

MFM harddisk Reader/Emulator

Vorwort

Bei dieser Beschreibung handelt es sich um das Sichern von Daten aus alten Festplatten, welche eine ST506 Schnittstelle haben und mit MFM formatiert wurden. Mit der im folgenden beschriebenen Hardware können von besagten Festplatten Komplettabzüge (Images) erstellt und dann ebenfalls mit der beschriebenen Hardware und Umstecken von Kabel / Jumper die vorher eingelesene Festplatte für den Hostcomputer emuliert werden.

Ich möchte hier ausdrücklich meiner Begeisterung für die Entwicklung von David Gesswein kundtun.

Auf <http://www.pdp8online.com/mfm/> hat David den von ihm entwickelten MFM Hard Disk Reader/Emulator beschrieben und davon eine Miniserie für Interessenten produziert.

Der MFM Hard Disk Reader/Emulator besteht aus einer Platine zum Anschluss der Festplatte und wird als Adapterboard auf ein Beagle Board gesteckt. Das Beagle Board hat 2x PRU 32-bit Microcontroller (PRU= programmable real-time unit = programmierbare Echtzeit-Einheit) und 2x 46 Pin Buchsenleisten für stapelbare Erweiterungen.

Die von David geschriebene Software nutzt die im BBB vorhandenen Microcontroller (PRU), Betriebssystem ist ein DEBIAN Kernel 3.8.

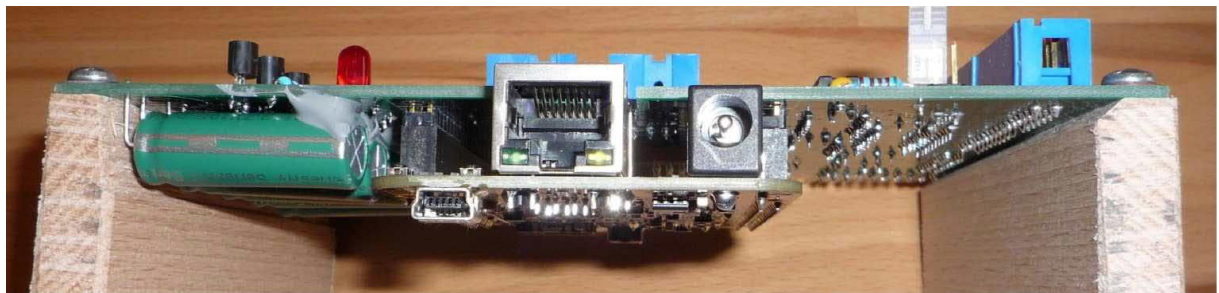
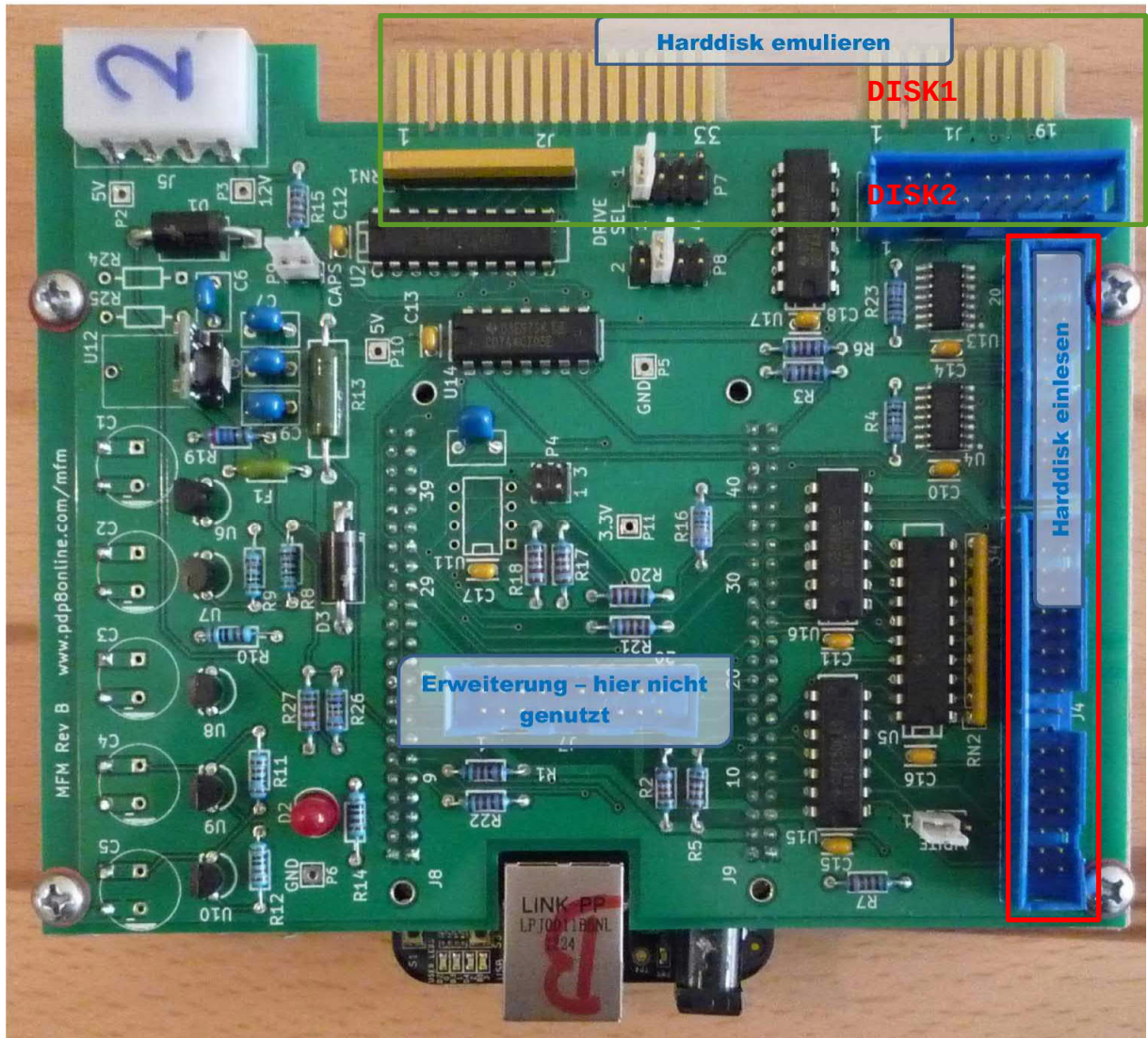
Hinweis: Verwenden Sie keinen Kernel neuer als den 3.8-Kernel. Neuere Kernels unterstützen nicht die Methode, mit der David die BBB-Pins konfiguriert.

Das Image von Davids Installation beinhaltet die Betriebssystemumgebung ohne grafische Oberfläche, ich habe für mich als GUI XFCE installiert.

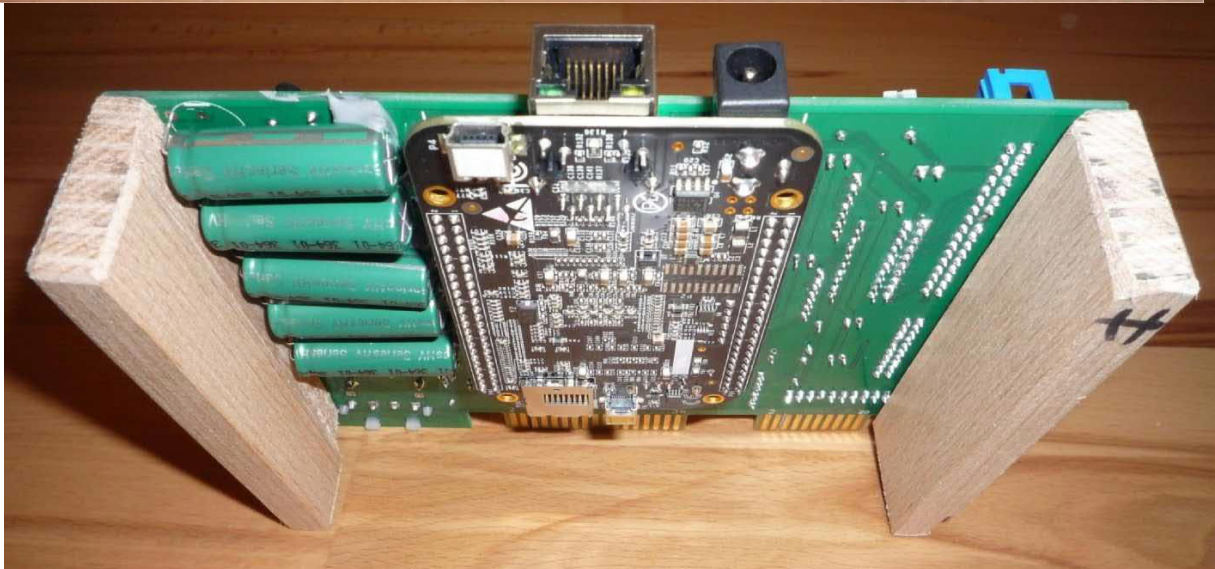
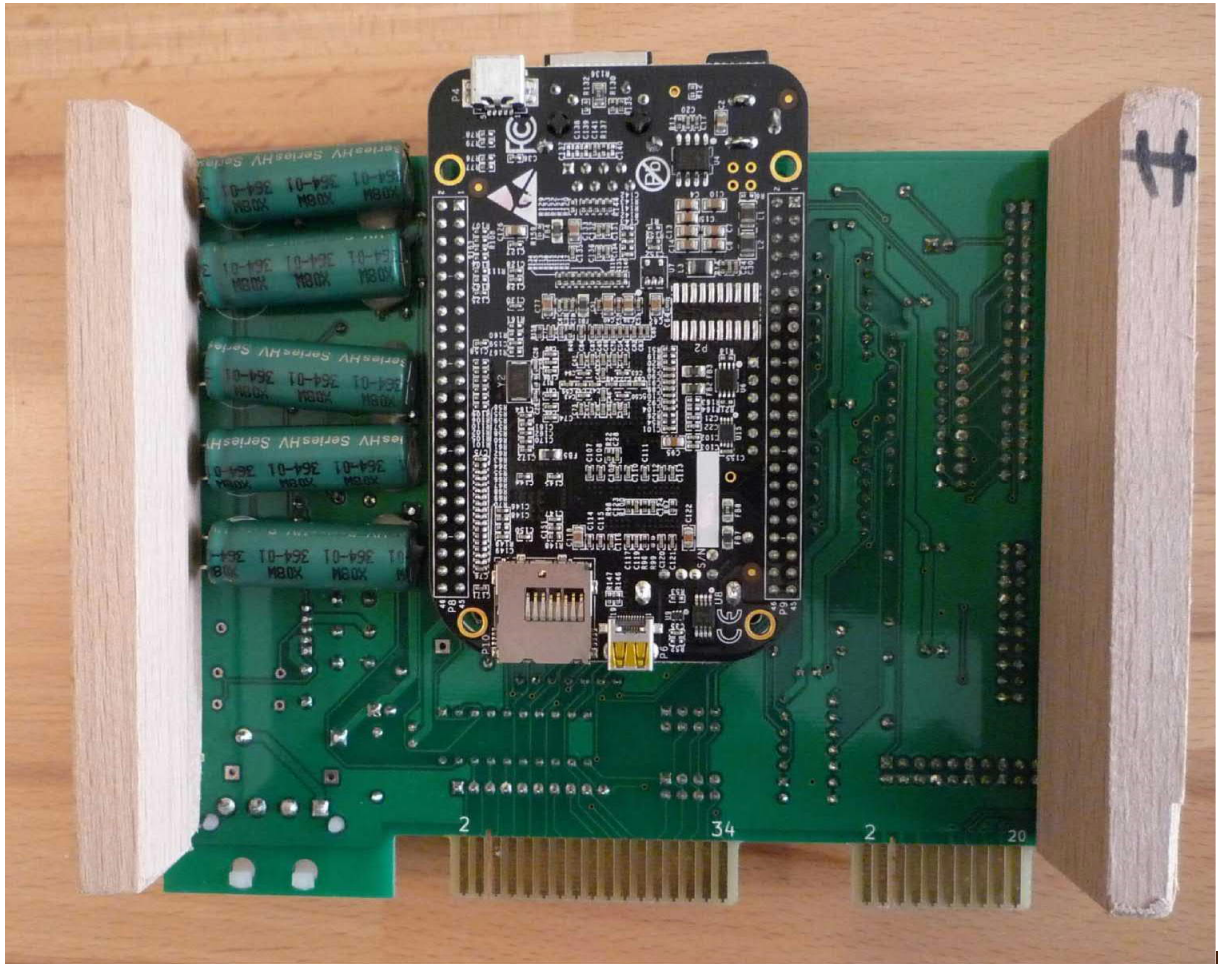
5 großen Kondensatoren dienen als unterbrechungsfreie Stromversorgung zum sauberen Herunterfahren des Betriebssystems, wenn die Stromversorgung abgeschaltet wird. Die Emulation startet wieder automatisch (**Powerfail**) beim Einschalten der Versorgungsspannung, womit von der Emulation wie bei einer ersetzten Festplatte, das Hostbetriebssystem gebootet werden kann.

MFM harddisk Reader/Emulator

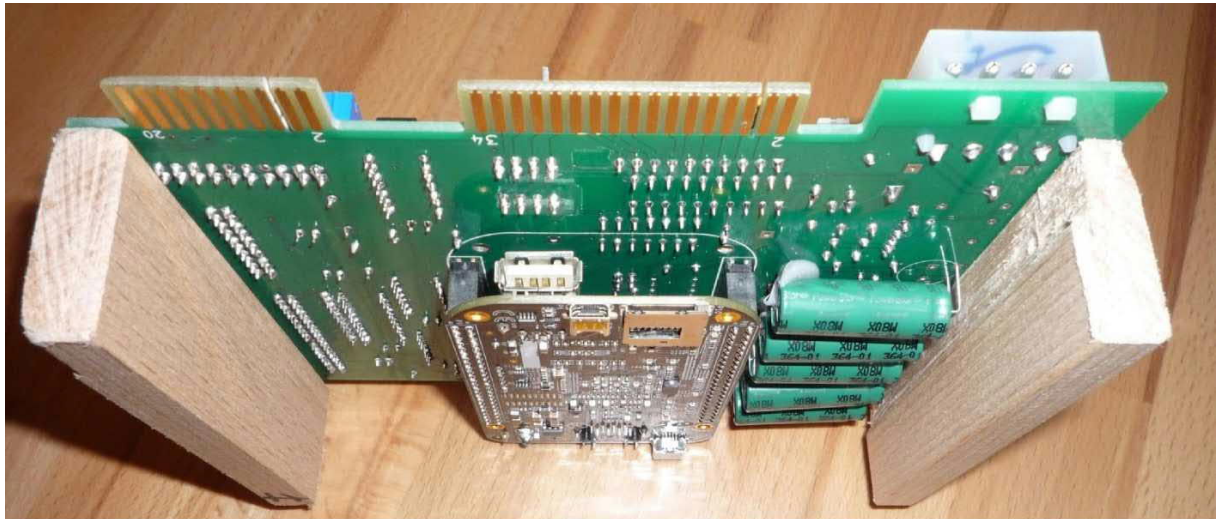
Bilder:



MFM harddisk Reader/Emulator



MFM harddisk Reader/Emulator



Support/FAQ: http://elinux.org/Beagleboard:BeagleBoneBlack_Debian

Unterschied Revision B und C:

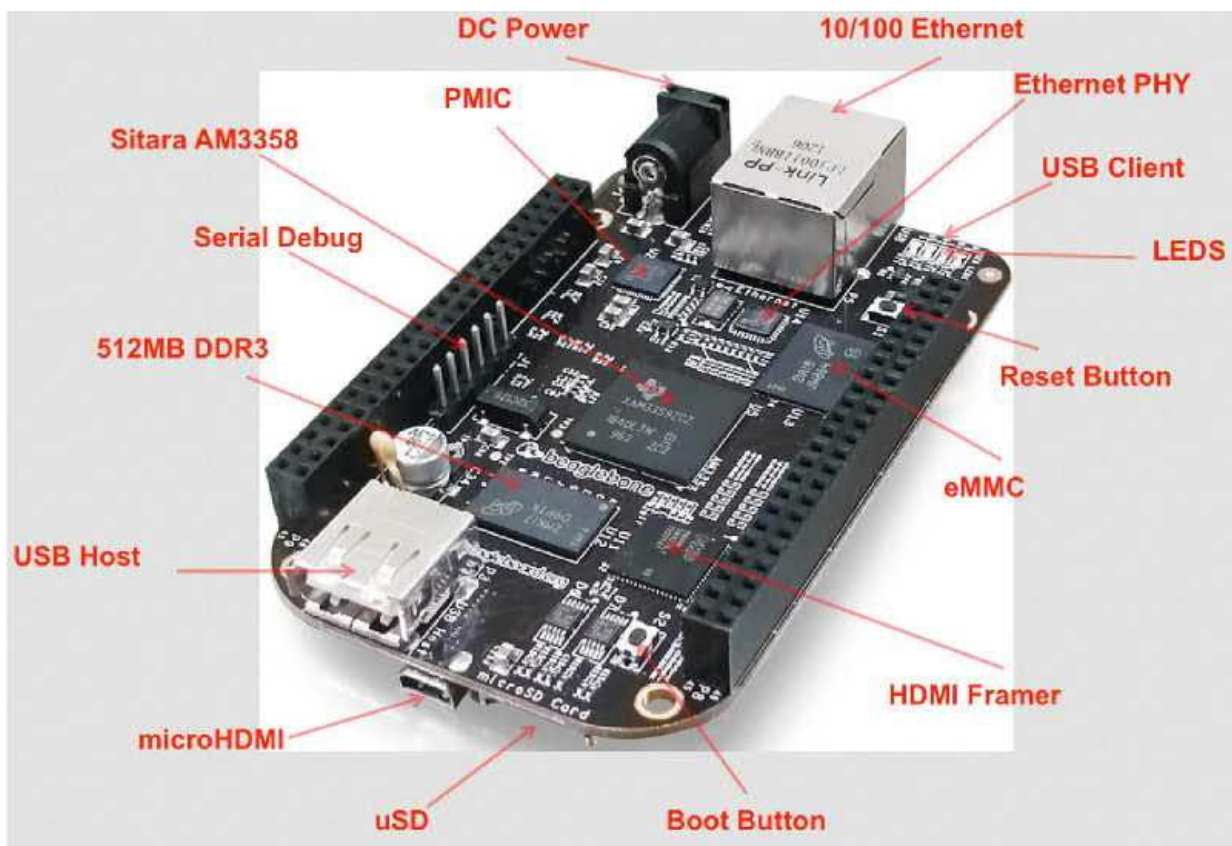
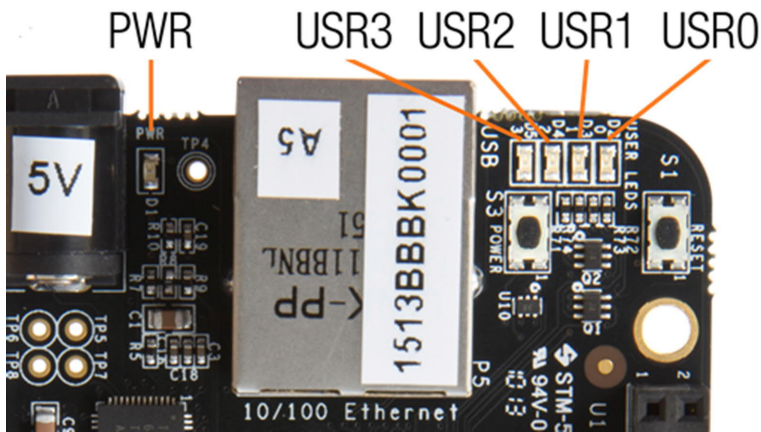


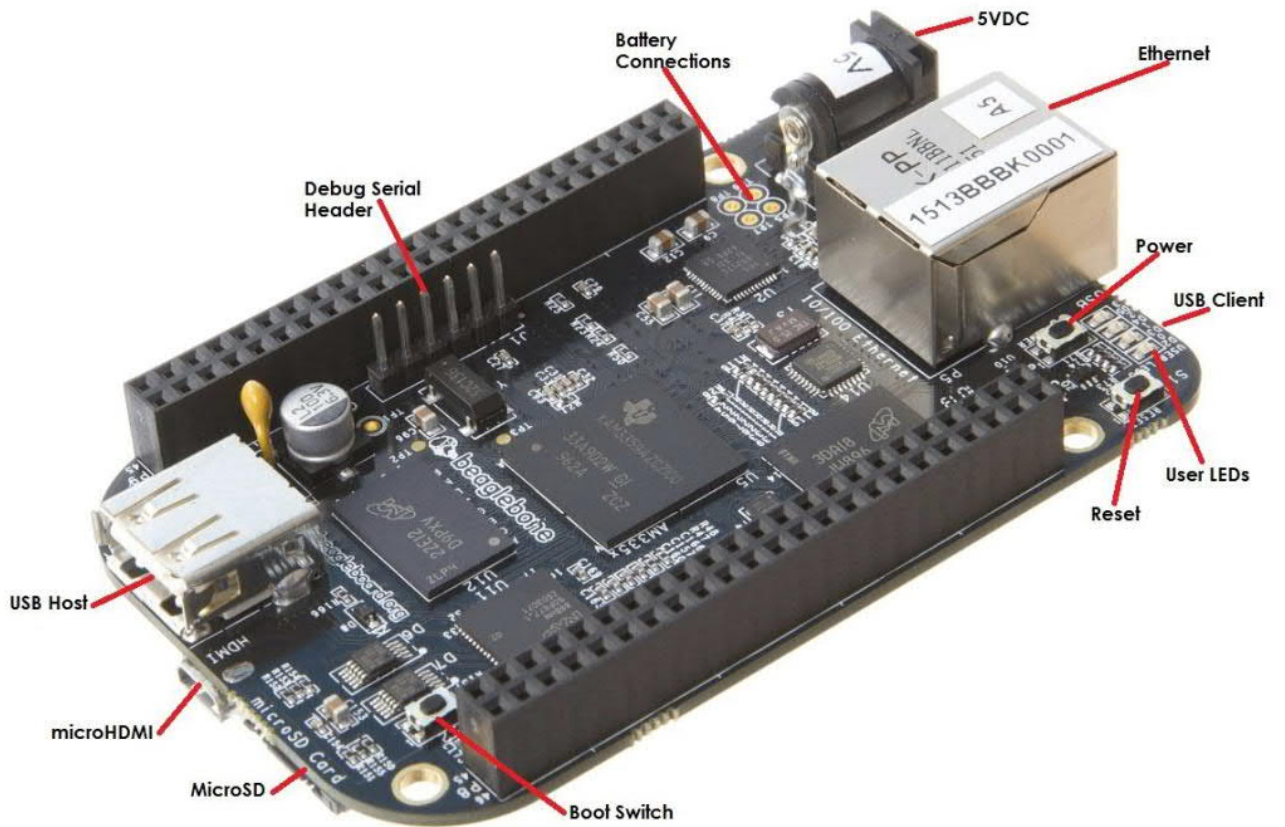
Bild: BeagleBone Black, Revision B mit 2 Taster (Boot, Reset)

MFM harddisk Reader/Emulator



Revision C, hier sind 3 Taster vorhanden. An den Leutdioden RESET und POWER (user) und auf der anderen Seite der Boot Taster.

rote B



Hard- und Software aus <http://www.pdp8.net/mfm/>

MFM harddisk Reader/Emulator

Kurzbeschreibung Board:

Festplatte Emulieren:

J1 20pol. Daten HD1 - Emulation Festplatte, hierzu den Festplatten-
J6 20pol. Daten HD2 controller mit J1(J6) und J2 verbinden
J2 34pol. Steuerung für beide Laufwerke. Drehung am Ende von
4 Adern

Festplatte Einlesen oder Beschreiben(beta):

J3 20pol. Datenkabel
J4 34pol. Steuerungskabel

Zum Einlesen der Fstplatte diese J3 und J4 verbinden. Ein Beschreiben der Festplatte ist auch möglich, allerdings noch im Teststadium und bedarf einer Konfigurationsänderung mit Neustart.

Jumper:

Jumper **P1** = WRITE - nur zum "Schreiben auf HD" stecken da sonst auch beim Einlesen Fehler auftreten können.

Jumper **P9** = CAPS - Die Kondensatoren dienen als Notstromversorgung und werden bei gesetzten P9 geladen

Beim Ausschalten wird die fehlende Versorgungsspannung registriert und das Betriebssystem fährt runter. Bei eingeschalteter Versorgungsspannung wird das Betriebssystem (debian) automatisch gestartet. (den Autostart des Emulators habe ich abgestellt)

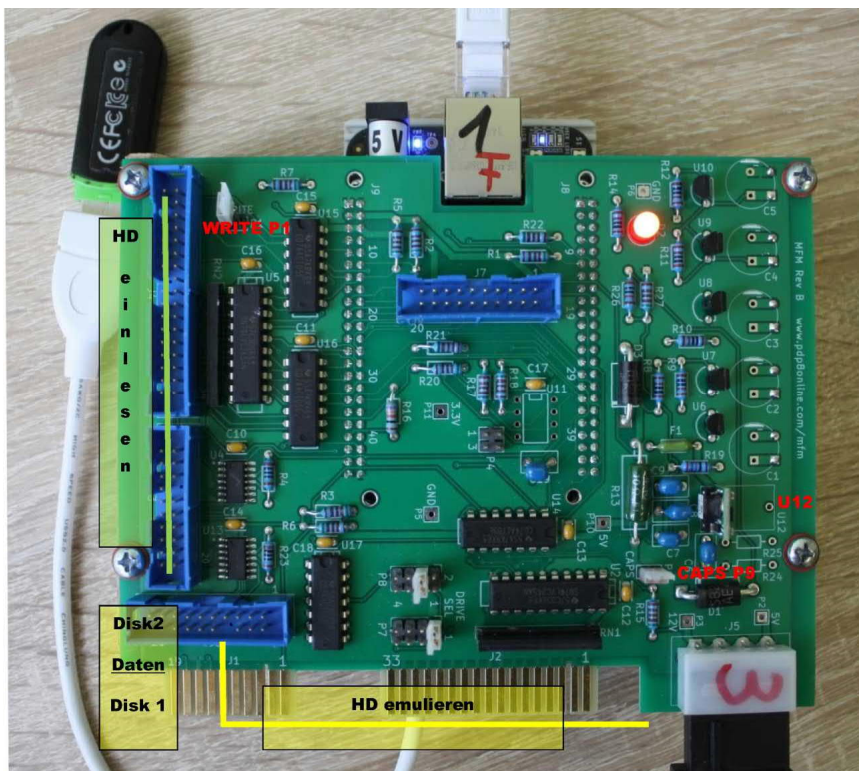


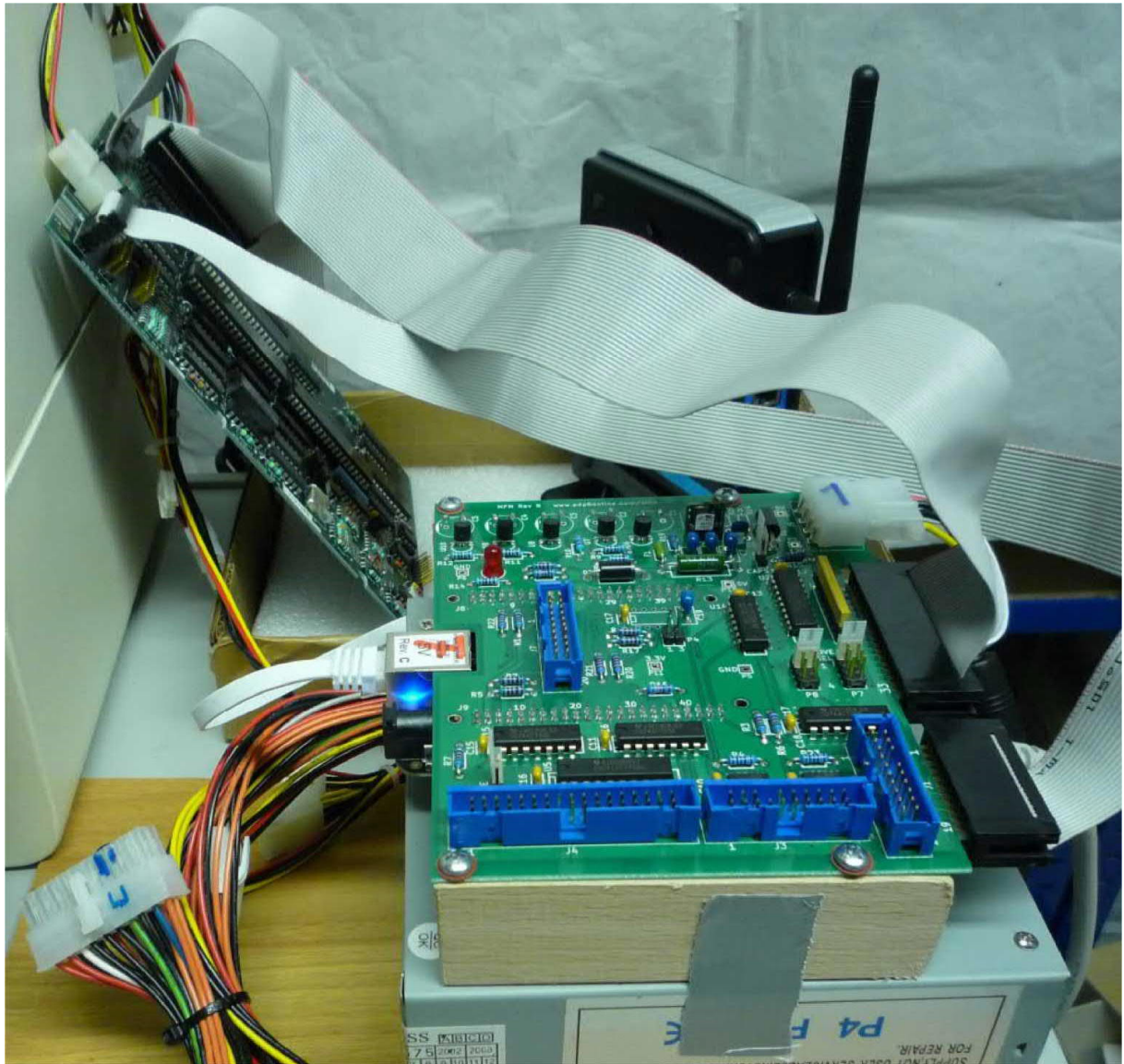
Bild aus einer HD-Emulation mit NCR DMV (J9=CAPS ist hier nicht gesetzt, LED ist aus.)

Beschaltung des ATX-Netztes

MFM harddisk Reader/Emulator

siehe

<https://www.elektronik-kompodium.de/sites/com/0601151.htm>



Die Stromversorgung des BeagleBoard geht über den Spannungswandler U12 der 12V nach 5V wandelt. Bei gesetztem CAPS Jumper und leuchtender Diode D2 darf das BeagleBoard **nicht** von der MFM-Emu-Platine ab- oder anstecken werden. Bei defektem Spannungswandler (bei einem meiner MFM Boards war der Spannungswandler U12 defekt und wurde von mir getauscht) kann das BeagleBoard über seine eigene Stromversorgung betrieben werden.



Mauser No:	5 8 0 -OKR-T/1.5-W12-C
Mfr. No:	OKR-T/1.5 -W12-C
Mfr.:	Murata Power Solutions
Description:	Non-Isolated DC/DC Converters 12Vin - 6Vout 1.5A. Positive Logic

MFM harddisk Reader/Emulator

Grundlegendes zum MFM-Emulator:

Ausführliches steht auf <http://www.pdp8.net/mfm/>. David Gesswein hat die Platine entwickelt und eine Miniserie gefertigt. Meine Version nutzt ein [Beagle Board Black](#) mit **4GB** 8-bit eMMC on-board "Flash Speicher", wogegen das ältere BBB nur **2GB** "Flash Speicher" hatte.

Mit 4GB eMMC habe ich mich entschlossen, das [Debian Image \(BBB-mfm-emu xxxxx.img.xz\)](#) in der aktuellen Version für Konsole aus dem Image von David um die XFCE Oberfläche zu erweitern. Zusammenbau, Installation und Testen beschreibt David ausführlich [hier](#).

MFM Reader/Emulator mit XFCE und Remote Desktop auf Beagle Board Black 4GB eMMC

Debian System für MFM - EMU installieren

BBB mit dem Image von David flashen. Aktuelles Image ist unter <http://www.pdp8.net/mfm/revb/> (momentan BBB-mfm-emu_v2.09.img.xz) zu finden, wobei das IMAGE mit Endung .XZ eine komprimierte Datei ist. Entpacken unter Windows ist mit [7-Zip](#) ab der Version 9 und [WinRAR](#) ab Version 5 möglich.

Das Image wird z.B. mit dem [WIN32 DISKIMAGER](#) auf eine 2 bis 4GB große Micro-SD-Karte geschrieben. Anschließend die Micro-SD-Kartins BBB einsetzen, USER SWITCH drücken und Stromversorgung einstecken.

1. Die Stromversorgung der DC Buchse sollte mindestens 5V / 2A betragen. Alternativ kann auch ein USB-Adapter verwendet werden.
2. Trennen das Netzwerk(Ethernet)-Kabel und entfernen Sie alle sonstigen Stecker und USB-Peripheriegeräte
3. BBB ausschalten durch USB / Netzkabel physisch trennen.
4. Beschriebenen Micro-SD-Karte in den Micro-SD-Kartensteckplatz der BBB stecken
5. Halte die Boot-Taste (S2) oben rechts (in der Nähe des SD-Kartensteckplatzes) gedrückt, und während diese Taste gedrückt gehalten wird das USB / Netzkabel einstecken um die Stromversorgung einzuschalten. Halte die Taste gedrückt, bis die LEDs zu blinken beginnen. Die blauen On-Board-LEDs sollten nacheinander leuchten und dann für die nächsten 5-25 Minuten weiter blinken (abhängig von der Größe und der Geschwindigkeit der SD-Karte).
6. Warte bis die LEDs aufhören zu blinken und alle 4 LEDs leuchten. Dies kann je nach verwendetem Image 5-25 Minuten dauern. Wenn der Flashvorgang fehlschlägt - z. B. blinken keine LEDs, oder es

MFM harddisk Reader/Emulator

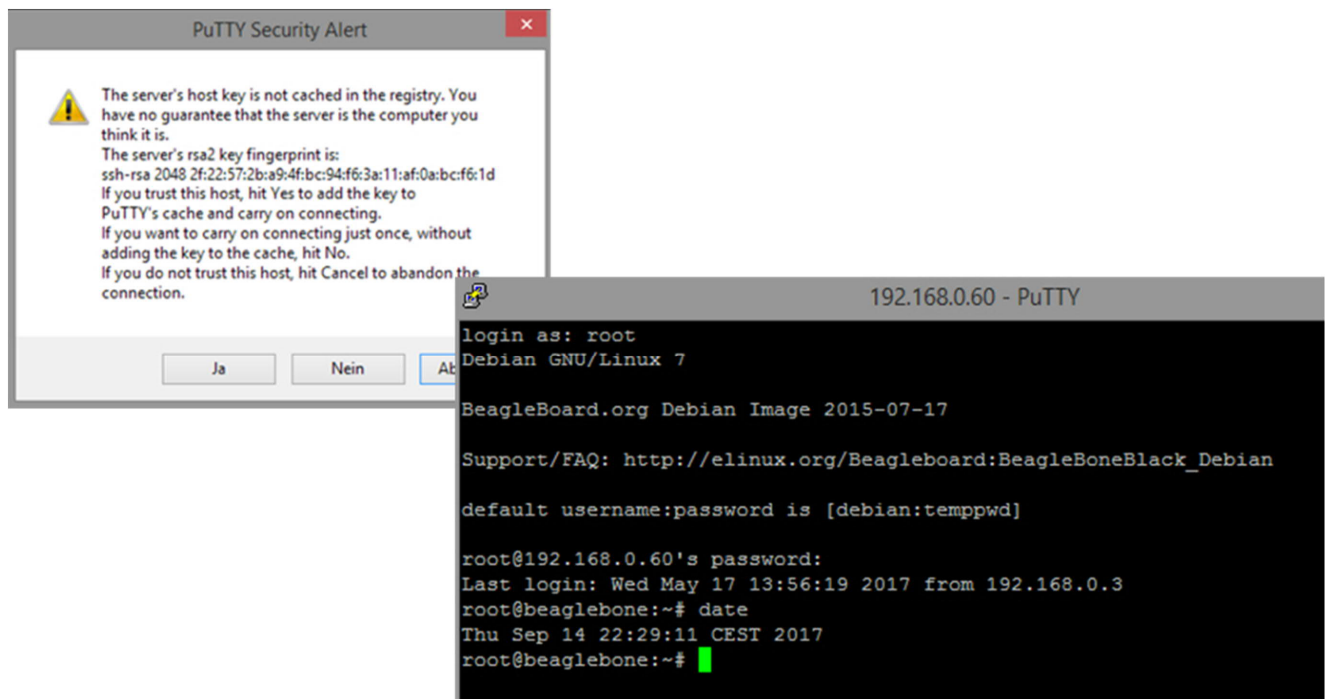
dauert länger als 45 Minuten - dann trenne die Stromversorgung und versuchen die BBB mit der Taste S2 zu resetten.

- Entferne die Micro-SD-Karte damit kein neuer automatischer Flash-Vorgang gestartet wird.
- Schließlich drücke den Einschalter (S3), um die Karte einzuschalten und das gerade installierte System sollte starten.

[\[hier klicken: Anleitung aus WWW im Original in englischer Sprache\]](#)

User einrichten:

Das BBB ist gestartet und per Netzwerk erreichbar. IP finden -> die MAC Adresse steht seitlich auf dem BBB und zum Zugriff über USB-RS232 oder IP nutze ich PUTTY. Die IP wird meist per DHCP über Router, Firewall oder Accespoint vergeben. Beim ersten Zugriff mit PUTTY kommt eine Warnmeldung zum RSA Fingerprint - diesen bestätigen und schon ist man auf der Console.



Dem ROOT ein Passwort zuweisen, damit Login per Remote Desktop geht

default [username:password] is [debian:temppwd]

MFM harddisk Reader/Emulator

login as: root

Da ich mit RDP (Remote Desktop) arbeite muß der ROOT ein Passwort haben.

```
root@beaglebone:~# passwd
Enter new UNIX password: geheim
Retype new UNIX password: geheim
passwd: password updated successfully
root@beaglebone:~#
```

Weiter geht es mit dem Einrichten des Systems.

Inzwischen ist Wheezy nicht mehr als aktive Distribution vorhanden, sondern hat den EOL (End Of Life) Status erreicht. Hierdurch sind die Einträge in `/etc/apt/sources.list` veraltet und nicht mehr gültig. Aktuell liegen die Dateien im Netz unter `/archive.debian.org/`. Beim **apt-get update** kommen daher entsprechende Fehlermeldungen.

Die Funktion des fertigen Images von David ist aber nicht beeinträchtigt. Da ich zum installierten Image noch einige Dateien installieren möchte muß ich die Einträge in `/etc/apt/sources.list` anpassen.

Beispiel aus einem Debian10 - diese Einträge sind für BBB zu ändern.

```
/etc/apt/sources.list 638/638
# https://wiki.debianforum.de/Sources.list
deb http://ftp.de.debian.org/debian/ stable non-free contrib main
deb http://http.debian.net/debian/ stable main contrib non-free
# deb-src http://ftp.de.debian.org/debian/ stable non-free contrib main
# deb http://ftp.de.debian.org/debian/ stable-updates non-free contrib main
deb http://security.debian.org/ stable/updates main contrib non-free
##deb-src http://security.debian.org/ stable/updates main contrib non-free
##deb-src http://ftp.de.debian.org/debian stable-updates main contrib non-free
deb [arch=amd64] https://download.virtualbox.org/virtualbox/debian buster contrib
```

Zu Debian Wheezy aus dem Netz:

Das Wheezy-updates-Repository enthielt Pakete, die bereitgestellt wurden, um das Haupt-Repository, d.h. Wheezy, zwischen den Minor Releases zu aktualisieren. Im Laufe der Zeit, als Wheezy auf 7.1, 7.2 und schließlich auf 7.11 aktualisiert wurde, wurden die in Wheezy

MFM harddisk Reader/Emulator

Updates enthaltenen Pakete in das Wheezy- Hauptrepository verschoben. Als Wheezy im Mai 2018 EOLed bekam, wurden keine Pakete in Wheezy-Updates aufbewahrt, also gab es keinen Grund, Wheezy-Updates in das Debian-Archiv zu verschieben.

Sie können die Liste aller Debian-Distributionen, die vom Archiv-Repository unterstützt werden, einsehen unter:

<http://archive.debian.org/debian/dists/>.

Besuchen Sie auch die README-Dateien unter

<http://archive.debian.org/README> und

<http://archive.debian.org/debian/README>

um eine noch vollständigere Liste der Archivinhalte zu erhalten.

Das Gleiche gilt für das security.debian.org Repository, da Wheezy EOL ist, erhalten Sie dort keine Pakete mehr.

Lösung:

Einträge in `/etc/apt/sources.list`:

```
deb http://archive.debian.org/debian wheezy main
deb http://archive.debian.org/debian-archive/debian-security/ wheezy
                                     updates/main
```

Hier die sources.list von david: (Zeilenumbruch entfernen)

(Anmerkung: als Datei im internet hinzufügen)

```
#deb http://ftp.us.debian.org/debian/ wheezy main contrib non-free
deb http://archive.debian.org/debian wheezy main contrib non-free
#deb-src http://ftp.us.debian.org/debian/ wheezy main contrib non-free

#deb http://ftp.us.debian.org/debian/ wheezy-updates main contrib non-free
#deb-src http://ftp.us.debian.org/debian/ wheezy-updates main contrib non-free

#deb http://security.debian.org/ wheezy/updates main contrib non-free
deb http://archive.debian.org/debian-archive/debian-security/
wheezy/updates main contrib non-free
#deb-src http://security.debian.org/ wheezy/updates main contrib non-free

#deb http://ftp.debian.org/debian wheezy-backports main contrib non-free
##deb-src http://ftp.debian.org/debian wheezy-backports main contrib non-free

#Kernel source (repos.rcn-ee.com) :
https://github.com/RobertCNelson/linux-stable-rcn-ee
#
```

MFM harddisk Reader/Emulator

```
#git clone https://github.com/RobertCNelson/linux-stable-rcn-ee
#cd ./linux-stable-rcn-ee
#git checkout `uname -r` -b tmp
#
deb [arch=armhf] http://repos.rcn-ee.com/debian/ wheezy main
#deb-src [arch=armhf] http://repos.rcn-ee.com/debian/ wheezy main
```

Hat man die *sources.list* bearbeitet, muss noch die Liste der verfügbaren Pakete mittels **aptitude update** oder **apt-get update** aktualisiert werden. Erst danach wird die Änderung an den verfügbaren Paketen berücksichtigt.

System aktualisieren:

Damit die Dateien per Update geladen werden in **/etc/apt/apt.conf.d/o2compress-indexes „gzip“** auf false stellen.

```
Acquire::GzipIndexes "true"; Acquire::CompressionTypes::Order:: "gz";
```

Nach

```
Acquire::GzipIndexes "false"; Acquire::CompressionTypes::Order:: "gz";
```

ändern.

Jetzt das System updaten (keinen neueren kernel einspielen!)

```
apt-get update    „damit wird die paket-datenbank aktualisiert“
apt-get upgrade   „damit wird die installierten Pakete aktualisiert“
apt-get install mc „filemanager“
```

NTP:

Das BBB hat leider keinen Accu und behält die Systemzeit nicht weshalb die Zeit und das Datum beim Start per NTP gesetzt werden sollte.

```
root@beaglebone: apt-get install ntp
```

Anschließend gewünschten ntp-server in **/etc/ntp.conf** eintragen. Ich habe local unter 192.168.0.1 einen NTP Server

```
X-----X
# You do need to talk to an NTP server or two (or three).
#server ntp.your-provider.example
# localer server von fritz

server 192.168.0.1
```

MFM harddisk Reader/Emulator

```
# pool.ntp.org maps to about 1000 low-stratum NTP servers. Your
server will
# pick a different set every time it starts up. Please consider
joining the
# pool: <http://www.pool.ntp.org/join.html>
server 0.debian.pool.ntp.org iburst
server 1.debian.pool.ntp.org iburst
server 2.debian.pool.ntp.org iburst
server 3.debian.pool.ntp.org iburst
X-----X
```

Manuell ntp neu starten:

```
root@beaglebone:/etc/init.d# sh ntp stop
```

```
[ ok ] Stopping ntp (via systemctl): ntp.service.
```

```
root@beaglebone:/etc/init.d# sh ntp start
```

```
[ ok ] Starting ntp (via systemctl): ntp.service.
```

```
root@beaglebone:/etc/init.d#
```

```
root@beaglebone:/etc/init.d# ntpd -g [justiere einen großen
Zeitfehler]
```

```
root@beaglebone:/etc/init.d# date
```

```
Sat Mar 4 12:08:47 UTC 2017
```

```
root@beaglebone:/etc/init.d#
```

```
X-----X
```

XFCE installieren:

Für angenehme Benutzung XFCE und XRDP installieren

-> <https://wiki.debian.org/Xfce>

```
root@beaglebone:/# apt-get install xfce4
```

irgendwann ist xfce fertig.

```
apt-get install xfce4-goodies
```

```
apt-get install tuxcmd
```

```
apt-get install tuxcmd-modules
```

```
apt-get install xfce4-terminal
```

```
apt-get install leafpad
```

nur bei 4GB FLASH - die ältere BBB Version hat nur 2GB

MFM harddisk Reader/Emulator

```
apt-get install libreoffice-writer
```

```
apt-get install xrdp    (für den grafischen Zugang)
```

dann ein **REBOOT**

Es dauert eine Minute bis XFCE4 eingerichtet ist und das Bild aufgebaut wird. Die DEFAULT Einrichtung bitte bestätigen. Ausloggen und als ROOT neu einloggen.

```
user: ROOT
```

```
pw: geheim
```

Wer möchte kann weitere hilfreiche Anwendungen installieren.

WWW browser wählen

```
Sudo apt-get install midori  
vielleicht auch  
sudo apt-get install netsurf
```

Bilderbetrachter

```
sudo apt-get install Ristretto
```

PDF Viewer

```
sudo apt-get install xpdf
```

Der **Beagle Bord Black** ist nun mit dem MFM Emulatorimage von David und zur leichte Bedienung mit XFCE eingerichtet. Per RDP ist der grafische Zugang möglich.

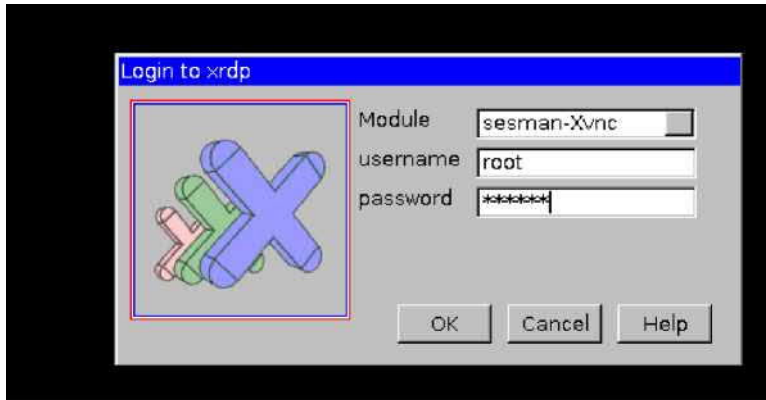
System starten und einloggen:

Kabel entsprechend der gewünschten Nutzungsart (Einlesen, Emulieren) anschließen, Netzwirkabel anschließen und Stromversorgung (12V) einschalten.

Im Netzwerk zur MAC-Adresse (auf dem Bord aufgedruckt) die IP finden.

Verbinden an die entsprechende IP mit z.B. [PUTTY](#) per [SSH](#). Ich habe ja auf meinem System XFCE mit [Remote Desktop \(RDP\)](#) installiert da hiermit das Handling einfacher ist.

MFM harddisk Reader/Emulator



Aktuell bei mir als Remotedesktopverbindung zur IP 192.168.0.60 verbinden.

Beispielbilder: (Der Cursor blinkt nicht)

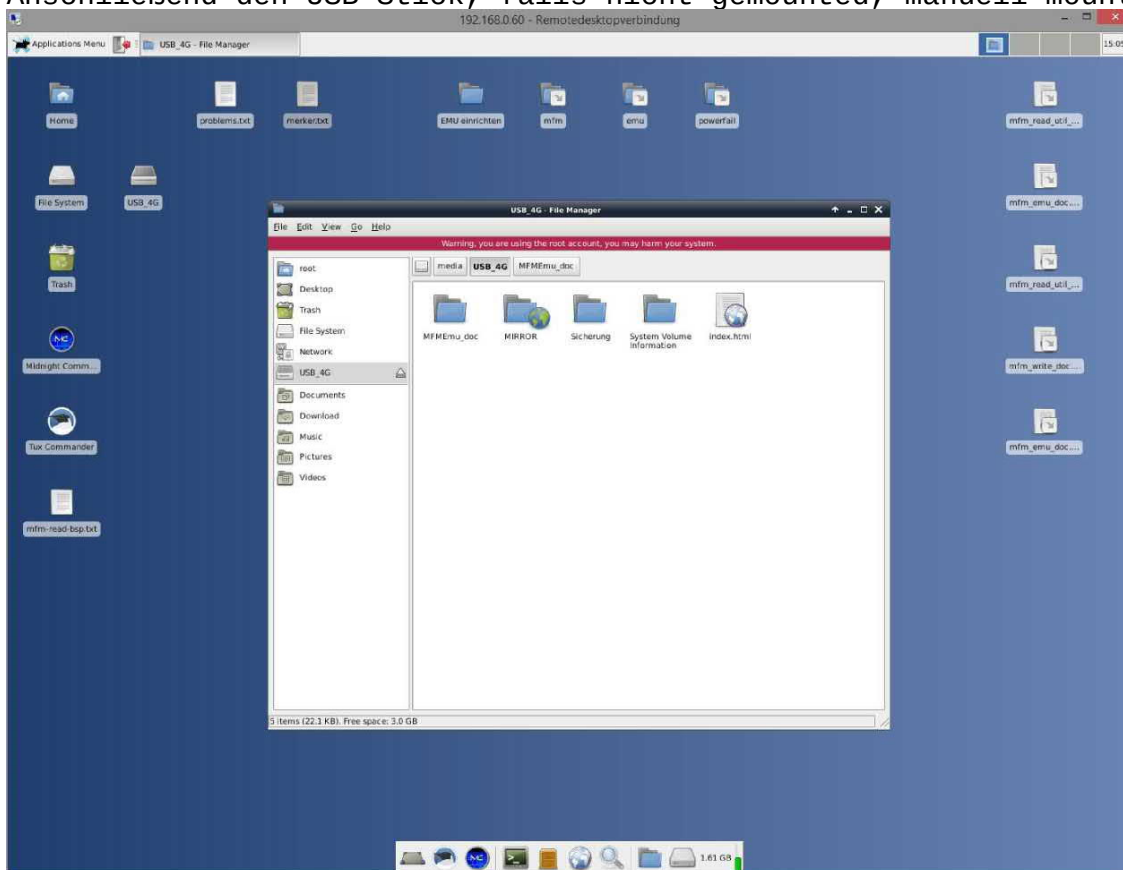
Login:

Root | geheim
(mein default wegen remote Desktop login)

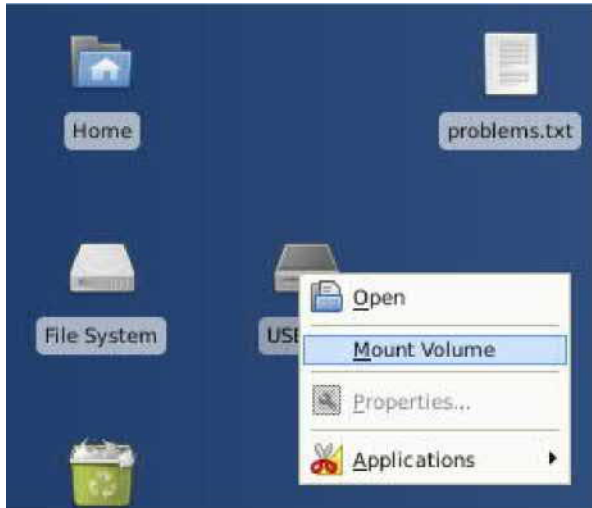
oder als

User: debian | tempwd

Anschließend den USB Stick, falls nicht gemounted, manuell mounten.



MFM harddisk Reader/Emulator

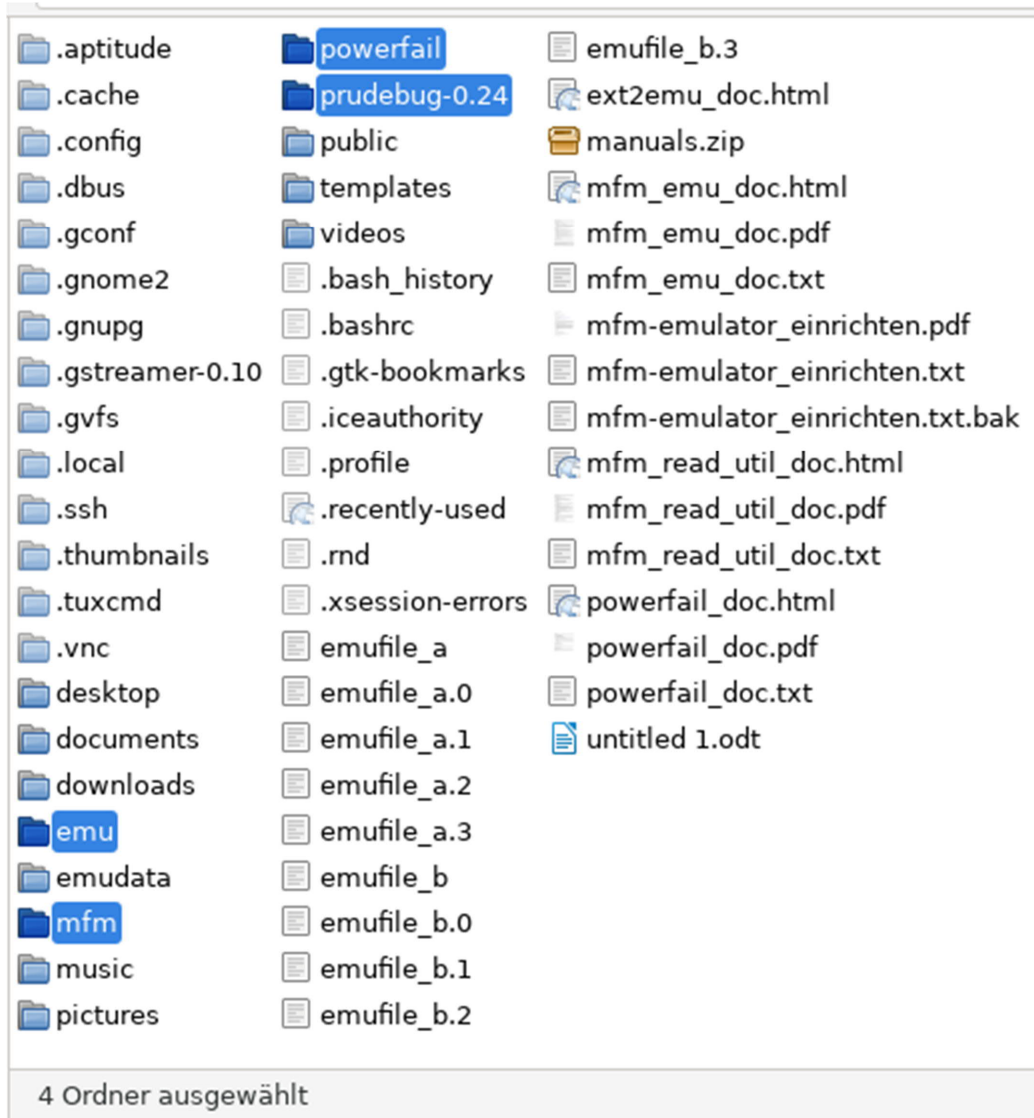


Hier lassen sich die ausgelesenen Daten der Festplatte(n) auf den USB Stick speichern und dann auf den PC übertragen.

Das ROOT Directory:

Die Verzeichnisse zum MFM-Emulator sind blau markiert. Im Folgenden werden die Verzeichnisse **MFM** und **EMU** betrachtet. Ich habe schon einiges im ROOT Verzeichniss gemacht, weshalb hier mehr Dateien als in der Originalinstallation vorhanden sind.

MFM harddisk Reader/Emulator



MFM harddisk Reader/Emulator

Ich sichere mit einem kleinen Script bestimmte Dateien und Verzeichniss als TAR auf den USB-Stick. Gerade bei dem BBB mit 2GB Speicherplatz ist das zu empfehlen.

```
#!/bin/bash
```

```
DATE=$(date +%Y-%m-%d-%H%M%S)
```

```
# Quelle aus https://wiki.ubuntuusers.de/Skripte/Backupsript/
```

```
# pfad sollte nicht mit "/" enden!
```

```
# Dies ist nur ein Beispiel - bitte an eigene Beduerfnisse anpassen.
```

```
# Man muss schreibberechtigt im entsprechenden Verzeichnis sein.
```

```
# Datei nach /usr/bin/ kopieren und ausführbar machen. So kann das backup
```

```
# von überall oder per cron gestartet werden
```

```
# Zieldatentraeger muss gemountet sein !
```

```
# mount /dev/sdc1 /media/usb-drive/
```

```
#[ mount | grep sdc1 /dev/sdc1 on /media/usb-drive type vfat
```

```
#[(rw,relatime,mask=0022,dmask=0022,codepage=437,ioccharset=utf8,shortname=mixed,err#o  
rs=remount-ro)]
```

```
# bei mir:
```

```
mount /dev/sda /media/USB-2GB
```

```
BACKUP_DIR="/media/USB-2GB/backup"
```

```
# Hier Verzeichnisse auflisten, die gesichert werden sollen.
```

```
# Dies ist nur ein Beispiel - bitte an eigene Beduerfnisse anpassen.
```

```
# Bei Verzeichnissen, fuer die der User keine durchgehenden Leserechte hat (z.B. /etc) sind #  
Fehler vorprogrammiert.
```

```
# Pfade sollte nicht mit "/" enden!
```

```
SOURCE="/etc /root/mfm /root/emu /root/Desktop"
```

```
tar -cjpeg $BACKUP_DIR/backup-$DATE.tar.bz2 $SOURCE
```

```
EXIT
```

Die Datei wird nach **/usr/bin/backup.sh** kopiert und auf dem Desktop ein Starticon abgelegt.

MFM harddisk Reader/Emulator

Einlesen einer HD

(Beschreibung der Kommandozeilenbefehle in [mfm_read_doc.](#))

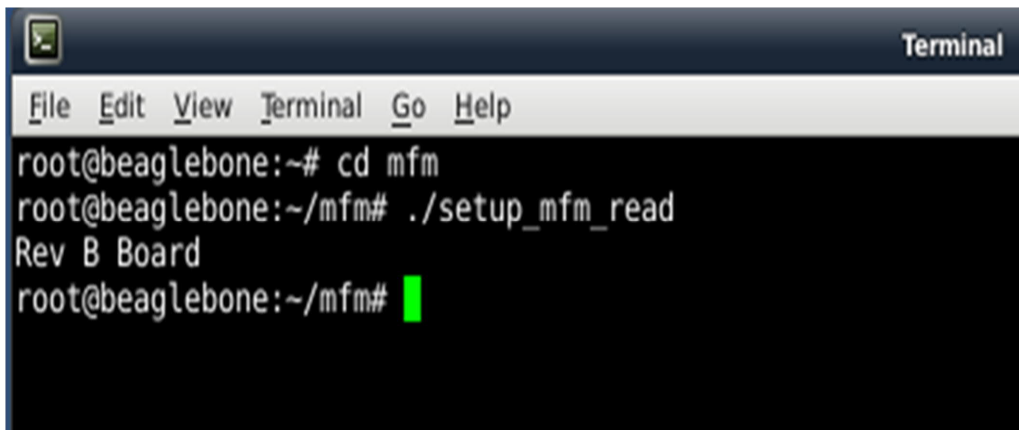
1. Mit „cd mfm“ ins MFM-Verzeichniss wechseln.
2. Mit dem Befehl **./setup_mfm_read** das Board konfigurieren. Es meldet als Bestätigung „Rev B Board“

Festplatte einlesen:

J3 20pol. Daten | Festplatte einlesen - hierzu die Festplatten
J4 34pol. Befehle | mit J3 und J4 verbinden.

Jumper P1 = Write abziehen.

1. Mit „cd mfm“ ins MFM-Verzeichniss wechseln.
2. Mit dem Befehl ./setup_mfm_read das Board konfigurieren. Es meldet bei mir als bestätigung „Rev B Board“.



```
Terminal
File Edit View Terminal Go Help
root@beaglebone:~# cd mfm
root@beaglebone:~/mfm# ./setup_mfm_read
Rev B Board
root@beaglebone:~/mfm#
```

3. Die Festplatte wird durch Eingabe von **./mfm_read -a** analysiert und die Daten der der angeschlossenen HD zur Kontrolle angezeigt. Dies wird beim echten Einlesen nochmals durchgeführt.

Beispiel: HD Einlesen mit folgendem Kommando:

```
./mfm_read -a -e st125_ext -m st125_emu -t st125_tra
```

-e steht für „Extract“, d.h. das ist ein Hexdump

-m steht für „Emulator“, d.h. diese Datei ist zur Emulation der
HD notwendig

-t steht für „Transitions“, also der rohe Bitstrom vom
Controller.

MFM harddisk Reader/Emulator

Hier MFM-HD von NCR DM-V CP/M-80

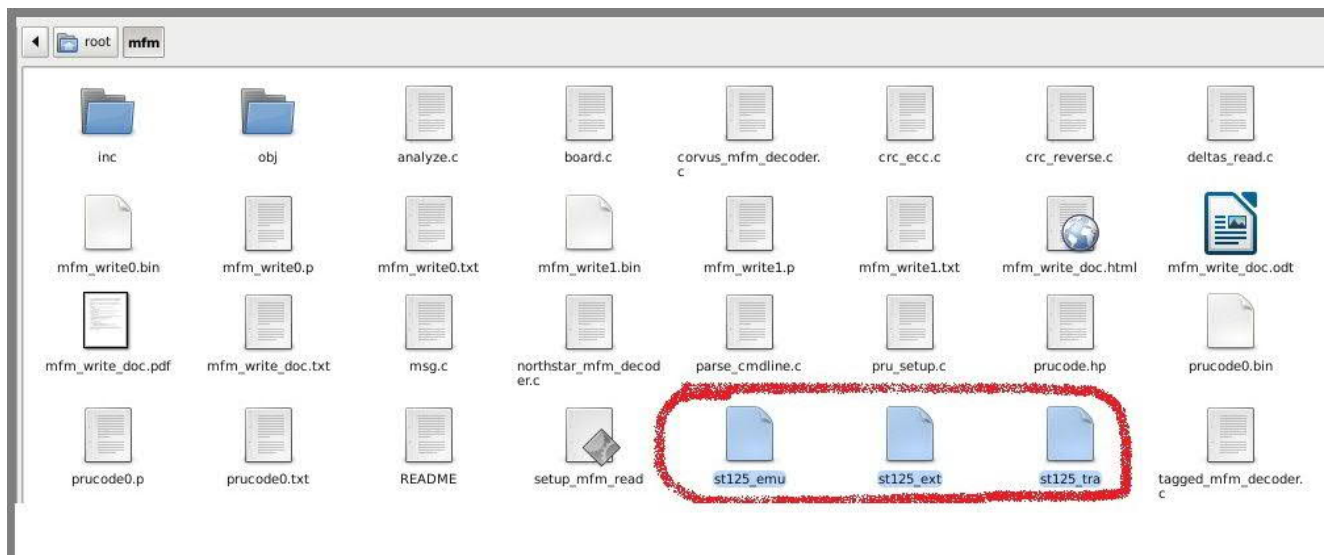
```
Terminal
File Edit View Terminal Go Help
root@beaglebone:~/mfm# ./mfm_read -a
Board revision B detected
Found drive at select 1
Drive RPM 3601.5
Matches count 34 for controller WD_1006
Header CRC: Polynomial 0x1021 length 16 initial value 0xffff
Sector length 512
Data CRC: Polynomial 0x140a0445 length 32 initial value 0xffffffff
Selected head 4 found 0, last good head found 3
Read errors trying to determine sector numbering, results may be in error
Number of heads 4 number of sectors 17 first sector 0
Interleave (not checked): 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Drive supports buffered seeks (ST412)
No sectors readable from cylinder 615
Disk has recalibrated to track 0
Stopping end of disk search due to recalibration
Number of cylinders 615, 21.4 MB

Command line to read disk:
--sectors 17,0 --heads 4 --cylinders 615 --header_crc 0xffff,0x1021,16,0 --data_crc 0xffffffff,0x140a0445,32,5 --format WD_1006 --sector_length 512 --retries 50,4 --drive 1
root@beaglebone:~/mfm#
```

Voraussetzung zum Extrahieren der Daten aus dem rohen Bitstrom ist, daß der verwendete Festplattencontroller bekannt ist.

Falls bei der Analyse „MFM_READ -a“ Fehler angezeigt werden und die Analyse recht lange dauert ist aller Wahrscheinlichkeit nach der Festplattencontroller unbekannt, womit die Parameter zu `-data_crc` und `-header_crc` fehlen. Hier kann zur Archivierung mit „-t Transitions“ nur der rohe Datenstrom gelesen werden. => **siehe Unbekannter Festplattencontroller**

Nach dem Einlesen befinden sich die 3 Dateien im Verzeichniss MFM.



MFM harddisk Reader/Emulator

Konsolenausgabe während des Einlesens:

```
root@beaglebone:~/mfm# ./mfm_read -a -e st125_ext -m st125_emu -t
st125_tra Board revision B detected Found drive at select 1 Drive RPM
3602.1
```

```
Matches count 34 for controller WD_1006
Header CRC: Polynomial 0x1021 length 16 initial value 0xffff Sector
length 512
Data CRC: Polynomial 0x140a0445 length 32 initial value 0xffffffff
Selected head 4 found 0, last good head found 3
```

```
Read errors trying to determine sector numbering, results may be in
error \lumber of heads 4 number of sectors 17 first sector 0
Interleave (not checked): 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Drive supports buffered seeks (ST412)
```

```
No sectors readable from cylinder 615
Disk has recalibrated to track 0
Stopping end of disk search due to recalibration
Number of cylinders 615, 21.4 MB
```

Command line to read disk:

```
--sectors 17,0 --heads 4 --cylinders 615 --header_crc
0xffff,0x1021,16,0 --data_crc 9xffffffff,0x140a0445,32,5 --format
WD_1006 --sector_length 512 --retries 50,4 --drive 1
```

```
Retries failed cyl 51 head 1
Bad sectors on cylinder 51 head 1: 4
Retries failed cyl 53 head 1
Bad sectors on cylinder 53 head 1: 4
Retries failed cyl 54 head 1
Bad sectors on cylinder 54 head 1: 4
Retries failed cyl 55 head 1
Bad sectors on cylinder 55 head 1: 4
Found cyl 0 to 614, head 0 to 3, sector 0 to 16
Expected 41820 sectors got 41816 good sectors, 0 bad header, 4 bad
data 9 sectors marked bad or spare
```

```
9 sectors corrected with ECC. Max bits in burst corrected 5 Track
read time in ms min 27.931500 max 1684.372292 avg 44.853666
```

```
root@beaglebone:~/mfm#
```

MFM harddisk Reader/Emulator

Emulation einer HD:

Emulation: (Beschreibung der Kommandozeilenbefehle in [mfm emu doc](#))

J1 20pol. Daten HD1 | Emulation - Hierzu den originalen Festplatten-
J2 34pol. Steuerung | controller mit J1 und J2 verbinden.
J6 20pol. Daten HD2 | 2. Festplattenemu falls gewünscht

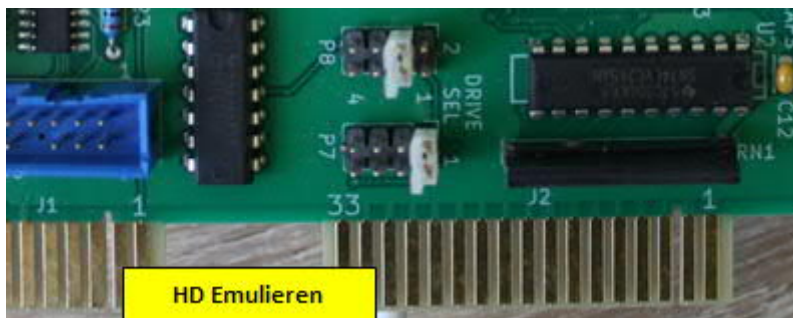
Es wurden mit `./mfm_read -a -e st125_ext -m st125_emu -t st125_tra -n "Test HD einlesen ST125"` 3 Dateien geschrieben st125_ext, st125_emu und st125_tra.

-a Analyse HD Format analysieren
-e TEST_ext Extracted Data File.
-m TEST_emu Emulation Datei
-t TEST_tra RAW MFM Flußwechsel

Entferne die Kabel zum Lesen eines Laufwerks, bevor ein Laufwerk emuliert werden soll.

Verbinde die Kabel vom Controller mit J1 und J2 (Platinenstecker). Setze die Driveselect-Jumper P7 und P8 auf die Laufwerksnummer, die emuliert werden soll.

Default:



RN1 sollte installiert werden, es sei denn ein Laufwerk ist als letztes am Kabel terminiert.

Beachten dass `setup_emu` nur einmal pro Boot ausgeführt werden muss.

Starte Emulation:

```
cd ~/emu
./setup_emu
./mfm_emu --drive 1 --file ../emu_file
```

Hier `./mfm_emu --drive 1 --file ../st125_emu`

MFM harddisk Reader/Emulator

Versuchen Sie dann, den an den Laufwerksemulator angeschlossenen Rechner zu starten oder auf das emulierte Laufwerk zuzugreifen. Der MFM-Emulator sollte auf der console die Zugriffe anzeigen. Der Rechner sollte sich so verhalten, als hätte er die normale Festplatte angeschlossen.

Wenn keine zu emulierende Festplatte eingelesen wurde kann auch mit einem unformatierten Laufwerk versucht werden.

Emulation mit unformatierter Festplatte. Hierzu müssen die Laufwerkparameter explizit angegeben werden.

```
cd ~/emu
```

```
./mfm_emu --drive 1 --file ../emu_file --initialize --cylinders # --heads #
```

Ersetzen Sie # durch die richtigen Nummern für das Laufwerk, das Sie emulieren möchten (es wird keine Angabe der Sektoren benötigt) Führen Sie dann den Befehl zum Formatieren auf niedriger Ebene auf dem Computer aus, der an den Laufwerksemulator angeschlossen ist. Der mfm-Emulator sollte auf der Console die Zugriffe anzeigen und das Format sollte fehlerfrei sein.

Wenn Sie versuchen möchten, zwei Laufwerke zu emulieren, verbinden Sie J6 mit Ihrem Controller und verwenden Sie --drive 1,2 --file file1, file2 in der Befehlszeile. Dies funktioniert nur, wenn das System für beide emulierten Laufwerke dasselbe Steuerkabel verwendet.

MFM harddisk Reader/Emulator

Unbekannter Festplattenkontroller:

Im Juni 2018 hatte ich eine Festplatte zum Nixdorf 8870 Quattro/7 welche gesichert werden sollte. Das Lesen der Festplatte wurde leider nach längerem leseversuch mit einer Fehlermeldung beendet.

Befehl:

```
mfm_read --analyze --transitions_file raw_data --extracted_data_file  
extracted_data test --note "irgendetwas"
```

Mit obigem Kommando versucht die Software aus den Rohdaten dem Flußwechsel entsprechende binäre Daten zu gewinnen.

Das Format des verwendeten Nixdorf Harddiskkontrollers ist leider nicht kompatibel zu den im Programm vorhandenen Controllern und es erfolgte eine Fehlermeldung die allerdings nicht direkt Rückschlüsse auf das Problem gab.

Lösung:

Einlesen der Rohdaten zur Sicherung:

```
./mfm_read --transitions_file raw_data --cylinders # --heads # --drive  
#
```

Es bedarf der Analyse der „**header_crc**“ und „**data_crc**“ durch David Gesswein djg@pdp8online.com, damit aus den Rohdaten die Daten extrahiert und decodiert werden können.

MFM harddisk Reader/Emulator

Hier Teile der Originaldokumentation von David und Antworten zum Beschreiben von Festplatten.

#####

Starting Emulation on Power On

#####

Using my prebuilt image the default when enabled is to emulate a single drive from /root/emufile_a. The emulation file will not be backed up on boot. If you want to start the emulator at boot with these options execute the following command

```
systemctl enable mfm_emu.service
```

If you wish to change the options edit /etc/mfm_emu.conf. The file has comments describing what the configuration variables do. For example if you wish to emulate two drives set EmuFN2 to the second file name. If your emulated drive has information you wish not to lose you may wish to enable backup. Set the Backup variable to the type of backup. Copy just copies the emulator file. rdiff and xdelta do a binary difference between the files to take less space. If you do something that changes most of the image file such as defragment the binary difference may take long enough for your computer to timeout. The straight copy is quicker but for small changes will take much more space. I didn't find a clear winner between rdiff and xdelta.

It seems to take about 12 seconds from power on until the mfm emulator is running if no backup is performed.

MFM harddisk Reader/Emulator

Versuch die HD zu beschreiben:

Jumper P1 = Write zum Beschreiben der HD setzen, zum Lesen abziehen.

Das Programm mfm_write kann eine Emulatordatei auf eine Festplatte schreiben. Die ist allerdings noch im Teststadium.

MFM_WRITE:

mfm_write can write an emulator file to a disk drive. I got it to the state I could write the disk I needed but it is not a finished program. Currently it can only write an entire disk at once. You need to edit mfm_write.c main routine to set the parameters for your drive. I have not fixed command line parsing.

These command line options will be supported at some time along with options for setting the write precompensation. Currently only emulation_file, version, and quiet work. Emulation_file must be specified.

- begin_time -b #
The number of nanoseconds to delay from index to start reading track
- cylinders -c #
The number of cylinders.
- drive -d #
Drive number to select for reading. Only valid for read command. Drives are number 1 to 4.
- emulation_file -m filename
File name to write emulation bit data to. No file created if not specified
- heads -h #
The number of heads.
- quiet -q #
Bit mask to select which messages don't print. 0 is print all messages. Default is 1 (no debug messages). Higher bits are more important messages in general.
- unbuffered_seek -u
Use unbuffered/ST506 seeks. Default is buffered/ST412.
- version -v
Print program version number.

MFM harddisk Reader/Emulator

To work mfm_read-00A0.dts

```
line 0x190 0x07 // OUT P9_31 = gpio3_14
```

needs to be commented out and this line uncommented before running setup_mfm_read:

```
//0x190 0x2d // OUT P9_31 = pr1_pru0_pru_30_0
```

Use mfm_read to verify the disk is properly written. The first attempt had a couple tracks that seem to be written to the wrong head. The next run worked ok. This program does not do anything to avoid using bad locations on the disk.

Ausschnitt aus mfm_read-00A0.dts:

```
// All inputs pullup
// All outputs fast pullup disabled
// These go to PRU0

0x034 0x06 // OUT P8_11 = pr1_pru0_pru_30_15
0x03c 0x35 // IN/OUT P8_15 = pr1_ecap0 0x184
0x36      // IN  P9_24 =
pr1_pru0_pru_31_16
0x1ac 0x2d //      OUT  P9_25 =      pr1_pru0_pru_30_7
0x1a4 0x36 //      IN   P9_27 =      pr1_pru0_pru_31_5
0x19c 0x36 //      IN   P9_28 =      pr1_pru0_pru_31_3
0x194 0x36 //      IN   P9_29 =      pr1_pru0_pru_31_1
0x198 0x36 //      IN   P9_30 =      pr1_pru0_pru_31_2

!      //Until we support write make pin GPIO
!      0x190 0x07 // OUT P9_31 = gpio3_14 [for write hd comment out]
!      //0x190 0x2d // OUT P9_31 = pr1_pru0_pru_30_0 [for write uncomment]
0x1a8 0x2d // OUT P9_41 = pr1_pru0_pru_30_6
0x1a0 0x36 // IN  P9_42.1 = pr1_pru0_pru_31_4
```

MFM harddisk Reader/Emulator

Hinweise von David zum MFM_WRITE:

Hier Hinweise von David zum Versuch das Image für die Festplatte eines ICL Computers zurückzuschreiben.

Betreff: Re: Write image with MFM Reader/Emulator to a second Harddiskdrive.

Von: David Gesswein <djg@pdp8online.com>

If you have an emulator image you are wanting to write to another drive it is likely to work. A recent release fixed picking up the drive geometry from the emulator file. The pre compensation and drive to write to are still hard coded in the source and will need to be set for your drive.

```
drive_params.write_precomp_cyl = 512;

drive_params.early_precomp_ns = 10;

drive_params.late_precomp_ns = 10;

drive_params.drive = 1;
```

Looks like that format uses begin_time. I haven't tested that it works. If it doesn't I can fix it. Rereading with the emulator will give some indication the write is good though that won't guaranteed if will work on the real computer. I think you will be the first person to use this code other than me. You will need to reboot after writing before a read will work. Also pull the write jumper before the reboot (and install before trying to write).

If you are needed to create a disk from a extracted data file the format info would need to be added to inc/mfm_decoder.h

--

#####

Betref: Re: Write image with MFM Reader/Emulator to a second Harddiskdrive

Von: David Gesswein <djg@pdp8online.com>

Datum: 07.09.2017 04:04

If you are needed to create a disk from a extracted data file the format info would need to be added to inc/mfm_decoder.h

If one of the controllers for the systems you have to test with use begin_time that would be the better one to test with. The begin_time setting will print out when you read the disk.

MFM harddisk Reader/Emulator

Another issue I didn't previously say is that my code doesn't attempt to deal with bad sectors on the disk. If the target disk has bad sectors whatever ends up on them will get errors when read. Any flagging of bad tracks/sectors on the original disk will still be marked bad though they are likely good on the target disk.

#####

Betref: Re: Write image with MFM Reader/Emulator to a second Harddiskdrive

Von: David Gesswein <djg@pdp8online.com>

Datum: 10.09.2017 15:20

n Sun, Sep 10, 2017 at 03:07:27AM -0700, Fritz wrote:

>1. "controllers for the system " do you mean the MFM-Emulator hardware (I have several) or the MFM-Controller of the old computer? The controller in the old computer. The only way you would know would be to read the disk with the MFM emulator or use mfm_util on an emulator file.

>2. "begin_time".. with this I have to look into your documents and hope I understand a little bit. Please explain the "begin_time" for a novice like me.

[David]

It's a command line option.

http://www.pdp8online.com/mfm/code/mfm/mfm_read_util_doc.html

If you read a drive with mfm_read --analyze it will print the value in option list when it prints "Command line to read disk:"

When reading a disk it reads the transitions from index pulse to index pulse. If an index pulse occurs in the middle of a sector it then can't decode it. Begin time shifts the actual start and end of reading by that time interval so it happens in the fill data between sectors. Index pulse is a signal that goes active one per revolution of the disk. Most controllers when formatting the disk wait for the index pulse then start writing from the first sector. Some controllers seem to be really slow getting to writing the first sector so the last sector straddles the index pulse.

The one ICL computer I have information on uses the Xebec S1410 which needs non zero begin time. Other models may use different controllers which don't need it.

The system I tried mfm_write with uses 0 begin time so non zero may never have been tested, I was just saying if one of the two systems you had to test with used non zero begin time that would be a better

MFM harddisk Reader/Emulator

test. You can also test by making an emulator file that uses begin time, writing to a drive, and reading it back. Since Xebec isn't currently supported by ext2emu you can test using northstar format.

Create a file with 16*heads*cylinders. For this example 4 heads and 100 cyl.

Adjust as needed for your drive

```
dd if=/dev/zero of=/tmp/zeros bs=512c count=6400
```

```
ext2emu --emu /tmp/emu --ext /tmp/zeros --cyl 100 --heads 4 --format
```

```
NorthStar_Advantage
```

Then use `mfm_write` to write `/tmp/emu` to a drive and then use `mfm_read` to read it.

You need to look up the drive and see what cylinder is recommended for write precompensation and edit line in `mfm_write.c`. Unless you have more information leave the `precom_ns` values as is.

Also see http://www.pdp8online.com/mfm/code/mfm/mfm_write_doc.html

for change needed to `mfm_read-00A0.dts`. Will need to reboot each time the file is changed. Will need to be changed back to read. Also write jumper needs to be installed for write and best to remove it for reading.

```
mfm_write --emu /tmp/emu
```

```
mfm_read --ana --ext /tmp/t
```

Should get something like

Command line to read disk:

```
--sectors 16,0 --heads 4 --cylinders 100 --header_crc 0x0,0x0,16,0 --  
data_crc 0x0,0x0,32,0 --format NorthStar_Advantage --sector_length 512  
--retries 50,4 -drive 0 --begin_time 230000
```

And the disk reads with only the number of errors expected from the bad sectors on the disk.