Retro-Computing
Simulation – Emulation – Projekte
“Exotic Flavor”

Impressum

Author: Peter Sieg

Rabishauerstr. 9
37603 Holminden
Germany

Stand: Dezember 2015

Mit der Freigabe unter einer Creative Commons Lizenz verbindet sich der Wunsch, meine Arbeit der Community zurück zu geben in der Hoffnung, das es dort weiter gepflegt und ergänzt wird und somit zu einem aktuellen Nachschlagewerk weiter wächst.
Frei nach dem ersten Star Trek Film – V'ger :-)

Inhaltsverzeichnis

Vorwort

Durch mein erstes Buch Commodore-Hardware-Retrocomputing hatte ich bei Streifzügen durch die Weiten des Internet auch viele Projekte kennengelernt, die nichts mit Commodore und auch nichts direkt mit originalen Homecomputer und deren Erhaltung zu tun hatten.

Von einigen dieser Projekte geht es in diesem Buch.

Bei den entdeckten Projekten handelt es sich nicht nur um Hardwareprojekte, sondern auch um reine Softwareprojekte in Form von Simulationen und Emulatoren.

Auch davon stelle ich einige in diesem Buch vor, die ebenfalls etwas exotischere Systeme durch Emulation auf modernen Windows/Linux/Mac OS X zum Ausprobieren zur Verfügung stellen.

Ja. Hier geht es nicht um den sog. Mainstream! Also nicht um VICE zum Emulieren von CBM, PET, C64 & Co oder UAE der Universal Amiga Emulator etc. pp.

Nein hier wird es um SIMH, Xerox Alto, Apple Lisa, Kenbak, Transputer, Cray, Vectrex und spezielle Hardware Projekte gehen.

Dabei geht es nicht um historisch korrekte Beschreibungen, nicht um exakte Nachbauanleitungen, sondern eher ein Streifzug durch Systeme, die ich während meiner Internetrecherchen kennen und lieben gelernt habe.

Es ist eine Einladung zum Nachmachen, Mitmachen, Freude haben beim Lesen und Spielen mit den Informationen und den Systemen. Dabei soll es keine exakten Schritt für Schritt Anleitungen geben, sondern eher ein erster Einstieg soll erreicht werden. Zu jedem System/Emulator gibt es dann noch viel mehr selbst zu entdecken!

Leider benötigen viele Emulatoren original Romdateien, die größtenteils noch Copyrights unterliegen. Hier sollte man also im Besitz eines Originalsystems sein. In einigen Fällen wurden die Dateien aber inzwischen auch freigegeben, was den Einsatz von Emulatoren doch deutlich vereinfacht.

Das Internet ist volatil. D.h. evtl. sind Links inzwischen nicht mehr gültig, Webseiten umgezogen. Dann sollte durch eine Suche nach geeigneten Stichworten, die genannten System & Projekte wieder zu finden sein.

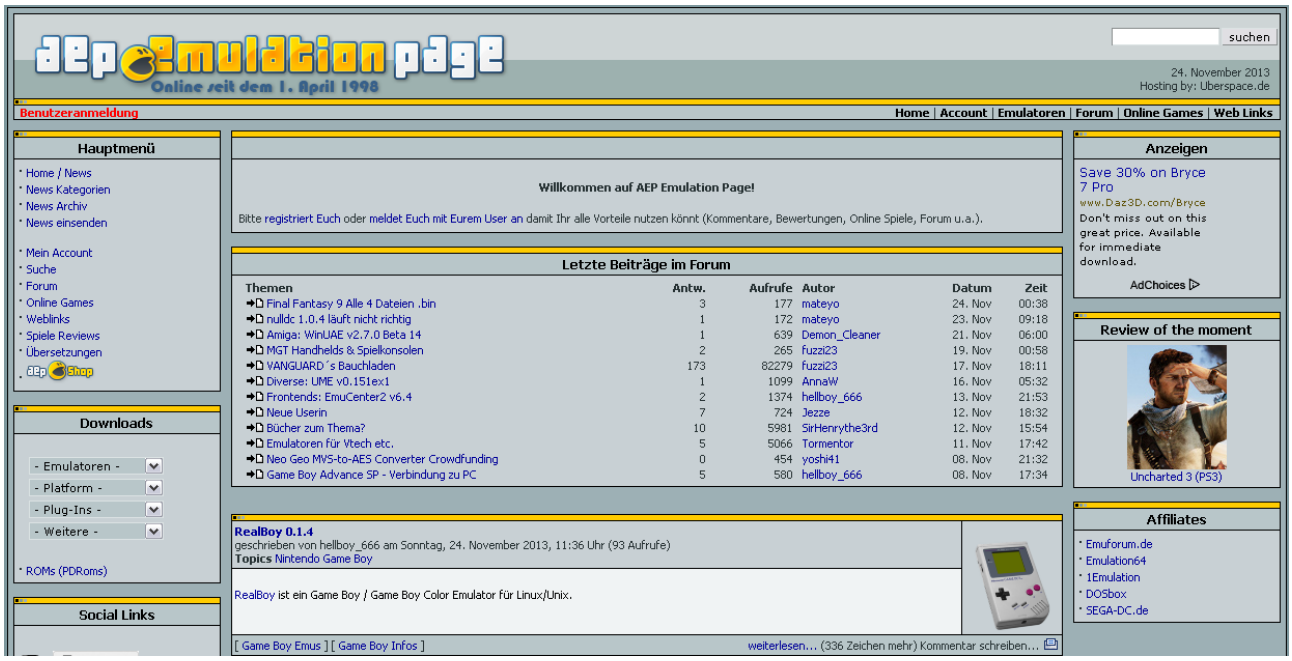
Im ersten Teil werde ich einige Anlaufstellen zu Emulatoren und Foren nennen.

http://petersieg.bplaced.com

Für meine Familie: Heike, Robin und Janis.
Holzminden, August 2015.

Allgemeine Anlaufstellen

AEP Emulation Webseite



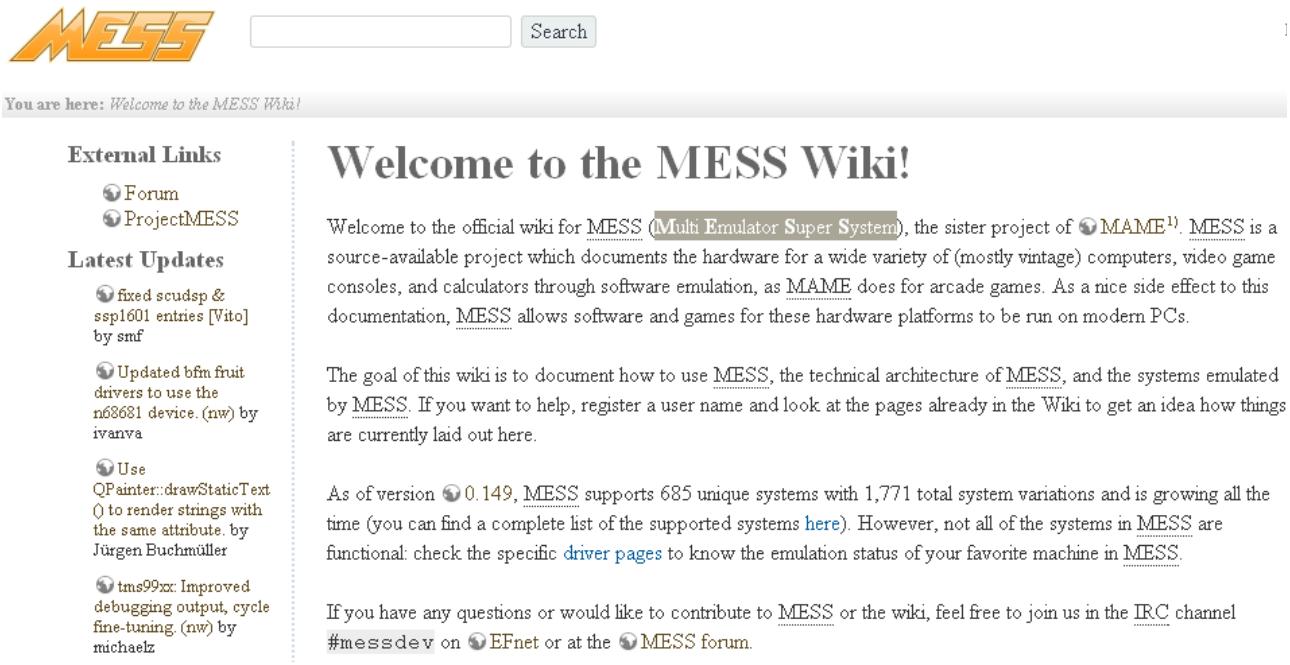
Hier gibt es zu so ziemlich fast jedem System einen oder mehrere Emulatoren für unterschiedlichste Hostsysteme. Es macht einfach Spaß hier zu stöbern und auszuprobieren.

Über suchen:

kann man die verfügbaren/bekannten Emulatoren für ein gesuchtes System anzeigen lassen.

MESS multi-emulator-super-system

Links:
http://www.mess.org/



MESS emuliert unter anderem folgende Systeme:

Sega 32x, Panasonic 3DO, Atari A2600, Atari A5200, Atari A7800, Commodore Amiga
Apple I, Apple II, Atari 8-bit, Acorn BBC Micro / Master, Atari ST / STE
Commodore C16/Plus4, Commodore C64
Coleco Colecovision, Amstrad / Schneider CPC
Diverse / Misc, Sega Dreamcast
Nintendo Game Boy, Nintendo Game Boy Advance, Nintendo Game Boy Color, Nintendo SNES
Sega Game Gear, Sega Genesis / Mega Drive, Sega Saturn, Sega CD
Gamepark GP32, Mattel Electronics Intellivision
Atari Jaguar, Japanische Systeme, Atari Lynx
Apple Macintosh, Sega Master System
MSX, Nintendo N64, Nintendo NES
SNK NeoGeo, SNK NeoGeo CD, SNK NeoGeo Pocket
IBM PC, NEC PC-Engine, Sony Playstation
MB Vectrex, Nintendo Virtual Boy, Bandai Wonderswan, Sinclair ZX Spectrum

MAME - Multiple-Arcade-Machine-Emulator

Links:
http://www.mamedev.org/

Emulation vieler Arcade Automaten und Spiele. Aber auch z.B. Grundlage des später noch vorgestellten P8000 Emulators.

Emulationen in Java Script

Link:
http://www.cambus.net/emulators-written-in-javascript/

Java Script wird auf aktuellen Browsern/Betriebssystemen ohne Installationen ausgeführt und daher sind darauf basierende Emulationen ideal zum Ausprobieren und Rumspielen. **Frederic Cambus** hat auf seiner Seite die z.Z. (2014) bekannten Emulationen in Java Script zusammengestellt. Bitte dabei beachten, das Java Script Emulationen ggf. einen schnellen Rechner benötigen und meistens im Funktionsumfang auch eingeschränkt sein können. Auszug:

Amstrad

- http://www.cpcbox.com/ - Amstrad CPC emulator in JavaScript
- http://roland.retrolandia.net/ - An Amstrad CPC emulator written in JavaScript

Apple

- http://www.scullinsteel.com/apple2/ - An Apple][Emulator in JavaScript

- http://porkrind.org/a2/ - A fast, WebGL optimized Apple][+ emulator.
- http://www.megidish.net/apple2js/ - A JavaScript emulator for the Apple II

Atari

- https://estyjs.azurewebsites.net/ - A pretty fast and functional JavaScript Atari ST emulator (Source)
- http://zerstoerung.de/jsa8e/ - JavaScript version of the A8E Atari 800 XL Emulator

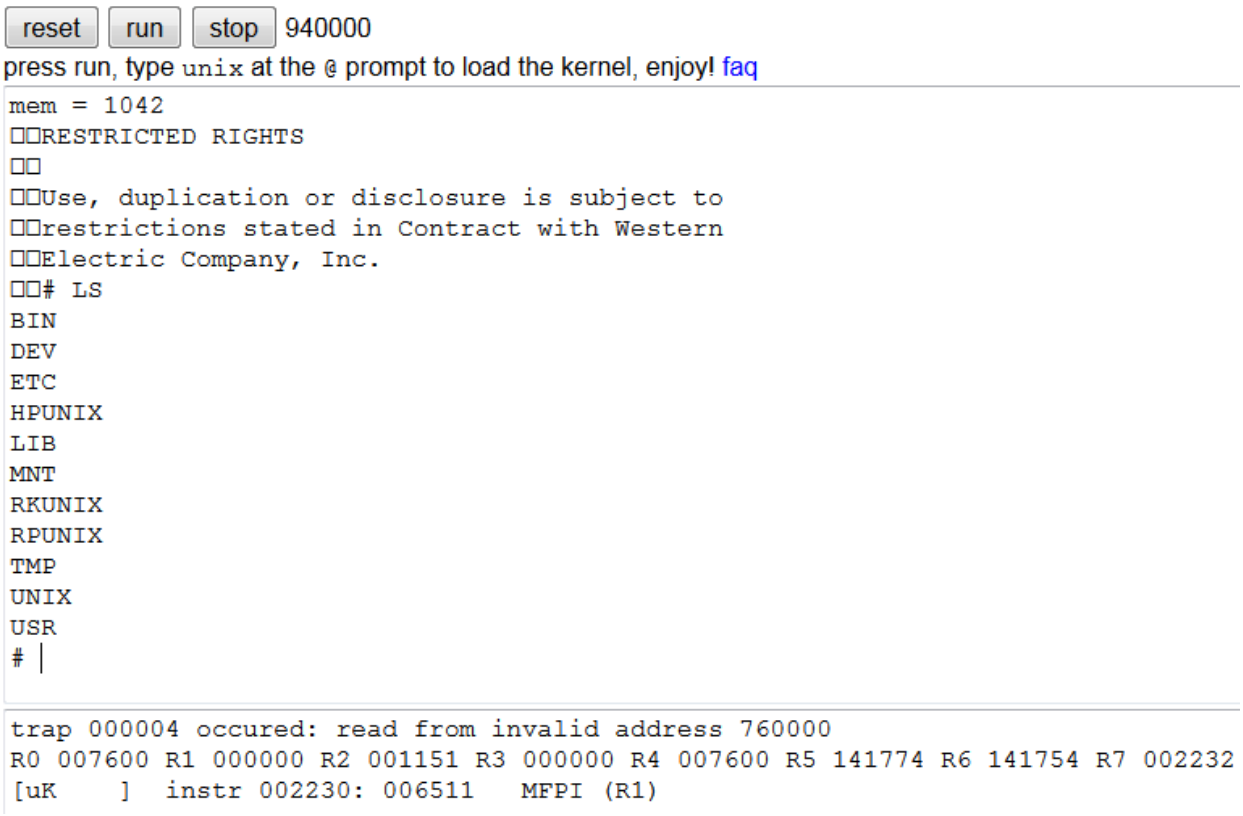
Commodore

- http://www.kingsquare.nl/jsc64 - Commodore 64 emulator written in JavaScript (Source)

- http://scriptedamigaemulator.net/ - Scripted Amiga Emulator (Source)
- http://mdawson.net/vic20chrome/vic20.php - JavaScript VIC-20 emulator
- http://retroplay.co/ - Versatile Commodore Emulator for JavaScript (Source)

DEC

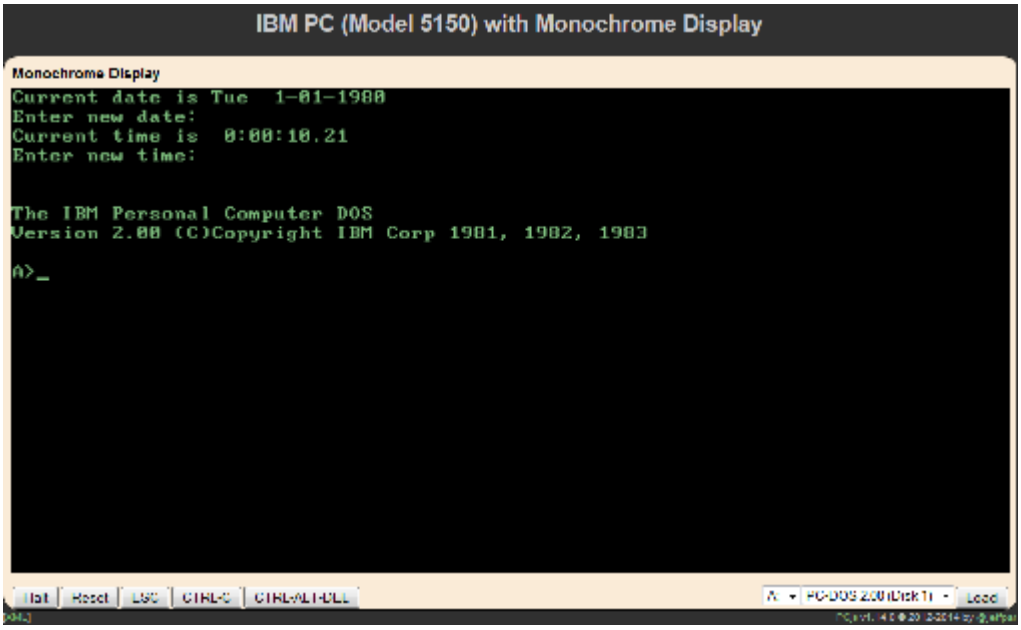
- http://pdp11.aiju.de/ - A JavaScript PDP-11 emulator



- https://pdp11-js.googlecode.com/git/unixv6.html - PDP-11 emulator with UNIX V6

PC Emulators

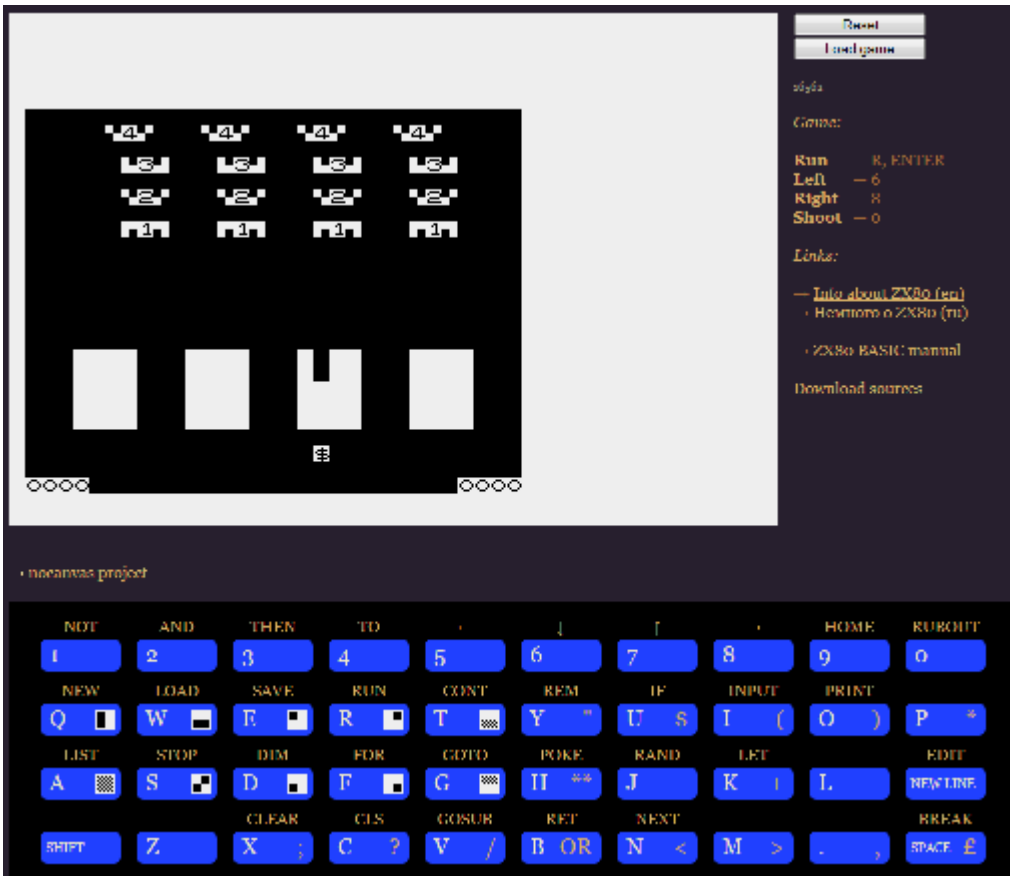
- http://bellard.org/jslinux/ - JavaScript PC emulator



- http://jsdosbox.sourceforge.net/ - JavaScript PC DOS emulator (Source)
- http://www.pcjs.org/ - IBM PC Model 5150 emulator
- http://copy.sh/v24/ - An x86 emulator written in JavaScript (Source)
- http://jemul8.com/ - An object-oriented JavaScript x86 emulator for Node.js and the Browser (Source)

Sinclair

- http://jsspeccy.zxdemo.org/ - A ZX Spectrum emulator in JavaScript (Source)
- http://torinak.com/qaop - ZX Spectrum emulator
- http://nocanvas.zame-dev.org/0004/ - JavaScript ZX80 Emulator



- http://jbacteria.retrolandia.net/ - The smallest JavaScript Spectrum emulator

Tandy

- http://mc-10.com/ - Emulator for the TRS-80 MC-10 microcomputer
- http://people.cs.ubc.ca/~pphillip/trs80.html a Javascript emulator for the TRS-80 Model III
- http://jtandy.retrolandia.net/ - Another JavaScript TRS-80 emulator

Miscellaneous

- http://jsmess.textfiles.com/ - The JavaScript MESS (Multi Emulator Super System) (Source)
- http://www.hampa.ch/pce/ - PC emulators in JavaScript (Atari ST, IBM PC 5150, Macintosh, RC759 Piccoline)
- http://www.tramm.li/i8080/ - Emulates a minimal Intel 8080 Microcomputer that runs CP/M
- http://bbc.godbolt.org/ - JavaScript BBC Micro emulator (Source)
- http://www.twitchasylum.com/jsvecx/ - JavaScript port of the VecX Vectrex emulator
- http://www.nanowasp.org/ - A MicroBee emulator
- http://s-macke.github.io/jor1k/ - OpenRISC OR1K JavaScript emulator running Linux with network support (Source)
- http://jsmsx.sourceforge.net/ - The first MSX emulator 100% written in JavaScript
- http://jupiler.retrolandia.net/ - Jupiter Ace emulator written in JavaScript

Universeller Assembler

Links
http://john.ccac.rwth-aachen.de:8000/as

Für viele Retro-Computer Systeme möchte man ggf. auch selbst mal kleinere Routinen und Programme schreiben. Dazu ist vielfach auf der fast untersten Ebene ein Assembler nötig.

Einen davon der eine Vielzahl an CPU unterstützt ist der Arnold Assembler von Alfred Arnold.

Der Makro Assembler ist für Unix Systeme, DOS, Windows und nun auch für Mac verfügbar.

Er unterstützt eine Vielzahl von Target CPU's:

- Motorola 68000..68040,683xx inkl.Koprozessor und MMU
- Motorola ColdFire
- Motorola DSP5600x,DSP56300
- Motorola/IBM MPC601/MPC505/PPC403/MPC821
- Motorola M-Core
- Motorola 6800, 68(HC)11(K4) sowie Hitachi 6301
- Motorola/Freescale 6805, 68HC(S)08
- Motorola 6809 / Hitachi 6309
- Motorola/Freescale 68HC12(X) inklusive XGATE
- Freescale 68RS08
- Motorola 68HC16
- Hitachi H8/300(H)
- Hitachi H8/500
- Hitachi SH7000/7600/7700
- Rockwell 6502 und 65(S)C02
- CMD 65816
- Mitsubishi MELPS-740
- Mitsubishi MELPS-7700
- Mitsubishi MELPS-4500
- Mitsubishi M16
- Mitsubishi M16C
- Intel 4004/4040
- Intel MCS-48/41
- Intel MCS-51/251, Dallas DS80C390
- Intel MCS-96/196(Nx)/296
- Intel 8080/8085
- Intel i960
- Signetics 8X30x
- Signetics 2650
- Philips XA
- Atmel (Mega-)AVR
- AMD 29K
- Siemens 80C166/167
- Zilog Z80, Z180, Z380
- Zilog Z8, eZ8
- Xilinx KCPSM/KCPSM3 ('PicoBlaze')
- LatticeMico8
- Toshiba TLCS-900(L)
- Toshiba TLCS-90
- Toshiba TLCS-870
- Toshiba TLCS-47
- Toshiba TLCS-9000
- Microchip PIC16C54..16C57
- Microchip PIC16C84/PIC16C64

- Microchip PIC17C42
- SGS-Thomson ST6
- SGS-Thomson ST7
- SGS-Thomson ST9
- SGS-Thomson 6804
- Texas Instruments TMS32010/32015
- Texas Instruments TMS3202x
- Texas Instruments TMS320C3x
- Texas Instruments TMS320C20x/TMS320C5x
- Texas Instruments TMS320C54x
- Texas Instruments TMS320C6x
- Texas Instruments TMS9900
- Texas Instruments TMS7000
- Texas Instruments TMS370xxx
- Texas Instruments MSP430
- National Semiconductor SC/MP
- National Semiconductor INS807x
- National Semiconductor COP4
- National Semiconductor COP8
- National Semiconductor SC144xx
- Fairchild ACE
- NEC æPD78(C)1x
- NEC æPD75xx
- NEC æPD75xxx (alias 75K0)
- NEC 78K0
- NEC 78K2
- NEC æPD7720/7725
- NEC æPD77230
- Fujitsu FýMC8L
- Fujitsu FýMC16L
- Symbios Logic SYM53C8xx (ja, die kann man programmieren!)
- Intersil CDP1802/1805
- XMOS XS1

SIMH: Forward... Into The Past

Links:

http://simh.trailing-edge.comSoftware

Was ist SIMH

SIMH ist ein hoch portables Framework um historische Computersysteme zu simulieren.
SIMH läuft auf:

- X86 Linux,NetBSD,OpenBSD, FreeBSD (gcc)
- X86 Windows 95, Windows 98, Windows 2000, Windows XP (Visual C++ orMingW)
- WindowsCE
- Mac OS/9 (Codewarrior) or OS/X (Apple Development Tools)
- Sun Solaris (gcc)
- HP/UX (gcc)
- AIX (gcc)
- Alpha Unix (DEC C)
- VAX/VMS, Alpha/VMS, IA64/VMS (DEC C)
- OS/2 (gmx)

SIMH simuliert folgende Systeme:

- Data General Nova, Eclipse
- Digital Equipment Corporation (DEC) PDP-1, PDP-4, PDP-7, PDP-8, PDP-9, PDP-10, PDP-11, PDP-15, VAX
- GRI-909
- IBM 1401, 1620, 1130, System/3
- Hewlett-Packard (HP) 2116, 2100, 21MX
- Interdata(Perkin-Elmer) 16b, 32b architectures
- Honeywell H316/H516
- MITS Altair 8800, both 8080 and Z80 versions
- Royal-McBeeLGP-30, LGP-21
- Scientific Data Systems SDS-940

Mit SIMH kann man folgende Software betreiben:

- PDP-1 Lisp and DDT, early interactive systems
- PDP-11 Unix V5, V6, V7, the earliest extent releases of Unix (V1 to V4 are lost)
- Interdata7/32 Unix V6, the first port of Unix (and the first port to a 32b system)
- PDP-10 TOPS-10, TOPS-20, ITS
- PDP-11 DOS, RT-11, RSX-11M, RSX-11M+, RSTS/E
- PDP-15 ADSS-15, F/B-15, DOS-15, DOS/XVM
- PDP-8 OS/8, TSS-8, ETOS, DMS
- VAX/VMS, VAX/Ultrix, VAX/BSD, VAX/NetBSD
- Nova RDOS, Eclipse AOS
- HP DOS, RTE-III, RTE-IV
- MITS Altair CP/M, DOS
- /3 SCP, CMS

Warum

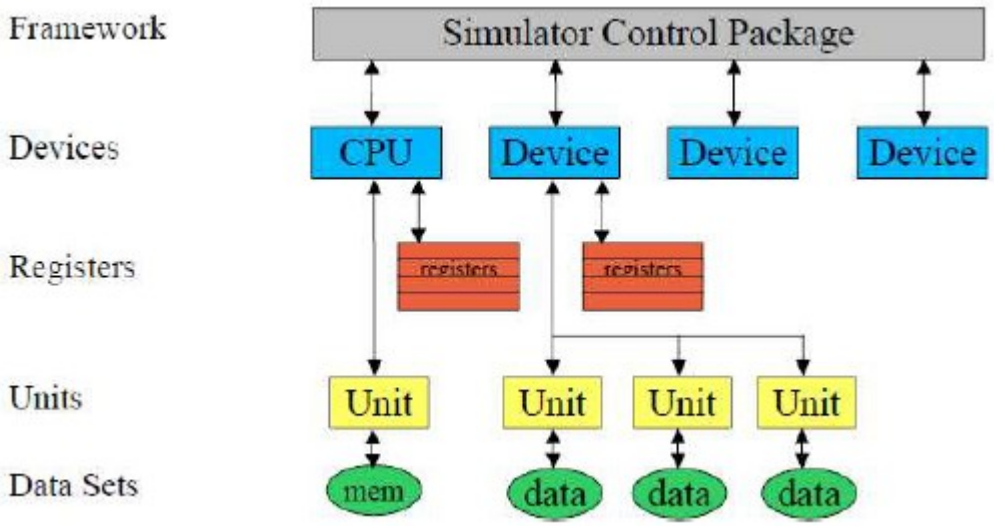
- Noch funktionierende Computer Hardware aus der Vergangenheit wird immer weniger, bis auch das letzte System seinen Dienst eines Tages versagt.
- Software und Dokumentation droht verloren zu gehen.
- Pioniere der vergangenen Tage sterben.

Ziele

- Zur Verfügungstellung von historischen Computern und Software für alle Interessierten ohne große Kosten.
- Simulation anstatt fast nicht mehr zu beschaffende, teure und möglicherweise defekte original Hardware
- Hoch portabel und weite Verfügbarkeit

SIMH Design

Simulatoren bestehen aus Devices (eine CPU ist z.B. ein solches Device, das Befehle ausführt) Devices enthalten benannte Register, welche die Statusinformation beinhalten und Units, welche letztlich Daten beinhalten)

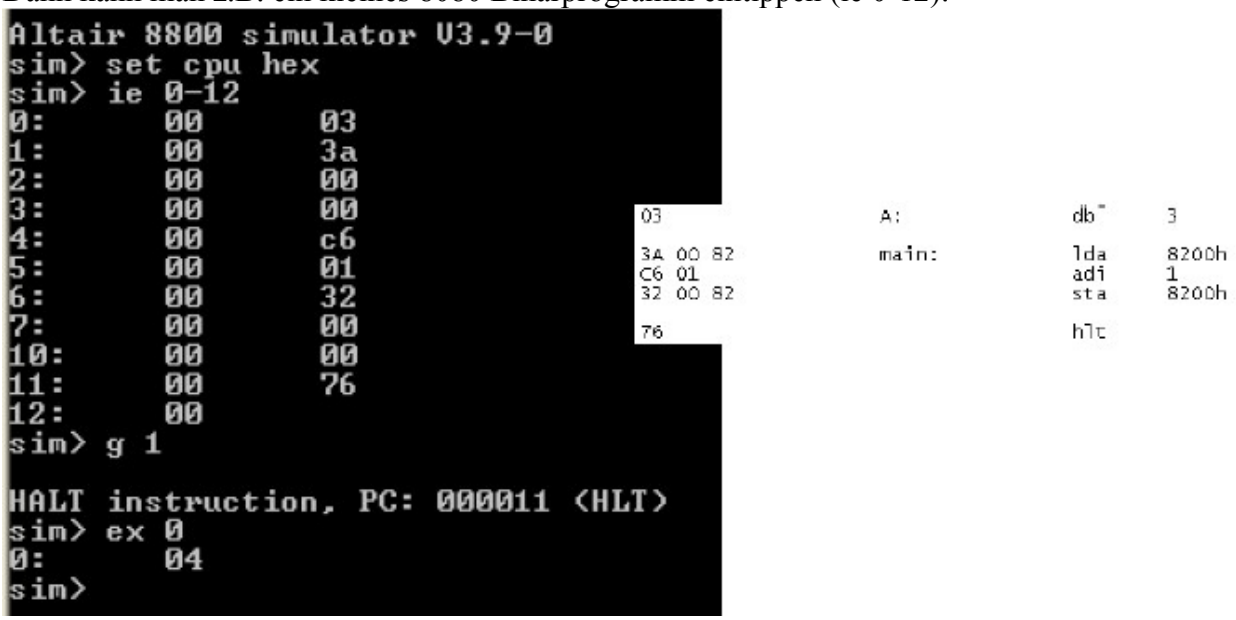


SIMH – 8080 bare bone

SIMH dient war primär dazu ganze Systeme inkl. Software Umgebung zu simulieren, aber man kann auch minimalst, quasi auf Du und Du mit der CPU sprechen.

Dazu Altair.exe starten und am Prompt eingeben:
set cpu hex <return> - So kann man auch Hex Werte eingeben

Dann kann man z.B. ein kleines 8080 Binärprogramm eintippen (ie 0-12):



Mit g 1 für GO ab Adresse 1 wird unser kleines Testprogramm gestartet. An Adresse 11 hält das Programm mit einer HALT (HLT = 76h) an. Das Programm inkrementiert den Inhalt der Adresse 0. Nach jedem g 1 wird der Inhalt also um 1 erhöht.

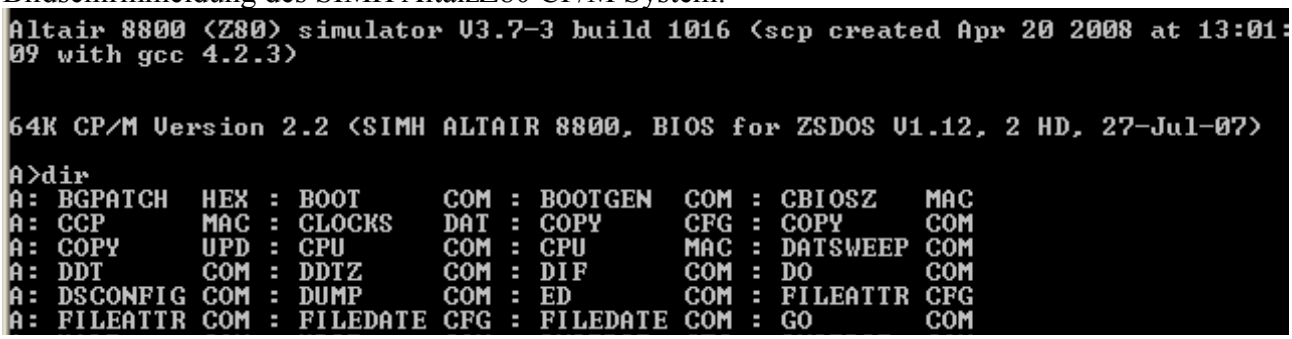
SIMH – Z80 CP/M

Batchdatei z.B. Start.bat erstellen:
altairz80 zsdos

Die Datei zsdos enthält dann z.B.:
d tracks[0-7] 254
attach dsk zsdos.dsk
;attach hdsk i.dsk
attach hdsk CPMDSK_B.IMG
set cpu 64k
set cpu itrap
set cpu z80
set cpu altairrom
set cpu nonbanked
reset cpu
boot dsk

Dabei ist zsdos.dsk die Bootdisk und CPMDSK_B.IMG eine 8MB großes Harddisk Image, das wir etwas später dann bei AVR CP/M wieder sehen werden.

Bildschirmmeldung des SIMH AltaizZ80 CP/M System:

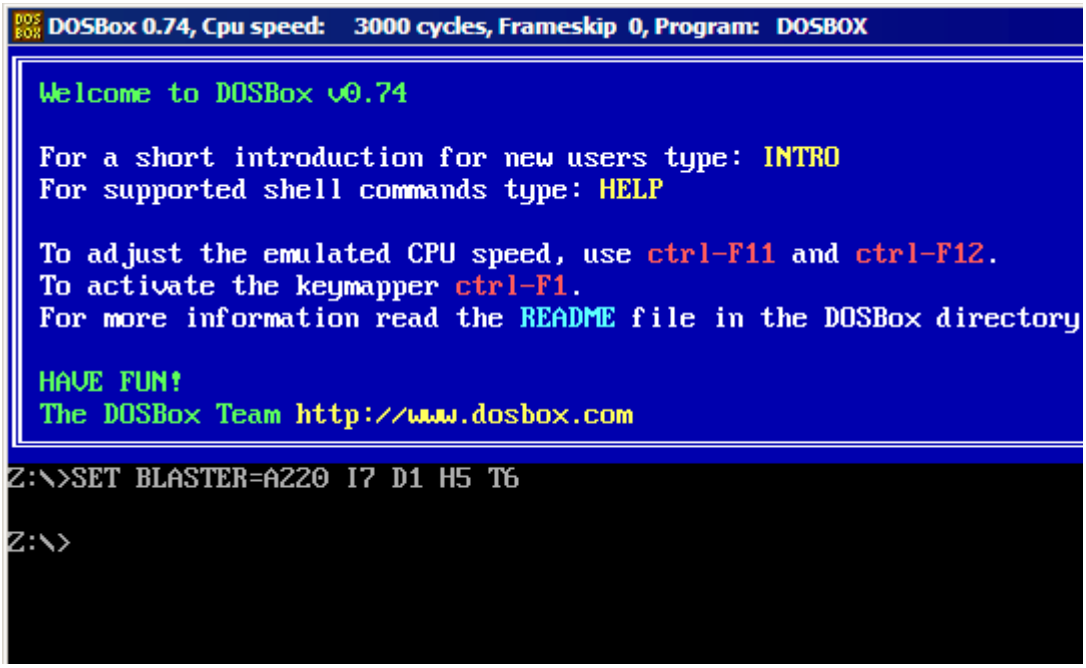


Hier laufen Compile-Vorgänge rasend schnell und mittels R und W kann man Dateien zwischen der Emulationsumgebung und dem Hostsystem austauschen.

DosBox

Link:
http://www.dosbox.com/

DosBox ist eine virtuelle DOS Umgebung. Dabei werden sowohl DOS Infrastrukturen und Aufrufe als auch damalige Hardware (Soundblaster; CGA-VGA) emuliert, ohne das man DOS selbst noch installieren müßte bzw. benötigt. DosBox gibt es für so gut wie alle aktuellen Betriebssystems (Win7, Linux, Mac OS X).



DOS Programme können auf dem Host-System in einem beliebigen Verzeichnis abgelegt werden. Als erstes sollte man dann z.B. den Laufwerksbuchstaben C: auf dieses Verzeichnis einstellen.

Sind also z.B. die DOS Programme alle unter c:\daten\dos abgelegt so gibt man in der Befehlszeile ein:

mount c c:\daten\dos

Damit zeigt nun das virtuelle DOS Laufwerk C: zu diesem Verzeichnis. Mit der Eingabe von

c: <Enter>

wechselt man dann auf dieses virtuelle Laufwerk bzw. in das dahinterliegende Verzeichnis.

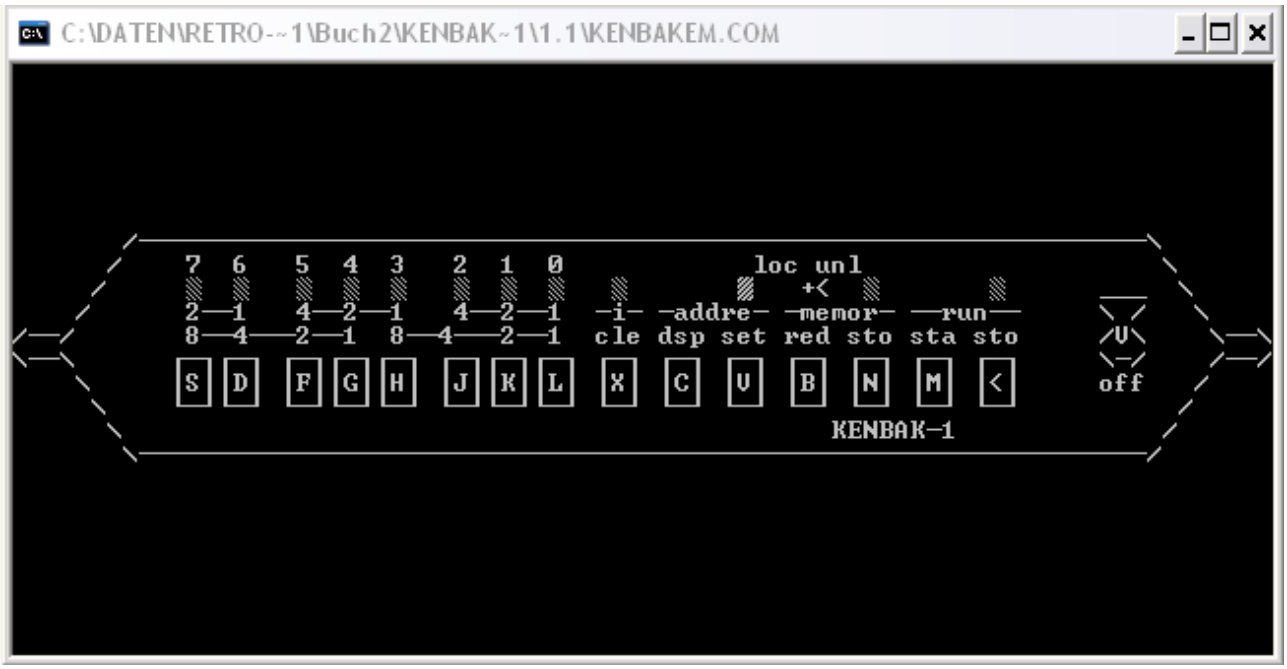
Kenbak-1

Links:
http://www.kenbak-1.net/
http://www.funnypolynomial.com/software/arduino/kenbak.html
http://petersieg.bplaced.net/?Kenbak-1
http://www.neocomputer.org/kenbak/kenbak1-JS.html

Der Kenbak-1 gilt als der erste Personalcomputer der Geschichte. Er wurde um 1971 für 750 US$ verkauft. Sein Aufbau setzte noch keine fertige CPU als 1-Chiplösung ein, denn die gab es so noch nicht, sondern wurde komplett aus TTL Logikbausteinen konstruiert. Das Ram war nur 256 Bytes klein!

Software Emulator

Bildschirm



Emulatortasten

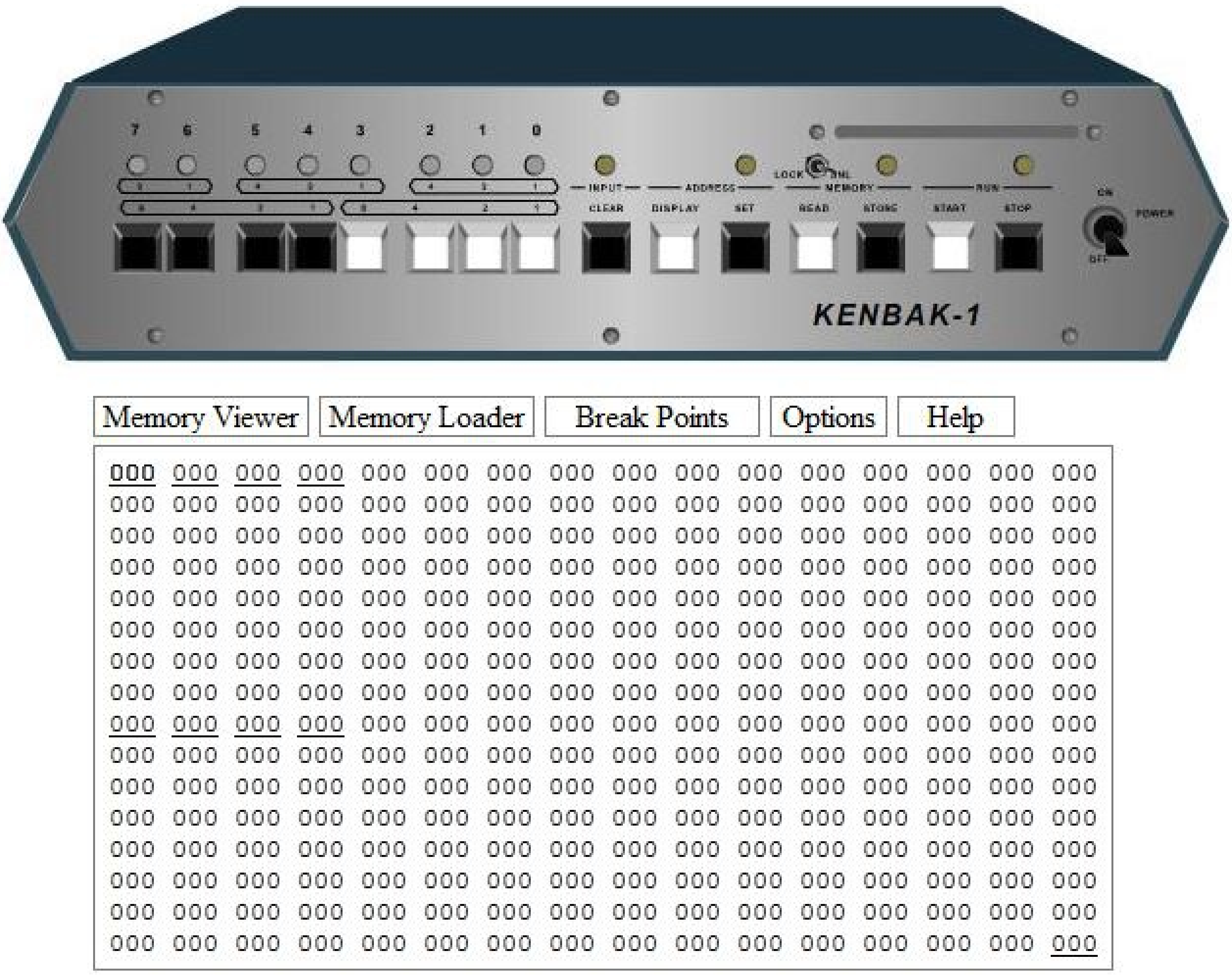
L = Set bit 0 at input register
K = Set bit 1 at input register
J = Set bit 2 at input register
H = Set bit 3 at input register
G = Set bit 4 at input register
F = Set bit 5 at input register
D = Set bit 6 at input register
S = Set bit 7 at input register
X = Clear Input
C = Display adress
V = Set adress
B = Read data
N = Store data
Space = Toggle memory lock/unlock
M = Start
< (or "," on some European KeyBoard's) = Stop
> (or "." on some European KeyBoard's) = Single-Step (as for BIOS Int 16h doesn't return the scancode for more than 1 key at a time)

Programmkontrolle

+	= Faster clock speed
-	= Slower clock speed
Enter	= Restore clock speed to default.
Escape	= Quit
Backspace	= Display/hide memory-dump output (Octal)
F1 - F10	= Load State 1-10
Shift + F1 - F10	= Save State 1-10

Emulator in Java Script

Da Java Script wohl von so jedem aktuellem Browser/Betriebssystem unterstützt wird, ist dieser Emulator ohne jede Installation zu nutzen. Um ihn nutzen zu können, bitte das Einschalten nicht vergessen ☺ (Mit Maus auf Powerswitch klicken).



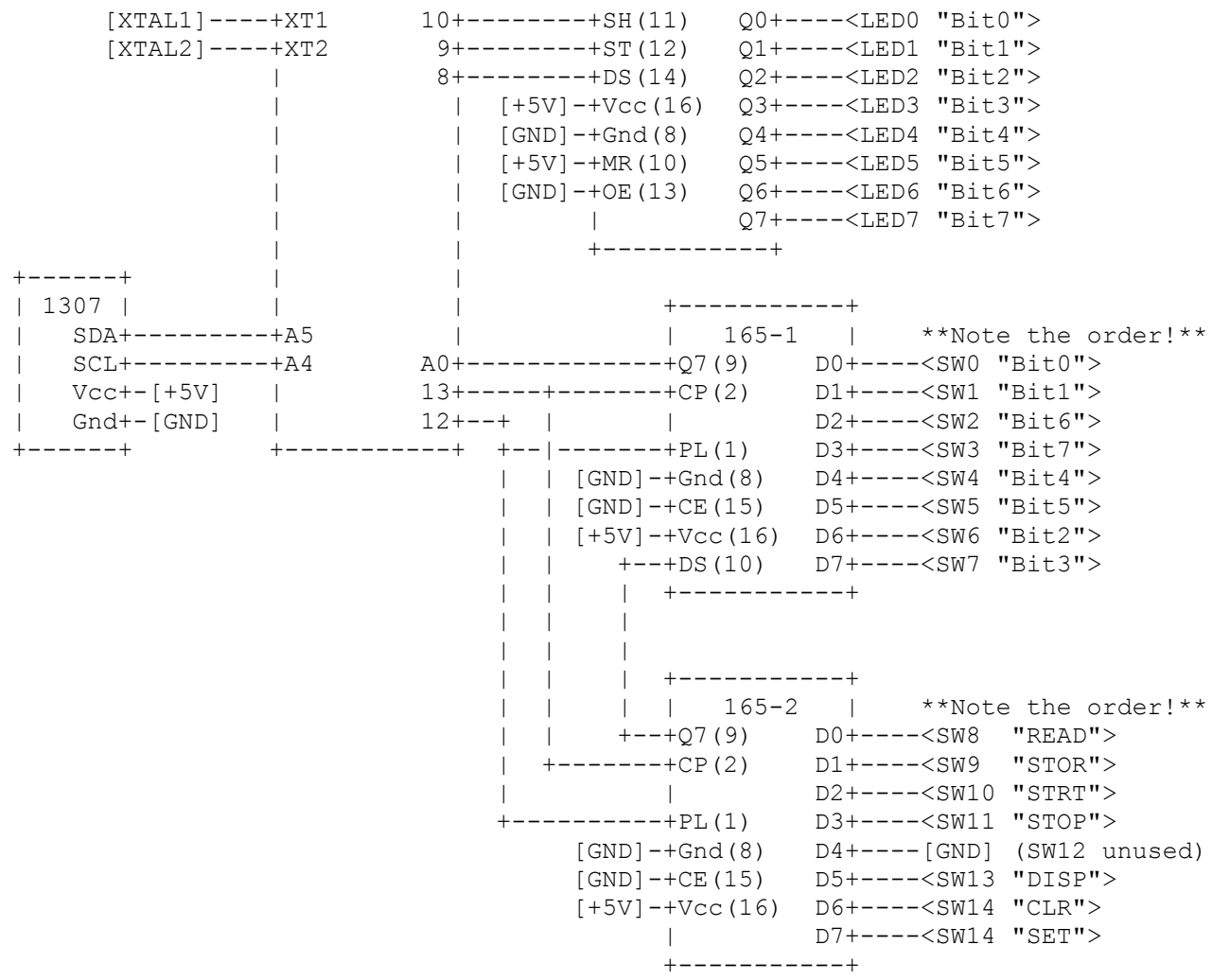
Emulator auf AVR Basis

Entwickelt von Mark Wilson (Danke für die Nutzung von Bild und Textmaterial). Im AVR sind gleich einige fertige Programme enthalten, die auch die zusätzliche Peripherie (Uhr) nutzen.

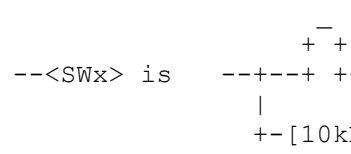
System

Schaltplan als ASCII Grafik

```
This is a *SCHEMATIC*
Pins not listed are unused/floating.
Component list:
  ATMega328, DS1307, 74HC595, 74HC165 (x2),
  LED (x12), Push-button normally open (x15),
  Resistor 220 Ohm (x12), Resistor 10k Ohm (x15),
  16MHz crystal.
                "KENBAK-uino" - Mark Wilson, 2011
                +-----------+
                |    328    |
  <USB>--[+5V]---+Vcc   PWM11+-------<LED11 "RUN">
  <USB>--[+5V]---+AVcc     A3+-------<LED10 "MEM">
  <USB>--[GND]---+Gnd(22)  A2+-------<LED9  "ADDR">
  <USB>--[GND]---+Gnd(8)   A1+-------<LED8  "INP">
  <USB>---[TX]---+TX         |
  <USB>---[RX]---+RX         |        +-----------+
                |            |        |    595    |
```



where:



--<LEDx> is --[220R]--[LED]--[GND]

Altair 8800 und IMSAI 8080

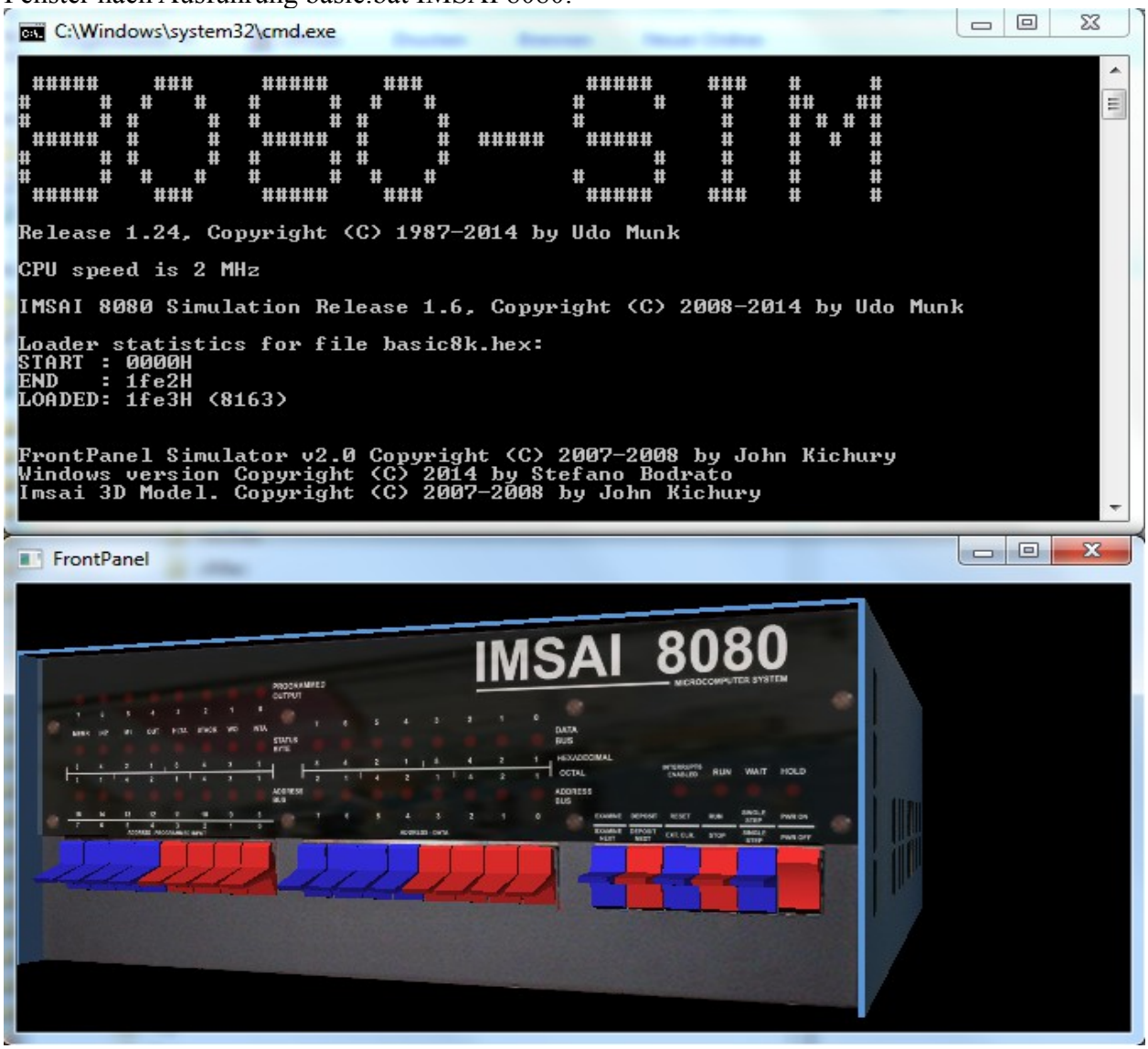
Links:
http://www.autometer.de/unix4fun/z80pack/
http://www.classiccmp.org/cpmarchives/cpm/mirrors/www.unix4fun.org/z80pack/ftp/altair/index.html

z80pack ist ein Zilog Z80 und Intel 8080 Cross Development Paket für UNIX und Windows. Kompletter Sourcecode liegt unter einer BSD Style Lizenz. Dabei kann das Paket sowohl die beiden genannten Systeme über simulierte Schalter und LED's emulieren, als auch komplett ausgestattete CP/M Systeme in einer emulierten Umgebung zur Verfügung stellen.

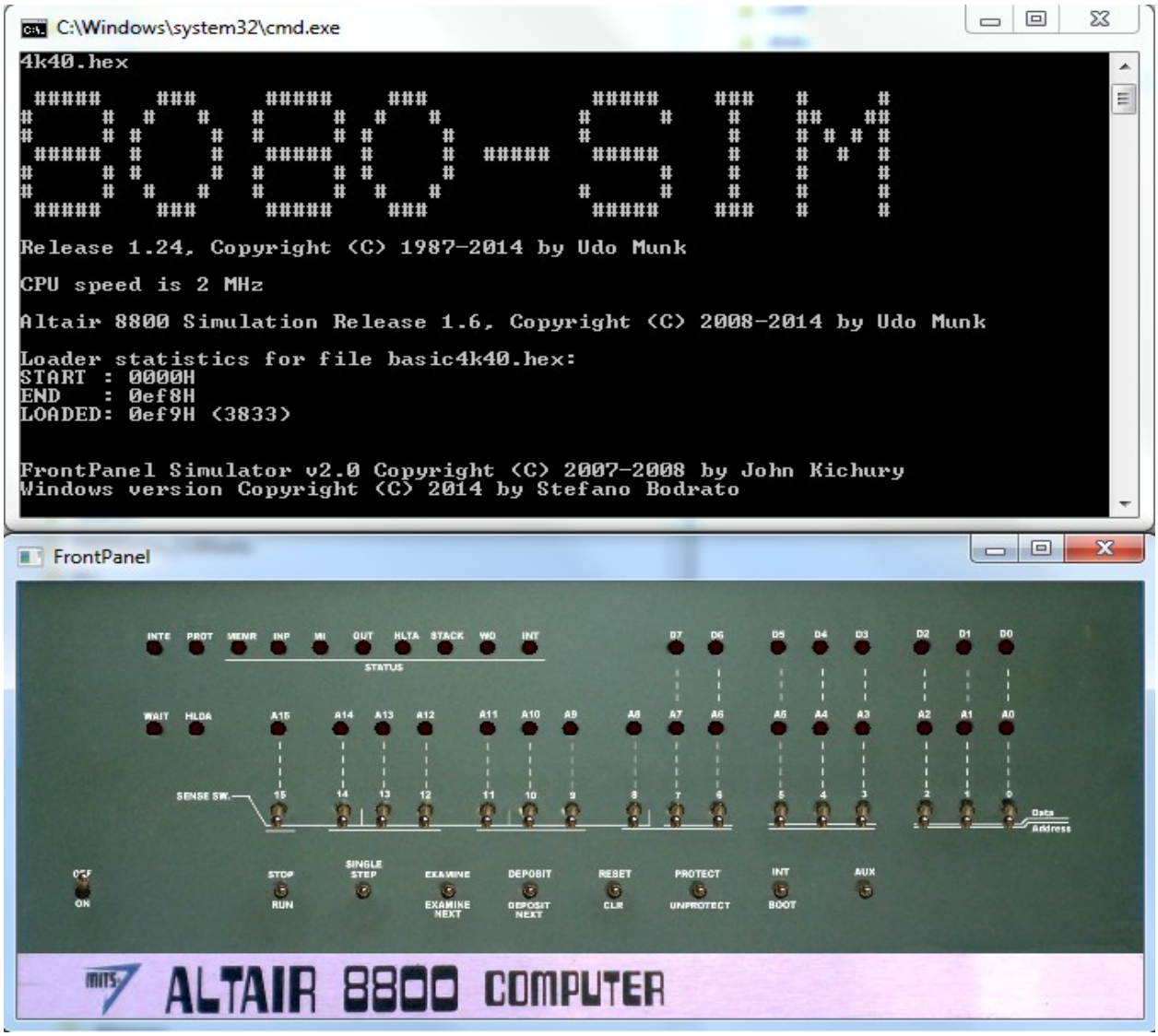
Zum ‚Einschalten' also zuerst auf den Off/On-Schalter an der ‚On' Position klicken.
Danach sollten einige LED's leuchten.

Dann den jeweiligen Systemhandbüchern folgen.

Fenster nach Ausführung basic.bat IMSAI 8080:



Fenster nach basic.bat für Altair 8800:



Heathkit ET3400

Links:
http://sharp6800.codeplex.com/
https://groups.yahoo.com/neo/groups/ET-3400/info

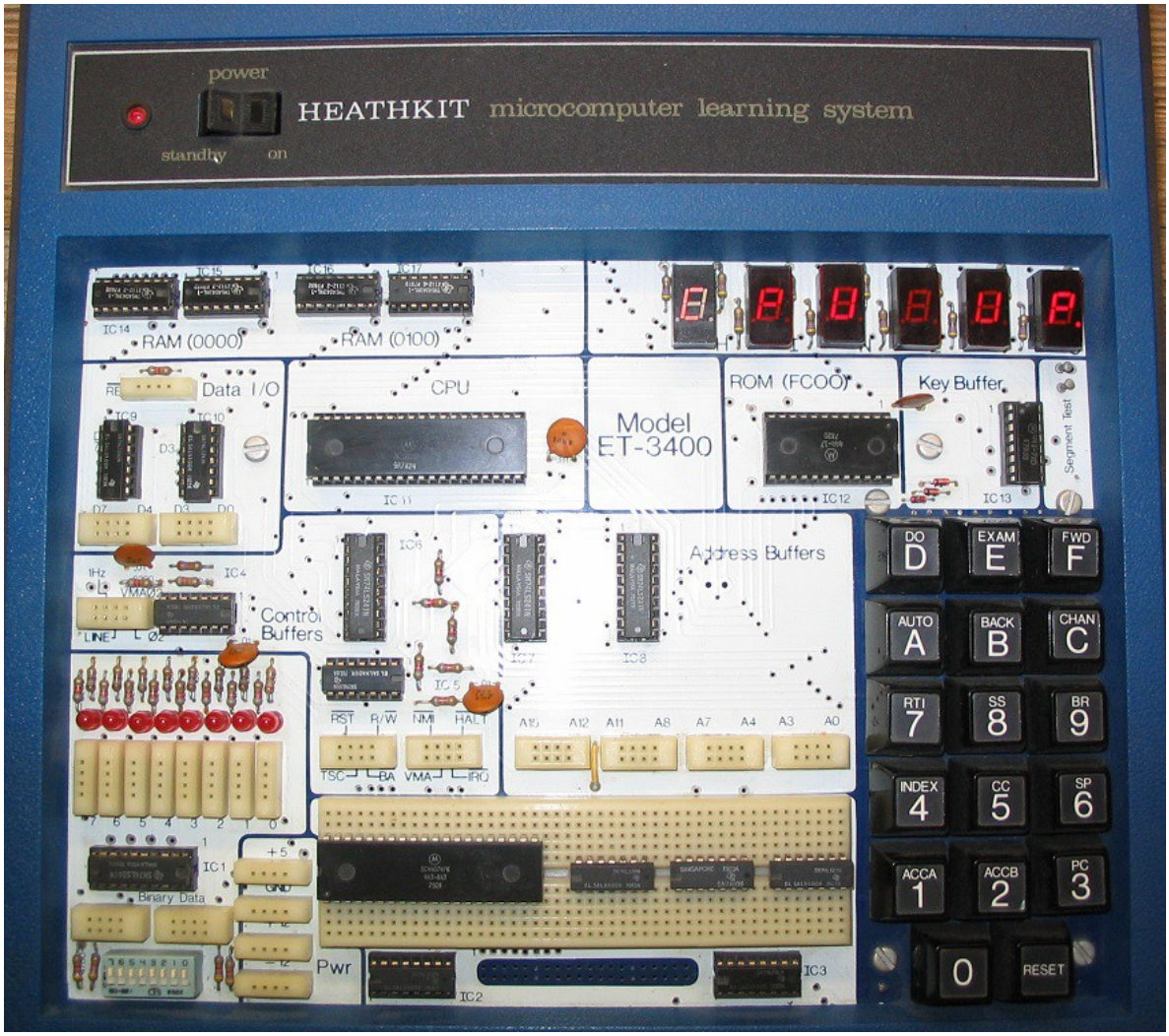
Das Einplatinencomputerkit Heathkit 3400 war ein System auf Basis einer 6800 CPU und 256 Byte Ram. Ein- und Ausgabe über Hex-Tastatur und 7-Segment Display.

Ursprünglich wurde es nur als Bausatz angeboten und musste vom Käufer zuerst zusammen gebaut werden.

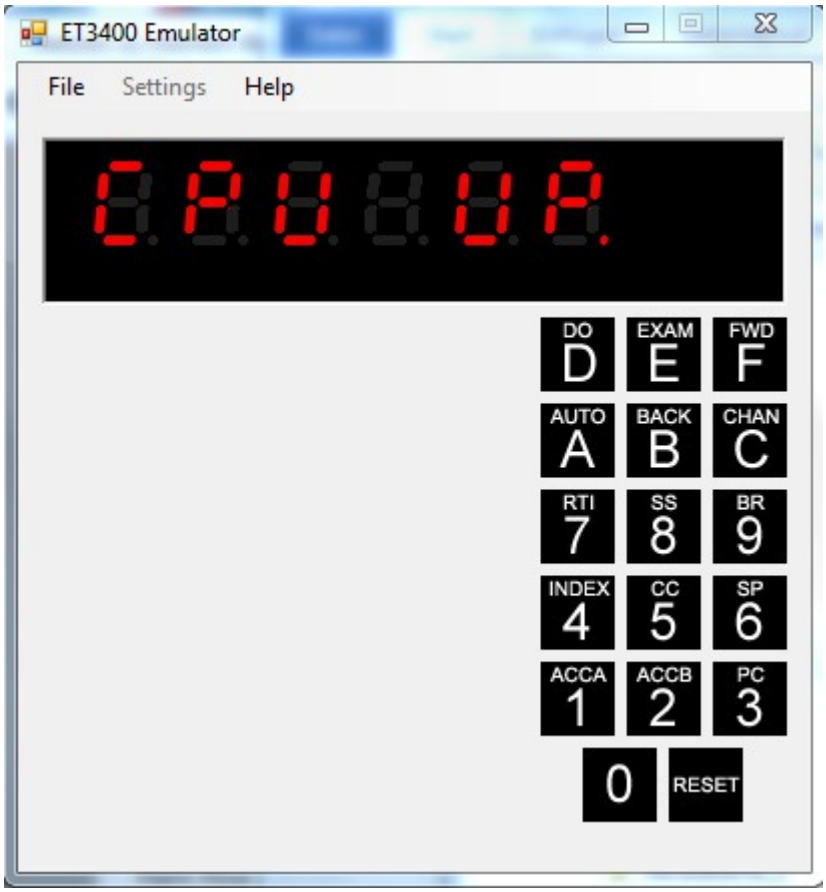
Model	CPU	an Jahr	Static RAM	Memory
ET-3400	6800	1976	256 bytes	2 chips - 2112 – exp.to 4x2112
ET-3400A	6808	1981	512* bytes	2 chips - 2114
ET-3400A	6802	1987	512* bytes	2 chips - 2114

(*) Obwohl 2x2114 1k Byte ergeben, waren nur 512 byte nutzbar, weil die höchste Adressleitung auf GND gelegt war.

Bild eines ET3400:



Emulator von David Kristepher Santos in C#:



Programme lassen sich im Motorola S19 Format laden, die sich mit dem universellen Arnold Assembler erzeugen lassen:

```
@echo off
set file=%1
if "%1"=="" goto hilfe
set bin=C:\Daten\work\aswcurr\bin
%bin%\asw.exe -L "%file%.asm" -i C:\Daten\work\aswcurr\include
%bin%\p2bin.exe -r $-$ "%file%.p"
%bin%\p2hex.exe -r $-$ "%file%.p"
del "%file%.p"
rem bdiff "%file%.rom" "%file%.bin"
goto end
:hilfe
echo Aufruf: as.bat file (ohne Endung .asm)
echo erzeugt aus file.asm Datei file.bin
:end
```

Kosmos CP1

Links:
http://www.g-heinrichs.de/minipc/
http://sourceforge.net/projects/cp1-sim/
http://www.8bit-homecomputermuseum.at/computer/kosmos_computer_praxis_cp1.html
http://www.rigert.com/ee-forum/viewtopic.php?t=1657

Der Kosmos CP1 Lerncomputer ist in einiger Hinsicht etwas Besonderes. Er wurde ab 1983 vom Kosmos Verlag vertrieben. Er enthält nur sehr wenige Bauteile. Eine CPU 8049 mit einem 2k internen ROM, ein 8155 256 Byte SRam und 2x 8-bit I/O Ports, neben einer 6-stelligen Hex-Anzeige und einer Folientastatur.

Das Besondere ist, das nicht etwa direkt Maschinensprache des 8049 im Ram abgelegt wird, sondern auf dem 8049 eine quasi virtuelle CPU läuft die vereinfachte Opcodes interpretiert.

Der System hat 128 Speicherstellen für XX.YYY Informationen.
XX = 00 bei Daten oder Befehlscode
YYY = Daten oder Adressen (0-128)

Man speichert z.B. den Werte "00.023" in der Speicherzelle "000", indem man tippt:
000 Out
00023 Inp

Man startet ein Programm, was z.B. in der Speicherzelle "000" beginnt so:
000 PC
Run

Hier eine Liste der verfügbaren Befehle:

HLT 01.000
Halt.

ANZ 02.000
Akku-Inhalt anzeigen.

VZG 03.xxx
Verzoegern um xxx Millisekunden.

AKO 04.xxx
Konstante xxx in Akku laden.

LDA 05.xxx
Inhalt von Zelle xxx in Akku laden.

ABS 06.xxx
Akku-Inhalt in Zelle xxx speichern.

ADD 07.xxx
Zum Akku-Inhalt den Inhalt von Zelle xxx addieren. Ergebnis im Akku.

SUB 08.xxx
Vom Akku-Inhalt den Inhalt von Zelle xxx subtrahieren. Ergebnis im Akku.

SPD 09.xxx
Unbedingt auf Adresse xxx springen.

VGL 10.xxx
Pruefen, ob Akku-Inhalt gleich Inhalt von Zelle xxx.

SPB 11 .xxx
Bedingt auf Adresse xxx springen.

VGR 12.xxx
Pruefen, ob Akku-Inhalt groesser als Inhalt von Zelle xxx.

VKL 13.xxx
Pruefen, ob Akku-Inhalt kleiner als Inhalt von Zelle xxx.

NEG 14.000
Akku-Inhalt negieren (nur "0" oder "1").

UND 15.xxx
UND-Verknuepfung zwischen Akku-Inhalt und Inhalt von Zelle xxx (nur"0"oder"1").

CPE 16.00x
Information an In-Port x in den Akku bringen ("0" oder "1").

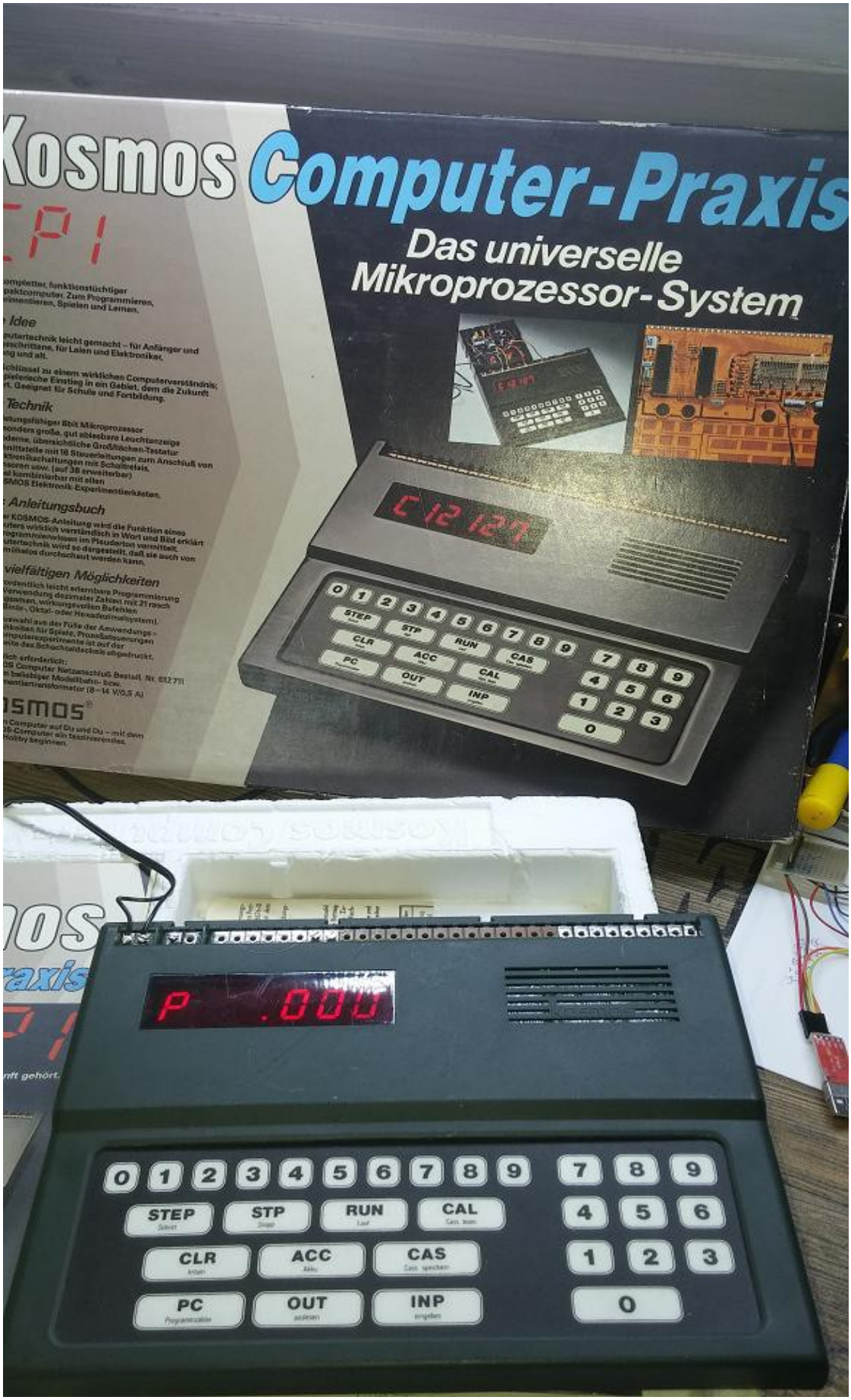
CPA 17.00x
Akku-Inhalt ("0" oder "1") an Out-Port x bringen.

LIA 19.xxx
Akku indirekt laden (mit Inhalt der Zelle, deren Adresse unter xxx steht).

AIS 20.xxx
Akku-Inhalt indirekt speichern (in der Zelle, deren Adresse unter xxx steht).

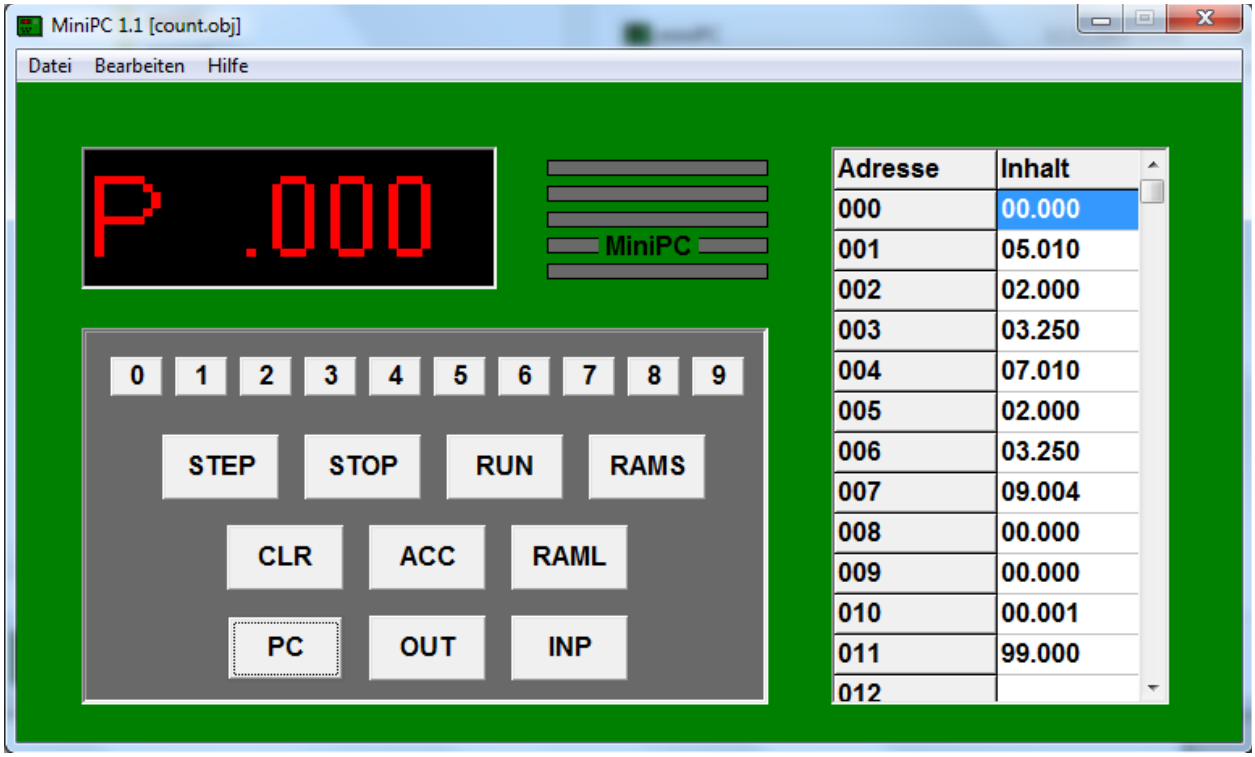
SIU 21.xxx
Indirekt unbedingt springen (auf Adresse, die unter xxx steht).

Bild des Kosmos CP1:

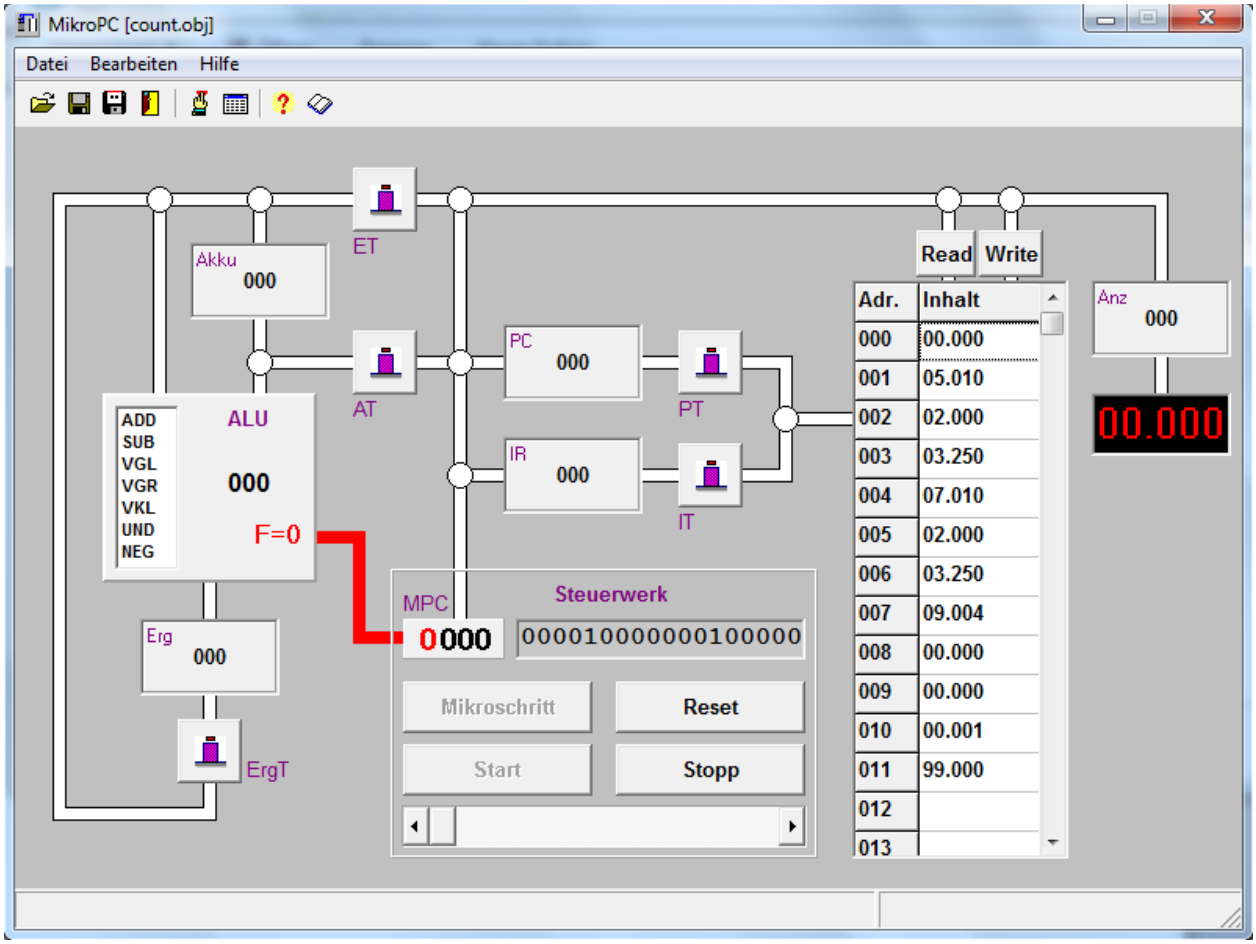


Georg Heinrichs hat dazu eine Emulation MiniPC geschrieben. Daneben wird noch ein weiteres Programm zur Darstellung der Verarbeitung sog. Mikroschritte zur Verfügung gestellt. MiniPC erlaubt auch einen Download an eine myAVR Umgebung auf denen G.Heinrichs einenfalls einen Interpreter implementiert hat.

MiniPC:



MikroPC:


MikroPC [count.obj]
Datei Bearbeiten Hilfe
ET
Akku
000
AT
PC
000
PT
IR
000
IT
ADD
SUB
VGL
VGR
VKL
UND
NEG
ALU
000
F=0
Erg
000
ErgT
MPC
0000
Steuerwerk
0000100000001000 0
Mikroschritt
Reset
Start
Stopp
Read
Write
Adr.
Inhalt
000 00.000
001 05.010
002 02.000
003 03.250
004 07.010
005 02.000
006 03.250
007 09.004
008 00.000
009 00.000
010 00.001
011 99.000
012
013
Anz
000
00.000

myAVR Board:


Reset
ISP
GND
5V
Taster1
Taster2
RxD
TxD
Summer
AREF
LED-Rot
LED-Gelb
LED-Grün
Pot.1
Pot.2
Photo
Port-B
Port-C
LO
HI
www.myAVR.de
R/W
Kontrast
Beleuchtung
An
M-B1
myAVR LCD V2.5
www.myAVR.de
D2=RS,D3=E,D4-7=D4-7
** MiniPC-Sim **
Programm laden!

HP 98010/20/30 Programmierbare Rechner

Links:
http://sourceforge.net/projects/hp9800e/files/go9800/
http://hp9800e.sourceforge.net/

GO9800 ist wirklich eine wunderschöne Emulationsumgebung der programmierbaren Rechner der Firma Hewlett-Packard HP9810, HP9820 und HP9830. Hier wurde viel Wert auf Detailtreue und möglichst echtem Feeling der Originalsysteme gelegt – bis hin zur Emulation der Geräuschkulisse der Systeme. Das Emulationspaket ist in Java geschrieben und somit auf vielen Plattformen lauffähig.

Entpackter Win32 Zip-Ordner

Name	Änderungsdatum	Typ	Größe
applications	19.12.2013 08:10	Dateiordner	
changes_1.60	27.05.2012 17:08	Textdokument	10 KB
GO9800	27.05.2012 17:09	Executable Jar File	9.462 KB
GO9800_User_Manual_1.60	27.05.2012 15:45	Adobe Acrobat D...	1.932 KB
HP9810A.cfg	10.05.2010 22:23	CFG-Datei	1 KB
HP9810A	27.05.2012 15:33	Windows-Befehlss...	1 KB
HP9810A2.cfg	08.04.2010 21:47	CFG-Datei	1 KB
HP9810A2	27.05.2012 15:34	Windows-Befehlss...	1 KB
HP9810A2-keyb.cfg	18.03.2008 19:06	CFG-Datei	2 KB
HP9810A-keyb.cfg	29.09.2008 22:27	CFG-Datei	2 KB
HP9820A.cfg	16.10.2011 21:58	CFG-Datei	1 KB
HP9820A	27.05.2012 15:34	Windows-Befehlss...	1 KB
HP9820A-keyb.cfg	12.07.2008 10:41	CFG-Datei	2 KB
HP9821A.cfg	31.10.2011 02:21	CFG-Datei	1 KB
HP9821A	27.05.2012 15:34	Windows-Befehlss...	1 KB
HP9821A-keyb.cfg	12.07.2008 11:41	CFG-Datei	2 KB
HP9830A.cfg	31.10.2011 21:01	CFG-Datei	1 KB
HP9830A	27.05.2012 15:34	Windows-Befehlss...	1 KB
HP9830A-keyb.cfg	29.09.2008 22:55	CFG-Datei	2 KB
keynames.cfg	19.11.2008 22:49	CFG-Datei	1 KB
license	24.11.2006 21:47	Textdokument	18 KB

Die CMD Dateien enthalten den Java Aufruf. Z.B.:
```
java -Dsun.java2d.d3d=false -jar GO9800.jar HP9810A
```

Startbildschirm:

Diese Rechnersysteme arbeiten in der sog. Umgekehrten Polnischen Notation (UPN; engl: RPN). D.h. erst die Operanden, dann die Operation.

Anstatt 2 x 2 = 4 also 2 2 x =

In der obigen Hardcopy als kleine Eingewöhnung:

Pi
x<->y
2
X

Generell liegt hier vor einem ein Eintauchen in diese wirklich faszinierende Welt der historischen, programmierbaren und wissenschaftlichen Rechnersystemen! Ohne durcharbeiten der Handbücher kommt man hier allein mit rumprobieren nicht wirklich weit. Die Emulationsumgebung unterstützt eine Vielzahl von Peripheriegeräten wie z.B. Plotter etc. pp.

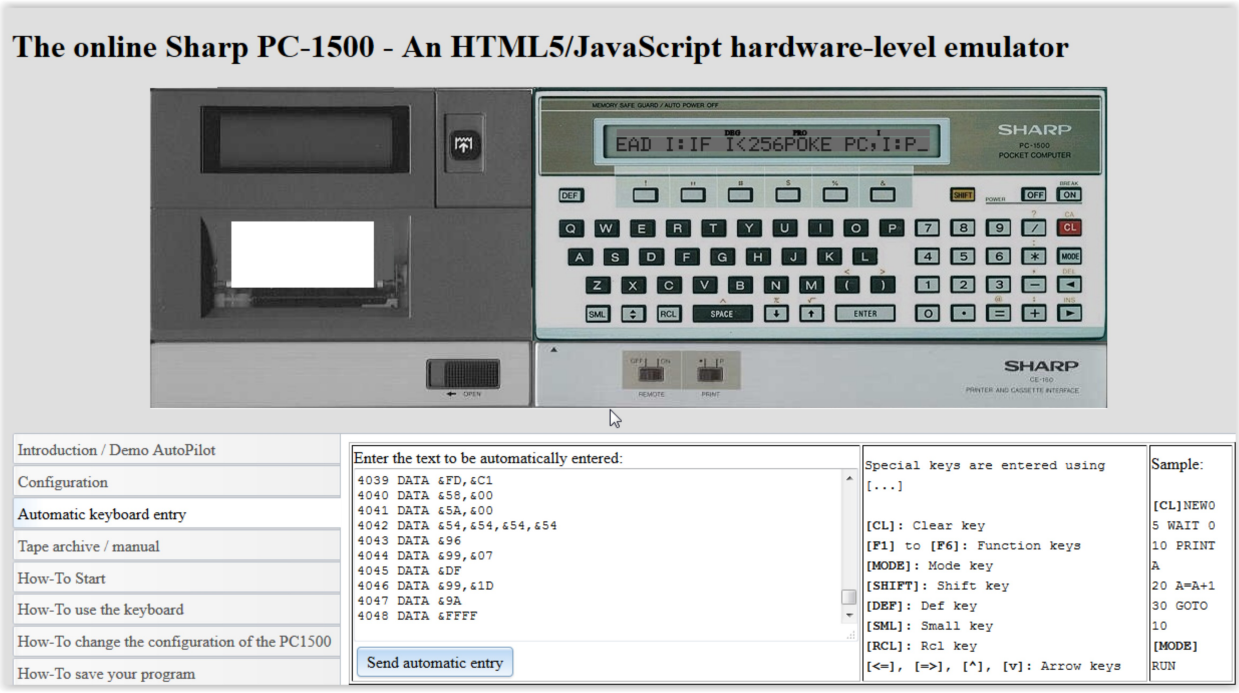
Sharp PC1500 und weitere

Links:
http://de.wikipedia.org/wiki/Sharp_PC-1500
http://www.forever1500.fr/index1500.html
http://pockemul.free.fr/
http://www.pc1500.com/

Der Sharp PC-1500 war ein Taschencomputer der 1982 mit einer 8-Bit Z80 ähnlichen CPU und 8 KB eingebautem Arbeitsspeicher auf den Markt kam.
Als Programmiersprache war ein eingebautes BASIC vorhanden. Zudem besaß er zahlreiche eingebaute wissenschaftliche Funktionen. Programme konnten auf einen Kasettenrecorder als Snap-In Zusatzoption gespeichert werden. Da das Ganze sehr kompakt und leicht zu programmieren war, erfreuten sich diese Taschencomputer großer Beliebtheit. Dank der beiden hier gezeigten Emulationen kann sich jeder mit diesen faszinierenden Geräten auseinander setzen.

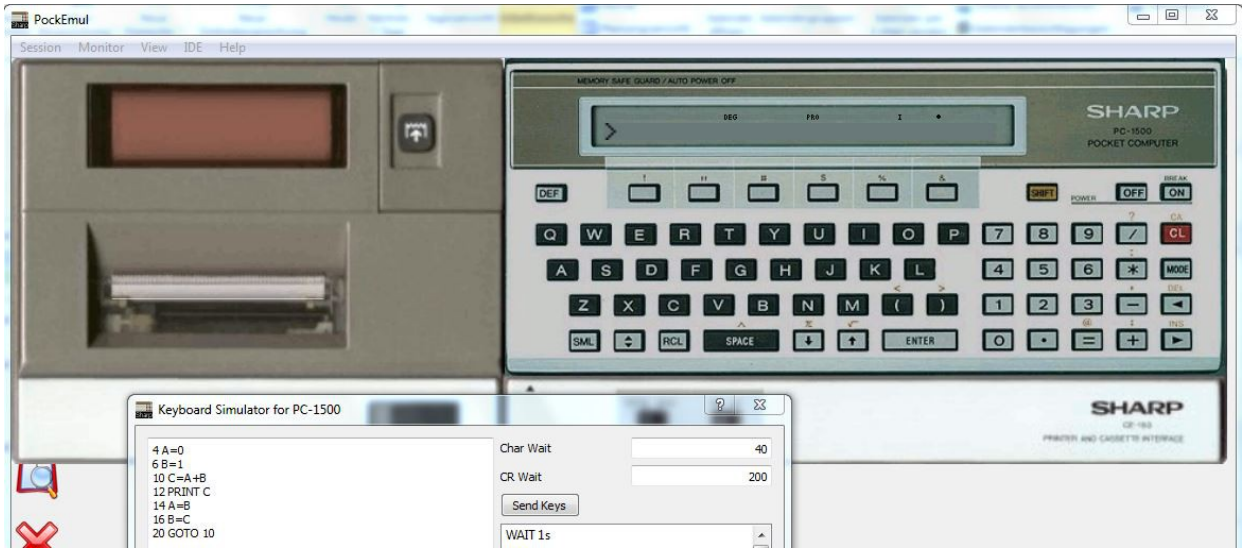
Bild eines orig. Sharp PC 1500:

Emulation in Java Script



Über die ‚Automatic keyboard entry' Funktion lassen sich Basic Programme automatisiert eintippen und somit laden.

Pockemu



Pockemu unterstützt eine Vielzahl von Sharp Taschencomputern und weiteren Modellen.
Ganz Hipp ist die Auswahl des Modelles, indem man die Bilder nach links oder rechts ‚wischt'.

Wichtig: Man sollte, nachdem man sich seine Wunschkonfiguration (Modell + Peripherie) zusammen geklickt hat, diese erst einmal sichern. So kann man jederzeit wieder auf diesen Stand zurückgreifen. Genauso, sollte man nach dem Laden von Programmen (auch hier wird ein Keyboard Simulator genutzt) die sog. Session sichern.

Solche Sessions kann man dann auch über das Internet verteilen und von dort fertige Sessions laden.

K100x Emulation der programmierbaren Desktop Rechner der DDR

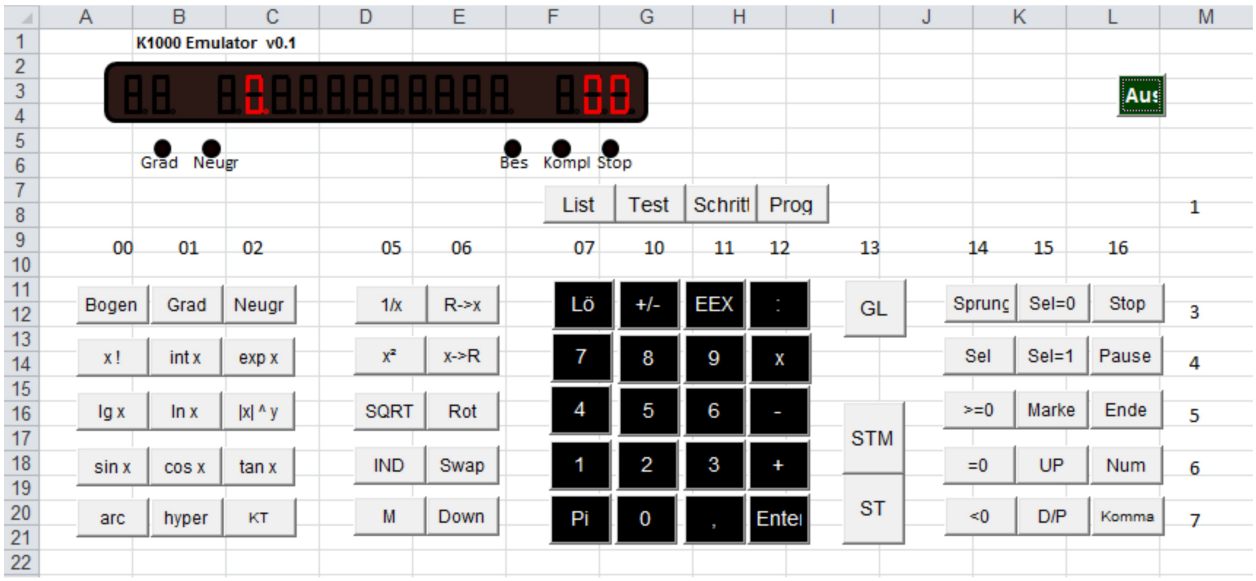
Links:
http://k1000.aibologie.de/index.php/Hauptseite
http://www.k1000uc.de/

K1000 ist die Familienbezeichnung einer Serie von "Kleinstrechnern", die von 1978 bis 1985 vom VEB Robotron in der damaligen DDR produziert wurden. So ein Gerät wiegt ca. 20 kg. Sie sind ebenfalls wie die HP98n0 Rechner in UPN bedienbar.

Bild des Originals und des AVR Nachbaus:

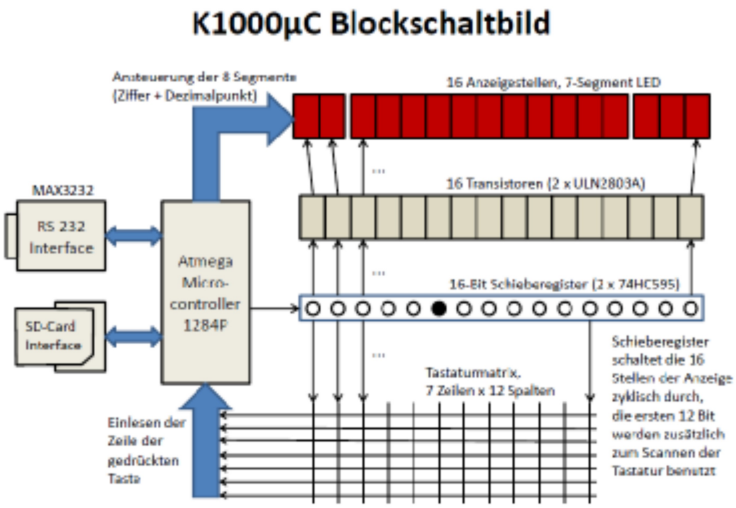
Michael Berger hat die Emulation zuerst in reiner Software als Excel Makro (nur Proof of Concept) und danach als Nachbau mit AVR Mikrocontroller der Firma Atmel erstellt.

Excel Emulator (Makros aktivieren + Einschalter anklicken):



AVR Nachbau

Es folgt eine stark vereinfachte Erklärung wie der K1000uC “tickt”, dabei wird Bezug auf das Blockschaltbild genommen:



Über ein Ausgabepin des Microcontrollers wird in ständiger Wiederholung eine Folge von 16 Bits ausgegeben: 1 mal eine "1", dann 15 mal eine "0". Diese Ausgabe ist auf den Eingang eines 16-Bit Schieberegisters geschaltet, das funktioniert wie eine Warteschlange. Jeder neue Input schiebt den Inhalt eine Stelle weiter. Die eine "1" wird also Takt für Takt durch diese Warteschlange geschoben, wenn sie schließlich ganz hinten rauskommt wird gerade vorn wieder eine "1" nachgeschoben.

Die 16 Bit der Warteschlange werden sowohl für die Ansteuerung der Anzeige als auch für die Abfrage der Tastatur verwendet. Zur Anzeige geht es pro Anzeigeelement noch über einen Transistor, der stellt lediglich sicher dass der jeweilige Pin des Warteschlangenbausteins nicht elektrisch überlastet wird.

So wird jede Stelle der Anzeige kurz zugeschaltet, danach die nächste usw. Eigentlich leuchten nie zwei Anzeigeelemente gleichzeitig. Aber da dieser Vorgang 60 mal pro Sekunde abläuft bekommt das Auge den Eindruck vermittelt, dass mehrere Anzeigestellen gleichzeitig leuchten.

Jeweils in der Zeit wenn eine bestimmte Stelle aktiviert ist, gibt der Atmega über 8 Pins das Segmentmuster für die gewünschte Ziffer zur Anzeige aus. Auf der Anzeige sind die jeweils gleichen Segmente sämtlicher Ziffern leitend verbunden.

Die ersten 12 Stellen des Schieberegisters sind außerdem mit den 12 Spalten der Tastaturmatrix verbunden. Wird eine Taste gedrückt, gibt der geschlossene Kontakt das Signal über die Leitung der zugehörigen Zeile zum Mikrocontroller zurück, wo diese Information ausgewertet wird.

Im Innern des Atmega läuft eine komplett in C geschriebene Emulation des U808 (Intel 8008) Prozessors, der das „Herz“ der Robotron K1000 Rechner darstellte. Die Emulation arbeitet mit Teilen der Firmware des echten K1000.

Als Massespeicher kommen beim K1000uC SD-Karten zum Einsatz. Diese werden auf Blockebene beschrieben. Auf einer SD-Karte können 1000 verschiedene Programme mit den dreistelligen Nummern 000 bis 999 abgespeichert werden.

Über eine Serielle Schnittstelle kann man den K1000uC mit einem PC koppeln, um Daten auszutauschen oder die Vorgänge im Innern des K1000uC zu debuggen.

Fertiges Gerät:

Salto – Xerox Alto II Simulator

Xerox PARC Alto - Die Anfänge der grafischen Benutzeroberflächen

Die Anfänge der heutigen Benutzeroberflächen begannen in den 70iger Jahren in Xerox PARC Laboren. Xerox selber erkannte das zukunftsweisende Potential nicht, sehr wohl aber Bill Gates (Microsoft) und Steve Jobs (Apple). Insbesondere bei Apple fanden sich wesentliche Merkmale wieder in der Lisa und den MacIntosh Systemen.

Nach geeigneten Unterlagen und Emulatoren musste ich lange suchen.

Installation

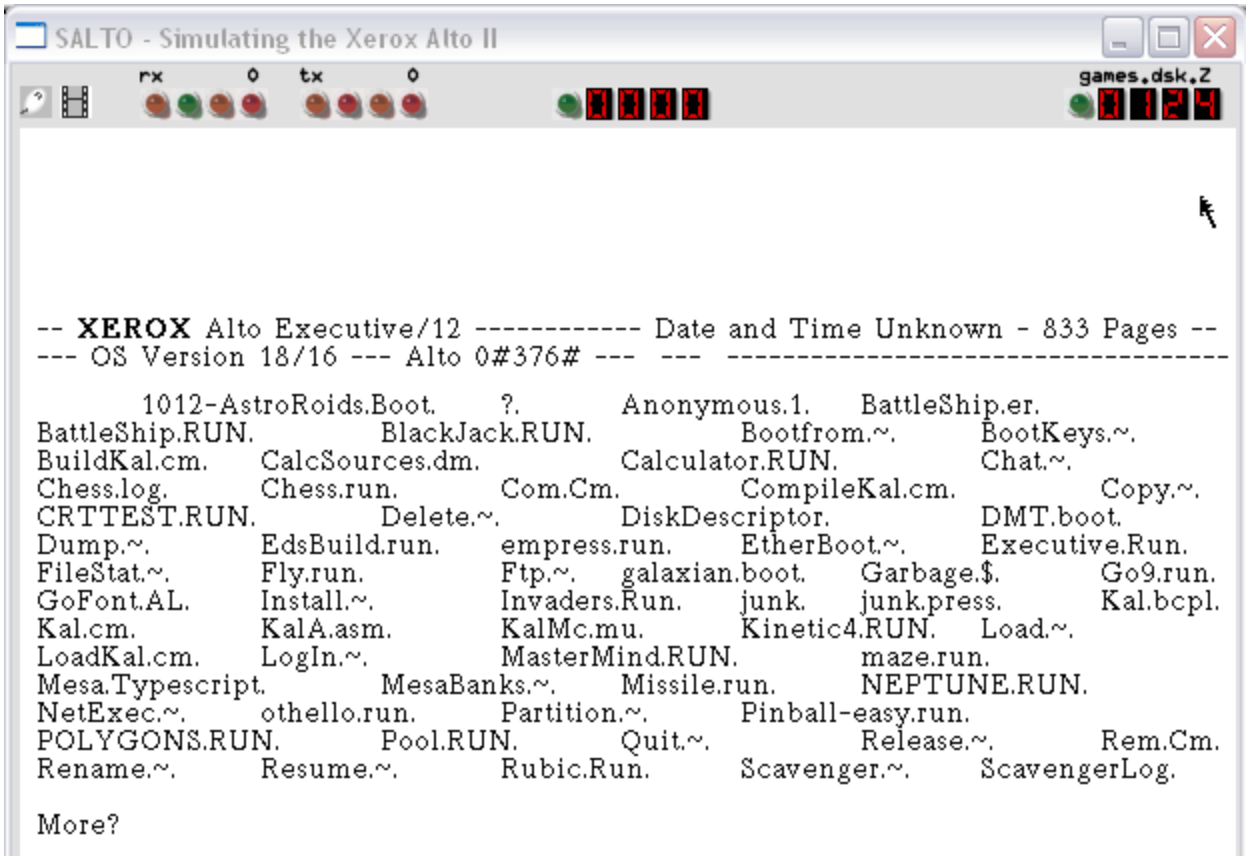
saltowin.rar entpacken; erzeugt Verzeichnis saltowin. Diskimage z.B: games.dsk.Z dort herunterladen.

Batchdatei z.B. start.bat mit folgendem Inhalt erstellen:

salto games.dsk.Z

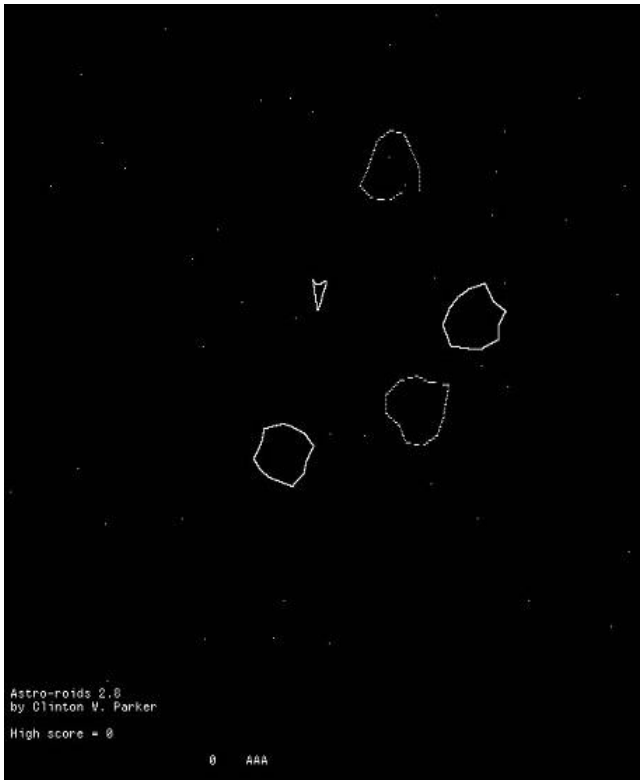
Batchdatei ausführen. Nach ca. 10-15s erscheint der Prompt '>'.

? zeigt einige Kommandos:

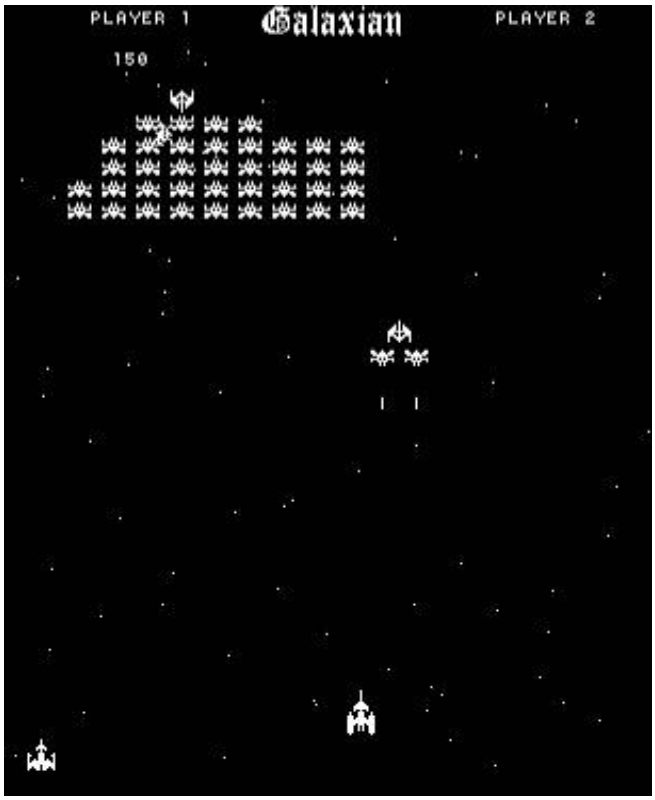


Generell: ? = Dateiliste; <programm> <RETURN> startet das Programm; F10 beendet salto.

Spiele die durch bootfrom <name>.boot gestartet werden:
AstroRoids

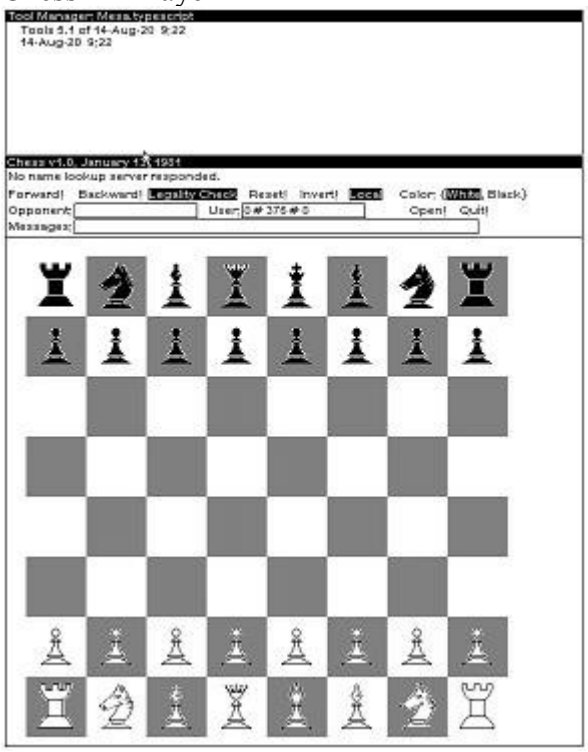


Galaxian

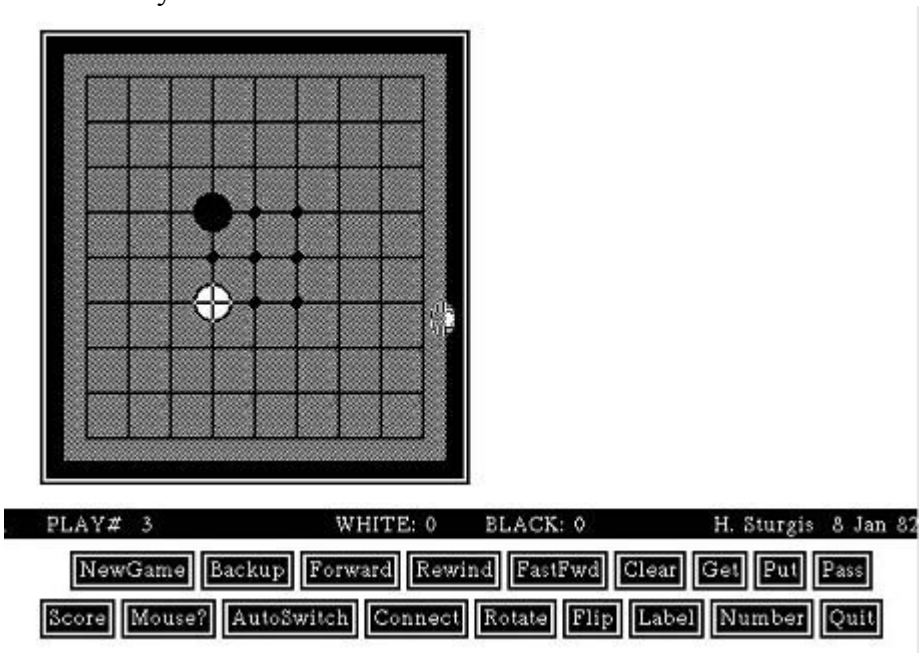


Spiele die einfach durch <programm< <return> gestartet werden:

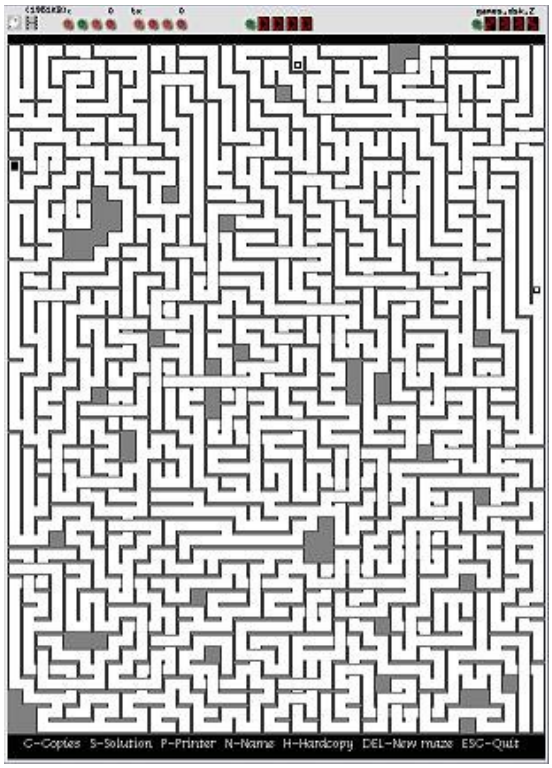
Chess – 2 Player



Go9 – 2 Player

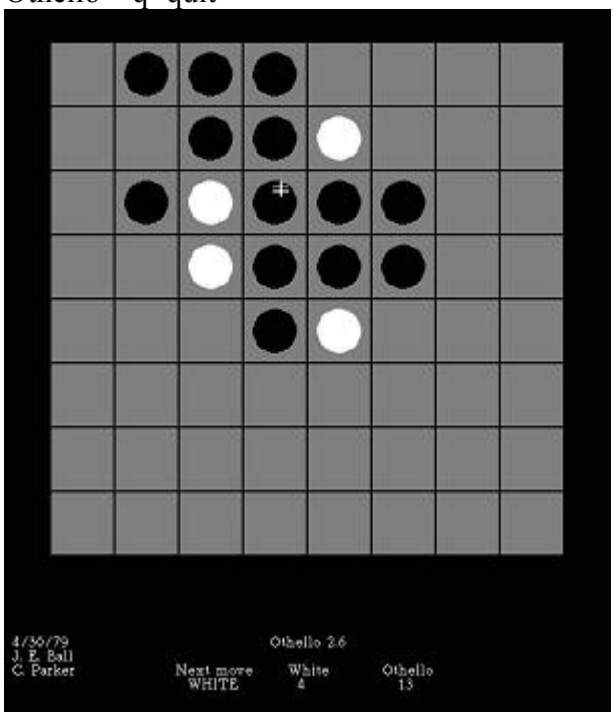


Maze

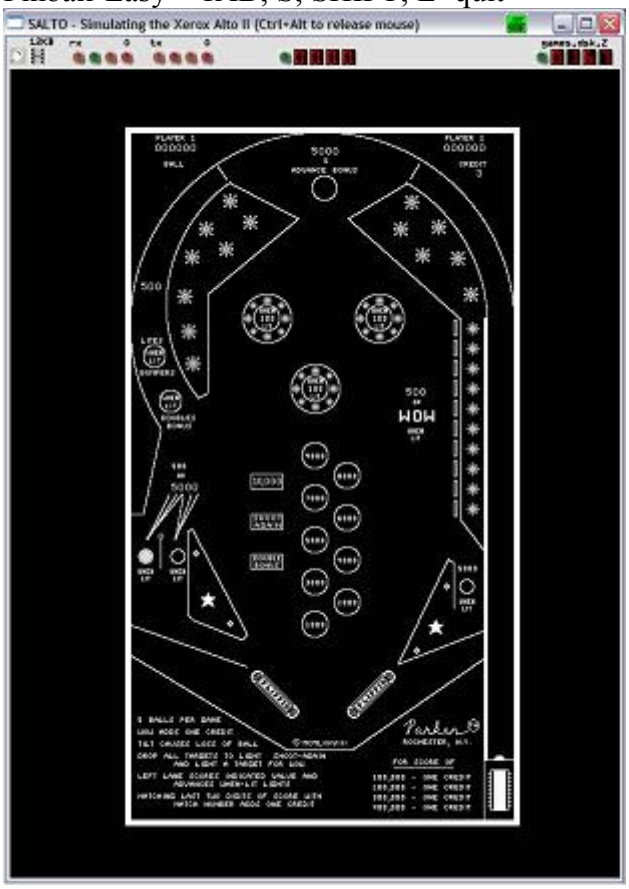

C-Copies S-Solution P-Printer N-Name H-Hardcopy DEL-New maze ESC-Quit

Missile

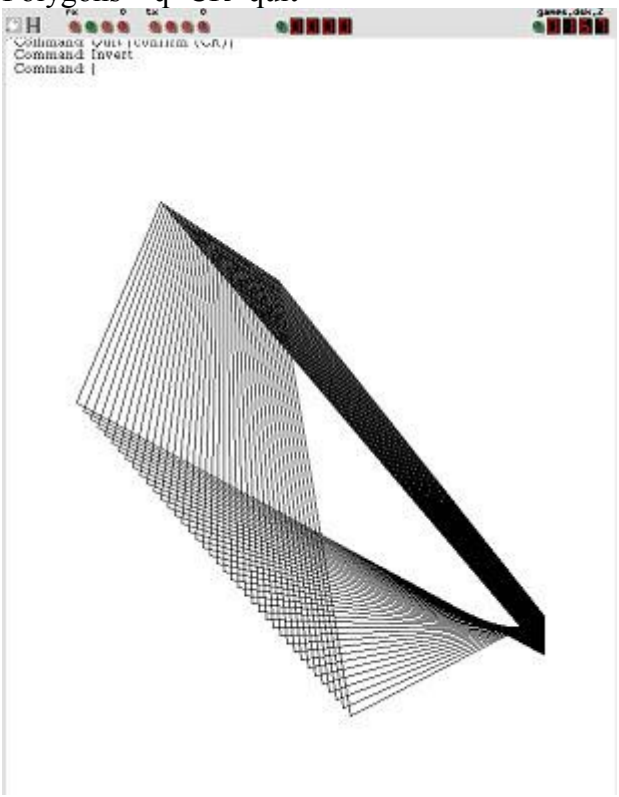
Othello – q=quit



Pinball-Easy – TAB; S; SHIFT; E=quit



Polygons – q+CR=quit



Spacewar

Quadrant: 15

Type command:

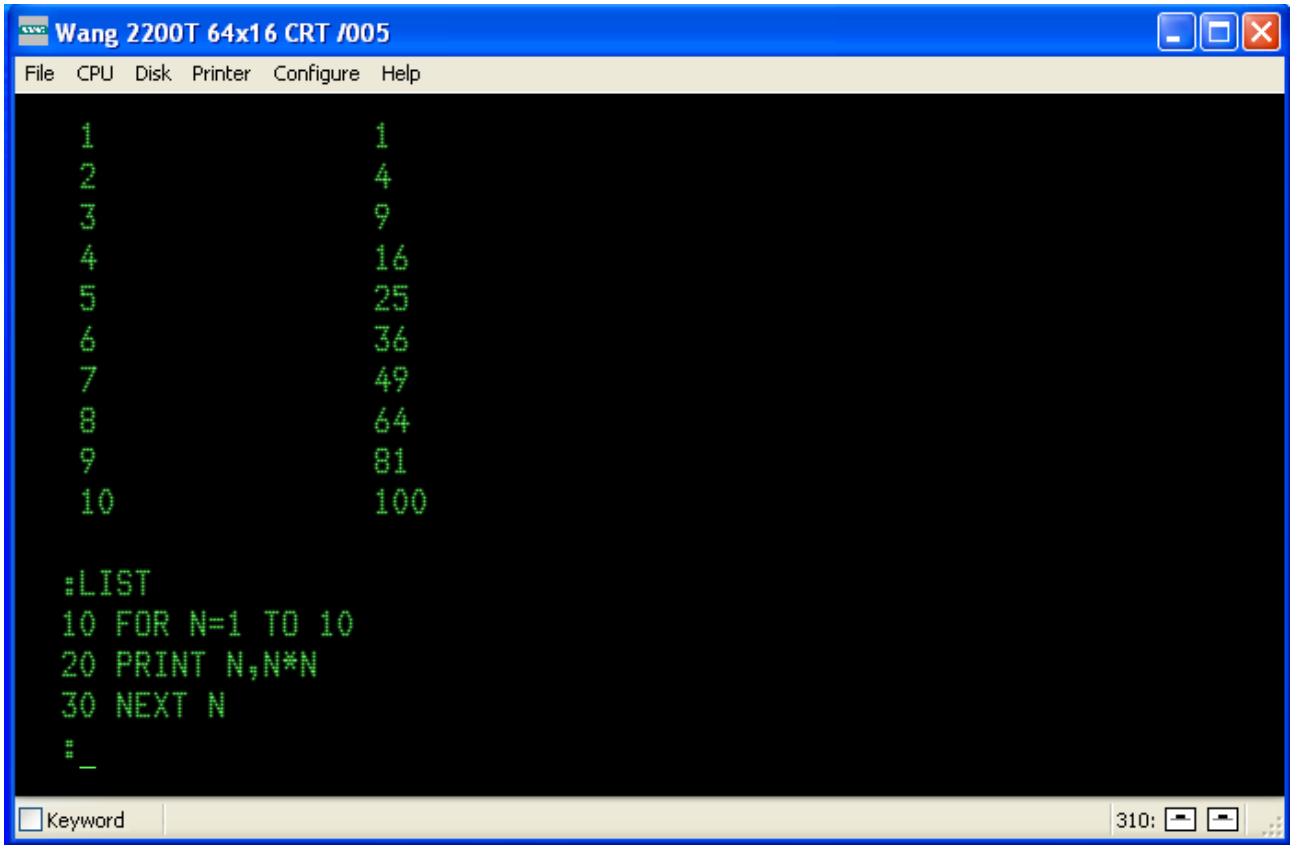
Und es gibt noch einiges mehr zu entdecken!

Wang2200

Links:
http://www.wang2200.org/

Die Wang 2200 ist ein System aus der Vor-CPU Ära, gebaut aus hunderten TTL Chips im Mai 1973. Direkt nach dem Einschalten stand ein leistungsfähiger Basic Interpreter zur Verfügung. Programme konnten auf Kassetten gesichert und von dort wieder geladen werden.

Bild des Emulators (unter dem Link zu laden):

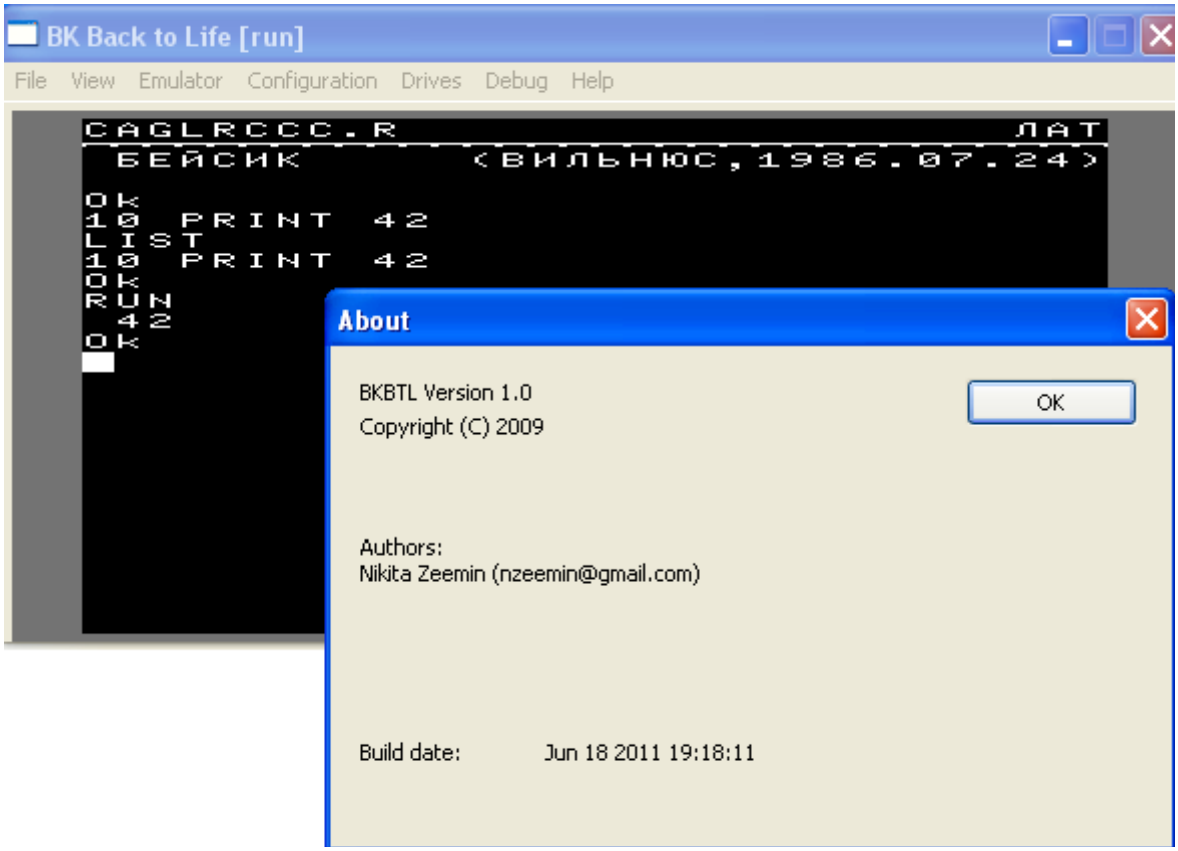


Elektronika BK

Links:
https://en.wikipedia.org/wiki/Electronika_BK
https://github.com/nzeemin/bkbtl

Der/die Elektronika BK waren 16-bit Computer mit einer PDP-11 kompatiblen CPU, welche in der Sovietunion ab 1984 gebaut wurden.

Bild des Emulators:



Apple 1

Links:
http://pom1.sourceforge.net
http://www.chez.com/apple1/Apple1project/index.html
http://www.applefritter.com/apple1/
http://www.brielcomputers.com/replica1.html

Der Apple I war ein von Steve Wozniak und Steven Jobs entwickelter Computer für Heimanwender, der zum ersten Modell der Firma Apple Computer wurde.

Der Apple I wurde am 1. April 1976 auf einem Treffen des Homebrew Computer Club vorgestellt. Er war als nackter Einplatinencomputer konzipiert. Ausgestattet mit einer Videoschnittstelle (S/W; nur Text), 4 KB dynamischem RAM, Tastatur und der 6502 CPU. Das Video-System war auf eine sehr eigenwillige Weise aufgebaut, denn es nutzte Schieberegister als Bildschirmspeicher, da diese damals noch billiger waren als dynamisches RAM.

Der eigentliche Rechner wurde von Apple als fertig bestückte Platine geliefert und musste vom Händler oder Besitzer noch mit Netzteil, Tastatur, Bildschirm und optionales Gehäuse ausgestattet werden.

Originale Apple I Computer sind heute fast unerschwinglich teuer!

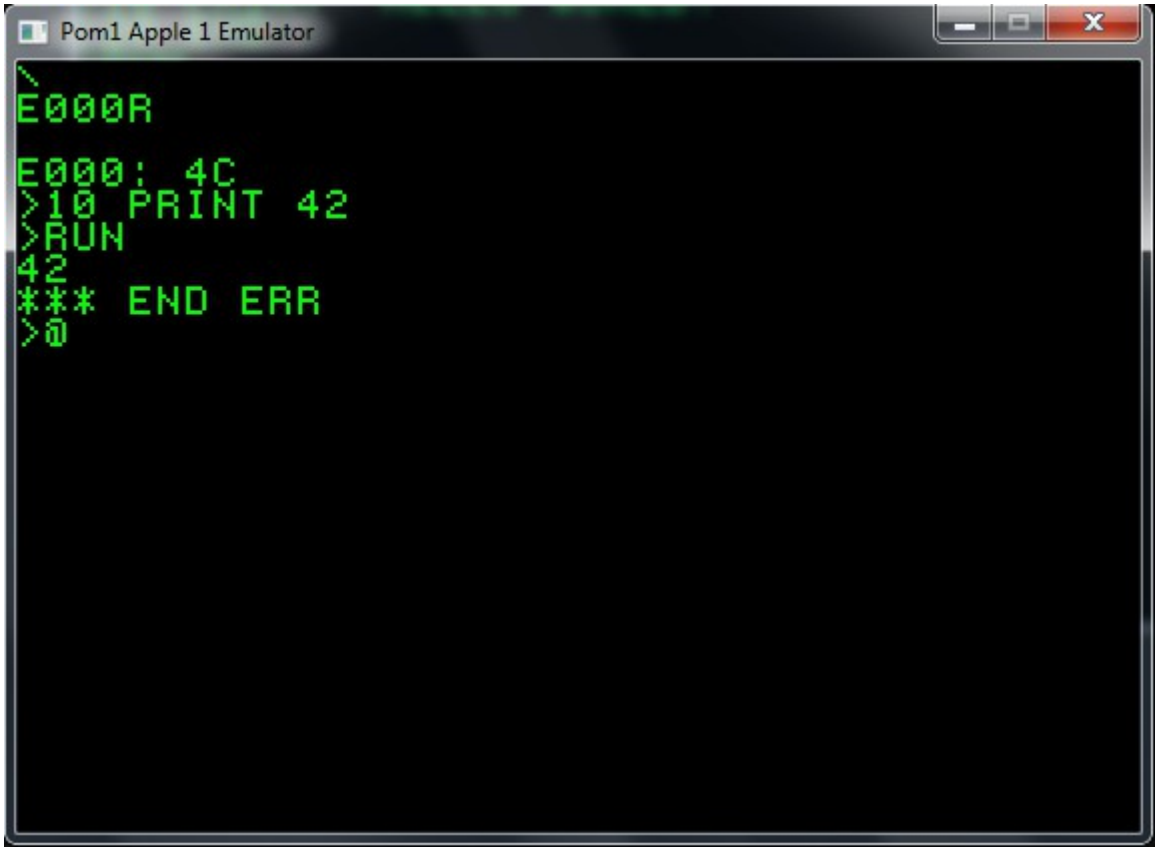
Optionen des POM1 Apple I Emulators

Option über Ctrl+<Buchstabe>. Einige Optionen sind auch über Kommandozeilen Parameter konfigurierbar.

Option	Letter	Parameter	Description
Load Memory	L		Load memory from a binary or ascii file.
Save Memory	S		Save memory to a binary or ascii file.
Quit	Q		Quit the emulator.
Reset	R		Soft reset the emulator.
Hard Reset	H		Hard reset the emulator.
Pixel Size	P	-pixelsize <n>	Set the pixel size (1 or 2).
Scanlines	N	-scanlines	Turn scanlines on or off (pixel size 2 only).
Terminal Speed	T	-terminalspeed <n>	Set the terminal speed (Range: 1 - 120).
RAM 8K	E	-ram8k	Use only 8KB of RAM or entire 64KB of RAM.
Write In ROM	W	-writeinrom	Allow writing data in ROM or not.
IRQ/BRK Vector	V		Set address of interrupt vector.
Fullscreen	F	-fullscreen	Switch to fullscreen or window.
Blink Cursor	B	-blinkcursor	Set the cursor to blink or not.
Cursor Block	C	-blockcursor	Set the cursor to block or @.
Show About	A		Show version and copyright information.

POM1 Emulator

Pom1 Apple I Emulator ist eine C+SDL oder Android Implementierung eines Apple I Computers Von John D. Corrado und Basis der Vorarbeiten von Verhille Arnaud. Der Basic Interpreter ist bereits unter E000 verfügbar in der Emulation.



Unter applefritter.com und Dokumentation gibt es z.B. ein Apple 1 Manual in Englisch.

Briel Computer (USA) stellt mit dem Replica1 einen Nachbau als Bausatz zu Verfügung.

Apple 1 Emulation auf Basis Arduino

Links:
http://petersieg.bplaced.net/?Arduino_Apple_1_Emulator%26nbsp%3B
http://forum.arduino.cc/index.php?topic=291681.0
http://www.sbprojects.com

Auf Basis der 6502 Emulation von Mike Chambers entstand dieser Apple 1 Emulator.
Die Arduino Plattform ist 2014/5 sehr preiswert zu bekommen. Ein Pro bekommt man für <5€.
Ein Mega 2560 R3 für <15€ und ein ARM-CortexM3 basierendes Due Board für <25€.

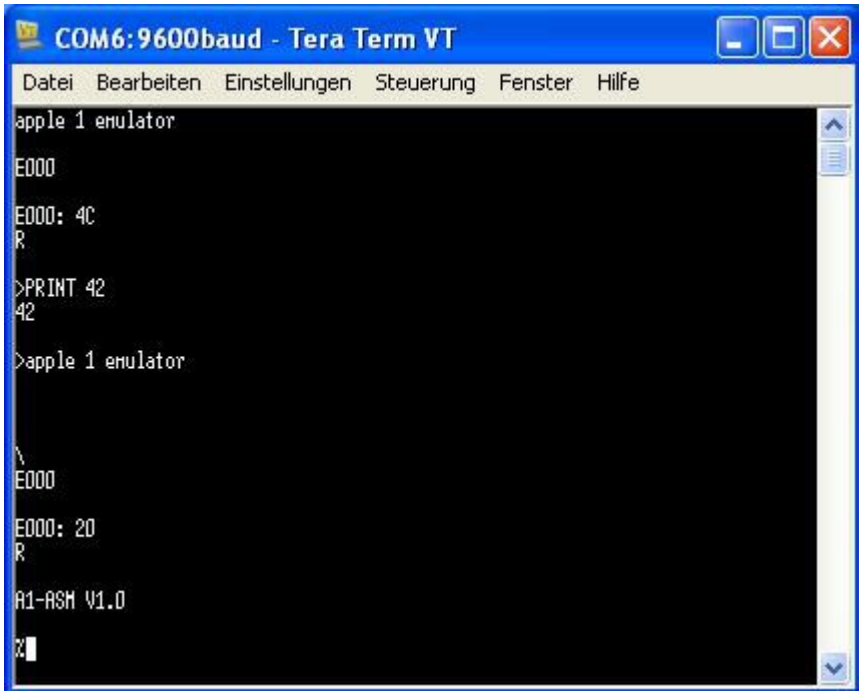
Grundsätzlich läuft das auf eine Pro Board mit Atmega328P. Dadurch, daß dieser AVR Chip aber nur 2Kb Ram hat, bleiben hier nur 1,5k Ram für den emulierten Apple 1 übrig. Zu wenig für Basic etc. Das reicht dann max. für kleinere Maschinenspracheprogramme.

Bei einem Mega2560 Board ist ein Atmega2560 drauf, der 8Kb Ram hat und somit 4-6Kb der Emulation bleiben. Das reicht für viele Dinge.

Ideal ist ein Due Board. Hier kann man der Emulation 32Kb spendieren und der ARM Chip läuft mit 80MHz.

Die Bedienung erfolgt über eine serielle Verbindung mit 9600 8N1. Darüber können auch Programme in Maschinensprache (in HEX/Monitorformat) oder Basicprogramme geladen werden (ggf. Zeichen- und Zeilenverzögerungen im Terminalprogramm einstellen).

Über Pin 2 auf GND läßt sich anstatt des Apple Integer Basic der A1 Assembler (sbprojects) einbinden beim Systemstart oder Reset.



Apple Lisa

Links:
http://lisa.sunder.net/

Lisaem ist einer der wenigen / der einzige Apple Lisa Emulator. Author: Ray Arachelian. Die Nutzung ist nicht ganz einfach. Am besten liest man sich die sehr ausführliche, englische Anleitung (PDF) durch, die im Download Archiv lisaem.1.2.6.win32.zip enthalten ist.

Installation

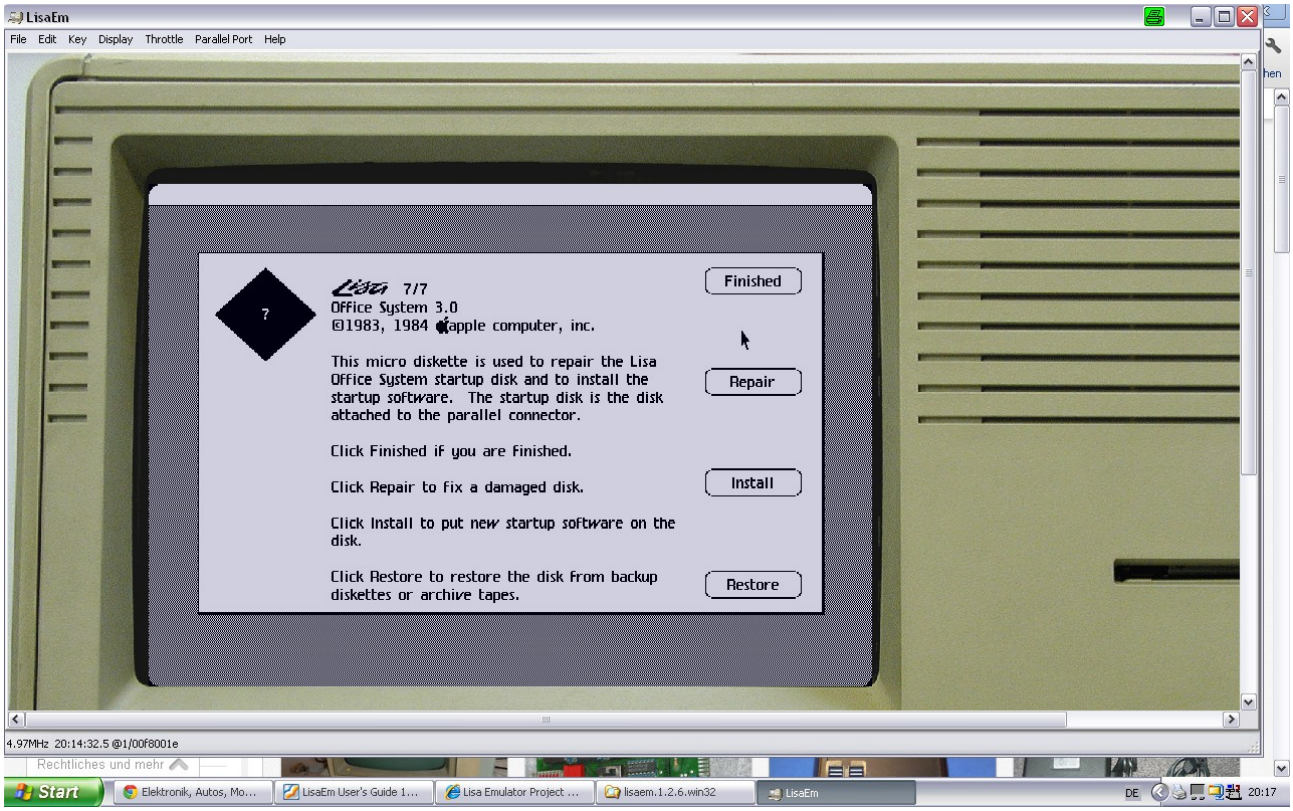
Obiges Archiv herunter laden und entpacken. Lisaem.exe ist der Emulator.
Man sucht sich noch die 5 Diskimages des Apple Lisa Office Systems:

(Sind im Internet als gezippete Dateien zu finden).

Dann den Emulator starten und erst einmal ein leeres Profile Festplatten Image anlegen.
Diese sieht man oben in dem Directory als lisaem-profile.dc42 – 5MB Image.

Wirklich schön gelöst ist hier die virtuelle Geräuschkulisse der Lisa Emulation!

Danach von Disk 1 booten und Install wählen.



Nach und nach die Diskimages nach Aufforderung virtuell einlegen. Nach einem Neustart kann man anschließend von dem Festplattenimage booten.

Mini vMac – Emulator für 68k Macintosh

Links:
http://minivmac.sourceforge.net/
http://www.knubbelmac.de/

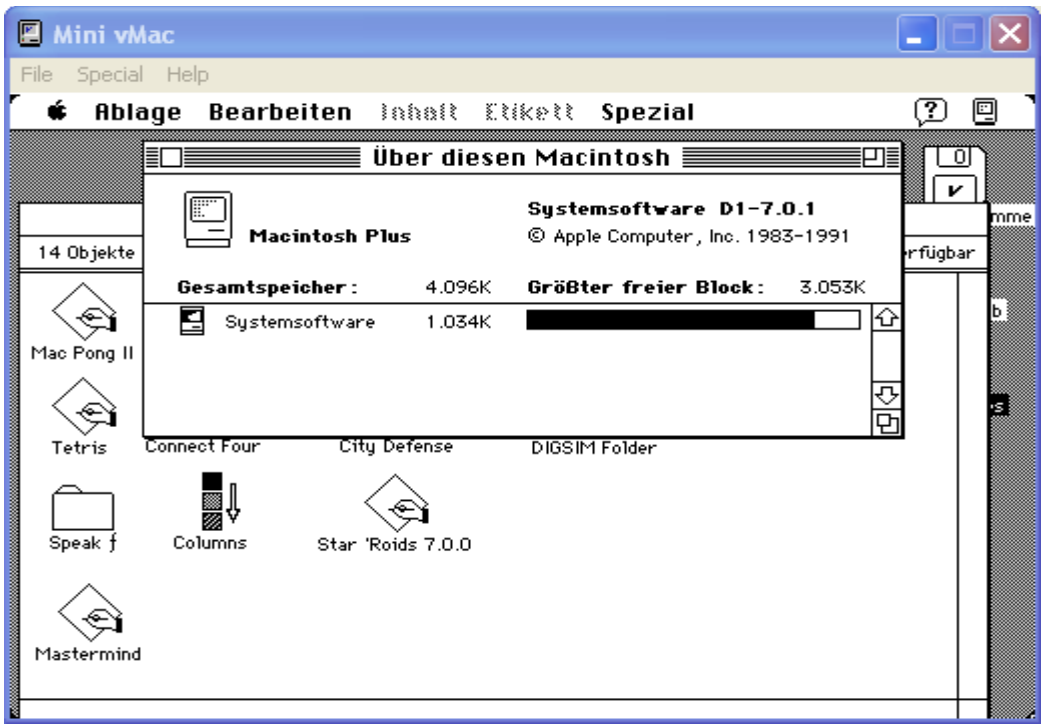
Mini vMac ist eine Abspaltung und Weiterentwicklung von vMac.

Typische Verzeichnisstruktur:

lunar_phantom_1_0.dsk	400 KB	DSK-Datei
mac_games.dsk	1.440 KB	DSK-Datei
Mini vMac.exe	92 KB	Application
nad75.dsk	1.440 KB	DSK-Datei
Psion Chess.img	800 KB	IMG-Datei
rfloppy.dsk	800 KB	DSK-Datei
StarWars.dsk	800 KB	DSK-Datei
sys701.img	1.440 KB	IMG-Datei
System Additions	1.441 KB	Datei
System Startup	1.441 KB	Datei
tooldisk.img	1.440 KB	IMG-Datei
Transferhelfer.image	1.440 KB	IMAGE-Datei
vMac.ROM	136 KB	ROM-Datei
disk1.dsk	1.440 KB	DSK-Datei

Neben dem Emulator Mini vMac.exe (Windows) wird noch die ROM Datei eines Mac Plus mit Namen vMac.ROM und ein bootfähiges Diskettenimage (siehe Links) benötigt.
Diskettenimages mit Namen disk1..6.dsk werden automatisch beim Start des Emulator geladen. Ansonsten können bis zu max.6 solches Images einfach per Drag an Drop auf den Desktop des Emulator gezogen werden, um sie in der Emulation nutzbar zu machen.

Emulationsbildschirm:



Hatari – Atari ST Emulator

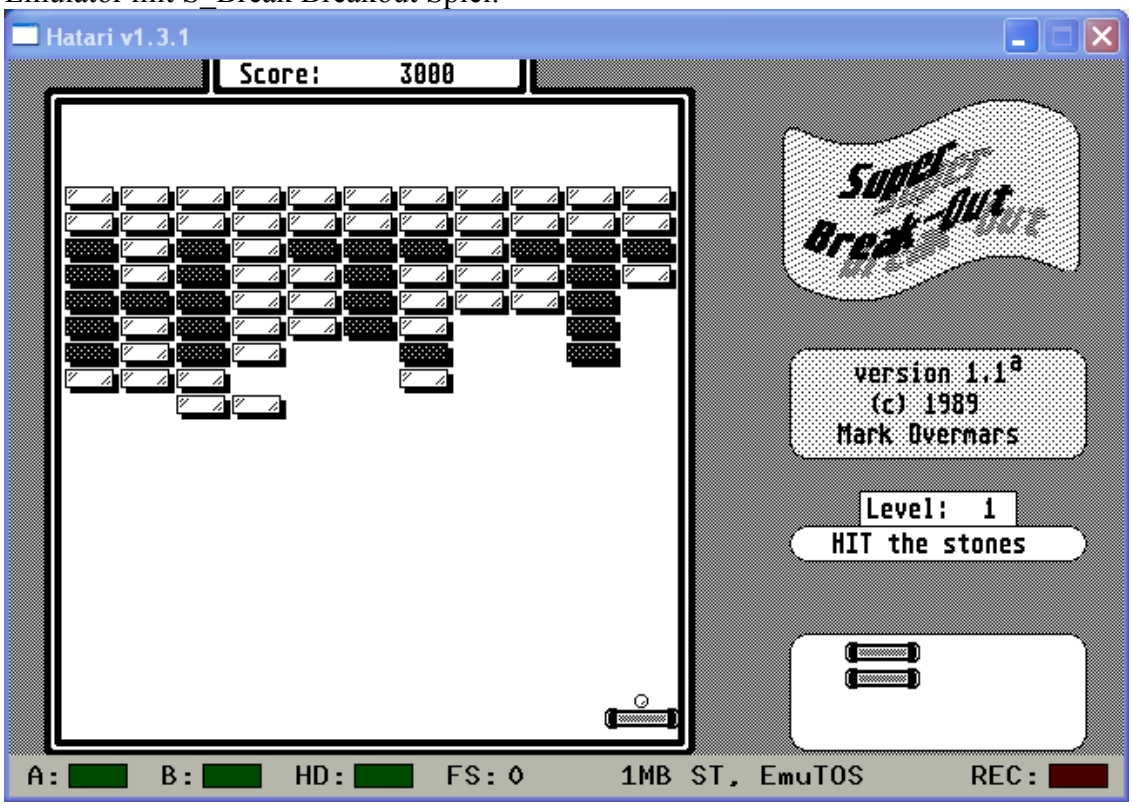
Links:
http://hatari.tuxfamily.org/
http://emutos.sourceforge.net/en/

Hatari nutzt die SDL (Simple Direct Layer) Bibliothek, die auf unterschiedlichste Plattformen portiert wurde. Zusammen mit dem EmuTOS Ersatz Projekt, bekommt man so einen schönen Atari ST Emulator. F11 schaltet auf Vollbild und zurück. F12 ruft die Konfiguration auf. Hier kann man als Festplatte auch einfach ein Verzeichnis angeben und schon ist alles was dort liegt dann auch im Emulator verfügbar.

Typische Verzeichnisstruktur mit Emulator EXE, SDL Libs und Tos.img:

gpl.txt	18 KB	Textdokument
hatari.cfg	4 KB	Config.Document
hatari.exe	1.480 KB	Application
hatari.nvram	1 KB	NVRAM-Datei
PHILOACC.ACC	17 KB	ACC-Datei
readme.txt	4 KB	Textdokument
README-first.txt	1 KB	Textdokument
README-SDL.txt	1 KB	Textdokument
SDL.dll	314 KB	Application Extension
stderr.txt	1 KB	Textdokument
stdout.txt	0 KB	Textdokument
tos.img	512 KB	IMG-Datei
zlib1.dll	98 KB	Application Extension

Emulator mit S_Break Breakout Spiel:



EAW P8000 System

Links:
http://www.robotron-technik.de
http://www.knothusa.net/
http://www.pofo.de/P8000/

Die P8000 wurde von EAW einem Betrieb der ehemaligen DDR entwickelt, ab 1987 produziert und stellte innerhalb der DDR-Computer eine eigene Rechnergattung dar. Grundanliegen war die gemeinsame Arbeit von mehreren Menschen an einem Rechner. Als Betriebssystem wurde dazu ein UNIX-Derivat namens WEGA benutzt.

Das System beinhaltet einen 8-bit Teil und einen 16-bit Teil. Letzterer wird für den Multi-User Betrieb benötigt. Bis zu 8 User konnten an seriellen Terminals mit dem System arbeiten.

Das System selbst bestand aus einer Systemeinheit mit der Elektronik plus Floppylaufwerk und einem zweiten Teil mit Festplatte und zugehörigem Kontroller.

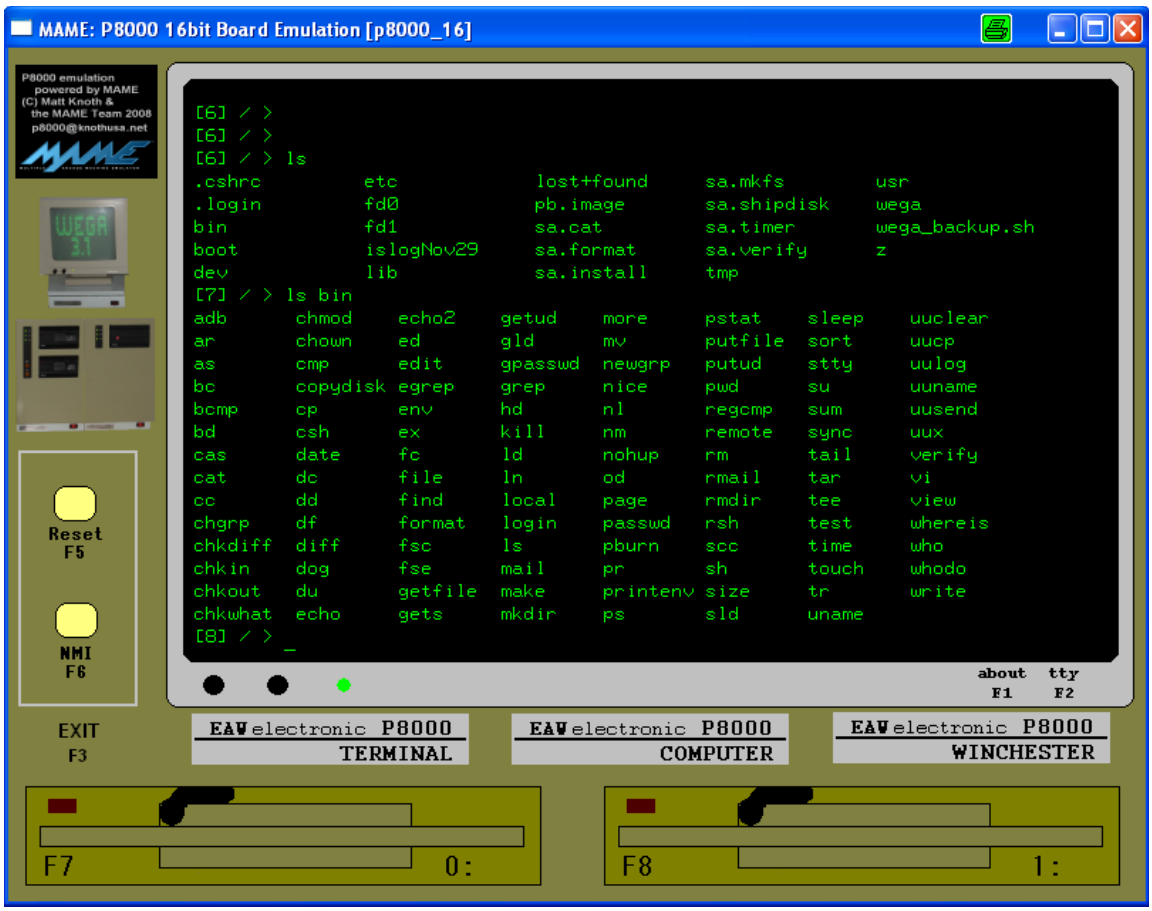
Heute existieren leider nur noch wenige funktionsfähige Exemplare.

Bild einer EAW P8000 Anlage

Der Emulator oder besser das Emulationspaket von Matt Knoth ist eine wirklich schöne Zusammenstellung eines P8000 Emulators auf Basis MAME und Dokumentationen der Hard- und Software. Login Wega: wega/root

WEGA

WEGA war das UNIX-Betriebssystem für die Computer P8000 bzw. der Compact Version der P8000 (16-Bit-Varianten) und wurde von EAW vertrieben. WEGA wurde aus der UNIX-Version 3 abgeleitet.



Ein C-Compiler gehört auch dazu. Auch durch die inkludierten Handbücher, gibt es hier viel zu entdecken und zu lernen über diese faszinierende Maschine.

JKCEMU – Emulation vieler Rechner der ehemaligen DDR.

Links:

http://www.jens-mueller.org/jkcemu/index.html

Jens Müller hat sich mit seinem erstklassigem Emulator von DDR Rechnersystemen einen Namen gemacht. Der Emulator ist dank Java portabel und unterstützt nicht nur eine Vielzahl von original Systemen, sondern auch eine Vielzahl an Zusatzhardware. Er enthält auch viele Werkzeuge (Assembler, Debugger, Hex-Editor, etc.), die beim Umgang mit den Systememulationen sehr nützlich sind.

Hier mit Emulation eines KC85/3 zu sehen. Start mit: java -jar jkcemu-0.9.2.jar
Die Einstellungen sind unter Extra.

Emulierte Systeme

- A5105(BIC,ALBA PC)
- AC1
- BCS3
- C-80
- HC900
- Hübler/Evert-MC
- Hübler-Grafik-MC
- KC85/1
- KC85/2
- KC85/3
- KC85/4
- KC85/5
- KC87
- KC compact
- Kramer-MC
- LC-80
- LLC1
- LLC2
- PC/M
- Poly-Computer 880
- Schachcomputer SC2
- SLC1
- VCS80
- Z1013
- Z9001

Integrierte Werkzeuge

- Assembler
- BASIC-Compiler
- Debugger
- Reassembler
- Hex-Dateivergleicher
- Hex-Editor
- Speichereditor
- Texteditor
- Rechner
- Datei-Browser
- Dateikonverter

Spezielle Themen

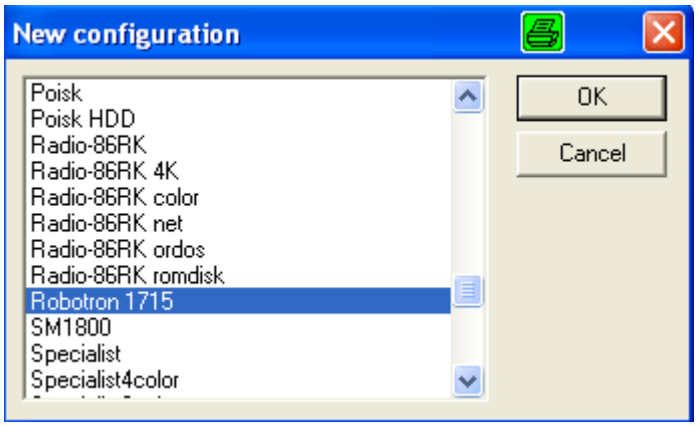
- Netzwerkemulation
- USB-Emulation
- Festplattenemulation/GIDE
- Diskettenemulation
- CP/M-Diskettenabbilddateien erstellen
- CP/M-Disketten- und -abbilddateien entpacken
- Dateien und Programme von Kassette laden
- Joysticks
- ZEXALL und ZEXDOC

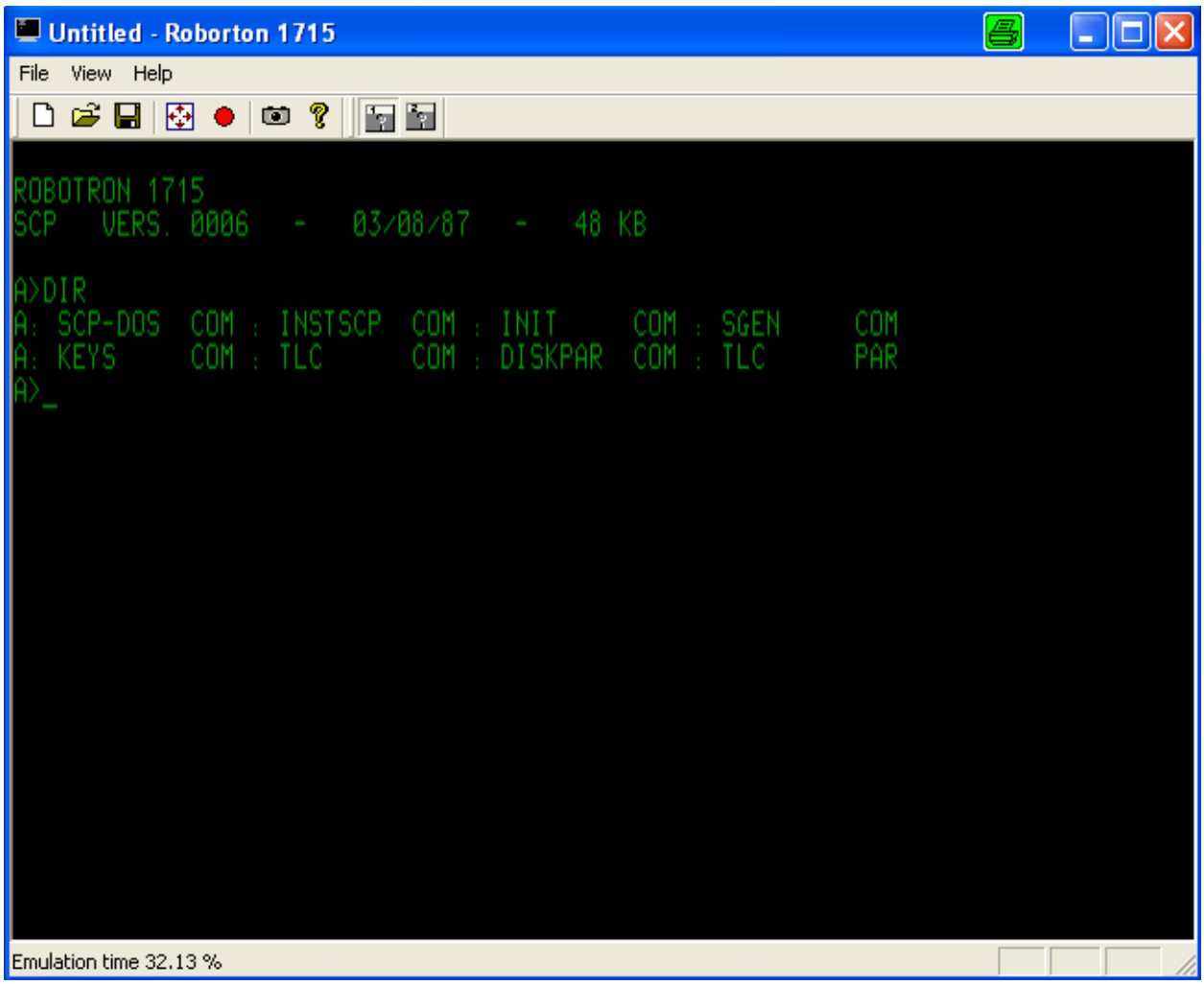
PC1715 Emulator (und andere)

Links:
http://bashkiria-2m.narod.ru/
http://bashkiria-2m.narod.ru/index/files/0-11 emu.rar und r1715fdd.rar.

Die Website ist auf russisch! Die Disketten Images müssen in den Robotron Ordner geschoben werden, danach sind sie im Emulator sichtbar.

Start von emu.exe (Hier sieht man die Vielzahl der emulierten Systeme):





Viel zu entdecken. Leider wenig Hilfestellungen durch das Programm.

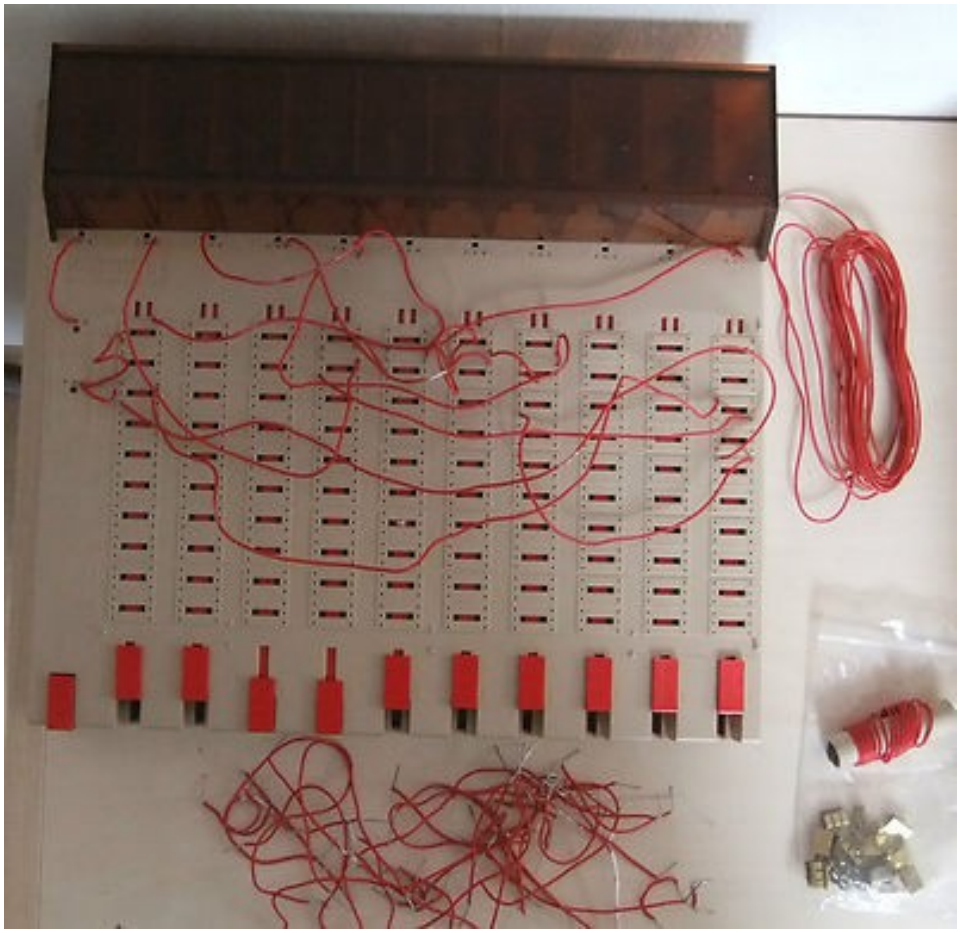
Kosmos Logikus

Links:
http://www.logikus.info/
http://petersieg.bplaced.net/?Kosmos_Logikus
http://www.winuae.net/

Der Lern-"Computer" Logikus wurde 1968 vom Kosmos-Lehrmittelverlag, Stuttgart, herausgebracht. Mit dem „Computer“ konnten auf einem Steckfeld Logik-Schaltungen (und, oder und nicht) durch Drahtbrücken aufgebaut werden. 10 Schaltschieber und ein Taster steuerten entsprechend dieser Programmierung bis zu 10 Glühlämpchen, vor die eine Transparentpapier-Schablone mit Symbolen zur einfacheren Interpretation der Programmabläufe angebracht werden konnte.

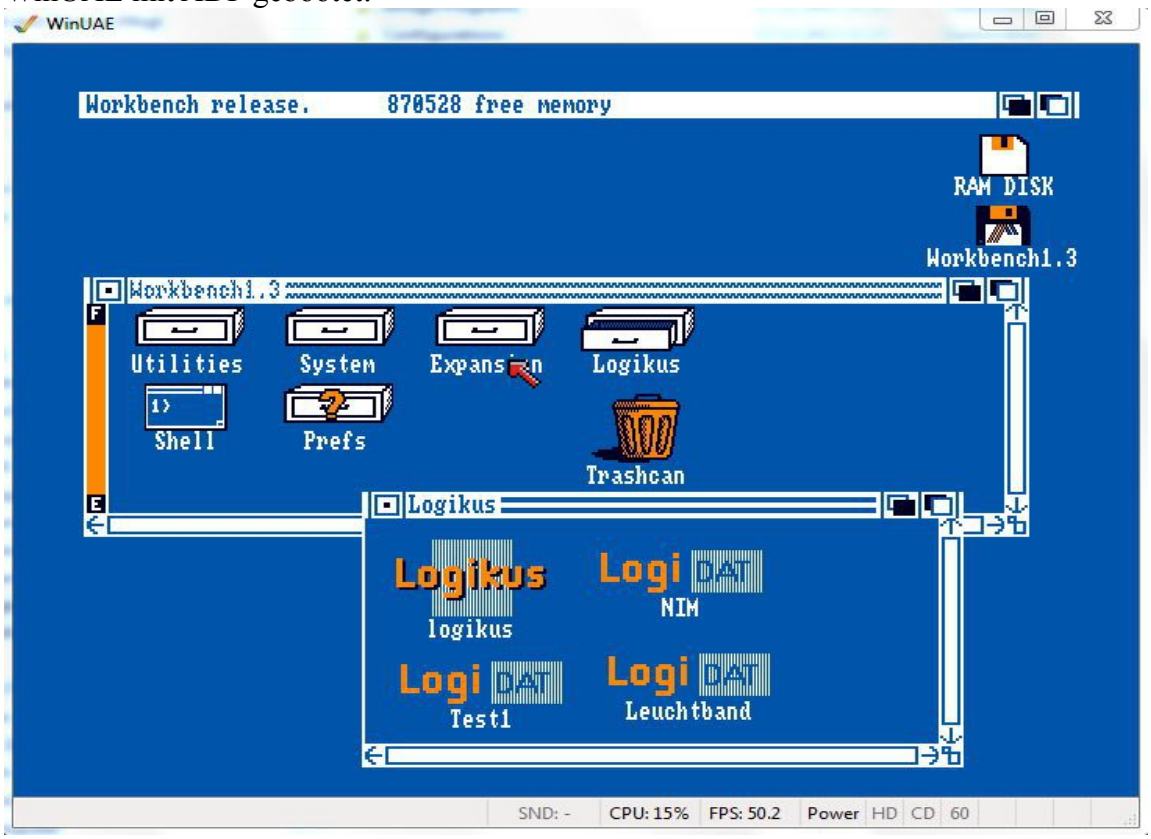
Original Karton:

Inhalt:

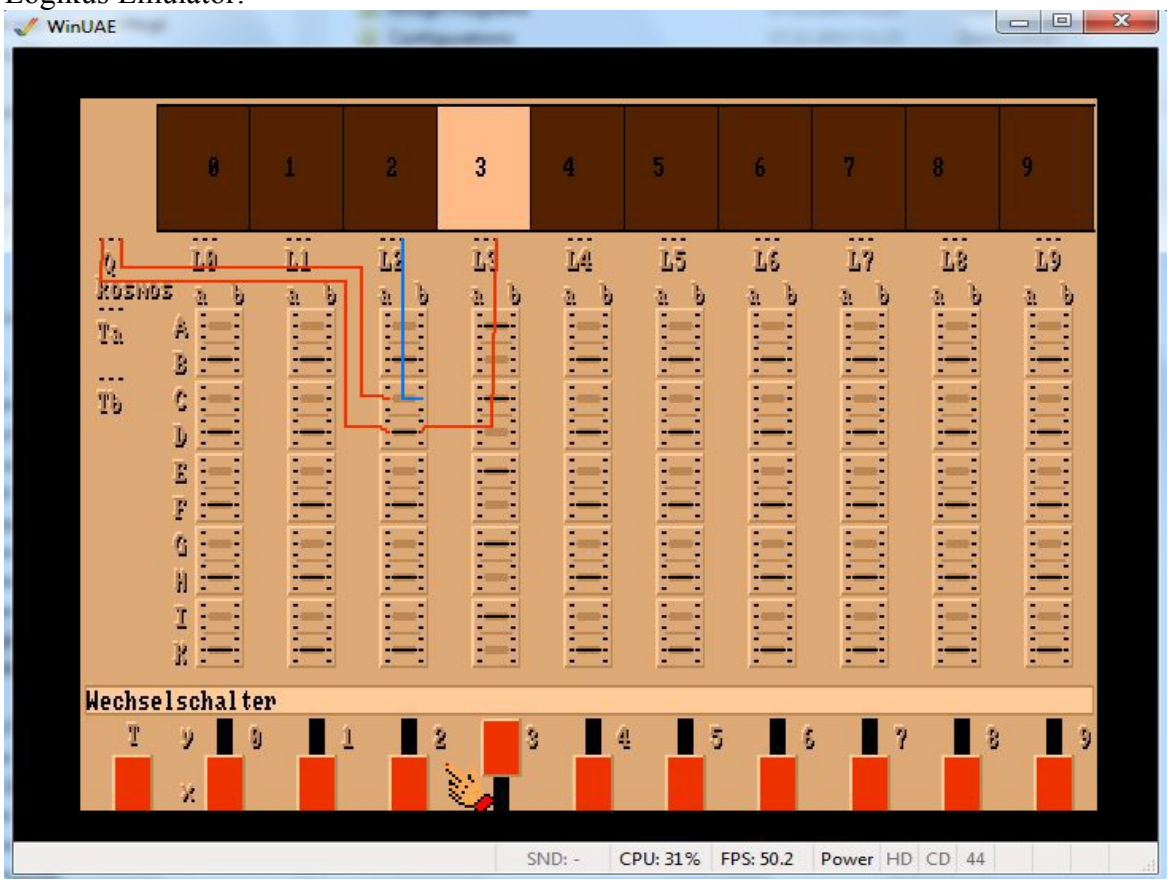


Der einzigste Emulator dazu arbeitet unter dem Amiga Betriebssystem. Daher müssen wir hier unter Windows noch einen Amiga Emulator davor setzen. So zu sagen der Emulator im Emulator. Daher zuerst WinUAE installieren/entpacken. Zum Betrieb wird hier noch ein Kickstart 1.3 ROM benötigt. Dann die ADF Diskimage Datei aus dem obigem Link einbinden. Im Ordner Logikus auf der Workbench Diskette befindet sich der Emulator und ein paar Beispiele. Mit der rechten Maustaste kann man am oberen Bildschirmrand ein Menu öffnen. Sehr schön gelöst ist bei dem Emulator die Darstellung aktiver Signalwege.

WinUAE mit ADF gebootet:



Logikus Emulator:



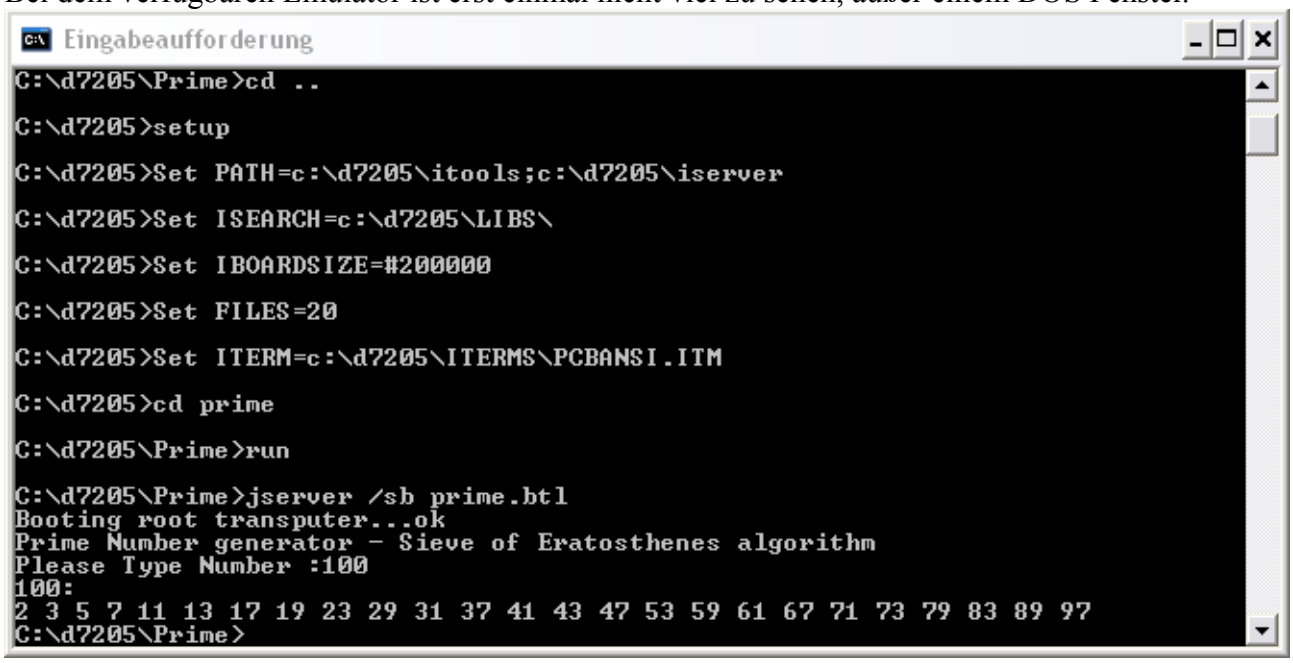
Transputer

Links:
https://sites.google.com/site/transputeremulator/
http://www.wehavemorefun.de/fritzbox/ISDN-Controller_B1_PCI
http://jonathanschilling.de/content/elektrotechnik/computerbasteln/transputer/avm_b1
http://www.classiccmp.org/transputer/software/languages/occam/compilers/inmos/commercial/
http://www.geekdot.com/avm-b1.html
http://www.classiccmp.org/transputer/

Transputer waren spezielle Chips/CPU's der Firma INMOS zur hoch parallelen Datenverarbeitung.
Transputer Chips konnten über schnelle Link-Verbindungen zu Clustern aufgebaut werden.
Als Hochsprache, die parallel Prozesse unterstützt wurde damals Occam entwickelt.
Die Chips und Rechnerboards kamen ab 1983 auf dem Markt. Heute sind solche Boards nur noch sehr schwer zu bekommen und sehr teuer – mit Ausnahme.. siehe später.

Emulation

Bei dem verfügbaren Emulator ist erst einmal nicht viel zu sehen, außer einem DOS Fenster.



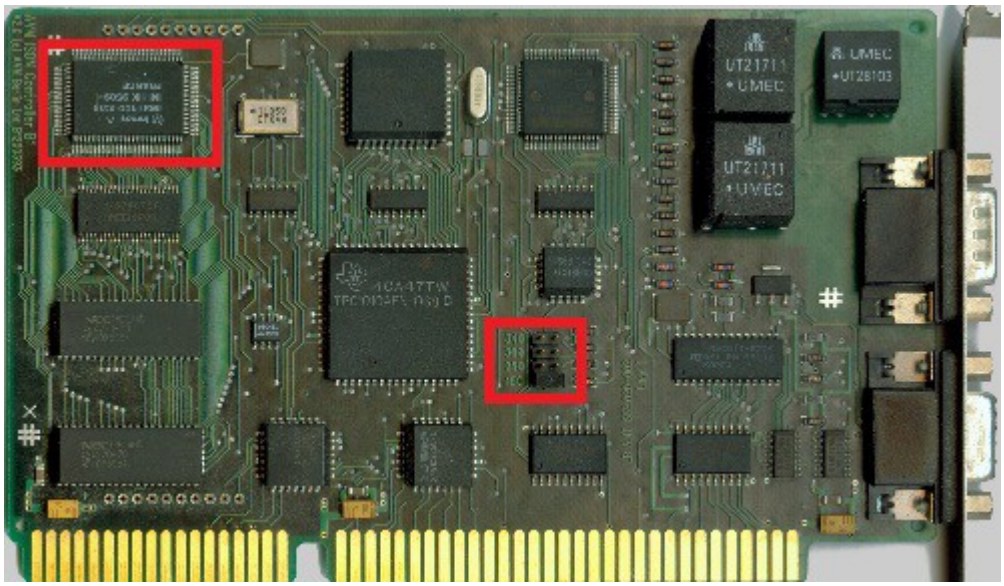
Aber unter den Links gibt es jedoch noch viel mehr zu entdecken über diese faszinierende Systemwelt der Transputer. Es gibt Raytracing Renderer, Mandelbrot Programme und vieles mehr.

The Real Thing

Viele Informationen hier sind durch die Arbeiten von **Axel Muhr** & **Jonathan Schilling** erst ermöglicht worden! Siehe auch ihre Seiten unter Links.

Normalerweise sind Transputer Karten heute nur sehr schwer zu bekommen und dann meistens sehr teuer. Aber auf einer alten ISDN Karte B1 von AVM ist ein T400 Transputer Chip drauf und die Karte ist sogar registerkompatibel zur originalen INMOS B004 Karte!

Der AVM B1 ist ein aktiver ISDN-Controller und bis Version 3.0 für den ISA-Bus gebaut.
Bis Version 3.0 wird ein Transputer vom Typ T400 verwendet. V1+V2 sind geeignet.



Auf der linken Seite der Karte findet sich ein kleines Transputer-System mit T400 Transputer Chip (rechts oben rot markiert) und 1MB RAM sowie einem TRAM-ähnlichen Erweiterungsinterface. Rechts unten auf der Karte sitzt der C011, der auf den 4 jumperbaren Adressen (Mitte unten rot markiert) vom ISA-Bus aus angesprochen werden kann. Die Standardadresse sollte 0x150 sein! Der C011 ist auf der B1 fest auf Link 0 des T400 verdrahtet. Die Links sind von AVM auf der Platine auf 10MBit/s verdrahtet.

Also alten DOS Rechner mit ISA Slots aufsetzen, Karte rein und schon hat man seinen eigenen, kleinen Transputer zuhause.

Auf: http://www.wizzy.com/wizzy/ispy.html

Gibt es die Tools ispy und mtest. Mit eingesetzter B1 Karte und Adressjumper auf 0x150 gesetzt, sollte eine Aufruf von ispy | mtest in der DOS Kommandozeile etwa folgende Rückmeldung erzeugen:

```
> ispy | mtest Using 150 ispy 3.21 | mtest 3.21
# Part rate Link# [ Link0 Link1 Link2 Link3 ] RAM,cycle
0 T400b-20 43k 0 [ HOST ... ... ... ] 2K,1 1022K,6.
```

So einfach hat man ein funktionierendes Transputersystem! Das Mandelbrot Programm kann so direkt gestartet werden und alles passt auf eine FreeDOS Bootdiskette 1,44MB.

Transputer Cluster

Das besondere an Transputer ist die Möglichkeit Transputer über ihre Links zu vernetzen und so Transputer Netze oder Cluster zu bilden.

Man kann dazu zwei AVM B1 Karten in einem PC als kleines Transputer-Cluster betreiben. Eine Karte wird auf 0x150 gejumpert, die andere z.B. auf 0x160 (damit wird sie erst einmal nicht erkannt von der PC Software). Beide Karten werden nun über drei kurze Verbindungen miteinander verbunden:

(Dabei wird vorher der Reset-Pin auf Pin 2 der 10-poligen Leiste verdrahtet, wie auf J.Schillings Seite beschrieben)

Link1in(1) <- Link1out(2) (Pin 8)
Link1out(1) -> Link1in(2) (Pin 9)
Reset(1) <-> Reset(2) (Pin 2)

Bild des Aufbaus und ispy Anzeige:

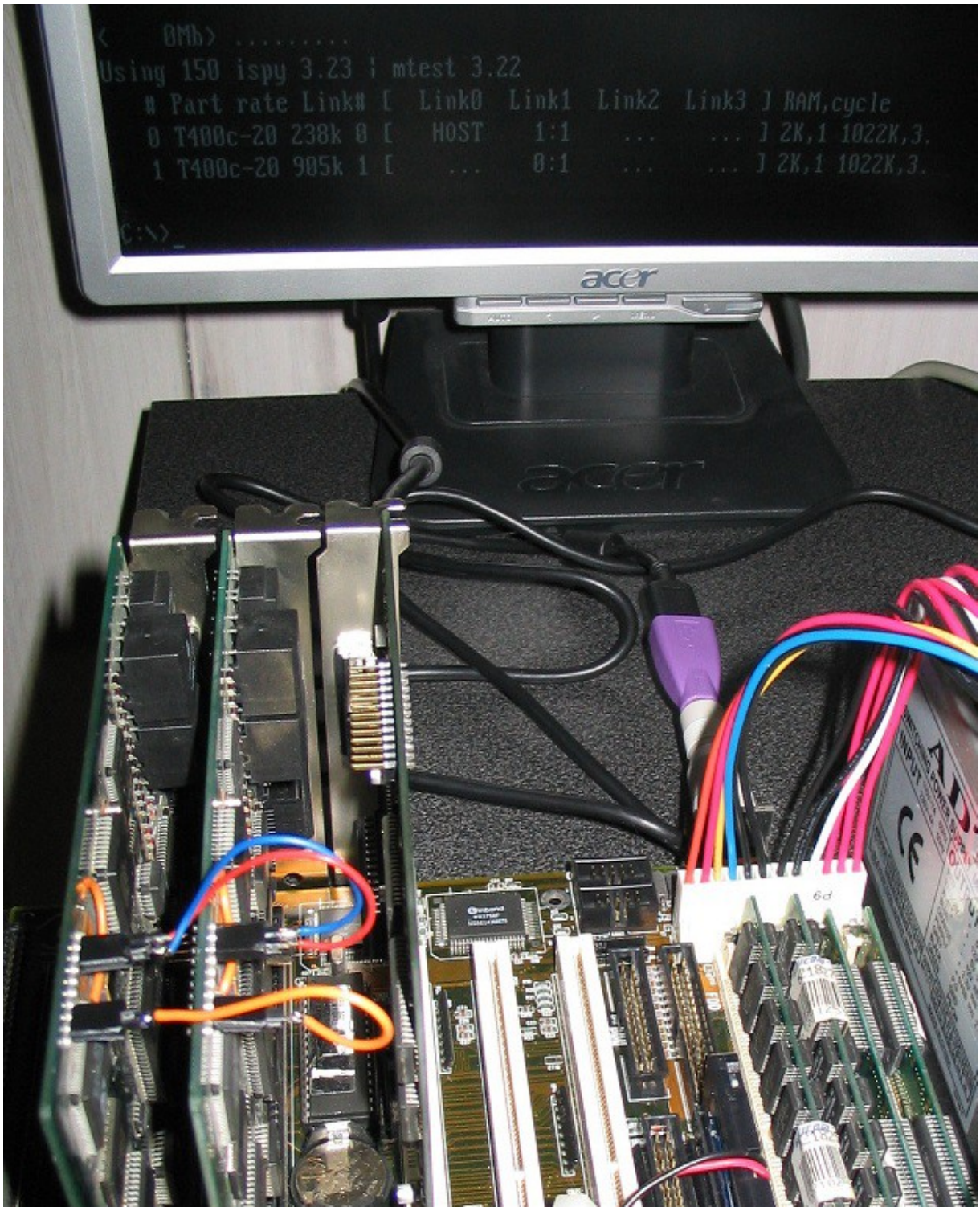
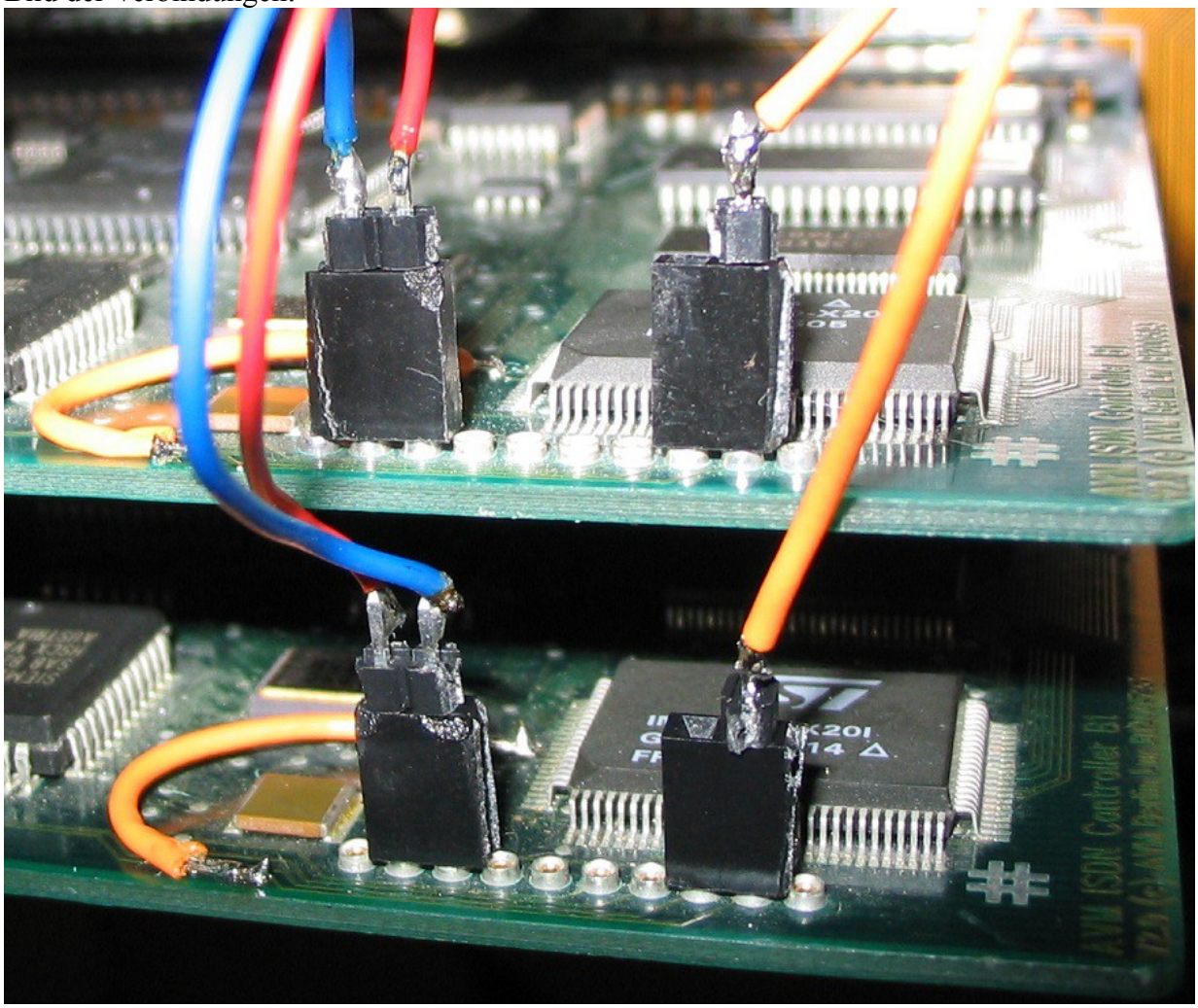


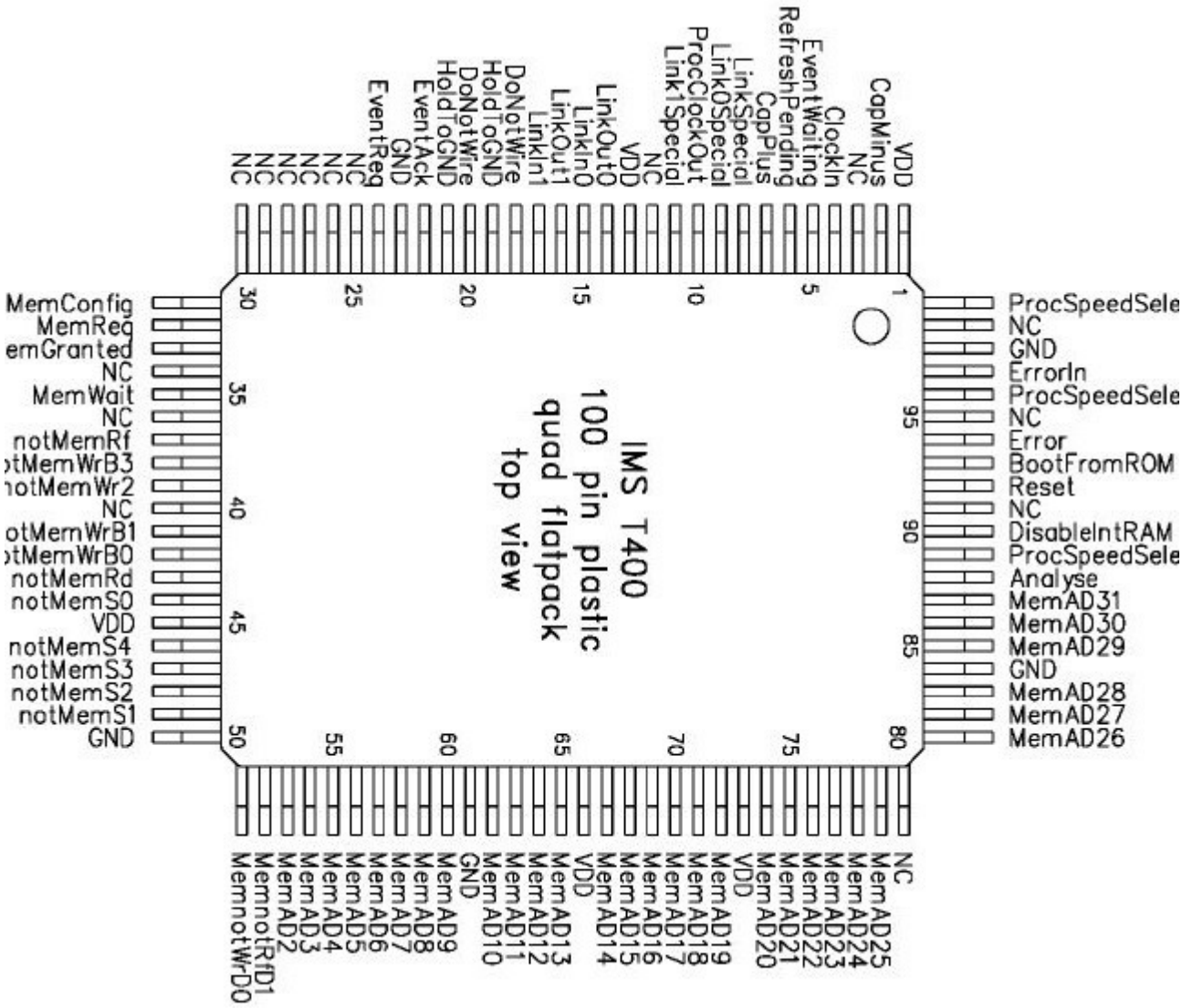
Bild der Verbindungen:



Das kann man noch weiter fortführen. Dazu müssen Karten zwischen der ersten und der letzten Karte Link 0 zum PC unterbrochen werden und zur nächsten Karte gelegt werden. Da meistens max. vier ISA Slots im PC sind, sehe die Verdrahtung so aus:

	Link 0	Link 1
Karte 1	PC	Karte 2
Karte 2	Karte 1	Karte 3
Karte 3	Karte 2	Karte 4
Karte 4	---	Karte 3

Die beiden Link 0 Signale liegen direkt links neben den Link 1 Signalen:

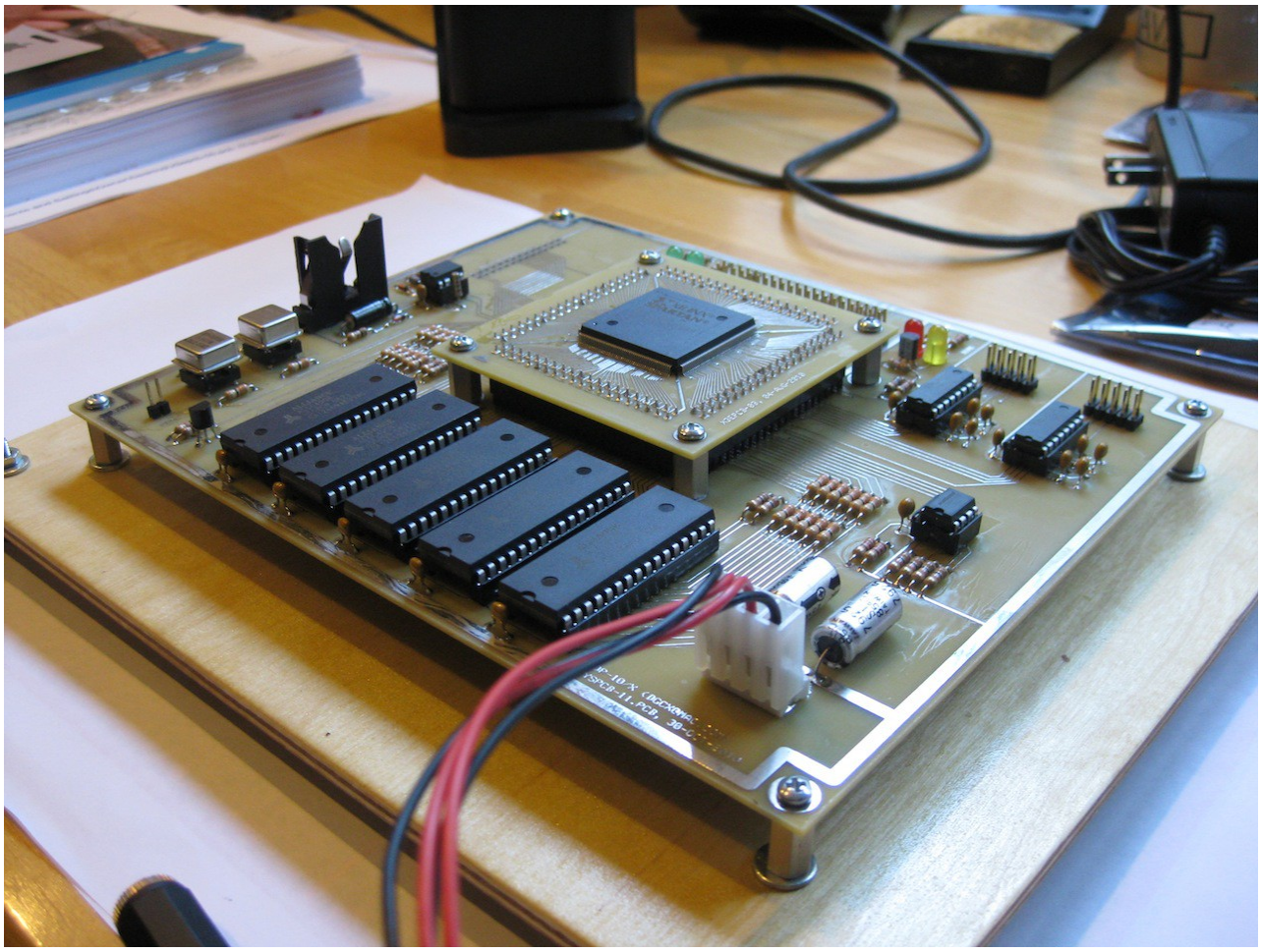


PDP-10/X

Links:
http://fpgaretrocomputing.org/pdp10x/

David G. Conroy hat eine PDP-10/X auf Basis eine Xilinx XC3S500E FPGA und SRams aufgebaut.
Das ist sein dritter Aufbau einer PDP-10 Emulation. Es setzt eine FPGA Board ein um den Chip mit seinen I/O Anschlüssen besser händelbar zu machen und eine Basisplatine mit Strom- und Taktversorgung, Rams und weiterer Peripherie.

Bild des Aufbaus:

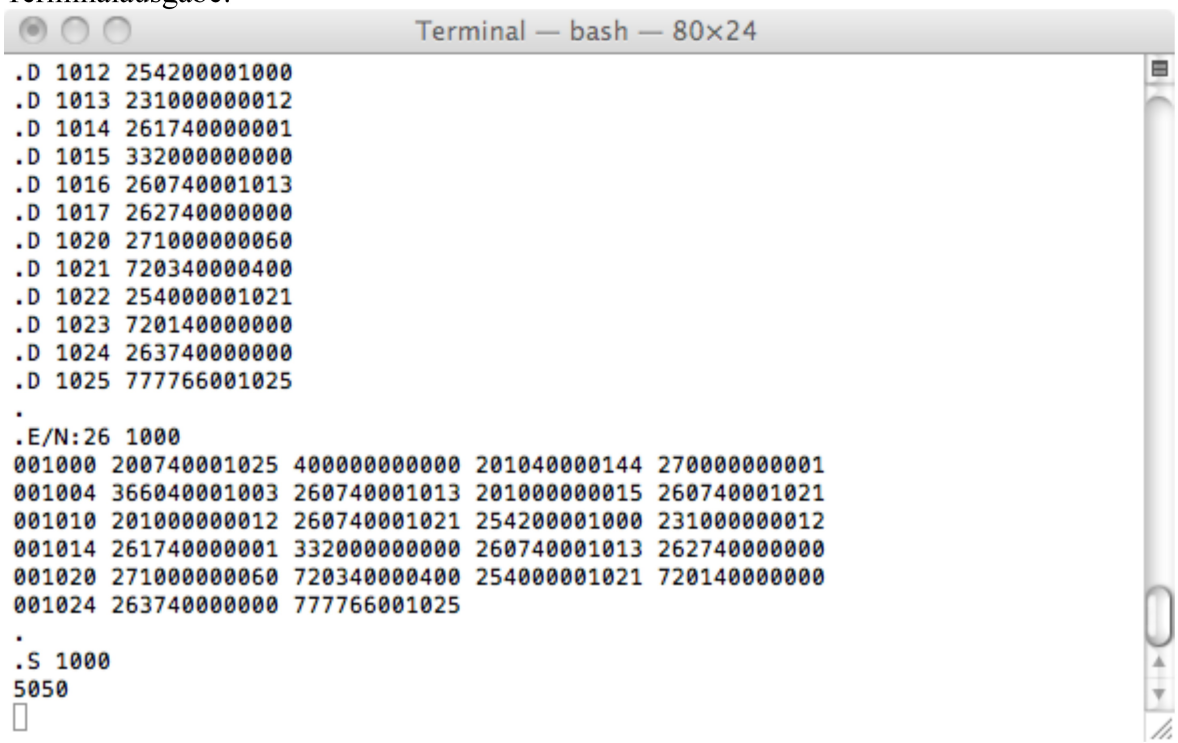


Im November 2010 bekam er zum ersten Mal einen Boot Prompt der emulierten PDP-10 und konnten über die Funktionen des ROMs ein kleines Programm eintippen, was die ersten 100 Integerwerte addierte und ausgab.

Programmcode:

```
/*
initial begin
//  001000, main
w(19'o001000, 36'o200740001025);        //  main:   move 17,pdl
w(19'o001001, 36'o400000000000);        //          setz 0
w(19'o001002, 36'o201040000144);        //          movei 1,144
w(19'o001003, 36'o270000000001);        //  x0:     add 0,1
w(19'o001004, 36'o366040001003);        //          sojn 1,x0
w(19'o001005, 36'o260740001013);        //          pushj 17,pdec
w(19'o001006, 36'o201000000015);        //          movei 0,15
w(19'o001007, 36'o260740001021);        //          pushj 17,pchr
w(19'o001010, 36'o201000000012);        //          movei 0,12
w(19'o001011, 36'o260740001021);        //          pushj 17,pchr
w(19'o001012, 36'o254200001000);        //  x1:     jrst 4,x1
//  001013, pdec
w(19'o001013, 36'o231000000012);        //  pdec:   idivi 0,12
w(19'o001014, 36'o261740000001);        //          push 17,1
w(19'o001015, 36'o332000000000);        //          skipe 0
w(19'o001016, 36'o260740001013);        //          pushj 17,pdec
w(19'o001017, 36'o262740000000);        //          pop 17,0
w(19'o001020, 36'o271000000060);        //          addi 0,60
//  001021, pchr
w(19'o001021, 36'o720340000400);        //  pdec:   conso tty,400
w(19'o001022, 36'o254000001021);        //          jrst pchr
w(19'o001023, 36'o720140000000);        //          datao tty,0
w(19'o001024, 36'o263740000000);        //          popj p,
//  001025, -10,s-1
w(19'o001025, 36'o777766001025);        //  pdl:    xwd -10,s-1
end
*/
```

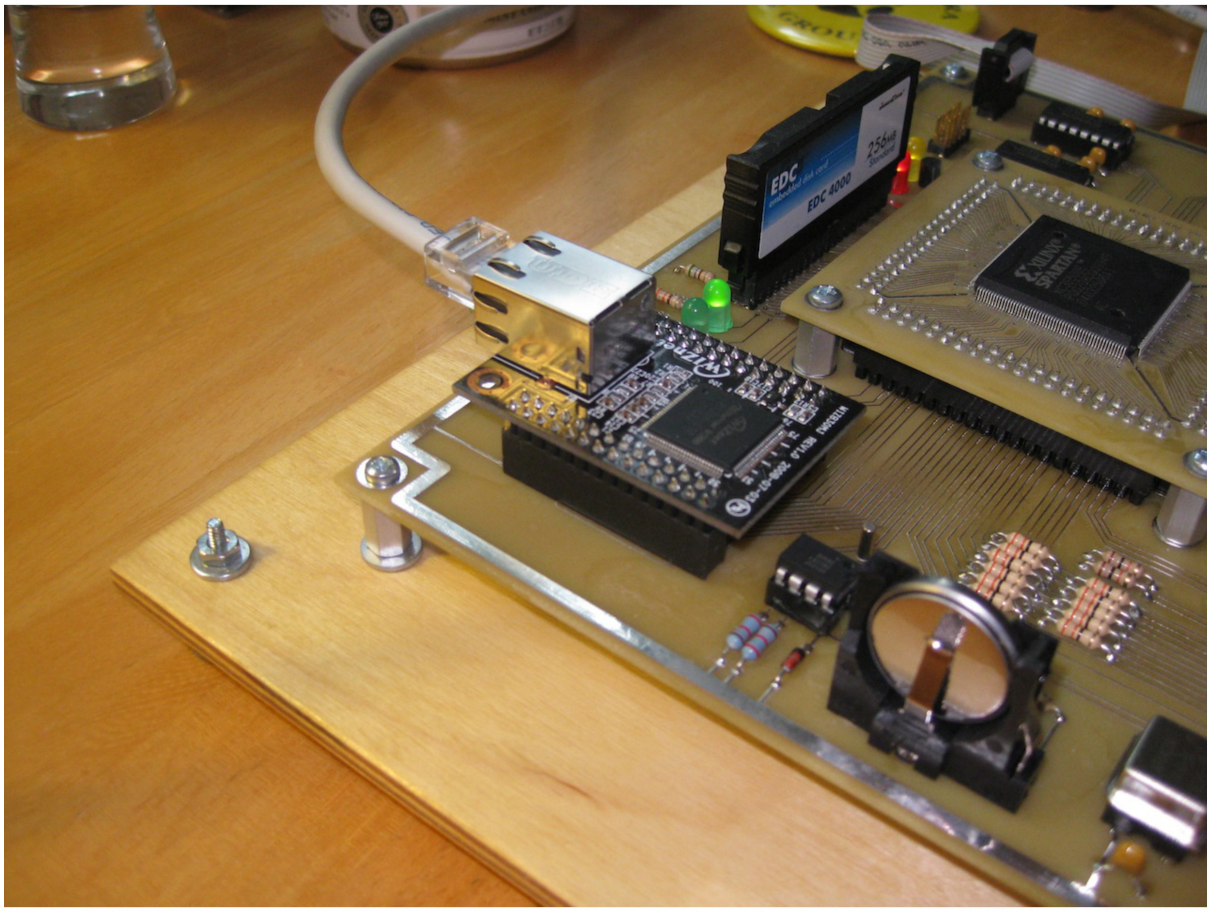
Terminalausgabe:



Im Dezember 2010/Januar 2011 wurde mittels eines 250MB DOM Moduls das Betriebssystem ITS Erfolgreich gebootet.

```
GlassTeletype: /dev/cu.KeySerial1
ROM10X.84
.B
.............................................................

Salvager 266

.TEMP. has no files, User File Directory DELETED


TX ITS 1666 IN OPERATION AT 18:00:21
TX ITS 1666 SYSTEM JOB USING THIS CONSOLE.
IT IS NOW  6:00:21 PM EST, FRIDAY, DEC 31,2010
IT IS NOW  6:00:50 PM EST, FRIDAY, DEC 31,2010
```

Inzwischen wurde CHAOSNET implementiert und insgesamt die Emulation weiterentwickelt.



Cray 1

Links:
http://www.chrisfenton.com/homebrew-cray-1a/
http://modularcircuits.tantosonline.com/blog/articles/the-cray-files/

Die Cray-1 war der erste Supercomputer der Firma Cray, dessen Architektur vom Team um Seymour Cray entwickelt wurde. Seymour Cray war dabei für die Technologie der Vektor-Register zuständig. Die erste Cray-1 wurde 1976 am Los Alamos National Laboratory in Betrieb genommen.

Mit Preisen zwischen 5 Mio. und 8 Mio. $ wurden ungefähr 80 Cray-1 weltweit verkauft.

Die Cray-1A wog einschließlich des Kühlsystems über 5 Tonnen. Um die Kabellängen innerhalb des Gehäuses kurz zu halten wurde sie in einer Hufeisenform gebaut. Die Cray-1A war ein Vektorrechner. Ausgestattet mit einer für damalige Verhältnisse enormen Taktfrequenz von 80 Megahertz, 64 Vektor-Registern in der Wortbreite von 64 Bit sowie einer Million sehr schnellen 64 Bit breiten Speichern (entsprechend 8 MB RAM), erreichte die Cray-1A über 80 Millionen Gleitkommazahl-Operationen pro Sekunde (FLOPS).

1978 wurde das erste Standardsoftwarepaket für die Cray-1 herausgegeben. Es bestand aus einem Betriebssystem, dem Cray Operating System (COS), der Cray Assembler Language (CAL) und Cray Fortran, einem Fortran-Compiler, der als erster vollautomatisch vektorisieren konnte.

Cray 1 Emulator in FPGA

Die Seiten von Chris Fenton lesen sich wie ein guter Krimi. Chris dachte sich, man baut erst einmal einen Emulator nach Hardwarebeschreibungsdokumenten mit einem FPGA Board und danach sucht man sich einfach Software zum testen im Internet.

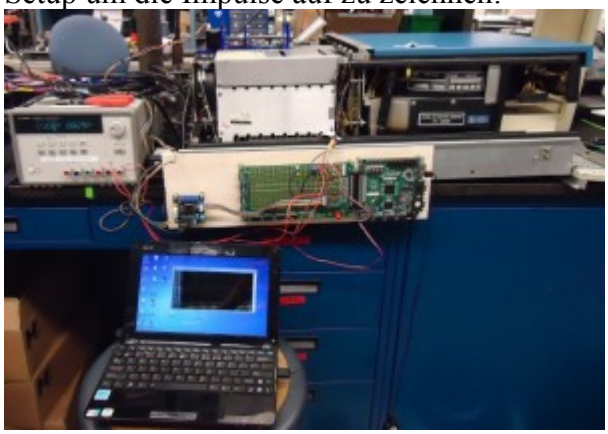
Falsch gedacht! Cray Computer wurden von US Behörden mit 3 Buchstaben gekauft und diese halten sich sehr bedeckt, Programme und Software heraus zu geben. Cray und dessen Nachfolge Firmen hatten wohl auch nichts mehr an Sicherungen. Und so startete eine Odyssee um an irgendeine Software zu kommen.

Was Chris bekam war dann ein 80MB Plattenstapel mit Software drauf. Aber kein entsprechendes Laufwerk und kein System um daran dann das Laufwerk anzuschließen.

Plattenstapel:

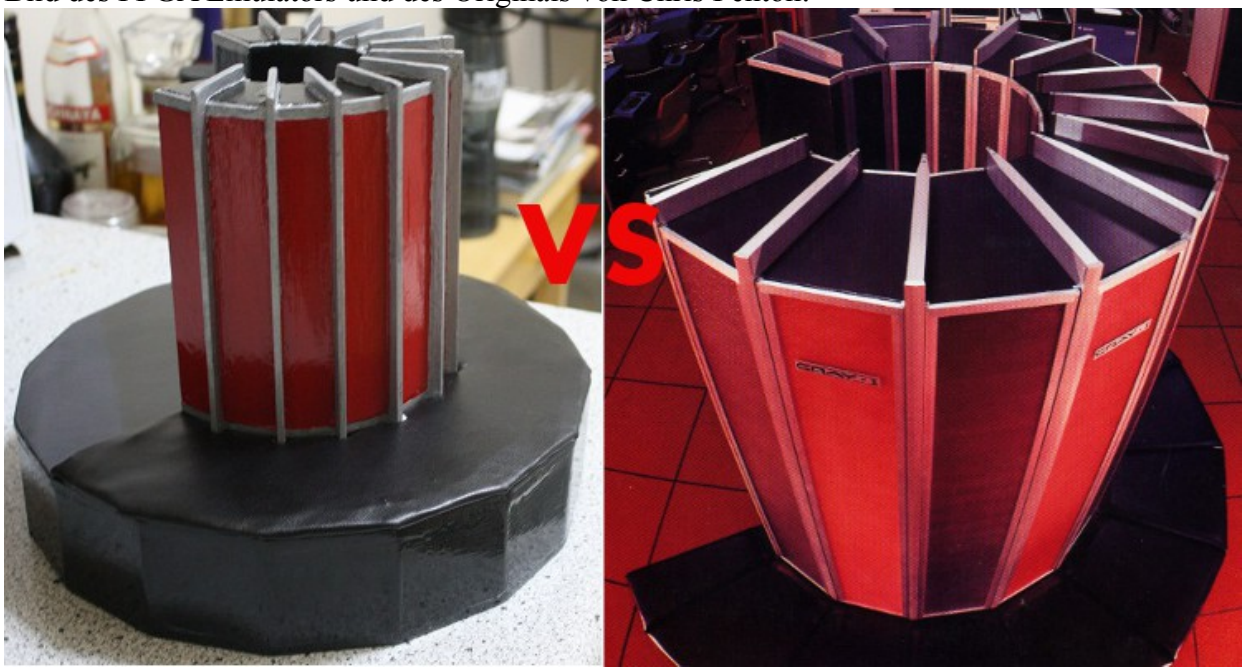
Nach vielen fehlgeschlagenen Versuchen, digital an das Laufwerk ein Interface zu bauen, loggte er anschließend die magnetischen Impulsfolgen mit und sicherte es dann in einer 35GB großen Rohdatei.

Setup um die Impulse auf zu zeichnen:



Und dann kam das Internet. Yngve AAdlandsvik aus Norwegen brachte es fertig diese Rohdaten soweit aufzubereiten, das später andere es letztendlich wieder in teilweise nutzbare Software zurück verwandeln konnten!

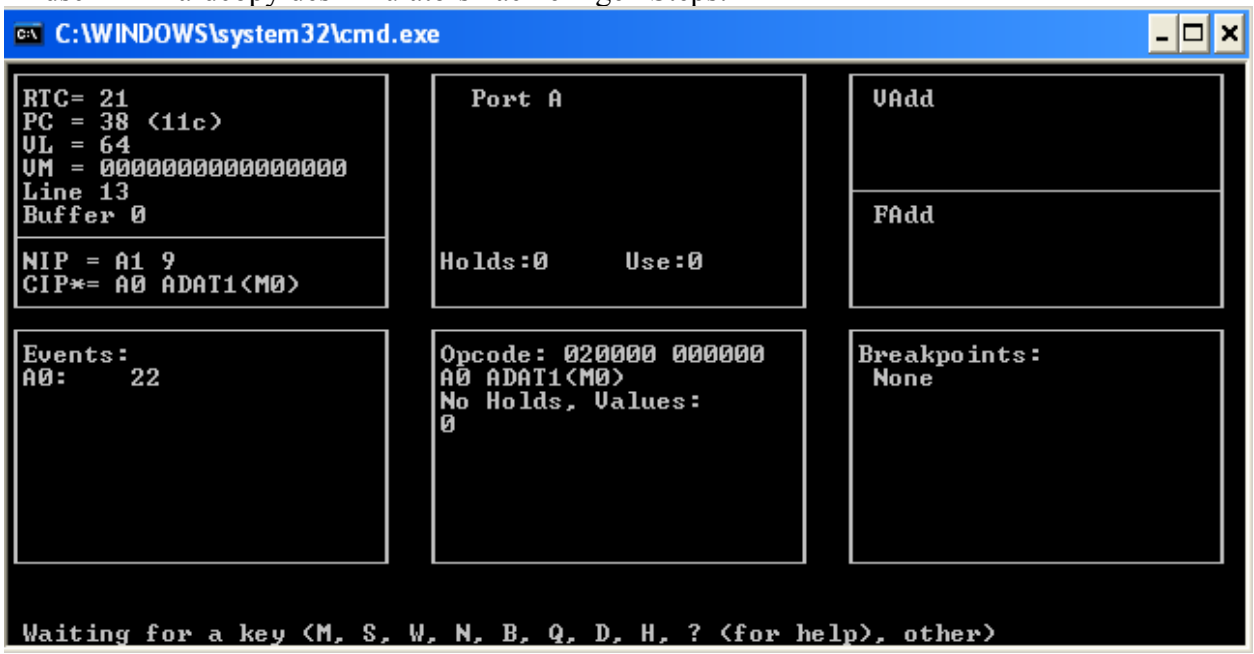
Bild des FPGA Emulators und des Originals von Chris Fenton:



Auf Chris Seiten gibt es neben einem Assembler (CAL) auch einen DOS Emulator einer Cray YMP.

Aufruf des DOS Emulators: xmpsim sim1.in sim1.out

Bildschirm Hardcopy des Emulators nach einigen Steps:



Man möge mich bitte nicht fragen was hier wie funktioniert. Ohne tieferes Einsteigen in diese faszinierende Welt der Supercomputer Historie wird das wohl immer ein Geheimnis bleiben.

Cray 1 Emulator von Andras Tantos

Andras Tantos hat die von Chris Fenton erzeugten Rohdaten und die Aufbereitungen durch Internet Nutzer – insbesondere durch Yngve AAdlandsvik – wieder in nutzbare Software verwandelt und dann noch gleich einen Emulator geschrieben der dies alles in eine Cray Umgebung auf einem PC verwandeln. Dabei werden 4 Terminalsession geöffnet. Das Betriebssystem ist IOP. Anwendersoftware wird aber immer noch verzweifelt gesucht.

Aufruf:
cray_xmp_sim xmp_sim.cfg

Bildschirmkopie:

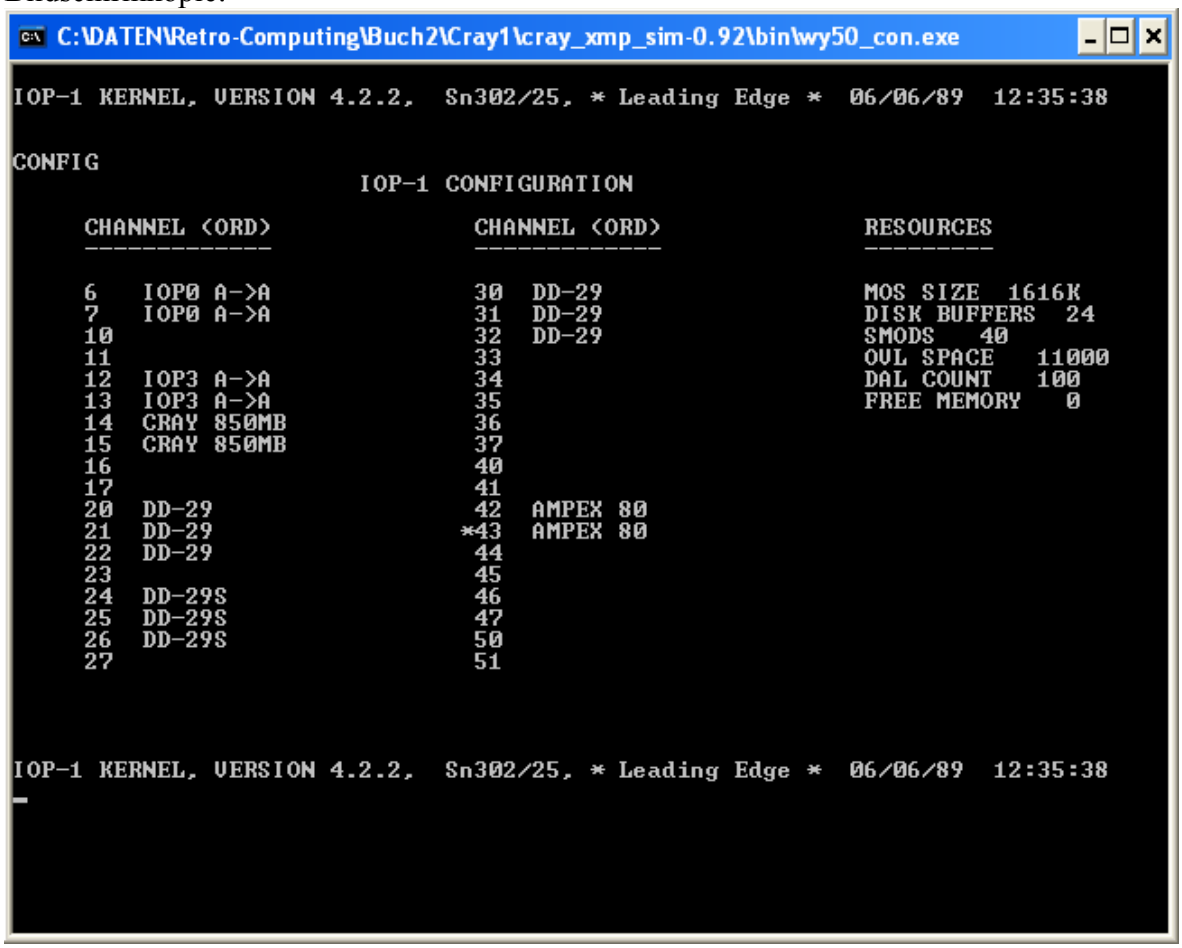


Bild einer Cray X-MP:

Vectrex

Links:

http://www.vectrexnews.com
http://www.vectrex.nl
http://www.pelikonepeijoonit.net/vec/index.html
http://www.playvectrex.com/designit/chrissalo/toc.htm
http://www.vectrex.biz/
http://geovector.tripod.com/_sgg/f10000.htm
http://www.valavan.net/vectrex.html
http://mitglied.lycos.de/vectrex/start.htm
http://www.emix8.org/static.php?page=VectrexThrustSource
http://www.madtronix.com/html/vectrex.html
ftp://ftp.csus.edu/pub/vectrex/

Das Vectrex ist eine Spielkonsole, die 1982 auf den Markt kam. Auffälligstes Merkmal ist der eingebaute Hochformat-S/W-Bildschirm zur Ausgabe der Vektorgrafik. Dieser Kompaktaufbau führte zur Einstufung durch Fachzeitschriften in eine eigene Kategorie: Mini-Arcade. Federführender Entwickler war Jay Smith, welcher bereits 1979 das Microvision Handheld für MB entwarf. Hergestellt und veröffentlicht wurde die Konsole in den USA von General Consumer Electric (GCE) ab Oktober 1982. In Europa und Japan übernahm MB den Vertrieb. 1984 stellte MB den Vertrieb ein.

Der Rechnerteil bestand aus einem mit 1,5 MHz getakteten Motorola 6809-Mikroprozessor mit 1 KB RAM und 8 KB ROM (4 KB 'Executive' und 4 KB für Minestorm Spiel). Als Soundchip kam ein AY-3-8912 von General Instrument zum Einsatz. Der Bildaufbau erfolgte komplett CPU-gesteuert, d.h. die CPU steuerte in Echtzeit das Zeichnen auf dem Schirm über eine X/Y-Ansteuerung der Bildröhre (Vektorgrafik anstatt Rastergrafik).

Eine weitere Besonderheit ist es, das heute alle Rom's auch der Spiele seitens des Herstellers GCE zur freien Verfügung frei gegeben wurden! Das heißt, man darf die Rom's mit Emulatoren bundeln und zusammen zum Download anbieten. Das erleichtert alles ungemein und man muss nicht erst umständlich Romdump's erstellen etc.

Als Emulator eignet sich MESS. Oder auch - ohne Sound - Vecx. Portabel und inkl. Sourcecode in C! Läßt sich so mit mingw compilieren. (Tasten: a,s,d,f, csr-links+rechts) Für DirectX (Benötigte Header zum compilieren gibts bei www.sdl.org). Als Assembler eignet sich a09.

Vecx

Core emulator vecx von Valavan Manohararajah.

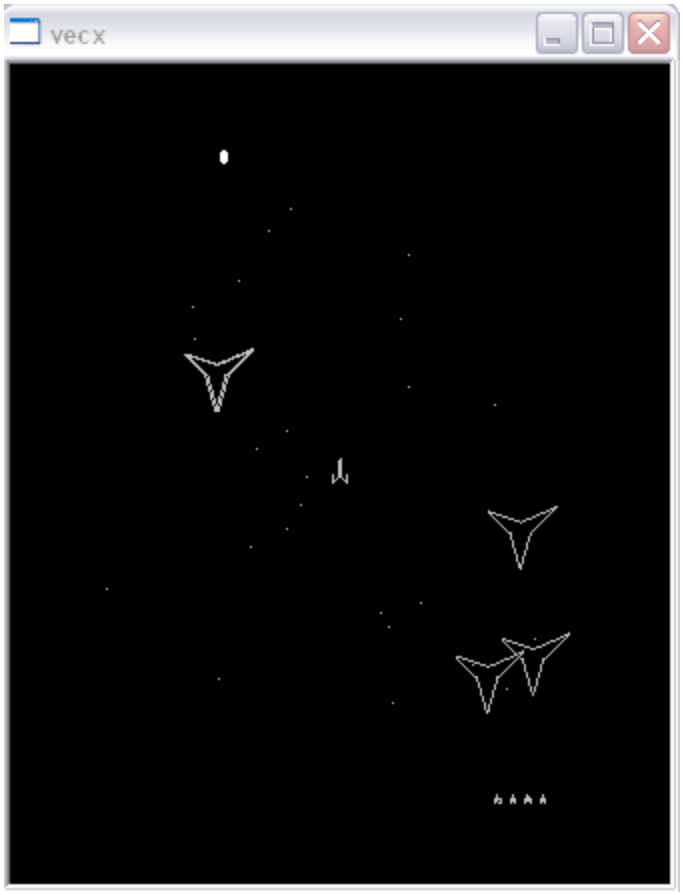
Homepage: http://www.valavan.net

Wie man sehen kann, ein kleiner und feiner Emulator:

Name	Größe	Typ
e6809.c	49 KB	C Source File
e6809.h	1 KB	H-Datei
Makefile	1 KB	Datei
osint.c	14 KB	C Source File
osint.h	1 KB	H-Datei
rom.dat	8 KB	DAT-Datei
vecx.c	20 KB	C Source File
vecx.exe	40 KB	Anwendung
vecx.h	1 KB	H-Datei

vecx.exe started den Emulator mit dem eingebauten Spiel Minestorm.

Vecx.exe -c <cartridge> started den Emulator mit dem Rominage cartridge.



Wer selbst mal ein Spiel dafür entwickeln möchte. Als Assembler eignet sich z.B.:

A09 - 6809 / 6309 Assembler

Copyright (c) 1993,1994 L.C. Benschop

Parts Copyright (c) 2001-2005 Hermann Seib

Based on Lennart Benschop's C core that can be found somewhere on the 'net (last address known to me was http://koti.mbnet.fi/~atjs/mc6809/Assembler/A09.c), I built a complete Macro Assembler that is functionally better than the TSC Flex9 Assembler (no wonder, I got multimegabytes to play with, whereas this excellent piece of software works within 50K or so!). It can deliver binary, Intel Hex, Motorola S1, and Flex9 binary output files, plus Flex9 RELASMB-compatible relocatable object modules.

I taylored the original to my taste by working through the source code; since A09 has reached a level of complexity that doesn't really lend itself to following the "Use the Source, Luke!" principle if you just want to hack a little 6809 assembly program, I've added this documentation. Have fun!

Hier ein kleines A09 Assembler Testprogramm:

```
; Draws a figure

; Based on code by Christopher Tumber

; Based on work by man@sci.fi

; (c) 2007 Peter.Sieg1@gmx.de

; Public Domain for all Vectrex freaks

; First we rename all the BIOS routines

; we'll be needing
```

```
waitrecal             EQU     $f192
movepen7ftod          EQU     $f2fc
intensitytoA          EQU     $f2ab
movedrawVL4           EQU     $f3b7
reset0ref             equ     $f354
delayb                equ     $F57A

; *** Needed variables
value1                equ     $c881
value2                equ     $c882
value3                equ     $c883
value4                equ     $c884

; You can give them your own names if you want to
; just make sure you understand them, and preferably
; that someone else reading your source code
; understands them as well. (EQU = equals)

; All programs start from address $0000
; In this example, all numbers are actual numbers
; not hexadecimals like in the tutorials
; Assembly is quite capable of handling real numbers
: and they are much more easy for a beginner to
; understand
        ORG     $0000

; Init block that needs to be in every program
; the GCE text has to be in place, it is checked
; by BIOS
        FCB     $67,$20
        FCC     "GCE SIEG"
        FCB     $80
        FDB     music
        FDB     $f850
        FDB     $30b8
        FCC     "Z-ROTATING VECTOR"
        FCB     $80,$0

; Here's the main program
; that will draw some vectors
```

```
drawvec:                    ; Main program's label
        lda   #32
        sta   value1
        sta   value2
        sta   value3
        sta   value4
fig:
        lda   value1
        deca
        sta   value1
        cmpa  #0
        beq   fig1
        jsr   restore
        ldx   #figur0       ; Gets the vector list
        jsr   draw
        bra   fig
fig1:
        lda   value2
        deca
        sta   value2
        cmpa  #0
        beq   fig2
        jsr   restore
        ldx   #figur1       ; Gets the vector list
        jsr   draw
        bra   fig1
fig2:
        lda   value3
        deca
        sta   value3
        cmpa  #0
        beq   fig3
        jsr   restore
        ldx   #figur2       ; Gets the vector list
        jsr   draw
        bra   fig2
fig3:
        lda   value4
        deca
        sta   value4
        cmpa  #0
        beq   drawvec
```

```
        jsr   restore
        ldx   #figur1         ; Gets the vector list
        jsr   draw
        bra   fig3
; end of main program
; subroutine section starts here
restore:
        jsr   waitrecal       ; Resets the BIOS
        jsr   reset0ref
        lda   #127            ; Gets the Intensity
        jsr   intensitytoA    ; Sets the Intensity

        lda   #0              ; Y - coordinate
        ldb   #0              ; X - coordinate
        jsr   movepen7ftod    ; Moves pen to (Y,X)
        rts
draw:
;       ldx   #figurN         ; Gets the vector list
        lda   #3              ; Number of vectors
                              ; = how many vectors
                              ; will be drawn, the
                              ; starting point isn't
                              ; counted
        ldb   #$80            ; gets the scale
        jsr   movedrawVL4     ; BIOS-routine that
        jsr   delay
        rts
delay:
        ldb   #255            ; delay value
        jsr   delayb          ; wait a while
         rts
; It jumps back, because on Vectrex, you have to be
; drawing all the time to keep the vectors visible
; bra = branch
; lda = load to register a
; ldb = load to register b
; ein Dreieck an 50,0
figur0: fcb   50,0            ; 0,0 = Bildschirm-Mitte
        fcb   -32,32          ; y=-32=gehe 32 Pixels nach unten und x=32=gehe 32 Pixels nach rechts
        fcb   0,-64           ; y=0=bleibe an y-Position und x=-64=gehe 64 Pixels nach links
        fcb   32,32           ; y=32=gehe 32 Pixels nach oben und x=32gehe 32 Pixels nach rechts
figur1: fcb   50,0
```

```
        fcb   -32,16
        fcb   0,-32
        fcb   32,16
figur2: fcb   50,0
        fcb   -32,0
        fcb   0,0
        fcb   32,0
music:
       FDB   $fee8
       FDB   $feb6
       FCB   $28,$08
       FCB   $0,$80
; This basicly tells Vectrex to play nothing
```

Vecxsdlsnd

Links:

http://petersieg.kilu.de/vectrex/vectrex.html

Vectrex Emulator mit Overlays, Gameroms (Frei gegeben!) + Sound! SDL+Overlay code by Thomas Mathys. Sound emulation code by James Higgs. Homepage: http://jum.pdroms.de

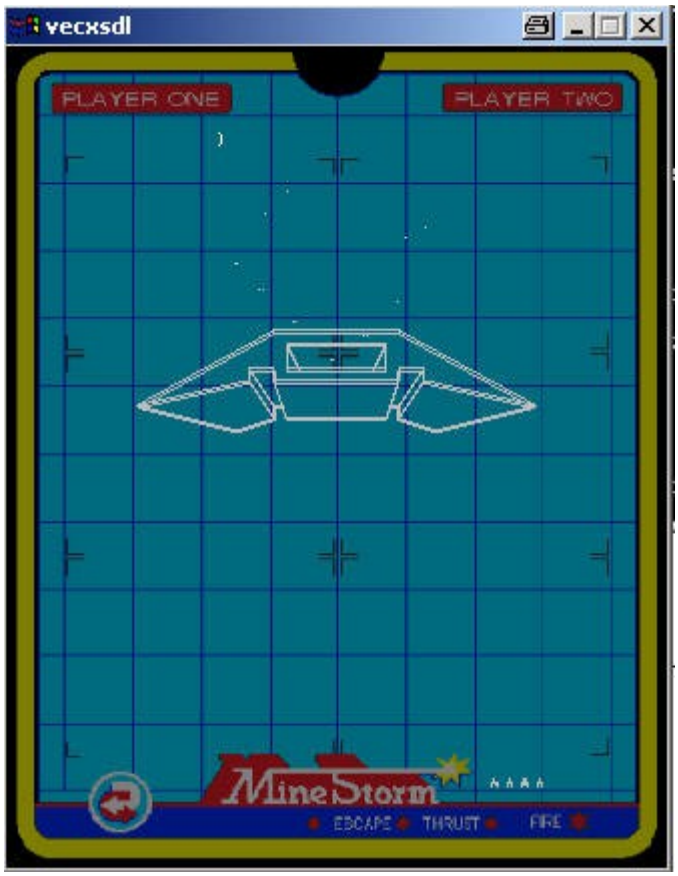
E-Mail von: James Higgs am: Thu, 06. Dec 2007 05:06:25 -0800

Hi Peter

I'm glad it worked so easily. I think this is mostly due to how vecx was originally written to work cross-platform. I don't mind if you release your version of vecxsdl on

your site. My code can be GPL, however I don't know what the other authors want for their code.

Regards, James Higgg



Meine Multicard

Hier beschreibe ich meine Versuche, eine sog. Multicard zu bauen. Als einziges Modul zum 'schlachten' hatte ich Blitz (8k). Zun öffnen einfach die Schraube an der Unterseite lösen und die Plastiknasen/-haken an der Einschubseite lösen. Ich wollte zuerst das ROM sauber auslöten. Hatte schon entsprechende Erfahrung mit dem Auslöten von IC's beim C64 gesammelt. Aber ich hatte nichts, um diese kleine Platine vernünftig zu fixieren.. so gings gar nicht. Also habe ich mich kurzer Hand entschieden, die PIN's nah an der Platine abzukneifen und einen passenden Sockel einfach aufzulöten. Dabei immer wieder kontrollieren, das alles korrekt verbunden ist! Es sollen 16 x 4k Programmbereiche über 4 Steckbrücken/Schalter wählbar sein.

Cartridge slot Belegung:

OBEN	UNTEN
2. Vcc (+5v)	1. /HALT
4. Vcc (+5v)	3. A7
6. A8	5. A6
8. A9	7. A5
10. A11	9. A4
12. /OE (1)	11. A3
14. A10	13. A2
16. /CE (2)	15. A1
18. D7	17. A0
20. D6	19. D0
22. D5	21. D1
24. D4	23. D2
26. D3	25. GND
28. GND	27. GND
30. R/*W (3)	29. A12
32. /CART(4)	31. A13
34. /NMI	33. A14
36. /IRQ	35. (5)

(1) Inverted copy of 'E' clock from CPU. Valid CPU access when E is high
(2) A15 from CPU according to the nut. ;) However, the commercial carts I've seen use this line for chip enable.
(3) Note: unused on many carts
(4) Active low Cartridge enable - low when E is high and A15 low. Although I verified that it seems to work, many carts do not use it.
(5) Port B, Bit 6 of the 6522 PIA (software controlled line)

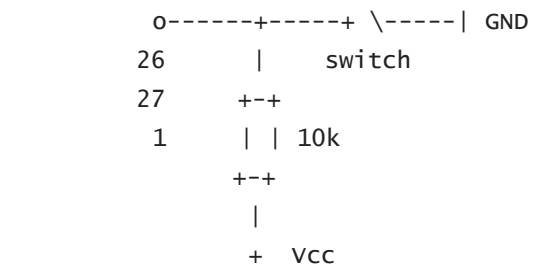
Attention: A11 and /CE ist exchanged at ROM! For Eprom (JEDEC 27XXX), this change must be reversed!

Hier ein 2764 (8k) Eprom und ein 27512 (64k) Eprom im Vergleich:

```
    +-------\/-------+           +-------\/-------+
 1 -|Vpp          Vcc|- 28    1 -|A15          Vcc|- 28
 2 -|A12         /pgm|- 27    2 -|A12          A14|- 27
 3 -|A7            nc|- 26    3 -|A7           A13|- 26
 4 -|A6            A8|- 25    4 -|A6            A8|- 25
 5 -|A5    2764    A9|- 24    5 -|A5   27512    A9|- 24
 6 -|A4          A11|- 23    6 -|A4          A11|- 23
 7 -|A3          /OE|- 22    7 -|A3       /OE-Vpp|- 22
 8 -|A2          A10|- 21    8 -|A2          A10|- 21
 9 -|A1          /CE|- 20    9 -|A1      /CE-/PGM|- 20
10 -|A0           D7|- 19   10 -|A0           D7|- 19
11 -|D0           D6|- 18   11 -|D0           D6|- 18
12 -|D1           D5|- 17   12 -|D1           D5|- 17
13 -|D2           D4|- 16   13 -|D2           D4|- 16
14 -|GND          D3|- 15   14 -|GND          D3|- 15
    +----------------+           +----------------+
```

Und hier der 'Plan' der Umschaltung:

Connect pin 26, 27 and 1, 2 (A12, A13, A14, A15) via pullup resistor of 10k and switch to GND for address range selection
of the 16 x 4k blocks:



Nun wie es aussieht, kann man ein 27512 einfach auch direkt einstecken, wobei dann nur ein 4k Block aktiv ist
(wahrscheinlich der obere).

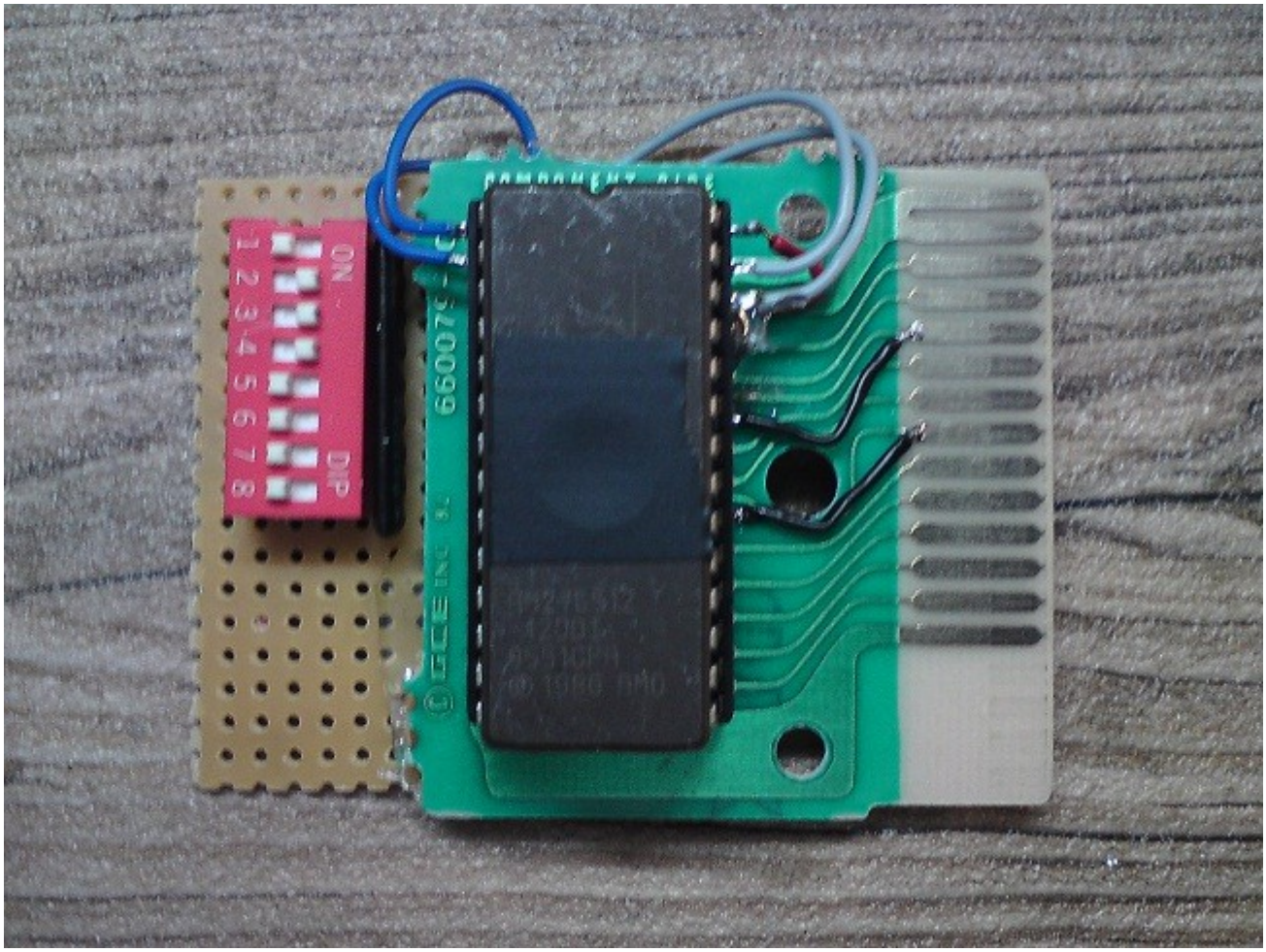
16x 4kb Spiele:

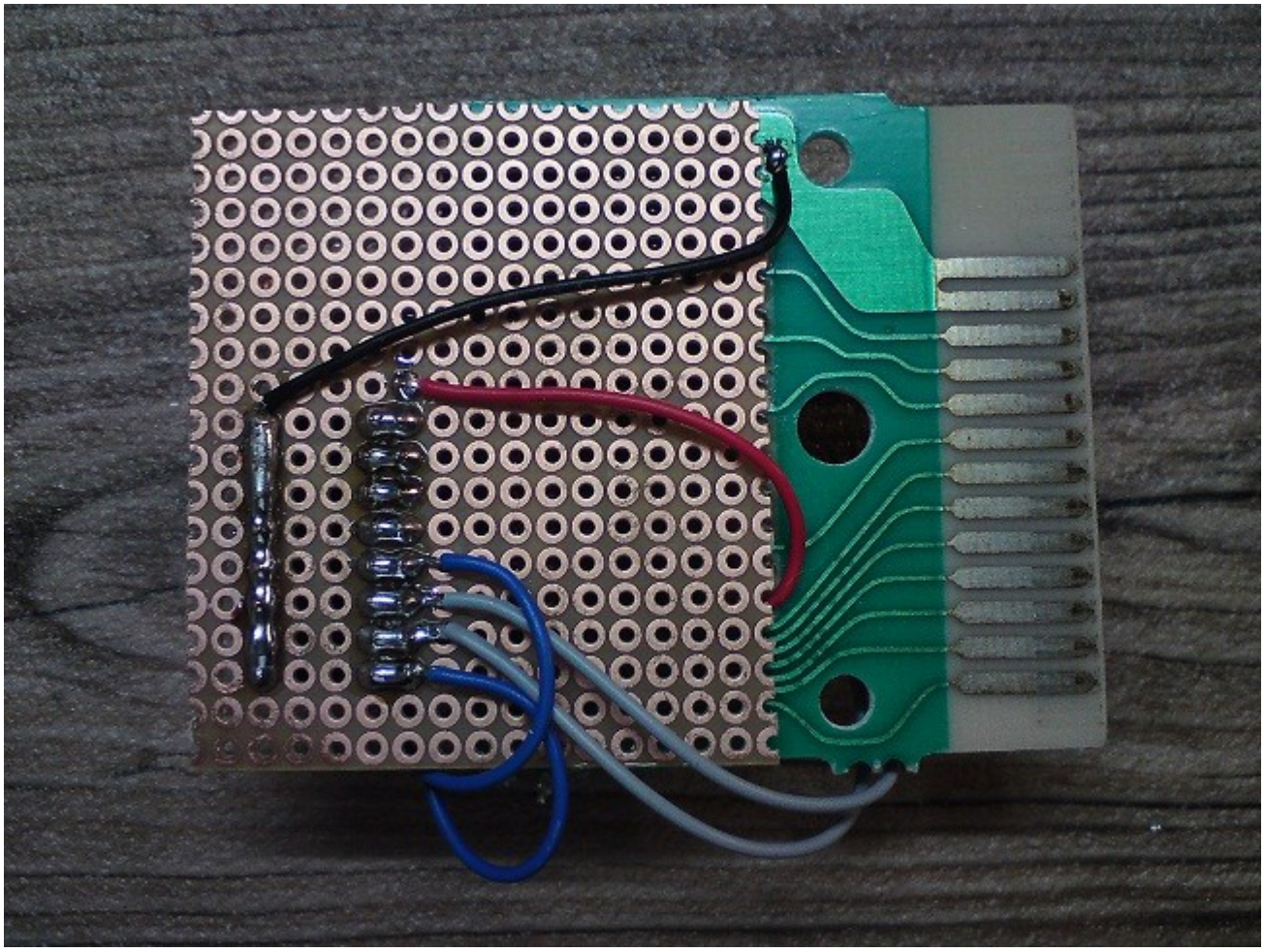
AAAA<- Adressleitungen
1111
5432
0000 - armor
0001 - bedlam
0010 - berzerk
0011 - castle
0100 - chasm
0101 - hyper
0110 - mstorm2
0111 - ripoff
1000 - rocks
1001 - space
1010 - starhawk
1011 - sweep
1100 - vpong
1101 - ski
1110 - philo
1111 - castle

Nun, es war nicht ganz wie erwartet.. A12+A15 scheinen auf 0V zu sein.
16x4k = mstorm2 wird gestartet und läuft!

Also sind:
AAAA
1111
5432
0110 - Pegel verdrahtet.

Ich habe dann einen Adapter für das 16x4k Eprom gebastelt, wobei A12-A15 über 10k Widerstand (pullup) an 5V liegen und über DIP-Schalter an GND gelegt werden können. Und so sieht es aus:

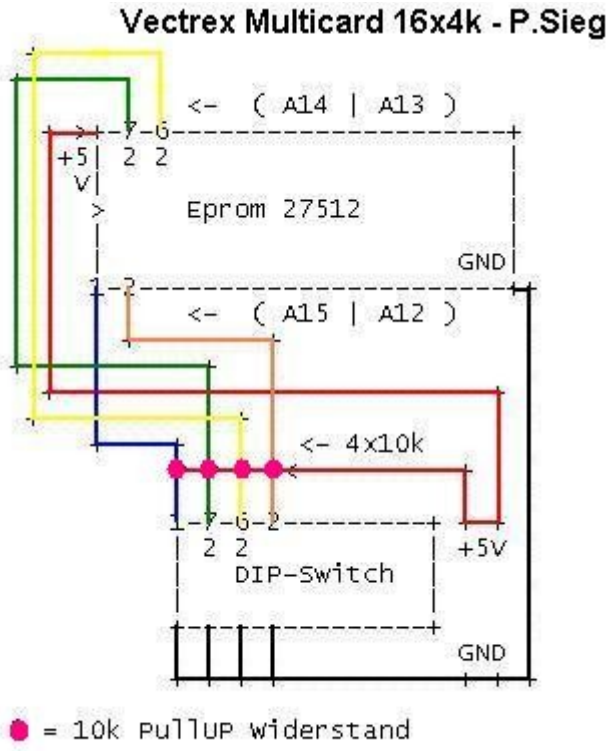




Schwarz=GND, Rot=+5V, die 4 andere Drähte gehen zu A12..A15. Also eingelötet ist ein Präzisionssockel. Dann kommt ein Billigsockel, an dem PIN: 1,2,26,27 zur Seite gebogen wurden (1=A15,2=A12,26=A13,27=A14). Damit kein Kontakt zum unteren Sockel möglich ist, muß man die entsprechenden Kontakte am unteren Sockel z.B mit Isolierband abkleben!
Achtung! Unbedingt +5V am alten PIN 24 des ROM Sockels von A13 des 27C512 isolieren!

Nicht schön, aber es funktioniert!! Ich habe mal versucht ein Layout der Bauteilplazierung und Verschaltung zu erstellen (Ich hatte nur ein 8-fach DIP-Switch.. 4-fach hätte gereicht). Dazu habe ich zuerst als ASCII Zeichnung probiert. War mir aber zu unübersichtlich. Hardcopy gezogen und in Paint die Verbindungen farbig nachgezogen.

Hier das Ergebnis:



2650 Selbstbaucomputer

Links:

http://amigan.yatho.com/

Nach dem Buch:
Mein Homecomputer selbstgebaut
von Joseph Glagla / Dieter Feiler
Otto Maier Verlag Ravensburg
ISBN 3-473-44005-1

System stellt am Netzteil -5V und +5V zur Verfügung. System selbst kommt aber mit +5V aus.
Einschaltmeldung: H A L L O.

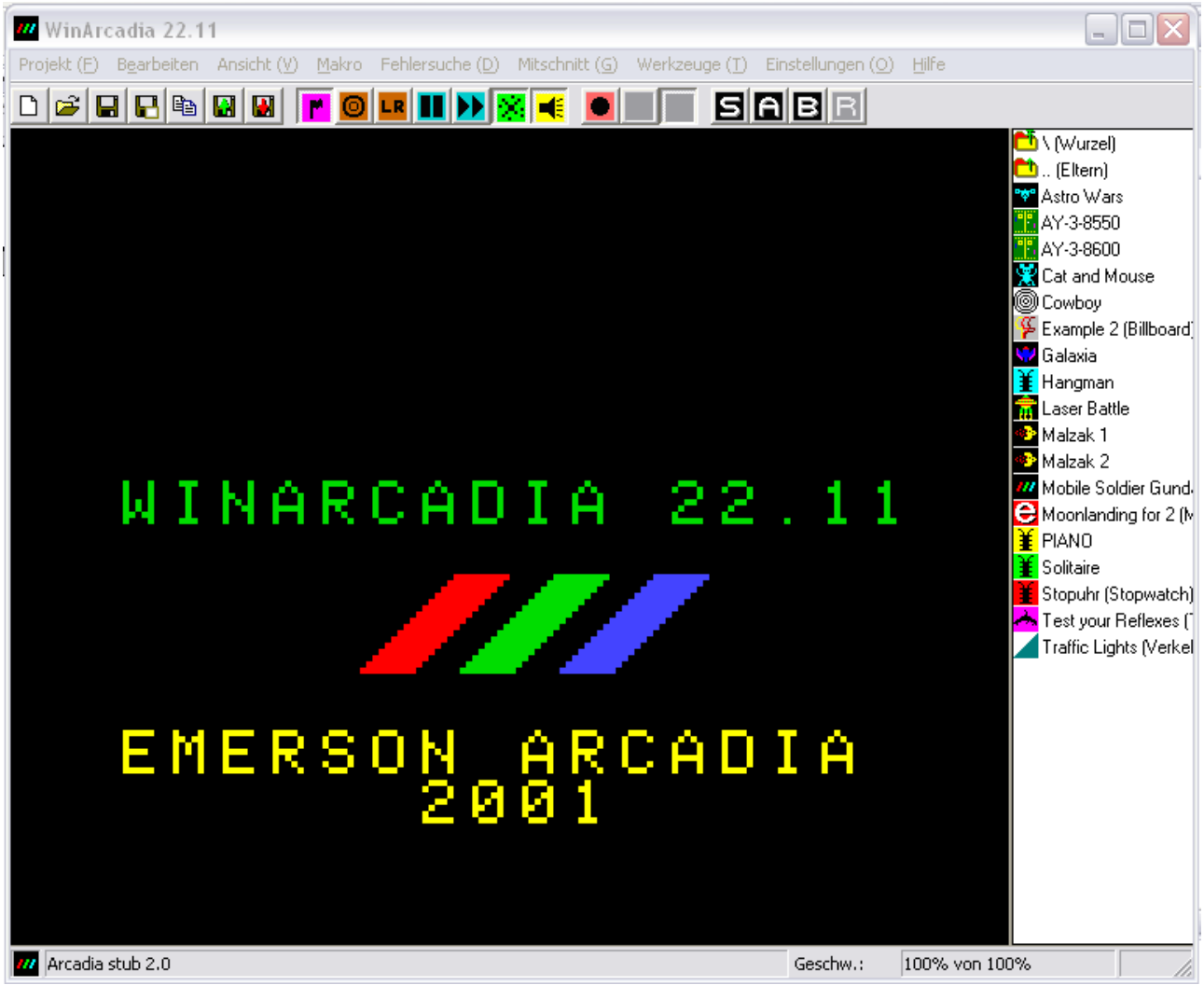
Kleiner Test mit Portplatine und Outport = 09:
PC
0 9 0 0 - Ramadresse
NXT
0 4 - LODI,R0
NXT
5 5 - Bitmuster
NXT
d 4 - WRITE,R0
NXT
0 9 - auf Port 09
NXT
4 0 - HALT
NXT
RST - Reset
PC - Program Counter
0 9 0 0 - auf 0900
GO - Start ab Adresse PC 0900

Gesamtsystem auf ca. 34x34cm Platte:

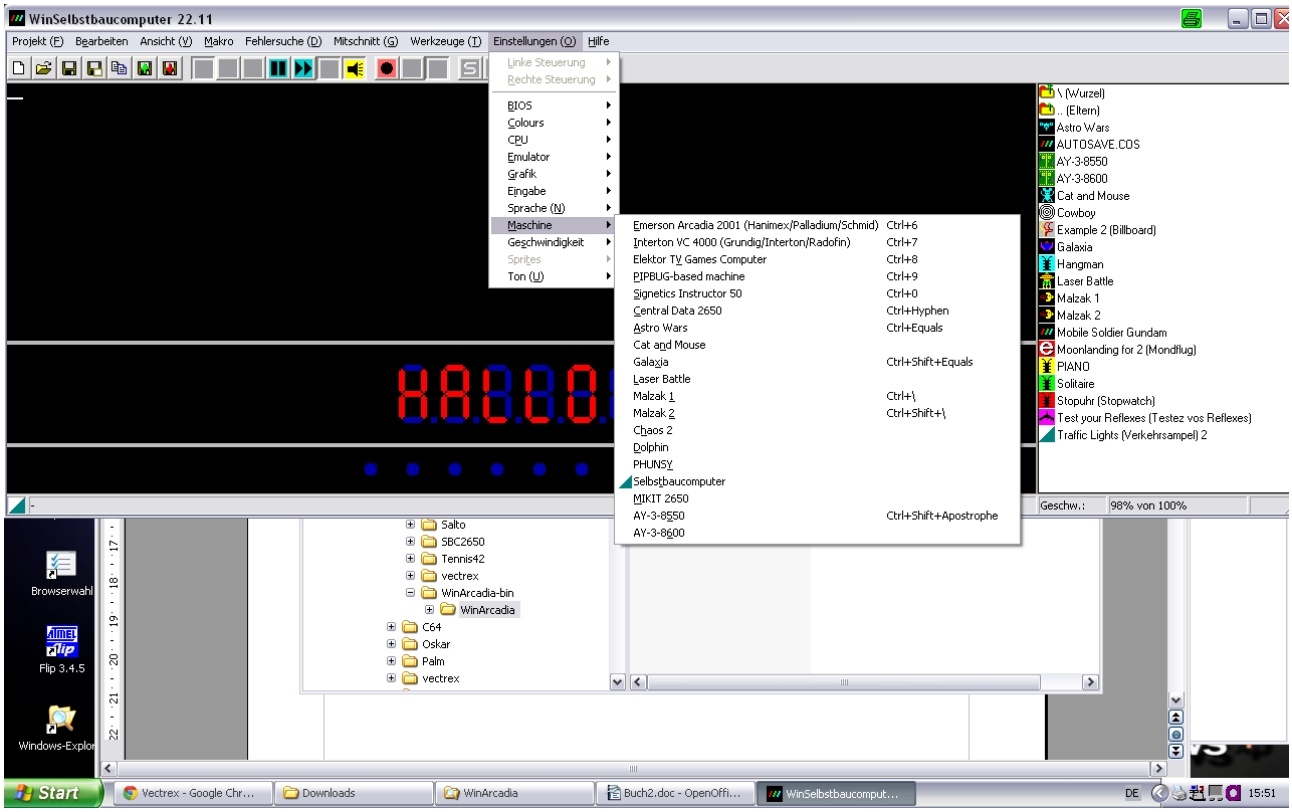
Dank an James Jacobs ist die Emulation nun auch in WinArcadia enthalten.

Installation WinArcadia

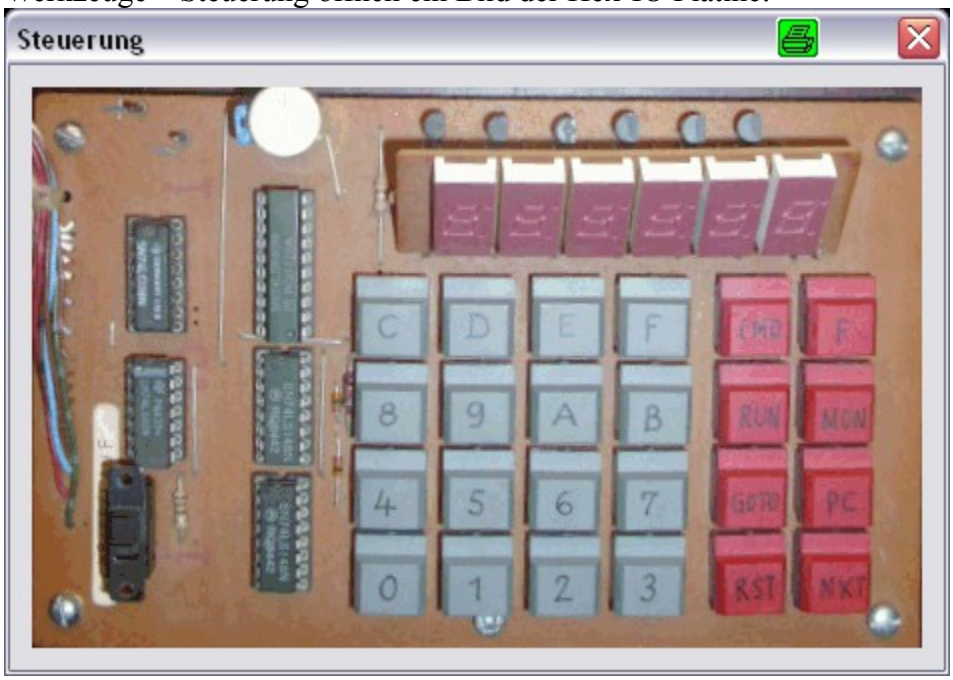
WinArcadia-bin.rar downloaded und entpacken ergibt Verzeichnis WinArcadia-bin und darin WinArcadia. Dort ist der Emulator WinArcadia.exe. Starten:



Einstellungen – Maschine – Selbstbaucomputer:
Einstellungen – BIOS – 1.0



Werkzeuge – Steuerung öffnen ein Bild der Hex-IO Platine:



Hier nun einfach die Tasten anklicken. Genial einfach – einfach genial.

Sharp MZ80B

Links:
http://www.basoft.co.uk/wordpress/emulation/computer-system-emulation/sharp-mz-80b/

Der Sharp MZ80B ist explizit als “Personal Computer” ausgezeichnet und sollte sich im Erscheinungsjahr 1981 von den Homecomputern der MZ-Serie abheben. Dies wurde erreicht durch ein kompaktes Design mit professioneller Tastatur und eingebautem Monitor
sowie einem zusätzlichem Floppylaufwerk.

Als CPU ist ein mit 4MHz getakteter Z80 Klon von Sharp mit der Bezeichnung “LH0080A” eingesetzt. In der Grundausstattung beträgt der Speicher 32Kb Ram, der VRam beträgt 2K. Optional war eine 32K Ram Erweiterung, Centronics-Schnittstelle sowie die Grafik1+Grafik2 Karte zur Erweiterung des VRam aus zwei Bildschirmseiten mit je 8K sowie “Hires” mit 320×200 Punkten erhältlich.

Ein weiteres Detail ist die Servicefreundlichkeit des Sharp. Nach dem Lösen von 2 Schrauben auf der Rückseite kann man das Oberteil über ein Scharnier hochklappen. Damit es nicht wieder zurückfällt hat Sharp hier einen Feststellhebel wie bei einer Motorhaube integriert (wie .B beim Commodore PET/CBM Systemen).

Die Datasette wird elektromechanisch gesteuert und arbeitet vollautomatisch mit 1800 Baud (PWM Modulation).

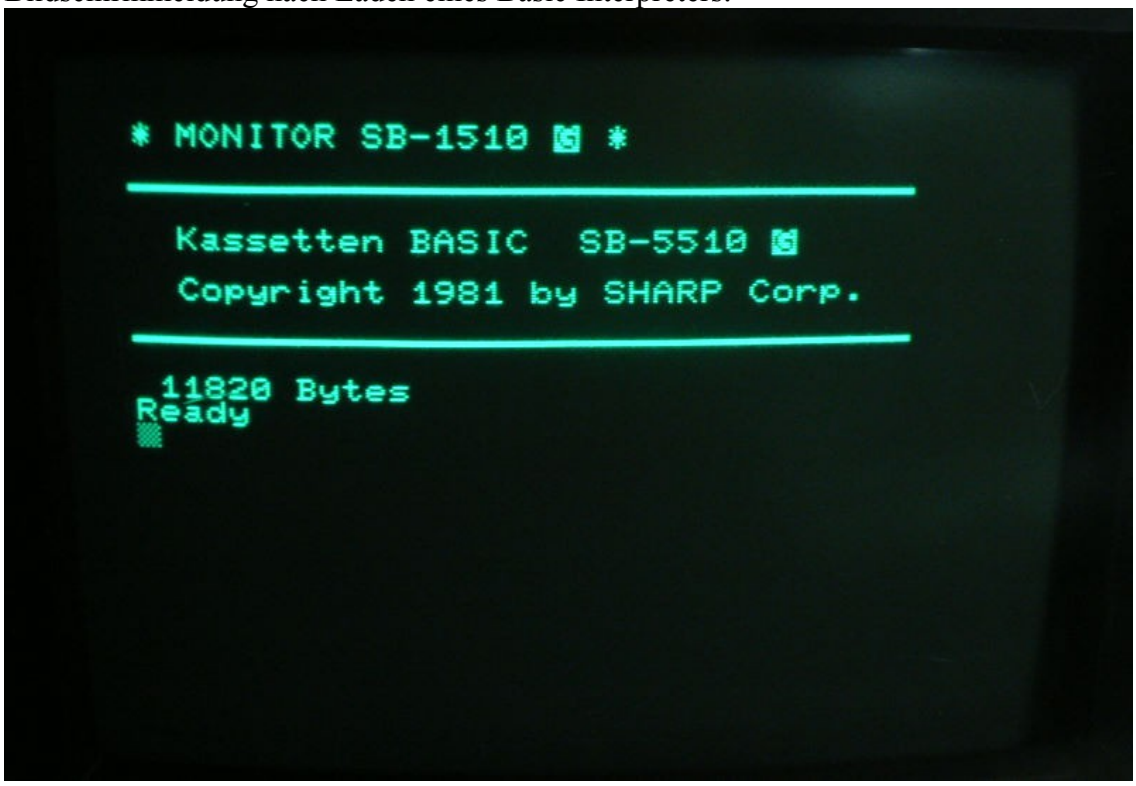
Eine Besonderheit zu vielen anderen Rechnern der Zeit ist, das hier kein Interpreter/Betriebssystem fest eingebaut ist sondern ein Urlader (IPL), welcher verschiedenste Compiler/Interpreter und Programme bis zu CP/M laden kann.
Nach dem Start erscheint die IPL-Meldung, das Kassettenfach öffnet sich und erwartet eine entsprechende Kassette. Nach dem Schließen der Klappe sucht der IPL selbstständig nach dem ersten Programm.

Bild Front (MZ80B Bilder mit freundlicher Genehmigung von Andre Bryx):

Bild Seite:

Bild von innen:

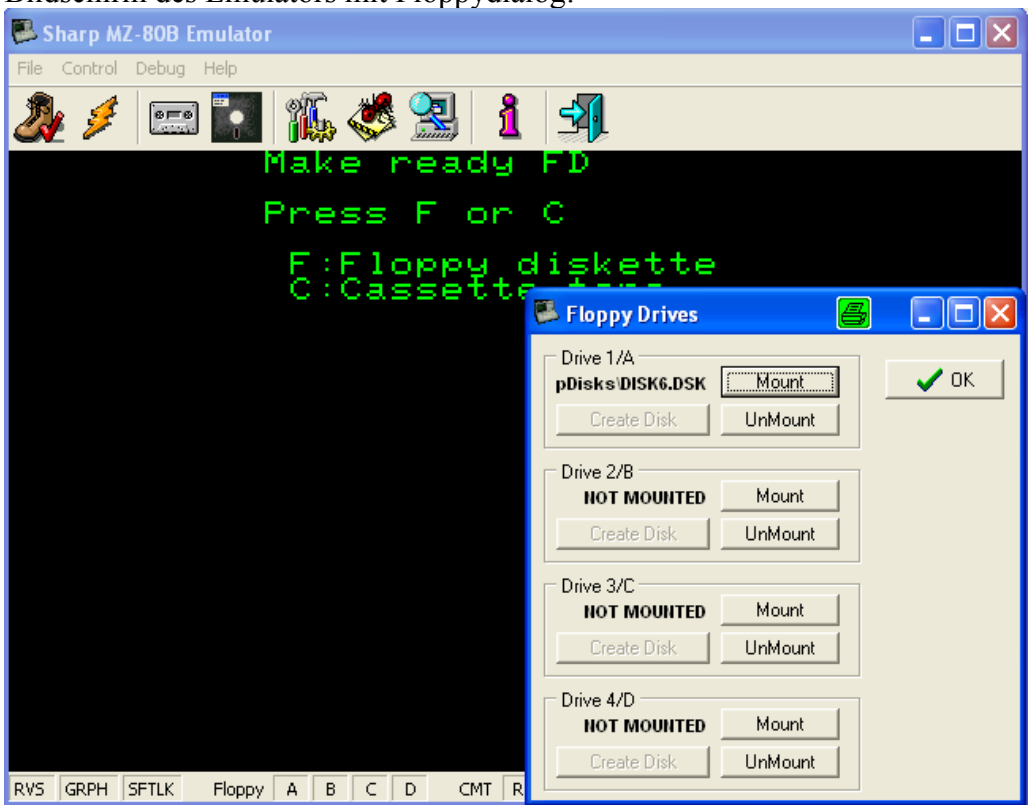
Bildschirmmeldung nach Laden eines Basic Interpreters:



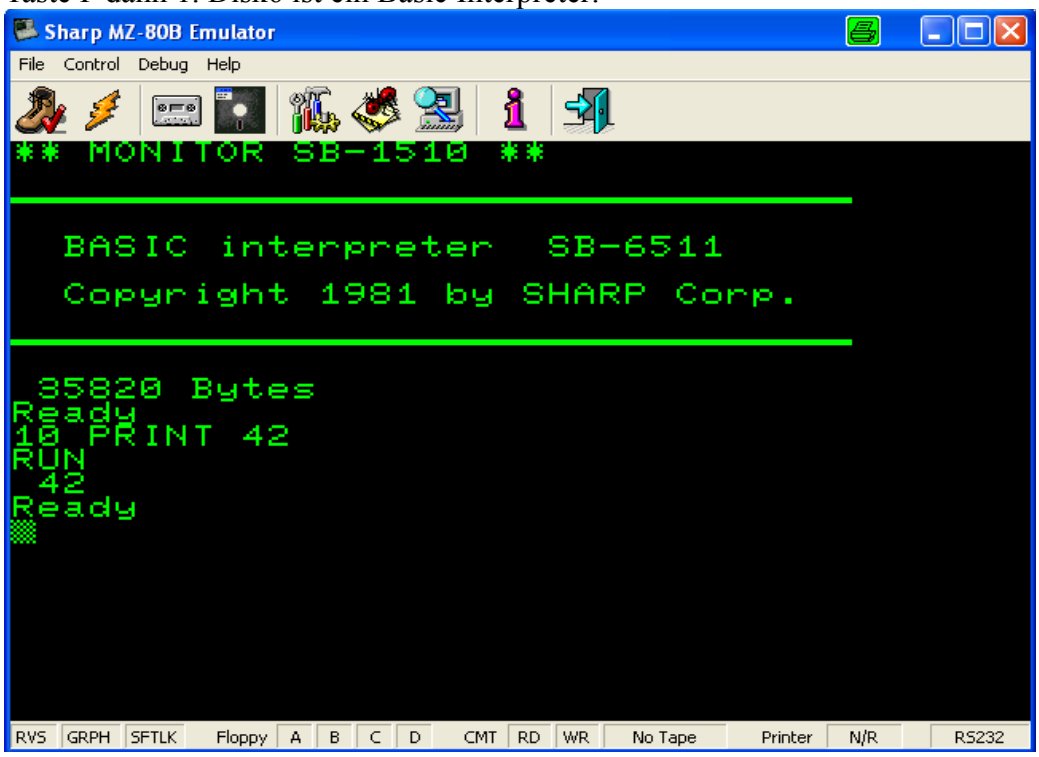
MZ80B Emulator

Tony Friery hat dazu einen MZ80B Emulator geschrieben. Im obigen Link gibt es den Emulator, Sourcecode und einige Floppyimages.

Bildschirm des Emulators mit Floppydialog:



Taste F dann 1. Disk6 ist ein Basic Interpreter:

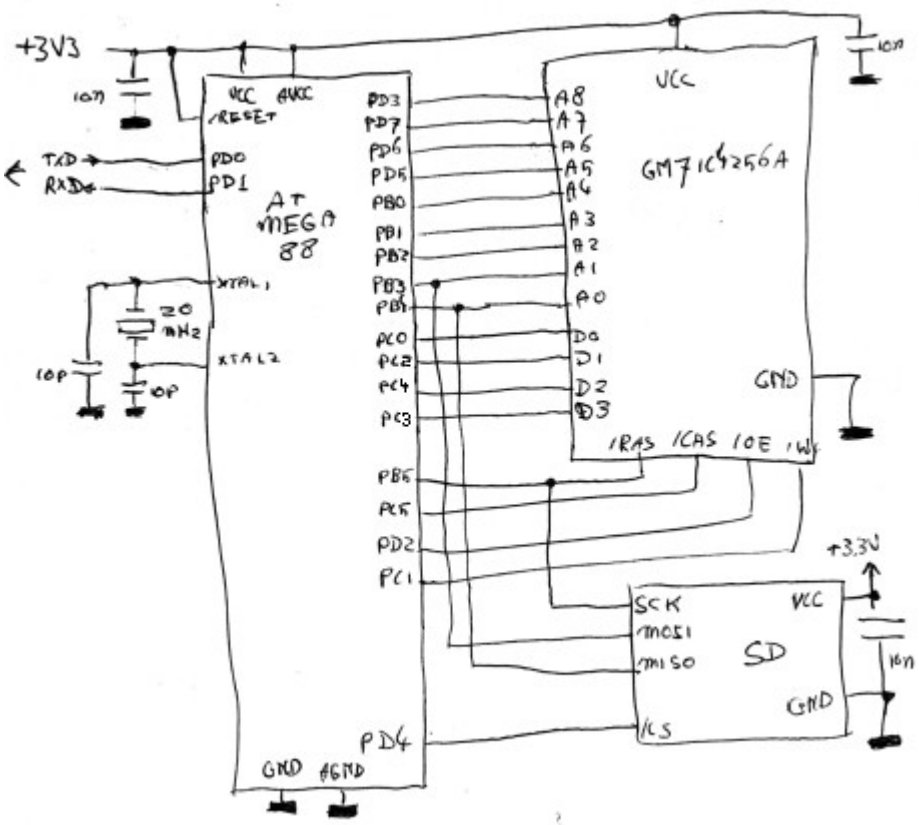


CP/M auf ATmega88 - Thread-ID: 177481

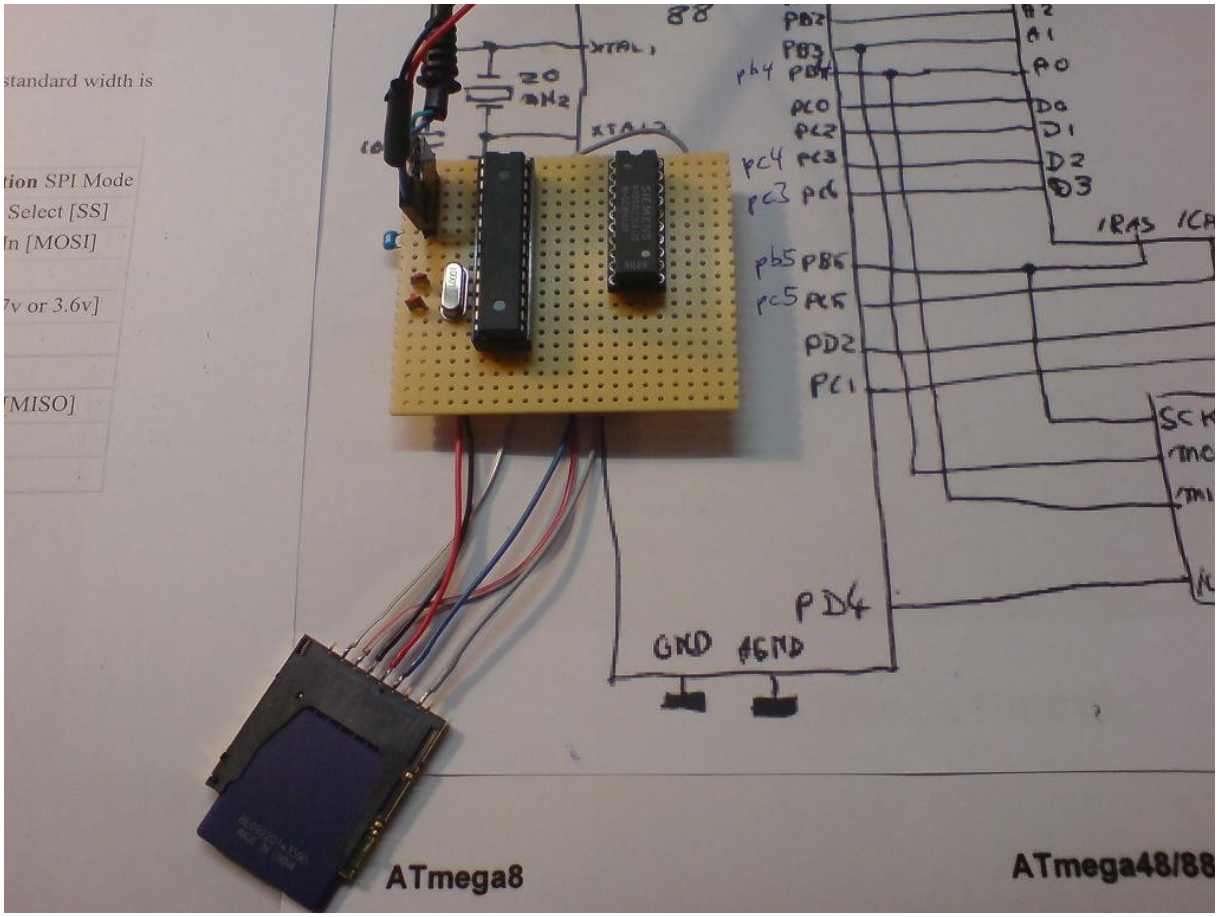
Links:

http://spritesmods.com/?art=avrcpm - zur sprite_tm Seite
http://www.mikrocontroller.net/topic/177481 - zum Forum
http://www.mikrocontroller.net/articles/AVR_CP/M - zum Artikel
http://cloudbase.homelinux.net/viewvc/avr-cpm/avrcpm/trunk/ - zum CVS

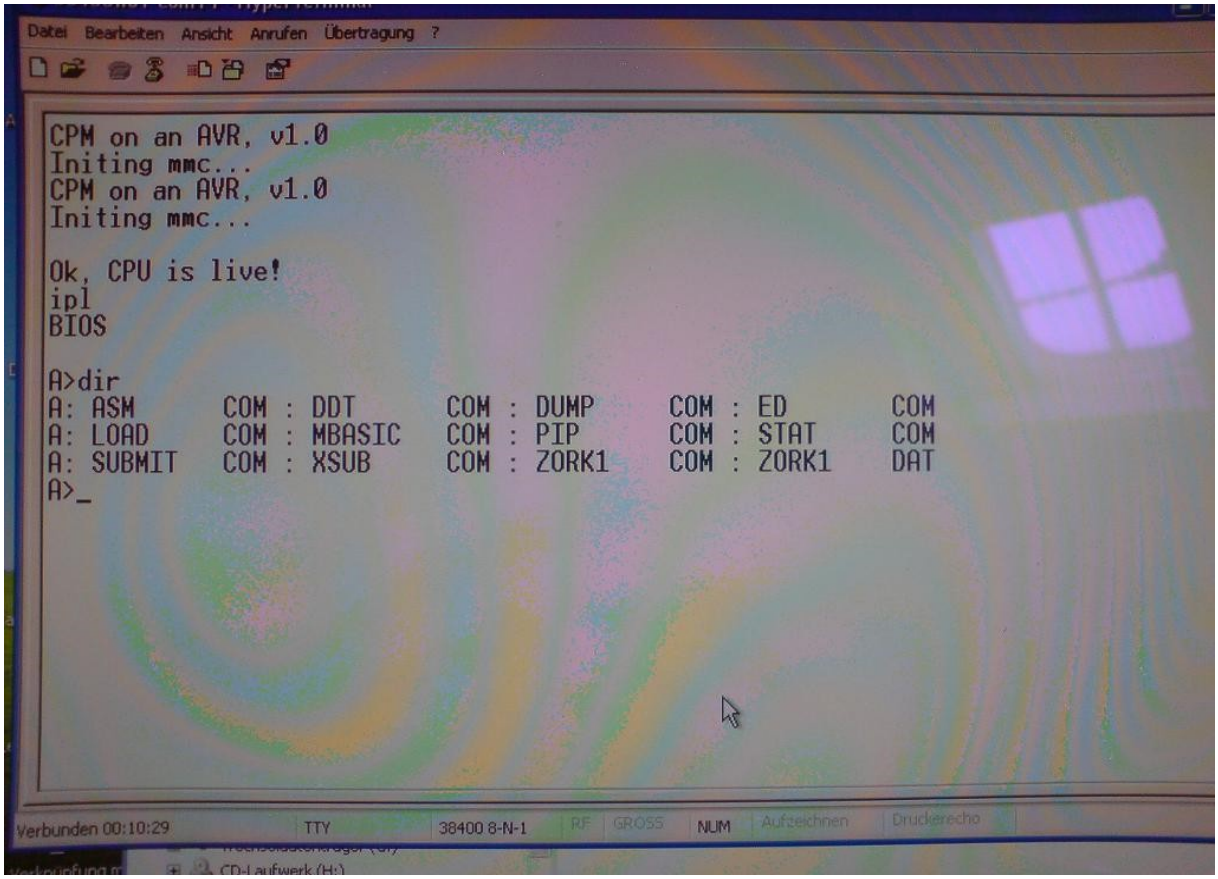
Auf www.hackaday.com sah ich die Meldung über ein CP/M System mit nur 2 IC's von sprite_tm. Hier die Schaltplanskizze, mit der alles anfing:



Nun das weckte mein Interesse! Leider so einige Leitungen (Adress und Daten) zw. den Komponenten zu verbinden in einem Lochrasteraufbau. Nun denn, wer nicht wagt, der nicht gewinnt. 2-3 Tage später war der Lochrasteraufbau fertig und das Ganze funktionierte auch noch:



Es lebt:



In dieser sehr ersten Version lief ZORK1 Adventure, MBasic lief noch nicht. Es wurde nur 8080 CPU emuliert (auch noch mit einigen Fehlern) und das Laufwerk A: Image musste hart auf eine SD Karte mit DD o.ä. ohne jedes Dateisystem/Partitionen geschrieben werden.

Trotzdem! Hallo?

Ein funktionierenden CP/M System mit nur 2 ICs auf Lochraster aufzubauen..

Ich war so begeistert, dass ich das einfach im Mikrocontroller Forum vorstellen musste. Und zwar am 12.05.2009:

Forum: Mikrocontroller und Digitale Elektronik

CP/M auf ATmega88

Forenliste | Threadliste | Neuer Beitrag | Suchen | Anmelden | Benutzerliste | Bildergalerie | Hilfe | Login

Important announcement: there is an **English version** of this forum on EmbDev.net. Posts you create

CP/M auf ATmega88

Autor: Peter Sieg (Gast)
Datum: 12.05.2010 10:28

```
Hi.

Wollte mal auf diese Projekt aufmerksam machen:
http://spritesmods.com/?art=avrcpm

Dort wurde ein ATmega88, ein 256kx4 DRam-chip und eine SD Karte zusammen
geschaltet, um über einen 8080 Emulator letzlich CP/M 2.2 von der SD
Karte zu booten!

Ich dachte erst.. das kann doch nicht sein.. und hielt das für einen
Aprilscherz.. aber er funktioniert wirklich! Siehe:
http://avr.cwsurf (dot) de/?AVR_CP%2FM

Es wäre wunderbar, wenn sich hierfür noch mehr Leute begeistern könnten
und
1. Schauen, welche 8080 CP/M Programme jetzt auch schon laufen.. also
ggf.
   erweiterte diskimage's erstellen und testen.. BTW: MBASIC steigt mit
einem
   fehlenden Opcode aus.. ZORK läuft! Prima wären ein paar Spiele z.B
Sargon
   Schach etc...
2. Versuchen den AVR Assemblersource zu erweitern (mehr opcodes/Z80).

Peter
```

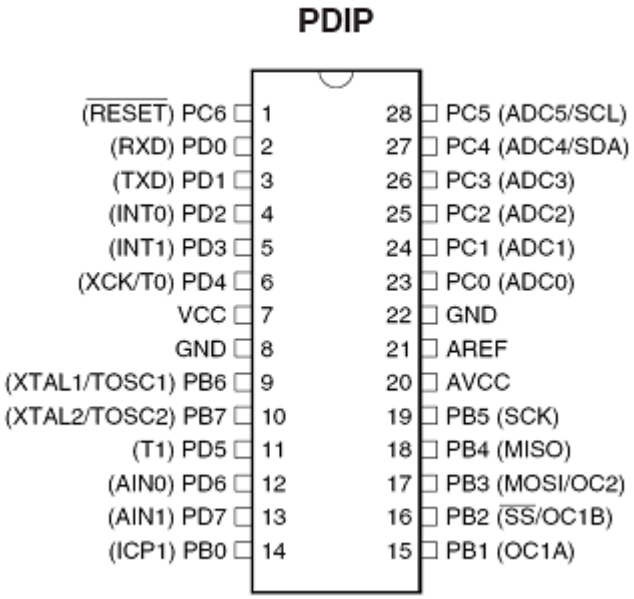
Das Projekt fand reges Interesse und die Weiterentwicklung (auch mit kürzeren und längeren Unterbrechungen) mit wichtigen Meilensteinen will ich hier kurz skizzieren:

30.05.2010: Erste Sammelbestellung von Platinen
27.07.2010: Größere Disk Formate und bis zu 4 Partitionen/Laufwerke und Wordstar läuft.
30.07.2010: Wir gehen auf 8-bit Dram und BDS-C Compiler steht zur Verfügung.
12.08.2010: Erste Z80 Erweiterungen.
17.09.2010: Anfänge der FAT16 Unterstützung von Laufwerk Images.
07.07.2011: AVR CP/M USB Stick Version.
07.03.2012: Z80 Emulation und Turbo Pascal 3 läuft.

Inzwischen haben wir auf ZSDOS umgestellt.
Es gibt Diskimages mit Fortran, Algol, PL/1, muMath und Spielen ;-)

Bauteile Pinouts:

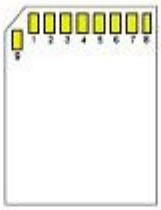
ATmega88:



Dram 256x4:

```
44256, 44258
256kx4 DRAM.

      +----------+
  D0 |1  +--+ 20| GND
  D1 |2       19| D3
 /WE |3       18| D2
/RAS |4       17| /CAS
     |5 44256 16| /OE
  A0 |6 44258 15| A8
  A1 |7       14| A7
  A2 |8       13| A6
  A3 |9       12| A5
 VCC |10      11| A4
     +----------+
```

SD Karte:

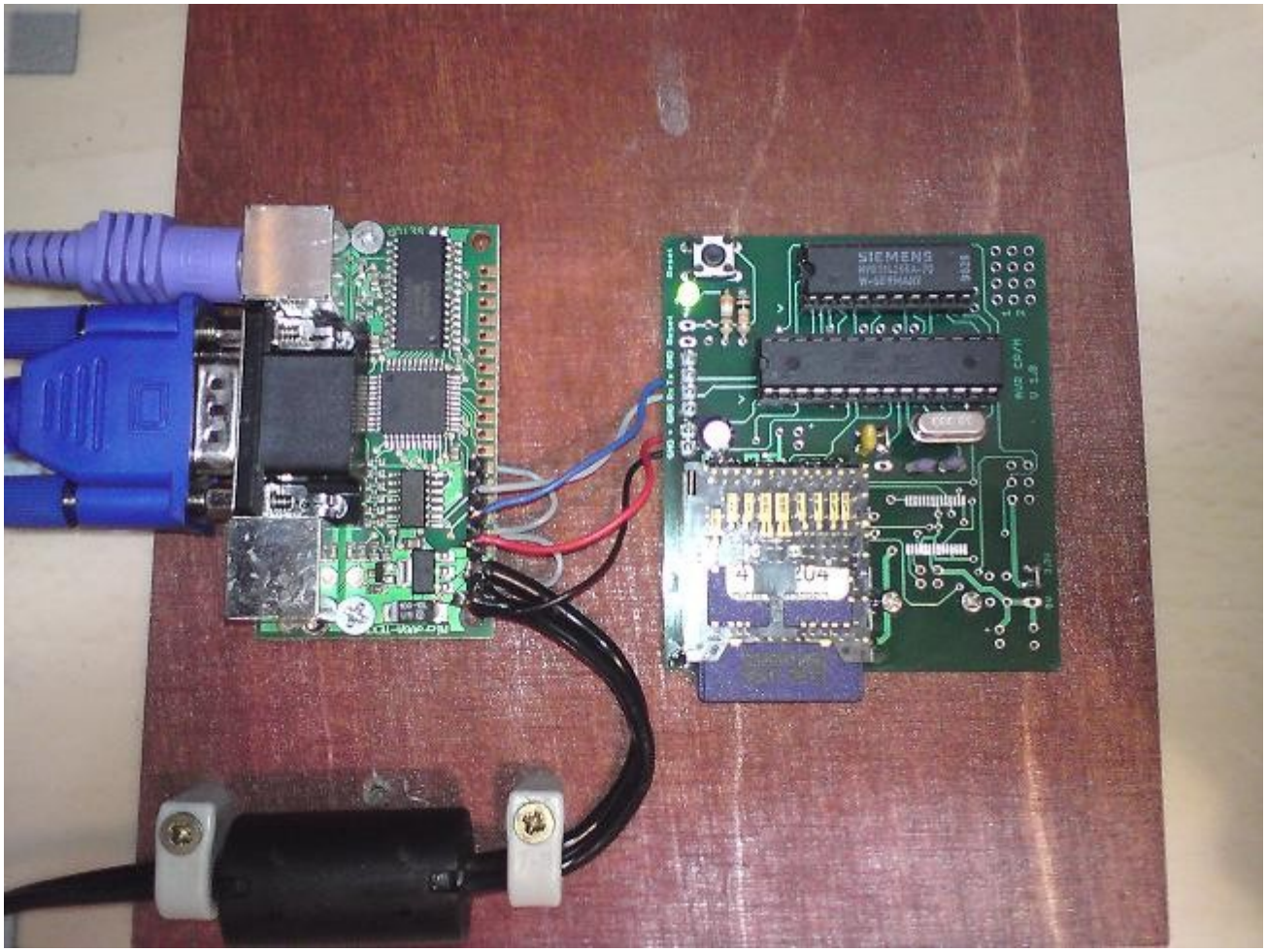


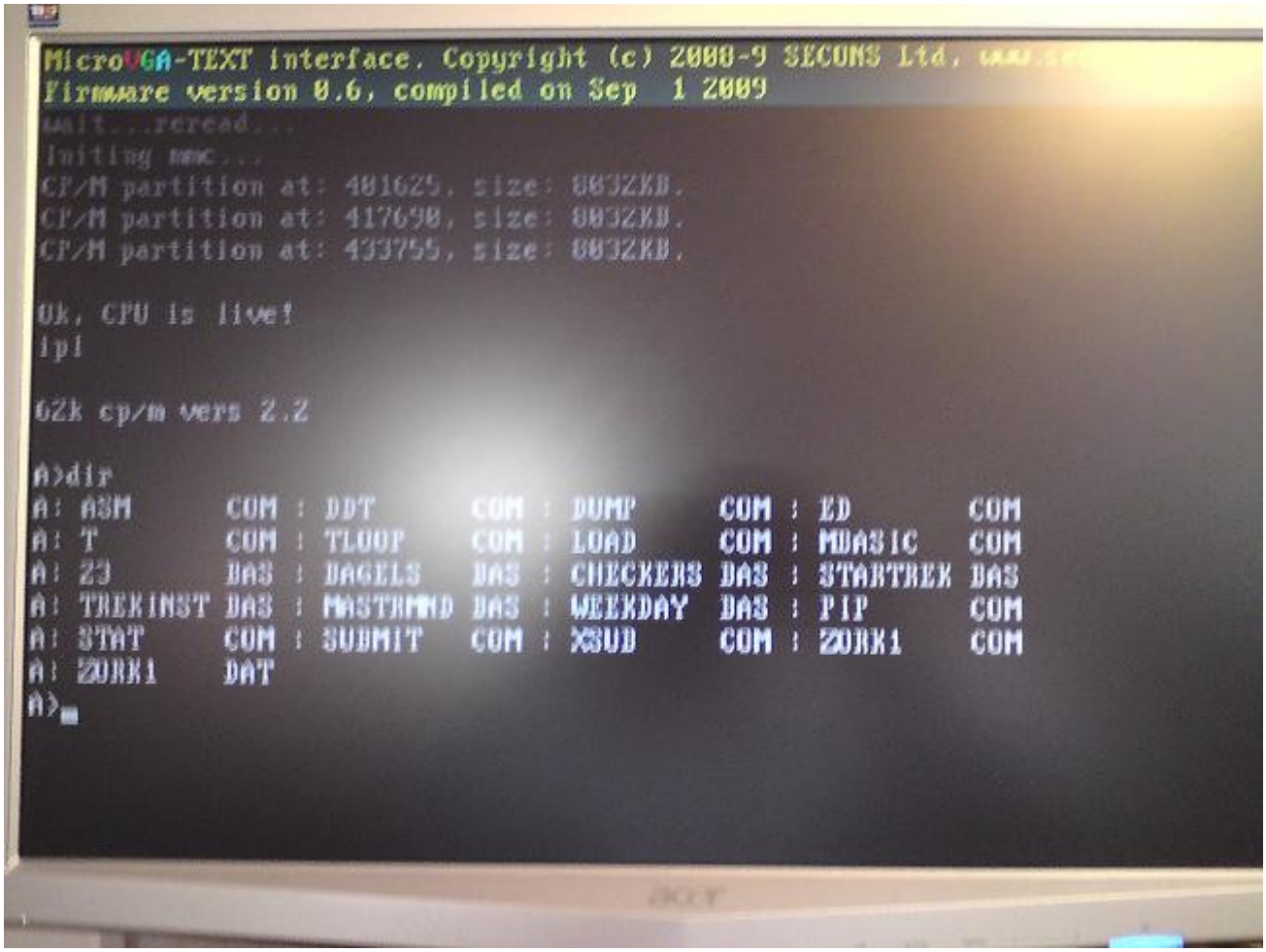
SD Card

The Secure Digital card dimensions are: 24mm wide x 32mm long. The standard width is 2.1mm, while the Thin SD Memory Card has a width 1.4mm.

Secure Digital Pinout			
Pin #	**Pin Name**	**SD Signal Function** SD Mode	**SPI Signal Function** SPI Mode
1	DAT3/CS	Data Line 3	Chip Select/Slave Select [SS]
2	CMD/DI	Command Line	Master Out/Slave In [MOSI]
3	VSS1	Ground	Ground
4	Vdd	Voltage Supply [2.7v or 3.6v]	Voltage Supply [2.7v or 3.6v]
5	Clock	Clock	Clock [SCK]
6	Vss2	Ground	Ground
7	DAT0/D0	Data Line 0	Master In Slave Out [MISO]
8	DAT1/IRQ	Data Line 1	Unused or IRQ
9	DAT2/NC	Data Line 2	Unused

Hier ein paar Bilder der Platine mit MicroVGA als Terminal:



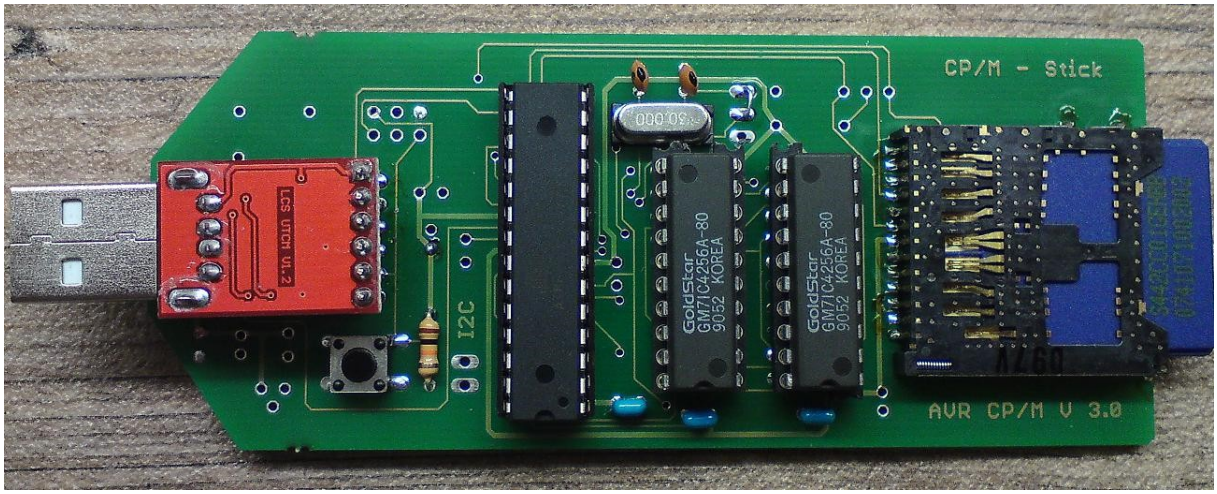


Erreichte Geschwindigkeiten durch Optimierungen und Übertakten:

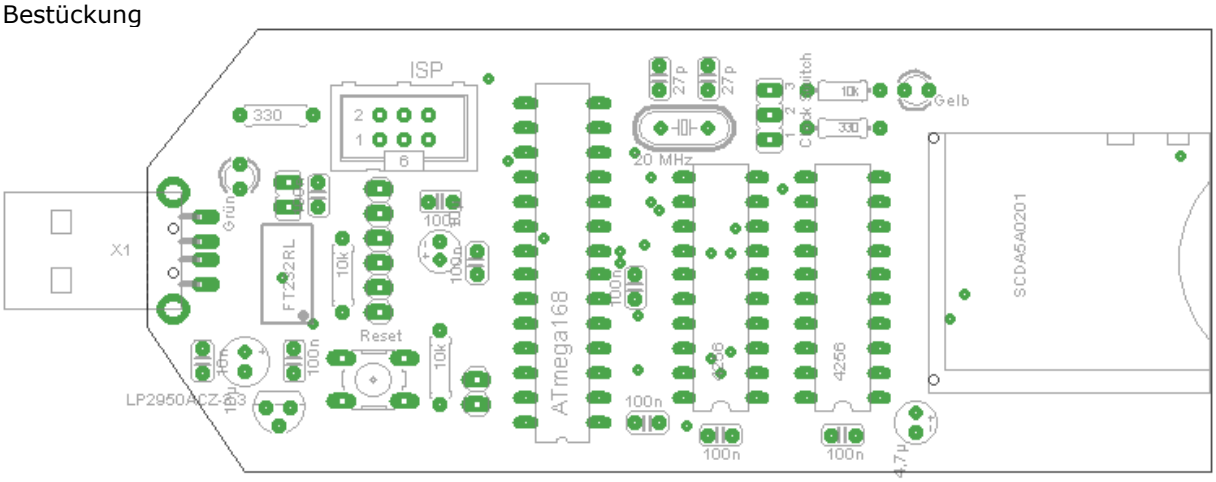
	Peters	Leos		
	Schleife	Schleife		
AVR MHz	Zeit [s]	Zeit [s] "Speed"	HW	SW
88 20	35.999		original hardware	
88 20	21.380		4	(unoptimized code)
168 24	17.620		4	(unoptimized code)
88 25	16.871		4	(unoptimized code)
8 20	12.180		4	0
8 20	12.453	3.488 827 KHz	4	1
8 20	11.500	3.488 827 KHz	4	3
88 30	7.573		4	?
8 20	7.599	2.039 1380 KHz	8	2
8 20	6.680	2.039 1380 KHz	8	3
8 20	6.418	1.958 1470 KHz	8	4
*** Hochrechnung ***				
88 40	5.6	~1600 KHz	4	?
X8 30	4.3	~1800 KHz	8	
X8 40	3.25	~2000 KHz	8	

"HW 4" ist die jetzt aktuelle 4-bit Verdrahtung und "HW 8" die bald neu kommende 8-Bit Dram-Access Variante.

So sieht die USB Stick Platinen Variante aus:



Bestückung



Wer nun auch Lust bekommt hier mitzumachen sei herzlichst willkommen!
(und der folge den genannten Links.)

AX81 - ZX81 auf AVR Basis

Links:

http://www.jcwolfram.de/projekte/avr/ax81/main.php
http://www.mikrocontroller.net/topic/236397
http://www.jupiter-ace.co.uk/
http://home.micros.users.btopenworld.com/zx80/zx80.html

ZX81 Homecomputer von Sinclair aus ca. 1982/3

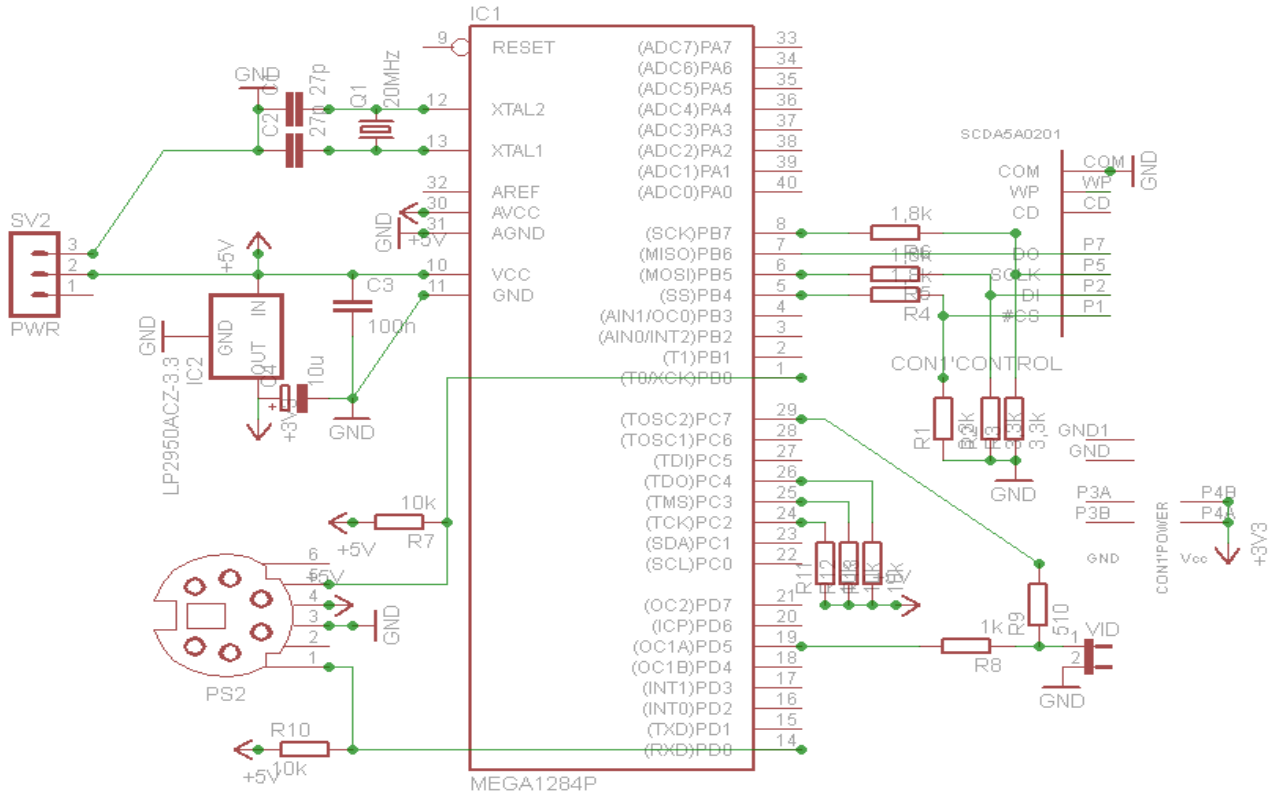
Der ZX81 war zu seiner Zeit einer der günstigsten Homecomputer! CPU: Z80, dazu 1-2k RAM, ein 8k ROM mit Basic, und 1 Spezial-IC die ULA. S/W TV Darstellung und Folientastatur mit Mehrfachbelegung machten den 'Charm' des Systems aus ;-) Gespeichert und geladen wurden Programme von Kasettengeräten.

Bild:

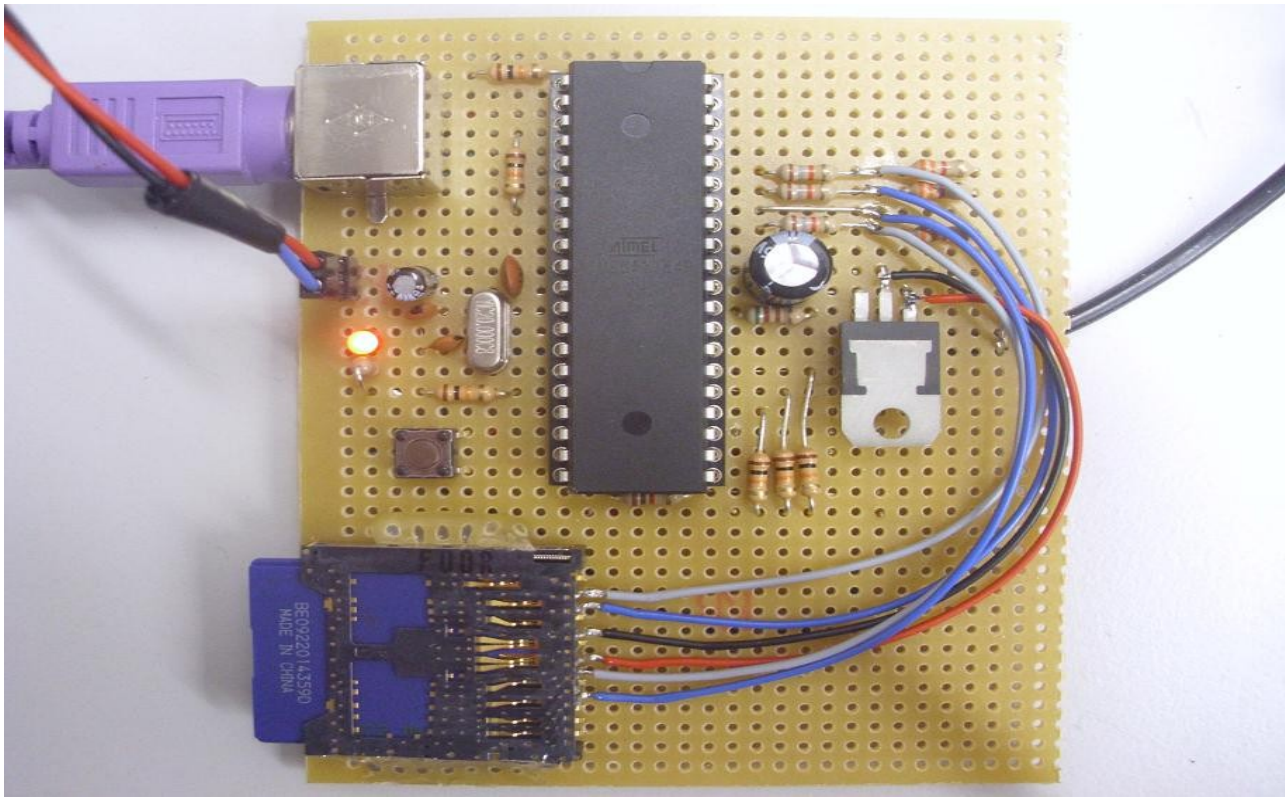
AX81

Jörg Wolfram, der auch schon die AVR Basic Computer entworfen und gebaut hat, hat in 2011 die Z80 Emulation auf AVR implementiert und einen ZX81 Emulator drauf basierend implementiert und gebaut. Ausgabe geht über BAS/Video, VGA oder auch kontrolerlose LCD Displays. Eingabe über eine PS/2 Tastatur. Als Ersatz eines Kassettenlaufwerks, dient eine SD Karte als Massenspeicher. Dabei werden in einem ca. 135MB großen Dateiimage 8192 16k Programmspeicherplätze reserviert. Jeweils 32 davon repräsentieren ein virtuelles Tape. Virtuelle Tapes lassen sich dabei über POKE 99,x (x=0..255) wechseln. Außer einem AVR mit Beschaltung, braucht man nur ein paar Widerstände und den SD Kartenslot mit Pegelanpassung.

Schaltplan

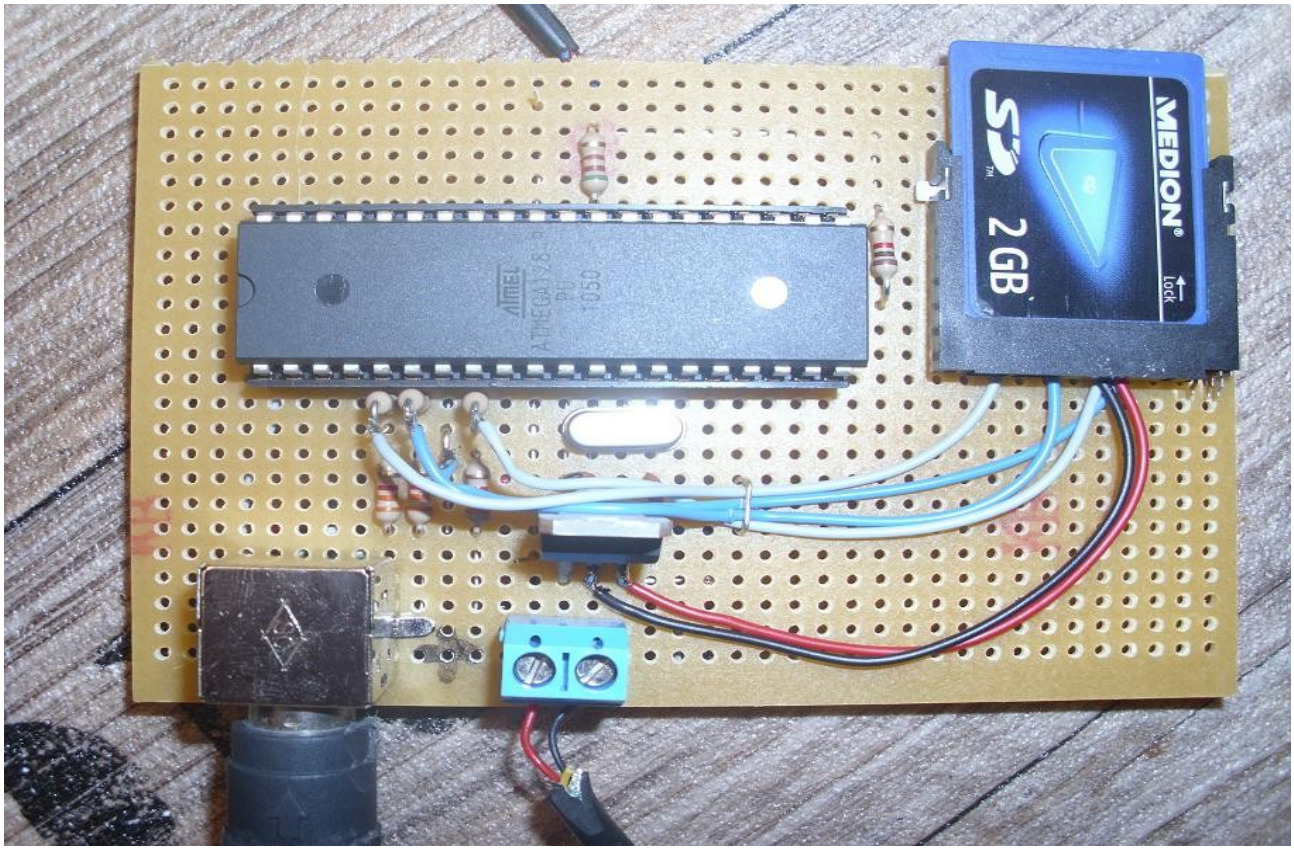


Lochrasteraufbau

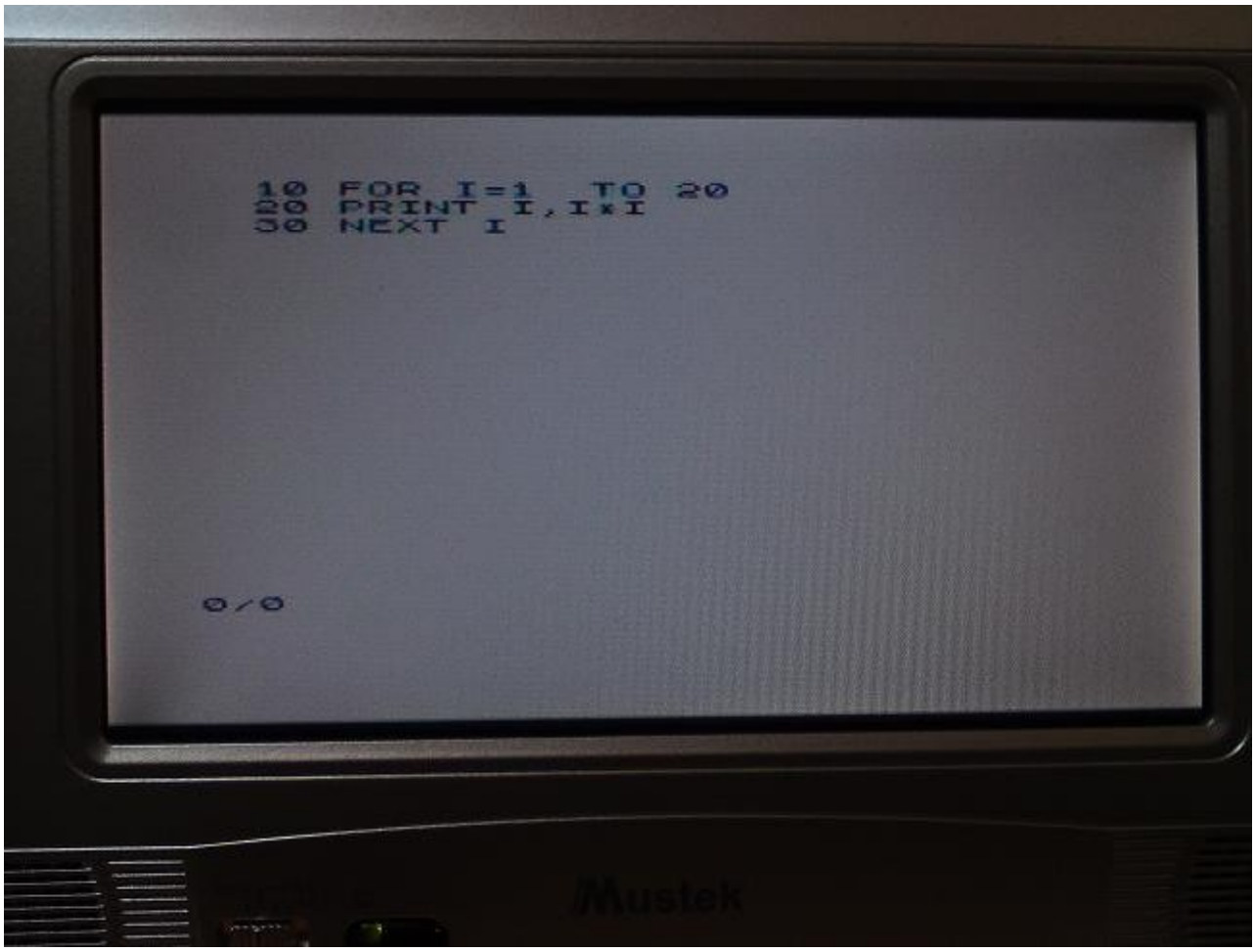


Lochrasteraufbau Nr.2

Die beiden Pullup Widerstände am PS/2 Anschluss sind nicht bestückt. Auch die Widerstände zur Wahl des Videomodus sind weggelassen worden. Wichtig ist aber ein Elko/Tantal an der 3.3V Versorgung der SD Karte (hier auf der Unterseite bestückt.



Basicprogramm:



Tennis for two

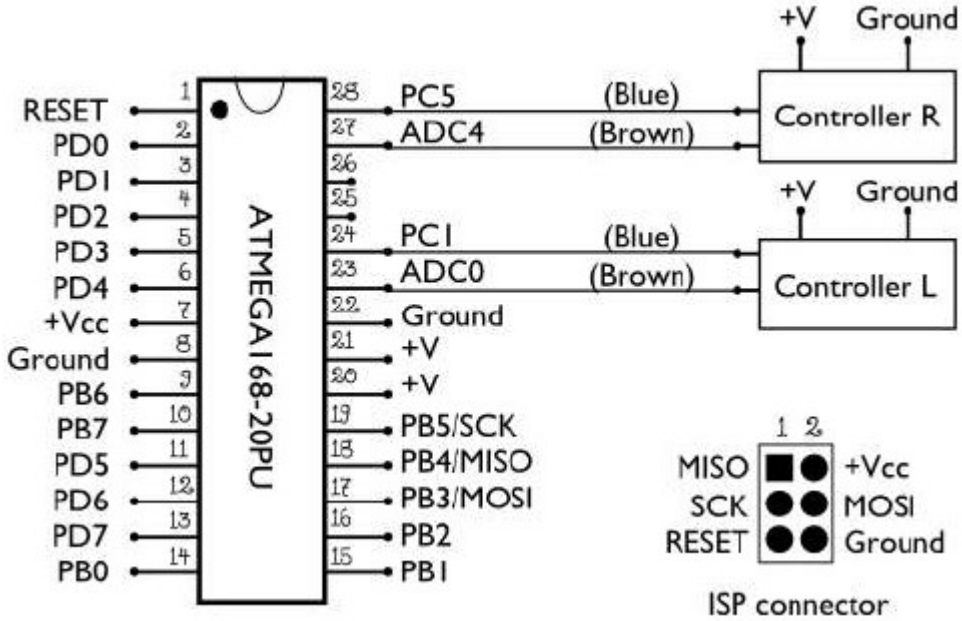
Link:

http://www.evilmadscientist.com/2008/resurrecting-tennis-for-two-a-video-game-from-1958/

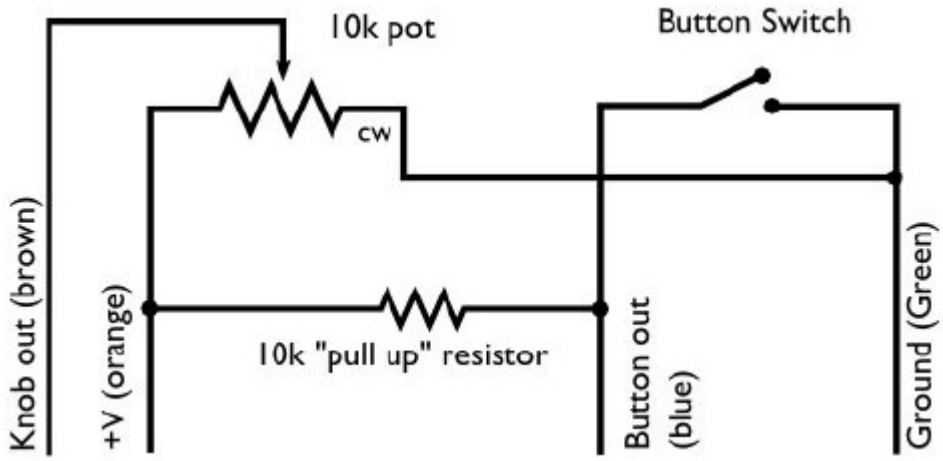
Tennis for Two ist ein Spiel, das 1958 vom amerikanischen Physiker William Higinbotham am Brookhaven National Laboratory entwickelt und konstruiert wurde. Es kann als das erste Computerspiel angesehen werden. Die Hardware bestand aus einem Analogcomputer und einem fünf Zoll (12,5 cm) kleinen Oszilloskop. Die Gesamtanlage bestand aus mehreren Teilen und war etwa fünf Meter breit. Das Spiel war der Vorgänger des populären Pong. Die Ansicht bei Tennis for Two zeigt eine seitliche Darstellung des Tennisplatzes, der Ball wird von der Gravitation beeinflusst, und er muss über ein Netz gespielt werden.
Vorgeführt wurde das Spiel am Tag der Offenen Tür des Kernforschungszentrums in dessen Sporthalle.
Als Eingabe dienten zwei kleine Kästen, mit einem Knopf zum Schlagen des Balles und einem Knauf zum Einstellen des Abprall-Winkels.

Anlässlich der Classic Computing Austellung des Vereins zum Erhalt klassischer Computer in der Räumen der Humbolt Universität zu Berlin, konnten wir dank der Unterstützung der Universität ein Tennis42 mit einem echten Analogcomputer zeigen.

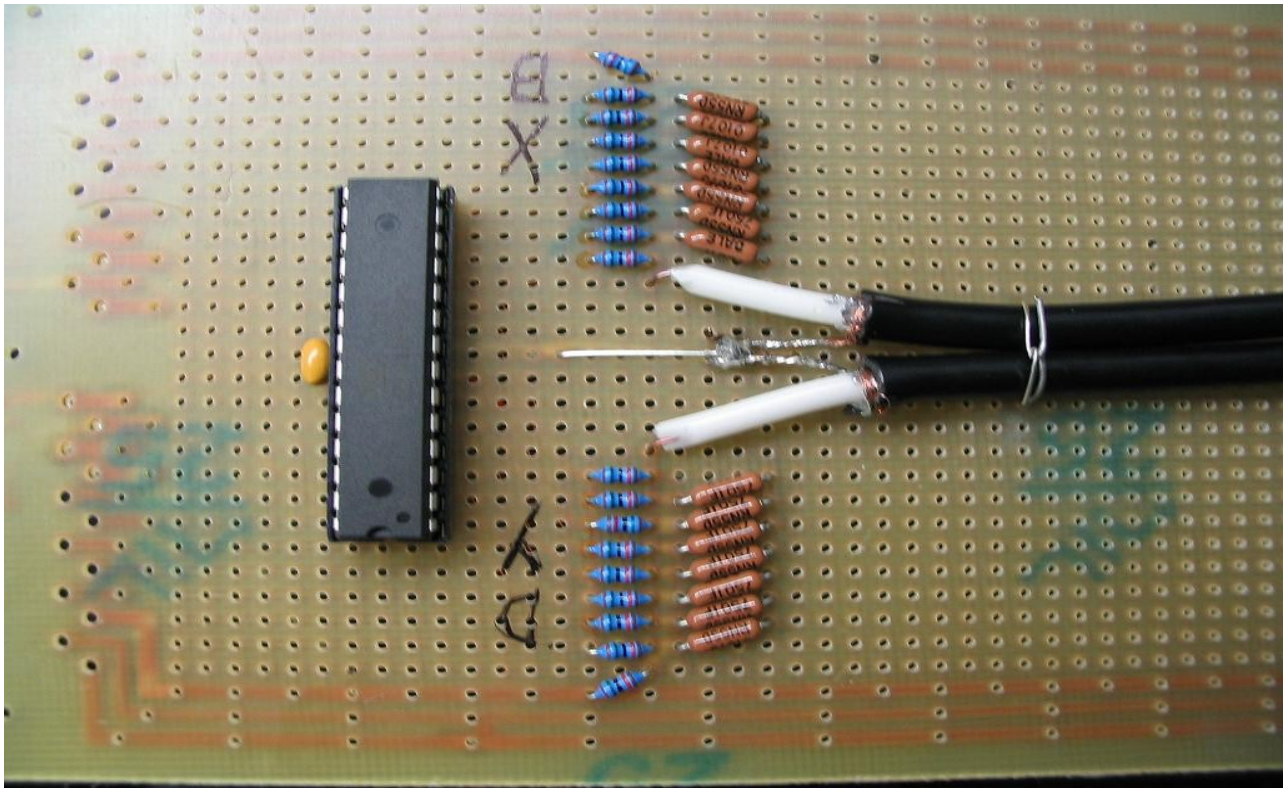
Schaltplan AVR



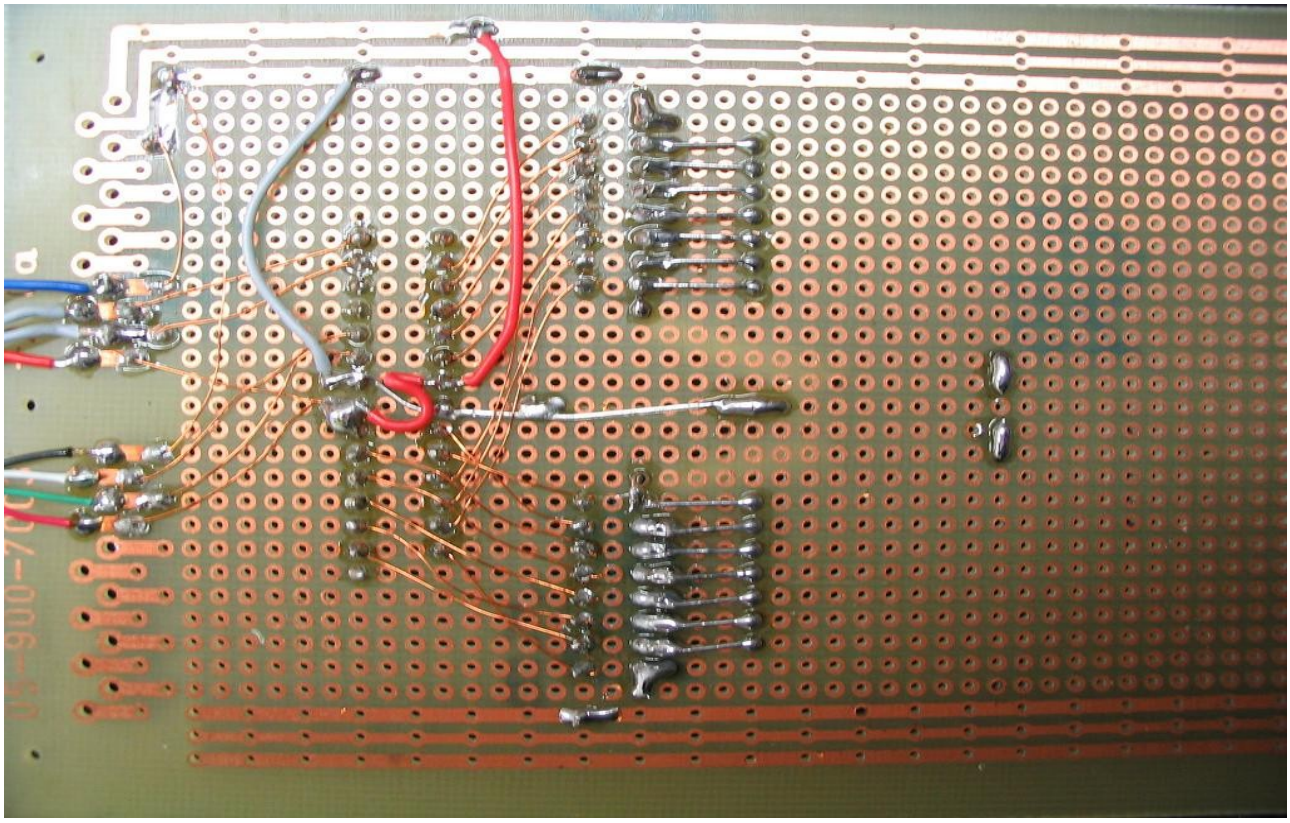
Schaltplan Kontroller



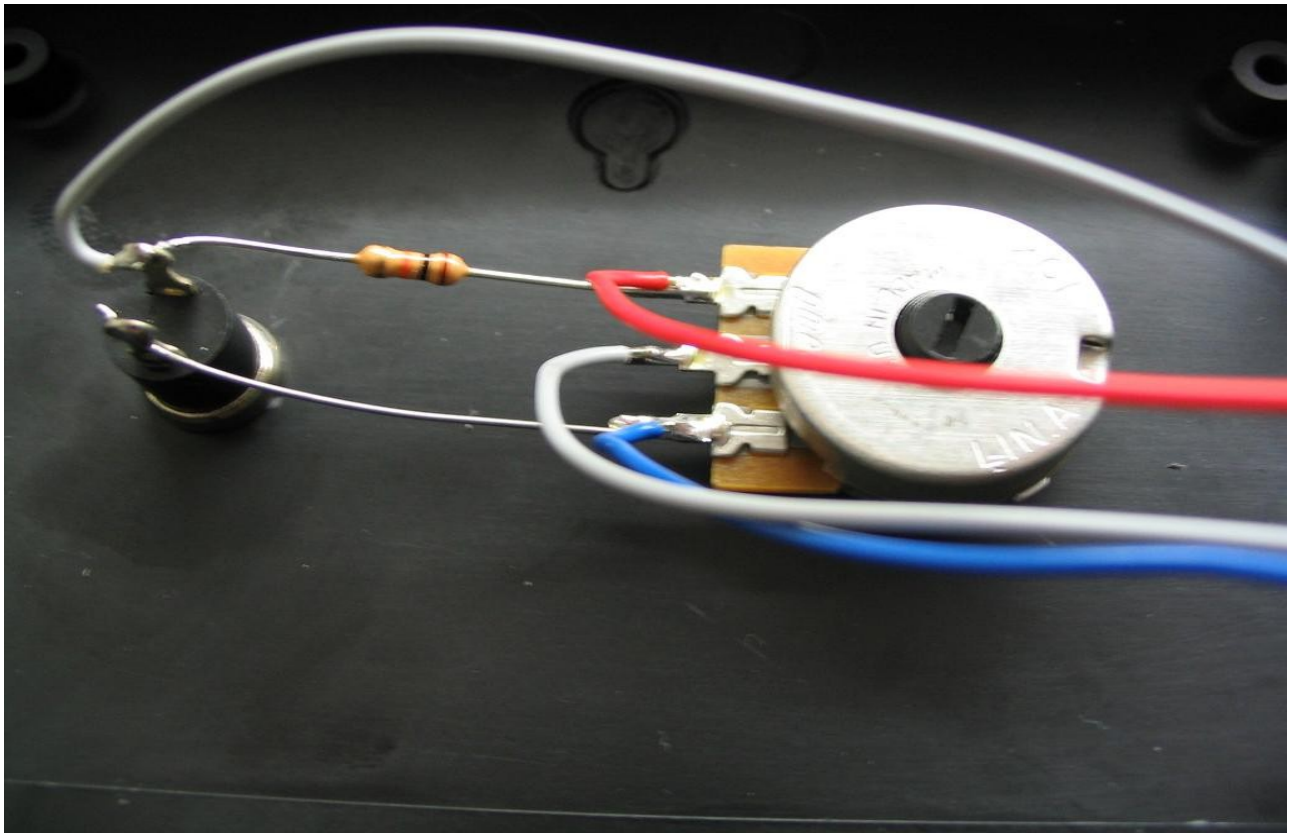
Lochrasterplatine oben



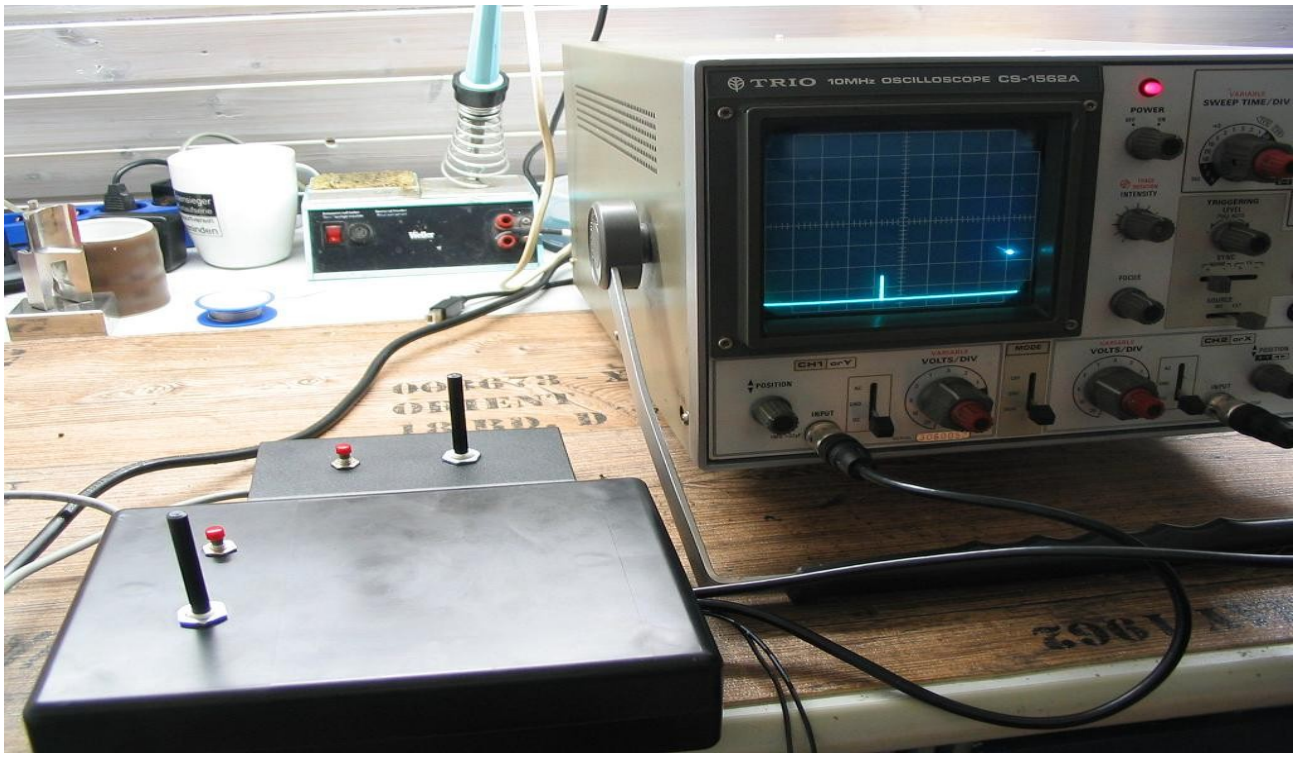
Lochrasterplatine unten



Kontroller Detail



Komplette Tennis42 Anlage mit TRIO CS1562A Oszilloskop



Linux auf 8-bit AVR

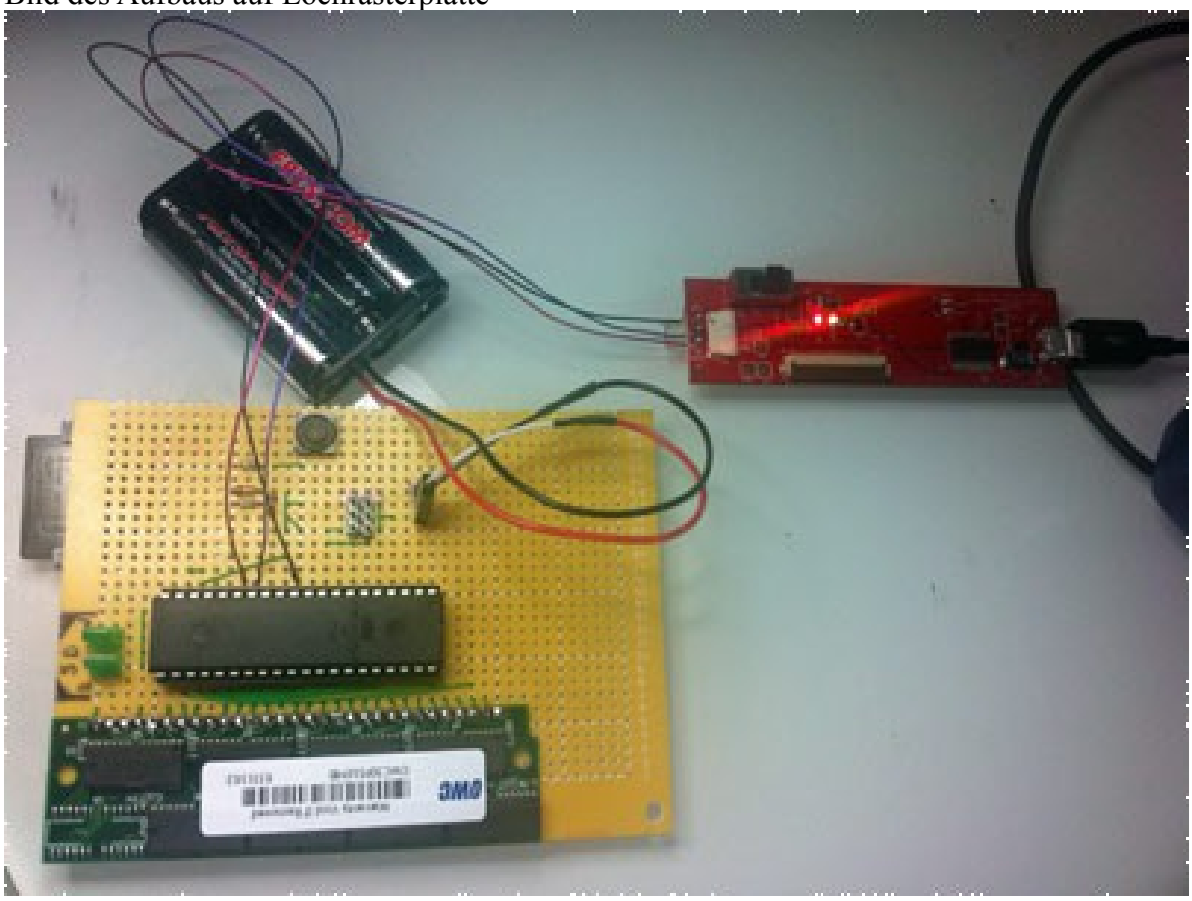
Links:
http://dmitry.gr/index.php?r=05.Projects&proj=07.%20Linux%20on%208bit
http://www.wengenroth.co/projects/adding-i2c-to-the-avr-arm-emulator/

Wenn man so in einschlägigen Foren fragt, was die Mindestanforderungen für ein lauffähiges Linux System sind, hört man eher 32-bit CPU, Memory Management Unit (MMU) und 1MB Ram.

Nun, dachte sich sicher Dimitry Grinberg, das wollen wir doch mal sehen.

Er implementierte einen ARM Emulator auf einem 1284P 8-bit AVR Atmega Chip, verband das Ganze mit einem 16MB Simm Ramchip und einer SD Karte, um von dort die Software zu laden.

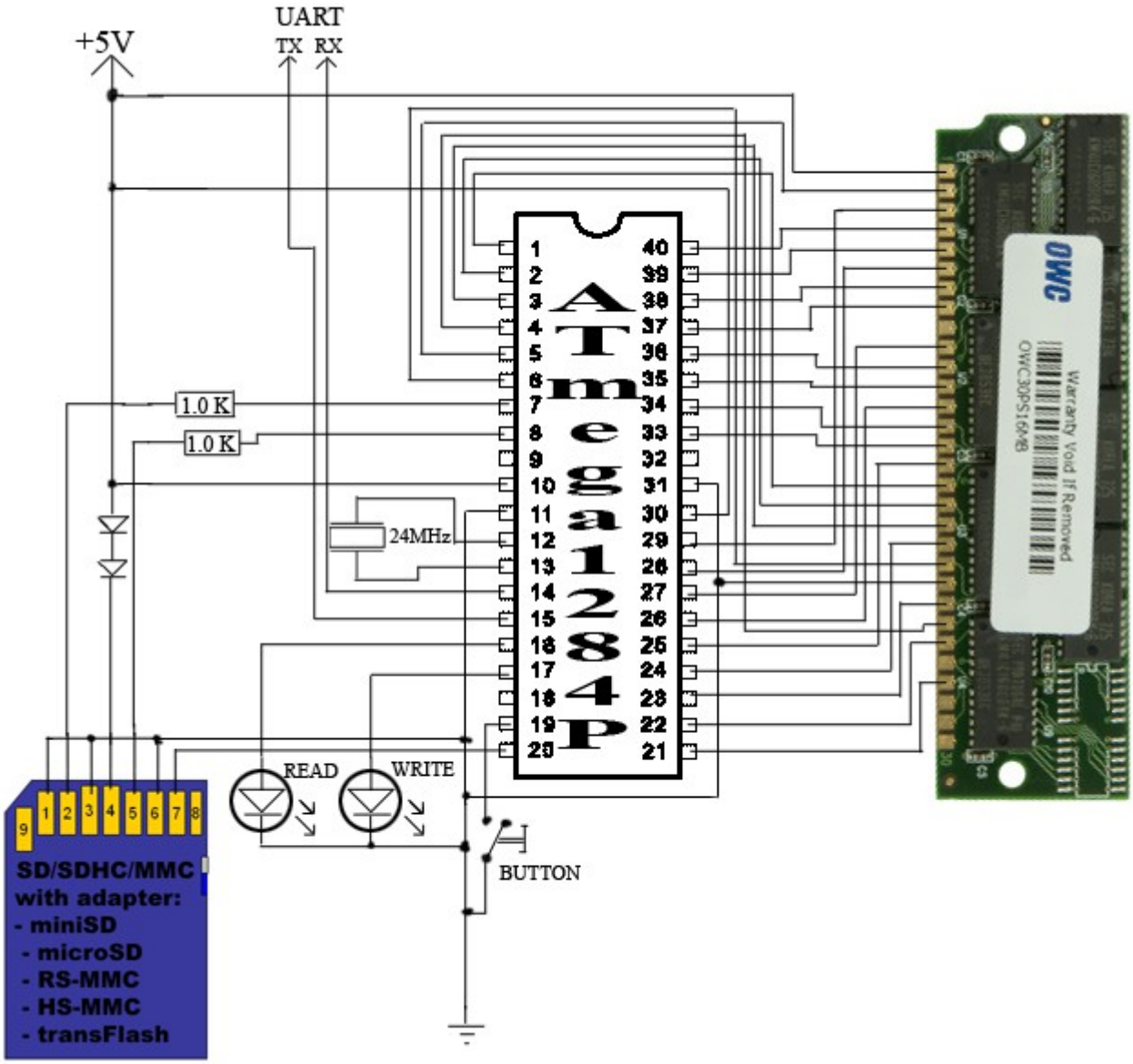
Bild des Aufbaus auf Lochrasterplatte



Wirklich benutzbar ist das System allerdings nicht wirklich. Bis zum Prompt braucht es ca. 2h!

Schaltung

RAM DQ0..DQ7 -> AVR C0..C7. RAM A0..A7 -> AVR A0..A7. RAM A8..A11 -> AVR B0..B3. RAM nRAS nCAS nWE -> AVR D7 B4 B5. SD DI SCK DO -> AVR B6 B7 D6. LEDs read write -> AVR D2 D3 (LED's anderes Bein an GND). Button -> AVR D4 (anderes Bein an GND).
Ram kann irgendein 30-pin 16MB SIMM sein, das CAS-before-RAS refresh of 4K cycles every 64ms hat.

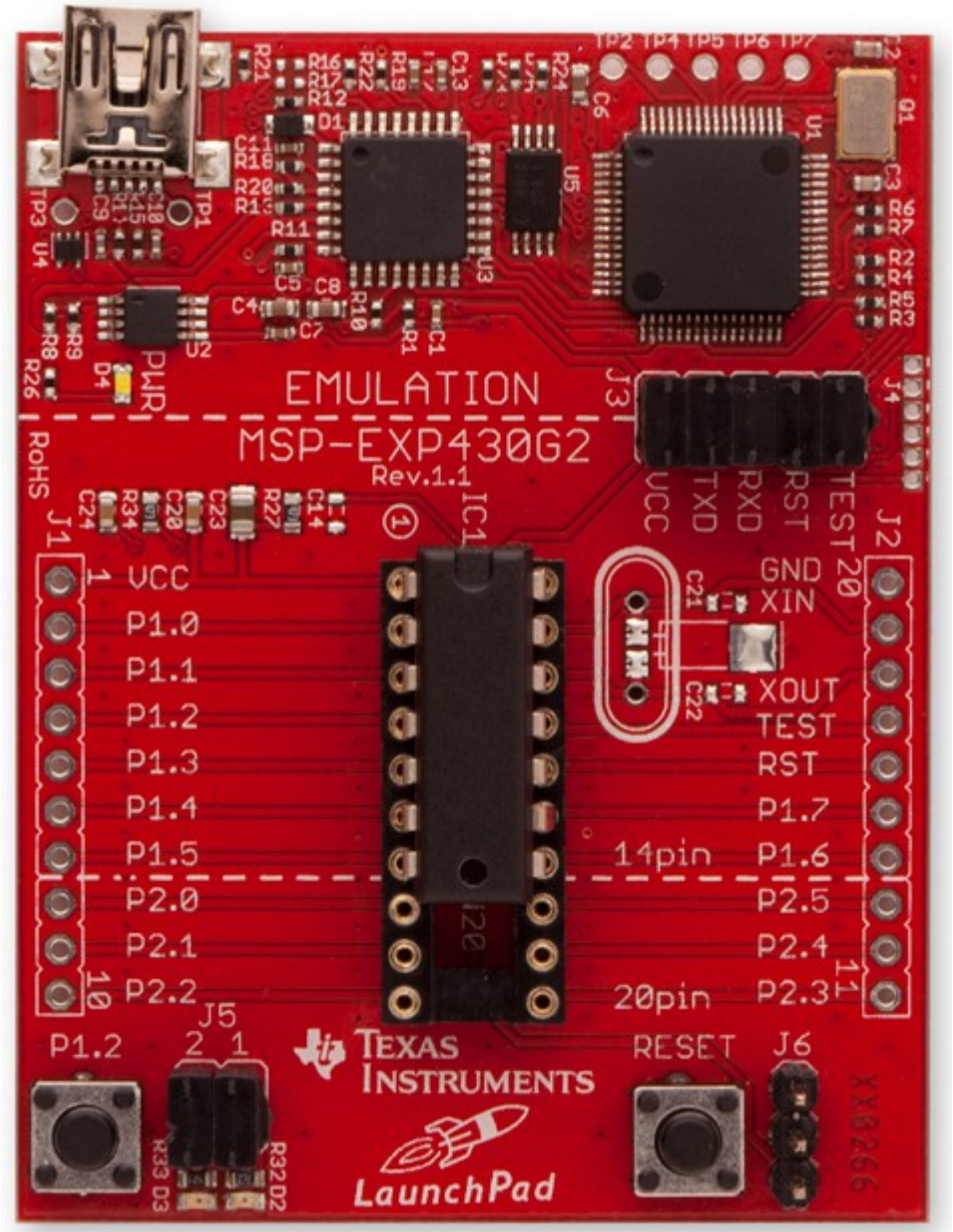


4e4th – Forth auf MCU

Links:

Ti: http://www.ti.com/launchpad
Forth e.V: http://www.forth-ev.de
4e4th Projekt: http://www.4e4th.eu

Kleines ans-forth auf MSP430 MCU im Ti-Launchpad. Keine weitere HW nötig.
Preis: 4,30 US$ / 4€



Forth

Forth ist eine imperative, stackbasierte Programmiersprache. Ein Forth-System beinhaltet ein Betriebssystem zum Ablauf und eine Entwicklungsumgebung zur Erstellung von Forth-Programmen. Daher nennt man ein tatsächlich implementiertes Forth auf einem Rechner ein Forth-System.

Charles H. Moore entwickelte Forth in den Jahren um 1970. Einsatzgebiet waren Steuerungen von z.B Radioteleskopen.

Die wesentlichen Elemente der internen Forth-Architektur sind die beiden Stapel (Data-Stack und Return-Stack) und die sog. Umgekehrte polnische Notation (UPN) von Anweisungen:

Herkömmlich: 3 * (2 + 3) = 15
UPN: 2 3 + 3 * . = 15 (. = Ausgabe des obersten Datenstackwertes)

Installation unter Windows

Benötigt:

- TI Launchpad mit MSP430G2553 MCU
- 4e4th.a43 Firmware
- MSP430Flasher.exe

Launchpad anschließen und Treiber:
430cdc.* installieren lassen (Virtueller COM Port)

Flashen:
MSP430Flasher.exe -i USB -n MSP430G2553 -w 4e4th.a43 -v
Launchpad trennen und **Rx/Tx Jumper um 90° drehen**!

Launchpad wieder anschließen und Terminalprogramm mit virt. COM-Port mit 9600 8N1 ohne Flußkontrolle verbinden = OK Prompt.

Erste Schritte

Berechnungen: 2 3 + 3 * .

Rote LED einschalten: red cset

Neues Wort definieren:
: quadrat
dup * ;

Testen:
8 quadrat .

Alle definierten Wörter listen: words

Morse Code

LED's ausschalten:

red cclr green cclr

Worte kurz und lang definieren:
: kurz red cset 300 ms red cclr 200 ms ; ok
: lang red cset 600 ms red cclr 200 ms ; ok

Buchstaben _s und _o definieren:
: pause 500 ms ; ok
: _s kurz kurz kurz pause ; ok
: _o lang lang lang pause ; ok

SOS morsen:
_s _o _s ok

Natürlich kann man das auf alle anderen Buchstaben erweitern.

FPGA

Ein Field Programmable Gate Array (FPGA) ist ein integrierter Schaltkreis (IC), in den eine logische Schaltung konfiguriert werden kann.

Im Gegensatz zu einer klassischen Programmierung, wo durch das Programm die zeitlichen Abfolgen von Aktionen festgelegt werden, wird durch die Konfiguration die Verschaltung von logischen Einheiten abgebildet die dann parallel und gleichzeitig aktiv sind.

Durch die spezifische Konfiguration können in einem FPGA verschiedene Schaltungen realisiert werden. Diese reichen von Schaltungen geringer Komplexität, wie z. B. einfache Logikschaltungen, bis zu hochkomplexen Schaltungen wie Mikroprozessoren und komplette Computersysteme.

Insbesondere letzteres eröffnet für das Thema dieses Buches interessante Möglichkeiten. Beispiele solcher kompletten Systeme, die mittels FPGA realisiert wurden sind Z1013, C64, CPC bis hin zu Amiga und Atari ST.

Weitere programmierbare Logikbausteine sind:
PAL und GAL – heute kaum mehr verwendet
CPLD (Complex Programmable Logic Device) – wird dort verwendet, wo ein FPGA eher unterfordert wäre, also für einfachere Logikfunktionen

Im Unterschied zu FPGA behalten CPLD ihre Konfiguration auch ohne Strom, da ihre Konfiguation in permanenten Speichern abgelegt ist (Flash). FPGA selbst haben normalerweise nur SRAM für ihre Konfiguration und daher stellt man meistens noch ein Konfiguration-Flash daneben, wo sich der FPGA dann beim Start seine Konfiguration heraus lädt.

Es gibt mehrere Hersteller von FPGA Bausteinen. Die bekanntesten sind:

- Altera
- Xilinx

Beide Hersteller haben neben kommerziellen und teilweise sehr teuren Softwarepaketen zur Entwicklung auch kostenlose Pakete im Angebot für nicht kommerzielle Anwendungen.
Bei Altera ist die Quartus II und bei Xilinix ist es ISE Webpack.

Für beiden Bausteinfamilien gibt es auch sog. Evaluation Boards, die einen FPGA, Konfigurations-Flash, teilweise Ram und verschiedenste Peripherie und Anschlüsse bereitstellen. Eine Programmiermöglichkeit über USB ist ebenso meist bereits enthalten.

Für Altera sind dies z.B. die Altera DE0, DE1 und DE2 Boards. Für Xilinix die Spartan 3/E/AN Boards, die so zw. 100-200€ kosten.

Compukit UK101

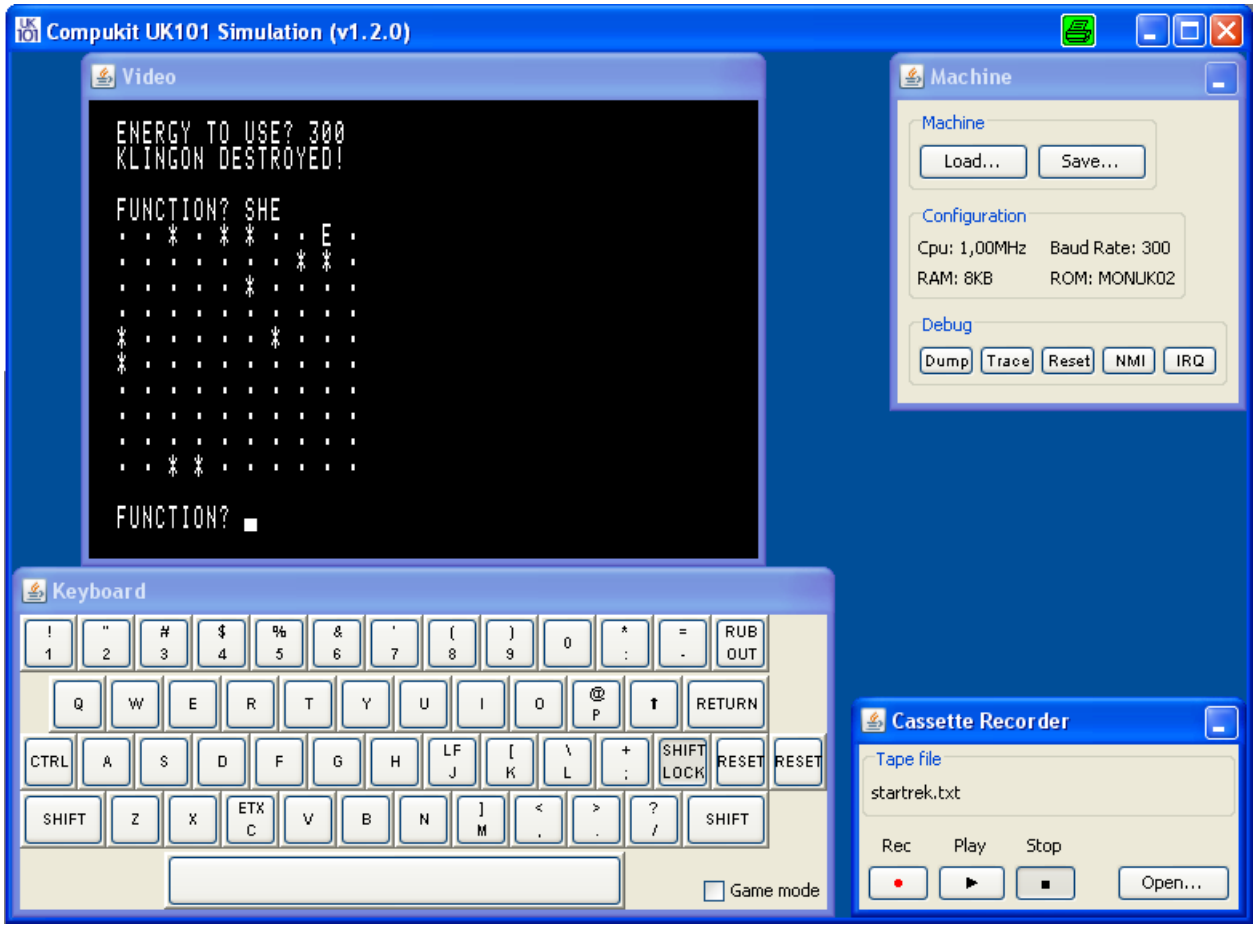
Links:
http://searle.hostei.com/grant/uk101FPGA/index.html
http://uk101.sourceforge.net/

Der Compukit UK101 war ein Einplatinencomputer zum Selbstbau auf Basis 6502 CPU, 2k ROM, 2k CHAR-Rom, 8k Basic-Rom und 8k Ram mit S/W Videoausgabe und Tastatur auf der Hauptplatine. Erscheinungsjahr 1979.

Das Basic war von Microsoft.

Tim Baldwin hat dazu einen sehr schönen Emulator in Java geschrieben. Dank Java somit lauffähig unter Windows, Mac OS X und Unix/Linux.

Java Emulator:

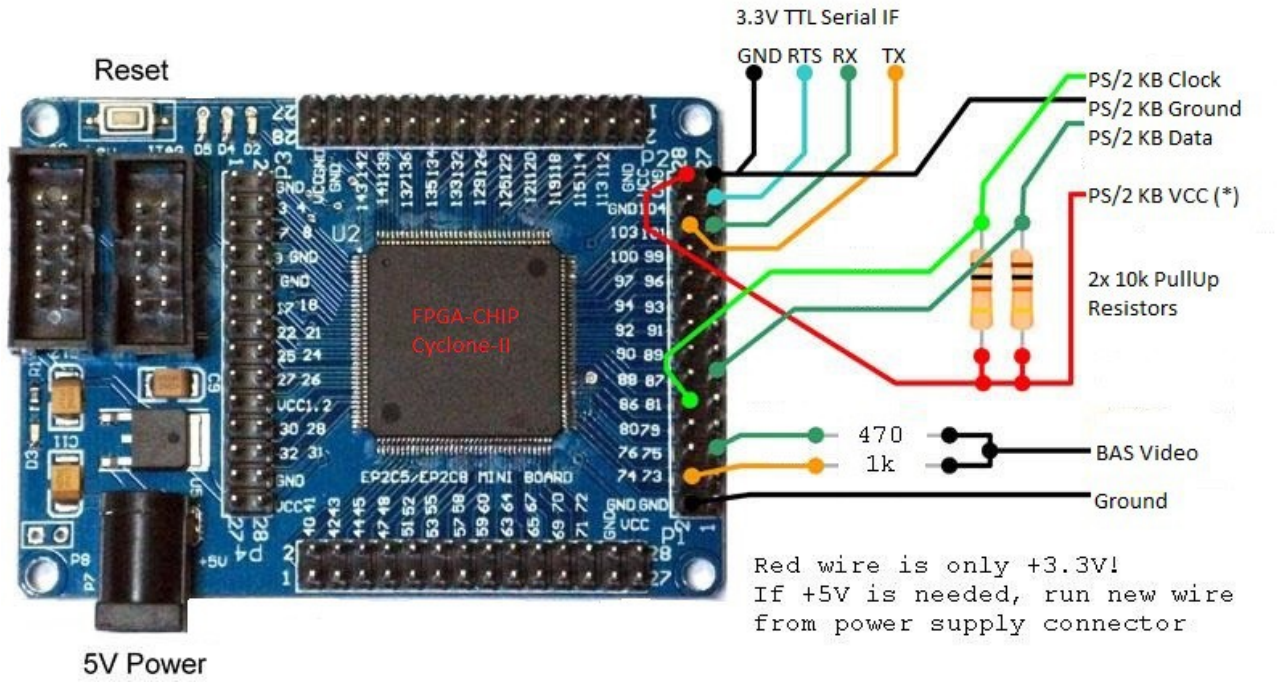


Basicprogramme laden:
LOAD + <Enter>
Am Casettenrecorder Datei laden und Play-Button anklicken.
Nach Laden an der Eingabe <Space>+<Enter> drücken nacheineinder.
Dann sollte sich das geladene Basicprogramm mit LIST listen und mit RUN starten lassen.

Grant Searle hat in 2013-4 den UK101 in FPGA Logik gebaut. Basis ist ein in Anfang 2014 sehr günstiges Altera Mini FPGA Board auf Basis CycloneII. Ohne USB Blaster zum programmieren (<10€) für ca. 14-15€ erhältlich. Die POF Datei wird über Quartus II 13.0 SP1 Programmer mittels AS (Active Serial Programming) in den Flash ECS4 Baustein (ist auf der Unterseite) geflasht.

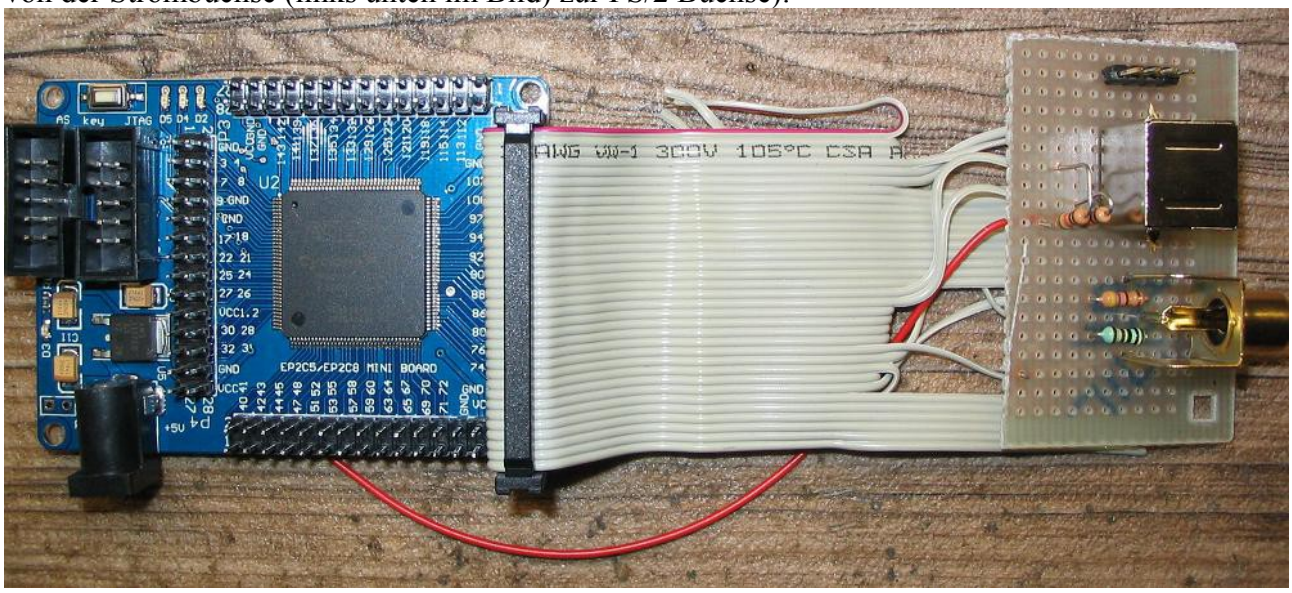
Es wird dann noch eine kleine Schaltung für Videoausgang und PS/2-Tastatur benötigt:

```
Pin - Signal
75  - Video -470--+-------
74  - Sync  -1k---+

86  - Data  -(10k Pullup)-
87  - Clock -(10k Pullup)-
```



Wie man sieht, kann man zum Datenaustausch mit dem PC auch eine serielle Schnittstelle über die günstig erhältlichen USB-TTL (3,3V) Wandler erreichen. Diese Schnittstelle läuft mit 9600 8N1 und Hardware-Handschake (Dazu CTS des Wandlers mit RTS des FPGA Boards verbinden). So lassen sich prima Basic ASCII Dateien mittels Textdatei senden übertragen.

Bild eines aufgebauten Systems inkl. Video und Tastaturadapter (Das rote Kabel bringt +5V direkt von der Strombuchse (links unten im Bild) zur PS/2 Buchse):



Die 4 Pin-Header oben rechts sind für die serielle Schnittstelle.

Einschaltmeldung:

Die UK101 Implementierung ist ein wirklich gut geeignetes Einsteigsprojekt in die Emulation/Simulation von Retro-Computern in FPGA. Die Investitionen sind relativ gering (30-40€). Und Grant Searle stellt neben dem Uk101 noch weitere FPGA Projekte für dieses Board auf seinen Webseiten vor. Darunter auch Z80 basierende Systeme.

Z1013 FPGA System

Links:
http://www.mikrocontroller.net/articles/Retrocomputing_auf_FPGA
http://www.z1013.de/
http://abnoname.blogspot.de/2013/07/z1013-auf-fpga-portierung-fur-altera-de1.html

Volker Urban hat in 2013 ein Z1013 System auf einem Xilinix Spartan3 Board implementiert.

Das Z1013 Computersystem ist ein kleines und einfaches Einplatinen-Bausatz-System auf Z80 Basis der ehemaligen DDR. Neben Z80 CPU gibt es ein Monitor-ROM von 2kb und 16kb Ram in der Grundausführung. Ausgabe erfolgt auf einem Monitor (BAS). Eingabe über eine Matrixtastatur. Programme werden über eine Kassettenschnittstelle geladen und gespeichert.

Volker Urban und zwei Bausteine (Z80 CPU und VGA-Grafikausgabe) von opencores.org verwendet. Der SRAM des Z1013 wurde mit Xilinix BRAM realisiert. Dadurch kann z.B. auch bereits ein ‚geladenes' Programm im Speicher vorbelegt werden (z.B. Tiny Basic/Galactica).

Das 2kb Monitorprogramm hat einfache Kommandos um Speicher anzusehen, zu füllen, zu verschieben und von und auf Kassette zu laden bzw. zu speichern.

User abnoname hat den FPGA Z1013 dann von Xilinix Spartan3 auf Altera DE1 portiert.
Dort ist dann auch das vorhandenen SRAM genutzt worden und eine Programmladefunktion über serielle Schnittstelle wurde implementiert.

Bild des Z1013 mit Monitorfunktionen auf einem Spartan 3E Eval. Board:

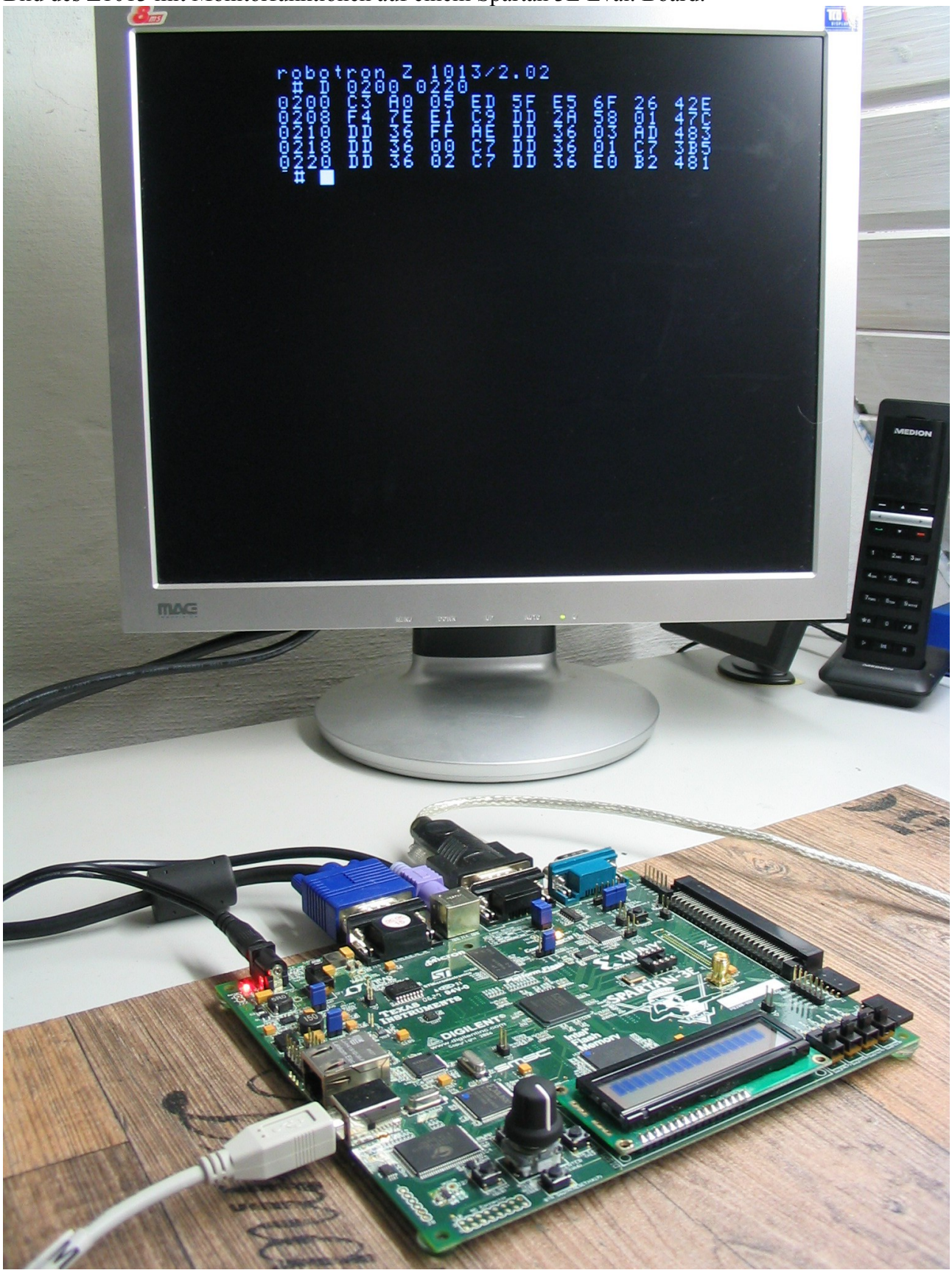
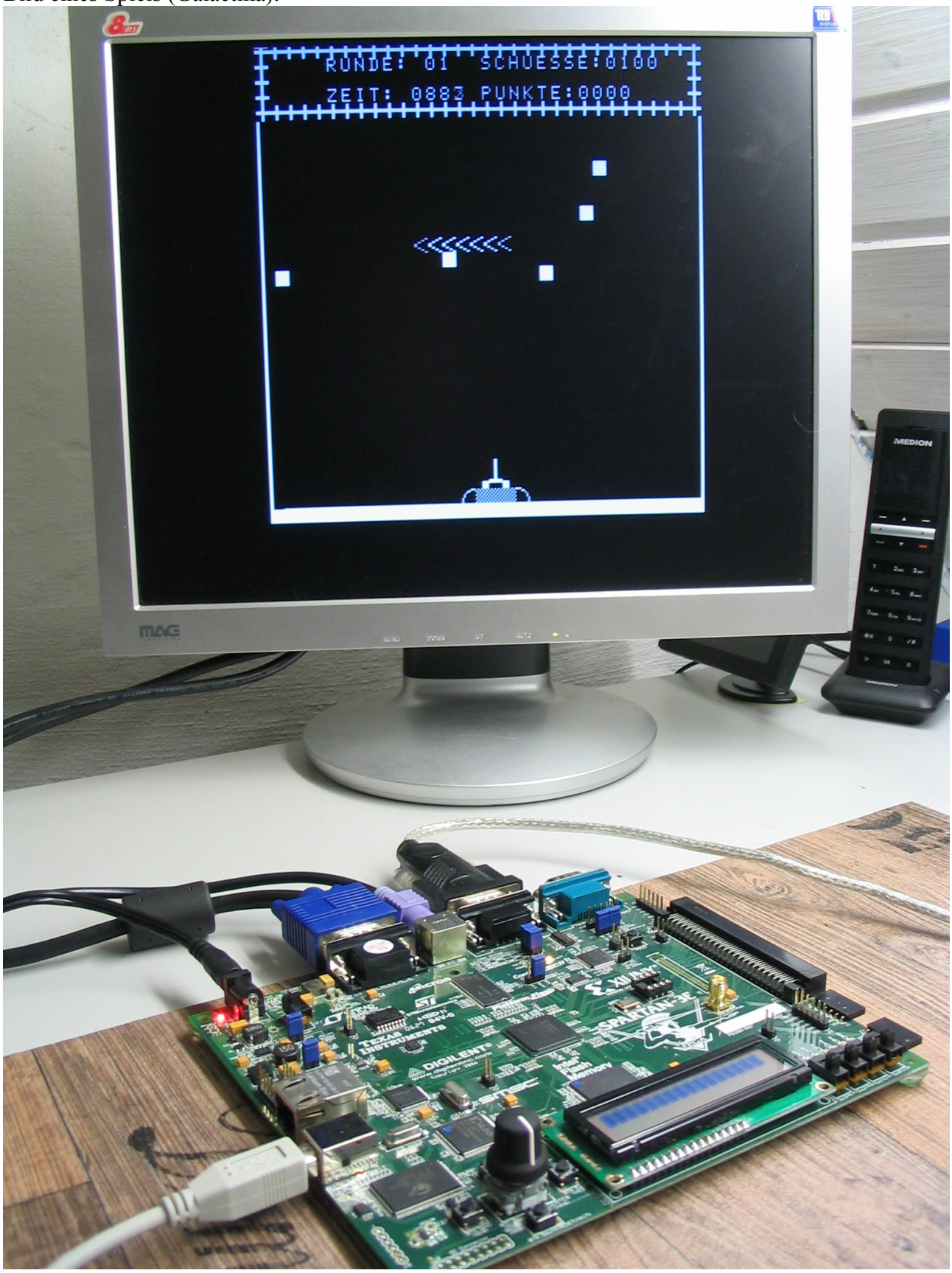


Bild eines Spiels (Galactika):



Rekonstrukt – FPGA Forth System

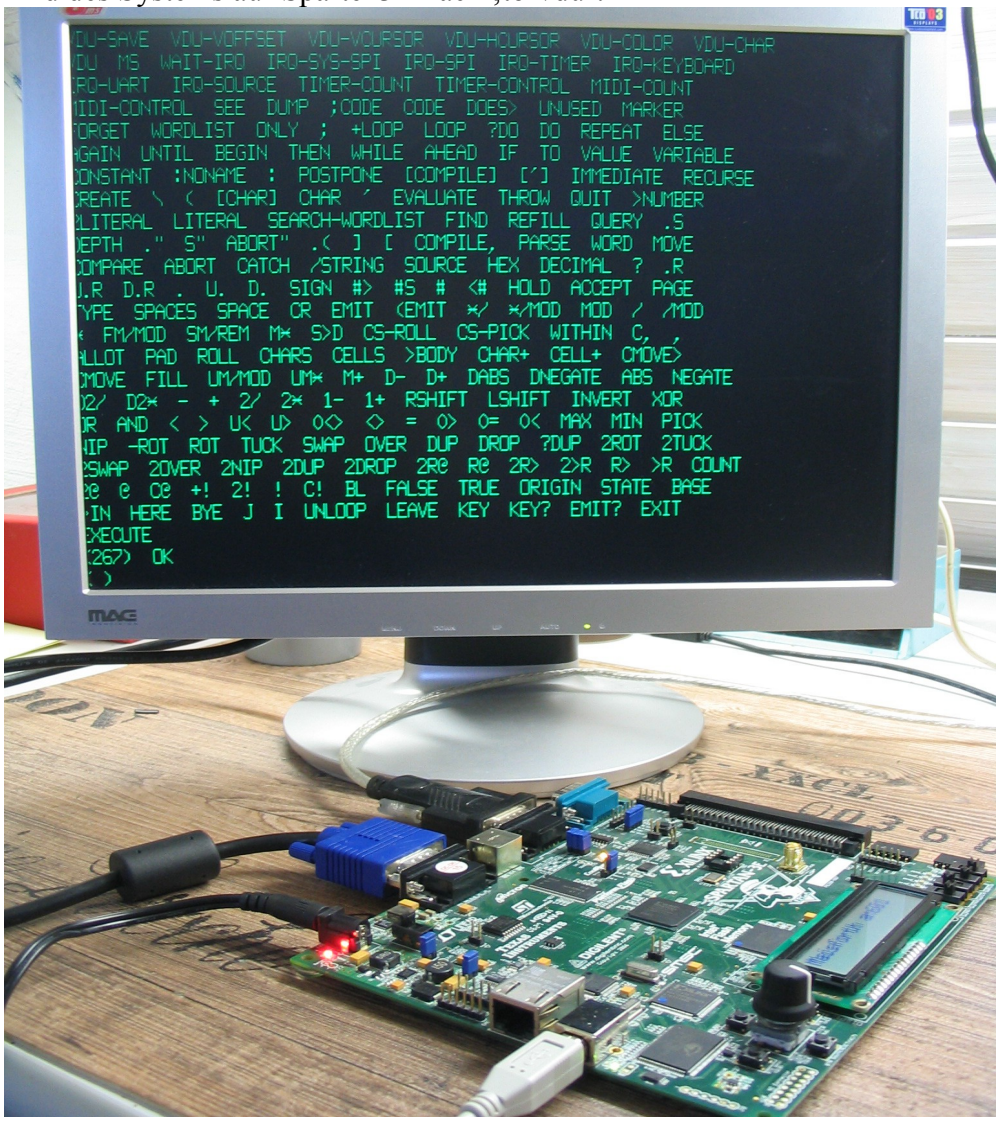
Links:
http://code.google.com/p/rekonstrukt/
http://netzhansa.blogspot.com/2009/02/new-forth-machine.html

Hans Hübner hat ein Forth System auf Basis einer 6809 CPU (system09) und Maisforth auf FPGA Eval. Boards implmentiert. Auch hier wurden wieder Teile von opencores.org verwendet.

Basis ist ein Computersystem auf Basis einer 6809 CPU. Die Kommunikation erfolgt über ein serielles Terminal mit 19200 8N1.

Besonderheit hier ist, das die Peripherie der Boards weitgehend unter Forth zur Verfügung steht. D.h. man kann Ausgaben auf VGA erzeugen, LED leuchten lassen, Schalter abfragen, etc.

Bild des Systems auf Sparten3E nach ‚to-vdu‘:



System09 – VHDL 6809 System on a Chip

Links:
http://members.optushome.com.au/jekent/system09/index.html
http://www.swtpc.com/
http://opencores.org/project,system09

System09 ist eine FPGA-Implementierung eines 6809 Computersystems ähnlich zu dem SWTPC System von John Kent. Es enthält einen kleinen 2kb Sys09Bug Monitor, der Anzeige, Änderungen am Memory ermöglicht, aber auch laden und sichern von Speicher / Programmen über die serielle Schnittstelle. So können dann z.B. selbst geschriebene Assemblerprogramme geladen und gestartet werden.

Die Implementierung enthält:

- 6809 instruction compatible CPU
- Sys09bug Monitor program
- 6850 compatible ACIA/UART
- PS/2 Keyboard interface
- Text based 80 x 25 character Display
- Simple 8 bit timer
- Parallel I/O port
- Hardware breakpoint / Trap for debug purposes.
- IDE Compact Flash interface (Wenn das Board so etwas verfügbar hat)

Eine Portierung ist verfügbar für folgende Boards:

- BurchED B3 XC2S200 Spartan 2
- BurchED B5-X300 XC2S300 Spartan 2
- Digilent XC3S200 & XC3S1000 Spartan 3 Starter boards
- Digilent XC3S500E Spartan 3E starter board (limittierter Speicher)
- XESS XSA-3S100 with XST-3.0 carrier
- Terasic DE1 (bald?)
- Terasic DE2-70 (bald?)

C65 in FPGA

Links:
http://c65gs.blogspot.com.au/

Der C65 war 1991 als Nachfolger des erfolgreichen C64 geplant, kam aber nie in eine Serienproduktion – auch um den Amiga aus eigenem Hause keine Konkurrenz zu machen.

Daher wurden nur zw. 200-400 Stück gebaut (Zahlen schwanken je nach Herkunft), was die wenigen existieren Prototypen extrem teuer macht für Sammler und Liebhaber.

Ausstattung:

- Prozessor: CSG 4510 ("Victor")
- Taktfrequenz: 3,54 MHz
- 128 KB ROM mit: C64 Kernal, BASIC 2.2, BASIC 10.0, DOS, Zeichensätze mit länderspezifischem Tastatur-Layout
- 128 KB RAM, mit RAM-Karte erweiterbar auf bis zu 8 MB
- Videochip: CSG 4567 ("Bill" oder VIC-III) unterstützt alle Videomodi des VIC II; max. 256 Farben aus einer Palette von 4096; Textmodi mit 40/80 Zeichen × 25 Zeilen; Grafikauflösungen von 160 × 200 bis 1280 × 400 Pixel; Kompletter C64 Grafik Mode enthalten; synchronisierbar mit externer Videoquelle (Genlock)
- DMA-Controller (Bit blit)
- DMA-Custom-Chip "DMAgic"
- Floppy Disk Controller ("FDC") F011 (CSG 4571 bzw. 4581)
- Tastatur mit 77 Tasten und abgesetztem Cursorblock
- SID CSG8580 mit 2 × 3 Stimmen (Kanäle gemixt, keine getrennte Stereo-Ausgabe) getrennte Regelung für Lautstärke, Filter und Modulation
- Ein-Ausgabe: RGB, Modulsteckplatz, Userport, zwei Joystick-Ports, Fernsehanschluss (TV-Modulator), RGB, Video-Port mit Composite und S-Video-Signal (beinhaltet auch Audio), Serieller Port für Drucker und Diskettenlaufwerke

Seit Anfang 2014 bis heute (Mitte 2014) arbeitet Dr. Paul Gardner-Stephen daran den C65 auf einem FPGA Board zu implementieren. Inzwischen laufen erste Spiele mit noch kleineren Fehlern. Die Ausführungsgeschwindigkeit liegt z.Z. bei dem Faktor 30. Es gibt aber noch Vieles zu implementieren, zu testen und zu validieren, bevor man die C64 FPGA Implementierung als zu fast fehlerfrei und vollständig einstufen kann. Wünschen wir uns und Paul hier weiterhin gutes Gelingen.

Die ersten Schritte dabei waren zuerst den C64 Mode zu implementieren:

Selbst bis zum funktionierenden Basic 10 waren es gut 5 Monate harte Arbeit:

Erste Spiele laufen (mit noch kleineren Fehlern):

Minimig

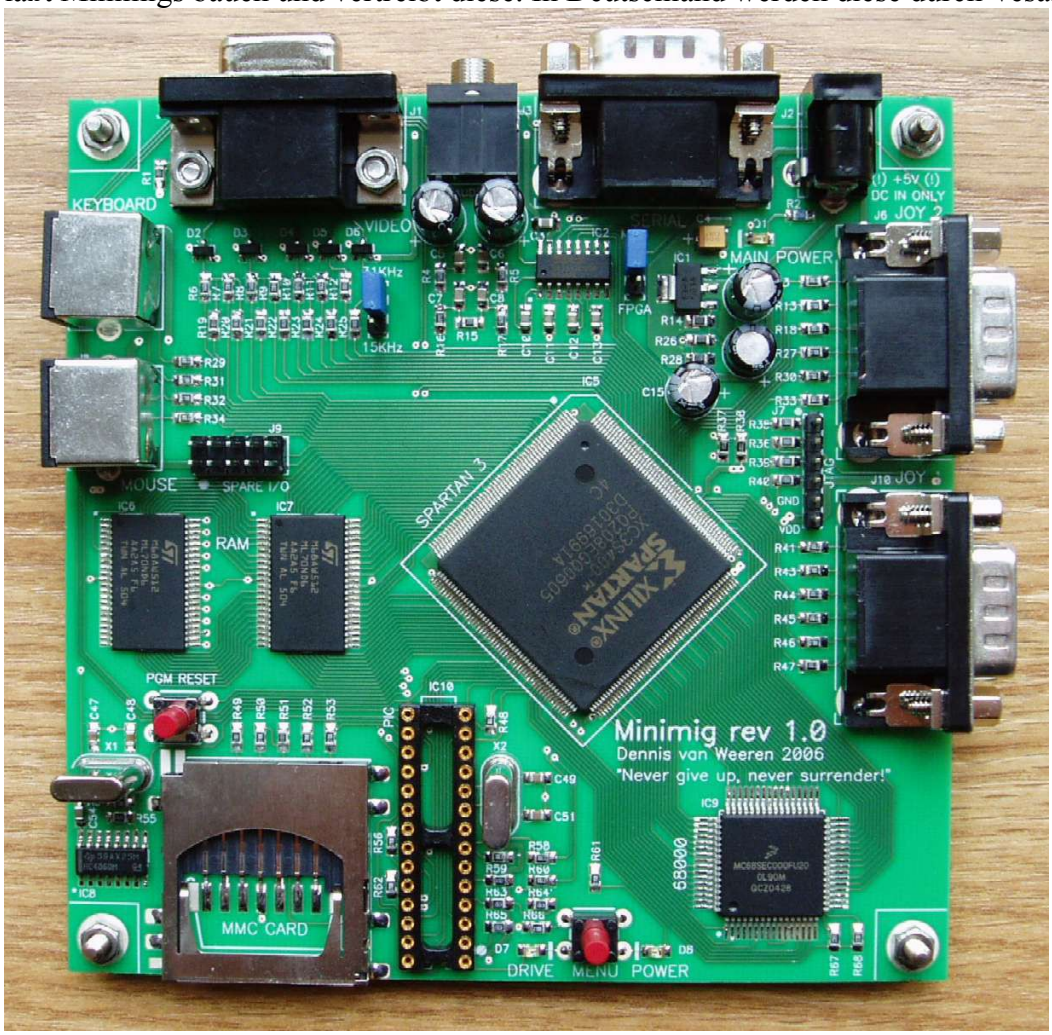
Links

http://home.hetnet.nl/~weeren001/
http://gamesource.groups.yahoo.com/group/minimigtg68/
http://www.amiga.org/modules/newbb/viewtopic.php?topic_id=44173&forum=8&viewmode=flat&order=ASC&start=0
http://home.freeuk.com/fpgaarcade/index.htm

Minimig (Kurzform für Mini Amiga) ist eine Open Source Reimplementation eines Amiga 500 in FPGA (Field Programmable Gate Array). Es wird hier stellvertretend für die immer weiter wachsende Gemeinde der Hardware Emulationen auf Basis von FPGA Bausteinen vorgestellt.

Minimig wurde entworfen und gebaut vom dem Niederländischen Ingenieur Dennis van Weeren. Auslöser waren nicht endende Diskussionen in der Amiga Gemeinschaft, ob sowas möglich wäre. Die Entwicklungszeit ging von ca. Januar 2005 bis ca. Mitte 2007. Am 25.07.2007 wurden die Projektdaten (Sourcecode, Schaltplan) unter der GNU General Public License Version 3 gestellt.

Seitdem sind einige Minimig's in Eigenregie gebaut worden. Aber auch Acube Systems aus Italien läßt Minimigs bauen und vertreibt diese. In Deutschland werden diese durch Vesalia vertrieben.



Minimig auf Altera DE1 Board

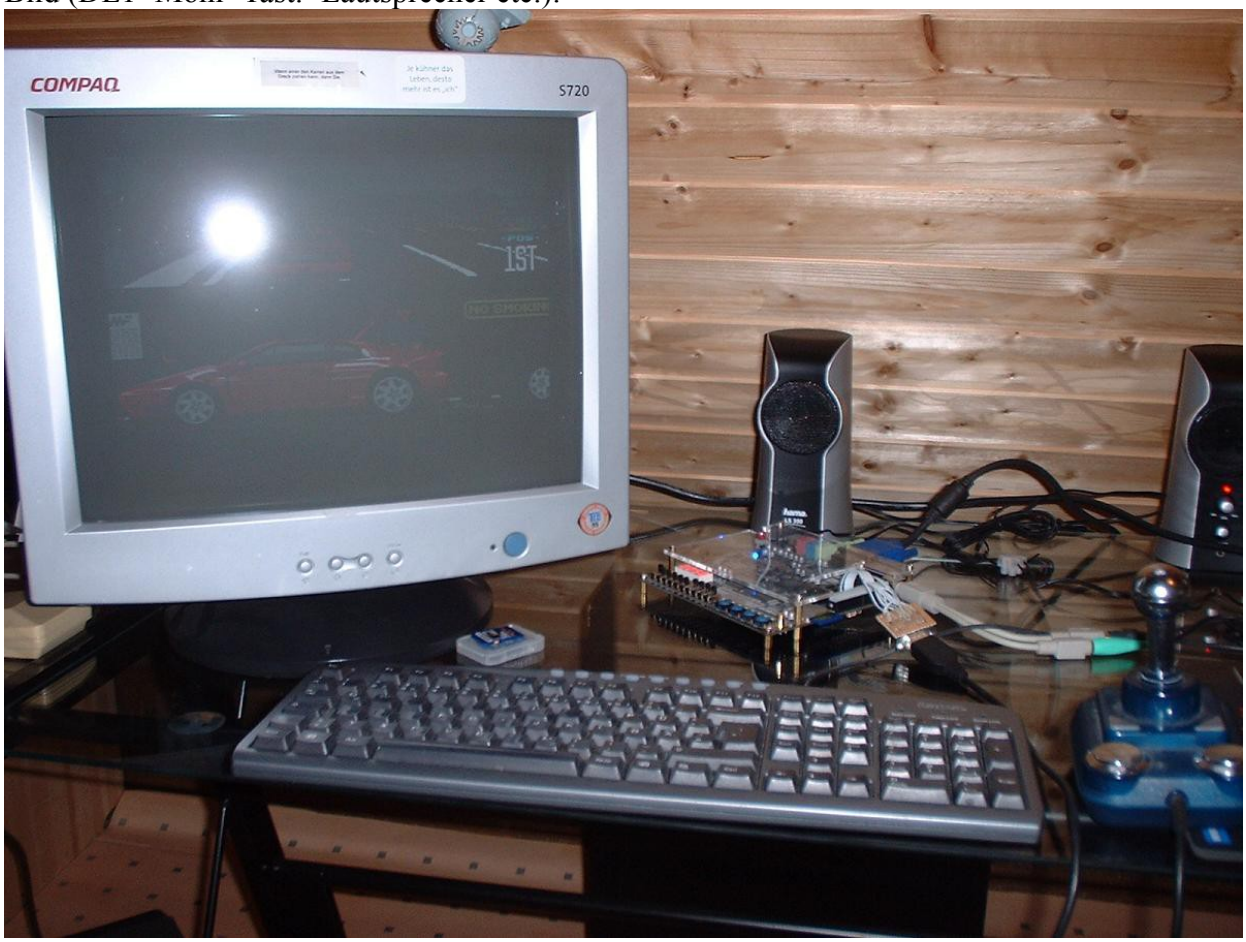
Minimig von Dennis van Weeren ist die Reimplementation eines Amiga 500 in FPGA. Von tobiflex (Tobias Gubener) mit einem 68000 Core erweitert und deshalb lauffähig auf DE1/2 Boards von Altera. Es wird noch eine Kickstart-ROM Datei mit Namen 'kick.rom' und Größe 512kb im Rootverzeichnis benötigt (Kick 1.3 = 256k; einfach mit copy /b kick13.rom+kick13.rom kick.rom zu einer 512kb Datei zusammen kopieren.) Im core Archive ist die Datei 'spihost.rom' enthalten. Diese muß ebenfalls auf die SD Karte ins Rootverzeichnis. Dann noch ein paar ADF Dateien drauf und schon kanns losgehen..

Die Schiebeschalter SW 0-3 und SW9 müßen oben stehen.

SW[9]	Monitorfrequenzumschaltung.
SW[0]	aus-an löst einen Komplett-Reset aus.
SW[3..0]	müssen an sein - damit lassen sich einzelne Teile des Cores resetten.
SW[8..4]	ungenutzt.

Debugausgaben kommen über die RS232 mit 115000 Baud und 8N1.

Bild (DE1+Moni+Tast.+Lautsprecher etc.):



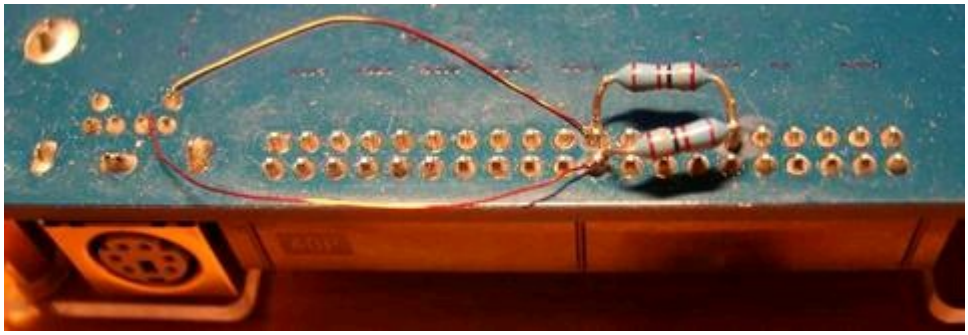
Die SOF Datei ist zum einmaligen Konfigurieren des FPGA, nach dem Ausschalten ist sie wieder weg. Die Programmiersoftware muß dazu im Mode JTAG stehen und ein Häkchen bei PROGRAM/Configure beim geladenen SOF File.

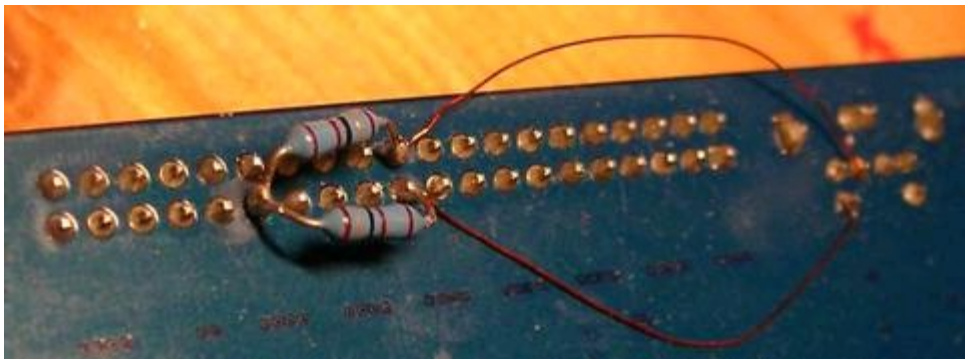
Um die POF Datei ins Start-Eeprom zu programmieren muß der Schiebeschalter links oberhalb der

LEDs Richtung LEDs geschoben werden. Der Programmiermode muß dazu auf Active serial Programmierung stehen. Der Vorgang dauert um einiges länger als die normale Konfiguaration. Haken bei PROGRAM/Configure nicht vergessen. Danach den Schiebeschalter wieder Richtung USB Stecker.

Zum Laden der Pof/Sof Datei benötigt man nicht die volle Quartus II Web Edition (800MB), sondern es gibt auch einen Programmer Only mit ca. 60MB. Für letzteren benötigt man auch keine spezielle Lizenz, welche bei der Web Edition beantragt werden muß (kostenfrei – aber zeitlich begrenzte Gültigkeit).

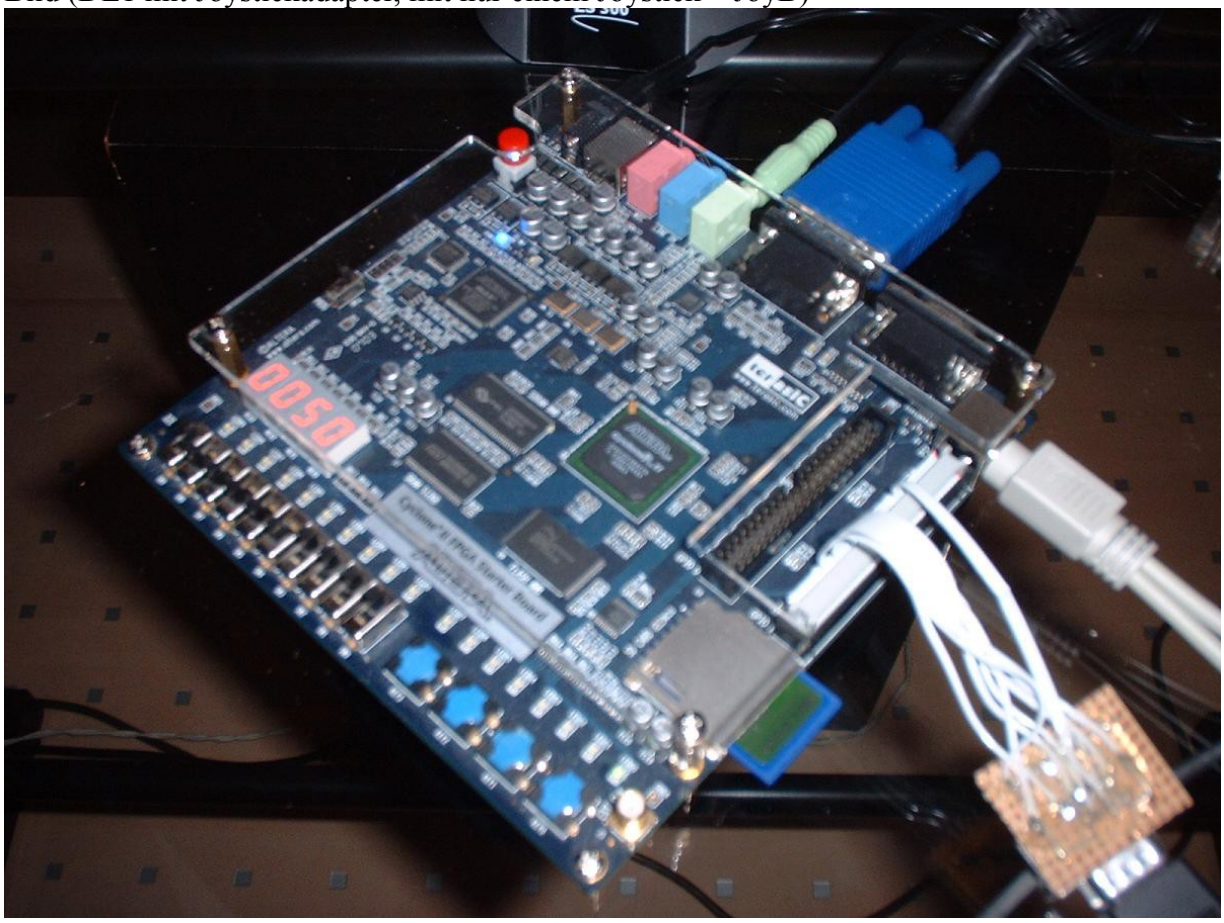
PS/2 Mauserweiterung an vorhandener PS/2 Tastaturbuchse (für Y-Kabel – Achtung: Anschluß umgekeht zur Beschriftung! Tastatur an Mausbuchse; Maus an Tastaturbuchse):



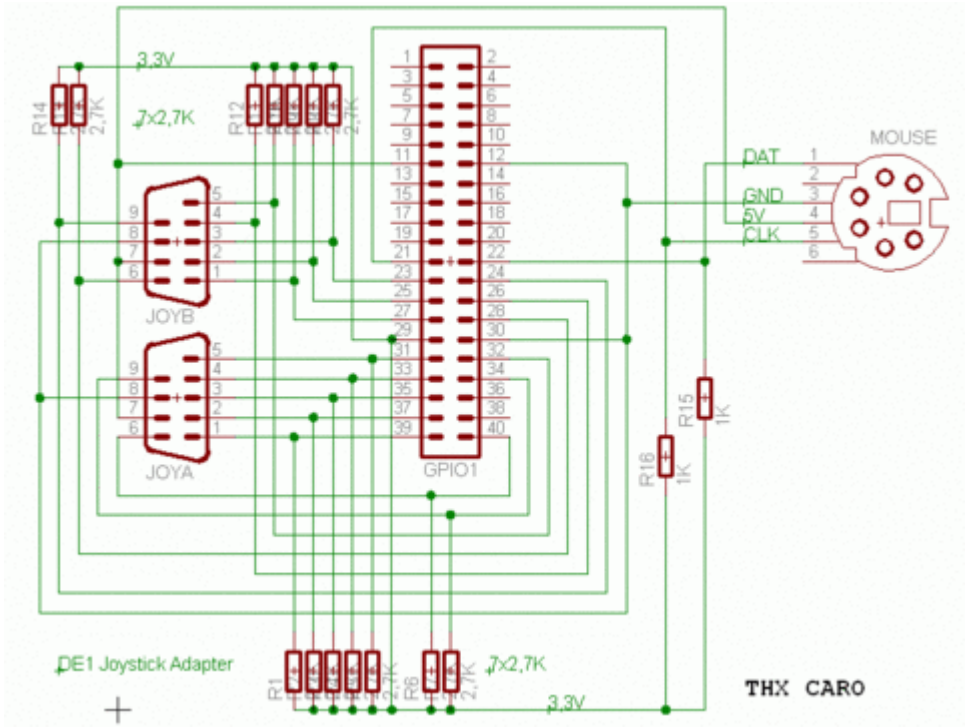


(2 x 1k Widerstand)

Bild (DE1 mit Joystickadapter, mit nur einem Joystick – JoyB)



Schaltbild Joystickadapter



Tastaturbelegung PS/2

F12 - Minimig-Menü aufrufen/beenden
PgUp - Menue hoch
PgDn - Menue runter
Pos1 - Auswahl

Micro KBD - Micro-Tastaturersatz für Minimig

Man nehme: Atmega8, 5 Taster, PS/2-Kabel+Stecker, Lochraster+Kabel, micro_kbd.hex und fertig ist die Miniatur Tastaturersatz fürs Minimig...

C0 = F12
C1 = PgUp
C2 = PgDn
C3 = Home/Pos1
C4 = F1

B0 = Clock-PS/2
D3 = Data-PS/2

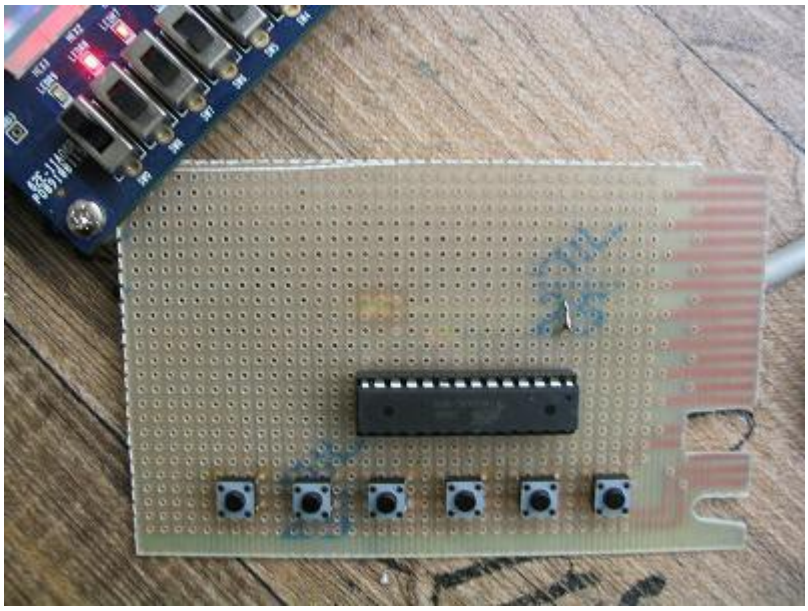
GND und +5V zw. PS/2 und AVR verbinden. Cn = PortC Taster gegen GND.

Sourecode für Micro KBD

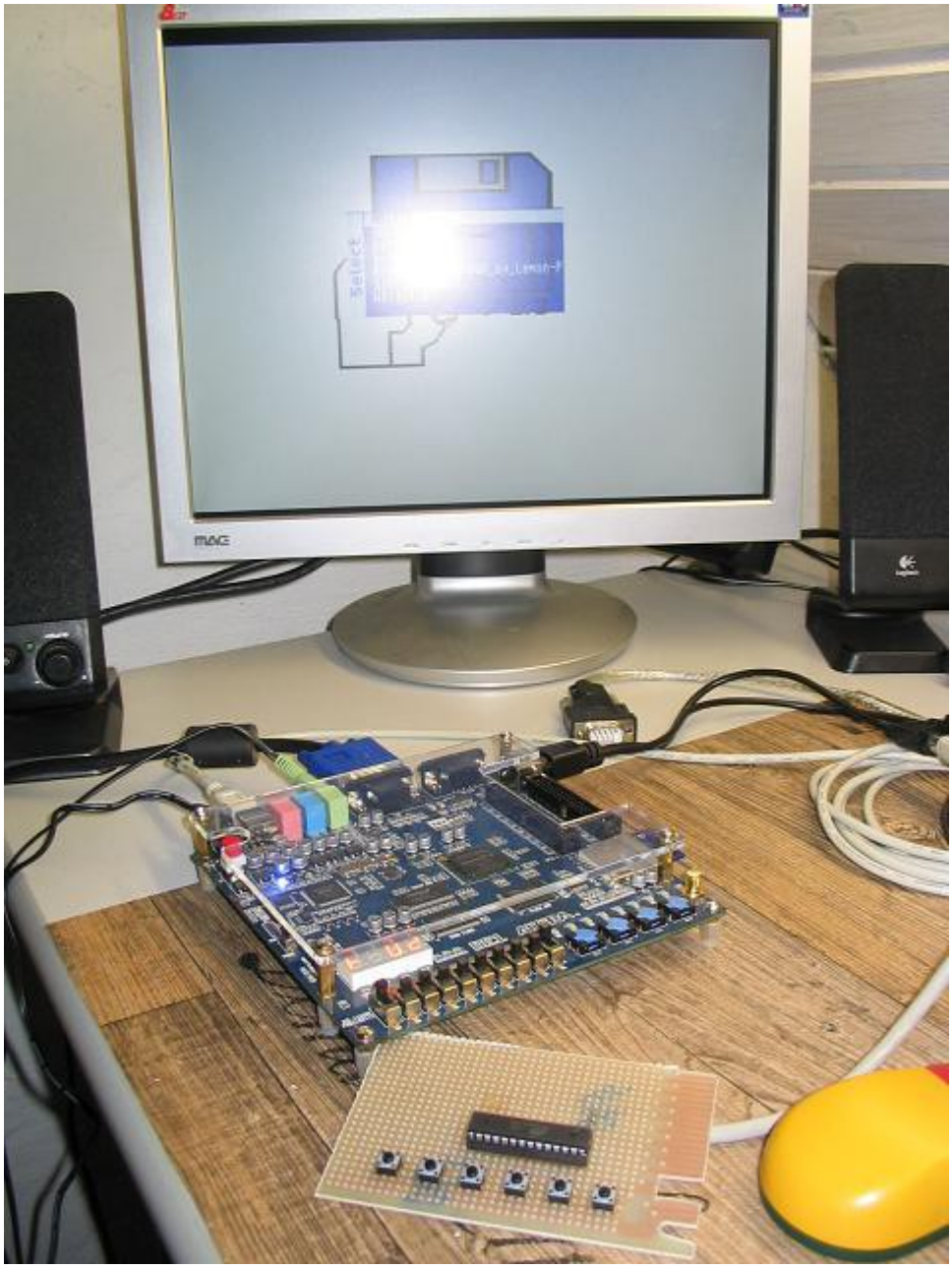
```
'-----------------------------------------------------------------------
'                            micro_kbd.BAS
'                  (c) 2008 peter.sieg1@gmx.de
'                  PS2 AT Keyboard emulator for Minimig
' NOTE THAT THE LIBRARY IS AN OPTIONAL ONE and by default is not included !
'-----------------------------------------------------------------------
$regfile = "m8def.dat"
$crystal = 1000000

$lib "mcsbyteint.lbx"                                       ' use optional lib
since we use only bytes

Config Portc = Input                                        ' port c for taster
Portc = 255                                                 ' pullups on

'configure PS2 AT pins
Enable Interrupts                                           ' you need to turn
on interrupts yourself since an INT is used
Config Atemu = Int1 , Data = Pind.3 , Clock = Pinb.0
'                 ^------------------------ used interrupt
'                              ^----------- pin connected to DATA
'                                       ^-- pin connected to clock
'Note that the DATA must be connected to the used interrupt pin

Waitms 500                                                  ' optional delay
Do
    Waitms 150
    If Pinc.0 = 0 Then Sendscankbd F12                ' send a scan code
    If Pinc.1 = 0 Then Sendscankbd Pgdn               ' send a scan code
    If Pinc.2 = 0 Then Sendscankbd Pgup               ' send a scan code
    If Pinc.3 = 0 Then Sendscankbd Pos1               ' send a scan code
    If Pinc.4 = 0 Then Sendscankbd F1                 ' send a scan code
    Waitms 150
Loop

F12:
Data 3 , &H07 , &HF0 , &H07

F1:
Data 3 , &H05 , &HF0 , &H05

Pgup:
Data 5 , &HE0 , &H7D , &HE0 , &HF0 , &H7D

Pgdn:
Data 5 , &HE0 , &H7A , &HE0 , &HF0 , &H7A

Pos1:
Data 5 , &HE0 , &H6C , &HE0 , &HF0 , &H6C
```

-> http://petersieg.webng.com/minimig/micro_kbd.zip

Aufgebauter Micro KBD



Minimig Gesamtsystem



Danksagungen

An alle die unermüdlich sich dem Erhalt der Informationen, Software, Rominhalte und Systemen der klassischen Rechnertechnik verschrieben haben.

Den vielen Entwicklern von Hard- und Softwareprojekten im Retro-Computing Umfeld, von deren Entwicklungen einige hier im Buch aufgezeigt werden konnten.

Den Mitgliedern und dem Vorstand des Vereins zum Erhalt klassicher Computer e.V.

http://www.classic-computing.de

Dem Forum64

http://www.forum64.de

Dem deutschsprachigem Amiga Forum

http://www.a1k.org

Dem Robotron Technik Forum. Sehr viele interessante Informationen und Berichte der DDR Rechentechnik.

http://www.robotrontechnik.de/

Dem Mikrocontroller Forum

http://www.mikrocontroller.net/

Dem Hack a Day Forum. Hier werden ständig neue Hacks und ‘verrückte’ Projekte vorgestellt.

http://hackaday.com/