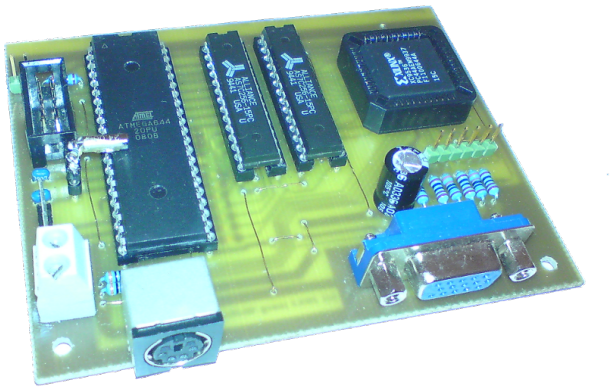
AX82: Ein ZX Spectrum Emulator auf AVR Basis

V0.32 (c) 2012 Jörg Wolfram



1 Rechtliches

Die Programme unterliegen der GPL (GNU General Public Licence) Version 3 oder höher, jede Nutzung der Software/Informationen nonkonform zur GPL oder ausserhalb des Geltungsbereiches der GPL ist untersagt!

Die Veröffentlichung dieses Projekts erfolgt in der Hoffnung, daß es Ihnen von Nutzen sein wird, aber OHNE IRGENDEINE GARANTIE, auch ohne die implizite Garantie der MARKTREIFE oder der VERWENDBARKEIT FÜR EINEN BESTIMMTEN ZWECK.

Alle im Text genannten Marken sind Eigentum des entsprechenden Inhabers.

2 Features

Als Eingabegerät dient eine normale PS2-Computertastatur, als Ausgabegeräte können Fernsehgeräte mit Scart- oder BAS-Eingang (S/W) oder VGA-Monitore verwendet werden.

- ATMega644(P), auf 24 MHz ”übertaktet”
- 64K externes RAM mit Schreibschutz für den ROM Bereich
- VGA/TV(PAL) über Jumper einstellbar
- Ca. 100% Originalgeschwindigkeit bei TV-Ausgabe, 85% bei VGA
- PS2-Tastatur zur Eingabe, derzeit wird nur DE Layout unterstützt
- Eigenes Tape-orientiertes Dateisystem mit Image-Datein und Installationsprogramm
- Maschinensprache-Monitor mit Breakpoint
- vorbereitet für externes Kempston-kompatibles Interface

3 Konzept

Kernstück des Ganzen ist ein ATmega 644(P) dazu kommen noch ein CPLD als Adresslatch und Video-Shifter sowie 2 mal 32Kx8 SRAM, die restliche Hardware besteht aus passiven Bauelementen und Steckverbindern. Das Videosignal wird im Timing-Zusammenspiel zwischen MCU, CPLD und RAM erzeugt. Da während der Bildschirmausgabe keine Z80-Emulation stattfindet, ist eine Soundausgabe von starken Nebengeräuschen geprägt. Aus diesem Grunde habe ich nach einigen Versuchen letztendlich darauf verzichtet.
Das Videotiming erzeugen Timer 1 und Timer 2, Timer 0 generiert den Pixeltakt. Die PS2-Tastatur wird über den USART0 des ATMega angeschlossen, an die SPI-Schnittstelle kann eine SD-Karte als Datenspeicher angeschlossen werden.

3.1 Systemvoraussetzung Host für das Assemblieren des AVR Quellcodes

Das Programm wurde unter Linux mit dem AVRA Assembler übersetzt. Eventuell muß der Pfad zu den Include-Dateien (bei mir **/usr/local/include/avr**) angepasst werden. Die Hex-Files sollte sich aber auch unter anderen Betriebssystemen brennen lassen.

3.2 Systemvoraussetzung Host für das Übersetzen des CPLD Quellcodes

Der VHDL-Quellcode wurde unter Linux mit dem XILINX WebPack und meinem Tool **vhdl2cpld** übersetzt. Das dazugehörige Kommando wäre
vhdl2cpld -speed -noburn
Wenn das Ergebnis gleich ind CPLD programmiert werden soll, muss das -noburn weggelassen werden. Das JEDEC-File sollte sich aber auch unter anderen Betriebssystemen brennen lassen.

3.3 Systemvoraussetzung Host für das Übersetzen der Tool-Sourcen

Normalerweise reicht ein installierter GCC auf einem Linux-system aus.

3.4 Tools zur Verwaltung der Images

Die im **tool**-Verzeichnis des Binärpaketes befindlichen Programme sind zum Erstellen des Dateisystems sowie zum Lesen und Schreiben von und auf das Image notwendig. Sinnvollerweise sollten sie nach /usr/local/bin kopiert werden. Alternativ befinden sich im **package** Verzeichnis ein rpm- und ein deb-Paket, über die sich die Programme ebenfalls installieren lassen.

3.5 Das AX82 Dateisystem auf SD-Karte

Für die aktuelle Version habe ich das Dateisystem an das des AX81b angepasst. Da das FAT Dateisystem aber teilweise patentbehaftet ist, plane ich für zukünftige Versionen und Projekte ein eigenes Dateisystem mit entsprechendem Partitionstyp, welches sich auch leichter auf Flash-bausteine übertragen lassen soll.

3.5.1 Datenformat auf der SD-Karte

Für das Dateisystem ist zwingend mindestens eine FAT16 Partition auf dem Mediuum notwendig, Medien ohne Partitionstabelle oder fehlender FAT16 Partition werden nicht unterstützt.
Die Programme liegen in in einer Image-Datei im Hauptverzeichnis, die Größe kann bei der Installation festgelegt werden.

Jedes Image hat einen 512 Bytes großen Header-Sektor, in dem z.B. die Anzahl der Tapes vermerkt ist. Danach folgen die Programme der einzelnen Tapes lückenlos hintereinander. Ein Programm besteht aus einem Header-Block (512 Bytes) und dem Datenblock von 48 KBytes. Es werden grundsätzlich Snapshosts gespeichert. Im Headerblock stehen der Programmname und danach 2 Bytes Dateilänge. Ab Byte 256 liegen die für den Snapshot relevanten Daten (Registerinhalte etc.).
Der Übersichtlichkeit halber sind je 32 Dateien zu (rein logisch) einem Tape zusammengefasst, von denen es maximal 256 Stück geben kann. Rein rechnerisch können so maximal 8192 Programme auf einer SD-Karte abgelegt werden und belegen dort ca.400 MBytes. Es werden SD, SDHC und MMC-Karten (nicht getestet) unterstützt. Zusätzlich gibt es einen Unterordner in dem sich weitere Unterordner entsprechend der Anzahl der Tapes befinden. Über diese Ordner erfolgt der ”Datenaustausch” mit PC-Systemen.

3.5.2 das Dateisystem initialisieren

Dazu sollte die SD-Karte frisch formatiert sein. Für meine Methode der Vorbereitung einer SD-Karte wird eine Root-Shell benötigt. Ich habe versucht, die folgenden Anweisungen möglichst allgemein zu halten, benutze aber selbst nur OpenSuSE. Erfahrungsberichte oder Anleitungen unter anderen Distributionen werde ich aber gerne mit einarbeiten.
Über den Desktop wird ein Terminal geöffnet und anschließend zum User ”Root” mit dem Kommando **su** und nachfolgender Passworteingabe gewechselt.
Zunächst hängt man das SD-Karten Laufwerk ein, indem man es über den Desktop öffnet wie wenn man z.B. Daten von dort kopieren will. Danach sucht man nach dem zuletzt eingebundenen Laufwerk, indem man den Befehl **cat /etc/mtab** in der Root-Shell eingibt. Dort müsste dann als letzter Eintrag etwas so ähnlich wie:

```
/dev/mmcblk0p1 /media/E945-1FF8 vfat ...
```

stehen. Wichtig ist in dem Fall das ”/dev/mmcblk0”, das p1 zeigt nur an, dass es sich um die erste Partition auf dem Medium handelt. Nun sollte das Laufwerk vom Desktop aus wieder ausgeworfen werden, zur Kontrolle kann /etc/mtab nochmals angesehen werden, die entsprechenden Einträge sollten nicht mehr existieren.
Der nächste Schritt ist die Erstellung einer passenden Partitionstabelle. Dazu geben wir den Befehl **fdisk /dev/mmcblk0** ein, fdisk sollte sich mit

```
Befehl (m für Hilfe):
```

melden. Als Befehl geben wir einfach ein ”**p**” (print partition table) ein. In der nachfolgenden Ausgabe interessiert nur die Partitionstabelle am Ende, die z.B. so aussehen kann

```
        Gerät  boot.     Anfang        Ende     Blöcke   Id  System
/dev/mmcblk0p1            2048      985087      491520    b  W95 FAT32
```

Ist unter den Partitionen mindestens eine mit der ID 6 (FAT16), so können wir die Root-Shell schließen und die vorhandene Partition einfach nutzen.
Im dargestellten Fall ist dies nicht so, aber auch nicht weiter kompliziert. Wenn bereits mindestens eine Partition angelegt

ist, kann der nächste Schritt übersprungen und einfach der Partitionstyp geändert werden. Wenn sich keine Partition auf dem Medium befindet, muss mindestens eine angelegt werden. Das geschieht dann mit dem Befehl "**n**", danach wählt man "p" für primäre Partition. Danach werden noch die Partitionsnummer ("1"), Startsektor (nur mi ENTER bestätigen) und Partitionsgröße (z.B. **+256M**) abgefragt. Mit dem Befehl "**t**" muss noch der Typ der Partition geändert werden, als Hexcode wird für FAT16 eine "6" eingegeben werden. Freier Speicher auf der Karte lässt sich mit weiteren, nicht für den AX81 genutzten Partitionen füllen.
Nun kann mittels des Kommandos "**w**" die neue Partitionstabelle auf die Karte geschrieben werden, danach beendet sich fdisk automatisch. Als letzter Schritt muß nun die Partition noch formatiert werden. Dazu gibt man **mkdosfs -F 16** gefolgt von der zu formatierenden Partition (hier: /dev/mmcblk0p1) ein, im Beispiel wäre das

mkdosfs -F 16 /dev/mmcblk0p1

Das "-F 16" sorgt für eine FAT16-Struktur, wenn der Vorgang abgeschlossen ist, sollte die Karte in den AX81 gesteckt und das System gestartet werden. Es sollte nun der Kartentyp und "NO IMAGE FOUND" angezeigt werden. Danach die Karte wieder zurück in den PC, denn es fehlen ja noch die Image-Dateien und Verzeichnisse.
Dazu dient das Programm **ax82-fsinstall** welches einfach im Hauptverzeichnis der Karte gestartet wird. Als Parameter müssen die Anzahl der Tapes für die verschiedenen Emulationen angegeben werden. Wenn das Programm ohne Parameter gestartet wird, wird eine Kurzbeschreibung ausgegeben.

ax81-fsinstall 10 10 5 5

erzeugt z.B. je 10 Tapes für ZX81, ZX80 und je 5 Tapes für Jupiter ACE und ZX Spectrum. Für jeden der 4 Parameter sind Werte zwischen 0 und 255 zulässig, wobei beim Wert 0 das betreffende Image nicht angelegt wird. Weiterhin sollte die Summe aller Tapes nicht größer als die Hälfte der Partitionsgröße (im Megabytes) sein, da dann etwa die Hälfte der Partition durch die Image-Dateien belegt wird und der Rest für die zu kopierenden Programme frei bleibt.

3.5.3 Dateien auf SD-Karte schreiben

Dazu dient das Programm **ax82-write** aus dem "tool' Verzeichnis. Für die einzelnen Emulationen gibt es nur noch ein Programm, der Betriebsmodus wird über den Verzeichnisname beim Aufruf vorgegeben. Es schreibt alle (maximal 32) im aktuellen Verzeichnis befindlichen Files auf die freien Plätze des dem Unterverzeichnisname entsprechenden Tapes. Vor dem Schreiben wird das Tape formatiert (alle Programme darin gelöscht). Es werden ur Snapshots (Endung .z80) akzeptiert.

Als Vorgehensweise empfehle ich, die zu schreibenden Dateien in das zu wählende TAPExxx Verzeichnis zu kopieren, dann in diesem Verzeichnis ein Terminal zu starten und dort den Befehl **ax82-write** einzugeben.

3.5.4 Dateien von SD-Karte lesen

Dazu dient das Programm **ax82-read** aus dem "tool' Verzeichnis. Es liest alle im (dem Unterverzeichnis entsprechenden) Tape vorhandenen Programme aus. Bestehende Programme werden ohne Nachfrage überschrieben. Auch hier ist es am einfachsten, in das TAPExxx Verzeichnis zu wechseln, hier ein Terminal zu starten und dort den Befehl **ax81-read** einzugeben. Zur Zeit ist das Programm noch unvollständig, ACE und Z80 Dateien können nicht zurückgelesen werden.

3.5.5 Nutzer propietärer Betriebssysteme

können sich das vorbereitete Image herunterladen und in das Wurzelverzeichnis der ersten FAT16-Partition auf der SD-Karte kopieren. Ausserdem sollte sich der Quellcode der Tools nach kleineren Modifikationen auch mit dem MinGW Compiler übersetzen lassen.

4 Was noch fehlt

Das Projekt ist nur bedingt vollständig. Neben sicher fälligen Bugfixes werden auch Teile der Funktionalität und der Dokumentation erst im Laufe der Zeit fertig werden.
Die folgende Liste stellt keine verbindliche Aufstellung dar, vieles hängt auch vom notwendigen Zeitaufwand und verfügbaren Zeitrahmen ab.

- Liste der lauffähigen Programme
- Rückleseprogramm für SD-Karte
- Joystick-Interface (externer Controller und Firmware-Erweiterung)

5 Changelog

28.9.2012 Version 0.32

- erste öffentliche Version