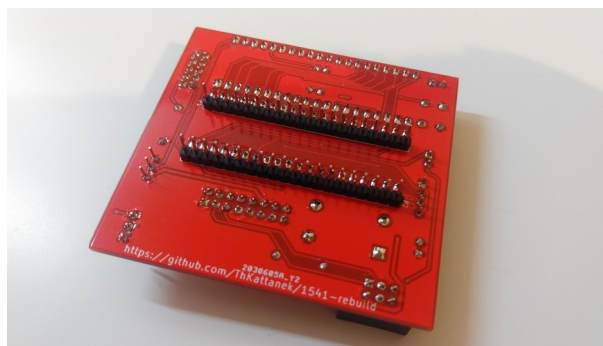
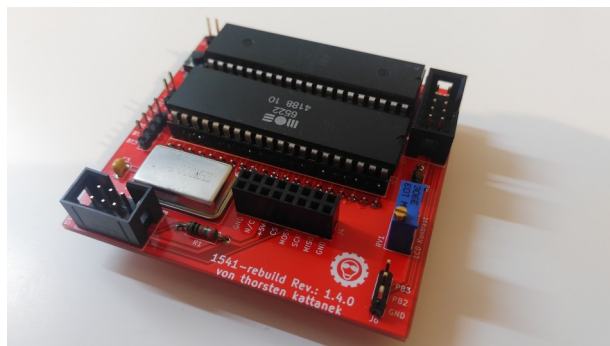


1541-rebuild

von Thorsten Kattaneck
Berlin, 30.05.2020

PCB Rev.: 1.4.0

Firmware: 1.3.0 (Noch kein offizielles Release!)



Inhaltsverzeichnis

1. Was ist 1541-rebuild.....	3
2. Beschreibung der Internen Prozesse.....	3
2.1 Emulation des Schrittmotors.....	3
2.2 Emulation des Laufwerkmotors.....	4
2.3 Schreib-/ Lesekopf (GCR Daten).....	4
3. Spannungsversorgung.....	4
4. Hardware - Aufbau (PCB Rev. 1.4.0).....	5
4.1 Werkzeuge und Hilfsmittel.....	5
4.2 Aufbau der Platine in Bild und Text.....	5
4.3 LCD Display anschließen und Kontrast einstellen.....	7
4.4 Taster für die Bedienung anschließen.....	7
4.5 Drehimpulsgeber als Eingabe nutzen.....	7
4.6 SD Card Modul.....	7
5. Flashen der Firmware (Bsp. unter Linux).....	7
6. 1541-rebuild Bedienungsanleitung.....	8
6.1 SD Karten und Filesystem.....	8
6.2 Info Screen (Startscreen).....	9
6.3 Hauptmenü.....	9
6.3.1 Orientierungshilfe.....	9
6.4 Disk Image Menü.....	9
6.4.1 Insert Image.....	9
6.4.2 Remove Image.....	9
6.4.3 Write Protect.....	9
6.5 Settings.....	10
6.5.1 Debug LED.....	10
6.6 Info.....	10
7. Quellen.....	10

1. Was ist 1541-rebuild

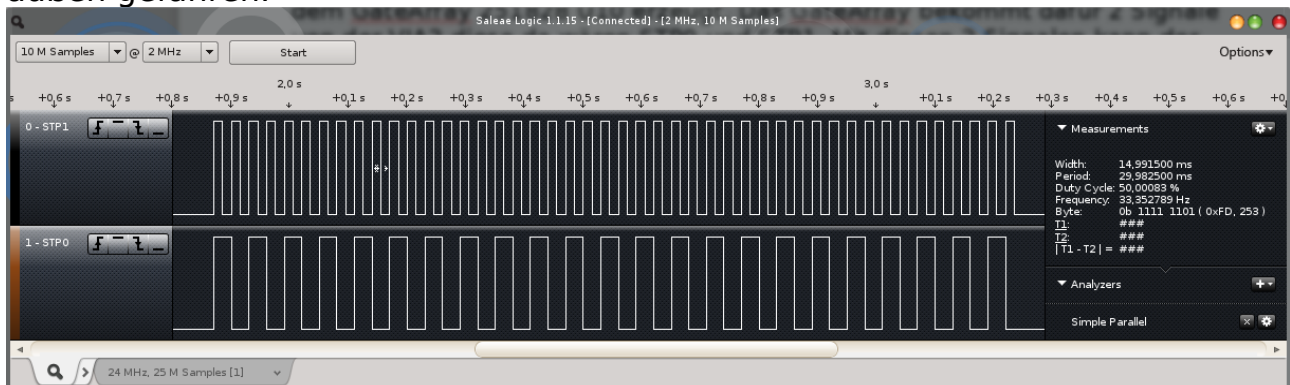
1541-rebuild ersetzt den mechanisch-analogen Teil der Commodore Floppy 1541 und 1541 II durch eine Elektronik. Die Floppy Disk wird in einem Mikrocontroller emuliert. Die Disk Images werden über eine SD Karte geladen. Die Bedienung erfolgt über 3 Taster und ein LCD Display. Als Mikrocontroller kommt ein Atmega1284P zum Einsatz, der mit 24 MHz getaktet wird.

2. Beschreibung der Internen Prozesse

2.1 Emulation des Schrittmotors

Der Schrittmotor ist der Antrieb für den Schreib- Lesekopf der Floppy. Dieser wird auf die zu zugreifende Spur gefahren. Der Schrittmotor hat 4 Anschlüsse welche an die 4 Spulen des Schrittmotors gehen. Durch geeignete Ansteuerung der Spulen kann der Schrittmotor um ein ganz kleinen Winkel, nach links oder rechts gedreht werden. Die vier Steuersignale für den Schrittmotor werden von dem GateArray 251828 U10 erzeugt. Das GateArray bekommt dafür 2 Signale von der VIA2 diese da wären STP0 und STP1. Mit diesen 2 Signalen kann der Lesekopf um genau eine Halbspur vor oder zurück bewegt werden. Diese beiden Signale muss in unserem Fall der µC auswerten um intern die Virtuelle Spur darstellen zu können.

Hier mal die Signale mit einem Logic Analyzer aufgenommen, wenn die Floppy Disk formatiert wird. Der Lesekopf wird dabei an den Anschlag (BUMP) nach außen gefahren.



Die beiden STP Signale sind „verdreht“ am VIA2 angeschlossen. STP1 ist am PB0 und STP0 ist am PB1 angeschlossen. Wenn wir die Signale mal in einer Tabelle aufnehmen sieht man die Wirkungsweise der Signale.

Step	PB1 (STP0)	PB0 (STP1)	Dezimal
0	1	1	3
1	1	0	2
2	0	1	1
3	0	0	0

4	1	1	3
5	1	0	2

Man sieht hier deutlich das der VIA2 nur die beiden PINS „runter zählen“ muss um den Lesekopf nach außen zu bewegen und zwar immer um eine Halbspur. Zählt die VIA2 hoch fährt der Lesekopf entsprechend nach innen. Steht der Lesekopf nun auf eine „voll“ Spur (Bit 0 der aktuellen Halbspur gelöscht) so müssten sofort die neuen Daten von der aktuellen Spur in das RAM geladen werden. Um das aber zu verhindern und dabei Zeit zu sparen, werden erst neue Daten geladen wenn es 20ms keine Signaländerungen an STP0 und STP1 gegeben hat. Je nach Position der Spur muss die Schreib-/ Lesegeschwindigkeit der GCR Daten neu gesetzt werden. Dazu mehr weiter unten im Kapitel 2.3.

Das ist eigentlich auch schon alles was man wissen muss um die Signale auszuwerten zu können und den internen Spurzeiger zu setzen.

2.2 Emulation des Laufwerkmotors

Der Laufwerksmotor wird gesteuert durch die VIA2 PB2. Ist es High läuft der Motor ist er Low ist er aus. Dieses Signal wird benutzt um das senden der GCR Bytes zu steuern. Ist das Signal High werden aus dem Ringpuffer fleißig die GCR Bytes der aktuellen Spur gesendet. Ist das Signal Low werden nur noch 0 Bytes gesendet. (Keine Flusswechsel da keine Drehung !)

2.3 Schreib-/ Lesekopf (GCR Daten)

Gleich erst mal vorweg. Im momentanen Entwicklungszustand unterstützt die 1541-rebuild nur das lesen von einem Disketten Image (d64, g64). Das schreiben und insbesondere das zurückschreiben in einem Disketten Image auf die SD Karte ist noch nicht vollständig implementiert. Es wurde schon auf Machbarkeit geprüft und G64 Dateien konnten beschrieben werden, jedoch ist das ganze in der aktuellen Firmware (1.2.3) sehr unzuverlässig.

Kommen wir nun zum lesen der Daten von einem GCR

3. Spannungsversorgung

Die Spannungsversorgung erfolgt über ein Steckernetzteil mit 5VDC und 1A. Leider ist das Originale Floppy Netzteil etwas schwachbrüstig. Die Spannung brach mit der Zusatzschaltung auf 4.7V ein. Wenn ich den Datenport komplett auf High gesetzt habe, brach sie sogar auf 4.2V ein. Somit konnte der 251828 intern die Transistoren nicht mehr komplett durchschalten und der Effekt war eine glühende GateArray.

Momentan läuft bei mir die Schaltung auch mit dem Originalen Floppy Netzteil!


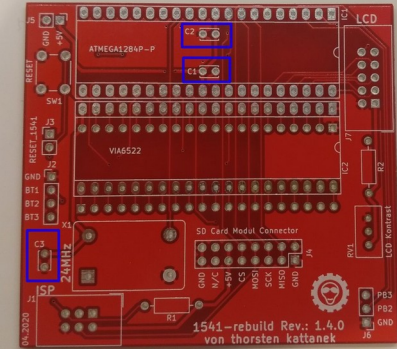

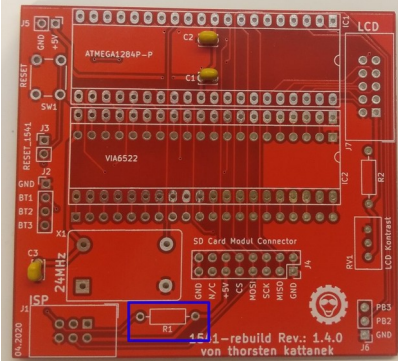

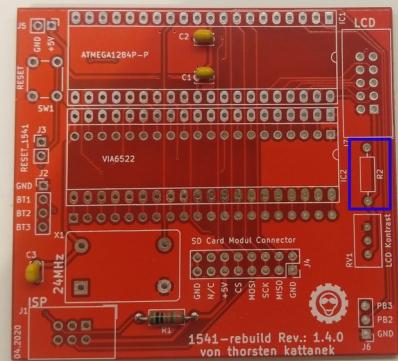

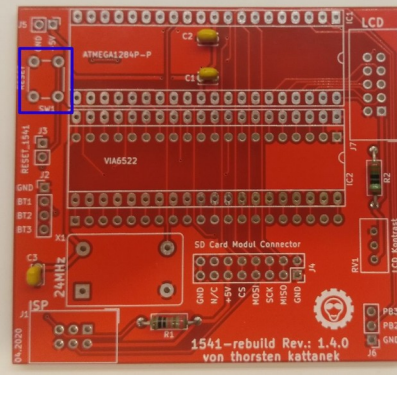
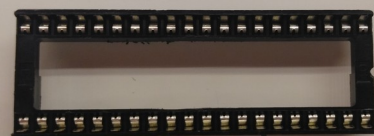
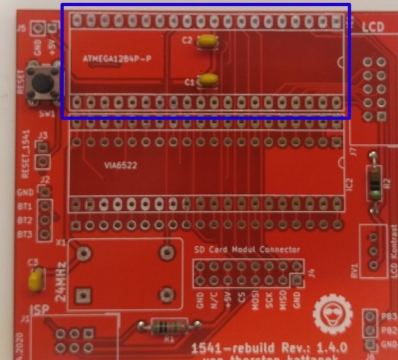
4. Hardware - Aufbau (PCB Rev. 1.4.0)

4.1 Werkzeuge und Hilfsmittel

- LötKolben oder eine Lötstation
- Lötzinn inkl. Flussmittel
- Seitenschneider
- Optional - eine Lupe und eine 3.Hand

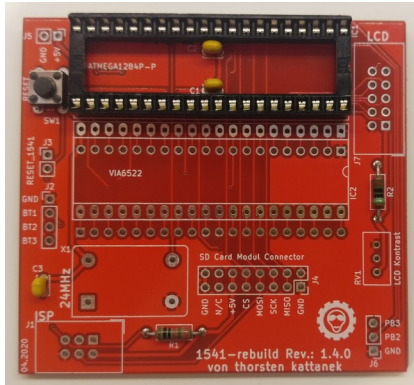
4.2 Aufbau der Platine in Bild und Text

Hier zeige ich euch den Aufbau Schritt für Schritt. Haltet euch an die Reihenfolge der durchnummerierten Bilder, dann solltet ihr das ohne Probleme hinbekommen. Alle Teile werden immer durchgesteckt und von der anderen Seite verlötet. Ich habe bewusst auf SMD Technik verzichtet, um es den Hobbybastler einfacher zu machen.

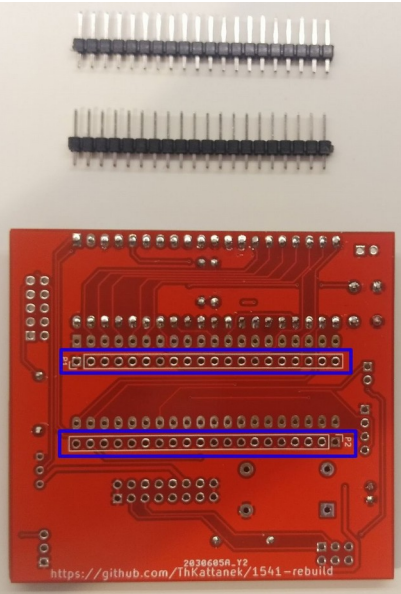
<p>1 3x 100nF Kondensatoren (C1-C3)</p>  	<p>2 10K Ohm Widerstand (R1)</p>  	<p>3 150 Ohm Widerstand (R2)</p>  
<p>4 Kurzhubtaster (SW1) Hinweis: Die Pins sind nicht Quadratisch angeordnet !</p>  	<p>5 (Optional) Bei mir hatte der Sockel genau an der Stelle wo die Kondensatoren sind einen Steg. Den habe ich einfach mit einem Seitenschneider entfernt und mit einer Feile begradigt. Beim Einlöten auf die Kerbe rechts achten !</p> 	<p>6a Sockel 2x20 Pin (IC1)</p> 

6b

So soll es dann aussehen

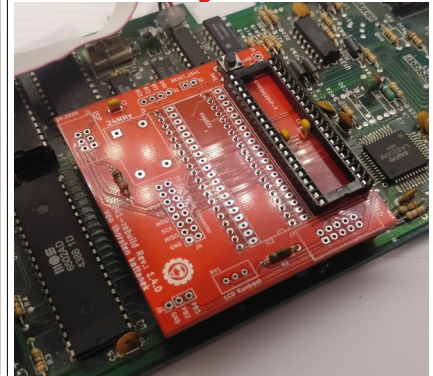
**7a**

2x Stifleiste 20 Pin (P1,P2)

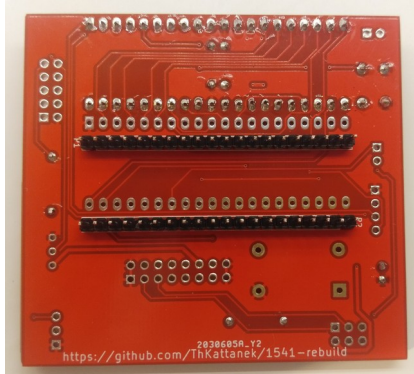
**7b**

Damit die Stiftleisten nachher auch auf den Sockel der VIA passen, habe ich vor dem einlöten das ganze schon mal auf den Sockel gesetzt und dann die Pins fest gelötet.

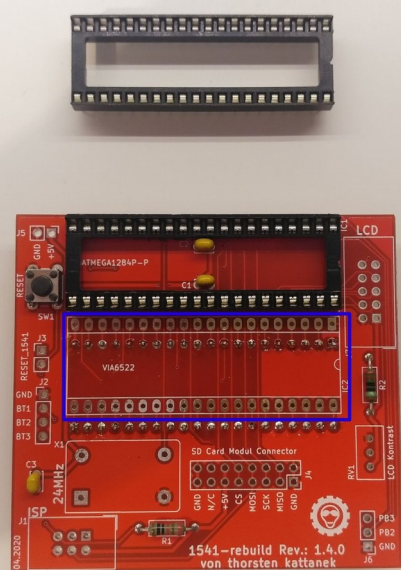
Achtung! Das sind die einzigen Bauteile die von der Rückseite eingelötet werden!

**7c**

So sollte es dann aussehen.

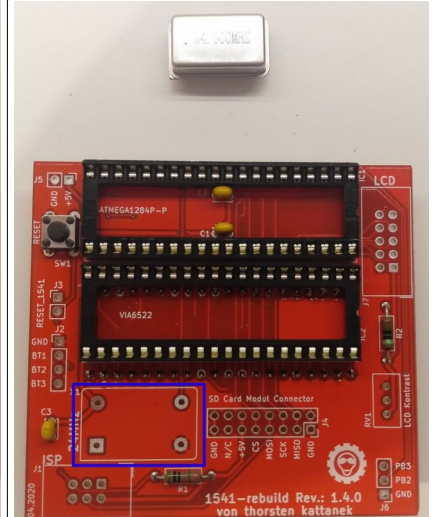
**8**

Sockel 2x20 Pin (IC2)

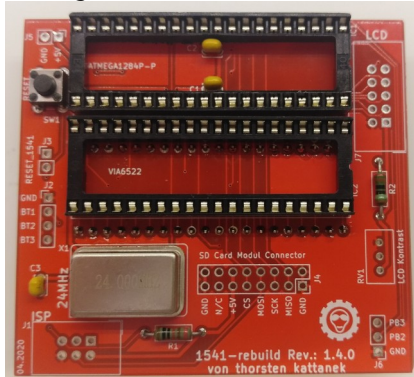
**9a**

24MHz Quarzoszillator (X1)

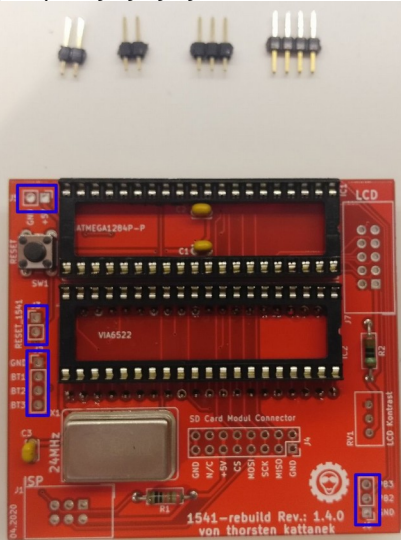
Der Oszillator hat eine Ecke, diese muss links unten sein!

**9b**

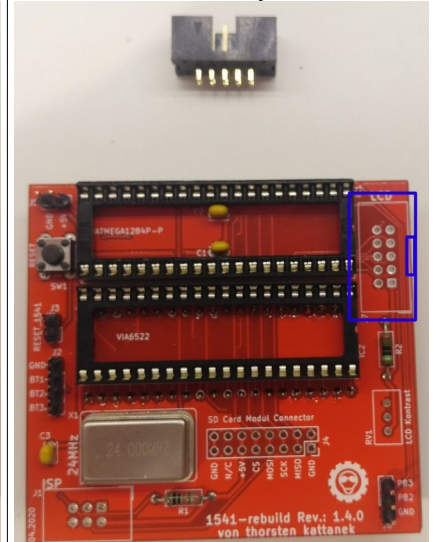
Richtiger Sitz des Oszillators

**10**

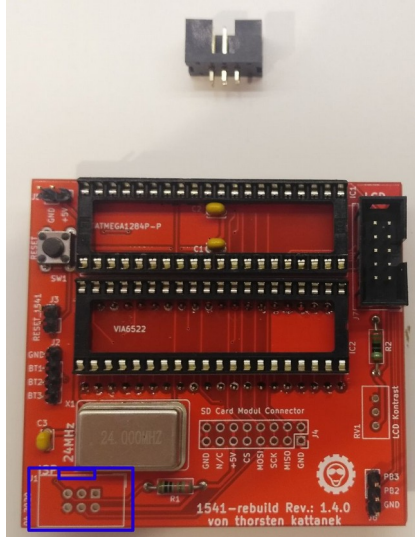
Jumper (J2,J3,J5,J6)

**11**

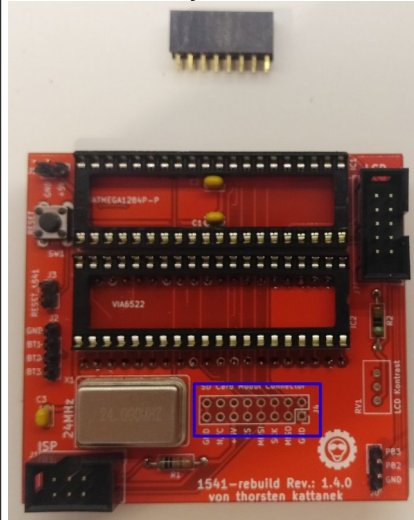
LCD Pinheader 2x5 (J7)



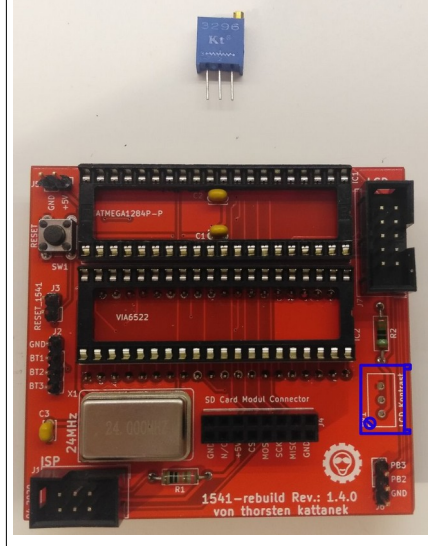
12
ISP PinHeader 2x3 (J1)



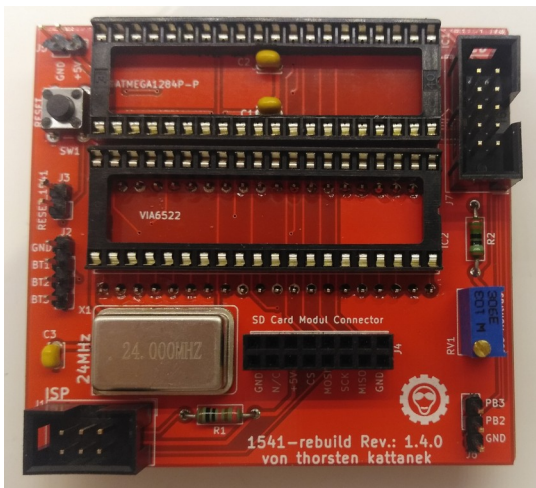
13
SD Card Modul Sockel 2x8
Buchsenleiste (J4)



14
Präzisionspoti 10K Ohm (RV1)



Wenn ihr alles befolgt habt, sollte ihr nun eine Wunderschöne 1541-rebuild vor euch zu liegen haben. In den oberen Sockel kommt der Mikrocontroller (AtMega1284P) und darunter die VIA aus der 1541, wo dann am Ende die 1541-rebuild wieder eingesteckt wird.

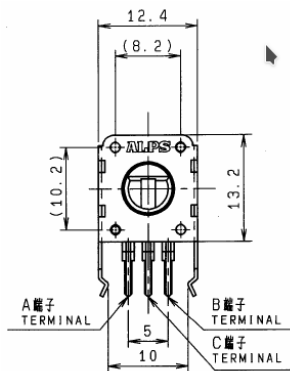


4.3 LCD Display anschließen und Kontrast einstellen

4.4 Taster für die Bedienung anschließen

4.5 Drehimpulsgeber als Eingabe nutzen

Wird ein Drehimpulsgeber für die Eingabe eingesetzt, so wird der Jumper J2 wie folgt belegt. Als Beispiel wird hier ein Drehencoder von ALPS verwendet.



[Drehung im Uhrzeigersinn → Cursor nach unten]

BT1 = Signal A

BT2 = Signal B

BT3 = Taster extra oder im Drehencoder

GNG = GND

4.6 SD Card Modul

5. Flashen der Firmware (Bsp. unter Linux)

Die Fuses des Controllers müssen wie folgt gesetzt werden.

- LFUSE: 0xD0
- HFUSE: 0xD3
- EFUSE: 0xFF

AVRDUDE Argumente:

-U lfuse:w:0xd0:m -U hfuse:w:0xd3:m -U efuse:w:0xff:m

6. 1541-rebuild Bedienungsanleitung

Das rebuild-1541 wird über 3 Buttons gesteuert. Diese befinden sich auf dem Jumper J2. Der BT3 übernimmt dabei 2

BT1 = Hoch / Rechts / +

BT2 = Runter / Links / -

BT3 = Enter - beim loslassen (Zeit vom drücken bis loslassen kleiner 750ms)

BT3 = Zurück / Abbruch - beim loslassen (Zeit vom drücken bis loslassen größer 750ms)

- Also kurzes an tippen des BT3 bedeutet Enter und längeres drücken zurück oder abbrechen. Einfach mal ausprobieren klingt komplizierter als es ist. Daran sollte man sich relativ schnell gewöhnen.

6.1 SD Karten und Filesystem

Es wird die Library von Roland Riegel verwendet, diese sollten folgende SD Karten unterstützen.

- SD, SDHC, miniSD, microSD, microSDHC

Als Filesystem wird FAT16 und FAT32 unterstützt.

- **Hinweis! Es werden Filenamen mit maximal einer Länge von 31 Zeichen unterstützt. (inkl. File Extension)**

Es werden alle Dateien im 1541-rebuild Filebrowser so angezeigt wie sie Physikalisch auf der SD Karte abgelegt sind. Es wird kein sortieren der Dateinamen unterstützt. Möchten Sie aber, das die Dateien auf der SD Karte Alphabetisch geordnet sind, müssen sie das Filesystem der SD Karte auf einem PC sortieren lassen.

Für Linux gibt es das Tool „fatsort“ welches für mich gute Dienste Leistet.

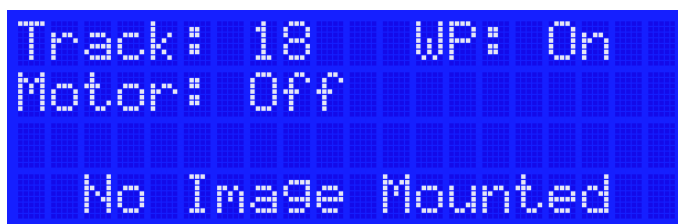
Beispiel Aufruf aus der Konsole raus:

```
sudo fatsort -n /dev/sdb1
```

 (Es wird die Partition 1 des Datenträgers sdb sortiert)

6.2 Info Screen (Startscreen)

Nach dem Einschalten begrüßt einem das 1541-rebuild mit der aktuellen Firmware Versionsnummer und wartet gegebenenfalls auf das einlegen einer SD Karte. Danach befindet man sich im Info Screen.



- Track – Zeigt an, auf welcher Spur sich die 1541-rebuild gerade befindet.
 - WP – Zeigt an ob der Schreibschutz (Write Protect) aktiviert ist oder nicht.
 - „No Image Mounted“ zeigt an das gerade kein Disketten Image eingelegt ist. An dieser Stelle steht sonst der Dateiname des eingelegten Image.
- **Drückt man die Entertaste im Info Screen länger als 3 Sekunden So wird ein Neustart der 1541-rebuild durch geführt!**

6.3 Hauptmenü

In das Hauptmenü gelangt man durch das drücken der Enter Taste im Infoscreen. Oder durch den Menüpunkt „Back“ in einem Untermenü.

6.3.1 Orientierungshilfe

In allen Menüs werden durch kleine Pfeile in den Ecken Rechts/Oben und Rechts/Unten, angezeigt ob es eventuell noch weitere Menüpunkte in der jeweiligen angezeigten Richtung gibt. Da das LCD nur 4 Zeilen besitzt kann nicht immer alles auf einmal Dargestellt werden. Deshalb diese kleine Orientierungshilfe.

6.4 Disk Image Menü

6.4.1 Insert Image

6.4.2 Remove Image

6.4.3 Write Protect

Write Protect wird immer automatisch beim einlegen (Insert Image) oder beim entfernen (Remove Image) eingeschaltet. So wird ein versehentlich ungewolltes schreiben auf dem aktuellen eingelegten Image verhindert. Möchte man auf ein Image (momentan nur bei G64 Images) Schreibzugriff haben, so kann man das hier einstellen. Durch drücken des Enter Buttons, wird zwischen On und Off hin und her geschaltet.

Write Protect On → Schreibschutz aktiviert

Write Protect Off → Schreibzugriff möglich

Bei meinen Tests habe ich raus gefunden das ein Formatieren mittels „Cyberpux Retro Replay“ auch bei aktivierten Schreibschutz funktioniert! Vorsicht Bitte hier!

6.5 Settings

6.5.1 Debug LED

Über diesen Menüpunkt kann eine LED, die an den JUMPER J6 angeschlossen ist, Ein- oder wieder Ausgeschaltet werden. Die Anode wird an PB2 angeschlossen und die Kathode an GND. Es können darüber auch andere Sachen geschaltet werden die 5V benötigen. Jedoch ist zu beachten das ein Port Pin maximal 20mA treiben darf und der gesamte Port nicht mehr als 100mA! (lt. Datenblatt ATmega 1284P)

➤ Hinweis! Debug LED Off → PB2 ist HiZ / On → PB2 Source 5V

Bitte einen Vorwiderstand für die LED mit einberechnen sonst ist sie hin!

6.6 Info

7. Quellen

- Die Floppy 1541 von Karsten Schramm
- Datenblatt Atmega 1284P
- SD Karten Library von Roland Riegel <http://www.roland-riegel.de/sd-reader/>