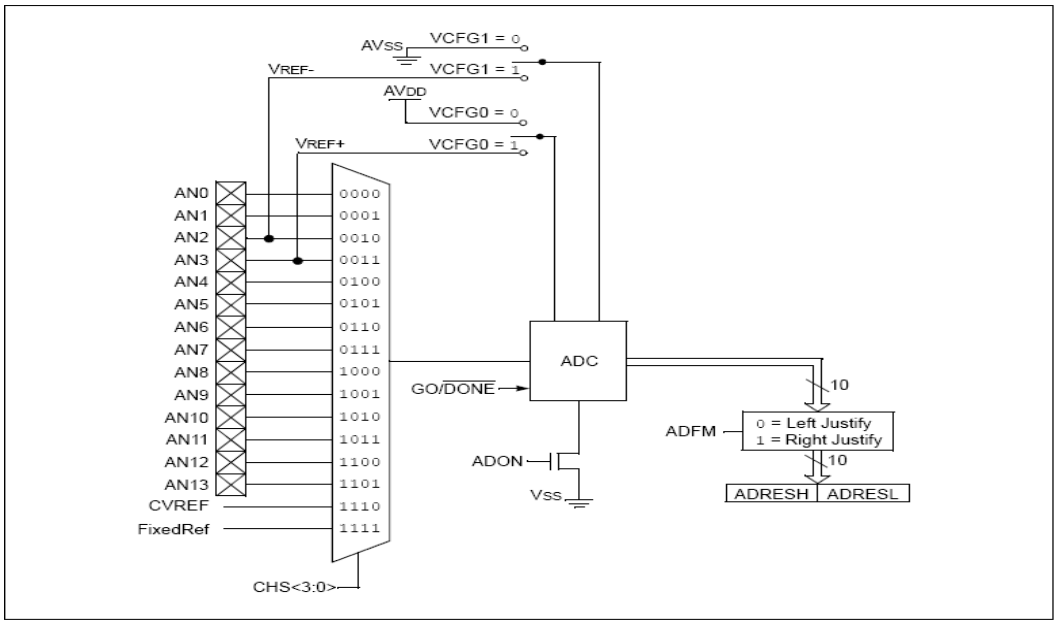
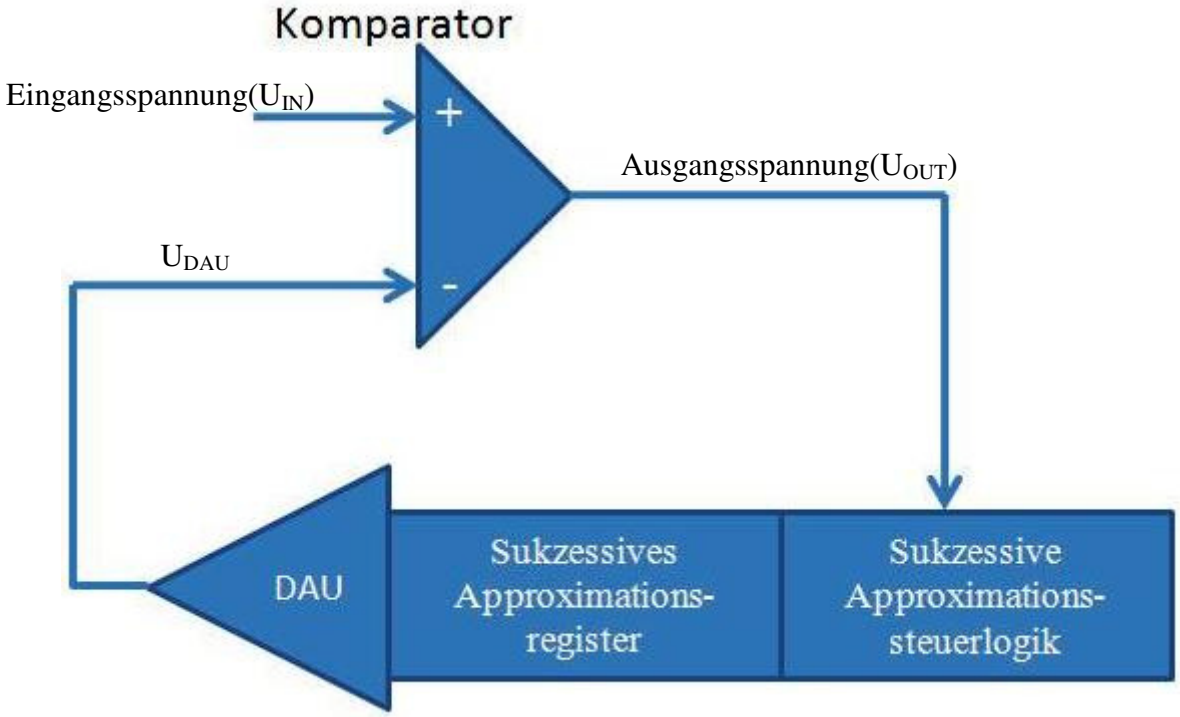


Bild 4.3.1 Blockschaltbild des ADU(PIC16F Serie)

Das Prinzip der sukzessiven Approximation wird anhand des Bilds 4.3.2 erläutert. Die ADU-Einheit enthält einen Digital/Analog-Umsetzer, welcher eine Vergleichsspannung (U_{DAU}) zur Verfügung stellt. Die Ausgangsspannung des Komparators wird der sukzessiven Approximationssteuerlogik zugeführt, die wiederum die sukzessiven Approximationsregister

steuert [1]. Das Eingangssignal (U_{IN}) wird somit in n Schritten digitalisiert. In jedem Schritt wird die Eingangsspannung (U_{IN}) mit einer Referenzspannung, die durch den DAU erzeugt wird verglichen. Je nachdem ob, (U_{IN}) größer oder kleiner als die Spannung (U_{DAU}) ist, wird die Referenzspannung um die halbe Schrittweite des letzten Schritts nach oben oder nach unten verändert (siehe Bild 4.3.3). Dadurch nährt sich die Spannung (U_{DAU}) schrittweise der Eingangsspannung (U_{IN}). Wenn das letzte Bit des DAU gesetzt ist, entspricht der Wert des DAU dem Eingangsspannungswert und die Umsetzung ist abgeschlossen. Bild 4.3.3 Veranschaulicht den Vergleich der (U_{DAU}) mit dem Eingangssignal (U_{IN}). Die Genauigkeit des Ergebnisses bei einem ADU, der nach diesem Prinzip arbeitet, hängt sowohl von der Auflösung des ADU als auch von der Genauigkeit des Komparators ab [7].



Bilds 4.3.2 sukzessive Approximation(1)

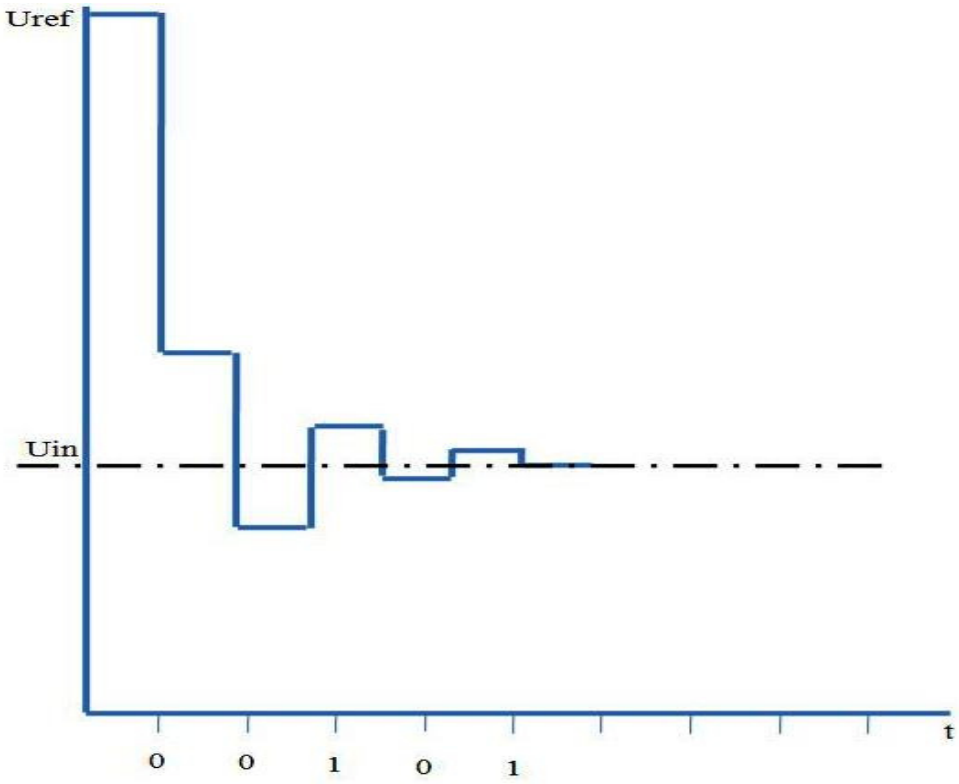


Bild 4.3.3 sukzessive Approximation(2)

Register, die für die A/D- Umsetzung zuständig sind:

- **ADCON0** : A/D CONTROL REGISTER 0
- **ADCON1**: A/D CONTROL REGISTER 1
- **ADRESH**: ADC RESULT REGISTER HIGH
- **ADRESL**: ADC RESULT REGISTER LOW

Aufbau das ADCON0 Registers:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit2	Bit 1	Bit 0
ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/DONE	ADON

Bild 4.3.4 ADCON0 Register

Der ADU benötigt für die Umwandlung einen Arbeitstakt. Dieser kann mit einem Frequenzteiler aus dem Takt des Kontrollers abgeleitet werden. Der Controller-Takt kann zwischen wenigen Kiloherzt und 20 MHz liegen. Deshalb muss je nach verwendetem

Controller-Takt ein Frequenzverhältnis zwischen ADU-Takt und Controller-Takt eingestellt sein. Dies erfolgt mit Bit 7 und Bit 6 des ADCON0[3]-Registers.

Bit 7-6 ADCS<1:0>: A/D Conversion Clock Select bits

00	FOSC/2
01	FOSC/8
10	FOSC/32
11	FRC (clock derived from a dedicated internal oscillator = 500 kHz max)

Bild 4.3.5 ADU-Takt Auswahl

Mit den Bits 5 bis 3 des ADCON0-Registers verbindet man den benötigten Eingangskanal mit dem ADU.

CHS<3:0>: Analog Channel Select bits

0000	AN0
0001	AN1
0010	AN2
0011	AN3
0100	AN4
0101	AN5
0110	AN6
0111	AN7
1000	AN8
1001	AN9
1010	AN10

Bild 4.3.6 ADU- Kanalauswahl

[3] ADCON0 steht für Analog/Digital Control Register 0

Mit dem setzen von Bit 1 des ADCON0-Registers wird der Analog/Digital-Umsetzung. Nach erfolgter Umsetzung wird das Bit wieder auf 0 gesetzt. Mit dem setzten von Bit 0 wird der ADU in Betrieb genommen.

Aufbau das ADCON1 Registers:

Bild 4.3.7 ADCON1 Registers

Das Ergebnis des Analog/Digital-Umsetzers ist 10-Bit lang und wird in den Registern ADRESH[4] und ADRESL[5] abgelegt (Bild 4.3.6). Die beiden Register sind zusammen 16-Bit groß. Mit dem setzen bzw. löschen des Bit 7 des ADCON1[6]-Registers wird bestimmt, ob das Ergebnis linksbündig oder rechtsbündig in den beiden Registern abgelegt wird. Die ungenutzten Stellen werden mit Nullen ausgefüllt.

Mit Bit 5 und Bit 4 wird dem ADU mitgeteilt, ob für die Messung eine externe Referenzspannung an den dazugehörigen Pins anliegt oder, ob die Betriebsspannung als Referenzspannung verwendet werden soll.

Bit6, Bit3, Bit2, Bit1, Bit0 haben keine Funktion.

ADRESH und ADRESL

Wie bereits erwähnt, wird das Ergebnis des Analog/Digital-Umsetzers in diese beiden Register geschrieben.

[4] ADRESH steht für ADC Result Register High
[5] ADRESL steht für ADC Result Register Low
[6] ADCON1 steht für Analog/Digital Control Register 1

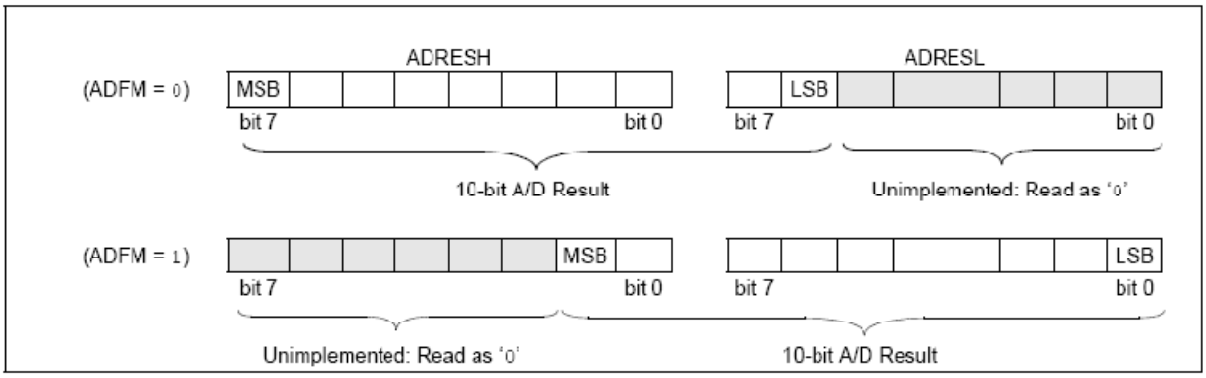


Bild 4.3.8 Ergebnisformate des ADU

4.4 Erweiterte Serielle Schnittstelle EUSART

Die Serielle Schnittstelle des Controllers heißt EUSART[7]

Eine USART-Schnittstelle dient zum Senden und Empfangen von Daten über Datenleitungen.

Für die Steuerung des EUSART sind folgende Register zuständig:

- **TXSTA** : (Transmit Status And Control Register)
- **RXSTA** : (Receive Status And Control Register)
- **SPBRG** : (Serial Port Buad Rate Register)
- **TXREG** : (Transmit Register)
- **RCREG** : (Receive Register)

Das **TXSTA**[8]-Register ist für das Senden mit der EUSART-Schnittstelle zuständig.

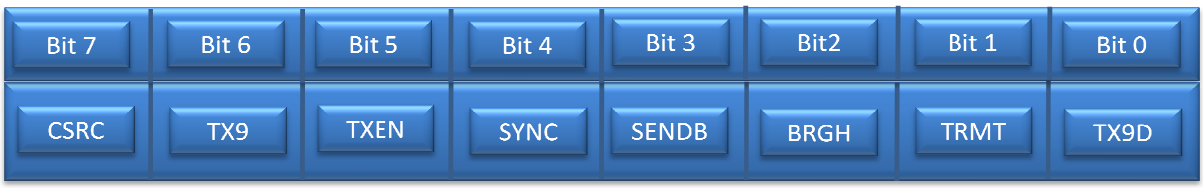


Bild 4.4.1 TXSTA-Register

Bedeutung der Einzelnen Bits:

CSRC : Clock Source Select Bit, Im asynchronen Modus irrelevant

TX9 : Transmit Enable

[7] EUSART = ENHANCED UNIVERSAL SYNCHRONUS ASYNCHRONOUS RECIVER TRANSMITTER

[8] TXSTA = Transmit status and Control Register

- 1 : Neun Bit werden gesendet
- 0 : Acht Bit werden gesendet

TXEN :Transmit Enable Bit

- 1 : Sender aktiviert
- 0 : Sender deaktiviert

SYNC : EUSART Mode Select Bit

- 1 : asynchrone Modus
- 0 : synchrone Modus

SENDB : Hier handelt es sich um ein „Break Character Feature“, welches von einem LIN-Bus angefordert wird. Im synchronen Modus irrelevant.

BRGH : High Baud Rate Select bit, im synchronen Modus irrelevant

- 1 : High Speed
- 0 : Low Speed

TRMT :Transmit Shift Register Status bit

- 1 : Senderegister leer
- 0 : Senderegister voll.

TX9D **:**Das neunte Bit, kann ein Adresse-/ Daten- oder Parität- Bit sein.

Das **RXSTA**[9]-Register ist für das Empfangen mit der EUSART-Schnittstelle zuständig.

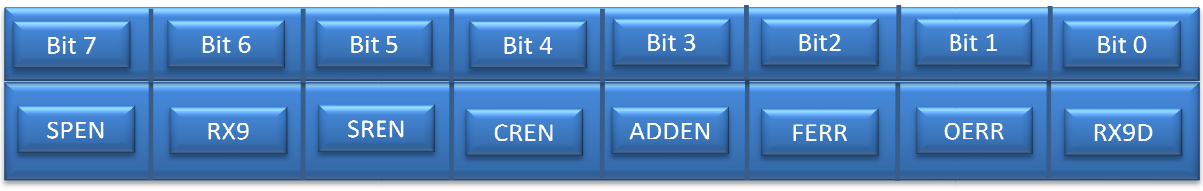


Bild 4.4.2 RXSTA-Register

[9] RXSTA = Receive status and Control Register

Bedeutung der Einzelnen Bits:

SPEN : Serial Port Enable bit.

- 1: RX -und TX-Pins sind als serieller Port-Pins konfiguriert.
- 0 : serielle Port ausgeschaltet

RX9 : 9-bit Receive Enable bit

- 1 : ein9-BitDatenbitempfangen
- 0 : ein 8-Bit Datenwort empfangen.

SREN : Signal Receive Enable bit. Im Asynchronen Modus allgemein und Synchronen Modus als Slave ist dieses Bit irrelevant. Im Synchronen Modus als Master wird bei:

- 1 : Der kontinuierliche Empfang aktiviert
- 0 : Der kontinuierliche Empfang deaktiviert

ADDEN : Address Detect Enable bit.

FERR : Framing Error bit.

- 1: Übertragungsrahmenfehler wurde entdeckt
- 0: kein Fehler entdeckt.

OERR : Overrun Error bit

- 1: Ein Überlauffehler wurde entdeckt
- 0: Kein Überlauffehler entdeckt.

TX9D : Das neunte Bit, kann ein Adresse-/ Daten- oder Parität- Bit sein.

4.5 RS485- BUS

In diesem Kapitel wird der RS485-Bus detailliert erklärt. Es wird genauer auf die Funktionsweise des Busses eingegangen, die Vor- und Nachteile erläutert und Unterschiede zur anderen bekannten Bussystemen aufgezeigt.

Die RS485-Schnittstelle ist für die serielle Hochgeschwindigkeitsdatendatenübertragung über große Entfernungen entwickelt worden. RS485 ist eine differenzielle Schnittstelle, die für den Betrieb mindestens zwei Datenleitungen benötigt, was bedeutet, dass für jedes Signal zwei Leitungen benötigt werden. Eine Leitung überträgt das Signal unverändert, wohingegen die andere Leitung das Signal invertiert überträgt. Wenn z.B. „1“ übertragen werden soll, ist hat die eine Leitung +5V, die andere Leitung 0V. Wenn eine „0“ übertragen werden soll, ist es umgekehrt. Die invertierte Leitung wird in der Regel durch den Buchstaben „A“ gekennzeichnet. Während die nicht invertierte Leitung mit „B“ bezeichnet wird. Der Empfänger wertet die Differenz zwischen beiden Leitungen und gibt das Original Signal aus. Die Leitungslänge beim RS485 ist von der Signaldämpfung abhängig. Da diese Frequenzabhängig ist, ist die Leitungslänge auch von der Übertragungsrate abhängig. Es ist festgelegt, dass die Differenzspannung zwischen den Leitungen A und B (U_{AB}) +0.2V für eine binäre 0 und -0.2V für eine binäre 1erreicht werden muss [6].

Der RS485 kann überall eingesetzt werden, wo größere Datenmengen über längere Strecken übertragen werden und wo mehrere Gräten miteinander kommunizieren müssen.

Der RS485-Bus kann sowohl „Half Duplex“ als auch „Full Duplex „betrieben werden. Half Duplex bedeutet dass, eine Übertragung immer abwechselnd in die eine Richtung oder in die andere Richtung erfolgt. Es können also keine Daten gleichzeitig gesendet und empfangen werden. Für diese Anwendung genügt zwischen den Teilnehmer auf dem Bus eine zweidraht Verbindung. Für die Full Duplex Variante, bedeutet dies, dass Daten gleichzeitig gesendet und empfangen werden können, und hierfür vier Leitungen benötigt werden.

Vorteile des RS485-Busses

- Bis zu 128 Teilnehmer können an dem Bus angeschlossen werden
- Bis zu 500m Leitungslänge möglich
- Unempfindlich gegen Störungen
- Kein vorgegebenen Protokoll wie bei anderem Bussystemen
- Hohe Datenübertragungsrate
- Geringer Hardwareaufwand, da außer einem Treiber, keine weiter Bauteile benötigt werden
- Geringe Kosten und Platz bedarf

Nachteile des RS485-Busses

- Mindestens zwei Leitungen für ein Datenkanal werden benötigt
- Mit zwei Leitungen ist nur eine Half Duplex- Anwendung möglich
- Für eine Full Duplex- Anwendung werden vier Leitungen benötigt
- Ein Steuerungssignal für die Datenrichtung ist notwendig

Der Mikrocontroller verfügt wie es im Kapitel 4.4 erwähnt ist, eine EUSART Schnittstelle, über die die Kommunikation abgewickelt wird. Der Controller besitzt jedoch keine RS485-Schnittstelle. Dieses Problem lässt sich durch einen geeigneten Treiberbaustein in Form eines zusätzlichen ICs beseitigen. Die Firma D+H verwendet dafür den Treiberbaustein DS36C278.

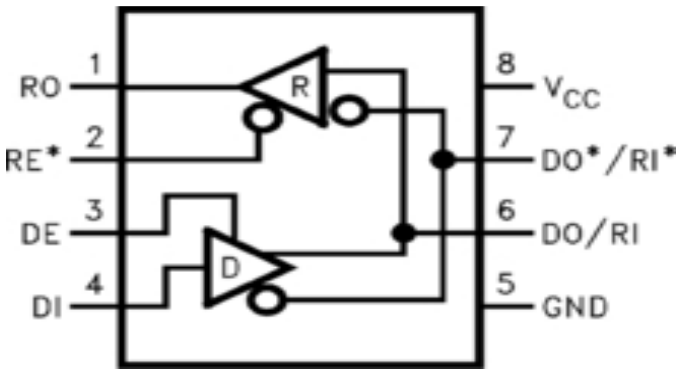


Bild 4.7.1 DS36C278

Im Bild 4.7.1 ist der schematische Aufbau des RS485 Treibers dargestellt.

- RE* und DE sind die enable Leitungen für Sender und Empfänger.
- RO ist der Dateneingang zum Controller
- DI ist der Daten Ausgang vom Controller
- GND ist die Masseleitung
- Vcc ist die Versorgungsspannung. Diese soll zwischen +4.75V und +5.25V betragen.
- DO/RI ist die Datenleitung A.
- DO*/DI* ist die Datenleitung B.

Im Kapitel 6. Konzept und Implementierung wird die Beschaltung des Treibers mit dem Mikrocontroller aufgezeigt. In Kapitel 5. Softwarebeschreibung wird die Programmierung der Schnittstelle ausführlich erklärt.

4.6 Seilzugsensor

Wie es im Kapitel 2. Aufgabenstellung erwähnt ist, gehört zur Aufgabe des ISUS-Moduls eine Wegmessung. Dies erfolgt mit Hilfe einen sogenannten Seilzugsensors. Dieser Sensor wandelt lineare Bewegung durch Aus- und Einzug eines Seiles in Drehbewegungen um. Diese werden von den nachgeschalteten Drehgebern in entsprechend elektrische Signale umgewandelt.

Bild 4.6 Seilzugsensor

Mechanische Kennwerte des Seilzugsensors laut Hersteller:

Messbereich:	Bis 2000mm
Absolutgenauigkeit:	±0,1 % des Messwertes
Wiederholgenauigkeit	±0,15 mm je Anfahrtrichtung
Auflösung (inkremental):	0,1 mm (Standardgeber) mit 1000 Imp./Umdr.
Verfahrgeschwindigkeit:	max. 800 mm/s
erforderl. Auszugskraft:	ca. 10 N (am Seil)
Werkstoffe:	Gehäuse: Kunststoff verstärkt Seil: Stahl, rostfrei ø 0,45 mm
Gewicht:	ca. 0,210 kg

Bild 4.6 Mechanische Kennwerte des Seilzugsensors

4.7 ICSP (In Circuit Serial Programming)

Es ist möglich den PIC mit Hilfe einer seriellen Datenübertragung zu programmieren. Dazu benötig man folgende fünf Leitungen:

1. Eine Taktleitung
2. Eine Datenleitung
3. Eine Masseleitung
4. Eine 5V Betriebsspannungsleitung
5. Eine 15V Programmierspannungsleitung

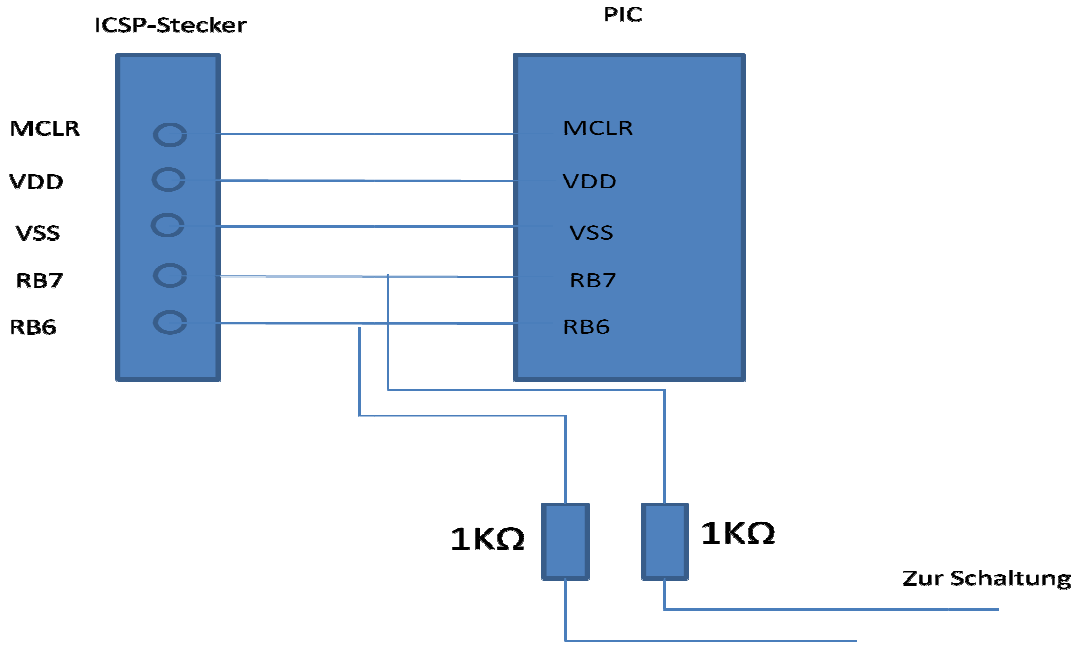


Bild 4.5 ICSP-Verbindung

Der Vorteil der ICSP-Verbindung besteht darin, dass der Controller programmiert werden kann nach dem er bereits in seiner Anwendungsschaltung eingebaut ist [8]. Die zwei Leitungen RB7 und RB6 aus dem Bild 4.5 entsprechen der Takt- und der Daten-Leitung, welche für die ICSP-Schnittstelle vorgesehen sind. Es ist möglich die zwei Pins des Controllers für andere Zwecke zu benutzen. Es ist wichtig dabei zu beachten, dass die beiden Leitungen über jeweils einen Kilo Ohm Widerstand vom Rest der Schaltung abgetrennt sind. Werden die zwei Leitungen ohne Widerstände für andere Zwecke eingesetzt, ist das Programmieren des Mikrocontrollers auf diese Art und Weise nicht möglich.

5. Softwarebeschreibung

Die Software wurde in der Programmiersprache C mit dem „MPLAB[10]-“Paket der Firma Microchip entwickelt. Das Paket beinhaltet einen Editor, Debugger, dazugehörige Dokumentation und ein Programmiergerät. In diesem Kapitel wird das C-Programm beschrieben. Es wird erklärt welche Funktionen vorhanden und aus welchen Grund diese notwendig sind. Es wird darauf verzichtet, die einzelnen Teile des Programms detailliert zu erläutern. Im Anhang befindet sich eine kommentierte Version des Programmes. Die gesamte Software besteht aus zwei Interruptroutinen, Hauptprogramm und vier Funktionen. In den folgenden beiden Bildern (Bild 5.1 und 5.2) ist der Flussplan abgebildet, welcher den Programmablauf in vereinfachter Form aufzeigt [4].

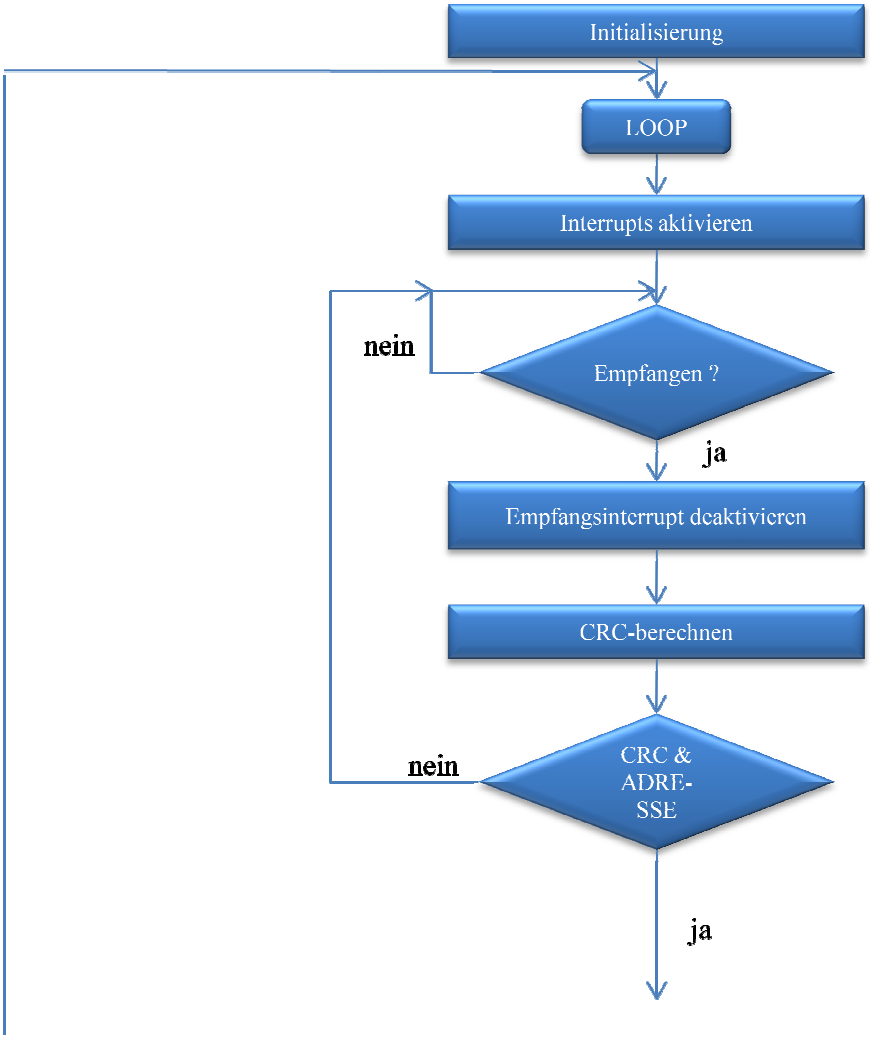


Bild 5.1 Flussplan der gesamten Software (1)

[10] MPLAB ist ein freies Tool von der Firma Microchip und kann zum programmieren der PIC benutzt werden

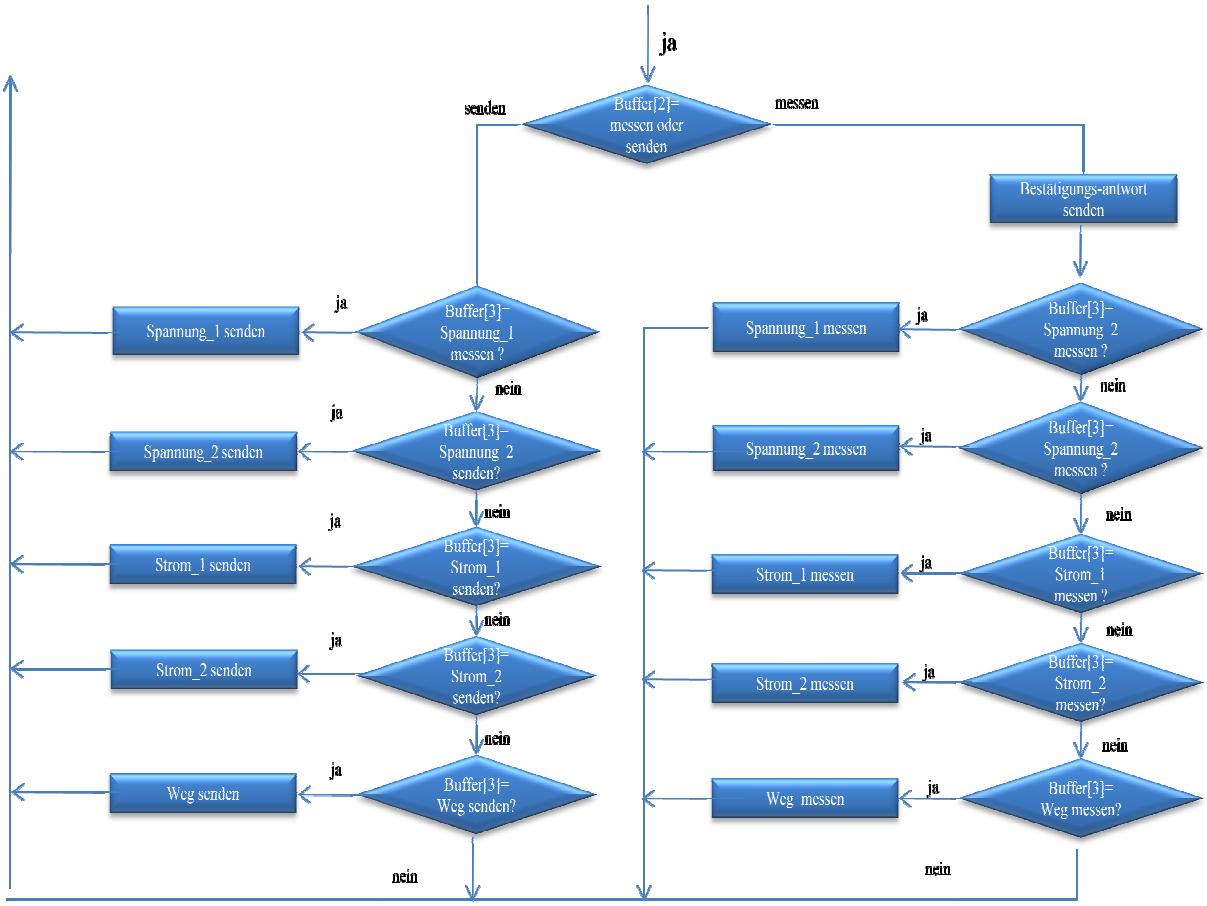


Bild 5.2 Flussplan der gesamten Software (2)

In dem Initialisierungsblock werden die Adressen der Mikrokontrollerports und die Frequenz mit der der Controller betrieben wird, festgelegt. Die serielle Schnittstelle, globale Variablen und die Funktionsprototypen werden deklariert. Die Interrupteinstellungen werden durchgeführt und die Interrupts werden freigeschaltet. Sobald die Ausgangssignale des Seilzugsensors sich ändern, wird die Interrupt-Service-Routine, die für die Wegmessung zuständig ist, gestartet. Dies bedeutet, dass das ISUS-Modul auch ohne Anforderung von der Steuereinheit eine Wegmessung ausführt. Wenn die Signale des Seilzugsensors sich nicht ändern, wartet das ISUS-Modul auf Anfragen von der Steuereinheit.

void main (void)

Im Hauptprogramm wird in einer Schleife solange gewartet bis ein Paket von der Steuereinheit empfangen worden ist. Nach dem Empfangen eines Paketes, deaktiviert das Programm den Interrupt, so dass keine weiteren Pakete empfangen werden können. Dies hat den Vorteil, dass das ISUS-Modul während der Bearbeitung des Pakets nicht gestört wird. Das ISUS-Modul wertet das empfangene Paket aus und erledigt dementsprechend von ihm geforderte Aufgabe. Nach dem das ISUS-Modul die von ihm geforderte Aufgabe bearbeitet hat, aktiviert es den Empfangsinterrupt wieder, so dass weiter Pakete empfangen werden können. Das Modul wartet dann auf neue Aufgaben. Für die reibungslose Kommunikation zwischen der Steuereinheit und dem ISUS-Modul wurde ein Kommunikationsprotokoll für den RS485-Bus festgelegt.

Das ISUS-Modul empfängt immer fünf Byte von der Steuereinheit und sendet je nach Aufgabe entweder vier oder sieben Byte zurück. Anhand der folgenden beiden Bilder(Bild 5.3 und Bild 5.4) wird das Anfrage- bzw. Antwortprotokoll detailliert erklärt. Daraus ergibt sich auch, wie das ISUS-Modul die Anfragen der Steuereinheit auswertet und bearbeitet.

Anfrageprotokoll (Steuereinheit (Master))

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Vorlauf	Adresse	Befehl	Auftrag	CRC

Bild 5.3 Anfrageprotokoll

Byte 1: Ist der Vorlauf (0x64) und sehr entscheidend. Es werden die weiteren Bytes erst empfangen, wenn der Wert des Vorlaufs mit dem zu erwartenden Wert (0x64) übereinstimmt.

Byte 2: Ist die Adresse des ISUS-Moduls und hat den Wert (0x11). Alle Module der Prüfeinheit empfangen alle Pakete der Steuereinheit. Nur das Modul, das seine eigene Adresse empfängt, wird angesprochen. Alle anderen Module sind hochohmig und blockieren den Bus nicht.

Byte 3: Ist ein Befehlsbyte und hat entweder den Wert (0x0) oder (0x1).

- **(0x0):** Das ISUS-Modul sendet sofort eine Bestätigungsantwort, was bedeutet, dass das Modul die Anfrage empfangen hat und die Befehle ausführen kann.
- **(0x1):** Das Modul soll einen bereits gemessenen Wert an die Steuereinheit senden.

Byte 4: Der Wert des vierten Byte legt fest, welche Größe gemessen werden soll oder welcher Messwert an die Steuereinheit gesendet werden soll. Der zu sendende Wert hängt vom Inhalt von Byte 3 ab.

- **0x0:** bedeutet erste Spannung messen oder senden
- **0x1:** bedeutet zweite Spannung messen oder senden
- **0x2:** bedeutet ersten Strom messen oder senden
- **0x3:** bedeutet zweiten Strom messen oder senden
- **0x4:** bedeutet den Weg messen oder senden

Byte 5: Dieses Byte enthält die CRC-Summe[11]: Eine Erklärung der CRC-Summe ist am Ende dieses Kapitels zu finden.

Antwortprotokoll (ISUS-Modul (Slave))

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Vorlauf	Adresse	Was gemessen wurde	MSB	LSB	Fehlercode	CRC

Bild 5.4 Antwortprotokoll

Byte 1: Vorlauf (0x64)

Byte 2: Adresse (0x11)

Byte 3: Der Wert von dem dritten Byte legt fest, welcher Größe gemessen wurde und kann je nach gemessenem Wert, verschiedene Werte annehmen. Im Folgenden sind diese ausgeführt.

- **0x0:** bedeutet erste Spannung wurde gemessen
- **0x1:** bedeutet zweite Spannung wurde gemessen
- **0x2:** bedeutet erste Strom wurde gemessen
- **0x3:** bedeutet zweite Strom wurde gemessen

[11] CRC = cyclic redundancy check

- **0x4:** bedeutet Weg wurde gemessen

Byte 4: Die gemessenen Spannungen, Ströme und Wege werden in mV, mA und in mm an der Steuereinheit gesendet. Da zu erwarten ist, dass die gemessenen Werte den darstellbaren Bereich eines Bytes überschreiten, dieser liegt zwischen 0 und 255, wird der Messwert in zwei Bytes übertragen. Das vierte Byte beinhaltet das höchstwertigste Bit des gemessenen Wertes (MSB)

Byte 5: Das fünfte Byte beinhaltet das niederwertigste Bit des gemessenen Wertes (LSB)

Byte 6: Der Wert des sechsten Bytes ist ein Fehlercode und kann je nach Fehler, verschiedene Werte annehmen. Im Folgenden sind diese ausgeführt.

- **(0x0):** Prüfablauf wurde fehlerfrei durchgeführt
- **(0x40):** Steuereinheit erwartet die erste Spannung, obwohl sie die Messung der ersten Spannung nicht gefordert hat oder die Messung nicht ausgeführt werden konnte.
- **(0x41):** Steuereinheit erwartet die zweite Spannung, obwohl Sie die Messung der zweiten Spannung nicht gefordert hat oder die Messung nicht ausgeführt werden konnte.
- **(0x42):** Steuereinheit erwartet den ersten Strom, obwohl Sie die Messung des ersten Stroms nicht gefordert hat oder die Messung nicht ausgeführt werden konnte.
- **(0x43):** Steuereinheit erwartet den zweiten Strom, obwohl Sie die Messung des zweiten Stroms nicht gefordert hat oder die Messung des zweiten Stroms nicht ausgeführt werden konnte.
- **(0x44):** Steuereinheit möchte die Strecke(Weg) übermittelt bekommen, obwohl sie die Messung nicht gefordert hat oder die Messung nicht ausgeführt werden konnte.
- **(0x45):** Erste Spannung ist größer als 30V.
- **(0x46):** Zweite Spannung ist größer als 30V.

Byte 7: CRC

Weg/Richtungs- Bestimmung

#INT_EXT

void ext_interrupthnd_seilzug_pulse_zaehlen(void)

Eine 16Bit-große globale Variable „Volataile unsigned int16 SEIL_PULSE“ wurde deklariert. In dieser Variable werden die Anzahl der Pulse des Seilzugsensors gespeichert. In der Interrupt-Service-Routine werden die Ausgangsignale des Seilzugsensors eingelesen und ausgewertet. Die zwei digitalen Rechteck- Ausgangssignale (A , B) des Seilzugsensors sind um 90° verschoben. Die Interrupt-Service-Routine wird ausgeführt, sobald das Seil des Sensors eingezogen bzw. ausgezogen wird.

Die Interrupt-Service-Routine wird sowohl bei der positiven als auch bei der negativen Flanke (des Signals A) ausgeführt, somit sind zwei Fälle für die Auswertung zu unterscheiden (siehe Bild 5.5 Wegmessungsinterrupt).

- Fall 1: Das Signal A weist eine 1 auf

Das Signal B ist ebenfalls 1, so wird die Variable SEIL_PULSE inkrementiert[12]. Weist Signal B eine 0 auf, so wird die Variable SEIL_PULSE dekrementiert[13]

- Fall 2: Das Signal A weist eine 0 auf

Das Signal B = 1 somit wird diesmal die Variable SEIL_PULSE dekrementiert und bei Signal B = 0 die Variable SEIL_PULS inkrementiert.

Der Wert der Variable SEIL_PULSE entspricht nun der Anzahl der Pulse des Seilzugsensors. Wie bereits im Kapitel 4.6 Seilzugsensor erwähnt, gibt der Seilzugsensor 10 Pulse pro Millimeter aus. Teilt man nun die Variable SEIL_Pulse durch 10, so erhält man die Streck, die das Seils zurückgelegt hat.

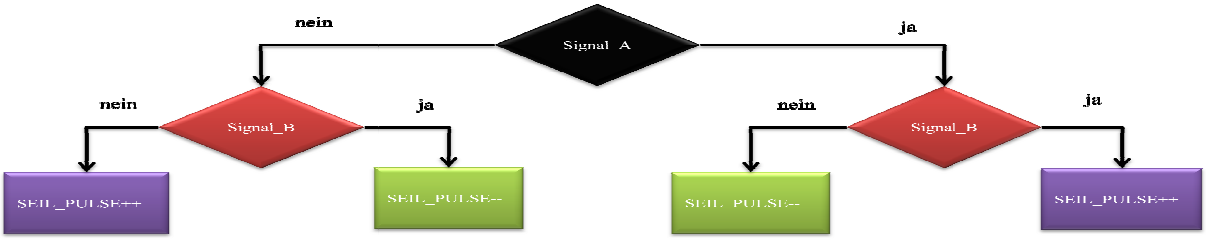


Bild 5.5 Flussplan der Interrupt Service Routine zur Wegmessung

[12] Inkrementiert bedeutet um eins erhöht
[13] Dekrementiert bedeutet um eins erniedrigt

#INT_RDA

void interrupthnd_daten_empfang(void)

Diese Interrupt-Service-Routine ist für das Empfangen der Pakete zuständig. Sobald das ISUS-Modul ein Byte empfangen hat, wird dies überprüft. Stimmt der Wert des empfangenen Bytes mit dem vordefinierten Vorlaufwert überein, wird eine Hilfsvariable „next_in“ um eins erhöht und es werden weitere Bytes empfangen. Stimmt der Vorlauf nicht, wird die Variable „nexit_in“ auf Null gesetzt. Von der Steuereinheit werden mindestens fünf Pakete erwartet. Der Anzahl der empfangenen Bytes entsprechen dem Wert der Variable „next_in“. Also wird der Wert der Variable „next_in“ abgefragt. Ist der Wert größer oder gleich fünf, wird eine Weitere Variable namens „empfangen“, die als Merker (Flag) dient, auf „TRUE“ gesetzt. Das Setzen der Variable „empfangen“ bedeutet, dass alle Bytes erfolgreich empfangen worden sind. Im Hauptprogramm wird gewartet bis dieser Merker gesetzt ist. Dies hat als Folge, dass die Bearbeitung des empfangenen Paketes gestartet wird. Der Merker wird im Hauptprogramm nach der Bearbeitung eines Paketes zurückgesetzt.

void ext_interupt_aktiv (void)

In dieser Funktion werden die Einstellungen des externen Interrupts wie folgt vorgenommen:

- Das Wegmessungsinterrupt wird bei steigende Flanke ausgeführt
- Der externe Interrupt wird freigeschaltet
- Interrupts allgemein werden freigeschaltet

void set_channel_adc (unsigned char kanal_nr)

Hier werden die Einstellungen des Analog/Digital-Umsetzers wie folgt vorgenommen:

- Die benötigten Pins des Mikrocontrollers, die als Eingänge für den Analog/Digital-Umsetzers vorgesehen sind, werden als analoge Eingänge eingestellt.
- Die Taktfrequenz der ADU wird eingestellt.
- Die Funktion erwartet die einzustellende Kanalnummer als Parameter. Durch die Eingabe der Kanalnummer und dem Aufruf der Funktion werden die Einstellungen des ADU für diesen Kanal vorgenommen und die Umsetzung wird gestartet. Laut Datenblatt des Controllers muss eine gewisse Zeit (50µs) gewartet werden bis der ADU eine weitere Messung mit einem anderen Kanal starteten kann. Dies wird auch in dieser Funktion durch eine Warteschleife realisiert.

void rs485_init (void)

In dieser Funktion wird der Empfangsinterrupt aktiviert und die serielle Schnittstelle für den Datenempfang eingestellt. Das Einstellen der seriellen Schnittstelle auf empfangen erfolgt in dem die serielle Schnittstelle für das Senden deaktiviert wird. Mehr dazu im folgenden Kapitel 6. Konzept und Implementierung

void rs485_data_senden(int anzahl, int data[])

Hier wird die Anzahl der zu sendenden Bytes angegeben, wobei die maximale Anzahl der zu sendenden Bytes auf 25 Bytes begrenzt ist. Weiterhin wird die Schnittstelle für das Senden der Daten aktiviert und die Daten werden seriell gesendet.

unsigned int redundanz_berechnung(int data[], int anzahl)

in dieser Funktion wird das CRC-Byte zusammengestellt. CRC bedeutet auf Deutsch zyklische Redundanzprüfung. Und ist ein Verfahren, welches bei der Übertragung von binär dargestellten Daten eingesetzt wird. Es ist so konstruiert, dass fast alle Fehler, die bei der Übertragung der Daten durch beispielsweise eine Störung auf der Leitung erkannt werden. CRC wird vor dem Beginn der Übertragung berechnet. Nach der Übertragung der Daten wird der CRC erneut berechnet. Beide Prüfwerte werden mit einander verglichen. Stimmen die Prüfwerte vor und nach der Übertragung überein, ist die Transaktion der Daten fehlerfrei verlaufen. Für die Berechnung der CRC ist diese Funktion zuständig. Diese Funktion ist Teil einer Softwarebibliothek der Firma D+H und soll hier nicht weiter erläutert werden.

6. Konzept und Implementierung

In diesem Kapitel wird sowohl das Konzept als auch die Implementierung des ISUS-Moduls dargestellt. Es wird dem Leser erklärt wie die in dem Kapitel Aufgabenstellung erwähnten Anforderungen des ISUS-Moduls konzipiert und implementiert werden.

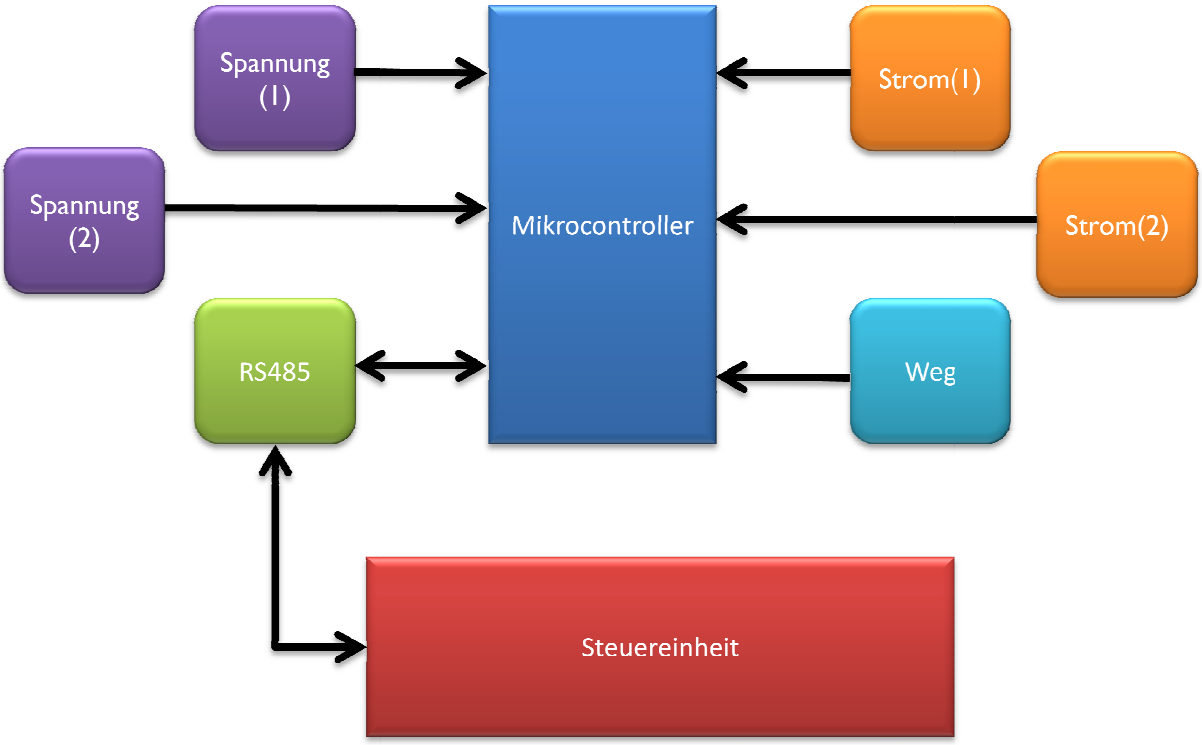


Bild 6.1 ISUS Übersicht

Bild 6.1 zeigt, dass das Projekt in vier Aufgabenteilen aufgeteilt ist. Das erste Teil beschäftigt sich mit der Wegmessung. Das zweite und das dritte Teil des Projekts beschäftigen sich mit der Spannung- und Strommessungen. Der vierte Teil des Projekts beschäftigt sich mit der Kommunikation zwischen des ISUS-Moduls und der Steuereinheit. Zunächst wird erklärt wie der Mikrokontroller beschaltet werden muss, damit er in Betrieb genommen werden kann.

6.1 Beschaltung des Kontrollers

Der PIC 16F886 besitzt acht verschiedene Oszillator-Modi

- EC – External clock with I/O on OSC2/CLKOUT.
- LP – 32 kHz Low-Power Crystal mode.
- XT – Medium Gain Crystal or Ceramic Resonator Oscillator mode.
- HS – High Gain Crystal or Ceramic Resonator mode.
- RC – External Resistor-Capacitor (RC) with Fosc/4 output on OSC2/CLKOUT.
- RCIO – External Resistor-Capacitor (RC) with I/O on OSC2/CLKOUT.
- INTOSC – Internal oscillator with Fosc/4 output on OSC2 and I/O on OSC1/CLKIN.
- INTOSCIO – Internal oscillator with I/O on OSC1/CLKIN and OSC2/CLKOUT

Wie es im Bild 6.1 zu entnehmen ist, wurde ein Resonator von 20MHz gewählt, um eine schnelle Abarbeitung des Programms zu gewährleisten. Auch für die Messung des Weges musste der Controller schnell getaktet sein, um alle Pulse des Seilzugsensors erfassen zu können. Für die Glättung der Versorgungsspannung des Controllers wurde ein 100nF Kondensator (C1) Parallel zum Kontroller gegen Masse angeschlossen. Die Steckverbinder (X1, X2, X4, X5, X6) sind wie bereits im Kapitel 4.7 beschrieben wurde für das Programmieren des Controllers über die ICSP- Schnittstelle vorgesehen.

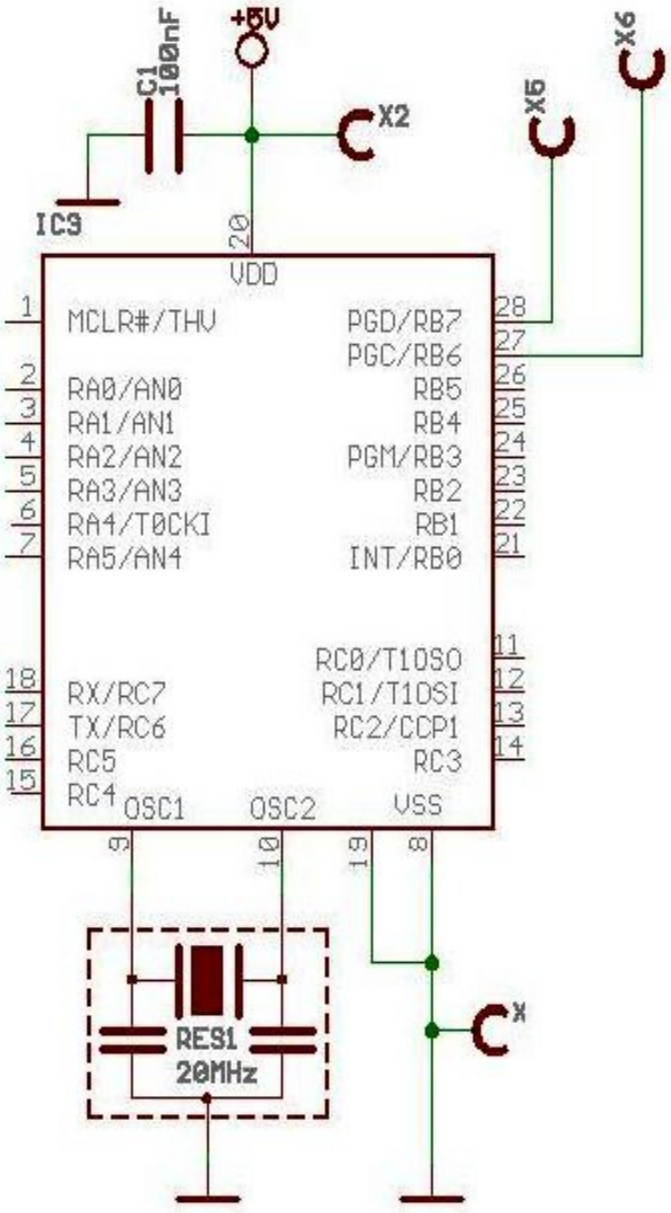


Bild 6.2 Beschaltung des Controllers

6.2 Wegmessung

In vielen Anwendungen des Mikrocontrollers muss auf bestimmte Ereignisse sehr schnell reagiert werden. Dies kann mit Hilfe sogenannter Interrupts realisiert werden. Ein Interrupt ist eine kurze Unterbrechung eines Programmablaufs, um ein anderes Teilprogramm, das zeitkritisch ist, abzuarbeiten. Das Teilprogramm heißt Interrupt-Service-Routine und sollte möglichst kurz gehalten sein. Nachdem die Interrupt-Service-Routine durchgeführt ist, wird die Ausführung des Programms an der unterbrochenen Stelle fortgesetzt. Am Port B des Mikrocontrollers kann der Pin0 sowohl als normaler I/O Pin als auch ein externer Interruptpin konfigurieret werden. Der externe Interrupt des Controllers hat zwei Einstellmöglichkeiten. Die Interrupt-Service-Routine wird ausgeführt wenn eine positive oder negative Flanke am Interruptpin erkannt wird. Das Reagieren auf beiden Flanken ist bei dem PIC16F886 nicht möglich. Wie bereits im Kapitel 4. 6 erwähnt wurde, wird die Wegmessung mit Hilfe des Seilzugsensors erfasst. Zieht man an dem Seil des Sensors, so gibt der Sensor zwei elektrische Signale aus. Diese beiden Signale wurden mit Hilfe eines Oszilloskops erfasst und sind im Bild 6.2.1 dargestellt.

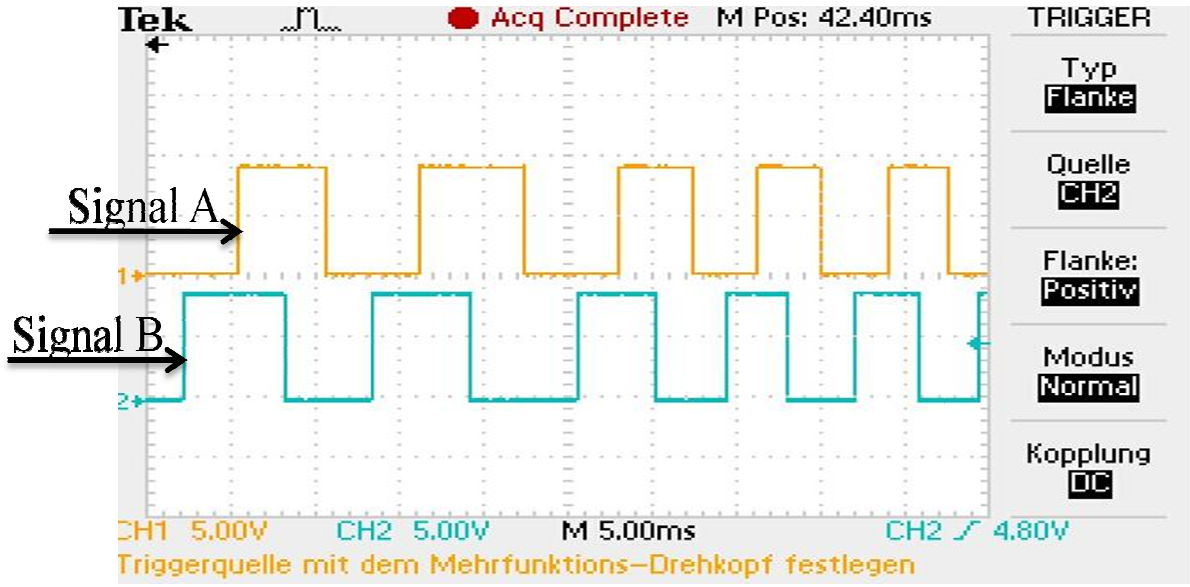


Bild 6.2.1 Ausgangssignale des Seilzugsensors

Das Signal (A) des Seilzugsensors (siehe Bild 6.2.2) wird über einen 10KΩ Widerstand (R3) an Port B Pin 0 angeschlossen. Ebenso wird Das Signal B über R5 an Port B Pin 1 angeschlossen. Die Werte der jeweiligen Widerstände sind dem Datenblatt des Seilzugsensors

entnommen worden. Um die Spannungen an den beiden genutzten Pins des Mikrocontrollers auf maximal 5.1V begrenzen zu können, wurden zwei Zener- Dioden(D12 und D13) parallel zum Kontroller gegen Masse angeschlossen.

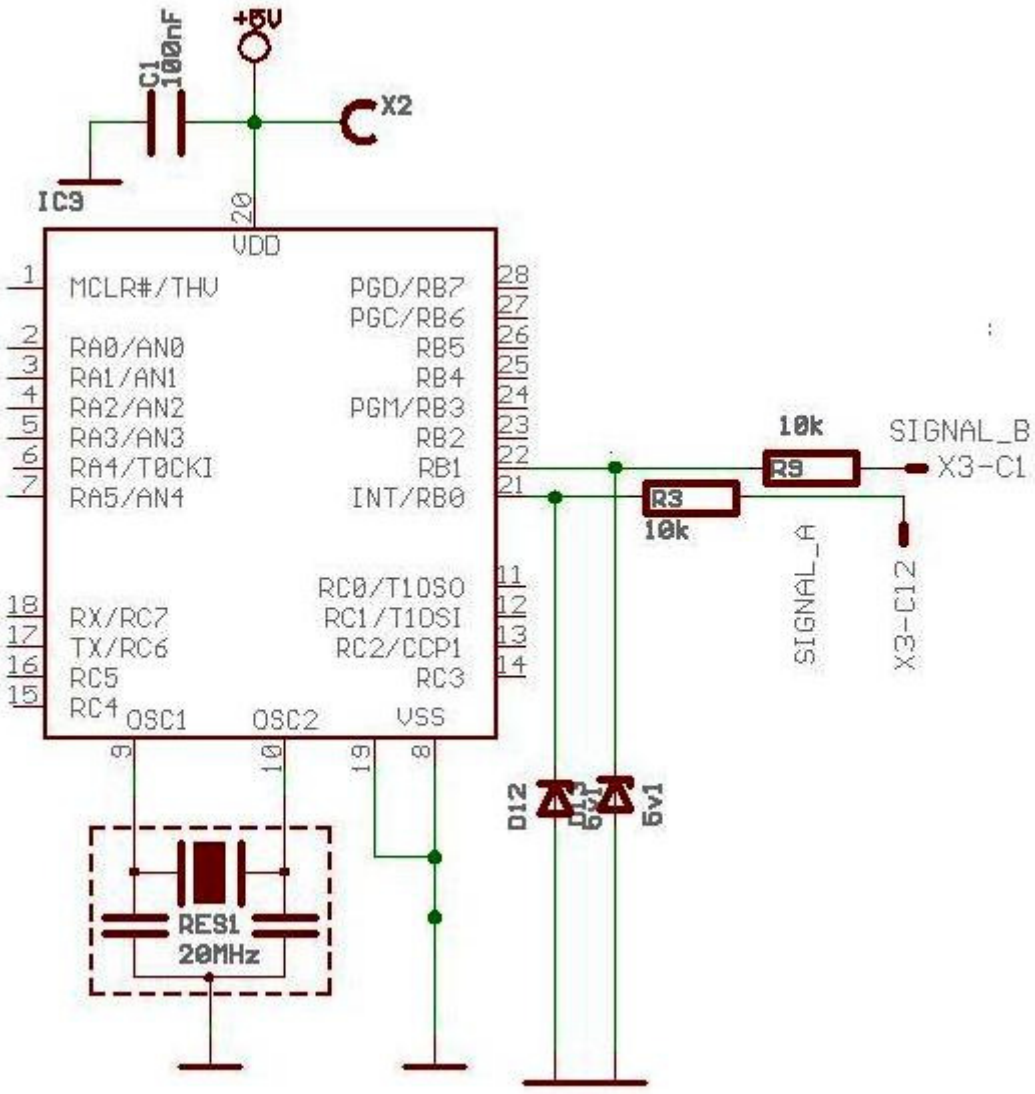


Bild 6.2.2 Schaltung für die Wegmessung

Der Interrupt wird so initialisiert, dass bei einer positiven Flanke die Interrupt-Service-Routine ausgeführt wird. Sobald eine positive Flanke erkannt wurde, wird die Einstellung so geändert, dass der Interrupt auf negative Flanken reagiert. Auf diese Art und Weise könnten sowohl die positiven als auch die negativen Flanken erkannt werden. Um den Weg messen zu können, wird bestimmt, in welcher Richtung sich das Seil bewegt und dementsprechend wird eine Variable hoch bzw. runter gezählt werden.

6.3 Spannungsmessung

Die maximale Spannung, die direkt mit dem ADU des Mikrocontrollers gemessen werden kann beträgt 5V. Alle Spannungen die großer als 5V sind, müssen durch einen Spannungsteiler auf eine für den Mikrokontroller geeignete Größe reduziert werden.

Da der ADU des PIC (16F886) eine Auflösung von 10Bit hat, kann er Spannungen von 0V bis 5V mit einer

Genauigkeit von Genauigkeit = $\frac{5V}{2^{10}} = 4.88mV / LSB$

Formel (6.1)

messen. Anhand des nachfolgenden Bildes (Bild 6.3.1) wird die Schaltung für Spannungsmessung ersichtlich.

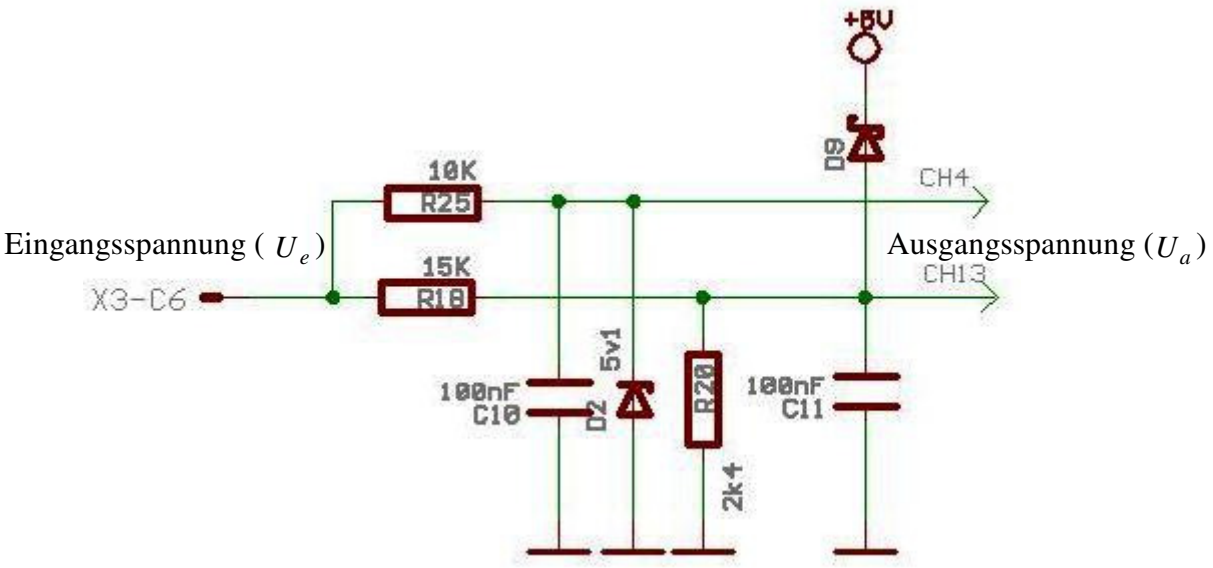


Bild 6.3.1 Schaltung für Spannungsmessung

Wie es aus dem Bild (6.3.1) zu entnehmen ist dienen die Widerstände R18(15 KΩ) und R20 (2.4 KΩ) als Spannungsteiler. Die Diode D9 begrenzt die Spannung auf maximal 5.7 Volt in dem Fall, dass die Eingangsspannung viel großer als 30 Volt ist und schützt somit den Mikrocontroller. Die zwei Kondensatoren C10 und C11 dienen zur Glättung der

Eingangsspannung. Die Zener-Diode D2 begrenzt die Spannung auf Maximal 5.1V und Schützt ebenfalls den Controller vor höheren Spannungen [2].

Zunächst wird der Messbereich in zwei Stufen unterteilt. In der ersten Stufe werden Spannungen von 0V bis 4V gemessen. In der zweiten Stufe werden Spannungen von 4V bis 30V gemessen. Dafür werden zwei Kanäle des ADU benötigt. Die Spannungen von 0V bis 4V werden mit dem Kanal vier des ADU(CH4) gemessen. Es ist dabei wichtig zu beachten, dass während eine Messung mit einem ADU-Kanal durchgeführt wird, an keinem anderen Kanal des ADU eine Spannung, die größer ist als 5.5V anliegen darf. Dies würde zu einer Verfälschung des Messergebnisses führen. Es wird davon ausgegangen, dass die zumessende Spannung großer als 4V ist. Deshalb wird die Spannung über den Spannungsteiler zum ADU-Eingang (CH13) geleitet. Der Spannungsteiler (Formel 6.3) ist so ausgelegt, dass bei 30V Eingangsspannung eine Spannungsgröße von 4V am Kanal 13 des ADU anliegt.

$$\frac{U_e}{U_a} = \frac{R_{18} + R_{20}}{R_{20}}$$

$$\frac{30V}{U_a} = \frac{2.3K\Omega + 15K\Omega}{2.3K\Omega}$$

$$U_a = \frac{30V}{7.5} = 4V$$

Formel (6.2)

Das Ergebnis des ADU ist ein digitaler Wert und liegt im Bereich von 0 bis 1023. Dieser wird mit folgender Formel in eine Spannung umgewandelt. Spannung (U) = ADU-Wert* Referenzspannung(U_{REF})/Auflösung der ADU

$$U = \frac{ADU - Wert * 5V}{1024}.$$

Formel (6.3)

Wenn die gemessene Spannung größer als 4V ist, wird dies ohne weitere Messungen an die Steuereinheit übermittelt. Wurde festgestellt, dass die gemessene Spannung kleiner als 4V ist, wird die Spannung zum ADU (CH4) ohne vorgeschaltetem Spannungsteiler geleitet und die

Messung wird wiederholt. Anschließend wird der Spannungswert nach einer Anfrage an die Steuereinheit geschickt.

Um exakte Wandlungsergebnisse bei Der A/D- Umwandlung zur erhalten, benötigt der Mikrocontroller eine stabile Referenzspannung. Dies Wurde mit Hilfe eines Spannungsreglers realisiert. Im folgenden Bildes 6.3.2 ist Beschaltung des Spannungsreglers zusehen. Diese wurde dem Datenblatt des Spannungsreglers entnommen.

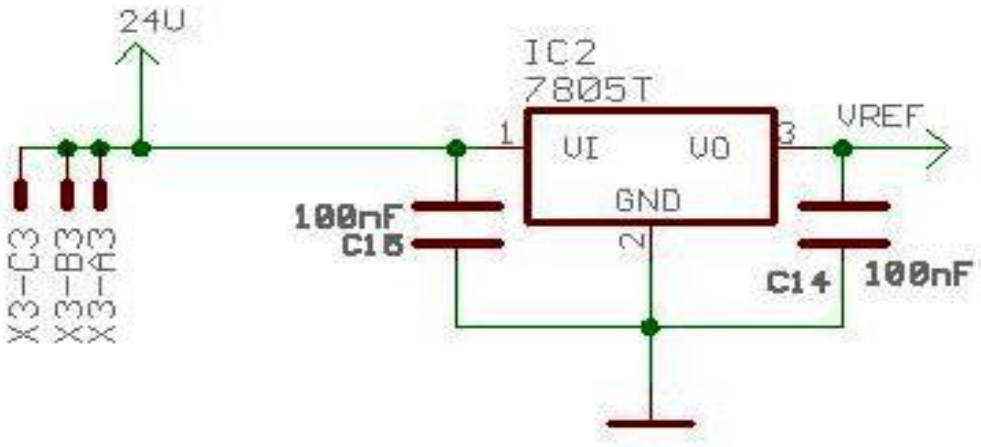


Bild 6.3.2 Referenzspannung

Bei der Initialisierung des Controllers muss angegeben werden, ob die Referenzspannung extern oder intern anliegt. Die positive externe Referenzspannung (V_{REF+}) liegt am Kanal 3 des Mikrocontrollers und beträgt 5V. Die negative Referenzspannung (V_{REF-}) liegt am Kanal 2 und beträgt 0V. Diese beiden Kanäle stehen daher für die Messung nicht mehr zu Verfügung. Da das Modul zwei Spannungen dieser Art messen können muss, wurde diese Schaltung zweimal implementiert.

6.4 Strommessung

Um den Strom messen zu können, wurde der Highside- Instrumentenverstärker (INA168) eingesetzt. Der Verstärkungsfaktor der INA168 ist über einen Widerstand einstellbar. Der INA168 besitzt eine hohe Eingangsimpedanz, niedriger Ausgangsimpedanz, eine hohe Verstärkung mit guter Verstärkungsgenauigkeit und einen breiten Gleichtaktbereich mit guter Gleichtaktunterdrückung. Bild 6.4.1 zeigt wie der INA168 schematisch aufgebaut ist. Die Eingangsspannung „V_{IN+}“ kann je nach Anwendung zwischen 2.7V bis 60V gewählt werden. Die Betriebsspannung des INA kann zwischen -0.3V bis 75V gewählt werden. Naturgemäß kann die Ausgangsspannung des INA nicht größer sein als die Spannung mit der der INA betrieben wird. Zwischen den Pins (3,4) des INA ist der Shunt widerstand platziert. An Pin 4 ist der Motor bzw. die Last anzuschließen. Die Ausgangsspannung (V_O) lässt sich nun aus der gegebenen Formel berechnen.

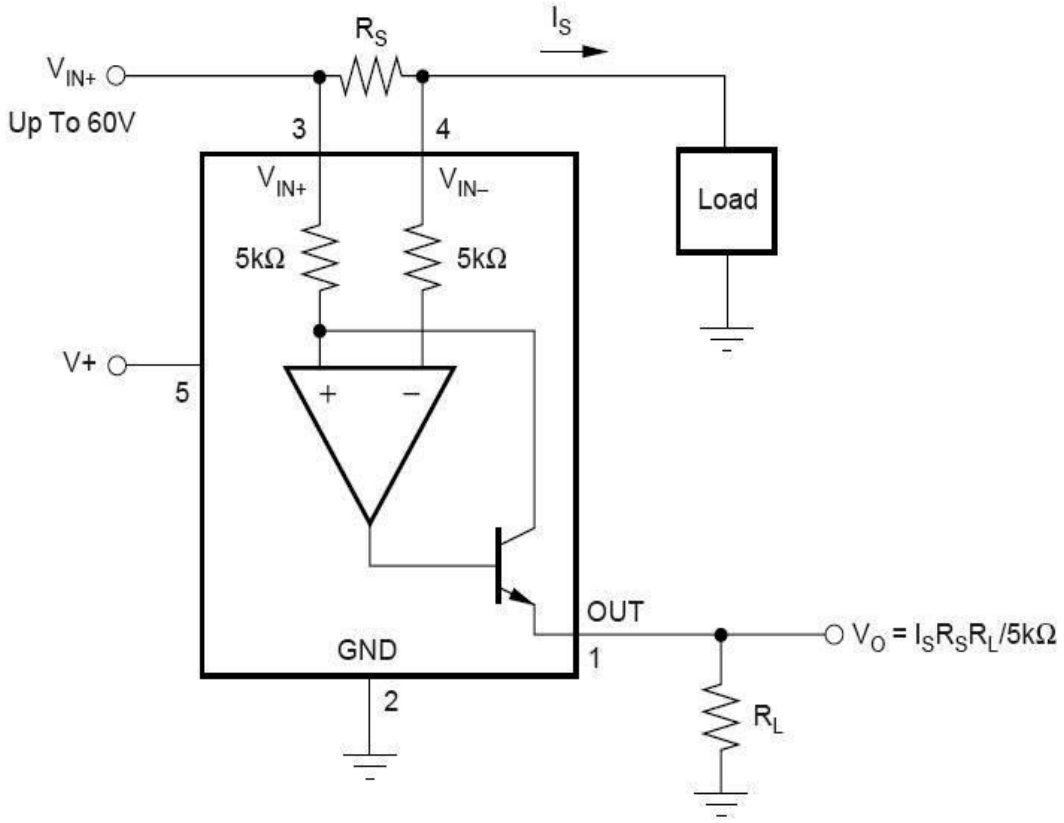


Bild 6.4.1 INA168

Der Maximale Strom (I_s), der gemessen werden muss, beträgt 6Amper. Der Shunt Widerstand (R_s) sollte nicht größer als 0.1Ω sein, so dass der Spannungsabfall am

Widerstand (R_s) nicht größer als 0.6V ist. Somit wird die Nennspannungen für die Motorsteuerung, nicht unterschritten. Der Widerstand (R_L) ist so dimensioniert, dass bei Is = 6A eine Ausgangspannung (V_O) = 5V beträgt.

$$V_O = \frac{I_s * R_s * R_L}{5k\Omega} ==> R_L = \frac{Vo * 5k\Omega}{Is * R_s}$$

$$R_L = \frac{5V * 5k\Omega}{6A * 0.1\Omega} = 41.666k\Omega$$

Formel (6.4)

Anhand des nachstehenden Bildes 6.4.2 wird die Implementierung dargestellt.

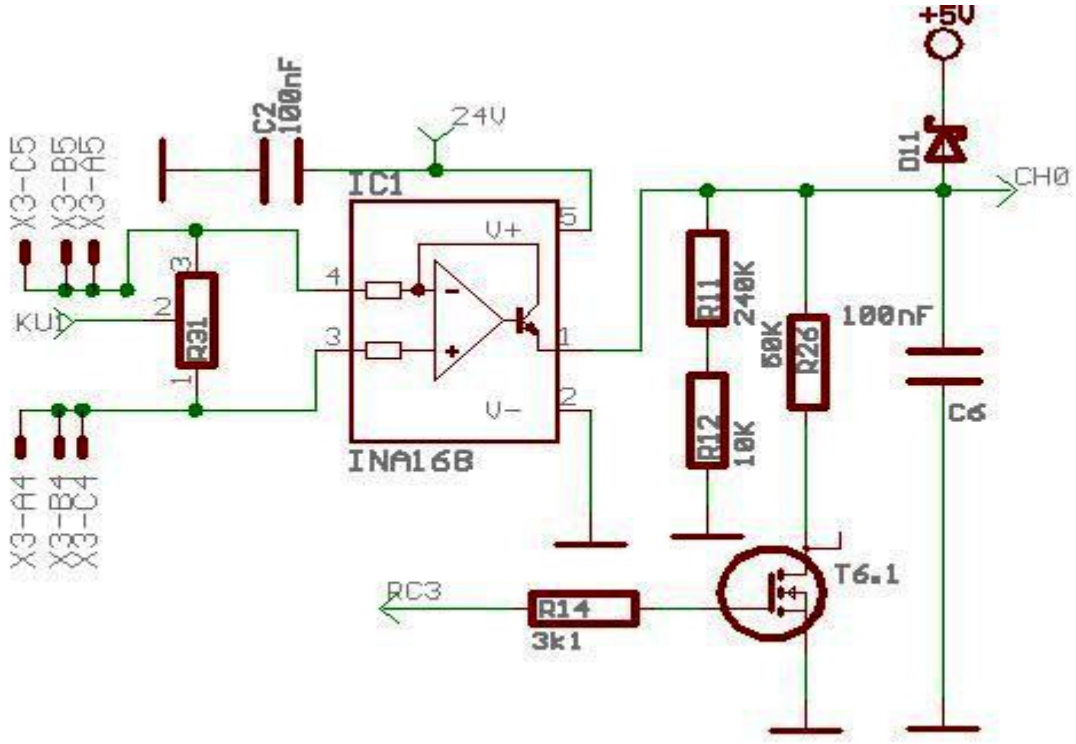


Bild 6.4.2 Realisierung einer Strommessung mit einem Highside-Verstärker

Der INA168 wurde mit 24V über den Kondensator C2(100nF) betrieben. Der Kondensator C2 hat die Aufgabe, die Versorgungsspannung des INA zu glätten. An den Pins (3,4) wird die Eingangsspannung bzw. der Motor angeschlossen, dessen Strom gemessen werden soll. Die

Wiederstände R11(240 kΩ) , R12 (10 kΩ) , R26(50kΩ) sind R_L und ergeben insgesamt 41.666 kΩ wenn der Transistor T6.1 leitend ist. Der Transistor T6.1 wird über den Widerstand R14(3.1 kΩ) per Mikrocontroller angesteuert. Der gesamte Widerstand R_L lässt sich durch folgend Formel berechnen.

R_L = R11+R12 // R26 ➔

$$R_L = (\frac{(R_{11+}R_{12}) * R_{26}}{R_{11+}R_{12} + R_{26}}) = \frac{(240k\Omega + 10k\Omega) * 50k\Omega}{240k\Omega + 10k\Omega + 50k\Omega} = 41.666k\Omega$$

Formel (6.5)

Um die Ströme von 0Amper bis 1Amper möglichst genau messen zu können, wird der Transistor T6 ausgeschaltet so dass R_L = R11+R12 kΩ=250 kΩ beträgt.

$$V_O = \frac{I_s * R_s * R_L}{5k\Omega} ==>$$

$$V_O = \frac{1A * 0.1\Omega * 250k\Omega}{5k\Omega} = 5V$$

Formel (6.6)

Die Diode D5 begrenzt die Spannung auf maximal 5.7V und schützt damit den Mikrocontroller. Der Kondensator C6(100nF) dient zur Glättung der ADU-Eingangsspannung. Zunächst wird der Transistor T6.1 so geschaltet das $R_L = 41.666k\Omega$ beträgt. Dies bedeutet, es werden Ströme erwartet, die größer als 1A sind. Die Ausgangsspannung (V_O) wird mit dem ADU (Kanal 0)gemessen. Da nun Die Spannung (V_O) und der Widerstand (R_L) bekannt sind, wird der vorgegeben Formel (Bild 6.4.1) nach der Strom (I_s) aufgelöst. $I_s = \frac{V_o * 5K\Omega}{R_s + R_L}$

Formel (6.7)

Ist der Strom größer als 1A, wird das Ergebnis ohne weitere Messungen an der Steuereinheit gesendet. Wurde festgestellt, dass der Strom kleiner als 1A ist, wird der Widerstand (R26) mit Hilfe der Transistor freigeschaltet und die Spannungsmessung wird wiederholt. Der Strom erneut berechnet und die Werte werden an der Steuereinheit übermittelt. Da das Modul zwei Ströme dieser Art messen können soll, wurde diese Schaltung zweimal implementiert.

6.5 Kommunikation

An der USART Schnittstelle (Pin RC6 und RC7) und einem Enable Pin RC5 ist der bereits im Kapitel 4.5 erwähnter RS 485 Transceiver (IC5) angeschlossen. Zur Glättung der Betriebsspannung des Transceiver (Bild6.5) dient der Kondensator C5. Die Bus Leitung A hat R1 als Pullup -und Leitung B hat R2 als Pulldown-Widerstand, damit am Transceiver auch bei offenen Busleitungen ein definierter Zustand anliegt. R5 und R6 schützen den Transceiver vor hohen Strömen, die höhere Temperatur zu Folge hätten, wodurch der Widerstandswert steigen und der Stromwert bei gleicher Eingangsspannung sinken würde.

Die Widerstände R7 und R8 sind zum Schutz von Überspannungen, da diese im Überspannungsfall fast verzögerungsfrei einen sehr kleinen Widerstandswert annehmen und Ladung ableiten würden.

Die Zener-Dioden (D1, D2 und D3) schützen den Transceiver durch symmetrische Spannungsbegrenzung zum Beispiel für den Fall, dass der Bus falsch beschaltet wurde Maßnahmen der Vorbeschaltung sind von der Firma D+H vorgegeben.

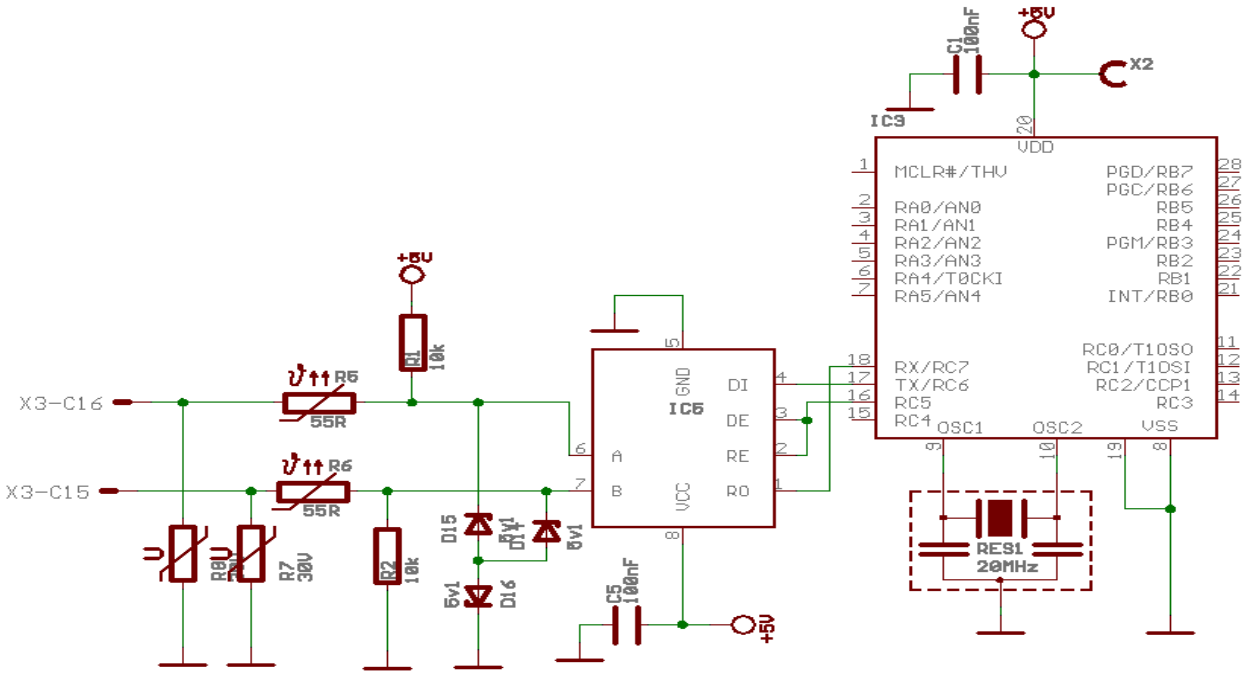


Bild 6.5.1 Beschaltung des RS485-Transceiver

Für die Kommunikation zwischen dem ISUS-Modul und der Steuereinheit sind drei Status-LEDs angebracht. Die Status-LEDs sind an Port C Pins (0, 1, 2), wie es im Bild 6.5.2 zu sehen ist, angeschlossen. Während der Kommunikation zwischen der Steuereinheit und dem

ISUS-Modul geht die rote LED an, wenn die gemessenen Stromwerte an die Steuereinheit übermittelt werden. Die gelbe LED zeigt an, dass die Spannungswerte abgeholt werden. Die grüne LED blinkt kurz, wenn die gemessene Strecke an die Steuereinheit gesendet wird. Alle drei LEDs gehen kurz gemeinsam An und Aus, wenn das ISUS-Modul die Steuereinheit mitteilt, dass es breit ist, eine Messung durchzuführen.

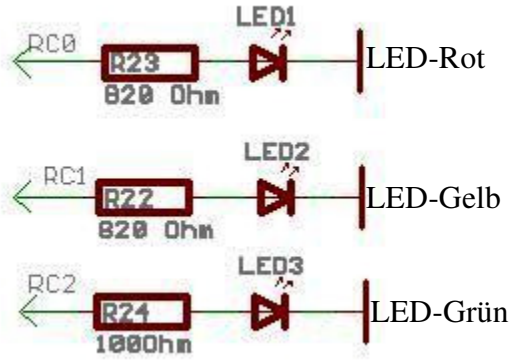


Bild 6.5.2 Status LEDs

Zusätzlich haben die rote- und die grüne-LED weitere Funktionen. Die grüne LED geht an wenn an dem Seil des Seilzugsensors gezogen wird und zeigt somit, dass z.B. ein Fenster öffnet. Hingegen geht die rote LED an, wenn das Seil in den Sensor einfährt und zeigt, dass das Fenster schließt.

7. Platine

Die Platine für das ISUS-Modul wurde mit Hilfe des Layout-Programms „Eagle“ der Firma CadSoft entwickelt. Die Platine wurde doppelseitige Ausführung entwickelt und bestückt.

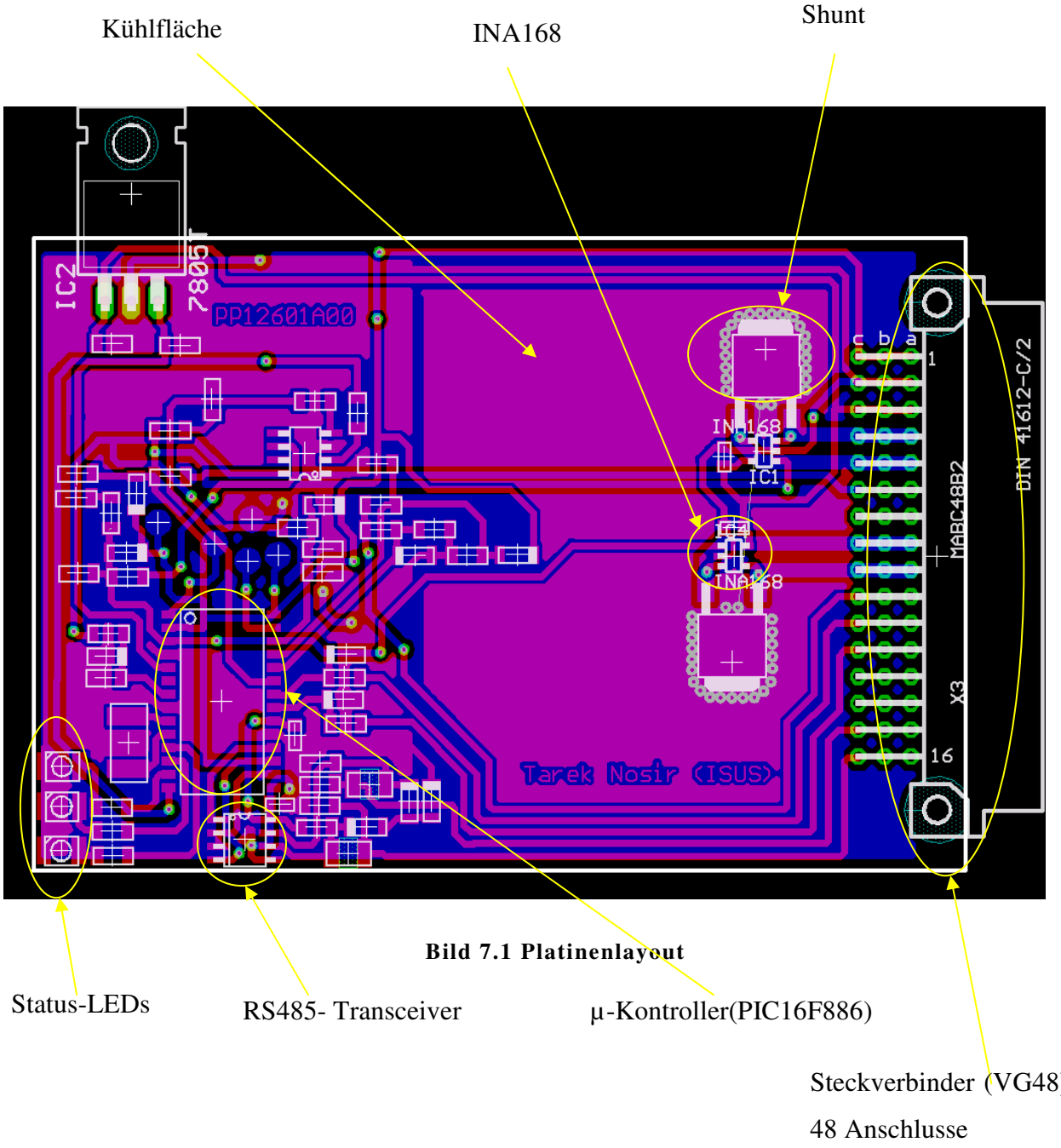


Bild 7.1 Platinenlayout

Beim Erstellen des Layouts wurde darauf geachtet, dass alle Bauteile sich innerhalb des erlaubten Rahmen (60mmx90mm) befinden. Die Bauteile die zusammen gehören wurden auch nebeneinander platziert, so dass keine langen Leiterbahnen entstehen. So nah wie möglich wurde neben jedem IC einen Kondensator platziert, um die Versorgungsspannung des ICs zu glätten. Nach dem alle Bauteile platziert und miteinander verbunden waren, wurde nach Vereinfachungen gesucht. Es wurden Widerständen als Drahtbrücken genutzt um längere Leiterbahnen zu sparen. Wie aus dem Bild 7.2 zu entnehmen ist, wurden die kupferlosen Zwischenräume mit 2 unterschiedlichen Signalen befühlt. Dies wurde mit Hilfe des Eagle-Befehls Polygon erreicht. Daraus sind zwei Flächen Kupfer entstanden. Die erste Fläche dient als Massefläche und sorgt dafür, dass die gesamte Platine störungsresistenter gegen elektromagnetische Strahlung wird. Die zweite Fläche dient als Kühlfläche, da es bei einem Strom (I) von 6A und ein Shunt (R_{sunt}) von 0.1Ω eine Leistung von 3.6Watt entsteht.

$$W = I^2 * R_{shunt} \rightarrow \text{W} = 6^2 * 0.1 = 3.6W$$

Formel (7. 1)

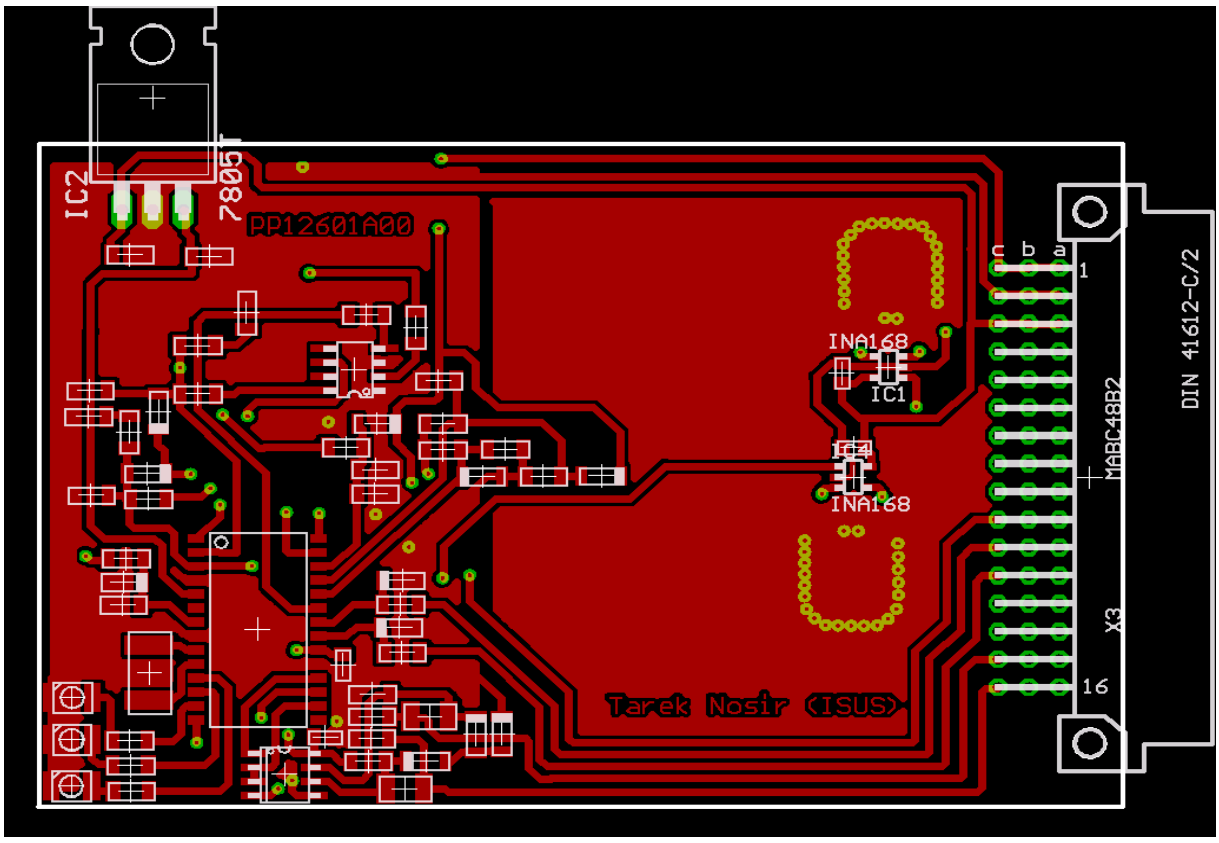


Bild 7.2 Platinenlayout Vorderseite

Damit die durch die relativ große Leistung entstehende Wärme abgegeben werden kann, wurde auf der Rückseite der Platine eine dritte Kupferfläche, die auch als Kühlfläche dient (Bild 7.3), eingefügt. Beide Kühlflächen auf der Vorder- und Rückseite der Platine wurden anhand von Durchkontaktierungen miteinander verbunden.

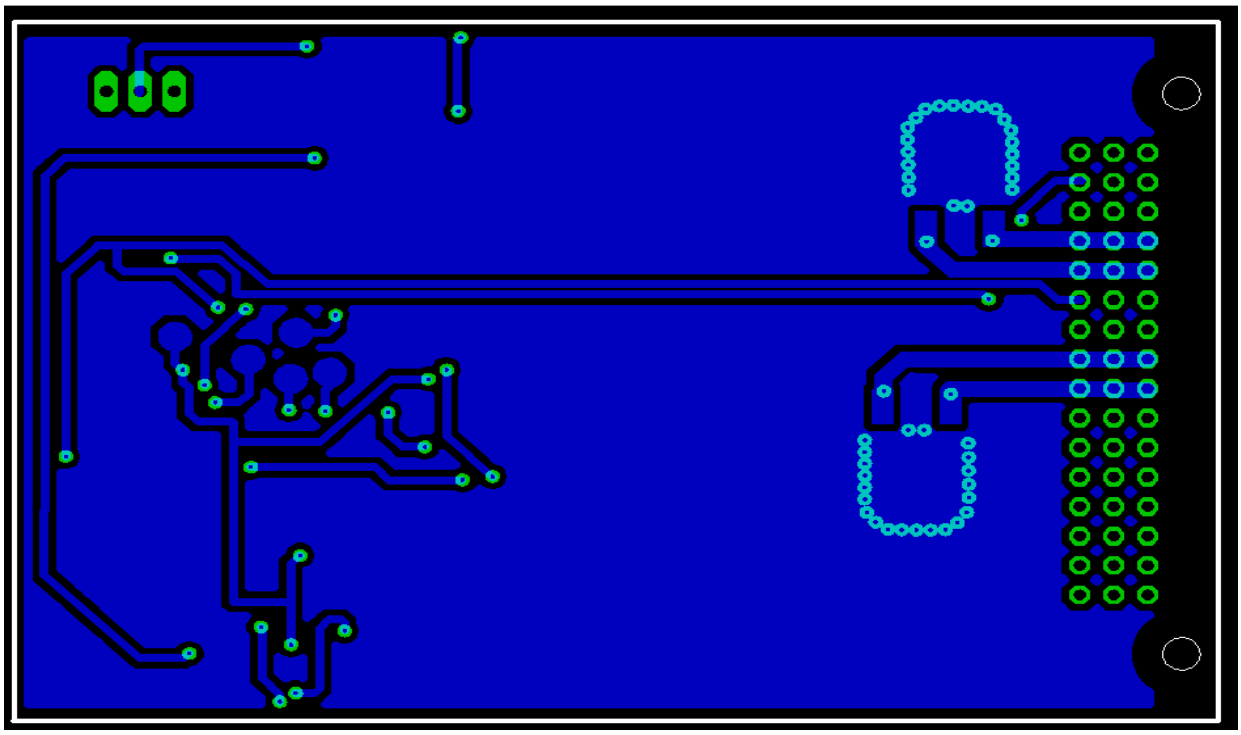


Bild 7.3 Platinenlayout Rückseite

Im folgenden Bild7.4 ist ein Belegungsplan für den VG84-Steckverbinder zu sehen.

A1	Versorgungsspannung (+5V)	B1	Versorgungsspannung (+5V)	C1	Versorgungsspannung (+5V)
A2	Masse(0V)	B2	Masse(0V)	C2	Masse(0V)
A3	Versorgungsspannung (+24V)	B3	Versorgungsspannung (+24V)	C3	Versorgungsspannung (+24V)
A4	Spannungsversorgung Antrieb(1)	B4	Spannungsversorgung Antrieb(1)	C4	Spannungsversorgung Antrieb(1)
A5	Motoranschluss (1)	B5	Motoranschluss (1)	C5	Motoranschluss (1)
A6		B6		C6	Spannungsversorgung Antrieb(2)
A7		B7		C7	
A8	Spannungsversorgung Antrieb(2)	B8	Spannungsversorgung Antrieb(2)	C8	Spannungsversorgung Antrieb(2)
A9	Motoranschluss (2)	B9	Motoranschluss (2)	C9	Motoranschluss (2)
A10		B10		C10	Spannungsversorgung Antrieb(1)
A11		B11		C11	
A12		B12		C12	
A13		B13		C13	
A14		B14		C14	
A15		B15		C15	Data A (RS485)
A16		B16		C16	Data B (RS485)

Bild 7.4 Belegungsplan Steckverbinder (VG48)

8. Funktionstest

Um das ISUS-Modul auf seine korrekte Arbeitsweise hin testen zu können, wurde eine Testplatine entwickelt. Die Testplatine besteht aus demselben Kontroller wie beim ISUS-Modul und einen Anzeige Display. Die Kommunikation zwischen dem ISUS-Modul und der Testplatine erfolgte über den RS485-Bus. Bild 8.1 zeigt eine grobe Übersicht zum Hardwareaufbau. Die Testplatine ersetzte in der Testphase die Steuereinheit, das Kommunikationsprotokoll wurde jedoch einfacher Aufgebaut.

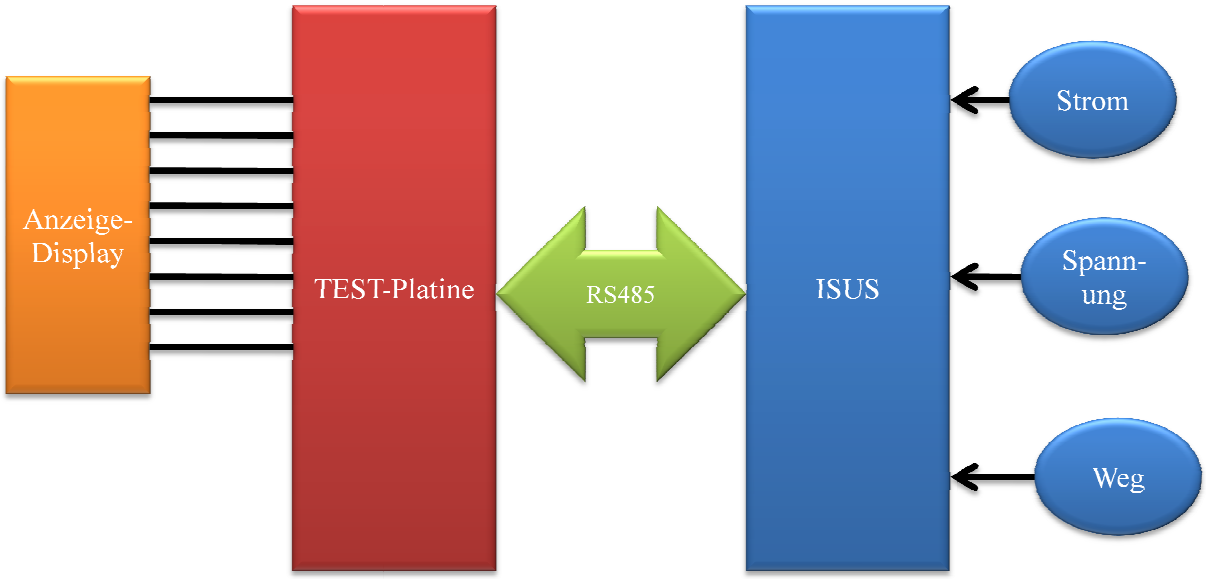


Bild 8.1 Übersicht Funktionstest

Zum Anzeigen der gemessenen Werte wurde eine einfaches 16x2 -LCD-Display mit 16 Anschlusspins verwendet. Anhand des folgenden Bildes 8.2 werden die Bedeutungen der einzelnen Pins erläutert. Es wird erklärt, wie das Display an den Mikrokontroller angeschlossen ist.

Pin NO.	Symbol	Level	Description
1	VSS	0V	Ground
2	VDD	5.0V	Supply voltage for logic
3	VO	---	Input voltage for LCD
4	RS	H/L	H : Data, L : Instruction code
5	R/W	H/L	H : Read mode, L : Write mode
6	E	H, H → L	Chip enable signal
7	DB0	H/L	Data bit 0
8	DB1	H/L	Data bit 1
9	DB2	H/L	Data bit 2
10	DB3	H/L	Data bit 3
11	DB4	H/L	Data bit 4
12	DB5	H/L	Data bit 5
13	DB6	H/L	Data bit 6
14	DB7	H/L	Data bit 7
15	BLA	---	Back light anode
16	BLK	---	Back light cathode

Bild 8.2 Pinbelegung LCD-Display

- **VSS** : wird an die Masse angeschlossen
- **VDD** : wird an die +5V angeschlossen
- **VO** : wird an einem Potentiometer angeschlossen um den Kontrast des Display einstellen zu können
- **RS** : Dieser Pin breitet das Display darauf vor Befehle oder Daten zu empfangen
- **R/W** : Anhand dieses Pins, wird entschieden ob, das Display etwas sendet oder empfängt
- **E** : ist der Freigabe Pin
- **DB0-DB7:** Sind Datenbits und legen fest, was und wo etwas auf dem Display geschrieben wird
- **BLA/BLK:** Sind für die Hintergrundbeleuchtung zuständig und werden mit 0V und +5V angeschlossen.

9. Zusammenfassung

Das Ziel dieser Diplomarbeit war es ein Modul zu entwickeln, welches Strom, Spannung und Weg messen kann und die gemessen Werte an eine Steuereinheit über den RS485- Bus übermittelt. Das Modul gehörte zur einen Prüfeinheit, die unterschiedliche Antriebe auf Ihre korrekte Arbeitsweise testete.

Zunächst wurden die Anforderungen an das Modul gesammelt, dann wurden diese in mehreren Aufgaben unterteilt. Für die Wegmessung wurde ein Seilzugsensor der Firma Kübler eingesetzt. Dieser Sensor wandelt lineare Bewegung durch Aus- und Einzug eines Seiles in Drehbewegungen um. Diese werden von den nachgeschalteten Drehgebern oder Potentiometern in entsprechend elektrische Signale umgewandelt. Die elektrischen Signale des Sensors wurden mit Hilfe eines Mikrocontrollers ausgewertet. Die Strom- und Spannungsmessungen wurden mit Hilfe des Analog/Digital- Umsetzers des Mikrocontrollers (PIC16F886) realisiert.

Die für diese Arbeit benötigte Schaltungen und das Platinenlayout wurden mit dem Layout-Programm „Eagle“ der Firma CadSoft erstellt. Eine gewisse Zeit für die Einarbeitung in dieses Programm war notwendig.

Die Software für diese Arbeit wurde mit dem „MPLAB-“Paket der Firma Microchip in der Programmiersprache C entwickelt.

Bilderverzeichnis

Formelverzeichnis

Literaturverzeichnis

[1] **U. Tietze, Ch. Schenk**

Halbleiter-Schaltungstechnik

Springer- Verlag

[2] **Herbert Bernstein**

Werkbuch Mechatronik

[4] **Joachim Goll, Ulrich Bröckl, Manfred Dausmann**

C als erste Programmiersprache

B.G.Teubner Stuttgart. Leibzig.Wiesbaden

Verwendete Internetseiten

[5] http://www.feuer-und-rauch.de (Datum des Zugriffs: 14.Januar 2009)

[6] http://www.wut.de/e-6wwww-11-apde-000.php (Datum des Zugriffs: 01.November. 2008)

[7] http://www.vias.org/mikroelektronik/adc_succapprox.html (Datum des Zugriffs: 18.Dezember 2008)

[8] http://www.sprut.de/electronic/pic/icsp/icsp.htm (Datum des Zugriffs: 16.Oktober 2008)

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §25(4) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe

Hamburg, 06. April 2009

Ort, Datum

Unterschrift