ZX-TEAM MAGAZIN

SONDERAUSGABE 1 2012



```
;; START
L0000:  OUT     (SFD),A         ; Turn off the NMI generator if this ROM is
                                ; running in ZX81 hardware. This does nothing
                                ; if this ROM is running within an upgraded
                                ; ZX80.
        LD      BC,$7FFF        ; Set BC to the top of possible RAM.
                                ; The higher unpopulated addresses are used for
                                ; video generation.
        JP      L03CB           ; Jump forward to RAM-CHECK.
;; ERROR-1
L0008:  LD      HL,($4016)      ; fetch character address from CH_ADD.
        LD      ($4018),HL      ; and set the error pointer X_PTR.
        JR      L0056           ; forward to continue at ERROR-2.
;; PRINT-A
L0010:  AND     A               ; test for zero - space.
        JP      NZ,L07F1        ; jump forward if not to PRINT-CH.
        JP      L07F5           ; jump forward to PRINT-SP.
        DEFB    $FF             ; unused location.
;; GET-CHAR
L0018:  LD      HL,(4016)       ; set HL to character address CH_ADD.
        LD      A,(HL)          ; fetch addressed character to A.
;; TEST-SP
L001C:  AND     A               ; test for space.
        RET     NZ              ; return if not a space
        NOP                     ; else trickle through
        NOP                     ; to the next routine.
;; NEXT-CHAR
L0020:  CALL    L0049           ; routine CH-ADD+1 gets next immediate
                                ; character.
        JR      L001C           ; back to TEST-SP.
        DEFB    $FF, $FF, $FF   ; unused locations.
;; FP-CALC
L0028:  JP      L199D           ; jump immediately to the CALCULATE routine.
;; end-calc
L002B:  POP     AF              ; drop the calculator return address RE-ENTRY
        EXX                     ; switch to the other set.
        EX      (SP),HL         ; transfer H'L' to machine stack for the
                                ; return address.
                                ; when exiting recursion then the previous
                                ; pointer is transferred to H'L'.
        EXX                     ; back to main set.
        RET                     ; return.
;; BC-SPACES
L0030:  PUSH    BC              ; push number of spaces on stack.
        LD      HL,($4014)      ; fetch edit line location from E_LINE.
        PUSH    HL              ; save this value on stack.
        JP      L1488           ; jump forward to continue at RESERVE.
;; INTERRUPT
L0038:  DEC     C               ; (4)  decrement C - the scan line counter.
        JP      NZ,L0045        ; (10/10) JUMP forward if not zero to SCAN-LINE
```

In BEARBEITUNG – bitte noch nicht weitergeben!

*** SEMI-HRG *** BILDBETRACHTER *** ANDY'S ULA-NACHBAU *** ZX81 MINI ARCADE *** MEMOPAK-HRG BASICODE *** SPIROGRAPH *** USB-SPEICHER AM ZX81 *** ROBOTERARM *** QL SZENE *** ZXPAND AM ZX80 *** DR BEEP'S 1K GAMES *** VIELES MEHR ***

RUN 

Inhalt

Vorwort von Peter	2
Semi-HRGs auf dem ZX81	4
1K HRG - Mehr Spielspaß pro Byte	10
Der ZX81-HRG-Bildbetrachter	14
Kurz vorgestellt: ZX81 - IDE	19
Spirograph in HRG	20
Neues vonne Küste von Willi	22
Codename ZX83 - Der Sinclair QL	24
Kurz vorgestellt: ZX81 - IDE	27
BasiCode in der MEMOPAK-HRG	28
EPROM mit USB-Treiber für ZX81 64K RAM	32
USB Anschluss mit dem VDIP1 MODUL	38
Kurz vorgestellt: Entwickler Turbo ZX81XT	43
TTL-USB-Zeddies selbst gelötet	44
ZX81-Mini-Arcade	50
Das ZXPAND am ZX80, die Story	54
Roboterarm am ZX 81 (1)	58
Roboterarm am ZX 81 (2)	60
Die Umschaltbox von Ulrich	64
Andy's ULA-Nachbau im englischen Forum	67
Buchrezension: The ZX Spectrum ULA	72
Dies & Das	74

Dieses Magazin ist eine Sammlung von Beiträgen *von und für* Mitglieder des ZX TEAM, zusammengestellt und bearbeitet von

Oliver Lange oliver@zx81.de

Besonderer Dank für die tatkräftige Unterstützung gilt

Jens Sommerfeld
Klaus-Dieter Jahnke
Paul-Olaf Veltjens
Thomas Rademacher

Bitte insbesondere bei Nachbauten beachten: Die Autoren haben größte Sorgfalt walten lassen, dennoch gibt es ***keinerlei Zusagen oder Garantien bezüglich Eignung, Funktionalität oder Zulässigkeit des Dargestellten.*** Bitte unbedingt selber prüfen!

Bei Fragen und Anmerkungen ist auch das Forum http://zx81.tlienhard.com/ eine erstklassige Anlaufstelle, viele der Autoren sind im Forum aktiv.

Vorwort von Peter

„Entdecken Sie die unendlichen Dimensionen Ihres ZX81“

Erinnert Ihr Euch noch an dieses Büchlein, das 1982, also vor immerhin schon 30 Jahren, auf den Markt kam? Wenn nein, macht nichts!**

Der Titel hat es dennoch in sich.

Aber das konnte damals ja noch keiner ahnen. Der ZX81 war gerade erst ein Jahr alt, es gab jedoch schon etliche „bessere“ Home-Computer auf dem Markt, die ohne die vielen Einschränkungen auskommen mussten, die unseren Lieblings-PC auszeichnen. Und genau diese vermeintlichen Einschränkungen, die für viele Computerfreaks ein Grund waren, sich anderen Modellen zuzuwenden, verleiten uns immer noch dazu, uns aufzumachen, die unendlichen Dimensionen des SINCLAIR ZX81 zu entdecken.

Über die vielen Hard- und Softwareentwicklungen seit der Gründung des ZX-TEAMs 1991 muss ich hier wohl nichts schreiben. Jeder Versuch einer Aufzählung wäre entweder unvollständig oder würde den Rahmen sprengen. Fest steht für mich jedoch, dass all die kleinen und großen Projekte tatsächlich schon lange sämtliche vor 31 Jahren bekannten Dimensionen des ZX81 gesprengt haben.

Erfreulich finde ich, dass auch außerhalb des ZX-TEAMs, das jahrelang fast alleine auf der Welt (abgesehen von einigen Einzelkämpfern) die Zeddy-Fahne hochgehalten hat, wieder vermehrt auch internationale Aktivitäten und neue „alte“ Aktivisten für den ZX81 zu beobachten sind.

Moderne Emulatoren und Entwicklungsumgebungen, sowie der ZX81 im ATMega, Ersatztastaturen und eine Nachbau-ULA tragen dazu bei, dass unser Zeddy lebendig bleibt und wir nie wirklich die letzte Dimension in

Sichtweite haben werden. Hinter dem Horizont geht es weiter, das gilt auch für uns.

Damit werden wir immer mehr von den unendlichen Dimensionen des ZX81 entdecken.

Mein Dank geht an die Initiatoren und Redakteure dieser Sonderausgabe des ZX-TEAM-MAGAZINs, für das ich sehr gerne der Bitte nach einem Editorial nachgekommen bin.

peter@zx81.de

*** 106**, das hätte eigentlich schon die Ausgabe 1/2009 des ZX-TEAM-MAGAZINs werden sollen. Ich möchte mich hier noch einmal entschuldigen, dass ich Euch vor drei Jahren so urplötzlich als Redakteur im Stich gelassen habe. Mein Leben hat sich seit dem Eintritt in den Vorruhestand anders entwickelt, als ich mir das lange vorher ausgemalt hatte. Aber keine Bange, es gab keine weiteren Katastrophen, nur meine Prioritäten haben sich doch deutlich verändert.
Von daher freue ich mich ganz besonders darüber, dass ich dem ZX-TEAM damit nicht den Garaus gemacht habe, sondern dass über die ZX81-Foren und unsere jährlichen ZX-TEAM-Treffen viele neue Aktivitäten entstanden sind und erst recht viele neue ZX81-Freunde mit neuen Ideen und neuem Schwung zu uns gefunden haben.

****** „Entdecken Sie die unendlichen Dimensionen Ihres ZX81“ Autor Tim Hartnell, erschienen 1982 im Verlag Cooperation, München. Das Buch war eine Übersetzung aus dem englischen mit dem Original-Titel „Getting acquainted with your ZX81“. Letzteres könnte man mit „Erste Schritte mit Ihrem ZX81.“ übersetzen. Und das hat mit unendlichen Dimensionen kaum etwas zu tun.



Das Büchlein enthält auf 144 Seiten eine Sammlung von einfachen BASIC-Programmen, die zumeist nur kurz beschrieben, aber keinesfalls so dokumentiert waren, dass die Anfänger, für die es geschrieben wurde, sich damit in höhere Dimensionen aufschwingen konnten.

Der englische Titel war einfach ehrlicher.

SEMI-HRG

SEMI-HRG

Semi-HRGs auf dem ZX81

Auch ohne Modifikation an der obligatorischen Speichererweiterung lassen sich dem ZX81 mit einigen Tricks hochauflösende Grafiken entlocken.

Von Th. Rademacher

Auf unserem Zeddy sind bekanntermaßen feiner aufgelöste Grafiken möglich als nur die ursprünglich vorgesehenen Bilder, die aus dem Standard-Zeichensatz erstellt werden können und (bis auf die Schraffuren) aus Klötzchen von 4*4 Pixeln bestehen.

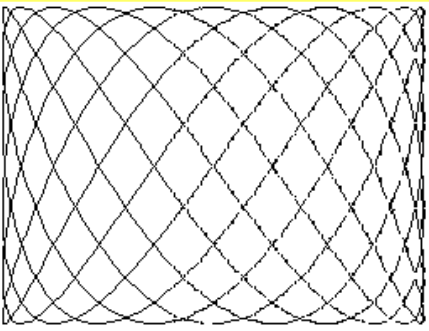


Einige von ANDRE***s Postern, hier allerdings auf ein Viertel ihrer Größe verkleinert. Solche Grafiken in der Auflösung 64 x 48 aus dem Standard-Zeichensatz sind ursprünglich nur auf dem ZX81 möglich.

Wer das Team Magazin 2/2007 (oder die PDF) zur Hand hat, kann vorbereitend noch einmal den Artikel "Grafik auf dem ZX81" nachlesen.

*Warum heute noch Semi-**H**igh-**R**esolution **G**raphics?*

Bereits 1982 brachte die britische Firma Software Farm ein erstes Hi-Res-Spiel für den ZX81 heraus: Forty Niner. [1] Das übertrumpfte Clive Sinclairs eigene Visionen und ließ den Vorsprung des Zeddy-Nachfolgers Spectrum wieder ein wenig schrumpfen. Die Semi-HRGs (auch Pseudo-HRGs genannt) haben den Vorteil, daß sie reine Software-Lösungen sind, außer externem RAM keine Hardwaremodifikationen benötigen, sondern "gleich so" auf jedem 16K-Zeddy funktionieren. Wer sich also erst einmal nur über das Thema informieren möchte und/oder Bedenken hat, selbst den kleinen Eingriff am "Türstopper" vorzunehmen, der den externen RAM grafikfähig macht, hat mit den Semi-HRGs die beste Möglichkeit dazu und sieht hoffentlich gern darüber hinweg, daß die Bilder etwas "krümelig" aussehen.



Eine Lissajous-Figur in Voll-HRG (links) und Pseudo-HRG (rechts)

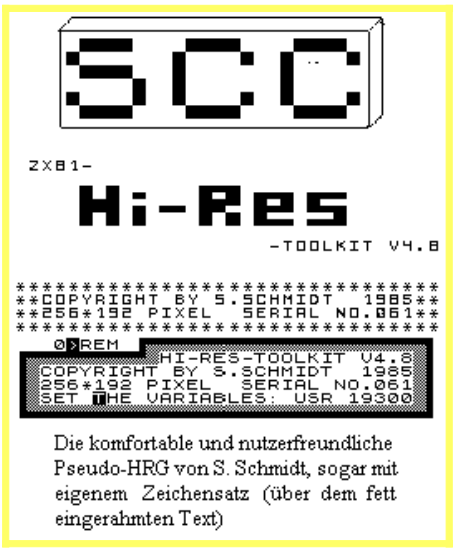
Es gibt verschieden komfortabel ausgestattete Lösungen und es findet sich bestimmt eine, die auf der jeweiligen Konfiguration des betreffenden Nutzers lauffähig ist.

Wie funktionieren Semi-HRGs?

Das Prinzip wurde ausführlich und verständlich in den Heften 5 und 6/1984 der Zeitschrift Happy Computer vorgestellt. [2] Ein auf dem ZX81 abgespeichertes Programm enthält außer den Basic-Zeilen auch die zum Zeitpunkt des Abspeicherns belegten Variablen und Felder und auch den Zustand des Bildschirms im Moment des Speicherns. Im Handbuch wird der Bildschirminhalt als Variable D_FILE bezeichnet. Er besteht aus 24 mal 32 Bytes, jede Zeile wird jeweils noch mit einem Code 118 abgeschlossen. Die Pseudo-HRGs legen eine Variable an, die sich an diesen Aufbau prinzipiell anlehnt. Der Unterschied ist, daß hier 192 (oder mehr, s.u.) Gruppen von 32 Bytes angelegt werden und die Zeilen/Gruppen durch den Code 201 abgeschlossen werden. Der Hintergrund ist, daß sozusagen die Buchstabenhöhe von acht Pixeln auf ein Pixel reduziert wird und dadurch statt 24 Zeilen 192 verwaltet werden müssen.

Wer die Wahl hat ...

Am Artikelende werden kurz die mir bekannten Semi-HRGs mit ihren für diesen Artikel interessierenden Besonderheiten vorgestellt. Das HRG-Toolkit von S.Schmidt ist wahrscheinlich die beste Wahl für Neueinsteiger bei diesem Thema. Es hat einen komfortablen Befehlsumfang (einschließlich eines eigenen Zeichensatzes für Textausgabe im Grafik-Betrieb) und stellt auch im Grafikmodus die gewohnten zwei untersten Zeilen für die Befehlseingabe bereit, umgeht somit Eingaben "im Blindflug".



Die komfortable und nutzerfreundliche Pseudo-HRG von S. Schmidt, sogar mit eigenem Zeichensatz (über dem fett eingerahmten Text)

Die Grafikdaten liegen hier in einer REM-Zeile am Programmanfang, Konflikte mit möglicherweise über RAMTOP liegenden Treibern werden so vermieden.

Lauffähige Programme

Mein besonderes Interesse gilt der Version 3 von BasiCode und ich habe versucht, zu jeder HRG einen Bascoder und ein oder zwei

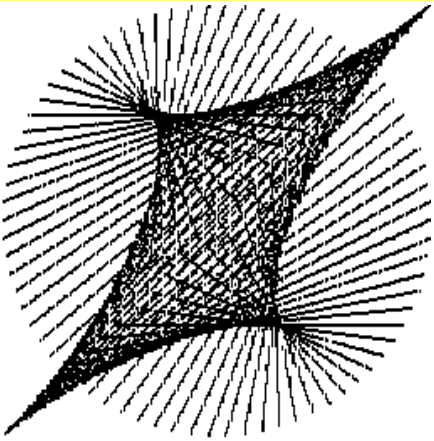
Beispielprogramme zu erstellen. Das bringt den Vorteil, daß die Bedienung vereinheitlicht wird, man sich nicht die überall anders aufgebaute Befehlssyntax einprägen muß. Die Programme hat freundlicherweise Siggi auf seinen Server übernommen. [3]

Wo Licht, da auch Schatten

Ein Nachteil der Semi-HRGs muß hier leider erwähnt werden. Natürlich beanspruchen die MC-Routinen Speicherplatz, der zu Lasten von Basic-Programmzeilen und -Variablen oder -Feldern geht, erst recht trifft das aber für die Grafikdaten zu, von denen für 256 x 192 Bildpunkte immerhin 6336 Bytes zusammenkommen. Bei S.Schmidts HRG-Toolkit bleiben beispielsweise 6287 Bytes frei, 6118 nach Initialisierung der Variablen. Es sind daher nur vergleichsweise weniger aufwendige Programme realisierbar - für das Spirograph-Programm habe ich deswegen auf die schnellere Abarbeitung durch vorbereitete Tabellen verzichten müssen und lasse stattdessen die erforderlichen Sinus- und Cosinuswerte immer wieder neu berechnen - der Platz gabs einfach nicht her... Die französische HRG7.0 hat sogar 222 und die Heupt-Version 256 Bildzeilen. Das ist mir erst später aufgefallen, deswegen soll hier noch einmal die komplette Grafik der Demo von Stefan Heupt abgebildet werden.

Kompatibilität

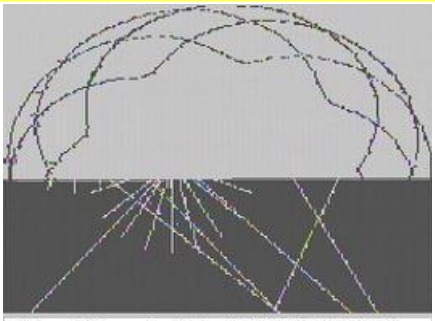
Bei der Beschäftigung mit den diversen Pseudo-HRG-Lösungen kam mir der Gedanke, ob die Ansammlungen von Grafikdaten untereinander kompatibel sind. Die Zuordnungen in den Manuals zur Taylor-und zur Körber-HRG ließen diese Theorie aufkommen.



Als Nachtrag zur Seite 25 von TM 2/2007 die komplette Grafik, die durch Stefan Heupts Demo, einer Pseudo-HRG mit 256 Zeilen, erzeugt wird.

Die nachträglich entwickelten Massenspeicherlösungen (von der Commodore-Floppy bis zum USB am VDRIVE2) am ZX81 bringen Befehle mit, die das Speichern und Laden von Speicherabzügen ermöglichen, mithin kann man die Bilder als .B-Dateien aus dem RAM extrahieren und in einer anderen HRG wieder in die dort verwendete Speicherposition einbauen. Soweit die Theorie, die sich

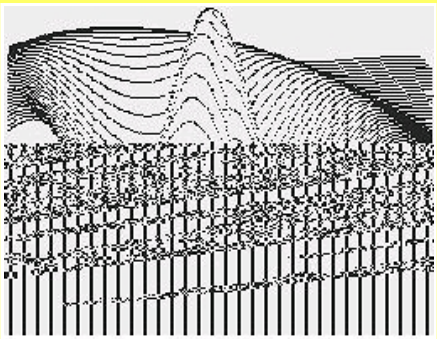
aber auch prompt in der Praxis als zutreffend bestätigte. Hier möchte ich Klaus für seine geduldige Mitarbeit danken, er überprüfte solche Sachen auf der realen Hardware, ich verfüge nur über emulierte Zeddies.



Hier wird gerade mittels USB-BLOAD-Befehls eine unter der französischen Pseudo-"HRG7.0" erzeugte Grafik in den Bildatenbereich des "Hi-Res-Toolkit" von S. Schmidt eingelesen: eine Demonstration, daß die Grafikbytes unterschiedlicher Semi-HRGs in ihrem Aufbau untereinander kompatibel sind.

Coming soon? - Diashow in Semi-HRG

Nun blieb noch eine Frage offen: können Bilder aus Voll-HRGs so umgewandelt werden, daß sie (wenn auch mit Qualitätseinbußen) in Pseudo-HRGs dargestellt werden können? Das "bröselige" Aussehen der Semi-HRG-Bilder ist dadurch verursacht, daß nicht alle (256) denkbaren Kombinationen von acht benachbarten Pixeln wirklich korrekt darstellbar sind, sondern nur nicht einmal die Hälfte davon; die restlichen müssen durch möglichst ähnlich aussehende ersetzt werden, bei denen halt mal ein Pixel fehlt oder eins zu viel gesetzt wird. Man braucht also eine Übersetzungstabelle, die jedem der 256 möglichen Voll-HRG-Bytes ein Byte der Semi-HRGs zuordnet. Diese Tabelle stellte ich zusammen, indem ich mit dem PLOT-Befehl Entsprechungen von Dualzahlen in den Bildschirm schrieb und dann mit PEEK auslas, welches Byte hier dadurch erzeugt wurde (Programm MISSING.P, in Anlehnung an Darwins „missing link“, das fehlende Bindeglied in der Evolution der Semi- und Full-HRGs).



Unterm Emulator NO$ZX8.EXE läßt sich die Transformation von Voll- nach Pseudo-HRG beobachten, auf dem Original-Zeddy nicht, da die (für die Semi-HRG) teilweise ungültigen Grafikbytes und die fehlenden Codes 201 am Zeilenende zum Absturz führen und deswegen erst nach der Erstellung des kompletten Bildes in den Grafikmodus umgeschaltet werden kann. Aus Platzmangel muß die Voll-HRG-Grafik im Bereich der Pseudo-Grafik-Bytes abgelegt werden (sichtbar im unteren Bereich des Bildes). Die Bytes werden anders interpretiert: acht ungesetzte Pixel haben in der Voll-HRG den Code 0, was in der Pseudo-HRG aber das Bitmuster 00011000 erzeugt - die senkrechten Balken in der unteren Hälfte werden nach der Umwandlung wieder leerer Hintergrund, haben dann aber den Code 158.

Auch dieses Projekt gelang schließlich und es entstand das Programm UMWA01.P, in das man per USB-BLOAD-Befehl ein Full-HRG-Bild hineinladen kann und nach der Umwandlung als Semi-HRG-Bild per BSAVE abspeichern kann. Auch für diese Programme gilt, daß sie von Siggis Server oder von mir bezogen werden können.

Kurzvorstellung einiger Semi-HRGs

Nils Körber 1983 "HIRES-TK" RAMTOP-Senkung vor dem Laden POKE 16389,103 und NEW Ort der Grafik-Daten h6700-h7FBF (192 Pixelzeilen) Beispiel für den PLOT-Befehl: POKE 16516,X POKE 16517,Y LET FK=USR 17020
Richard Taylor 1983 "16K ZX81 HIGH RESOLUTION" RAMTOP-Senkung vor dem Laden POKE 16389,103 und NEW Ort der Grafik-Daten h6700-h7FBF (192 Pixelzeilen) Beispiel für den PLOT-Befehl: POKE 16507,X POKE 16508,Y LET ZZZ=USR 17023
Stefan Heupt [4] RAMTOP-Senkung wird im Programm vorgenommen Ort der Grafik-Daten h5EA8-h7FA7 (256 Pixelzeilen) Beispiel für den PLOT-Befehl: POKE 17189,X POKE 17190,Y RAND USR 17177
HRG 7.0 1983 J.M.Cohen [5]*, nicht zu verwechseln mit der (Voll-)HRG7 von "DIDI", 1985, die Joachim im ZXTM 1/91 vorstellte* RAMTOP-Senkung vor dem Laden POKE 16389,99 NEW Ort der Grafik-Daten h6362-h7FFF (222 Pixelzeilen) Beispiel für den PLOT-Befehl: POKE 16507,X POKE 16508,Y RAND USR 16528
HRG aus "Happy Computer" 1984 Helmut Tisch(n?)er [6] RAMTOP-Senkung wird im Programm vorgenommen Ort der Grafik-Daten h6740-h7fff (192 Pixelzeilen) Beispiel für den PLOT-Befehl: PRINT USR 16881,X,Y
S. Schmidt 1985 (HRG Toolkit) RAMTOP-Senkung nicht erforderlich Ort der Grafik-Daten h4B7D-h643C Beispiel für den PLOT-Befehl: *(nach Initialisierung der HRG-Variablen mit RAND USR 19300) PRINT USR PLOT,X,Y*

Th. Rademacher / Januar 2012

Quellen und Weblinks zu den Semi-HRGs

[1] http://www.pictureviewerpro.com/hosting/zx81/softwarefarm.htm, ZX-TEAM-CD 2007 \ZX81\programme\PETER\FORTY.P

[2]ftp://ftp.worldofspectrum.org/pub/sinclair/magazines/HappyComputer/Issue8405/Pages/HappyComputer840500044.jpg und nachfolgende Seiten, auch des nächsten Hefts

[3] http://zx81-siggi.endoftheinternet.org/d:applic/hrg/basicod3/

[4] ZX-TEAM-CD 2007 \ZX81\programme\PETER\HIRES.P

[5] zx81stuff.org.uk/zx81/generated/tapeinfo/h/HRG70.html

[6] ZX-TEAM-CD 2007 \ZX81\programme\PETER\HRG.P und niederländische Version \ZX81\programme\PETER\HI-RES.P

HISTORISCHE WERBUNG

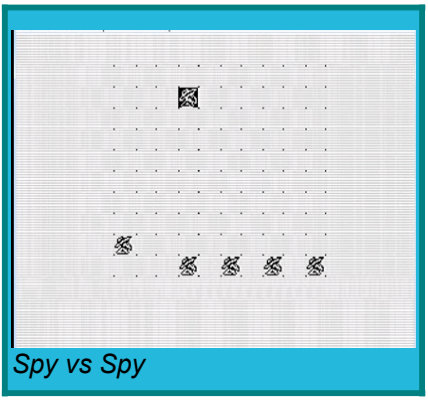


1K HRG - Mehr Spielspaß pro Byte

Wer diese Spiele mit hochauflösender Grafik auf dem Bildschirm sieht, wird erstaunt sein, was sich aus dem ZX81-Grundgerät mit seinen 1024 Byte RAM Speicher herausholen lässt.

Von „Dr Beep“

Interviewer/Einleitung: Oliver



Spy vs Spy

Der ZX81 wurde von seinen Machern für reine Text-Darstellung vorgesehen, Grafik war im Zeddi-Basic nur als grobe 64 x 48 Blöcke möglich. Dieses Prinzip hat seine Vorteile, man spart RAM-Speicher, der ja in den Achtzigern noch knapp und teuer war. Spricht man von HRG auf dem ZX81, denken viele an ein spezielles Zusatzmodul oder an eine mit Dioden modifizierte 16k-RAM-Erweiterung, die solch hochauflösende Grafik ermöglicht. Jedoch ist echte HRG mit Einzelpixelansteuerung tatsächlich schon mit dem eingebauten 1K RAM möglich, ganz ohne Modifikation.

Dr Beep aus dem Forum [1] hat diese Technik zur Perfektion getrieben und begonnen, eine Serie von 1K-HRG-Spielen zu schreiben. Die Spiele sollen nach Fertigstellung für den guten Zweck verkauft werden, die Einnahmen kommen dem Kampf gegen Krebs zu Gute. Hier der Aufruf von Dr Beep im Forum:

> Zum 30. Geburtstag des Zeddy's bin ich angefangen ein spezialer Pakete von Spiele zu programmieren. Diese Paket wird 10 Spielen enthalten welche ALLE auf ein 1K ZX81 laufen werden.
>
> Nichts spezials daran? Doch!
>
> Es sind 10 Spiele in HiRes auf ein 1K ZX81!
>
> Das Paket würde angboten für etwa 10 pound von CronoSoft und alle Verkauf wird als 'SoftAid' geschenkt als '1K Fightagainst Cancer'.
>
> Ziel ist für 1000 pound (= 1K) zu verkaufen, also 100 Pakete.

Zu Details und Hintergrund haben wir den Autor befragt - ein Interview mit Dr Beep:

***ZXTM**: Wie bist du zu Sinclair-Computern gekommen?*

Dr Beep: Seit 1983 habe ich schon einen ZX Spectrum. Der ZX81 ist erst Ende 1995 dazu gekommen, als jemand gerne 3D Monster Maze spielen wollte, jedoch keinen ZX81 mehr hatte. Dann bin ich angefangen ein ZX81 Emulator für den ZX Spectrum zu programmieren. Das Programm konnte man dann auch auf seinem SAM Coupe laufen. In April 1997 sind SAM81 (SAM) und ZX81EMUL (Spectrum) als freie Version (PD) herausgekommen. Im Jahr 2006 habe ich dann SAM2ZX81 programmiert, womit man Hi-Res-Spiele am SAM Coupe laufen lassen kann.

***ZXTM**: Ist Programmieren für dich Hobby oder auch Beruf?*

Dr Beep: Programmieren ist für mich ein Hobby, aber manchmal habe ich die Kenntnisse auch schon für meinen Job genutzt - bei Besprechungen mit Entwicklungsabteilungen.

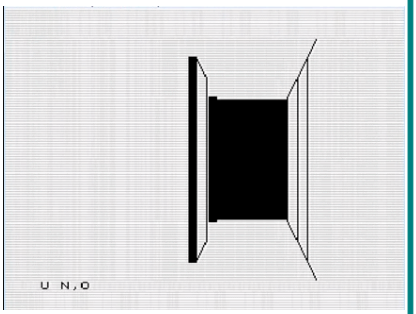
***ZXTM**: Wie bist du darauf gekommen, 1K-HRG-Spiele zu schreiben?*

Dr Beep: Seit 2007 programmiere ich MiniGames zum MinigameCompo. Das sind Spiele mit 1K, 2K oder 4K Dateigröße auf dem Datenträger (z.B. Tape). Dabei darf man aber mehr RAM-Speicher der Maschinen nutzen, durch Kompression kann man da viel erreichen.

Ich wollte das ZX Spectrum Spiel SHOGUN für den Zeddy umschreiben. Da das Spiel auf dem Spectrum UDG-benutzerdefinierte Grafikzeichen nutzt, würde ich am Zeddy Hi-Res brauchen. Mittels einer Google-Suche habe ich es gefunden und dann Shogun für den Zeddy umgesetzt.

Bei der Suche traf ich aber auch auf ein 1K HiRes Demo - nur einige Kreise in einem kleinen Bild. Dann kam die Idee: "Kann man auch ein Spiel in 1K Hires machen? "

Mit diesen Routinen als Vorbild wurde "Wiwo Dido, the case of Mazeddy's Castle" programmiert.



3D Mace [2]

***ZXTM:** Was ist die besondere Herausforderung?*

Dr Beep: Da Speicher knapp ist muss man sich viele Tricks ausdenken, um alles in 1KByte zu codieren. Beim 3D Maze z.B. wird das Labyrinth sogar komprimiert aufgebaut. Während

beim ZX Spectrum oder beim 16K Zeddy jedes Feld 1 Byte nutzen kann, werden beim 1K Zeddy nun 4 Felder in einem Byte gespeichert. Dazu braucht man dann ein größere Ausleseroutine, aber die ist kürzer als die eingesparten Daten zum Labyrinth. Um Programme in 1K Speicher zu bekommen müssen Routinen oftmals optimiert werden. Einige Beispiele für kürzeren Code:

```
up INC B
  defb 62; Hide DEC B in LD A,N
         ; no JR TEST needed,
         ; save 1 Byte
down DEC B
test LD A,B
  CP 8
  JR NC,FALSEMOVE ; only 0 -7
                  ; allowed
```

aber auch Timing: 40 T States in nur 3 Bytes (= 10 Positionen am Bild):

```
  PUSH HL
  EX (SP),HL
  POP HL
```

***ZXTM**: Was ist die technische Grundlage zur 1K-HRG-Technik?*

Dr Beep: Alle Spiele haben eine eigene Hi-Res-Routine. Es gibt kein Beispiel welches für alle Spiele zu nutzen ist.

Der Trick: Auf bestimmte Weise alle Bilddaten in wenig Speicher unterbringen und dann für jedes Spiel eine eigene spezielle Bildroutine einbauen.

Beispiele: "Wiwo Dido" hat ein kleines Bild von 5x5 Character, also 25x8 Bytes. Das Bild wird nur für die X/Y Koordinaten aufgebaut, wo sich die Spielfigur gerade befindet - alle anderen Bildbereiche sind leer. Sobald eine Bewegung stattfindet wird ein neues Bild aufgebaut und an der neuen XY-Position dargestellt.

"Blocky" hat einen Speicher von 8x32 Bytes, das sind max. 8 Bildbereiche mit Daten (4x Character, 4x Platforms). Während des Bildaufbaus wird ein Zähler benutzt – sobald der Zähler der Bildposition entspricht wird eine Bilddatenzeile geschrieben, sonst eine Leerzeile.

"Bowling" ist sehr ähnlich zu "Blocky", aber statt einer Leerzeile wird eine Zeile mittels Anfangs- und Endpixel aufgebaut, um so ein Bahn zu erhalten.

Jedes Spiel hat eine Besonderheit - entweder in der Hi-Res-Routine oder im Programm. „Wiwo Dido“ nutzt einen kürzeren Bereich, damit die X-Position gesetzt wird. „Blocky“ nutzt den Character selber zum auslesen der Bit-Position. „Return to Mazeddy“ baut jede Bildbzeile 3x auf. „Ghost Hunt“ nutzt zwei Bild-Routinen, eine für den Geist und eine für das Kreuz. „3D Maze“ ändert den Bildspeicher während des Aufbaus. „Othello“ schreibt die Bildzeile bis zu 14 mal, wobei der Bildspeicher gleichzeitig Brettspeicher ist. Bowling nutzt zwei Teilbilder - feststehende Kegel und beweglicher Ball.

„Spy vs Spy“ setzt das R-Register immer neu auf das entsprechende Zeichen. Das Zeichen wird dabei bei Bedarf invertiert oder gelöscht. Das Bild kommt dabei also nicht als Ganzes aus dem Speicher, sondern die Zeile wird durch die Aufbauroutine zehn mal verändert dargestellt.

***ZXTM**: Welche Art von Spielen eignet sich für 1KByte-Programme?*

Dr Beep: Zuerst sollte das Spiel nur wenige genutzte Bildzeilen haben oder sehr kurze Bildzeilen.

„Wiwo Dido“ hat 40 Bildzeilen mit Länge 5. „Blocky“ hat nur 8 Bildzeilen, aber mit Länge 32. Am besten funktionieren Spiele welche mit einer Bildzeilenlänge von 4/8/16/32 arbeiten, wobei dann nie die 256-Grenze oder 128-Grenze des R-Registers überschritten wird.

Bei anderen Längen muss man sich auf einen Hi-Res-Speicher kleiner als 256 Bytes beschränken *und* die Grenze von 128 oder 256 sollte richtig behandelt werden. Bildzeilen kürzer als 32 werden immer mit Leerzeichen bis 32 aufgefüllt.

Ein Labyrinth mit 3D Ansicht funktioniert, da dieses Bild nur ein halbes Bild im Speicher braucht, und dann wird alles nochmal gespiegelt aufgebaut. Ein Zwei-Spieler „TANK BATTLE“ oder „GUN FIGHT“ ist auch möglich, aber dann wird man von oben nach unten schießen. Hinzu kommen alle Spiele, die man in 6x8 Zeichen darstellen kann (das war bei mir ein 15-Scheiben-Puzzle).

Die letzten Spiele des Pakets waren zum Zeitpunkt des Interviews noch in Entwicklung. Durch eine Datenpanne wurde Dr Beep danach leider zurückgeworfen.

Hoffen wir, dass diese großartigen Programme sehr bald in großen Stückzahlen für die gute Sache über den elektronischen Verkaufstresen gehen. Verdient hätten es die kleinen Kunstwerke mit Sicherheit.

Quellen und Weblinks

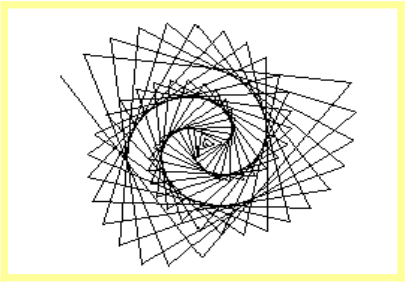
[1] *http://forum.tlienhard.com/phpBB3/viewtopic.php?f=2&t=592*

[2] http://www.youtube.com/user/zx81hires

HRG BILDBETRACHTER

Der ZX81-HRG-Bildbetrachter – ein Traum wird wahr

Bildershow vom USB-Stick mit „fließenden“ Übergangen im SLOW-Modus



Von K.-D. Jahnke und T. Rademacher

ZX81-HRG-Bilder haben ihren ganz besonderen eigenen Reiz. Mit nur 192x256 schwarzen und weißen Bildpunkten kann man nicht nur Strichmännchen oder mathematische Kurven darstellen, sondern auch Digitalfotos nachdem man sie entsprechend aufgearbeitet und gerastert hat. Mit dem USB-Zeddy ist jetzt auch die Bildbetrachtung am laufenden Band möglich indem die Bilder direkt und “live“ vom USB-Stick geladen werden.

Wir haben den HRG-MS Grafiktreiber (Version 2.6) von Matthias Swatosch für die Bilddarstellung gewählt und das READ DATA/RESTORE MC-Programm von G.W. Seefried (1983), sowie den Vdrive USB-Treiber von Thomas Lienhard in der TTL Modifikation von Oliver Lange (VDR15.P, 2011) um den Traum von der Zeddy-HRG-Dia-Show wahr werden zu lassen.

Hardwarevoraussetzungen und HRG-Bildherstellung

Als notwendige Hardware braucht man einen USB-Zeddy mit HRG-fähigem 16K RAM sowie maximal 251 HRG-Bilder im Full-HRG-Format, die man entweder mit dem USB-BSAVE-Befehl aus der HRG-MS-Anwendung herauskopiert oder mit PC-Bildbearbeitungsprogrammen aus regulären Digitalbildern herstellt hat, so, wie in ZXTM 02/2006 beschrieben. Mit einem 64kRAM Zeddy kann man die Bildumwandlung auch direkt mit GTRANS.P auf dem ZX81 durchführen so, wie von Joachim Merkl im ZX81 Forum beschrieben. Das GTRANS-Programm existiert in verschiedenen Versionen und ermöglicht die Umwandlung von BMP-Dateien in ZX-HRG-B-Dateien (und umgekehrt) für HRG8 und HRG-MS 2.x (Grafikbank 3).

Zurück zu unserem ZX81 Bildbetrachter. Die Bildnamen müssen standardmäßig dem 12345678.B-Zeichenformat (= 8Zeichen +Dezimalpunkt+B, z.B.BILD1000.B) entsprechen, damit sie korrekt erkannt und decodiert werden können. Ist der Name kürzer, so muss er auf 8 Zeichen verlängert werden. Die Bilddateien (12345678.B), das BASIC-Programm (ZX81DIA1.P) und der 16k-HRG-MS-Treiber (HRG-16K.P) müssen sich im selben USB-Unterverzeichnis befinden.

Laden und Benutzen des Programms

Zuerst lädt man den HRG-MS-Treiber. Auf dem 56K-TTL-USB-Zeddy geht das so: PRINT USR 8192, "LHRGMS-16K.P"

Der HRG-MS-Treiber installiert sich automatisch über RAMTOP und schafft dort Platz für die Bilddatei in Grafik-BANK 3 (Adresse 26592 (=h67E0) bis Adresse 32736(=h7FE0)), die wir für unser Bildprogramm benötigen. Die anderen HRG-MS Grafikbänke 4,5,6 und 7 werden nicht benutzt.

Jetzt wird das Bildbetrachtungsprogramm geladen mit dem USB-Befehl: PRINT USR 8192, "LZX81DIA1.P"

Das Programm wird mit GOTO 2000 (nicht RUN) gestartet und schon geht es los. Die Menüpunkte sind selbsterklärend. Man kann das Programm mit der Standard-BREAK-Taste unterbrechen und bei Bedarf wieder mit GOTO 2000 starten . Innerhalb der Ladesequenzen der Grafik kann man durch anhaltendes Drücken der Null-Taste (0) zum Hauptmenü zurück.

Wenn sich im Grafikmodus eine Datei nicht ladbar ist, so wird nur der aktuelle Bildspeicherinhalt permanent angezeigt und man kommt durch Halten der Null-Taste zurück zum Hauptmenü.

```
****** ZX81 HRG-DIA-SHOW *******
***** START MIT GOTO 2000 ******
*****************************

T. RADEMACHER UND K.-D. JAHNKE
         JANUAR 2012

      SYSTEMANFORDERUNGEN
ZX81 MIT 16 K RAM  HRG-FAEHIG
HRG-TREIBER HRG-MS >= VERS. 2.2
USB-VDRIVE ALS BILDSPEICHER
MAXIMAL 251 BILDER
ABBRUCH DER EINGABE MIT "0"
*****************************
<1> ALLE BILDER NACHEINANDER
<2> ALLE BILDER ZUFAELLIG
<3> VORWAERTS BLAETTERN
<4> RUECKWAERTS BLAETTERN
<5> BILDNAMEN/DATENBANK AENDERN
<6> ENDE
*****************************
```

Programmaufbau ZX81DIA1.P

Das Programmlisting ist in großen Teilen selbsterklärend. Entscheidend für sein Funktionieren sind unter anderem die Zeilen 10 bis 40, die unverrückbar an dieser Position bleiben müssen, da sie vom Programm selbst modifiziert werden. Zeile 10 enthält den USB-Bildladebefehl, der vor dem Laden jedes Bildes verändert und dann ausgeführt wird. Zeile 30 enthält das READ

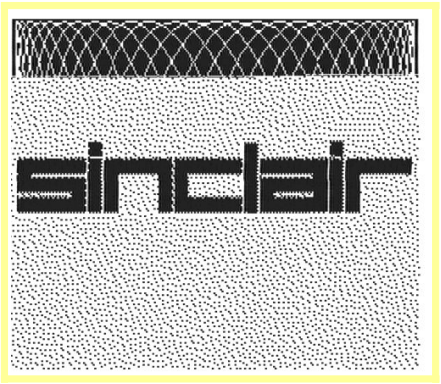
DATA/RESTORE Maschinenprogramm und Zeile 40 die Datenbank für die Bildnamen.

Die in Zeile 40 aufgelisteten Bilder müssen sich als HRG-Bilder (Bildlänge je 6144 Byte) auf dem USB-Stick befinden. Bei Verwendung des USB-Treibers von Oliver Lange (VDR15.P) sieht man den Ladevorgang "live" , indem die Bilder von oben nach unten wie ein Vorhang ausgerollt werden. Andere USB-Treiber laden das Bild im FAST-Modus und führen zum Flackern des Bildschirmes.

Bekannte Schwächen („known bugs")

Die Programmversion 1.0 (ZX81DIA1.P) wurde ausführlich getestet und läuft relativ absturzsicher. Dennoch gibt es zwei Phänomene, die wir bisher nicht erklären konnten. Die Bildnummern 49, 99, 177 und 199 sind nicht möglich und führen bei Verwendung zum Programmabsturz. Sie werden deshalb durch die Programmzeilen 485, 1140 und 1310 verhindert und ausgeschlossen.

Außerdem wird beim Speichern des Programms auf USB in Zeile 10 und manchmal in Zeile 40 (Bild 33) ein inverses Cursor-K geschrieben (genannt die "K-Krankheit"). Beim nächsten Laden des ZX-Programms führt der Fehler in Programmzeile 10 zu einer Fehlermeldung, die man aber beseitigen kann, indem man Zeile 10 erneut eintippt. Das Löschen des inversen Cursor-Ks ist nicht möglich. Das Phänomen scheint nur aufzutreten, wenn man zuvor den BASIC-Programmcode editiert und dann das Programm laufen lassen hat



Bildschirmkopie vom USB-Ladevorgang

Danksagung

Thomas Lienhard und Oliver Lange möchten wir an dieser Stelle für ihre Schaltpläne der ZX81-USB-VDrive-Hardware und die Erstellung der USB-Treibersoftware danken.

Referenzen (Auswahl)

G.W. Seefried: Data für den ZX81. ZX-User Club 1983, Seite 59.
M. Swatosch.: Handbuch HRG-MS 2.6 für den ZX81(2009).
http://www.swatosch.de/zx81/index.html
T. Rademacher:. Bilder von anderen Computern auf den ZX81 übertragen. ZX-Team Magazin 2/2006,
T. Lienhard: USB-Speichersticks am ZX81. ZX-Team Magazin 05/2007

T. Lienhard: Das USB-Interface und seine Folgen. ZX-Team Magazin 02/2008
O. Lange. USB-VDrive-Interface und interner HRG-RAM – Die Sparversion. 06.01.2011 In: ZX-Team Forum (http://forum.tlienhard.com)
J. Merkl. Grafiktransformation mit GTRANS. E-Mail 2009 und 2010 ZX-Team Forum (http://forum.tlienhard.com)
K.-D. Jahnke. TTL-USB-Zeddies selbst gelötet. ZX-Team Magazin Sonderausgabe März 2012

Programmlisting ZX81DIA1.P (Version 1.0, Januar 2012)

```
  10 PRINT USR
8192,"L.........B,67E0"
  20 RETURN
  30 REM
11F340ED53EC40C9ED5BEC402A10
407EFE46280323 18F80103000922
EC4006083E00772310FC2AEC4001
000013ED53EC401AFE1AC8FE7628
C703772318EE00002941
```

Zeile 30 enhaelt READ DATA/RESTORE G.W. Seefried 1983, das auf die hier verwendeten Adressen angepasst wurde

```
  40 REM
ZX81B000,ZX810000,HRGMS270,
3DWEB100,3ZU20000,4ZU30000,
...
```

*** In Zeile 40 stehen hintereinander insgesamt 251 Bildnamen ***

```
  50 REM ** SEQUENTIELLE
ANZEIGE DER BILDER **
  60 LET HRG=24064
  70 REM ** RAND USR 16556 =
RESTORE
  80 RAND USR 16556
  90 REM ** P= BILDNUMMER
 100 FOR P=1 TO 251
 110 CLS
 120 REM ** A$=BILDNAME=8
ZEICHEN
 130 REM LET A$="          "
 140 REM ** RAND USR 16564 =
READ DATA AUS ZEILE 40
 150 RAND USR 16564
 160 REM ** DER BILDNAME
WIRD IN ZEILE 10 GE-POKED
 170 REM ** R= JEDES
EINZELNE DER 8 ZEICHEN
 180 FOR R=1TO 8
 190 POKE 16527+R,CODE A$(R)
 200 NEXT R
 210 REM ** BILD WIRD MIT
GOSUB IN BANK 3 GELADEN
 220 GOSUB 10
 225 IF INKEY$ ="0"THEN GOTO
2000
 230 REM ** BILD WIRD
ANGEZEIGT
 240 PRINT USR HRG,ON
 250 PRINT USR HRG,BANK,3
 260 REM PRINT USR HRG,TEXT
 280 NEXT P
 290 PRINT USR HRG,OFF
 295 GOTO 2000
 300 STOP
 310 REM ** ** ** ** ** **
** ** ** ** **
 400 REM ** ** ** ** ** **
** ** ** ** **
 410 REM ** ZUFALLSGENERATOR
FUER BILDER 1 BIS 251
 420 REM LET A$="  "
 430 LET HRG=24064
 440 REM ** RAND USR 16556 =
RESTORE
 450 RAND USR 16556
 460 REM ** P= BILDNUMMER**
 470 RAND 0
 480 LET P=1+INT (RND *251)
 485 IF P=49 OR P=99 OR
P=177 OR P=199 THEN GOTO 480
 486 REM PRINT P
 487 REM STOP
 490 LET S=16618+9*P
 494 REM PRINT P
 495 REM STOP
 500 POKE 16621,INT (S/256)
 510 POKE 16620,S-256*INT
(S/256)
 520 REM LET A$="          "
 530 RAND USR 16564
 540 REM PRINT A$
 550 REM RETURN
 560 REM STOP
 570 REM ** DER BILDNAME
WIRD IN ZEILE 10 GE-POKED
 580 REM ** R= JEDES
EINZELNE DER 8 ZEICHEN
 590 FOR R=1TO 8
 600 POKE 16527+R,CODE A$(R)
 610 NEXT R
 620 REM STOP
 630 REM ** BILD WIRD MIT
GOSUB IN BANK 3 GELADEN
 640 GOSUB 10
 645 IF INKEY$ ="0"THEN GOTO
2000
 650 REM ** BILD WIRD
ANGEZEIGT
 660 PRINT USR HRG,ON
 670 PRINT USR HRG,BANK,3
 700 GOTO 470
 710 PRINT USR HRG,OFF
 720 STOP
 800 REM
**********************
 810 REM ** BILDNAMEN
EINGEBEN **
 820 CLS
 825 PRINT "BILD-NR. ? ";
 830 INPUT U
 835 IF U=0THEN GOTO 2000
 838 IF U=49OR U=99OR
U=177OR U=199THEN GOTO 825
 840 LET S=16618+9*U
 850 POKE 16621,INT (S/256)
 860 POKE 16620,S-256*INT
(S/256)
 870 REM LET A$="          "
 880 RAND USR 16564
 890 PRINT U;" ";A$
 900 PRINT "NEUER NAME: ";
 910 INPUT B$
 920 PRINT B$
 930 IF LEN B$=8THEN GOTO
970
 940 IF LEN B$=0THEN RETURN
 950 PRINT "BITTE EXAKT 8
ZEICHEN,          LEERSTRING
FUER ENDE"
 960 GOTO 900
 970 FOR V=1TO 8
 980 POKE 16618+9*U+V,CODE
B$(V)
 990 NEXT V
1000 GOTO 800
```

```
1100 REM
**********************
1110 REM ** BLAETTERN **
1120 PRINT "AB WELCHER BILD-
NR. ? ";
1130 INPUT P
1140 IF P=49 OR P=99 OR
P=177 OR P=199 OR P>255 OR
P<1THEN GOTO 1120
1150 CLS
1160 LET S=16618+9*P
1170 POKE 16621,INT (S/256)
1180 POKE 16620,S-256*INT
(S/256)
1190 REM LET A$="          "
1200 RAND USR 16564
1210 FOR R=1TO 8
1220 POKE 16527+R,CODE A$(R)
1230 NEXT R
1240 GOSUB 10
1250 IF INKEY$ ="0"THEN GOTO
2000
1260 PRINT USR HRG,ON
1270 PRINT USR HRG,BANK,3
1280 LET B$=INKEY$
1290 LET P=P-1+2*(W$="3")
1300 LET P=P+(P=0)-(P=256)
1310 IF P=49 OR P=99 OR
P=177 OR P=199 THEN GOTO
1290
1320 REM PRINT P;" ";
1330 REM IF INKEY$ <>
"0"THEN GOTO 1160
1360 GOTO 1140
1370 REM
**********************
2000 LET A$="DIA-SHOW"
2001 REM ****** HAUPT-MENUE
**************
2005 LET HRG=24064
2010 CLS
2015 PRINT USR HRG,OFF
2020 PRINT "****** ZX81 HRG-
DIA-SHOW *******"
2030 PRINT "***** START MIT
GOTO 2000 ******"
2040 PRINT
"**************************
*****"
2042 PRINT ""
2045 PRINT "T. RADEMACHER
UND K.-D. JAHNKE"
2047 PRINT "
JANUAR 2012"
2049 PRINT ""
2050 PRINT "
SYSTEMANFORDERUNGEN "
2070 PRINT "ZX81 MIT 16 K
RAM  HRG-FAEHIG"
2080 PRINT "HRG-TREIBER HRG-
MS >=  VERS. 2.2"
2090 PRINT "USB-VDRIVE ALS
BILDSPEICHER"
2105 PRINT "MAXIMAL 251
BILDER"
2115 PRINT "ABBRUCH DER
EINGABE MIT "" 0"" "
2120 PRINT
"**************************
*****"
2130 PRINT "<1> ALLE BILDER
NACHEINANDER"
2140 PRINT "<2> ALLE BILDER
ZUFAELLIG"
2150 PRINT "<3> VORWAERTS
BLAETTERN "2160 PRINT "<4>
RUECKWAERTS BLAETTERN "
2170 PRINT "<5>
BILDNAMEN/DATENBANK AENDERN"
2180 PRINT "<6> ENDE "
2190 PRINT
"**************************
*****"
2195 LET W$=INKEY$
2210 IF W$="1"THEN GOTO 50
2220 IF W$="2"THEN GOTO 400
2225 IF W$="3"OR W$="4"THEN
GOTO 1100
2230 IF W$="5"THEN GOTO 800
2240 IF W$="6"THEN STOP
2260 GOTO 2195
```

.

KURZ VORGESTELLT

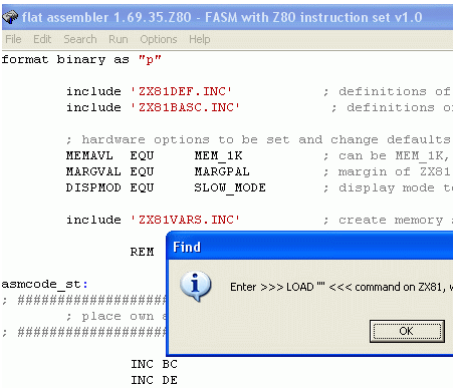
ZX81-IDE – SW-Entwicklungstool

Integrierte Entwicklungsumgebung für Zeddi-“Cross“-Entwickler

Von Karl-Heinz Dahlmann

Basierend auf dem Flatassembler wurde eine neue IDE (Entwicklungs-Umgebung) für den ZX81 geschaffen. Das Tool beherrscht neben Z80 Assembler und Zugriff auf Z80 Speicherstrukturen über entsprechende Header Files auch über eine integrierte Schnittstelle zum Laden der Programme auf einem normalen ZX81 ohne zusätzliche Hardware über die vorhandene TAPE Schnittstelle und einer ganz normalen PC Soundkarte.

Die TAPE Schnittstelle basiert auf dem ZX81 Tape Converter auf Java Basis von Simon Holdsworth. Neben dem aktuell zu assemblierendem File können beliebige andere .P-Files oder .81-Files über das Dateisystem geladen und zum ZX81 übertragen werden. Die IDE kann beliebig viele Textfiles gleichzeitig öffnen und bearbeiten und verfügt über Syntax Highlighting.



Für die Assemblerprogrammierung existiert ein Framework basierend auf REM Statements. Es ist geplant das komplette ZX Basic zu integrieren um auch reine BASIC Programmierung für den ZX81 zu ermöglichen bzw. Mischprogrammierung mit Assembler zusammen und weitere Tools für die Entwicklung von Spielen oder Programmen zu implementieren.

Die aktuelle Version kann über den Link im Forum gedownloaded werden

Karl-Heinz Dahlmann, München

HRG-SPIROGRAPH

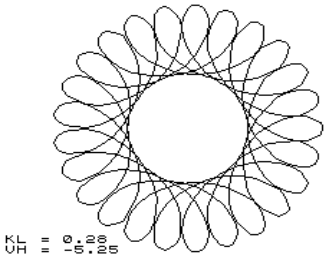
Spirograph in HRG-MS 2.4

Nachdem Matthias seine HRG immer attraktiver macht, will ich sie in den Mittelpunkt des nächsten vorzustellenden für den ZX81 umgesetzten BasiCode-Programms stellen.



Das Programm simuliert auf dem Computer ein Kinderspielzeug, den Spirographen[1]. In einem umgebenden Kreis rollt eine kleinere runde Scheibe ab. Am Rand der kleineren Scheibe läuft die Spitze eines Bleistifts mit, somit hinterläßt der Bleistift eine Spur auf dem Papier, ein symmetrisches verschlungenes Gebilde[2], das man z.B. als Mandala verwenden, also die einzelnen Felder farbig ausmalen, kann.

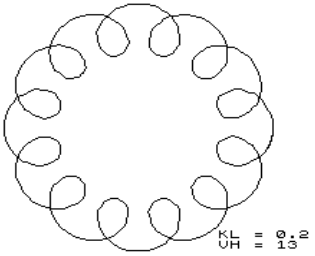
Wir laden also zunächst die HRG-64K[3], bestätigen die Abfrage mit "Y" und laden dann "SPIRMS24" nach (wie gewohnt auf Siggis Webserver zu finden) und starten mit "RUN".



Prinzipiell ließe sich das Programm in jeder der uns bekannten HRGs nutzen, das Problem ist wieder einmal der verfügbare Speicher. Außer für den Bascoder und für die (bei BasiCode auf andere Computer übertragbaren) Zeilen ab der 1000 wird Platz für Sinus- und Cosinus-Tabellen benötigt, die das Programm als erstes anlegt. Um dem Nutzer zu verdeutlichen, daß sich etwas tut und der Computer nicht hängt, habe ich hier einen Countdown eingefügt.

Ist der zur Verfügung stehende RAM in anderen HRGs zu klein, könnte man noch versuchen, statt der vorberechneten Tabellen Sinus und Cosinus "vor Ort" zu berechnen - allerdings würde dann das Programm natürlich noch langsamer arbeiten.

Nun fragt das Programm noch zwei Werte ab: KL ist das Verhältnis des Radius der kleineren Scheibe zu dem des umgebenden Kreises und kann zwischen 0 und 1 liegen. Der zweite anzugebende Wert ist VH, hier werden Werte zwischen -20 und 20 akzeptiert, wie angezeigt ohne den Bereich zwischen

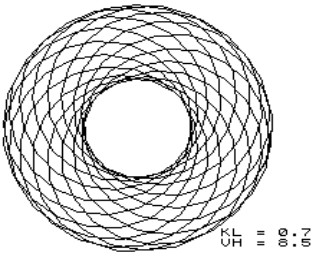


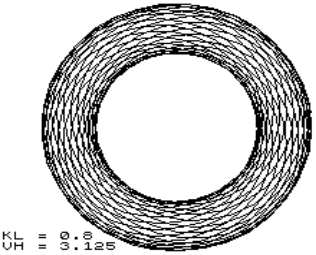
[1] von Denys Fisher erfunden und 1965 auf der Nürnberger Spielwarenmesse vorgestellt - vgl. http://de.wikipedia.org/wiki/spirograph_%28Spielzeug%29

[2] Mathematiker nennen es Hypozykloide: http://www.mathematische-basteleien.de/spirograph.htm und http://mathworld.wolfram.com/Hypocycloid.html

[3] ohne daß ich eine Ursache erkennen kann, läuft es nicht mit HRG-16K (Robson-Emulator auf DOS-PC)

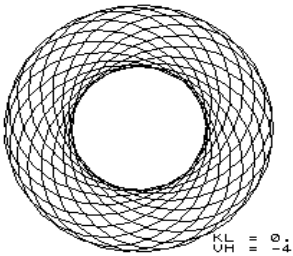
-0.1 und 0.1. Diese Zahl gibt an, nach wievielen Umläufen der Bleistift wieder auf seine eigene Spur trifft. Somit sind nicht nur ganzzahlige Werte interessant, sondern auch "glatte" Brüche dazwischen, also z.B. -3.1, 7.5, 8.625 oder -11.75. Das entstehende Bild wird dann dichter (im ersten Fall ist das Bild nicht nach drei Umläufen fertig, sondern nach dreißig), bei einem "nicht glatten" Bruchteil entsteht allmählich eine, uninteressante, komplett schwarze Fläche. Negative Zahlen bewirken den umgekehrten Drehsinn der inneren Scheibe, die Schlaufen entstehen nun außen statt innen. Am einfachsten kommt man durch Experimentieren hinter das Prinzip, hier sind zur Illustration einige Beispiele abgedruckt.



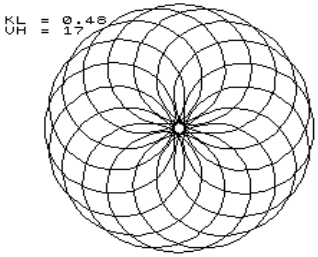


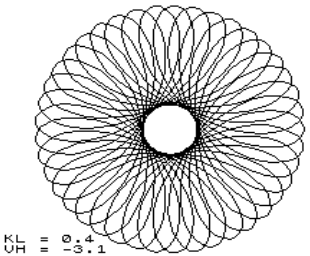
Anders als beim Lissajous-Programm[4] zeichnet das Programm nach dem Wiedererreichen des Startpunkts weiter, durch die unvermeidlichen kleinen Rundungsfehler werden die Linien nun verstärkt. Man kann mit der Leertaste abbrechen und mit "PRINT USR HRG,ON" wieder in den Grafikbildschirm zurückkehren, um beispielsweise einen Bildschirmabzug auszudrucken.

Nach "PRINT USR HRG,OFF", "CONT" und beliebigem Tastendruck kann die nächste Figur mit veränderten Parametern gezeichnet werden.




```
9010 REM  SPIRO-GRAAF IS GESCHRE
VEN DOOR P.ZEVENHOVEN
9030 REM  EERSTE PUBICATIE IN C.
V.C. INFO NUMMER 2/3
9050 REM  AANGEPAST AAN BASICODE
 3 IN JANUARI 1986
9070 REM  TROS RADIO DD 860802
9080 REM
9090 REM  ERSTAUSSTRAHLUNG: DS-K
ULTUR
9100 REM  -SPEZIAL 900711
```





Thomas Rademacher · Dezember 2008

[4] TM 1/2008 S. 22, 23

PSYCHOLOGIE AKTUELL

Neues vonne Küste

Ist Sinclairose heilbar?

Der User Willi.M im Gespräch mit Prof.Dr.Cornerpinning



Prof. Dr. Cornerpinning

Liebe Leser, liebe ZX-Teamler. Der ZX-Fachgruppe Nord ist es gelungen nach vielen Anläufen ein Interview mit dem weltweit anerkannten Verhaltensforscher und Psychoanalytiker Herrn Prof. Dr. Cornerpinning von der Kasseedorfer Paranoia-Fakultät zu führen. Wir erhofften uns Informationen über die immer weiter um sich greifende Krankheit der Sinclairose. Das leider nur sehr kurze Interview folgt nun. Ich glaube aber, die Fachgruppe Nord hat es irgendwie vermasselt....

Willi:

Herr Professor, die Sinclairose wird von Ihnen eindeutig den Neurosen zugeordnet, ist also keineswegs ein körperliches Leiden?

Prof. Cornerpinning:

Nein, auf keinen Fall! Schon die Aussage der Sinclairosiker: "Ein Leben ohne ZX ist möglich, aber sinnlos" deutet klar auf eine Zwangsstörung hin, die Sinclairosiker als in ihnen selbst liegend erkennen. Die Tiefenpsychologie in der Psychoanalyse behandelte erste Fälle dieser ZX-bedingten sozialen Ausgrenzungen schizoider Störungen als differentialdiagnostisches Kriterium bereits Anfang der dunklen 80er Jahre in England..

Willi:

Moment bitte, Ja, äh, aha, klar, natürlich, aber können Sie das für die doch überwiegend schlichten Gemüter unter den

Sinclairosikern eventuell etwas einfacher erklären, anhand von Beispielen?

Prof. Cornerpinning:

Also, ich muß Sie doch sehr bitten, die Triebgebundenheit der sozialen Norm von Sinclairosikern nicht mit den schwer triebgeschädigten an Sinclairotik Erkrankten zu verwechseln! Sie harmonierte noch nie mit dem verhaltenstheoretischen Konzept der Tendenz zum Speicherverlust. Das wird Ihnen jeder Fachmann bestätigen. Lesen Sie sich doch erst einmal in die Primärtheorie von Artur Janov ein bevor Sie mich mit Ihren sachkenntnisfreien Fragen nerven.

Willi:

Entschuldigung Herr Doktor, äh Professor, tut mir leid, ähem, das habe ich nicht gemeint, ich wollte doch einfach nur von Ihnen...

Prof. Cornerpinning:

Mir reichts! Glauben Sie unterbelichteter Kabelresteaufheber, daß es mir Freude macht Ihre abartigen Clubmitglieder zu behandeln? Individuen die nichts Besseres zu tun haben als Autobahnen zu verstopfen, Gemeinschaftsräume zu verwüsten, Landjugendheime zu ruinieren, Beziehungen zu belasten und ihr gesamtes Dasein mit dem Aufstecken von Speichermodulen zu verplempern? Die jeden technischen Fortschritt ignorieren, kleinen schwarzen Götzen huldigen und sich ihrer perversen Neigungen [redacted] ...

Anmerkung der ZX-Fachgruppe Nord:

An dieser Stelle mußten wir das Interview leider abbrechen. Ein Akademiker, ein hochangesehener Mann, er läßt uns alleine mit unseren Sorgen und Nöten. Wie krank sind wir wirklich? Besteht eine Aussicht auf Heilung? Wird Sinclairose als Berufskrankheit anerkannt? Es bleiben für uns so viele Fragen offen.....

Sorry, wir bleiben dran, Eure Fachgruppe Nord.

Im März 2012

Codename ZX83 - Der Sinclair QL



Nach den uns wohlbekannten Computern ZX80 und ZX81 und dem kommerziell überaus erfolgreichen ZX Spectrum (Codename ZX82) war der Sinclair QL die letzte große Neuentwicklung der Sinclair Research Ltd. Der interne Codename des QL war "*ZX83*". Mit dem QL begann der finanzielle Niedergang der Firma Sinclair. Obwohl der QL somit von vielen als Fehlschlag gewertet wird hat die eckige schwarze Kiste bis heute seine Anhänger.

Von Oliver

Sinclair hat den QL am 12. Januar 1984 in einer Pressekonferenz der Öffentlichkeit vorgestellt, kurz bevor Apple den Ur-Macintosh vorgestellt hat. Die Revolution des Computers weg vom 8-Bitter lag also in der Luft und Sinclair war eigentlich in einer guten Ausgangsposition. Es gab jedoch Defizite beim Grundkonzept und bei der Entwicklung der neuen Maschine, resultierend in erheblichen Lieferschwierigkeiten und Qualitätsproblemen.

Licht und Schatten

Der Sinclair QL hat einen leistungsfähigen Mikroprozessor vom Typ Motorola 68008, der intern mit 32-Bit Registern arbeitete und dessen Bruder 68000 auch in sehr viel teureren Rechnern wie dem Macintosh, Amiga und Atari ST eingesetzt wurde. Revolutionär war damals auch das recht professionelle QL-Betriebssystem "QDOS" mit echtem Multitasking und dem Ansatz zur Fenstertechnik. Die Softwareschmiede Psion steuerte serienmäßig mitgelieferte professionelle Büroanwendungssoftware bei. Nicht zuletzt ist das von Rick Dickenson entworfene elegante äußere Design des QL hervorzuheben - dazu gibt es eine spannende Webseite des ehemaligen Sinclair-Mitarbeiters unter [1].

Allerdings litt der QL auch unter Entwicklungsfehlern und fragwürdigen Kompromissen. So wurde der QL mit Microdrives ausgestattet, die in den Augen vieler Benutzer keine echte Alternative zu Disketten

darstellen. Die Video-Elektronik bremst den Prozessor bei actionreichen Spielen aus. Die Bilddarstellung am Monitor ist nicht optimal, da Sinclair den ZX83 usprünglich als transportablen Computer mit einer eigenen Flachbildröhre ausstatten wollte, man dann aber kurzfristig auf Standard-Monitore umstellen musste. Der im ROM eingebaute Basic-Interpreter ist zwar sehr gut, verzögerte aber die Entwicklung – andere Computer wie PCs oder Mac hatten keine eingebaute Sprache, sondern setzen auf ladbare Interpreter und Compiler.

Ohne den QL kein Linux?

Dennoch stürzten sich Computer-Begeisterte auf den QL - unter ihnen übrigens auch Linus Torvalds, der später das Linux-Betriebssystem für PCs aus der Taufe hob. Schnell entwickelte sich damals eine Industrie, die mit RAM-Erweiterungen, externen Diskettenlaufwerken und Prozessor-Aufrüstmodulen dem QL Beine machte. Es gab auch nach dem Abgang von Sinclair noch QL-kompatible Computer wie den Thor und die Q40/Q60, die mit schnellen 32-Bit-Microprozessoren den Macs und PCs noch eine ganze Weile Paroli bieten konnten. Allerdings blieb der QL ein Computer für Technikbegeisterte, er konnte nie die Massenmärkte der Computerspiele oder der professionellen Computeranwendungen erreichen.

Die QL Szene

Heute gibt es für QL-Fans diverse Anlaufstellen, zumeist in englischer Sprache. Da wäre die Benutzer-Vereinigung "Quanta", die seit 1984 existiert. Für Mitglieder gibt Quanta ein regelmäßiges Magazin mit sechs Ausgaben im Jahr heraus. In England gibt es auch Regionalgruppen. Bedingt durch die sinkenden Mitgliederzahlen geht Quanta derzeit durch einige Anpassungsprozesse, man war dort wohl sehr "vereinsmäßig" formal ausgerichtet und versucht nun, die Grundregeln zukunftssicherer zu gestalten [4].

Aus Duisburg kommt das regelmäßige Magazin "QL Today" [5]. Es gibt Redaktions- und Leserbeiträge zu Hard- und Software, News und Ankündigungen. In dieser Zeitschrift hat beispielsweise Tony Tebby - der Autor des QL-Betriebssystem - interessante Interna über die QL-Entwicklung bei Sinclair ausgeplaudert. Eine Archiv-DVD zu QL-Today ist seit diesem Jahr erhältlich.

Infos zum QL im Internet

Im Netz gibt es zunächst die "ql-users" Mailingliste, in der sich vorwiegend der harte Kern der QLer austauscht. Besser geeignet für den Einsteiger oder interessierten Gelegenheitsbesucher ist das hervorragende QL-Forum [2], das interessanterweise von zwei QL-Spät-einsteigern betrieben wird.

Spannend für alle an der Historie des QL interessierten ist auch die Homepage des Schweizers Urs König, der anlässlich des 25ten QL-Jubiläums eine hervorragende Präsentation zusammengestellt hat [3].

Spät- oder Wiedereinstieg?

Gebrauchte QL Computer werden im elektronischen Hafen derzeit für etwa 50-100 Euronen angeboten. Während die ersten QLs qualitativ keine gute Reputation hatten, wurde die Fertigung für den internationalen Markt später durch ein damals noch recht unbekanntes koreanisches Unternehmen namens "Samsung" übernommen. Insbesondere bei vielen dieser Samsung-QLs funktionieren die Microdrives auch heute noch. Wer Ersatzteile wie beispielsweise eine neue Tastaturfolie braucht kann unter dem Stichwort "RWAP Software" einen sehr engagierten englischen Händler finden.

Der Wieder- oder Späteinstieg in die QLerei ist im Vergleich zu anderen Homecomputern der 80er dadurch erschwert, dass der QL keinen Kassettenanschluss hat. Man kann also nicht wie bei den Zeddies einfach über die Soundkarte des PC Programme aus dem Internet übertragen. Da nahezu alle ernsthaften QLer in den späten 80ern irgendwann Diskettenlaufwerke für ihren Rechner gekauft haben geschieht der Softwareaustausch zwischen QL und PC traditionell über Disketten. Für Leute ohne QL-Diskettenlaufwerk muss daher erstmal die serielle Schnittstelle herhalten, oder man benutzt einen Emulator.

Aus heutiger Sicht ist der Sinclair QL ein interessantes Zwitterwesen aus der Zeitenwende zwischen Heim- und Personal-Computern – so etwas wie ein "Missing Link" in der Archäologie. Was ihn in jedem Fall als echten Sinclair-Computer ausweist, ist dass er sich deutlich von allen anderen Computern seiner Zeit abhebt und dass eiserne Fans bis heute daran basteln und programmieren.

Quellen und Weblinks

[1] http://www.theproductdesigners.com/archive.html

[2] http://www.qlforum.co.uk/

[3] http://www.qlvsjaguar.homepage.bluewin.ch

[4] www.quanta.org.uk

[5] http://qltoday.com/

KURZ VORGESTELLT

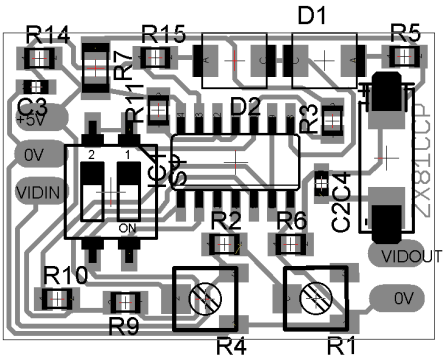
ZX81CCP – Videoausgang für ZX81

Aktive Aufbereitung des Videosignals mit Inverter und Schwarzschulter

Von Karl-Heinz Dahlmann

Die vorliegende modifizierte Schaltung wurde angelehnt an einen Vorschlag von Bodo (Original aus Chip 6/84), aber in einigen Punkten verbessert. So arbeitet sie mit beiden ULA Versionen (184 oder 210), welche unterschiedliche Signalpegel liefern und erzeugt eine Schwarzschulter sofern keine vorhanden ist. Außerdem kann man über den DIP Switch alternativ schwarz auf weiß oder weiß auf schwarz einstellen.

Das analoge Videosignal (Tri-State) wird zunächst digitalisiert und dann neu gemischt. Durch diese Technik wird das Hintergrundrauschen, welches bereits aus dem entsprechenden ULA Pin „verschmutzt“ kommt, nahezu vollständig eliminiert. Mit den Potis R1 und R4 wird der Eingangspegel auf die ULA eingestellt und erlaubt gleichzeitig eine Feinjustage für eine knackige Zeichen-Wiedergabe.

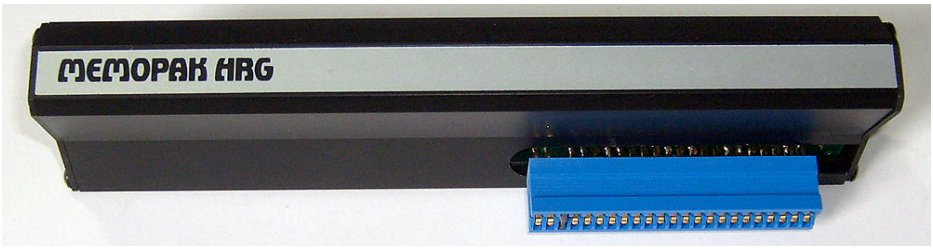


Das in SMD Technik realisierte Board mit 30 x 22 mm passtmechanisch in das Modulatorgehäuse und wird ab März 2012 für GBP 9,90 (ca. EUR 12,-) auf http://www.sellmyretro.com komplett bestückt angeboten.

Karl-Heinz Dahlmann, München

MEMOPAK-HRG

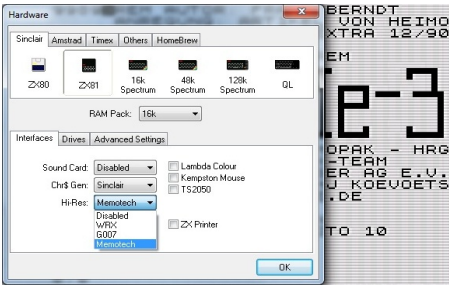
BasiCode in der MEMOPAK-HRG



Von Th. Rademacher

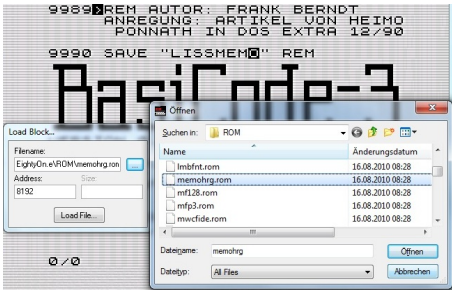
Schon längere Zeit schwebt mir vor, auch für die MEMOPAK-HRG (eine Voll-HRG im Ansteckmodul) einen Bascoder zu schreiben. Das wollte ich auf dem 2012er Mahlerts-Treffen angehen, Klaus hatte schon zugesagt, seine Hardware dafür mitzubringen.

Da die Zeit in Mahlerts immer viel zu schnell verfliegt, machte ich mich jetzt schon an vorbereitende Arbeiten. Der EightyOne-Emulator zeigte sich als gut geeignet. Im Internet fanden sich die erforderlichen Informationen zur HRG in Form einer Handbuch-PDF [1].



Man muß lediglich unter Options -> Hardware... auf "Interfaces" unter "Hi-Res:" Memotech einstellen, daß unter "RAM-Pack" mindestens 16k angewählt werden müssen, versteht sich von selbst.

Außerdem muß virtuell das HRG-Kästchen angesteckt werden - unter File -> Load Memory Block wird auf Adresse 8192 das 2K-File memohrg.rom aus dem Verzeichnis ROM geladen.

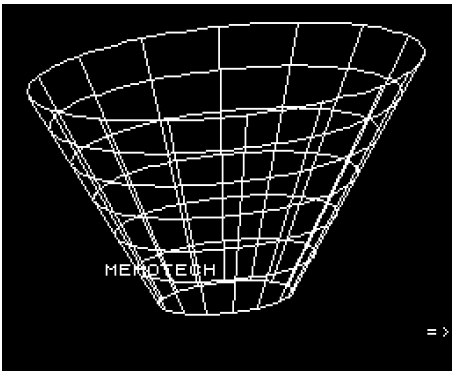


Stattdessen kann auch das ZX81-ROM hinten um diese zwei Kilobyte

verlängert werden, so stehen (nach Anwahl dieses modifizierten ROMs) die Grafikroutinen sofort zur Verfügung.

Die Lösung von MEMOTECH bietet den Vorteil, daß man die Grafikdaten überall im RAM ansiedeln kann, es muß nicht einmal eine "glatte" Adresse sein. Um die RAMTOP-Klimmzüge zu umgehen, entschied ich mich für die Lösung, die auch von S.Schmidts HRG-Toolkit verwendet wird: eine REM-Zeile am Programmanfang. Eine Überprüfung mit der im Handbuch beschriebenen Selbsttestfunktion zeigte, daß es funktioniert.

Die Grafik-Routinen zum Punkt-Setzen und -Löschen, zum Linie-Ziehen und -Löschen und zum Text-Schreiben und -Löschen und natürlich zum Initialisieren sowie zum Ein- und Auschalten der HRG (mehr ist für BasiCode nicht erforderlich) waren schnell angepaßt.



Hier ist die Gelegenheit, das BasiCode-Prinzip am Beispiel des Zeichnens einer Linie zu erläutern. Im Team Magazin 5/2007 wurden ab Seite 24 die BasiCode-Grafikroutinen erläutert. Schauen wir in das Programm LISSMEMO hinein. Dort wird zunächst ein Rahmen um die Zeichenfläche gezogen, die obere Kante in diesen Zeilen:

```
1100 GOSUB 600
1101 LET HO=0
1102 LET VE=0
1103 LET CN=1
1104 GOSUB 620
1105 LET CN=0
1110 LET HO=.9999
1111 LET VE=0
1112 GOSUB 630
```

In Zeile 1100 wird in den (bereits initialisierten) Grafikmodus umgeschaltet und in den Zeilen 1101 bis 1104 der (unsichtbare) Startpunkt der Linie gesetzt.

BasiCode braucht drei Argumente für die LINETO-Funktion: die horizontale und vertikale Koordinate des Zielpunkts (HO und VE, Werte von [einschließlich] 0 bis [ausschließlich] 1, sozusagen Prozent-Angaben) und die Zeichenfarbe (CN - bei CN=0 wird die Linie in Vordergrundfarbe gezeichnet und bei CN=1 in Hintergrundfarbe, das heißt gelöscht). Die Ausführung wird mit GOSUB 630 ausgelöst.

Die MEMOPAK-HRG verwendet folgende Syntax:

```
LET X=hor1
LET Y=ver1
LET P=hor0
LET Q=ver0
```

```
LET Z$="LINE"
RAND USR 8192
```

Hier sind die Koordinatenvariablen ganze Zahlen.

Schauen wir uns also an, was in LISSMEMO ab Zeile 630 passiert. Zunächst werden ungültige Koordinatenangaben durch ein schlichtes RETURN (das in Zeile 649 steht) abgewehrt.

```
630 IF HO<0 OR HO>=1 OR VE<0 OR VE>=1 THEN GOTO 649
```

Dann werden X und Y (verbindliche Variablennamen der MEMOPAK-HRG) aus HO und VE ermittelt.

```
631 LET X=HO*HG
632 LET Y=VG-VE*VG
```

Hierbei werden HG und VG (verbindliche Variablennamen für BasiCode) verwendet. Diese waren am Anfang des Bascoders zugewiesen worden:

```
22 LET HG=247
23 LET VG=185
```

Sie geben die Größe der Zeichenfläche an. In der MEMOPAK-HRG können horizontal 248 Punkte dargestellt werden, BasiCode verwendet ein Breite/Höhe-Verhältnis von exakt 4:3, daraus ergibt sich 186 für die Höhe. HG und VG sind jeweils um eins niedriger als diese Zahlen, weil die Zählung bei 0 beginnt.

Warum ist Y nicht einfach gleich VE*VG? Weil in der MEMOPAK-HRG der Punkt 0,0 in der linken unteren Ecke liegt, aber bei BasiCode in der linken oberen Ecke.

Woher kommen die Koordinaten des Startpunktes? Sie wurden nach Ausführung des vorangegangenen Punkt- oder Linienbefehls zugewiesen:

```
640 LET P=X
642 LET Q=Y
```

Zusammengefaßt ist also der Bascoder (die Zeilen bis 999 eines BasiCode-Programms) die Schnittstelle zwischen dem (auf andere Computer übertragbaren) Programm in den Zeilen ab 1000 und dem speziellen Computer (in diesem Fall dem ZX81 mit der HRG-Hardwareerweiterung): die Festlegungen des BasiCode-Protokolls werden mit den Gegebenheiten der Hard- und Software des konkret verwendeten Computers in Einklang gebracht.

Interessant ist, daß in der MEMOPAK-HRG Linienstücke rückwärts, also vom Zielpunkt zum Startpunkt, gezogen werden. Ein kleiner Nachteil dieser HRG-Lösung zeigte sich: ergeben die Pixel zufällig das Byte h76, wird hier die Pixelzeile abgebrochen. Zum Glück tritt dieser Fall nicht häufig ein, die Chance ist halt 1 zu 256.

In Mahlerts wollen Klaus und ich die Programme (wie gewohnt, auf Siggis

Server zu finden) auf der echten Hardware testen, unterm EightyOne laufen sie jedenfalls schon mal.

Th. Rademacher / Februar 2012

Quellen und Weblinks

[1]http://www.zx81stuff.org.uk/zx81/generated/hardwareinfo/m/MemopakHRG.html

HISTORISCHE WERBUNG



EPROM mit USB-Treiber für ZX81 64K RAM

PIO-USB im ZX81-Puristen-Design – Um den Treiber stets verfügbar zu haben residiert er am besten in einem nichtflüchtigen Speicher

Von K.-D. Jahnke

Das Treffen in Mahlerts von 2009 hatte es in sich. Zeitlose Stunden mit unserem Lieblings-Computer, nette Leute, toller ZX-Flohmarkt und sehr, sehr kurze Nächte. Da bekam man nur wenig von dem schönen Wetter in der Rhön mit.

Seit dem Jahr 2008 durfte ich auch an der ZX81-USB-Revolution teilhaben, die von Thomas Lienhard in Gang gesetzt wurde und die ihren Höhepunkt 2009 noch nicht erreicht hatte. Die Anzahl der ZX81-USB-Begeisterten nahm noch zu wie man an den vielen VDRIVE-Bestellungen bei Jens Sommerfeld sieht. Für mich ist der ZX-USB-Drive ein Quantensprung an Speicher- und Bedienkomfort. Er passt perfekt zusammen mit der Entwicklung der neuen HRG-Treiber durch Matthias S. Es ist jetzt relativ leicht möglich, nicht nur die HRG-Programme selbst, sondern auch die Grafik-Speicherabbilder auf USB-Stick zu bannen und in Sekundenschnelle wieder aufzurufen.

Mein altes USB-System

Okay, so alt ist es ja noch gar nicht. Bei allen Vorteilen hatte mein erstes USB-System von 2008 (siehe ZTM 3, 2008) doch ein kleines Manko: Das Laden des USB-Treibers entweder per Kassette oder per ZX81LOAD aus dem PC dauerte ziemlich lang, bevor man den USB-Stick benutzen konnte. Thomas Lienhard sagte mir schon 2008 voraus, dass ich bald ein Instant-USB-System per EPROM oder batteriegepuffertem RAM würde haben wollen, um das lange Laden des Treibers loszuwerden. Er hatte natürlich Recht und ich hatte großes Glück, dass er dieses Jahr nach Mahlerts kam.

ZX81-Hardware ist schön

Die äußere Form des ZX81 ist für mich genauso wichtig wie seine innere Schönheit und Leistungsfähigkeit. Ich zähle mich zu den „moderaten“ ZX81-Puristen. Das soll heißen, ich möchte die Original-Innen- und Außen-Architektur des ZX81 möglichst wenig verbasteln oder zerstören. Veränderungen an der Hardware und am Gehäuse sollten moderat, reversibel und möglichst unsichtbar sein. Daher überlege ich immer sehr lange, bevor ich irgendwo ein Loch bohre oder eine Leiterbahn

durchtrenne, und suche immer zuerst nach einer zerstörungsfreien und reversiblen Lösung. Immerhin sind unsere edlen ZX-Maschinen und die Memotech-Zusatzteile echte Raritäten und Oldtimer, mit denen man vorsichtig umgehen muss, weil sie unwiederbringlich sind.

Startpunkt Memotech-HRG-Baustein

Da mich die ZX81-HRG fasziniert, habe ich mir natürlich auch den Memotech-HRG-Baustein besorgt, den man zwischen den Zeddy und das RAM-Modul steckt. Es hat ein internes EPROM mit dem Memotech-HRG-Treiber, der den Speicherberich zwischen 8k und 10k belegt. Das Bauteil funktionierte klaglos, es gab jedoch nur sehr wenige Memotech-HRG-Programme dafür (Dank Henning Räder für das Finden dieser Raritäten!) und war im Bedienkomfort und in der Geschwindigkeit dem HRG-Treiber von Matthias (HRG-MS v2.5) deutlich unterlegen. In meinem einfachen und spontanen Gemüt fragte ich Thomas L. per E-Mail, ob man nicht einfach das Memotech-HRG-EPROM gegen eines mit USB-Treiber austauschen könne. Nach Analyse der Beschreibung und der Fotos riet Thomas davon ab und später in Mahlerts zu einer anderen, besseren Lösung.

RÖM-ische USB-Philosophie in Mahlerts

In Mahlerts erklärte mir Thomas, dass es wie so oft im Leben mehrere Wege nach ROM oder besser gesagt nach EP-ROM gibt. Eine elegante Lösung sind wiederbeschreibbare batteriegepufferte RAMs sogenannte Uhren-Chips (z.B. DS1643 mit 8K RAM), die eine eingebaute, etwa 10 Jahre haltbare Lithiumbatterie und eine interne Uhr haben. Dazu hat er noch adressierbares und beschreibbares statisches RAM (z.B. 8K), das durch die interne Stromversorgung auch nach Abschalten des Computers seine Information behält. Die Beschreibung des RAMS mit einem ZX-Treiberprogramm ist einfacher als die eines echten EPROMS und geschieht mit einer Art BLOAD-Befehl. Sobald die interne Batterie im Chip jedoch verbraucht ist, sind allerdings alle Informationen weg. Anders ist das mit einem EPROM (Erasable Programmable ROM), das mit einer besonderen Apparatur (dem Brenner) und besonderer Software beschrieben werden muss und seine Information auch ohne Batteriestromversorgung sicher behält. Löschen kann man das EPROM über ein Sichtfenster mit UV-Licht, danach ist es wiederbeschreibbar. Diese Lösung wurde für meinen Zeddy ausgesucht, da die Platine relativ einfach war und vor Ort in Mahlerts alle Bauteile und Hilfsmittel vorhanden waren.

Teileliste

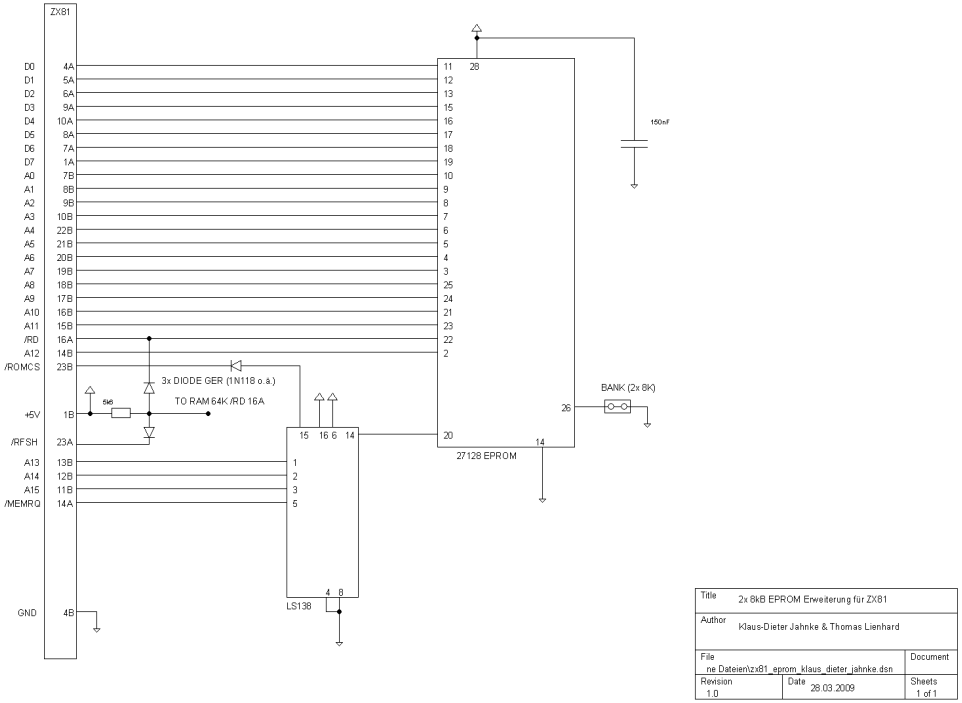
Lochplatine
23 polige ZX-Buchse mit sehr langen Lötkontakten
23 poliger ZX-Stecker (mit Kerbe)

16KB EPROM : GI 27C128-15 (28 polig) mit Sockel
Adressdecoder SN74LS138N (16 polig) mit Sockel
1x Germaniumdiode (1N118 o.ä)
1x Kondensator 150 nF

Für die HRG-fähigkeit
2x Germaniumdioden (1N118 o.ä)
1x 5.6 Kiloohm Widerstand

USB-Treiber: ***Klaus3*** V1.0 (siehe ZXTM Nr. 3, 2008)

Der Schaltplan



Den Schaltplan hatte Thomas schon nach kurzer Zeit „freihändig“ aufgezeichnet und die Teile waren auch schnell zusammengesucht. Ich hatte sogar eine genau passende leere Loch-Platine für das Memotech-HRG-Gehäuse und fing an, nach Thomas´ Plänen zu Löten, allerdings

ging das Ganze bei mir so langsam voran und ich machte so viele Fehler, dass ich es bald aufgab und Thomas wie ein Weltmeister weiterlötete und zunächst ohne EPROM die Stromversorgung (5V) und die Durchgängigkeit der Kontakte prüfte.

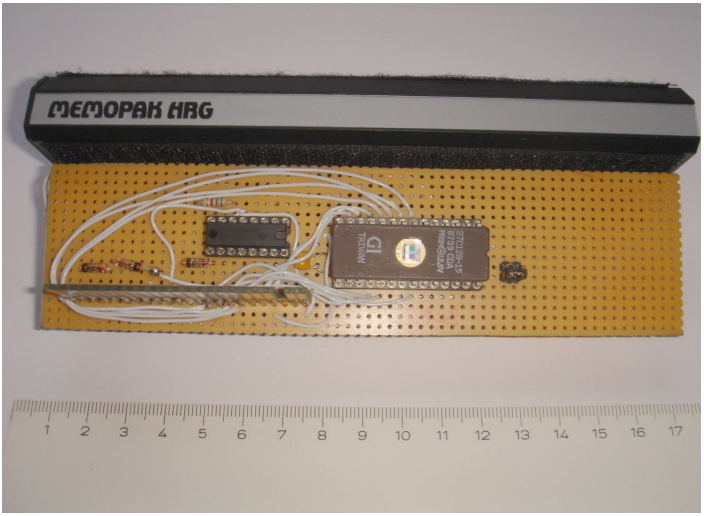
Die HRG-Fähigkeit des RAM wurde mit zwei zusätzlichen Germanium-Dioden und dem 5.6 kOhm-Widerstand hergestellt. Der von Joachim beschriebene zweite Widerstand (5-20 Kiloohm) vom RAM-Pack auf 5 V ist nicht nötig und wurde weggelassen (siehe ZXTM 01/1991). Der große Augenblick des EPROM-Brennens kam am Samstag erst nach Mitternacht. Und siehe da, das Teil funktionierte fast auf Anhieb. Jubel, Jubel und bravo Thomas!

Zu guter letzt passte die Platine auch noch in das Memotech-Gehäuse hinein und ließ sich mit Zeddy und 64K-RAM und meinem PIO-USB-Baustein zusammenstecken. Der Aufruf des USB-Treibers erfolgte wie gewohnt mit PRINT USR 8192, “I“ oder jetzt noch einfacher mit PRINT USR 8888, “I“ .

ZX-Herz, was willst Du mehr? ZX einfach einschalten und zack mit dem USB-Stick loslegen, bequemer geht es fast nicht. Die fertige USB-EPROM-Platine passt in das Memotech-HRG-Gehäuse. Das 16KB EPROM hat 2 Bänke zu je 8KB, die durch den Jumper wählbar sind.



Thomas L. macht den Schaltplan für die USB-EPROM-Platine



Die fertige USB-EPROM-Platine passt in das Memotech-HRG-Gehäuse. Das 16KB EPROM hat 2 Bänke zu je 8KB, auswählbar mittels Jumper.

Besonderheiten der Speicherbelegung

Die HRG-fähige EPROM-Platine wird zwischen ZX und RAM-Modul gesteckt. Zum korrekten Betrieb muss im 64K-RAM der Speicherbereich von 8k bis 16k ausgeschaltet werden. Das entspricht der Schalterstellung D im Memotech-RAM-Baustein (Schalterstellung Off,Off, Off,On, siehe Memotech-Beschreibung). Dieser Bereich wird von der EPROM-Platine beansprucht. Außerdem ist noch zu beachten, dass der USB-Treiber in den oberen Speicherbereich von 65280 bis 65535 (Hexadezimal: FF00 bis FFFF) Daten ablegt, die für den Zugriff auf den USB-Stick nötig sind. Das kollidiert leider mit mit der HRG-Bildspeicherung mit dem Grafiktreiber HRG-MS v2.5 von Matthias S. in Bank 7 (Adressen 59360 bis 65504). Diese Grafikbank ist zwar für ein vorher vom USB-Stick geladenes HRG-MS-Programm benutzbar, aber sobald ein USB-Zugriff erfolgt, wird die Grafik in Bank 7 zerstört. Der Versuch, diese Grafik mit dem BSAVE-Befehl auf den Stick zu speichern führt auch zum Gesamtabsturz des Grafikprogramms. Um Konflikte zu vermeiden, sollte man vorsichtshalber Grafik-Bank 7 nicht verwenden. Es bleiben ja noch drei weitere Bildspeicher für das 64K-RAM übrig. Das sind die Grafikbänke 4, 5 und 6. Die beigefügte Tabelle faßt die Möglichkeiten und die Adressdaten zusammen.

Speicherbelegung zur Bildspeicherung mit HRG-MS v2.5 und USB-EPROM-Treiber

HRG-MS Speicher-bereich	HRG-MS Treiber V2.5	Dezimal-adressbereich (Ende: Anfangsadresse + 6143 Byte)	Hexadezimal-adressbereich	Anmerkung/ Verwendung mit USB-EPROM möglich
Bank 3	Nur 16KRAM*	26592 – 32735	67E0h – 7FDFh	Ja
Bank 4	16KRAM* 64KRAM	34784 – 40927	87E0h –9FDFh	Ja
Bank 5	16KRAM* 64KRAM	42976 – 49119	A7E0h –6FDFh	Ja
Bank 6	16KRAM* 64KRAM	51168 – 57311	C7E0h –DFDFh	Ja
Bank 7	16KRAM* 64KRAM	59360 – 65503	E7E0h -FFDFh	Nein **

*) Der 16K-Treiber ist auch auf 64K RAM lauffähig, beansprucht aber Speicherplatz im BASIC-Bereich, so dass nur kleinere BASIC-Programme (max. 8416 Byte) möglich sind.
**) USB-Systemvariablen im Adressbereich 65280 bis 65535 (Hexadezimal: FF00 bis FFFF)

Danksagung

Ich danke Thomas L. für sein Know-How, seinen Großeinsatz bei der Herstellung der EPROM-Platine und die Überlassung der ICs. Dank auch an Jens Sommerfeld für die Stiftung des EPROM Sockels und ***last not least*** Stefan für den 5.6 kOhm-Widerstand. ☺

ZXKlaus Bielefeld,
30.03.2009/23.02.2012

Literatur und Weblinks

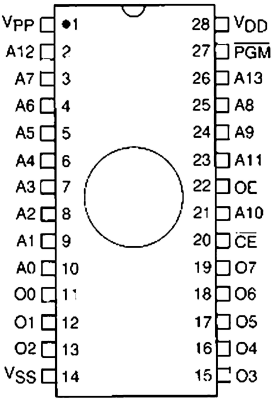
Produktbeschreibung Memotech 64K-RAM (aus ZX-Team CD)

HIRES-Graphik auf jedem ZX81, ZXTM Nr.1, 1991

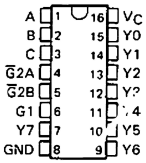
ZX81 mit 64k RAM und PIO-USB, ZXTM Nr. 3, 2008

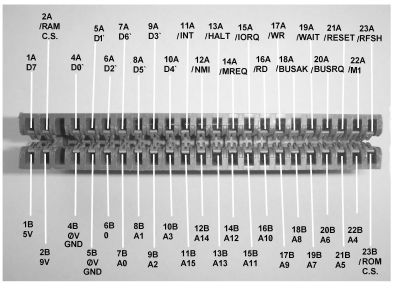
Manual HRG-MS v2.5 siehe www.swatosch.de/zx81.html

27128 EPROM



LS138





Oben

ZX81

Unten

USB Anschluss mit dem VDIP1 MODUL

Neben Elektronik und Software stellt sich für den USB-Interface-Selbstbau auch die Frage nach der angemessenen mechanischen Integration - statt eines kompletten VDRIVE2 wurde ein VDIP1 Einbaumodul verwendet.

Von Ulrich

Wie man einen USB-Stick am Zeddy betreibt, hatte uns Thomas aus der Schweiz unter Verwendung eines VDRIVE2 schon auf dem 12ten Treffen vorgeführt. Nun hat Siggi die Software weiter perfektioniert und im Forum vorgestellt. Diese Komfortsteigerung hat mich motiviert, den USB-Anschluss in meiner Erweiterungsbox fest zu installieren. Die PIO für die Real Time Clock hatte noch den Port B frei und sollte so eine kurze Leitungsführung zum USB Modul ermöglichen.

Meine bisherige Lösung, externes VDRIVE2 über D-SUB-Stecker an der PIO angeschlossen, wollte ich aber dafür nicht zerlegen. Bei der Suche nach einem Lieferanten für ein USB-Modul wurde ich bei Farnell [1] fündig.

Das dort gezeigte VDIP1 Modul schien für mein Vorhaben geeigneter zu sein. Als Privatperson habe ich das Modul bei HBE [2] online bestellen können, zusätzlich, um auf den Mindestbestellwert zu kommen, habe ich noch ein paar Batterien mitbestellt. Drei Tage später (Vorauszahlung), erhielt ich dann die Ware per UPS.

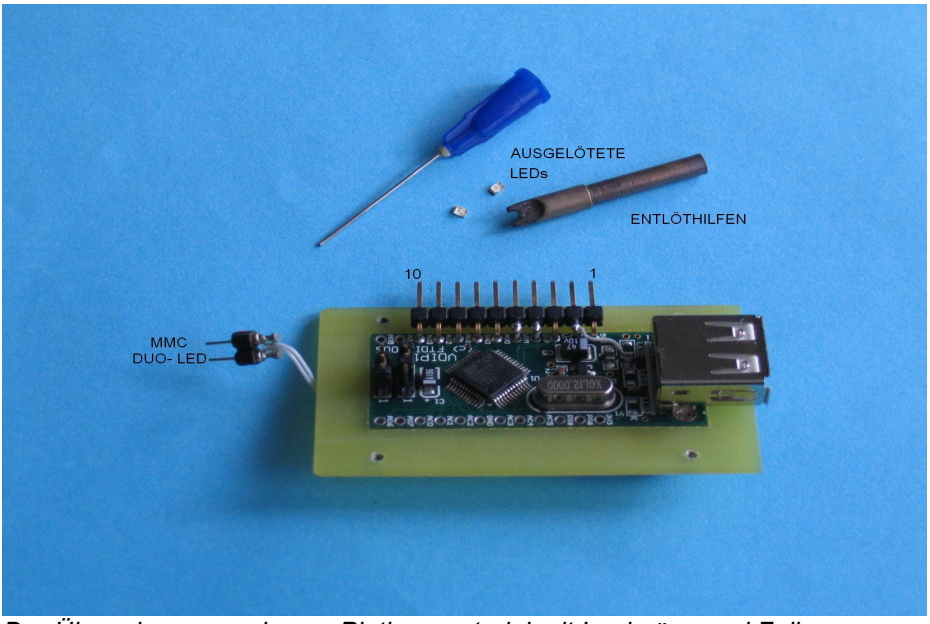
Den Frontplattendurchbruch für den USB-Stecker wollte ich aus optischen und praktischen Erwägungen über dem Schlitz für die MMCard “Wechselplatte D“ anordnen. Auch der Unterbau der MMC-Steckverbinderplatine bot sich gleichermaßen für eine Befestigung an, doch die Umsetzung dieser Vorstellung, insbesondere eine stabile Befestigung, waren eine Herausforderung.

Modifikation des Moduls

Um die Bauhöhe zu reduzieren, mußten die Sockelstifte ausgelötet werden. Gut geschützt gegen statische Aufladung, mit Armkettchen geerdet, habe ich zuerst die Verbindungsstege der Kunststoffummantelung mit einer Minibohrmaschine mittels Diamantscheibe durchtrennt. Die an den Stiften verbliebenen Isolierteile wurden vorsichtig mit dem Seitenschneider entfernt. Den Lötkolben an

der Unterseite des Kontaktes angesetzt, wurde dann mit Hilfe einer stumpf geschliffenen Injektionsnadel (Durchmesser 0.6mm) von der Oberseite der Platine Stift für Stift aus der Durchkontaktierung herausgeschoben. Mittels Lötsauglitze, unter Zugabe von Flussmittel (z.B. Löthonig, mit Isopropanol verdünnt) wurde das überschüssige Lot entfernt. Anschließend wurde die Platine unter Verwendung eines kurzborstigen Pinsels mit Isopropanol gereinigt.

Die beiden grün leuchtenden SMD-LEDs wurden von der Platine abgelötet, denn dort wären sie nach dem Einbau dem Blick entzogen. Eine spezielle Lötspitze, u-förmig ausgefeilt, erleichtert das Ablöten von SMD Bauteilen.

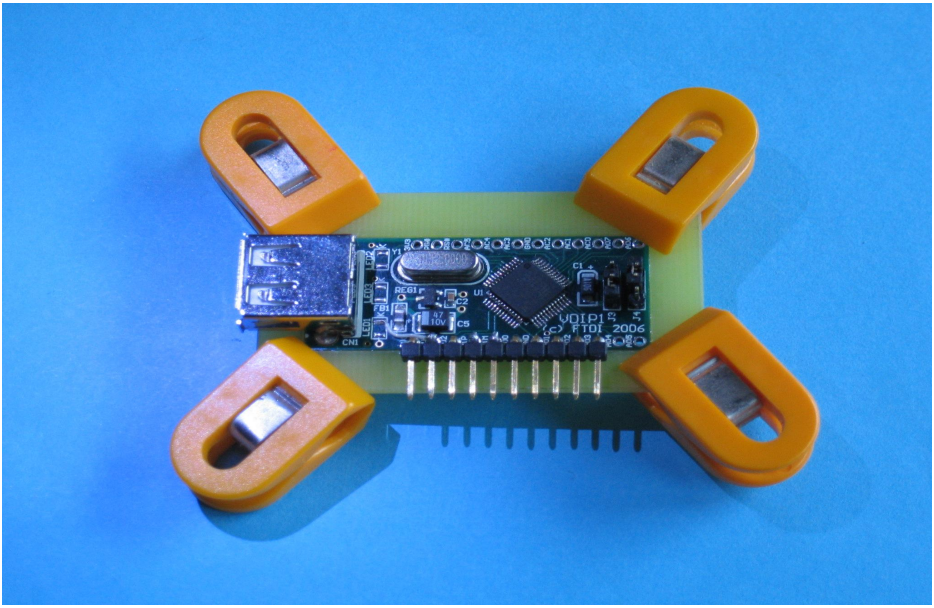


Der Überrahmen wurde aus Platinenmaterial mit Laubsäge und Feile bearbeitet und anschließend mit UHUplus mit dem Modul verklebt.

Durch das Auslöten der Sockelstifte konnte nun das Modul ca. 7mm tiefer montiert werden. Um es aber irgendwie befestigen zu können, kam mir die Idee mit dem “Passepartout“. Dieser Überrahmen ist aus Platinenmaterial gefertigt. Der Ausschnitt wurde leicht unsymmetrisch positioniert, so dass sich der USB-Steckerausschnitt dann mittig in der Frontplatte über dem Schlitz der MMC “Wechselplatte D“ befindet.

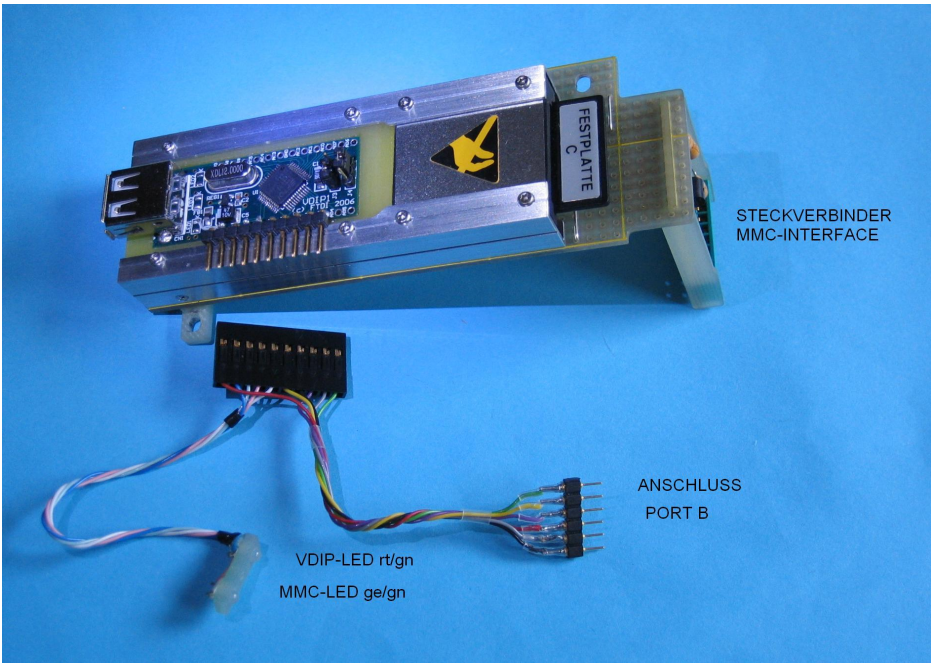
Die Steckplätze für die MMCs sind auf einer Lochrasterunterkonstruktion verschraubt. Diese sogenannte “2. Etage“ ist im Urzustand in der Bildergalerie als Anhang [3] zum Team-Magazin-Beitrag Ausgabe 100 zu sehen. Zur mechanischen Verbindung mit der “2. Etage“ wurden zwei 85mm lange Abschnitte eines ALU-Rechteckrohres (12.5 x 7.5 x 1mm, aus dem Baumarkt) derart bearbeitet, dass mit den daraus entstandenen U-Profilen nunmehr sowohl der Überrahmen als auch die Steckaufnahme für “Laufwerk C“ MMCard verschraubt werden konnten.

Für die MMC Ansteuerung der Laufwerke “C“ und “D“ waren bereits Bohrungen für die LEDs in der Frontplatte vorhanden. Was lag daher näher, diese LEDs durch (bipolare) DUO-LEDs zu ersetzen?



Nun leuchtet die untere grün für “D“und gelb für “C“. Die darüber liegende LED signalisiert die VDIP1 Ansteuerung in den Farben rot/grün, wie schon vom VDRIVE2 gewohnt. Alle Anschlüsse führen über einen abgewinkelten 10pol. Steckverbinder, der anstelle der ursprünglichen Stifte 1 - 10 in die VDIP1-Platine eingelötet wurde.

Für die jetzt bipolare LED an Stift 2 und 3 ist nur Stift 3 mit der Platine verlötet. Stift 2 (gekürzt), ist über ein kurzes Drähtchen mit dem K-Anschluß an LED1 verbunden. Der Platinenaufdruck und der Vergleich der Datenblätter VDIP1 / VDRIVE2 waren hilfreich für diese Änderung. Die Stifte 4 und 5, ebenfalls gekürzt, haben somit auch keinen Kontakt zur Platine. Mit isolierten Drähten verlötet, führen sie, durch die Durchkontaktierungen gefädelt, zu einer 2-pol. Buchse auf der MMC-Lochrasterplatine und stellen damit die Verbindung zur Ansteuerung der DUO-LED ge/gn her. Die übrigen Anschlüsse führen nun auf kürzestem Wege zur PIO und ersparen somit den bisher erforderlichen Buffer (s. Forum, ZX-Team Magazin, Bericht: “VDRIVE2 Nachlese“).





Der Einbau im Gehäuse

Von der Funktion ist das VDIP1 mit dem VRIVE2 identisch, auch das Aufspielen der Firmware 3.62 erfolgt nach gleicher Prozedur.

Das Ergebnis ist nun eine kompakte Massenspeichereinheit, die meinen eingangs aufgeführten Vorstellungen voll entspricht.

Danke den Wegbereitern Thomas, Joachim und Siggi.

Ulrich

Quellen und Weblinks

[1]http://de.farnell.com/ftdi/vdip1/entwicklungsmodul-f-vnc1l/dp/1329313

[2]http://www.hbe-shop.de/

[3]http://www.zx-team.de./zx-team/100/bilder.html (Bild 10-12)

KURZ VORGESTELLT

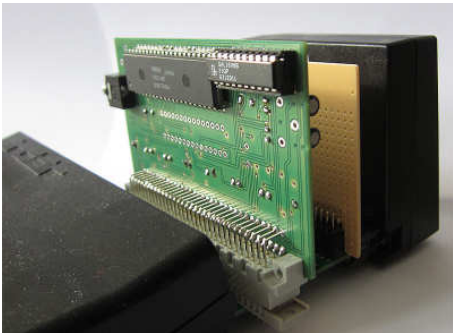
Hardware Entwickler Turbo ZX81XT

Port-Erweiterung und Anschluss an das ZX96-Steckersystem mit dem Bus-Extender

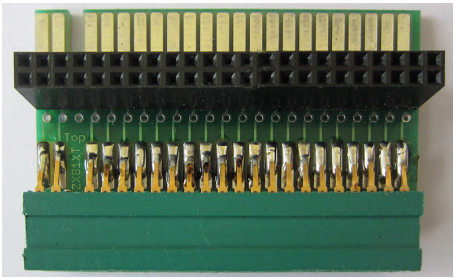
Von Karl-Heinz Dahlmann

Der Mangel an Schnittstellen des ZX81 veranlaßte mich zu der nachfolgenden Entwicklung. Schließlich hat der kleine schwarze Kasten nur eine einzige Verbindung in die Außenwelt, den ZX-Bus. Wie soll man dort Hardware adaptieren, wenn man z.B. auf einen zusätzlichen Sinclair Memory Pack angewiesen ist oder auch gleichzeitig das beliebte ZXpand benutzen möchte ?

Das brachte mich zu folgendem Miniboard als Extender für den ZX-Bus.



Dabei wird ein beliebiges (neues) Entwicklungsboard ähnlich wie eine PCI Karte im PC in den schwarzen Slot eingesteckt und beliebige weitere Hardware hinten aufgesteckt.



Alternativ lässt sich aufgrund der Belegung auch ein VG64 Steckverbinder benutzen (A/C Reihe), der kompatibel ist zu dem ZX96 Bus und auch entsprechende Komponenten aufnehmen kann. Das Board lässt sich beliebig kaskadieren, im Bild unten ist ein MrX (ZonX) Sound Board zu sehen mit VG64 Stecker, ein nacktes Entwicklungsboard und der 16k Memory Block.

Das Board wird über http://www.sellmyretro.com für GBP 7,00 (ca. EUR 8,50) inklusive ZX-Stecker angeboten.

Karl-Heinz Dahlmann, München

TTL-USB-Zeddies selbst gelötet

Ein Erfahrungsbericht mit der einfach-Schaltung für das VDrive-Modul

Von K.-D. Jahnke

Noch nie war es so leicht einen ZX81 mit einer USB-Schnittstelle auszurüsten und damit tatsächlich in die unendlichen Dimensionen des ZX81 vorzustoßen. Das Editieren, Speichern, Laden und die Übertragung von Daten und Programmen mit dem ZX81 und das Übertragen von ZX81-Dateien auf jeden PC sind fast zum Kinderspiel geworden. Dies war möglich durch die Idee von Thomas Lienhard, das IT High-Tech-Bauteil VDRIVE2 an den ZX81 als USB-Schnittstelle anzubinden (ZXTM 05/2007). Für mich war es der Beginn einer neuen Zeddy-Beziehung, der Beginn endloser Stunden schierer Begeisterung für die alte Sinclair Technik. Ja, für die alte Technik, mit der ich mich auseinandersetzen musste, um das High-Tech-Bauteil unterzubringen. Es gibt tatsächlich mehrere Möglichkeiten einen USB-Zeddy herzustellen, mit interner Verbastelung und auch ohne jede innere Veränderung des ZX81, durch ein externes USB-Ansteckmodul. Die zahlreichen zum Thema ZX81-USB erschienen Artikel sind in Literatur und Referenzen aufgeführt.



TTL-USB-VDrive – A very BASIC and very nice solution

Ich möchte mich hier auf die Version des TTL-USB-Drives beschränken, wie sie von Olli Lange erdacht und realisiert wurde. Ganz im Sinne von Clive Sinclair ist der TTL-USB-Vdrive eine echte „Sparversion“ und „very basic“, weil er (neben dem High-Tech-Vdrive2) als Peripherie-elektronik nur Standard-TTL-ICs (TTL = Transistor-Transistor-Logic) im Wert von ein paar Cent benötigt und die Verwendung einer Z80-PIO oder programmierte GAL ICs überflüssig macht.

Die TTL-USB-Vdrive-Grundversion erfordert nur die wenigen nachfolgend abgebildeten Teile sowie einen modifizierten USB-Treiber der sich wahlweise in einem 16K-RAM Zeddy über RAMTOP (ab Adresse 16514) installiert oder in einem 56k-Zeddy in den 8-10k Adressbereich, (ab Startadresse 8192) , falls dieser RAM-Bereich vorhanden ist. Der USB-Treiber wird konventionell über Kassette oder als WAV-Datei über die Soundkarte aus einem PC (oder MP3-

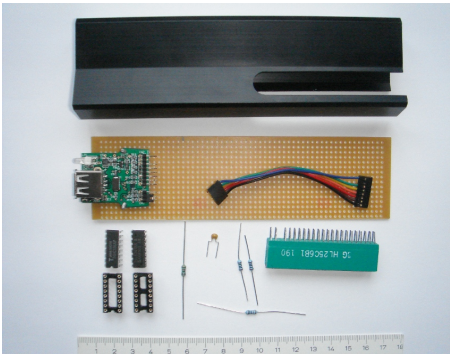
Player) per LOAD-Befehl geladen.

Der Luxus lockt – batteriegestützer 56kRAM mit permanentem USB-Treiber

Nach der „Very-Basic“-Version mit nur 16kRAM wird man bald anspruchsvoller und möchte mehr RAM und einen „permanenten“ (nicht flüchtigen) USB-Treiber. Auch hierfür gibt eine Lösung von Oliver Lange: Die Kombination eines batteriegestützten 56kRAMs mit dem TTL-USB-VDrive (Oliver Lange im ZX-Team Forum 06.01.2011). Diese Variante wurde ZX- intern und auch ZX-extern realisiert. Der USB-Treiber wird nach Adresse 8192 geladen und bleibt im batteriegestützten RAM erhalten und kann jederzeit mit dem USR-Befehl aufgerufen werden. Richtig sicher und robust wird der batteriegestützte RAM aber erst durch den Zusatz einer Schutzschaltung, für den 8-12k-Adressbereich. Jetzt ist der USB-Treiber vollkommen geschützt und kann auch nicht durch einen versehentlichen Zeddy-Absturz gelöscht werden.

Intern oder extern, das ist die Frage

Da ich mich zu den „gemäßigten“ ZX81-Puristen zähle, die möglichst wenig am Original-Zeddy verbasteln wollen, habe ich meine erste TTL-USB-Platine tatsächlich als externe Platine realisiert. Mein Ziel war es, die Platine in einem kleinen Memotech Gehäuse extern unterzubringen, so, wie es für das Memotech-Drucker Interface oder das HRG-Modul verwendet wird. Das Layout auf einer Rasterplatine und die Anordnung der Bauteile hatte ich vorher ausführlich mit dem Lochmaster PC-Programm simuliert, bevor ich anfing zu löten, damit alles auch später in das kleine Gehäuse hineinpasste.



Externe TTL-USB Sparversion vor dem Zusammenbau (K.-D. Jahnke)

Das Lochmaster-Programm habe ich bei Joachim Merkl 2010 in Mahlerts gesehen und gleich danach gekauft. Als elektronischer Anfänger kam ich mit diesem Programm auf Anhieb gut zurecht. Die einfachste TTL-USB-Platine lief tatsächlich sofort und war eine große Motivation für mich mutiger zu werden und weiter zu löten. Der Verlust des USB-Treibers jedes mal nach jedem Abschalten des ZX81 war aber etwas lästig und einen EPROM-Brenner habe ich nicht. Ein batteriegestützter RAM mit einem Adressbereich von 8-16k in dem MC-Programme laufen wäre die

ideale Lösung für mich. Als ich Oliver Lange meine Wunschträume mitteilte, zauberte er nach kurzer Bedenkzeit seine genial modifizierte Version des batteriegestützten 32K-Willi-RAMs aus dem Hut, den er auch noch auf Hutgröße 56K RAM erweiterte. Genau DAS war die Lösung für meine Zwecke.

ZX-Tabubruch – Der interner USB-Umbau mit internem 56K RAM

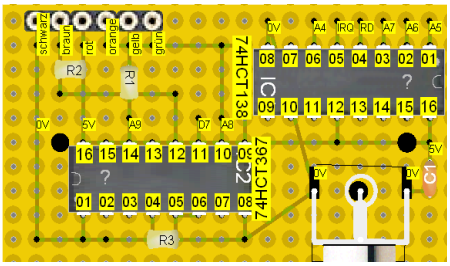
Alles, was ich mit meinen Zeddies seit ihrer Wiederentdeckung im Jahre 2001 realisieren konnte, habe ich vom ZX-Team abgeschaut und mit seiner Hilfe löttechnisch realisiert. So auch der interne Einbau des Vdrives in das enge Zeddy-Gehäuse an die Stelle des TV-Modulators. Als ich das bei Thomas Lienhard (2010) zum ersten Mal sah, wollte ich es „so ähnlich“ auch haben.



Dazu musste ich nur meinen Sensortasten-Super-Zeddy innerlich noch etwas mehr verbasteln und noch etwas mehr Mut zum Löten haben. Mit dem Kabellabyrinth in meinem TTL-USB-56K-BATT-RAM-Super-Zeddy werde ich sicher keinen Schönheitswettbewerb gewinnen, OK, aber die Kiste läuft und sie ist tatsächlich die robusteste und benutzerfreundlichste ZX81-Maschine geworden, den ich je hatte.

Wie viele TTL-USB-Zeddies gibt es?

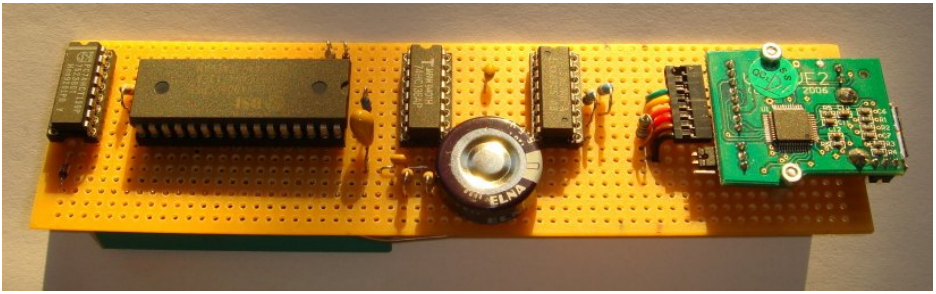
Nach meinem gegenwärtigen Kenntnisstand gibt es mindestens fünf (5) TTL-USB-Zeddies: Das Original auf externer Platine mit Tastatur-Puffer von Oliver Lange, eine externe Version von Jens Sommerfeld, meine externe Version, meine interne Version, sowie eine externe Version von Stefan, die er im ZX81Forum beschreibt.



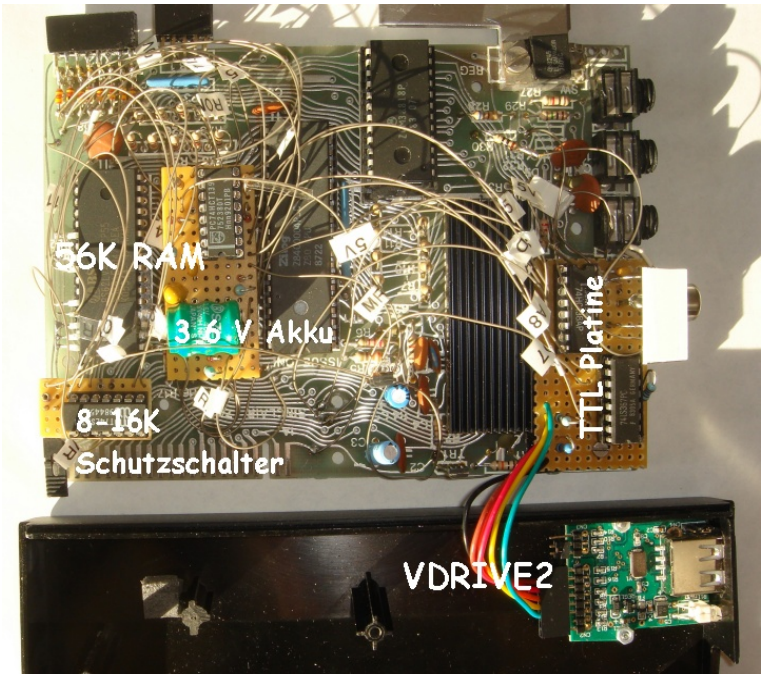
Lochmaster-Layout der internen TTL-USB Sparversion (K.-D. Jahnke)

Danksagung

Oliver Lange möchte ich für die Weiterentwicklung des klassischen Willi-RAMs mit TTL-USB-Vdrive und seine fernelektronischen Diagnosen meiner Lötversuche per E-Mail danken. Auch Dank an Joachim Merkl der mich auf das Lochmaster-PC-Programm brachte.

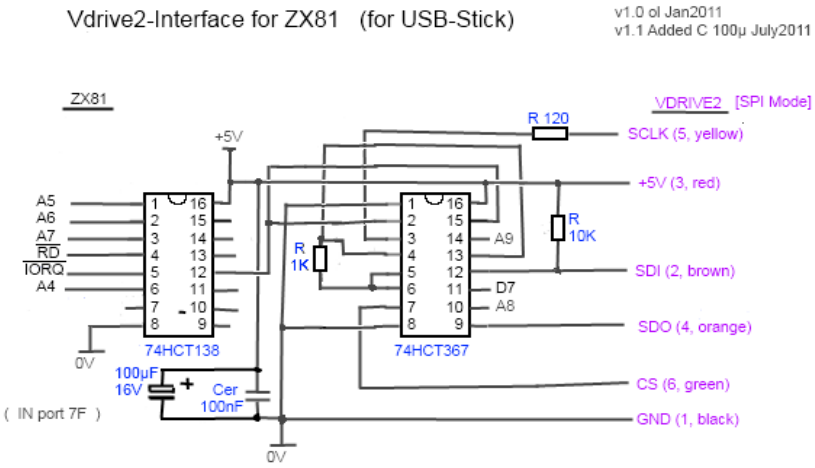


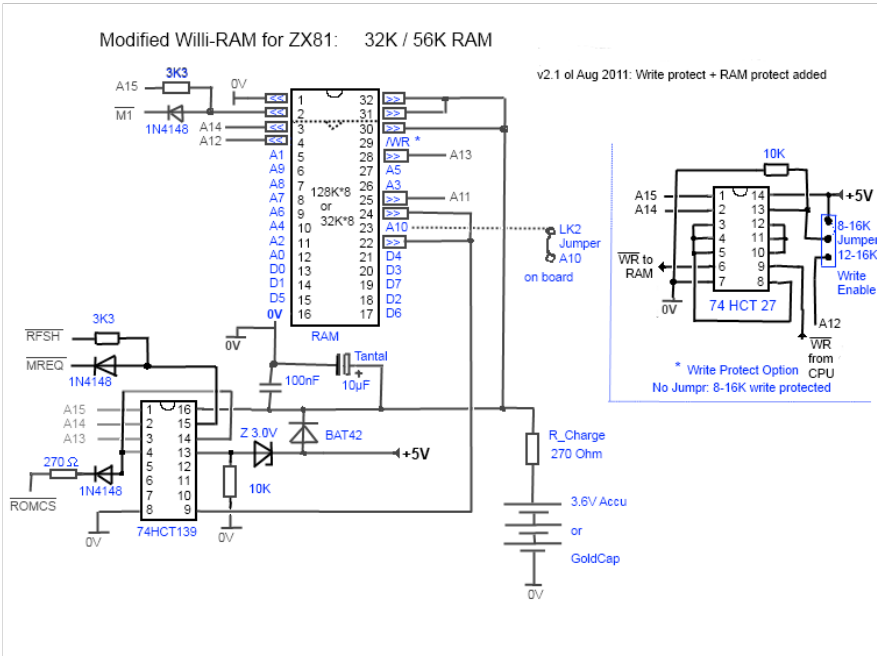
Externe TTL-USB Sparversion mit batterie/kondensator gestütztem 56K RAM, ohne Schutzschaltung für den 8-16 K Adressbereich (K.-D. Jahnke).



Interne Externe TTL-USB-Sparversion mit akkugestütztem 56K RAM und RAM-Schutzschaltung für den 8-16 K Adressbereich (K.-D. Jahnke).

Schaltpläne von O. Lange







Der USB-Zeddy von Jens im Einsatz

Literatur und Referenzen (Auswahl)

T. Lienhard: USB-Speichersticks am ZX81. ZX-Team Magazin 05/2007
T. Lienhard: Das USB-Interface und seine Folgen. ZX-Team Magazin 02/2008
O. Lange. USB-VDrive-Interface und interner HRG-RAM – Die Sparversion. 06.01.2011
In: ZX-Team Forum (http://forum.tlienhard.com)
W. Mannertz. Willi-RAM für den ZX81. ZXTM 1/1991, 1/1992, 5/1992, 3/1993, 6/1995.
U. Strohmeyer: Vdrive2 Nachlese. 23.08.2010. In: ZX-Team Forum (http://forum.tlienhard.com)
T. Lienhard: PIO-Ersatz für Vdrive2-Anschluss mit GAL. 16.03.2010. In: ZX-Team Forum (http://forum.tlienhard.com)
T. Lienhard: ZX81 Speichererweiterung mit GAL . 14.03.2010. In: ZX-Team Forum (http://forum.tlienhard.com)

..

MINI-ARCADE

ZX81-Mini-Arcade – ein Baubericht

Der Zeddi - reif gemacht für die Spielhalle

Von Jens

Hallo User! Gute Ideen kommen oft über Nacht – bei mir war es tatsächlich ziemlich genau um Mitternacht (nein, ich bin nicht beim Bildaufhängen auf dem Toilettensitz ausgerutscht – es geht hier ja auch nicht um den Flux-Kompensator).

Das genaue Datum weiß ich zwar nicht mehr, es muss aber so im August 2011 gewesen sein. Ich saß in meinem Hobbykeller an meinem ZX81 und spielte ein Game mit meinem selbstgebauten Joystick, der allerdings ab und zu etwas hakelig ist. Da ich noch einen professionellen Arcade-Joystick herumliegen hatte, beschloss ich, diesen an den ZX81 anzuschließen (steuerbar über die Tasten 5/6/7/8 und 0). Und dann blitze es im Hirn: warum nicht einen eigenen Arcade-Automaten bauen?! Natürlich transportabel und für den Küchentisch!! Kurz im Kopf die Bauteile gecheckt:

ZX81-Bastelplatine … *check*

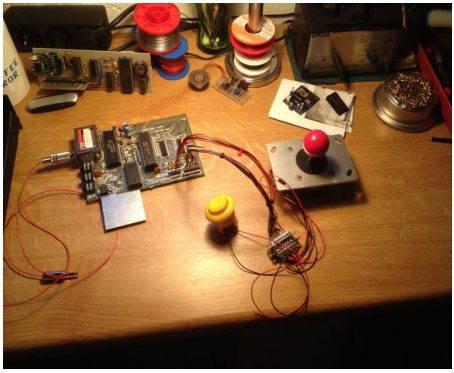
Joystick... *check*

Feuerknopf... *check* (aus einem alten Joystick ausgebaut)

Mini-Fernseher... muss ich noch besorgen (ich bekam gleich zwei – einen von Klaus auf dem letzten Minitreffen bei Olli in Bremen und einen von Ingo - Schulfreund)

Gehäuse... äh – später.

Ich fing an zu basteln (erstes Ziel: Joystick und Feuerknopf). Die Platine ist eine ISSUE ONE mit gesockelten IC´s – die Tastaturanschlüsse waren nicht mehr vorhanden – ideale Voraussetzungen. Also den Lötkolben zur Hand genommen und diesen Testaufbau gelötet (der dann sicher auch verbaut wird – sieht ja keiner):



Der Testaufbau – ZX81-Bastelplatine mit Joystick und Feuerknopf

Das hat also schon mal geklappt. Die Tastatur wird dann parallel dazu angeschlossen (die wird ja nur zum Starten des Games gebraucht – wenn überhaupt). Und hier kam die nächste Idee (ist auf dem Minitreffen entstanden, als Paul mir das ZXpand

näher erklärte): Warum baue ich nicht das ZXpand mit dem ZXpand-AY-Soundmodul ein?! Das lässt sich über den von „kmurta“ entwickelten ZXpand-Browser super steuern und die Programme einfach starten. Quasi nach dem AAP-System („Anschalten, Aussuchen und Play“). Das ist einfacher, als über Kassette zu laden. Dazu müsste allerdings die Browser-Steuerung angepasst werden, damit das über den Joystick gesteuert werden kann (das ist bereits geschehen). Ich hatte auch zuerst an das TTL-USB-Olli-IF gedacht – aber das ZXpand lässt sich quasi ohne Tasten (nur SHIFT muss beim Anschalten gedrückt gehalten werden) starten. Also entschieden...

Das Gehäuse sollte einem echten Arcade-Automaten ähnlich sehen – es gibt schon einige Tisch-Varianten (zB für das iPad) – diese sollen hier als Inspiration dienen. Hier zwei Beispiele:



Mini-Cabinet-Holz (billig)



iPad-Arcade (99,- Euro)

Somit bekommt die Sache schon eine Richtung. Das Gehäuse wird natürlich aus Holz gebaut. Eine schöne Lackierung wäre auch toll... mit ZX81-Schriftzug natürlich. Die „echten“ Arcade-Automaten haben fast alle den Joystick LINKS und den Feuerknopf RECHTS. Zum Ballern ist das sicher besser. Also ist das auch entschieden.Zuerst müssen die Grundmaße festgelegt werden. Der Fernseher wird so eingebaut, dass nur der Bildschirm zu sehen ist – die Bauteile sind dann von hinten über

eine kleine Klappe zugänglich (falls das Bild nachjustiert werden muss usw.). Der ZX81 kommt oben in die sogenannte „Markise“ - die SHIFT-Taste wird durch einen Schalter realisiert, der nach den Anschalten der Stromversorgung wieder umgelegt wird. Außerdem brauchen wir noch Lautsprecher.

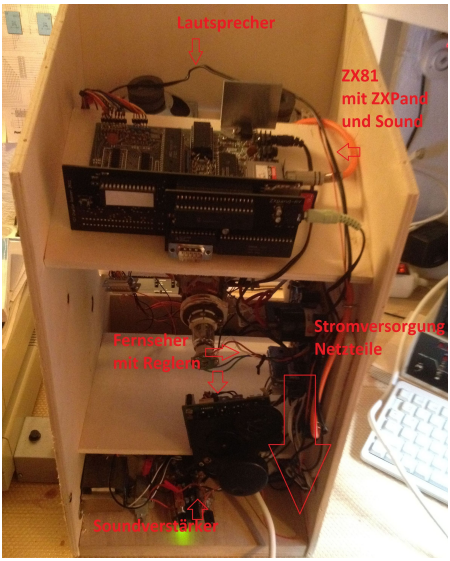
Als nächstes wurde der kleine Fernseher „entkernt“. Mit Gehäuse wäre das zu groß geworden. Um die Gefahr eines Stromstoßes geringer zu halten (was beim Fernseher ja Hochspannung bedeutet – das kann wehtun) wurde der Fernseher zuerst eingebaut. Der erste Schritt (Gehäuse abbauen) war nicht einfach, hat aber geklappt. Das Gehäuse ist inzwischen im Rohbau und sieht jetzt so aus:



Mein Sohn Pau und sein Schulfreund Lennard haben mir beim Zeichnen, Sägen, Feilen, Schrauben und überhaupt beim Gehäuse geholfen. Eigentlich brauchte ich kaum etwas tun.

Inzwischen ist der Fernseher (das war nicht so einfach) eingebaut, ebenso der ZX81 plus ZXPand und ZXPand-AY-Soundmodul. Die Netzteile habe ich im unteren Teil platziert. Zuerst dachte ich auch an einen Münzeinwurf, der das Spiel startet. Das muss ich aber auf später verschieben (ich möchte erst mal das Grundgerät fertig bauen)...

Den Joystick und Feuerknopf stecke ich ebenfalls in eine Holzplatte.



Anschließend – wenn alles läuft – kümmere ich mich noch um die

Lackierung. Das Design wird an Arcarde-Automaten angelehnt sein und natürlich auch den ZX81 wieder spiegeln (der steckt schließlich drin).



Die Front (wie ich sage „Das Zyklopen-Auge“ - kommt ein Zyklop zum Augearzt) des Fernsehers wird auch noch verblendet. Damit sollte das Gerät dann auch schick aussehen!

Da das ZXPand ja durch eine Erweiterung über Sound verfügt, besorgte ich mir für 6,99 Euro ein einfaches Paar Computer-Lautsprecher – fix und fertig. Diese wurden dann aus dem Gehäuse ausgebaut und in das Gehäuse gesteckt...

Zuletzt kam dann noch die Klebefolie (matt/schwarz), welche ich leider nicht sehr professionell anbringen konnte – zumindest sieht es an manchen Stellen doch sehr nach Pfusch aus – aber es ist ja „nur“ ein Prototyp ;-)

Und nun das endgültige Gerät, welches auf dem Treffen 2012 spielbereit zu sehen sein wird – natürlich wie es sich gehört mit ZX81-Schriftzug:



Jens (jens@zx81.de)

Das ZXPAND am ZX80, die Story

Mit dem ZXPAND hat SirMorris eine SD-Karten Anbindung für den ZX81 inklusive 32K RAM heraus gebracht. Kann das auch am ZX80 funktionieren?

Von Paul

Das Modul hat ein gepatchtes ROM, das interne ROM des ZX81 wird abgeschaltet und schon kann man mit den SAVE und LOAD Befehlen auf die SD-Karte zugreifen als hätte der ZX81 nie etwas anderes gemacht. Das klappt so gut das ich dachte, geht das nicht auch mit dem ZX80?

Der Anfang

Nachdem ich mich mit SirMorris kurzgeschlossen hatte war ich mir seiner Unterstützung sicher und fing mit den grundsätzlichen Überlegungen an. Der Erweiterungsbus des ZX80 unterscheidet sich von dem des ZX81 in einem wesentlichen Punkt: Er hat kein /ROMCS. Dies kann nachgerüstet werden, aber wer einen original ZX80 hat möchte auf dessen Platine keine unwiderruflichen Veränderungen durchführen. Da das ZXPAND ROM beim Start aktiv ist liefert es Daten auf dem Datenbus die mit den Daten des ZX80-ROM kollidieren. Daher habe ich für einen ersten Versuch das ZXPAND ROM entfernt.

Weitere Schwierigkeiten bereitet das 32K RAM dessen Adressbereich sich mit dem ROM Schatten überschneidet. Daher darf dieses nicht angesprochen werden, was aber der ZX80 normalerweise auch nicht tun würde. Mit der Konfiguration RAM von 16-48 K startet der ZX80 dann auch. Das eingebaute und gesockelte 4K ROM des ZX80 habe ich gegen ein 8K EEProm getauscht.

EightyOne

Zwischenzeitlich hatte SirMorris eine EightyOne Version mit ZXPAND Unterstützung erweitert. Diese startet dann statt des ZX81 ROM das ZX81ZXPAND ROM und somit sind die Erweiterungen verfügbar. Die ZXPAND Hardware hat er in eine DLL gepackt so das diese leicht austauschbar ist. Da EightyOne in C programmiert ist und die ZXPAND Firmware ebenso, war das Erstellen dieser DLL (nach seiner Aussage) sehr einfach.

Nach winzigen Änderungen des EightyOne für mich akzeptiert dieser die DLL jetzt auch im ZX80 Modus so das ich mit ersten Programmierungen beginnen konnte. Dazu nahm ich den Quelltext des ZX80 ROM von G. Wearmouth und TASM.

Das ZX80-ROM

SirMorris hatte die Idee die Funktionstabelle des ZX80 ROM zu erweitern da man dort Funktionen hinzufügen kann die nicht im sonstigen Syntaxcheck enthalten sein müssen (wie TL$, PEEK...)

Allerdings funktioniert dies nur mit Funktionen und nach kurzer Abwägung wurde dieser Schritt von uns verworfen. Wir wollten LOAD und SAVE wie im ZX81 verwenden können.

Ich gab SirMorris den Hinweis auf die Tabelle der Befehle und schon hatte SirMorris eine Kopie davon in Bereich 4-8K angelegt sowie den Verweis darauf umgelegt (wir verwenden in diesem Zusammenhang einen gemeinsamen Dropbox-Ordner). Dies war notwendig weil das 4K ROM voll war und die Sprungtabelle nur 12Bit breit war, also nur innerhalb der 4K Sprünge erlaubte. Die Kopie hat er dann natürlich 16 Bit breit gemacht. Da der Einsprung selbst 16 Bit breit war brauchten wir im unteren Bereich (die original 4K des Basic) nur wenige Bytes zu ändern.

Der erste Befehl

Meine erste richtige Arbeit bestand darin den Befehl CAT hinzuzufügen und aufzurufen. Die freie Stelle in der Tokentabelle die „Z“ entspricht gesucht und die Einsprungpunkte über Labels erweitert. Das war ja einfach. Das Problem fing jetzt aber erst an. Wie hole ich den String, wie finde ich heraus ob der leer ist? Wie zeige ich etwas auf dem Bildschirm an und wie gehe ich in das Basic zurück zum nächsten Befehl? Wie lese ich ein Zeichen von der Tastatur?

Für die eigentlichen ZXPAND Befehle benutzt SirMorris kleine Makros. Wie diese aufgerufen wurden zeigte mir das disassemblieren des ZX81-ZXPAND ROMs. Ich habe die komplette ZX81-CAT-Routine kopiert und so lange angepasst bis ich die gleichen Dinge an die DLL schickte. Ohne EightyOne mit seinen Debug-Funktionen wäre ich dort niemals angekommen.

Nach einigen Versuchen bekam ich manchmal etwas auf den Bildschirm. Oft mit einem Absturz danach. Ich musste irgendwo Pufferspeicher nutzen, der ZX81 nutzt hier den 33 Byte Druckerpuffer. Diverse Versuche mit festen RAM-Adressen (nur für erste Tests) oder einem tiefer gelegten Stack erfolgten. Derzeit sind wir um den Puffer herum gekommen. Debuggen brachte dann folgendes ans Licht: Der String den ich an ZXPAND schickte war im ZX80 anders als im ZX81. Auf jeden Fall musste ich die Ausgabe des ZXPAND auf den ZX81 Zeichensatz anpassen (Sonderzeichen), denn der ist nicht wirklich identisch zum ZX81 Zeichensatz. Hinzu kamen Stack Probleme: Hatte ich den Stack an der einen Stelle der

Routine noch korrekt, war er nach der nächsten Subroutine zerstört. Ich musste erstmal herausfinden was man besser mit CALL und was mit JP aufruft, insbesondere bei ROM-Routinen die das ROM selbst mal so und mal so aufruft. Sehr hilfreich war das Buch „Mastering Machine Code on Your ZX81“von Toni Baker. Z.B. die Print Routine konnte ich direkt übernehmen. ROM Routinen die im ZX80 fehlten, ZXPAND im ZX81 aber nutzte, habe ich so nachgebildet.

Das Verhalten des ZXPAND musste an einigen Stellen angepasst werden. So sind die Default Adressen für Laden und Speichern anders, die Default-Extension ist nicht „.P“ sondern „.O“. Das einfachste war es dafür eigene Aufrufe im ZXPAND zu implementieren was SirMorris gerne tat. Ich hatte bereits das Character-Mapping in den ROM-Routinen des ZX80 bei der Kommunikation des ZXPAND angepasst, aber da SirMorris dies auch im ZXPAND implementierte habe ich dieses wieder aus dem ROM entfernt. Er hat dann auch noch einen Jumper im GAL anders abgefragt, so dass man damit das Verhalten des ZXPAND steuern kann. Ist der Jumper1 gesetzt wird das ROM disabled und das RAM fest auf 16-32K begrenzt. Dadurch ist es nicht mehr notwendig das ROM des ZXPAND zu entnehmen und vorher das korrekte RAM-Setting zu aktivieren.

Man kann also festhalten das die ZX80-Umsetzung ohne die Hilfe von SirMorris gar nicht realisierbar gewesen wäre und er großen Anteil daran hatte. Ich bin ihm sehr dankbar das er sich immer wieder Zeit genommen hat Details zu klären und Anpassungen für mich zu machen.

Hier eine Übersicht über die Änderungen im ROM Bereich 0-4K

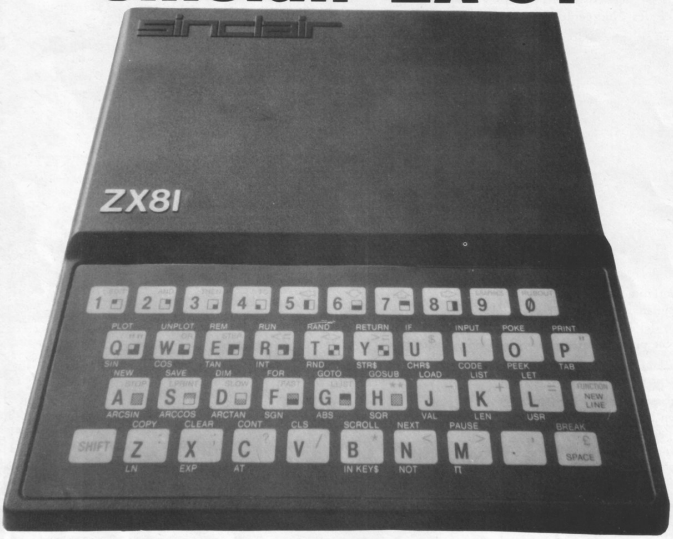
Adresse	Original	Neu	Funktion
0000h	JP 0261h	JP 1410h	Statt RAMFILL jetzt eigene Startroutine inkl. Ramfill und ZXPAND Initialisierung. Hier kann ggf. Autostart nachgerüstet werden
033Ah	LD HL, 006Bh	LD HL, 0FFFh	Statt alter Key Table jetzt neue verwenden
05A9h	LD HL, 00BAh	LD HL, 104Eh	Statt alter Token Table jetzt neue verwenden
079Ch	DEFW 084Ah	DEFW 0849Ah	Leeres REM korrigiert
07E8h	CALL 001Ah	JP 10DCh	Neue Routine die Zwei Byte Offsets benutzt

Über die genaueren Vorgänge, wie Befehle im oberen Bereich des ZX80 ROM hinzugefügt wurden, berichte ich in einer anderen Geschichte namens

„Die Erweiterung des ZX80 ROM“ **Gruß Paul**

HISTORISCHE WERBUNG

Sinclair ZX 81



Ihr Einstieg in die Mikroelektronik

Der ZX 81 kann, was Sie von ihm verlangen. Am Anfang hilft er zu lernen, wie ein Computer arbeitet. Grundlage dabei ist die Computersprache BASIC. Das Lernen geht recht einfach, denn Sie erhalten in der Grundausstattung alles, was Sie zunächst brauchen. Die Packung enthält den handlichen Rechner, ein Netzteil sowie Verbindungskabel zum Cassettenrecorder (Klinkenstecker 3,5 mm) und eines zum Fernsehgerät.

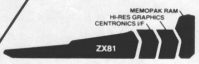
Weiterhin gehören zum Lieferumfang eine deutsche Betriebsanweisung mit Erklärung der BASIC-Befehle sowie eine Originalanleitung in Englisch.

Der ZX 81 kennt natürlich die vier Grundrechenarten Addieren, Subtrahieren, Multiplizieren und Dividieren, dazu logarithmische und trigonometrische Funktion. Er kennt die e-Funktion und Zusammensetzungen davon. Zahlen werden in Fließkomma dargestellt, auf 9,5 Dezimalstellen genau. Der Rechner kennt logische Überknüpfungen wie AND, OR, NOT; und vergleichende Operationen mit größer, kleiner, gleich kann er auch abarbeiten. Daß sich Unterprogramme aufrufen und Schleifen ausführen lassen, versteht sich nach diesen Ausführungen fast von selbst.

Daten des ZX 81

Kunststoffgehäuse:	16,7 cm x 17,5 cm x 4 cm
Gewicht:	350 g
Rechenwerk:	vier Chips, darunter ein Z 80 A Mikroprozessor
Speicher:	8 kByte ROM (BASIC), 1 kByte RAM (durch Steckmodule erweiterbar)
Anzeige:	mit 24 Zeilen zu je 32 Zeichen auf jedem handelsüblichen Fernsehgerät
Eingabe-Prüfung:	automatischer Syntax-Check jeder Eingabe-Zeile
Zahlen:	± 3 x 10^{-39} bis ± 7 x 10^{38} mit einer Genauigkeit von 9 1/2 Dezimalstellen; Fließkomma-Darstellung
Grafik:	Eingabe von 20 grafischen und 54 negativ (investiert) darstellbaren Zeichen über Tastatur, mit Zusatz erweiterbar auf hochauflösende Grafik

Sinclair ZX 81 jetzt mit großer Zubehörpalette:



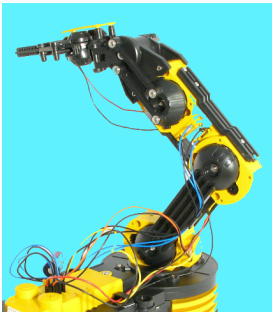
Speichermodule für 16, 32 und 64 K Ram. Hochauflösende Grafik. Centronics-Parallel-Schnittstelle für Profi-Drucker.

ROBOTERARM

„Zwei-Finger-Joe“

Roboterarm am ZX 81 *(1)*

Ein Zusatzgerät lässt den Zeddy handgreiflich werden – die ersten Schritte dieses Projekts waren in Mahlerts zu bewundern



Von Jens

Schon als Kind träumte ich davon, dass ein Roboter mir im Haushalt helfen würde – zum Beispiel beim Müll runtertragen oder beim Abwaschen. Leider konnten sich meine Eltern solch ein Gerät nicht leisten – also mußte ich weiterhin diese Arbeiten verrichten! Wie bedauerlich.

Mein Cousin hatte Fischertechnik – und siehe da, dort war ein Roboterarm ziemlich einfach zu realisieren – natürlich nicht Computergesteuert, denn das Interface für den Commodore 64 kostete damals ca. 600,- Mark.

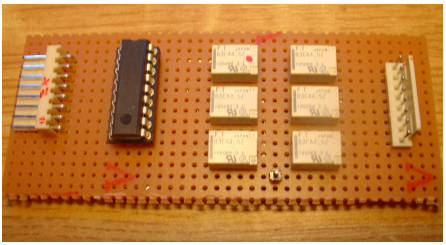
Bei 1,- Mark Taschengeld in der Woche = 4,- im Monat = 48,- im Jahr = *12,5 Jahre sparen* = Technik veraltet und keine Schokoladenzigaretten mehr = *undenkbar, niemals, never ever.*

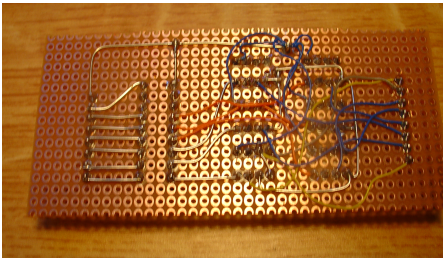
Als ich 1999 ins ZX-TEAM kam, schwebte mir eine Lichtorgel und ein Roboterarm vor, die vom ZX81 gesteuert würden. Meine ersten zaghaften Versuche, einen Roboter mit Servos und Holz zu bauen, schlugen zu 99% fehl. Die Servos hab ich noch... die Lichtorgel geht nicht, weil es nur noch Energiesparlampen gibt – und die brauchen 10 Minuten, um anzuspringen...

Weihnachten 2010 war es dann endlich soweit: bei www.pearl.de [1] (ja, der Katalog mit den süßen Weihnachtsfrauen) sah ich einen Roboterarm für 49,90 Euro... aus Plastik und mit einer Fernbedienung, die per Kabel angeschlossen war. Dazu gab es optional ein USB-IF, mit dem der Windoof-Rechner den Arm steuern konnte. Fünf Motoren und eine Lampe, damit der Greifer auch was sieht :-) … die Bestellung war in drei Minuten erledigt und das Paket drei Tage später zu Hause. 3 Stunden später war er aufgebaut.

Nach ein paar Telefonaten mit Olli aus Bremen wurde schnell klar: ja, eine Steuerung über den ZX81 ist möglich und relativ einfach. Die Teile hat Olli dann für mich bestellt und sogleich auch einen ersten Entwurf einer Relais-Schaltung gezeichnet. Auf dem Treffen hatte er alles dabei

und die Sache wurde in Angriff genommen. Das erste IF sah dann so aus:

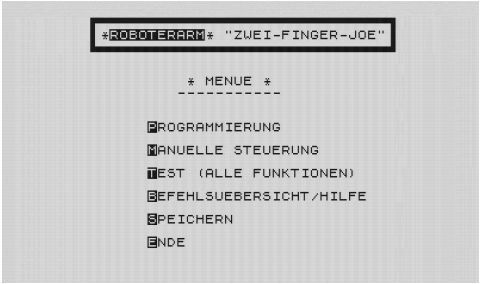




Die Steuersoftware war zunächst nur ein kleines BASIC-Programm mit ein paar Pokes, die zeitgesteuert ein Relais schalteten. Für jeden Motor gab es ein Relais, das sechste Relais schaltete die Richtung des jeweiligen Motors.

Noch auf dem Treffen konnte der Roboterarm, der von Thomas Rademacher spontan auf „Zwei-Finger-Joe“ getauft wurde, einen Zuckerwürfel automatisch greifen und in die Kaffeetasse werfen.

Zu Hause angekommen schrieb ich dann ein einfaches „MENUE“-Programm, welches diese Funktionen bietet:



Die Funktionen im Einzelnen:

Programmierung – hier können Befehlsfolgen eingegeben und „ZFJ“ programmiert werden

Manuelle Steuerung – hier wird eine Befehlsfolge eingegeben und direkt ausgeführt

Test – alle möglichen Bewegungen werden getestet

Befehlsübersicht/Hilfe – hier werden der Roboterarm und die Befehle erläutert

Speichern – Speichern der Befehlsfolgen (auf Kassette, C64-Diskette und USB)

Ende – äh...

Eine Befehlsfolge sieht zB so aus:

```
A-4B-5D-8D-4G+2C-5A-
1G+3A+5C+5B+5D-8D-8D-4A-2G-4
```

An einem Menue-Programm, welches in Assembler geschrieben ist, träume ich noch. Aber wer weiß, vielleicht wird das ja auf dem Treffen mit Hilfe von Euch etwas :-)

Quellen und Weblinks

[1] www.pearl.de

Interface für„Zwei-Finger-Joe“

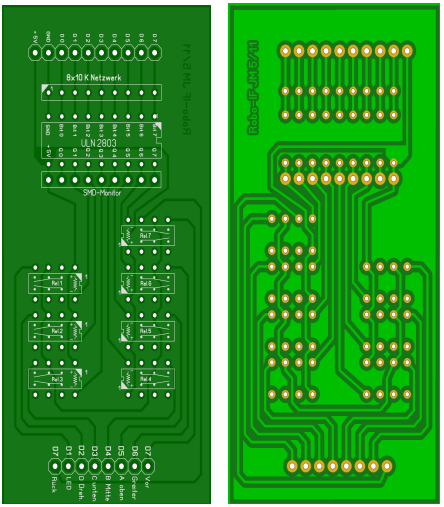
Roboterarm am ZX 81 *(2)*

Die optimierte PIO-Ansteuerung

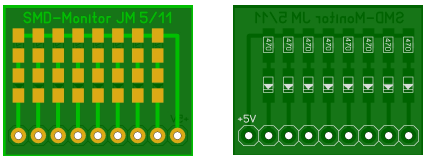
Von Joachim

Als ich auf dem 2011er ZX-Team-Treffen Jens' Roboterarm sah, mußte ich auch einen haben. Die Ansteuerplatine hatten Jens und Oliver auf dem Treffen auf Lochraster zusammengelötet und ein Prográmmchen zum Zuckerstückchen-in-die-Tasse-geben hatten sie auch schon geschrieben.

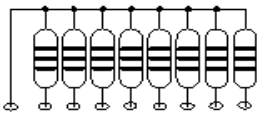
Zuhause angekommen studierte ich zunächst das selbstgestrickte Interface und erstellte davon diesen Schaltplan mit ein paar Optimierungen *(siehe Schaltplan)*.



Anschließend routete ich die Platine. Zur Unterstützung bei der Programmierung entwarf ich noch diesen LED-Monitor, den man bei Bedarf anstecken kann.



Zwischen dem Stecker, der zur PIO führt, und dem ULN2803 ist ein Widerstandsnetzwerk mit 8 Widerständen (4,7 KOhm bis 10 KOhm). Als Ersatz geht auch das:



GND 8 x 10 KOhm

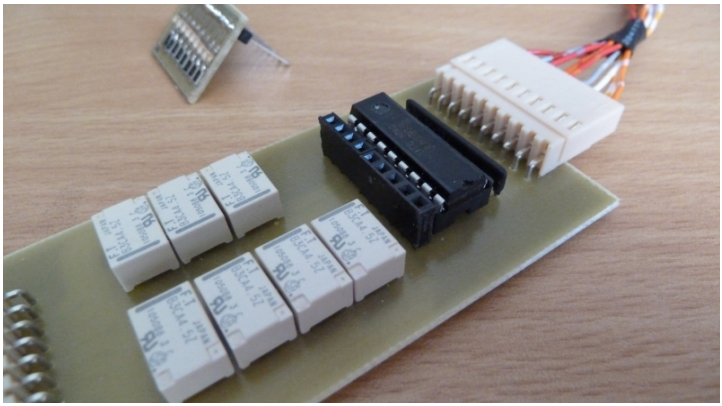
Das Netzwerk sorgt als Pulldown-Widerstand dafür, dass beim Einschalten Low-Pegel am ULN2803 anliegt. Damit wird kein Relais angesteuert. Nach dem ULN2803 sind noch neun Bohrungen. Diese sind für einen optionalen SMD-LED-

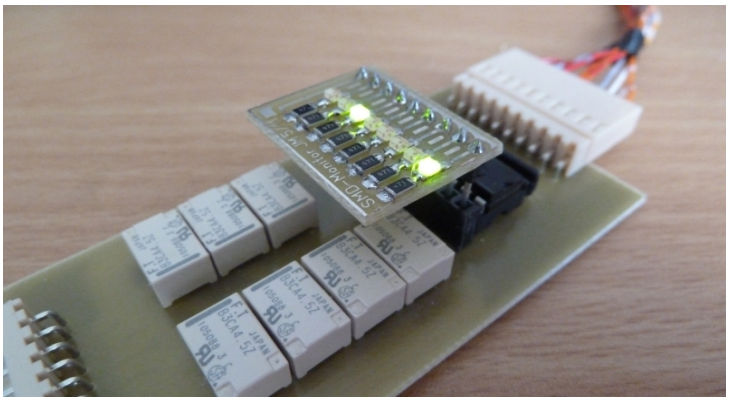
Monitor.
In diese Bohrungen wird eine Buchsenleiste gelötet. Dort kann man den Monitor dann einstecken.

In die Monitor-Platine kommt eine Stiftleiste. Der Monitor wird wie auf dem Bild gezeigt mit der Stiftleiste in die Buchse gesteckt. Der Monitor ist für die Programmierung recht hilfreich. Man sieht, was ausgegeben wird und muß nicht unbedingt den Roboterarm angesteckt haben.

Bei den Relais vom Typ FTR B3CA4,5Z (Reichelt) habe ich mich vertan. Ich dachte die Pins sind DIL. Leider nicht, deshalb muß man ein wenig pfrimeln. Geht aber.

Für einen schnellen Funktionstest mit dem Roboterarm habe ich das folgende Maschinenprogramm erstellt.



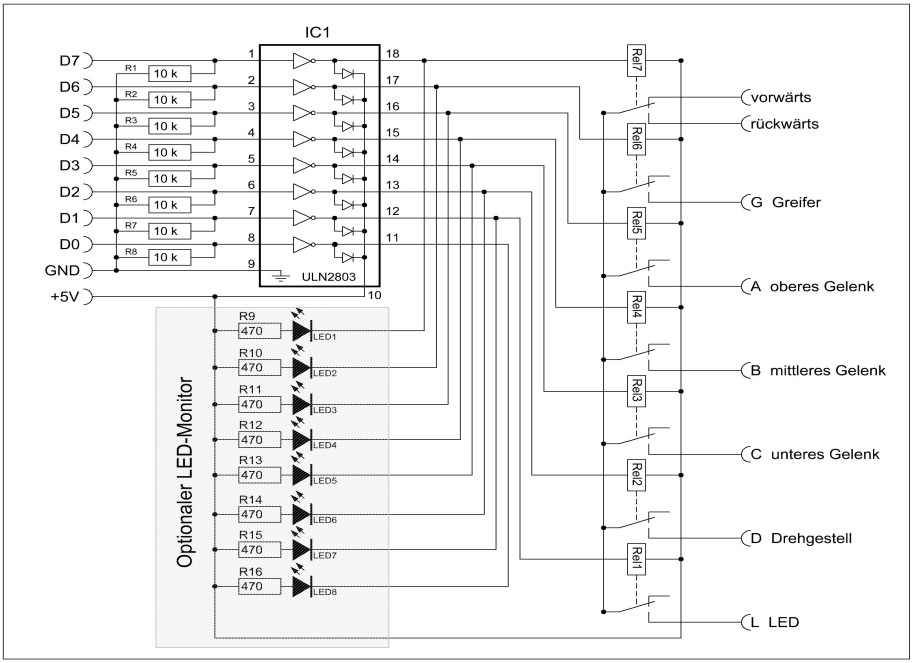


```
4084 START CALL INIT      CD8A40
40D4       RES 7,A        CBBF
4087       JP MAIN        C3E540
40D6       OR E           B3
408A INIT  LD A,$0F       3E0F
40D7       OUT DB,A       D3CF
408C       OUT PB,A       D3DF
40D9 LDWN  CALL INKEY     CD9440
408E       XOR A          AF
40DC       CP C           B9
408F       OUT DB,A       D3CF
40DD       JR NZ,STOP     2002
4091       LD E,$00       1E00
40DF       JR LDWN        18F8
4093       RET            C9
40E1 STOP  XOR A          AF
4094 ;;
40E2       OUT DB,A       D3CF
4094 INKEY PUSH BC        C5
40E4       RET            C9
4095       PUSH DE        D5
40E5 ;;
4096       CALL $02BB     CDBB02
40E5 MAIN  CALL INKEY     CD9440
4099       LD A,$FF       3EFF
40E8       CP $26         FE26
409B       CP L           BD
40EA       CALL Z,CMDA    CCA740
409C       JR Z,EXIT      2806
40ED       CP $27         FE27
409E       LD B,H         44
40EF       CALL Z,CMDB    CCAA40
409F       LD C,L         4D
40F2       CP $28         FE28
40A0       CALL $07BD     CDBD07
40F4       CALL Z,CMDC    CCAD40
40A3       LD A,(HL)      7E
40F7       CP $29         FE29
40A4 EXIT  POP DE         D1
40F9       CALL Z,CMDD    CCB040
40A5       POP BC         C1
40FC       CP $2C         FE2C
40A6       RET            C9
40FE       CALL Z,CMDG    CCB340
40A7 ;;
4101       CP $31         FE31
40A7 CMDA  LD B,OG        0620
4103       CALL Z,CMDL    CCB640
40A9       RET            C9
4106       CP $23         FE23
40AA CMDB  LD B,MG        0610
4108       CALL Z,UP      CCC240
40AC       RET            C9
```

```
410B       CP $22         FE22
40AD CMDC  LD B,UG        0608
410D       CALL Z,DWN     CCD240
40AF       RET            C9
4110       CP $70         FE70
40B0 CMDD  LD B,DG        0604
4112       CALL Z,UP      CCC240
40B2       RET            C9
4115       CP $71         FE71
40B3 CMDG  LD B,GR        0640
4117       CALL Z,DWN     CCD240
40B5       RET            C9
411A       CP $72         FE72
40B6 CMDL  LD A,E         7B
411C       CALL Z,UP      CCC240
40B7       XOR LED        EE02
411F       CP $73         FE73
40B9       LD E,A         5F
4121       CALL Z,DWN     CCD240
40BA LCMDL CALL INKEY     CD9440
4124       CP $21         FE21
40BD       CP $31         FE31
4126       CALL Z,DWN     CCD240
40BF       JR Z,LCMDL     28F9
4129       CP $24         FE24
40C1       RET            C9
412B       CALL Z,UP      CCC240
40C2 UP    LD C,A         4F
412E       CP $00         FE00
40C3       LD A,B         78
4130       CALL Z,STOP    CCE140
40C4       SET 7,A        CBFF
4133       CP $E3         FEE3
40C6       OR E           B3
4135       RET Z          C8
40C7       OUT DB,A       D3CF
4129       CP $24         FE24
40C9 LUP   CALL INKEY     CD9440
412B       CALL Z,UP      CCC240
40CC       CP C           B9
412E       CP $00         FE00
40CD       JR NZ,STOP     2012
4130       CALL Z,STOP    CCE140
40CF       JR LUP         18F8
4133       CP $E3         FEE3
40D1       RET            C9
4135       RET Z          C8
40D2 DWN   LD C,A         4F
4136       JR MAIN        18AD
40D3       LD A,B         78
4138 ;;
```

Man kann damit den Roboterarm mit der Tastatur steuern. Aufruf ist RAND USR 16516. Für die Gelenke die Tasten A-C, für das Drehgelenk D, für den den Greifer G und die LED das L drücken. Damit gibt man erst mal vor, was bewegt werden soll. Einfach einmalig drauf drücken. Mit den vier Pfeiltasten wird dann das gewählte Gelenk nach oben oder unten bewegt. Sobald man eine der Pfeiltasten losläßt, stoppt der Vorgang. Nothalt ist die Space-Taste. Soll ein anderes Gelenk dran kommen, dann einfach eine entsprechende Gelenk-Taste drücken. Um aus dem Programm zurück zu Basic zu kommen, einfach die STOP-Taste (Shift A) drücken. Die L-Taste ist eine Spezialität: Einmal gedrückt, wird die LED bei jedem Vorgang eingeschaltet, nochmals gedrückt wird die LED ausgeschaltet.

Zur Ansteuerung wird der Port B einer Z80-PIO mit den Adressen $C7 / $D7 verwendet.



Schaltplan – Interface zum Roboterarm

UMSCHALTBOX

Die Umschaltbox

Diese kleine 'Kiste' ist ein Requisit aus der Zeddy-Mailbox-Zeit.

Von Ulrich



Das Modem wurde über die PIO angesteuert, damit war für die Dauer der Mailboxverbindung der rückseitige Anschluss meiner Erweiterungsbox belegt. Sollte die PIO für andere Dinge verwendet werden, war erst eine Umstöpselei erforderlich. Diese lästige Fummelei hatte mich zum Bau der Umschaltbox veranlaßt. Vier Relais mit jeweils zwei Umschaltkontakten sorgen dafür, dass die acht Port-Leitungen (nur Port A) bedarfsweise umgeschaltet werden. Mit dem Einschalten des Modems wurde der Transistor angesteuert, der die Relais aktiviert. Dadurch erfolgte die Umschaltung automatisch.

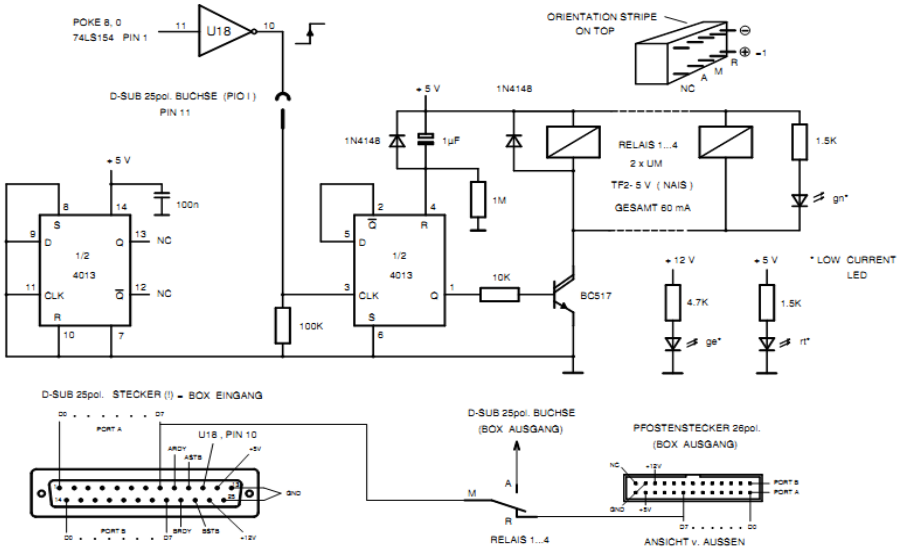
Mailbox und Modem sind aber schon lange nicht mehr im Einsatz. Die Umschaltung der Port-Leitungen sollte aber auch ohne Einschränkung erfolgen, deshalb wurde die Relaisansteuerung nun geändert.

Gleichzeitig dient die Umschaltbox auch als Adapter vom D-SUB-Stecker auf den 26pol.-Pfostenverbinder für Flachbandkabel, denn der Auswurfmechanismus dieser Steckverbinder ermöglicht eine schonende Trennung der Verbindungen zu weiterer Hardware (Messmodule, Schrittmotorsteuerung, VDRIVE u.v.m.).

Die Lösung: POKE 8,0

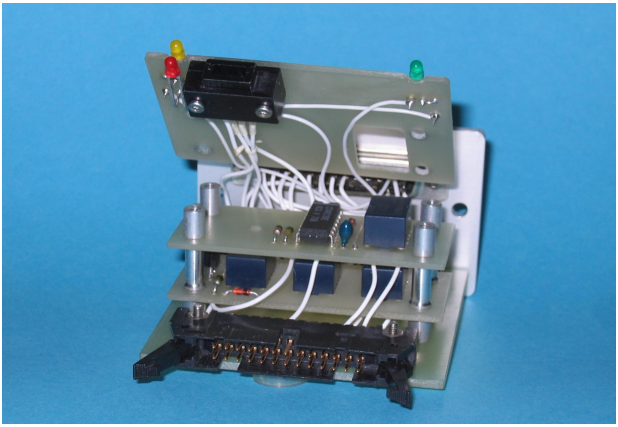
Dieses Kommando erzeugt bei der POKE 8-Schaltung am IC 74LS154, Pin 1 dauerhaft einen LOW-Pegel. Wird aber ein anderer POKE 8-Befehl gegeben, geht Pin 1 wieder auf HIGH, so also nicht zu gebrauchen. Die Relaisansteuerung erfolgt nun durch Speicherung des Kommandos mit einem Flip-Flop. Dafür wurde das D-Flip-Flop 4013 (CMOS) eingesetzt. Mit jeder positiven Flanke am Clockeingang (3) wird der Pegel am D-Eingang (5) auf die gegenphasigen Ausgänge (1u.2) übertragen. Durch die Verbindung von Pin 2 zu Pin 5 wird erreicht, dass am Ausgang (1) abwechselnd ein HIGH bzw. LOW

ansteht. Für die positive Flanke bei Kommandogabe muß das POKE 8,0 Signal invertiert werden. Dafür wurde ein bisher ungenutzter Inverter (74LS04) eingangsseitig mit dem Pin 1 des LS154. verbunden. Der Ausgang führt nun über einen freien Pin (11) der D-SUB-Buchse (PIO-Ausgang) meiner Erweiterungsbox nach 'draußen', von dort über das Kabel in die Umschaltbox zum Pin 3 am IC 4013.

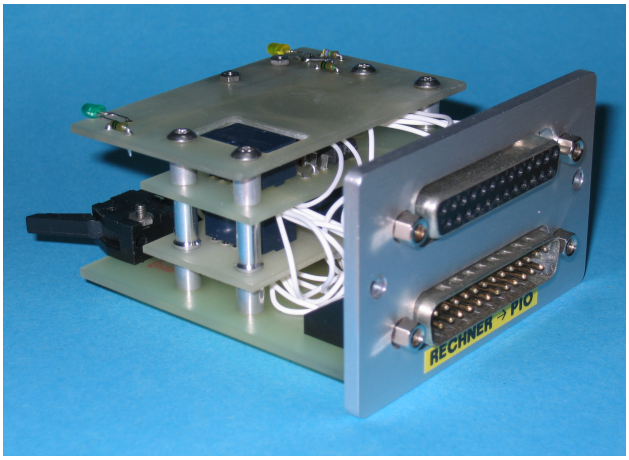


Das Schaltbild zeigt, wie die Verbindungen angelötet werden müssen, das IC 4013 ist freiverdrahtet nachgerüstet. Kleine Hohlniete (gekürzte Aderendhülsen) dienen der Fixierung der Bauelemente auf der Platine. Das zweite Flip-Flop im selben IC-Gehäuse bleibt ungenutzt.

Sind die Relais angezogen, wird über die grüne Leuchtdiode die Umschaltung angezeigt. Die beiden anderen Leuchdioden signalisieren die am PORT-Stecker verfügbaren Spannungen, rot für +5 Volt, gelb für +12 Volt. Auf den Bildern ist an der Frontplattenseite noch eine kleine 3-pol. Buchse zu sehen, sie wird für den Anschluß eines Funkgongs verwendet (s.a. TM-Magazin 4/2004, S.22) .



Aufbau auf 3 Etagen



Der kompakte Aufbau in Rückansicht

*Hier noch eine Anmerkung zur POKE 8,... Schaltung * (POKE-KARTE V 2.0)*

Nach dem Einschalten ist immer POKE 8,15 aktiv! Dadurch kann man unmittelbar mit RAND USR 8192 MEFISDOS aufrufen.

Ulrich

Quellen und Weblinks

http://www.fischerkai.de/zxteam/hdif_d.htm

http://www.fischerkai.de/zxteam/pokekart.gif

.

ANDY'S ULA

Andy's ULA-Nachbau im englischen Forum

Die ULA ist das Spezialbauteil im ZX81, welches Sinclair's Firmengeheimnisse birgt. Im englischen Forum ist der ZX-Bastler Andrew Rea seit einiger Zeit mit Nachbauten dieses Spezialchips sehr aktiv. Nun ist dabei ein käufliches Produkt herausgekommen, welches mehr ist als nur ein Ersatzteil.

Von Paul

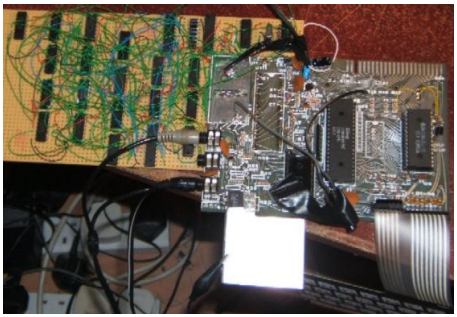
Da die ZX81 ULA recht empfindlich ist und im Betrieb recht heiß wird ist sie, neben der Folientastatur, die Hauptausfallursache bei Zeddys. Bei mir liegen z.B. neben drei funktionsfähigen ZX81 Platinen weitere neun defekte, davon fünf bereits ohne ULA, bei den anderen ist die ULA verlötet und nur deshalb noch drin.

Historisches

Nachdem Andy in seiner Jugend ein paar ZX81 hatte gab es eine Weile Abstinenz von den Zeddys. Der Emulator von Michael Wyne [6] weckte Andys Interesse an den Zeddys erneut und er baute zwei ZX80. Der erste nach eigenen Ideen lief leider nie, daher baute er einen weiteren nach den Informationen von Grant Searle, den er mit ein wenig Hilfe von Rodney Knapp auch ans laufen bekam.

Nachdem er wieder einen ZX80 hatte kaufte er kurz darauf einen ZX81. Einen NMI-Generator hatte Andy nicht gebaut da er aufgrund des gekauften ZX81 wieder ZX81 Programme nutzen konnte.

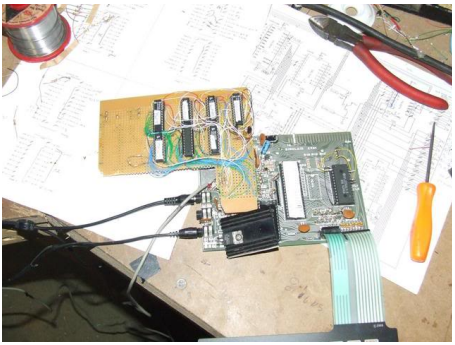
Im Alten TS1000 Forum gab es diverse Diskussionen um ULA-Ersatz, insbesondere der ULA-Nachbau von Bodo Wenzel [1] hatte es ihm angetan. Da er allerdings (bis heute) kein Abel entziffern kann hat er im April 2007 die Idee eine ganze ULA in TTL nach zu bauen umgesetzt. Dazu hat er die ULA in unabhängige Module aufgeteilt zu denen er viele wichtige Informationen von Siggi und Wilf Rigter erhalten. Das kann man in [2] mal nachlesen.



Den dazu verwendeten ZX81, den er ohne ULA und ohne CPU erwarb hat er noch heute.

Nachdem Andy mal wieder eine kleine Zeddypause überwunden hatte wunderte er sich das immer noch

niemand einen praktikablen ULA-Ersatz entwickelt hatte. Nach einem Gespräch mit SirMorris über GALs und der Feststellung das sein Eprommer auch GALs programmieren kann hat Andy Anfang März 2011 angefangen mit GALs zu experimentieren, zwei Wochen später hatte er etwas lauffähiges [3].

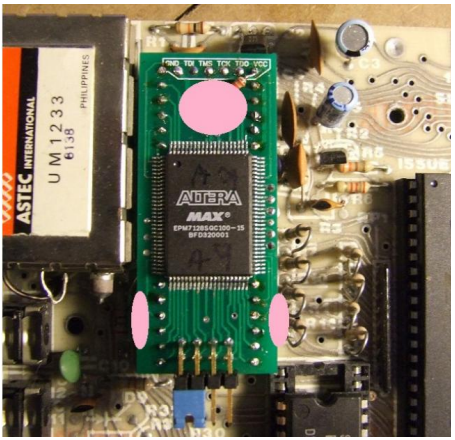


Darauf hin hat SirMorris gemeint CPLDs seien ja fast das gleiche und da er ja jetzt schon mal mit wincupl umgehen kann.... SirMorris neigt hierbei gerne zu Untertreibungen wenn er meint etwas sei eigentlich ganz einfach, jedenfalls für jemanden der es kann. Darauf hat Andy es dann mal mit einem CPLD versucht. Ursprünglich hatte Andy etwas billiges im Sinn. Daher hat er ein (zu) kleines 3.3V CPLD (5V tolerant) mit einem GAL kombiniert. Allerdings gab es diverse Probleme mit statischen Ladungen und gelegentlichen Abstürzen. Man hatte schon Sorgen das dieses Projekt so kurz vor dem Ziel nie zum Abschluss käme.

Aktuelles

Dann hat Andy in den sauren Apfel gebissen und eine Quelle für 5V CPLDs gesucht und gefunden. Das der verwendete Chip grösser ist vereinfacht das Layout, nebenbei verkleinert sich die Platine um 20%.

Daraus entstand dann die inzwischen käuflich erwerbbare Andy-ULA. Bei Sellmyretro [4] inklusive Einbau.



Diese funktioniert ohne zusätzliche Lötarbeiten (bei gesockelter ULA) und bietet dann die original Funktionalität.

Funktionelle Erweiterungen

Andy hat noch ein paar Extras eingebaut. Damit diese genutzt werden können benötigt die Andy-ULA noch ein paar weitere Adressleitungen (A9..A13) die von den Tastaturdioden an die Andy-ULA-Platine verbunden werden können sowie die Resetleitung damit

die Andy-ULA bei einem Reset wieder in ihren ursprünglichen Modus gesetzt wird. Dann kann man die ULA mit zwei Jumper in einen von vier Modi setzten, es gibt ZX81, ZX81 erweitert, ZX80 und ZX80 erweitert. Im Lieferzustand ist ZX81 gesetzt.

Der erweiterte ZX81 Modus

Die Erweiterungen werden per POKE gesteuert, Dazu sollte man sich für den ZX81 die ROM-Adressen 101 und 1001 merken. Per POKE 101 werden die Erweiterungen geschaltet, per PEEK 101 erfährt man was man gesetzt hat. Das wird Binär gesetzt und zwar folgende BITs:

Bit 7: Turbo
Bit 6: M1NOT
Bit 5: Nur den Rand invertieren (benötigt ebenfalls BIT 4)
Bit 4: Bild Invertieren, also Rand und Ausgabebereich
Bit 3: ROMPATCH: Im Turbomodus kann normalerweise nicht zuverlässig von FAST auf SLOW gewechselt werden da die Warteschleife zum Test des NMI zu knapp bemessen ist. Deshalb kann die Adresse 531 des ROMs mit diesem Bit von 17 auf 34 gepatcht werden um die Wartezeit wieder den 6,5 Mhz anzupassen.
Bit 2: Invertiert die 50/60 Hz Erkennung. So können Programmierer testen ob ihre Programme im jeweils anderen Modus auch laufen ohne Jumper zu setzen
Bit 1: erlaubt im FAST Modus auch Turbo zu triggern. Dies geschieht übe POKE 1001,x. Der Wert x ist dabei egal, es muss nur diese Adresse geschrieben werden um TURBO zu aktivieren, bei jedem Input oder Stop, also immer wenn Bild angezeigt werden muss wird automatisch Turbo wieder deaktiviert. Nach jedem Input sollte also wieder ein POKE 1001,0 oder so ähnlich stehen damit die Post abgeht. Normale ZX81 mit alten ULAs stört das ja nicht. Für diese Funktionalität muss (natürlich) auch Bit 7 gesetzt sein.
Bit 0: LOCK verhindert das weitere POKE 101,x etwas bewirken bis zum nächsten Reset oder Einschalten.

Der Turbomodus bereitet Benutzern von Soundmodulen Probleme (wechselnde Frequenz am Takteingang) weshalb es, neben einem Ausgang für eine TURO-LED, auch einen Pin gibt der permanent 3,25 Mhz liefert und auch bei allen Takt-Umschaltungen stabil bleibt, so dass man dies auf den Expansionport legen kann (die Original Taktleitung muss dazu unterbrochen werden).

Der erweiterte ZX80 Modus

Wie bereits erwähnt hat die ULA auch einen ZX80 Modus.

Beim ZX80-Modus ist naturgemäß kein NMI aktiv so das man nur noch das ZX80 ROM benötigt und einen ZX80 kompatiblen Rechner hat. Damit die Erweiterungen (möglichst viele davon) benutzbar bleiben gibt es ein paar Änderungen:

Der POKE 101,x wird ersetzt durch POKE 77,x. Also schaltet POKE 77,128 den Turbomodus ein. Allerdings benötigt der ZX80 keinen Rompatch und kein retriggern des Turbomodus. Bit 3 und 1 sind also beim ZX80-Modus unbenutzt.

Die Bilddarstellung erfolgt in 3,25 Mhz, der Rest schnell. Vor Load und Save also abschalten.

Leider ist die Adresse 77 nicht ganz so gut geeignet wie 101 beim ZX81 dadurch liefert PEEK 77 Bit 0 immer 0 und nicht 1 wenn die Konfiguration festgeschrieben wurde.

Probleme mit dem Turbomodus?

Generell kann man vor der Verwendung des Turbomodus warnen da hierzu ggf. zu langsame Komponenten gegen schnellere ausgetauscht werden müssen. Das ROM ist mit der Aufschrift „D2364C“ ist hierbei ein Kandidat, das ROM mit der Aufschrift „ZCM38818P“ scheint schnell genug zu sein, diese Liste muss dann ggf. erweitert werden.

Die 4Mhz CPU wird wahrscheinlich im Dauerbetrieb abstürzen, falls sie überhaupt funktioniert (Bauteil-streuungen), mindestens eine 6 MHz CPU wäre ratsam. Beim RAM gab es seltener Probleme, auch hier werden wir ggf. weitere Erfahrungen sammeln müssen.

Der Videoausgang

Die ULA läuft bei mir seit längerem problemlos, auch mal mehrere Wochen am Stück.

Einen kleinen Pferdefuß gibt es aber auch: Der Videoausgang ist als solcher gedacht und nicht als Quelle für unsere Modulatoren. Hier muss noch genauer geforscht werden wie man am besten auf die Original-Impendanz anpasst. Ich mag es lieber wenn an der Modulatorbuchse sowohl das Videosignal als auch das Fernsehsignal gleichzeitig anliegen [7]. Das habe ich zur Zeit nur mit Qualitätseinbußen hinbekommen. Wenn das Videosignal direkt auf einen Monitor gegeben wird bekommt man allerdings ein super Bild. Dafür hat Andy dies entwickelt und beschreibt den Umbau auch in seiner Anleitung [5].

Selber CPLD programmieren?

Ich hoffe auf möglichst viele Erweiterungen auf Basis dieser ULA da deren Firmware sehr leicht erweiterbar ist. Andy hat den Altera EPM7128SQC100-15 gewählt und hat dabei noch etwa 25 freie IO's die jedoch nicht auf Lötpads herausgeführt wurden, also direkt angelötet werden müssen. Dies gibt den Bastlern (also uns) jede Menge Freiheit. Die Programmier-schnittstelle JTAG die dazu benötigt wird, liegt auf Lötpads, an die ein passendes Kabel für den Programmer

gelötet werden kann.

Hier die Liste aller in der Andy-ULA bereits verfügbaren Leitungen:

5V, 0V, OSC, /RD, /WR, /IORQ, /MREQ, /M1, /NMI, /HALT, D0..D7, A0'..A8', A9..A13 (bei erweiterter Verdrahtung), A14, A15, KBD0..KBD5, TV-TAPE, TAPE IN, UK/US, /RAMCS, /ROMCS, /CLK

(Input: Blau, In&Out: Grün, Output: Rot)

Es besteht die Möglichkeit eine Firmware für die ULA zu erzeugen die A0'..A8' nur dann mit dem UDG Teil überschreibt wenn die Adresse in einem festgelegten Bereich z.B. von 0-12K liegt. Dadurch kann RAM im Bereich von 8-12K für UDG genutzt werden wenn diese Adressleitungen verwendet werden statt A0..A8. Alles andere dagegen als Hires. Hier schwebt mir eine Erweiterung mit 512K vor die auf dem ROM-Sockel sitzt, Batterie gepuffert wird (oder gleich MRAM?), RAM statt ROM einblenden kann, UDG und Hires per Poke oder Jumper einstellbar bekommt sowie natürlich Bankumschaltung unterstützt.

Man sieht das mit einer frei veränderbaren ULA ganz andere Möglichkeiten bestehen als mit Add-Ons. Wie wäre es wenn die ULA direkt den SPI der MMC bedienen würde und die Parallel-Seriell-Wandlung dabei übernimmt?

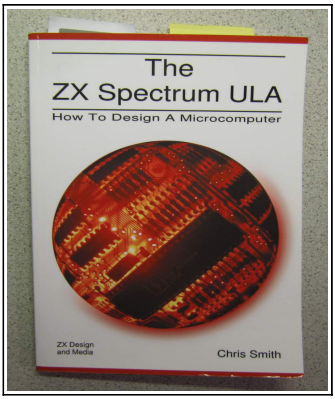
Gruß Paul

Quellen und Weblinks

[1] http://members.fortunecity.com/bodo4all/zx/zx81vid.html
[2] http://homepage.ntlworld.com/deborah.clayton1/ULA/ula.html
[3] http://www.rwapservices.co.uk/ZX80_ZX81/forums/ula-revistited-t445s150.html#p3620
[4] http://www.sellmyretro.com
[5] http://www.debraclayton.webspace.virginmedia.com/zxulamk2.pdf
[6] http://www.aptanet.org/eightyone/
[7] http://forum.tlienhard.com/phpBB3/viewtopic.php?f=7&t=521

BUCHREZENSION

Der ULA ins Silizium geschaut



Chris Smith
The ZX Spectrum ULA
How To Design A Microcomputer
ZX Design and Media, UK 2010, 300S.

Rezensent: Oliver

Es gibt Bücher, bei denen ist auf Grund des sehr speziellen Themas bereits vor dem Erscheinen klar, dass keine allzu hohe Auflage zu erreichen ist. In solchen Fällen ist zumeist die starke Begeisterung des Autors für die Sache die treibende Kraft. „The ZX Spectrum ULA“ ist solch ein Buch. Chris Smith hat sich mit unglaublicher Akribie dem Spezialchip des ZX Spectrum gewidmet. Für den Sinclair- und ZX81-Fan fallen dabei auch viele Informationen ab, denn die Grundtechnologien sind bei allen ULA Bausteinen gleich. Neben rein technischen Fakten gibt es auch Geschichten und Geschichtliches.

Zunächst geht es um die britische Firma Ferranti Semiconductors, Hersteller der ULA-Chips. Ferranti hat in den 1960ern die ersten integrierten Halbleiterbausteine Europas hergestellt. Gegen die großen amerikanischen Hersteller konnte Ferranti sich aber trotz viel versprechender technologischen Ansätze nicht durchsetzen. Daher konzentrierte man sich auf den Nischenmarkt der Spezial-ICs. ULA steht für Uncommited Logic Array, übersetzt etwa „nicht festgelegtes Feld von Logikzellen“. Grundprinzip dabei ist, dass die funktionalen Teile eines ULA-Chips von Ferranti standardmäßig immer gleich produziert werden, und ein Industriekunde wie Sinclair dann nur die Verschaltung der Zellen im IC individuell festlegt. Damit muss nur die oberste Metallisierungsmaske kundenspezifisch erstellt werden und man spart gegenüber einem „echten“ Spezial-IC bei mittleren Stückzahlen viel Entwicklungsaufwand.

Die Ferranti-ULA-Bausteine sind technisch gesehen recht exotische bipolare Halbleiter. Damit unterscheiden sie von den normalerweise für höher integrierte ICs benutzten Feldeffekttransistor-basierten (MOS

bzw. CMOS) Bausteinen. Der bipolare Aufbau des ULA-Chips erlaubt aber die direkte Integration von analogen Funktionen wie Taktoszillator und Videoausgang. Damit ist aber auch klar, dass eine Zeddy-ULA durch ein modernes CMOS-IC nicht komplett funktionsgleich ersetzt werden kann, sondern ein Nachbau wohl immer externe Analogbauteile benötigt.

Die ZX81 ULA entstammt Ferranti's 2000er Serie mit „very high speed grade“ C, daher das 2C... im Namen.

JimWestwood, Entwickler bei Sinclair, hat für den ZX81 sein modifiziertes ZX80-Design mit den vielen Logikbausteinen in ein Ferranti-ULA-IC integriert und damit den Grundstein zum Erfolg des kostengünstigen Computers gelegt. Unterstützt wurde er vom neu angeheuerten Richard Altwasser, der dann später auch für die ZX Spectrum Hardware verantwortlich war. Jim Westwood war einer der zentralen Mitarbeiter bei Sinclair, nach Aussage des Sinclair-Mitarbeiters Brian Flint in [1] war Westwood dann um 1990 auch der letzte noch verbliebene Mitarbeiter nach dem Kollaps der Firma

Dazu wird dann noch die spannende Geschichte von der Inbetriebnahme der ersten funktionierenden Spectrum ULA erzählt. Diese gelang nur durch einen unglaublichen Zufall - nämlich durch ein Staubkorn, welches sich im Fertigungsprozess an exakt der richtigen Stelle eingeschlichen hatte.

Der Autor Chris Smith hat für die genaue Analyse der ZX Spectrum ULA tatsächlich mehrere dieser ICs aufsägen lassen und die Verschaltung unter dem Mikroskop mühevoll bis ins Detail analysiert. Genau erklärt er Funktion und Fehler in der ULA, die zum Teil auf der Spectrum-Leiterplatte durch Extralötungen umgangen wurden. Wer schon immer wissen wollte, wieso beim Spectrum die Adressierung der Zeilen im Videospeicher so komisch organisiert ist, findet im Buch die Erklärung; hier nur soviel - es ist kein Fehler, sondern ein Erfordernis durch das Zeitverhalten der Speicherbausteine.

Smith hat die definitive Bibel zur ZX Spectrum ULA geschrieben. Dafür hat er alte papierne Archive durchforstet, denn Ferranti ist schon vor Beginn des Internetzeitalters von der Bildfläche verschwunden, und er hat ehemalige Sinclair-Mitarbeiter befragt. Damit ist das Buch eine informative und unterhaltsame Lektüre für interessierte ZXler.

Quellen und Weblinks

[1] http://www.polymathperspective.com/?P=414

PROMI-KLATSCHPRESSE

Die folgende Meldung ging im April 2010 durch die englische Presse – als ZX Team Magazin werden wir damit unserer Chronistenpflicht gerecht. Man beachte den in einem Computermagazin nicht immer sofort gegenwärtigen feinen Unterschied zwischen *Laptop* und *Lapdance* – notfalls mal im Internet nachschauen oder jüngere Leser sollten besser erst bei den Eltern nachfragen...


The Telegraph

HOME | NEWS | SPORT | FINANCE | COMMENT | BLOGS | CULT

UK | World | Politics | Obituaries | Education | Earth | Science

HOME » NEWS » CELEBRITY NEWS

Computer tycoon Sir Clive Sinclair, 69, w former lapdancer fiancée

Sexagenarian computer tycoon Sir Clive Sinclair has married l former lapdancer Angie Bowness, in a low-key Las Vegas cere reported.

image 1 of 2

Sir Clive with Angie Bowness Photo: MURRAY SANDERS

7:59AM BST 27 Apr 2010

Sir Clive, 69, who invented the pocket calculator and the ill-fated C5 electric car, is said to have married Miss Bowness in the US city's Civ

Aufgespürt und recherchiert von Ali:

Im Alter von 69 Jahren hat Clive Sinclair wieder geheiratet. Diesmal eine 33-jaehrige Frau/eine 36 Jahre juengere Frau. Er hat sie in einem Striplokal beim Lap Dance kennen-gelernt. Angie Bowness, eine ehemalige Schoenheitskoenigin, will Clive natuerlich nur aus Liebe heiraten …

Quelle:

http://www.telegraph.co.uk/news/celebritynews/7637479/Computer-tycoon-Sir-Clive-Sinclair-69-weds-33-year-old-former-lapdancer-fiancee.html

DEMNAECHST ... 4MBYTE AM ZX81

Leider hat die Zeit bis zum Redaktionsschluss nicht ganz gereicht für einen ausführlichen Artikel zur **4-MByte-ZX81-Speichererweiterung** von **Ali**. Die Grunddaten lesen sich dabei schon höchst spannend: Viel Platz gibt es für die acht statischen RAM-Bausteine auf einer großen Leiterplatte im Doppeleuroformat. Eine Bankumschaltung per POKE-9-Befehl wird von einer LED-Anzeige unterstützt. Komplett gepufferte Busleitungen und Erweiterungsstecker zählen zu den weiteren Features dieses Projekts, das derzeit in den Expertenhänden von **Kai** ans Laufen gebracht wird. Die Schaltung funktioniert bereits, Details lassen sich dann Live in Mahlerts bewundern.

KLEINANZEIGEN

Biete für aktive ZX-TEAM-Mitglieder eine emailadresse@zx81.de oder emailadresse@ts1000.org in Form einer Weiterleitung auf Eure bestehende Mailadresse. Vorzugsweise in der Form "vorname@zx81.de" oder "Forumsnickname@zx81.de"

Für "verdiente Aktivisten der Zeddyklasse" ist gegen eine kleine Spende auch eine echte IMAP bzw. POP3-Mailadresse möglich.

peter@zx81.de

EIN BLOG AUF PAPIER ...

Ein Blog auf Papier...

Samstag Vormittag im Februar. Diesen Tag hatte ich extra freigehalten, denn heute sollten die Artikel des ZXTM Sonderhefts - soweit noch nicht geschehen - formatiert und zusammengestellt werden.

10:30. Es klingelt an der Tür. Ach ja, ist ja Samstag: Zwei freundliche Herren fragen mich höflich, ob ich schon mal etwas von Harmagedon gehört habe. Kurz schießt mir der Gedanke in den Kopf, es könne sich dabei um ein ZX81-Spiel handeln. Um nicht

zu ketzerisch zu erscheinen behalte ich den Gedanken jedoch für mich.

Für Ablenkung ist aber eigentlich keine Zeit, also schnell zurück zum Layout. Die anfänglichen Bedenken kommen in Erinnerung: Würden überhaupt genügend Beiträge für ein ZX Team Magazin Heft zusammen kommen? Ist ein Magazin in Papierform nicht veraltet angesichts der vielfältigen Möglichkeiten der elektronischen Kommunikation und Vernetzung?

Veraltet? Mooooooment. Als ZX81-Fan kennt man das Wort 'veraltet' doch gar nicht - oder man versteht es als Herausforderung.

Das Zuhause des ZX TEAM bilden zweifelsohne Tom Lienhard's großartiges Forum und das von Peter organisierte jährliche Treffen in Mahlerts. Das Treffen hat konstant hohe Teilnehmerzahlen, die ZX-Freunde legen dafür viele Kilometer zurück. Mahlerts 2012 wird, gemessen an den Vorankündigungen, wieder ein echtes Highlight. Im ZX81-Forum sprudelt es mitunter geradezu vor Aktivität, spannende Projekte werden dort gemeinschaftlich vorangetrieben.

In Ergänzung dazu findet sich im Magazin vielleicht die Nische für die abgeschlossenen Projekte - von den Autoren sorgfältig aufbereitet und dokumentiert für das papierne Archiv, das den Zugriff auch bei Stromausfall und Kerzenlicht noch erlaubt.

Dank der vielen Einsendungen und Beiträge ist dieses Heft viel umfangreicher und vielfältiger als erhofft. Ein riesiges Dankeschön an alle Autoren, aber vor allem an die Mitstreiter des Redaktionsteams - Klaus, Thomas, Paul, Jens - die für das Zustandekommen dieses Sonderheftes gesorgt haben. In jedem Fall hat die Arbeit daran unheimlich viel Spaß gemacht. An dieser Stelle sei aber auch noch mal ein riesengroßer Hut gezogen vor Peter und Joachim, die so etwas mit großem Einsatz über Jahre und nicht nur mal 'just for fun' gemacht haben.



Redaktionsklausur mit hohem Praxisanteil

Sofern der Weltuntergang nicht dazwischen kommt wird die Zukunft der ZX-Gemeinde wohl immer wieder neue Themen bringen – womöglich so etwas wie "Facebook in der ZX81-Cloud", "Röhren-Touchscreens" und "Gigaherz-ULA-Overclocking". Bin schon sehr gespannt.

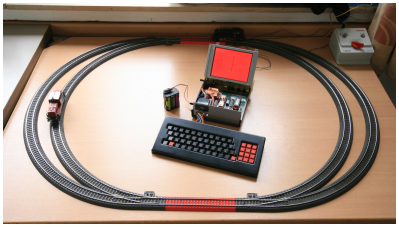
oliver@zx81.de

MAHLERTS 2011











...