

# Technikerarbeit 2012

Werner-von-Siemens Schule  
Mannheim

## CNC-Steuerung mit Touch-Display und USB-Speichermedium



Erstellt von:  
**Dennis Hohmann**

Betreut von:  
**Dr. Bernhard Spitzer**

## Erklärung

Die vorliegende Technikerarbeit, **CNC-Steuerung mit Touch-Display und USB-Speichermedium**, habe ich, **Dennis Hohmann geb. 12.12.1984**, selbstständig und nur mit den angegebenen Hilfsmitteln angefertigt.

Alle Stellen, die dem Sinn oder dem Wortlaut nach anderen Werken entnommen sind, wurden durch Angabe der Quellen kenntlich gemacht. Alle wörtlich entnommenen Stellen sind als Zitate gekennzeichnet.

---

Ort, Datum

Unterschrift

## Inhaltsverzeichnis

<b>1 Vorwort .....</b>	<b>5</b>
<b>1.1 Einleitung .....</b>	<b>5</b>
<b>1.2 Copyright .....</b>	<b>5</b>
1.3 Danksagung .....	6
1.4 Abbildungsverzeichnis .....	7
1.5 Abkürzungsverzeichnis .....	9
<b>2 Projektbeschreibung .....</b>	<b>11</b>
2.1 Motivation .....	11
2.2 Aufgabenstellung .....	12
2.3 Zeitplan .....	13
2.4 Planung .....	14
<b>3 Hardware .....</b>	<b>15</b>
3.1 Controller .....	15
3.2 Speichermedium .....	16
3.3 Display .....	17
3.4 Portalfräse .....	18
3.5 UNI1500 .....	20
3.6 Genauigkeit .....	21
3.6.1 X- & Y- Achsen .....	22
3.6.2 Z-Achse .....	24
3.7 Geschwindigkeit .....	24
3.8 Werkzeugdrehzahl .....	25
3.9 Hardware- & Softwarekonfiguration .....	26
3.9.1 Atmel ATmega1284P-PU Fuses & Lockbits .....	26
3.9.2 Vinculum VDrive2 Firmware 3.68 .....	28
3.9.3 EAGLE .....	28
3.9.4 PCB-GCODE-ULP 3.5.2.11 .....	29
<b>4 Software .....</b>	<b>31</b>
4.1 Einleitung .....	31
4.2 Controller-Programm .....	32
4.2.1 Globale Definitionsdatei globdef.h .....	34
4.2.2 Schrittmotor-Handling gocnc.c .....	35
4.2.3 USB-Kommunikation vnc1l.c .....	36
4.2.4 Serielle Schnittstelle uart.c .....	37
4.2.5 Serieller Datenbus i2cmaster.c .....	38
4.2.6 Display-Handling edip240.c .....	39
4.2.7 gCode-Struktur gcode.c .....	40
4.3 EA KitEditor-Programm .....	42
4.4 SmallProtocoll .....	46
<b>5 Die Steuerplatine .....</b>	<b>48</b>
5.1 Aufbau .....	48

---

5.2 Das Design .....	49
<b>6 Zusammenfassung.....</b>	<b>50</b>
6.1 Fazit .....	50
6.2 Ausblick .....	50
<b>7 Quellenverzeichnis.....</b>	<b>51</b>
<b>8 Softwareverzeichnis.....</b>	<b>53</b>
<b>9 Anhang .....</b>	<b>54</b>
9.1 Schaltpläne .....	55
9.1.1 Mainboard.....	55
9.1.2 Dip240-7 .....	56
9.1.3 VDrive2 .....	57
9.1.4 Board-Layout .....	58
9.1.5 Gehäuse .....	59
9.1.6 Fertige Platine.....	60
9.2 Datenblätter .....	61
9.2.1 EA eDIP240B-7LWTP .....	61
9.2.2 Vinculum VDrive2 .....	82
9.2.3 Gehäuse .....	90
9.2.4 Gleichrichter.....	91
9.3 Dokumentations-CD .....	94

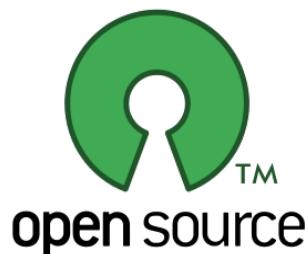
## 1 Vorwort

### 1.1 Einleitung

Die Weiterbildung zum staatlich geprüften Techniker Elektrotechnik beinhaltet begleitend zu den Theorieinhalten der Fachstufen die Umsetzung einer praktischen Abschlussarbeit. Die Technikerarbeit soll zeigen, dass erlerntes Wissen über die vermittelten Theorieinhalte hinaus angewandt und komplexe Aufgabenstellungen erarbeitet und gelöst werden können. Der Aufbau einer Technikerarbeit ist unterteilt in die Ausarbeitung eines Themas, die Planung des Projektes, die praktische Umsetzung und die Erstellung einer schriftlichen Dokumentation. Ebenso ist die abschließende Präsentation Bestandteil.

### 1.2 Copyright

Die Veröffentlichung dieser Dokumentation, sowie die Verwendung der enthaltenen Pläne und Layouts, ist ohne meine Zustimmung gestattet, sofern die Angabe der Quelle enthalten bleibt.



### 1.3 Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die mich bei der Realisierung dieses Projekts unterstützt haben. Besonderer Dank gilt folgenden Personen und Firmen:

Dr. Bernhard Spitzer

Für die fachliche Betreuung.

ELECTRONIC ASSEMBLY GmbH<sup>1</sup>

Für die kostenfreie Überlassung eines eDIP240B-7LWTP Touchdisplays.

Atmel<sup>2</sup>

Für die kostenfreie Überlassung der Controller als Samples,  
sowie ein sehr günstiges Angebot für einen AVR JTAGICE<sup>3</sup> Debugger.

CadSoft<sup>4</sup>

Für die kostenfreie Überlassung einer Version EAGLE 5.7 Pro OSX<sup>5</sup>.

---

<sup>1</sup> (ELECTRONIC ASSEMBLY GmbH, 1)

<sup>2</sup> (Atmel Corporation, 1)

<sup>3</sup> (Atmel Corporation, 2)

<sup>4</sup> (CadSoft Computer GmbH, 1)

<sup>5</sup> (CadSoft Computer GmbH, 2)

## 1.4 Abbildungsverzeichnis

Abbildung 1.2.1: OpenSource	5
Abbildung 2.3.1: Projektzeitplan	13
Abbildung 3.1.1: Atmel ATmega1284P-PU	15
Abbildung 3.2.1: FTDI VDrive2 USB	16
Abbildung 3.3.1: Electronic Assembly eDIP240-7	17
Abbildung 3.4.1: CNC-Bausatz www.gocnc.de „HOBBY A4“	18
Abbildung 3.5.1: USOVO UNI1500 Steuerplatine	20
Abbildung 3.5.2: USOVO UNI1500 Pinbelegung	20
Abbildung 3.6.1: Parallelität der Achsen	22
Abbildung 3.6.2: Leerschritte	22
Abbildung 3.6.3: Umkehrspiel der Y-Achse	23
Abbildung 3.7.1: PWM Schrittmotor	24
Abbildung 3.9.1: Logo EAGLE	28
Abbildung 3.9.2: PCB-GCODE Menu	29
Abbildung 4.2.1: Controller Programmdateien	32
Abbildung 4.2.2: Initialisierung	32
Abbildung 4.2.3: Bitfeld mit zwei unsigned CHAR Variablen	34
Abbildung 4.2.4: Fehler einer gerasterten Linie (Wikipedia, 2012)	35
Abbildung 4.2.5: USB-Sequenz	36
Abbildung 4.2.6: uart_gets()	37
Abbildung 4.2.7: I2C-Kommunikation	38
Abbildung 4.2.8: Kartesisches Koordinatensystem	41
Abbildung 4.3.1: EA KitEditor Programmdateien	42
Abbildung 4.3.2: Stukrogramm PowerOn-Makro	43
Abbildung 4.3.3: Start des Controllers	44
Abbildung 4.3.4: Warnung	44
Abbildung 4.3.5: Menü-Struktur	45
Abbildung 4.4.1: Protokollrahmen	46
Abbildung 4.4.2: Protokollrahmen DC2	46
Abbildung 5.1.1: Blockschaltbild Steuerplatine	48
Abbildung 5.1.2: Stromverbrauch	48
Abbildung 5.2.1: Steuerplatine Designentwurf	49
Abbildung 6.2.1: Software & Versionen	53

Abbildung 9.1.1: Schaltplan der Steuerplatine 55

Abbildung 9.1.2: Schaltplan des eDIP240 mit USB Board 56

Abbildung 9.1.3: Schaltplan VDrive2 57

Abbildung 9.1.4: Gehäuse-Oberseite 59

Abbildung 9.1.5: Fertige Platine 60

## 1.5 Abkürzungsverzeichnis

AC	Alternating Current	Wechselstrom
BAUD		Einheit der Symbolrate
Brown-Out		Spannungsabfall
CAD	Computer Aided Design	Computergestützte Konstruktion
CNC	Computerized Numerical Control	Computergestützte Steuerung
DC	Direct Current	Gleichstrom
DIL	Double In Line	Bauform des Controllers
DRAM	Dynamic Random-Access Memory	flüchtiger, wiederbeschreibbarer Speicher
EEPROM	Electr. Erasable Programmable ROM	elekt. löschbarer Nur-Lese-Speicher
FLASH	Nicht-flüchtiger Speicher	Programmspeicher
gCode		Programmiersprache
Gerber		Dateiformat
HMI	Human Machine Interface	Mensch-Maschinen-Schnittstelle
HPGL	Hewlett Packard Graphic Language	Seitenbeschreibungssprache
I2C	Inter-Integrated Circuit	auch TWI, 2-Draht-Bus-System
JTAG	Joint Test Action Group	Debug-Schnittstelle
LC-Display	Liquid Crystal Display	Flüssigkristallanzeige
LCD	Liquid Crystal Display	
NC	Numerical Control	Numerische Steuerung
ROM	Read-Only-Memory	Nur-Lese-Speicher
RS-232	siehe UART	Serielle Schnittstelle
RS-485	siehe UART	Serielle Schnittstelle
SPI	Serial Peripheral Interface	Serielle Schnittstelle
SRAM	Static Random-Access Memory	nicht-flüchtiger Speicher (unter Ub)
TWI	Two Wire Interface	Auch I2C-Bus
UART	Univ. Asynchron Receiver Transmitter	Digitaler Schnittstellen-Standard
UBRRn	UART Baud Rate Register	Spezialregister des Controllers
VIA	Vertical Interconnect Access	Durchkontaktierung einer Platine



## 2 Projektbeschreibung

### 2.1 Motivation

Für die Herstellung einer Platine gibt es verschiedene Methoden. Das Ausdrucken auf Transferfolie, auf Belichtungsfolie oder die Ausgabe der Daten an eine Isolationsfräse. Als erstes muss zunächst der Schaltplan erstellt werden. Ist dieser erstellt, folgt als nächster Schritt das Erstellen des Layouts. Je nach verwendeter Elektro-CAD-Software kann dies auf verschiedene Weise erfolgen. Die hier verwendete Software „Einfach Anzuwendender Grafischer Layout-Editor“, kurz „EAGLE“, der Firma CADSoft<sup>6</sup>,

bietet hier die „AutoRoute“-Funktion. Diese ermöglicht es, das Layout, automatisch an Hand des Schaltplans zu designen.

Hier angekommen, stehen im Hobbybereich 3 grundsätzliche Methoden zur Herstellung der Platine zur Auswahl:

#### Das Ausdrucken des Layouts auf eine Transferfolie

Diese Methode funktioniert nur mit einem Laserdrucker. Hierbei wird das Layout in der höchsten Auflösung gedruckt. Die Transferfolie nimmt nicht den kompletten Toner auf, sodass der überschüssige Toner thermisch, mit dem Bügeleisen oder Ähnlichem auf eine unbeschichtete Platine übertragen werden kann. Nun ist diese vorbereitete Platine bereit zum Ätzen.

#### Das Ausdrucken des Layouts auf eine Belichtungsfolie

Im Gegensatz zur vorherigen Methode kann hier auch ein geeigneter Tintenstrahldrucker verwendet werden. Die bedruckte Folie wird auf einer Platine mit Fotobeschichtung ausgerichtet und mittels UV-Licht auf das Basismaterial belichtet. Im Anschluss an das Belichten erfolgt das chemische Fixieren und Entwickeln. In diesem Schritt wird die überschüssige Fotobeschichtung entfernt. Ab jetzt ist diese Platine ebenfalls bereit zum Ätzen.

#### Die Ausgabe der Daten für eine Isolationsfräse

---

<sup>6</sup> (CADSoft, 2)

---

Der Vorteil dieses Verfahrens gegenüber den beiden anderen liegt darin, dass hier keine Chemie zum Einsatz kommt. Das Layout wird hierbei nicht geätzt, sondern mit einer geeigneten Fräse und Werkzeug aus einer Kupfervollschichtplatine gefräst. Trotz dieses Vorteils, hat auch diese Methode einen nicht unerheblichen Nachteil, welcher sich bei größeren Layouts zeigt. Abhängig von der Platinengröße, den vorgenommenen Einstellungen und der Isolationsbreite, kann der Fräsvorgang mehrere Stunden in Anspruch nehmen. Auch der dabei entstehende Bohr- und Frässtaub muss abgeführt werden. Der entscheidende Vorteil dieser Methode ist die hohe Genauigkeit. Eine Platine mit beispielsweise 40 Bauteilen und einigen Durchkontaktierungen kann leicht auf über 200 **Bohrungen** kommen. Diese mit der Hand auszuführen erfordert eine hohe Konzentration sowie eine ruhige Hand.

Das Projekt CNC-Steuerung setzt an der Schnittstelle zwischen Elektro-CAD Software und einer Isolationsfräse an.

## 2.2 Aufgabenstellung

Die Aufgabenstellung für dieses Projekt ist es, Hard- und Software zu entwickeln, welche mindestens ein Ausgabeformat des EAGLE Elektro-CAD-Programms interpretieren kann und für eine 2D-CNC-Maschine ohne eigene Steuerung umsetzt. Um Flexibilität zu gewährleisten, sind die Daten dem Gerät über ein portables Speichermedium zuzuführen. Die Kommunikation und Bedienung der CNC-Steuerung soll über ein geeignetes Display erfolgen. Die Spannungsversorgung soll aus dem 230 V AC-Netz erfolgen. Aus Sicherheitsgründen ist die Versorgungsspannung auf Schutzkleinspannung zu transformieren und als Versorgung für die CNC-Steuerung zu verwenden. Die mittels EAGLE erstellten Bohr- und Fräsdaten können grundsätzlich in verschiedenen Formaten vorliegen, wie z.B. HPGL, Gerber, oder gCode. Da die aus EAGLE erzeugten Daten meist in dem Format gCode vor liegen, konzentriert sich dieses Projekt auf dieses Format.

Die Dauer der Projektarbeit beträgt maximal sechs Monate, Abgabetermin ist der **16.04.2012**.

## 2.3 Zeitplan

Abbildung 2.3.1: Projektzeitplan

## 2.4 Planung

Zu Beginn werden die Rahmenbedingungen und der Gesamtumfang des Projekts festgelegt. Dazu wird das Projekt in sieben Teilabschnitte gegliedert:

1. Auswahl der zu verwendenden Komponenten, Erstellung der Pläne und Aufbau eines geeigneten Entwicklungssystems
2. Aufbau der Kommunikation zwischen einem PC und dem Controller zum Steuern und Ausgeben von Meldungen
3. Ansteuerung der Schrittmotoren, zunächst für einen einzelnen Motor, im Folgenden dann für zwei Motoren parallel realisiert
4. Einlesen von Daten von einem USB-Stick sowie die Bereitstellung und Übergabe dieser Daten für weitere Funktionen implementiert
5. Auswertung der gCode-Daten aus den übergebenen Daten; Steuerung der daraus erforderlichen Aktionen umgesetzt
6. **Displayanbindung, Displayprogrammierung**
7. Umsetzung des Entwicklungssystems auf eine Platine und der Einbau in ein geeignetes Gehäuse

Die Dokumentation des gesamten Projektes läuft parallel zu diesen Schritten.

### 3 Hardware

#### 3.1 Controller

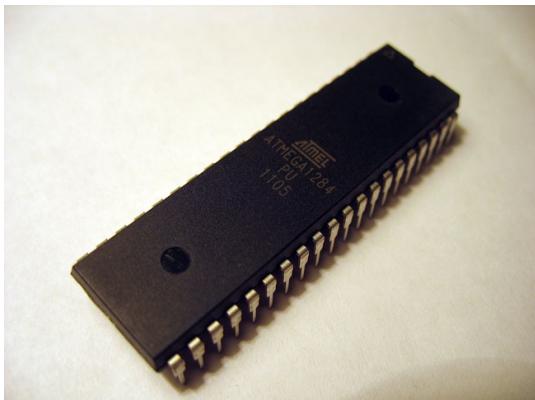


Abbildung 3.1.1: Atmel ATmega1284P-PU

Die Auswahl des Controllers, dem Kern des Projekts, ist von verschiedenen Faktoren abhängig. Als wichtigstes Auswahlkriterium gilt es, ein 8bit-Controller, der zum einen weit verbreitet und günstig ist, zum anderen einfach und unkompliziert mit dem AVR-Studio der Firma Atmel projektiert werden kann, zu finden. Außerdem kamen die Rahmenbedingungen wie zum Beispiel eine Versorgungsspannung

von 5 V DC und mindestens vier Schnittstellen, zwei davon UART, als eines der Hauptkriterien hinzu. Ebenso sollte der Controller mind. 16 I/Os zur Verfügung stellen. Aufgrund meiner bisherigen persönlichen, positiven Erfahrung, aus anderen Projekten, mit der ATmega-Serie der Firma Atmel beim Einsatz in anderen Projekten, fiel die Entscheidung auf den Atmel ATmega1284P-PU. Die Merkmale dieses Modells sind:

- 8bit-Controller
- 40 Pin DIL-Gehäuse
- bis zu 20 MHz CPU-Takt mit externem Quarz
- 32 I/Os
- 128kB Flash
- 16kB SRAM
- 4kB EEPROM
- 2x UART
- 1x SPI
- 1x TWI

Der ausgewählte Controller ist pinkompatibel zum ATmega32 und kann somit einfach auf dem in der Entwicklungsphase eingesetzten Pollin ATMEL Evaluations-Board<sup>7</sup> betrieben werden. Der Vorteil dieses Boards liegt darin, keine weitere Hardware zu benötigen. Alle wichtigen Komponenten zum Betrieb des Controllers sind auf der Platine vorhanden und

<sup>7</sup> (Pollin, 2007)

können über Steckbrücken zugeschaltet werden. Alle Pins sind über einen Wannenstecker herausgeführt und können nach Bedarf beschaltet werden.

Die Kosten für diesen Controller liegen mit der benötigten Außenbeschaltung bei rund 8 Euro<sup>8</sup>.

### 3.2 Speichermedium



Abbildung 3.2.1: FTDI VDrive2 USB

Zum Transport der am PC generierten Daten, sind zwei Varianten in der näheren Auswahl: SD-Karten und USB-Sticks. Beide haben Vor- und Nachteile. Für die SD-Karte spricht, dass sie sehr leicht und ohne großen technischen Aufwand an einen Controller angebunden werden kann. Die Schnittstelle einer SD-Karte entspricht dem Standard SPI. Der Nachteil der SD-Karte ist der verwendete Spannungspegel. Nach den SD Association<sup>9</sup>-SD Standards liegt dieser bei 3,3 Volt. Da der Controller und weitere Komponenten auf 5 Volt ausgelegt sind, ist hier zusätzliche Hardware zur Pegelwandlung nötig. Der USB-Stick kann im Gegensatz zur SD-Karte nicht direkt an den Controller angebunden werden. Hierzu ist ein zusätzlicher Schnittstellenbaustein nötig.

Der Vorteil der USB-Variante liegt darin, dass auch ältere PCs bzw. Notebooks über USB-Ports, jedoch nicht zwangsläufig auch SD-Kartenleser, verfügen. Der USB-Stick hat sich im Bereich der Datenmobilität klar durchgesetzt. Derzeit gibt es nur wenige Hersteller, welche die hier benötigten Schnittstellenkomponenten anbieten. Der führende Hersteller in diesem Sektor ist die Firma FTDI<sup>10</sup>. Das hier verwendete Modul Vinculum VDrive2 basiert auf dem USB-Kommunikationschip **VNC1L von FTDI**.

Das VDrive2-Modul wird mit der entsprechenden Firmware und der dazugehörigen Software „**Vinculum Firmware Customiser**“ konfiguriert. (Siehe Kapitel 3.9.2 **Vinculum VDrive2 Firmware 3.68**.) Er bietet zwei Schnittstellen auf der Hardwareebene, SPI und UART. Zunächst war geplant den VDrive2 über SPI einzubinden. Abweichend vom Standard arbeitet das integrierte SPI mit neun statt mit acht Datenbits. Dies hätte zur Folge gehabt, dass die Kommunikation zwischen VDrive2 und Controller nicht über das Hardware-SPI des Con-

<sup>8</sup> (Pollin, 2007)

<sup>9</sup> (The SD Association, 1)

<sup>10</sup> (FTDI Chip, 1)

trollers, sondern nur über ein bereitzustellendes Soft-SPI funktioniert hätte. Da zu diesem Zeitpunkt die genaue Auslastung des Controllers noch nicht abzusehen war, wurde der VDrive2 über UART angebunden.

### 3.3 Display



Abbildung 3.3.1: ELECTRONIC ASSEMBLY eDIP240-7

Schon zu Beginn des Projektes steht fest, wie die Kommunikation zwischen Mensch und Maschine aufgebaut sein soll. Als HMI war ein Display vorgesehen. Die Möglichkeit zur Steuerung der Maschine über ein Touch-Display ist der Firma ELECTRONIC ASSEMBLY GmbH zu verdanken. Dank der kostenfreien Überlassung eines eDIP240B-7LWTP kann dieses in das Projekt integriert werden. Bei dem überlassenen Display handelt es sich um ein monochromes LC-Display mit einer Auflösung von 240 x 128 Pixel. In diesem Display ist eine Touchfolie im 4-Wire-System integriert, welche direkt vom Display verwaltet und ausgewertet wird. Die Informationen der Touchfolie werden über die Schnittstellen des Displays bereitgestellt. Es stehen acht I/Os zur Verfügung sowie folgende vier unabhängige Schnittstellen:

- 1x RS-232
- 1x RS-485
- 1x SPI

1x TWI

Das Display wird mit der mitgelieferten Software „LCD-Tools“ und dem ebenfalls zur Verfügung gestellten USB-Programmer Board der Firma ELECTRONIC ASSEMBLY für dieses Projekt separat programmiert.

Die Kommunikation mit dem Controller erfolgt über I2C da diese noch am Controller verfügbar war, **mit dem SmallProtocol**. Auf dieses Protokoll wird im Kapitel 4.5 SmallProtocol näher eingegangen.

### 3.4 Portalfräse

Die eigentliche Portalfräse ist nicht Bestandteil des Projekts und wird daher nur kurz beschrieben. Das Gerät mit der Bezeichnung HOBBY A4 wurde als Bausatz der Firma **GO!CNC.de<sup>11</sup>** über die Homepage [www.gocnc.de](http://www.gocnc.de) bestellt.



Abbildung 3.4.1: CNC-Bausatz [www.gocnc.de](http://www.gocnc.de) „HOBBY A4“

Nach einer Lieferzeit von über sechs Wochen ist der Bausatz vom Autor aufgebaut und in Betrieb genommen worden. Der Bausatz ist durch den Einbau von drei Endschaltern sowie eines Werkzeulgängensensors modifiziert worden. Die mitgelieferte Schrittmotor-

<sup>11</sup> (goCNC, 1)

Treiberplatine „UNI1500“ der Firma USOVO ist im Originalzustand übernommen worden. Auf die Schnittstelle dieser Treiberkarte wird im nächsten Abschnitt näher eingegangen. Die mitgelieferte 5 mm-Holzplatte ist durch eine 8 mm-Aluminiumplatte ersetzt. Dies verleiht der gesamten Mechanik mehr Stabilität.

### 3.5 UNI1500

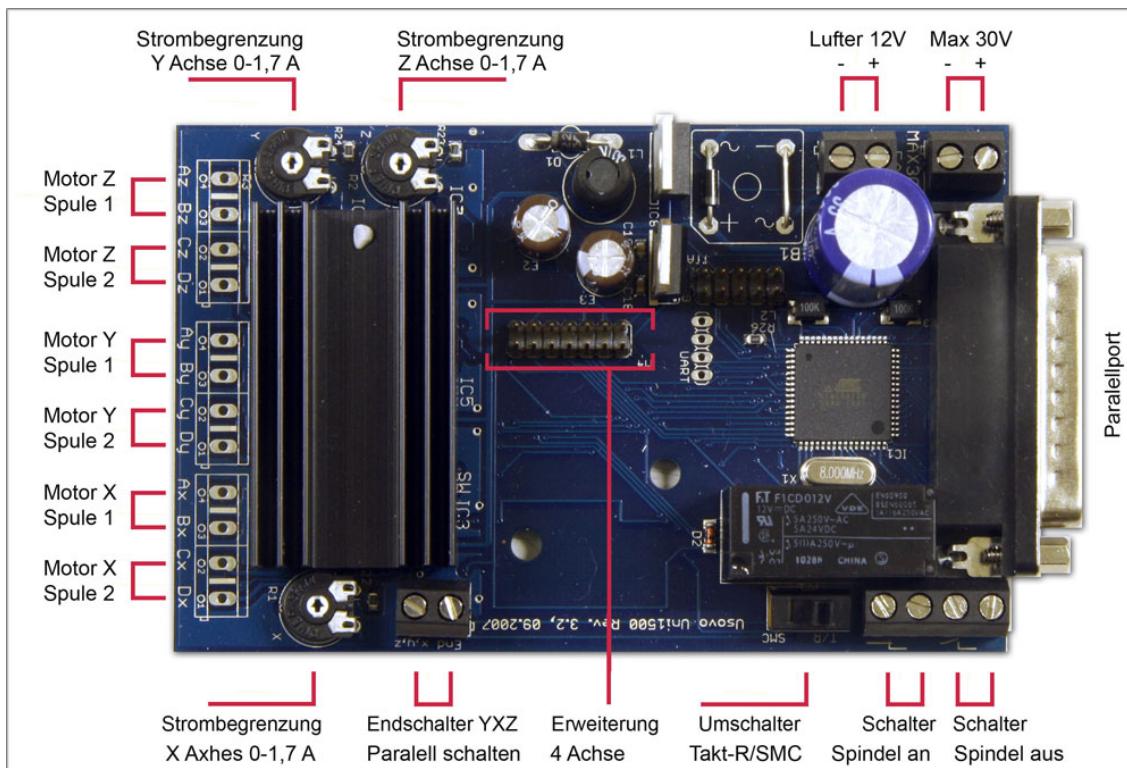


Abbildung 3.5.1: USOVO UNI1500 Steuerplatine

Die Steuerkarte ist auf Takt- / Richtungssignal eingestellt. WARUM IS DIE HIER DRIN?

PENS Daraus ergibt sich folgende Pinbelegung am Parallelport der Karte:

Pin 1	nicht verbunden	Pin 10	nicht verbunden
Pin 2	Richtung X	Pin 11	Referenzschalter X Y Z
Pin 3	Takt X	Pin 12	Referenz Achse 4
Pin 4	Richtung Y	Pin 13	nicht verbunden
Pin 5	Takt Y	Pin 14	Spindel Relais
Pin 6	Richtung Z	Pin 15	nicht verbunden
Pin 7	Takt Z	Pin 16	nicht verbunden
Pin 8	Richtung Achse 4	Pin 17	nicht verbunden
Pin 9	Takt Achse 4	Pin 18 - 25	Masse

Abbildung 3.5.2: USOVO UNI1500 Pinbelegung

Die Pots ausschreiben der Motorstrombegrenzung sind auf 1A eingestellt. Der Relaisausgang für die Spindelansteuerung ist aktuell nicht belegt, im Programm jedoch zur späteren Integration vorbereitet.

### 3.6 Genauigkeit

Die Genauigkeit einer CNC-Maschine ist abhängig von einer Vielzahl von Faktoren wie zum Beispiel dem mechanische Aufbau, der Steifigkeit der Konstruktion, des Lagerspiels, dem Führungsspiel, Toleranzen des Spindelsystems, der Hysterese der Referenzschalter, den verwendeten Motoren und nicht zuletzt von der eingesetzten Steuerung. In der CNC-Technik wird die Genauigkeit in zwei Gruppen unterteilt: Positionier- und Wiederholgenauigkeit. Unter der Positioniergenauigkeit versteht man das genaue Anfahren der Zielkoordinaten. Bei der Wiederholgenauigkeit geht es darum, mit möglichst geringer Abweichung, den selben Punkt mehrmals auf der selben Bahn anzufahren.

Die hier entwickelte CNC-Steuerung arbeitet intern nur mit Steps, dass heisst die eingelesenen Koordinaten werden zunächst in Steps entsprechend der Spindelauflösung wie folgt umgerechnet:

$$\text{Steps zur Zielkoordinate} = \frac{\text{Schritte pro Umdrehung} * \text{Zielkoordinate}}{\text{Steigung der Spindel}}$$

Die Steigung der hier verwendeten Gewindespindeln beträgt 1,5 mm pro Umdrehung. Die Schrittauflösung der Schrittmotoren beträgt 144 Steps pro Umdrehung. Für eine Zielkoordinate von X=10 ergeben sich 960 Steps für diese Achse. Für die Berechnung der tatsächlich zu bewegenden Steps gilt es dann natürlich zu beachten, ob es sich um Absolute oder Relative Koordinaten handelt. Hieraus geht auch hervor, dass die kleinste Recheneinheit für einen Position, ein Step ist. Dies bedeutet wiederum, dass die Genauigkeit der Berechnung von der Spindelsteigung und der Schrittauflösung der Motoren abhängt. Für die hier eingesetzte Portalfräse lässt sich eine theoretische Positioniergenauigkeit von **10,42µm** errechnen.

Ein weiterer Punkt, der Einfluss auf die Positionier- und Wiederholgenauigkeit nimmt, ist das so genannte Lagerumkehrspiel. Dies bezeichnet das Spiel zwischen Spindel und Mutter bei Richtungsumkehr bis die Umdrehungen der Spindel in eine Bewegung der Achse umgesetzt werden. Diese Schritte nennt man Leerschritte.

### 3.6.1 X- und Y-Achsen

Zur Messung der Parallelität der Achsen zum Werkstücktisch wird eine Messuhr mit einer Auflösung von 10um verwendet. Beim Überfahren des Tisches ergeben sich folgende Abweichungen:

Fahrweg	Ausschlag Messuhr
X:0mm – X:160mm bei Y:0mm Z: 20mm	+ / - 0,02mm Sonderzeichen für Vorzeichen
X:0mm – X 160mm bei Y:200mm und Z: 20mm	+ / - 0,01mm
Y 0mm – Y 200mm bei X:0mm Z: 20mm	+/- 0,03mm
Y 0mm – Y 200mm bei X:160mm Z: 20mm	+ / - 0,02mm
X:0mm – X:160mm Y:0mm – Y:200mm bei Z: 20mm	+ / - 0,03mm
X:0mm – X:160mm Y:200mm – Y:0mm bei Z: 20mm	+ / - 0,03mm

Abbildung 3.6.1: Parallelität der Achsen

Das Lagerumkehrspiel der Y- Achse liegt außerhalb eines akzeptablen Bereichs. Dieses Spiel wird durch schrittweises Takten der Motoren ermittelt. Dazu wird das Werkzeug auf die Werkstückoberseite gefahren, soweit abgesenkt, bis Material abgetragen wird. Von jetzt an wird die zu untersuchende Achse schrittweise, in entgegengesetzter Richtung der letzten Fahrt, getaktet. Dabei werden die Schritte gezählt, bis wieder Material abgetragen wird. Hierbei zeigt sich:

Achse	Leerschritte
X-Achse	5
Y-Achse	42

Abbildung 3.6.2: Leerschritte

Die X-Achse ist mit 5 Leerschritten, mit 0,052mm, noch in einem akzeptablen Bereich. Die Y-Achse dagegen liegt mit 42 Leerschritten außerhalb eines nutzbaren Bereichs. Das Umkehrspiel entspricht einer Distanz von 0,438mm. An dieser Stelle ist zu überlegen, einen Kompensationsfaktor softwareseitig einzubinden, da mit dieser Abweichung das fräsen kleiner Leiterbahnen unmöglich ist. Dies wird besonders deutlich wenn man ein am Stück gefrästes Quadrat mit einer Seitenlänge von 10mm genauer betrachtet. Siehe Abbildung 3.6.3: Umkehrspiel der Y-Achse. Hier zeigt sich an der rechten, oberen Ecke des linken Quadrates ein Überfräsen, im rechten Quadrat ein Unterfräsen. Dieser Versatz entspricht den 42 Leerschritten.

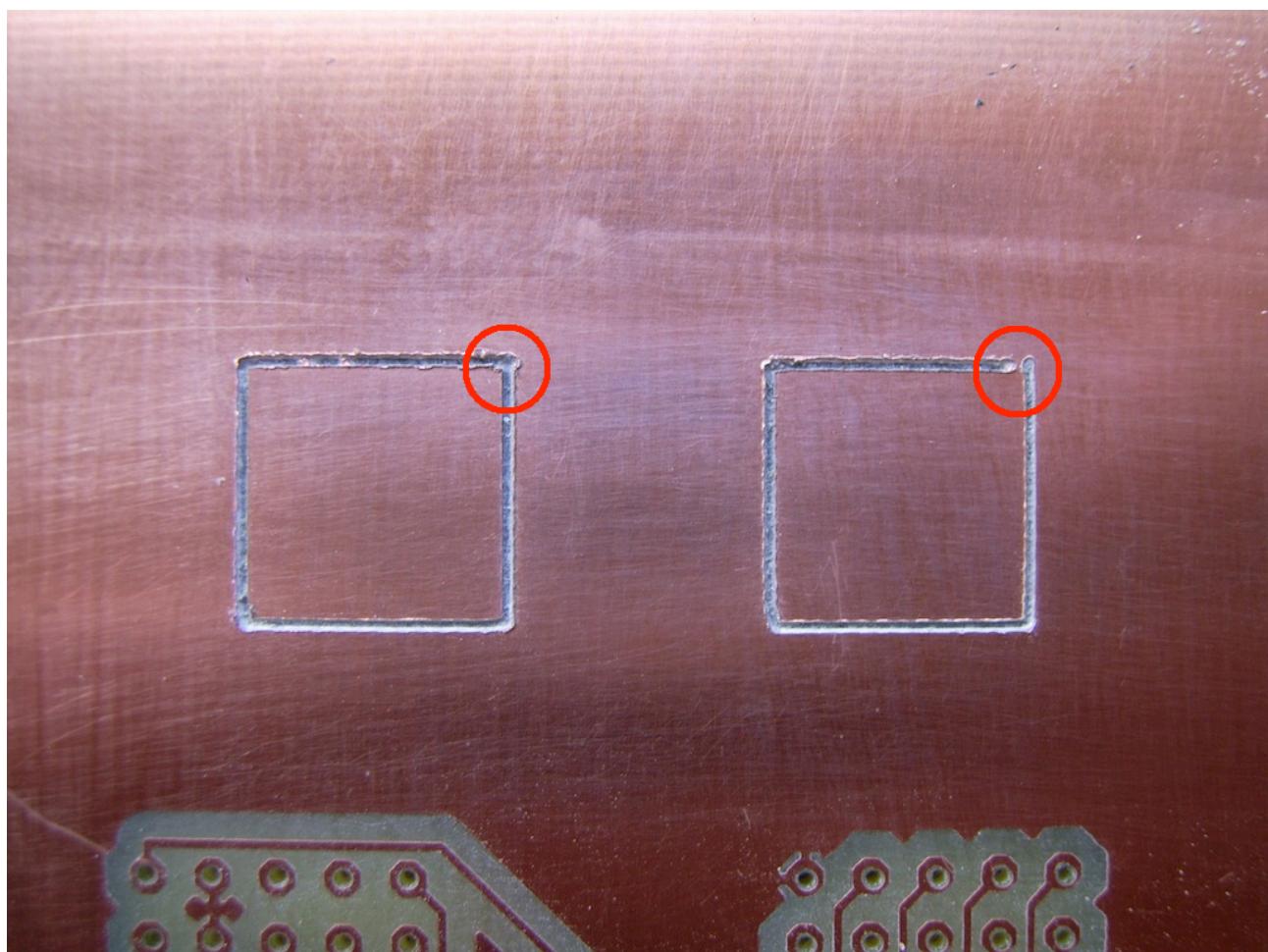


Abbildung 3.6.4: Umkehrspiel der Y-Achse

Das Bohren der Platine ist an dieser Stelle unkritisch, da die Bohrungen von oben nach unten berechnet werden. Die Richtungsumkehr findet erst am Ende der Platine statt. Im nächsten Schritt wird die X-Achse verfahren, die Y-Achse startet wieder bei Y=0.

### 3.6.2 Z-Achse

Das Anfahren verschiedenster Positionen der Z-Achse zeigt, dass die Positioniergenauigkeit, wie auch die Wiederholgenauigkeit der Z-Achse größer ist, als die Auflösung der eingesetzten Messuhr. Somit ist es nicht möglich, an dieser Stelle eine belegbare Aussage über die Genauigkeit der Z-Achse zu treffen. Die reale Abweichung ist kleiner 0,01mm. Eine möglicher Grund für diese hohe Genauigkeit ist das bei der Z-Achse vernachlässigbare Lagerumkehrspiel. Da diese Achse durch die Schwerkraft nach unten gezogen wird, liegt das Spindellager immer an der unteren Steigungsflanke der Spinde auf. Dadurch wird jeder Schritt des Z-Achsenmotors direkt in einen Bewegung umgesetzt.

## 3.7 Geschwindigkeit

Die Bewegungsgeschwindigkeit einer Achse errechnet sich aus der Drehzahl, sowie aus der jeweiligen Steigung der Spindel pro Umdrehung. Wie bereits erläutert, haben die hier eingesetzten Spindeln eine Steigung von 1,5mm pro Umdrehung. Die Bestimmung der Drehzahl der Spindel hingegen ist etwas aufwändiger, da hier in Schritten gerechnet wird. Eine Umdrehung entspricht 144 Schritten. Die Ansteuerung der Schritte erfolgt über eine Impulsvorgabe pro Schritt.

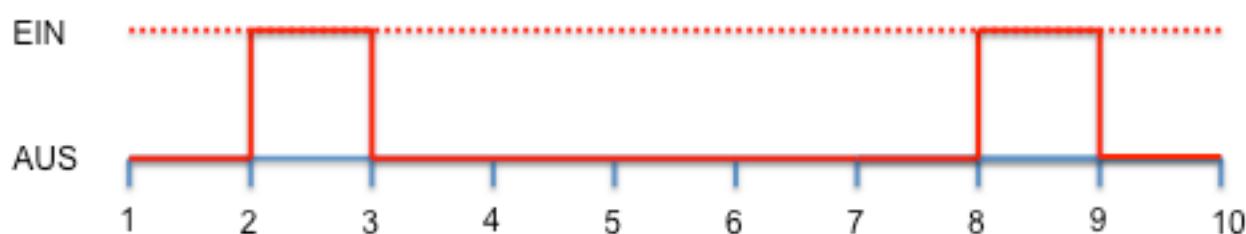


Abbildung 3.7.1: PWM Schrittmotor

$$\text{PulsPauseVerhältnis} = \frac{tEIN}{(tAUS + tEIN)}$$

Aus diversen Versuchen in der Entwicklungsphase, hat sich gezeigt, dass mit einem Puls-Pause-Verhältnis von 16,66% ein optimaler, ruhiger Lauf der Schrittmotoren möglich ist. Bei einem größeren Verhältnis kam es zum Brummen bis hin zum totalen Stillstand der Motoren.

Die Vorgabe der Geschwindigkeit erfolgt durch die Vorgabe **der tAUS-Zeit**. Dadurch ist bei der Berechnung der tatsächlichen Zykluszeit pro Schritt, **der tAUS-Zeit 1/5 der tAUS-Zeit** auf zuaddieren. So wird aus einer tAUS-Zeit von **800us** eine Zykluszeit von 960us pro Schritt. Die Umrechnung der Geschwindigkeit in die Vorgabe der tAUS-Zeit ist an folgender Formel verdeutlicht:

$$tAUS \text{ in Sekunden} = \frac{\text{Steigung der Spindel in mm} * 60 * 5}{\text{mm pro Minute} * \text{Schritte pro Umdrehung} * 6}$$

$$\text{mm pro Minute} = \frac{\text{Steigung der Spindel in mm} * 60 * 5}{tAUS \text{ in Sekunden} * \text{Schritte pro Umdrehung} * 6}$$

Anhand dieser Formeln errechnet sich für eine Geschwindigkeit von 254mm/min eine tAUS-Zeit **von 2051us**. Aus ebenfalls zahlreichen Versuchen in der Entwicklung zeigte sich einen maximalen tAUS-Zeit von **750us** als absoluter Grenzwert. Dies entspricht einer maximalen Geschwindigkeit **von 694 mm/min**. Wird eine kleinere tAUS-Zeit gewählt, kommt es bei der Y-Achse zum überspringen von Schritten. Dies hat dann eine undefinierte Position der Achse zur Folge. Es muss manuell eine Referenzfahrt durchgeführt werden. Dies hat wiederum den Abbruch des aktuellen Programms zur Folge.

### 3.8 Werkzeugdrehzahl

Zur **Drehzahl des Werkzeugs kann gesagt werden: das sich das Ergebnis mit höherer Drehzahl verbessert**. Eine Drehzahl von **25.000rpm** hat sich für ein saubereres Bohrbild bewährt. Beim Fräsen hingegen ist die Drehzahl sehr stark vom verwendeten Werkzeug abhängig. Auch der Verschleiß des Werkzeugs ist zu berücksichtigen. Ein sauberer Rand einer Bahn, ohne Grat, ist bei einem Gravierstichel auch von der Eintauchtiefe abhängig. Bei einem 0,5mm 60° Stichel zeigt sich bei einer Eintauchtiefe von 0,25mm und einer Drehzahl von 33.000rpm eine Gratfreie Bahn. Mit einem 0,1mm 45° Stichel hingegen wird das Ergebnis erst bei einer Eintauchtiefe von 0,35 mm und einer Drehzahl von **35.000rpm** akzeptabel. An dieser Stelle sind einige Probebahnen, mit unterschiedlichen Einstellungen, vor dem produktiven Fräsen angeraten.

## 3.9 Hardware- & Softwarekonfiguration

### 3.9.1 Atmel ATmega1284P-PU Fuses & Lockbits

Die Steuerung des Controllers und dessen Anlaufverhalten wird über spezielle Bits, den sog. Fusebits, festgelegt. Die Clock Fusebits sind so gesetzt, dass der externe 18,432MHz Quarz 16k Zyklen plus 14 Zyklen & 65ms Zeit zum einschwingen hat. Diese Einstellung ist dem Datenblatt des ATmega1284P-PU Kapitel 7.4 Full Swing Crystal Oscillator zu entnehmen. **Die Entscheidung einen sog. BAUD-Quarz mit krummer Frequenz einzusetzen, ist blablabladhskdhershtjleshtjkrlrshdfgkjh Aufgrund der hohen BAUD des VDrive2.** Dieser ist über UART1 angebunden. UART0 ist für die Kommunikation mit einem PC oder ähnlichem vorgesehen. Da die UART-Schnittstelle asynchron arbeitet, ist die genaue Frequenz des Senders (TX) und des Empfängers (RX) von sehr großer Bedeutung. **Lediglich das Startbit teilt den Beginn einer Übertragung mit und erzeugt einen synchronen Start.** Danach werden die Bits anhand der eigenen Frequenz, der Sample-Rate gelesen oder gesendet. Ein maximaler Frequenzversatz von 3%<sup>12</sup> liegt noch in einem akzeptablen Bereich. Ist der Versatz größer 3% ist eine einwandfreie Übertragung nicht mehr möglich. Die Ablaufsteuerung des UART wird von 4 sog. Special Function Registern übernommen. Eines dieser Register ist das **UART BaudRateRegister**. Es wird zur Bestimmung der Kommunikationsgeschwindigkeit, der sog. **BAUD**, mit einem Teilerwert belegt. Gemäß Datenblatt<sup>13</sup> errechnet sich die BAUD und der Teilerwert für UBRRn wie folgt:

$$BAUD = \frac{fOSC}{16 * UBRRn} - 1$$

$$UBRRn = \frac{fOSC}{16 * (UBRRn + 1)}$$

$f_{osc}$  = Oszillator- oder Quarz-Frequenz

UBRRn = Teiler des UARTr

BAUD = Schrittgeschwindigkeit

n = Index der Schnittstelle

<sup>12</sup> (MAXIM, 2003)

<sup>13</sup> sgdgfdsgfsd

---

Mit einem  $f_{osc} = 20$  MHz Quarz errechnet sich für den Teiler ein Wert von 4,425. Dieser Wert muss zunächst gerundet werden, um als Integer in UBRRn geladen werden zu können. Die Gegenrechnung mit UBRR1 = 4 ergibt eine BAUD von 250kbps. Der hieraus resultierende Frequenzversatz von mehr 8 % liegt oberhalb des zulässigen Versatzes von 3%. Mit dem hier verwendeten, 18,432MHz, Quarz ergibt sich rechnerischer Versatz von 0%. Auch die UART0, mit einer BAUD von 9,6kbps, hat mit diesem Quarz einen Versatz von 0%.

Der Watchdog ist nicht aktiviert. Bei einem Absturz des Programms hätte dies, wie bei einem Reset, der Neustart des Programms zu Folge. Nach der Systeminitialisierung wartet das Programm auf Eingaben durch den Benutzer und ist somit in einem stabilen Zustand. Außerdem ist eine automatischer Reset durch den Watchdog während dem Debuggen nicht von Vorteil, da hier die Gründe für den Absturz nachträglich nicht mehr ersichtlich sind. Die Spannungsversorgung ist für den stationären Betrieb ausgelegt, daher wird nicht mit einem „Brown-Out“, dem absinken der Versorgungsspannung, gerechnet. Daher wird hier auf die Brown-Out-Detection verzichtet. Die Lockbits sind ebenfalls deaktiviert. Diese Steuern den Schreib-Lese-Schutz des Controllers. Die SPI- und JTAG-Schnittstelle sind aktiviert, ebenso wie der On-Chip-Debug-Mode. Dies ermöglicht das schrittweise Ausführen einzelner Befehle direkt im Controller, das sog. Source-Stepping. Die Boot-Sector-Size ist auf dem Standartwert von 4096 Byte belassen. Aus diesen Einstellungen ergeben sich folgende Fuses und Lockbits:

- Extended: **0xFF**
- High: **0x19**
- Low: **0xFF**
- Lockbits: **0xFF**

### 3.9.2 Vinculum VDrive2 Firmware 3.68

Als Ausgangsdatei für die individuell angepasste Firmware wird die Standard-Firmware-File VDAP-ROM 3.68 verwendet. Diese wird mit dem Vinculum Firmware Customiser wie folgt konfiguriert:

- Baud Rate: **230400 bps**
- Data Bits: **8-Bit**
- Stop-Bit: **1-Bit**
- Parity & Flow-Controll: **NONE**
- Input Mode **IPA-Mode (ASCII)**
- Command Mode **SCS (Short Command Set Mode)**
- LEDs Flash on Power-on: **Aktive**
- Show Prompt: **Aktive**
- Firmware Code **003**

### 3.9.3 EAGLE

„EAGLE ist ein EDA-Programm der Firma CadSoft zur Erstellung von Leiterplatten. Der Name ist ein Initialwort, gebildet aus Einfach Anzuwendender Grafischer Layout-Editor. Eagle und die Unternehmung CadSoft wurden im September 2009 an Premier Farnell verkauft, einen Lieferanten elektronischer Bauteile.“<sup>14</sup>

Der Umgang mit dieser Software wird als bekannt voraus gesetzt.

---

<sup>14</sup> (Wikipedia, 2012)

### 3.9.4 PCB-GCODE-ULP 3.5.2.11

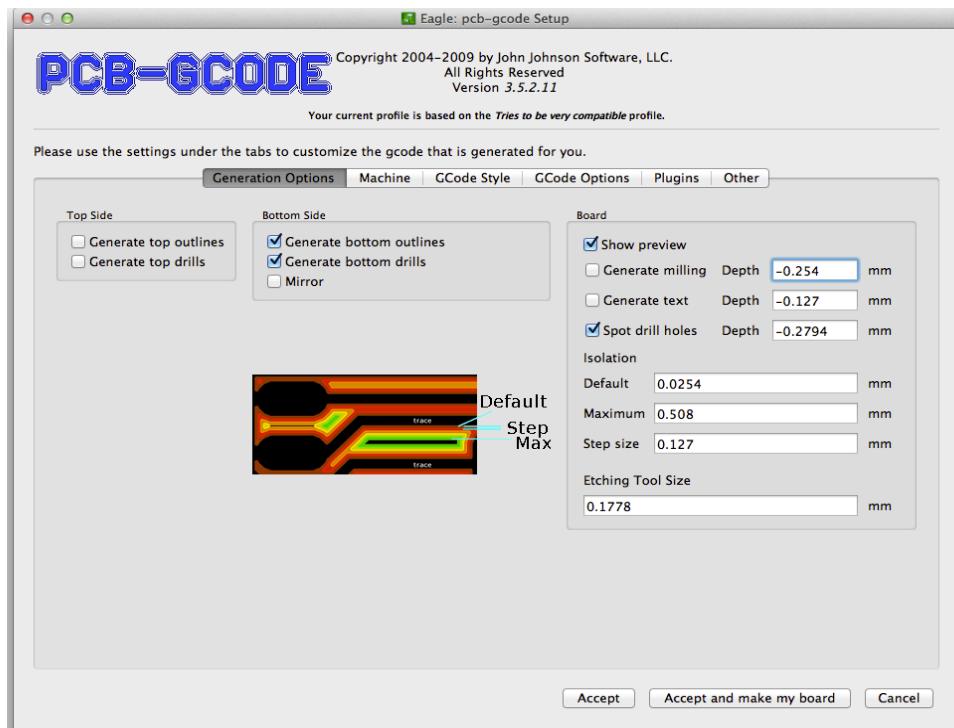


Abbildung 3.9.1: PCB-GCODE Menu

Ein Universal Language Programm, kurz ULP, ist ein in C programmiertes Unterprogramm welches direkt aus EAGLE Board heraus gestartet wird. Das hier verwendete ULP PCB-GCode<sup>15</sup> erzeugt aus dem erstellen Board-Layout je nach Einstellungen, eine oder mehrere gCode-Dateien. Diese können zum Beispiel für ein das Layout der Oberseite oder der Unterseite sein. Abweichend von den Default-Werten des ULP wurde folgendes eingestellt und optimiert:

#### Generation Options:

- Etching Tool Size: **0.254mm** Durchmesser des Gravierstichels

#### Maschine:

- |                 |                   |                              |
|-----------------|-------------------|------------------------------|
| ➤ Z-High:       | <b>16mm</b>       | Warteposition über Platine   |
| ➤ Z-Up:         | <b>2.54mm</b>     | Arbeitsposition über Platine |
| ➤ Z-Down:       | <b>- 0.1778mm</b> | Eintauchtiefe beim fräsen    |
| ➤ Z-Drilldepth: | <b>- 0.75mm</b>   | Bohrtiefe                    |
| ➤ Position Z:   | <b>40mm</b>       | Werkzeugwechselhöhe          |

GCode Style: **generic.pp** Standart gCode

#### GCode Options:

- Zeilennummer Format: **N%05d** Schrittnummer

<sup>15</sup> (Johnson, 2009)

## File Naming:

- |                    |                           |                  |
|--------------------|---------------------------|------------------|
| ➤ Wort für Etch:   | <b>e</b>                  | fräsen           |
| ➤ Wort für Drill:  | <b>d</b>                  | bohren           |
| ➤ Wort für Top:    | <b>t</b>                  | Oberseite        |
| ➤ Wort für Bottom: | <b>b</b>                  | Unterseite       |
| ➤ Extension:       | <b>tap</b>                | Dateierweiterung |
| ➤ Filename:        | <b>\$SIDE\$FILE.\$EXT</b> |                  |

Daraus ergibt sich z.B. für den Bohrplan der Unterseite folgender Dateiname:

**bd.tap**

## 4 Software

### 4.1 Einleitung

Das Projekt CNC-Steuerung wurde in der Hochsprache C programmiert. Die Entwicklungsumgebung war die bereits erwähnte Software der Firma Atmel, AVR-Studio 5.1. Zum flashen und debuggen der Software kam der ebenfalls von Atmel entwickelte Debugger AVR JTAGICE3 zum Einsatz. Als primäres Entwicklungsboard diente das Pollin Evaluationsboard V2.0.1<sup>16</sup> der Firma pollin.de, sowie ein Breadboard der Firma Wanjie. Ein Breadboard ist ein Steckbrettsystem, mit einem Rastermaß von 2,54mm. Standardbauteile, die diesem Rastermaß entsprechen, können ohne Löten, über entsprechende Kabelsteckbrücken miteinander verbunden werden. In der ersten Entwicklungsphase wurde eine Atmel ATmega8, später ein ATmega32 und in der finale Version, ein ATmega1284P als Zielsystem ausgewählt. Diese Abstufung ist auf die unterschiedlichen Entwicklungsphasen zurück zuführen. Am Anfang der Entwicklung wurden nur die Motoren angesteuert, hierzu reichte der ATmega8, mit einer UART als Debug-Ausgabe zum PC aus. In der nächsten Entwicklungsstufe wird jedoch der VDrive2 über diese UART implementiert. Um dennoch die Möglichkeit zum Debuggen zu erhalten, wird das ganze System auf einen ATmega32 portiert der eine JTAG Schnittstelle zur Verfügung stellt. In der vorletzten Entwicklungsstufe wurden alle Funktionen zusammen geführt. Es war nun erforderlich, zur Ablaufsteuerung ist es nun erforderlich dem Controller Befehle zuzuführen. Diese Funktion war ursprünglich für die UART0 vorgesehen. Somit wurde das System auf den ATmega1284P-PU portiert und die Schnittstellen wie folgt belegt:

- UART0      Ausgabe von Meldungen und Empfang von Steuerbefehlen vom PC
- UART1      VDrive2 USB
- PinPortA     goCNC Portalfräse

Im letzten Schritt der Entwicklung wurde das Display implementiert. Dieses ist über das TWI angebunden.

---

<sup>16</sup> (Pollin, 2007)

## 4.2 Controller-Programm

C-Programm-Dateien	Header-Dateien	Funktion	Autor
main.c	globdef.h	Hauptprogramm	Dennis Hohmann
uart.c	uart.h	UART-Library	Peter Fleury <a href="http://jump.to/fleury">http://jump.to/fleury</a> Dennis Hohmann
i2cmaster.c	i2cmaster.h	I2C-Library	Peter Fleury <a href="http://jump.to/fleury">http://jump.to/fleury</a>
eDIP240.c	eDIP240.h	Display-Routinen	Dennis Hohmann
vnc1l.c	vnc1l.h	USB-Routinen	Dennis Hohmann
gcode.c	gcode.h	gCode-Routine	Dennis Hohmann
gocnc.c	gocnc.h	Bewegungsroutinen	Dennis Hohmann

Abbildung 4.2.1: Controller Programmdateien

Maschine initialisieren
I/O Ports definieren DDRA & DDRD
Zeit bis Peripherie bereit ist _delay_ms(500)
UART's initialisieren UART0 19,2 kbs UART1 230.4 kbs
I2C initialisieren Master-Slave Mode 100kHz
IRQ initialisieren INT2 auf steigende Flanke
Statusbits löschen
Ausgabe "Init OK"
IRQ aktivieren

Beim Systemstart wird als erste Aktion, die I/O Register des Controllers gesetzt. Im nächsten Schritt wird 500ms gewartet, bis die übrige Peripherie ebenfalls bereit ist. Diese Zeit ist besonders für den VDrive2 wichtig, da ein zu frühes Ansprechen diesen im Anlauf unterbricht. Nach Ablauf der Wartezeit werden die seriellen Schnittstellen mit der entsprechenden BAUD initialisiert, im Anschluss folgt der Datenbus I2C. Ist auch diese Schnittstelle erfolgreich initialisiert, wird das Interrupt-Control-Register des ATmega1284P mit der gewünschten Betriebsart geladen. Diese sieht einen Interrupt-Auslösung bei fallender Flanke, also „falling edge“ vor (→ 4.2.6). Zuletzt werden noch die entsprechenden Statusbits im Bitfeld **M\_FLAGS** gesetzt (→ 4.2.1).

Abbildung 4.2.2: Initialisierung

Nun ist der Controller bereit und gibt dieses als Statusmeldung an das Display aus. Ab jetzt sind die Interrupts aktiv, der Anlauf ist beendet. Sollte einer dieser Initialisierungsschritte nicht erfolgreich verlaufen, wird die entsprechende Meldung über die bereits zur Verfügung stehenden Schnittstellen ausgegeben.

Ab jetzt wird die main-Schleife zyklisch durchlaufen. Nach jedem Durchlauf wird das **M\_FLAGS.LIFEBIT** getoggelt. Der Zustand des Bits wird über eine, am Pin D5 angegeschlossene LED angezeigt und dient der optischen Kontrolle des Programmablaufs.

Es werden folgende Bits aus **M\_FLAGS** zyklisch ausgewertet und die entsprechenden Aktionen eingeleitet.

#### GCODE\_FILESET

Ist dieses Bit gleich 1, wird die, in der globalen Variablen FILENAME definierte Datei vom USB-Stick geöffnet.

#### M\_FLAGS->TOOL\_Change

Dieses Bit zeigt durch den Zustand 1 an, dass das Werkzeug gewechselt werden muss. Es wird die entsprechende Routine zum Werkzeugwechsel geladen. Nach erfolgreichem beenden der Routine wird das Bit gelöscht.

#### AUTO\_Action

Das AUTO\_Action-Bit signalisiert, dass der Automatik-Betrieb läuft.

#### ABORT\_Action

Dieses Bit kann zusätzlich zum AUTO\_Action-Bit gesetzt sein, jedoch ist das ABORT\_Action-Bit dominant ausgeführt und überschreibt das AUTO\_Action-Bit zum Ende des aktuellen Zyklus.

#### EDIP\_NEWPOS

Dieses Bit gibt an, ob sich die Position einer beliebigen Achse im letzten Zyklus verändert hat. Wenn dies der Fall ist, wird die Routine zur Aktualisierung der Positionsdaten im Display aufgerufen. Dieses Bit wird durch die Routine wieder zurück gesetzt.

#### 4.2.1 Globale Definitionsdatei globdef.h

Dieses Header-File ist die zentrale Definitions-Datei. Hier wird die globale Struktur **M\_FLAGS** deklariert. Eine Struktur ist ein Datensatz der aus verschiedenen Werten und Formaten zusammengesetzt sein kann. Die hier verwendete Struktur besteht größtenteils aus Bitfeldern. Diese haben den Vorteil, dass die Größe einer Variablen manuell festgelegt werden kann. Standardmäßig werden für eine Variable vom Datentyp *char* acht Bit reserviert. Somit kann diese einen Wert von 0 bis 255 bei *unsigned* bzw. -128 bis 127 bei *signed* annehmen. Der Unterschied zwischen den Definitionen *signed* und *unsigned* liegt darin das *signed* Variablen positive, negative Werte oder „0“ annehmen können. *Unsigned* können nur positiv oder „0“ sein. Ist die Variable als Bitfeld mit einer Länge von ein deklariert, werden zwar immer noch acht Bit reserviert, jedoch nur noch ein Bit tatsächlich verwendet. Hier kann die Variable nur noch den Wert „0“ oder „1“ annehmen. Der reservierte Bereich wird mit der nächsten Variablen aufgefüllt.

Speicheradresse	0x15	0x14	0x13	0x12	0x11	0x10	0x09	0x08	0x07	0x06	0x05	0x04	0x03	0x02	0x01	0x00
<b>unsigned CHAR x (8 Bit)</b>																
<b>unsigned CHAR y (8 Bit)</b>																
<b>unsigned CHAR x als Bitfeld Länge 1</b>																
<b>unsigned CHAR y als Bitfeld Länge 1</b>																

Abbildung 4.2.3: Bitfeld mit zwei *unsigned CHAR* Variablen

Mit dieser Methode kann gerade bei der Verwendung mehrerer Variablen sehr viel Platz gespart werden. Acht Variablen vom Typ *unsigned char* benötigen 64 Bit Speicherplatz. Wird jeweils nur ein Bit ausgewertet, sind 56 Bit „verschwendet“. Definiert man diese acht *unsigned char* Variablen in einem Bitfeld je mit der Länge eins, so benötigen diese nur noch acht Bit Speicherplatz.

Diese Definitionsdatei enthält des Weiteren die Frequenzen F\_CPU (Quarz-Frequenz) und die BAUDs der verwendeten Schnittstellen, UART0, UART1 und I2C. Auch die maximalen Positionen der Achsen sowie die Offset-Position des Werkzeuglängen-Sensors sind hier festgelegt.

#### 4.2.2 Schrittmotor-Handling gocnc.c

Mit dieser Bibliothek werden alle notwendigen Funktionen für die Bewegungen der Achsen zur Verfügung gestellt. Die kleinste Verfahr-Einheit einer Achse ist ein einzelner Step. Dieser wird mit einer vorher festgelegten Geschwindigkeit in eine definierte Richtung mit der Funktion `axis_move_single(int8_t AxisSelect,int32_t AxisGoto,uint16_t AxisSpeed)` ausgeführt. Die Variable `AxisGoto` ist mit der Zielposition in Steps zu belegen. Es wird intern die Differenz zwischen Ist- und Soll-Position ermittelt und die entsprechende Achse mit dieser Differenz verfahren. Die Umrechnung der Koordinaten von  $\mu\text{m}$  in Steps übernimmt die Funktion `um_to_steps(int32_t umGoTo)`, in andere Richtung für die Ausgabe der Position werden die Steps mit der Funktion `steps_to_um(int32_t steps)` zurück in  $\mu\text{m}$  gerechnet.

Besonderes Augenmerk gilt dem Zusammenspiel der X- mit der Y-Achse. Das Fahren einer Diagonalen im  $45^\circ$  Winkel stellt für den Controller keine große Schwierigkeit dar. Es werden beide Achsen parallel mit der übergebenen Geschwindigkeit verfahren. Dies ist zum schnellen Positionieren über der Platine ausreichend. Wenn es darum geht, geometrische Formen zu fahren, wie zum Beispiel ein Diagonale mit beliebigem Winkel erreicht man mit Geräten, die einem Raster unterliegen, sehr schnell an die Grenzen der Auflösung.

Eine einfache Methode zur Berechnung einer Diagonalen mit variabler Steigung ist der Bresenham-Algorithmus. Dieser Algorithmus wird dazu verwendet, Geraden oder Kreise so zu zerlegen, dass diese auf Rasterzeichengeräten mit möglichst geringer Abweichung dargestellt werden können. Im Prinzip wird die Abweichung zwischen der tatsächlichen Linie und dem Mittelpunkt der umliegenden Pixel in einem Abschnitt ermittelt. Es wird dann das Pixel mit der kleinsten Abweichung ausgewählt.

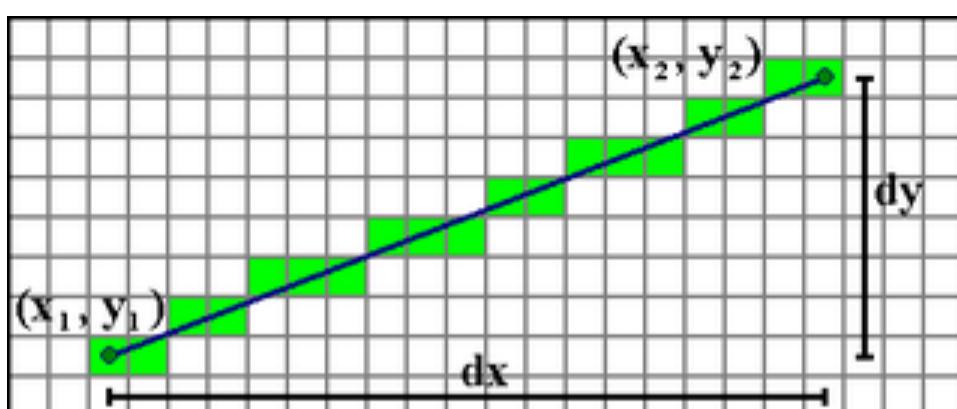


Abbildung 4.2.4: Fehler einer gerasterten Linie (Wikipedia, 2012)

Es wird zwischen zwei Gruppen von Bewegungen unterschieden, zum einem relative, zum anderen absolute Bewegungen. Der Unterschied dieser Gruppen liegt nur in der Berechnung der zu fahrenden Strecke.

Die Berechnung der Geschwindigkeit wird durch die Funktion `axis_set_speed(uint8_t speed)` übernommen.

#### 4.2.3 USB-Kommunikation vnc1l.c

Wird ein USB-Stick gesteckt und korrekt erkannt, kann eine entsprechende Datei aus dem ROOT-Verzeichnis von dem Programm mit der Funktion `usb_open_file(FILENAME)` geöffnet werden. Dateien in Unterverzeichnissen werden aktuell nicht unterstützt. Ist die Datei erfolgreich geöffnet, stehen zwei weitere Funktionen zur Verfügung:

`usb_close_file(FILENAME)` schließt die angegebene Datei wieder.

`usb_get_block(int8_t byte)` gibt die übergebene Anzahl an Bytes an die globale Variable `STRING` zurück. Von dort kann der `STRING` mit der Funktion `usb_get_command(void)`, wie im Struktogramm dargestellt, ausgewertet werden.

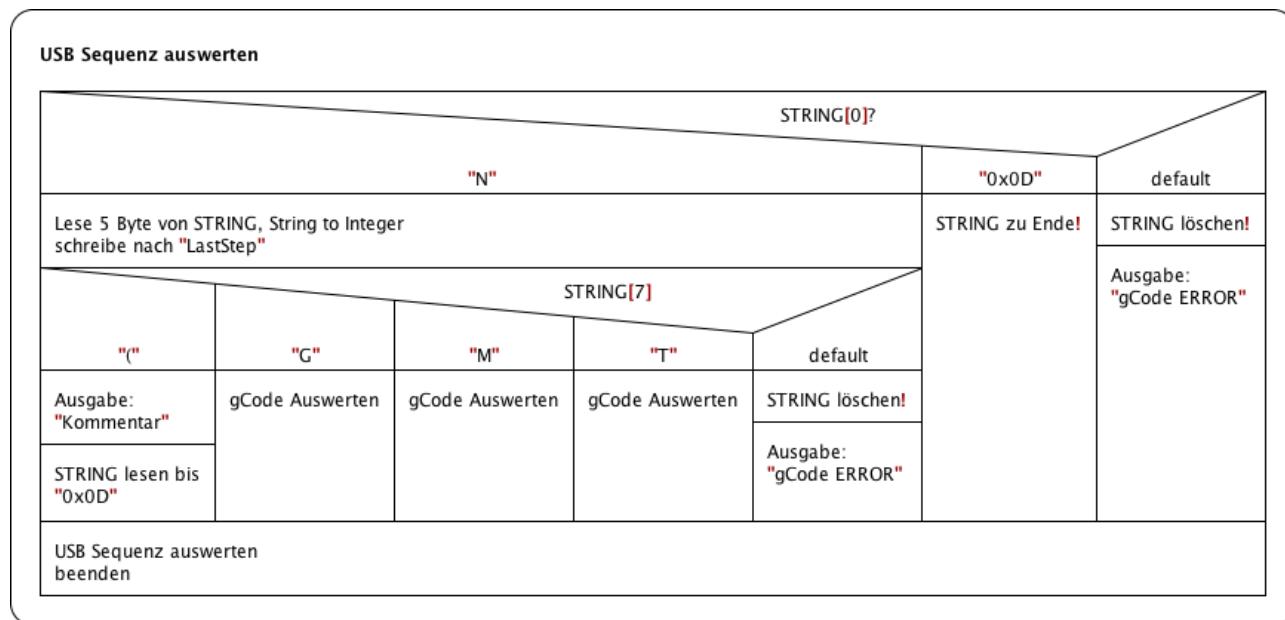


Abbildung 4.2.5: USB-Sequenz

Die Funktion `usb_string_reset(void)` verwirft diesen `STRING`.

Mit der Funktion `usb_get_state(void)` wird im Automatik-Modus der Status des USB-Sticks abgefragt.

#### 4.2.4 Serielle Schnittstelle uart.c

Diese Bibliothek wurde von Peter Fleurys UART-Library<sup>17</sup> aus der Version 1.6.2.2 übernommen und um eine bisher nicht existierende Funktion erweitert. Diese neue Funktion dient dem Einlesen von Strings, welche durch ein Trennzeichen, dem *separator*, untereinander abgegrenzt sind. Aus den separierten Strings wird der jeweils letzte String an den *BUFFER* übergeben.

```
*****
Function: uart_gets()
Purpose: receive string from UART
Input: pointer to chararray, separator
Returns: none
*****
void uart1_gets(char* BUFFER, char separator)
{
    uint8_t NextChar;
    uint8_t counter = 0;

    // BUFFER leeren!
    BUFFER[0] = '\0';

    // Warte auf und empfange das nächste Zeichen
    NextChar = uart1_getc();

    // Sammle solange Zeichen, bis:
    NextChar = (unsigned char)NextChar;

    // das String Ende Zeichen kam
    while (NextChar != separator && NextChar != NULL)
    {
        while (NextChar != separator && NextChar != NULL)
        {
            BUFFER[counter++] = NextChar;
            NextChar = uart1_getc();
            NextChar = (unsigned char)NextChar;
            while (NextChar == NULL)
            {
                NextChar = uart1_getc();
                NextChar = (unsigned char)NextChar;
            }
        }
        // CMD_CR wieder anhängen!
        BUFFER[counter++] = separator;

        // C-Standart-String-Terminierung
        BUFFER[counter++] = '\0';
    }
}
```

Abbildung 4.2.6: uart\_gets()

Diese Bibliothek steuert die Kommunikation zwischen dem Controller und dem VDrive2, sowie die Debugging- und Steuerkommunikation mit einem PC.

<sup>17</sup> (Fleury, Peter Fleury UART Library, 2006)

#### 4.2.5 Serieller Datenbus i2cmaster.c

Wie auch die UART-Library, stammt die hier verwendete I2C- Bibliothek ebenfalls von Peter Fleury. Sie wurde komplett in der Version 1.3 ohne Änderungen übernommen. Diese Bibliothek steuert die Kommunikation zwischen dem Controller und dem Display.

Es werden sechs wichtige Funktionen aus dieser Bibliothek benötigt:

`i2c_init()`, `i2c_start(ADRESSE)`, `i2c_rep_start(ADRESSE)`, `i2c_stop()`, `i2c_write(CHAR)`, sowie `i2c_readAck`.

Der Ablauf der Kommunikation mit dem Display wird in der nachfolgenden Grafik verdeutlicht.

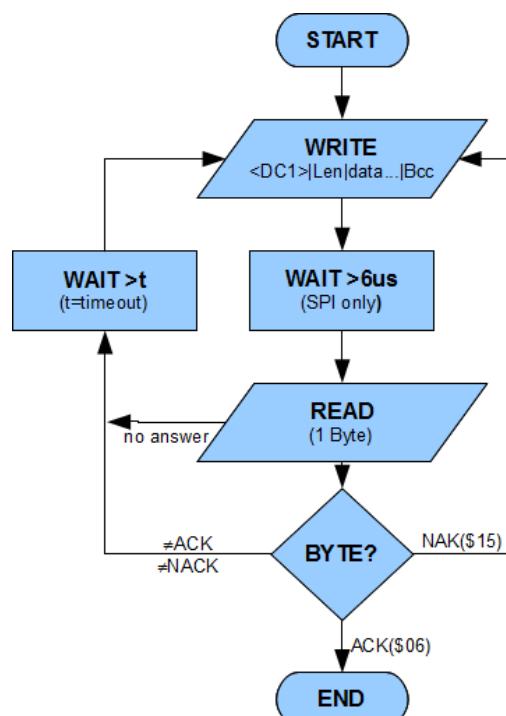


Abbildung 4.2.7: I2C-Kommunikation

Weitere Informationen zu dieser Library, sind der Dokumentation von Peter Fleury<sup>18</sup> zu entnehmen.

<sup>18</sup> (Fleury, Peter Fleury I2C Master library, 2006)

#### 4.2.6 Display-Handling edip240.c

In dieser Bibliothek

**BAUSTELLE**

#### 4.2.7 gCode-Struktur gcode.c

gCode ist eine, 1950 vom Massachusetts Institute of Technology entwickelte Programmiersprache zur CNC-Maschinensteuerung. Eine gCode-Datei ist meist an den cGode-typischen Dateiendungen zu erkennen. Diese sind beispielsweise .nc, .mpf, .tap. Der Aufbau einer gCode-Datei ist strukturiertes Klartext und durch die DIN66025-1<sup>19</sup> definiert. Ein Programm ist nach einer definierten Syntax aufgebaut. Die kleinste Informationseinheit ist ein **Wort**. Mehrere Worte, mindestens jedoch zwei, bilden einen **Satz**. Die Zusammensetzung der Sätze ergibt das **Programm**. Zu Beginn eines jeden Satzes steht die **Adresse**, bestehend aus einem Buchstaben, sowie einer Ziffernfolge, der **Nummer**. Jedes Programm beginnt mit der Adresse **N00000**. Diese wird satzweise in Zehner-Schritten erhöht. Die Adresse des 2. Satzes lautet somit **N00010**. Der so entstehende Abstand zwischen den Sätzen kann für manuelle Korrekturen oder Zwischensätze genutzt werden. Als Beispiel: **N00011**. Als Trennzeichen der einzelnen Worte dient ein Leerzeichen (Hex 0x20), die Trennung der Sätze erfolgt mit einem Zeilenvorschub (Hex 0xA). Ein Wort kann in 3 Datengruppen eingeteilt werden:

- Programmdaten
- Geometriedaten
- Technologiedaten

Adressen gehören zur Gruppe der Programmdaten. Das Ausführen von Geometrie- oder Technologiedaten wird durch eine Programmanweisung eingeleitet. Eine weitere Anweisung aus dem Bereich der Programmdaten ist das Programmende **M02**.

Dies bildet das letzte Wort eines Programms. Hieraus ergibt sich das minimalste gCode-Programm, bestehend aus einem Satz mit 2 Wörtern:

**N00000 M02**

Dieses Programm hat demnach keine Aufgabe und beendet sich sofort. Geometriefunktionen werden, mit einigen Ausnahmen, von einem oder mehreren weiteren Wörtern begleitet. Die Funktion **G00** (schneller Vorlauf) als Beispiel wird immer begleitet von Koordinaten. Zusätzlich können zu Geometriefunktionen noch weitere Worte, wie **Fxxxxx** angehängt

---

<sup>19</sup> (Deutsches Institut für Normung e.V., 1)

werden. Dieses Wort gibt die Feed-Rate, also die Bewegungsgeschwindigkeit des aktuellen Satzes an.

**N00120 G01 X12.250 Y-5.750 F120.0**

**G01** gibt an, dass es sich bei diesem Satz um eine geradlinige Bewegung handelt. Die begleitenden Zielkoordinaten sind X:12.250, Y:-5.750. Als Geschwindigkeit sind F:120.0 angegeben. Die Einheiten dieser Angaben werden zu Beginn des Programms über die Programmdaten **G20** für inch, bzw. **G21** für mm festgelegt.

Handelt es sich um absolute Koordinaten wird dies zu Beginn mit dem Wort **G90** definiert. Für relative Koordinaten mit **G91**. Beide richten sich nach dem kartesischen Koordinatensystem<sup>20</sup>.

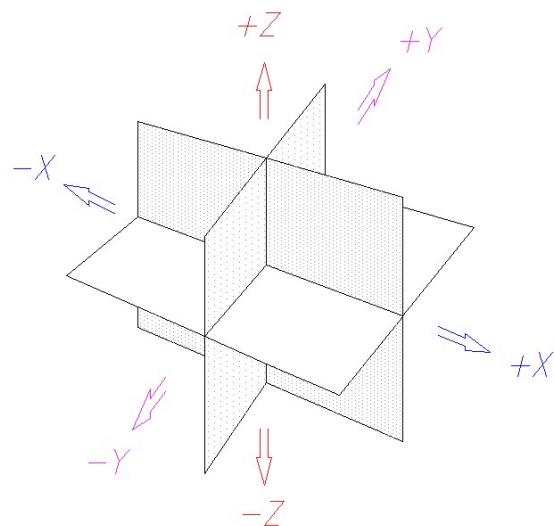


Abbildung 4.2.8: Kartesisches Koordinatensystem

---

<sup>20</sup> (Universität Karlsruhe, 1)

Kommentare werden mit ( eingeleitet und mit ) beendet. Ein Programm welches ein Loch bei den Koordinaten X:10, Y:10 mit der Tiefe 10mm bohrt, würde wie folgt aussehen:

```
N00000 G21
N00010 G90
N00020 G00 X10 Y10
N00030 G01 Z-10 F128.0
N00040 G01 Z10
N00045 G00 X0 Y0
N00050 M02
```

Zusätzlich zu den Programm- und Geometriedaten, gibt es noch die Gruppe der Technologiedaten. Diese beinhaltet zum Beispiel die Befehle zur Steuerung der Spindel. **M03** startet den Motor im Rechtslauf, **M04** im Linkslauf und **M05** stoppt den Motor. Das Wort **M06** kündigt einen Werkzeugwechsel an, welches von dem Wort **Txx** begleitet wird. Das **Txx** gibt die Nummer des nächsten zu verwendenden Werkzeugs an. Der Satz **M06 T02** wählt das Werkzeug 2 aus.

### 4.3 EA KitEditor-Programm

Programm-Dateien	Funktion	Autor
Technikerarbeit2012.kmc	Hauptprogramm	Dennis Hohmann
konstanten.kmi	Definieren von Konstanten	
fonts.kmi	Definieren der Schriftarten	
bilder.kmi	Definieren von Bildern	
errors.kmi	Definieren der Störmeldungen und Störungshandling	
function.kmi	Definieren der Maschinenfunktionen HAND, AUTO	

Abbildung 4.3.1: EA KitEditor Programmdateien

Der Umgang mit dem KitEditor, der Entwicklungssoftware des Displays, ist selbsterklärend. Die Online-Hilfe ist sehr ausführlich und enthält zahlreiche kommentierte Beispiele.

Auch das Demo-Programm ist ausreichend erklärt. Der komplette zur Verfügung stehende Befehlssatz ist sowohl im Datenblatt des Displays, als auch in den Beispiel-Programmen vorhanden.

Der Programmaufbau ist ähnlich der C-Programm-Struktur. Es gibt eine main-ähnliche Hauptdatei. Diese ist durch die Endung .kmc zu erkennen. Dort befindet sich ein Include-Bereich, welcher alle benötigten Dateien deklariert. Dies sind im aktuellen Projekt fünf Dateien, welche an der Endung .kmi erkennbar sind. Diese Struktur ist nicht zwingend notwendig, erhöht jedoch die Übersichtlichkeit um ein Vielfaches.

Im Gegensatz zu einem C-Programm ist das Display-Programm nicht über eine Schleife gesteuert. Die Ablaufsteuerung läuft über programmierte Makros. Ein Makro ist eine Zusammenfassung einzelner oder mehrere Befehlen. Es gibt grundsätzlich zwei Wege, im Display Befehle zur Ausführung zubringen. Das Senden des Befehls, direkt über die Schnittstelle oder durch das Auslösen eines Makros welches wiederum Befehle ausführen kann. Ein Makro kann wiederum auf zwei unterschiedlichen Wegen ausgelöst werden. Zeitgesteuert oder Ereignisgesteuert. Die zeitgesteuerten Makros werden für eine bestimmte Zeit oder nach Ablauf einer bestimmten Zeit ausgeführt. Ereignisgesteuerte Makros können über eine Zustandsänderung an einem I/O-Pin des Displays oder über die Touch-Folie ausgelöst oder beendet werden. Die letzteren werden auch als Touchmakros bezeichnet. Makros können beliebig verschachtelt werden, der Nachteil hierbei ist jedoch der Verlust der Übersichtlichkeit.

Beim Starten des Displays wird automatisch das PowerOn-Makro geladen. Dies geschieht ebenso nach einem Reset, in Verbindung mit einem Reset-Makro.

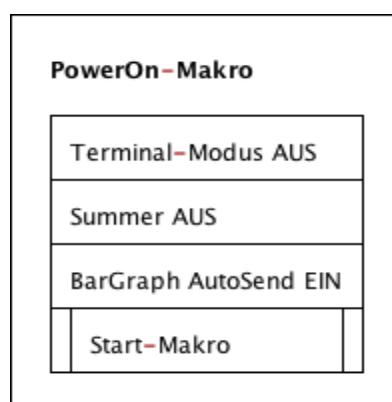


Abbildung 4.3.2: Stuktogramm PowerOn-Makro

In diesem Schritt werden der Betriebsmodus, sowie das Verhalten bei Wertänderung eines Schiebereglers (BarGraph) festgelegt. Das Start-Makro legt das weitere Verhalten des Displays sowie die Visualisierung fest.

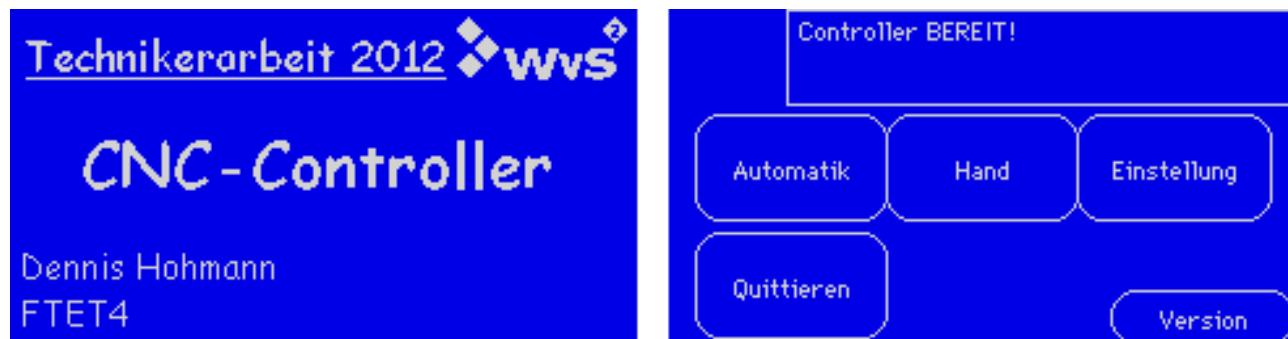


Abbildung 4.3.3: Start des Controllers

Ist der Controller ohne Fehler angelaufen, wird nach ca. fünf Sekunden das Hauptmenu aufgerufen. Im oberen Drittel des Displays befindet sich das Statusfenster. Hier werden alle Meldungen wie z.B. „Controller BEREIT“ angezeigt. Sollte es während des Anlaufs zu einem Problem mit z.B. dem VDrive2 kommen, wird dies mit einer Klartext-Meldung „USB nicht gefunden“ im Statusfenster angezeigt. Das Statusfenster ist in allen Betriebsmodi verfügbar, wie auch der Button „Quittieren“. Es gibt zwei Klassen von Meldungen. Einfache Meldungen und Warnungen. Letztere sind mit einem blinkenden Symbol (→ Abb 4.3.4) gekennzeichnet und erfordern einen Eingriff durch den Benutzer. Diese Meldungen müssen quittiert werden.



Abbildung 4.3.4: Warnung

Das Hauptmenu ist in 3 Gruppen aufgeteilt. Automatik, Hand und Einstellungen. Es sind Menu-Punkte vorbereitet, welche jedoch noch nicht vollständig verfügbar sind. Diese sind mit einem ! gekennzeichnet. Diese Funktionen werden zu einem späteren Zeitpunkt vollständig implementiert, da sie für die Funktion der CNC-Steuerung nicht zwingend notwendig waren. Sie bieten lediglich einen zusätzlichen Bedienkomfort.

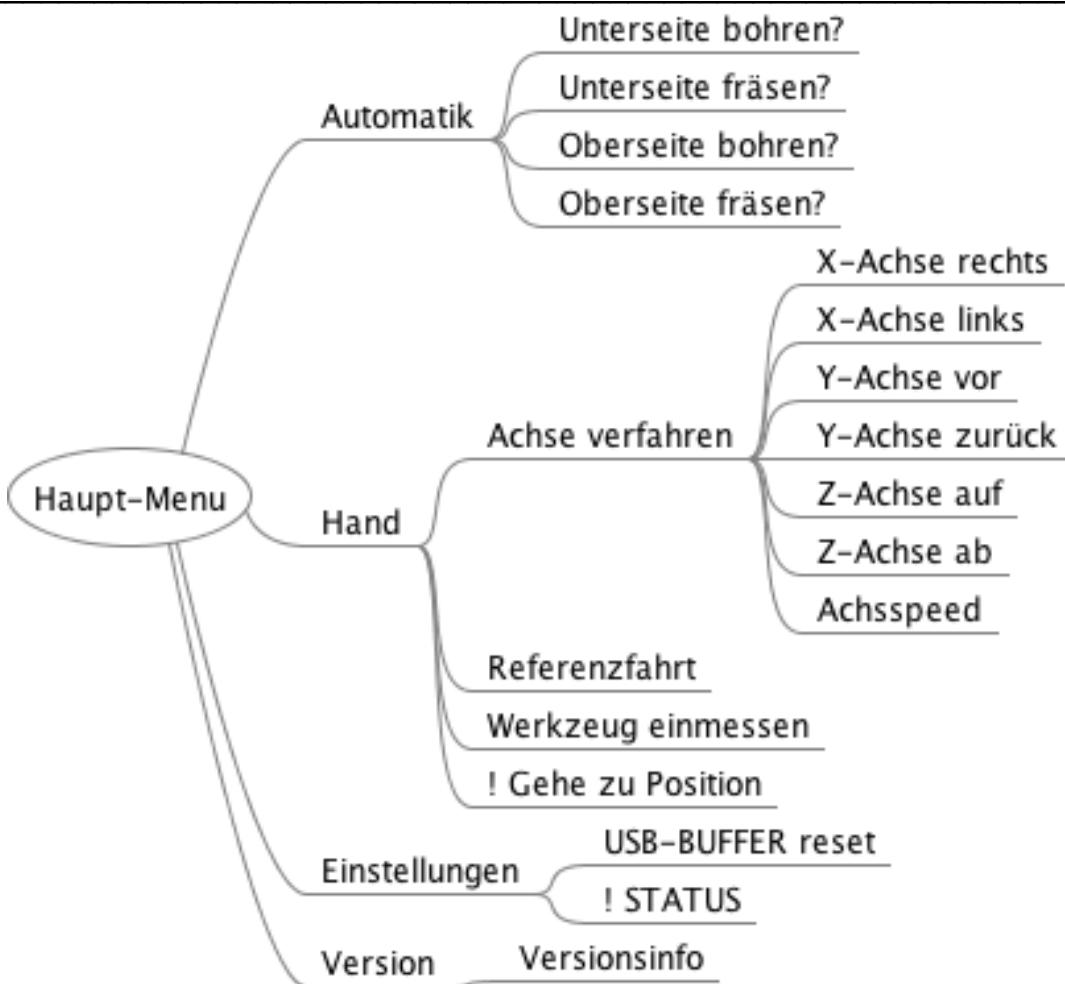


Abbildung 4.3.5: Menü-Struktur

Die Gruppe der Automatik-Funktionen enthält die aktuell vier möglichen Funktionen zur Bearbeitung einer Platine. Diese Funktionen greifen direkt auf den USB-Stick zu und lesen die Daten der entsprechenden Datei ein. Nur in den Gruppen der Automatikfunktionen stehen nach dem Start der Datei weitere Funktionen wie z.B. Pause oder Abbrechen zur Verfügung. Sie dienen der Unterbrechung des aktuellen Programms oder gar dem Abbruch. Die genaue Funktion der hier aufgeführten Befehle sind dem Kapitel 4.2 Controller-Programm zu entnehmen.

Der Austausch von Informationen zwischen dem Controller und dem Display findet über Makro-Aufrufe statt. Umgekehrt sendet das Display bei den entsprechenden Events eine Zeichenkette mit entsprechendem Inhalt. Diese sind ebenfalls dem Kapitel 4.2 Controller-Programm zu entnehmen.

#### 4.4 SmallProtocoll

Das SmallProtokoll ist ein Protokollrahmen, unabhängig von der verwendeten Schnittstelle. Es wird in zwei Handshakegruppen unterschieden, DC1 und DC2, gefolgt von der Anzahl der Nutzbytes und der Checksumme.



Abbildung 4.4.1: Protokollrahmen<sup>21</sup>

Die Checksumme wird durch die Summierung des Handshake-Typs, der Anzahl der Nutzdaten, sowie allen Nutzdatenbytes errechnet. Modulo-Operation der Summe / 256 ergibt die Checksumme. Ist die Checksumme korrekt, sendet das Display ein ACK. Diese Antwort bestätigt jedoch nur ob das Datenpaket korrekt empfangen wurde. Eine Syntaxprüfung findet hier nicht statt. Sollte während der Übertragung ein Byte beschädigt sein oder gar verloren gegangen sein, gibt es zwei mögliche Reaktionen des Displays. Die erste Möglichkeit ist die Bestätigung des fehlerhaften Pakets mit NAK. Die zweite Möglichkeit ist das Ablaufen der voreingestellten Timeoutzeit und das damit verbundene, Verwerfen des aktuellen Pakets ohne eine Antwort.

Um die voreingestellten Protokolldaten, wie die maximale Timeoutzeit oder die maximale Anzahl an Nutzdatenbytes, zu ändern, wird ein DC2 Protokoll mit folgendem Aufbau gesendet:



Abbildung 4.4.2: Protokollrahmen DC2<sup>22</sup>

<sup>21</sup> (www.goBlack.de, 1)

<sup>22</sup> (www.goBlack.de, 1)

---

Diese Einstellungen sind direkt nach der Verarbeitung des Datenpakets gültig. Ein Neustart des Displays ist nicht nötig.

## 5 Die Steuerplatine

### 5.1 Aufbau

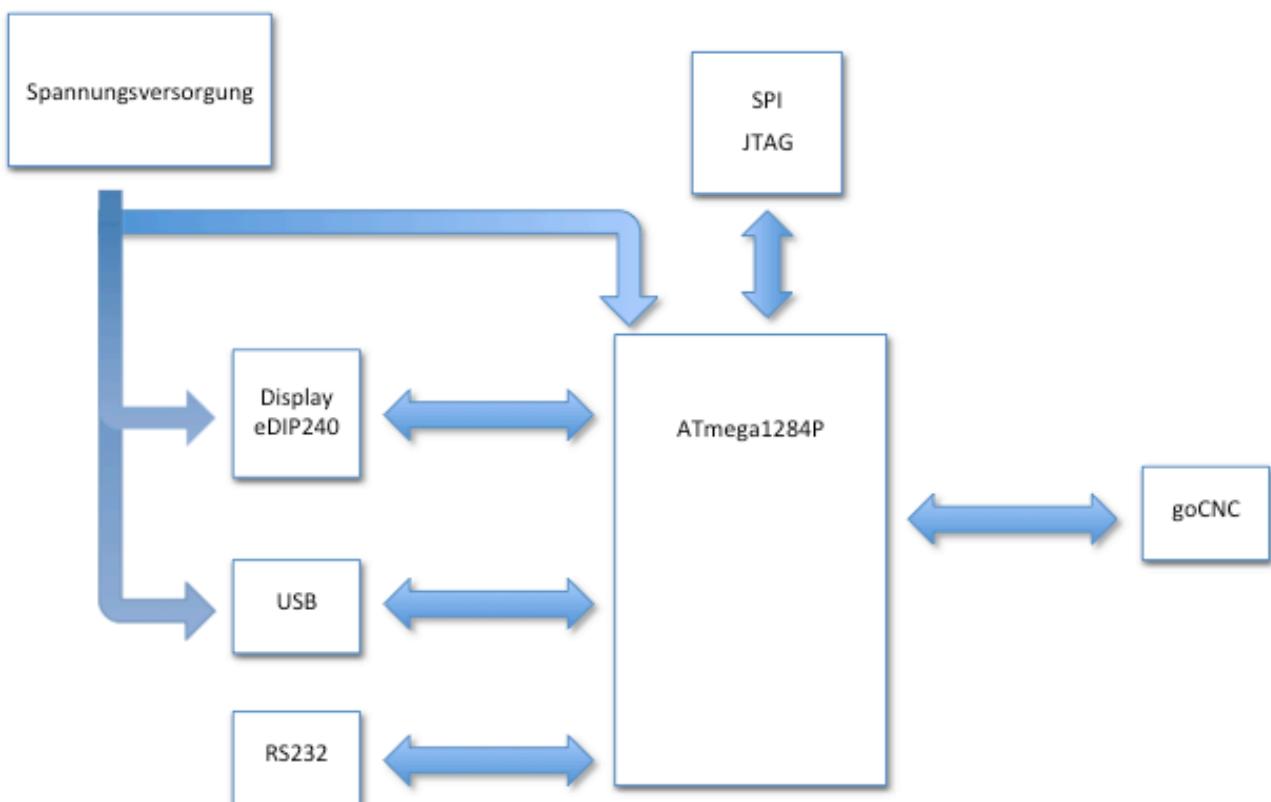


Abbildung 5.1.1: Blockschaltbild Steuerplatine

Die Spannungsversorgung ist auf einen Verbrauch von 445mA bei 5V DC ausgelegt. Dieser setzt sich wie folgt, aus den gemessenen Verbrauchswerten, zusammen:

ATmega1284P-PU	35 mA
VDrive2-USB	200 mA
eDIP240-Display	210 mA

Abbildung 5.1.2: Stromverbrauch

Der verwendete Brückengleichrichter ist auf 35 V und 4 A ausgelegt. Der Festspannungsregler 7805 ist mit 1 A belastbar. Die Leiterbahnen der Versorgungsspannung sind in 0,60 mm ausgeführt. Die Berechnung der Leiterbahnbreite wurde mit der PCB-Toolbox iPhone App<sup>23</sup> unter Berücksichtigung einer doppelseitigen Platine mit einer Kupferstärke von 35 µm und einer maximalen Erwärmung von 4 °C bei 1 A durchgeführt.

<sup>23</sup> (LeitOn, 1)

Die ICs ATmega1284P-PU sowie der MAX232 sind gesockelt eingelötet, alle weiteren Bauteile sind direkt eingelötet.

## 5.2 Das Design

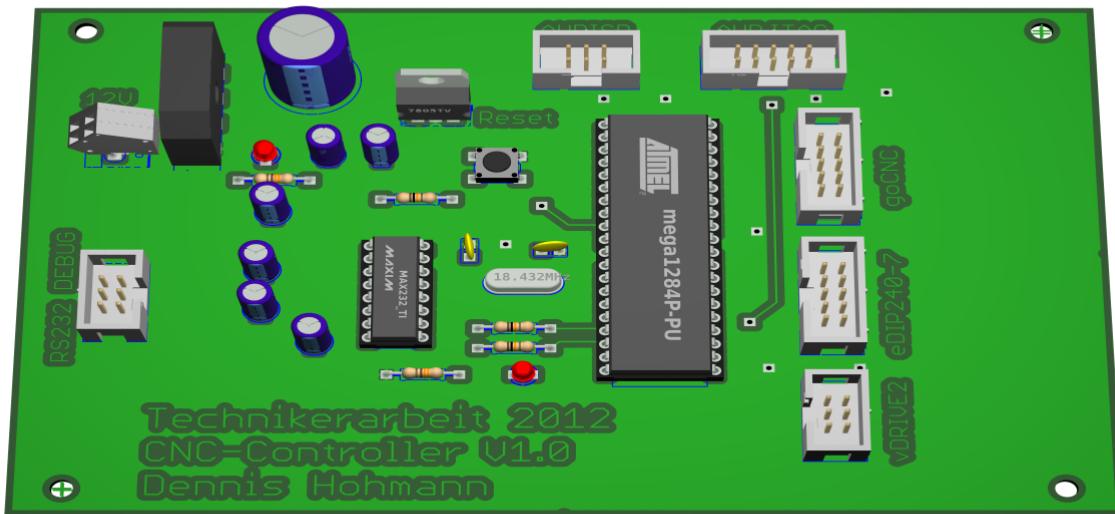


Abbildung 5.2.1: Steuerplatine Designentwurf

Die Steuerplatine wurde ebenfalls mit EAGLE designet. Zur Herstellung wurde die Belichtungsmethode verwendet. Die Bohrungen der Bauteile und VIAs wurden jedoch von der hier entwickelten Steuerung und Portalfräse ausgeführt.

Beim entwerfen der Platine wurde als äußerer Maß Euro-Platinen-Format (L x B x H) 160 mm x 100 mm x 1,5 mm gewählt. Das Einbau-Display mit den Maßen 128mm x 94 mm x 32 mm, sowie der VDrive2 mit 41 mm x 20 mm x 42 mm, hat die groben Maße für das Gehäuses vorgegeben. Dieses hat eine Abmessung von 200 mm x 120 mm x 77 mm.

Aufgrund der geringen Anzahl an Bauelementen wurde auf die Verwendung von SMD-Bauteilen verzichtet. Das Design wurde zur besseren Übersicht und Lötbarkeit doppelseitig ausgeführt. Somit entfallen die Lötbrücken und werden durch VIAs ersetzt.

## 6 Zusammenfassung

### 6.1 Fazit

Das Projekt CNC-Steuerung hat sich, durch das Zusammenspiel verschiedenster Komponenten, während der Entwicklung als sehr umfangreich und komplex dargestellt. Insbesondere die Programmierung des Displays hat sich als besonders anspruchsvoll erwiesen, da es eine eigene Entwicklungsumgebung, sowie eine eigene Programmiersprache verwendet. Dennoch ist das gesamte Projekt immer im Zeitplan (→ Abb. 2.3.1) verlaufen. Alle sieben Teilabschnitte sind erfolgreich umgesetzt. Leider ist es auf Grund beschriebenen mechanischen Ungenauigkeiten nicht gelungen, die Steuerplatine komplett mit der hier eingesetzten Portalfräse herzustellen. Das Bohren der Bauteil-Löcher, sowie der VIAs, wurde mit der maximal möglichen Präzision der CNC-Steuerung und der verwendeter Portalfräse von dieser übernommen. Die Bedienung der Steuerung, mit dem übersichtlich und strukturiert gestalteten Display bietet dem Benutzer alle benötigten Informationen. Auch die, in der Aufgabenstellung geforderte, Mobilität und Flexibilität ist mit der verwendeten USB-Lösung gegeben. Des Weiteren hat mir persönlich, dieses Projekt mein Wissen und meine Fähigkeiten im Umgang mit der Mikrocontroller-Programmierung vertieft. Aus dieser Erfahrung und dem Gelernten werde ich auch in Zukunft bei weiteren Projekten profitieren können.

### 6.2 Ausblick

Die CNC-Steuerung entspricht zwar in vollem Umfang der Aufgabenstellung, dennoch wird es im Anschluss an diese Projekt weiter Modifikationen und Optimierungen an Hard- und Software geben. Zum einen ist die Kompensation des Lagerumkehrspiels ein großes Thema, zum anderen die Positionierung auf eine direkt als Koordinaten eingegeben Position. Dieses Technikerarbeit wird auch in Zukunft weiter entwickelt werden.

## 7 Quellenverzeichnis

- Atmel. (1). *Atmel*. Abgerufen am 01. 11 2011 von <http://www.atmel.com/>
- Atmel. (3). *Atmel AVR Studio 5.1*. Abgerufen am 12. 03 2012 von  
<http://www.atmel.com/tools/ATMELAVRSTUDIO.aspx>
- Atmel. (2). *JTAGICE3 - Atmel*. Abgerufen am 02. 04 2012 von  
<http://www.atmel.com/tools/JTAGICE3.aspx>
- Barrett, S. F. (2009). *Embedded System Design with the Atmel Avr Microcontroller*. California, USA: Morgan & Claypool.
- CadSoft. (1). *CadSoft*. Abgerufen am 03. 04 2012 von <http://www.cadsoft.de/eagle-pcb-design-software/?language=de>
- CadSoft. (2). *CadSoft EAGLE*. Abgerufen am 12. 01 2012 von  
<http://www.cadsoft.de/eagle-pcb-design-software/product-overview/?language=de>
- Crawford & Prinz, T. &. (2006). *C in a Nutshell* (Bd. 1). (P. Printz, Übers.) Deutschland: O'Reilly.
- Deutsches Institut für Normung e.V. (1). *D/N66025-1*. Abgerufen am 04. 04 2012 von  
<http://www.nwm.din.de/cmd?artid=1012276&contextid=nwm&bcrumblevel=1&subcommiteeid=79267823&level=tpl-art-detailansicht&committeeid=54739075&languageid=de>
- Electronic Assembly. (1). *LCD-Module*. Abgerufen am 17. 11 2011 von <http://www.lcd-module.de/>
- Fleury, P. (2006). *Peter Fleury I2C Master library*. Abgerufen am 06. 02 2012 von  
[http://homepage.hispeed.ch/peterfleury/group\\_pfleury\\_ic2master.html](http://homepage.hispeed.ch/peterfleury/group_pfleury_ic2master.html)
- Fleury, P. (2006). *Peter Fleury UART Library*. Abgerufen am 29. 11 2011 von  
[http://homepage.hispeed.ch/peterfleury/group\\_pfleury\\_uart.html](http://homepage.hispeed.ch/peterfleury/group_pfleury_uart.html)
- FTDI Chip. (1). *FTDI Chip Home Page*. Abgerufen am 15. 02 2012 von  
<http://www.ftdichip.com/>
- goCNC. (1). *CNC Technik für den Hobby- Bereich*. Abgerufen am 16. 10 2011 von  
<http://gocnc.de/>
- Johnson, J. (29. 11 2009). *Phorum PCB gCode*. (J. Johnson, Produzent) Abgerufen am 16. 12 2011 von <http://pcbgcode.org/read.php?12,361>
- LeitOn. (1). *Leiterplattenentwicklung*. Abgerufen am 02. 12 2011 von  
<http://www.leiton.de/leiterplatten-toolbox-iphone.html>
- MAXIM. (07. 08 2003). *Determining Clock Accuracy Requirements for UART Communications*. Abgerufen am 20. 10 2011 von <http://www.maxim-ic.com/app-notes/index.mvp/id/2141>
- Pollin. (30. 11 2007). Atmel Evaluation Board Version 2.0.1.
- The SD Association. (1). *The SD Association SD Standards*. Abgerufen am 02. 04 2012 von <https://www.sdcard.org/home/>
- Universität Karlsruhe. (1). *CNC-Programmierung*. (T. Schönwald, Produzent) Abgerufen am 12. 03 2012 von <http://www.lehrer.uni-karlsruhe.de/~za685/cnc/programmierung.htm>
- Wikipedia. (2012). *EAGLE*. Abgerufen am 15. 12 2011 von  
<http://de.wikipedia.org/wiki/Eagle>

---

www.goBlack.de. (1). *Grafik-TouchDisplay EA eDIP240 Datenblätter Protokollrahmen* .  
Abgerufen am 12. 04 2012 von  
<http://www.goblack.de/desy/digitalt/aktoren/touchdisp-edip240/ea240protokoll.html>

www.mikrocontroller.net. (2007). *AVR-GCC-Tutorial/Der UART*. Abgerufen am 20. 10  
2011 von [http://www.mikrocontroller.net/articles/AVR-GCC-Tutorial/Der\\_UART](http://www.mikrocontroller.net/articles/AVR-GCC-Tutorial/Der_UART)

## 8 Softwareverzeichnis

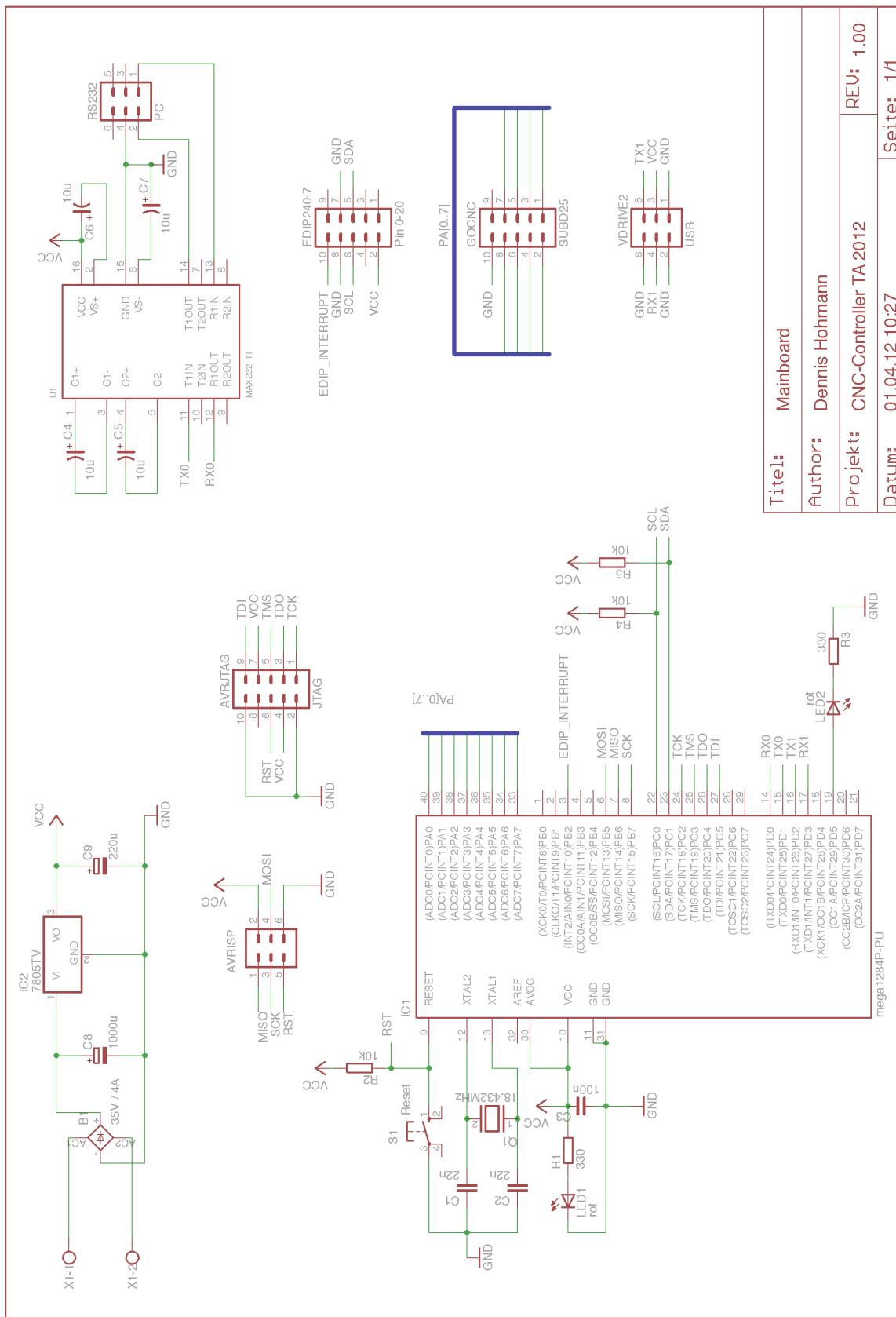
Firma	Software	Version	Quelle
Apple	Mac OS X Lion	10.7.3	kostenpflichtig Unix-Betriebssystem
VMware	VMware Fusion	4.1.0 Mac OS X	kostenpflichtig
Microsoft	Windows XP	WinXP SP3	kostenpflichtig Betriebssystem
Atmel	AVR Studio 5	5.1.1163 WinXP	Freeware Dokumentations-CD
Atmel	Programmer AVRISP mkII	1.e	Hardwareprogrammer
Atmel	JTAG AVR JTAGICE3	1.24	Hardwareprogrammer
CADSoft	EAGLE	5.7.0 Mac OS X	kostenpflichtig
John Johnson Software	PCB-GCODE	3.5.2.11	Freeware Dokumentations-CD
ELECTRONIC ASSEMBLY	LCD-Tools	4.3 WinXP	Freeware Dokumentations-CD
Vinculum	VncFWMod	1.1b WinXP	Freeware Dokumentations-CD
Vinculum	VDrive2 Firmware	3.68	Freeware Dokumentations-CD

Abbildung 6.2.1: Software und Versionen

## **9 Anhang**

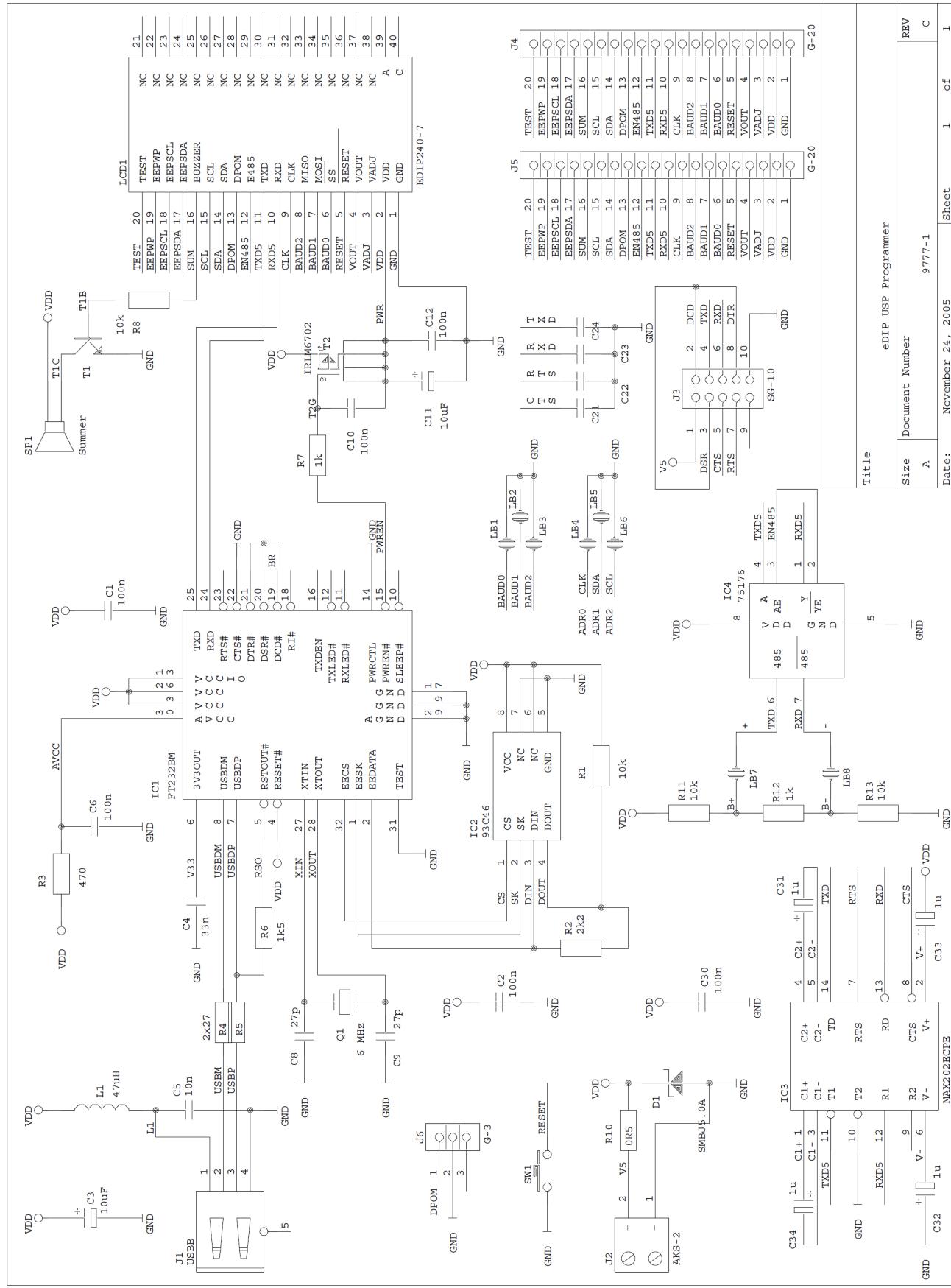
## 9.1 Schaltpläne

### 9.1.1 Mainboard



**Abbildung 9.1.1: Schaltplan der Steuerplatine**

## 9.1.2 Dip240-7



**Abbildung 9.1.2: Schaltplan des eDIP240 mit USB Board**

### 9.1.3 VDrive2

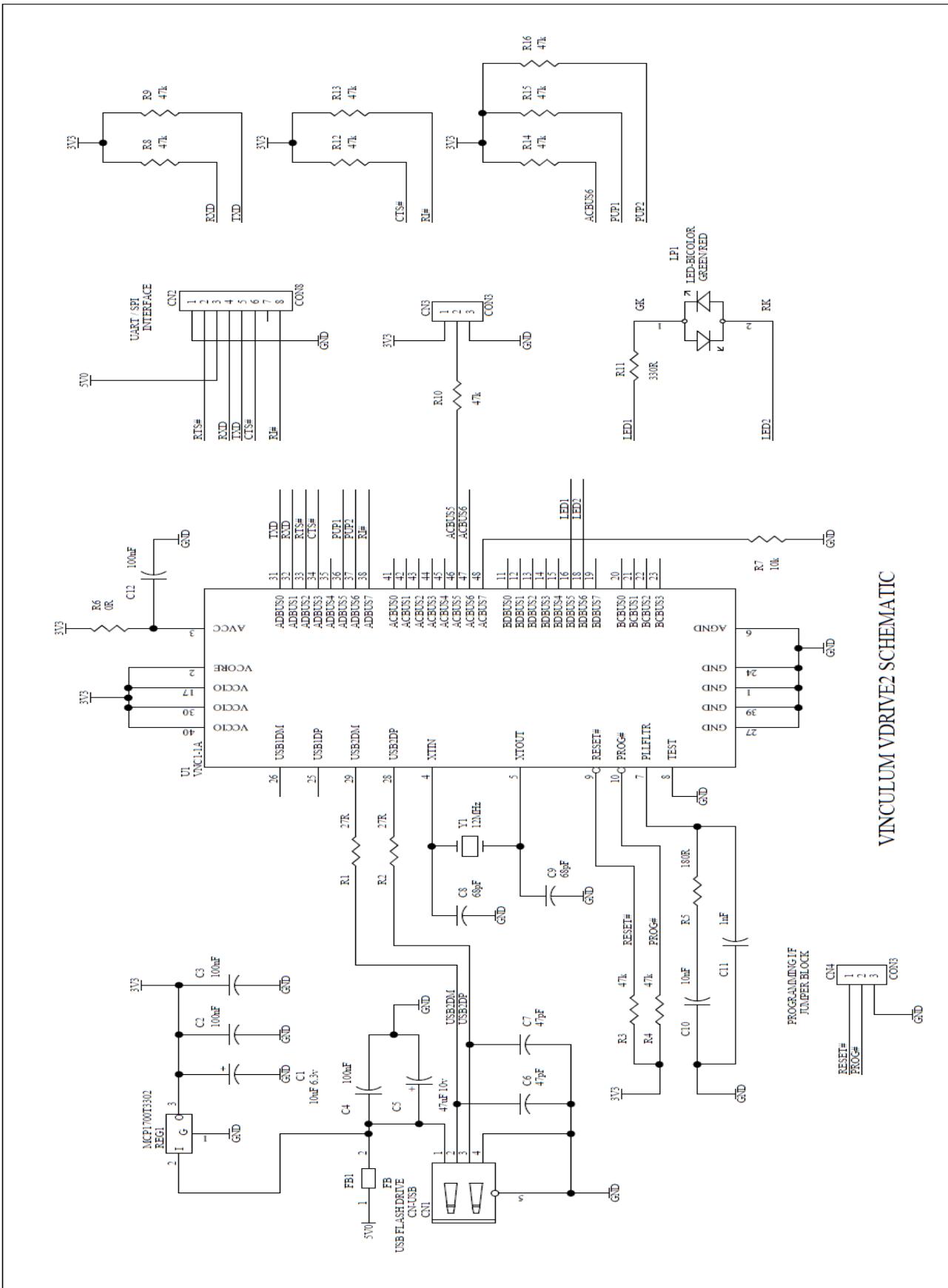


Abbildung 9.1.3: Schaltplan VDrive2

#### 9.1.4 Board-Layout

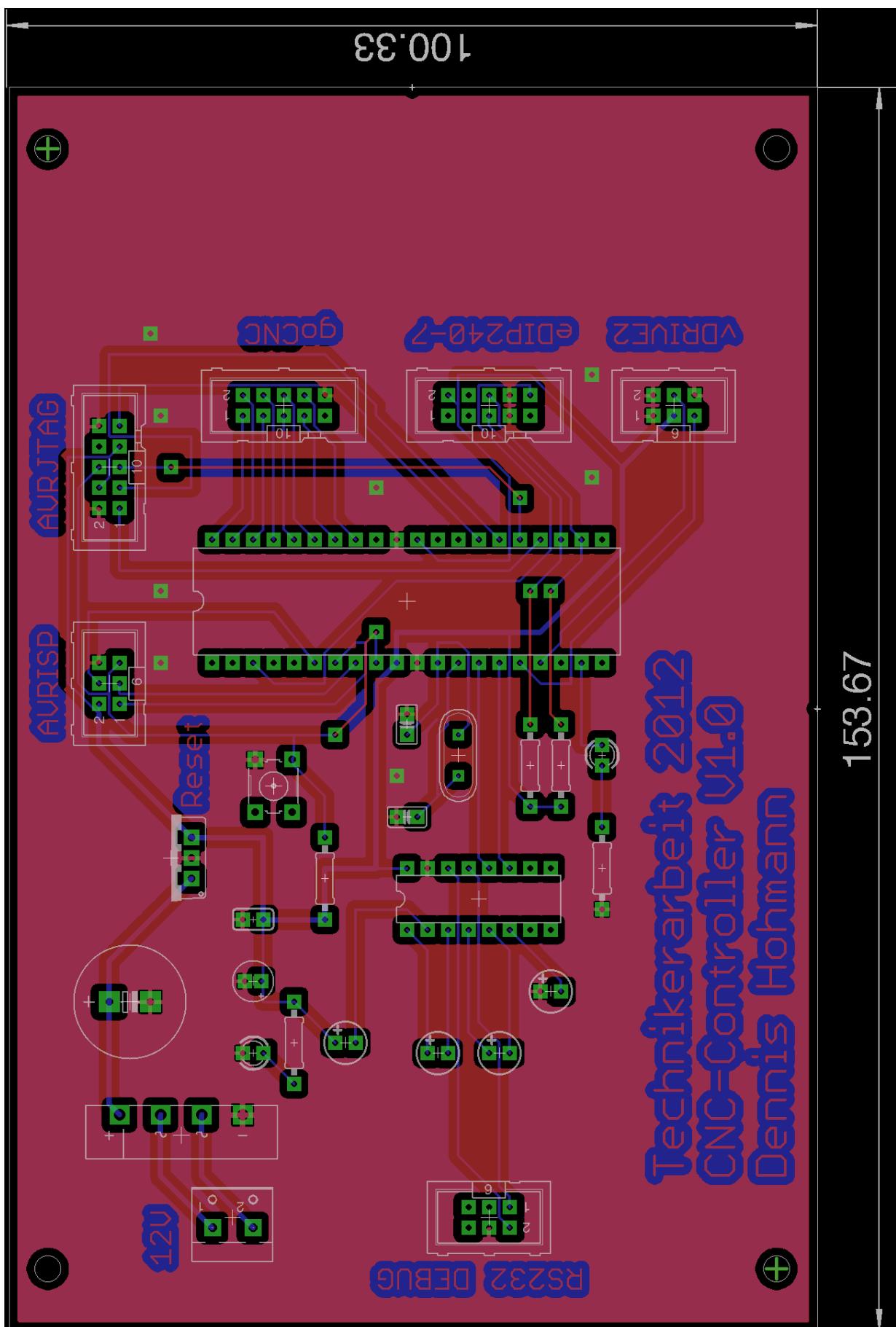


Abbildung 9.2.4: Layoutplan

### 9.1.5 Gehäuse

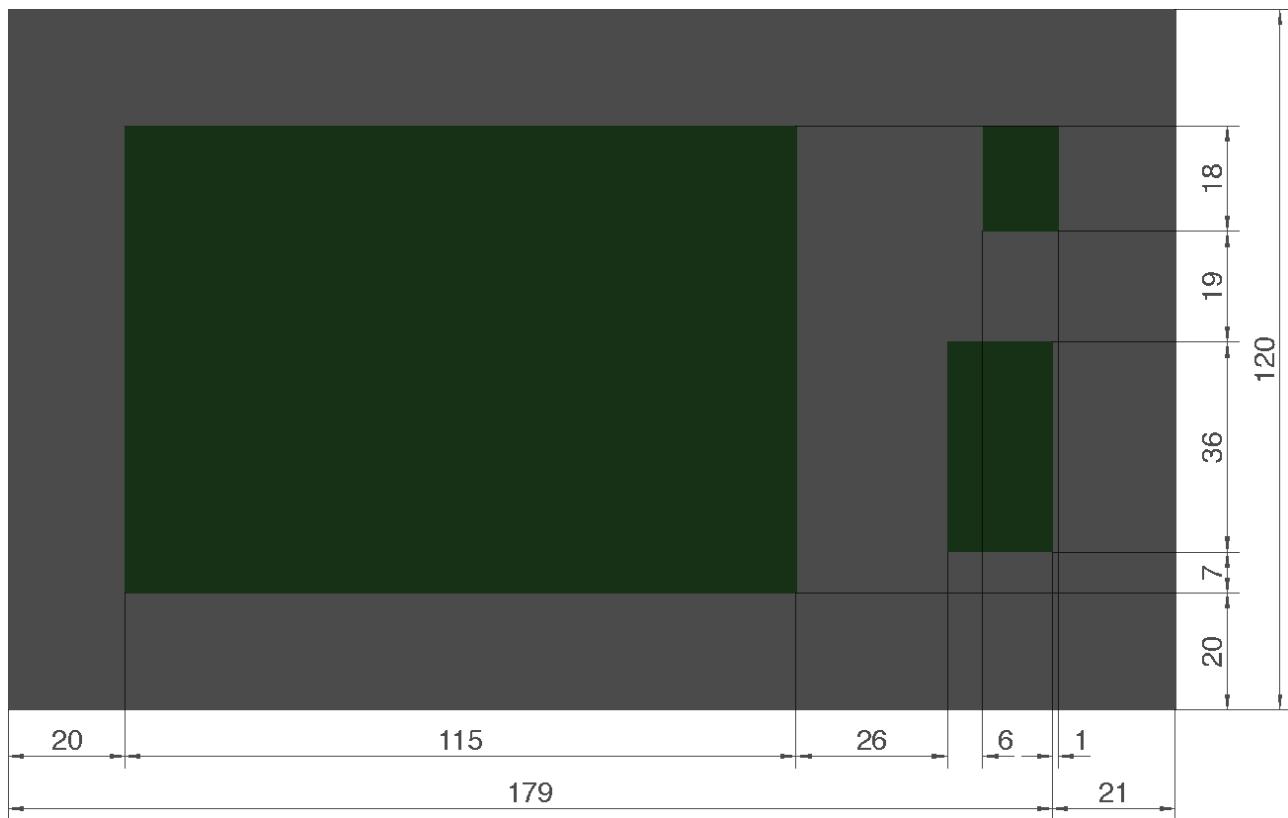
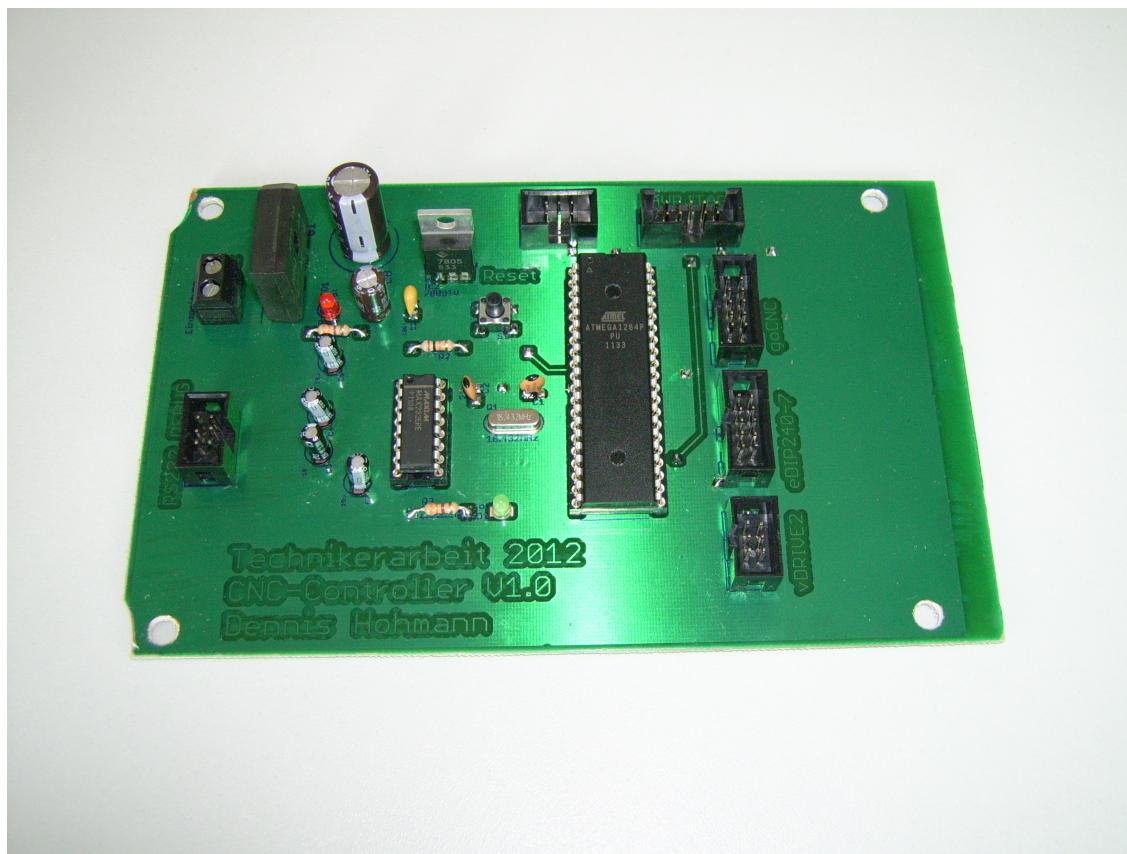


Abbildung 9.1.4: Gehäuse-Oberseite

Weitere Bilder zum Gehäuse befinden sich auf der Beigefügten Dokumentations-CD im Verzeichnis Bilder.

## 9.1.6 Fertige Platine



**Abbildung 9.1.5: Fertige Platine**

Weitere Bilder zum Platine befinden sich auf der Beigefügten Dokumentations-CD im Verzeichnis Bilder.

## 9.2 Datenblätter

### 9.2.1 EA eDIP240B-7LWTP









































### **9.2.2 Vinculum VDrive2**















### **9.2.3 Gehäuse**

#### **9.2.4 Gleichrichter**





### **9.3 Dokumentations-CD**