

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**, 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

1 Vorwort	5
1.1 Einleitung	5
1.2 Copyright	5
1.3 Danksagung	6
1.4 Abbildungsverzeichniss	7
1.5 Abkürzungsverzeichnis	9
2 Projektbeschreibung	10
2.1 Motivation	10
2.2 Aufgabenstellung	11
2.3 Zeitplan	12
2.4 Planung	13
3 Hardware	14
3.1 Der Controller	14
3.2 Das Speichermedium	15
3.3 Das Display	16
3.4 Die Portalfräse	17
3.5 UNI1500	18
3.6 Genauigkeit	19
3.6.1 X- & Y- Achsen	20
3.6.2 Z-Achse	22
3.7 Geschwindigkeit	22
3.8 Spindeldrehzahl	23
3.9 Hardware- & Softwarekonfiguration	24
3.9.1 Atmel ATmega1284P-PU Fuses & Lockbits	24
3.9.2 Vinculum VDrive2 Firmware 3.68	26
3.9.3 EAGLE	26
3.9.4 PCB-GCODE-ULP 3.5.2.11	27
4 Software	29
4.1 Einleitung	29
4.2 Controller-Programm	30
4.2.1 Globale Definitionsdatei globdef.h	32
4.2.2 Schrittmotor-Handling gocnc.c	33
4.2.3 USB-Kommunikation vnc1l.c	34
4.2.4 Serielle Schnittstelle uart.c	35
4.2.5 Serieller Datenbus i2cmaster.c	36
4.2.6 Display-Handling edip240.c	37
4.3 gCode-Struktur	38
4.4 EA KitEditor-Programm	40
4.5 SmallProtocoll	44
5 Die Steuerplatine	46
5.1 Aufbau	46

5.2 Das Designe	47
6 Zusammenfassung.....	48
6.1 Fazit	48
6.2 Ausblick	48
7 Literatur & Quellverzeichnis.....	49
8 Softwareverzeichnis.....	51
9 Anhang	52
9.1 Schaltpläne	53
9.1.1 Mainboard.....	53
9.1.2 Dip240-7	54
9.1.3 VDrive2	55
9.1.4 Board-Layout	56
9.1.5 Gehäuse	57
9.1.6 Fertige Platine.....	58
9.2 Datenblätter	59
9.2.1 EA eDIP240B-7LWTP	59
9.2.2 Vinculum VDrive2	80
9.2.3 Gehäuse	88
9.2.4 Gleichrichter.....	89
9.3 Dokumentations-CD	92

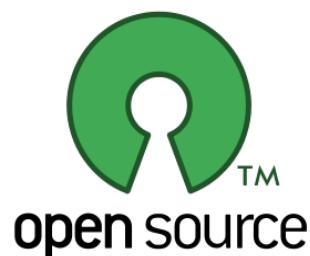
1 Vorwort

1.1 Einleitung

Die Weiterbildung zum staatlich geprüften Techniker Elektrotechnik beinhaltet begleitend zum 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

Es wird gestattet, ohne meine Zustimmung, die Dokumentation „CNC-Steuerung“ zu veröffentlichen und die in dieser Dokumentation enthaltenen Pläne und Layouts für einen Nachbau zu verwenden, oder diese unter Angabe der Quelle weiter zu entwickeln.



1.3 Danksagung

An dieser Stelle möchte ich mich bei all denjenigen bedanken, die mich bei der Realisierung dieses Projekts in irgendeiner Art und Weise unterstützt haben. Besonders Dank möchte ich folgende Personen und Firme richten:

Dr. Bernhard Spitzer

Für die fachliche Betreuung, als Ansprechpartner und Unterstützer.

ELECTRONIC ASSEMBLY¹

Für die kostenfreie Überlassung eines eDIP240B-7LWTP Touchdisplay

Atmel²

Für die kostenfreie Überlassung der Controller als Samples,
sowie ein sehr günstiges Angebot für einen AVR JTAGICE3³ Debugger

CadSoft⁴

Für die kostenfreie Überlassung einer Version EAGLE 5.7 Pro OSX⁵

¹ (Electronic Assembly, 1)

² (Atmel, 1)

³ (Atmel, 2)

⁴ (CadSoft, 1)

⁵ (CadSoft, 2)

1.4 Abbildungsverzeichniss

Abbildung 2.3.1: Projektzeitplan.....	12
Abbildung 2.4.1: Atmel ATmega1284P-PU	14
Abbildung 3.2.1: FTDI VDrive2 USB	15
Abbildung 3.3.1: Electronic Assembly eDIP240-7.....	16
Abbildung 3.4.1: CNC-Bausatz www.gocnc.de "HOBBY A4"	17
Abbildung 3.5.1: USOVO UNI1500 Steuerplatine	18
Abbildung 3.5.2: USOVO UNI1500 Pinbelegung	18
Abbildung 3.6.1: Parallelität der Achsen	20
Abbildung 3.6.2: Leerschritte	20
Abbildung 3.6.3: Umkehrspiel der Y-Achse	21
Abbildung 3.7.1: PWM Schrittmotor	22
Abbildung 3.9.1: Logo EAGLE	26
Abbildung 3.9.2: PCB-GCODE Menu	27
Abbildung 4.2.1: Controller Programmdateien	30
Abbildung 4.2.2: Struktogramm Init.....	30
Abbildung 4.2.3: Bitfeld mit 2 unsigned CHAR Variablen	32
Abbildung 4.2.4: USB-Sequenz	34
Abbildung 4.2.5: uart_gets().....	35
Abbildung 4.2.6: I2C-Kommunikation.....	36
Abbildung 4.3.1: Kartesisches Koordinatensystem.....	39
Abbildung 4.4.1: EA KitEditor Programmdateien	40
Abbildung 4.4.2: Struktogramm PowerOn-Makro.....	41
Abbildung 4.4.3: Start des Controllers	42
Abbildung 4.4.4: Warnung.....	42
Abbildung 4.4.5: Menu-Struktur	43
Abbildung 4.5.1: Protokollrahmen	44
Abbildung 4.5.2: Protokollrahmen DC2	44
Abbildung 5.1.1: Blockschaltbild Steuerplatine	46
Abbildung 5.1.2: Stromverbrauch.....	46
Abbildung 5.2.1: Steuerplatine Designentwurf	47
Abbildung 6.2.1: Software & Versionen	51
Abbildung 9.1.1: Schaltplan der Steuerplatine	53
Abbildung 9.1.2: Schaltplan des eDIP240 mit USB Board.....	54

Abbildung 9.1.3: Schaltplan VDrive2.....	55
Abbildung 9.1.4: Gehäuse-Oberseite	57
Abbildung 9.1.5: Fertige Platine	58

1.5 Abkürzungsverzeichnis

AC	Alternating Current	Wechselstrom
CAD	Computer Aided Design	Computergestützte Konstruktion
CNC	Computer Numerical Control	Computergestützte Steuerung
DC	Direct Current	Gleichstrom
EEPROM	Electr. Erasable Programmable ROM	
FLASH	Nicht-flüchtiger Speicher	Programmspeicher
G-Code		
Gerber		
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
NC	Numerical Control	Numerische Steuerung
RS-232	siehe UART	Serielle Schnittstelle
RS-485		
ROM	Read-Only-Memory	Speicher (nur Lesen)
SPI	Serial Peripheral Interface	Serielle Schnittstelle
SRAM	Static Random-Access Memory	Speicher (???)
TWI	Two Wire Interface	Auch I2C-Bus
UART	Univ. Asynchron Receiver Transmitter	RS232
LC-Display	Liquid Crystal Display	Flüssigkristallanzeige
DIL	Double In Line	Bauform des Controllers
BAUD		
UBRRn		
VIA		
Brown-Out		

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 CADSoft⁶, 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 jedoch 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 Inkjet-Drucker 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

Der Vorteil dieses Verfahrens gegenüber den beiden anderen liegt darin das hier keine Chemie zum Einsatz kommt. Das designte 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

⁶ (CADSoft, 2)

bei größeren Layouts zeigt. Es kann vorkommen das, abhängig von der Platinengröße, dem Routing und den vorgenommenen Einstellungen an z.B. der Isolationsbreite, der Fräsvorgang mehrere Stunden in Anspruch nehmen kann. Auch der dabei entstehende Bohr- und Frässtaub muss abgeführt werden. Der entscheidende Vorteil dieser Methode ist die hohe Genauigkeit und Präzision. Eine Platine mit ca. 40 Bauteilen und ein paar VIA's 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 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 der Ausgabeformate eines Elektro-CAD-Programms ein interpretieren kann und für eine Standard-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 erfolgt aus dem 230V AC Netz, wird auf Schutzkleinspannung, zwischen 6V und 32V DC transformiert und als Versorgung für die CNC-Steuerung verwendet.

Die mittels Elektro-CAD-Software 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 6 Monate, Abgabetermin ist der 16.04.2012

2.3 Zeitplan

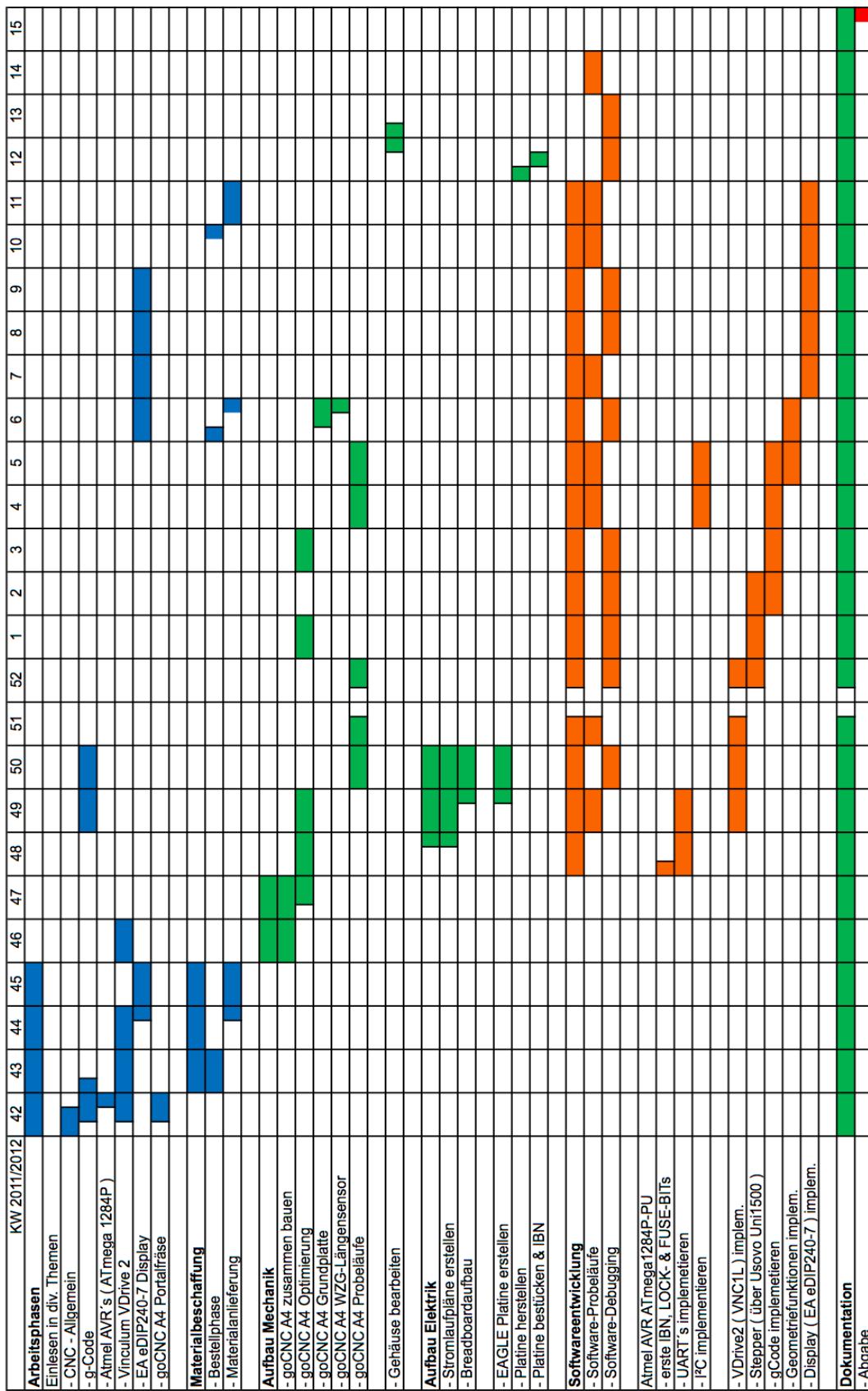


Abbildung 2.3.1: Projektzeitplan

2.4 Planung

Zunächst galt es die Rahmenbedingungen sowie den gesamten Umfang des Projekts fest zulegen. Dazu wurde das Projekt in zunächst 7 Teilabschnitte gegliedert:

1. Die Auswahl, der zu verwendenden Komponenten sowie die Erstellen der Pläne und dem Aufbau eines geeigneten Entwicklungssystems.
2. Die Kommunikation zwischen einem PC und dem Controller zum steuern und ausgeben von Meldungen herstellen.
3. Die Ansteuerung der Schrittmotoren, zunächst von einem einzelnen Motor, dann für 2 Motoren parallel.
4. Das Einlesen von Daten vom USB-Stick sowie die Bereitstellung und Übergabe der Daten für weitere Funktionen.
5. Die Auswertung des gCode aus den übergebenen Daten, Steuerung der daraus erforderlichen Aktionen.
6. Display-Anbindung, Display-Programmierung
7. Umsetzen des Entwicklungssystems auf eine Platine und der Einbau in eine geeigneten Gehäuse.

Die Dokumentation des gesamten Projektes läuft parallel zu alldiesen Schritten.

3 Hardware

3.1 Der Controller

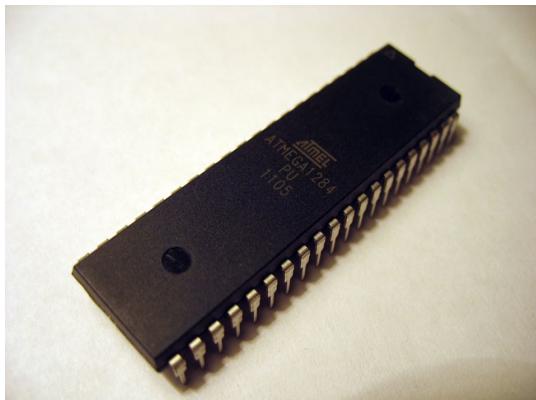


Abbildung 2.4.1: Atmel ATmega1284P-PU

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

5V DC und mind. 4 Schnittstellen, 2 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 Typs sind:

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

Als zusätzliche Eigenschaft, ist der ausgewählte Controller absolut Pin-Kompatibel zum ATmega32 und kann somit für einfach auf dem in der Entwicklungsphase eingesetzten Pollin ATMEL Evaluations-Board⁷ betrieben werden. Der Vorteil des Boards liegt darin, zunächst keine weitere Hardware zu benötigen. Alle wichtigen Komponenten zum Betrieb des Controllers sind auf der Platine vorhanden und können über Steckbrücken zugeschal-

⁷ (Pollin, 2007)

ten werden. Alle Pins sind über einen Wannenstecker heraus geführt und können frei nach Wahl beschaltet werden.

Die Kosten für diesen Controller liegen mit der benötigten Außenbeschaltung bei rund 8 Euro. Details zur Projektierung folgen im weiteren Verlauf.

3.2 Das Speichermedium



Abbildung 3.2.1: FTDI VDrive2 USB

Zum Transport der am PC generierten Daten, waren zunächst 2 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 der Standard SPI-Schnittstelle. Der Nachteil der SD-Karte ist der verwendete Spannungspegel. Nach den SD Association⁸-SD Standards liegt dieser bei 3,3 Volt.

Da der Controller und weitere Komponenten auf 5 Volt ausgelegt sind, wäre hier ein zusätzlicher Hardwareaufwand nötig gewesen. Der USB-Stick im Gegensatz zur SD-Karte, kann nicht direkt an den Controller angebunden werden. Hierzu ist ein zusätzlicher Schnittstellenbaustein nötig. Der Vorteil der USB-Variante liegt darin das auch ältere PCs bzw. Notebook 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 eine paar Hersteller, welche die hier benötigten Schnittstellenkomponenten anbieten. Der führende Hersteller in diesem Sektor ist die Firma FTDI⁹. 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 Vinculum VDrive2 Firmware 3.68. Er bietet 2 Schnittstellen auf der Hardwareebene. Zunächst war geplant den VDrive2 über SPI einzubinden. Abweichend vom Standard arbeitet das integrierte SPI mit 9 statt mit 8 Datenbits. Dies hätte zur Folge gehabt, dass die Kommunikation zwischen VDrive2 und Controller nicht über das Hardware-SPI des Controllers, sondern nur über eine bereit

⁸ (The SD Association, 1)

⁹ (FTDI Chip, 1)

zustellende 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 Das Display



Abbildung 3.3.1: Electronic Assembly eDIP240-7

Schon zu Beginn des Projektes stand bereits fest wie die Kommunikation zwischen Mensch und Maschine aufgebaut sein soll. Als HMI war ein Display vorgesehen. Der genaue Funktionsumfang und die Möglichkeit zur Steuerung der Maschine über ein Touch-Display ist der Firma ELECTRONIC ASSEMBLY zu verdanken. Dank der kostenfreien Überlassung eines „eDIP240B-7LWTP“ konnte dieses mit seinem vollen Funktionsumfangs in das Projekt implementieren. Bei dem überlassenen Display handelt es sich um ein monochromes LC-Display mit einer Auflösung von 240 x 128 Pixel. In diesem Display integriert ist eine Touchfolie im 4-Wire-System, welche direkt vom Display verwaltet und ausgewertet wird. Die Informationen der Touchfolie werden über die Schnittstellen des Display bereit gestellt. Es stehen 4 unabhängige Schnittstellen, sowie 8 I/Os zur Verfügung:

- 1x RS-232
- 1x RS-485
- 1x SPI
- 1x TWI
- 8x I/O

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 läuft über I2C mit dem SmallProtocol. Auf dieses Protokoll wird im Kapitel 4.5 SmallProtocol näher eingegangen.

3.4 Die Portalfräse

Die eigentliche Portalfräse als dezentrale Komponente ist nicht Bestandteil des Projekts und wird daher nur kurz beschrieben. Die Portalfräse mit der Bezeichnung „HOBBY A4“ wurde als Bausatz der Firma „GO!CNC.de¹⁰“ über die Homepage www.gocnc.de bestellt.



Abbildung 3.4.1: CNC-Bausatz www.gocnc.de "HOBBY A4"

Nach einer Lieferzeit von über 6 Wochen wurde der Bausatz von mir aufgebaut und in Betrieb genommen. Es wurden Modifikationen an dem Bausatz vorgenommen wie z.B. das montieren von 3 Endschaltern, der Einbau eines Werkzeuglängentasters. Die mitgelieferte Schrittmotor-Treiberplatine „UNI1500“ der Firma USOVO wurde im Originalzustand übernommen. Auf die Schnittstelle dieser Treiberkarte wird im nächsten Abschnitt näher eingegangen.

Die mitgelieferte 5mm-Holzplatte wurde durch eine 8mm-Aluminiumplatte ersetzt. Dies verleiht der gesamten Mechanik mehr Stabilität.

¹⁰ (goCNC, 1)

3.5 UNI1500

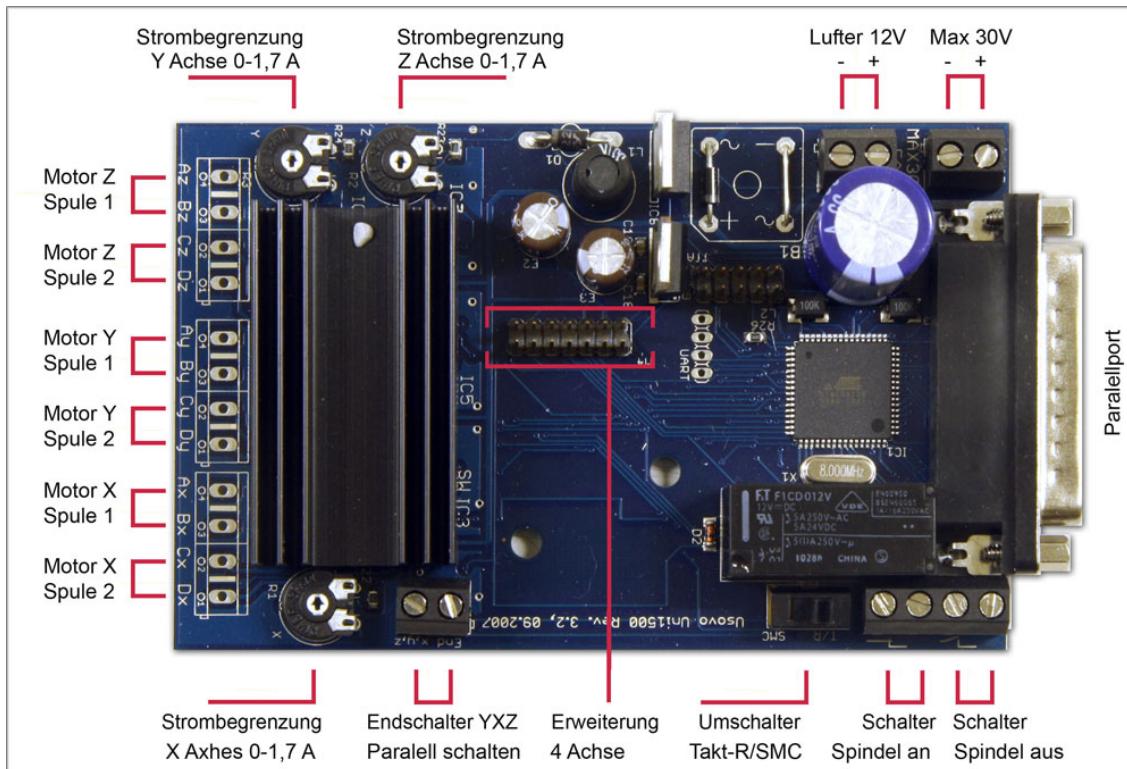


Abbildung 3.5.1: USOVO UNI1500 Steuerplatine

Die Steuerkarte ist auf Takt- / Richtungssignal eingestellt. Daraus ergibt sich folgendes 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 Poti's 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 und Präzision einer CNC-Maschine ist abhängig von einer Vielzahl von Faktoren wie z.B. der mechanische Aufbau, die Steifigkeit der Konstruktion, das Lagerspiel, das Führungsspiel, Toleranzen des Spindelsystems, die Hysterese der Referenzschalter, die verwendeten Motoren und nicht zu Letzt von der eingesetzten Steuerung. In der CNC-Technik wird die Genauigkeit und Präzision in zwei Gruppen unterschieden. Die Positionier- und die Wiederholgenauigkeit. Unter der Positioniergenauigkeit versteht man das genaue Anfahren der Zielkoordinaten. Bei der Wiederholgenauigkeit geht es darum, mit möglichst geringer Abweichung, ein und denselben Punkt mehrmals auf der selben Bahn anzufahren.

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

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

Die Steigung der hier verwendeten Gewindespindeln beträgt 1,5mm pro Umdrehung. Die Schrittauflösung der Schrittmotoren beträgt 144 Steps pro Umdrehung. Für eine Zielkoordinate von X 10,000 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 das die Genauigkeit der Berechnung von der Spindelsteigung und der Schrittauflösung der Motoren abhängt. Mit der hier eingesetzten Portalfräse errechnet sich eine theoretische Positioniergenauigkeit **0,01042mm**.

Ein weiterer Punkt der Einfluss auf die Positionier- und Wiederholgenauigkeit nimmt, ist das sog. Lagerumkehrspiel. Dies bezeichnet das Spiel zwischen Spindel und Mutter bei Richtungsumkehr bis die Umdrehungen der Spindel in ein Bewegung des Achse umgesetzt wird. Die Anzahl dieser Schritte nennt man Leerschritte.

3.6.1 X- & Y- Achsen

Zur Messung der Parallelität der Achsen zum Werkstücktisch wurde eine Messuhr mit einer Auflösung von 0,01mm verwendet. Beim Überfahren des Tisches ergab sich hier folgende Abweichungen:

Fahrweg	Ausschlag Messuhr
X:0mm – X:160mm bei Y:0mm Z: 20mm	+ / - 0,02mm
X 0mm – X 160mm bei Y:200mm 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 hingegen liegt außerhalb eines akzeptablen Bereichs. Dieses Spiel wurde durch schrittweises Takten der Motoren ermittelt. Dazu wurde das Werkzeug auf die Werkstückoberseite gefahren, soweit abgesenkt, bis Material abgetragen wurde. Von jetzt an wurde die zu Untersuchende Achse schrittweise, in entgegengesetzter Richtung der letzten Fahrt getaktet. Dabei wurden die Schritte gezählt bis wieder Material abgetragen wird. Hierbei zeigte sich:

Achse	Leerschritte
X-Achse	5
Y-Achse	42

Abbildung 3.6.2: Leerschritte

Die X-Achse ist mit 5 Leerschritten, 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, ein Kompensationsfaktor Software-Seitig einzubinden, da mit dieser Abweichung das fräsen kleiner Leiterbahnen unmöglich ist. Dies wird besonders deutlich wenn sich ein am Stück gefrästes Quadrat mit einer Seitenlänge von 10mm genauer betrachtet. Hier zeigt sich an der rechten, oberen Ecke des linken Quadrates ein Überfräsen, im linken Quadrat ein Unterfräsen. Dieser Versatz entspricht den 42 Leerschritten.

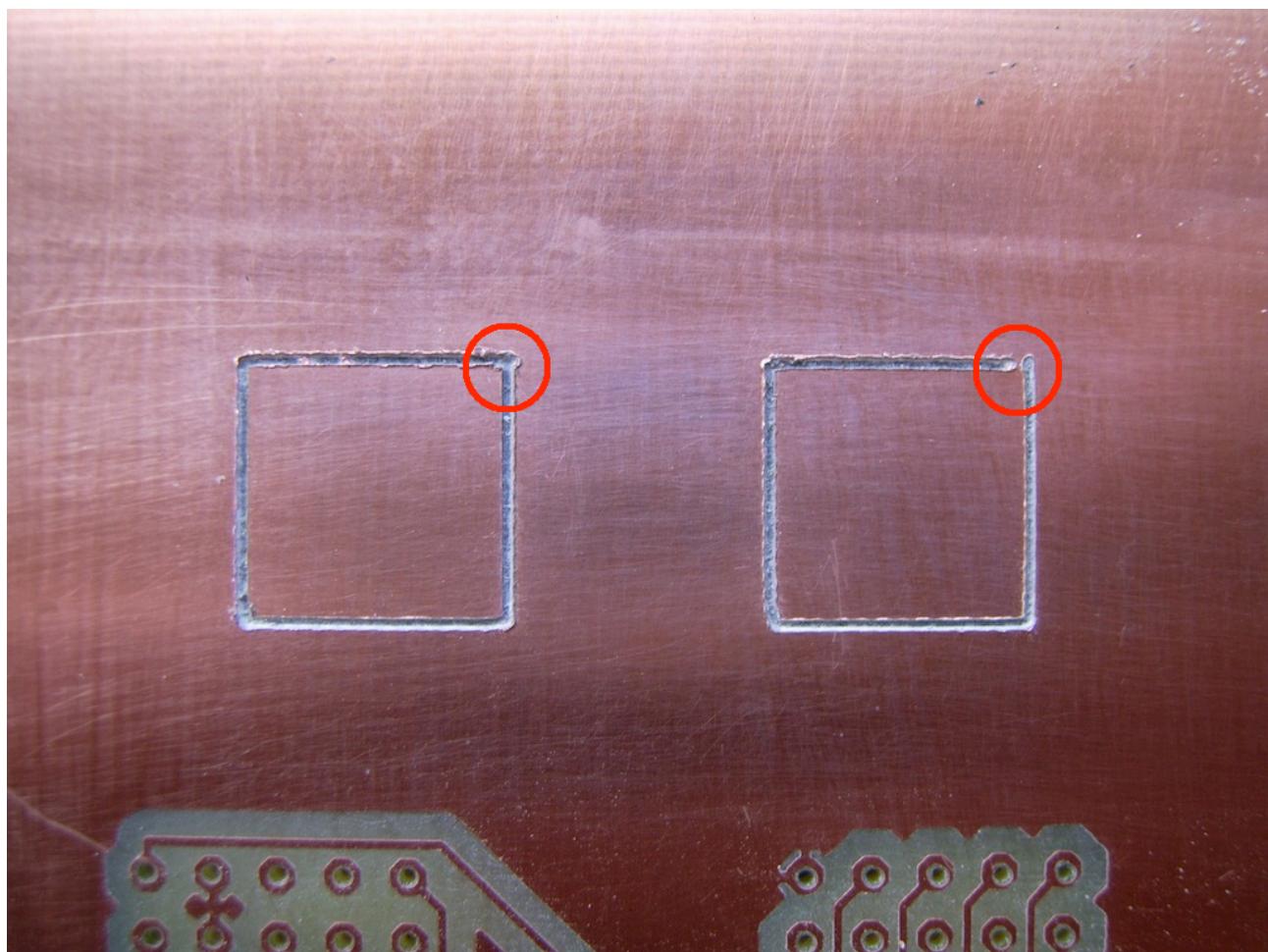


Abbildung 3.6.3: Umkehrspiel der Y-Achse

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 zeigte, 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 vermutlich 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 Spinden auf. Dadurch wird jeder Schritt des Z-Achsenmotors direkt in einen Bewegung umgesetzt.

3.7 Geschwindigkeit

Die Bewegungsgeschwindigkeit einer Achsen errechnet sich aus der Drehzahl der jeweiligen Spindel, sowie 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.

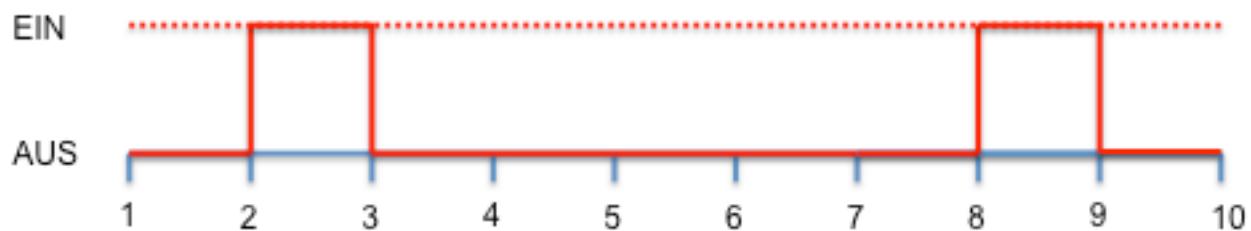


Abbildung 3.7.1: PWM Schrittmotor

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

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 **694mm/min**. Wird eine kleinere tAUS-Zeit gewählt, kommt es bei der Y-Achse zum überspringen von Schritten. Dies hat dann den Verlust der Referenz zur Folge. Es muss eine neue Referenzfahrt durchgeführt werden. Dies hat wiederum den Abbruch des aktuellen Programms zur Folge.

3.8 Spindeldrehzahl

Zur Drehzahl der Spindel kann gesagt werden, je höher, desto ein besseres das Ergebnis. Eine Drehzahl von 25.000rpm hat sich für ein sauberes 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 0,35mm und einer Drehzahl von 35.000rpm sauber. 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, fiel 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%¹¹ liegt noch in einem akzeptablen Bereich. Ist der Versatz >3% ist eine sauber und einwandfreie Übertragung nicht mehr möglich. Die Ablaufsteuerung des UART wird von 4 sog. SpecialFunctionRegistern übernommen. Eines dieser Register ist das **UART BaudRateRegister**. Es wird zur Bestimmung der Kommunikationsgeschwindigkeit, der sog. BAUD, mit einem Teilerwert belegt. Die BAUD und den Teilerwert für UBRRn errechnet sich wie folgt:

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

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

fOSC = Oszillator- oder Quarz-Frequenz

UBRRn = Teiler der n-UART

BAUD = Übertragungsfrequenz

n = Index der Schnittstelle

¹¹ (MAXIM, 2003)

Mit einem fOSC = 20MHz Quarz errechnet sich ein UBRRn 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 außerhalb 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 deaktiviert. Bei einem Absturz des Programms hat 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 wurde auf Standard 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 wurde 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



Abbildung 3.9.1: Logo 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.“¹²

Der Umgang mit dieser Software wird als bekannt vorraus gesetzt.

¹² (Wikipedia, 2012)

3.9.4 PCB-GCODE-ULP 3.5.2.11

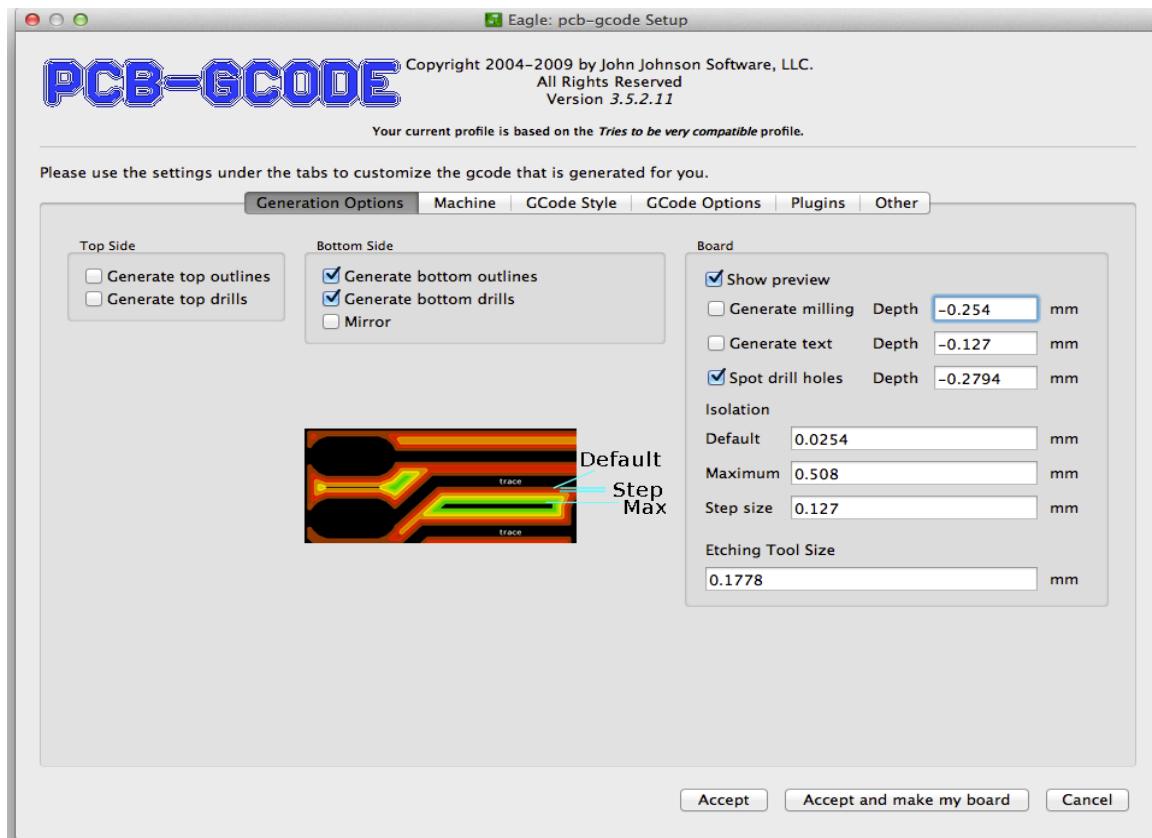


Abbildung 3.9.2: PCB-GCODE Menu

Ein Universal Language Programm, kurz ULP, ist ein in C programmiertes Unterprogramm welches direkt aus EAGLE Board heraus gestartet. Das hier verwendete ULP PCB-GCode¹³ 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: | 16mm | Warteposition über Platine |
| ➤ Z-Up: | 2.54mm | Arbeitsposition über Platine |
| ➤ Z-Down: | - 0.1778mm | Eintauchtiefe beim fräsen |
| ➤ Z-Drilldepth: | - 0.75mm | Bohrtiefe |
| ➤ Position Z: | 40mm | Werkzeugwechselhöhe |

GCode Style:

generic.pp Standart gCode

¹³ (Johnson, 2009)

GCode Options:

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

File Naming:

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

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 V2.0.1¹⁴ 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 wurde jedoch der VDrive2 über diese UART0 implementiert. Um dennoch die Möglichkeit zum Debuggen zu erhalten, wurde das ganze System auf einen ATmega32 portiert. Hier stand ab sofort die JTAG Schnittstelle zur Verfügung. In der vorletzten Entwicklungsstufe wurden alle Funktionen zusammen geführt. Es war nun erforderlich, zur Steuerung des Ablaufs, dem Controller Befehle zuzuführen. Diese Funktion war ursprünglich für die UART0 vorgesehen. Somit wurde das System auf den ATmega1284P-PU portiert und wie folgt belegt:

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

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

¹⁴ (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 http://jump.to/fleury Dennis Hohmann
i2cmaster.c	i2cmaster.h	I2C-Library	Peter Fleury http://jump.to/fleury
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

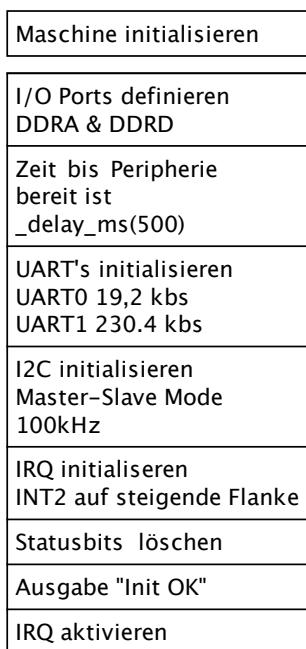


Abbildung 4.2.2: Struktogramm Init

Zu Beginn des Anlaufs, wird das komplette System initialisiert. Es werden als erste Aktion, die I/O Register des Controllers gesetzt. Im nächsten Schritt wird 500ms gewartet, bis die ganze Peripherie ebenfalls vollständig angelaufen ist. Diese Zeit ist besonders für den VDrive2 wichtig, da ein zu frühes Ansprechen, diesen im Anlauf unterbricht. In diesem Zustand hilft nur ein Spannungsreset des VDrive2.

Nach Ablauf der Wartezeit werden die seriellen Schnittstellen initialisiert.

Sind auch diese mit der entsprechenden BAUD initialisiert, wird nun der serielle Datenbus initialisiert. Ist auch diese Schnittstelle erfolgreich initialisiert, wird das Interrupt-Control-Register des ATmega1284P mit der gewünschten Betriebsart geladen. Diese sieht einen Interrupt-Auslösung bei fallende Flanke, also „falling edge“ vor. Zuletzt werden noch die entsprechenden Statusbits im Bitfeld **M_FLAGS** gesetzt. Nun ist der Controller bereit und gibt diese Status-

meldung an das Display aus. Ab jetzt sind die Interrupts aktive, der Anlauf ist beendet. Sollte einer dieser Initialisierungsschritte nicht erfolgreich verlaufen, wird die entsprechende Meldung über die bereits zur Verfügung stehende Schnittstelle ausgegeben.

Ab jetzt wird die main-Schleife zyklisch durchlaufen. Nach jedem Durchlauf wird das **M_FLAGS.LIFEBIT** getoggelt. Diese Bit wird direkt über den PinD5 an eine LED ausgegeben.

Es werden folgende Bits aus M_FLAGS zyklisch ausgewertet und die entsprechenden Aktionen eingeleitet.

GCODE_FILESET

Ist dieses Bit == 1, wird der, 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 anstehen, jedoch ist das Abbruch-Bit dominant ausgeführt und überschreibt das Automatik-Bit zum Ende des aktuellen Zyklus.

EDIP_NEWPOS

An dieser Stelle wird geprüft, ob sich die Position einer beliebigen Achse zum letzten Zyklus verändert hat. Wenn dies der Fall ist, wird die Routine zur Aktualisierung der Positions-werte im Display aufgerufen. Dieses Bit wird durch die Routine wieder gelöscht.

4.2.1 Globale Definitionsdatei globdef.h

Diese Header-File ist die zentrale Definitions-Datei. Hier wird die globale Struktur **M_FLAGS** deklariert. Eine Struktur ist ein Datensatz der aus verschiedensten Werten und Formaten zusammen gesetzt sein kann. Die hier verwendete Struktur besteht größten Teils aus Bitfeldern. Diese haben den Vorteil, dass die Größe einer Variablen manuell festgelegt werden kann. Standardmäßig wird für eine Variable vom Datentyp CHAR 8 Bit reserviert. Somit kann diese einen Wert von 0 bis 255 bei unsigned bzw. -128 bis 127 bei signed annehmen. Der Unterschied zwischen der Definition signed und unsigned liegt darin das signed Variablen positiv und negative Werte annehmen können. Unsigned können nur positive oder 0 sein. Ist die Variable als Bitfeld mit einer Länge von 1 deklariert, werden zwar immer noch 8 Bit reserviert, jedoch nur noch 1 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
unsigned CHAR x (8 Bit)																
unsigned CHAR y (8 Bit)																
unsigned CHAR x als Bitfeld Länge 1																
unsigned CHAR y als Bitfeld Länge 1																

Abbildung 4.2.3: Bitfeld mit 2 unsigned CHAR Variablen

Mit dieser Methode kann gerade bei der Verwendung mehrerer Variablen sehr viel Platz gespart werden. 8 Variablen vom Typ unsigned CHAR benötigen 64 Bit Speicherplatz. Wird jeweils nur 1 Bit ausgewertet, sind 56 Bit „verschwendet“. Definiert man diese 8 unsigned CHAR Variablen in einem Bitfeld je mit der Länge 1, so benötigen diese nur noch 8 Bit Speicherplatz.

Diese Definitionsdatei enthält des Weiteren die Frequenzen F_CPU (Quarz-Frequenz) und 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

Axis usw...

BAUSTELLE

4.2.3 USB-Kommunikation vnc1l.c

Wurde der ein Stick gesteckt und korrekt erkannt, kann eine entsprechende Datei aus dem ROOT-Verzeichnis vom Programm mit der Funktion „usb_open_file(FILENAME)“ geöffnet werden. Dateien aus Unterverzeichnissen werden aktuell nicht unterstützt. Ist die Datei erfolgreich geöffnet, stehen 2 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.

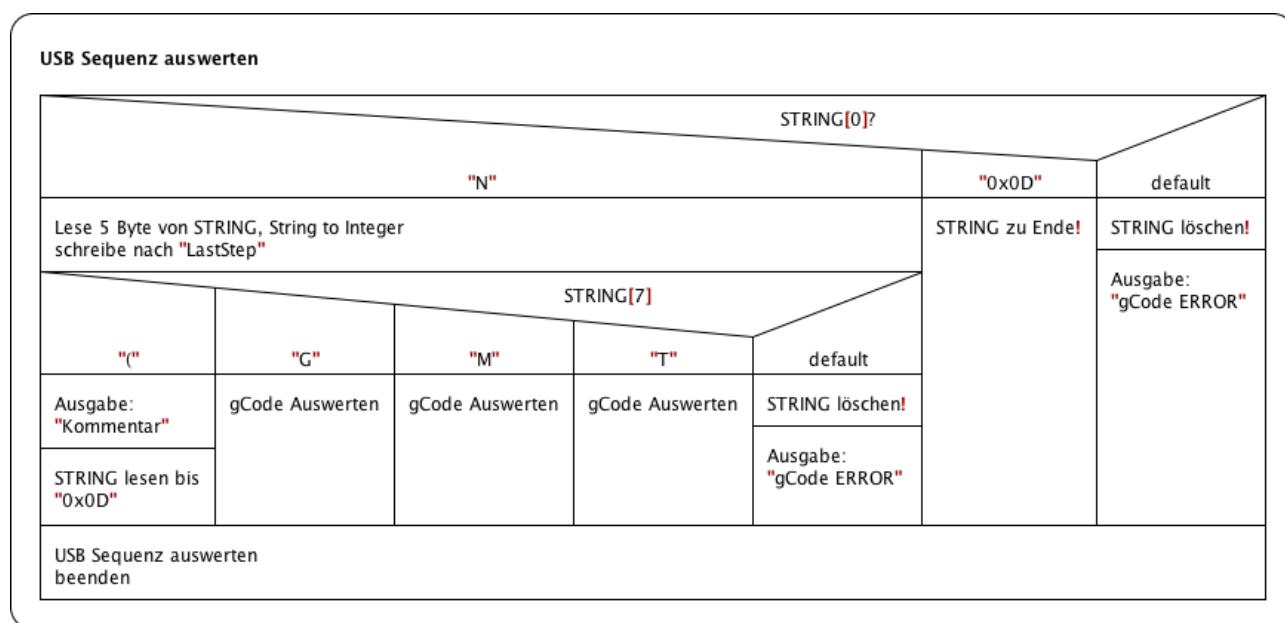


Abbildung 4.2.4: USB-Sequenz

Von dort kann der STRING mit der Funktion „usb_get_command(void)“ wie im Struktogramm dargestellt, ausgewertet werden.

Die Funktion „usb_string_reset(void)“ wird dieser STRING verworfen.

Mit der Funktion „usb_get_state(void)“ wird im Automatik-Modus geprüft, ob der USB-Stick noch vorhanden ist, oder ob er gezogen wurde.

4.2.4 Serielle Schnittstelle uart.c

Diese Bibliothek wurde von Peter Fleurys „UART-Library“ aus der Version 1.6.2.2 übernommen und um eine, bisher nicht existierende Funktion erweitert. Die neue Funktion dient dem einlesen von Strings, welche durch ein Trennzeichen, dem „seperator“ 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, seperator
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 4.2.5: uart_gets()

Weiter Funktionen dieser Library sind unter anderem uartx_init(BAUD), uartx_getc(), uartx_putc(CHAR) und uartx_puts(STRING). Diese Funktionen sind jeweils für beide UARTs ausgelegt. Weitere Informationen zu dieser Library, sind der Dokumentation von Peter Fleury zu entnehmen.

4.2.5 Serieller Datenbus i2cmaster.c

Wie auch die UART-Library, stammt die hier verwendete I2C-Library ebenfalls von Peter Fleury. Sie wurde komplett in der Version 1.3 ohne Änderungen übernommen.

Es werden 4 wichtige Funktionen aus dieser Library 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 aus nachfolgender Grafik deutlich.

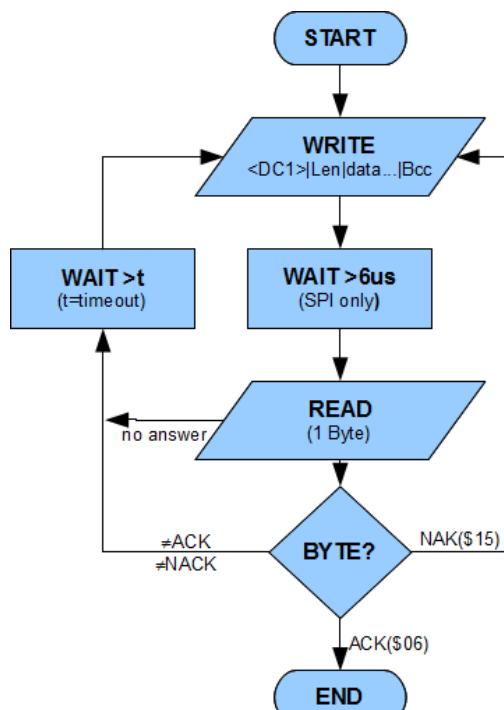


Abbildung 4.2.6: I2C-Kommunikation

Weitere Informationen zu dieser Library, sind der Dokumentation von Peter Fleury¹⁵ zu entnehmen.

¹⁵ (Fleury, Peter Fleury UART Library, 2006)
(Fleury, Peter Fleury I2C Master library, 2006)

4.2.6 Display-Handling edip240.c

BAUSTELLE

4.3 gCode-Struktur

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 strukturierten Klartext und durch die DIN¹⁶ definiert. Ein Programm ist nach einem definierten Syntax aufgebaut. Die kleinste Informationseinheit ist ein **Wort**. Mehrere Worte, mindestens jedoch 2, 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 10er-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 unterteilt 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. Geometriefunktionen werden, mit einigen Ausnahmen, begleitet von einem oder mehreren weiteren Wörtern. Die Funktion **G00** als Beispiel wird immer begleitet von Koordinaten. Zusätzlich können zu Geometriefunktionen noch weitere Worte, wie **Fxxxxx** angehängt werden. Dieses Wort gibt die Feed-Rate, also die Geschwindigkeit des aktuellen Satzes an.

¹⁶ (Deutsches Institut für Normung e.V., 1)

N00120 G01 X12.250 Y-5.750 F120.0

G01 gibt an das es sich bei diesem Satz um eine Interpolationsbewegung (diagonale Bewegung) handelt. Die begleitenden Zielkoordinaten sind 12.250 für X, -5,750 für Y. Als Geschwindigkeit sind 120.0 angegeben. Die Einheiten dieser Angaben werden zu Beginn des Programms über die Programmdateien **G20** für inch, bzw. **G21** für mm festgelegt.

Ebenso wie die Eigenschaft der Koordinaten. Handelt es sich um absolute Koordinaten wird dies zu Beginn mit den Wort G90, bzw. mit G91 für relative Koordinaten definiert. Beide richten nach dem Kartesischen Koordinatensystem¹⁷.

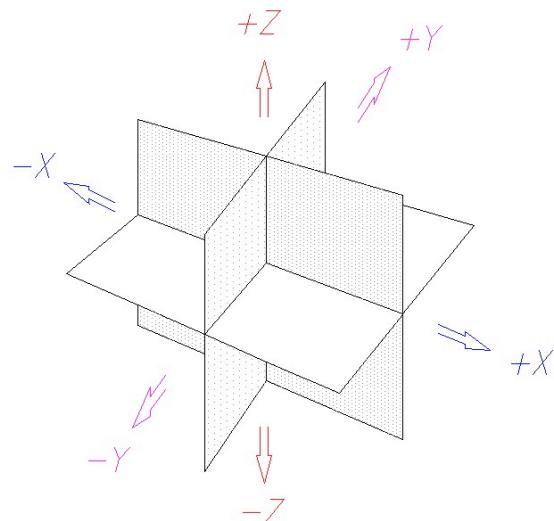


Abbildung 4.3.1: Kartesisches Koordinatensystem

¹⁷ (Universität Karlsruhe, 1)

Kommentare werden mit „(“ eingeleitet und mit „)“ beendet. Ein Programm für ein Loch bei den Koordinaten X10 /Y10 mit der Tiefe 10mm würde wie folg aussehen:

```
N00000 G21
N00010 G90
N00020 G00 X10 Y10
N00030 G00 Z-10
N00040 G00 Z10
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 mit **Txx** Wort begleitet wird. Das Txx gibt die Nummer des nächsten zu verwendenden Werkzeug an. **M06 T02** wählt das Werkzeug 2 aus.

4.4 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 4.4.1: EA KitEditor Programmdateien

Der Umgang mit dem KitEditor ist selbsterklärend, die Hilfe ist sehr ausführlich mit zahlreichen Beispielen. Auch das Demo-Programm ist ausreichend erklärt. Der komplett zur Verfügung stehende Befehlssatz ist sowohl im Datenblatt des Display, als auch in den Beispiel-Programmen vorhanden.

Die Struktur 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, welche alle benötigten Dateien deklariert. Diese sind im aktuellen Projekt 5 Dateien erkennbar an der Endung .kmi. Dieses 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 mehrerer Befehlen. Es gibt grundsätzlich 2 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 2 unterschiedlichen Wegen ausgelöst werden. Zeitgesteuert oder Eventgesteuert. Die Zeitgesteuerten Makros werden für eine bestimmte Zeit oder nach Ablauf einer bestimmten Zeit ausgeführt. Eventgesteuerte Makros können über eine Zustandsänderung an einem I/O-Pin des Displays oder über die Touch-Folie ausgelöst oder beendet werden. Die letzteren nennt man Touchmakros. Makros können beliebig verschachtelt werden, der Nachteil hierbei ist jedoch der Verlust der Übersichtlichkeit.

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

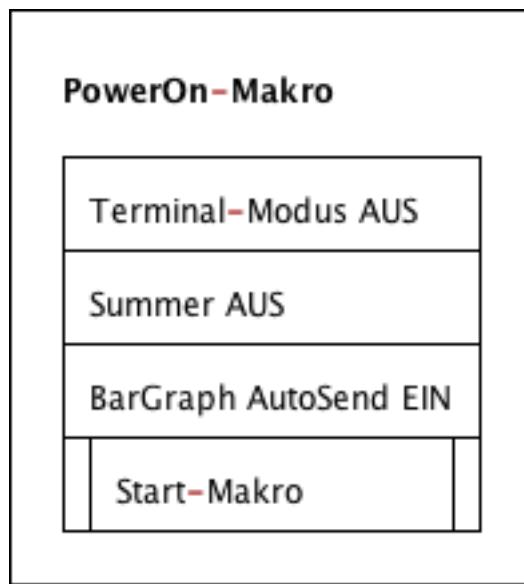


Abbildung 4.4.2: Stuktogramm PowerOn-Makro

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

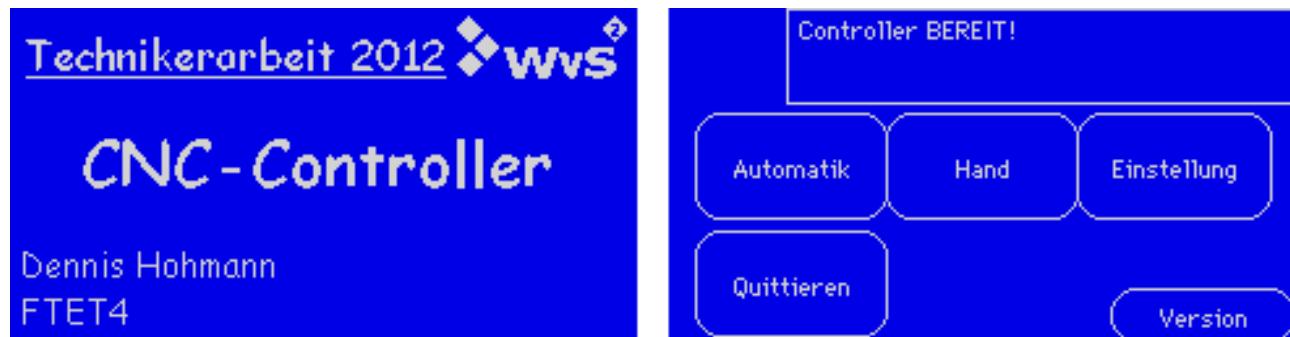


Abbildung 4.4.3: Start des Controllers

Ist der Controller ohne Fehler angelaufen, wird nach ca. 5 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 2 Klassen von Meldungen. Einfache Meldungen und Warnungen. Letztere sind mit einem blinkenden Symbol gekennzeichnet und erfordern einen Eingriff durch den Benutzer. Diese Meldungen müssen quittiert werden.



Abbildung 4.4.4: Warnung

Das Hauptmenu ist übersichtlich in 3 Gruppen aufgeteilt. Automatik, Hand und Einstellungen. Es sind Menu-Punkte vorbereitet, welche jedoch noch nicht vollständig verfügbar sind. Diese sind mit ein „!“ 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.

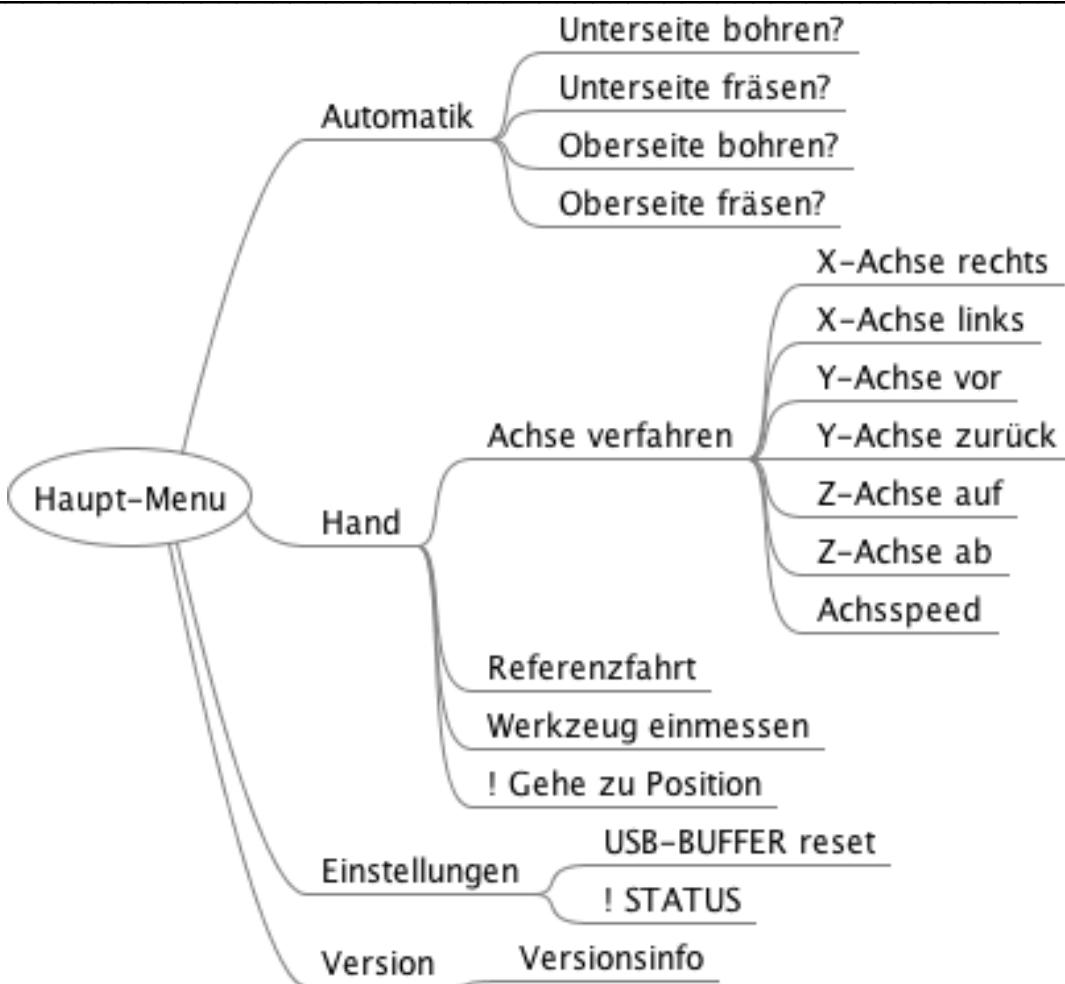


Abbildung 4.4.5: Menu-Struktur

Die Gruppe der Automatik-Funktionen enthält die aktuell 4 möglichen Funktionen zur Herstellung 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.

4.5 SmallProtocoll

Das SmallProtokoll ist ein Protokollrahmen, unabhängig von der verwendeten Schnittstelle. Es wird in 2 Handshakegruppen unterscheiden, DC1 und DC2, gefolgt von der Anzahl der Nutzbytes und der Checksumme.



Abbildung 4.5.1: Protokollrahmen¹⁸

Die Checksumme wird durch die Summierung des Handshake-Typs, der Anzahl der Nutzdaten, sowie allen Nutzdatenbytes errechnet. Modulo-Operation Summe / 256 ergibt die Checksumme. Ist die Checksumme korrekt, sendet das Display ein ACK. Diese Antwort ist jedoch nur die Antwort, 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 2 mögliche Reaktionen des Displays. Die erste Möglichkeit ist die Bestätigung des fehlerhaften Pakets mit NAK. Die 2. Möglichkeit ist das Ablauen 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:

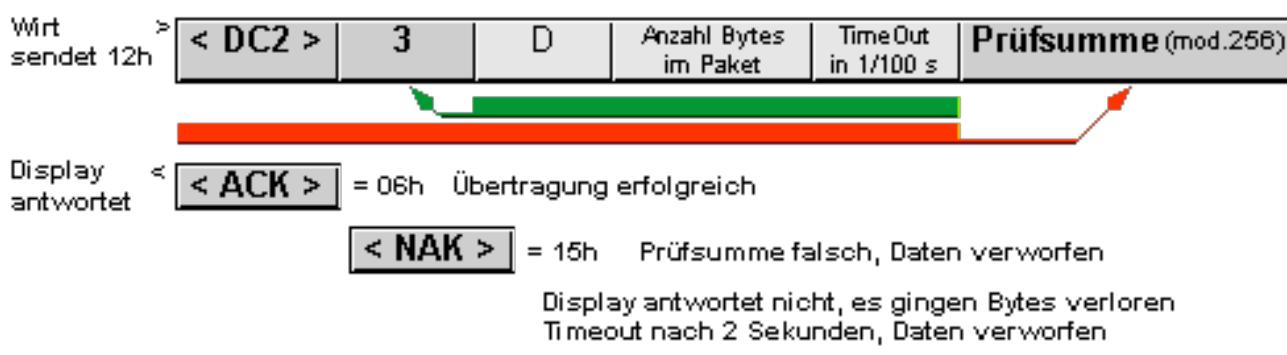


Abbildung 4.5.2: Protokollrahmen DC2¹⁹

¹⁸ (www.goBlack.de, 1)

¹⁹ (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

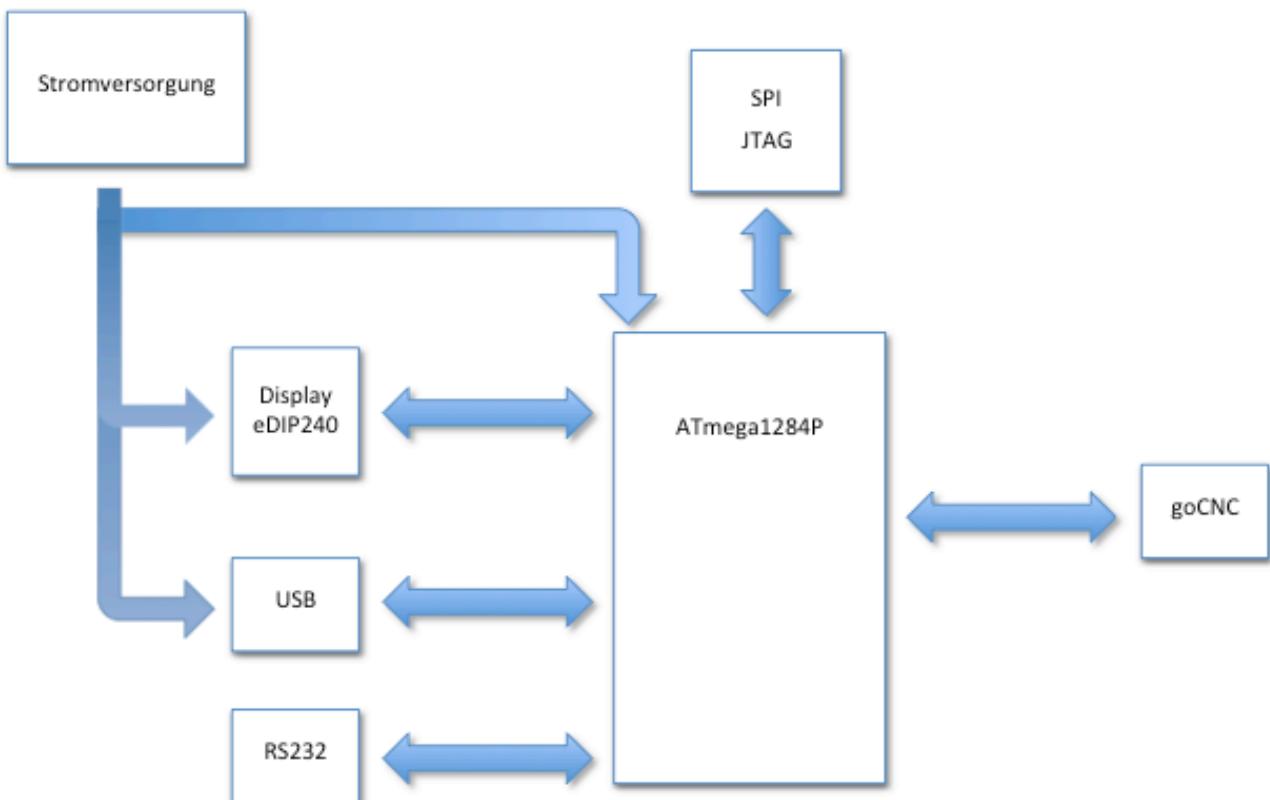


Abbildung 5.1.1: Blockschaltbild Steuerplatine

Die Stromversorgung 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 35V und 4 Ampere ausgelegt. Der Festspannungsregler 7805 ist mit 1 Ampere belastbar. Die Leiterbahnen der Versorgungsspannung sind in 0,60 mm ausgeführt. Die Berechnung der Leiterbahnbreite wurde mit der PCB-Toolbox iPhone App²⁰ unter Berücksichtigung einer Doppelseitigen Platine mit einer Kupferstärke von 35um und einer maximalen Erwärmung von 4°C bei 1A durchgeführt. Die

²⁰ (LeitOn, 1)

IC's ATmega1284P-PU sowie der MAX232 sing gesockelt eingelötet, alle weiteren Bauteile sind direkt eingelötet.

5.2 Das Designe

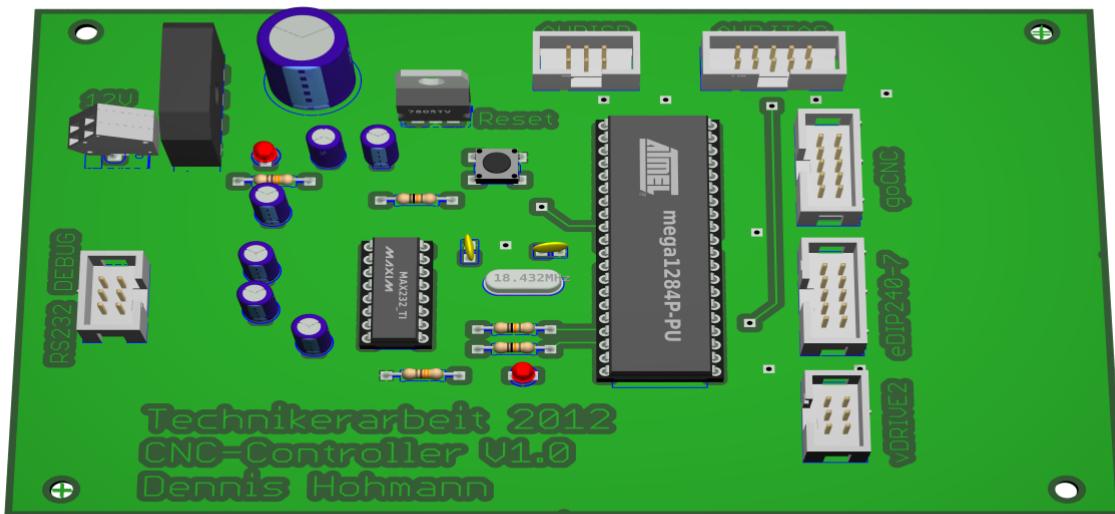


Abbildung 5.2.1: Steuerplatine Designentwurf

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

Aufgrund der geringen Anzahl an Bauelementen, wurde auf die Verwendung von SMD-Bauteilen verzichtet. Das Designe wurde zur besseren Übersicht und Lötbarkeit, doppelseitig ausgeführt, da so die Verwendung von Lötbrücken entfällt und diese durch VIA's ersetzt werden.

6 Zusammenfassung

6.1 Fazit

Das Projekt "CNC-Steuerung" hat sich, durch das Zusammenspiel verschiedenster Komponenten, während des 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 dargestellten Zeitplan verlaufen. Alle der 7 Teilabschnitte sind erfolgreich umgesetzt. Leider ist es auf Grund mechanischer Ungenauigkeit nicht gelungen, die Steuerplatine komplett mit der hier eingesetzten Portalfräse herzustellen. Das Bohren der Bauteil-Löcher, sowie der Via's, 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 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 Technikerarbeit weiter Modifikationen und Optimierungen an Hard- und Software geben. Zum einen ist die Kompensation des Lagerumkehrspiels ein großes Thema, zum anderen, die Positionierung auf einen direkt, als Koordinaten eingegeben Position. Dieses Projekt wird auch in Zukunft weiter entwickelt werden.

7 Literatur & Quellverzeichnis

- 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
- Fleury, P. (2006). *Peter Fleury UART Library*. Abgerufen am 29. 11 2011 von
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

8 Softwareverzeichnis

Firma	Software	Version	Quelle
Apple	Mac OSX Lion	10.7.3	Kostenpflichtig UNIX-Betriebssystem
VMware	VMware Fusion	4.1.0	Kostenpflichtig Mac OSX VirtualMaschine
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 OSX	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 & Versionen

9 Anhang

9.1 Schaltpläne

9.1.1 Mainboard

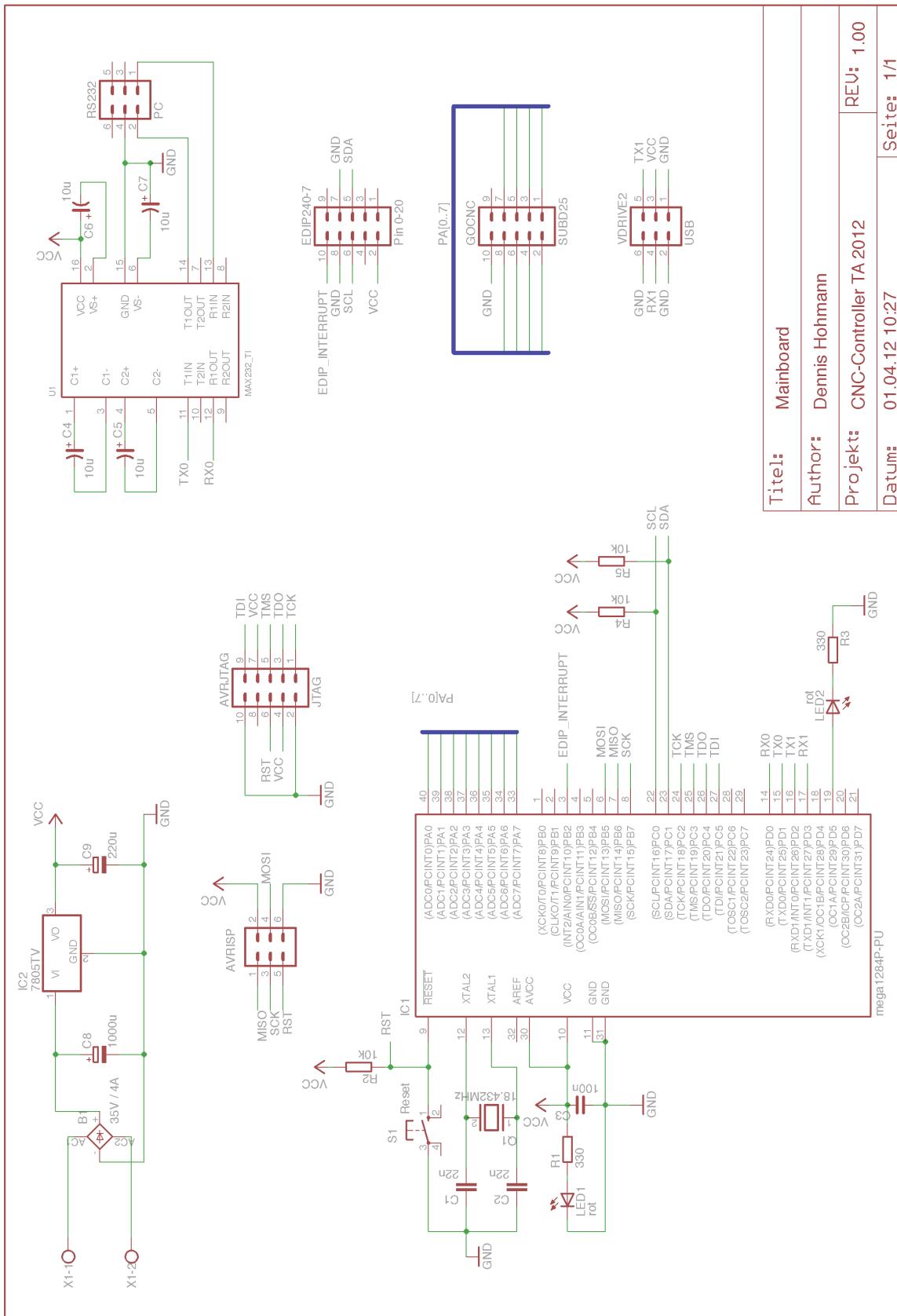


Abbildung 9.1.1: Schaltplan der Steuerplatine

9.1.2 Dip240-7

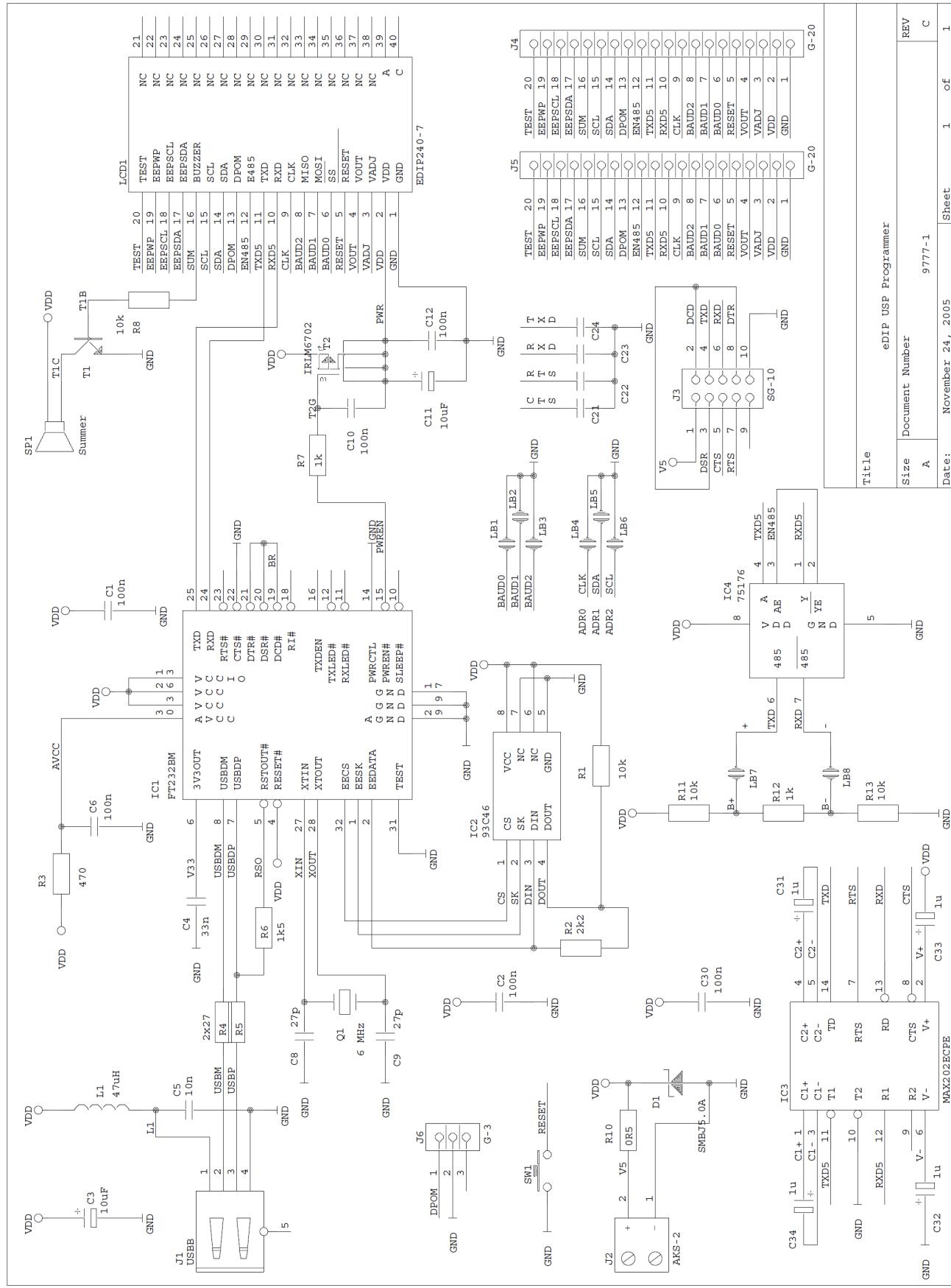


Abbildung 9.1.2: Schaltplan des eDIP240 mit USB Board

9.1.3 VDrive2

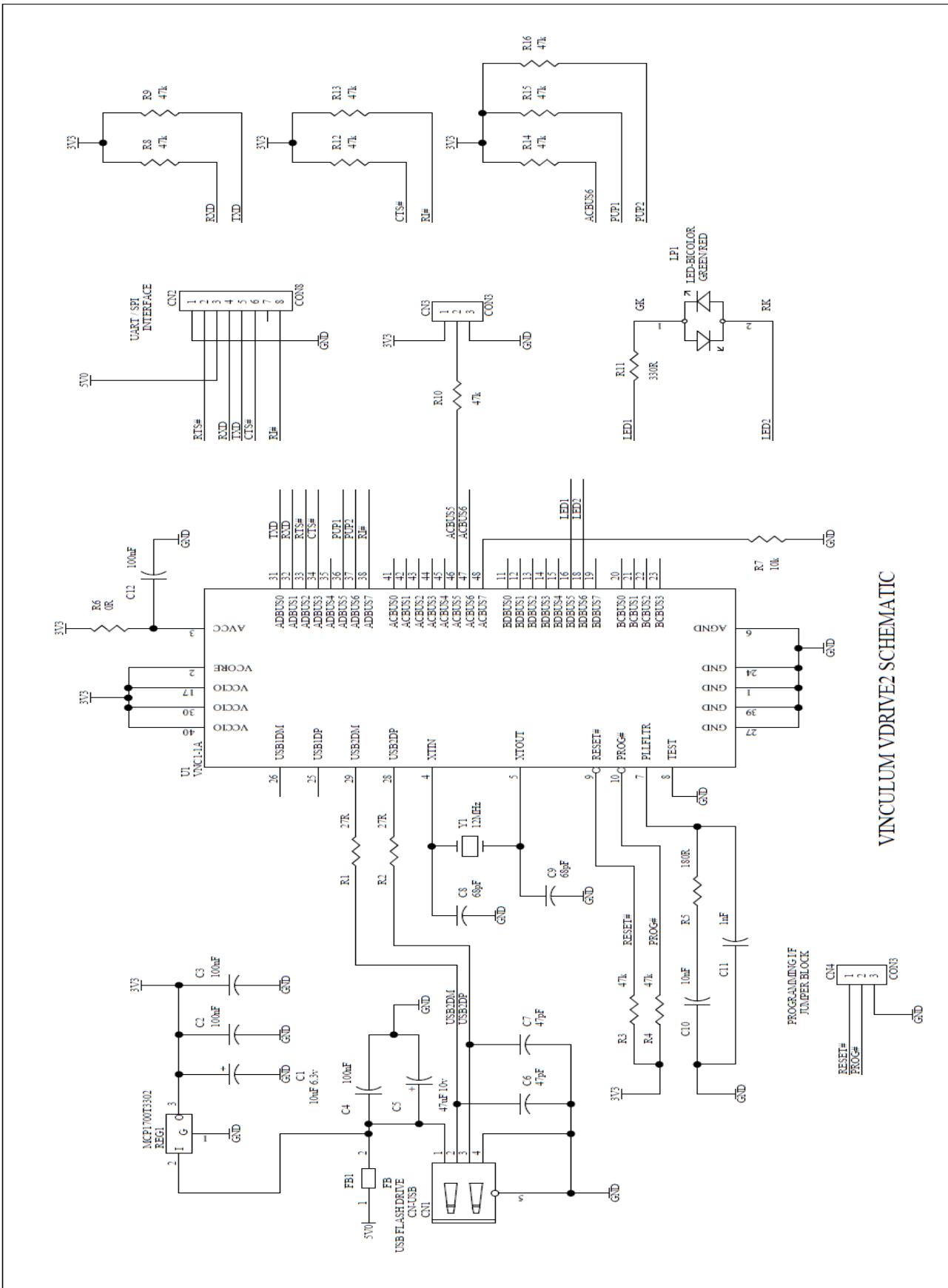


Abbildung 9.1.3: Schaltplan VDrive2

9.1.4 Board-Layout

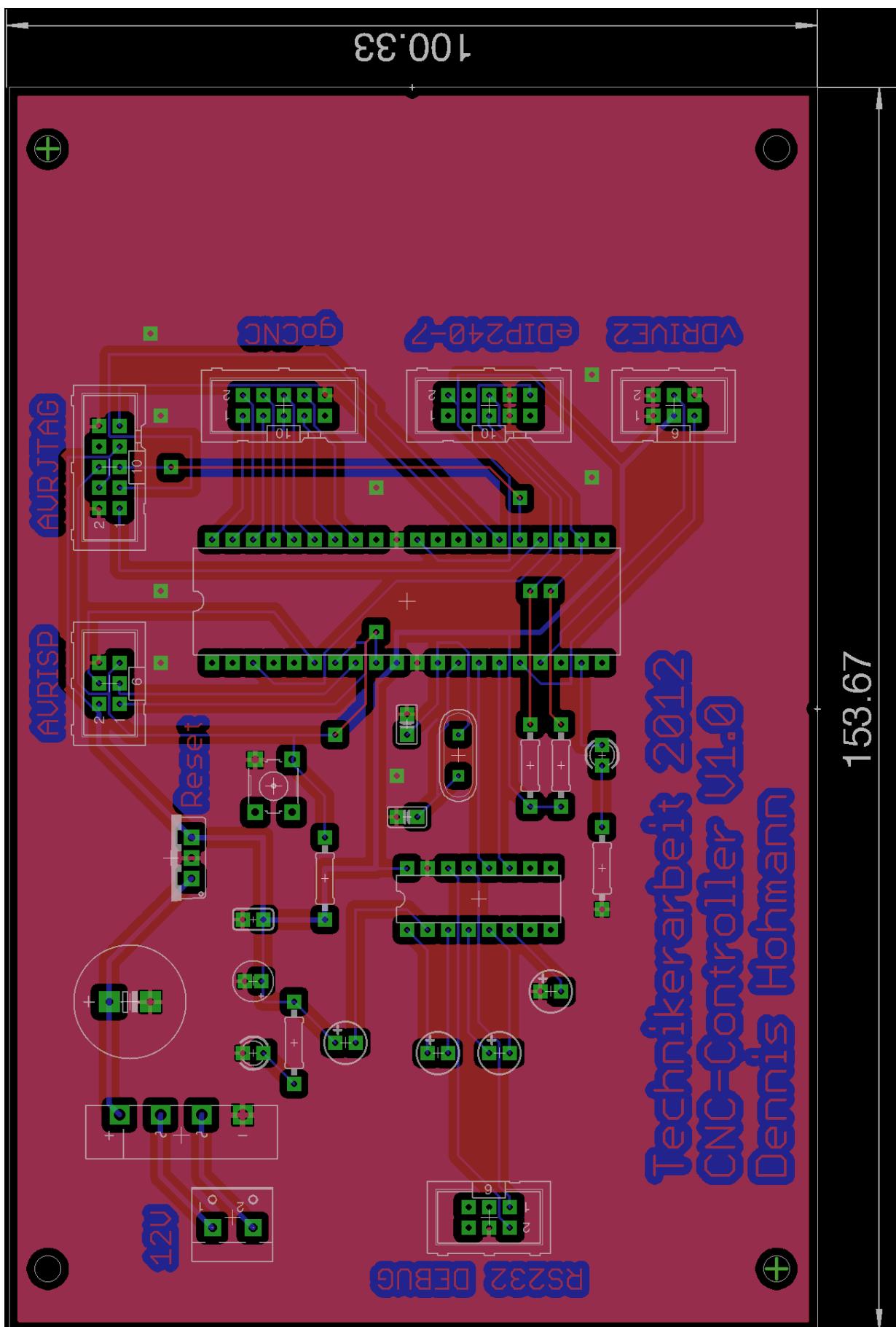


Abbildung 9.2.4: Layoutplan

9.1.5 Gehäuse

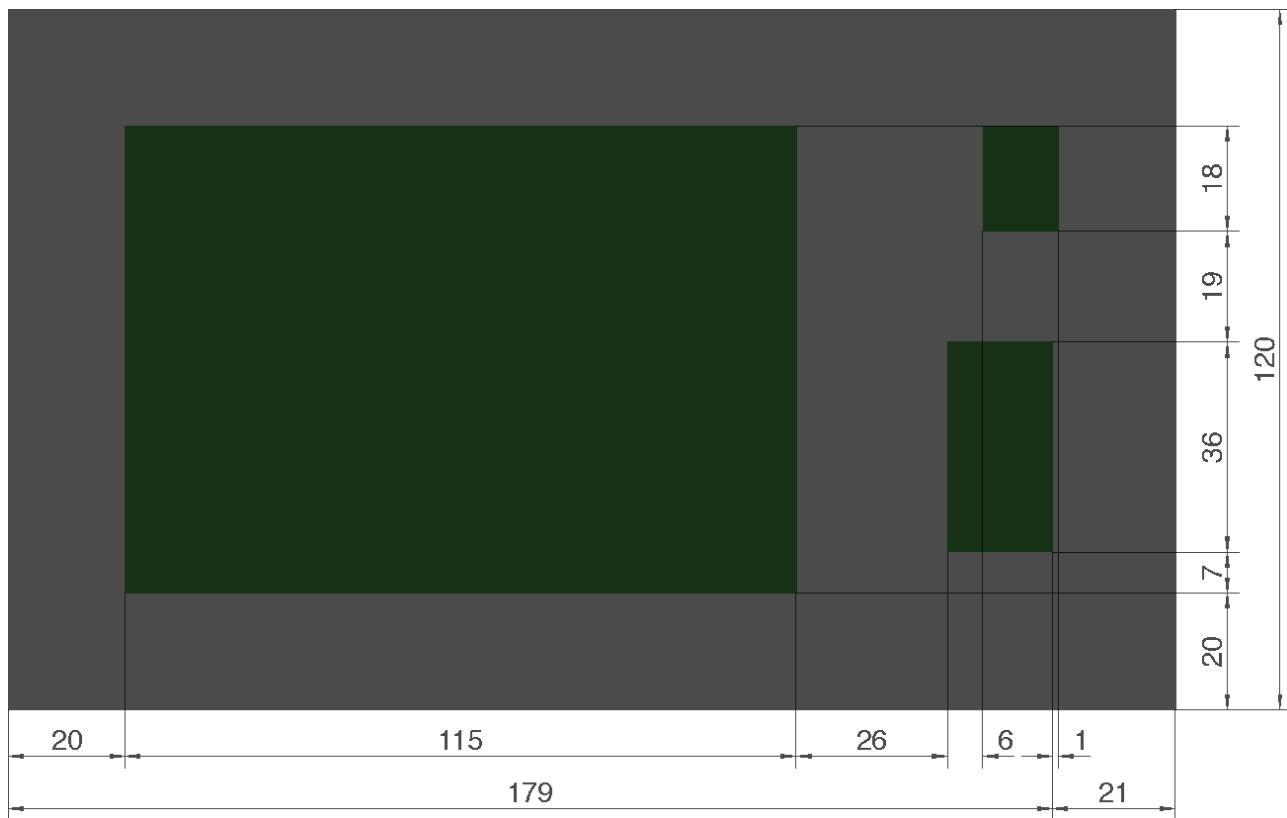


Abbildung 9.1.4: Gehäuse-Oberseite

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

9.1.6 Fertige Platine

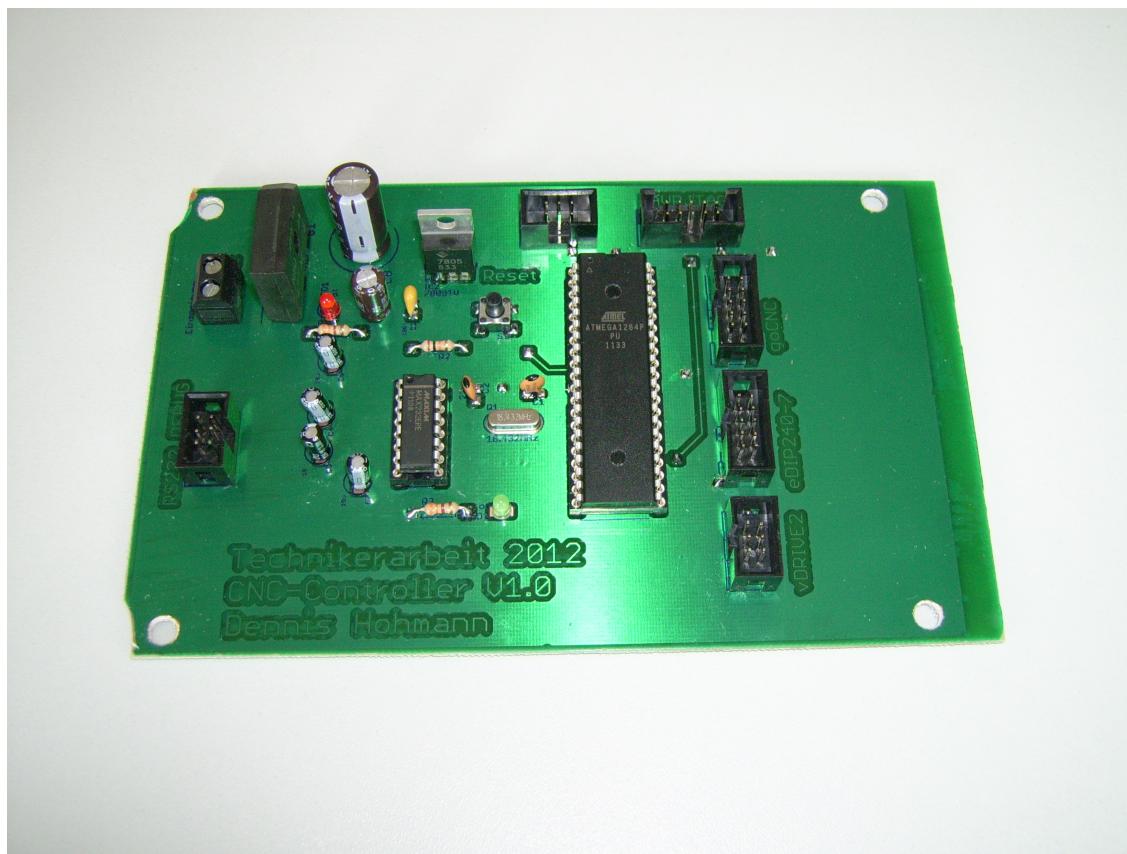


Abbildung 9.1.5: Fertige Platine

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