

1541-rebuild

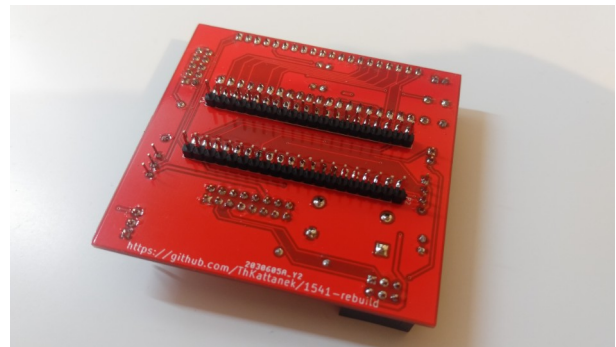
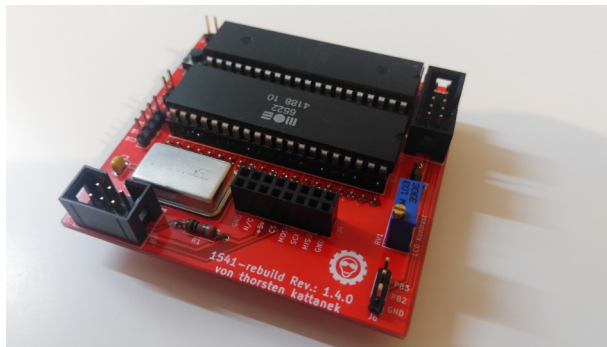
original von Thorsten Kattaneck
erweitert von Fook42
17.12.2021

PCB Rev.: 1.4.2 / Firmware: 1.3.7

Inhaltsverzeichnis

1. Was ist 1541-rebuild.....	2
2. Beschreibung der Internen Prozesse.....	2
2.1. Emulation des Schrittmotors.....	2
2.2. Emulation des Laufwerkmotors.....	3
2.3. Schreib-/ Lesekopf (GCR Daten).....	3
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
4.2.1. PCB version 1.4.2 (Erweiterung Fook42).....	7
4.3. Displays.....	8
4.3.1. LCD Display verbinden und Kontrast einstellen.....	8
4.3.2. I ² C Display nutzen.....	9
4.4. Eingabeelemente anschließen und konfigurieren.....	9
4.4.1. Drei Taster als Eingabe nutzen (Button Mode).....	10
4.4.2. Dreh-Encoder als Eingabe nutzen (Encoder Mode).....	10
4.5. SD Karten Slot Modul.....	11
5. Flashen der Firmware (Bsp. unter Linux).....	12
6. 1541-rebuild Bedienungsanleitung.....	13
6.1. SD Karten und Filesystem.....	13
6.2. Info Screen (Startscreen).....	13
6.3. Hauptmenü.....	14
6.3.1. Orientierungshilfe.....	14
6.4. Disk Image Menü.....	14
6.4.1. Insert Image.....	14
6.4.2. Remove Image.....	14
6.4.3. Write Protect.....	14
6.4.4. New Image.....	15
6.4.5. Save Image.....	15
6.5. Settings.....	15
6.5.1. Pin PB2 und Pin PB3.....	15
6.5.2. Restart.....	15
6.6. Info.....	15
7. Quellen.....	15

1. Was ist 1541-rebuild



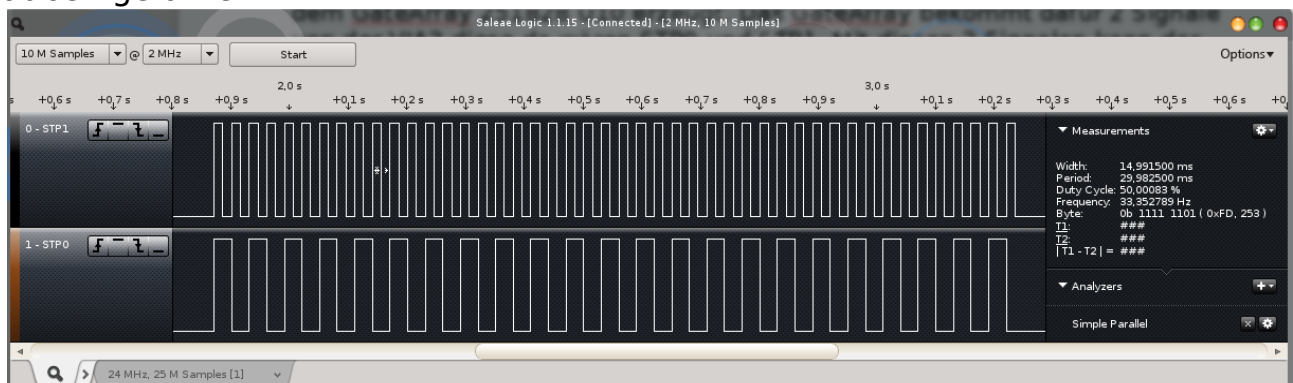
1541-rebuild ersetzt den mechanisch-analogen Teil der Commodore Floppy 1541 und 1541 II durch eine Elektronik. Die Floppy Disk wird in einem Mikrocontroller emuliert. Die Disk Images werden über eine SD Karte geladen. Die Bedienung erfolgt über 3 Taster und ein LCD Display. Als Mikrocontroller kommt 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 einen 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 μ C 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, dass 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 einer „vollen“ 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)

Lesen und Schreiben von Floppydaten wird in der 1541 mittels GCR-codierten Daten realisiert. Dieses Format der Datenspeicherung erzeugt einen Datenstrom in welchem lange Abfolgen von gleichen Bits (0 oder 1) nicht vorkommen können. Nachteilig ist, dass dazu für herkömmliche Bitmuster aus 4 Bits nun jeweils ein 5 Bit breites Wort abgespeichert werden muss.

Weiterhin ist hier zu beachten, dass, je nach Spur, eine leicht andere Übertragungsgeschwindigkeit der Bit-daten berücksichtigt werden muss. Hierfür lassen sich 4 Zonen identifizieren (Spuren 1-17, 18-24, 25-30, 31-40) welche

jeweils einen etwas anderen Bittakt nutzen. Bei einem echten Laufwerk hängt dieser Takt vom kleineren Radius und damit geringerer Geschwindigkeit der inneren Spuren ab. Zudem lassen sich auf den äußeren Spuren auch mehr Sektoren abspeichern – auch das ist im 1541-rebuild berücksichtigt und wird entsprechend emuliert.

Die 1541-rebuild unterstützt nun 2 verschiedene Disketten-Abbildformate: D64 und G64.

Das G64-Format entspricht dabei einem 1:1 Abbild des GCR Datenstroms wie der auf einer echten Diskette stehen würde – inklusive Synchronisations-Mustern und Prüfsummen für einzelne Blöcke. Lesen und Schreiben in diesem Format ist trivial, da die Daten direkt an/vom Floppy-Controller zur SD-Karte durchgereicht werden können.

Beim D64-Format sind die Lese- & Schreib-Routinen komplexer, da bei jedem Zugriff eine GCR-Kodierung und Dekodierung stattfinden muss um dem Floppy-Controller vorzumachen, dass es sich um einen Datenstrom einer „echten“ Floppydisk handelt. Tests mit verschiedenen Programmen und Routinen dazu waren erfolgreich, jedoch kann nicht ausgeschlossen werden, dass diese zeitaufwändigen Algorithmen mit bestimmten Schnell-Lade-Routinen einiger C64-Programme (Fastloadern) zu Problemen führen.

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!


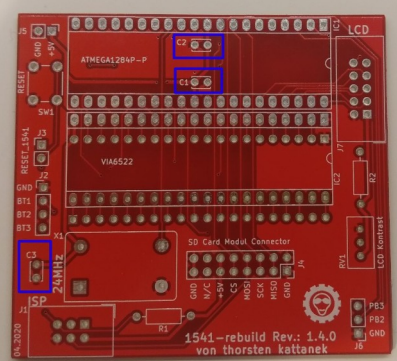

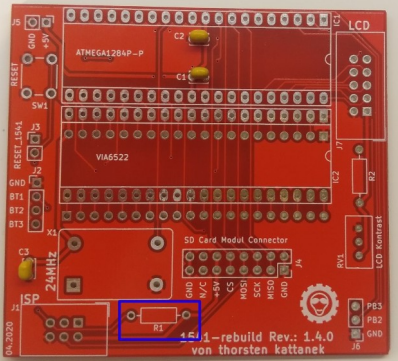

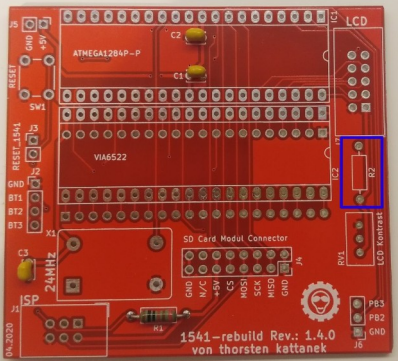

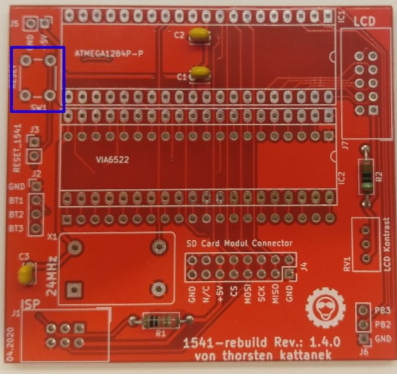
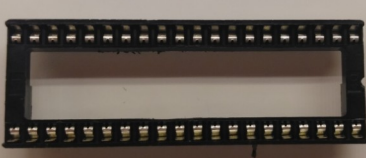

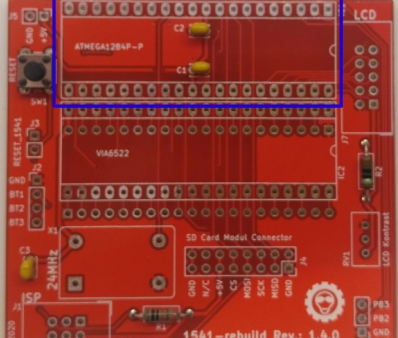
4. Hardware - Aufbau (PCB Rev. 1.4.0)

4.1. Werkzeuge und Hilfsmittel

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

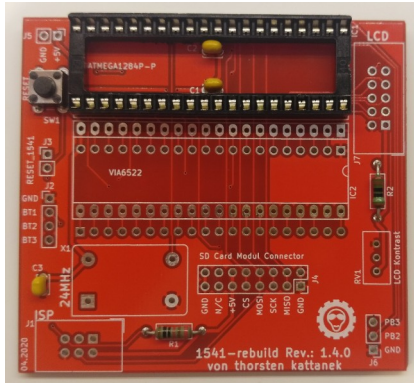
4.2. Aufbau der Platine in Bild und Text

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

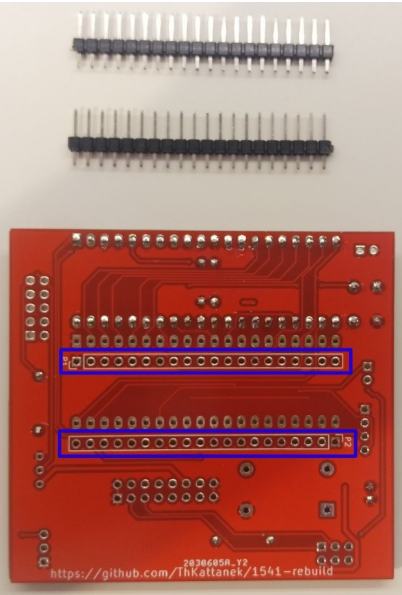
<p>1 3x 100nF Kondensatoren (C1-C3)</p>  	<p>2 10K Ohm Widerstand (R1)</p>  	<p>3 150 Ohm Widerstand (R2)</p>  
<p>4 Kurzhubtaster (SW1) Hinweis: Die Pins sind nicht Quadratisch angeordnet !</p>  	<p>5 (Optional) Bei mir hatte der Sockel genau an der Stelle wo die Kondensatoren sind einen Steg. Den habe 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

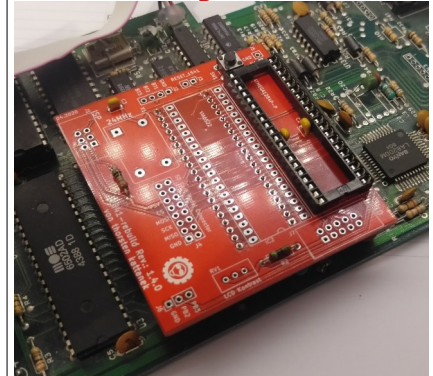
**7a**

2x Stifleiste 20 Pin (P1,P2)

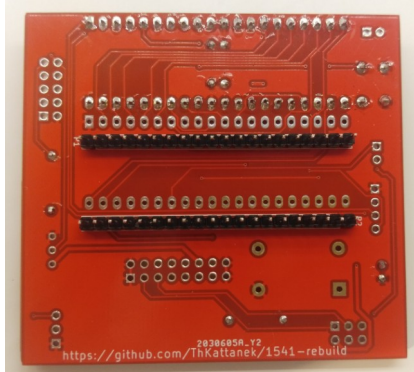
**7b**

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.

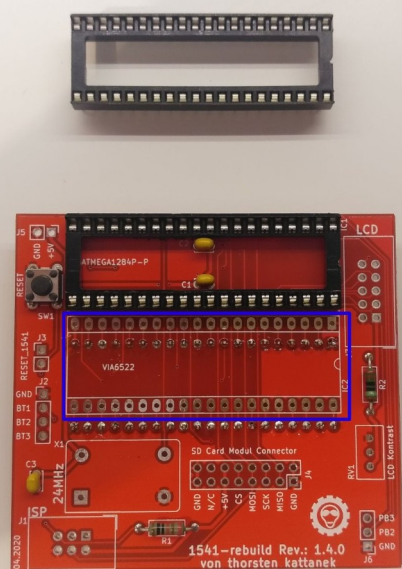
Achtung! Das sind die einzigen Bauteile die von der Rückseite eingelötet werden!

**7c**

So sollte es dann aussehen.

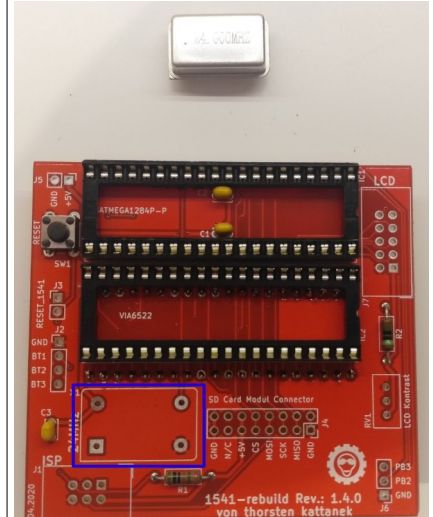
**8**

Sockel 2x20 Pin (IC2)

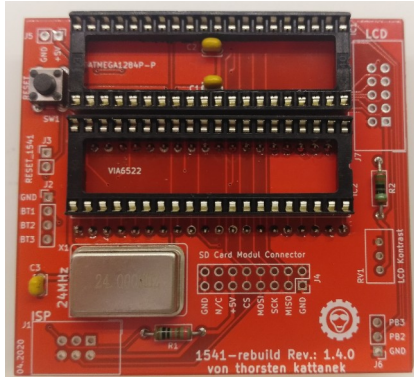
**9a**

24MHz Quarzoszillator (X1)

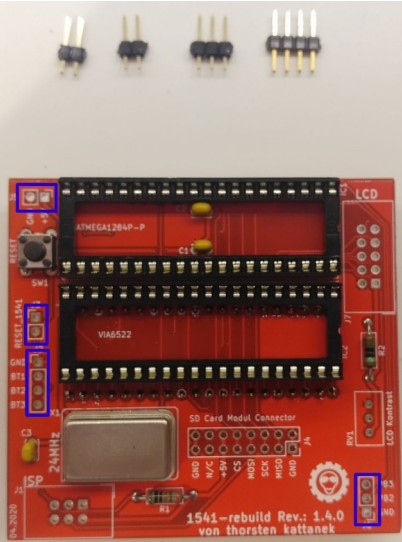
Der Oszillator hat eine Ecke, diese muss links unten sein!

**9b**

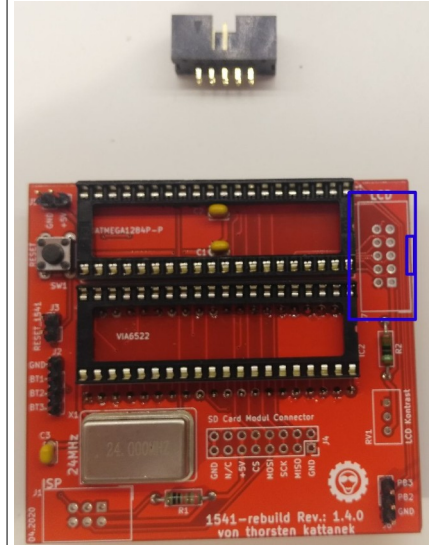
Richtiger Sitz des Oszillators

**10**

Jumper (J2,J3,J5,J6)

**11**

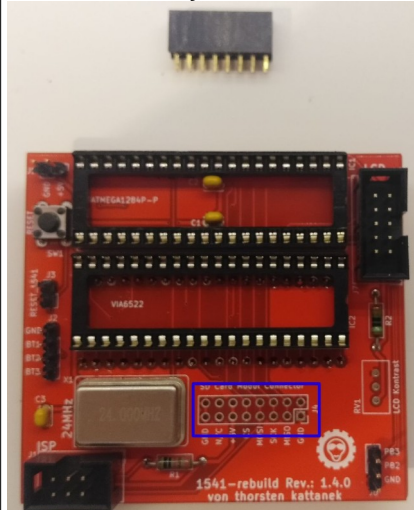
LCD Pinheader 2x5 (J7)



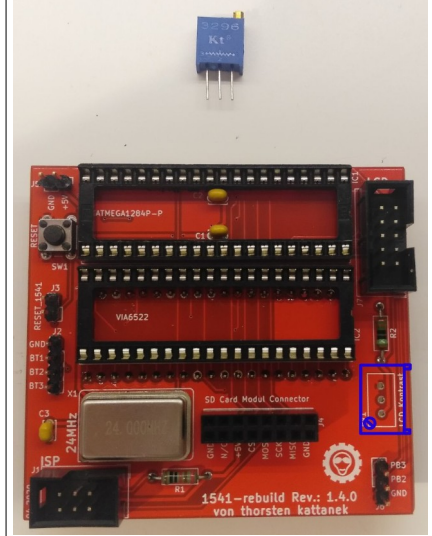
12
ISP PinHeader 2x3 (J1)



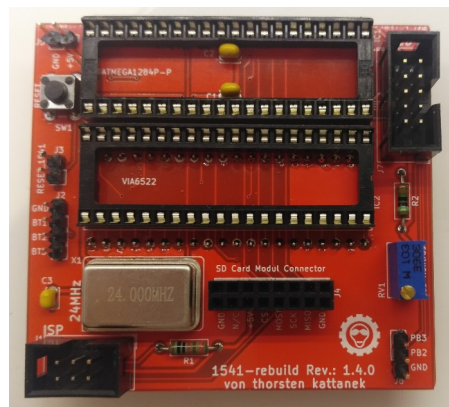
13
SD Card Modul Sockel 2x8
Buchsenleiste (J4)



14
Präzisionspoti 10K Ohm (RV1)



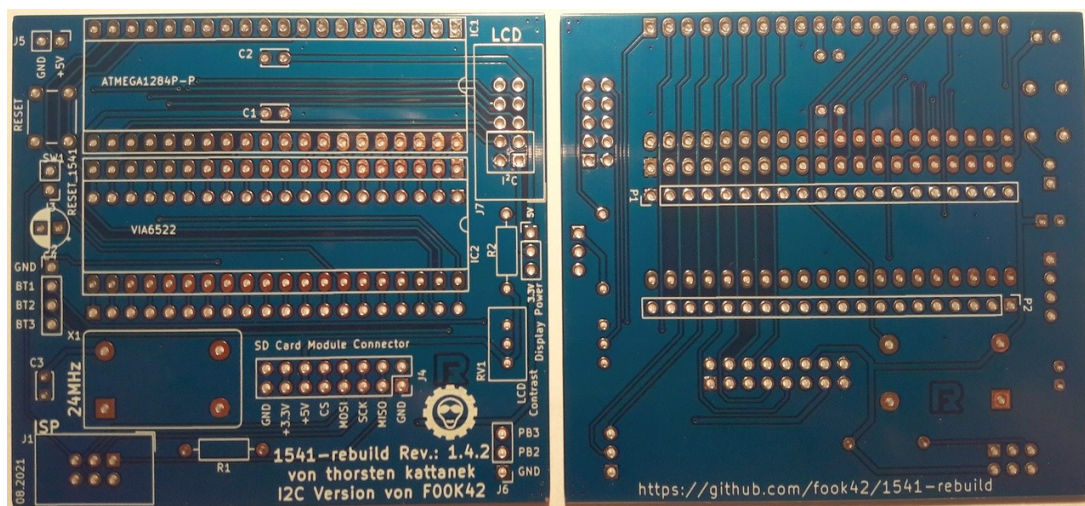
Wenn ihr alles befolgt habt, sollte ihr nun eine wunderschöne 1541-rebuild vor euch liegen haben. In den oberen Sockel kommt der Mikrocontroller (AtMega1284P) und darunter die VIA aus der 1541, wo dann am Ende die 1541-rebuild wieder eingesteckt wird.



4.2.1. PCB version 1.4.2 (Erweiterung Fook42)

Für diese Board Revision ist ein komplett neu geroutetes Layout entstanden, welches den Einsatz anderer Display-Typen (per I²C-Schnittstelle) unterstützt.

Zusätzlich wurden ein Stützkondensator (C4) mit 47µF/16V sowie ein Jumper für die Selektion der Versorgungsspannung des Display-Anschlusses integriert.



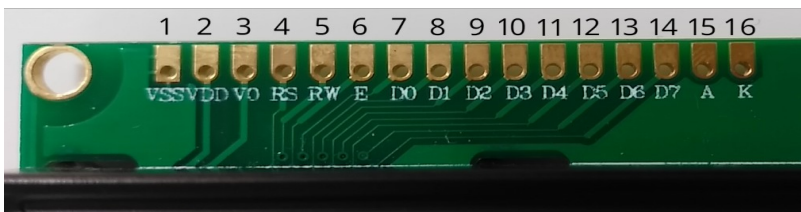
4.3. Displays

Die 1541-rebuild ermöglicht den Anschluss eines Displays zur Selektion des Diskettenimages sowie zur Status-Anzeige und anderer nützlicher Informationen (z.B. zur SD-Karte).

Mit PCB version 1.4.0 können nur LCDs (→ 4.3.1) über ein herkömmliches, 10 poliges Interface angesteuert werden. Ab PCB version 1.4.2 werden zudem I²C-Displays (→ 4.3.2) unterstützt.

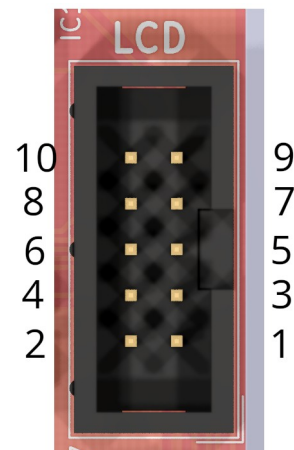
4.3.1. LCD Display verbinden und Kontrast einstellen

Das LCD Display wird über den Port J7 angeschlossen. Das ganze kann über ein Flachband Kabel mit 10 Leitungen und einer 10-Poligen Pfostenbuchse erfolgen. Als LCD können 16x2 als auch 20x4 Displays verwendet werden.



LCD Anschluss

Wannenstecker auf 1541-rebuild	LCD Display
Pin 1	Pin 2 (+5V)
Pin 2	Pin 1, 5, 16 (GND)
Pin 3	Pin 11 (D4)
Pin 4	Pin 12 (D5)
Pin 5	Pin 13 (D6)
Pin 6	Pin 14 (D7)
Pin 7	Pin 6 (E)
Pin 8	Pin 4 (RS)
Pin 9	Pin 15 (LED Anode)
Pin 10	Pin 3 (VO)



Wannenstecker auf 1541-rebuild

Kontrast einstellen:

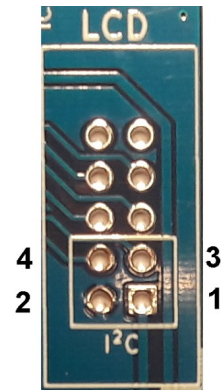
Wichtig ist, dass nach dem ersten Start der 1541-rebuild der Kontrast des LCD Displays eingestellt wird! Dafür muss auch der Mikrocontroller programmiert sein. Am besten man dreht das Poti RV1 in einer Richtung bis zum Ende (welche Richtung ist egal). Sollte bis dahin nichts auf dem LCD zu sehen sein, das ganze dann in andere Richtung drehen. Dann sollte auf jeden Fall irgendwann etwas auf dem LCD erscheinen.

Wenn nichts zu sehen ist, liegt es evtl. an einem falsch programmierten Mikrocontroller oder an einem falsch angeschlossenen LCD Display. Die Fuse Bits des Mikrocontrollers könnten auch falsch gesetzt sein. Bitte alle Fehlerquellen untersuchen und gegebenenfalls korrigieren.

4.3.2. I²C Display nutzen

Für den Anschluss eines Displays mit I²C-Interface (bei Atmel auch als „TWI“= Two-Wire-Interface bezeichnet) werden nur 4 Leitungen benötigt (VCC, GND, SDA und SCL). Diese sind ab PCB version 1.4.2 ebenfalls über den Display-Anschluss verfügbar und werden nur durch eine automatische Erkennungs-routine beim Start des 1541-rebuild aktiviert wenn dort ein I²C-Display angeschlossen ist, andernfalls wird der „normale“ LCD-Anschluss genutzt wie in 4.3.1 beschrieben.

LCD Anschluss auf 1541-rebuild v. 1.4.2	I ² C Display
Pin 1	+5V / +3.3V *
Pin 2	GND
Pin 3	SCL (Serial CLock)
Pin 4	SDA (Serial DAta)



*Display-Anschluss
der 1541-rebuild
version 1.4.2*

* die Auswahl der richtigen Versorgungsspannung kann über den neuen Jumper „Display Power“ erfolgen – hierbei sollte entsprechend der Anforderungen des Displays die richtige Spannung selektiert werden! Die 3.3V Spannung wird vom SD-Karten-Modul erzeugt.

4.4. Eingabeelemente anschließen und konfigurieren

Es ist möglich mit der aktuellen Firmware (ab 1.3.0) zwischen 2 verschiedenen Eingabe Modi zu wechseln:

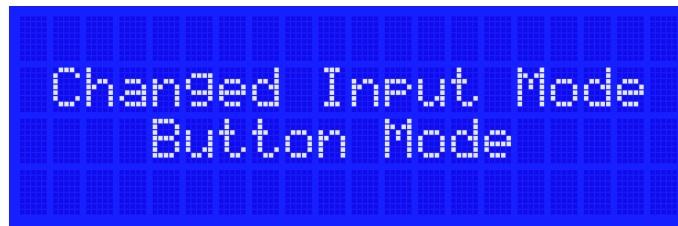
- **Button Mode:** Es werden alle Eingaben über 3 Taster realisiert
- **Encoder Mode:** Es werden alle Eingaben über einen Dreh-Encoder und einen Taster realisiert

Beide Modi benötigen alle Pins der Stiftleiste J2.

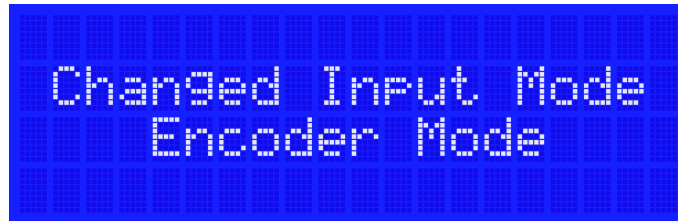
Das Wechseln zwischen den beiden Modi wird wie folgt vorgenommen:

Es muss während des Resets der Taster BT3 gedrückt gehalten werden. Also den Taster BT3 drücken und halten und dann die Reset Taste drücken. Auf dem Display erscheint eine Meldung für 3 Sekunden welcher Modus jetzt aktiviert wurde. Diese Einstellung wird im EEPROM des Mikrocontrollers gespeichert.

LCD Ausgabe → Es wurde der „Button Mode“ aktiviert



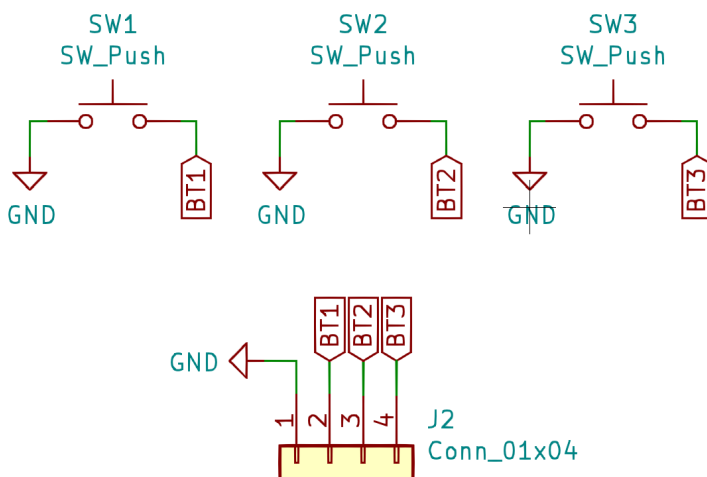
LCD Ausgabe → Es wurde der „Encoder Mode“ aktiviert



➤ **Die Grund Einstellung ist: Button Mode**

4.4.1. Drei Taster als Eingabe nutzen (Button Mode)

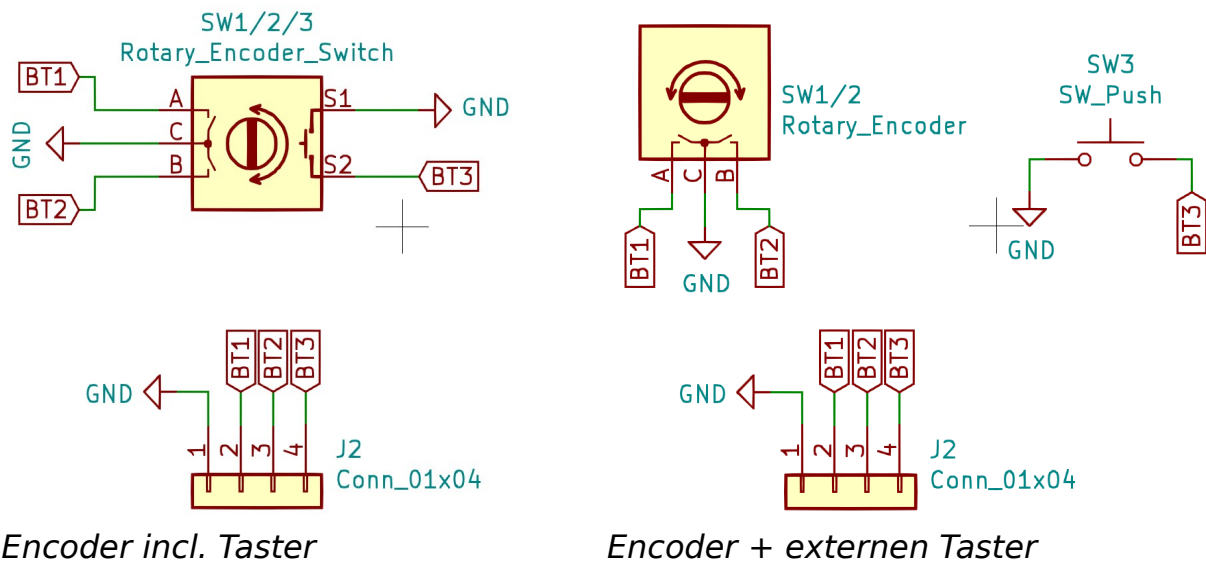
Zur Eingabe werden 3 Taster benutzt. Diese werden alle wie folgt angeschlossen.



➤ **Wichtig! Es muss der „Button Mode“ eingestellt werden.
Siehe Kapitel 4.4**

4.4.2. Dreh-Encoder als Eingabe nutzen (Encoder Mode)

Wird ein Dreh-Encoder für die Eingabe eingesetzt, so wird das ganze wie folgt angeschlossen. Ich empfehle einen Encoder mit mind. 24 Rastungen)



[Drehung im Uhrzeigersinn → Cursor nach **oben**]

[Drehung im entgegen des Uhrzeigersinn → Cursor nach **unten**]

BT1 = Signal A

BT2 = Signal B

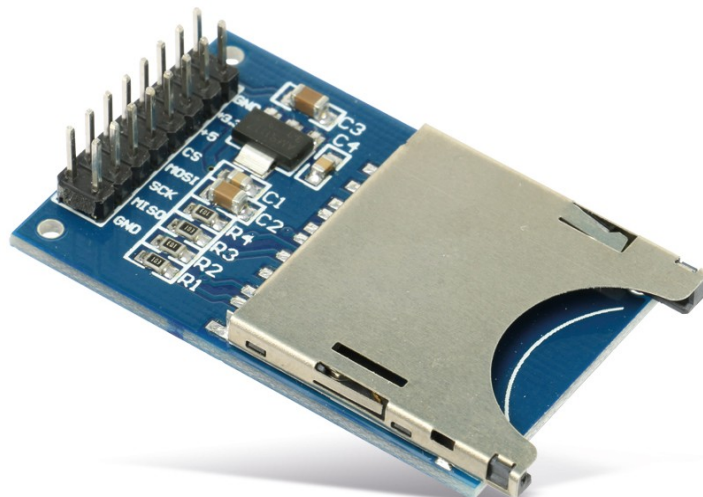
- **Wichtig! Es muss der „Encoder Mode“ eingestellt werden. Siehe Kapitel 4.4**

4.5. SD Karten Slot Modul

Als SD Karten Slot kommt ein fertiges Modul zum Einsatz, welches einfach auf den Sockel J4 aufgesteckt wird oder mittels 6 Verbindungskabel verbunden werden kann um den SD Slot an beliebiger Stelle anbringen zu können.

Eine Bezugsquelle ist z.B. Pollin. Diese Teile sollten aber alle bei den üblichen Verdächtigen zu finden sein (Ebay und co.)

<https://www.pollin.de/p/sd-speicherkartenmodul-810359>



5. Flashen der Firmware (Bsp. unter Linux)

Ich beschreibe hier lediglich die Programmierung des AVR Mikrocontrollers mittels dem Konsolen Tool **avrdude**. Dieses gibt es für verschiedene Betriebssysteme. Ich benutze als Programmiergerät den *Diamex All-AVR Programmer*.

<https://www.diamex.de/dxshop/Diamex-AVR-Prog-Programmer-fuer-ISP-PDI-TPI>

Der Diamex ist „AVRISP mkII“ kompatibel und wird auch so als Parameter an **avrdude** (-c avrisp2) übergeben. Wenn ihr einen anderen Programmer habt müsst ihr in dessen Bedienungsanleitung schauen.

So kann der Atmega1284P geflasht werden:

```
avrdude -u -c avrisp2 -p m1284p -U flash:w:"1541-rebuild.hex":a
```

Die Fuses des Controllers müssen wie folgt gesetzt werden:

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

```
avrdude -u -c avrisp2 -p m1284p -U lfuse:w:0xD0:m -U hfuse:w:0xD3:m -U efuse:w:0xFF:m
```

Eine GUI für **avrdude** findet ihr hier:

<https://blog.zakkemble.net/avrdudess-a-gui-for-avrdude/>

6. 1541-rebuild Bedienungsanleitung

Das 1541-rebuild kann über 3 Buttons oder einen Dreh-Encoder + Button gesteuert werden. Die Anschlüsse dafür befinden sich auf der Stiftleiste J2.

- **Wie die Modi eingestellt werden, kannst du im Kapitel 4.4 nachlesen**

Button Mode:

- BT1 = Hoch / Rechts / +
- BT2 = Runter / Links / -

Encoder Mode:

- Drehen in Uhrzeigersinn = Hoch / Rechts / +
- Drehen entgegen Uhrzeigersinn = Runter / Links / -

Der BT3 übernimmt 2 Funktionen:

- *Enter* - (Zeit vom Drücken bis Loslassen < 750ms)
- *Zurück / Abbruch* - (Zeit vom Drücken bis Loslassen > 750ms)

Also kurzes Antippen des BT3 bedeutet *Enter/Auswählen* und längeres Drücken *Zurück zum vorherigen Menü/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 unterstützt folgende SD-Karten-Formate:

- SD, SDHC, miniSD, microSD, microSDHC

Als Filesystem kann sowohl FAT16 als auch FAT32 genutzt werden.

- **Hinweis! Nur Filenamen mit einer maximalen Länge von 31 Zeichen werden unterstützt (inkl. File Extension). Längere Dateinamen führen zu Fehlern.**

Es werden alle Dateien im 1541-rebuild Filebrowser so angezeigt wie sie physikalisch auf der SD-Karte abgelegt sind. Die Dateinamen werden nicht automatisch sortiert. Möchten Sie aber, dass die Dateien 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.

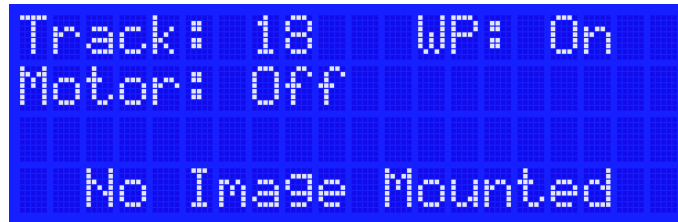
Beispiel Aufruf aus der Konsole raus:

```
sudo fatsort -n /dev/sdb1 (Es wird die 1.Partition des Datenträgers sdb sortiert)
```

6.2. Info Screen (Startscreen)

Nach dem Einschalten begrüßt einen 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: 18    WP: On
Motor: Off
No Image Mounted
```

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, dass gerade kein Disketten Image eingelegt ist. An dieser Stelle steht sonst der Dateiname des eingelegten Images.

- **Drückt man die Entertaste im Info Screen länger als 3 Sekunden
So wird ein Neustart (Reset) 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 Displays nur eine gewisse Anzahl an Zeilen (z.B. 2 oder 4) besitzen, kann nicht immer alles auf einmal dargestellt werden. Deshalb diese kleine Orientierungshilfe.

6.4. Disk Image Menü

6.4.1. Insert Image

Hier gelangt man in den Filebrowser um ein Disk Image von der SD Karte auszuwählen. Unterstützt werden aktuell nur Dateien mit der Endung *d64* und *g64*. Bei allen anderen erscheint eine Fehlermeldung falls diese ausgewählt werden. Anderenfalls wird das Image geöffnet und man gelangt sofort wieder in den Info Screen.

6.4.2. Remove Image

Ein eingelegtes Disk Image wird ausgeworfen und das System springt automatisch wieder zum Info Screen.

6.4.3. Write Protect

Der Schreibschutz eines Images (*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 Schreibzugriff haben, so kann das hier eingestellt werden. Durch Drücken des Enter Buttons, wird zwischen *On* und *Off* hin- und her- geschaltet.

Write Protect: On → Schreibschutz aktiviert (Standard Einstellung)

Write Protect: Off → Schreibzugriff möglich

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

6.4.4. New Image

Noch nicht implementiert!

6.4.5. Save Image

Noch nicht implementiert!

6.5. Settings

6.5.1. Pin PB2 und Pin PB3

Es können über diese Pins, externe Schaltungen mit 5V geschaltet werden. 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!

- Pin PB2/3 **Off** → PB2/3 ist Hi-Z
- Pin PB2/3 **On** → PB2/3 Source 5V

6.5.2. Restart

Hier wird das 1541-rebuild System neu gestartet (Reset).

6.6. Info

In diesem Menü können die aktuelle Systemversion des 1541-rebuild aber auch Informationen zur eingelegten SD-Karte (Hersteller, Größe, Formatierung) angezeigt werden.

7. Quellen

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